International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:2, No:12, 2008

Hybrid Coding for Animated Polygonal Meshes

Jinghua Zhang, Charles B. Owen, Jinsheng Xu

***Abstract*—**A new hybrid coding method for compressing animated polygonal meshes is presented. This paper assumes the simplistic representation of the geometric data: a temporal sequence of polygonal meshes for each discrete frame of the animated sequence. The method utilizes a delta coding and an octree-based method. In this hybrid method, both the octree approach and the delta coding approach are applied to each single frame in the animation sequence in parallel. The approach that generates the smaller encoded file size is chosen to encode the current frame. Given the same quality requirement, the hybrid coding method can achieve much higher compression ratio than the octree-only method or the delta-only method. The hybrid approach can represent 3D animated sequences with higher compression factors while maintaining reasonable quality. It is easy to implement and have a low cost encoding process and a fast decoding process, which make it a better choice for real time application.

***Keywords*—**animated polygonal meshes, compression, delta coding, octree.

I. Introduction

3D graphics are rapidly achieving mainstream success, both in motion pictures and computer gaming. But, as 3D models grow more complex, more detailed, and longer format, it becomes increasingly difficult to efficiently store and transmit these models. The focus of this paper is to design an efficient and powerful algorithm for compressing and transmitting animated geometric data. Geometry compression is the compression of the 3D geometric data that provides a computer graphics system with the scene description necessary to render images. Animated 3D models require even larger memory and transmission bandwidth since a sequence consists of a large number of frames and every frame is a large static 3D object. Animated geometry compression is the compression of temporal sequences of geometric data. Therefore, efficient compression techniques need to be applied to the 3D animation sequence before distributing it over the network.

The general field of geometry compression is concerned with efficient compression of 3D data, be it animated or static. Most initial work has been focused on static geometry. Deering first proposed geometry compression techniques in 1995 with compression ratios of 6-10 to 1 [1]. Since that time, various strategies for geometry compression have been reported [2-5]. Boosen provides a comprehensive overview of the reported geometry compression techniques [6]. Another geometry compression survey paper was written by Shikhare [7]. Luebke wrote a survey on polygonal simplification algorithms [8]. All these techniques focus on static geometry. Less attention has been paid to animated geometry compression [9-14].

Lengyel presented a prediction compression method that splits vertices into sets and uses affine transformations to approximate the vertex paths [9]. His method is effective only when the animation is well represented by the supported transformations. Alexa and Müller proposed an interpolation predictor to represent animations by principal components [10]. Principal Component Analysis (PCA) makes their approach expensive. It requires significant memory usage and processing time. The PCA approach cannot achieve high compression ratios if the number of frames is significantly smaller than the number of vertices. Ibarria and Rossignac proposed a time-space predictor for all the vertices and all the frames [11]. Briceño et al. proposed an algorithm to generate geometry video from 3D animated meshes [12]. They extended the geometry images proposed by Gu et al.[15]. To construct a geometry video, all the frames in the animation sequence are replaced by the corresponding geometry images. Conventional video compression techniques are applied to encode geometry video.

Zhang and Owen proposed octree-based animated geometry compression method [14].

The goal of this paper is to design hybrid compression algorithms for animated geometry that improve on existing methods, both in compression ratios and quality. The hybrid approach combines the octree approach [14] with delta coding method to achieve higher compression ratio given similar quality requirement. Similar to the octree approach, this paper focuses on the compression of vertex positions for each frame in the animation sequence and the connectivity is assumed to be the same for all the frames.

This paper is organized as follows. Section II reviews octree-based motion representation method. Section III describes a delta coding approach to represent animated geometric data. It presents evaluation results between the octree-based approach and the delta coding approach in terms of compression ratio. Section IV presents a hybrid coding approach that combines the octree-based approach and delta coding approach. By selectively incorporating these two mechanisms, the method is able to achieve better performance than any of these methods individually. Evaluation results for the mechanism are presented in detail. Section V includes the

Jinghua Zhang is with the Department of Computer Science, Winston Salem State University, Winston-Salem, NC 27110 USA; (corresponding author) Phone: 336-750-3324; Fax: 336-750-2499; E-mail: zhangji@wssu.edu.

Charles B. Owen is with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824 USA; E-mail: cbowen@cse.msu.edu).

Jinsheng Xu is with the Department of Computer Science, North Carolina A & T University, Greensboro, NC, 27401 USA; E-mail: jxu@ncat.edu.

conclusions and describes possible extensions of the current work.

II. Octree-based animated geometry compression

The octree approach takes advantage of the spatial and temporal coherence present in the data. Two consecutive frames are needed to generate a reduced set of motion vectors that represent the motion from the previous frame to the current frame. The motion vectors are used to predict the vertex positions in the decoder side for each frame except for the first frame. The process generates a hierarchical octree motion representation for each frame.

A. Encoding process

In the motion modeling process, each frame except the first frame is represented by a set of motion vectors, which are stored in an octree. In a client–server scenario, when the server wishes to send an animated 3D model to the client, it will send the encoded first frame (using an intraframe encoding method) together with an encoded octree for each of the subsequent frames rather than sending vertex positions for each new frame. The encoded octree is significantly smaller than the corresponding original frame. The vertex locations for each frame are approximated by the corresponding octree on the client side.

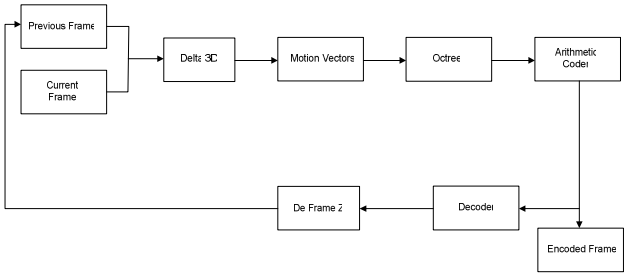

Fig. 1Logical flow of encoding process of the octree approach

The logical flow of the encoding process is illustrated in Fig. 1. To encode the current frame, the previous frame needs to be available to the decoder, so the first frame is encoded using an intra-frame coding technique. A complete system will include occasional intra-coded frames so as to provide synchronization points, as in MPEG video. The previous and current frames are used to generate vertex delta values representing the differential motion between the frames. Motion vectors are obtained by using a Least Square Error estimation method to estimate the motion of enclosed vertices. To continue encoding the following frames, a local decoder is called to generate a decoded version of the current frame as known to the receiving system, which can be used to encode the next frame at the encoder side. This process is repeated for each new frame. For details, please refer to the paper [14].

III. Delta approach

This section presents a new *delta coding approach* and evaluation of the performance of this method for animated geometry compression. The delta coding approach generates 3D delta values from two consecutive frames. The method achieves the compression by quantizing the 3D delta values. Different frames in the animation sequence can use different quantization levels. Quantized values are coded using arithmetic coding to achieve further data reduction. This method is simple and easy to implement. The delta coding method does not achieve the consistent level of the performance demonstrated by the octree approach. However, in limited cases the delta approach does outperform. Hence, it is explored here in isolation, and then incorporated into a hybrid compression approach in Section IV.

A. Delta encoding process

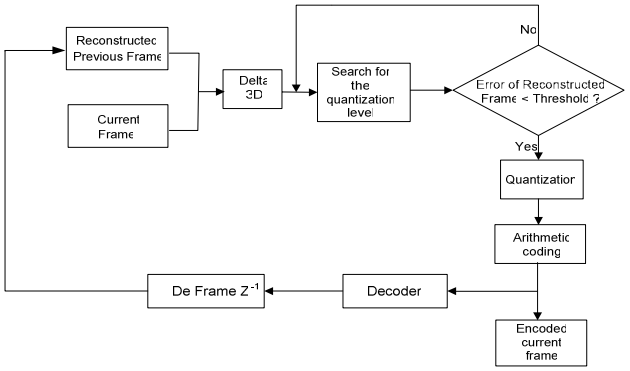

Fig. 2 Logical flow of the encoding process of the delta method.

The logical flow of the encoding process is illustrated in Fig. 2. To encode the current frame, the previous reconstructed frame needs to be available in the encoder side. Therefore the first frame is encoded differently from other frames in the sequence. It is encoded using an intra-frame coding method. The first step in the delta encoding process is to generate delta 3D values between two consecutive frames, namely the reconstructed previous frame and the current original frame. Delta 3D data presents the differential motion between two frames. After the first step, the delta approach seeks the best quantization level with the minimum number of bits for the current delta 3D that satisfies a predefined threshold. After the number of bits for the quantization is determined, the linear quantization over the range of each possible vertex is applied to the delta 3D as illustrated in (1). The quantized values are encoded using arithmetic coding to further achieve data reduction. To this step, the current frame is completely delta encoded. In the local decoding step, a linear inverse quantization is applied to the quantized samples as shown in (2). The above steps are repeated for every frame except for the first frame in the animation sequence.

$$\bar{x} = \left\lfloor 2^{q}\left(\frac{x - x_{\min}}{x_{\max} - x_{\min}}\right)\right\rfloor \tag{1}$$

$$x = \frac{(x_{\max} - x_{\min}) * \bar{x}}{2^{q}} + x_{\min} \tag{2}$$

B. Evaluations

In order to compare the results of different approaches, the same test data sets as those in octree approach are used to evaluate the performance of the delta approach. The data sets include "Chef", "Chicken Crossing" and "Dance". The Chef data was generated from a 3D Studio Max animation. "Chicken Crossing" was created by Andrew Glassner et al.[15]. The "Dance" sequence was created by the MIT CSAIL Graphics Lab. Table. 1 shows the properties of "Chef", "Chicken Crossing" and "Dance" animation sequences.

TABLE.I TEST DATA SETS INFORMATION

Data sets Properties	Chef animation	Chicken animation	Dance animation
Number of vertices	4241	3030	7061
Number of triangles	8162	5664	14118
Number of frames	75	400	201
Total size (bytes)	3,816,900	14,544,000	17,031,132

Compression ratio results of the delta approach

The compression ratio results for Chef, Chicken Crossing and Dance animation sequences given different threshold settings are shown in Table. 2, 3 and 4 respectively.

TABLE.II COMPRESSION RATIOS OF THE CHEF ANIMATION USING THE DELTA APPROACH. THE CHEF ANIMATION HAS 75 FRAMES AND EACH FRAME HAS 8162 TRIANGLES AND 4241 VERTICES. THE OBJECT SIZE IS 440 UNITS IN X DIMENSION, 148 UNITS IN Y DIMENSION AND 455 UNITS IN Z DIMENSION. ORIGINAL FILE SIZE IS 3,816,900 BYTES. MAX DISTANCE THRESHOLD =P% *455 UNITS, WHERE P=1, 2, 3, …6

Chef sequence	Max Distance					
	1%	2%	3%	4%	5%	6%
Compression Ratio	17:1	21:1	31:1	32:1	32:1	34:1

TABLEIII COMPRESSION RATIOS OF THE CHICKEN ANIMATION USING THE DELTA APPROACH. THE CHICKEN CROSSING ANIMATION HAS 400 FRAMES AND EACH FRAME HAS 5664 TRIANGLES AND 3030 VERTICES. THE OBJECT SIZE IS 2.556 UNITS IN X DIMENSION, 2.23 UNITS IN Y DIMENSION AND 1.07 UNITS IN Z DIMENSION. THE ORIGINAL FILE SIZE IS 14,544,000 BYTES. MAX DISTANCE THRESHOLD= P% *2.556 UNITS, WHERE P=2.5, 5, 7.5, 10, 15, 20

Chicken sequence	Max Distance					
	2.5%	5%	7.5%	10%	15%	20%
Compression Ratio	21:1	26:1	28:1	30:1	33:1	35:1

TABLE IV COMPRESSION RATIOS OF THE DANCE ANIMATION USING THE DELTA APPROACH. THE DANCE ANIMATION HAS 201 FRAMES AND EACH FRAME HAS 14118 TRIANGLES AND 7061VERTICES. THE OBJECT SIZE IS 0.0758 UNITS IN X DIMENSION, 0.172 UNITS IN Y DIMENSION AND 0.074 UNITS IN Z DIMENSION. ORIGINAL FILE SIZE IS 17,031,132 BYTES. MAX DISTANCE THRESHOLD=P%*0.172 UNITS, WHERE P=1, 2, 3, 4, 5, 6

Dance sequence	Max Distance					
	1%	2%	3%	4%	5%	6%
Compression Ratio	10:1	15:1	18:1	19:1	21:1	23:1

Overall compression ratio comparison: delta vs. octree

Compression ratio is one of the most important measurements to evaluate the performance of an approach. The overall compression ratio comparison between the delta approach and the octree approach are shown in Fig. 3, 4 and 5respectively.

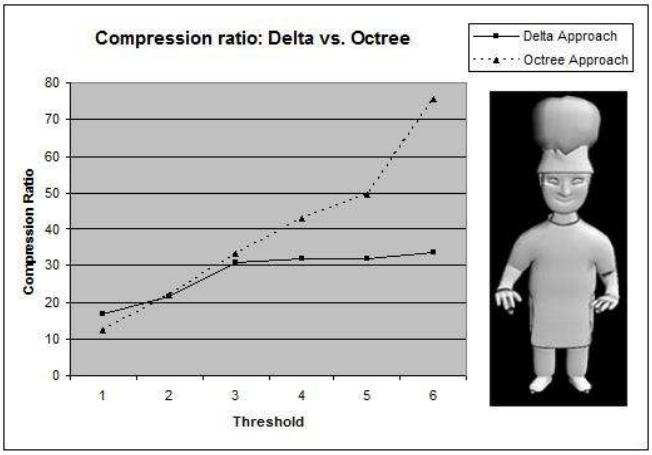

Fig. 3 Compression ratio comparison between the delta and octree approach using the Chef animation.

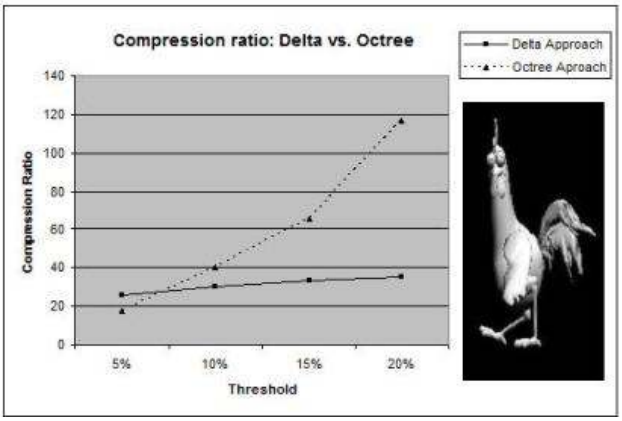

Fig. 4 Compression ratio comparison between the delta and octree approach using the Chicken animation.

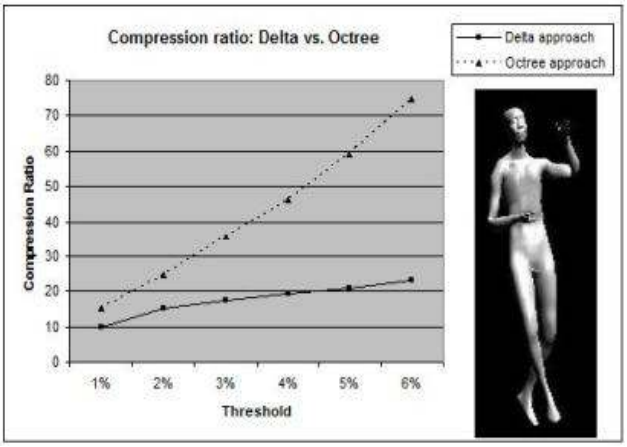

Fig. 5 Compression ratio comparison between the delta and octree approach using the Dance animation.

Frame-wise compression ratio comparison: delta vs. octree

The overall compression ratio comparison results show that the delta coding method does not achieve the consistent level of performance demonstrated by the octree approach. However, in limited cases the delta approach does outperform. The frame-wise compression ratio comparisons were conducted. Fig. 6 shows the frame-wise compression ratio comparison between the delta and octree approach given the same threshold settings for the Chef animation. In this case, the delta approach is better than the octree approach for 18% frames. Fig. 7 shows the frame-wise comparison results for the Chicken Crossing animation. For 22% frames, the delta approach can achieve higher compression ratios given the same threshold settings.

Fig. 8 shows the comparison results for the Dance animation. The delta approach is better than the octree approach for 9% frames.

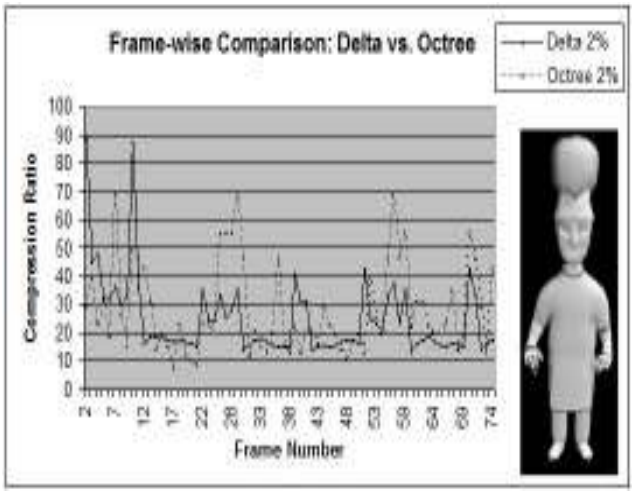

Fig. 6 Frame-wise compression ratio comparison between the delta and octree approach using the Chef animation given the same threshold settings. (threshold=2%)

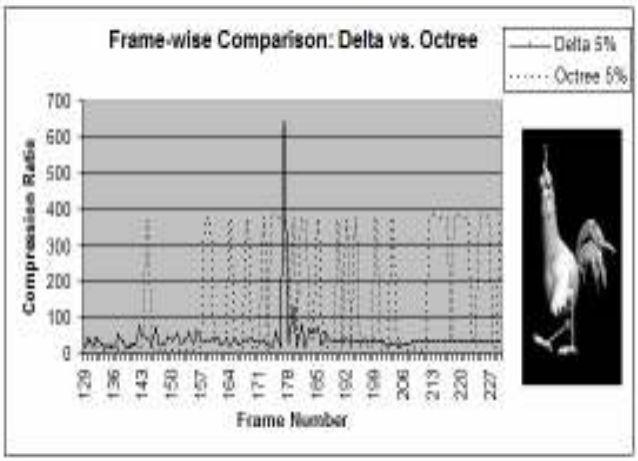

Fig. 7 Frame-wise compression ratio comparison between the delta and octree approach using the Chicken animation given the same threshold settings. (threshold=5%).

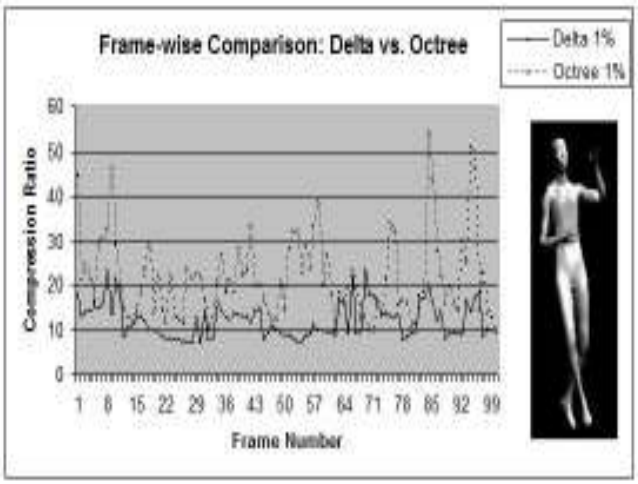

Fig. 8 Frame-wise compression ratio comparison between the delta and octree approach using the Dance animation given the same threshold settings. (threshold=1%).

The delta approach is straight forward and is easy to implement. But it has its limitation. The big problem is that it can not achieve consistently high compression ratios. From the compression ratio comparisons with the octree approach, the compression ratio curves for the delta approach stop growing after they achieve certain level. Obviously the delta approach could not be used alone to achieve animated geometry compression.

However, frame-wise compression ratio comparison results between the delta approach and octree approach showed that the delta approach could achieve higher compression ratios for some frames of the animation sequence. Although the delta approach does not have good performance when it is used alone, it may increase the performance of the octree approach if they are combined. The delta approach, like the octree approach, is a pair-wise approach and can be applied to frames in isolation. In the following section, a hybrid approach is presented based on the combination of these two mechanisms.

IV. Hybrid Approach

This section presents a *hybrid coding* approach to compress animated geometric data by combining the octree and delta coding approaches. The motivation of this approach comes from the observation that the encoded octree size is sometimes larger than the encoded size of the delta encoding. The encoding process of the hybrid approach is presented. The results of this approach and the comparison among the octree approach, delta approach and hybrid approach are also included.

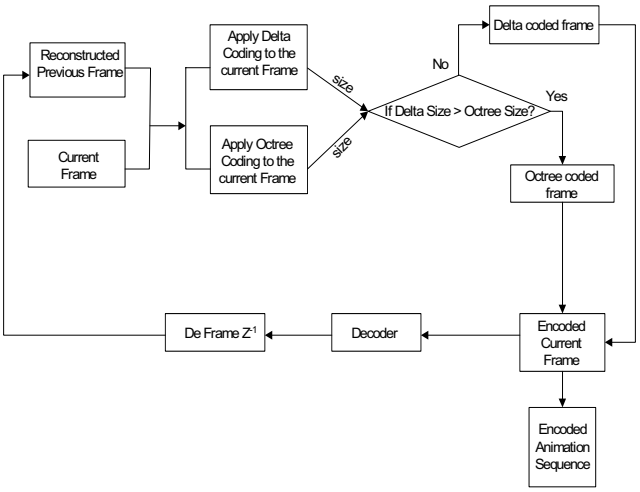

Fig. 9 Logical flow of the hybrid encoding process.

A. Hybrid encoding process

Fig. 9 illustrates the logical flow of the hybrid encoding process. This approach uses a threshold to control the quality of the output. The threshold is used to control the subdividing process of the octree coding and the number of bits used in the quantization step of the delta coding. For the octree encoding process, if the error (Euclidean distance) of any reconstructed vertex is larger than the threshold, it will subdivide the current bounding box and repeat the motion vector computation step. For the delta encoding process, if the error of any reconstructed vertex is larger than the threshold, it will continue search for the best quantization level that drops the reconstruction error below the threshold. Therefore, any reconstructed vertex in any frame in the reconstructed animation sequence satisfies the criteria that the error between this vertex and the corresponding original vertex is smaller than the threshold as shown in (3). The threshold in the hybrid coding method is set to be the maximum allowed distance between the reconstructed vertex and its corresponding original vertex. It is represented by the percentage of the edge length of the initial cubic bounding box of the object as illustrated in (4).

$$\forall v \in current\ cell$$
$$\sqrt{(v_x - v_x')^2 + (v_y - v_y')^2 + (v_z - v_z')^2} < threshold \quad (3)$$

v and v' represent the original vertex and the corresponding reconstructed vertex.

$$Threshold = \%Length \quad (4)$$

To encode each frame except the first frame in the sequence, both the octree coding method and delta coding method are applied in parallel. If the size of the encoded delta is smaller than the encoded octree, the encoded octree is replaced with the encoded delta. A single bit in the output data stream is used to indicate if it is an octree coding or delta coding.

B. Evaluations

This section presents the compression ratio results of the hybrid approach. It also includes the compression ratio comparison among the hybrid, octree and delta approach given the same threshold settings.

Hybrid approach results

The hybrid method generates a mixed sequence of the octree encoded and delta encoded frames. Table. 5 shows the percentage of the octree encoded and delta encoded frames for different animation sequences and different thresholds. The delta encoding contributes significantly to the hybrid method. It takes up to 41% of the encoded frames. However, as the threshold increases, the percentage of the delta coded frames in the encoded sequence decreases. This is because the performance of the octree approach is much better than the delta coding approach as the threshold increases.

Table V Percentage of the octree encoded and delta encoded frames

Hybrid	Threshold	Octree	Delta
Chef	1%	59%	41%
	2%	82%	18%
	3%	86%	14%
	4%	90%	10%
Chicken	2.5%	64%	36%
	5%	78%	22%
	7.5%	85%	15%
	10%	88%	12%
Dance	1%	91%	9%
	2%	92%	8%
	3%	95%	5%
	4%	97%	3%

Table 6, 7 and 8 present the compression ratios of the Chef, Chicken Crossing and Dance animation sequence using the hybrid approach. Some reconstructed sample frames of the Chef, Chicken Crossing and Dance animation sequence using the hybrid approach are shown in Fig. 10, 11 and 12 respectively.

TABLE VI COMPRESSION RATIOS OF THE CHEF ANIMATION USING THE HYBRID APPROACH. THE CHEF ANIMATION SEQUENCE HAS 75 FRAMES AND EACH FRAME HAS 8162 TRIANGLES AND 4241 VERTICES. THE OBJECT SIZE IS 440 UNITS IN X DIMENSION, 148 UNITS IN Y DIMENSION AND 455 UNITS IN Z DIMENSION. ORIGINAL FILE SIZE IS 3,816,900 BYTES. MAX DISTANCE THRESHOLD =P% *455 UNITS, WHERE P=1, 2, 3, …6

Hybrid Chef animation	Max Distance					
	1%	2%	3%	4%	5%	6%
Compression Ratio	22:1	32:1	41:1	65:1	78:1	96:1

TABLE VII COMPRESSION RATIOS OF THE CHICKEN CROSSING ANIMATION USING THE HYBRID APPROACH. THE CHICKEN CROSSING ANIMATION HAS 400 FRAMES AND EACH FRAME HAS 5664 TRIANGLES AND 3030 VERTICES. THE OBJECT SIZE IS 2.556 UNITS IN X DIMENSION, 2.23 UNITS IN Y DIMENSION AND 1.07 UNITS IN Z DIMENSION. THE ORIGINAL FILE SIZE IS 14,544,000 BYTES. MAX DISTANCE THRESHOLD= P% *2.556 UNITS, WHERE P=2.5, 5, 7.5, 10, 15, 20

Hybrid Chicken animation	Max Distance					
	2.5%	5%	7.5%	10%	15%	20%
Compression Ratio	29:1	47:1	60:1	77:1	112:1	153:1

TABLE VIII COMPRESSION RATIOS OF THE DANCE ANIMATION USING THE HYBRID APPROACH. THE DANCE ANIMATION HAS 201 FRAMES AND EACH FRAME HAS 14118 TRIANGLES AND 7061VERTICES. THE OBJECT SIZE IS 0.0758 UNITS IN X DIMENSION, 0.172 UNITS IN Y DIMENSION AND 0.074 UNITS IN Z DIMENSION. ORIGINAL FILE SIZE IS 17,031,132 BYTES. MAX DISTANCE THRESHOLD=P%*0.172 UNITS, WHERE P=1, 2, 3, 4, 5, 6

Hybrid Dance animation	Max Distance					
	1%	2%	3%	4%	5%	6%
Compression Ratio	17:1	28:1	37:1	48:1	62:1	72:1

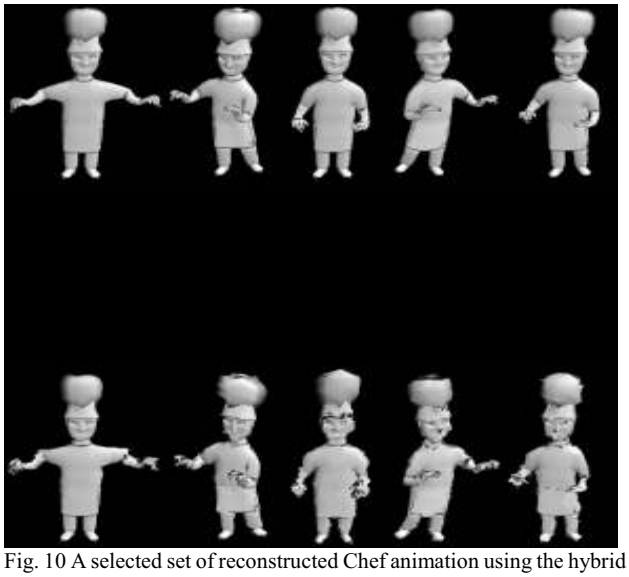

Fig. 10 A selected set of reconstructed Chef animation using the hybrid approach. The top row is the original animation. The reconstructed animation by using *Max Distance threshold* = 1% and 5% are shown in row 2and 3 respectively. The compression ratio is 22:1 and 78:1 in row 2 and 3 respectively

Fig. 11 A selected set of reconstructed Chicken Crossing animation using the hybrid approach. The top row is the original one. The reconstructed animation by using *Max Distance threshold* = 5%, 10% and 15% are shown in row 2, 3, and 4 respectively. The compression ratio is 47:1, 77:1 and 112:1 in row 2, 3 and 4 respectively.

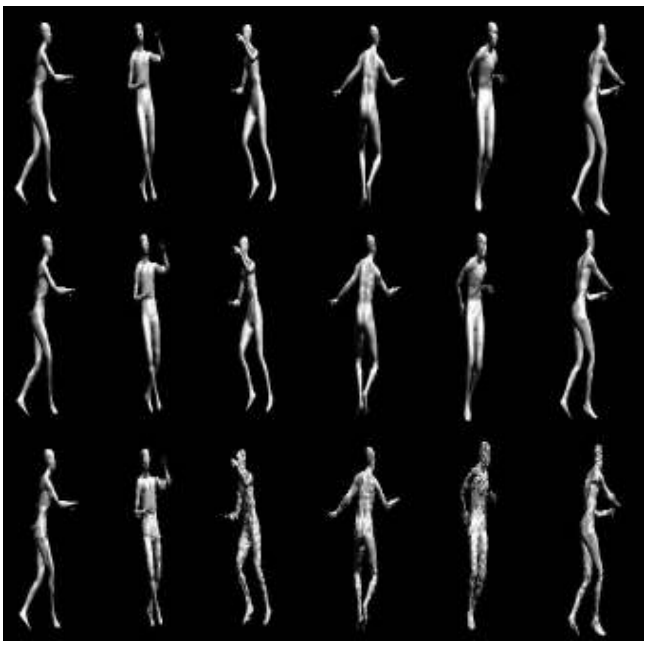

Fig. 15 A selected set of reconstructed Dance animation using the hybrid approach. The top row is the original animation. The reconstructed animation by using max distance threshold = 1% and 3% are shown in row 2 and 3 respectively. The compression ratio is 17:1 and 37:1 in row 2 and 3 respectively

Compression ratio comparison: hybrid, octree and delta

This section presents the compression ratio comparison results among the hybrid, octree and delta approaches. Fig. 13, 14 and 15 show the results using the Chef, Chicken Crossing and Dance animation respectively.

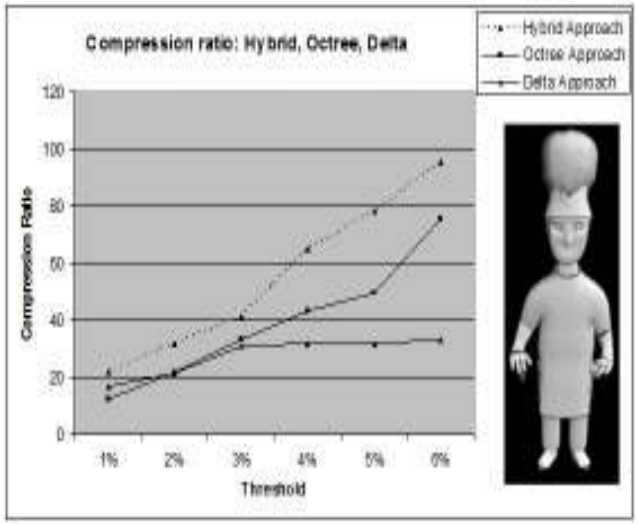

Fig. 13 Compression ratio comparison among the hybrid, octree and delta approach using the Chef animation.

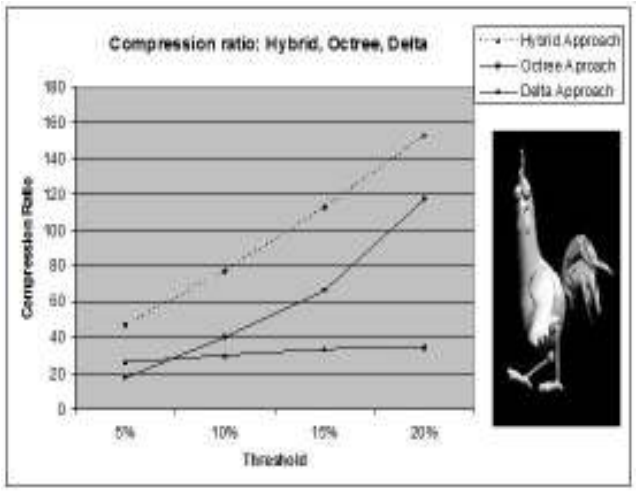

Fig. 14 Compression ratio comparison among the hybrid, octree and delta approach using the Chicken animation.

V. Conclusion and Future Work

This section presents some conclusions that can be drawn from this paper and some possible future extensions to the current work. A delta coding mechanism was introduced that represents the data coherence in a 3D animation sequence and is efficient as a compression mechanism on some pair-wise frame sequences. This approach was evaluated by comparing its performance to the octree-based approach in terms of overall and frame-wise compression ratios. In some pair-wise frame cases it did out-perform the octree-based method. However, it is shown that it is not suitable as a general solution because the compression ratio does not scale with the number of vertices of a frame. Although the delta coding approach could not achieve good performance when it is used alone, it can increase the performance of the octree-based approach when they are combined.

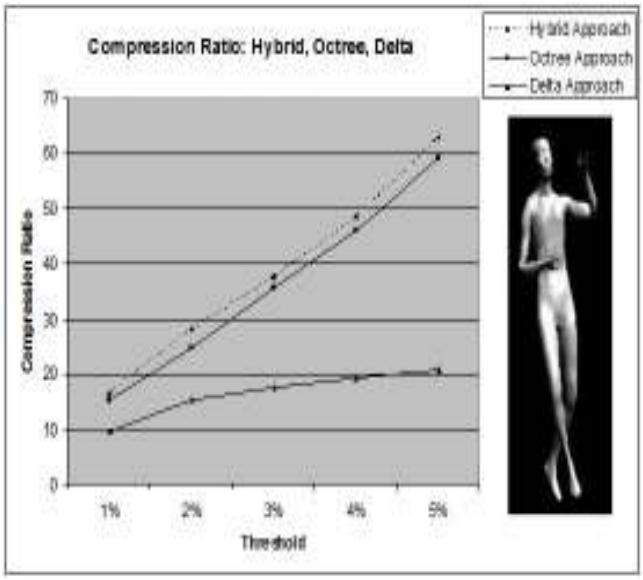

Fig. 15 Compression ratio comparison among the hybrid, octree and delta approach using the Dance animation.

A hybrid approach was then presented that combines the octree-based method and delta coding method. The results show that the hybrid approach outperforms both the octree-based and delta-encoding approaches in terms of compression ratio. This hybrid approach provides a better technique to predict the differential motion of any consecutive frames in the animation sequence.

One possible extension to the current work is to implement a frame-based geometry system using hybrid coding approach with particular emphasis on support for practical playback requirements including stream joining, playback controls, and error correction.

Acknowledgment

We want to thank Andrew Glassner for giving us access to the Chicken data. The chicken character was created by Andrew Glassner, Tom McClure, Scott Benza, and Mark Van Langeveld. This short sequence of connectivity and vertex position data is distributed solely for the purpose of comparison of geometry compression techniques. We also want to thank MIT CSAIL Graphics Lab for the dance and cow sequences.

References

[1] M. Deering, "Geometry Compression," Proceedings of ACM SIGGRAPH'95, pp. 13-20, 1995.

[2] G. Taubin and J. Rossignac, "Geometric compression through topological surgery," ACM Transactions on Graphics, vol. 17, no. 2, pp. 84-115, 1998.
[3] G. Taubin and J. Rossignac, "3D Geometry Compression," ACM SIGGRPAH'98 Course Notes 21, Orlando, Florida, 1998.
[4] C. Touma and C. Gotsman, "Triangle Mesh Compression.," Proceedings of 24th Conference on Graphics Interface (GI-98), pp. 26-34, San Francisco, 1998.
[5] J. Rossignac, "Edgebreaker: Connectivity Compression for Triangle Meshes," IEEE Transactions on Visualization and Computer Graphics, vol. 5, no. 1, pp. 47-61, 1998.
[6] F. Bossen, "On The Art Of Compressing Three-Dimensional Polygonal Meshes And Their Associated Properties," Ph.D. Thesis, cole Polytechnique Fdrale de Lausanne (EPFL), 1999.
[7] D. Shikhare, "State of the Art in Geometry Compression," National Centre for Software Technology, 2000.
[8] D. Luebke, "A survey of polygonal simplification algorithms," Dept. Computer Science, University of North Carolina, Chapel Hill, Tech. Report TR97-045, 1997.
[9] J. E. Lengyel, "Compression of Time-Dependent Geometry," Proceedings of ACM Symposium on Interactive 3D Graphics, pp. 89 -95, New York, ACM Press, 1999.
[10] M. Alexa and W. Müller, "Representing Animations by Principal Components," Computer Graphics Forum, vol. 19, no. 3, pp. 411-418, 2000.
[11] L. Ibarria and J. Rossignac, "Dynapack:Space-Time Compression of the 3D animations of triangle meshes with fixed connectivity," Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation, 2003.
[12] H. M. Briceño, P. V. Sander, L. McMillan, S. Gortler, and H. Hoppe, "Geometry Videos: A new representation for 3D Animations," Proceedings of Eurographics/SIGGRAPH Symposium on Computer Animation(SCA03), San Diego, California, 2003.
[13] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," Computers & Graphics, vol. 28, pp. 25-34, 2004.
[14] J. Zhang and C.B. Owen, “Octree-based Animated Geometry Compression”, Computers & Graphics, Volume 31, Issue 3, pp 463-479, June 2007.
[15] A. Glassner, T. McClure, S. Benza, and M. V. Langeveld, "Chicken Crossing," SIGGRAPH Video Review, 1996.