

Single Resolution Compression of Arbitrary Triangular Meshes with Properties *

Chandrajit L Bajaj Valerio Pascucci Guozhong Zhuang
Dept. of CS and TICAM, University of Texas, Austin, TX 78712
{bajaj, pascucci, zgz}@cs.utexas.edu

1 Introduction

Polygonal meshes have been used as the primary geometric model representation for networked gaming and for complex interactive design in manufacturing. Accurate polygonal mesh approximation of a surface with sharp features (holes, highly varying curvatures) requires extremely large number of triangles. Transmission of such large triangle meshes is critical to many applications that interactively manipulate geometry models in remote networked environments. The need for succinct representation is therefore not only to reduce static storage requirements, but also to consume less network bandwidth and thus reduce the transmission time. Although geometry compression and coding is an emerging discipline, these techniques have matured for 2D digital images into standards such as JPEG [16] and MPEG [6]. In designing efficient geometry compression schemes, one attempts to take advantage of existing 2D image compression techniques.

Prior Work Deering [5] represents triangular mesh connectivity by generalized triangle strips. Stack operators are used in order to reuse vertices. In this way, the total number of random accesses to all vertices of the mesh is reduced. This method does not directly compress non-manifold meshes and its compression ratio is not high. Chow [3] presents an algorithm which represents a mesh by several generalized meshes. This method is optimized for real-time rendering but is not compression efficient because of the large requirement of connectivity encoding ($(\log n + 9)$ per vertex). Again, this method only considers manifold meshes.

In Topological Surgery [14], vertices are organized into a spanning tree and triangles into simple polygons which are further grouped into a series of triangle strips. The connectivity coding of this scheme is efficient, about 2-3 bits per triangle. One of its disadvantages is its inability to directly encode non-manifold meshes. As a preprocessing step, it splits a non-manifold object into several manifold components, thereby duplicating all non-manifold features: vertices, edges, and faces. Touma and Gotsman present an algorithm for connectivity coding of orientable manifold meshes [15]. The efficiency of this lossless connectivity coding is determined by the distribution of the degrees of all vertices. Progressive transmission and embedded coding are discussed in [11, 10, 12]. A compact representation of multiresolution surfaces that support progressive transmission is present in [2].

In this paper, we propose a new layering structure to partition an arbitrary triangular mesh (no-manifold and arbitrary-genus) into generalized triangle strips. An efficient and flexible encoding of the connectivity, vertex coordinates and attribute data yields excellent single resolution compression. This scheme gracefully solves the “crack” problem and also prevent error propagation while providing efficient prediction coding for both geometry and

*This research is supported in part by grants from NSF-CCR-9732306, NSF-KDI-DMS-9873326, DOE-ASCI-BD-485, and NASA-NCC 2-5276.

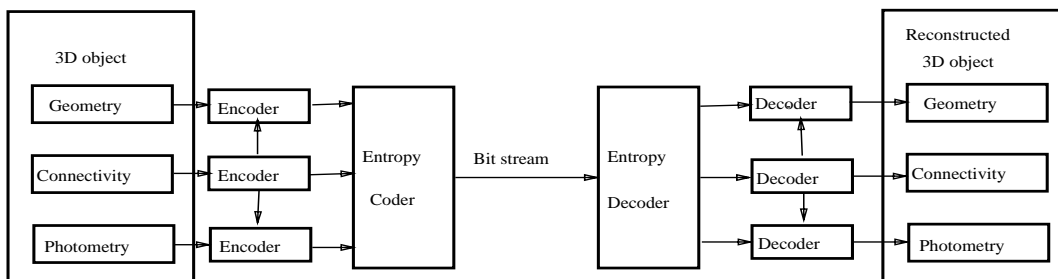


Figure 1: Block diagram of the encoder and decoder

photometry data such as positions, color, normal, and texture coordinates. Figure 1 shows the diagram for both the encoder and decoder.

The rest of this paper is as follows. Section 2 introduces the layering scheme to partition input data. Section 3 and Section 4 address the coding of connectivity and geometry. The attribute coding is discussed in Section 5. Experimental results are presented in Section 6.

2 The Layering Scheme

Layering structures are mainly used to describe the connectivity information of surface meshes. There are two basic kinds of layers: vertex-layers and triangle-layers. Assume that a triangle mesh has vertex set \mathcal{V} , face set \mathcal{F} , and edge set \mathcal{E} . A graph of the mesh is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with all mesh vertices being the graph nodes and mesh edges being the graph edges.

Definition 2.1 (*vertex-layer*) The 0^{th} vertex-layer is a randomly chosen vertex of the mesh. The k^{th} vertex-layer (with $k > 0$) includes a vertex V if V is not included in any previous vertex-layer and there exists an edge $E = (V, V^*)$ where V^* is included in the $(k - 1)^{\text{th}}$ vertex-layer.

Definition 2.2 (*triangle-layer*) The k -th triangle-layer (with $k \geq 0$) includes a triangle T if T has one vertex in k^{th} vertex-layer and T is not included in any previous triangle-layer.



Figure 2: The layering structure. Triangle-layers are alternatively colored black and white for both apple and horse. (right) Alternative layers of the horse.

Vertices and triangles categorized above have the following properties: any edge in \mathcal{E} can only span two vertex-layers; any triangle in \mathcal{F} can only span two vertex-layers; all the vertex-layers form a partition of \mathcal{V} ; all the triangle-layers form a partition of \mathcal{F} . Based on the layering structure, a mesh edge is either a *chord* or a *transversal*. A chord is an edge that connects two vertices in different vertex-layers while a transversal is an edge that connects two vertices in the same vertex-layer.

Geometry Primitives The four geometry primitives we use to encode in our single-resolution compression method are: contours, branching points, triangle strips, and triangle fans.

Definition 2.3 A contour is an ordered sequence of vertices $\{v_0, v_1, \dots, v_n\}$ in a vertex-layer where each vertex pair (v_i, v_{i+1}) is connected by a transversal edge and every intermediate vertex $v_i (0 < i < n)$ is incident to exactly two transversal edges. Vertices v_0 and v_n can be the same point in which case the contour is called closed. A contour can be a single vertex which is also called an isolated point.

We call v_0 the starting vertex and v_n the ending vertex of the contour.

Definition 2.4 A branching point is a vertex in a vertex-layer which is incident to more than two transversal edges.

By this definition, a branching point is contained in more than one contour and thus it can only be either the starting or the ending vertex of a contour.

Definition 2.5 A triangle strip is an ordered sequence of triangles in a triangle-layer where each pair of consecutive triangles share a common edge. The set of vertices of the triangles in the strip must belong to two contours in two separate vertex-layers.

Definition 2.6 A triangle fan is an ordered sequence of triangles in a triangle-layer where all triangles have a common vertex, each pair of consecutive triangles share a common edge, and no edge is shared by more than two triangles. It is possible that the first and last triangles share a common edge.

Figure 2 exhibits triangle strips in the layering structure. Triangle strips are formed by stitching individual triangles in the same triangle-layer. In this way, non-manifold meshes can be represented by the structure. This representation avoids the “crack” problem which occurs when a non-manifold mesh is converted into several manifold components and non-manifold features (vertices, edge, faces) are duplicated [7].

Claim 2.1 Triangle meshes of any topological types can be represented by the layering structure outlined above.

Vertex indexing gives each vertex a reference which is used in connectivity encoding. Vertex indexing is directly related to coding efficiency. Three kinds of vertex indices will be used in our scheme: local indices, global indices and relative indices. The first two are defined below while relative indexing will be explained in the next section.

Local Indexing Local indexing is only meaningful to individual vertex-layers. Every vertex in a vertex-layer is assigned a unique local index. If n is the total number of vertices in a vertex-layer, then local indices of this vertex-layer are $0, 1, \dots, n - 1$. The numbering order of local indices in a vertex-layer is as follows: (1) all branching vertices are numbered first; (2) for each contour, all non-branching vertices are numbered, from the starting vertex. It is important that all branching vertices be numbered separately because they will be referenced by more than one contour.

Global Indexing The global index of a vertex is defined as the summation of its local index value and the total number of vertices in all previous vertex-layers. Indexing starts from the 0^{th} vertex-layer. Suppose that there are totally V vertices in the mesh, then the smallest global index is 0 and the biggest one is $V - 1$. The incidence of the reconstructed mesh is expressed with global indices. Let $BC(m)$ be the minimal number of bits to correctly code a non-negative integer smaller than m . Obviously, $BC(m) := \min\{k | 2^k > m\}$.

Suppose that L is the number of vertices in a vertex layer, then the local index of each vertex in that layer can be coded by $BC(L)$ bits. However, any global index needs $BC(V)$ with a total number of V vertices in the mesh. Since L is generally much smaller than V , $BC(L) < BC(V)$, local index coding saves $BC(V) - BC(L)$ bits. A local index must be converted back to the global index in the decoding process. A global index g and a local index l of a vertex v have the relation $g = l + \sum_{i=0}^{k-1} L_i$ where L_i is the total number of vertices in i^{th} vertex-layer and layer k is where v is located.

3 Connectivity Coding

Connectivity encoding is the central part of any 3D compression method. It guides the geometry and photometry coding. Single-resolution compression does not change the connectivity in the sense that the decoder can perfectly recover the connectivity, modulo a permutation of the vertices and triangles.

Scheme Outline Our connectivity scheme is based on the layering structure. For an input mesh, vertex-layers and triangle-layers are constructed by using a breadth-first traversal algorithm [4]. According to the set of transversal edges, contours are built for each vertex-layer while all branching points are recorded. Similarly, according to the set of chords, triangle strips are constructed for each triangle-layer and triangle fans are formed from the remaining triangles which are not contained in any strip. The entire connectivity encoding procedure is: (1) encode the total number of layers; (2) encode the layout of each vertex-layer; and (3) encode all triangle strips and fans in each triangle layer.

The layout of a vertex-layer is specified by the number of branching points; the number of contours; for each contour the number of vertices, and the characteristics of the starting and ending vertices (one bit each to indicate if it is a branching vertex; if it is, its local index is coded). A triangle strip has two boundary contours. The one in the previous vertex-layer is called the *parent contour*, the other is called the *child contour*.

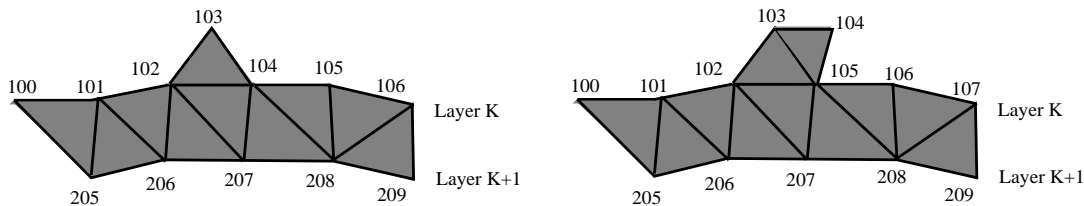


Figure 3: Exceptional triangle strips where local indices are listed: (left) one bubble triangle (102, 104, 103) (right) two bubble triangles (102, 105, 103) and (103, 105, 104)

The Coding of Bubbles An exceptional triangular strip possibly has some special triangles attached to its parent contour. These attached triangles are called bubble triangles because their vertices are located in the same contour. Exceptional triangle strips are shown to be good basic geometric primitives for the connectivity encoding [17]. To encode an exceptional triangle strip, the following information must be coded: local indices of two starting vertices, the bit march string, the number of bubbles, and bubbles. The bit march string is further encoded by an entropy coding algorithm (Huffman coding [9] or arithmetic coding [13]). Figure 3 shows two strips with attached bubbles. To encode a bubble on a triangle strip, the following information is coded: location of the starting vertex involved; vertex span which is the total number of involved vertices; number of triangles in the bubble; triangle index triples. Bubble coding is expensive. However, not all the above values need

to be put into the bit stream. We use relative indexing to reduce the overall coding bits. A relative index is defined with respect to a bubble. Suppose that a bubble spans n vertices and the local index of the starting vertex of the bubble is X . Then the relative indices of all involved vertices are defined as $0, 1, \dots, n - 1$. A triangle in a bubble is encoded by a triple of relative indices. For instance, since the relative indices of 102, 103, 104, 105 in Figure 3 (b) are 0, 1, 2, 3, the two bubble triangles are encoded as (0, 3, 1) and (1, 3, 2). In this case, coding a triangle only costs 6 bits.

Another technique to enhance the efficiency of bubble coding is based on the observation that in practice most bubbles appear in the form of a one-triangle bubble with a vertex span of 3, and relative index triple fixed as (0, 2, 1). An efficient way to encode is to use an extra bit to indicate if a bubble is a one-triangle bubble and no more information is needed.

The Coding of Bit March Strings Once the two starting vertex positions and possible bubbles are coded, the remaining problem of coding an exceptional strip is reduced to coding generalized triangle strips. A simple and direct way to encode a bit march string is to put the 0s and 1s directly into the bit stream. However, it is not an efficient way. Symbols 0101 and 1010 are found to appear more frequently than other symbols such as 1111 or 0000 in a typical bit march string if every four consecutive bits are grouped into a symbol.

A static Huffman table is designed for all the sixteen symbols. Construction of this table is based on the symbol occurrences over a number of large models. In case the length of a bit march string is not a multiple of 4, its remaining bits are simply be coded one by one. Entropy coding of bit march strings uses about 25% less space than direct coding.

The Coding of Triangle Fans Triangles that do not belong to any strip are special. Their vertices usually span more than three contours or two contours in the same vertex-layer. A triangle fan can be expressed by a sequence of vertices $\{v_0, v_1, \dots, v_m\}$ where v_0 is the common vertex and (v_0, v_i, v_{i+1}) ($i = 1, \dots, m - 1$) is a triangle. To code a triangle fan, the following information is put into the bit stream : the number m of triangles in the fan; a sequence of local indices of $m + 1$ vertices with the first one as the center (the commonly shared vertex) of the fan.

For large models, the connectivity cost of our scheme can be as good as less than one bit per triangle. Experiments show that connectivity coding cost is on average 3 bits per triangle for common objects [17].

4 Geometry Coding

With the encoded connectivity information, the overall geometry encoding scheme is straightforward. For each vertex-layer, positions of all branching vertices are encoded directly in the order they are locally indexed. For each contour, positions of all its non-branching vertices are predictively encoded.

The procedure of our geometry coding scheme involves bounding box coding, prediction, quantization, and entropy coding. The bounding box of an input mesh is specified by the maximum and minimum values of all the x, y, z coordinates: $Xmin, Xmax, Ymin, Ymax, Zmin, and Zmax$. Using these values, all vertex positions can be normalized so that they are located in a unit cube.

Predictive Geometry Coding Predictive coding has been extensively utilized in 2D image compression methods. For instance, the JPEG standard [16] provides predictive lossless coding mode for still image compression. The predictive technique can also be used to remove redundant information within the geometry and attributes of a triangular mesh. A vertex position predictor combines the positions of previously encoded neighboring

vertices to form a prediction of the current vertex position. There are three commonly used geometry predictors: linear predictors [5], high-order predictors [14], and parallelogram predictors [11]. A correction, more suitable for efficient coding, is defined as the difference between the actual vertex position and its predicted position. Either Cartesian or spherical coordinates can be used to express a position or a correction which is quantized into an integer code and then entropy coded.

To quantize a position or a correction vector, we first compute its spherical coordinates (r, ϕ, θ) . Then the three components are vector quantized to a choice of vectors chosen uniformly in a solid sphere. With the bounding box and other pre-normalization, r is in the interval $[0, 1]$, and ϕ and θ are in $[0, 2\pi]$ and $[0, \pi]$ respectively. The details of the codebook design, uniform quantization of sphere and error propagation prevention can be found [17].

Second-Order Code Prediction Linear prediction removes redundancy by identifying similar bit values between coordinates of adjacent vertices. However, it is not an optimal way, especially for models without many sharp features.

Suppose ΔP_k and ΔP_{k+1} are two adjacent correction vectors and the two integer triples (r_k, ϕ_k, θ_k) and $(r_{k+1}, \phi_{k+1}, \theta_{k+1})$ are their corresponding codes. We observe that since the angular change from ΔP_k to ΔP_{k+1} is generally small, so $\Delta \phi_k = \phi_{k+1} - \phi_k$ and $\Delta \theta_k = \theta_{k+1} - \theta_k$ are usually small integers. It must be pointed out that the true correction values Δr , $\Delta \phi$ and $\Delta \theta$ could be negative. However, there is no need to spend one extra bit to indicate the sign of each correction value. A value N is added if the correction value is negative. The decoder takes care of this convention by making sure that the recovered values r , ϕ and θ be within the range $[0, N - 1]$. Here r , ϕ and θ stand for the integer codes, not the true float values. A simple example may help to understand this approach. Suppose that we wish to encode r_1 and r_2 , the codes of the lengths of two consecutive correction vectors. Also assume $r_1 > r_2$. Thus $\Delta r = r_2 - r_1 < 0$. Now we encode $\Delta r + N$ which is in the interval $[0, N - 1]$, instead of encoding negative Δr . After having recovered r_1 and the correction value $\Delta \bar{r}$, the decoder first computes $\bar{r}_2 (= r_1 + \Delta \bar{r})$. If \bar{r}_2 is in the interval $[0, N - 1]$, then r_2 equals to \bar{r}_2 . In our case, $\bar{r}_2 \geq N$, which means that $\Delta \bar{r}$ is the summation of the true difference and N . So r_2 equals to $\bar{r}_2 - N$ according to the fact that $\bar{r}_2 = r_1 + (r_2 - r_1 + N) = r_2 + N$. This approach can also be applied to the other two components ϕ and θ .

Experimental results show that encoding correction code differences is more efficient than directly encoding correction codes. Figure 4(a-c) shows the frequencies of symbols of different sizes where horizontal values are symbol values and vertical values are their corresponding frequencies. In this figure, solid curves show the frequency of correction code differences while dash lines show the frequency of correction codes from the direct encoding scheme. The Huffman coding method [9] is also used to encode code differences.

5 Attribute Coding

Besides vertex positions, a mesh may have attached attributes such normals, colors, and texture coordinates which are used for enhancing shading effect. There are four ways in VRML [8] to bind these attributes with a triangular mesh: no binding, per vertex, per face, and per corner.

The position predictors can be generalized to code normals, colors, and texture coordinates. In this paper, we present our prediction schemes for normal coding with the ‘‘per vertex’’ binding. For the other two attributes and other bindings, similar schemes are designed in [17].

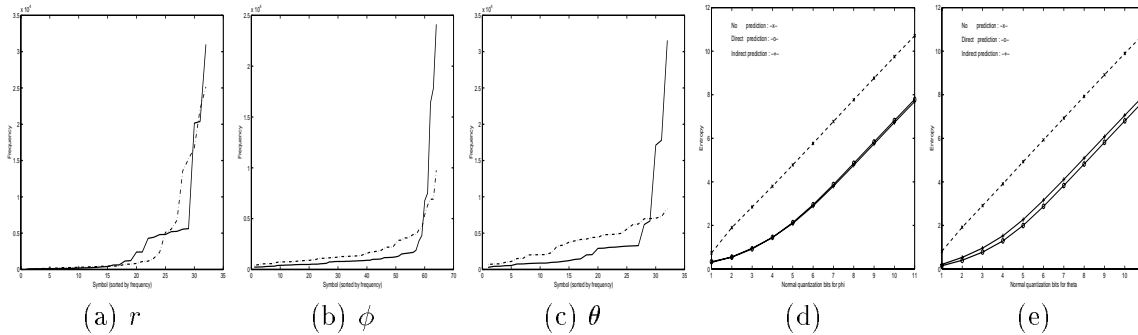


Figure 4: (a-c) Frequency curves for two encoding schemes: solid curves for code difference and dash lines for the direct encoding. The symbols are sorted by frequency for comparison. These frequency curves show that in the code difference approach most codes are accumulated around only a few symbols. (d-e): Comparison of geometry coding: no prediction, indirect prediction, and direct prediction.

Just like coding a vertex position, one can use a linear combination of encoded normals of neighboring entities (vertices or faces) to predict a normal. This is direct prediction. However, the normal associated with each entity usually reflect local geometry variation around the entity. This implies that the normal can be predicted from neighboring geometry which is supposed to be already recovered by the decoder. We call this approach indirect prediction. Several methods to estimate vertex normals can be used in an indirect prediction approach: average of normals of incident faces, quadratic surface fitting, and subdivision. Since indirect predictors are based on the geometry information of the reconstructed mesh, the encoder must use the same geometry information when it does normal predictions. The first predictor averages normals of all incident faces and uses the average normal vector as the prediction vector. The second predictor uses geometry information of neighboring vertices to construct a quadratic fitting surface and uses the normal of the fitting surface at the predicated vertex as the prediction. The subdivision predictor is similar to the first predictor but the incident faces are changed. Figure 4(d-e) compares the efficiency of normal coding of three different method: no prediction, indirect prediction by averaging, and direct prediction from the normal of a neighboring vertex. Clearly, coding normals without prediction is least efficient. For the other two cases, the coding results are largely dependent on the input data.

For any unit normal (n_0, n_1, n_2) , its corresponding sphere coordinates can be written as $(1, \phi, \theta)$. So direct coding (no prediction) only needs to code the ϕ and θ components. Any correction vector of a normal does not need to be of unit length. However, since both the prediction normal and the true normal are of unit length, it is sufficient to code the ϕ and θ components of the correction vector.

6 Experimental Results

Geometry Error Suppose A is the original model and B is the reconstructed model. The geometric error between A and B is defined as $E(A, B) = \frac{1}{n} \sum_{i=0}^{n-1} \min_{0 \leq j < n} \|P_i - Q_j\|_2$ where $P_i (0 \leq i < n)$ and $Q_j (0 \leq j < n)$ are the vertex positions of A and B respectively, and A is normalized such that its bounding box diameter is 100. This geometric error is used for results in Table 1 where our results are compared with the GZIP compressed files. Figure 7 compares the compression results of our method compressed gzip files over a sample set of 300 VRML models are tested. Every point in this figure is a compression

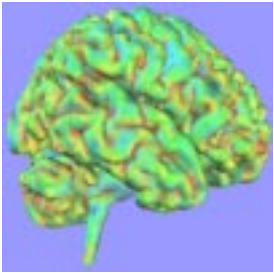
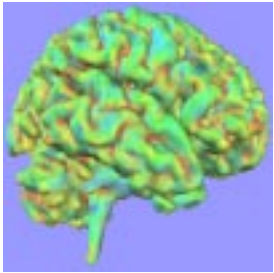
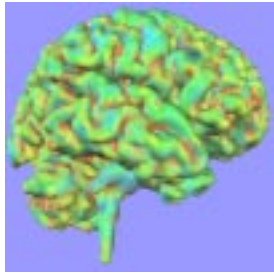
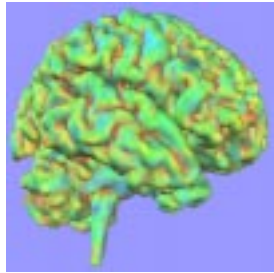
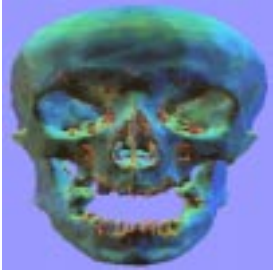
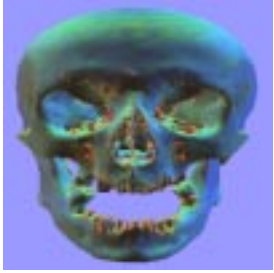
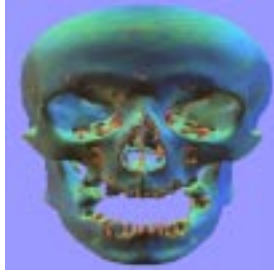
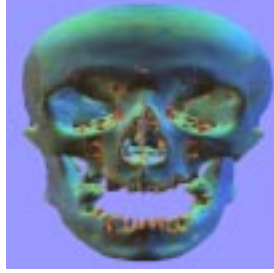
			
177,652	253,360	253,360	4,453,579
			
456,734	633,794	823,867	11,260,390

Figure 5: Compression of normals and colors. Both models (brain and skull), whose originals are shown in the last column (GZIP sizes: 981,827 and 2,431,665 bytes), have normal and color properties. For the first three columns, geometry quantization bits are 12, 18 and 24; normal quantization bits are 8, 12, 16; color quantization bits are 12, 18 and 24. Total storage sizes are reported below each model in bytes.

ratio which is the quotient of size of VRML/GZIP file divided by size of our compressed file. The horizontal axis is the number of vertices in each model while the vertical axis shows is the compression ratio. Figure 6 shows our compression results of size triangular models where the original files are stored in VRML format [8].

Model	Original	GZIP	CC	GC (25 bpv)	GC (15 bpv)
Crocodile	1,212,767	395,849	13,055	63,127 (0.010)	39,674 (0.073)
Teapot	5,471,965	1,459,284	12,644	150,900 (0.007)	93,090 (0.063)
Honda	397,730	118,377	3,768	25,739 (0.012)	16,459 (0.101)
Buddha	40,443,806	10,309,238	713,809	1,950,072 (0.003)	1,240,212 (0.011)

Table 1: Comparison of compressed file sizes in bytes. The fourth column (CC) is connectivity coding cost. The last two columns are geometry coding (GC) cost with two different quantization bits per vertex (bpv). The numbers in parenthesis are geometry errors.

Besides the ability of handling any kind of triangular meshes, our layering compression scheme also cherishes a blocking feature that makes it particularly suitable for both incremental transmission/display and error resilient streaming [1].

7 Conclusion

We have described a space efficient encoding for both a lossless and an error-bounded lossy compression scheme for triangular meshes. The compression is achieved by capturing the

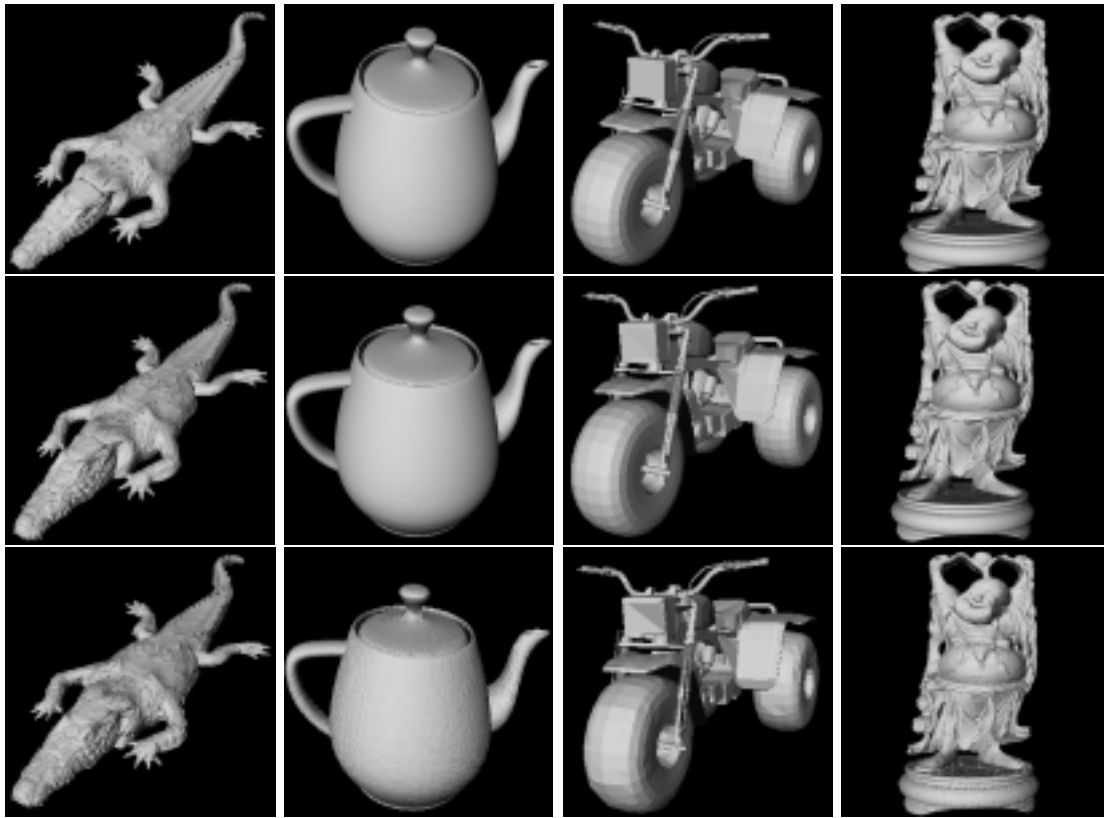


Figure 6: Compression of triangle meshes. The first row shows the original models. The second and third rows are the compressed models which use 25 and 15 bits per vertex.

redundant information in both the topology (connectivity) and geometry, and possibly property attributes. Error-bounded lossy geometry (without loss of topology) is achieved by a vector predictor and corrector encoding. Example models and results of our implementation are also provided. Our future research concentrates on progressive encoding and transmission of both topology and geometry information for large models, as well as in error recovery and error concealment encoding schemes.

References

- [1] C. Bajaj, S. Cutchin, V. Pascucci, and G. Zhuang. Error Resilient Streaming of Compressed VRML. Technical report, TICAM, The University of Texas at Austin, 1998.
- [2] C. Bajaj, V. Pascucci, and G. Zhuang. Progressive Encoding and Transmission of Arbitrary Triangular Meshes with Properties. Technical report, TICAM, The University of Texas at Austin, 1998.
- [3] M. Chow. Optimized Geometry Compression for Real-time Rendering. In *Proceedings of IEEE Visualization '97*, pages 347–354, Phoenix, AZ, 1997.
- [4] T. H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, 1991.
- [5] M. Deering. Geometric Compression. *Computer Graphics (SIGGRAPH '95 Proceedings)*, pages 13–20, 1995.

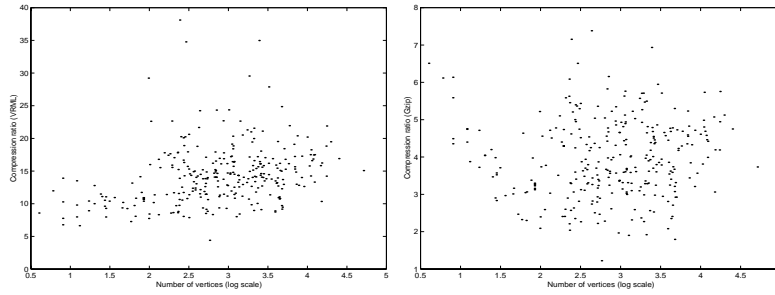


Figure 7: Compression ratio plotted for a lossless encoding of 300 CAD models. (left) with respect to the original VRML file size; (right) with respect to the gzipped file size.

- [6] D. Le Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM, CACM*, 34(4):46–58, 1991.
- [7] A. Gueziec, G. Taubin, F. Lazarus, and W. Horn. Cutting and Stitching: Efficient Conversion of a Non-Manifold Polygonal Surface to a Manifold. *Proceedings of IEEE Visualization 98*, 383–390, 1998.
- [8] J. Hartman and J. Wernecke. *The VRML 2.0 Handbook*. Addison-Wesley Publishing Company, 1996.
- [9] D. Huffman. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, 40(10):1098–1101, 1952.
- [10] J. Li and C. Kuo. Embedded Coding of Mesh Geometry. Technical report, ISO/IEC JTC1/SC29/WG11 MPEG98/M3325, March 1998, 1998.
- [11] J. Li, J. Li, and C. Kuo. Progressive Compression of 3D Graphic Models. In *IEEE Proceedings of Multimedia Computing and Systems*, Ottawa, Canada, 1997.
- [12] J. Li, J. Li, and C. Kuo. Progressive Coding of 3D Graphic Models. In *IEEE Multimedia and Systems*, 1998.
- [13] A. Moffat, R. Neal, and I. Witten. Arithmetic Coding Revisited. In *IEEE Data Compression Conference, Snowbird, Utah*, 1995.
- [14] G. Taubin and J. Rossignac. Geometric Compression through Topological Surgery. *ACM Transactions on Graphics*, 17(2):84–115, 1996.
- [15] C. Touma and C. Gotsman. Triangle Mesh Compression. In W. Davis, K. Booth, and A. Fourier, editors, *Proceedings of the 24th Conference on Graphics Interface (GI-98)*, pages 26–34, San Francisco, 1998. Morgan Kaufmann Publishers.
- [16] G. K. Wallace. The JPEG Still Picture Compression Standard. *Communications of the ACM, CACM*, 34(4):30–44, 1991.
- [17] G. Zhuang. *Compression and Progressive Transmission of Three-Dimensional Models*. PhD thesis, Department of Computer Sciences, Purdue University, 1998.