PREDICTIVE VECTOR QUANTIZATION OF 3-D POLYGONAL MESH GEOMETRY BY REPRESENTATION OF VERTICES IN LOCAL COORDINATE SYSTEMS

Ulug Bayazıt,
Electronics Eng. Dept.
Isik University, Istanbul, Turkey
bayazit@isikun.edu.tr

Ozgur Orcay,
Computer Eng. Dept.
Dogus University, Istanbul, Turkey
oorcay@dogus.edu.tr

Umut Konur and Fikret S. Gurgen
Computer Eng. Dept.
Bogazici University, Istanbul, Turkey
konur@boun.edu.tr gurgen@boun.edu.tr

ABSTRACT

A large family of lossy 3-D mesh geometry compression schemes operate by predicting the position of each vertex from the coded neighboring vertices and encoding the prediction error vectors. In this work, we first employ entropy constrained extensions of the predictive vector quantization and asymptotically closed loop predictive vector quantization techniques that have been suggested in [3] for coding these prediction error vectors. Then we propose the representation of the prediction error vectors in a local coordinate system with an axis coinciding with the surface normal vector in order to cluster the prediction error vectors around a 2-D subspace. We adopt a least squares approach to estimate the surface normal vector from the non-coplanar, previously coded neighboring vertices. Our simulation results demonstrate that the prediction error vectors can be more efficiently vector quantized by representation in local coordinate systems than in global coordinate systems.

1. INTRODUCTION

Polygonal meshes are the primary representation tools used for the visualization of 3-D objects in manufacturing, entertainment, defense industry, computer aided design and architecture applications that allow general and restricted access to these objects over communication networks. 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 methods. Typically, polygonal mesh coding is performed in three distinct parts: Connectivity encoding concisely represents the topological information about the degrees of faces and vertices of a mesh as well as the adjacency relations between the faces bounded by edges and vertices. Geometry coding concisely represents the coordinate values of the vertices. Property encoding describes features such as texture coordinates and material attributes.

The state of the art in lossless connectivity coding can compress most regular meshes down to the range of 1-3 bits/vertex. On the other hand, full precision representation of the vertex coordinates typically starts at 8 bits/coordinate quantization. Vertex coordinates represented by 10bits/coordinate quantization can be losslessly compressed by simple entropy coding to achieve 6-7bits/coordinate rate. Since such rates are fairly significant with respect to the lossless connectivity coding rates, there is a need for lossy vertex coordinate compression.

Linear prediction methods are widely used in lossy vertex coordinate compression to exploit the correlation between the adjacent vertices. These methods require an order of traversal of the mesh vertices that is determined at the encoder and can be tracked by the decoder. For single resolution meshes, such orders are conveniently provided by the order of vertex processing during connectivity coding. Without loss of generality to employing other methods, the connectivity coding method of [9], is adopted in the current work for determining the order of predictive coding of the vertices. This method is briefly outlined in Section 2.

Simple linear prediction methods, such as the delta modulator of [4], as well as more advanced ones ([5,8,11]) that form the prediction as a linear combination of previously coded vertices, have been proposed. In [8] and [11], the prediction is the fourth corner of a parallelogram with the other three corners being the previously coded vertices from a neighboring face and/or from the same face as the predicted vertex. Geometry compression methods for multiresolution meshes ([6,7,12]) also employ some form of linear prediction. The parallelogram prediction method of [11], that we have adopted for our work, is described in Section 3.

Recently, the statistical dependencies among the prediction errors of the vertex coordinates have been efficiently exploited by vector quantizing the vector of coordinates of a vertex ([2,3]). The mapping of channel indices to codevectors (table look- up) for the reconstruction of prediction error vectors in vector quantization is also very suitable for hardware implementation .

The main contribution of this work is the utilization of a local coordinate system for the efficient vector quantization of each prediction error vector. Each prediction error vector is represented in its local coordinate system whose z-axis coincides with the surface normal vector at the location where the prediction is performed. Since three or more previously coded vertices of each coded polygonal face may be available to both the encoder and the decoder, a least squares estimation approach, as described in Section 3.1, is employed to determine the surface normal vector. The successive rotations described in Section 3.2, that transform the prediction error vector from the global coordinate system to its local coordinate system, do not change its norm, but reduce the variability in the surface normal component of the prediction error vector. Since the transformation clusters the prediction error vectors around a 2-D planar subspace, the components represented in a local coordinate system exhibit more statistical dependency than the components represented in the global coordinate system which can be efficiently exploited by block coding the components.

A similar transformation is mentioned in [2], but suffers from several drawbacks. Primarily, the nonorthogonal coordinate system of [2] counteracts the decorrelation goal of efficient compression schemes. Secondly, the prediction, taken to be the vertex at the far end of the triangle neighboring the one on which the predicted vertex lies, is of poor quality when compared to parallelogram prediction. Finally, the coordinate system of [2] is based on only the three previously coded vertices of the neighboring triangle. For polygonal meshes, where the original vertex to be predicted may lie on a face with more than three previously coded vertices, determining the surface normal vector using only three previously coded vertices does not always exploit all the available knowledge.

As a secondary contribution, we demonstrate the promise of the entropy coded and entropy constrained vector quantization (ECVQ, [1]) in 3-D mesh geometry compression. Previous vector quantization approaches to geometry compression ([2,3]) aimed to minimize distortion with no constraints on the coding rate.

In Section 4, we provide details of the asymptotical closed loop codebook design strategy of [13] developed for vector quantization

of prediction error vectors that vitally enables the design of codebooks on error vectors of predictions based on the *lossy coded* neighboring vertices. While, in [3], this strategy was applied to predictive minimum distortion vector quantization of 3-D mesh vertices, ours is an extension to the entropy constrained case.

The simulation results in Section 5 demonstrate the performance advantage of vector quantizing prediction error vectors represented in the local coordinate systems over vector quantizing prediction error vectors represented in a global coordinate system as well as the performance advantage of ECVQ over minimum distortion VQ in block coding prediction error data.

2. CONNECTIVITY CODING FOR POLYGONAL MESHES

In the connectivity coder of [9], the region growing algorithm proceeds by inserting a new face to one of one or more connected processed regions. The region to which the new face is inserted is bounded on the inside or the outside by a list of vertices called the active boundary. The boundaries of those connected regions that are not currently grown reside in a stack. When the active boundary collapses onto itself due to the insertion of the new face in a processed region, the boundary is split into an inside and an outside boundary. One of these is pushed onto a stack and the other becomes the new active boundary. When the active boundary collapses onto one of the boundaries in the stack, that boundary is merged with the active boundary. If an active boundary collapses onto itself with no more faces to add, the boundary from the top of the stack is popped and becomes the new active boundary.

The edge or the set of edges on the active boundary that defines the border between the newly inserted face and the processed region to which it is added is called the focus. For the insertion of the next face, priority is given to faces bordering vertices that have zero free valences and are closest along the active boundary in the counterclockwise direction to the previously inserted face. If there is no vertex with zero free valence on the active boundary, then vertices with one free valence are searched for in a similar way. If this search also fails, the edge that has been inserted last to the active boundary becomes the focus.

For each inserted face, the state of operation (normal, split, merge) is entropy coded and the face's degree is entropy coded conditional on the average valence of its previously coded vertices . In a counterclockwise order, the valence of each newly coded vertex of an inserted face is also entropy coded conditional on the face's degree. Since the alphabet size is small, arithmetic coding ([10]) has been preferred in our implementation.

3. GEOMETRY COMPRESSION ON A LOCAL COORDINATE SYSTEM

The parallelogram prediction of vertex x is formed as

$$\hat{\tilde{v}}_0 = \tilde{v}_1 + \tilde{v}_2 - \tilde{v}_3 \qquad (1)$$

where $\tilde{v}_i$, $i \in \{1,2,3\}$ represents the vector of reconstructed coordinates of the i'th vertex used in the prediction of v_0. The notation $\hat{\tilde{v}}_0$ emphasizes the dependence of the prediction on the reconstructed rather than the original vertices. The vertices are predicted in the counterclockwise order of traversal of the vertices in a new face during connectivity coding. If $\tilde{v}_1$ and $\tilde{v}_2$ are the only previously reconstructed vertices of the newly coded face on the boundary of the processed region, then $\tilde{v}_3$ is the first (reconstructed) vertex in the clockwise direction neighboring $\tilde{v}_1$ and $\tilde{v}_2$ in the last coded face, and the operation is termed *across prediction* ([11]).

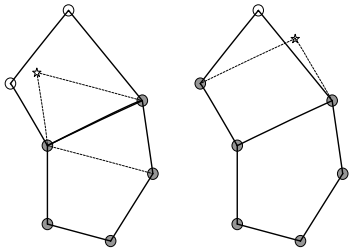

Figure 1. **Left: Across Prediction - The clockwise neighbor of the left coded vertex of the focus edge in the new face is predicted as the fourth corner of a parallelogram whose three corners (dark circles) are the two previously coded vertices of the new face on the focus edge and the clockwise neighbor of the focus edge in the previously coded face. Right: Within Prediction - Clockwise neighbor of the previously coded vertices of the new face is predicted as the fourth corner of a parallelogram whose three corners are the previously coded vertices (dark circles) of the new face. The predictions are shown as stars.**

Otherwise, $\tilde{v}_3$ is between $\tilde{v}_1$ and $\tilde{v}_2$ on the new face, and the operation is termed *within prediction* ([11]). Since each newly coded face has at least two previously coded vertices, the first vertex to be coded is either within predicted or across predicted while the others are within predicted. Figure 1 depicts each type of prediction.

For the parallelogram rule to work efficiently, the number of vertices on a face should be four. Another implicit assumption is that the vertices used for prediction and the predicted vertex are coplanar. In general, this assumption does not hold true for the following reasons:

i) The original vertices of polygonal faces designed by some mesh generation applications may be non-coplanar, although major deviations from coplanarity are infrequent. Applications usually partition highly non-coplanar faces into smaller coplanar ones.

ii) Even when the original faces are coplanar, the three reconstructed vertices used in prediction will, in general, not be coplanar due to the presence of quantization noise in $\tilde{v}_i$, i.e., $\tilde{v}_i = v_i + q_i$, $i \in \{1,2,3\}$. The quantization noise q_i has a component normal to the plane of the original face.

iii) Lastly, across prediction employs the vertices of one face to predict a vertex of a neighboring face. Even when the original faces are coplanar and the quantization noise is negligible, there could be a substantial surface normal component of the prediction error vector due to the crease angle between the two faces.

Linearly combining all three components of $\tilde{v}_i$, $i \in \{1,2,3\}$ by direct application of Eqn. (1) does not make use of the fact that most mesh generation applications tend to generate coplanar faces. To see this, let us assume that the three original vertices used in prediction are coplanar and employ two 2-D rotations that transform their plane to the x-y plane of a new coordinate system. Since rotations preserve distances, neither the norm of the prediction error vector nor the quantization error statistics changes. However, it is easier to impose the coplanarity constraint in the new coordinate system. The overall operation of performing these rotations for each face is termed the *local coordinate transformation*.

In the local coordinate system, $q_{i,z}$, the surface normal component of the quantization error q_i, has zero mean. Consider the situation where the sample quantization error values are such that $q_{1,z} > 0$, $q_{2,z} > 0$, and $q_{3,z} < 0$. In this case, the direct application of Eqn. (1) results in a cumulative error of $q_0 = q_1 + q_2 - q_3$ with respect to $\hat{v}_0 = v_1 + v_2 - v_3$ so that

$\hat{\tilde{v}}_0 = \hat{v}_0 + q_0$. The surface normal component of q_0 is $q_{0z} = q_{1z} + q_{2z} - q_{3z} > 0$. However, the best estimate for the surface normal component should be $E[q_{0z}] = E[q_{iz}] = 0$ if we assume, or our best guess is that the predicted vertex is coplanar with the vertices used in prediction. This implies that rather than directly applying the paralelogram rule in the 3-D space for determining the z component in the local coordinate system, the z component of the predicted vertex should always be set to zero and only the x-y components should be determined with the application of the parallelogram rule in a 2-D planar subspace.

When the vertices used in prediction as well as the predicted vertex are coplanar, the z-component of prediction error vector will be always zero so that this component does not need to be coded. In general, due to the three reasons stated above, this component has a nonzero variance that is lower than the variances of the x and y components. Since there exists virtually no memory between the prediction error vector components, a prudent lossy coding strategy for the prediction error vector would be to scalar quantize its components by allocating fewer bits to the z component than the x and y components. Vector quantization of the prediction error vector is an alternative strategy that exploits the lower variance characteristic of the z-component of the prediction error vector.

3.1 Least squares estimation of surface normal vector

Due to the noncoplanarity of the reconstructed vertices of a face, only the plane that best fits these vertices can be estimated.

Let the equation for the estimated plane of the face be

$$z = m_1 x + m_2 y + b$$

and let the coordinates of the N vertices of this face that have been previously reconstructed at the decoder be $(x_k, y_k, z_k)\, k = 1,..., N$. The least squares estimation for the vector of parameters, $p = (m_1\ m_2\ b)^T$ can be obtained as $p = (A^T A)^{-1} A^T h$ where $h = (z_1\ z_2\ ...\ z_N)^T$ and

$$A = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ . & . & . \\ x_N & y_N & 1 \end{bmatrix}$$

In order to satisfy the conditions $N \geq 3$ and $rank(A) = 3$ needed to avoid an underdetermined system of equations, the $N \geq 3$ vertices should not be on a plane perpendicular to the x-y plane in the global coordinate system. If $N \geq 3$, but $rank(A) < 3$, then a similar approach may be followed after writing the equation of the plane by expressing the y coordinate in terms of the x and z coordinates, or the x coordinate in terms of y and z coordinates depending on which form leads to $rank(A) = 3$.

3.2 Local coordinate transformation by rotations

Once the surface normal vector is estimated, then a first rotation with θ degrees that maps the surface normal vector to a vector in the x-z plane and a second rotation with ϕ degrees that further maps this vector to a vector on the z axis of the global coordinate system are determined. These rotations are illustrated in Figure 2 on a sample vector s_1. By applying these rotations in this order, all vertices that are used in prediction as well as the predicted vertex are placed in the local coordinate system. As discussed above, the z component of the prediction in the local coordinate system may be set to zero, while the other two components are determined from three reconstructed vertices by applying the parallelogram rule. Finally, the prediction error vector is found by subtracting the prediction from the predicted vertex in the local coordinate system.

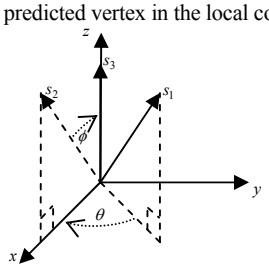

Figure 2. Transformation from the global to the local coordinate system: Rotations that are applied to the prediction vectors (or the vertices used to form them) are depicted above on a sample vector s_1 where the first rotation rotates s_1 by θ degrees and maps to s_2 on the xz plane and the second rotation rotates s_2 by ϕ degrees and maps s_2 to s_3 on the z axis.

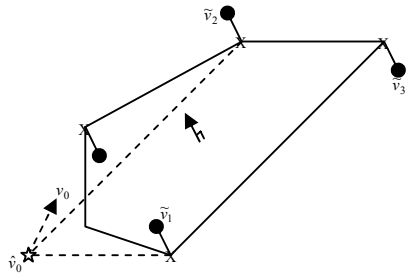

Figure 3. The parallelogram rule is applied on a plane whose surface normal vector is determined by least squares estimation. The three corner points that are used to form the prediction are denoted by X marks. These points are the projections of the reconstructed vertices $\tilde{v}_i$ onto the plane. The prediction error vector is shown by the dashed-dotted arrow.

The overall prediction operation is depicted in Figure 3. Setting the z-component of the prediction vector to zero is equivalent to projecting the three reconstructed vertices used in prediction onto the plane whose surface normal vector is determined by least squares estimation and forming the prediction vector by applying the parallelogram rule to these projections. Let $q_{i,z}$ be the realizations of random variable Q_z. Minimization of $\sum_i q_{i,z}^2 \approx E[Q_z^2]$ enables $E[(\hat{v}_{0,z} - \hat{\tilde{v}}_{0,z})^2] = 3E[Q_z^2]$ to be minimized assuming surface normal components of quantization errors of vertices are i.i.d. as Q_z. When v_0 is coplanar with v_i, $\hat{v}_{0,z} = v_{0,z}$. This implies the minimization of $E[(v_{0,z} - \hat{\tilde{v}}_{0,z})^2]$.

4. ASYMPTOTIC CLOSED LOOP ECVQ DESIGN

In predictive vector quantization, in order to ensure that the test source data resembles the training data as much as possible, codebook design must be performed in an iterative way termed the asymptotic closed loop (ACL).

In order for the decoder to track the prediction step, the encoder employs the vertices $\{\tilde{v}_i : i = 1,2,3\}$ that have been lossy compressed and reconstructed by vector quantization during prediction. On the other hand, in open loop (OL) codebook design, the prediction vector is formed as a linear combination of the

original vertices $\{v_i : i = 1,2,3\}$. Consequently, there is quantization noise $\{q_i : i = 1,2,3\}$ in the coordinates of the vertices employed in prediction during actual encoding, but there is no quantization noise in the coordinates of the vertices employed in prediction during open loop codebook design. Therefore, the average prediction error vector norm is less for open loop codebook design than for actual encoding. In order to remove this mismatch between test and training data, lossy compressed and reconstructed vertices should be used in prediction during codebook design. Since it is not possible to make use of the currently reconstructed vertices for prediction, one can at best make use of the most recently reconstructed ones. This suggests an iterative design technique.

Asymptotic closed loop (ACL) codebook design algorithm:
1.Initialization: Design an open loop predictive vector quantizer codebook to generate a set of prediction error vectors.
2. Generic step (N=1; N<=N_MAX;N=N+1):

a. *Design Codebook N by the ECVQ algorithm based on the current sequence of prediction error vectors.*
b. *For all vertices (I=1;I<=I_MAX;I=I+1): Quantize prediction error vector I by Codebook N and add the result to prediction for vertex I to get reconstruction for vertex I.*
c. *For all vertices (I=1;I<=I_MAX;I=I+1): Predict vertex I using reconstructed vertices with indices from the causal set {1,...,I-1} and subtract the prediction from the original vertex I to get the prediction error vector I.*

As suggested in [13], the predictions are based on the reconstructed vertices from the previous iteration so that the codebook can be 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.

In the first generic step, an open loop codebook is initially designed by running the LBG algorithm (rate constraint parameter set to zero). At each following generic step, the ECVQ iterations of [1] are performed until convergence with a nonzero rate constraint parameter where the initial codebook is the final codebook of the previous step. The generic steps are repeated N_MAX times or until there is no change between the final codebooks of two successive generic steps. At each generic step 2.b, the first vertex to be coded. is not predicted, but its coordinates are absolutely coded with a large number of bits.

5. SIMULATION RESULTS

In the simulations, prediction error data was collected from six models to train the vector quantizer codebooks. Prior to training and encoding, each model was normalized by its vertex coordinate standard deviation.

As stated in Section 3, Table 1 shows that the z component of the prediction error vector represented in the local coordinate system (shaded column) has a substantially smaller variance than its x or y components or the components of the prediction error vector represented in the global coordinate system.

Distortion vs. Rate curves displayed in Figure 4 for the coding of the test model, Teapot, are representative of the results obtained for the other three test models. Coding with an open loop trained ECVQ codebook yields a significantly better distortion-rate performance on a local coordinate system than on a global coordinate system. The codebooks designed by N_MAX=5 iterations of the ACL algorithm yield slightly better overall performance than those obtained by open loop design.

6. ACKNOWLEDGMENT

This work was partially supported by and carried out under Project No. 103E004 of TUBITAK (Turkish National Scientific Technical Research Organization).

REFERENCES

[1] P. A. Chou et. al., "Entropy-Constrained Vector Quantization," IEEE Trans. on ASSP, vol. 37, no. 1, pp. 31-42, Jan. 1989.
[2] E.-S. Lee, H.-S. Ko , "Vertex Data Compression for Triangular Meshes", Pacific Graphics '00, pp. 225-234, 2000.
[3] P. H. Chou, T. H. Meng, "Vertex Data Compression through Vector Quantization," IEEE Trans. Visualization and Computer Graphics, vol. 8, no. 4, Oct.-Dec. 2002.
[4] M. Deering, "Geometry Compression," Proceedings of Siggraph '95 pp.13-20, August 1995.
[5] G. Taubin, J. Rossignac, "Geometric Compression Through Topological Surgery," ACM Trans. Graphics, vol. 17, no. 2, pp. 84-115, 1998.
[6] H. Hoppe, "Progressive Meshes," Proceedings of Siggraph '96, pp. 99-108, August 1996.
[7] G. Taubin et. al, "Progressive Forest Split Compression," Proceedings of Siggraph '98, pp. 123-132, July 1998.
[8] C. Touma, C. Gotsman, "Triangle Mesh Compression," Proceedings of Graphics Interface '98, pp. 26-34, 1998.
[9] M. Isenburg, "Compressing Polygon Mesh Connectivity with Degree Duality Prediction", Proc. Graphics Interface, pp.161-170, 2002.
[10] I. H. Witten et. al., "Arithmetic Coding for Data Compression," Comm. of the ACM, vol. 30, no. 6, pp. 521-540, June 1987.
[11] M. Isenburg, P. Alliez, "Compressing Polygon Mesh Geometry with Parallelogram Prediction," Proc. of the Conference of Visualization 2002, pp. 141-146, Boston, Massachusetts, 2002.
[12] R. Pajarola, J. Rossignac, "Compressed Progressive Meshes," IEEE Trans. Visualization and Computer Graphics, vol. 6, no. 1, s. 79-93, Jan.-Mar. 2000.
[13] H. Khalil et. al., "The asymptotic closed-loop approach to predictive vector quantizer design with application in video coding," IEEE Trans. Image Proc., vol. 10, no. 1, pp. 15-23, Jan. 2001.

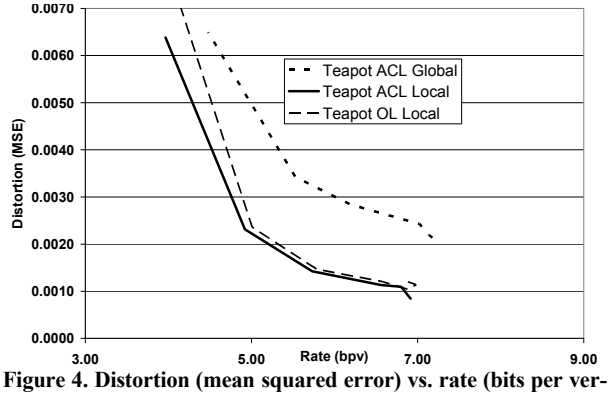

Figure 4. Distortion (mean squared error) vs. rate (bits per vertex) curves for the geometry compression of Teapot.

	Local			Global		
	x	y	z	x	y	z
Al	3.10E-3	2.78E-3	8.24E-4	1.89E-3	1.59E-3	2.04E-3
Galleon	1.74E-4	1.83E-4	3.2E-5	9.1E-5	1.2E-4	1.06E-4
Teapot	4.85E-3	9.53E-3	4.31E-4	4.87E-3	3.47E-3	4.16E-3
Triceratops	2.27E-4	3.06E-4	9.4E-5	1.64E-4	1.51E-4	1.77E-4

Table 1. Variances of prediction error vector components represented in global and local coordinate systems (ACL LBG coding with 512 codevectors).