RECENT ADVANCES IN COMPRESSION OF 3D MESHES

Pierre Alliez

INRIA Sophia-Antipolis
06902 Sophia Antipolis cedex, FRANCE
phone: +33 4 92 38 76 77, fax: +33 4 92 38 76 43, email: pierre.alliez@sophia.inria.fr
http://www-sop.inria.fr/geometrica/team/Pierre.Alliez/

ABSTRACT

3D meshes are widely used in graphic and simulation applications for approximating 3D objects. When representing complex shapes in a raw data format, meshes consume a large amount of space. Applications calling for compact storage and fast transmission of 3D meshes have motivated the multitude of algorithms developed to efficiently compress these datasets. In this paper we survey recent developments in compression of 3D surface meshes. We also list some open questions and directions for future research.

1. INTRODUCTION

The emerging demand for visualizing and simulating 3D geometric data in networked environments has motivated research on representations for such data. Slow networks require data compression to reduce the latency, and progressive representations to transform 3D objects into streams manageable by the networks. We distinguish between single-rate and progressive compression techniques depending on whether the model is decoded during, or only after, the transmission. In the case of single-rate lossless coding, the goal is to remove the redundancy present in the original description of the data. In the case of progressive compression the problem is more challenging, aiming for the best trade-off between data size and approximation accuracy (the so-called rate-distortion tradeoff). Lossy single-rate coding may also be achieved by modifying the data set, making it more amenable to coding, without losing too much information. These techniques are called *remeshing*.

1.1 Basic Definitions

The specification of a polygon surface mesh consists of combinatorial entities: vertices, edges, and faces, and numerical quantities: attributes such as vertex positions, normals, texture coordinates, colors, etc. The *connectivity* describes the incidences between elements and is implied by the topology of the mesh. For example, two vertices or two faces are adjacent if there exists an edge incident to both. The *valence* of a vertex is the number of edges incident to it, and the *degree* of a face is the number of edges incident to it (see Fig. 1). The *ring* of a vertex is the ordered list of all its incident faces. The total number of vertices, edges, and faces of a mesh will be denoted V, E, and F respectively.

We classify the techniques into two classes:

- Techniques aiming at coding the original mesh without making any assumption about its complexity, regularity or uniformity. This also includes techniques specialized for massive datasets, which cannot fit entirely into main memory. Here we aim at restoring the original model after decoding (for carefully designed models or applications where lossy compression is intolerable).
- Techniques which remesh the model before compression. The original mesh is considered as just one instance of the shape geometry.

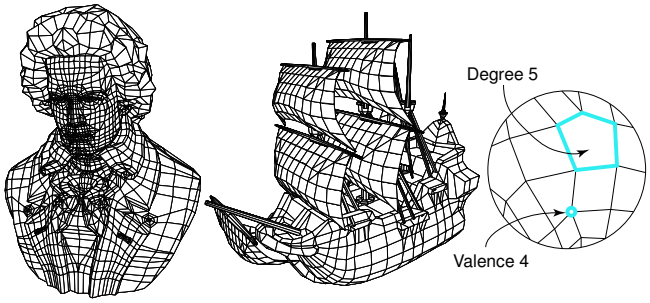

Figure 1: Examples of polygon meshes: (left) Beethoven mesh (2812 polygons, 2655 vertices) - (right) Galleon mesh (2384 polygons, 2372 vertices). Close-up of a polygon mesh: the *valence* of a vertex is the number of edges incident to this vertex, while the *degree* of a face is the number of edges enclosing it.

2. TRIANGLE MESHES

The triangle is the basic geometric primitive for standard graphics rendering hardware and for many simulation algorithms. This partially explains why much of the work in the area of mesh compression prior to 2000 has been concerned with triangle meshes only. The *Edgebreaker* coder [26] gives a worst-case bound on the connectivity compression bit rate of 4 bits per vertex. Besides the popular *Edgebreaker* and its derivatives [19, 8, 30, 14], two techniques transform the connectivity of a triangle mesh into a sequence of valence codes [31, 12], hence can automatically benefit from the low statistical dispersion around the average valency of 6 when using entropy encoding. This is achieved either through a deterministic conquest [31] or by a sequence of half edge collapses [12]. In [31], Touma and Gotsman proposed the conquest approach and compress the connectivity down to less than 0.2 bit per vertex (b/v) for very regular meshes, and between 2 and 3.5 b/v otherwise, in practice. The so-called conquest consists of conquering the edges of successive pivot vertices in an orientation-consistent manner and generating valence codes for traversed vertices. Three additional codes: *dummy*, *merge* and *split* are required in order to encode boundaries, handles and conquest incidents respectively. The *dummy* code occurs each time a boundary

is encountered during the conquest; the number of *merge* codes is equal to the genus of the mesh being encoded. The *split* code frequency is linked mainly to the mesh irregularity. Intuitively, if one looks at the coding process as a conquest along a spiraling vertex tree, the *split* codes thus indicate the presence of its branching nodes. The *Mesh Collapse Compression* scheme by Isenburg and Snoeyink [12] performs a sequence of edge contractions until a single vertex remains in order to obtain bit rates of 1 to 4 b/v. For a complete survey of these approaches, we refer the reader to [6, 3].

3. POLYGON MESHES

Compared with triangle meshes, little work has been dedicated to the harder problem of connectivity coding of 2-manifold graphs with arbitrary face degrees and vertex valences. There are a significant number of non-triangular meshes in use, in particular those carefully designed contain a surprisingly small proportion of triangles. Likewise, few triangles are generated by tessellation routines in existing modeling softwares. The dominant element in these meshes is the quadrilateral, but pentagons, hexagons and higher degree faces are also common.

The performance of compression algorithms is traditionally measured in bits per vertex (b/v) or bits per edge (b/e). Some early attempts to code general graphs [32, 15], which are the connectivity component of a geometric mesh, led to rates of around 9 b/v. These methods are based on building interlocking spanning trees for vertices and faces. Chuang et al. [4] later described a more compact code using canonical ordering and multiple parentheses. They state that any simple 3-connected planar graph can be encoded using at most $1.5\log_2(3)E + 3 \simeq 2.377$ bits per edge. Li and Kuo [23] proposed a so-called "dual" approach that traverses the edges of the dual mesh and outputs a variable length sequence of symbols based on the type of a visited edge. The final sequence is then coded using a context based entropy coder. Isenburg and Snoeyink coded the connectivity of polygon meshes along with their properties in a method called *Face Fixer* [13]. This algorithm is gate-based, the gate designating an oriented edge incident to a facet that is about to be traversed. A complete traversal of the mesh is organized through successive gate labeling along an active boundary loop. As in [31, 26], both the encoder and decoder need a stack of boundary loops. Seven distinct labels F_n, R, L, S, E, H_n and $M_{i,k,l}$ are used in order to describe the way to *fix* faces or holes together while traversing the current active gate. King et al. [20], Kronrod and Gotsman [21] and Lee et al. [22] also generalized existing methods to quad, arbitrary polygon and hybrid triangle-quad meshes respectively. However, none of these polygon mesh coders come close to the bit rates of any of the best, specialized coders [31, 2] when applied to the special case of a triangle mesh. At the intuitive level, given that a polygon mesh with the same number of vertices contains fewer edges than a triangle mesh, it should be possible to encode it with fewer bits. These observations motivated the design of a better approach to code the connectivity of polygonal meshes.

Since the Touma-Gotsman (TG) valence coder [31] is generally considered to have the best performance, it seems natural to try to generalize it to arbitrary polygon meshes. This was done independently by Khodakovsky et al. [16] and Isenburg [10]. The generalization relies on the key concept of *duality*. Consider an arbitrary 2-manifold triangle graph $\mathscr{M}$. Its dual graph $\tilde{\mathscr{M}}$, in which faces are represented as dual vertices and vertices become dual faces, should have the same connectivity information since dualization neither adds nor removes information. The valences of $\tilde{\mathscr{M}}$ are now all equal to 3, while the face degrees take on the same values as the vertex valences of $\mathscr{M}$. Since a list of all 3s has zero entropy, coding just the list of degrees of $\tilde{\mathscr{M}}$ would lead to the same bit rate as found for the valences of $\mathscr{M}$. Conversely, if a polygon mesh has only valence-3 vertices, then its dual would be a triangle mesh. Hence, its entropy should be equal to the entropy of the list of its degrees. This observation leads to the key idea of the degree/valence approach : the compression algorithm should be *self-dual*, in the sense that both a mesh and its dual are coded with the same number of bits. A direct consequence of this is that the coding process should be symmetric in the coding of valences and degrees. A second direct consequence is that the bit rate of a mesh should be measured in *bits per edge* (b/e), since the number of edges is the only variable not changing during a graph dualization. This contrasts with the former practice of measuring the coding efficiency for triangle meshes in bits/vertex.

The core technique underlying the algorithm described in [10, 16] is similar to most connectivity compression methods: a seed element is chosen and all its neighbors are traversed recursively until all elements of the corresponding connected component are "conquered". A new seed element of the next connected component is then chosen and the process continues. Every time the encoder conquers the next element of the mesh, it outputs some symbol which uniquely identifies a new state. From this stream of symbols, the decoder can reconstruct the mesh. Various coding algorithms differ in the way they traverse the mesh and in the sets of symbols used for identifying the encoder state. During the mesh traversal of [10, 16], two sets of symbols are generated to code vertex valences and face degrees using an entropy encoder. At any given moment in time, both encoder and decoder know with which type of symbol (face or vertex) they are dealing. While the valence and degree sequences of a mesh dominate the mesh code, they are not sufficient to uniquely characterize it. As in [31], some extra "split", and possibly other symbols may be required during the mesh conquest. To minimize the occurrence of such symbols – hence improve the compression ratios – both techniques [10, 16] drive the traversal by various heuristics inspired from the valence-driven approach [2]. To better exploit correlation between streams and between symbols within each stream, it is possible to use a *context-based* arithmetic coder.

4. REMESHING FOR GEOMETRY COMPRESSION

The majority of mesh coders adapt to the regularity and the uniformity of the meshes (with the noticeable exception of [5] that adapts to the non-uniformity). Therefore, if the application allows lossy compression, it is prudent to exploit the existing degrees of freedom in the meshing process to transform the input into a mesh with high regularity and uniformity. Recent work produces either (i) semi-regular

meshes by using the subdivision paradigm, or (ii) highly regular remeshing by local mesh adaptation, or (iii) perfectly regular remeshing by surface cutting and global parameterization.

The main idea behind semi-regular remeshing techniques [18, 9, 17, 24] is to consider a mesh representation to have three components: geometry, connectivity and parameterization, and assume that the last two components (i.e. connectivity and parameterization) are not important for the representation of the geometry. The common goal of these approaches is therefore to reduce these two components as much as possible. This is achieved through semi-regular remeshing of an input irregular mesh, and efficient compression of the newly generated model. The remesher proceeds by building a semi-regular mesh hierarchy designed to best approximate the original geometry. An irregular base mesh, homeomorphic (i.e. topologically equivalent) to the original mesh, is first built by mesh simplification. This base mesh constitutes the coarsest level in the semi-regular hierarchy. The hierarchy is then built by regular or adaptive subdivision (typically by edge bisection) of the base mesh.

Surazhsky and Gotsman [29] generate a triangle mesh with user-controlled sample distribution and high regularity through a series of atomic Euler operators and vertex relocations applied locally. A density function is first specified by the user as a function of the curvature onto the original mesh. This mesh is kept for later reference to the original surface geometry, and the mesh adaptation process starts on a second mesh, initialized to a copy of the original mesh. The vertex density approaches the prescribed ideal density by local decimation or refinement. A new area-based smoothing technique is then performed to isotropically repartition the density function among the mesh vertices. A novel component of the remeshing scheme is a surprisingly efficient algorithm to improve the mesh regularity. The high level of regularity is obtained by performing a series of local edge-flip operations as well as some edge-collapses and edge-splits. The vertices are first classified as black, regular or white according to their valence deficit or excess (respectively < 6, $= 6$ and > 6). The edges are then classified as regular, long, short, or drifting according to their vertex colors (regular if both vertices are regular, long if both are white, short if both are black and drifting if bi-colored). Long edges are refined by edge-split, and short edges are removed by edge-collapse until only drifting edges remain. The drifting edges have the nice property that they can migrate through regular regions of the mesh by edge-flips without changing the repartition of the vertex valences. Improving the mesh regularity thus amounts to applying a sequence of drifting-edge migrations until they meet irregular vertices, and then have a chance to generate short or long edges whose removal becomes trivial. As a result the models are better compressed using the TG coder which benefits from the regularity in mesh connectivity and geometry.

Gu et al. [7] proposed a technique for completely regular remeshing of surface meshes using a rectangular grid. Surfaces of arbitrary genus must be cut to reduce them to a surface which is homeomorphic to a disc, then parameterized by minimizing a geometric-stretch measure [27], and finally represented as a so-called *geometry image* that stores the geometry, the normals and any attributes required for visualization purposes. Such a regular grid structure is compact and drastically simplifies the rendering pipeline since all cache indirections found in usual irregular mesh rendering are eliminated. Besides its appealing properties for efficient rendering, the regular structure allows direct application of "pixel-based" image-compression methods. The authors apply wavelet-based coding techniques and compress separately the topological sideband due to the cutting. After decoding, the topological sideband is used to fuse the cut and ensure a proper welding of the surface throughout the cuts. Despite its obvious importance for efficient rendering, this technique reveals a few drawbacks due to the inevitable surface cutting: each geometry image has to be homeomorphic to a disk, therefore closed or genus> 0 models have to be cut along a cut graph to extract either a polygonal schema [7] or an atlas [28]. Finding a "smart" cut graph (i.e. minimizing a notion of distortion) is a delicate issue and introduces a set of artificial boundary curves, associated pairwise. These boundaries are later sampled as a set of curves (i.e. 1-manifolds) and therefore generate a visually displeasing seam tree. Another drawback comes from the fact that the triangle or the quad primitives of the newly generated meshes have neither orientation nor shape consistent with approximation theory, which makes this representation not fully optimized for efficient geometry compression as reflected in the rate-distortion tradeoff.

5. COMPARISON AND DISCUSSION

A recent trend in mesh connectivity compression is generalization from triangle meshes to general polygon meshes, with arbitrary genus and boundaries. Adapting to the regularity of the mesh, i.e. the dispersion in the distribution of valences or degrees, usually reflects in the coding schemes. Semi-regularity being a common property of "real-world" meshes, this is a very convenient feature.

On the theoretical side, the bit-rates achievable by degree/valence connectivity coders have been shown to approach the Tutte entropy lower bound. Because of some remaining "split" symbols, whose number has not been bounded, some additional work has to be done in order to design truly optimal polygon mesh coders which also adapt to regularity. In particular, the connectivity coder of Poulalhon and Schaeffer [25] for triangle meshes offers some promise for extension to polygonal models. As for volume meshes, although some recent work has demonstrated a generalization of the valence coder to hexahedral meshes [11], nothing has been proven concerning the optimality of this approach. In order to benefit most from the adaptation of a coding scheme to regularity or uniformity in the input mesh, recent work advocates highly (or even completely) regular remeshing without distorting the geometry too much. In particular, the geometry images [7] technique demonstrates the efficiency of modern image compression techniques when applied to geometry which has been remeshed in a completely regular manner.

A more recent trend takes the remeshing paradigm further, with the design of efficient meshes for approximation of surfaces [1]. This leads to anisotropic polygon meshes,

that "look like" carefully designed meshes. The efficiency of such a scheme is expressed in terms of error per number of geometric primitives. The question that now naturally arises is whether the remeshing process should be influenced by the mesh compression scheme used, namely, should it remesh in a manner that suits the coder best. Since rapid progress in the direction of efficient surface meshing is emerging, it seems that it will certainly motivate new approaches for dedicated single-rate mesh compression schemes.

REFERENCES

[1] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun. Anisotropic Polygonal Remeshing. In *ACM SIGGRAPH Conference Proceedings*, 2003.

[2] P. Alliez and M. Desbrun. Valence-Driven Connectivity Encoding of 3D Meshes. In *Eurographics Conference Proceedings*, pages 480–489, 2001.

[3] Pierre Alliez and Craig Gotsman. *Recent Advances in Compression of 3D Meshes*. 2005.

[4] R.C-N. Chuang, A.Garg, X. He, M-Y. Kao, and H-I Lu. Compact Encodings of Planar Graphs via Canonical Orderings and Multiple Parentheses. In *ICALP: Annual International Colloquium on Automata, Languages and Programming*, pages 118–129, 1998.

[5] P.-M. Gandoin and O. Devillers. Progressive Lossless Compression of Arbitrary Simplicial Complexes. *ACM Transactions on Graphics*, 21:372–379, 2002. ACM SIGGRAPH Conference Proceedings.

[6] C. Gotsman, S. Gumhold, and L. Kobbelt. Simplification and Compression of 3D Meshes, 2002. In Tutorials on Multiresolution in Geometric Modelling (Munich Summer School Lecture Notes), A. Iske, E. Quak, M. Floater (Eds.), Springer, 2002.

[7] X. Gu, S. Gortler, and H. Hoppe. Geometry Images. In *SIGGRAPH Conf. Proc.*, pages 355–361, 2002.

[8] S. Gumhold. New Bounds on the Encoding of Planar Triangulations. Technical Report WSI-2000-1, Univ. of Tübingen, 2000.

[9] I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal Meshes. In *Proceedings of SIGGRAPH*, pages 95–102, 2000.

[10] M. Isenburg. Compressing Polygon Mesh Connectivity with Degree Duality Prediction. In *Graphics Interface Conference Proc.*, pages 161–170, 2002.

[11] M. Isenburg and P. Alliez. Compressing Hexahedral Volume Meshes. In *Pacific Graphics Conference Proceedings*, pages 284–293, 2002.

[12] M. Isenburg and J. Snoeyink. Mesh Collapse Compression. In *Proceedings of SIBGRAPI'99, Campinas, Brazil*, pages 27–28, 1999.

[13] M. Isenburg and J. Snoeyink. Face Fixer: Compressing Polygon Meshes With Properties. In *ACM SIGGRAPH Conference Proceedings*, pages 263–270, 2000.

[14] M. Isenburg and J. Snoeyink. Spirale Reversi: Reverse Decoding of the Edgebreaker Encoding. In *Proceedings of 12th Canadian Conference on Computational Geometry*, pages 247–256, 2000.

[15] Keeler and Westbrook. Short Encodings of Planar Graphs and Maps. *Disc. Ap. Math.*, 58:239–252, 1995.

[16] A. Khodakovsky, P. Alliez, M. Desbrun, and P. Schröder. Near-Optimal Connectivity Encoding of 2-Manifold Polygon Meshes. *Graphical Models, special issue*, 2002.

[17] A. Khodakovsky and I. Guskov. Compression of Normal Meshes. In *Geometric Modeling for Scientific Visualization*. Springer-Verlag, 2003.

[18] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive Geometry Compression. *ACM SIGGRAPH Conference Proceedings*, pages 271–278, 2000.

[19] D. King and J. Rossignac. Guaranteed 3.67V bit Encoding of Planar Triangle Graphs. In *11th Canadian Conference on Computational Geometry*, pages 146–149, 1999.

[20] D. King, J. Rossignac, and A. Szmczak. Compression for Irregular Quadrilateral Meshes. Technical Report TR–99–36, GVU, Georgia Tech, 1999.

[21] B. Kronrod and C. Gotsman. Efficient Coding of Non-Triangular Mesh Connectivity. *Graphical Models*, 63(263-275), 2001.

[22] H. Lee, P. Alliez, and M. Desbrun. Angle-Analyzer: A Triangle-Quad Mesh Codec. In *Eurographics Conference Proceedings*, pages 383–392, 2002.

[23] J. Li and C.-C. Jay Kuo. Mesh Connectivity Coding by the Dual Graph Approach, July 1998. MPEG98 Contribution Document No. M3530, Dublin, Ireland.

[24] F. Payan and M. Antonini. 3D Mesh Wavelet Coding Using Efficient Model-based Bit Allocation. In *Proceedings of 1st Int. Symposium on 3D Data Processing Visualization and Transmission*, pages 391–394, 2002.

[25] D. Poulalhon and G. Schaeffer. Optimal Coding and Sampling of Triangulations, 2003. 30th international colloquium on automata, languages and programming (ICALP'03).

[26] J. Rossignac. Edgebreaker : Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 1999.

[27] P. Sander, S. Gortler, J. Snyder, and H. Hoppe. Signal-Specialized Parametrization. In *Eurographics Workshop on Rendering 2002*, 2002.

[28] P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe. Multi-Chart Geometry Images. In *Proceedings of Eurographics Symposium on Geometry Processing*, 2003.

[29] V. Surazhsky and C. Gotsman. Explicit Surface Remeshing. In *Proceedings of Eurographics Symposium on Geometry Processing*, 2003.

[30] A. Szymczak, D. King, and J. Rossignac. An Edgebreaker-Based Efficient Compression Scheme for Regular Meshes. *Computational Geometry*, 20(1-2):53–68, 2001.

[31] C. Touma and C. Gotsman. Triangle Mesh Compression. *Graphics Interface 98 Conference Proceedings*, pages 26–34, 1998.

[32] G. Turan. Succinct Representations of Graphs. *Discrete Applied Mathematics*, 8:289–294, 1984.