Progressive Compression of Arbitrary Triangular Meshes

Daniel Cohen-Or David Levin Offir Remez

Computer Science Department
School of Mathematical Sciences
Tel Aviv University
Tel Aviv 69978, Israel
{daniel, levin, offirr}@math.tau.ac.il

Abstract

In this paper we present a mesh compression method based on a multiresolution decomposition whose detail coefficients have a compact representation and thus smaller entropy than the original mesh. Given an arbitrary triangular mesh with an irregular connectivity, we use a hierarchical simplification scheme, which generates a multiresolution model. By reversing the process we define a hierarchical progressive refinement process, where a simple prediction plus a correction is used for inserting vertices to form a finer level. We show how the connectivity of an arbitrary triangulation can be encoded efficiently by a coloring technique, and recovered incrementally during the progressive reconstruction of the original mesh.

Keywords: compression, streaming, progressive meshes, simplification

1 Introduction

The most common representation of 3D geometric models is triangular meshes. Although they have had prominent representation in computer graphics for a long time, only recently has more attention been devoted to their compression [4, 19, 14, 20, 8, 18]. With the increasing popularity of Web-based applications, compression and streaming techniques are today more important than ever. The rich knowledge available for data compression, and in particular for images [15], cannot be directly applied to triangular meshes, mainly because arbitrary triangular meshes have no regular structure, as is the case in image data for example.

Mesh compression algorithms are required to compress both the *geometry* data and the *connectivity* data. The geometry is represented by the set of coordinates of the mesh's vertices. To enable effective compression of the geometry, the coordinate values are first quantized to a fixed number of bits. The connectivity data is the vertex/triangle adjacency list, sometimes also referred to as the *topology*. In a naive representation, the connectivity data is about twice as large as the geometry data. Mesh compression algorithms are normally required to use a lossless compression of the connectivity data.

Current mesh compression methods are based on the triangle-strips technique [4, 1, 7]. In [19, 20, 14, 8] the triangular mesh is traversed along sequences of triangles, which look like peeled strips. The vertices along the strips are encoded as displacement vectors from a point extrapolated from previous vertices along the strips. Touma and Gotsman [20] and independently Rossignac [14] developed methods that compress the connectivity data to less than 2 bits per vertex on average. However, a successful compression of the connectivity alone is not enough since then the compression of geometric data dominates the results. Indeed, Denny and Sohler [5] have shown that the connectivity of a given graph can be encoded in zero bits by a permutation of a sufficiently large set of vertices. This is an interesting theoretical result; encoding an arbitrary permutation of coordinates is, however, too expensive for a space-efficient encoding of a mesh.

Current mesh compression methods are basically "flat", or can be applied to refine one discrete level of detail to the next [18]. They do not have the desired property of a progressive mesh [9], in which every prefix of the encoded data is a progressive approximation of the original 3D shape. The progressiveness property is important to compensate for low network bandwidth and transmission latency. The compression method presented here is based on a multiresolution decomposition, which inherently has a progressive property. However, unlike the progressive meshes, here the size of the data that is required to faithfully recover the original mesh is comparable to any known technique of mesh compression. Recently, a new progressive meshes technique has been developed [13], where a batch of vertex-split operations are encoded efficiently to yield a compressed progressive mesh representation.

2 Multiresolution Decomposition

Multiresolution analysis and wavelets have matured as a versatile tools for representing functions and analyzing features at multiple levels of detail. In recent years they gained attention in the computer graphics community, among various applications, also as a mechanism to analyze 3D meshes [12]. The basic idea is to recursively filter, or decompose a given shape into a lower resolution segment and a higher resolution detailed segment (known as detail coefficients). The motivation is that the low-resolution coarser versions of the shape are a good approximation of the original shape, while the detail coefficients locally perturb the shape to contribute the small details. If the decomposition is successful, many of the detail coefficients are small and their contribution to the final shape is minor. Many image compression methods are based on a thresholding scheme that eliminates the insignificant detail coefficients [15].

Multiresolution analysis methods for the compression of 3D meshes have been applied only in terms of the number of triangles representing the mesh at various levels of detail. However, lossless compression methods of 3D meshes, which compress in terms of the total number of bits required to represent the mesh have not been reported. In this paper we show a multiresolution analysis for which the representation of a given mesh is compact at every level of detail, and in particular for the original mesh.

Traditional wavelets are constructed over regular structures. The construction of wavelets over arbitrary meshes is currently an interesting challenge [10]. Pioneering work by Lounsbery et al. [12] introduced subdivision wavelets defined over arbitrary surfaces, but requires the mesh to have a *subdivision connectivity*. To overcome this prerequisite Eck et al. [6], and recently also Lee et al. [11] have developed remeshing techniques by which the arbitrary mesh can be retriangulated into a subdivision connectivity, where the re-

fined mesh is guaranteed to converge to the original shape within a specified tolerance. Yet, this two-step technique does not provide the means to fully restore the original mesh. It should be emphasized that mesh compression techniques are required to restore the original connectivity. Indeed, most of the efforts of the mesh compression methods have been invested in a compact encoding of the unstructured connectivity of the triangulation [14, 20, 8, 18].

In this paper we show a lossless compression method based on a multiresolution decomposition where the detail coefficients have a compact representation and thus smaller entropy than the original mesh. Similarly to [10], we use a hierarchical simplification scheme, which generates a multiresolution model of the given triangular mesh. By reversing the process we define a hierarchical progressive refinement process, where a simple prediction plus a correction is used for reconstructing vertices to form a finer level. Furthermore, the connectivity of triangulation is encoded efficiently and recovered incrementally during the progressive reconstruction of the original mesh.

3 Prediction and Vertex Insertion

Assume that we start with an abstract set M_n of samples. The set can be partitioned into two sets M_{n-1} and W_n, which are referred to as wavelet subsets. By assembling the two subsets, the original set M_n can be recovered. Using the correlation that exists in the set, one can try to predict the set W_n using an abstract predictor P, in the hope that $P(M_{n-1})$ is close to W_n. Then, we can define new wavelet coefficients as $w_n = W_n \ominus P(M_{n-1})$. Now, the original set M_n is reassembled by $M_n = M_{n-1} \oplus (P(M_{n-1}) \oplus w_n))$. By iterating this process the original set can be represented by the set $\{M_0, w_n ... w_1\}$. If the prediction is successful, the coefficients w_n have a small magnitude and the new representation has a smaller entropy than the representation of the original set M_n. Such a prediction technique is used as one of the building blocks in the *lifting scheme* [17] as well as in multigrid methods [3].

Assuming the original mesh has a subdivision connectivity, different subdivision schemes can be used as good prediction operators. The original mesh M_n can be decomposed into M_{n-1} and W_n by applying some decimation algorithm over its vertices [16], where W_n consists of the set of removed vertices, and M_{n-1} is the simplified mesh. Then by interpolating over the triangles of M_{n-1} we create a set of points $P(M_{n-1})$, which serves as a prediction for the set W_n. The displacement vectors between the removed vertices and the interpolated points are the shorter coefficients w_n.

The key idea is to construct a multiresolution of an arbitrary mesh with irregular connectivity. Unlike traditional wavelets, here the domain is unstructured, and therefore the refinement is not applied uniformly during the reconstruction stage. An interpolation scheme predicts a point to which we add a displacement vector to recover a vertex p. This new vertex is inserted into the triangulation while restoring its connectivity in M_{i+1}. Note that recovering the original connectivity is necessary to correctly decode the data encoded over the representation of M_{i+1}. This will become clearer in the next section where we show how to encode and decode a given mesh.

4 Encoding and Decoding

The series of vertex insertion operations, which reconstruct a given mesh, is found by reversing a mesh decimation procedure [16]. Given a mesh M_i we apply a simplification algorithm that iteratively removes sets of vertices u_i to yield a simplified version M_{i-1}. However, at each iteration the selected set u_i must be an *independent set* [2], that is, there is no edge connecting any two vertices in u_i. Removing a vertex from a triangulation requires removing all the edges connected to the vertex and retriangulating the hole with a new set of triangles. Let us define the triangles that cover a given hole as a *patch*. Once all the holes are triangulated, patches are interpolated to predict a set of points, which serves as a base for the displacement vector to the removed vertices. The predicted points are quantized so the displacement vectors can be represented by a small number of bits, with smaller entropy than the original vertices. For each patch, one displacement vector is stored.

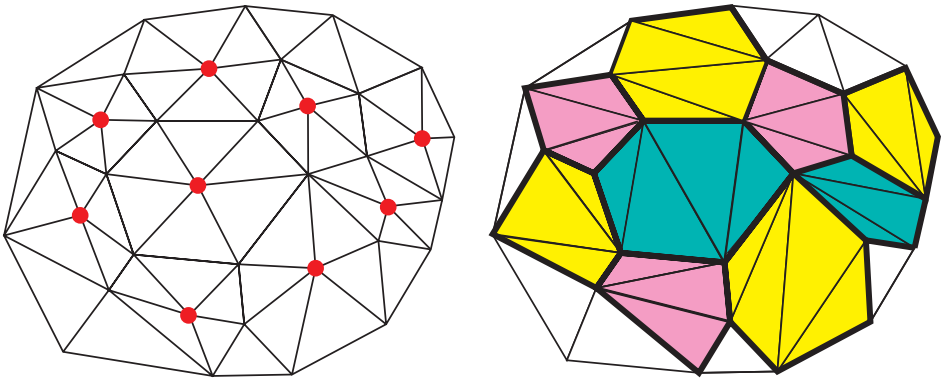
The key idea is to encode the triangles of a patch by means of coloring them, such that the decoder can detect the patches during the reconstruction stage based on the triangle colors. Thus, adjacent patches cannot be assigned the same color, where two patches are said to be adjacent if they share an edge. The triangles of M_i are recursively traversed and each patch is assigned a color that is different from the colors assigned to its adjacent patches. Since the patches do not tessellate the entire mesh, we use a null color for the triangles that are not included in any patch. The rest of the triangles are colored in only three colors (see Figure 1). Three colors are not always enough, but in practice such cases are rare, and can be avoided by giving up the removal of some vertices.

This coloring technique requires 2 bits per triangle. Thus, the cost of encoding a vertex is the cost of coloring the triangles of the patch created by its removal. Assuming the degree of a vertex is 6, then its removal requires coloring four triangles, that is, 8 bits per vertex removal. Note that there is some overhead since some triangles are not included in any patch. To reduce this overhead, when selecting the vertices to be removed we strive to create a maximal independent set.

The above coloring technique can be improved by triangulating the patches by a dependent triangulation that can be encoded with only one bit per triangle. A hexagonal patch (the most popular patches in common triangulation) is triangulated by the three edges of the shape of the letter *Z* (see Figure 2(1-2)). Then its two middle triangles are encoded with a '1' bit (purple in the figure) and the two external triangles with a '0' bit (white in the figure). Pentagons are triangulated with three triangles where the middle one is encoded with a '1' and the others two with a '0'. Here the middle triangle T_m must be selected such that the two other triangles share a vertex with the smallest angle in T_m. For an n-gon the concept is the same; the sequences of 'alternating' triangles are colored with '1's and the two externals with '0's (see Figure 2(3)). While encoding adjacent patches we need to avoid edge-adjacent '1'-encoded triangles. Recovery of the patches is guaranteed since the sequence of adjacent '1's has a known shape, from which the two external '0'-encoded triangles are uniquely recovered and, as a consequence, the boundary of the patch is also uniquely recovered. Note that with this technique, a quadrilateral cannot be encoded. However, this encoding requires only one bit per triangle. Figure 3 illustrates a 2-coloring of a mesh.

A sequential order of the triangles of M_i is defined by a breadth-first traversal of the triangles. One bit is associated with each triangle and stored in a binary vector, which represents the colors of the triangles. The length of the vector is $|M_i|$; the number of triangles of the mesh, M_i. The vector is then compressed by some lossless compression technique. In our implementation we used an LZ encoder.

During reconstruction, for each recovered patch we remove its triangles and predict the location of the vertex that was removed when the patch was created. By adding the associated displacement vector to the predicted point, the original location of the vertex is recovered. Then we simply connect the vertices of the patch to the inserted vertex.



(1) (2)

Figure 1: The 4-color encoding scheme. (1) The original mesh consists of an arbitrary triangulation. The red circles are the vertices selected to be removed - the independent set. (2) The mesh after removing the red vertices and triangulating the hole. The new triangles are colored in three colors; the rest remains white. See the color plates.

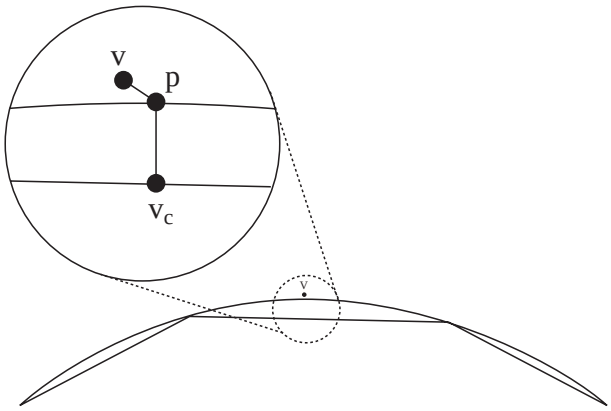


Figure 4: The prediction scheme.

5 The Prediction Operator

Given a patch the prediction operator predicts the location of a new vertex based on the known vertices of the patch $\{v_i\}$ and their immediate neighbors. Lacking any prior knowledge, the best guess seems to assume the surface to be smooth and use a prediction based upon a local polynomial interpolant with respect to a local reference plane. The basic idea is illustrated in Figure 4: let $v_c = \frac{1}{n}\sum_{i=1}^{n} v_i$, and P be the predicted point "above" v_c. If the encoded vertex v is indeed closer to P than to v_c, then encoding $v - P$ is better than $v - v_c$. However, this is not necessarily the case. Assume the patch has a diameter h and the surface is locally smooth, $\|P - v_c\| = O(h^2)$, while $\|v - P\|$ as well as $\|v - v_c\|$ is $O(h)$. Therefore, the prediction of the polynomial interpolant does not pay off whenever h is either small or large. Since the decompression time is an important factor, we make a simple and effective prediction by means of v_c.

6 Results

As discussed in Section 4, the patches can be colored by using either four colors or two. The 4-color technique requires 2 bits per triangle and the 2-color technique only 1 bit per triangle. Denoting by m the number of bits used to color the triangles, the cost of encoding a d-vertex is $m(d-2)$ bits, since the patch created by removing a d-degree vertex consists of only $d-2$ triangles. Using the 2-color technique, the removal of a 6-degree vertex requires 4 bits, and a 5-degree vertex only 3 bits.

From the Euler formula we know that the average of the degrees is always close to six. However, the distribution of the vertex's degrees can vary. If the mesh of a given level of detail consists mainly of vertices of degree 5 and higher, the 2-color technique is very effective. However, if the mesh consists mainly of vertices of degree 4 and 3, the 4-color technique is more effective since the 2-color technique cannot be applied to low degree vertices. On the other hand, since the patches created by the removal of low degree vertices consists of one or two triangles only, the cost of 2 bits per triangle means that the cost of encoding the insertion is only 2-4 bits. Thus, roughly speaking, if the mesh consists of either 5-6 degree vertices or 3-4 degree vertices, the encoding of the connectivity requires no more than 4 bits per vertex. Since the independence set is not optimal, there are many triangles that are not included in any of the patches. Thus, in practice the cost is higher than 4 bits per vertex (see Table 1). In any case the stream of bits that colors the mesh is further compressed by an LZ encoder. In our implementation the coloring technique is selected according to the distribution of degrees in the given level of detail. See Figure 5 which shows the first four intermediate levels and their coloring.

Our 2-coloring encoding technique requires only one bit per triangle. However, in terms of the number of bits per vertex the cost is at least 4 bits per vertex, or 2 bits per triangle. Note that the number of triangles in the entire hierarchy is about three times the number of triangles in the original mesh. This cost is about that reported for the progressive forest split [18], almost twice more than reported in [20, 8], and eight times better than the cost of a vertex split of the progressive mesh [9].

Regarding the compression of geometry, the stream of the displacement is encoded using Huffman encoding. We have tested the results with 12-bit precision per coordinate. Table 2 compares our results with those of Touma and Gotsman's technique [20], which are the best published so far. On average our results are only about 8% higher than those achieved by Touma and Gotsman (see Table 2). It should be emphasized that the exact compression values are dependent on the specific implementation of the Huffman encoder. However, we believe that the exact compression ratios are not crucial as the superiority of our technique is in its progressiveness. We should emphasize that with respect to the original VRML mod-

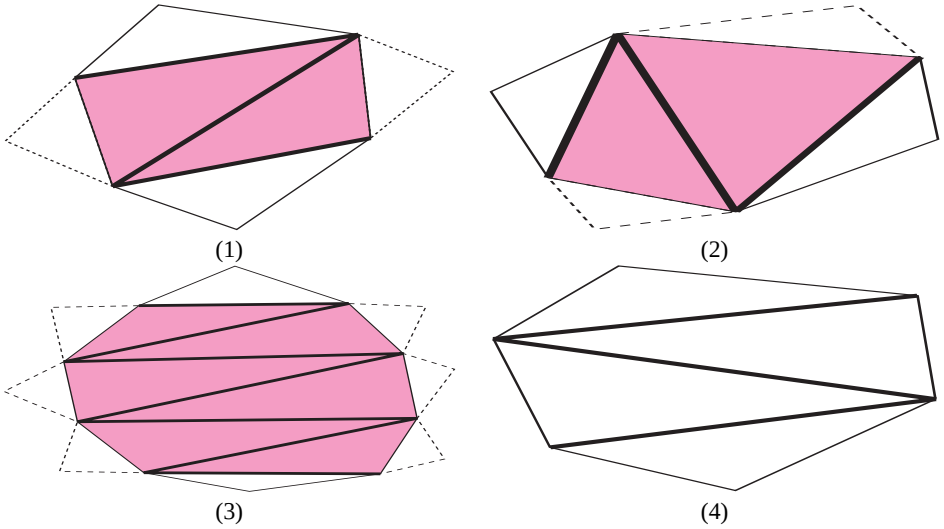


Figure 2: The 2-color encoding scheme. (1) A 2-colored hexagon. Note the 'Z' shape of the triangulation, which implies on the two end triangles of the patch. (2) Another legal 2-coloring of the same hexagonal patch. (3) A 2-colored 10-gon patch. The 'Z' (alternating) triangulation implies on the two end triangles of the patch. (4) An illegal triangulation. The 'Z' is mirrored.

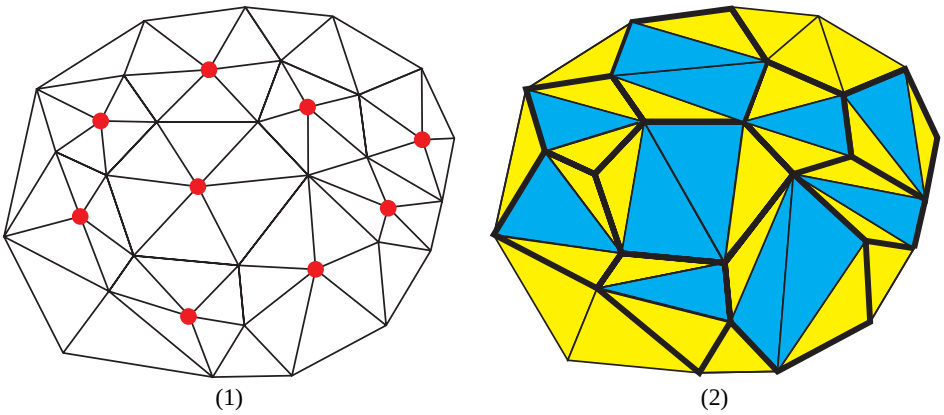


Figure 3: The 2-color encoding scheme. (1) The original mesh consists of an arbitrary triangulation. The red circles are the vertices selected to be removed - the independent set. (2) The mesh after removing the red vertices and triangulating the hole.

els, Touma and Gotsman's and our technique achieve an average of 3.1% and 3.4% compression ratios, respectively.

7 Conclusions

The progressive compression method presented above is based on the space-efficient encoding of a vertex insertion refinement process, where a mesh coloring technique encodes the connectivity of the intermediate meshes. Unlike previous compression methods, which use a sequential linear prediction scheme to encode the displacements, here we use a spatial predictor. The primary advantage of our compression method is that it inherently has the feature of a progressive mesh. Thus, the compressed data is a stream of vertex insertions, which progressively refines an initial coarse mesh into the original fine mesh. We showed that this property is achieved without sacrificing the compression ratio that can be achieved by non-progressive techniques.

The advantage of our technique is that it can deal with non-manifold meshes without decomposing them first into a set of manifold meshes. Since our technique is based on reversing the robust vertex decimation operation, it can also deal well with boundary polygons.

In our work we only tested models with no special attributes, such as normal vectors, colors or texture coordinates. Encoding such attributes is similar to that of the coordinates. If the attributes have some spatial coherence as do the coordinates, the local predication function and the displacements are expected to behave similarly to the coordinates. We will test this in our future work.

References

[1] Reuven Bar-Yehuda and Craig Gotsman. Time/space tradeoffs for polygon mesh rendering. *ACM Transactions on Graphics*, 15(2):141–152, April 1996.

[2] M. De Berg and K. Dobrindt. On levels of detail in terrains. *Graphical Models and Image Processing*, 60:1–12, 1998.

Table 1: *The models and the connectivity compression results. The first two columns show the number of vertices and triangles in each model. The third column shows the number of vertices removed. The fourth column the number of bytes of the connectivity stream, and the last column shows the number of bits per vertex. The average of the bit/vertex column is 5.98*

model	vertices	triangles	removed	connectivity	bits/vertex
horse	19851	39698	15494	11090	5.72
jaw	12349	24585	9453	6966	5.89
blob	8036	16068	6067	4507	5.94
holes3	5884	11776	4246	3526	6.64
tricerotops	2832	5660	2171	1550	5.71

Table 2: *Compression results. The same models as in Table 1 with 12 bits per coordinates precision. The second column shows the size of the stream and the base is the size of the base model compressed. Their sum is presented in the next column. The TG column show the results of Touma and Gotsman's method. The right most column contains the compression ratio between TG and our progressive compression method. The average ratio is 1.08*

model	stream	base	total	TG	ratio
horse	42001	10445	52446	47108	1.11
jaw	29533	8090	37623	34577	1.08
blob	19928	5757	25685	21396	1.18
hole3	8182	4001	12183	13452	0.90
tricerotops	7252	2026	9278	7871	1.17

[3] P.J. Burt and E.H Adelson. The laplacian pyramid as a compact image code. *IEEE Trans. Communications*, 31(4):532–540, April 1983.

[4] Michael F. Deering. Geometry compression. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 13–20. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[5] M. Denny and C. Sohler. Encoding a triangulation as a permutation of its point set. In *Canadian Conference on Computational Geometry*, pages 39–43, 1997.

[6] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 173–182. ACM SIGGRAPH, Addison Wesley, August 1995.

[7] F. Evans, S. Skiena, and A. Varshney. Optimizing triangle strips for fast rendering. In R. Yagel and G. M. Nielson, editors, *Proceedings of the IEEE Visualization'96*, pages 319 – 326, October 1996.

[8] Stefan Gumhold and Wolfgang Straßer. Real time compression of triangle mesh connectivity. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 133–140. ACM SIGGRAPH, Addison Wesley, July 1998.

[9] Hugues Hoppe. Progressive meshes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996.

[10] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 105–114. ACM SIGGRAPH, Addison Wesley, July 1998.

[11] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. Maps: Multiresolution adaptive parameterization of surfaces. *Computer Graphics Proceedings (SIGGRAPH 98)*, pages 95–104, 1998.

[12] Michael Lounsbery, Tony D. DeRose, and Joe Warren. Multiresolution analysis for surfaces of arbitrary topological type. *ACM Transactions on Graphics*, 16(1):34–73, January 1997.

[13] Renato Pajarola and Jarek Rossignac. Compressed progressive meshes. Technical Report GIT-GVU-99-05, Georgia Institute of Technology Washington, 1999.

[14] Jarek Rossignac. Edgebreaker: Compressing the incidence graph of triangle meshes. Technical Report GIT-GVU-98-17, Georgia Institute of Technology Washington, July 1998.

[15] David Salomon. *Data Compression, the complete reference*. Springer, 1998.

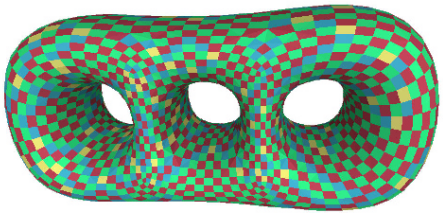
[16] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[17] W. Sweldens. The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In A. F. Laine and M. Unser, editors, *Wavelet Applications in Signal and Image Processing III*, pages 68–79. Proc. SPIE 2569, 1995.

[18] Gabriel Taubin, André Gueziec, William Horn, and Francis Lazarus. Progressive forest split compression. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 123–132. ACM SIGGRAPH, Addison Wesley, July 1998.

[19] Gabriel Taubin and Jarek Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, 17(2):84–115, April 1998.

[20] C. Touma and C. Gotsman. Triangle mesh compression. *Graphics Interface'98*, pages 26–34, June 1998.



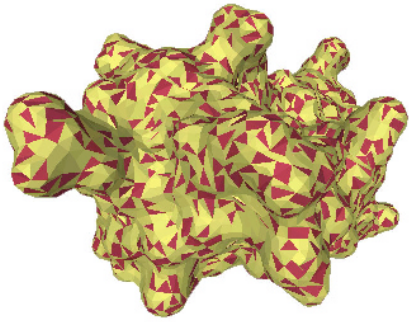
(Level1)

(Level2)

(Level3)

(Level4)

Figure 5: The hole3 colored at different levels of detail. Note that in the first two levels it uses a 4-coloring, as most of the vertices are of degree four or three, and a 2-coloring in the next two levels. See the color plates.



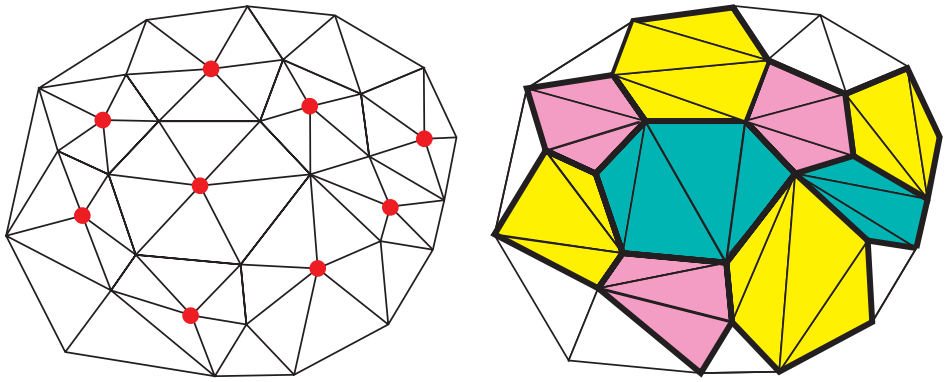
(Horse)

(Blob)

(Tricerotops)

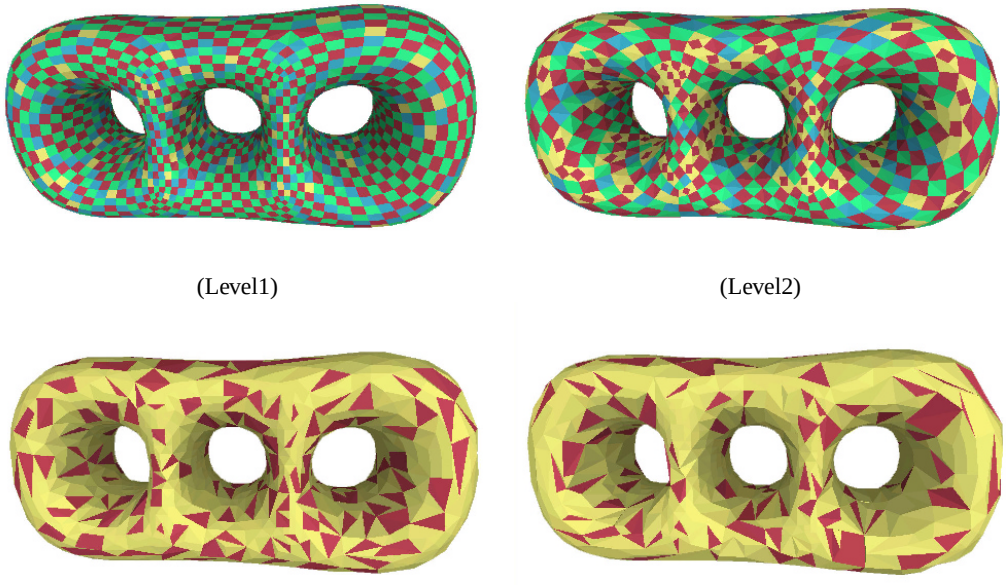
(Jaw)

Figure 6: Two models colored with the 2-color technique.



(1) (2)

Figure 2: The 4-color encoding scheme. (1) The original mesh consists of an arbitrary triangulation. The red circles are the vertices selected to be removed - the independent set. (2) The mesh after removing the red vertices and triangulating the hole. The new triangles are colored in three colors; the rest remains white.



(Level1) (Level2)

(Level3) (Level4)

Figure 6: The hole3 colored at different levels of detail. Note that in the first two levels it uses a 4-coloring, as most of the vertices are of degree four or three, and a 2-coloring in the next two levels.