Multidim Syst Sign Process (2016) 27:121–143
DOI 10.1007/s11045-014-0293-4

An efficient algorithm for compression of motion capture signal using multidimensional quadratic Bézier curve break-and-fit method

Murtaza Ali Khan

Received: 19 November 2013 / Revised: 11 July 2014 / Accepted: 30 July 2014 /
Published online: 12 August 2014

Abstract The emergence of applications to capture, process, store, and transmit motion capture (MoCap) signal has raise the interest in research community to investigate and devise better techniques for parameterization and compression of MoCap signal. In this work, we present a novel and efficient method for parametric representation and compression of motion signal for skeletal animation. The method exploits the temporal coherence of motion signal using quadratic Bézier curve (QBC) fitting. The method treats the 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 QBC least square fitting. Break and fit criterion is used to minimize the number of curve segments required to fit the data. Precise control of fitting accuracy is achieved by user specified tolerance of error limit. We compared the performance of the proposed method with principal component analysis and wavelet transform based methods of MoCap signal compression. The method leads to smaller storage and better visual quality compared to other methods. The low degree of QBC ensures computationally efficient fitting algorithm, especially for the real-time applications.

Keywords Motion · Multidimensional signal · Animation · Fitting · Compression · Quadratic Bézier curve

1 Introduction

Motion capture (MoCap) data is used in many application such as in clinical medicine to evaluate human motion, in virtual reality environments to drive avatars, in video games to animate characters, in movies for CG effects (Xiao et al. 2010), etc. The size of uncompressed motion data is very large. A compact representation and efficient encoding and decoding techniques are needed to store and process motion signal. In our work, we focused on compression of MoCap signal that is use for skeletal animations, though the method can be applied to other

M. A. Khan (✉)
College of Computer and Information Systems, Umm Al-Qura University, Mecca, Saudi Arabia
e-mail: makkhan@uqu.edu.sa

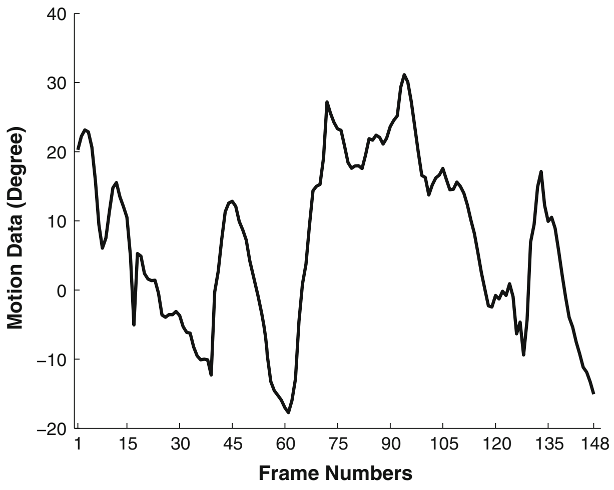

Fig. 1 Motion signal of *right toe* joint (rotation about X-axis) in 148 frames

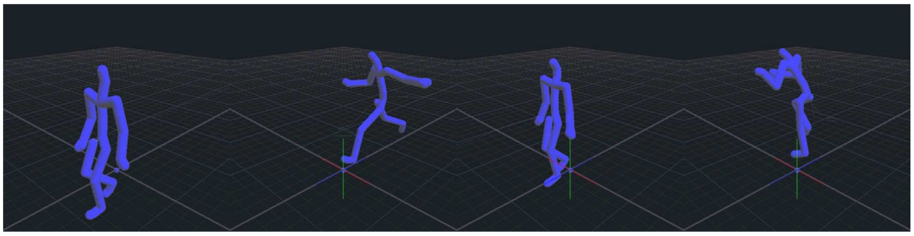

Fig. 2 Characters from *left* to *right* show *jogging*, *soccer-kick*, *walking*, and *climbing* motion activities reconstructed from compressed motion signal by the proposed Quadratic Bézier curve (QBC) method

types of motion signal as well. Lossless compression (Wang et al. 2013) though desirable but yields less compression. Lossy compression or approximation methods are more promising for motion signal.

Spline and curve are widely used in computer-aided design and computer graphics because of simplicity of their construction, accuracy of evaluation, and their capability to approximate complex shapes through curve fitting and interactive curve design (Bartels et al. 1995). Spline can fit large number of data points with far less number of control points. Control points can be encoded by some appropriate encoding technique. During the decoding process, fitted data points are regenerated by Spline interpolation of control points. The method we present is based on quadratic Bézier curve (QBC) fitting, essentially a lossy compression technique. But this loss is tolerable when taking into consideration the limitations of human visual system and high frame rate of animation.

Skeletal animation is usually represented as rotational hierarchies parameterized by a translation, rotation, and a set of joint angles. The motion is described by a set of motion curves (signal) each giving the value of one of the model's parameters (joints) in a sequence of frames as a function of time. Figure 1 shows rotational motion signal of a joint (right toe) in 160 frames. Figure 2 shows different motion activities reconstructed from the compressed motion signal by the proposed method. Organization of the rest of the paper is as follows: Related work is discussed in Sect. 2. Basics concepts about animation and motion data are explained in Sect. 3. Compression of motion signal using principal component analysis (PCA) and discrete wavelet transform (DWT) are briefly described in Sect. 4 and 5 respectively. The mathematical model of QBC is described in Sect. 6. The break-and-fit strategy of QBC to compress the motion signal is illustrated in Sect. 7. Selected experiments and results are presented in Sect. 8. Section 9 analyzes results and gives insight view of the proposed method. Final concluding remarks are in Sect. 10.

2 Related work

In the past there has been intensive research on the compression of image and video data. Techniques based on Spline (Khan 2012; Khan and Ohno 2007), PCA (Ho et al. 2005; Wang and Jeng 2012), discrete cosine transform (DCT) (Ahmed et al. 1974) (classic paper), (Suzuki and Ikehara 2010; Zhao et al. 2012), and DWT (Shapiro 1993; Said and Pearlman 1996) (classic papers) (Shinoda et al. 2008), are well established for image and video data compression. However, the area of motion signal compression is flourishing and many techniques are proposed recently. Some of the ideas of motion signal compression are originated from image/video compression methods and consequently employ Spline, PCA, DCT, and DWT. We used curve fitting method for compression of motion signal and for comparison purpose we choose PCA and DWT based methods, as they are well known and successfully tested for image, video, and animation data compression.

Many authors (Liu and McMillan 2006b; Karni and Gotsman 2004; Liu and McMillan 2006a; Lee et al. 2000; Sattler et al. 2005; Ormoneit et al. 2001) used PCA to compress the motion signal. Liu and McMillan (2006b) proposed a motion compression method that exploits both spatial and temporal coherences and divides a motion sequence into segments of simple motions, each of which falls into a space with low linear dimensionality. Segments are compressed individually using PCA and only the key frames' projections are stored, the other in-between frames are interpolated via Spline functions. In the method of Karni and Gotsman (2004), spatial correlations of animation sequences are captured using PCA, then second-order linear prediction coding (LPC) is applied to the PCA coefficients to further reduce the code size. Ormoneit et al. (2001) used PCA to build a statistical Eigen model of the motion curves.

DWT based methods suitable for compression of motion signal are described by Firouzmanesh et al. (2011), Beaudoin et al. (2007). The method of Firouzmanesh et al. (2011) divides the animation channel data into slices (subsequences), apply wavelet transform, quantize coefficients and finally applies run-length and entropy coding to store the compressed data. DWT based method of Beaudoin et al. (2007) uses a cubic interpolating bi-orthogonal wavelet basis to exploit the temporal coherence of skeletal animations. Since the search space of wavelet coefficients is very large, Beaudoin et al. (2007) gives a heuristic for optimized wavelet coefficient selection. But if chosen metric is not dependent enough on the joint hierarchy then the heuristic is too limited to yield optimize results. A Discrete Cosine Transform (DCT) based method to compress the motion signal is presented by Kwak and Bajic (2013). The method uses low-rate quantization and adapting quantizer step sizes to control the bitrate.

A hybrid method to compress large databases of MoCap signal is presented in Arikan (2006). This method approximates (compresses) short clips of motion using combination of cubic Bézier curves, clustered PCA and DCT. The technique uses virtual markers as an internal representation and thus requires conversion to and from this representation. Another method to compress MoCap database is proposed by Gu et al. (2009).

The method organizes motion markers into a hierarchy of human body nodes. Then the motion sequence corresponding to each body part is extracted using K-means clustering and coded. Both the methods of Arikan (2006) and Gu et al. (2009) are suitable for large databases of motion clips and relatively complex and computationally expensive compared to our method. A method for compression MPEG-4 Body Animation Parameters (BAP) for low bandwidth and reduce power consumption environment is presented by Chattopadhyay et al. (2007). The method is based on smart indexing through exploiting structural information of skeletal virtual human model. However, the method requires exhaustive search to obtain values of quality control parameters. A technique based on active contour fitting for compression of motion trajectories is investigated by Cheneviere and Boukir (2004). The method initially takes first and last points of the input motion trajectory in the active contour then add more points in an iterative process based on minimizing energy related to active contour. Our method is also iterative but we used least square QBC fitting and represent the motion signal using control points. Further, in order to improve the fitting efficiency, we provided a set of initial points at a regular interval from the MoCap signal as keyframes. A method to represent and compress the motion signal in quaternion space using sparse decomposition model is described by Zhu et al. (2012). In our method, we used Euclidean space, which is simpler and used by most of the motion data file formats and motion data processing systems such as animation and game engines.

3 Basics of animation and motion data

Following is the basic terminology used to describe different aspects of a animation, motion data and motion data file format.

– *Skeleton* It is a whole character on which motion data is applied to create animation. Figure 3 shows a basic skeleton of a humanoid. A skeleton is comprised of a number of bones.

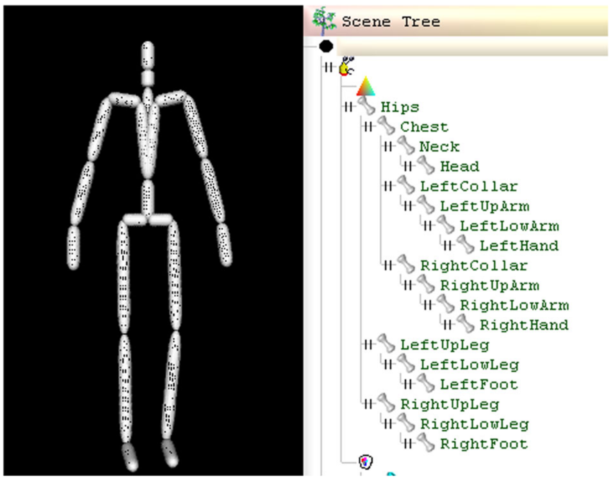

Fig. 3 A Skeleton of Humanoid

Table 1 Motion data of m channels, each channel has n frames

Frames	Channels			
	Chn.-1	Chn. − 2	…	Chn.-m
Frame-1	x_{11}	x_{12}	…	x_{1m}
Frame-2	x_{21}	x_{22}	…	x_{2m}
⋮	⋮	⋮	⋮	⋮
Frame-n	x_{n1}	x_{n2}	…	x_{nm}

- *Bone* A bone represents the smallest segment that is subject to individual translation and/or rotation changes during the animation. Bones are usually organized in a hierarchical structure, as illustrated in right side of the Fig. 3. As a consequence, when a joint moves, its connected joints lower in the hierarchy moves too. For example, if the shoulder is connected to the arm then as the shoulder moves, the arm also moves.
- *Channel or Degree of Freedom (DOF)* Each bone within a skeleton can be subject to translation, rotation, and scale changes over the course of the animation, where each parameter is referred to as a channel or degree of freedom (DOF). For example, if the root joint has freedom to translate in X, Y, and Z directions (t_x, t_y, t_z) and freedom to rotate about X, Y, and Z axis $(\theta_x, \theta_y, \theta_z)$ then it constitutes 6-DOF or six-channels (assuming no freedom to scale). The changes in the channel data over a period of time produce animation. Translation and scale changes are measured in unit of length, while rotational changes are measured in degrees or radians.
- *Frame/Frame rate* An animation is comprised of a number of frames where for each frame the channel data for each bone is defined. Frame rate refers to number of frames generated in one second. Generally frame rate varies between 24 to 60 frames per second (fps).
- *File format of motion data* Different file formats e.g., .BVH, .BVA, .C3D, .ASF/.AMC, etc. are in use to store motion data. We used most widely used Acclaim motion capture file format also known as .ASF/.AMC file format. Acclaim motion capture file is made up of two files; the .ASF (Acclaim Skeleton File), file which describes the actual skeleton and its hierarchy, and the .AMC (Acclaim Motion Capture data), file which contains the motion data. This separation of files allows one .ASF file that describes the skeleton and multiple .AMC motion files for the same skeleton. An interested reader can find the more details about file formats of motion data in Menache (2010), Parent (2009),
- *Motion data and matrix form* Let suppose that motion data of an animation consists of m channels (DOF) and each channel has rotational/translational values in a sequence of n frames as shown in Table 1. Then for efficient manipulation, motion data is store into a 2D matrix of size $n \times m$ as follows:

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

Each row of X in (1) corresponds to a single complete frame data of all the channels. Alternatively, we can say that each column of X corresponds to motion signal of a specific channel for n frames. For example, $X[i, 1 \dots m]$ contains the motion signal of ith frame for all the channels $(1 \dots m)$ and $X[1 \dots n, j]$ contains the motion signal of jth channel for all the frames $(1 \dots n)$.

4 Compression of MoCap signal by principal component analysis

PCA is an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. PCA is theoretically the optimum transform for given data in least square terms.

In order to apply principal components analysis to motion signal, we organized the motion signal into a matrix form as described in Sect. 3. We used PCA to reduce the number of channels by exploiting correlations among channels. From the input matrix of motion signal, we determined covariance matrix, Eigen values, and Eigen vectors. We sort the Eigen values and corresponding Eigen vectors in order of decreasing Eigen values (that is why it is called principal component analysis). Compression is achieved by retaining only q eigenvectors, where $q < m$. Note that for $q = m$, it will be lossless coding

5 Compression of MoCap signal by wavelet transform

DWT consists of decomposing a signal into a hierarchical set of *approximations coefficients* (low frequencies) and *details coefficients* (high frequencies). Approximations coefficients contain coarser details while details coefficients contain fine details of the signal at each level. Both the approximation and details coefficients can be obtained by convolving the coefficient of approximation at a coarser resolution with a filter and down sampling it by a dyadic scale (factor of 2). The wavelet compression 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.

In order to apply wavelet transform to MoCap signal we decomposed the rotational and translation data of each channel in a sequence of frames separately. For example, for root joint, the n rotational values along X-axis, $\theta_X = \left\{\theta_X^1, \theta_X^2, \dots, \theta_X^n\right\}$, are considered as a signal. Motion data signal is decomposed into approximation coefficients and detail coefficients using DWT up to chosen level k. Then the detail coefficients less than or equal to some predefined threshold, let say t_w, are quantized to zero. Finally, approximation coefficients, detail coefficients, and decomposition structure are saved. Due to quantization of detail coefficients they have long sequence of zeros. Therefore, they are coded using run-length encoding before saving. Approximated motion signal of each channel is reconstructed (decoded) by applying the inverse wavelet transform. The inverse wavelet transform uses the original approximation coefficients of level k and the threshold detail coefficients of levels from 1 to k.

The adaptive wavelet transform (AWT) method (Beaudoin et al. 2007) is essentially a DWT based method with some heuristic to select coefficients for truncation. For each channel, it retains certain number of largest coefficients, let say a_w, and set the remaining coefficients to zero (like thresholding of DWT). Then using a heuristic simulated annealing technique it redistribute the coefficients by increasing a_w of some channels and decreasing a_w of other channels to minimize distortion error. However, due to limited search space and without any matrix to measure the importance of channels the technique does not guarantee optimal results.

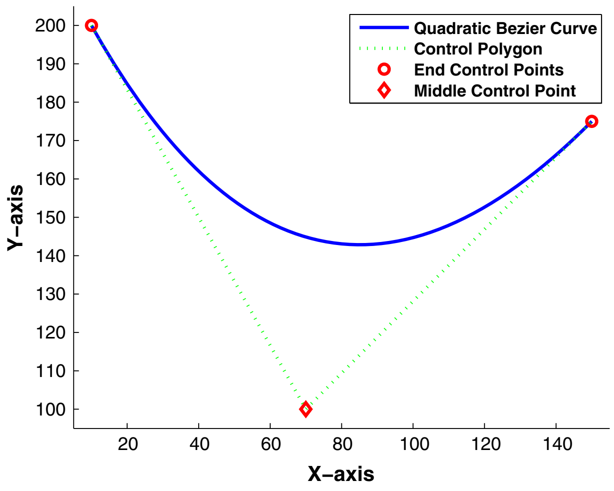

Fig. 4 A quadratic Bézier curve segment in 2-D space

6 Quadratic Bézier curve (QBC)

A QBC is a C^0 continuous curve. A QBC segment is defined by three control points: P_0, P_1, and P_2. A QBC segment (sold line) and its control polygon (dotted line) are shown in Fig. 4. P_0 and P_2 are called *end control points* (ECP), while P_1 is called *middle control point* (MCP). A QBC interpolates (passes through) its *end control points*, while the *middle control point* is used to control the shape of the curve. Equation of a QBC segment can be written as follows:

$$Q(t_i) = (1 - t_i)^2\, P_0 + 2t_i\,(1 - t_i)\, P_1 + t_i^2 P_2, \tag{2}$$

where t_i is a parameter of interpolation, $0 \le t_i \le 1$. In order to generate m points between P_0 and P_2 inclusive, the parameter t_i is uniformly divided into $(m - 1)$ intervals between 0 and 1 inclusive as follows:

$$t_i = \begin{cases} 0 & i = 1; \\ t_i = t_{i-1} + \delta, \quad \delta = \frac{1}{m-1}, & 2 \le i \le m - 1; \\ 1 & i = m. \end{cases} \tag{3}$$

$Q(t_i)$ is evaluated at m values of t_i. Since a QBC interpolates its first and last control points, therefore, $Q(t_1 = 0) = P_0$ and $Q(t_m = 1) = P_2$. More than one Bézier curves can be connected together to generate a multi-segment continuous curve. A multi-segment continuous curve interpolates $k + 1$ points using k QBC segments. For connected Bézier curves, the P_2 of jth segment is equal to P_0 of $(j+1)$th segment. Figure 5 shows a continuous multi-segment curve obtained by joining seven QBCs segments.

7 Compression of MoCap signal by quadratic Bézier curve

This section describes the most significant contribution of our research work that is compression of motion signal using QBC fitting. Fitting process is applied to motion signal of each

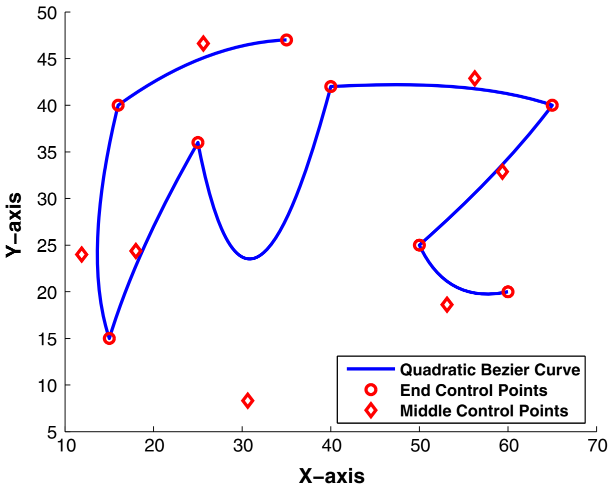

Fig. 5 A multi-segment quadratic Bézier curve in 2-D space

joint individually, a joint data may consists of multiple channels. The sample data we used has at least one channel (1-DOF) and at most six channels (6-DOF) i.e., three translational and three rotational channels. The fitting is applied in Euclidean space R^N. In the data we used, N can have any value between 1 and 6 inclusive, depending on the number of channels (DOF) of a particular joint. Even though the proposed algorithm and its implementation is general and can work for $N > 6$. We describe the fitting process to motion signal of a single joint. The same procedure is applied to all the joints.

Let input data (motion signal) of a joint consists of set of translation and rotation values in n frames. We consider input data as a set of points, $I = \{p_1, p_2, \ldots, p_n\}$. A point p_i is an Euclidean space R^N. For example, if a joint has three rotational channels (3-DOF), i.e., rotational values along X, Y and Z axes then point is in Euclidean space R^3, i.e., $p = (\theta_X, \theta_Y, \theta_Z)$.

As an input to our fitting algorithm two parameters are required: (1) *upper limit of error* d^2_{lmt}, i.e., maximum allowed squared distance between original and fitted data, e.g., $d^2_{lmt} = 100$, (2) *initial breakpoint interval* Δ, i.e., after Δ points (frames) a point (rotational and/or translational value of the channel) is taken as a *breakpoint*. For example, $\Delta = 80$ then set of breakpoints is $BP = \{p_1, p_{81}, p_{161}, \ldots, p_n\}$ (first and last points of input data are always taken as a breakpoints). The fitting process divides the data into segments based on breakpoints, i.e., $S = \{S_1, S_2, \ldots, S_l\}$. A segment is a set of all points between two consecutive breakpoints, e.g., if $\Delta = 80$ then $S_1 = \{p_1, p_2, \ldots, p_{81}\}$, $S_2 = \{p_{81}, p_{82}, \ldots, p_{161}\}$, etc. Since we are fitting a continuous curve, therefore, the last point of each segment S_i and first point of next segment S_{i+1} are common (except for last segment which does not have next segment).

Each segment is fitted (approximated) by a QBC. We describe the process of fitting a segment S_1. Let the segment $S_1 = \{p_1, p_2, \ldots, p_m\}$ has m points ($m = \Delta + 1$). The first and the last points of a segment are taken as *end control points* (ECP) of QBC, i.e.,

$$P_0 = p_1, \tag{4}$$
$$P_2 = p_m. \tag{5}$$

The *middle control point* (MCP) i.e., P_1 is obtained by least square method. Least square method gives the *best* value of *middle control point* that minimizes the squared distance between the original data of the segment, S_1 and approximated data, $Q(t_i)$. We can write the least square equation for segment $S_1 = \{p_1, p_2, \ldots, p_m\}$ as follows:

$$U = \sum_{i=1}^{m} [p_i - Q(t_i)]^2. \tag{6}$$

Substituting the value of $Q(t_i)$ from (2) in (6) yields:

$$U = \sum_{i=1}^{m} [p_i - (1-t_i)^2 P_0 + 2t_i(1-t_i)P_1 + t_i^2 P_2]^2. \tag{7}$$

To find value of P_1 differentiating (7) partially with respect to P_1 yields:

$$\frac{\partial U}{\partial P_1} = 0. \tag{8}$$

Solving (8) for P_1 gives:

$$P_1 = \frac{\sum_{i=1}^{m} \left[p_i - (1-t_i)^2 P_0 - t_i^2 P_2\right]}{\sum_{i=1}^{n} 2t_i(1-t_i)}. \tag{9}$$

(4), (5), and (9) give us the values of three control points of QBC. Once all three control points are determined then approximated data of the segment S_1 is obtained by substituting the values of control points, i.e., P_0, P_1 and P_2 in (2) and varying the value of parameter t_i uniformly in (2) using (3).

Same fitting procedure is repeated for each segment. Common points between each pair of adjacent segments are taken only once. This yields n values of approximated data, $Q = \{q_1, q_2, \ldots, q_n\}$ for a joint. Then we compute error of fitting, i.e., squared distance of each point between the original data and the approximated data, $d_i^2 = |p_i - q_i|^2$, $1 \le i \le n$. Among all the values of d_i^2, we compute maximum squared distance, $d_{max}^2 = Max\left(d_1^2, d_2^2, \ldots, d_n^2\right)$. If maximum squared distance of any jth segment is greater than d_{lmt}^2 then this segment is split and replaced with two new segments, j_1th and j_2th, i.e., $S = \left[\{S\} - \{S_j\}\right] \bigcup \left\{S_{j_1}, S_{j_2}\right\}$. A new breakpoint bp_{new} from original data is added in the set of breakpoints where the error is maximum, i.e., $BP = \{BP\} \bigcup \{bp_{new}\}$. For example, if segment S_1 splits at point p_{35} then a new breakpoint $bp_{new} = p_{35}$ is inserted between breakpoints p_1 and p_{81} $\left(BP = \left\{p_1, \overbrace{p_{35}}^{bp_{new}}, p_{81}, p_{161}, \ldots, p_n\right\}\right)$ and two new segments $\{p_1, p_2, \ldots, p_{35}\}$ and $\{p_{35}, p_{36}, \ldots, p_{161}\}$ replace S_1. Figures 6 and 7 elaborate splitting of a segment into two segments. In Fig. 6, motion signal of joint in Euclidean space R^1 (1-DOF) is fitted using a single Bézier curve segment. The points where the distance is maximum between original data and fitted data are marked and a line is drawn between them to indicate d_{max}^2. In Fig. 7, a new breakpoint is inserted where the error is maximum (d_{max}^2) between original data and fitted data. The fitting process is repeated with new set of breakpoints until mean squared error of all the segments is equal to or less than d_{lmt}^2. The same fitting process is applied to motion signal of all the joints. Figures 8, 9 and 10 show QBC least square fitting to motion signal of three different joints in Euclidean space R^1, R^2, and R^3 respectively.

The algorithm of fitting QBC to motion signal of a channel in n frames is formally described as follows:

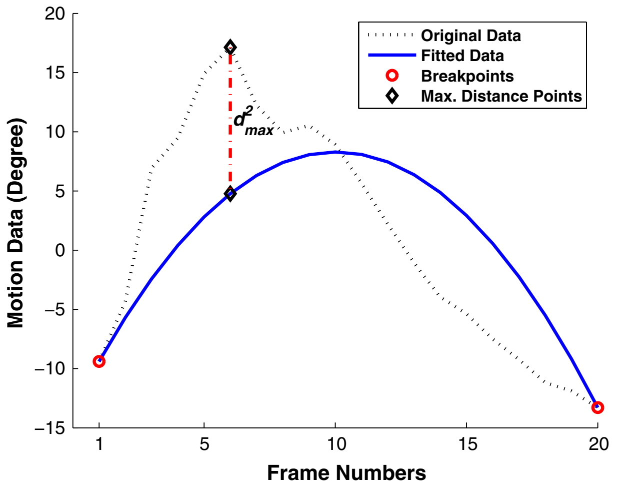

Fig. 6 Maximum error i.e., d^2_{max} occurs at frame 6

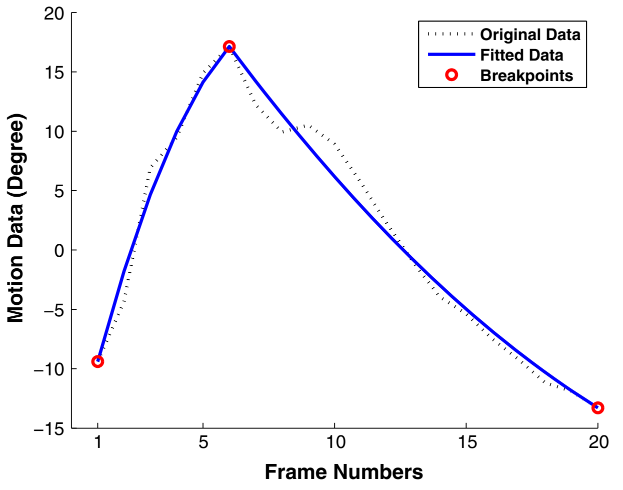

Fig. 7 A new breakpoint is inserted at frame 6, where the error was maximum (see Fig. 6)

Algorithm: QBC encoding (fitting) of motion signal of a joint in n frames.

1. Specify input data and parameters:

   (a) $I \leftarrow \{p_1, p_2, \ldots, p_n\}$, motion signal of a joint for n-frames as a set of points.
   (b) d^2_{lmt}, *maximum allowed squared distance* between input data and fitted data.
   (c) Δ, *initial breakpoint interval*. After Δ points (frames) a point is taken as a *breakpoint*.

2. Determine set of breakpoints BP

   (a) Find number of breakpoints u,
   $u \leftarrow a + \left\lceil \frac{n-b}{\Delta} \right\rceil$, $a \leftarrow \left\lceil \frac{n}{\Delta} \right\rceil$, $b \leftarrow (a-1)\Delta + 1$, $u \ll n$

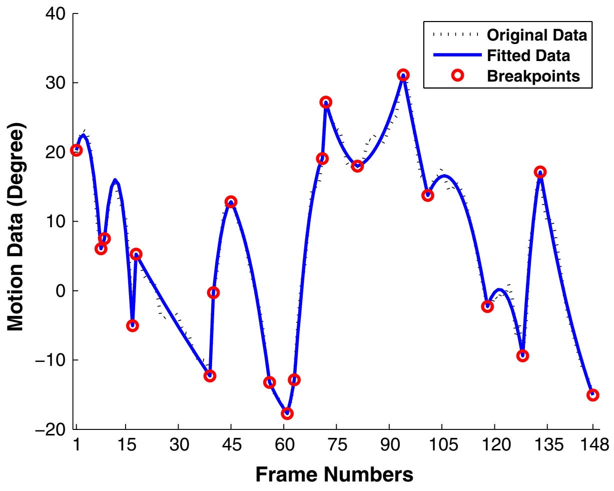

Fig. 8 Quadratic Bézier curve least square fitting in Euclidean space R^1 to motion signal of *right toe* joint in 148 frames, $d_{max}^2 = 8$. The joint has one channel (1-DOF), i.e., rotation about X-axis. Note that the fitting is applied only to vertical axis motion signal, frames numbers along the horizontal axis are shown only to depict the corresponding frames

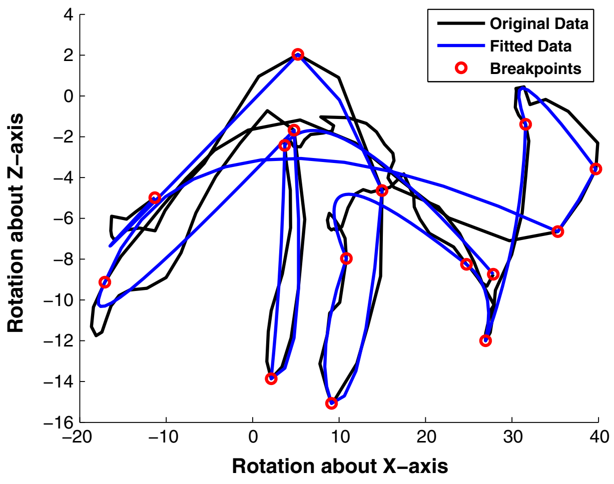

Fig. 9 Quadratic Bézier curve least square fitting in Euclidean space R^2 to motion signal of *right foot* joint in 148 frames, $d_{max}^2 = 8$. The joint has two channels (2-DOF), i.e., rotations about X and Z axes

(b) Get *indices of breakpoints*,
$BPI \leftarrow \{1, 1+\Delta, 1+2\Delta, \ldots, n\}$,
$BPI \leftarrow \{v_1, v_2, \ldots, v_u\}$,

(c) Get set of *breakpoints*,
$BP \leftarrow \left\{p_{v_1}, p_{v_2}, \ldots, p_{v_u}\right\}$,

3. Divide I into segments
$S \leftarrow \{S_1, S_2, \ldots, S_{u-1}\}$, $S_i \leftarrow \left\{p_{v_i}, \ldots, p_{v_{i+1}}\right\}$.

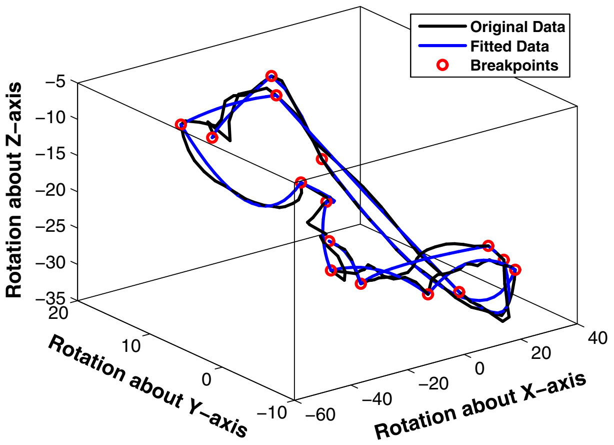

Fig. 10 Quadratic Bézier curve least square fitting in Euclidean space R^3 to motion signal of *left femur* joint in 148 frames, $d^2_{max} = 8$. The joint has three channels (3-DOF), i.e., rotations about X, Y and Z axes

```
4. Find $MCP$ of QBC of each segment using (9), i.e., find $MCP \leftarrow \left\{P_1^{S_1}, P_1^{S_2}, \ldots, P_1^{S_{u-1}}\right\}$
5. Fit a QBC to each segment using (2), i.e., find $Q \leftarrow \{q_1, q_2, \ldots, q_n\}$
6. Computer error, $d_i^2 \leftarrow |p_i - q_i|^2, \quad 1 \le i \le n$
7. Computer maximum error,
   $d^2_{max} \leftarrow Max\left(d_1^2, d_2^2, \ldots, d_n^2\right)$
8. WHILE $d^2_{max} > d^2_{lmt}$
9. $d^2_{max} \in k$th frame, $k \in j$th segment
10. $u \leftarrow u + 1$
11. $bp_{new} \leftarrow p_k$
12. $BPI \leftarrow \{BPI\} \cup \{k\}$
13. $BP \leftarrow \{BP\} \cup \{bp_{new}\}$
14. Split $j$th segment into $j_1$th and $j_2$th segments
15. $S \leftarrow \{S\} - \left\{S_j\right\} \cup \left\{S_{j_1}, S_{j_2}\right\}$
16. $MCP \leftarrow \{MCP\} \cup \left\{P_1^{j_1}, P_1^{j_2}\right\}$
17. Using updated $BP$ fit QBC to $j_1$ & $j_2$ segments
18. Find $d_i^2$ of $j_1$ & $j_2$ segments
19. $d^2_{max} \leftarrow Max\left(d^2_{max}, d^2_{j_1}, d^2_{j_2}\right)$
20. END-WHILE
21. Find count of interpolating points for each segment from $BPI$,
    $C \leftarrow \{c_1, c_2, \ldots, c_{u-1}\}$, $c_i = v_{(i+1)} - v_i + 1$
22. Save $BP$, $MCP$ and $C$
```

The QBC output data to be stored for approximated motion signal of each joint consists of:

1. Set of *control-points*, i.e., CP, where $CP_i = \{BP_i, MCP_i\}$, $BP_i = \left\{P_{0_i}, P_{2_i}\right\}$ for all the segments. Note that breakpoints BP are nothing but ECP, see (4) and (5). Obviously, if a joint has N-channels then its control points are in Euclidean space R^N

2. Set of *count of interpolating points*, i.e., C for all the segments. C is needed to calculate number of interpolating points between ECP of each segment. *Count of interpolating points* are always in Euclidean space R^1, regardless of the dimension of the joint motion signal.

Let r connected Bézier curve segments are required to approximate the input data of a joint. Then the set of *control-points* is $CP = \{CP_1, CP_2, \ldots, CP_r\}$, and the set of *count of interpolating points* is $C = \{C_1, C_2, \ldots, C_r\}$. For a jth segment, $1 \le j \le r$, CP_j and C_j are *control-points* (P_{0_j}, P_{1_j}, and P_{2_j}) and *count of interpolating point* respectively. We used uniform parameterization, therefore, there is no need to store values of parameter t_i, it can be generated on the fly during decoding using (3). Compression is achieved due to the fact that large input data of each joint is approximated with very few control-points.

The reconstruction/decoding of a segment S_j of a channel is very simple. Let CP_j is P_{0_j}, P_{1_j}, and P_{2_j}. P_{0_j} and P_{2_j} are ECP while P_{1_j} is MCP. Let count of interpolating points of S_j between P_{0_j} and P_{2_j} is $C_j = m$ (all this is obtained from stored output data). Then the parameter value t_{i_j}, $1 \le i \le m$, is obtained by uniformly dividing t_{i_j} into $m - 1$ intervals between 0 and 1 inclusive. Finally approximated values of the segment S_j are obtained by substituting the value of P_{0_j}, P_{0_j}, and P_{2_j} and t_{i_j} in (2). All the segments of all the channels are decoded in the same way.

The procedure of reconstruction/decoding of motion signal of a joint in n frames is formally described as follows:

Algorithm: QBC decoding (reconstruction) of motion signal of a channel in n frames.

```
1. Load $BP$, $MCP$ and $C$
2. $u \leftarrow |BP|$, $u$ is number of breakpoints
   $(u-1)$ is number of segments
3. FOR $j \leftarrow 1$ to $u-1$
4.     $P_{0_j} \leftarrow BP_j$, $P_{1_j} \leftarrow MCP_j$, $P_{2_j} \leftarrow BP_{j+1}$
5.     $m \leftarrow C_j$, i.e., count of interpolating points
6.     Evaluate $m$ values of $Q_j$ ($j$th segment) using (2). Parameter $t_i$ is determined using (3)
           $Q_j \leftarrow \begin{cases} \{q_1, q_2, \ldots, q_m\} & j = 1; \\ \{q_2, q_3, \ldots, q_m\} & 2 \le j \le u-1. \end{cases}$
7.     $Q \leftarrow \{Q\} \cup \{Q_j\}$, i.e., build global $Q$ from $Q_j$.
8. END-FOR
9. return $Q$,
   $Q = \{q_1, q_2, \ldots, q_n\}$ is reconstructed/decoded data of a joint in $n$ frames
```

We encoded/decoded the motion signal of each joint independently. We used the break-and-fit strategy that splits the motion curve of joint at the frame where the distance between approximated and fitted data exceeds the predefined threshold. Therefore, number of break-points are not equal for motion signal of different joints. However, since initially we took Δ as initial breakpoint interval, this ensures that motion signal of every joint has a break-point after Δ frames. For example, if $\Delta = 80$ then joint 1 may have final break points $BP_{joint-1} = \{p_1, p_{35}, p_{81}, p_{98}, p_{161}, \ldots, p_n\}$ and joint 2 may have final break points $BP_{joint-2} = \{p_1, p_{81}, p_{110}, p_{161}, \ldots, p_n\}$. Note that $\{p_1, p_{81}, p_{161}, \ldots, p_n\}$ are common breakpoints between joint 1 and joint 2 due to Δ, while other breakpoints (frames) are

Table 2 Environment of experiments

Hardware	Inter Core Duo 2.4 GHz
Operating system	Windows XP SP3
Programming language	MATLAB 7.0.4 SP2
Animation testing	Endorphin 2.7.1

Table 3 Details of input animations

Anim. name	Anim. code	Channel count	Joint count	Frame count	File-size (bits)
Jogging	38_01	62	31	352	2,246,890
Soccer-kick	10_03	62	31	362	2,305,144
Walking	05_01	62	31	598	3,838,930
Climbing	143_37	62	31	551	3,507,090

Frame rate 60-Hz, file format .ASF/.AMC

not common due to splitting of motion curve at arbitrary points (frames). The common breakpoints (frames) after every Δ frames have another implication. The complete frames of encoded data can be previewed/played at these common breakpoints (frames) without decoding. This feature is very helpful when a user wants to preview an animation without completely decoding it. Similar technique is used in video coding, where these frames are called I frames.

8 Experiments and results

We have applied above described methods on motion signal of several animations. Table 2 provides details of environment of experiments. Table 3 gives the details of representative input animations used in the simulations.

We compressed the motion signal of each animation using PCA, DWT, AWT (Beaudoin et al. 2007) and QBC methods. Then in order to measure the performance of all the methods we computed following parameters:

1. Compression ratio (CR), i.e., ratio between size of original motion signal file f_o (.AMC file) and size of compressed motion signal file f_c:
$$CR = \frac{f_o}{f_c}. \tag{10}$$
2. Mean squared error (MSE) between original motion signal X_o and reconstructed motion signal X_c from the compressed representation:
$$MSE = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\|X_o - X_c\|^2 . \tag{11}$$

In (11), the motion signal matrix (X_o or X_c) has the form as described in (1).

The input parameters to control the rate-distortion (MSE vs. CR) depend on *number of largest value Eigen-vectors to be retained* for PCA, *wavelet threshold* value for DWT, *number of coefficients retained* for AWT and *maximum allowed square distance* for QBC. These input parameters are not related to each other. Therefore, it is very difficult to get the

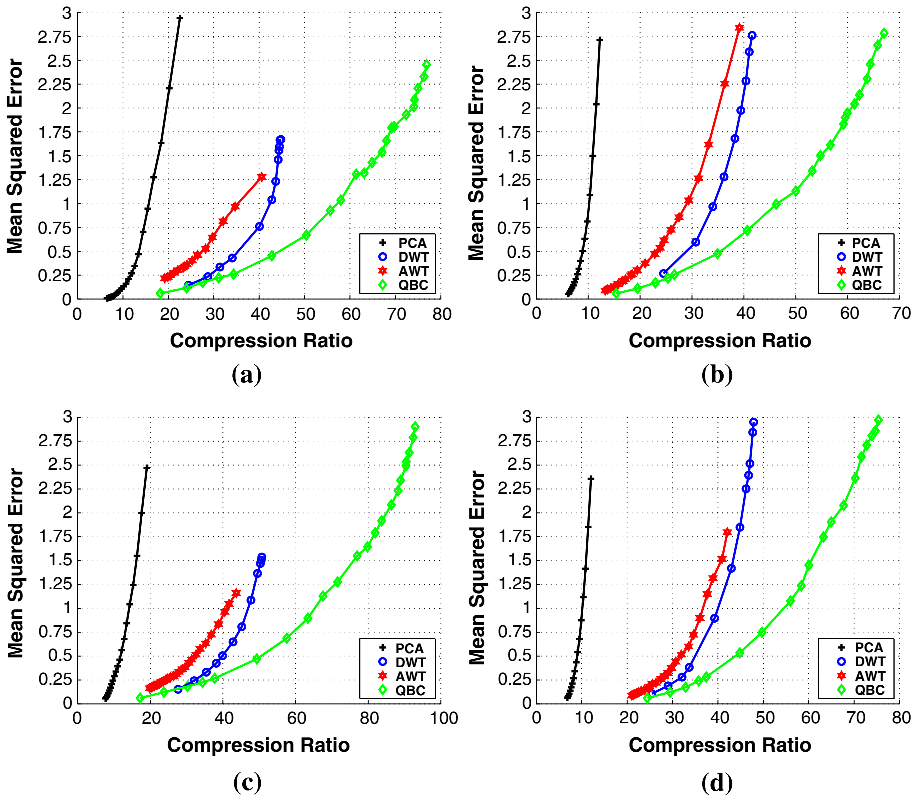

Fig. 11 Comparative performance (*CR* vs. *MSE*) of PCA, DWT, AWT and QBC methods for each animations individually. **a** Jogging. **b** Soccer-kick. **c** Walking. **d** Climbing

same value of MSE or CR for all the methods at given values of input parameters. Therefore, in order to evaluate performance of PCA, DWT, AWT and QBC, the best we can do is to vary the value of their respective input parameters and obtain the multiple values of MSE and CR and plot the distortion curve for these methods. Figures 11 and 12 show the MSE versus CR performance of PCA, DWT, AWT and QBC methods for all the animations listed in Table 3.

In all the results, the values of principal component transform matrix, detail coefficients of DWT/AWT and control points (BP and MCP) of QBC are rounded to 2 decimal places. DWT/AWT coefficients are also run-length encoded in spirit of their characteristics. Final data is stored in MATLAB'S MAT-file format (http://www.mathworks.com/) for each method separately.

9 Discussion

9.1 The comparison among different algorithms

Figures 11 and 12 show the compression ratio (CR) versus mean squared error (MSE) for PCA, DTW, AWT and QBC methods for all the animations. It is desirable to have low value of MSE and high value of CR. In order to easily compare PCA, DWT, AWT and QBC, we

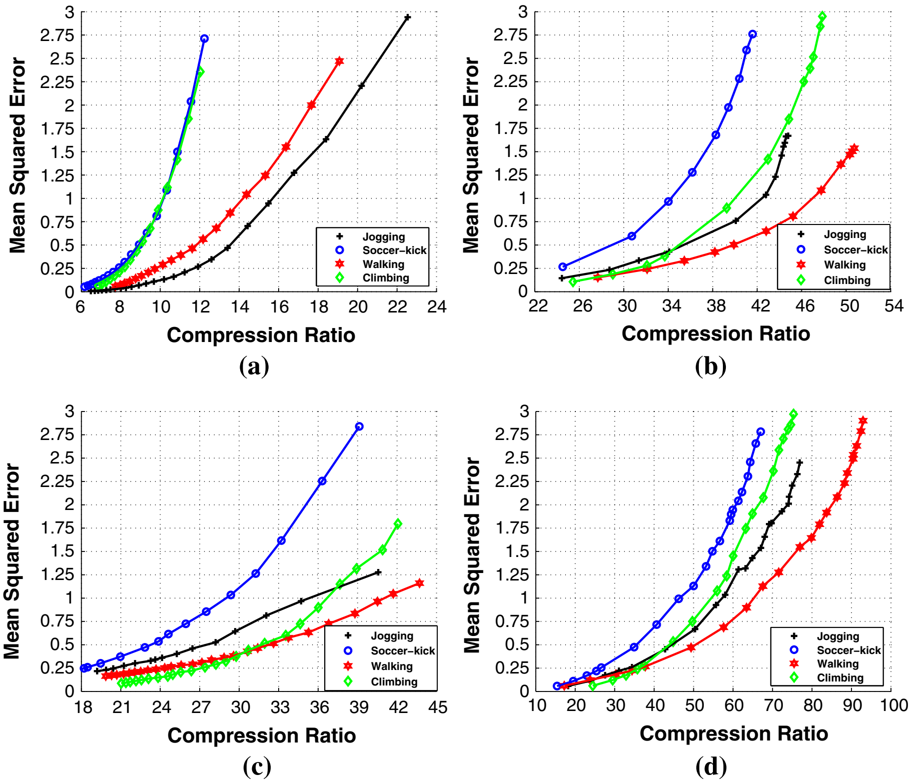

Fig. 12 Comparative performance (*CR* vs. *MSE*) of Jogging, Soccer-kick, Walking and Climbing animations for each methods individually. **a** PCA. **b** DWT. **c** AWT. **d** QBC

kept scaling of MSE axis same for all the methods. The QBC method yields the best results. DWT performs slightly better than AWT, while PCA performance is lowest. As an example, Table 4 shows the mean squared error and compression ratio comparison of all the methods for a single observation selected arbitrarily. For example, the (MSE, CR) pairs are (1.4155, 10.855), (1.4196, 43.097), (1.5174, 40.902) and (1.4065, 59.410) for PCA, DTW, AWT and QBC respectively for the *Climbing* animation. In the Table 4, we choose the value of MSE for all the methods as close as possible. Figures 13, 14, 15 and 16 show arbitrary selected frames for each of the animation described in the Table 4. Each figure (Figs. 1314, 15, 16) has five parts (a–e) to show the original, PCA, DWT, AWT and QBC frames. For PCA, DWT, AWT and QBC, their respective frames are reconstructed from the encoded data and may have some loss of information. From Figs. 11 and 12, we can conclude that in general QBC performs the best. Now consider the performance of each method individually for the input animations by analyzing the same set of figures independently. It can be observed that PCA method performs best for *Jogging* animation, while DWT, AWT and QBC methods yield best results for *Walking* animation. *Soccer-kick* animation has comparatively more incoherent motion compared to other animations, all the methods perform lowest for this animation.

MoCap signal is inherently smooth in temporal dimension due to inertia property of human body motion. The QBC method performs better than other methods because Spline and curve are well suited for approximation of smooth temporal data. Transformation based techniques

Table 4 Mean squared error (MSE) and compression ratio (CR) comparison of all the methods for a single observation of each animation

Record no.	Animation name	PCA			DWT			AWT			QBC		
		q	MSE	CR	t_w	MSE	CR	a_w	MSE	CR	d^2_{lmt}	MSE	CR
1	*Jogging*	13	0.346	12.59	4	0.334	31.35	39	0.341	23.62	7	0.329	37.94
2	*Soccer-kick*	22	0.258	7.919	3	0.266	24.88	53	0.259	18.46	5	0.255	26.60
3	*Walking*	13	1.042	14.37	15	1.085	47.84	41	1.046	41.71	24	1.087	67.07
4	*Climbing*	17	1.415	10.85	15	1.419	43.09	38	1.517	40.90	28	1.406	59.41

For PCA, q is *Number of largest value Eigen vectors*. For DWT, t_w is *wavelet threshold*. For AWT, a_w is *average number of coefficients retained*. For QBC, d^2_{lmt} is *maximum allowed square distance*

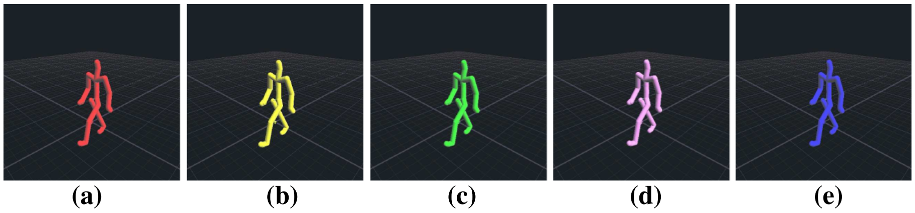

Fig. 13 Frame 300 of *Jogging* animation. See record no. 1 in Table 4 for comparative statistics. **a** Original. **b** PCA. **c** DWT. **d** AWT. **e** QBC

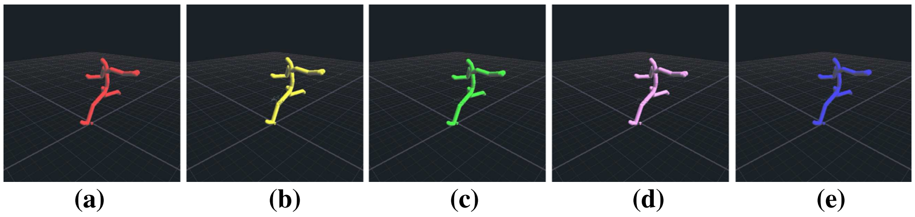

Fig. 14 Frame 120 of *Soccer-kick* animation. See record no. 2 in Table 4 for comparative statistics. **a** Original. **b** PCA. **c** DWT. **d** AWT. **e** QBC

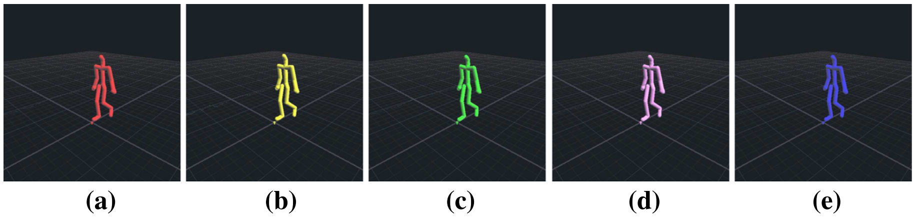

Fig. 15 Frame 210 of *Walking* animation. See record no. 3 in Table 4 for comparative statistics. **a** Original. **b** PCA. **c** DWT. **d** AWT. **e** QBC

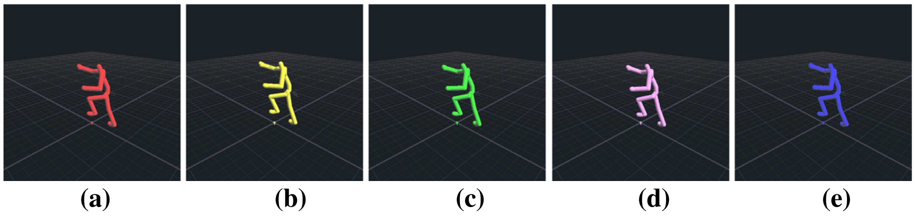

Fig. 16 Frame 104 of *Climbing* animation. See record no. 4 in Table 4 for comparative statistics. **a** Original. **b** PCA. **c** DWT. **d** AWT. **e** QBC

such as discrete cosine transform and discrete wavelet transform (DWT and AWT) are better suited to de-correlate spatial correlation of image and video data (e.g. in JPEG and MPEG) but they are not used to approximate temporal correlation; the temporal correlation in video data is exploited using non-transformation based technique, i.e., motion estimation via block matching. PCA technique is suitable to reduce the dimensionality of data (variables) but it does not take advantage of temporal correlation of motion signal of each joint (individual

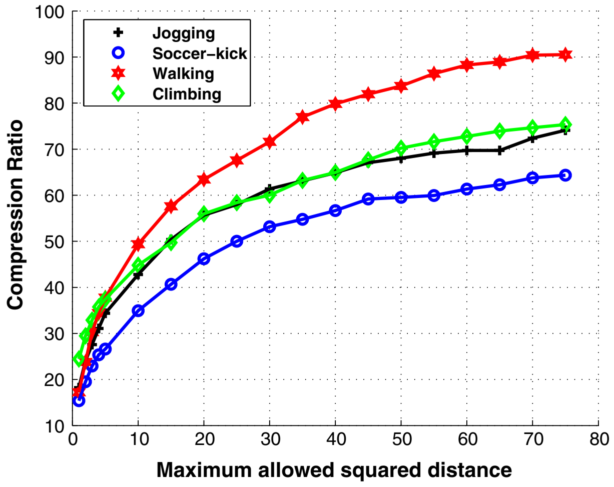

Fig. 17 *Maximum allowed squared distance* versus *compression ratio* performance of QBC for all animations

variable). Further, reconstructed PCA data is not smooth and produces jerks in the motion and increase the error.

9.2 Measure of distortion

Authors use different matrices to measure the error of approximated data. In his work, Arikan (2006) used RMS, which is essentially a square root of MSE, and similar to our matrix. Another matrix to measure the error is positional distortion error (PDE) used by Beaudoin et al. (2007). We found that MSE is better related to compression ratio and corresponding quality of final animation compared to PDE. Regardless of any measure of error the relative performance and quality of animations for PCA, DWT, AWT and QBC remain same.

9.3 Effect of error controlling parameters

In our method, the error controlling parameters is *maximum allowed square distance* i.e., (d^2_{lmt}). Figure 17 shows the effect of d^2_{lmt} on compression ratio (CR) for QBC method. Increasing value of d^2_{lmt} reduces the number of control points and consequently increases the compression ratio. Figure 18 shows the effect of d^2_{lmt} on mean squared error (MSE). Increasing the value of d^2_{lmt} increases the MSE. Around $d^2_{lmt} \leq 18$, we get MSE ≤ 1 for all the animations. Since d^2_{lmt} is directly related to MSE, it means a non-technical animator can control the MSE easily using d^2_{lmt}. Further, if we define some acceptable MSE then proposed technique (QBC) does not require any user defined parameters.

9.4 Relative performance

The author of method Beaudoin et al. (2007) reported no noticeable artifacts at compression ratio of around 30–37 for individual clips. The author of Arikan (2006) reported compression ratio 30–35 for the whole CMU database. Our method yields compression ratio of around 40–45 without any noticeable artifacts at $1 \leq MSE \leq 0.75$.

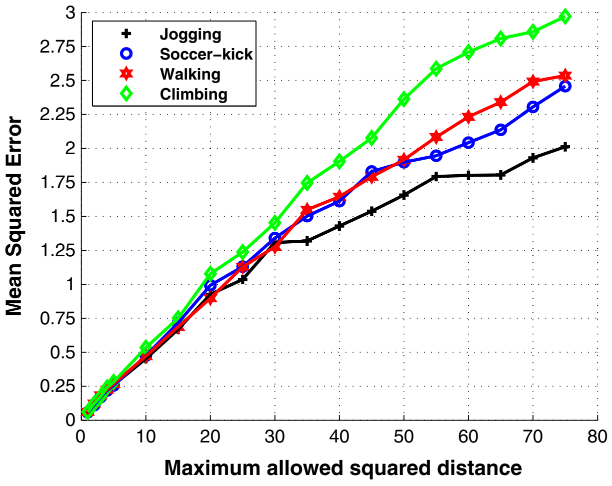

Fig. 18 *Maximum allowed squared distance* versus *mean squared error* performance of QBC for all animations

9.5 Fitting data of each joint individually

MoCap system provides motion signal of each joint individually. Therefore, it is more natural to compress motion signal of each joint separately and this is the approach we used in QBC fitting. This fitting also provides the flexibility to encode/decode the motion signal of any joint independent of other joints. This flexibility is very useful for interactive motion editing system where the animator is only interested in motion editing of selected joints.

9.6 QBC fitting in higher dimensional space

It can be observed from Figs. 8, 9 and 10 that the QBC fitting works well in Euclidean space R^N. We fit the root joint in Euclidean space R^6 very well, though it is difficult to graphically show the plot of this fitting. Applying fitting to motion signal in Euclidean space R^N is not only a novel approach but it also saved the storage space of *count of interpolating points*, i.e., C. Alternatively, if we apply fitting in Euclidean space R^1, i.e., individually to each channel of a joint then we have to save *count of interpolating points* for each channel separately. Because each channel may break-and-fit with different number of control points at different locations even for the same value of d^2_{lmt}. For example, the *left femur* joint has three channels (3D data). Therefore, we applied fitting in R^3 and consequently we have to save only one set *count of interpolating points* for three channels of *left femur* joint. Fortunately, *count of interpolating points* always remains a scalar (1D) value for fitting in higher dimensions.

9.7 Computational and storage efficiency of QBC

QBC is computationally more efficient than other curves e.g., Natural cubic Spline, B-Spline, cubic Bézier curve, etc. Moreover, for QBC we need to store only one middle control point where as cubic Bézier curve requires to store two middle control points.

9.8 Continuity and break-and-fit strategy of QBC

If we analyze behavior of motion signal of any joint, we would observe that for some sequence of frames the joint moves smoothly, consequently the motion curve is smooth. On the other hand, when a joint changes its position or orientation then there is discontinuity and the motion curve is not smooth (e.g., see Fig. 1). The single segment of QBC is sufficiently smooth to approximate the smooth region of a joint signal. But when there is discontinuity in the motion signal then a single segment of QBC may not approximate it up to desired level of accuracy. At this point, our algorithm uses break-and-fit strategy and inserts a breakpoint at the point of discontinuity (e.g., see Figs. 6, 7) and approximate the data using more than one segment. Break-and-fit strategy of QBC method allows to achieve desired level of fitting accuracy by simply setting appropriate upper limit of error. We did not enforce higher order continuity at joining point of two segments. Because, if the original motion signal exhibits discontinuity then there is no justification to enforce higher order continuity artificially. Nevertheless, geometric continuity is always there, as we used connected Bézier curves. Our objective is to approximate the original motion signal as closely as possible, rather than to smooth the motion signal. However, if desired the method of Hsieh (2002) or Rosenhahn et al. (2007) can be used to smooth the motion signal prior to applying our method.

9.9 Environmental contacts

To enforce constraint on environmental contact (typically foot) is a difficult problem. Foot sliding on the ground may cause visible artifacts. One solution of this problem is to save foot data using lossless encoding. This would guarantee accuracy but at the cost of lesser compression. Another approach which does not require special treatment for foot signal is to correct the foot contact during decoding. A technique presented by Ikemoto et al. (2006) automatically detects and corrects foot skate. The technique of Ikemoto et al. (2006) can be applied to our method during decoding process. However, in our experiments, we did not enforce constraint on environmental contacts and left it as future work.

10 Conclusion

We presented an efficient method for compression of MoCap signal using multidimensional QBC fitting. The QBC method parameterizes and approximates the motion signal of each joint independently in Euclidean space R^N. We used least square solution to find the optimal value of middle control point of QBC and used break-and-fit strategy to enforce the upper limit of error. In the proposed method, initially a user has to set few parameters then rest of the process is fully automated. Practically the method can decode the compressed data and play the animation in real-time. Experimental results show that QBC performs better than PCA, DWT and AWT based methods and yields higher compression ratio. The subjective quality of QBC reconstructed animation is also very good. The method has the limitation that it does not check and enforce constraint on environmental contacts. Due to low computational cost and low space requirement the proposed method is suitable for compressing motion signal for interactive and real time applications.

Acknowledgments In this research work, the author gratefully acknowledges Muhammad Arif and Ahmed Kattan both belong to Department of Computer Science, Umm Al-Qura University for their valuable comments.

References

Ahmed, N., Natarajan, T., & Rao, K. (1974). Discrete cosine transform. *IEEE Transactions on Computers*, *C–23*(1), 90–93. doi:10.1109/T-C.1974.223784.

Arikan, O. (2006). Compression of motion capture databases. *ACM Transactions on Graphics*, *25*(3), 890–897. doi:10.1145/1141911.1141971.

Bartels, R. H., Beatty, J. C., & Barsky, B. A. (1995). *An introduction to splines for use in computer graphics and geometric modeling*. Los Altos, CA: Morgan Kaufmann.

Beaudoin, P., Poulin, P., & van de Panne, M. (2007). Adapting wavelet compression to human motion capture clips. *Graphics Interface*, *2007*, 313–318.

Chattopadhyay, S., Bhandarkar, S., & Li, K. (2007). Human motion capture data compression by model-based indexing: A power aware approach. *IEEE Transactions on Visualization and Computer Graphics*, *13*(1), 5–14. doi:10.1109/TVCG.2007.13.

Cheneviere, F., & Boukir, S. (2004). Deformable model based data compression for gesture recognition. In *ICPR '04: Proceedings of the pattern recognition, 17th international conference on (ICPR'04)* (Vol. 4, pp. 541-544). Washington, DC: IEEE Computer Society.

Firouzmanesh, A., Cheng, I., & Basu, A. (2011). Perceptually guided fast compression of 3-d motion capture data. *IEEE Transactions on Multimedia*, *13*(4), 829–834. doi:10.1109/TMM.2011.2129497.

Gu, Q., Peng, J., & Deng, Z. (2009). Compression of human motion capture data using motion pattern indexing. *Computer Graphics Forum*, *28*(1), 1–12.

Ho, P. M., Wong, T. T., & Leung, C. S. (2005). Compressing the illumination-adjustable images with principal component analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, *15*(3), 355–364. doi:10.1109/TCSVT.2004.842601.

Hsieh, C. C. (2002). Motion smoothing using wavelets. *Journal of Intelligent and Robotic Systems*, *35*(2), 157–169. doi:10.1023/A:1021161132760.

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* (pp. 49–53). ACM. doi:10.1145/1111411.1111420.

Karni, Z., & Gotsman, C. (2004). Compression of soft-body animation sequences. *Computer and Graphics*, *28*, 25–34.

Khan, M. A. (2012). A new method for video data compression by quadratic Bézier curve fitting. *Signal, Image and Video Processing*, *6*(1), 19–24.

Khan, M. A., & Ohno, Y. (2007). Compression of video data using parametric line and natural cubic spline block level approximation. *IEICE Transactions*, *90–D*(5), 844–850.

Kwak, C. H., & Bajic, I. (2013). Mocap data coding with unrestricted quantization and rate control. In *IEEE international conference on acoustics, speech and signal processing (ICASSP)* (pp. 3741–3745). doi:10.1109/ICASSP.2013.6638357.

Lee, B. G., Lee, J. J., & Yoo, J. (2000). Representing animations by principal components. *Computer Graphics Forum*, *19*(3), 411–418.

Liu, G., & McMillan, L. (2006a). Compression of human motion data sequences. In *3DPVT* (pp. 248–255).

Liu, G., & McMillan, L. (2006b). Segment-based human motion compression. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 127–135). Aire-la-Ville: Eurographics Association.

Menache, A. (2010). *Understanding motion capture for computer animation* (2nd ed.). San Francisco, CA: Morgan Kaufmann.

Ormoneit, D., Sidenbladh, H., Black, M. J., & Hastie, T. (2001). Learning and tracking cyclic human motion. In: *NIPS* (pp. 894–900). The MIT Press.

Parent, R. (2009). *Computer animation complete: All-in-one learn motion capture, characteristic, point-based, and Maya winning techniques*. San Francisco: Morgan Kaufmann.

Rosenhahn, B., Brox, T., Cremers, D., & Seidel, H. P. (2007). Online smoothing for markerless motion capture. In *DAGM-symposium* (pp. 163–172).

Said, A., & Pearlman, W. (1996). A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, *6*(3), 243–250. doi:10.1109/76.499834.

Sattler, M., Sarlette, R., & Klein, R. (2005). Simple and efficient compression of animation sequences. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on computer animation* (pp. 209–217). ACM, New York, NY. doi:10.1145/1073368.1073398.

Shapiro, J. (1993). Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, *41*(12), 3445–3462. doi:10.1109/78.258085.

Shinoda, K., Kikuchi, H., & Muramatsu, S. (2008). Lossless-by-lossy coding for scalable lossless image compression. *IEICE Transactions*, *91-A*(11), 3356–3364. doi:10.1093/ietfec/e91-a.11.3356

Suzuki, T., & Ikehara, M. (2010). Integer dct based on direct-lifting of dct-idct for lossless-to-lossy image coding. *IEEE Transactions on Image Processing*, *19*(11), 2958–2965. doi:10.1109/TIP.2010.2051867.

Wang, C. W., & Jeng, J. H. (2012). Image compression using pca with clustering. In *International symposium on intelligent signal processing and communications systems (ISPACS)* (pp. 458–462). doi:10.1109/ISPACS.2012.6473533.

Wang, P., Pan, Z., Zhang, M., Lau, R. W. H., & Song, H. (2013). The alpha parallelogram predictor: A lossless compression method for motion capture data. *Information Sciences*, *232*, 1–10. doi:10.1016/j.ins.2013.01.007.

Xiao, J., Zhuang, Y., Wu, F., Guo, T., & Liang, Z. (2010). A group of novel approaches and a toolkit for motion capture data reusing. *Multimedia Tools and Applications*, *47*(3), 379–408. doi:10.1007/s11042-009-0329-1.

Zhao, X., Zhang, L., Ma, S., & Gao, W. (2012). Video coding with rate-distortion optimized transform. *IEEE Transactions on Circuits and Systems for Video Technology*, *22*(1), 138–151. doi:10.1109/TCSVT.2011.2158363.

Zhu, M., Sun, H., & Deng, Z. (2012). Quaternion space sparse decomposition for motion compression and retrieval. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation, SCA '12* (pp. 183–192). Aire-la-Ville: Eurographics Association. http://dl.acm.org/citation.cfm?id=2422356.2422383.

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 2001-02, 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 for three consecutive years from 2005-07, Japan.