HYBRID LOW-DELAY COMPRESSION OF MOTION CAPTURE DATA

Choong-Hoon Kwak and Ivan V. Bajić

School of Engineering Science, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada

ABSTRACT

Motion Capture (MoCap) is becoming important in many areas of technology, science, and art, including graphics, visualization, gaming, and medical applications. In parallel with its increased use and abundance, compression of this kind of data is becoming more important. In this paper, we propose a hybrid low-delay compression scheme for MoCap data that is particularly suitable for interactive applications such as online gaming or telemedicine. Experimental results confirm the superiority of the proposed approach against state-of-the-art methods for MoCap compression in both compression efficiency and delay, making it suitable for interactive applications.

Keywords— Motion capture, MoCap data compression, transform coding, hybrid coding

1. INTRODUCTION

Motion Capture (MoCap) has sparked a revolution in the way people play computer games. In addition to this well-known use scenario adopted by major gaming platforms (Wii, Xbox, and PlayStation), MoCap is finding use in an increasing number of other applications, including entertainment [1], arts [2], and rehabilitation [3]. In parallel with its increased use and abundance, compression of MoCap data is becoming more important. The type of compression that is appropriate for a particular application depends on various requirements of that application. For storage, compression efficiency and the ease of browsing and indexing are usually the most important criteria. For interactive applications such as online gaming, real-time encoding and decoding, as well as low delay, are crucial.

In the case of human skeletal MoCap data, 41 motion markers are usually placed on the subject according to the marker placement guide [4] to capture the movement of all flexible joints of the skeleton. In other cases, appropriate number of markers can be placed according to the specific application requirement, e.g., hand or face animation. Two of the most popular MoCap file formats are C3D and BVH. A C3D file contains the global 3D coordinates of each marker at each sampling instant. On the other hand, BVH data is obtained by processing global 3D marker coordinates to obtain the position of each joint of the human skeleton, possibly at instants different from marker position sampling instants. Hence, BVH data really represents (an estimate of) the articulated human skeletal movement. The two types of data are illustrated in Fig. 1, where we show C3D data (marker locations) as black squares, and BVH data (human skeletal joints) as red circles connected by skeletal bones. To facilitate comparison with prior work, all experiments in this paper are based on C3D human skeletal MoCap data. However, the proposed approach is rather general and applicable to other kinds of MoCap data as well.

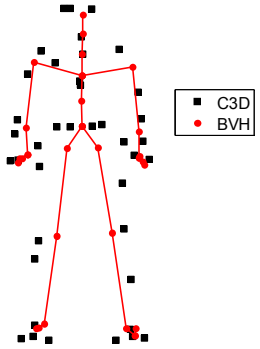

Fig. 1. Human Motion Capture (MoCap) data: global positions of markers (C3D) vs. locations of articulated joints (BVH).

In this paper we propose a low-delay compression method for MoCap data, which is appropriate for interactive applications (e.g., online gaming) that cannot tolerate large encoding delay. Experimental results demonstrate that despite having a much lower encoding delay than the state-of-the-art MoCap compression methods, our approach is able to deliver superior compression efficiency as well. Prior work on MoCap data compression is reviewed in Section 2. In Section 3, we describe the details of the proposed compression approach. Experimental results are presented in Section 4, and conclusions are drawn in Section 5.

2. PRIOR WORK ON MOCAP DATA COMPRESSION

Due to the physical continuity of real object motion trajectories in space, MoCap data exhibits high redundancy

This work was supported in part by the NSERC/CCA New Media Grant STPGP 350740. E-mail: cka21@sfu.ca, ibajic@ensc.sfu.ca.

978-1-61284-350-6/11/$26.00

in the temporal direction. Several methods have been proposed in recent years to exploit this redundancy for MoCap data compression. Principal Component Analysis (PCA) [7], [8] is one of the popular techniques widely used in this field due to its dimensionality-reduction properties. For example, the global PCA (GPCA) approach [7] applies PCA to the entire MoCap sequence and retains the eigenvectors that account for most of the signal energy. The threshold on how much error can be globally tolerated determines how many eigenvectors are kept. Projection of MoCap data onto the space spanned by the retained eigenvectors is then encoded. The downside of this method is that the characteristics of the motion may change considerably over a long sequence, so GPCA typically needs to retain more eigenvectors than would be necessary if the PCA were to be applied locally.

To mitigate this problem, a piecewise PCA can be applied to shorter segments of motion. To apply piecewise PCA, one needs to determine the points in time where the motion changes significantly. These points, sometimes called "key frames," mark the positions where the PCA eigenvectors should change in order to better approximate the motion in the next segment. In [8], two such approaches have been examined. In one approach, which we shall refer to as PPCA, piecewise PCA is applied directly to the segments in-between the identified key frames. In another approach, which we shall call SEG, PCA is applied in conjunction with spline interpolation within each segment to further exploit temporal coherence of MoCap data. Related approaches based on motion segmentation have been proposed in [9] and [10]. There have also been proposals to modify PCA to a transform that is more appropriate for approximating human motion. For example, in [11], a the extension of PCA named Principal Geodesic Analysis (PGA) is used for MoCap data compression. In this scheme, which was developed specifically for articulated human skeletal motion (e.g., BVH files), one needs the positions of end-joints and root joints across time during a motion segment before PGA can be applied. All these methods impose a certain encoding delay due to the fact that the redundancy-removal (through PCA or its variation) is applied temporally, which means that a number of MoCap data frames have to be captured and stored before compression of the first frame in the segment can start. Such approach is not suitable for interactive applications.

PCA is not the only way to remove temporal redundancy from MoCap data. For example, the approach in [12], which uses Body Animation Parameters (BAPs), analyzes motion across time and creates a dictionary of most frequent patterns, which are then used for indexing BAPs of individual frames. In this approach, the entire MoCap sequence needs to be processed before compression can start, which again makes it unsuitable for interactive applications. MPEG-4 Bone Based Animation (BBA) described in [5] and [6] uses two other methods for removing temporal redundancy. In one approach, temporal differencing of neighboring frames is used, while in the other approach, Discrete Cosine Transform (DCT) is applied temporally across 16 consecutive MoCap frames. The latter approach introduces 15 frame encoding delay, which may make it unsuitable for interactive applications. The former approach has no encoding delay, however it's compression efficiency is not particularly good, as will be seen in the results in Section 4.

3. HYBRID LOW-DELAY MOCAP DATA COMPRESSION

Each frame $\mathbf{f}[n]$ of raw MoCap data consists of global x-, y-, and z-coordinates of each marker:

$$\mathbf{f}[n] = (x_1[n], y_1[n], z_1[n], ..., x_m[n], y_m[n], z_m[n])^T, \quad (1)$$

where m is the number of markers and n is the frame index. In MoCap data, redundancy exists temporally (e.g., between $\mathbf{f}[n]$ and $\mathbf{f}[n+1]$) as well as spatially (e.g., between $x_i[n]$ and $x_j[n]$). In order to remove both types of redundancy, we employ the so-called 'hybrid' strategy from video compression, where two different kinds of redundancy removal are applied in spatial and temporal direction. In particular, to satisfy the low-delay requirement, we use differential encoding temporally, while for spatial redundancy removal we use the Discrete Cosine Transform (DCT). The block diagram of the proposed MoCap encoder is shown in Fig. 2 below. Its structure is similar to well-known hybrid video encoders, with appropriate modifications to make it suitable for MoCap compression. This structure provides both low delay and high compression efficiency, as will be shown in Section 4. In the remainder of this section we describe various encoder components.

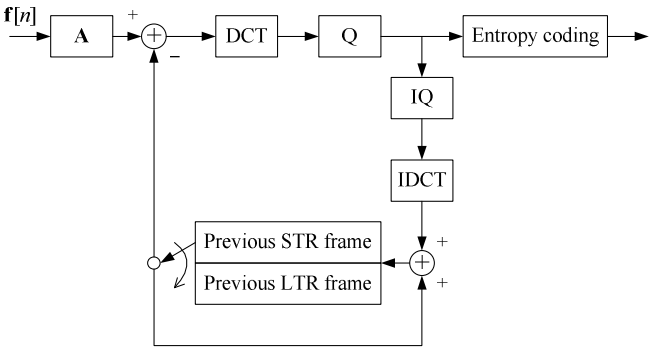

Fig. 2. Block diagram of the proposed hybrid MoCap encoder.

3.1. Reordering of MoCap data

One of the reasons for the success of transform-based image and video coding is that in natural images, most of the signal energy is concentrated at low frequencies. This fact is exploited by spatial transforms such as DCT and wavelets.

MoCap data does not exhibit such a behavior. An illustration is given in Fig. 3, where we show amplitudes of DCT coefficients of several frames from the MoCap sequence 85_12, which is one of the test sequences used in the experiments in Section 4. As seen in the figure, a substantial amount of signal energy is at high frequencies. The reason is that groups of markers are often clustered together in the 3D space (e.g., near the feet of the skeleton in Fig. 1), which results in their x-, y-, and z-coordinates having similar values, i.e.,

$$x_i[n] \approx x_{i+1}[n] \approx \ldots \approx x_{i+k}[n], \quad (2)$$

where k is the size of the cluster, and similarly for the y- and z-coordinates. This creates local periodicities in $\mathbf{f}[n]$, which gives rise to energy at higher frequencies. For this reason, we reorder the components of $\mathbf{f}[n]$ into a new vector $\mathbf{g}[n]$ where x-, y-, and z-coordinates are grouped together:

$$\mathbf{g}[n] = (x_1[n],\ldots, x_m[n], y_1[n],\ldots, y_m[n], z_1[n],\ldots, z_m[n])^T. \quad (3)$$

This reordering is a linear transformation of the vector $\mathbf{f}[n]$, which can be represented in the following form

$$\mathbf{g}[n] = \mathbf{A}\,\mathbf{f}[n], \quad (4)$$

where $\mathbf{A}$ is the $(3m)\times(3m)$ circulant matrix shown below.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}. \quad (5)$$

After such a transformation, the energy in vector $\mathbf{g}[n]$ becomes concentrated at low frequencies. An illustration is given in Fig. 4, which shows the amplitudes of DCT coefficients of the set of frames from Fig. 3 after reordering. This linear reordering is the first component of our MoCap encoder, represented by $\mathbf{A}$ in Fig. 2. After reordering, the characteristics of MoCap frames become more similar to the characteristics of natural images. This improves the chance of spatial transform to provide efficient compression *without explicit modeling of the spatial configuration of markers*, which in turn makes our method applicable to a wide range of MoCap data, not only human skeletal data.

3.2. Predictive coding

The sampling frequency of MoCap data is usually fairly high compared to video, typically 120 frames per second (fps), so high compression gain is expected from temporal predictive coding. In particular, due to the close temporal spacing between neighboring MoCap frames, the most recent previous frame usually provides a very good prediction of the current frame.

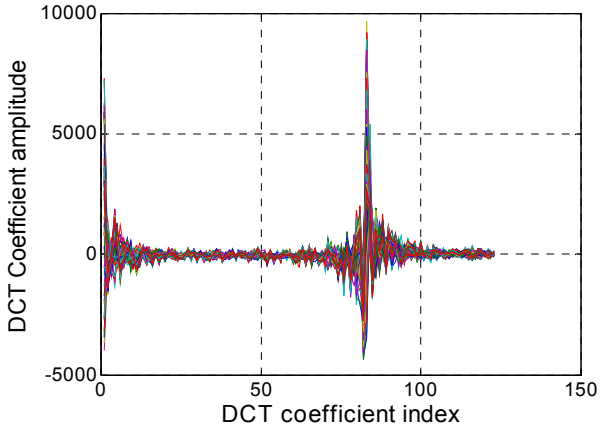

Fig. 3. Amplitudes of DCT coefficients of several frames **f** from MoCap sequence 85_12.

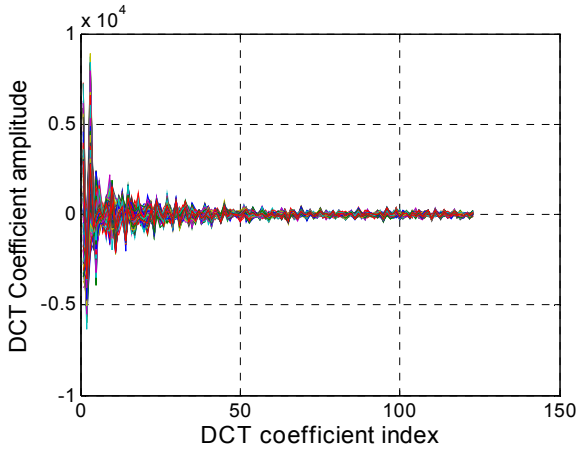

Fig. 4. Amplitudes of DCT coefficients of reordered MoCap frames **g** = **Af** corresponding frames **f** from Fig. 3.

We define two types of MoCap frames: Long-Term Reference (LTR) frames, which are encoded either in the intra-mode, or predictively from the previous LTR frame, and Short-Term Reference (STR) frames, which are encoded predictively from the immediately preceding frame, be it LTR or STR. The first frame in the sequence is always an intra-coded LTR frame. In the experiments in Section 4, all LTR frames after the first one were coded predictively. LTR frames were repeated at regular intervals (every 30 or 60 frames, as indicated in the particular set of results), while all other frames were encoded as STR frames.

Predictive coding loop in Fig. 2 includes a memory buffer for two frames, one LTR frame, and one STR frame. Both buffers contain the most recent frame of each type. When an LTR frame is being encoded, the switch points to the LTR buffer and prediction is performed from the

previous LTR frame. After reconstruction, this LTR frame replaces the previous LTR frame in the LTR buffer. The switch remains in the same position for encoding the next STR frame (i.e., the first frame following an LTR frame), after which the switch is pointed to the STR frame buffer and remains there until the next LTR frame. This prediction structure also allows simple temporal scalability, since LTR frames can be decoded at a lower frame rate independently of STR frames.

3.3. DCT transform

DCT transform is applied spatially to each frame. If m is the number of markers, each frame is a vector of length $3m$, so a one-dimensional DCT of length $3m$ is used. DCT is applied directly on the first frame (after subtracting the mean), while for subsequent LTR or STR frames, DCT is applied to the prediction residual, as indicated in Fig. 2. In Section 3.1, we showed that linear reordering of MoCap data makes the energy concentrate at low frequencies, which in turn makes spatial transform more effective in compression. With such an arrangement, our system does not require spatial modeling of the marker configuration, which is necessarily application-specific. Instead of geometric constraints, we can exploit statistical redundancy of reordered data through spatial DCT.

3.4. Quantization

Upon transformation, DCT coefficients are quantized using a uniform mid-tread quantizer denoted by Q in Fig. 2. The reverse quantization block is denoted by IQ. The quantization step size (Δ) can be chosen to achieve the desired tradeoff between reconstruction quality and compression efficiency. Different step sizes can be chosen for LTR and STR frames. As in video compression, the step size for LTR frames will normally be lower than the step size for STR frames. This makes LTR frames of higher relative quality and ensures better prediction of subsequent STR frames.

In applications where the dynamic range of the signal is known in advance, the number of quantization bins B is usually determined as the ratio of the dynamic range and Δ. However, the dynamic range of MoCap data, as well as the dynamic range of DCT coefficients, may vary within a sequence. For example, if the subject moves away from the chosen origin of the 3D coordinate system, the x and y components of $\mathbf{f}[n]$ will tend to increase with n. Further, the dynamic range cannot be determined in advance in low-delay applications, since one would have to buffer and examine all data in order to find the minimum and maximum. Unless some precautions are made, with a fixed Δ, the number of quantization bins would vary, which would complicate the encoding of quantization bin indices. To mitigate this problem, we examined the dynamic range of DCT coefficients on a large set of MoCap data [13], and determined the appropriate number of bins for various values of Δ. That way, when a particular Δ is chosen, the encoder simply finds the appropriate number of bins B from a look-up table. A segment of this look-up table is shown in Table 1. The outermost quantization bins are left open on one side, i.e. the lowest quantization bin is $(-\infty, -(B-2)\Delta/2)$, and the highest quantization bin is $[(B-2)\Delta/2, \infty)$. Their reconstruction levels are $-(B-1)\Delta/2$ and $(B-1)\Delta/2$, respectively. If a particular DCT coefficient is too large or too small and falls into one of these bins, its quantization error may be larger than $\Delta/2$, which is known as "overload noise." However, with the look-up table designed on the dataset from [13], the occurrence of overload noise is so infrequent that it does not affect compression efficiency appreciably.

Table 1. Part of the look-up table with quantization step sizes (Δ) and the corresponding number of bins (B).

Δ	Number of bins (B)
≥ 100	25
33.3	49
16.7	73
10.0	97
7.7	121

3.5. Entropy coding

Entropy coding is the last stage of the encoder in Fig. 2. For this purpose, we use the range coder [14], [15], which is a particular implementation of adaptive arithmetic coding. The entropy coder maintains two histograms of quantization bin counts, one for LTR frames, the other for STR frames. Each histogram is updated with each encoding of an LTR or STR frame. Once the encoding of a particular frame (LTR or STR) is completed, the encoder buffer is flushed, but the histogram is carried over for encoding the next frame of the same type. That way, the encoder can adapt to the changing statistics of quantization bin occurrences in different types of frames. The STR-frame histogram is reset to uniform after each LTR frame. The decoder follows the same procedure to track the histogram evolution at the encoder.

4. EXPERIMENTAL RESULTS

To facilitate comparison with prior work, in particular the methods called SEG, GPCA, and PPCA in Section 2, we used four MoCap sequences from the CMU database [13]. These four test sequences were also used in [8]. Their description is summarized in Table 2. Each test sequence contains a number of frames with 3D locations of $m = 41$ markers in C3D format. The frame rate of each sequence is 120 fps. Each coordinate of each marker is represented by an 8-byte floating point number, so the total uncompressed file size is given by

$$\text{file size} = 3 \times 41 \times (\text{number of frames}) \times (8 \text{ bytes}).$$

Table 2. MoCap test sequences from [13].

Sequence	Type of motion	Total frames	Size (kbytes)
13_29	Jumping, …	4592	4413
85_12	Breakdance, …	4499	4324
86_02	Walk, squats, …	10590	10177
86_08	Walk, squats, …	9206	8847

In the field of MoCap compression, it is customary to measure compression efficiency in terms of the reconstruction error and the compression ratio. The compression ratio is simply the ratio of the uncompressed file size (Table 2) and the compressed file size. The reconstruction error is commonly measured by the so-called "distortion rate" d [8], defined as:

$$d = 100\frac{\left\|\mathbf{F}-\widetilde{\mathbf{F}}\right\|}{\left\|\mathbf{F}-E(\mathbf{F})\right\|}, \qquad (6)$$

where $\mathbf{F}$ is a $(3m) \times K$ matrix containing all the original frames $\mathbf{f}[n]$ as columns, K is the total number of frames, $\widetilde{\mathbf{F}}$ is the matrix containing all reconstructed frames, $E(\mathbf{F})$ represents the temporal average values of marker coordinates, and $\|\cdot\|$ denotes the Euclidean norm.

We compare several MoCap compression methods in terms of d and compression ratio. Methods called SEG, GPCA, and PPCA were described in Section 2. DPCM (Differential Pulse Code Modulation) is a simple predictive encoding method where each frame is predicted from the previous one, and the prediction residual is directly quantized and encoded without spatial transform. It is equivalent to one of the MPEG-4 BBA compression methods mentioned in Section 2, except that here it is applied to raw MoCap data rather than articulated human joints. DPCM+t_DCT_N for $N \in \{30, 60\}$ is similar to the other MPEG-4 BBA compression scheme from Section 2. This method follows up temporal DPCM by temporal DCT applied over N frames. Our method is labeled Hybrid_N, where $N \in \{30, 60\}$ denotes the period of LTR frames.

All methods are summarized in Table 3, which lists their encoding delay in frames. Encoding delay is the number of frames that the encoder has to buffer and process before starting to encode the first frame. The encoding delay of our method and that of DPCM is 0, because encoding of the frame can start as soon as the frame is captured. This makes these two methods particularly suitable for low-delay interactive applications such as online gaming. Meanwhile, the encoding delay of DPCM+t_DCT_N is $N - 1$, since the entire group of N frames needs to be buffered before encoding can start. SEG and PPCA have variable delay because they apply PCA over an adaptively segmented group of frames. The particular segmentation employed in [8] starts with a group of 200 frames and then increases it in steps of 10 as necessary. Hence the encoding delay of these two schemes is no less than 200. Finally, GPCA needs to process all frames in the sequence prior to encoding.

Table 3. Encoding delay (in frames) for various methods

Compression method	Encoding delay (frames)
SEG	Variable (≥ 200)
GPCA	All frames in the sequence
PPCA	Variable (≥ 200)
DPCM	0
DPCM+t_DCT_N	N–1
Hybrid_N	0

The results are shown in Figs. 5 – 8 for the sequences from Table 2. The results for SEG, GPCA, and PPCA are obtained from [8], where they were reported for only one compression ratio. Hence, we only show one point in each graph for each of these three methods. On the other hand, by varying the quantization step size Δ, we can obtain various compression ratios and reconstruction qualities for the remaining three methods including ours. Hence, DPCM, DPCM+t_DCT_N, and Hybrid_N are represented by curves, rather than single points.

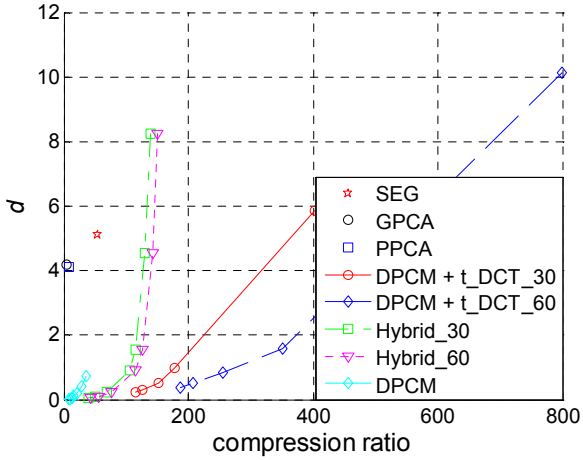

Fig. 5. Compression performance on MoCap sequence 13_29.

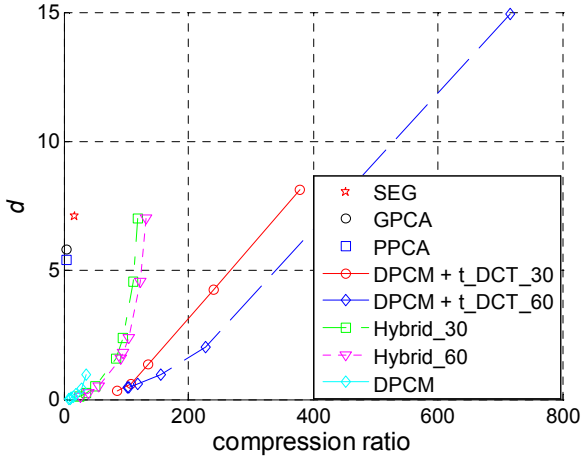

Fig. 6. Compression performance on MoCap sequence 85_12.

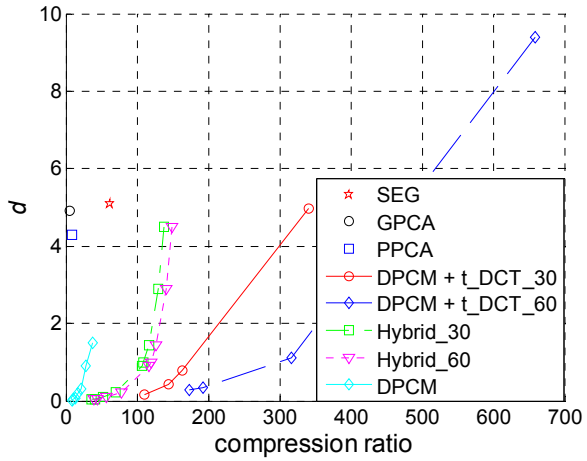

Fig. 7. Compression performance on MoCap sequence 86_02.

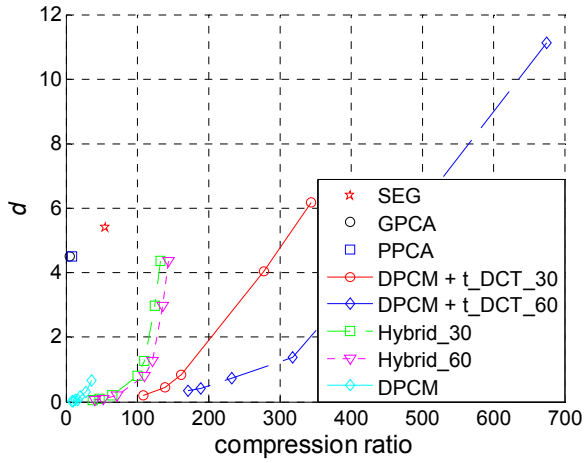

Fig. 8. Compression performance on MoCap sequence 86_08.

As seen in the results, DPCM+t_DCT_*N* has the best compression performance (highest compression ratio for a given reconstruction quality) among all the tested methods. As *N* increases, the compression efficiency increases, but the encoding delay increases as well. Even with $N = 30$ and sampling rate of 120 Hz, the encoding delay is equivalent to a quarter of a second, which is likely to be too large for interactive online gaming, considering that this is not the only source of delay in the overall end-to-end system. When $N = 1$, this scheme reduces to simple DPCM. In that case the encoding delay is eliminated, but the compression performance suffers. On the other hand, through the use of data reordering and spatial DCT, our scheme is able to achieve high compression efficiency and low delay simultaneously. All these three methods (DPCM, Hybrid_*N*, and DPCM+t_DCT_*N*) show considerably better reconstruction quality than SEG, GPCA, and PPCA over the range of compression ratios where they can be compared.

5. CONCLUSIONS

In this paper, we have demonstrated the potential of hybrid encoding structure (temporal prediction + spatial transform), which has evolved in video coding, for MoCap data compression. A simple linear reordering makes such a structure an efficient encoder for MoCap data. The results indicate that the hybrid structure offers both improved compression efficiency and lower encoding delay compared to several state-of-the-art methods.

6. REFERENCES

[1] M. Z. Patoli, M. Gkion, P. Newbury, and M. White, "Real time online motion capture for entertainment applications," *Proc. IEEE DIGITEL 2010*, pp. 139-145, Apr. 2010.

[2] A. Andreadis, A. Hemery, A. Antonakis, G. Gourdoglou, P. Mavridis, D. Christopoulos, and J. Karigiannis, "Real-time motion capture technology on a live theatrical performance with computer generated scenery," *Proc. IEEE PCI'10*, Tripolis, Greece, Sept. 2010.

[3] S. Subramanian, L. A. Knaut, C. Beaudoin, B. J. McFadyen, A. G. Feldman, and M. F. Levin, "Virtual reality environments for post-stroke arm rehabilitation," *J. Neuroengineering Rehabil.* vol. 4, no. 20, Jun. 2007.

[4] CMU Marker Placement Guide [Online] Available: http://mocap.cs.cmu.edu

[5] M. Preda, B. Jovanova, I. Arsov, and F. Preteux, "Optimized MPEG-4 animation encoder for motion capture data," *Proc. Web 3D '07*, pp. 181-190, Apr. 2007.

[6] B. Jovanova, M. Preda, and F. Preteux, "MPEG-4 Part 25: A generic model for 3D graphics compression," *Proc. 3DTV Conference (3DTV-CON)*, pp. 101-104, May 2008.

[7] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computers & Graphics*, vol. 28, no. 1, pp. 25-34, 2004.

[8] G. Liu and L. McMillan, "Segment-based human motion compression," *Proc. SCA'06*, pp. 127-135, Sep. 2006.

[9] Q. Gu, J. Peng, and Z. Deng, "Compression of human motion capture data using motion pattern indexing," *Computer Graphics Forum*, vol. 28, no. 1, pp. 1-12, Mar. 2009.

[10] Y. Lin and M. McCool, "Nonuniform segment-based compression of motion capture data," *Proc. ISVC'07*, pp. 56-65, Nov. 2007.

[11] M. Tournier, X. Wu, N. Courty, E. Arnaud, and L. Reveret, "Motion compression using principal geodesics analysis," *Computer Graphics Forum*, vol. 28, no. 2, pp. 355-364, 2009.

[12] S. Chattopadhyay, S. M. Bhandarkar, and K. Li, "Human motion capture data compression by model-based indexing: A power aware approach," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, no. 1, pp. 5-14, Jan. 2007.

[13] The CMU Motion of Body (MoBo) Database [Online], Available: http://mocap.cs.cmu.edu

[14] M. Servais, "Range coding MATLAB source code," http://www.ee.surrey.ac.uk/CVSSP/VMRG/hdtv/code.htm

[15] G. N. N. Martin, "Range encoding: An algorithm for removing redundancy from a digitised message," *Proc. Video and Data Recording Conference*, Jul. 1979.