Surface decimation with global error control and multiresolution output management

A. Ciampalini*, P. Cignoni*, C. Montani*, R. Scopigno†

I.E.I.-C.N.R., CNUCE - C.N.R.

ABSTRACT

The paper presents the first results of our project on surface simplification. A new solution, JADE (Just Another DEcimator), is presented. It enhances the Mesh Decimation approach by providing better approximation precision and multiresolution output management. Results are reported on empirical time complexity, approximation quality, and simplification power. Moreover, we show that with a small increase in memory, which is needed to store the multiresolution data representation, we are able to extract at run time any level of detail representation in an extremely efficient way.

CR Descriptors: I.3.3 [**Computer Graphics**]: Picture/image generation - *Display algorithms*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling - *Curve, surface, solid and object representation*; I.3.6 [**Computer Graphics**]: Methodology and Techniques.

INTRODUCTION

Surface simplification is a very hot topic in visualization, for several application–driven motivations. In fact, huge surface meshes are produced in a number of fields:

- volume visualization: isosurfaces composed of millions of facets can be extracted on high resolution volume datasets;
- virtual reality: very complex models have to be generated to cope with the complexity of real environments, e.g. to build high quality *walk throughs* in virtual worlds;
- automatic modeling: the technology for the automatic acquisition of the shape of 3D objects is emerging (e.g. 3D range scanners), and the high precision produces very complex meshes (in the order of hundreds of thousands of facets);
- free form surfaces: optimized polyhedral approximations of parametric surfaces produce highly detailed meshes.

*Istituto per l'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.

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 go over the sustainable memory/performance of the current mid level graphics subsystems.
The interest is proved by the large number of research groups which started projects on this topic and the many papers published in recent years. Among others, Univ. of Washington [7, 5], General Electric [10], Univ. of North Carolina at Chapel Hill [12, 4], and IBM Yorktown Research Center [8, 11] have obtained significant results in this field.

The goals of our activity on surface simplification are:

- **approximation error**: accurate estimation of the approximation error introduced in the simplification process;
- **compression factor**: provide a reduction factor, #(simplif_mesh)/#(orig_mesh), comparable or better than other approaches, under the same level of approximation;
- **multiresolution management**: once the simplification process has been run, we want to make possible the interactive extraction of any LOD (Level Of Detail) representation, with the complexity of a single LOD extraction linear to the LOD output size.
- **working domain**: wide generality, the algorithm should not rely on the correctness of the surfaces (self intersecting, not manifold, not orientable surfaces are common on the real world data);
- **space/time efficiency**: short simplification times and low memory consumption, to allow large meshes management.

We consider the third point one of the most significant in our approach. We want to provide a real *multiresolution representation*. To clarify this point, we must first give a characterization of two terms which are sometimes considered synonymous. Given a surface S, we define:

- *LOD* representation of S: a data structure which holds a *constant* number k of different representations of the surface, at different levels of detail or approximation;
- *multiresolution* representation of S: a data structure which allows the compact representation of $O(f(n))$ representations at different levels of detail/approximation, with n the data size (e.g. the number of faces in S) and $f(n)$ an increasing monotone function.

LOD representations are widely used in a number of leading edge applications (e.g. virtual reality, VRML etc.). A multiresolution representation improves over LOD with valuable characteristics: the user or the application have much more flexibility in the selection of the "best" level of detail; in many cases it is better to leave that choice at run time, instead of forcing it at the time of the preprocessing simplification phase.

The structure of the paper is as follows. At the heart of our solution is a kernel based on the Mesh Decimation approach. We therefore briefly introduce this algorithm in the next section. Then, we define the global approximation error criterion adopted. The triangulation heuristics and algorithms used are described in the fourth section. Multiresolution management of the output data is presented in the fifth section. The last section outlines the results and conclusions.

MESH DECIMATION

The Mesh Decimation method [10] reduces mesh complexity by applying multiple passes over the triangle mesh and using local geometry and topology to remove vertices that pass a distance or angle criterion. The resulting holes are then patched using a local triangulation process (3D recursive loop splitting). A characteristic of the method is that the simplified mesh has vertices which are a subset of the original ones.
The criterion is based on the *local error* evaluation. Each vertex v is classified topologically (looking at the loop of facets incident in v). For each vertex v which 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 in 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 semi–cycle 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 the vertex removal and the associated re–triangulation; on the contrary, 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 which affect the same mesh area may produce an accumulated approximation error much higher than the maximum allowed.
On the other hand, significant sharp edges are maintained by adopting a *threshold* for the maximal *solid angle*, which prevents vertex elimination. The power of Decimation is its time efficiency, which is higher than other approaches. The method supports the construction of LOD representations by means of k successive iterations of the Decimation process, under different error threshold settings.

GLOBAL ERROR CONTROL

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 in v; and, finally, the new triangulation T_v' which will replace T_v after the elimination of v, the error introduced may be measured in terms of:

- **local error**: measures the local approximation introduced by replacing T_v in S_i with T_v', and generating S_{i+1}; e.g. the local error may be estimated as the distance of v from T_v' [10];

- **global error**: measures the error of approximation introduced if the new mesh parcel T_v' is introduced to represent the corresponding sub–area of the original input mesh S.

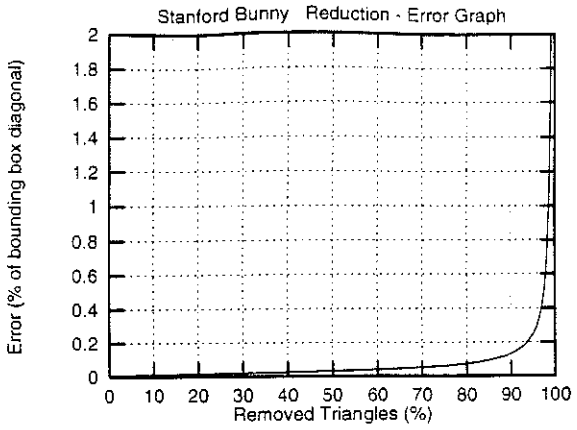

Figure 1: The approximation error

Providing a *global* error management is crucial to give the user an accurate control of the approximation error introduced. It is also crucial to allow an easy management of multiresolution, as we will clarify in the Multiresolution section.
Taking into account the global error may be extremely costly if we require a precise evaluation. On the other hand, a number of efficient heuristics may be used to approximate the global error.

A first solution is to accumulate the errors introduced in each simplification step: we take into account the error accumulated on each facet of the mesh. Initially, error is set to null for each triangle $t \in S$; at each time step, the global error of the current mesh s_i is the max of the global error of the facets in s_i. For each simplification action, the global error assigned to each $t' \in T_v'$ is computed as the sum of the current local error introduced by the removal of vertex v and the biggest of the global errors held by the facets in the removed patch T_v.

Candidate Vertex Selection

The strategy to select the vertices that are candidate for removal is analogous to the Mesh Decimation algorithm as far as the topological classification of vertices is concerned. On the other hand, the *order* in which vertices are decimated is unique to our solution. Instead of processing vertices in a random order, we sort candidate vertices in order of increasing accumulated global error and process them following this order.
Moreover, because each simplification action modifies the topology, the sorted vertex list has to be updated after each decimation step (classification and local error computation, limited to all the vertices on the border of T_v').
The decimation order chosen has three positive effects: (a) approximation error increases slower, and the quality of the obtained mesh improves; (b) at each step the mesh approximation (defined as the max of the global errors of its facets) increases; (c) a smooth error growth ensures the quality of the models that can be extracted from the multiresolution representation.

BEST FITTING TRIANGULATION

After the removal of a vertex v, the hole resulting from the absence of the patch T_v of triangles must be retriangulated. To get the best approximation of the removed patch T_v, and to reduce the error introduced in each individual simplification step, some more processing has to be carried out on the obtained triangulation T_v'. How

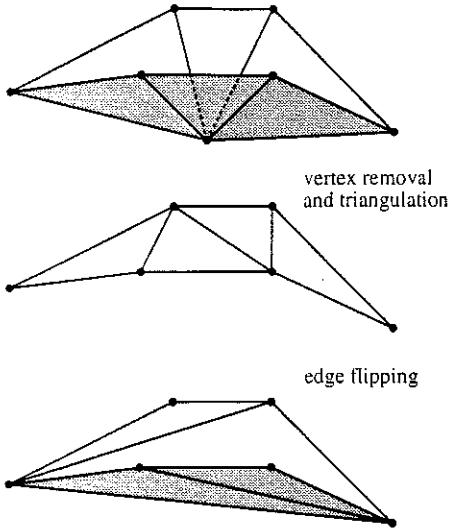

Figure 2: The example shows how flipping may improve mesh quality.

much the triangles in T_v' actually approximate the removed mesh has to be carefully evaluated (see Figure 2).
We increase the quality of the approximation through a series of *edge flip* actions: on each internal edge e of T_v', we evaluate whether the new pair of adjacent triangles obtained by flipping e give a better approximation with respect to the original mesh S and, if they do, we perform the flip. A number of different heuristics have been devised to drive flipping.

Flipping has proved to be crucial in increasing the quality of the reduced mesh.

MULTIRESOLUTION REPRESENTATION

Given a decimation process, we want to produce in output a multiresolution representation S_M, such that given any precision threshold ε the retrieval of an approximate model which satisfies precision ε should be allowed 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 is characterized by two time stamps: its creation (or *birth*, i.e. when t is generated as part of a new patching sub-mesh) and its elimination (or *death*, i.e. when t is found as one of the triangles incident on a vertex candidate for removal).

These time stamps have a direct connection with the approximation error of the intermediate meshes S_j of which t is part.
Each facet 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, with their ε_b and ε_d errors explicitly stored for each facet t.
The extraction of a given precision ε representation out of S_M is therefore straightforward: S_M contains sufficient information to reconstruct it. S_ε is composed of all of the facets in S_M such 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 is to use an interval tree [6] to retrieve all the facets whose life intervals (i.e. ε_b and ε_d) covers ε. 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 showed us that the use of an interval tree permits us to interactively extract approximated models from the Bunny (see next section for details on the dataset) multiresolution representation with a constant per-triangle rate of the order of 1.5 M triangles per sec.

Stanford Bunny (Bounding Box 15.6x15.4x12.1)					
JADE				Metro Eval.	
N_{Vert}	N_{Triang}	T_{Sec}	E_{ext_gl}	E_{max}	E_{avg}
34,834	69,451	-.-	-.-	-.-	-.-
4,707	9,221	381.75	0.10	0.121	0.021
2,113	4,064	411.42	0.20	0.255	0.047
781	1,452	422.32	0.50	0.539	0.120
499	934	424.86	0.75	0.914	0.183
341	653	426.22	1.00	1.362	0.252

Table 1: Bunny Dataset: numerical evaluation of simplified mesh quality.

Four snapshots of a simple tool which allows the interactive extraction and visualization of different levels of detail out of the multiresolution representation are shown in Figure 3.

Experimental results showed 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), although a high number of different resolutions are available.

The *shape* of the curve of graph in Figure 1 is important. While performing decimation, the approximation error growth should be slow and smooth to ensure that all the models extracted from a multiresolution representation would guarantee a good reduction factor.

A similar multiresolution approach has also been used for managing volume datasets (adopting a simplicial representation) [1].
For a deeper analysis on the use of interval trees in computer graphics see [2].

RESULTS AND CONCLUSIONS

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

In order to evaluate both simplification rate and quality, we ran a number of tests on a public domain dataset, representing a "bunny" acquired by a range scanner[1]; the bunny original mesh is composed by 34,834 vertices and 69,451 triangles. In Table 1 we present the precision obtained in a number of runs of our code. Times are in CPU seconds and measure the cost for the construction of a multiresolution representation which stores all approximations in the error range 0.0 – E_{ext_gl}. The approximation error was evaluated using **Metro**, a tool we developed to measure the difference between meshes [3]. The table reports E_{ext_gl}, the *estimated global error* returned by JADE, and values measured using Metro to compare the original mesh and the simplified one: E_{max}, the max error; E_{avg}, the average error. All errors reported in the table are given as a percentage of the dataset bounding box diagonal.

In Figure 1 we show how the approximation error increases (following a very smooth curve) during the simplification process.
In Figure 3 four meshes are interactively extracted and rendered; the errors reported in the figure are measured as percentages of the dataset bounding box diagonal. In the snapshots are also reported the total number of vertices and facets in the multiresolution representation (*Total Vertices* and *Total Faces*) and the number of facets of the mesh currently extracted and visualized (*N. ofFixed Err. Faces*).

[1] The original model was created from laser range data using Turk and Levoy's mesh zippering algorithm [13]