Computer Graphics Forum, Vol.28, No.6, pp.1529-1540, ISSN 0167-7055, 2010

COBRA: Compression of basis of PCA represented animations

L. Váša[1] and V. Skala[1]

[1]Department of Computer Science and Engineering, Faculty of Applied Science, 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 the side 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 pioneering 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 a 25% increase of performance of the compression algorithm without any significant loss of accuracy.

Categories and Subject Descriptors (according to 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 solve and properties to exploit.

Generally, the task of dynamic mesh compression is the following: The input is a series of static triangular meshes $M_1..M_n$, 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 approaximate 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 to use 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. Most of the approaches then focus on the efficient encoding of the coefficient vectors, while 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. Therefore, we have derived a compression scheme for the basis, which is based on non-uniform quantization and non-least squares optimal linear prediction, which addresses this issue.

The rest of the paper is organized as follows: Section 2 will briefly describe existing approaches to dynamic mesh compression with special detail about PCA based approaches. Section 3 will sketch some possible approaches and derive the choices we have made to construct our algorithm. Section 4 will give detail about the non-uniform quantization we are using to increase efficiency. Finally, section 5 will present results of our approach applied to one algorithm based on temporal PCA and section 6 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. The position of the new vertex is expressed in local coordinate system defined by a neighboring 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 behavior 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 eigen frames. 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 initialized.

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.

Finally, Vasa and Skala [VS07] have presented a trajectory-based PCA coding combined with EdgeBreaker-like [Ros99] predictor. Their Coddyac algorithm predicts the PCA coefficients by the well known paralellogram 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 neighboring 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 minimizing the overall tension. Generally, the method is not easy to implement and suffers from some artifacts, especially for objects with complex geometry.

Recently, 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.

3. Algorithm derivation

We will demonstrate the effect of basis compression on a representative of the class of PCA based compression algorithms, the Coddyac scheme. We will breifly describe it in order 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 $V \times 3F$, where each row represents a trajectory of a single vertex over the duration of the animation:

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

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

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

An autocorrelation matrix $M^T M$ is constructed and its eigenvectors are found. These eigenvectors are then used as a new basis of the row 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 row (trajectory $\mathbf{t}$) is then expressed in the new basis,

while 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.

The decompression is then quite simple. The decoder extracts the connectivity and the basis in a form of $N_B \times 3F$ matrix B. Then 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 a vector $\mathbf{c}$ of N_B assigned.

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

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

The key observation for following derivations is that the basis vectors, which need to be encoded, still retain the character of trajectories. In other words, if one inteprets one 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.

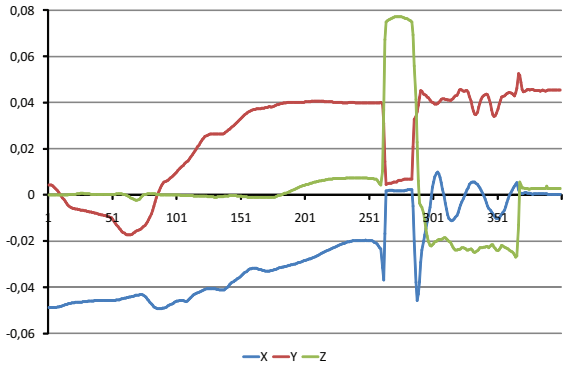

Figure 1: *First basis vector of the chicken sequence. The sudden jump in frames 250-300 is the chicken popping eyes, the subsequent sinusoidal development of the X axis is the flapping of wings.*

This observation has been made previously by Karni and Gotsman in [KG04], who noticed 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 behavior) and their values are found in a least squares optimization process applied by the encoder on the whole sequence. For more details see the original source.

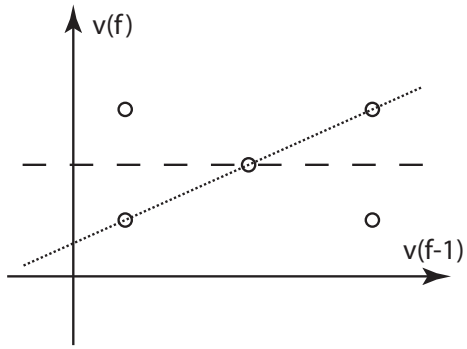

Figure 2: *The two linear predictors for a given data set (each data point is represented by a circle).*

We have first followed this suggestion for the basis as well, however we have found that for the purposes of efficient encoding it is sub-optimal. Imagine a situation depicted in figure 2. In this simplified scenario, we are given a value v_{f-1} (preceeding value, one of the XYZ coordinates) and we want to predict the value v_f by a linear formula

$$v_f = kv_{f-1} + q \tag{4}$$

The LPC algorithm suggests to apply least squares minimization to find the values k and q, in our case we will get the dashed line. If we now quantize the residuals, then unless we get all zeroes, we'll get residual entropy as follows:

$$E = -\sum p\log_2(p) \tag{5}$$
$$= -(\frac{2}{5}\log_2(\frac{2}{5}) + \frac{2}{5}\log_2(\frac{2}{5}) + \frac{1}{5}\log_2(\frac{1}{5})) \tag{6}$$
$$= 1.522[b] \tag{7}$$

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

$$E = -\sum p\log_2(p) \tag{8}$$
$$= -(\frac{3}{5}\log_2(\frac{3}{5}) + \frac{1}{5}\log_2(\frac{1}{5}) + \frac{1}{5}\log_2(\frac{1}{5})) \tag{9}$$
$$= 1.379[b] \tag{10}$$

This rather artificial example shows that there are cases, when least squares solution leads to sub-optimal prediction. Indeed, it is a generally known problem [Wei02] of the least squares optimization that the solution can be lead astray by

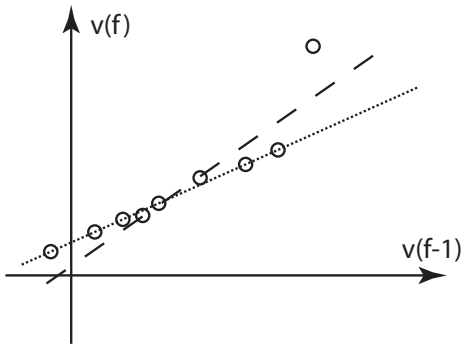

Figure 3: *Least squares solution is lead away from a good predictor of most of the data by an outlier.*

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.

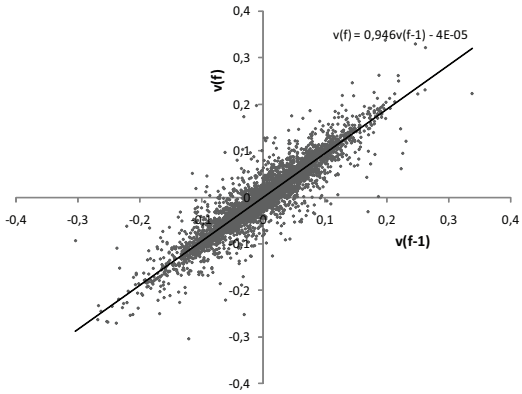

Figure 4: *Real data prediction.*

Figure 4 shows a real world example. The dataset is the PCA basis of the first 100 frames of the chicken sequence. The samples show the dependency of a basis value (v_f axis) on its predecessor (v_{f-1} value) where appropriate predecessor is available. A least squares fitting of such data produced the depicted line, which can be used to predict v_f from v_{f-1}. However, if we consider the real situation, we can derive a simple formula in a form $pred(v_f) = v_{f-1}$, which is least-squares sub-optimal, however it delivers residuals with smaller entropy.

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're trying to predict. We can either use least squares fitting, or linear movement prediction in a form

$$pred(v_f) = v_{f-1} + (v_{f-1} - v_{f-2}) = 2v_{f-1} - v_{f-2} \quad (11)$$

The last predictor that we have experimented with uses three preceding values (v_{f-3}, v_{f-2}, v_f) to predict the current value v_f, estimating the speed s and the acceleration a to obtain prediction as follows:

$$s = v_{f-1} - v_{f-2} \quad (12)$$

$$a = (v_{f-1} - v_{f-2}) - (v_{f-2} - v_{f-3}) \quad (13)$$

$$= v_{f-3} - 2v_{f-2} + v_{f-1}$$

$$pred(v_f) = v_{f-1} + s + a = 3v_{f-1} - 3v_{f-2} + v_{f-3} \quad (14)$$

The overall prediction algorithm must also prevent error accumulation by using quantized values in the encoder. Thus, the general scheme is as follows:

input: basis vector B_i
input: quantization 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 $coord \leftarrow 0$ **to** 2 **do**
  for $j \leftarrow (1 + coord * F)$ **to** $(o + coord * F)$ **do**
    $q \leftarrow round(B_{i,j}/Q_i)$;
    send q to entropy coder;
    $B_{i,j} \leftarrow q * Q_i$;
  end
  for $j \leftarrow (o + 1 + coord * F)$ **to** $(F + coord * F)$ **do**
    $pred \leftarrow predictor(B_{i,j-1}, B_{i,j-2}, ..., B_{i,j-o})$;
    $residual \leftarrow B_{i,j} - pred$;
    $q \leftarrow round(residual/Q_i)$;
    send q to entropy coder;
    $B_{i,j} \leftarrow pred + q * Q_i$;
  end
end

4. Quantization

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

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

$$\mathbf{t} = B_1.c_1 + B_2.c_2 + \cdots + B_{N_B}.c_{N_B} + \mathbf{m} \quad (15)$$

where B_i represent the rows of the matrix B, i.e. the basis vectors. The error introduced by the quantization of the PCA coefficients c_i is equal for each term of equation 15, as all the coefficients are quantized with equal quantization 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 well predicted - the first coefficient is usually much bigger than the second, which is bigger than the third etc., which is a behavior caused by the nature of PCA.

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

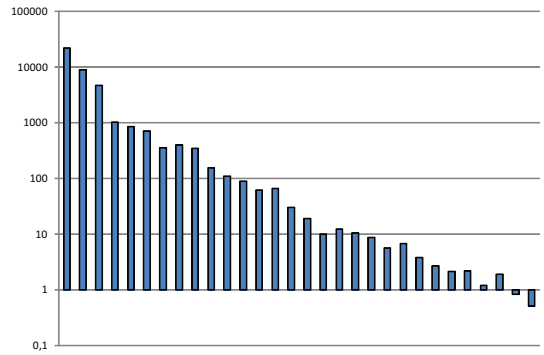

Figure 5: *Coefficient sums.*

When given a quantization constant Q from user, we can split this constant to the given number of terms of equation 15, however we should follow the magnitudes of corresponding coefficients c_i. To do this, we can first compute the average

$$a_i = \frac{1}{V} \sum_{v=1..V} \|c_i^v\| \tag{16}$$

of absolute values of each coefficient over all vertices $v_{1..V}$. This gives an approximation of how many times will an error 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 equation 15 equal, then we must use quantization constants inversely proportional to the averages of coefficients. Thus, the quantization constant for i-th basis vector should be computed as:

$$Q_i = \frac{Q}{N_B + 1} a_i^{-1} \tag{17}$$

In this equation i is the order of the basis vector and Q is the quantization constant, which must be divided into $N_B + 1$ parts, since the equation 15 has $N_B + 1$ terms.

We could use an exact value for each coefficient, however that would require to transmit the quantization constant with each basis vector. To avoid this, we have decided to use a power function approximation to compute the quantization constants for each basis vector at both encoder and decoder. We perform a least squares minimization to obtain the constants k and q in the equation 18.

$$k i^q = Q_i \tag{18}$$

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

$$k.1^q + a = k + a = Q_1 \tag{19}$$

$$a = Q_1 - k \tag{20}$$

This way, we only need to transmit the constants k, q and a, and we get a series of increasing quantization 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 equation 11 or equation 14 and the residuals should be quantized using quantization constant Q_m:

$$Q_m = \frac{Q}{N_B + 1} \tag{21}$$

5. Results

We have tested the derived algorithm with an implementation of the Coddyac compression scheme. We have used a direct encoding of the basis (64 bits per double precision value), the prediction schemes defined by equations 11 (denoted "linear") and 14 (denoted "accelerated") combined with uniform quantization and with non-uniform quantization. We are also comparing to LPC approach of order 2 and order 3. As an entropy coder for the residuals we have used a simple implementation of Huffman coding [Huf52].

First, we have used the well known chicken crossing sequence of 3030 vertices and 400 frames. For this sequence, we have used 50 basis vectors. Second, we have performed tests with the human jump sequence [SMP03, AG04], which consists of 15700 vertices and 222 frames. For this dataset, we have used 60 basis vectors. We have also used other datasets (cowheavy, dolphin, humanoid, dance etc.), with equivalent results.

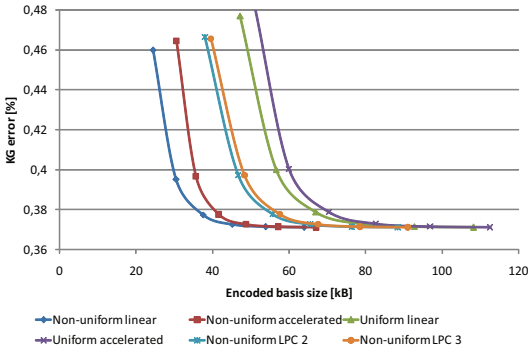

Figure 6: *Chicken sequence compression accuracy.*

For the accuracy testing, we have used the KG-error metric proposed by Karni and Gottsman [KG04]. This metric is basically a normalized mean squared error, however it processes the XYZ coordinates separately and thus it is not rotation invariant, and its value also depends on the length of the sequence. 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.

We have used several datasets to test the performance, but for each dataset we have used constant settings for the compressor, as we're not testing the performance of the compression scheme, but of the basis compression. We have selected such settings that have produced a KG error of about 0.5%, which is considered unnoticeable.

Figure 6 shows the accuracy testing for the chicken sequence with uniform and non-uniform quantizations. The direct coding of the basis would take

$$400[vertices] * 3[coordinates] * 50[basisvectors]... \\ *8[bytes/doublevalue] = 480kB$$

and the error value in such case is about 0,371% (this error is introduced by truncating the basis and by quantization of the coefficients). The figure clearly shows that the non-uniform quantization largely outperforms the uniform case. Surprisingly, the linear prediction scheme provides better results than the accelerated one, however both of these straightforward schemes outperform the LPC approach significantly.

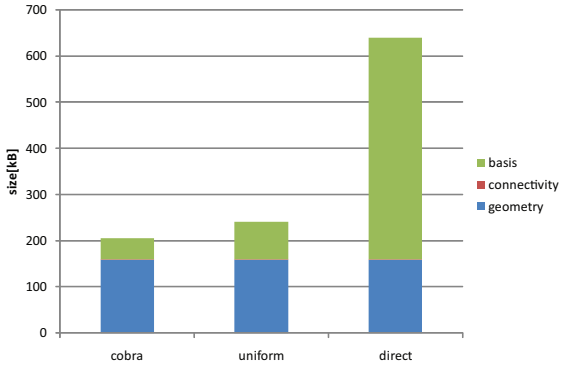

Figure 7: *Relations in the chicken sequence compression. Accuracy of the examples is equal* $\pm 1\%$.

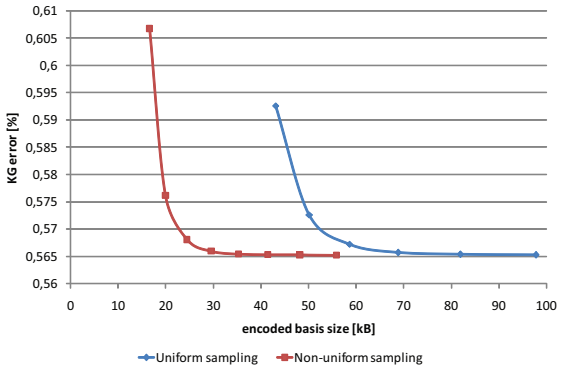

Figure 8: *Jump sequence compression accuracy.*

The gain of the linear prediction scheme can be explained by the fact that in such case less values need to be encoded directly, and it moreover seems that the third preceding value brings more distortion than accuracy into the prediction. Another explanation is that the chicken sequence has been created artificially, and as such does not necessarily follow the rules of mass dynamics.

Figure 7 shows the relation of the parts of which the compressed representation of the chicken sequence consists. It shows that when not treated appropriately, the basis may cause a significant performance loss.

Figures 8 and 9 are equivalent results for the jump sequence. We are not showing the result of acceleration based predictor, as in this case it is almost identical to the result of the linear predictors. This supports our explanation that the efficiency of the acceleration predictor follows from the physical properties of the data, because the jump sequence is a result of scanning a real human.

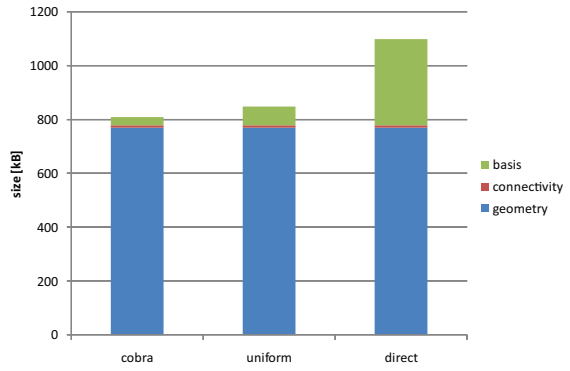

Figure 9: *Relations in the jump sequence compression. Accuracy of the examples is equal* $\pm 1\%$.

6. Conclusions and future work

We have presented a pioneering 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 to direct encoding.

Our main contribution is the result of testing of the LPC method, which despite its sophistication provides sub optimal results with respect to encoding and residual entropy. Our second contribution is the non-uniform quantization technique, which is mathematically based and practically proven to be efficient.

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

7. 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) and project 1P04LA240.

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).

[Amj07] AMJOUN R.: Efficient compression of 3d dynamic mesh sequences. In *Journal of the WSCG* (Feb. 2007). to appear.

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

[GGH02] GU X., GORTLER S., HOPPE H.: Geometry images, 2002.

[Huf52] HUFFMAN D.: A method for the construction of minimum-redundancy codes. 1098–1101.

[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* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 126–135.

[KG04] KARNI Z., GOTSMAN C.: Compression of soft-body animation sequences. In *"Computers & Graphics 28, 1"* (2004), pp. 25–34.

[Len99] LENGYEL J. E.: Compression of time-dependent geometry. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 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: 621–624.

[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 (October 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* (Tozeur, Tunisia, november 2005).

[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* (may 2007), vol. 0.

[SMP03] SAND P., MCMILLAN L., POPOVIČ J.: Continuous capture of skin deformation. In *SIGGRAPH '03:*

ACM SIGGRAPH 2003 Papers (New York, NY, USA, 2003), ACM Press, pp. 578–586.

[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 Conference Proceedings* (2007).

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