Modeling and Compression of Motion Capture Data

Conference Paper · February 2017

CITATIONS
0

READS
14

3 authors, including:

Murtaza Khan
Umm Al-Qura University
23 PUBLICATIONS **137** CITATIONS
SEE PROFILE

Muhammad Arif
Umm Al-Qura University
128 PUBLICATIONS **750** CITATIONS
SEE PROFILE

Modeling and Compression of Motion Capture Data

Murtaza A. Khan
Department of Computer Science
Umm Al-Qura University
Saudi Arabia
Email: makkhan@uqu.edu.sa

Muhammad Arif
Department of Computer Science
Umm Al-Qura University
Saudi Arabia
Email: mahamid@uqu.edu.sa

Arshad Kamal
Department of Preparatory Year - Basic
Umm Al-Qura University
Saudi Arabia
Email: arshadhep@gmail.com

***Abstract*—Motion capture (MoCap) system uses sensors or markers, placed on human body joints, to record the movements of a human in space over time. Motion capture data is used in many entertainment applications such as in virtual reality environments to drive avatars, in video games to animate characters, in movies to produce CG effects, etc. In this paper, we present an efficient method for modeling and compression of motion capture data. The method uses quadratic Bézier curve fitting to smoothly model and compress the MoCap data. The temporal variation of MoCap data of each joint is approximated and parameterized using Bézier segments. Simulation results shows that our method uses smaller storage and better visual quality compared to other methods. The low degree of quadratic Bézier curve ensures computationally efficiency required for the real-time gaming applications.**

I. Introduction

Motion capture (MoCap) data is widely used in many multimedia and entertainment applications such as in online gaming [1], [2] and streaming animation [3] et al. Due to big size of MoCap data, a compact representation and storage that can support real-time applications is very important. In our work, we focused on modeling and compression of motion capture data using quadratic Bézier curve (QBC) fitting. A Bézier curve can model large number of data points with far less number of control points. This is essentially an approximate modeling and lossy compression technique. However, this loss is acceptable when taking into account the limitations of human visual system and high frame rate of motion capture data.

MoCap data is represented as hierarchies of joints parameterized by a translation and rotation. Joints are usually organized in a tree like hierarchical structure as illustrated in Figure 1. Consequently, when a joint moves, its connected joints lower in the hierarchy move too. For example, elbow is connected to wrist, which is connected to hand. Then as the elbow moves, the wrist and the hand 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 skeleton may have more than a hundred DOFs. Motion data of a joint in a sequence of frames can be considered as a function of time. Figure 2 shows rotational data of *right-wrist* joint in 148 frames. Let MoCap data consists of m channels and each channel has translation and/or rotation values in a series of n frames. Then we can store MoCap data in a matrix of size $n \times m$ as written in the Eq. (1).

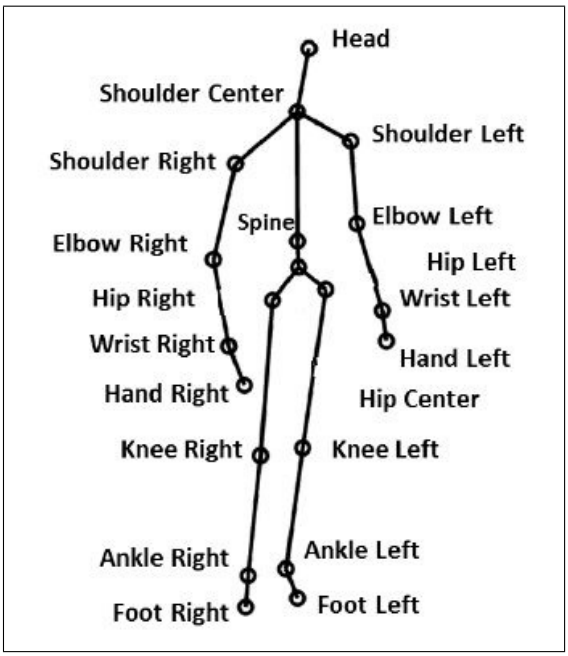

Fig. 1. Skeleton and joints.

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

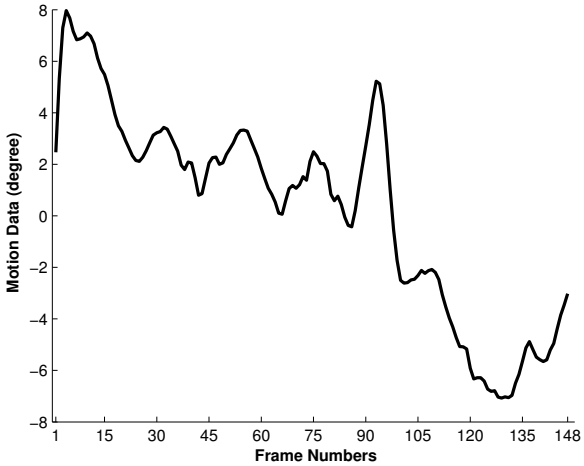

Fig. 2. Motion data of *right wrist* joint (rotation about X-axis) in 148 frames.

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

Organization of the rest of the paper is as follows: Related work is discussed in Section II. Compression of motion data using principal component analysis (PCA) and discrete wavelet transform (DWT) are briefly described in Section III and IV respectively. The mathematical model of quadratic Bézier curve (QBC) is described in Section V. The modeling and compression strategy of MoCap data via QBC is illustrated in Section VI. Selected simulation results are presented in Section VII. Section VIII analyzes results and gives insight view of the proposed method. In Section sec:makerspace, we discussed how our method can support and contribute to the success of Makerspace. Final concluding remarks are in Section X.

II. Related Work

Several authors [4], [5], [6], [7], [8], [9] et al. used Principal Component Analysis (PCA) to compress the motion data. [4] 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 [5], 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.

Discrete Wavelet Transform (DWT) based techniques suitable for compression of motion data are described by [10], [11] et al. DWT based method of [11] 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, [11] 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 good optimization results. A 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 then computes a 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.

A hybrid method to compress large databases of motion capture data is presented in [13]. 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 [14]. 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. A technique based on active contour fitting for compression of motion trajectories is investigated by [15]. 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.

III. Compression of MoCap Data by PCA

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, the second greatest variance on the second coordinate, and so on. We used PCA to reduce the number of channels of MoCap data by exploiting correlations among channels. From the input matrix of motion data, we determined covariance matrix, Eigen values, and Eigen vectors. We sort the Eigen values and corresponding Eigen vectors in order of decreasing Eigen values. Compression is achieved by retaining only q eigenvectors, where $q<m$. Note that for $q=m$, it will be lossless coding

IV. Compression of MoCap Data by DWT

Discrete 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 and some of the detail coefficients. In order to compress motion data by DWT we decomposed each channel data into approximation coefficients and detail coefficients up to DWT 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. Approximated data 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.

V. Quadratic Bézier Curve (QBC)

A quadratic Bézier curve (QBC) is a smooth 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. 3. 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:

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

where t_i is a parameter of interpolation, $0\leq t_i\leq 1$. In order to generate m points between P_0 and P_2 inclusive, the parameter

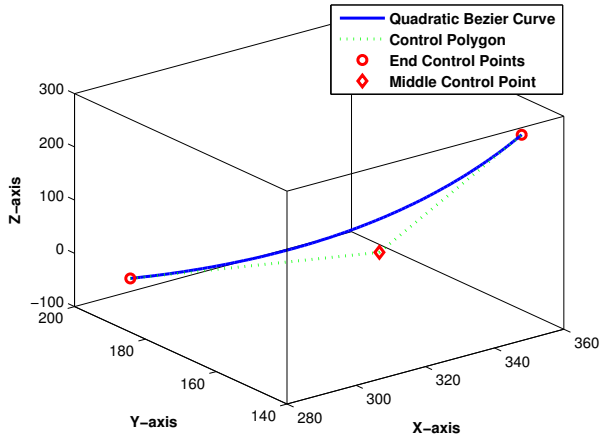

Fig. 3. A quadratic Bézier curve segment in 3-D space.

t_i is uniformly divided into $(m-1)$ intervals between 0 and 1 inclusive as follows:

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

$Q(t_i)$ 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.

VI. Compression of MoCap Data by QBC

This section describes the parametric representation (modeling) and compression of motion data using quadratic Bézier Curve (QBC) fitting. The process is applied to motion data of each joint individually. A joint data consists of at least one and at most six channels. Now we would describe the fitting process to motion data of a single joint.

Let input data 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 in a frame has three rotational channels (3-DOF), i.e., rotational values along X, Y and Z axes then a point is in Euclidean space R^3, i.e., $p = (\theta_X, \theta_Y, \theta_Z)$.

As an input to our 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 modeled by a parametric quadratic Bézier curve. Let a 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 quadratic Bézier curve, i.e.,

$$P_0 = p_1, \quad (4)$$

$$P_2 = p_m. \quad (5)$$

The *middle control point* (*MCP*) i.e., P_1 is obtained by least square method. Least square method 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:

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

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

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

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

$$\frac{\partial U}{\partial P_1} = 0. \quad (8)$$

Solving (8) for P_1 gives:

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

Equations (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 Eq. (2) and varying the value of parameter t_i uniformly in EQ. (2) using Eq. (3).

Same 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, $d_i^2 = |p_i - q_i|^2, \quad 1 \le i \le n$. Among all the values of d_i^2, we compute maximum squared distance, $d^2_{max} = Max\left(d_1^2, d_2^2, \ldots, d_n^2\right)$. If maximum squared distance of any j^{th} segment is greater than d^2_{lmt} then this segment is split and replaced with two new segments, j_1^{th} and j_2^{th}, i.e., $S = \left[\{S\} - \{S_j\}\right] \bigcup \{S_{j_1}, S_{j_2}\}$. 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} ($BP = \left\{p_1, \overbrace{p_{35}}^{bp_{new}}, p_{81}, p_{161}, \ldots, p_n\right\}$) and two new segments $\{p_1, p_2, \ldots, p_{35}\}$ and $\{p_{35}, p_{36}, \ldots, p_{161}\}$ replace S_1. 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^2_{lmt}. The same fitting process is applied to motion data of all the joints. Fig. 4 show quadratic Bézier curve least square fitting to motion data of *right femur* joint in Euclidean space R^3. The QBC output data to be stored for approximated motion data of each joint consists of:

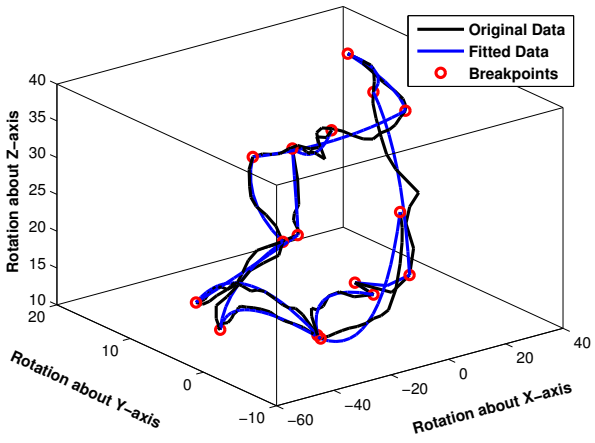

Fig. 4. Quadratic Bézier curve least square fitting in Euclidean space R^3 to motion data of *right femur* joint in 148 frames, $d^2_{max} = 8$. The joint has three channels (3-DOF), i.e., rotations about X, Y and Z axes.

1) Set of *control-points*, i.e., *CP*, where $CP_i = \{BP_i, MCP_i\}$, $BP_i = \{P_{0_i}, P_{2_i}\}$ for all the segments. Note that breakpoints *BP* are nothing but *ECP*, see Eq. (4) and Eq. (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 data.

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 j^{th} segment, $1 \leq j \leq r$, 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 Eq. (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}, $1 \leq i \leq m$, 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. We encoded/decoded the motion data 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 breakpoints are not equal for motion data of different joints. However, since initially we took Δ as initial breakpoint interval, this ensures that motion data of every joint has a breakpoint 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 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 playing it. Similar technique is used in video coding, where these frames are called I frames.

VII. Experiments and Results

We have tested above described methods on motion capture data obtained from [16] on several animations. Table I provides details of environment of experiments. Table II gives the details of selected clips. We compressed the motion data of each animation using PCA, DWT and QBC methods. Then in order to measure the performance of all the methods we computed Compression ratio (CR) and Mean squared error (MSE) as follows:

TABLE I. Environment of Experiments

Hardware	Intel Core i3-3217U, 1.8 GHz
Operating System	Windows 10 Home, 64-bit
MoCap data Processing	MATLAB R2012b
MoCap data Playback	MotionBuilder 2014

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

$$CR = \frac{f_o}{f_c}. \tag{10}$$

2) Mean squared error (MSE) between original motion data X_o and reconstructed motion data 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 Eq. (11), the motion data 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, and *maximum allowed square distance* for QBC. These input parameters are not related to each other. Therefore, it is very difficult to get the same value of MSE or CR for all the methods at given value of input parameters. Therefore, in order to evaluate performance of PCA, DWT 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. Fig. 5, 6, and 7 show the MSE vs. CR performance of PCA, DWT and QBC respectively for all the animations listed in Table II. The CR values of these methods is in different ranges for the same range of MSE. Therefore, we draw the distortion curves of PCA, DTW and QBC in separate graphs.

TABLE II. DETAILS OF MOCAP CLIPS. EACH CLIP HAS 31 JOINTS AND 62 CHANNELS. FRAME RATE 60-HZ, FILE FORMAT .ASF/.AMC.

Seq. No.	Seq. Activity	Frame Count	No. of data samples	File-size (bits)
38_01	*Jogging*	352	21824	2246890
10_03	*Soccer-kick*	362	22444	2305144
05_01	*Walking*	598	37076	3838930
143_37	*Climbing*	551	34162	3507090

In all the results, the values of principal component transform matrix, detail coefficients of wavelet transform and control points (*BP* and *MCP*) of quadratic Bézier curve are rounded to 2 decimal places. Wavelet detail coefficients are also run-length encoded in spirit of their characteristics. Final data is stored in MATLAB'S MAT-file format [17] for each method separately.

It is worth to mention that rather than comparing our results with statistics of other methods as reported in literature, we preferred to use our own implementation of PCA and DWT for comparison. This is due to lack of uniform standard of distortion measure, different simulation environments, selection of different input animations, the way final data is encoded, characteristics of file formats, etc. In our experiments, we selected the input animations with variety of motion activities with medium to high motion fluctuations. All the methods (PCA, DTW, and QBC) have same implementation and simulation environment and final data is encoded and stored with fairness. Nevertheless, the compression ratio (CR) of our method is very high and competitive to any method reported in the literature. Even for *Soccer-kick* animation, which has high motion activity, the CR of our method is greater than 45 at $MSE < 1$. For *Walking* animation, which has low motion activity, the CR of our method reaches to 64 at $MSE < 1$.

VIII. DISCUSSION

A. Rate distortion curves

Figures 5, 6, and 7 show the compression ratio (CR) vs. mean squared error (MSE) performance for PCA, DTW, and QBC respectively. In order to easily compare PCA, DWT, and QBC, we kept scaling of MSE axis same for all the methods. The QBC method yields best results, while DWT performs lesser than QBC and better than PCA. As an example, Table III 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, 42.975), (1.4065, 58.9308) for PCA, DTW, and QBC respectively for the *Climbing* animation. In the Table III, we choose the value of MSE for three methods (PCA, DWT, and QBC) as close as possible.

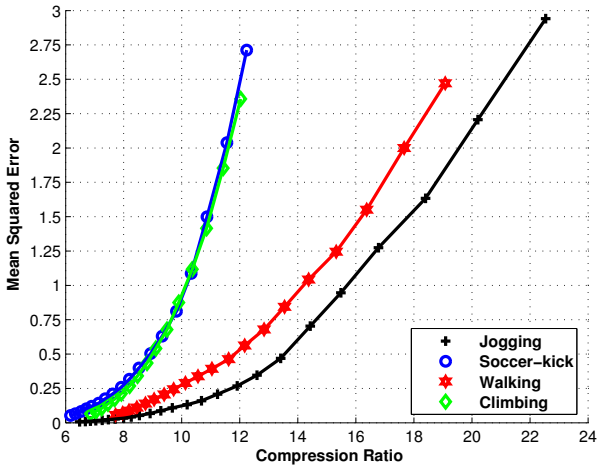

Fig. 5. *Compression ratio* vs *mean squared error* performance of PCA for all animations.

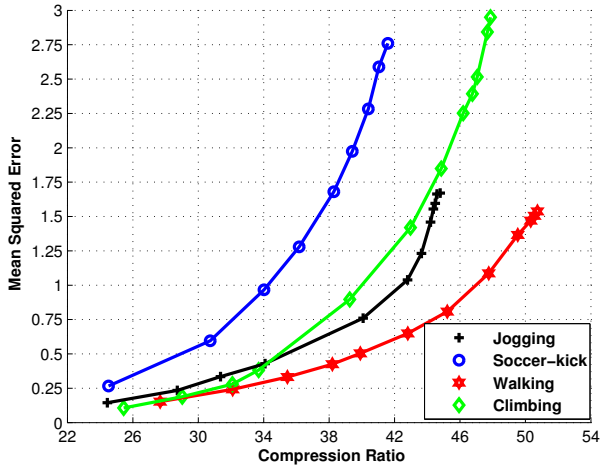

Fig. 6. *Compression ratio* vs *mean squared error* performance of DWT for all animations.

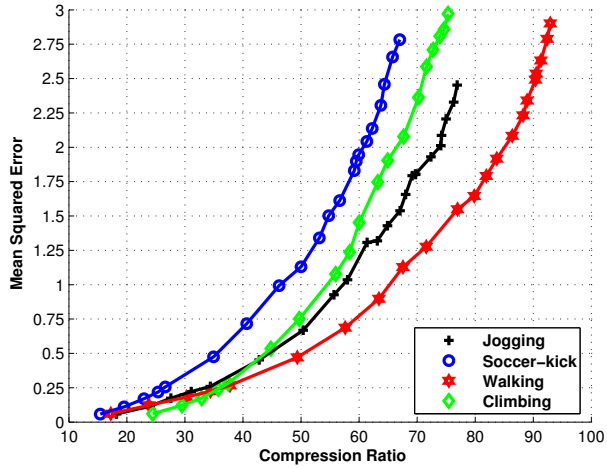

Fig. 7. *Compression ratio* vs *mean squared error* performance of QBC for all animations.

B. Subjective quality of reconstructed frames

Fig. 8, 9, 10 and 11 show arbitrary reconstructed frames for each of the animation described in the Table III. The frames are reconstructed by decoding the encoded data of QBC method.

C. Fitting data of each joint individually

Motion capture system records motion data of each joint individually. Therefore, it is more natural to parameterize the motion data of each joint separately and this is the approach

TABLE III. MEAN SQUARED ERROR (MSE) AND COMPRESSION RATIO (CR) COMPARISON OF ALL THE METHODS FOR A SINGLE OBSERVATION OF EACH ANIMATION. FOR PCA, q IS *Number of largest value Eigen vectors*. FOR DWT, t_w IS *wavelet threshold*. FOR QBC, d^2_{lmt} IS *maximum allowed square distance*.

Seq. No.	Animation Name	PCA			DWT			QBC		
		q	MSE	CR	t_w	MSE	CR	d^2_{lmt}	MSE	CR
38_01	*Jogging*	13	0.3466	12.593	4	0.3345	31.357	7	0.3297	37.944
10_03	*Soccer-kick*	22	0.2586	7.9199	3	0.2664	24.887	5	0.2552	26.606
05_01	*Walking*	13	1.0421	14.373	15	1.0857	47.848	24	1.0875	67.077
143_37	*Climbing*	17	1.4155	10.855	15	1.4196	43.097	28	1.4065	59.410

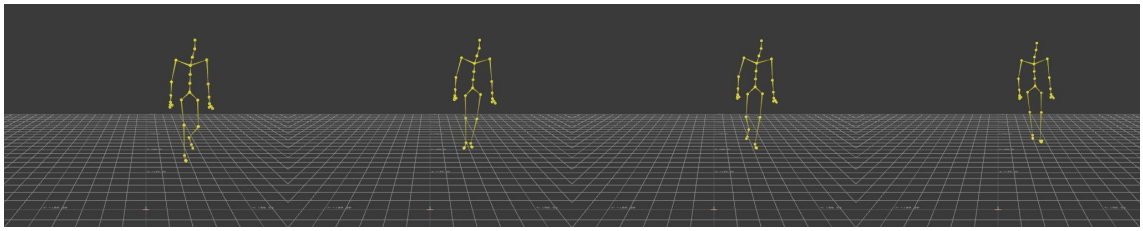

Fig. 8. Four arbitrary frames of 38_01 sequence reconstructed from decoded MoCap data.

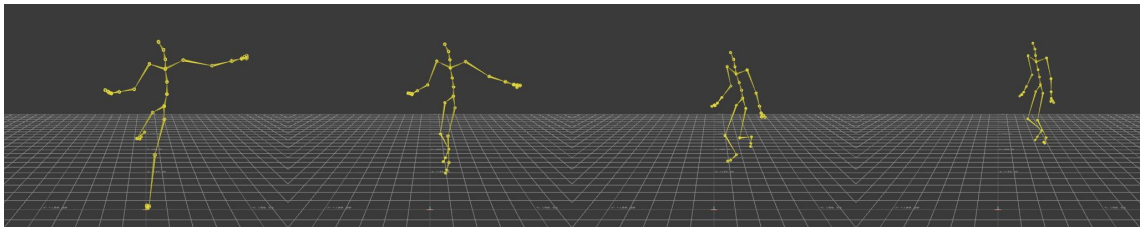

Fig. 9. Four arbitrary frames of 10_03 sequence reconstructed from decoded MoCap data.

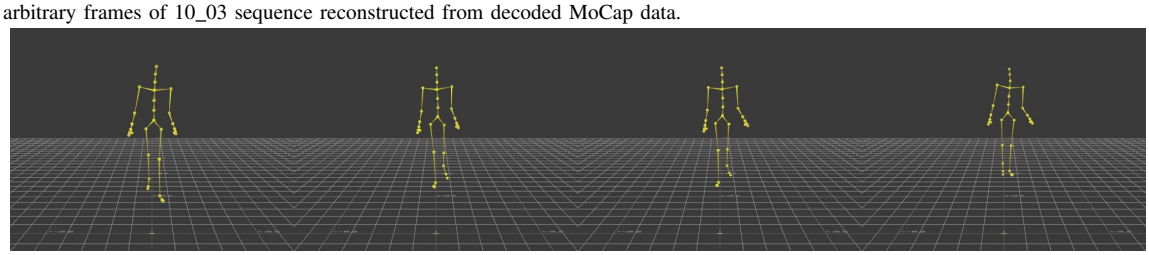

Fig. 10. Four arbitrary frames of 05_01 sequence reconstructed from decoded MoCap data.

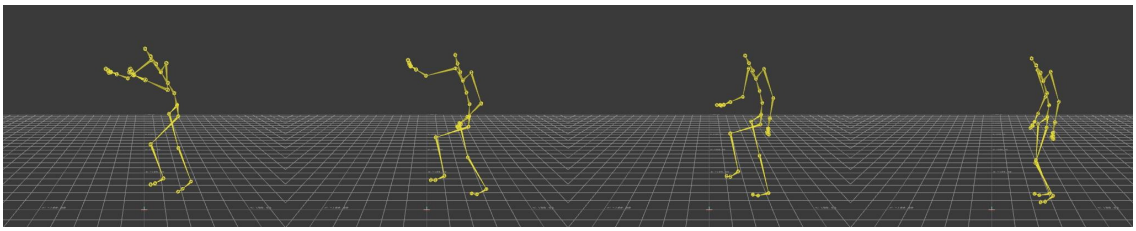

Fig. 11. Four arbitrary frames of 143_37 sequence reconstructed from decoded MoCap data.

we adopted. This approach also provides the flexibility to encode/decode the motion data 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.

D. Fitting in higher dimensional space

QBC fitting works well for high dimensional motion data as shown in Fig. 4. We fit the root joint in Euclidean space R^6 very well. However, it is difficult to graphically show the plot this fitting. Applying fitting to motion data in Euclidean space R^N is a novel approach

E. Time and Space efficiency of QBC

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

F. Limitations

At high value of *upper limit of error*, i.e., d^2_{lmt}, foot skating 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 less compression. Another solution for foot skating is to correct the foot contact during decoding. A technique presented by [18] automatically

detects and corrects foot skate. The technique of [18] 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.

IX. Support and Success of Makerspace

A makerspace is a community center that provides technology, manufacturing equipment and educational opportunities to the public. The free exchange of ideas and resources is a central tenet of makerspaces. Motion capture (MoCap) data is used in many fields such as in education, clinical medicine, video games, movies etc. We envisage that people from different fields get together at makerspace and use our method to model and compress MoCap data then use this data to build their applications of interests. At Umm Al-Qura university, we are working on a project that uses MoCap data in computer animation to teach Salah to children. This is an example, how our method can provide educational opportunities to public (children) at makerspace. At makerspace, a constructive discussion among people of different areas will further explore the opportunities of improving our method and using it for many more applications for the benefit of community. This will contribute to the success of makerspace.

X. Conclusion

We presented an efficient method for modeling and compression of motion capture (MoCap) data using quadratic Bézier curve. The method parameterizes and approximates the motion data of each joint independently. We used least square method to find optimal solution and break-and-fit strategy to enforce the upper limit of error. Initially a user has to set values of 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 proposed method performs better than principal component analysis and discrete wavelet transform methods. The subjective quality of reconstructed animation is also very good. The method has the limitation that it does not address the issue of foot skating at hight values of error-limit. Due to low computational cost and low space requirement the proposed method is suitable for modeling and compressing motion data for interactive and real time applications. We also highlighted how our method can support and contribute to the success of Makerspace.

References

[1] W. Cai, V. C. M. Leung, and L. Hu, "A cloudlet-assisted multiplayer cloud gaming system," *MONET*, vol. 19, no. 2, pp. 144–152, 2014. [Online]. Available: http://dx.doi.org/10.1007/s11036-013-0485-4

[2] L. Liu, A. Jones, N. Antonopoulos, Z. Ding, and Y. Zhan, "Performance evaluation and simulation of peer-to-peer protocols for massively multiplayer online games," *Multimedia Tools Appl.*, vol. 74, no. 8, pp. 2763–2780, 2015. [Online]. Available: http://dx.doi.org/10.1007/s11042-013-1662-y

[3] M. Patoli, M. Gkion, P. Newbury, and M. White, "Real time online motion capture for entertainment applications," in *Digital Game and Intelligent Toy Enhanced Learning (DIGITEL), 2010 Third IEEE International Conference on*, April 2010, pp. 139–145.

[4] G. Liu and L. McMillan, "Segment-based human motion compression," in *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 127–135.

[5] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computer and Graphics*, vol. 28, pp. 25–34, 2004. [Online]. Available: http://www.cs.technion.ac.il/ zachik/publications/pcalpc.pdf

[6] G. Liu and L. McMillan, "Compression of human motion data sequences," in *3DPVT*, 2006, pp. 248–255.

[7] B.-G. Lee, J. J. Lee, and J. Yoo, "Representing animations by principal components," *Computer Graphics Forum*, vol. 19, no. 3, pp. 411–418, September 2000.

[8] M. Sattler, R. Sarlette, and R. Klein, "Simple and efficient compression of animation sequences," in *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. New York, NY, USA: ACM, 2005, pp. 209–217.

[9] D. Ormoneit, H. Sidenbladh, M. J. Black, and T. Hastie, "Learning and tracking cyclic human motion," in *In NIPS*. The MIT Press, 2001, pp. 894–900.

[10] I. Cheng, A. Firouzmanesh, and A. Basu, "Perceptually motivated lspiht for motion capture data compression," *Comput. Graph.*, vol. 51, no. C, pp. 1–7, Oct. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.cag.2015.05.002

[11] P. Beaudoin, P. Poulin, and M. van de Panne, "Adapting wavelet compression to human motion capture clips," in *Graphics Interface 2007*, May 2007, pp. 313–318.

[12] J. Hou, L. P. Chau, N. Magnenat-Thalmann, and Y. He, "Human motion capture data tailored transform coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 21, no. 7, pp. 848–859, July 2015.

[13] O. Arikan, "Compression of motion capture databases," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 890–897, 2006.

[14] Q. Gu, J. Peng, and Z. Deng, "Compression of human motion capture data using motion pattern indexing." *Comput. Graph. Forum*, vol. 28, no. 1, pp. 1–12, 2009.

[15] F. Cheneviere and S. Boukir, "Deformable model based data compression for gesture recognition," in *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 541–544.

[16] Cmu, "Carnegie-Mellon Mocap Database," http://mocap.cs.cmu.edu/.

[17] http://www.mathworks.com/.

[18] L. Ikemoto, O. Arikan, and D. Forsyth, "Knowing when to put your foot down," in *I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, 2006, pp. 49–53.