Available online at www.sciencedirect.com
ScienceDirect
J. Vis. Commun. Image R. 18 (2007) 341–353
JOURNAL OF VISUAL Communication & IMAGE Representation
www.elsevier.com/locate/jvci

Predictive vector quantization of 3-D mesh geometry by representation of vertices in local coordinate systems ☆

Ulug Bayazit [a,*], Ozgur Orcay [b], Umut Konur [c], Fikret S. Gurgen [c]

[a] *Electronics Engineering Department, Isik University, Istanbul, Turkey*
[b] *Computer Engineering Department, Dogus University, Istanbul, Turkey*
[c] *Computer Engineering Department, Bogazici University, Istanbul, Turkey*

Received 7 August 2006; accepted 14 March 2007
Available online 24 March 2007

Abstract

In predictive 3-D mesh geometry coding, the position of each vertex is predicted from the previously coded neighboring vertices and the resultant prediction error vectors are coded. In this work, the prediction error vectors are represented in a local coordinate system in order to cluster them around a subset of a 2-D planar subspace and thereby increase block coding efficiency. Alphabet entropy constrained vector quantization (AECVQ) of Rao and Pearlman is preferred to the previously employed minimum distortion vector quantization (MDVQ) for block coding the prediction error vectors with high coding efficiency and low implementation complexity. Estimation and compensation of the bias in the parallelogram prediction rule and partial adaptation of the AECVQ codebook to the encoded vector source by normalization using source statistics, are the other salient features of the proposed coding system. Experimental results verify the advantage of the use of the local coordinate system over the global one. The visual error of the proposed coding system is lower than the predictive coding method of Touma and Gotsman especially at low rates, and lower than the spectral coding method of Karni and Gotsman at medium-to-high rates.

Keywords: Mesh geometry compression; Entropy constrained vector quantization; Local coordinate system; Shannon lower bound; Parallelogram prediction

1. Introduction

Within the last 10 years, general and restricted access to 3-D objects over communication networks for visualization purposes has gained widespread interest. Polygonal meshes are the primary representation tools used for the visualization of 3-D objects in the various application areas such as manufacturing, entertainment, defense industry, computer aided design and architecture. Recent multimedia standards such as VRML (Virtual Reality Markup Language) and MPEG-4 (Motion Pictures Expert Group-4) have embodied polygonal mesh coding and representation tools.

Typically, the encoding of polygonal meshes is performed in three distinct parts: connectivity coding concisely represents the topological information about the arrangements and relations of faces and edges interconnecting the vertices of a mesh. Geometry (vertex coordinate) coding concisely represents the coordinate values of the vertices. Property coding describes features such as texture coordinates and material attributes. A comprehensive review of the recent developments in geometry and connectivity coding may be found in [2].

☆ This work was partially supported by and carried out under Project No. 103E004 of TUBITAK (The Scientific & Technological Research Council of Turkey).
* Corresponding author.
E-mail addresses: bayazit@isikun.edu.tr (U. Bayazit), oorcay@dogus.edu.tr (O. Orcay), konur@boun.edu.tr (U. Konur), gurgen@boun.edu.tr (F.S. Gurgen).

1047-3203/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2007.03.001

1.1. Previous work in geometry coding and related connectivity coding

Considerably more bits are needed to losslessly represent the vertex coordinate values of a polygonal mesh than to losslessly represent its connectivity. Despite this fact, most prominent earlier mesh compression methods [11–17] have emphasized connectivity coding. The state of the art in connectivity coding can compress most meshes down to the range of 1–3 bits/vertex.

The earliest notable work of [11] on lossy geometry compression employs simple linear prediction to exploit the correlation between the neighboring vertex coordinate values. The coordinates are prequantized to 16 bits precision and their delta differences are entropy coded at 10 bits/coordinate resolution to achieve 5.5–7.5 bits/coordinate rate. Since such rates were still fairly significant with respect to the lossless compression rates in connectivity coding, a need emerged for more advanced lossy geometry compression schemes.

More advanced linear prediction schemes of [12,15,22] form the prediction as a linear combination of previously coded vertices. The widely popular parallelogram prediction rule of [15,22] forms the prediction as the fourth corner of a parallelogram with the other three being the three previously coded vertices. In [15], this prediction is corrected by a curvature estimate. In a local coordinate system similar to the one employed here, [23] derives a higher order prediction for the normal component from the tangential components. Linear prediction is also employed in the geometry compression methods of [13,14,24] for progressive transmission of multiresolution meshes. The current work adopts the parallelogram prediction rule of [15,22] for single resolution polygonal mesh coding.

In predictive coding of the vertex coordinates, the encoder and decoder need to agree on a vertex traversal order. The order of vertex processing in connectivity coding is commonly used for this purpose. Single resolution triangular mesh connectivity coding is usually based on a region growing technique [12,15–17] that inserts a triangle into a processed region at each generic step. Generalizations to polygonal meshes may be found in [18–20]. The approach of [18] determines the optimal location on the boundary of the processed region for inserting the face and codes the face's degree and the valences of its new vertices. Without loss of generality to adopting other approaches, the current work adopts the approach of [18] for determining the vertex traversal order.

With the advent of [25,29], transform based methods gained acceptance in mesh geometry coding. The spectral transform in [25], that generalizes the classical 1-D Fourier transform to 3-D irregular meshes, is derived from mesh topology. Compression is achieved by discarding the spectral coefficients that are assumed to have low energy. Notable extensions for reducing the implementation complexity and improving the visual quality of [25] are presented in [26] and [27,28], respectively.

State-of-the-art wavelet transform methods of [29,30,33,36] treat geometry compression of densely sampled meshes as a surface approximation problem. They employ a multi-level representation by changing irregular connectivity to subdivision connectivity. An original irregular mesh is first simplified to yield a base level coarse mesh. Starting with the base level mesh, the highly complex iterative process of subdivision of the mesh at a coarse level to arrive at the mesh at a finer level, yields the desired resolution mesh at the end.

In normal meshes [33,40], a base point is predicted by subdivision using vertices in the coarser level, and a line drawn in a normal direction from the base point is intersected with the original surface to yield a finer level vertex. Most finer level vertices are expressed in this manner with a single (normal) coordinate in a local coordinate system. Normal and tangential coordinates are compressed by advanced coding tools such as zerotrees and the estimation/quantization algorithm, originally proposed by [31,32], that can exploit the correlation among them.

Vector quantization (VQ) has also been investigated in [1,8–10] within the context of 3-D lossy predictive geometry coding. Predictive VQ yields good compression performance at medium to high coding rates by exploiting the statistical dependencies among the components of the vertex prediction error vector. Additionally, the mapping of the prediction error vectors to the channel indices by the VQ encoder is very suitable for parallel hardware implementation and the mapping of these indices to the reconstruction vectors by the VQ decoder requires low computational complexity. Predictive VQ may be preferred to transform based coding in applications where low complexity is desired along with high reconstruction fidelity.

In the predictive VQ methods of [8] and [9], the distortion is minimized without a rate constraint. Both [8] and [9] address the mismatch problem of the actual prediction error vectors to the codebooks designed in an open loop way by using the original vertices in prediction. In [8], the prediction error vectors (model space vectors) are represented in the local coordinate system and compressed using minimum distortion vector quantization (MDVQ). The codebook is transmitted as overhead. To prevent error overflows due to quantization distortion accumulation in predictive coding, [8] inserts correction terms into the bitstream which are scalar quantized normalized delta differences of parallelogram prediction errors. In [9], codebook design is a closed loop iterative process, where in each pass, the predictions are based on the reconstructed vertices from the previous pass and the designed codebook is, in turn, applied to the quantization of the prediction error vectors used in its design. Assuming that this more optimal quantization of the prediction error vectors yields better prediction performance, the convergence of the iterations is guaranteed. Low complexity pyramid VQ, which saves from the codebook overhead rate, is also investigated in [9] for geometry coding.

Unlike [8] and [9], the proposed VQ method incorporates a rate constraint. A low complexity variant of the high performance Alphabet Entropy Constrained Vector Quantization (AECVQ) method of [5] is employed in this work. The universally designed codebook is partially adapted to the prediction error vector source by the normalization of the prediction error vectors prior to codebook design and encoding. This approach is simpler to design and implement, and more efficient than the fully adaptive codebook design approach of [6,9].

1.2. The local coordinate system

The primary contribution of the current work is the utilization of a local coordinate system for efficient vector quantization of the prediction error vectors in a predictive VQ based geometry compression system. The local coordinate system clusters these vectors around two quadrants of a 2-D planar subspace. As a result, their components can be more efficiently block coded.

The parallelogram prediction is ineffective in certain cases. Across highly non-convex or non-planar triangle pairs, the multi-way prediction of each vertex [41], and in high degree polygon dominated meshes, the high degree polygonal vertex prediction by setting the highest Fourier frequency to zero [42], were shown to improve prediction performance. As a secondary contribution, the current work proposes a derivative descent technique in the local coordinate system for low complexity estimation (and compensation) of the systematic bias in the parallelogram prediction rule for triangular mesh coding.

The proposed geometry compression system is depicted in Fig. 1. Since the transformation for each vertex is derived from the previously decoded vertices, no side information about the transformation is transmitted from the encoder to the decoder. On the other hand, the bias, estimated by the encoder from the original vertex data, is transmitted as side information (not shown in the figure) to the decoder.

The local coordinate system in [8] bears some resemblance to the one proposed in this paper. The vertex at the far end of the triangle, that is a neighbor of the new triangle, is used as the prediction. Since the two edges incident on this vertex define the basis vectors of the coordinate system of [8], VQ in this coordinate system simulates parallelogram prediction. However, the non-orthogonality of the coordinate system counteracts the decorrelation goal of efficient compression schemes. Hence, the rate-distortion performance reported in [8] is often short of that of [15].

Similar local coordinate systems have also been explored in [23,37–40] to boost geometry compression performance. In [23], the local coordinate system is used to estimate the bending angle for an improved prediction. In [38], the vector between the new vertex and the prediction taken to be one of the two focus (gate) vertices (the origin of the local coordinate system) is encoded. This prediction is likely to be inferior to parallelogram prediction which exploits the

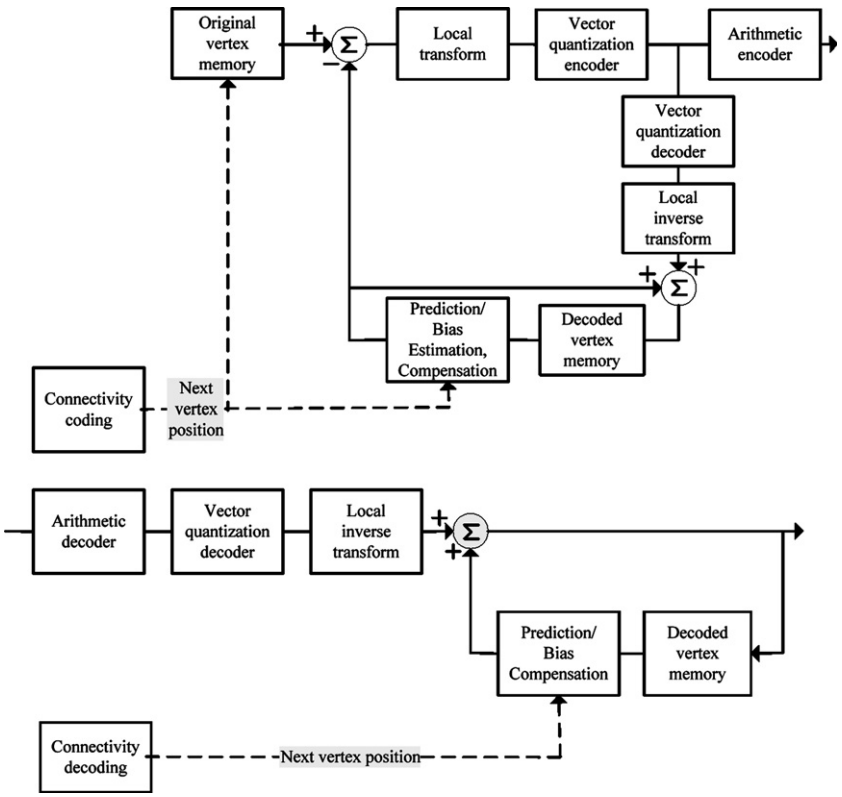

Fig. 1. The geometry encoder (top) and the geometry decoder (bottom).

correlations between the corresponding dimensions of the adjacent triangles. In the progressive coder of [39], the vector formed between the barycenter of a previously coded patch and the new vertex in the middle of the patch is coded. Similarly, the multiresolution mesh generation of [40] represents finer level vertices as 1-D displacements along the direction of normals from the coarser level faces. In the local coordinate system employed in [37] for tetrahedral meshes, the z-axis is normal to the gate triangle of the processed volume, and the new vertex of a tetrahedron bordering the gate triangle is predicted.

Next section outlines the connectivity coding algorithm of [18] used to determine the vertex traversal order during geometry coding. After a review of the parallelogram prediction rule of [15,22], Section 3 covers the proposed geometry coding system. The local coordinate transformation is described and the related clustering of the prediction error vectors is explained in Section 3.1. Section 3.2 explains the rate-distortion theoretic performance advantage of block coding in a local coordinate system. Section 3.3 gives implementation details of AECVQ. Section 3.4 discusses the bias in the parallelogram prediction rule along the y-axis direction of the local coordinate system and its estimation and compensation for a higher coding gain. In Section 4, we present simulation results for performance assessment of the proposed ideas. Here, computational complexities of the key components of the proposed method are also reported.

2. Connectivity coding

In the region growing approach of [18], the connectivity coding approach of [15] for triangular meshes is generalized to polygonal meshes. The encoder and the decoder of the proposed method use its vertex traversal order for coding vertex valences to synchronize the predictive coding and decoding of the vertex coordinates. For completeness, we present it here.

At each generic step, the algorithm of [18] inserts a new face to one of one or more connected processed regions. The region, to which the new face is inserted, is bounded by a list of vertices termed the active boundary. The boundaries of those connected regions that are not currently grown reside in a stack. The edge or the set of edges on the active boundary, that defines the border between the new face and the processed region to which it is inserted, is called the focus. After the new face is inserted on the focus, its degree is coded, and the valences of its previously uncoded vertices are coded in a clockwise order where the last traversed edge of the inserted face becomes the exit focus. If the exit focus has a zero free valence vertex on one end, it is *widened* on that end. Otherwise, if one of the two edges neighboring the exit focus on the active boundary has smaller free valence vertices on its ends, the exit focus is moved to that edge. A more elaborate decision is made in [18] to move the exit focus.

As a result of the face insertion, the active boundary may lead to a previously encoded vertex on the active boundary, or a boundary in the stack. In the first case, if no more faces can be inserted into that region, the boundary from the top of the stack is popped and becomes the new active boundary. Otherwise, the boundary is split into two. One boundary is pushed onto the stack and the other is designated the new active boundary. In the second case, the active boundary is merged with the boundary in the stack.

The coordinates of the previously uncoded vertices of a new face are predictively coded in the same clockwise order as their valences. Fig. 2 illustrates the main steps of the region growing algorithm.

3. The geometry compression system

In linear predictive coding of vertex coordinates, a prediction vector for the currently coded vertex is formed as a linear combination of the previously coded vertices, the prediction error vector components are quantized and the resultant quantization levels are entropy coded.

Predictive coding is susceptible to prediction error drift and loss of tracking at the decoder in the presence of channel errors. Nevertheless, linear predictive coding is widely employed in geometry coding to achieve high compression efficiency by exploiting the high correlations between the values of the corresponding coordinates of neighboring vertices.

The parallelogram prediction of vertex v_a is formed as the vector of coordinates

$$\hat{\tilde{v}}_a = \tilde{v}_b + \tilde{v}_c - \tilde{v}_d \tag{1}$$

where $\tilde{v}_i$, $i \in \{b, c, d\}$, denotes the vector of reconstructed coordinate values of the ith vertex used in the prediction of v_a. The notation $\hat{\tilde{v}}_a$ emphasizes the dependence of the prediction on the reconstructed rather than the original

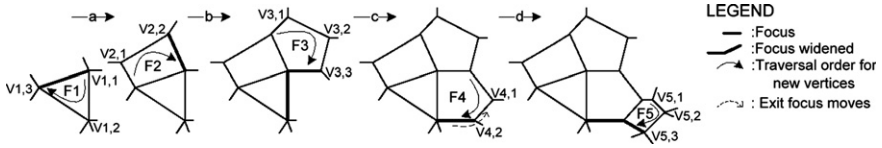

Fig. 2. Region growing algorithm steps: each new face is added on the focus by coding its degree and the valences of its vertices in a clockwise order. Exit focus becomes the last edge. If one end of the focus is a zero free valence vertex as in (c) it is widened on that end. If either neighboring edge on the active boundary has ends at vertices with fewer free valences then exit focus moves to that edge as in (d).

vertices. If only $\tilde{v}_b$ and $\tilde{v}_c$ are the previously coded vertices of the new face, then $\tilde{v}_d$ is the first previously coded vertex in the clockwise direction from the edge $\tilde{v}_b$ and $\tilde{v}_c$ in a previously coded neighboring face. This operation is termed *across prediction* [22]. Otherwise, $\tilde{v}_d$ is between $\tilde{v}_b$ and $\tilde{v}_c$ on the new face. In this case, the operation is termed *within prediction* [22]. The first coded vertex of a new face is either across predicted or within predicted (focus widened), while its other coded vertices are within predicted. Fig. 3 shows each type of prediction.

The parallelogram rule works efficiently, if the number of vertices on a face is three, in the case of across prediction, and four, in the case of within prediction. Another implicit assumption is that the vertices used in prediction and the predicted vertex are coplanar. In general, this is not true due to the surface normal component of the quantization noise in the reconstructed vertices used in prediction and the crease angle between adjacent faces (surface curvature).

3.1. The local coordinate transformation

The first step of the local coordinate transformation is the determination of a surface normal vector. This vector can be determined as the cross product of the two vectors between the three vertices used in prediction. In polygonal meshes, where more previously coded vertices of a face may be available, least squares estimation of the surface normal vector of the plane that best fits these vertices yields better coding performance.

Once the surface normal vector is estimated, two rotations that first map the surface normal vector to a vector in the x–z plane and then map this vector to a vector on the z-axis of the global coordinate system are determined. Fig. 4 illustrates these rotations. In the same order, the two rotations are then applied to all the vertices used in prediction as well as the predicted vertex to place them in a new coordinate system.

For within prediction in a polygonal face, the direct application of Eq. (1) in the new coordinate system could be suboptimal, since it ignores the fact that most mesh generation applications tend to generate planar faces. In this case, only the x–y components of the prediction should

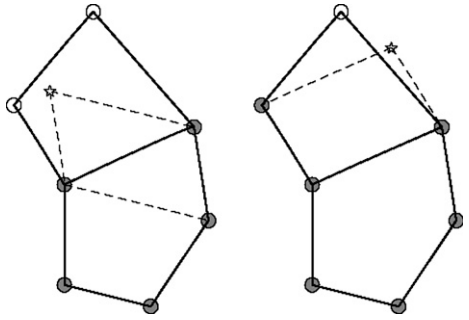

Fig. 3. Left: across prediction. Right: within prediction.

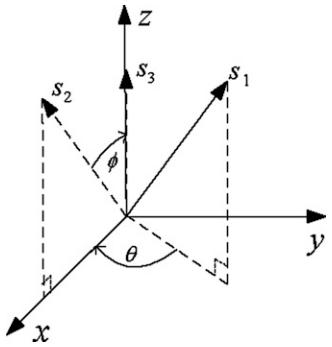

Fig. 4. Sample vector s_1 is rotated by θ degrees to yield s_2 on the x–z plane which is further rotated by ϕ degrees to yield s_3 on the z-axis of the global coordinate system.

be formed by the application of the parallelogram rule in a 2-D planar subspace. To see why, consider the situation in which the realizations of the zero mean random variable for the surface normal component of the quantization error vector are such that $q_{b,z} > 0$, $q_{c,z} > 0$, and $q_{d,z} < 0$. In this case, the direct application of Eq. (1) results in a cumulative error vector of $q_a = q_b + q_c - q_d$ with respect to $\hat{v}_a = v_b + v_c - v_d$ so that $\hat{\tilde{v}}_a = \hat{v}_a + q_a$. The surface normal component of q_a is $q_{a,z} = q_{b,z} + q_{c,z} - q_{d,z} > 0$. However, the best estimate for the surface normal component should be $E[q_{a,z}] = E[q_{i,z}] = 0$ if we assume that the predicted vertex is coplanar with the vertices used in prediction. This implies that, in the new coordinate system, the z-component of the prediction should be set equal to the z-intercept of the least squares estimated plane subjected to the above rotations rather than the value predicted by the parallelogram rule.

The prediction error vector is found by subtracting the prediction vector from the predicted vertex in the new coordinate system. When the vertices used to form the prediction as well as the predicted vertex are coplanar, the z-component of the prediction error vector in the new coordinate system is zero and does not have to be coded. In general, due to the two reasons stated above, this component has a nonzero variance. However, this variance is smaller than the variances of each of the other two components for high resolution mesh models with largely smooth surfaces. Effectively, the prediction error vectors are clustered around the x–y plane.

Consider a hypothetical flat local mesh surface with adjacent triangles forming parallelograms. In this case, the variance of each prediction error vector component is the sum of the variances of the quantization errors of the corresponding coordinates of the three reconstructed vertices used in prediction assuming independence of the quantization errors in reconstructed vertices. In a real model, any curvature of the local surface increases the variance of the z-component, and any deviation from the ideal parallelogram geometry increases the variances of the x–y components beyond the hypothetical case. However, deviations from ideal parallelogram geometry are much more

Table 1
The variances of prediction error vector components represented in the global and local coordinate systems (obtained with MDVQ coding with 8192 codevectors)

	Bunny (34834 vert.)	Horse (19851 vert.)	Venus (50002 vert.)
Local w/o bias comp.			
σ_X^2	5.921e − 8	4.512e − 7	9.421e − 6
σ_Y^2	2.721e − 7	1.780e − 6	1.999e − 5
σ_Z^2	3.769e − 8	5.475e − 8	6.978e − 7
γ^2	0.6885	0.4632	0.5066
Local with bias comp.			
σ_X^2	5.835e − 8	4.544e − 7	9.384e − 6
σ_Y^2	1.617e − 7	3.975e − 7	1.129e − 5
σ_Z^2	3.737e − 8	5.405e − 8	6.700e − 7
γ^2	0.8233	0.7077	0.5820
Global			
σ_X^2	1.095e − 7	4.940e − 7	8.310e − 6
σ_Y^2	9.173e − 8	4.145e − 7	1.185e − 5
σ_Z^2	9.707e − 8	4.707e − 7	1.020e − 5
γ^2	0.9973	0.9973	0.9896

pronounced than the crease angle between adjacent faces for a high resolution model with a largely smooth surface. Therefore, one can expect much larger variances for the x or y components than for the z-component. Table 1 verifies this claim on several standard triangular mesh models.

For further clustering of the prediction error vectors, a translation and a third rotation are applied so that the imaginary line connecting the vertices $\tilde{v}_b$, $\tilde{v}_c$ used in prediction is aligned with the y-axis, each vertex is equidistant from the x-axis and the third vertex $\tilde{v}_d$ used in prediction has a negative x-coordinate. Since the crease angles between faces are highly unlikely to be larger than 90°, the predicted vertex is not likely to have a negative x-coordinate if the quantization errors in the coordinates of the vertices used in prediction are not large. We term the overall operation of performing the three rotations plus the translation for each face *the local coordinate transformation*.

3.2. The rate-distortion advantage of the use of the local coordinate system

The prediction error vector represented in the global coordinate system exhibits an isotropic distribution with no component biased towards a higher or a lower variance than the other two. On the other hand, as discussed above, the z-component of the prediction error vector in the local coordinate system is biased to have a lower variance than the x or y components. In this section, we shall discuss how the nonuniformity in the variances of the components of the prediction error vector in the local coordinate system translates to a coding gain in the light of a rate-distortion performance bound derived using the Shannon lower bound [7] for Laplacian sources.

Prediction error vector components are commonly modeled by Laplacian pdfs. Even though an analytical closed form formulation for the rate-distortion function with squared error criterion for an i.i.d. source with a Laplacian pdf does not exist, Shannon lower bound [7, p.92] yields valuable insight into the performance of practical codecs. Let component ξ of the prediction error vector be modelled as a scalar source with variance σ_ξ^2. For distortion D_ξ in component ξ, the lower bound on rate may be written as

$$R_{L,\xi}(D_\xi) = \frac{1}{2}\log\left(\frac{e\sigma_\xi^2}{\pi D_\xi}\right), \quad 0 \leqslant D_\xi \leqslant \frac{e\sigma_\xi^2}{\pi} \tag{2}$$

Then, a lower bound on the average rate as a function of the component distortions may be obtained as

$$\begin{aligned} R_L(D_X, D_Y, D_Z) &= \frac{1}{3}\sum_{\xi;\xi\in X,Y,Z} R_{L,\xi}(D_\xi) \\ &= \frac{1}{2}\log\left(\frac{e}{\pi}\frac{\left(\prod_{\xi;\xi\in X,Y,Z}\sigma_\xi^2\right)^{\frac{1}{3}}}{\left(\prod_{\xi;\xi\in X,Y,Z} D_\xi\right)^{\frac{1}{3}}}\right) \qquad (3) \\ &\geqslant \frac{1}{2}\log\left(\frac{e}{\pi}\frac{\gamma^2\overline{\sigma}^2}{\frac{1}{3}\sum_{\xi;\xi\in X,Y,Z} D_\xi}\right) \qquad (4) \end{aligned}$$

where

$$\gamma^2 = \frac{\left(\prod_{\xi;\xi\in X,Y,Z}\sigma_\xi^2\right)^{\frac{1}{3}}}{\frac{1}{3}\sum_{\xi;\xi\in X,Y,Z}\sigma_\xi^2} \quad \text{and} \quad \overline{\sigma}^2 = \frac{1}{3}\sum_{\xi;\xi\in X,Y,Z}\sigma_\xi^2$$

The inequality follows from the relation between the geometric and arithmetic means. When $D_\xi = D$, $R_L(D_X, D_Y, D_Z)$ is minimized with respect to D_X, D_Y, D_Z over the constraint set $\frac{1}{3}\sum_{\xi;\xi\in X,Y,Z} D_\xi = D$ to yield the equality case of the inequality in Eq. (4). The quantity γ^2 assumes values in the interval $[0,1]$ and measures the nonuniformity of the component variances. The closer the component variances are to their arithmetic mean, $\overline{\sigma}^2 = \frac{1}{3}\sum_{\xi;\xi\in X,Y,Z}\sigma_\xi^2$, the closer γ^2 will be to 1.

Let us assume for a moment that the prediction error vector has zero mean. In this case, since the prediction error vector norm does not change as a result of the orthonormal local coordinate transformation, $\overline{\sigma}^2$ will be the same in the global and local coordinate systems. However, the γ^2 parameter is smaller for the local coordinate system where the component variances are nonuniform. Therefore, Eq. (4) suggests that, for a given average distortion D, the rate can be expected to be lower when the prediction error vectors are represented in the local coordinate system than in the global coordinate system.

Since the prediction error vector components are virtually memoryless, Eq. (2) with $D_\xi = D$ and $\sigma_X^2 > \sigma_Z^2$ and $\sigma_Y^2 > \sigma_Z^2$ suggests that these components may be scalar quantized by allocating fewer bits to the z-component than to each of the other two components. Vector quantization

of the prediction error vector, discussed in the next section, is an alternative strategy that can realize the advantage of block coding in the local coordinate system.

3.3. AECVQ and partial codebook adaptation

In ECVQ [4], the problem of minimization of distortion D subject to rate $R \leqslant R_{\max}$ is posed equivalent to the problem of minimization of the cost functional $J = D + \lambda\ R$. For a given λ, this minimization is achieved by joint optimization of the codebook of reproduction codevectors, source vector to codevector index mapping and entropy codeword length for each codevector index. The codebook for the nth entropy constraint λ_n is initialized with the final codebook for the (n-1)th entropy constraint λ_{n-1} ($\lambda_{n-1} < \lambda_n$).

In the AECVQ approach of [5], the codebook design process is considerably simplified with negligible coding performance loss by fixing the reproduction codevectors as MDVQ codevectors rather than optimizing them for the constraint parameter λ as in [4]. The iterative design in [5] jointly optimizes the source vector to codevector index mapping and the entropy codeword length for each index.

Since the source distribution is generally not available, the traditional design approaches of [4] and [5] involve iterative optimization of the components of the encoder by employing a set of training vectors. While this works fine for coding a (test) source whose distribution matches exactly the distribution from which the training vectors are drawn, any distribution mismatch between the training and test sources results in inferior performance at low coding rates. The reason for this is a fixed entropy constrained source vector to codevector index mapping designed using index probabilities estimated from the training source. For acceptable ECVQ or AECVQ performance at low coding rates, the index probabilities should be estimated from the encoded (test) source. To meet this requirement, the traditional design approaches would run the iterative optimizations over numerous encoding passes. Since this is undesirable, the proposed approach uses a single encoding pass.

Let $C = \{y_i: i = 1, \ldots, N\}$ be a MDVQ codebook of size N. In the proposed approach, the prediction error vector x is assigned to codevector y_k by the entropy constrained assignment rule according to

$$x \to y_k \quad \text{if } k = \arg\min_j [d(x, y_j) - \lambda \log_2 p(y_j)] \tag{5}$$

where $d(x, y_j)$ is taken as the Euclidean distance between x and y_j, and $-\log_2 p(y_k)$ is the optimal codeword length of codevector y_k. The index k is adaptively arithmetic coded [21].

The entropy codeword lengths can be estimated from codevector index frequencies. These frequencies are based on previously coded indices, so that the decoder can track these estimates without the need for the encoder to transmit them as side information. The index frequencies are initially set equal to 10 for the large models and 1 for the smaller models. The frequency of each coded index is incremented.

The designed codebook is partially adapted to the encoded source by normalization of the prediction error data prior to codebook design and encoding. Specifically, after their (small) mean vector is subtracted from the prediction error vectors, the resultant error vector components are divided by their standard deviation. After decoding, the reconstructed normalized prediction error vectors are inverse normalized. Since the statistics can be obtained exactly only after actual encoding, they are estimated in an initial pass where the predictions are based on original vertices. This partial codebook adaptation approach is much simpler to implement and more effective than the ACL (asymptotic closed loop) based iterative closed loop codebook design approach of [6,9] that matches the codebooks to the encoded prediction error vectors. In order for the ACL codebooks to match the encoded source, they need to be transmitted with an overhead rate which could be large for meshes with as many as 30000 vertices. On the other hand, the overhead rate for the statistics used in normalization for partial codebook adaptation is negligible.

Fig. 5 shows that for low as well as high rates, the prediction error vector clouds are covered well by the codevector clouds due to normalization.

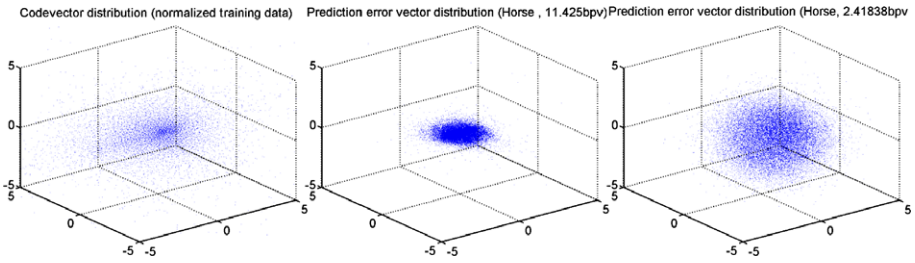

Fig. 5. Codevector distribution (left) fairly well covers the prediction error vector clouds at high (middle) as well as low (right) rates.

3.4. Exploiting the parallelogram prediction error vector bias

In this section, we discuss an effective technique to improve the parallelogram prediction rule in triangular mesh coding and the use of the local coordinate system for its computationally efficient implementation. Generalization to polygonal meshes is not straightforward and is not attempted in this work.

The parallelogram prediction rule carries a bias in the local y-coordinate value of the predicted vertex. From a large number of models, we note that the event of the actual local y-coordinate value exceeding the local y-coordinate value of the parallelogram prediction in absolute value has a probability significantly smaller than 0.5. This could be because the mesh generation applications tend to avoid the skinny triangles that result when this event happens. Skinny triangles could lead to degenerate triangles when the vertex coordinates are quantized. Based on the notion that relative positions of the vertices scale with the local geometry, a substantial reduction in the y-component variance of the prediction error vector can be obtained when the y-coordinate value is predicted as a fixed fraction of the value predicted by the parallelogram rule.

Let this new prediction for the y-coordinate value of the nth coded vertex be expressed as $(1+\delta_y)\hat{\tilde{v}}_{n,y}$ in the local coordinate system. Here $\delta_y\hat{\tilde{v}}_{n,y}$ is the correction to the parallelogram prediction. Note that the correction scales with the y-coordinate value of the parallelogram prediction. δ_y can be estimated to minimize the variance of the y-component of the prediction error vector. A fast, but effective method on large mesh models is to employ derivative descent whose generic update step is

$$\delta_y^{n+1} = \delta_y^n + \mu^n * \epsilon^n * \hat{\tilde{v}}_{n,y} \tag{6}$$

where ϵ^n is the final prediction error for the nth coded vertex due to the employment of $(1+\delta_y^n)\hat{\tilde{v}}_{n,y}$ as the most likely y-coordinate.

As with all gradient descent schemes, the choice for μ is not too critical and setting $\mu^n = \frac{0.001}{\frac{1}{n}\sum_{i=1}^{n}(v_{i,y})^2}$ ensures a fast enough convergence to the local minimum without oscillatory behaviour for a large set of tested models.

The bias is also estimated in the initial pass with predictions based on original vertices. In the second pass, δ_y^N, the estimate at the end of the first pass, is used to predict the y-coordinates of all vertices in the local coordinate system.

For the test models Bunny, Horse and Venus Head, the derivative descent scheme described above estimated significant bias values of $\delta_y = -0.3689$, $\delta_y = -0.9823$ and $\delta_y = -0.8694$, respectively, in the local y-coordinate. Again referring to Table 1, one notes that the compensation of the bias results in a considerable reduction in the y-component variance of the prediction error vector. The variances of the other two components are only slightly affected. Quite frequently this translates to a coding gain.

Since the bias is only in the local y-coordinate, the local coordinate system provides a convenient setting to estimate and compensate for this bias. On the other hand, bias estimation and compensation in the global coordinate system requires a projection of the parallelogram prediction onto the line connecting the vertices $\tilde{v}_b$ and $\tilde{v}_c$.

Below, we summarize the codebook design and encoding processes. The decoding process is very much similar to the decoding process except that the error vector is not computed.

AECVQ codebook design:

1. For each mesh in the training set {
   a. Initialize the bias to zero.
   b. For each vertex in the mesh, in the order dictated by connectivity coding {
      i. Determine the local coordinate system (l.c.s.):
         • Determine the normal vector by the cross product rule or the Least Squares algorithm using the *original* coordinates of other vertices of the same face.
      ii. Represent all vertices of the face in the l.c.s. by applying the two rotations
      iii. Form the initial prediction by the parallelogram rule using original vertices.
      iv. Modify the y-coordinate of the prediction by the bias and obtain the prediction error vector.
      v. Update the bias by the derivative descent rule of Eq. (6) using the y-component of the error vector and the initial prediction.

   }
   c. Normalize the prediction error vectors of the mesh by using their statistics.

   }
2. Design an MDVQ codebook by the LBG algorithm [3] using the normalized prediction error vectors.

Encoding of a test mesh:

1. Run step 1. of the design process on the test mesh to get its statistics and bias.
2. For each vertex in the test mesh,in the order dictated by connectivity coding {
   a. Determine the local coordinate system (l.c.s.):
      • Determine the normal vector by the cross product rule or the Least Squares algorithm using the *reconstructed* coordinates of other vertices of the same face.
   b. Represent all vertices of the face in the l.c.s. by applying the two rotations.
   c. Form the initial prediction by the parallelogram rule using reconstructed vertices.
   d. Modify the y-component of the prediction vector by the bias and obtain the error vector.

e. Normalize the error vector by using the statistics.
f. Encode the normalized error vector.
   - Apply the rate constrained mapping of Eq. (5) to determine the codevector.
   - Arithmetic encode the codevector index. Update the probability model of the arithmetic encoder.
g. Inverse normalize the codevector to get the reconstructed error vector.
h. Add the reconstructed error vector to the modified prediction vector to get the reconstructed vertex in the l.c.s.
i. Represent the reconstructed vertex in the global coordinate system by applying the inverse rotations.

}

4. Experimental results

Primary purpose of the simulations is to compare the VQ rate-distortion performance in the global and the local coordinate systems. Of secondary interest is the advantage of the use of AECVQ over MDVQ, partial codebook adaptation by normalization over ACL based full codebook adaptation, and the advantage of estimation and compensation of parallelogram prediction bias for triangular mesh models.

Let the vector of smoothness values of the vertex coordinates be specified by the Laplacian $S(v_i) = v_i - \frac{\sum_{j\in N(i)} l_{ij}^{-1} v_j}{\sum_{j\in N(i)} l_{ij}^{-1}}$ where $l_{ij} = \| v_i - v_j \|$ and $N(i)$ is the valence of the ith vertex. A smoothness value measures how well a vertex is predicted as the weighted average of the vertices surrounding it. As in [25], we use $E_{\text{vis}} = \frac{1}{2M}(M_q + S_q)$ as the visual distortion criterion. $M_q = \sqrt{\sum_{i=1}^{M} \| v_i - \hat{v}_i \|^2}$ is the norm of the geometric distance between the original and reconstructed vertex coordinate values, $S_q = \sqrt{\sum_{i=1}^{M} \| S(v_i) - S(\hat{v}_i) \|^2}$ is the norm of the distance between the original and the reconstructed smoothness values of the coordinates, and M is the number of model vertices. The importance of incorporating smoothness into the visual distortion criterion is emphasized in [28] where it is suggested that S_q should perhaps have a substantially larger weight than M_q, despite lack of evidence as to what it should be.

The simulations have been conducted on two sets of models. The first set contains mostly small models from http://www.cs.unc.edu/isenburg/pmc/ with polygonal faces. Six models (Beethoven, Cow, Cupie, Raptor, Snake and Wolf) were included in the training set. The second set contains larger triangular mesh models. Eleven models (Dinosaur, Feline, Femur, Fibula, Fish, Head, Rabbit, Skull1, Skull2, Tibia, and Woman) were included in the training set.

The initial highest rate AECVQ data point was obtained by designing and encoding with MDVQ codebooks of 2048 and 8192 codevectors, for the first and second sets,

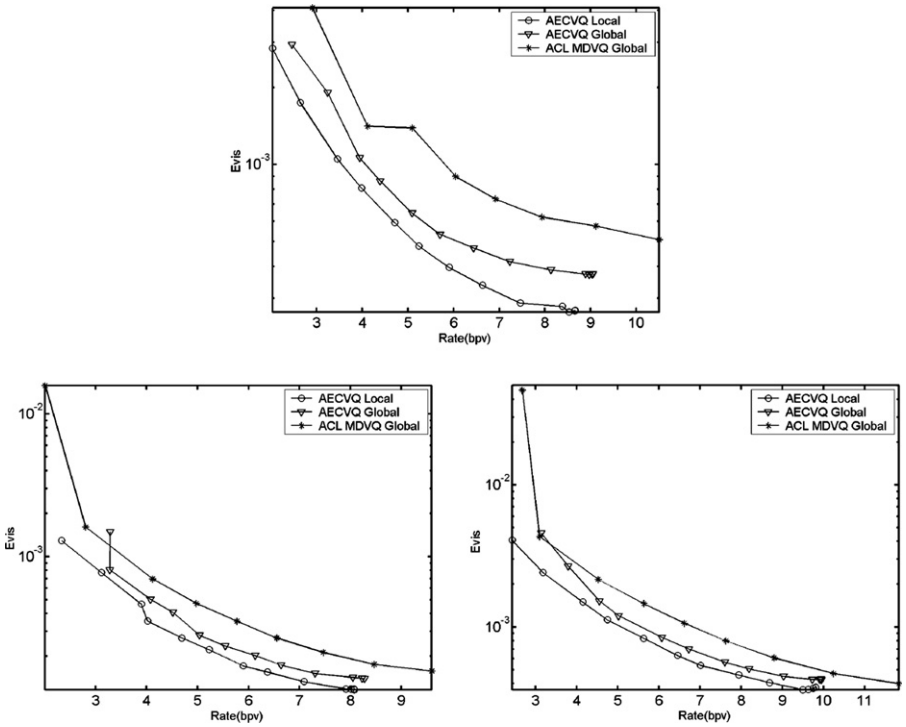

Fig. 6. Visual distortion (E_{vis})–Rate (bpv) curves, Top: Al (3618 vertices), Bottom-left: Lion (16302 vertices), Bottom-right: Triceratops (2832 vertices).

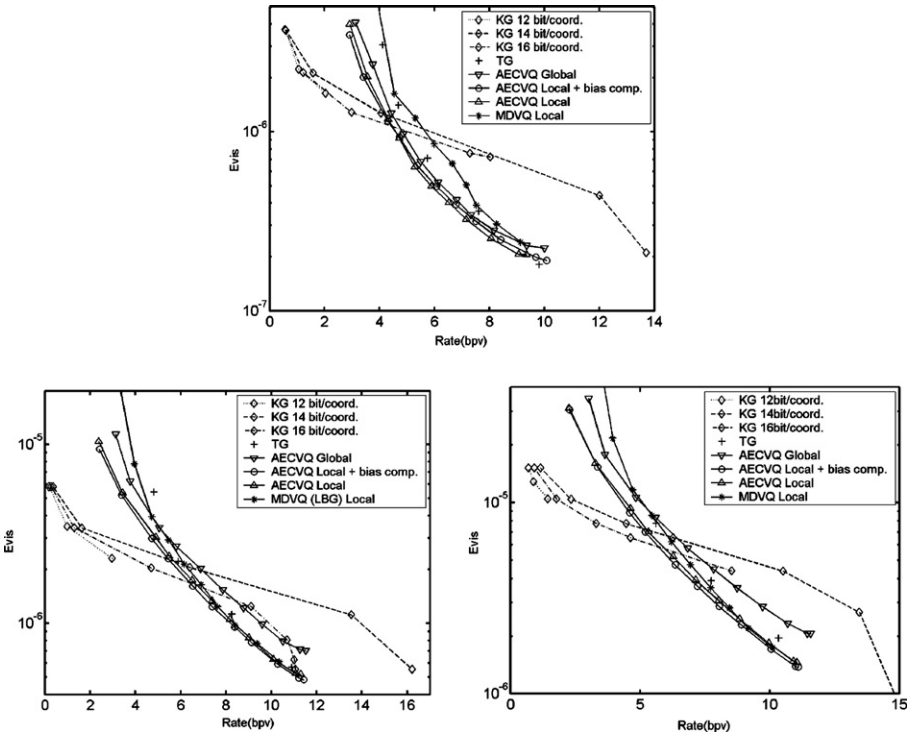

Fig. 7. Visual distortion (E_{vis}) vs. Rate (bpv) curves, Top: Bunny (34834 vertices), Bottom-left: Horse (19851 vertices), Bottom-right: Venus Head (50002 vertices).

Table 2
Rate (bpv of original mesh) and Hausdorff distance (wrt. b.box diag.) obtained with the proposed method and Normal Mesh Compression (NMC) method of [33] employing the loop wavelet transform (with flatness threshold of 0.01 during inverse transform)

Bunny 34834 vert.				
Proposed				
Rate (bpv)	10.080	6.836	4.338	2.841
Hausdorff dist.	0.002637	0.001496	0.001893	0.005615
NMC				
Rate (bpv)	11.554	4.665	2.368	1.220
Hausdorff dist.	0.005350	0.005370	0.005332	0.007420
Horse 48485 vert.				
Proposed				
Rate (bpv)	9.685	4.826	4.045	2.439
Hausdorff dist.	0.004005	0.005157	0.004375	0.005738
NMC				
Rate (bpv)	8.294	3.344	1.694	0.870
Hausdorff dist.	0.003854	0.003981	0.004057	0.005189
Venus 50002 vert.				
Proposed				
Rate (bpv)	11.111	8.055	4.597	2.275
Hausdorff dist.	0.001170	0.001080	0.002559	0.008122
NMC				
Rate (bpv)	8.055	3.256	1.656	0.856
Hausdorff dist.	0.003258	0.003583	0.003797	0.004424

The number of bits reported for NMC includes the number of bits for TG coding of coarsest level.

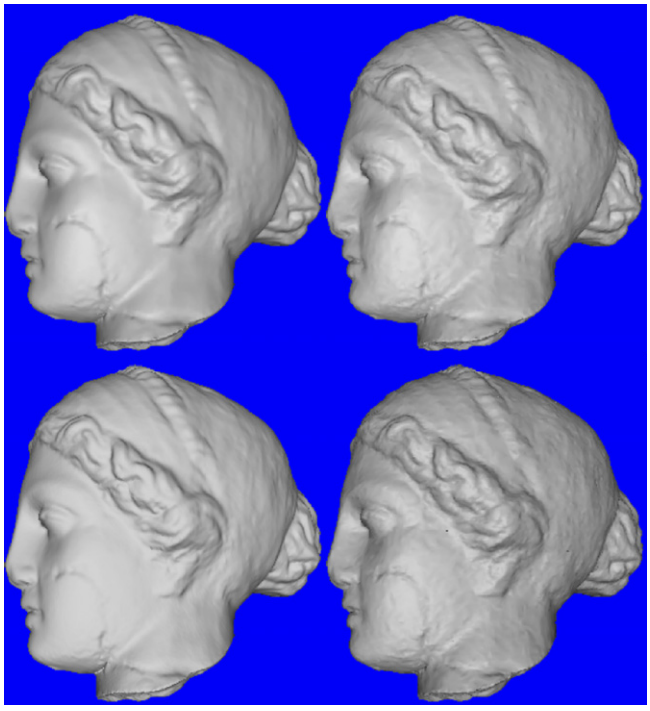

Fig. 8. Venus Head—Top-left: Original. Top-right: AECVQ coded at 7.729 bpv in the local coordinate system with bias compensation. Bottom-left: NMC [33] coded at 8.055 bpv. Bottom-right: TG [15] coded at 7.750 bpv. (Smooth rendering by virtue 3-D optimizer.)

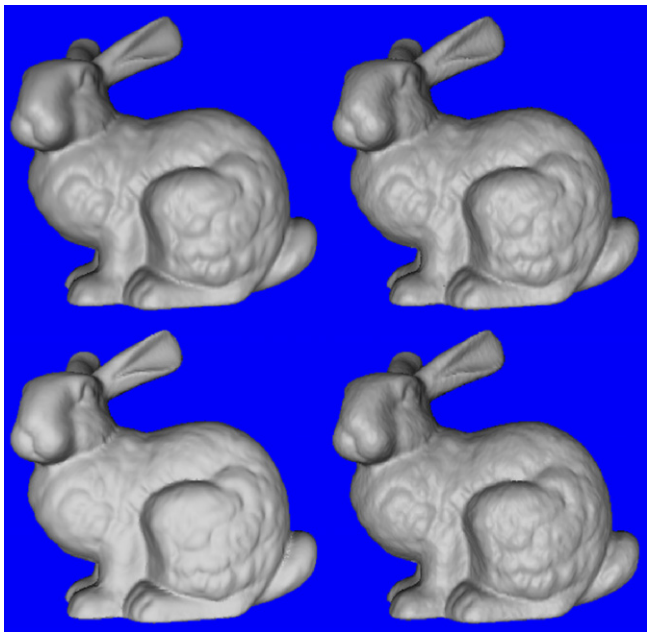

Fig. 9. Bunny Top-left: Original. Top-right: AECVQ coded at 5.688 bpv in the local coordinate system with bias compensation. Bottom-left: NMC [33] coded at 11.554 bpv. Bottom-right: TG [15] coded at 5.738 bpv. (Smooth rendering by virtue 3-D optimizer.)

respectively. The initial MDVQ codebook size had to be constrained so that the average frequency per probability model bin in arithmetic coding would not be small. For models larger than those tested here, larger initial codebook sizes can be used in order to achieve higher coding rates.

The visual distortion vs. rate curves displayed in Fig. 6 for the coding of the three polygonal test models, Al, Lion and Triceratops show that the AECVQ coding performance is better when the prediction error vectors are represented in the local coordinate system than in the global coordinate system. A second comparison based on Fig. 6 reveals that AECVQ coding with partial adaptation of the codebook via normalization by the statistics of the encoded model yields far better compression performance than MDVQ coding employing full codebook adaptation via the ACL algorithm of [9]. Since the highest rate point ($\lambda = 0$) on the AECVQ curve corresponds to MDVQ, a substantial part of the gain is due to the use of partial codebook adaptation.

The visual distortion vs. rate curves are displayed in Fig. 7 for the coding of the triangular test models, Venus Head, Horse and Bunny. AECVQ coding performance is again distinctly higher in the local coordinate system than in the global coordinate system. The gap between the two is especially large at low coding rates. Bias estimation and compensation yields a small coding gain for the Horse and Venus Head models at all rates. However, as suggested by the coding results for the Bunny model, reduction in prediction error variance (see Table 1) does not necessarily translate to a coding gain at all rates.

The AECVQ coding in the local coordinate system also exhibits a performance advantage to the TG predictive coding method of [15]. The gap is especially significant at low rates. A 20% gain in bit rate with respect to the TG method is also reported in [38], but at fairly higher rates than tested here.

It is also observed from Fig. 7 that the coding performance of the proposed method can exceed that of the KG spectral coding method of [25] at medium to high coding rates where the fidelity of the coded models is acceptable for most applications. Note that in the visual distortion vs. rate curves of [25], prequantization to 12 bit/coordinate followed by entropy coding results in maximum rates of only about 2–3 bpv. Prequantization to higher number of bits allows for a wider range of rates, but yields inferior coding performance due to the inefficient bit ordering (LSB's of certain coefficients coded before MSB's of others) in coding due to coefficient truncation.

In Table 2, we also provide geometry compression performance comparisons with the Normal Mesh Compression (NMC) method of [33]. Since [33] treats geometry compression as a surface approximation problem, we report the symmetrical Hausdorff distance (measured by the METRO tool of [34]) between the original and reconstructed surfaces, as in [35]. The symmetrical Hausdorff distance between surfaces S and S' is

$$d_s(S,S') = \max\left(\max_{p\in S} d(p,S'), \max_{p\in S'} d(p,S)\right)$$

where $d(p,S) = \min_{p'\in S'} \| p - p' \|$. At low rates, the performance of the wavelet method is always superior to the proposed method. However, at high rates, the proposed method frequently outperforms the wavelet method since the remeshing error becomes significant.

In Figs. 8 and 9, we show the original Venus Head and Bunny models, and the reconstructions obtained with the proposed method, the TG method and the NMC method. Unlike the TG method or the proposed method, the NMC method yields almost perfect results on smooth surfaces at

Table 3
Average run times for the components of the proposed compression method

	AECVQ encoding (8192 codevectors)	AECVQ decoding (8192 codevectors)	Transform + Inv. Tfm (encoder)	Transform + Inv. Tfm (decoder)
Horse (19851 vertices)	21.08 s	1.27 s	0.061 s	0.048 s
Venus Head (50002 vertices)	51.14 s	2.83 s	0.197 s	0.152 s

high rates. However, remeshing results in large high frequency features to be partially wiped out. Consequently, the curls on the hair of Venus Head and folds of skin on the hind leg of Bunny are better reconstructed with the proposed method than the NMC or the TG methods.

In Table 3, we report run times for the key components of the geometry compression system averaged over several runs on a Pentium 4, 2.6 GHz system with 1 Gb RAM, 800 MHz FSB. The encoder computational complexity is mainly due to the search for the optimal codevector according to Eq. (5) and could be considerably cut down by a parallel implementation of this search. On the other hand, the AECVQ decoding complexity, is mainly due to the large alphabet arithmetic decoding. The table look-up for mapping the entropy decoded codevector indices to the codevectors is very fast (several ms). In the case of polygonal models, the least squares estimation of the surface normal vector takes several minutes for a polygonal model of the same size as Horse and is not suitable for real-time operations.

5. Conclusions

In this paper, we have proposed the application of a local coordinate transformation prior to the predictive vector quantization of 3-D object mesh geometry. The rate-distortion advantage of the proposed transformation is explained by means of a rate-distortion lower bound for k-tuple ($k = 3$) coding of i.i.d. sources. The lower bound is parameterized by a measure of uniformity of the prediction error vector component variances. Significant coding gains are realizable with the local coordinate transformation since it yields a small uniformity measure.

Partial codebook adaptation via normalization of the encoded prediction error vector source with its own statistics has substantially lower implementation complexity than the ACL based full codebook adaptation scheme of [8,9]. This scheme yields better coding performance than the ACL based scheme since the ACL based design suffers from the mismatch of the universal codebook and the encoded source if the codebook is not transmitted with an expensive overhead. On the other hand, the overhead rate required for partial codebook adaptation is very little.

On the average, a slight improvement in triangular mesh coding performance may be obtained by estimating and compensating for the bias in the parallelogram prediction rule. The local coordinate system provides a natural setting to apply this technique.

Predictive AECVQ coding in a local coordinate system is a low complexity alternative to the transform based methods at medium to high rates and the TG predictive coding method at almost all, but especially low rates.

Possible directions of future research include investigating structured VQ schemes as well as hybrid of fixed-length and variable length codes to reduce the computational complexities at the encoding and decoding sides.

References

[1] U. Bayazit et al., Predictive vector quantization of 3-D polygonal mesh geometry by representation of vertices in local coordinate systems, Proc. of EUSIPCO (2005).
[2] J. Peng, C.-S. Kim, C.-C. Jay Kuo, Technologies for 3-D mesh compression: a survey, J. Vis. Commun. Image Rep. 16 (2005) 688–733.
[3] Y. Linde, A. Buzo, R.M. Gray, An algorithm for vector quantization design, IEEE Trans. Comm. 28 (1980) 84–95.
[4] P.A. Chou, T.D. Lookabaugh, R.M. Gray, Entropy-constrained vector quantization, IEEE Trans. ASSP 37 (1) (1989) 31–42.
[5] R.P. Rao, W.A. Pearlman, Alphabet-and entropy-constrained vector quantization of image pyramids, Opt. Eng. 30 (1991) 865–872.
[6] H. Khalil et al., The asymptotic closed-loop approach to predictive vector quantizer design with application in video coding, IEEE Trans. Image Proc. 10 (1) (2001) 15–23.
[7] T. Berger, Rate Distortion Theory, Prentice Hall, Englewood Cliff, NJ, 1971.
[8] E.-S. Lee, H.-S. Ko, Vertex data compression for triangular meshes, Proc. Pac. Graph. (2000) 225–234.
[9] P.H. Chou, T.H. Meng, Vertex data compression through vector quantization, IEEE Trans. Vis. Comp. Graph. 8 (4) (2002) 373–382.
[10] J. Li, D. Tian, G. AlRegib, Vector quantization in multi-resolution mesh compression, IEEE Signal Proc. Lett. 13 (10) (2006) 616–619.
[11] M. Deering, Geometry compression, Proc. SIGGRAPH (1995) 13–20.
[12] G. Taubin, J. Rossignac, Geometric compression through topological surgery, ACM Trans. Graph. 17 (2) (1998) 84–115.
[13] H. Hoppe, Progressive meshes, Proc. SIGGRAPH (1996) 99–108.
[14] G. Taubin et al., Progressive forest split compression, Proc. SIGGRAPH (1998) 123–132.
[15] C. Touma, C. Gotsman, Triangle mesh compression, Proc. Graph. Interface (1998) 26–34.
[16] J. Rossignac, Edgebreaker. Connectivity compression for triangle meshes, IEEE Trans. Vis. Comp. Graph. 5 (1) (1999) 47–61.
[17] S. Gumhold, W. Strasser, Real time compression of triangle mesh connectivity, Proc. SIGGRAPH (1998) 133–140.
[18] M. Isenburg, Compressing polygon mesh connectivity with degree duality prediction, Proc. Graph. Interface (2002) 161–170.
[19] A. Khodakovsky, et al., Near-optimal connectivity encoding of 2-manifold polygon meshes, INRIA Sophia Antipolis Research Report, Cedex, France, October 2002.
[20] M. Isenburg, J. Snoeyink, Face fixer: compressing polygon meshes with properties, Proc. SIGGRAPH (2000) 263–270.
[21] I.H. Witten, R.M. Neal, J.G. Cleary, Arithmetic coding for data compression, Comm. ACM 30 (6) (1987) 521–540.
[22] M. Isenburg, P. Alliez, Compressing polygon mesh geometry with parallelogram prediction, in: Proc. Conf. Visualization, Boston, MA, 2002, pp. 141–146.
[23] S. Gumhold, R. Amjoun, Higher order prediction for geometry compression, Proc. Int. Conf. Shape Model. Appl. (2003) 59–63.
[24] R. Pajarola, J. Rossignac, Compressed progressive meshes, IEEE Trans. Vis. Comp. Graph. 6 (1) (2000) 79–93.
[25] Z. Karni, C. Gotsman, Spectral compression of mesh geometry, Proc. SIGGRAPH (2000) 279–286.
[26] Z. Karni, C. Gotsman, 3D mesh compression using fixed spectral bases, in: Proc. Graph. Interface, 2001, Ottawa, Ontario, Canada, 2001, pp. 1–8.
[27] P. Rondao-Alface et al., Lapped spectral compression for 3-D triangle mesh compression, Proc. Int. Conf. Image Proc. 1 (2003) 781–784.
[28] O. Sorkine, D. Cohen-Or, S. Toledo, High-pass quantization for mesh encoding, in: Proc. Eurograph./ACM SIGGRAPH Symp. Geometry Process., pp. 42–51.
[29] A. Khodakovsky, P. Schroder, W. Sweldens, Progressive geometry compression, Proc. SIGGRAPH (2000) 271–278.

[30] S. Lavu, et al., Geometry compression of normal meshes using the estimation quantization algorithm, in: Proc. Eurograph. Symp. Geometry Proc. Aachen, Germany, 2003, pp. 52–61).
[31] A. Said, W.A. Pearlman, A new fast and efficient image codec based on set partitioning in hierarchical trees, IEEE. Trans. Circ. Syst. Video Tech. 6 (1996) 243–250.
[32] Z. Xiong, K. Ramchandran, M.T. Orchard, Space-frequency quantization for wavelet image coding, IEEE Trans. Image Proc. 6 (5) (1997) 677–693.
[33] A. Khodakovsky, I. Gustov, Normal Mesh Compression, Geometric Modelling for Scientific Visualization, Springer-Verlag, 2002.
[34] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, in: Computer Graphics Forum, Blackwell Publishers, vol. 17 (2), 1998, pp. 167–174.
[35] R. Klein, G. Liebich, W. Strasser, Mesh reduction with error control, in: Proc. Vis., San Francisco, CA, 1996, pp. 311–318.
[36] F. Payan, M. Antonini, An efficient bit allocation for compressing normal meshes with an error-driven quantization, Comp. Aided Geom. Des. Special Issue Geometric Mesh Process. 22 (5) (2005) 466–486.
[37] S. Gumhold, S. Guthe, W. Strasser, Tetrahedral mesh compression with the cut border machine, Proc. Vis. (1999) 51–58.
[38] H. Lee, P. Alliez, M. Desbrun, Angle-analyzer: a triangle-quad mesh codec, Proc. Eurograph. 21 (3) (2002) 383–392.
[39] P. Alliez, M. Desbrun, Progressive Compression for Lossless Compression of Triangle Meshes, Int. Conf. Comp. Graph. Interactive Tech. (2001) 195–202.
[40] I. Guskov et al., Normal meshes, Proc. SIGGRAPH (2000) 95–102.
[41] D. Cohen-Or et al., Multi-way geometry encoding, Tech. Report, Computer Science Dept., Tel Aviv University, 2002.
[42] M. Isenburg et. al., Geometry prediction for high degree polygons, in: 21st Spring Conf. Comp. Graph., Slovakia, May 2005, pp. 147–152.