IMPROVEMENTS OF MPEG-4 STANDARD FAMC FOR EFFICIENT 3D ANIMATION COMPRESSION

Oldřich Petřík (opetrik@kiv.zcu.cz), Libor Váša (lvasa@kiv.zcu.cz)

University of West Bohemia, Dept. of Computer Science and Engineering
Univerzitní 22, Plzeň, Czech Republic *http://graphics.zcu.cz*

ABSTRACT

Advances in both graphics hardware and scanning technology have allowed for retrieving and rendering complex 3D models with thousands of vertices. With the recent popularity gain of interactive web applications, the data size is beginning to be the tightest bottleneck, especially for detailed 3D animations. The MPEG-4 standard was recently supplemented with an algorithm for compression of dynamic triangle meshes called Frame-based Animated Mesh Compression, which is considered the state of the art in this area. We have thoroughly analysed the algorithm and identified some weak spots and unclarities. In this paper, we present several improvements and optimisations, which attempt to resolve the problems we found. These changes result in combined performance gain of up to 32 percent on tested datasets.

Index Terms— dynamic mesh, compression, 3D animation, FAMC, MPEG-4

1. INTRODUCTION AND RELATED WORK

As the technology of 3D scanning technologies and graphics hardware evolve, they allow for retrieving and rendering complex 3D models with many thousands of vertices, often also with time variable positions. Such models with constant connectivity and dynamic geometry are usually referred to as dynamic meshes. These 3D animations are applied in many areas. They are a crucial part of 3D television, and interactive applications, such as computer games and virtual reality. With the growing popularity of web-based interactive applications, the main issue is quickly becoming to be the data size of these detailed animated models. Over the last decade, many dynamic mesh compression algorithms were developed. Recently, one of them, called Frame-based Dynamic Mesh Compression by Mamou et al. [1], was accepted as the MPEG-4 standard for dynamic triangle mesh compression [2]. This algorithm is considered the state of the art in this area.

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics).

978-1-61284-162-5/11/$26.00 © 2011 IEEE

Since its standardisation, several improvements to the algorithm were proposed. Mamou et al. suggested using Principal Component Analysis (PCA), or optimised Discrete Wavelet Transform (DWT) with bit allocation instead of Discrete Cosine Transform (DCT) to encode the transformed data [3]. Recently, Bici and Akar have introduced new predictors for the Layered Decomposition step [4]. We have focused on the clustering and transformation stages prior to the encoding step and adjusted these stages to further improve the rate-distortion (RD) performance of the algorithm.

2. FAMC OVERVIEW

Since the rest of the paper deals with improvements of and additions to the FAMC algorithm, we will now describe it very briefly. To fully understand the details of the algorithm, and this paper, please refer to the MPEG-4 standard [2] and Mamou et al. [1].

The FAMC algorithm is based on an improved version of the skinning approach by Mamou et al. [5]. The main idea is to partition the vertices of the animated mesh into groups, whose motion in each frame can be predicted well with a single affine transform. This part of the a algorithm is called *Motion-based Segmentation*. The segmentation is performed by a series of half-edge collapse steps. At the beginning, each vertex of the dynamic mesh is considered a cluster. All edges of the mesh are given costs of their collapses. Each cost is calculated as the error caused by virtually merging the clusters of the two edge endpoints, and approximating the motion of this new virtual cluster c relative to the first frame (key frame) by a single affine transform A_f^c per each frame f using least squares minimisation (LSM) over all vertices of c. In every collapse step, the edge with the lowest cost is contracted to a single vertex and a new cluster is formed by merging the clusters of both endpoints of the edge. The costs of all affected edges are then updated. Once the lowest collapse cost exceeds a given threshold, the segmentation process stops.

Once the partition is complete, the motion of each cluster c between the first frame and frame f is approximated by a single affine transform per each frame. Each vertex v is then assigned a vector of weights w^v, which describes how the motion of v is influenced by the transforms of its parent cluster c

(w_0^v) and of all neighbouring clusters of c ($w_1^v \dots w_{K^c}^v$, where K^c is the number of neighbours of c). Vector w^v is obtained using LSM over all the frames of the animation.

The weighted transforms are then applied to each vertex v for each frame f and a correction vector (prediction residue) $e_f^v = v_f - \tilde{v}_f$ between the original position v^f and the transformed position $\tilde{v}_f$ is calculated. Finally, the entire animation is stored as the group of the key frame mesh (including connectivity) encoded by MPEG-4 3DMC [6], a partition information, a transform matrix for each cluster and frame, a weight vector for each vertex and a residue for each vertex and frame. Transform matrices and residues may be predicted through Layered Decomposition [7], and/or encoded using DCT or DWT. All the values are then uniformly quantised with different quantisation constant for transform matrix coefficients, cluster weights, and prediction residues. The quantised values are encoded using CABAC.

3. IMPROVEMENTS

3.1. Handling More Connected Components

When reimplementing the FAMC algorithm, we have noticed, that the original papers do not mention the behaviour of the algorithm in the case of animations consisting of more connected components. Since some sections of different components can express similar movement (e.g. a head and its eye balls), it would be advantageous to either cluster these sections together, or include them to the neighbour clusters of one another. However, since there is no edge connecting the components, the clustering process cannot result in any of these situations. According to the authors, in order to handle such cases, a preprocessing step is performed introducing a virtual edge per each component pair, which connects a random vertex from one component with a random vertex from the other component. However, this approach does not produce meaningful results in most situations and, in extreme cases, leads to scattering parts of the clusters across the mesh.

To correctly handle multicomponent meshes, we add a virtual edge for each pair of components connecting the closest pair of vertices, one from each of the components. Because the distances between the vertices may change over the span of the animation, we approximate the closest pair by finding the pair with lowest distance of average positions. This approach is much faster than finding the pair with the lowest average distance, yet still accurate enough.

3.2. Overcoming Random Edge Collapses

As mentioned above, the clustering process is based on a succession of half-edge collapses in an order determined by collapse costs. To calculate an edge-collapse cost, we need to obtain the 4×3 affine transform for the virtual cluster of the edge. To uniquely determine this transform, the cluster has to contain at least four vertices. However, at the beginning of the segmentation process, there is one vertex per cluster only, i.e. two vertices per edge. The original algorithm considers the error for less than five vertices to be 0, which causes the edges to collapse in a random manner, until all the remaining edges have at least five linearly independent vertices. Hence, this random phase of the segmentation may create clusters containing vertices of completely unrelated movement, which are then difficult to merge with other clusters.

We propose to include the neighbour vertices of the edge endpoints to the virtual cluster until the endpoint clusters contain enough vertices. Thus, the edges in areas of very affine-like movement are collapsed first and the borders of the final clusters much closely reflect the borders of areas of similar motion. Since the decoder only needs the information about the final clustering, this technique and the one above do not induce any changes to the decoding scheme, and are therefore *fully compliant with the original MPEG-4 specification*.

3.3. Weight Renormalisation

The quantisation step damages the values by adding quantisation noise (QN) of magnitude in the interval $[-\frac{1}{2}q, \frac{1}{2}q)$, q being the quantisation step. We have noticed that the quantisation of cluster weights, especially at low bitrates (under 1 bpfv), creates visible artefacts around cluster borders. These artefacts are caused by the change in the sum of weights for each vertex due to QN. The original sum of weights is approximately 1, as each cluster transform is calculated to be used for the cluster directly, i.e. with unit weight. However, the sum of weights damaged by QN diverges from this value causing distortions in the vertex positions. We minimise these distortions by renormalising the decoded weights $\hat{w}_1^v \dots \hat{w}_{K^c}^v$ of vertex v in cluster c to sum up to 1 (equation 1). This is similar to deblocking techniques used in video decoding.

$$norm(\hat{w}_i^v) = \frac{\hat{w}_i^v}{\sum_{k=0}^{W^v} \hat{w}_k^v} \tag{1}$$

Performing the renormalisation alone introduces a small distortion to the positions, as the sum of weights is not exactly 1 for most vertices. To minimise this distortion, we have altered the computation of the weights. Assuming the sum of the weights equals to 1, we can express weight w_i^v using the rest of the weights (equation 2). Then, we substitute this formula for w_i^v in the original weight calculation system and perform the least squares optimisation to obtain the weights except w_i^v. Finally, w_i^v is calculated from equation 2. We use this substitution for the parent cluster weight, w_0^v, which makes the calculation independent on the number of neighbour clusters. Experiments show that it also produces slightly better results than substituting other weights.

$$w_i^v = 1 - \left(\sum_{k=0}^{i-1} w_k^v\right) - \left(\sum_{k=i+1}^{W^v} w_k^v\right) \tag{2}$$

	original FAMC								adjusted FAMC							
	dance		**chicken**		**cow**		**jump**		**dance**		**chicken**		**cow**		**jump**	
	R	D	R	D	R	D	R	D	R	D	R	D	R	D	R	D
KG	0.25	2.50	0.99	4.33	0.49	3.14	0.26	2.49	0.25	1.31	0.86	4.30	0.54	2.39	0.25	2.32
	0.57	0.57	2.52	1.01	1.18	1.19	0.59	1.27	0.52	0.43	2.28	0.99	1.11	1.19	0.60	1.18
	0.98	0.22	3.36	0.56	1.46	0.96	1.01	0.79	0.90	0.18	3.24	0.52	1.39	0.96	0.96	0.80
	1.30	0.15	4.46	0.32	2.11	0.69	1.46	0.53	1.29	0.12	4.32	0.27	2.01	0.68	1.40	0.54
	2.00	0.08	5.85	0.19	3.13	0.46	2.11	0.34	1.86	0.07	5.75	0.15	3.24	0.40	2.11	0.33
	2.61	0.05	7.38	0.12	5.11	0.26	3.17	0.20	2.42	0.04	7.40	0.07	5.01	0.23	3.15	0.20
STED ($\times$1000)	0.23	0.79	1.64	1.70	0.68	1.18	0.20	1.20	0.22	0.44	1.50	1.38	0.66	1.13	0.14	1.03
	0.43	0.28	2.30	1.25	1.14	0.97	0.23	0.78	0.39	0.23	2.06	1.04	1.06	0.95	0.17	0.78
	0.92	0.17	3.43	0.81	1.59	0.86	0.29	0.61	0.67	0.14	2.73	0.78	1.46	0.84	0.25	0.61
	1.50	0.12	4.43	0.53	2.20	0.76	0.58	0.49	1.35	0.10	4.21	0.37	2.04	0.70	0.42	0.49
	3.38	0.06	6.39	0.26	3.13	0.64	1.04	0.41	3.01	0.05	6.05	0.19	2.96	0.57	0.87	0.41
	4.57	0.03	7.40	0.19	7.25	0.36	4.62	0.22	4.13	0.03	7.40	0.10	6.85	0.35	4.48	0.22

Table 1. Compression performance of the original and our adjusted FAMC algorithm using KG and STED distortion measures. The STED distortion values have been multiplied by 1000 to fit better into the table.

KG				STED			
dance	**chicken**	**cow**	**jump**	**dance**	**chicken**	**cow**	**jump**
27.37 %	22.13 %	9.06 %	2.52 %	19.36 %	32.52 %	9.98 %	2.49 %

Table 2. Average distortion improvements of the adjusted algorithm over the original FAMC algorithm.

3.4. Weight Encoding

The aim of the segmentation process is to optimally cluster the mesh, so that each cluster can be described using one affine transform only. This means, that for nearly all the vertices, the weight of the motion of their parent cluster, or w_0, is the highest weight. Moreover, by avoiding the random edge collapses at the beginning of the segmentation process, all vertices should belong to the cluster that influences their movement the most. And since the sum of the weights is near one, w_0 also approaches 1.

To reflect this behaviour, instead of storing w_0 directly, we store $1 - w_0$, which results in smaller absolute values. These values are subsequently encoded using CABAC with Exp-Golomb binariser [8], thus the closer the values are to zero, the less space is required to store the information. Another advantage of this approach is that it keeps the sum of weights from dropping to 0. Even if the quantisation is so coarse that it would cause all the weights to round to 0, weight w_0 gets decoded to 1, as $1 - w_0$ was quantised to 0.

3.5. Affine Transform Encoding

According to the original design of the FAMC algorithm, the affine transform of a cluster for each frame f is encoded by transforming a selected set of four non-coplanar points, $M_1^1 \dots M_4^1$, by this transform and storing their transformed positions, $M_1^f \dots M_4^f$. For the input points, we are using the origin and the axis vectors of the base coordinate system: $M_1^1 = (0,0,0), M_2^1 = (1,0,0), M_3^1 = (0,1,0), M_4^1 = (0,0,1)$. With this in mind, we can predict the transformed position of the fourth point using cross product (equation 3) and store only the prediction residue $M_4^f - \tilde{M}_4^f$. The prediction is perfect for any rigid transform, i.e. any combination of rotation, translation, and uniform scaling.

$$\tilde{M}_4^f = M_1^f + (M_2^f - M_1^f) \times (M_3^f - M_1^f) \qquad (3)$$

4. EXPERIMENTAL RESULTS

We have implemented all the adjustments described above into the FAMC encoder and evaluated its RD performance. In table 1, you can see the results obtained with four different dynamic meshes on several configurations with and without the adjustments using the commonly used Karni-Gotsman (KG) distortion metric [9] and the new perception-based STED metric by Váša and Skala [10]. The results are presented in the form of bitrate R in bits per frame and vertex and distortion D calculated using either the KG or the STED method at optimal parameter configurations. Table 2 shows the average improvements in KG and STED distortions on the four meshes. These values were calculated by integrating the vertical relative difference between RD curves over the measured bitrate interval. RD curves for the *chicken* and *jump* dynamic meshes are shown in figure 1. The results show that our adjustments work best for *chicken* and *dance* mesh sequences, while for the *jump* sequence, they have almost no contribution whatsoever. This behaviour depends on the vertex density of the mesh, which is much higher in the *jump* sequence. The more vertices each cluster contains, the less error is in-

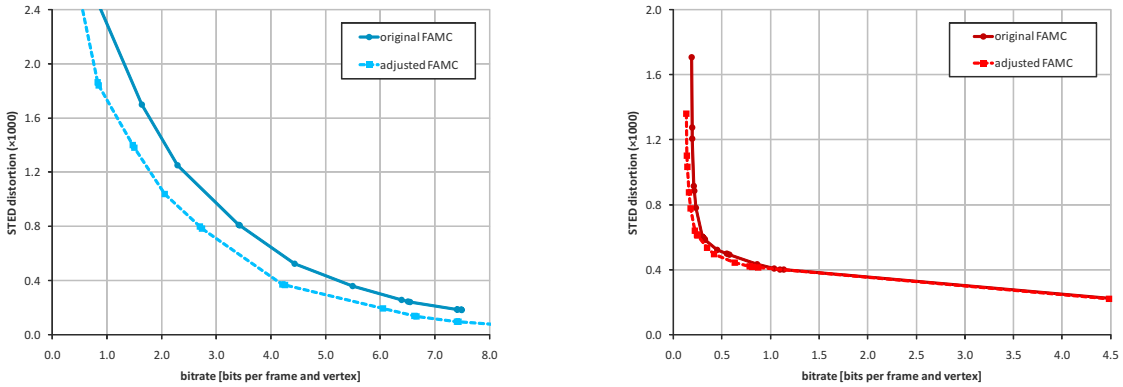

Fig. 1. RD curves of the original and the adjusted FAMC in STED metric for the chicken (left) and jump (right) animation.

troduced, when the cluster border deviates from its optimal shape due to random edge collapses, and the more space is required for the residual values, compared to the cluster transforms, which do not change. Note also, that the *jump*, *cow*, and *dance* animations only consist of one connected component, thus the improved component connecting has no effect in their cases. Computation time of the improved compressor is only 0–6 % longer, while the improved clustering takes 60–170 % more time for single-component meshes and up to 300 % more time for multiple-component meshes. This time could be, however, significantly reduced by using an appropriate data structure. The extent limitation for the paper does not allow us to include more test cases, so we have included the ones producing both the best and the worst results.

5. CONCLUSION

We present five individual adjustments to the MPEG-4 FAMC algorithm for dynamic triangle mesh compression, which aim at improving the RD performance of the algorithm by refining some of its original strategies. We propose a new way of creating virtual edges between connected components and a mechanism for avoiding random edge collapses to improve the result of the segmentation stage. These adjustments maintain compatibility with the original standardised FAMC decoder. A new encoding of animation weights reduces the space required to store the weights and a renormalisation step attempts to recover the damage caused by quantisation. Finally, by adding a prediction step to the encoding of cluster transforms, we were able to further improve the RD performance. Depending on the properties of the input data, these adjustments combined decrease the average distortion by up to 27 % in Karni-Gotsman metric and 32 % in STED metric on our test meshes.

6. REFERENCES

[1] K. Mamou, T. Zaharia, F. Prêteux, N. Stefanoski, and J. Ostermann, "Frame-based compression of animated meshes in MPEG-4," in *2008 IEEE International Conference on Multimedia and Expo*, 2008, pp. 1121–1124.

[2] ISO, *ISO/IEC 14496-16, amd. 2: Frame-based Animated Mesh Compression (FAMC)*, 2009.

[3] K. Mamou, T. Zaharia, F. Prêteux, A. Kamoun, F. Payan, and M. Antonini, "Two optimizations of the MPEG-4 FAMC standard for enhanced compression of animated 3D meshes," in *15th IEEE International Conference on Image Processing*, 2008, pp. 1764–1767.

[4] M. O. Bici and G. B. Akar, "Improved prediction for layered predictive animated mesh compression," in *17th IEEE International Conference on Image Processing*, 2010, pp. 3413–3416.

[5] K. Mamou, T. Zaharia, and F. Prêteux, "A skinning approach for dynamic 3D mesh compression," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 337–346, 2006.

[6] ISO, *ISO/IEC 14496-2: Visual*, 2001.

[7] N. Stefanoski, L. Xiaoliang, P. Klie, and J. Ostermann, "Scalable linear predictive coding of time-consistent 3D mesh sequences," in *3DTV Conference*, 2007, pp. 1–4.

[8] H. Kirchhoffer, D. Marpe, K. Müller, and T. Wiegand, "Context-adaptive binary arithmetic coding for frame-based animated mesh compression," in *2008 IEEE International Conference on Multimedia and Expo*, 2008, pp. 341–344.

[9] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computers and Graphics*, vol. 28, pp. 25–34, 2004.

[10] L. Váša and V. Skala, "A perception correlated comparison method for dynamic meshes," *IEEE Trns. on Visualization and Computer Graphics*, vol. 17, no. 2, pp. 220–230, 2011.