Multilinear Motion Synthesis with Level-of-Detail Controls

Tomohiko Mukai and Shigeru Kuriyama
Toyohashi University of Technology
1-1 Hibarigaoka, Tenpaku-cho, Toyohashi, Aichi 441-8580, Japan
{mukai, kuriyama}@ics.tut.ac.jp

Abstract

Interactive animation systems often use a level-of-detail (LOD) control to reduce the computational cost by eliminating unperceivable details of the scene. Most methods employ a multiresolutional representation of animation and geometrical data, and adaptively change the accuracy level according to the importance of each character. Multilinear analysis provides the efficient representation of multidimensional and multimodal data, including human motion data, based on statistical data correlations. This paper proposes a LOD control method of motion synthesis with a multilinear model. Our method first extracts a small number of principal components of motion samples by analyzing three-mode correlations among joints, time, and samples using high-order singular value decomposition. A new motion is synthesized by interpolating the reduced components using geostatistics, where the prediction accuracy of the resulting motion is controlled by adaptively decreasing the data dimensionality. We introduce a hybrid algorithm to optimize the reduction size and computational time according to the distance from the camera while maintaining visual quality. Our method provides a practical tool for creating an interactive animation of many characters while ensuring accurate and flexible controls at a modest level of computational cost.

Keywords: level-of-detail, motion interpolation, multilinear analysis, geostatistics

1 Introduction

The simulation of a large crowd of human characters has been widely used in many areas such as film production and urban engineering. The crowd simulation often requires significant amount of computational cost for planning behaviors of individual characters, synthesizing motions, and rendering the scene. The level-of-detail (LOD) control is a technique commonly used to accelerate the simulation and rendering of a large-scale crowd animation. The basic strategy of LOD control is the elimination of a redundant computation for unperceivable details of the scene. For example, the geometric LOD decreases the number of polygons of the character geometry according to the distance between the character and the camera. Simulation LOD technique reduces the computation of physics-based motion generation by automatically decreasing the degrees of freedom of dynamics models. Such LOD control is also suitable for efficiently synthesizing character animations.

Data driven motion generation techniques have been used for creating natural human animations. The motion graph technique is suited to crowd animation because it efficiently generates a new motion by resequencing segmented motion samples. The flexibility of animation controls, however, depends on the variety of the motion samples. The motion interpolation technique synthesizes a new motion by blending several motion samples with weighting functions, thus providing a continuous motion control with only a few parameters. For example, curved locomotion can be controlled by blending several walking motions in different directions with a parameter of path curvature. Most methods separately interpolate a rotational and positional element at each time and joint. However, such naive interpolation is redundant because human motion data indicates a multimodal correlation of joints, time series, and control parameters.

To improve computational efficiency, the redundancy of a motion dataset should be eliminated by a dimensionality reduction technique such as principal component analysis (PCA), by which the essential feature of the multidimensional data is extracted. PCA-based methods, however, analyze unimodal correlations separately, for example, only along the time series. For overcoming this limitation, a multilinear analysis has been introduced to computer graphics and vision technologies as a high-order generalization of traditional multivariate analysis. It extracts multimodal correlations of given data within a multilinear subspace, and can compactly and efficiently represent a group of motion samples. Moreover, a multilinear model is well suited to the multiresolutional representation of motion data.

We propose a practical technique to control a level-of-motion-detail (Motion LOD). The motion LOD combines a traditional motion interpolation technique with a multilinear model to optimize the computational cost and interpolation accuracy. High-order singular value decomposition (HOSVD), which is a high-order generalization of singular value decomposition (SVD), is used to extract a small number of principal components from the motion dataset. The dimensionality of the principal components is automatically reduced by a greedy search algorithm to optimize a trade-off between computational cost and visual quality. A nonlinear regression technique is then used to synthesize an accurate motion with the control parameter in the low-dimensional multilinear subspaces. Our algorithm is summarized in the following three steps:

1. **Multilinear analysis of motion dataset**
   Motion samples compose a motion tensor that is factorized into three correlation components: joint, time, and sample correlations. We use a HOSVD to simultaneously analyze these correlation factors. The original motion is approximated by a product of reduced components, which enables an elimination of the computational redundancy of existing motion interpolation.

2. **Progressive elimination of motion details**
   We introduce a progressive reduction algorithm to decrease the dimensionality of principal components according to the distance from the camera while maintaining visual quality. We use a hybrid dimensionality reduction technique whose parameters are automatically optimized using a greedy search algorithm that best reduces computational and storage costs.

3. **Motion synthesis using nonlinear regression**
   A new motion is synthesized by interpolating the reduced principal components within a multilinear subspace. We employ a universal kriging, which is a method of spatial statistics for accurately synthesizing a motion corresponding to given control parameters.

In the following section, we explain related work. Section 3 and 4 explain the detailed algorithm of motion synthesis and the progressive elimination of motion details. Experimental results are shown in Section 5, and we present our conclusions in Section 6.

2 Related Work

2.1 LOD Controls of Animation

Various types of crowd animation systems have been proposed using LOD and impostor techniques. Geometrical simplification [6] is a technique commonly used to reduce the rendering time by adaptively changing the number of polygons of a geometric model according to the distance from the camera. The impostor rendering technique replaces the character geometry with simpler models such as boxes and billboards [12]. Although the billboarding technique successfully reduces the rendering time, motion synthesis cannot be flexibly controlled because the animation is generated by sequentially rendering the precomputed images of an animating character on the billboard.

Simulation LOD techniques have been proposed for reducing the computational cost of the dynamics simulation of the character motion. Carlson and Hodgins [7] proposed a simulation LOD technique that adaptively switches the dynamics model of legged-creatures among three levels: whole-body, simplified body, and point-mass models. A simplified dynamics model is also used for accelerating physics-based motion generation [23]. Rendon et al. [25] introduced a simplification method of forward dynamics of articulated bodies by switching between an active and rigid joint according to the effect on the simulation result. Though these methods are well suited to multi-body dynamics simulation, their computational cost is still too high for an interactive system. On the other hand, a skeletal LOD algorithm was introduced for reducing the degrees of freedom of a character skeleton [1, 2]. This method improves the computational cost of forward kinematics by replacing a complicated skeletal structure with the simplified model.

2.2 Motion Synthesis

Data-driven motion generation techniques are widely used for creating natural human animations. A motion graph technique [5, 15, 19] generates a long motion sequence from short clips. The common reassembling approach segments motion samples into short clips and resequences them. Since this method requires less computation for selecting optimal segments and creating the transition, it is also utilized for planning group behavior [18, 30]. While this method resequences several motion segments to satisfy given kinematical constraints, the resulting animation is restricted by the variety of motion samples.

Various types of motion interpolation have been proposed using some basis functions. A linear function with stochastic sampling involves a computational cost that exponentially increases with the number of motion samples [35]. A radial basis function introduced for controlling motions with multiple parameters [26] can be adjusted to solve inverse kinematics problems [27] where the accuracy is enhanced by adaptively adding pseudo-examples with heuristics. K-nearest neighbor interpolation also proliferates parameterized motions to improve the accuracy of inverse kinematics [17]. Spatial statistics is introduced to accurately predict interpolation kernels without proliferating

motions [21]. A parametric motion graph is constructed by introducing an interpolation method to graph-based motion synthesis [24, 29]. Although these methods utilize a correlation between motion samples, they neglect correlations of joints and time series.

2.3 Statistical Analysis of Human Motion

Multivariate analysis is widely applied to computer animation technologies. PCA represents high-dimensional data with lower dimensional principal components, and is utilized for compressing multidimensional data such as mesh animation sequences [3]. Since human motion capture data indicates a high joint correlation and temporal coherence, it can be efficiently compressed by PCA. A clustered PCA is performed on the control points of a Bezier curve that approximates the motion signal [4]. Liu et al. [20] have also performed a PCA on key frames that reasonably approximates original motion data. PCA was also used to optimize dynamics parameters within a compact subspace [28] and to generate performance-driven animations with only a few signals [8]. These methods utilize a joint correlation in a single motion sequence or several motion segments for compressing the motion data. Such methods, however, should be extended to manage the correlation among samples of motion interpolation. Additionally, the complex data manipulations they required are unsuited to flexible LOD controls.

Our method utilizes a multilinear model that provides a high-order SVD (HOSVD) for the correlation analysis of multimodal, multidimensional data. HOSVD is a high-order generalization of PCA and SVD, which can simultaneously analyze multiple correlation factors based on multilinear algebra. It finds a multilinear subspace to compactly represent data. TensorTexture [33] compresses many bilinear texture functions with HOSVD. Wang et al. [34] have proposed an out-of-core HOSVD for a huge amount of visual data such as temporal volume data. A multilinear model is also used for motion recognition and annotation [32]. The HOSVD is utilized for further reducing the data dimensionality by using multimodal correlations against single-mode (or matrix) PCA. However, these methods use a simple linear function for interpolating principal components, whereas we introduce spatial statistics for enhancing the accuracy of the motion synthesis within multilinear subspace.

3 Multilinear Motion Synthesis

3.1 Motion Interpolation

Each pose of an articulated figure is usually represented by a pose vector $\mathbf{y}(\tau)$ with a discrete time τ, which consists of the rotation of all joints as well as a 3D position and orientation of the root node, with all rotational elements represented by exponential maps [13]. Each motion sample is represented by a time series of the pose vector and is temporally aligned with the same duration T via dynamic timewarping [16]. The incremental inverse timewarp function [22] is added to the pose vector for interpolating motion speed. The s-th motion sample $\mathbf{M}_s$ is then represented as a matrix (second order tensor) as $\mathbf{M}_s = [\mathbf{y}_s(1)\,\mathbf{y}_s(2)\,\cdots\,\mathbf{y}_s(T)]$, where $\mathbf{y}_s(\tau)$ denotes the pose vector at time τ of the s-th motion sample. We here express the joint degrees of freedom (DOF) as $\dim(\mathbf{y}) = P$.

A new motion is parametrically synthesized by blending the motion samples with a few control parameters $\mathbf{c}$, such as kinematical constraints or emotional parameters. Given S motion samples $\mathbf{M}_s$ and corresponding control parameters $\mathbf{c}_s$, a new motion $\mathbf{M}(\mathbf{c})$ is synthesized by the weighted average of the samples as

$$\mathbf{M}(\mathbf{c}) = \sum_{s=1}^{S} b_s(\mathbf{c})\mathbf{M}_s \ , \tag{1}$$

where $b_s(\mathbf{c})$ denotes an interpolation kernel which is optimized to satisfy the given control parameters [17, 21, 26, 27]. The resulting animation is synthesized via inverse timewarping using the incremental inverse timewarp function included in the pose vector.

Although the motion interpolation is efficient enough to generate the motion of a single character, the computational cost linearly increases according to the number of samples S and characters, joint DOF P, and the duration of motion samples T. The data redundancy of the motion samples is therefore eliminated using HOSVD based on these three-mode correlations.

3.2 Motion Tensor

A motion tensor $\mathcal{M} \in \Re^{P\times T\times S}$ is composed by lining up all motion samples in one dimension, where an element $\mathcal{M}(p,\tau,s)$ represents the p-th element of the time-aligned s-th sample $\mathbf{y}_s(\tau)$. The motion tensor is decomposed into a core tensor and three unitary matrices using HOSVD as

$$\mathcal{M} = \mathcal{Z} \times_1 \mathbf{U}_P \times_2 \mathbf{U}_T \times_3 \mathbf{U}_S \ , \tag{2}$$

where $\mathcal{Z} \in \Re^{P\times T\times S}$ is a core tensor that indicates the principal components, $\times_i$ denotes i-mode tensor product [10], and $\mathbf{U}_P \in \Re^{P\times P}$, $\mathbf{U}_T \in \Re^{T\times T}$ and $\mathbf{U}_S \in \Re^{S\times S}$ are the basic matrices of joint, time, and samples, respectively. The core tensor $\mathcal{Z}$ has an ordering property as $\mathcal{Z}(i_1,\cdots,i_n,\cdots,i_N) \gg \mathcal{Z}(i_1,\cdots,i_n+1,\cdots,i_N)$ for all possible n and i_n, which means that the higher-dimensional data is less important in approximating the original data.

Consequently, the motion tensor $\mathcal{M}$ can be approximated by a truncated HOSVD [10] as

$$\mathcal{M} = \tilde{\mathcal{Z}} \times_1 \tilde{\mathbf{U}}_P \times_2 \tilde{\mathbf{U}}_T \times_3 \tilde{\mathbf{U}}_S \ , \tag{3}$$

where $\tilde{\mathcal{Z}} \in \Re^{R_P \times R_T \times R_S}$ is the truncated core tensor, and $\tilde{\mathbf{U}}_P \in \Re^{R_P \times P}$, $\tilde{\mathbf{U}}_T \in \Re^{R_T \times T}$ and $\tilde{\mathbf{U}}_S \in \Re^{R_S \times S}$ are the truncated basic matrices with reduced ranks (R_P, R_T, R_S), respectively. Since the truncated HOSVD is generally not an optimal approximation, an alternative least square algorithm is used to estimate the globally optimal core tensor and basic matrices [11]. We combine these two reduction techniques for optimal LOD control as explained in Section 4.

A pose vector is synthesized from the approximated core tensor and basic matrices as

$$\mathbf{y}(\tau, \mathbf{c}) = \tilde{\mathcal{Z}} \times_1 \tilde{\mathbf{U}}_P \times_2 \mathbf{w}_T \times_3 \mathbf{w}_S \ , \tag{4}$$

where $\mathbf{w}_T$ and $\mathbf{w}_S$ denote the basic vector associated with the desired time τ and the control parameter $\mathbf{c}$, respectively. For the parameters satisfying $\tau_i \leq \tau \leq \tau_{i+1}$, the basic vectors $\mathbf{w}_T$ is computed by a linear interpolation of two row vectors of the basic matrix $\tilde{\mathbf{U}}_T$ as

$$\mathbf{w}_T = (1-\alpha)\mathbf{u}_{T,i} + \alpha \mathbf{u}_{T,i+1} \ , \tag{5}$$

where the coefficients α is determined by the internal ratios of τ. The basic vectors $\mathbf{w}_S$, however, cannot be linearly interpolated because the corresponding control parameter is a vector value arranged in arbitrary order. Radial basis functions are often utilized for the interpolation in a multidimensional space, but the resulting estimated data is not sufficiently accurate, which results in undesirable artifacts. We consider this defect to be common between interpolations of the pose vector and those of the tensor.

3.3 Universal Kriging for Motion Tensor

A geostatistical interpolation can optimize the interpolation kernel $b_i(\mathbf{c})$, given with $\mathbf{w}_S = \sum_{i=1}^{S} b_i(\mathbf{c})\mathbf{u}_{S,i}$, by estimating the correlation between the basic vector $\mathbf{u}_{S,i}$ and control parameter $\mathbf{c}_i$. However, their values usually have no correlation, which is against the basic assumption of geostatistics, because basic vectors are composed to be orthogonal to each other.

We here rewrite Equation (4) as

$$\begin{aligned} \mathbf{y}(\tau, \mathbf{c}) &= \tilde{\mathcal{Z}} \times_3 \left(\sum_{i=1}^{S} b_i(\mathbf{c})\mathbf{u}_{S,i} \right) \times_1 \tilde{\mathbf{U}}_P \times_2 \mathbf{w}_T \quad (6) \\ &= \left(\sum_{i=1}^{S} b_i(\mathbf{c})\mathcal{K}_i \right) \times_1 \tilde{\mathbf{U}}_P \times_2 \mathbf{w}_T \ , \quad (7) \end{aligned}$$

where $\mathcal{K}_i = \tilde{\mathcal{Z}} \times_3 \mathbf{u}_{S,i} \in \Re^{R_P \times R_T}$ is a core tensor of the i-th sample and therefore has a strong correlation with $\mathbf{c}_i$. Consequently, the kernel $b_i(\mathbf{c})$ can be optimized with geostatistics within the compact subspace.

Kriging is a practical technique of geostatistics for estimating the continuous distribution of $\mathcal{K}_i$ in the control space [9]. However, the values of $\mathcal{K}_i$ often indicate a large-scale variation in the control space, which does not meet the statistical precondition called *intrinsic stationarity* [9]. We therefore introduce a universal kriging technique [14] to manage the trend in the motion data. This decomposes each sample element $\mathcal{K}_i$ into the trend $\mathcal{K}_m(\mathbf{c})$ and the residual components $\mathcal{K}_{r,i} = \mathcal{K}_i - \mathcal{K}_m(\mathbf{c}_i)$, and assumes that the residual components satisfy the intrinsic stationarity.

The term of the blending core tensor $\mathcal{K}(\mathbf{c}) = \sum_{i=1}^{S} b_i(\mathbf{c})\mathcal{K}_i$ in Equation (7) is then replaced by

$$\mathcal{K}(\mathbf{c}) = \mathcal{K}_m(\mathbf{c}) + \sum_{i=1}^{S} b_i(\mathbf{c})\mathcal{K}_{r,i} \ . \tag{8}$$

We assume that the trend component can be represented by a hyperplane of $[\mathcal{K}_m(\mathbf{c})]_{\mathbf{v}} = \mathbf{F}[\mathbf{c}\ 1]^t$ with a matrix $\mathbf{F} \in \Re^{R_P R_T \times (\dim(\mathbf{c})+1)}$ that approximates $[\mathcal{K}_i]_{\mathbf{v}}$ by $\mathbf{F}[\mathbf{c}_i\ 1]^t$, where $[\ ]_{\mathbf{v}}$ and $[\ ]^t$ denote the vectorization of a tensor and transposition of a matrix and vector, respectively. The optimal $\mathbf{F}$ can be uniquely determined in a least-squares sense as $\mathbf{F} = \mathbf{K}\mathbf{C}^{\dagger}$, where $[\ ]^{\dagger}$ denotes a pseudo-inverse of a matrix, and the i-th column vectors of $\mathbf{K} \in \Re^{R_P R_T \times S}$ and $\mathbf{C} \in \Re^{(\dim(\mathbf{c})+1) \times S}$ are $[\mathcal{K}_i]_{\mathbf{v}}$ and $[\mathbf{c}_i\ 1]^t$, respectively.

Universal kriging optimizes weighting kernels $b_i(\mathbf{c})$ by analyzing a statistical correlation between the spatial distance between control parameters and the dissimilarity of corresponding motion samples as explained in Appendix A. The simplest dissimilarity measure between two motions is the dynamic timewarping distance, which is a Euclidean distance of two time-aligned motions as $\|\mathbf{M}_i - \mathbf{M}_j\|$. Because the Euclidean norm of a tensor is preserved under the decomposition [11], it can be approximated by $\|\mathcal{K}_i - \mathcal{K}_j\|$. An existing method introduced the nonlinear conversion of this metric at each frame; the pose distance was computed by the squared sum of positional differences between every joint pair [16]. Though the Euclidean norm between the two tensors does not directly reflect visual differences, we have experimentally confirmed that the resulting motions sufficiently preserve their accuracy. Notice that the previous interpolation separately computes individual kernels at each time frame [21], whereas our kernel optimization is computed only once for whole time sequences, a simplification that has not reduced the prediction accuracy in our experimental results.

4 Progressive Reduction of Motion Details

We here introduce a dimensionality reduction algorithm for progressively eliminating motion details according to

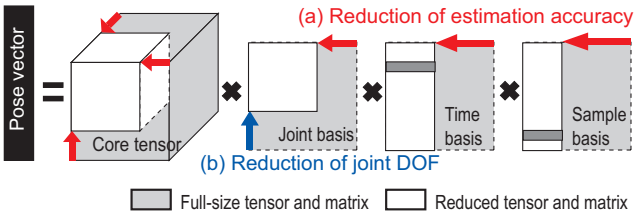


Figure 1. Pose vector approximation with reductions in core tensor and basic matrices.

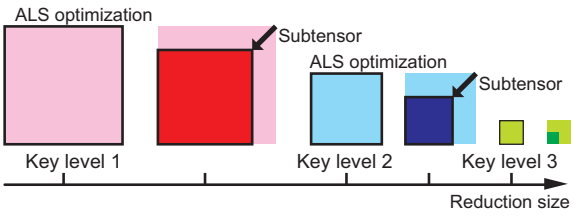


Figure 2. Hybrid reduction scheme of motion tensor with ALS optimization and tensor truncation.

the visual importance of a character. Our method simultaneously controls the prediction accuracy and joint DOF of the resulting motion as illustrated in Figure 1. The former is controlled with two versions of the tensor reduction method described in Section 3.2, which eliminates a high-frequency component of motion details. The number of active joints is adaptively decreased to reduce the computational cost of forward kinematics by truncating a basic matrix of joint correlation.

4.1 Tensor Truncation and Optimization

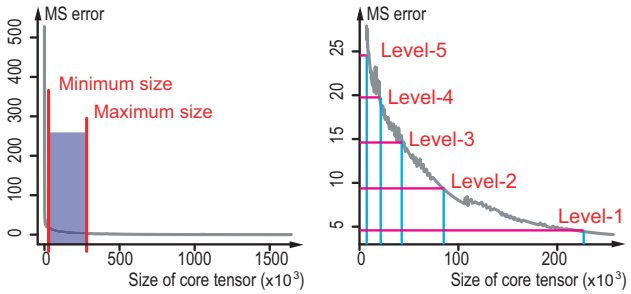
The accuracy of motion synthesis is controlled by decreasing the dimensions of a full-size core tensor $\mathcal{Z}$ and basic matrices $\mathbf{U}_P$, $\mathbf{U}_T$, and $\mathbf{U}_S$ as illustrated in Fig 1(a), and two types of reduction technique are used for dimensionality control. The simpler approach, which is similar to PCA-based data compression [3, 4, 20, 28], generates reduced core tensors by truncating the high-dimensional elements of a full-size core tensor [10]. This technique is memory efficient because it requires only one full-size core tensor generated by HOSVD. The other approach precomputes reduced core tensors using alternative least square (ALS) optimization so that the resulting core tensor preserves the Frobenius norm of an original full-size tensor for arbitrary tensor dimensions [11]. This method is more computationally efficient than the simple truncation because the precomputed data are efficiently switched according to their accuracy level in a runtime process. However, it requires redundant storage cost to store all precomputed data.

We experimentally confirmed that a slight truncation from an optimized core tensor causes only a small increase in the approximation error. For example, D-dimensional ALS optimization can be replaced by the truncation of a $(D+1)$-dimensional optimized tensor, since the truncation error is negligible within the small variation in a tensor dimension. We therefore combine these two technique to optimize a trade-off between computational and storage costs by utilizing the advantages of each approach as mentioned in the next section. Moreover, we confirmed that the visual artifact is negligible even if the dimension of a reduced core tensor is only a tenth that of the full-size tensor.

4.2 Hybrid Control of Prediction Accuracy

We introduce a hybrid algorithm for approximating a motion tensor using both simple truncation and ALS optimization. The *key level* KL_i at which the corresponding reduced core tensor is separately computed using ALS optimization, where $KL_i < KL_{i+1}$ always holds. The reduced core tensor at an arbitrary accuracy level j except at the key levels is then computed by truncating the core tensor computed at the key level KL_i satisfying $KL_{i-1} < j < KL_i$ as illustrated in Figure 2. The average computational cost improves with more key levels, but there is a trade-off between the computational and storage costs. The number of key levels is therefore manually specified depending on the type of application, and the key tensors are then automatically optimized using a greedy search so that the approximation error linearly increases according to the key level.

We here explain the details of greedy search algorithm for key tensors. Given an original motion tensor $\mathcal{M}$ and the number of key levels N, a maximum and minimum core tensor $\tilde{\mathcal{Z}}_1$ and $\tilde{\mathcal{Z}}_N$ are first composed by ALS optimization, where these tensor dimensions are manually specified (Figure 3(a)). Secondly, their approximation errors E_1 and E_N are computed by mean-squared error as $\frac{1}{PTS}\|\mathcal{M} - \tilde{\mathcal{M}}\|$, where $\tilde{\mathcal{M}}$ denotes the approximated motion tensor using a reduced core tensor. An error threshold coefficient C_E is then computed by $C_E = \frac{E_N}{NE_1}$ so that the approximation error proportionally increases according to the key level l as $E_l \approx lC_E E_1$. This is because we assume that the character in twice distant location allows to show twice larger errors considering their projected effects. The next key tensor $\tilde{\mathcal{Z}}_2$, whose approximation error of ALS optimization is larger than $2C_E E_1$, is searched by decrementing the tensor dimension from the preceding key tensor $\tilde{\mathcal{Z}}_1$. This greedy process is repeated until the size of the key tensor reaches



(a) Maximum and minimum tensor selection.

(b) Key tensor search with ALS optimization.

Figure 3. Greedy key tensor search.

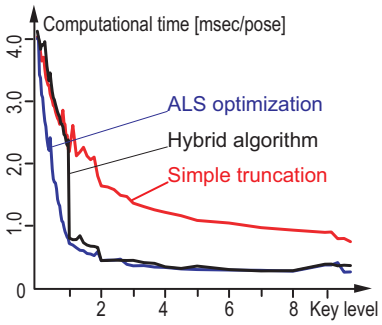


Figure 4. Comparison of computational cost of pose vector synthesis among simple truncation, ALS optimization, and hybrid algorithm.

the minimum key level (Figure 3(b)).

Figure 4 shows a comparison of the computational time of a pose vector synthesis among simple truncation, ALS optimization, and hybrid algorithm. The computational times are measured by linearly reducing the dimension of the core tensor according to the accuracy level using each method. The result of simple truncation reveals higher computational cost due to the additional computational cost of online truncation, which is propotional to the dimension of a full-size core tensor PTS (Appendix B). On the other hand, the storage requirement of all methods increases proportionally to PTS, and those of ALS optimization and hybrid algorithm are also proportional to the number of accuracy levels N_A and key levels N_K, respectively. The hybrid algorithm improves the storage efficiency if N_K is given to be much smaller than N_A. Note that the approximation errors of both methods exponentially increase according to the reduction of the core tensor, and that the ALS optimization slightly improves the accuracy of simple truncation.

4.3 Progressive Reduction of Joint DOF

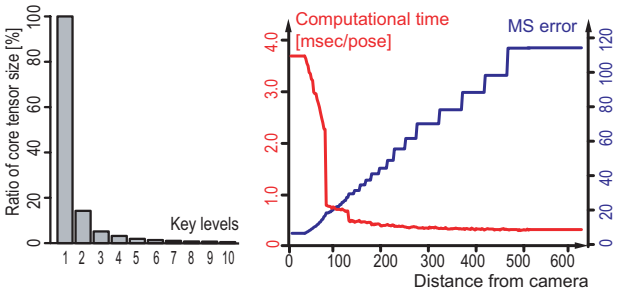
As accurate motion synthesis is redundant for a character appearing small on the screen, it is unnecessary to compute all joint rotational elements for such characters. A progressive reduction in joint DOF is therefore introduced by truncating row vectors of the basic matrix of joint correlation as illustrated in Figure 1(b), where all row vectors are manually sorted in descending order according to visual importance so that the reduction begins with joints such as wrists and ankles having less effect on motion appearance. An automated sorting method [2] should be investigated in future. Note that only the position and orientation of the root node are computed for characters at the maximum reduction level or in an invisible area.

5 Experimental Results

We demonstrate gait motions controlled with three control parameters: the height of the character, and the 2D displacement of the root position after two strides. The sample data are two-step walking motions including $S = 186$ motions of three performers, which are measured at $120Hz$ with joint DOF $\dim(\mathbf{y}) = P = 94$. The motion tensor $\mathcal{M} \in \Re^{94\times100\times186}$ is then constructed by temporally aligning all motion samples with $T = 100$ frames. A long-term sequence of the gait motion is generated by blending the second half of the synthesized motion with the first half of the following motion using ease-in/out functions.

5.1 Performance Evaluation

The $N = 10$ key tensors are generated from the motion tensor $\mathcal{M}$, where the maximum and minimum dimensions of the core tensor are $R_P \times R_T \times R_S = 40 \times 40 \times 60$ and $7 \times 7 \times 9$, respectively. The maximum core tensor is used for characters less than 30.0 meters distant from the camera, and the key level is incremented every 50.0 meters, while the joint DOF is decremented by three every 10.0 meters. Figure 5(a) shows the size ratios of a key tensor to the maximum, and Figure 5(b) is the performance result measured on 2.6 GHz Athlon 64 FX-60 CPU with 4.0 GB RAM. The approximation error is computed using the mean squared (MS) positional error of every joint from an original motion sample. The MS error increases monotonically according to the distance, and a discontinuous change is observed over 200.0 meters because the tensor truncation is seldom used due to the small variation in the dimensions of key tensors. However, the visual artifacts caused by such discontinuity are negligible. On the other hand, the computational time and dimension of a key tensor are exponentially decreased



(a) Size ratio of key tensors to the first key level.

(b) Approximation error and computational time.

Figure 5. Performance of motion synthesis with LOD control.

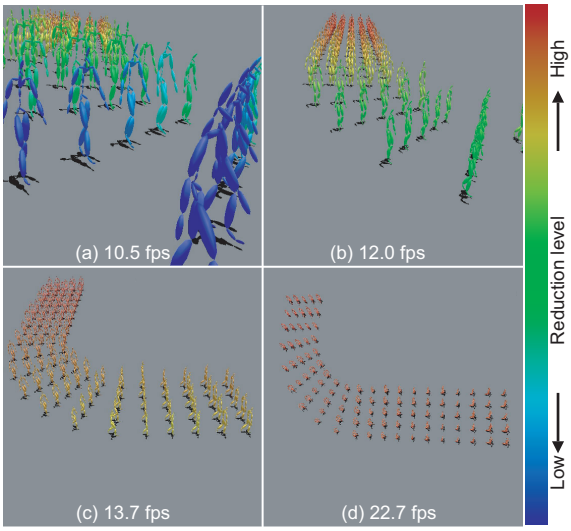


Figure 6. One hundred walkers animated by our method. Various types of walking motions generated from an identical tensor.

according to the distance, indicating that the computational cost for synthesizing the animation in an entire scene significantly depends on the number of characters whose images occupy a larger display area. The total memory requirement to store all key tensors and basic matrices is 1.45 MB in a binary format, whereas 9.7 MB data is required when all reduced core tensors are computed by ALS optimization. Our motion LOD successfully reduces the storage cost of naive interplation requiring 13.34 MB to store all motion samples.

We compared the performance of our method with PCA-based data reduction. The data dimensionality of joint DOF is decreased from $P = 94$ to $R_{P,max} = 40$ by truncating higher-dimensional principal components that are analyzed by a matrix SVD. The number of principal components $R_P < R_{P,max}$ is adaptively reduced in the runtime computation of a pose vector synthesis, in a way similar to our method. The total 5.7 MB memory was required for storing the reduced principal components and a basic matrix. This storage cost is about four times larger than that required in our method. On the other hand, the computational algorithm of PCA-based method is less complex than tensor operation, and the size of its data reduced only for joints' redundancy is larger than those fully reduced with multilinear analysis. Owing to this trade-off, the total computational cost of PCA-based method is approximately equal to that of our method. Consequently, our method can reduce the storage cost of PCA-based method while maintaining the computational efficiency and accuracy.

5.2 Crowd Animations

We demonstrate a gait animation of hundreds of characters whose motions are generated using our method. The animation is created using the motion LOD control under the same conditions described in the previous section. Moreover, we adaptively control the transition of short motion clips to create continuous locomotion with less computation. Continuous motion sequences of important characters should be generated via the transition of the synthesized short motion clips with ease-in/ease-out functions to enhance visual quality. However, since such a smooth transition is not required for the distant characters, we therefore simply switch motion clips at the appropriate frames without using any blending function for such characters.

Figure 6 shows some snapshots of the resulting animation of one hundred characters, where the color of the character indicates the reduction level. The computational time changes according to the accuracy level and the number of characters in the field of view. The worst, average, and best computational time for motion synthesis are about 9.5, 12.0, and 25.0 frames per second, respectively. Using naive interpolation without the motion LOD control, the average computational time is 3.9 frames per second. Our motion LOD control successfully reduces the computational cost of naive motion interpolation with a small visual artifact.

6 Conclusions

This paper has introduced a motion LOD method with a multilinear model. Multilinear representation provides a flexible control of the computational cost depending on the desired level of accuracy. Moreover, the storage cost is reduced by eliminating the redundancy of motion samples us-

ing the multimodal correlation factors. On the other hand, geostatistical interpolation enhances the prediction accuracy of synthesized motions. As a result, our integration of two numerical methodologies proves more accurate than the simple restorations of a multilinear model, and requires less computational and storage costs than existing motion interpolations. The progressive reduction of a motion tensor is proposed for LOD control of motion synthesis, where the data dimensionality at each accuracy level is determined according to the importance of the characters. We have introduced a hybrid method to optimize a trade-off among computational efficiency, storage cost, and visual quality. The computational and storage cost can be optimized by manually adjusting the number of key levels.

The improvements of computational and storage efficiency are not so drastic; their reduction ratios might exceed one tenth even at best. The recent work on motion data compression [4] revealed the essential difficulty in drastically reducing data size while preserving visual quality, compared to well-known algorithms for compressing image data. The motion interpolation based on spatial statistics [21] was proved that its computational cost is sufficiently low owing to its precomputable linear system. For these reasons, we believe that our system achieves reasonable performance in reducing these costs.

Our multilinear motion synthesis is unsuited for aperiodic or overly dynamic motions because the dimensionality reduction with HOSVD utilizes the multimodal correlation among similar motion clips. For example, our method works well for highly correlated motions such as gait, but dynamic movement like those in sports often indicate low correlations. This property is common to motion interpolation, and an advanced correlation analysis such as clustered tensor approximation [31] should be investigated for overcoming this limitation.

Multilinear analysis has the potential to be applied to other motion editing techniques such as motion transition and warping. We therefore plan to develop another motion LOD for motion editing or parametric motion graphs [24]. Moreover, synthesized motions are not limited to capturing human movement; any type of motion data could be managed if they satisfy the statistical preconditions. The implementation of multilinear motion synthesis on GPU could enhance its usability. The verification of these potential is the subject of our future work.

Acknowledgements

We would like to thank anonymous reviewers for helpfull comments. This work was supported by the MEXT Grant-in-Aid for Young Scientists (B) 19700090, Scientific Research (B) 18300068, and The Hori Information Science Promotion Foundation.

References

[1] J. Ahn, S. Oh, and K. Wohn. Optimized motion simplification for crowd animation. *Computer Animation and Virtual Worlds*, 17:155–165, 2006.

[2] J. Ahn and K. Wohn. Motion level of detail: A simplification method on crowd scene. In *Proc. Computer Animation and Social Agents 2004*, pages 129–137, 2004.

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

[4] O. Arikan. Compression of motion capture databases. *ACM Transactions on Graphics*, 25(3):890–897, 2006.

[5] O. Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, 2002.

[6] A. Aubel, R. Boulic, and D. Thalmann. Real-time display of virtual humans: Level of details and impostors. *IEEE Transactions on Circuits and Systems for Video Technology (Special Issue on 3D Video Technology)*, 10(2):207–217, 2000.

[7] D. A. Carlson and J. K. Hodgins. Simulation levels of detail for real-time animation. In *Proc. Graphics Interface*, pages 1–8, 1997.

[8] J.-X. Chai and J. Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics*, 24(3):686–696, 2005.

[9] N. A. Cressie. *Statistics for Spatial Data*. Wiley-Interscience, 1993.

[10] L. de Lathauwer, B. de Moor, and J. Vadewalle. A multilinear singular value decomposition. *SIAM J. Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[11] L. de Lathauwer, B. de Moor, and J. Vadewalle. On the best rank-1 and rank-$(R_1, R_2, \cdots, R_N)$ approximation of higher-order tensors. *SIAM J. Matrix Analysis and Applications*, 23(1):243–255, 2000.

[12] S. Dobbyn, J. Hamill, K. O'Conor, and C. O'Sullivan. Geopostors: A real-time geometry / impostor crowd rendering system. In *Proc. Symposium on Interactive 3D Graphics and Games*, pages 95–102, 2005.

[13] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

[14] C. J. Huijbregts and G. Matheron. Universal kriging. In *Proc. International Symposium on Techniques for Decision-Making in Mineral Industry*, pages 159–169, 1971.

[15] L. Kovar and M. Gleicher. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, 2002.

[16] L. Kovar and M. Gleicher. Flexible automatic motion blending with registration curves. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 214–224, 2003.

[17] L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568, 2004.

[18] M. Lau and J. Kuffner. Precomputed search trees: Planning for interactive goal-driven animation. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Cmputer Animation 2006*, pages 299–308, 2006.

[19] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transaction on Graphics*, 21(3):491–500, 2002.

[20] G. Liu and L. McMillan. Segment-based human motion compression. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2006*, pages 127–135, 2006.

[21] T. Mukai and S. Kuriyama. Geostatistical motion interpolation. *ACM Transactions on Graphics*, 24(3):1062–1070, 2005.

[22] S. I. Park, H. J. Shin, T. H. Kim, and S. Y. Shin. Online motion blending for real-time locomotion generation. *Computer Animation and Virtual Worlds*, 15(3-4):125–138, 2004.

[23] Z. Popović and A. Witkin. Physically-based motion transformation. In *Proc. SIGGRAPH 1999*, pages 11–20, 1999.

[24] H. Rachel and M. Gleicher. Parametric motion graphs. In *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games 2007*, 2007.

[25] S. Redon, N. Galoppo, and M. C. Lin. Adaptive dynamics of articulated bodies. *ACM Transactions on Graphics*, 24(3):936–945, 2005.

[26] C. Rose, B. Bodenheimer, and M. F. Cohen. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

[27] C. F. Rose, P.-P. J. Sloan, and M. F. Cohen. Artist directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum*, 20(3):239–250, 2001.

[28] A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics*, 23(3):514–521, 2004.

[29] H. J. Shin and H. S. Oh. Fat graphs: Constructing an interactive character with continuous controls. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2006*, pages 291–298, 2006.

[30] M. Sung, L. Kovar, and M. Gleicher. Fast and accurate goal-directed motion synthesis for crowds. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2005*, pages 291–300, 2005.

[31] Y.-T. Tsai and Z.-C. Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Transactions on Graphics*, 25(3):967.

[32] M. A. O. Vasilescu. Human motion signatures: Analysis, synthesis, recognition. In *Proc. International Conference on Pattern Recognition 2002*, pages 456–460, 2002.

[33] M. A. O. Vasilescu and D. Terzopoulos. Tensortextures: Multilinear image-based rendering. *ACM Transactions on Graphics*, 23(3):336–342, 2004.

[34] H. Wang, Q. Wu, L. Shi, Y. Yu, and N. Ahuja. Out-of-core tesor approximation of multi-dimensional matrices of visual data. *ACM Transactions on Graphics*, 24(3):527–535, 2005.

[35] D. J. Wiley and J. K. Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45, 1997.

A Kernel Optimization with Simple Kriging [21]

Given a set of S sample data $\{x_i\}$ and a set of corresponding parameters $\{\mathbf{c}_i\}$, kriging optimizes the interpolation kernel $\mathbf{b}(\mathbf{c}) = [b_1(\mathbf{c})\, b_2(\mathbf{c})\ \cdots b_S(\mathbf{c})]^t$ with a variogram function γ. First, the distance $h = \|\mathbf{c}_i - \mathbf{c}_j\|$ is computed for all pairs of samples. The distance range is equally divided by each $\triangle h$ into S_I intervals $I_u (u = 1, 2, \cdots, S_I)$ whose center is denoted by $h_u = \triangle h(u - 1/2)$, and the average of dissimilarity $\bar{\gamma}_u$ is calculated in each I_u as follows:

$$\bar{\gamma}_u = \frac{1}{n(I_u)} \sum^{n(I_u)} (x_i - x_j)^2/2 \ ,$$

where $n(I_u)$ is the number of pairs of samples included in I_u. Next, a theoretical variogram function $\gamma(h)$ is fitted to best approximate the averages $\bar{\gamma}_u (u = 1, 2, \cdots, S_I)$ in the least-squares sense. Finally, the kernel $\mathbf{b}(\mathbf{c})$ is optimized with a linear system as

$$\begin{bmatrix} \mathbf{b}(\mathbf{c}) \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{\Gamma} & \mathbf{1} \\ \mathbf{1}^t & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\gamma}(\mathbf{c}) \\ 1 \end{bmatrix}, \mathbf{\Gamma} = \{\gamma(\|\mathbf{c}_i - \mathbf{c}_j\|)\}_{ij} \ ,$$
$$\boldsymbol{\gamma}(\mathbf{c}) = [\gamma(\|\mathbf{c} - \mathbf{c}_1\|) \quad \gamma(\|\mathbf{c} - \mathbf{c}_2\|) \ \cdots \ \gamma(\|\mathbf{c} - \mathbf{c}_N\|)]^t$$

where λ is a Lagrange multiplier, and $\mathbf{1}$ is an S-dimensional column vector filled with 1.

B Computational Overhead Cost of Subtensor Operation

An efficient implementation of n-mode product of truncated core tensor and basic matrix requires of the following four steps:

1. *Unfolding* a full-size core tensor into a matrix along n-th mode of the tensor [10].
2. Generating submatrices of the unfolded tensor and basic matrix.
3. Multiplying the two submatrices.
4. Constructing a tensor from the multiplication result.

The same procedure is applied to the n-mode product of a truncated core tensor and basic vector. The unfolding tensor becomes the computational bottleneck, since its cost depends on the dimension of a full-size core tensor but is independent of the truncation size. Furthermore, computing a pose vector from a decomposed motion tensor requires the above procedure with the same number of iterations as the tensor size. Notice that this overhead cost is specifically for high-order tensor operations. In fact, submatrix operations do not require such costs.