COMPUTER GRAPHICS forum

DOI: 10.1111/j.1467-8659.2008.01309.x

COMPUTER GRAPHICS forum
Volume 28 (2009), number 1 pp. 1–12

Compression of Human Motion Capture Data Using Motion Pattern Indexing

Qin Gu[1], Jingliang Peng[2] and Zhigang Deng[1]

[1]Computer Graphics and Interactive Media Lab, Department of Computer Science University of Houston, Houston, TX, USA
ericgu@cs.uh.edu; zdeng@cs.uh.edu
[2]Qualcomm, Inc., San Diego, CA, USA
jingliap@gmail.com

Abstract

In this work, a novel scheme is proposed to compress human motion capture data based on hierarchical structure construction and motion pattern indexing. For a given sequence of 3D motion capture data of human body, the 3D markers are first organized into a hierarchy where each node corresponds to a meaningful part of the human body. Then, the motion sequence corresponding to each body part is coded separately. Based on the observation that there is a high degree of spatial and temporal correlation among the 3D marker positions, we strive to identify motion patterns that form a database for each meaningful body part. Thereafter, a sequence of motion capture data can be efficiently represented as a series of motion pattern indices. As a result, higher compression ratio has been achieved when compared with the prior art, especially for long sequences of motion capture data with repetitive motion styles. Another distinction of this work is that it provides means for flexible and intuitive global and local distortion controls.

Keywords: motion compression, motion pattern, distortion control, motion capture

ACM CCS: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; E.4 [Coding and Information Theory]: Data Compaction and Compression

1. Introduction

Human motion capture data are increasingly used in many applications such as 3D animations, digital films, interactive games and medical simulations. Due to the rich styles of human motion, it is extremely difficult for animators to manually make up realistic human movements. Instead, in practice animators find it convenient to directly apply the motion data captured from real human beings to 3D human models. The motion capture data, however, consume a huge number of data bits in their raw format. For instance, more than 10 MB can easily be consumed by a 30-s motion sequence. In order to break the bottleneck of bandwidth and/or storage, it is essential to efficiently compress the 3D motion capture data.

In this work, we propose a novel motion data compression scheme that greatly reduces the redundancy in a motion sequence by utilizing the fact that the motion of a meaningful human part (e.g. arm, leg, etc.) usually exhibits similar patterns over the time. Figure 1 shows compression comparisons between our approach and the state of the art techniques. The major contributions of the proposed scheme include:

1. *Motion pattern indexing:* The proposed motion pattern indexing technique greatly reduces the redundancy in motion capture data. As such, the larger the original motion database is, the higher compression ratio this scheme can achieve, which is a distinction over the previous approaches that achieve approximately constant compression ratio, independent of the size of the original motion capture database. The proposed motion pattern indexing technique can also find applications in motion retrieval and motion recognition.
2. *Efficient coding performance:* Due to the techniques of hierarchical human body construction and motion pattern

Submitted December 2007
Revised April 2008
Accepted April 2008

Figure 1: *Results of our human motion compression approach and their comparisons with other approaches.*

indexing, great rate-distortion and compression ratio advantages have been observed over the prior art, especially for long sequences with repetitive motion styles.

3. *Flexible and intuitive distortion control:* In addition to the global distortion control which only specifies the average reconstruction error of entire body, the proposed scheme provides means for the local distortion control of particular body parts, which facilitates feature-based motion compression.

The remainder of this paper is organized as follows. A review of related works is given in Section 2. The proposed motion compression scheme is described in Section 3. Section 4 describes the global and local distortion controls. Section 5 describes the parameter trade-offs in our scheme. Experimental results are reported in Section 6. Finally, in Section 7, concluding remarks are drawn and future research directions discussed.

2. Related Works

Intensive research has been pursued on the compression of static 3D models. The state-of-the-art works in static 3D model compression include [AD01, AFSR03, GGH02, HPKG06, KG02, KSS00, KSW05, KV05, PK05, SK06]. In these works, the spatial correlation among the vertices or points in a 3D model is usually utilized to reduce the entropy of data.

Later on, following the work by Lengyel [Len99], many works were proposed to encode animated mesh sequences. They typically utilize the spatio-temporal correlation in an animated mesh sequence in order to achieve coding efficiency. Works along this line include [ZO04, ZO05] that use the octree-based motion representation, [IR03, YKL02] that perform vertex-wise motion prediction, [AM00, KG04, SSK05] that employ the principal component analysis (PCA) technique, BSM*03] that generalizes the geometry image coding [GGH02, GK04] that adopts wavelet coding techniques and [YKL04, YKL06] that support coding of non-isomorphic animated mesh sequences. For a comprehensive survey on 3D mesh compression and animated-mesh coding technologies, we refer the readers to [PKK05].

In 3D animation production, human motion capture data are widely used to generate realistic sequences of animated meshes. In this context, it is often more economical to compress the motion data rather than the animated mesh sequences. Besides, the stored motion data can be reused to generate other animated mesh sequences. A number of techniques have been proposed to efficiently compress human motion capture data [Ari06, BPvdP07, CBL07, CCW*04, LM06]. Arikan [Ari06] proposed the seminal work in motion capture data compression. In his work, Bezier curves are used to represent motion clips, and clustered principle component analysis applied to reduce their dimensionality with minor loss of visual quality. Environmental contacts are encoded separately based on the discrete cosine transformation (DCT). Liu and Mcmillan [LM06] proposed a data-driven approach that first segments a motion data sequence, then applies PCA to each motion data segment such that an approximation with reduced dimensionality is derived and coded. In addition, they employed key frame selection to find the frame with sharpest coordinates change of motion and spline interpolation techniques to further improve the coding performance. Motion data indexing concept was also exploited for the purpose of retrieval [CCW*04, FF05, MR06] or compression [CBL07]. Forbes and Fiumes [FF05] use a K-means clustering approach for the purpose of motion retrieval. In this approach, the clustering is performed at a weighted-PCA space, not on semantic separate human parts. In the work proposed by Chattopadhyay *et al.* [CBL07], based on the statistical distribution of the floating point numbers in a motion matrix, each floating point number is represented as an integer index, which takes much less data bits. This algorithm is good for low-resolution coding of motion capture data, and achieves fast decoding with quick table lookups. However, when higher resolutions are demanded, the R-D performance deteriorates quickly.

In the above-mentioned human motion capture data compression algorithms, the structure of a human body is not explicitly exploited, and no means are provided for local distortion control, (e.g. the 3D residuals cannot be controlled for specific markers.) In our work, we propose to explicitly utilize the human body structure, and thereafter identify motion patterns corresponding to each body part in order to reduce the data entropy. Because of the explicit utilization

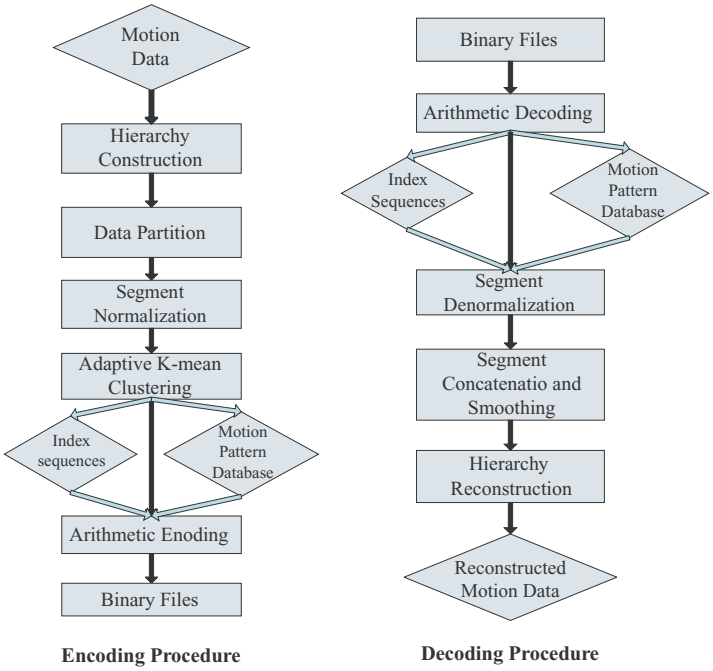

Figure 2: *Pipeline of the encoding/decoding procedure of our approach.*

of human structure, local distortion control, in addition to global distortion control, is achieved.

3. Our Approach

Figure 2 shows the pipeline of the encoding and decoding procedures of our approach. In the encoding procedure, first a hierarchical structure will be constructed, where each node contains motion markers corresponding to a meaningful part of the human body. For each node in the hierarchy, the motion paths of all markers in it form a motion sequence specific to that node. Then, for each node, the corresponding motion sequence is partitioned into motion segments at certain chopping points that are determined by an adaptive segmentation algorithm. After that, a normalization procedure transforms all motion segments using a time-warping technique such that they have the same number of frames. In the motion pattern extraction step, the adaptive K-means clustering is performed which treats a normalized motion segment as a high-dimensional vector (obtained by concatenating 3D positions of all markers in all frames) and calculates the mean of each motion segment cluster. Each of the mean values resulting from the K-means clustering process is stored as a motion pattern in a motion pattern database. Thereafter, each motion segment is represented by an index to the motion pattern database. Finally, the resulting index sequences and the motion pattern database are are compressed using entropy coding techniques. The decoding procedure (as illustrated in Figure 2) is basically the reverse of the encoding procedure.

3.1. Hierarchy construction

In general, the original motion data acquired from a motion capture system contain absolute 3D coordinates of markers at each frame as ASCII text files, which are often of big magnitude and therefore not good for efficient storage. In addition, the absolute coordinate representation is formed in the global world frame, instead of directly oriented for motion pattern identification. For instance, when a person walks, although the absolute coordinates of the markers on an arm can be highly variant, their coordinates relative to the main body typically exhibit highly repetitive patterns. Furthermore, if we only focus on the motions of a certain body part, the probability of finding similar local motion patterns in different motion sequences is significantly increased. Figure 3 shows a specific example of two different human motion sequences with high local similarity at two arms. As such, a better way to store the data is to represent the data in a structural scheme. In this work, a hierarchical structure is built to represent the motion data, where each node corresponds to a meaningful part of the human body and typically contains multiple motion markers. The positions of all markers in a node are represented as coordinates relative to a reference marker in their parent node.

In this work, we construct a five-level hierarchical structure as illustrated in Figure 4. First, a specific marker close to the centre of body is chosen as the root marker. In this work, the marker behind the top spine is chosen as the root marker, and its absolute position is recorded for each frame in the

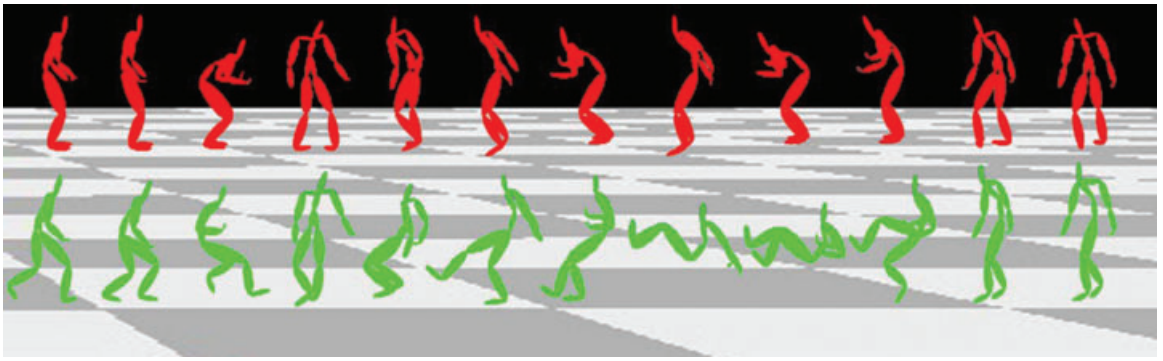

Figure 3: *Two different motion sequences with high local similarity at two arms. Both motion capture sequences are taken from the CMU motion capture database [cmu07].*

sequence. In the hierarchy, the root node of the hierarchy only contains the root marker; the second-level nodes contain the markers corresponding to the head, the left shoulder, the torso and the right shoulder, respectively; the third-level nodes contain the markers corresponding to the left arm, the left hip, the right hip and the right arm, respectively, and so forth. The parent–child relationship between nodes are illustrated in the right part of Figure 4. In real human movements, the movement of a child node is typically constrained by the position of its parent node. In our work, we choose as the reference marker, for each node, the one close to the joint with the child node(s). The positions of markers in a node are represented as coordinates relative to the reference marker of the parent node. It should be noted that the above hierarchy construction process is manually performed based on the first frame in the motion sequence. An automatic hierarchy construction algorithm, although generally more efficient, often leads to less semantic accuracy and hence less coding efficiency. Therefore, a manual approach is adopted in this work, driven by the goal of high coding efficiency. Interestingly, manual interaction is also adopted in [Ari06] to specify the environment contacts for a better visual quality of the reconstructed motion.

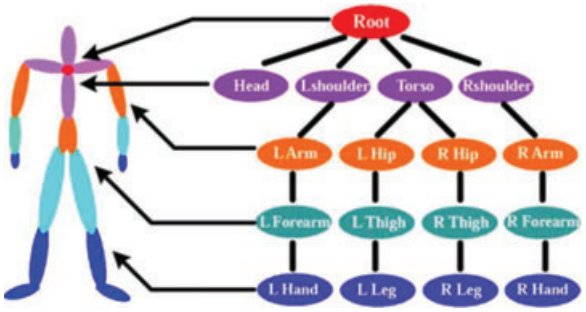

Figure 4: *Hierarchy structure for the human body skeleton.*

3.2. Segmentation and normalization of motion data

For a given motion sequence corresponding to a node in the hierarchy as constructed in Section 3.1, we first partition it to shorter motion segments at chopping points that correspond to sharp changes in the human motion. By doing this, we expect that each resultant segment contains smooth motion, and that a high degree of similarity in motion exists among the resultant segments. In this work, an adaptive algorithm is used to automatically detect optimal *chopping points* in a motion sequence. Basically, a metric is introduced to measure the smoothness of motion change when the next frame is considered. If the metric is larger than a predefined threshold, then a chopping point is detected and a new motion segment is initiated; otherwise the frame is added to the current motion segment. Algorithm 1 describes the details of this algorithm. Here, $AveFrameDiff(S_i)$ is the averaged difference between two consecutive frames of the ith motion segment S_i, and $LastFrm(S_i)$ is the last frame of the ith motion segment S_i.

Algorithm 1 Motion Data Partition

Input: $F[1...N]$, motion frames for one motion node.
Input: W, a weight vector (for markers).
Input: θ, a predetermined threshold.
Output: $S_1, S_2, \ldots S_k$, outputed motion segments

1: idx = 1;
2: $S_{idx} = \{F_1\}$;
3: **for** $i = 2$ to N do
4: **if** $\frac{W^T * |F_i - LastFRM(S_{idx}|}{W^T * |AveFrameDiff(S_{idx})|} \leq \theta$ then
5: $S_{idx} = S_{idx} + \{F_i\}$;
6: **else**
7: Output S_{idx};
8: $idx = idx + 1$;
9: $S_{idx} = \{F_i\}$;
10: **end if**
11: **end for**

The above partitioning algorithm takes as parameters the threshold θ and the weight vector W. A smaller θ value typically results in a larger number of resulting motion segments, which costs more coding bits but leads to better approximation quality, and vice versa. Hence, this parameter is used to balance the trade-off between compression ratio and compression quality. The weight vector W controls the relative importance of markers. If a marker is particularly important, it can be assigned a higher weight, and its motion will be better preserved in the decoded motion data. Section 4 will describe in details how to use this parameter to perform local distortion control. It is noteworthy that the above motion segmentation algorithm runs separately for each hierarchical node. In other words, different nodes can have different sets of chopping points.

Due to the adaptivity of the above motion data partition algorithm, the thus-generated motion segments typically have different numbers of frames. In preparation for our motion pattern extraction algorithm as detailed in Section 3.3, we apply a normalization procedure on all motion segments such that they will have the same number of frames, which we call reference frame number, after the normalization. In this work, the average number of frames in each motion segments is taken as the reference frame number. Linear interpolation and linear smoothing are applied to generate new frames during the process of motion segment normalization.

3.3. Motion pattern extraction

A motion pattern is essentially a representative motion segment for a node in the hierarchy as constructed in Section 3.1 (corresponding to a meaningful part of the human body). In order to adaptively construct a motion pattern database for each node, we need to measure the difference between the corresponding motion segments and calculate the representative motion segment. As mentioned in the above sections, normalized motion segments have the same frame number. We convert a normalized motion segment to a high-dimensional vector by concatenating the 3D positions of markers of all frames in the motion segment.

Our motion pattern extraction algorithm, essentially a vector quantization algorithm, can be described as follows: starting from an initial K value, the K-means clustering algorithm [HRF01] is applied to a group of high-dimensional vectors (normalized motion segments), and then an error metric is computed to measure the quality of the clustering. If the error is larger than a specified threshold, then the number of clusters, K, is increased and the K-means clustering is applied again. This iteration process continues until the error goes below the threshold. The statistical average of each cluster will be stored as one motion pattern in a database. Two options can be adopted as the error metric: the first is the average clustering error (Eq. 1), and the second is the maximum clustering error (Eq. 2).

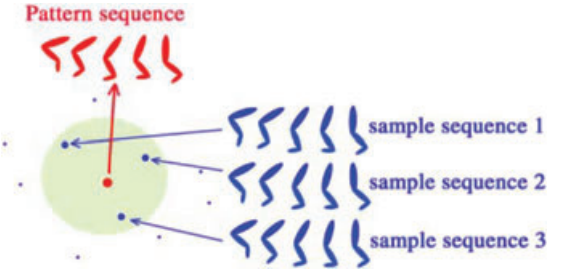

Figure 5: *Motion pattern extraction through an adaptive K-means clustering for the left leg motion. The red point (motion segment) will be stored as a motion pattern.*

$$AveError = \frac{\sum_{i=1}^{K} \sum_{x_j \subset C_i} |x_j - \mu_i|^2}{\sum_{i=1}^{K} Num(C_i)}, \tag{1}$$

$$MaxError = \max_{i=1\ldots K} \max_{x_j \subset C_i} |x_j - \mu_i|. \tag{2}$$

In the above equations (Eqs. 1 and 2), μ_i is the cluster mean, C_i represents the set of the ith cluster, x_j is a data point of the cluster C_i, and $Num(C_i)$ is the number of data points enclosed in the cluster C_i. In our experiments, we found that the maximum error metric works better although it is slightly more time-consuming. In addition, the maximum error metric provides a natural way for local distortion control (as detailed in Section 4). Figure 5 illustrates the idea of motion pattern extraction using the adaptive K-means clustering for the left leg.

After the motion pattern database for each node in the human body hierarchy is generated, each motion segment of that node is represented as an index into the database and the number of original frames it contains before normalization.

3.4. Motion sequence coding

In this work, we employ the arithmetic coding technique – one type of entropy coding techniques – to compress both the motion database and the motion pattern indices. Typically, an entropy coding technique uses different numbers of bits to represent different symbols in an input sequence, with more/less bits assigned to less/more frequent symbols. According to Shannon's source coding theorem, the optimal number of binary coding bits for an input symbol is $-log_2 P$ where P is the probability of that input symbol. Huffman coding and arithmetic coding are two commonly used entropy coding techniques, of which arithmetic coding often approaches the optimal code length more closely. For the detail of Shannon's source coding theorem and entropy coding techniques, please refer to [Mac02].

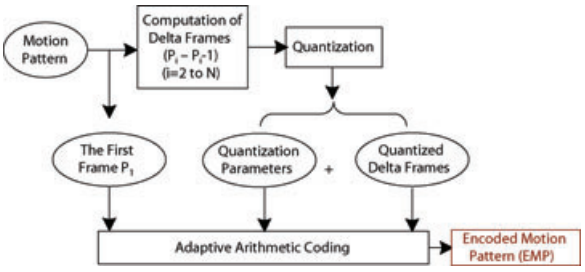

Figure 6: *Coding of motion pattern.*

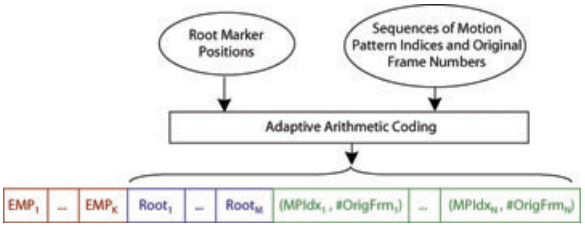

Figure 7: *Coding of a motion sequence.*

For each node in the human body hierarchy, the corresponding motion sequence is encoded in several steps. First, all motion patterns in the corresponding motion pattern database are encoded; then, the absolute coordinates of the root marker in all the frames are encoded; finally, for each motion segment, the motion pattern index and the number of original frames are encoded.

The work-flow of the motion pattern coding process is illustrated in Figure 6. For a motion pattern that is composed of multiple motion frames, we first entropy code the 3D marker coordinates in the first frame. Then, we compute a delta frame for each pair of adjacent motion frames, which is composed of delta coordinates of all the 3D markers. Next, we perform a uniform quantization to all the delta coordinate values in all the delta frames, and the corresponding quantization parameters are entropy coded. In this work, eight-bit quantization is applied to the delta coordinates. Finally, the quantized delta coordinate values are entropy coded. Any motion pattern in the motion pattern database is encoded in the same way.

Having encoded the motion pattern database, we then encode the root marker positions in all the motion frames, and for each motion segment we encode its motion pattern index and its original frame number. As illustrated in Figure 7, the bitstream of the encoded motion patterns ($EMP_1 \sim EMP_K$) is followed by that of the encoded root marker positions ($Root_1 \sim Root_M$), where K and M are the total number of motion patterns and the total number of motion frames, respectively. Thereafter, for each of the N motion segments, two integral numbers, $MPIdx_i$ and $\#OrigFrm_i (1 \leq i \leq N)$, are entropy coded where $MPIdx_i$ and $\#OrigFrm_i$ stand for the motion pattern index and the number of original frames, respectively. In this work, the adaptive arithmetic coding algorithm in [ari07] is used as our entropy coder.

3.5. Motion reconstruction and smoothing

The decoding and motion reconstruction process is essentially a reverse of the encoding process as described in Section 3.1. For each node in the human body hierarchy, the corresponding motion pattern database is first decoded; then the sequence of root marker positions are decoded which are used to restore the global positions of the other markers; finally a pair of ($MPIdx_i$, $\#OrigFrm_i$) is decoded for each motion segment, from which the corresponding motion data pattern is retrieved from the database at index $MPIdx_i$ and denormalized to have $\#OrigFrm_i$ frames.

In the decoding procedure (Figure 2), when reconstructed motion segments are concatenated together, there might exist non-smooth jumps between two motion segments due to the lossy nature of the encoding process. As such, a spline filter [BW95] is applied to the boundary area of two neighbouring motion segments as a smoothing operator.

4. Distortion Control

Distortion control is, in general, an important issue in the context of lossy data compression. This proposed scheme is a lossy compression scheme, because several intermediate steps will cause information loss, including the normalization/denormalization, the clustering, and the motion data quantization. In order to achieve high level intuitive controls over the compression loss in the encoding/decoding procedures, the proposed scheme provides means for both global and local distortion controls.

Rate-distortion is a widely adopted metric of coding performance in the literature of data compression, which measures the distortions of decoded data, as compared with the original, at different bit rates. In our work, distortion is measured as the average distance between the reconstructed and the original positions of each 3D motion marker . Global distortion is obtained by measuring the distortion over all the 3D motion markers placed on a human body, while local distortion is obtained by measuring over a subset of the 3D motion markers.

In order to find the optimal balance between compression quality and compression ratio, our global distortion controls are provided through the threshold parameters in motion data partition and motion segment clustering steps. These parameters can control the overall compression quality: tightening these threshold parameters can increase the quality of reconstructed motion, with sacrificed compression ratios, and vice versa.

However, these global distortion controls take effects from a general perspective, and they do not provide fine controls over the accuracy of the reconstructed motions of a particular human part or a particular group of markers. In human body motion capture applications, often users need accurate controls over certain human parts or markers, e.g. contact constraints. For example, in human walking motion sequences, it is generally required that the feet exactly touch the floor at some frames. This scheme provides fine local distortion controls to maintain contact constraints. As mentioned in the motion data partition step, the weight vector can control the importance of different markers. Increasing the weights of contact markers can improve the accuracy of their reconstructed motions. In the segment clustering step, users can use the MaxError metric (instead of the AverageError metric) to adaptively cluster motion segments, which essentially specifies the maximum distortion tolerance in this step. In our experiments, we found that the AveError metric (Eq. 1) is a good distortion metric for relatively simple motions like walking and running. When long, complicated motion sequences are involved, we found that the MaxError metric provides a better distortion metric since the offset between any single outlier and its centre is taken into account.

Figure 8 shows an example how local distortion controls affect compression quality. From this comparison figure, we can see that, by increasing the weight of a marker or a group of markers, the reconstruction accuracy of the markers is measurably increased (more closer to the original trajectory).

In general, by assigning higher weights for certain markers, we can achieve better visual reconstruction results for specific markers with slightly sacrificed overall compression ratios. In rare cases, however, if the reconstruction accuracy of certain markers needs to strictly enforced, these markers can be singled out and compressed separately using the Discrete Cosine Transform technique, at the expense of extra storage [Ari06].

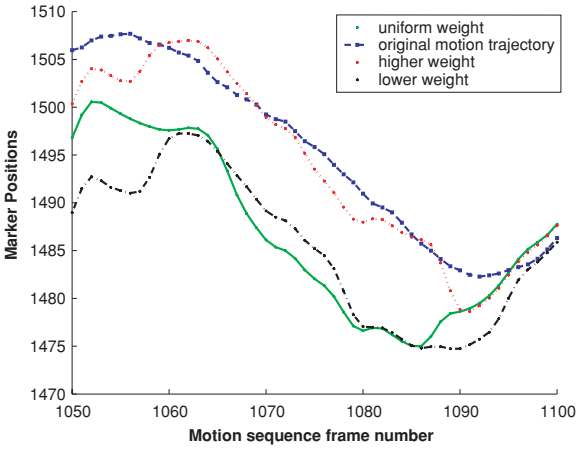

Figure 8: *The Z-trajectories of a reconstructed marker (a shoulder marker). Here, different weights are assigned to this particular marker (to emphasize its importance). 'Uniform weight assignment' means all markers has the same weight (=1), 'lower weight assignment' means this marker is given a lower weight (=0.1) and other markers keep the same weight (=1), and 'higher weight assignment' means this marker is given a higher weight (=10). Other key parameters used for this comparison are: key-frame selection threshold =0.1, initial number of clusters = 10, and clustering distortion threshold = 3. This motion capture sequence is taken from the CMU motion capture data library [cmu07].*

5. Parameter Trade-Off

In our scheme, several major parameters are used for the purpose of balancing compression ratio and quality (reconstruction accuracy). Here, we describe the trade-off and practical considerations on these parameters.

1. *Key-frame selection threshold:* This parameter is used in the adaptive key-frame selection step. If the difference between current motion frame and previous key-frame is larger than this threshold, the current motion frame will be considered as a new key-frame and the start of a new motion segment. However, it's not feasible for users to preset a fixed value to this parameter due to the variety of the coordinate systems used in different motion sequences. Thus, our algorithm automatically evaluates the Average Diagonal Distance of the Head (ADDH) by computing the average distance between diagonal head markers for the initial frames (e.g. 200 frames) of any given motion sequence. There are four markers on the head (square distribution) for every capture subject. We use two markers on the diagonal, and compute the average distance between them over time, as our segmentation metric due to the fact that these distances on the head always have least variance from the entire motion. Based on the ADDH, users just need to set a coefficient to determine the threshold parameter. For example, if this coefficient is 0.5, it means the key-frame selection threshold is a half of the ADDH. In the paper, we set this parameter to 0.1 as shown in result figures. In our experiments, setting this coefficient between 0.05 and 0.5 generally yields visually convincing results as well.
2. *K-means clustering parameters:* These parameters include the initial number of clusters and the clustering distortion threshold. The iterative process of motion segment clustering starts with an initial number of clusters and ends when the clustering error is below the clustering distortion threshold that is defined as the upper bound of distance (*MaxError* defined in Eq. 2) between any data point and its corresponding cluster centre. A larger number of initial clusters and/or

Table 1: *ARMS error of various test motion capture data sets using our approach.*

Motion sequence	Original size (MB)	Segments	Clusters	Compressed size (MB)	ARMS error	Compression ratio
Short jumping motion	10.12	2724	121	0.41	6.35	25
Long jumping motion	530.12	154 262	3816	12.62	4.12	42
Short combined motion	10.30	3017	154	0.45	6.48	23
Long combined motion	592.42	169 845	6126	21.16	5.32	28

The combined motion is mixed with different types of motion sequences, such as walking, jumping, climbing, dangling and sit drop. All the motion sequences are taken from the CMU motion database [cmu07]. Number of Segments and Number of Clusters include the results from all body parts. Here, the used parameters are: key-frame selection threshold =0.05, initial number of clusters = 10, clustering distortion threshold = 1, and even distortion control weights.

a smaller clustering distortion threshold tend(s) to produce higher visual quality and lower compression ratio due to the increased size of the motion pattern database, and vice versa. Based on our experiments, an initial number of clusters in the range of 10–100, and a value of clustering distortion threshold in the range of 1–5% (relative to the ADDH) lead to high compression ratios with visually convincing quality as well.

3. *Distortion control weights:* The weight distribution over markers controls the accuracy of reconstruction for different human parts. In Algorithm 1, given a fixed value of the threshold θ on the right hand side, if the weight for a certain marker is larger, the absolute coordinate change of this marker (the second operand on the left hand side including numerator and denominator) has to be smaller to satisfy the segmenting criterion (θ), which means shorter length for each segment but larger amount of segments. Thereafter, more indexes are created during clustering to identify the compressed motion. As a result, higher reconstruction accuracy will be achieved at the expense of more storage requirement. By default, the weights of all markers are the same.

6. Results

To measure the reconstruction errors caused by our compression scheme, we first define an average RMS error metric – ARMS (Eq. 3). Table 1 shows ARMS error of various test motion capture data sets. As we can see from this table, our scheme is quite stable in terms of ARMS error. In Table 1, more than 900 motion capture sequences from the CMU motion capture database [cmu07] (530 MB for single type motions and 592 MB for complex motions) were used as our test data set.

$$ARMS = \frac{\sum_{j=1}^{FrmNum} \sqrt{(\sum_{i=1}^{MrkNum} ||P_i - \hat{P}_i||^2)/MrkNum}}{FrmNum}. \quad (3)$$

Various motion capture sequences of multiple subjects (total size is 529 MB) from the CMU public motion capture data library [cmu07] were used as our test data set. Based on the reconstructed motion trajectory curves, we compared the ARMS error of our approach with other motion compression approaches including global PCA method and the state of the art compression technique [Ari06] in Figure 9. In the global PCA method, the number of retained PCA coefficients is determined by the sum of retained corresponding eigen values (in this comparison, the sum of retained corresponding eigen values is greater than 95% of the sum of all eigen values). Furthermore, we also compared the compression ratio among the above approaches, sub-sampling method, and two popularized data compression tools (WinZip and RAR). Figure 10 illustrates the results from processing human motion data concatenated by various human motion types from CMU subject 1, 2, 5, 6, 9, 12, including walk, jump, climb, dangle and tai chi. The sub-sampling method uses 12 as a sub-sampling factor, i.e., every 12 frames, one frame will be kept in the compressed files.

As we can see from Figure 10, the compression ratio achieved by our approach is pleasing, furthermore, when the size of original motion capture database becomes larger (along X axis), the compression ratio achieved by our scheme climbs up. Meanwhile, the compression ratios achieved by some other approaches (sub-sampling, ZIP and RAR) approximately keep at a constant level. When the size of motion database is small, our method is comparable to the Arikan's method [Ari06], in terms of compression ratio and reconstructed visual quality. However, when the size of motion database is increased, our method tends to achieve higher compression ratios without sacrificing its visual quality. This distinctive feature (higher compression ratio for larger motion database) is particularly useful when compressing huge human motion capture database.

We also performed another comparison experiment on the single type of human motion data (e.g. playing violin). Since most single-type motions are quite short in the CMU database, we duplicate the short motion clips and append them to the original one to create a long motion until it reaches the same size with the previous mixed type motion, 60 MB in this scenario. As shown in single-type

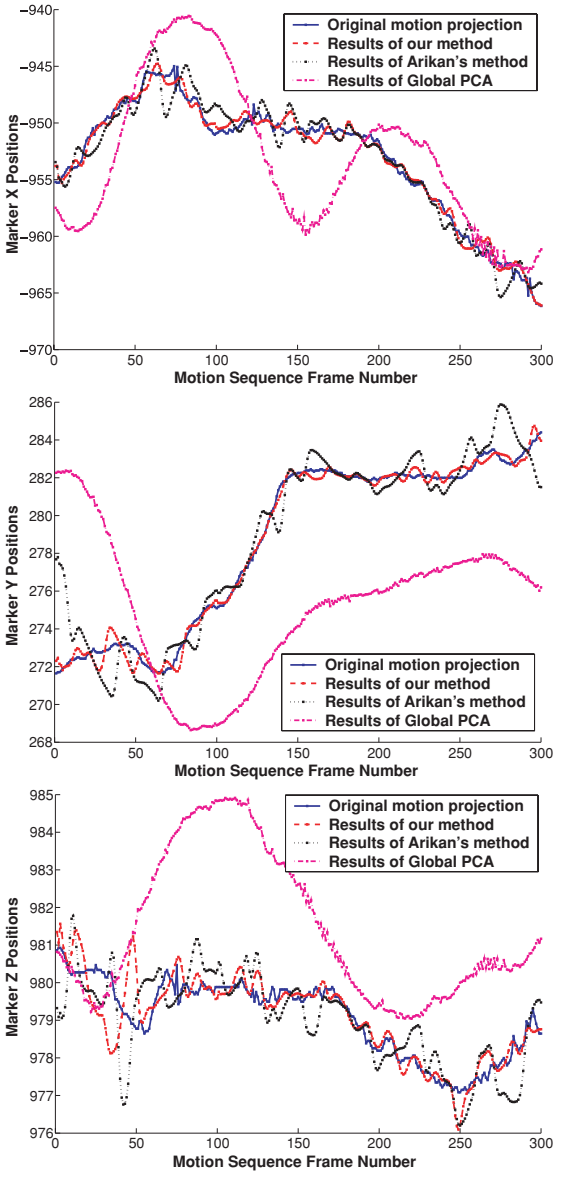

Figure 9: *Reconstructed motion trajectory comparisons between our approach and other techniques (the motion sequence is object #30 from the CMU motion capture database [cmu07]). The global PCA retains 95% of the motion variation. Key parameters used for the Arikan's method [Ari06] in this comparison are: the number of clip frames (K) = 24, reconstruction error bound =0.1, and the number of clusters for CPCA = 10. The key parameters used for our approach in this comparison are: key-frame selection threshold =0.1, initial number of clusters = 10, clustering distortion threshold = 5, and default (even) weight distribution. By using these default parameters, our result appears most similar to the original curve (ARMS error 5.32), slightly better than Arikan's result (ARMS error 5.87), and much better than traditional Global PCA (ARMS error 38.54).*

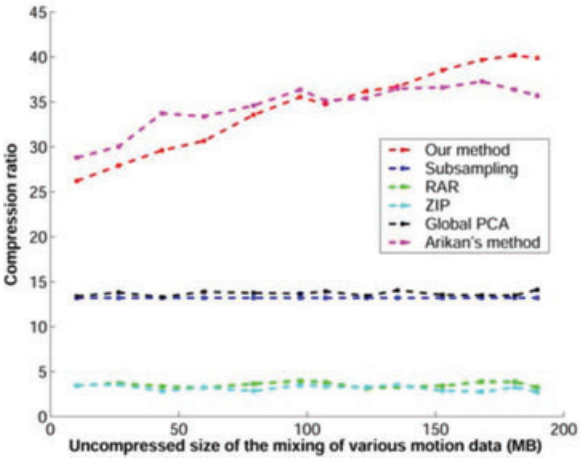

Figure 10: *Compression ratio comparisons between our approach and other compression approaches on the mixing of various motion data (one of them is the object 12 of the CMU motion capture database [cmu07] – 'tai chi'). The sub-sampling factor used in the sub-sampling method is 12, and the parameters of other methods are the same with in Figure 9.*

Figure 11 (comparing with Figure 10), the compression ratio achieved by our approach climbs up more quickly when the size of motion database is increased, due to the fact that there is a larger probability to encounter the occurrences of motion patterns and gain higher compression ratios for the same type of human motion data set. In real-world applications, the motion database may contain various sequences including mixed type motions and single type ones. However, our method deals with each motion sequence separately before clustering (segmentation, normalization, etc.). In the clustering step, complicated motions will contribute more motion patterns (cluster centres) in the clustering step whereas single type motions such as walking and jumping will contribute less patterns. Therefore, the final combined compression ratio will be between the results of Figures 9 and 10, depending on the distribution of motion types in the database.

We tested many motion capture sequences taken from the CMU motion data library [cmu07]. The following Figure 12 compares original motion capture frames and reconstructed motion capture frames side-by-side. As we can see in these two example sequences, the quality of the reconstructed motion capture frames are very close to that of the original motion capture frames.

7. Discussion and Conclusions

In this paper, we propose a human motion compression scheme based on hierarchical structure construction and motion pattern indexing. The key assumption of this scheme is that there is a high degree of temporal and spatial similarity in human motion based on hierarchy body structure, and

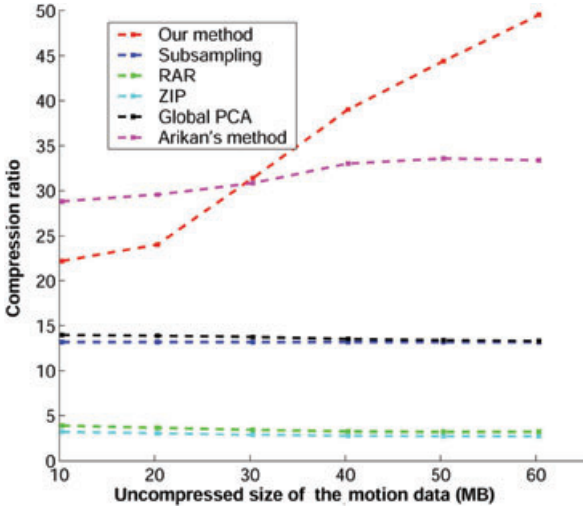

Figure 11: *Compression ratio comparisons between our approach and other compression approaches on the same type of motion data (object 79 of the CMU motion capture database [cmu07] – 'playing violin'). The parameters used for this comparison is the same as those in Figure 10. Note that a distinctive feature of our approach is that when dealing with a single pattern motion, it achieves higher compression ratio when the size of original motion database is increased.*

those motion patterns can be identified to form a database of motion patterns. As such, a motion capture sequence can be converted as a motion pattern index sequence. This work achieves higher compression ratio as the size of the original motion database becomes larger, because comparing with previous work [Ari06], it is more efficient at exploiting redundant motion patterns in a large amount of motion data set. This scheme also provides means for flexible global distortion controls and fine local distortion controls via parameter control.

The success of this scheme is partly due to the reason that the inherent structure information in motion capture data is exploited for the purpose of efficient compression. Compared with previous methods, our hierarchy construction is proved to be effective in finding local similarity which is in turn utilized to reduce the data redundancy. The adaptive segmentation followed by normalization can efficiently extract smooth motion clips which become better clustering candidates than simply segmenting the whole motion sequence through uniform clip length, as done in [Ari06]. As an inevitable disadvantage suffered by all the compression methods that use marker coordinates as metric, this approach will, more or less, modify the lengths of bones during compression and reconstruction. However, in our experiments, we found that more accurate clustering (e.g. increasing the number of clusters) can greatly reduce this negative effect to provide visually acceptable reconstruction results. Furthermore, if strict/absolute accuracy is required, global IK reconstruction can be applied to solve the problem at the expense of longer processing time. In our experiment, the time of compressing single 5 MB motion sequence is about one second. The increase rate of time consumption, however, is above the linear increase of the input size due to the nonlinearity of K-mean clustering, e.g. 30 min for 500 MB input motion.

In the future, we plan to extend this work in the following aspects. First, the proposed manual hierarchy construction may be prone to error amplification from root markers to leaf markers, such as hands and feet. One simple solution is giving smaller threshold to root part and larger threshold to the leaf parts. However, this solution could be very time-consuming and inflexible. Therefore, we plan to investigate automatic algorithms to adaptively build optimal hierarchical structure from motion capture data. The proposed key-frame selection procedure may also be improved by adopting other sophisticated pre-processing methods, such as [BSP*04, BvdPP07, FMJ02]. Plus, a multi-step clustering including inter-object and intra-object clustering is in progress

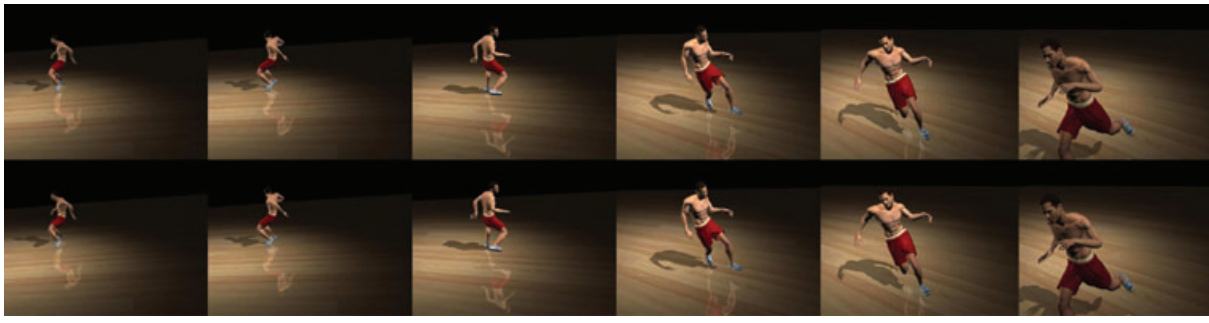

Figure 12: *Side by side comparison between original motion frames (top row) and reconstructed motion frames by our approach (bottom row). The test sequence is a motion capture sequence of playing basketball, from the CMU motion data library object 78 [cmu07].*

to reduce the prohibitive clustering time consumption of extremely large input motion data. Furthermore, we are aware that only human body motion capture data were extensively experimented on this scheme so far, we plan to extend our scheme to process as well other types of motion capture data, such as facial motion capture data and skin deformation capture data. In addition, the current work cannot perform online compression – compression of motion capture data on the fly, frame by frame. As future work, we plan to look into effective techniques for online compression of motion capture data.

Acknowledgements

We would like to thank Qing Li and other members at the UH Computer Graphics and Interactive Media Lab for their constructive suggestions and help. This research work is supported by the new faculty startup fund provided by the University of Houston.

References

[AD01] ALLIEZ P., DESBRUN M.: Progressive encoding for lossless transmission of triangle meshes. In *ACM SIGGRAPH* (2001), pp. 198–205.

[AFSR03] ATTENE M., FALCIDIENO B., SPAGNUOLO M., ROSSIGNAC J.: Swingwrapper: Retiling triangle meshes for better edgebreaker compression. *ACM Transactions on Graphics*, *22*, 4 (2003), 982–996.

[AM00] ALEXA M., MÜLLER W.: Representing animations by principal components. *Computer Graphics Forum*, *19*, 3 (2000), 411–418.

[Ari06] ARIKAN O.: Compression of motion capture databases. *ACM Transactions on Graphics*, *25*, 3 (2006), 890–897.

[ari07] Rpi adaptive arithmetic coding. http://www.cipr.rpi.edu/wheeler/ac/ (2007).

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

[BSM*03] BRICEÑO H. M., SANDER P. V., MCMILLAN L., GORTLER S., HOPPE H.: Geometry videos: A new representation for 3D animations. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 136–146.

[BSP*04] BARBIC J., SAFONOVA A., PAN J.-Y., FALOUTSOS C., HODGINS J. K., POLLARD N. S.: Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface (GI 2004)* (2004), pp. 185–194.

[BvdPP07] BEAUDOIN P., VAN DE PANNE M., POULIN P.: Automatic construction of compact motion graphs. In *Technical Report* (May 2007), p. 1296.

[BW95] BRUDERLIN A., WILLIAMS L.: Motion signal processing. In *Proceedings of SIGGRAPH'95* (1995), pp. 97–104.

[CBL07] CHATTOPADHYAY S., BHANDARKAR S. M., LI K.: Human motion capture data compression by model-based indexing: A power aware approach. *IEEE Transaction on Visualization and Computer Graphics*, *13*, 1 (2007), 5–14.

[CCW*04] CHIU C. Y., CHAO S. P., WU M. Y., YANG S. N., LIN H. C.: Content-based retrieval for human motion data. *Journal of Visual Communication and Image Representation*, *15*, 3 (2004), 446–466.

[cmu07] Cmu motion capture library. http://mocap.cs.cmu.edu (2007).

[FF05] FORBES K., FIUME E.: An efficient search algorithm for motion data using weighted pca. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2005), ACM, pp. 67–76.

[FMJ02] FOD A., MATARIĆ M. J., JENKINS O. C.: Automated derivation of primitives for movement classification. *Auton. Robots*, *12*, 1 (2002), 39–54.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *ACM SIGGRAPH* (2002), pp. 355–361.

[GK04] GUSKOV I., KHODAKOVSKY A.: Wavelet compression of parametrically coherent mesh sequences. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2004), pp. 183–192.

[HPKG06] HUANG Y., PENG J., KUO C.-C. J., GOPI M.: Octree-based progressive geometry coding of point clouds. In *Eurographics Symposium on Point-Based Graphics* (2006), pp. 103–110.

[HRF01] HASTIE T., RIBSHIRANI R., FRIEDMAN J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York, NY, 2001.

[IR03] IBARRIA L., ROSSIGNAC J.: Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 126–135.

[KG02] KHODAKOVSKY A., GUSKOV I.: Normal mesh compression. In *Geometric Modeling for Scientific Visualization*. G. Brunnet, B. Hamann, H. Müller and L. Linsen

(Eds.). Springer-Verlag, Heidelberg, Germany (2002), pp. 189–205.

[KG04] KARNI Z., GOTSMAN C.: Compression of soft-body animation sequences. *Computer & Graphics, Special Issue on Compression*, *28*, 1 (2004), 25–34.

[KSS00] KHODAKOVSKY A., SCHRÖDER P., SWELDENS W.: Progressive geometry compression. In *ACM SIGGRAPH* (2000), pp. 271–278.

[KSW05] KRÜGER J., SCHNEIDER J., WESTERMANN R.: Duodecim – a structure for point scan compression and rendering. In *Eurographics Symposium on Point-Based Graphics* (2005), pp. 99–107.

[KV05] KALAIAH A., VARSHNEY A.: Statistical geometry representation for efficient transmission and rendering. *ACM Transactions on Graphics*, *24*, 2 (2005), 348–373.

[Len99] LENGYEL J.: Compression of time dependent geometry. In *ACM 1999 Symposium on Interactive 3D Graphics* (1999), pp. 89–95.

[LM06] LIU G., MCMILLAN L.: Segment-based human motion compression. In *Proceedings of Symposium on Computer Animation'06* (2006), pp. 127–135.

[Mac02] MACKAY D. J.: *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, UK (2002).

[MR06] MÜLLER M., RÖDER T.: Motion templates for automatic classification and retrieval of motion capture data. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), Eurographics Association, pp. 137–146.

[PK05] PENG J., KUO C.-C. J.: Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. In *ACM SIGGRAPH* (2005), pp. 609–616.

[PKK05] PENG J., KIM C.-S., KUO C.-C. J.: Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation*, *16*, 6 (2005), 688–733.

[SK06] SCHNABEL R., KLEIN R.: Octree-based point cloud compression. In *Eurographics Symposium on Point-Based Graphics* (2006), pp. 111–120.

[SSK05] SATTLER M., SARLETTE R., KLEIN R.: Simple and efficient compression of animation sequences. In *SCA '05: Proceedings of the 2004 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (2005), pp. 209–217.

[YKL02] YANG J.-H., KIM C.-S., LEE S.-U.: Compression of 3D triangle mesh sequences based on vertex-wise motion vector prediction. *IEEE Transactions Circuits and Systems for Video Technology*, *12*, 12 (2002), 1178–1184.

[YKL04] YANG J.-H., KIM C.-S., LEE S.-U.: Progressive compression of 3D dynamic sequences. In *Proceedings of IEEE International Conference on Image Processing* (Oct. 2004), pp. 1975–1978.

[YKL06] YANG J.-H., KIM C.-S., LEE S.-U.: Semi-regular representation and progressive compression of 3D dynamic mesh sequences. *IEEE Transactions on Image Processing*, *15*, 9 (2006), 2531–2544.

[ZO04] ZHANG J., OWEN C. B.: Octree-based animated geometry compression. In *Proceedings of IEEE Data Compression Conference* (2004), pp. 508–517.

[ZO05] ZHANG J., OWEN C. B.: Hybrid coding for animated polygonal meshes: Combining delta and octree. In *Proceedings of IEEE International Conference on Information Technology* (2005), pp. 68–73.