DE GRUYTER OPEN

JAISCR, 2013, Vol. 3, No. 3, pp. 277 – 289
DOI 10.2478/jaiscr-2014-0020

DIMENSIONALITY REDUCTION OF DYNAMIC MESH ANIMATIONS USING HO-SVD

Michał Romaszewski, Piotr Gawron, Sebastian Opozda
Institute of Theoretical and Applied Informatics, Polish Academy of Sciences

Abstract

This work presents an analysis of Higher Order Singular Value Decomposition (HO-SVD) applied to reduction of dimensionality of 3D mesh animations. Compression error is measured using three metrics (MSE, Hausdorff, MSDM). Results are compared with a method based on Principal Component Analysis (PCA) and presented on a set of animations with typical mesh deformations.

1 Introduction

The goal of this paper is to provide an analysis of Higher Order Singular Value Decomposition [1] (HO-SVD) applied to reduction of dimensionality of dynamic mesh animations. The intention is to employ tensor decomposition as an element of a compression algorithm. The paper includes an estimation of lossy reconstruction quality using three error metrics and a comparison with a method based on Principal Component Analysis (PCA). Results are presented using a diverse set of well-known mesh animations, representing several common cases of 3D shape deformation.

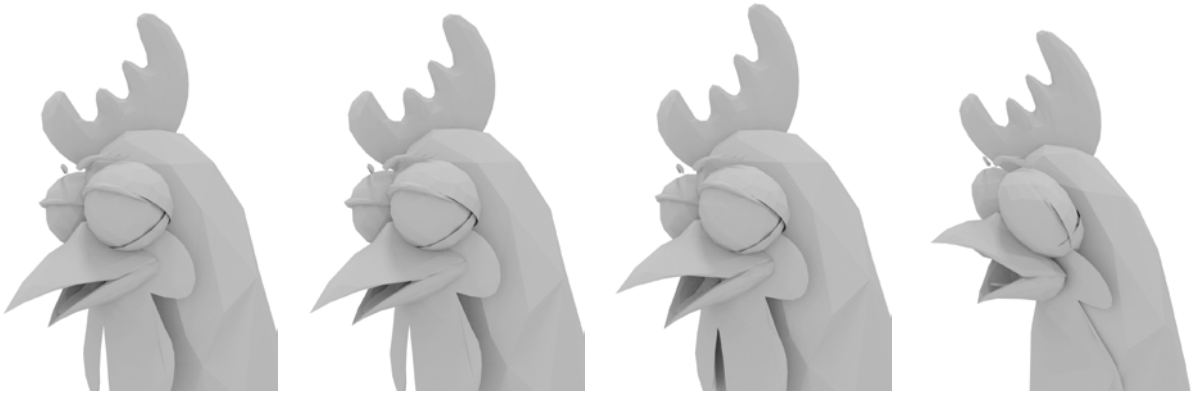
The paper is an extension of [2], with corrected equations, detailed description of algorithms and improved presentation of methods.

A compression algorithm usually consist of elements including compensation of motion (like in [3]), reduction of dimensionality and entropy encoding [4]. In this work we will concentrate on HO-SVD-based dimensionality reduction with only a simplified approach to frame aligning.

HO-SVD is a multi-linear generalization of Singular Value Decomposition. It has been shown (e.g. in [5]) that HO-SVD is an efficient method for dimensionality reduction of data represented as tensors, also called N-way arrays. Consecutive frames of a 3D animation can naturally be represented as a 3-mode tensor (a data cube), by stacking arrays of their vertices.

When using PCA-based compression, dimensionality reduction is often applied to animation frames (e.g. [6], [7]), reducing their number to a sequence of significant key-frames. On the contrary, HO-SVD allows for multidimensional reduction of the data tensor. In our experiment we truncated the number of components obtained through tensor decomposition, associated with mesh vertices and animation frames. Proportion of reduced components was found using a simple heuristic procedure. Reduction of components associated with 3D coordinates of vertices is not advised since it results in a significant loss of information and low quality of reconstructed data. For estimation of reconstruction quality we used three metrics. The Mean Squared Error (MSE) and the Hausdorff distance are both widely used for measuring 3D mesh distortions. Additionally, we decided to include a perceptual method, the Mesh Structural Distortion Measure (MSDM), since according to [8], it correlates well with human perception of errors in 3D data. An example of a distortion resulting from a lossy reconstruction of an animation using HO-SVD is presented in Fig.1.

The article is organised as follows. In the two following subsections, the related work and HO-SVD decomposition are presented. Definitions and



(a) (b) (c) (d)

Figure 1. A fragment of a reconstructed animation sequence for *Chicken* animation. Panel (a) presents an original model, in further panels the data tensor is compressed to (b): 5.1%, (c): 2.1%, and (d): 1.1% of its original size.

methodology of our experiments are presented in Section 2. Obtained results can be found in Section 3, while their summary along with our comments are presented in Section 4.

1.1 Related work

Due to their amount, data generated by using 3D scanners or animation software require effective compression methods for their storage, transmission, and processing. Particularly, compression of dynamic mesh animations is a subject to intensive research. A dimensionality reduction for 3D animations using PCA was introduced in [6] and refined in [7] where authors performed motion clustering on an animation and applied PCA to its subsegments. PCA-based compression is presented in [9] and [10]. Methods employing mesh connectivity are presented in [3] and [11]. Frame-based Animated Mesh Compression was also promoted within the MPEG-4 standard and is described in [12].

Higher Order Singular Value Decomposition (HO-SVD) may be treated as a natural extension of PCA for high-dimensional data. A survey of tensor properties as well as the description of higher-order tensor decomposition is provided in [13].

Tensor decomposition was successfully applied to compression and classification of images [14], face recognition [15] or watermarking of videos [16]. In [17] HO-SVD was applied to Level-of-Detail reduction in animation of human crowds. In [18] authors presented the decomposition of a motion tensor and applied it for animation dimensionality reduction, denoising and gap filling. In [19], an approach based on tensor decomposition and scalable hierarchical volume representation of spatial data is used for fast 3D visualization.

1.2 Higher Order Singular Value Decomposition

Higher Order Singular Value Decomposition, also called Tucker decomposition, is a generalisation of SVD from matrices to tensors (N-way arrays). In this section we recall basic facts about tensors and HO-SVD. We follow conventions presented in [13].

To describe this decomposition, first we will recall basic notions regarding operations on tensors. Let a tensor

$$T = \{t_{i_1,i_2,\ldots,i_n}\}_{i_1,i_2,\ldots,i_n=0}^{I_1-1,I_2-1,\ldots,I_N-1} \in \Re^{I_1,I_2,\ldots,I_N} \quad (1)$$

be given - we say that this tensor has n modes. Each of the indices corresponds to one of the modes *i.e.* i_l to mode l.

By *multiplication* of tensor T by matrix $U = \{u_{i_l d}\}_{i_l,d=0}^{I_l-1,D} \in \Re^{I_l,D}$ in mode l we define tensor $T' \in \Re^{I_1,\ldots,I_{l-1},D,I_{l+1},\ldots,I_N}$, such that

$$T' = (T \times_l U)_{i_1\ldots i_{l-1} d\, i_{l+1}\ldots i_N} = \sum_{i_l=0}^{I_l-1} t_{i_1 i_2\ldots i_l\ldots i_N} u_{i_l d}. \quad (2)$$

By *unfolding* tensor T in mode l we define matrix $T_{(l)}$ such that

$$(T_{(l)})_{i,j} = t_{i_1 \ldots i_{l-1} j i_{l+1} \ldots i_N}, \tag{3}$$

where

$$i = 1 + \sum_{\substack{k=1 \\ k \neq l}}^{N} J_k \text{ and } J_k = \prod_{\substack{m=1 \\ m \neq l}}^{k-1} I_m.$$

Given tensor T, defined as in Eq. (1), a new *sub-tensor* $T_{i_n=\alpha}$ can be created according to the equation with the following elements:

$$\mathrm{T}_{i_l=\alpha} = \{t_{i_1 i_2 \ldots i_{l-1} i_{l+1} \ldots i_n}\}_{i_1=0, i_2=0, \ldots, i_l=\alpha, \ldots, i_n=0}^{I_1-1, I_2-1, \ldots, \alpha, \ldots, I_N-1} \in \Re^{I_1, I_2, \ldots, 1, \ldots, I_N}. \tag{4}$$

The *scalar product* $\langle A, B \rangle$ of tensors $A, B \in \Re^{I_1, I_2, \ldots, I_N}$ is defined as

$$\langle A, B \rangle = \sum_{i_1=0}^{I_1-1} \sum_{i_2=0}^{I_2-1} \cdots \sum_{i_N=0}^{I_N-1} b_{i_1, i_2, \ldots, i_n} a_{i_1, i_2, \ldots, i_n}. \tag{5}$$

We say that if scalar product of tensors equals 0, then they are orthogonal.

The *Frobenius norm* of tensor T is given by

$$||T|| = \sqrt{\langle T, T \rangle}. \tag{6}$$

Given tensor T, in order to find its HO-SVD, in the form of the so called Tucker operator $CU^{(1)}, \ldots, U^{(N)}$, such that $C \in \Re^{I_1, \ldots, I_N}$ and $U^{(k)} \in \Re^{I_k \times I_k}$ are orthogonal matrices, Algorithm 1.2 can be used.

Algorithm 1: HO-SVD algorithm

Input: Data Tensor $\mathcal{T}$
Output: Tucker operator $[\![C; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]$
for $k \in \{1, \ldots, N\}$ **do**
  $\mathbf{U}^{(k)} =$ left singular vectors of $T_{(k)}$ in unfolding k;
end
$C = \mathcal{T} \times_1 \mathbf{U}^{(1)T} \times_2 \mathbf{U}^{(2)T} \ldots \times_N \mathbf{U}^{(N)T}$;
return $[\![C; \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}]\!]$;

Tensor C is called the core tensor and has the following useful properties.

– Reconstruction:

$$T = C \times_1 U^{(1)} \times_2 U^{(2)} \times_3 \ldots \times_N U^{(N)}, \tag{7}$$

where $U^{(i)}$ are orthogonal matrices;

– Orthogonality:

$$\langle C_{i_l=\alpha}, C_{i_l=\beta} \rangle = 0 \tag{8}$$

for all possible values of l, α and β, such that $\alpha \neq \beta$;

– Order of sub-tensor norms:

$$||C_{i_n=1}|| \leq ||C_{i_n=2}|| \leq \ldots \leq ||C_{i_n=I_n}|| \tag{9}$$

for all n.

Therefore, informally, one can say that larger magnitudes of a core tensor are denoted by low values of indices. This property is the basis for the development of compression algorithms based on HO-SVD.

Formally

$$\tilde{T} = \tilde{C} \times_1 \tilde{U}^{(1)} \times_2 \tilde{U}^{(2)} \times_3 \ldots \times_N \tilde{U}^{(N)}, \tag{10}$$

where

$$\tilde{C} = \{c_{i_1, i_2, \ldots, i_n}\}_{i_1, i_2, \ldots, i_n=0}^{R_1-1, R_2-1, \ldots, R_N-1} \in \Re^{R_1, R_2, \ldots, R_N} \tag{11}$$

is a truncated tensor in such a way that in each mode l indices span from 0 to $R_l - 1 \leq I_l - 1$ and $\tilde{U}^{(l)} \in \Re^{R_l \times I_l}$ matrices whose columns are orthonormal and rows form orthonormal basis in respective vector spaces. A visualization of 3-mode truncated tensor is provided in Fig. 2

Given $(R_l)_{l=1}^N$ one can form tensor $\tilde{T}$ that approximates tensor T in the sense of their euclidean distance $||\tilde{T} - T||$. This approximation can be exploited to form lossy compression algorithms of signals that are indexed by more than two indices. It should by noted that the choice of $(R_l)_{l=1}^N$ in a given application is non-obvious and depends on the properties of processed signals.

2 Method

Our experiments aim to assess the effectiveness of HO-SVD for reduction of dimensionality

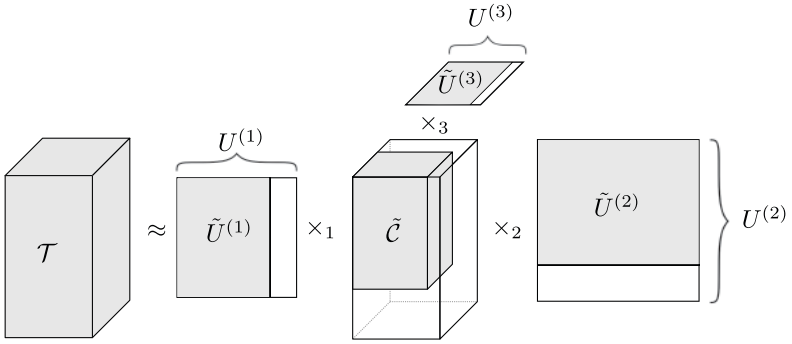


Figure 2. Truncated HO-SVD decomposition of tensor T. Its approximation, tensor $\tilde{T}$, can be reconstructed from a truncated tucker operator $\tilde{C}\tilde{U}^{(1)},\tilde{U}^{(2)},\tilde{U}^{(3)}$. The visualization is inspired by [13].

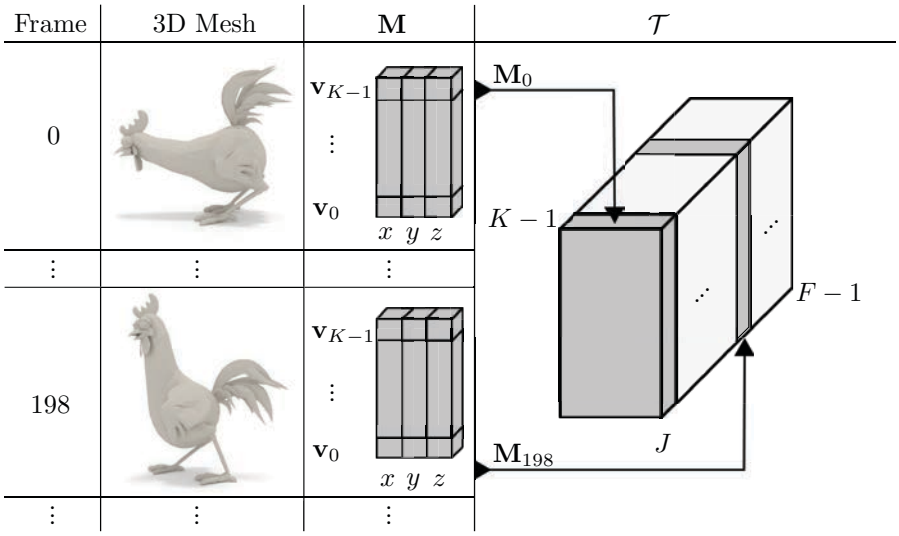


Figure 3. Visualization of data tensor T. It is formed by stacking vertices of meshes, corresponding to F animation frames, represented as $K \times J$ arrays.

of 3D animations. We will introduce a simple heuristic approach to choosing the proportion of preserved spatial and temporal components in order to maintain good quality of reconstructed data. We will present results using multiple error metrics and compare them to a method based on PCA.

2.1 Input data

Three-dimensional mesh will be treated as a $K \times J$ matrix M, with mesh vertices $v_i \in \Re^J, i \in \{0,\ldots,K-1\}$ as rows, together with a set of triangle faces G defined as three element tupples of vertex indices. We denote $J = 3$ as a number of spatial dimensions. An animation consists of F successive frames enumerated with k, each containing a mesh $M_k, k \in \{0,\ldots,F-1\}$, with the same topology, but different coordinates of vertices. Therefore, input data can form tensor $T = t_{i,j,k} \in \Re^{K\times J\times F}$ while meshes M_k, following the notation in [13], form frontal slices $T_{::i}$. Tensor visualization is presented in Fig. 3. A pair (T,G) contains all available information about the animation. We apply compression only to T, a set of faces G is used only for data visualization.

2.2 Dimensionality reduction testing procedure

Algorithm 2.2 presents stages of the procedure aimed at calculating the quality of mesh reconstruction.

Algorithm 2: A procedure for estimating the quality of HO-SVD compression for 3D animation. Similar procedure is performed for PCA.

Input: Data Tensor $\mathcal{T}$, Compression rate CR, Quality metric d
Output: Quality of $\mathcal{T}'$
`/* ` $\mathcal{X}$ `is a normalised tensor`
`/* ` $\Re$ `is a sequence of homography matrices`
$\mathcal{X}, \Re$ = Rigid Motion Estimation($\mathcal{T}$);
`/* ` $[\![C;\mathbf{U}^{(1)},\mathbf{U}^{(2)},\mathbf{U}^{(3)}]\!]$ `is the Tucker operator`
$[\![C;\mathbf{U}^{(1)},\mathbf{U}^{(2)},\mathbf{U}^{(3)}]\!]$ = HOSVD Decomposition(X);
`/* ` VTF `is a scalar of Vertex-to-Frame ratio`
VTF = Estimate VTF($[\![C;\mathbf{U}^{(1)},\mathbf{U}^{(2)},\mathbf{U}^{(3)}]\!]$, CR);
$[\![\tilde{C};\tilde{\mathbf{U}}^{(1)},\tilde{\mathbf{U}}^{(2)},\tilde{\mathbf{U}}^{(3)}]\!]$ = Truncate($([\![C;\mathbf{U}^{(1)},\mathbf{U}^{(2)},\mathbf{U}^{(3)}]\!]$)
$\tilde{\mathcal{T}}$ = Reconstruct($[\![\tilde{C};\tilde{\mathbf{U}}^{(1)},\tilde{\mathbf{U}}^{(2)},\tilde{\mathbf{U}}^{(3)}]\!]$, VTF, CR, $\Re$);
return d($\mathcal{T},\tilde{\mathcal{T}}$);

The algorithm consist of the following steps:

1 *Rigid motion estimation*:
The rigid motion of a mesh over the whole animation is estimated and subtracted from consecutive frames. Sequence $R = (R_i)_{i=0}^{F-1}$ is created where R_i is a matrix of an rigid transformation between frame i and 0. Consecutive frames are translated into the coordinates of the first frame, forming normalised data tensor X : $\forall_i X_{::i} = T_{::i}R_i^T$, used in the following steps.

2 *Tensor decomposition*:
Tensor X is decomposed using HO-SVD, so Tucker decomposition
$CU^{(1)},U^{(2)},U^{(3)}$ is obtained.

3 *HO-SVD compression parameter estimation*:
We estimate the Vertices-to-Frame ratio VTF between a number of spatial and temporal components of the decomposed tensor X, required to achieve the desired compression rate, by searching for the best set in the parameter space.

4 *HO-SVD compression*:
Compression is performed by truncating a number of spatial and temporal components of $CU^{(1)},U^{(2)},U^{(3)}$, forming truncated Tucker operator $\tilde{C}\tilde{U}^{(1)},\tilde{U}^{(2)},\tilde{U}^{(3)}$.

5 *Reconstruction*:
Tensor $\tilde{X}$ is reconstructed from $\tilde{C}\tilde{U}^{(1)},\tilde{U}^{(2)},\tilde{U}^{(3)}$. Then, frames of the animation are transformed to their original coordinates by applying a corresponding inverse transformation from R to each frame, forming tensor $\tilde{T} : \forall_i \tilde{T}_{::i} = \tilde{X}_{::i}(R_i^T)^{-1}$.

6 *Estimation of the reconstruction quality*:
The quality of reconstruction $(T,\tilde{T})$ is computed using three 3D quality metrics: Mean Squared Error, Hausdorff distance and Mesh Structural Distortion Measure.

Steps 1 to 4 correspond to the animation compression process. Step 5 represents data decompression. After step 4, an additional compression of floating-point data should be performed. We provide a detailed description of these steps in the following subsections.

2.3 Rigid motion estimation

The first step of the algorithm follows the idea from [6] and applies a simple rigid normalization of a dynamic mesh animation. If a set of

faces G of mesh M is constant through the animation, mesh state in frame i, can be described by the sum of changes applied to M in each frame: $M_i = \sum_{j=1}^{i}(M_j - M_{j-1}) = \sum_{j=1}^{i}\Delta M_j$. Assuming that animation is represented in homogeneous coordinates, the difference between two consecutive frames $\Delta M_j = D_j R_j^T$, where R_j is a rigid transformation between frames, and D_j corresponds to deformation of mesh vertices. Therefore $M_i = \sum_j D_j R_j^T$, where R_j is a rigid transformation between frames 0 and j.

The output of this step is sequence $R = (R_1, \ldots, R_F)$ of transformation matrices between frame 0 and all consecutive ones, as well as a new, transformed data tensor $X : \forall_i X_{::i} = T_{::i} R_i^T$.

2.4 Higher Order Singular Value Decomposition

For the purpose of a compression algorithm, data tensor X containing normalised animation frames is decomposed using HO-SVD. The resulting Tucker operator $CU^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}$ is passed to further steps of the algorithm.

2.5 Dimensionality reduction and reconstruction

Vertices of a 3D mesh form $K \times J$ matrix M, where $J = 3$. The number of memory units required to store or transmit an animation of F frames, not considering a set of faces G, may be expressed as

$$S = K \times F \times J \times d_s,$$

where d_s is the size of a single floating-point variable, e.g. $d_s = 4$ bytes. HO-SVD allows to reduce the amount of memory required to store an animation, by decomposing data tensor T and storing only the truncated Tucker operator $\tilde{C}\tilde{U}^{(1)}, \tilde{U}^{(2)}, \tilde{U}^{(3)}$. Theoretically there are three compression parameters, corresponding to J dimensions of T. However, since the reduction of mode-2 components heavily impacts the quality of the reconstructed mesh, we will only consider the reduction of K mode-1 and F mode-3 components. The amount of data required to store the Tucker operator $CU^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}$ equals

$$S^{(\text{hosvd})} = (v \times K + J^2 + f \times F + v \times J \times f) \times d_s,$$

where v corresponds to the number of mode-1 and f to mode-3 components kept. Therefore

$$CR^{(\text{hosvd})} = \frac{S^{(hosvd)}}{S} =$$

$$= \frac{v \times K + J^2 + f \times F + v \times J \times f}{K \times F \times J}. \tag{12}$$

For visualization of results, space savings (SS) will be used in place of compression rate, defined as

$$SS = (1 - CR)100\%, \tag{13}$$

so $SS = 99\%$ denotes only 1% of data remaining after compression.

In addition, we need to store a set of transformation matrices R, obtained during the first step of the algorithm. Its size is $S^{(\text{R})} = 12 \times F$, and it will be included in our results.

2.6 HO-SVD compression parameter estimation

Application of HO-SVD for 3D mesh compression requires a strategy of choosing the proportion of preserved components for each mode, resulting in the required CR. Mode-1 components correspond to spatial information (vertices) and mode-3 to temporal information (frames). If we denote the number of preserved mode-1 components as v and the number of mode-3 components as f, $\frac{v}{f}$ is the Vertices-To-Frames ratio (VTF).

We estimate VTF by searching for a pair $(v_{\min}, f_{\min})$ that gives the lowest reconstruction error among candidates obtained by using Algorithm 2.6. We simplify this time-consuming task, thanks to our observation that for a list of parameters obtained from Algorithm 2.6, the distortion of reconstruction performed by truncating the Tucker operator $CU^{(1)}, U^{(2)}, U^{(3)}$ can usually be approximated using an unimodal function. Therefore, a minimum can be estimated with a simple iterative procedure presented as Algorithm 2.6.

Algorithm 3: A search for a sequence of (v,f) parameters, that allow to obtain the truncated Tucker operator $\tilde{C}\tilde{U}^{(1)},\tilde{U}^{(2)},\tilde{U}^{(3)}$ with the desired compression rate *CR*. K is the number of mesh vertices, F is the number of animation frames, and λ denotes the desired *CR*. The relation between (v,f) parameters and *CR* is described by Eq. (12) and will be denoted as Ψ.

```
/* δ is a tolerance margin.                                          */
Input: K, F, λ, δ
Output: a sequence of (v,f) pairs
𝔓=all pairs (v,f) such that v ∈ {1,...,K},f ∈ {1,...,F} and Ψ(v,f) − λ ≤ δ;
/* List is an empty sequence of pairs.                               */
if V > F then
    for v ∈ 𝔓 do
        List = create sequence of pairs (v,f) such that |Ψ(v,f) − λ| is minimal
        amongst all values of f;
    end
else
    for f ∈ 𝔓 do
        List = create sequence of pairs (v,f) such that |Ψ(v,f) − λ| is minimal
        amongst all values of v;
    end
end
if V > F then
    Sort(List) by v
else
    Sort(List) by f
end
return List
```

Algorithm 4: Estimation of the best pair of parameters (v,f), that allows reconstruction of T from $\tilde{C}\tilde{U}^{(1)},\tilde{U}^{(2)},\tilde{U}^{(3)}$ with a minimal error.

```
Function FindMinimum(List)
    Input: List: a sequence of (v,f) pairs
    Output: Index of the best element i_min
    /* s is a number of samples.                                     */
    indices = s indices of a uniformly sampled List;
    /* Errors is an empty sequence.                                  */
    for i ∈ indices do
        /* 𝒯 is a data tensor                                        */
        /* [[C; U^(1), U^(2), U^(3)]] is a tucker operator           */
        /* ℜ is a sequence of homography matrices                    */
        [[C̃; Ũ^(1), Ũ^(2), Ũ^(3)]] =Truncate([[C; U^(1), U^(2), U^(3)]])
        𝒯̃ = Reconstruct([[C̃; Ũ^(1), Ũ^(2), Ũ^(3)]], VTF, CR, ℜ);
        Append (d(𝒯, 𝒯̃)) to Errors;
    end
    i_min=ArgMin(Errors);
    /* I is a limit for iterations.                                  */
    if recursion depth==I then
        return i_min
    else
        return FindMinimum (List[indices[i_min − 1]:indices[i_min + 1]])
    end
```

2.7 Reconstruction quality estimation

Reconstruction errors were measured by using two standard metrics:

- Mean Squared Error: $_{\mathrm{MSE}}(\mathbf{v},\mathbf{v}') = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{v}'-\mathbf{v})^2$, where $\mathbf{v}$ is the original data vector and $\mathbf{v}'$ is its reconstruction.
- Hausdorff distance:

$$_{\mathrm{H}}(A,B) = \max\{\sup_{x\in A}\inf_{y\in B}{}_{\mathrm{e}}(x,y), \sup_{y\in A}\inf_{x\in B}{}_{\mathrm{e}}(x,y)\},$$

  where A is the original, B – a reconstructed data set and $_{\mathrm{e}}$ denotes the euclidean distance.

Since these metrics may not correspond well with human perception of quality for 3D objects, an additional, perceptual metric called Mesh Structural Distortion Measure (MSDM) described in [8] was applied. This metric compares two shapes based on differences of curvature statistics (mean, variance, covariance) over their corresponding local windows. A global measure between the two meshes is then defined by the Minkowski sum of the distances over local windows. Since the metric compares static meshes, the final result for dynamic sequence is averaged between animation frames.

2.8 Comparison of HO-SVD and PCA application for 3D animation compression

In order to verify the performance of HO-SVD, we compared it with a simple method of 3D animation dimensionality reduction. Following the idea from [6] we performed experiments using PCA.

Principal Component Analysis [20] may be defined as follows.

Let $X = [\mathbf{x}_1, \mathbf{x}_2 \ldots, \mathbf{x}_L]$ be a data matrix, where $\mathbf{x}_i \in \Re^p$ are data vectors with zero empirical mean. The associated covariance matrix is given by $E = XX^T$. By performing eigenvalue decomposition of $E = ODO^T$ such that eigenvalues $\lambda_i, i = 1,..,p$ of D are ordered in a descending order $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p > 0$, one obtains the sequence of principal components $[\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_p]$ which are columns of O. One can form a feature vector $\mathbf{y}$ of dimension $p' \leq p$ by calculating $\mathbf{y} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{p'}]^T\mathbf{x}$.

In order to apply PCA, tensor $T = t_{i,j,k} \in \Re^{F\times J\times K}$ must be unfolded according to Eq. (3). Therefore mode-1 unfolding is performed so the data is flattened row by row to form matrix $X_T \in \Re^{F\times JK}$.

Compression is performed by storing only a limited number of principal components of E. When reconstructing matrix X, the dimension of the desired feature vector p' equals the number of principal components $\mathbf{y} = [\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{p'}]^T\mathbf{x}$ used for its calculation and is the only parameter. The ratio of reduction depends on number f' of the key-frames left. The compression rate for an animation of a 3D mesh using PCA can be expressed as

$$CR^{(pca)} = \frac{(V\times J + F)\times f' \times d_s}{S}$$

3 Results

Presentation of results is performed by using a set of well-known 3D animations, summarised in Table 1. *Chicken* and *Gallop* are artificial sequences of moving animal models. *Collapse* uses the same model as *Gallop* but the applied deformation is an elastic, non-rigid transformation. *Samba*, *Jumping*, *Bouncing* are motion capture animations of moving and dancing humans.

The impact of proportion of mode-1 and mode-3 components (VTF) on reconstruction quality is presented in Fig. 4. Panel (a) shows how the reconstruction error drops sharply as the number of components grows. Panel (b) presents VTF ratio as the rate of data reduction grows.

Observable deformations for artificial animated meshes (*Chicken*, *Gallop*) are almost unnoticeable for $SS \sim 90\%$ and only minor distortion is present for $SS \sim 95\%$. For motion capture sequences (*Samba*, *Jumping*, *Bouncing*), major deformations are present for $SS \sim 95\%$, and only minor ones for $SS \sim 85\%$, with unnoticeable distortions for $SS \sim 70\%$. Reconstruction errors are higher for the *Collapse* mesh, as its animation is hard to describe using rigid transformations. Major deformations are observable for $SS \sim 90\%$, minor ones are present up to $SS \sim 70\%$, and no noticeable distortions for $SS \sim 50\%$ were present. Frames from reconstructed animations are presented in Fig. 5 (*Chicken*), Fig. 6 (*Collapse*) and 7 (*Samba*).

A comparison of the reconstruction error occurring when using HO-SVD and PCA is presented

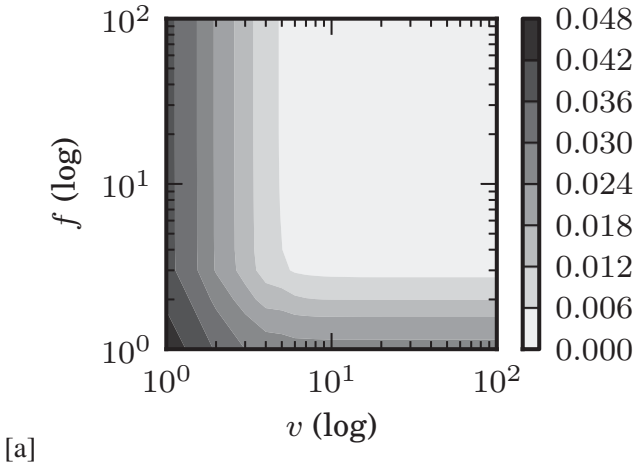
Table 1. An overview of animations used for visualization of results.

a *Chicken* animation was published by Jed Lengyel (http://jedwork.com/jed)

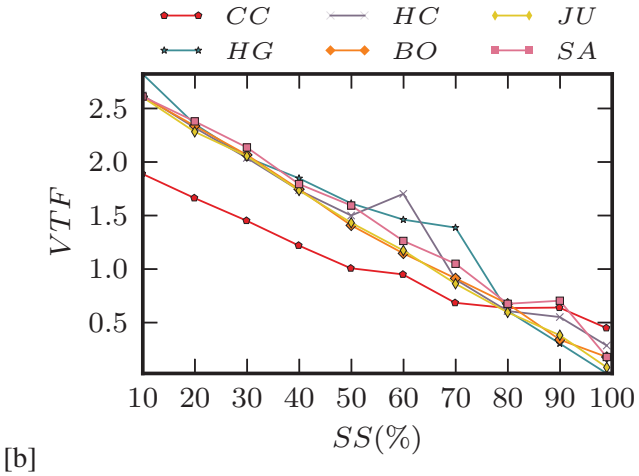
b *Gallop* and *Collapse* animations, described in [21], were obtained from the website of Doug L. James and Christopher D. Twigg (http://graphics.cs.cmu.edu/projects/sma).

c Motion capture sequences were obtained from the website of Daniel Vlasic (http://people.csail.mit.edu/drdaniel/mesh$_{a}$*nimation*).

Name	Referenced as	Vertices	Frames	Description
Chicken Crossinga	*Chicken*	3030	400	animation
Horse Gallopb	*Gallop*	8431	48	animation
Horse Collapse	*Collapse*	8431	48	animation
Sambac	*Samba*	9971	174	motion capture sequence
Jumping	*Jumping*	10002	149	motion capture sequence
Bouncing	*Bouncing*	10002	174	motion capture sequence

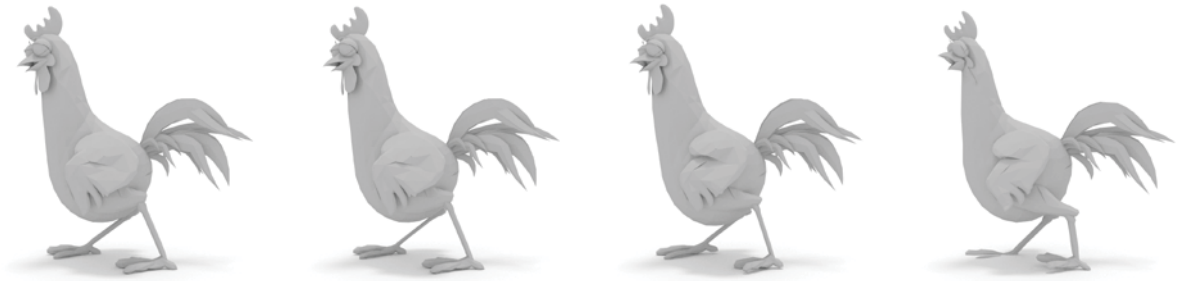


[a]



[b]

Figure 4. An impact of HO-SVD parameter selection on MSE reconstruction for the *Chicken* animation. Panel (a) presents the reconstruction error as a function of the number of mode-1 (v) and mode-3 (f) components. Note that the distortion drops sharply with only a few first components. Panel (b) presents Vertices-to-Frame ratio as a function of *SS*

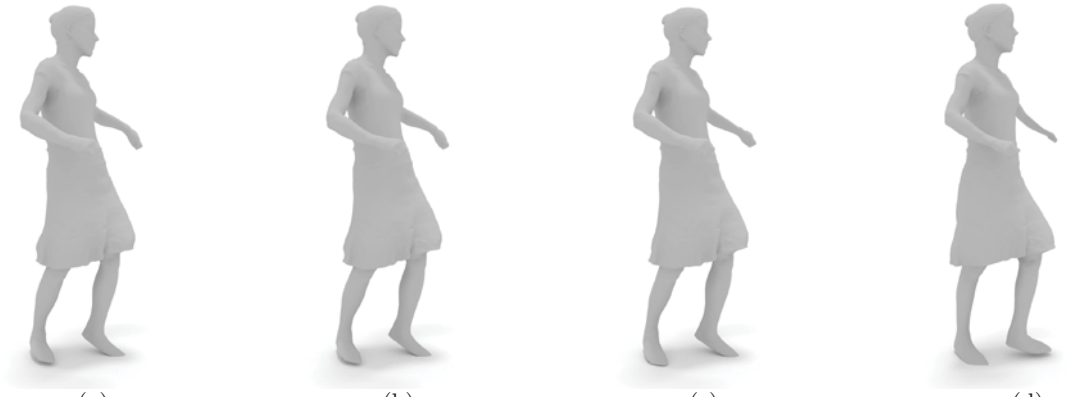


(a) (b) (c) (d)

Figure 5. Visualization of a reconstructed model for *Chicken*. (a): original, (b): SS=94.8%, (c): SS=97.8%, (d): SS=98.8%.

(a) (b) (c) (d)

Figure 6. Visualization of a reconstructed model for *Collapse*. (a): original, (b): SS=69.9%, (c): SS=84.9%, (d): SS=97.9%.



(a) (b) (c) (d)

Figure 7. Visualization of a reconstructed model for *Samba*. (a): original, (b): SS=89.9%, (c): SS=94.9%, (b): SS=97.9%.

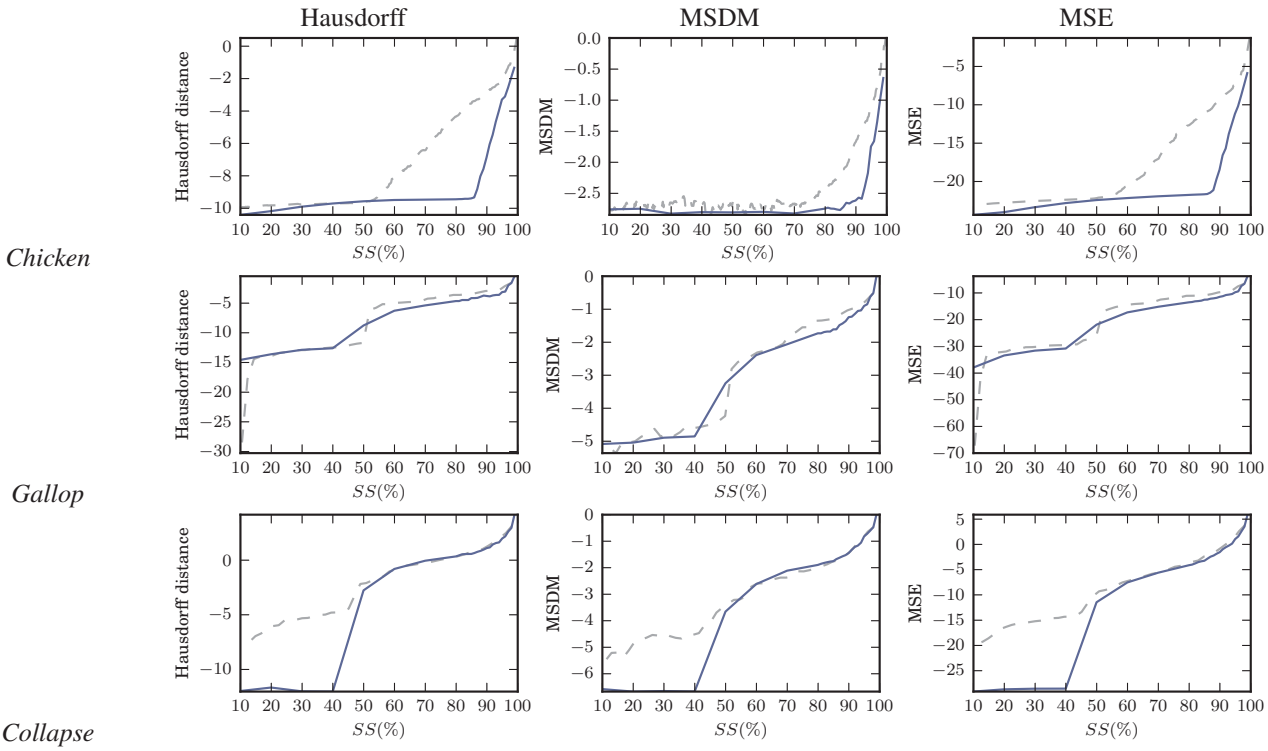


Figure 8. A comparison of HO-SVD (solid line) and PCA (dashed line) reconstruction errors for artificial animations. Distortion is presented in the logarithmic scale as a function of *SS*. Lower values of distortion indicate higher reconstruction quality.

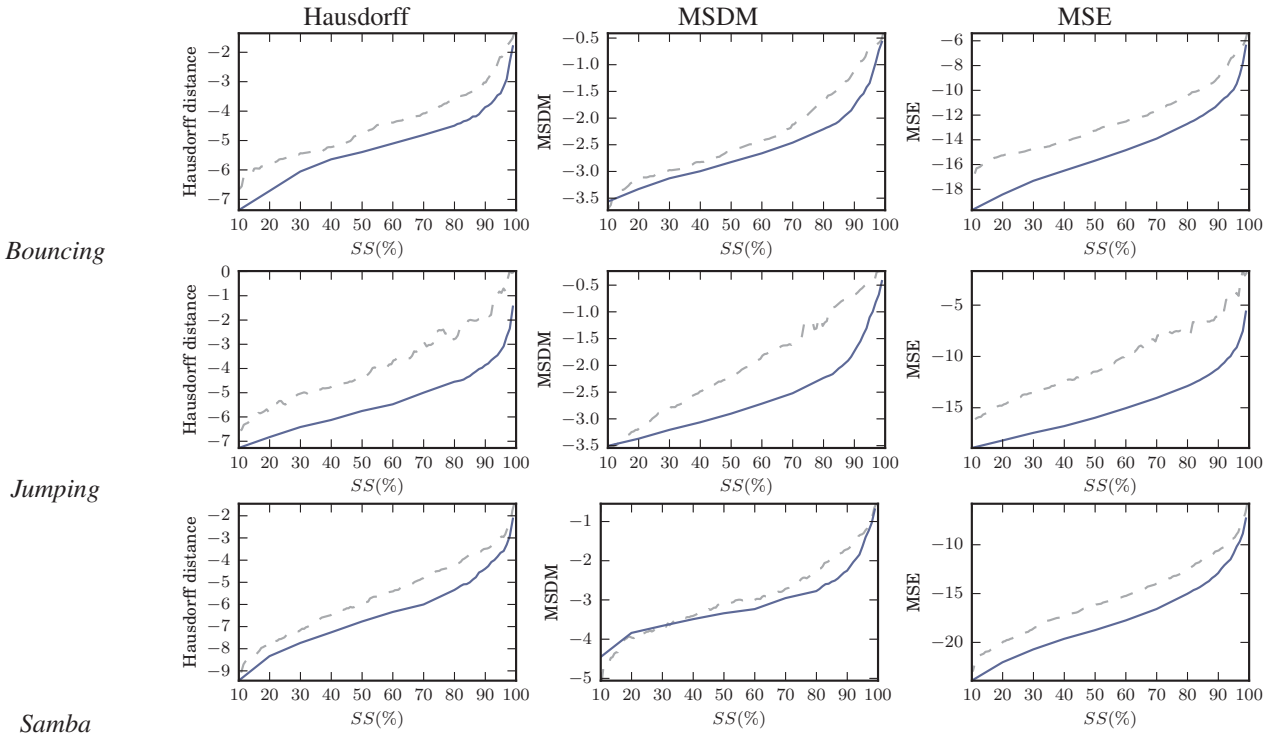


Figure 9. A comparison of HO-SVD (solid line) and PCA (dashed line) reconstruction errors for artificial animations. Distortion is presented in the logarithmic scale as a function of *SS*. Lower values of distortion indicate higher reconstruction quality.

in Fig. 8 for *Chicken*, *Gallop*, *Collapse* and Fig. 9 for *Samba*, *Jumping*, *Bouncing*. HO-SVD reduction gives better result for a majority of animations. Its advantage is visible especially for motion-capture sequences. Results for *Collapse* show that both methods have problems with describing non-rigid transformations, and their results are similar for high values of compression ratio with HO-SVD introducing lower distortion for low values.

4 Conclusions

Our experiments show that HO-SVD allows to achieve good reconstruction quality when applied to reduction of dimensionality of 3D animations, and usually outperforms the application of PCA. For most of the animated models and motion-capture sequences, $SS \sim 90\%$ produces a reconstruction very similar to the original.

The reconstruction error can be measured by using objective metrics, which allows reliable control over compression parameters. Parameters related to the proportion of preserved components in each mode, after performing data decomposition, can be estimated using a simple heuristic approach.

Acknowledgements

This work has been partially supported by the National Science Centre projects: M. Romaszewski by NN516405137 (decision 2011/03/D/ST6/03753), P. Gawron by NN516481840 (decision 4818/B/T02/2011/40), and S. Opozda by NN516482340 (decision 4823/B/T02/2011/40).

References

[1] De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM journal on Matrix Analysis and Applications **21**(4) (2000) 1253–1278

[2] Romaszewski, M., Gawron, P., Opozda, S.: Dimensionality reduction of dynamic animations using ho-svd. In Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L., Zurada, J., eds.: Artificial Intelligence and Soft Computing. Volume 8467 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2014)

[3] Ibarria, L., Rossignac, J.: Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association (2003) 126–135

[4] Sayood, K.: Introduction to data compression. Access Online via Elsevier (2012)

[5] Inoue, K., Urahama, K.: DSVD: a tensor-based image compression and recognition method. In: Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on. (2005) 6308–6311 Vol. 6

[6] Alexa, M., Müller, W.: Representing Animations by Principal Components. Computer Graphics Forum **19**(3) (2000) 411–418

[7] Sattler, M., Sarlette, R., Klein, R.: Simple and efficient compression of animation sequences. In: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation. SCA '05, New York, NY, USA, ACM (2005) 209–217

[8] Lavoué, G., Drelie Gelasca, E., Dupont, F., Baskurt, A., Ebrahimi, T.: Perceptually driven 3D distance metrics with application to watermarking. In: SPIE Applications of Digital Image Processing XXIX. (August 2006)

[9] Karni, Z., Gotsman, C.: Compression of soft-body animation sequences. Computers & Graphics **28**(1) (2004) 25–34

[10] Váša, L., Skala, V.: Cobra: Compression of the basis for pca represented animations. In: Computer Graphics Forum. Volume 28., Wiley Online Library (2009) 1529–1540

[11] Váša, L., Skala, V.: Geometry-driven local neighbourhood based predictors for dynamic mesh compression. In: Computer Graphics Forum. Volume 29., Wiley Online Library (2010) 1921–1933

[12] Mamou, K., Zaharia, T., Preteux, F. In: FAMC: The MPEG-4 standard for Animated Mesh Compression. IEEE (Oct 2008) 2676–2679

[13] Kolda, T.G., Bader, B.W.: Tensor Decompositions and Applications. SIAM Review **51**(3) (September 2009) 455–500

[14] Shashua, A., Levin, A.: Linear image coding for regression and classification using the tensor-rank principle. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Volume 1., IEEE (2001) I–42

[15] Wang, H., Ahuja, N.: Facial expression decomposition. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on. (2003) 958–965 vol.2

[16] Abdallah, E.E., Hamza, A.B., Bhattacharya, P.: Mpeg video watermarking using tensor singular value decomposition. In: Image Analysis and Recognition. Springer (2007) 772–783

[17] Mukai, T., Kuriyama, S.: Multilinear Motion Synthesis with Level-of-Detail Controls. In: Computer Graphics and Applications, 2007. PG '07. 15th Pacific Conference on. (2007) 9–17

[18] Akhter, I., Simon, T., Khan, S., Matthews, I., Sheikh, Y.: Bilinear spatiotemporal basis models. ACM Transactions on Graphics **31**(2) (April 2012) 17:1–17:12

[19] Suter, S.K., Makhynia, M., Pajarola, R.: Tamresh–tensor approximation multiresolution hierarchy for interactive volume visualization. In: Computer Graphics Forum. Volume 32., Wiley Online Library (2013) 151–160

[20] Jolliffe, I.: Principal Component Analysis (2nd ed). Springer (2002)

[21] James, D.L., Twigg, C.D.: Skinning Mesh Animations. ACM Trans. Graph **24** (2005) 399–407