Recent Advances in Compression of 3D Meshes

Pierre Alliez[1] and Craig Gotsman[2]

[1] INRIA, Sophia-Antipolis, France pierre.alliez@sophia.inria.fr
[2] Technion, Haifa, Israel gotsman@cs.technion.ac.il

Summary. 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 survey the main ideas and intuition behind techniques for single-rate and progressive mesh coding. Where possible, we discuss the theoretical results obtained for asymptotic behavior or optimality of the approach. 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*.

Section 2 gives some basic definitions for surface meshes. Section 3 surveys recent algorithms for single-rate compression, and Section 4 surveys recent techniques for progressive compression.

2 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.

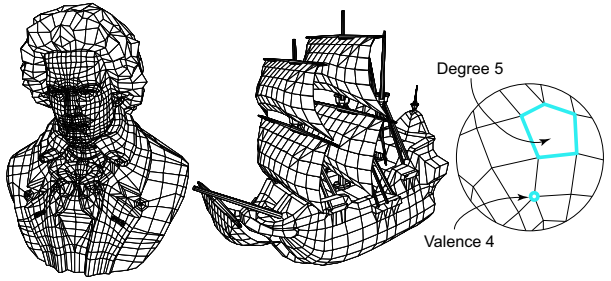

Fig. 1. Examples of polygon meshes: (left) Beethoven mesh (2812 polygons, 2655 vertices) - (right) Galleon mesh (2384 polygons, 2372 vertices). Close-up of a polygon mesh: the *valence* of a vertex is the number of edges incident to this vertex, while the *degree* of a face is the number of edges enclosing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

3 Single-rate Compression

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.

3.1 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 [52] gives a worst-case bound on the connectivity compression bit rate of 4 bits per vertex. Besides the popular *Edgebreaker* and its derivatives [39, 17, 59, 30], two techniques transform the connectivity of a triangle mesh into a sequence of valence codes [62, 28], 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 [62] or by a sequence of half edge collapses [28]. In [62], 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 [28] 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 [14].

One interesting variant on the Edgebreaker technique is the Cut-Border Machine (CBM) of Gumhold and Strasser [18]. The main difference is that the CBM encodes the split values as a parameter like the valence based schemes. This makes an upper bound on the resulting code more difficult to establish (although there is a bound of 5 b/v in [17]), but on the other hand allows for single pass coding and decoding. This difference is significant for coding massive data sets.

3.2 Non-triangle 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, e.g., the high-quality 3D models of the Viewpoint library [65] 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 [63, 34], 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. [7] 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 [48] proposed a so-called "dual" approach that traverses the edges of the dual mesh[3] 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* [29]. 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 [62, 52], both the encoder and decoder need a stack of boundary loops. Seven distinct labels F_n, R, L, S, E, H_n and $\mathrm{M}_{i,k,l}$ are used in order to describe the way to *fix* faces or holes together while traversing the current active gate. King et al. [40], Kronrod and Gotsman [42] and Lee et al. [46] 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 [62, 3] 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.

The Degree/Valence Approach

Since the Touma-Gotsman (TG) valence coder [62] 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. [35] and Isenburg [23]. The generalization relies on the key concept of *duality*. Consider an arbitrary 2-manifold triangle graph $\mathcal{M}$. Its dual graph $\tilde{\mathcal{M}}$, in which faces are represented as dual vertices and vertices become dual faces (see Fig. 2), should have the same connectivity information since dualization neither adds nor removes information. The valences of $\tilde{\mathcal{M}}$ are now all equal to 3, while the face degrees take on the same values as the vertex valences of $\mathcal{M}$. Since a

[3] See Fig. 2 for an illustration of a dual mesh.

list of all 3s has zero entropy, coding just the list of degrees of $\tilde{\mathcal{M}}$ would lead to the same bit rate as found for the valences of $\mathcal{M}$. 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.

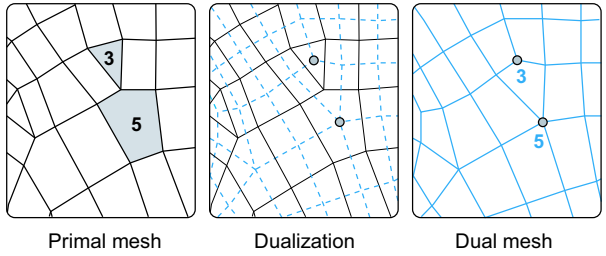

Fig. 2. Left: a polygon mesh with highlighted faces of degree 3 and 5. Middle: the dual mesh is built by placing one node in each original face and connecting them through each edge incident to two original faces. Right: the dual mesh now contains corresponding vertices of valence 3 and 5.

The core technique underlying the algorithm described in [23, 35] 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 [23, 35], 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 [62], 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 [23, 35] drive the traversal by various heuristics inspired from the valence-driven approach [3]. To better exploit correlation between streams and between symbols within each stream, it is possible to use a *context-based* arithmetic coder.

3.3 Connectivity: Entropy and Optimality

The entropy is a measure of the information content of a set of symbols, equipped with a probability distribution. It is thus the minimal average *number of bits per symbol* required for lossless encoding of a sequence of symbols from the set, each appearing with frequency given by its probability:

$$\text{entropy} = \sum_{i=1}^{N} p_i \ \log_2 \frac{1}{p_i}. \tag{1}$$

When the probability is not specified, this means that all symbols are equiprobable. When coding the connectivity of a mesh using entropy coding of its valences as introduced by Touma and Gotsman [62] for the case of triangular meshes, the bit-rates obtained are mostly dictated by the distribution of the valences. This automatically benefits from the *regularity* in the mesh. A triangle mesh is perfectly regular when the valence of all vertices is 6. The vertex valence distribution then has an entropy of zero. Later works, mainly generalizations of the Edgebreaker technique, were developed to explicitly take advantage of mesh regularity, and their performance has been shown to scale with this measure [39, 16, 59]. In [35], Khodakovsky et al. discuss the optimality of their valence/degree approach and show that the entropy of both the valence and degree sequences is no more than the entropy of the class of planar graphs as established by Tutte in the sixties [64]. Gotsman [13] later showed that the precise entropy of the valence and degree sequences is actually strictly less, but not by much, than the Tutte entropy, and the difference is made up by the split commands. Hence the number of split commands, albeit very small, is not negligible.

Entropy and constructive enumeration. Given a finite class of discrete elements, all equally probable, the entropy e of the class is the logarithm of the number of elements in the class. Obviously, the best possible performance of any algorithm coding this class of elements is to use at most e bits to encode one arbitrary element in the class. Hence, the issue of optimal coding of a class

is equivalent to the one of constructive enumeration [41]. Poulalhon and Schaeffer [51] have described a provably optimal coder for connectivity of meshes homeomorphic to a sphere, using a bijection between a triangulation and a Schnyder tree decomposition (i.e. a constructive enumeration of the connectivity graph). Although effective bounds are obtained, the code lengths do not adapt to the mesh regularity (every mesh consumes the same number of bits, whatever the distribution of valences). An objective of theoretical interest is to add flexibility to these methods in order to benefit from mesh regularity. Another obvious extension is to obtain similar results for high genus and non-triangular graphs.

3.4 Geometry Compression

Although the geometry data is often given in precise floating point representation for representing vertex positions, some applications may tolerate the reduction of this precision in order to achieve higher compression rates. The reduction of the precision involves *quantization*. The resulting values are then typically compressed by entropy coding after *prediction* relying on some data smoothness assumptions.

Quantization. The early works usually quantize uniformly the vertex positions for each coordinate separately in Cartesian space [10, 61, 62], and a more sophisticated vector quantization has also been proposed by Lee and Ko [45]. Karni and Gotsman [33] have also demonstrated the relevance of applying quantization in the space of spectral coefficients (see [14] for more details on this approach). In their elegant work, Sorkine et al. [57] address the issue of reducing the visual effect of quantization errors. Building on the fact that the human visual system is more sensitive to normal than to geometric distortion, they propose to apply quantization not in the coordinate space as usual, but rather in a transformed coordinate space obtained by applying a so-called "k-anchor invertible Laplacian transformation" over the original vertex coordinates. This concentrates the quantization error at the low-frequency end of the spectrum, thus preserving the fine normal variations over the surface, even after aggressive quantization (see Fig. 3). To avoid significant low-frequency errors, a set of anchor vertex positions are also selected to "nail down" the geometry at a select number of vertex locations.

Prediction. The early work employed simple delta coding [10] or linear prediction along a vertex ordering dictated by the coding of the connectivity [61, 62]. The approach proposed by Lee et al. [46] consists of quantizing *in the angle space* after prediction. By applying different levels of precision while quantizing the dihedral or the internal angles between or inside each facet, this method achieves better visual appearance by allocating more precision to the dihedral angles since they are more related to the geometry and normals. Inspired by the TG parallelogram prediction scheme [62], Isenburg and

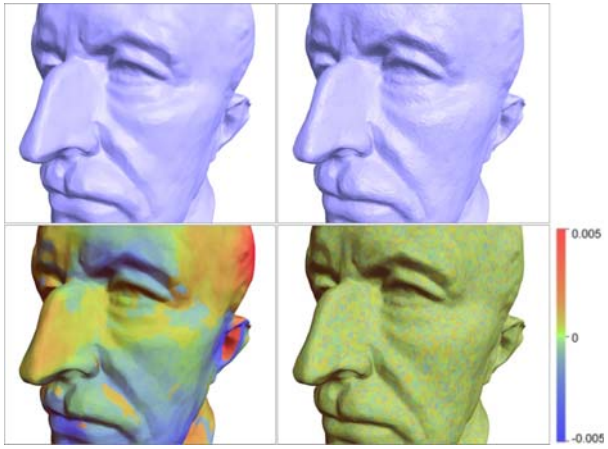

Fig. 3. The delta-coordinates quantization to 5 bits/coordinate (left) introduces low-frequency errors to the geometry, whereas Cartesian coordinates quantization to 11 bits/coordinate (right) introduces noticeable high-frequency errors. The upper rows shows the quantized model and the bottom figures use color to visualize corresponding quantization errors. Data courtesy O.Sorkine.

Alliez [25] complete the techniques described in [23, 35] by generalizing it to polygon mesh geometry compression. The polygon information dictates where to apply the parallelogram rule used to predict the vertex positions. Since polygons tend to be fairly planar and fairly convex, it is more appropriate to predict within polygons rather than across them. Intuitively, this idea avoids poor predictions resulting from a crease angle between polygons.

Despite the effectiveness of the published predictive geometry schemes, they are not optimal because the mesh traversal is dictated by the connectivity scheme. Since this traversal order is independent of the geometry, and prediction from one polygon to the next is performed along this, it cannot be expected to do the best job possible. A first approach to improve the prediction was the prediction trees [43], where the geometry drives the traversal instead of the connectivity as before. This is based on the solution of an optimization problem. In some case it results in an decrease of up to 50% in the geometry code entropy, in particular in meshes with significant creases and corners, such as CAD models. Cohen-Or et al. [9] suggest a multi-way predic-

tion technique, where each vertex position is predicted from all its neighboring vertices, as opposed to the one-way parallelogram prediction. An extreme approach to prediction is the feature discovery approach by Shikhare et al. [56], which removes the redundancy by detecting similar geometric patterns. However, this technique works well only for a certain class of models and involves expensive matching computations.

3.5 Optimality of Spectral Coding

Karni and Gotsman [33] showed that the eigenvectors of the Laplacian matrix derived from the mesh connectivity graph may be used to transform code the three Cartesian geometry n-vectors (x, y, z). The eigenvectors are ranked according to their respective eigenvalues, which are analogous to the notion of frequency in Fourier analysis. Smaller eigenvalues correspond to lower frequencies. Karni and Gotsman showed empirically that when projected on these basis vectors, the resulting projection coefficients decrease rapidly as the frequency increases. Hence, similarly to traditional transform coding, a good approximation to the geometry vectors may be obtained by using just a linear combination of a small number of basis vectors. The code for the geometry is then just this small number of coefficients (quantized appropriately). While this method seems to work quite well, and intuitively it seems that the Laplacian is a linear operator which captures well the smoothness of the mesh geometry relative to the mesh neighbor structure, there was no proof that this is the optimal basis for this purpose. The only indication that this might be the case is that in the case of a regular mesh, the eigenvectors of the Laplacian are the well-known 2D Fourier basis, which is known to be optimal for common classes of signals [20].

Ben-Chen and Gotsman [5] have imposed a very natural probability distribution on the class of meshes with a given connectivity, and then used principal component analysis (also known as the Karhunen-Loeve transform) to derive the optimal basis for that class. A series of statistical derivations then shows that this basis is identical to the eigenvectors of the symmetric Laplacian of the given connectivity (the sparse matrix whose diagonal is the vertex valence sequence, a negative unit entry for an edge, and zero otherwise). While this is a very interesting result, it remains theoretical, since computation of the Laplacian eigenvectors is still considered too expensive to be practical.

3.6 Coding Massive Data Sets

Due to their size and complexity, massive datasets [47] require dedicated algorithms since existing mesh compression are effective only if the representation of the mesh connectivity and geometry is small enough to fit "in-core". For large polygonal models that do not fit into main memory, Ho et al. [21] propose cutting meshes into smaller pieces that can be encoded in-core. They

process each piece separately, coding the connectivity using the Edgebreaker coder, and the vertex positions using the TG parallelogram linear predictor. Additional information required to stitch the pieces back together after decoding is also recorded, leading to bit-rates 25% higher than the in-core version of the same compression algorithm. A recent out-of-core technique introduced by Isenburg and Gumhold [27] makes several improvements upon [21] by (i) avoiding the need to *explicitly* break the model into several pieces, (ii) decoding the entire model in a single pass without any restarts, and (iii) streaming the entire mesh through main memory with a small memory foot-print. The core technique underlying this compression method consists of building a new external memory data structure – the *out-of-core mesh* – in several stages, all of them being restricted to clusters and active traversal fronts which fit in-core. The latter traversal order, consisting of a reordering of the mesh primitives, is computed in order to minimize the number of memory cache misses, similar in spirit to the notion of a "rendering sequence" [6] developed for improving performance of modern graphics cards, but at a much larger scale. The resulting compressed mesh format can stream very large meshes through the main memory by providing the compressor transparent access to a so-called *processing sequence* that represents a mesh as a fixed, yet seamless interleaved ordering of indexed triangles and vertices. At any point in time, the remaining part of the mesh data is kept on disk.

3.7 Remeshing for Single-rate Geometry Compression

The majority of mesh coders adapt to the regularity and the uniformity of the meshes (with the noticeable exception of [12] 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) piecewise 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.

Szymczak et al. [60] first split the mesh into relatively flat patches with smooth boundaries. Six axis-aligned vectors (so-called defining vectors) first determine some reference directions. From these vectors a partition of the mesh is built with a set of patches whose normals do not deviate more than a prescribed threshold. An approximation of the geodesic distance using Dijkstra's algorithm is then used in combination with a variant of the farthest point Voronoi diagram to smooth the patch boundaries. Each patch is then resampled by mutual tessellation over a regular hexagonal grid, and all the original vertices, but the boundary ones, are removed by half edge collapses (see Fig. 4). The connectivity of the resulting mesh is encoded using a version of Edgebreaker optimized for regular meshes, and vertex positions are compressed using differential coding and separation of tangential and normal

components.

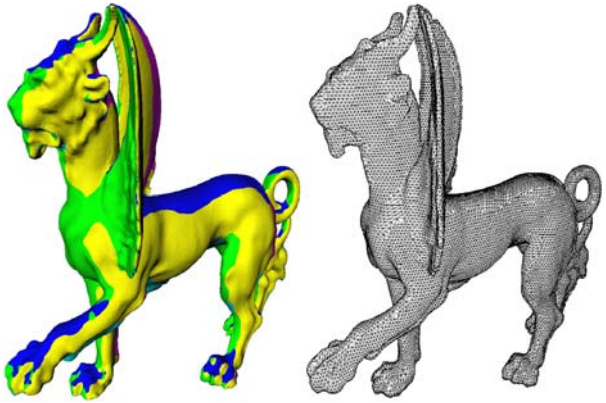

Fig. 4. Piecewise regular remeshing (data courtesy A.Szymczak).

Attene et al. [4] tile the input mesh using isosceles "triangloids". From each boundary edge of the tiling process, they compute a circle centered on the edge mid-point and lying on the bisecting plane between the two edge vertices. The location where the circle pierces the original mesh designates the tip vertex of the newly created triangloid tile. The original mesh is this way wrapped, the regions already discovered being identified as the triangles lying inside the regions bounded by geodesic paths between the three vertices of the new tile. Connectivity of the new mesh is compressed by Edgebreaker, while geometry is compressed by entropy coding one dihedral angle per vertex, after quantization.

Surazhsky and Gotsman [58] 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.

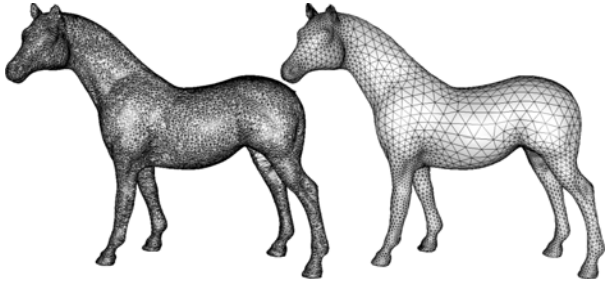

Fig. 5. Highly regular remeshing (data courtesy V.Surazhsky and C.Gotsman).

Gu et al. [15] 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 [53], 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 [15] or an atlas [54]. 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.

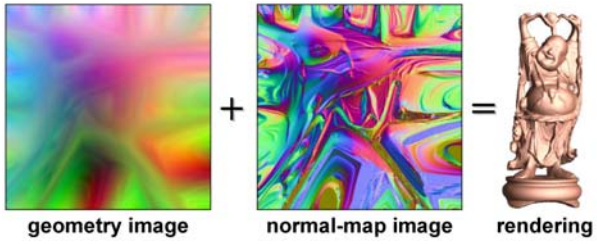

Fig. 6. Geometry image (data courtesy D.X.Gu).

3.8 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 [51] 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 [24], nothing has been proven concerning the optimality of this approach.

Most of the previous work has studied the coding of geometry as dictated by the connectivity code, the vertex positions being predicted in an order dictated by the connectivity coder. This happens even though the geometry component dominates the code sizes, so the result will tend to be suboptimal. One attempt to change this was to make the coding of the geometry cooperate with the coding of the connectivity, using prediction trees [43] or multi-way prediction techniques [9]. Other work [57] compresses the geometry globally, showing that applying quantization in the space of Laplacian transformed coefficients, instead of in the usual space of Cartesian coordinates, is very useful. In a way, the latter is an extension of the multi-way approach since it amounts to predicting each vertex as the barycenter of its neighbors. More recent work [5] aims to find an optimal basis best suited to decorrelate the geometric signal.

Isenburg et al. provide an on-line implementation of the degree/valence coder for bench marking purposes [26]. Isenburg also demonstrates an ASCII-based compression format for web applications able to achieve bit-rates within 1 to 2 percent of those of the binary benchmark code [31].

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 [15] 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.

4 Progressive Compression

Progressive compression of 3D meshes uses the notion of refinement: the original mesh is transformed into a sequence (or a hierarchy) of refinements applied to a simple, coarse mesh. During decoding the connectivity and the geometry are reconstructed incrementally from this stream. The main advantage of progressive compression is that it provides access to intermediate states of the object during its transmission through the network (see Fig. 7). The challenge then consists of rebuilding a least distorted object at all points in time during the transmission (i.e. optimization of rate-distortion tradeoff).

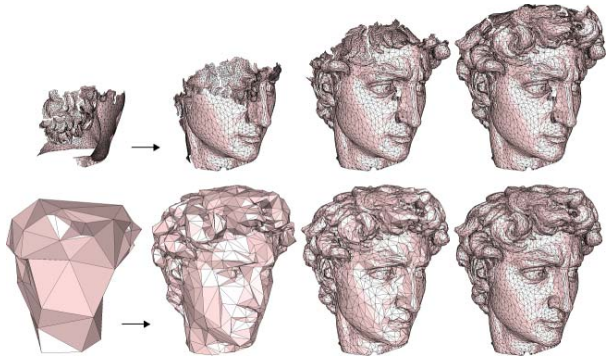

Fig. 7. Intermediate stages during the decoding of a mesh using a single-rate (top) or a progressive technique (bottom).

4.1 General Techniques

We call lossless the methods that restore the original mesh connectivity and geometry once the transmission is complete. This is even though intermediate stages are obviously lossy. These techniques mostly proceed by decimating the mesh while recording the (minimally redundant) information required to reverse this process. The basic ingredients behind most of progressive mesh compression techniques are (i) the choice of an atomic mesh decimation operator, (ii) the choice of a geometric distance metric to determine the elements to be decimated, and (iii) an efficient coding of the information required to reverse the decimation process (i.e. to refine the mesh). At the intuitive level, one has to encode for the decoder both the location of the refinement ("where"

to refine) and the parameters to perform the refinement itself ("how" to refine).

The progressive mesh technique introduced by Hoppe [22] transforms a triangle surface mesh into a stream of refinements. During encoding the input mesh undergoes a sequence of *edge collapses*, reversed during decoding as a sequence of *vertex splits*. The symbols generated provide the *explicit* location of each vertex being split and a designation of two edges incident to this vertex. This is a very flexible, but rather expensive code. In order to reduce the bit consumption due to the explicit vertex location, several researchers have proposed to specify these locations implicitly, using *independent sets* defined on the mesh. This approach improves the compression ratios, at the price of additional constraints during decimation (the decimation sequence cannot be arbitrary). Pajarola and Rossignac [49] group some edge collapses into a series of independent sets, each of them corresponding to a level of detail. The location of each vertex to decimate is done by a 2-coloring of the mesh vertices, leading to 1 bit per vertex, for each set. Experimental results show an amortized cost of 3 bits per vertex for vertex location for all sets, plus the cost of local refinements inverting the edge collapses, leading to 7.2 bits per vertex. Cohen-or et al. [8] define an alternation of 4- and 2-coloring over the triangles in order to locate an independent set of vertices to decimate. A local, deterministic retriangulation then fills the holes generated by vertex removal at no cost, leading to 6 bits per vertex.

Observing the local change of repartition of valences when removing a vertex, Alliez and Desbrun [2] improved the previous approaches by generating an alternation of independent sets composed of patches centered onto the vertices to be removed. Each independent set thus corresponds to one decimation pass. The even decimation passes remove valence ≤ 6 vertices, while the odd ones remove only valence 3 vertices. Such a selection of valences reduces the dispersion of valences during decimation, the latter dispersion being further reduced by a deterministic patch retriangulation designed to generate valence-3 vertices, later removed by odd decimation passes. This way the decimation is coordinated with the coding, and for "progressively regular" meshes the decimation generates a regular inverse $\sqrt{3}$-subdivision, and coding one valence per vertex is sufficient to rebuild the connectivity. For more general meshes some additional symbols are necessary. The latter approach can be seen as a progressive version of the TG coder [62].

Using the edge collapse as the atomic mesh decimation operator, Karni et al. [32] build a sequence of edge collapses arranged along a so called "vertex sequence" that traverses all the mesh vertices. The mesh traversal is optimized so that the number of jumps between two non-incident vertices is minimized. The decoding process is this way provided with an access to the mesh triangles optimized for efficient rendering using the modern vertex buffers. Compression

rates are similar to the progressive valence approach [2], with the additional benefit of fast rendering.

4.2 Geometry-driven Coding

For progressive compression of a discrete point set in arbitrary dimension, Devillers and Gandoin [11] decompose the space in a kD-tree and transmit only a set of point occurrences, i.e. the number of points located in each cell of the kD-tree hierarchy (see Fig. 8). They demonstrate that transmitting only occurrences of points during successive space subdivision is enough to reconstruct in a progressive manner – and lossless in the end – a discrete point set. The compression achieved by this is due to *bit sharing* intrinsic to the notion of transmission of occurrences, rather than transmission of explicit point locations. For example, transmitting the information "300 points" located in one cell at the beginning of the transmission process is equivalent to *sharing* the first high-level bits of 300 points, simultaneously. Some compression gain due to the sharing is thus obtainable for all cells containing more than one point. More precisely, the more populated the cell, the higher the compression gain due to the bit-sharing principle. When all points are *separated* – each cell containing only one point – the bits subsequently used are equivalent to bit-plane coding for progressively refining the point locations in space.

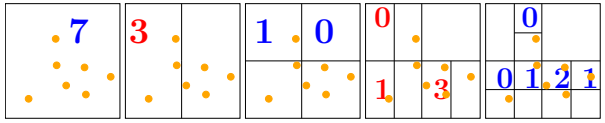

Fig. 8. The geometry coder on a two-dimensional example. The number of points located in each cell of the 2D-tree hierarchy is encoded (data courtesy P.-M.Gandoin).

During the decoding process, the only available information corresponds to the number of occurrences in each cell, i.e. a progressively refined location of the positions. At the end of the transmission, and if one does not care about any particular ordering of the original points, the original information has been restored in a lossless manner since every point is correctly located in a cell that corresponds to the initial precision over the points. Compared to a plain enumeration of the point coordinates, the information of the order over the points is lost. This is precisely what saves some bits for compression. It is proven that these savings are never less than $\log n - 2.402$ bits per point. Moreover, in this case – and contrary to other prediction-based methods that benefit from a uniform sampling – the uniform distribution corresponds to the worst-case scenario for compression since it minimizes the possibility of

bit-sharing.

The approach described in [11] codes only a set of discrete points. The authors have shown how a geometric triangulation (e.g., Delaunay) of the points allow to progressively transmit a meshed surface. More recently, Devillers and Gandoin [12] have adapted this technique for progressive coding of simplicial complexes (possibly non-manifold) by using the edge collapse operator for coding the connectivity. Contrary to other methods, the connectivity coding process is driven by the geometry alone.

4.3 Remeshing for Progressive Geometry Compression

When the original mesh is considered as one instance of the surface geometry that is to be compressed, *geometry compression* has to be considered rather than mesh compression. To this end, geometry compression techniques proceeding by *semi-regular remeshing* are among the best reported to date.

The main idea behind semi-regular remeshing techniques [38, 19, 36, 50] 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. In the case of *regular* subdivision by edge bisection of a triangle base mesh, all new vertices have valence 6. The finest level of the mesh hierarchy is therefore built from patches of regular vertices, separated by (possibly irregular) vertices from the base mesh. In the case of *adapted* subdivision, some irregular vertices may be generated by adding a few conforming edges (note that this choice can be decided on the decoder side, depending if it cares about reconstructing the adaptivity pattern or not). We now describe how the connectivity and the parametric components are reduced:

- *Reducing the connectivity component.* The regularity intrinsic to the subdivision process deliberately removes almost all of the connectivity information from the mesh since much of the resulting vertices have valence 6 for triangle meshes, and valence 4 for quad meshes.
- *Reducing the parametric component.* In a semi-regular mesh hierarchy generated by subdivision for purpose of geometric approximation, we use the

term *detail* to describe the differential vector component stored at each newly inserted vertex during the construction of the mesh hierarchy. The normal component of the detail coefficients stores the geometric information, whereas the tangential component carries the parametric information. Experiments show that by doing things right almost all of the parametric components can be "predicted", i.e. removed from the representation. Intuitively, this means that sliding a vertex in the tangent plane does not modify the local geometry.

The way the parametric component is reduced is the main distinction between the two compression methods described in this section. The first method [38] uses local frames and different quantization of normal/tangential components, whereas the second *normal mesh* compression method [36] is specifically designed to produce detail coefficients with no tangential components. Normal meshes were introduced by Guskov et al. [19] as a new way to represent geometry. A normal mesh is a multiresolution representation where almost all the details lie in the local normal direction and hence the mesh geometry is described by a single scalar per vertex instead of three as usual (see Fig. 9). Beyond the remeshing algorithm, both progressive geometry coding methods proposed by Khodakovsky et al. [38, 36] require a wavelet transform and a zerotree coder that we now briefly describe.

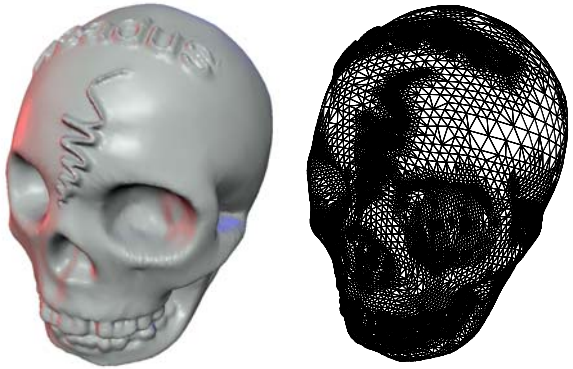

Fig. 9. Adaptive normal mesh for the skull model (data courtesy A.Khodakovsky).

Wavelet transform. A semi-regular surface representation is a sequence of approximations at different levels of resolution. The corresponding sequence of nested refinements is transformed by the *wavelet transform* into a representation that consists of a base mesh and a sequence of wavelet coefficients that express differences between successive levels of the mesh hierarchy. The art of compression then consists of choosing appropriate wavelets to best model the statistical distribution of the wavelet coefficients. "Best" means decorrelating the geometry so as to obtain a distribution of wavelet coefficients favorable to efficient compression. For subdivision schemes designed to produce C2-differentiable surfaces almost everywhere (e.g., Loop), it produces excellent results for smooth surfaces since the geometric correlation can be exploited through the prediction of finer level geometry based on the coarser level geometry (by low-pass filtering intrinsic to the subdivision scheme). The reconstruction artifacts at low bit-rates depend mainly on the shape of subdivision basis functions. Hence a surface reconstructed from Loop wavelets has a visually more pleasing shape compared to Butterfly whose basis functions exhibit some "spikes" which look unnatural on a smooth surface.

The zerotree coder (a popular method for wavelet-based image coding [55]) extends also to geometric wavelets. It is shown in [38] that a semi-regular mesh can be represented as a hierarchy over the edges since the wavelet coefficients are attached to the edges (and not the faces). The wavelet coefficients are therefore naturally encoded in a forest of trees, where each wavelet coefficient at the first level is the root of a tree. The branches of this tree may have variable depth since all regions need not be subdivided to the finest level. For the latter case the edges of the base mesh are subdivided in the tree until an approximation quality is met during coding, and until an adaptive flatness threshold is met during decoding (the subdivision scheme is prolongated to produce a smooth surface even with null wavelet coefficients). Note that an approximating wavelet scheme such as Loop requires uniform mesh subdivision. A zerotree coder is therefore used as a separate procedure to reflect this adaptivity, a "zerotree" symbol coding at a given branching node in the tree representing a sub-tree filled entirely with coefficients below the significance threshold. It then remains to compress the zerotree symbols (significance bits, sign bits and refinement bits), which is done using an arithmetic coder.

When using lossless compression, performance is measured by plotting the rate-distortion curve. Measuring bits per vertex here would be irrelevant since the initial mesh is not considered as optimal for approximation, and the remeshing stage changes the number of vertices. The main observation of [38] is that in many cases surface representation does not really matter for geometry (i.e. there are many degrees of freedom in the meshing), but can lead to high penalty for compression. Therefore, several degrees of sophistication in the remeshing process can lead to significant gains for compression:

- semi-regular remeshing reduces the connectivity penalty.
- uniform remeshing reduces the parameterization penalty compared to non-uniform meshes.
- uniform remeshing while considering local frames with different quantization for different components reduces the influence of the parameterization even further.
- normal remeshing, explicitly eliminating the tangential component by building a normal mesh representation makes a significant improvement [36] for certain classes of wavelets (e.g., normal Butterfly is better than normal Loop).

Another observation of [36] is that normal Loop behaves better than [38] because the normal parameterization is smoother than MAPS [44], which leads to faster decaying wavelet coefficients and therefore more efficient compression. Moreover, recent experiments confirm the importance of the smoothness of the parameterization for semi-regular remeshing and hence for geometry compression [37]. Finally, a model-based bit-allocation technique has been proposed by Payan and Antonini [50] to efficiently allocate the bits across wavelet subbands according to their variance.

4.4 Comparison and Discussion

Most of the recent techniques for "lossless" progressive coding of carefully designed meshes use the *independent set* concept to drive the mesh refinement operations, be they organized into a set of patches or along a chain of edges optimized for efficient rendering. Vertex positions are coded using various prediction schemes (barycentric, etc.) after uniform quantization applied in vertex coordinate space. As already observed in Section 3, less work has been done for geometry coding than for connectivity coding. There is even less work on progressive coding techniques, since they are obviously lossy (at least at intermediate stages), and the difficulty to objectively quantify the loss makes it difficult to analyze their performance.

Although the successful single-rate valence-based approach generalizes to progressive coding of triangle meshes [2], nothing has been done for progressive coding of polygonal meshes. The key problem here is to find a mesh decimation scheme capable of transforming an arbitrary mesh into a polygon mesh during decimation so that the so-called rate-distortion tradeoff is optimized. At the intuitive level, each bit transmitted during decoding should lead to the largest decrease in geometric error. Although this problem is similar to that encountered when designing a decimation scheme, the amount of information represented by a given refinement operation has yet to be quantified.

Wavelet coding schemes, coupled with an initial semi-regular remeshing stage to generate a good base mesh, have proven to be very successful for

shape compression [38]. One important question there is how to generate the "best" base mesh, in the sense that it best reflects the mesh geometry and, mainly, features. Another question is the choice of a wavelet scheme suited to best decorrelate the geometric "signal" present in the mesh. Is it a subdivision-related scheme or something more elaborate? The choice of an error metric for driving both the approximation and the measurement of the rate-distortion tradeoff also plays a key role. The latter point has already proven to be of crucial importance for applications related to visualization (see the concept of visual metric in [33, 57]). In fact, the optimization of the rate-distortion tradeoff involves many challenging issues linked to sampling and approximation theory, differential geometry, wavelets and information theory.

References

1. P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun. Anisotropic Polygonal Remeshing. In *ACM SIGGRAPH Conference Proceedings*, 2003.
2. P. Alliez and M. Desbrun. Progressive Encoding for Lossless Transmission of 3D Meshes. In *ACM SIGGRAPH Conference Proceedings*, pages 198–205, 2001.
3. P. Alliez and M. Desbrun. Valence-Driven Connectivity Encoding of 3D Meshes. In *Eurographics Conference Proceedings*, pages 480–489, 2001.
4. M. Attene, B. Falcidieno, M. Spagnuolo, and J. Rossignac. SwingWrapper: Retiling Triangle Meshes for Better EdgeBreaker Compression. *ACM Transactions on Graphics*, 2003. to appear.
5. M. Ben-Chen and C. Gotsman. On the Optimality of the Laplacian Spectral Basis for Mesh Geometry Coding, 2003. preprint.
6. A. Bogomjakov and C. Gotsman. Universal Rendering Sequences for Transparent Vertex Caching of Progressive Meshes. *Computer Graphics Forum*, 21(2):137–148, 2002.
7. 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.
8. D. Cohen-Or, D. Levin, and O. Remez. Progressive Compression of Arbitrary Triangular Meshes. In *IEEE Visualization Conference Proceedings*, pages 67–72, 1999.
9. R. Cohen D. Cohen-Or and T. Ironi. Multi-way Geometry Encoding, 2002. Technical report.
10. M. Deering. Geometry Compression. *ACM SIGGRAPH Conference Proceedings*, pages 13–20, 1995.
11. O. Devillers and P-M. Gandoin. Geometric Compression for Interactive Transmission. *IEEE Visualization Conference Proceedings*, pages 319–326, 2000.
12. 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.
13. C. Gotsman. On the Optimality of Valence-Based Connectivity Coding. *Computer Graphics Forum*, 22(1):99–102, 2003.

14. 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.
15. X. Gu, S. Gortler, and H. Hoppe. Geometry Images. In *ACM SIGGRAPH Conference Proceedings*, pages 355–361, 2002.
16. S. Gumhold. Improved Cut-Border Machine for Triangle Mesh Compression. *Erlangen Workshop'99 on Vision, Modeling and Visualization*, 1999.
17. S. Gumhold. New Bounds on the Encoding of Planar Triangulations. Technical Report WSI-2000-1, Univ. of Tübingen, 2000.
18. S. Gumhold and W. Strasser. Real Time Compression of Triangle Mesh Connectivity. In *SIGGRAPH'98 Conference Proceedings*, pages 133–140, 1998.
19. I. Guskov, K. Vidimce, W. Sweldens, and P. Schröder. Normal Meshes. In *Proceedings of SIGGRAPH*, pages 95–102, 2000.
20. Henry Helson. *Harmonic Analysis*. Wadsworth & Brooks/Cole, 1991.
21. J. Ho, K-C. Lee, and D. Kriegman. Compressing Large Polygonal Models. In *IEEE Visualization Conference Proceedings*, pages 357–362, 2001.
22. H. Hoppe. Progressive meshes. In *ACM SIGGRAPH Conference Proceedings*, pages 99–108, 1996.
23. M. Isenburg. Compressing Polygon Mesh Connectivity with Degree Duality Prediction. In *Graphics Interface Conference Proc.*, pages 161–170, 2002.
24. M. Isenburg and P. Alliez. Compressing Hexahedral Volume Meshes. In *Pacific Graphics Conference Proceedings*, pages 284–293, 2002.
25. M. Isenburg and P. Alliez. Compressing Polygon Mesh Geometry with Parallelogram Prediction. In *IEEE Visualization Conference Proceedings*, pages 141–146, 2002.
26. M. Isenburg, P. Alliez, and J. Snoeyink. A Benchmark Coder for Polygon Mesh Compression, 2002. http://www.cs.unc.edu/∼isenburg/pmc/.
27. M. Isenburg and S. Gumhold. Out-of-Core Compression for Gigantic Polygon Meshes. In *SIGGRAPH conference proceedings*, 2003. To appear.
28. M. Isenburg and J. Snoeyink. Mesh Collapse Compression. In *Proceedings of SIBGRAPI'99, Campinas, Brazil*, pages 27–28, 1999.
29. M. Isenburg and J. Snoeyink. Face Fixer: Compressing Polygon Meshes With Properties. In *ACM SIGGRAPH Conference Proceedings*, pages 263–270, 2000.
30. 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.
31. M. Isenburg and J. Snoeyink. Binary Compression Rates for ASCII Formats. In *Proceedings of Web3D Symposium*, pages 173–178, 2003.
32. Z. Karni, A. Bogomjakov, and C. Gotsman. Efficient Compression and Rendering of Multi-Resolution Meshes. In *IEEE Visualization Conference Proceedings*, 2002.
33. Z. Karni and C. Gotsman. Spectral Compression of Mesh Geometry. In *ACM SIGGRAPH Conference Proceedings*, pages 279–286, 2000.
34. Keeler and Westbrook. Short Encodings of Planar Graphs and Maps. *Discrete Appl. Math.*, 58:239–252, 1995.
35. A. Khodakovsky, P. Alliez, M. Desbrun, and P. Schröder. Near-Optimal Connectivity Encoding of 2-Manifold Polygon Meshes. *Graphical Models, special issue*, 2002.

36. A. Khodakovsky and I. Guskov. Compression of Normal Meshes. In *Geometric Modeling for Scientific Visualization*. Springer-Verlag, 2003.
37. A. Khodakovsky, N. Litke, and P. Schröder. Globally Smooth Parameterizations with Low Distortion, 2003. to appear at SIGGRAPH.
38. A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive Geometry Compression. *ACM SIGGRAPH Conference Proceedings*, pages 271–278, 2000.
39. 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.
40. D. King, J. Rossignac, and A. Szmczak. Compression for Irregular Quadrilateral Meshes. Technical Report TR–99–36, GVU, Georgia Tech, 1999.
41. D. Knuth. Exhaustive Generation, 2003. volume 4 of The Art of Computer Programming, in preparation, available electronically, `http://www.cs.stanford.edu/~knuth`.
42. B. Kronrod and C. Gotsman. Efficient Coding of Non-Triangular Mesh Connectivity. *Graphical Models*, 63(263-275), 2001.
43. B. Kronrod and C. Gotsman. Optimized Compression of Triangle Mesh Geometry Using Prediction Trees. *Proceedings of 1st International Symposium on 3D Data Processing, Visualization and Transmission*, pages 602–608, 2002.
44. A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution adaptive parameterization of surfaces. *Computer Graphics*, 32(Annual Conference Series):95–104, August 1998.
45. E. Lee and H. Ko. Vertex Data Compression For Triangular Meshes. In *Proceedings of Pacific Graphics*, pages 225–234, 2000.
46. H. Lee, P. Alliez, and M. Desbrun. Angle-Analyzer: A Triangle-Quad Mesh Codec. In *Eurographics Conference Proceedings*, pages 383–392, 2002.
47. M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The Digital Michelangelo Project. In *ACM SIGGRAPH Conference Proceedings*, pages 131–144, 2000.
48. J. Li and C.-C. Jay Kuo. Mesh Connectivity Coding by the Dual Graph Approach, July 1998. MPEG98 Contribution Document No. M3530, Dublin, Ireland.
49. R. Pajarola and J. Rossignac. Compressed Progressive Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):79–93, 2000.
50. 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.
51. D. Poulalhon and G. Schaeffer. Optimal Coding and Sampling of Triangulations, 2003. 30th international colloquium on automata, languages and programming (ICALP'03).
52. J. Rossignac. Edgebreaker : Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics*, 1999.
53. P. Sander, S. Gortler, J. Snyder, and H. Hoppe. Signal-Specialized Parametrization. In *Eurographics Workshop on Rendering 2002*, 2002.
54. P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe. Multi-Chart Geometry Images. In *Proceedings of Eurographics Symposium on Geometry Processing*, 2003.
55. J.M. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, 1993.

56. D. Shikhare, S. Bhakar, and S.P. Mudur. Compression of Large 3D Engineering Models using Automatic Discovery of Repeating Geometric Features. In *proceedings of 6th International Fall Workshop on Vision, Modeling and Visualization*, 2001.
57. O. Sorkine, D. Cohen-Or, and S. Toldeo. High-Pass Quantization for Mesh Encoding. In *Proceedings of Eurographics Symposium on Geometry Processing*, 2003.
58. V. Surazhsky and C. Gotsman. Explicit Surface Remeshing. In *Proceedings of Eurographics Symposium on Geometry Processing*, 2003. To appear.
59. A. Szymczak, D. King, and J. Rossignac. An Edgebreaker-Based Efficient Compression Scheme for Regular Meshes. *Computational Geometry*, 20(1-2):53–68, 2001.
60. A. Szymczak, J. Rossignac, and D. King. Piecewise Regular Meshes: Construction and Compression. *Graphical Models*, 2003. to appear.
61. G. Taubin, W. Horn, J. Rossignac, and F. Lazarus. Geometry Coding and VRML. In *Proceedings of the IEEE*, volume 86(6), pages 1228–1243, june 1998.
62. C. Touma and C. Gotsman. Triangle Mesh Compression. *Graphics Interface 98 Conference Proceedings*, pages 26–34, 1998.
63. G. Turan. Succinct Representations of Graphs. *Discrete Applied Mathematics*, 8:289–294, 1984.
64. W. Tutte. A Census of Planar Maps. *Canadian Journal of Mathematics*, 15:249–271, 1963.
65. Viewpoint. *Premier Catalog (2000 Edition) www.viewpoint.com*. Viewpoint editor, 2000.