Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2006)
M.-P. Cani, J. O'Brien (Editors)

Segment-Based Human Motion Compression

Guodong Liu and Leonard McMillan

Department of Computer Science, University of North Carolina at Chapel Hill, U.S.A.

Abstract

As more and more human motion data are becoming widely used to animate computer graphics figures in many applications, the growing need for compact storage and fast transmission makes it imperative to compress motion data. We propose a data-driven method for efficient compression of human motion sequences by exploiting both spatial and temporal coherences of the data. We first segment a motion sequence into subsequences such that the poses within a subsequence lie near a low dimensional linear space. We then compress each segment using principal component analysis. Our method achieves further compression by storing only the key frames' projections to the principal component space and interpolating the other frames in-between via spline functions. The experimental results show that our method can achieve significant compression rate with low reconstruction errors.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-dimensional Graphics and Realism — Animation; I.2.10 [Vision and Scene Understanding]: Motion.

1. Introduction

Human motion data have been used in many research fields and applications, such as animating human-like computer characters in video games, driving avatars in virtual reality environments, and generating special effects in movies. In particular, online video games often use motion data to interactively control the game characters from a remote site across the internet. As more and more motion data become available, the need to compress motion data for compact storage and fast transmission becomes imperative.

Motion data are often acquired from human actors using a motion capture system. A common form of motion capture (mocap) uses optical sensing of strategically placed *markers* (Figure 1). A mocap system utilizes the triangulation from multiple cameras to estimate the 3D positions of markers. Typically, this requires a minimal set of 40-50 markers to capture a full skeleton human motion, although capturing subtler human movements necessitates as many as over 300 markers. The 3D trajectories of the markers constitute a motion sequence with each frame, i.e. pose, represented by a vector of the marker positions. As the number of markers or the number of frames increases, motion data may easily grow very large. It is very important to develop an effective compression method to facilitate storage and transmission of motion data over the internet.

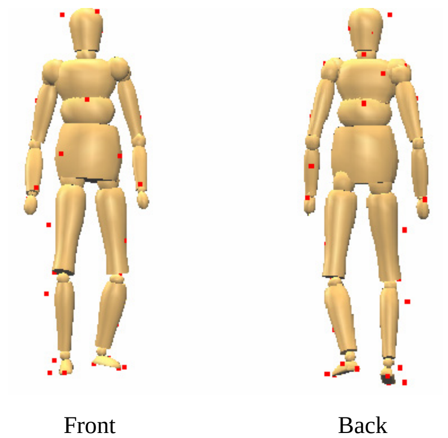


Figure 1: *A human pose constructed from the mocap marker placements. The red dots indicate the marker positions.*

As a special type of multi-dimensional time series, human motion capture data sequences exhibit considerable spatial as well as temporal coherence. Grochow et al. [GMHP04] and Safonova et al. [SHP04] demonstrated that one can accurately describe specific simple motions via a low-dimensional parameterization based on dimensionality reduction techniques. A long and complicated motion sequence exhibits local linearity, i.e. poses of each subsequence fall into a low dimensional linear subspace [LZWM06]. Besides the spatial coherence described above, temporal coherence in a human motion sequence manifests itself as a result of the strong correlations among temporally adjacent frames. Chai and Hodgins [CH05] utilized temporal coherence of the control signals to accelerate the nearest neighbor search for similar poses and dynamically constructed a local linear model for the poses to be estimated.

In order to achieve greater compression while still being able to retain high fidelity to the original motion sequences, we propose a motion compression method by exploiting both spatial and temporal coherences inherent in the human motion sequence. First, we segment a motion sequence into segments of simple motions, each of which falls into a space with low linear dimensionality. We then compress the segments individually by Principal Component Analysis (PCA). We compute PCA from each motion segment and approximate each pose position vector by its projection onto the space spanned by the leading principal components. Segment-based PCA compression typically needs fewer principal components than that of global PCA on a whole sequence to achieve similar accuracy. To take advantage of the temporal coherence and further compress the PCA projections of the poses of each motion segment, we adaptively select the key frames from each motion segment and use them as the control points for the cubic spline interpolations in the principle component space. Our compression method is efficient and easy to implement, with a corresponding decompression process that is simple and fast as well.

The rest of the paper is organized as follows. We briefly review the related work in section 2, and then introduce our compression/decompression approach in section 3. We demonstrate the performance of our method through experimental results in section 4 and finally conclude the paper with discussions and future work.

2. Previous Work

Many techniques have been developed for compression of various types of data (see [Sal00] for a complete survey). Recently there have been more developments on compression of animation mesh data due to the increasing popularity of animation movies and video games [AM00, GSK02, IR03, GK04, KG04, SSK05].

A typical animation mesh has at least thousands of vertices with fixed connectivity among vertices. An animation data sequence consists of the vertex positions for each frame/mesh. To exploit the spatial correlations among animation vertices, Alexa and Müller [AM00] constructed a compact representation of an animation sequence by conducting PCA out of a whole animation sequence. Karni and Gotsman [KG04] built upon their method and further compressed the PCA projections of frames with the linear predictor coding (LPC), which also exploited the temporal coherence. Both of these methods applied global PCA on the whole data sequence, and therefore may not be very efficient when a data sequence shows local linearity. As discussed previously, human motion capture data especially exhibit local linearity. Simple motions in fact lie near a linear subspace with much lower dimensionality. Global PCA in compression of motion capture data tends to require more principal components to be retained and thus may fail to achieve a higher compression rate. In contrast, our method starts with motion segmentation to identify the local linearity of motion sequence and then compute PCA for each segment individually. Ibarria and Rossignac [IR03] introduced a Dynapack algorithm to exploit inter-frame coherence and uses two predictors to encode the mesh motion. We exploit the temporal coherence using the spline interpolations to further compress the PCA projections of the frames.

Lengyel [Len99] proposed the decomposition of the mesh into subparts and described these parts as rigid-body motions, while only a heuristic solution to segmentation was provided. Recently Sattler et al. [SSK05] proposed a new geometry method for compressing animation sequences based on clustered principal component analysis (CPCA). They considered the trajectory of a vertex as a data point and clustered the trajectories into parts that moved almost independently. These mesh parts could then be compressed separately by PCA. Both methods attempted to break down a whole data sequence into simpler parts in the spatial domain. We, on the other hand, segment data sequences in the temporal domain. We take into account the fact that the local linearity exists with simple motions and hence choose to divide the long motion sequence into simple segments along the temporal axis. Since motion capture data sequences are usually much longer than the mesh data sequences and the number of markers is smaller than that of vertices in the animation mesh, our approach is more applicable for compression of motion data.

3. Proposed Method

In this section we describe the proposed compression and decompression methods in detail. Throughout the paper we treat each pose of motion data as a data point represented by a *3m*-dimensional column vector, $\mathbf{y} \in R^{3m}$, containing 3D positions (x, y and z coordinates) of *m* markers. Thus a motion data set with *N* pose instances can be represented by a $3m \times N$ data matrix $\mathbf{Y}=[\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$, where $\mathbf{y}_i$ is a column vector of marker positions ($i=1,\ldots, N$).

Typical motion data are captured in an absolute world coordinate system. However, we describe the relative motions in a model-rooted coordinate system where poses' differences in orientation and translation are ignored. We refer to this transformation as normalization and the new coordinate system as a normalized coordinate system. Analyzing motions in a normalized coordinate system may result in a more compact but accurate PCA model that leads to better compression. In our experiment, we select three special markers (i.e., *normalization* markers) to normalize the remaining *non-normalization* markers.

Figure 2 shows a flow chart of the compression and decompression pipeline. Note that we compress *normalization* markers and *non-normalization* markers separately. An overview of each component in the compression process is given below.

Normalization. We select the *normalization* markers and use them to normalize motion data with a model-rooted homogeneous coordinate system, where the poses that differ by translations and rotations are treated as the same.

Motion segmentation. We segment the normalized motion data sequence into subsequences of simple motions that lie near a low dimensional linear space.

Compression of segments by PCA. For each motion segment, we approximate the pose position vectors by their projections onto the space spanned by the leading principal components.

Key frame selection for spline interpolation. Given PCA projection data of each frame of a motion segment, we adaptively select the key frames as *control points* such that the spline interpolation of the rest frames yields an approximation error below a preset threshold.

Since the accuracy of *normalization* markers is crucial to the de-normalization process, we only compress the positions of these markers by selecting the key frames without PCA approximation. In other words, *normalization* markers are only involved in the key frame compression while the rest markers go through the whole compression pipeline.

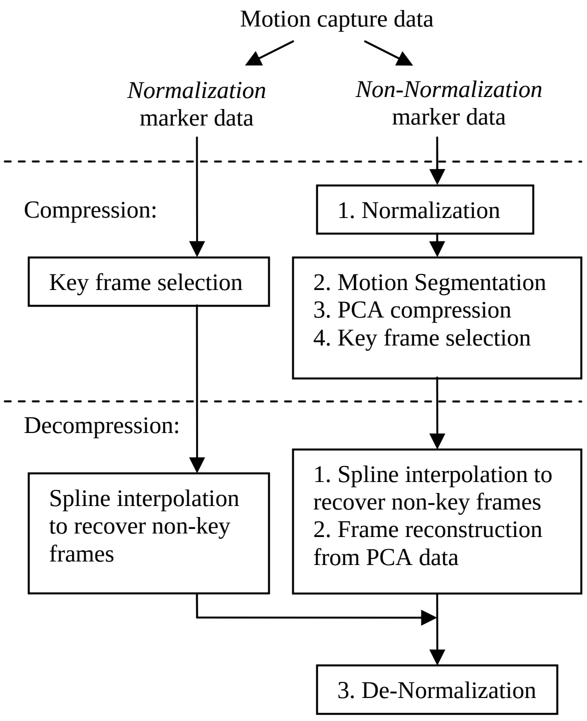

Motion capture data
Normalization marker data
Non-Normalization marker data
Compression:
1. Normalization
Key frame selection
2. Motion Segmentation
3. PCA compression
4. Key frame selection
Decompression:
Spline interpolation to recover non-key frames
1. Spline interpolation to recover non-key frames
2. Frame reconstruction from PCA data
3. De-Normalization

Figure 2: *Flow chart of motion data compression.*

The decompression is quite straightforward. We use spline interpolation to recover the positions of the *normalization* markers, as well as the PCA projections of the non-key frames for the *non-normalization* markers. We then reconstruct the positions of the *non-normalization* markers from the PCA data and finally de-normalize the *non-normalization* markers to obtain the marker positions in the original absolute world coordinate system.

3.1. Normalization

We select three markers located at the C7 vertebrae, left shoulder and right shoulder as the *normalization* markers. We use the marker at the C7 vertebrae as the origin. We define the x, y and z axes in the normalized coordinate system as follows. We retain the z-axis of the original world coordinate system. We compute a vector from the left shoulder marker to the right shoulder marker and then project it onto the horizontal plane. The projected vector is considered as our x-axis. The cross product between the newly defined z- and x- axes yields our y-axis.

3.2. Motion Segmentation

The goal of segmentation is to divide a long and complicated motion sequence into segments of simple motions, which typically lie in a space with much lower dimensionality than the whole sequence. Compressing each motion segment individually enables us to achieve a higher compression rate and retain a reasonable reconstruction quality at the same time.

There have been studies on motion segmentation [PRM00, LWS02, BSP*04]. We apply the probabilistic PCA (PPCA) approach presented in [BSP*04] to segment a motion sequence into segments of distinct behaviors. As an extension of the traditional PCA, PPCA is based on a probabilistic model [TB99] and models the residual variance discarded by PCA. Motion data are considered as an ordered sequence of poses and are modeled with Gaussian distributions in PPCA. A motion sequence is segmented where there is a local change in the distribution of the poses.

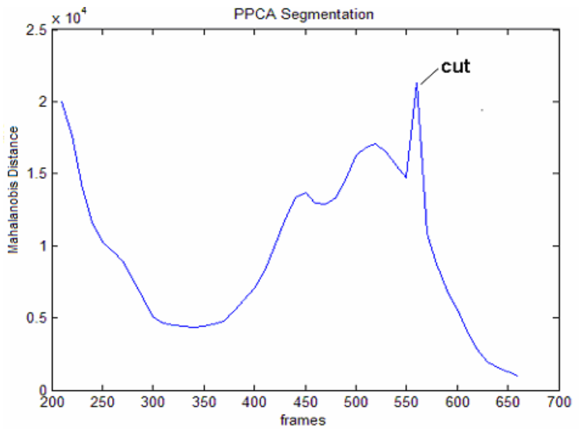


Figure 3: *Plot of Mahalanobis distance H as K is repeatedly increased by Δ in PPCA segmentation.*

For clarity, we briefly review how to implement PPCA for motion segmentation as described in [BSP*04]. We perform PPCA to model the first K frames of a motion sequence with a Gaussian distribution, where the covariance is determined by the intrinsic dimensionality estimated from PCA, as well as the average of discarded variance in PCA. We choose the intrinsic dimension as *p* such that the leading *p* principal components in PCA cover the portion of the variance not less than a preset threshold τ. We then estimate how likely motion frames K+1 through K+T belong to the Gaussian distribution defined by the first K frames. We do this by calculating an average Mahalanobis distance H. Next we increase K by a small number of frames Δ and repeat the estimation of distribution for the first K frames (K:= K+Δ), and compute the distance H with respect to the new distribution. Figure 3 shows a plot of H as K repeatedly increases by Δ. In the plot, a cut is declared at the peak following a valley and when the difference in H between the valley and the peak is greater than a threshold R. In general, increasing R results in fewer segments and decreasing R results in a finer segmentation. When a cut is made, the first K frames constitute one segment and are removed from the sequence. We continue to segment the rest subsequence until the length of the remaining subsequence is less than the sum of the initial value of K and T. In our experiments, we set the initial value K=200, τ = 0.95, T=150, Δ=10, and R=500. The segmentation results are not very sensitive to the minor adjustment of K, T or Δ, although bigger adjustments would certainly affect the size of the segments.

3.3. Compression of Segments by PCA

PCA is a dimensionality reduction technique, which retains those characteristics of a dataset that contribute most to its variance. For a motion segment whose pose position vectors lie near a much lower dimensionality, PCA is a very effective method to find the low dimensional space. We compute PCA for each motion segment and keep the first *k* eigenvectors such that the residual variance covered by the discarded eigenvectors is less than a preset threshold. The projections of the pose position vectors onto the *k*-dimensional principle component space are used as the approximations of the original poses. A motion segment is represented by *k*-dimensional trajectories over time, with *k* being varied in different motion segments.

The reconstruction errors of all the markers are not perceived on the same scale by our sensing system. Human vision tends to be more sensitive to errors on certain body parts. For example, even a very small error on the positions of foot markers could be detected and perceived as a major artifact called sliding feet [KGS02, IAF06]. To address this issue, we compress the foot markers separately from the rest markers with a tighter PCA error tolerance.

3.4. Key Frame Selection for Spline Interpolation

Human motion capture data demonstrate strong temporal coherence as most time series do. We can reliably estimate a frame with its temporally adjacent neighbours. We opt to select and store only the PCA projections of the key frames from a motion segment to achieve further compression. In decompression we can apply the cubic spline interpolation to recover the non-key frames using the saved key frames as control points for the spline function.

We adopt the cubic spline interpolation approach because of its computational simplicity, good approximation property and implicit smoothness (minimum curvature property). Selection of the key frames as control points is an adaptive process. We start by fitting each PCA projected trajectory by a cubic spline function with four evenly distanced control points, two at both ends of the motion segment and the other two at the 1/3 and 2/3 temporal positions of the segment. We then interpolate all of the frames using the cubic spline functions and compute the interpolation errors. For each frame, the approximation error of spline interpolation is calculated as the L2 norm of the difference between the k-dimensional interpolated vector and the original projection vector. If the approximation error of the interpolation exceeds a preset threshold for any frame between the two existing control points, then the frame in the middle of those two control points is selected as a new control point and is added to the list of the existing control points. We continue adding control points and interpolating frames until the interpolation errors of all the frames are within an error threshold.

As mentioned earlier, the three *normalization* markers only go through one-level compression via the key frame selection. Since these three markers are crucial to the de-normalization of the rest markers and ultimately affect the final decompression result, we apply a more stringent error tolerance in selection of key frames.

For each compressed motion segment, we need to store the key frames for the three *normalization* markers; one mean vector, the *k* principal component vectors and the PCA projections of the key frames for the *non-normalization* markers.

3.5. Decompression

Decompression of motion data is conducted for *normalization* markers and *non-normalization* markers separately as follows.

- *Normalization* markers:

Since we only compress the motion data of the *normalization* markers in their original measurement space by selecting the key frames as control points for the spline interpolation, we simply need to reconstruct the non-key frames with spline interpolation using those key frames as control points.

- *Non-normalization* markers:

Given the PCA projections of the key frames in each segment, we run *cubic* spline interpolation using those key frames as control points to estimate the PCA projections of other frames. Then we reconstruct the marker positions in the normalized coordinate system using the principal component vectors associated with the segments. Finally we transform the normalized data back to the original coordinate system using the reconstructed positions of the *normalization* markers.

An inherent shortcoming with the local linear modeling approach is the temporal discontinuity at the transitions between PCA models, manifested as visible jerkiness in the reconstructed motion. For example, if we approximate two temporally adjacent motion segments using two different sets of principal component vectors, then it is likely to see jerkiness at the transition frames between these two segments.

Instead of using only one PCA model to reconstruct the poses at the transition of local PCA models, we use a mixture of PCA models associated with the particular pose and its neighboring poses. Let $\mathbf{b}_{t,i}$ be a column vector containing the reconstructed 3D positions of markers at pose t based on PCA model i, we estimate the marker positions $\mathbf{y}_t$, as

$$\mathbf{y}_t = \Sigma_i\, w_i\, \mathbf{b}_{t,i}\,,$$

where $w_i = r_i / (h+1)$ is a weight for the ith PCA model, r_i is the number of frames corresponding to the ith model among the frames t-h/2 to t+h/2. Basically, we put more weight on the model that is favored by more of the h +1 poses. In our experiments, h=10-20 works very well.

4. Experimental Results

We evaluated our compression method with Carnegie Mellon University's Graphics Lab motion capture database available at http://mocap.cs.cmu.edu. We used the 3D motion data on a set of 41 markers. The sampling rate is 120 frames per second. Table 1 shows the basic information of the motion sequences.

Sequence	Description	#Frames	Size (Kbytes)
1	Jumping jacks, side twists, bending over, squats	4,592	4,413
2	Long breakdance sequence	4,499	4,324
3	Walk, squats, running, stretches, jumps, punches, and drinking	10,590	10,177
4	Walk, squats, stretching, kicking, and punching	9,206	8,847

Table 1: *Used motion capture data sequences.*

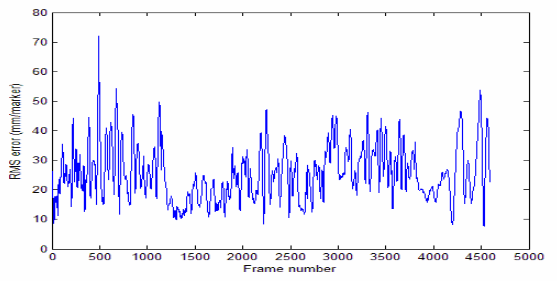
As discussed previously, the reconstruction errors may not be perceived on the same scale for different markers, so we chose different error tolerances according to the importance of markers. The PCA residual error threshold was set as 10 mm/marker for the six foot markers and 30 mm/marker for the other *non-normalization* markers. The spline interpolation error threshold was set as 10 mm/marker for *non-normalization* markers and 1 mm/marker for *normalization* markers. Larger error tolerances typically result in more visible artifacts in the reconstruction. Extreme cases include apparent shifting of certain body parts as well as jerkiness at the transitions. We assessed the performance of our compression method with regards to compression ratio and reconstruction quality. We visually evaluated the reconstruction results using our motion model viewer and quantitatively evaluated the reconstruction errors with two distortion measures. The first one is a distortion rate d similar to [KG04], which measures the quality of reconstruction for the whole motion sequence, and is defined as

$$d = 100\frac{\| A - \tilde{A} \|}{\| A - E(A) \|},$$

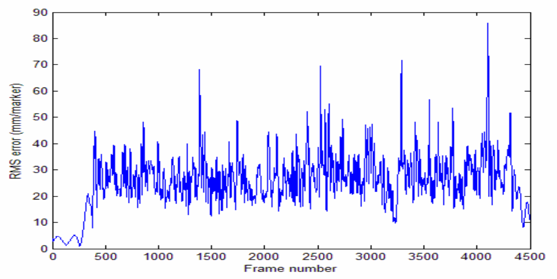
where A is a $3m$ x N data matrix containing the original motion sequence collected from m markers over N frames. $\tilde{A}$ is the reconstructed result of the same motion sequence after decompression. $E(A)$ is an average matrix in which each column consists of the average marker positions for all the frames. The second distortion measure is the per-frame distortion, which is defined as the Root Mean Squared (RMS) error (mm/marker) in each frame.

Sequence	1	2	3	4
# of Segments	6	9	10	9
Average # of Principle Components	4.6	10.9	3.1	4.2
# of Control Points	117	211	235	221
Compression Ratio	1:55.2	1:18.4	1:61.7	1:56.0
Distortion Rate d (%)	5.1	7.1	5.1	5.4
Compression Time (ms/frame)	1.3	1.4	1.3	1.2
Decompression Time (ms/frame)	0.7	0.7	0.7	0.7

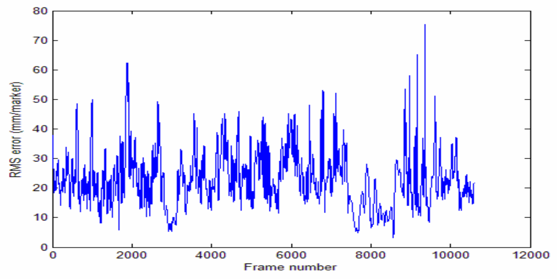
Table 2: *Compression and reconstruction results.*



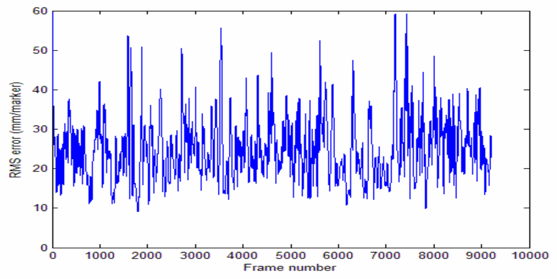
(a) Sequence 1



(b) Sequence 2



(c) Sequence 3



(d) Sequence 4

Figure 4: *Reconstruction errors of decompression.*

The compression and reconstruction results of four motion sequences are presented in Table 2. The frame-by-frame distortions are shown in Figure 4. These results showed that our compression method achieved a high compression ratio with low distortion rate. As motion sequences became more complicated, more principle components and key frames were needed to ensure the reconstruction quality. For example, the long sequence of *breakdance* was quite fast-paced and sophisticated, thus it required more principle components and key frames than the other sequences to obtain comparable reconstruction quality, resulting in a compression rate not as high as those of the other motion sequences. The accompanying video demonstrates that the reconstructed motion sequences are of reasonably good quality with fairly smooth transitions from one PCA-modeled segment to another. We also show one example in the video that using a tighter PCA residual error tolerance for the foot markers instead of a uniform tolerance for all the markers greatly reduces the sliding-feet artifact and thus significantly improves the perceived visual quality of the reconstructed motions. Some reconstructed sample frames are given in Figure 5.

Sequence	Method	Compression Ratio	Distortion Rate (%)
1	Global PCA	1:5.3	4.2
	Piecewise PCA	1:8.0	4.1
	Our method	1:55.2	5.1
2	Global PCA	1:3.7	5.8
	Piecewise PCA	1:5.0	5.4
	Our method	1:18.4	7.1
3	Global PCA	1:6.4	4.9
	Piecewise PCA	1:9.4	4.3
	Our method	1:61.7	5.1
4	Global PCA	1:5.3	4.5
	Piecewise PCA	1:8.4	4.5
	Our method	1:56.0	5.4

Table 3: *Comparison of different compression methods.*

We compared our method to the global PCA method and the segment-based piecewise PCA method without spline interpolation (Table 3). Piecewise PCA compression method improved the compression performance for all the sequences as compared to the global PCA compression method. However, the most significant improvement came from its use combined with the spline interpolation which efficiently incorporated the temporal coherence of the sequences. Note that 3 out of 41 markers were used as normalization markers in our experiment and only 38 markers actually went through the segment-based PCA compression, so the compression effect of piecewise PCA may not be fully demonstrated. However, we expect significant advantage of PCA compression when the number of markers increases substantially.

We run the experiments in Matlab V7 on a Dell Inspiron Laptop, with 1.4GHz CPU and 512M physical memory. Both of our compression and decompression algorithms are very fast and scale linearly with the number of frames. As Table 2 shows, the compression time is about 1.3 ms/frame for all four motion sequences; while the decompression time for each sequence is 0.7 ms/frame.

5. Conclusions and Future Work

We present a novel method to compress human motion data sequences. We exploit both spatial and temporal coherences of motion data to achieve a considerably high compression rate with low degree of distortions. The experimental results show that it is important to segment a long and complicated motion sequence into subsequences of short and simple motions and compress each of these subsequences separately. Our segment-based approach requires many fewer principal components to achieve the same error threshold than with traditional global PCA compression, and this leads to a higher compression rate and better reconstruction results. PCA approximation is an effective way to characterize the correlations among markers and significantly reduce the dimensionality of the marker data without the loss of important aspects of the motion. Spline interpolation on top of the PCA approximation further improves the already high compression rate by taking into account the correlations among temporally adjacent frames.

Our compression method is a lossy compression scheme. If more accuracy is demanded, quantization can be used to store the differences between the reconstruction results and the original true values so that the actual errors can be further reduced to the quantization errors only. However, doing so means we have to allocate extra space to store these quantized errors and the trade-off would be the reduced compression rate. Our experiment results show that few visual artifacts appear when the information reflected by the residual errors is discarded, so the use of quantization may not be necessary.

Motion capture data are often represented as joint angles in animation research. Since joint angle is a hierarchical representation, it is difficult to achieve high quality compression by directly compressing the joint angle data due to the accumulation of errors along the chain of the joint angles. As a solution, we can first convert joint angles into joint positions and then compress these joint positions instead. In a concurrent work by Arikan [Ari06], the conversion to positional data of so-called *virtual markers* was also involved in the compression of joint angle data. It is worth examining how effective our method is as compared to Arikan's approach in handling the joint angle data.

While we demonstrated our compression method with full-body human motion capture data, in our future work we would like to evaluate how well our method performs in compressing other types of human motion data, such as facial expressions. In addition, the application of our compression method to other formats of data such as animation meshes, sequences of point clouds and range scan data also merit further study.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The data used in this project was obtained from CMU's motion capture database (mocap.cs.cmu.edu), which was created with support from NSF EIA-0196217.

References

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

[Ari06] Arikan O.: Compression of Motion Capture Databases. To appear in *Proceedings of SIGGRAPH 2006* (2006).

[BSP*04] BARBIČ J., SAFONOVA A., PAN J., FALOUTSOS C., HODGINS J., POLLARD N.: Segmenting motion capture data into distinct behaviors. *Graphics Interface*, (2004), pp. 185–194.

[CH05] CHAI J., HODGINS J.: Performance animation from low-dimensional control signals. In *Proceedings of SIGGRAPH 2005,* (2005), pp. 686–696.

[GMHP04] GROCHOW K., MARTIN S. L., HERTZMANN A., POPOVIC Z.: Style-based inverse kinematics. In *Proceedings of SIGGRAPH 2004* (2004), pp. 522–531.

[GSK02] GUPTA S., SENGUPTA K., KASSIM A. A.: Compression of dynamic 3d geometry data using iterative closest point algorithm. *Computer Vision and Image Understanding* (2002), pp. 116–130.

[GK04] GUSKOV I., KHODAKOVSKY A.: Wavelet compression of parametrically coherent mesh sequences. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 183–192.

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

[IAF06] IKEMOTO L., ARIKAN O., FORSYTH, D.: Knowing when to put your foot down. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Interactive 3D Graphics* (2006), pp. 49–53.

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

[KGS02] KOVAR L., GLEICHER M., SCHREINER J.: Footstake cleanup for motion capture editing. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002), pp. 97–104.

[Len99] LENGYEL J. E.: Compression of time-dependent geometry. In *Proceedings of the 1999 ACM SIGGRAPH Symposium on Interactive 3D Graphics* (1999), pp. 89–95.

[LWS02] LI Y., WANG T., SHUM H. Y.: Motion texture: A two-level statistical model for character motion synthesis. In *Proceedings of SIGGRAPH 2002,* (2002), pp. 465–472.

[LZWM06] LIU G., ZHANG J., WANG W., MCMILLAN L.: Human motion estimation from a reduced marker set. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Interactive 3D Graphics* (2006), pp. 35–42.

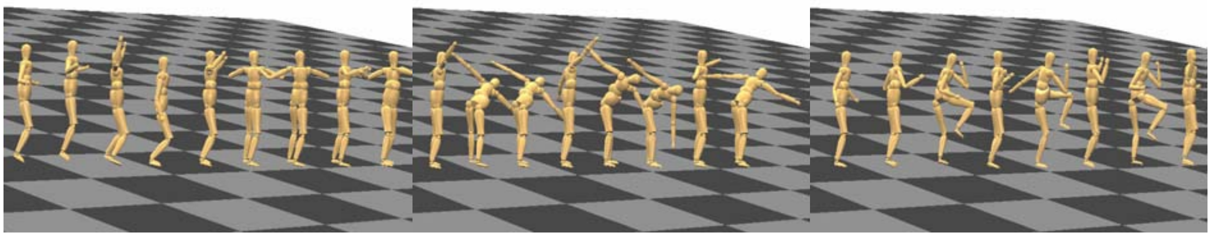
[PRM00] PAVLOVIC V., REHG J. M., MACCORMICK J.: Learning switching linear models of human motion. In *Proceedings of NIPS 2000* (2000), pp. 981–987.

[Sal00] SALOMON D.: Data Compression: The Complete Reference, second ed., 2000.

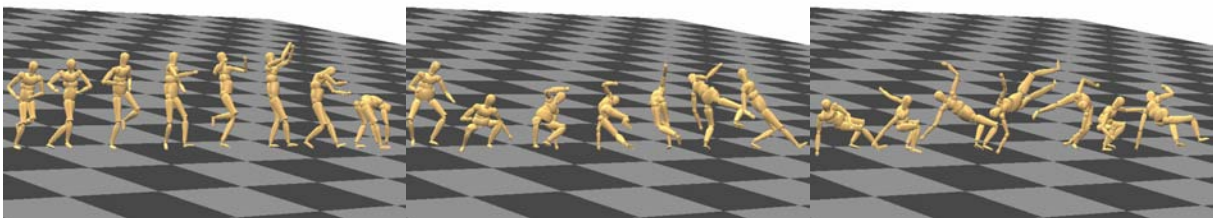
[SHP04] SAFONOVA A., HODGINS J., POLLARD N. P.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *Proceedings of SIGGRAPH 2004,* (2004), pp. 514–521.

[SSK05] SATTLER M., SARLETTE R., KLEIN R.: Simple and efficient compression of animation sequence. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 209–217.

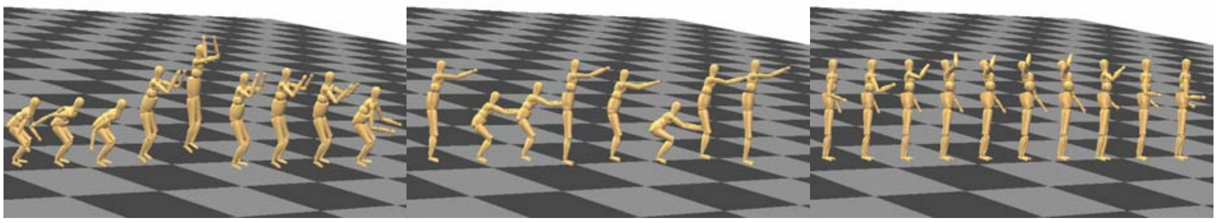
[TB99] TIPPING M. E., BISHOP, C. M.: Probabilistic principal component analysis. *Journal of the Royal Statistical Society* Series B, (1999) Vol. 61(3), pp. 611–622.



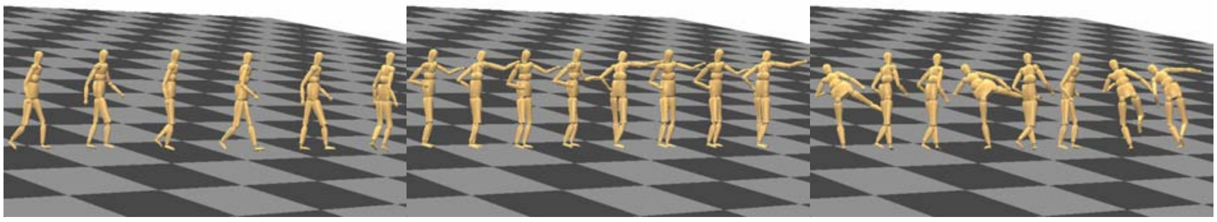
Sequence 1: (a) jumping jack & side twists (b) bending over (c) stretching



Sequence 2: long sequence of breakdance



Sequence 3: (a) jumping (b) squats (c) drinking



Sequence 4: (a) walking (b) punching (c) kicking

Figure 5: *Sample frames of reconstructed motion sequences.*