Motion Compensated Coding of 3-D Animation Models

Jeong-Hwan Ahn*[a], Chang-Su Kim**[b], C.-C. Jay Kuo[b] and Yo-Sung Ho[a]

[a]Dept. of Information and Communications, Kwangju Institute of Science and Technology
[b]Dept. of Electrical Engineering, University of Southern California

ABSTRACT

In this paper, we propose a new algorithm to code the animated three-dimensional (3-D) mesh model. After we classify the mesh frames in the animation model into intra- and inter-meshes, each mesh frame is decomposed into linear triangular strips, which in turn are partitioned into several fixed-length segments. The intra-mesh is compressed by differential coding of vertex coordinates. The inter-mesh is motion-compensated segment by segment from the previous meshes and residual errors are transformed by 1-D DCT. The transform coefficients are then entropy coded. We demonstrate that the proposed algorithm yields an improved coding gain than the MPEG-4 SNHC codec.

Keywords: 3-D Animation Models, Triangular Mesh, Graphic Compression

1. INTRODUCTION

Three-dimensional (3-D) animation models are widely used in various multimedia applications, such as Internet services, computer graphics, and synthetic imaging systems. A 3-D static model is often represented by triangular meshes which can be defined by vertices and their associated edges. A 3-D animation model is a sequence of 3-D static models which represent 3-D objects at consecutive time instances. In order to transmit or store the huge amount of data for the animation model, we need to encode the dynamic mesh information efficiently. However, relatively little work has been performed for efficient coding of the 3-D animation model.

The MPEG-4 SNHC[1] subgroup has developed a 3-D mesh coding (3DMC) algorithm to code triangle connectivity and vertex positions of 3-D static models. In order to code a 3-D animation model, we can directly apply the 3DMC codec to each mesh frame of the underlying model independently. However, in this approach, temporal correlation between adjacent mesh frames cannot be exploited appropriately. Lengyel[2] proposed a mesh sequence coding algorithm. In this work, the geometry sequence of the 3-D model was segmented, and each segmented part was predicted by an affine transformation. Then, transform coefficients and residuals were quantized and encoded. Ahn and Ho[3] improved the segmentation scheme by taking advantage of the temporal coherence in the spherical coordinate system. These algorithms do not provide sufficient reconstruction quality at low bit rates; therefore, it is difficult to distribute 3-D animation contents in real time over a narrow-bandwidth network. In this research, we propose a new motion-compensated coding scheme, which yields an improved coding gain than the previous works.

2. THE PROPOSED ALGORITHM

While a 2-D image is defined on a regular rectangular grid, a 3-D mesh has an arbitrary connectivity in the 3-D space. Due to the irregular structure of triangle meshes, it is not easy to design a universal coding scheme for 3-D models based on a specific data distribution. Therefore, we convert the triangular mesh structure into a triangular linear strip to represent the connectivity in a systematic way. With the aid of the decomposition process, we can borrow several well-developed concepts for 2-D video compression techniques.

After we divide the triangular strip into segments of the same length, each segment is treated as a coding unit. More specifically, a segment is the counterpart of a macroblock in H.263 or MPEG. In popular video coding standards, such

* jhahn@kjist.ac.kr; phone: +82-62-970-2258; fax: +82-62-970-2204; http://visual.kjist.ac.kr/~jhahn; 1 Oryong-Dong, Buk-Gu, Kwangju, 500-712, Korea.
** cskim@ieee.org; Integrated Media Systems Center and Dept. of Electrical Engineering, Univ. of Southern California, Los Angeles, CA 90089-2564, USA.

Visual Communications and Image Processing 2002,
C.-C. Jay Kuo, Editor, Proceedings of SPIE Vol. 4671 (2002) © 2002 SPIE · 0277-786X/02/$15.00

as H.261, H.263, MPEG-1, MPEG-2 and MPEG-4, we use the repeating pattern of frame types, such as IPPP or IBBPBB. Figure 1 illustrates inter-frame dependencies in the IBBPBB pattern.

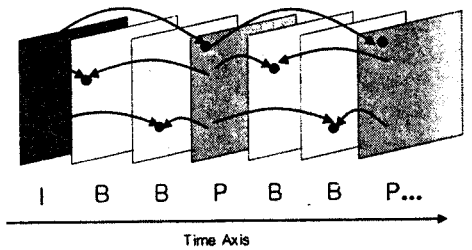


Figure 1. Temporal frame structure

An arrow from one frame X to another frame Y implies that each macroblock in Y contains a motion vector specifying a similar macroblock in X. Based on the temporal frame structure, we perform motion estimation to exploit high temporal correlation in the animation model. In a similar way, we can define three mesh types for an animation model.

(a) I (intra-coded or reference) meshes are compressed without reference to any other meshes. As a result, they consume more bits, but provide random access points. I-meshes are encoded by a DPCM coding technique[7].

(b) P (predicted) meshes are coded using motion compensated prediction from its previous I- or P-mesh. Since prediction errors are relatively small, P-meshes can be compressed more compactly than I-meshes.

(c) B (bi-directional) meshes are motion-compensated from two nearest P and/or I-meshes in the past and the future instances. The difference between the current vertex position and the average of the past and the future positions is encoded. In addition, B-meshes are generally encoded with a higher quantization level, which increases the compression ratio at the cost of accuracy.

A block diagram of the proposed scheme based on the hybrid transform-DPCM structure is shown in Figure 2.

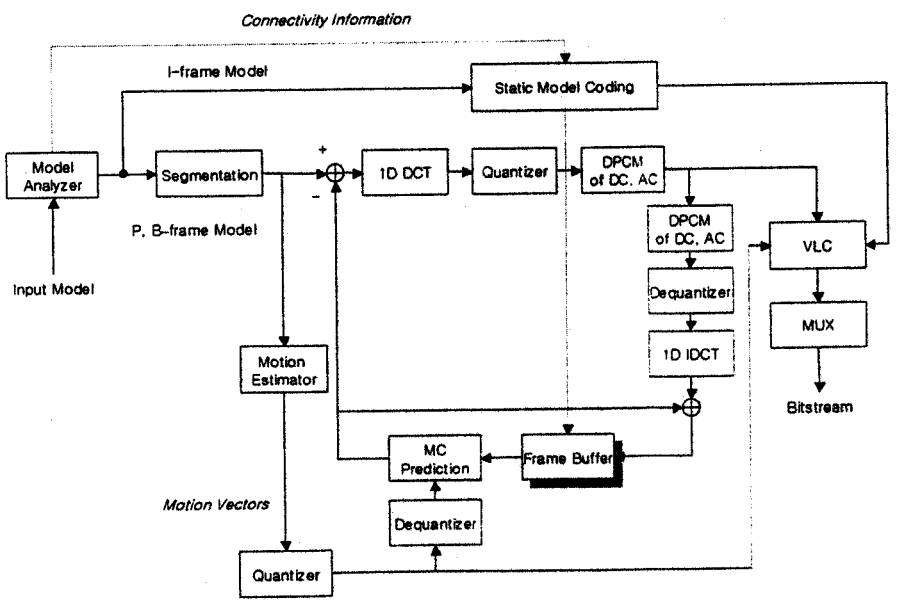
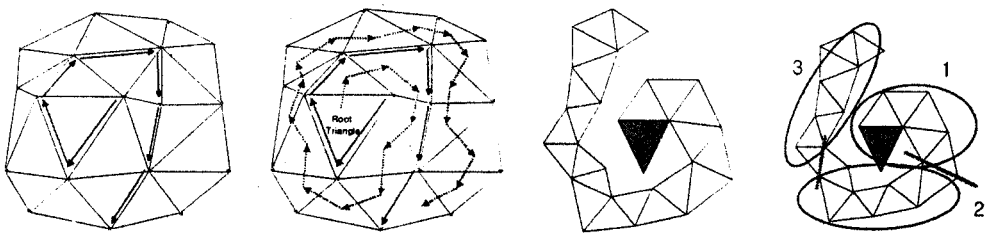


Figure 2. Block diagram of the proposed encoding scheme

The encoder basically consists of four functional blocks: analyzer, segmentation of the given 3-D model, coding of intra-meshes, and coding of inter-meshes. Once we select an I-mesh from a 3-D animation model, the 3-D static model corresponding to the I-mesh is encoded by conventional geometry compression algorithms[4-9].

For the P-mesh or the B-mesh, we decompose the given 3-D mesh into linear triangular strips. The basic notion of the triangular strip representation is to cut the connectivity of the mesh into a long strip of triangles[4,5]. We randomly select one root triangle of the given model and traverse the neighboring vertices of the root triangle in the clockwise direction, as shown in Figure 3(a). These traversed edges are called cutting edges, and we cut the mesh along cutting edges, as shown in Figure 3(b). As a result, we obtain the triangular strip, given in Figure 3(c). Then, we can easily segment the triangle trip by grouping the same number of vertices, as shown in Figure 3(d).



(a) Cut edge (b) Traversing (c) Triangle strip (d) Segmentation

Figure 3. Representation and segmentation of triangular strips

This segmentation algorithm produces several independent parts according to the topology of the 3-D model. Since this decomposition can group vertices that are close in the spatial domain, we can take advantage of their spatial and temporal coherence during the encoding operation.

After the segmentation task, we can obtain the motion vector (MV) and prediction errors for each segmented block. The prediction errors for a segmented block are transformed by 1-D discrete cosine transform (DCT)[10]. The DCT coefficients are quantized, and then coded based using the DPCM structure. Finally, the coded index is entropy coded and multiplexed.

3. INTRA-MESH CODING

Figure 4 shows the block diagram of the intra-mesh coding scheme. The proposed scheme is based on the DPCM structure. Instead of attempting to encode the absolute coordinate values of each vertex point, we take a differential coding approach to exploit strong correlation of coordinate values among neighboring vertex points. The encoder consists of three stages: preprocessing, quantization, and entropy coding.

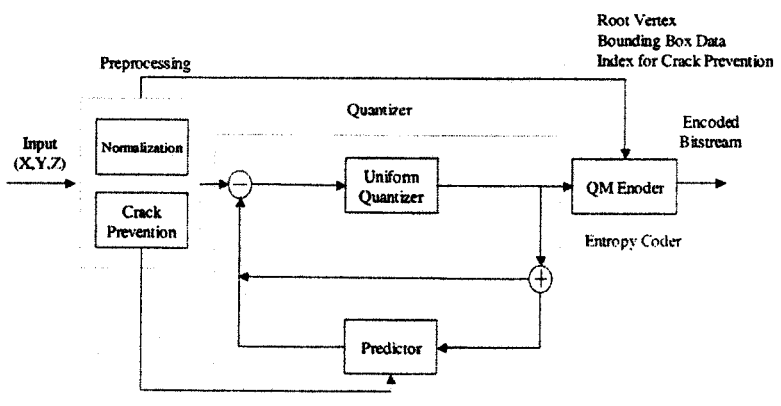


Figure 4. Block diagram of intra-mesh coding

3.1. Preprocessing Stage

A. Multiple Component Coding

In general, a 3-D model is composed of multiple components. The first vertex of each component is called as the root vertex. The root vertex plays a pivotal role as the anchor point for the entire component. Since the root vertex of each

component has no preceding vertices, this root vertex cannot be coded in the same way as the other vertices. If the root vertex is coded with its own floating-point numbers, it is not efficient. Therefore, a root vertex is also predicted by the last vertex of the previous component.

Sometimes, several components of a 3-D object are created independently and put together by sharing some vertices. More specifically, when a 3-D model is generated using a typical authoring tool, some parts are often duplicated through a cut-and-paste procedure, while other parts are grabbed out of readily usable object libraries. Whenever such independently created parts are placed to have shared vertices, a small gap can occur between boundaries of two adjoining components due to a limited precision of the authoring tool. This anomaly is called as the crack problem[7,8]. In order to solve this problem, we design a preprocessor to identify shared vertices that will be used for the crack prevention and to enforce them to have the same value. In our scheme, a list of shared vertices is delivered to the decoder. The side information enables the encoder to skip the encoding of duplicated descriptions of shared vertices. Thus, for models with many shared vertices, this approach also leads to a significant reduction in the overall bit rate.

B. Normalization

In the normalization stage, we calculate the tightest bounding box containing the input 3-D model and then normalize the vertex coordinates so that they lie within the interval [0.0, 1.0]. The normalization is needed to limit the dynamic range of prediction errors for vertex coordinates, since excessively large residuals or prediction errors can lead to visually unacceptable geometric distortion, as shown in Figure 5.

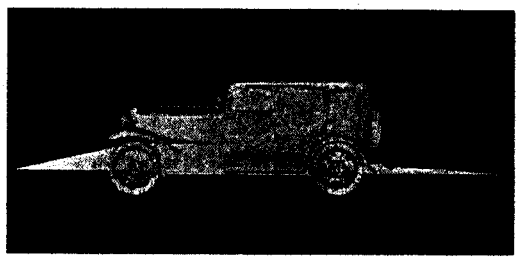


Figure 5. Example of visually unacceptable 3-D models

3.2. Predictive Encoding

In order to employ a differential coding approach, we obtain a prediction value $\hat{v}_n = (\hat{x}_n, \hat{y}_n, \hat{z}_n)$ of each vertex point $v_n = (x_n, y_n, z_n)$ based on the parallelogram prediction rule[6]. As illustrated in Figure 6, the parallelogram prediction is suitable for the triangular strip representation. When we traverse from triangle 1 to triangle 2 in the strip, the vertices v_1, v_2 and v_3 are already encoded. The opposite vertex v_4 from the common edge (v_1, v_2) is predicted from $v_1 + v_2 - v_3$. Note that the predicted vertex $\hat{v}_4$, together with its three ancestors, forms a parallelogram and belongs to the same plane. We then calculate the prediction error Δv by $\Delta v = v_4 - \hat{v}_4$.

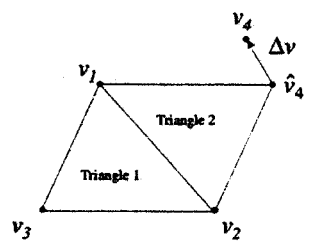


Figure 6. Parallelogram prediction rule

After the predictor estimates the current vertex position, the residual error is uniformly quantized[11]. Since we normalize the vertex position in the preprocessing stage, the dynamic range of prediction errors is within [-1.0, 1.0]. Thus, we can design a good uniform quantizer within the range of prediction errors. The quantizer index is encoded by the QM coder[12].

4. INTER-MESH CODING

In general, vertex positions in the current frame can be effectively predicted from those in the previous frame and the entropy of the prediction error is relatively small. In other words, the prediction error will contain many zeros or small values, requiring fewer bits to encode. The prediction gain can be significantly enhanced using the representative motion vector for each segmentation block.

It is assumed that the topology structure is preserved for the whole sequence of the animation model so that the motion trajectory of any vertex can be determined accurately. We define the representative motion vector $\overline{MV}_t(k)$ of the k-th segment at the t-th frame as the average of motion vectors of vertices in that segment, *i.e.*,

$$\overline{MV}_t(k) = \frac{1}{N}\sum_{n=0}^{N-1} MV_t(n) = \frac{1}{N}\sum_{n=0}^{N-1}(v_t(n) - v_{t-1}(n)), \quad k = 0,1,\ldots,M-1, \tag{1}$$

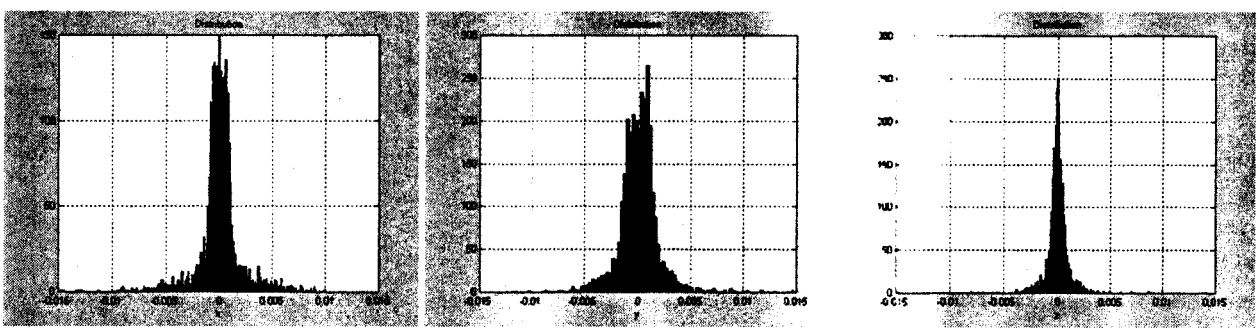
where $v_t(n)$ denotes the position of the n-th vertex in the segment at the t-th frame, N is the number of vertices in the segment, and M is the number of segments in the mesh. Since the motion vector is highly correlated with the adjacent motion vectors, we can predict the current motion vector from those of its neighboring segments. In this work, we calculate the difference vector between quantized $\overline{MV}_t(k)$ and $\overline{MV}_t(k-1)$, and encode the difference vector.

4.1. Residuals for Inter-Mesh

After defining the motion vector for each segment, we obtain residual signals by subtracting the real motion vector $MV_t(n)$ of each vertex from the representative motion vector $\overline{MV}_t(k)$, i.e.,

$$e_t(n) = \overline{MV}_t(k) - MV_t(n). \tag{2}$$

Figure 7 shows the histogram of prediction residuals $e_t(n)$ for P-mesh in the CHICKEN CROSSING sequence. We can see that prediction residuals are highly concentrated around zero.



(a) X coordinate (b) Y coordinate (c) Z coordinate

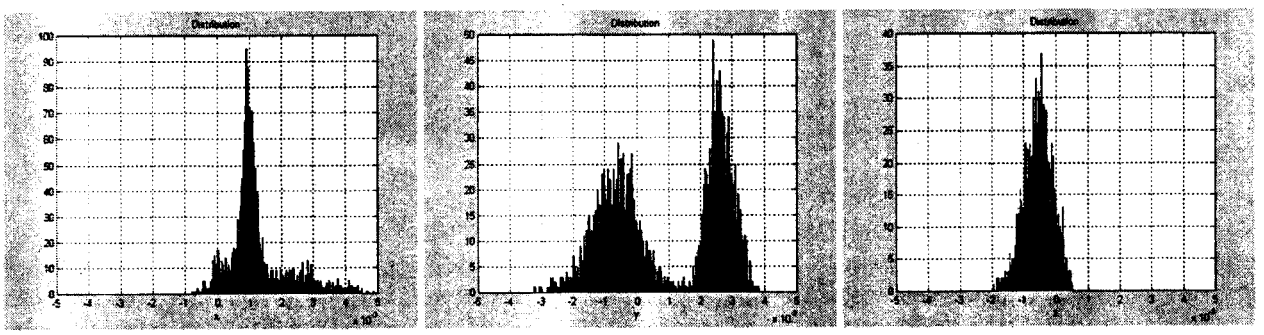
Figure 7. Distribution of residual errors for P-mesh

The mesh that is predicted only using the information in the previous frame by Eq.(1) is called as a P-mesh. However, we can further increase the prediction gain for the current mesh using bi-directional linear interpolation of vertex positions from the previous and the subsequent time instances. This mesh is called as a B-mesh. More specifically, the vertex position $v_t(n)$ in the B-mesh is predicted, vertex by vertex, from the weighted sum of the previous and the subsequent vertex positions by

$$\hat{v}_t(n) = \frac{m}{m+l} v_{t-l}(n) + \frac{l}{m+l} v_{t+m}(n). \tag{3}$$

The prediction residual error $e_t(n)$ is obtained by $v_t(n) - \hat{v}_t(n)$ for each vertex.

Figure 8 shows the histogram of prediction residuals $e_t(n)$ for B-mesh in the CHICKEN CROSSING sequence. As illustrated in Figure 8, residual errors for B-meshes are much smaller than those for P-meshes, thus requiring a lower bit rate for transmission or storage.

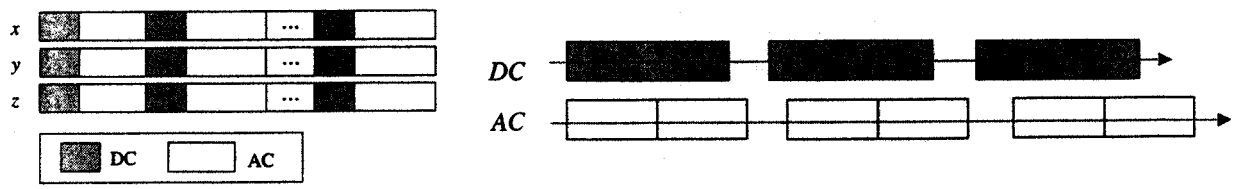


(a) X coordinate (b) Y coordinate (c) Z coordinate

Figure 8. Distribution of residual errors for B-mesh

4.2. DCT Transform

Even after we perform the segment-based motion compensation operation, residual errors $e_t(n)$ exhibit high correlation between adjacent vertices. This high spatial correlation can be effectively exploited by transform coding. In our work, we transform residual errors along the x-, y-, and z-coordinates by 1-D DCT, respectively, as shown in Figure 9(a).



(a) DCT coefficients (b) Reordered DCT coefficients

Figure 9. DCT coefficients grouping

Then, we design the bounding box quantizer by

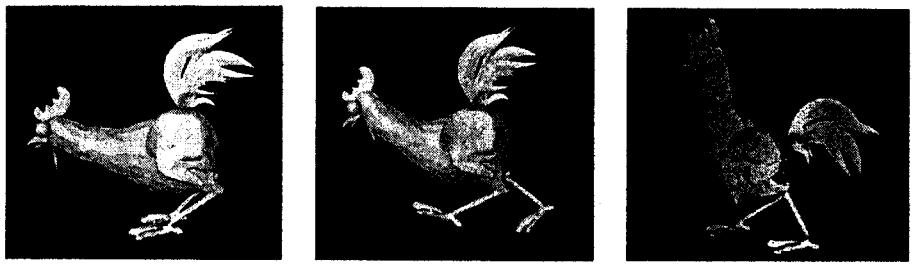
$$Q[d] = round\left\{\frac{x[d]-\min(x[d])}{\max(x[d])-\min(x[d])}\times(2^{quantization\_bits}-1)\right\}, \tag{4}$$

$$x[d] = \left\{\left(\frac{Q[d]\times(\max(x[d])-\min(x[d]))}{2^{quantization\_bits}-1}\right)+\min(x[d])\right\}.$$

Since quantized DCT coefficients for the present block are correlated with those of the preceding block, we group DC coefficients and AC coefficients separately, as shown in Figure 9(b). Then, quantized DC and AC coefficients are independently encoded using a DPCM scheme. The DPCM-coded indices can be further compressed by the context-based QM coder with the 113-state Markov model for the probability estimation.

5. EXPERIMENTAL RESULTS

We have performed computer simulations to evaluate the performance of the proposed algorithm. Figure 10 shows the CHICKEN CROSSING animation model, which comprises 3030 vertices, 400 frames and 5665 triangles. The original model requires 14,577,900 bytes (3030 vertices × 400 frame × 3 coords/vertex × 4 bytes/coord + 5665 triangles × 3 indices/triangle × 2 bytes/index), which corresponds to 36,444 bytes/frame.



(a) 1st frame (b) 100th frame (c) 200th frame

Figure 10. CHICKEN CROSSING animation model

In the field of image coding, the root mean squared error (RMSE) or the peak-signal-to-noise ratio (PSNR) is often used as an objective distortion measure. Although, these error metrics are not always consistent with the perceived signal quality, they allow us to estimate the subjective quality to some extent. However, in 3-D model coding, because excessively large quantization errors in just a few vertices may lead to a catastrophic deformation of the shape of the 3-D model, RMSE or PSNR cannot represent the overall geometry distortion appropriately. Therefore, the Hausdorff distance, which is a max–min type distance measure between two sets of points, is more meaningful in 3-D geometry compression. In order to evaluate our 3-D animation coding scheme, we adopt the Hausdorff distance that is defined as follows:

Let A and B denote the original 3-D polygonal model and the reconstructed one, respectively. Given two sets of points $A=\{a_1,a_2,....,a_m\}$ and $B=\{b_1,b_2,....,b_n\}$, the Hausdorff distance is defined as

$$H(A,B)=\max(h(A,B),h(B,A)),$$

where

$$h(A,B)=\max_{a\in A}(\min_{b\in B}\|a-b\|).$$

Since the function $h(A,B)$ is not symmetric, it is called as the directed Hausdorff distance from A to B. The Hausdorff distance $H(A,B)$ measures the degree of mismatch between two sets, as it selects the larger of the two directed distances, $h(A,B)$ and $h(B,A)$. Intuitively, if the Hausdorff distance is d, every point in A must be within the distance d from some point in B, and vice versa.

In figure 11, we compare the performances of the proposed scheme, and the MPEG-4 SNHC coder on the CHICKEN CROSSING model.

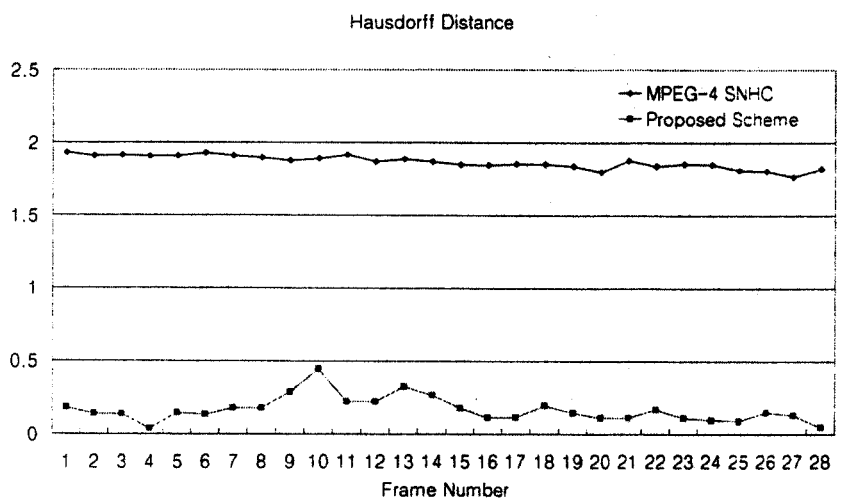


Figure 11. Performance comparison (10:1 compression ratio)

In this simulation, we assume that the connectivity data are constant across frames. Our target bitrate is 3,000 bytes per mesh, which corresponds to a compression ratio of about 10:1. For the MPEG-4 encoder, each frame of the animation is compressed and the bits required for the connectivity are counted just once at the first frame. For the proposed scheme, the temporal frame structure is set as IBBPBBP...PBBP. The segmentation size is 16 and the quantization bits are allocated as follows: 5 bits for MV, and 11, 5 and 2 bits for the vertex coordinates in I-, P- and B-meshes, respectively.

In Figure 11, the horizontal axis represents the frame number and the vertical axis shows the Hausdorff distortion. It can be seen that the proposed algorithm yields about 10 times lower distortion than MPEG-4 at the same bit rate.

Figure 12 shows the 8, 9 and 11th frames of reconstructed models for visual quality comparison. The first row shows the original model. The second row shows the reconstructed frames by the MPEG-4 codec. We can notice that the reconstructed model contains severe shape distortion. The third row shows the reconstructed frames by the proposed algorithm. We observe that the reconstructed model is visually indistinguishable from the original model. The proposed algorithm exploits the temporal coherence with motion compensated prediction and encodes prediction residuals efficiently using the DCT transform.

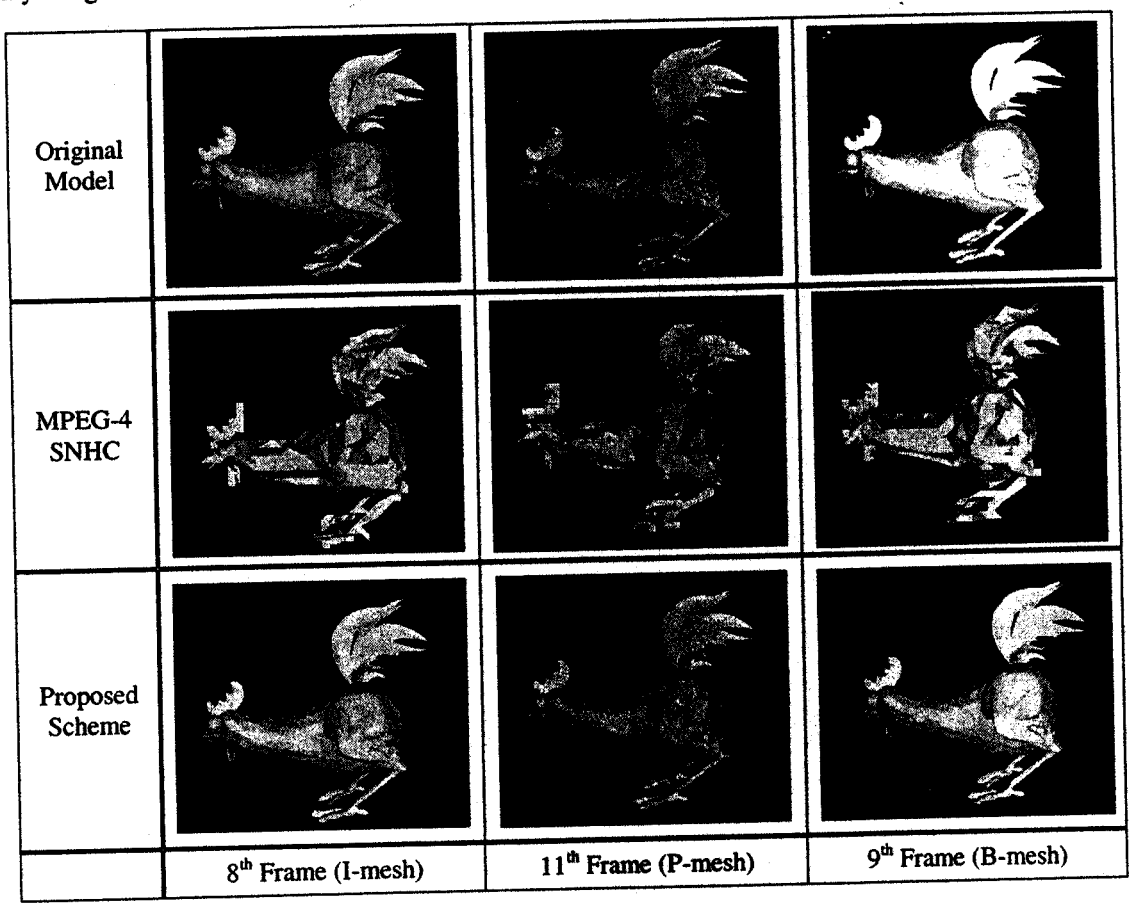


Figure 12. Reconstructed Models

6. CONCLUSIONS

In this paper, we have proposed a new compression algorithm for 3-D animation models using mesh segmentation, motion-compensated prediction, and transform coding. Experimental results demonstrated that the proposed scheme outperforms the MPEG-4 scheme, both in objective and subjective visual qualities. The proposed algorithm can be used to facilitate the real-time streaming of 3-D animation data over narrow bandwidth networks. We can further apply the rate-distortion technique in the encoding process and dynamically allocate the bitrate according to the bandwidth requirement.

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. "Description of core experiments on 3-D model coding," *ISO/IEC JTC1/SC29/WG11 MPEG98/N244rev1*, Atlantic City, October 1998.
2. J.E. Lengyel, "Compression of time-dependent geometry," in *Proc. ACM Symposium on Interactive 3-D Graphics*, Atlanta, Aug. 1999.
3. J.H. Ahn and Y.S Ho, "Segmentation and compression techniques for 3-D animation models based on motion trajectory in the spherical coordinate system," in *Proc. SPIE Visual Communications and Image Processing 2001*, pp.593-601, Jan. 2001.
4. M. Deering, "Geometry compression," in *Proc. ACM Computer Graphics, SIGGRAPH '95*, 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," in *Proc. Graphics Interface '98*, pp. 26-34, 1998.
7. J.H Ahn and Y.S. Ho, "Geometry compression of 3-D models using adaptive quantization for prediction errors," in *Proc. Picture Coding Symposium*, pp. 193-197, April 1999.
8. J. Li and C.-C. J. Kuo, "Progressive coding of 3-D graphic models," *Proc. IEEE*, vol. **86**, pp.1252–1263, June 1998.
9. J. S. Choi, Y. H. Kim, H. J. Lee, I. S. Park, M. H. Lee and C. T. Ahn, "Geometry compression of 3-D mesh models using predictive two-stage quantization," *IEEE Trans. Circuits Syst. Video Tech.*, vol.**10**, no.2, pp.312-322, Dec. 2000.
10. K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, New York, NY: Academic Press, 1990.
11. K. R. Rao and J. J. Hwang, *Techniques & Standards For Image·Video & Audio Coding*, Prentice Hall. 1996.
12. A. Gersho and R. M. Gray, *Vector Quantization and Signal Compres-sion*.Boston, MA: Kluwer, 1992, pp. 451–159.
13. W. B. Pennebaker, *JPEG Still Image Data Compression Standard*, New York: Van Nostrand Reinhold, 1993.
14. "VRML97: The virtual reality modeling language," *ISO/IEC JTC1/SC29/WG11 MPEG/N14772-1*, Dec. 1997.