Pacific Graphics 2010
P. Alliez, K. Bala, and K. Zhou
(Guest Editors)

Volume 29 (2010), Number 7

Feature Oriented Progressive Lossless Mesh Coding

paper1060

Abstract

A feature-oriented generic progressive lossless mesh coder (FOLProM) is proposed to encode triangular meshes with arbitrarily complex geometry and topology. In this work, a sequence of levels of detail (LODs) are generated through iterative vertex set split and bounding volume subdivision. The incremental geometry and connectivity updates associated with each vertex set split and/or bounding volume subdivision are entropy coded. Due to the visual importance of sharp geometric features, the whole geometry coding process is optimized for a better presentation of geometric features, especially at low coding bitrates. Feature-oriented optimization in FOLProM is performed in hierarchy control and adaptive quantization. Efficient coordinate representation and prediction schemes are employed to reduce the entropy of data significantly. Furthermore, a simple yet efficient connectivity coding scheme is proposed. It is shown that FOLProM offers a significant rate-distortion (R-D) gain over the prior art, which is especially obvious at low bitrates.

Categories and Subject Descriptors (according to ACM CCS): E.4 [Data]: CODING AND INFORMATION THEORY—Data compaction and compression

1. Introduction

High resolution 3D models, which have become common with the advent of automatic 3D scanning devices, take a huge amount of space to store in raw formats. Reducing the storage, bandwidth and rendering costs of 3D data has been an active research topic for over a decade. With the rapid development in network-based interactive 3D applications, progressive 3D model coding has become vital to the transmission of large models over networks of limited bandwidth. Further, such coding techniques have to handle generic models including manifolds and non-manifolds, encode both geometry and connectivity and achieve good rate-distortion performance. In this paper, we propose techniques to address all of the above requirements in a mesh coding scheme.

In the proposed 3D mesh compression scheme, called FOLProM, we progressively encode geometry as well as connectivity information. Compared to the previous work, the proposed FOLProM coder has the following distinctive features.

- **Feature-oriented geometry coding.** Geometric features like high curvature regions, usually convey important visual information. Hence the entire geometry coding process in our coder is optimized to preserve geometric features of the input model.
  - *Feature-directed hierarchy control.* A hierarchy of model representations is considered in order to create an appropriate level of detail of the model at a given bit-rate budget. Since normal vectors and hence curvatures convey important geometric and perceptual information, an adaptive distance metric based on normal vectors for clustering control and a Gaussian function based curvature weighting for representative computation are proposed to optimize the approximation quality at the intermediate level of detail within the hierarchy.
  - *Feature-based adaptive quantization.* The quantization parameters to encode geometry are adaptively determined based on the local shape of the surface. For a given number of quantization bits, local distortion is minimized by assigning higher quantization resolution to the more important spatial dimension.
- **Effective entropy reduction techniques.** The employment of cylindrical coordinates, prediction of polar angles and delta coding of heights leads to a significantly reduced data entropy.
- **Simple yet effective connectivity coder.** Our method employs a simple connectivity coder that takes advantage of the local distribution of vertices to reduce the data entropy and at the same time handles arbitrary variation in local topology, including non-manifold connectivity.

As compared with related work, FOLProM with optimization leads to outstanding rate-distortion performance with well-preserved geometric features especially at low bit-rates.

1.1. Related Work

3D mesh compression techniques can be classified using different attributes [PKK05]: single-rate and progressive, lossless and lossy, connectivity-driven and geometry-driven approaches. The method presented in this work is a progressive lossless geometry-driven mesh coder that can handle meshes of arbitrary topology including non-manifolds, even complex triangle soups. We briefly highlight relevant work in this section. For a comprehensive survey on 3D mesh coding technologies, we refer the reader to [PKK05, AG03, GGK02].

Single-rate mesh coders encode a 3D mesh, part by part, in one resolution. They are lossless coders in that they preserve the original connectivity and allows only negligible geometry quantizationerrors [TR98, BPZ99b, TG98, AD01b, GS98, Ros99, CR04]. Later on, lossy single-rate mesh codecs such as [SRK02, AFSR03] were proposed, offering a higher coding gain by combining the compression process with remeshing. Most of the single-rate coders are connectivity driven in the sense that mesh traversal order is determined by the connectivity, which restricts the order and the coding of vertex positions.

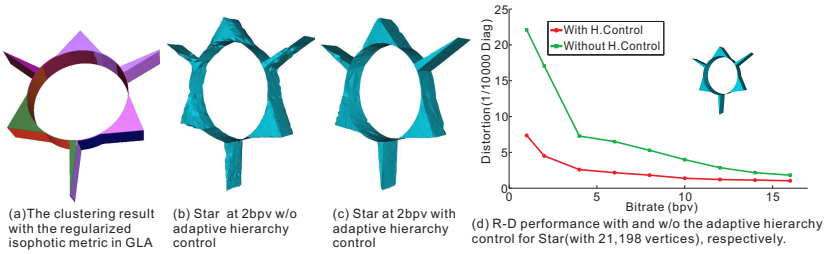
In contrast to a single-rate mesh coder, a progressive encoder represents the input mesh as a sequence of models with different resolutions, called levels of detail (LODs), from coarse to fine, and encodes the differences between adjacent LODs. There are connectivity-driven approaches [Hop96, PH97, TGHL98, PR00, COLR99, AD01a, LK98, BPZ99a, KBG02, VP04] and geometry-driven approaches [GD02, DG00, PK05a, KG00a, KSS00, KG00b, GGH02, GP05, VCP09] to progressive mesh coding. The former usually simplify an input mesh through a sequence of topological operations and encode the reverse of this simplification process while the latter usually give a higher priority to geometry coding, which dictates the order and method of connectivity coding, or even discard the original connectivity through remeshing.

Although remeshing-based coders produce outstanding rate-distortion performance, they are effective only for manifold or almost manifold meshes. Furthermore, they are not suitable for applications that demand faithful representation of the input mesh that call for lossless mesh coders. Progressive lossless encoders that work with manifold meshes only [Hop96, TGHL98, PR00, COLR99, AD01a, LK98, BPZ99a, KBG02, VP04, VCP09]) or general meshes including non-manifolds [PH97, GD02, PK05a, PEK06] encode the original connectivity and keep the geometric error within bounds as determined by the quantization parameters. It should be noted that the majority of the above-mentioned lossless mesh coders prequantize the vertex coordinates before encoding. There are also lossless geometry coders such as [ILS04, ILS05] that do not prequantize but encode the original floating-point vertex coordinates. State-of-the-art progressive lossless mesh coders include [GD02, PK05a, VP04, VCP09]. In [VP04], the progressive coding process relies on a wavelet-based multiresolution analysis of irregular meshes. The connectivity coder encodes the insertion of new vertices, the face subdivision and the edge flips associated with each incremental topology refinement. The geometry coder encodes the coefficients resulting from a wavelet decomposition of the geometry. As claimed in [VP04], its coding performance outperforms that of [AD01a] and [KBG02] in general. The algorithm in [VCP09] remodels the progressive mesh compression as a model generation problem. Starting from a base model, it iteratively inserts vertices to the approximation and performs local Delaunay mesh refinements correspondingly. Connectivity of the intermediate models will be corrected through a sequence of edge flips only after the geometry information has been fully transmitted. As demonstrated in [VCP09], it in general leads to efficient coding of models with smooth surfaces. But its performance on models with sharp features is not well documented and its performance is highly dependent on the order of vertex insertion which sometimes may cause degeneration in the reconstructed connectivity. Both [GD02] and [PK05a] apply a hierarchical kd-tree- or octree-based partitioning to the bounding box of an input mesh, and encode the geometry and connectivity updates associated with each tree node subdivision. They achieve outstanding final bitrates, but generate highly aliased intermediate meshes. In [PK05b] the geometry of the 3D mesh is progressively encoded, but the connectivity is not. It performs hierarchical vertex set splits, calculates a representative for each newly generated vertex subset, and encodes the associated geometry updates.

The FOLProM coder proposed in this work is a generic progressive lossless triangular mesh coder that can encode triangular meshes with arbitrary geometric and topological complexity. It does not resample the original surface, preserves the original connectivity, and restores each vertex up to a tolerance from its original position. Its superiority over [PK05b] is demonstrated in Section 2, and its comparisons with several state-of-the-art progressive mesh coders that encode both geometry and connectivity, [VP04, PK05a, VCP09, KG00a], are detailed in Section 4.

1.2. Algorithm Overview

The FOLProM coder first splits a vertex set into several child vertex sets using generalized Lloyd algorithm (GLA), and the geometric representative of each vertex set is computed. The representative is chosen to be the vertex that is close to the averaged geometry center and has high curvature. The number of children is entropy encoded and the offsets of



(a)The clustering result with the regularized isophotic metric in GLA (b) Star at 2bpv w/o adaptive hierarchy control (c) Star at 2bpv with adaptive hierarchy control (d) R-D performance with and w/o the adaptive hierarchy control for Star(with 21,198 vertices), respectively.

Figure 1: *Effectiveness of the adaptive hierarchy control for Star (with 21,198 vertices).*

their representatives from the representative of the parent vertex set are quantized and entropy encoded. If the vertex set has only one vertex, the residual between the representative after local quantization and the real vertex position is further refined through subdivision of its bounding volume. In this case, the index of the nonempty child bounding volume is encoded. The vertex set split and/or bounding volume subdivision processes are iteratively performed until a terminating criterion is met. In addition, vertex sets with more than one vertex are prioritized over those with single vertex in split. Further, the associated local connectivity updates are encoded. At each intermediate stage, the representatives of all current vertex sets and the connectivity between them form an approximation to the original mesh.

In essence, for each cluster in the FOLProM coder, we employ novel placement and prediction of vertex cluster representatives and introduce feature-based prioritization of vertex set split operations to preserve geometric features. These help us achieve outstanding R-D performance, especially at low bitrates. In addition, a simple yet effective connectivity coder is adopted.

2. Geometry Coder

The FOLProM encoder splits a parent vertex set into K (a user-defined constant) children if the vertex set contains more than K vertices, and m children if the set contains m vertices ($2 \leq m \leq K$). Each child vertex set has a representative calculated as its geometric center or a vertex that is close to the geometric center and has a high curvature. The geometry of the representative is encoded by its offset from the parent set representative. We split vertex sets with more than one vertex first till we reach the stage of exactly one vertex per set. Thereafter, each set will not be split but its representative is further refined and moved towards its correct position through bounding volume subdivision as described in Section 2.4. The number of children, their offsets and bounding volume indices are all arithmetic-coded.

Regarding the choice of value for K, too small a value will not enable us to fully utilize the advantage of prediction techniques as described in Section 2.3; too big a value will lead to an explosive number of vertex sets and hence the coding bits in the intermediate meshes while the geometry precision is still low. In our experiments, $K = 4$ consistently gave good results to us.

When compared to [PK05b], the proposed geometry coder uses sophisticated techniques including adaptive hierarchy control, adaptive offset quantization, prediction techniques and detail refinement, as detailed below.

2.1. Adaptive Hierarchy Control

In the generation of vertex set and representative hierarchy, the Generalized Lloyd Algorithm (GLA) process automatically adapts to the specific distribution of geometry samples in a mesh in order to optimize the quality of geometry approximation in an intermediate LOD. It returns when the decrease in distortion between two consecutive iterations is below a threshold or a maximum number of iterations is reached.

Two key ingredients in GLA that we can control are the definition of the distance metric, $d(\mathbf{v_1}, \mathbf{v_2})$ between two points $\mathbf{v_1}$ and $\mathbf{v_2}$ and the computation of the representative for each vertex set.

Distance Metric

A straightforward choice is the Euclidean distance as adopted in [PK05b]. However, it tends to collapse feature regions that contain vertices close in distance but high variance in their normals. On the other hand, a purely normal-based distance metric for shape approximation does not work for our top-down hierarchy construction since we do not want to group distant vertices with similar normals, especially at an early stage of the hierarchy construction process. Hence, we propose a distance metric that is a convex combination of the Euclidean and normal-based metrics:

$$d(\mathbf{v_1}, \mathbf{v_2}) = w_n \cdot d_n(\mathbf{n_1}, \mathbf{n_2}) + (1 - w_n) \cdot d_e(\mathbf{v_1}, \mathbf{v_2}),$$

where the Euclidean distance d_e and the normal difference d_n are defined by

$$\begin{aligned} d_e(\mathbf{v_1}, \mathbf{v_2}) &= |\mathbf{v_1} - \mathbf{v_2}|, \\ d_n(\mathbf{n_1}, \mathbf{n_2}) &= |\mathbf{n_1} - \mathbf{n_2}|, \end{aligned}$$

and where $\mathbf{n_1}$ and $\mathbf{n_2}$ are the unit normals at points $\mathbf{v_1}$ and

$\mathbf{v_2}$, and w_n is a weighting factor which is adaptively determined based on the geometric feature of a vertex set to split. Generally, if the normal variation is large within a vertex set, w_n should be large. To be more specific, for a vertex set with K vertices, w_n is determined as

$$w_n = 1 - a(1 - \frac{1}{2} ave(d_n(\mathbf{n_i}, \mathbf{n_R})))^b,$$

where $\mathbf{n_i}$ represents the normal of any vertex in the vertex set, $\mathbf{n_R}$ is the unit average normal over all the member vertices, given by $norm(\frac{1}{K}\sum_{i=1}^{K}\mathbf{n_i})$. Since the maximum value of $ave(d_n(\mathbf{n_i}, \mathbf{n_R}))$ can be 2, it is normalized by a factor of 1/2. a and b are user-defined constants with $0 \leq a \leq 1$ and $b \geq 1$, and w_n is in the range of $[0,1]$. In our experiments, $a = 1$ and $b = 8$ gave good results for meshes with geometric coordinates in the range of $[0, 2^{12})$. A general rule in the choice of a is that it should be inversely proportional to the geometric scale of the input mesh in order to maintain the relative scale of $d_e(\mathbf{v_1}, \mathbf{v_2})$ and $d_n(\mathbf{n_1}, \mathbf{n_2})$ in the proposed distance metric, where the former's scale varies with different input meshes, while the latter's keeps constant.

Representative Computation

A straightforward approach is to use the averaged geometric center of a vertex set as its representative. But it tends to smooth out regions with sharp creases and/or corners which are often seen in CAD models (*e*.g., Fandisk). Based on the fact that those regions contain high variation of local curvatures, we identify those regions and adopt a Gaussian function based curvature weighting approach for the computation of representatives.

For a given vertex set V, we first check if the standard deviation of vertex curvatures, D_c, is above a threshold, T_c. For every vertex $\mathbf{v_i} \in V$, we use the standard deviation of its adjacent facets' normals (normalized to the range [0,1]) as a measure of the curvature c_i. If $D_c > T_c$, we use the Gaussian function based curvature weighting approach to select the representative, which will be described below; Otherwise, we compute the representative as the averaged geometric center.

We compute a priority values p_i for each $v_i \in V$, and choose the vertex with the largest p_i as the representative. A desirable representative should be close to the averaged geometric center and have high curvature at the same time. Thus, we compute p_i as the weighted curvature at $\mathbf{v_i}$, *i*.e., the product of c_i and a Gaussian function of the distance between $\mathbf{v_i}$ and the geometric center $\mathbf{c}$ of V. Specifically, for each $\mathbf{v_i} \in V$, we calculate its priority value p_i as

$$p_i = \frac{c_i}{\sigma\sqrt{2\pi}} e^{-|\mathbf{v_i}-\mathbf{c}|^2/(2\sigma^2)}$$

where σ is the parameter which determines the shape of the Gaussian function curve. In order for the shape of the Gaussian function curve to adapt to the dimension of any vertex set, we specify σ as $\sigma = r_g \times A$ where A is the average distance of $\mathbf{v_i}$ to $\mathbf{c}$ and r_g is a system parameter which is set to 3 in our experiments.

The clustering for Star (with 21,198 vertices) corresponding to an intermediate LOD is illustrated in Fig. 1(a), where different clusters are colored differently. It clearly demonstrates the effect of the normal term in the proposed regularized isophotic metric of distance, *i*.e., adjacent surface patches are neatly separated by sharp edges in most cases.

In order to demonstrate the effectiveness of the proposed adaptive hierarchy control techniques, we show in Fig. 1(d) the R-D curves for the coding of Star with and without the adaptive hierarchical control, respectively. In this figure, the bitrate is reported in the unit of bits per vertex (bpv), and the distortion is the mean error as measured with the METRO tool [CRS98] on a scale of 10^{-4} with respect to the diagonal of the original mesh's bounding box. In addition, the visual results of two intermediate models decoded at 2bpv without and with the adaptive hierarchical control are shown in Fig. 1(b) and Fig. 1(c), respectively. Comparing Fig. 1(b) and Fig. 1(c), we see more indentations on the model surface in Fig. 1(b) due to the clustering of spatially close vertices with highly different normals, such as those on the two sides of each thin wing; we see less sharp edges in Fig. 1(b) due to the computation of representatives without taking into account the curvature information, which tends to smooth out the regions around sharp edges.

2.2. Adaptive Offset Quantization

Once the vertex set is split, the child representatives are encoded by their geometric offsets from the parent representative. In FOLProM, we use cylindrical coordinates (ρ, θ, z), where ρ is the distance from the axis of the cylinder, θ is the angle from the X-axis of the local coordinate system of the cylinder, and z is the height from the tangent plane. In a typical vertex split, we expect that child representatives are rather evenly distributed around the parent, which is typically observed in many 3D meshes. Hence, the children's $\rho(z)$ coordinates should have similar magnitudes, and the angles between the projections of children on XY plane with respect to the z axis should be similar too, all of which are good for entropy reduction. The Cartesian coordinates used in [PK05b] do not have such an advantage. Furthermore, the ρ and z components in cylindrical coordinates roughly correspond to the tangential and normal dimensions of a local neighborhood, respectively, which provides a distinct advantage over spherical coordinates since it facilitates adaptive quantization dictated by the local surface feature as detailed below. It should be noted that local quantization is also used in [LAD02] for geometry coding. Compared to [LAD02], our quantization scheme adapts to the relative importance of normal and tangential accuracy and the cylindrical coordinates representation enables effective prediction and leads to significantly reduced data entropy.

Unlike most progressive lossless mesh coders developed

before, the FOLProM encoder does not pre-quantize input mesh vertices in a global frame. Instead, for each vertex set split, the cylindrical coordinates of child representative offsets are quantized in a local frame that has the parent vertex-set representative **p** as its origin, normal at **p** as the Z axis, projection of one of the global frame base vectors on to the tangent plane at **p** as the X-axis, and the Y axis as $Z \times X$. Note that we obtain the normal at **p** from the best fit plane at **p** along with its neighbors computed using linear least squares. The quantization parameter is fixed for the θ components, but adaptively determined for the ρ and z components so as to adaptively assign higher quantization resolution to the more important dimension as explained below.

According to [KSS00, KG00b], normal accuracy is often more important than parametric (or tangential) accuracy in manifold mesh approximation. However, for local surfaces with high curvature, tangential accuracy may be more important. This is illustrated by 2D examples in Fig. 2. The left and right plots show a polyline (in black) with low and high curvatures at the middle vertex, respectively. In the left plot, perturbation of the middle vertex along the tangent is visually less disturbing and less distinguishable from the original, than perturbing it by the same amount in the normal direction. In the right plot, the effect is opposite. These 2D examples demonstrate that the relative importance of normal and tangential accuracy depends on the local curvature.

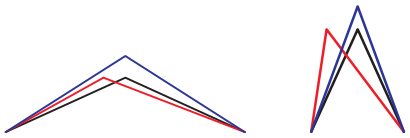


Figure 2: *Illustration of the relative importance of normal/tangential accuracy to the curve approximation quality with (a) a relatively flat polyline and (b) a highly curved polyline.*

A quantization parameter is the distance between two adjacent quantization levels. It should be small (fine) for more accuracy in representation. Let G_ρ, G_θ and G_z be the quantization parameters of ρ, θ, and z respectively. For a vertex, v, to split, we first find the ranges of its neighbors' ρ and z coordinates, denoted by R_ρ and R_z, respectively. Generally speaking, the smaller (larger) the ratio R_z/R_ρ, the flatter (more curved) the local surface and, hence, the more important the normal (tangent) accuracy. Thus, $G_\rho \propto R_\rho$ and $G_z \propto R_z$. Specifically, we have

$$G_\rho = R_\rho/Q,\ G_\theta = 2\pi/Q \text{ and } G_z = R_z/Q,$$

where Q is a parameter representing the number of quantization levels for each coordinate component. Parameters G_ρ and G_z not only adapt to the relative importance of normal and tangential components but also lead to empirically more compact distribution of quantized values of ρ and z, which is good for coding efficiency. Using the proposed adaptive quantization scheme, the L_2 distortion is reduced by more than a third for most of our test models over a wide range of bit-rates.

2.3. Prediction Techniques

Assuming even distribution of children around the parent, we expect even distribution of children's θ coordinates in the range of $[0, 2\pi)$, and similar delta-z coordinate between each child and the parent, which serve as the basis of our prediction techniques as detailed below. Suppose that vertex v in a given LOD is to be split into K child vertices, c_i, with quantized offsets (ρ_i, θ_i, z_i), where $i = 1, \cdots, K$ and $K > 1$. The encoder maintains the K child offsets in the lexicographic order of θ, ρ and z. In order to reduce the redundancy in the offset coordinates, predictors are adopted to predict the θ and z coordinates, and the prediction residuals are arithmetic-coded. The ρ coordinates are directly arithmetic-coded without prediction.

The coordinate θ_1 of the first child vertex is directly arithmetic-coded. The value θ_i, $2 \le i \le K$, can be predicted from θ_1 and θ_{i-1} based on the following idea. The 2D polar angles of c_1 and c_{i-1} are

$$\alpha_1 = \theta_1 G_\theta \text{ and } \alpha_{i-1} = \theta_{i-1} G_\theta,$$

respectively, where G_θ is the quantization parameter for the θ offsets. By assuming that the 2D polar angles of the remaining $K - i + 1$ offsets are evenly distributed between α_{i-1} and $2\pi + \alpha_1$, but not beyond 2π, we can predict θ_i by

$$\theta_i' = \min[\theta_{i-1} + \frac{2\pi + \alpha_1 - \alpha_{i-1}}{(K - i + 2)G_\theta}, \frac{2\pi}{G_\theta}].$$

Subsequently, the residual $\theta_i - \theta_i'$ is arithmetic-coded.

Next, we encode the z_i value of each child c_i, $i = 1, \cdots, K$. If we denote the Z coordinate of the parent v as z_p, z_i is predicted by

$$z_i' = z_p.$$

The residual $z_i - z_i'$ is arithmetic-coded.

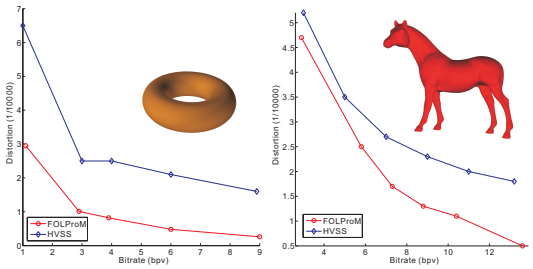
Using the proposed predictors, the entropies of θ- and z-information are significantly reduced. Predictions on ρ yield little performance improvement in our experiments and, hence, the values of ρ are directly arithmetic-coded.

2.4. Detail Refinement

During the encoding and decoding process, both the encoder and the decoder perform the same calculation of the local frame and the local quantization parameters associated with each vertex split. As a result, they always maintain the same bounding volume centered on each newly generated representative. To be more specific, if we denote

the locally quantized polar coordinates of the representative as (ρ_c, θ_c, z_c) and the quantization parameters as G_ρ, G_θ and G_z, respectively, the original position of the representative is bounded by the cylindrical volume defined by two corner points, $[\rho_c - 0.5G_\rho, \theta_c - 0.5G_\theta, z_c - 0.5G_z]$ and $[\rho_c + 0.5G_\rho, \theta_c + 0.5G_\theta, z_c + 0.5G_z]$. When it comes to the point that only one original vertex is contained in a vertex set, the encoder and the decoder perform the same iterative cylindrical bounding volume subdivision process to refine the geometry of the contained vertex. At each step, we subdivide the cylindrical bounding volume along one of the three dimensions (*i.e.*, ρ, θ and z), resulting in two child cylindrical bounding volumes, and arithmetic-code one bit specifying the nonempty one. In the reconstructed mesh, the representative position will be refined to the centroid of the nonempty cylindrical child bounding volume.

All the above novel techniques and improvements in geometry coding lead to a significant R-D advantage of the proposed FOLProM geometry coder over that of [PK05b] called HVSS (Fig. 3). Note that [PK05b] does not encode connectivity.



(a) (b)

Figure 3: *Comparison of geometry R-D curves for (a) Torus and (b) Horse with FOLProM and HVSS.*

3. Connectivity Coder

3.1. Overview

Although single-rate connectivity coders [TG98, AD01b, CR04] have achieved excellent compression ratios, they are not compatible with progressive mesh coding, and only applicable to manifold or almost manifold mesh coding. Connectivity coders have also been proposed in lockstep with progressive geometry coders, among which [AD01a, GD02, PK05a] yield the state-of-the-art connectivity compression ratio. It should be noted that [AD01a] processes only manifold meshes, while [GD02, PK05a] can handle meshes of arbitrary topology. The proposed connectivity coder also applies to meshes of arbitrary topology, even triangle soups, and it is compatible with the proposed progressive geometry coder.

After encoding the geometry of child vertices, we encode their connectivity affected by the vertex split. If we denote two vertices in an intermediate LOD as r_1 and r_2 and their corresponding vertex sets as V_1 and V_2, respectively, r_1 and r_2 are connected if and only if an edge in the original mesh connects an original vertex in V_1 and an original vertex in V_2. Associated with each vertex split, triangular facets incident on the parent vertex are discarded, while a new triangular facet is constructed for every three vertices that include at least one of the children and are mutually connected. In this context, we assume that a triangle facet is double-sided and, in the original model, each three mutually connected vertices form a facet. Under these assumptions, it is provable that the proposed connectivity coder can accurately restore the original topology. Such assumptions are not uncommon in the compression literature and are used in [GD02, PK05a].

3.2. Connectivity Coding Algorithm

This section describes the process of splitting a vertex set into two sets. This process is repeated $K-1$ times in order to realize a 1-to-K vertex set split. Let the parent vertex set representative be v and the child representatives be v_1 and v_2. Furthermore, let the neighbors of v be N_i before the split, and neighborhood set $S_v = \{N_i\}$. In the splitting process, the following information have to be encoded.

- one bit representing if v_1 and v_2 are connected,
- $|S|$ bits representing if each N_i is connected to both v_1 and v_2, and
- $|S|$ bits representing if each N_i is connected to either v_1 or v_2 (not both).

The neighbors N_i that are connected to both v_1 and v_2 are called *pivot* neighbors, and other neighbors are *non-pivot* neighbors. Note that it is not possible that N_i is connected to neither v_1 nor v_2.

Depending on the mesh construction process, the order of neighbors in the encoder and the decoder may not be the same. In order to make a consistent coding context and reduce the data entropy, we order neighbors based on their likelihood of being a *pivot* vertex. Prevalent 3D modeling techniques strive to produce regular meshes where a dominant number of triangles over the surface are close to equilateral triangles. Thus, for each neighbor N_i, the closer $\triangle(N_i, v_1, v_2)$ is to an equilateral triangle, the more probable that N_i is a *pivot*. We measure the regularity, R_i, of $\triangle(N_i, v_1, v_2)$ as the ratio of its squared perimeter to its area:

$$R_i = \frac{S_i \times S_i}{A_i},$$

where

$$S_i = \frac{(|N_i v_1| + |v_1 v_2| + |v_2 N_i|)}{2},$$

$$A_i = \sqrt{S_i \times (S_i - |N_i v_1|) \times (S_i - |v_1 v_2|) \times (S_i - |v_2 N_i|)}.$$

It can be proved that R_i reaches its minimum if and only if $\triangle(N_i, v_1, v_2)$ is an equilateral triangle.

Thereafter, the neighbors can be ordered according to their regularities. Tiny offsets in floating point calculation may lead to inconsistent ordering of neighbors between the encoder and the decoder in some cases. In order to resolve this issue, the encoder and the decoder generate and maintain consistent and unique vertex IDs for each vertex in the same order. If the difference in regularity of two neighbors is less than a threshold, their relative order is determined in both the encoder and the decoder by their unique global IDs instead of their regularities. Thus, consistent neighbor ordering is always maintained in the encoder and the decoder. The bits that are used to flag *pivot*s are ordered correspondingly, forming a bit-pattern which is arithmetic-coded. This bit prioritization technique reduces the bit budget by about 50% for our test manifold meshes.

Given the above order, the *non-pivot* neighbors are predicted to be connected to the closer child vertex, and we use a one bit flag for each *non-pivot* neighbor to indicate if the prediction is correct or not. All the bit patterns are arithmetic-coded. This simple distance-based prediction technique reduces the bit budget by about 70% for the manifold meshes used in our experiments.

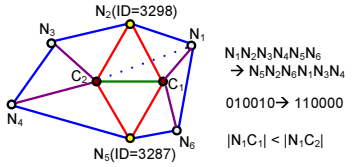


Figure 4: *Illustration of the 1-to-2 connectivity coding.*

The proposed 1-to-2 connectivity coding algorithm is illustrated in Fig. 4. The original order of neighbors are $N_1N_2N_3N_4N_5N_6$; after prioritization, their order is changed to $N_5N_2N_6N_1N_3N_4$. Correspondingly, the bit-pattern that specifies the *pivot* neighbors are changed from 010010 to 110000. Here, N_2 and N_5 have very close regularity values, and the ambiguity is resolved by their IDs, 3298 and 3287. For the *non-pivot* neighbor N_1, we predict it to be connected to a closer child vertex, C_1. Correctness of this prediction is encoded using a bit.

Similar to the proposed connectivity coder, [KBG02] also encodes the update to the local connectivity in the context of a 1-to-2 vertex split, and also the information as to which neighbors become *pivots* after the split. However, the connectivity coder in [KBG02] is based on a manifold local neighborhood and needs special codes to address the cases of non-continuous vertex split and illegal edge collapse, while our work uniformly processes a configuration of generic local neighborhood.

Our design criteria for the connectivity coder are its simplicity, range of applicability (should handle all sorts of triangular models) and run-time efficiency. The fact that the geometry bitrate is often a more dominant factor than the connectivity bitrate in progressive mesh compression gives us a wide space in design decisions. Connectivity coders described in [AD01a, GD02, PK05a] either handle only manifold meshes affecting their range of applicability, or use complex prediction and neighbor segmentation techniques that reduce their run-time efficiency. Hence these methods, although in general have better compression ratios, do not meet our design criteria. In our experiments, the proposed connectivity coder encodes the full-resolution mesh connectivity with about 5 bits per vertex (bpv) for manifold meshes, and about 10 bpv for triangle soups.

4. Experimental Results

4.1. Rate-Distortion Performance Comparison

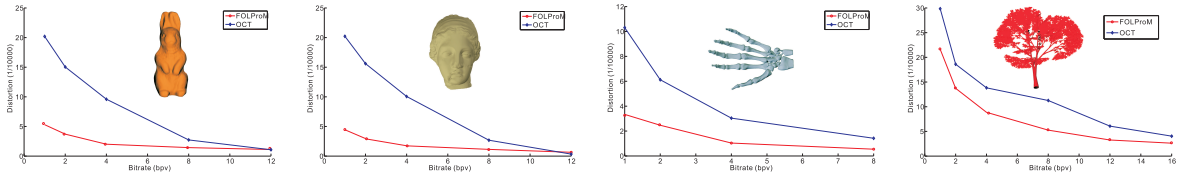
We compare the coding performance with state-of-the-art progressive mesh coders that encode both geometry and connectivity, [VP04, PK05a, VCP09, KG00a]. Further, meshes with a wide range of complexity and topology types have been used in our experiments.

Table 1: *R-D statistics for test meshes with distortion measured as mean error with the METRO tool [CRS98].*

Mesh(#v)	Encoder	Bitrates (bpv)					
		1.0	2.0	4.0	8.0	12.0	16.0
David	OCT	12.5	7.9	5.9	2.9	1.0	N/A
(258,329)	FOLProM	4.6	3.5	1.8	1.2	0.8	0.6
Donna	OCT	32.1	22.2	14.3	7.0	2.7	1.2
(50,691)	FOLProM	16.7	9.1	5.8	3.1	2.3	1.5
Dragon	OCT	11.4	8.5	4.9	1.5	0.8	N/A
(437,645)	FOLProM	4.2	2.2	1.5	0.9	0.7	0.6
Feline	OCT	27.5	17.2	10.5	5.0	1.8	0.7
(49,864)	FOLProM	16.7	9.1	6.1	2.9	1.9	1.2
Horse	OCT	33.82	20.3	14.9	4.3	1.9	1.0
(19,851)	FOLProM	16.0	10.3	5.3	3.4	2.0	1.3
Igea	OCT	20.2	15.6	10.1	2.7	0.9	N/A
(134,345)	FOLProM	4.6	2.9	2.1	1.4	1.2	0.7
Maple	OCT	29.8	18.6	13.8	11.3	6.1	4.0
(45,499)	FOLProM	21.5	13.6	8.4	5.4	3.7	2.8
Rabbit	OCT	20.2	15.0	9.6	2.7	1.1	N/A
(67,039)	FOLProM	5.2	3.9	2.1	1.6	1.2	0.5
Skeleton_hand	OCT	10.3	6.1	3.0	1.4	N/A	N/A
(327,323)	FOLProM	3.4	2.5	1.2	0.9	0.6	0.5

Table 1 shows the R-D results of the OCT mesh coder [PK05a] and the proposed FOLProM coder applied to various models.

This table gives the mean error (as measured with METRO) of each input mesh when coded at a sequence of bitrates using a specific mesh encoder (OCT or FOLProM), where the distortion is reported on a scale of 10^{-4} with respect to the diagonal of the bounding box of the original mesh. An “N/A" in this table indicates unavailable data due to the early termination of OCT coding when the octree has expanded to the maximum level (determined by the quantization resolution). The bitrates listed are the total bitrates, including bitrates for both geometry coding and connectivity coding, expressed in bits-per-vertex with respect to the



(a) (b) (c) (d)

Figure 5: *The R-D performance curves for (a) Rabbit, (b) Igea, (c) Skeleton_hand and (d) Maple. Note that the Maple model is a non-manifold triangle soup.*

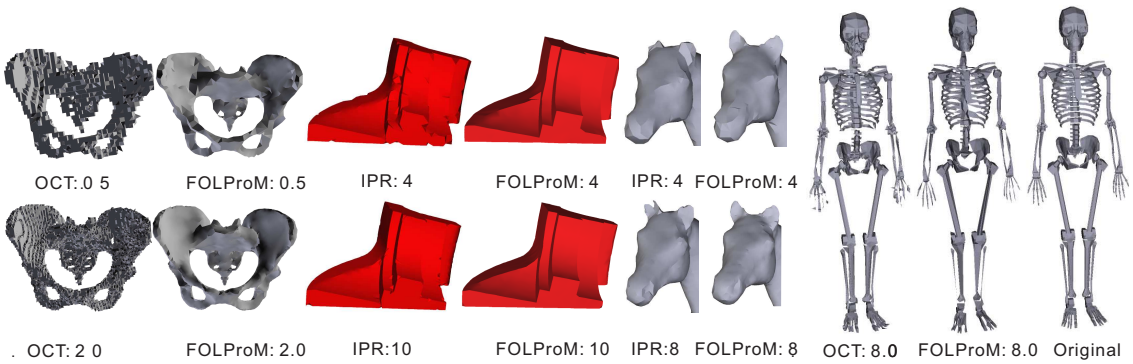
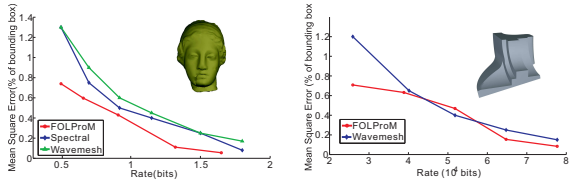


Figure 6: *Intermediate meshes for Donna, a non-manifold model, Fandisk, Horse and Skeleton, another non-manifold model, at different bitrates as indicated by bpv using OCT, IPR and FOLProM.*

total number of vertices in the original mesh. It can be seen that FOLProM has significant R-D advantage over OCT for all meshes, especially with large meshes, for which efficient compression is more critical. It is also observed that the R-D advantage of FOLProM for triangle soups (*e*.g., Maple) is not as sharp as that for meshes with significant manifold components. The reason may be that the techniques like GLA partition, adaptive offset quantization, and offset prediction in FOLProM, although is applicable for non-manifolds meshes also, are most effective when the local surface is manifold and smooth. It is worthwhile to point out that Donna is a non-manifold model with dominant manifold components. From table 1, we see a clearly better R-D performance of FOLProM than that of OCT for Donna, and Fig. 6 shows the visual comparison using one of the intermediate decoded meshes of Donna. We also illustrate the R-D performance gain of FOLProM over OCT on other models in Fig. 5.

We compare with the wavelet-based mesh coder(Wavemesh) [VP04] in Fig. 7, and with the spectral geometry coder(Spectral) [KG00a] in Fig. 7(a). We demonstrate in Fig. 7(a) the R-D advantage of FOLProM over Wavemesh and Spectral. In Fig. 7(a), the R-D curves of the three mesh coders on the Venus Head model (with 8,268 vertices) are plotted. We measure, for FOLProM, the distortion of an intermediate mesh by its mean square error (using METRO) to the original mesh divided by the diagonal of the original mesh's bounding box. The R-D data for Wavemesh and Spectral were extracted from Fig.8 in [VP04]. The R-D advantage of FOLProM is clearly demonstrated in Fig. 7(a). Fig. 7(b) shows the R-D curves for FOLProM and Wavemesh on Fandisk (with 6,475 vertices), where we observe a clearly better performance of FOLProM over Wavemesh in the low and the high ends of bitrates while Wavemesh performs slightly better in the middle. We are not able to compare with [KG00a] on Fandisk since its coding performance on Fandisk was not reported in [VP04]. It is worthwhile to point out that Wavemesh is suitable only for manifold mesh compression.

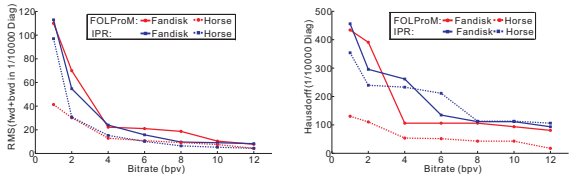
In Fig. 8, we compare the R-D performance with the incremental-parametric-refinement-based mesh coder(IPR) [VCP09] using the RMS and the Hausdorff distance in Fig. 8(a) and Fig. 8(b), respectively. In this comparison we use two models – Horse and Fandisk. From Fig. 8(a), we see a comparable performance between IPR and FOLProM when RMS is adopted as the distortion metric. However, in terms of Hausdorff distortion, FOLProM achieves a significantly better performance than IPR,



(a) (b)

Figure 7: *Comparison of R-D curves for Venus Head (with 8,268 vertices) and Fandisk (with 6,475 vertices).*

as illustrated in Fig. 8(b). This can be explained by the fact that FOLProM tries to preserve salient features even at the low end of bitrates, which accounts for the sharp reduction in Hausdorff distortion when compared with IPR. The effect of the reduced Hausdorff will be visually illustrated in Section 4.2.



(a) (b)

Figure 8: *Comparison of R-D curves for Horse (with 19,851 vertices) and Fandisk (with 6,475 vertices). Although measuring the error with two different metrics show different differences in the methods, the visual comparison as shown in Fig.6 clearly shows that FOLProM performs better in preserving geometric features at low bit rates.*

Total bitrates when the connectivity is fully restored are listed in Table 2 for several models with the settings of our experiments. It should be noted that beyond the complete restoration of connectivity, the vertex positions can still be iteratively refined through bounding volume subdivision.

Table 2: *Total bitrates (in bpv) when connectivity is fully restored.*

Model	Venus Head	Horse	Feline	Rabbit	Igea
Bitrate	18.7	15.8	17.4	15.4	15.8

4.2. Visual Performance Comparison

To demonstrate the superior visual quality of decoded intermediate meshes of the FOLProM coder at low bitrates, several LODs for Donna, Fandisk, Horse and Skeleton at different bitrates are shown in Fig. 6, where the coder and the bitrates are marked below each reconstructed mesh. We see from these figures that OCT suffers from the aliasing artifact while FOLProM yields much better visual quality, especially at low bitrates. It is worthwhile to point out that Donna and Skeleton are both non-manifold models.

Fig. 8 shows the R-D comparison and Fig. 6 show visual comparison between the results of IPR [VCP09] and FolPROM. In particular, we see that the sharp features in the models including ears in Horse and the edges and corners in Fandisk are well preserved under FolPROM especially at low bit rates. It should also be noted that IPR [VCP09] can compress only manifold models while FolPROM can compress models with arbitrarily complex topology.

4.3. Timing Statistics

Table 3: *Timing statistics for encoding and decoding of selected models. The data are based on 16bpv encoding and decoding.*

	Horse (19,851)	Feline (49,864)	Donna (50,691)	Rabbit (67,039)	Igea (134,345)
Encoding	0.9s	2.2s	2.2s	4.7s	7.6s
Decoding	0.5s	1.1s	1.1s	2.2s	3.8s

Timing statistics for the encoding and the decoding of selected models are reported in Table 3. The data are collected by running our program on a Laptop platform of Sony(R) VAIO Z55 with Intel(R) Core(TM)2 Duo CPU P8800 @ 2.66GHz 2.67GHz and 4.00GB RAM, and the data are based on 16bpv encoding and decoding. The time taken for encoding and decoding are roughly proportional to the number of vertices in the original mesh. It should be noted that our research code for the encoder and the decoder is not yet optimized for running efficiency.

5. Conclusion and Future Work

A feature-oriented progressive lossless mesh coder is proposed, which can encode arbitrarily complex triangular meshes, even triangle soups. The coding scheme is driven by the process of iterative vertex set split and/or bounding volume subdivision, which strives to preserve geometric features and minimize surface distortion in any intermediate mesh. In addition, novel techniques have been proposed to reduce the data entropy. As a result, the proposed mesh coder achieves a significant R-D gain over the prior art and is particularly effective in preserving sharp features at the low end of bitrates.

As to future extension, it is worthwhile to try bottom-up hierarchical construction as in [GWH01]. It is also interesting to investigate the issue of optimal bit allocation between coding more vertices and coding more bits per vertex in any intermediate mesh, as discussed in [KR99]. In addition, we may investigate wavelet coding of the child offsets using

well-designed local shape descriptors as contexts. Furthermore, we may try coding the original floating-point vertex coordinates with minor adaption of our current work.

References

[AD01a] ALLIEZ P., DESBRUN M.: Progressive encoding for lossless transmission of triangle meshes. In *ACM SIGGRAPH* (2001), pp. 198–205.

[AD01b] ALLIEZ P., DESBRUN M.: Valence-driven connectivity encoding for 3D meshes. In *EUROGRAPHICS* (2001), pp. 480–489.

[AFSR03] ATTENE M., FALCIDIENO B., SPAGNUOLO M., ROSSIGNAC J.: Swingwrapper: Retiling triangle meshes for better edgebreaker compression. *ACM Transactions on Graphics 22*, 4 (2003), 982–996.

[AG03] ALLIEZ P., GOTSMAN C.: Recent advances in compression of 3d meshes. In *Proc. of the Symp. on Multiresolution in Geometric Modeling* (Sep 2003).

[BPZ99a] BAJAJ C., PASCUCCI V., ZHUANG G.: Progressive compression and transmission of arbitrary triangular meshes. In *IEEE Visualization* (1999), pp. 307–316.

[BPZ99b] BAJAJ C. L., PASCUCCI V., ZHUANG G.: Single resolution compression of arbitrary triangular meshes with properties. *Computational Geometry: Theory and Applications 14* (1999), 167–186.

[COLR99] COHEN-OR D., LEVIN D., REMEZ O.: Progressive compression of arbitrary triangular meshes. In *IEEE Visualization* (1999), pp. 67–72.

[CR04] COORS V., ROSSIGNAC J.: Delphi: geometry-based connectivity prediction in triangle mesh compression. *The Visual Computer 20*, 8-9 (2004), 507–520.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Comp. Graph. Forum 17*, 2 (1998), 167–174.

[DG00] DEVILLERS O., GANDOIN P.: Geometric compression for interactive transmission. In *IEEE Visualization* (2000), pp. 319–326.

[GD02] GANDOIN P. M., DEVILLERS O.: Progressive lossless compression of arbitrary simplicial complexes. *ACM Trans. Graphics 21*, 3 (2002), 372–379.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *ACM SIGGRAPH* (2002), pp. 355–361.

[GGK02] GOTSMAN C., GUMHOLD S., KOBBELT L.: Simplification and compression of 3D meshes. In *Tutorials on Multiresolution in Geometric Modelling* (2002).

[GP05] GABRIEL PEYRÉ S. M.: Surface compression with geometric bandelets. In *ACM SIGGRAPH* (2005), pp. 601–608.

[GS98] GUMHOLD S., STRASSER W.: Real time compression of triangle mesh connectivity. In *ACM SIGGRAPH* (1998), pp. 133–140.

[GWH01] GARLAND M., WILLMOTT A., HECKBERT P. S.: Hierarchical face clustering on polygonal surfaces. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), pp. 49–58.

[Hop96] HOPPE H.: Progressive meshes. In *ACM SIGGRAPH* (1996), pp. 99–108.

[ILS04] ISENBURG M., LINDSTROM P., SNOEYINK J.: Lossless compression of floating-point geometry. In *Proceedings of CAD'3D* (2004), pp. 1–4.

[ILS05] ISENBURG M., LINDSTROM P., SNOEYINK J.: Lossless compression of predicted floating-point geometry. *Computer-Aided Design 37*, 8 (2005), 869–877.

[KBG02] KARNI Z., BOGOMJAKOV A., GOTSMAN C.: Efficient compression and rendering of multi-resolution meshes. In *VIS '02: Proceedings of the conference on Visualization '02* (2002), pp. 347–354.

[KG00a] KARNI Z., GOTSMAN C.: Spectral compression of mesh geometry. In *ACM SIGGRAPH* (2000), pp. 279–286.

[KG00b] KHODAKOVSKY A., GUSKOV I.: Normal mesh compression. *Preprint* (2000).

[KR99] KING D., ROSSIGNAC J.: Optimal bit allocation in compressed 3d models. *Journal of Computational Geometry, Theory and Applications 14*, 1-3 (1999), 91–118.

[KSS00] KHODAKOVSKY A., SCHRÖDER P., SWELDENS W.: Progressive geometry compression. In *ACM SIGGRAPH* (2000), pp. 271–278.

[LAD02] LEE H., ALLIEZ P., DESBRUN M.: Angle-analyzer: A triangle-quad mesh codec. In *EUROGRAPHICS* (2002), pp. 383–392.

[LK98] LI J., KUO C.-C. J.: Progressive coding of 3-D graphic models. *Proceeding of the IEEE 86*, 6 (Jun 1998), 1052–1063.

[PEK06] PENG J., ECKSTEIN I., KUO C.-C. J.: A novel and efficient progressive lossless mesh coder. In *SIGGRAPH Sketches* (2006).

[PH97] POPOVIC J., HOPPE H.: Progressive simplicial complexes. In *ACM SIGGRAPH* (1997), pp. 217–224.

[PK05a] PENG J., KUO C.-C. J.: Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. In *ACM SIGGRAPH* (2005), pp. 609–616.

[PK05b] PENG J., KUO C.-C. J.: Progressive geometry coding of 3d meshes using hierarchical vertex set split. In *SPIE Applications of Digital Image Processing XXVIII* (2005), pp. 340–348.

[PKK05] PENG J., KIM C.-S., KUO C.-C. J.: Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation 16*, 6 (2005), 688–733.

[PR00] PAJAROLA R., ROSSIGNAC J.: Compressed progressive meshes. *IEEE Trans. Visualization and Comp. Graphics 6*, 1 (2000), 79–93.

[Ros99] ROSSIGNAC J.: Edgebreaker: Connectivity compression for triangle meshes. *IEEE Trans. Visualization and Comp. Graphics 5*, 1 (1999), 47–61.

[SRK02] SZYMCZAK A., ROSSIGNAC J., KING D.: Piecewise regular meshes: construction and compression. *Graphical Models 64*, 3/4 (2002), 183–198.

[TG98] TOUMA C., GOTSMAN C.: Triangle mesh compression. In *Proceedings of Graphics Interface* (1998), pp. 26–34.

[TGHL98] TAUBIN G., GUEZIEC A., HORN W., LAZARUS F.: Progressive forest split compression. In *ACM SIGGRAPH* (1998), vol. 32, pp. 123–132.

[TR98] TAUBIN G., ROSSIGNAC J.: Geometric compression through topological surgery. *ACM Trans. Graphics 17*, 2 (1998), 84–115.

[VCP09] VALETTE S., CHAINE R., PROST R.: Progressive lossless mesh compression via incremental parametric refinement. *Computer Graphics Forum 28*, 5 (2009), 1301–1310.

[VP04] VALETTE S., PROST R.: Wavelet-based progressive compression scheme for triangle meshes: Wavemesh. *IEEE Transactions on Visualization and Computer Graphics 10*, 2 (2004), 123–129.