Volume xxx, (1998) number yyy pp. 000–000

Metro: measuring error on simplified surfaces

P. Cignoni†, C. Rocchini‡ and R. Scopigno§

Istituto per l'Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche, Pisa, Italy

Technical Note (Short contribution)

Abstract

This paper presents a new tool, Metro, designed to compensate for a deficiency in many simplification methods proposed in literature. Metro allows one to compare the difference between a pair of surfaces (e.g. a triangulated mesh and its simplified representation) by adopting a surface sampling approach. It has been designed as a highly general tool, and it does no assuption on the particular approach used to build the simplified representation. It returns both numerical results (meshes areas and volumes, maximum and mean error, etc.) and visual results, by coloring the input surface according to the approximation error.

Keywords: *surface simplification, surface comparison, approximation error, scan conversion.*

1. Introduction

Many applications produce or manage extremely complex surface meshes (e.g. volume rendering, solid modeling, 3D range scanning). Excessive surface complexity causes non interactive rendering, secondary–to–main memory bottlenecks while managing interactive visual simulations, or network saturation in 3D distributed multi-media systems. In spite of the constant increase in processing speed, the performances required by interactive graphics applications are in many cases much higher than those granted by current technology.

Substantial results have been reported in the last few years, aimed at reducing surface complexity while assuring a good shape approximation [13, 6]. The techniques proposed simplify [triangular] meshes either by merging/collapsing elements or by re-sampling vertices, using different error criteria to measure the fitness of the approximated surfaces. Any level of reduction can be obtained with these approaches, on the condition that a sufficiently coarse approximation threshold is set (an example is drawn in Figure 1).

† Email: cignoni@iei.pi.cnr.it
‡ Email: rocchini@calpar.cnuce.cnr.it
§ Email: r.scopigno@cnuce.cnr.it

A general comparison of the simplification approaches is not easy, because the criteria to drive the simplification process are highly differentiated and there is no common way of measuring error; an attempt has been recently presented [3]. In fact, many simplification approaches do not return measures of the *approximation error* introduced while simplifying the mesh. For example, given the complexity reduction factor set by the user, some methods try to "optimize" the shape of the simplified mesh, but they give no measure on the error introduced [18, 9, 8]. Other approaches let the user define the maximal error that can be introduced in a single simplification step, but return no *global error* estimate or bound [17, 7]. Some other recent methods adopt a global error estimate [10, 15, 2, 5] or simply ensure the introduced error to be under a given bound [4]. But the field of surface simplification still lacks a formal and universally acknowledged definition of *error*, which should involve shape approximation and hopefully preservation of feature elements and mesh attributes (e.g. color).

For these reasons, a general tool that would measure the actual geometric *"difference"* between the original and the simplified meshes would be strategic both for researchers, in the design of new simplification algorithms, and for users, to allow them to compare the results of different simplification approaches

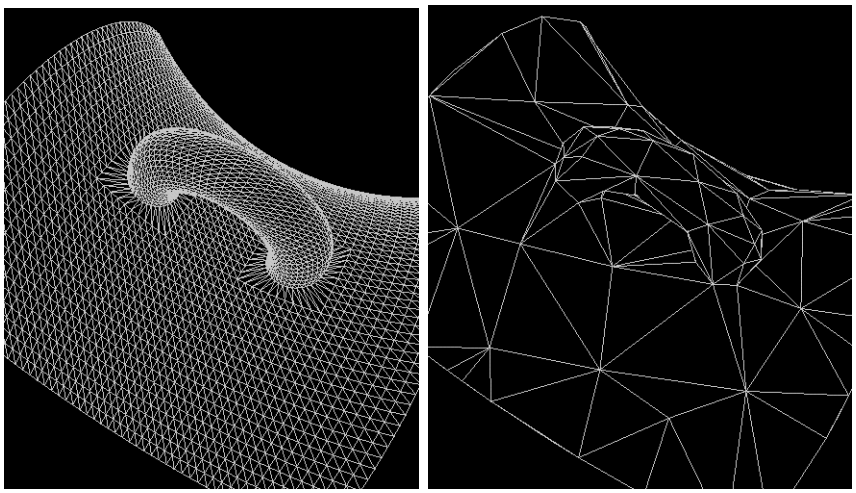

Figure 1: *A mesh simplification example: the original mesh (7,960 triangles) is on the left, a simplified one (179 triangles) is on the right.*

on the same mesh and to choose the simplification method that "best fits" the target mesh. In fact, even bounded precision methods [10, 15, 2, 5, 4] behave differently on different meshes. They generally ensure the user that the approximation will not be larger than a given threshold, but di not give data on the actual error distribution on the mesh. An example is the following query: are there sections of the mesh which hold an approximation much better than the given bound? And, if yes, what is their size and distribution?

Metro has been defined as a tool which is general and simple to implement. It compares numerically two triangle meshes, which describe the same surface at different levels of detail (*LOD*). *Metro* requires no knowledge on the simplification approach adopted to build the reduced mesh. *Metro* evaluates the difference between two meshes, on the basis of the *approximate distance* defined in the following section.

2. Terminology

We define here some terms that will be used in the following section (actually, all the measures evaluated by *Metro* follow the definitions below).
The approximation error between two meshes may be defined, as follows, as the distance between corresponding sections of the meshes. Given a point p and a surface S, we define the distance $e(p, S)$ as:

$$e(p, S) = \min_{p' \in S} d(p, p')$$

where $d()$ is the Euclidean distance between two points in E^3. The one-sided distance between two surfaces S_1, S_2 is then defined as:

$$E(S_1, S_2) = \max_{p \in S_1} e(p, S_2).$$

Note that this definition of distance is not symmetric. There exist surfaces such that $E(S_1, S_2) \neq E(S_2, S_1)$. A two-sided distance (Hausdorff distance) may be obtained by taking the maximum of $E(S_1, S_2)$ and $E(S_2, S_1)$.
Given a set of uniformly sampled distances, we denote the *mean* distance E_m between two surfaces as the surface integral of the distance divided by the area of S_1:

$$E_m(S_1, S_2) = \frac{1}{|S_1|} \int_{S_1} e(p, S_2)\, ds$$

If the surface S_1 is orientable we can extend the definition of distance between a point p of S_1 and S_2 so that, informally speaking, this distance e' is *positive* if the nearest point $p' \in S_2$ is in the outer space with respect to S_1, and *negative* otherwise (see Figure 2). Or, in other words, if N_p is the vector normal to S_1 in the sampled point p and $p' \in S_2$ is the nearest point, then the sign of our distance measure is the sign of $N_p * (p' - p)$.

This definition of signed distance is introduced to let *Metro* distinguish between positive and negative distances between two surfaces as follows:

$$E^+(S_1, S_2) = \max_{p \in S_1} e'(p, S_2)$$

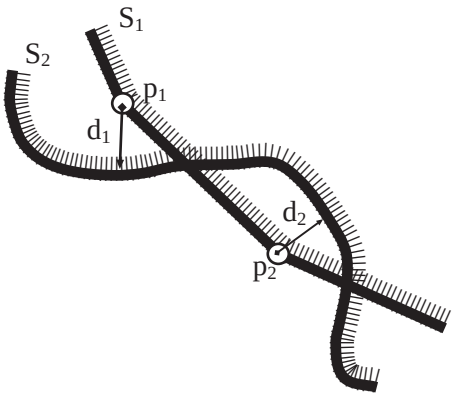

Figure 2: *Signed distance evaluation; distance is positive in p_1 and negative in p_2 (S_1 is the sampled curve).*

$$E^{-}(S_1, S_2) = |\min_{p \in S_1} e'(p, S_2)|$$

Signed distances are used by *Metro* to give an independent evaluation to the sections of the first mesh which are in the interior or in the exterior space with respect to the second mesh.

3. The *Metro* Tool

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* measure defined in the previous section. 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 the distance to the not–pivot mesh.
The idea is therefore to adopt an integration process over the surface. Surface sampling is achieved by *scan converting* triangular faces under a user-selected sampling resolution. The sampling resolution characterizes the precision of the integration, and we observed that in most cases a sufficiently thin sampling step size is 0.1% of the bounding box diagonal.
We also implemented a Montecarlo approach (generate random k points in the interior of each face, with the number k of samples proportional to the facet area), which gave similar results in terms of precision. Moreover, the adoption of Montecarlo sampling makes not possible the error visualization via error-texture mapping, because the latter requires a regular, raster sampling.

In an early version of our tool (*Metro v.1*) a ray-casting approach was adopted to compute *point–to–surface* distances. In order to improve performances and precision we adopted a different approach in the current release of *Metro, v.2*. Distances from the sampling point and the non-pivot mesh are now computed efficiently by using a bucketed data structure. Uniform grid (UG) techniques are very effective in geometric computations because in many cases elements which are far apart generally have little or no effect on each other [1]. Local processing can, therefore, highly reduce empirical complexity for many geometric problems. A 3D *uniform grid* is used in *Metro v.2* as an indexing scheme for the fast search of the nearest face to the sampling point. The bounding box of mesh S_2 is partitioned into cubic cells following a regular pattern. Then, we store in each cell c_{ijk} the list of faces of S_2 which intersect c_{ijk}. For each sampling point p, firstly we compute the distance between p and all the faces of the non-pivot mesh S_2 contained in the same grid cell of p. Then, adjacent grid cells are processed, in order of increasing distance from p, until we find that all not tested cells are farther than the current nearest face.
The distance between p and a single face of S_2 is computed using an optimized algorithm contained in the source code of the POV ray-tracer [12].

The strategy adopted implies that *uniqueness* of the nearest point is not ensured. According to the definition in Section 2, we might find multiple faces at minimal distance from the current sampling point. But, if we are looking for unsigned *approximation error*, then uniqueness is not a problem (because we are interested only in the value of this distance). Conversely, in the case of signed approximation error evaluation, having points at the same distance but holding different sign forces *Metro* to operate a random choice (and introduces a potential imprecision).

The worst case computational complexity of Metro depends on the surface area $\mathcal{A}(S_1)$ of the pivot mesh (measured in squared sampling step units) times the number n_f of faces of the non-pivot mesh. The resulting complexity is $O(\mathcal{A}(S_1) * n_f)$. But, if we use an UG, then we can expect that a much lower number of faces will be tested to compute the minimal distance for each sampling point. We measured in a number of runs that the mean number of faces evaluated for each sampling point is only few tens (as presented in Table 1). In Table 1 we report also the running times and the number of samples executed by *Metro* on three different pairs of meshes. Times are in seconds, measured on a SGI O2 workstation (R5000 180 Mhz, 96MB RAM).

An option is provided by *Metro* to compute a symmetric evaluation of the maximal error. At the end

S_1	S_2	sampling step	samples no.	tested faces no.	time
(faces no.)	(faces no.)			(per sample)	(sec.)
4,001	69,451	0.2	365,307	30.3	29
2,867	28,322	0.1	540,667	29.3	24.7
6,369	67,607	0.1	1,670,420	24.8	89.8

Table 1: *Number of sampling points, sampling step size, time and number of faces tested per sample on three different meshes.*

of the sampling process, if the $-s$ option is set, then *Metro* switches the pivot and not–pivot meshes and executes sampling again.

Given a sampling step, the mesh may contain triangles which have an area smaller than the squared sampling step. *Metro* manages this special case by adopting a random choice: a random variable is generated, with the probability of its TRUE value equal to the ratio between the triangle area and the squared sample area. If the random value is TRUE, a single point–to–surface distance is computed; otherwise, *Metro* starts the scan conversion of the next face.

Metro Input

Metro has a command-line input interface. The options available are shown, as usual, by typing: metro -h. The options available are shown in Figure 3.
The data formats accepted in input are either the OpenInventor [19] format or a raw indexed representation (a list of vertex coordinates, and a list of triangular faces, defined by the three indices to the vertex list).
The two meshes should have similar shapes (as in multiple level of detail representation). If the shapes differ too much, with the disappearance of significant features, the computation of the error might be locally imprecise. *Metro* considers excessive the difference between two meshes if their bounding box diagonals differ in length by more than 10%.
If the surfaces to be compared are *not orientable* or *multiple-connected*, then it would be impossible to distinguish between positive and negative errors (i.e. if the low detail mesh passes below or above the high detail mesh).

Metro Output

Metro returns both *numerical* and *visual* evaluations of surface meshes "likeness" (Figure 5 shows a snapshot of its GUI).

The format of the *numerical results* is reported in Figure 4. It contains data on input meshes characteristics (topology, size, surface area, mesh volume, feature edges total length, diagonal of the minimal bounding box, diameter of the minimal bounding sphere); the mean and maximum distances between meshes (returned using absolute measures and as a percentages of the diagonal of the mesh bounding box); and a very rough approximation of the positive, negative and total volume of the difference between the two meshes (i.e. the total volume `Vt` is the volume of $(S_1 - S_2) \cup (S_2 - S_1)$).
All the positive/negative measures follows the definitions in Section 2, and can be computed only if the input surfaces are orientable and single-connected.

Error is also *visualized* by coloring the pivot mesh with respect to the evaluated approximation error. Two different color mapping modalities are available:

- *per-vertex* mapping: for each vertex, we compute the error on each mesh vertex (as the mean of the errors on the incident faces), and assign a color proportional to that error. The faces are then colored by interpolating vertex colors;
- *error-texture* mapping: for each face, a rgb-texture is computed which stores the color-coded errors evaluated on each sampling point (mapped on a color scale).

The error-texture mapping approach gives visual results which in general are more precise, but whose visualization depends on the sampling step size used by *Metro*. See for example in Figure 6 the different visual representation of the same mesh zone.
In both cases, a histogram reporting the error distribution is also visualized on the left of the *Metro* output window (Figure 5).

When the *error-texture* mapping is used, we can also visualize the error by considering its sign: zero error maps to green, negative and positive to red and blue (see Figure 7).

Limited numerical precision management

The error evaluated by *Metro* may be affected by the limited 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

```
Usage: Metro file1 file2 [-a# -e# -h -l# -s ] [-r] [-q|v] [-b|bs|t]

  file1, file2 :  input meshes to be compared;

  -a#  crease angle setting for feature edges detection and
       classification. The angle value "#" is given in degrees,
       from 0 (all edges are classified 'feature edge') to 180 degrees.
       (it is used to measure the total length of the feature edges);
  -b   show error using "error-texture" mode (DEFAULT is "per-vertex" mode)
  -bs  show error using "signed error-texture" mode (green==> error=0);
  -e#  set the maximal absolute error in the histogram scale and color mapping;
       (it is useful to compare visually the results of two different runs of Metro);
  -h   show the Metro command syntax (and the options available);
  -l#  select the scan conversion step (value "#": percentage of the mesh bounding box);
  -q   use "quiet" (i.e. very synthetic) output;
  -r   use "Montecarlo" sampling (DEFAULT: use scan conversion);
  -s   compute symmetric maximum distance (double run);
  -t   set text mode only, do not visualize results under OpenInventor;
  -v   verbose output.

Example:  metro -v meshcomp.iv mesh.iv -l0.5 -a45
```

Figure 3: *Metro input options.*

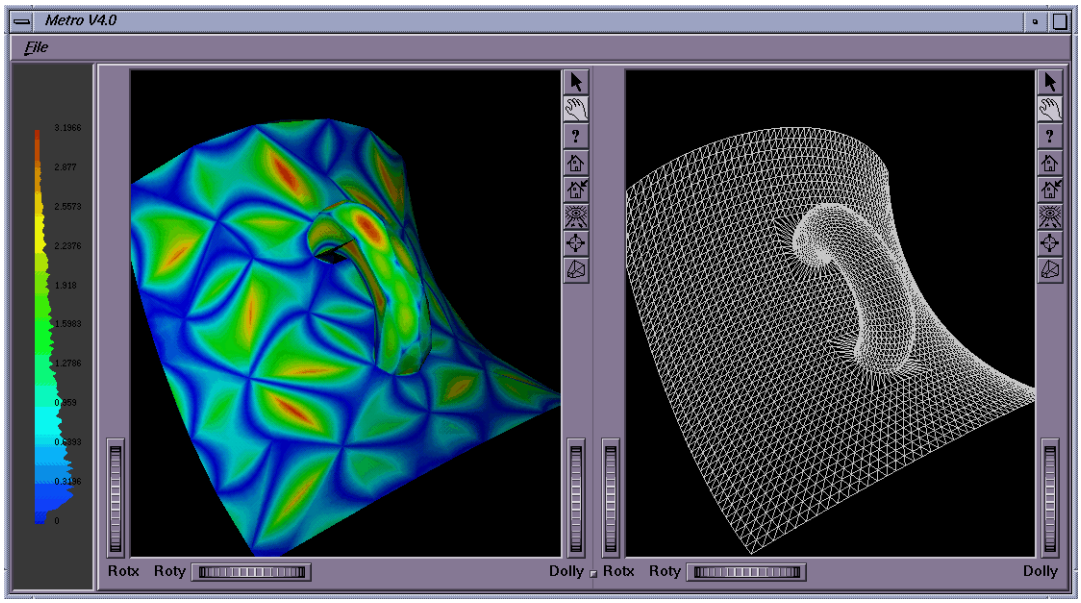

Figure 5: *The Metro graphic output window.*

with small area, and very elongated triangles.
Another problem may be the computation of the sum of hundreds of thousands of nearly zero values. To minimize rounding errors in the computation of the sum, we used a fanin algorithm (binary tree structured sum [11]).

4. Concluding Remarks

We have introduced a new tool, *Metro*, to allow simple comparisons between surfaces. Its main use is in the evaluation of the error introduced in the simplification of surfaces. *Metro* returns both numerical and visual evaluations of the meshes' likeness. These measures are computed using an error defined as an ap-

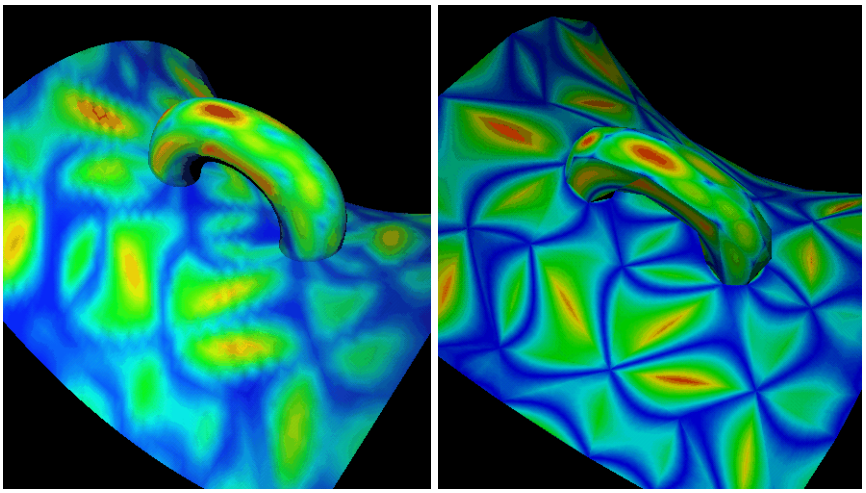

Figure 6: *Different color mapping modality:* per-vertex *mapping on the left, and* error-texture *mapping on the right.*

proximation of the *surface–to–surface* distance. The error is evaluated by: (1) scan converting the first mesh faces with a user–specified sampling step, and (2) computing a *point–to–surface* distance for each scan–converted point. The tool adopts well known techniques and can be simply implemented.

We tested with *Metro* the simplified meshes obtained with some public domain software. In the case of a bounded precision method, the Simplification Envelopes [4], we obtained error values very similar to the threshold set; in general, a slightly lower error is measured: 0.759 for a mesh simplified under target error 0.77, or 0.0884 for the relative target error 0.0895. But the added value of *Metro* in the case of a bounded error method is to give the possibility to *view* the distribution of the error on the mesh (Figure 5).

An important point to be considered in the evaluation of a surface simplifier is to what extent it preserves feature edges. The current implementation of *Metro* detects feature edges and returns, for each mesh, their total length. But this may not be sufficient: even two meshes with nearly equal total length of the feature edges might differ a lot.
Metro could be easily extended to get rid of this limitation. Given two set of feature edges F_1 and F_2, we might apply again a sampling approach. For each feature edge $e \in F_1$ and each sampling points $p_i \in e$, let us evaluate the minimal distance between p_i and the edges in F_2. These minimal distances can then be used to compute the maximum and mean displacements between the set of feature edges or also the maximum and mean angles between pairs of corresponding feature edges.

A limitation of *Metro* regards the topology changes that some simplification algorithms can introduce in the simplified surfaces [5, 14, 16]. *Metro* can only partially cover this issue. It returns the number of connected components of each mesh (and also if they are orientable and closed), and therefore in many cases we may detect if a topology change has occurred. But a more sophisticated approach is needed to detect each single change of topology and to measure the associated impact on meshes disparity.

5. Acknowledgements

Metro v.2 is available as public domain software at the *Visual Computing Group* web site of the CNUCE and IEI, C.N.R. Institutes at Pisa *(http://miles.cnuce.cnr.it/cg/metro.html)*.
This work was partially financed by the *Progetto Finalizzato Beni Culturali* of the Italian National Research Council (CNR).

References

1. V. Akman, W.R. Franklin, M. Kankanhalli, and C. Narayanaswami. Geometric computing and uniform grid technique. *Computer-Aided Design*, 21(7):410–420, Sept. 1989.
2. A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13(5):228–246, June 1997.
3. P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers And Graphics*, 22(1):37–54, 1998.

```
                Mesh1        Mesh2
             +------------+------------+
Orientable.. |         yes|         yes|
2Manifold... |         yes|         yes|
Closed...... |          no|          no|
Vertices.... |         115|        4079|
Triangles... |         179|        7960|
Conn. Comp.. |           1|           1|
BBox Diag... |     345.234|     345.842|
Diameter.... |      330.88|     329.457|
Edge Length. |     1628.01|     198.537|
Area........ |     78732.6|     78702.8|
Volume...... |    N/A     |    N/A     |
             +------------+------------+

                Mesh1        Mesh2
             +------------+------------+
Samples..... |                2462768|
             +------------+------------+
+---------------- Maximal Error ---------+
 E+ ..:       3.3314 (0.9649%) (1.0068%)
 E- ..:       3.2004 (0.9270%) (0.9672%)
+---------------- Mean Error ------------+
 E+ ..:       0.8271 (0.2395%) (0.2499%)
 E- ..:       0.8903 (0.2578%) (0.2690%)
 Et ..:       0.8636 (0.2501%) (0.261%)
+------------ Mean Square Error --------+
 E+ ..:       1.0622 (0.3076%) (0.3210%)
 E- ..:       1.1258 (0.3261%) (0.3402%)
 Et ..:       1.0994 (0.3184%) (0.3322%)
+------------ Volume of Difference -----+
 V+ ..:       2.86346e+10
 V- ..:       5.75679e+10
 Vt ..:       1.67459e+11

LEGEND:
Conn. Comp. = no. of connected components
Diameter    = is an approximate measure
Edge Length = total, FEATURE edges only
Samples     = no. of sampling points evaluated
```

Figure 4: *Numerical results produced by Metro on the meshes in Figure 1 (in this case the* `Volume` *measure is not available because the meshes are not closed).*

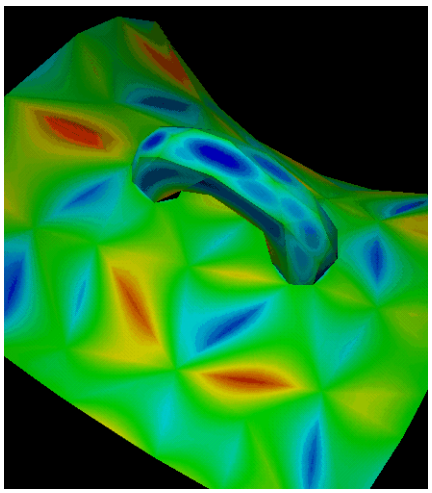

Figure 7: *Color may be mapped considering the* sign *of the error (i.e. the sign of the evaluated distance, defined only for orientable meshes).*

4. 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.

5. M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '97), ACM Press*, pages 209–216, 1997.

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

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

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

9. 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.

10. 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.

11. Peter Linz. Accurate floating-point summation. *Communications of the ACM*, 13(6):361–362, June 1970.

12. POV-Team. Persistence of vision raytracer 3.0. Publicly available on web: `http://www.povray.org/`, 1996.

13. E. Puppo and R. Scopigno. Simplification, LOD, and Multiresolution - Principles and Applications. In *EUROGRAPHICS'97 Tutorial Notes (ISSN 1017-4656)*. Eurographics Association, Aire-la-Ville (CH), 1997 (PS97 TN4).

14. M. Reddy. Scrooge: Perceptually-driven polygon

reduction. *Computer Graphics Forum*, 15(4):191–203, 1996.

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

16. 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.

17. 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.

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

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