Chapter 8 Model based coding

8.1 Introduction

Under the general heading of virtual humans, *model* can either be understood as geometric modeling or motion modeling. Geometric modeling refers to the *synthetic* model or appearance of the computer rendered virtual human figure. This type of modeling, as opposed to traditional pixel-based approaches, already introduces a significant bandwidth reduction. Two-dimensional pixel arrays require vast amounts of storage and transmission bandwidth, whereas a synthetically rendered image requires only the definition of meshes, materials and textures. Once the definition process has been completed, no further information is required to reproduce a high quality image at the receiving end. However, to reproduce a computer rendered image that exhibits credible motion, we need to transmit motion parameters as well. The transmission of motion parameters is a continuous process as opposed to geometric modeling, which is usually completed at the beginning of a session. If we accept the compression that is inherent to geometric modeling as a given, further reduction can be found by examining the motion modeling. Model based coding for virtual humans can therefore be described as any technique whereby parameters that define and control a mathematical model are used to predict and/or reproduce existing motion in a controlled environment.

Similar to many other coding methods, purely temporal methods and purely spatial methods can be regarded as the two extremes in model based coding. These two methods are orthogonal to each other, where "orthogonal" is not used in the strict mathematical sense. The differences between these methods relate to the way the model parameters are obtained and used. The temporal approach is to obtain time-varying *dynamic* actions or motions using a model, and to reproduce the motion using the same or a similar model. The spatial approach is to obtain a set of *static* key postures, and to use a model to reproduce the in-between motion. Of these two approaches, the temporal method has more potential, but it is also more difficult to implement for a wide range of arbitrary motion. The spatial approach is insensitive to most motion variations, but the compression ratio tends to saturate as the static postures become uncorrelated. The spatial approach is related to the temporal approach in that it *invokes* a model based animation upon changes in posture. Naturally, this is accompanied by a coding delay, which is undesirable, but at least controllable. On the other hand, motions recognized and reproduced using the temporal method cannot so easily be defined and controlled. The ideal solution lies somewhere in between these two extremes.

We define *postures*, *gestures* and *actions* as the essential elements of human motion or behaviour. Postures refer to static, spatial poses that the human figure can assume, and are described independently of temporal information. Gestures are based on temporal motions and behaviours, such as speed and acceleration. Some gestures require postural information, such as start or end conditions, but in general we try to keep the two definitions separate. Actions are a combination of both posture and gesture information. A sequence of one or more actions defines a complete motion. This terminology will be used in the remainder of this chapter.

In its most basic form, a model based coding/decoding scheme consists of three stages:

- A detection stage that recognizes postures, gestures, actions or any combination thereof,
- A reduction stage that reduces the information that is required to reproduce these elements according to a well defined error measurement, and
- A reproduction stage that animates the human figure.

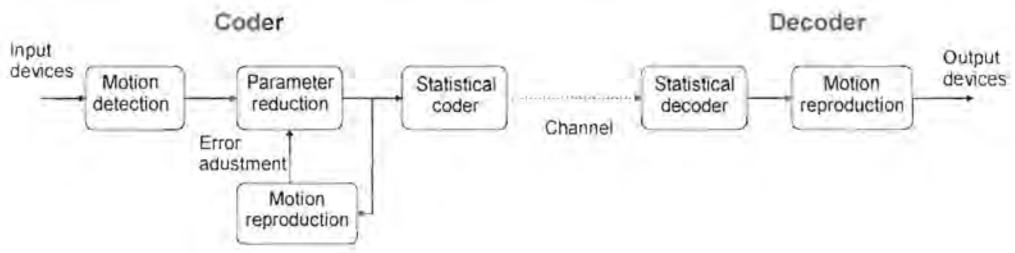

Figure 8-1: Generalized model based coder/decoder.

Figure 8-1 shows a diagram of a generalized model based coder. The reproduction stage is essentially part of the receiver, but also controls the error measurement of the reduction stage. Motion detection in itself is a particularly intricate subject, and can be classified in varying levels of complexity. A fair amount of previous (and current) research in human motion has focused on motion and action recognition [52-57,34]. For most of these systems an essential goal is *machine understanding*. Experimental motion recognition to date usually consists of a small alphabet of possible actions and an elaborate detection and classification algorithm that attempts to overcome problems such as gesture variation and performer independence. Methods such as Hidden Markov Models (HMM) [52] and Phase Space Constraints [54] have been implemented with a fair amount of success. These are high level methods, and applied to motion compression can result in tremendous reduction, but they lack generality.

Figure 8-2 depicts our view of the complexity of motion detection versus the number of parameters involved. At the low end of complexity, the number of parameters is high, such as the individual DOFs of the figure. For high complexity motion detection methods, there are very few parameters involved, in fact, it is conceivable that the whole motion or even the situation can be described by a single parameter. Figure 8-2 also shows four discrete levels on which motion detection can operate. The attribute level operates on individual DOFs, and each is processed independently from any other quantity. The posture level involves a group of DOFs that together form a static posture. On the gesture level, temporal information such as speed and acceleration are added. Finally, a situation level detection scheme attempts to classify whole sequences of motion, and tries to put the meaning of the action into context.

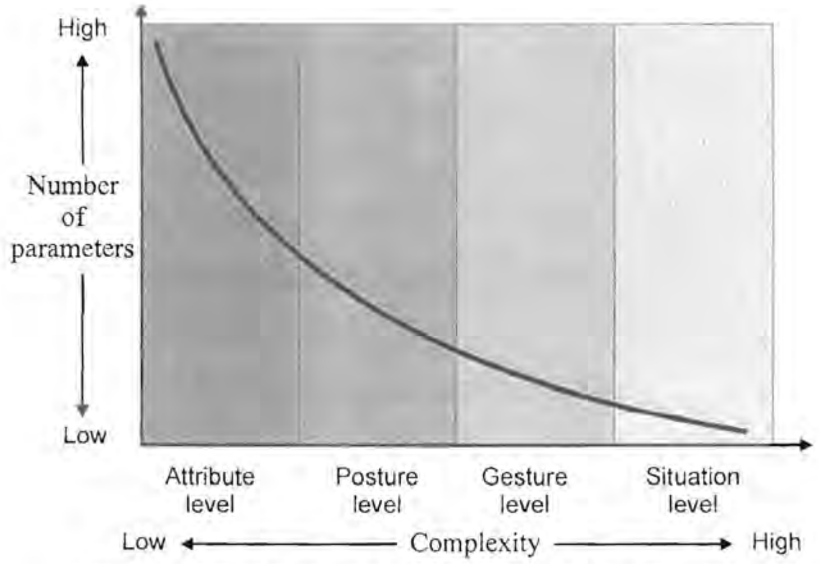

Figure 8-2: Motion detection complexity levels.

Fortunately, in our case machine understanding is not a requirement. As long as well defined and well behaved motion parameters can be detected, the actual *context* or *meaning* of those parameters is of no concern. In the model based coder of Figure 8-1, the motion detection stage plays a very important role. Proper motion detection will determine both the effectiveness and method of the reduction and animation stages. Attribute level parameters require vastly different animation techniques compared to situation level parameters. In this study, we limit ourselves to the first three levels of motion parameters. Attribute level parameters were discussed mainly in chapter 7 on waveform coding techniques (we rather use *parameter* instead of the term *detection*). In this chapter a posture level detection algorithm and a gesture level detection algorithm is investigated.

A final note: Although it is possible to achieve phenomenal compression ratios with model based coding methods, there are two severe penalties. Firstly, the inherent approach of parameter extraction requires a number of samples to be present. This introduces a variable and unpredictable coding delay, which could be as high as two seconds. Secondly, due to the highly synthetic nature of the decompressed motion, it is virtually impossible to compare the resulting motion on an objective scale to the original motion. Although the PSNR used in the waveform coding section still indicates a trend, this quantity frequently falls below 0 dB, which of course indicates more "noise" than "signal". The VPSNR is used in this chapter to better effect. The coefficients that define the VPMSE can be adjusted to provide an error measure in any range, but we would like to keep it compatible with the normal MSE. The coefficients used throughout this chapter are $\alpha_i = \{0.25, 0.5, 1, 2.25\}$ (refer to chapter 5). Using these values, the VPSNR typically lies between 0 and 20 dB for the model based methods. A VPSNR above 20 dB can be considered as very good quality, while values below 0 dB are considered unacceptable. Although useful, the VPSNR still cannot distinguish between natural, credible motion and completely artificial motion. These penalties should be taken into consideration when various waveform and model based methods are compared in terms of real-time interaction, quality and bit-rate.

8.2 Human motion segmentation

Motion detection and extraction of suitable parameters form the basis of a model based compression algorithm. Once appropriate parameters are available, they can be processed, compressed and transmitted. At the receiving end, the parameters are decompressed and processed into a suitable form for animation purposes. It is the nature of these parameters that primarily determine the way in which processing, compression and animation will be achieved. If human motion is seen as a temporal sequence of events it is evident that we need to "sample" the process at certain points in time to obtain appropriate parameters. The decision of exactly where and how to sample the motion should be made with the objective to obtain the maximum amount of information with the minimum number of parameters. In the chapter on waveform coding, sampling was not really an issue since each DOF was seen as an independent time signal sampled at regular interval by an input device. Clearly, there is quite a lot of variation in human motion, and uniform sampling is not necessarily the optimal approach. The term *segment* is used to describe the portion of motion between two non-uniform samples. In the following discussion, two different segmentation methods are developed, a posture based approach and a gesture based approach.

8.2.1 Posture level segmentation

The posture based segmentation algorithm is a relatively low-complexity method, since no temporal information is directly used. In this approach, the motion of a group of DOFs is sampled at constant distances, where *distance* is defined as a scalar quantity that is obtained by integrating and combining individual DOFs. The scalar distance at sample n is defined as

$$s_i(n) = \sum_{m=1}^{n} \sqrt{\sum_{j=0}^{K-1} \left(\theta_{i,j}(m) - \theta_{i,j}(m-1)\right)^2}, \tag{8-1}$$

where K is the number of DOFs for the ith group (see table 5-1). Clearly, equation (8-1) is a monotonically increasing function, and by sampling s_i at constant distances d apart, a unique sample value n can be obtained on the time axis. These time values are the locations at which the original motion is non-uniformly sampled, and segments are represented as the portion between such samples. This concept is illustrated in Figure 8-3. Since the rate of the input signal is known, and the exact location of each non-uniform sample is known, the length of each segment can be calculated if desired. Care should be taken not to use values of d that are too large, as the resulting aliasing effects would make the information unrecoverable.

Figure 8-3 also shows an example of the segmentation scheme for the right arm group. The wave sequence with $d = 40$ was used in this example. For clarity, only the upper arm twist angle $\theta_{22,2}$ is shown. Also shown are the monotonic function s_i and the resulting segments. It can be seen that the group motion is segmented frequently during moments of high activity, and less so during low activity.

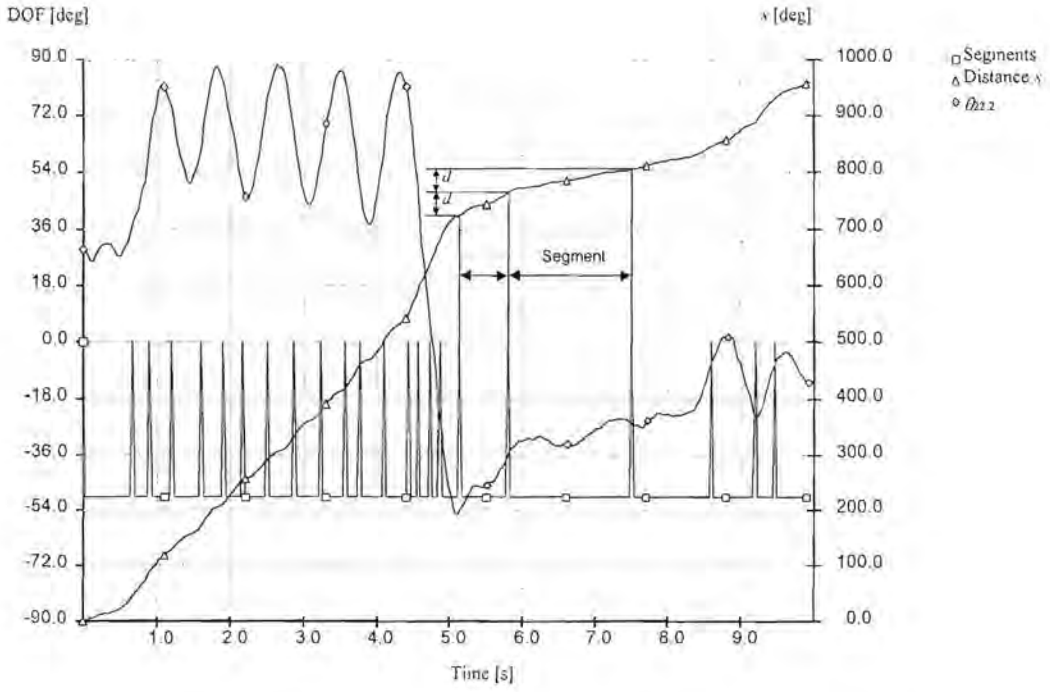

Figure 8-3: Posture based segmentation example.

8.2.2 Gesture level segmentation

Our gesture segmentation scheme follows a completely different approach than the posture based method. In fact, posture information is completely discarded and the temporal information contained in the first and second derivatives (speed and acceleration) of a group of DOFs is used. In this temporal context, a *gesture* is defined as the motion of a group of joints or DOFs between two successive stationary (or almost stationary) moments. A segment is the portion between the local minimums of the speed, provided that it is not larger than a certain threshold. The location of these minimum values are used to non-uniformly sample the motion. In order to segment the gestures of a group of DOFs as a whole, and still retain the convenience of mathematically tractable equations, we need to define some combination of individual DOF derivative information. A scalar speed quantity ω_i is defined as the length of a K-dimensional speed vector, whose elements are the first derivatives of the K DOFs of the ith group. It can be written as

$$\omega_i(n) = \sqrt{\sum_{j=0}^{K-1} \dot{\theta}_{i,j}(n)^2}, \tag{8-2}$$

which is a positive quantity. Clearly, there is a relationship between equations (8-2) and (8-1), i.e. $\omega_i(n) = \dot{s}_i(n)$. However, we use a more stable estimation of the speed as explained below. The scalar acceleration is simply given by

$$\alpha_i(n) = \dot{\omega}_i(n), \tag{8-3}$$

which is a bipolar quantity. Note that the acceleration is *not* defined as the length of the original DOF acceleration vector. There are a number of methods for estimating the derivative of a sequence of discrete time samples. We fit a third order polynomial through a set of points using a least squares solution, and then analytically calculate the derivative at the desired sample. It is reasonable to assume that a small enough window will not have an adversely averaging effect. We have found that a window length of seven samples gives acceptable results.

Practical implementation of the segmentation algorithm is a bit more involved than the posture based method. We propose an adaptive speed threshold algorithm to identify possible windows of interest. The zero crossings of the acceleration quantity within the window are then used to find the exact sampling locations. Figure 8-4a shows a blown up region of a typical signal. The threshold signal is a low-pass filtered version of the speed, and is given by

$$\zeta_i(n) = \beta \sum_{k=0}^{M-1} \omega_i(n-k)h(k). \tag{8-4}$$

The impulse response $h(k)$ can be any suitable form of a low-pass filter. The choice of filter length M is also not critical, and can be adjusted to suit the characteristics of the segmented motion. The quantity β is a scaling factor that determines the aggressiveness of the threshold process, and is a convenient single parameter with which to adjust the performance of the algorithm.

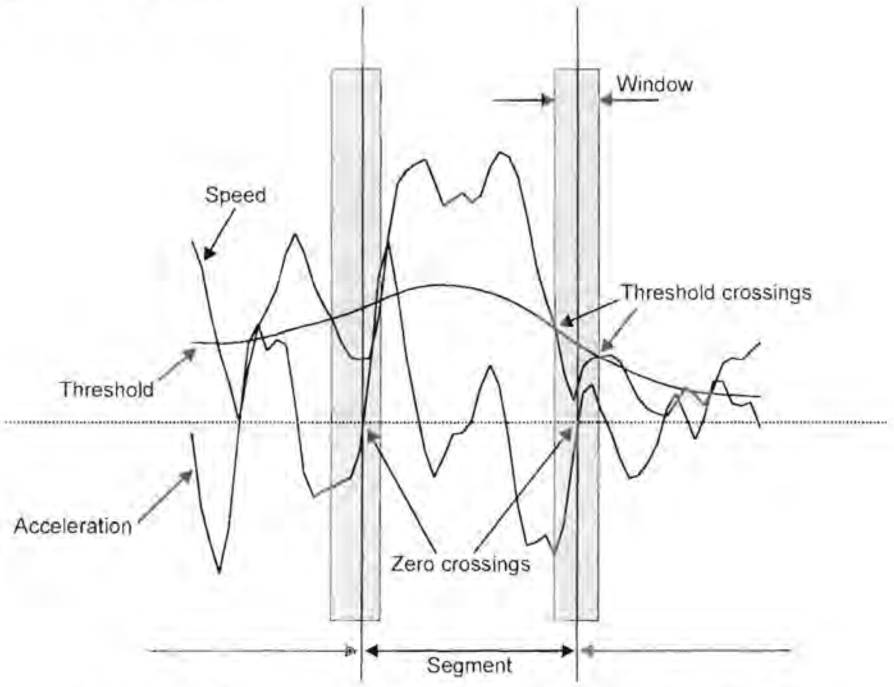

Figure 8-4a: Gesture based segmentation algorithm.

The algorithm is implemented recursively as follows: A possible window of interest is identified between an above/below threshold crossing and a below/above threshold crossing for the speed (see Figure 8-4a). The acceleration is then evaluated within the window for all the zero crossings. A minimum number of samples between each non-uniform sample are specified, and any values in this range are discarded. A maximum number of samples are also specified. When any new segment detects an excessive length, the value of β is adjusted to a slightly higher value and the algorithm is recursed again for the offending region. Some types of motion may result in an endless loop, and it is prudent to put an upper limit on the value of β, after which the recursion immediately unwinds. Excessively long segments can be left as they are, or subdivided by other means. It should be noted that for such long segments the motion probably had very little information content anyway, and the non-uniform sampling should be acceptable as it is.

Figure 8-4b shows an example of the segmentation scheme for the right arm group, and can be compared with Figure 8-3. Also indicated are the speed, threshold and acceleration quantities. It is clear that the segmentation follows a completely different pattern compared to the posture based approach. There are fewer samples, and they are more uniformly spaced. The well defined sampling of $\theta_{22,2}$ can clearly be seen (i.e. where the derivative is almost zero), compared to the more random behaviour of the posture based approach in Figure 8-3.

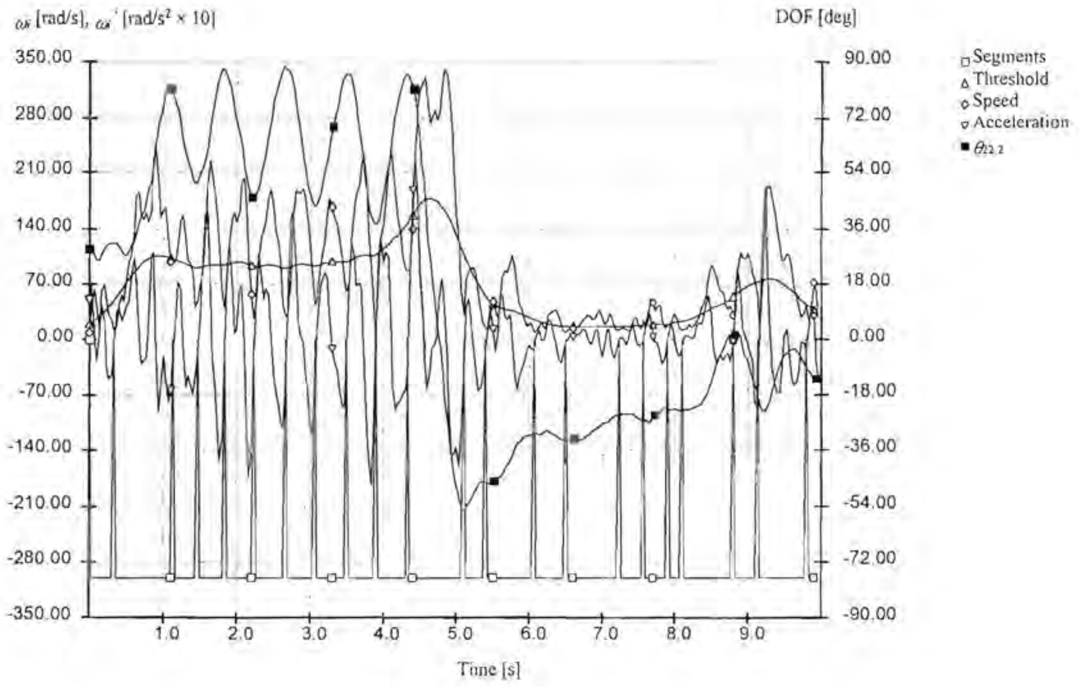

Figure 8-4b: Gesture based segmentation example.

8.3 Segment animation

As will be discussed later, data reduction can be achieved by transmitting only the non-uniform samples and possibly information concerning the lag between samples. This implies that the "missing" motion contained in the segments must be estimated. Segment animation can be done in a number of ways, of which three will be discussed briefly, namely end point interpolation, dynamic simulation and motion capture segments.

8.3.1 Interpolation

The most basic form of animation is the interpolation of motion between two static start and end postures. A variety of interpolation functions can be used depending on the boundary conditions. Smooth motion animation is a strong requirement. This condition implies that the interpolation function should at least have G^0 and G^1 continuity at the endpoints, i.e. the end/start positions of two consecutive segments should match exactly, and the first derivative should be equal on both sides. We have found that Hermite cubic polynomial curve segments [12] between two non-uniform samples provide satisfactory results. Such a function can be written in parameterized form as

$$w(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, \tag{8-5}$$

where t is a parameter in [0, 1]. The lag between the actual start and end points should be normalized to the parameter t for equation (8-5) to be meaningful. It can be shown that the coefficients a_i are given by

$$\begin{aligned} a_0 &= \theta(m-1), \\ a_1 &= \Delta(m-1), \\ a_3 &= 3\big(\theta(m) - \theta(m-1)\big) - 2\Delta(m-1) - \Delta(m), \\ a_4 &= 2\big(\theta(m-1) - \theta(m)\big) + \Delta(m-1) + \Delta(m). \end{aligned} \tag{8-6}$$

at each non-uniform sample m. Note that the subscript i,j has been dropped from the DOFs $\theta(m)$ for clarity. The quantity $\Delta(m)$ is the slope (first derivative or speed of the DOF) of the signal at sample m. $\Delta(m)$ is not directly available, and must be estimated since it is undesirable to transmit this value along with the posture information for *every* DOF in the group. We use an estimate of the form

$$\Delta(m) = s\big(\theta(m) - \theta(m-1)\big) + (1-s)\big(\theta(m+1) - \theta(m)\big), \tag{8-7}$$

where

$$s = \frac{r(m+1)}{r(m)+r(m+1)} \tag{8-8}$$

is a scaling factor in [0, 1], and $r(m)$ is the lag between non-uniform sample m and m-1. By using a sample at m+1, even further coding delay is incurred since we have to wait for the sample to arrive. However, the slope estimate of equation (8-7) is much more accurate than a first order approach, and especially at zero-derivative turning points the overshoot is much less.

It will be seen later that the gesture based segmentation method provides the implicit and very useful information that the derivatives of the interpolated DOF are zero (or almost zero) at the endpoints. After the work done by Perlin [18,19], we have found that a raised cosine interpolation function

$$w(t) = \frac{1+\cos(\pi t)}{2}, \tag{8-9}$$

where t is a parameter in [0, 1] gives very natural results, and fulfills the derivative condition at the endpoints. Interpolation using this function is straightforward, and is given by

$$\theta(t) = w(t)\theta(m-1) + \left(1-w(t)\right)\theta(m), \tag{8-10}$$

where it is understood that the parameter t has been normalized between the non-uniform samples at m-1 and m.

8.3.2 Dynamic simulation

The use of dynamic simulation of human motion has been put to very effective use recently in the field of computer animation. By making use of efficient algorithms, the basic equations of motion can be solved for complex hierarchical structures in real-time.

The use of dynamic simulation has been discussed in chapters 2 and 3, and Appendix II gives a summary of an algorithm that can be used.

Unfortunately, proper dynamic simulation requires well defined initial conditions and parameters to ensure a stable solution of the differential equations. The rather poor estimate of the speed (or slope) using the posture based approach causes numerical instabilities and overshoot when using dynamic simulation as a segment animation method. On the other hand, the closely spaced and fixed segments of the gesture based approach diminish the benefits of full dynamic simulation. Due to the adaptive algorithm, segments are seldom overly long, and dynamic simulation typically gives similar results as simple interpolation. The fact that the speed is not always exactly zero at the segment endpoints also destroys the finer detail and subtlety that could have been obtained with dynamic simulation.

8.3.3 Motion capture segments

A third approach to segment animation is to fill the segment with actual motion captured data. Given a substantial list of properly processed motion capture segments, a search can be performed against a given error criteria, and the best match fitted to the segment. When the error requirement cannot be fulfilled, the segment can simply be interpolated using one of the previous methods. The actual generation of a suitable table of motion entries is quite a daunting task, and requires extensive training sequences. The concept of such a table of motion is described in more detail in section 8.4, where the actual implementation of a model based coder is discussed.

Segmented motion requires a method of combining a sequence of actions to form smooth, continuous motion. Similar to the interpolation scheme discussed above, the end and start sections of two motion captured segments can be blended together using an interpolation function. Figure 8-5 shows an example of blending two DOFs $\theta_{i,j}$ and $\theta'_{i,j}$ from completely different motion segments. In practical compression methods, such severe differences will usually not be tolerated. The interpolation function of equation (8-9) has been found to give good results and is used for blending motion segments.

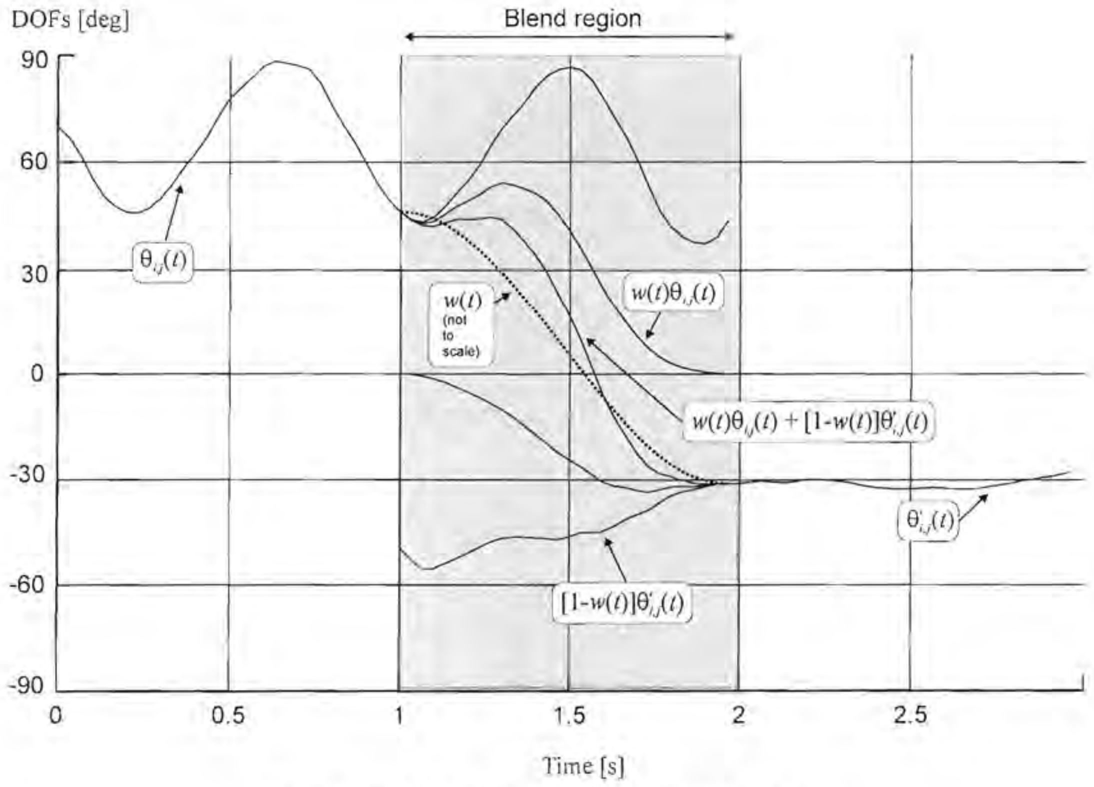

Figure 8-5: Example of motion blending.

8.4 A model based coder/decoder

Once a proper segmentation and animation algorithm is in place, a model based coder can be constructed by realizing that it is simply needed to transmit the non-uniform samples, as well as some information conveying the location of the samples. As with waveform coding techniques, it is crucial to quantize the actual values of the samples to a well-defined set of symbols in a controlled fashion. Although the values of the individual DOFs in a group can be quantized separately, more gain is achieved by making use of the correlated nature of the DOFs. It is therefore more appropriate to use some form of vector quantization technique. As discussed in the previous section, a further quantity that will be transmitted is a code describing the motion segment that will be fitted between the current samples. We define the term *posture table* for the vector quantization table containing the group

DOF values, and the term *gesture table* for the table containing the list of motion segment candidates.

8.4.1 Posture table

A posture table can be constructed in a similar fashion to the spatial vector quantization method presented in the previous chapter. However, in this case we have found that the *position* of joints in 3D space gives better results than the joint angles directly. A vector sequence $\{\mathbf{p}_i(n)\}$, consisting of all the x, y and z Cartesian joint position components of group i, is constructed using a *training* motion sequence. The individual components of $\mathbf{p}_i$ are denoted by $p_{i,j}$, $0 \leq j < K$, where K is the dimension of the composite position vector. We use a greedy $O(N^2)$ search algorithm to obtain the M vectors that are spaced a distance d or greater apart. This is an alternative approach to the clustering LBG algorithm presented in chapter 7. Similar to the error measure of equation (6-13), all the entries in the sequence which satisfies

$$\sum_{j=0}^{K-1} \frac{a_{i,j}\left(p_{i,j}(n) - p_{i,j}(m)\right)^2}{b_{i,j}} > d^2, \quad 1 \leq n \leq N, n < m \leq N, \qquad (8\text{-}11)$$

for a sequence length of N are retained. For simplicity, we set the matrices $a_{i,j}$ and $b_{i,j}$ to unity and take the square root to obtain units of meters. By adjusting the value of d, the number of codebook entries M can be adjusted. The joint angles corresponding to a vector position $\{\mathbf{p}_i(n)\}$ are also stored in the resulting table of entries, so that a position vector can directly be mapped to a set of angles for animation purposes. The complete figure contains 8 groups (see table 5-1 for details), and there are therefore 8 separate vector quantization tables. The posture table for group i is denoted by $\mathbf{P}_i$, and contains M codebook entries $\{\mathbf{p}_m\}_i$, $0 \leq m < M$.

8.4.2 Gesture table

The gesture table is constructed similarly to the posture table, except that we now consider small sequences of motion, rather than a single sample or static posture of motion. We propose the following method for the construction of a gesture table: A large training sequence is segmented using the gesture based algorithm described above. Although in theory it is possible to use posture based segmentation, the resulting segments will be poorly defined and not of much use. After segmentation, the gesture table contains *all* the motion from the training sequence. Segments that are similar are discarded according to some error or distance measure. We use a simple weighted MS similarity measure of the form

$$\varepsilon = \sum_{n=1}^{N} w(n) \sum_{j=0}^{K-1} \left(\theta_{i,j}(n) - \theta'_{i,j}(n)\right)^2, \tag{8-12}$$

where N is the length of the segment and K is the number of DOFs in the ith group. The weighting function $w(n)$ is given by

$$w(n) = \begin{cases} b + \dfrac{(a-b)}{n}, & 0 < n \le \dfrac{N}{2},\ a \ge b \\ b + \dfrac{(a-b)}{N-n+1}, & \dfrac{N}{2} < n \le N,\ a \ge b \end{cases}. \tag{8-13}$$

For $b = a$, equation (8-13) is constant over the segment. For $b < a$ it can be seen that less importance is given to the interior of the segment. The segment start and end values are important to ensure a good match at the boundaries. A greedy $O(N^2)$ search algorithm is used to discard all the segments that are closer than some error distance. In order to accommodate segments of varying length, we propose that each segment is warped or normalized in time to a predetermined length. Small segments are therefore stretched in time, and long segments shrunk. The normalized length is a function of the maximum segment length that is allowed by the segmentation algorithm. We have found that a value of one second (or 30 samples at an input sampling rate of 30 Hz) is appropriate. Similar to the notation used for the posture table, the gesture table for group i is denoted by $\mathbf{G}_i$, and contains M normalized codebook sequences $\{\mathbf{g}_m(n)\}_i$, $0 \leq m < M$. The quantity $\mathbf{g}_m$ is a vector for which its elements are the DOFs of the group in question.

8.4.3 Posture based compression

We define posture based compression as posture based segmentation followed by vector quantization of the samples using a posture table. After segmentation, a group position vector $\mathbf{p}'_i(m)$ is generated at each non-uniform sample m for the sequence to be coded (the coding sequence is similar, but not identical, to the training sequence). The position vector for the mth sample, $\mathbf{p}'_i(m)$, is matched against the posture table $\mathbf{P}_i$, and the closest entry is selected (we simply use an L_2-norm). The result is a string of codes that is passed to a statistical coder to obtain a symbol stream of optimal length. Decompression reverses the process by decoding the symbols and obtaining the corresponding posture from the posture table. The method described by equations (8-5) to (8-8) is used to interpolate the motion between consecutive non-uniform samples.

The algorithm has a penalty in the form of variable coding delay, which can be quite severe in some cases. For practical reasons, it is convenient to compensate for the (known) delay when generating comparison results such as rate distortion graphs. Figure 8-6a shows the results for two different posture segmentation distances d. The example shows the right upper arm twist angle $\theta_{22,2}$ for the wave sequence. Clearly, if d is too large severe aliasing occurs and the motion cannot be reconstructed. For a smaller value of d, the motion is non-uniformly sampled at sufficiently small intervals to reconstruct the motion, albeit in a very synthetic fashion. Figure 8-6b depicts a number of 3D wireframe images from the dance sequence at $d = 20$, which corresponds to roughly 450 bits/second. Although there is a substantial difference, the relationship between the original and coded motion is still clear. These results are best viewed from the video clips provided on CD-ROM.

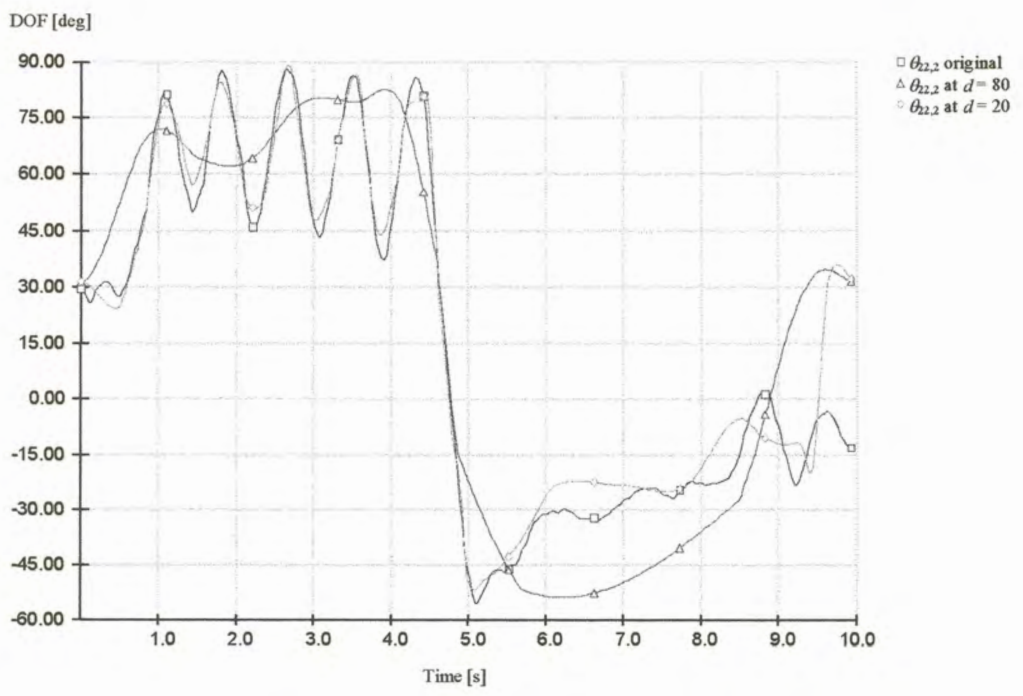

Figure 8-6a: Posture based coding of $\theta_{22,2}$ for the wave sequence.

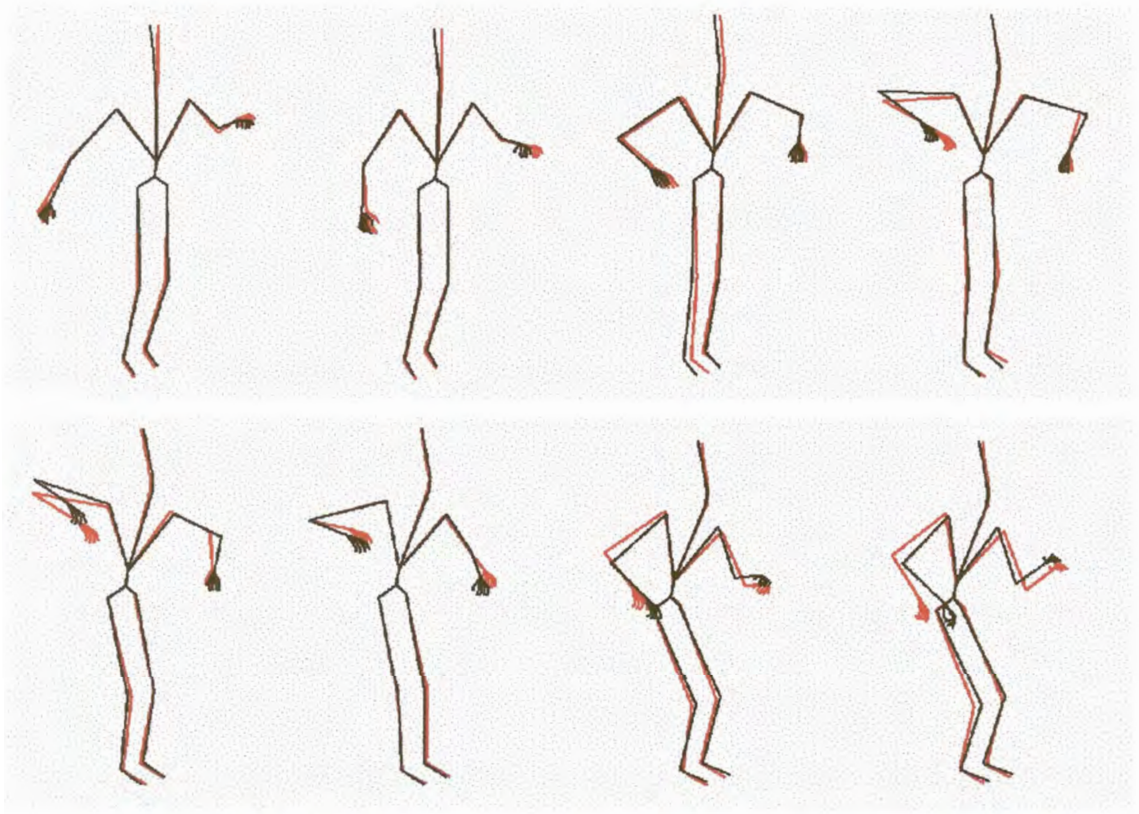

Figure 8-6b: Wireframe images from the dance sequence with $d = 20$.

Figure 8-7 shows the VPSNR against bit-rate for the various test sequences. The gesture example contains only compression results for the right hand, hence the lower bit-rate. Note the extreme change of bit-rate scale compared to the waveform coding examples. It can be seen that model based coding outperforms the previous methods by almost two orders of magnitude. In addition, a much larger variation in bit-rates is observed between the test sequences compared to the waveform coding techniques. The large variation indicates the high sensitivity of model based coding to the type of motion that is being coded. The highly active dance sequence requires many more bits than the passive conversational sequence. It is also clear from Figure 8-7 that not much is visually gained at higher bit-rates. This is primarily due to the sensitivity of the motion interpolation process to quantization errors. As the samples become more closely spaced, the vector quantization effects of the posture table cause significant and frequent errors in the slope estimation. A poor slope estimate in turn introduces overshoot in the interpolation function, and the resulting high-frequency errors are highlighted immediately by the formulation of the VPMSE.

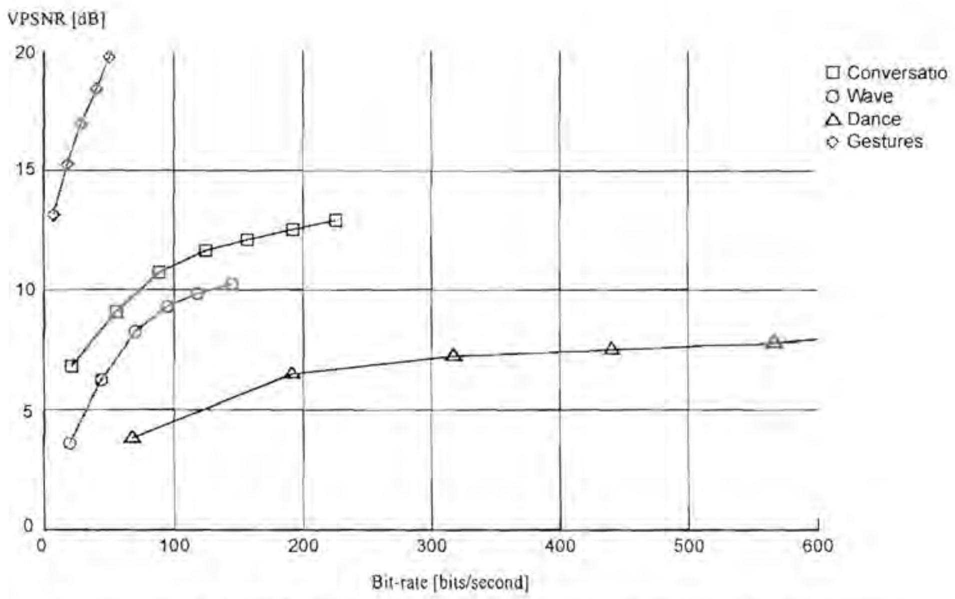

Figure 8-7: VPSNR vs. bit-rate for posture based compression.

8.4.4 Gesture based compression

We define gesture based compression in a similar fashion as posture based compression, except that gesture based segmentation is used, followed by gesture quantization (using a gesture table) and possibly also posture quantization.

After segmentation, a group gesture sequence $\{\mathbf{g}'(n)\}$ is extracted from each segment for the sequence to be coded. The sequence $\{\mathbf{g}'(n)\}$ is normalized to the same length as the entries in the gesture table $\mathbf{G}_i$, and the closest entry is selected using equation (8-12). The resulting string of codes is passed to a statistical coder to obtain a symbol stream of optimal length. Decompression reverses the process by decoding the symbols and obtaining the corresponding motion segment from the gesture table. The method shown in Figure 8-5 and equation (8-9) is used to interpolate the motion between consecutive segments.

Although the gesture table will always return the entry that is closest given the similarity measure, the error is often still too large. The resulting blended motion is usually completely inappropriate in the context. Note that except in cases of severe misalignment, the motion is almost never unnatural, just inappropriate. The concept of inappropriateness is different from the continuous unnatural synthetic look of pure posture interpolation. However, if a certain error threshold e is exceeded, we have found that it is visually still more acceptable to dismiss the motion segment completely and to revert to posture interpolation. In practice, extra bits are required to convey such changes, but do not significantly change the performance of the algorithm.

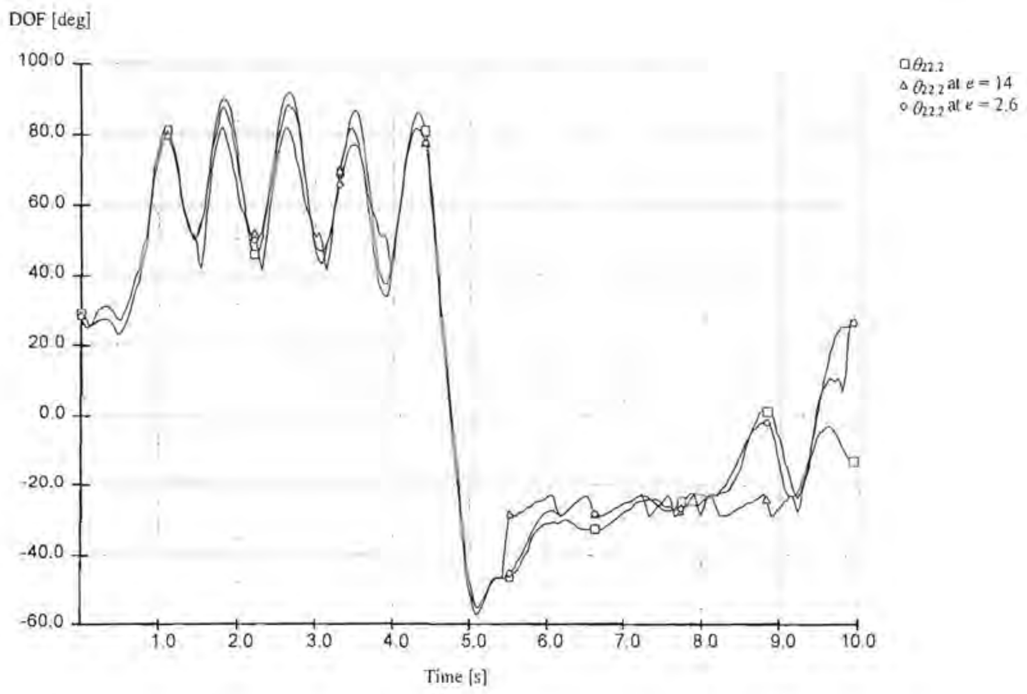

Figure 8-8a: Gesture based coding of $\theta_{22,2}$ for the wave sequence.

The gesture based algorithm also suffers from a variable coding delay, although not as severe as the posture based method due to the restrictions placed on the segmentation process. The known coding delay has again been compensated for in the following results. Figure 8-8a shows the results for two different similarity thresholds. The example shows the right upper arm twist angle $\theta_{22,2}$ for the wave sequence. For large values of the gesture/posture switch threshold e, the inappropriate choice of motion segments can be seen, especially during low activity. For smaller values of e, there is a clear improvement, albeit at the expense of a higher bit-rate. As is to be expected, Figure 8-8a shows more motion detail than Figure 8-6a, which is an inherent feature of using actual motion to fill in missing segments. Figure 8-8b depicts a number of 3D wireframe images from the dance sequence at $e = 5$, which corresponds to roughly 220 bits/second. The motion is coded at a lower rate, and follows the actual motion more closely compared to the posture based approach of figure 8-6b. These results are best viewed from the video clips provided on CD-ROM.

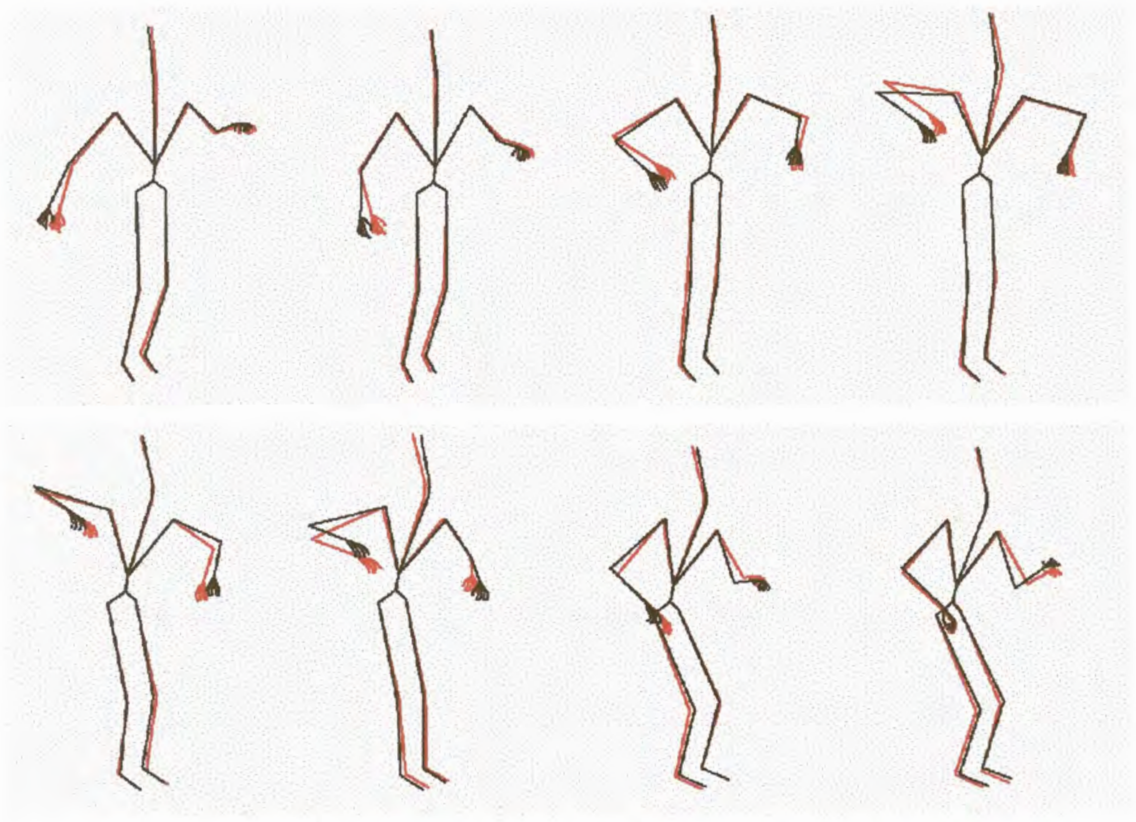

Figure 8-8b: Wireframe images from the dance sequence with *e* = 5.

Figure 8-9 shows the VPSNR against bit-rate for the various test sequences. Unfortunately, the gesture based segmentation is inherently fixed, and the spacing of non-uniform samples cannot be so conveniently adjusted as in the posture based segmentation scheme. The gesture based algorithm is also more complex than the posture based algorithm, and it is difficult to define a set of parameters that can be easily changed to produce a wide range of rate-distortion values. In any case, even the limited range in Figure 8-9 shows that the gesture based algorithm offers the same remarkable compression ratios as the posture based method. It exhibits the same sensitivity to variation in the type of motion, but there is a clear improvement in sensitivity to quantization noise. This is primarily due to the fact that there is no slope estimation requirement, and the use of the inherently more stable interpolation function of equation (8-9) compared to equation (8-5).

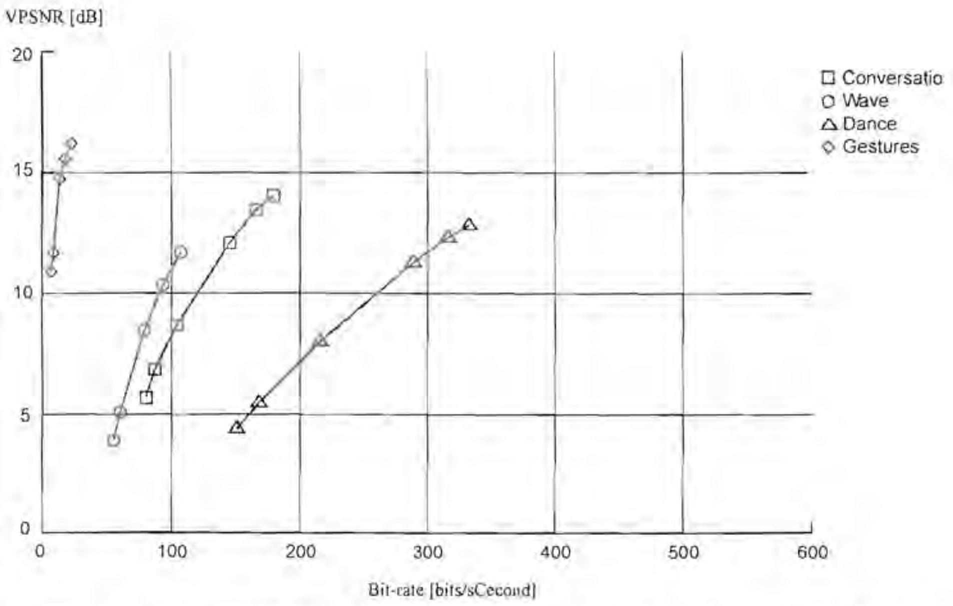

Figure 8-9: VPSNR vs. bit-rate for gesture based compression.

8.5 Hybrid coding

Model based coding methods, and especially motion interpolation between non-uniform samples, result in very high compression ratios. The price that we pay is that the result often does not resemble the original motion at all, and the animation has an unnatural and synthetic look to it. One way to overcome this would be to sacrifice a drop in compression ratio and to combine some of the waveform techniques from the previous chapter with model based methods. Obviously there are numerous ways to accomplish this combination. For example, one can simply use waveform coding techniques on groups where high accuracy is desired and model based techniques on the remaining groups. Visual artifacts in the root and torso group propagate through the whole figure, and high accuracy waveform coding on this group would be beneficial. Another example would be to continuously switch and blend on a temporal level between waveform coding and model based coding according to some rule. If the concept of virtual humans is extended to a whole virtual environment inhabited by other virtual entities and objects, interaction and collisions might force the use of higher resolution coding to resolve ambiguities.

There are endless possibilities and combinations of model based and waveform coding methods, most of which are quite trivial to implement, once the compression basis has been set. Rather than stating the obvious, we develop a more obscure combination of gesture based segmentation with transform coding of segments. It has been seen in chapter 7 that the DCT based method provides very efficient compression, but is plagued by block start/end mismatch effects. In model based coding, segment mismatch effects have been eliminated quite successfully through interpolation and blending techniques. Furthermore, it has been shown that the raised cosine interpolation function of equation (8-9) gives very natural looking results when combined with the gesture based segmentation method. This is primarily due to the fact that the first derivative (or speed) of the coded DOF is very small at the sampling points compared to the whole speed range. This raises the question of how well Fourier techniques can be used to represent motion, and what role each frequency component plays. Refer to [58] and [59] for an interesting discussion on such techniques.

The methods of DCT coding and gesture based segmentation have already been discussed, and will not be repeated here. In order to combine the methods, the DOFs of a group are segmented on a gesture level using the scalar speed method of equations (8-2) and (8-3) to obtain a segment of N samples. The segment is transformed using the DCT, which results in N frequency components. Note that N might not be a power of two and therefore an efficient DCT algorithm could not be used. The first three frequency components are retained, and each is quantized to a predetermined number of levels. The resulting codes are sent to a statistical coder, from which the optimal symbol stream can be transmitted. The decoder simply performs the inverse operations, and the contents of the segment can be obtained. By performing these steps, it is immediately clear that the more animation friendly non-uniform sampling method reduces the visual artifacts of the DCT block effects considerably, and it is possible to use fewer frequency components with coarser quantization. Since the segmentation cannot always be perfect, there are occasional jerks in the motion. We simply use the blending concept shown in figure 8-5 to blend the first few samples of the decoded segment with the last sample of the previous segment. The result is smooth motion throughout. The same variable coding delay issues are present here, but since the waveform based DCT methods also have a (fixed) coding delay, not too much is sacrificed.

Figure 8-10 shows the results of the hybrid technique for the right upper arm twist angle $\theta_{22,2}$ for the wave sequence. Two possible transform coefficient bit allocations are indicated, one for low quality low bit-rate and one for high quality (the exact quantization scheme is detailed in Appendix III). It is quite clear that the results are not comparable with higher accuracy DCT methods, but the resulting bit-rate is substantially lower. We therefore still have to use the VPSNR measure as discussed below. The visual results are similar or better than the equivalent pure model based approaches. The benefits of such a hybrid method is that it is much more robust and general than some of the model based techniques. There is no requirement for extensive motion specific training sequences, and the hybrid coder is able to resolve some of the finer detail that is missing in more synthetic approaches.

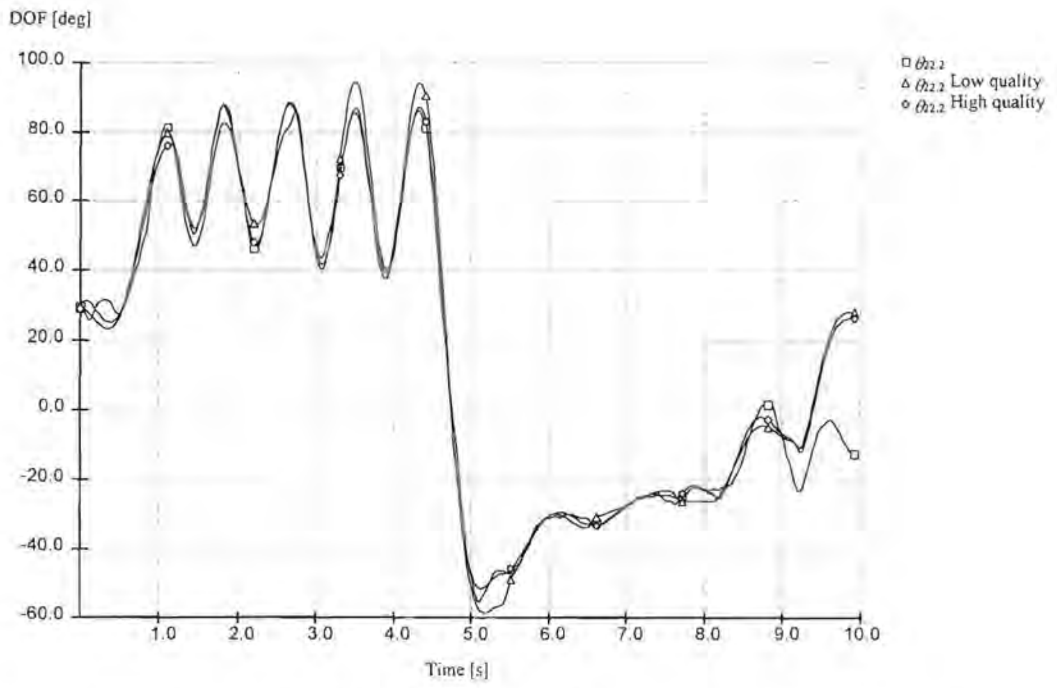

Figure 8-10: Hybrid coding of $\theta_{22,2}$ for the wave sequence.

Figure 8-11 shows the VPSNR against bit-rate for the various test sequences. Although the algorithm has its benefits, the information that needs to be sent is substantially more. Additionally, the information is on a per-DOF basis, and not in vectorized format. The expected drop in compression ration can clearly be seen in figure 8-11, compared to figures 8-7 and 8-9. The performance of the algorithm lies roughly halfway between equivalent model based and waveform based methods.

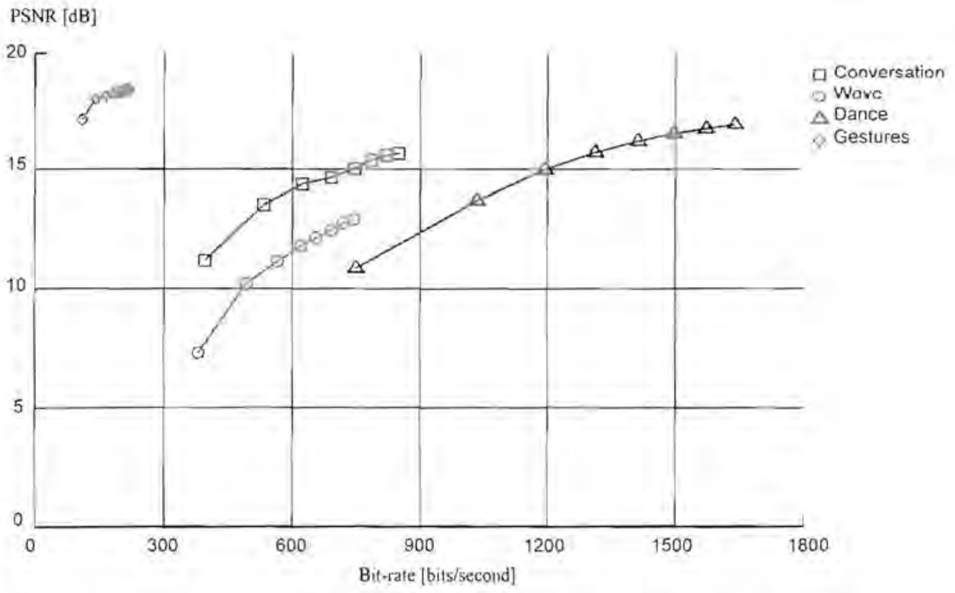

Figure 8-11: VPSNR vs. bit-rate for hybrid compression.

8.6 Summary

This chapter presented two model based compression methods as well as a hybrid model/waveform compression technique. The concept of non-uniform sampling or segmentation was introduced. Two segmentation methods were developed, namely posture or spatial based segmentation and gesture or temporal based segmentation. A combination of these segmentation methods with posture and gesture quantization tables led to the development of highly effective compression algorithms. Although the resulting motion animation from strictly model based compression techniques is very synthetic, there is no doubt about the underlying conveyance of information. A more robust hybrid compression technique was also developed by combining transform based waveform coding with model based segmentation.

Chapter 9 Comparison and discussion

The previous chapters presented the compression results by type of *coding* process. A comparison on the performance of a specific technique could therefore be evaluated for the various test sequences. Once a certain type of motion has been established it is also desirable to compare the performance of various coding methods against each other. This chapter presents the compression results by type of *motion*. Unfortunately the waveform and model based approaches cannot be directly compared, as the MS error measurement is used for the first, and the VMS error for the latter. However, a lot of insight can still be gained by comparing various methods, both objectively and subjectively. Naturally, subjective comparison can only be done properly by evaluating the video clips provided on CD-ROM.

9.1 Waveform coding comparison

Figure 9-1a to 9-1d show the PSNR against bit-rate for the various waveform coding algorithms. Although there is some overlap, a clear distinction can be observed between the performances of the various methods. In general, the dead-reckoning and straight quantization algorithms do not perform very well. The adaptive predictive method is a good choice for a low complexity, low delay method that yields high signal-to-noise ratios at moderate bit-rates. If higher complexity and coding delay is tolerable, the DCT based method clearly outperforms all the other waveform coding techniques. The temporal vector quantization method shows interesting behaviour, but the underlying lack of generality of the method makes it undesirable. Spatial vector quantization performs well at very low bit-rates, but the almost non-existent signal-to-noise ratio renders it useless as a naive waveform coding method. However, note the interesting behaviour for the gesture sequence in figure 9-1d. This can be explained by the high spatial correlation that exists between the finger joints in general gesturing motion.

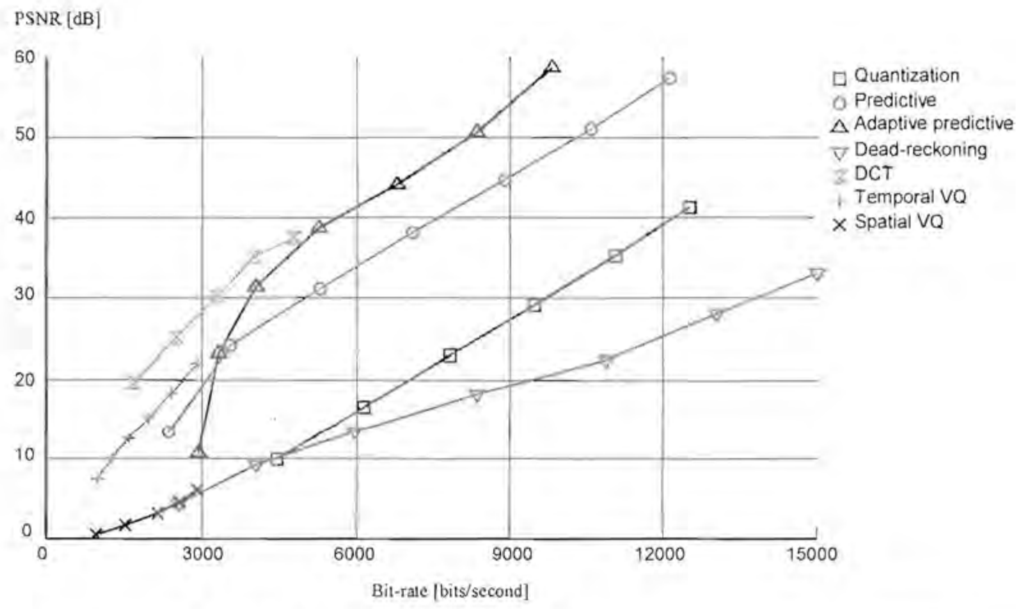

Figure 9-1a: PSNR vs. bit-rate for the conversational sequence.

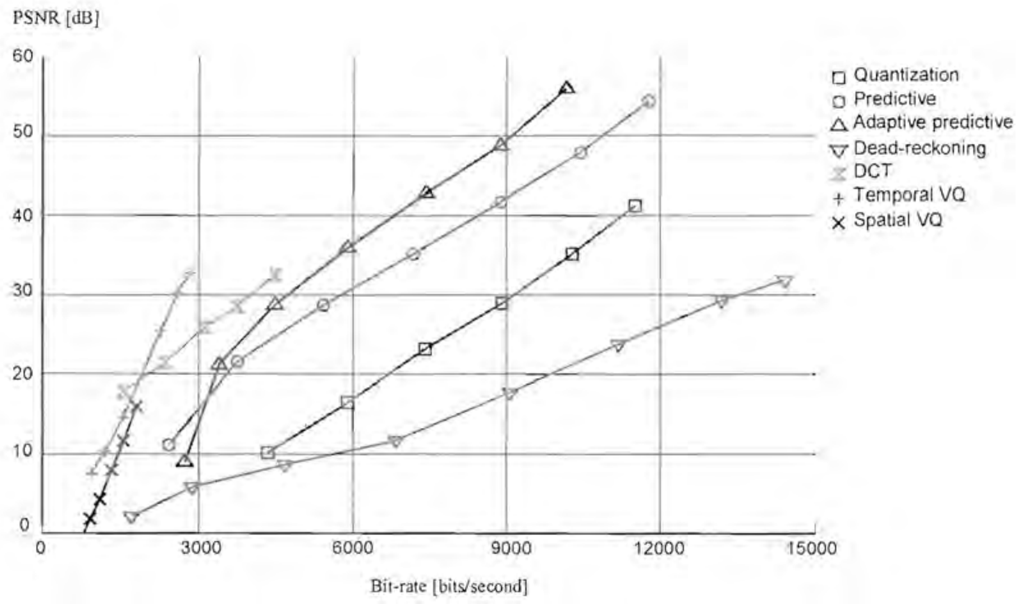

Figure 9-1b: PSNR vs. bit-rate for the wave sequence.

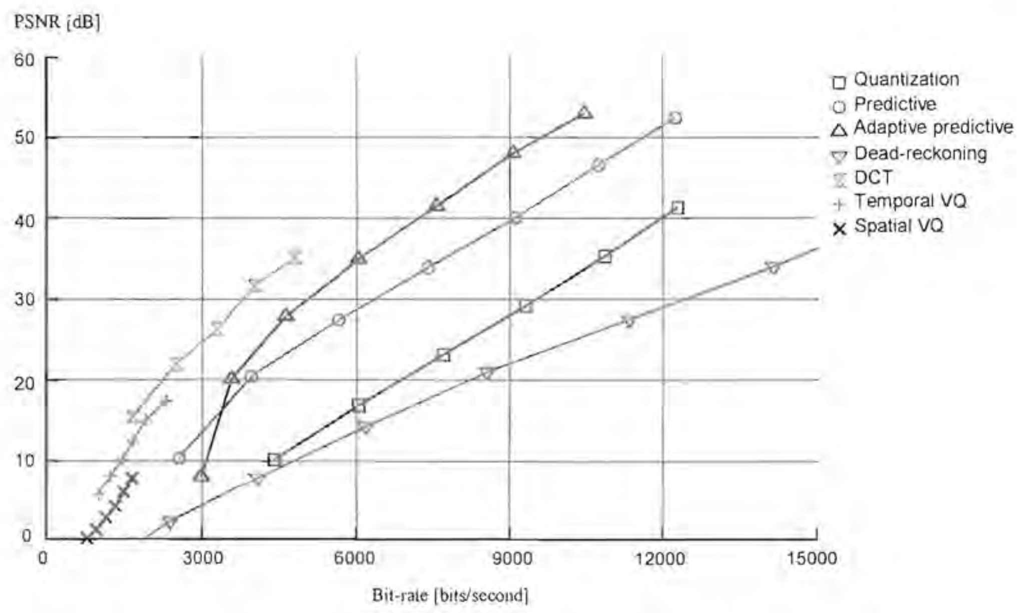

Figure 9-1c: PSNR vs. bit-rate for the dance sequence.

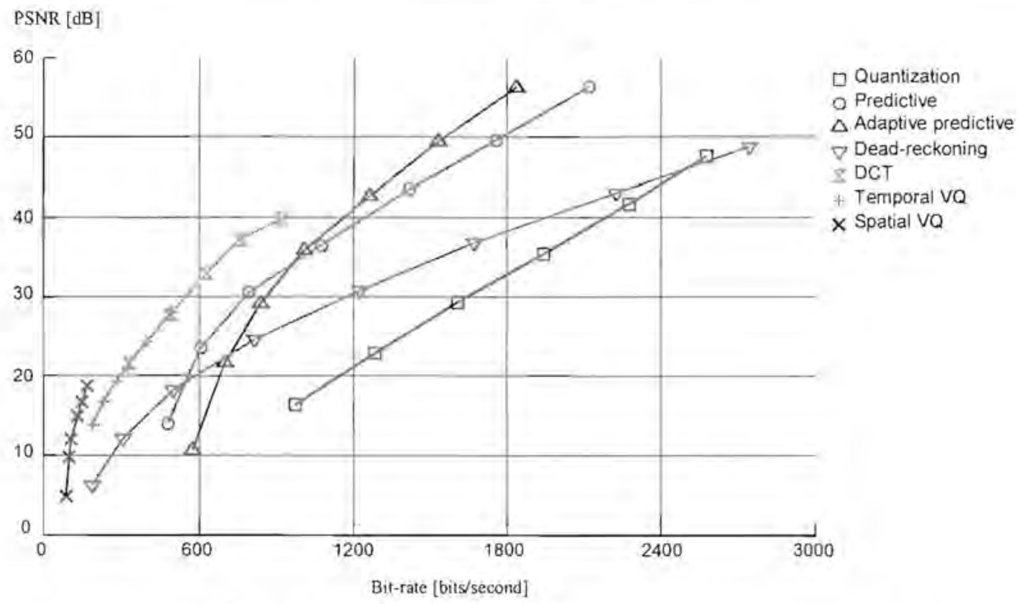

Figure 9-1d: PSNR vs. bit-rate for the gesture sequence.

At a glance, it can be seen that the waveform coding methods are surprisingly insensitive to type of motion. This is explained by the fact that most of the methods are based on uniform sampling techniques, and that the amount of "information" stays relatively constant from sample to sample. An exception of course is dead-reckoning, and it is indeed clear that this technique exhibits the most variation.

9.2 Model based comparison

Figures 9-2a to 9-2d show the VPSNR against bit-rate for the model based and hybrid coding algorithms. For all the test sequences there is a substantial increase in bit-rate for the hybrid method. This is to be expected, since the algorithm is more generalized and requires more information to be sent. The posture based and gesture based methods occupy more or less the same working range, and they intersect at a point within this range. However, at the extremes there are radical differences. The gesture based approach is in general more behaved, but it cannot cover the bit-rate range of the posture based approach. This is due to the fact that the non-uniform sampling process of the gesture based segmentation method is inherently rigid, and cannot be changed without sudden and serious degradation in the sampling process. The sampling process of the posture based segmentation can easily be changed by simply adjusting the sampling distance, as explained in the previous chapter. The result is that the posture based approach can achieve extraordinary high compression ratios, but the resulting motion is completely synthetic and unnatural. At low compression ratios, the posture based method suffers severely from quantization errors, and little is gained by using the coder in this range. The gesture based methods are much less sensitive to quantization errors. This is evident in the hybrid technique, where there is not much drop in signal-to-noise ratio for low bit-rates (i.e. coarse quantization).

All three of the model based compression algorithms presented in the previous chapter are quite useful. The posture based method can be used at extremely low bit-rates, provided that synthetic motion is acceptable. The gesture based method also gives excellent compression ratios, and although the motion might not always closely resemble the original, it appears smooth and natural. The compression of the hybrid method is not as substantial, but there is a gain in generality and robustness. These two aspects are a valid concern, especially compared to the gesture based segmentation, which on occasion can fall apart completely for peculiar motion behaviour.

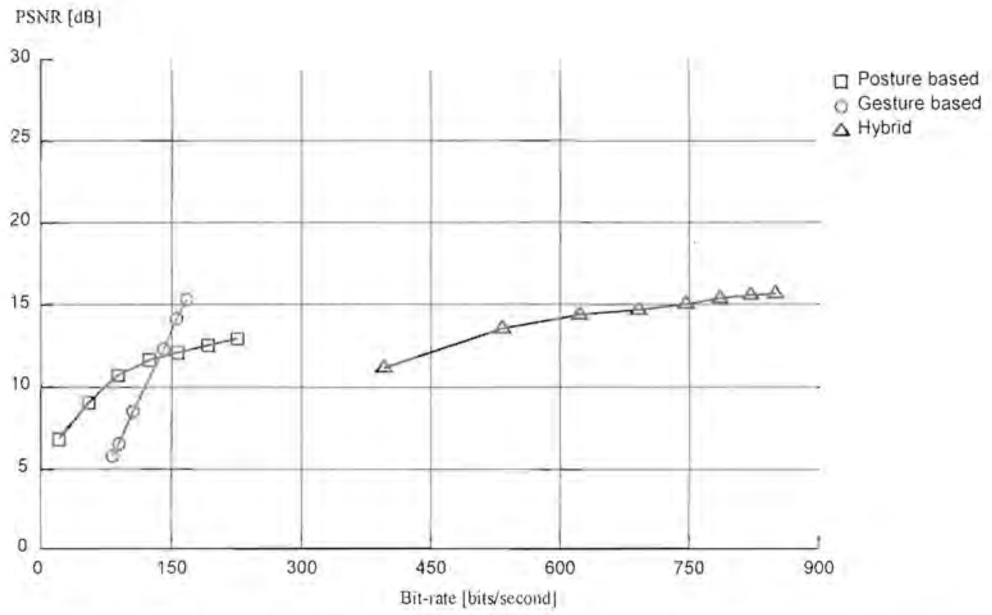

Figure 9-2a: VPSNR vs. bit-rate for the conversational sequence.

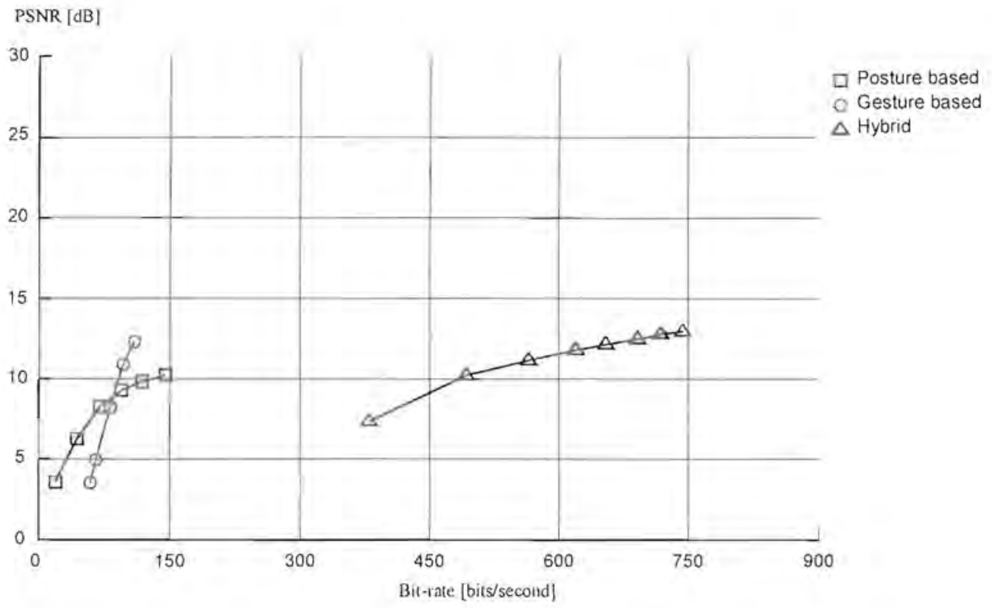

Figure 9-2b: VPSNR vs. bit-rate for the wave sequence.

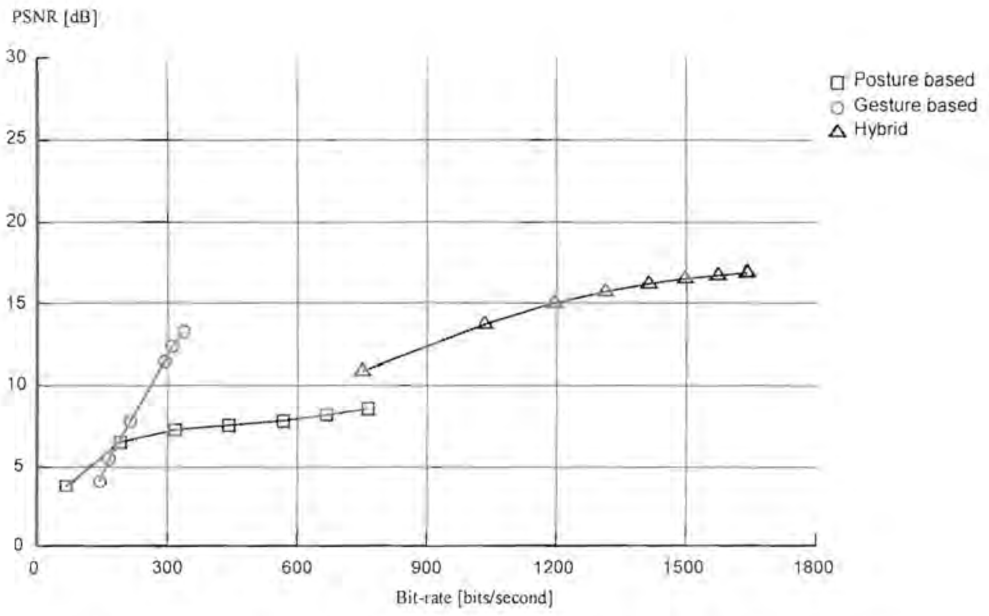

Figure 9-2c: VPSNR vs. bit-rate for the dance sequence.

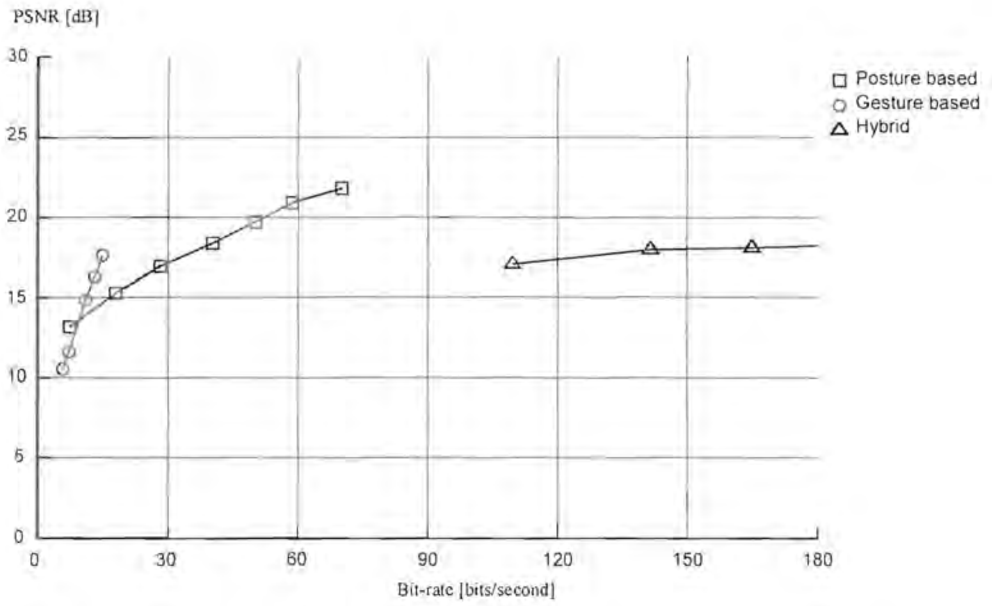

Figure 9-2d: VPSNR vs. bit-rate for the gesture sequence.

Clearly there is much more variation in the bit-rate spread between the various types of motion compared to the waveform coding methods. This is explained by the fact that the non-uniform sampling process or segmentation is highly dependent on the motion characteristics. For both the posture and gesture based methods there are frequent sampling during high activity and much less during low activity. This can clearly be seen for the dance sequence in figure 9-2c, which occupies more than twice the bit-rate range of the preceding graphs. It is possible to adjust the waveform coding algorithms to produce constant rate bit streams, but the model based methods are strictly variable rate techniques, and suggest more complex network protocols and hardware.

9.3 Information entropy

The question "what is the entropy of the signal?" is invariably asked. In 1948, Shannon [60] published the famous paper, "A Mathematical Theory of Information". The foundation of Shannon's work rests on the concept of *entropy*. Briefly, the average information of a source is specified by its entropy, which is defined as

$$H(\theta) = -\sum_{i=1}^{N} P(\theta_i)\log_2 P(\theta_i),$$

where θ is some vector representing motion (in the context of this thesis), such as a temporal sequence or a spatial posture. There are N possible states of θ, denoted by all the θ_i's, each with an *a priori* probability of occurrence $P(\theta_i)$. The entropy definition above conveniently uses a base-two logarithm, and therefore specifies a binary measurement in bits. According to Shannon's "noiseless coding theorem", it is theoretically possible to code a source of entropy $H(\theta)$ bits without distortion using $H(\theta)+\varepsilon$ bits, where ε is an infinitesimally small positive quantity. It is understood that *distortion* is used in the context of information, and it is not necessarily the same as motion or animation distortion.

Unfortunately it is virtually impossible to measure the information of human motion contained in $P(\theta_i)$ and N. This is a problem even in the well defined field of speech analysis, a research area in which posture, gesture and situation level motion information analysis lags far behind. The reason for this difficulty lies primarily in the inability to properly measure error or distortion, and its relation to information loss. However, the following reasoning gives an approximation for the entropy of hand gesture motion.

Consider a set of possible information symbols for a human hand to be made up of open and closed finger positions. We use the average flexure for each finger, which can be represented by five bits of information, i.e. open (binary 1) or closed (binary 0). From informal experiments we have found that a reasonably dexterous person can perform approximately three to four recognizable symbols per second. The entropy is therefore roughly 3*5 = 15 to 4*5 = 20 bits/second. The counting test sequence example that was depicted in figure 5-1c is slightly less frantic, and about 1.5 symbols (or gestures) are produced every second. Finger counting is universally understood, and it is reasonable to assume that no information is lost using only five bits. We therefore conclude that the entropy of the gesture sequence is roughly 1.5*5 = 7.5 bits/second. If this entropy value is compared to figure 9-1d, it is clear that the waveform coding techniques still have a long way to go before the entropy limit is reached. On the other hand, figure 9-2d paints a completely different picture. The gesture based method rapidly approaches the entropy limit, and even so the VPSNR is still quite respectable. Figure 9-3 shows symbols "one" to "four" (starting at the index finger) extracted from the video clip. There is no question about information loss, even if the animated motion is a little synthetic.

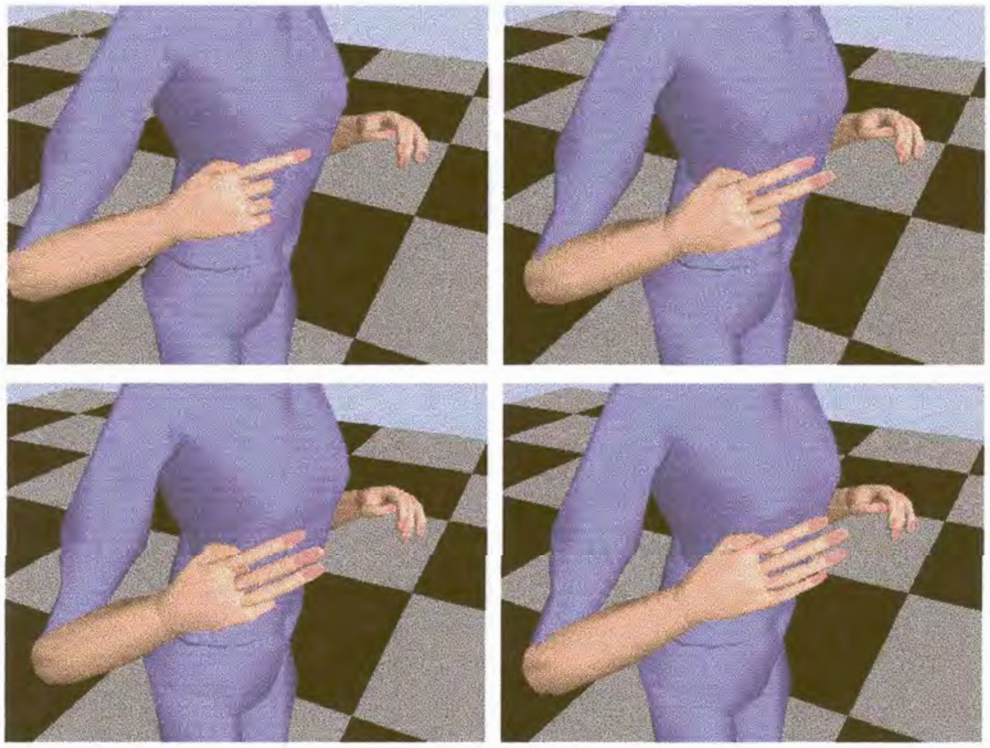

Figure 9-3: Symbols "one" to "four" for the gesture sequence.

Chapter 10 Conclusion

Future telecommunication applications will require the combination, interaction and transmission of natural audio and video streams with synthetic computer graphics models. Of these applications, those that employ aspects of virtual environments populated by virtual humans will play a key role. From the current state of applications it is clear that the insertion of highly detailed and articulated virtual humans into networked systems could place a burden on already taxed network resources. Research on networking virtual humans is still in its infancy. It would still be some time before the state of synthetic audio/visual object compression is on par with equivalent natural audio and video compression methods. Currently the research on facial parameter estimation, compression and transmission is well underway and its implementation in MPEG-4 well defined. The same cannot be said for full body motion methods.

Towards this end we have investigated and developed compression methods applicable to full body motion of virtual humans. A background study that puts the current state of virtual human research into context was presented. A virtual human model was introduced in terms of physical modeling, hierarchical modeling, surface or visual modeling and dynamic modeling. Human motion capture methods that employ inverse kinematic techniques to obtain high degree of freedom joint data, were developed. A detailed motion data analysis was presented, and it was shown that a general statistical model for human motion could be established for specific types of motion that included the properties of ergodicity and being wide sense stationary. The concept of a new visual coding error measurement that exhibits better performance than traditional quantitative methods was introduced. Waveform compression methods were used with good results as general, low complexity coding techniques. The bit-rate reduction that was obtained is comparable with results from research on facial motion done by others. Arguably these techniques are not new, but the successful implementation on a "new" class of motion data is encouraging. Model based compression techniques based on a novel posture and gesture segmentation scheme were developed, and the results exhibited remarkable compression ratios with little loss in information. These techniques show the extreme importance of non-uniform sampling and segmentation methods as a preprocessing step for model based motion compression.

This study has clearly indicated the potential of human motion compression for communication purposes. Without compression, it would barely be possible to insert even two virtual humans into a networked virtual environment if current low-cost network technology is utilized. Using waveform compression techniques this number could be increased by a factor five. Model based compression techniques could increase it even further by a factor fifty or more, provided the loss of naturalness and subtlety is acceptable. This study contributes directly towards the current research of the SNHC group within the MPEG-4 standardization process, and brings the state of full body coding research closer to that of facial coding.

Future research

It is possible to adjust and refine the waveform coding methods to achieve better performance, but it is doubtful whether it will have dramatic effects on bit-rate requirements. Still, subjective testing by a panel of observers is required to optimize some of the coding parameters such as quantizer bit allocations and adaption settings. On the other hand, it is clear that model based methods show enormous potential for further research. Basic methods for segmentation and consequent segment animation were presented. Recent animation editing and transition methods based on combinations of motion capture, kinematic constraints and dynamic simulation can substantially improve the synthetic "feel" of these segment animation techniques. The use of higher level motion detection algorithms could also improve the non-uniform sampling methods and lower the bit-rate even more. However, care should be taken not to accidentally exceed the entropy limit (supposing that it can be measured). In human speech coding, intelligibility is clearly defined, even without the original speech. In contrast, it is possible to produce "coded" human motion that does not resemble the original motion, yet it appears credible and natural. Lastly, further research can also benefit knowledge of the effects and implications of the variable coding delay and variable bit-rate characteristics of model based coding on real-time virtual environments.