Chapter 7 Waveform coding

7.1 Introduction

Waveform coding implies algorithms and methods that focus on single variables, such as body position and joint angles. No knowledge of the actual action that the figure is performing (such as walking, waving etc.) is assumed, and the exact source of the motion is also not under consideration, only that it is valid human motion. Whether the motion is captured in real-time, or generated by synthetic animation techniques, is of no concern. It is assumed that all of the body parameters can be decomposed into single DOF values that are independent of each other. An exception to this is the spatial vector quantization method presented at the end of the chapter, where it is assumed that there is a correlation between the variables.

In general a distinction can be made between coding (or compression) in the temporal domain and coding in the spatial domain. These two domains can be seen as orthogonal[1] to each other, and it is often advantageous to *combine* methods from each domain to get maximum compression. Temporal coding techniques take advantage of the temporal correlation of a single variable, while spatial techniques take advantage of spatial correlation between several variables.

Another distinction that can be made is the concept of *uniform* vs. *non-uniform* sampling. Traditionally we have become accustomed to sampling, frame or simulation updates that occur at well specified, regular intervals. However, there are many random processes in nature that need not be discretized in such a way, of which human motion is probably one. As has been reported by [45] for head orientations, human movement remains relatively

[1] Orthogonal is used not in a strictly mathematical sense.

static except for occasional bursty moments. The speed and acceleration of the movement are non-zero only during these moments of erratic actions. It is therefore natural to use a lower sample rate during slow movements and to increase the rate proportionally to faster movements. The solution of exactly *how* to do this is not very obvious. One such example is the dead reckoning algorithm, which is discussed a little later in this chapter. In the next chapter on model based coding, the use of non-uniform sampling will be discussed more extensively.

7.1.1 Compression

Compression is defined as the procedure that takes a stream of input samples $\{\theta(n)\}$, and transforms them to a finite string of codes or messages $\{c(n)\}$ that is a compressed version of the input stream. Decompression is the procedure that takes the string $\{c(n)\}$ and converts it to an equivalent output stream $\{\hat{\theta}(n)\}$. If the output stream is an exact or very similar duplicate of the input stream, the compression scheme is lossless. Lossy compression schemes introduce a controlled amount of distortion in the output stream in exchange for better compression. All of the compression methods discussed in this chapter are of a lossy nature (except statistical coding, but it is never used on its own).

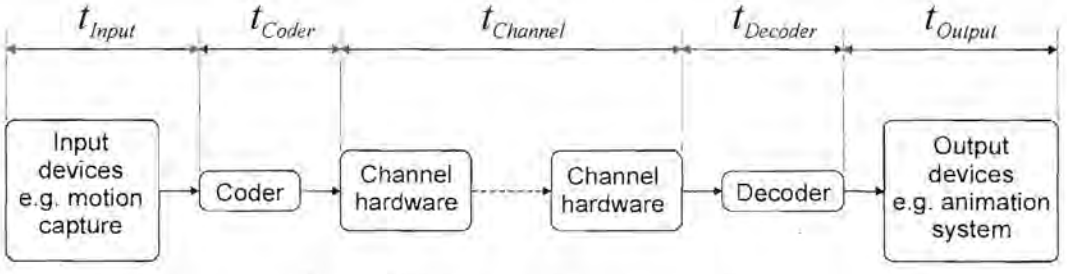

Figure 7-1: Coding/decoding delay.

7.1.2 Delay

Figure 7-1 shows a generic layout of a human motion coding/decoding system. The total coding delay is the lapsed time from the execution of an action until reconstruction at the receiving end, and is given by

$$t_T = t_{Input} + t_{Coder} + t_{Channel} + t_{Decoder} + t_{Output}.$$

Of these, we are not concerned with t_{Input}, $t_{Channel}$ and t_{Output}, which are the delays for the input devices, communications channel and output devices respectively. The time t_{Coder} is the time the coding or compression step takes from input to output, and similarly is $t_{Decoder}$ the time the decoding or decompression step takes. The delay $t_{Channel}$ is often quite severe, but there is usually not much that can be done about that. The values of t_{Coder} and $t_{Decoder}$ should be kept as low as possible – every little bit helps. Unfortunately, we are committed to a discrete sampling system at the input stage. The more coder delay we allow, the more samples we have to work with and the better information estimation we can get. When comparing compression results with the original motion, it is often convenient to compensate for the coding delay in order to use convenient error measures such as the MSE. However, it is still important to properly define the effects and tolerability of delay, especially when different systems are compared. Some systems can be used for off-line storage purposes, while others are more suitable for real-time interactive applications. In this thesis, we are more interested in the latter, hence more attention will be given to such systems.

The rest of this chapter discusses various methods for human motion compression. We start with the definition of quantization and statistical coding. Both of these methods are not really used on their own, but are "building blocks" for other compression algorithms. We then look at the class of predictive and adaptive predictive coders. This is followed by a DCT coding method as an example of a frequency domain algorithm. Vector quantization is difficult to classify as a waveform coding technique, since it can be used temporally or spatially or both, but is presented at the end of this chapter anyway. Typical results are presented with each method in the form of a representative DOF for each test sequence, as well as an overall rate-distortion graph. The following DOFs were arbitrarily chosen:

- The head angle $\theta_{3,0}$ for the conversational sequence,
- The left upper arm elevation angle $\theta_{5,0}$ for the wave sequence,

- The left elbow hinge angle $\theta_{6,0}$ for the dance sequence,
- The right hand index finger flex angle $\theta_{27,1}$ for the gesture sequence.

In each case, the rate-distortion graph shows the average PSNR (defined in chapter 6) for the whole body, except for the gesture sequence, which shows *only* the results for the right hand. The effective bit-rate is therefore considerably lower. The PSNR is used instead of the VPSNR measure for waveform coding techniques in order to compare the results with related work done by others [36]. In the following chapter on model based coding, it will be seen that the PSNR fails to give a good error measure and the VPSNR will be used instead. As a general rule-of-thumb, we have found that a PSNR of roughly 20–30 dB is visually acceptable. A PSNR of less than 10 dB is considered completely unacceptable, and a PSNR of more than 40 dB is considered almost lossless. In chapter 5, the undistorted bit-rate requirement for the whole body was found to be roughly 15000 bits/second. A compression method is regarded as useful when it can reduce the information by at least a factor two while still maintaining an acceptable error level. Any method that exceeds 8000 bits/second is therefore regarded as not worth the effort. The undistorted bit-rate for the right hand alone is roughly 2500 bits/second, and rates of less than 1250 bits/second are regarded as useful.

The results shown in this chapter are but a very small subset of the complete human due to space limitations. Representing a DOF graphically as a time varying signal is also not very intuitive, but that is the best that can be done on paper. On occasion a rendered sequence of the human figure is shown, but the results are best viewed using the video clips provided on CD-ROM with this document. Appendix III describes the contents of the accompanying CD-ROM, as well as the parameters used for each coding algorithm.

7.2 Quantization

Quantization is the mapping of a variable θ to an approximated variable $\hat{\theta}$, $\hat{\theta} = Q(\theta)$, where Q is some sort of quantization function. It can be described as the process of comparing a real value $\theta \in \mathbf{R}$ to a set of decision levels d_i and a set of reconstruction

levels r_i, where i is a *finite* integer. The problem entails the specification of a set of decision levels and reconstruction levels such that if

$$d_i \leq \theta < d_{i+1}, \tag{7-1}$$

the input variable is quantized to the reconstruction value r_i. The decision and reconstruction levels are chosen to minimize some error measure between θ and $\hat{\theta}$. An example of a mathematically tractable measure is the mean-square error, and is often used. If θ is seen as a random variable, then for N quantization levels the mean-square error is

$$\varepsilon = E\{(\theta - \hat{\theta})^2\} = \sum_{i=0}^{N-1} \int_{d_i}^{d_{i+1}} (\theta - r_i)^2 p(\theta) d\theta, \tag{7-2}$$

where $p(\theta)$ is the probability density function (PDF) of θ. It can be shown that the optimum placement of r_i can be found by minimizing ε with respect to r_i, and is given by

$$r_i = \frac{d_i + d_{i+1}}{2}, \tag{7-3}$$

which is the midpoint between each pair of decision levels. Finding the optimum choice of decision levels d_i involves the minimization of ε with respect to d_i. This is rather involved and requires knowledge of the probability density function $p(\theta)$. Max [48] developed a solution for optimum levels of a Gaussian distribution, and it can be extended to include uniform, Laplacian and Rayleigh densities. Calculation of the probability density function of human motion is virtually impossible due to the wide variation in human physiology and human motion. We will look at the more general case of uniform quantization and non-uniform quantization, and the minimum parameters that define each.

7.2.1 Uniform quantization

Uniform quantization is applicable to variables with a uniform PDF. This is mostly the case if quantization is to be directly applied to joint angles (see chapter 4). The parameters that define a uniform quantizer are the lower and upper limits, denoted by θ_L and θ_U respectively, and the number of quantization steps N. This means that the input variable θ must be restricted to $\theta_L \leq \theta \leq \theta_U$. For practical purposes N should also be restricted to a power of two, since we do not want to deal with split bits in an output bit stream. In this case N will be an even number, and the quantizer can be designed to be symmetric or asymmetric in the case of a bipolar system. If quantization is to be applied directly to joint angles, the joint limits define the lower and upper limits as well as the symmetry. It is a good idea to have separate quantizers for variables with radically different limits. If some other quantity is to be quantized, the defining parameters should be known or calculated. The generalized equation for decision level i is given by

$$d_i = \theta_L + \frac{i(\theta_U - \theta_L)}{N}, \tag{7-4}$$

and the reconstruction is given by equation (7-3). The term

$$\frac{\theta_U - \theta_L}{N}$$

in equation (7-4) is the *step size* of the quantizer and is often denoted by Δ.

7.2.2 Non-uniform quantization

Non-uniform quantization will be applied to variables with non-uniform PDFs. The spacing of decision levels is narrow in large amplitude regions of the PDF and widens in low amplitude portions of the PDF. Other than that not much can be said about the exact mathematical expression for the decision levels. There are a number of non-linear functions that can be used to generate an appropriate quantizer. Popular examples are the

A-law and μ-law quantizers used in speech coding. We have found that the bipolar μ-law quantizer gives good results, and can easily be adjusted to match a variety of non-uniform PDFs, such as the Laplace density that is encountered in predictive or difference coding. The basic parameters to specify such a non-linear quantizer are the value of μ (a "measure" of the non-linearity), the maximum bipolar limits θ_{MAX} and the number of steps N. The generalized equation for the positive half (i.e. $\theta > 0$) ith decision level is given by

$$d_i = \frac{\theta_{MAX} \log\left(1 + \mu \frac{i}{(N/2)}\right)}{\log(1+\mu)}, \tag{7-5}$$

and the reconstruction is given by equation (7-3). The negative half is a mirror of the positive half.

Finding the reconstruction level r_i given an input θ is trivial in the case of a uniform quantizer, and involves the conversion of θ to the integer space of i using simple multiply and add operations. The same cannot be said for a non-uniform quantizer, and some search algorithm has to be implemented. We use a recursive binary method, where the input level θ is compared with the midpoint of two decision levels, and a choice between the left or right branch is made.

7.2.3 Quantization noise

A useful mathematical concept is that of quantization noise, i.e. a measure that indicates the amount of distortion introduced by the quantizer. By "noise" we mean *visual* noise, and not the more traditional term of audible noise. Severe quantization noise is much more offensive in the visual sense compared to audible noise, and can render some compression algorithms completely useless. When analyzing quantization noise, it is useful to represent the quantized samples as

$$\hat{\theta}(n) = \theta(n) + e(n), \tag{7-6}$$

where $e(n)$ is the quantization noise or error. To study the effects of quantization noise, and in order to solve certain mathematical equations, it is convenient to assume a simple statistical model for the quantization noise:

- The quantization noise is a stationary white noise process, i.e.

$$E[e(n)e(n+m)] = \sigma_e^2, \quad m = 0$$
$$= 0, \quad otherwise$$

- The quantization noise is uncorrelated with the input signal, i.e.

$$E[\theta(n)e(n+m)] = 0, \quad \forall m$$

- The quantization error distribution is uniform over each quantizer interval Δ.

Although these assumptions are unrealistic for some types of human motion, our experiments have shown that it is reasonable for a step size Δ that is small enough.

Quantization can be seen as a compression technique, since the quantized output usually occupies fewer bits than the original signal for a given error in representation. Figures 7-2a to 7-2d show the results for direct quantization of the representative joint angles discussed previously. Shown are the original, 8-level quantized, 64-level quantized and the error signal of the 64-level quantization. Using less than 64 levels (or 6 bits) usually results in severe visual artifacts, except for the interesting case shown in figure 7-2d, where the open/close gesture movements can be quantized quite well with very few levels. In any case, direct quantization of DOF values results in an effective compression ratio below 2:1, and such a naive method is not recommendable as a compression mechanism. Figure 7-2e depicts a number of consecutive 3D wireframe images from the dance sequence. The original is overlaid with a 16-level quantized sequence (shown in red), and the frame-to-fame difference can clearly be seen.

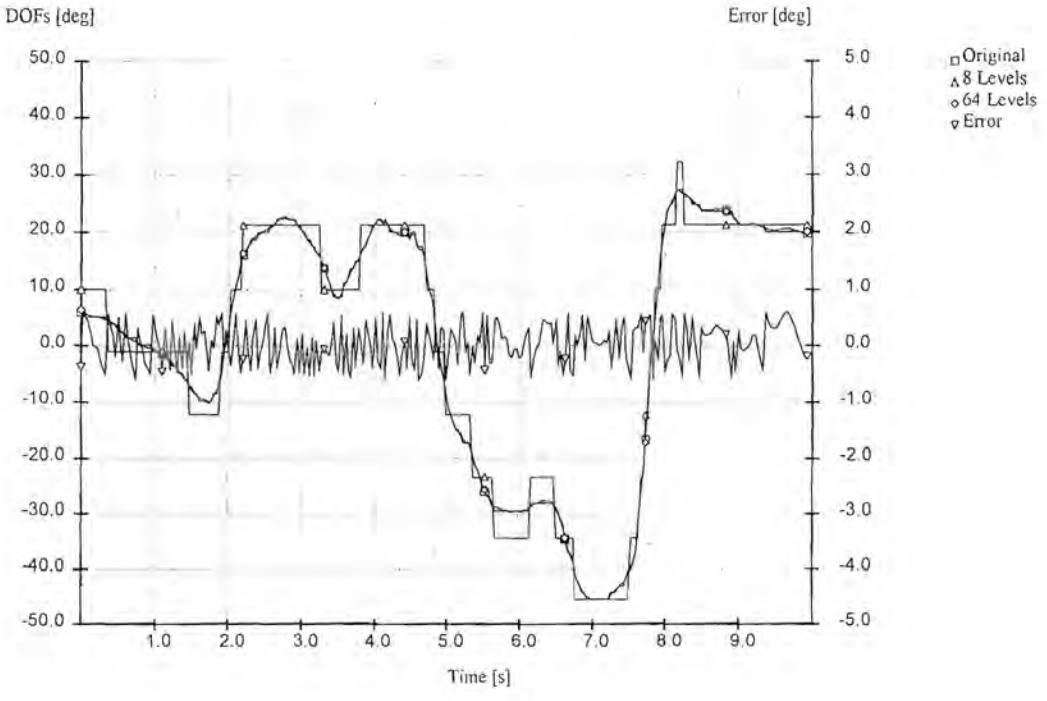

Figure 7-2a: Quantization of $\theta_{3,0}$ with 8 and 64 levels for the conversational sequence.

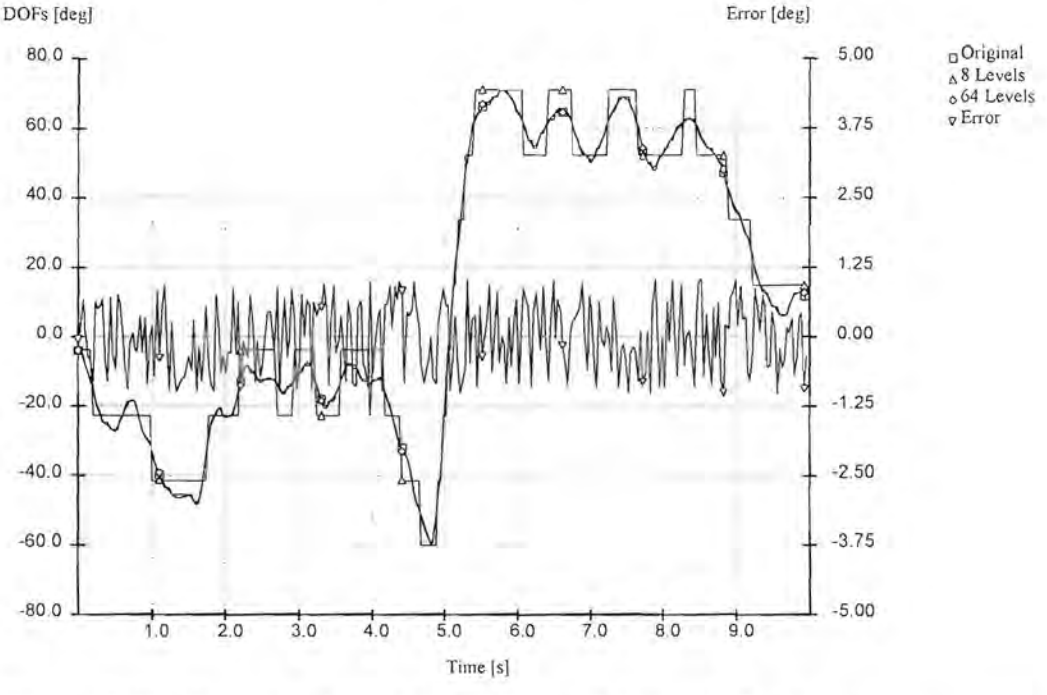

Figure 7-2b: Quantization of $\theta_{5,0}$ with 8 and 64 levels for the wave sequence.

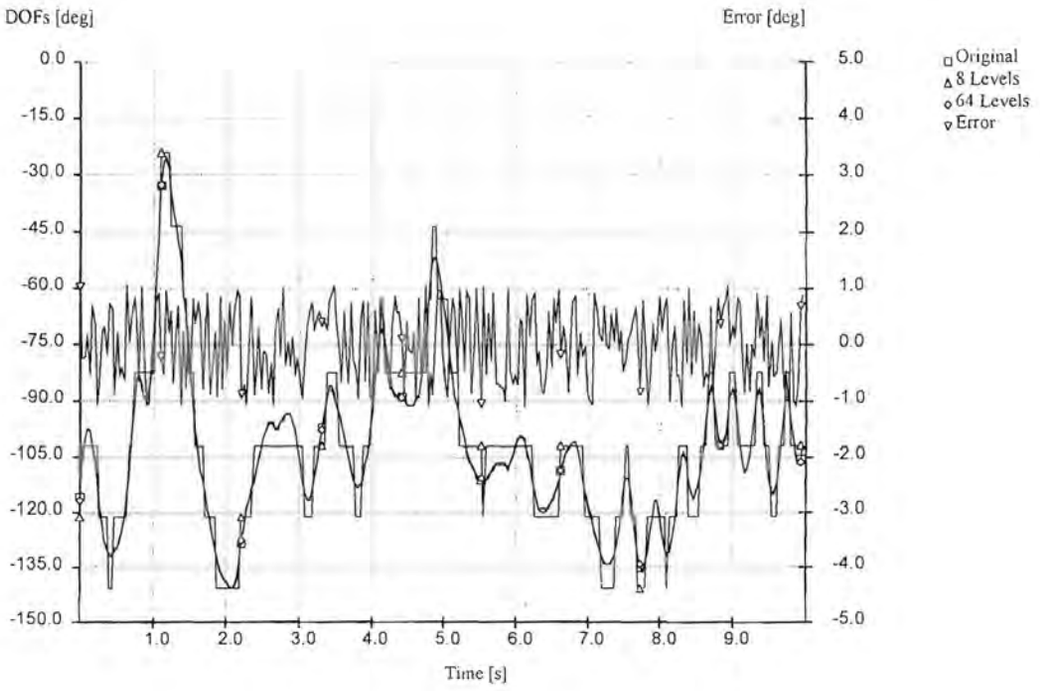

Figure 7-2c: Quantization of $\theta_{6,0}$ with 8 and 64 levels for the dance sequence.

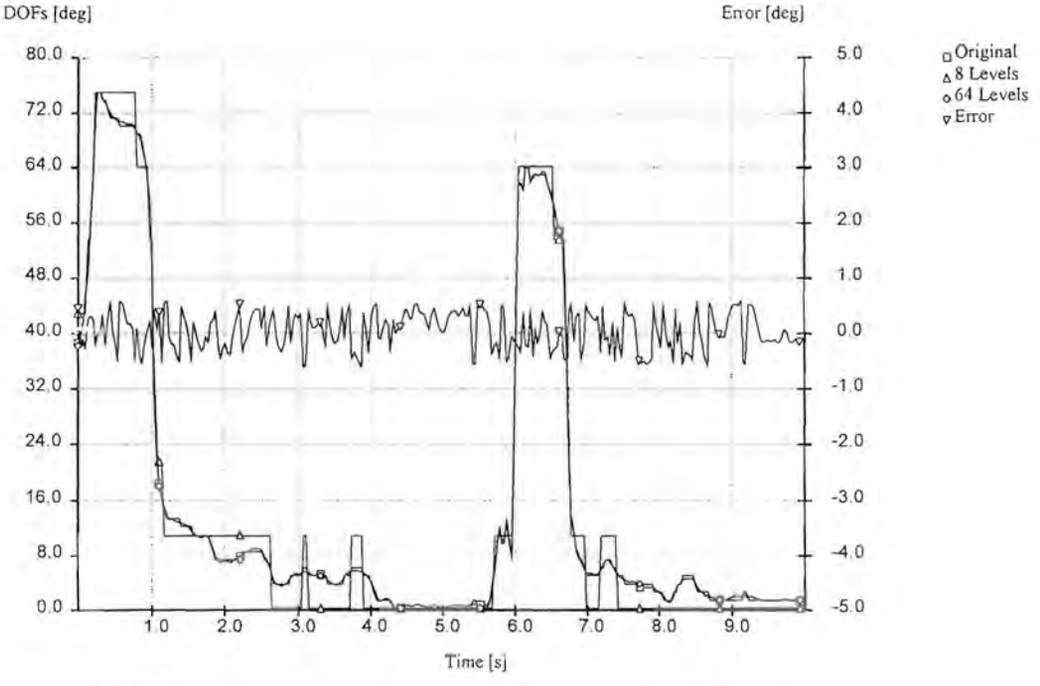

Figure 7-2d: Quantization of $\theta_{27,1}$ with 8 and 64 levels for the gesture sequence.

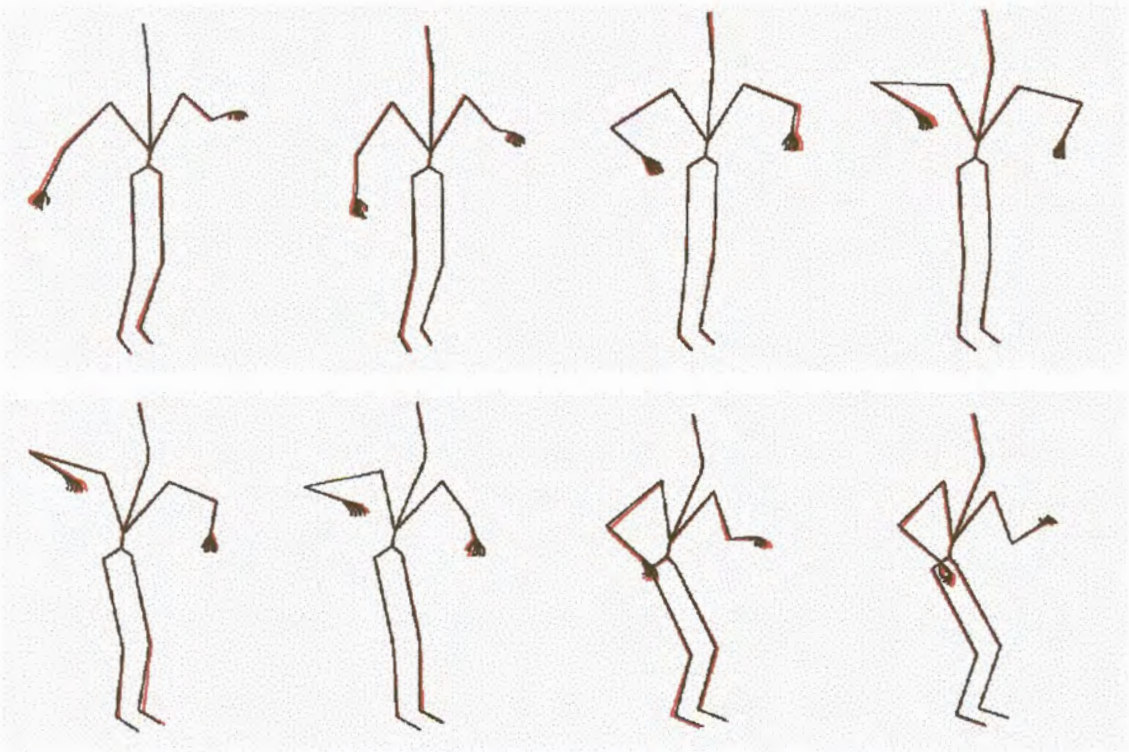

Figure 7-2e: Wireframe images from the dance sequence with 16-level quantization.

Figure 7-3 shows the PSNR against bit-rate. Note that the effective bit-rate and error for the gesture sequence is for the right hand only. The other body DOFs contained almost no information, and was not compressed.

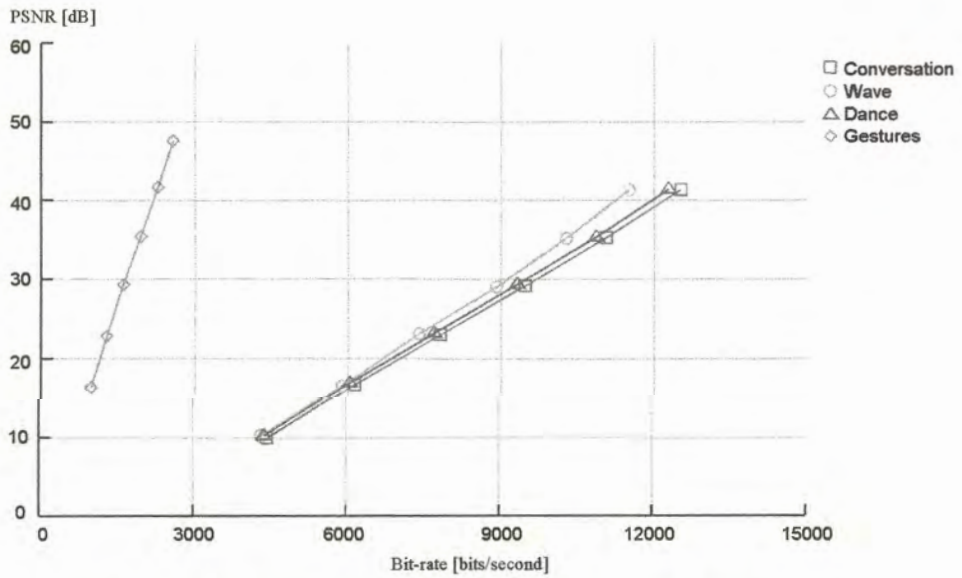

Figure 7-3: Distortion vs. bit-rate for direct quantization.

7.3 Adaptive quantization

One is often confronted with the dilemma of choosing the correct quantization step size Δ. On the one hand it should be large enough to accommodate the maximum peak-to-peak range of the input signal. On the other hand it should be small enough to minimize quantization noise. One way of alleviating this problem is to use non-linear quantization, while the other is to adapt the quantizer to some property of the input signal.

The basic idea of adaptive quantization is to let the quantizer levels and ranges vary to match the variance of the input signal, or alternatively to adjust the gain of the input signal inversely with the variance of the input signal. There are two methods of doing this. A *feed-forward* scheme estimates the matching function from the input itself. A feedback scheme estimates the matching function from the output of the quantizer (or even the whole coding system). Feed-forward systems require us to transmit the quantizer settings as well (albeit only every nth update), while the feedback system can use the received messages to derive the quantizer settings.

For simplicity, we have chosen a simple feedback algorithm where the step size $\Delta(n)$ of a uniform quantizer is modified at update n by a function of the form

$$\Delta(n) = \beta\Delta(n-1), \tag{7-7}$$

where β is a step size multiplier and is a function of the previous code $c(n-1)$. In practice, we use a table containing values of β for each code word. These values have been obtained in a heuristic fashion to accommodate a large variety of input signals. Direct adaptive quantization of DOF variables is not recommended because the joint angles are generally non-zero-mean quantities, and do not exhibit symmetric behaviour. It is difficult under such circumstances to establish a proper adaptation table for β, and the performance of the adaptive scheme approaches that of the standard quantization method discussed in the

previous section. Adaptive quantization will be used extensively in the more advanced compression techniques discussed below.

7.4 Statistical coding

Although statistical coding (sometimes referred to as entropy coding) is not a lossy compression technique in itself, it is usually inserted at the end of a lossy compression pipeline to ensure that the stream of codes have optimal statistics. As a starting point for statistical coding development, it is necessary to model, estimate or measure the probabilities of occurrence for each value to be encoded. In our case, we use statistical coding after some other coding technique, and this measurement will be done in "message space" rather than in joint angle space. Most often a quantizer is superseded by a statistical coder. In this case, suppose that the probability of a quantized value $\hat{\theta}$ (or message) to be equal to the nth reconstruction level, is given by

$$P(n) = P\{\hat{\theta} = r_n\}. \tag{7-8}$$

In the coding process, a code word of $b(n)$ bits is assigned to each quantization level, resulting in an average code length of

$$L = \sum_{n=0}^{N-1} P(n)b(n), \tag{7-9}$$

where N is the length of the code book. There are a number of techniques that can be used to produce a codebook [71,72,73]. These include arithmetic coding, Shannon-Fano coding and Huffman coding, of which the latter is the most efficient in terms of length. In this coding process [71], the two messages with the lowest probability are combined in a tree structure and their probabilities summed at the junction. The probability is then combined again in the same manner with the next lowest probability until the tree converges to a single junction. The branches of the tree are then assigned arbitrarily bit values of one or

zero. A code is formed by traversing the tree back to the message node in question and recording the path designation.

It is possible to calculate a fixed codebook beforehand, or to adaptively build the codes as the transmission progresses. In the latter case, we start with a codebook of equal length codes. For every message sent the probabilities of the codes are updated and the codebook is calculated according to the method discussed above. For a fixed codebook, the probabilities can be calculated using an appropriate test data set.

Most of the compression algorithms in the remainder of this chapter use a statistical coder as a "black box" between the coder and decoder sections. The statistical coder can never increase the average code length, and can have no adverse effects on the coding process if used correctly. However, a properly implemented compression algorithm should not rely on the use of a statistical coder to achieve high compression ratios. In fact, more than a 20-30 percent decrease in average code length is an indication that the code words from the output of the compression algorithm have a non-uniform distribution. This implies that the compression algorithm is probably poorly designed, and that further gain can be achieved with a better implementation.

7.5 Predictive coding

In chapter 5, it was shown that there is considerable correlation between adjacent samples. On average, joint angles do not change rapidly from sample to sample, therefore the difference between adjacent samples should have a lower variance than the original signal itself. Figure 7-4 depicts the general layout of a *predictive coder*. The dotted lines indicate an adaptive section, and can be ignored for now. The input to the quantizer is a difference signal

$$d(n) = \theta(n) - \tilde{\theta}(n), \tag{7-10}$$

where $\tilde{\theta}(n)$ is a *predicted* version of the input signal $\theta(n)$. If the prediction is good, the variance of $d(n)$ will be smaller than that of $\theta(n)$, and the quantizer could be adjusted to give a smaller quantization error for a fixed number of levels. Figure 7-4 also shows the layout of a corresponding decoder. The output is given by

$$\hat{\theta}'(n) = \tilde{\theta}'(n) + \hat{d}'(n), \tag{7-11}$$

where $\tilde{\theta}'(n)$ is the output of a similar predictor as in the coder. Clearly if $c'(n) = c(n)$, then $\hat{\theta}'(n) = \hat{\theta}(n)$, and the only difference between the input and output is the quantization error incurred in $d(n)$.

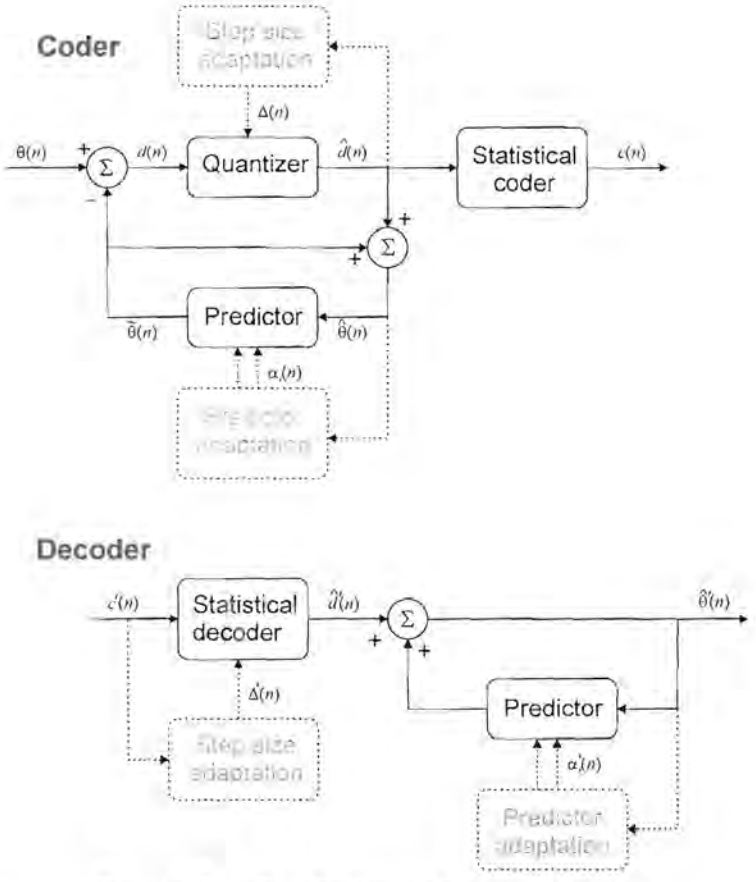

Figure 7-4: Predictive coder and decoder.

The predictor could have a variety of forms. A mathematically tractable and widely used form is a linear predictor [72,73], i.e. the output is a linear combination of past input values. The general form of the predictor can be written as

$$\tilde{\theta}(n) = \sum_{k=1}^{p} \alpha_k \hat{\theta}(n-k). \tag{7-12}$$

Since we would like to minimize the variance of $d(n)$, as denoted by σ_d^2, it would be appropriate to differentiate σ_d^2 with respect to each coefficient α_i:

$$\frac{\partial \sigma_d^2}{\partial \alpha_i} = 0, \quad 1 \le i \le p. \tag{7-13}$$

Finding an exact solution for equation (7-13) is quite involved and requires extensive knowledge about the input signal. In [73] a number of approximations are discussed. It has been found that not much is gained with high order predictors, and that it is best to keep $p < 4$. For comparison purposes, we use a first order predictor in this section, and a higher order adaptive predictor in the next section. In the case of $p = 1$, it can be shown [73] that

$$\alpha_1 = \frac{R_\theta(1)}{R_\theta(0)}, \tag{7-14}$$

where R_θ is the autocorrelation function of θ.

Figures 7-5a to 7-5d show the results for predictive coding of the representative joint angles. Indicated are the original, 8-level quantized, 64-level quantized and the error signal of the 8-level quantization. Even with as little as 4 quantization levels (not shown), the coder still provides acceptable results. It can be seen that in most cases the 8-level error signal is similar to that of 64-level direct quantization, which is a saving of almost 3 bits. The use of 64-level (or 6 bit) quantization results in motion that is almost indiscernible from the original.

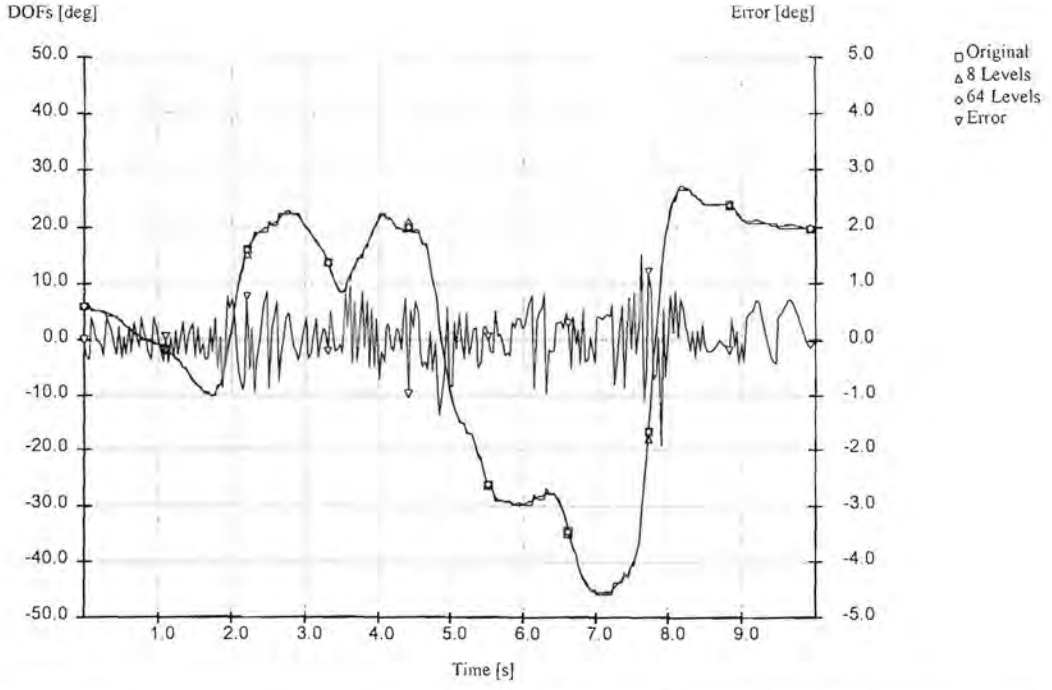

Figure 7-5a: Predictive coding of $\theta_{3,0}$ with 8 and 64 levels for the conversational sequence.

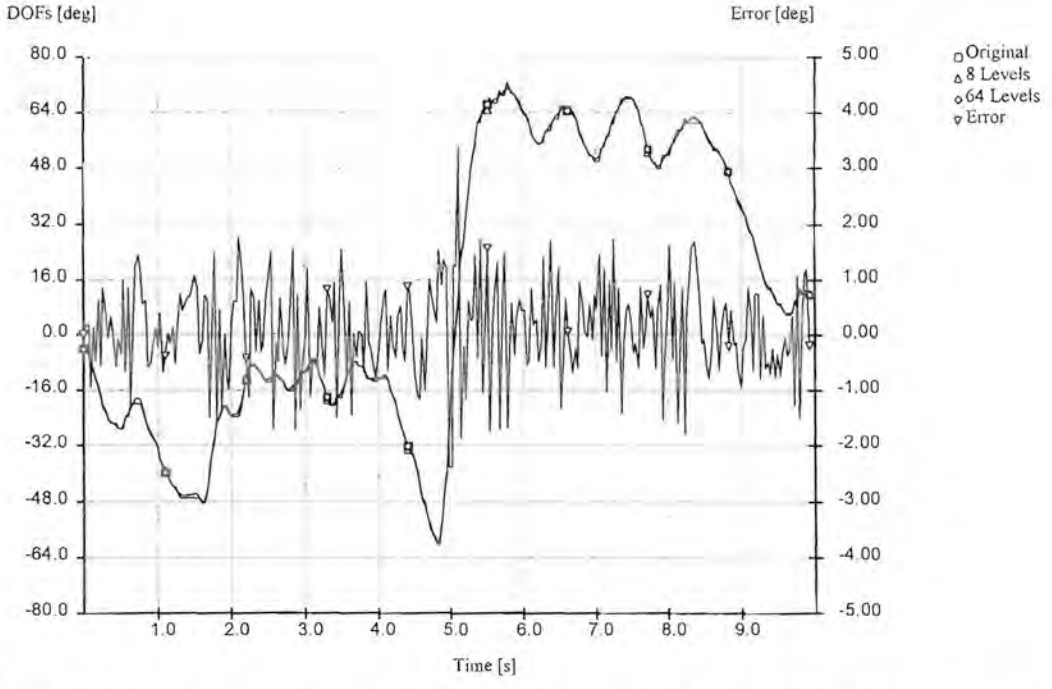

Figure 7-5b: Predictive coding of $\theta_{5,0}$ with 8 and 64 levels for the wave sequence.

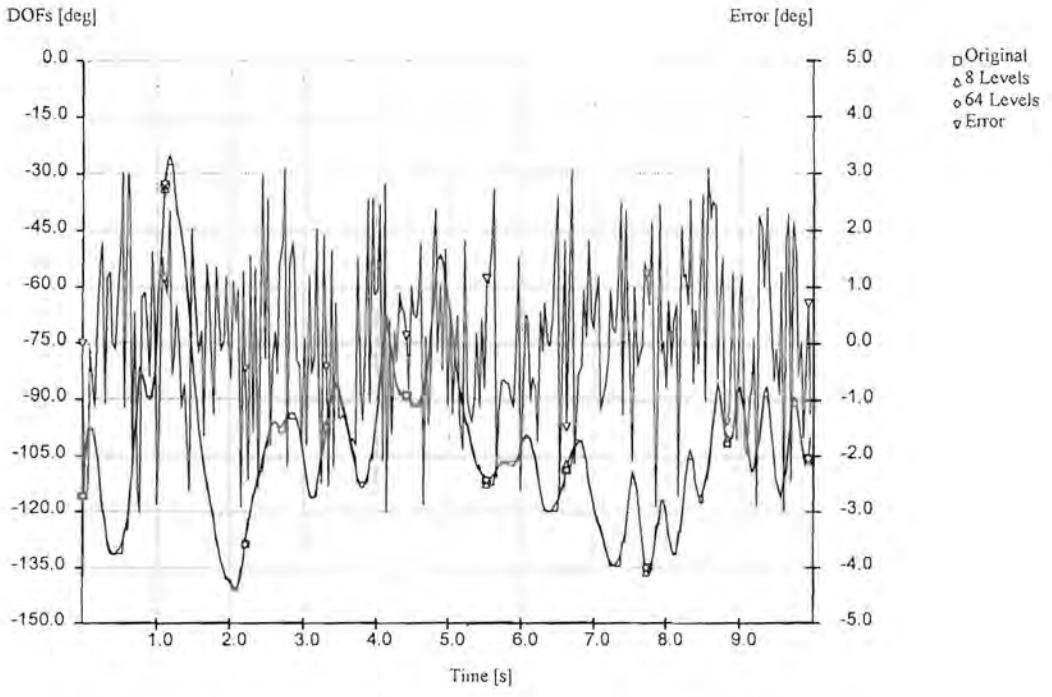

Figure 7-5c: Predictive coding of $\theta_{6,0}$ with 8 and 64 levels for the dance sequence.

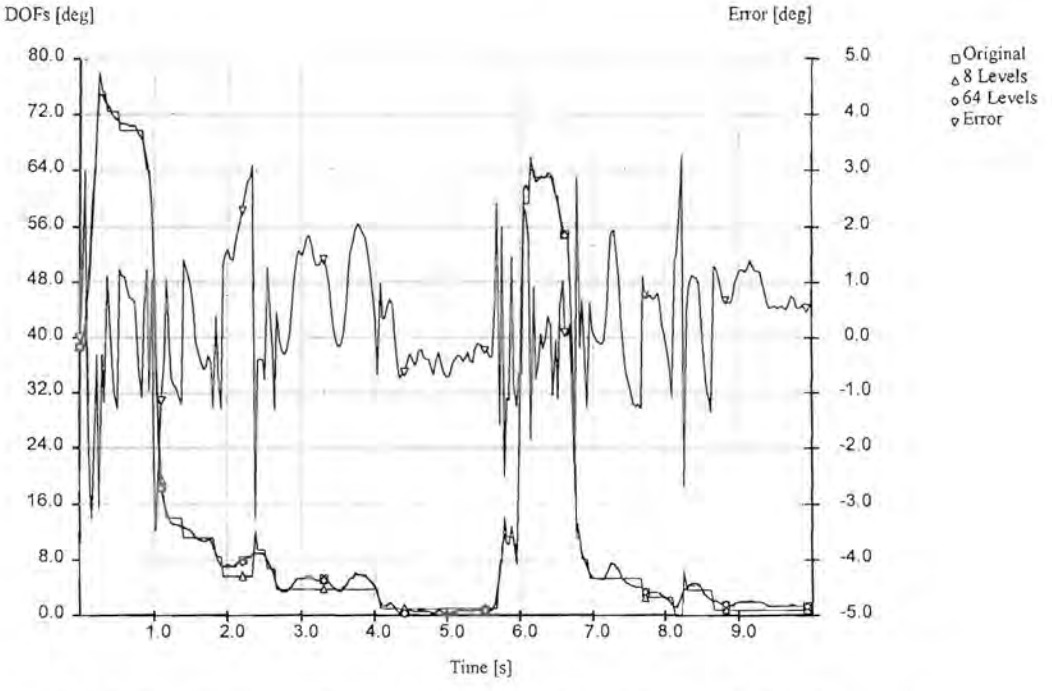

Figure 7-5d: Predictive coding of $\theta_{27,1}$ with 8 and 64 levels for the gesture sequence.

Figure 7-6 shows the PSNR against bit-rate for predictive coding. Note that the effective bit-rate and error for the gesture sequence are for the right hand only. The saving over

direct quantization can clearly be seen (chapter 9 contains a comparison between various coding methods for the same sequence). The graphs are also more spread out relative to each other, which indicates that the coding method is sensitive to temporal variation, which direct quantization is not. It can be seen that predictive coding, for all practical purposes, becomes lossless for more than 257-level quantization. However, the practical range for this method lies between 3000 and 6000 bits/second.

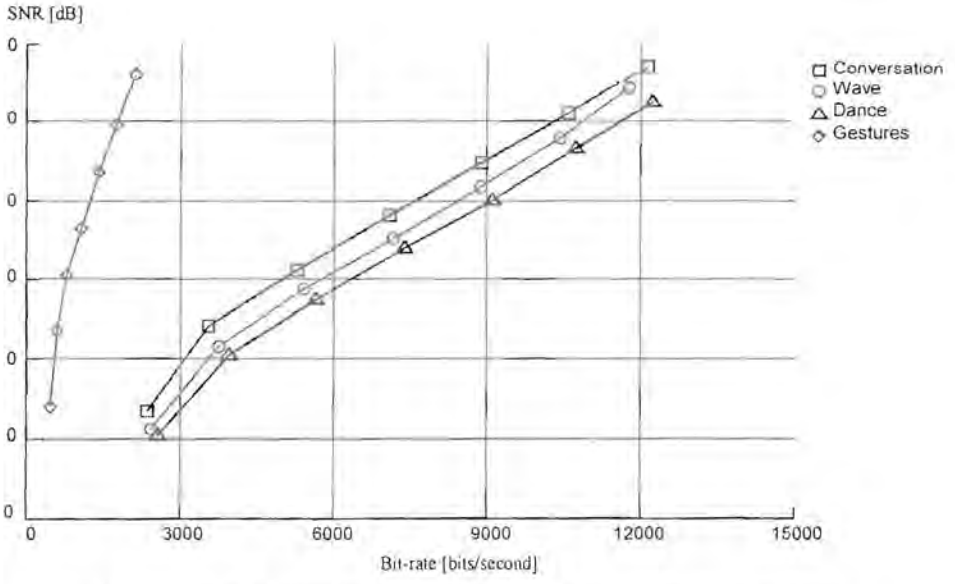

Figure 7-6: Distortion vs. bit-rate for predictive coding.

7.6 Adaptive predictive coding

The structure of an adaptive predictive coder is very similar to that of a predictive coder, and it shares the same layout as shown in figure 7-4. However, in the case of an adaptive predictive coder it is important to note that either the quantizer, or the predictor, or both are adapted to give an improved output $\hat{\theta}(n)$. Note that the scheme shown in figure 7-4 is a feedback system, i.e. the adaptation parameters are calculated from the decoded signal. The concept of adaptive quantization was already covered in section 7.3. It is natural to consider adapting both the quantizer and predictor to match the temporal variations in the human motion signal. Adaptive prediction implies that the prediction coefficients $\{\alpha_i\}$ are

now functions of the current sample or update, i.e. they should be written as $\{\alpha_l(n)\}$. The predictor function of equation (7-12) now becomes

$$\tilde{\theta}(n) = \sum_{k=1}^{p} \alpha_k(n)\hat{\theta}(n-k). \tag{7-15}$$

Similar to speech signals, the prediction coefficients could be chosen to minimize the average mean-squared prediction error over short time intervals. Again there are numerous approaches and methods to solve this problem, such as the autocorrelation and covariance methods [73]. By evaluating equation (7-13) for short motion segments and omitting the effects of quantization noise, it can be shown that the autocorrelation method provides a set of equations that can be written in matrix form

$$\mathbf{A}\boldsymbol{\alpha} = \mathbf{B}, \tag{7-16}$$

where $\mathbf{A}$ is a $p \times p$ matrix of autocorrelation values

$$\mathbf{A} = \begin{bmatrix} R_\theta(0) & R_\theta(1) & R_\theta(2) & \dots & R_\theta(p-1) \\ R_\theta(1) & R_\theta(0) & R_\theta(0) & \dots & R_\theta(p-2) \\ R_\theta(2) & R_\theta(0) & R_\theta(0) & \dots & R_\theta(p-3) \\ \dots & \dots & \dots & \dots & \dots \\ R_\theta(p-1) & R_\theta(p-2) & R_\theta(p-3) & \dots & R_\theta(0) \end{bmatrix}, \tag{7-17}$$

and $\mathbf{B}$ is a $p \times 1$ vector of autocorrelation values

$$\mathbf{B} = \begin{bmatrix} R_\theta(1) \\ R_\theta(2) \\ R_\theta(3) \\ \dots \\ R_\theta(p) \end{bmatrix}. \tag{7-18}$$

The matrix **A** has a Toeplitz structure, and the solution $\alpha = \mathbf{A}^{-1}\mathbf{B}$ can be computed using a variety of numerical methods. In equations (7-17) and (7-18), the term $R_\theta(n)$ refers to the *short-term* autocorrelation function of the sequence $\{\theta(n)\}$, which can be calculated from a windowed segment of motion. The choice of window and window length depends on the characteristics of the DOF in question and the type of motion. We use window lengths of between 0.5 seconds (15 samples at 30 Hz) and 4 seconds (120 samples at 30 Hz).

We have found that not much is gained for values of p greater than 2, and the following results were obtained using a second order adaptive predictor, together with an adaptive quantizer. Figures 7-7a to 7-7d show the results for adaptive predictive coding of the representative joint angles. Depicted are the original, 8-level quantized, 64-level quantized and the error signal of the 8-level quantization. It can be seen that the error for an 8-level quantizer is worse than that of simple first order non-adaptive prediction, a fact that is also evident on the rate-distortion graph of figure 7-8. The sharp increase in error at very low bit-rates is due to the omission of quantization noise effects in the calculation of the adaptation coefficients (the difference signal is severely quantized to 4 levels at these low rates). At low rates the system also exhibits oscillatory quantization behaviour, which can clearly be seen in figure 7-7d. For higher bit-rates, the adaptive coding scheme clearly outperforms the non-adaptive techniques.

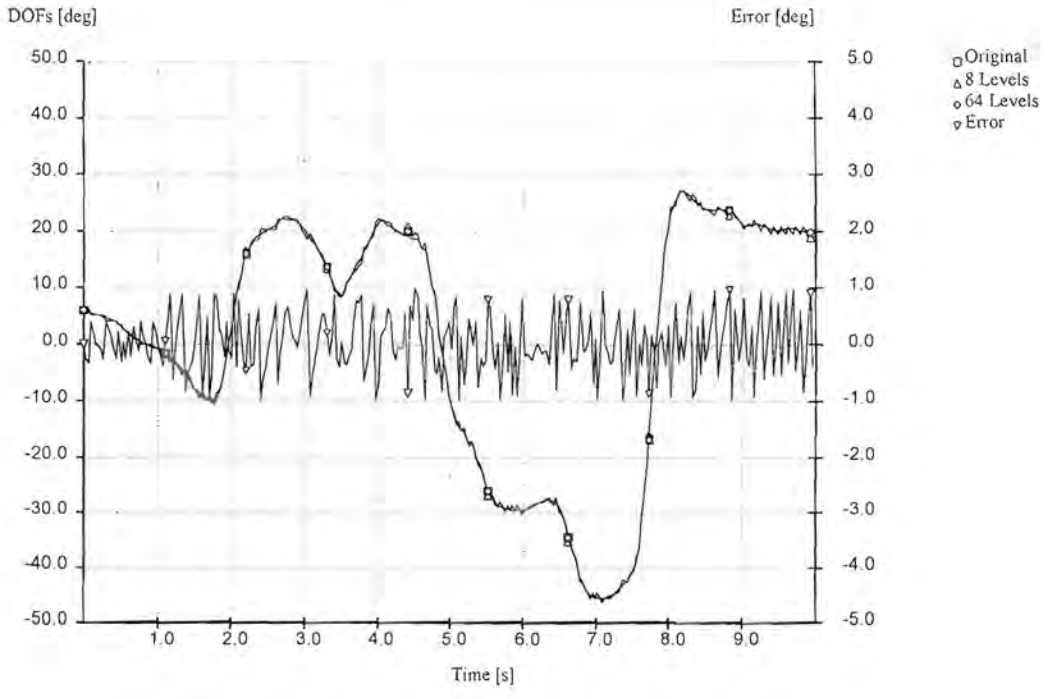

Figure 7-7a: Adaptive predictive coding of $\theta_{3,0}$ with 8 and 64 levels for the conversational sequence.

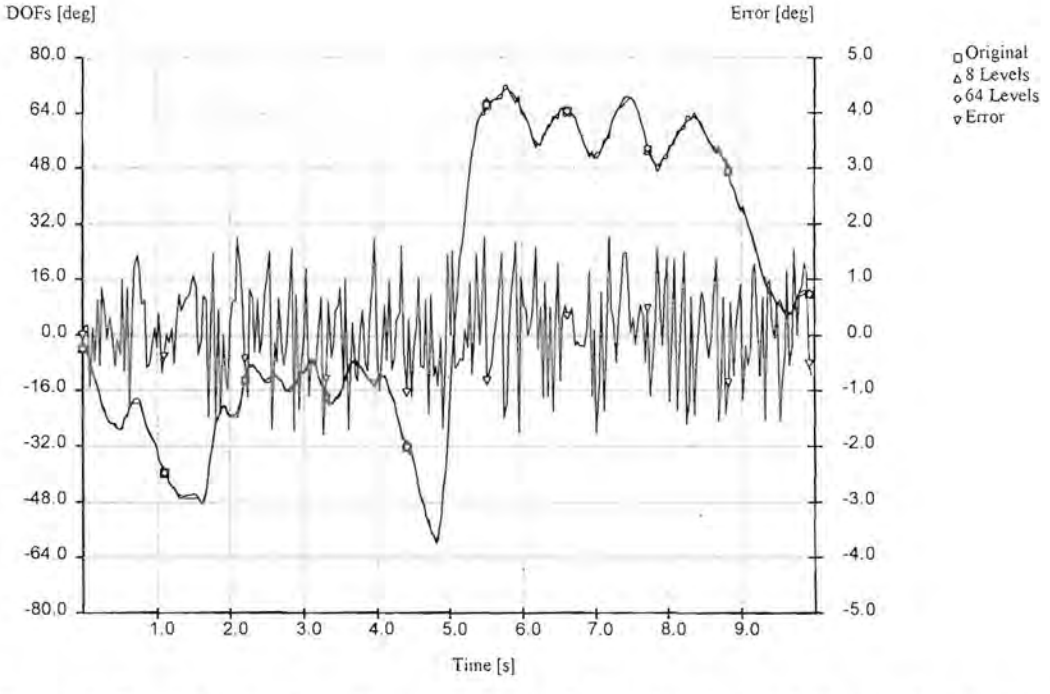

Figure 7-7b: Adaptive predictive coding of $\theta_{5,0}$ with 8 and 64 levels for the wave sequence.

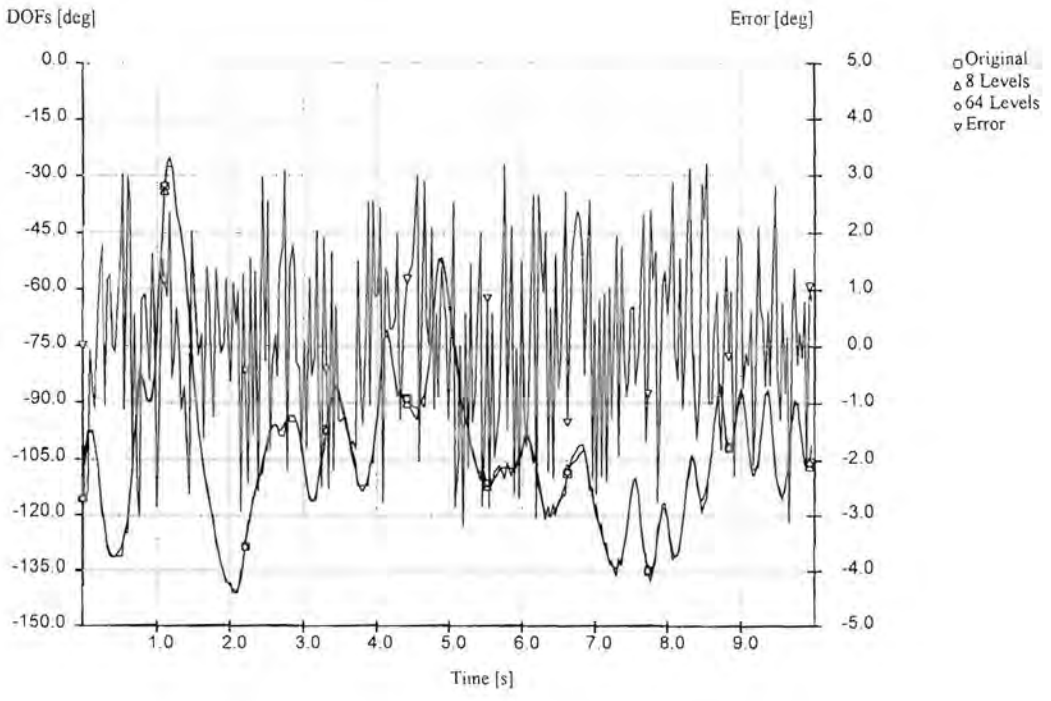

Figure 7-7c: Adaptive predictive coding of $\theta_{6,0}$ with 8 and 64 levels for the dance sequence.

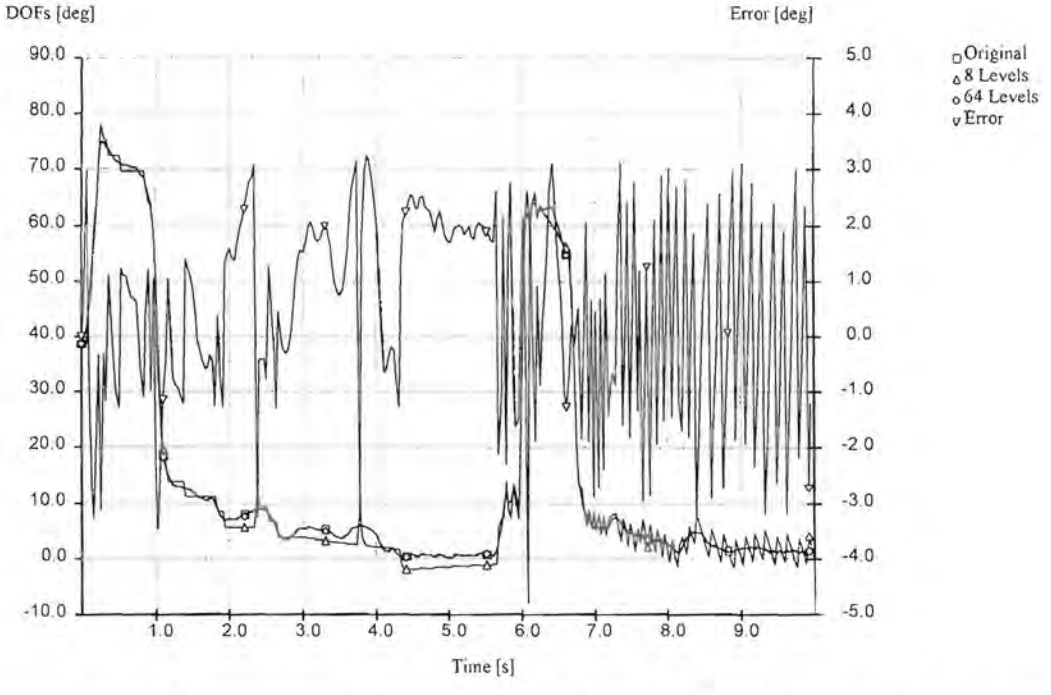

Figure 7-7d: Adaptive predictive coding of $\theta_{3,0}$ with 8 and 64 levels for the gesture sequence.

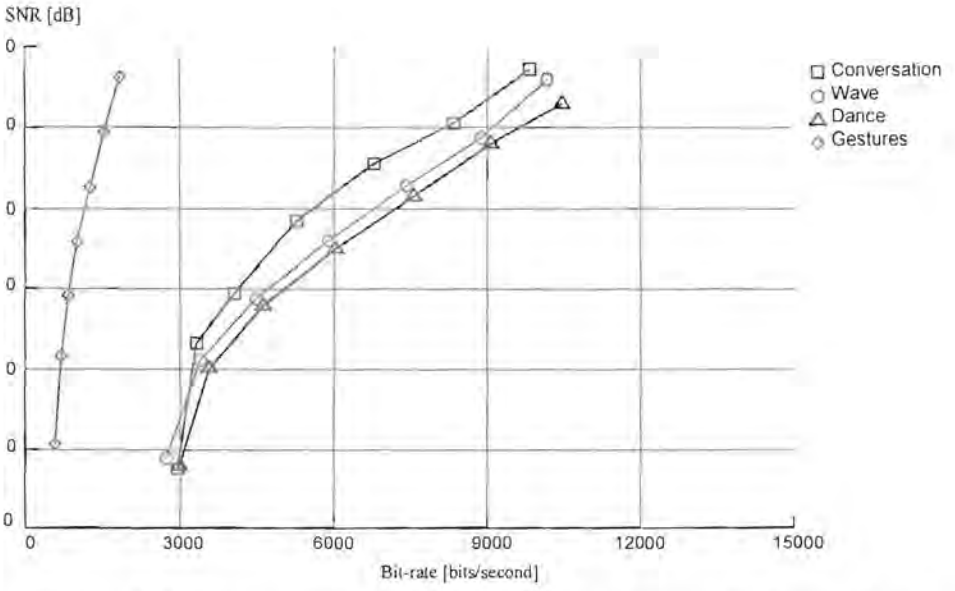

Figure 7-8: Distortion vs. bit-rate for adaptive predictive coding.

7.7 Dead reckoning

Dead reckoning (DR) refers to a traditional method of calculating the position of an object given its velocity and/or acceleration. This concept can be used to decrease the number of update messages that need to be sent in a networked virtual environment, since the position of the object can be extrapolated. Examples of such usage can be found in the Distributed Interactive Simulation (DIS) protocol [49] and the NPSNET system [30,31].

More formally, assume that there is a known function $\theta(t)$ of time, and that the first and second derivatives $\dot{\theta}(t)$ and $\ddot{\theta}(t)$ are known, or can be calculated. For convenience, let $\omega(t) = \dot{\theta}(t)$ and $\alpha(t) = \dot{\omega}(t) = \ddot{\theta}(t)$. If $\alpha(t)$ is constant over a time period $0 \le t < T$ with a value of α, we can write

$$\begin{aligned} \omega(t) &= \omega_0 + \alpha t, \\ \theta(t) &= \theta_0 + \omega_0 t + \frac{1}{2}\alpha t^2, \quad t < T \end{aligned} \tag{7-19}$$

where $\theta_0 = \theta(0)$ and $\omega_0 = \omega(0)$. Similarly, if $\omega(t)$ is constant over a period $0 \le t < T$ with a value of ω, we can write

$$\theta(t) = \theta_0 + \omega t, \quad t < T \tag{7-20}$$

where $\theta_0 = \theta(0)$. Clearly, with θ_0, ω_0, ω and/or α known, the function $\theta(t)$ can be evaluated over the period $0 \le t < T$. The dead reckoning algorithm uses this concept in the following manner:

> Two copies of the human model are maintained. One is the "real" or local human model, as obtained from the input devices or animation system. The other is a so-called *ghost* model, which is updated by the likes of equations (7-19) and (7-20). Denote a single DOF in the real model as the function $\theta(t)$ [2] and the corresponding DOF in the ghost model as $\theta'(t')$. Similarly we define the functions $\omega'(t')$ and $\alpha'(t')$. We also define some error measure $\varepsilon = e(\theta(t), \theta'(t'))$ between the local and ghost model. For simplicity, we will look only at the case of constant $\omega(t)$ (equation (7-20)) in the following discussion. It can at any time be extended to use equation (7-19) as well. At time t, we set $\theta'_0 = \theta(t)$ and obtain a value for the constant ω. The real model is locally updated by the input devices, and the ghost model is updated using equation (7-20). After some time the approximation of constant $\alpha(t)$ or $\omega(t)$ will not hold anymore, and the value of ε will exceed some threshold. At this point, we reset the dead reckoning time t' to zero, again set $\theta'_0 = \theta(t)$ and obtain a new value for ω. By adjusting the error threshold for ε, we can control the resetting frequency of the DR process.

[2] In the case of joint angles, $\theta(t)$ conveniently becomes the angle, $\omega(t)$ becomes the angular velocity and $\alpha(t)$ becomes the angular acceleration.

This concept is shown in figure 7-9. In terms of a networked environment, we only need to transmit messages containing θ_0' and ω when the error threshold is exceeded. The above reasoning assumes that the relative time reference of the local and remote process is exact or at least very close. A similar DR process is used at every receiver for each participant in the virtual world. A problem that is immediately evident from figure 7-9 is that we cannot simply set $\theta_0' = \theta(t)$ during a reset, as it will result in a discontinuity in $\theta'(t')$. There are various methods to compensate for this, but they will not be discussed here. For us, the important point is that it should not influence the frequency at which new updates are sent, nor should it influence the amount of information that needs to be sent.

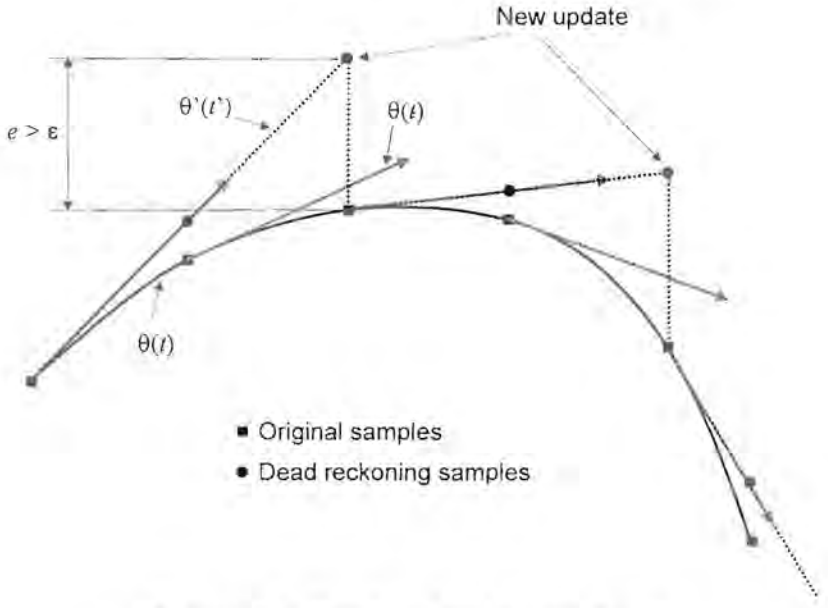

Figure 7-9: Dead reckoning operation.

A generalized schematic for a DR coder and decoder is shown in figure 7-10. This includes the use of a quantizer, and possibly a statistical coder. Note that the DR update parameters are taken *after* the quantizer to account for quantization noise. It is not clear from other DR implementations [35] whether quantization and/or statistical coding are used at all. In our opinion, at least quantization is crucial for further reduction, as this is a basic component of most compression systems. In addition, we do not transmit the absolute position for each new update, but rather the *difference* between the desired and predicted position. The difference can be quantized with fewer levels and results in a further rate reduction. There

is the need for an accurate estimation of the speed and acceleration, if it cannot be measured directly. The use of a Kalman filter has been proposed by [35] for noisy data. We have found that our motion data is relatively noise free, and that it is sufficient to estimate the quantities $\dot{\theta}(t)$ and $\ddot{\theta}(t)$ by fitting at least a second order polynomial to $\theta(t)$, using a simple least-squares solution with a 7 sample window. This method introduces a 3 sample delay, and should be kept in mind when different systems are compared to each other.

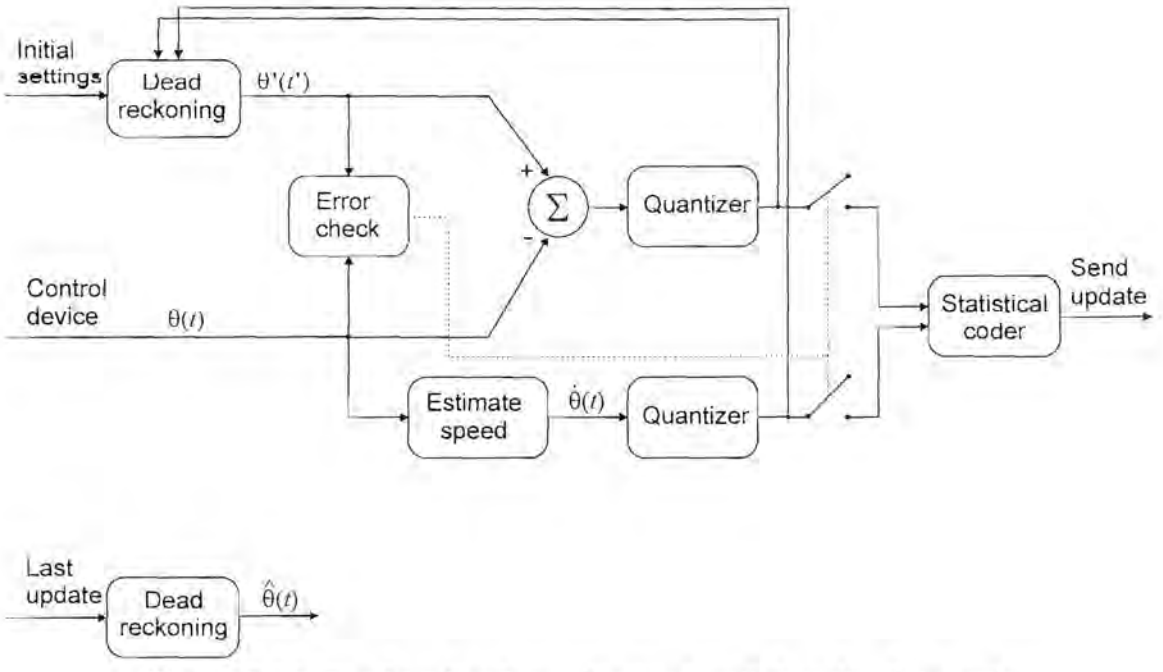

Figure 7-10: Dead reckoning coder (top) and decoder (bottom).

The error measurement can be done in a variety of ways, each of which will result in different performances of the algorithm. A number of possibilities are discussed in chapter 6. The most obvious method for independent single DOF variables is to take the distance $\varepsilon = |\theta(t) - \theta'(t')|$ as the error. However, some DOFs contribute less to visual errors, and it is often advantageous to group a number of DOFs together and to use a weighed mutual error measurement, such as joint distance.

It is possible to use a higher level DR algorithm that analyzes actions and/or motion control methods to make an update decision. Such algorithms will fall under the more

general case of non-uniform sampling methods rather than dead reckoning, and is beyond the scope of this chapter.

Figures 7-11a to 7-11d show the results for the dead reckoning compression method on the representative joint angles. All the DOFs were processed separately using a simple absolute difference as the error measurement. Figure 7-12 shows PSNR against bit-rate for the various test sequences. It can be seen that the dead reckoning algorithm does not perform very well, except possibly for the gesture sequence. In fact, it is evident from the dance sequence that the resulting bit-rate can even *exceed* the original raw bit-rate. This can happen when twice the amount of original information is sent (i.e. both angle and angular speed) too frequently. A check can be done by calculating a histogram of the frequency of updates for the whole sequence. Figure 7-13 shows such an average histogram that covers the whole of the rate-distortion range in figure 7-12. It is clear that the conversational sequence requires updates for *every* new sample for almost 70% of the time. The gesture sequence performs better, and requires new updates only 40% of the time. Still, such frequent updates result in high bit-rates, considering that the value, as well as the derivative, of a DOF is sent.

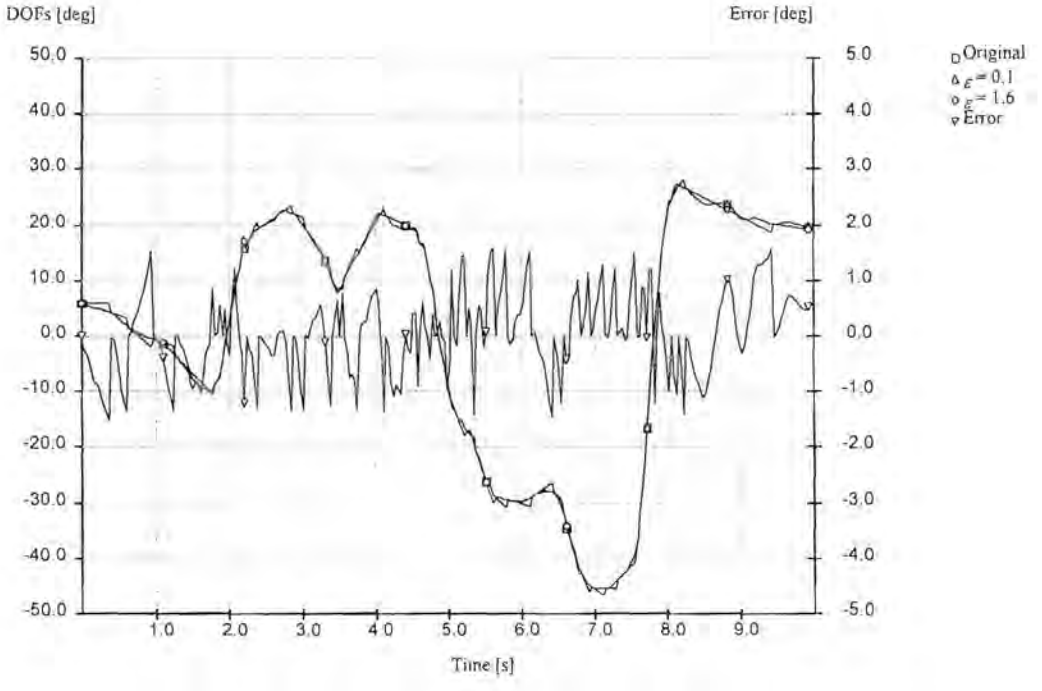

Figure 7-11a: Dead reckoning of $\theta_{3,0}$ for the conversational sequence.

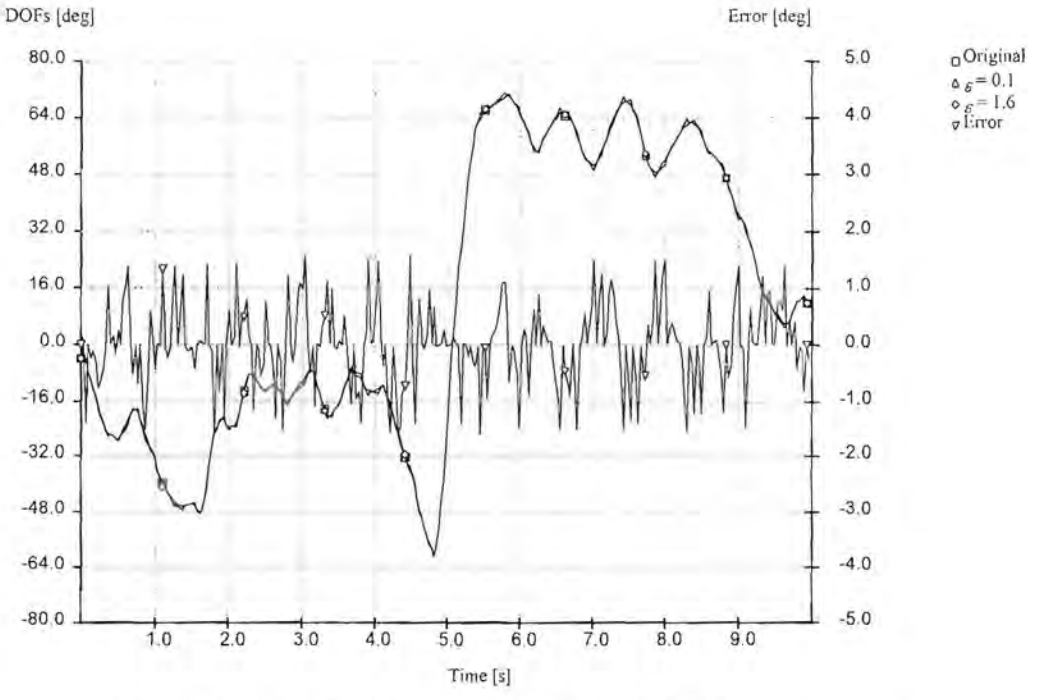

Figure 7-11b: Dead reckoning of $\theta_{5,0}$ for the wave sequence.

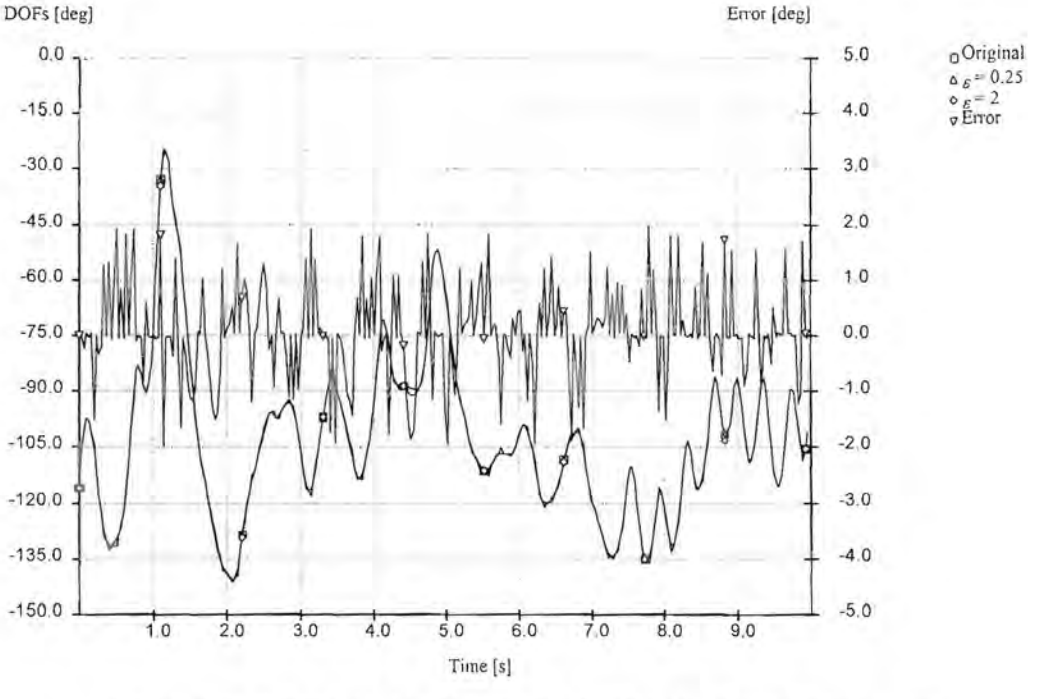

Figure 7-11c: Dead reckoning of $\theta_{6,0}$ for the dance sequence.

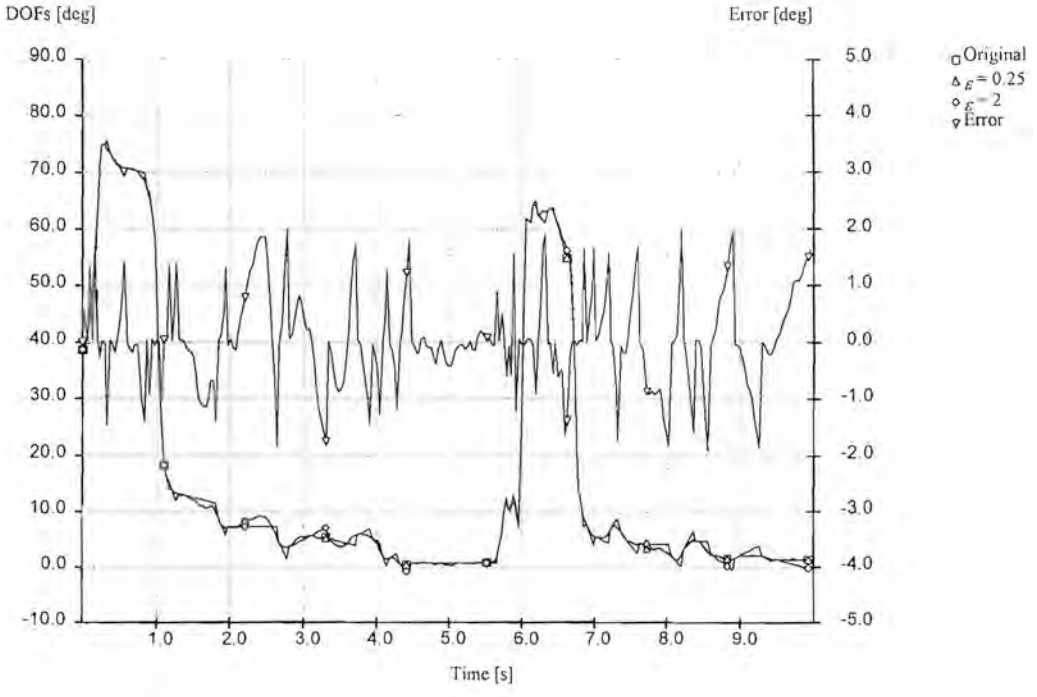

Figure 7-11d: Dead reckoning of $\theta_{27,1}$ for the gesture sequence.

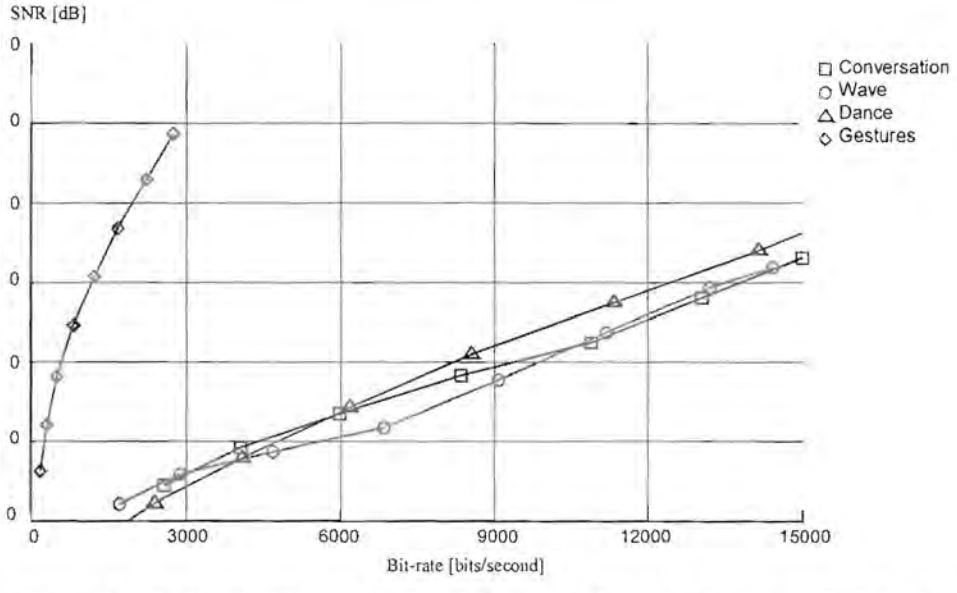

Figure 7-12: Distortion vs. bit-rate for the dead reckoning algorithm.

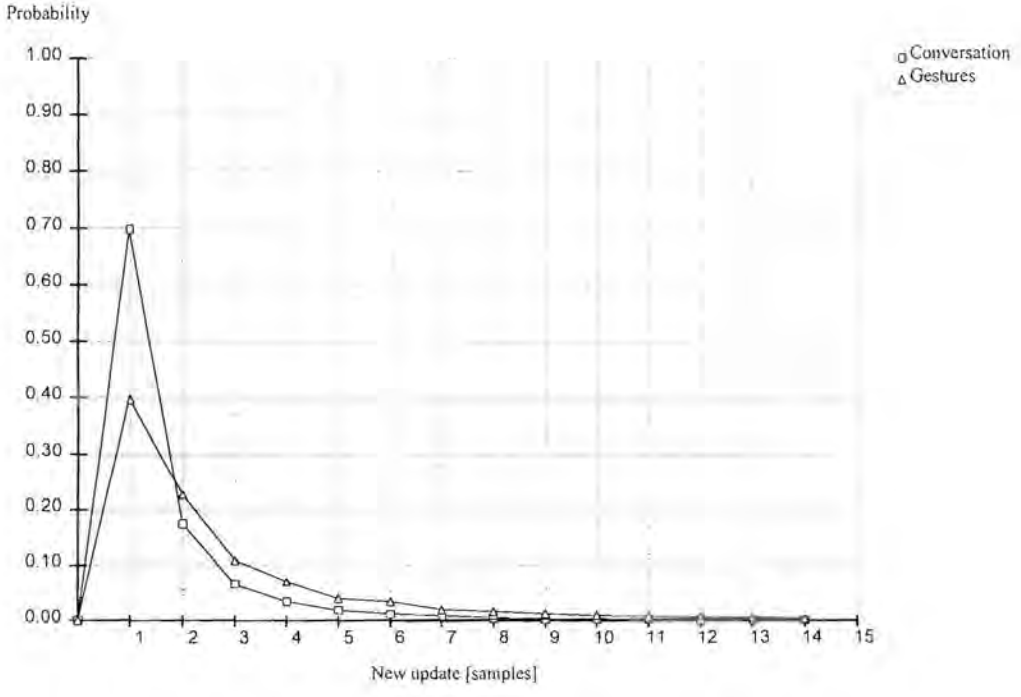

Figure 7-13: Probability of updates.

The dead reckoning algorithm performs reasonably well for slow motion with infrequent bursts of fast motion, and indicates that non-uniform sampling is well suited to such movement. Motion with continuous fast and "busy" sections cannot be predicted accurately with a first order curve. The dance sequence fails completely at low bit-rates. Second order prediction requires an additional acceleration parameter, and we have found that the bit-rate performance even worse. A higher level dead reckoning technique, such as one where the prediction is based on a full dynamic simulation of human motion, might perform better, but is not investigated here.

7.8 Frequency based coding

Frequency or transform based coding methods usually imply a transformation from the time to frequency domain, from which a normal coding route is then taken. Many signals have a more suitable representation in the frequency domain for coding than the time domain representation. The reason for this is that the inherent sample-to-sample correlation that exists in most natural signals tends to cluster the energy in the frequency domain in a relatively small number of transform samples. To achieve bandwidth reduction, those

frequency components or samples with low magnitude could be grossly quantized or even discarded, without introducing serious degradation. Unfortunately, human motion is characterized by large temporal and frequency variations. Natural human motion consists of slow movements with long time windows and small frequency windows, with occasional fast movements with narrow time windows and large frequency windows. It would therefore be convenient to describe the signal in both the time *and* frequency domain. This can be accomplished by a number of so-called time-frequency methods, such as short-term Fourier spectrum manipulation and wavelet decomposition. Wavelet decomposition is particularly attractive, since it helps observing rapidly changing functions by using shorter time windows, and low frequency components by using longer time windows (this is different from the Fourier transform, where the bases are characterized by an infinite time window).

7.8.1 Discrete cosine transform (DCT)

The Karhuhen-Loeve Transform (KLT) (sometimes referred to as the eigenvector transform) is a technique for transforming a signal into a set of uncorrelated representational coefficients. However, it is well known that signals with Markovian properties can be decorrelated with faster and simpler approaches such as the Discrete Cosine Transform (DCT), while approaching the efficiency of the KLT process. In chapter 4, we have seen that human motion indeed exhibits such temporal causal relations, or Markov properties, due to the inherent inertial forces at work.

The DCT transforms a real sequence $\{\theta(n)\}$ of length N into an array $\{\Theta(n)\}$ of length N frequency coefficients or components. The value of $\Theta(0)$ is often referred to as the DC component, and represents the average or mean of the sequence $\{\theta(n)\}$. The rest of coefficients are referred to as the AC components, and contains increasingly higher frequency information about $\{\theta(n)\}$. Human motion data have relatively low frequencies, and the higher frequency components of $\{\Theta(n)\}$ are often very small and can be discarded. This could dramatically reduce the amount of data that is presented to the channel. The forward DCT is defined by the formula

$$\Theta(k) = C(k)\sum_{n=0}^{N-1}\theta(n)\cos\left(\frac{(2n+1)\pi k}{2N}\right), \tag{7-21}$$

and the inverse DCT is defined by

$$\theta(n) = \sum_{k=0}^{N-1}C(k)\Theta(k)\cos\left(\frac{(2n+1)\pi k}{2N}\right). \tag{7-22}$$

The quantity $C(k)$ is a scaling coefficient and can be implemented in various ways. We use

$$C(k) = \begin{cases} \frac{1}{\sqrt{N}}, & k = 0 \\ \frac{\sqrt{2}}{\sqrt{N}}, & k > 0 \end{cases}. \tag{7-23}$$

Refer to [50] for more details on the DCT.

Figure 7-14 shows a block diagram of a DCT based coder and decoder. Disregard the adaptive section for a moment. The input sequence is segmented in groups of length N, where N is a power of two, to be able to use a fast algorithm for the DCT. The resulting DC and AC coefficients are quantized and statistically coded. Again, there is a choice between using inter-frame prediction for the coefficients, or using straight quantization. This section of the coder is very similar to the predictive and adaptive predictive coders discussed earlier, and will not be explained in detail. The DCT coder is a bit more complicated than the simple predictive coder in the sense that there are more parameters to be optimized, especially regarding the quantization of various frequency components. We use empirically determined step sizes and ranges for the different AC and DC frequency components. The DCT based decoder, also shown in figure 7-14, simply performs the inverse operations of the coder, including an Inverse Discrete Cosine Transform (IDCT).

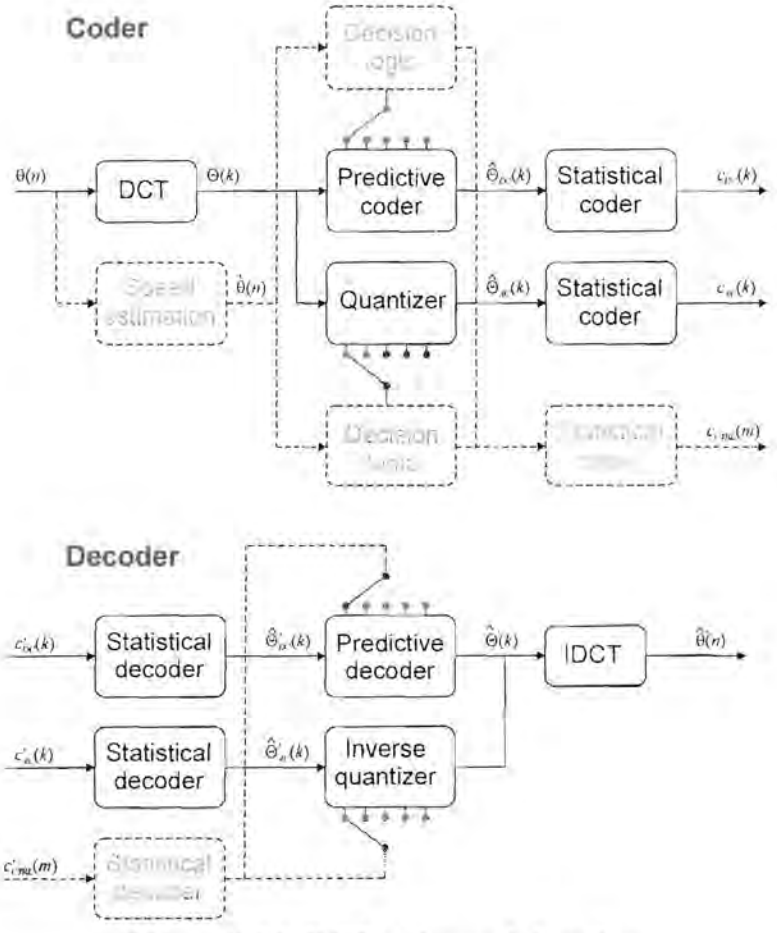

Figure 7-14: DCT coder and decoder.

Similar to techniques used in video compression such as the MPEG format, the operation of the DCT coder can be adapted to short-term motion characteristics. We have found that the visual artifacts generated by quantization errors are less noticeable during fast movement. The quantization can therefore be coarsened in such situations, resulting in an increased compression ratio. The dotted lines in figure 7-14 indicate an adaptive solution that has been developed for human motion. An estimate of the first derivative (i.e. the speed) is obtained from the input sequence. This quantity is then passed through a threshold decision algorithm, which in turn chooses from a set of quantizers and predictive coders. This is a feed-forward approach, and the decision information must be transmitted along with the coded frequency components. It should be noted that the use of adaptive coding based on an subjective quality observation would adversely affect the use of the MS error measurement, although there is a decrease in bit-rate. The results and PSNR measurements presented at the end of the section were obtained using the adaptive method.

The DCT operation could obtain a more efficient frequency estimate of the input sequence for larger N, but we also have to realize that we introduce a delay of N samples by doing so. Furthermore there are more non-zero AC coefficients to be coded. On the other hand, for small N there are more DC components to be coded, each of which uses more bits than the corresponding AC components. We use a relatively small segment length of $N = 16$, which was determined empirically. Even so, at an input sample rate of 30 Hz, this represents a coding delay of 0.5 seconds. This delay sets the DCT method completely apart from the other techniques presented in this chapter. If real-time interaction is of cardinal importance, high delay techniques such as transform coding cannot be used. However, since the exact delay is known, it can be compensated for when comparison studies with other algorithms are done.

Figures 7-15a to 7-15d show the results for the DCT compression method on the representative joint angles. Depicted are the original signal, the decoded signal with two different adaptive threshold decision schemes, as well as the error signal of the second adaptive method. The threshold and quality parameters are given in Appendix III. It is clear that the DCT method outperforms predictive coding by at least a factor two. The errors introduced by the DCT algorithm are also visually much more pleasing, except for block effects due to the finite length window used, which results in a periodic jerk. We compensate for this by smoothing the first and last samples of two adjacent blocks (not shown in figure 7-15). The MS error increases slightly in doing so, but the visual results are much more pleasing, as can be seen clearly in the video clips.

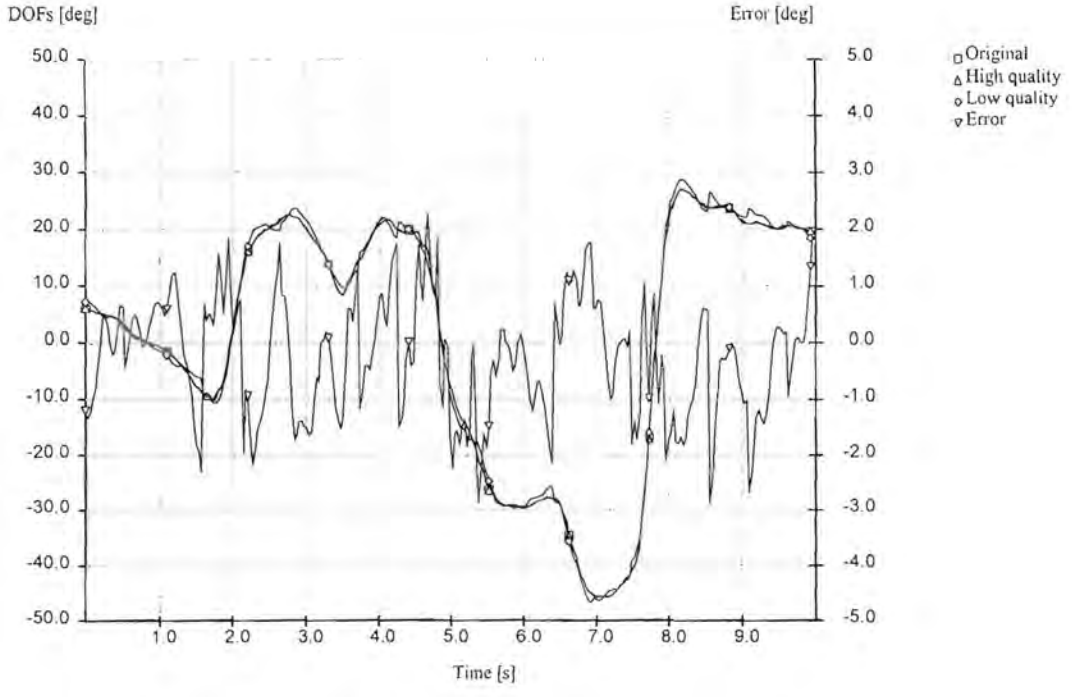

Figure 7-15a: DCT coding of $\theta_{3,0}$ for the conversational sequence.

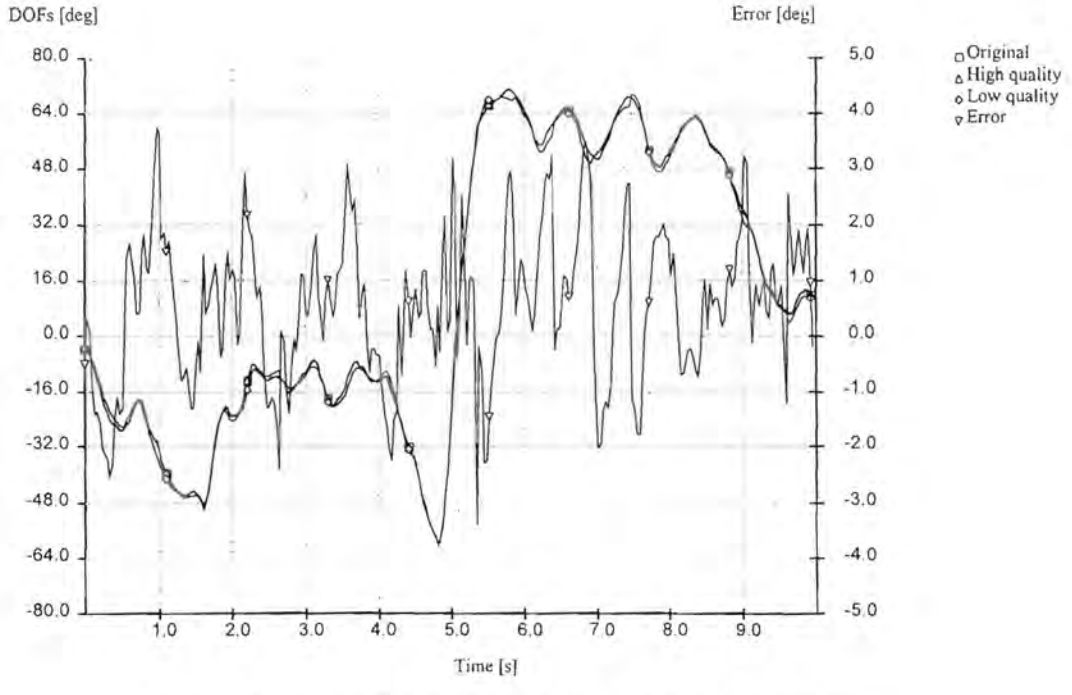

Figure 7-15b: DCT coding of $\theta_{5,0}$ for the wave sequence.

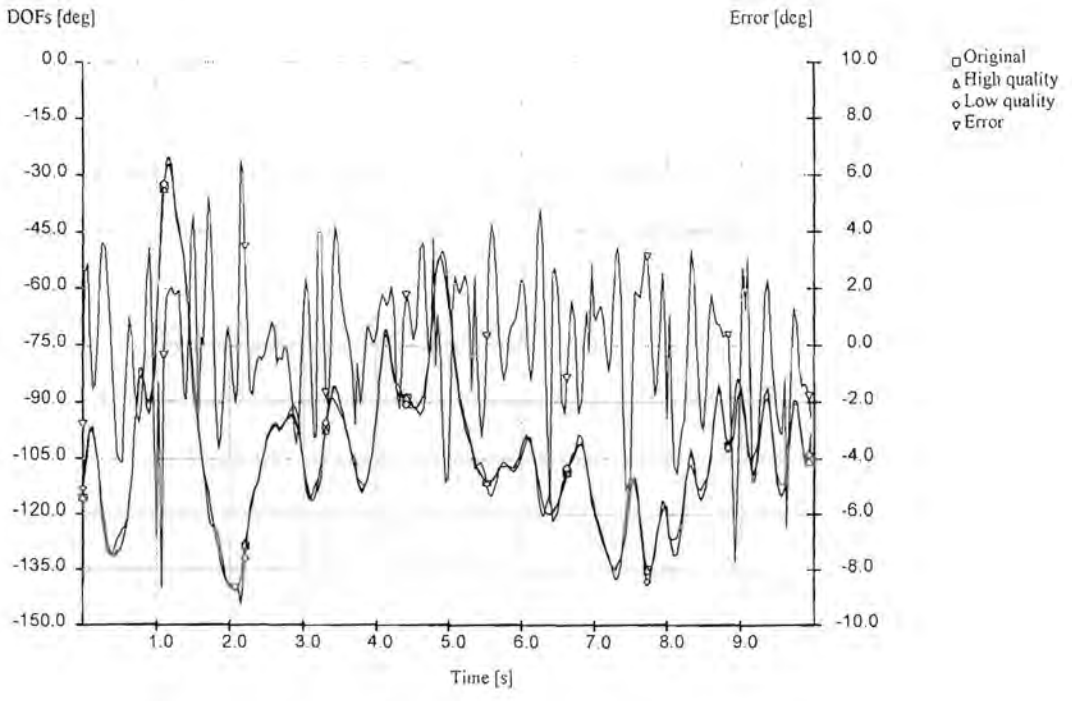

Figure 7-15c: DCT coding of $\theta_{6,0}$ for the dance sequence.

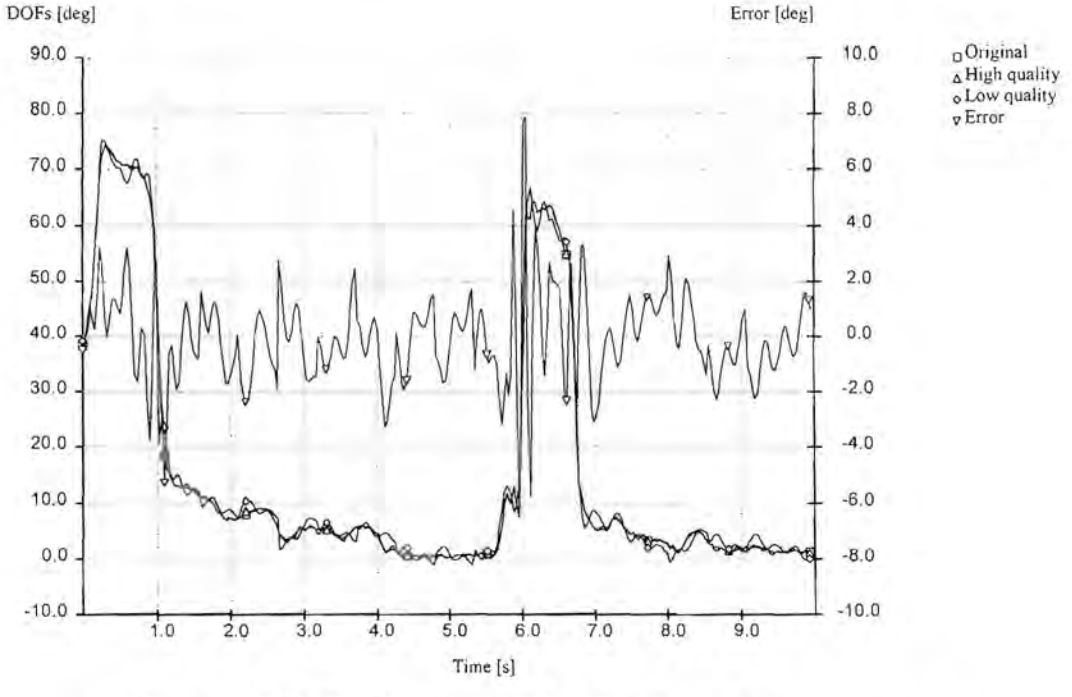

Figure 7-15d: DCT coding of $\theta_{27,1}$ for the gesture sequence.

Figure 7-16 shows the PSNR against bit-rate for the DCT coding method. There is a clear advantage compared to the other methods presented thus far.

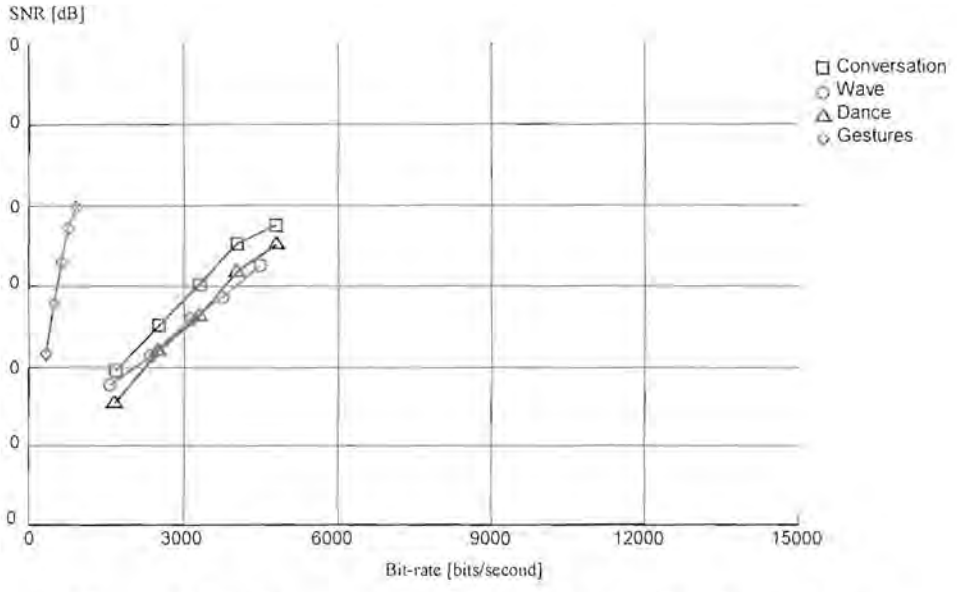

Figure 7-16: Distortion vs. bit-rate for the DCT algorithm.

7.8.2 Time-frequency methods

We briefly investigate the use of wavelet decomposition and subband coding as human motion compression methods. Both of these methods offer a more intuitive approach towards quantization strategies. For example, the structure of a subband coder closely resembles the definition of the VMSE discussed in the previous chapter. A natural bit allocation and quantization strategy could therefore be applied to favour the formulation of the VMSE in order to obtain visually pleasing decoded motion. This concept is very similar to the noise masking characteristics of the human ear. Speech coders (such as subband coders) often make use of these characteristics to enhance the perceived audio quality. In a similar manner wavelet decomposition allows us to choose and manipulate various properties of the motion signal to achieve a better visual effect.

Figure 7-17 shows a generalized subband coder/decoder and figure 7-18 a basic wavelet approach. The subband coder divides the input signal into a number of frequency bands, each of which is separately coded with a predictive coder. The performance of the predictive coders can be adjusted to better suit visual fidelity, even if it results in a drop in PSNR. We use a frequency division scheme similar to figure 6-1. The wavelet approach

decomposes the input signal into a number of decomposition levels. We use a full decomposition, i.e. there are $\log_2(N)$ levels for an input length N that is a power of two. The M largest decomposition values are retained by means of a threshold system, where $M < N$. These values are separately coded with predictive coders that can be adjusted for visual fidelity.

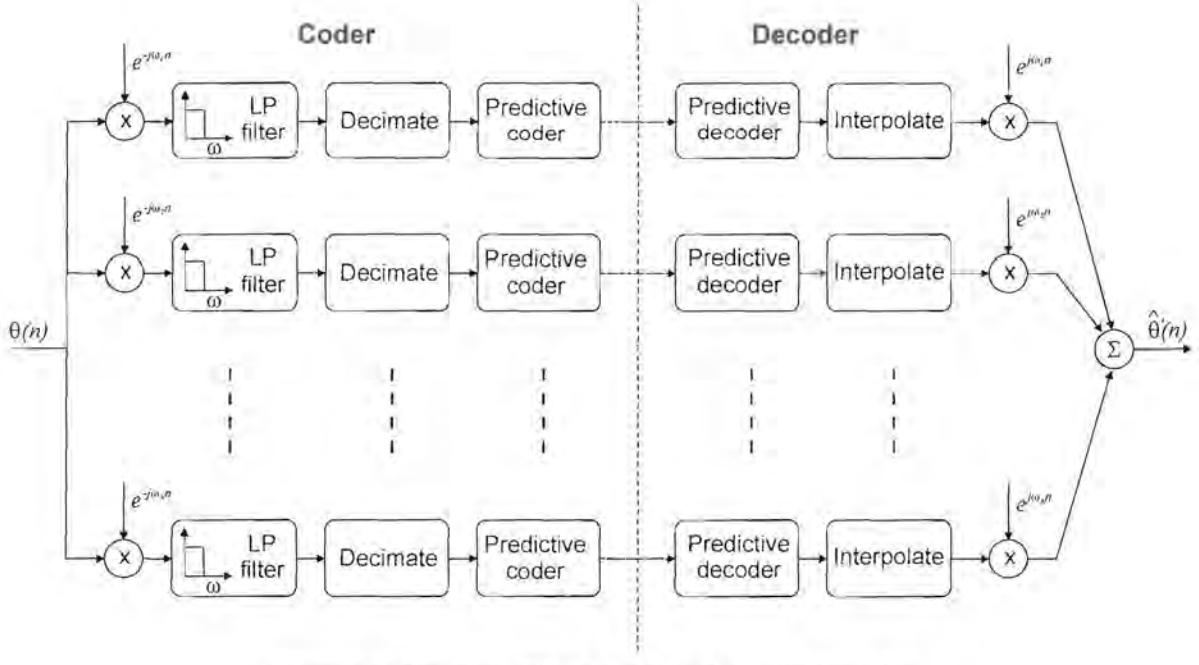

Figure 7-17: Simplified subband coder/decoder.

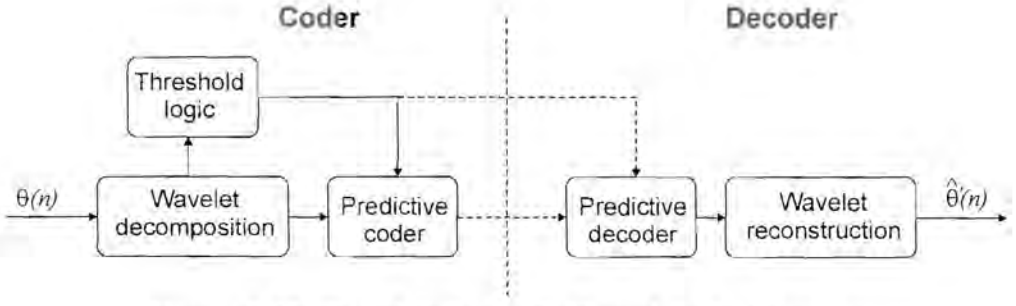

Figure 7-18: Simplified wavelet based coder/decoder.

In theory, the time-frequency methods appear very appealing and intuitive. However, due to the very low sampling rate there are few samples to work with. Similar to the DCT method, we cannot afford a coding delay of much more than 0.5 seconds or 16 samples. This results in practical issues such as filter length constraints and windowing problems.

The actual implementation also suffers from annoying artifacts caused by block edge effects that completely mask the visual gain obtained by the selective coding process. Due to these edge effects we have found the resulting performance to be roughly the same as the DCT method, and visually below that of the adaptive DCT method for similar bit-rates.

7.9 Vector quantization methods

Vector quantization (VQ) involves the grouping of a number of variables together to form a vector. This group can then be quantized, as opposed to the single variable quantization discussed above. Variables or joints that are grouped together must be correlated in some fashion, otherwise nothing is gained and the results will be no different from single variable quantization. Variables can either be grouped spatially or temporally, or both.

VQ can be formulated as follows [51]: Assume that Θ is a k-dimensional vector consisting of real-valued random variables. Vector quantization is defined as the mapping of Θ onto another k-dimensional vector Θ' such that we can write

$$\Theta' = Q(\Theta). \tag{7-24}$$

Θ' takes one of a *finite* set of values $\{\theta_i\}, 1 \le i \le N$. The set $\{\theta_i\}$ is referred to as the codebook, the individual entries are the code vectors and the size N of the codebook is referred to as the number of levels. To design the codebook, we divide the k-dimensional space of Θ into N regions $\{\mathbf{C}_i\}, 1 \le i \le N$, with a vector θ_i associated with each region $\mathbf{C}_i$. The quantizer then assigns the code vector θ_i if Θ is in $\mathbf{C}_i$.

If we denoted some error measure between Θ and Θ' as $e(\Theta,\Theta')$ the overall average error is given by

$$\varepsilon = \lim_{M\to\infty} \frac{1}{M} \sum_{n=1}^{M} e(\Theta(n),\Theta'(n)). \tag{7-25}$$

The quantizer is optimal in terms of the error measure if the overall error is minimized over all N levels. Two conditions are necessary for this. The first is that the quantizer must be realized by using a nearest neighbour selection rule

$$Q(\boldsymbol{\Theta}) = \theta_i \quad \text{iff} \quad e(\boldsymbol{\Theta}, \theta_i) \leq e(\boldsymbol{\Theta}, \theta_j), \quad i \neq j, \quad 1 \leq j \leq N. \tag{7-26}$$

The second condition is that the code vector θ_i must be chosen to minimize the average error in region $\mathbf{C}_i$. Such a vector θ_i is called the centroid of region $\mathbf{C}_i$, and again depends on the error measure being used.

The establishment of an optimal codebook that fulfills the above requirements generally requires and exhaustive search method and is computationally extremely expensive. An alternative sub-optimal method is the Linde-Buzo-Gray (LBG) algorithm [72], which repeatedly splits a region into two smaller regions and assign a codebook entry to each, until a desired size is reached. The algorithm consists of the following steps:

- Create an initial region that contains the entire training set. The initial codebook therefore has one entry corresponding to the centroid of the entire set.
- Split the region into two using a well defined procedure. The codebook now has twice the amount of entries.
- Repeat the splitting process until the codebook reaches its desired size.

The splitting procedure has a big impact on the optimality of the codebook. Ideally each region should be split or divided by a hyperplane that is normal to the direction of maximum distortion. This ensures that the maximum distortions of the two new regions will be less than that of the parent region. However, calculating the maximum distortion as the codebook size increases becomes computationally expensive, and often a simpler scheme is used. We simply use Euclidean distances (L_2-norms) as error measurement and a "diameter splitting" technique, similar to the alternative LBG algorithm [72].

7.9.1 Spatial quantization

Spatial vector quantization can be done on a group of spatially correlated DOFs. In chapter 5 it was intuitively shown that there is correlation between joints and segments that form part of a limb, such as the arms, legs, torso etc. However, establishing a vector quantization codebook with sufficient entries to accommodate a large number of possible movements requires huge training sets and computational power. Such codebooks are also restricted to a very specific type of motion, and is difficult to generalize. Spatial vector quantization does not perform very well for arbitrary body movement where the spatial interrelationships between DOFs are not clearly specified (i.e. correlation is not necessarily measured accurately by the L_2 norm). Figures 7-19a to 7-19b show the results for the conversational and gesture sequences respectively, each with a codebook length of 32 and 256 entries. Also shown is the error for the 256 entry coding. The reconstruction of the other sequences is similar or even worse, and is not shown. Figure 7-20 depicts the PSNR for all the sequences. Note the change of scale compared to previous rate-distortion results. Although quite high compression ratios are achievable (10:1 or more), the error is clearly unacceptable. The video clips also confirm this.

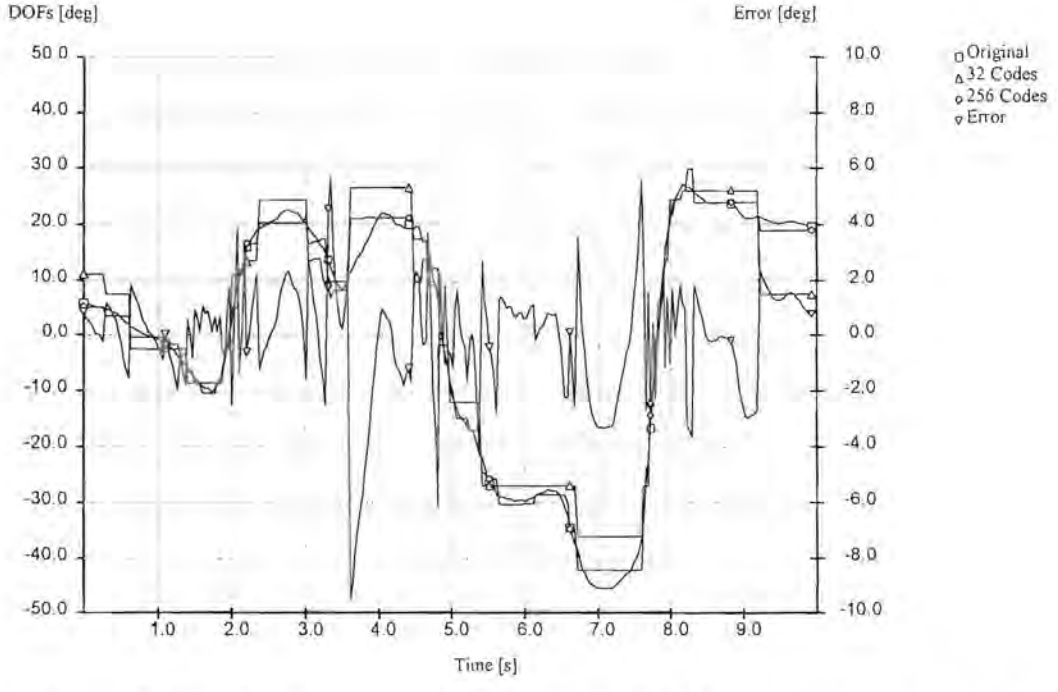

Figure 7-19a: Spatial vector quantization of $\theta_{3,0}$ for the conversational sequence.

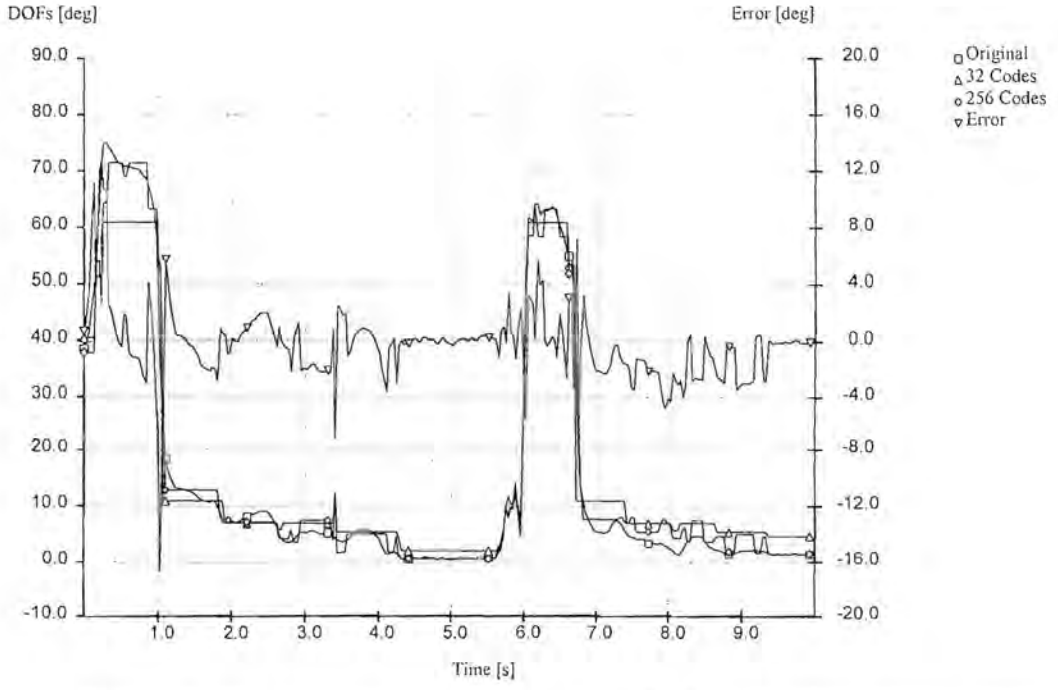

Figure 7-19b: Spatial vector quantization of $\theta_{27,1}$ for the gesture sequence.

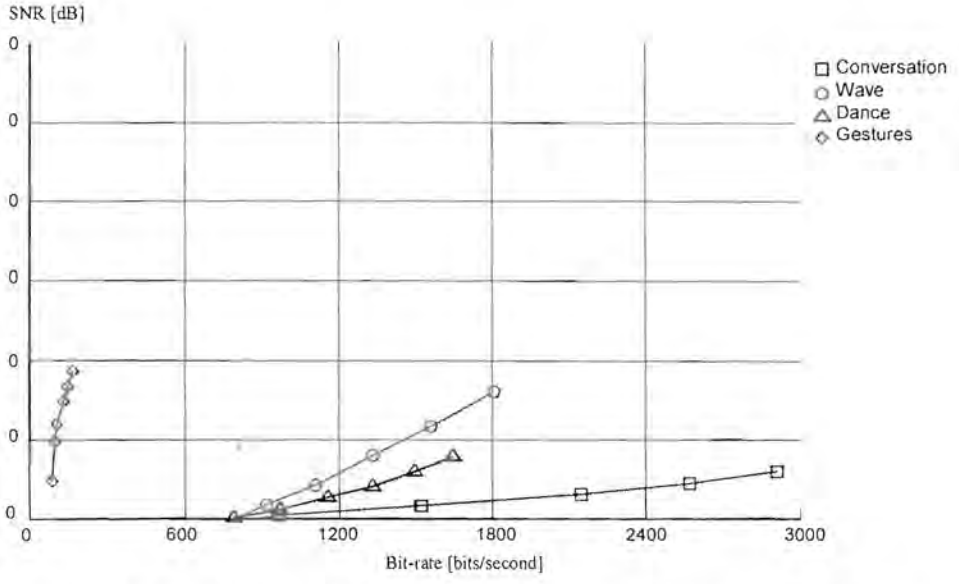

Figure 7-20: Distortion vs. bit-rate for spatial vector quantization.

One example of where spatial vector quantization does perform well, is hand gestures. There are a limited number of universally recognized hand gestures that are frequently used by people. It can almost be seen in the context of a universal sign language. These

gestures are usually a combination of open and closed fingers. From this viewpoint, the whole hand can intuitively be quantized with 32 codes (5 bits – one for each finger). Building such a codebook using the techniques described earlier requires a representative gesture motion sequence that contains examples of all possible gestures. Our gesture sequence contains a fair number of gestures (the small example segment in figure 5-1c shows a counting sequence). It should be noted that such severe vector quantization approaches a gesture recognition system, which is more appropriately described by model based coding (chapter 8). Although the PSNR shown in figure 7-20 indicates an improvement over other body parts, the MS error measurement is not appropriate for spatial vector quantization, especially if combined with some smoothing technique. For example, even at extremely low bit-rates (and hence high MS errors), the information contained in the counting sequence is still quite evident in figure 7-19, and even more so in the video clips.

7.9.2 Temporal quantization

We have already established in previous chapters that there is a high temporal correlation between adjacent DOF samples. A number of consecutive samples may be grouped together to form a vector, which can then be quantized as described in the previous section. Although this technique introduces a delay that is proportional to the dimension of the vector, quite high compression ratios can be achieved with an acceptable MS error. Due to the temporal correlation, a vector dimension of more than one will always yield better results than straight quantization (one-dimensional) of the DOF in question. Figures 7-21a to 7-21d show the results for the representative DOFs of all the sequences, as well as the reconstruction error for 256 codes. Figure 7-22 depicts the PSNR against bit-rate. In all cases a temporal vector dimension of 6 samples (180 ms) were used. Note the change of scale compared to previous rate-distortion graphs. A clear improvement can be seen compared to the results for direct quantization shown in figures 7-2 and 7-3.

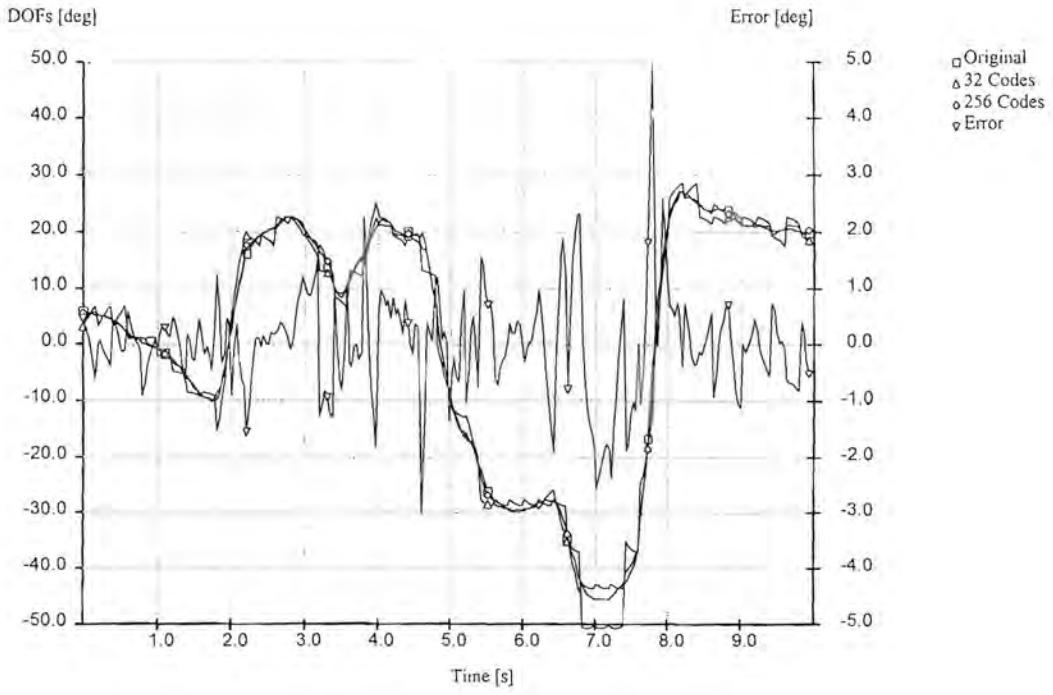

Figure 7-21a: Temporal vector quantization of $\theta_{3,0}$ for the conversational sequence.

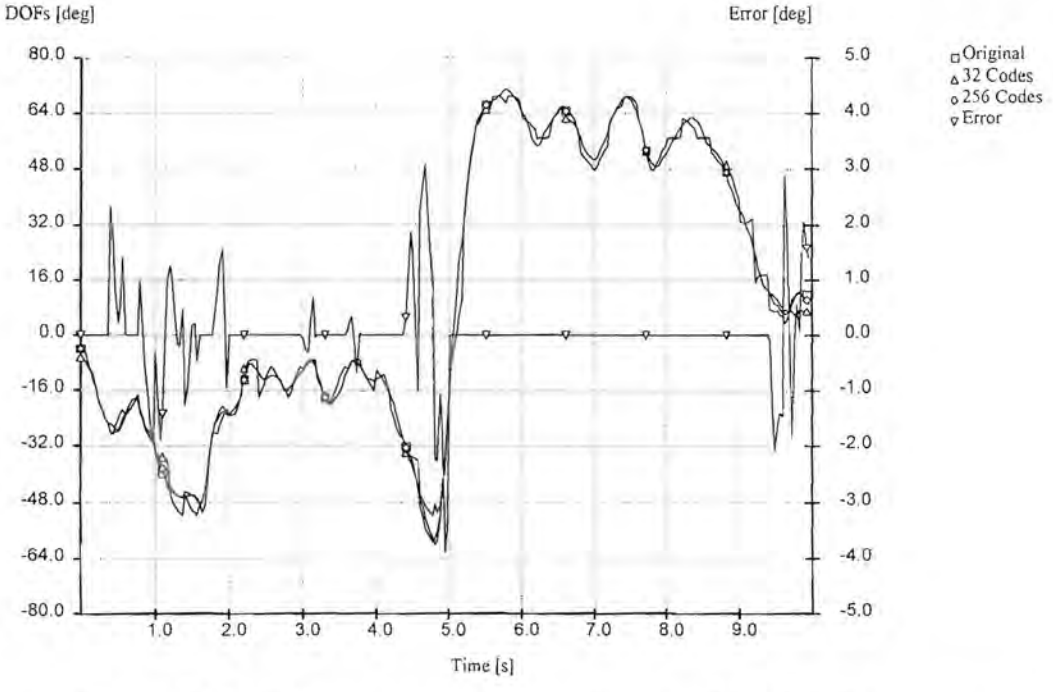

Figure 7-21b: Temporal vector quantization of $\theta_{5,0}$ for the wave sequence.

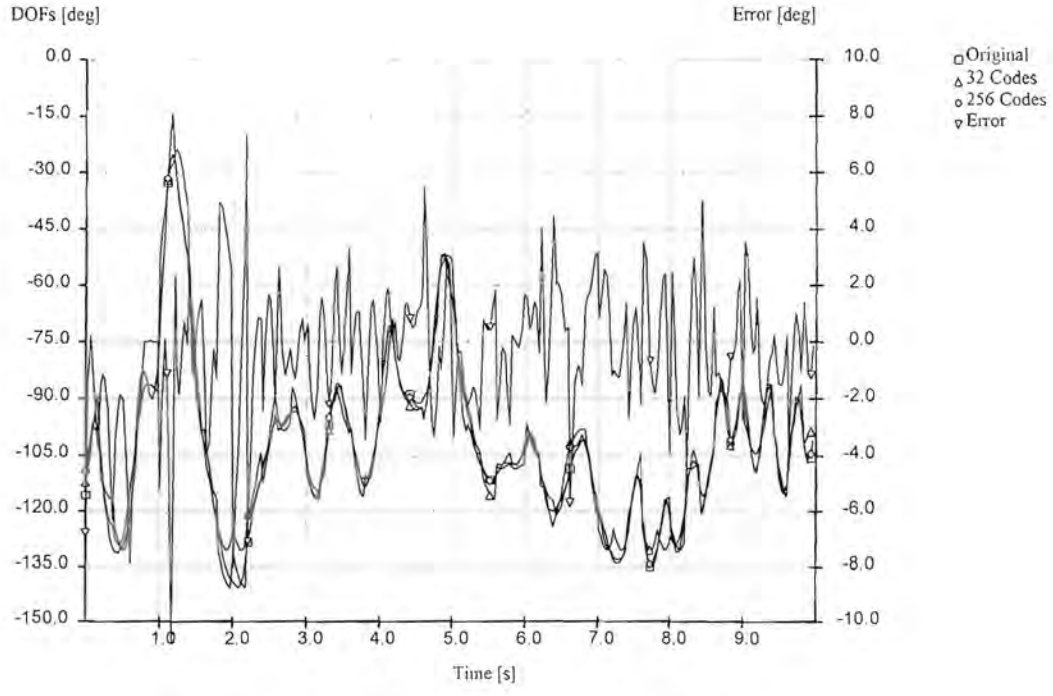

Figure 7-21c: Temporal vector quantization of $\theta_{6,0}$ for the dance sequence.

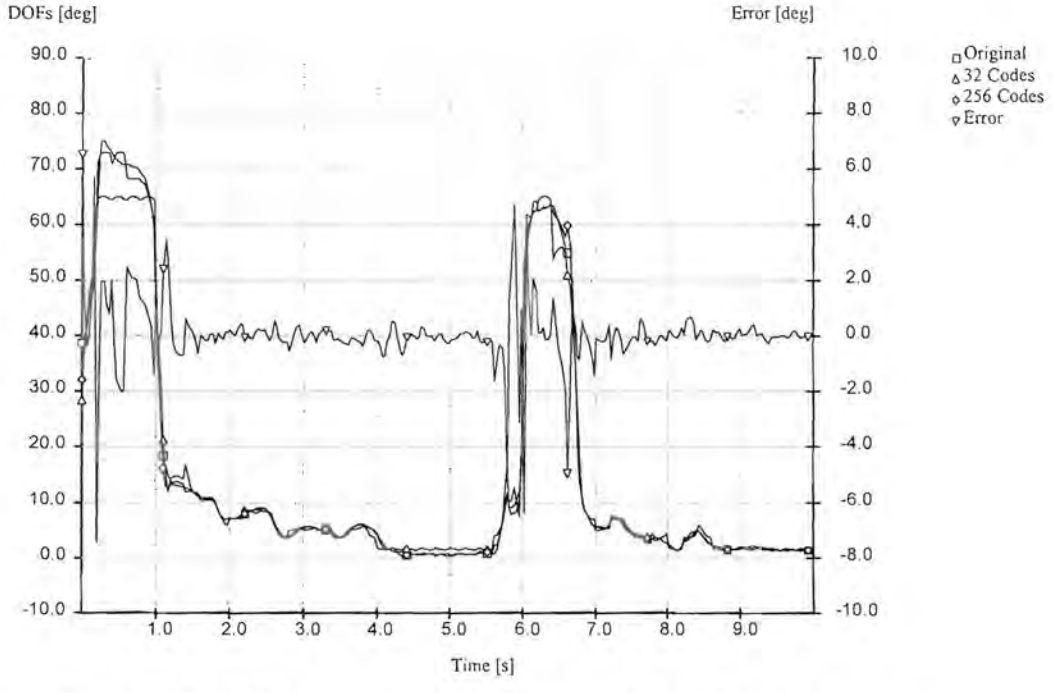

Figure 7-21d: Temporal vector quantization of $\theta_{27,1}$ for the gesture sequence.

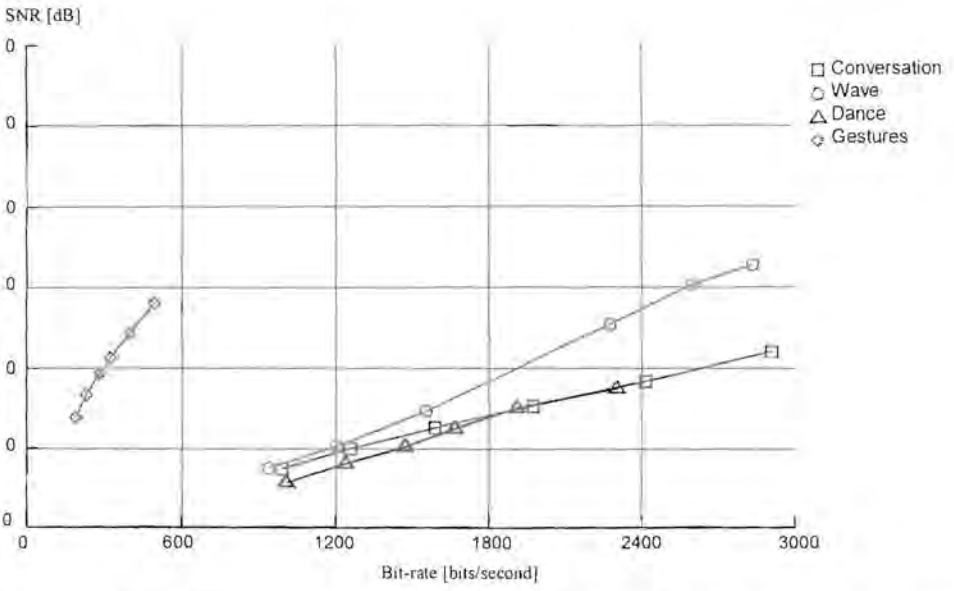

Figure 7-22: Distortion vs. bit-rate for temporal vector quantization.

7.10 Summary

This chapter presented a number of waveform compression methods and results. The use of direct joint angle quantization has been proposed by others [5,29], but at reasonable compression ratios the annoying quantization artifacts prohibits the use of such a naive method. Quantization is left as a functional step in higher complexity methods. The same reasoning is applied to adaptive quantization and statistical coding processes. Predictive coding and especially adaptive predictive coding were quite successfully applied as human motion compression methods. Compression ratios in the order of 5:1 can easily be achieved with these low complexity, low delay methods. Although dead reckoning has been successfully applied to synthetic objects in military applications as a bandwidth reduction technique [30], the results for human motion was not encouraging. Transform coding methods showed great potential, and compression ratios in excess of 10:1 can be expected. However, due to the inherent low sampling rate of human motion, these techniques are plagued by coding delay and block artifacts. The use of vector quantization methods was also investigated, but the requirement of a huge representative training sequence prohibits general use.