Compression of Motion Capture Databases

Okan Arikan

University of Texas, Austin

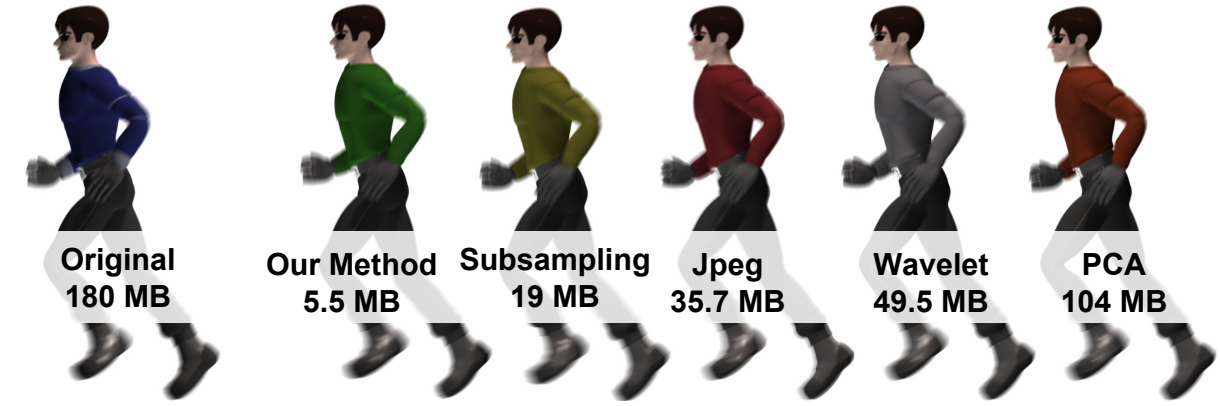

Figure 1: We present a compression method for large databases of motion capture data. The leftmost figure is uncompressed and is a frame from a database containing an hour and a half of motion data. The green character is the same frame in the compressed version, which takes only 5.5 MB to store. The other characters are from the same database compressed using subsampling, motion JPEG, wavelet and PCA compression.

Abstract

We present a lossy compression algorithm for large databases of motion capture data. We approximate short clips of motion using Bezier curves and clustered principal component analysis. This approximation has a smoothing effect on the motion. Contacts with the environment (such as foot strikes) have important detail that needs to be maintained. We compress these environmental contacts using a separate, JPEG like compression algorithm and ensure these contacts are maintained during decompression.

Our method can compress 6 hours 34 minutes of human motion capture from 1080 MB data into 35.5 MB with little visible degradation. Compression and decompression is fast: our research implementation can decompress at about 1.2 milliseconds/frame, 7 times faster than real-time (for 120 frames per second animation). Our method also yields smaller compressed representation for the same error or produces smaller error for the same compressed size.

Keywords: Motion Capture, Compression, Perception of Motion

1 Introduction

Data acquisition technologies such as motion capture can produce a large volume of animation data. If this data is to be used in a computer game, we would like to pack as much of it as possible in a limited amount of memory. Compression may also be important in a production environment for easy access to animation assets.

Established audio/video compression algorithms make use of perceptual models. Compression involves a study of perceptually unimportant features that we can omit, or equivalently, perceptually important features that we should maintain. Therefore motion compression is also important for understanding essential qualities of motion, especially human motion.

The biggest goal of compression is creating a compressed representation of motion that is perceptually as close to the original motion as possible. As we will explore later in this paper, a small numerical error does not necessarily correspond to a perceptually close motion. We would like compression and decompression to be as quick as possible. In practice motion capture databases can be very big. Therefore another goal for compression and decompression is to be able to process without holding the entire database in the memory, which may not be possible. Depending on the application we may want to "stream" the data so that the decompressor can decode incrementally. We may also want to be able to decode a piece of the database without having to decompress any other motion.

In this paper, we present a lossy compression algorithm for compressing large databases of motion capture data. Our method breaks the database into groups of clips that can be represented compactly in a collection of linear subspaces. An important quality of motion is contact with the environment, such as foot strikes. We would like to avoid making errors around parts of the body in contact with the environment, because it is perceptually jarring. Our method includes a different compression for these parts in contact with the environment.

Our algorithm produces compressed representations that are very small and also maintain visual fidelity. Our method is fast: we can compress and decompress faster than realtime, without holding the entire database in the memory. We allow decompression of clips independently, without needing to decompress any other clip.

2 Related Work

Audio and video compression is an important problem and many good solutions have been proposed. 3D animation compression methods use some machinery and extrapolate some of the ideas from audio/video compression. [Salomon 2000] provides a very good overview of the mainstream compression algorithms.

Previous work on animation compression mainly focuses on compressing animated meshes. Literature on compression of static meshes is rich [Rossignac 1999; Karni and Gotsman 2000]. However compressing animation frames individually is suboptimal. Ibarria and Rossignac proposed a predictor/corrector method [2003] for taking inter-frame coherence into account. Guskov and Andrei offer another way to exploit spatial coherence using wavelets [2004]. They also encode the differential wavelet coefficients to compress an animation sequence. PCA can be used to compress shapes or animations [Alexa and Muller 2000; Sloan et al. 2001]. One can also take coherence into account by finding portions of the mesh that move rigidly and only encoding the rigid transformation and residuals [Lengyel 1999; Gupta et al. 2002]. In the animation context, identifying rigidly moving portions of the mesh is also valuable for efficient display [Wang and Phillips 2002; Mohr and Gleicher 2003; James and Twigg 2005].

Clustered PCA is an effective way of compressing high dimensional data. Sattler et al. introduced a clustered PCA based approach for compressing mesh animations [Sattler et al. 2005]. They cluster mesh animations to identify linearly related vertex trajectories. Our method uses clustered PCA for identifying linearly related snippets of motion clips. Sloan et al. presented another successful usage of clustered PCA for compressing high dimensional radiance transfer functions on meshes [2003].

Most of the test examples in mesh animation are generated using a skeleton based character animation system. The position of a vertex in an animation is a function of the skeletal degrees of freedom (a skinning function). If the character is displayed using *linear blend skinning*, we expect a high degree of coherence in vertex positions, because nearby vertices tend to have the same blend weights. Therefore, finding coherence in a mesh animation is easier than finding coherence in skeletal animation degrees of freedom. Nonetheless, previous research on biomechanics suggest that human (or animal in general) joints move coherently for some types of motion [Alexander 1991]. As such, our paper focuses on exploiting this coherency to compress skeletal animations, rather than compressing the generated mesh animations. Compressing mesh animations that do not have an underlying skeleton system are better suited for the mesh animation compression methods mentioned above.

The coherency between degrees of freedom in a motion implies a lower dimensional space of motion. Previous work on motion synthesis and texturing implicitly make use of this lower dimensionality [Rose et al. 1998; Pullen and Bregler 2002; Safonova et al. 2004; Grochow et al. 2004]. For example, the switching linear dynamic systems of [Pavlovic et al. 2000] and [Li et al. 2002] can be interpreted as lower dimensional representations. Chai and Hodgins were successful in producing good motions by searching a database for clips that meet sparse constraints [2005]. Our paper attempts to utilize this correlation for producing a smaller representation of a motion database. Identifying lower dimensional representation of motion is also useful for activity recognition and robot motion planning [Vecchio et al. 2003].

It is well known that editing operations on motion may introduce visually disturbing footskate. Lossy compression can also create footskate. There are methods that address this issue [Kovar et al. 2002b] or inverse kinematics. Our method will address this issue by compressing environmental contacts using a separate mechanism.

3 Overview

Typical animal motion has important properties that we can use for compression. **Degrees of freedom are correlated with each other**. Pullen and Bregler used this observation for motion synthesis/texturing [2002] and Jenkins and Mataric [2003] used it for identifying behavior primitives. **Degrees of freedom have temporal coherence**. This property makes motion synthesis an interesting research problem, because there are physical limits to how different two subsequent frames in an animation sequence can be [Arikan and Forsyth 2002; Kovar et al. 2002a; Lee et al. 2002]. **In a large database, there will be many different copies of similar looking motions**. This observation makes data driven motion synthesis in motion databases possible. Recent research also suggests that similar motions can be blended to obtain other physically plausible motions [Safonova and Hodgins 2005].

Our compression technique works on short clips of motion sequences. We represent these clips as cubic Bezier curves and perform Clustered Principal Component Analysis (CPCA) to reduce their dimensionality. This technique utilizes temporal coherence (fitting Bezier curves) and correlation between degrees of freedom (CPCA). High level of compression comes at the expense of smoothing high frequency detail in the motion.

Unfortunately, human motion contains valuable high frequency detail. This is why people often capture motions at high sampling rates (commonly 120-240 Hz). The high frequency detail is usually due to environmental contacts such as foot strikes. Ground reaction force is quite significant (more than the weight of the entire body) and applies over a very short amount of time in a typical gait. Therefore, it fundamentally affects what motion looks like.

A part of the body that is in contact with the environment should not move (we exclude sliding contacts). In addition to maintaining important high frequency content in motion, we must also enforce this constraint to maintain visual quality. We introduce a way of encoding the environmental contacts (for example the feet) and enforce the contact detail during the decompression.

4 Compression

We assume the motion database is a single (possibly long) motion. We will represent this motion as vector valued function $M(t)$, where t refers to the frame number (or equivalently time). Every frame $M(t)$ contains the degrees of freedom that control the character. We assume the motion is sampled at regular intervals. We also assume the number of degrees of freedom does not change frame to frame. For motion capture, these degrees of freedom are typically the character's global position/orientation and a set of joint angles that relate each bone to its parent.

For compression, we split the motion database into clips of k subsequent frames. For example, the first k frames is the first clip and the next k subsequent frames is the second. The clip size (k) is a **compression parameter** that affects the compression. We will discuss compression parameters in Section 4.5.

We rotate and translate each clip so that, at the first frame, the character is located at the origin in a standard orientation. Each clip stores the absolute position/orientation of the character before this transformation (6 numbers). During decompression we undo this transformation and put every clip back in its absolute position/orientation.

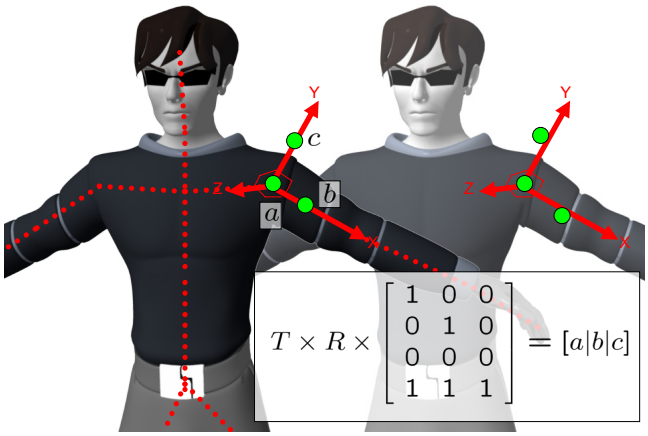

Figure 2: Joint angle space is not very suitable for compression. We convert rigid coordinates frames for each bone into a set of virtual markers placed at fixed positions ($[1\,0\,0\,1]^t, [0\,1\,0\,1]^t, [0\,0\,0\,1]^t$ in this figure). We can convert back to the rigid coordinate frame and solve for the translation T and rotation R matrices using least squares.

We first convert the degrees of freedom into a positional representation (Section 4.1). This new representation is lossless and provides a more linear space to represent these clips. We then fit cubic Bezier curves to this positional representation (Section 4.2). This allows us to embed clips in a compact space without the clutter of high frequencies. We then fit locally linear models to the Bezier control points and reduce the clips' dimensionality (Section 4.3).

We encode important contacts with the environment using a separate, JPEG-like technique (Section 4.4).

4.1 Motion Representation

No matter how we represent them, orientations have an intrinsic nonlinearity: a halfway orientation between a and b is not necessarily $(a+b)/2$. Therefore finding a linear subspace in joint angle representation is difficult. Another problem comes from the hierarchical representation of joints: angles closer to the root effect more bones and they are more important. Unfortunately finding a weighting scheme for hierarchical joint angles seems difficult.

Joint positions behave more linearly. For example, if the position of the feet the are same in two frames, any linear blend between joint positions will keep the feet at the same location. If we blend the joint angles, the feet may move even though the 3D positions of the feet in the two frames were the same. This property is useful for maintaining environmental contacts. Therefore, we work with 3D trajectories of joints rather than joint angles.

If we blend joint positions, the distance between two endpoints of a bone may change. One can fit a rigid skeleton and recover the original joint angles using Inverse Kinematics (IK). But this is a nonlinear minimization that can fall into different local minima between frames. Fortunately we can compute the joint angles directly if we compute the global position of 3 different and known points in the local coordinate of each bone (see figure 2). This allows us to work with *virtual marker* positions (3 for each bone) rather than joint angles.

This motion representation requires 3 times more storage (3 virtual marker positions for each frame as opposed to 3 joint angles). According to our experience, this over-complete representation is more perceptually uniform and is more compressible than joint angles.

4.2 Smooth Approximation

Each clip contains the 3D trajectory of virtual markers over k frames. We fit a 3D cubic Bezier curve (using least squares) and represent each of these trajectories with their control points. For every virtual marker we store only 4×3 numbers (4 control points), rather than $3 \times k$.

4.3 Clustering and Projection

We represent a virtual marker's trajectory using 12 numbers (the x,y,z coordinates of 4 Bezier control points) and there are 3 virtual markers for every bone. Therefore each clip is a point in a $d = 12 \times 3 \times$ *number of bones* dimensional space. We will call this vector x_i for clip i. We could perform linear dimensionality reduction using PCA, but it may be difficult to find a compact set of basis directions that span a diversified motion database.

To remedy this situation, we group similar looking clips into distinct groups by clustering. We used spectral clustering using Nystrom approximation [Shi and Malik 2000; Fowlkes et al. 2004]. This method provides a computationally efficient way of clustering in our high dimensional space. Within each cluster, we create a new orthogonal coordinate system using PCA. For each cluster c, we compute the cluster mean $m_c \in \Re^d$ and a basis matrix $P_c \in \Re^{d \times d}$, where the rows of P_c are the eigen vectors of the covariance matrix for the cluster. A clip x_i can be transformed into the local coordinate space of cluster c by $\widetilde{x_i} = P_c(x_i - m_c)$ and similarly, can be transformed back using $x_i = P_c^{\mathsf{T}} \widetilde{x_i} + m_c$.

The rows of P are sorted in descending order of the corresponding eigen values so that row 1 spans the direction of most variance in the cluster, and row d spans the direction of least variance. Due to this property, the cluster coordinates of a clip i ($\widetilde{x_i}$) tend to decrease rapidly. Given a user specified error threshold τ, we can find the number of rows of P_c that would be required to approximate a clip below the L_2 error τ. The error threshold τ and the number of clusters to use are **compression parameters**.

The compressed version of a clip records the cluster index (integer), the number of the rows of P_c that it is using (integer) and the coordinates in the coordinate system of the cluster (only those coordinates on the used rows of P_c). The compressed version of the database also includes the m_c vectors and the P_c matrices that define the local coordinate systems for each cluster. Practically, only the first few rows of P_c are usually used and we store only the used rows.

In big motion databases, some motions are very common. These common clips belong to the same cluster and they have similar projected coordinates. We take advantage of this observation by quantizing the projected coordinates into 16 bits. The resulting integers have low entropy due to the dense clustering of coordinates. We compress the integer coordinates using entropy encoding (we used Huffman codes).

The quantization we mentioned above may cause errors. We use 16 bits to represent the projected coefficients which can introduce a maximum error of $(max(\widetilde{x_i}) - min(\widetilde{x_i}))/2^{16}$. If we assume we are compressing a human motion capture data, $(max(\widetilde{x_i}) - min(\widetilde{x_i}))$ is about 3 meters (the maximum conceivable L_1 distance between any two points on the body). Therefore the quantization error is less than 0.05 millimeters, which we find acceptable. Thus we do not include the number of quantization bits as a compression parameter.

4.4 Environmental Contacts

Feet typically contain extra detail due to the contact with the floor. For example a foot should not move when it is in contact. This is a very difficult constraint to enforce when working with joint angles. Because we work with virtual markers that move rigidly with the

underlying bone, we can compress the feet differently and make sure the floor contacts are maintained during decompression.

After we transform the clips so that they start at the origin in a standard orientation, we represent the x, y, z coordinates of the virtual markers on the feet as separate 1D signals and transform them into a frequency space using Discrete Cosine Transform (DCT). We then quantize the resulting coefficients into a finite number of bits. DCT has excellent energy compaction properties: most of the DCT coefficients (especially the ones in the high frequencies) will quantize into the same bin. Therefore the quantized stream tends to have low entropy, which we exploit using entropy encoding (we used Huffman codes).

For a perfectly stationery foot throughout a clip, only the DC component of the frequency transform is non-zero and all other DCT coefficients are zero. This way of compressing the feet is nicely suited for joints that come into contact with the environment. Notice also that we do not remove high frequencies; we simply compress them. Therefore we can still capture the impact of the feet with the environment.

This compression scheme is applicable to other parts of the body that come into contact with the environment. Our compression technique needs to know the body part that is interacting with the environment (and possibly the time). Automatic detection of these contacts is possible using methods like those of [Ikemoto et al. 2005]. Because feet-ground interaction is very common and important, our implementation compresses the entire trajectory of both feet.

4.5 Compression Parameters

Our method comes with parameters that control the compression accuracy vs. compressed size (just like any audio / video codec). Our method has 3 main parameters:

1. k: the number of frames in a clip. The bigger this number is, the smoother the reconstructed signal will be. If k is too small, then the compression will not be able to take full advantage of the temporal coherence. If it is too big, the correlation between joints will not be linear and CPCA will perform poorly. Optimal numbers we found are 16-32 frames (130 - 270 milliseconds).

2. τ: the upper bound on the reconstruction error of CPCA. The smaller this number is, the more coefficients will be needed for each clip. We define this number relative to the standard deviation of the clips. We obtained good results with $\tau = 0.01 - 0.1$.

3. Number of clusters for CPCA: A diversified database requires more clusters for optimal representation. However, since the compressed model also contains data for each cluster (m_c, P_c), a large number of clusters may create extra overhead. For the datasets we tried, 1-20 clusters provided accurate representation

To generate our results as well as the baseline results, we sampled these parameters randomly (within reasonable bounds) and compared the compressed motions to the originals.

5 Decompression

The compressed payload contains m_c, the needed rows of P_c for each cluster, and for each clip:

1. A cluster index

2. Entropy encoded coordinates in the cluster

3. Entropy encoded DCT of the feet trajectory

4. The absolute position/orientation in the world at the first frame

We can decompress each clip individually. We first perform the entropy decoding and undo the quantization for the cluster coordinates ($\widetilde{x_i}$) and for virtual markers on the feet. We obtain the Bezier control points for the virtual markers by $x_i = {P_c}^{\mathsf{T}} \widetilde{x_i} + m_c$. We now have 12 control points for every virtual marker for the clip. We resample these Bezier curves to reach the 3D position for each frame within the clip. We decompress the virtual markers on the feet by performing inverse DCT. These trajectories we decompressed for feet overwrite the virtual markers we decoded using the Bezier curves.

We fit a rigid body coordinate frame to the 3 virtual markers belonging to the same bone for each frame. We can now use these coordinate frames directly for displaying a skinned character. However, the original data was in joint angles, and we would like to go back to the same representation. The relative rigid body transformation between adjacent bones i and j is $T_{ij} = T_i^{-1} T_j$. We then convert the rotation part of T_{ij} to joint angles.

After the conversion to joint angles, feet may not reach the position that we decompressed because of the loss in compression. As a last step, for each frame, we perform IK so that foot positions that we decoded are satisfied. We used the IK method described in [Tolani et al. 2000]. In practice the required adjustment is small. Therefore we believe a Jacobian based IK would also be efficient.

Once the joint angles are recovered for each clip, we translate/rotate the clip to position it in the global coordinate frame.

Due to the lossy compression, there may be discontinuities between adjacent clips in time. We get rid of these usually small discontinuities using the method outlined in Appendix A.

Other important features of our method are:

1. **Block decompression**: Every k frame clip is compressed as a whole. Therefore, to decompress an individual frame, we need to decompress the entire clip that contains it. This is not a big problem in practice, because animation applications tend to have coherence in the frames they require.

2. **Compression/Decompression time**: Our method is fast for compression and decompression. Our Matlab/C++ implementation can compress at 1 milliseconds/frame and decompress at 1.2 milliseconds per frame on average (7 times faster than real-time).

3. **Offline compression**: It is not always possible to hold the entire motion database in memory at the same time. Thefore it is important for the compressor to be able to process small chunks at a time. In order to have this feature, we perform the clustering in the CPCA on a randomly selected subset of the entire database (we pick a random 10,000 clips). The rest of the clips are processed independently one at a time, and hence the entire database never needs to be in the memory all at once.

4. **Random access**: Our decompression algorithm does not operate incrementally. This allows us to decode any clip in the database without decompressiong others.

5. **Incremental compression**: Once the CPCA is performed and local linear subspaces are identified, we process clips individually. This allows us to append clips to an already compressed database. If the newly added clips change the statistical distribution of the clips, the user can run the clustered PCA to take this change into account.

6 Sources of Error

There are 3 lossy steps in our compression method:

1. Approximating each clip with smooth cubic Bezier curves smooths large accelerations. Fortunately, such high accelerations are usually due to environmental contacts and we compress these high frequency degrees of freedom using a different method.

2. Projecting each clip onto a linear subspace introduces error. Here we take advantage of the correlation between joints and temporal coherence. In order to further increase the correlation, we perform clustering and compute a linear subspace for each cluster separately. Existing research on motion blending explicitly uses this observation for generating transitions. Furthermore, the error in this step is controlled by a user threshold. It is usually possible to find a compact representation that meets even the most conservative error bounds.

3. The quantization step of the frequency compression of the feet introduces error. This method is very similar to JPEG compression of images [JPEG 2000]. JPEG performs poorly around sharp changes (C0 discontinuities) because of the information lost in the high frequency spectrum. Fortunately, the virtual markers on the feet always move with C0 continuity.

7 Baseline Methods

In this section, we describe reasonable baseline methods for motion compression. We ran the following baseline algorithms on the joint angle representation of the motion. We also tested running them on the virtual marker trajectories, but the redundancy in that representation yielded poor compression ratios, because most of these baseline methods compress degrees of freedom individually.

7.1 Motion JPEG

Encouraged by Section 4.4, we compressed the entire body using a JPEG like method. Each degree of freedom is a 1D signal in time. We compress these signals separately. We first split a degree of freedom into k frame clips (analogous to $8x8$ pixel blocks in JPEG). We transform each clip into a frequency space using DCT. We quantize the DCT coefficients, which gives us a low entropy integer signal. We compress these sequence of integers using entropy coding (we used Huffman codes again).

The decompression first reconstructs the quantized DCT coefficients with entropy decoding. We then perform inverse DCT to obtain the motion signal. Due to the quantization step, this is a **lossy** compression method.

The compression parameters for this method are the clip size (k) and the number of bits to use for quantization. Large block sizes make entropy encoding more efficient, but also lead to wider energy spectrum. Optimal block sizes range from 256 - 1024 frames. The number of quantization bits is related to the range of input values. For joint angles bounded between 0 and 2π, 8-12 bits provide reasonable reconstruction.

7.2 Wavelet Compression

Wavelet compression is the same as Motion JPEG except that it uses the Wavelet transform instead of DCT. In this work, we used the Haar wavelet basis. Haar wavelets provide an orthonormal and local basis. Therefore they are better at capturing local detail. The compression and decompression process is exactly the same as in section 7.1 with the only difference in using the Haar transform instead of DCT (and the inverse Haar transform instead of inverse DCT).

For image compression, the strength of Wavelet compression lies in the fact that the basis functions are local (DCT bases are not) and therefore are better at capturing localized image detail. It is our experience that larger block sizes for wavelet compression yields better compressed quality/compressed size for motion.

7.3 Per Frame PCA

Another way to compress a collection of frames is to perform linear dimensionality reduction. Let $M(t)$ to be a column vector. Using PCA, we can find a projection $N(t) = P(M(t) - m)$. m is the mean of all frames and P is the matrix whose rows are the eigen vectors of the covariance of the frames in the database. The rows of P corresponding to the small eigen vectors can be omitted to make the dimensionality of N less than the original dimensionality of M. The original frames can be approximated by $M(t) \approx P^{\mathsf{T}} N(t) + m$.

The only parameter for this compression scheme is the number of rows of P to keep. More rows allow better reconstruction at the expense of worse compression. We only perform PCA on the joint angle degrees of freedom. The global position/orientation of the character can be compressed using wavelet or motion JPEG.

7.4 Subsampling

A straightforward way of compressing a time varying signal is subsampling the database. For example, instead of storing every frame, we can store every other frame and reach a representation that occupies half the space. The ignored frames can be approximated by interpolation (linear or higher order). The only parameter for this method is the subsampling amount.

8 Evaluation Metrics

In this paper, we are primarily interested in obtaining a compressed representation of motion that is as close to the original as possible. One may also be interested in having a compressed representation looking plausible without necessarily looking like the original. If the latter is our objective, then we must evaluate the naturalness of the compressed motions. Promising results have been presented by [Ikemoto and Forsyth 2004; Ren et al. 2005; Arikan et al. 2005], but robust and automatic quantification of all motion is still difficult. Furthermore automatic quantification is difficult to generalize to all types of animals.

To evaluate our results, we need to define error metrics for quantifying "closeness". Let us denote the compressed version of a motion with $M_c(t)$ (we will refer to the motions that has been decompressed from a compressed representation as *compressed*). The Root Mean Squared error in the degrees of freedom is defined as:

$$RMS = \sqrt{\frac{1}{n}\sum_{t=1}^{n} |M(t) - M_c(t)|^2} \qquad (1)$$

where n is the number of frames in the database. Unfortunately, if the degrees of freedom are joint angles, then RMS error is a poor indicator of the closeness between motions.

Motion is usually displayed on a 3D character. We can define closeness to be the distance between the skin vertices of the compressed motion versus the original. This definition is more informative. We therefore replace $M(t)$ (and $M_c(t)$) in equation 1 with the coordinates of the skin vertices at time t. This will be the definition of RMS we will use.

We compare our results to baseline methods in Figure 3. As the figure demonstrates, for comparable RMS error, our method creates a compressed representation that is smaller than other methods. Although we could discuss this figure in further detail, it actually provides very little information about the perceptual closeness of the compressed motion to the original.

Audio and video compression methods build on extensive research on models of audio/visual perception. Therefore these methods can remove perceptually insignificant detail from the signal and achieve high compression results. Case studies on perception of animation has been presented in [O'Sullivan et al. 2003; Reitsma and Pollard 2003]. Unfortunately, perceptual models of general human motion is still not a mature research area. In our experience in dealing with motion capture data, we made the following empirical observations, some of which will be obvious to the reader.

1. 3D positions (virtual markers) provides a more perceptually uniform space than joint angles. Sometimes a tiny error in an angle creates a big perceptual change. This is an unlikely scenario when dealing with absolute positions. For example in motion JPEG and wavelet compression, some small errors in root position/orientation causes large visual artifacts.

2. In a typical motion, some parts of the body contact the environment (such as feet). People tend to be more sensitive to the errors in the parts of the body in contact. For example in subsampling, the compressed motion may appear to be swimming when upsampled. This effect is visually very jarring.

3. People tend to be more sensitive to high frequency error than errors in low frequency. This is probably due to the fact that it takes more power for a person to introduce high frequencies to his/her motions. For example PCA, motion JPEG and wavelet compression tends to produce jittery results for aggressive compression parameters.

4. It is easier to spot errors when the compressed motion is displayed spatially close to the original motion.

5. Perception of "closeness" also depends on the viewing angle of the motion.

6. An compressed motion may look close to the original, but the same error on a different motion may be perceptually jarring.

The statements above are merely our empirical observations and they do not provide a definitive guideline for user studies. Here lies the difficulty in evaluating the results of a compression method for motion.

In the attached video we include examples that demonstrate the statements we made above. We also provide side by side comparison of original motion versus compressed version of the same motion (displayed as close as possible without interfering with each other). We invite the reader to evaluate our results in our video. For example, in Figure 3, subsampling seems to perform nicely, producing low RMS error. We ask the reader to compare the perceptual quality of subsampled motions against our method.

9 Results

Widely agreed upon sets of rules for the perceptual quality of human motion have not yet been established. It is difficult to design experiments/user studies for this reason. Even for audio/video compression where perceptual rules exist, a common practice is looking at the compressed sequence and tweaking the compression parameters until we obtain desirable results.

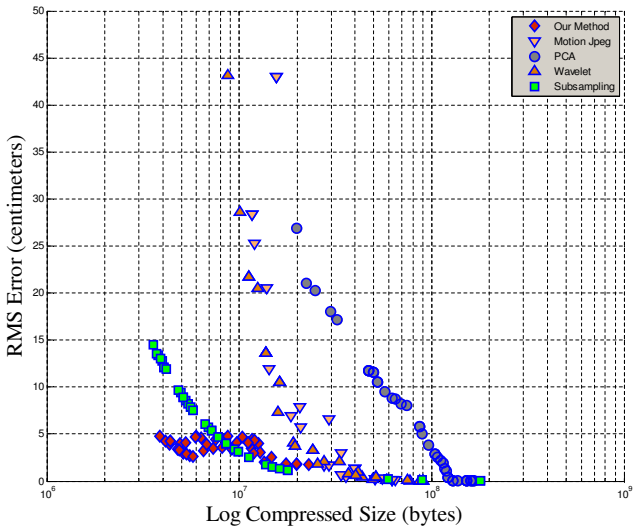

Figure 3: This figure plots the log of the compressed size (in bytes) against the RMS error (in centimeters). The uncompressed size of this database is 180 MB (1:30 hours long). As we would expect, as the compressed size goes down, the RMS error increases. Our method performs better than other baseline algorithms and produces a compressed representation that is smaller for the same RMS error or produces a smaller RMS error for the same compressed size.

We demonstrate our results on two datasets. Dataset 1 consists of different kinds of locomotion (standing, walking, running, skipping, being pushed etc.). It contains 620K frames sampled at 120Hz (1:30 hours long) and amounts to 180 MB of storage in the uncompressed form (32 bit IEEE floating point for each degree of freedom for each frame). All the motions in this dataset belong to the same skeleton.

Dataset 2 is the motion capture collection maintained at Carnegie Mellon University (as of 12/22/2005) and is greatly diversified. It contains 2.9M frames sampled at 120Hz (6:30 hours long) and uses 1085 MB of storage in the uncompressed form. This is a challenging dataset, because it contains noisy and corrupted motions. It also features motions that are recorded from different people with different sizes (different skeletons). Fortunately the skeleton topology is the same for all subjects. Therefore, we can find the corresponding bones in different sequences and apply our algorithm as if it were recorded for the same skeleton.

We will present motions in database 1 on a skinned character. Due to the different skeletons involved in database 2, we will display the motions in this database on a stick character which is automatically generated from the skeleton description.

Figure 3 shows the RMS error in the skin positions of the compressed motion (y axis) against the size of the compressed database (x axis). Notice that the x axis is in the log scale, so the points that seem close in the x axis may have very different sizes. This figure has been computed for dataset 1. It shows that our method produced smaller RMS error for the same size (or smaller compressed size for the same error). However the RMS error is not a good predictor of visual quality.

In the attached video, we provide side-by-side comparison of compressed motions with the same compression ratio (for our method and baseline algorithms). For example, in Figure 3, subsampling seems to be the closest competitor to our method. For the same compression ratio, subsampling produces motions that look like they are swimming while our method maintains the important

	Us	Sub	JPEG	Wavelet	PCA	ZIP
CMU	35.4	92.1	237.2	184.7	520.4	788
1085MB	30:1	12:1	5:1	6:1	2:1	1.4:1
Sony	5.5	17.9	35.71	49.48	104.23	165
180MB	32:1	10:1	5:1	4:1	1.7:1	1.1:1

Table 1: This table provides a comparison of the compression methods for the same amount of visual quality. For each method and dataset, we record the size (in megabytes) of the compressed representation for acceptable visual quality. The gray numbers represent the compression ratio. The last column corresponds to lossless LZW compression.

contact detail with the floor. For the same RMS error, motion JPEG and wavelet compression produces motions that are sometimes jittery while motions compressed using our method are liquid. PCA compression can also create jittery motions because some low variance degrees of freedom can create large changes in the pose.

In the extreme compression case, our method starts producing motions that seem to be swimming (similar to subsampling) for the upper body. The feet maintain more detail since we compress them separately. Because we use IK to enforce positions of the feet, the decompression may produce results where the knee bends incorrectly. A more sophisticated IK mechanism may remedy this situation. We demonstrate this extreme compression case in the video.

A common artifact of IK is known as the "knee pop". It refers to the knee snapping to and from the fully extended configuration. This artifact is due to the rigid skeletal structure that we use for character animation. In an extreme compression case, our method can also produce knee pops if the loss in the motion is such that the distance between the hip and the feet is greater than the length of the leg. The common practical solution to this problem is allowing small changes to bone lengths [Kovar et al. 2002b]. This solution is effective (perceptual effects of length changes has been studied in [Harrison et al. 2004]) and is used by commercial programs. However, we enforced rigidity of the bones in order to go back to the same representation of motion.

The major limitation of our method is that we need to know the contacts. Feet are easy because they are usually in contact with the environment and hence do not need to be annotated. We are exploring the automatic detection of environmental contacts using the method of [Ikemoto et al. 2005]. Once the contacts are known, they can be compressed like the feet.

Table 1 shows a comparison of our algorithm against the baseline methods for the same level of perceptual quality. This is an informal table that compares the compression rates (the original size : the compressed size) for the same level of visual quality. We emphasize that we used our own subjective judgment and encourage the reader to verify our results presented in the attached video.

Our compression gets better as the size of the database increases. For example, when we compress only $1/8^{th}$ of the CMU database, each frame takes about 27 bytes. If we compress half, we get down to 14 bytes/frame for the same visual quality. The entire database takes about 12.4 bytes/frame. This is due to the increased number of common poses that can be represented linearly. Unfortunately, each compressed clip also stores a compressed version of the feet. Therefore there is a linear trend between the size of the compressed database and the number of frames it contains.

In practice, we tested this algorithm on human motion capture databases, because they are more common. We expect the same framework to work for general animal motion as well.

Our compression method is effective: We are able to compress database 1 from 190MB to 5.5MB (35:1 compression ratio) and database 2 from 1080MB to 35 MB (31:1 compression ratio) with very little visual degradation. This means we can store 6.7 hours of high quality motion capture data in only 35 MB of memory. We can decompress at 1.2 millisecond / frame on a P4 3.4 with 3 GB of RAM. This means our decompression is about 7 times faster than real-time. Therefore our method is practical using today's hardware.

10 Acknowledgments

We would like to thank David Forsyth, Leslie Ikemoto and our anonymous reviewers for their valuable input. This research was supported by generous donations from Intel, Pixar and Autodesk. Portion of the data used in this project was obtained from mocap.cs.cmu.edu. The database was created with funding from NSF EIA-0196217. Rest of the motion capture data we used was generously donated by Sony Computer Entertainment America.

A C1 Continuous Merge

Due to loss in compression, there may be discontinuities in a motion signal where subsequent clips join. Let us assume a clip spans frames i thru j and define $\Delta M(i) = M(i) - M(i-1)$. We solve for a new clip $F(t)$, that is C1 continuous at the beginning and at the end of the clip and follows the derivative of the clip by solving:

$$\begin{aligned} F(i) &= [M(i) + 2\times M(i-1) - M(i-2)]/2 \\ \Delta F(i+1) &= [\Delta M(i+1) + \Delta M(i-1)]/2 \\ \Delta F(i+2) &= \Delta M(i+2) \\ &\cdots \\ \Delta F(j-1) &= \Delta M(j-1) \\ \Delta F(j) &= [\Delta M(j) + \Delta M(j+2)]/2 \\ F(j) &= [M(j) + 2\times M(j+1) - M(j+2)]/2 \end{aligned}$$

The first two equations enforce C1 continuity at the beginning of the clip and the last two equations enforce C1 continuity at the end of the clip (the red dots and black arrows in figure 4). This sparse, linear and banded system of equations can be solved efficiently to obtain the continuous signal F for each clip.

B Implementation Details

During compression, the user may want to compress different degrees of freedom differently. For example, we compressed feet differently than other degrees of freedom in our algorithm. Another way of doing this would be to have a bit rate allocation where joint angles higher up in the hierarchy get more bits than others. This can be accomplished in most of the methods mentioned in this paper. For example, we may allocate quantization bits proportional to the requested bit rates in JPEG/Wavelet compression. However, for character motion, finding a good bit rate distribution between degrees of freedom is not easy.

For JPEG/wavelet compression, joint angles must be converted to a smooth signal in time. We accomplish this by adding (or subtracting) 2π if there is a discontinuity bigger than π between subsequent frames.

References

ALEXA, M., AND MULLER, W. 2000. Representing animations by principal components. In *Eurographics Computer Animation and Simulation*, vol. 19, 411–418.

ALEXANDER, R. M. 1991. Optimum timing of muscle activation for simple models of throwing. *J. Theor. Biol. 150*, 349–372.

ARIKAN, O., AND FORSYTH, D. 2002. Interactive motion generation from examples. In *Proceedings of SIGGRAPH 2002*, 483–490.

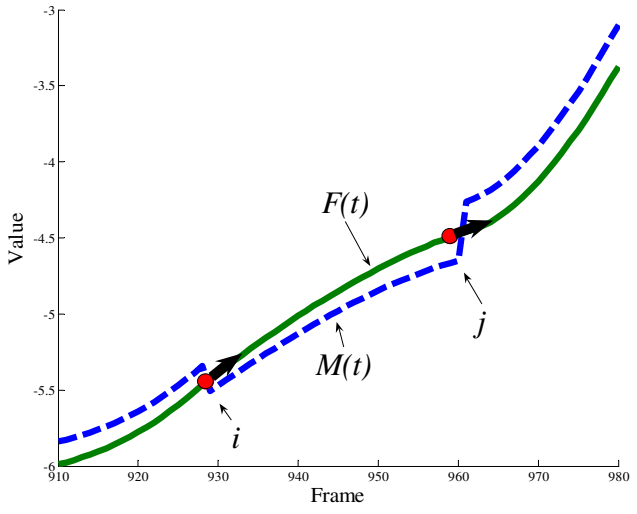

Figure 4: When we put the clips together, there may be small discontinuities. We get rid of them by solving for a new clip $F(t)$ that is C1 continuous at the beginning and the end (with the previous and succeeding clips), and also follows the derivative of $M(t)$ within the clip.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Pushing people around. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer animation*, ACM Press, 59–66.

CHAI, J., AND HODGINS, J. K. 2005. Performance animation from low-dimensional control signals. *Proceedings of SIGGRAPH 2005 24*, 3, 686–696.

FOWLKES, C., BELONGIE, S., CHUNG, F., AND MALIK, J. 2004. Spectral grouping using the nystrom method. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, 214–225.

GROCHOW, K., MARTIN, S. L., HERTZMANN, A., AND POPOVIC;, Z. 2004. Style-based inverse kinematics. *Proceedings of SIGGRAPH 2005 23*, 3, 522–531.

GUPTA, S., SENGUPTA, K., AND KASSIM, A. A. 2002. Compression of dynamic 3d geometry data using iterative closest point algorithm. *Comput. Vis. Image Underst. 87*, 1-3, 116–130.

GUSKOV, I., AND KHODAKOVSKY, A. 2004. Wavelet compression of parametrically coherent mesh sequences. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 183–192.

HARRISON, J., RENSINK, R. A., AND VAN DE PANNE, M. 2004. Obscuring length changes during animated motion. *Proceedings of SIGGRAPH 2004 23*, 3, 569–573.

IBARRIA, L., AND ROSSIGNAC, J. 2003. Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 126–135.

IKEMOTO, L., AND FORSYTH, D. A. 2004. Enriching a motion collection by transplanting limbs. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 99–108.

IKEMOTO, L., ARIKAN, O., AND FORSYTH, D. 2005. Knowing when to put your foot down. In *I3D: Symposium on Interactive 3D Graphics and Games*, 49–53.

JAMES, D. L., AND TWIGG, C. D. 2005. Skinning mesh animations. *Proceedings of SIGGRAPH 2005 24*, 3, 399–407.

JENKINS, O. C., AND MATARIC, M. J. 2003. Automated derivation of behavior vocabularies for autonomous humanoid motion. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, 225–232.

JPEG, 2000. Jpeg 2000 - http://www.jpeg.org/jpeg2000/index.html.

KARNI, Z., AND GOTSMAN, C. 2000. Spectral compression of mesh geometry. In *Proceedings of SIGGRAPH 2000*, 279–286.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *Proceedings of SIGGRAPH 2002*, 473–482.

KOVAR, L., GLEICHER, M., AND SCHREINER, J. 2002. Footstake cleanup for motion capture editing. In *ACM SIGGRAPH Symposium on Computer Animation 2002*, 97–104.

LEE, J., CHAI, J., REITSMA, P., HODGINS, J., AND POLLARD, N. 2002. Interactive control of avatars animated with human motion data. In *Proceedings of SIGGRAPH 2002*, 491–500.

LENGYEL, J. E. 1999. Compression of time-dependent geometry. In *SI3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, 89–95.

LI, Y., WANG, T., AND SHUM, H. Y. 2002. Motion texture: A two-level statistical model for character motion synthesis. In *Proceedings of SIGGRAPH 2002*, 465–472.

MOHR, A., AND GLEICHER, M. 2003. Building efficient, accurate character skins from examples. *Proceedings of SIGGRAPH 2003 22*, 3, 562–568.

O'SULLIVAN, C., DINGLIANA, J., GIANG, T., AND KAISER, M. K. 2003. Evaluating the visual fidelity of physically based animations. *Proceedings of SIGGRAPH 2003 22*, 3, 527–536.

PAVLOVIC, V., REHG, J. M., AND MACCORMICK, J. 2000. Learning switching linear models of human motion. In *NIPS*, 981–987.

PULLEN, K., AND BREGLER, C. 2002. Motion capture assisted animation: Texturing and synthesis. In *Proceedings of SIGGRAPH 2002*, 501–508.

REITSMA, P. S. A., AND POLLARD, N. S. 2003. Perceptual metrics for character animation: sensitivity to errors in ballistic motion. *Proceedings of SIGGRAPH 2003 22*, 3, 537–542.

REN, L., PATRICK, A., EFROS, A. A., HODGINS, J. K., AND REHG, J. M. 2005. A data-driven approach to quantifying natural human motion. *Proceedings of SIGGRAPH 2005 24*, 3, 1090–1097.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5, 32–41.

ROSSIGNAC, J. 1999. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics 5*, 1 (*/*), 47–61.

SAFONOVA, A., AND HODGINS, J. K. 2005. Analyzing the physical correctness of interpolated human motion. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 171–180.

SAFONOVA, A., HODGINS, J. K., AND POLLARD, N. S. 2004. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *Proceedings of SIGGRAPH 2004 23*, 3, 514–521.

SALOMON, D. 2000. *Data Compression: The Complete Reference*, second ed.

SATTLER, M., SARLETTE, R., AND KLEIN, R. 2005. Simple and efficient compression of animation sequences. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM Press, 209–217.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8, 888–905.

SLOAN, P.-P. J., CHARLES F. ROSE, I., AND COHEN, M. F. 2001. Shape by example. In *SI3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, 135–143.

SLOAN, P.-P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered principal components for precomputed radiance transfer. *Proceedings of SIGGRAPH 2003 22*, 3, 382–391.

TOLANI, D., GOSWAMI, A., AND BADLER, N. I. 2000. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical models 62*, 5, 353–388.

VECCHIO, D. D., MURRAY, R. M., AND PERONA, P. 2003. Classification of human motion into dynamics based primitives with application to drawing tasks. In *Proc. of European Control Conference*.

WANG, X. C., AND PHILLIPS, C. 2002. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 129–138.