

A new method for simplification and compression of 3D meshes

MARCO ATTENE

Rapporto Tecnico N. 14/2001

Genova, Dicembre 2001

Abstract

We focus on the lossy compression of manifold triangle meshes. Our SwingWrapper approach partitions the surface of an original mesh M into simply-connected regions, called *triangloids*. We compute a new mesh M' . Each triangle of M' is a close approximation of a pseudo-triangle of M . By construction, the connectivity of M' is fairly regular and can be compressed to less than a bit per triangle using EdgeBreaker or one of the other recently developed schemes. The locations of the vertices of M' are compactly encoded with our new prediction scheme, which uses a single correction parameter per vertex. For example, a variety of popular models retiled with our approach yield 10 times fewer triangles without exceeding an error of 1% of the radius of the bounding ball. Vertices of M' are encoded with an average of 6 bits, which results in a total storage of 0.4 bits per triangle of the original mesh. The proposed solution may also be used to encode crude meshes for adaptive transmission and for controlling subdivision surfaces.

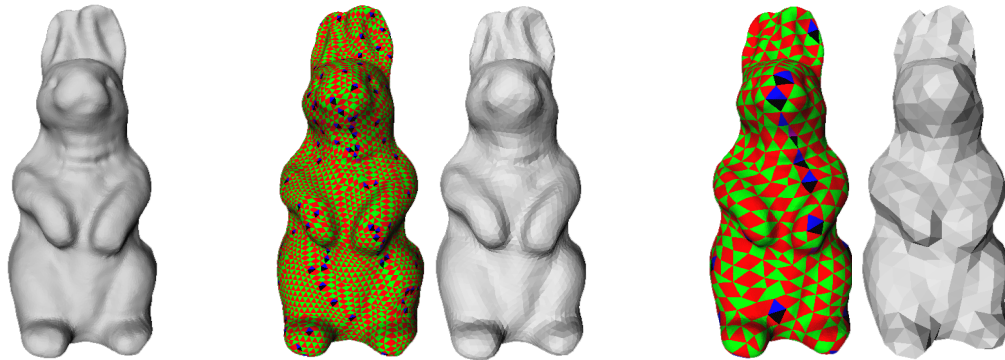


Figure 1: The original model (first from left) containing 134,074 triangles requires 4,100,000 bytes when stored as a wrl file; A dense partitioning of its surface into regular triangloids (second) was produced by SwingWrapper and color-coded according to their EdgeBreaker labels; The corresponding retiled triangle mesh (third) was generated by flattening the triangloids. It deviates from the original by less than 0.6% of the radius of the bounding ball. Its 13642 triangles were encoded with 3.5 bits per triangle using Edgebreaker's connectivity compression combined with a novel geometry predictor. The resulting total of 6042 bytes represents a 678-to-1 compression ratio; A coarser partitioning (fourth) decomposes the original surface into 1505 triangloids. The corresponding retiling (last) approximating the original within 4%, is encoded with 980 bytes: A 4000-to-1 compression).

INTRODUCTION

3D graphics plays an increasingly important role in applications where 3D models are accessed through the Internet. Due to improved design and model acquisition tools, to the wider acceptance of this technology, and to the need for higher accuracy, the number and complexity of these models are growing more rapidly than phone, network, and air bandwidth. Consequently, it is imperative to continue increasing the terseness of 3D data transmission formats and the performance and reliability of the associated compression and decompression algorithms.

Although many representations have been proposed for 3D models, polygon and triangle meshes are the de facto standard for exchanging and viewing 3D models, because they are simple to generate, store, and process, and because they are well supported by graphics adapters, APIS and standards. Triangle meshes that form accurate representations of 3D shapes involve large numbers of triangles and thus require a significant amount of storage space or transmission time. Thus it is important to compress them.

A triangle mesh may be represented by its vertex data and by its connectivity. Vertex data comprises coordinates of the vertices and optionally the vertex colors or normal and texture coordinate. In its simplest form, connectivity captures the incidence relation between the triangles of the mesh and their bounding vertices. It may be represented by a triangle-vertex incidence table, which associates with each triangle the references to its three bounding vertices.

Many applications do not require that the exact original mesh (which is often an approximation of some real object or of an ideal curved shape) be preserved. Thus it is appropriate and often advantageous to use lossy compression.

The SwingWrapper approach proposed here and illustrated in Figure 1 produces an approximating mesh particularly well suited for compression.

Simplification and compression have been separated in the past. Simplification was focused on reducing the triangle count while minimizing or not exceeding some error estimate. Most compression techniques have been lossless (except for the quantization of the vertex coordinate). We combine them here, proposing a particular retiling approach that reduces the triangle-count while generating a meshes particularly suitable for compression, because it is fairly regular (so that its connectivity can be encoded concisely) and because the locations of its vertices are constrained (so that they are each defined by a single parameter) and quantized (so that the difference between their predicted and actual location can be encoded with few bit integer).

PRIOR ART

For manifold meshes with few handles, the number of triangles is roughly twice the number of vertices. Consequently, when pointers or integer indices are used as vertex-references and when floating point coordinates are used to encode vertex locations, uncompressed connectivity data consumes twice more storage than vertex coordinates. This resulted in the development of a number of algorithms for the compression of the connectivity. The connectivity of a triangle mesh that is homeomorphic to a sphere corresponds to a planar graph. Compact encodings of such graphs and their worst case bounds have been studied for over 40 years [1] and remain a vibrant research topic [17][3][4][18][19][20][21]. Recent results guarantee less than 2 bits per triangle encodings for the connectivity of such meshes [19][18]. The connectivity of regular meshes, where most triangles have exactly six neighbors, may be encoded at a significantly lower cost [4][21][24]. To benefit from these advances, we strive to produce fairly regular retillings, however, the main benefit of the SwingWrapper approach described in this paper lies in the way in which we constrain vertex locations, so as to reduce their storage.

3D compression schemes developed over the last seven years have already impacted the design of hardware graphic adapters [2], of the MPEG-4 standard [3], and of 3D graphics software products [4]. These schemes encode triangle meshes and thus allocate a significant amount of storage to the precise location of vertices and to the connectivity of the mesh. Yet, most applications do not require that the precise vertex positions on the surface and the original mesh connectivity be preserved. Instead, there is a growing need for the most effective compression technique capable of transmitting a sufficiently close geometric approximation of a surface and of its photometric properties. Techniques based on a retiling of the model promise to provide higher compression ratios. For example, a mesh can be retiled [34][35] by inserting few new vertices, distributed regularly with a density that may be uniform or that may depend on the local curvature. In a second step, the numerous original vertices are removed. In a different approach, when the approximating surface is formulated as the result of a regular subdivision process applied to a coarse triangle mesh [5], the cost of storing connectivity is drastically reduced. When vertices are constrained to lie on specific rays emanating from a coarse mesh [6], their position may be encoded using a single coordinate. These approaches involve the delicate process of establishing a one-to-one mapping between the original surface and an approximating triangle mesh.

In many situations, considerable savings may be achieved by initially transmitting a crude approximation and by holding off the transmission of its refinements until they become necessary, so several methods for a progressive encoding [7][8][9][10][11] have been developed.

3D compression has also been investigated for meshes with properties [12][13] and it has been improved for rendering and visualization purposes [14][15], where, when a slight loss of information is tolerable, high frequencies can be simplified [16].

Although it may easily be formulated as a modification of schemes that were designed to take advantage of the regularity of the connectivity of the mesh [4][21], for simplicity, we chose to implement SwingWrapper as a modification of the EdgeBreaker compression scheme [19][24].

As several other compression schemes [3][4][17], Edgebreaker visits the triangles in a spiraling (depth-first) triangle-spanning-tree order and generates a string of descriptors, one per triangle, which indicate how the mesh can be rebuilt by attaching new triangles to previously reconstructed ones. The popularity of Edgebreaker lies in the fact that all descriptors are symbols from the set $\{C,L,E,R,S\}$. A particular edge separating a previously processed triangle from one that has not yet been processed is called the gate. The tip of the triangle, its left and right edges and neighbor triangles are defined with respect to that gate. At each step of the Edgebreaker compression and decompression, the unprocessed triangle attached at the gate is processed and a new gate is selected. When the tip of the new triangle corresponds to a vertex that has not been previously visited, the triangle is associated with the

symbol C. Otherwise, only four cases are possible. If only the right neighbor of the new triangle has been previously visited, the new triangle is labeled R and its left edge becomes the gate. The symmetric situation corresponds to the L label. When both neighbors have not been visited, the algorithm starts a recursion on the right edge as gate and then resumes processing with the left edge as gate. When both neighbors have been visited, the recursion returns or the algorithm terminates.

No other parameter is needed. Because half of the descriptors are Cs, a trivial code (C=0, L=110, E=111, R=101, S=100) guarantees 2 bits per triangle. A slightly more complex code guarantees 1.73 bits per triangle [18]. For large meshes, entropy codes further reduce the storage to less than a bit per triangle [24]. The string of descriptors produced by Edgebreaker is called the *clers* string. (*No relation with any of the strings pulled by Claire.*) An efficient decompression algorithm for the *clers* sequence [24] interprets the symbols to build a simply connected triangulated polygon, which represents the triangle-spanning tree. Then, it zips up the borders of that polygon by matching pairs of its bounding edges in a bottom-up order with respect to the vertex-spanning-tree that is the dual of the triangle-spanning-tree. An alternative, called Spirale Reversi [25], interprets the reversed *clers* string and builds the triangle tree from the end. The Edgebreaker compression scheme has been extended to manifold meshes with handles and holes [19], to triangulated boundaries of non-manifold solids [22], and to meshes that contain only quadrilaterals or a combination of simply-connected polygonal faces with an arbitrary number of sides [23]. It was also optimized for meshes with nearly regular connectivity [20]. Nevertheless, for sake of simplicity, in this paper, we restrict our focus to manifold and orientable closed T-meshes.

Vertex coordinates may be compressed through various forms of vector quantization [2][3][26]. Most vertex compression approaches exploit the coherence in vertex locations by using local or global predictors to encode corrections instead of absolute vertex data. Both the encoder and the decoder use the same prediction formula. The encoder transmits the difference between the predicted and the correct vertex data. It uses variable length codes for the corrections. The better the prediction, the shorter the codes. The decoder receives the correction, decodes it and adds it to the predicted data to obtain the correct information for the next vertex. Thus the prediction can only exploit data that has been previously received and decoded. Most predictive schemes require only local connectivity between the next vertex and its previously decoded neighbors. Some global predictors require having the connectivity of the entire mesh. Thus it is imperative to optimize connectivity compression techniques that are independent of vertex data.

Mesh simplification algorithms are used to reduce the number of polygons of the input model. Existing approaches are mainly based on vertex [27], edge [7][28][29] or face [30][31] simplification primitives; roughly speaking, at each step the algorithm chooses an element to be collapsed, depending either on some error metrics or on characteristics such as the surface curvature, then the selected element is eliminated and the influenced region is re-triangulated. Most of these simplification techniques are particular versions of vertex clustering. The simplest and most efficient vertex clustering, [32][33] overlays a 3D grid on the model and collapses all vertices within each cell of the grid to the single most important vertex within the cell.

Some simplification techniques provide a bound or an estimate on the error between the simplified and the original models. Evaluating the difference between two 3D models is complex [36] and may be approached in different ways. For example, in [7] energy functions have been used to measure the total squared distance, while in [31] the distances of original vertices and the simplified surface are used. In this paper, we use the *Hausdorff Distance* to evaluate the error. By definition, two surfaces are within Hausdorff distance r from each other if and only if each point on one surface is within distance r from some point of the other one and vice-versa. The publicly available *Metro Tool* [37], uses one-directional version of the Hausdorff distance for evaluating simplification algorithms.

The SwingWrapper approach presented here and the Normal Mesh encoding [6] are both lossy compression techniques based on a retiling of the original mesh and on a constraint that restricts the location of each vertex to lie on sampled points along a specific curve. In the case of the Normal Meshes, a crude mesh is constructed first and then subdivided in an adaptive manner, restricting the location of the refinement vertices to lie on the intersection between an estimated normal and the original surface. Situations where the normal does not intersect the original surface or where the topology of the resulting mesh would disagree with the topology of the original one must be detected and handled through local refinement. Recursive subdivision relies on a separate simplification process to construct the original mesh with the correct topology. Furthermore, that original mesh and the regular subdivision process limit the way in which the retiling can adapt to the local shape characteristics, and thus may result in less effective compression ratios. For example, regular meshes may lead to sub-optimal triangulations for surfaces with high curvature regions and saddle points, where vertices of valence different than 6 are more appropriate. By contrast, SwingWrapper does has the flexibility to adapt to connectivity to better fit the local shape. It also does not

require the computation of a starting crude mesh. Thus it appears to be more effective than Normal Meshes for encoding crude approximations or meshes with significant shape variations relative to the triangle size. SwingWrapper does not, by itself support progressive transmission, but could be used effectively to build and encode the crude meshes for subdivision surfaces and possibly for Normal Meshes.

SWING-WRAPPER

To understand how one can improve the compression of the geometry (vertex location), it is valuable to study what happens during the decompression of a *clers* string. Whenever we meet a ‘C’ symbol, we must create a new vertex, and thus need to obtain its three coordinates. Coordinates can be transmitted as absolute values or they can be predicted by the decompression and adjusted by corrective-vectors that are received as part of the transmission; in both cases we need to store three parameters for each vertex. Let us suppose that each C triangle is isosceles with the gate as its base and its left and right edges having both length L . The position of the tip vertex of the triangle is completely defined by the dihedral angle between the two triangles that are incident upon the gate. The decoder can estimate that angle to be 180 degrees or a pre-computed value that depends on the average curvature of the model. The decompression algorithm is notified by a single bit whether that estimate is sufficient to satisfy a predefined error tolerance. If not, that is a single number is decoded and used to adjust the dihedral angle. Furthermore, such a correction may be quantized with few bits without producing significant errors.

The desired length L of the sides of the C triangle may be chosen to equal the length of the gate (so as to form equilateral triangles) or to be identical throughout the mesh. When a triangle with the desired length would violate topological constraints or the desired Hausdorff distance limit, the decoder is notified to use a smaller value of L (computed so as to produce a triangle with half the height of the desired one). Note that L, R, and S triangles may produce edges, and thus gates, whose lengths vary, depending on the local curvature.

The scenario described above is the essence of our approach: Encode the connectivity using Edgebreaker and the vertex locations using the dihedral angle scheme.

1.1 Resampling M

To provide a retiling M' of a given original mesh M , we partition M into simply connected triangular-like regions, called *triangloids*. The triangloids correspond to the triangles of M' . Each triangloid is bounded by three piecewise linear paths on M , and each path corresponds to an edge of M' . Note that some triangles of M may not fall within a single triangloid. The algorithm subdivides them into smaller triangles that do. We describe in this section the retiling process. The initial steps are illustrated in Figure 2.

Let L denote the provided step and p_1 the first vertex of M' . The desired length, L , is either provided by the user or estimated from the desired accuracy and a crude analysis of the curvature of the model. The initial vertex p_1 is typically chosen randomly, although it may be specified by the user (Fig. 2a).

p_2 , the second vertex of M' , is taken on the curve C where M intersects the sphere centered in p_1 with radius L (Fig. 2b). For simplicity, we consider that M is connected. Thus, if C is empty, the desired distance L is considered too large for the model. If C has more than one component, we consider only the portion of C that bounds the part of M inside the sphere specified above. If C is non-manifold, it must be split into components.

Now, let e be the edge connecting p_1 and p_2 and let m_e be its middle point; the third point, p_3 , is on the two intersections between the mesh and the circle centered in m_e having for radius $L\sqrt{3}/2$ (the height of the equilateral triangle with edge length L) and lying on the plane perpendicular to e (Fig. 2c). The fourth sampling point, p_4 , is the other intersection. If the intersection is not made of two points, the provided sampling step L is reduced and the process restarted.

The above operations locate three points (p_1 , p_2 and p_3) on M defining the first equilateral triangle, t' , of M' . We refer to e as the base edge of t' . Now we must compute the three paths on M (Fig. 2d) that bound the corresponding triangloid t , then we must mark its inner elements so as to ensure that they are not later associated with another triangle of M' . Tracking a path may require the insertion of new vertices where the path crosses edges of M . Thus edge and/or triangle splits are performed in order to build a coherent partitioning of M . The paths are approximations of the corresponding geodesic shortest paths (GSP) and are computed as described in [38]. A second equilateral triangle (Fig. 2f) has for vertices p_1 , p_2 and p_4 . The two paths that bound the corresponding triangloid are computed using the same approach as above.

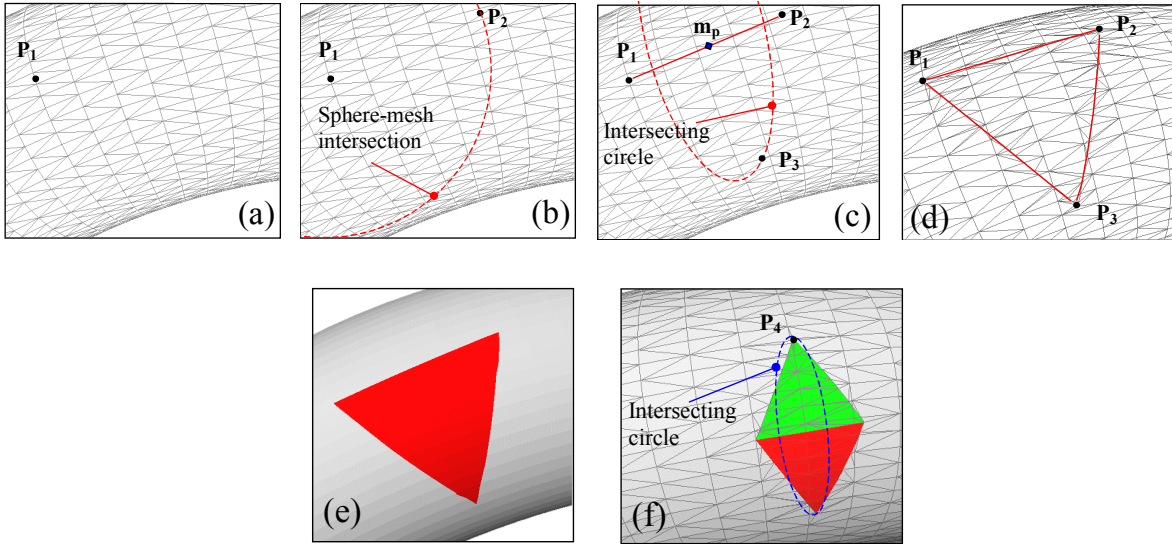


Figure 2: Construction of the initial triangles of M' .

To build each one of the remaining triangles, we start with a gate edge bounded by two points, say p_i and p_j , and locate a sample V on M . V must lie on the circle centered at the gate mid-point, having radius $L\sqrt{3}/2$, and lying on the bisecting plane between p_i and p_j . The circle and M may have two or more intersections. We select the intersection that is the furthest away from the tip of the previously decoded triangle bounded by the gate. If V is close enough to a previously decoded vertex that is part of the border of the decoded triangles, we select the closest of these, say W , and consider it to be the tip of the next triangle. If W is a neighbor of p_i and p_j (that is, they are connected by edges of M'), then the triangle corresponds to the EdgeBreaker label E . If it is only a neighbor of one of them, the new triangle is either R or L , depending whether W follows or precedes the gate (Fig. 3). Otherwise, we have an S triangle (Fig. 4). We also perform a snap if it happens that the new triangoloid to be created intersects an existing one.

If we perform the snap whenever the distance between V and W is less than $L/2$, we are guaranteed that no edge in the final triangulation is shorter than $L/2$ or longer than $3L/2$.

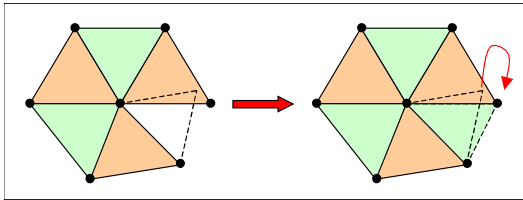


Figure 3: Simple Snap (EdgeBreaker's L or E)

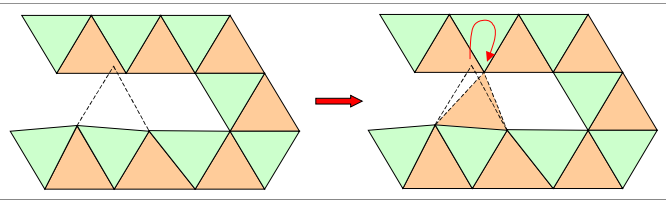


Figure 4: Complex Snap (EdgeBreaker's S)

1.2 Topological validity

At each step of the sampling procedure described above, we must compute and mark the corresponding triangoloid (by tracing the GSP paths that bound it) and also verify that it is simply connected and disjoint from previously encountered triangoloids.

When a triangoloid fails the test, we can adopt one of two strategies: shorten the length L for this particular triangle (adaptive strategy) or restart with a smaller constant L for all triangles (uniform strategy).

Adaptive strategy: This adaptive scheme lets us start with a large L so that fairly flat regions can be split into larger triangoloids than regions of high curvature. However, it requires a separate bitstream in the compression format to identify those triangles for which the length must be adjusted and the amount of adjustment. For example one bit per triangle could indicate whether the standard L is acceptable. If not, subsequent bits would indicate how much we

wish to reduce it.

Uniform strategy: This approach limits the magnitude of L for a given model and may result in an over-sampling of flat regions. Nevertheless, it is simpler and produces uniformly sampled meshes, which may have advantages for certain rendering and animation applications.

We have established experimentally that the uniform strategy produces excellent results for a large variety of models commonly used to report results of simplification and compression algorithms. Thus the rest of this paper is focused on uniform retiling.

1.3 Encoding

The described process can be used to generate a *clers* string representing the connectivity of the simplified triangulation and, since we are guaranteed that 'C' triangles are isosceles and their height is $L\sqrt{3}/2$, we can use the dihedral angle scheme for encoding the geometry. Summarizing, the compressed model is represented by the coordinates of the first triangle vertices, a *clers* string, and a sequence of dihedral angles. The mesh connectivity can be reconstructed using the Wrap&Zip algorithm [24]. The location of the first three vertices is explicit, while the others can be computed starting from the already reconstructed mesh and the dihedral angle information; the used sampling step does not need to be encoded because it is the distance between two of the three original vertices.

The geometry is encoded by quantizing each dihedral angle with a fixed number of bits. Using 8 bits guarantees a precision of $360/256 \cong 1.41$ degrees (corresponding to a Hausdorff distance of $\sin(1.41)L\sqrt{3}/2$). Since angles are quantized, vertices of the approximating mesh do not lie exactly on the original triangulation. To avoid error propagation during the decompression, the re-meshing must compute each new vertex starting from an approximating adjacent triangle, whose vertices do not necessarily lie on the original mesh. Thus the compression must simulate the work of the decompressor using only previously decoded information.

1.4 Results

The whole method guarantees that the re-mesh with V vertices, T triangles and H handles can always be encoded with $8V + 1.8T + 2H\log(T)$ bits. For simplicity, we assume that the number of handles, H , is negligible with respect to V , and thus that $T=2V$.

For large meshes, entropy encoding further reduces storage, bringing the connectivity cost down to about $1T$ bits and the geometry cost to about $6V$ bits (or equivalently $3T$ bits). Thus the total size is about $4T$ bits for large meshes, while is guaranteed not to exceed $5.8T$ bits for meshes without handles.

We have developed a prototype to test the described method. The input is the original mesh. The first step of the SwingWrapper system is to compute the radius of the smallest bounding ball and to let the user express the desired edge length L as a ratio of this radius. Then, SwingWrapper attempts to compute a retiling and either reports failure, in which case the user may try a smaller L , or produce a valid retiling and report the largest error measured as the Hausdorff distance between each triangle of M' and the corresponding triangloid.

We have found that many of the original triangle meshes that have been used to demonstrate simplification and compression results in the literature may be retiled with our approach down to 10 times fewer triangles without exceeding an error of 1% of the radius of the smallest ball that contains the model. Thus the resulting compression yields a **guaranteed** storage of only **0.58 bits per triangle** of the original model, and an expected **0.4 bits per triangle** for large meshes.

To report our results in this paper, we have used models downloaded from <http://www.cyberware.com/samples/index.html>. They are shown in Table 1 and in figs. 5, 6 and 7, where the Santa Claus model has been compressed within two different error tolerances; notice that an error of 0.8% of the bounding ball results in a model that is visually indistinguishable from the original one, while as this error increases the resulting model becomes smoother.

We tested our prototype on a PentiumIII 450 equipped with 512M of ram and running Linux and, excluding input/output operations, statistics on running time reported an average of $1.2 \cdot 10^{-4}$ seconds per triangle of the original mesh M (over 8000 triangles per second).

CONCLUSIONS

In this paper, we have presented a new method for compressing triangle meshes with a controlled loss of information. We exploited both simplification and compression techniques in order to obtain a compact encoding. The method described here is suitable for improvements in the encoding scheme; for example, we found that an average of 85% of the vertices of the resulting model have exactly six incident triangles, so valence driven encoding schemes may give further compression of the connectivity.

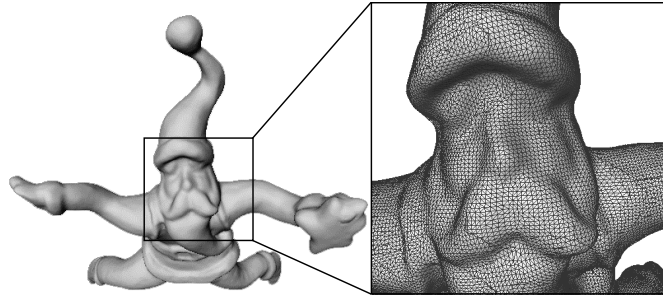
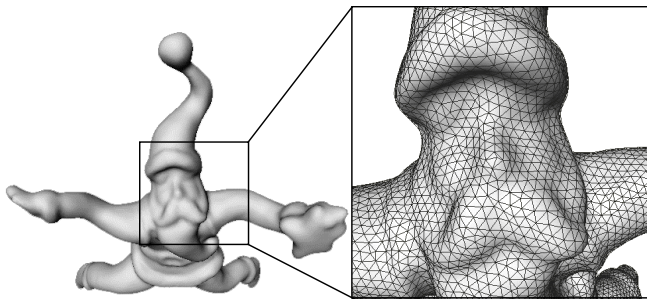
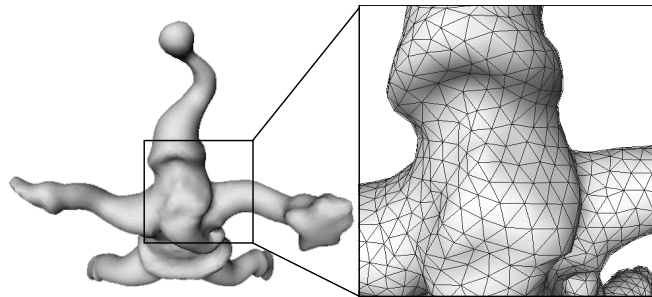


Figure 5: Original mesh: 151560 triangles, vrml file size: 4774373 bytes



*Figure 6: Re-mesh with 0.8% error with 23183 triangles built in 16 seconds and encoded with 12329 bytes;
compression rate: 0.65 bits/T.*



*Figure 7: Re-mesh with 1.9% error with 6474 triangles built in 15 seconds and encoded with 3573 bytes;
compression rate: 0.189 bits/T.*













 <p>Original: 274K triangles WRL file: 8.7 Mbytes</p>	 <p>$\epsilon = 0.5\%$: 56K triangles encoded with 29958 bytes</p>	 <p>$\epsilon = 1.1\%$: 24K triangles encoded with 12906 bytes</p>	 <p>$\epsilon = 4.3\%$: 7K triangles encoded with 4116 bytes</p>
 <p>Original: 268K triangles WRL file: 8.5 Mbytes</p>	 <p>$\epsilon = 0.4\%$: 62K triangles encoded with 37072 bytes</p>	 <p>$\epsilon = 1.6\%$: 18K triangles encoded with 10314 bytes</p>	 <p>$\epsilon = 4.1\%$: 9K triangles encoded with 5624 bytes</p>
 <p>Original: 375K triangles WRL file: 11.9 Mbytes</p>	 <p>$\epsilon = 2.7\%$: 18K triangles encoded with 9312 bytes</p>	 <p>Original: 80K triangles WRL file: 2.54 Mbytes</p>	 <p>$\epsilon = 1.1\%$: 10K triangles encoded with 6424 bytes</p>

Table 1: Retilings produced with SwingWrapper.

REFERENCES

- [1] W. Tutte, "A census of planar triangulations", Canadian Journal of Mathematics, pp.21-38, 1962.
- [2] M. Deering. Geometry compression. In Computer Graphics (SIGGRAPH '95 Proceedings), pp. 13-20, 1995.
- [3] G. Taubin and J. Rossignac, "Geometric Compression through Topological Surgery", ACM Transactions on Graphics, 17(2), 84-115, April 1998.
- [4] C. Touma and C. Gotsman, "Triangle Mesh Compression", Proceedings Graphics Interface 98, pp. 26-34, 1998.
- [5] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle, Multiresolution Analysis of Arbitrary Meshes, Proc. ACM SIGGRAPH'95, pp. 173-182, Aug. 1995.
- [6] I. Guskov, K. Vidimce, W. Sweldens, P. Schroder, Normal Meshes, Proc. SIGGRAPH, pp. 95-102, 2000.
- [7] H. Hoppe, "Progressive Meshes", Proc. ACM Siggraph'96, pp. 99-108, August 1996.
- [8] C.L. Bajaj and V.Pascucci and G.Zhuang. "Progressive Compression and Transmission of Arbitrary Triangular Meshes", IEEE Visualization '99, pp. 307-316, October 1999.
- [9] D. Cohen-Or, D. Levin and O. Remez, "Progressive Compression of Arbitrary Triangular Meshes". In Proc. of Visualization 99, pp. 67-72, October 1999.
- [10] A. Khodakovsky, P. Schroder and W. Sweldens, "Progressive Geometry Compression", Proc. of SIGGRAPH 2000, pp. 271-278, July 2000.
- [11] P. Alliez and M. Desbrun, "Progressive Encoding for Lossless Transmission of 3D Meshes", Proc. of SIGGRAPH 2001, pp., August 2001.
- [12] C.L. Bajaj and V. Pascucci and G. Zhuang, "Single Resolution Compression of Arbitrary Triangular Meshes with Properties", IEEE Data Compression Conference, 1999.
- [13] M. Isenburg and J. Snoeyink, "Face-Fixer: Compressing Polygon Meshes with Properties", Proc. of SIGGRAPH 2000, pages 263-270, July 2000.
- [14] M. Chow, "Optimized Geometry Compression for Real-time Rendering", In Proc. of the IEEE Visualization '97, pp. 346-354, November 1997.
- [15] M.H. Gross, L. Lippert and O.G.Stadt, "Compression methods for Visualization", Future Generation Computer Systems, Vol. 15, No. 1, pp.11-19, 1999.
- [16] Z. Karni and C. Gotsman, "Spectral Coding of Mesh Geometry", Proc. of SIGGRAPH 2000, pp. 279-286, July 2000.
- [17] S. Gumhold and W. Strasser, "Real Time Compression of Triangle Mesh Connectivity", Proc. ACM Siggraph, pp. 133-140, July 1998.
- [18] D. King and J. Rossignac, "Guaranteed 3.67V bit encoding of planar triangle graphs", 11th Canadian Conference on Computational Geometry (CCCG'99), pp. 146-149, Vancouver, CA, August 15-18, 1999.
- [19] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes", IEEE Transactions on Visualization and Computer Graphics, 5(1), 47-61, Jan-Mar 1999.
- [20] A. Szymczak, D. King, J. Rossignac, "An Edgebreaker-based Efficient Compression Scheme for Connectivity of Regular Meshes", Journal of Computational Geometry: Theory and Applications, 2000.
- [21] P.Alliez and M.Desbrun, "Valence-Driven Connectivity Encoding for 3D Meshes", Proc. of Eurographics '2001.
- [22] J. Rossignac and D. Cardoze, "Matchmaker: Manifold Breps for non-manifold r-sets", Proceedings of the ACM Symposium on Solid Modeling, pp. 31-41, June 1999.
- [23] D. King and J. Rossignac, "Connectivity Compression for Irregular Quadrilateral Meshes" Research Report GIT-GVU- 99 -29, Dec 1999.
- [24] J. Rossignac and A. Szymczak, "Wrap&Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker", Computational Geometry, Theory and Applications, 14(1/3), 119-135, November 1999.
- [25] M. Isenburg and J. Snoeyink, "Spirale Reversi: Reverse decoding of the Edgebreaker encoding", Tech. Report TR-99-08, Computer Science, UBC, 1999.
- [26] M. Garland and P. Heckbert. Simplifying Surfaces with Color and Texture using Quadratic Error Metric. Proceedings of IEEE Visualization, pp. 287-295, 1998.
- [27] W. Schroeder, J. Zarge and W.E. Lorensen, "Decimation of triangle meshes", Proc. ACM Siggraph 92, pp. 65-70, July 1992.
- [28] M. E. Algorri and F. Schmitt, "Mesh simplification", Proc. Eurographics 96, 15(3), pp. 78-86, 1996.
- [29] R. Ronfard and J. Rossignac, "Full range approximation of triangulated polyhedra", Proc. Eurographics 96, 15(3), pp. 67-76, 1996.
- [30] P. Hinker and C. Hansen, "Geometric Optimization", IEEE Visualization '93 Proc., pp 189-195, October, 1993.
- [31] A.D. Kalvin and R.H. Taylor, "Superfaces: Polygonal mesh simplification with bounded error". IEEE Computer Graphics and Applications, 16(3), pp. 64-67,1996.
- [32] J. Rossignac and P. Borrel, "Multi-resolution 3D approximations for rendering complex scenes", Geometric Modeling in

Computer Graphics, Springer Verlag, Berlin, pp. 445-465, 1993.

- [33] K-L. Low and T.S.Tan, "Model Simplification using vertex clustering", Proc. Symp. Interactive 3D Graphics, ACM Press, NY, pp. 75-82, 1997.
- [34] G. Turk, "Re-tiling polygonal surfaces", Proc. ACM Siggraph 92, pp. 55-64, July 1992.
- [35] M. Attene, S. Biasotti and M. Spagnuolo, "Re-meshing techniques for topological analysis", Proc. Shape Modeling International, pp. 142-151, 2001.
- [36] A. Khodakovsky, P. Schroder, W. Sweldens, Progressive Geometry Compression, Proc. ACM SIGGRAPH, pp. 271-278, 2000.
- [37] P. Cignoni, C. Rocchini and R. Scopigno, "Metro: measuring error on simplified surfaces", Proc. Eurographics '98, vol. 17(2), pp 167-174, June 1998.
- [38] K. Polthier, M. Schmies, "Geodesic Flow on Polyhedral Surfaces", Procs. of Eurographics Workshop on Scientific Visualization, Vienna 1999.