Online MoCap Data Coding with Bit Allocation, Rate Control, and Motion-Adaptive Post-Processing

Choong-Hoon Kwak and Ivan V. Bajić, *Senior Member, IEEE*

***Abstract*—With the advancements in methods for capturing 3D object motion, motion capture (MoCap) data are starting to be used beyond their traditional realm of animation and gaming in areas such as the arts, rehabilitation, automotive industry, remote interactions, and so on. As the amount of MoCap data increases, compression becomes crucial for further expansion and adoption of these technologies. In this paper, we extend our previous work on low-delay MoCap data compression by introducing two improvements. The first improvement is the bit allocation to long-term and short-term reference MoCap frames, which provides a 10-15% reduction in coded bitrate at the same quality. The second improvement is the post-processing in the form of motion-adaptive temporal low-pass filtering, which is able to provide another 9-13% savings in the bitrate. The experimental results also indicate that the proposed online MoCap codec is competitive with several state-of-the-art offline codecs. Overall, the proposed techniques integrate into a highly effective online MoCap codec that is suitable for low-delay applications, whose implementation is provided alongside this paper to aid further research in the field.**

***Index Terms*—Motion capture, low-delay compression, rate control, bit allocation, motion-adaptive filtering**

I. Introduction

THE use of motion capture (MoCap) technology has diversified into fields such as entertainment, performing arts, automotive industry, security, and healthcare. Some of the widely known applications of MoCap technology include animating virtual characters in movies such as *Avatar*, *King Kong*, and many others. In the gaming industry, MoCap is used to enable realistic motion of player-controlled virtual characters in various sports and action games. More recently, MoCap has found use in clinical applications, such as in the muscle fatigue monitoring tool in [1], which is used to help reduce sports injuries caused by incorrect muscle usage and fatigue. Another clinical application is [2], whose focus is gait analysis. In new media art, systems such as [3], [4], [5] utilize MoCap technology to encourage the interaction of virtual characters with actors and (remote) audience in real time. Another arts-related project [6] focuses on mapping, controlling or expressing the motion data in the form of sound or music.

Early MoCap applications relied on the offline processing of stored MoCap data, as is the case in the animation of virtual characters in the film industry. Since one of the requirements of such applications is storage of MoCap data, the corresponding storage-targeted MoCap compression schemes were developed. In [7], Arikan studied the compression of an entire MoCap database rather than individual MoCap clips. In [8], [9], the authors used the well-known principal component analysis (PCA) technique to compress MoCap data. PCA is applied temporally to either the entire MoCap sequence or its segments. In [10], Tournier proposed a compression method based on principal geodesic analysis (PGA), an extension of PCA, combined with the inverse kinematic tool. In the animation industry, the key frame selection technique is widely used for MoCap data compression. The encoder stores the important (key) frames, and the decoder reconstructs in-between frames using an interpolation tool such as splines. In [11], Lin *et al.* analyzed repetitive motion, grouped segments of similar motion and then applied PCA for the dimensionality reduction. Furthermore, they used a spline-based approximation of the coefficients and applied adaptive quantization with entropy coding for compression. In [12], Váša and Brunnett used PCA and Lagrangian optimization to compress BVH MoCap data. In the BVH MoCap format, where the relative positions of joints are described hierarchically with the help of Euler angles, the parent joint has more influence on the overall distortion than the children joints due to error propagation. Within the Lagrangian framework, the influence of local distortion of each joint was analyzed in [12] and compression was adjusted to minimize the overall distortion.

In [13], Chattopadhyay *et al.* focused on reducing the power consumption of compression to encourage potential use on mobile devices. Transform coding based on wavelet transform or discrete cosine transform (DCT) has also been used in MoCap data compression. In [14], [15], [16], [17], the encoder applies the wavelet transform temporally to articulated human skeletal joints. In [18] and [19], the MPEG-4 bone-based animation (BBA) was proposed using delta prediction or temporal differencing followed by temporal DCT transform applied to 16 consecutive frames. More recently, Hou *et al.* [20] proposed a MoCap data compaction technique using tensor decomposition across spatial and temporal dimensions.

As shown above, the majority of methods for MoCap compression involve temporal transformation, which requires buffering MoCap data and delaying compression. In some cases, the entire MoCap sequence must be stored and processed before compression can start. This approach is incompatible with the recent trends in MoCap usage, which emphasize real-time, interactive applications. For this reason, our focus is online MoCap compression, in which the data can be encoded immediately after being captured. Our prior work [21], [22] introduced a hybrid architecture for MoCap compression, incorporating temporal prediction and spatial

The authors are with the School of Engineering Science, Simon Fraser University, Burnaby, BC, V5A 1S6, Canada. Tel: 1-778-782-7159. Fax: 1-778-782-4951. E-mails: cka21@sfu.ca, ibajic@ensc.sfu.ca.

transformation. From experience with video coding, it is known that such an architecture is capable of delivering high compression simultaneously with a low delay. In addition, error resilience can be built into the system [23], making it even more suitable for interactive applications across a network.

In this work, we improve the compression performance of the codec in [22] in two ways. First, we develop a bit allocation strategy that allows the adaptive allocation of bits across different types of MoCap frames to improve the rate-distortion performance. Second, we develop a decoder-side post-processing strategy to improve the quality of reconstructed MoCap data through motion-adaptive filtering. Together, these strategies reduce the bitrate by 22-26% at the same reconstruction quality compared to the current state-of-the-art in online MoCap coding [22]. The resulting codec is also competitive with well-known offline BVH codecs [10], [11], [12].

The remainder of this paper is organized as follows. We begin with a brief review of the coding architecture from [21], [22] and the relevant quantization, entropy coding and rate control mechanisms in Section II. Next, the proposed bit allocation method is presented in Section III, followed by a description of the post-processing of decoded MoCap data in Section IV. The experimental results are presented in Section V, followed by conclusions in Section VI. A MATLAB implementation of the proposed online MoCap codec is available at http://geomm.ensc.sfu.ca/papers/MoCap-coding/.

II. Preliminaries

A. MoCap data formats

There are several popular data formats for MoCap data. Since the present paper primarily addresses the C3D and BVH formats, we will only focus on these two formats. C3D (coordinate 3D) provides 3D coordinates of motion markers relative to a chosen origin. A typical C3D file contains three sections: header, parameter, and data sections. The header contains the pointer to the start of the parameter and data sections. The parameter section contains the labels of motion markers and other information relevant to interpreting the data. Finally, the data section contains 3D coordinates of motion markers, which are stored frame-by-frame. The C3D data format makes no assumptions about the body or object that the motion was collected from; hence, it is very general.

In contrast to C3D, the BVH file format is tailored to skeletal data, specifically for human or animal motion. BVH is popular in the animation community because it allows easy computation of skeletal motion. Skeletal constraints are embedded in the data representation by referring the positions of all joints hierarchically to their parent joints. A BVH file contains two sections: header and channel. The header contains the hierarchical information with the initial skeletal pose. The channel section contains the motion information of the skeleton in terms of Euler angles, allowing one to represent such motion with fewer degrees of freedom compared to plain 3D position coordinates.

The efficiency of BVH for describing skeletal motion is also its limitation in terms of its overall ability to represent motion.

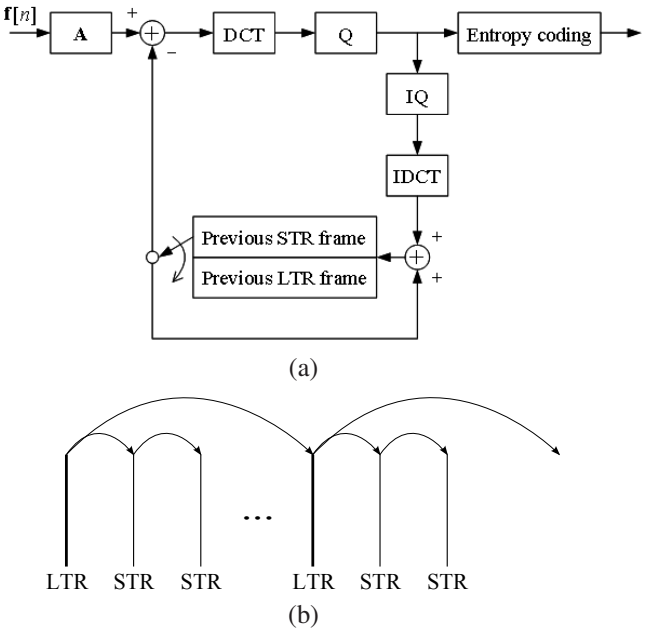


Fig. 1: (a) Block diagram of the MoCap encoder from [21], [22]. (b) Prediction structure involving LTR and STR frames.

BVH cannot represent soft tissue motion or, for example, a flag waving in the wind. C3D does not possess such limitations. Our original MoCap codec [21], [22] was developed for the general C3D format, but in this work, we also provide a version that is adapted to BVH to enable comparison with BVH MoCap codecs.

B. Hybrid MoCap coding

Fig. 1(a) shows a block diagram of the hybrid low-delay MoCap encoder from [21], [22]. A C3D MoCap data frame $\mathbf{f}[n]$, consisting of 3D coordinates of motion markers at time n, is reordered by matrix $\mathbf{A}$ such that all x-coordinates are stacked columnwise, followed by y- and z-coordinates, which concentrate the signal energy at low frequencies. Specifically, for $\mathbf{f}[n] = (x_1[n], y_1[n], z_1[n], ..., x_m[n], y_m[n], z_m[n])^T$, the reordered frame is $\mathbf{g}[n] = \mathbf{A}\mathbf{f}[n]$, where

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix},$$

and $\mathbf{g}[n] = (x_1[n], ..., x_m[n], y_1[n], ..., y_m[n], z_1[n], ..., z_m[n])^T$. It is shown in [21] that such reordering leads to the concentration of energy in $\mathbf{g}[n]$ at low frequencies. Several recent works have explored alternative representations and transformations of MoCap data [24], [25], [26].

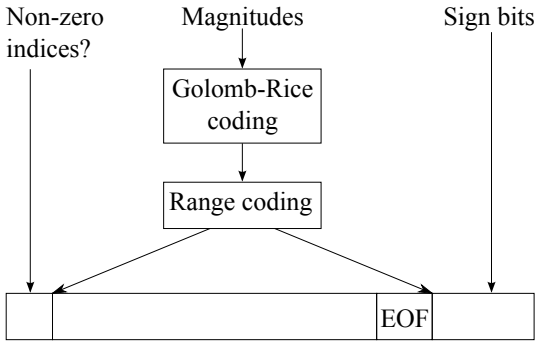


Fig. 2: Structure of the encoded bitstream for one frame of MoCap data.

The reordered data frame is then predicted either from a long-term reference (LTR) frame or from a short-term reference (STR) frame, and the prediction residual is spatially transformed, quantized, and entropy coded.

Prediction involving LTR and STR frames is illustrated in Fig. 1(b). A group of frames (GOF) is defined as a set of frames starting with an LTR frame and including all subsequent STR frames until the next LTR frame. An STR frame is always predicted from the previous frame (either LTR or STR), similar to the P frame in video coding. The LTR frame is always predicted from the previous LTR frame, except for the first frame in the sequence, which is intra-coded. This type of prediction structure enables a certain level of error resilience in MoCap transmission because errors in STR frames cannot propagate beyond the next LTR frame. This fact can be used to help error concealment [23].

A 1D DCT is applied to the prediction residual of each frame. Markers with similar motion will end up with similar prediction residuals, which, after transformation, will result in the concentration of energy at low frequencies. This is subsequently exploited by quantization (after which many high-frequency coefficients are zero) and entropy coding (which exploits non-uniform probabilities of quantized DCT magnitudes) to achieve high compression. In particular, a uniform midtread quantizer is applied to each DCT coefficient x to obtain the quantization index

$$q = \mathrm{sign}(x)\,\lfloor |x|/\Delta + 0.5 \rfloor\,, \tag{1}$$

where $\lfloor\cdot\rfloor$ denotes the round-down operation and the step size Δ is adaptively adjusted as described in Section II-C. The quantized coefficient magnitudes are entropy coded, first by adaptive Golomb-Rice coding [27] followed by adaptive arithmetic coding [28]. The sign-bits of non-zero quantized transform coefficients are stored uncoded at the end of the frame's bitstream, as illustrated in Fig. 2. Readers are referred to [21], [22] for further details.

C. Rate control

When transmitting encoded MoCap data in a real-time interactive application, the available bitrate may vary with time; therefore, it is important to be able to control the instantaneous output bitrate of the encoder. In [22], a rate control method for MoCap data encoding was proposed, in which the quantizer step size Δ in (1) is varied depending on the data statistics and the target bitrate. Statistical tests in [22] revealed that the prediction residuals of LTR frames can be modeled as Laplacian random variables, whereas the residuals of STR frames are better modeled as Gaussian random variables. The relationship between the rate R and a uniform midtread quantizer step size Δ for a Laplacian random variable in the low-rate regime is given by [29]:

Symbol	Description
r_t	target bitrate (bits per second)
F	frame rate (frames per second)
N_c	number of DCT coefficients per frame
N_{GOF}	number of frames in a GOF
R^t_{GOF}	target number of bits per GOF
R^t_{LTR}	target bits per coefficient in LTR
R^t_{STR}	target bits per coefficient in STR

TABLE I: Notation used in rate control

$$R(\Delta) = \mathcal{H}\left(\sqrt{\theta}\right) + \sqrt{\theta}\left(1 - \log_2\left(1-\theta\right)\right) - \sqrt[3]{\theta}\frac{\log_2\theta}{1-\theta}, \tag{2}$$

where $\theta = e^{-\lambda\Delta}$, $\lambda = \sqrt{2}/\sigma_r$, and $\mathcal{H}(\cdot)$ is the binary entropy function. Moreover, the corresponding relationship for a Gaussian random variable is approximated by [30]:

$$\begin{aligned} R(\Delta) &\approx -\sum_{k=-1}^{1} p_k \log_2 p_k, \\ p_k &= \frac{1}{2}\left(\mathrm{erf}\left(\frac{a\,(2k+1)}{2}\right) - \mathrm{erf}\left(\frac{a\,(2k-1)}{2}\right)\right), \end{aligned} \tag{3}$$

where $a = \Delta/(\sqrt{2}\sigma_r)$ and $\mathrm{erf}(\cdot)$ is the error function. The symbol σ_r in (2) and (3) represents the standard deviation of the decoded coefficients of the reference frame, which is thresholded from below by a small number to avoid numerical instability in the case of very slow motion when the true standard deviation approaches zero [22].

Algorithm 1 summarizes the rate control scheme from [22] using the symbols explained in Table I. The target number of bits per GOF is R^t_{GOF}, the target number of bits per coefficient in the LTR frame is R^t_{LTR}, and the target number of bits per coefficient in the STR frames is R^t_{STR}. If N_c is the number of transform coefficients per frame, then R^t_{GOF} can be expressed as

$$R^t_{\mathrm{GOF}} = N_c R^t_{\mathrm{LTR}} + N_c(N_{\mathrm{GOF}} - 1)R^t_{\mathrm{STR}}. \tag{4}$$

Let $\alpha = R^t_{\mathrm{LTR}}/R^t_{\mathrm{STR}}$ be the ratio of target bits per coefficient in LTR vs. STR frames. Then, from (4), we obtain

$$\begin{aligned} R^t_{\mathrm{STR}} &= \frac{R^t_{\mathrm{GOF}}}{N_c(\alpha + N_{\mathrm{GOF}} - 1)}, \\ R^t_{\mathrm{LTR}} &= \alpha R^t_{\mathrm{STR}}, \end{aligned} \tag{5}$$

which is used in Step 2 of Algorithm 1 to compute R^t_{LTR}. Further details and discussion can be found in [22].

Algorithm 1 Encoder rate control for each GOF

Input: R^t_{GOF} ▷ target bits for current GOF
Output: $R^t_{\mathrm{GOF,next}}$ ▷ target bits for next GOF
$R^a_{\mathrm{GOF}} \leftarrow 0$ ▷ actual bits in current GOF
Compute R^t_{LTR} from (5)
Solve $R(\Delta) = R^t_{\mathrm{LTR}}$ for Δ using (2)
Encode LTR frame; R^a_{LTR} is the number of bits spent
$R^a_{\mathrm{GOF}} \leftarrow R^a_{\mathrm{GOF}} + R^a_{\mathrm{LTR}}$
for $n = 2$ to N_{GOF} **do** ▷ for each STR frame
  $R^t_{\mathrm{STR}} \leftarrow \frac{R^t_{\mathrm{GOF}} - R^a_{\mathrm{GOF}}}{N_c(N_{\mathrm{GOF}} - n + 1)}$
  Solve $R(\Delta) = R^t_{\mathrm{STR}}$ for Δ using (3)
  Encode STR frame; R^a_{STR} is the number of bits spent
  $R^a_{\mathrm{GOF}} \leftarrow R^a_{\mathrm{GOF}} + R^a_{\mathrm{STR}}$
end for
return $R^t_{\mathrm{GOF,next}} = r_t N_{\mathrm{GOF}}/F + R^t_{\mathrm{GOF}} - R^a_{\mathrm{GOF}}$

III. Bit Allocation

In [22], the target ratio between the bits allocated to LTR and STR frames was fixed at $R^t_{\mathrm{LTR}}/R^t_{\mathrm{STR}} = \alpha = 10$. This ratio, however, is not appropriate for all target bitrates. At low bitrates, when the available bits are scarce, it is advantageous to invest more of them into LTR frames (i.e., large α), which directly or indirectly influence all subsequent frames in the sequence. As the bitrate increases and more bits are available to distribute, the STR frames' share should increase (i.e., lower α). In this section, we analyze the problem of bit allocation to LTR and STR frames and develop an allocation strategy that improves the rate-distortion performance across different bitrates.

Our goal is to minimize the distortion in a GOF consisting of one LTR frame and $(N_{\mathrm{GOF}} - 1)$ STR frames subject to the total rate constraint

$$N_c R^t_{\mathrm{LTR}} + N_c(N_{\mathrm{GOF}} - 1)R^t_{\mathrm{STR}} \le R^t_{\mathrm{GOF}}. \tag{6}$$

Based on the results in [31] for bit allocation to non-identical random variables in the high-rate regime, an approximate solution has the form

$$R^t_{\mathrm{LTR}} = \overline{R^t} + \frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{LTR}}}{\rho^2} + \frac{1}{2}\log_2\frac{h_{\mathrm{LTR}}}{H} \tag{7}$$

$$R^t_{\mathrm{STR}} = \overline{R^t} + \frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{STR}}}{\rho^2} + \frac{1}{2}\log_2\frac{h_{\mathrm{STR}}}{H} \tag{8}$$

where $\overline{R^t} = (R^t_{\mathrm{LTR}} + (N_{\mathrm{GOF}} - 1)R^t_{\mathrm{STR}})/N_{\mathrm{GOF}}$ is the average number of target bits per coefficient in a GOF, ρ^2 is the geometric mean of the variances of coefficients in LTR and STR frames, and H is given by

$$H = \left(h_{\mathrm{LTR}} \cdot (h_{\mathrm{STR}})^{(N_{\mathrm{GOF}}-1)}\right)^{1/N_{\mathrm{GOF}}}. \tag{9}$$

Constants $h_i, i \in \{\mathrm{LTR}, \mathrm{STR}\}$ are related to the probability density function (pdf) of the corresponding random variable:

$$h_i = \frac{1}{12}\left\{\int_{-\infty}^{\infty}\left[f_i(x)\right]^{1/3}\right\}^3, \tag{10}$$

where f_i is Laplacian pdf for $i = \mathrm{LTR}$ and Gaussian pdf for $i = \mathrm{STR}$. Evaluating (10) for zero-mean unit-variance Laplacian and Gaussian densities provides

$$h_{\mathrm{LTR}} = \frac{9}{2}, \qquad h_{\mathrm{STR}} = \frac{\sqrt{3}\pi}{2}. \tag{11}$$

Eliminating $\overline{R^t}$ from (7) and (8), we obtain

$$\begin{aligned} R^t_{\mathrm{LTR}} &= R^t_{\mathrm{STR}} + \frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + \frac{1}{2}\log_2\frac{h_{\mathrm{LTR}}}{h_{\mathrm{STR}}} \\ &\approx R^t_{\mathrm{STR}} + \frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + 0.363, \end{aligned} \tag{12}$$

where the second line follows from (11). Hence, if the variances of the LTR and STR frames were equal, each coefficient in the LTR frame should receive, on average, 0.363 bits more than a coefficient in the STR frame to account for the difference in their distributions. If the variances are different, the allocation changes according to the logarithm of their ratio.

Unfortunately, we cannot use this result directly because it assumes high-rate quantization and fixed-length coding, whereas we would like to be able to use variable-length coding and low to medium rates. To account for the difference arising from the modeling assumptions, we modify (12) as follows:

$$R^t_{\mathrm{LTR}} = R^t_{\mathrm{STR}} + \frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + g\left(\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}}\right), \tag{13}$$

where $g(\cdot)$ is a function of the ratio of variances of the LTR and STR frames. This function $g(\cdot)$ will serve to model the difference between the high-rate fixed-length coding result (12) and the corresponding result for our target application, which involves low-rate variable-length coding. Specifically, we take $g(\cdot)$ to be a logarithmic function of the variance ratio:

$$g\left(\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}}\right) = a \cdot \log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + b. \tag{14}$$

With such $g(\cdot)$, the relationship between the target bits per coefficient for the LTR and STR frames becomes

$$R^t_{\mathrm{LTR}} = R^t_{\mathrm{STR}} + \frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + \left(a \cdot \log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + b\right). \tag{15}$$

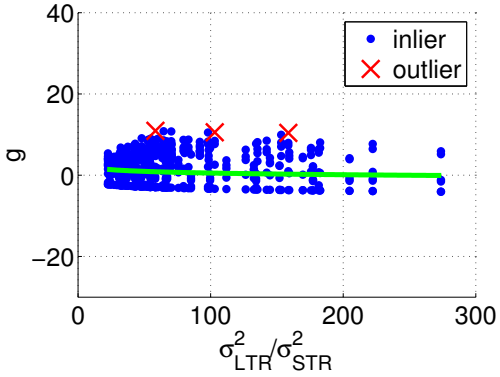
To find the ratio $\alpha = R^t_{\mathrm{LTR}}/R^t_{\mathrm{STR}}$, we substitute R^t_{LTR} into equation (4) and solve for R^t_{STR} to obtain

$$\begin{aligned} R^t_{\mathrm{STR}} \;=\; \frac{1}{N_{\mathrm{GOF}}}\Bigg[&\frac{R^t_{\mathrm{GOF}}}{N_c} - \frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} \\ &- \left(a \cdot \log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + b\right)\Bigg] \end{aligned} \tag{16}$$

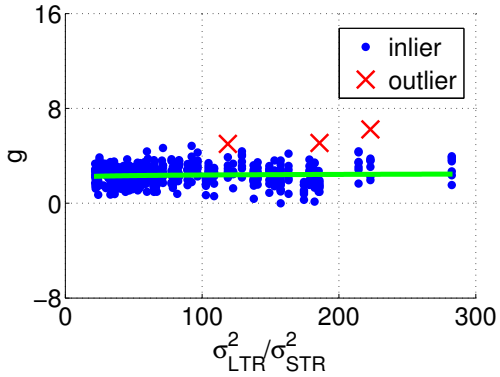
Then, dividing (15) by (16) provides

$$\alpha = 1 + \frac{N_{\mathrm{GOF}}\left[\frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + \left(a \cdot \log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + b\right)\right]}{\frac{R^t_{\mathrm{GOF}}}{N_c} - \frac{1}{2}\log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} - \left(a \cdot \log_2\frac{\sigma^2_{\mathrm{LTR}}}{\sigma^2_{\mathrm{STR}}} + b\right)}. \tag{17}$$

To estimate a and b, we encoded 20 training MoCap sequences (Table II, Section V) at a variety of target bitrates and GOF sizes $N_{\mathrm{GOF}} \in \{20, 30, 40, 50\}$. For each bitrate and GOF size, encoding was performed using fixed bit ratios

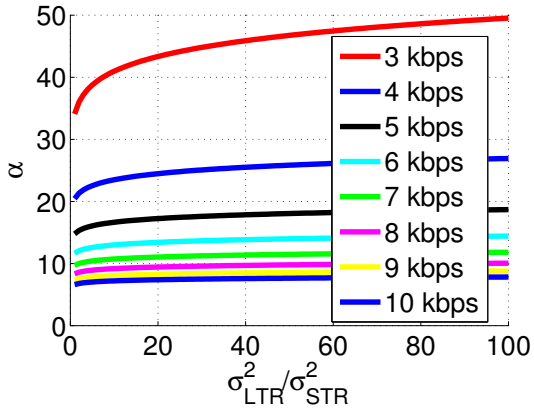


(a) C3D

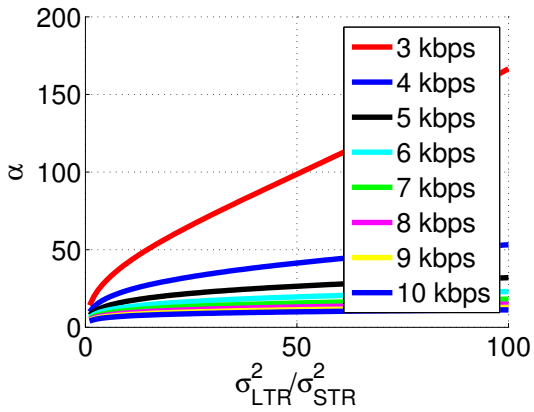


(b) BVH

Fig. 3: g in (14) vs. $\sigma^2_{\mathrm{LTR}}/\sigma^2_{\mathrm{STR}}$.



(a) C3D



(b) BVH

Fig. 4: α from (17) vs. $\sigma^2_{\mathrm{LTR}}/\sigma^2_{\mathrm{STR}}$ for various target bitrates.

of $1 \leq \alpha \leq 140$, and the ratio α that led to the lowest mean squared error (MSE) was selected. Using this α, the corresponding value of g was computed from (13), where R^t_{LTR} and R^t_{STR} were computed from (5). Scatter plots of such values of g are shown against the corresponding variance ratios $\sigma^2_{\mathrm{LTR}}/\sigma^2_{\mathrm{STR}}$ in Fig. 3, where Fig. 3(a) corresponds to C3D MoCap sequences and Fig. 3(b) corresponds to BVH MoCap sequences. The parameters a and b were then estimated to find a least squares fit of (14) to the data, with 1% regression-based outlier removal. For C3D data, the estimated parameter values were $a = -0.4121$ and $b = 3.2745$, whereas for BVH, they were $a = 0.0500$ and $b = 2.0492$. The fitted model is shown as a green line, and the outliers are indicated by a red X in Fig. 3.

Fig. 4 shows the α computed from (17) using the estimated values for the parameters a and b for various target bitrates and LTR to STR residual variance ratios, $\sigma^2_{\mathrm{LTR}}/\sigma^2_{\mathrm{STR}}$. As shown in this figure, for a given target bitrate, α is predicted to increase as the ratio $\sigma^2_{\mathrm{LTR}}/\sigma^2_{\mathrm{STR}}$ increases because the fraction of bits assigned to LTR frames would need to be larger as their variance increases relative to STR frames. Moreover, for a given ratio $\sigma^2_{\mathrm{LTR}}/\sigma^2_{\mathrm{STR}}$, α is predicted to decrease as the target bitrate increases. This result is also logical because as the number of available bits increases, the fraction assigned to STR frames can increase, and thus, α decreases. These predictions made by the model agree well with our intuition.

For each GOF, rate control with adaptive bit allocation is accomplished via Algorithm 2, which is similar to Algorithm 1 except for step 2 involved in computing the bit allocation ratio α between LTR and STR frames. Specifically, step 2 computes the value of α for the current GOF from (17). This value involves the estimates of the variances of LTR and STR frames in the current GOF based on the already encoded LTR and STR frames from the previous GOF.

IV. Post-processing

In the proposed MoCap coding system, quantization is applied to prediction residuals (Fig. 1(a)). At low bitrates, where the quantizer step size is large, quantization causes discontinuities between the marker positions in neighboring frames, resulting in visual judder. This can also be observed in the frequency domain by analyzing the spectra of the original and decoded MoCap sequences. Fig. 5 shows the spectral envelope of marker position signals from test sequence 13_29 (Section V) encoded at various bitrates. Observe that for frequencies above approximately 5 Hz, the encoded sequences have considerably higher energy than the original sequence due to the quantization noise effects mentioned above.

One way to suppress such quantization noise is to apply temporal low-pass filtering to the decoded marker/joint po-

Algorithm 2 Encoder rate control with adaptive bit allocation

Input: R^t_{GOF} ▷ target bits for current GOF
Output: α ▷ α to be used in current GOF
Output: $R^t_{\mathrm{GOF,next}}$ ▷ target bits for next GOF
$R^a_{\mathrm{GOF}} \leftarrow 0$ ▷ actual bits in current GOF
Compute α from (17) using σ^2_{LTR} and σ^2_{STR} from previous GOF
Compute R^t_{LTR} from (5)
Solve $R(\Delta) = R^t_{\mathrm{LTR}}$ for Δ using (2)
Encode LTR frame; R^a_{LTR} is the number of bits spent
$R^a_{\mathrm{GOF}} \leftarrow R^a_{\mathrm{GOF}} + R^a_{\mathrm{LTR}}$
for $n = 2$ to N_{GOF} **do** ▷ for each STR frame
  $R^t_{\mathrm{STR}} \leftarrow \frac{R^t_{\mathrm{GOF}} - R^a_{\mathrm{GOF}}}{N_c(N_{\mathrm{GOF}} - n + 1)}$
  Solve $R(\Delta) = R^t_{\mathrm{STR}}$ for Δ using (3)
  Encode STR frame; R^a_{STR} is the number of bits spent
  $R^a_{\mathrm{GOF}} \leftarrow R^a_{\mathrm{GOF}} + R^a_{\mathrm{STR}}$
end for
return α
return $R^t_{\mathrm{GOF,next}} = r_t N_{\mathrm{GOF}}/F + R^t_{\mathrm{GOF}} - R^a_{\mathrm{GOF}}$

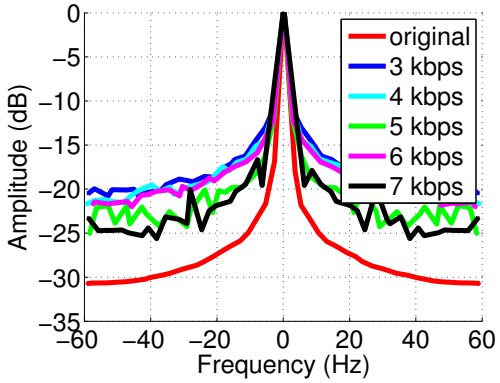
sition signals. However, a one-fits-all filter does not appear to be appropriate because different markers, attached to different body parts, may move in different ways, and signals corresponding to their positions may have different frequency contents. Consider, for example, a person standing in place and waving his arms. His feet are static, and thus, the position signals for the markers on the feet are constant and have only DC energy, allowing a low-pass filter with a very low cutoff frequency. Meanwhile, markers on the arms are moving, and thus, the filter applied to those signals needs to have a higher cutoff frequency to accommodate such motion.

We therefore propose a post-processing module, as illustrated in Fig. 6. The decoded position signal $p[n]$, which represents the x-, y- or z-coordinate of a given marker (or skeletal joint in the case of BVH data) at frame n, is fed to the motion analysis module, whose task is to decide which filter is appropriate for the corresponding signal. The filters in the filterbank are temporal low-pass filters with different cutoff frequencies. Upon filtering, the decoded value $p[n]$ is replaced by the filtered value $p_f[n]$. Note that only the decoded position signal $p[n]$ is needed to select a filter from the filter bank, without any side information. Since filtering is needed mostly to suppress large quantization noise at low to medium bitrates, the filter bank may be switched off at high bitrates to avoid smoothing out the occasional high-frequency feature of the natural motion, such as an impact of an object with a hard surface.

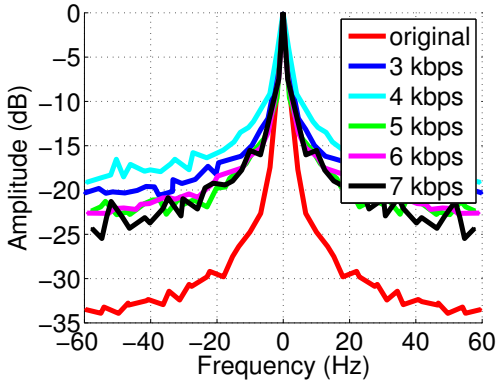
A. Filters

In the simulations presented in Section V, we used $N_f = 3$ finite impulse response (FIR) filters with cutoff frequencies of 1, 3, and 5 Hz in the filterbank. These FIR filters were designed using the window method [32]. Specifically, to obtain the filter's impulse response, we applied the Hamming window

$$w[n] = 0.54 + 0.46\cos\left(\frac{2\pi n}{2K+1}\right), \tag{18}$$



(a) C3D (13_29)



(b) BVH (85_12)

Fig. 5: Normalized energy spectral envelopes of test sequence at various bitrates.

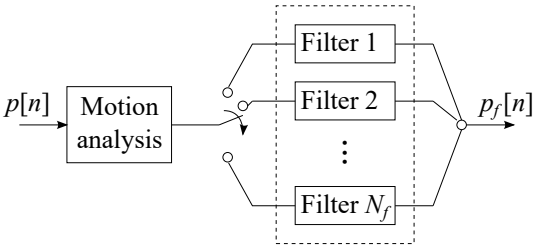


Fig. 6: Motion-adaptive post-processing.

where $|n| \le K$ and $(2K+1)$ is the filter length, to the ideal low-pass filter impulse response

$$h_D[n] = \begin{cases} \frac{\sin(2\pi f_c n)}{\pi n} & \text{if } n \neq 0, \\ 2f_c & \text{if } n = 0, \end{cases} \tag{19}$$

where f_c is the cutoff frequency in Hz. The filter takes $2K+1$ samples of the position signal $p[n]$, namely, $p[n-K], p[n-K+1], ..., p[n+K]$, where n represents the current frame, and applies the impulse response to compute $p_f[n]$. Hence, buffering K future frames is necessary to execute this filtering operation, which introduces a K-frame delay at the decoder when filtering is used. The effect of filter length on the final distortion is further analyzed in Section IV-C.

B. Motion Analysis

To decide which filter to use for the particular position signal, we employ a second-order motion predictor [33] to the past (and already filtered) signal values and check how well it predicts the current signal value. Specifically, we compute the estimates of velocity ($\hat{v}$), acceleration ($\hat{a}$) and position ($\hat{p}$) as follows:

$$\hat{v}[n-1] = p_f[n-1] - p_f[n-2], \tag{20}$$

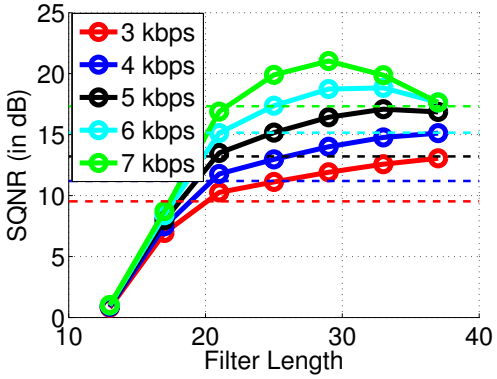
$$\hat{a}[n-1] = \hat{v}[n-1] - \hat{v}[n-2], \tag{21}$$

$$\hat{p}[n] = p[n-1] + \hat{v}[n-1] + \hat{a}[n-1]/2, \tag{22}$$

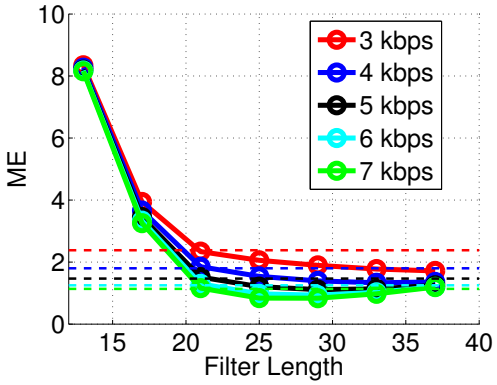
and then compare the predicted position value $\hat{p}[n]$ with the decoded value $p[n]$. Under constant acceleration, the second-order predictor is accurate [33]. If the difference $|\hat{p}[n] - p[n]|$ is sufficiently small, it means that the constant-acceleration assumption employed in the predictor (approximately) holds, which means that the motion is smooth. Therefore, the filter with the lowest cutoff frequency can be applied. As the difference between $\hat{p}[n]$ and $p[n]$ increases, the constant-acceleration assumption becomes less true, which requires the use of filters with higher cutoff frequencies. For filters arranged in ascending order of cutoff frequency, the decision criterion that we used was:

$$\begin{cases} \text{if } |\hat{p}[n] - p[n]| \le T_1 & \text{use Filter 1,} \\ \text{if } T_1 < |\hat{p}[n] - p[n]| \le T_2 & \text{use Filter 2,} \\ \text{if } |\hat{p}[n] - p[n]| > T_2 & \text{use Filter 3.} \end{cases} \tag{23}$$

The empirical values for T_1 and T_2 that were found to work well on the MoCap data in our simulations were $T_1 = 20$ and $T_2 = 120$.



(a) C3D



(b) BVH

Fig. 7: Distortion vs. filter length for various target bitrates for 85_12. Dashed horizontal lines indicate distortion with no filtering for the corresponding target bitrate.

C. Filter Length

The effect of filter length on the final distortion is shown in Fig. 7 for various target bit rates for sequence 85_12. Similar results were obtained for other sequences. The signal-to-quantization noise ratio (SQNR, defined in (24) in Section V-A) is shown vs. filter length for C3D sequences in Fig. 7(a). For BVH data, the mean error (ME, defined in (25) in Section V-A) has commonly been used as a distortion metric [12], so ME is shown vs. filter length in Fig. 7(b) for BVH sequences. At each data point, the length of all three filters in the filterbank is the same. For each target bitrate, the distortion without filtering is shown as a horizontal dashed line of the same color, for reference.

From digital signal processing [34], it is known that in window-based filter design, the longer the filter is, the better the passband characteristics and the sharper the roll-off in the stopband. Short filters do not provide much attenuation to high-frequency quantization noise, whereas their poor passband characteristics distort the main low-frequency signal; thus, short filters have worse performance (lower SQNR, higher ME) than no filtering. As the filter length increases, the performance improves and becomes better than that of no filtering for lengths above approximately 20, depending on the bit rate. As the filter length further increases, its sharper roll-off causes ever more attenuation at high frequencies, and the performance eventually starts to degrade because the filter overly suppresses not only the quantization noise but also the high-frequency components of the motion signal itself. This is visible in Fig. 7 at higher bit rates, where the amount of quantization noise is comparatively lower. At lower bit rates, the effect occurs for larger filter lengths, beyond those shown in Fig. 7. This phenomenon can be explained by Wiener filter theory [35], which states that the optimal filter gain at a given frequency is governed by the signal-to-noise ratio at that frequency. Hence, when the sharper roll-off causes the gain to drop below the optimal value, the filtering performance will degrade. This also suggests that side information in the form of a motion model (for example, a bio-mechanical model of human motion for human MoCap coding) may be helpful in improving the filtering performance by allowing the decoder to estimate the energy of a (noise-free) uncompressed signal within a (noisy) compressed signal and thereby select the optimal filter gain at each frequency. This would be an interesting topic for future research.

Based on the results in Fig. 7 and similar experiments with other MoCap sequences, we have selected a filter length of 25 (corresponding to $K = 12$) for our experiments. A larger K leads to better performance (up to a point), but the

downside is increased buffering in the post-processing module, which increases end-to-end latency. The appropriate level of latency will depend on the application. According to [36], for example, the most demanding classes of games (e.g., first-person shooters) should limit the latency to 100 msec to maintain a reasonable level of player performance. For the 120 Hz MoCap data used in our simulations, the chosen value of $K = 12$ makes the buffering delay exactly 100 msec.

A more recent study [37] found that 45 msec is a better estimate of the latency demands of the most fast-paced games. If this, or even lower, latency is required, the filtering can simply be turned off at the decoder. In such a case, the bitrate may need to be increased somewhat to reach the required level of accuracy.

V. Experimental Results and Discussion

A. Experimental setup

To evaluate the compression effectiveness for C3D data, we use the well-known signal-to-quantization-noise ratio (SQNR) [31]. Let $\mathbf{f}[n]$ be the n-th frame containing the original 3D marker coordinates and N be the total number of frames in the sequence. Then, the SQNR in dB is computed as:

$$\text{SQNR} = 10\log_{10}\frac{\frac{1}{N}\sum_{n=1}^{N}(\mathbf{f}[n])^T(\mathbf{f}[n]))}{\frac{1}{N}\sum_{n=1}^{N}(\mathbf{f}[n]-\widehat{\mathbf{f}}[n])^T(\mathbf{f}[n]-\widehat{\mathbf{f}}[n])}, \tag{24}$$

where $\widehat{\mathbf{f}}[n]$ is the decoded n-th frame.

Meanwhile, for BVH data, mean error (ME) [12], which is the average error per joint, is a more common metric. Let N_j be the number of joints in the skeleton, and let $\mathbf{p}_j[n]$ be the 3D coordinate of the j-th joint in frame n. Then, the ME is defined as:

$$\text{ME} = \frac{1}{N\cdot N_j}\sum_{n=1}^{N}\sum_{j=1}^{N_j}\sqrt{(\mathbf{p}_j[n]-\widehat{\mathbf{p}}_j[n])^T(\mathbf{p}_j[n]-\widehat{\mathbf{p}}_j[n])}, \tag{25}$$

where $\widehat{\mathbf{p}}_j[n]$ is the decoded position of the j-th joint in frame n.

Table II shows the 20 MoCap sequences used in the training. These sequences were available in both C3D and BVH formats; thus, they provided a convenient training pool to estimate model parameters a and b in Section III. C3D coding is examined first in the next section, followed by BVH coding in Section V-C.

B. C3D MoCap coding

Four MoCap sequences from the CMU database [38] were used for testing C3D coding: 13_29 (jumping), 85_12 (breakdance), 86_02 (walking) and 86_08 (walking). Each test sequence consists of position signals for 41 markers attached to a human body, stored in the C3D format, and the sampling frequency is 120 Hz. All the experiments use $N_{\text{GOF}} = 30$. Four codecs are compared:

- MOD is the MoCap codec from [22] without rate control,
- RC is the codec from [22] with rate control and with fixed bit allocation ratio $\alpha = 10$,
- RCA is the MoCap codec that uses adaptive bit allocation in Algorithm 2 (Section III) without any post-processing,
- FIL is the same as RCA but with post-processing described in Section IV.

Hence, RCA and FIL are the new methods developed in the present paper, whereas MOD and RC are from previous works.

Sequence	Description	Total Frame
01_01	long jump	2750
05_02	dance - expressive arms, pirouette	1123
14_09	jump up to grab, reach for, tiptoe	3287
16_02	jump up once	466
17_06	whistle, walk jauntily	6200
47_01	walk forward turn around walk back	1319
49_01	modern dance, gymnastics	625
64_03	golf	443
85_01	jump twist	998
94_15	Indian dance	2210
105_11	lavish walk	1863
105_17	quick walk	1520
105_57	walk forward	1177
118_17	long jump	646
124_02	baseball pitch	1319
125_01	swimming	4257
131_04	start hop stop	1086
132_02	walk with arms out, balancing	1141
135_01	martial arts walks	2000
135_04	front kick	1316

TABLE II: MoCap training data from [38].

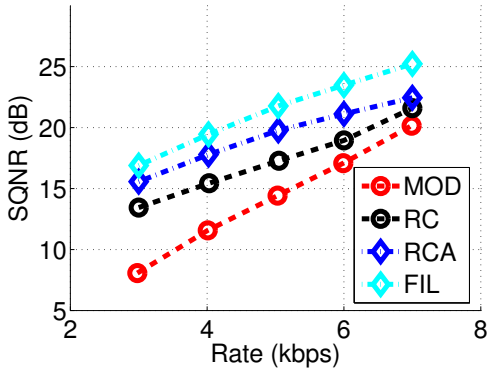
Seq.	BD SQNR (dB)			BD bitrate (%)		
	RC	RCA	FIL	RC	RCA	FIL
13_29	3.24	5.44	7.35	−19.96	−35.45	−44.16
85_12	2.04	3.47	5.42	−18.39	−29.51	−41.21
86_02	2.45	4.61	6.49	−8.41	−21.38	−34.31
86_08	3.94	6.12	8.02	−17.79	−30.53	−40.22

TABLE III: BD SQNR and BD bitrate compared to MOD for C3D data.

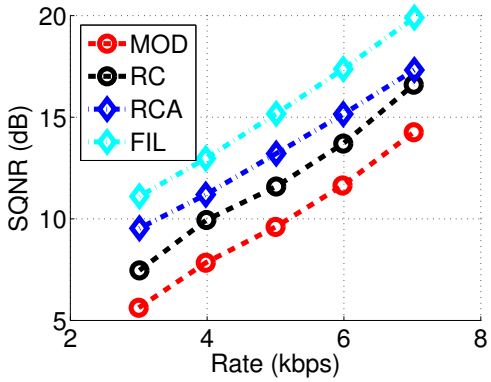
Figs. 8-9 show the operational rate-distortion curves of the four methods on the four test sequences for bitrates between 3 and 7 kbps. As shown in these figures, adaptive bit allocation (RCA) improves the performance by up to 2.5 dB compared to fixed bit allocation (RC), whereas post-processing (FIL) adds another 1-3 dB to the SQNR. In addition, we observe that the RCA curve approaches the RC curve as the bitrate increases to 7 kbps. This is because at 7 kbps, the optimal bit allocation ratio α is close to 10 (the fixed value used in RC) for a wide range of LTR/STR variance ratios, as illustrated in Fig. 4(a).

As a summary of the operational RD performance on C3D data, Table III shows the Bjontegaard Delta (BD) [39] for SQNR and bitrate using MOD as the reference. As shown in this table, RC provides, on average, 8-20% bitrate reduction compared to MOD at the same SQNR. RCA offers additional bitrate reduction by 10-15% over RC, whereas post-processing (FIL) allows an additional 9-13% reduction in bitrate. Overall, the full-featured method FIL, which includes both adaptive bit allocation and post-processing, developed in this paper improves the SQNR by 5-8 dB (alternatively, reduces the bitrate by 34-44%) compared to the previous state-of-the-art for online MoCap coding [22].

As indicated by the above results, adaptive bit allocation

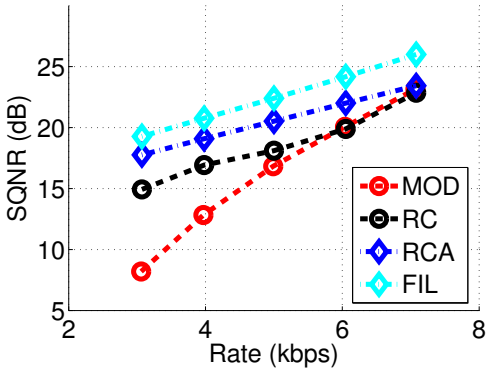


(a) 13_29

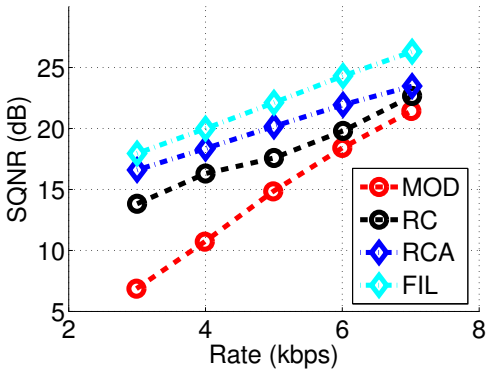


(b) 85_12

Fig. 8: SQNR vs. bit rate (C3D).



(a) 86_02



(b) 86_08

Fig. 9: SQNR vs. bit rate (C3D).

Target	RC		RCA	
	Actual	Error (%)	Actual	Error(%)
2.98	3.00	0.67	3.00	0.67
4.01	4.03	0.50	4.02	0.25
5.05	5.05	0.40	5.04	0.20
5.99	6.00	0.17	6.00	0.17
6.99	7.00	0.14	7.00	0.14

TABLE IV: Rate control results in terms of bitrate (kbps) on 13_29 (C3D).

Target	RC		RCA	
	Actual	Error (%)	Actual	Error(%)
3.00	3.01	0.33	3.01	0.33
3.98	3.99	0.25	3.98	0.00
5.00	5.01	0.20	5.01	0.20
5.98	5.99	0.17	5.99	0.17
7.02	7.02	0.00	7.03	0.14

TABLE V: Rate control results in terms of bitrate (kbps) on 85_12 (C3D).

improves the operational RD performance of MoCap coding. However, it is also important to examine whether the rate control performance is compromised in the process. To this end, we encoded the test sequences at several bitrates using the MOD codec [22]; then, we measured the exact bitrate produced by the MOD codec and used those values as the target bitrates for RC and RCA. The results for 13_29 and 85_12 are shown in Tables IV and V, respectively, and similar results were also obtained on other sequences. As shown in the tables, RCA has a very similar overall rate control performance as RC, despite having better RD performance. The errors in the final produced bitrates are well within 1% of the target bitrate. Note that post-processing does not play a role in this case; thus, the FIL results are not shown (since they are the same as those of RCA).

To examine the instantaneous bitrate, Fig. 10 illustrates the actual number of bytes produced per GOF by the MOD, RC and RCA codecs at the target bitrate of 6 kbps. As shown in Fig. 10, the instantaneous bitrate of MOD fluctuates widely, whereas RC and RCA both keep the instantaneous bitrate much closer to the target.

Table VI shows a more quantitative view of the instantaneous bitrate produced by the three methods. In particular, this table shows the standard deviation of bytes per GOF across five target bitrates for each test sequence to quantify how much the instantaneous bitrate fluctuates around the target value. As shown in this table, RC and RCA produce considerably lower standard deviations than MOD, indicating that they are better able to keep the instantaneous bitrate close to the target value. We do observe, however, that RCA produces a slightly higher standard deviation, which was not obvious in Fig. 10. Therefore, we conclude that the rate control performance of RCA is slightly worse than that of RC. However, this slight loss in rate control performance comes with a considerable

Sequence	MOD	RC	RCA
13_29	42.5	12.9	14.5
85_12	50.4	10.4	9.7
86_02	60.3	10.8	13.1
86_08	51.5	9.3	10.4

TABLE VI: Average standard deviation of bytes per GOF on C3D data.

gain in RD performance, which we believe is a good tradeoff.

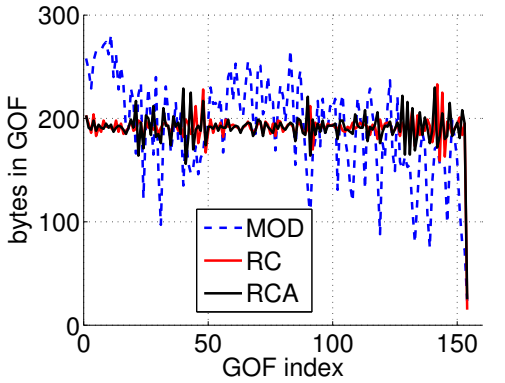
C. BVH MoCap coding

As previously mentioned, in the animation community, the BVH format for skeletal motion data has been very popular. Although our codec was developed for the general C3D data, it is also interesting to compare it against state-of-the-art BVH codecs. For this purpose, we used four BVH MoCap sequences from the CMU database: 17_08, 17_10, 85_12, and 15_04. Compression results on these sequences were reported in [12] for several BVH codecs, which allows us to compare the BVH version of our codec to these earlier codecs.

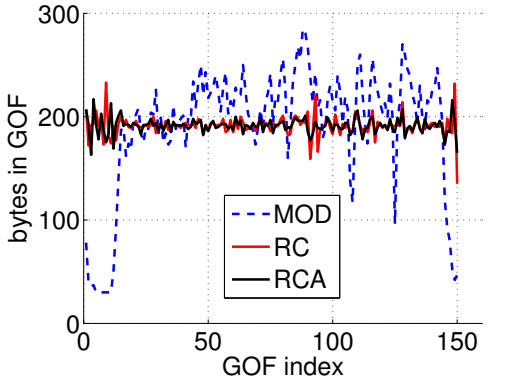
A popular way to compute the distortion of a compressed BVH file is via mean error (ME) [12], shown in (25). To compute the ME for a typical BVH codec, as illustrated in Fig. 11(a), one would encode the BVH file and then decode it and convert Euler angles of the joints to 3D coordinates (denoted XYZ in the figure). The ME is then computed in the Cartesian 3D coordinate space. Since our codec operates on C3D data, which is already in the XYZ format, we first take the BVH file and convert it to XYZ coordinates, and then we run the codec on these coordinates and compute the ME, as shown in Fig. 11(b). For the BVH version of our codec, some minor modifications had to be made compared to the C3D version. For example, there are 38 joints in the BVH files used in this comparison, whereas there were 41 markers in the C3D MoCap files in the previous section. Additionally, the bit allocation model parameters a and b (Section III) are different from those for C3D. Nonetheless, parameter estimation for BVH data follows the same procedure as for C3D data, and the codec architecture (including post-processing) is the same.

We compared the BVH version of our codec to the three BVH codecs considered in [12]. One of them is the codec proposed in [12], which we denote VASA, and the other two are from earlier works: LTC from [11] and TWC from [10]. According to the results in [12], VASA is the state-of-the-art in BVH MoCap coding. Figs. 12 and 13 show the ME vs. file size for the four test sequences. The data points for VASA, LTC and TWC come from Table 1 in [12]. For our codec, we show two curves. One is for RC, which is the codec from [22], without adaptive bit allocation (using fixed $\alpha = 10$) and no post-processing. The other curve is for FIL, which is our full-featured codec that includes adaptive bit allocation from Section III and post-processing from Section IV.

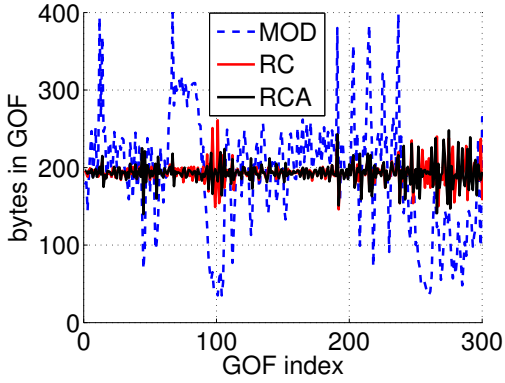
As shown in the figures, even RC by itself is better than LTC and TWC in most cases and competitive with VASA on 85_12 and 17_10. Meanwhile, FIL outperforms VASA on 85_12 and 17_10 and achieves comparable performance on the other two sequences. This result is particularly encouraging considering



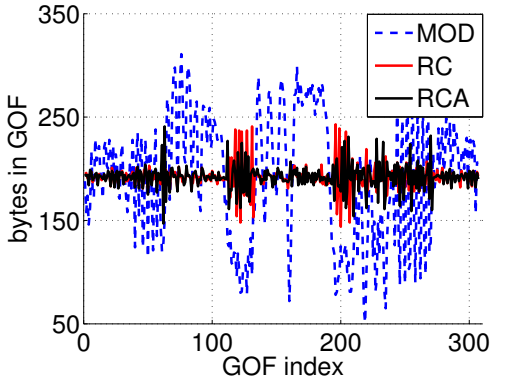
(a) 13_29



(b) 85_12

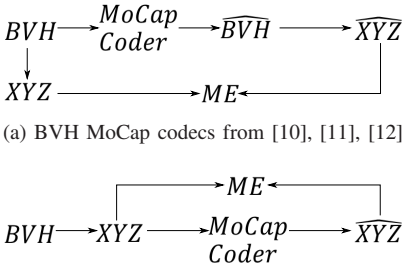


(c) 86_02



(d) 86_08

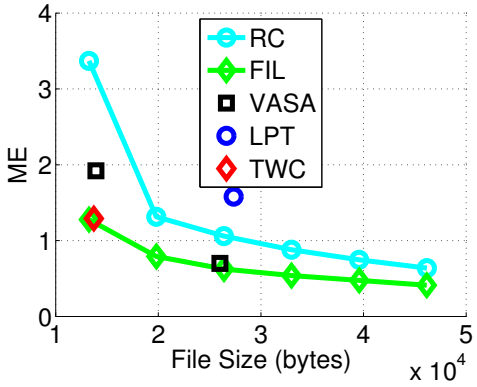
Fig. 10: Bytes per GOF at 6 kbps for C3D



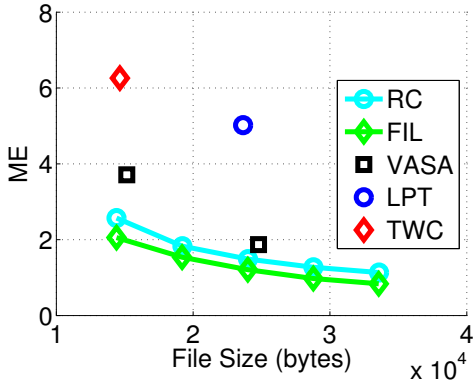
(a) BVH MoCap codecs from [10], [11], [12]

(b) BVH version of the proposed MoCap codec

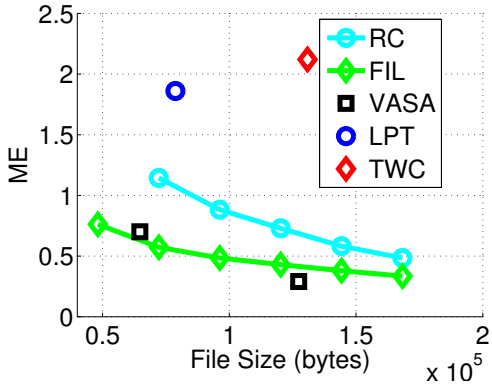
Fig. 11: Computing ME for BVH MoCap codecs in the comparison.



(a) 17_08

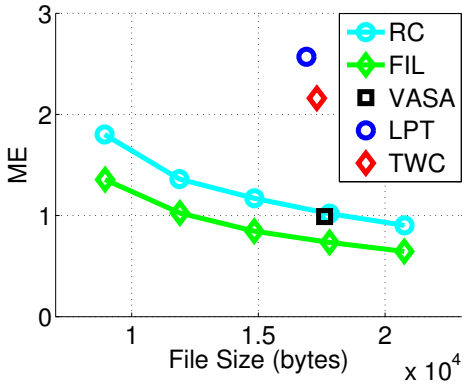


(a) 85_12



(b) 15_04

Fig. 12: ME vs. file size



(b) 17_10

Fig. 13: ME vs. file size

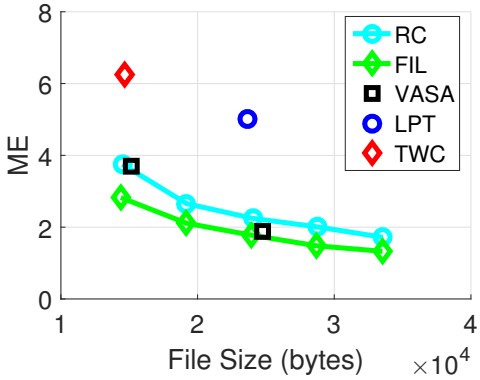


Fig. 14: ME vs. file size including virtual markers for 85_12

that our codec is an *online* codec with very low end-to-end delay, suitable for interactive applications. For illustration, end-to-end delay (in frames), which includes both encoding and decoding delays, is shown in Table VII. As shown, the existing BVH codecs typically require access to all the frames in the sequence before even starting the encoding process. Meanwhile, our encoder uses only causal prediction and is therefore able to encode a MoCap frame as soon as it is captured, without waiting for the next frame. The only delay in our FIL codec comes from post-processing at the decoder, and it is equal to approximately half the filter length of the post-processing filter.

Although the results in Figs. 12 and 13 indicate the accuracy of representing joint positions, these data are not directly usable in skinning applications for human motion animation because rolling rotations around bones are not captured by joint positions alone. To overcome this problem, the concept of "virtual markers" has been proposed [7]. In Fig. 14, we show the ME vs. file size for sequence 85_12, where virtual markers are included in the data. The inclusion of virtual markers makes the raw data size three times larger than just the joint positions [7], but it also makes it easier to use the data

Codec	End-to-end delay (frames)
VASA [12]	All frames in the sequence
LTC [11]	Variable, but at least 200
TWC [10]	All frames in the sequence
RC and RCA	0
FIL	12 (with 25-tap post-processing filter)

TABLE VII: End-to-end delay (in frames) for various codecs.

Target	RC		RCA	
	Actual	Error (%)	Actual	Error(%)
3.00	3.00	0.00	3.00	0.00
4.00	4.00	0.00	4.00	0.00
5.00	5.00	0.00	5.00	0.00
6.00	6.00	0.00	6.00	0.00
7.00	7.00	0.00	6.99	0.14

TABLE VIII: Rate control results in terms of bitrate (kbps) on 85_12 (BVH).

in skinning applications. By comparing Fig. 12(a) and Fig. 14, we observe that the inclusion of virtual markers increases the compressed file size by approximately 50% to 100%. This is less than the three-fold increase in raw data size because there is some statistical redundancy in the virtual markers, which is being exploited by our encoder. Regardless, our codec remains competitive with VASA and better than LTC and TWC.

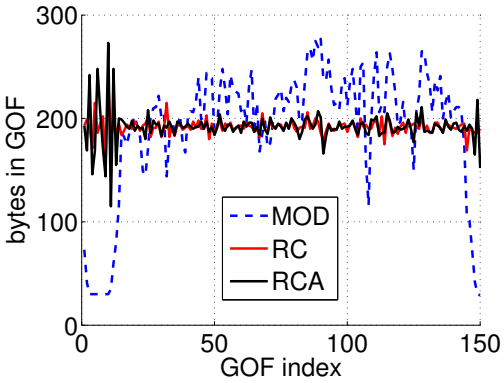
Next, we examine the accuracy of rate control on BVH data. We encoded test sequences with target bitrates set at 3, 4, 5, 6, and 7 kbps and measured the exact bitrate produced by RC and RCA (FIL has the same bitrate as RCA). The results are shown in Tables VIII and IX for 85_12 and 17_10, and similar results were obtained on the other two sequences. As shown in these tables, the rate control for BVH data is quite accurate; the actual produced bitrate is well within 1% of the target value, as was the case with the C3D data. The actual number of bytes produced per GOF by the MOD, RC and RCA codecs at 6 kbps is shown in Fig. 15. As was the case with the C3D data, we see that codecs with rate control (RC and RCA) produce a smaller deviation from the target compared to MOD. This is also confirmed quantitatively in Table X, which shows the standard deviation of the instantaneous bitrate.

Sample BVH data produced by the four codecs (MOD, RC, RCA, and FIL) at 7 kbps are shown in Fig. 16 for visual comparison. In this figure, the original skeleton is shown in red, and the reconstructed skeleton is shown in blue. As shown in this figure, the reconstructed skeleton approaches the original skeleton as the codec effectiveness improves, from MOD to RC, RCA and finally FIL. Several animation videos are available online[1] for further illustration, along with codec implementation.

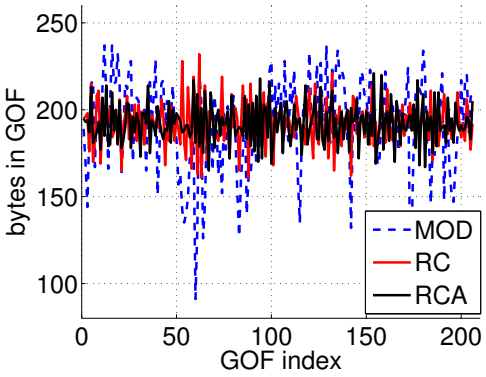
D. Subjective evaluation

Similar to [40], we conducted a subjective experiment to test the perceptual quality of animations produced from decoded bitstreams. Animations were produced from decoded C3D sequences 85_12 (breakdance) and 86_08 (walking) at four compression ratios (40:1, 60:1, 80:1, and 100:1) using

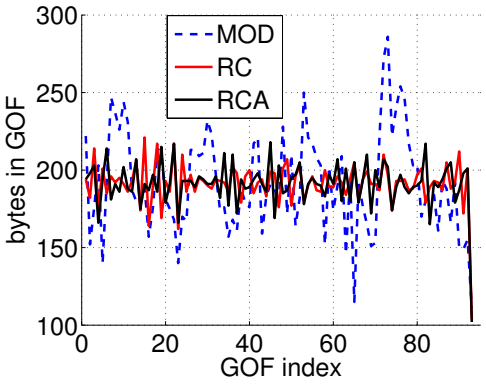
[1]http://geomm.ensc.sfu.ca/papers/MoCap-coding/



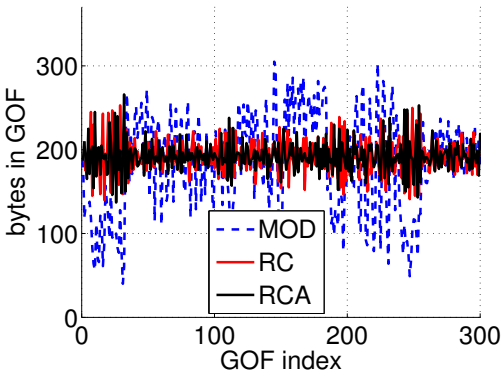
(a) 85_12



(b) 17_08



(c) 17_10

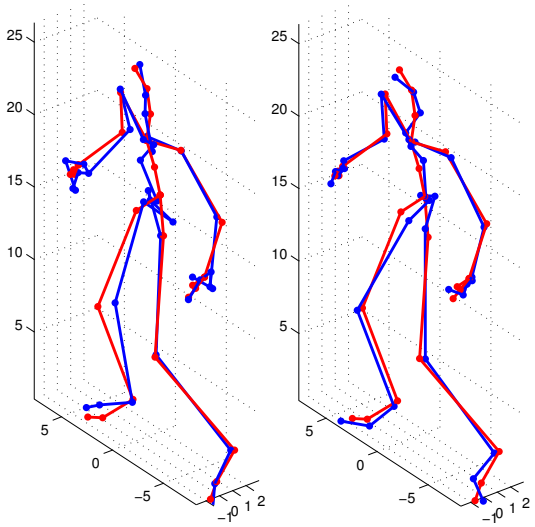


(d) 15_04

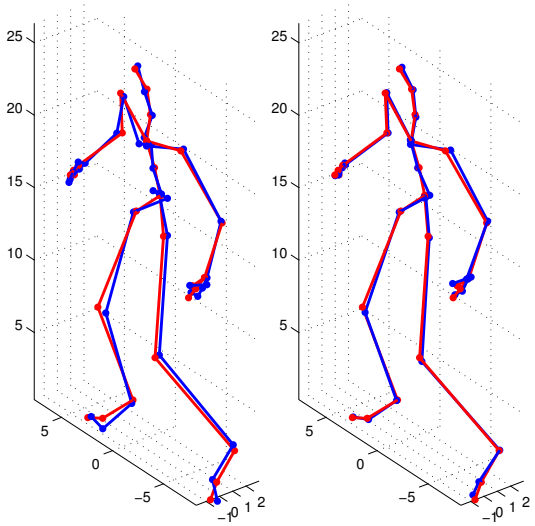
Fig. 15: Bytes per GOF at 6 kbps for BVH

Target	RC		RCA	
	Actual	Error (%)	Actual	Error (%)
3.00	3.01	0.33	3.02	0.67
4.00	4.01	0.25	4.01	0.25
5.00	5.00	0.00	5.00	0.00
6.00	6.00	0.00	6.00	0.00
7.00	7.00	0.00	6.99	0.14

TABLE IX: Rate control results in terms of bitrate (kbps) on 17_10 (BVH).



(a) MOD (ME = 0.7228) (b) RC (ME = 0.4658)



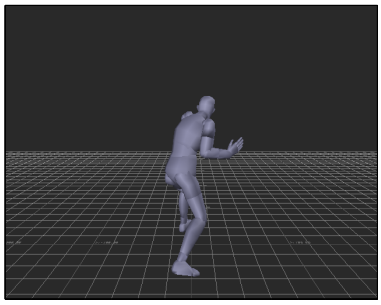
(c) RCA (ME = 0.3829) (d) FIL (ME = 0.1947)

Fig. 16: Decoded BVH skeleton at frame 2008 from sequence 17_08 coded at 7 kbps with (a) MOD, (b) RC, (c) RCA, and (d) FIL. The original skeleton is shown in red, and the reconstructed skeleton is shown in blue. The mean error (ME) is shown below each figure.

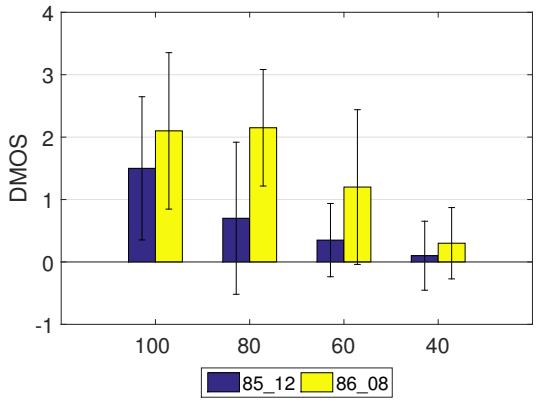
MotionBuilder. They were played to 20 participants on a Dell 1704FPTT monitor in accordance with the DSCQS protocol [41] with a 5-point quality scale (1-Bad, 2-Poor, 3-Fair, 4-Good, and 5-Excellent). A sample frame from the animation

Sequence	MOD	RC	RCA
85_12	51.7	11.9	12.4
17_08	24.9	10.5	9.3
17_10	30.4	11.9	12.6
15_04	56.7	16.6	13.9

TABLE X: Average standard deviation of bytes per GOF on BVH data.



(a)



(b)

Fig. 17: (a) Illustration of a frame from the animation used in subjective testing. (b) Average DMOS vs. compression ratio, with error bars indicating the standard deviation of DMOS.

is shown in Fig. 17(a). The results are shown in Fig. 17(b), where the bar height is the average differential mean opinion score (DMOS) between animations produced by compressed and uncompressed MoCap sequences, and the error bars indicate standard deviation. Similar to [40], we found that at a compression ratio of 40:1, DMOS is well within one standard deviation from zero, indicating that the animation produced from the compressed MoCap sequence is very similar to the one produced from the original uncompressed sequence. As the compression ratio increases, the difference becomes more obvious and DMOS becomes closer to 2.

E. Complexity

Finally, we present a note on complexity. Table XI shows the encoding and decoding times for RCA running on BVH data. These were measured in MATLAB R2011a on a laptop machine with an Intel Core i5-5200 @ 2.20 GHz processor. On average, the encoding takes 14 milliseconds per frame, whereas the decoding takes 11 milliseconds per frame. If

Sequence	Target	Enc.	Dec.	Dec. + post-proc.
85_12	3.00	0.0108	0.0089	0.0119
	4.00	0.0128	0.0096	0.0124
	5.00	0.0132	0.0100	0.0129
	6.00	0.0137	0.0105	0.0133
	7.00	0.0142	0.0110	0.0138
15_04	3.00	0.0116	0.0141	0.0168
	4.00	0.0183	0.0148	0.0159
	5.00	0.0171	0.0149	0.0164
	6.00	0.0175	0.0145	0.0156
	7.00	0.0167	0.0157	0.0170
17_08	3.00	0.0122	0.0098	0.0125
	4.00	0.0135	0.0102	0.0129
	5.00	0.0136	0.0104	0.0131
	6.00	0.0138	0.0104	0.0131
	7.00	0.0134	0.0105	0.0132
17_10	3.00	0.0158	0.0084	0.0111
	4.00	0.0124	0.0087	0.0116
	5.00	0.0129	0.0092	0.0119
	6.00	0.0131	0.0095	0.0123
	7.00	0.0133	0.0098	0.0126
Average		0.0140	0.0110	0.0135

TABLE XI: Encoding and decoding times in seconds per frame for BVH data at various target bitrates (in kbps).

post-processing is enabled, the average decoding time is 13.5 milliseconds per frame.

In comparison, the codec from [12] implemented as a Windows executable and tested on a somewhat more powerful processor (Intel Core i7-920 CPU @ 2.67 GHz) was reported to encode up to 3000 frames per second without pre-processing, whereas the pre-processing took approximately twice the time of the actual encoding, bringing the total encoding time to 1000 frames per second. This is equivalent to 1 millisecond per frame, or approximately 14 times faster than our encoding. However, we believe that an optimized version of our encoder implemented as an executable on a faster processor could be comparable with that of [12] in terms of speed. As an illustration, simply replacing MATLAB-based adaptive arithmetic codec by an executable version generated from C++ would bring our encoding time per frame down to 5.2 milliseconds and decoding time down to 2.6 milliseconds.

VI. Conclusions and Future Work

In this paper, we presented an online MoCap codec with several novel features. The first is an adaptive bit allocation between LTR and STR frames, which works within a rate control module and decides on the bit allocation ratio that will minimize the distortion for the given bitrate. The second novel feature is a motion-adaptive post-processing strategy that filters the decoded signals and suppresses quantization noise at high frequencies. Combined, the two new features bring about a significant gain in operational rate-distortion performance, with a minimal increase in algorithmic delay and with excellent rate control accuracy. Moreover, we adapted our MoCap codec to the popular BVH skeletal data format and compared it with state-of-the-art BVH codecs. The comparison showed that the BVH version of our codec is competitive with state-of-the-art BVH codecs in terms of rate-distortion performance, despite being an online codec and using only causal encoding. In addition, unlike other BVH codecs, our approach offers accurate rate control, making it more suitable for interactive applications across a network, such as online gaming. Future improvements to online MoCap coding could include more advanced spatial transforms (for example, graph-based transform [26]), which might be able to better exploit the structure of MoCap data.

References

[1] F. Mokaya, B. Nguyen, C. Kuo, Q. Jacobson, and P. Zhang, "[MARS] a real time motion capture and muscle fatigue monitoring tool," in *Proc. ACM Conf. Embedded Network Sensor Systems*, 2012, pp. 385–386.

[2] E. Wade and M. J. Mataric, "Design and testing of lightweight inexpensive motion-capture devices with application to clinical gait analysis," in *Proc. IEEE Int. Conf. Pervasive Computing Technologies for Healthcare (PervasiveHealth'09)*, 2009, pp. 1–7.

[3] A. Andreadis, A. Hemery, A. Antonakakis, G. Gourdoglou, P. Mauridis, D. Christopoulos, and J. N. Karigiannis, "Real-time motion capture technology on a live theatrical performance with computer generated scenery," in *Proc. IEEE PCI'10*, Tripolis, Greece, 2010, pp. 148–152.

[4] Q. Wu, P. Boulanger, M. Kazakevich, and R. Taylor, "A real-time performance system for virtual theater," in *Proc. ACM SMVC*, 2010, pp. 3–8.

[5] F. Hülsken, C. Eckes, R. Kuck, J. Unterberg, and S. Jörg, "Modeling and animating virtual humans for real-time applications," *Int. J. Virtual Reality*, vol. 6, no. 4, pp. 11–20, 2007.

[6] C. Dobrian and F. Bevilacqua, "Gestural control of music: using the vicon 8 motion capture system," in *Proc. Conf. New Interfaces for Musical Expression*. National University of Singapore, 2003, pp. 161–163.

[7] O. Arikan, "Compression of motion capture databases," *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 890–897, 2006.

[8] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computers & Graphics*, vol. 28, no. 1, pp. 25–34, 2004.

[9] G. Liu and L. McMillan, "Uniform threshold scalar quantizer performance in Wyner-Ziv coding with memoryless, additive Laplacian correlation channel," in *Proc. SCA'06*, Sep. 2006, pp. 127–135.

[10] M. Tournier, X. Wu, N. Courty, E. Arnaud, and L. Reveret, "Motion compression using principal geodesics analysis," *Computer Graphics Forum*, vol. 28, no. 2, pp. 355–364, 2009.

[11] I.-C. Lin, J.-Y. Peng, C.-C. Lin, and M.-H. Tsai, "Adaptive motion data representation with repeated motion analysis," *IEEE Trans. Visualization and Computer Graphics*, vol. 17, no. 4, pp. 527–538, 2011.

[12] L. Váša and G. Brunnett, "Rate-distortion optimized compression of motion capture data," *Computer Graphics Forum*, vol. 33, no. 2, pp. 283–292, 2014.

[13] S. Chattopadhyay, S. M. Bhandarkar, and K. Li, "Human motion capture data compression by model-based indexing: A power aware approach," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, no. 1, pp. 5–14, 2007.

[14] P. Beaudoin, P. Poulin, and M. van de Panne, "Adapting wavelet compression to human motion capture clips," in *Proc. Graphics Interface (GI'07)*, 2007, pp. 313–318.

[15] A. Firouzmanesh, I. Cheng, and A. Basu, "Perceptually guided fast compression of 3-D motion capture data," *IEEE Trans. Multimedia*, vol. 13, no. 4, pp. 829–834, 2011.

[16] C.-H. Lee and J. Lasenby, "An efficient wavelet-based framework for articulated human motion compression," in *Advances in Visual Computing*. Springer, 2008, pp. 75–86.

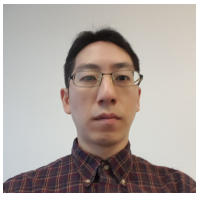
[17] S. Li, M. Okuda, and S.-i. Takahashi, "Compression of human motion animation using the reduction of interjoint correlation," *EURASIP J. Image and Video Processing*, vol. 2008, pp. 1–15, 2008.

[18] M. Preda, B. Jovanova, I. Arsov, and F. Prêteux, "Optimized MPEG-4 animation encoder for motion capture data," in *Proc. ACM Web 3D*, May 2007, pp. 181–190.

[19] B. Jovanova, M. Preda, and F. Preteux, "MPEG-4 part 25: A generic model for 3D graphics compression," in *Proc. IEEE 3DTV Conference (3DTV-CON)*, 2008, pp. 101–104.

[20] J. Hou, L.-P. Chau, N. Magnenat-Thalmann, and Y. He, "Scalable and compact representation for motion capture data using tensor decomposition," *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 255–259, March 2014.

[21] C.-H. Kwak and I. V. Bajić, "Hybrid low-delay compression of motion capture data," in *Proc. IEEE ICME'11*, Barcelona, Spain, Jul. 2011, pp. 1–6.
[22] ——, "Mocap data coding with unrestricted quantization and rate control," in *Proc. IEEE ICASSP'13*, Vancouver, BC, May 2013, pp. 3741–3745.
[23] ——, "Error concealment strategies for motion capture data streaming," in *Proc. IEEE ICME'11 Workshops - StreamComm*, Barcelona, Spain, Jul. 2011, pp. 1–6.
[24] J. Hou, L.-P. Chau, N. Magnenat-Thalmann, and Y. He, "Low-latency compression of mocap data using learned spatial decorrelation transform," *Comput. Aided Geom. D.*, vol. 43, no. C, pp. 211–225, Mar. 2016.
[25] J. Hou, L. P. Chau, N. Magnenat-Thalmann, and Y. He, "Human motion capture data tailored transform coding," *IEEE Trans. Vis. Comput. Graphics*, vol. 21, no. 7, pp. 848–859, Jul. 2015.
[26] J. Y. Kao, A. Ortega, and S. S. Narayanan, "Graph-based approach for motion capture data representation and analysis," in *Proc. IEEE ICIP*, Oct. 2014, pp. 2061–2065.
[27] D. S. Taubman and M. W. Marcellin, *JPEG2000: Image compression fundamentals, standards, and practice*. Kluwer Academic Publishers, 2002.
[28] M. Servais, *Range Coding in MATLAB*. [Online]. Available: http://www.ee.surrey.ac.uk/CVSSP/VMRG/hdtv/code.htm.
[29] V. Sheinin, A. Jagmohan, and D. He, "Uniform threshold scalar quantizer performance in Wyner-Ziv coding with memoryless, additive Laplacian correlation channel," in *Proc. IEEE ICASSP'06*, vol. 4, Toulouse, France, May 2006, pp. 217–220.
[30] V. Sheinin and A. Jagmohan, "Low rate uniform scalar quantization of memoryless Gaussian sources," in *Proc. IEEE ICIP'06*, Atlanta, GA, 2006, pp. 793–796.
[31] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.
[32] E. C. Ifeachor and B. W. Jervis, *Digital Signal Processing: A Practical Approach*. Addison-Wesley, 1993.
[33] R. Azuma and G. Bishop, "A frequency-domain analysis of head-motion prediction," in *Proc. ACM Conf. Computer Graphics and Interactive Techniques*, 1995, pp. 401–408.
[34] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time signal processing*, 2nd ed. Prentice Hall, 1999.
[35] H. Stark and J. W. Woods, *Probability, statistics, and random processes for engineers*, 4th ed. Pearson Prentice Hall, 2012.
[36] M. Claypool and K. Claypool, "Latency and player actions in online games," *Communications of the ACM*, vol. 49, no. 11, pp. 40–45, 2006.
[37] K. Raaen and T.-M. Grønli, "Latency thresholds for usability in games: A survey," in *Proc. Norwegian Informatics Conference (NIK)*, Nov. 2014, pp. 1–12.
[38] *The CMU Motion of Body (MoBo) Database*. [Online]. Available: http://mocap.cs.cmu.edu.
[39] G. Bjontegaard, "Calculation of average PSNR differences between RD-curves," *ITU-T VCEG-M33*, 2001.
[40] I. Cheng, A. Firouzmanesh, and A. Basu, "Perceptually motivated LSPIHT for motion capture data compression," *Computers & Graphics*, vol. 51, pp. 1–7, Oct. 2015.
[41] ITU-R, "Recommendation ITU-R BT.500-13: Methodology for the subjective assessment of the quality of television pictures," Jan. 2012.



Choong-Hoon Kwak received the B.S. and M.S. degrees in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1999 and 2002. He is currently pursuing the Ph.D degree in the School of Engineering Science at Simon Fraser University, Burnaby, BC, Canada. His current research interests include signal processing, motion capture and dynamic 3D mesh data compression, and multimedia communications.

Ivan V. Bajić (S'99-M'04-SM'11) is Associate Professor of Engineering Science at Simon Fraser University, Burnaby, BC, Canada. His research interests include signal, image, and video processing and compression, multimedia ergonomics, and communications. He has authored about a dozen and co-authored another eight dozen publications in these fields. He has served on the organizing and/or program committees of various conferences in the field, including GLOBECOM, ICC, ICME, and ICIP. He was the Chair of the Media Streaming Interest Group of the IEEE Multimedia Communications Technical Committee from 2010 to 2012. He is currently serving as Associate Editor of IEEE TRANSACTIONS ON MULTIMEDIA and IEEE SIGNAL PROCESSING MAGAZINE, and the Chair of the Vancouver Chapter of the IEEE Signal Processing Society.