COMPUTER GRAPHICS forum

DOI: 10.1111/j.1467-8659.2008.01304.x

COMPUTER GRAPHICS forum
Volume 28 (2009), number 6 pp. 1529–1540

COBRA: Compression of the Basis for PCA Represented Animations

L. Váša and V. Skala

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

Abstract

In this paper, we present an extension of dynamic mesh compression techniques based on PCA. Such representation allows very compact representation of moving 3D surfaces; however, it requires some side information to be transmitted along with the main data. The biggest part of this information is the PCA basis, and since the data can be encoded very efficiently, the size of the basis cannot be neglected when considering the overall performance of a compression algorithm.

We present a new work in this area, as none of the papers about PCA based compression really addresses this issue. We will show that for an efficient and accurate encoding there are better choices than even sophisticated algorithms such as LPC.

We will present results showing that our approach can reduce the size of the basis by 90% with respect to direct encoding, which can lead to approximately 25% increase of performance of the compression algorithm without any significant loss of accuracy. Such improvement moves the performance of the PCA encoder beyond the performance of current state of the art dynamic mesh compression algorithms, such as the recently adopted MPEG standard, FAMC.

Keywords: Mesh animation, PCA, compression, basis, quantisation

ACM CCS: I.3.7 [Computer Graphics]: Animation

1. Introduction

Compression of dynamic meshes is a topic which has gained increased attention during the last period. The problem can be seen as equivalent to static mesh compression; however, with dynamicity there appears a whole new group of problems to be solved and also properties to be exploited.

Generally, the task of dynamic mesh compression is the following: The input is a series of static triangular meshes $M_1 \ldots M_n$, where each mesh has the same connectivity, yet different geometry. The set of meshes represents a temporal development of some surface; thus, there are no sudden changes in the geometry. The task is to find an approximate representation of the input data which will take smallest possible number of bits, and which will introduce smallest possible error to the data.

The problem has been addressed by many research papers in the past, and even currently there is quite intensive research in the field. One of the most efficient approaches is the use of principal component analysis to represent the animation by a PCA basis and some coefficient vectors, which either describe the desired combination of eigenshapes or eigentrajectories. Majority of the approaches focuses on the efficient encoding of the coefficient vectors, whereas the PCA basis is encoded directly, because the effect of lossy encoding of the basis is not easy to predict.

However, in our experiments we have found out that when the coefficients are encoded efficiently, then the basis can take up to 50% of the encoded data size. To comprehend the importance of basis see Figure 12, where the relation of basis and coefficients is shown for eight different models at KG error (a derivative of mean squared error defined in [KG04]) approximately 0.3. We are also showing the size of triangle connectivity, which takes only a negligible part of the encoded size, because it is shared by all the frames.

The only current approach for basis compression, a simple quantisation, provides some improvement, but still it does not use all the properties of the basis. Therefore, we are proposing a compression scheme for the PCA basis, which is based on non-uniform quantisation and non-least-squares optimal linear prediction, which addresses this issue.

The idea of the non-uniform quantisation is well known, and it has been used in algorithms such as JPEG compression, etc.; however, it is the insight into the character of the compressed data that allows using appropriate quantisation constants to achieve efficient compression. Our first contribution is showing how to steer the quantisation so that all the required values are encoded neither too accurately nor too coarsely.

The idea of predictive encoding is not novel either; however, the choice of the best predictor is not trivial, and it 's discussion is our second contribution.

The rest of the paper is organised as follows: Section 2 will briefly describe existing approaches to dynamic mesh compression with special detail about PCA based approaches. Section 3 will give a brief description of the trajectory-space PCA based algorithms, which form the background of our scheme. Section 4 will sketch some possible approaches and derive the choices we have made to construct our algorithm. Section 5 will give detail about the non-uniform quantisation we are using to increase efficiency. Finally, Section 6 will present results of our approach applied to one algorithm based on temporal PCA and Section 7 will draw conclusions and directions for future work.

2. Related Work

First attempt to dynamic mesh compression has been published in the paper by Lengyel [Len99], who suggested subdividing the mesh into clusters in which the movement can be described by a single transformation matrix.

Ibarria and Rossignac [IR03] later suggested a spatio-temporal prediction schemes ELP and Replica, which were used to predict next vertex position during a mesh traversal using the EdgeBreaker state machine. A similar approach has been used by Stefanoski in his angle preserving predictor [SO06]. The position of the new vertex is expressed in local coordinate system defined by a neighbouring triangle.

A predictor based approach has been improved by Mueller *et al.* [MSK*06, MSK*05] by including spatial subdivision using an octree data structure. For each cell, an appropriate predictor is selected, which best suits the given cell, or the cell is further divided when the behaviour is predicted badly.

Wavelet theory has been used for dynamic mesh compression in the work of Payan [PA05], who suggested treating separate vertex trajectories as sampled signal. However, their method did not use the spatial coherence present in the data.

A different class of approaches has been pioneered by Alexa and Mueller [AM00], who suggested using PCA in the space of frames, expressing each frame as a linear combination of eigenshapes. However, this method had problems with rigid movement, which had to be compensated in preprocessing step, where a transformation matrix for each frame has been found using the least-squares approach.

The method has been subsequently improved by Karni and Gotsman [KG04], who suggested exploiting the temporal coherence of the PCA coefficients by encoding them using linear prediction coding (LPC), thus achieving lower entropy of the encoded data. Another improvement has been proposed by Sattler *et al.* [SSK05], who suggested using PCA in the space of trajectories, and finding clusters of vertices where PCA worked well (Clustered PCA). However, their iterative clustering method did not always reach the same clustering because it has been randomly initialised.

The compression of the basis of PCA represented animations has been discussed by Heu *et al.* [HYKL06]; however, their result has been only applied on the case of eigenshapes based PCA representation. Their suggestion is to derive the quantisation constants from the eigenvalues of the autocorrelation matrix. We will show that this approach is not correct.

Another addition to the PCA based method has been proposed in 2007 by Amjoun [Amj07], who suggested using trajectory based analysis along with expressing each trajectory in a local coordinate frame defined for each cluster. Additionally, a bit allocation procedure is applied, assigning more bits to cluster where more PCA coefficients are needed to achieve desired precision. This paper also mentions basis compression; however, it suggests simple direct encoding without prediction and with uniform quantisation of the basis matrices.

Finally, Váša and Skala [VS07] have presented a trajectorybased PCA coding, combined with EdgeBreaker-like [Ros99] predictor. Their Coddyac algorithm predicts the PCA coefficients by the well known parallelogram local predictor, which allows better performance than the clustering based approaches.

Mamou [MZP06] has proposed an approach similar to PCA, called skinning based compression. The mesh is first segmented to parts that move in an almost rigid fashion. Each cluster 's movement is expressed by a transformation matrix, and subsequently, each vertex is assigned a vector of weights that tells how to combine the transforms of the neighbouring clusters to obtain the movement of the vertex.

A resampling approach has been proposed by Briceno [BSM*03] in his work on Geometry Videos. The idea is an extension of the previously proposed Geometry Images [GGH02]. The geometry of the object is unwrapped and projected onto a square, which is regularly sampled. The resulting image is encoded using some off the shelf algorithm. The

extension to videos solves the problems of finding a single mapping of a moving content onto a square while minimising the overall tension. Generally, the method is not easy to implement and suffers from some artefacts, especially for objects with complex geometry.

There are also scalable approaches appearing, such as the scheme proposed by Stefanoski *et al.* [SLKO07]. These approaches allow progressive transmission of level of detail of the dynamic mesh, and also achieve better compression ratios by using sophisticated local predictors which use the data from coarser detail levels.

Finally in 2007, a new standard for dynamic mesh compression has been adopted by MPEG, called Frame-based Animated Mesh Compression (FAMC). This approach is described by MPEG-4 standard, Amendment 2 of part 16 Animation Framework eXtension (AFX). It is a combination of approaches of [MZP06] and [SLKO07], which provides high performance and scalability.

3. Algorithm Background

We will demonstrate the effect of basis compression on a representative of the class of PCA based compression algorithms, the Coddyac scheme. We will briefly describe it to make the basis compression easy to understand.

The input of the algorithm is a series of F frames of unchanged connectivity, which has T triangles and V vertices. The algorithm first stores all the data in a matrix M of size $3F \times V$, where each column M_i represents a trajectory of a single vertex over the duration of the animation:

$$M = [M_1, M_2, \ldots, M_V] \tag{1}$$

$$M_i = \left[X_1^i, \ldots, X_F^i, Y_1^i, \ldots, Y_F^i, Z_1^i, \ldots, Z_F^i\right]^T \tag{2}$$

Subsequently, PCA is performed. As a first step, a mean value for every row is computed and subtracted from the corresponding row so that the data is centred around the origin. The vector of the means $\mathbf{m}$ is encoded into the output stream.

An autocorrelation matrix $M \cdot M^T$ is constructed and its eigenvectors are found. These eigenvectors are then used as a new basis of the column space of the matrix M. The key property of the new basis is that the components of the original data are uncorrelated when expressed in this basis, and moreover, most of them are close to zero.

Each column (trajectory $\mathbf{t}$) is then expressed in the new basis, whereas only a user-specified number N_B of most important basis vectors is used. At this point, each vertex of the shared connectivity has assigned a vector of PCA coefficients. The connectivity is traversed in the EdgeBreaker fashion, and the coefficient vectors are predicted using the parallelogram rule. The residuals are finally quantized and encoded into the output stream using some kind of entropy coding (we have used the Huffman coding [Huf52] for the experiments; however, we will show the actual entropies in the results).

The decompression is then quite simple. The decoder extracts the connectivity and the basis in a form of $3F \times N_B$ matrix $\mathbf{B}$. Subsequently, it traverses the connectivity in the same order as the encoder, performs prediction and correction of coefficient vectors and finally reaches the state when each vertex has assigned a vector $\mathbf{c}$ of N_B components.

The final trajectory of each vertex is then computed by a simple matrix multiplication:

$$\mathbf{t} = \mathbf{B} \cdot \mathbf{c} + \mathbf{m}. \tag{3}$$

Selecting an optimal pair of compressor parameters [N_B, Q], where Q is the quantisation constant used for encoding the prediction residuals, is not an easy task. Increasing either of the parameters leads to an increase of the rate and a decrease of the error (under the standard KG-error metric); however, the actual relation of the parameters depends on the character of each data set. The only way to ensure the optimal selection is using some iterative optimisation process.

4. Prediction Schemes

The key observation for the following derivations is that the basis vectors, which need to be encoded, still retain the character of trajectories. In other words, if we interpret each basis vector as a trajectory of a moving point, then the point moves smoothly. Figure 1 shows the first basis vector of the chicken sequence interpreted as three trajectories, one for each coordinate.

This observation has been made previously by Karni and Gotsman in [KG04], who noted this behaviour of PCA coefficients of subsequent frames (note that they have used an eigenshape based PCA). Their suggestion was to apply linear predictive coding, LPC, to predict and encode the values. The LPC concept is based on predicting a given value in a series as a linear combination of a given number of previous values. The same set of combination coefficients is used for the whole sequence (or for multiple sequences of the same behaviour), and their values are found in a least-squares optimisation process, applied by the encoder on the whole sequence. For more details see the original source.

We have first followed this suggestion for the basis as well; however, we have found that for the purposes of efficient encoding it is suboptimal. Imagine a situation depicted in Figure 2. In this simplified scenario, we are given a value x, and we want to predict the value y by a linear formula:

$$v_f = k v_{f-1} + q. \tag{4}$$

The LPC algorithm suggests to apply least-squares optimisation to find the values k and q; in our case, we will get

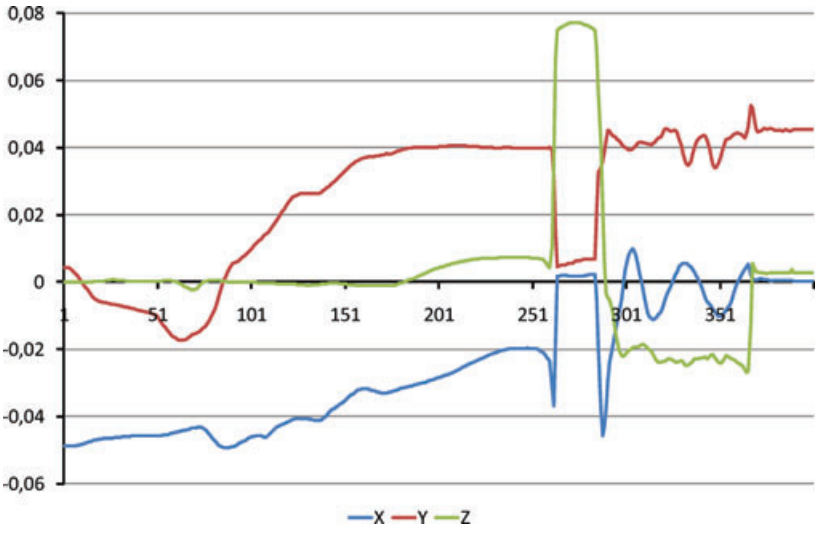

Figure 1: *First basis vector of the chicken sequence. The sudden jump in frames 250–300 is the chicken popping eyes, the subsequent sinusoidal development of the X axis is the flapping of wings.*

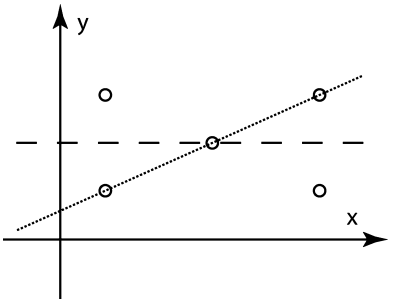

Figure 2: *Two linear predictors for a given data set (each data point is represented by a circle). The prediction task is to estimate the y component from the x component.*

the dashed line. We can now perform quantisation, which will result only in three possible values: 0 for the exactly predicted point; $+\alpha$ for the points above the prediction and $-\alpha$ for points below it. The vector of residuals is:

$$\mathbf{r} = [-\alpha, +\alpha, 0, -\alpha, +\alpha]. \tag{5}$$

Thus, the probabilities are:

$$p(-\alpha) = \frac{2}{5}, \qquad p(+\alpha) = \frac{2}{5}, \qquad p(0) = \frac{1}{5}. \tag{6}$$

The entropy is computed as follows:

$$\begin{aligned} H &= -\sum p \log_2(p) \\ &= -\left(\frac{2}{5}\log_2\left(\frac{2}{5}\right) + \frac{2}{5}\log_2\left(\frac{2}{5}\right) + \frac{1}{5}\log_2\left(\frac{1}{5}\right)\right) \\ &= 1.522[b]. \end{aligned} \tag{7}$$

However, if we construct a different linear predictor, such as the one drawn in the figure as a dotted line, we get following residuals, probabilities and entropy:

$$\mathbf{r} = [0, +\alpha, 0, -\alpha, 0], \tag{8}$$

$$p(-\alpha) = \frac{1}{5}, \qquad p(+\alpha) = \frac{1}{5}, \qquad p(0) = \frac{3}{5}, \tag{9}$$

$$\begin{aligned} H &= -\sum p \log_2(p) \\ &= -\left(\frac{3}{5}\log_2\left(\frac{3}{5}\right) + \frac{1}{5}\log_2\left(\frac{1}{5}\right) + \frac{1}{5}\log_2\left(\frac{1}{5}\right)\right) \\ &= 1.379[b]. \end{aligned} \tag{10}$$

This rather artificial example shows that there are cases, when least-squares solution leads to suboptimal prediction. Indeed, it is a generally known problem [Wei02] of the least-squares optimisation that the solution can be led astray by outliers because the squared difference of these has a large influence on the overall solution. Figure 3 shows a more realistic case, when most of the data points are linearly dependent and can be accurately predicted by the dotted line; however, the least-squares solution will be twisted by the outlier point, which will cause a significant increase of residual entropy.

Figure 4 shows a real world example. The data set is the PCA basis of the first 100 frames of the chicken sequence. The samples show the dependence of a basis value (v_f axis) on its predecessor (v_{f-1} value), where an appropriate predecessor is available. A least-squares fitting of such data has produced the depicted line, which can be used to predict v_f from v_{f-1}. However, if we consider the real situation, we

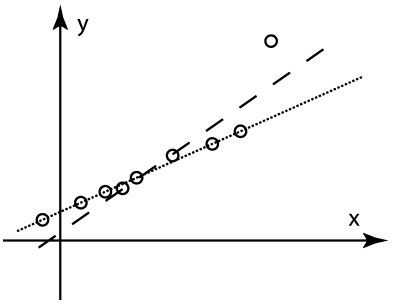

Figure 3: *Least-squares solution is led away from a good predictor of most of the data by an outlier.*

can use simple delta coding expressed as:

$$pred_1(v_f) = v_{f-1}. \tag{11}$$

Such coding is least-squares suboptimal; however, it delivers residuals with smaller entropy. Note that this least-squares suboptimality does not affect the error introduced, because we are always transmitting the quantized prediction residuals, and therefore, the error only depends on the quantisation constant.

In a similar way, we can fit the triplets (v_{f-2}, v_{f-1}, v_f), where v_{f-2} is a value from frame $f-2$, v_{f-1} is the value from frame $f-1$ and v_f is a value in frame f, which we are trying to predict. We can either use least-squares fitting, or linear movement prediction in a form:

$$pred_2(v_f) = v_{f-1} + (v_{f-1} - v_{f-2}) = 2v_{f-1} - v_{f-2}. \tag{12}$$

The last two predictors that we have experimented with use three and four preceding values to predict the current value v_f, estimating the speed s, the acceleration a, and the change in acceleration c to obtain prediction as follows:

$$\begin{aligned} s &= v_{f-1} - v_{f-2}, \\ a &= (v_{f-1} - v_{f-2}) - (v_{f-2} - v_{f-3}) \\ &= v_{f-3} - 2v_{f-2} + v_{f-1}, \\ c &= ((v_{f-1} - v_{f-2}) - (v_{f-2} - v_{f-3})) \\ &\quad - ((v_{f-2} - v_{f-3}) - (v_{f-3} - v_{f-4})); \end{aligned} \tag{13}$$

$$pred_3(v_f) = v_{f-1} + s + a; \tag{14}$$

$$pred_4(v_f) = v_{f-1} + s + a + c; \tag{15}$$

The prediction algorithm must also prevent error accumulation by using quantized values in the encoder. The overall scheme is summarised in algorithm 1.

5. Quantisation

The final step in encoding the values is quantisation. The predictor produces a floating point value, which is divided by a quantisation constant, the result is truncated and passed to an entropy coder for encoding.

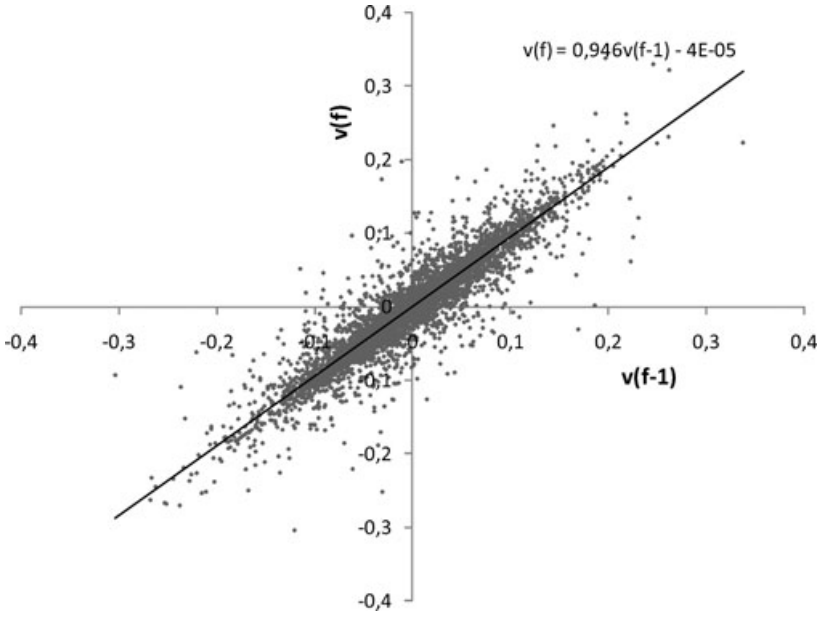

Figure 4: *Real data prediction, each point represents a data value (vertical axis) and its predecessor (horizontal axis).*

Algorithm 1: Basis compressions.

input: basis vector $B_i, i = 1..3F$, where F is the number of frames. The vector has been obtained by PCA of the original trajectories, where each trajectory has been represented by a vector of form described by (2).
input: quantisation constant Q_i (its computation will be described in the following section)
input: order of prediction o i.e. the number of preceding values needed by the predictor

```
for coord ← 0 to 2 do
    for j ← (1 + coord ∗ F) to (o + coord ∗ F) so
        q ← round(B_{i,j} / Q_i) ;
        send q to entropy coder;
        B_{i,j} ← q ∗Q_i;
    end
    for j ← (o + 1 + coord ∗F) to (F + coord ∗F) do
        pred ← predictor(B_{i,j−1}, B_{i,j−2}, . . . , B_{i,j−o});
        residual ← (B_{i,j} − pred);
        q ← round(residual/ Q_i);
        send q to entropy coder;
        B_{i,j} ← pred + q ∗Q_i;
    end
end
```

However, we have found out that careful treatment of basis quantisation may lead to further improvement of compression ratio. Recall the decompression Eq. (3). It can be expanded to following form:

$$\mathbf{t} = B_1.c_1 + B_2.c_2 + \ldots + B_{N_B}.c_{N_B} + \mathbf{m}, \qquad (16)$$

where B_i represents the i-th column of the matrix B i.e. a basis vector. The error introduced by the quantisation of the PCA coefficients c_i is equal for each term of Eq. (16), as all the coefficients are quantised with equal quantisation constant.

We can also see that the error is generally additive, and therefore, we want it to be equal for every term. However, the size of the coefficients c_i varies very significantly. Fortunately this variance can be predicted well – the first coefficient is usually much bigger than the second, which is bigger than the third, etc., which is a behaviour caused by the nature of PCA.

Thus, if we had used an uniform quantisation, we can expect the error of half the quantisation constant, which will be multiplied by a very large constant in the first term. Such behaviour is undesirable, and thus, we must use finer quantisation for the more important basis vectors, whereas the less important ones can be quantized more coarsely.

Given a quantisation constant Q from user, we can split this constant to the given number of terms of Eq. (16); however, we should follow the magnitudes of corresponding coefficients c_i. To do this, we can first compute the sum

$$s_i = \sum_{v=1..V} \|c_i^v\| \qquad (17)$$

of the absolute values of each coefficient over all vertices $v_{1..V}$. This gives an approximation of how many times an error will be repeated in the final decompressed sequence. Figure 5 shows the summed absolute values of the PCA coefficients in the chicken sequence; note that the scale of the figure is logarithmic.

If we want to keep the error of the terms in Eq. (16) equal, then we must use quantisation constants inversely

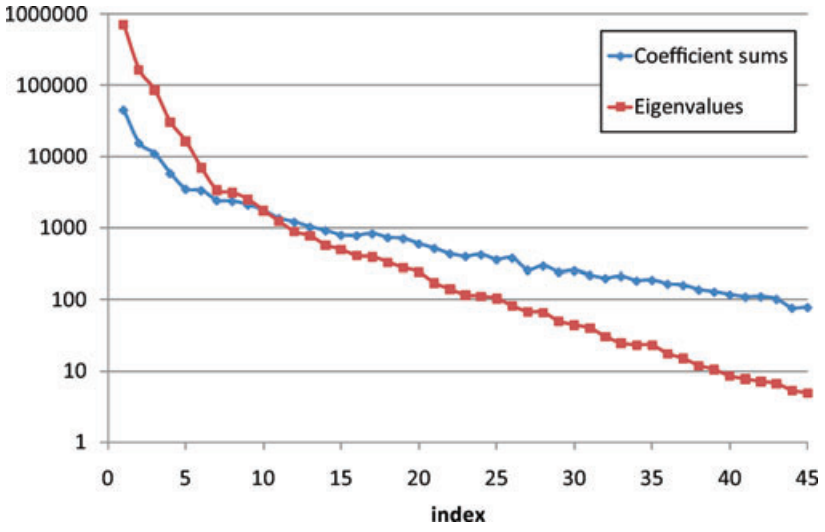

Figure 5: *Coefficient sums (absolute values) and corresponding eigenvalues. Note that the scale is logarithmic and the actual discrepancy is even more significant.*

proportional to the sums of coefficients. Thus, the quantisation constant for i-th basis vector should be computed as:

$$S = \sum_{i=1..N_B} s_i, \tag{18}$$

$$Q_i = \frac{Q \cdot S}{(N_B + 1) \cdot s_i}. \tag{19}$$

In this equation, i is the order of the basis vector and Q is the quantisation constant, which must be divided into $N_B + 1$ parts, since the Eq. (16) has $N_B + 1$ terms.

Note that this approach is fundamentally different from the one proposed by Heu *et al.* [HYKL06], where it has been suggested to use the inverse of the corresponding eigenvalue. Although the eigenvalue corresponds to the significance of each eigenvector (eigenshape or eigentrajectory), its actual value is equal to the sum of squared coefficients, which has no relevance to the actual inflicted error.

We could use an exact value for each coefficient; however, that would require to transmit the quantisation constant with each basis vector. To avoid this, we have decided to use a power-function approximation to compute the quantisation constants for each basis vector at both encoder and decoder. We perform a least-squares optimisation to obtain the constants k and q in the Eq. (20).

$$k i^{\,q} = Q_i. \tag{20}$$

Since the quantisation of the first basis vector is most important, we subsequently shift the approximation curve by a constant a so that the first quantisation constant perfectly fits the user specified intended error:

$$k.1^q + a = k + a = Q_1, \tag{21}$$

$$a = Q_1 - k. \tag{22}$$

This way, we only need to transmit the constants k, q and a, and we get a series of increasing quantisation constants that well fits the distribution of absolute values of PCA coefficients, and for the first coefficient, gives exactly the user specified error amount.

This derivation can be directly applied to compression of the means vector, which is transmitted along with the basis. Its components can be predicted using either any of the Eqs. (11)–(15), and the residuals should be quantised using quantisation constant Q_m:

$$Q_m = \frac{QV}{N_B + 1}. \tag{23}$$

6. Results

We have tested the proposed algorithm with an implementation of the Coddyac compression scheme. We have used a direct encoding of the basis (64 bits per double precision value), LPC predictors of order from 1 to 4 and the four data independent physical predictors defined by (11), (12), (14) and (15). We have also used both uniform and non-uniform quantisation and the quantisation according to Heu *et al.*, to test the efficiency of our scheme.

For the testing of accuracy, we have used the KG-error metric proposed by Karni and Gottsman [KG04]. This metric is basically a normalised mean-squared error; however, it has several drawbacks that make it only a rough estimation of the introduced distortion. We are aware of the drawbacks of this error metric; however, it has been accepted by the dynamic mesh compression community, and therefore, we use it to provide results comparable with competing algorithms.

First, we will test the efficiency of the non-uniform quantisation using the chicken run sequence(3030 vertices, 400 frames). We have configured the Coddyac algorithm so that the quantisation of coefficients is very fine, and thus the error is mainly introduced by basis quantisation. First, we have evaluated the error for the uniform and non-uniform quantisation. Subsequently, we have observed what happens if the quantisation constant used for i-th basis vector is tripled.

Figure 6 shows the dependence of KG error on the index of artificially increased quantisation for the case of uniform quantisation. The error value constantly decreases with the index, showing that tripling the quantisation constant used for the first basis vector has a radical effect on the error, whereas tripling, for example, the tenth quantisation constant, has virtually no effect at all. This means that the basis vectors with higher indices are quantised too finely, their quantisation can be much coarser without significant influence on the error.

On the other hand, Figure 7 shows the result of the experiment, using the proposed non-uniform quantisation constants. It shows that tripling any of the constants introduces error of roughly the same magnitude, which proves that the quantisation constants are well balanced and no basis vector is quantised too finely.

Figure 5 suggests that Heu *et al.* use finer quantisation for basis vectors with lower indices (eigenvalue is higher, and therefore, the quantisation constant is smaller) and coarser quantisation for basis vectors with higher indices. Indeed, figure 8 shows that basis vectors of low index are quantised too finely (tripling the quantisation constant used for the most significant basis vector had almost no effect at all), whereas the biggest error is caused by the coarse quantisation of less important basis vectors.

Now we will compare the efficiency of the predictors. We will compare the entropies of residuals, using the

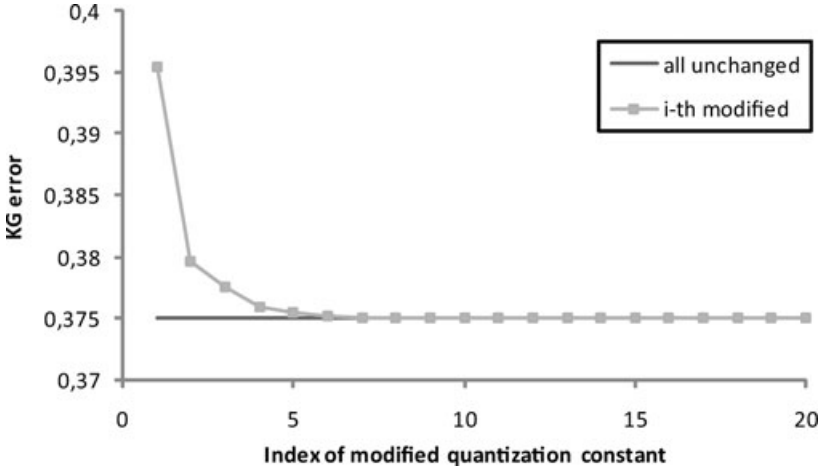

Figure 6: *Experiment with uniform quantisation.*

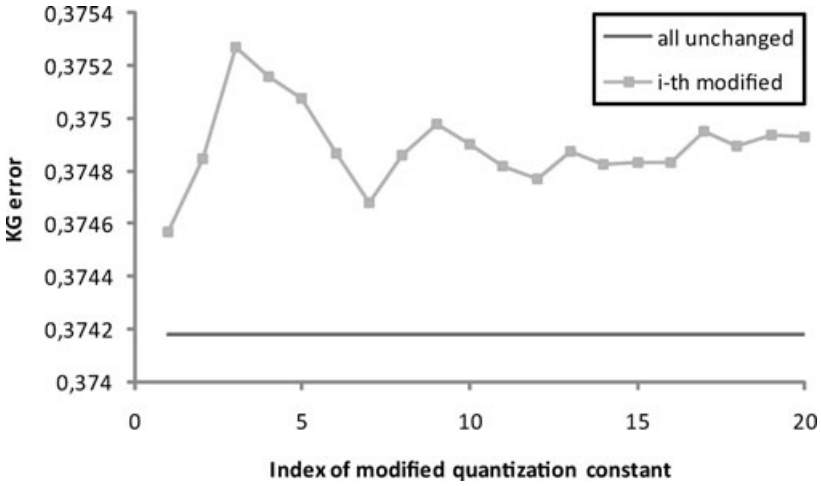

Figure 7: *Experiment with non-uniform quantisation.*

non-uniform quantisation, which has been shown to work well. Table 1 shows the entropies for the LPC predictor of order from 1 to 4, denoted by LPC1–LPC4, and the data independent physical predictors defined by (11), (12), (14) and (15), denoted by Cob1–Cob4. The table shows that the physical predictors outperform the LPC predictors for all the available data sets. Note that the sum of squared residuals has decreased with the order of LPC predictor, and the physical predictor has always produced residuals with higher sum of squares than the LPC of corresponding order, which is the expected behaviour. This, however, does not affect the final compression error, which depends on the selected quantisation constant, since we are transmitting the quantised prediction residuals.

Usually best results are provided by linear motion or accelerated motion predictors, with only two exceptions: for the dolphin data set at 25 basis vectors, the fourth-order predictor works best, and for the snake sequence at 50 basis vectors, the first-order predictor works best. The dolphin case is caused by the character of the data set, where there are basically only two important basis trajectories of the sine and cosine shape. These trajectories are very smooth, and thus well predicted by a higher order predictor. The snake case is the exact opposite – the movement of the snake is very fast and shaky, the basis trajectories are not smooth at all, and therefore, they are best predicted by simple delta coding. The most efficient overall approach is to test the performance of the physical predictors in the encoder, and then spend two bits to determine the predictor to be used.

Next we will compare the performance with state of the art algorithms. PCA based algorithms are generally considered obsolete, outperformed by the current state of the art.

Table 1: *Residual entropies for various predictors. The green colour coded values denote low entropy, the red values denote high entropy within the given row.*

dataset	basis	KG	LPC1	LPC2	LPC3	LPC4	Cob1	Cob2	Cob3	Cob4
jump	50	0,78	9,22	8,46	8,21	8,22	9,11	8,14	7,96	8,30
jump	25	1,42	9,12	8,18	7,69	7,66	9,01	7,82	7,41	7,59
chicken	50	0,48	9,87	9,90	9,87	9,87	9,10	8,11	8,21	8,70
chicken	25	2,36	9,38	9,30	9,29	9,30	8,86	7,77	7,83	8,32
cow	50	0,66	10,77	10,59	10,57	10,57	10,69	10,45	10,62	11,09
cow	25	1,42	11,06	10,75	10,66	10,66	11,00	10,55	10,56	10,90
dolphin	50	0,24	6,07	6,06	6,07	6,08	5,71	5,55	5,55	5,79
dolphin	25	0,24	7,12	6,97	6,76	6,73	6,68	6,01	5,61	5,54
dance	50	0,36	10,08	10,07	10,06	10,07	9,93	9,61	9,84	10,43
dance	25	0,76	10,22	9,60	9,47	9,48	10,14	9,43	9,41	9,89
cloth	50	0,38	9,61	9,53	9,49	9,50	8,93	7,78	7,78	8,33
cloth	25	1,03	9,63	9,26	9,07	9,11	9,26	7,72	7,50	7,95
snake	50	0,22	10,30	10,20	10,11	10,11	10,04	10,23	10,87	11,62
snake	25	0,31	10,38	10,42	10,21	10,15	10,18	9,99	10,46	11,14
walk	50	0,32	8,93	8,80	8,82	8,83	8,78	8,40	8,79	9,49
walk	25	0,37	9,06	8,53	8,55	8,57	9,00	8,29	8,57	9,22

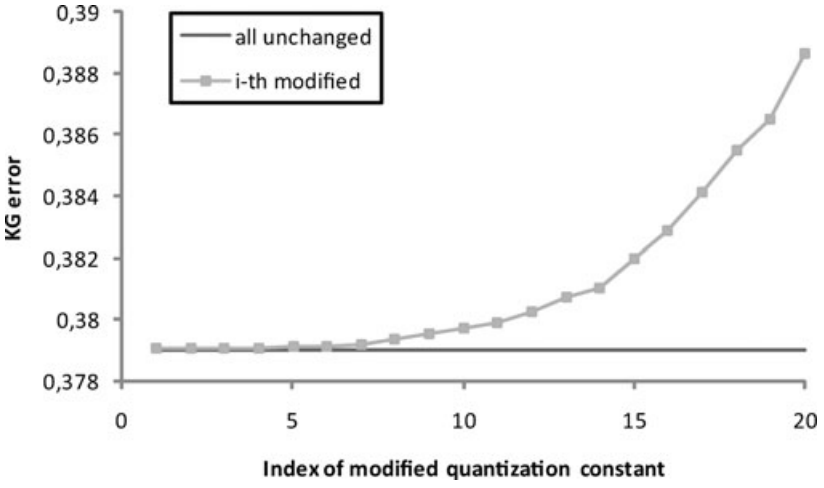

Figure 8: *Experiment with quantisation according to Heu et al. [HYKL06].*

Figure 9 indeed shows that PCA based schemes, such as Coddyac, are outperformed by the current state of the art, namely the FAMC algorithm. This algorithm has been adopted in 2007 as the new MPEG standard, and it has been shown that it outperforms other algorithms such as [SSK05] or [Amj07].

The Figure 9 also shows, that it is the improvement by Cobra that moves the performance of Coddyac beyond the performance of FAMC. Note that FAMC is not a PCA based algorithm, PCA in the legend refers only to one subpart of the FAMC algorithm.

We have performed experiments with other data sets of varying nature, such as scanned real world action (human jump – [SMP03, AG04], see Figure 10), modelled animations (chicken sequence, see Figure 11), modelled human motion (walk sequence, created by Poser) or garment simulation (falling cloth sequence, created by garment simulation in 3ds max), reaching equivalent results – enhancing PCA based compression by adding our encoding of basis boosts the performance of the algorithm beyond the results of the known approaches.

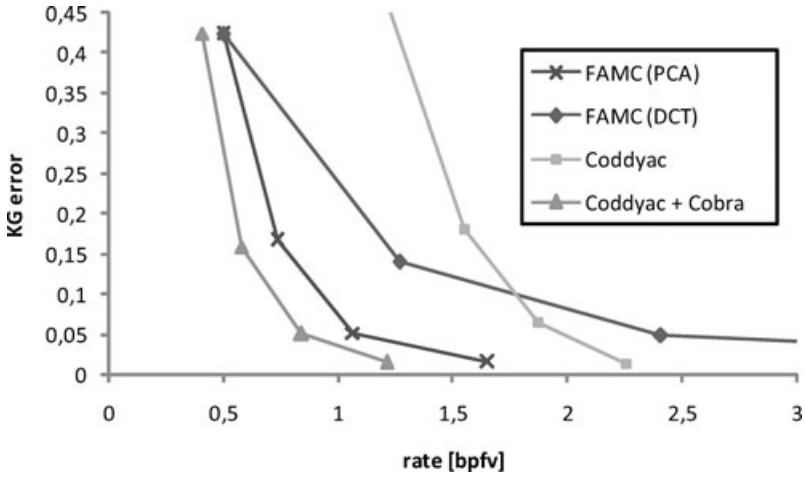

Figure 9: *Comparison against state of the art algorithms on the dance data set.*

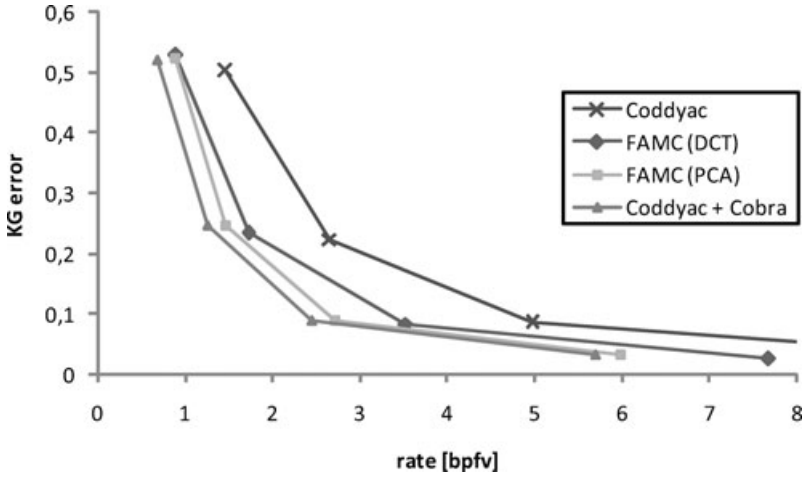

Figure 10: *Comparison against state of the art algorithms on the human jump data set.*

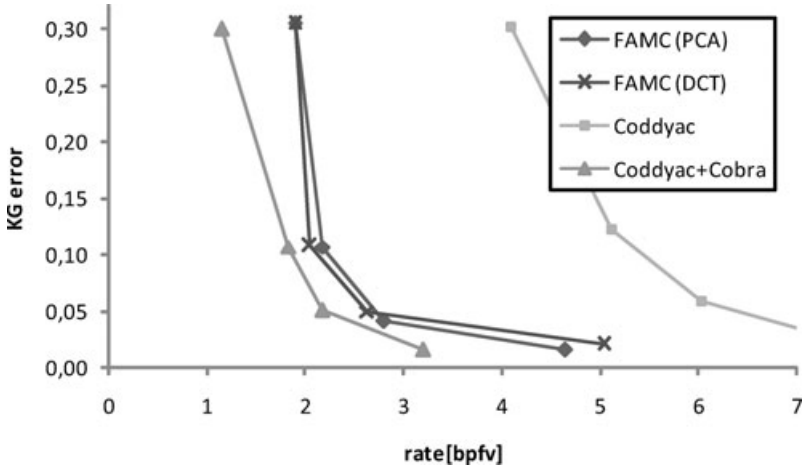

Figure 11: *Comparison against state of the art algorithms on the chicken data set.*

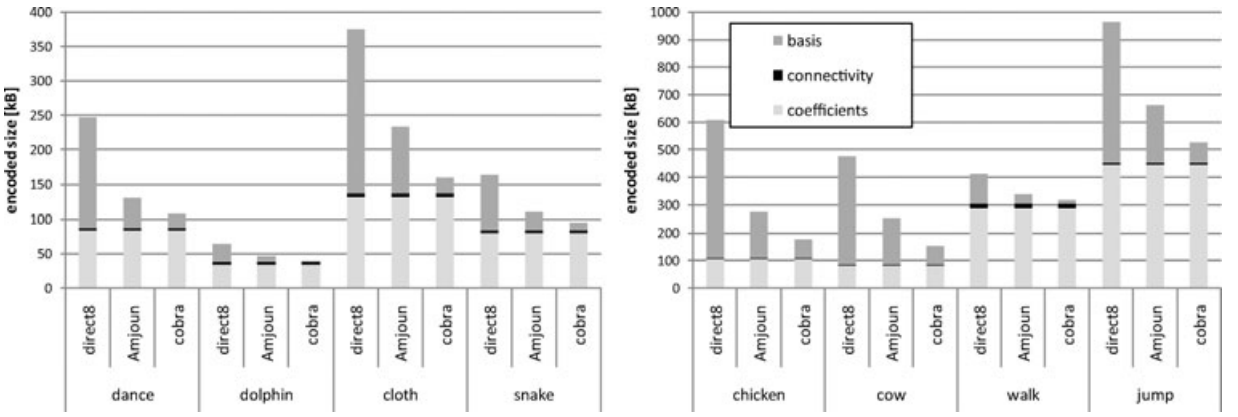

Figure 12: *Relations of coefficients, basis and connectivity for various data sets. Direct8 denotes direct encoding of 8-bytes per double value, Amjoun denotes the result of quantisation without prediction, and cobra denotes our result.*

Finally, we give the relations of different parts of encoded data for all the data sets in Figure 12. All the data sets are compressed at KG error of approximately 0.3, and we are giving results for direct encoding (8 bytes per float number), encoding suggested by Amjoun (uniform quantisation) and our algorithm.

7. Conclusions and Future Work

We have presented a new work in the area of compression of basis for PCA represented animations. The presented algorithm can be directly applied to multiple current approaches ([SSK05; Amj07; VS07]), where it delivers an improvement of performance, which depends on the character of the animation and can reach up to more than 50% compared with direct encoding. Such improvement shifts the performance of the PCA based algorithms beyond the performance of current state of the art algorithms of various kinds (i.e. even the ones not based on PCA, such as FAMC).

We have presented a straightforward idea of predictive coding with non-uniform quantisation, however, our main contribution is the answer to questions 'How to determine efficient quantisation constants ? ' and 'What predictor should be used ? ', which is the key to efficient encoding.

In the future, we will focus on further compression of the basis exploiting its smoothness by tools like wavelet decomposition and scalable encoding.

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).

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.

References

[AG04] ANUAR N., GUSKOV I.: Extracting animated meshes with adaptive motion estimation. In *VMV* (2004), pp. 63–71.

[AM00] ALEXA M., MÜLLER W.: Representing animations by principal components. *Computer Graphics Forum 19*, 3 (2000), 411–418.

[Amj07] AMJOUN R.: Efficient compression of 3d dynamic mesh sequences. In *Journal of the WSCG 15* (2007), 99–106.

[BSM*03] BRICENO H. M., SANDER P. V., MCMILLAN L., GORTLER S., HOPPE H.: Geometry videos: A new representation for 3d animations. In *ACM Symposium on Computer Animation 2003* (2003), pp. 136–146.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (2002), ACM Press, pp. 355–361.

[Huf52] HUFFMAN D.: A method for the construction of minimum-redundancy codes. *PIRE 40*, 9 (Sept. 1952), 1098–1101.

[HYKL06] HEU J., YANG J.-H., KIM C.-S., LEE S.-U.: Effective quantisation scheme for principal components of 3-d mesh sequences. *IEE Electronics Letters 42*, 14 (July 2006), 799–800.

[IR03] IBARRIA L., ROSSIGNAC J.: Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), Eurographics Association, pp. 126–135.

[KG04] KARNI Z., GOTSMAN C.: Compression of soft-body animation sequences. In *Computers & Graphics* (2004), vol. 28, pp. 25–34.

[Len99] LENGYEL J. E.: Compression of time-dependent geometry. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), ACM Press, pp. 89–95.

[MSK*05] MULLER K., SMOLIC A., KAUTZNER M., EISERT P., WIEGAND T.: Predictive compression of dynamic 3d meshes. In *ICIP05* (2005), pp. I-621–I-624.

[MSK*06] MULLER K., SMOLIC A., KAUTZNER M., EISERT P., WIEGAND T.: Rate-distortion-optimized predictive compression of dynamic 3d mesh sequences. *SP:IC 21*, 9 (Oct. 2006), 812–828.

[MZP06] MAMOU K., ZAHARIA T., PRETEUX F.: A skinning approach for dynamic 3d mesh compression: Research articles. *Comput. Animat. Virtual Worlds 17*, 3–4 (2006), 337–346.

[PA05] PAYAN F., ANTONINI M.: Wavelet-based compression of 3d mesh sequences. In *Proceedings of IEEE ACIDCA-ICMI '2005* (Nov. 2005), pp. 77–88.

[Ros99] ROSSIGNAC J.: Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (1999), 47–61.

[SLKO07] STEFANOSKI N., LIU X., KLIE P., OSTERMANN J.: Scalable linear predictive coding of time-consistent 3d mesh sequences. In *3DTV-CON, The True Vision - Capture, Transmission and Display of 3D Video*, Kos, Greece (May 2007).

[SMP03] SAND P., MCMILLAN L., POPOVIČ J.: Continuous capture of skin deformation. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (2003), ACM Press, pp. 578–586.

[SO06] STEFANOSKI N., OSTERMANN J.: Connectivity-guided predictive compression of dynamic 3d meshes. In *Proc. of ICIP '06 - IEEE International Conference on Image Processing* (Oct. 2006).

[SSK05] SATTLER M., SARLETTE R., KLEIN R.: Simple and efficient compression of animation sequences. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2005), ACM Press, pp. 209–217.

[VS07] VÁŠA L., SKALA V.: Coddyac: Connectivity driven dynamic mesh compression. In *3DTV-CON, The True Vision - Capture, Transmission and Display of 3D Video*, Kos, Greece (May 2007).

[Wei02] WEISSTEIN E. W.: Least squares fitting, 2002. From Mathworld – a Wolfram Web Resource. http://mathworld.wolfram.com/LeastSquaresFitting.html.