Multimed Tools Appl
DOI 10.1007/s11042-016-3944-7

Multiresolution coding of motion capture data for real-time multimedia applications

Murtaza Ali Khan[1]

Received: 25 January 2016 / Revised: 20 July 2016 / Accepted: 6 September 2016

Abstract In this work, we present a novel and efficient method for coding of motion capture (MoCap) data obtained from recording of human actions. MoCap data is represented as hierarchies of joints and parameterized by translation and rotation of channels or degree-of-freedom (DOF) in a sequence of frames as a function of time. The proposed method approximates the MoCap data of each channel independently using multiresolution discrete wavelet transform (DWT). In order to improve the performance, a skeleton dependent quantization of wavelet coefficients is used that computes a local threshold of each joint based on a global threshold and depth of the joint in the hierarchy of joints. The multiresolution DWT based coding allows to control the bitrate and to decode (reconstruct) the single instance of compressed MoCap data into multiple instances from high to low resolution (quality). We also compared the performance of proposed method with recent and state of the art methods. The proposed method yields smaller storage space and faster encoding time. The method is well suitable for real-time multimedia applications due to its low time and space requirements.

Keywords Multimedia · Coding · Compression · MoCap data · Wavelet transform · Multiresolution

1 Introduction

Compression techniques have played an important role in the world of multimedia and telecommunication systems, where the storage space and bandwidth are valuable resources. Motion capture (MoCap) data is extensively used in many multimedia applications such as

✉ Murtaza Ali Khan
makkhan@uqu.edu.sa

[1] College of Computer and Information Systems, Umm Al-Qura University, Makkah Al-Mukarramah, Saudi Arabia

Published online: 16 September 2016

in entertainment e.g., dance recording and training [5], in gaming e.g., cloud gaming [3], online gaming [18], etc., streaming animation [19] and online physical rehabilitation [20], [17] et al. Due to big size of MoCap data, a compact storage, multiresolution representation, an efficient transmission and real-time coding algorithm is crucial. In order to support real-time interactions, most online multimedia applications have low bit-rate requirements.

In this work, we focused on multiresolution coding of MoCap data obtained from recording the actions of humans and used for multimedia applications. Multiresolution representation of MoCap data allows scalable media streaming i.e., coding technique which fragments a single high-quality media-stream to n sub-streams that progressively increase quality [22]. Conventional video data is generally recorded at 25–30 frames per second (fps), while MoCap data is recorded at higher frame rate, typically 120–240 fps. The high frame rate and limitations of human visual system make it feasible to use coding methods that allow some loss of information. The coding method presented in this paper is based on Discrete Wavelet Transform (DWT) and suitable for lossy coding. The contributions of this research work are:

1. Multiresolution coding that allows to control the bitrate and to decode the single instance of compressed MoCap data into multiple instances from high to low resolution (quality).
2. Using simple and efficient criteria of wavelet coefficients thresholding using joints' depth information.
3. Optimized and fast encoding and decoding algorithms suitable for real-time multimedia applications.
4. Better performance in terms of time and space compared to other methods.

Skeletal animation is a technique in computer graphics in which a character is represented in two parts. (1) A set of bones called skeleton. Bones are connected by joints. (2) A polygon mesh model of the character called skin is attached to the skeleton. When a skeleton is animated using MoCap data, it animates the skin and consequently the whole character animates. Figure 1 shows a character of a humanoid with skin and skeleton. A bone represents the smallest segment that is subject to individual translation and rotation during the animation. Joints are usually organized in a tree like hierarchical structure as illustrated in Fig. 2. Consequently, when a joint moves, its connected joints lower in the hierarchy move too. For example, the wrist is connected to the hand which is connected to fingers, then as the wrist moves, the hand and fingers also move. The ability of a joint to move is referred as a channel or degree-of-freedom (DOF). An individual joint usually has between one and six DOFs, but all together, a detailed character may have more than a hundred DOFs in the entire skeleton. Motion data of a joint in a sequence of frames can be considered as a function of time. Figure 3 shows rotational data of *right-foot* joint in 148 frames.

Let assume that recorded MoCap data consists of m channels and each channel has translation/rotation values in a series of n frames. Then for efficient manipulation, MoCap data can be stored in a matrix of size $n \times m$ as written in the (1).

$$X = \begin{bmatrix} \theta_{11} & \theta_{12} & \dots & \theta_{1m} \\ \theta_{21} & \theta_{22} & \dots & \theta_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \theta_{n1} & \theta_{n2} & \dots & \theta_{nm} \end{bmatrix}. \tag{1}$$

Each row of the matrix X in (1) corresponds to motion data of all the channels for a single specific frame, i.e., $X[i, 1 \dots m] = \{\theta_{i1}, \theta_{i2}, \dots, \theta_{im}\}$ contains the motion data of i^{th} frame

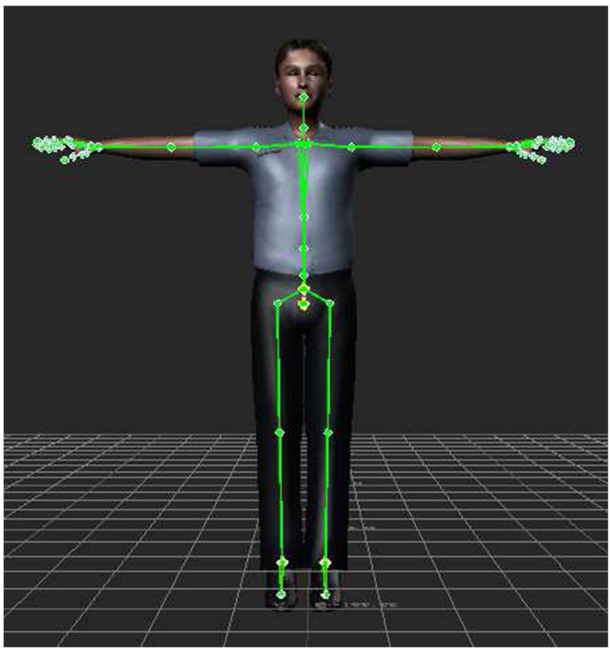


Fig. 1 A character of humanoid with skin and skeleton

for channels $(1 \ldots m)$. Each column of matrix X corresponds to motion data of a specific channel for n frames, i.e., and $X[1 \ldots n, j] = \{\theta_{1j}, \theta_{2j}, \ldots, \theta_{nj}\}$ contains the motion data of j^{th} channel for frames $(1 \ldots n)$.

2 Related work

An extensive research material on the coding of multimedia data particularly image and video data can be found, while the area of MoCap data coding is flourishing. Techniques based on principal component analysis (PCA) [25] et al., spline/curve [14] et al., discrete cosine transform (DCT) [23, 26] et al., and discrete wavelet transform (DWT) [6, 7, 9, 11] et al., are well known for image and video data coding. These techniques can be adapted and applied to compress motion capture data.

An algorithm to compress the MoCap data using quadratic Bézier curve is proposed by the author [15]. The method exploits the temporal coherence of motion data by treating rotational and translation variations of a joint in a sequence of frames as input points in N-dimensional Euclidean space. The input points are parameterized and approximated using quadratic Bézier curve least square fitting. In the Section 4, we compared performance of our method with the method of [15].

Another recent work that uses discrete cosine transform (DCT) to compress the MoCap data is proposed by [12]. The method segments the MoCap sequences into clips represented as 2D matrices. The rows of a MoCap matrix are smooth since each channel has a smooth trajectory. The method takes advantage of this smooth variations of channels and computes a

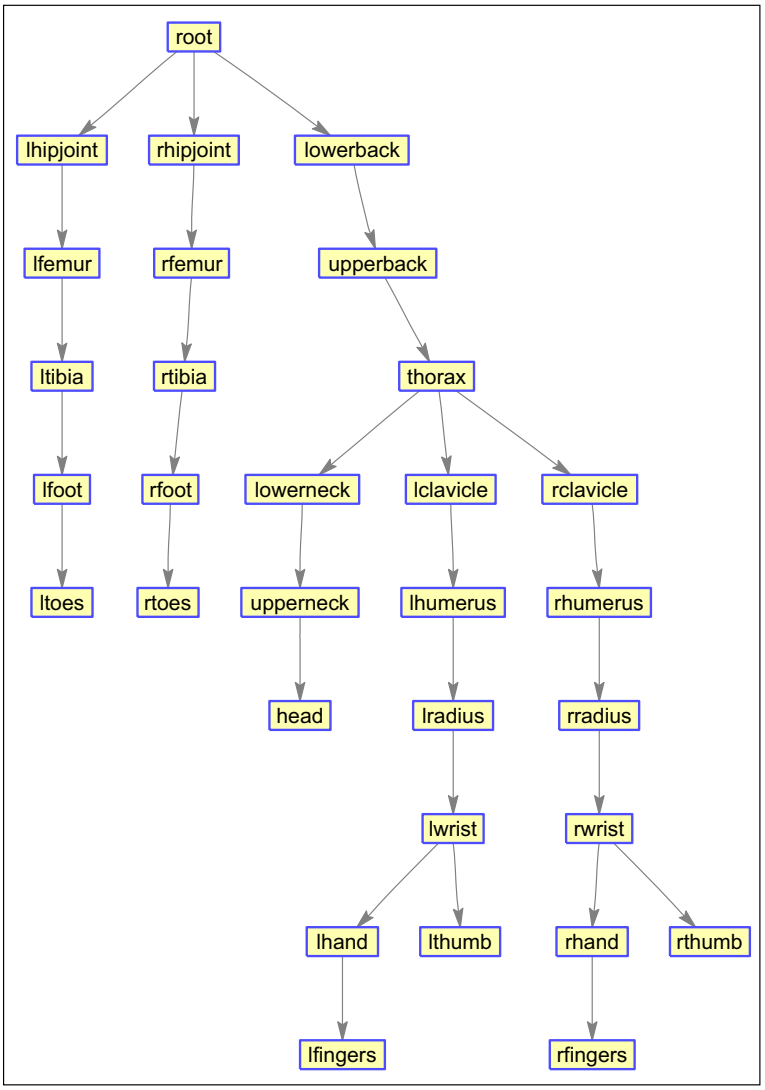


Fig. 2 Hierarchy of joints

set of MoCap data dependent orthogonal bases (DCT) to transform the matrices to frequency domain. Finally entropy coding is applied to quantized coefficients and the bases. In the Section 4, we compared performance of our method with the method of [12].

Like image and video data, discrete wavelet transform (DWT) proved out to be a successful tool for compressing MoCap data as described by [2, 4, 8] et al. Cheng [4] proposed a scalable method for compressing MoCap data using Linear Set Partitioning In Hierarchical Trees (LSPIHT) applied on individual channels of MoCap clips. In his method the bit rate allocation to channels is based on perceptual factors. The method of [8] combines the effects of relative bone length and the variation of each channel to find the relative importance factor of each channel then apply wavelet transform based on this factor. However,

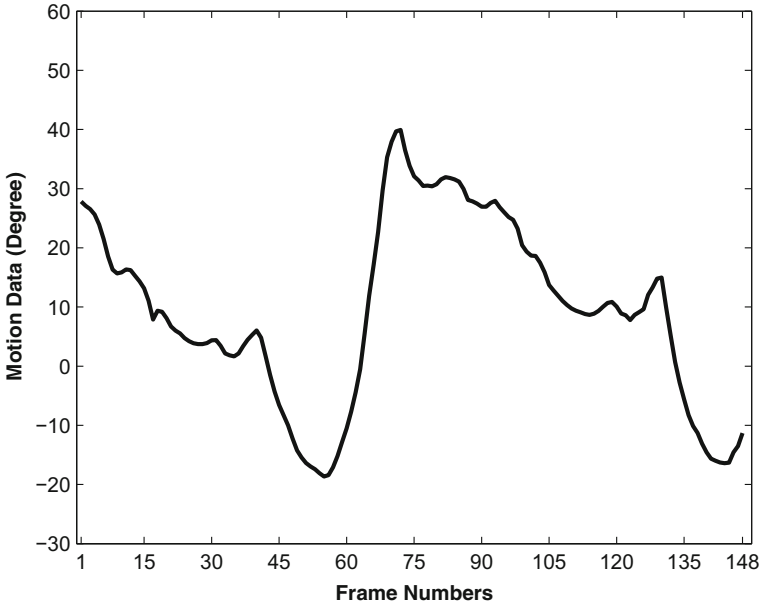


Fig. 3 MoCap data (rotation about X-axis) of *right-foot* in 148 frames

finding variation of each channel is computationally very expensive and requires analyzing all the MoCap data (possibly millions of values). On the other hand, our method analyzes very small size skeleton hierarchy (only $\approx$ 60–296 values) rather than MoCap data and computes depth of all joints. Further, editing MoCap data (common practice by animators to remove noise and foot skating) does not affect our method, because hierarchy of joints does not change. On the contrary, the method of [8] requires to compute again the variations of channels after editing in MoCap data. Another DWT based method to compress the motion data is of [2]. This method uses cubic interpolating bi-orthogonal wavelet basis to find the temporal coherence of channels data and proposed a heuristic for optimized wavelet coefficient selection. However, the heuristic is computationally expensive and does not guarantee to yield optimized results.

Several authors [21, 24] et al., used principal component analysis (PCA) to compress the MoCap data. In the method of [24], initially poses manifold are approximated using principal geodesics analysis (a generalization of PCA) in the configuration space of the skeleton. Then approximate manifold is searched for poses by matching end-effectors constraints using an iterative minimization algorithm. The compression is achieved by only storing the approximate manifold parameterization along with the end-effectors and root joint trajectories. Poses are reconstructed using the inverse kinematics algorithm given the end-effectors trajectories.

A method that exploits both inter and intra clip correlations to compresses MoCap database using Bézier curves, PCA and DCT is proposed by [1]. The method is not only complex but it uses virtual markers as an internal representation of motion data which requires conversion to and from this representation; consequently increase the burden on coding process and not practical for a real-time animation system. A method to compress human motion capture data based on hierarchical structure construction and motion

pattern indexing is presented by [10]. The method first organizes 3D motion markers into a hierarchy of human body nodes. Then identify motion patterns that form a database for each meaningful body part. Finally, the motion sequence corresponding to each body part is extracted and coded. Both the methods of [1] and [10] are suitable for large databases of motion clips and exploits the correlation between similar clips. But if the user wants to add/delete individual clips (a common practice) to/from the database then intra-clip similarity will no longer be valid and it is not clear how the system will work. Our method process motion clips individually and independently further it is computationally more efficient than the methods of [1] and [10].

3 Coding of MoCap data by DWT

The Discrete Wavelet Transform (DWT) provides powerful insight into a signal's spatial and frequency characteristics. The multiresolution DWT decomposes a signal into approximation and detail components. The approximation component is then recursively decomposed into approximation and detail components at successively coarser scales (resolutions). The wavelet coding is based on the concept that the regular signal component can be accurately approximated using a small number of approximation coefficients (at a suitably chosen level) and some of the detail coefficients. For motion capture data, wavelet transform works by decomposing the rotational/translation data of each channel (dimension) separately. We will describe the method of coding of a single channel data by discrete wavelet transform.

As an input to the algorithm user specifies following three parameters: (1) A global threshold T_g to truncate detail coefficients, (2) A quantization value Q to quantize approximation and detail coefficients, and (3) Maximum number of wavelet decomposition levels N. From the matrix X (see (1)), data of a channel is extracted (a single column). Let the set of n extracted values of a channel are denoted as $I = \{x_1, x_2, \ldots, x_n\}$. For example, for the *right-foot* joint (j^{th} channel), the n rotational values (in n frames) along Y-axis are $X[1 \ldots n, j] = \{\theta_{1j}, \theta_{2j}, \ldots, \theta_{nj}\} = \{x_1, x_2, \ldots, x_n\} = I$. The set I is considered as a signal $x(t)$. DWT up to level N is applied to I, this yields a set of approximation coefficients cA, a set of detail coefficients cD, and a book-keeping vector V. A local threshold T_l is computed based on the global threshold T_g and the depth d of the joint the channel I belongs. Threshold T_l is applied to detail coefficients of all the levels, i.e., all the detail coefficients less than or equal to T_l are set to zero. Quantization is applied to approximation and detail coefficients, i.e., cA and cD are rounded to Q decimal places. After thresholding and quantization, run-length encoding is applied to detail coefficients as they have long chains of zeros. Then deflate encoding is applied to approximation and detail coefficients. Finally the encoded data i.e., approximation coefficients, detail coefficients and book-keeping vector are saved in the decoded file.

The same procedure is applied to motion data of all the channels. Figure 4 shows Daubechies wavelet approximation of rotational data of a channel in 148 frames. The procedure of encoding of motion data X is formally described in Algorithm 1. The procedure to decode (reconstruct) the approximated motion data $\widetilde{X}$ is simple and it is described in Algorithm 2. The size of book-keeping vector V is $N + 2$ and it contains information about number of approximation coefficients at level N (1 value), number of detail coefficients at each level (N values), and the size of input data (1 value). Note that all the channels have same V, therefore, we have to save V only once.

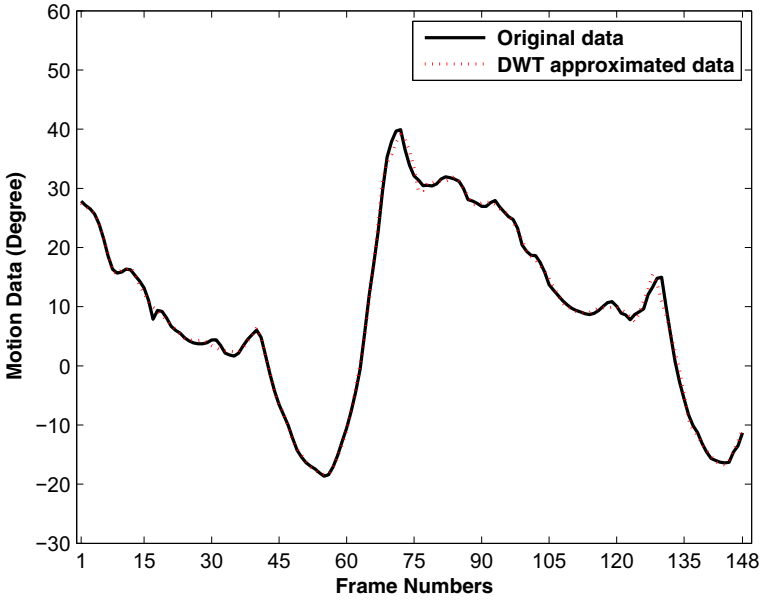


Fig. 4 Wavelet approximation of MoCap data of *right-foot* in 148 frames, threshold, $T_g = 1$ and level of decomposition, $N = 3$. Detail coefficients of level 1 and 2 are set to 0

Algorithm 1 Encoding of MoCap data

Require: X, T_g, Q, N
for $k = 1$ to m **do**
  $I_k = X[1 \ldots n, k]$
  $[cA_k, cD_k, V] = ApplyDWT(I_k, N)$
  $T_l = \sqrt{log_2(d_k + 2)} \times T_g$
  $cD_k = ApplyThreshold(cD_k, T_l)$
  $[cA_k, cD_k] = ApplyQuantization(cA_k, cD_k, Q)$
  $cD_k = ApplyRunlengthEnoding(cD_k)$
  $[cA_k, cD_k] = ApplyDeflateEnoding(cA_k, cD_k)$
  $Save(cA_k, cD_k)$
end for
$Save(V)$

Algorithm 2 Decoding of MoCap data

Require: $cA_1 \ldots cA_k, cD_1 \ldots cD_k, V$
for $k = 1$ to m **do**
  $[cA_k, cD_k] = ApplyInflateDecoding(cA_k, cD_k)$
  $cD_k = ApplyRunlengthDecoding(cD_k)$
  $\widetilde{I_k} = ApplyInverseDWT(cA_k, cD_k, V)$
  $\widetilde{X}[1 \ldots n, k] = \widetilde{I_k}$
end for

Table 1 Depth of joints

Normalized depth	Names of joints
0	root
0.111	lhipjoint, rhipjoint, lowerback
0.222	lfemur, rfemur, upperback
0.333	ltibia, rtibia, thorax
0.444	lfoot, rfoot, lowerneck, lclavicle, rclavicle
0.556	ltoes, rtoes, upperneck, lhumerus, rhumerus
0.667	head, lradius, rradius
0.778	lwrist, rwrist
0.889	lhand, rhand, lthumb, rthumb
1	lfingers, rfingers

In MoCap data, joints higher in the skeleton hierarchy (e.g., hips) are more important because they influence the movements of joints lower in the hierarchy. Therefore, in order to apply the threshold to detail coefficients, we include the depth factor. We compute the normalized depth value of each joint in the skeleton hierarchy. The root joint has depth value 0 (highest in the hierarchy) while lowest joint(s) (e.g., fingers) have depth value 1. All the channels (DOFs) of a particular joint have same depth value. As a input to encoding algorithm, the user specifies the global wavelet threshold T_g, while the local threshold T_l of every joint is based on its depth d. T_l is computed using (2).

$$T_l = \sqrt{log_2(d+2)} \times T_g. \tag{2}$$

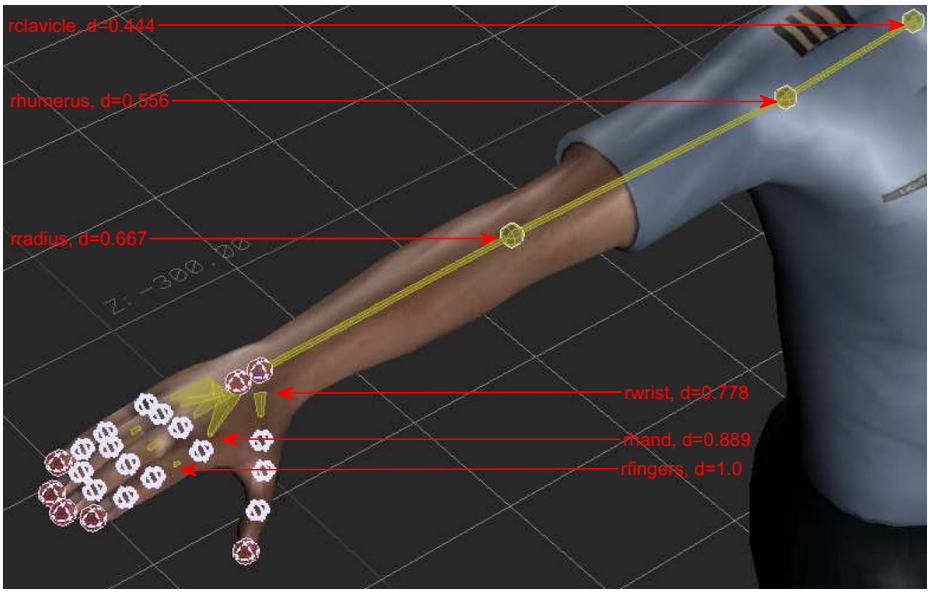


Fig. 5 Hierarchy of rclavicle, rhumerus, rradius, rwrist, rhand, and rfingers joints with their normalized depth values

Table 2 Environment of experiments

Hardware	Intel Quad Core i5-4690K , 3.5GHz, 8GB RAM
Operating system	Windows 10, 64 bit
Programming language	MATLAB 8 R2012b
Animation testing	MotionBuilder 2013
Processing MoCap files	Mocap toolbox for MATLAB [16]

A joint higher in the skeleton hierarchy has smaller value of d and consequently get lower value of threshold T_l by (2). Lower value of T_l causes fewer detail coefficients set to zero which mean lesser loss of information and better reconstruction. Logarithm and square root in (2) are used so that threshold T_l changes with low-rate from level to level. Otherwise, the T_l of lower level joints will be significantly higher compare to T_l of higher level joints which causes noticeable artifacts in the reconstructed animations. The $+2$ factor in (2) ensures that T_l of root joint ($d = 0$) is not 0. Without $+2$ factor, T_l of root joint would have been always 0 regardless the value of T_g.

Normalized depth values of all the joints are given in Table 1. Hierarchy of rclavicle, rhumerus, rradius, rwrist, rhand, and rfingers joints (along with their normalized depth values) is shown in Fig. 5.

4 Simulations and results

An extensive simulation analysis is carried out to test the effectiveness of proposed method. Table 2 provides details of environment of simulations. Details of representative input animations used in the simulations are listed in Table 3. We compressed the MoCap data of each animation listed in Table 3 using our method and compared the results with methods of Khan [15] and Hou [12]. The statistics of [15] method is otained by our implementation of method and its simulations. The statistics Hou's method is taken from Table 4 of [12]. The comparison is based on following parameters:

1. Compression ratio (CR), i.e., ratio between size of original motion data file f_o (i.e., ACM file) and size of compressed motion data file f_c:

$$CR = \frac{f_o}{f_c}. \tag{3}$$

2. The measure of error (distortion) is average Euclidean distance between original data and reconstructed approximated data [12]. The equation of distortion can be written as:

$$\varepsilon_d = \frac{1}{rn} \sum_{i=1}^{r} \sum_{j=0}^{n} \sqrt{(p_{i,j} - \widehat{p}_{i,j}) - (p_{i,j} - \widehat{p}_{i,j})}, \tag{4}$$

where r and n are the numbers of joints and frames respectively, $p_{i,j} = [x_{i,j}, y_{i,j}, z_{i,j}]$ and $\widehat{p}_{i,j} = [\widehat{x}_{i,j}, \widehat{y}_{i,j}, \widehat{z}_{i,j}]$ are original and reconstructed 3D coordinates of the i^{th} channel in the j^{th} frame respectively.

It is desirable to have high value of compression ratio (CR) and low value of error (ε_d). The threshold parameter (T_g) to control the distortion (ε_d) of our method is not related to distortion controlling parameters of comparative methods ([15] and [12]). Therefore, it is

Table 3 Details of Input Animations. Frame rate 120-fps, file format ASF/AMC

Seq. no.	Channel count	Joint count	Frame count	Animation description
15_04	62	31	22549	Wash windows, paint, hand signals, dance, dive, twist, boxing
17_08	62	31	6179	Muscular, heavyset person's walk
17_10	62	31	2783	Boxing
85_12	62	31	4499	Jumps, flips, breakdance

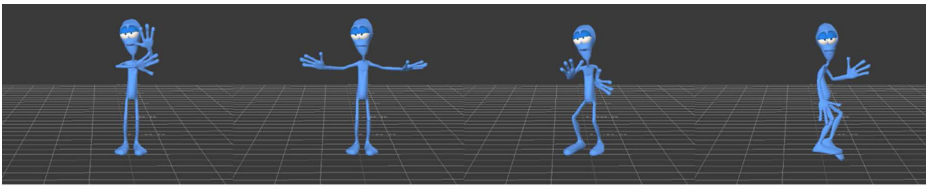
very difficult to get the same value of ε_d or CR for all the methods. Therefore, in order to evaluate comparative performance, the best we can do is to vary the value of threshold parameter (T_g) of our method and obtain the multiple values of (ε_d) and CR and select those values at which CR of our method is closest to CR of of Khan [15] and Hou [12] methods. Then at those values of CR we can compare the distortion (ε_d) of all the three methods. Such comparative results are shown in Table 4. Figure 6 shows the four reconstructed frames of compressed sequences. These frames are obtained by decoding of compressed data of proposed method.

It can be observed from the Table 4 that the distortion (ε_d) of our method is significantly less than other methods for almost equal value of compression ratio (CR) in all the cases. Conversely, we also found that compression ratio of our method is higher compared to other methods for the almost equal values of distortion (ε_d). The top row of Fig. 7 shows comression ratio (CR) vs. distortion (ε_d) curves for all the sequences. The bottom row of Fig. 7 shows comression ratio (CR) vs. global threshold (T_g) curves for all the sequences.

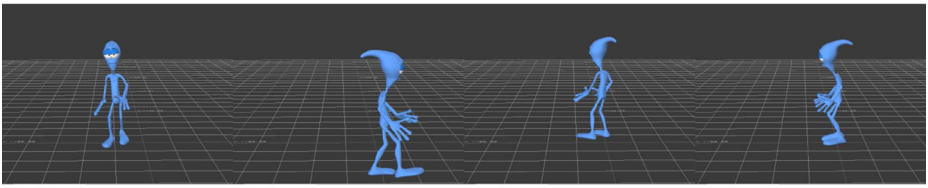
Compression ratio (CR) and distorting error (ε_d) are independent of software & hardware (Operating System, CPU, RAM, etc.) configurations used in the simulations. But encoding and decoding times are strongly software & hardware dependent. However, for the sake of completion, we listed the encoding (E_{fps}) and decoding (D_{fps}) of all the methods in Table 4. The method of [15] is quadratic in nature (quadratic Bézier curve), while method of [12] (DCT based) and our proposed method (DWT based) are linear. Therefore, as expected, the running time performance of [15] is lowest, while performance of our method and [12] are quite comparative. In two instances of encoding time (E_{fps}) the method of [12] performs better and in two instances our method has better E_{fps}. For decoding time (D_{fps}), in three instances our method performs better and in one instance method of [12] has better decoding rate. Interestingly the sequence 15_04 has largest number of frames (22549) and motion

Table 4 Efficiency compression of all the methods for a single observation of each animation. In each row, compression ratio (CR) of all the method is almost same, the distortion (ε_d) signifies the comparative performance of each method. E_{fps} and D_{fps} are encoding and decoding rates in frames per seconds (fps) respectively

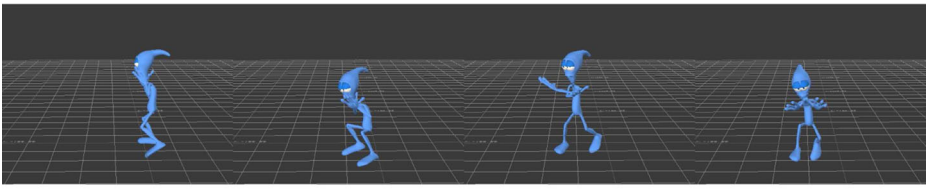
Seq.	Khan's method [15]				Hou's method [12]				Our proposed method			
No.	CR	ε_d	E_{fps}	D_{fps}	CR	ε_d	E_{fps}	D_{fps}	CR	ε_d	E_{fps}	D_{fps}
15_04	115.8	0.90	1576	18894	114.1	0.29	169567	281478	115.9	0.104	158590	292131
17_08	93.1	1.13	1476	12168	93.8	0.37	69856	91729	94.7	0.133	69930	91773
17_10	63.2	1.73	1030	10794	63.2	1.07	32348	40239	64.5	0.204	35746	41132
85_12	60.0	1.87	948	10470	59.3	0.84	52269	68546	60.5	0.225	47782	65996



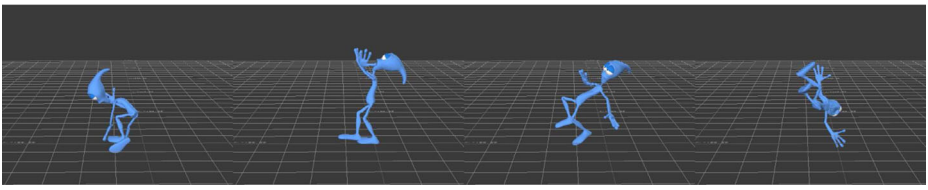
(a) Reconstructed frames: 3850, 6419, 13559, 16485 of 15_04 sequence. (CR, ε_d, T_g)=(115.9, 0.104, 4.4).



(b) Reconstructed frames: 200, 800, 1448, 2032 of 17_08 sequence. (CR, ε_d, T_g)=(94.7, 0.133, 4.8).



(c) Reconstructed frames: 137, 724, 853, 1444 of 17_10 sequence. (CR, ε_d, T_g)=(64.5, 0.204, 12).



(d) Reconstructed frames: 800, 1108, 2338, 2610 of 85_12 sequence. (CR, ε_d, T_g)=(60.5, 0.225, 12).

Fig. 6 Reconstructed frames. These frames are obtained by decoding of compressed data of proposed method

activity of character is slowest in this animation. Consequently, it has highest encoding and decoding rates. Most of the animation applications (e.g.,games), not only require motion data but other elements of scene and frame construction such as 3D characters, objects, lights, etc. Therefore, for real-time applications the decoding rate must be sufficiently higher than animation playback rate (120 fps). It can be observe from Table 4 that decoding rate of our method is very fast, i.e., in the range of tens of thousands of frames per second (fps).

5 Discussion

5.1 Multiresolution representation of MoCap data

An important characteristic of DWT is to represent a signal at multiresolution. The single instance of MoCap data compressed using DWT can be decoded (reconstructed) from high

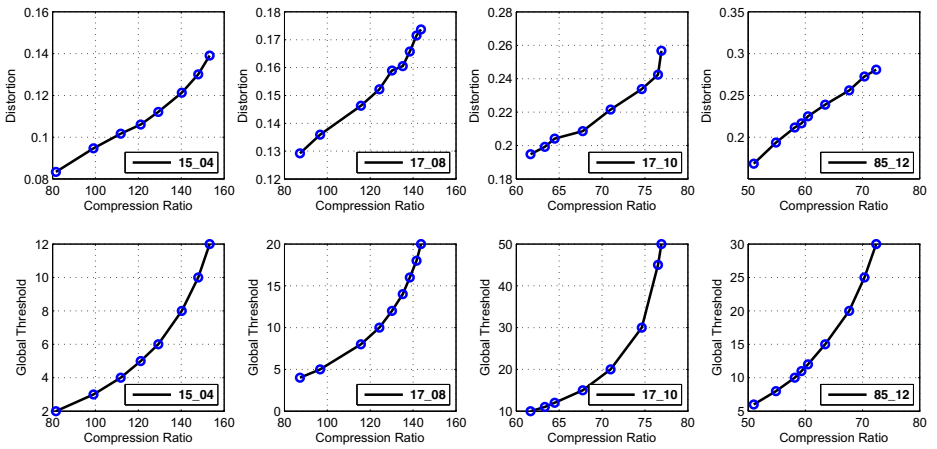


Fig. 7 Top: Comression ratio (CR) vs. distortion (ε_d) curves for all the sequences. Bottom: Comression ratio (CR) vs. Global Threshold (T_g) curves for all the sequences

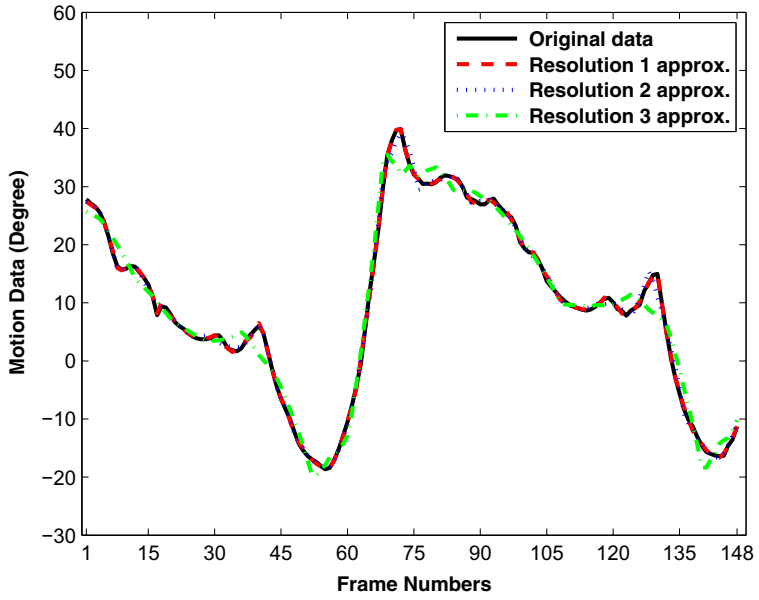
to low resolution (quality) by keeping the approximation coefficients intact while thresholding the detail coefficients to zero up to desire level. For example, the highest resolution animation requires to use all the detail coefficients, the next higher resolution animation requires to threshold detail coefficients of level 1 to zero, the next higher resolution animation requires to threshold detail coefficients of level 1 and 2 to zero, so on. Figure 8 shows multiresolution DWT approximation of MoCap data of a single channel. Figure 8a shows approximation of frames 1 to 148. Figure 8 shows enlarge view of approximation of frames 76 to 85. It is evident from this figure and given mean square error (MSE) statistics that as detail coefficients of more levels are set to zero, the accuracy of approximation decreases. However, lesser resolution approximation yields higher compression ratio due to longer chains of zeros.

5.2 Controlling bitrate

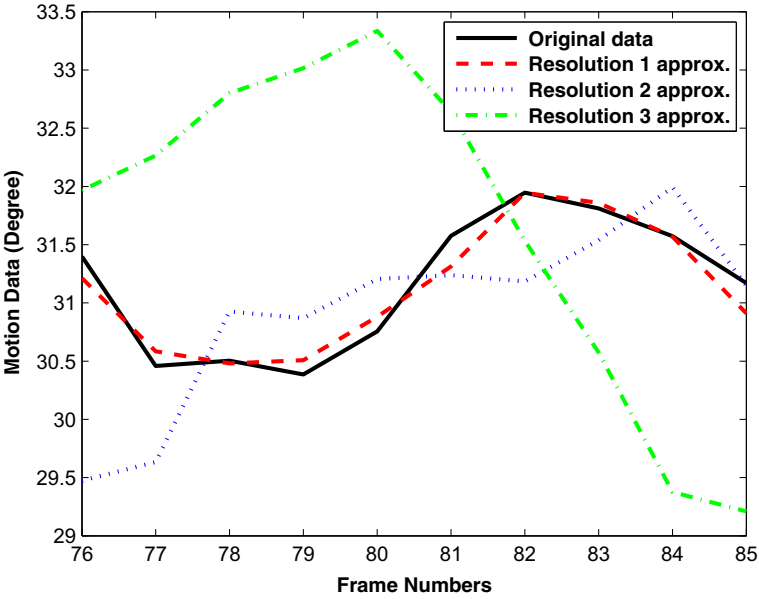
Multiresolution wavelet representation of MoCap data is helpful to control the bitrate. Consider the case of real-time online animation games using MoCap data where multiple players (clients) are connected to the server where the MoCap data is stored. The server can send the different resolutions bitstreams to different players based on their bandwidth. Obviously, low resolution representation of animation uses fewer bits and thus need less bandwidth. Multiresolution representation can also control the bitrate of an offline animation by decoding only low resolution data, i.e., using only approximation coefficients and discarding some or all of the detail coefficients.

5.3 Computational and space efficiency

Discrete wavelet transform is a linear transformation. Therefore, the proposed method is computationally very efficient. The fast encoding/decoding speed ensures that decoded MoCap data is available to rendering engine which renders and playback actual image frames including objects, light, texture, etc. The proposed method codes each channel of



(a) Frames 1 to 148



(b) Frames 76 to 85

Fig. 8 Multiresolution DWT approximation of MoCap data of a single channel. Resolution 1: all the detail coefficients are used ($MSE = 0.10$). Resolution 2: detail coefficients of level 1 and 2 are set to zero ($MSE = 0.66$). Resolution 3: detail coefficients of level 1, 2, and 3 are set to zero ($MSE = 4.69$). Figure 8a shows approximation of frames 1 to 148. Figure 8b shows enlarge view of approximation of frames 76 to 85

MoCap data individually and independently of other channels which is very efficient in terms of space requirement.

6 Future work

To impose constraints on environmental contacts, such as feet joint, so that they do not dig below or hang above the ground is a typical issue. Feet above or below the contact floor may cause noticeable artifacts at large value of wavelet threshold. One remedy of this issue is to save feet data using lossless encoding or use very low value of threshold for all the channels of foot joint. An alternative solution is to correct the feet contacts during animation playback using inverse kinematics. A technique to automatically detect and correct the foot skate is proposed by [13]. This method can be applied after reconstruction of approximated motion data. However, we did not implement this foot correction method and left it as a possible future work.

7 Conclusion

We described an efficient method for coding of motion capture (MoCap) data using multiresolution discrete wavelet transform (DWT). The method encodes the MoCap data of each channel independently using thresholding of wavelet coefficients based on depth of the joint (channel) in the hierarchical skeleton. Multiresolution characteristic of DWT allows reconstructing multiple instances of varying quality MoCap data from the single compressed bitstream. The multiresolution wavelet representation also facilitates to control the bitrate. Practically the method can encode and decode the MoCap data in real-time. Simulation results show that the method performs better than state of the art methods [15] and [12]. The subjective quality of reconstructed animations by the proposed method is also very good. Due to high computational efficiency and low space requirement, the proposed algorithm is well suitable for compressing MoCap data for skeletal animation and games in interactive and multimedia applications.

References

1. Arikan O (2006) Compression of motion capture databases. ACM Trans Graph 25(3):890–897. doi:10.1145/1141911.1141971
2. Beaudoin P, Poulin P, van de Panne M (2007) Adapting wavelet compression to human motion capture clips, pp 313–318
3. Cai W, Leung VCM, Hu L (2014) A cloudlet-assisted multiplayer cloud gaming system. MONET 19(2):144–152. doi:10.1007/s11036-013-0485-4
4. Cheng I, Firouzmanesh A, Basu A (2015) Perceptually motivated lspiht for motion capture data compression. Comput Graph 51(C):1–7. doi:10.1016/j.cag.2015.05.002
5. Choensawat W, Nakamura M, Hachimura K (2014) Genlaban: a tool for generating labanotation from motion capture data. Multimedia Tools Appl 74(23):10,823–10,846. doi:10.1007/s11042-014-2209-6
6. Deng C, Lin W, Cai J (2012) Content-based image compression for arbitrary-resolution display devices. IEEE Trans Multimedia 14(4):1127–1139. doi:10.1109/TMM.2012.2191270
7. DeVore R, Jawerth B, Lucier B (1992) Image compression through wavelet transform coding. IEEE Trans Inf Theory 38(2):719–746. doi:10.1109/18.119733
8. Firouzmanesh A, Cheng I, Basu A (2011) Perceptually guided fast compression of 3-d motion capture data. IEEE Trans Multimedia 13(4):829–834. doi:10.1109/TMM.2011.2129497

9. Garbas JU, Pesquet-Popescu B, Kaup A (2011) Methods and tools for wavelet-based scalable multiview video coding. IEEE Trans Circuits Syst Video Technol 21(2):113–126. doi:10.1109/TCSVT.2011.2105552
10. Gu Q, Peng J, Deng Z (2009) Compression of human motion capture data using motion pattern indexing. Comput Graph Forum 28(1):1–12
11. Hachicha W, Kaaniche M, Beghdadi A, Cheikh F (2015) Efficient inter-view bit allocation methods for stereo image coding. IEEE Trans Multimedia 17(6):765–777. doi:10.1109/TMM.2015.2417099
12. Hou J, Chau LP, Magnenat-Thalmann N, He Y (2015) Human motion capture data tailored transform coding. IEEE Trans Vis Comput Graph 21(7):848–859. doi:10.1109/TVCG.2015.2403328
13. Ikemoto L, Arikan O, Forsyth D (2006) Knowing when to put your foot down. In: I3D '06: Proceedings of the 2006 symposium on interactive 3D graphics and games. ACM, pp 49–53. doi:10.1145/1111411.1111420
14. Khan MA (2012) A new method for video data compression by quadratic bézier curve fitting. SIViP 6:19–24. doi:10.1007/s11760-010-0165-9
15. Khan MA (2016) An efficient algorithm for compression of motion capture signal using multidimensional quadratic bézier curve break-and-fit method. Multidim Syst Signal Process 27(1):121–143. doi:10.1007/s11045-014-0293-4
16. Lawrence N Mocap toolbox for matlab. Available on-line at http://www.cs.man.ac.uk/neill/mocap (2011)
17. Lin JS, Kulic D (2014) Online segmentation of human motion for automated rehabilitation exercise analysis. IEEE Trans Neural Syst Rehabil Eng 22(1):168–180. doi:10.1109/TNSRE.2013.2259640
18. Liu L, Jones A, Antonopoulos N, Ding Z, Zhan Y (2015) Performance evaluation and simulation of peer-to-peer protocols for massively multiplayer online games. Multimedia Tools Appl 74(8):2763–2780. doi:10.1007/s11042-013-1662-y
19. Patoli M, Gkion M, Newbury P, White M (2010) Real time online motion capture for entertainment applications. In: 3rd IEEE international conference on digital game and intelligent toy enhanced learning (DIGITEL), 2010, pp 139–145. doi:10.1109/DIGITEL.2010.39
20. Rahman MA (2014) Multimedia environment toward analyzing and visualizing live kinematic data for children with hemiplegia. Multimedia Tools Appl 74(15):5463–5487. doi:10.1007/s11042-014-1864-y
21. Sattler M, Sarlette R, Klein R (2005) Simple and efficient compression of animation sequences. ACM, New York, pp 209–217. doi:10.1145/1073368.1073398
22. Sheikh A, Fiandrotti A, Magli E (2014) Distributed scheduling for low-delay and loss-resilient media streaming with network coding. IEEE Trans Multimedia 16(8):2294–2306. doi:10.1109/TMM.2014.2357716
23. Suzuki T, Ikehara M (2010) Integer dct based on direct-lifting of dct-idct for lossless-to-lossy image coding. IEEE Trans Image Process 19(11):2958–2965. doi:10.1109/TIP.2010.2051867
24. Tournier M, Wu X, Courty N, Arnaud E, Revret L (2009) Motion compression using principal geodesics analysis. Comput Graphics Forum 28(2):355–364. doi:10.1111/j.1467-8659.2009.01375.x
25. Wang CW, Jeng JH (2012) Image compression using pca with clustering. In: International symposium on intelligent signal processing and communications systems (ISPACS), 2012, pp 458–462. doi:10.1109/ISPACS.2012.6473533
26. Zhang M, Tong X (2014) A new algorithm of image compression and encryption based on spatiotemporal cross chaotic system. Multimedia Tools Appl 74(24):11,255–11,279. doi:10.1007/s11042-014-2227-4

Dr. Murtaza Ali Khan is Assistant Professor in the Department of Computer Science, Umm Al-Qura University, Makkah, Saudi Arabia. He obtained his PhD from Keio University, Japan in 2008. He is author of one book and several papers in referred journals and conferences. His areas of research include signal processing, data compression, and computer graphics. Dr. Khan received many awards including Excellence in Research Award for year K.F.U.PM, KSA; Yoshida Scholarship Award for doctorate studies 2004-07, Japan; and research grant from Keio Leading-edge Laboratory of Science and Technology 2005-07, Japan.