Subtractive Clustering of Vertices for CPCA based Animation Geometry Compression

Sanjib Das*
Department of ECE
IIT Guwahati
Guwahati-781039, India
sanjib@iitg.ernet.in

Prabin Kumar Bora
Department of ECE
IIT Guwahati
Guwahati-781039, India
prabin@iitg.ernet.in

Anup Kumar Gogoi
Department of ECE
IIT Guwahati
Guwahati-781039, India
akg@iitg.ernet.in

ABSTRACT

In the Clustered PCA(CPCA) algorithm for compressing the animation geometry sequences, the vertex trajectories are clustered using the K-means algorithm followed by the Principal Component Analysis(PCA) of the clusters. However, the compression performance of the method is constrained by the initial random selection of the cluster centres. This paper presents a stable method for initializing the cluster centres by the subtractive clustering technique prior to the application of the K-means algorithm. Simulation results on some test animation sequences show better performance of the CPCA with the proposed initialization compared to the CPCA with random initialization.

Keywords

Subtractive Clustering, Animation Geometry Compression, CPCA

1. INTRODUCTION

With the recent advancement of computer graphics hardware and related software in recent decades, the need and popularity of multimedia applications featuring 3D objects in the form of 3D-animations have increased extensively. For graphics rendering, 3D objects are represented in polygonal mesh forms, mostly in triangular mesh forms consisting of vertices, edges and faces. The attributes describing a triangular mesh are: (i) 'geometry' components defining the positions of vertices in each mesh, (ii) 'connectivity' information depicting how the triangles are associated with the vertices, (iii) 'surface color' defining color of the mesh, (iv) 'surface normal' representing the orientation of object and (v) 'texture' quantifying the variation in intensity to account for surface roughness or smoothness. This representation has been used in the hardware of all the manufacturers of leading graphics rendering cards. 3D mesh models with complex structures having all the above attributes always require a large storage space and reconstruction time for visualization. For compact storage, the meshes are to be compressed heavily. The reliable transmission of such meshes through the band limited channel also calls for compression of the animation data.

3D animations are defined as a sequences of frames having 3D dynamic meshes with changing attributes over time. Animations again can be broadly classified into two groups: (a) animations with rigid-body motion where the whole mesh moves as one entity because all the vertices will be having fixed positions and (b) animations with soft-body motion [5] where each vertex can move independently from other vertices, making the animation smooth and realistic. However, the rigid-body motion is not practical, as it can not capture the smooth and realistic motion required for live animations. Soft-body motion does not require any relation between the vertices in a mesh, as long as they form a meaningful mesh.

The soft-body animated meshes are widely used in computer games, computer generated movies and other 3D scientific simulations and commercial applications. Because of the independent movement of each vertex across frames, animation as a whole consists of voluminous data, and this is the major disadvantage of soft-body animation. Therefore, raw animation data is not practical for real time transmission over the limited bandwidth. It is important to develop a compressed representation that significantly reduces the storage space required for animation still maintaining a good visual quality. Therefore, the compression of 3D animation, specifically of soft-body animation, has produced much interest among researchers.

In general animations, the geometry and connectivity of vertices in a 3D mesh may be changing over time. However for mesh sequences whose connectivity remains constant over time, the animation is characterized by changes in geometry only. Such sequences are termed as dynamic geometry sequences.

The Principal Component Analysis (PCA)[4] is a method that reduces the data dimensionality while retaining the essential variation present in the original dataset. It searches for directions in the data that have largest variance and subsequently project the data onto it. The dimensionality reduction is achieved by performing the covariance analysis of the data and approximating the data in terms of a few principal eigen vectors of the covariance matrix. These principal eigen vectors correspond to the largest eigen values of the covariance matrix. As such, it is suitable for data sets in multiple dimensions, such as animation with large number

*Corresponding author

ICVGIP '10, December 12-15, 2010, Chennai, India

of frames.

This work aims at enhancing the potential of the PCA as an animation compression tool by combining it with efficient clustering of vertex trajectories. We deal only with the geometry part of the animation sequence for compression and treat the connectivity as constant for the whole animation sequence.

2. PREVIOUS WORK

Lengyel [7] was the first researcher to work on the compression of dynamic meshes. He proposed to segment the mesh into smaller sub-meshes, and the motion of these parts was described as that of the rigid body. Only a heuristic solution was provided for the segmentation process. His approach is effective only when the animation primarily consists of the rigid-body motion. Alexa et al. [1] used the PCA for compressing the geometry component of animation sequences. The PCA was applied in the temporal direction and only a few principal components were used to represent the whole animation. This was a lossy compression technique and the amount of loss was controlled by the required compression ratio or the visual reconstruction quality. Karni and Gotsman [5] enhanced the PCA based method of [1] by combining the linear prediction coding (LPC) with the PCA. The PCA was applied in spatial direction and the second order LPC was applied on the principal components in the temporal direction to exploit the large temporal coherence present in the sequence.

In [10], clustering is combined with the PCA in a method called the clustered PCA (CPCA). First the mesh was segmented into meaningful clusters based on motion of vertices by using Lloyd's clustering method [8]. Then the PCA is applied independently on each cluster and using only few PCA components for each cluster gives compression. This scheme is reported to give better results than the direct PCA [1] and the PCA with LPC [5] methods. Amjoun et. al. [2] developed a new PCA based method called the local PCA (LPCA) method by clustering the meshes using local similarity properties, and a local coordinate system is defined for each cluster. Now the cluster motion is encoded by applying the PCA on this local coordinates instead of world coordinates. Results indicate that the LPCA has better performance than CPCA and other previous methods. Recently a new PCA based animation compression scheme called COBRA [12] was proposed by L. Vasa et. al.. This method applies PCA on the temporal direction of the animation matrix and then uses a new prediction scheme along with non-uniform quantization scheme to encode the basis vector. The results presented by them show compression performance better than existing PCA based methods.

The above literature review suggests that most of the existing methods use the PCA as the important block for compression. This work is concerned with the application of the PCA on the clusters of vertex trajectories for better compression. As in [10], the K-means based clustering algorithm is used for grouping the vertex trajectories of the animation sequence based on the temporal information of the vertex positions across the frames. A stable method to initialize the cluster centers for K-means algorithm is proposed using the subtracting clustering method [3].

3. ANIMATION GEOMETRY COMPRESSION USING THE CPCA

3.1 Representation of 3D Animation Geometry

Simple animations can be described by equations modeling the trajectories of the vertices across frames. In general, the movement of vertices across different frames of a sequence can be represented by a matrix. Consider an animation sequence of F frames with V vertices in each frame. Each vertex is basically a point in 3D space having three coordinate components v_x, v_y and v_z corresponding to X, Y and Z axes. Therefore, the animation sequence is represented by a matrix $\mathbf{B}$ of dimensions $V \times 3F$ given by

$$\mathbf{B} = \begin{bmatrix} v_{x_1}^{f_1} & v_{y1}^{f_1} & v_{z_1}^{f_1} & ... & v_{x_1}^{f_F} & v_{y_1}^{f_F} & v_{z_1}^{f_F} \\ v_{x_2}^{f_1} & v_{y2}^{f_1} & v_{z_2}^{f_1} & ... & v_{x_2}^{f_F} & v_{y_2}^{f_F} & v_{z_2}^{f_F} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ v_{x_V}^{f_1} & v_{yV}^{f_1} & v_{z_V}^{f_1} & ... & v_{x_V}^{f_F} & v_{yV}^{f_F} & v_{z_V}^{f_F} \end{bmatrix} = \begin{bmatrix} \mathbf{t_1} \\ \mathbf{t_2} \\ \vdots \\ \mathbf{t_V} \end{bmatrix} \quad (1)$$

where $v_{x_i}^{f_j}$, $v_{y_i}^{f_j}$ and $v_{z_i}^{f_j}$ represents the x, y and z coordinates respectively of the ith vertex $v_i, i = 1, \ldots, V$ of the jth frame $f_j, j = 1, \ldots, F$ of the animation sequence. In the matrix $\mathbf{B}$, each row [size $3F \times 1$] represents a trajectory $\mathbf{t_i}$ of a single vertex v_i across F frames and each column [size $V \times 1$] represents the values any of the three coordinate components v_{x_i}, v_{y_i} or v_{z_i} of a vertex v_i in a frame f_j.The main idea of any geometry compression algorithm is to represent this matrix $\mathbf{B}$ in a more compact way for efficient storage and transmission.

3.2 Geometry Compression using the Direct PCA

In this case, the straight-forward approach will be to apply the PCA on the animation data matrix $\mathbf{B}$ directly. Performing the PCA on the matrix involves the following steps [11]:

1. Estimate and subtract the mean trajectory μ from all trajectories or rows of $\mathbf{B}$ to obtain the mean centered matrix $\mathbf{B_{ms}}$.

2. Calculate the eigenvalues and the corresponding eigenvectors of the covariance matrix $\mathbf{B_{ms}^T B_{ms}}$.

3. Consider only c ($c << 3F$) principal eigenvectors based on their eigen values to represent the animation data by projecting the original mean subtracted data on to the space spanned by these c eigen vectors and retain the corresponding PCA coefficients.

3.3 Geometry Compression using the CPCA method

The vertex trajectories in the animation matrix $\mathbf{B}$ are grouped into clusters based on their similar movements across all the frames and then the PCA is applied on each cluster. Generally the K-means algorithm is used for clustering applications for its simplicity. For clustering of vertex trajectories, the Euclidian distances between the vertex trajectories are used as the similarity measure. The steps for the CPCA algorithm are as follow:

(a). For a given number of clusters K, K vertex trajectories $\mathbf{t_{c_j}}$, $j = 1, 2, \ldots, K$ are randomly chosen as initial cluster centers.

(b). Find the Euclidian distances of all the V vertex trajectories from these K centered trajectories and put them in a distance matrix $\mathbf{D}$ given by

$$D_{ij} = \left\|\mathbf{t_i} - \mathbf{t_{c_j}}\right\|^2 \ , \ i = 1, 2, \ldots, V \ , \ j = 1, 2, \ldots, K \tag{2}$$

(c). Assign each trajectory to its nearest cluster center based on minimum distance from the matrix $\mathbf{D}$ and put the cluster number it belongs to in an array of cluster index I_c.

(d). Update the cluster centers of each cluster with the mean of the vertex trajectories belonging to that cluster as

$$\mathbf{t_{c_j}} = \frac{1}{N_j} \sum \mathbf{t_k} \tag{3}$$

where N_j is the number of vertex trajectories in each cluster as per cluster index array I_c.

(e). Repeat the above steps (b),(c) and (d) until there is no change in the cluster index I_c.

After the clustering process, the input matrix $\mathbf{B}$ will be divided into K cluster matrices $\mathbf{B^{(j)}}$, $j = 1, \ldots, K$ i.e

$$\mathbf{B} = \left[\mathbf{B^{(1)}}|\mathbf{B^{(2)}}|\cdots|\mathbf{B^{(K)}}\right]^T \tag{4}$$

(f). Apply the PCA based compression on each cluster matrix $\mathbf{B^{(j)}}$ and c principal eigen vectors per cluster is selected based on their eigen values.

At the end of above steps, the original animation geometry matrix $\mathbf{B}$ will be represented by K numbers of Eigen vector matrix of size $[3F \times c]$, PCA coefficient matrix for each cluster of size $[N_j \times c]$, mean trajectory vectors for each cluster of size $[3F \times 1]$ and the cluster index vector I_c of size $[V \times 1]$. Finally all these PCA data can be quantized to achieve further compression.

The decompression is done by first de-quantizing all the PCA eigen vectors, PCA coefficients, mean vectors and cluster index vectors followed by reconstruction of all the clustered trajectory matrix using the decoded PCA eigen vectors, PCA coefficients and the mean trajectory vector. Finally, all the trajectory cluster matrix rearranged and combined based on cluster index vector to get the original reconstructed animation geometry matrix.

This CPCA method gives better compression results compared to the direct PCA method on the whole animation matrix $\mathbf{B}$. The reason is that the clustering process combines the vertex trajectories based on some similarity measure and hence there exists much redundant information to be exploited by applying PCA afterward.

4. PROPOSED SUBTRACTIVE CLUSTERING BASED CPCA (SC-CPCA) METHOD

The success of the above CPCA algorithm depends on the effectiveness of the clustering process. With random initialization of cluster centers, the K-means algorithm does not necessarily give the optimal partitions of the input vectors in each run as it may converges to numerous local minima instead of the global minima. Therefore, there is a scope for improving clustering process by proper initialization of cluster centers instead of random initialization[13, 6] which will results in better compression performance.

This work proposes to apply the density function based subtractive clustering method [3] to initialize the cluster centres for the K-means algorithm. Here a density function like Gaussian density function $\exp\left(-\alpha \left\|\mathbf{t_i} - \mathbf{t_j}\right\|^2\right)$ is assumed at each vertex trajectory, where α is a positive constant and $\left\|\mathbf{t_i} - \mathbf{t_j}\right\|^2$ is the square of the distance between any two vectors $\mathbf{t_i}$ and $\mathbf{t_j}$. The sum of the density functions is computed at each vertex trajectory based on its Euclidian distances from all other trajectories and the first cluster center is chosen at the trajectory with the highest density value. To get the next cluster center, the density function for all the trajectories are modified by subtracting a value inversely proportional to the distance of the trajectories to the found cluster centre. After modifying the density function, the next cluster center is obtained at the point having the largest density value. This subtractive process continues until the required numbers of cluster centers are obtained. The steps for this method are as follows:

1. Given the number of clusters K, density function at every vertex trajectory $t_i, i = 1, \ldots, V$ is calculated using

$$M_i = \sum_{j=1}^{V} \exp\left(-\alpha \left\|\mathbf{t_i} - \mathbf{t_j}\right\|^2\right), i = 1, 2, \ldots, V \tag{5}$$

where α is a positive constant and $\left\|\mathbf{t_i} - \mathbf{t_j}\right\|^2$ is the square of the distance between the vertex trajectories $\mathbf{t_i}$ and $\mathbf{t_j}$. Since the density function calculated for each trajectory is a function of its distances from all other trajectories, a vertex with many neighborhood vertices which are moving coherently across whole animation frames will have high density value. The constant α is actually a radius of influence of the neighborhood vertex trajectories on the density value.

2. Initialize k=1

3. Select the cluster centre trajectory $\mathbf{t_{c_k}}$ as the trajectory having the largest density value

$$M_{C_k} = \underset{i}{Max}\left(M_i\right), i = 1, \ldots, V. \tag{6}$$

4. Calculate the density measure of each vertex trajectory $\mathbf{t_i}$ using the equation

$$M_i = M_i - M_{C_k} \exp\left(-\beta \left\|\mathbf{t_i} - \mathbf{t_{c_k}}\right\|^2\right) \tag{7}$$

where β is a positive constant, $\mathbf{t_{c_k}}$ is the kth cluster center trajectory and M_{C_k} is its density value. Because of this modification, the density values of the vertex trajectories near the first cluster center will be scaled down as per the factor β so that they will not be selected as the next cluster center during next iteration.

5. After modifying the density function, increment k by 1 and continue this subtractive process to find the next cluster centres by iterating steps (3) and (4) until the required numbers of cluster centres are obtained.

The cluster centres finally obtained at step (4) above will be used by the K-means clustering algorithm as the initial cluster centres to get the optimum clusters of vertex trajectories and the cluster index vector. After clustering of vertex trajectories, the PCA is applied on each of these clusters to select the required number of eigen trajectories (vectors) and the corresponding PCA weights per cluster. Finally, all the PCA eigen vectors, PCA weights, mean vectors and the cluster index vector are quantized to get the compressed form of the input animation geometry data. The block diagram of encoder and decoder for the proposed SC-CPCA method is shown in figure 1.

5. EXPERIMENTAL RESULTS

5.1 Details of Animation Sequence

To study the performance of the proposed SC-CPCA method, some standard test animation sequences have been considered with geometry details shown in Table 1. All these data sets are of soft-body animation sequences whose faces and connectivity are constant over time.

Table 1: The animation sequences used with details

Name	**Vertices (V)**	**Triangles (T)**	**Frames (F)**
Face	757	1468	950
Cow	2904	5804	204
Dolphin	6179	12337	101
Chicken	3030	5664	400
Dance	7061	14118	201

5.2 Performance Metrics

The compression result of the proposed method is measured by some of the performance metrics used by most of the researchers in the literature. The measures are *compression ratio* (CR), *distortion factor* (D_a) and the *bits per vertex frame* ($bpvf$).

- *Compression Ratio* (CR): The compression ratio for geometry data is determined by the ratio of number of elements in the original animation matrix to the total number of elements in the compressed data. For the above K-mean clustering based PCA algorithm, the compressed data is consisted of principal eigen vectors, PCA coefficients and mean vectors of all the cluster matrix and the cluster index vector. So, the calculation of CR is as follows:

$$CR = \frac{3VF}{lE_k + lc_k + lm_k + Ix} \tag{8}$$

  where lE_k, lc_k and lm_k is the total size of eigen vectors matrix, PCA coefficients matrix and mean vectors over all clusters respectively and Ix is the size of cluster index.

- *Distortion factor* (D_a): To check the distortion in the reconstructed animation with respect to original one, we have used the distortion measure D_a also known as KG-error metric after Karni and Gottsman [5] and defined by

$$D_a = 100 \frac{\left\| \mathbf{B} - \hat{\mathbf{B}} \right\|_F}{\|\mathbf{B_{ms}}\|_F} \tag{9}$$

  where $\|.\|_F$ represents the 'Frobenius norm' or 'entry-wise Euclidean norm' given by the square root of the sum of the squares of the elements of the matrix.

- *Bits per vertex frame* ($bpvf$): It signifies the number of bits per vertex per frame and is a unit for bandwidth usage measurements. The calculation of $bpvf$ for the K-mean clustering based PCA algorithm is given by

$$bpvf = \frac{q_v lE_k + q_c lc_k + q_m lm_k + 6V + 5V}{VF} \tag{10}$$

  where q_v,q_c and q_m are the number of quantization bits used to represent the PCA eigen vectors, coefficients and the mean vector respectively. The term $5V$ represents the bits required to encode the cluster index with 5 bits as mentioned in [10] and $6V$ in the numerator corresponds to the number of bits required to encode the connectivity information as per Edgebreaker [9] algorithm for connectivity compression.

5.3 Results

In the first set of experiments, we have tested the CPCA algorithm on different animation sequences to get the compression results based on different performance metrics. Table 2 shows the result of applying this method on the 'Chicken' sequence. From the results it has been observed that because of random cluster centres initialization, each iteration of CPCA algorithm gives different values for D_a, CR and $bpvf$.

Table 2: Results using CPCA method (random initialization) on "Chicken" sequence

Selection of eigen vectors per cluster is based on total energy threshold of 0.999					
Run No.	No. of clusters (K)	Total Eigen vectors	CR	D_a	$bpvf$
1	5	64	28.838	0.187	1.652
2	5	61	30.399	0.188	1.566
3	5	59	30.238	0.159	1.575
4	5	64	28.838	0.187	1.652
5	5	58	30.969	0.173	1.537

Selection of fixed no. of eigen vectors per cluster (c=12)					
Run No.	No. of clusters (K)	Total Eigen vectors	CR	D_a	$bpvf$
1	5	60	30.974	0.261	1.537
2	5	60	30.974	0.204	1.537
3	5	60	30.974	0.364	1.537
4	5	60	30.974	0.288	1.537
5	5	60	30.974	0.288	1.537

In the next set of experiments, we have applied the proposed SC-CPCA method of different animation sequences. Figure 2 shows the clustering results after application of SC-CPCA method on 'Cow', 'Chicken', 'Dolphin' and 'Dance' animation sequence for a particular frame. The black stars in each figure indicate the cluster centers for each cluster.

The performance metrics obtained using the proposed SC-CPCA method for different number of clusters (K) and eigenvectors per cluster (c) are listed in Table 3. In this experiment, we have taken α=10, β=5, and a uniform quantization factor of q=16 bits to quantize the eigen vectors, PCA coefficients and the mean vectors. We have also shown

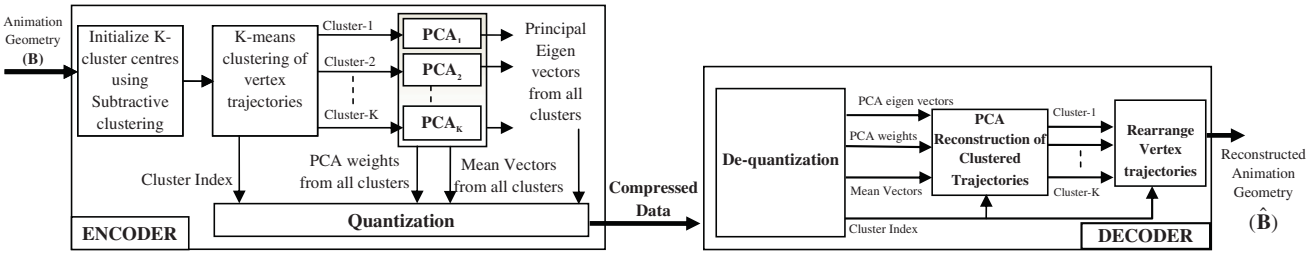
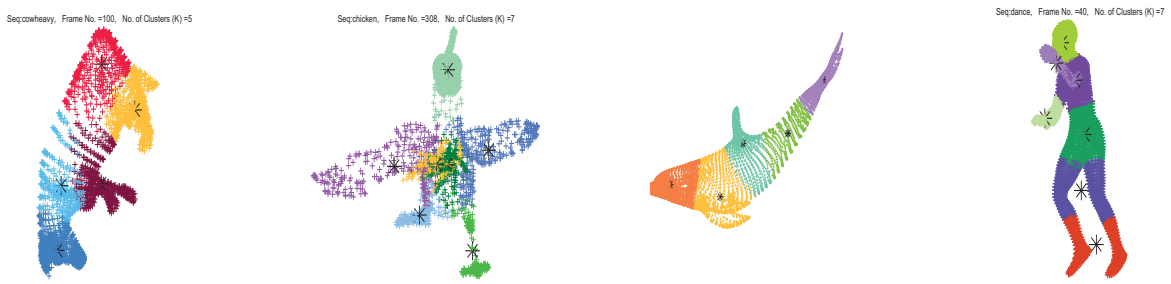


Figure 1: Encoder and Decoder block diagram of proposed SC-CPCA method



(a) 'Cow' with K=5 (b) 'Chicken' with K=7 (c) 'Dolphin' with K=5 (d) 'Dance' with K=7

Figure 2: A frame from different animation sequences showing the results of clustering algorithm

the results obtained using CPCA method in the same for comparison. From the results, it has been observed that for fixed number of eigen vectors (c) per cluster, the proposed SC-CPCA method gives better D_a for same CR and $bpvf$ compared to the CPCA method for the 'Chicken', 'Cow' and 'Face' sequence whereas for the 'Dolphin' and 'Dance' animation sequence it gives almost similar results as CPCA. The major drawback of the CPCA method is that it does not give optimal clustering result because of random initialization of cluster centres. Figure 3 shows one original and one reconstructed frame each from the 'Cow' and 'Chicken' animation sequences that were compressed using the proposed method for K=5 clusters and c=10 per cluster. The decompressed results after compression ratio of 27.09 in the case of 'Cow' and 36.61 in the case of 'Chicken' sequence show negligible visual artifacts.

6. CONCLUSIONS

This paper proposed to compress the geometry component of an animation sequence in a better way by using a subtractive clustering based clustered PCA (SC-CPCA) method. It gives a stable initialization of cluster centers for K-means algorithm used in CPCA algorithm instead of random initialization. This method gives stable results in terms of CR, D_a and $bpvf$ in each run of the algorithm. The proposed SC-CPCA method has been tested on some standard animation test sequences and the experimental results show better performance compared to the CPCA based geometry compression techniques. Our future work is directed towards testing other stables methods of cluster centers initialization for K-means algorithm as found in literatures and developing an efficient adaptive non-uniform quantization scheme for the PCA basis vectors and transformed coefficients of each cluster along with a suitable coding/decoding method for these quantized values to get better coding rate.

Table 3: Comparative results using proposed SC-CPCA method and the CPCA method

Proposed SC-CPCA method					CPCA method		
No. of clusters (K)	c per cluster	CR	D_a	$bpvf$	CR	D_a	$bpvf$
"Dolphin"							
2	10	25.08	0.029	1.86	25.08	0.029	1.86
3	10	24.01	0.015	1.95	24.01	0.015	1.95
4	10	23.03	0.009	2.03	23.03	0.010	2.03
"Chicken"							
5	5	67.11	**0.984**	0.70	67.11	1.180	0.70
5	10	36.61	**0.388**	1.30	36.61	0.591	1.30
5	20	19.17	**0.054**	2.49	19.17	0.091	2.49
"Cow"							
5	10	27.09	**0.748**	1.75	27.09	1.043	1.75
5	20	14.19	**0.291**	3.36	14.19	0.376	3.36
10	10	17.90	**0.468**	2.65	17.90	0.507	2.65
"Face"							
3	10	21.08	**0.190**	2.27	21.08	0.669	2.27
4	10	16.14	**0.151**	2.97	16.14	0.279	2.97
5	10	13.07	**0.153**	3.67	13.07	0.155	3.67
"Dance"							
5	10	38.42	**0.713**	1.22	38.42	1.009	1.22
5	15	26.41	0.220	1.79	26.41	0.220	1.79
10	10	29.57	0.332	1.60	29.57	0.332	1.60

7. ACKNOWLEDGMENTS

The authors would like to thank Zachi Karni, Rachida Amjoun, S. Ramanathan and L. Vasa for providing links for the animation sequences. The 'Face' animation sequence is

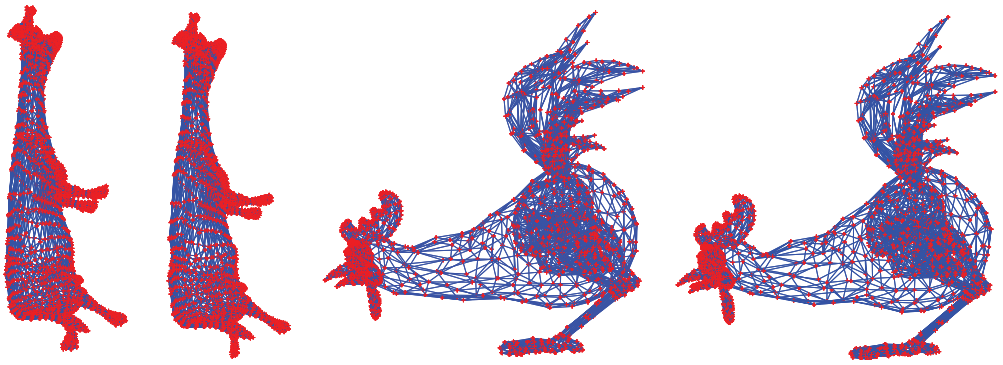


Figure 3: (a) Original and (b) Reconstructed Frame of 'Cow' sequence (c) Original and (d) Reconstructed Frame of 'Chicken' sequence

the property of 'Visage Technologies', the 'Cow' sequence is from the ETH Zurich and the 'Chicken' sequence is property of Microsoft Inc.

8. REFERENCES

[1] M. Alexa and W. Müller. Representing animations by principal components. *Wiley Inter Science Computer Graphics Forum*, 19(3):411–418, 2001.

[2] R. Amjoun and W. StraSSer. Efficient compression of 3d dynamic mesh sequences. *Journal of WSCG, placeEurope*, pages 99–106, 2007.

[3] S. L. Chiu. An efficient method for extracting fuzzy classification rules from high dimensional data. *Journal of Advanced Computational Intelligence*, 1(1):31–36, 1997.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

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

[6] S. S. Khan and A. Ahmad. Cluster center initialization algorithm for k-means clustering. *Pattern Recognition Letters*, 25(11):1293 – 1302, 2004.

[7] J. E. Lengyel. Compression of time-dependent geometry. In *In I3D Š99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 89–95. ACM, 1999.

[8] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28:129–137, 1982.

[9] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5:47–61, 1999.

[10] M. Sattler, R. Sarlette, and R. Klein. Simple and efficient compression of animation sequences. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, July 2005.

[11] J. Shlens. A tutorial on principal component analysis. http://www.snl.salk.edu/ shlens/pub/notes/pca.pdf, December 2005.

[12] L. Vása and V. Skala. Cobra: Compression of the basis for pca represented animations. *Comput. Graph. Forum*, 28(6):1529–1540, 2009.

[13] J. Xu, B. Xu, W. Zhang, W. Zhang1, and J. Hou1. Stable initialization scheme for k -means clustering. *Wuhan University Journal of Natural Sciences*, 14(1):24–28, February 2009.