3D mesh compression: survey, comparisons and emerging trends

ADRIEN MAGLO, Visualink
GUILLAUME LAVOUÉ, Université de Lyon, CNRS, LIRIS, INSA-Lyon
FLORENT DUPONT, Université de Lyon, CNRS, LIRIS, Université Lyon 1
CÉLINE HUDELOT, MAS Laboratory, École Centrale Paris

3D meshes are commonly used to represent virtual surface and volumes. However, their raw data representations take a large amount of space. Hence, 3D mesh compression has been an active research topic since the mid 90's. In 2005, two very good review articles describing the pioneering works were published. Yet, new technologies have emerged since then. In this article, we summarize the early works and put the focus on these novel approaches. We classify and describe the algorithms, evaluate their performance and provide synthetic comparisons. We also outline the emerging trends for future researches.

Categories and Subject Descriptors: E.4 [**Data**]: Coding And Information Theory

General Terms: Algorithms, Performance, Theory, Experimentation.

Additional Key Words and Phrases: 3D mesh, compression, single-rate, progressive, random accessible, dynamic.

ACM Reference Format:
Adrien Maglo, Guillaume Lavoué, Florent Dupont, Céline Hudelot, 2013. 3D mesh compression: survey, comparisons and emerging trends. *ACM Comput. Surv.* 9, 4, Article 39 (September 2013), 40 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

3D meshes may be considered to be the most popular discrete virtual surface and volume representation. Its simplicity makes it so popular today that electronic ships, called GPUs (Graphical Processing Units), partially specialized in the rendering of images from 3D meshes, are integrated in nearly all personal computers, tablets and smart phones.

In all application areas using meshes (computational simulation, entertainment, medical imaging, digital heritage, computer-aided design, e-commerce, etc.), the need for precision has never ceased to increase. This leads to the generation of meshes composed of a large number of elements, whose processing, visualization and storage are complex. 3D mesh compression is a potential tool for solving the issues raised. First, it reduces data size, which is useful for storing and transmitting meshes. Then, some algorithms also embed several versions of the input model in the compressed data. This enables progressive transmission and interactive visualization of large meshes on low capability terminals.

In 2005, two very complete reviews, summarizing the pioneering work on mesh compression, were published [Alliez and Gotsman 2005][Peng et al. 2005]. However, since then, new methods have emerged, bringing new features to 3D mesh compression al-

This work was sponsored by ...

DOI: http://dx.doi.org/10.1145/0000000.0000000

gorithms (large mesh compression, mesh sequence compression, random access, etc.). The aim of this new review paper is to sum up the first approaches and focus on the relevant contributions developed since 2005.

Meshes can be either *static* or *dynamic*. Following an introduction of preliminary knowledge, the first part of this paper is dedicated to static mesh compression approaches classified in three types.

Single-rate algorithms build a compact representation of an input mesh. The decompression algorithm generates a mesh that is either identical to the input model or only slightly different.

Progressive algorithms, during decompression, reconstruct successive levels of detail as more data are decoded. The user does not have to wait for all the data to be downloaded and decompressed to visualize the model, which is useful in a remote visualization context. These algorithms are also used to select the best level of detail to display according to device rendering capabilities, network constraints or the viewpoint.

Random accessible algorithms are used to decompress only requested parts of the input mesh to save resources. They provide access to models that do not fit into the device main memory, but the user has no overview of the non-selected parts. *Progressive random accessible* approaches are used to decompress different parts of the input model at different levels of detail.

Progressive and random accessible algorithms can also be used for single-rate compression, although they are not as efficient as pure single-rate methods in general.

The last part of the article is dedicated to *dynamic mesh compression*. Dynamic meshes constitute an emerging media content, which may carry large amounts of data (large meshes $\times$ large frame numbers). Like 2D videos, dynamic mesh compression requires specific features such as scalability and streaming capability.

Figure 1 provides a taxonomy of existing techniques; for each part described above, a synthetic table summarizes and compares the most relevant approaches.

2. PRELIMINARIES ON MESHES AND COMPRESSION

A mesh can be defined as the hierarchical assembly of the different elements represented on Figure 2. *Vertices* are the base elements of the mesh. They define a position in a common 3D Cartesian space. *Edges* are segments that connect two vertices of the mesh. *Faces* are polygons defined by a closed path of edges. In this survey, we will assume that each face is simply connected: it does not have holes and its boundaries do not have vertices with more than 2 edges of that face incident on them. 3D cells named *3-cells* are polyhedron defined by their boundaries, which are closed surfaces formed by selected faces. The information contained in a mesh is often divided into three categories:

- The **geometry** information is the position of each vertex of the mesh in the 3D Cartesian space.
- The **connectivity** information (sometimes called topology or structure) describes the incidence relations between the mesh elements.
- The optional **attribute** information associates scalar or discrete properties to the mesh elements useful for the applications (colors, normals, texture coordinates...).

This review only deals with the compression of the geometry and connectivity information. Most compression rates, for both information types, will be given in bits per vertex (bpv).

Surface meshes define surfaces that can be closed or not. They do not need to be boundaries of 3-cells. A surface mesh is said to contain boundaries if some of its edges are adjacent to only one face. *Volume meshes* define a volume filled by a set of 3-cells.

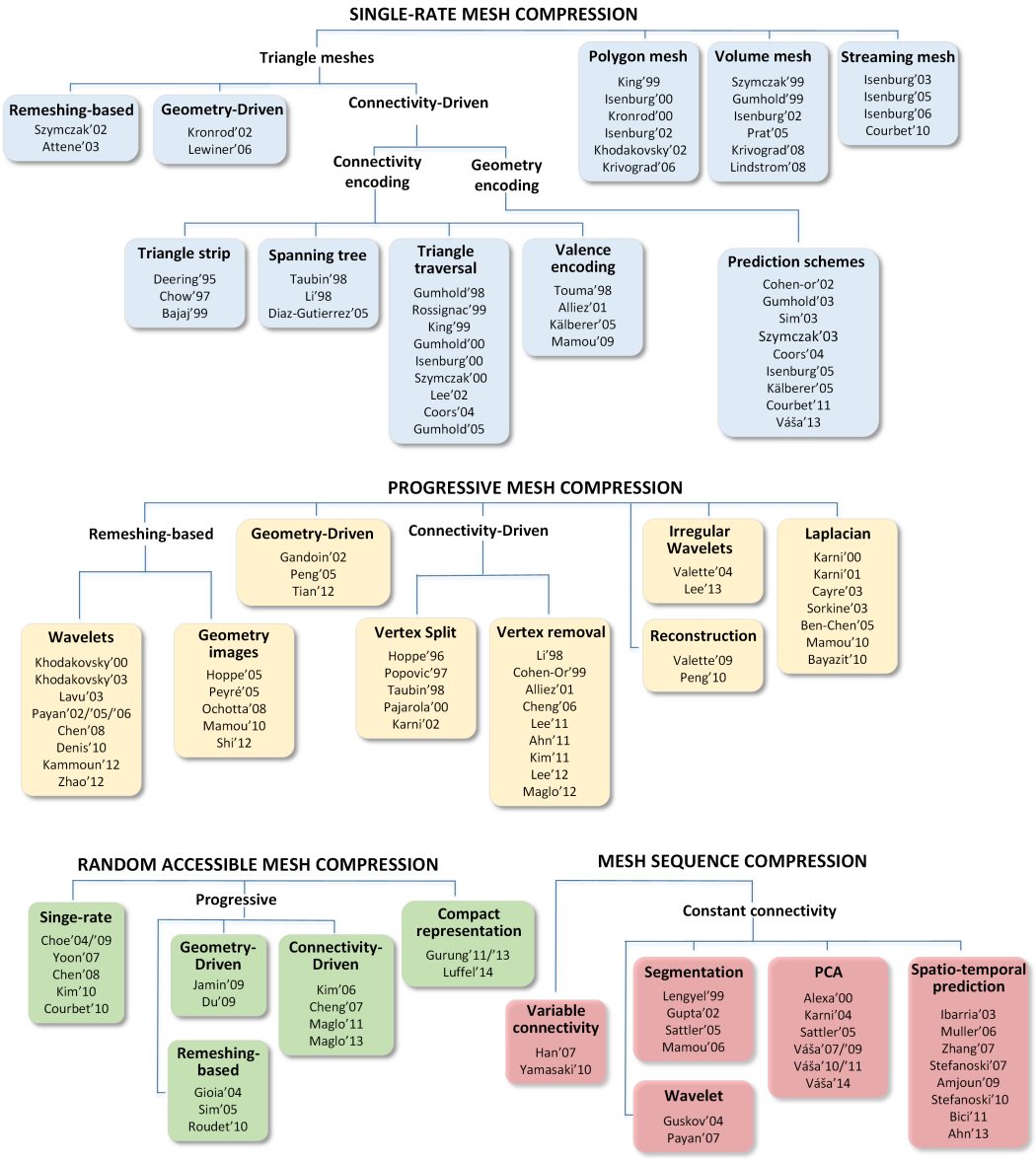

Fig. 1. Taxonomy of 3D mesh compression techniques.

A *mesh sequence* is a series of meshes ordered in time. Temporal coherency is required and necessitates temporal sampling at a rate which is sufficiently rapid in relation to the evolution of the object or the scene. We can distinguish two different types of sequences:

- The case of mesh sequences with a constant number of vertices, connectivity and topology, which we may call *temporally coherent mesh sequences* [Arcila et al. 2012], are usually referred as *dynamic meshes* or *animated meshes*.

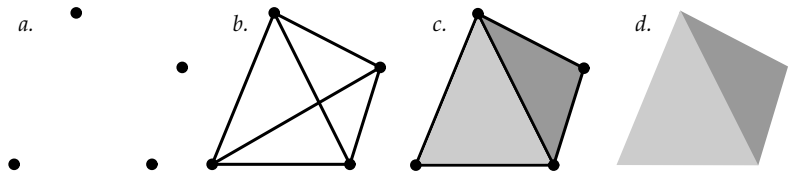

Fig. 2. Elements of a mesh. *a.* Vertices. *b.* Edges. *c.* Faces. *d.* A 3-cell.

- The case of mesh sequences with a variable number of vertices and variable connectivity over time, with a topology that may be variable or constant have been defined as *temporally incoherent mesh sequences* [Arcila et al. 2012].

In the first case, the sequence consists of a geometric evolution of the vertices of the initial mesh over time. In the second case, there is not necessarily natural link between a vertex at instant *t* and a vertex at instant *t+1*. An example of a sequence is shown on Figure 3.

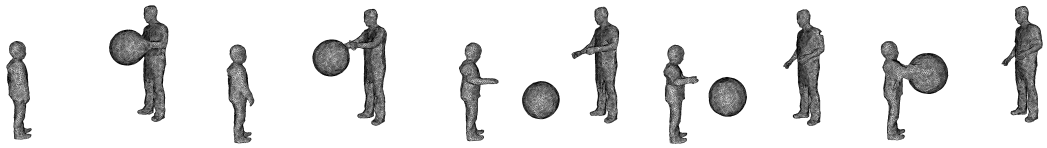

Fig. 3. Balloon mesh sequence (http://4drepository.inrialpes.fr)

The number of edges incident to a vertex is called the *valence* (or *degree*) of this vertex. The number of edges of a face or the number of faces of a 3-cell are also called valence (or degree) of this face or this 3-cell. The faces can have a degree superior or equal to three. The common cases for surface meshes are degree three (triangle face) and degree four (quadrangle or quad face). If its degree is superior to three, the face is not necessarily planar. However it may be tricky to render if some of its vertices are far from a median plan. In the same way, the common cases for volume meshes are 3-cells with a degree equal to four (tetrahedron) or six (hexahedron).

Some mesh connectivities are said to be *regular* because they are composed of the repetition of the same pattern. If only few elements of the mesh do not have a regular connectivity, then the mesh is said to be *semi-regular*. When the connectivity of the mesh does not contain regular structures, it is said to be *irregular*.

A surface mesh is *2-manifold* if all its vertices have a neighborhood homomorphic to a closed or open fan, as shown on Figure 4. The neighborhood of a vertex is homomorphic to an open fan if its contains boundaries. This review will discuss the compression of manifold and non-manifold meshes.

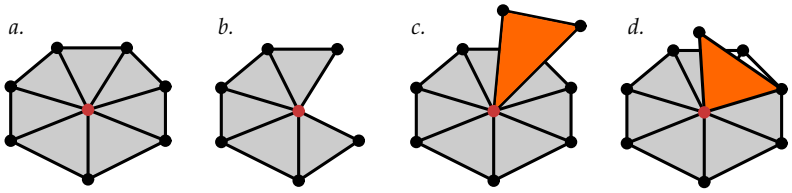

Fig. 4. Manifold and non-manifold vertices. *a.* The red vertex is manifold because its neighborhood is equivalent to a closed fan. *b.* The red vertex is manifold because its neighborhood is equivalent to an open fan. *c.* & *d.* The red vertex is non-manifold because its neighborhood is not equivalent to an open or closed fan.

The orientation of a face is a the cyclic order of its vertices. In a mesh, the orientations of two adjacent faces are *compatible* if they are opposite. A mesh is said to be *orientable* if all the orientations of its faces are compatible.

The Euler characteristic χ of a 2-manifold orientable mesh defines its topological type. It is equal to: $\chi = v - e + f$, where v, e and f are respectively the number of vertices, edges and faces of the mesh. A mesh is said to have genus g if it can be cut along $2g$ closed loops without disconnecting it. Intuitively, the genus is the number of handles of the mesh. It can be proven that the Euler characteristic is also equal to: $\chi = 2(s-g)-b$, where s is the number of connected components of the mesh and b is its number of boundary loops. This allows to prove that a 2-manifold orientable triangle mesh with a low genus and a significant number of faces has about twice more faces than vertices and that the average valence of its vertices is six.

Many mesh file formats (OFF, PLY, OBJ, VRML, X3D...) are based on an indexed data structure. The first part of this structure is composed of the list of all the vertex coordinates that also introduces an ordering of the mesh vertices. The second part describes the connectivity of the mesh. Each entry lists all the vertices of one face in a cyclic order. For volume meshes, the faces of each 3-cell can also be stored in the same way. This data structure has the advantage of being simple. Moreover, it can store any type of meshes. Most mesh compression algorithms take this structure as input.

3. STATE OF THE ART ON SINGLE-RATE MESH COMPRESSION

Compressing meshes is different than compressing other types of multimedia data such as sound, images or videos. The common point between sound, images and videos is that their structure is known in advance by the encoder and the decoder. A mesh is, however, not necessarily regularly structured. Its connectivity is completely unknown to the encoder before the compression. So, besides having to code the geometry (vertex positions), as the pixel colors would be coded for an image, a mesh encoder must encode the structure, which is the connectivity.

3.1. Connectivity compression

The general principle behind all connectivity compression techniques is to perform a traversal of the mesh elements and emit symbols, among few possibilities, depending on the encountered configurations. The main point is that this traversal defines a new numbering of the mesh elements, different from the one used in the input indexed data structure. The generated symbols are then entropy-coded using, for example, a Huffman coder [1952] or an arithmetic coder [Rissanen and Langdon 1979]. The compression rates provided in this subsection concerns only the connectivity.

3.1.1. Triangle strip encoding. The rendering of a significant 3D mesh is a task that requires high computational capabilities. The GPUs are designed to perform the mesh rendering operations quickly in parallel. Efficient solutions to transfer the mesh data from the computer central memory to the GPU memory are of prime interest because memory transfer is a costly operation that consumes a lot of CPU cycles. A good method to transfer mesh data can significantly decrease the rendering time. *Triangle strips* and *triangle fans* are mesh representations that are used for this purpose. A triangle fan is a sequence of vertices that defines a set of triangles sharing a common center vertex. A triangle strip is a sequence of vertices where each added vertex defines a new triangle with the two previous vertices of the strip. These methods are much more efficient than the indexed representation that requires three vertex indices to code each triangle. Indeed, after having encoded the first triangle, a new vertex index codes the connectivity of a new triangle.

In a triangle strip, to form a new triangle, the vertices are always taken in the same order. *Generalized triangle strips* do not always respect this order so as to generate longer strips. However, to preserve the triangle strip structure, a dummy triangle is added when the standard order is not preserved. To encode a dummy triangle, a vertex reference must be repeated. If the generalized triangle strip is long enough, the ratio between the number of triangles and the number of vertices is close to 1.

One of the first work in mesh compression is the generalized triangle mesh format of Deering [1995], which includes generalized triangle strips. Deering noted that, in a generalized triangle strip, many of the interior vertices are encoded twice. So, instead of encoding twice their positions traversing the mesh, he proposed to push them in a 16 positions queue and refer to them by their location in the queue when needed. This queue reduces the cost of reused vertices encoding. Yet, all the mesh vertices are still encoded twice on average, which is not very efficient. Experimental results report that Deering's algorithm achieves connectivity compression rates from 3.3 to 9.8 bpv.

Generalized triangle meshes must maximize the number of vertices reused from the queue. The strip sizes and the strip orders obtained with most mesh stripification techniques are not suited to this aim. Therefore, Chow [1997] designed algorithms that generate efficient generalized triangle meshes. It selects the new generalized triangle strip to encode in function of the number of reused vertices of the previous strip.

Bajaj et al. [1999] proposed an alternative layered representation. Vertex layers are non-crossing strings of vertices. In most cases, they are separated by a distance of one edge and form concentric circles on the mesh. The triangle layers between the vertex layers contain triangle strips and fans. This method compresses the mesh connectivity at about 1.5 to 6 bpv and can process non-manifold meshes.

3.1.2. Spanning tree encoding of planar graphs. The connectivity of a mesh can be modeled as a graph. For surface meshes, the vertices are nodes of the graph linked through edges to form faces. This analogy between meshes and graphs explains why some well-known results from the graph theory can be applied to the compression of mesh connectivity. Tutte first proposed formula that enumerate planar triangulations [Tutte 1962] and planar maps [Tutte 1963]. The first enumeration allows to compute what was later called Tutte's entropy. This entropy, approximately equal to 3.25 bpv, stands for an upper bound of the entropy of any arbitrary surface triangular mesh connectivity. This value was used to prove the optimality of several triangular mesh connectivity encoding schemes [Alliez and Desbrun 2001b]. Turán [1984] was actually the first to propose a practical method to encode, with about 12 bpv, a planar unlabeled graph thanks to a vertex spanning tree.

3.1.3. Spanning tree encoding of meshes. The influence of planar graph encoding techniques can be felt on some 3D mesh encoding techniques. For instance, the *topological surgery* algorithm of Taubin and Rossignac [1998] encodes a triangular mesh with about 2.5 to 6 bpv thanks to two spanning trees: a vertex and a triangle spanning tree. In the case there is twice more triangles than vertices, a variation of this algorithm can guarantee a 6 bpv compression of the connectivity (2 bits per vertex for vertex spanning tree and 2 bits per triangle for the triangle spanning tree telling whether each node has zero, one or two child). The Hand-and-Glove algorithm of Diaz-Gutierrez et al. [2005] encodes a genus-0 mesh with two types of vertex spanning trees (the Hand and Glove trees). These trees are built in order to form a triangle strip loop traversing the entire mesh. The trees are encoded with 2 bits per vertex and one additional bit per triangle allows to reconstruct the triangle strip. The total guaranteed cost is therefore 4 bpv. This representation also embeds a hierarchyless multiresolution structure (through the two types of tree collapsible edges) and a stripification. Experimental results report connectivity compression rates of about 1.5 bpv on average. The algorithm

of Li and Kuo [1998a] encodes the connectivity of a triangle mesh with its dual graph. Each node of the dual graph is incident to three edges. So, the encoding traversal has just to perform a breadth-first traversal of the mesh dual graph and outputs one binary symbol per edge telling if this edge is connected to an already visited node or not.

3.1.4. Triangle traversal encoding. Some algorithms were proposed to encode a mesh with a region growing approach by generating a triangle spanning tree composed of strips. To generate such trees, the algorithm iteratively processes the mesh triangles with a breadth first traversal. At each step of the compression, some faces have already been traversed and other not. There can be one or more closed edge borders between these two types of faces. The advantage of such methods is their simplicity. The spanning trees are easy to generate and directly contain all the mesh elements.

The *Cut-Border machine* [Gumhold and Straßer 1998] follows this strategy. It extends the border formed by an initial triangle by iteratively traversing adjacent triangles. 7 different symbols code whether the border was extended by inserting a new vertex, if the border was split or two borders were joined. The scheme can compress manifold triangle mesh connectivities with about 4 bpv. This result is however only valid for reasonably regular meshes. When two borders are joined, an offset must be encoded to designate the concerned vertices. Therefore, the algorithm does not guarantee a tight upper bound for the compression rate.

The Edgebreaker algorithm of Rossignac [1999] with its fixed format (one symbol per triangle and no additional offset) however guarantees a cost of 4 bpv. In practice, after entropy coding, a mesh is encoded with about 1.8 and 2.4 bpv. The algorithm encodes the connectivity of triangular meshes by iteratively nibbling its faces. Each time a new face is traversed, the configuration of its patch among the five depicted on Figure 5 is encoded. The face is then removed and an adjacent face is processed.

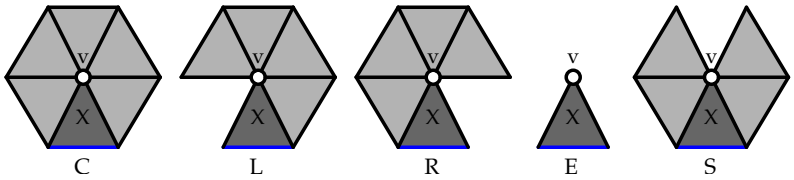

Fig. 5. The five patch configurations of the Edgebreaker algorithm. v is the patch center vertex and X is the current triangle. The active gate is the blue edge. C: there is a complete triangle fan around v. L: there are missing triangles at the left of the active gate. R: there are missing triangles at the right of the active gate. E: v is only adjacent to X. S: there are missing triangles elsewhere than the left or the right of the active gate.

Some Edgebreaker derived schemes were proposed to guarantee a worst-case coding cost of 3.67 bpv [King and Rossignac 1999] and then 3.55 bpv [Gumhold 2000]. Decoding algorithms with linear time and space complexities were proposed [Isenburg and Snoeyink 2000b][Rossignac and Szymczak 1999]. Szymczak et al. [Szymczak et al. 2001][Szymczak 2003] optimized the original scheme to encode meshes with a high regularity. This method has a worst-case coding rate of 1.62 bpv for large regular meshes. Coors and Rossignac [2004] with the Delphi coder added a connectivity prediction method based on the mesh geometry. The Edgebreaker configuration of the current vertex is predicted with the distances between the position of its parallelogram prediction (see Section 3.2.2) and the positions of the active gate vertices. The authors claim to obtain, on average, connectivity compression rates that are below 0.75 bpv. Gumhold [2005] proposed to optimize a Markov model, for which each Edgebreaker symbol is a state, in order to design an asymptotic optimal arithmetic coder for the

Edgebreaker encoder. Ying et al. [2010] designed an algorithm to select the next edge gate to proceed in order to minimize the number of 'S' symbols. They also built a table that ranks the Edgebreaker symbols depending on the number of traversed faces connected to the gate vertices. This *code-mode*, which stands for a connectivity prediction method, allows after an improved entropy coding.

The angle-analyser scheme [Lee et al. 2002] encodes the connectivity of a manifold triangular mesh at 1.5 bpv on average with five symbols. The next face to proceed is chosen in order to maintain the border between conquered and not conquered regions of the mesh the most convex as possible. Lee and Park [2005] later showed that this rate can be slightly reduced by using contexts for the arithmetic coder based on the angle between successive gates.

3.1.5. Valence encoding. A manifold triangle mesh contains approximately twice less vertices than triangles. So, an algorithm that focuses on the insertion of new vertices and generates one symbol per vertex to describe its local connectivity produces less symbols than a triangle traversal approach. Consequently, if the entropy per symbol is not twice larger this algorithm will lead to a better connectivity compression performance. One way to describe vertex connectivities is to encode their valences.

The pioneering valence-driven approach is the algorithm of Touma and Gotsman [1998]. Its principle is to consider the edge boundary formed by an initial triangle and expand this boundary by iteratively adding adjacent vertices. The connectivity is encoded by the valence of the inserted vertices, typically concentrated around six. Therefore, the generated list of vertex valences can be efficiently compressed by an entropy coder (2.3 bpv). It is still today seen as one of the most efficient connectivity compression method. Alliez and Desbrun [2001b] later proposed some modifications to further reduce the compression rates. They also demonstrated that the compression rate obtained with their method for a mesh with a maximized entropy matches Tutte's entropy [1962] (see Section 3.1.2). Thus, they claimed having demonstrated the optimality of valence-based approaches.

The Freelence encoder [Kälberer et al. 2005] has a slightly different approach. Instead of directly encoding the valence of vertices, it codes the number of not conquered edges incident to the processed vertex. This method goes along with a geometry-driven mesh traversal to keep the active list as convex as possible like the angle-analyzer encoder [Lee et al. 2002]. This algorithm yields an average improvement of 35% compared to the Alliez and Desbrun approach [2001b] for the compression of manifold triangle mesh connectivities.

3.1.6. Topological extensions. Many of the algorithms presented above do not accept any topology as input. Thus, the *topological algorithm* [Taubin and Rossignac 1998] first assumes a genus-0 manifold oriented triangle mesh with no boundary. The authors then propose some preprocessing and some additional information encoding so as to remove the genus-0 and no boundary conditions. The *Cut-Border machine* [Gumhold and Straßer 1998] and the Touma and Gotsmam valence approach [Touma and Gotsman 1998] can process manifold triangle meshes with boundaries and arbitrary genus. For the *Cut-Border machine*, an extension that codes an additional bit for some operations is proposed to process non orientable meshes. Mamou et al. [2009] proposed an extended valence approach to encode non-manifold and non-oriented triangle meshes. This algorithm partitions a triangle mesh into a set of triangle fans. For each fan, the encoded information is mainly its configuration code among 10 and its degree. The first version of the *Edgebreaker* algorithm [Rossignac 1999] takes as input genus-0 manifold triangle meshes with boundaries but Lopes et al. [Lopes et al. 2002] removed the genus limitation. Their scheme encodes additional symbols to describe how to turn the input mesh into into a single boundary loop mesh.

3.1.7. Compressing polygon meshes. Triangles are the most common type of faces inside meshes but not the only one. Meshes with higher face degrees have also their interest to represent some surfaces. The previously described methods cannot be directly applied to polygon meshes because they assume the mesh face degree is three to reduce the quantity of information to encode. A preliminary triangulation allows to compress polygon meshes with algorithms restricted to triangle meshes. However, this method do not restore the initial connectivity and may lead to higher compression rates as edges must be added. Therefore, the compression of polygon meshes has also been studied in the literature.

The algorithm of King et al. [1999] generalizes the Edgebreaker algorithm [Rossignac 1999][Rossignac and Szymczak 1999] to allow the compression of quad and quad-triangle mesh connectivities. Its principle is to split each quadrangle into two triangles that are on the same Edgebreaker traversal sequence. This leads to efficient connectivity compression (from 0.25 to 0.85 bpv for quad meshes and from 0.78 and 1.14 for quad-triangle meshes) because some combinations of the Edgebreaker algorithm become impossible. The paper also explains how to compress a polygon mesh with a guaranteed cost of 5 bpv.

The *Face Fixer* algorithm [Isenburg and Snoeyink 2000a] compresses the connectivity of manifold polygon meshes with arbitrary face degrees using a face traversal of the mesh. The encoder generates one symbol per edge. Experimental results yield connectivity compression rates ranging from 1.7 to 2.9 bpv. The connectivity encoder of Kronrod and Gotsmam [2000] codes the degree of each face of the mesh and its relation to the active border. The authors report their method as being slightly less efficient than [Isenburg and Snoeyink 2000a] but easier to describe and implement.

Isenburg [2002] and Khodakovsky et al. [2002] independently proposed valence-based approaches to encode the connectivity of manifold polygon meshes. Inspired by the work of Touma and Gotsman [1998], their approaches encode both the valence of vertices and faces with a face traversal of the mesh. Khodakovsky et al. [2002] demonstrated the optimality of such schemes by proving that the entropy of the two valence lists matches Tutte's entropy for planar graphs [1963]. For both methods, experimental results range approximately from 0.8 to 2.6 bpv, thus improving over the Face Fixer scheme [Isenburg and Snoeyink 2000a].

3.1.8. Compressing volume meshes. When the application requires to mesh the interior of a volume, volume meshes must be used. They contain more information than surface meshes to define the cells. It is therefore also interesting to compress them.

Szymczak and Rossignac [1999] proposed a spanning tree approach to encode tetrahedral meshes with about 7 bits per tetrahedron. The compressed format is composed of a tetrahedral spanning tree string and a folding spanning tree. This second tree codes the folding operations on the tetrahedral spanning tree edges needed to restore the initial mesh. Gumhold et al. [1999] proposed a border extension approach based on the original Cut-Border machine [Gumhold and Straßer 1998]. But here, instead of being a curve composed of edges, the border is a surface composed of triangles. Ten different symbols encode the connectivity. The connectivity of the tested models is compressed at less than 2.4 bits per tetrahedron.

Isenburg and Alliez [2002a] investigated the compression of hexahedral volume meshes with a valence-driven approach. Their scheme encodes the mesh edge degrees, their number of incident faces, during an iterative traversal of all the mesh hexahedra. The obtained average compression rate is 1.5 bits per hexahedron for the tested models. The hexahedral mesh connectivity compressor of Krivograd et al. [2008] first compresses the mesh boundary ; then, the interior of the mesh is compressed with six different symbols and vertex degrees. Experimental results show that this approach

compresses better (up to 50%) big meshes with a low genus compared to the scheme of Isenburg and Alliez [2002a].

Lindstrom and Isenburg [2008] proposed an original approach to compress hexahedral mesh connectivities. Their method encodes the vertex indices from the input file indexed data structure. In this file, the hexahedra are described by a table with 8 columns of vertex indices. The principle is to compress independently each column of the table. Indeed, if the vertices are coherently ordered, a context of few indices can accurately predict the next value. The prediction residual are then coded with byte-aligned variable length coding. The obtained string is later compressed by a standard *gzip* data compression algorithm. Even if this algorithm is not as efficient as state-of-the-art approaches, it has numerous advantages. Among them, it is fast and memory efficient. It processes non-manifold meshes without any adaptation. It is easy to implement: it does not need any particular mesh data structure and it heavily relies on the freely available data compression algorithm *gzip*.

Prat et al. [2005] described a generic algorithm to compress the connectivity of any kind of manifold meshes (surface or volume, orientable or not, with arbitrary face or cell degrees). They based their scheme on a generalized map data structure containing a single type of primitive elements called darts. Connectivity relations between these elements called involutions are stored. Even if this method is not competitive in term of compression rates with specialized approaches, its main advantage is its generic property. Its ability to process different types of meshes is an advantage in scenarios when the mesh type can vary.

Approaches have also been proposed for 3D regular grid compression, also known as voxel compression. As for lower dimensional gridded data (e.g. images and videos), several compression methods have been applied to volumetric data as for instance : vector quantization [Ning and Hesselink 1992], Discrete Cosine Transform (DCT) [Yeo and Liu 1995; Lum et al. 2002], Discrete Fourier Transform (DFT) [cker Chiueh et al. 1997], Run Length Encoding(RLE) [Anagnostou et al. 2000] or wavelets [Muraki 1992; Guthe and Straßer 2001; Bajaj et al. 2001; Rodler 1999]. These approaches are all lossy compression schemes. In [Fowler and Yagel 1995], the authors proposed the first lossless compression scheme for voxel data based on prediction. Similarly, in [Ibarria et al. 2003], the Lorenzo predictor is introduced for the compression of n-dimensional scalar fields.

3.2. Geometry compression

Mesh geometry compression (the compression of the vertex coordinates) is very important as, in most cases, it is bigger than the connectivity information. Usually the compression of the geometry of a mesh begins with the *quantization* of all the coordinates of its vertices. Then, during each compression iteration, the position of an encoded vertex is predicted thanks to the position of its already encoded neighbors. If the *prediction* is accurate, the prediction error is small so, it can be later efficiently entropy-coded. The compression rates provided in this subsection concerns only the geometry.

3.2.1. Quantization. In input files, or in the memory of computers, the vertex coordinates are often represented by 3 IEEE 32-bit floating-point numbers. The precision provided by such a representation is not needed for most applications. Quantization can significantly reduce the quantity of data to encode without any identifiable quality loss.

Scalar quantization consists in transforming the floating-point number positions into integer positions. The mesh bounding box is partitioned into a 3D grid. The number of cells per axis depends on the maximum integer that can be coded with the num-

ber of quantization bits. The size of each cell can be either uniform or non-uniform. Each vertex of the mesh is moved to the center of the cell it belongs to. The integer position is then composed of the three index coordinates of the cell.

Most of the geometry encoders that go along with the previously described well-known connectivity compression schemes [Deering 1995][Taubin and Rossignac 1998] [Rossignac 1999][Touma and Gotsman 1998] use an uniform scalar quantization. The number of quantization bits usually ranges from 8 to 16. The mesh geometry is consequently slightly altered contrary to the connectivity.

Bajaj et al. [1999] and then Lee et al. [2002] proposed to encode the vertex positions with three angles. For the angle-analyzer encoder [Lee et al. 2002] two internal angles and one dihedral angle are computed. Instead of being performed on global vertex coordinates, the quantization is performed on these local angles. By applying different quantizations to the different angles, this method can achieve a better rate-distortion performance. Lee and Park [2005] proposed to locate the vertices within 4 different range sizes as they noticed that very few vertices are located in the biggest range. To encode the position of the vertex within a range, the ranges are more or less subdivided depending on their size. The position of the vertex is therefore encoded by the type of the range and the subcell number.

Vector quantization is an alternative technique that divides the set of points to quantize into arbitrary shaped groups. Quantization cells are no longer cuboids. Their shape can better adapt to the data. Each group has a representative point. All of these points constitute the *codebook* that must be saved with the compressed data. Vector quantization has experimentally demonstrated its ability to achieve better rate-distortion performance than scalar quantization technique [Lee and Ko 2000] [Chou and Meng 2002][Bayazit et al. 2007][Lu and Li 2008][Meng et al. 2010]. However, the determination of the quantization cells can lead to intensive computations.

3.2.2. Prediction. The first mesh compression approaches [Deering 1995][Chow 1997] only used *delta prediction*. The position of the next vertex to encode is predicted to be the position of the previous vertex. The delta, the vector between the two positions, is then encoded. Bajaj et al. [1999] uses a *second order predictor* that encodes the differences between consecutive delta predictions.

The topological surgery algorithm [Taubin and Rossignac 1998] uses *linear prediction*. The position of the next vertex to encode is predicted to be a linear combination of the K previous vertices in the vertex spanning tree. The K parameters of the linear function minimize the mean square prediction error over the mesh. Their values are stored in the compressed file to be available to the decoder.

Besides valence-driven connectivity encoding, Touma and Gotsman [1998] also introduced *parallelogram prediction*. Like valence-driven connectivity encoding, parallelogram prediction is a founding idea that has inspired many later schemes. The compression algorithm introduces a new vertex with a triangle from one edge. The new vertex predicted position forms a parallelogram with the two edge vertices and the third vertex of the opposite triangle (Figure 6 *a*). Experimental results show that triangle mesh geometry can be compressed at about 8.5 bpv with a 8 bit quantization.

The *dual parallelogram prediction* [Sim et al. 2003] uses the average position given by two parallelogram predictions whenever it is possible (Figure 6 *b*). Dual parallelogram can be used in about 75 percent of the cases. It results to slight improvements over *parallelogram prediction*.

Isenburg and Alliez [2002b] showed that parallelogram prediction can also be used for the geometry compression of polygon meshes with arbitrary face degrees. Isenburg et al. [2005a] later presented a generalization of the parallelogram prediction for arbitrary polygons. Missing vertex positions of a polygon are predicted with weights com-

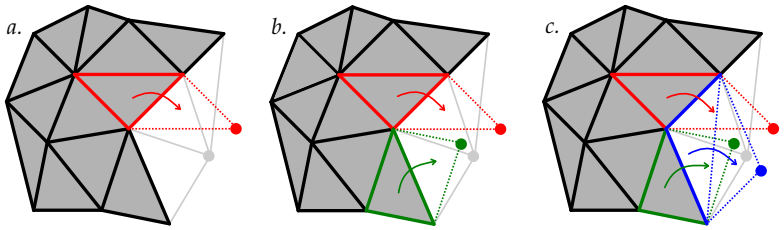

Fig. 6. Parallelogram prediction. The already encoded part is in gray. *a.* Simple parallelogram prediction. *b.* Dual parallelogram prediction. The predicted position is the average of the two parallelogram predictions. *c.* Freelence prediction. The predicted position is the average of the three parallelogram predictions.

puted for polygons of different degrees with different known vertices. These weights come from a Fourier decomposition of polygons where the highest frequencies are set to 0 to ensure that the polygons are nicely shaped.

The Freelence coder [2005] uses the combination of three parallelogram predictions to encode the geometry of triangular meshes: two standard ones, as with dual parallelogram prediction, and a new one applied across a virtual edge joining the two outer vertices (Figure 6 *c*).

Cohen-or et al. introduced *multiway prediction* [2002]. The principle is to predict the position of new vertices with as many as possible parallelogram predictions based on already encoded vertices.

The geometry predictor of Gumhold and Amjoun [2003] performs as many as possible parallelogram predictions to determine the tangential components of the prediction. However, to estimate the normal component, encoded by a dihedral angle, the encoder fits a high order surface to the already encoded part of the geometry. The predicted position is therefore at the intersection of the higher order surface with the circle defined by the tangential components. Ahn et al. [2006] also use as many as possible parallelogram predictions. They also added their own dihedral angle prediction scheme that averages all the neighboring dihedral angles.

Recently, Váša and Brunett [2013] proposed weighted parallelogram prediction methods, which weights are computed in function of the vertex valences. The three proposed weight calculation methods have an increasing computational cost. Experimental results report that residuals have a computed Shannon entropy up to 20% lower than with standard parallelogram prediction.

Courbet and Hudelot [2011] used a Taylor expansion to determine prediction weights for various prediction stencils. This method allows to theoretically determine the best weights for the parallelogram prediction. It proves that the Freelence weights [Kälberer et al. 2005] work better than the dual parallelogram prediction weights [Sim et al. 2003] and determines better weights for the polygon prediction [Isenburg et al. 2005a].

Regarding volume meshes, Gumhold et al. [1999] resorted to delta coding to encode the geometry of tetrahedral meshes. Isenburg and Alliez [2002a] compressed the geometry of hexahedral meshes with intra and inter hexahedron parallelogram predictions.

3.2.3. Geometry-driven compression. For a compressed mesh, the size of the geometry information is often superior to the size of the connectivity. Nevertheless for most algorithms, the encoding is driven by the mesh connectivity. So, the idea of focusing on geometry encoding before connectivity has been investigated.

Kronrod and Gotsman [2002] designed a compression scheme where the mesh encoding is driven by the geometry. This approach builds a mesh cover tree that contains all the mesh vertices. This tree, which defines a particular traversal of the mesh ver-

tices, is generated to minimize the parallelogram prediction errors needed for each vertex position. The geometry compression becomes up to 50 percent more efficient at the cost of a slightly higher connectivity compression rate. This approach particularly benefits to CAD models, which often have a non-smooth geometry.

The tetrahedral mesh compression scheme of Chen et al. [2005] is also driven by the geometry. It tries to build an optimal traversal tree that minimizes the prediction errors of the new vertex positions. The used predictor is the generalization of the parallelogram prediction for tetrahedral meshes.

Shikhare et al. [2001] pushed the geometry-driven compression idea one step further. Their scheme tries to find repeated geometric patterns inside 3D models. The recognized patterns can be either components, regions within components or region across components. This approach particularly benefits to large CAD or digital heritage models. Cai et al. [2009] later proposed a similar scheme that achieves slightly better performance. They included scaling transformations for the repeated patterns.

The connectivity encoding of the previous schemes is guided by the mesh geometry. But the mesh geometry and connectivity are compressed at the same time and their data is interleaved in the compressed stream. Lewiner et al. [2006] with the GEncode algorithm proposed an alternative geometry-driven encoding technique. The mesh geometry is first compressed completely independently from the connectivity by encoding a kd-tree decomposition of the quantized space. Then, a surface reconstruction algorithm iteratively attaches new triangles to the border of the mesh by selecting a new vertex among candidates around. The reference to the right vertex among the candidates is encoded. This algorithm can compress any kind of triangle meshes. This approach is very competitive for the compression of tetrahedral meshes [Lewiner et al. 2006] compared to the Grow & Fold [Szymczak and Rossignac 1999] and streaming [Isenburg et al. 2006] approaches.

In a similar idea, Chaine et al. [2009] proposed a mesh connectivity compression scheme based on surface reconstruction. For the decompression, this technique assumes that the geometry has already been decoded. For both the encoding and decoding, a Delaunay triangulation is generated from the point sets. Then a convection algorithm restores the initial connectivity of the input mesh. The connectivity of meshes generated with similar techniques is therefore encoded at a very low cost.

3.2.4. Compressing floating-point positions. As described above, most of the mesh compression algorithms quantize the coordinates of the vertices. But some applications may require the exact restoration of the floating-point coordinates. Isenburg et al. [2005b] investigated the lossless compression of floating-point geometry. The floating-point coordinates are broken into sign, exponent, and mantissa components. The prediction errors of these components are compressed independently with different arithmetic contexts. This method is able to compress the geometry of a mesh at about 35 bpv.

3.3. Handling large meshes

Some models are too large to fit into the main memory of computers. *Out-of-core* schemes were designed to compress such models. They dynamically load and unload the different parts of the mesh depending on the currently processed region of the mesh.

In [Ho et al. 2001], the authors proposed to partition the input model. Each part is then compressed independently in-core. The Edgebreaker compression algorithm [Rossignac 1999] compresses the connectivity. Additional symbols are integrated to stitch the parts together during the decompression. For the geometry compression the parallelogram prediction [Touma and Gotsman 1998] is used. In the same idea, Ueng

[2003] proposed to compress large tetrahedral meshes connectivity by dividing them into blocks called octans with an octree data structure based on the geometry. Each octan is compressed in-core by encoding tetrahedral strips.

Isenburg and Gumhold [2003] proposed an out-of-core mesh data structure for the compression of large polygon meshes. This data structure is built on a segmentation of the input mesh. However, contrary to the previous out-of-core approaches, the clusters are compressed together with a *streaming* approach. During the compression, the data structure dynamically loads the mesh clusters that are on the active list of the Touma and Gotsman encoder [1998] and unload them when they are no longer needed. The decompression is performed with a small memory footprint composed only of the active list. The authors claim that the obtained compression rates are about 25 percent lower and the decompression speeds about 100 times faster than Ho et al. scheme [2001].

Following their idea of processing meshes with a streaming approach, Isenburg and Lindstrom [2005] proposed a general I/O efficient streaming format for meshes. This format interleaves indexed triangles and vertices with extra information describing when mesh elements are introduced and finalized. Therefore, the application keeps only in memory the small active part of the mesh currently being processed. The authors propose several strategies to reorder the meshes indices in a layout compatible with streaming.

This framework has been the basis of several streaming compression techniques such as [Isenburg et al. 2005c] for triangle meshes, [Isenburg et al. 2006] for tetrahedral meshes and [Courbet and Isenburg 2010] for hexahedral meshes. Compared to their non-streaming counterparts, these techniques have in general equivalent geometry compression rates but worse connectivity compression rates. Yet, the I/O efficiency of the streaming approach allows them to compress large meshes quickly with a very low memory footprint.

The *Tetstreamer* algorithm [Bischoff and Rossignac 2005] has been designed for client-server visualization purposes also following a streaming strategy. The server compresses and sends to the client a tetrahedral mesh in the back to front visibility order. The input mesh is decomposed into surface layers called sheets that are compressed by the *Edgebreaker* algorithm [Rossignac 1999]. The client also receives a bit stream indicating how to attach each tetrahedron to the current sheet. The connectivity of tetrahedral meshes is compressed at about 1.7 bits per tetrahedron on average.

3.4. Compression and remeshing

Most of the compression schemes do not alter the connectivity of the mesh. But some applications do not require the restoration of the initial connectivity after the decompression. In this case, the compression rates can be further improved by exploiting this new degree of freedom.

Szymczak et al. [2002] proposed a remeshing algorithm that produces piecewise regular meshes. The input mesh is first segmented into approximately flat regions called reliefs. Each relief belongs to one of six different directions. They are then resampled over a regular hexagonal grid defined by three families of parallel lines. All the original vertices, except the boundary ones, are collapsed. A stitching algorithm reforms a closed mesh. This scheme is associated with an Edgebreaker compression algorithm optimized for regular meshes for the connectivity. The Swingwrapper scheme [Attene et al. 2003] produces also semi-regular meshes. It aims at generating isosceles triangles to encode only one quantized dihedral angle when their inserted vertex is encoded by an Edgebreaker encoder. Both methods benefit from the high regularity of the generated meshes to compress them very efficiently (on average 4 bpv in total for [Szymczak et al. 2002] and 6 bpv for the geometry for [Attene et al. 2003]).

Mesh simplification can also be considered as lossy compression as it reduces the connectivity and the geometry. A review of the mesh simplification method is beyond the scope of this paper but we can redirect the interested reader to this useful survey: [Cignoni et al. 1998]

Table I summarize what we consider as most relevant and seminal single-rate compression approaches.

Table I. Summary of the main single-rate compression algorithms.

Algorithm	Connect. comp. rates (bpv)	Compress polygon meshes	Embed strips	Remarks
Deering [Deering 1995]	11	no	yes	
Topological surgery [Taubin and Rossignac 1998]	6 max. 2.5 to 6	no	no	
Cut border machine [Gumhold and Straßer 1998]	4.4 on avg.	no	no	
Valence coder [Touma and Gotsman 1998]	2.3 on avg.	no	no	Introduced valence encoding & parallelogram prediction
Edgebreaker [Rossignac 1999] [Gumhold 2000]	3.55 max. 2.1 on avg.	no	yes	Many derived schemes
Valence polygonal [Khodakovsky et al. 2002] [Isenburg 2002]	1.8 on avg. for poly. meshes	yes	no	Proven near-optimality of the connect. coding
Valence coder [Alliez and Desbrun 2001a]	2.1 on avg.	no	no	Proven optimality of the connect. coding

4. STATE OF THE ART ON PROGRESSIVE MESH COMPRESSION

With progressive algorithms, a coarse version of the mesh can be quickly displayed to the user. It is then progressively refined as more data is decompressed until the initial model has been restored. *Connectivity-preserving* schemes restore during the decompression the connectivity of the input model while *connectivity-oblivious* schemes resort to remeshing to encode an input model with a higher compression performance.

4.1. Connectivity-preserving schemes

4.1.1. Vertex split and edge collapse. Hoppe first introduced the concept of *progressive meshes* (PM) [Hoppe 1996]. The idea is to incrementally decimate a mesh using the edge collapse operator (see Figure 7) driven by an optimization procedure that minimizes an energy function. The compressed representation consists of the base mesh followed by all parameters required for the incremental reverse operations, called *vertex splits*. Hoppe encoded the connectivity of a vertex split by storing the index of the vertex v_s and approximately five bits to designate the vertices v_l and v_r (see Figure 7). For the geometry encoding, the vertex positions are globally quantized and encoded through delta prediction. The main advantage of this scheme is its high multiresolution granularity, together with the possibility to perform selective refinement during decoding. Such granularity is achieved at the cost of low compression rates: in the order of 37 bpv with a 10 bit quantization. Popović and Hoppe [1997] then extended the PM representation to arbitrary simplicial complexes. Their main contribution is the introduction of the *generalized vertex split* (and its counterpart the *vertex unification*) which allows topological changes along the levels of detail which may thus be composed of 2-, 1- and 0-simplices (i.e. triangles, edges and vertices). To keep a good visualization of the levels of detail, 1- and 0-simplices are approximated by cylinders and spheres of appropriate areas.

In order to come closer to compression rates of single-rate methods, some methods were proposed to encode the vertex split operations in batches. Taubin et al. [1998] built a progressive mesh compression scheme inspired by the single-rate *topological*

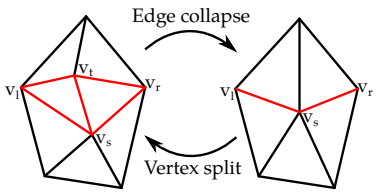

Fig. 7. Edge collapse and vertex split operations. The edge collapse merges the vertex v_t with the vertex v_s. The vertex split inserts the vertex v_t and the two triangles with the red edges.

surgery algorithm [Taubin and Rossignac 1998]. Their *progressive forest split* representation encodes a manifold triangular mesh with a base mesh and a sequence of *forest split* operations. The connectivity is encoded by marking the cut edges and the way the holes are retriangulated.

Pajarola and Rossignac [2000] proposed to perform as many as possible edge collapse operations to generate of a new level of detail. Vertex splits are then encoded by building a vertex spanning tree on the mesh and saving each split vertex and cut edges. The vertex positions are uniformly quantized but Li et al. [2006] later demonstrated that the use of vector quantization (see Section 3.2.1) improves the rate-distortion performance of the algorithm. Karni et al. [2002] adapted this scheme to design a progressive compression scheme which enables the fast rendering of all the levels of detail with vertex buffers. Vertex buffers have been introduced by the graphic hardware manufacturers to accelerate rendering with the reuse of vertex data through generalized triangle strips (see Section 3.1.1). The first step of the proposed algorithm is to create an efficient vertex rendering sequence composed of series of incident vertices. The mesh is then decimated by collapsing edges along this sequence.

Better compression ratios are achieved with these approaches (about 30 bpv [Taubin et al. 1998] and 22 bpv [Pajarola and Rossignac 2000] with a 10 bit quantization). Nevertheless the multiresolution granularity is lower than in the PM representation.

4.1.2. Vertex removals. Other progressive compression schemes use vertex removals instead of edge collapses. Li and Kuo [1998b] pioneered a method based on vertex removal followed by a local patch retriangulation. The connectivity is encoded with a local index which specifies the patch pattern and a global index which locates this pattern in the whole mesh. The geometry data is encoded with a barycentric error prediction. The authors also pioneered the idea of adapting the vertex quantization along the transmission of the levels of detail.

Cohen-Or et al. [1999] used the same decimation mechanism. However, they grouped the vertex removal into batches to generate discrete levels of detail. For the generation of a new level, all the patches of the removed vertices must be independent. The patches are then identified by assigning an identical color among three to triangles belonging to the same patch. Compression rates competitive with single rate techniques can be achieved with this method (about 23 bpv with a 12 bit quantization).

Alliez and Desbrun [2001a] proposed what could be seen as a progressive version of the Touma-Gotsman single-rate encoder [1998]. At each iteration, the mesh is decimated by two deterministic mesh traversals. The *decimating conquest* (from a to b in fig. 8) removes vertices with a valence inferior or equal to 6 (for the border vertices, only vertices with a valence of 3 or 4 are removed). The created holes are then filled by a deterministic retriangulation strategy which operates as follows : during the conquest, driven by a fifo queue of seed gates (blue arrows), vertices are tagged either $\oplus$ or $\ominus$ depending if their valence should be increased or decreased (to obtain a mesh as regular as possible); then, for each gate, a table provides the accurate triangulation ac-

cording to these flags and propagates accurate flag values to untagged vertices. After this *decimating conquest*, a *cleansing conquest* (from b to c in fig. 8) removes valence 3 vertices. The mesh connectivity is encoded through the valence of the removed vertices plus one null patch code to encode triangles not belonging to patches. The geometry is encoded through the patch barycentric error prediction in a local Frenet frame. The obtained compression rates are about 21 bpv with a 12 bit quantization.

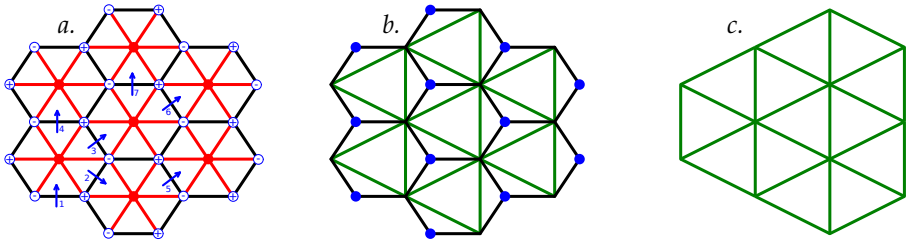

Fig. 8. Decimation of a regular mesh by the Alliez-Desbrun progressive encoder. a. The red vertices and the red edges of the input mesh will be removed by the decimating conquest. b. The deterministic triangulation fills the holes with the green edges. c. The blue vertices have been removed by the cleansing conquest.

Since valence-driven encoding of the mesh connectivity has demonstrated its efficiency for single-rate and progressive mesh compression, the original Alliez-Desbrun has inspired numerous derived schemes that have improved its rate-distortion performance. For instance, Cheng et al. [2006] forbid the removal of some vertices, called anchors and detected by their principle curvatures, during the decimation conquests. Experiments show that keeping these vertices in all levels of detail allows to improve the rate-distortion performance. Lee et al. [2012] demonstrated that the rate-distortion trade-off of the Alliez-Desbrun coder is improved by using an adaptive quantization method. The idea consists in interleaving decimation operations and global quantization operations encoded with the Peng and Kuo approach [2005] (see section 4.1.3). They obtain both better rate-distortion performances and compression rates (about 1bpv improvement with a 12 bit quantization).

Ahn et al. [2011] optimized the mesh traversal to maximize the number of removed vertices per decimation step. A curvature prediction is also used for encoding the geometry. The latter shares the general idea of spectral methods (see. Section 4.1.6) because a topology-based *Karhunen-Loève transform* concentrates the distribution of geometry residuals. The residuals are entropy coded with a bit plane coder. Decimation conquests are interleaved with the transmission of bit planes to improve the rate-distortion performance. They significantly improve the compression rates of the Alliez-Desbrun coder (about 4 bpv reduction with a 12 bit quantization).

Lee et al. [2011] significantly improved the geometry compression (up to 60%) by introducing two new prediction methods. The *dual ring* prediction aims at having the same barycentric prediction for one vertex and its one-ring neighbor vertices. The *minimum mean square error* prediction constructs a linear predictor based on vertices having a topological distance of 1 or 2 with the predicted vertex. The algorithm first segments the input level of detail by an algorithm based on the mesh connectivity. Then, the most efficient method is chosen to encode each cluster.

To improve the connectivity encoding, Kim et al. [2011] proposed to predict the valence of the current patch inserted vertex during the decoding. With the geometry information, each patch possibility corresponding to one valence value is tried. The position of the inserted vertex is provided by the encoded geometry information. Then all the possibilities are ranked by measuring the regularity of the generated triangle.

The predicted possibility is the one with the highest regularity. For each level of detail, the frequencies of each valence value is also used for better prediction.

4.1.3. Geometry-driven progressive mesh compression. As with single-rate mesh compression (see Section 3.2.3), the idea of focusing on geometry compression before connectivity has also been tested for progressive mesh compression.

In the scheme of Gandoin and Devillers [2002], the vertex positions are stored in a kD-tree. The initial cell is successively split along the three axis until the desired geometry precision is obtained. Each time a split is performed, the number of vertices of one of the child cell is encoded (see the Figure 9). The connectivity is encoded with generalized vertex splits and prediction mechanisms based on the geometry.

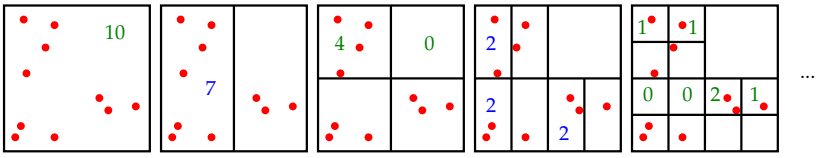

Fig. 9. Progressive geometry encoding of the Gandoin-Devillers algorithm in 2D. For each step, the emitted symbols are the numbers that represent the number of vertices in each cell.

Peng and Kuo [2005] also developed a geometry-driven progressive mesh compression algorithm based on an octree data structure. After a subdivision, instead of encoding the number of vertices inside each cell as in [Gandoin and Devillers 2002], the number of non-empty child cells is encoded. Then, the neighbor vertices allow to predict which cells may be non-empty. Rate-distortion performance is improved by prioritizing the subdivision of important cells. The connectivity of the mesh is encoded through vertex splits. The number of vertices connected to the two vertices generated by the split, called *pivot vertices* is encoded. Then a prediction algorithm based on the geometry determines the most probable candidates. Triangle meshes are compressed at about 15 bpv with a 12 bit quantization. Tian et al. [2012] later proposed an alternative method to predict non-empty cells based on smoothness measure. They also rank the cells that must be subdivided according to the valence of their representative vertex.

One advantage of these progressive compression methods based on space subdivision is that they can compress arbitrary simplicial complexes. Besides, they achieve very efficient compression rates with manifold meshes. However, their rate-distortion performance suffers at low rates due to the low quantization. No control of the important vertices removal is permitted. The mesh simplification cannot be driven with a metric contrary to connectivity-guided approaches.

4.1.4. Wavelet for irregular meshes. Wavelet frameworks are traditionally reserved for semi-regular connectivities as explained in Section 4.2.1 but Valette et al. [2004a] proposed a subdivision scheme that can generate irregular meshes. They later built a progressive mesh compression algorithm based on this framework: Wavemesh [Valette and Prost 2004b]. The initial mesh is progressively decimated with the subdivision scheme tailored to irregular meshes. The connectivity data is composed of all face subdivision operations. A triangle face can be subdivided into 4, 3, 2 faces or be unchanged. The connectivity is encoded with three types of data described on Figure 10. The geometry is encoded through a wavelet lifting scheme. The positions of the inserted vertices are predicted as being at the middle of their parent edges. The obtained compression rates are slightly better than those from [Alliez and Desbrun 2001a] (about 19bpv with a 12 bit quantization).

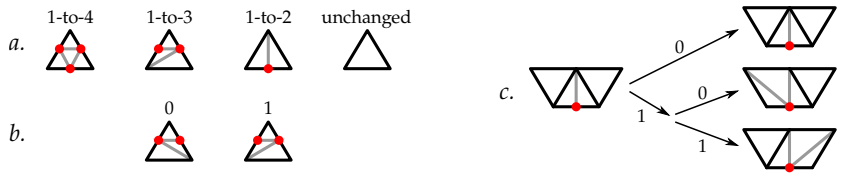

Fig. 10. Connectivity encoding with the Wavemesh scheme [2004b]. *a.* The four types of face subdivisions. Red vertices are added to split the edges. *b.* In the case a face is subdivided into 3 faces, one bit encodes the direction of the faces. *c.* In the case a face is subdivided into 2, to restore an initial connectivity modified to merge faces, one bit encodes if an edge needs to be flipped. If yes, an other bit codes which edge.

Lee et al. [2013] improved the compression performance of the original Wavemesh scheme by 16.9% on average. To improve the connectivity encoding, they proposed methods based on mixture of Gaussian probability models to predict the inserted vertices, the face directions and the edge flips from the geometry. For the geometry encoding, they divide the new vertices of a level of detail into three groups. To encode the vertex positions of a later group, the encoder uses the positions of the already encoded group(s). Vertex positions are predicted with the dual-ring prediction scheme from [Lee et al. 2011]. The residuals are encoded in a local coordinate frame like in [Alliez and Desbrun 2001a] with a bit plane coder.

4.1.5. Progressive compression through reconstruction. Valette et al. [2009] cast the progressive mesh compression problem as a mesh generation problem in their incremental parametric refinement framework. The encoder starts from a coarse version of the initial mesh generated by the simplification scheme from [Garland and Heckbert 1997]. The base mesh is incrementally refined by splitting the longest edge. The position of the inserted vertex corresponds to the position of one vertex of the original mesh. Positions are adaptively quantized. The number of quantization bits depends on the level of refinement. At each refinement step, the triangulation is modified by means of edge flips to satisfy a local Delaunay property or to fix connectivity drifts. This process is summarized on Figure 11. When all vertices of the original mesh have been inserted, the initial connectivity is restored by flipping edges guided by a flip distance heuristic. A bit per edge is encoded to tell if an edge must be flipped. This algorithm is known to compress efficiently (about 15bpv with 12 bits quantization) and achieves good rate-distortion performances. The complete connectivity restoration process is, however, not guaranteed to succeed.

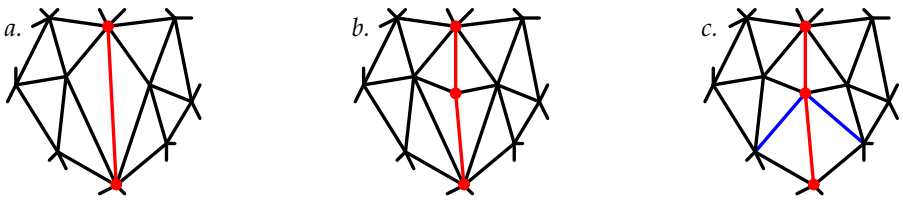

Fig. 11. Incremental parametric refinement [2009]. *a.* The longest mesh edge in red is selected for refinement. *a.* The red edge is split into two. *b.* The two blue edges are flipped to satisfy a local Delaunay property.

The idea of computing the best decimated version of an initial mesh has been recently further investigated in [Peng et al. 2010]. The algorithm starts from the initial mesh vertex set and recursively splits it into several child subsets using generalized Lloyd's algorithm [1982]. Each time a new vertex subset is generated, a representative vertex of this set is selected to be close to the geometric center of the set and to

have high curvature. In this hierarchy, the number of children of a set is encoded. The offsets between a representative and its parent representative are predicted in a cylindric frame and adaptively quantized. The connectivity is encoded through vertex splits with prediction of pivot vertices. This algorithm yields compression rates at about 16 bpv with a 12 bit quantization.

4.1.6. Geometry compression with the Laplacian operator. Fourier analysis has been commonly used for sound and image compression. Projecting the data into the frequency domain and encoding the low frequencies often allows to retrieve during the decompression a quality approximation of the original signal with few data. Karni and Gotsman [2000] proposed to apply this technique to compress mesh vertex positions. To project the coordinates from the space domain to the frequency domain, the encoder uses the mesh *Laplacian operator*. The eigenvectors of the Laplacian matrix of a mesh form an orthogonal basis of $\mathbb{R}^n$ and their associated eigenvalues are considered as frequencies. The projections of each coordinate component vectors on the basis vectors are the mesh spectrum. The underlining principle is that if the geometry is smooth enough, its spectrum will be concentrated around low frequencies. So the spectral coefficients of low frequencies, after being quantized and entropy coded, are sufficient to build a good approximation of the initial mesh. In practice, spectral compression achieves excellent rate-distortion performance. Incidentally, Ben-Chen and Gotsman [2005] proved that spectral compression is optimal for certain classes of geometric mesh models as it is equivalent to principal component analysis on these classes. Mahadevan [2007] replaced the laplacian bases by diffusion wavelet bases and showed their ability to better represent an input mesh with the same number of basis functions.

As eigenvector decomposition is a high complexity operation ($O(n^3)$), the mesh must be segmented in regions of about 500 vertices in [Karni and Gotsman 2000]. The computation becomes realistic but is still very heavy. A later approach [Karni and Gotsman 2001] proposed to use a fixed basis derived from a regular connectivity mapped to the irregular connectivity of the mesh. The hope is that the average energy will still be concentrated on low frequencies. The compression becomes less efficient but the decompression is accelerated since the eigenvector decomposition is not anymore computed. Bayazit et al. [2010] reduced the computational complexity of the mapping method. They also used a bit plane encoder for the spectral coefficients. Cayre et al. [2003] showed that rate-distortion gains can be obtained by introducing overlap between the segmented regions.

Mamou et al. [2010] devised an alternative algorithm that computes the Laplacian matrix of a mesh. The mesh is then approximated by solving a heat equation with a minimal set of control points. The vertex locations are encoded as residuals from the approximation given by the heat equation. The connectivity is encoded by the Touma and Gotsman single-rate encoder [1998]. This scheme achieves an excellent compression ratio (about 10 bpv with a 12 bit quantization). However, its high complexity due to solving the heat equation is a significant drawback.

4.1.7. Polygon meshes. Most of the approaches described above can only compress triangular meshes. For some of them, simple extensions were proposed to compress meshes with arbitrary face degrees. A preliminary triangulation allows to compress polygon meshes with an existing method restricted to triangle meshes. Additional data has then to be encoded to restore the initial connectivity after the decompression. Taubin et al. [1998] followed this approach to extend the progressive forest split algorithm. Li et al. [1998b] stated also that their compression scheme can be used on polygonal meshes. However, they provide no details about how the algorithm would be adapted. In more recent work, Peng and Kuo [2005] noted that, as their octree coder can compress arbitrary connectivity between vertices, it is possible to modify the face

construction algorithm to reconstruct polygon faces with two bits per split as additional information.

The encoder of Maglo et al. [2012] was specifically designed to progressively encode polygon meshes. Successive levels of detail are generated by a patch decimation operator which removes patch center vertex and remeshes locally. The mesh connectivity is encoded by two lists of Boolean symbols: one for the inserted edges and the other for the faces with a removed center vertex. The mesh geometry is encoded with a barycentric error prediction of the removed vertex coordinates. This scheme remains also competitive for the compression of triangle meshes.

4.1.8. Volume meshes. As far as we know, very few works have been proposed for the progressive compression of volume meshes. Pajarola et al. [1999] designed a progressive compression algorithm for the connectivity of tetrahedral meshes. To generate the levels of detail, batches of independent edge collapses are performed. The connectivity is encoded by marking with one bit the vertices that must be split during the decompression and identifying the cut-faces around a split-vertex. An alternative approach can be to use the geometry-driven approaches [Gandoin and Devillers 2002][Peng and Kuo 2005] (see Section 4.1.3) as they can compress arbitrary simplicial complexes. They, however, do not integrate the notion of cells.

4.2. Connectivity-oblivious schemes

As for single-rate mesh compression (see Section 3.4), when the restoration of the initial mesh connectivity is not crucial, resorting to remeshing can further reduce the compression rates.

4.2.1. Wavelet for semi-regular meshes. In the general case, progressive mesh compression schemes based on wavelet start from a coarse irregular version of the input mesh and progressively refine it with a subdivision scheme that produces semi-regular meshes. After each subdivision, the vertices are moved to reduce as much as possible the distortion between the subdivided model and the input mesh. These delta displacements, which are the wavelet coefficients, are then encoded. Loop's subdivision [1987] or the butterfly subdivision [Dyn et al. 1990] are often used for this purpose.

In image coding, wavelet representations are known to decorrelate efficiently the original data. As a consequence, Khodakovsky et al. [2000] proposed to also use wavelet for the compression of surfaces of arbitrary topology. However, this compression scheme cannot encode any connectivities. A remeshing of the input model is performed by the MAPS algorithm [Lee et al. 1998]. The wavelet transform, based on Loop's subdivision [1987], replaces the original mesh with a coarsest irregular mesh and a sequence of wavelet coefficients expressing the difference between successive semi-regular levels of detail. The wavelet coefficients are represented in a local frame and later encoded with a zero-tree coder. This scheme was later improved [Khodakovsky and Guskov 2003] thanks to the normal mesh representation [Guskov et al. 2000]. Normal mesh is a wavelet decomposition where the detail coefficients contain only one normal component wherever it is possible (above 90% of the cases) instead of three coefficients for standard subdivision. As there is only one wavelet coefficient to encode instead of three, normal mesh compression yields significantly better compression ratios.

Payan and Antonini [2002][2005][2006] allocate the bits across the wavelet subbands for the standard and normal mesh representations in order to improve the rate distortion performance. The proposed optimization framework varies the quantization of the wavelet coefficients to minimize the global reconstruction error. To improve the compression of normal meshes, Lavu et al. [2003] optimize locally the quantization of the normal component based on values previously encoded in the neighborhood.

Kammoun et al. [2012] proposed to optimize the prediction scheme of lifting-based wavelet transforms (Butterfly and Loop). For each level of detail, the best parameters of the predictor are computed to minimize the set of detail coefficients. Experimental results show that, compared to classic wavelet decomposition, the root mean square distortion is slightly reduced (about 2%) at similar rate. Chourou et al. [2008] resorted to mesh segmentation in order to optimize the wavelet operators for each of the generated clusters. The parameters of lifting scheme prediction step are chosen to minimize the variance of the detail coefficients. Zhao et al. [2011] based their compression scheme on matrix-valued Loop's subdivision for better shape control. The encoder of Denis et al. [2010] exploits the statistical dependencies between the intraband and composite wavelet coefficients to determine the best quantizers.

Chen et al. [2008] proposed to compress an input surface by regularly remeshing it with quads. The quadrilateral subdivision splits each face into 4 new faces by inserting one vertex. The wavelet decomposition is formulated through a lifting scheme. Zerotree coding is also used to encode the coefficients.

Other mesh methods using wavelet transforms focus on mobile decompression [Ma et al. 2009] and the support of lossy transmission [Luo and Zheng 2008]. The ideas behind are to reduce the number of computations needed for the decompression and to use error protection techniques to fit mobile computational capabilities and lossy networks.

As wavelet-based compression schemes resort to remeshing, it is not really possible to give absolute compression rates for compressed models. Indeed, neither the geometry (even with a set quantization) or the connectivity are restored during the decompression. Nevertheless in general, wavelet-based algorithms provide a significantly better rate-distortion performance compared to their lossless counterparts.

4.2.2. Geometry image. Geometry image [2002] is an original approach that compresses manifold meshes with a wavelet image compression scheme. To resample the input model over a regular 2D grid, the mesh is first cut in order to be homeomorphic to a disc. Then a parametrization function that maps the points of the cut mesh to the points of an unit square allows to compute x y z values for each pixel of the image. During the decompression, the geometry image pixels are used to build a triangle mesh approximation of the original mesh. However, the lossy compression leads to 'cracks' along the surface cuts. Hoppe and Praum [2005] later proposed to construct the geometry image using a spherical remeshing approach. To unfold the sphere on to geometry image, they proposed a scheme based on a regular octahedron domain and an other scheme based on flattened octahedron domain. The geometry of these domain fits nicely with the use of spherical wavelet, thus avoiding boundary reconstruction issues.

Shi et al. [2012] improved the compression of normal-map images since they are generally more difficult to compress due to their high variations. Their framework exploits the correlation between the normal-map image, the geometry image and between the three components of the normal-map image. Sander et al. [2003] mapped the surface piecewise on several charts so as to reduce the distortion. Peyré and Mallat [2005] investigated the compression of geometry images with bandelets. The aim of the bandeletization is to remove the correlation between high amplitude wavelet coefficients. Experimental results show that the distortion can be reduced by about 1,5 db compared to classical wavelet compression at similar rates.

Mamou et al. [2010] proposed a progressive mesh compression method based on b-splines and geometry images. After a segmentation of the input mesh, each patch is parameterized and approximated by a b-spline surface. The B-Spline control points are quantized and encoded into three gray-scales geometry images (one per axis) with

the progressive JPEG 2000 encoder. The connectivity of the patches is lossless encoded with the Touma and Gotsman algorithm [1998]. Additional encoded information allows to recover the patch adjacency relations.

Ochotta and Saupe [2008] proposed an alternative image-based surface compression method. The input mesh is first partitioned. Each region is then projected on a plane. The resulting height fields are transformed into images and compressed with an adaptive wavelet coder. After the decompression, the partitions are stitched in order to generate a closed mesh. The obtained experimental results are similar to the results of the normal mesh approach [2000].

Table II summarizes most important and seminal approaches. *Prog. Granularity* represents the granularity of progressiveness (from 1 - lowest, to 5 - highest) and refers to the number of new elements added when increasing the level of detail. The highest granularity is obtained by the progressive algorithm from Hoppe [1996], for which one vertex is added at each level of details. A low granularity is obtain for Wavemesh [Valette and Prost 2004b] which generates a low number of levels of detail (i.e. because adding many elements at each level refinement).

Table II. Summary of the main progressive mesh compression algorithms.

Algorithm	Lossless connect. comp.	Total comp. rates (bpv)	Compress non-manifold meshes	Progr. granularity	Remarks
Progr. meshes [Hoppe 1996]	yes	37 (10 bit)	no	5/5	
Compressed progr. meshes [Pajarola and Rossignac 2000]	yes	22 (10 bit)	no	3/5	
Valence encoder [Alliez and Desbrun 2001a]	yes	21 (12 bit)	no	3/5	
Wavemesh [Valette and Prost 2004b]	yes	19 (12 bit)	no	1/5	Low number of levels of detail
Spectral compression [Karni and Gotsman 2000]	yes	19 (12 bit)	no	3/5	No progr. coding of the connect.
Kd-tree coder [Gandoin and Devillers 2002]	yes	19 (12 bit)	yes	2/5	High distortion at low rates
Octree coder [Peng and Kuo 2005]	yes	15 (12 bit)	yes	2/5	Like above
Incremental parametric refinement [Valette et al. 2009]	yes	15 (12 bit)	no	5/5	Original connect. may fail to be restored
Geometry image [Gu et al. 2002]	no	-	no	-	Can generate cracks in decomp. models
Wavelet compression [Khodakovsky et al. 2000]	no	8 (eq. 12 bit)	no	3/5	Fits well to smooth and dense meshes
Normal meshes [Guskov et al. 2000]	no	6 (eq. 12 bit)	no	3/5	Like above

Note: the figures in the parenthesis are the number of global quantization bits.

5. STATE OF THE ART ON RANDOM ACCESSIBLE MESH COMPRESSION

The issue with single-rate and progressive mesh compression algorithms is that, when the user wants to access a specific part of a very large mesh, the full model must be downloaded and decompressed. Random accessible algorithms allow to decompress only the required parts. Some algorithms give access to only the original level of detail of the mesh but others allow to decompress different parts at different levels of detail.

5.1. Random accessible compression

5.1.1. Cluster-based random accessible compression. The algorithm of Choe et al. [2004][2009] first segments the input mesh using Lloyd's method [1982]. Generated

charts are desired to be planar and compact in order to achieve high compression ratios. The important characteristic of this codec is that each chart is independently compressed using the single rate Angle Analyzer encoder [2002]. In order to not duplicate the geometry information of the border vertices shared by two charts, their positions are compressed independently in sequences called *wires*. The position of the next wire vertex to encode is predicted as a linear combination of the two previously encoded positions. The connectivity between the clusters is encoded under the form of a polygonal mesh with the encoding scheme proposed by Khodakovsky et al. [2002]. The Figure 12 illustrates this compression scheme.

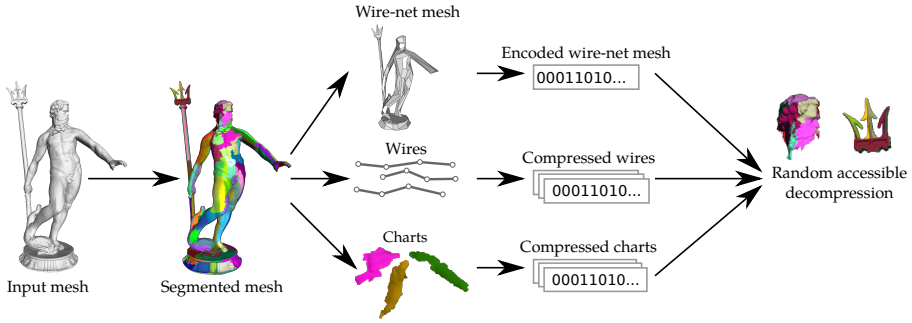

Fig. 12. Cluster-based random-accessible mesh compression [Choe et al. 2004][Choe et al. 2009].

Chen et al. [2008] used a segmentation algorithm that generates meaningful regions. Each cluster is then compressed with the Edgebreaker algorithm [1999]. The difference here is that the boundaries between the clusters are triangle strips and not wires. These strips are also compressed by the Edgebreaker algorithm but no geometry information is embedded in the border data. The vertex positions are encoded inside each cluster.

Yoon and Lindstrom [2007] also built a cluster-based random accessible compression scheme. They added the random-accessible support to streaming mesh compression [2005c]. The mesh is compressed by sequentially accessing and grouping the mesh triangles and vertices in a cache-oblivious layout. Each cluster is composed of a constant number of triangles (a few thousand) and are compressed independently with the streaming mesh compression scheme. The geometry information of the border vertices is not duplicated: it is encoded one time in a cluster and referenced for the others. This scheme comes with a mesh access programming interface that allows to access any element of the mesh by their identifier at a low computational cost. Experimental results report 45:1 speedup over standard streaming mesh compression. However, the compression overhead is about 40% compared to the approach of Choe et al. [2004].

The approach of Yoon and Lindstrom [2007] was recently extended to support geometry random access by compressing bounding volume hierarchies composed of axis-aligned bounding boxes [Kim et al. 2010]. In this hierarchy, a parent bounding volume is split into two children. The compression aims at preserving the cache coherency of the generated hierarchy. A set of bounding volume clusters that contains about four thousand bounding volumes is generated. These clusters are then compressed independently to enable the random access. The vertex indices of the triangles are encoded for each leaf bounding volume to enable the geometry random access to the mesh surface.

5.1.2. Hierarchical random accessible compression. Courbet and Hudelot [2009] proposed an alternative hierarchical representation based on sequences of vertices also called wires. The input mesh, which contains an already encoded exterior boundary, is split into two balanced partitions. The start and end vertices of the border wire between the two clusters as well as its size are encoded and form the connectivity information. The wire geometry is encoded with the same linear predictor as in [Choe et al. 2009]. Both partitions are then recursively split in the same way until each partition contains only one polygon. The decoding of one element is done by recursively splitting the wires until hitting an unsplittable polygon. This tree structure allows the decompression of only one of its paths. Therefore, the random accessibility granularity is high (see Figure 13). Besides, polygon meshes can be directly compressed. However, the compression efficiency for triangle meshes is inferior to the two previous approaches. The average cost is 3 bits per polygon for connectivity and 14 bpv for geometry using a 12 bit quantization.

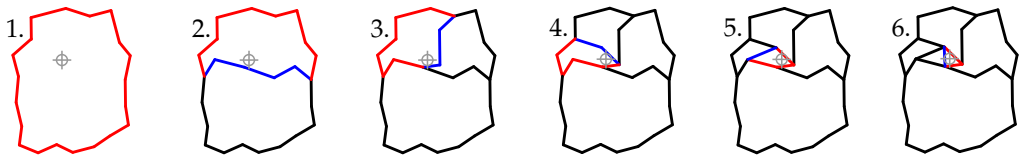

Fig. 13. Random-accessible hierarchical decompression of a triangle mesh [2009]. The target points the requested part of the mesh. For each step, the current decompressed wire is in blue. It forms with the red wire, the cluster that is split in the next step.

5.2. Progressive and random accessible compression

5.2.1. Connectivity-based algorithms. Kim et al. [2006] based their multiresolution random accessible mesh compression algorithm on their previous mesh refinement framework [Kim and Lee 2001]. During the decompression, a vertex can be split even if its neighbors are not the same than during the decimation. This breaks the symmetry of the operations of standard progressive mesh representation [Hoppe 1996]. Thus, on the decompressed model finely refined regions can be adjacent to coarse regions. The connectivity and the geometry are compressed through an efficient encoding of the vertex split hierarchy. This approach is therefore connectivity-based. It offers a fine-grained multiresolution random access but the compression performance is limited (about 31 bpv with a 12 bit quantization).

Cheng et al. [2007] described a progressive random accessible mesh compression algorithm based on an initial *meaningful* segmentation. The clustering is obtained by cutting the mesh into parts along concave feature contours. Each part of the mesh is then encoded with a modified version of the progressive encoder from [Alliez and Desbrun 2001a]. The positions of inserted vertices are encoded with polar coordinates and different quantizations for each component. No attention is, however, provided to the part boundaries. Similarly, Maglo et al. [2011] extended the single-rate random accessible coder of Choe et al. [2009] to progressively encode the charts. Chart border vertices are still encoded independently but methods are proposed to fill the holes between charts not fully decompressed.

The POMAR encoder [Maglo et al. 2013] however, does not require an initial segmentation of the input model. Discrete levels of detail are generated with halfedge collapses during the compression. The coarse levels are compressed globally as with a progressive algorithm. Nevertheless, for the fine levels, clusters are generated by dividing the vertex split hierarchy. By imposing a maximum difference of one level of

detail between adjacent clusters, this scheme generates smooth transitions between the coarse and fine decompressed regions and achieves an efficient compression performance (15 bpv with a 12 bit quantization).

5.2.2. Geometry-based algorithms. Better compression performances were obtained by geometry-based encoders. Jamin et al. [2009] and Du et al. [2009] proposed both to add random access support to the original Gandoin and Devilliers progressive algorithm [2002]. Both algorithms are out-of-core (see Section 3.3). During the decompression, different levels of detail can be selected for the different cells of the kd-tree. However, a post-processing step must handle the boundaries between cells decompressed at different levels of detail.

The *CHuMI viewer* from Jamin et al. [2009] partitions the mesh bounding box into a hierarchical structure called *nSP-tree*. This structure is composed of *SP-cells* encoded independently. Vertices that belong to several SP-cells are duplicated to allow independent decoding of each cell. Each SP-cell is subject to a Gandoin and Devilliers kd-tree decomposition [2002] that encodes the connectivity and geometry information as in the original progressive algorithm. The authors also bring a solution to the problem of blocky effect at low rates due to the low quantization. For this purpose, their scheme encodes some geometry bits in advance, in relation to the connectivity.

The approach of Du et al. [2009] decomposes the kd-tree of the original Gandoin and Devilliers algorithm [2002] into two layers. The top tree is the first layer while the set of its child subtrees is in the second layer. The top tree and each of the child sub-trees are compressed separately to enable the random access. During the decompression, the border vertices can be decompressed independently from the internal vertices. The subtrees must be compressed and decompressed in a predefined order because to decompress a subtree, the border vertices of the previous subtrees in the dependency list must be decompressed.

5.2.3. Connectivity-oblivious schemes. As for single-rate and progressive mesh compression algorithms, resorting to remeshing for progressive and random accessible compression offers a new degree of freedom that allows to achieve better compression rates. These schemes are often based on wavelet decomposition framework for semi-regular meshes as their progressive counterparts (see Section 4.2.1).

Sim et al. [2005] used the normal mesh representation [Guskov et al. 2000] in their scheme that assigns new edges to clusters after the butterfly subdivision. This framework also includes a distortion model, a visibility test method and a visibility priority method to view-dependently decompress the model. In [Roudet 2010], one level of the wavelet decomposition of an input semi-regular mesh is used to segment the model into regions of homogeneous coefficient magnitudes. Each of these regions are later projected on the input semi-regular model to be later encoded separately. The framework presented by Gioia et al. [2004] allows to dynamically add and remove wavelet coefficients to refine or coarse the model without having to decode the whole decomposition.

Wavelet schemes have a key advantage for progressive random accessible mesh compression. No complex algorithms have to be developed to access randomly the original connectivity graph. The difficulty comes indeed from the fact that the connectivity must be compressed into independent clusters. For wavelet schemes, the only requirement is that the wavelet decomposition of one cluster must only depend on the vertices of this cluster. The connectivity is only set by the base mesh and the subdivision scheme. It has not to be encoded for each level of detail.

5.3. Compact representation

Recently, researchers have proposed interesting data structures that support random access and constant time traversal operators while being very compact (approaching the storage of compressed formats). Gurung et al. [2011a] first proposed the Squad representation for triangle meshes; they combine triangles into quads and sort them that i^{th} quad is the one matched with the i^{th} vertex. This ordering allows them to store only a table of swings for the quad corners; vertex indices are not stored explicitly and are inferred at run time by examining a small set of candidates in the neighborhood. The storage cost for the connectivity is about 2 integer references per triangle. This representation was recently extended [Luffel et al. 2014] to support on-the-fly streaming construction and processing as in [Isenburg and Lindstrom 2005], mostly by interleaving geometry and connectivity and improving the coherency between vertices and triangles. Gurung et al. [2011b] proposed another compact data structure: the Laced Ring (LR) representation which needs only 1 reference per triangle. The principle is to reorder the vertices along a nearly-Hamiltonian cycle (called the ring). The same authors then introduced the improved Zipper approach [Gurung et al. 2013] able to store only 6 bits per triangle, mostly thanks to differential coding of the vertex indices and improved ring construction.

Table III summarizes most relevant approaches. *Random access granularity* (from 1 - lowest, to 5 - highest) refers, in a qualitative way, to the number of elements that must be decompressed to access a specific part of the mesh. For instance a low granularity is obtained when, for accessing a given vertex, it is necessary to decompress a large region around it.

Table III. Summary of the main random accessible (first part) and progressive random accessible (second part) mesh compression algorithms.

Algorithm	Lossless connect. comp.	Total comp. rates (bpv)	Compress non-manifold meshes	Random access granularity	Remarks
Streaming random accessible [Yoon and Lindstrom 2007]	yes	28 (12 bit)	no	2/5	Preserve the streaming layout
Hierarchical compression [Courbet and Hudelot 2009]	yes	20 (12 bit)	no	5/5	High random-accessibility granularity
Chart-based compression [Choe et al. 2009]	yes	16 (12 bit)	no	3/5	High compression performance
SQuad representation [Gurung et al. 2011a]	yes	2 references per triangle	no	5/5	First compact representation
Dependency free vertex splits [Kim et al. 2006]	yes	31 (12 bit)	no	5/5	Produce smooth transitions between mesh parts
Kd-tree cell compression [Jamin et al. 2009]	yes	21 (16 bit)	yes	3/5	Need post-processing to stich adjacent mesh parts
POMAR [Maglo et al. 2013]	yes	20 (16 bit)	no	3/5	Produce smooth transitions between mesh parts
Layered Kd-tree compression [Du et al. 2009]	yes	17 (16 bit)	yes	3/5	High distortion at low rates Dependency between parts
Normal mesh compression [Sim et al. 2005]	no	?	no	2/5	Fits well to smooth and dense meshes

Note: the figures in the parenthesis are the number of global quantization bits.

6. STATE OF THE ART ON MESH SEQUENCE COMPRESSION

Thanks to advances in the domain of camera-based 3D reconstruction, mesh sequences are becoming increasingly widespread. Using short time steps, these are extremely bulky and especially require efficient compression, as in the case of video compression, which exploits the temporal coherence of successive images. 3D animation is another domain which produces large quantities of mesh sequences. Almost all the techniques proposed so far by the scientific community only compress sequences with constant

connectivity (i.e. temporally coherent mesh sequence)(see Section 2). In the case of sequences with variable connectivity (i.e. temporally incoherent mesh sequence), the complexity of compression problems is considerably increased because spatio-temporal redundancies are much harder to exploit without prior mapping. To our knowledge only Han et al. [2007] and Yamasaki and Aizawa [2010] have tackled this problem; inspired by video compression techniques, the authors consider a division of meshes into blocks, then block matching and coding of the residuals.

Since most of the existing techniques concern temporally coherent mesh sequences, we focus this state of the art on the compression of this kind of sequences (i.e. dynamic meshes). We classify dynamic mesh compression methods into five categories: methods with prior segmentation, PCA-based methods, methods that consider spatio-temporal prediction, wavelets and finally the MPEG algorithms. Before starting we have to provide some insights on the useful functionalities of a dynamic mesh compression algorithm. We distinguish two main features that characterize existing algorithms:

- *Local vs Global:* As raised in [Amjoun and Straß er 2009], some algorithms analyze the global coherence of the dynamic mesh to compress it, while on the other side, others focus on local frame-to-frame changes. The main benefit of global approaches is an improved compression performance, while local algorithms allow for faster (often real-time) compression/decompression.
- *Scalability:* Dynamic meshes carry important amounts of data, hence the scalability is of importance for them. A scalable compression algorithm allows to decode only parts of the embedded bit stream, in order to display the sequence at a reduced frame rate (temporal scalability) or a reduced mesh resolution (spatial scalability). Scalable methods for dynamic meshes are counterparts of progressive methods for static meshes with the addition of the progressivity in the time dimension.

Scalable and local methods are of particular interest for scenarios of low-latency streaming, especially on hand-held devices. Global and non-scalable methods are usually limited to download-and-play scenarios.

6.1. Compression based on prior segmentation

Several authors have proposed to partition the vertices of the dynamic mesh into groups of points with similar movement. Lengyel [1999] proposed a method of this type in 1999 (the first dynamic mesh compression algorithm). The basic principle involves estimating the movement of a group of vertices by a rigid transformation, which then acts as a predictor. The segmentation information, the parameters of each transformation and the prediction errors are then coded to allow reconstruction of the sequence. Alignment between successive meshes can be used to improve results, as demonstrated by Gupta *et al.* [2002]. Sattler et al. [2005] proposed an improved segmentation based on vertex trajectories using Lloyd's clustering in combination with principal component analysis (PCA) and then compress each cluster using PCA (see section 6.2). Mamou et al. [2006] also consider a segmentation into almost rigid parts; the motion of each vertex is then described as a weighted linear combination of the cluster's motions (a skinning model) plus a motion compensation error compressed using discrete cosinus transform (DCT). This later approach makes possible a progressive transmission (i.e. spatial scalability) by ordering the DCT coefficients.

6.2. PCA-based methods

Alexa and Muller [2000] were the first authors to consider principal component analysis (PCA) for compression of dynamic mesh. Their idea is to represent each frame as a linear combination of principal component bases obtained through SVD decomposition of the matrix representation of the sequence ($3v \times f$ with v and f respectively the num-

ber of vertices and frames, 3 stands for the number of coordinates - x,y,z). The first base vector is the average shape and the others represent differences to this shape. Then by transmitting only a few principal bases instead of the whole set, very efficient compression ratio can be obtained. It's important to notice that before the decomposition, the frames have to be normalized using an affine transform that minimizes the distance of corresponding vertices. Their method was then improved by Karni and Gotsman [2004] who applied linear prediction coding to the PCA coefficients. These authors also introduced the KG distortion measure (basically the Frobenius norm of the matrix differences) used by many authors as a reference for rate distortion evaluation. Improved results were obtained by Sattler et al. [2005] who applied PCA on spatial clusters and Luo et al. [2013] who applied it on temporal clusters. Some authors have also proposed to apply PCA on the space of trajectories instead of shapes; the main benefit is that it involves the eigenvalues decomposition of a covariance matrix of $3f \times 3f$ instead of $3v \times 3v$. Vása and Skala have proposed several contributions based on trajectory space PCA. In [Váša and Skala 2007], they predict PCA coefficients with the parallelogram rules; they then improved this approach with an efficient mechanism to predict PCA basis [Váša and Skala 2009] ($+25\%$ gain), accurate new predictors [Váša and Skala 2010] ($+20\%$ gain) and traversal order optimization [Váša 2011] ($+15\%$ gain). This latter method provides currently the best results (between 0.5 and 5 bit per frame per vertex - bpfv - for a KG error of 0.05%). These PCA approaches are very efficient because they analyze the coherence of the entire sequence (i.e. they can be referred as global methods). Moreover they theoretically allow for spatial scalability by ordering the basis vectors, however this is not done in practice and they are only applicable in a download-and-play scenario.

6.3. Methods using spatio-temporal prediction

Some methods exploit the spatio-temporal redundancy of sequences by predicting the position of a vertex from surrounding spatial and/or temporal positions (by interpolation) or from previous positions (by extrapolation). On the contrary to global PCA algorithms, these methods exploit local coherences, hence they are usually computationally simpler. Ibarria et al. [2003] present two spatio-temporal predictors, the Extended Lorenzo Predictor (ELP) and REPLICA. The first extends the parallelogram rule taking into account the prediction of the position of the vertex at the previous time step. The REPLICA predictor makes the previous predictor robust to rigid transformations such as rotations and changes of scale, using local reference points and normalization. Zhang and Owen [2007] and Muller et al. [2006] place vertex displacement vectors into a hierarchical tree structure (an octree), which allows exploitation of spatio-temporal coherence. Amjoun and Strasser [2009] propose to use local coordinate systems (which separate tangential and normal components) to encode delta vectors. Stefanoski et al. [2007] propose to simplify the frames into spatial layers (i.e. levels of detail) and then exploit spatial and temporal dependencies between neighboring spatial layers and consecutive frames to estimate the motion of each vertex. Their method supports spatial scalability. More recently Stefanoski and Ostermann [2010] proposed their scalable predictive coding (SPC), which relies on spatial and temporal layer decomposition and performs the prediction in the space of rotation-invariant coordinates in order to compensate local rigid motion. Their scheme was the first to support spatio-temporal scalability and provides excellent compression ratio (between 3.5 and 8 pbfv). It was improved by Bici and Akar [2011] who proposed novel prediction structures. Finally, Ahn et al. [2013] still improved this multi-layer prediction to obtain a 30% gain in performance compared with SPC (and 20% again Bici and Akar's method). They obtain compression ratio between 2 and 6 bpfv.

6.4. Methods using wavelets

Some authors also proposed algorithms based on a wavelet representation. Guskov and Khodakovsky [2004] decompose the frames into layers using wavelet decomposition (spatial scalability) and propose a delta coding of the wavelet coefficients among adjacent frames. In a different way, Payan and Antonini [2007] consider temporal wavelet filtering (temporal scalability) combined with an efficient bit allocation process. Coding rates are between 5 and 12 bpfv.

6.5. The MPEG framework

In 2007 a standard for dynamic mesh compression was adopted by the Moving Picture Experts Group (MPEG), referred to as *Frame-based Animated Mesh Compression* (MPEG-4 FAMC) [Mamou et al. 2008]. It combines the skinning-based approach from Mamou et al. [2006] and the scalable approach from Stefanoski et al. [2007]. This approach provides very good performance and, more importantly, proposes three modes: downloadable (no scalability), scalable (both temporal and spatial) and finally streamable (the bit stream is partitioned into different packets encoded independently). The downloadable mode provides the best compression rates (between 2 and 7 pbfv).
Table IV summarize what we consider as most important approaches.

Table IV. Summary of the main dynamic mesh compression algorithms.

Algorithm	Comp. rates (bpfv)	Local/global	Scalability	Remarks
PCA [Alexa and Müller 2000]	-	Global	No	First spatial PCA method
Dynapack [Ibarria and Rossignac 2003]	-	Local	No	First real-time algorithm
PCA + Linear Prediction [Karni and Gotsman 2004]	-	Global	No	Very efficient for long sequences of relatively coarse meshes
Anisotropic Wavelet Transform [Guskov and Khodakovsky 2004]	-	Local	Spatial	First wavelet method
Scalable Predictive Coding [Stefanoski and Ostermann 2010]	3.5 to 8	Local	Spatial and temporal	Focus on low latency streaming
MPEG-4 FAMC [Mamou et al. 2006] [Stefanoski and Liu 2007] [Mamou et al. 2008]	2.5 to 8	Global	Spatial and temporal	Different modes (dowloadable/streamable /scalable)
Fine Granular Scalable [Ahn et al. 2013]	2 to 6	Local	Spatial and temporal	Focus on low latency streaming
CoDDyAC + Optimizations [Váša and Skala 2007; 2009] [Váša and Skala 2010; Váša 2011]	0.5 to 5	Global	No	Trajectory space PCA

Note: compression rates in bit per vertex per frame for a KG error equal to 0.05%.

7. CONCLUSION

Pioneering mesh compression approaches were single-rate methods that focused on connectivity compression. Some proposed schemes were based on triangle strips, followed by vertex spanning trees. While better compression rates were obtained by compressing face spanning trees, the best performances were obtained by valence-driven schemes, whose optimality was proven. Seeing that geometry size is often far more significant than connectivity size, the community started to become more interested in developing efficient geometry predictors and quantization methods.

Progressive mesh compression methods quickly appeared after the first single-rate approaches. Algorithms that do not restore initial connectivity (wavelet-based approaches, spectral compression, etc.) lead to a better rate-distortion performance. Among lossless progressive algorithms, geometry-driven approaches based on space

subdivision provide very efficient final compression performance but with a high distortion at low rates. Excellent results are obtained with algorithms based on reconstruction.

The interest in random accessible mesh compression came later with the desire to compress and access large meshes. Compared to single-rate and progressive mesh compression, few approaches have been proposed. For single-rate random accessible mesh compression, two paradigms were proposed: the segmentation and the hierarchical approaches. For progressive random accessible mesh compression, wavelet-based schemes have shown their high rate-distortion performance if connectivity can be modified. Connectivity-preserving schemes guided either by mesh connectivity or geometry have also been proposed.

Regarding dynamic mesh compression, initial methods have focused on pure compression rate using global schemes. The current trend is to consider local scalable methods suited to low-delay streaming.

We summarized in tables I, II, III and IV what we judged to be the main approaches. We see the future of mesh compression to be directed by the following two main points:

Evolution of the field

Until now, three independent fields of research have tackled the problems of 3D mesh data storage and access for efficient visualization:

(1) Pure compression methods (the main topic of this survey); they include single-rate and progressive algorithms. They are targeted for archival use and transmission.
(2) Compact and/or streamable data structures (briefly described in this survey). These approaches, e.g. the recent Squad and LR representations [Gurung et al. 2011a][Gurung et al. 2011b], allow mesh traversal and constant-time access to neighboring elements, while being more compact than classical data structures (e.g. Half-Edge).
(3) Hierarchical data structure for real-time rendering of massive datasets (not discussed in this paper, because usually no compression is involved), e.g. [Cignoni et al. 2004]. In this case, the goal is to propose smart hierarchical structures compliant with out-of-core construction, and optimized for parallel processing and rendering. Simplification and approximation methods could also be mentioned here [Krishnamurthy and Levoy 1996; Cohen-Steiner et al. 2004].

While each of these three fields of research has its own specific set of applications, we believe that the new challenges facing 3D data access and visualization will focus on the convergence of these approaches. This convergence is already starting to take place; for instance, very recent compact data structures are now getting close to the bitrate of pure compression approaches [Gurung et al. 2013] and/or support data parallel processing and streamed out-of-core access [Luffel et al. 2014]. Random access compression methods may also be seen as hybrid between these three fields of research. In particular, the recent random access progressive method from [Kim et al. 2010] presents a good compression ratio, allows fast random access, and is optimized for fast rendering (it is based on Bounding Volume Hierarchies, a structure optimized for ray tracing acceleration). More broadly, future compression methods will have to:

- *be aware of rendering*, since it is usually the ultimate goal of content transmission. An old yet excellent example is the work of Karni et al. [2002], who define their compression mechanism in order to optimize the rendering sequence on decompression (getting the benefits of the vertex cache of modern GPUs).
- *be compliant with out-of-core and parallel processing*, as modern computer architecture is evolving towards increasingly more cores. Recently, Meyer et al. [2012] described an innovative method for the GPU. They built a new mesh compression

scheme based on generalized triangle strips, and encoded in parallel with *scan-operations*. These kinds of smart ways to parallelize the compression process (which is intrinsically incremental) should be sought.
- *adapt to the context*. The use of 3D graphics is now spreading toward the general public. This implies an increasing heterogeneity of contexts and terminals used for 3D data visualization. For instance, with the emergence of technologies such as WebGL, the embedding of 3D meshes inside Web applications is exploding; the visualization of 3D data through handheld devices is also emerging. Consequently, compression methods have now to adapt to this heterogeneous context by presenting different settings. For instance, for mobile platforms and scripted Web environments, some of the compression features (complex prediction, entropy coding) could be disabled to decrease complexity and minimize use of the CPU on decompression. Some works compliant with Web and/or mobile platforms have been recently proposed in this sense (e.g. [Chun 2012][Gobbetti and Marton 2012][Lavoué et al. 2013][Limper et al. 2013]). Along the same lines, we can imagine a "random-access" setting that would disable the entropy coding to encode symbols on fixed-size records.
- *be able to handle generic 3D data* (arbitrary genus, surface or volume, manifold or not, with borders or not, etc.) with any kinds of attributes (texture coordinates, normals, colors). This would help the process of standardization already begun in the MPEG and X3D communities.

Perceptual metrics, a new paradigm for design and evaluation

Mesh compression approaches are usually compared by their rate distortion performance. To measure distortion, the traditional Root Mean Square or Hausdorff distances are often used. However, these measurements do not correlate with human visual perception. Since the final destination of a compressed model is often to be displayed to a human observer, the perceived visual quality should be the criterion to take into account when designing, configuring or evaluating a compression algorithm. Cohen-Steiner et al. proposed the $\mathcal{L}^{2,1}$ metric based on the normal [Cohen-Steiner et al. 2004] as our visual system is more sensitive to changes in normals rather than in changes in positions. To overcome the weakness of classical geometric measurements, mesh visual quality (MVQ) metrics have been recently introduced by the scientific community [Corsini et al. 2013]; their goal is to predict the perceived visual fidelity of distorted 3D data with respect to the original. Among existing works, Lavoué [2011] has proposed a multi-scale perceptual metric (MSDM2) based on mean curvature statistics, while Wang et al. [Wang et al. 2012] have considered the Laplacian of the Gaussian curvature for designing their FMPD metric. Subjective experiments with human observers demonstrated that these metrics can capture the perceived visibility of distortions better than RMS or Hausdorff distances. A perceptual metric was also recently proposed for dynamic meshes [Váša and Skala 2011].

Such visual quality assessment metrics provide a new paradigm for the evaluation, control and optimization of compression algorithms. For instance, as illustrated in [Lavoué 2011], comparison of the rate-distortion curves of progressive mesh compression algorithms computed with such perceptual metrics yields results different to those obtained with traditional metrics. Consequently, a comprehensive study might markedly change the currently admitted ranking of mesh compression algorithms.

More generally, as was the case for images and videos several years ago, 3D visual quality metrics should now constitute a new gold standard for designing and evaluating mesh compression algorithms. These new metrics, combined with recent efforts to better understand the perceptual mechanisms linked to 3D scene visualization (e.g.

visual making, contrast sensitivity function) should have a large impact on the design of future 3D mesh compression algorithms.

8. PUBLICLY AVAILABLE ALGORITHMS

Martin Isenburg, Pierre Alliez and Jack Snoeyink provided an on-line Java implementation and a downloadable standalone version of their single-rate compression algorithm [Touma and Gotsman 1998; Alliez and Desbrun 2001b; Isenburg 2002; Isenburg and Alliez 2002b; Khodakovsky et al. 2002] at `http://www.cs.unc.edu/\~isenburg/pmc/`. Source codes of the TFAN algorithm [Mamou et al. 2009] are available at `https://github.com/KhronosGroup/glTF/wiki/Open-3D-Graphics-Compression`.
Source codes and executables of the progressive algorithm from Lee et al. [2012] are available on the Mesh Processing Platform [Lavoué et al. 2012] (`http://liris.cnrs.fr/mepp/`). The executable of the Wavemesh algorithm from Valette et al. [2004b] is available at `http://www.creatis.insa-lyon.fr/site/fr/users/valette`. The source code of the algorithms from [Maglo et al. 2012] and [Maglo et al. 2013] can be downloaded at `http://magsoft.dinauz.org/sourceCode.php`. A CHuMI viewer [Jamin et al. 2009] executable is available at `http://clementjamin.free.fr/CHuMI/`. For dynamic mesh compression, the Coddyac algorithm from Váša et al. [2007] is available at `http://meshcompression.org/index.php/software-tools`. Besides these academic works, some open source compression software also exist, e.g. OpenCTM (`http://openctm.sourceforge.net/`).

REFERENCES

Jeong-Hwan Ahn, Chang-Su Kim, and Yo-Sung Ho. 2006. Predictive compression of geometry, color and normal data of 3-D mesh models. *IEEE Transactions on Circuits and Systems for Video Technology* 16, 2 (2006), 291–299.

Jae-kyun Ahn, Yeong Jun Koh, and Chang-su Kim. 2013. Efficient Fine-Granular Scalable Coding of 3D Mesh Sequences. *IEEE Transactions on Multimedia* 15, 3 (2013), 485–497.

Jae-Kyun Ahn, Dae-Youn Lee, Minsu Ahn, and Chang-Su Kim. 2011. R-D optimized progressive compression of 3D meshes using prioritized gate selection and curvature prediction. *The Visual Computer* 27, 6 (2011), 769–779.

Marc Alexa and Wolfgang Müller. 2000. Representing animations by principal components. *Computer Graphics Forum* 19 (2000), 411–418.

Pierre Alliez and Mathieu Desbrun. 2001a. Progressive compression for lossless transmission of triangle meshes. In *Proceedings of SIGGRAPH*. 195–202.

Pierre Alliez and Mathieu Desbrun. 2001b. Valence-Driven Connectivity Encoding for 3D Meshes. *Computer Graphics Forum* 20, 3 (2001), 480–489.

Pierre Alliez and Craig Gotsman. 2005. Recent Advances in Compression of 3D Meshes. In *Advances in Multiresolution for Geometric Modelling*. 3–26.

Rachida Amjoun and Wolfgang Straß er. 2009. Single-rate near lossless compression of animated geometry. *Computer-AidedDesign* 41, 10 (2009), 711–718.

Kostas Anagnostou, Tim J. Atherton, and Andrew E. Waterfall. 2000. 4D volume rendering with the Shear Warp factorisation. In *Volviz*. 129–137.

Romain Arcila, Cédric Cagniart, Franck Hétroy, Edmond Boyer, and Florent Dupont. 2012. Segmentation of temporal mesh sequences into rigidly moving components. *Graphical Models* 75, 1 (2012), 10–22.

Marco Attene, Bianca Falcidieno, Michela Spagnuolo, and Jarek Rossignac. 2003. SwingWrapper: Retiling triangle meshes for better edgebreaker compression. *ACM Transactions on Graphics* 22, 4 (2003), 982–996.

Chandrajit L. Bajaj, Insung Ihm, and Sanghun Park. 2001. 3D RGB image compression for interactive applications. *ACM Trans. Graph.* 20, 1 (2001), 10–38.

Chandrajit L. Bajaj, Valerio Pascucci, and Guozhong Zhuang. 1999. Single-resolution compression of arbitrary triangular meshes with properties. In *Proceedings of the Data Compression Conference*. 247–256.

Ulug Bayazit, Umut Konur, and Hasan F. Ates. 2010. 3-D Mesh Geometry Compression With Set Partitioning in the Spectral Domain. *IEEE Transactions on Circuits and Systems for Video Technology* 20, 2 (2010), 179–188.

Ulug Bayazit, Ozgur Orcay, Umut Konur, and Fikret S. Gurgen. 2007. Predictive vector quantization of 3-D mesh geometry by representation of vertices in local coordinate systems. *Journal of Visual Communication and Image Representation* 18, 4 (2007), 341–353.

Mirela Ben-Chen and Craig Gotsman. 2005. On the optimality of spectral compression of mesh data. *ACM Transactions on Graphics* 24, 1 (2005), 60–80.

M. Oguz Bici and Gozde B. Akar. 2011. Improved prediction methods for scalable predictive animated mesh compression. *Journal of Visual Communication and Image Representation* 22, 7 (Oct. 2011), 577–589.

U. Bischoff and Jarek Rossignac. 2005. TetStreamer: compressed back-to-front transmission of Delaunay tetrahedra meshes. In *Data Compression Conference, 2005. Proceedings.* 93–102.

Kangying Cai, Yu Jin, Wencheng Wang, QuQing Chen, Zhibo Chen, and Jun Teng. 2009. Compression of massive models by efficiently exploiting repeated patterns. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. 229–230.

François Cayre, Patrice Rondao-Alface, Francis Schmitt, Benoît Macq, and Henri Maître. 2003. Application of spectral decomposition to compression and watermarking of 3D triangle mesh geometry. *Signal Processing: Image Communication* 18, 4 (2003), 309 – 319.

Raphaëlle Chaine, Pierre-Marie Gandoin, and Céline Roudet. 2009. Reconstruction Algorithms as a Suitable Basis for Mesh Connectivity Compression. *IEEE Transactions on Automation Science and Engineering* 6, 3 (2009), 443–453.

Dan Chen, Yi-Jen Chiang, Nasir Memon, and Xiaolin Wu. 2005. Geometry Compression Of Tetrahedral Meshes Using Optimized Prediction. In *Proceedings of the European Conference on Signal Processing*.

Lijun Chen and Nicolas D. Georganas. 2008. Region-based 3D Mesh Compression Using an Efficient Neighborhood-based Segmentation. *Simulation* 84, 5 (2008), 185–195.

Ren Chen, Xiaonan Luo, and Hao Xu. 2008. Geometric compression of a quadrilateral mesh. *Computers & Mathematics with Applications* 56, 6 (2008), 1597 – 1603.

Zhi-Quan Cheng, Shi-Yao Jin, and Hua-Feng Liu. 2006. Anchors-based lossless compression of progressive triangle meshes. In *Proceedings of Pacific Graphics*. 45–50.

Zhi-Quan Cheng, Hua-Feng Liu, and Shi-Yao Jin. 2007. The progressive mesh compression based on meaningful segmentation. *The Visual Computer* 23, 9-11 (2007), 651–660.

Sungyul Choe, Junho Kim, Haeyoung Lee, and Seungyong Lee. 2009. Random Accessible Mesh Compression Using Mesh Chartification. *IEEE Transactions on Visualization and Computer Graphics* 15, 1 (2009), 160–173.

Sungyul Choe, Junho Kim, Haeyoung Lee, Seungyong Lee, and Hans-Peter Seidel. 2004. Mesh compression with random accessibility. In *Proceedings of the 5th Korea-Israel Bi-National Conference on Geometric Modeling and Computer Graphics*. 81–86.

Peter H. Chou and Teresa H. Meng. 2002. Vertex data compression through vector quantization. *IEEE Transactions on Visualization and Computer Graphics* 8, 4 (2002), 373–382.

Asma Chourou, Marc Antonini, and Amel Benazza-Benyahia. 2008. 3D mesh coding through region based segmentation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 1381 –1384.

Mike M. Chow. 1997. Optimized geometry compression for real-time rendering. In *Proceedings of Visualization*. 347–354.

Won Chun. 2012. WebGL Models: End-to-End. *OpenGL Insights* (2012), 431–454.

Paolo Cignoni, Fabio Ganovelli, Enrico Gobbetti, Fabio Marton, Federico Ponchio, and Roberto Scopigno. 2004. Adaptive tetrapuzzles: efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. In *Proceedings of SIGGRAPH*. 796–803.

P. Cignoni, C. Montani, and R. Scopigno. 1998. A comparison of mesh simplification algorithms. *Computers & Graphics* 22, 1 (1998), 37–54.

Tzi cker Chiueh, Chuan-Kai Yang, Taosong He, Hanspeter Pfister, and Arie E. Kaufman. 1997. Integrated volume compression and visualization.. In *IEEE Visualization*. 329–336.

Daniel Cohen-Or, David Levin, and Offir Remez. 1999. Progressive Compression of Arbitrary Triangular Meshes. In *Proceedings of the IEEE Visualization Conference*.

Daniel Cohen-Or, Cohen Rami, and Revital Irony. 2002. *Multi-way Geometry Encoding*. Technical report. The School of Computer Science, Tel-Aviv University.

D. Cohen-Steiner, P. Alliez, and M. Desbrun. 2004. Variational shape approximation.. In *ACM Siggraph*. 905–914.

Volker Coors and Jarek Rossignac. 2004. Delphi: geometry-based connectivity prediction in triangle mesh compression. *The Visual Computer* 20 (2004), 507–520. Issue 8-9.

M. Corsini, M. C. Larabi, G. Lavoué, O. Petřík, L. Váša, and K. Wang. 2013. Perceptual Metrics for Static and Dynamic Triangle Meshes. *Computer Graphics Forum* 32, 1 (Feb. 2013), 101–125.

Clément Courbet and Céline Hudelot. 2009. Random accessible hierarchical mesh compression for interactive visualization. In *Proceedings of the Symposium on Geometry Processing*. 1311–1318.

Clément Courbet and Céline Hudelot. 2011. Taylor Prediction for Mesh Geometry Compression. *Computer Graphics Forum* 30, 1 (2011), 139–151.

Clément Courbet and Martin Isenburg. 2010. Streaming compression of hexahedral meshes. *The Visual Computer* 26 (2010), 1113–1122. Issue 6-8.

Michael Deering. 1995. Geometry compression. In *Proceedings of SIGGRAPH*. 13–20.

Leon Denis, Shahid M. Satti, Adrian Munteanu, Jan Cornelis, and Peter Schelkens. 2010. Scalable Intraband and Composite Wavelet-Based Coding of Semiregular Meshes. *IEEE Transactions on Multimedia* 12, 8 (2010), 773–789.

Pablo Diaz-Gutierrez, M. Gopi, and Renato Pajarola. 2005. Hierarchyless Simplification, Stripification and Compression of Triangulated Two-Manifolds. *Computer Graphics Forum* 24, 3 (2005), 457–467.

Zhiyan Du, Pavel Jaromersky, Yi-Jen Chiang, and Nasir Memon. 2009. Out-of-Core Progressive Lossless Compression and Selective Decompression of Large Triangle Meshes. In *Proceedings of the Data Compression Conference*. 420 –429.

Nira Dyn, David Levine, and John A. Gregory. 1990. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics* 9, 2 (1990), 160–169.

James E. Fowler and Roni Yagel. 1995. Lossless Compression of Volume Data.. In *Symposium on Volume* (2002-12-18). 43–50.

Pierre-Marie Gandoin and Olivier Devillers. 2002. Progressive lossless compression of arbitrary simplicial complexes. In *Proceedings of SIGGRAPH*. 372–379.

Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH*. 209–216.

Patrick Gioia, Olivier Aubault, and Christian Bouville. 2004. Real-time reconstruction of wavelet-encoded meshes for view-dependent transmission and visualization. *IEEE Transactions on Circuits and Systems for Video Technology* 14, 7 (2004), 1009–1020.

Enrico Gobbetti and Fabio Marton. 2012. Adaptive quad patches: an adaptive regular structure for web distribution and adaptive rendering of 3D models. *Proceedings of the international conference on 3D Web technology* (2012).

Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. 2002. Geometry images. In *Proceedings of SIGGRAPH*. 355–361.

Stefan Gumhold. 2000. *New Bounds on the Encoding of Planar Triangulations*. Technical Report WSI-2000-1. University of Tübingen.

Stefan Gumhold. 2005. Optimizing markov models with applications to triangular connectivity coding. In *Proceedings of the annual ACM-SIAM symposium on Discrete algorithms*. 331–338.

Stefan Gumhold and Rachida Amjoun. 2003. Higher order prediction for geometry compression. In *Proceedings of Shape Modeling International, 2003*. 59–66.

Stefan Gumhold, Stefan Guthe, and Wolfgang Straßer. 1999. Tetrahedral mesh compression with the cut-border machine. In *Proceedings of the conference on Visualization (VIS '99)*. 51–58.

Stefan Gumhold and Wolfgang Straßer. 1998. Real time compression of triangle mesh connectivity. In *Proceedings of SIGGRAPH*. 133–140.

S. Gupta, K. Sengupta, and A. Kassim. 2002. Compression of dynamic 3D geometry data using iterative closest point algorithm. *Computer Vision and Image Understanding* 87, 1 (2002), 116–130.

Topraj Gurung, Daniel Laney, Peter Lindstrom, and Jarek Rossignac. 2011a. SQuad : Compact Representation for Triangle Meshes. *Computer graphic forum* 30, 2 (2011).

Topraj Gurung, M Luffel, and Peter Lindstrom. 2011b. LR: compact connectivity representation for triangle meshes. *ACM Transactions on Graphics* 30, 4 (2011).

Topraj Gurung, Mark Luffel, and Peter Lindstrom. 2013. Zipper : A compact connectivity data structure for triangle meshes. *Computer-Aided Design* 45, 2 (2013), 262–269.

Igor Guskov and Andrei Khodakovsky. 2004. Wavelet Compression of Parametrically Coherent Mesh Sequences. In *Symposium on Computer Animation*. 183–192.

Igor Guskov, Kiril Vidimče, Wim Sweldens, and Peter Schröder. 2000. Normal meshes. In *Proceedings of SIGGRAPH*. 95–102.

Stefan Guthe and Wolfgang Straßer. 2001. Real-Time Decompression and Visualization of Animated Volume Data. In *IEEE Visualization*.

Seung-Ryong Han, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2007. Time-Varying Mesh Compression Using an Extended Block Matching Algorithm. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 11 (2007), 1506–1518.

Jeffrey Ho, Kuang Chih Lee, and David Kriegman. 2001. Compressing large polygonal models. In *Proceedings of the conference on Visualization '01 (VIS '01)*. 357–362.

Hugues Hoppe. 1996. Progressive meshes. In *Proceedings of SIGGRAPH*. 99–108.

Hugues Hoppe and Emil Praun. 2005. Shape Compression using Spherical Geometry Images. In *Advances in Multiresolution for Geometric Modelling*. 27–46.

David Huffman. 1952. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE* 40, 9 (1952), 1098–1101.

Lawrence Ibarria, Peter Lindstrom, Jarek Rossignac, and Andrzej Szymczak. 2003. Out-of-core Compression and Decompression of Large n-dimensional Scalar Fields. *Comput. Graph. Forum* 22, 3 (2003), 343–348.

Lorenzo Ibarria and Jarek Rossignac. 2003. Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity. In *Symposium on Computer Animation*. 126–135.

Martin Isenburg. 2002. Compressing Polygon Mesh Connectivity with Degree Duality Prediction. In *Graphics Interface Conference Proceedings*.

Martin Isenburg and Pierre Alliez. 2002a. Compressing hexahedral volume meshes. In *Proceedings of Pacific Graphics*. 284–293.

Martin Isenburg and Pierre Alliez. 2002b. Compressing polygon mesh geometry with parallelogram prediction. In *Proceedings of the conference on Visualization '02 (VIS '02)*. 141–146.

Martin Isenburg and Stefan Gumhold. 2003. Out-of-core compression for gigantic polygon meshes. In *Proceedings of SIGGRAPH*. 935–942.

Martin Isenburg, Ioannis Ivrissimtzis, Stefan Gumhold, and Hans-Peter Seidel. 2005a. Geometry prediction for high degree polygons. In *Proceedings of the Spring Conference on Computer Graphics*. 147–152.

Martin Isenburg and Peter Lindstrom. 2005. Streaming meshes. *Proceedings of Visualization* (2005), 231–238.

Martin Isenburg, Peter Lindstrom, Stefan Gumhold, and Jonathan Shewchuk. 2006. Streaming compression of tetrahedral volume meshes. In *Proceedings of Graphics Interface*. 115–121.

Martin Isenburg, Peter Lindstrom, and Jack Snoeyink. 2005b. Lossless compression of predicted floating-point geometry. *Computer-Aided Design* 37, 8 (2005), 869–877.

Martin Isenburg, Peter Lindstrom, and Jack Snoeyink. 2005c. Streaming Compression of Triangle Meshes. In *Proceedings of the Eurographics Symposium on Geometry Processing*. 111–118.

Martin Isenburg and Jack Snoeyink. 2000a. Face fixer: compressing polygon meshes with properties. In *Proceedings of SIGGRAPH*. 263–270.

Martin Isenburg and Jack Snoeyink. 2000b. Spirale reversi: Reverse decoding of EdgeBreaker encoding. In *Proceedings of the 12th Canadian Conference on Computational Geometry*.

Clément Jamin, Pierre-Marie Gandoin, and Samir Akkouche. 2009. CHuMI viewer: Compressive huge mesh interactive viewer. *Computers & Graphics* 33, 4 (2009), 542 – 553.

Aymen Kammoun, Frédéric Payan, and Marc Antonini. 2012. Sparsity-based optimization of two lifting-based wavelet transforms for semi-regular mesh compression. *Computers & Graphics* 36, 4 (2012), 272–282.

Zachi Karni, Alexander Bogomjakov, and Craig Gotsman. 2002. Efficient compression and rendering of multi-resolution meshes. In *Proceedings of the conference on Visualization*. 347–354.

Zachi Karni and Craig Gotsman. 2000. Spectral compression of mesh geometry. In *Proceedings of SIGGRAPH*. 279–286.

Zachi Karni and Craig Gotsman. 2001. 3D mesh compression using fixed spectral bases. In *Proceedings of Graphics interface*. 1–8.

Zachi Karni and Craig Gotsman. 2004. Compression of Soft-Body animation sequences. *Computers & Graphics* 28, 1 (2004), 25–34.

Andrei Khodakovsky, Pierre Alliez, Mathieu Desbrun, and Peter Schröder. 2002. Near-optimal connectivity encoding of 2-manifold polygon meshes. *Graphical Models* 64 (2002), 147–168. Issue 3-4.

Andrei Khodakovsky and Igor Guskov. 2003. Compression of Normal Meshes. In *Geometric Modeling For Scientific Visualization*. 189–206.

Andrei Khodakovsky, Peter Schröder, and Wim Sweldens. 2000. Progressive geometry compression. In *Proceedings of SIGGRAPH*. 271–278.

Junho Kim, Sungyul Choe, and Seungyong Lee. 2006. Multiresolution Random Accessible Mesh Compression. *Computer Graphics Forum* 25, 3 (2006), 323–331.

Junho Kim and Seungyong Lee. 2001. Truly selective refinement of progressive meshes. In *Proceedings of Graphics interface*. 101–110.

Junho Kim, Changwoo Nam, and Sungyul Choe. 2011. Bayesian AD coder: Mesh-aware valence coding for multiresolution meshes. *Computers & Graphics* 35, 3 (2011), 713–718.

Tae-Joon Kim, Bochang Moon, Duksu Kim, and Sung-Eui Yoon. 2010. RACBVHs: Random-Accessible Compressed Bounding Volume Hierarchies. *IEEE Transactions on Visualization and Computer Graphics* 16, 2 (2010), 273–286.

Davis King and Jarek Rossignac. 1999. Guaranteed 3.67 V bit Encoding of Planar Triangle Graphs. In *Proceedings of the 11th Canadian Conference on Computational Geometry*.

Davis King, Andrzej Szymczak, and Jaroslaw R. Rossignac. 1999. Connectivity Compression for Irregular Quadrilateral Meshes. *GVU Technical Report GIT-GVU-99-36* (1999).

Felix Kälberer, Konrad Polthier, Ulrich Reitebuch, and Max Wardetzky. 2005. FreeLence - Coding with Free Valences. *Computer Graphics Forum* 24, 3 (2005), 469–478.

V. Krishnamurthy and M. Levoy. 1996. Fitting smooth surfaces to dense polygon meshes.. In *ACM Siggraph*. 313–324.

Sebastian Krivograd, Mladen Trlep, and Borut Žalik. 2008. A hexahedral mesh connectivity compression with vertex degrees. *Computer-Aided Design* 40, 12 (2008), 1105–1112.

Boris Kronrod and Craig Gotsman. 2000. Efficient Coding of Non-Triangular Mesh Connectivity. In *Proceedings of the 8th Pacific Conference on Computer Graphics and Applications*. 235.

B. Kronrod and C. Gotsman. 2002. Optimized compression of triangle mesh geometry using prediction trees. In *Proceedings of the International Symposium on 3D Data Processing Visualization and Transmission*. 602–608.

Guillaume Lavoué. 2011. A Multiscale Metric for 3D Mesh Visual Quality Assessment. *Computer Graphics Forum* 30, 5 (2011), 1427–1437.

Guillaume Lavoué, Laurent Chevalier, and Florent Dupont. 2013. Streaming compressed 3D data on the web using JavaScript and WebGL. In *Web3D '13: Proceedings of the 18th International Conference on 3D Web Technology*. 19–27.

Guillaume Lavoué, Martial Tola, and Florent Dupont. 2012. MEPP - 3D Mesh Processing Platform. In *International Conference on Computer Graphics Theory and Applications*.

Sridhar Lavu, Hyeokho Choi, and Richard Baraniuk. 2003. Geometry compression of normal meshes using rate-distortion algorithms. In *Proceedings of the Symposium on Geometry processing*. 52–61.

Aaron W. F. Lee, Wim Sweldens, Peter Schröder, Lawrence Cowsar, and David Dobkin. 1998. MAPS: multiresolution adaptive parameterization of surfaces. In *Proceedings of SIGGRAPH*. 95–104.

Dae-Youn Lee, Jae-Kyun Ahn, Minsu Ahn, J.D.K. Kim, Changyeong Kim, and Chang-Su Kim. 2011. 3D mesh compression based on dual-ring prediction and MMSE prediction. In *Proceedings of the IEEE International Conference on Image Processing*. 905–908.

Dae-Youn Lee, Sanghoon Sull, and Chang-Su Kim. 2013. Progressive 3D mesh compression using MOG-based Bayesian entropy coding and gradual prediction. *The Visual Computer* (2013), 1–15.

Eung-Seok Lee and Hyeong-Seok Ko. 2000. Vertex data compression for triangular meshes. In *Proceedings of Pacific Graphics*. 225 –234.

Haeyoung Lee, Pierre Alliez, and Mathieu Desbrun. 2002. Angle-Analyzer: A Triangle-Quad Mesh Codec. *Computer Graphics Forum* 21, 3 (2002), 383–392.

Ho Lee, Guillaume Lavoué, and Florent Dupont. 2012. Rate-distortion optimization for progressive compression of 3D mesh with color attributes. *The Visual Computer* 28 (2012), 137–153. Issue 2.

Haeyoung Lee and Sujin Park. 2005. Adaptive Vertex Chasing for the Lossless Geometry Coding of 3D Meshes. In *Advances in Multimedia Information Processing - PCM 2005*. Lecture Notes in Computer Science, Vol. 3767. 108–119.

Jerome Edward Lengyel. 1999. Compression of time-dependent geometry. In *Symposium on Interactive 3D graphics (I3D '99)*. ACM, New York, NY, USA, 89–95.

Thomas Lewiner, Marcos Craizer, Hélio Lopes, Sinésio Pesco, Luiz Velho, and Esdras Medeiros. 2006. GEncode: Geometry-driven compression for General Meshes. *Computer Graphics Forum* 25, 4 (2006), 685–695.

Jiankun Li and C.-C.J. Kuo. 1998a. A dual graph approach to 3D triangular mesh compression. In *Proceedings of the International Conference on Image Processing*, Vol. 2. 891–894.

Jiankun Li and C.-C. Jay Kuo. 1998b. Progressive coding of 3-D graphic models. In *Proceedings of the IEEE*, Vol. 86.

Junlin Li, Dihong Tian, and Ghassan AlRegib. 2006. Vector Quantization in Multiresolution Mesh Compression. *IEEE Signal Processing Letters* 13, 10 (2006), 616–619.

Max Limper, Stefan Wagner, Christian Stein, Yvonne Jung, and André Stork. 2013. Fast delivery of 3D web content: a case study. In *Proceedings of the International Conference on 3D Web Technology*. 11–17.

Peter Lindstrom and Martin Isenburg. 2008. Lossless Compression of Hexahedral Meshes. In *Proceedings of the Data Compression Conference*. 192–201.

Stuart P. Lloyd. 1982. Least squares quantization in PCM. *IEEE Transactions on Information Theory* 28, 2 (1982), 129–137.

Charles Loop. 1987. *Smooth Subdivision Surfaces Based on Triangles*. Master's thesis. University of Utah, Department of Mathematics.

Hélio Lopes, Geovan Tavares, Jarek Rossignac, Andrzej Szymczak, and Alla Safanova. 2002. Edgebreaker: A Simple Compression for Surfaces with Handles. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications (SMA '02)*. 289–296.

Zhe-Ming Lu and Zhen Li. 2008. Dynamically restricted codebook-based vector quantisation scheme for mesh geometry compression. *Signal, Image and Video Processing* 2 (2008), 251–260. Issue 3.

Mark Luffel, Topraj Gurung, Peter Lindstrom, and Jarek Rossignac. 2014. Grouper : A Compact , Streamable Triangle Mesh Data Structure. *IEEE Transactions on Visualization and Computer Graphics* 20, 1 (2014), 84–98.

Eric B. Lum, Kwan-Liu Ma, and John Clyne. 2002. A Hardware-Assisted Scalable Solution for Interactive Volume Rendering of Time-Varying Data. *IEEE Trans. Vis. Comput. Graph.* 8, 3 (2002), 286–301.

Guoliang Luo, Frederic Cordier, and Hyewon Seo. 2013. Compression of 3D mesh sequences by temporal segmentation. *Computer Animation and Virtual World* (2013), 365–375.

Xiaonan Luo and Guifeng Zheng. 2008. Progressive meshes transmission over a wired-to-wireless network. *Wireless Networks* 14 (2008), 47–53. Issue 1.

JianPing Ma, Qiang Chen, Bo Chen, and Hong Wang. 2009. Mobile 3D graphics compression for progressive transmission over wireless network. In *Proceedings of the IEEE International Conference on Computer-Aided Design and Computer Graphics*. 357–362.

Adrien Maglo, Clément Courbet, Pierre Alliez, and Céline Hudelot. 2012. Progressive compression of manifold polygon meshes. *Computers & Graphics* 36, 5 (2012), 349–359.

Adrien Maglo, Ian Grimstead, and Céline Hudelot. 2011. Cluster-based Random Accessible and Progressive Lossless Compression of Colored Triangular Meshes for Interactive Visualization.. In *Proceedings of Computer Graphics International*.

Adrien Maglo, Ian Grimstead, and Céline Hudelot. 2013. POMAR: Compression of progressive oriented meshes accessible randomly. *Computers & Graphics* 37, 6 (2013), 743–752.

Sridhar Mahadevan. 2007. Adaptive mesh compression in 3D computer graphics using multiscale manifold learning. In *Proceedings of the international conference on Machine learning*. 585–592.

Khaled Mamou, Christophe Dehais, Faten Chaieb, and Faouzi Ghorbel. 2010. Shape approximation for efficient progressive mesh compression. In *Proceedings of the IEEE International Conference on Image Processing*.

Khaled Mamou, Titus Zaharia, Françoise Prêteux, Fourier Evry, and Francoise Preteux. 2006. A Skinning Approach for Dynamic 3D Mesh Compression. *Computer Animation and Virtual Worlds* 17, 3-4 (2006), 337–346.

Khaled Mamou, Titus Zaharia, and Françoise Prêteux. 2009. TFAN: A low complexity 3D mesh compression algorithm. *Computer Animation and Virtual Worlds* 20, 2-3 (2009), 343–354.

Khaled Mamou, Titus Zaharia, Françoise Prêteux, Nikolče Stefanoski, and Jörn Ostermann. 2008. Frame-based compression of animated meshes in MPEG-4. *International Conference on Multimedia and Expo* (2008).

Shaoliang Meng, Aili Wang, and Shengming Li. 2010. Compression of 3D triangle meshes based on predictive vector quantization. In *International Symposium on Systems and Control in Aeronautics and Astronautics*. 1403–1406.

Quirin Meyer, Benjamin Keinert, Gerd Sußner, and Marc Stamminger. 2012. Data-Parallel Decompression of Triangle Mesh Topology. *Computer Graphics Forum* 31, 8 (2012), 2541–2553.

Karsten Müller, Aljoscha Smolic, Matthias Kautzner, Peter Eisert, and Thomas Wiegand. 2006. Rate-distortion-optimized predictive compression of dynamic 3D mesh sequences. *Signal Processing: Image Communication* 21, 9 (Oct. 2006), 812–828.

Shigeru Muraki. 1992. Approximation and Rendering of Volume Data Using Wavelet Transforms. In *IEEE Visualization*. 21–28.

Paul Ning and Lambertus Hesselink. 1992. Vector Quantization for Volume Rendering. In *VVS*. 69–74.

Tilo Ochotta and Dietmar Saupe. 2008. Image-Based Surface Compression. *Computer Graphics Forum* 27, 6 (2008), 1647–1663.

Renato Pajarola and Jarek Rossignac. 2000. Compressed Progressive Meshes. *IEEE Transactions on Visualization and Computer Graphics* 6 (2000), 79–93. Issue 1.

Renato Pajarola, Jarek Rossignac, and Andrzej Szymczak. 1999. Implant sprays: compression of progressive tetrahedral mesh connectivity. In *Proceedings of the conference on Visualization*. 299–305.

Frédéric Payan and Marc Antonini. 2002. 3D Mesh Wavelet Coding Using Efficient Model-based Bit Allocation. In *Proceedings of the First International Symposium on 3D Data Processing Visualization and Transmission*. 391–394.

Frédéric Payan and Marc Antonini. 2005. An efficient bit allocation for compressing normal meshes with an error-driven quantization. *Computer Aided Geometric Design* 22 (2005), 466–486. Issue 5.

Frédéric Payan and Marc Antonini. 2006. Mean square error approximation for wavelet-based semiregular mesh compression. *IEEE Transactions on Visualization and Computer Graphics* 12, 4 (2006), 649–657.

Frédéric Payan and Marc Antonini. 2007. Temporal wavelet-based compression for 3D animated models. *Computers & Graphics* 31, 1 (Jan. 2007), 77–88.

Jingliang Peng, Yan Huang, C.-C. Jay Kuo, Ilya Eckstein, and M. Gopi. 2010. Feature Oriented Progressive Lossless Mesh Coding. *Computer Graphics Forum* 29, 7 (2010), 2029–2038.

Jingliang Peng, Chang-Su Kim, and C.-C. Jay Kuo. 2005. Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation* 16, 6 (2005), 688–733.

Jingliang Peng and C.-C. Jay Kuo. 2005. Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. In *Proceedings of SIGGRAPH*. 609–616.

Gabriel Peyré and Stéphane Mallat. 2005. Surface compression with geometric bandelets. In *Proceedings of SIGGRAPH*. 601–608.

Jovan Popović and Hugues Hoppe. 1997. Progressive simplicial complexes. In *Proceedings of SIGGRAPH*. 217–224.

Sylvain Prat, Patrick Gioia, Yves Bertrand, and Daniel Meneveaux. 2005. Connectivity compression in an arbitrary dimension. *The Visual Computer* 21 (2005), 876–885. Issue 8-10.

Jorma Rissanen and Glen G. Langdon. 1979. Arithmetic Coding. *IBM Journal of Research and Development* 23, 2 (1979), 149–162.

Flemming Friche Rodler. 1999. Wavelet Based 3D Compression with Fast Random Access for Very Large Volume Data. In *Pacific Conference on Computer Graphics and Applications*. 108–117.

Jarek Rossignac. 1999. Edgebreaker: Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics* 5 (January 1999), 47–61. Issue 1.

Jarek Rossignac and Andrzej Szymczak. 1999. Wrap&Zip decompression of the connectivity of triangle meshes compressed with Edgebreaker. *Computational Geometry* 14, 1–3 (1999), 119–135.

Céline Roudet. 2010. A Region-Based Progressive Coding of Semi-Regular 3-D Meshes for View-Dependent Transmission. *Int. Conf. on Signal Image Technology and Internet Based Systems* (2010), 51–59.

Pedro V. Sander, Zoë J. Wood, Steven J. Gortler, John M. Snyder, and Hugues Hoppe. 2003. Multi-chart geometry images. In *Proceedings of Symposium on Geometry Processing*. 146–155.

Mirko Sattler and Ralf Sarlette. 2005. Simple and efficient compression of animation sequences. In *ACM SIGGRAPH/Eurographics Symp. Comput. Animat.* 209–217.

Yunhui Shi, Bo Wen, Wenpeng Ding, Na Qi, and Baocai Yin. 2012. Realistic mesh compression based on geometry image. In *Picture Coding Symposium (PCS), 2012*. 133 –136.

Dinesh Shikhare, Sushil Bhakar, and Sudhir P. Mudur. 2001. Compression of Large 3D Engineering Models using Automatic Discovery of Repeating Geometric Features. In *Proceedings of the Vision Modeling and Visualization Conference*. 233–240.

Jae-Young Sim, Chang-Su Kim, C.-C.J. Kuo, and Sang-Uk Lee. 2005. Rate-distortion optimized compression and view-dependent transmission of 3-D normal meshes. *IEEE Transactions on Circuits and Systems for Video Technology* 15, 7 (2005), 854–868.

Jae-Young Sim, Chang-Su Kim, and Sang-Uk Lee. 2003. An efficient 3D mesh compression technique based on triangle fan structure. *Signal Processing: Image Communication* 18, 1 (2003), 17–32.

Nikolče Stefanoski and Xiaoliang Liu. 2007. Scalable linear predictive coding of time-consistent 3d mesh sequences. *3DTV Conference* (2007), 9–12.

Nikolče Stefanoski and Jörn Ostermann. 2010. SPC: Fast and Efficient Scalable Predictive Coding of Animated Meshes. *Computer Graphics Forum* 29, 1 (March 2010), 101–116.

Andrzej Szymczak. 2003. Optimized Edgebreaker encoding for large and regular triangle meshes. *The Visual Computer* 19, 4 (2003), 271–278.

Andrzej Szymczak, Davis King, and Jarek Rossignac. 2001. An Edgebreaker-based efficient compression scheme for regular meshes. *Computational Geometry* 20, 12 (2001), 53–68.

Andrzej Szymczak and Jarek Rossignac. 1999. Grow & fold: compression of tetrahedral meshes. In *Proceedings of the fifth ACM symposium on Solid modeling and applications*. 54–64.

Andrzej Szymczak, Jarek Rossignac, and Davis King. 2002. Piecewise Regular Meshes: Construction and Compression. *Graphical Models* 64, 3–4 (2002), 183–198.

Gabriel Taubin, André Guéziec, William Horn, and Francis Lazarus. 1998. Progressive forest split compression. In *Proceedings of SIGGRAPH*. 123–132.

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

Jiang Tian, Wenfei Jiang, Tao Luo, Kangying Cai, Jingliang Peng, and Wencheng Wang. 2012. Adaptive coding of generic 3D triangular meshes based on octree decomposition. *The Visual Computer* 28 (2012), 819–827. Issue 6-8.

Costa Touma and Craig Gotsman. 1998. Triangle Mesh Compression. In *Graphics Interface 98 Conference Proceedings*. 26–34.

György Turán. 1984. On the succinct representation of graphs. *Discrete Applied Mathematics* 8, 3 (1984), 289–294.

William Tutte. 1962. A census of planar triangulations. *Canadian Journal of Mathematics* 14 (1962), 21–38.

William Tutte. 1963. A census of planar maps. *Canadian Journal of Mathematics* 15 (1963), 249–271.

Shyh-Kuang Ueng. 2003. Out-of-core encoding of large tetrahedral meshes. In *Proceedings of the Eurographics/IEEE TVCG Workshop on Volume graphics*. 95–102.

Sébastien Valette, Raphaëlle Chaine, and Rémy Prost. 2009. Progressive lossless mesh compression via incremental parametric refinement. In *Proceedings of the Symposium on Geometry Processing*. 1301–1310.

Sébastien Valette and Rémy Prost. 2004a. Wavelet-based multiresolution analysis of irregular surface meshes. *IEEE Transactions on Visualization and Computer Graphics* 10, 2 (2004), 113–122.

Sébastien Valette and Rémy Prost. 2004b. Wavelet-Based Progressive Compression Scheme for Triangle Meshes: Wavemesh. *IEEE Transactions on Visualization and Computer Graphics* 10 (2004), 123–129. Issue 2.

Libor Váša. 2011. Optimised mesh traversal for dynamic mesh compression. *Graphical Models* 73, 5 (Sept. 2011), 218–230.

Libor Váša and Guido Brunnett. 2013. Exploiting connectivity to improve the tangential part of geometry prediction. *IEEE Transactions on Visualization and Computer Graphics* (2013).

Libor Váša and Václav Skala. 2007. CoDDyAC: Connectivity Driven Dynamic Mesh Compression. In *3DTV Conference*.

Libor Váša and Václav Skala. 2009. COBRA: Compression of the Basis for PCA Represented Animations. *Computer Graphics Forum* 28, 6 (Sept. 2009), 1529–1540.

Libor Váša and Václav Skala. 2010. Geometry-Driven Local Neighbourhood Based Predictors for Dynamic Mesh Compression. *Computer Graphics Forum* 29, 6 (Sept. 2010), 1921–1933.

Libor Váša and Vaclav Skala. 2011. A Perception Correlated Comparison Method for Dynamic Meshes. *IEEE Transactions on Visualization and Computer Graphics* 17, 2 (2011), 220–230.

Kai Wang, Fakhri Torkhani, and Annick Montanvert. 2012. A Fast Roughness-Based Approach to the Assessment of 3D Mesh Visual Quality. *Computers & Graphics* (2012).

Toshihiko Yamasaki and Kiyoharu Aizawa. 2010. Patch-based compression for time-varying meshes. *International Conference on Image Processing* (2010), 3433–3436.

Boon-Lock Yeo and Bede Liu. 1995. Volume Rendering of DCT-Based Compressed 3D Scalar Data. *IEEE Trans. Vis. Comput. Graph.* 1, 1 (1995), 29–43.

Liu Ying, Dai Mingli, Han Zhongming, and Duan Dagao. 2010. An Edgebreaker & code-mode based connectivity compression for triangular meshes. In *Proceedings of the International Conference on Advanced Computer Control*, Vol. 2. 96–101.

Sung-Eui Yoon and P. Lindstrom. 2007. Random-Accessible Compressed Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1536–1543.

Jinghua Zhang and Charles B Owen. 2007. Octree-based animated geometry compression. *Computers and Graphics* 31, 3 (2007), 463–479.

Chong Zhao, Hanqiu Sun, and Kaihuai Qin. 2011. Efficient wavelet-based geometry compression. *Computer Animation and Virtual Worlds* 22, 2-3 (2011), 307–315.