The Visual Computer

Multiresolution decimation based on global error

A. Ciampalini[1], P. Cignoni[1], C. Montani[1], R. Scopigno[2]

[1] Istituto di l'Elaborazione dell'Informazione – Consiglio Nazionale delle Ricerche, via S. Maria 46, 56126 Pisa, Italy
E-mail: {cignoni,montani}@iei.pi.cnr.it
[2] Istituto CNUCE – Consiglio Nazionale delle Ricerche, via S. Maria 36, 56126 Pisa, Italy
E-mail: r.scopigno@cnuce.cnr.it

Due to surface meshes being produced at increasing complexity in many applications, interest in efficient simplification algorithms and multiresolution representation is very high. An enhanced simplification approach and a general multiresolution data scheme are presented here. JADE, a new simplification solution based on the mesh decimation approach, has been designed to provide both increased approximation precision, based on global error management, and multiresolution output. Moreover, we show that with a small increase in memory, which is needed to store the multiresolution data representation, we are able to extract any level of detail representation from the simplification results in an extremely efficient way. Results are reported on empirical time complexity, approximation quality, and simplification power.

Key words: Surface modeling – Mesh simplification – Bounded approximation error – Multiresolution

Correspondence to: R. Scopigno

1 Introduction

Surface simplification is a very hot topic in visualization because of several application-driven motivations. In fact, huge surface meshes are produced in a number of fields: volume visualization, where isosurfaces composed of millions of facets can be extracted on high-resolution volume datasets; virtual reality, where very complex models are built to cope with the complexity of real environments, e.g., to support high quality walk-throughs in virtual worlds; automatic modeling, where technologies for the automatic acquisition of the shape of 3D objects are emerging (e.g., 3D range scanners), and the high precision infers very complex meshes (in the order of hundreds of thousands of facets) (Curless and Levoy 1996); and free form surface modeling, where optimized polyhedral approximations of parametric surfaces produce highly detailed meshes.

Reducing the complexity of surface meshes is therefore a must to guarantee interactivity in rendering or, in some cases, to make the rendering itself possible. In fact, such large meshes often require more than the available storage and/or adversely affect the graphics performance of the current mid-level graphics subsystems.

This interest in surface simplification is also manifested by the large number of research groups who have started projects on this topic and by the many papers published in recent years.

Our goals in the design of a new solution for surface simplification are:

- *Approximation error*. An accurate estimation of the approximation error introduced in the simplification process.
- *Compression factor*. A reduction factor comparable or better than that of other approaches at the same level of approximation.
- *Multiresolution management*. Once the simplification process has been run, we want to make possible the interactive extraction of any level of detail (LOD) representation, with the complexity of a single LOD extraction being linear, in practice, with respect to the LOD output size.
- *Working domain*. Wide generality: the algorithm should not rely on the correctness of the surfaces (self-intersecting, nonmanifold, nonorientable surfaces are common in real-world data).

The Visual Computer (1997) 13:228–246

- *Space/time efficiency*. Short simplification times and low memory consumption to allow the management of large meshes.

We consider the third point one of the most significant in our approach: our goal is to provide a real *multiresolution representation*. To clarify this point, we first give a characterization of two terms that are sometimes considered synonymous. Given a surface S, we define them here.
A LOD representation is a data structure that holds a constant number k of different representations of the surface S, at different levels of detail or approximation (Funkhouser and Sequin 1993).
A multiresolution representation is a data structure that allows the compact representation of m representations, at different levels of detail, where m is somewhat proportional to the data size (e.g., the number of faces in S).
LOD representations are widely used in leading edge applications (e.g., virtual reality, 3D web graphics, etc.). A multiresolution representation is better than a LOD representation because the user or the application has much more flexibility in the selection of the "best" level of detail; in many cases, it is better to postpone the choice at run time, instead of forcing it in the preprocessing, simplification phase.
Our solution, just another decimator (JADE), has been designed as an enhanced decimation algorithm that provides bounded error management, high precision, good efficiency, and multiresolution management. The main points of our proposals are:

1. JADE is based on a *decimation approach*, which reduces mesh complexity by removing vertices. Topological classification of the vertices follows the original proposal by Shroeder et al. (1992), which is therefore briefly introduced in Sect. 3.
2. After the topological classification, we try to remove all the vertices that are candidates for removal. The criteria used for the evaluation of the error introduced by each removal are peculiar to our solution. We provide a *global approximation error* criterion, defined in Sect. 4, which supports high-precision error management.
3. To reduce accumulation of error during removal of vertices, the *ordering* adopted for the candidate vertices is critical. An ad hoc strategy for candidate vertex selection is presented in Sect. 5.
4. Sophisticated triangulation heuristics, extremely important for the minimization of the approximation error introduced in each removal action, are described in Sect. 6.
5. Finally, the availability of a global error measure enabled us to design a simple and efficient method to manage multiresolution. This is presented in Sect. 7.

We evaluated the JADE algorithm in depth with a number of well-known public domain datasets to prove the asserted precision and efficiency of our solution. For this reason, we compared (Sect. 8) the results obtained with JADE and those produced with other state-of-the-art simplification codes.

2 Related work

Substantial results have been reported in the last few years on surface simplification. The data domain of the solution proposed generally covers any type of triangular meshes (e.g., laser range data, terrains, and synthetic surfaces). These meshes are simplified either by merging elements or by resampling vertices, different error criteria being used to measure the fitness of the approximated surfaces. In general, any level of reduction can be obtained with most of the approaches listed next, on condition that a sufficiently coarse approximation threshold is set.
Among the existing methods we have:

- *Coplanar facet merging*. Coplanar or nearly coplanar data are searched for in the mesh, merged in larger polygons, and then retriangulated into fewer simple facets than those originally required (Hinker and Hansen 1993; Kalvin et al. 1991). *Superfaces* is a new approach based on face merging and bounded approximation, recently proposed by Kalvin and Taylor (1996).
- *Retiling*. New vertices are inserted at random on the original surface mesh, and then moved on the surface to be displaced on maximal curvature locations. The original vertices are then iteratively removed and a retiled mesh,

built on the new vertices, is given in output (Turk 1992).
- *Mesh decimation.* Based on multiple filtering passes, this approach analyzes locally the geometry and topology of the mesh and removes vertices that pass a minimal distance or curvature angle criterion (Schroeder et al. 1992). New decimation solutions that support global error control have been recently proposed (Bajaj and Schikore 1996; Cohen et al. 1996; Klein et al. 1996). In particular, the *simplification envelopes* method (Cohen et al. 1996) supports bounded error control by forcing the simplified mesh to lie between two offset surfaces. A local geometrical optimality criterion was also paired with the definition of a tolerance volume to drive edge collapsing and maintain bounded approximation (Gueziec 1996).
- *Mesh optimization.* In a way similar to that for mesh decimation, mesh optimization is achieved by evaluating an energy function over the mesh and minimizing such a function either by removing/moving vertices or collapsing/swapping edges (Hoppe et al. 1993). An enhanced version, *progressive meshes*, was recently proposed to provide multiresolution management, mesh compression, and selective refinements (Hoppe 1996).
- *Multiresolution analysis.* This approach uses remeshing, resampling, and wavelet parametrization to build a multiresolution representation of the surface from which any approximated representation can be extracted (Eck et al. 1995).
- *Vertex clustering.* Based on geometric proximity, the approach groups vertices into clusters and, for each cluster, computes a new representative vertex. The method is efficient, but neither topology nor shape are preserved (Rossignac and Borrel 1993).

In a recent Siggraph tutorial on surface simplification, Schroeder (1995) classifies the approaches just listed by highlighting two main orthogonal classifications: approaches that preserve mesh topology (e.g., mesh decimation and mesh optimization) and those that do not (e.g., vertex clustering); and approaches based on vertex subset selection (e.g., coplanar facets merging and mesh decimation) or resampling (e.g., mesh optimization, retiling, and multiresolution analysis). Other characteristics that can also be at the base of a classification are precise control and measurability of the approximation error introduced (e.g., simplification envelopes), preservation of discontinuities (e.g., mesh decimation and progressive meshes), and multiresolution management (e.g., multiresolution analysis and progressive meshes).

A general comparison of these approaches is not easy because efficiency depends largely on the geometrical and topological structure of the mesh and on the results required. For example, the presence of sharp edges or solid angles would be managed better by a decimation approach, while on smooth surfaces mesh optimization or retiling would give better results. However, the good results in the precision and conciseness of the output mesh given by mesh optimization and retiling techniques are counterbalanced by substantial processing times. Although no time comparisons of the various methods have been reported in the literature, an informed guess would be that the decimation approach is the most efficient method.

Several approaches have also been proposed for a particular occurrence of the simplification problem: how to reduce the complexity of the isosurfaces fitted on high-resolution volume datasets. These techniques are peculiar to volume-rendering applications and are less general than the previous ones. A possible classification is:

- *Adaptive fitting approaches.* Ad hoc data traversing and fitting are devised to reduce output data complexity by using approximated fitting (Criscione et al. 1996), bending the mesh (Moore and Warren 1992), or adapting the cell size to the shape of the surface (Muller and Stark 1993).
- *Datasets simplification approaches.* First, the volume datasets are organized into a hierarchical (Criscione et al. 1996; Wilhelms and van Gelder 1994) or a multiresolution (Cignoni et al. 1994) representation, and then isosurfaces are fitted into the simplified datasets.

3 Mesh decimation

The mesh decimation method (Schroeder et al. 1992) reduces mesh complexity by applying multiple passes over the triangle mesh and using local geometry and topology to remove vertices that

fulfil a distance or angle criterion. The resulting holes are then patched with a local triangulation process (3D recursive loop splitting). One characteristic of this method is that the simplified mesh has vertices that are a subset of the original ones. The criterion is based on local error evaluation. Each vertex v is classified topologically (looking at the loop of facets incident to v). For each vertex v that is a candidate for elimination, the algorithm computes:

1. The distance of v from the average plane, in the case of simple vertices (i.e., those surrounded by a complete cycle of triangles, and where each edge incident to v is shared by exactly two triangles in the cycle).
2. The distance of v from the new boundary edge, in the case of boundary vertices (i.e., those on the boundary of the mesh and within a semicycle of incident triangles).

In both cases, the approach computes an approximated estimate of the local error. The estimate is approximated because decimation does not compute precisely the error introduced by removing the vertex and retriangulating the hole. In fact, only a simplified and approximated estimate of the error between the selected vertex and the resulting mesh is considered.
In this way, the returned simplified mesh does not guarantee a bounded precision: successive simplification steps that affect the same mesh area may produce an accumulated approximation error that is much higher than the maximum allowed. However, significant sharp edges (which we call feature edges) are maintained by not removing all of their vertices. These vertices are detected by setting an angle threshold and testing, for all edges, the magnitude of the angle between the two adjacent faces. The power of the decimation approach is its time efficiency, which is higher than that of other methods. The construction of LOD representations is made possible by means of k successive iterations of the decimation process, with various error threshold settings.

4 Global error control

Our simplification solution is based on a vertex-decimation approach, with vertex selection driven by a criterion based on global error evaluation. There are two crucial reasons for providing a global error management: firstly, to give the user accurate control of the approximation error introduced and secondly, to allow easy management of multiresolution, as we clarify in Sect. 7.
Given: an input mesh S; an intermediate mesh S_i, obtained after i steps of the decimation process; a candidate vertex v on mesh S_i; the patch T_v of triangles in S_i incident to v; and, finally, the new triangulation T'_v that replaces T_v in S_{i+1} after the elimination of v. Then, the error introduced while removing vertices can be measured in terms of local and global error:

- *Local error* measures the local approximation introduced by replacing the patch T_v in S_i with T'_v and generating the new mesh S_{i+1}. The local error can be estimated either in an approximate manner, by computing the distance of v from the mean plane over T_v (Schroeder et al. 1992), or in a more precise approach, proposed in the following;
- *Global error* measures the error of approximation introduced if the corresponding subarea of the original input mesh S is represented by the new mesh parcel T'_v.

The exact global error can be defined as the symmetric Hausdorff distance between the two surfaces (Klein et al. 1996). Although global error management may be not cheap if we require a precise evaluation, a number of efficient heuristics can be used to approximate it.
A first technique that generally yields an overestimation of the actual global error is based on accumulating the local errors introduced in each simplification step. We maintain the error accumulated on each facet of the mesh. Initially, the error is set to null for each triangle $t \in S$. For each simplification step, we first compute the new local error ε_{l_1} generated by the removal of the candidate vertex v. Then, we select the facet $t_k \in T_v$ that has the largest accumulated global error, and set the sum of $\varepsilon_{l_1}(v, T_v)$ and of the global error in t_k as the new global error of each facet in the new patch T'_v.
Following this approach, the global error on each vertex v_j in the mesh S_i is computed as the maximum of the global error of all the facets incident to v_j. The global error of the current

mesh S_i is the maximum of the global error of all the facets in S_i.

As far as the local error estimate is concerned, the original decimation criterion (distance of v from the mean plane or from the edge) gives an excessively rough approximation of the local error (Schroeder et al. 1992).

Another possibility is to compute, for each simplification step, the distance between the new patch T'_v and the previous patch T_v. For each $t' \in T'_v$, we select a set of points Q on t'. Q contains the baricenter of t', the three points halfway between the baricenter and the vertices of t', and n points generated at random in the interior of t' (with n directly dependent on the surface area of t'). Then, for each point $q \in Q$, we compute the distance between q and the patch T_v. We therefore compute the local error ε_{l_1} on t' as the maximum of these distances:

$$\varepsilon_{l_1}(t', T_v) = \max_{q \in Q}(dist(q, T_v)),$$

and the global error on a single facet is therefore evaluated as:

$$\varepsilon_{g_1}(t') = \varepsilon_{l_1}(t', T_v) + \max_{t \in T_v} \varepsilon_{g_1}(t)$$

with $t \in T_v$, $t' \in T'_v$ and $S_{i+1} = S_i - T_v + T'_v$, and the precision of the simplified mesh S_{i+1} is obviously defined as the maximum of the errors on the facets of S_{i+1}.

This approach guarantees that the global error will show an increasing monotonic behavior as far as simplification proceeds, but it returns, in general, an over-estimation of the effective error (Fig. 1) or, in a few cases, an under-estimation. Global error under-estimations are rare (especially after a few simplification steps), but are possible because a local error computation based on sampling always returns an under-estimation of the real error.

A second possibility is to compute the global error directly. To do this, we maintain, during the simplification, a trace of the removed vertices, and each facet $t \in S_{i+1}$ is linked to the subset of the removed vertices that are "nearest" to t. A relation of proximity between removed vertices and facets of the mesh S_{i+1} is therefore defined and incrementally updated. Given the removed vertices, for each facet t we can compute an estimate of the global error by choosing the maximum of the distances between these vertices and t.

Given the ith simplification step, where vertex v is removed and patch T_v is replaced by T'_v, the proximity relation is updated as follows:

1. Detect the facet $t_j \in T'_v$ that is nearest to v and assign v to t_j.
2. For each $t \in T_v$, redistribute the sublist of removed vertices ($vert\_removed(t)$) on the facets of the new patch T'_v.
3. For each facet $t' \in T'_v$ and for each vertex in $vert\_removed(t')$, compute the new vertex-face distance $dist(v_i, t')$.

The global error on a single facet is now defined as:

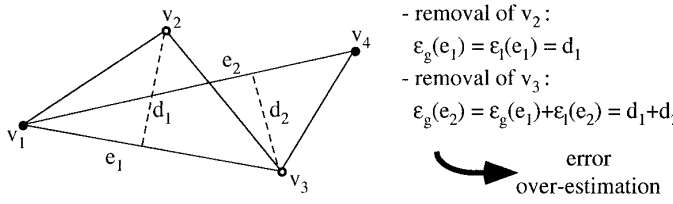
$$\varepsilon_{g_2}(t') = \max_{v_i \in vert\_removed(t')} dist(v_i, t').$$

This second estimate requires shorter computing times than the previous one, but again it may produce an under-estimation of the effective error. Using this approach, the first stages of simplification are highly critical because a large number of facets t' are connected to only a few (or any) removed points. Obviously, an evaluation of distances on a small set of sampling points may lead to an under-estimation of the error.

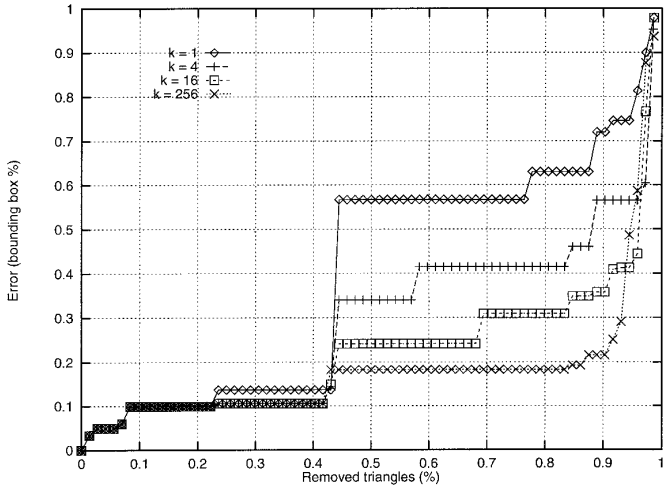
For this reason, we decided to use both approaches at the same time, in order to assure maximal precision in error evaluation. The joint global error is therefore defined as:

$$\varepsilon_{g_3}(t') = \max(\varepsilon_{g_1}(t'), \varepsilon_{g_2}(t')).$$

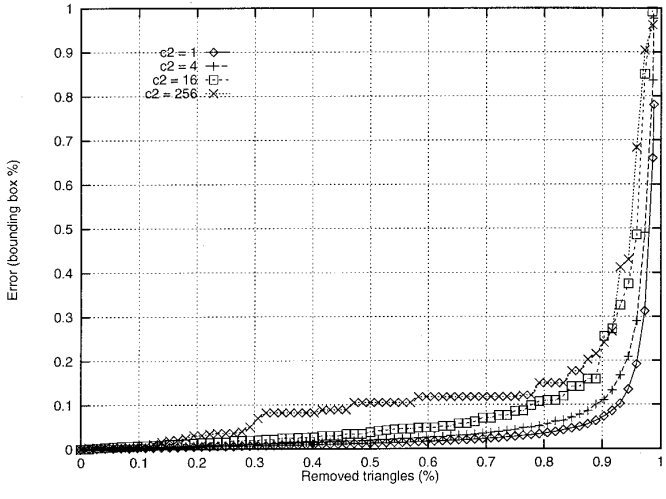
The precision of the proposed approaches has been evaluated empirically by a number of tests on real meshes. Some results are reported in Table 1, where we compare a number of meshes produced by JADE with different settings of the threshold error selected by the user (ε*). The two global error estimates $\varepsilon_{g_1}(t')$ and $\varepsilon_{g_3}(t')$ just defined were used to produce the results presented. The effective error of these meshes was measured in terms of maximum (E_{max}) and mean error (E_{avg}) with the Metro tool (Cignoni et al. 1996b), which is briefly described in Sect. 8.



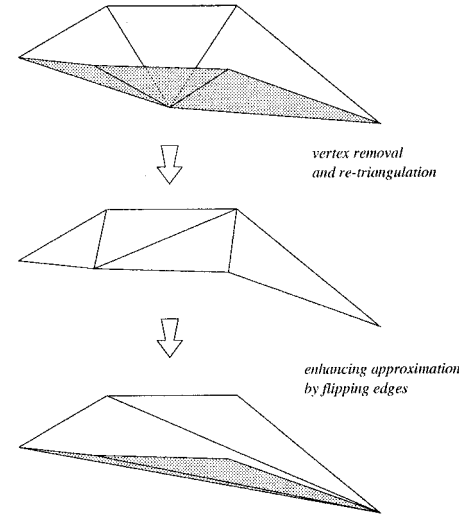
1



2



3



4

Fig. 1. Possible over-estimation of global error (using the first evaluation approach)

Fig. 2. Global error increases when various values are associated with the constant k in the $\varepsilon_l^*(j)$ function (cosine surface data)

Fig. 3. Increase in the global error with different settings of the amplification constant c_2 in $\varepsilon_g^*(j)$ function (cosine surface data)

Fig. 4. Flipping may improve mesh quality

Table 1. Various global error estimates (ε_{g_1} and ε_{g_3}) are compared on the Bunny dataset (34 834 vertices and 69 451 faces)

Stanford Bunny (bounding box $15.6 \times 15.4 \times 12.1$)								
	ε_{g_1}				ε_{g_3}			
ε^*	N_{triang}	T_{sec}	E_{max}	E_{avg}	N_{triang}	T_{sec}	E_{max}	E_{avg}
0.10	6646	255.79	0.147	0.0275	7545	285.38	0.139	0.0238
0.20	3021	264.38	0.282	0.0575	3397	310.05	0.225	0.0503
0.50	1051	271.49	0.747	0.1506	1176	331.30	0.499	0.1336
0.75	654	279.89	1.227	0.2216	726	336.96	0.736	0.2027
1.00	455	278.23	1.458	0.2986	510	340.11	0.985	0.2784

5 Setting an order in candidate vertex selection

In our algorithm, the strategy of selecting the vertices that are candidates for removal is analogous to the mesh decimation approach as far as the topological classification of vertices is concerned. However, the order in which vertices are decimated is unique to our solution. We process candidate vertices in order of increasing both local and accumulated global errors.
The order introduced in the selection of the candidate vertices has two positive effects:

1. The approximation error (defined as the maximum of the global errors of its facets) increases more slowly, and the quality of the obtained mesh improves (Fig. 6).
2. A smooth error growth is mandatory to ensure the high quality of the meshes that will be retrieved from the multiresolution representation (see Sect. 7 for details).

Let us call ε^* the target error. It is set by the user and it has to be satisfied by the final simplified mesh. The original mesh decimation algorithm classifies a vertex as a candidate for selection if the local error ε_l is lower than ε^* (and for this reason it cannot guarantee the final accumulated error to be lower than ε^*).
In our solution, we take into account ε^*, ε_l, and ε_g and we adopt a criterion that takes into account the progressive state of simplification. To evaluate whether a given candidate vertex exceeds the error threshold set by the user, we adopt an estimate rule that takes into account the current iteration depth. This is because we do not compare the current local and global errors with ε^*, but with a function over the iteration depth that returns values that progressively increase up to ε^*. Taking into account the iteration depth is fundamental. For each jth iteration, we scan the list of vertices and select for removal any vertex that has local and global errors lower than the iteration j target errors. We therefore define two functions that set the current iteration targets, for both the local and global errors:

$$\varepsilon_l^*(j) = \min\left(\frac{\varepsilon^*}{k} \cdot c_1^j, \varepsilon^*\right)$$

$$\varepsilon_g^*(j) = \min(\varepsilon_l^*(j) \cdot c_2, \varepsilon^*),$$

where k, c_1, and c_2 are constants.
With the first function, selection is driven by choosing at each iteration only the vertices that satisfy a "low" local error, and the threshold used to evaluate the error fitness increases with the iteration level. Moreover, with the second function, we select among the previous ones only those vertices such that the sum of their accumulated global error, together with the current local error, does not exceed the current iteration global target error $\varepsilon_g^*(j)$. In this way, at each iteration we manage to keep the global error of the obtained mesh as low as possible. Then, at the end of each iteration (i.e., when no more candidate vertices are available) we increase these target values until the value set by the user for the target error is reached. The number of iterations performed by our algorithm will obviously depend on the values chosen for the constants in the two functions.
Regarding the $\varepsilon_l^*(j)$ function, the best results have been obtained by setting $k \geq 256$ and $c_1 \leq 1.3$. By

"best results" we mean the lowest increase in the accumulated global error, as shown in Fig. 2. With regard to the $\varepsilon_g^*(j)$ function, the experiments show that low values of the constant c_2 (in the range 1. … 2.) have to be set. Figure 3 represents how much the approximation error is affected by various choices for the global error constant c_2. Figures 2 and 3 report graphs of the error returned by JADE on the cosine surface (a synthetic dataset, see Fig. 5).

6 Best fitting triangulation

We designed the triangulation as a two-phase process. Firstly, we compute a valid triangulation T_v' that fills the hole generated by the removal of vertex v. Then, we improve the quality of the new patch by a series of edge-flip actions.

We used a 2D approach for the triangulation of the hole resulting from the removal of patch T_v. This was to reduce the complexity of the problem and to increase robustness. The solution implemented evaluates a set of 14 planes by projecting the border of T_v on each of them until it finds a "valid" projection plane (i.e., a plane where the projection of T_v has no intersecting edges). The planes evaluated progressively are: (a) the average plane with respect to T_v; (b) the three planes orthogonal to the axes; (c) for each axes pair, the two quadrant bisector planes (in total six planes); and (d) the four planes orthogonal to the octant diagonals. If none of these 14 planes gives a valid projection, the candidate vertex removal action fails (but no more than 0.05% occurrences of this case have been found in practical experiments).

Two 2D triangulators were implemented and tested: an ear-cutting solution (Hinker and Hansen 1993; O'Rourke 1994) and minimum angle modification of the previous one.

The ear-cutting algorithm is a very simple incremental solution, which builds the triangulation by cutting from the current n-sided polygon each couple of adjacent edges that form an angle lower than π. The algorithm iterates for a maximum of n times, and it stops as soon as $n-2$ facets have been cut.

The minimum angle solution adopts an incremental approach similar to the previous one, but here we choose the next vertex onto which the next triangle is built by adopting a minimum angle criterion. The internal angles of the polygon are computed and sorted, and at each step we choose the minimum one to build the next triangle.

The validity of the 2D triangulation obtained is then checked in 3D space by testing:

- Whether any internal edge intersects the border of T_v
- Whether there are pairs of intersecting edges
- Whether all triangles are contained in the interior of T_v
- Whether any new edge was also part of the S_{i-1} mesh (this occurrence would modify the topology of the mesh, and we want to prevent that)
- Whether any triangle has a particular ill-conditioned shape (aspect ratio evaluation).

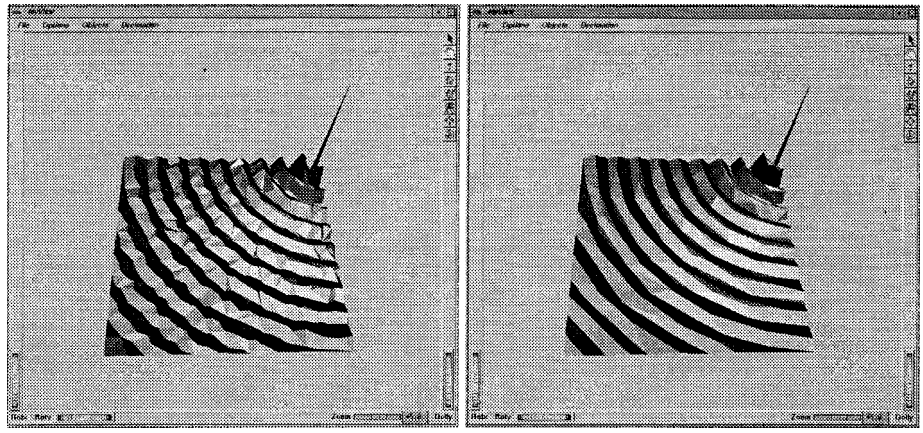
If any of the previous conditions are verified, then the 2D-valid, but 3D-invalid triangulation is rejected.

The minimum angle solution proved to be faster. We therefore use the ear-cutting triangulator only in the case where the minimum angle one does not produce a 3D-valid triangulation. This approach guarantees computational efficiency and reduces the number of unsuccessful triangulations.

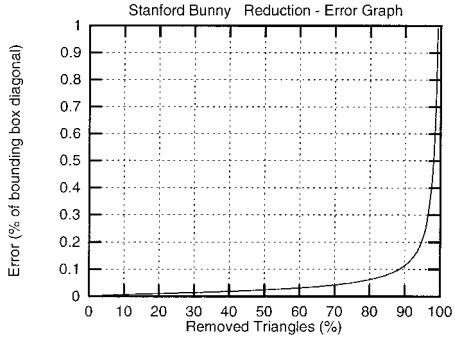
To get the best approximation of the removed patch T_v, and to reduce the error introduced in each individual simplification step, we process triangulation T_v' further. How well the triangles in T_v' actually fit the removed mesh has to be carefully evaluated (Fig. 4). In many cases, the regularity in shape of the mesh is not sufficient to guarantee high precision in the approximation of the original mesh, which is our main objective.

The quality of the approximation may be significantly increased through a series of edge-flip actions. Each internal edge e of T_v' has two incident triangles, t_1 and t_2, whose union is a quadrilateral with e as diagonal. If we flip e, two new triangles are generated to replace t_1 and t_2, the pair t_{flip1} and t_{flip2}. We evaluate whether the new pair of triangles gives a better approximation with respect to the original mesh S and, if they do, we perform the flip. To do this, a number of heuristics may be devised:

- *Edge-vertex distance.* We compute the distances between the two edges e and e_{flip}, and the removed vertex v.



5



6

Fig. 5a, b. Simplification of a synthetic mesh (cosine-mesh): **a** without flipping; **b** with flipping

Fig. 6. Intermediate errors obtained in the simplification of the Stanford Bunny dataset

- *Approximated volumetric error*. For each pair of triangles, we compute the volume of the two tetrahedra they form with the removed vertex v (base: one triangle, apex: vertex v) and compare these volumes: the lower the volume, the better the approximation.
- *Sampled local error*. We compute the error associated with the two triangle pairs, one before and the other after flipping, by sampling distances with the patch T_v (Sect. 4).
- *Aspect ratio*. The couple with the best aspect ratio is preferred.
- *Area difference*. We compute the difference between the area of T_v and T_v': the smaller the difference, the better the approximation.

Table 2 shows how both the times and precision of the mesh increase when volumetric or sampled local criteria are used independently to drive flipping. The errors reported are measured in a post-processing phase with the Metro tool (Sect. 8).

In the current JADE implementation, we make use of a composed evaluation function that takes into account some of the previous heuristics. For each flipping action, the aspect ratio cannot worsen more than a user-specified threshold (usually, a threshold of 50% is set) with respect to the removed patch T_v aspect ratio, and the flip action is accepted if both volumetric and sampled local error criteria detect an improvement of the approximation with respect to the patch before flipping. The adopted approach is therefore to maintain a good aspect ratio and to improve approximation.

Flipping has proved to be crucial in increasing the quality of the reduced mesh. Figure 5 shows that a dramatic improvement may be introduced on mesh precision by edge flipping. The meshes

Table 2. Comparison of results with different flip heuristics. Errors (E_{max}, E_{sq_avg}) are measured as percentages of the mesh bounding box diagonal

Edge flipping				
	Triangles	Time (in seconds)	E_{max} (%)	E_{sq_avg} (%)
Original mesh	28322	–·–	–·–	–·–
No flip	1026	30.09	1.221	0.104
Volumetric	409	72.95	0.589	0.108
Sampled local	412	97.10	0.535	0.093

presented in Fig. 5 are the simplifications of analytic datasets, called cosine-meshes, that are obtained by evaluating the function:

$$z = \frac{\cos(\mathrm{sqrt}(x^2 + y^2))}{\mathrm{sqrt}(x^2 + y^2) + 0.001}.$$

In the test of Fig. 5, we used the same target error in both simplifications, and we applied flipping to produce the mesh in the image of Fig. 5b. Using flipping, we obtain a more precise approximation with a number of facets nearly one-half of those used without flipping (mesh of Fig. 5b: 412 triangles and Fig. 5a mesh: 1026 triangles).

7 Multiresolution representation

Given a decimation process, we want to produce a multiresolution representation S_M such that, given any precision threshold ε, the retrieval of an approximate model that satisfies precision ε should be achieved by a simple and fast traversal of S_M.

Let us consider the set T of all the triangles that were generated during the whole decimation process, including the triangles of the original mesh. Each facet $t \in T$ is characterized by two time stamps: its creation (or birth, i.e., when t is generated as part of a new patching submesh) and its elimination (or death, i.e., when t is found as one of the triangles incident to a vertex candidate for removal).

An intermediate mesh S_j is associated by definition with each time stamp j, and therefore we can associate with each time stamp j the global approximation error held by mesh S_j. Given the birth and death time stamps, each facet $t \in T$ is therefore tagged with two errors ε_b and ε_d, with $\varepsilon_b < \varepsilon_d$, called the birth and death errors of t, respectively.

Our multiresolution representation S_M is therefore the set of facets in T, having the associated ε_b and ε_d errors explicitly stored for each facet t.

The extraction of a representation S_ε from S_M at a given precision ε is therefore straightforward: S_M contains sufficient information to reconstruct it. S_ε is composed of all of the facets in S_M so that their life interval contains the error threshold searched for ($\varepsilon_b < \varepsilon \leq \varepsilon_d$).

A very efficient technique to extract a model S_ε from S_M is to use an interval tree (Edelsbrunner 1980) to retrieve all the facets whose life intervals (i.e., ε_b and ε_d) covers ε. [For an example of the application of the interval tree data structure in visualization and evaluation of the results, see Cignoni et al. (1996a).] Using such a structure, we are able to retrieve all the m facets of a model S_ε with a complexity of $O(m \log k)$, where k is the number of decimation steps. Some experiments show that the use of an interval tree permits interactive extraction of approximated models from the Bunny (see Sect. 8 for details on the dataset specifications), in multiresolution representation with a constant per-triangle rate of the order of 1.5 M triangles/s.

A significant advantage of our multiresolution approach is its generality. It does not depend on the particular simplification approach adopted. More generally, the definition of the model works either in the case of refinement-based or simplification-based approaches, provided that intermediate meshes with resolutions close to each other can be related through local changes and that a global, monotonic error measure is defined on the meshes.

Four snapshots of a simple tool that allows the interactive extraction and visualization of various levels of detail from the multiresolution representation are shown in Figs. 7 and 8.

Experimental results show that the number of facets stored in S_M is not much larger than the number of facets in the model at maximum resolution S (approximately less than three times), though a large number of resolutions are available. In Table 3, we show the storage size of our multiresolution representation on the cosine-mesh datasets, and we compare it with the size of

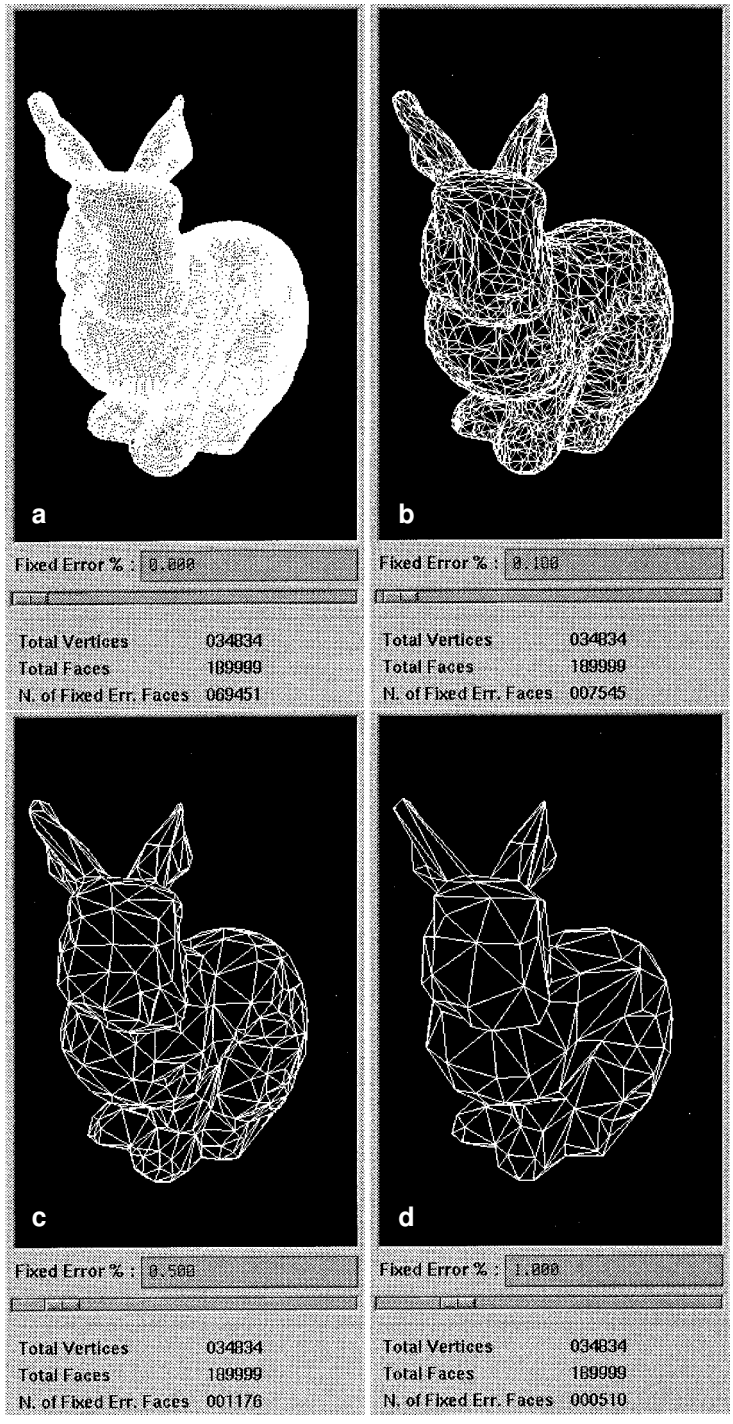


Fig. 7a–d. Four meshes extracted at different resolutions (the full resolution Bunny mesh is shown in **a**)

three more classical LOD representations (with errors set to 0%, 1.25%, 2.5%, 3.75%, and 5% in the case of the five models in LOD representation. Storage size is evaluated by considering a naive representation of the mesh. In the case of LOD representation, we count 12 bytes/vertex and 12 bytes/triangle. In the case of the multiresolution representation, the sizes are 12 bytes/vertex and

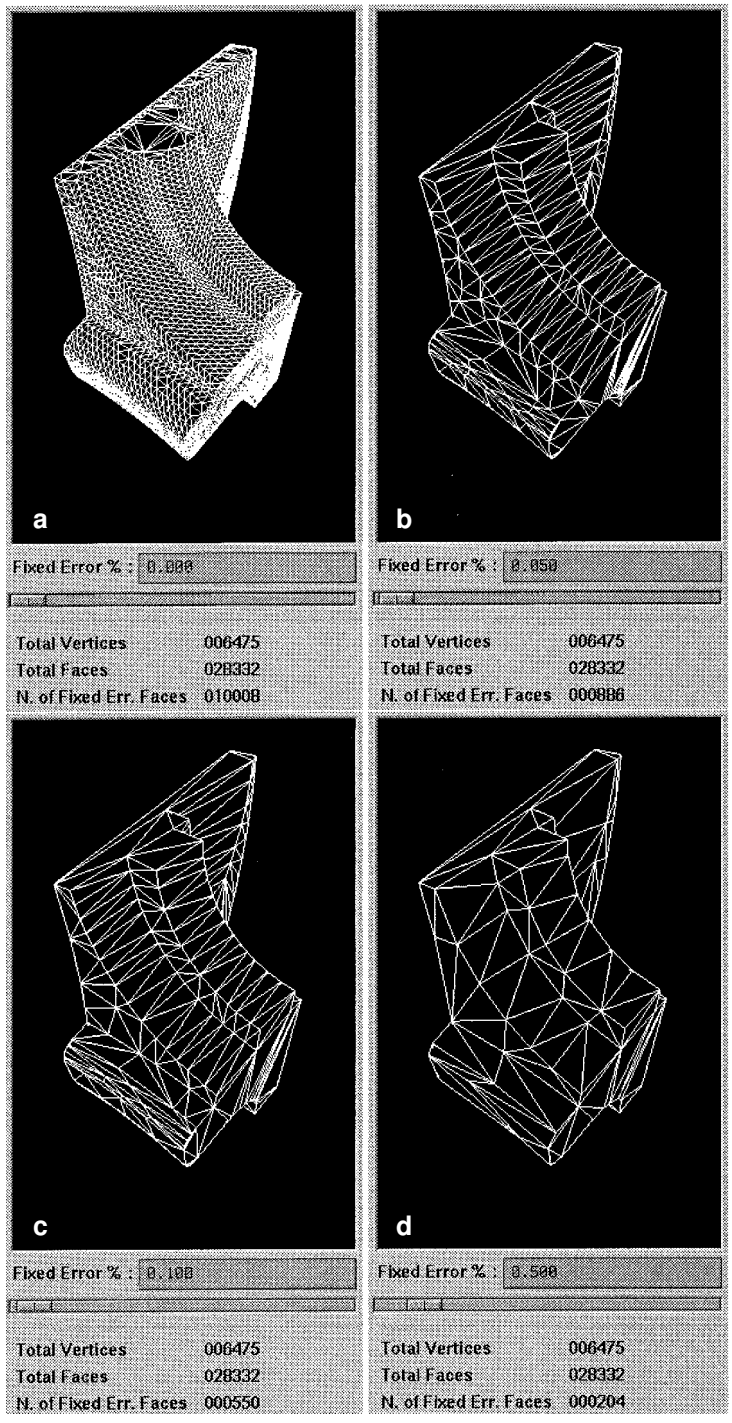


Fig. 8a–d. Four meshes extracted at different resolutions (the full resolution Fandisk mesh is shown in **a**)

20 bytes/triangle (12 bytes for the three vertex indices, 8 bytes for the ε_b and ε_d errors). Obviously, techniques that reduce mesh size are available, based on vertex/triangle proximity and quantization (Deering 1995), and can be applied to both types of representations.

Moreover, if a LOD representation is built with n independent runs of a standard surface

Table 3. Multiresolution versus level of detail (LOD) representation: space and construction time comparison (input mesh: cosine-mesh dataset – 28322 triangles, 14400 vertices)

	Triangles	Size (bytes)	Time (in seconds)
Multiresolution-861 models	78404	1740895	144
LOD-3 models	41936	754848	167
LOD-5 models	69587	1252566	280
LOD-9 models	125603	2260854	505

simplification code, then the times required to build multiresolution or LOD models are in favor of the multiresolution approach (see the times presented in Table 3).

As introduced in Sect. 5, the shape of the approximation error curve is critical. While performing decimation, the approximation error growth should be slow and smooth to ensure that a large number of different approximated meshes S_j are stored in S_M and to ensure that all the meshes S_j extracted from S_M guarantee a good reduction factor. Figure 6 shows that, thanks to the heuristics adopted for the candidate vertex selection and the attention paid to finding the best triangulation for each removed patch, the approximation error follows a very smooth curve indeed.

8 Results and evaluations

The proposed algorithm, JADE, has been implemented and is distributed on the public domain (see the Acknowledgements section). The results presented here were obtained on an SGI Indigo2 workstation (R4400 200 MHz cpu, 16 KB primary cache, 1 MB secondary cache, 32 MB RAM, IRIX 5.3).

In order to evaluate both the simplification rate and the quality of the approximation, we ran a number of tests on two public domain datasets: the "Bunny" mesh, acquired at Stanford with a range scanner, modeled with 34834 vertices and 69451 triangles; and the "Fandisk" mesh, a synthetic CAD model of a gas turbine engine component modeled with 6475 vertices and 12946 triangles. [The original model of the "Bunny" was created from laser range data with Turk and Levoy's (1994) mesh zippering algorithm. The original model of the "Fandisk" is included in the mesh optimization package (Hoppe et al. 1993).] The first mesh was chosen as a valid representative of free-form or acquired surfaces, and it presents a "bumpy" surface that increases mesh complexity. On the contrary the second is a typical CAD model, with sharp edges and surfaces characterized by sophisticated curvature.

Tables 4 and 5 present the precision obtained in a number of runs of our code on the two test datasets. In the runs reported, the value selected for feature edges classification is $\pi/6$ (i.e., all those edges where a pair of adjacent faces form an angle lower than 60 degrees).

Times are given in CPU seconds and measure the cost for the construction of a multiresolution representation that stores all approximations from error 0.0 to the threshold set by the user.

The approximation error obtained after simplification was evaluated with Metro, a tool we developed to measure the difference between meshes (Cignoni et al. 1996b). To compare two triangle

Table 4. Bunny dataset: numerical evaluation of simplified mesh quality

Stanford Bunny (bounding box $15.6 \times 15.4 \times 12.1$)					
JADE				Metro Evaluation	
N_{Vert}	N_{Triang}	T_{Sec}	ε^*	E_{max}	E_{avg}
34834	69451	– . –	– . –	– . –	– . –
3845	7545	285.38	0.10	0.139	0.0238
1764	3397	310.05	0.20	0.225	0.0503
628	1176	331.30	0.50	0.499	0.1336
391	726	336.96	0.75	0.736	0.2027
276	510	340.11	1.00	0.985	0.2784

Table 5. Fandisk dataset: numerical evaluation of simplified mesh quality

Fandisk (bounding box $4.8 \times 5.6 \times 2.7$)					
JADE				Metro Evaluation	
N_{Vert}	N_{Triang}	T_{Sec}	ε^*	E_{max}	E_{avg}
6475	12946	– . –	– . –	– . –	– . –
445	886	30.01	0.05	0.063	0.0061
277	550	32.00	0.1	0.099	0.0134
183	362	33.10	0.2	0.217	0.0282
104	204	34.21	0.5	0.490	0.0681
74	144	35.33	1.0	0.921	0.1291

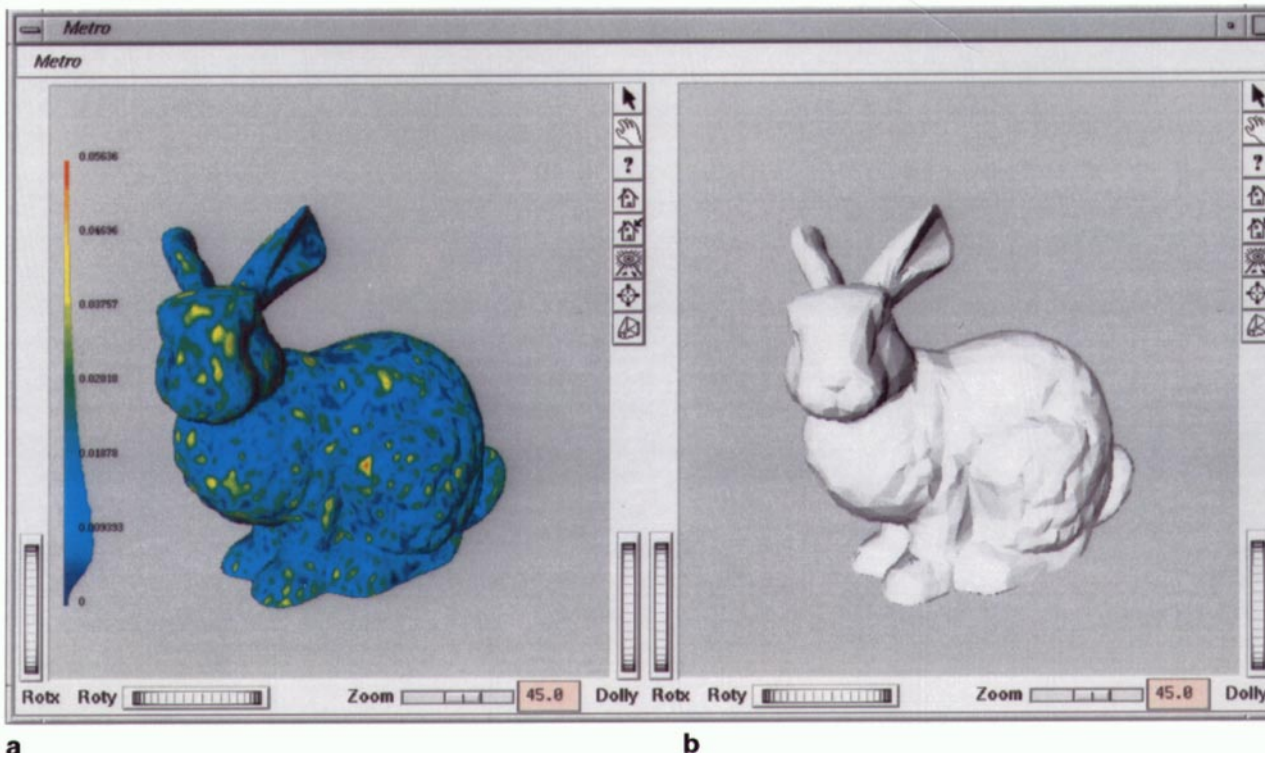


Fig. 9. A simplified mesh produced from the Bunny dataset, produced with a target error of 0.2%, rendered by Metro with color mapping proportional to the approximation error

meshes numerically, Metro adopts an approximated and symmetric approach based on surface sampling and point-surface distance computation. Metro requires no knowledge of the simplification approach adopted to build the reduced mesh. It returns both numerical results and error magnitude visualization.

Tables 4 and 5 report the target error ε^* set by the user and the values returned by Metro after comparing the original mesh and the simplified one: E_{max}, the maximum error and E_{avg}, the average error. All errors reported in the tables are given as percentages of the dataset bounding box diagonal.

One of the functionalities of Metro is the possibility of producing renderings of the measured mesh, where facets are colored and shaded to visualize the magnitude of the local approximation error. An example of this feature, relative to the Bunny dataset is shown in Fig. 9. Please note that the Metro output window contains the histogram of the distribution of error (Fig. 9a). The numbers reported in the histogram are measured as absolute values (therefore, the mesh shown in the figure has a maximal absolute error equal to 0.056, which is 0.2% of the bounding box diagonal). The histogram shows that most of the surface has approximation error lower than one-third of the maximal error.

Figures 7 and 8 present four different meshes extracted interactively from the two multiresolution representations built on the test datasets. The errors reported in the figures are measured as percentages of the dataset bounding box diagonal. The total number of vertices and facets in the multiresolution representation and the number of facets of the mesh currently extracted and visualized are also reported.

8.1 Comparison with other simplification approaches

We compared our algorithm with one of the "fastest" approaches, the mesh decimation

implementation provided in the visualization toolkit (Schroeder et al. 1995), and one of the most "precise" approaches, the original implementation of the mesh optimization algorithm (Hoppe et al. 1993).

A number of runs of the three solutions were done, in order to produce simplified meshes of "similar" sizes (sizes differ by no more than 5–10 triangles).

The results obtained (Tables 6 and 7) show that JADE is half way between them. It takes approximately ten times longer than mesh decimation, but approximately 20–30 times less than mesh optimization.

Given a target number of vertices for the simplified meshes, the error in our solution is from four to ten times lower than mesh decimation, and it is very close to that of mesh optimization. Again, approximation errors were measured by using Metro to compare each simplified mesh with the input mesh.

Times were compared by running JADE and the mesh decimation system on an SGI Indigo2 workstation. The mesh optimization algorithm was run on a Digital 3000/900 workstation (Alpha 64 bit 275 MHz, 128 MB RAM) because only a Digital executable is available in the public domain. Then, Digital times were scaled into "SGI-expected" times (the scaling factor was computed by running JADE on both architectures, on the same datasets). The times relative to the mesh optimization algorithm are therefore Digital-to-SGI scaled times.

In the case of Fig. 7, it may seem contradictory that the simplification at lower precision requires lower times, even if a larger number of vertex removal operations are executed. This depends on the higher frequency of invalid triangulations that are produced when a higher precision threshold is set.

One important advantage of our solution is the measurability of the global approximation error introduced in the simplification. For this reason, a comparison with some extremely recent approach would be interesting, such as the simplification envelopes algorithm (Cohen et al. 1996) or

Table 6. Comparison of simplification algorithms (errors are measured as percentages of the dataset bounding box; times are in seconds)

Stanford Bunny (bounding box $15.6 \times 15.4 \times 12.1$)

		JADE			Decimation			Optimization		
N_{Vert}	N_{Triang}	E_{max}	E_{avg}	$Time$	E_{max}	E_{avg}	$Time$	E_{max}	E_{avg}	$Time$
34834	69451	–.–	–.–	–.–	–.–	–.–	–.–	–.–	–.–	–.–
17145	34120	0.037	0.0039	163.30	0.155	0.0049	22.41	0.307	0.0100	4461
3430	6820	0.137	0.0270	284.60	0.421	0.0226	26.51	0.281	0.0157	4746
590	1160	0.578	0.1438	318.99	2.757	0.3210	28.10	0.467	0.0542	5116

Table 7. Comparison of simplification algorithms (errors are measured as percentages of the dataset bounding box; times are in seconds)

Fandisk (bounding box $4.8 \times 5.6 \times 2.7$)

		JADE			Decimation			Optimization		
N_{Vert}	N_{Triang}	E_{max}	E_{avg}	$Time$	E_{max}	E_{avg}	$Time$	E_{max}	E_{avg}	$Time$
6475	12946	–.–	–.–	–.–	–.–	–.–	–.–	–.–	–.–	–.–
1025	2020	0.0181	0.00114	37.90	0.427	0.0071	4.7	0.307	0.00321	1373
530	1065	0.0465	0.00462	36.29	0.998	0.0280	5.5	0.241	0.00401	1495
280	560	0.0965	0.01260	36.10	1.804	0.0880	5.0	0.295	0.00673	1468
175	325	0.2489	0.03197	35.44	1.916	0.0820	5.2	0.340	0.01374	1391
100	195	0.5444	0.07234	35.76	1.917	0.1690	6.7	0.595	0.02856	1293

the method based on tolerance volumes (Gueziec 1996), which support bounded global error evaluation. However, an empirical comparison is not possible, since implementations of these methods are not available to us.

JADE and the simplification envelopes algorithm are both based on a decimation approach. The simplification envelopes algorithm should be slightly less efficient, due to the cost of the envelope-intersection test executed for each $t' \in T'_v$ and the cost of the expensive preprocessing for computing the envelopes. Nonetheless, although JADE gives a very good estimate of global error, it does not provide an exact bound on maximal error (due to possible under-estimations in global error evaluation; see Sect. 4).

Results obtained on the Bunny datasets are reported in the simplification envelopes paper (Cohen et al. 1996). The simplification rates are comparable to those obtained with our solution, whereas simplification envelope times are twice those of JADE (and the times reported in the paper were measured on an HP 735/125 workstation [136 SPEC92int, 201 SPEC92fp], which is slightly faster than the SGI Indigo2 we used in our runs [140 SPEC92int, 131 SPEC92fp]).

Concerning the method based on tolerance volumes (Gueziec 1996), the results presented in the paper have been obtained on proprietary surfaces, and therefore comparison with our results is not easy. Adopting edge collapsing instead of vertex decimation has the advantage of avoiding the retriangulation of the removed patches. However, the approach adopted in the tolerance volumes method to estimate and bound the approximation error is fairly complex, and it returns an overestimation of the actual error, which may be the origin of a lower simplification ratio.

Another approach that supports bounded error management has been recently proposed by Bajaj and Schikore (1996). We became acquainted with their proposal only during the revision of this paper, and we thank the anonymous referee for that. There are some similarities between this work and our solution. Bajaj and Schikore adopt a vertex decimation approach and estimate a global error bound by accumulating local errors. For each decimation step, the local error is measured by mapping the edges of the new patch on the removed patch and dividing it into pieces. Due to the linear variation of geometry on each piece, local error is evaluated only at the intersections of the edges (simplifying computations) and is used as an upper bound for each triangle in the new patch. They also use edge flipping to improve the quality of the mesh.

8.2 Comparison with other multiresolution approaches

The only two approaches that support multiresolution are multiresolution analysis (Eck et al. 1995) and progressive meshes (Hoppe 1996). Again, an empirical comparison is not possible, because these codes are not available on the public domain.

Following Hoppe's (1996) evaluations, we can say that our solution has a number of advantages over multiresolution analysis. It is a lossless representation, it manages any feature edges or discontinuities in the input mesh with higher precision, and it is much more efficient both in construction of the multiresolution representation and in the reconstruction by remeshing the approximate meshes S_j (which costs tenths of minutes on an SGI Onyx ws).

A comparison with the Progressive Meshes approach is more debatable. Our approach is characterized by:

- Lower simplification times: the progressive meshes approach is based on a revised [and probably slightly more expensive] mesh optimization approach, and we have proved that our solution is much more efficient than the original mesh optimization code. However, the methodology proposed here for vertex selection and mesh update might be also applied to the methods based on edge collapsing.
- Measurability of the global approximation error held by simplified meshes.
- Faster extraction of a single level of detail model S_j from the multiresolution representation S_M: we only need to search for ε-valid triangles, with complexity $O(m \log k)$ and empirical efficiency in the order of 1.5 M facets/s (Sect. 7). However, progressive meshing requires the execution of $O(m - m_0)$ slightly more complex inverse edge-collapsing actions and mesh updates, with m being the number of facets of the mesh S_j and m_0, the number of

facets in the lower resolution mesh contained in S_M.

Our multiresolution scheme, like the progressive meshes representation, can support the progressive transmission of meshes to show progressively better approximations while the high/mid-level mesh is being transmitted on a slow communication line.

An advantage of the progressive meshes approach is the ability to apply geomorphs between different LOD representations, thus increasing visual quality in LOD switching. This is not possible with JADE, due to the current design of the S_M multiresolution data structure. Geomorphing is not possible because, once the S_M representation has been built, we lose the interference relation that, for each simplification step, links the removed patch T_i to the new updating patch T_i'. However, this is not a limitation of our multiresolution approach because alternative designs of S_M that maintain the updating relation ($T_i \leftrightarrow T_i'$) are possible (De Floriani and Puppo 1995). We decided to avoid an explicit representation of the $T_i \leftrightarrow T_i'$ relation to allow a more compact representation.

Other interesting features of the progressive meshes approach are (1) the ability to extract a single mesh at various precisions, a feature that Hoppe calls selective refinement, and (2) both scalar and discrete attributes preservation (e.g., the normals or colors defined on the input mesh). We are extending JADE with the same objectives, as briefly described here.

We recently proposed an approach that provides selective refinements on multiresolution terrain maps (Cignoni et al. 1997b); we designed a data structure that can encode dynamic changes in a triangulation obtained by progressive refinements or by simplification. The structure is based on embedding triangles in 3D space, and it maintains topological relations between pairs of triangles that are adjacent at some time during dynamic construction. The data structure supports efficient traversal algorithms to obtain either a triangulation at some fixed error, or some special triangulations that cover different parts of the domain with triangles with different approximation errors.

We are now extending this approach to generalized surfaces in 3D space. Starting from the multiresolution data produced by JADE, we support the interactive selective refinement/simplification of the mesh (Cignoni et al. 1997a). The domain of this system is resolution modeling or, using a metaphor, geometric painting. Given an input surface and the results of its simplification, the idea is to allow the user to choose a constant approximation level and then, interactively, to modify the mesh by increasing or reducing the precision in selected areas. A logical interface extremely similar to those provided by image painting systems (Haeberli 1990) is used.

Preserving scalar and discrete attributes can be introduced in a decimation-based approach, as well as in an optimization-based one, by choosing an appropriate error metric for vertex removal. We completely agree with Hoppe's considerations regarding the importance of preserving attribute discontinuities (Hoppe 1996), and the extensions we are designing for the JADE code were largely inspired by his work.

8.3 Conclusions

We have presented JADE, a new algorithm for surface simplification. JADE adopts a mesh decimation approach and fulfills three major goals: to minimize error with sustainable processing costs, to provide a bounded error management, and to produce a multiresolution representation at the cost of a low memory overhead.

The observed time complexity is from tenths to hundreds of seconds on medium complexity meshes (composed by tenths of thousands facets). A peculiar characteristic of JADE is that the multiresolution description of the simplified mesh comes free, at no added processing costs, and the extraction of a mesh at a given error ε is extremely efficient and can be computed in real time. Storing a multiresolution mesh requires a very limited overhead in space. Experimental results showed that, on average, it holds at most three times more facets than the single maximal precision mesh.

Acknowledgements. This work was partially financed by the Progetto Coordinato "*Modelli multirisoluzione per la visualizzazione di campi scalari multidimensionali*" of the Italian National Research Council (CNR). An executable (SGI version) of the JADE code is available on the public domain at address: http://miles.cnuce.cnr.it/cg/homepage.html.

References

1. Bajaj CL, Schikore D (1996) Error bounded reduction of triangle meshes with multivariate data. SPIE 2656:34–45
2. Cignoni P, De Floriani L, Montani C, Puppo E, Scopigno R (1994) Multiresolution modeling and rendering of volume data based on simplicial complexes. Proceedings of the 1994 Symposium on Volume Visualization, ACM Press, New York:19–26
3. Cignoni P, Montani C, Puppo E, Scopigno R (1996a), Optimal isosurface extraction from irregular volume data. Proceedings of the 1996 Symposium on Volume Visualization, ACM Press, New York:31–38
4. Cignoni P, Rocchini C, Scopigno R (1996b), Metro: measuring error on simplified surfaces. Technical Report B4-01-01-96, Istituto de Elaborazione dell'Informazione – CNR, Pisa, Italy
5. Cignoni P, Montani C, Rocchini C, Scopigno R (1997a) Resolution modeling. Technical Report C97-02, Istituto CNUCE – CNR, Pisa, Italy
6. Cignoni P, Puppo E, Scopigno R (1997b) Representation and visualization of terrain surfaces at variable resolution. Visual Comput 13 (in press)
7. Cohen J, Varshney A, Manocha D, Turk G, Weber H, Agarwal P, Brooks F, Wright W (1996) Simplification envelopes. Computer Graphics Proceedings Annual Conference Series (SIGGRAPH '96), ACM Press, New York:119–128
8. Criscione P, Montani C, Scateni R, Scopigno R (1996) DiscMC: an interactive system for fast fitting isosurfaces on volume data. In: Gobel M, David J, Slavik P, Wijk J van (eds) Virtual Environments and Scientific Visualization '96, Springer Computer Science, Vienna, pp 178–190
9. Curless B, Levoy M (1996) A volumetric method for building complex models from range images. Proceedings of SIGGRAPH '96, New Orleans, Computer Graphics Proceedings, Annual Conference Series, ACM Press, New York, pp 303–312
10. De Floriani L, Puppo E (1995) Hierarchical triangulation for multiresolution surface description. ACM Trans Graph 14:363–411
11. Deering M (1995) Geometry compression. Computer Graphics Proceedings Annual Conference Series (SIGGRAPH '95), ACM Press, New York:13–20
12. Eck M, Rose TD, Duchamp T, Hoppe H, Lounsbery M, Stuetzle W (1995) Multiresolution analysis of arbitrary meshes. Computer Graphics Proceedings Annual Conference Series (SIGGRAPH '95), ACM Press, New York:173–181
13. Edelsbrunner H (1980) Dynamic data structures for orthogonal intersection queries. Report F59, Inst. Informationsverarb., Technical University of Graz, Graz, Austria
14. Funkhouser T, Sequin C (1993) Adaptive display algorithm for interactive frame rates during visualization of complex environment. Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '93), ACM Press, New York:247–254
15. Gueziec A (1996) Surface simplification inside a tolerance volume. Technical Report RC 20440, IBM, T.J. Watson Research Center, Yorktown Heights, NY
16. Haeberli P (1990) Paint by numbers: abstract image representations. ACM Comput Graph (SIGGRAPH '90 Proc) 24:207–214
17. Hinker P, Hansen C (1993) Geometric optimization. IEEE Visualization '93 Proceedings, IEEE Computer Society Press, Los Aldmitos, CA:189–195
18. Hoppe H (1996) Progressive meshes. ACM Computer Graphics Proceedings Annual Conference Series (SIGGRAPH '96) ACM Press, New York:99–108
19. Hoppe H, DeRose T, Duchamp T, McDonald J, Stuetzle W (1993) Mesh optimization. ACM Computer Graphics Proceedings Annual Conference Series (SIGGRAPH '93) ACM Press, New York:19–26
20. Kalvin AD, Taylor R (1996) Superfaces: polygonal mesh simplification with bounded error. IEEE Comput Graph Appl 16:64–77
21. Kalvin A, Cutting C, Haddad B, Noz M (1991) Constructing topologically connected surfaces for the comprehensive analysis of 3D medical structures. SPIE Image Processing SPIE 1445:247–259
22. Klein R, Liebich G, Strasser W (1996) Mesh reduction with error control. In: Yagel R, Nielson G (eds) Proceedings of Visualization '96, IEEE Computer Society Press, Los Aldmitos, CA:311–318
23. Moore D, Warren J (1992) Compact isocontours from sampled data. In: Kirk D (ed) Graph Gems III, Academic Press, Boston, pp 23–28
24. Muller H, Stark M (1993) Adaptive generation of surfaces in volume data. Visual Comput 9:182–199
25. O'Rourke J (1994) Computational geometry in C. Cambridge University Press, Cambridge, Mass
26. Rossignac J, Borrel P (1993) Multi-resolution 3D approximation for rendering complex scenes. In: Falcidieno B, Kunii T (eds) Geometric modeling in computer graphics. Springer, Berlin Heidelberg New York, pp 455–465
27. Schroeder W (1995) Polygon reduction techniques. ACM Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '95), Course Notes No. 30 (Advanced Techniques for Scientific Visualization), 1.1–1.14
28. Schroeder WJ, Zarge JA, Lorensen WE (1992) Decimation of triangle meshes. ACM Comput Graph (SIGGRAPH '92 Proceedings) 26:65–70
29. Schroeder W, Martin K, Lorensen W (1995) The Visualization Toolkit: an object oriented approach to 3D graphics. Prentice Hall, Upper Saddle River, NJ
30. Turk G (1992) Re-tiling polygonal surfaces. ACM Comput Graph (SIGGRAPH '92 Proceedings) 26:55–64
31. Turk G, Levoy M (1994) Zippered polygon meshes from range images. ACM Comput Graph 28:311–318
32. Wilhelms J, Gelder A van (1994) Multi-dimensional trees for controlled volume rendering and compression. Proceedings of the 1994 Symposium on Volume Visualization, ACM Press, New York, pp 27–34

ANDREA CIAMPALINI is a Research Fellow at the Istituto di Elaborazione della Informazione of the National Research Council in Pisa, Italy. He received an advanced degree (Laurea) in Computer Science from the University of Pisa in 1996.

PAOLO CIGNONI is a Research Scientist at the Istituto di Elaborazione della Informazione of the National Research Council in Pisa, Italy. His research interests include computational geometry and its interaction with computer graphics, scientific visualization, and volume rendering. He received an advanced degree (Laurea) in Computer Science from the University of Pisa (1992), where he is currently a PhD student.

CLAUDIO MONTANI is a Research Director with the Istituto di Elaborazione della Informazione of the National Research Council in Pisa, Italy. His research interests include data structures and algorithms for volume visualization and rendering of regular or scattered datasets. Montani received an advanced degree (Laurea) in Computer Science from the University of Pisa in 1977. He is member of the IEEE.

ROBERTO SCOPIGNO is a Research Scientist at the Istituto CNUCE of the National Research Council in Pisa, Italy. Since 1990 he has had a joint appointment at the Department of Computer Engineering of the University of Pisa. His research interests include interactive graphics, scientific visualization, volume rendering, and parallel processing. He received an advanced degree (Laurea) in Computer Science from the University of Pisa in 1984. He is member of the IEEE.