AMDO2006 Conf.proceedings (Ed. Prales,F.J., Fischer,R.B.), Spain, Springer Verlag, pp.29-37, ISSN 0302-9743, 2006

CODDYAC: CONNECTIVITY DRIVEN DYNAMIC MESH COMPRESSION

Libor Váša, Václav Skala {lvasa|skala@kiv.zcu.cz}

Department of Computer Science and Engineering
Faculty of Applied Sciences
University of West Bohemia

ABSTRACT

Compression of 3D mesh animations is a topic that has received increased attention in recent years, due to increasing capabilities of modern processing and displaying hardware. In this paper we present an improved approach based on known techniques, such as principal component analysis (PCA) and EdgeBreaker, which allows efficient encoding of highly detailed dynamic meshes, exploiting both spatial and temporal coherence. We present the results of our method compared with similar approaches described in literature, showing that using our approach we can achieve better performance in terms of rate/distortion ratio.

Index Terms **—** Dynamic mesh, compression, PCA, EdgeBreaker, coherency exploitation, entropy

1. INTRODUCTION

Dynamic mesh is a common term used for a series of static triangular meshes that represents a development of some surface in time. Usually, two additional assumptions are made about the dynamic mesh:

- every mesh in the series has the same connectivity, i.e. there is an one-to-one correspondence of vertices from frame to frame of the animation
- the animation represents some physical process, i.e. there are no sudden changes in the geometry of the subsequent frames of the animation.

The problem of compressing such data structure has been addressed in the past by various approaches. One class of algorithms is based on various spatio-temporal predictors, which are an extension of the spatial predictors, such as the parallelogram predictor [1]. The spatio-temporal predictors apart from using positions of vertices in the current frame also use vertex positions form one or more preceding frames. One of the first attempts in this field are the two spatio temporal predictors introduced with the DynaPack algorithm by Ibarria and Rossignac [2], the ELP predictor, and the Replica predictor. The connectivity driven predictor proposed by Stefanoski [3] is another example of a spatio-temporal predictor.

The approach suggested by Mueller et al. [4] can be seen as an augmentation of the predictor based technique by using spatial subdivision by an octree structure according to the size of prediction residuals.

A slightly different approach has been proposed by Payan et al. [5]. Their approach is targeted on exploiting the temporal coherence by wavelet decomposition of the vertex trajectories, followed by a rigorous bit-allocation process that optimizes the rate-distortion ratio.

Alexa and Mueller [6] have suggested a PCA-based approach based on reducing the space of shapes (frames). Their approach first requires a rigid motion compensation (the least-squares method is used to find an affine transform that best fits the given frame with respect to the first frame of the animation), and each frame is then expressed as a linear combination of uncorrelated basis of the space of possible shapes, called EigenShapes. The method is based on singular value decomposition (SVD), as the space dimension (number of vertices) is much greater than the number of samples (number of frames).

Several improvements of the original Alexa's method have been proposed. First, Karni and Gotsman [7] have suggested using linear prediction coding (LPC) on the PCA coefficients in order to exploit temporal coherence of the data. Subsequently, Sattler et al. [8] have proposed an application of PCA onto trajectories instead of shapes, and combining the approach with spatial clustering which allows exploitation of spatial coherence.

A combination of PCA and local coordinate frame (LCF) has been suggested by Amjoun [9]. The approach is based on spatial clustering driven by the magnitude of vertex coordinates expressed in each cluster's LCF.

Recently, Mamou et al. [10] have proposed a method based on automated skinning, which also uses the spatial clustering. An affine transform that best fits each cluster movement in time is found, and each vertex is described by a weighted sum of the transforms associated with the cluster the vertex belongs into, and of the transforms assigned with the neighboring clusters.

More detailed description of the compression approaches can be found in [11]. Generally, the methods can be classified according to what approach is used to

exploit spatial and temporal coherence of the input data. Following table summarizes the methods mentioned above:

	spatial coherence	temporal coherence
Ibarria	predictor (three vertices)	predictor (one frame)
Stefanoski	predictor (three vertices)	predictor (one frame)
Mueller	octree	predictor (one frame)
Payan	wavelet decomposition	none
Alexa	PCA	None
Karni	PCA	LPC
Sattler	clustering	PCA
Amjoun	clustering, LCF	PCA
Mamou	skinning	least-squares affine transform fitting

The goal of dynamic mesh compression is to exploit as much spatial and temporal coherence as possible, and indeed the methods that combine most efficient methods for spatial and temporal coding provide the best results. However, the problem of some of the methods, such as EigenShapes based PCA method by Karni, is that there is a significant overhead of the PCA basis, where no coherence is exploited at all. The coordinates of the EigenShapes must be encoded very precisely, and no coherence between the EigenShapes can be used for better compression, because the EigenShapes are uncorrelated.

2. CODDYAC ALGORITHM DERIVATION

The current movie industry modeling tools usually work with low-poly models, which are only smoothed using some subdivision technique as the last step of the modeling process. However, the copyright issues and intellectual property protection will most likely enforce transmission of the final, smoothed, high-poly models. Also the current scanning techniques provide rather dense meshes, and the trend in computer games towards higher precision models is also well known. On the other hand, the length of the animation sequence is dictated by the rules of movie editing. It is well known that usual movie scene is not longer than a few seconds, being followed by a cut onto a different actor or location, in order to maintain the observer's attention.

Generally, we can expect growth of the complexity of separate frames, i.e. growth of number of triangles and vertices, while the length of the image sequences will remain almost the same, and for the purposes of 3D television it is not expected to be larger than 10 seconds, i.e. 250 frames.

Having this in mind, it seems to be preferable to use PCA to encode vertex trajectories rather than the whole frames. The advantages of such approach are following:

- smaller size of autocorrelation matrix, which can be therefore faster analyzed using eigenvalue decomposition
- smaller size of the PCA basis vectors
- significantly lower memory consumption during compression, due to the fact that SVD within shape-based PCA depends on the square of number of vertices, while EVD within trajectory-based PCA only depends on the square of number of frames, which is much smaller.

The experiments published in [11] also show that the temporal PCA provides for complex meshes better compression efficiency than the spatial PCA, given the space dimensionality is reduced by equal ratio.

Surprisingly, the temporal PCA has been only combined with very weak spatial compression tools, such as clustering. However, the available literature gives us a large variety of spatial encoding tools proposed for static mesh compression. From these, we have chosen the parallelogram predictor based on the EdgeBreaker [12] mesh traversal algorithm to predict the values of PCA coefficients in our CoDDyaC algorithm. The choice can be possibly replaced by an even more elaborate technique, however it provides a very good improvement over the spatial clustering approach, while it is still very easy to implement.

3. ALGORITHM DESCRIPTION

The compression algorithm can be summarized into following steps:

1. construct a matrix of input data, where each column represents the trajectory of one vertex
2. apply the PCA on the matrix
3. express each vertex's trajectory in the new basis, i.e. obtain a set of coefficients for each vertex
4. traverse the common frame connectivity according to the EdgeBreaker algorithm. For each C case (new vertex), calculate the parallelogram prediction of the coefficients of the new vertex
5. quantize and encode the prediction residuals
6. encode the PCA basis without quantization

In the first step, the input data are reorganized to form a set of trajectories. Note that for the PCA (unlike Wavelet decomposition) the order of values in the vector is irrelevant, and therefore it does not matter whether the vectors contains all the X values first, followed by all Y values and all Z values, or whether the vector contains XYZ triplets. The resulting matrix has the number of columns equal to the number of vertices of the shared connectivity, and 3f rows, f being the number of frames, which we don't expect to be more than 200.

Subsequently, we construct the autocorrelation matrix of the input data. Note that, in contrast to spatial PCA, here

we're having a non-singular, relatively small (600x600) matrix. This matrix is then analyzed using off-the-shelf eigenvalue decomposition algorithm.

The resulting eigenvectors represent the new basis of the space of trajectories, while corresponding eigenvalues represent the importance factors associated with the basis vectors. Now we select the first N basis vectors, and express every point trajectory within this new reduced basis by a simple matrix multiplication.

The data structure at this point of the algorithm can be seen as a single connectivity triangular mesh, where each vertex is associated to a vector of coefficients, which can be decompressed into the vertex's trajectory. Now we traverse the topology in the EdgeBreaker fashion, i.e. we start from a triangle and expand a borderline by one of the CLERS operations. Each vertex (except from the three initial ones) is used exactly once in the C operation, i.e. addition of a new vertex. In contrast to the EdgeBreaker, we do not encode the vertex coordinates, but we use the parallelogram predictor to predict the associated vector of PCA coefficients.

The prediction residual is quantized with some user-defined step, which affects the compression accuracy. The decoded quantized value replaces the original vector value, in order to avoid quantization error accumulation. Finally, the integer residuals are encoded using arithmetic coding, which allows reaching the entropy rate of the output stream.

The decoder is very simple. First, it decodes from the stream the PCA basis, and the vectors associated with the first triangle. Then it reads the CLERS string from the encoded stream, and reconstructs the shared connectivity. Simultaneously, whenever the decoder reads the C operation code, it reconstructs the vector of PCA coefficients at the new vertex. The EdgeBreaker algorithm ensures that the adjacent triangle's vectors are always available.

Finally, the decoder reads the PCA basis, and reconstructs the original vertex trajectories by multiplying the coefficients by the basis vectors. Overall, when efficiently implemented, the decompression can be performed in real-time, given that some lag is acceptable to buffer the following scene.

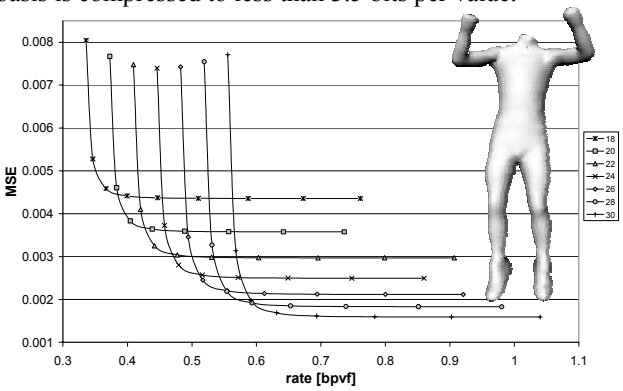
4. EXPERIMENTAL RESULTS

We have implemented and tested our algorithm, and we have compared it to the most similar approach, i.e. the LPC algorithm by Karni. For the testing purposes, we have chosen the datasets that fit the current trend of high precision. We have performed extensive testing with the human jump dataset [13][14] (15700 vertices, we have chosen a subset of 200 frames) and dance sequence (7000 vertices, 100 frames). Although there are other error measures available [15], we have performed MSE measures, in order to obtain results comparable with similar experiments.

First, we found out that our method can process significantly more complex meshes in the main system memory than the LPC based approach. While CoDDyaC is easily applicable on the whole length of the human jump sequence, we ran into out of memory problems with the LPC approach unless the sequence has been shortened to about 150 frames.

basis length	No. of encoded values	encoded values entropy	basis size	total bits [kb]	rate [bpvf]	MSE
18	284886	2.0132	10800	1235.1	0.399	0.00441
20	316540	1.9610	12000	1356.2	0.438	0.00364
22	348194	1.9157	13200	1476.4	0.477	0.00304
24	379848	1.8762	14400	1596.0	0.516	0.00257
26	411502	1.8417	15600	1715.1	0.554	0.00220
28	443156	1.8114	16800	1833.9	0.593	0.00191
30	474810	1.7844	18000	1952.4	0.631	0.00168

The graphs show the rate/distortion ratios of the CoDDyaC approach for different numbers of basis vectors. The accuracy comparison for the dance sequence is shown in the second table. We assume that the PCA basis is encoded in a lossless fashion, i.e. 64 bits per double value. Although this assumption may be considered too strong, we note that lossy compression of the basis will lead to a very hardly predictable error distribution over the whole length of the mesh sequence. It can be also seen that even reducing the precision to a 32 bit float numbers will not change the superiority of CoDDyaC, which will be preserved unless the basis is compressed to less than 3.5 bits per value.



5. CONCLUSIONS AND FUTURE WORK

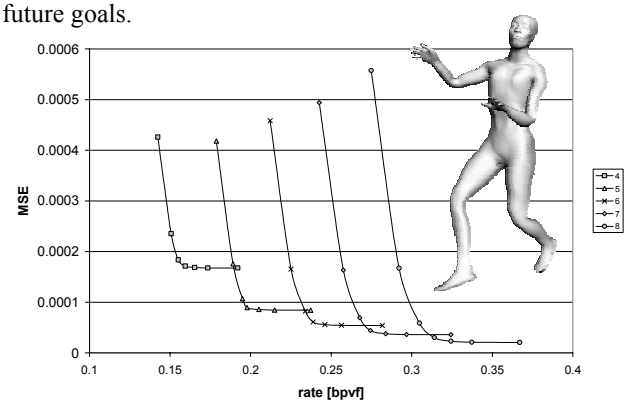
We have presented a new method for dynamic mesh compression based on EdgeBreaker and PCA. The CoDDyaC algorithm exploits both spatial and temporal coherence of the input data for most of the encoded values. The PCA basis overhead is reduced compared to the LPC coding algorithm. The memory requirements of the algorithm do not grow with the complexity of the meshes, because it only depends on the length of the mesh sequence. We report significant gains in the terms of rate/distortion ration, where distortion is measured by MSE. The cases

basis length	No. of encoded values	encoded values entropy	basis size	total size [kb]	rate [bpvf]	MSE
PCA+LPC						
20	1940	3.089	423660	26484	38.408	1.15E-06
18	1746	3.146	381294	23836	34.567	2.06E-06
16	1552	3.193	338928	21187	30.727	3.34E-06
14	1358	3.264	296562	18539	26.886	5.22E-06
12	1164	3.333	254196	15891	23.04	7.82E-06
10	970	3.388	211830	13242	19.204	1.31E-05
8	776	3.459	169464	10594	15.363	2.08E-05
6	582	3.465	127098	7945	11.522	4.07E-05
4	388	3.514	84732	5297	7.6819	8.22E-05
CoDDyaC						
10	70580	1.091	3000	262.7	0.3809	5.82E-05
9	63522	1.094	2700	236.6	0.3432	5.82E-05
8	56464	1.091	2400	210.2	0.3047	5.90E-05
7	49406	1.106	2100	184.6	0.2677	6.92E-05
6	42348	1.181	1800	161.3	0.2339	8.23E-05
5	35290	1.180	1500	134.4	0.1949	0.000107
4	28232	1.158	1200	106.9	0.1550	0.000184
3	21174	1.204	900	81.2	0.1177	0.00039

when the algorithm should not be employed are whenever a long sequence of very simple geometry occurs.

In the future we would like to combine the trajectory based PCA with more sophisticated prediction schemes. Another possibility is to use some simplification scheme on the shared connectivity on order to obtain a simplification algorithm for dynamic meshes.

We would also like to test our method using the state-of-the-art dynamic mesh comparison tools. We still believe that subjective quality measure is the best way to measure visual quality, and therefore subjective testing is one of our future goals.



ACKNOWLEDGEMENTS

This work has been supported by EU within FP6 under Grant 511568 with the acronym 3DTV and by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics).

REFERENCES

[1] Touma C., Gotsman C.: Triangle Mesh Compression. *Proceedings of Graphics Interface*, Vancouver, June 1998.

[2] Ibarria L., Rossignac J.: Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, San Diego, California, 2003.

[3] Stefanoski N., Ostermann J.: Connectivity-Guided Predictive Compression of Dynamic 3D Meshes. *International Conference on Image Processing (ICIP)*, 2006.

[4] Müller K., Smolic A., Kautzner M., Eisert P., Wiegand T.: Predictive Compression of Dynamic 3D Meshes. *Proc. ICIP2005, IEEE International Conference on Image Processing*, Genoa, Italy, 2005.

[5] Payan F., Antonini M.: Wavelet-based Compression of 3D Mesh Sequences. *Proceedings of IEEE 2nd ACIDCA-ICMI'2005, Tozeur*, Tunisia, 2005.

[6] Alexa M., Müller W.: Representing animations by principal components. *Computer Graphics Forum*, 19(3), pages 411-418, 2000.

[7] Karni Z., Gotsman C.: Efficient Compression of Soft-Body Animation Sequences. *Computers and Graphics*, 28:25-34, 2004.

[8] Sattler M., Sarlette R., Klein R.: Simple and efficient compression of animation sequences. *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (SCA 2005), pages 209-217, 2005.

[9] Amjoun R., Sondershaus R., Straßer W.: Compression of complex animated meshes. volume 4035, pages 606–613, *Computer Graphics International 2006 Conference*, 2006.

[10] Mamou K., Zaharia T., Prêteux F.: A skinning approach for dynamic 3D mesh compression, *Computer Animation and Virtual Worlds*, Volume 17, Numbers 3-4, pp. 337-346(10), 2006.

[11] Váša, L.: Methods for dynamic mesh size reduction, Technical report no. DCSE/TR-2006-07 at University of West Bohemia, 2006.

[12] Rossignac J.: Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 5, No. 1, 1999.

[13] Ankar N., Guskov I.: Extracting Animated Meshes with Adaptive Motion Estimation, *Proceedings of the 9th International Fall Workshop on Vision*, Modeling, and Visualization, November 2004.

[14] Sand P., McMillan L., Popovic J. : Continuous Capture of Skin Deformation, *ACM Transactions on Graphics*, 22(3), pp. 578-586, 2003.

[15] Váša L., Skala V.: A Spatio-Temporal Metric for Dynamic Mesh Comparison. *In proceedings of AMDO2006 Int.conf, Spain, Springer-Verlag* LNCS 4069, pages 29-37, 2006.