Progressive Compression of Manifold Polygon Meshes

Adrien Maglo, Clement Courbet, Pierre Alliez, Céline Hudelot

Progressive Compression of Manifold Polygon Meshes

Adrien Maglo[a], Clément Courbet[a], Pierre Alliez[b], Céline Hudelot[a]

[a]*MAS laboratory, Ecole Centrale Paris, France*
[b]*INRIA Sophia Antipolis - Méditerranée, France*

Abstract

This paper presents a new algorithm for the progressive compression of manifold polygon meshes. The input surface is decimated by several traversals that generate successive levels of detail through a specific patch decimation operator which combines vertex removal and local remeshing. The mesh connectivity is encoded by two lists of Boolean error predictions based on the mesh geometry: 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 and a local curvature prediction. We also include two methods that improve the rate-distortion performance: a wavelet formulation with a lifting scheme and an adaptive quantization technique. Experimental results demonstrate the effectiveness of our approach in terms of compression rates and rate-distortion performance.

1. Introduction

Surface meshes are of common use in a range of application domains such as computer-aided design, simulation, medical imaging, digital heritage and entertainment. The increasing needs for high precision models lead to the generation of complex meshes which must be stored and transmitted over heterogeneous networks. As data storage has a cost and network bandwidths do not grow as fast as the size of these data, solutions must be found to reduce the size of these models through mesh compression.

Among the mesh compression algorithms, the progressive ones allow during decompression to first obtain a coarse version of the mesh. This first level of details (LOD) is then progressively refined as more data is decompressed, until the input mesh is restored. The general goal is to achieve the best rate-distortion (R-D) performance in the sense that each LOD decoded must be as close as possible to the original mesh with the minimum amount of transmitted data.

Previous work on progressive mesh compression has focused on the compression of triangle surface meshes. However, a significant number of carefully designed meshes are composed of polygon faces. In addition, many recent work focused on quad mesh processing. Moreover, for applications such as remote scientific visualization, meshes can contain not only triangular faces and the decompression must restore the initial connectivity. While some approaches have been proposed

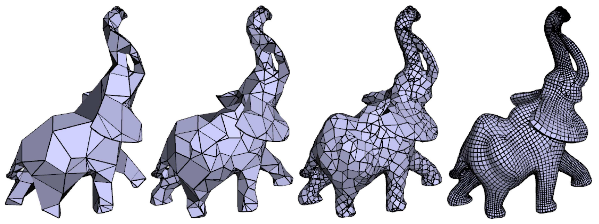
Figure 1: Levels of details of the quadrangle elephant model generated by our compression algorithm.

for single-rate compression [1, 2, 3, 4, 5] or random-accessible compression [6], to our knowledge no approaches were proposed for their progressive compression.

Our main contributions are summarized as follows:

1. We first propose a simple progressive mesh compression algorithm that compresses any 2-manifold mesh with arbitrary face degrees.
2. From this base algorithm, we describe a curvature prediction method and a connectivity prediction scheme to further reduce the size of geometry and connectivity.
3. We also include two complementary methods that improve the R-D performance. The first method consists in a wavelet formulation of the geometry compression. It contains a lifting step that slightly improves the R-D performance without increasing the final compression ratio. The second method is the adaptive quantization algorithm from [7].

It further improves the R-D performance but increases the final compression rate.

2. Previous work on progressive mesh compression

2.1. Connectivity-based

Hoppe introduced the concept of *progressive meshes* (PM) [8]. The idea is to incrementally decimate a mesh using the edge collapse operator. The compressed representation consists of the base mesh followed by all parameters required for the incremental reverse operations, called *vertex splits*. The main advantage of this scheme is its high multi-resolution 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 bits per vertex (bpv) with 10 bits quantization. Popović and Hoppe in [9] generalized the PM representation to arbitrary simplicial complexes. They introduced the *generalized vertex split* and its inverse, the *vertex unification* operations. With this representation, a model requires about 50bpv with 10 bits quantization.

In order to come closer to compression rates of single rates methods, some methods were proposed to encode the vertex split operations in batches. Taubin et al. [10] build a progressive mesh compression scheme inspired by the single-rate *topological surgery* algorithm. Their *progressive forest split* representation encodes a manifold triangular mesh with a base mesh and a sequence of *forest split* operations. The forest split operation consists in cutting the mesh through several sets of connected edges, filling the generated holes with triangles and relocating the vertices. Pajarola and Rossignac [11] improved the compression rates by imposing some restrictions for choosing the candidates to the edge collapse operations: the operations are grouped into batches during the traversal of a spanning tree. They also improve the geometry coding by using a butterfly predictor. Karni et al. [12] designed a progressive compression scheme which enables the fast rendering of all the LODs. The first step of their 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 30bpv [10] and 22bpv [11] for 10 bits quantization). Nevertheless the multiresolution granularity is impacted compared to the PM representation.

Other progressive compression schemes use vertex removals instead of edge collapses. Li and Kuo [13] 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 neighborhood 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 LOD. Cohen-Or et al. [14] used the same decimation mechanism combined with patch coloring to encode the face locations with respect to each patch. They achieved compression rates competitive with single rate techniques (about 23bpv with 10 bits quantization). Alliez and Desbrun [15] proposed what could be seen as a progressive version of the Touma-Gotsman single-rate encoder [16]. At each iteration, the mesh is decimated by two deterministic patch traversals, the connectivity being encoded through the valence of the removed vertices. The geometry is encoded through the patch barycentric error prediction in a local Frenet frame. The obtained compression rates are about 13bpv with 10 bits quantization.

Valette et al. build a progressive mesh compression algorithm through a wavelet framework [17]. The initial mesh is progressively decimated with a subdivision scheme tailored to irregular meshes. The connectivity data is composed of all face subdivision operations. The geometry is encoded through a wavelet lifting scheme. The compression rates are slightly better than those from [15] (about 19bpv with 12 bits quantization).

2.2. Geometry-based

Observing that the compressed size of geometry is generally higher than the one of connectivity, Gandoin and Devillers [18] designed a mesh compression algorithm driven by the geometry. In their scheme, the vertex positions are stored in a kD-tree and the vertex occurrences in each cell are entropy coded. The connectivity is encoded using vertex splits. Peng and Kuo also developed a geometry-driven progressive mesh compression algorithm [19] based on an octree data structure. This coder predicts the connectivity of the mesh from the neighbor vertices geometry during the vertex splits thanks to pivot vertices. This algorithm compresses triangle meshes with about 15bpv with 12 bits quantization.

2.3. Improving the Rate-Distortion trade-off

Optimizing the R-D trade-off has been the main focus of recent research on progressive mesh compression. Lee et al. [7] showed that the rate-distortion trade-off of the Alliez-Desbrun (AD) coder [15] is improved by using an adaptive quantization method. The idea consists in choosing between performing a decimation operation or a global quantization operation. They obtain both better R-D performances and compression rates (about 1bpv improvement with 12 bits quantization). Ahn et al. [20] proposed another improvement for the AD coder through an optimized 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 2.4) 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 R-D performance. They significantly improve the compression rates of the AD coder (about 4bpv reduction with 12 bits quantization).

Valette et al. cast the progressive mesh compression problem as a mesh generation problem [21]. The algorithm starts from a coarse version of the initial mesh that is progressively refined using a Delaunay mesh generation approach. When all vertices of the original mesh have been decoded, the initial connectivity is restored by flipping edges. This algorithm is shown to compress efficiently (about 15bpv with 12 bits quantization) and provides good rate-distortion performances. The complete connectivity restoration process is however not guaranteed to succeed. The idea of computing the best decimated version of an initial mesh has been recently further investigated [22]. The algorithm starts from the initial mesh vertex set and recursively splits it into several child subsets. Each time a new vertex subset is generated, a representative vertex of this set is computed. In this hierarchy, the number of children of a set is entropy coded. The offsets between a representative and its parent representative are quantized and entropy encoded. The hierarchy is encoded with the connectivity information in a specific order to achieve the best R-D trade-off. This algorithm yields compression rates at about 16bpv with 12 bits quantization.

2.4. *Laplacian operator-based*

Other schemes propose to compress the vertex positions using the frequency domain through the use of the mesh Laplacian operator. Karni and Gotsman [23] compress the geometry of a mesh by computing its spectrum with the eigenvector decomposition of its Laplacian matrix. The spectral coefficients, after being quantized and entropy coded, are sufficient to decompress a good approximation of the initial mesh. The compression, albeit lossy, achieves excellent R-D performance with few coefficients. Mamou et al. [24] devised an algorithm that computes the Laplacian matrix of a mesh. The mesh is then approximated with a heat equation and a minimal set of control points. The vertex locations are encoded as residuals from the approximation. The connectivity is encoded by the TG single-rate encoder [16]. This scheme achieves an excellent compression ratio (about 10bpv with 12 bits quantization) despite a high complexity due to the time spent at solving the heat equation.

2.5. *Wavelet-based*

When the restoration of the initial mesh connectivity is not crucial, some authors resort to semi-regular remeshing. Khodakovsky et al. [25] propose an algorithm that remeshes the initial mesh, then compresses it using a wavelet transform based on the Loop filter and a zero-tree coder. This scheme was later improved [26] through a normal mesh representation. Payan and Antonini [27, 28] allocate the bits across the wavelet subbands for the standard and normal mesh representations. In general, the wavelet-based algorithms incorporating a remeshing step provide better compression ratio than pure lossless algorithms.

2.6. *Handling polygon meshes*

An indirect approach to deal with polygon meshes consists of first triangulating the polygon mesh before resorting to an existing method restricted to triangle meshes. This approach was already proposed by Taubin et al. [10] to extend the *progressive forest split* algorithm. While being simple at first glance, this approach is not when caring about restoring the initial connectivity of the finer level. It requires encoding an extra information to remove the edges added during triangulation, and conceptually adds more connectivity by adding more edges than necessary. The theoretical result of Tutte's entropy [29] states that the entropy of a planar graph is expressed in bits per edge. This result and our experiments (see Section 6.1) confirm the intuition that adding extra edges increases entropy and hence that the size of the compressed triangulated mesh is in general superior to that of the compressed original mesh. Li et al. state also that their compression scheme [13] can be used on polygonal meshes. However, they provide no details about how the algorithm would be adapted. It can also be noticed that the experimental results presented in [10] and [13] do not compare favorably with recent state of the art techniques. In recent work [19], Peng and Kuo discuss the progressive compression of polygon meshes by an octree coder. As their algorithm can compress arbitrary connectivity between vertices, it is possible to modify the face construction algorithm to reconstruct polygon faces. The mesh connectivity is encoded through vertex splits and efficient prediction of pivots vertices. By definition, pivot vertices are connected to the two vertices of the edge generated by a vertex split, but there is no pivot vertex when the adjacent faces of this edge are not triangles. This encoding scheme is therefore optimized to compress triangle meshes and is not best suited to polygon meshes.

Our algorithm compresses progressively and effectively manifold polygonal meshes with arbitrary face degrees while still be competitive with previous approaches specialized to triangle mesh.

3. Base algorithm

3.1. Compression

The proposed algorithm is based on mesh decimation. It is composed of four main procedures that are successively repeated until the initial mesh M^n cannot be further simplified.

- The *decimation* step consists in applying the *patch decimation* operator to generate the mesh LOD M^{l-1} from M^l. This operator removes vertices and adds new edges to the mesh. M^l connectivity is transformed into face and edge Boolean flags. Face flags indicate if a face has a removed center vertex. Edge flags indicate if an edge was inserted by a remeshing operation. The prediction residuals of the positions of the removed vertices are also stored for later encoding.
- The *patch encoding* step builds the face symbol list S_f and the residual list S_r from the face flags and residuals. It consists of a deterministic face conquest of the mesh to encode the faces with a removed center vertex and the position of the removed vertices.
- The *edge encoding* step builds the edge symbol list S_e from the edge flags. A deterministic edge conquest encodes which edges have or have not been inserted.
- The *entropy coding* step compresses the S_f, S_r and S_e symbol lists.

3.1.1. Decimation step

The *decimation* step tries to simplify as much as possible a mesh LOD M^l to generate M^{l-1} using the *patch decimation* operator. A patch is a set of faces with a common center vertex. The algorithm first attempts to form a patch on the mesh. The degree of its center vertex v_j must greater than two (see Figure 3(a)). If it succeeds, it then creates a polygon f_i around v_j. To do so, it splits every faces around v_j that is not a triangle by adding a new edge between the two vertices of this face that are connected to v_j. We call this remeshing operation *re-edging* (see Figure 3(b)). This operation aims at generating faces with low degree. The inserted edges, are marked. v_j is removed (see Figure 3(c)) and the residual

$$\mathbf{r}_{f_i} = \mathbf{p}_{v_j} - \mathbf{b}_{f_i}, \tag{1}$$

where $\mathbf{p}_{v_j}$ is the position of v_j and $\mathbf{b}_{f_i}$ is the barycenter of f_i vertices, is stored. The barycenter of the vertices of a face f_i is simply defined as:

$$\mathbf{b}_{f_i} = \frac{1}{|V_{f_i}|} \sum_{v_k \in V_{f_i}} \mathbf{p}_{v_k},$$

where V_{f_i} is the set of the f_i vertices. $\mathbf{r}_{f_i}$ represents the geometry data. f_i becomes a face of the mesh. It is marked as having a removed center vertex and can no longer be implied in a *patch decimation* operation in the current *decimation* step.

The *patch decimation* operation by definition always removes one vertex v_j from the mesh. If v_j has t incident triangles, the variation of the number of faces in the mesh caused by the operation is equal to $1 - t$. This means that, if v_j is only incident to non-triangle faces, $t = 0$ and the number of faces is incremented by one, while the degree of the neighbor faces is decremented by one (see Figure 3). Other *patch decimations* on neighbor vertices may make these faces progressively disappear. If v_j is only incident to triangle faces, then no re-edging is needed and the number of faces in the mesh decreases (see Figure 4).
If v_j is a degree d vertex, f_i generation can be seen as the merging of d triangular faces, as achieved by *superface* creation in [30]. But in this previous work the process is driven by the mesh geometry. In our algorithm, the merging is driven by the connectivity because it must ensure for the decompression that all f_i vertices were connected to v_j. Besides, *re-edging* does not triangulate all mesh faces since it only happens where *patch decimations* are proceeded.

Patch decimation operations must preserve the manifold property of the mesh, so that if a patch border vertex shares an edge with more than 2 other patch border vertices, the patch is not decimated. The generated faces are not necessarily planar. This is not a problem since non-planar faces can be good local approximations. Concave faces however may be problematic

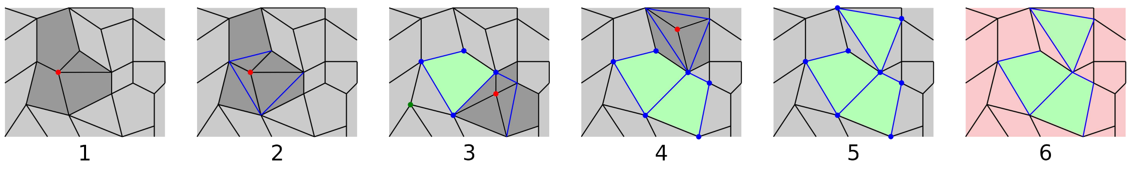

Figure 2: (a) One example *decimation* step. 1. A seed vertex (in red) is chosen to form the dark gray patch. 2. The non-triangular faces of the patch are split by inserting edges (in blue) between their two vertices connected to the center vertex. 3, 4, 5. The patch center vertex is removed. The face generated by the vertex removal is marked as having a center vertex removed (in green). Its vertices are marked as visited (in blue). All its unvisited adjacent vertices are added to a FIFO queue. A new patch center vertex (red) is popped out of the FIFO. It may not be possible to build a valid patch around this vertex (green) because, for example, its decimation would not preserve the manifold property. In this case, its unvisited adjacent vertices are added to the FIFO and an other vertex is popped. 6. No more patches can be decimated. The current *decimation* step is finished. The results are: a set of faces with a center vertex removed (green), a set of faces without a removed center vertex (pink), a set of inserted edges (blue) and a set of original edges (black).

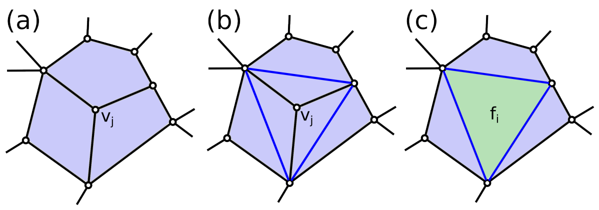

Figure 3: Decimation of a patch with non-triangle faces. (a) The active patch is in blue. (b) The faces are split by the re-edging operation with the inserted blue edges in order to create a new polygon around v_j. (c) v_j is removed. f_i is marked as having a center vertex removed.

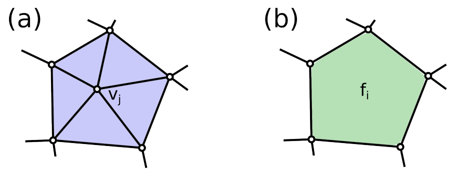

Figure 4: Decimation of a patch with triangle faces. (a) The active patch is in blue. (b) v_j is removed. f_i is marked as having a center vertex removed.

to render and may lead to further deteriorations during decimation. In our scheme, a face is said concave if its edges, projected on a plane directed by the face normal, form a concave polygon. A triangle face is by definition always convex. The normal of a non-planar face is computed with Newell's method [31]. If the generation of concave faces is not allowed, the first *decimation* steps generate only convex faces. When it is no longer possible, the next *decimation* steps further simplify the mesh by allowing the generation of concave faces. These last unconstrained decimations allow reducing the size of the base mesh that is not compressed in our current implementation. During the decompression, the first LOD displayed to the user is the first that contains only convex faces. The algorithm can also skip the decimation of important vertices to minimize the distortion through a volume-based metric as in [15].

Once the current *patch decimation* is over, the algorithm attempts to create other patches with unmarked faces of the mesh. The patch decimation order is not constrained. In our current implementation, the algorithm starts from a random seed vertex and progressively conquers the whole mesh by trying to generate new patches with adjacent vertices that do not belong to already marked faces. An example is depicted in Figure 2.

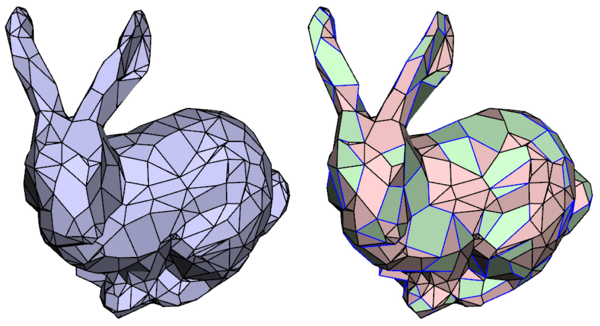

Figure 5: Decimation of an intermediate level of detail of the bunny model. Left: the initial level of detail. Right: the new level of detail after decimation. The inserted edges are depicted in blue. Faces with a removed vertex are depicted in green.

A new mesh LOD M^{l-1} is obtained when no more *patch decimation* operations can be performed. The result of the *decimation* step is the simplified level M^{l-1} with a set of marked inserted edges and a set of marked faces with a removed center vertex (see Figure 5). About 30% of the vertices of a LOD are removed during a *decimation step*.

The decimation algorithm stops when no more *decimation* steps can be performed. The lowest LOD M^0 is called the *base mesh*. Its size depends on the complexity of the mesh but is generally less than 1% of the total file

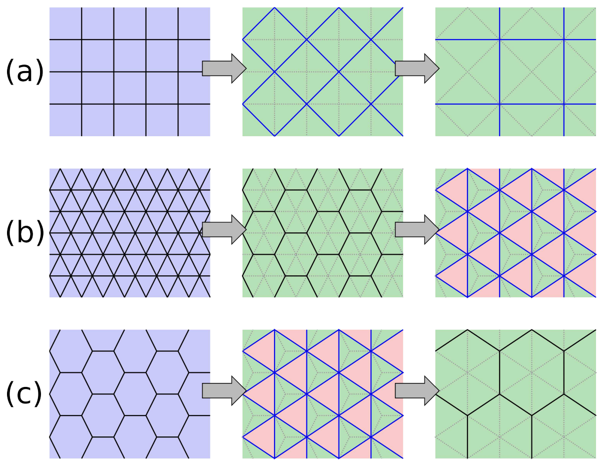

Figure 6: Examples of two successive decimations of regular connectivities: a regular grid (a), a regular triangle mesh (b) and a regular hexagonal mesh (c).

size. As shown by Figure 6, the *decimation* step preserves the regularity of the infinite regular structure it decimates. In practice however the regularity gets progressively worse during decimation, as the meshes are not specifically designed to preserve regularity during decimation.

3.1.2. Patch encoding step

The *patch encoding* step produces the list S_f of Boolean face symbols s_{f_i} and the list S_r of the residuals r_{f_i}. s_{f_i} codes if f_i has a removed vertex or not. The algorithm uses a gate FIFO queue to perform a deterministic conquest of the mesh. A gate is an edge between a conquered and an unconquered face. The first gate of the *patch encoding* step is specified for the whole mesh compression and hence can not be removed. When a face f_i is conquered, its symbol s_{f_i} is added to S_f. If s_{f_i} is *true*, we also add the projection of r_{f_i} in the local Frenet frame to S_r (see Figure 7). To avoid the post-quantization step mentioned in [15] and slightly reduce the entropy, we use the bijection proposed in [32] to transform the coordinates. The gates of f_i are then added to the queue in the counterclockwise order starting from the current gate. The next conquered face is the one pointed by the next gate in the queue. The same conquest order is followed by the decoder during the decompression. An example of a *patch encoding* step is depicted in Figure 8 (a).

3.1.3. Edge encoding step

To build the edge symbols list S_e, a deterministic full conquest of the mesh edges is performed using an edge

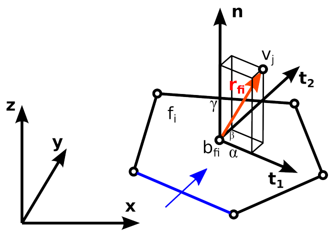

Figure 7: Encoding of the geometry residual $\mathbf{r}_{f_i}$ in the local Frenet frame $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{n})$. $\mathbf{t}_1$ takes the direction of the gate edge (in blue). $\mathbf{n}$ is the patch normal. And $\mathbf{t}_2 = -\mathbf{t}_1 \wedge \mathbf{n}$.

FIFO queue. When an unconquered edge e_k is encountered, a symbol s_{e_k} is generated to code if e_k was inserted. Note that if e_k belongs to two faces that do not have a removed center vertex, it is inevitably original. Therefore, no symbol is generated in this case. The neighboring edges of one e_k vertex are then added to the queue. The next edge to conquer is extracted from the queue. An example of an *edge encoding* step is given by Figure 8 (b).

3.1.4. Entropy encoding

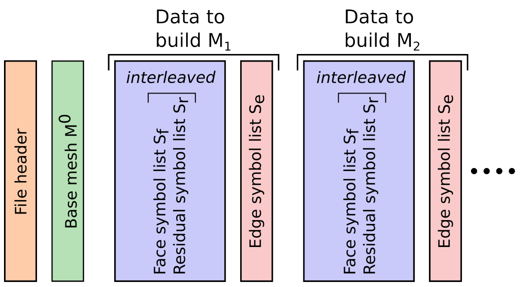

Figure 9: Structure of the compressed data generated by our coder.

The previously described face and edge binary symbol lists S_f and S_e represent the connectivity data. The simplification process, depending on the LOD, may lead to biased binary distributions of their symbols (see Figure 11). Therefore, an entropy coder with one adaptive context per list is used to encode S_f and S_e.
The geometry data, the S_r list, is also entropy coded with two adaptive contexts: one for the tangential components of $\mathbf{r}_{f_i}$ and one for its normal component. In our current implementation, we used the range encoder from Schindler [33]. Figure 9 depicts the structure of the compressed data generated by our coder.

3.2. Decompression

The mesh decompression starts by reconstructing M^0. Then, by applying the reverse operations of the

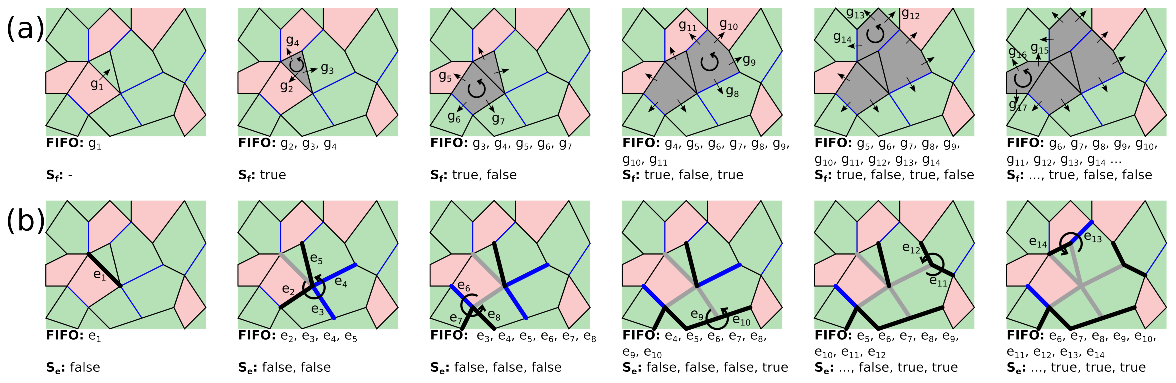
(a)
g1
g2 g3 g4
g5 g6 g7
g8 g9 g10 g11
g12 g13 g14
g15 g16 g17
FIFO: g1
Sf: -
FIFO: g2, g3, g4
Sf: true
FIFO: g3, g4, g5, g6, g7
Sf: true, false
FIFO: g4, g5, g6, g7, g8, g9, g10, g11
Sf: true, false, true
FIFO: g5, g6, g7, g8, g9, g10, g11, g12, g13, g14
Sf: true, false, true, false
FIFO: g6, g7, g8, g9, g10, g11, g12, g13, g14 ...
Sf: ..., true, false, false
(b)
e1
e2 e3 e4 e5
e6 e7 e8
e9 e10
e11 e12
e13 e14
FIFO: e1
Se: false
FIFO: e2, e3, e4, e5
Se: false, false
FIFO: e3, e4, e5, e6, e7, e8
Se: false, false, false
FIFO: e4, e5, e6, e7, e8, e9, e10
Se: false, false, false, true
FIFO: e5, e6, e7, e8, e9, e10, e11, e12
Se: ..., false, true, true
FIFO: e6, e7, e8, e9, e10, e11, e12, e13, e14
Se: ..., true, true, true

Figure 8: (a) Example of a patch encoding conquest. Faces with, resp. without, a removed center vertex are depicted in green, resp. pink. Conquered faces are depicted in gray. The arrows represent the gates. (b) Example of an edge encoding conquest. The inserted edges are depicted in blue. The edges in the FIFO queue are depicted in bold. The conquered edges are depicted in gray. During decompression the same conquest orders are followed to determine the faces with a removed center vertex and the inserted edges.

decimation step, the successive LOD M^l are progressively restored to finally obtain M^n. Thus, for each LOD M^l, a first full conquest is performed to decode the faces with a removed center vertex. These vertices are then inserted at their original positions from the encoded geometry data. A second full conquest decodes and removes the edges that were not present in M^l.

4. Improving connectivity and geometry encoding

The scheme described above allows compressing and decompressing any 2-manifold polygon mesh. We describe next improvements to further reduce the size of the connectivity and geometry. They make our scheme competitive in term of compression ratio with triangle specialized approaches. The typical improvement is 0.3bpv for geometry and 0.7bpv for connectivity.

4.1. Predicting connectivity from geometry

As depicted in Figure 10, after one decimation conquest, the average area of the faces with a removed center vertex is greater than the average area of the other faces. This observation allows predicting connectivity from geometry. The prediction algorithm works as follows. During the patch encoding and decoding conquests, the average area of the two types of faces are progressively updated when new faces are conquered. The predicted type of a new conquered face is the one which has the closest average area value to the current face area. A binary symbol is generated to indicate if the prediction is verified. This symbol replace s_{f_i} and is also later entropy coded. The binary distribution of this new symbol is for most cases more biased than the simple coding distribution.

For some connectivity however, such as the decimation of a regular hexagonal mesh (see Figure 6(c)), the average area of the two types of faces can be equal. In this case, the type of a face can be predicted from the type of its neighbors. When the difference between the face type pourcentages is below a threshold (experimentally set at 10%), our algorithm predicts the type of a face as the inverse of the type that is the most represented among the adjacent already conquered faces.

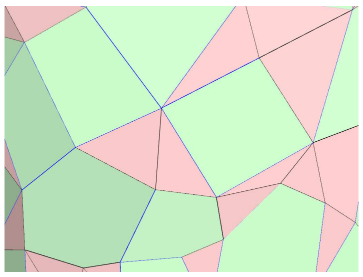

Figure 10: Result of a decimation conquest. The average surface of the face with a removed center vertex (green) is larger than the average surface of the other faces (red). The average length of the inserted edges (blue) is also superior to the average length of the others (black).

A similar algorithm is used for the prediction of the edge symbols s_{e_k}. If the polygons are well-shaped, then the inserted edges are in general longer than the original edges due to the *re-edging* process.

To predict the group a face or an edge belongs to, we use average values of face areas and edge lengths computed on the part of the mesh already conquered. We make the assumption that the mesh is uniformly sampled. When the current LOD regularity is bad or the last *decimation* step removed few vertices, the simple coding scheme is more effective. Therefore, at the beginning of each connectivity *entropy encoding* step, a binary symbol is generated to indicate if the connectivity prediction algorithm is used or not.

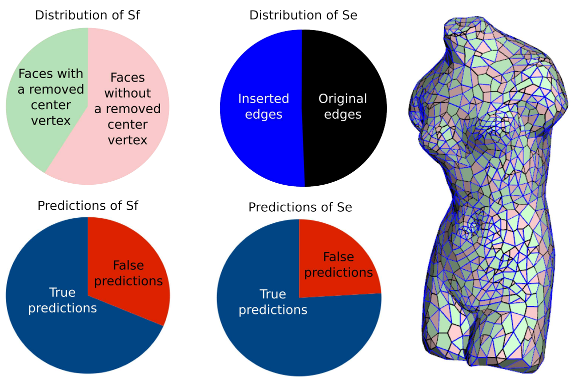

Figure 11: Example of connectivity symbol distributions after one *decimation* step shown on the right mesh. The first row shows the distribution of S_f and S_e. The second row shows the distributions of the prediction symbols.

4.2. Curvature prediction for geometry encoding

As described in Section 3.1.1 the geometry data is composed of the removed vertex residuals $\mathbf{r}_{f_i}$. To improve the geometry data encoding, we use a curvature prediction method. Instead of directly encoding the residual $\mathbf{r}_{f_i}$, we encode:

$$\mathbf{g}_{f_i} = \mathbf{r}_{f_i} - \alpha \mathbf{l}_{f_i}, \tag{2}$$

where $\mathbf{l}_{f_i}$ is the average Laplacian of the set V_{f_i} of the vertices of f_i. We define the Laplacian of a vertex v_j as:

$$\mathbf{l}_{v_j} = \frac{1}{|V_{v_j}|} \sum_{v_k \in V_{v_j}} \mathbf{p}_{v_k} - \mathbf{p}_{v_j},$$

with V_{v_j} the set of neighbor vertices of v_j. Therefore we have:

$$\mathbf{l}_{f_i} = \frac{1}{|V_{f_i}|} \sum_{v_k \in V_{f_i}} \mathbf{l}_{v_k}.$$

In our experiments, we set $\alpha = 0.5$. For most meshes, $\mathbf{g}_{f_i}$ has a more biased distribution than the one of $\mathbf{r}_{f_i}$ and hence can be more effectively entropy coded.

5. Improving the Rate-Distortion

We now describe the inclusion to our codec of two methods that improve its rate-distortion performance. The first is based on a wavelet decomposition with a *lifting step*. It improves the R-D ratios at low rates by about 25% without impacting the final compression rate. The second is the adaptive global quantization method taken from [7]. It improves the R-D ratios at low rates by about 35% but increases the final compression rate by 1 to 2 bpv.

5.1. Wavelet formulation of the geometry compression

We formulate here the mesh geometry compression as a wavelet decomposition using the lifting scheme [34]. The idea of using a wavelet decomposition for the geometry compression of irregular meshes is not new [17]. However the wavelet decomposition we present is specific to our method as we use different mesh decimation operators and geometry encoding schemes.

When a new level of details is generated, two types of geometry data are computed during the wavelet decomposition:

$$C^{l-1} = A^l . C^l, \tag{3}$$

$$D^{l-1} = B^l . C^l. \tag{4}$$

C^{l-1} is the $m \times 3$ global matrix of the coarse coefficients, the m vertex coordinates of M^{l-1}. D^{l-1} is the $p \times 3$ global matrix of the detail coefficients, the p local $\mathbf{r}_{f_i}$ values that are encoded. The analysis filters A^l and B^l are defined by the decimation operations completed to generate M^{l-1}. A^l is the matrix that extracts the vertex positions of M^{l-1} from the vertex positions of M^l. B^l is the matrix that computes all the details coefficients, as locally defined in (1) (see Figure 12 (a)).

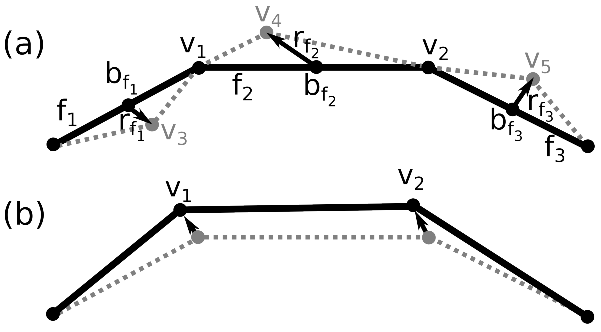

Figure 12: Transversal views of a mesh during the compression with the lifting scheme enabled. (a) The mesh before decimation is depicted by gray dotted lines. The mesh after decimation is depicted by black continuous lines. The detail coefficients $\mathbf{r}_{f_i}$ are the vectors between the barycenter of the face vertices (b_{f_1}, b_{f_2} and b_{f_3}) and the removed vertices (v_3, v_4 and v_5). (b) The mesh before the *lifting step* is in gray dotted lines. The mesh after is in black continuous lines. The *lifting step* moves the position of the remaining vertices v_1, v_2 and v_3 according to the neighbor face $\mathbf{r}_{f_i}$ values to improve the R-D distortion performance.

During the decompression, once the connectivity has been decoded, the coarse coefficients C^l can be obtained from the coarse coefficients C^{l-1} because the vertices of M^{l-1} are a subset of the vertices of M^l. The details coefficients D^{l-1} are decoded from the compressed data. From (1), we have:

$$\mathbf{p}_{v_j} = \mathbf{r}_{f_i} + \mathbf{b}_{f_i}.$$

So, thanks to this local formula, it is possible to recover C^l from C^{l-1} and D^{l-1}. This process can be written under the form of the following equation:

$$C^l = P^l.C^{l-1} + Q^l.D^{l-1},$$

where P^l and Q^l are the synthesis filters determined by the previously described process.

This scheme corresponds to the lazy wavelet transform. It just separates the low and high frequency terms during the compression and reassemble them during the decompression.

5.1.1. Lifting step

To improve the R-D performance of our coder, after a decimation conquest the position of the mesh remaining vertices (C^{l-1}) are moved in function of the positions of the removed vertices (see Figure 12 (b)). Each vertex position $\mathbf{p}_{v_j}$ is locally modified as follows:

$$\mathbf{p}_{v_j} = \mathbf{p}_{v_j} + \gamma \frac{1}{|F_{v_j}|} \sum_{f_i \in F_{v_j}} \mathbf{r}_{f_i},$$

where F_{v_j} is the set of the neighbor faces of v_j. If a face f_i does not have a removed vertex, then $\mathbf{r}_{f_i} = 0$. In our experiments, we set $\gamma = 0.5$ because it provided the best results with our datasets.

This local process can be formulated as a *lifting step* in our global wavelet formulation. So, (3) and (4) become:

$$C^{l-1} = A^l.C^l + \gamma L^l.B^l.C^l,$$
$$D^{l-1} = B^l.C^l,$$

where L^l is the matrix that computes the average residual of the neighbor faces (second term of the formula 5.1.1). During the decompression it is possible to rebuild the matrix L^l and then to restore the vertex positions with the decoded residuals using the following formula:

$$C^l = P^l.(C^{l-1} - \gamma L^l.D^{l-1}) + Q^l.D^{l-1}.$$

5.1.2. Curvature prediction and residual projection

As explained in Section 4.2, the entropy coder does not directly encode the residuals $\mathbf{r}_{f_i}$ but instead encodes the $\mathbf{g}_{f_i}$ values projected in the local Frenet frames. Given (2), the global matrix of the symbols values S^{l-1} is determined by the following equation:

$$S^{l-1} = U^{l-1}.(D^{l-1} - \alpha G^{l-1}.C^{l-1}),$$

where G^{l-1} is the matrix used to compute the $\mathbf{l}_{f_i}$ values and U^{l-1} is the matrix used to perform the local projections. During the decompression, the wavelet coefficients matrix can be restored with

$$D^{l-1} = V^{l-1}.(S^{l-1} + \alpha G^{l-1}.C^{l-1}),$$

where V^{l-1} is the matrix that performs the reverse local projections, before applying the rest of the lifting scheme. The whole process is summarized by Figure 13.

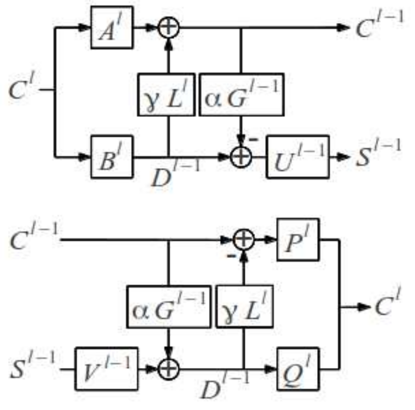

Figure 13: One level wavelet analysis and synthesis lifting scheme.

5.2. Adaptive quantization

To get the best rate-distortion performance when compressing a mesh, two variables can be played with: the number of vertices V and the number of geometry quantization bits B. For the single rate compression of triangle mesh, King and Rossignac proposed in [35] methods to optimize the choice of V and B to minimize either the approximation error or the file size. For the progressive compression of triangle meshes, Lee et al. [7] showed that the R-D performance of the original AD coder [15] can be significantly improved by interleaving decimation conquests with vertex global quantization operations (see Figure 14). The main rationale is that a precise level of the initial quantization is not needed for low LODs which have very few vertices. The decimation contests are encoded with the AD coder and the quantization contests are encoded with the Peng and Kuo geometry coder [19].

The authors propose two methods to choose at each iteration whether to decimate the mesh or to quantize its vertex positions. In the first method, denoted by *optimal* the decimated mesh and the quantized mesh are generated with their compressed data. The two R-D ratio are then compared. The chosen operation is the one that yields the lowest ratio. This method provides the best R-D performance for all levels of details but is computationally intensive. It requires to generate both meshes, to determine the size of the encoded data and to measure the two distortions with the initial mesh. Therefore, the authors recorded the choices made by their optimal coder on a mesh corpus to learn the parameters μ and β of a function $q_G(K_G)$ that provides the best number of quantization bits according to the level of decimation:

$$q_G(K_G) = round(\mu * log(K_G) - \beta)$$

where $round()$ corresponds to the nearest integer rounding function. K_g is defined as:

$$K_G = \frac{volume\ of\ bounding\ box}{area \times\ number\ of\ vertices}.$$

$\mu = -1.248$ and $\beta = -0.954$ are reported as the best parameter values for the selected triangle mesh corpus. By this way, a second method, denoted by *quasi-optimal*, was proposed to choose at each iteration whether to decimate or to quantize. If the current $q_G(K_G)$ value is lower than the current number of quantization bits, then the mesh is quantized. Else, it is decimated. The authors experimentally demonstrated that the *quasi-optimal* and *optimal* methods yield similar results. We implemented Lee's *quasi-optimal* method to improve the R-D performances of our coder at low rates.

6. Experimental results

The experimental results shown are obtained with an implementation based upon the halfedge data structure of the CGAL library [36]. We observe in our experiments that the computation times are approximately linear in the number of vertices. A 10M face mesh is compressed in 1m48s and decompressed in 1m22s on a desktop computer equipped with an Intel Core i7 CPU clocked at 2.80GHz and 8GB of RAM. We measure the distortion through the METRO software tool [37], after triangulating each polygon through inserting a vertex at the barycenter of its vertices and connecting it with all the polygon vertices. All results are obtained with the predictions described in Section 4.

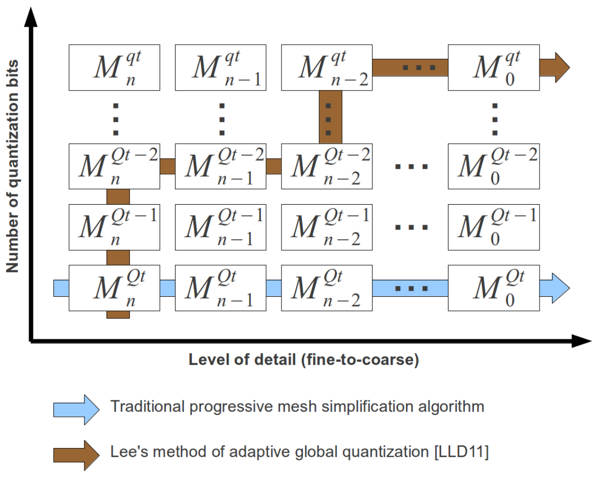

Figure 14: Progressive mesh traditional simplification vs. Lee's method for adaptive quantization [7]. Figure inspired from [7].

6.1. Progressive compression of polygon meshes

This section presents polygon mesh compression results. Figure 15, shows that the lifting scheme presented in Section 5.1.1 clearly improves the rate distortion curve at low rates without increasing the final compression rate. The R-D optimization algorithm provides even better distortion at low rates but increases the overall compression rate. Table 1 lists compression rates on polygon models depicted in Figure 19. In order to compare the effectiveness of our method against simple triangulation prior to compression (see Section 2.6), we triangulate polygonal models by choosing for each polygon an arbitrary vertex as pivot and adding edges between this vertex and all the others. The triangulated models are then compressed with the state of the art progressive compression method specialized to triangle meshes [7]. We add to the obtained compression rates the cost of the edge flag encoding required to restore the mesh initial connectivity after decompression (see Section 2.6). This cost is computed for each mesh with the Shannon entropy of the Boolean symbol sequence. The obtained values clearly improve over the trivial one. For our polygon mesh corpus, the trivial method costs on average 4 more bpv than our method.
Our codec results are also compared with the results of the single-rate coder from [38] to evidence the cost of the progressiveness. This cost is high for the simple models (Shark, Teapot, Triceratops, Beethoven, Fandisk, Elephant) as their regularity gets rapidly worse during the decimation. Complex, regular models (Neptune, Chinese lion, Gargoyle, Rabbit) are efficiently compressed. Our algorithm performs well with irregular models (Horse, Fertility, Ramesses, Dinosaur) com-

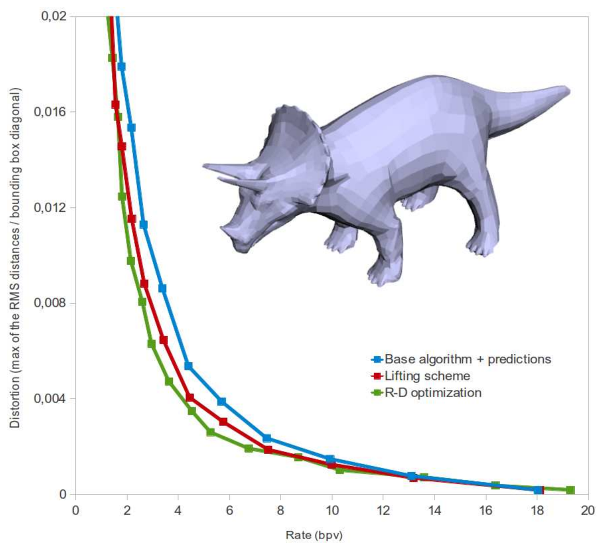

Figure 15: R-D curves for the compression of the Triceratops model with 10 bits quantization.

Model	# poly.	Quant.	Our scheme			[7]	[38]
			C.	G.	Tot.		
Beethoven	2812	10	5.7	16.7	22.4	26.0	16.6
Bunny	8814	10	3.8	11.8	15.6	20.3	12.2
Elephant	10895	12	3.4	12.3	15.8	25.4	11.8
Shark	2562	10	5.1	11.4	16.5	21.0	8.1
Teapot	1290	10	4.7	13.9	18.6	25.5	12.2
Triceratops	2834	10	5.4	12.6	18.0	22.1	11.9
Neptune	112658	12	1.5	6.7	8.1	17.8	5.6
Chinese lion	128339	12	1.4	8.0	9.4	19.6	6.9
Gargoyle	32126	12	2.7	12.5	15.3	23.9	12.1
Lucy VSA	76646	12	5.4	13.8	19.2	21.5	20.3
Hippo	32658	12	4.0	11.8	15.8	21,8	16.0
Horse	39698	12	4.1	15.2	19.3	20.4	19.1
Fandisk	12986	10	3.9	11.9	15.8	15.7	9.8
Dinosaur	28136	12	4.6	16.3	21.0	21.2	20.2
Venusbody	22720	12	3.6	12.6	16.1	16.9	13.4
Rabbit	134074	12	3.2	11.4	14.6	16.2	12.1
Fertility	483226	12	3.6	10.2	13.7	14.7	13.4
Ramesses	1652528	12	4.2	7.7	11.9	12.3	11.9

Table 1: Compression rates in bits per vertex, without any R-D optimizations. The first part of the table contains meshes with arbitrary face degrees. The second part contains only triangle meshes. C. stands for connectivity. G. stands for geometry. We first triangulate the polygon meshes before compressing them with [7]. We add to the obtained compression rates the cost of the edge flags required to restore the original connectivity.

pared to the single-rate approach. It even improves on very irregular models such as the VSA-remeshed [39] Lucy or the Hippo models. Figure 16 depicts the decompression of the Bimba quadrangle surface mesh.

6.2. *Progressive compression of triangle meshes*

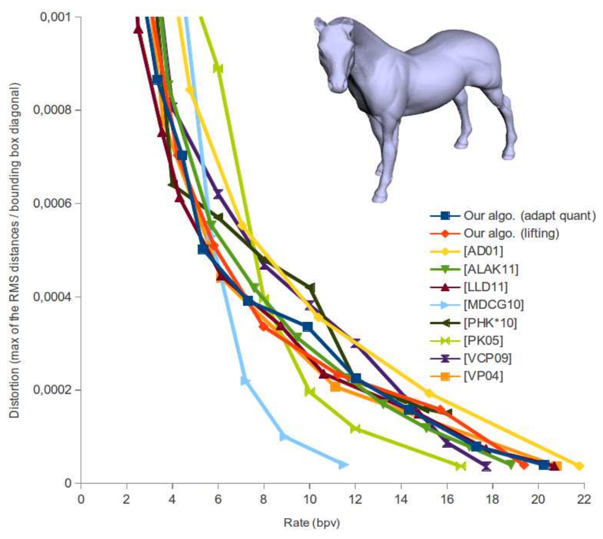

Figure 17: R-D curves for the compression of the Horse model with 12 bits quantization.

We now compare our coder with state-of-the-art compression methods specialized to triangle meshes. Figure 17 and 18 depict R-D curves obtained with methods specialized to triangle meshes, and our algorithm. For our algorithm, we provide a curve with the lifting scheme and a curve with the adaptive quantization algorithm. We notice that several coders achieve better results in term of compression ratio and R-D curves than our algorithm. In particular, [24] performs well in terms of compression ratio and R-D distortion, at the price of high compression and decompression times (3m for a mesh with 20,000 vertices). The approach of Valette et al. [21] also provides very good results: the compression ratio are high and the R-D curve is excellent with the rabbit model, at the price of not guaranteeing the restoration of the initial connectivity. The octree coder [19] gives good compression rates but the distortion is high at low rates. The R-D optimized coder from [20] also performs well in terms of compression ratio but is weaker than our coder in terms of distortion at low rates. For triangle meshes, our algorithm provides similar distortion at low rates than the approach described in [7]. It also yields better compression rates as shown in the second part of Table 1. For the two presented triangle meshes the lifting scheme yields slightly better final compression rates than the adaptive quantization method. For the irregular horse model however, the R-D performance is worse at some point.

While being more general than progressive coders specialized to triangle meshes, our coder achieves competitive results for the compression of triangle meshes. It works with any 2-manifold mesh and exhibits good R-D performances at low rate and average compression rates.

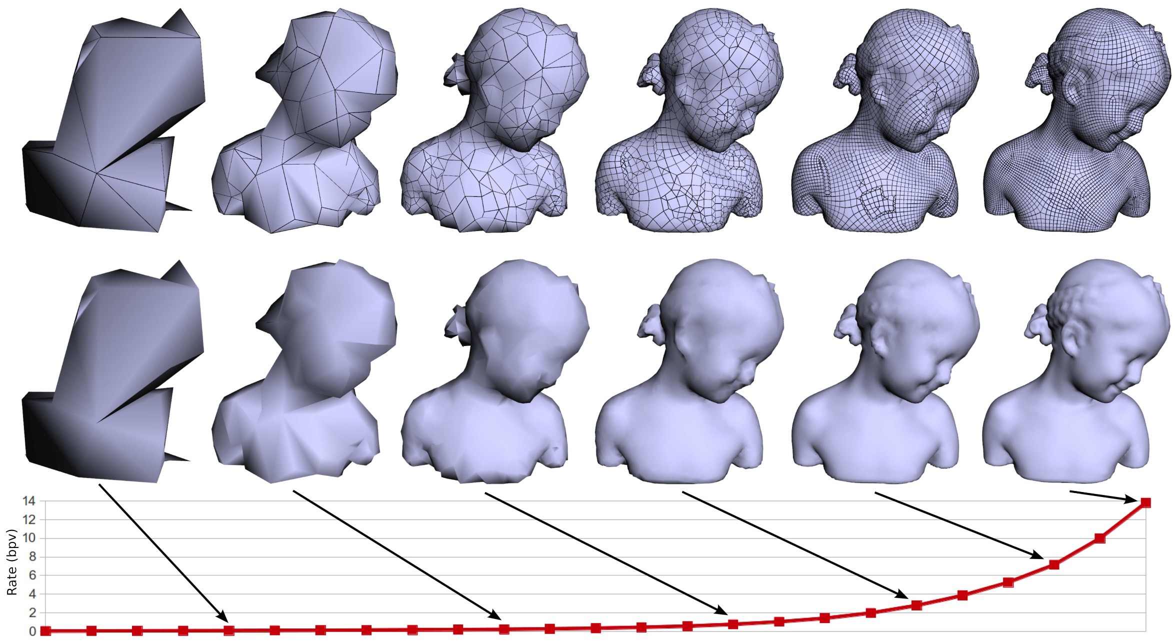

Figure 16: Decompression of the Bimba model (15770 quads) with the lifting scheme. The final compression rate is 13.8bpv with 12 bits quantization.

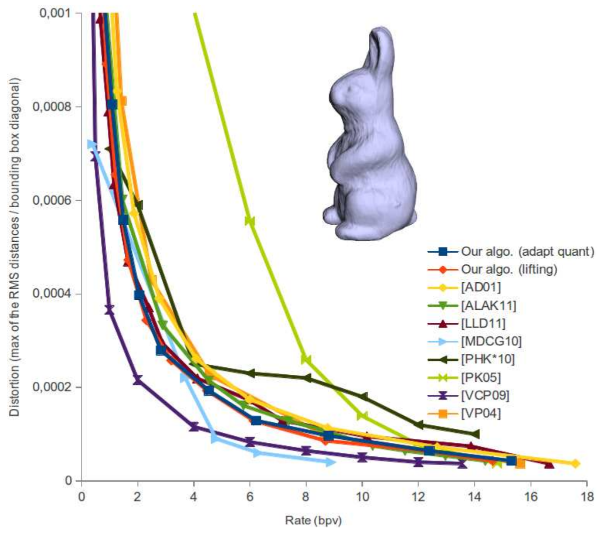

Figure 18: R-D curves for the compression of the Rabbit model with 12 bits quantization.

7. Conclusion

We introduced a new progressive mesh compression algorithm. One distinctive property of our method is that it can handle surface meshes with arbitrary face degree unlike previous approaches which only implement triangle mesh compression. Starting from a simple algorithm based on decimation traversals, we propose solutions to improve the compression of both geometry and connectivity. We then incorporate two methods designed to optimize the R-D performance: one based on wavelet lifting scheme, and the other based on adaptive global quantization. Experimental results show the effectiveness of our technical choices. Beside being more general than previous approaches, our method is also competitive for the compression of surface triangle meshes.

As future work, we wish to further improve the compression rates by optimizing the selection of the *patch decimation* operations to maximize the number of vertices removed per *decimation* step. We will also investigate the best parameters for the adaptive quantization method specialized to non-triangle meshes, and will handle polygon meshes with boundaries.

Acknowledgements

The authors would like to thank Ho Lee for providing some experimental compression results first published in [7] and Jae-Kyun Ahn for providing the experimental compression results of its encoder [20]. Some models are courtesy of the AIM@SHAPE Repository.
This work has been founded by French National Research Agency (ANR) through COSINUS program

(project COLLAVIZ n°ANR-08-COSI-003) and the European Research Council (ERC Starting Grant "Robust Geometry Processing", Grant agreement 257474).

References

[1] D. King, A. Szymczak, J. R. Rossignac, Connectivity compression for irregular quadrilateral meshes, GVU Tech Report GIT-GVU-99-36.

[2] M. Isenburg, J. Snoeyink, Face fixer: compressing polygon meshes with properties, in: Proc. of SIGGRAPH, 2000, pp. 263–270.

[3] B. Kronrod, C. Gotsman, Efficient coding of non-triangular mesh connectivity, in: Proc. of the 8th Pacific Conference on Computer Graphics and Applications, 2000, p. 235.

[4] M. Isenburg, Compressing polygon mesh connectivity with degree duality prediction, in: Graphics Interface Conference Proc., 2002.

[5] A. Khodakovsky, P. Alliez, M. Desbrun, P. Schröder, Near-optimal connectivity encoding of 2-manifold polygon meshes, Graphical Models 64 (2002) 147–168.

[6] C. Courbet, C. Hudelot, Random accessible hierarchical mesh compression for interactive visualization, in: Proc. of the Symposium on Geometry Processing, 2009, pp. 1311–1318.

[7] H. Lee, G. Lavoué, F. Dupont, Rate-distortion optimization for progressive compression of 3d mesh with color attributes, The Visual Computer 28 (2012) 137–153.

[8] H. Hoppe, Progressive meshes, in: Proc. of SIGGRAPH, 1996, pp. 99–108.

[9] J. Popović, H. Hoppe, Progressive simplicial complexes, in: Proc. of SIGGRAPH, 1997, pp. 217–224.

[10] G. Taubin, A. Guéziec, W. Horn, F. Lazarus, Progressive forest split compression, in: Proc. of SIGGRAPH, 1998, pp. 123–132.

[11] R. Pajarola, J. Rossignac, Compressed progressive meshes, IEEE Transactions on Visualization and Computer Graphics 6 (2000) 79–93.

[12] Z. Karni, A. Bogomjakov, C. Gotsman, Efficient compression and rendering of multi-resolution meshes, in: Proc. of the conference on Visualization, 2002, pp. 347–354.

[13] J. Li, C.-C. J. Kuo, Progressive coding of 3-d graphic models, in: Proc. of the IEEE, Vol. 86, 1998.

[14] D. Cohen-Or, D. Levin, O. Remez, Progressive compression of arbitrary triangular meshes, in: Proc. of the IEEE Visualization Conference, 1999.

[15] P. Alliez, M. Desbrun, Progressive compression for lossless transmission of triangle meshes, in: Proc. of SIGGRAPH, 2001, pp. 195–202.

[16] C. Touma, C. Gotsman, Triangle mesh compression, in: Graphics Interface 98 Conference Proc., 1998, pp. 26–34.

[17] S. Valette, R. Prost, Wavelet-based progressive compression scheme for triangle meshes: Wavemesh, IEEE Transactions on Visualization and Computer Graphics 10 (2004) 123–129.

[18] P.-M. Gandoin, O. Devillers, Progressive lossless compression of arbitrary simplicial complexes, in: Proc. of SIGGRAPH, 2002, pp. 372–379.

[19] J. Peng, C.-C. J. Kuo, Geometry-guided progressive lossless 3d mesh coding with octree (ot) decomposition, in: Proc. of SIGGRAPH, 2005, pp. 609–616.

[20] J.-K. Ahn, D.-Y. Lee, M. Ahn, C.-S. Kim, R-d optimized progressive compression of 3d meshes using prioritized gate selection and curvature prediction, The Visual Computer 27 (2011) 769–779.

[21] S. Valette, R. Chaine, R. Prost, Progressive lossless mesh compression via incremental parametric refinement, in: Proc. of the Symposium on Geometry Processing, 2009, pp. 1301–1310.

[22] J. Peng, Y. Huang, C.-C. J. Kuo, I. Eckstein, M. Gopi, Feature oriented progressive lossless mesh coding, Computer Graphics Forum 29 (7) (2010) 2029–2038.

[23] Z. Karni, C. Gotsman, Spectral compression of mesh geometry, in: Proceedings of SIGGRAPH, 2000, pp. 279–286.

[24] K. Mamou, C. Dehais, F. Chaieb, F. Ghorbel, Shape approximation for efficient progressive mesh compression, in: Proc. of the IEEE International Conference on Image Processing, 2010.

[25] A. Khodakovsky, P. Schröder, W. Sweldens, Progressive geometry compression, in: Proc. of SIGGRAPH, 2000, pp. 271–278.

[26] A. Khodakovsky, I. Guskov, Compression of normal meshes, in: Geometric Modeling For Scientific Visualization, 2003, pp. 189–206.

[27] F. Payan, M. Antonini, 3d mesh wavelet coding using efficient model-based bit allocation, in: Proc. of the First International Symposium on 3D Data Processing Visualization and Transmission, 2002, pp. 391–394.

[28] F. Payan, M. Antonini, An efficient bit allocation for compressing normal meshes with an error-driven quantization, Comput. Aided Geom. Des. 22 (2005) 466–486.

[29] Tutte, A census of planar maps, Canadian Journal of Mathematics 15 (1963) 249–271.

[30] A. D. Kalvin, R. H. Taylor, Superfaces: Polygonal mesh simplification with bounded error, IEEE Computer Graphics and Applications 16 (1996) 64–77.

[31] F. Tampieri, Newell's method for computing the plane equation of a polygon, 1992, pp. 231–232.

[32] H. Lee, G. Lavoué, F. Dupont, Adaptive coarse-to-fine quantization for optimizing rate-distortion of progressive mesh compression, in: Vision, Modeling, and Visualization Workshop, 2009.

[33] Michael Schindler's range coder: http://www.compressconsult.com/rangecoder/.

[34] W. Sweldens, The lifting scheme: A custom-design construction of biorthogonal wavelets, Applied and Computational Harmonic Analysis 3 (2) (1996) 186–200.

[35] D. King, J. Rossignac, Optimal bit allocation in compressed 3d models, Computational Geometry: Theory and Applications 14 (1999) 91–118.

[36] Cgal library: http://www.cgal.org/.

[37] P. Cignoni, C. Rocchini, R. Scopigno, Metro: Measuring error on simplified surfaces, Computer Graphics Forum 17 (2) (1998) 167–174.

[38] M. Isenburg, P. Alliez, Compressing polygon mesh geometry with parallelogram prediction, in: Proc. of the conference on Visualization, 2002, pp. 141–146.

[39] D. Cohen-Steiner, P. Alliez, M. Desbrun, Variational shape approximation, in: Proc. of SIGGRAPH, 2004, pp. 905–914.

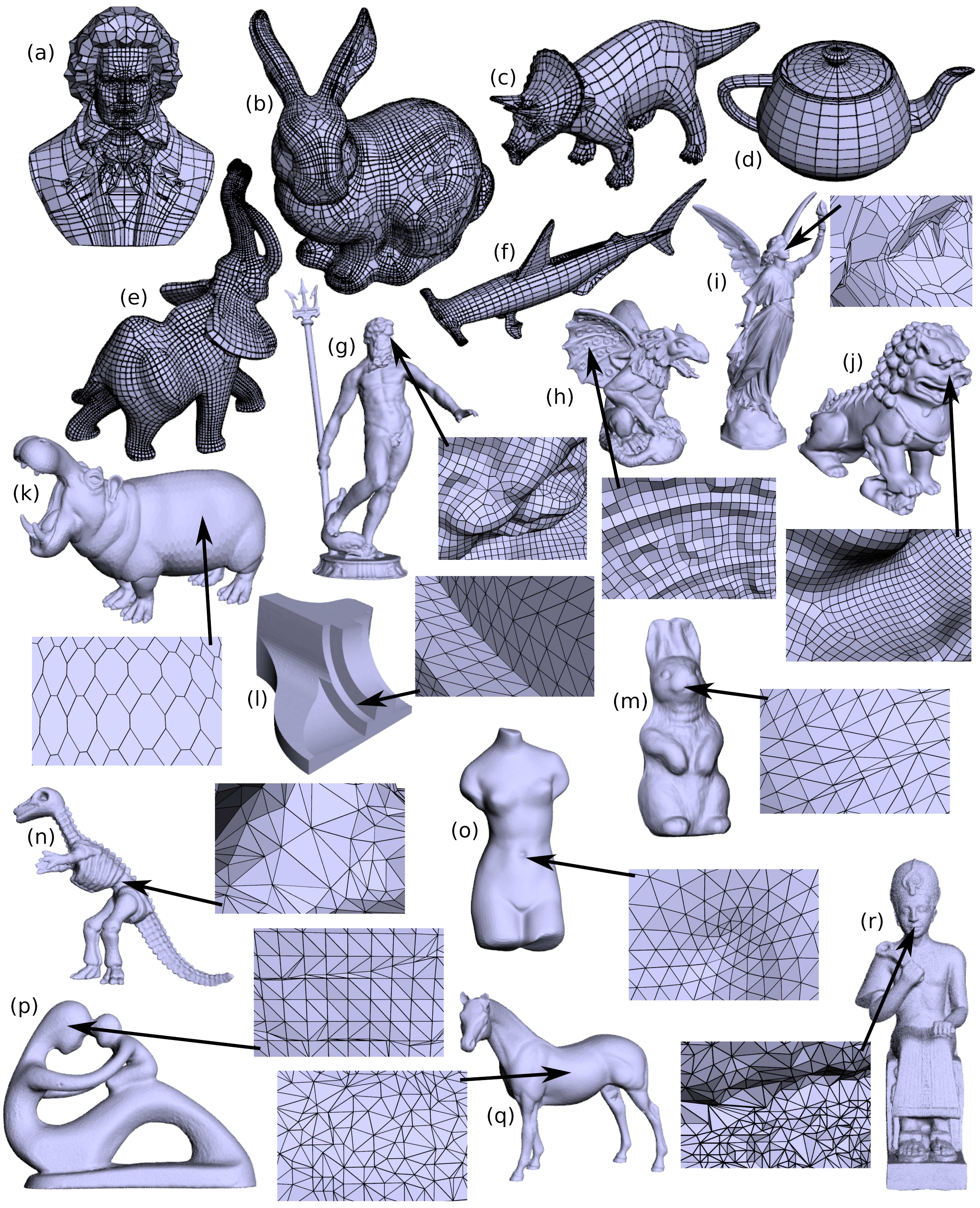

Figure 19: Input meshes from Table 1 and their tessellations: (a) Beethoven (b) Bunny (c) Triceratops (d) Teapot (e) Elephant (f) Shark (g) Neptune (h) Gargoyle (i) Lucy VSA (j) Chinese lion (k) Hippo (l) Fandisk (m) Rabbit (n) Dinosaur (o) Venusbody (p) Fertility (q) Horse (r) Ramesses.