Hierarchical 3D mesh coding with subdivision surfaces

Francisco Morán

Grupo de Tratamiento de Imágenes, Universidad Politécnica de Madrid
E.T.S. Ing. Telecomunicación — 28040 Madrid, Spain
E-mail: F.Moran@gti.ssr.upm.es; Tel.: +34.913.367.353

ABSTRACT

We build on state-of-the-art methods for multiresolution embedded coding of images, such as the zerotree coding of wavelet coefficients, and combine them with ideas for hierarchical modelling of 3D objects using subdivision surfaces, to obtain a new technique for 3D mesh compression. The compression ratios we obtain are better than or similar to previously reported ones but, perhaps more importantly, the "truly" hierarchical coding of 3D objects we propose allows their efficient multiresolution editing and animation.

1. INTRODUCTION

Subdivision surfaces [8] are defined as the limit of a recursive refinement process of both the connectivity and the geometry of a control mesh by splitting each of its polygons into several ones. The successive control meshes obtained while iterating this process are inherently nested, and therefore very useful to create multiresolution models of 3D objects by approximating their bounding surfaces, because they define a pyramid of LODs (Levels Of Detail) highly suitable for its progressive coding.

The simplest way to approximate a 3D surface is to tile it with flat polygons — usually triangles. 3D triangular meshes are becoming increasingly popular because inexpensive laser-based 3D scanning systems that output triangular meshes are increasingly widespread. Also, most CAD/CAM applications "understand" them, and even ISO standards for 3D object coding (VRML, MPEG-4) support them. But they are only piecewise linear approximations to arbitrarily complex surfaces and can thus lead to unacceptable approximation errors unless their number of triangles/vertices is arbitrarily large. One may end up having to deal with meshes of tens of thousands of triangles, which are extremely expensive to handle, store or transmit.

The most widely used primitives for modelling smooth surfaces are probably still NURBSs (Non-Uniform Rational B-Spline patches), which generate piecewise polynomial/rational surfaces from relatively small sets of control points. But one cannot model objects of non-planar topology with them without having to face cumbersome patch stitching and curve trimming problems. Unlike the ones of NURBSs, the control meshes of subdivision surfaces can have any topology at no extra cost. Nevertheless, if the subdivision method is carefully designed, the control mesh converges to a limit surface as smooth as a piecewise polynomial/rational one is. In fact, subdivision surfaces can generalise NURBSs and handle sharp features (corners, creases, etc.) with more ease.

Several subdivision schemes have been proposed and studied, one of the main differences between them being whether all vertices of all the successive control meshes obtained throughout the iterative process belong necessarily to the limit surface or not. Interpolatory schemes, like the butterfly one devised by Dyn et al. [1], are those for which the limit surface interpolates all vertices of all control meshes; in the case of non-interpolatory ones, all vertices need not lie on the limit surface. Both kinds of schemes could be useful for the purposes of LODs generation and hierarchical mesh coding, although we focus our study on the interpolatory butterfly scheme. Even its simplest version yields C^1-continuous limit surfaces in the topologically regular setting, that is, in regions of the initial control mesh whose vertices have valence six. Elsewhere, the limit surfaces may have sharp features, but those can be easily avoided, if needed, by modifying slightly the subdivision rules.

2. PREVIOUS RELATED WORK

The problem of automatic LODs generation for 3D mesh progressive coding is that of finding how to approximate a given target surface described by a very fine mesh, by coarser and coarser meshes, which differ more and more from the target surface according to some distance measure. This is frequently done by performing a series of edge col-

Work partially done within the MOMUSYS (MObile MUltimedia SYStems) project of the EU and funded by the CICYT (*Comisión Interministerial de Ciencia Y Tecnología*) of the Spanish Government.

lapses or vertex removals starting from the given finest LOD. The reconstruction phase, in which the finest mesh is recovered from the coarsest one, consists of a series of operations that are the inverse of the ones performed in the coarsening phase: vertex splits for edge collapses, and vertex insertions in the case of vertex removals. Although both these approaches certainly produce progressive meshes, in the sense that the finest mesh can be progressively recovered from the coarsest one, their LODs cannot be said to be hierarchically nested: they do not form a pyramid of LODs, as the ones produced by a subdivision approach do.

Other than the prior art on what we call "truly" hierarchical coding and review briefly in the next section, we must mention here the excellent work on progressive 3D mesh coding already reported. Version 2 of ISO's MPEG-4 standard [5], that will hopefully be released by the end of 1999, contains tools for the efficient progressive coding of 3D meshes based on edge collapses and vertex splits. On the other hand, the technique based on vertex removals and insertions by Li and Kuo [4] has the advantage of yielding a fully embedded code.

3. HIERARCHICAL CODING

One of the properties of subdivision methods is that the surfaces associated to the succesive control meshes are all homeomorphic (i.e., topologically equivalent) to the surface represented by the base mesh. It is important to make explicit that, unfortunately, as two homeomorphic surfaces can be represented by meshes of different triangle connectivities, it is rarely straightforward to extract a base control mesh from a given target mesh by making four-to-one merges of its triangles. In fact, the target mesh has in general to be remeshed to have it follow the simple subdivision connectivity rules, but automatic procedures to do it have been described [2]. The problem of automatic LODs extraction in the approach based on subdivision surfaces is then, basically, that of adequately remeshing the target surface.

Once a base control mesh has been extracted from the given target mesh, eventually remeshed, the subdivision scheme can be repeatedly applied upon it some number of times in the hope of recovering the target mesh. But, of course, one cannot expect the new vertices that appear at each stage of the subdivision process to lie precisely on the target surface. Fortunately, nothing prevents one from correcting the positions of these new vertices after the standard subdivision rules have misplaced them, and to use the resulting mesh as the input for the next step of the subdivision process.

It is easy to imagine then a hierarchical 3D mesh transmission scenario, in which an initial, coarse mesh has already been transmitted somehow (possibly following the MPEG-4v2 standard, for instance) and is taken as the base mesh for the subdivision process, so only the details to be added to the new vertices need to be sent. If both encoder and decoder have previously agreed upon a set of subdivision rules, the details can be considered to be prediction errors, because they measure the difference between the predicted vertices, that would result from the normal subdivision process, and the real ones, that lie on the target surface.

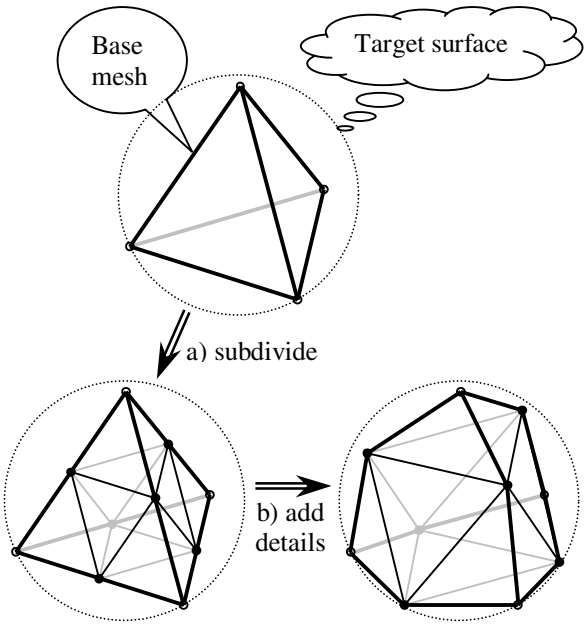

Figure 1: Subdividing a tetrahedron with a sphere in mind (first step of the process).

Figure 1 illustrates this idea: after subdividing the base mesh by splitting each of its triangles into four, details are added to the new vertices to have them also lie on the target surface, and the whole process can be repeated upon the resulting mesh.

The more obvious advantage of transmitting these details, instead of the full vertex positions, is the one any predictive coding scheme would offer: a more compact code can be obtained by using the prediction errors rather than the predicted values themselves, since the former have a smaller variance than the latter. Moreover, a smoothing subdivision scheme like the butterfly one will generally produce, for smooth target surfaces, smaller and smaller prediction errors (and, therefore, more and more efficient code), as the subdivision progresses.

A generic predictive coding scheme, although being advantageous, would not make any use of the organisation of the set of details, which is inherently hierarchical. This can be exploited for higher coding efficiency in a way similar to that used first by Shapiro [9] with his zerotrees, and later by Said and Pearlman [6] with their SPIHT (Space Parti-

tioning In Hierarchical Trees), to perform a fully embedded coding of the wavelet transform of the "signal": an image in their cases, a 3D mesh in ours. Schröder and Sweldens [7] used a family of second-generation wavelets to represent functions on a particular 2-manifold: the sphere. Kolarov and Lynch [3] adapted the SPIHT algorithm to compress scalar functions defined on any 2-manifold. They state explicitly that, although their work focusses on the representation of the information distributed about a surface, it is clear that their techniques are also applicable to the description of its geometry itself, which is what we do.

4. PROPOSED TECHNIQUE

4.1. Details quantization and hierarchy

Given a finest LOD target mesh of arbitrary triangle connectivity, we obtain through edge collapses a base mesh that will be the coarsest of our pyramid of LODs. Then we subdivide it some number of times and displace the new vertices introduced by each step of the process to have them lie on the initially given mesh. If **v** is the vertex corresponding to the midpoint of the edge **e** shared by triangles T_A (of oriented normal $\mathbf{n_A}$) and T_B ($\mathbf{n_B}$), its displacement or detail $\mathbf{d_v}$ is expressed in a local frame whose unit vectors point in the directions of **n** (the area-weighted sum of $\mathbf{n_A}$ and $\mathbf{n_B}$), **e** and **n**×**e**. It is precisely by expressing $\mathbf{d_v}$ in a local frame tied to the coarser level mesh that the editing or animation of the surface at different scales is possible.

$\mathbf{d_v}$ is then uniformly quantized with a total of 16 or 32 bits for its three components, depending on the required resolution for details. As, rather logically, the component that carries the most energy is the one aligned with **n**, we assign it half the bits and have the other two share the rest. This quantization may seem too crude, but in our experience the visual effect of the least significant bits of high resolution details (32 bits per detail) on the reconstructed mesh is completely imperceptible.

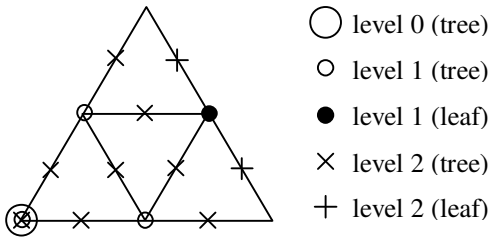

Figure 2: Hierarchical organization of details.

The hierarchical organization of the details is shown in Figure 2. For each base mesh triangle, a vertex is arbitrarily (but consistently in both encoder and decoder) chosen to cover it and determines which detail trees will be "sterile": those attached to the vertices that may appear on the opposite edge of the base mesh triangle, that have been marked as "leaf" in Figure 2. In that way, any "fertile" detail tree rooted at a given vertex covers the coarsest triangle that vertex belongs to, except for the other two vertices of that coarsest triangle.

The only exception to the rules described above has been introduced to avoid duplicating the details of vertices corresponding to the edges of the base mesh triangles. The ordering of the base mesh triangles is used to determine which of the detail trees of two neighbouring triangles will cover the vertices corresponding o their shared edge.

4.2. Coding algorithm

We have followed closely Said and Pearlman's SPIHT algorithm [6], except for the fact that our details are unsigned quantities used to represent the possibly assymetric interval $[d_{min}, d_{max}]$. It consists of a series of steps, each corresponding to a bit-plane, and each divided into a clever "sorting pass", in which some details are classified as "significant" (relative to the considered bit-plane), and a "refinement pass", in which the actual bits of these significant details are output/input.

4.3. Details file/bitstream

This is the structure of the details file/bitstream:

0. Header (33 bytes): scaling information, d_{min} and d_{max} for normal and tangent components, number of subdivisions, number of bit-planes *N*;
1. bit-plane *N*-1: sorting bits + refinement bits;
2. bit-plane *N*-2: sorting bits + refinement bits; ...

N. bit-plane 0: sorting bits + refinement bits.

Except for the header, that could also contain a compressed description of the base mesh, this code is fully embedded.

5. RESULTS AND CONCLUSIONS

It is difficult to compare the compression ratios we obtain to previoulsy reported ones, as the first thing that we require is that the finest LOD mesh be remeshed to have subdivision connectivity. But we have tried, for each given target mesh, to obtain a base mesh with a number of triangles that would yield a remesh of approximately the same number of triangles than the original one after some number of one-to-four splits. We have adopted Li's [4] estimate of an uncompressed mesh size: supposing 4 bytes are needed for each of the three floating point coordinates of each vertex, and also for each of the three pointers to vertices of each triangle, and as in typical meshes there are approximately two triangles per vertex, we may write:

$$raw\ size = (4\times3\times(nvtx+ntri) \cong 18\times ntri) \text{ bytes.}$$

model	target mesh		base mesh		details	
	ntri	*size*	*ntri*	*size*	*nsd*	*size*
cow	5.7K	102K	802	14K	1	10.6K
dinosaur	10K	180K	607	10.7K	2	15K
fist	9.6K	173K	154	2.7K	3	13.9K
sphere	1024	18K	4	96	4	926

Table 1: Compression ratios obtained (*ntri* and *nsd* are the number of triangles and subdivisions, resp.; all sizes are in bytes; details are entropy uncoded).

Table 1 shows some compression ratios achieved with our technique. We do not want to hide that the size of the base mesh must be added to that of the details, because both are needed by the decoder to reconstruct the target mesh. But those small base meshes of several hundred triangles could perfectly be coded using MPEG-4v2 or Li's techniques.

Also, in order to compare comparable things, it must be stressed that all detail sizes are given for entropy uncoded detail files: one can expect to obtain much higher compression ratios by feeding them into an arithmetic encoder (although it could not be used for progressive transmission, `gzip` reduces their size to around ¼ on the average).

The technique for hierarchical coding of 3D meshes based on subdivision surfaces that we propose allows to obtain high compression ratios (in the order of 10-20:1) that are better than or similar to previously reported ones. The code is fully embedded, and every single bit in the transmitted stream may be used by the decoder to reduce the reconstruction error.

Nevertheless, perhaps the main advantage of this kind of "truly" hierarchical representation of 3D objects, and one of its most important foreseeable applications is the ability to help solve the problem of 3D animated mesh coding. The ISO, for example, will have shortly two versions of both VRML and MPEG-4, all of which deal excusively with the coding of static 3D models. It is clear that, in future versions of those standards, 3D dynamic mesh coding will have to be addressed. And the same points that Zorin et al. [10] bring up to justify the power of wavelet-based representations to handle interactive multiresolution mesh editing can be used to explain why animation at different scales is very easy to achieve efficiently with them.

6. FUTURE WORK

Up to now, only systematic subdivision of all triangles of the successive meshes has been described, but, of course, further savings can be obtained by having the subdivision mechanism be adaptive instead. In regions of the mesh that have little detail, the prediction errors will fall below a certain threshold, so the time and space/bandwidth costs associated with the subdivision can be spared by simply not subdividing certain triangles. Given that the main purpose of the proposed coding technique for 3D meshes is their final rendering, adaptiveness should take into account the perceived result and should probably be based mostly on local surface curvature.

A price will have to be paid, however, for the efficiency benefits of adaptive subdivision, on which we are currently working: that of having to fix the cracks which may appear in the rendered surface wherever the mesh is non-conforming, because two adjacent triangles have been subdivided up to different levels. Fortunately, those potential cracks can in fact be easily taken care of locally if the "generational gap" between any two adjacent triangles is never deeper than one.

7. REFERENCES

[1] N. Dyn, D. Levin and J. Gregory, "A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control", *ACM Transactions on Graphics*, 9(2): 160-169, 1990.

[2] M. Eck et al., "Multiresolution Analysis of Arbitrary Meshes", *ACM SIGGRAPH (ACM Computer Graphics Proceedings)*, 1995.

[3] K. Kolarov and W. Lynch, "Compression of Scalar Functions Defined on 2-Manifolds", *Interval Research (internal report)*, 1996.

[4] J. Li and C. Kuo, "Progressive Coding of 3D Graphic Models", *Proceedings of the IEEE*, 86(6): 1052-1063, 1998.

[5] MPEG, "14496-2/PDAM1 — a.k.a. MPEG-4 Version 2, Visual part", *ISO (internal Comittee Draft)*, expected to be released as a Draft International Standard at the end of 1999.

[6] A. Said and W. Pearlman, "A New, Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees", *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3): 243-250, 1996.

[7] P. Schröder and W. Sweldens, "Spherical Wavelets: Efficiently Representing Functions on the Sphere", *ACM SIGGRAPH*, 1995.

[8] J. Schweitzer, "Analysis and Application of Subdivision Surfaces", *Ph.D. thesis, Univ. of Washington*, 1996.

[9] J. Shapiro, "Embedded Image Coding Using Zerotrees of Wavelet Coefficients", *IEEE Transactions on Signal Processing*, 41(12): 3445-3462, 1993.

[10] D. Zorin, P. Schröder and W. Sweldens, "Interactive Multiresolution Mesh Editing", *ACM SIGGRAPH*, 1997.