A comparison of mesh simplification algorithms

P. Cignoni[*], C. Montani[*], R. Scopigno[§]

[*] Istituto di Elaborazione dell'Informazione- Consiglio Nazionale delle Ricerche

Via S. Maria, 46 - 56126 Pisa ITALY - Email: `[cignoni|montani]@iei.pi.cnr.it`

[§] Istituto CNUCE – Consiglio Nazionale delle Ricerche

Via S. Maria, 36 - 56126 Pisa, ITALY - Email: `r.scopigno@cnuce.cnr.it`

Send correspondence to:

Roberto Scopigno, CNUCE - CNR,

Via S. Maria 36, 56126 PISA (Italy)

Phone: +39 50 593304

FAX: +39 50 904052

EMail: r.scopigno@cnuce.cnr.it

25th September 1997

Abstract

In many applications the need for an accurate simplification of surface meshes is becoming more and more urgent. This need is not only due to rendering speed reasons, but also to allow fast transmission of 3D models in network-based applications. Many different approaches and algorithms for mesh simplification have been proposed in the last few years. We present a survey and a characterization of the fundamental methods. Moreover, the results of an empirical comparison of the simplification codes available in the public domain are discussed. Five implementations, chosen to give a wide spectrum of different topology-preserving methods, were run on a set of sample surfaces. We compared empirical computational complexities and the approximation accuracy of the resulting output meshes.

1 Introduction

Triangles are the most popular drawing primitive. They are managed by all graphics libraries and hardware subsystems, and triangular meshes are thus very common in computer graphics. Very complex models, with hundreds of thousands of faces, are easily produced by current CAD tools, automatic acquisition devices (e.g. range scanners), or by fitting isosurfaces out of volume datasets. Unfortunately, the increase in data complexity still surpasses improvements in hardware performance. However, a highly complex data representation is not always required, either because mesh complexity may depend on the characteristics of the acquisition or fitting process, or because a full size model is not required for the generation of each frame of an interactive visualization. This has led to substantial research into devising robust and efficient techniques for the controlled simplification of surface meshes. Reasons for the use of both simplification and multiresolution representations of surfaces have been reviewed by Heckbert and Garland [22]. Among other uses, simplification is the basis for the construction of *level of detail* (LOD) representations [12]; the LOD approach is now widely used due to the support given in de-facto standard graphics libraries [46, 45].

Research on surface simplification has been intense in the last few years. Many papers and different approaches have appeared, and potential users are being overwelmed by diffuse, unstable or even contradictory knowledge. Survey papers on surface simplification are still rare [11, 37, 23, 33].

This paper presents a brief introduction to surface mesh simplification methods, and proposes a new taxonomy. Its main objective is to analyze and compare the different approaches adopted to measure the *approximation error* introduced in the simplification process, rather than to review the proposed algorithms in depth. The scope of our analysis is limited to simplification methods for manifold or non-manifold surfaces immersed in 3D space. Many other approaches have been proposed for other types of data: to simplify images, height fields, range maps or triangulated terrains, which is in some way a simpler instance of our problem; to simplify volume datasets; to reduce the complexity of meshes fitted on volume dataset by adopting an adaptive fitting approach. Moreover, alternative approaches which reduce storing or rendering complexity by adopting compressed representations [9] or triangle strip representations [43] are also out of the scope of this paper.

In order to give the reader not only a theoretical evaluation, but also an "objective" comparison of some of the methods reviewed, we adopted an empirical approach. Six implementations of representative simplification approaches which preserve topology were tested on three meshes, which are instances of different data sources/types. The simplification codes were evaluated and compared empirically by taking into account the processing resources consumed and the quality of the results produced. The results are reported in Section 4.

2 Simplification Approaches

Substantial results have been reported in the last few years on surface simplification. The data domain of the solutions proposed generally covers all types of triangular meshes (e.g. laser range data, terrains, synthetic surfaces).

Different error criteria have been devised to measure the fitness of the approximated surfaces; any level of reduction can be obtained with most of the approaches listed below, on the condition that a sufficiently coarse approximation threshold is set.

The following are some of the existing methods:

- *coplanar facets merging*:

  coplanar or nearly coplanar facets are searched for in the mesh, merged into larger polygons, and then retriangulated into fewer facets than those originally required [16, 28, 24]; face merging is driven by a co-planarity test.

  The *superfaces* method [27] extends this approach by providing bounded approximations and more robust re-triangulations of the merged faces;

- *controlled vertex/edge/face decimation*:

  these methods work by the iterative elimination of components (vertices, edges, triangles), chosen upon local geometric optimality criteria. All decimation methods are restricted to manifold surfaces, and generally preserve mesh topology.

  - the original *mesh decimation* approach [40] applies multiple passes over the triangle mesh and progressively removes those vertices that pass a distance or angle criterion (based on local geometry and topology checks). The resulting holes are then patched using a local re-triangulation process. The candidate vertex selection criterion is based on a *local error* evaluation;
  - a decimation approach can also be adopted to simplify a mesh by iteratively collapsing edges into vertices [15, 36, 1], or by collapsing faces [17];
  - extensions to the decimation method which support *global error*[1] control have been proposed. In particular, the *simplification envelopes* method [8] supports bounded error control by forcing the simplified mesh to lie between two offset surfaces (but it works only on orientable manifold surfaces). Some other methods adopt heuristics for the evaluation of the *global* error introduced by each vertex removal and re-triangulation step, and work under an incremental simplification framework [42, 5, 3, 29, 36, 15];
  - controlled local modifications of re-triangulated patches, based on edge flipping, have been proposed to improve approximation accuracy in mesh decimation [3, 5];
  - the decimation approach has also been generalized to the simplification of 3D simplicial decompositions (tetrahedral sets) [35, 6, 18];

- *re–tiling*:

  new vertices are inserted at random on the original surface mesh, and then moved on the surface to be dis-

[1] *Global error* is defined here in opposition to *local error*, i.e. whether the approximation error introduced by the elimination of the current vertex is operated by comparing the resulting new mesh patch with the initial mesh M^0 or with the current, partially simplified mesh M^i; see Section 3 on error evaluation for a more precise definition.

placed over maximal curvature locations; the original vertices are then iteratively removed and a re–tiled mesh, built on the new vertices, is given in output [44];

- *energy function optimization*:

  the *mesh optimization* approach, originally proposed in [26], defines an *energy function* which measures the "quality" of each reduced mesh. Mesh reduction is iteratively obtained by performing legal moves on mesh edges: collapsing, swapping or splitting (in the latter case, a new vertex is inserted in the edge and two new edges connect it to the front-most vertices). Legal moves selection is driven by an optimization process of the energy function. At each step, the element whose elimination causes the lowest increase in the energy function is deleted.

  An enhanced version, *progressive meshes*, provides multiresolution management, mesh compression, selective refinements and enhanced computational efficiency [25, 32], and is based only on edge collapsing actions;

- *vertex clustering*:

  based on geometric proximity, this approach groups vertices into clusters, and for each cluster it computes a new representative vertex [38]. The method is efficient, but neither topology nor small-scale shape details are preserved. The visual and geometric quality of the meshes simplified with a clustering approach have been improved in [30].

  Another extension to the clustering approach was proposed to cope with the perceptual effects of degradation [34]; couples of edges internal to each cluster are merged if a test based on curvature and size is positively verified.

  A very recent approach [13] applies an efficient error evaluation, based on *quadric error matrices*, to a clustering approach which performs only vertex pair contractions (a vertex pair is eligible for contraction if either a connecting edge exists or the vertices satisfy a proximity criterion). The solution is characterized by its high computational efficiency and the capability to simplify disconnected or non-manifold meshes;

- *wavelet-based approaches*:

  the wavelet decomposition approach seems very promising for surface simplification (and, moreover, multiresolution comes for free). But a regular, hierarchical decomposition is required to support wavelet decomposition, and computational efficiency is not at the best. Wavelet approaches have been proposed to manage regularly gridded meshes [14, 21] or more generic meshes [10, 4].

  In particular, the *multiresolution analysis* approach is based on a three-phase process (re–meshing, re-sampling and wavelet parametrization) to build a multiresolution representation of the surface, from which any approximated representation can be extracted [10]. An extension to this approach manages the approximation of both geometry and surface color [4];

- *simplification via intermediate hierarchical representation*:

  an intermediate octree representation [2] may be adopted to automatically produce simplified representations, because the octree may be purged at various levels and then converted into a (simplified) boundary representation;

alternatively, an intermediate voxel-based hierarchical representation (built using signal-processing techniques for the controlled elimination of high-frequency details), together with adaptive surface fitting, was proposed in [19, 20].

2.1 A characterization

A classification of the simplification approaches can be based on the characterization of the input/output *data domain*, on the *simplification goal*, on the strategy adopted to drive/evaluate *mesh approximation*, and, last but not least, by considering if the simplification process follows an *incremental approach*.

I/O data domain

All of the methods briefly reviewed above accept in *input* simplicial meshes, but only a few of them can manage non–manifold meshes (e.g. vertex clustering and intermediate hierarchical representation).
Most of them return in *output* manifold simplicial meshes (e.g. decimation, energy function optimization, re-tiling), while others may produce not 2-manifold geometries (e.g. vertex clustering may produce dangling faces, edges, or points). Moreover, taking into account the output produced, simplification methods may be characterized by highlighting two main orthogonal classes [39]:

- approaches which *preserve mesh topology* (e.g. mesh decimation, mesh optimization), and those which don't (e.g. vertex clustering, intermediate hierarchical representation);
- approaches based on *vertex subset selection* (e.g. coplanar facets merging, mesh decimation) or *re-sampling* (e.g. mesh optimization, re–tiling, multiresolution analysis, intermediate hierarchical representation).

The importance of *preserving mesh topology* depends directly on the application domain. It is not mandatory if the goal is to speedup rendering, at least for the lower resolution representation of a LOD model (and topology simplification is generally a must to produce highly simplified models out of topology–rich objects). On the other hand, topology has to be preserved if the simplification goal is to produce a representation which might be nearly indistinguishable from the original, or which preserves shape features (e.g. medical application requirements).
The choice between using a *subset* of the original vertices or using *re-sampled* vertices again depends on the application and this usually affects approximation precision. There are, in fact, many applications where re-sampling is not allowed or feasible, e.g. in the case of datasets where the sampling of a scalar/vectorial field is associated with the mesh vertices and we cannot safely recompute the field value in the re-sampled locations. On the other hand, better approximation accuracy is obtained when vertices are resampled, e.g. by moving the vertices on the lines of maximal curvature.

Simplification goal

Another possible classification may be based on the simplification goal [8]:

- *Min-#*: when, given some error bound ε, the objective is to build the approximated mesh of a minimal size which satisfies precision ε (size is generally measured in number of vertices);
- *Min-*$varepsilon$: when, given an expected size for the approximated mesh, the objective is to minimize the error, or difference, between the original and the resulting mesh.

Heuristics for the evaluation of mesh approximation

In this framework, important aspects are:

- if a *local* or a *global* approach is adopted; in the first case, mesh modifications are operated upon a local optimization criterion (e.g. simplification envelopes and other decimation approaches); in the second one, a global optimization process is applied to the whole mesh (e.g. energy optimization approaches, re-tiling, multiresolution decimation, and multiresolution analysis);
- the measurability and preservation under tight bounds of the *approximation error* introduced (e.g. simplification envelopes and some other decimation approaches);
- the preservation of *geometric* or *attribute discontinuities*, for example feature edges and color or pictorial information (e.g. mesh decimation, progressive meshes).

Incrementality of simplification

An approach is *incremental* if simplification proceeds through a sequence of local mesh updates which, at each step, reduce the mesh size and monotonically decrease the approximation precision (e.g. mesh decimation, progressive meshes, clustering via quadric matrices). Incrementality is a key aspect for supporting easily the production of a *multiresolution output*. Although multiresolution output is explicitly produced by only a few approaches (multiresolution analysis, progressive meshes, multiresolution decimation), it is possible to introduce it with simple extensions in most other incremental methods.

An attempt to give an overall characterization of different simplification algorithms is presented in Table 1. Columns 2-4 characterize the strategy adopted to manage mesh approximation: the goal which drives the simplification process (*Min-#*, *Min-*ε, or *both*); if the approach simplifies the mesh *incrementally*; and the *topologic entity* taken into account during simplification (*v*: vertices, *e*: edges, *f*: faces, *v-pair*: vertex pairs).

Columns 5-8 characterize the approximation error management policy. The ε_{loc} column is marked if for each simplification step the L_∞ *error* introduced is evaluated by a *local* shape comparison between the modified patch and the corresponding patch just before the current step; ε_{glob} is marked if a *global* shape comparison with the starting input mesh is performed (using an L_∞ norm again); or column *other* is marked if another policy is adopted, e.g. energy function optimization (which adopts an L_2 norm), or clustering evaluation. Moreover, we mark those methods which guarantee *bounded* accuracy on the whole mesh in column 8.

The multiresolution column highlights those methods which produce in a single run a real *multiresolution output*, encoded with an *ad hoc* representation.

Preserving mesh characteristics is evaluated in columns 10-12 in terms of: the preservation of global mesh topology (column *meshTop*); possible relocation of the vertices of the simplified mesh (column *vertLoc*), with value *unchanged* or *relocated*; the preservation of feature/solid edges or angles (column *featEdg*).

The estimated simplification *speed* reported in column 13 (measured in *KTr/sec*, i.e. thousands of triangles simplified for CPU second) has been taken directly from the results presented in the original papers. Since these results were obtained on different meshes and on different machines, they only give a rough and imprecise estimate of the efficiency of the algorithms, but are presented in the table to give the order of magnitude of simplification times (and also to emphasize proposals which did not report any evaluation of running times, indicated in the table with the "*??*" tag).

Finally, column 14 lists whether the code is available in the *public domain*, as part of a *commercial product*, or is *not available* at all.

The capability to preserve discontinuities of vertices/faces attributes is a very important feature, but it has been not included in Table 1. This is because although this feature is only supported by a few proposals [34, 25, 4, 41], most other approaches could simply be extended to support it (e.g. by providing an enhanced classification of vertices for the vertex decimation approach).

An overall comparison of simplification approaches is not easy, because simplification accuracy largely depends on the geometric and topological structure of the input mesh and on the required results. For example, the presence of sharp edges or solid angles is managed better by *coplanar facet merging* and *decimation* approach, while on smooth surfaces *mesh optimization* and *re–tiling* give better results. On the other hand, the good results in the precision and conciseness of the output mesh given by *mesh optimization* and *re–tiling* techniques are counterbalanced by substantial processing times. Although no time comparisons between different methods have been reported in the literature, an informed guess would be that the *mesh decimation* and the recent *quadric matrices clustering* approaches are the most efficient methods.

	Method Char.			Approximation Error				Multi-res	Preserve Mesh Charact.			Speed	Availability
	optim. goal	incre- mental	top. entity	ε_{loc}	ε_{glob}	other crit.	bound.	output	mesh topol.	vert. locat.	feature edges	KTr/sec.	
Coplanar Facet Merging Approaches													
Geom. Opt. [24]	Min-#		f			x	no		yes	unch.	yes	0.7-2.7	not avail.
Superfaces [27]	Min-#		f		x		yes		yes	unch.	yes	0.3-0.8	not avail.
Decimation Approaches													
Mesh Decimat. [40]	Min-ε	x	v	x			no		yes	unch.	yes	2.-2.5	publ.dom.
Triangle Remov. [17]	Min-#	x	f			x	no		yes	unch.	yes	??	not avail.
Hierarch.Triang.[42]	Min-#	x	v		x		yes		yes	unch.	yes	??	comm.prod.
Err.Bound.TMR [3]	Min-#	x	v		x		yes		yes	unch.	yes	??	not avail.
Multires.Dec. [5]	both	x	v		x		yes	x	yes	unch.	yes	0.15-0.2	publ.dom.
Hausd.Distance [29]	Min-#	x	v		x		yes		yes	unch.	yes	??	not avail.
Simpl.Envelop. [8]	Min-#		v		x		yes		yes	unch.	yes	0.07-0.09	publ.dom.
Toler.Volumes [15]	Min-#	x	e		x		yes		yes	reloc.	yes	0.08-0.1	not avail.
Full-range Appr. [36]	both	x	e		x		yes		no	unch.	yes	??	not avail.
Mesh Simpl. [1]	Min-#		e	x			no		yes	reloc.	yes	0.2	not avail.
Energy Optimization Approaches													
Mesh Opt. [26]	Min-#	x	v+e			x	no		yes	reloc.	≈	0.008	publ.dom.
Progr.Meshes [25]	Min-#	x	e			x	no	x	yes	reloc.	yes	0.04	not avail.
Clustering Approaches													
Vert.Clust. [38]	Min-#		v+e+f			x	yes		no	reloc.	no	??	comm.prod.
Percept.Clust. [34]	Min-#		e			x	yes		no	unch.	yes	0.1-0.05	not avail.
Quadric Err.Matr. [13]	both	x	v-pairs		x		no	x	no	reloc.	no	4.5	not avail.
Intermediate Hierarchical Representation Approaches													
Octree-based [2]	Min-#		–			x	yes		no	reloc.	no	??	not avail.
Voxel-based [20]	Min-#		–			x	yes		no	reloc.	no	??	not avail.
Other Approaches													
Re-Tiling [44]	Min-ε		v			x	no		yes	reloc.	no	??	not avail.
Multires.Anal. [10]	Min-#		–		x		yes	x	yes	reloc.	no	0.04	not avail.

Table 1: Characterization of different simplification algorithms.

3 Simplification Error Evaluation

This section presents the various techniques for evaluating and bounding the approximation error introduced in the mesh simplification process. A keen control of the approximation accuracy is critical, for example to prevent highly perceivable discrepancies between different LODs or to produce simplified and hopefully nearly indistinguishable representations of the highly complex meshes acquired via range scanners.
A definition of the *approximation error* between two meshes, based on the L_∞ norm[2], may be given as follows [8, 29, 5].

Definition 1 *Given two piecewise linear objects M_i and M_j, M_i and M_j are ε-*approximations *of each other* **iff** *every point on M_i is within a distance ε of some point of M_j and every point on M_j is within a distance ε of some point of M_i.*

The approximation error is managed in many different manners by the various simplification approaches. A characterization may be based on the policy chosen to bound the approximation error:

1. approaches which support **locally bounded** errors, i.e. the approximation accuracy is known around each surface entity (e.g. most of the mesh decimation methods [40, 42, 3, 5, 29, 36]);

2. approaches which only support **globally bounded** approximation errors, i.e. the accuracy is known only for the entire simplified mesh (e.g. the simplification envelopes [8], superfaces [27] and clustering approaches [38], methods based on the conversion into an intermediate hierarchical representation [2, 20]);

3. approaches which control accuracy with **other criteria**, which are not compatible with Definition 1; usually, curvature is taken into account to define a global bound on the surface (e.g. geometric optimization [24], triangle removal decimation [17], mesh simplification [1]);

4. approaches which **do not evaluate** the approximation accuracy (and are generally driven by the user-required simplification rate).
   (e.g. re-tiling [44]; methods based on the evaluation of an energy function [26, 25] may be included in this class, if we do not consider the energy function as a valid measure of the approximation error, as defined in Def.1);

Methods of class (1), *locally bounded*, are generally iterative methods based on a sequence of local updates to the mesh geometry/topology. For each iteration, the current mesh M_i is slightly modified to produce mesh M_{i+1}. Modifications are limited to the two patches T_i and T_i', which (a) surround the decimated/collapsed/flipped element e_i, and (b) share the border. In this case, different methods have been proposed to evaluate, at each step, the variation in the local error bounds:

- **local evaluation**; we evaluate only the approximation introduced by replacing patch T_i with T_i' :

[2] A L_2 norm has also been adopted in some simplification approaches [25, 26].

  - using a fast *approximated* approach, e.g. measuring the distance of the decimated vertex from the average plane to patch T_i [40] (Figure 1.a);
  - using a *precise* approach, which is based on the observation that the mutual projection of the two patches T_i and T_i' subdivides the associated hole into pieces within which both geometries vary linearly [3]. Thus, it suffices to compute errors at the intersections of the projected edges and at the internal vertex; the maximal of these errors gives an upper bound of the local error (see Figure 1.b);

- **accumulation of local errors**; at each step, we assign to each new face/vertex in T_i' the sum of (1) the current *local evaluation* of the approximation error and (2) the maximal error associated with the faces/vertices in T_i [3, 5];

- **global evaluation**; we directly estimate the approximation introduced by representing with the simplified patch T_i' the corresponding section of the initial mesh M_0. Many approaches have been proposed; they can be divided into two classes:

  - *approximate* approaches;
    - if all removed vertices are stored with the current simplified face f onto which they "project" (e.g. which are at the shortest distance from f), a global error approximation is the maximal distance from these vertices to f [42, 5] (see Figure 1.c). This criterion is efficient, but returns an underestimation because the L_∞ *mesh–to–mesh* distance might not be located on one of the initial mesh vertices; moreover, the criterion is very imprecise in the first simplification steps when few, or even none, of the removed vertices are associated with each simplified face;
    - another approach has been proposed for methods based on edge collapsing [36]. At initialization time, for each vertex we store the list of planes where the faces incident in the vertex lie. Planes lists have to be maintained and updated during simplification (after each edge collapse action, the two lists associated with the extremes of the collapsed edge are merged). The error is then evaluated at each step as the maximum distance between the new collapsed vertex position and all the planes in the vertex list. The evaluation of distances is much more efficient in [13], where *quadric error matrices* are used. This approach returns an upper-bound for the approximation error, but in some cases the bound might be over-estimated with respect to the actual error.
  - *precise* approaches; these compute the Hausdorff distance between the original and the simplified mesh (*mesh–to–mesh* distance). This can be done either: by maintaining trace, during simplification, of all the original faces that map to simplified faces, and then computing *face–to–mesh* distances by performing, if needed, an adaptive decomposition of faces [29]; or, by extending Bajaj et al.'s local method [3], to compare patch T_i' with the corresponding section of the initial mesh M_0. In both cases, computing times are now proportional to the complexity of meshes T_i' with respect to the corresponding M_0 subsection (i.e. the more we proceed with simplification, the higher the complexity of each section of M_0 which

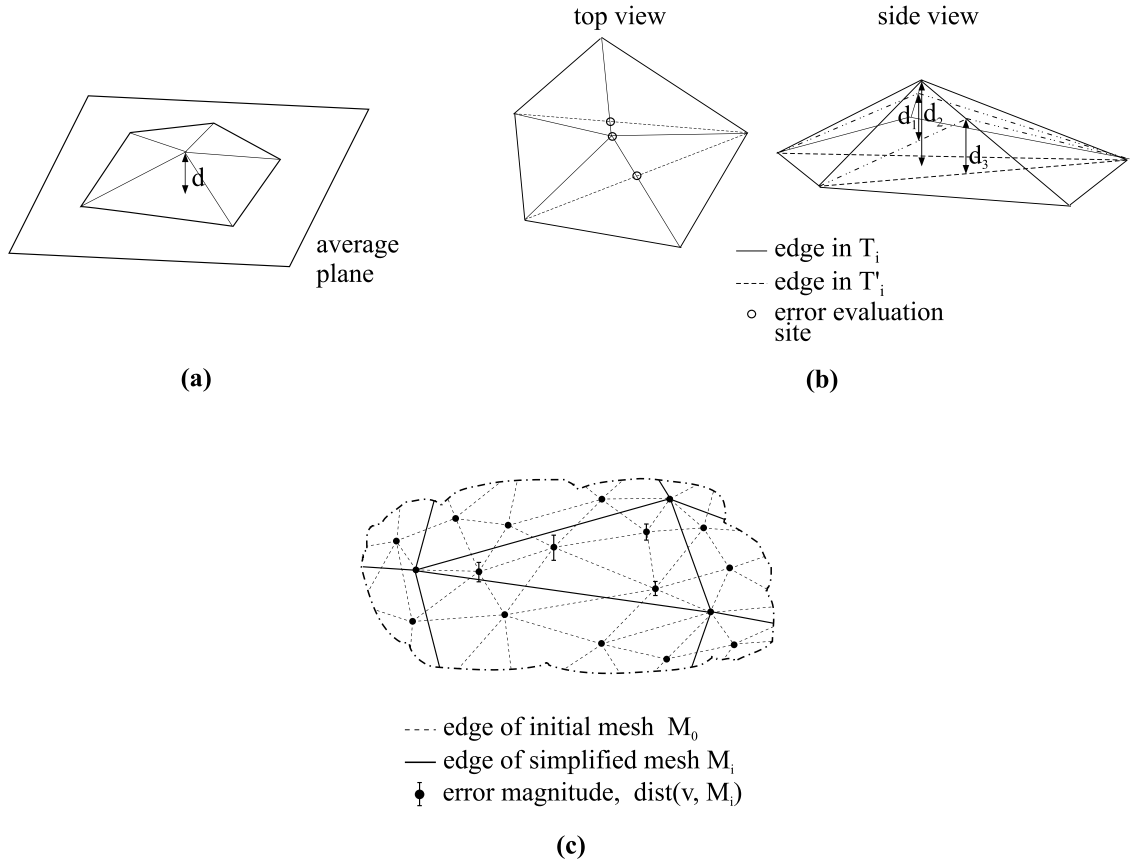


Figure 1: Various methods to evaluate approximation error: (a) approximated local , (b) precise local, (c) approximated global.

is associated with the current patch, and the higher the processing cost for computing edge intersection and distances);

In the case of incremental methods, the accuracy of the simplified mesh may be improved by adopting a greedy approach based on *edge flipping* [5, 3], operated over each new patches. But in order to effectively improve the approximation accuracy, edge flipping has to be driven by the evaluation of the *global error* variation caused by each potential flipping, and not only by a simpler equiangularity test.

Most of the methods reviewed offer no immediate provision to accurately control the perceptual effect of the degradation, because in most cases the approximation introduced into simplification has no immediate interpretation in terms of *visual degradation* [34]. Perceivable visual degradation may be caused either while visualizing a single simplified representation (e.g. in the case of eccessively approximated representation, loss of topology features, fuzziness of the simplified surface, etc.), or while changing the level of detail, the so called *inter-frame flickering* which is common if meshes in a LOD representation present large visual differences.

Defining a measure for visual degradation is no easy task and is being hotly debated. Driving simplification by preserving curvature and sharp edges gives good control on the appearance of the shape, one reason being that most renderers draw elementary components by shading colors according to surface normals [34]. But taking into account the shape is not enough: pictorial information (color or texture) is an important factor in perception, and therefore color discontinuities have to be managed carefully [34, 25, 4, 41].

3.1 The Metro tool

Due to the many approaches adopted to evaluate simplified mesh accuracy, a uniform and general tool for the evaluation of approximation precision is needed to compare the results of different simplification methods. For this reason we developed an ad-hoc tool, called *Metro*.

The first release of *Metro* was described in [7]. The current version of the tool, rel. 2.0, has been completely re-designed in order to increase precision in the evaluation of mesh accuracy, improve efficiency (it is now nearly ten times faster), and reduce memory allocation.

Metro numerically compares two triangle meshes S_1 and S_2, which describe the same surface at different levels of detail. It requires no knowledge of the simplification approach adopted to build the reduced mesh. *Metro* evaluates the difference between the two meshes on the basis of the *approximation error* previously stated in Definition 1. It adopts an approximate approach based on surface sampling and the computation of *point–to–surface* distances. The surface of the first mesh (hereafter *pivot* mesh) is sampled, and for each elementary surface parcel we compute a point–to–surface distance with the not–pivot mesh. *Point–to–surface* distances are computed efficiently by using a bucketed data structure for the representation of the non-pivot mesh.
The idea is therefore to adopt an integration process over the surface; the sampling resolution characterizes the precision of this integration (users may select the *sampling step size*). Sampling on the surface is achieved by adopting a classical incremental scan conversion approach or a Montecarlo sampling approach.
At the end of the sampling process, we switch the pivot and not–pivot mesh and execute sampling again, to get a symmetric evaluation of the error (but we observed that when a sufficiently thin sampling step is adopted, for example 0.01% of the bounding box diagonal, nearly equal values were obtained whatever mesh was chosen as the pivot).

Metro returns both *numerical* and *visual* evaluations of surface meshes "likeness" (Figure 5 shows a snapshot of its GUI). Most of the numerical results (mesh surface area, feature edges total length, mean and maximum distances between meshes, mesh volume) are reported in the tables in Section 4 . Error is also *visualized* by rendering the higher resolution mesh with a color for each vertex which is proportional to the error. A histogram reporting the error distribution is also visualized.

The error evaluated by *Metro* may be affected by finite numerical precision, although double precision is adopted in numerical computations. An "ad hoc" management has been provided for a number of dangerous cases, such as nearly coincident vertices, facets with small areas, and very elongated triangles.

4 Empirical Evaluation of Simplification Codes

To test some representative (and available) simplification codes, listed below, we chose three datasets, which represent three main classes of data:

- meshes acquired with an automatic range scanner — **bunny** is a model of a plastic rabbit, scanned at Stanford University. Mesh size: 34,834 vertices, 69,451 triangles. Available at `http://www-graphics.stanford.edu/data/`
- meshes produced with a standard CAD system — **fandisk** is a valid representative of CAD models and it is characterized by sharp edges and sophisticated surface curvature; it is enclosed in the Mesh Optimization distribution package. Mesh size: 6,475 vertices, 12,946 triangles. Available at `http://research.microsoft.com/research/graphics/hoppe/`
- meshes extracted from volume datasets — **femur** is an isosurface from a CT scan of a human femur, courtesy of the Istituto Ortopedico Rizzoli (IOR) [3]. Mesh size: 76,794 vertices, 153,322 triangles. Available at `http://miles.cnuce.cnr.it/cg/homepage.html`

The simplification codes are compared in terms of the size of the meshes produced, the approximation quality, and the running times. The simplified meshes were compared by using the *Metro* tool (see Subsection 3.1).

Simplification Codes

The following simplification codes were tested:

1. **Mesh Decimation** [40]; code provided in the Visualization Toolkit 1.3 (VTK) by Bill Lorensen, Ken Martin and William Schroeder (`http://www.cs.rpi.edu/~martink/`);
2. **Simplification Envelopes** rel 1.2 [8]; code developed at the Department of Computer Science of the University of North Carolina, code courtesy of Jonathan Cohen et al. (`http://www.cs.unc.edu/~cohenj`);
3. **Multiresolution Decimation** [5]; code Jade rel. 2.0[4], implemented by the Visual Computer Group of CNUCE/IEI-C.N.R. (`http://miles.cnuce.cnr.it/cg/jade.html`);
4. **Mesh Optimization** [26]; code developed by Hugues Hoppe et al., Univ. of Washington (`http://research.microsoft.com/research/graphics/hoppe/`);
5. **Progressive Meshes** [25]; code developed by Hugues Hoppe, Microsoft inc. (`http://research.microsoft.com/research/graphics/hoppe/`)

[3]IOR is an orthopaedic hospital located in Bologna (Italy).

[4]Jade rel. 2.0 has been slightly improved in terms of approximation error management with respect to the description and results reported in [5].

6. **Quadric Error Metrics Simplification [13]**; code developed by M.Garland and P.Heckbert, Carnegie Mellon Univ. (`http://www.cs.cmu.edu/afs/cs/user/garland/www/home.html`)

We initially also planned to test a representative of commercial tools, i.e. the Polygon Reduction Editor available under *SGI Cosmo Worlds*[5]. This simplifier seems to be based on the *clustering* approach. It presents a simple GUI which allows the user to set threshold values to delete points by curvature, edges by length and triangles by area. The simplification process is driven by a *trial and error* approach. The quality of the mesh produced therefore depends on the skill (and the luck) of the user, and results of a quality comparable to the simplified meshes produced using the codes above appear to be nearly impossible (Figure 7). The new mesh simplification module provided in the *SGI OpenGL Optimizer*[6] was not available at the time of this test.

Hardware used

Simplification codes 1, 2, 3 and 4 were run on an SGI Indigo2, R4400 200MHz CPU, 16 KB data cache, 16 KB instruction cache, 1MB secondary cache, 128 MB RAM.

The Mesh Optimization code is distributed only in executables for Digital workstations. We ran it on a Digital 3000/900, Alpha 275 MHz CPU, 128MB RAM; run times were then scaled back to SGI Indigo2 units (scaling was done on the basis of an ad hoc comparative benchmark run on both Digital and SGI ws).

The Progressive Meshes code is not available in the public domain, and tests were done courtesy of Hugues Hoppe on a SGI Indigo2 Extreme, R4400 150MHz CPU, 128MB RAM. Quadric Error Metric simplificator is also not available in public domain, and our tests were done courtesy of Michael Garland on a SGI Indigo2, R4400 250MHz CPU, 128MB RAM.

4.1 Numerical Evaluation

Tables 2, 3, 4 and 5 present the numerical results obtained with our tests.

In particular, Mesh Decimation and Simplification Envelopes were not able to reach a high simplification rate on the Femur dataset (Table 4 and 5).

Volumes were evaluated on the Fandisk meshes only, because other meshes are open (and therefore the volume is not defined). The *Edge length* was evaluated on all meshes; we set the dihedral angle threshold to 30 degrees (i.e., each mesh edge with a dihedral angle lower than 30 degrees is classified as a feature edge and its length is summed to the current Edge length).

In the case of the Progressive Meshes code, we report only the overall time needed to produce the full simplification of the mesh. This is because the Progressive Meshes code first simplifies the dataset and then builds a multiresolution output file (called PM file), at the speed of 0.03-0.05 KTr/sec. Simplification times are shorter than those of the Mesh Optimization code, but they are still high due to the complex error evaluation and simplification

[5] `http://www.sgi.com/Products/cosmo/worlds/CosmoWorlds_UG/Reference/polyed.htm`

[6] `http://www.sgi.com/Technology/OpenGL/optimizer/presentation.html`

criteria adopted. Different and simplified error metrics could be used in Progressive Meshes, but that would probably imply a degradation in simplification accuracy.

Once an off-line simplification has been run, Progressive Meshes can reconstruct any level of approximation from the PM file, with a reconstruction rate of 83 KTr/sec and an offline simplification rate of 104KTr/sec on an R4400 Indigo2 Extreme.

The Multiresolution Decimation code also allows individual approximations to be reconstructed out of the multiresolution history file, with performances similar to those of the Progressive Meshes code ($\approx$ 100 KTr/sec).

The results relative to the evaluation of the approximation error are also summarized in the graphs in Figure 2-4. In the graphs on the left we plot the *maximal error* (E_{max}) evaluated by Metro on simplified meshes of different sizes. The *average error* (E_{avg}) is reported in the graphs on the right. For all graphs, the simplified mesh size is mapped on the X axis, and the error is mapped on the Y axis.

As we expected, the best results in terms of *average error* are often given by the Progressive meshes and mesh Optimization codes (which are based on an L_2 metric over the object surface, meaning that they try to minimize the root mean square error). On the other hand, methods based on the L_∞ metric produce better results when we consider the *maximal error*.

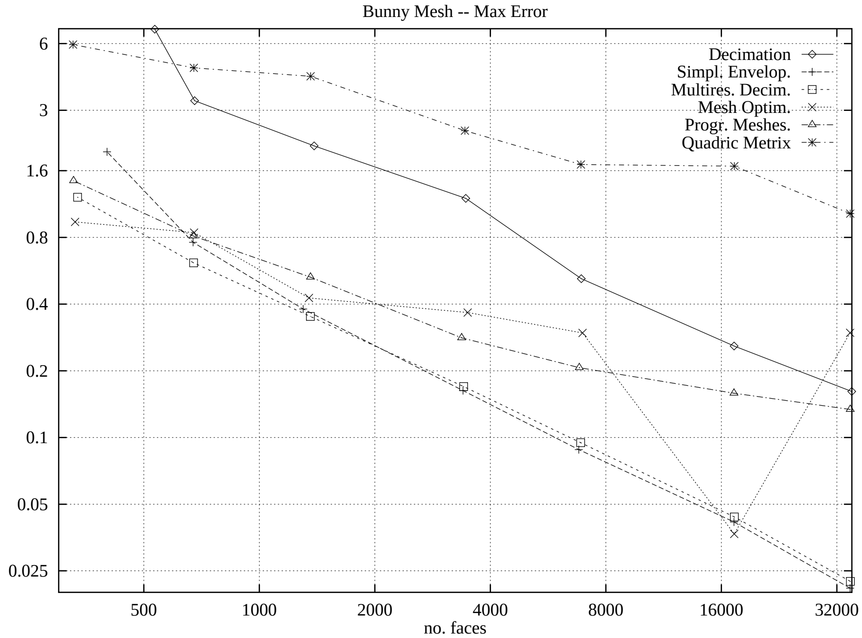
It is noticeable that Simplification Envelopes and Multiresolution Decimation produce the best results when high accuracy is needed (i.e. for reduction factors not higher than 75%).

Moreover, both the speed of the Quadric Error Metric and its precision are really impressive. In the case of the Fandisk mesh, precision of this approach is comparable to the best. This is not the case with meshes with open boundaries (Femur and Bunny), where large errors are generated in correspondence of the mesh boundary by the current implementation of this method. But authors comment that error on open boundaries may be bounded by inserting perpendicular planes to each boundary edge, and assigning to such edge a large penalty factor [13].

Mesh Decimation and *Simplification Envelopes* showed a particular behaviour on the Femur dataset: simplification rates higher than 95% and 99%, respectively, were not possible. This might depend on the policy adopted to remove vertices. Both approaches remove vertices in random order (both methods do not take into account the effective global error introduced by each decimation action, and do not sort candidate vertices in order of increasing approximation). Therefore if high removal percentages are requested, the first decimation steps modify the mesh so crudely that further decimation becomes not feasible. A partial solution to this problem can be to iterate multiple times these codes on the results of the previous simplification, reducing the decimation factor progressively at each step.

4.2 Visual Evaluation

Metro also enables the error magnitude to be plotted directly on the mesh, by setting the color of the vertices proportional to the evaluated error. For reasons of space, we only present error-mapped images of the Fandisk mesh (Figure 5 and 6); they refer to the comparison of the original mesh with a 25:1 reduced mesh ($\approx$ 250 faces). Images



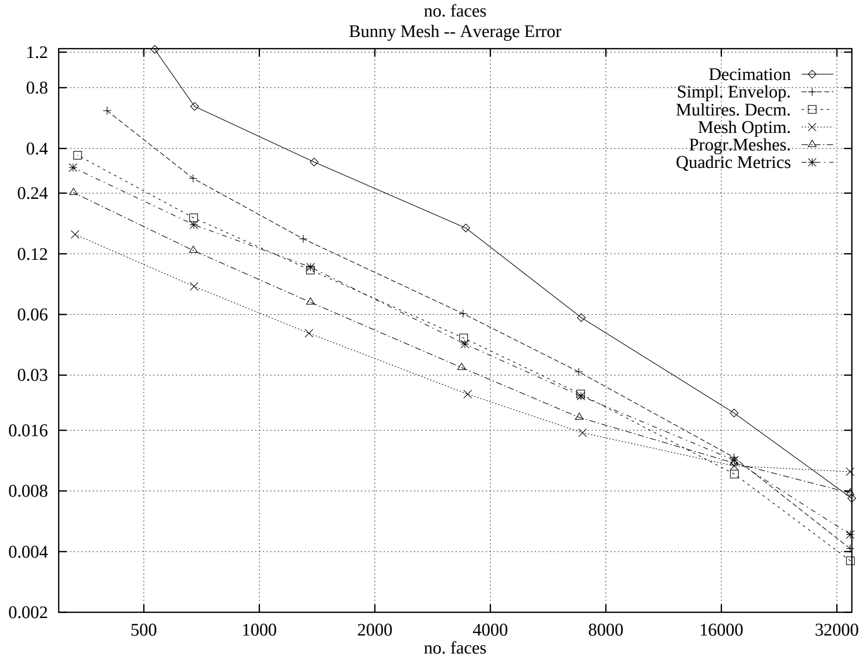
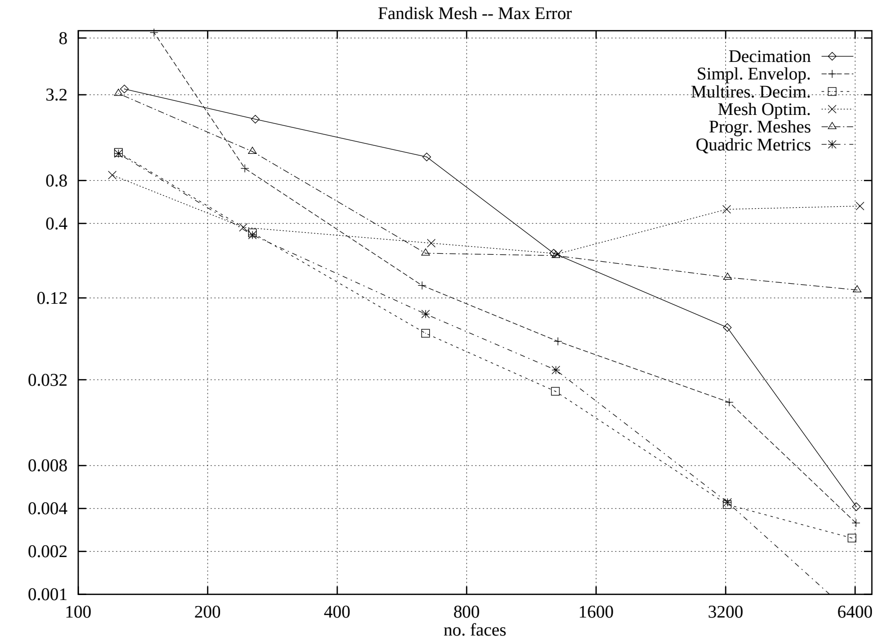


Figure 2: The graphs show the performance of the various simplification codes on the Bunny mesh.



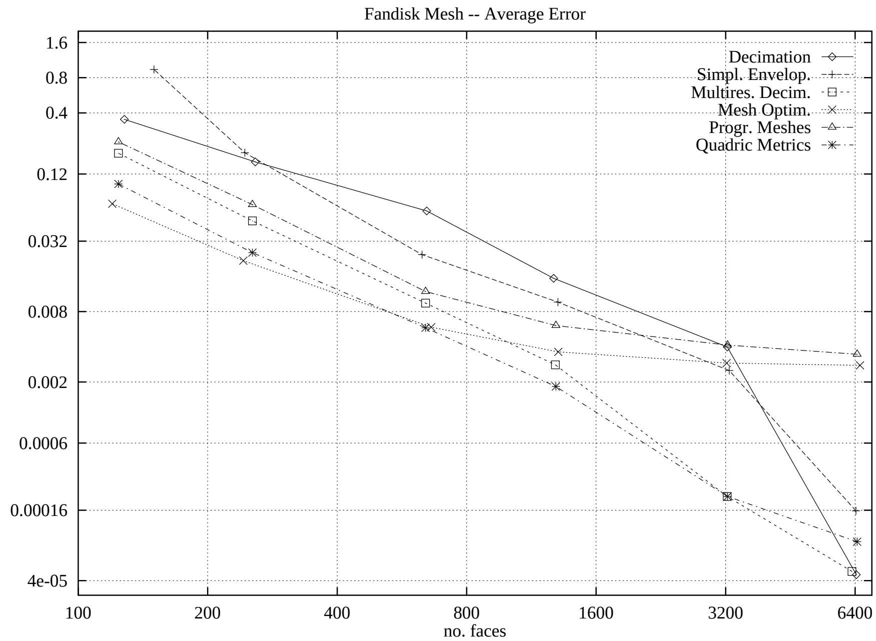
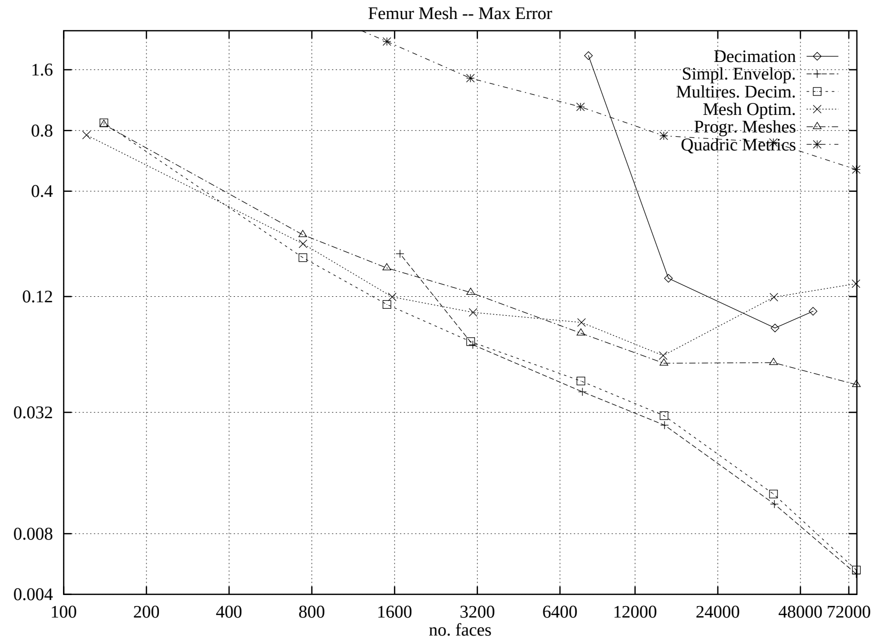


Figure 3: The graphs show the performance of the various simplification codes on the Fandisk mesh.



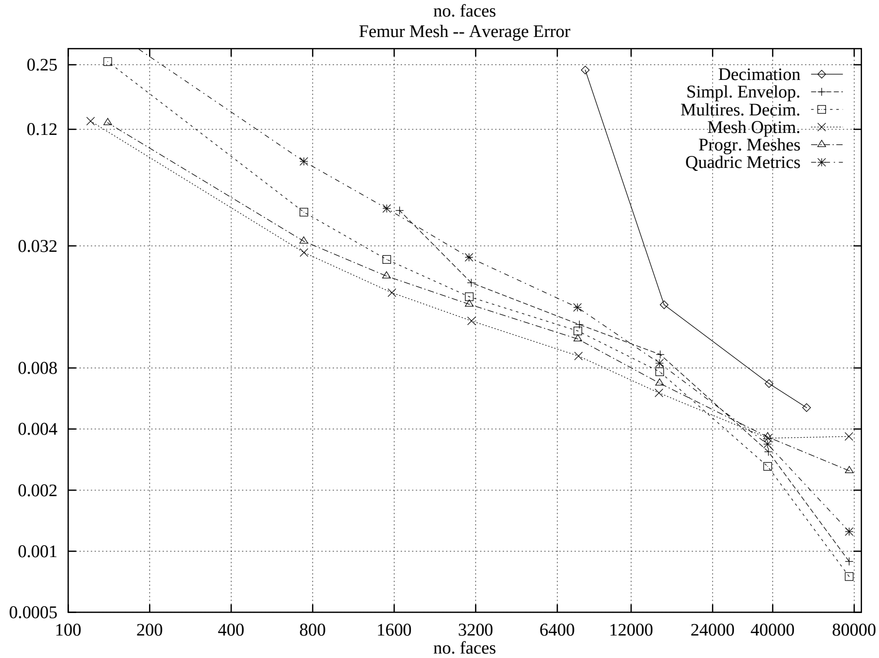


Figure 4: The graphs show the performance of the various simplification codes on the Femur mesh.

of the other meshes will be included in a longer version of this paper.

Bunny (34,834 vertices, 69,451 triangles, bounding box 15.6x15.4x12.1)							
Edge Length 189.099, Area 571.288 (Volume is not defined: the surface is open)							
Mesh Decimation							
N_{Vert}	N_{Triang}	E_{max}	E_{avg}	*Time*	*EdgeLength*	*Area*	*Mem. Kb*
17,566(50%)	34,965	0.1614	0.00735	43.97	194.189	571.457	14,600
8,705 (25%)	17,267	0.2586	0.01947	37.55	262.279	570.801	14,600
3,505 (10%)	6,900	0.5212	0.05791	46.22	382.084	568.489	14,600
1,775 (5%)	3,451	1.2000	0.16120	25.68	554.331	563.537	14,600
701 (2%)	1,389	2.0721	0.34230	32.22	572.173	555.250	14,600
344 (1%)	678	3.3117	0.64630	26.96	481.120	542.551	14,600
272 (0.5%)	534	6.9596	1.23980	30.08	507.002	520.262	14,600
Simplification Envelopes							
17,418(50%)	34,643	0.02089	0.00414	932.17	322.801	571.316	62,800
8,709 (25%)	17,252	0.04154	0.01174	942.93	299.725	570.910	63,300
3,472 (10%)	6,801	0.08813	0.03117	944.66	396.175	570.962	63,800
1,763 (5%)	3,395	0.16290	0.06066	964.03	478.302	569.133	64,600
678 (2%)	1,301	0.38020	0.14230	1003.01	468.944	562.575	65,200
355 (1%)	672	0.75990	0.28380	988.44	491.263	556.315	65,800
217 (0.5%)	401	1.94720	0.61530	1584.64	496.031	547.501	66,400
Multiresolution Decimation (Jade 2.0)							
17,417(50%)	34,679	0.0224	0.0036	208.95	207.260	571.591	9,300
8,708 (25%)	17,289	0.0438	0.0097	371.03	218.088	571.537	9,900
3,483 (10%)	6,874	0.0948	0.0242	388.68	272.793	571.145	10,200
1,741 (5%)	3,408	0.1697	0.0459	438.62	361.983	570.392	10,350
696 (2%)	1,358	0.3519	0.0997	475.73	416.856	567.973	10,400
348 (1%)	674	0.6141	0.1810	502.09	406.549	564.115	10,600
174 (0.5%)	336	1.2147	0.3697	529.65	426.135	557.285	10,800
Mesh Optimization							
17,410(50%)	34,643	0.29680	0.00996	7,100	1346.74	579.836	44,300
8,699 (25%)	17,279	0.03668	0.01064	7,400	488.486	576.048	44,300
3,501 (10%)	6,956	0.29640	0.01554	7,600	358.650	577.560	44,300
1,758 (5%)	3,491	0.36660	0.02415	7,400	364.890	581.119	44,300
686 (2%)	1,347	0.42620	0.04846	8,000	392.326	585.416	44,300
349 (1%)	676	0.84070	0.08265	8,500	404.878	588.034	44,300
173 (0.5%)	331	0.93970	0.14970	9,000	429.610	596.371	44,300
Progressive Meshes							
17,417(50%)	34,667	0.1339	0.00781	–	255.977	571.704	N.A.
8,708 (25%)	17,252	0.1585	0.01100	–	283.746	572.252	//
3,483 (10%)	6,821	0.2065	0.01851	–	327.247	573.149	//
1,741 (5%)	3,367	0.2817	0.03273	–	391.144	573.949	//
696 (2%)	1,359	0.5290	0.06897	–	451.311	574.997	//
348 (1%)	673	0.8122	0.12460	–	442.708	575.426	//
171 (0.5%)	328	1.4409	0.24140	1,450	439.415	573.827	//
Quadric Error Metrics							
17,417(50%)	34,666	1.0240	0.00486	20.50	203.919	568.542	N.A.
8,708 (25%)	17,293	1.6769	0.01135	24.70	223.754	567.082	//
3,483 (10%)	6,888	1.7070	0.02363	26.42	259.867	565.728	//
1,741 (5%)	3,434	2.4256	0.04279	26.93	360.545	563.636	//
696 (2%)	1,360	4.2700	0.10310	27.36	429.640	558.917	//
348 (1%)	675	4.6614	0.16720	27.41	449.767	556.754	//
171 (0.5%)	327	5.9382	0.32140	27.40	432.310	545.680	//

Table 2: Comparison of various simplification algorithms on the Bunny mesh (errors are measured as percentages of the datasets bounding box diagonal; times are in seconds).

Fandisk (36,475 vertices, 12,946 triangles, bounding box 4.8x5.6x2.7)								
Edge Length 69.9526, Area 60.6691, Volume 20.2433								
Mesh Decimation								
N_{Vert}	N_{Triang}	E_{max}	E_{avg}	*Time*	*EdgeLength*	*Area*	*Vol.*	*Mem. Kb*
3,224 (50%)	6,444	0.00412	4.502e-05	4.50	69.9526	60.6691	20.2432	11,400
1,616 (25%)	3,228	0.07452	0.00398	7.38	69.9500	60.6667	20.2557	11,400
639 (10%)	1,274	0.24710	0.01539	10.00	73.8872	60.6585	20.2650	11,400
325 (5%)	646	1.17080	0.05800	7.62	79.8004	60.6400	20.3816	11,400
131 (2%)	258	2.15660	0.15230	5.97	84.1135	60.3522	20.6055	11,400
66 (1%)	128	3.51280	0.35270	5.71	86.5037	59.9931	21.0034	11,400
Simplification Envelopes								
3,216 (50%)	6,428	0.00317	0.000158	203.49	87.0931	60.6691	20.2432	15,600
1,633 (25%)	3,262	0.02227	0.002505	228.14	78.6976	60.6732	20.2466	15,400
654 (10%)	1,304	0.05958	0.009646	185.22	70.3327	60.6747	20.2731	15,400
317 (5%)	630	0.14680	0.024520	170.93	72.7624	60.6733	20.3143	15,200
129 (2%)	244	0.97260	0.182700	159.42	113.677	60.0575	20.7452	15,200
77 (1%)	150	8.73700	0.940600	476.46	108.785	63.7828	24.7631	17,200
Multiresolution Decimation (Jade 2.0)								
3,149 (50%)	6,294	0.00248	4.847e-05	28.26	69.9526	60.6691	20.2433	4,000
1,615 (25%)	3,226	0.00427	0.00021	37.72	69.9526	60.6694	20.2433	4,000
645 (10%)	1,286	0.02657	0.00280	52.26	70.5093	60.6716	20.2497	4,000
323 (5%)	642	0.06778	0.00944	64.79	70.2319	60.6814	20.2686	4,000
129 (2%)	254	0.34370	0.04767	70.55	80.4990	60.6798	20.3724	4,000
64 (1%)	124	1.25980	0.18060	75.78	75.9090	60.4357	20.7143	4,000
Mesh Optimization								
3,287 (50%)	6,570	0.5297	0.002776	2,000	112.0870	60.8125	20.2395	10,500
1,611 (25%)	3,218	0.5021	0.002901	2,100	89.7844	60.8107	20.2429	10,500
655 (10%)	1,306	0.2452	0.003614	2,300	73.9007	60.7556	20.2412	10,500
333 (5%)	662	0.2910	0.005877	2,500	72.4195	60.7721	20.2435	10,500
123 (2%)	242	0.3759	0.021800	2,200	78.5812	60.8709	20.2475	10,500
62 (1%)	120	0.8734	0.066800	2,200	77.9844	61.3126	20.2775	10,500
Progressive Meshes								
3,237 (50%)	6,470	0.13660	0.003451	–	70.6122	60.6412	20.2410	N.A.
1,618 (25%)	3,232	0.16740	0.004139	–	71.1858	60.6515	20.2401	//
647 (10%)	1,290	0.23770	0.006077	–	72.7364	60.6812	20.2420	//
323 (5%)	642	0.24700	0.011860	–	79.7237	60.7464	20.2522	//
129 (2%)	254	1.27770	0.065760	–	80.0257	60.6341	20.2769	//
64 (1%)	124	3.26100	0.225900	285	93.7107	60.3985	20.2175	//
Quadric Error Metrics								
3,237 (50%)	6,470	0.00067	8.622e-05	42.74	87.6357	61.3195	20.2433	N.A.
1,618 (25%)	3,232	0.00442	0.00021	44.41	79.0333	61.1082	20.2434	//
647 (10%)	1,290	0.03746	0.00183	45.11	76.4246	61.1372	20.2456	//
323 (5%)	642	0.09253	0.00581	47.63	74.6483	61.0833	20.2537	//
129 (2%)	254	0.33210	0.02552	45.61	76.3882	60.6684	20.2927	//
64 (1%)	124	1.23730	0.09879	47.58	90.9378	60.7801	20.4183	//

Table 3: Comparison of various simplification algorithms on the Fandisk mesh (errors are measured as percentages of the datasets bounding box diagonal; times are in seconds).

Femur (76,794 vertices, 153,322 triangles, bounding box 9,153x4,539x25,300)							
Edge Length 2,018.96, Area 2.89109e+08 (Volume is not defined: the surface is open)							
Mesh Decimation							
N_{Vert}	N_{Triang}	E_{max}	E_{avg}	*Time*	*EdgeLength.*	*Area*	*Mem. Kb*
26,707 (50%)	53,321	0.1015	0.0051	59.50	13,901.3	2.89192e+08	20,400
19,432 (25%)	38,779	0.0838	0.0067	70.27	13,318.3	2.89180e+08	20,400
7,963 (10%)	15,879	0.1479	0.0164	90.70	27,122.1	2.88517e+08	20,400
4,070 (5%)	8,126	1.8803	0.2353	145.70	224,766.0	2.84554e+08	20,400
1,535 (2%)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
767 (1%)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
383 (0.5%)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
76 (0.1%)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Simplifcation Envelopes							
38,365 (50%)	76,579	0.00505	0.00089	2,370.82	114,579.0	2.89111e+08	134,000
19,331 (25%)	38,556	0.01122	0.00309	2,413.18	68,985.3	2.89068e+08	135,000
7,717 (10%)	15,361	0.02760	0.00932	2,461.68	88,639.9	2.89089e+08	136,000
3,891 (5%)	7,720	0.04043	0.01310	2,828.98	109,368.0	2.89068e+08	137,000
1,565 (2%)	3,081	0.06924	0.02104	2,840.66	60,768.0	2.88223e+08	138,000
853 (1%)	1,675	0.19560	0.04780	3,317.23	102,810.0	2.88191e+08	139,000
383 (0.5%)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
76 (0.1%)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Multiresolution Decimation (Jade 2.0)							
38,397 (50%)	76,650	0.00528	0.00075	443.24	72,621.5	2.89111e+08	18,900
19,198 (25%)	38,305	0.01258	0.00262	655.27	52,913.2	2.89091e+08	20,000
7,679 (10%)	15,293	0.03080	0.00767	833.54	46,286.6	2.89036e+08	20,600
3,839 (5%)	7,624	0.04574	0.01217	928.86	79,860.1	2.89012e+08	20,800
1,535 (2%)	3,027	0.07177	0.01795	1,056.48	75,289.2	2.88739e+08	21,300
767 (1%)	1,501	0.10960	0.02741	1,099.67	52,753.9	2.88465e+08	21,600
383 (0.5%)	742	0.18710	0.04688	1,167.77	46,586.6	2.88291e+08	21,800
76 (0.1%)	140	0.87270	0.25900	1,329.72	221,591.0	2.84388e+08	22,400

Table 4: Comparison of various simplification algorithms on the Femur mesh (errors are measured as percentages of the datasets bounding box diagonal; times are in seconds).

Femur (76,794 vertices, 153,322 triangles, bounding box 9,153x4,539x25,300)							
Edge Length 2,018.96, Area 2.89109e+08 (Volume is not defined: the surface is open)							
Mesh Optimization							
38,299 (50%)	76,467	0.1390	0.003677	17,600	904,774.0	2.91810e+08	89,900
19,255 (25%)	38,416	0.1192	0.003607	17,800	204,190.0	2.90270e+08	89,900
7,621 (10%)	15,194	0.0612	0.006027	17,800	78,208.7	2.90141e+08	89,900
3,851 (5%)	7,663	0.0892	0.009159	18,600	93,085.0	2.89682e+08	89,900
1,558 (2%)	3,088	0.1001	0.013660	20,500	62,242.7	2.89021e+08	89,900
798 (1%)	1,569	0.1196	0.018810	21,600	56,873.5	2.89145e+08	89,900
383 (0.5%)	743	0.2192	0.029670	22,600	28,172.6	2.89388e+08	89,900
65 (0.1%)	121	0.7590	0.131700	25,200	227,970.0	3.00278e+08	89,900
Progressive Meshes							
38,397 (50%)	76,667	0.04385	0.00249	–	46,731.9	2.89222e+08	N.A.
19,198 (25%)	38,291	0.05645	0.00366	–	40,162.9	2.89343e+08	//
7,679 (10%)	15,286	0.05603	0.00673	–	45,424.0	2.89510e+08	//
3,839 (5%)	7,621	0.07896	0.01111	–	83,468.1	2.89411e+08	//
1,535 (2%)	3,027	0.12570	0.01648	–	74,620.8	2.89024e+08	//
767 (1%)	1,499	0.16630	0.02269	–	50,663.5	2.88807e+08	//
383 (0.5%	741	0.24310	0.03370	–	47,303.7	2.89016e+08	//
76 (0.1%)	140	0.85610	0.12940	2,860	170,563.0	2.92318e+08	//
Quadric Error Metrics							
38,397 (50%)	76,620	0.5118	0.00125	81.58	8447.2	2.88199e+08	N.A.
19,198 (25%)	38,264	0.6979	0.00337	101.90	9531.4	2.87968e+08	//
7,679 (10%)	15,263	0.7525	0.00845	115.09	26771.6	2.87756e+08	//
3,839 (5%)	7,604	1.0475	0.01591	119.81	55753.2	2.87085e+08	//
1,535 (2%)	3,022	1.4530	0.02809	121.51	58198.9	2.85770e+08	//
767 (1%)	1,501	2.2104	0.04889	123.46	48837.9	2.83578e+08	//
383 (0.5%)	742	3.3207	0.08336	122.89	86139.0	2.81512e+08	//
76 (0.1%)	141	8.1436	0.37860	123.49	225425.0	2.70867e+08	//

Table 5: Comparison of various simplification algorithms on the Femur mesh (errors are measured as percentages of the datasets bounding box diagonal; times are in seconds).

5 Concluding remarks

The paper presented a brief survey of the different mesh simplification methods proposed in the last few years. A characterization of the fundamental methods has been given, based on the simplification strategy, the error management policy and the capability to preserve mesh characteristics (e.g. topology, feature edges). Different error management strategies have been discussed and classified, with particular emphasis to the methods which support bounded error evaluation.

Moreover, the results of an empirical comparison of the simplification codes available in the public domain were presented. Six academic implementations, chosen to give a wide spectrum of different methods, were run on a set of sample surfaces. We compared empyrical computational cost and the approximation accuracy of the resulting output meshes.

From the accuracy point of view, the results obtained showed that *decimation* approaches based on global error evaluation produce the best results in terms of maximal error (under L_∞ norm), while their average error remains competitive to that produced by more computationally complex codes based on an *energy optimization* approach.

Finally, all of the solutions tested but the Quadric Error Metrics share a common weakness: they are defined to work on a single, topologically–sound mesh. This is not the general case in rendering CAD models or in virtual reality sessions, where we may need to simplify scenes or objects composed by multiple components, with a not topological–clean composition between components. New solutions are required for these applications to provide increased generality and robustness. First attempts in this direction have been recently proposed [31, 13].

6 Acknowledgements

We acknowledge the kind and timely cooperation of Hugues Hoppe and Micheal Garland, who ran their codes on the benchmark datasets. We would also like to thank our collaborators Andrea Ciampalini, who was responsable for the implementation of the Jade code and executed the simplification tests, and Claudio Rocchini, who implemented the Metro tool. Finally, we would also like to thank Marco Viceconti of IOR (Istituto Ortopedico Rizzoli) for the medical dataset he provided.

This work was partially financed by the Progetto Finalizzato "*Beni Culturali*" of the Italian National Research Council (CNR).

References

[1] M.E. Algorri and F. Schmitt. Mesh simplification. *Computer Graphics Forum (Eurographics'96 Proc.)*, 15(3):78–86, 1996.

[2] C. Andujar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Sole'. Automatic generation of multiresolution boundary representations. *Computer Graphics Forum (Eurographics'96 Proc.)*, 15(3):87–96, 1996.

[3] C. L. Bajaj and D.R. Schikore. Error bounded reduction of triangle meshes with multivariate data. *SPIE*, 2656:34–45, 1996.

[4] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '96), ACM Press*, pages 91–98, Aug. 6-8 1996.

[5] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13(5):228–246, June 1997.

[6] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution Representation and Visualization of Volume Data. Technical Report C97-05, Istituto CNUCE – C.N.R., Pisa, Italy, January 1997.

[7] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. Technical Report B4-01-01-96, I.E.I. – C.N.R., Pisa, Italy, January 1996.

[8] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Computer Graphics Proc., Annual Conf. Series (Siggraph '96), ACM Press*, pages 119–128, Aug. 6-8 1996.

[9] M. Deering. Geometry compression. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '95), ACM Press*, pages 13–20, 1995.

[10] M. Eck, T. De Rose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Computer Graphics Proc., Annual Conf. Series (Siggraph '95), ACM Press*, pages 173–181, Aug. 6-12 1995.

[11] Carl Erikson. Polygonal simplification: An overview. Technical Report TR96-016, Department of Computer Science, University of North Carolina - Chapel Hill, February 16, 1996.

[12] T.A. Funkhouser and C.H. Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex environment. In *Computer Graphics Proc., Annual Conf. Series (SIGGRAPH 93)*, pages 247–254. ACM Press, 1993.

[13] M Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '97), ACM Press*, 1997. (to appear).

[14] M.H. Gross, O.G. Staadt, and R. Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Trans. on Visual. and Comp. Graph.*, 2(2):130–144, June 1996.

[15] A. Guéziec. Surface simplification inside a tolerance volume. Technical Report RC 20440, I.B.M. T.J. Watson Research Center, 1996.

[16] M.J. De Haemer and M.J. Zyda. Simplification of objects rendered by polygonal approximations. *Computers & Graphics*, 15(2):175–184, 1991.

[17] B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197–214, 1994.

[18] B. Hamann and J.L. Chen. Data point selection for piecewise trilinear approximation. *Computer Aided Geometric Design*, 11:477–489, 1994.

[19] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel-based object simplification. In *IEEE Visualization '95 Proceedings*, pages 296–303. IEEE Comp. Soc. Press, 1995.

[20] T. He, L. Hong, A. Varshney, and S. Wang. Controlled topology simplification. *IEEE Trans. on Visualization & Computer Graphics*, 2(2):171–183, 1996.

[21] D.J. Hebert and H.-J. Kim. Image encoding with triangulation wavelets. *Proceedings SPIE*, (2569(1)):381–392, 1995.

[22] P. Heckbert and M. Garland. Multiresolution Modeling for Fast Rendering. In *Graphics Interface '94 Proceedings*, pages 43–50, 1994.

[23] P. Heckbert and M. Garland. Survey of surface simplification algorithms. Technical report, Carnegie Mellon University - Dept. of Computer Science, 1997. (to appear).

[24] P. Hinker and C. Hansen. Geometric optimization. In *IEEE Visualization '93 Proc.*, pages 189–195, October 1993.

[25] H. Hoppe. Progressive meshes. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '96)*, pages 99–108, 1996.

[26] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '93)*, pages 19–26, 1993.

[27] A. D. Kalvin and R.H. Taylor. Superfaces: Poligonal mesh simplification with bounded error. *IEEE C.G.&A.*, 16(3):64–77, 1996.

[28] A.D. Kalvin, C.B. Cutting, B. Haddad, and M.E. Noz. Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures. *SPIE Vol. 1445 Image Processing*, pages 247–259, 1991.

[29] R. Klein, G. Liebich, and W. Straßer. Mesh reduction with error control. In R. Yagel and G. Nielson, editors, *Proceedings of Visualization '96*, pages 311–318, 1996.

[30] K.L. Low and T.S Tan. Model simplification using vertex clustering. In *1997 ACM Symposium on Interactive 3D Graphics*, pages 75–82, 1997.

[31] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '97)*, pages 199–208, 1997.

[32] J. Popovic and H. Hoppe. Progressive simplicial complexes. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '97)*, pages 217–224, 1997.

[33] E. Puppo and R. Scopigno. Simplification, LOD, and Multiresolution - Principles and Applications. Technical Report C97-12, CNUCE, C.N.R., Pisa (Italy), June 1997. (also in: EUROGRAPHICS'97 Tutorial Notes, Eurographics Association, Aire-la-Ville (CH)).

[34] M. Reddy. Scrooge: Perceptually-driven polygon reduction. *Computer Graphics Forum*, 15(4):191–203, 1996.

[35] K.J. Renze and J.H. Oliver. Generalized unstructured decimation. *IEEE C.G.&A.*, 16(6):24–32, 1996.

[36] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum (Eurographics'96 Proc.)*, 15(3):67–76, 1996.

[37] J. Rossignac, editor. *Geometric Simplification (ACM SIGGRAPH Course Notes No.35)*. ACM Press, 1996.

[38] J. Rossignac and P. Borrel. Multi-resolution 3D approximation for rendering complex scenes. In B. Falcidieno and T.L. Kunii, editors, *Geometric Modeling in Computer Graphics*, pages 455–465. Springer Verlag, 1993.

[39] W. Schroeder. Polygon reduction techniques. In *ACM Comp. Graph. Proc., Annual Conf. Series (Siggraph'95), Course Notes n. 30 (Advanced Techniques for Scientific Visualization)*, pages 1.1–1.14, Aug. 6-12 1995.

[40] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *ACM Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[41] Marc Soucy, Guy Godin, and Marc Rioux. A texture-mappping approach for the compression of colored 3d triangulations. *The Visual Computer*, (12):503–514, 1996.

[42] Marc Soucy and Denis Laurendeau. Multiresolution surface modeling based on hierarchical triangulation. *Computer Vision and Image Understanding*, 63(1):1–14, 1996.

[43] G. Taubin and J. Rossignac. Geometric compression through topological surgery. Technical Report TR RC-20340(n.89924) 01/16/96, IBM Research Report, Yorktown (NY), January 1996.

[44] Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *ACM Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.

[45] *The Virtual Reality Modeling Language Specification - Version 2.0*, August 1996.

[46] Josie Wernecke. *The Inventor mentor: programming Object-oriented 3D graphics with Open Inventor*. Addison Wesley, 1994.

Figure 5: Simplified fandisk mesh ($\approx$ 250 faces).

Figure 6: Simplified fandisk meshes ($\approx$ 250 faces).

Figure 7: Fandisk mesh simplified with the Cosmo Poligon Reduction Editor (6,300 faces on the left, 1,278 faces on the right.