Adapting Wavelet Compression to Human Motion Capture Clips

Philippe Beaudoin Pierre Poulin
Université de Montréal
{beaudoin,poulin}@iro.umontreal.ca

Michiel van de Panne
University of British Columbia
van@cs.ubc.ca

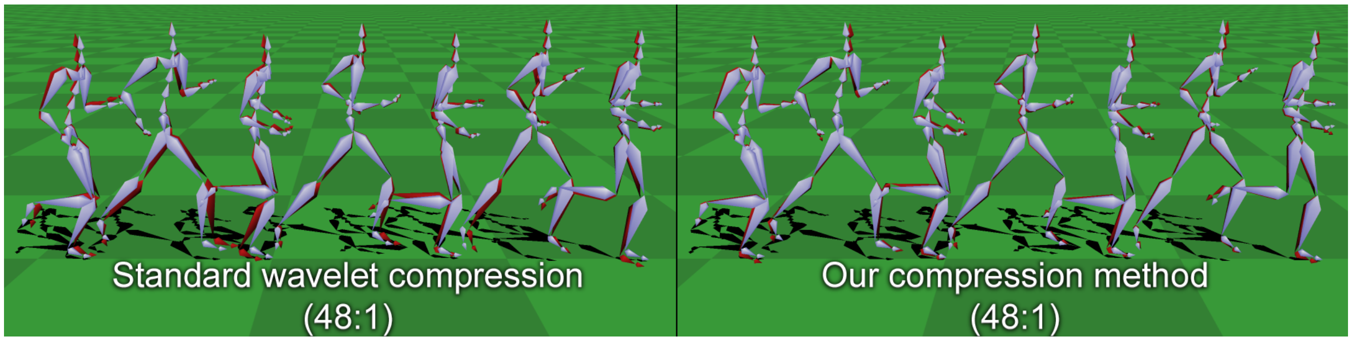

Figure 1: Comparison between standard wavelet compression and optimized wavelet coefficient selection with inverse kinematics correction. The compressed sequence is shown in red with the original sequence overlaid in gray in both images.

Abstract

Motion capture data is an effective way of synthesizing human motion for many interactive applications, including games and simulations. A compact, easy-to-decode representation is needed for the motion data in order to support the real-time motion of a large number of characters with minimal memory and minimal computational overheads. We present a wavelet-based compression technique that is specially adapted to the nature of joint angle data. In particular, we define wavelet coefficient selection as a discrete optimization problem within a tractable search space adapted to the nature of the data. We further extend this technique to take into account visual artifacts such as footskate. The proposed techniques are compared to standard truncated wavelet compression and principal component analysis based compression. The fast decompression times and our focus on short, recomposable animation clips make the proposed techniques a realistic choice for many interactive applications.

CR Categories: I.3.7 [Three-Dimensional Graphics and Realism]: Animation—Compression

Keywords: Compression, Wavelet, Skeletal Animation, IK correction

1 Introduction

Motion capture data enables rapid production of huge collections of skeletal animations. As a result, it is now a common method for synthesizing character motion in video games, simulations, and controllable avatars. As the amount of data contained in such collections increases, it becomes crucial to develop compression techniques suited to skeletal animations. Lossless compression may be useful for various applications, but the best gains can be obtained when we take into consideration human perceptual limitations. This paper is therefore interested in lossy compression of skeletal animation data.

A "good" compression technique depends to a significant degree on the characteristics of the application. Is the cache footprint important? Is it important to have fast access to a subset of the joint data? Is it perceptually important to have accurate foot placement? Is it important that motion clips be compressed independently of each other? In our work, we shall assume that the answer to each of the above questions is 'yes', based on the knowledge that many interactive applications need to: animate multiple characters in real time using minimal resources; support multi-way blending between multiple motions; support separate blending of individual limbs or upper and lower bodies; support accurate foot placement; and need tools and techniques that integrate easily with current pipelines for processing and animating motion capture data.

Typical motion capture data exhibits *temporal coherence* and *joint correlations*, both of which can be exploited for the purposes of compression. The positions and orientations of a character's body parts vary smoothly over time when viewed at an appropriately small time scale, thus being amenable to being modeled by splines, wavelets, or even local linear dynamical systems. The correlated movement between joints arises from the highly coordinated and structured nature of human motions. Given these two types of correlations, some fundamental questions arise. How much compression can be achieved by exploiting only temporal correlation, or using only joint correlation? How should these two types of compression be ordered so as to exploit both correlations?

Recent work in animation and compression provides answers to some of these questions. Principal component analysis (PCA) applied globally to motion capture data shows that 10-20 principal components are typically required to represent the motion of a typical human skeleton modeled using 40-60 degrees of freedom (DOF) [18, 15]. This typically achieves a compression factor of 2-4, once the costs of the reconstruction matrix have been amortized, and assuming similar quantization for the original and compressed data. Applying PCA to well-chosen coherent local motion segments [15] can yield further improvements, requiring on the order of 3-5 principal components for typical motion segments. This is somewhat offset by the additional storage required by the new bases for each motion segment. There are fewer answers related to temporal compression alone. Arikan [1] notes that simple subsampling provides 12:1 compression on the 120 Hz CMU database for the particular level of visual quality used as a benchmark to compare techniques. ¿From this, the largest gains may come from temporal compression rather than joint correlations. The same work reports compression ratios in the range of 30-35:1 using a scheme that exploits both joint correlations (PCA) and temporal coherence.

Our work looks only at temporal compression. We avoid combining this with exploitation of the joint correlations for several reasons. First, it is useful to gain an understanding of how much compression *the right kind* of temporal schemes can achieve on their own. Second, compressing individual channels of motion allows for flexibility in the reuse of motions. For example, the upper body of one motion may be combined with the lower body of another motion. This allows for motion variety and gives game engines flexibility to let characters achieve their goals. Last, interactive applications often use a linear blend of multiple motions in order to do parameterized interpolation between motions and also to blend between successive motion clips. When combined with the need to animate more than 10 characters simultaneously, this may easily lead to upwards of 30 motions being decompressed simultaneously with limited resources. In these situations, the cache footprint and computational overhead required to support PCA-based schemes are potentially problematic, and thus applications are likely to eschew maximal compression in favor of one that has lower computational or memory requirements.

We use Euler joint angles as our underlying motion representation and show that with appropriate care and attention, joint space can be a suitable domain for motion compression. Joint angle representations are an integral part of current game and simulation engines, and trivially support blending between motions. An alternative that is explored in two examples of recent work [1, 15] is to use internal virtual marker representations as a basis for motion compression. This necessitates extra overhead in converting to the original joint angle representations in order to support traditional run-time engines for the display of animated characters.

The specific contributions of this paper are as follows. We propose two techniques for adapting a wavelet compression scheme to joint angle data. We evaluate these ideas by comparing them against PCA alone, as well as an unoptimized wavelet scheme (see Figure 1). The compression schemes we introduce are well suited to the constraints of real-time character animation that result from requiring the simultaneous decompression of a large number of motions. We conclude that the right kind of temporal-only, joint-angle-based compression schemes offer compression ratios not far removed from other schemes that take joint correlations into account. Two recent papers [15, 1] note that joint angle data is itself not suitable for compression because of the hierarchical nature of the limbs. We show that an *appropriately adapted* compression scheme *can* achieve high quality compression directly on joint angle data.

The remainder of this paper is structured as follows. Section 2 presents related work, while Section 3 introduces a number of preliminary definitions used throughout the paper. Section 4 describes the direct application of truncated wavelet compression techniques to skeletal animation data. Section 5 defines the wavelet coefficient selection as a discrete optimization problem and shows how it can be efficiently solved in the context of skeletal animation data. This technique is further extended using inverse kinematics (IK) to handle situations where body-environment interactions are important. Section 6 presents results and discussions, followed by conclusions in Section 7.

2 Related Work

The need for compression arises in many domains. We focus our discussion on the compression of skeletal animation data, although we note that there are also methods focused on the related-but-distinct problem of compression of animated meshes, *e.g.*, [4]. We begin by noting that reduced bases and keyframe extraction methods have been proposed for applications other than compression. Representations of poses in a reduced dimensional space have been used for animation retrieval [7], for the development of various motion editing tools [18, 3, 9], and for motion synthesis [17, 8, 5]. Temporal simplification has also been used to extract key poses in an animation sequence [14, 11, 2], for space-time optimization [16], or for motion editing [12, 13].

Liu and McMillan [15] compress the raw 3D marker positions obtained from motion capture by breaking the motion into contiguous segments that can be compactly expressed in a reduced PCA basis. Temporal coherence is then exploited by adaptively fitting cubic splines to the reduced-basis coefficients and only storing the keyframes for the resulting cubic splines. To minimize footskate, the foot markers are compressed separately from the remaining markers and a tighter PCA residual error tolerance is used. A full-body IK-based post-process must be applied when angular values are desired. Compression ratios on the order of 50-60:1 are reported for a given set of residual-error thresholds. They note that the piecewise PCA by itself gives a compression factor of approximately 8, while temporal compression provides an additional factor of roughly 7. Because this work compresses raw marker motion (with possibly redundant markers), the compression rates may not be directly comparable to our work, which compresses the joint angles more commonly used as a motion capture data format.

Arikan [1] presents a motion capture database compression scheme. His technique exploits joint correlations and time coherence as applied to large motion databases. The technique is based on the use of virtual markers as an internal representation and thus requires conversion to and from this representation. A wavelet compression based on Haar wavelets is used as a point of comparison. Our work differs in that we investigate a scheme that can compress motion clips independently of each other and that allows for the efficient extraction of individual channels of motion from the compressed data. We work directly with joint angles and use a cubic interpolating spline wavelet basis appropriate to the continuous nature of animation data.

3 Preliminary Definitions

This paper focuses on compression techniques based on a truncated wavelet representation of the degrees of freedom. The wavelet basis is a natural choice given its use for compressing various other time-dependent signals such as audio. It yields a better rate-distortion than piecewise linear approximations [14] at a minimal increase in computational cost. We use a cubic interpolating bi-orthogonal wavelet basis built using the lifting scheme [19]. We noticed some artifacts due to the lack of continuity in the reconstructed signals when wavelets of a degree inferior to 3 were used. The interpolating basis is a matter of convenience; it allows us to build the signal directly from the samples without having to first convert it to a non-interpolating representation such as B-Splines.

The data we are working with contains k DOF: 3 signals tracking the position of the skeleton root and $k-3$ signals tracking the Euler angles at the different joints. For our results, $k = 62$. A complete animation is therefore represented by n samples in a k-dimensional space noted $\Theta(t) = (\theta_0(t), ..., \theta_{k-1}(t))$, with $(\theta_0(t), \theta_1(t), \theta_2(t))$ being the root position. The result of a lossy compression applied to this animation will be noted $\hat{\Theta}(t)$.

3.1 Distortion Metrics

In order to evaluate the efficiency of a compression method, it is necessary to evaluate its distortion. Since the data is meant to be viewed by human observers, the distortion should depend on the visual accuracy of the compressed animation and account for known perceptual limitations. Measuring such a distortion is difficult and is typically done by visual inspection of the final results. However, we shall still require a formally defined distortion metric in order to yield repeatable numerical results. This metric will also be used with the optimization-based compression scheme presented below.

The simplest metric is the RMS error in the DOF. However, as noted earlier, animation quality usually depends much more on the accuracy of the 3D positions of the joints than that of the angles that control them. Although both measures are related, the effect of a small angular error on the final pose depends both on the model hierarchy and its current pose. Given the static skeletal information and the value of all the angles at time t, it is easy to compute, as a series of matrix multiplications, the 3D positions of the joints of an animation. If the skeleton contains p joints, then these positions can be expressed as p 3D vectors $(\mathbf{x}_0(t), ..., \mathbf{x}_{p-1}(t))$. In a similar way we can define a corresponding set of 3D vectors $(\hat{\mathbf{x}}_0(t), ..., \hat{\mathbf{x}}_{p-1}(t))$ for the compressed animation. Finally, the static skeleton information gives us the length $(l_0, ..., l_{p-1})$ of the bone driven by each joint. These values can be used to define the positional distortion:

$$\varepsilon_{\mathbf{x}} = \sqrt{\frac{1}{n}\sum_{t=0}^{n-1}\sum_{i=0}^{p-1}|\mathbf{x}_i(t) - \hat{\mathbf{x}}_i(t)|^2 \frac{l_i}{l}}.$$

The fraction l_i/l, where $l = \sum l_i$, is used to weight the joints based on their local influence. A joint driving a finger has less impact on the positional error over the skeleton than a joint driving a forearm.

3.2 PCA Compression

For comparison purposes, we introduce a simple compression scheme based on PCA. This method works by building the covariance matrix of the angular DOF $\tilde{\Theta}(t) = (\theta_3(t), ..., \theta_{k-1}(t))$, corresponding to poses without the root position. The eigenvectors corresponding to the largest eigenvalues are then built into a matrix defining a lower-dimensionality linear space into which each pose $\tilde{\Theta}(t)$ is projected. The number of eigenvectors retained will then dictate the size of the compressed data and the quality of the reconstruction.

The data required for reconstruction of the original set of signals is: the projection matrix; the reduced basis representation of the motion over time corresponding to the projected $\tilde{\Theta}(t)$; three signals corresponding to $(\theta_0(t), \theta_1(t), \theta_2(t))$; and a $(k-3)$-dimensional vector containing the mean of each $\tilde{\Theta}(t)$. We finally quantize all stored data to 16 bits, which was determined experimentally to yield good rate-distortion results.

4 Standard Wavelet Compression

In this section we first show how standard truncated wavelet compression techniques can be used with skeletal animations. Given a signal of n samples perfectly represented by n wavelet coefficients, a target compression ratio r can be obtained by keeping the n/r largest coefficients and making the others equal to zero. In situations where wavelet encoding works well, most wavelet coefficients are very small and a heavily truncated wavelet representation faithfully reproduces the original signal.

For skeletal animations, the data is comprised of k different signals, one for each Euler angle and three for the root position. A first approach consists of considering each signal independently. In the case of truncated wavelet compression, this would mean keeping the n/r largest wavelet coefficients in each of the k signals. This method is optimal in a least-square sense for each of the k dimensions independently. However, it is not optimal if the data is viewed as a k-dimensional vector. A common value of r means that the same number of coefficients will be allocated to each signal, even if some of them contain strong variations while others are nearly constant.

Another approach consists of keeping the kn/r largest coefficients among all the kn wavelet coefficients representing the data. This produces a least-square optimal representation over the entire k-dimensional space. Using the selected kn/r coefficients, we can build a vector $\mathbf{w} \in [0,n]^k$ identifying how many coefficients are kept from each of the k signals.

Direct application of the above technique with skeletal animation data causes a problem, however, since some DOF encode Euler angles while others encode positions. In order to make these values comparable, we represent each angle in radians and multiply it by the length of the bone driven by the associated joint. This scaling is only performed to compute the $\mathbf{w}_i$, which are then used to compress the original signals.

Vector $\mathbf{w}$ is used to build the standard wavelet compression technique where the compressed animation is obtained using $\hat{\theta}_i(t) = \text{trunc}(\theta_i, \mathbf{w}_i)$. We use the notation $\text{trunc}(f, a)$ to represent the signal obtained from f by keeping only the a largest wavelet coefficients. Compressing the animation is simply a matter of performing a wavelet transform and then building $\mathbf{w}$ by identifying the kn/r largest coefficients. Decompression is achieved by an inverse wavelet transform. The fast wavelet transform algorithm, with linear execution time, allows this to be computed efficiently.

To efficiently encode the non-zero coefficients of each signal, we first order them from the widest to the narrowest wavelet basis. We then quantize each coefficient to 16 bits. In the spirit of run-length encoding, a small number of 16 bit codes are reserved to represent sequences of consecutive zeros. Another code is used to indicate a point past which all the remaining coefficients of the signal are null. As an optional step, we can further entropy-encode the data using the Lempel-Ziv [21] compressor *gzip* on a clip-per-clip basis. The use of *gzip* would be unsuitable for online use but is included here as an example of the gains that could be achieved using a custom entropy-encoding step.

5 Optimized Wavelet Coefficient Selection

For a given ratio r, standard wavelet compression minimizes the RMS error in the DOF. However, if the goal is to be perceptually faithful to the original animation, it is necessary to take positional distortion into account. A first way to achieve this is to redistribute the non-zero wavelet coefficients in an optimal way.

Formally, this problem consists of selecting, among all the kn wavelet coefficients, exactly kn/r coefficients to keep in order to minimize positional distortion. The size of the search space for this discrete optimization problem is $\binom{kn}{kn/r}$. Given the usually large number of samples n and the non-linear coupling between the joint angles and the final pose, this selection represents a daunting task.

We propose to simplify this problem to the selection of k discrete variables identifying how many of the n largest wavelet coefficients should be kept for each of the k signals. Formally, we seek a vector $\mathbf{m} \in [0,n]^k$ such that the compressed signals $\hat{\theta}_i(t) = \text{trunc}(\theta_i, \mathbf{m}_i)$ minimize positional distortion and that $\sum \mathbf{m}_i = kn/r$.

This simplification can be seen as a way to increase the importance of some signal components while reducing that of others. This would be a somewhat arbitrary process if we were dealing with generic temporal signals. However, in the case of skeletal animations, the signals correspond to joints organized in a hierarchical way and it seems natural to give more importance to angles that are high in the hierarchy.

One could think of obtaining the vector **m** using bone lengths or any other information available in the static skeleton. Unfortunately, this would be equivalent to considering the k signals independently, which was shown to be sub-optimal for standard wavelet compression. We find that the optimal choice of **m** depends not only on the static skeleton and the relative complexity of the different signals, but also on the poses in the animation. For example, in a boxing animation sequence where the actor holds his arms close to his body, a small rotation of the torso can have less influence on the positional distortion than a rotation at the elbow or the shoulder.

The optimal vector **m** will therefore be different for each animation clip. In this case, the size of the search space is approximately $\binom{kn/r}{k}$, which is much smaller than before. This discrete optimization problem can be solved in a number of ways. In our implementation, we use a heuristic simulated annealing technique described as Algorithm 1.

The algorithm sets vector **m** equal to vector **w** obtained with standard wavelet compression. The initial step size s is set to $\lfloor \max_i\{\mathbf{m}_i/2\}\rfloor$. Each successive step of the algorithm randomly selects an index $i \in [1,k]$ and the component $\mathbf{m}_i$ is immediately reduced by the value $s' = \min(s,\mathbf{m}_i)$. A linear search is then performed to identify another component $j \in [1,k]$ that, when increased by the value s', produces a new vector $\mathbf{m}'$ and new signals $\hat{\theta}'_i(t)$ that minimize positional distortion $\varepsilon_\mathbf{x}$.

```
1  $\mathbf{m} \leftarrow \mathbf{w}$
2  $s \leftarrow \lfloor \max_i\{\mathbf{m}_i/2\}\rfloor$
3  while $s \neq 0$ do
4      $i \leftarrow \text{random}(1,k)$
5      $s' \leftarrow \min(s,\mathbf{m}_i)$
6      $\mathbf{m}' \leftarrow \mathbf{m}$
7      $\mathbf{m}'_i \leftarrow \mathbf{m}'_i - s'$
8      Find $j \in [1,k]$ so $\mathbf{m}'_j + s'$ minimizes $\varepsilon_\mathbf{x}$
9      if $j \neq i$ then $\mathbf{m}'_j \leftarrow \mathbf{m}'_j + s'$, $\mathbf{m} \leftarrow \mathbf{m}'$
10     else if too many successive failures then $s \leftarrow \lfloor s/2 \rfloor$
11 end
```

Algorithm 1: Optimized coefficient selection.

If the search produces $j \neq i$, then the vector is updated as $\mathbf{m} \leftarrow \mathbf{m}'$. If we find $j = i$, then this particular choice of i has failed and the vector **m** remains unchanged. After a pre-defined number of successive failures, the step size is updated as $s \leftarrow \lfloor s/2 \rfloor$. In our implementation the step size is reduced after 10 successive failures. The algorithm terminates when s reaches 0 and the final vector **m** can be used to compress and store the data in the same way as standard wavelet compression.

This technique could be used with metrics other than positional distortion. However, if the chosen metric is not dependant enough on the joint hierarchy, then our initial assumption fails and the restricted search space can prove too limited to yield good optimization results. We experienced such problems when using a metric that penalized errors at joints interacting with the environment.

5.1 Inverse Kinematics Correction

The previous technique can achieve very satisfactory rate-distortion results with the positional distortion metric. However, this metric does not consider some events that are known to create important visible artifacts such as a contact foot sliding on the ground. Also, as mentioned earlier, our experiments showed that simply using an optimization metric penalizing such artifacts did not yield good results.

A first way to correct these errors would be to use the technique of Ikemoto *et al.* [10] that automatically detects and corrects footskate. This classifier-based method is useful when no correct reference animation is available but, in the case of compression, it risks introducing motions that were not present in the original sequence.

We propose to first use wavelet coefficient selection in order to obtain a lossy reconstruction of the motion. Then, for each pose showing contact errors, IK is applied to move the incorrect joints to their true positions obtained from the original motion. We did not address the issue of automatically identifying contact errors. Instead, since the feet are the most important source of such errors, we corrected their positions at every frame.

This requires us to store 3 extra signals for each foot. This constitutes an important overhead since our animations are expressed using 62 signals. It is therefore essential to compress the positional signals.

Trying to directly encode positions causes a problem. This is due to the fact that these signals have a very wide range of possible output values and encoding them with enough precision would require a lot of wavelet coefficients. Failure to keep enough coefficients could even have the catastrophic result of dramatically reducing the compression quality.

To overcome this problem, we encode a vector containing the difference between the real position and the position obtained after wavelet coefficient selection. This difference is close to zero and covers a much smaller range than the positions themselves, making it much easier to encode using a small number of wavelet coefficients. Moreover, there is no more risk of reducing the compression quality.

The signals containing these difference vectors are compressed independently from the DOF using standard wavelet compression. The compression ratio r_{IK} for the difference signals can be chosen independently from that of the other signals. Here again the coefficients are quantized to 16 bits and are further compressed using run-length codes and, optionally, *gzip*.

The choice of a good IK solver also has some impact on the final result. In particular, the solver should find a solution that does not lie too far from the initial pose. In our implementation, we used a simple multiple-constraint cyclic-coordinate descent [20], but more efficient techniques could also yield good results.

6 Results

The skeletons used to produce the results presented here all have the same hierarchy and DOF. Timings were obtained on a 2 GHz AMD Athlon 64 bit running the Linux operating system.

Assessing the perceptual quality of a compressed animation is very difficult to do and depends on many parameters. Common practices [1] involve the use of both a gross numerical evaluation of the error, such as the one we obtain with positional distortion $\varepsilon_\mathbf{x}$, as well as a qualitative evaluation through visual inspection of the results.

The various techniques mentioned in this section have been tested on a subset of 15 animations taken from the CMU motion capture library [6] and portray an array of actions and tasks. The sampling rate is 120 Hz and the test sequences ranged in size from 148 frames (around 1 sec.) to 5423 frames (45 sec.). Average results are presented as well as results for a short 148-frame running motion and a 2085-frame motion with an actor performing various kinds of jumps. The uncompressed files are arrays of IEEE 32 bit floats encoding the DOF in time.

6.1 Optimized Wavelet Coefficient Selection

We first study the rate-distortion behavior of optimized wavelet coefficient selection using positional distortion $\varepsilon_\mathbf{x}$. We compare it to

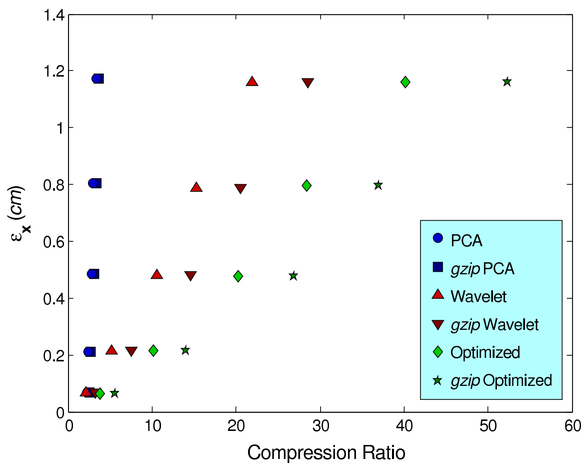

Figure 2: Average rate-distortion over the 15 test sequences for various compression techniques.

	ε_x	PCA	Wavelet	Optimized
Running	1.4	6.02 (6.0:1)	1.27 (28:1)	0.75 (48:1)
	0.96	6.49 (5.5:1)	1.51 (24:1)	0.94 (38:1)
	0.58	7.67 (4.7:1)	2.31 (16:1)	1.24 (29:1)
	0.26	9.42 (3.8:1)	3.69 (9.7:1)	2.18 (16:1)
	0.08	11.5 (3.1:1)	8.74 (4.1:1)	5.08 (7.0:1)
Jumping	0.67	153 (3.3:1)	18.6 (27:1)	9.97 (51:1)
	0.45	157 (3.2:1)	24.8 (20:1)	13.5 (37:1)
	0.29	164 (3.1:1)	33.8 (15:1)	18.4 (27:1)
	0.14	168 (3.0:1)	62.7 (8.1:1)	32.9 (15:1)
	0.05	177 (2.9:1)	135 (3.7:1)	79.2 (6.4:1)

Table 1: File size (in KB) and compression ratio with various ε_x (in *cm*) for the different compression techniques discussed in the paper. Ratios are given with respect to the uncompressed files which are 35.8 KB for the 148-frame running animation and 505 KB for the 2085-frame jumping animation. Applying *gzip* to the uncompressed files yield a compression ratio of 1.1:1.

standard wavelet compression and PCA compression. We do not include IK correction results since its goal is to increase visual realism by eliminating footskate, sometimes at the expense of an increase in positional distortion.

Numerical compression results for the running and jumping test sequences are presented in Table 1. For each technique we present the file size obtained after the optional *gzip* compression. On average, the file size before running *gzip* is 1.1-1.5 times larger. We observed that r, the ratio of the number of coefficients kept, is approximately 0.6 times the desired compression ratio. The rate-distortion graph averaged over the 15 test sequences is presented in Figure 2. Figure 3 compares the results obtained with standard compression to those obtained with optimized wavelet coefficient selection for the same compression ratio.

Through visual inspection of the sequences compressed with standard wavelet and optimized wavelet coefficient selection, we found that no noticeable artifacts were visible with $\varepsilon_x \leq 0.5$. On average, optimized wavelet coefficient selection can achieve this while maintaining a compression ratio of around 25:1. This ratio can reach up to 40:1 with longer sequences. Naturally, the target value of ε_x depends on the context. One might wish to achieve a lower value for very close viewpoints or, conversely, a higher value if the animation plays far in the background. For values of $\varepsilon_x > 0.5$,

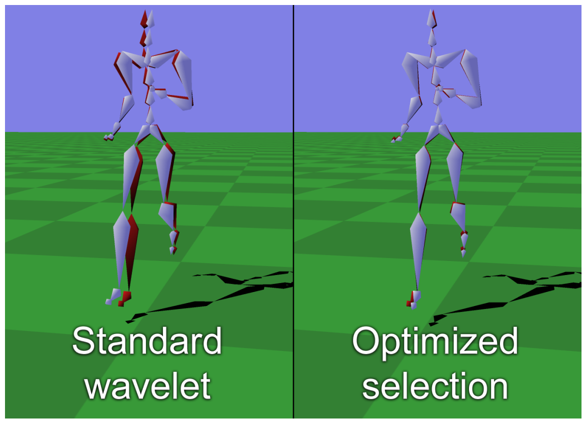

Figure 3: Comparison of standard wavelet compression and optimized wavelet coefficient selection with the same compression ratio. The original motion is shown in red in both images.

	Running (148 frames)		Jumping (2085 frames)	
Ratio	Time (*ms*)	Iterations	Time (*ms*)	Iterations
50:1	133	264	308	247
40:1	214	415	312	243
30:1	254	496	391	311
15:1	293	542	456	369
7:1	327	669	483	380

Table 2: Execution time per frame (in *ms*) and number of iterations for optimized coefficient selection.

the most noticeable artifact is footskate.

Execution time for the optimized wavelet coefficient selection depends on the length of the sequence and the desired compression ratio. Timing results and the number of iterations of the optimization algorithm for the test animations are shown in Table 2. When *gzip* is not used, decompression time is 30 μs/frame on average, which in principle makes it possible to interactively decompress more than 200 animations in real time.

6.2 Inverse Kinematics Correction

As mentioned earlier, when ε_x increases, the first visible artifact is caused by the foot sliding on the ground. This is the reason that motivated the introduction of a correction of the feet based on IK.

This method requires us to store 6 extra signals for the 3D position of the feet. Therefore, for the same file size, IK correction must keep less wavelet coefficients for the DOF than the other techniques. The outcome is that, for the same compression ratio, ε_x is usually larger with IK correction than with optimized wavelet coefficient selection. However, since IK correction resolves the most problematic artifacts related to sliding feet, we found by visual inspection that it could achieve higher compression ratios with no visible artifacts. For example, Figure 4 shows that, with the same compression ratio, IK correction can significantly reduce footskate while adding only minor distortions to the overall animation. This can also be observed in the accompanying video where the complete sequence is presented. The video can be found online at `www.iro.umontreal.ca/labs/infographie/papers`.

For the 15 test sequences, IK correction could reach compression ratios of 35:1 with no noticeable artifacts. We evaluated this by visually comparing the motions obtained with IK correction to the ones obtained with wavelet techniques when $\varepsilon_x = 0.5$. The com-

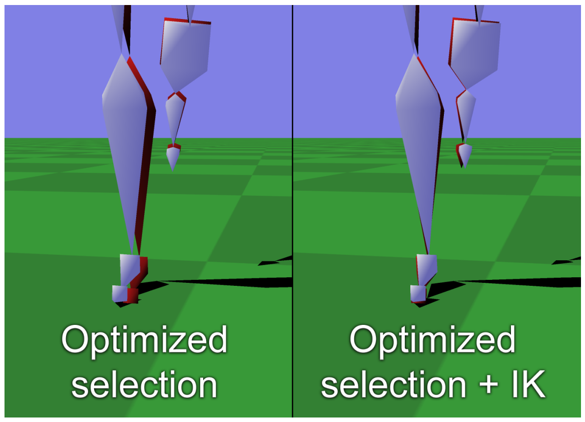

Figure 4: Comparison of optimized wavelet coefficient selection and IK correction with the same compression ratio. The original motion is shown in red in both images. In an animation, the error observed on the left image translates into a sliding foot.

pression parameters used to obtain the results shown in the accompanying video are $r = 30$ and $r_{\mathrm{IK}} = 16$. Figure 1 compares the results of standard wavelet to those with IK correction using an aggressive compression ratio.

The extra average compression time required to extract the 6 positional signals is 45 μs/frame. Total average decompression time is 300 μs/frame. This large number is mostly due to the simple and inefficient cyclic-coordinate descent IK solver we have implemented. This number could be significantly reduced by using a solver more finely tuned to this specific application.

6.3 Adaptive Wavelet Compression of Quaternions

We also experimented with the effect of compressing a quaternion representation. Quaternions do not exihibit the inherent DOF limitations of the skeleton; a 1-DOF knee still requires 4 values as a quaternion. The quaternion representation is thus larger than Euler angles to begin with. Wavelet compression reduces about equally, and thus we found the compressed quaternion representations to still be larger than the compressed Euler angles.

7 Conclusion

We have presented techniques to adapt standard truncated wavelet compression techniques to the nature of skeletal animation data. In particular, we proposed a way to optimize wavelet coefficient selection by searching through a reduced and tractable search space. This allows for important improvements on the visual quality of the compressed results while maintaining the fast decompression times associated with truncated wavelet compression. This technique was further extended to correct visually important artifacts such as footskate.

One of the problems we have encountered is the lack of a good metric to evaluate the perceptual distortion of the compressed animations. Such a metric would arguably be quite difficult to devise, given the fact that perceptual realism can depend on many factors external to the animation. Camera placement, viewer expertise, and surrounding environment are but a few of these factors. Still, we believe it would be possible to design a metric more predictive of perceived motion quality than the proposed positional distortion. This metric would not only give us a better way to evaluate final results, but could also be used during wavelet coefficient selection.

In the future, we wish to explore ways of exploiting large scale redundancies within a motion capture database, as well as level-of-detail streaming.

Acknowledgements

The data used in this project was obtained from `mocap.cs.cmu.edu`. The database was created with funding from NSF EIA-0196217. The authors acknowledge financial support from NSERC and FQRNT.

References

[1] O. Arikan. Compression of motion capture databases. In *SIGGRAPH'06*, pages 890–897, 2006.
[2] J. Assa, Y. Caspi, and D. Cohen-Or. Action synopsis: Pose selection and illustration. In *SIGGRAPH'05*, pages 667–676, 2005.
[3] J. Barbič, A. Safonova, J.-Y. Pan, C. Faloutsos, J.K. Hodgins, and N.S. Pollard. Segmenting motion capture data into distinct behaviors. In *Graphics Interface*, pages 185–194, 2004.
[4] H.M. Briceño, P.V. Sander, L. McMillan, S. Gortler, and H. Hoppe. Geometry videos: A new representation for 3D animations. In *Symp. Computer Animation*, pages 136–146, 2003.
[5] J. Chai and J.K. Hodgins. Performance animation from low-dimensional control signals. In *SIGGRAPH'05*, pages 686–696, 2005.
[6] CMU graphics lab motion capture database. mocap.cs.cmu.edu, April 2006.
[7] K. Forbes and E. Fiume. An efficient search algorithm for motion data using weighted PCA. In *Symp. Computer Animation*, pages 67–76, 2005.
[8] P. Glardon, R. Boulic, and D. Thalmann. A coherent locomotion engine extrapolating beyond experimental data. In *Computer Animation and Social Agents*, pages 73–84, 2004.
[9] K. Grochow, S.L. Martin, A. Hertzmann, and Z. Popović. Style-based inverse kinematics. In *SIGGRAPH'04*, pages 522–531, 2004.
[10] L. Ikemoto, O. Arikan, and D. Forsyth. Knowing when to put your foot down. In *Symp. Interactive 3D Graphics and Games*, pages 49–53, 2006.
[11] K. Kondo and K. Matsuda. Keyframes extraction method for motion capture data. *Journal for Geometry and Graphics*, 8(1):81–90, 2004.
[12] J. Lee and S.Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH'99*, pages 39–48, 1999.
[13] J. Lee and S.Y. Shin. Multiresolution motion analysis with applications. In *Intl Workshop on Human Modeling and Animation*, pages 131–143, 2000.
[14] I.S. Lim and D. Thalmann. Key-posture extraction out of human motion data by curve simplification. In *Intl Conf IEEE Engineering in Medicine and Biology Society*, volume 2, pages 1167–1169, 2001.
[15] G. Liu and L. McMillan. Segment-based human motion compression. In *Symp. Computer Animation*, pages 127–135, 2006.
[16] Z. Liu, S.J. Gortler, and M.F. Cohen. Hierarchical spacetime control. In *SIGGRAPH'94*, pages 35–42, 1994.
[17] K. Pullen and C. Bregler. Motion capture assisted animation: Texturing and synthesis. In *SIGGRAPH'02*, pages 501–508, 2002.
[18] A. Safonova, J.K. Hodgins, and N.S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *SIGGRAPH'04*, pages 514–521, 2004.
[19] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29(2):511–546, 1998.
[20] C. Welman. Inverse kinematics and geometric constraints for articulated figure manipulation. Master's thesis, Simon Fraser University, 1993.
[21] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24:530–536, 1978.