PREDICTIVE CODING OF COLOR AND NORMAL VECTOR INFORMATION FOR THREE-DIMENSIONAL MESH REPRESENTATION

6-12.

Jeong-Hwan Ahn[a], Chang-Su Kim[b], and Yo-Sung Ho[a]

[a]Visual Communications Lab
Dept. of Information and Communications
Kwangju Institute of Science and Technology
1 Oryong-dong, Buk-gu, Kwangju, 500-712, Korea
E-mail: {jhahn,hoyo}@kjist.ac.kr

[b]Signal Processing Lab
School of Electrical Engineering
Seoul National University
San 56-1, Shillm-dong, Kwanak-gu, Seoul, 151-742, Korea
E-mail: cskim@ieee.org

ABSTRACT

Three-dimensional (3-D) mesh models have attribute data, such as colors, normal vectors, and texture coordinates to render or shade the surface of the mesh. Although several coding schemes have been developed to represent the topology and geometry information of the 3-D mesh, coding of attribute data has received less attention. In this paper, we propose a new predictive coding scheme for colors and normal vectors of the 3-D mesh model, where we predict colors and normals based on several ancestors along the vertex ordering. In order to encode the color information, we propose a mapping table that specifies how colors are mapped into other vertices. The mapping table can represent frequently-occurring color patterns efficiently. For normal vectors, we also propose an average predictor and the 6-4 subdivision quantizer in the spherical coordinate system. The proposed scheme has demonstrated reasonably good coding efficiency for various VRML test data.

1. INTRODUCTION

In recent days, 3-D models are popularly used in various applications, such as Internet services, computer graphics, and synthetic imaging systems. The 3-D model usually requires a large number of polygons to represent details of the model accurately. Since transmission bandwidth and storage capacity are limited in many applications, we need to find compact representation of the 3-D model [1,2].

Polygonal meshes are frequently used to represent surfaces of 3-D objects for fast interactive visualization. In general, 3-D mesh models have connectivity, geometry, and attribute data. While the connectivity data describe the connection relationship among vertices and characterize the topology of the 3-D model, the geometry data specify locations of the vertices in the 3-D space. The attribute data comprise normal vectors, colors and texture coordinates which are needed to paint and shade the 3-D model. The attribute data is often attached to vertices, faces, or corners of the 3-D mesh model.

Under the limited network bandwidth, we need to reduce the number of transmission bits as much as possible, while maintaining reasonable reconstruction quality. Since the geometry and attribute data specified by floating-point numbers are the major parts of the 3-D model representation, coding of those information is effective in reducing the total number of bits. Although current research for the 3-D mesh representation has focused on connectivity and geometry coding, coding of the attribute data has received less attention.

Figure 1 shows the encoder configuration for 3-D mesh model.

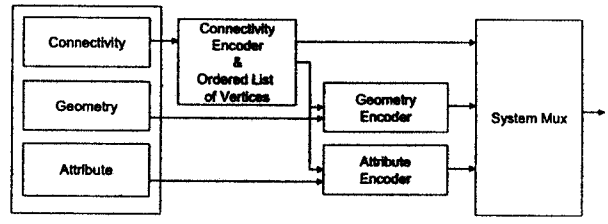


Figure 1. System configuration of encoder

In Figure 1, the connectivity coder produces an efficient representation of the association between each mesh and its sustaining vertices. The geometry coder is for lossy or lossless coding of vertex positions. The attribute coder is for lossy or lossless compression of color, normal vectors, and texture coordinates.

Typically there are four steps in coding of the 3-D mesh model: vertex ordering, data prediction, quantization, and entropy coding. In vertex ordering, we rearrange the vertices of the 3-D mesh model into a certain structure to describe the local correlation among the vertices explicitly. The order of the vertices is generally determined by the connectivity encoder. In the prediction step, we utilize such a structure to remove redundancy in the data and produce a sequence of prediction residuals. Then, we encode the prediction errors using a variable-length code in the quantization and entropy coding steps.

Deering has first proposed an algorithm for 3-D mesh coding based on a generalized triangle strip [3,4]. Linear quantizer is applied to both geometry and attribute properties along the generalized triangle strip. A bounding box is defined to convert from the floating-point representation to the fixed-point representation. The fixed-point numbers are then differentially coded and Huffman coding is applied. For normal vectors, innovative quantization and coding scheme is proposed. A normal vector is represented as a point on the unit radius sphere and then parameterized by

two angles, θ and ϕ, using the spherical coordinate system. Points on the sphere are folded first by octant, and then the sphere split into six sextants. Quantized spherical coordinate values are used inside each sextant. This compression scheme is currently used as the compressed format in the Java3D package [3]. However, the aim of coding was to reduce the bandwidth requirement of the rendering engine rather than efficient transmission over a network.

Taubin and Rossignac have introduced a topological surgery scheme, where the current geometry and color are linearly predicted along the vertex spanning tree [5]. The normal vector is also quantized in the spherical coordinate system. (θ,ϕ) is represented on the octant of the unit sphere, which determine the location of the normal vector. Then, each octant is spanned by a base triangle defined by the three canonical vectors. An octant is discretized by recursively subdividing the base triangle n times. Although they have introduced novel coding techniques, their emphasis was not so much on coding efficiency.

Touma and Gotsman have proposed an encoding scheme with implicit traversal of the mesh based on the degree of each vertex [6]. The geometry and attributed data of the 3-D mesh are differentially coded with respect to the predicted value by the so-called parallelogram rule. The difference between the predicted and actual values is then entropy-coded using a Huffman code. Bossen has applied this parallelogram prediction rule to the topological surgery and increased coding efficiency by the QM coder. Later, this coding scheme has been adopted as the MPEG-4 SNHC 3DMC standard [7].

However, those previously proposed coding schemes pay less attention to the compression problem of attribute data, such as colors and normal vectors. In this paper, we propose a new predictive encoder for colors and normal vectors. The encoder is based on the differential pulse code modulation (DPCM) structure [9]. Although the conventional schemes employ the geometry predictor and quantizer to encode colors and normal vectors, attribute data associated with each vertex usually have their own characteristics. This implies that predictors and quantizers for colors and normal vectors should be designed separately.

In order to encode the color information, we introduce a mapping table that specifies how *(r,g,b)* color vectors are mapped into other vertices. Since neighboring vertices have similar color values in the 3-D mesh model, it would be more efficient to encode typical color patterns of the 3-D model, instead of coding all color values independently. For normal vectors, we propose an average predictor where we take average of the normal vectors of all adjacent vertices and use the average normal vector as a prediction value. In addition, we adopt the 6-4 subdivision method [8] to quantize the normal (θ,ϕ) in the spherical coordinate system. The proposed scheme demonstrates reasonably good coding efficiency for various VRML test data.

2. COLOR CODING

Color data play an important role both in human perception of objects and in automated image understanding based on computer vision. Color information can be attached to vertices, faces, or corners of the 3-D mesh model. In this paper, we propose a compression algorithm for color data with the per-vertex binding.

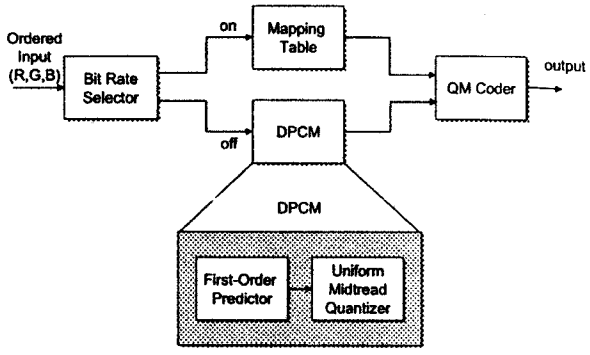


Figure 2. Encoder for colors

Figure 2 shows the block diagram of the proposed color encoder. Color data is predicted along the vertex traversal in the connectivity encoder. In the bit rate selector, we choose whether color data are compressed by a DPCM scheme or a mapping table. Specifically, if the overall size of the mapping table is larger than the required bit rate, the color data is encoded by the DPCM scheme with a large quantizer step size. Otherwise, the mapping table is employed to encode frequently recurring colors efficiently.

2.1 DPCM

We employ the first-order predictor with coefficient 1 to encode color data differentially. In other words, each color is simply predicted by only one preceding color. Generally, between two adjacent vertices, the color correlation is higher than the position correlation. Therefore, although higher-order predictors, such as the parallelogram predictor, are useful to remove redundancies, the first-order predictor is sufficient for color prediction.

After we estimate the current color value based on the previously coded color value, the prediction error is uniformly quantized. The dynamic range of the prediction errors is limited to $[-1,1]\times[-1,1]\times[-1,1]$. Thus, we employ a uniform midtread quantizer of M levels with step size $\Delta = 2/(M-1)$, and encode the index with the QM coder [10].

2.2 Mapping Table

3-D mesh models can be obtained by a laser scanning system or generated using an authoring tool. In these cases, some parts of the models are often painted with the same color due to the limited precision of the system. In other words, 3-D models often have only a small set of colors that recur frequently over many vertices.

For instance, the HUMIDITY model in Figure 7(a) contains 39,072 vertices painted with only 401 different colors. The proposed mapping table scheme can provide a high coding gain for such 3-D models that have frequently recurring colors. In the mapping table, a color value is encoded only once when it appears first during the vertex layer traversal. When the same recurs on another vertex

later, the encoder records the index of the last vertex with the same color, instead of the color itself. The mapping table can reduce the bit rate for color data significantly, since it avoids duplicated encoding of the same color.

Let us encode the sequence in Table 1(a). The coded data for this sequence are summarized in Table 1(b).

Table 1. Mapping table

(a) Input sequence

Index	1	2	3	4	5	6	7	8	9	10
Color	4	7	6	5	4	6	4	6	8	4

(b) Coded data

Index	Color Index	Differential Color
1	0	4
2	0	3 (= 7 - 4)
3	0	-1 (= 6 - 7)
4	0	-1 (= 5 - 6)
5	4 (= 5 - 1)	-
6	3 (= 6 - 3)	-
7	2 (= 7 - 5)	-
8	2 (= 8 - 6)	-
9	0	2 (= 8 - 6)
10	3 (= 10 - 7)	-

Initially, the mapping table is empty. Since the first vertex has a color value of 4, the encoder records the color of index 0, which informs the decoder that the color value of 4 is first used. Since the second vertex has a color value of 7, which is not used yet, the encoder records the color index 0 and the differential color 3 that is the difference between the current and the preceding colors. The information for the third and fourth vertices is encoded in the similar way.

However, since the fifth vertex has a color value of 4 that is already used in the first vertex, the encoder records the color index 4 by the index difference between the current vertex and the last vertex that has the same color. Note that the encoder need not record the color value, since the decoder can reconstruct the color value using the color index. The encoding operation continues in this manner till the end of the sequence.

3. NORMAL VECTOR CODING

In this paper, we also propose a compression algorithm for normal vector data with the per-vertex binding. Since we only consider the normal vector of length 1, any normal vector on the unit sphere can be represented by $(1,\theta,\phi)$ in the spherical coordinate system. Thus, it is sufficient to encode only θ and ϕ coordinate values. In order to improve the coding gain for normal vectors, we predict each normal vector using the average prediction scheme, and quantize the prediction error using the 6-4 subdivision scheme.

As shown in Figure 3, the proposed encoding system for normal vectors consists of four stages: transform to the spherical coordinate system, average prediction, quantization by subdivision, and entropy encoder.

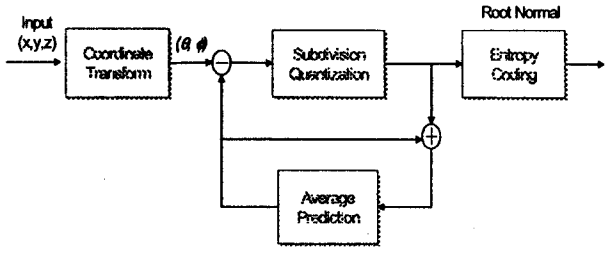


Figure 3. Encoder for normal vectors

3.1 Average Prediction

The proposed average prediction is based on the following two assumptions: 1) the face normal vector of a triangle is given by the average of the three vertex normal vectors of the triangle. 2) the face normal vector of a triangle can be approximated by the average of the face normal vectors of the neighboring triangles.

Using these two assumptions, we can obtain the prediction rule for each vertex normal vector. Figure 4 shows an example of the average prediction, where f_i denotes the face normal vector and n_i denotes the vertex normal vector.

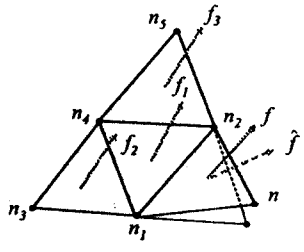


Figure 4. Prediction using the average rule

Suppose that the vertex normals n_1, n_2, n_3, n_4, and n_5 are already encoded, the vertex normal n is to be encoded, and the surface normal f is also unknown. By the second assumption, we have $f_1 \cong (f+f_2+f_3)/3$. In other words, we can predict f by

$$\hat{f} = 3f_1 - f_2 - f_3 \tag{1}$$

Let $\hat{n}$ denote the prediction of n. Then, by the first assumption, Eq. (1) can be rewritten in terms of the vertex normal vectors by

$$\left(\frac{\hat{n}+n_1+n_2}{3}\right) = 3\left(\frac{n_1+n_2+n_4}{3}\right) - \left(\frac{n_1+n_3+n_4}{3}\right) - \left(\frac{n_2+n_4+n_5}{3}\right).$$

Therefore, $\hat{n}$ is given by

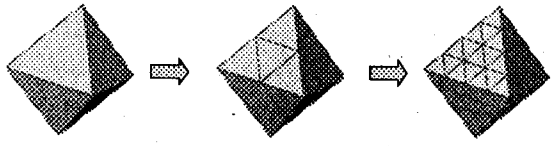
$$\hat{n} = n_1 + n_2 - n_3 + n_4 - n_5, \tag{2}$$

and the prediction residual is obtained by

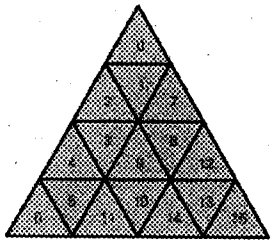
$$\begin{aligned} \Delta n &= n - \hat{n} = (1,\theta,\phi) - (1,\hat{\theta},\hat{\phi}) \\ &= (0,\Delta\theta,\Delta\phi). \end{aligned} \tag{3}$$

3.2 Quantization by Subdivision

In the MPEG-4 SNHC standard, the normal vector $(1,\theta,\phi)$ is pre-quantized using the 8-4 subdivision scheme [5,7]. After the unit sphere is divided into eight octants, as shown in Figure 5(a), each octant is approximated by a triangle, which is subdivided n times recursively. Thus, we can represent the normal vector by its octant number and triangle index, using the index pattern shown in Figure 5(b).



(a) Subdivision of the octahedron



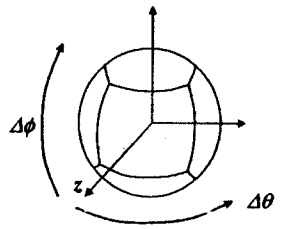
(b) Index pattern

Figure 5. Normal vector compression in MPEG-4 SNHC 3DMC

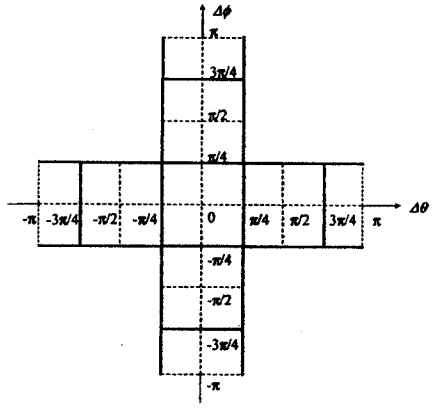
However, this approach has some shortcomings. Since the unit cube is approximated by an octahedron, quantized normal vectors are not ideally distributed. In other words, quantized normal vectors are uniformly distributed over an octahedron, instead of over a unit sphere. In addition, the index pattern in Figure 5(b) is not suitable for differential coding. Since the triangles in each octant are represented by one-dimensional indices, neighboring triangles can have dissimilar indices. For example, in Figure 5(b), although the triangles 3 and 9 are adjacent, their index difference is as large as 6. Furthermore, there is discontinuity of indices along the boundary between two octants.

In order to overcome these limitations, we adopt the 6-4 subdivision scheme [8] to quantize the normal vectors. As shown in Figure 6(a), the surface of the unit sphere is divided into six disjoint regions, called sextants, of the identical shape. The centers of these regions are located at (1,0,0), (-1,0,0), (0,1,0), (0,-1,0), (0,0,1) and (0,0,-1). For example, the region with a center at (0,0,1) is given by $\{(1,\theta,\phi)$: $\vee$ $-\pi/4 < \theta < \pi/4$ and $-\pi/4 < (\theta,\phi) < \pi/4\}$. Then, we divide each sextant into 2^{2n} cells by uniformly quantizing θ and ϕ into 2^n levels, respectively.

Using the 6-4 subdivision quantization, we can encode the prediction residual Δn in Eq. (3) efficiently. Around the sextant containing the prediction vector $\hat{n}$, we can unfold the unit sphere, as shown in Figure 6(b). Then, each cell is enumerated by a 2-D index, which indicates which row and column the cell lies on. Then, the 2-D index difference between n and $\hat{n}$ is encoded by the QM coder. In this way, we can prevent the discontinuity problem in the MPEG-4 3DMC algorithm, and we exploit the redundancy in normal vectors more effectively and improve the coding gain.



(a) Sextant partitioning



(b) Further partitioning

Figure 6. The 6-4 subdivision

4. EXPERIMENTAL RESULTS

We investigate the performances of the proposed color and normal algorithms on several test models, which have been collected and used by the MPEG-4 SNHC 3DMC group.

4.1 Error Metric

In some sense, color data of 3-D meshes are similar to 2-D images. Hence, the root-mean-squared error (RMSE) can be employed as the color error measure.

$$d_{color} = \sqrt{\sum_{i=1}^{n} \left\| c_i - \tilde{c}_i \right\|^2}, \qquad (4)$$

where c_i and $\tilde{c}_i$ denote the original and reconstructed colors of i-th vertex, respectively.

When we render the 3-D model, the angle between the normal vector and the light direction is used. Therefore, the error measure for normal vectors should consider the angle between the original normal vector and its reconstruction. We define the normal error measure by

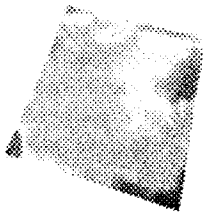
$$d_{normal} = \sum_{i=1}^{n} \cos^{-1}(n_i \cdot \tilde{n}_i), \qquad (5)$$

where n_i and $\tilde{n}_i$ denote the original and reconstructed normal vectors of i-th vertex, respectively

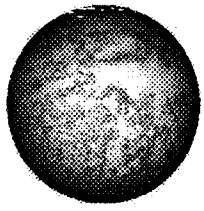
Although these error measures for colors and normal vectors may not be very close to perceived visual quality, they are simple and easy to compute.

4.2 Color Coding

Figure 8 shows 3-D test models for color compression, and Table 2 summarizes their properties.



(a) HYMIDITY



(b) SPHERE

(c) BRAIN

(d) C-SKULL

Figure 7. Test models for color compression

Table 2. Properties of 3-D test models

3-D Model	*nV*	*nF*	*nC*	*nCS*
HUMIDITY	39,072	77,504	39,072	401
SPHERE	41,369	82,734	41,369	337
BRAIN	34,278	69,180	34,278	9,419
C-SKULL	84,635	172,002	84,635	16,775

In Table 2, *nV* is the number of vertices, *nF* is the number of faces, *nC* is the number of listed colors to describe the model, and *nCS* is the number of different colors used to paint the model. In these models, the color value is assigned to each vertex; hence, $nC = nV$. Note that *nCS* is much smaller than *nC*, which implies that the same color recurs frequently to paint several vertices in these models.

In Table 3, we compare performance of the proposed color compression algorithm to that of the MPEG-4 SNHC 3DMC algorithm. There are two options in the proposed algorithm. If the overall size of the mapping table is larger than the required bit rate, the color data is encoded by the DPCM scheme with a large quantizer step size. Otherwise, the mapping table scheme is employed to encode frequently recurring colors effectively. In general, when we use the mapping table, 9~15 bits per color (bpc) are required.

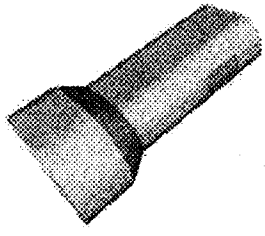
From Table 3, we can observe that the DPCM scheme provides better performance than the MPEG-4 SNHC 3DMC algorithm. Moreover, we note that the proposed algorithm provides much improved coding gains at high bit rates by employing the mapping table scheme.

Table 3. Simulation results of color compression

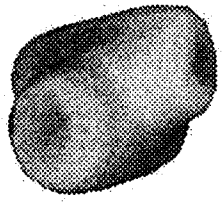
3-D Model	bpc	MPEG-4	Proposed	Option
HUMIDITY	5	0.0042	0.0041	DPCM
	8	0.0005	0.0003	DPCM
	10	0.0002	0	mapping table
SPHERE	5	0.3793	0.1820	DPCM
	7	0.1435	0.0931	DPCM
	10	0.0641	0.0004	mapping table
BRAIN	6	0.1441	0.1135	DPCM
	11	0.0234	0.0224	DPCM
	17	0.0036	0.0005	mapping table
C-SKULL	4	0.2162	0.1704	DPCM
	8	0.0539	0.0524	DPCM
	10	0.0332	0.0012	mapping table

4.3 Normal Vector Coding

The proposed algorithm for normal vector compression is tested on LEGS and TWO-CYLINDER models in Figure 8, and BRAIN and C-SKULL models in Figure 7.



(a) LEGS



(b) TWO-CYLINDER

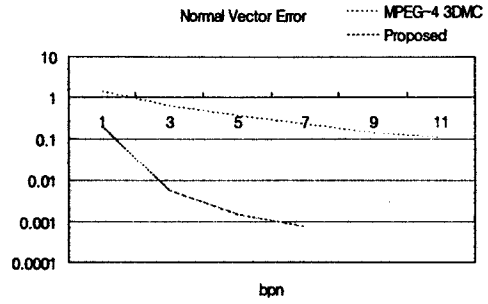
Figure 8. Test models for normal vector compression

Table 4 summarizes the properties of 3-D test models, where *nN* is the number of normal vectors. Since a normal vector is assigned to each vertex of the model, $nN = nV$. While the LEGS model has rather flat regions, the TWO-CYLINDER model has curved surfaces. The BRAIN and C-SKULL models contain larger numbers of vertices and more complex normal vectors.

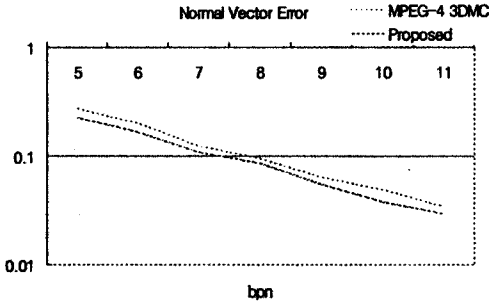
Table 4. Properties of 3-D test models

3-D Model	*nV*	*nF*	*nN*
LEGS	410	816	410
TWO-CYLINDERS	41,369	82,734	41,369
BRAIN	34,278	69,180	34,278
C-SKULL	84,635	172,002	84,635

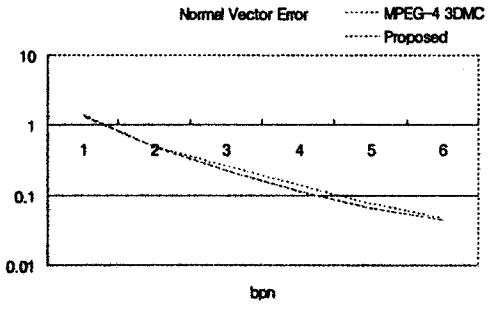
Performance of the proposed algorithm is compared to those of the MPEG-4 SNHC 3DMC algorithm in Figure 9, where the horizontal axis represents the bits per normal (bpn) and the vertical axis represents normal error in the logarithm scale.



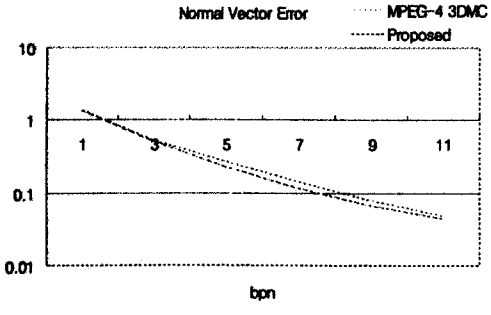
(a) LEGS



(b) TWO-CYLINDER



(c) BRAIN



(d) C-SKULL

Figure 9. Simulation results for normal vector compression

In Figure 9, we observe that the proposed algorithm provides higher coding gains than the MPEG-4 SNHC 3DMC algorithm for all the test models. Especially, for the LEGS model, a high coding gain is achieved. This is because the normal vectors in the LEGS model are highly correlated, and thus can be effectively predicted by the proposed algorithm.

Our simulation results indicate that the proposed algorithms for attribute data coding of 3-D mesh models are quite promising.

5. CONCLUSIONS

In this paper, we have proposed predictive coding schemes for colors and normal vectors of 3-D mesh models. By exploiting the characteristics of colors and normal vectors of the 3-D models, we have developed efficient prediction and quantization algorithms. For color data, we have proposed a coding scheme using a mapping table to encode frequently recurring patterns effectively. For normal vectors, we have presented an average predictor and the 6-4 subdivision quantizer. Simulation results have demonstrated that the proposed coding schemes outperform the MPEG-4 SNHC 3DMC standard for various test models.

As a future work, we need to investigate an efficient traversing scheme to achieve more saving in the overall code size, because coding performance heavily depends on the input vertex traversal ordering.

ACKNOWLEDGEMENT

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) through the Ultra-Fast Fiber-Optic Networks (UFON) Research Center at Kwangju Institute of Science and Technology (K-JIST), and in part by the Ministry of Education (MOE) through the Brain Korea 21 (BK21) project

REFERENCES

[1] J. Hartman and J. Wernecke, *The VRML 2.0 Handbook*, Addison-Wesley, 1996.

[2] The Virtual Reality Modeling Language, ISO/IEC 14772-1, September 1997, http://www.web3d.org

[3] Java3D, Sun Microsystems, 1998, http://java.sun.com

[4] M. Deering, "Geometry compression," Proceedings of SIGGRAPH, pp. 13-20, Aug. 1995.

[5] G. Taubin and J. Rossignac, "Geometry compression through topological surgery," ACM Transactions on Graphics, pp. 84-115, April 1998.

[6] C. Touma and C. Gotsman, "Triangle mesh compression," Proc. Graphics Interface '98, pp. 26-34, 1998.

[7] *Description of core experiments on 3D model coding*, ISO/IEC JTC1/SC29/WG11 MPEG/M4312, Dec. 1998.

[8] J. Li, C.-C. Kuo, and H. Chen, "Embedded coding of mesh geometry," ISO/IEC JTC1/SC29/WG11 MPEG/M3325, 1998.

[9] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, 1992.

[10] W. Pennebaker and J. Mitchell, *JPEG Still Image Compression Standard*, Van Nostrand Reinhold, 1993.