Perceptually Motivated Real-Time Compression of Motion Data Enhanced by Incremental Encoding and Parameter Tuning

A. Firouzmanesh and I. Cheng and A. Basu

Department of Computing Science, University of Alberta, Canada

Abstract

We address the problem of efficient real-time motion data compression considering human perception. Using incremental encoding plus a database of motion primitives for each key point, our method achieves a higher or competitive compression rate with less online overhead. Trade-off between visual quality and bandwidth usage can be tuned by varying a single threshold value. A user study was performed to measure the sensitivity of human subjects to reconstruction errors in key rotation angles. Based on these evaluations we are able to perform lossy compression on the motion data without noticeable degradation in rendered qualities. While achieving real-time performance, our technique outperforms other methods in our experiments by achieving a compression ratio exceeding 50 : 1 *on regular sequences.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism— E.4 [Coding and Information Theory]: Data Compaction and Compression—

1. Introduction

Using dynamic human motion data has become commonplace in many applications including interactive games, virtual environments and computer aided dance training systems [DLGY11]. In an online environment the bandwidth that needs to be allocated to motion data can be expensive, especially when multiple dynamic characters are present in the scene. Motion data with rhythmic and/or repetitive nature, such as dance and martial arts, show a significant amount of spatial and temporal correlation and thus can be compressed effectively to reduce transmission load. An important issue is how much can be compressed without degradation of visual quality. We propose a perceptually motivated lossy compression method which not only achieves real-time performance with high compression rate and visual quality, but also allows applications to decide the trade-off between visual quality and bandwidth usage.

Similar to videos, the duration of motion sequences can vary. A training application may want to deliver the complete long sequence of yoga as an overview to the reviewer. In contrast, for online games the designer is likely to define short motion sequences which are selected for rendering depending on the player's interaction. To the best of our knowledge, insufficient research has been performed to distinguish motion sequences of different sizes. We hypothesize that the effectiveness of a compression algorithm may decrease depending on the size of the motion sequence transmitted. Based on our experiments we define large sequences as over 50 MB versus regular sequences.

Arikan [Ari06] proposed a motion compression method based on a combination of Bezier curves and clustered principal component analysis (PCA) that compress motion databases at a rate between 30 : 1 and 35 : 1. While their results are promising on large motion databases which are treated as a single sequence, the compression ratio is not as high when the technique is applied to individual motion clips of regular size. In an online environment, parallel distribution of smaller size subsequences is common in order to speed up interaction and reduce the adverse effect of packet loss. While large motion sequences are suitable for offline processing, our focus is on efficient real-time rendering of regular size motion sequences. Subsequent studies show that by using wavelets to compress the motion files one can achieve the same or better compression ratio without exploiting the correlation between joints (spatial correlation) [BPvdP07]. Their concept was to define a distortion metric that takes into account the difference between the positions of each joint before and after compression. They perform a global optimization by selecting the desired portion of all wavelet coefficients such that the distortion metric is minimized. The main drawback of this approach is that the global optimization process is computationally expensive. Firouzmanesh et al. [FCB11] proposed a wavelet based approach which takes perceptual factors of animation into account. Their user studies suggest that motion clips can

DOI: 10.2312/conf/EG2013/short/061-064

be compressed at a rate of at least 25 : 1 without significant loss in visual quality. While their results are promising, the drawback is the delay, e.g. 2 seconds for a 120 fps sequence, which is not suitable for smooth real-time visualization. Gu et al. [GPD09] compressed the movement of each marker separately using a database of primitives. Since each database has a fixed size and for each motion a separate database needs to be transmitted, the compression rate decreases as the number of motion sequences increases due to the higher number of databases. Their approach is suitable for large motion sequences. In contrast, our approach is designed for motion sequences of regular sizes. Although our method can be applied to any motion data, the performance is particularly efficient on motions with repetitive and/or rhythmic nature. Our main contribution include: (1) Taking advantage of perceptual factors to achieve more efficient compression, (2) Categorizing large and regular sequences, and applying our technique to compress the latter with real-time performance, and (3) Allowing applications to select the trade-off between visual quality and bandwidth usage by adjusting a single parameter value.

2. Proposed Method

Using a motion primitive database facilitates the exploitation of temporal redundancies in repetitive and/or rhythmic motion. While databases have been used in related work [GPD09], there are a number of important issues that need to be resolved before previous approaches can be applied to real-time dynamic motions: (1) In an interactive real-time setting, the spontaneous motions initiated by the user cannot be predicted beforehand. So a complete predefined database is not available and primitives have to be created on-the-fly. Thus, (2) in order to minimize the online processing time, we divide the compression process into offline and online phases. In the offline phase a reference primitive database is created based on a set of training motion data. A reference database copy is stored at the rendering site. (3) In the online phase, the motion is compressed segment by segment. For each segment the reference to the closest primitive in the database is transmitted. If the difference between the segment and its closest database primitive is above a certain visual threshold, we employ incremental encoding to transmit the compressed difference. Details on visual threshold are given in Section 3.

Motion data is produced by tracking the position of a series of key points on different parts of a subject's body. Using a hierarchical biped structure the motion can be represented by the position of the root marker in the world coordinates, plus each marker orientation in its parent's coordinate system. Assuming the biped has j joints, the input motion data is composed of a set of signals $(i) = (p(i), q_1(i), \ldots, q_j(i))$, where i is the frame number, $p(i) \in R^3$ represents the root position and $q_1(i), \ldots, q_j(i)$ are quaternions representing the orientations of the markers.

The goal of motion segmentation is to divide the movements of each joint into clips of similar length. We use zero-crossings of the angular acceleration at each joint as the chopping points [KPS03]. In order to facilitate matching of segments with the database primitives, a normalization process is applied to make segments of same length (number of frames). New frames to make up the length are generated using Spherical Linear Interpolation (SLERP).

The reference database is created by using a training set which contains a number of example motions. The compression can be higher if the transmitted motions resemble the training set motions. However, as can be seen in our experiments our approach is effective even if the motions to be compressed vary from the database training set. We achieve this efficiency by generating additional combinations based on the training primitives. The motion of each joint is segmented and normalized to create input segments. A variation of K-MEANS++ [AV07] clustering in quaternion space is applied to input segments to create primitives. The distance between two segments is defined as the maximum distance between pairs of corresponding quaternions. In order to find the best number of clusters (K) we start from an initial K (5 in our experiments) and increase it until the clustering error falls below a predefined threshold or the maximum number of clusters is reached. These two values are 0.01 and 25 in the current implementation as described in the Experimental Setting Section.

For compression, each segment is replaced with its closest database primitive. If the error is above a threshold (t_j) we encode the difference between the primitive and current segments using wavelet transform. Theoretically, this threshold (t_j) is defined separately for each joint and can be found by measuring the sensitivity of human observers to the maximum error in each joint. In practice, we focus on the joints which generate visually significant impacts.

A 4-level wavelet transform is then applied to the difference between the current segment and its closest primitive. We quantize the wavelet coefficients and set coefficients to zero if their absolute value is below a certain threshold. This threshold is selected in a way that the distance between the reconstructed and the original segment falls below t_j. The output stream is produced by applying runlength encoding and entropy coding. For each segment the decoder receives the reference to the closest primitives and wavelet encoded difference (when the error is significant).

The most important parameter affecting the compression ratio and quality is the maximum error at each joint (t_j). This threshold can be measured by user experiments. However, measuring for each joint is computationally expensive, and redundant if visually insignificant. Note that human viewers tend to be more sensitive to errors in the body parts in contact with a surface (feet in most cases) [Ari06]. The dependency between the joints in the upper body and lower body is low and the position of feet is mostly determined

Table 1: *Compression ratio (CR) of the proposed method for* $t_L = 0.01$ *(94_2: Salsa dance, 61_4: Indian dance, 79_17: Violin playing, 79_44: Washing a window, 135_4: Front kick, 138_2: Marching, 90_2: Cartwheel) with an average of* 48 : 1.

Motion	94_2	61_4	79_17	79_44	135_4	138_2	90_2
CR	62 : 1	51 : 1	52 : 1	50 : 1	40 : 1	44 : 1	38 : 1

Figure 1: *Screen shots of the seven regular length motion sequences used in our experiments.*

by the rotation angles of lower body joints. For this reason, we define two visually significant thresholds: one for the lower body (t_L) and one for the upper body joints (t_U). After experimenting with several different proportions between t_L and t_U, we achieved best results (in terms of quality and compression) by setting $t_U = 2 \times t_L$. In order to find t_L we performed a user study following the 2-alternative forced choice (2AFC) methodology using the two-up-one-down staircase algorithm [CB06]. Distorted motions were generated by setting t_j to t_L for all the lower body joints and to t_U for all the upper body joints. Our proposed compression and decompression methods are then applied.

3. Experimental Setting and Analysis of Results

We implemented the proposed method using the Microsoft Visual C++ 2008 platform. The input motion capture files were from the CMU database [CMU] in AMC format. The skeleton has 30 joints with a total of 56 Degrees of Freedom (DOFs) (53 rotational and 3 translational). Processing times are measured by running the program on a desktop computer with an Intel Core 2Duo 2.66 GHz processor and 2 GB of memory. For each type of motion the number of primitives in the database for each joint was limited to 25. Each wavelet coefficient was quantized to 10 bits using uniform quantization. The 53 rotational DOFs were converted to 30 quaternions (equal to the number of joints) and compressed using the proposed method. All the segments were normalized to 16 frames. The three translational DOFs were divided into segments of 32 frames. The Cohen-Daubechies-Feauveau (CDF) 9/7 algorithm was applied to each segment and the smallest 50 % coefficients in absolute value were set to zero. The coefficients were then quantized to 10 bits. Run-length and entropy encoding were applied.

In the study to predict human perceptual sensitivity towards reconstruction error, participants were required to complete a series of comparisons where motions were displayed side by side. In each pair, one of the motions was always the original and the other was a distorted motion. The order (whether to show the original motion on the left or right) was chosen randomly. We followed the procedure described in [Gar98] (where the author had extensive analysis on how to perform the staircase test) and started with a high distortion error value of t_L (0.024) and decreased t_L by 0.002 whenever the participant chose the correct (original) motion twice in a row. For a wrong answer, t_L was increased by 0.002. Participants were asked to choose the more visually appealing and smooth motion. If they were unsure they had to select one randomly. The converging point was set as the average of the last three reversals points [Gar98]. Sequences from five different motions (namely Salsa dance, Indian dance, walking, washing windows, front kick) were alternatively used in the test cases. Participants sat at a viewing distance of about 50 centimeters in front of a 19" wide screen monitor with resolution of 1440×900. The study was performed in a room with indoor incandescent lighting. There were 20 participants and each took between 20 to 25 minutes to complete the tests. All the participants were computer science graduate or undergraduate students. The average converging threshold was $t_L = 0.012$, while the smallest was $t_L = 0.01$ and the largest was $t_L = 0.016$.

We created a reference database using three training motion files, namely 61_04 (Salsa dance), 94_01 (Indian dance) and 2_2 (simple walk) from the CMU database. In order to evaluate our compression ratios, we compressed seven different motions (Figure 1), including Salsa and Indian dance, which were used in the training set. We also used other motions: violin playing, washing window, Karate kick, cart wheel and marching which were not included in the training set (Table 1). To compare with other methods, we use the performance reported in the literature. The best compression reported in [Ari06], [BPvdP07], and [GPD09] are 30 : 1, 48 : 1, and 25 : 1 respectively. Although the performance of Optimized wavelet [BPvdP07] is comparable to our approach, they achieve this result by compressing large sequences. The compression ratio would be lower on regular size sequences when each is compressed separately. For the violin playing motion, the compression ratio could reach 50 : 1 in [GPD09] when the uncompressed data size exceeds 50 MB. However, for smaller size (less than 20 MB) a compression ratio of 25 : 1 is reported. For similar motion, our compression rate can be maintained at 52 : 1 even when the data size is less than 1 MB. To summarize, comparison results show that our proposed method outperforms others on regular size sequences and is more suitable for compressing dynamic motions in real-time applications.

Given the available bandwidth, t_L can be adjusted accordingly. Table 2 shows the effect on compression ratio by varying the parameter t_L from 0.0025 to 0.025 with 0.012 being the average Just-Noticeable-Difference (JND) threshold obtained from our user study. By increasing t_L to the right of 0.012 (high compression), more viewers will be aware of the distortions. One advantage of our method is the capability to incorporate the JND model [CYB12], enabling applications to estimate the average percentage of viewers who will be satisfied with the rendered motions given a compression rate or t_L value.

Table 2: *The Compression ratio (CR) can be increased or decreased by adjusting a single parameter value t_L.*

$t_L \times 1000$	2.5	5	7.5	10	15	20	25
CR (Salsa)	36	42	45	51	60	69	75
CR (Indian Dance)	45	51	60	63	72	81	90

Using default parameters and $t_L = 0.01$ the compression time of the proposed method is between 1600 and 2400 μs per frame depending on the type and length of the motion and decompression time is around 120 μs per frame . The compression time is for online processing which is a deciding factor to support real-time applications and our results demonstrate such performance. The compression time reported for optimized wavelet coefficient selection method [BPvdP07] is between 133 and 483 ms per frame on a 2 GHz AMD Athlon 64 bit, depending on the number of frames in the animation, showing that our proposed method is more efficient in processor usage. The time for database creation (offline phase) varies greatly depending on the choice of training motions. The decompression time per frame is quite short, making our algorithm suitable for a wide range of devices with regular processing power. The bandwidth and storage required to transmit the primitive database are low. Using a maximum of 30 primitives per joint for each training motion, the compressed database occupies about 280 Kilo Bytes (KB).

An important consideration in real-time applications is the maximum delay required to reconstruct the compressed motion. In the proposed method this translates to the maximum number of frames in each segment (S_M). The delay can be controlled by reducing S_M generating a smaller size sequence. The delay can be controlled by reducing S_M generating a smaller size sequence. For motion 61_06 compression ratio would be 46, 49, and 50 when the SM is 16, 32 and 64 respectively. For motion 91_04 the compression ratio would be 58, 61, and 63 respectively. This shows that reducing S_M from 64 to 32 frames has insignificant effect on compression rate. For a 120 frames per second motion sequence the delay would be $120/32 = 0.27$ second, showing the effectiveness of our method on regular size motion sequences.

4. Conclusion

We proposed a real-time compression technique making use encoding of incremental primitives and perceptual tolerance to exploit temporal motion redundancy. Results show that our method, in comparison with other state-of-the-art techniques, can achieve higher compression while preserving visual quality, with less processing time. In addition, the rendering delay can be controlled by limiting the number of frames in each segment (S_M) without adverse impact on the compression rate. A distinct feature of our method is that trade-off between quality and bandwidth can be easily controlled by varying a single parameter (t_L).

In future work, we would like to evaluate the proposed method in interactive real-time training applications (like dance training systems [DLGY11]), and perform a larger scale user study to examine the effect of other perceptual parameters, including viewer's region of interest.

References

[Ari06] ARIKAN O.: Compression of motion capture databases. *ACM (TOG) 25*, 3 (7 2006), 890–897. 1, 2, 3

[AV07] ARTHUR D., VASSILVITSKII S.: k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (2007), pp. 1027–1035. 2

[BPvdP07] BEAUDOIN P., POULIN P., VAN DE PANNE M.: Adapting wavelet compression to human motion capture clips. In *Graphics Interface 2007 (GI'07)* (2007), pp. 313–318. 1, 3, 4

[CB06] CHENG I., BISCHOF W.: A perceptual approach to texture scaling based on human computer interaction. In *In Proc. EUROGRAPHICS, Short Paper* (2006), pp. 49–52. 3

[CMU] CMU GRAPHICS LAB: CMU graphics lab motion capture database. `http:http://mocap.cs.cmu.edu`. 3

[CYB12] CHENG I., YING L., BASU A.: Perceptually coded transmission of arbitrary 3D objects over burst packet loss channels enhanced with a generic JND formulation. *IEEE J. Selected Areas of Communication 30*, 7 (2012), 1184–1192. 4

[DLGY11] DENG L., LEUNG H., GU N., YANG Y.: Real-time mocap dance recognition for an interactive dancing game. *Comp. Anim. Virtual Worlds 22*, 2–3 (2011), 229–237. 1, 4

[FCB11] FIROUZMANESH A., CHENG I., BASU A.: Perceptually guided fast compression of 3-d motion capture data. *IEEE Trans. Multimedia 13*, 4 (8 2011), 829–834. 1

[Gar98] GARCÍA-PÉREZ M. A.: Forced-choice staircases with fixed step sizes: asymptotic and small-sample properties. *Vision Research 38* (1998), 1861–1881. 3

[GPD09] GU Q., PENG J., DENG Z.: Compression of human motion capture data using motion pattern indexing. *Computer Graphics Forum 28*, 1 (2009), 1–12. 2, 3

[KPS03] KIM T., PARK S. I., SHIN S. Y.: Rhythmic-motion synthesis based on motion-beat analysis. *ACM Trans. Graph. 22*, 3 (7 2003). 2