Image Compression Techniques Using Vector Quantization.

Colin Scott Ramsay

A thesis submitted for the degree of

Doctor of Philosophy

University of Edinburgh

1994

Abstract

Image compression utilises data redundancy to generate a reduced data representation from which the original image can be reconstructed with the introduction of only small coding errors. Data compression reduces the storage or transmission bandwidth required by the application. In the development of compression methods the aim is to minimise the coding error while increasing the compression ratio. Also, the complexity of the coding algorithm must be considered if a cost-effective solution is being sought.

This thesis reviews the current status of image compression and identifies Vector Quantization (VQ) as a technique which provides good compression and error performance whilst retaining an essentially simple structure which is suitable for VLSI implementation. The coding performance that can be obtained with VQ is extremely sensitive to the quality of the codebook, which is very specific to the training data used to generate it. A novel codebook generation method is presented here which uses intelligent techniques to ensure that the codebook is effectively populated with a wide range of vector types for optimal coding performance.

This thesis also investigates the use of VQ for coding image sequences. Two separate techniques, developed by the author, which code image sequences without adapting the codebook over time, are presented. The performance of these techniques is fundamentally limited and the need for codebook adaption is clear. The aim of codebook adaption is to maintain a codebook which remains effective as the image content changes, whilst adding a minimum data and processing overhead. Methods for codebook adaption are analysed through experiment and provide a codebook adaption technique which is effective, whilst being efficient in bit-rate terms and incurring little additional processing costs.

Kohonen's Self-Organising Feature Maps (KSOFM) have been suggested as a possible method for providing an adaptive core on which to base interframe VQ. KSOFM has been tested by the author as a codebook generation method and found to significantly underperform the codebook generation method presented elsewhere in this thesis.

The thesis concludes with an analysis of possible VLSI implementations of the presented algorithms.

Declaration of originality

I declare that this thesis has been completed by myself and that the research documented in this thesis is entirely my own except where indicated to the contrary.

Acknowledgements

I would like to express gratitude to a number of people for their assistance over the last five years :

- In particular I would like to thank my supervisor David Renshaw without whose aid and encouragement I would have not have completed this thesis.
- Peter Denyer for his technical input to this thesis.
- All those who read my thesis and commented on it – Ken Sutherland, Harry Brash, Henry Bruce and Robbie Hannah as well as Dave Renshaw.
- Those who have passed through the ISG over these years and provided stimulating debate on a variety of topics.
- Deborah, Shona and Gavin for their support.
- My grandmother, Jessie, to whom I dedicate this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Videotelephony is a fast growing market and we will see its spread into the home over the forthcoming years. The technology behind videotelephony relies upon image coding techniques to compress the amount of data required to represent an image. This compression offers a cost saving by reducing the bandwidth required to transmit the image sequence.

Emerging standards for videotelephony are driving its expansion, but it is a feature of these standards that they use mature techniques. These are neither simple to implement nor offer the best compression performance. The search must continue for improved image compression methods which may form the basis of the next generation of standards. The main objective of this thesis is the search for low complexity, high performance image sequence coding methods, especially with a view to VLSI implementation for a cost-effective solution.

The field of image coding is a busy one with considerable research effort covering a wide range of methods and their use in a breadth of applications. An investigation of this field reveals Vector Quantization (VQ) as a good candidate to satisfy the main objective of this thesis. This is due to the simplicity, and regularity, of VQ codec (COder-DECoder) structures, and the level of coding performance which they can deliver. VQ was chosen as the method which would be given further investigation.

Existing VQ techniques have been assessed for their ability to tackle the presented problem and deliver a high performance solution. From this work the need for improved

VQ algorithms has been identified and the development of suitable techniques has been undertaken. The result of this research is a novel codebook generation technique and an original adaptive image sequence coding algorithm. These algorithms have been developed with a desire to simplify the computation required to implement them.

A more detailed outline of the work presented in this thesis will be given in section 1.2. Before this, in section 1.1, the resources and methods used for experimentation in this thesis will be introduced.

1.1 Experimental Resources

The experimental work presented in this thesis was done by modelling the coding algorithms in software. The computing hardware used was the Sun Microsystems workstations available within the university department. The software was written in the 'C' programming language.

Algorithmic development of image coding methods relies upon the ability to view the results for evaluation, or comparison, or indeed for the presentation of results as in this thesis. A number of the Sun workstations were specified with colour displays which enabled the display of greyscale images. This was done by designing the programs to utilise the SunWindows environment which allowed the display of 256 colours at once, and these could be defined to be 256 different intensities of grey. This equates to a monochrome image with each pixel quantized to 8 bits.

The test data used throughout this thesis was available in the public domain. Still images used included the *Lenna* image which is frequently seen as an example in the image coding literature. Another still image used for this thesis was of an urban street called *Mews* and was obtained from the same database. Image sequence algorithms were tested with sequences released by the CCITT, the international communications standards body, precisely for the purpose of image coding development. The sequences used were those called *Claire*, *Miss America*, *Salesman* and *Blue*.

1.2 Thesis Outline

Chapter 2 provides the reader with a broad review of the image coding field. In the light of this review, VQ is chosen as a method which warrants deeper analysis. Chapter 3 gives a fuller review of VQ and presents investigations into codebook generation techniques, including a novel and improved technique developed by the author.

The scope of chapter 4 is VQ coding for image sequences. Existing methods are analysed and the need for adaptive techniques is identified. Simple and effective adaption techniques are developed by the author.

The need for adaptive techniques led to the investigation of Kohonen's Self-Organising Feature Maps and their ability to generate vector quantizers. This is a neural network technique which, it is claimed, has a structure which maps well onto a VLSI implementation. This investigation is presented in chapter 5.

The thesis is concluded in chapter 6. A summary of the achievements of this thesis is given, as well as a discussion of suggestions for future work.

Chapter 2

Image Compression Techniques

2.1 Introduction

Image compression techniques aim to find the minimum data requirement necessary to represent an image. The reason for this is to minimise costs : both storage costs for images which are to be archived, and bandwidth costs for images which are to be transmitted. In most cases, the image, which is reconstructed from the compressed data, will be distorted slightly from its original form. Data compression becomes a compromise between the level of compression achieved and the resulting distortion, with the implementation complexity also an important consideration.

To enable testing of coding methods during their development we must have ways of judging the quality of their coding performance. This can be done qualitatively, where a human observer is used to assess the coded image, or quantitatively, where a numerical measurement is used to compare original and coded images. In the final analysis, any assessment must be qualitative, but during development quantitative methods are more convenient. Image assessment is discussed in section 2.2.

Image coding techniques utilise the fact that images contain a considerable amount of redundant information. This redundancy allows much of the data to be discarded by the coding process. Section 2.3 reviews the techniques in use for image coding.

For image transmission, compression can only work if the coding and decoding ends of the link are using the same methods. The expansion of markets such as videotelephony relies heavily on the adoption of standard methods and image formats. The emerging standards for communications technology and image coding methods are introduced in section 2.4.

The choice of implementation for image coding methods depends on the application with the most difficult being those where real-time compression and decompression are required. Section 2.5 discusses possible methods for implementing image coding algorithms.

2.2 Judging the performance of a compression method

Image quality is a feature which it is very difficult to quantify. It can best be judged by asking a sample of human viewers to give their opinion on it. In most tests used this requires them to scale it from, say, 1 (excellent) to 5 (unacceptable) [2]. However, results from these tests show high variance and thus require large sample sizes to make the results meaningful. Tests of this nature are cumbersome and difficult to carry out. A more reliable method is to offer viewers a choice of two images and ask them to indicate which one they feel is better. Unfortunately, even this does not make for much simpler testing methods.

In developing image coding algorithms, the only realistic proposition for the researcher is to use a quantitative measure of fidelity. These measures indicate how close, numerically, the coded image is to the original image. The most popular measurement in use is the average sample mean-squared-error (e_{ms}) :

$$e_{ms} = \frac{1}{XY} \sum_{i=1}^{X} \sum_{j=1}^{Y} (u_{i,j} - u^*_{i,j})^2 \qquad (2.1)$$

where X and Y are the dimensions of the image in pixels and $u_{i,j}$ and $u^*_{i,j}$ represent the original and coded images, respectively.

Alternatively, a measure called the normalised mean-squared-error (NMSE) has been used by some, [3] and is shown below in equation 2.2. This compensates for the intensity of the image, and will give a more constant result than e_{ms} for the same coding scheme used on images of various intensities.

$$NMSE = \sum_{i=1}^{X}\sum_{j=1}^{Y}\frac{(u_{i,j} - u^*_{i,j})^2}{u^2_{i,j}} \tag{2.2}$$

There are also two definitions of signal-to-noise ratio (SNR) used in the image coding literature, and these are SNR (equation 2.3) and peak signal-to-noise ratio (PSNR, equation 2.4) :

$$SNR = 10\log_{10}\frac{\sigma_u^2}{e_{ms}}(dB) \tag{2.3}$$

where σ_u^2 is the variance of the original image, and

$$PSNR = 10\log_{10}\frac{(peak\ to\ peak\ value\ of\ original\ image\ data)^2}{e_{ms}}(dB) \tag{2.4}$$

SNR is used more commonly than PSNR in signal processing literature, as it gives a value of 0dB for equal signal and noise power. However, PSNR is more frequently used in image coding, and for an image quantized to 8 bits equation 2.4 becomes

$$PSNR = 10\log_{10}\frac{(255)^2}{e_{ms}}$$

It is vital that these measures are not used without reference to a subjective assessment of the picture quality. It is possible for a coded image to achieve a good (i.e. low) value for e_{ms}, yet the picture quality may be poor because of the introduction of an artefact, such as 'blockyness'. A number of alternative fidelity measures have been proposed [4] which have a weighted response to 'tune' them to the human visual system.

The image coding literature quotes information rates in a number of different ways. Most commonly, for still picture compression, one will read of images being 'coded at a rate of n bits/pixel'. For image sequence coding data rates will often be given as 'n bits/frame'. For both still images and image sequences, *compression ratios* are often quoted : these indicate the ratio of the amount of data in the uncompressed image to that in the compressed image.

It is necessary to be sceptical of claims in the literature regarding the compression ratios which each method can deliver [5]. Obviously, it is in the interests of any author to claim that his method offers improved performance. It is common to read claims such as 'acceptable images were obtained at a rate of 1.5 bits/pixel' but one must consider

- What the authors have deemed to be 'acceptable',
- What the resolution of the original image was, and
- What medium is being used to view the images.

As a simple example, one author may be coding images from an original size of 512×512 pixels while another is starting with images of 128×128 pixels : the first contains 16 times the data of the second and will sustain considerably more compression before the image quality becomes unacceptable.

These problems lead to a requirement to set up standard 'benchmark' images and image sequences which coding methods can be tested on, and also a standard method for quoting the results. To do so would not be as difficult as it may seem because most image coding papers cite the use of images which are publicly available and familiar. The design of the benchmark tests may also need to incorporate some guidelines covering the training permissible while developing the coding algorithm, as it is possible in most cases to tailor the algorithm to one image, giving a false result. As with all benchmark tests, however, it would be foolish to read too much importance into their results as there are other details which require consideration, most notably the complexity of the coding algorithm.

2.3 Compression Methods

This section introduces the image compression techniques in use today. These vary from the simpler methods, outlined first, which have been in use in data compression for many years, to those methods which are still in their infancy.

2.3.1 Pulse Code Modulation

Pulse code modulation (PCM) is the technique used to produce what is called a *digital image* which is simply an image which has been discretely sampled both spatially and in brightness [4]. The result of this is a 2-dimensional matrix representing the image where the row and column indices identify a point in the image and the matrix coefficient identifies the brightness at that point. These coefficents are known as *pixels* or, less commonly, *pels*, both being a contraction of the term *picture elements*.

In a monochromatic image each pixel will generally be quantized to 6-, 7- or 8-bits which correspond to 64, 128 or 256 grey-levels. It has been shown that the human eye cannot discriminate between many more than 64 grey-levels which should mean that 6-bit quantization would be sufficient. However, due to the 8-bit nature of the computing platforms used in image processing, images are more often represented with 8-bit pixels.

PCM has performed compression on the original image by dint of the fact that it has been sampled from an infinite analogue source. The resulting image requires $X \times Y \times B$ bits to represent it, where the image has $X \times Y$ pixels, each quantized to B bits. This is the amount of data in what is considered an *uncompressed* image and it is the number against which other compression schemes are compared.

2.3.2 Error-Free Coding

In certain applications it is necessary that the compression method should allow the image to be reconstructed without error [6]. An example of such an application is the compression of satellite images where the amount of data which must be stored is enormous. Here, compression is highly desirable and profitable yet it is also vital that no information is lost.

One method for coding without error is to code the differences between neighbouring pixels : these will generally be much smaller than the pixel intensities. If these values are coded with 4 bits it is possible to record differences of between -7 and +6, and if the difference is larger then the 2 remaining codes can be used to 'shift-up' or 'shift-down' the difference codes.

Other error-free coding methods are contour coding and run length encoding. Contour coding describes a picture by listing the outer perimeters of all areas of constant intensity. Run length encoding maps an image into a sequence of integer pairs (g_k, l_k), where g_k is the intensity and l_k the run length. The run length is the number of consecutive pixels with the same intensity. Both of these methods work well for images which contain large areas of constant intensity.

2.3.3 Differential Pulse Code Modulation

Differential pulse code modulation (DPCM) exploits the fact that adjacent pixels (in space and time) are highly correlated. A *predictor* is used to estimate the value of a pixel based on the values of adjacent pixels, and a difference is formed by comparing this predicted value to the actual value. Assuming that the predictor is reasonably accurate, the difference will generally be of much smaller magnitude than the pixel values themselves. Compression can be gained by using fewer quantization levels (and thus fewer bits) to code the sequence of differences.

However, when the calculated difference is larger than that with which the quantizer is able to cope, *slope overload* occurs. This is manifested as an inability to accurately code sharp edges, where the predictor is most likely to be inaccurate. The encoder can be made to respond more quickly to a rapidly changing input by increasing the width of the quantization bins, but this results in *granular noise* when encoding a slowly varying signal.

Predictors vary in complexity from those using only the previous pixel on the same line to those which use pixels from previous lines or even frames (in an image sequence). All are straightforward to implement but all suffer from the problems mentioned above.

2.3.4 Block Truncation Coding

Block truncation coding (BTC) achieves compression on image data while retaining certain image statistics on a local basis. The image is first divided into small (most often 4×4 pixel) non-overlapping blocks. For each block, the mean and standard deviation are calculated and the pixels of the block are quantized to 1 bit by thresholding them with the mean. The resulting bit plane, the mean and standard deviation are used by the decoder to reconstruct the block such that the mean and standard deviation are preserved.

As a result of the 1-bit quantization and coarse quantization of the mean and standard deviation, data rates of $\sim$1.5 bits/pixel are possible. Sharp edges can look ragged but are generally coded well, and low contrast areas can be given false contours due to the quantization noise. Also, BTC images can have a 'blocky' nature to them which is very undesirable. The coding scheme is low in complexity and has small memory requirements and is thus easily implemented.

BTC is most commonly used in combination with transform coding (see section 2.3.7). This combination creates what is known as a Hybrid Technique.

2.3.5 Motion Compensation

A sequence of motion video contains a number of objects which move between frames. Some of this motion is purely translational relative to the camera, and if the trajectories of these objects were known, they could be used to aid compression. This is done by including Motion Compensation (MC) in a prediction loop. The predictor is able to more accurately predict the next frame and thus the variances of interframe differences are reduced which allows more scope for data compression.

The task of identifying all the objects in a frame and calculating their trajectories is far too complex for the computing power generally available today. A simpler method is that used in the video coding standards outlined in section 2.4. Here the image is segmented into small blocks and the next frame is searched for motion of these blocks. This is done by sliding the block up, down, left and right and finding the closest match to it in the next frame. Block matching can be performed by auto-correlation, or by minimising mean square error or mean absolute error. The search area will be limited, to perhaps 7 or 15 pixels in each direction. The motion vector to the closest match is used in the prediction loop and is sent to the receiver to enable decoding. The calculation of this motion vector is called Motion Estimation (ME) and is illustrated in figure 2–1.

Even this simplified method requires a great deal of computation. For example, if the search area were $\pm$15 pixels then a brute-force search would require 961 block matching calculations for each block of the image. This number is reduced by utilising a directed search method (also called a *logarithmic* search), where the search is performed

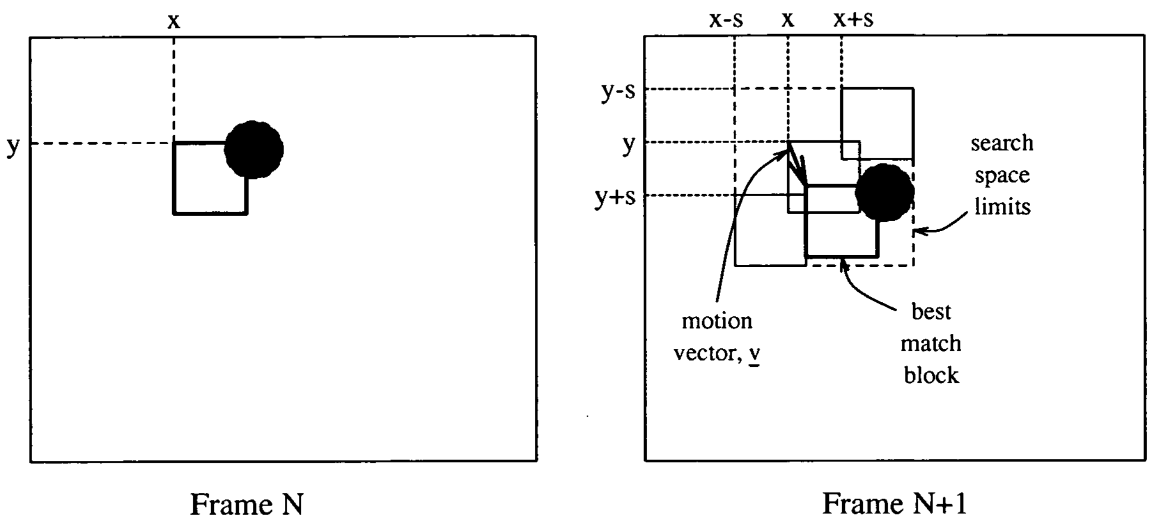


Figure 2–1: Motion Estimation with search space limited to s pixel movement in any direction from original (x, y) position of block. Motion vector, $\underline{v}$, describes translation to best match.

in a hierarchical fashion. A few widely spaced blocks are tested and the search is then centred on the best match of these : this is repeated until the blocks are spaced just one pixel apart. This search method does not guarantee finding the best match but does reduce the computational requirements considerably.

2.3.6 Entropy Coding

Entropy coding, also called Variable Length Coding (VLC), exploits the fact that the output codes from a quantizer occur with non-uniform frequency. Those codes which occur more commonly can be assigned short length codewords, with longer codewords for progressively less frequent events. This results in a reduction in the average bit rate of the codewords sent to the receiver. The average bit rate is bounded by the entropy of the quantizer output signal.

A commonly used example of VLC is Huffman Coding, and its use can be found in the compression standards described in section 2.4.

2.3.7 Transform Coding

Transform coding (TC) has dominated image coding research since the early 1970s and is now widely regarded as the most effective method which uses scalar quantization. A model of a transform codec is shown in figure 2–2.

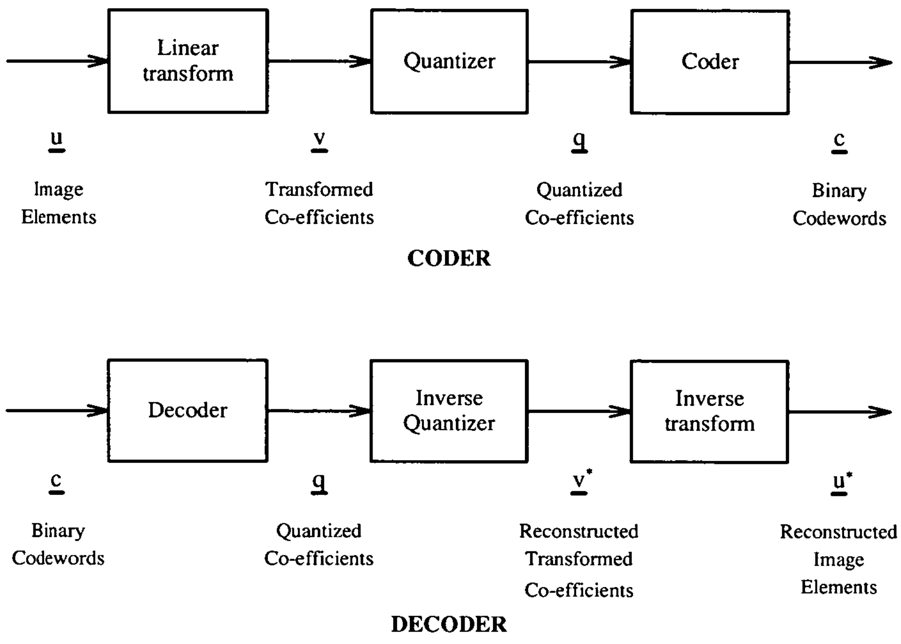


Figure 2–2: A Transform Codec Model

In TC the image is first divided up into subimages of $N \times N$ pixels, most commonly 8×8 or 16×16 pixels in size. An invertible mapping is applied to each subimage to create a set of coefficients, $\underline{v}$, which are less correlated than the image elements $\underline{u}$. These coefficients are then quantized and coded for transmission or storage. The decoder reverses this operation by creating the quantized coefficients and applying the inverse transform to these to restore the pixel intensities.

In removing correlation between the image elements, the transformation is compacting the information into fewer coefficients. This is possible as the variances of the coefficients are, in general, not equal. Compression can then be realised by quantizing only some of these coefficients and discarding others. This is called *zonal sampling*. Also, some of the remaining coefficients contain more information than others so these can be quantized with a variable number of bits. This is called *zonal coding*. Figure

2–3 illustrates both of these operations where the transform in use here has compacted the variances into the top left hand corner of the coefficient matrix.

7	6	5	4	3	3	2	2	2	1	1	1	1	1	0	0
6	5	4	4	3	3	2	2	1	1	1	1	1	0	0	0
5	4	4	3	3	2	2	2	1	1	1	1	1	0	0	0
4	4	3	3	3	2	2	2	1	1	1	1	1	0	0	0
3	3	3	3	2	2	2	1	1	1	1	1	0	0	0	0
3	3	2	2	2	2	2	1	1	1	1	1	0	0	0	0
2	2	2	2	2	2	1	1	1	1	1	0	0	0	0	0
2	2	2	2	1	1	1	1	1	1	1	0	0	0	0	0
2	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2–3: Zonal Sampling and Zonal Coding : typical bit allocation for a cosine transformed block of 16×16 pixels

The ideal transform will pack the most variance into the fewest number of coefficients. The Karhunen-Loeve transform (KLT) is an optimal transform, but is very difficult to compute, and is used only as a baseline for comparison with the performance of other transforms. The most commonly used transform is the discrete cosine transform (DCT), which has a performance close to that of the KLT for large subimage sizes. The DCT can be computed via the fast fourier transform (FFT) in $O(N \log N)$ operations. Simpler transforms such as the Hadamard, Hotelling, Haar and Slant transforms are based on square-waves rather than the sinusoidal FFT and are thus easier to perform but less optimal.

The pre-eminence of TC among image codecs over the last ten years has led to its adoption in the emerging coding standards. These standards are discussed in section 2.4.

2.3.8 Pyramid Coding

Pyramid coding [7, 8] takes its name from the data structure which is created as the first step in the coding process. The pyramidal data representation actually requires more bits than the original PCM coded image, but is such that it can simplify the coding process and enable data compression.

The pyramid decomposition is performed by first low-pass filtering the input image and representing the result with half the resolution in each of the x- and y- directions, making it one quarter the size. This process is then repeated by filtering the output image from the first stage to create an image which is one sixteenth the size of the original. This process can be iterated further and produces an array of images, each one quarter the size of its predecessor, which can be visualised as a pyramid. Burt and Adelson [7] call this the Gaussian pyramid because they perform the low-pass operation with Gaussian weighted filters. The Gaussian pyramid is then used to create another pyramid of images by forming the error images between successive layers of the Gaussian pyramid. The resulting pyramid contains approximately 4/3 times the amount of data in the original image, and it is this data, plus the smallest image in the Gaussian pyramid, which is coded for transmission.

The decomposition process is effected by convolving the image array with small two-dimensional symmetric weighting functions. Fast algorithms are available for this purpose and these can be efficiently implemented in custom VLSI.

Although pyramid decomposition can be regarded as a transform, the succession of image arrays can aid coding algorithms in ways which other transform coding methods can not. This is largely due to the fact that the spatial relationship of the data is retained, unlike other transform methods such as the DCT. In particular, motion estimation in an image sequence coder can be guided by operating first on the smallest image in the pyramid, using this as a basis to direct the searches in the higher resolution images.

Another feature of this coding method is that the image can be transmitted in such a way that the image at the receiver is progressively improved. The top level of the pyramid is sent first, allowing a very coarse rendition of the image to be reconstructed at the receiver. The transmission of subsequent levels of the pyramid improves the quality of the received image. In a similar fashion, a pyramid codec with motion compensation can adjust the depth of the pyramid coding to compensate for the amount of motion in the image, enabling it to retain a near constant bit rate.

2.3.9 Wavelet Transform Coding

Another coding method which shares much with pyramid coding is wavelet transform coding [9-13]. A wavelet transform applied to the input image creates a low-pass filtered and three high-pass filtered versions of the original. This type of coding has also been called sub-band coding [14].

Decomposition of the input image using a wavelet transform is presented by Lewis and Knowles [10]. The wavelet transform is implemented by quadrature mirror filters whereby a low-pass filter and a high-pass filter are applied to the image in both the x- and y- directions. The result of this is a low-pass sub-band and three orientation selective high-pass sub-bands. These four sub-bands, each represented at half the sampling density of the input image, contain in total the same amount of data as the input image. The filtering operation is then repeated on the low-pass sub-band to generate the next level of decomposition, and this can be repeated as often as required.

The image can then be coded by sending the low-pass sub-band at the last level of decomposition in full, then using a thresholding algorithm on the high-pass sub-bands to decide what other coefficients to code. The fact that the sub-bands are high-pass filtered means that edge information, which is visually the most important, is easily identified by thresholding the data. A tree-structure is naturally formed for the update information if coding is done in this way [10].

Blocking effect coding artefacts normally associated with transform codecs, such as the DCT, are not produced by such a coding method. Coding errors are seen as a blotchiness in flat areas and degredation of edges as the compression ratio is increased. Wavelet transform coding offers the same benefits for motion compensation and progressive image transmission as are seen with pyramid coding.

Wavelet transform coding claims to offer a number of advantages over DCT-based transform coding schemes. These advantages are the relative simplicity of the wavelet tranform over the DCT, the simpler motion compensation schemes, and the lack of blocking effects.

2.3.10 Vector Quantization

It is a fundamental result of Shannon's rate-distortion theory, the branch of information theory devoted to data compression, that a better result can always be obtained by coding vectors rather than scalars [15, 16]. Compression schemes such as transform coding (see section 2.3.7) which preprocess vectors to decorrelate the input samples are still sub-optimal in a Shannon sense because coding is still being performed on scalars. Coding vectors will, *in theory*, give better performance.

Very little research into the design of vector quantizers was done until in the late 1970s when an algorithm of Lloyd was found to extend easily from scalar to vector quantization. Since then there has been a great deal of work enhancing the basic technique and tailoring it for specific applications such as speech and image coding.

A vector quantizer is defined as a mapping Q of K-dimensional Euclidian space R^K into a finite subset Y of R^K. Thus,

$$Q : R^K \rightarrow Y \tag{2.5}$$

where $Y = (\underline{\hat{x}}_i;\ i = 1, 2, \ldots, N)$ is the set of reproduction vectors and N is the number of vectors in Y.

With respect to image coding the vectors to be quantized are sub-blocks of an image, often 3×3 pixels or 4×4 pixels in size. The set Y is called the codebook, and the reproduction vectors are called codewords. VQ can be split up into two functions: an encoder which views the input vector $\underline{x}$ and generates the address of the reproduction vector specified by $Q(\underline{x})$; and a decoder which uses this address to generate the reproduction vector, $\underline{\hat{x}}$. The mapping $Q(\underline{x})$ is designed to minimise the 'error' between $\underline{x}$ and $\underline{\hat{x}}$. Commonly the measure used (referred to as the *cost function*) is the mean square error distortion. Figure 2–4 shows a block diagram of a simple vector quantizer.

There are many variations on the basic vector quantization algorithm, and these are covered in more detail in chapter 3. Some are aimed at reducing the computational complexity, as a simple vector quantizer must perform a full search of the codebook and this is computationally expensive. Examples of such methods are Tree-Searched, Classified and Multi-Stage VQ.

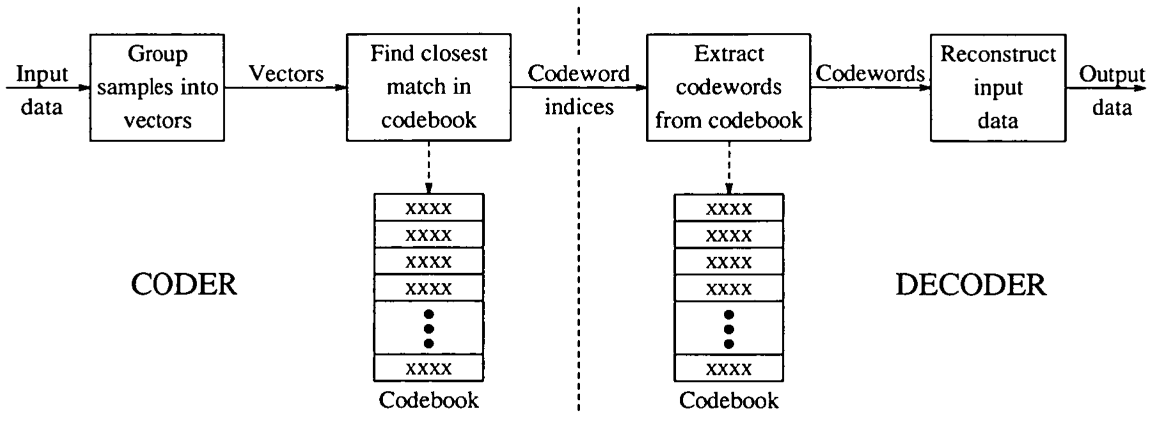


Figure 2–4: A Simple Vector Quantizer

VQ codebooks are generated by techniques which optimise them for a given training set. Codebook generation techniques are usually iterative and computationally expensive. Codebooks are optimal only for the data that they have been trained on.

It is also possible to combine VQ methods with other image compression schemes, such as transform coding or DPCM. A full review of VQ codecs and codebook design techniques is given in Chapter 3.

2.3.11 Future Codecs

To achieve very low bit rate image transmission, it will be necessary to develop techniques which are more intelligent than the methods outlined above. These new methods will need to view images as a collection of objects and extract and transmit the minimum number of parameters which will accurately describe them. At present, these methods are limited by the lack of algorithms to extract and process objects. Also, if these were available, the processing power required to construct a real-time system for coding image sequences is unavailable, or at least would be prohibitively expensive. However, the future is bound to see the development of such methods with the advance of the knowledge and technology which they require. Two methods which have the potential to code images in this way are model-based coding and fractal coding which are both described below.

Model-based coding

In perceiving images a human viewer processes them to extract the relevant data which will allow the scene content to be understood. Model-based coding (MBC) uses knowledge of the human visual system to extract enough information to describe an image scene.

The aim of MBC is to describe the content of a scene in such terms that a good reconstruction can be generated at the receiving end. This is done by modelling the objects in the scene and sending only a limited number of parameters which is sufficient to describe these objects. Objects will be accurately modelled if the 2-D image is processed to obtain parameters which describe the object in 3-D.

To simplify any MBC system, it is necessary to limit it to consider only objects which it already has knowledge about. This obviates the need to generate new 3-D models for new objects and the transmission of this information to the receiving end. This is sometimes called 'Semantic Image Coding', and is illustrated in figure 2–5.

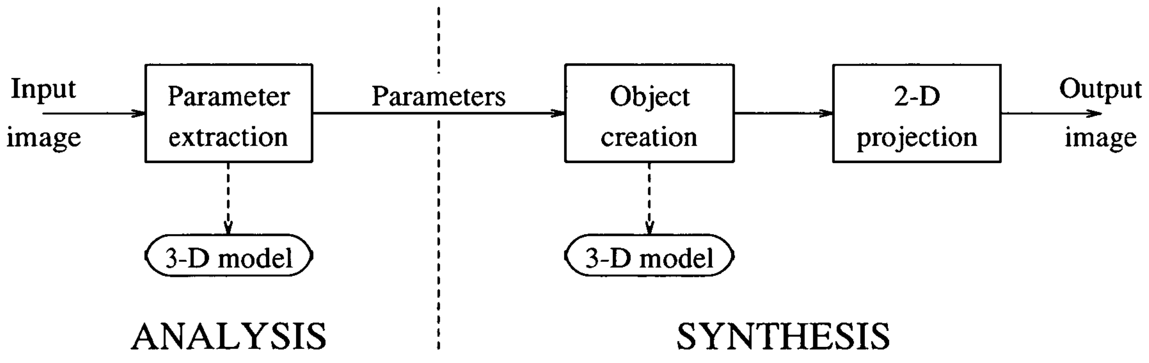


Figure 2–5: Semantic Image Coding

There has been much interest in modelling the human head and shoulders images of the type that are of use in videotelephony [17, 18]. In such a system the basic properties such as the shape of the head and facial features, colours and textures are extracted and transmitted. After this has been done, the coder and decoder both possess the 'static model' which describes the head and shoulders as they are at the start of the image sequence. Subsequently, parameters required to describe changes to this model are extracted and transmitted. These parameters are motion vectors defining the rotations and translations that the head has undergone, plus parameters which describe facial actions.

One commonly used method is to map a 2-D head and shoulders image onto a 3-D wire-frame model [19-25]. Facial features and movements are therefore defined by the positions of the vertices in the wire-frame model. These models vary in their complexity. A model with more vertices will render an image more accurately, but it will be more difficult to extract these points from the 2-D image and also require more information to be transmitted for each image. A relatively simple wire-frame model, called CANDIDE [25], is shown in figure 2–6.

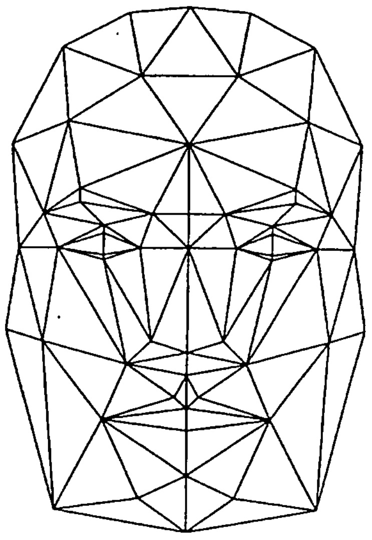


Figure 2–6: The CANDIDE wire-frame model

At present, it has been shown that animated sequences of faces can be produced effectively with parameterised models, giving enough detail to display emotions [19]. However, much work needs to be done on the algorithms required for parameter extraction. For example, there is no recognised method for locating the head and shoulders and separating it from the background [26]. Also, if a system is not to be limited only to face and shoulders images, the ability to generate models for new objects entering the scene must be provided. This, too, will necessitate large strides to be made in algorithm development. Once these problems are overcome, MBC has excellent potential for very low bit rate image coding.

Fractal Compression

Fractal mathematics offer the promise of image compression ratios in excess of 10,000 to 1 [27, 28]. Through fractals, there is the ability to describe an image in only a few hundreds or thousands of bytes. Initially, attention was drawn to fractal geometry because of its ability to simulate natural images such as landscapes and leaf patterns. Now attention is turning to the ability to take real images and imitate them with a collection of geometrical shapes generated from fractals.

The compression process starts with the segmentation of the image into areas which are then considered separately. Each segment is then compared to a library of fractals. This library does not contain the actual shapes, but sets of numbers called *iterated function system* (IFS) codes. These codes are sets of transformations and associated probabilities. The transformations, applied to a point in a random fashion governed by their respective probabilities, produce fractal shapes. When an IFS code is found that produces a shape that imitates the segment to be coded, then that code may be kept in place of the pixels which describe that segment. After this has been done for each segment the entire image can be described by a number of IFS codes, enabling enormous compression ratios.

Encoding images in this way is massively computation-intensive, involving a large number of transformations for each comparison and a very large library of IFS codes to be searched. Decoding is much simpler, and feasible in real-time with the aid of custom hardware. However, any real-time system for encoding images in this way is a long way from realisation.

2.4 Coding Standards

The emergence and adoption of open, international standards is vital if the benefits of image compression technology are to reach the consumer market. Three proposals are achieving widespread recognition and these are : the Joint Photographic Experts Group (JPEG) standard for still picture compression; the Consultative Committee on

International Telephony and Telegraphy (CCITT) recommendation H.261 for video teleconferencing; and the Moving Pictures Experts Group (MPEG) standard for motion picture compression on Digital Storage Media (DSM). All three are hybrid techniques, meaning that they incorporate a number of the coding methods outlined in 2.3. Each has a discrete cosine transform (DCT) coder at its core and they are described in more detail below.

The effect of these standards is to focus the efforts of many industrial concerns into the production of computing engines which perform image compression in compliance with these standards. Products are now becoming available which use all of these standards and growth in their usage will help to open up new applications for image compression. This in turn will drive a demand for better image compression techniques and as ongoing research improves these methods, new and better standards will be forged.

Before the emerging coding standards are described, it is necessary to introduce the standards for the communications networks and image formats which provide the structure on which these standards are based.

2.4.1 Communications Standards

Communications networks all over the globe are rapidly being modernised to make use of the advances in modern technologies. In particular, electromechanical switching and control systems are being replaced with modern digital exchanges, and copper analogue trunk transmission lines with very high bandwidth optical fibres [29]. As yet, though, there has been no great change in the services offered to the average user [30]. In the next few years the emergence of the Integrated Systems Digital Network (ISDN) will alter this.

The ISDN is described in a complex set of proposals made by the CCITT, called the I-Series of recommendations [31, 32]. These define a network of end-to-end digital connections and envisage a wide range of services : not only voice, but any digital data, including video. Users will have access to this network by a limited set of standard interfaces.

Two forms of interface are specified in the ISDN. *Basic access* provides $2B + D$ channels, where B and D channels are of 64kbit/s and 16kbit/s capacity respectively. Above this is another level of access, called *primary access*, which provides $pB + D$ access. Here B and D are both channels of 64kbit/s capacity, and p is a number from 1 to 30 in Europe and 1 to 23 in North America.

In the United Kingdom, British Telecom is now undertaking the implementation of ISDN-2, which incorporates some of the I-Series recommendations [33]. Users of this service are offered basic access only : the existing copper pair cabling which connects users to their local exchange has sufficient bandwidth capability to provide this access, but not primary access. So, in the near future, domestic and small business users will have only basic access, with primary access being limited to a much smaller number of high capacity users. In consequence, image coding techniques for basic access capacity lines will continue to be important for some time.

Further in the future, the ISDN is seen as evolving from the present recommendations, known as Narrowband ISDN (N-ISDN), into Broadband ISDN (B-ISDN) [34-37]. B-ISDN differs mainly in that access will not be constrained to integer multiples of 64kbit/s connections : it will employ packet switching to allow variable bit rate data. Thus line usage will be more efficient, cutting the costs to the customers. Also, many services envisaged for the ISDN, for instance video coding, are more amenable to a variable bit rate implementation. With the arrival of B-ISDN there will be a need to develop a new generation of video standards which use a variable bit rate.

2.4.2 Video Standards

There are currently three major television signal standards, namely PAL, NTSC and SECAM [38]. Each is incompatible with the others and this situation has led to the development of an international standard format. This was proposed by the CCITT in parallel with its H.261 video codec, and is called the Common Intermediate Format (CIF) [1].

CIF is a non-interlaced, digital representation of video information. It consists of three components : a luminance signal (denoted Y) at a resolution of 352×288 pixels, and two chrominance signals (C_r and C_b) at half this resolution, 176 $\times$ 144 pixels. The frame rate is 29.97 frames/second and each pixel is quantized to 8 bits.

Also defined is 'Quarter CIF' (QCIF) which has each component subsampled by 2:1 in each dimension, reducing the amount of data in a frame to a quarter that of CIF. QCIF arose out of the realisation that transmission of full CIF format down a single ISDN line would require greater compression ratios than were feasible at a reasonable cost. Also, QCIF offers ample resolution for good quality head-and-shoulders images of the type which would generally occur in face-to-face videotelephony.

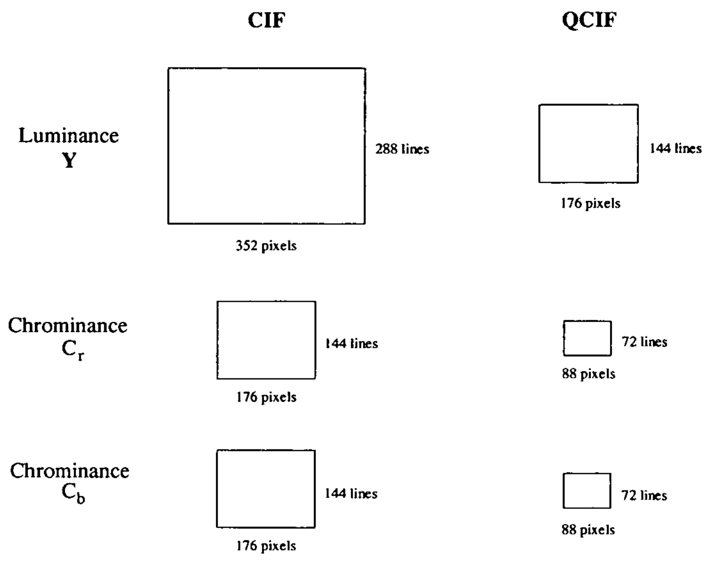


Figure 2–7: CIF and QCIF formats [1]

Colour representations and coding of colour images

A colour image is most commonly represented by each pixel having a value for each of the three primary colours : red, green and blue. It is in this form - the *RGB* form - that image data is specified for display purposes where each component will be used to control one of the electron guns in a colour cathode ray screen. This is not, however, the most appropriate choice of coordinate system for image coding, as there will be a high level of correlation between the three coordinates [39].

Coding is usually performed in the luminance-chrominance space. This is the representation that has been chosen by the CCITT for the CIF video format mentioned above. This relates the luminance, Y, and chrominances, C_r and C_b, to RGB in the following way :

$$\begin{aligned} Y &= 0.299R + 0.587G + 0.114B \\ C_r &= 112(R - Y)/0.701 + 128 \\ C_b &= 112(B - Y)/0.806 + 128 \end{aligned} \tag{2.6}$$

As a result of this transformation, most of the energy of a colour image is contained in the luminance component. This allows the chrominance signals to be subsampled by 2:1 in each dimension without significant loss of information. This means that there will be twice as much data in the luminance component as in the two chrominance components combined.

Although the conversion from RGB to YC_rC_b space has reduced the correlation between the three coordinates, there is still sufficient correlation to aid some coding methods [40]. For instance, in the H.261 video coding algorithm (see section 2.4), motion estimation is performed only on the luminance component, and motion compensation on the chrominance signals is calculated from the luminance result.

In many cases though, the three signals have become sufficiently independent that coding of each signal can be done separately. This means that the techniques used to code a luminance signal can be applied in an identical fashion to the two chrominance signals. There is, therefore, no need to spend a great deal of effort in developing different methods for coding colour signals as these methods will usually be similar to those used to code monochrome images. For this reason, the work in this thesis is concerned only with the development of algorithms for coding monochrome, or luminance only, images.

H.261 Motion Video Codec

The CCITT recommendation H.261 is the culmination of standardisation work which began in Europe in the late 1970s [41-43]. European industry identified the need for collaboration and this led to the project called COST 211 (Co-Operation in Scientific and Technical research) [44]. This work, and that done later under CEPT (Conference of European Posts and Telecommunications), resulted in a 2Mbit/s codec for transmission of PAL television signals which was produced in the early 1980s. Similar work in North America developed a codec for transmission of NTSC television signals at 1.544Mbit/s. The CCITT brought together these two standards into recommendations H.120 and H.130, for which hardware was developed and continues to be sold.

In parallel with this, proprietary codecs were being produced by telecommunications manufacturers independently of the CCITT's recommendations. However, it was realised that this was not a situation which benefited the market as a whole and all the world's leading players came together in the CCITT Study Group XV Specialists Group. The aim of this group was to develop a worldwide standard for videoconferencing at multiples of 384kbit/s up to 2Mbit/s [45-47]. This was to be followed with a separate recommendation for multiples of 64kbit/s. However, the decision was made to join the two and form a single '$p \times 64$kbit/s' codec, and this became recommendation H.261.

The collaborative work which created recommendation H.261 was characterised by the use of a *Reference Model* (RM). This was a detailed description of the proposed coding algorithm. Contributing bodies would simulate this algorithm in software and propose methods of improving the algorithm. Convincing improvements would be included in an updated version of the RM, and in this way the algorithm evolved from RM1 to RM8 (RM8 essentially describes H.261).

The decision was taken to adopt CIF as the video format to be used in the H.261 codec, and this is described in section 2.4.2. Thus the video source (from PAL, SECAM or NTSC) will require preprocessing to convert it to CIF before coding. Similarly, output from the decoder will have to be post-processed to convert it from CIF to the format required by the video output hardware. However, the use of CIF has a number of advantages:

- Communication between areas operating different video standards (say a North America to Europe link) is no different to an intra-regional connection.
- Hardware design for the core of the codec is simplified as it is the same worldwide.
- It was easier to reach agreement on the codec when the coding of only one format had to be considered.

The H.261 algorithm is shown in figure 2–8. It can be described as a hybrid DCT-DPCM coder with motion compensation. Blocks of 8×8 pixels from the current video frame are differenced with the output from the predictive coder. This predictor uses a copy of the previous decoded frame that will exist at the decoder, and applies motion compensation to each block of the image. The result of the difference (the *prediction error*) is DCT coded and quantized. The quantized coefficients and the motion vectors from the motion estimation are losslessly coded by a variable length coder and pass into a buffer. The output from this buffer is optionally coded for error correction before being transmitted to the receiver.

Information flows into the buffer at a variable rate, this rate being dependent on the accuracy of the predictive coder. In particular, the prediction errors will increase during periods of rapid scene change. However, communication is across a fixed bandwidth line and thus it is essential that the data rate is close to, but never more than, the channel capacity. Thus as the buffer becomes over-full or under-full it can change the step size of the quantizer to lower or raise the data rate.

The algorithm as it stands allows a degree of flexibility in the construction of the codec and a number of options are available. For instance, a decoder can demand of an encoder that it use reduced resolution (QCIF) and reduced frame rates (15, 10 or even 7.5 frames/s). Also, some of the more complex coding blocks are optional : notably, motion compensation, a very intensive process to perform at the coder, may be omitted. Any decoder must be able to cope with a motion compensated signal, but this is far simpler than the encoding process. This flexibility will allow a range of hardware to be produced for different applications, at different costs.

JPEG Still Picture Codec

JPEG convened for the first time in 1986 with the purpose of defining a standard technique for still picture compression [49, 50]. Work on picture coding began in 1982 when the International Standards Organisation (ISO) set up its Working Group 8 (WG8), and alongside this work the CCITT set up its New Image Communication Group (NICG) to investigate similar questions. JPEG was formed when experts from

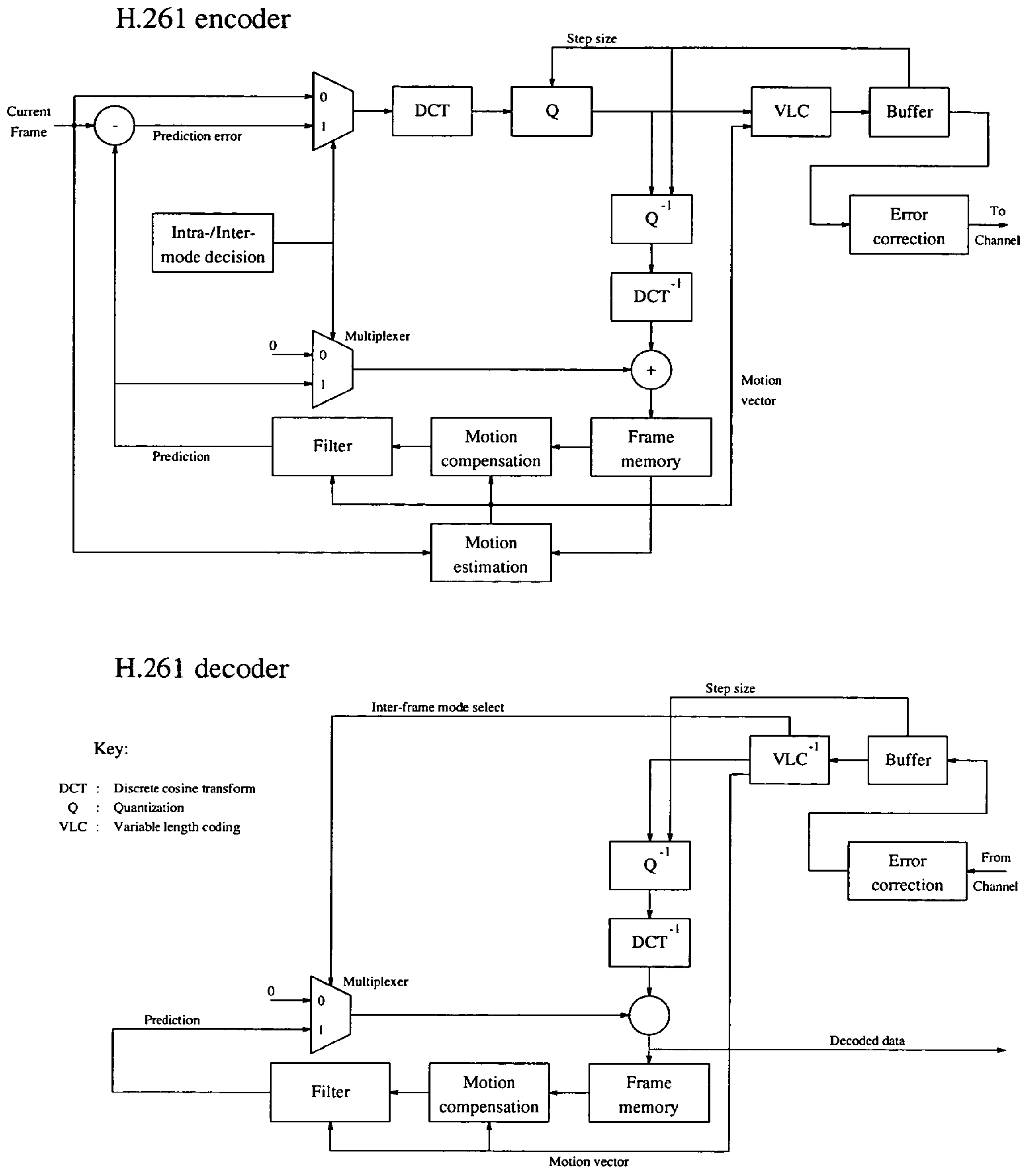


Figure 2–8: The H.261 Algorithm [48]

these two groups met, and at this time it included representatives from many of the world's leading telecommunications and computer companies.

From their pooled knowledge, the group first set about defining the requirements that they felt the standard should meet. They decided to adopt the International Radio Consultative Committee (CCIR) digital studio television recommendation 601 as the test picture format. This format has an 8-bit luminance component at a resolution of 720×575 pixels and two 8-bit chrominance components at half the horizontal resolution, giving an average of 16 bits/pixel and a total of 828kbytes.

They set a requirement for the compression to deliver an accepable image at 0.75 bits/pixel, which would mean a transmission time of under 5 seconds on a 64kbit/s ISDN line. Also, the coding scheme would be required to provide a progressive build-up of the image. On restricted bandwidth communications lines this allows a crude picture to be provided very quickly, and this can be improved in stages until a satisfactory quality is achieved. JPEG specified three stages with compression ratios of 0.25, 0.75 and 4.0 bits/pixel.

Having specified these requirements the participating bodies were then invited to develop a codec for a competition which was held in July 1987. Of the twelve proposals three were chosen for further investigation : these were Adaptive Discrete Cosine Transform (ADCT), Adaptive Binary Arithmetic Coding (ABAC) and Block Separated Progressive Coding (BSPC). The results from the initial selection procedure were used to help redefine the specifications for a final selection procedure held later that same year. At this meeting the ADCT technique emerged the clear winner [51, 52].

The chosen technique was then further categorised to consist of Baseline, Extended and Independent Functions. The Baseline Function compresses images having 8 bits/pixel per colour component, and operates only in sequential mode where the whole image is processed in a single pass. Progressive image build-up is performed by the Extended Function, which can code images having up to 12 bits/pixel per colour component. The Independent Function used DPCM and entropy coding to attain lossless compression ratios of approximately 2:1.

The baseline algorithm is shown in figure 2–9. The algorithm has three stages : the image is split up into blocks of 8×8 pixels to which a 2-D DCT transformation is applied; the DCT coefficients are quantized; and these coefficients are then Huffman coded, a common form of Variable Length Coding (VLC). At the decoder, these processes are reversed.

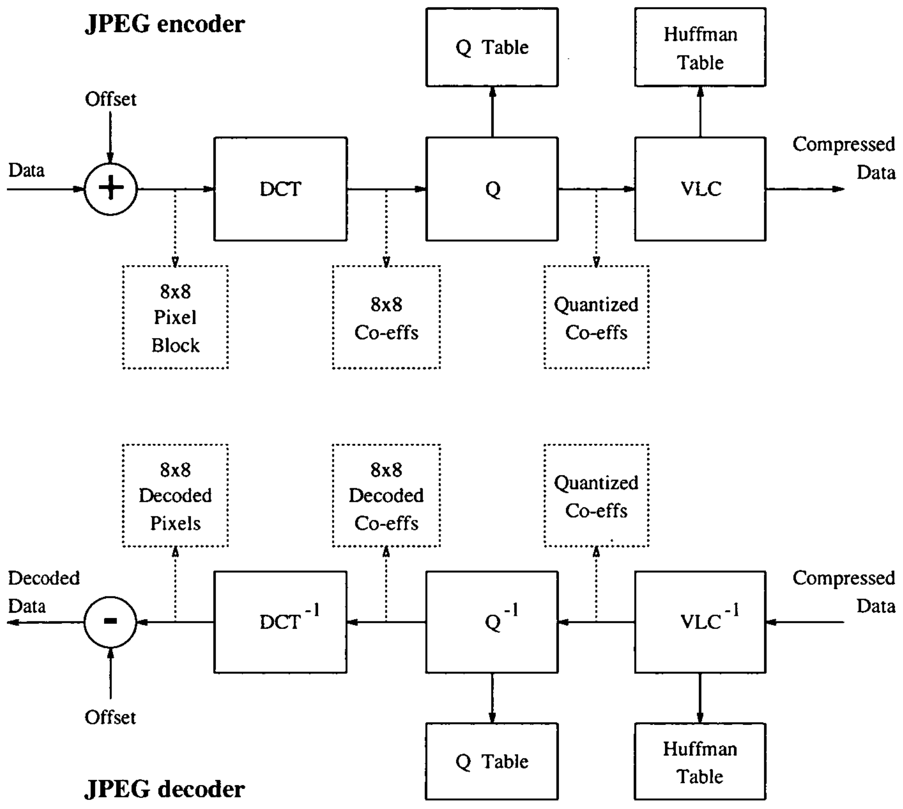


Figure 2–9: JPEG Baseline Algorithm [48]

MPEG Motion Video Codec for Digital Storage Media

The ISO set up MPEG in May 1988 to develop a standard for full-motion video compression for use with the digital storage media (DSM) of that time [53]. These media were compact disc (CD) and digital audio tape (DAT) which were originally developed for storing audio information, but with a throughput of approximately 1.5Mbits/s they have plenty of potential for storing coded moving pictures.

After over two years of standardisation work, MPEG published a draft proposal in September 1990. As with JPEG, the standardisation had involved a competitive phase to evaluate the effectiveness of a number of techniques for the requirements of this specific application. After this, the bodies which form MPEG collaborated to develop the chosen scheme further.

The specific requirements for DSM applications include :

- Reverse playback
- Fast forward/reverse playback
- Single frame access
- Random access

The standard has chosen to specify that the input sources are to be of SIF format, limited to 352 $\times$ 288 pixels at 25 frames/s. The algorithm itself is a hybrid scheme with motion compensated interframe prediction and DCT coding, and is largely similar to the H.261 standard. The biggest difference between these two schemes is that MPEG images can be coded not only with prediction from the past but also from the future.

Figure 2–10 shows how an MPEG sequence is made up of intra-coded (I), predictive coded (P) and bi-directionally predictive coded (B) pictures. The bi-directional prediction results in a more complex codec and increases coding/decoding delay but this is unimportant in DSM applications. Intra-coded pictures are inserted periodically to enable fast forward/reverse playback.

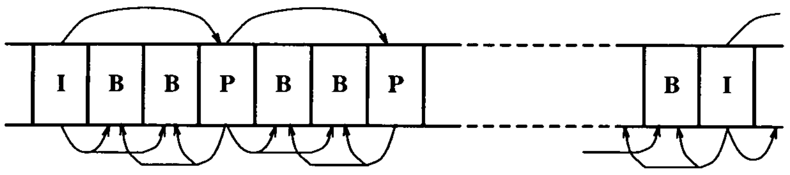


Figure 2–10: An MPEG Sequence

Although the MPEG standard has not been fully ratified yet, work has begun on a new standard for DSM. Dubbed MPEG-2, this standard will allow for the data throughput rates of approximately 10Mbits/s which the new generation of DSM are expected to offer.

2.5 Implementation of Coding Algorithms

For the purposes of research into algorithm development, all work is done in software on generic hardware platforms such as Sun Unix workstations. Images are drawn from disk storage, coded, with the results displayed on the computer screen to view the performance of the coding process. This allows for a great deal of flexibility in algorithm specifications as they are developed.

To create systems which will perform coding algorithms in real-time, it is necessary to use hardware dedicated to only that task. The options for doing this are application-specific integrated circuits (ASICs) or programmable digital signal processors (DSPs).

ASIC technology enables the design of silicon chips which are dedicated to the task of performing one coding algorithm, or perhaps even only one functional block of a coding algorithm. In the last few years, silicon vendors have started to offer chipsets which have been designed to implement the functional blocks of the coding standards outlined in section 2.4. These include chips which convert raster data to blocks of data, colour space (RGB to YC_rC_b) converters, and DCT/inverse DCT chips. The chipsets can be designed into a solution which implements one of the video standards on one or more boards.

General purpose DSP chips become more powerful every year, with the launch of chips that operate at ever-faster clock rates and include many innovations to speed up their operation. Most image coding algorithms, however, are so compute-intensive that they are still beyond the realms of single-processor DSP solutions. It would be possible to design systems which performed coding algorithms with multiple DSPs, and these would offer the attraction of being flexible due to the programmable nature of DSPs. However, this would be difficult to do efficiently, and would be very expensive to produce.

Custom ASIC solutions have the advantage of the best performance possible, due to their dedicated nature, and, more importantly, the advantage of cheaper cost than other solutions when produced in volume. They do, however, absorb large amounts of development time.

Many of the methods of image coding have proved to be amenable to an ASIC solution, as can be seen with the arrival of chips to perform all of the primitives of the image coding standards. Custom fractal ‘engines’ have been demonstrated which allow fractal encoded images to be decoded at rates of a few frames per second [27]. Another coding method not used in the standards, but which has also proved to be highly suitable to an ASIC implementation, is VQ. A number of researchers have produced prototype ASIC designs for different VQ algorithms [54-66]. From this research it is clear that VQ has a highly regular nature which allows it to gain greatly from being implemented in custom silicon.

2.6 Conclusions

This chapter has given a broad overview of the field of image coding. It has introduced the reader to the need for image compression and the multitude of compression methods which have been developed over the years.

In very recent times we have seen the emergence of image coding standards that have been set through industrial collaboration. There is a great deal of momentum behind the transform coding methods which form the basis of these standards. However, it is widely recognised that the performance of these methods could be bettered by other techniques. Research continues with the aim of developing alternative methods which can significantly outperform existing standards and could form the next generation of standards. Vector quantization is one of the methods which attracts continued research work.

The case for vector quantization is built on very a solid base as the coding of vectors, rather than scalars, offers theoretically better performance. Vector quantization is also a relatively straightforward technique, but one which can take many different forms for different applications. It has also been shown to be amenable to VLSI implementation, with a number of researchers quoting simulated and actual results for vector quantizer processors. In this light, it was decided to concentrate the study of this thesis on vector quantization methods.

Chapter 3

Vector Quantization Essentials

3.1 Introduction

Vector Quantization has been studied extensively as a data compression technique for more than a decade now, but the theory which supports it dates back to communications theory developed by Shannon in 1948 [67]. It was not until 1980 that Linde, Buzo and Gray [68] discovered a good design technique for vector quantizers, and attention became focussed on its potential for application in the field of image compression.

Rate-distortion theory, a branch of information theory, gives us the result that source coding with a fidelity criterion can always be achieved with better performance by coding vectors rather than scalars. The corollary of this is that vector coding at a given rate will achieve better distortion performance than scalar coding at the same rate. This result has been known since Shannon's published work in this field, but it has not been until recently that practical use has been made of vector codes. This has been due to the lack of appropriate design methods.

The breakthrough in vector quantizer design came with the publication of the design technique developed by Linde, Buzo and Gray. Their method was derived from a design technique for scalar quantizers by Lloyd [69] first published in 1957. Lloyd describes an algorithm for PCM quantizer design which uses knowledge of the source statistics to choose *quanta* which minimise the noise power in the quantized signal. A natural and intuitive extension of this provides an efficient algorithm for the design of good vector quantizers. This is now known as the LBG algorithm (after its authors) or the Generalised Lloyd algorithm. It continues to be the most common method for designing codebooks in use today.

The LBG algorithm requires the definition of a simple vector quantizer. This chapter begins with this formal definition, which is presented in section 3.2.

Alternative vector quantizer systems have been developed to try to reduce either the computational complexity or memory requirements compared with simple VQ. These efforts have produced a variety of systems with different costs and performances. VQ has been applied to a number of different image compression problems, and the variety of methods available allow the choice of a technique which is best suited to the application in question. Excellent reviews of vector quantization have been written by Gray [15] and Nasrabadi and King [70]. Those vector quantization schemes with application in the image coding field are introduced in section 3.3.

The task of codebook design is the selection of an optimal set of vectors to be used to code the input data. Most of the methods in use today are based on the LBG algorithm. This, and other methods for codebook design, are covered in section 3.4. However, the production of optimal codebooks remains as a problem worthy of research, as the difficulties it poses have by no means been overcome. As a result of the difficulties arising in codebook design, a new codebook design method has been developed by the author. This is presented in section 3.5 with experimental results which illustrate its effectiveness at overcoming the difficulties involved.

The distortion-rate function defines a minimum distortion which can be obtained for a given compression ratio [71]. The limit of this function is reached as the dimensionality of the vector tends to infinity. However, in a simple vector quantizer, which must compare each input vector with each vector in the codebook, the search (*i.e.* computational) complexity and memory requirements increase with the vector dimension. Vector quantizer design is subject to the trade-off between the costs of computational complexity and memory requirements against the aim of achieving a given compression rate and fidelity performance. A discussion of these trade-offs is given in section 3.6.

3.2 VQ Definition

This section defines a simple vector quantizer as presented by Linde, Buzo and Gray. As they showed, a vector quantizer is an intuitive step forward from a scalar quantizer.

A scalar quantizer maps an input value into one of a finite number, say N, of reproduction values. These reproduction vectors can be known collectively as the *codebook*, and individually as *codewords*. The encoder part of the quantizer examines the input and determines the codeword which will produce the minimum distortion when used to reproduce the input. The index to that codeword is then passed to the digital channel. The decoder reconstructs the input by looking up the codeword for that index in the codebook, which is known to the decoder also.

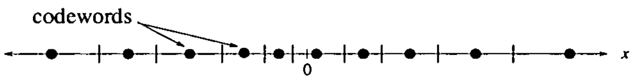


Figure 3–1: Scalar Quantization of the real line into ten regions.

The transmission rate of a scalar quantizer, r_s, is given by

$$r_s = \log_2 N \quad \text{bits/sample} \tag{3.1}$$

Vector quantization generalizes this scheme to the coding of groups of samples, or *vectors*, rather than single discrete samples. This can be visualised as the partitioning of the vector space into regions, each with its own reproduction vector. This is shown for two dimensions in Figure 3–2.

A vector quantizer is defined, as has already been stated in section 2.3.10, as a mapping Q of K-dimensional Euclidian space R^K into a finite subset Y of R^K :

$$Q : R^K \rightarrow Y \tag{3.2}$$

where $Y = (\underline{\hat{x}}_i;\ i = 1, 2, \ldots, N)$ is the set of reproduction vectors, or codebook, N is the number of vectors in Y, and K is the vector dimension.

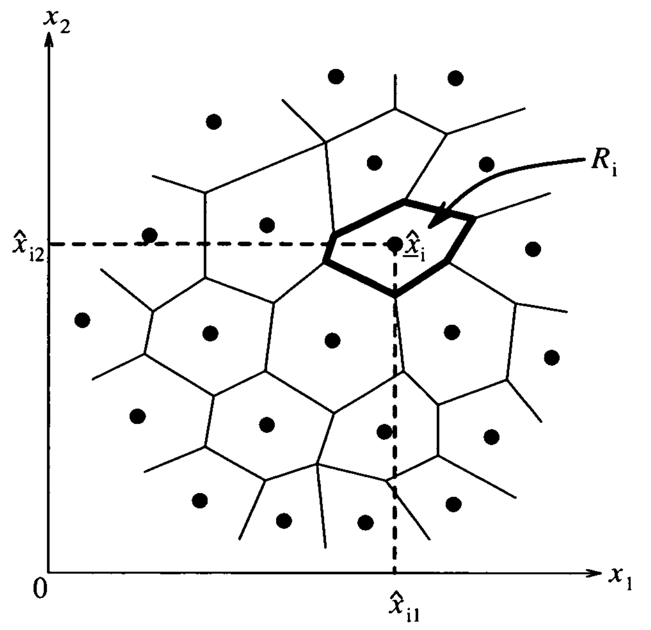


Figure 3–2: Partitioning of two-dimensional space into a number of regions. All vectors, $\underline{x}$, in the region R_i will be quantized as $\hat{\underline{x}}_i$, the codeword which is the centroid of region R_i. The regions can be very different shapes.

The transmission rate of a vector quantizer, r, is given by

$$r = \frac{1}{K} \log_2 N \quad \text{bits/sample} \tag{3.3}$$

Associated with the quantizer in R^K there is a partition

$$R_1, R_2, \ldots, R_N$$

where

$$R_i = Q^{-1}(\hat{\underline{x}}_i) = \{\underline{x} \in R^K : Q(\underline{x}) = \hat{\underline{x}}_i\} \tag{3.4}$$

With this definition it follows that

$$\bigcup_{i=1}^{N} R_i = R^K \text{ and } R_i \cap R_j = 0, \text{ for } i \neq j. \tag{3.5}$$

The quantizer is thus specified by the codebook, Y, and the corresponding partition, $\{R_i\}$.

In practice, a quantizer is actually composed of two functions : a *coder* and a *decoder*. The coder, γ, maps R^K into the index set J, and the decoder, β, maps J onto the output set Y, where $J = \{1, 2, \ldots, N\}$. The coder and decoder are defined by the mappings

$$\gamma : R^K \rightarrow J \text{ and} \tag{3.6}$$

$$\beta : J \rightarrow Y \tag{3.7}$$

and the quantizer Q can be defined as the concatenation of the two,

$$Q = \gamma \cdot \beta \tag{3.8}$$

3.3 VQ Codecs for Still Image Compression

This section provides a survey of the many VQ codecs which have been applied to the coding of still images.

3.3.1 Tree-Structured VQ

The most commonly used alternative to simple VQ is Tree-Structured VQ (TSVQ). It imposes structure on the codebook to reduce the computational burden relative to that required for a full-search VQ. The structure enables a vector to be coded with a good match from the codebook, but not necessarily the best match, so the price of the reduced complexity is paid in sub-optimal performance.

The simplest and most commonly implemented form of TSVQ is the case where a binary tree is used [72-75]. This is shown in figure 3–3 and used as an example for discussion.

Each node is represented by a codeword, v_i, which is not in the reconstruction codebook but is a representation, or average, of the reproduction codewords below that node in the tree. At each node, the encoder chooses the closest of the two codewords in the next layer, and advances along that branch of the tree. This is repeated until the the highest node in the tree is reached. That node will be represented by a reproduction codeword, $\hat{\underline{x}}_i$, for which the index i is transmitted or stored for later reconstruction by the decoder.

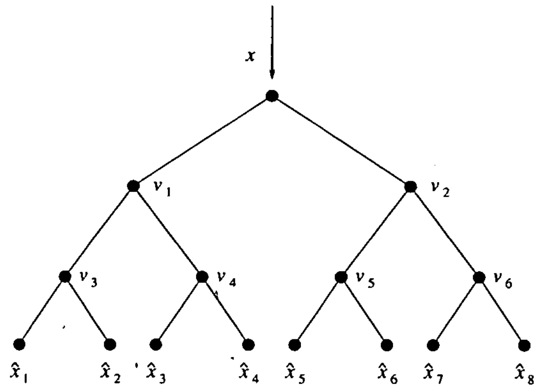


Figure 3–3: A Uniform Binary Tree for TSVQ

For full-search VQ, the number of multiply/add operations required to find the closest vector by a mean-squared-error criterion is NK, where N is the codebook size and K the vector dimension. For binary TSVQ this is reduced to $Klog_2N$ operations. This reduction in computation is most noticable when large codebooks need to be searched.

There are a number of demerits to this form of VQ. Most obvious is the inferior performance compared to full-search VQ, although this can be minimised by clever codebook and tree design. Also, there is a doubling of the memory requirements at the encoder, as the total number of nodes in the tree is $2N - 2$. But perhaps the greatest problem with TSVQ is the complexity of the codebook design.

The most obvious way to impose a tree structure on a codebook is to work backwards from an LBG designed codebook, grouping together close disjoint pairs of codewords and forming the ancestor node as the centroid of this pair. This, however, is a very difficult process to perform well, especially as the size of the codebook grows, which is just the time that the benefits of TSVQ are starting to have an effect.

Another possible design method is the *splitting* process. Here the initial codebook supplied to the LBG design process is formed by successively splitting one codeword into two from an initial single codeword at the centroid of the training set. However, the resulting tree structure would be largely lost in the LBG design process, as codewords move and unused codewords are replaced.

The problem of codebook design for TSVQ is further complicated when one is using a codebook adaption technique, such as those discussed in chapter 4, which may be necessary for motion picture applications [76]. The structure of the tree will be eroded as the codebook adapts and considerable effort is required to maintain a codebook which can perform well.

Of course, non-binary trees can be implemented as well. In this case each node requires a decision to be made between a number (greater than two) of possible branches. A simple example of this is classified VQ as presented by Ramamurthi and Gersho [77] and later studied by others [78]. Here the tree contains only one node : the vector to be coded is classified into one of a number of classes such as shade, horizontal edge, vertical edge, angled edge or mixed. For each class a codebook has been specifically designed and the relative sizes of the codebooks have been optimised. For instance, the sizes of the edge vector codebooks can be made relatively large so that the codec can code edges well, which is of great perceptual importance.

Non-binary trees do not save as much computational complexity as do binary trees. Their performance, however, is improved relative to binary trees, as shown in the above studies and other examples of k-dimensional tree structures [79]. However, they still cannot emulate the performance of a full-search algorithm.

3.3.2 Lattice VQ

In a similar way to TSVQ, lattice VQ imposes a structure on the codebook to enable an efficient search. This exploits the fact that for some sources the probability density function is highest, and roughly constant, in one region of the vector space. This region can then be filled with a uniform lattice of codewords.

Efficient coding algorithms have been developed for some different lattices [80-83]. As well as the search efficiency, lattice VQ offers a compression benefit in that the lattice coordinates can be used, which is more efficient than labels for every codeword. However, lattice VQ does have considerable disadvantages. Most notably, codebooks cannot be improved by any variation of the LBG algorithm without the structure of the lattice being destroyed, and an LBG designed codebook cannot generally be well approximated by a lattice. Also, it is doubtful that image or image sequence data has a probability density function which can be usefully exploited by a lattice of codewords.

3.3.3 Multi-Stage VQ

Multi-Stage, or Multi-Step VQ (MSVQ), is also similar in form to TSVQ. It differs in that at each layer of the tree, a single, small codebook is used rather than a different codebook for each node on that layer.

At the first layer, the vector is coded with a small codebook and an error formed between the vector and the chosen codeword. This error is then coded at the next stage by a second VQ, which has been trained to code the errors formed from the first stage. This process can be repeated with each successive stage attempting to code a finer error. A 2-stage implementation is shown in figure 3–4.

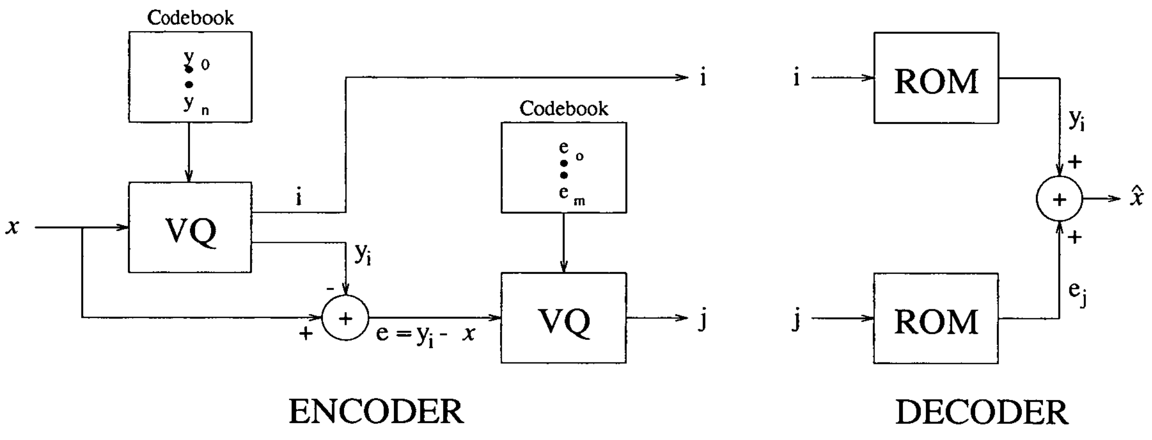


Figure 3–4: Multi-Stage VQ with 2 stages

The benefits of MSVQ are that the search incurs a similar computation cost to TSVQ, and that the memory storage requirements at the encoder are similar, or even less than, simple VQ. MSVQ is often used in conjunction with Hierarchical VQ (see section 3.3.5) [84, 85].

3.3.4 Product Codes

Product codes operate by separately encoding two features of a vector. This leads to a reduction in the the complexity of the search and the memory requirements at the encoder.

The codes are formed as the Cartesian product of the two coded features. The most commonly used product code in image coding is Mean/Residual VQ (MRVQ) [40]. In this case the mean of the presented vector is scalar quantized and the quantized mean is then subtracted from the vector. The residual, or error vector, is then coded with VQ. The vector is reconstructed by the Cartesian product of the scalar quantized mean and the vector quantized residual. This is shown in figure 3–5.

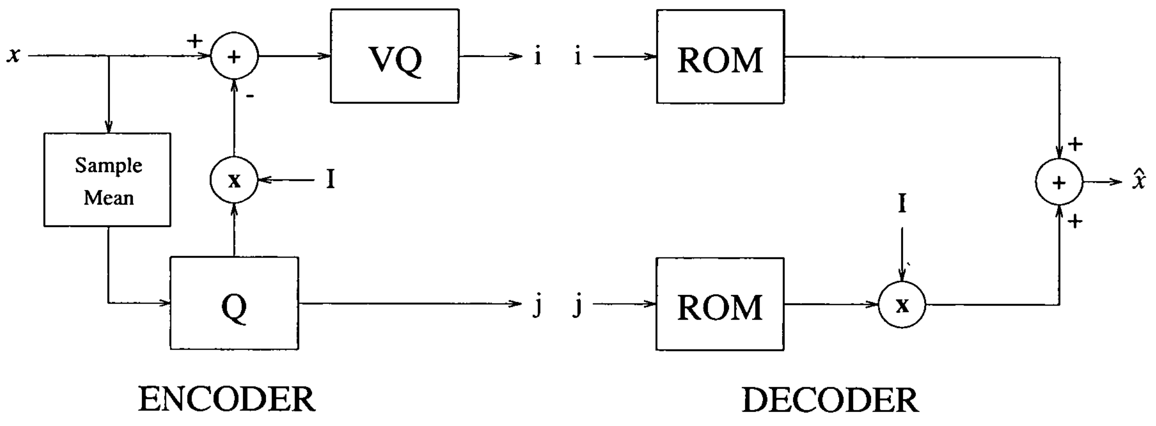


Figure 3–5: Mean/Residual VQ

A problem with MRVQ is that, after mean removal, most of the vectors to be coded lie very close to the origin while some – those containing edges – are very far away from the origin. This makes codebook generation difficult and in an effort to get round this Lee and Lee [86] propose adding a third stage to the encoding process whereby the vector is normalised after mean removal. This process places all the vectors on the surface of a hypersphere with the claimed result of better codebooks. The encoder in this case sends the quantized vector mean, quantized vector *amplitude*, and the quantized *shape* vector. This method offered a slight performance improvement over MRVQ.

3.3.5 Hierarchical VQ

What are known as *hierarchical* image coding methods have developed from the realisation that many images contain large areas of almost constant intensity. Images could be coded far more efficiently if the coding algorithm were able to recognise these areas and use different methods to code them from those used to code areas high in detail. In the last few years a number of different methods using VQ in a hierarchical algorithm have been proposed.

The basic process used in many of these schemes is quad-tree decomposition of the image. This segments the image into blocks of variable size, depending on the amount of detail present in that region of the image. The image is quartered or halved (assuming that we have a square image for simplicity) and for each subblock a homogeneity measure is calculated, such as the variance of the pixels. If the subblock is considered to contain greater than some threshold of detail it is again subdivided. This process is repeated until all blocks satisfy the homogeneity criterion or the smallest size block is reached and further subdivision is no longer possible. An example of the resultant quad-tree structure is shown in figure 3–6.

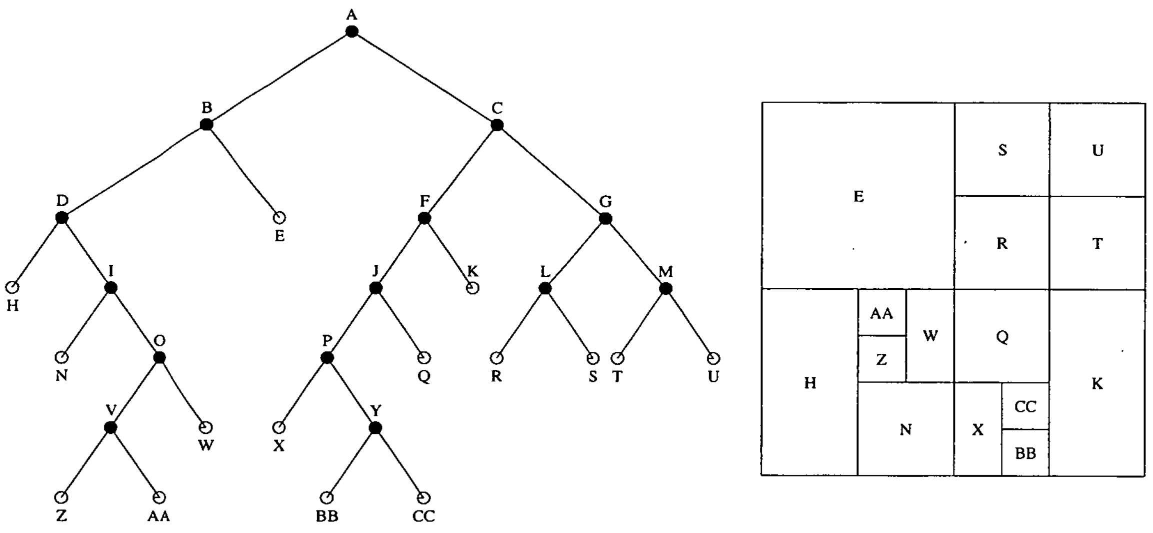


Figure 3–6: A Typical Quad-tree Image Decomposition

The quad-tree decomposition structure must first be coded so that the encoder knows how to reconstruct the image. Then the subblocks are coded, and a number of different ways of doing this have been proposed. Daly and Hsing [87] coded each subblock with Mean-Residual VQ (see section 3.3.4), using a different codebook for each size of subblock. Nasrabadi and King [70] proposed coding only the bottom two layers (2×2 and 4×4 pixel subblocks) with VQ, with larger subblocks being vector quantized in the transform domain. A similar method was also proposed by Vaisey and Gersho [88].

Quad-tree decomposition has also been used as a basis for other (non-VQ) schemes, notably transform coding [89, 90] and a binary interpolation technique [91].

A variation on the above methods is termed Pyramid Coding [92, 3] or Multi-Stage Hierarchical VQ (MSHVQ) [85, 84]. In these algorithms, a *difference* quad-tree is formed. Difference subblocks are calculated by subtracting the mean of the subblock above them (the subblock which has just been subdivided). This process is repeated all the way down to the smallest size subblocks and the result is the formation of a 'mean pyramid' and a 'difference pyramid' [92]. The mean pyramid and the difference pyramid can then be coded with VQ.

One of the benefits of Pyramid coding is that the image can be transmitted in such a way that it can be built up progressively at the decoder. As each layer is sent, the errors in the reconstruction are reduced, and the picture quality improves. This is very useful for still picture transmission over low-rate channels.

A problem with quad-tree decomposition is that it does not have the freedom to intelligently place large blocks in low detail areas in an optimal fashion. Other researchers have attempted to get away from this lack of flexibility by proposing the use of different segmentation algorithms. Boxer and Lee [93] developed unconstrained tiling VQ (UTVQ) which increases the tiling freedom to identify more large areas which are of low variance. In a similar fashion, Corte-Real and Alves [94] use a variable shape of block to achieve the same result. The problem with these methods is greatly increased complexity in the segmentation process, and the fact that the quad-tree structure to be transmitted is much more complex.

Another, rather different, hierarchical coding scheme utilising VQ has been developed by Dixit and Feng [95]. In their method, the smallest size subblocks are coded using standard VQ. Groups of 4 labels are then VQ coded with 'address codebooks', which contain codewords representing the 4 codeword labels in the label map. This process is then repeated with the label map of address codewords. The image is fully coded by the top layer of address codewords. Very large address codebooks were required to obtain acceptable results, with the problems of codebook generation which this brings. Results on still images were claimed to be very good.

Hierarchical methods have also been effectively applied to image sequence coding [96, 97]. They are effective for this purpose because interframe differences have large areas of constant (near zero) information.

3.3.6 Adaptive VQ

All of the VQ techniques introduced so far do not have any memory of past events. Memoryless vector quantizers exploit the high correlation and spatial redundancy between neighbouring pixels, but totally ignore the spatial redundancy between neighbouring blocks. Vector quantizers are forced to use small blocks because of the difficulty of designing codebooks of large dimension, and the need for large codebooks to achieve good performance, which increases the computational complexity. Thus, due to the small block sizes in use, much information is unused by a memoryless vector quantizer.

One way of incorporating memory into a VQ is to use a number of different codebooks, and for a given input vector the codebook to be used can be pre-chosen to be best suited for that vector. It cannot be guaranteed, of course, that the chosen codebook is the best one for the current vector, but in making it conditional on the previous vectors presented to the encoder, we can increase the possibility that the best codebook is in use. More accurately, the chosen codebook is dependent on previous outputs from the encoder, rather than on previous input vectors : in this way, the decoder can track the state of the encoder without the need to transmit any side-information.

What has been described above is known formally as Finite State VQ (FSVQ). A FSVQ system is defined by a space, S, whose members are called states, and that for each state s in S a quantizer being defined by an encoder, γ_s, a decoder, β_s, and a codebook, C_s. The operation of the coder is controlled by the next-state function, f_{NS}, which defines which state the coder will change to, and is a function of the current state and the chosen channel symbol, v. So, for a sequence of input vectors, $\{\underline{x}_n \; : \; n = 0, 1, 2, \ldots\}$, and an initial state s_0, the state sequence s_n, the channel symbol sequence v_n and the reproduction sequence $\hat{\underline{x}}_n$ are defined by

$$v_n = \gamma_{s_n}(\hat{\underline{x}}_n) \tag{3.9}$$

$$\hat{\underline{x}}_n = \beta_{s_n}(v_n) \tag{3.10}$$

$$s_{n+1} = f_{NS}(v_n, s_n) \tag{3.11}$$

A FSVQ is illustrated in figure 3–7.

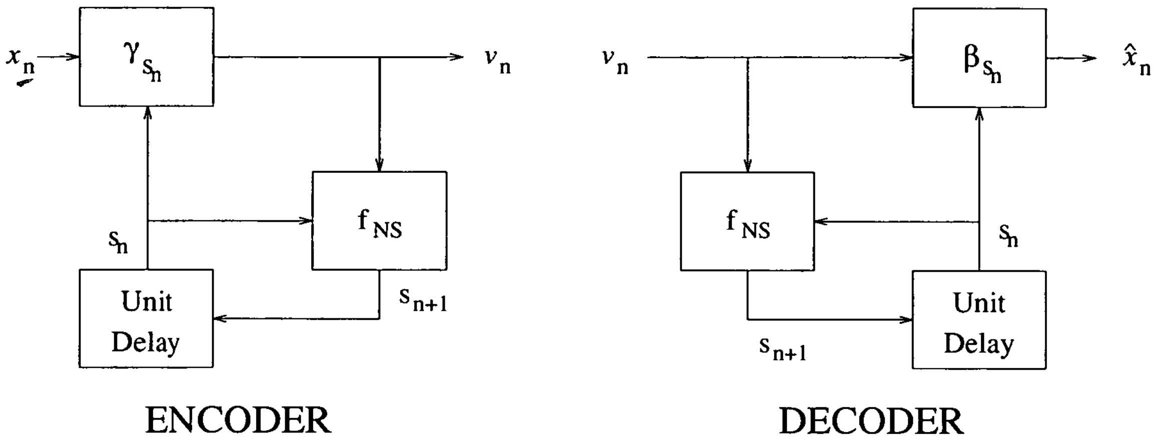


Figure 3–7: Finite State VQ

If the 'supercodebook', C, is defined as the union of all the state codebooks (where there are a finite number, N, of states),

$$C = \bigcup_{i=1}^{N} C_i \tag{3.12}$$

then hopefully for each input vector, $\underline{x}_n$, the current codebook C_s contains the nearest reproduction codeword in C. If this can be satisfied, FSVQ provides a zero-rate mechanism for reducing search time and bit rate. It is impossible, however, to guarantee that this condition can be met. Indeed, FSVQ design is exceedingly difficult to accomplish. As will be shown in section 3.4, LBG codebook design for simple VQ is an iterative process which takes an initial codebook and will converge to a good, if not optimal, result. FSVQ design requires a number of state codebooks, and an initial next-state function as initial conditions. From this point, the next-state function and the state codebooks must adapt together to iterate to a solution. The method by which LBG codebook generation converges to a good result cannot be applied to FSVQ with the guarantee of an improvement in encoder performance [15]. With FSVQ, a globally optimum solution may be impossible to find.

FSVQ is only one of the adaptive forms of VQ. Simpler means of incorporating memory into the process include vector predictive quantization which is simply a vector extension of DPCM, and a method proposed by Goldberg *et al* in which an entire codebook is constructed and transmitted for subparts of images. Product codes with predictive coding on the scalar quantized component have been used, such as that introduced by Baker and Shen [98, 99]. They also illustrate how the next-state function of FSVQ may be realised : they classify neighbouring blocks according to intensity, variance and motion (for interframe coding) and make the next state a function of these classes.

Another adaptive VQ algorithm is proposed by Nasrabadi and Feng [100] which they call Dynamic Finite State Vector Quantization (DFSVQ). In this method, a large supercodebook is dynamically re-ordered to hold the most likely codewords at the top of the supercodebook. The first N codewords in the supercodebook are searched, and if a codeword is found which is a good enough match (according to a pre-specified distortion) then its index is sent. If the match is not good enough, the entire supercodebook is searched and this fact is signalled to the decoder. Nasrabadi and Feng extend this idea to a technique which they call Address-Vector Quantization (A-VQ) [101-104]. Here they attempt to code a sequence of vectors with an 'address codebook' : if a sequence of codeword indices is present in the address codebook, the address codeword replaces the sequence of indices. In a similar fashion to DFSVQ, the address codebook is dynamically re-ordered after every quantization.

Adaptive VQ techniques extend naturally to the coding of image sequences. Interframe correlation provides more information which can be used to adapt the quantizer to operate best for its current vector. Interframe adaptive VQ is studied in more detail in section 4.7.

A problem with adaptive techniques is that channel errors can cause catastrophic errors. As the current state of any adaptive scheme is dependent on previous channel symbols, it is possible for the encoder and decoder to lose synchronicity if a channel error occurs on just one symbol. For this reason it is necessary to periodically send the current state of the encoder so that the decoder can reset itself to match this state.

3.4 Codebook Design

The task of codebook design is to find an optimal codebook, C , from the set of all possible codebooks. An optimal codebook is defined as that which yields the least average distortion when coding the input source. To generate this codebook, all codebook design methods use a *training set*. This is a set of vectors drawn from a single, or multiple, *training images*. The desired result is a codebook which is optimal for the training set used to generate it.

A training set defines a discrete probability density function of the data in the training images used to formulate it. The actual density function of the input source may not be known. Thus it is only possible to design an optimal codebook if the source statistics are known beforehand. In other words, it is possible to design an optimal codebook for a known image, but codebooks designed for this image will not be optimal when used to code any other image. In practice, it can be seen that codebooks designed for one image usually perform very badly when used to code another image. Sadly, a *global codebook* (of finite size), which can optimally code a full range of images, does not exist.

In certain applications, such as the coding of images for archival, it is valid to train a codebook for the optimal coding of a single picture. Much research effort has been directed towards the goal of developing design methods for this kind of codebook, and these are considered below. Other applications will not use these methods for their codebook design. One example is the coding of image sequences where, even in a simple scene, the image statistics change quite markedly over time. This topic is covered in Chapter 4. However, the techniques used in design of dedicated codebooks are of interest in any vector quantization research, as these techniques help in the understanding of the problem and can be extended to be used in other areas as well.

3.4.1 LBG

The Linde-Buzo-Gray (LBG) algorithm [68] is the most commonly used codebook design algorithm due to the fact that it was the earliest proposed method and consistently exceeds the performance of other methods in a variety of applications [105, 106]. It is an iterative technique which repeatedly moves codewords to cluster centroids in an effort to find a codebook which will display the lowest error when encoding the training data (*i.e.* a modified version of the K-Means clustering algorithm). The basic algorithm is as follows :

Step 1. Initialise the codebook.

Step 2. For each vector, $\underline{x}$, in the training set, calculate the Euclidian distance from it to each vector $\underline{\hat{x}}_j$ in the codebook.

$$d_j = \sum_{i=0}^{K-1} (x_i - \hat{x}_{ij})^2$$

where K is the dimensionality of the vectors.
The minimum distance selects the closest vector, $\underline{\hat{x}}_j^*$, in the codebook. Assign $\underline{x}$ to the cluster around $\underline{\hat{x}}_j^*$.

Step 3. Replace each codeword with the centroid of the vectors in the training set that have been assigned to it.

Step 4. If the total error in clustering the training data is still decreasing by a significant amount, return to **Step 2.** Otherwise, stop.

This algorithm should optimise the codebook by minimising the sum of the distances from each vector of the training set to its nearest codeword. It is not possible, however, to guarantee that the algorithm will reach a globally optimum solution. This hypothesis can be confirmed by the fact that the final codebook will change if the initial codebook is different.

LBG design is critically dependent on the initial codebook supplied to it. The most common way to select an initial codebook is to extract vectors from the training set in a random fashion. Alternatively, other, simpler, clustering techniques can be used to generate an initial codebook. There is no simple way, however, to guarantee that LBG will not end up with a sub-optimal codebook.

An initial codebook may contain codewords which are not used at all during LBG training. Unused codewords can evidently be replaced with new codewords. The task of creating new codewords to fill these gaps in the codebook is one which is not covered in the VQ literature.

Another major problem occurs when a codebook is generated which codes certain parts of an image well, but not other parts. The most common cause of this is when a codebook contains many similar codewords which are each used infrequently. There are ways to prevent LBG design from reaching these local minima, but again these are not well documented in the literature.

The fact that these problems remained to be addressed led to the development of a novel codebook design method. A series of experiments was carried out to discover how the local minima could be avoided, and how optimal codebooks could be attained with the least effort. Section 3.5 details the development of this modified LBG design method, which incorporates intelligent methods to cope with the problems cited above.

3.4.2 Other Methods

Other methods for codebook design are aimed either at reducing the computational requirements relative to LBG design, or at improving the quality of the codebooks which are produced.

Equitz [106, 107] presents a fast algorithm based on nearest neighbour clustering. In this, the training set is gradually clustered by joining the closest remaining pair of clusters. This is repeated until the number of clusters remaining is the desired size of the codebook. Resulting codebooks are poor relative to LBG, but are generated in a fraction of the time.

Other methods recognise that LBG design does not always produce optimal codebooks and attempt to overcome the problems behind this. One method is the use of *simulated annealing* to modify the operation of LBG design. Here, the codewords are perturbed by adding a random noise at each iteration [108]. This is in an effort to move codewords to areas where they will code different vectors from the training set. The quoted improvement in performance was not significant.

Another alternative is to alter the design of the quantizer so that separate codebooks are used for different vector types. Standard codebook design techniques are used to generate each codebook. This is called Classified VQ and has been covered earlier (section 3.3.1) along with other alternative VQ codecs.

3.5 A Novel Codebook Generation Method

This section presents a novel codebook generation method which utilises intelligent techniques to develop a codebook containing a wide spread of codeword types. The resulting codebook enables images to be coded with a low error and also codes image detail well. In particular, the codebooks generated for images which contain large areas of near-constant brightness, such as those found in videotelephony applications, are significantly improved by the new technique.

A distillation of the experimental work which helped develop the method is presented in the following pages. This presents the ideas which have been followed and discusses the basis for these ideas. Experimental results are published here to indicate which of those ideas are of value to the task of codebook generation.

The study commenced with the implementation of the LBG algorithm, to get an idea of the problems involved in codebook generation and establish a baseline performance against which other methods could be compared. As was outlined in section 1.1, this simulation and those following it were performed by modelling the algorithm in ‘C’ and running simulations on Sun workstations.

Of importance to this work was the development of utility programs which allowed images and graphical representations of codebooks to be viewed on the monitors of these computers. This provided an excellent opportunity to quickly observe and evaluate the effects of algorithmic developments. Images and codebooks are shown in the thesis where they are useful to illustrate the performance of the algorithm as it was developed.

LBG Performance

The LBG algorithm, detailed in full in section 3.4.1, was modelled in software. Initial codebooks supplied to the LBG process were random samples from the training set, or training image.

Figure 3–8 shows an image from the *Claire* sequence before and after it has been coded. The image is of QCIF standard size (see section 2.4.2), meaning that it is 176×144 pixels in size, with each pixel resolved to 8 bits. It was coded with a codebook containing 256 vectors of size 3×3 pixels. A codebook of size 256 means that 8 bits are required to represent each coded vector. The vector size is 9 pixels, so the resulting compression ratio is 9 : 1. This provides a sufficiently difficult requirement to enable coding errors to be clearly seen on images of this size, when the coded image is viewed beside the original image.

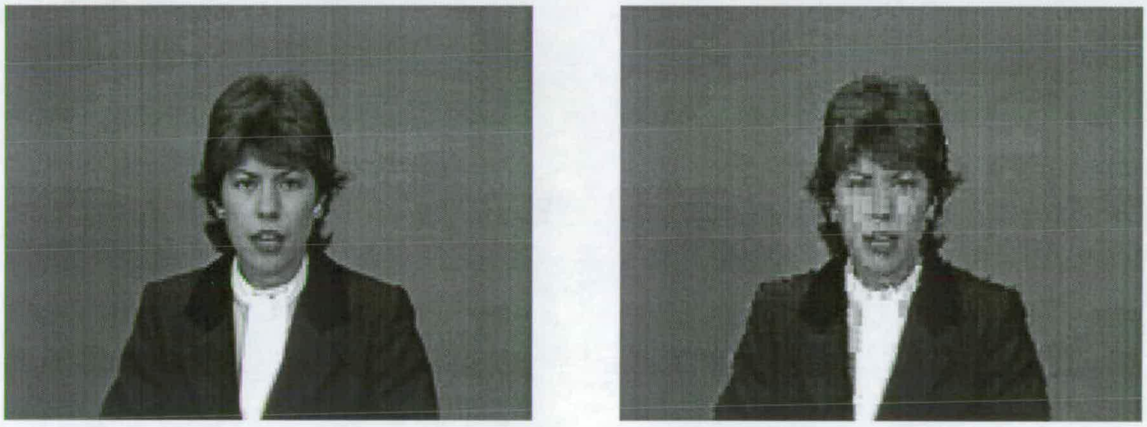


Figure 3–8: Original (left) and LBG coded (right) images

On inspection, the visible errors appear to be concentrated in the facial area and at high contrast edges, such as the shoulder-background edges. The errors in the facial area are particularly important because this is the area where a viewer is likely to be concentrating their attention. The background area appears to have been coded very well, although it is difficult to tell how large any coding errors are when the background intensity is so uniform and gives the human eye few cues to help guage these errors.

An alternative way to compare uncoded and coded images is to look at the *difference* or *error* image. This is formed by calculating the absolute pixel differences over the entire image. It is then necessary to multiply these differences by a constant to make them large enough to be seen when the difference image is viewed. The error image between the two images in figure 3–8 is shown in figure 3–9.

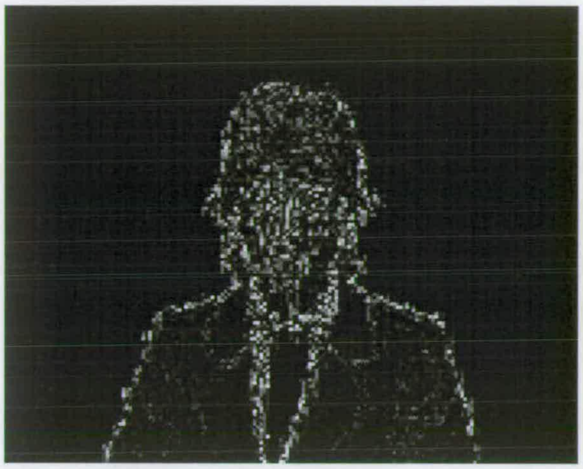


Figure 3–9: Error image for LBG coded Claire image. Errors have been multiplied by 8 to enable them to be seen well.

This investigation confirms that the largest errors are in the facial area and at high contrast edges. It is apparent that LBG codebook design has produced a codebook which is inefficient at coding the parts of this image which matter most to a human viewer.

If we take a look at the codebook which has been generated for this image, the most obvious point to note about it is the very large number of codewords which are very similar. Figure 3–10 illustrates this codebook. It is evident that a large proportion of the codebook has been designed specifically to code the background of the scene. It may well be that this results in the lowest possible error when coding the image. However, in perceived image quality terms, it is not important that this area is coded well. The areas of interest to a human viewer are those including the person.

This problem has arisen because of the inability of LBG to recognise that some codewords are more valuable than others. An initial codebook drawn randomly from this image is bound to contain a number of very similar codewords. We must find means of rejecting those codewords which are of poor value and replacing them with useful codewords.

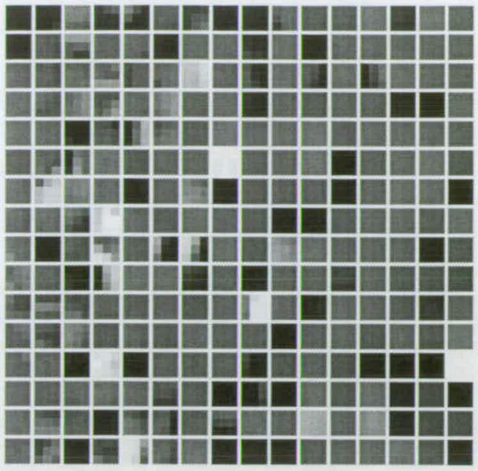


Figure 3–10: A codebook generated for the Claire image with the LBG algorithm.

A note on experimental procedures

The following pages present a few of the experiments which were undertaken in the development of the proposed algorithm. It is necessary to first introduce the reader to the format which has been chosen to quote results for these experiments.

All the algorithmic developments were tested on a wide range of types of image. For these experiments, results are quoted for four different images which were chosen to represent a good spread of image types. One image was drawn from each of the QCIF image sequences *Claire* and *Miss America*. The other two images used were the image of *Lenna* which is commonly seen in the image coding literature, and a picture of a residential street called *Mews*.

The two QCIF images used were at the standard QCIF resolution of $176 \times 144 \times 8$ bits. They were similar to each other in that they contained large background areas of near-constant illumination, which was seen previously to pose a problem for LBG codebook generation. The other two images used were resolved to $128 \times 120 \times 8$ bits. (They were generated by subsampling down from larger images and are hence referred to as 'Small Lenna' and 'Small Mews'.) These images were different from the other two in that they contained much more detail across the full image frame.

The coding methods were also tried out for three different codebook configurations.

Vector sizes of 2×2, 3×3 and 4×4 were used. The number of codewords in each codebook remained constant at 256.

This combination of images and codebook configurations was deemed sufficient to provide a suitable test for algorithmic developments. Results are given as PSNR measurements. (PSNR is defined as equation 2.4).

Splitting Methods

During LBG training the movement of codewords results in a very small number of them being unused at any training iteration. There are also a considerable number of codewords which are used only once, and it may profit us to discard these codewords and attempt to find valuable replacements to fill the now vacant slots in the codebook. The first experiment endeavours to find a useful method for re-populating the codebook.

The methods for codeword replacement which were tried vary widely in their complexity. Obviously, we are searching for a method which combines simplicity with good performance. A total of four methods are presented here, and these are outlined below.

Mode 1 : Replace discarded codeword with an average of itself and the most popular codeword. [1]

This method gives unused codewords a 'shake'. Hopefully, by shaking them towards the most popular codeword, they will become more useful.

Mode 2 : Rank the codebook according to popularity. Starting from the top of this list, split each codeword into two close copies of itself and replace any discarded codewords with one of the copies.

This is done by taking the codeword to be split and generating one copy which has pixel values slightly above the original (the 'big' copy), and one copy which has pixel values slightly below the original (the 'small' copy). The two copies are

[1] *popularity* is taken to mean the number of training vectors assigned to the cluster round that codeword during the current iteration of training. Thus the most popular codeword is that which has the most vectors from the training set assigned to its cluster.

placed in the slots of the discarded codeword and the codeword from which they were generated.

The splitting routine used to generate the two copies is as follows:

```
for each pixel in the vector
  {
    big copy pixel = 1.02 * original + 1
    small copy pixel = 0.98 * original - 1
  }
```

This technique is used also in the succeeding methods and is hereafter refered to as *splitting*.

The premise behind this method was an effort to find the largest clusters and to partition the training set more efficiently on the next iteration by offering a second codeword in that region.

Mode 3 : As with method 2, but rank the codewords on the sum of the distances to all the training vectors in their cluster.

This measure is not simply the popularity of the cluster, but a combination of its popularity and the volume (of Euclidian space) that it represents. Using this measure is an effort to find clusters which will more usefully be divided into two regions.

Mode 4 : As with method 2, but rank the codewords on

$$\sqrt{\text{popularity}} \times (\text{average Euclidian distance from codeword to cluster member})$$

Another measure used in an effort to find the most useful clusters to split.

Figure 3–11 presents the results for this experiment.

Very consistent results are shown for each mode across the full range of codebook and image combinations. Mode 1, though exceedingly simple to implement, offers little or no benefit. Mode 2 similarly offers little or no increase in PSNR. Modes 3 and 4, however, have improved from the LBG result in almost all examples.

Both are intelligent techniques which endeavour to allocate new codewords to areas of Euclidian space which will best profit from an extra codeword. This was considered

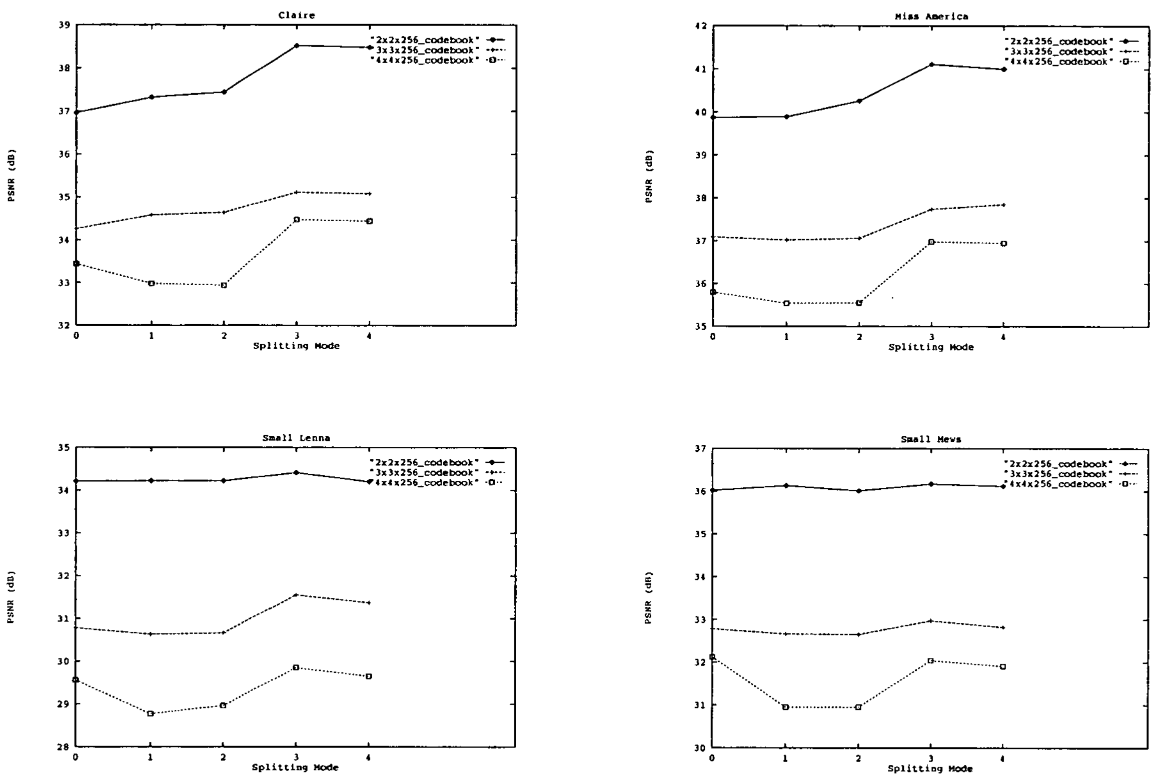


Figure 3–11: Results for the four different splitting modes. These results are compared against the results for standard LBG (shown as mode 0). PSNR results are shown for three codebook sizes and four separate images.

to be a search for clusters which were large in Euclidian volume and contained a large number of training vectors. These clusters, which are allocated a second codeword, are those which have contributed the largest part to the total coding error.

Mode 3, which uses the cluster distortion as a measure for finding useful clusters, performs better than mode 4 in 11 out of 12 examples. Mode 4 was proposed because it was thought that mode 3 was weighted too heavily towards codewords which were simply popular, rather than the centre of poorly coded clusters. However, this was not found to be the case.

Mode 3 has a surprisingly simple implementation. It requires that the cluster distortion is held for each codeword. This is simply the summation of the errors in coding each of the training vectors in that cluster, which can be accumulated as the training is performed. At the end of each iteration of the training, the clusters must be ranked according to this value.

In this software simultation, the ranking is done at the completion of each training iteration. However, this process could be done in parallel with the training, with adjustment of a league table after each training vector assignment. This would result in a smaller lag-time at the end of the training vector assignment at each iteration.

Images coded using mode 3 do show a perceptible improvement in quality. However, it is clear from observations of the resulting codebooks that the problem of large numbers of similar codewords remains. It was clear that other techniques for populating the codebook with a wider variety of codewords were required.

Codeword Redundancy

We wish to find some means, preferably computationally simple, which allows us to identify codewords which are similar to others in the codebook and could be usefully discarded and replaced with new, more valuable, codewords. This is a search, in other words, for *redundant* codewords. The training vectors assigned to a redundant codeword would be well coded by one of the existing codewords at the next iteration. The codeword which has been generated to fill the gap vacated by this rejection will hopefully enable the codebook to more effectively code the image.

One of the main aims was to to find a simple measure to use as a criterion for deciding that a codeword was redundant and should be discarded. It was realised from investigation of the results from previous experiments that many codewords were being used only a few times at each training iteration, and that many of these were very similar to each other and could perhaps be rejected. It was decided to use codeword popularity as a redundancy criterion. This would have the advantage of being a very simple measure to use.

The codeword popularity below which codewords would be rejected was varied. The algorithm for splitting which was presented in the previous section as mode 3 was retained. The same image/codebook combinations as used previously were coded across a range of values. Figure 3–12 shows the results for this experiment.

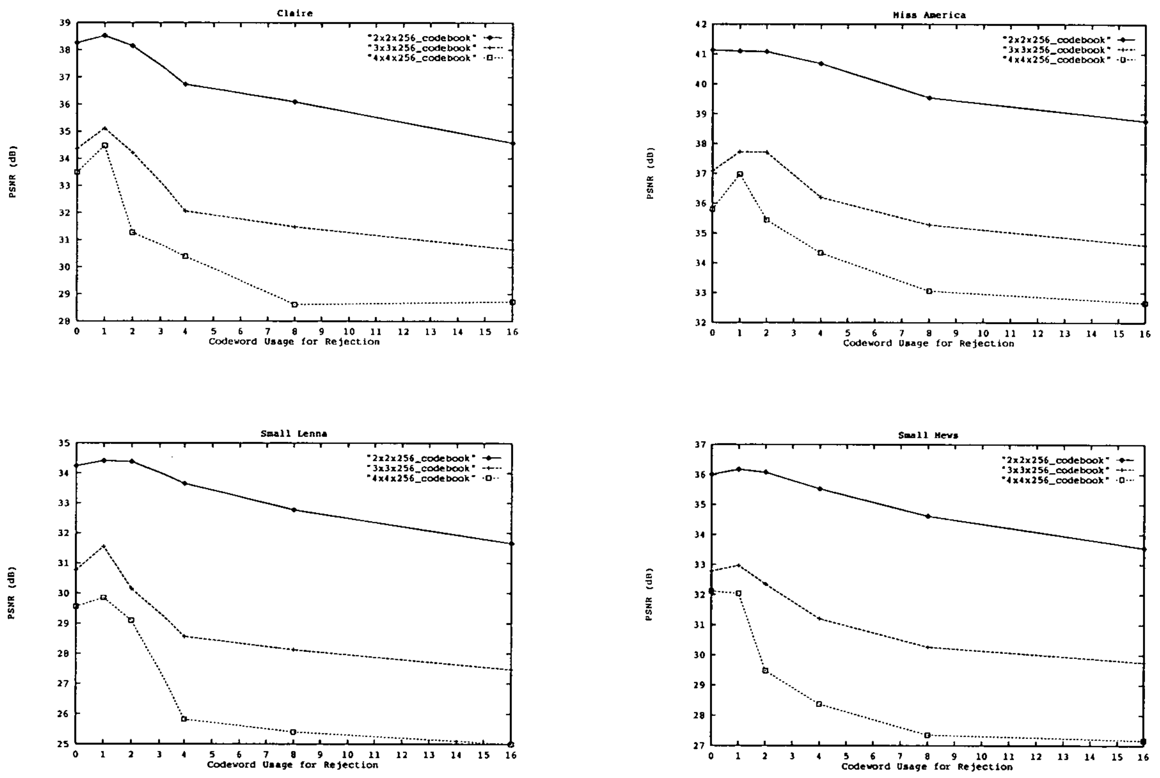


Figure 3–12: Results for codeword rejection on popularity

The results show that rejecting codewords which have been used only once gives the best PSNR values. In 10 out of 12 examples, one is the optimum popularity below which to reject. In the other two examples, the best result was given when codewords were rejected only when completely unused. In all cases, rejecting codewords which were used 2 or more times caused a marked degradation in performance. Clearly, rejecting codewords on this simple measure is not a good idea. It is obviously the case that some useful codewords are being rejected, as well as some redundant codewords.

A more rigorous approach to defining codeword redundancy is obviously necessary. One way to do this is to explicitly search the codebook for codeword pairs which are similar. It will then be necessary to decide which one of this pair should be discarded.

In the previous experiment the codebook was ranked on total cluster distortion. This was seen to place codewords in order of how valuable they were to split. Thus it can also be seen to be ranking the codewords in reverse order of how valuable it would be to reject them. This ranking scheme was retained for the next experiment. Here, the ranked list is searched from the bottom up for codewords which have very similar copies above them. If a close match is found, the codeword lower in the list is considered as redundant, and is replaced with a split codeword from the top of the list, as before.

The measure used for a test of similarity of codewords is whether the mean-squared-error between them is greater than some fraction of the average mean-squared-error of coding the entire training set. This was more effective than an absolute difference measure which would not take into account the dynamic range of the image (and hence the dynamic range of the codebook).

In the following experiment, the 12 image/codebook combinations were again coded, this time over a range of values controlling the similarity criterion. Figure 3–13 shows the results for this experiment.

The results indicate that this enhancement has been very beneficial. The *Claire* and *Miss America* images can be coded much better when codebooks are generated using this method. Optimum improvement for these images is when the fraction of average mean-squared-error is set to 0.1 or above. This offers an improvement in PSNR of more than 1dB for codeword sizes of 2×2 and 3×3 pixels, and a lesser improvement for codeword size of 4×4 pixels. The improvement in perceived image quality is considerable. Figure 3–14 shows the *Claire* image coded without redundant codeword rejection, and then with this addition. The facial areas and high-contrast edges have been coded much better.

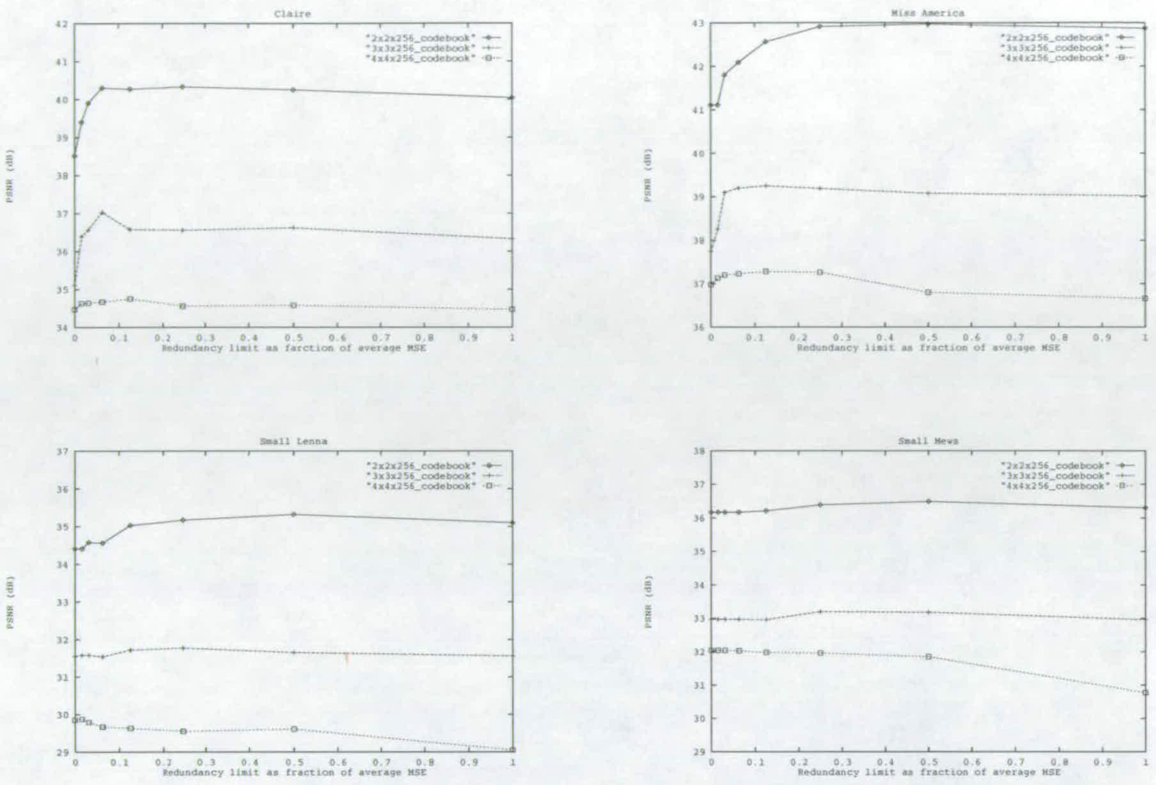


Figure 3–13: Results for codeword rejection on redundancy

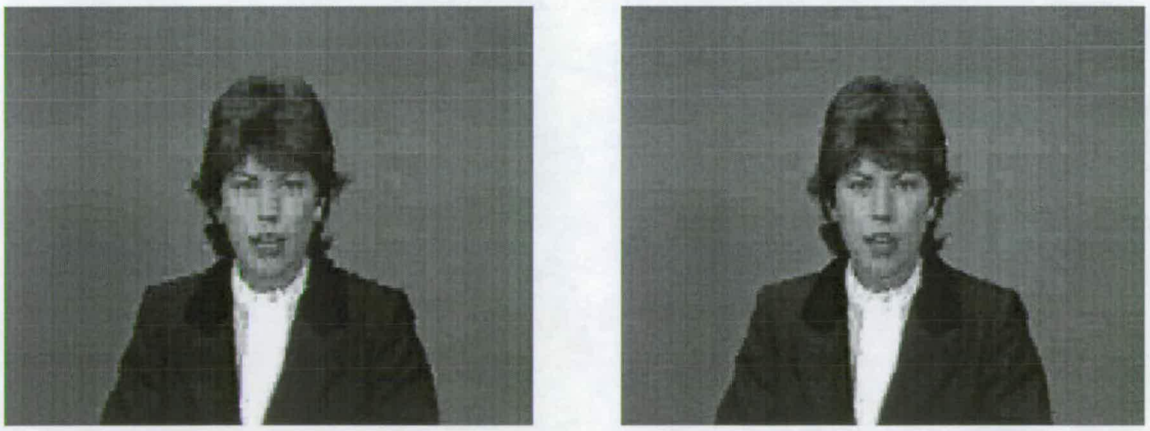


Figure 3–14: *Claire* image coded without (left) and with (right) redundant codeword rejection.

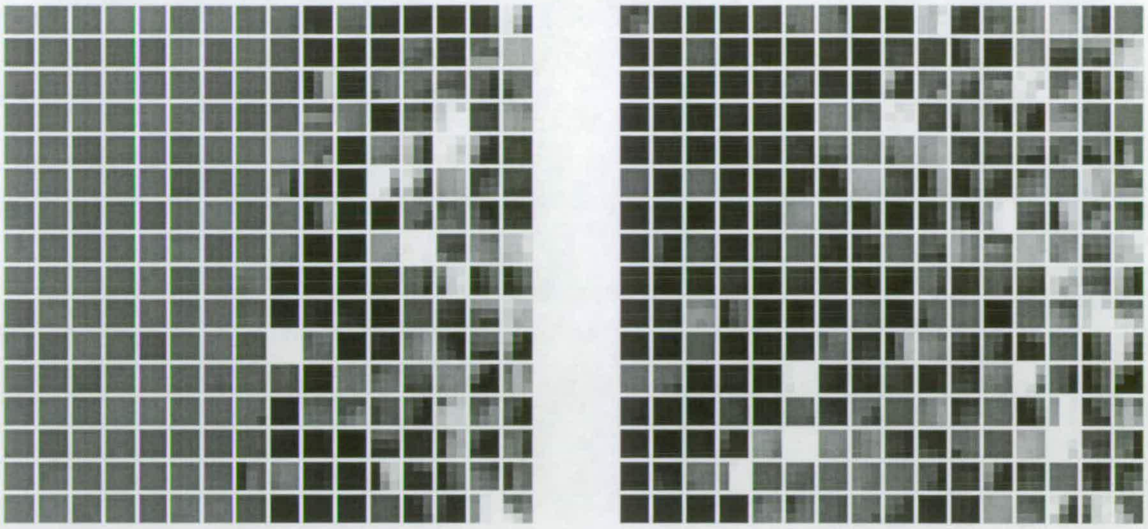


Figure 3–15: Codebooks for the *Claire* image generated without (left) and with (right) redundant codeword rejection.

A look at the codebooks for these two images explains why this improvement has occurred (see figure 3–15). Without redundant codeword rejection, the codebook contains a large number of light grey codewords which are used to code the background. With redundant codeword rejection, many of these have been replaced with other, more valuable codewords. This has not, however, resulted in large coding errors in the image background, as illustrated by the error image (figure 3–16). This also shows how the errors have been reduced in comparison with the LBG coded image (compare with figure 3–9).

Coding errors for the Small Lenna and Small Mews pictures are almost unchanged. Coded images show no perceptible change with the addition of redundant codeword rejection.

This work was developed to be of benefit when generating codebooks for images with large background areas of near-constant illumination. These are the type of images commonly expected in videotelephony. This improvement has been gained without causing a drop in performance for coding of high dynamic range images.

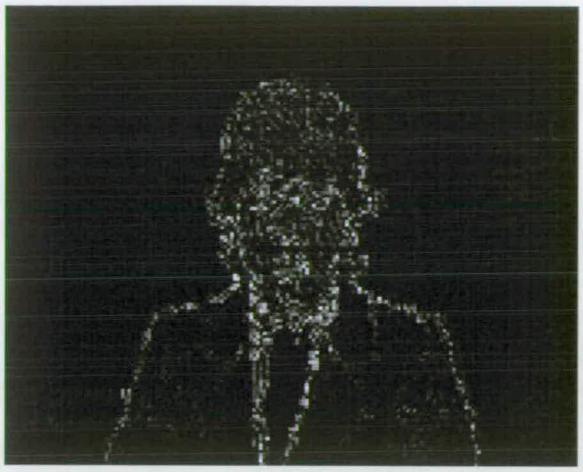


Figure 3–16: Error image for Claire image coded with redundant codeword rejection. Errors have been multiplied by 8 to enable them to be seen well.

Summary

The preceeding section has presented the work which led to the development of an effective codebook generation technique which is essentially a modified LBG approach. The new algorithm incorporates two techniques which enable a codebook to be effectively populated with a wide range of vector types for optimal coding performance. These enhancements ensure that a good codebook is generated even when the initial codebook supplied to the codebook generation process is poor.

The two intelligent techniques formulated here are an intelligent splitting technique for codeword replacement, and an intelligent method for rejecting redundant codewords.

The final modified LBG algorithm is outlined in figure 3–17.

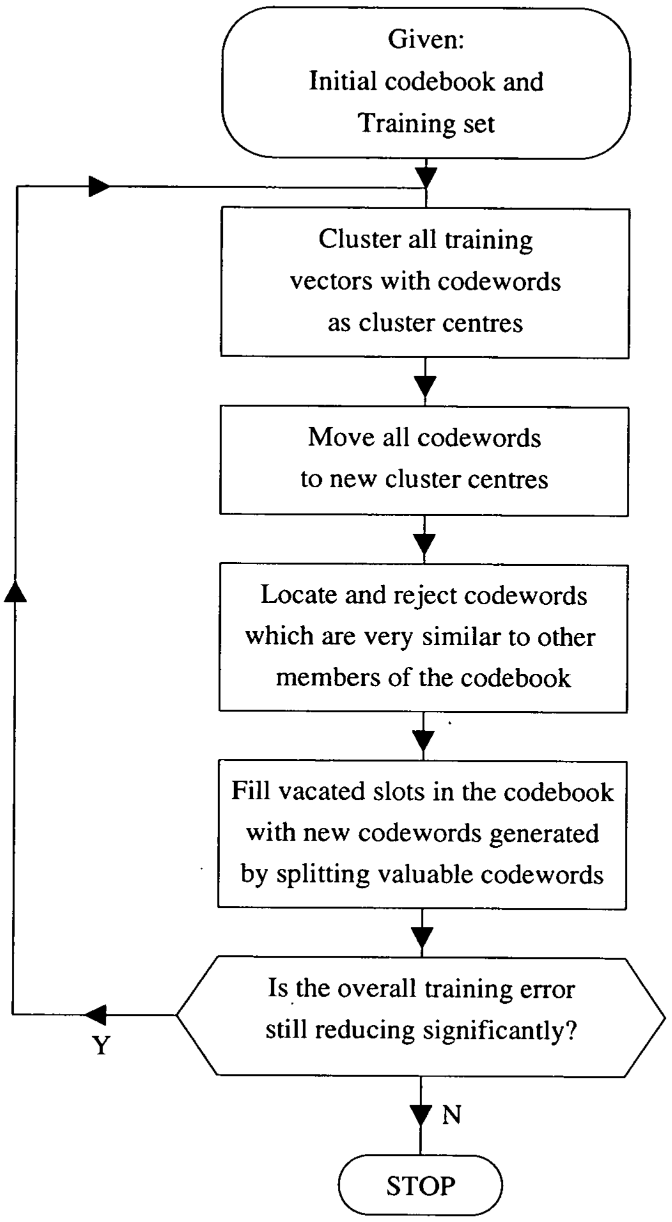

Given:
Initial codebook and
Training set
Cluster all training
vectors with codewords
as cluster centres
Move all codewords
to new cluster centres
Locate and reject codewords
which are very similar to other
members of the codebook
Fill vacated slots in the codebook
with new codewords generated
by splitting valuable codewords
Is the overall training error
still reducing significantly?
Y
N
STOP

Figure 3–17: Novel codebook generation algorithm.

3.6 Implementation Considerations: Simple VQ

This section takes a brief look at the trade-offs that need to be considered when designing for a VQ application. For a simple VQ codec, as introduced previously in section 3.2, it is possible to vary the codebook size, the codeword size and the codeword shape.

For variations in codebook size and codeword size we will calculate the effect on computational and memory requirements. This will be followed by a set of experiments which assess the performance of VQ coding as all the variables are changed. Here, distortion-rate curves will be used to illustrate how error and compression performance relate to each other as the variables are changed.

Codebook and Codeword Size

For a codebook containing N vectors, each $x \times y$ pixels in size, we wish to determine the number of calculations required to code an image of size $X \times Y$ pixels. The number of vectors in this image (assuming for simplicity that there are a whole number of vectors in the x- and y-dimensions of the image) is given by

$$\frac{XY}{xy} \tag{3.13}$$

To calculate a Euclidian distance, which is the usual distortion measure used, there is one subtraction, one multiply and one accumulate per pixel. To search the entire codebook to find the nearest neighbour to a single vector then needs xyN subtractions/multiplies/accumulates and N comparisons. Thus, coding the whole image takes

$$\begin{aligned} \frac{XY}{xy}xyN &= XYN \text{ subtractions/multiplies/accumulates} \\ &\quad + \frac{XY}{xy}N \text{ comparisons} \end{aligned} \tag{3.14}$$

The first term in this equation is the dominant one, and if the second term were to be ignored the computational cost would be proportional to XYN which is directly proportional to the number of codewords in the codebook.

For small codeword sizes, the second term would become significant and the computational burden would increase as the codeword size decreased. In a hardware implementation, however, these comparison operations could be done in parallel with the distance measures and the distance measures would again be the limiting factor.

The memory requirements of a simple VQ depend entirely on the amount of data in the codebook. Thus the memory requirements grow proportionally with increases in codeword dimension and codebook size.

Distortion-Rate Curves for Simple VQ

Here, the *Claire* image has been coded across a range of codebook and codeword sizes. Codebooks were generated using the modified LBG algorithm described in section 3.5.

The results are given as distortion-rate curves. Here, distortion, quantified as PSNR, is plotted against the bit rate of the compressed data, measured in bits per pixel (bpp). The bit rate is calculated from

$$\text{bit rate} = \frac{log_2 N}{8xy} \quad \text{bpp} \tag{3.15}$$

Figure 3–18 shows the distortion-rate curves for the image *Claire*. This image was coded for three different vector sizes (2×2, 3×3 and 4×4 pixels) and 4 different codebook sizes (64, 128, 256 and 512 codewords).

A few conclusions can be drawn from these curves. The results for other images are very similar and support the conclusions drawn here.

The most obvious is that an improvement in compression rate for a given performance would be best achieved by increasing the size of the codewords. For a specific PSNR of 34dB, the bit rate for a 3×3 codeword is approximately 0.9bpp, while for a 4×4 codeword the bit rate is approximately 0.55bpp.

Decreasing codebook size also achieves a rate improvement but this is accompanied by a steep degradation in performance. The degradation is even more marked for larger codeword sizes.

The other major point to note from these results is that the task of codebook generation becomes more difficult as the codebook and codeword size increase together. For

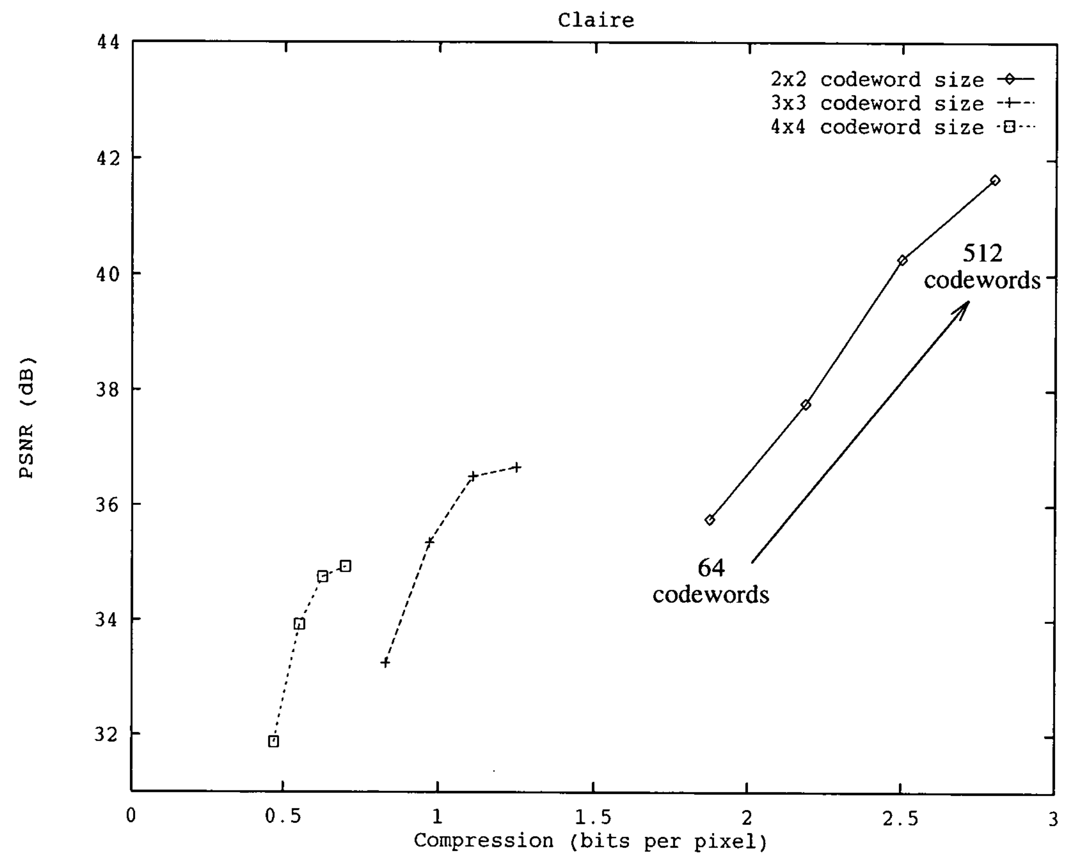


Figure 3–18: Distortion-rate curves for a variety of codebook configurations. The image *Claire* was coded with three vector sizes (the three curves) and four codebook sizes (64, 128, 256 and 512 codewords). The direction of increasing codebook size is indicated for the codebook containing vectors of 2×2 pixels.

codewords of size 3×3 and especially 4×4 there is a noticeable flattening off of the noise improvement gained by increasing the codebook size. The ultimate bound on the performance of VQ is thus the difficulty of generating optimal codebooks when these codebooks are large.

In summary, it can be said that the most important contributor to a rate improvement is an increase in the size of the codewords. This increases the memory requirements of the quantizer, and contributes a slight reduction in the quantization complexity. However, the task of codebook generation becomes more difficult as codeword size grows.

Codeword Shape

The optimal codeword shape is different for any given image. It can be seen to be a function of the orientation of the high-contrast edges which exist in the image. An image containing a lot of vertical high contrast edges will be coded much better with vectors which are taller than they are wide. Similarly, vectors which have more pixels in the horizontal direction than the vertical will be the best type for coding images with a preponderance on high-contrast edges in the horizontal direction. It follows that, if there is no *a priori* knowledge of the image type, square codewords are the best compromise. If the application is very restricted it may be the case that assymetrically shaped codewords are the best choice.

Rectangular codewords cause 'blockyness' errors which are particularly displeasing to the human eye. This problem is also associated with other coding techniques, such as transform coding, which are based on the processing of rectangular blocks of the image. In an effort to circumvent this problem, Pearson and Whybray [109] proposed using a tiled pattern of interleaved, irregularly shaped blocks. This did improve perceived image quality but at a cost of introducing quite complex addressing requirements.

3.7 Conclusions

This chapter was designed as an in-depth introduction to VQ. It aimed to introduce the reader to the potential which this method has to offer, and also to identify the problems which are associated with it.

The chapter began with a formal mathematical definition of VQ as it was presented by Linde, Buzo and Gray, who are today identified as the authors of the seminal publication in this field. This was followed by a survey of the many VQ codecs which have been developed for the coding of still images. These codecs aim to improve over simple VQ by improving the coding capability or reducing the cost of implementation.

Methods for codebook design were then introduced in section 3.4. The most common method in use today is that proposed by Linde, Buzo and Gray which is known as the LBG algorithm. Codebook design is done by optimising the vector quantizer for the coding of a training image through an iterative process. Other methods have been proposed to either reduce the computation required for this process, or to generate improved results. No methods exist which are shown to significantly improve upon the LBG algorithm.

It was recognised through preliminary experimentation that, for certain image types in particular, there are problems with the LBG method. The problems arise specifically when the training image contains large areas of very similar information, such as those found in videotelephony applications. The result of this situation is the generation of codebooks which are poor at coding important areas of the image, such as the face, which contain most of the detail in the image. The requirement to develop an improved codebook generation technique was clear.

Section 3.5 illustrated the development process that led to the improved codebook generation method with experimental results and discussion. The technique developed here is a modified version of the LBG method. The modifications are designed to identify redundant codewords, which are then deleted, and to fill the gaps created by this process with useful replacements. Redundant codewords were found to be those which were infrequently used and very similar to other members of the codebook. Suitable replacements were created by splitting codewords with large training clusters into two close copies of the original. This algorithm retains an essentially simple structure.

Significant performance improvements were seen to result from this algorithm. The images *Claire* and *Miss America* were seen in particular to benefit. Codebooks for these images contained a much wider variety of codewords. Qualitative assessment of coded images showed that this had given a great improvement over the standard LBG method, with facial detail in particular being coded considerably better.

This chapter concluded with a closer look at the trade-offs involved with a simple VQ implementation. This analysis showed that the most valuable contribution to an improvement in compression ratio is gained by increasing the dimension of the codewords. This increases the memory requirements of the quantizer and slightly reduces the computional complexity. As well as the increase in memory requirements, the limiting factor is the difficulty of generating codebooks as the codeword dimension grows.

Chapter 4

Interframe Vector Quantization

4.1 Introduction

In an image sequence, there will be a very high correlation between successive frames. We can take advantage of this fact by transmitting only those sections of the image which have changed. This is known as frame replenishment. Using VQ for interframe coding leads intuitively to the development of label replenishment schemes, where the changes in the image are identified by a change in the codeword used to code that vector of the image.

Codebooks are very specific to the image which has been used to train them. We will see in this chapter that, even for a simple videotelephony sequence, where there is no change to the position of the camera or the person in the frame, a codebook generated for the first image of the sequence is not efficient at coding the later images. This result leads to the development of techniques which allow codebook updates as well as label updates.

Section 4.2 introduces interframe VQ techniques in more detail, and reviews those schemes which have been presented in the literature. The remainder of this chapter concentrates on the experimental work done by the author in the analysis of the image sequence coding problem, and the development of new algorithms for this task. As a prelude to this work, the experimental procedures and the form of presented results are introduced in 4.3.

Two novel label replenishment schemes are presented by the author in section 4.4. These schemes both first identify those areas of the image which have changed the most, then code these areas as update information. This results in a reduction in processing. The work in this section provides an introduction to the image sequence coding problems, and illustrates the limitations of label replenishment methods.

Section 4.5 presents some investigations into methods which could be used to initiate a videotelephony link. The approach here is to consider how a real videotelephony link would build an image at the receiver in the shortest possible time. This work builds on the experimental work of section 4.4. Section 4.6 summarises the results from the previous two sections.

With the need for codebook replenishment clearly identified, this chapter continues in section 4.7 with the development and assessment of techniques for this purpose. The aim throughout this work is the identification of methods which are as simple as possible, and the generation of codebook updates which provide the most coding improvement for the least update information. The methods developed here build on those developed for the improved LBG codebook generator of section 3.5.

4.2 Interframe VQ Techniques

In this section a general introduction to interframe VQ coding methods is given. The techniques presented in the literature are summarised here also.

4.2.1 Label Replenishment Only

The simpler forms of interframe VQ update only the codeword labels which code each successive frame. This general technique is outlined by Nasrabadi and King [70] and shown in figure 4–1.

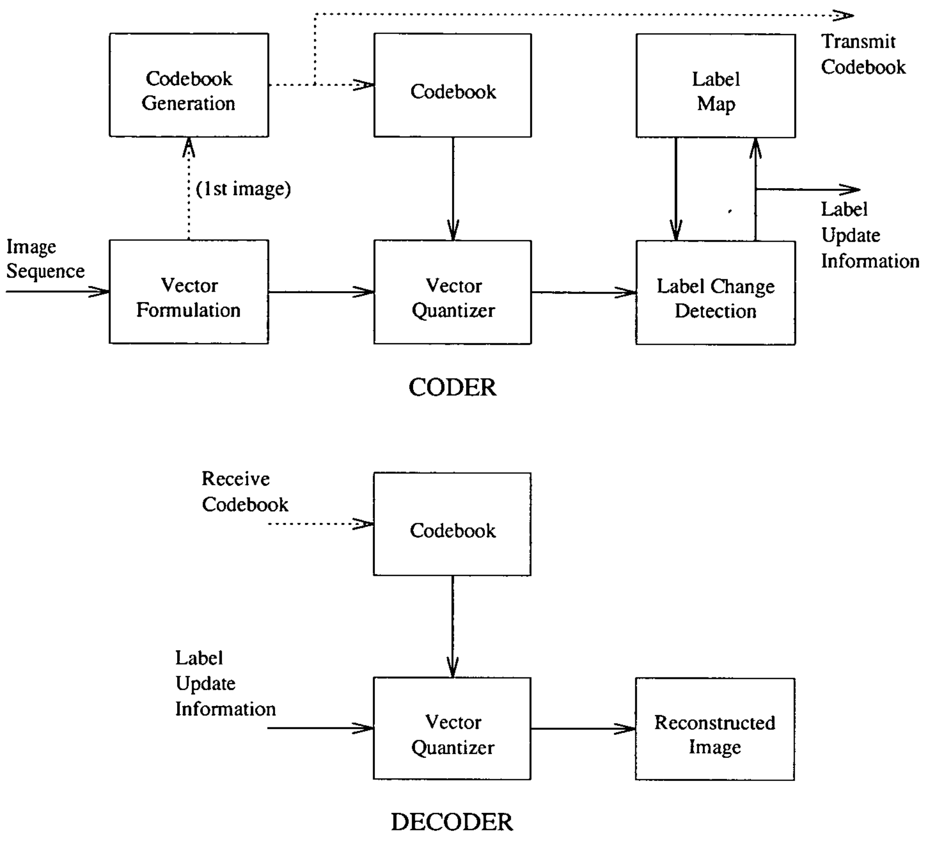


Figure 4–1: Label Replenishment Interframe VQ

Here, a codebook is generated for the first image of a sequence, and this image is then coded with this codebook, which creates a *label map* - an array specifying which codeword has been used to code each vector of the image. The codebook and label map must be transmitted to the receiver. Subsequent frames are coded with the same codebook. Areas of the image which have changed will be indicated by changes in the label map, and those areas which do not change will be coded by the same codeword, and the label map will not change. Label replenishment indicates that those labels which change will be transmitted to the receiver, and the reconstructed image will thus be updated in the areas which have changed.

In practice, a large percentage of the label map will change at each successive frame, although much of the image may change only slightly, due to a drift in the illumination level, for example. Thus, any label replenishment scheme is more complex than the simple description given above. The performance of label replenishment techniques is investigated in section 4.4.

Any sequence of images will, however, be poorly coded by a codebook trained on only one image of that sequence. Even if the camera position is fixed, the image statistics will be likely to change considerably over time due to changes in illumination, orientation of subject, distance to subject, and many other reasons. From this, the need for an adaptive codebook is clear. Section 4.7 introduces codebook replenishment.

There are adaptive interframe VQ methods which do not use codebook replenishment. Dixit and Feng [95] present one such algorithm. Here, one large codebook is split into two : one small codebook containing frequently used codewords, and the other, much larger, codebook containing the remaining, infrequently used codewords. Periodically, the codebooks are rearranged to reflect the recent usage patterns. This carries no bit rate penalty as both coder and decoder know the usage patterns.

4.2.2 Label Replenishment and Codebook Replenishment

Figure 4–2 presents the general form of a codebook replenishment interframe VQ.

In general, a codebook replenishment scheme transmits a number of new codewords for each frame of the coded sequence, as well as the label change information. These new codewords are intended to improve the performance of the VQ at each frame, relative to the coding performance which would be attained with an unchanged codebook. In this way, the performance of a VQ with codebook replenishment should not degrade over a sequence, but should be able to code successive images with a similar PSNR to the first image of the sequence.

Codebook replenishment obviously adds a data overhead and increases the bit rate. If a large number of codewords are to be updated each frame, the amount of data this adds can easily become comparable to the amount of data in the label change information. Changing only a few codewords per frame is desirable, and this requirement drives the research presented in section 4.7.

Many of the codebook replenishment schemes presented in the literature are largely similar in that they employ the same three basic operations. These three operations can be called *shift*, *split*, and *delete*. They relate to the operations used in the modified

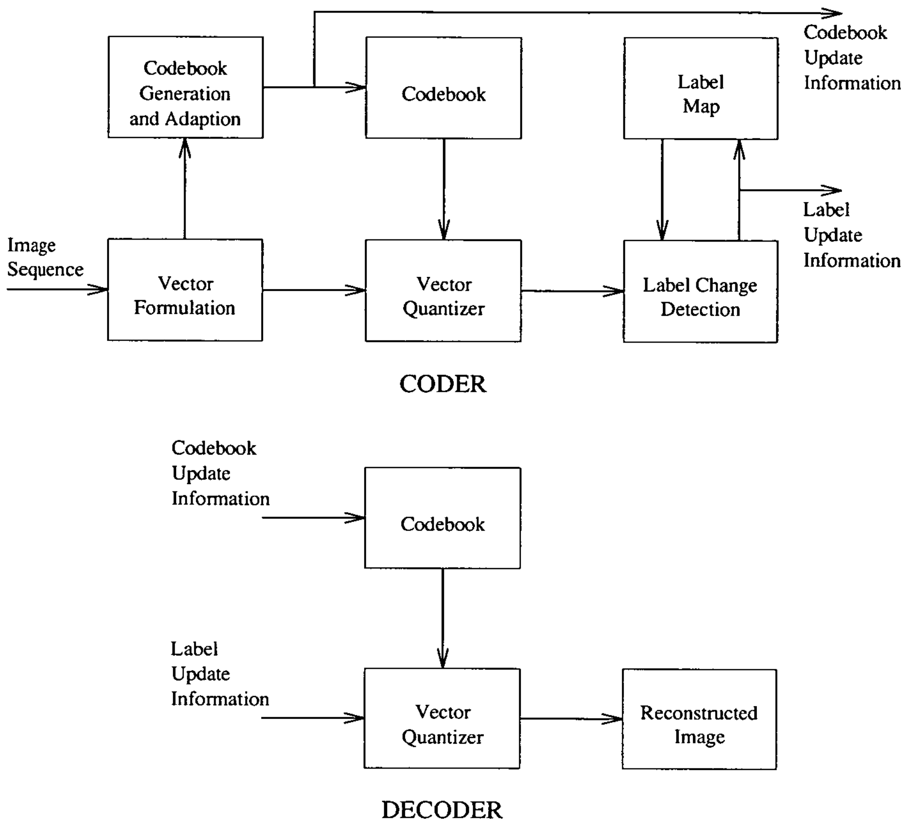


Figure 4–2: Interframe VQ with Label Replenishment and Codeword Replenishment

LBG approach presented in section 3.5. The major differences between the adaption techniques are in the criteria used to decide when to perform the *shift*, *split* or *delete* operations.

A *shift* operation refers to moving a codeword to the centre of its cluster (the group of vectors which have been coded with this codeword). A *split* operation takes one codeword and creates two different, close copies of the original. This is usually done by adding a small random value to each of the pixels in both of the vectors. A *delete* operation removes a redundant codeword from the codebook.

Sun and Goldberg [110-113] present a technique which they call Frame Adaptive Migrating Mean (FAMM). In FAMM, all codewords are moved to their centroid at the end of each frame, and those which move further than a preset threshold are updated at the coder and decoder. There is also provision in their algorithm for a *split* operation to be performed on one of the *shifted* codewords, whereby the new codeword created is used to replace a *deleted* codeword. They do not state what rules are used to decide which codewords are to be *deleted.*

Yeh [76] applies a codebook replenishment algorithm to coding a sequence of colour images using Mean-Residual VQ with a binary tree structured codebook. Progressive adaption is attained with only the *shift* operation. Each codeword is centred and the total distortion of the cluster to this new centroid is calculated. A *shift* operation is applied to those codewords whose new centroid gives the greatest reduction in total distortion. Yeh recognises that this technique is unable to maintain a good codebook over a long sequence, and occasionally creates an entirely new codebook. This is a highly compute-intensive technique.

Monet and Labit [114] present another highly complex algorithm, designed for coding of broadcast quality television pictures over a very high bandwidth (34Mbits/s) channel. They use a Classified VQ (see section 3.3.1) with a tree structured codebook for each class of vector. Codebook replenishment uses all three operations and allows for restructuring of the codebook.

Gersho and Yano [115] have developed a method which is used to code single images with a codebook replenishment algorithm. Once again, the three operations of *shift*, *split* and *delete* are used. Panchanathan and Goldberg also introduce a codebook replenishment scheme which is used to code only single images. They identify the problem that the above techniques cannot guarantee that all vectors will be coded with an error which is less than a pre-specified limit. Their algorithm allows for poorly coded vectors to be simply added to the codebook. They also employ a two-tier codebook structure based on usage, similar to that used by Dixit and Feng [95] (see section 4.4).

In all of the above research studies, there is a failure to assess the codebook updates for the benefits that they bring to the sequence coding problem. At best, the codebook updates will probably be inefficient, as many of the schemes update large portions of the codebook at each frame. At worst, the codebook adaption techniques may go no distance to improving the quality of the codebook. Certainly, little effort has been made to do all of this using simple techniques which could yield realisable solutions. In section 4.7 all these factors are taken into consideration.

4.3 Experimental Procedures and Presentation of Results

When studying interframe coding the commonest test of a codec is to use a heavily constrained sequence to simplify the problem. The *Claire* and *Miss America* sequences are such : they are head-and-shoulders shots with a fixed camera position, uniform background and near constant illumination levels over time. It is not the intention of this chapter as a whole to consider only this constrained problem.

However, it is recognised that label replenishment schemes are limited in the performance they can offer, and for the experiments on label replenishment schemes (section 4.4) only these two sequences will be considered. It is possible to draw conclusions from the work on these sequences, without attempting to solve a tougher problem.

A more realistic test of an image sequence codec is to remove these constraints on the test data. In particular, the response of the system to a complete scene change is of interest, as this is possibly the most difficult test of any codec. This situation is considered in section 4.7 to test the performance of the codebook replenishment scheme.

In most cases results will be presented as peak signal to noise ratio (PSNR) plotted against frame number in the sequence. This type of graph will be referred to as a sequential frame error graph. This is the most effective way of demonstrating how well a sequence has been coded as it shows the coding error at each frame, rather than averaging the errors across a sequence. It also shows trends in the coding errors, which are important results from which we can draw conclusions.

As variable update rates are considered here, it will also be necessary to show results in a way which allows the error performance to be evaluated against information rate. Here bit-rate (in bits per pixel, *bpp*) will be plotted against average PSNR. PSNR will be averaged over the full sequence, or, as is often more appropriate, over a part of the image sequence.

4.4 Label Replenishment Interframe VQ

This section presents some of the experimental work which was done to investigate label replenishment methods of interframe VQ coding. Label replenishment was studied because it provided an introduction to interframe coding and it may be able to form the basis of a low bit-rate, low complexity videotelephony system. The results from this work also give a datum from which to evaluate the codebook replenishment schemes developed in section 4.7.

The general structure of label replenishment interframe VQ seen in figure 4–1 suggests that all labels which have changed between frames should be updated in the label map and sent to the receiver for updating the reconstructed image. In practice, however, not all changed labels can be updated, as this is too large a fraction of the total number of vectors in an image. Initial experiments were designed to investigate the proportion of labels which need to be updated per frame to maintain a reasonable image.

The concept of updating only a fraction of an image at each frame stems, obviously, from the high correlation between frames in a sequence. There may be a benefit in coding the difference image between two frames, rather than each image itself, before label replenishment is done. Interframe Error VQ (IEVQ) is presented in section 4.4.2 to test this hypothesis. It was also hoped that this technique may be less vulnerable to scene changes than the standard label replenishment methods. Error images do not contain any intensity information which was seen in chapter 3 to heavily influence the spread of vectors in a codebook. Codebooks designed to code error images may be efficient at coding error images from a broad spectrum of image sequences. This hypothesis is also tested.

4.4.1 Interframe VQ with Conditional Label Replenishment

This experiment aimed to investigate how the performance of label replenishment is affected by the number of labels which are updated at each frame. We will try to estimate the amount of update information required to maintain a reasonable quality image for a typical videotelephony sequence. We will also look at the degradation in performance as the update rate is lowered. Finally, and crucially, the degradation of label replenishment over time will be studied.

The simplest implementation of a label replenishment codec would send every label which had changed from one frame to the next. This poses at least two major problems which are : the number of labels which change at each new frame is a large fraction of the total number in the frame, and this number is variable.

Initial experiments were carried out to estimate the number of labels which change per frame, and this number was found to be approximately 90%. This proportion is obviously prohibitively high to consider using label change as the criterion for sending the new label as update information. Also, it was clear that there was little value in sending many of the label updates, as the actual image vector had changed very little, but had been coded by a slightly different codeword. This was true of many of the codewords in the background of the scene where the image vectors changed very little, but there was sufficient change due to small fluctuations in illumination levels to result in the label changing between frames.

One must also realise that it is necessary to send the position in the image of each label update. There are a number of ways of doing this and some are discussed later in Appendix A. This information adds an overhead that is quite a considerable contribution to the information rate when small label update rates are being used. For the specific image and codebook sizes chosen in the example analysis of Appendix A, the addressing information exceeds or matches the label update information when less than approximately 15% of the total label map is updated.

It is also desirable for many applications that the data rate for a video codec is constant. This enables the maximum capacity of a communications channel to be used all the time. Some communications networks, such as Broadband-ISDN (see section 2.4.1), will offer variable bit-rate data channels in the future, but many applications will use technology which imposes the constraint of constant channel bandwidths.

The Algorithm

For the reasons outlined above, the system shown in figure 4–3 was used to test label replenishment methods. Here, the first image in the sequence is used as the training data to create a codebook. The codebook generation method used is that presented in section 3.5 which was shown to produce excellent codebooks. This codebook is sent to the receiver. The first image is then coded with this codebook and the entire label map is also sent to the receiver.

The task of codebook generation obviously takes some time, and there is also the fact that the codebook must be transmitted to the receiver. However, the process of initiating a transmission is not one we wish to study here. It will be studied later in section 4.5.

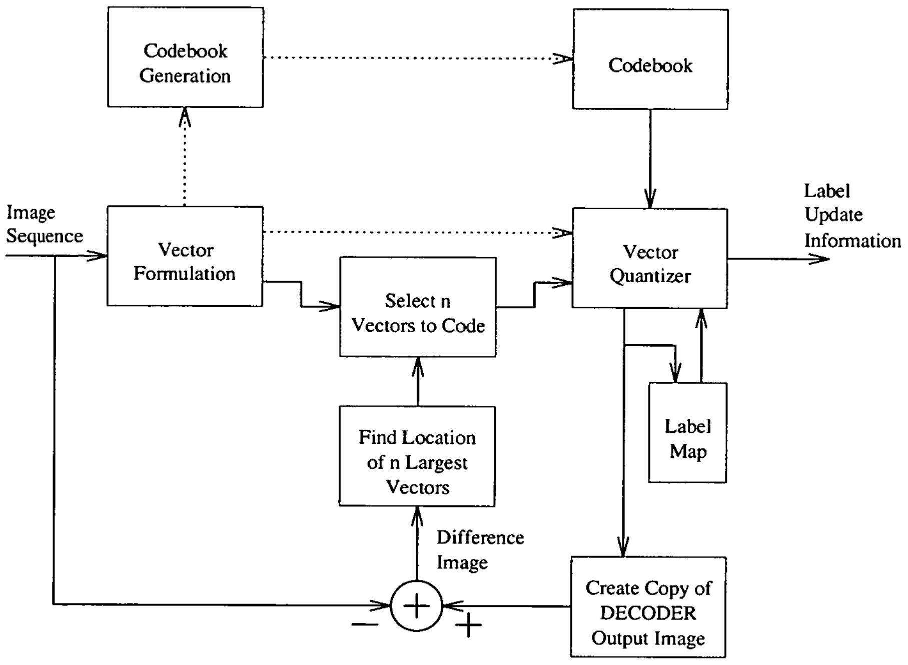


Figure 4–3: An Original Conditional Label Replenishment Scheme (CODER only shown). Dotted lines show the actions performed on the first image in the sequence.

The receiver is now able to reconstruct the first image with the codebook and label information it has received. This coded image is also created at the transmitter. The second frame in the sequence is subtracted from this reconstructed coded image to generate a difference image. The difference image is subdivided into its vectors and the

Euclidian size of each of these vectors is calculated. From this information the largest n vectors are identified, where n is some fixed number smaller than N, which is the total number of vectors in the image.

This process compares the coded image and the second image in the sequence and identifies the n vectors where the largest changes have occurred. These n vectors are then picked out of the current input image and are coded. The n label updates are sent to the receiver, and are used at both the transmitter and receiver to update the coded image. Subsequent images are then coded in exactly the same way as the second image. This method reduces the processing done by the vector quantizer by a factor of $\frac{n}{N}$. Against this, it adds the computation required to form the difference image and to identify the n largest vectors in this image.

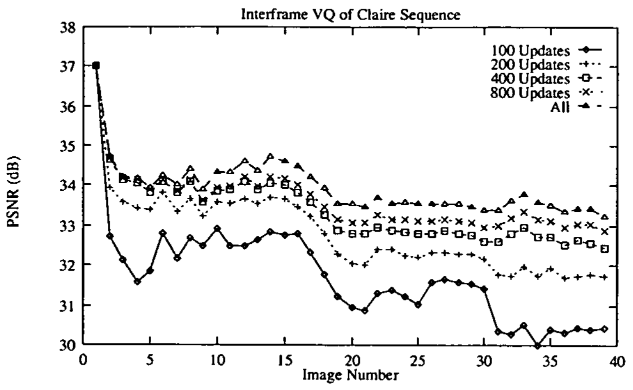
This algorithm was tested with a variety of sequences and a number of different codebook configurations. For this thesis, results will be shown for the two QCIF sequences, *Claire* and *Miss America*. The codebooks used contained 256 vectors of size 3×3 pixels, and were generated with the codebook generation scheme detailed in section 3.5, using the first image in the sequence as the training image. For each sequence, the number of label updates n was set at 100, 200, 400, 800 and 2832 (2832 is the total number of vectors in the image).

Results

Figure 4–4 presents the results for this experiment as a sequential frame error graph. These graphs plot PSNR against image number in sequence, and we can draw a number of conclusions from them :

PSNR vs n. PSNR performance improves as n is increased. This is an obvious result : it would be expected that more update information would improve the coded image. The relationship between update information rate and average PSNR is examined later.

Large interframe errors. When n is small, performance is seriously affected by large interframe differences. This can be seen especially with the Miss America sequence, where the subject moves her head a large distance which is reflected



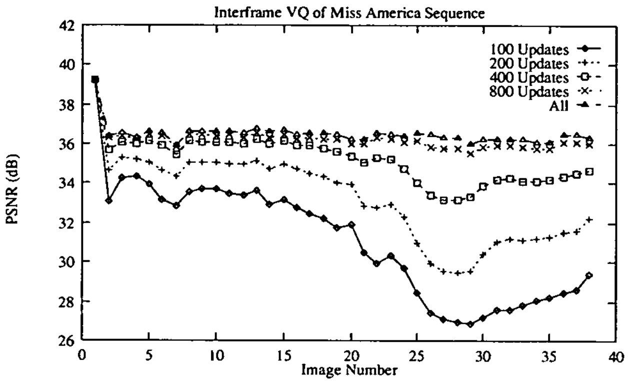


Figure 4–4: Sequential Frame Errors for Interframe VQ with Conditional Label Replenishment.

in the errors seen after image 20. This is because there are more vectors with large differences between frames than can be updated. This is seen in the coded sequence as a *trail* behind the movement and is very disturbing to the viewer. The coding scheme is able to recover some time after the period of large movements when the *trail* can be cleared up.

Codebook usefulness diverges. If we look at the performance when all vectors are updated at each frame, we can see that the codebook becomes less effective for coding this sequence as time progresses. This is more marked in the case of the *Claire* sequence. This is a very important result as it shows that label replenishment is fundamentally limited for coding sequences, as codebooks are indeed very image specific. These results, also, are for controlled videotelephony test sequences where there are no large scene changes. Any real sequence would include much greater image changes which would render simple label replenishment virtually useless.

We also wished to estimate the number of label updates required per frame to maintain a reasonable image at the decoder. To do this it is best to present the results as distortion-rate curves. This requires that the number of updates per frame, n, be translated into an information rate.

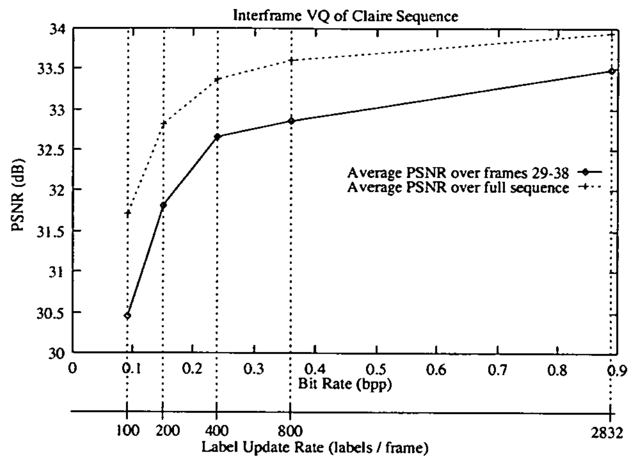
The information rate, in *bpp*, is calculted by dividing the number of bits required for each frame update by the number of pixels in each frame. Each frame update consists of the label updates plus the addressing information for the labels. Appendix A discusses some of the ways that the addressing can be done and calculates the number of bits needed to update 100, 200, 400 and 800 labels. There is no addressing overhead when all labels are being updated. Table 4–1 summarises the bit-rate calculations when a codebook of 256 vectors, of size 3×3 pixels is used.

Labels	Label Map (bits)	Labels (bits)	Total Bits	Bits per pixel
100	1200	800	2000	0.09
200	2244	1600	3844	0.15
400	2832	3200	6032	0.24
800	2832	6400	9232	0.36
2382	0	22656	22656	0.89

Table 4–1: Label update rates converted into bit rates in bits per pixel.

Figure 4–5 plots distortion-rate curves for the *Claire* and *Miss America* sequences with PSNR averaged over the full sequence and over the last 10 frames of the sequence. These results show little benefit is to be gained in updating more than 800 labels at each frame. Indeed, the average PSNR for 400 label updates is not appreciably worse than that for 800, and the bit-rate requirement is reduced by one third (from 0.36*bpp* to 0.24*bpp*) in this step.

These results coincide with a qualitative assessment of the coded sequences. With 400 label updates per frame the worst of the *trail* effects are eliminated and acceptable sequences are generated. Below this, considerable degradation occurs and the coded sequences are very poor, especially in periods of large motion.



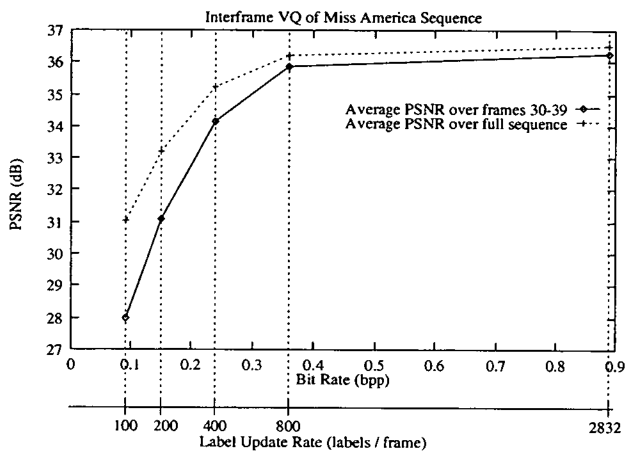


Figure 4–5: Distortion-Rate curves for Interframe VQ with Conditional Label Replenishment.

It may also be noted that the errors introduced by coding fewer vectors are much greater for the *Miss America* sequence. 100 label updates, compared with updating the full label map, incurs a 2.5dB penalty for *Claire* but a 5.5dB loss for *Miss America.* Although there are contributions to this from factors such as the contrast in the images, by far the largest effect is due to the larger interframe differences in the *Miss America* sequence. The image area covered by the head and shoulders in this sequence is greater, as is the distance of movement of the head. This is another result which confirms our predictions : larger interframe differences give more information to be coded.

The graphs also illustrate a result which we saw previously : that the effectiveness of the codebook diminishes as the sequence progresses. Average PSNR near the end of the sequence is lower than over the full sequence, even when the full label map is updated. This result again demonstrates the effectiveness of the codebook having diminished as the sequence progresses.

4.4.2 Interframe Error VQ

This section presents an investigation into an alternative label replenishment scheme which was developed by the author. The algorithm will be described, followed by some results for coding the two QCIF sequences used to test the coding scheme of the previous section. These results will then be analysed, including a comparison with those of the last section.

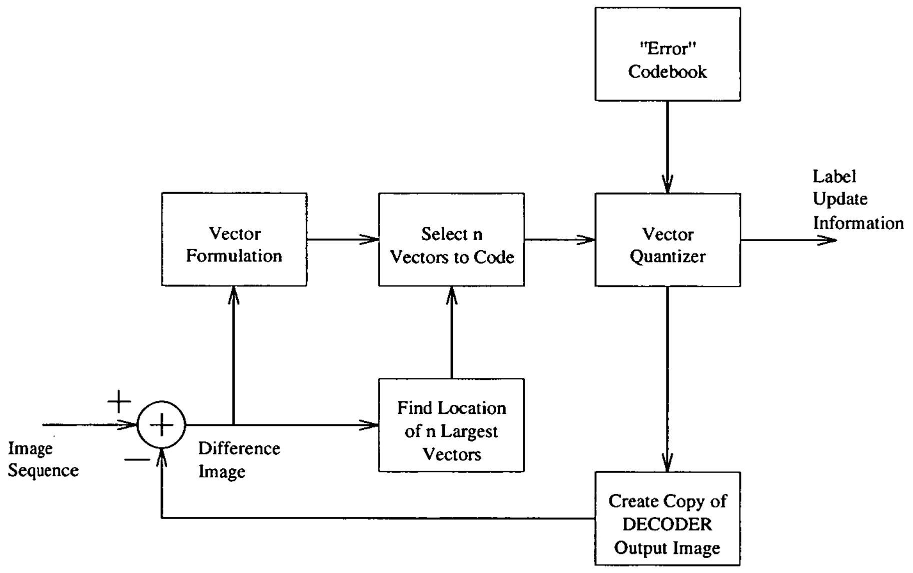


Figure 4–6: Interframe Error VQ

The Algorithm

The algorithm for Interframe Error VQ (IEVQ) is shown in figure 4–6. IEVQ attempts to gain advantage over standard label replenishment techniques by quantizing not the original image but the difference image between the current coded image and the next image in the sequence. It was hoped that, because there is less information content in a difference image, it would be possible to generate codebooks which could code the errors more effectively than a standard codebook could code the original images.

The coding scheme is very similar in operation to the label replenishment scheme of section 4.4.1. The fundamental difference is that the largest n vectors in the difference image are coded directly, rather than the associated vectors from the original image. These vectors are coded with the 'Error' Codebook.

Error Codebook Generation

Generation of the error codebook was done with a modified version of the method presented in section 3.5. Training data for this was created by modifying the IEVQ coder slightly as shown in figure 4–7. Those vectors chosen to be updated are copied in full to the coded image without the error that coding would introduce. This coding can be thought of as performed with a 'perfect' codebook. The same vectors are also collected to create a database of error vectors. At the end of coding a sequence in this way, there exists a sufficient database of vectors to form dummy images from error vectors, and these dummy images could then be used as the training images for codebook generation.

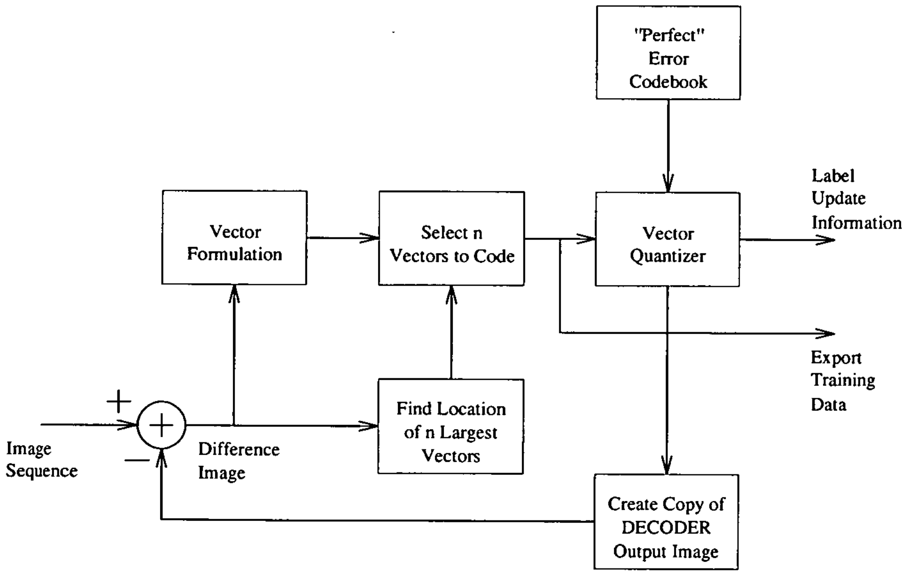


Figure 4–7: Creation of Codebook Generation Training Data for Interframe Error VQ

Results

As with the previous experiment, the results for the algorithm are presented for the *Claire* and *Miss America* sequences, and for codebooks of size 256 $\times$ 3 $\times$ 3. Figures 4–8 and 4–9 show the sequential frame errors and distortion-rate curves, respectively.

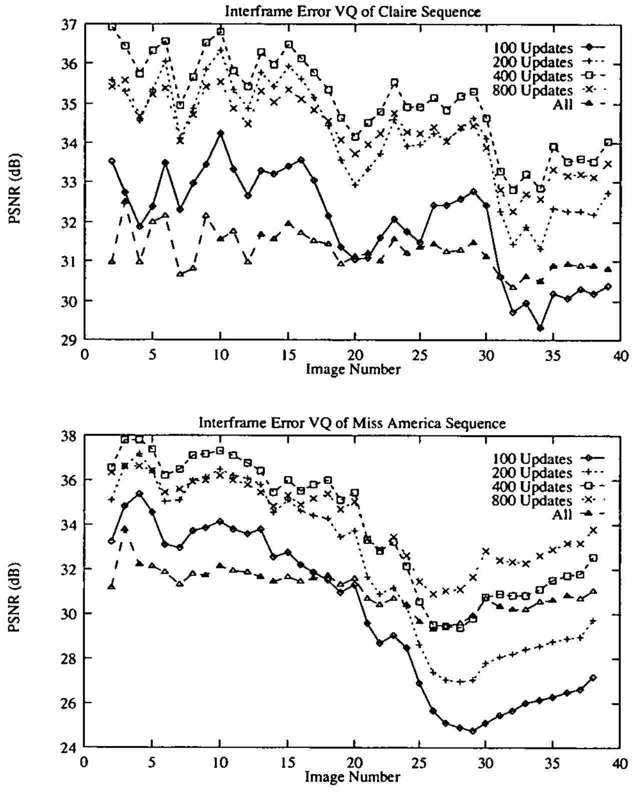
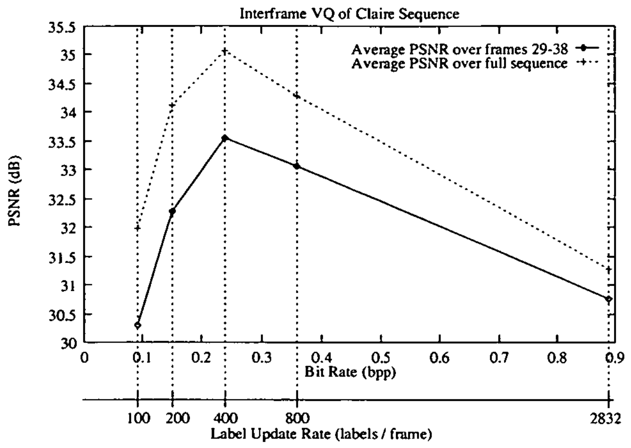


Figure 4–8: Sequential Frame Errors for Interframe Error VQ with Conditional Label Replenishment.

This algorithm does not include any means for coding the first frame of the sequence. For the purposes of this study, this was not considered an important factor. The tests were done with the decoder output image initialised to the first image in the sequence, and the input image sequence starting from image number 2. The problem of initialisation is studied separately in section 4.5.

Examining the results, perhaps the most striking feature of them is that noise performance does not always improve for an increase in the update rate. For both sequences, better results are obtained by updating 800 vectors rather than all the vectors in the im-



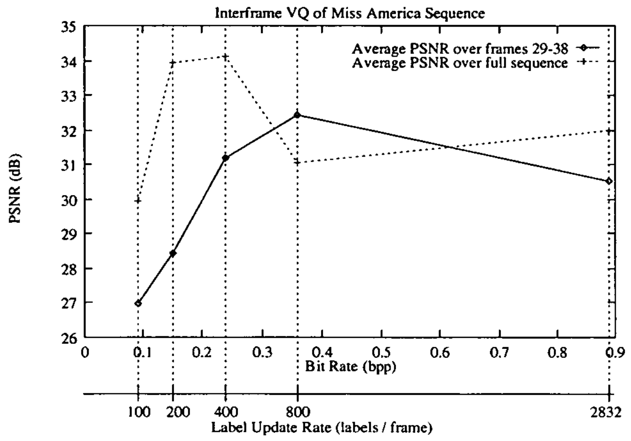


Figure 4–9: Distortion-Rate curves for Interframe Error VQ with Conditional Label Replenishment.

age. Indeed, for the *Claire* sequence, updating 400 labels always gives a better result than updating 800 labels.

This result was attributed to the fact that the codebooks were trained on error data, and were very poor at coding those areas of the difference image which were close to zero. If we examine the 400 updates and 800 updates curves for *Miss America*, we can see that the 800 update result is poorer in the first half of the sequence, but better in the second half (the transition occurs in the few frames after frame 20). This relates to the frames of the sequence where the head motion becomes much more marked, and difference images have more information content as a result of this. The greater number of vectors with considerable information are coded more effectively with the

800 updates than with 400 updates. Before this transition, the difference images contain fewer than 800 vectors which are large enough to need updating. A considerable number of the updates are used to code small vectors inefficiently, as the error codebook has not been designed to code this type of vector.

This result suggests that the error codebooks have been created using the wrong type of training data. Using full error images as training data would be one possible option. Alternatively, a lower threshold could be placed on the size of an error vector, below which the coder would not update the vector.

A subjective assessment of the coded sequences reveals results which are very close to those of the previous section. Again, we see *trailing* effects during periods of large movement when insufficient updates are available to code the difference image. The problems discussed above mean that the background of the image, where the interframe differences are small, is noisy when the update rate is higher than is required. This noise is not large, and does not have a large impact on any subjective assessment of the image sequence.

However, one aspect of the results which mirrored those of section 4.4.1 is that it was not possible to maintain a set noise performance for an indefinite time. Again, we see even that for a relatively simple test sequence, a non-adaptive coder will be unable to code it without a degradation in PSNR over a long sequence. This result once more suggests that adaptive techniques will be required to solve this problem effectively .

4.5 Initialisation Methods

This section takes a brief look at the problem of initialisation of an image coding link. The problem can be divided into the two separate problems of generation and transmission of the codebook, and coding and transmission of the first image. The performance of a simple, near-zero bit-rate initialisation method, which overcomes the problem of coding and transmitting the first image, is presented and a possible solution is analysed through experiment.

The first image

Coding the first image in a sequence is different, essentially, because there is no previous image with which to form a difference image. The formation of a difference image is done in both of the label replenishment schemes described earlier in this section. For the general label replenishment scheme described by figure 4–1, a label map does not exist before the first image is coded. Effectively, though, the problem is the same : to effectively code the first image we require much more information than is required once the link has been established.

The problem of initialisation exists when we are considering fixed bandwidth data channels. In this case it is not possible to increase the data transmission rate to allow for the extra coding information to be transmitted in the interval before the second frame of the sequence. If a variable bit-rate channel were being used, it would be expedient simply to code the first frame fully and update the entire label map.

The problem also exists in cases where there is a limit on the amount of the image which can be coded in the time between frames. This describes a system which has a limited *processing bandwidth.* In any efficiently designed system there will not be a great deal of unused processing bandwidth.

For the systems outlined earlier in this section, the bulk of the processing requirement is for the vector quantizer block itself. The amount of processing that the vector quantizer will do is directly proportional to the number of vectors, n, which have been selected to be coded. In many applications, n would be fixed, and the vector quantizer would be designed to match the required processing bandwidth. In such a system, it would not be possible to code more than n vectors in a single interframe period.

The codebook

We must also remember that, for label replenishment schemes, generation of a codebook is associated with the first image in a sequence. If we had access to unlimited processing bandwidth and information bandwidth it would be possible to generate a codebook by training on the first image of the sequence, and transmit this codebook before coding the first image. This is the situation which has been modelled in the experiments of section 4.4.1 and 4.4.2, but it is not realistic.

A realistic label replenishment scheme would have to generate a codebook not on the first image of a sequence, but on an image taken some time before the link has been started. Codebook generation would be done in a time dictated by the processing resources available, so the codebook generation technique would be tailored not only for error performance but for speed of convergence. The codebook would then be sent as quickly as the data channel would allow. The sequence coding could then begin, with the next frame from the video output being taken as the first image in the sequence.

An estimation of the time taken to generate and transmit a codebook can quickly reveal the fact that the initialisation delay it could cause would be quite considerable. It turns out that transmitting the codebook is the lesser of these two tasks. In the example scenario from section 4.4.1, the codebook of size $256 \times 3 \times 3$ pixels, each resolved to 8 bits, contains 18432 bits of information. This is the equivalent of approximately 2 frames of 800 label updates (at 9232 bits per frame, see table 4–1) or 3 frames of 400 updates (6032 bits per frame).

Generation of the codebook, however, would be a considerably worse problem if there were no extra processing power available for the task. The codebook generation method presented in section 3.5 would take approximately 10 iterations to converge. At each iteration the processing required is dominated again by the work done by the vector quantizer, which codes the training sequence once. Thus, if the training set were a single image, as would normally be the case, this codebook generation method would take, perhaps, the same time as it would take to code all the vectors in 10 frames. For the considered example, if the coder were designed to code 800 vectors per frame, which is approximately one quarter of the vectors in the image, codebook generation would then take perhaps the equivalent of 40 frames.

No formal methods for coping with the problems of codebook initialisation will be studied further here. Codebook replenishment techniques such as those developed in section 4.7 have the added advantage that they can let a global codebook, known to both the receiver and transmitter, be used from the start of a sequence, from where it can adapt to an image specific codebook. This will be studied in greater depth later. We will, however, look at the problem of coding the first image. This problem exists for both label replenishment and codebook replenishment schemes.

The basis of this problem, as was stated before, is that there is no previous image to use to form a difference image. If a constant illumination image is used as the 'previous' image, it is possible to initiate this with zero, or very near zero, data or processing overhead. We only require that both receiver and transmitter start with the same value. Thus, if the transmitter is to calculate the value it will use, only one byte of information would be needed to be sent to the receiver.

One could fix the illumination level, and as this could be known at both receiver and transmitter, there would be no need to transmit any data. A sensible value to fix this at would be mid-scale, or perhaps the average illumination level in the previous transmission. However, there are very simple ways to set the illumination level more intelligently.

The average intensity of the first frame could be used, which would be simple to calculate. This may, though, be a very ineffective method for certain images. Consider, as an example, an image of a chess board, where the white squares are the brightest possible pixels and the black squares are the darkest possible pixels. The average intensity of this image would then be mid-scale. However, all pixels will be badly coded. This example illustrates what it is that we must try to do : create an image which contains the fewest number of pixels needing updated. In other words, the image should contain the most number of pixels which do not need updated. This is done if we choose the intensity value to be the *modal* value of the first image.

The performance of this method was tested by coding the *Claire* and *Miss America* sequences using the conditional label replenishment technique of section 4.4.1 but with the modification shown in figure 4–10. Here the modal pixel value of the first image is calculated and all the pixels in the copy of the output image are set to this value, at both the coder and decoder. Coding is done from there on exactly as it was in the previous experiment.

The results for coding the image sequences with the above method are shown in figure 4–11 as a sequential frame error graph. Results are given for 400 and 800 label

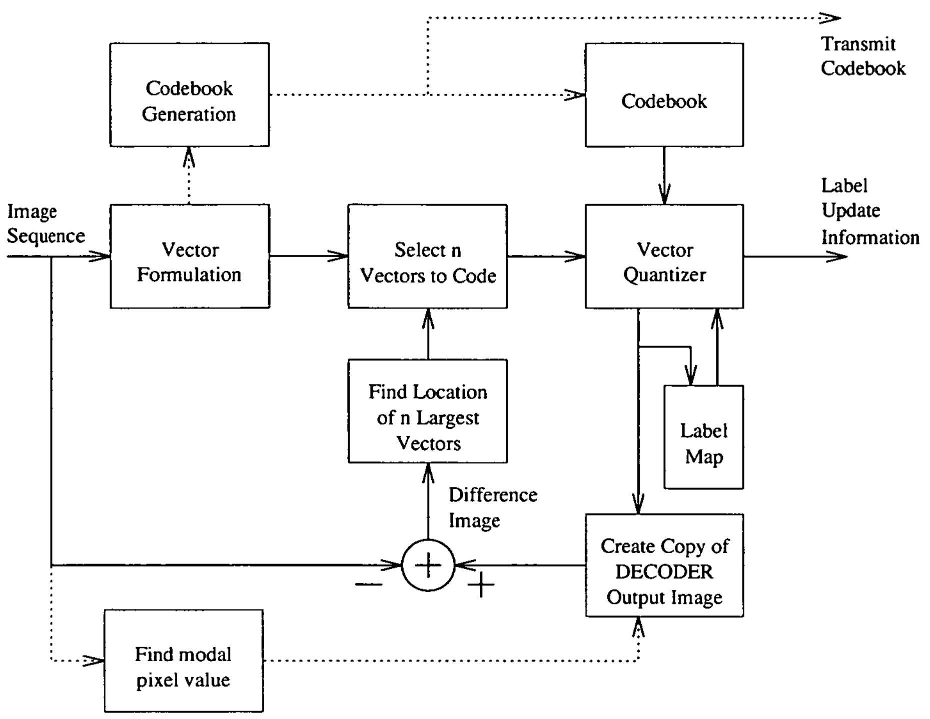


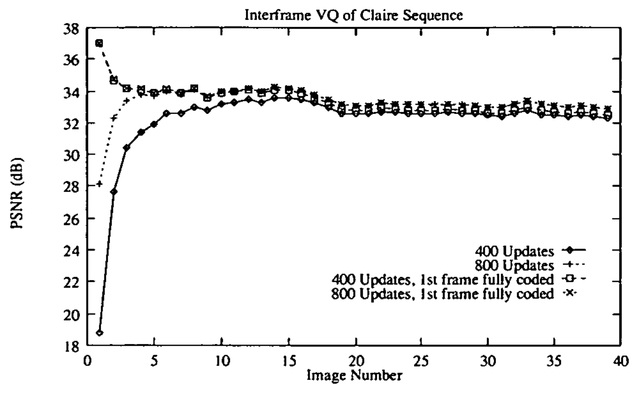
Figure 4–10: Conditional label replenishment with initialisation methods (CODER only shown). Dotted lines show the actions performed on the first image in the sequence.

updates per frame, and these are displayed along with the results for the same update rates from section 4.4.1, where the first frame was fully coded.

It can be seen that the error performance converges very rapidly towards the results for the sequences with fully coded first frames. When 800 label updates per frame are being coded, convergence to within a fraction of 1dB occurs within 5 frames for both sequences. For 400 label updates, convergence is neither so rapid, nor so complete. The error performance is still approximately 0.5dB worse in both sequences after 15 frames. However, to the human observer, the coded sequences for the two methods are almost indistinguishable after 5 images of the sequence.

Summary

In summary, we can say that the problem of initialisation can be broken down into the problem of creating and transmitting a codebook, and that of coding and transmitting



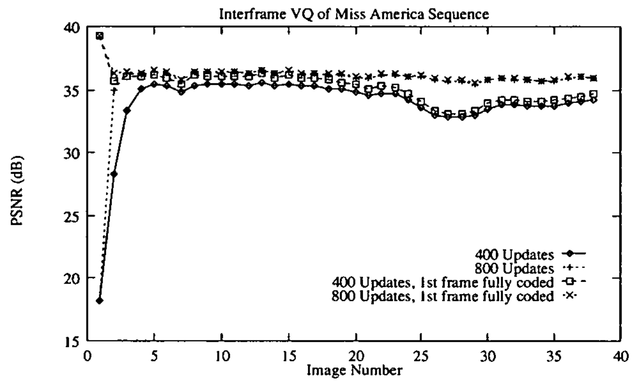


Figure 4–11: Sequential Frame Errors for Interframe VQ with Conditional Label Replenishment.

the first image. Codebook generation and transmission could result in a considerable time delay at startup. This is more likely to be caused by limitations on the processing power available than the time it would take to transmit the codebook from coder to receiver. Coding and transmitting the first image would cause a problem when either the processing bandwidth or channel information bandwidth are fixed and limited. However, a simple technique which uses the modal pixel value to set the initial condition of the coder has been shown to be very effective.

4.6 Summary of Label Replenishment Methods

Label replenishment techniques are the simplest form of interframe coding using vector quantization. Two techniques, developed by the author, have been presented. These algorithms use conditional label replenishment where the areas of the image which have changed the most between frames are identified and coded. This ensures that the data bandwidth and processing bandwidth available are used to the best advantage. These methods have been shown to produce good coded sequences for very low update rates. However, fundamental problems remain with label replenishment. The major problems are the generation and transmission of the codebook at the start of a transmission, and the degradation in the suitability of this codebook over time.

In section 4.5 the problems of initiating a transmission were discussed. Coding and transmitting the first image was seen to be a problem which could be countered simply and effectively. Generation and, to a lesser extent, transmission of a codebook was seen to be a major problem. Codebook generation would either require much greater processing power than the coder itself, or would take a considerable length of time.

Sections 4.4.1 and 4.4.2 presented two schemes where the initialisation problem was ignored by not considering the time or data transmission needed for codebook generation. Coding errors were analysed with respect to update rates, producing largely predictable results. However, the fundamental problem remained that even with no limit placed on the update rate, the usefulness of the codebook diminished with time.

In the next section codebook replenishment schemes are studied. We will look at the effectiveness of these methods at overcoming the problems identified above.

4.7 Codebook Replenishment

This section presents the development and assessment of codebook adaption techniques which have been developed by the author. The aim of this work is to create adaption techniques which are effective at maintaining a good codebook but incur small bit-rate and processing costs above a non-adaptive system. This section includes experimental results which show that the presented algorithms are capable of achieving this aim.

An ideal codebook adaption scheme would take the following form :

1. Present each input image as training data to a codebook generator, using the codebook from the previous frame as an initial codebook.

2. Code the input image with the codebook generated in 1.

3. Transmit the entire new codebook and a full label update to the receiver.

If we are considering a real application, this ideal scheme is far too expensive in processing and bit-rate terms. Codebook generation at each frame, even if only one iteration of an LBG-type generator were to be performed, would be a considerable processing overhead. Updating the entire codebook at each frame also would be prohibitively expensive in terms of the data overhead it would add to the update information. For the example configuration used in the experiments of section 4.4 (which will be used in this section also), the amount of information in a full codebook is 2306 bytes ($256 \times 3 \times 3$), which is not much less than a full label map of 2832 bytes.

Fortunately, it is clear that the ideal scheme is also highly inefficient, and it should be possible to obtain good results from much simpler methods. It is inefficient because the number of codewords which would change by any appreciable amount at each frame is likely to be small. If one image is not much different from the preceeding one, the codebook generated for this image will include very few changes.

Of course, if there is a considerable change in the image, this would be reflected by a marked change in the codebook. However, these occasions can be considered to be rare, as complete scene changes will be infrequent. The aim in this situation should be to adapt the codebook as quickly as possible, rather than require an instant response.

Section 3.5 presented an LBG based codebook generation technique with intelligent methods for deletion of redundant codewords and generation of useful codewords to replace those deleted. As with LBG codebook generation, the entire codebook was 'centred' at each iteration (*i.e.* each codeword was moved to the centroid of its cluster). Codebook generation can thus be split into the two actions of deleting/replacing and centring. These two actions will be analysed separately to optimise the gains that can be made with them when they are applied to codebook adaption for interframe coding.

The aims of this work were thus identified to be :

- To develop an optimal technique to delete and replace redundant codewords.

- To optimise the number of codewords to delete and replace.

- To identify the codewords which would benefit most from being centred.
- To optimise the number of codewords to centre.

The remainder of this section presents a summary of the experimental work performed in the pursuit of these aims. Section 4.7.2 considers the action of deleting and replacing codewords. In section 4.7.3, the action of centring is analysed. Section 4.7.4 presents a summary of the work done on codebook replenishment, including an analysis of the improvement in sequence coding that can be obtained. First, a brief note on the experiments presented here will be given in section 4.7.1.

4.7.1 A note on experimental procedures

We wish to test the ability of the codebook replenishment schemes to perform under two different sets of conditions : when the codebook is known to be good at the start of a sequence, and when it is known to be poor. The first case enables us to see if the codebook adaption can prevent the codebook usefulness from diminishing over time. The second case enables us to see how the adaption can generate a useful codebook from one which is initially ill-suited to the given image. This is analogous to the situation where a complete scene change has occurred.

Once more, the two QCIF sequences *Claire* and *Miss America* were used for these experiments. Three different initial codebooks were used, one generated from the first image of each of these two sequences, and one generated from a completely different image. This image was the first in another QCIF sequence called *Salesman*, and it was chosen because it produced a codebook which was very different from the other two. The codebooks were produced using the generation technique detailed in section 3.5. In this way, each of the two sequences was tested with one initially good codebook, and two initially bad codebooks. As in section 4.4, a codebook of size $256 \times 3 \times 3$ is used. These three codebooks are shown in figure 4–12.

For these experiments, the full label map was updated at each frame. The aim was to look at the problem of codebook replenishment on its own, and with full label replenishment this aspect was isolated for investigation.

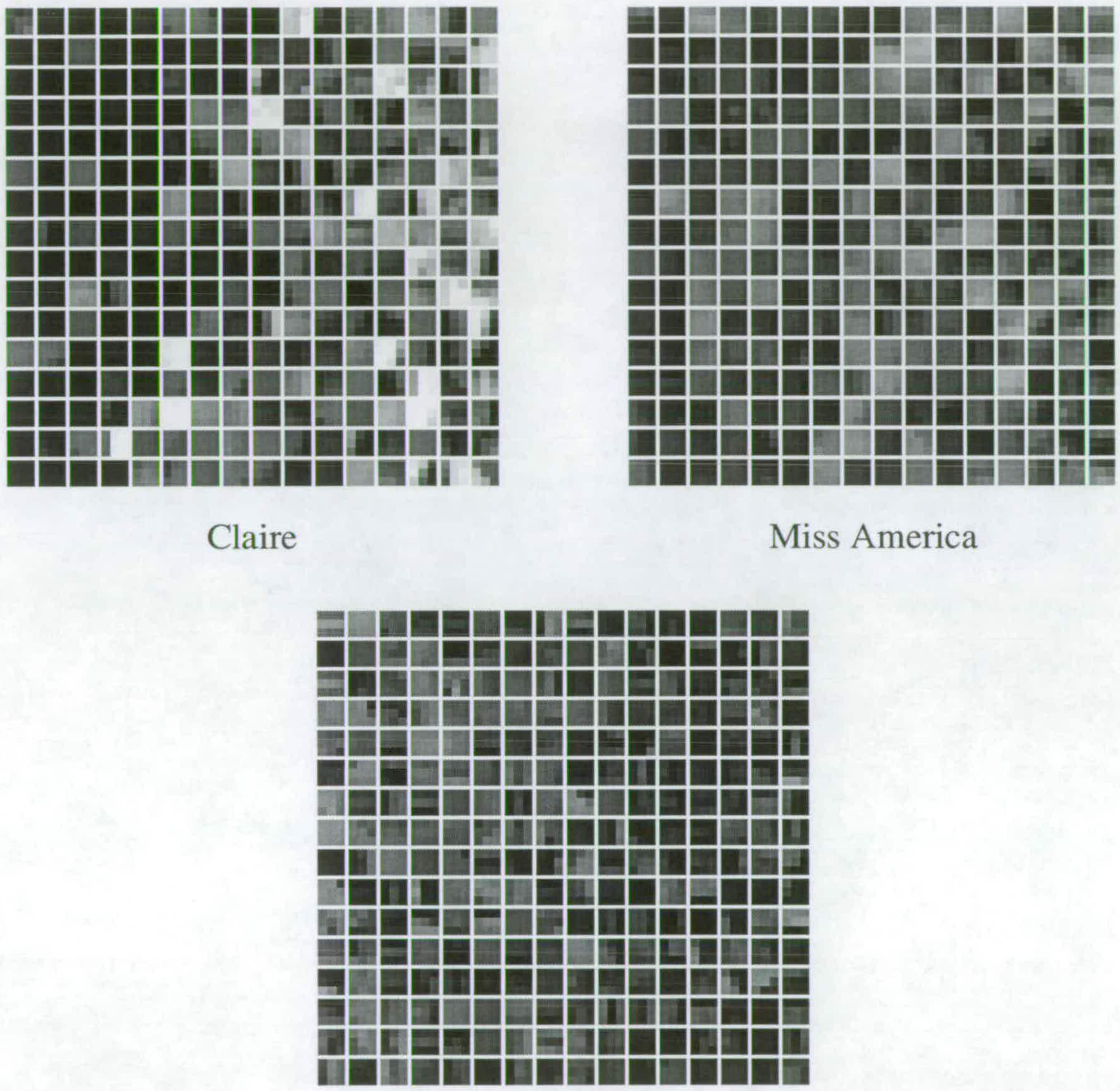


Figure 4–12: Three initial codebooks used in codebook replenishment experimentation.

4.7.2 Redundant codeword deletion and replacement

While this section details the delete/replace operation, the experiments were actually performed with the full adaption algorithm. This includes the operation of centring as well as that of delete/replace. For the purposes of the experiments in this section, the number of codewords centred at each frame was kept constant. The full codebook replenishment algorithm is shown in figure 4–13.

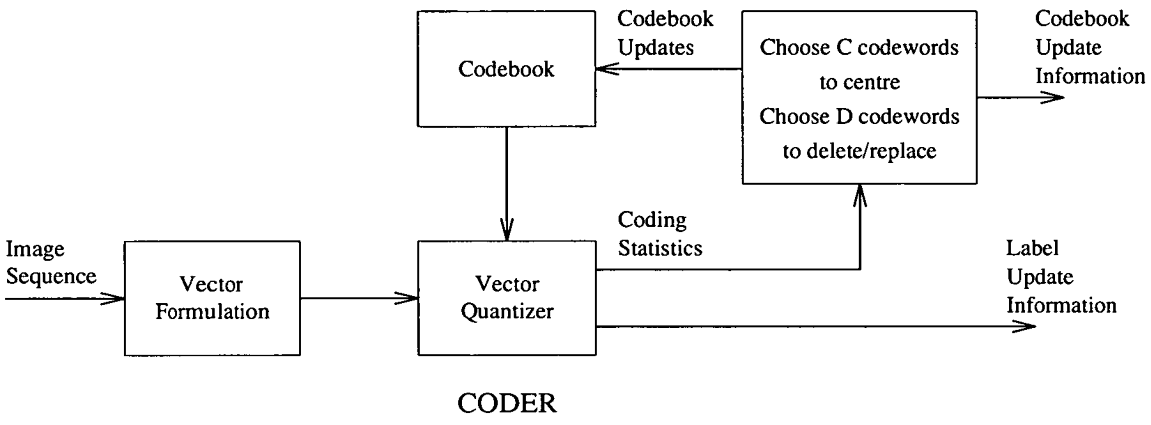


Figure 4–13: Codebook replenishment algorithm.

At each frame, the image is VQ coded as with a simple label replenishment system, and the full label map is sent to the receiver. The action of vector quantization creates, as a by-product, a collection of statistics on the codebook usage and cluster sizes. From these statistics, C codewords are chosen to be centred, and D codewords are selected to be deleted and replaced with another codeword. These codebook updates are then sent to the receiver to update the codebook before the next image is coded.

This describes a system which requires little extra processing above that done by the vector quantizer. There is some calculation to be done to select the codewords which are to be updated, but this amount is small compared to the vector quantization. This achieves our stated aim of reducing the additional processing overhead to a minimum.

It is not possible at this point to present all the different selection and replacement methods that were tested during the development of the algorithm. Instead, each component of the algorithm will be described, with some explanation of the way it was developed.

Redundant codeword selection

After vector quantization, a **redundancy** value was calculated for each codeword, given in equation 4.1 :

$$\textbf{redundancy} = \textbf{unused image count} + \frac{1}{\textbf{codeword}\ \textit{spread}} \tag{4.1}$$

where **unused image count** is the number of images for which the codeword has been unused, and **codeword** *spread* is a measure of the dynamic range of the pixel values in a codeword. Equation 4.2 gives **codeword** *spread* as

$$\textbf{codeword}\ \textit{spread} = \sum_{j=1}^{K}(x_j - \bar{x})^2 \tag{4.2}$$

for a codeword $\underline{x}$ of dimension K, where $\bar{x}$ is the average pixel value of the codeword.

The codebook is then ranked according to the **redundancy** value of each codeword. This places the longest unused and lowest spread codewords at the top of the ranked list, and it is these codewords which we wish to delete. The spread measure ensures that it is more difficult to discard codewords with edge detail in them than codewords with near-uniform intensity. This action was seen to prevent the codebook from becoming populated with uniform codewords.

Replacement codeword generation

Replacement codeword generation was performed by generating a close copy of one of the existing codewords. A similar process was used in the codebook generation technique of section 3.5. It requires a method of selecting a codeword to spawn from, and the spawning method itself.

Selection of a suitable codeword to spawn from was done by taking those codewords with a large number of vectors in their cluster, then ranking these vectors according to their cluster volume (in Euclidian terms). For these experiments valid clusters were identified as having greater than 8 vectors in their cluster. This process creates a ranked list of codewords with those having a large number of vectors in their cluster, and large cluster volumes, at the top of the list. This identifies valuable codewords in a similar manner to that used in the codebook generation scheme of section 3.5.

The codebook generation scheme spawned a second codeword by making one slightly brighter and one slightly darker copy of the chosen codeword. This method was tried here but was seen to be too destructive : useful codewords were being disturbed with a detrimental effect on the coding errors. Instead, the spawning method developed for this algorithm was to take the selected codeword and make a close copy of it by perturbing each of the pixels in it by a small random value. A rectangular distribution was used for the perturbation values. The selected codeword was left undisturbed. Through further experimentation the size of the perturbation was chosen to be a value of between 0 and 4 for each pixel.

Experimental Details

Using the algorithm described above, the two QCIF sequences were coded, using the three initial codebooks, as descibed in section 4.7.1. For each combination of sequence and initial codebook, the number of codewords to delete and replace, D, was varied across the range of values (0,1,2,4,8). In all cases the number of codewords centred, C, was set to 2. We shall study the action of centring codewords in detail later in section 4.7.3.

Results

Once again, we can examine the performance of an interframe coding technique by looking at the sequential frame error graphs. Figure 4–14 shows these for the *Miss America* sequence, with each of the three graphs representing a different initial codebook. In each graph, there are 5 curves showing the performance for each of the values of D. Figure 4–15 shows the same format of results for the *Claire* sequence.

The clearest conclusion we can draw from these graphs is that the action of deleting/replacing is of considerable value when the initial codebook is poor. This is seen to be most marked in the case of the *Claire* sequence where, with either of the poor initial codebooks, there is a considerable performance improvement to be gained : the results shown for $D = 0$ stand out as much poorer than those for $D \neq 0$.

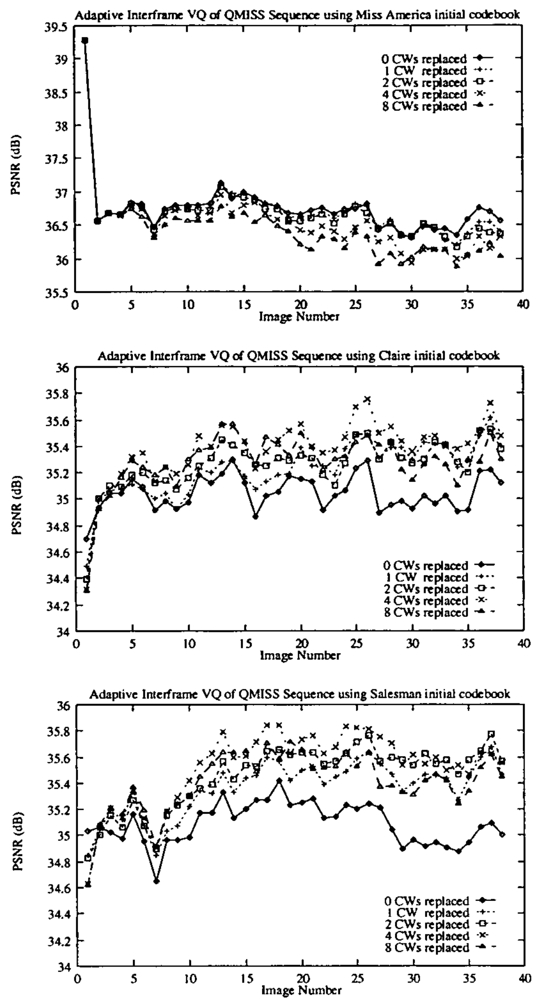


Figure 4–14: *Miss America* (or QMISS) sequence coded using three different initial codebooks, varying the number of codewords to delete/replace per frame while centring a constant 2 codewords per frame.

It is perhaps easier to interpret these results if they are presented in another way. For the *Miss America* sequence, figure 4–16 shows the results as an average PSNR (measured over the last 20 frames of the sequence) against D for each of the initial codebooks. Figure 4–17 shows the results for the *Claire* sequence displayed in the same manner.

This clearly shows that the delete/replace operation is essential to enable the codebook to adapt when it is initially poor. As was stated before, this condition is directly analogous to the occurrence of a complete scene change in an image sequence. It is also

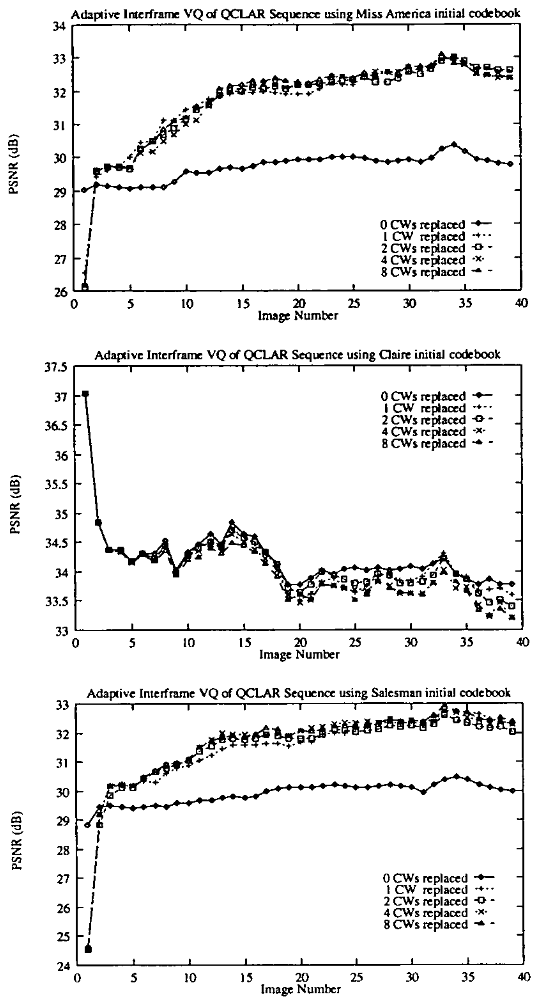


Figure 4–15: *Claire* (or QCLAR) sequence coded using three different initial codebooks, varying the number of codewords to delete/replace per frame while centring a constant 2 codewords per frame.

clear that only one of these operations per frame is sufficient to enable this codebook adaption to take place.

These graphs also show that the delete/replace operation is destructive in cases where the codebook is already good. Here, we wish to minimise the number of operations to prevent a good codebook from being gradually eroded. Taken in tandem with the results for initially poor codebooks, we can say that the optimum number of delete/replace operations to perform per frame is one. This is the best compromise which gives the adaption process sufficient volatility to generate completely new vectors when the

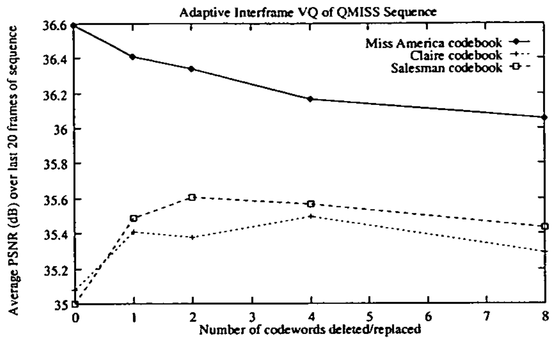


Figure 4–16: Average PSNR over the last 20 frames of the *Miss America* sequence coded using three different initial codebooks, varying the number of codewords to delete/replace per frame while centring a constant 2 codewords per frame.

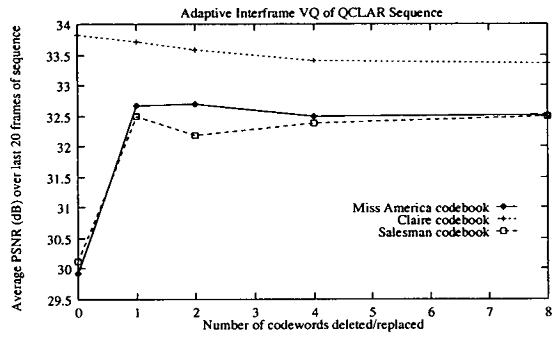


Figure 4–17: Average PSNR over the last 20 frames of the *Claire* sequence coded using three different initial codebooks, varying the number of codewords to delete/replace per frame while centring a constant 2 codewords per frame.

codebook is poor, and preventing the deletion of useful codewords when the codebook is good.

What these graphs also show is that the codebook adaption that is taking place here is insufficient to enable an initially poor codebook to adapt to a position where it is as good as a codebook which was initially good (within the limited sequence lengths used in these experiments). For both sequences, the performance after a poor initial start is approximately 1dB down on that where a good initial codebook is used. A fuller analysis of the performance of the algorithm as a whole is given in section 4.7.4.

4.7.3 Codeword centring

The tasks which make up this adaption operation are those of identifying the most valuable codewords to centre, and the task of actually moving the codeword to its cluster centroid. This is a considerably easier process to do than the delete/replace operation.

Moving the codewords to their cluster centroids is exactly the same process as used in LBG codebook generation. Each pixel in the codeword is set to the average value of the corresponding pixel in all of the vectors assigned to that cluster. This requires that a total for each pixel be accumulated as each vector is added to the cluster. This total is then divided by the number of vectors in that cluster on completion of the quantization of the image.

Identifying valuable codewords to centre could be quite a complex process if an ideal solution is sought. In such a case, the full codebook could be centred, and for each codeword the potential error saving could be calculated by finding the new cluster distortion. In this way the codewords which offered the biggest error savings could be identified. This, though, is a complex process, and a much simpler method was sought for this algorithm.

In practice, it was found that total cluster distortion was found to be an effective way of identifying those codewords we wish to centre. This number is another by-product of the quantization process, being simply the sum of the distances to each of the vectors in the cluster. The codebook was ranked on this value at the end of the quantization process, and those at the top of the list were those chosen to be centred.

Experimental Details

The experiments presented here are very similar in format to those of section 4.7.2. As before, the two QCIF sequences were coded using the three initial codebooks. For each combination of sequence and initial codebook, the number of codewords to centre, C, was varied across the range of values (0,1,2,4,8,16,32). In all cases the number of codewords deleted/replaced, D, was set to 1. This value was chosen to reflect the results from the previous section, 4.7.2, where deleting and replacing a single codeword at each frame was shown to be the best compromise between giving a codebook enough energy to adapt and preventing deletion of useful vectors.

Results

Initial assessment of the results can be done once again by looking at the sequential frame error graphs. Figure 4–18 shows these for the *Miss America* sequence, with each of the three upper graphs representing a different initial codebook. In each of these, there are 7 curves showing the performance for each of the values of C. The bottom of the four graphs shows the average PSNR (measured over the last 20 frames of the sequence) against C for each of the initial codebooks. Figure 4–19 shows the results for the *Claire* sequence in the same format.

These results show that as the number of centring operations, C, is increased, the error performance is improved. For the case where a poor codebook is used at the start of the sequence, it can be clearly seen that centring just one codeword per frame results in a marked improvement. This is especially true of the *Claire* sequence.

Looking at the graph of average PSNR against C, we see that an improvement is always obtained by increasing the number of codewords that are centred at each codebook update. However, it is clear that the first one or two codewords are the most important, and that centring more than this does not give much more of an improvement. In terms of the bit-rate costs for codeword replenishment by centring, the first two codewords can be seen as very valuable, with more updates than that being a relative luxury.

These results also show that the codebook adaption process is unable to take an initially poor codebook and make it as effective as an initially good codebook, at least not within the time allowed by the limited lengths of these sequences. However, looking at the results for the poor codebooks alone is very encouraging. We see that, for both sequences, the results for both poor codebooks are very similar. This suggests that from any reasonable starting point, the adaption process should be able to generate a codebook which performs adequately, within perhaps 1dB of the results for an ideal starting point. This result is very important because it suggests the feasiblity of a system where no codebook generation need be done at the start of a sequence.

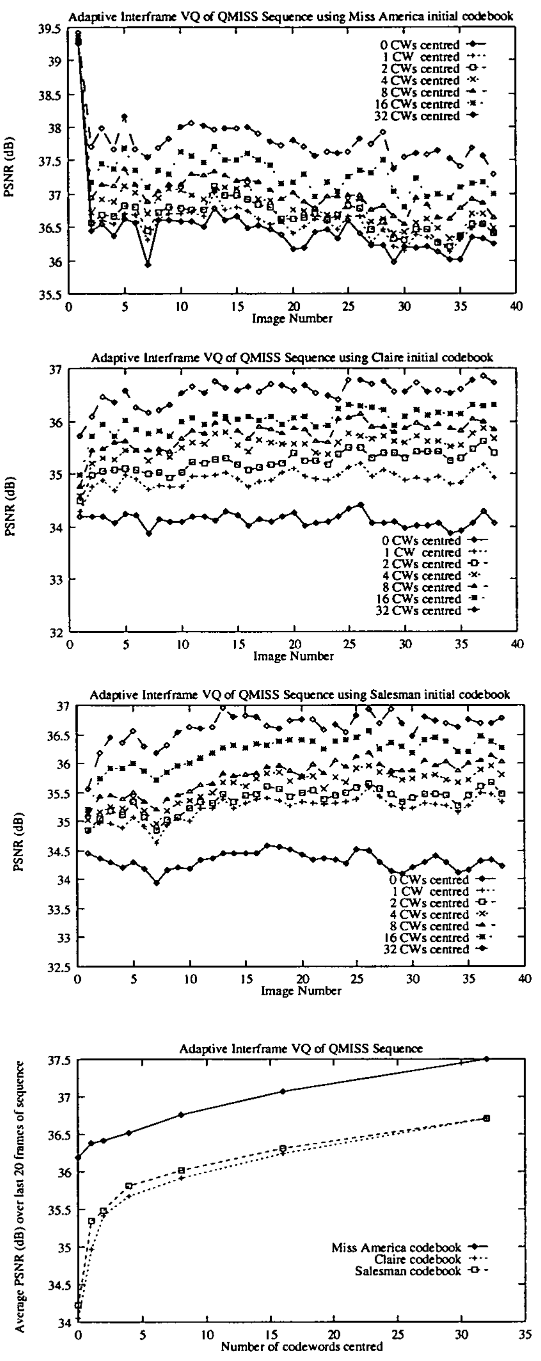


Figure 4–18: *Miss America* sequence coded using three different initial codebooks, varying the number of codewords to centre per frame while deleting/replacing a constant 1 codeword per frame. The bottom graph shows the average PSNR over the last 20 frames of the sequence.

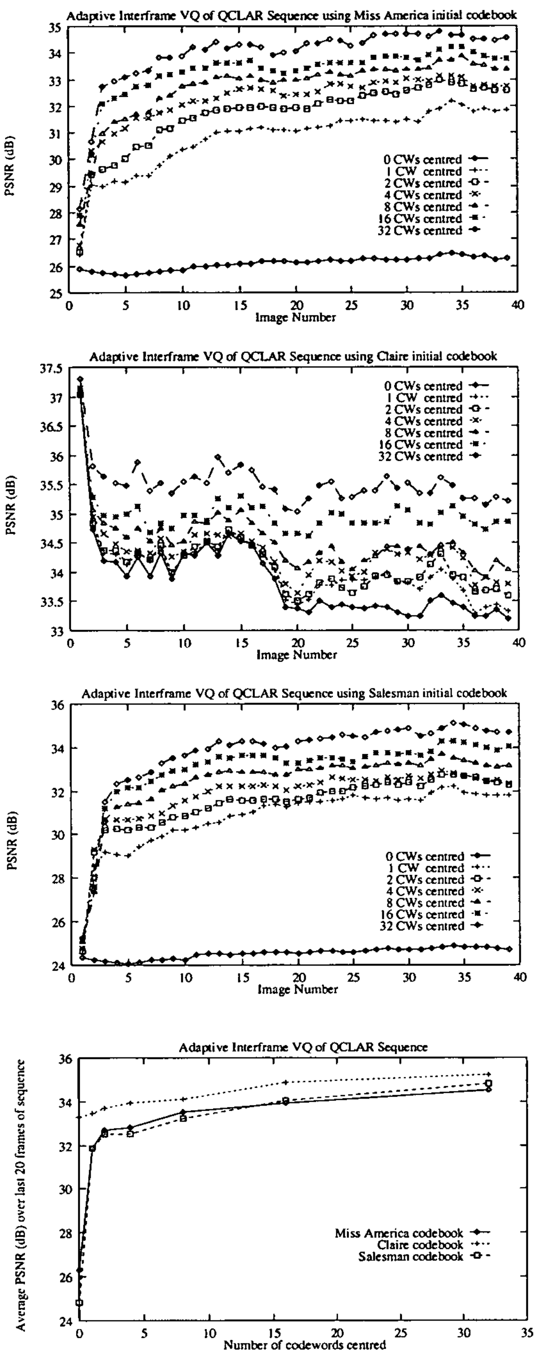


Figure 4–19: *Claire* sequence coded using three different initial codebooks, varying the number of codewords to centre per frame while deleting/replacing a constant 1 codeword per frame. The bottom graph shows the average PSNR over the last 20 frames of the sequence.

4.7.4 Summary of Codebook Replenishment

The preceeding pages of this section have described a codebook replenishment technique which :

- Performs codebook adaption with the processes of delete/replace to provide new codewords and centring to improve existing codewords.
- Identifies the codewords to be operated on with simple techniques using the statistics of the vector quantization of the current image.
- Requires only very small codebook update rates to obtain considerable performance improvement.

We will conclude this work with a look at the benefits that are obtained when the algorithm is used with the number to delete/replace set to one, and the number to centre set to two. We will also calculate the data overhead which this update rate incurs.

The six sequence/codebook combinations were coded with the above adaption parameters ($D = 1, C = 2$) and with no codebook adaption ($D = 0, C = 0$). Table 4–2 shows the average PSNR over the last 20 frames of the sequence for each of these cases. This table also shows the coding improvement gained by performing the codebook adaption.

	Initial	PSNR over last 20 frames (dB)		Adaption
Sequence	Codebook	D=0, C=0	D=1, C=2	Improvement (dB)
QMISS	Miss America	36.29	36.41	0.12
QMISS	Claire	33.83	35.41	1.58
QMISS	Salesman	33.87	35.49	1.62
QCLAR	Miss America	25.83	32.67	6.84
QCLAR	Claire	33.40	33.71	0.31
QCLAR	Salesman	24.31	32.49	8.18

Table 4–2: Summary of improvement gained by codebook adaption.

The data overhead for these codebook updates is 2 bytes for the delete/replace operation, and 20 bytes for the 2 centring operations. The 2 bytes for the delete/replace operation identify the codeword to be deleted and the codeword to be spawned from : both coder and decoder will use the same spawning algorithm so there is no need to transmit the entire codeword. Updating a codeword which has been centred takes 10 bytes : 1 byte to identify the codeword, and 9 bytes (3×3 pixels) to describe the new codeword in full. This data overhead of 22 bytes is less than 1% of the size of the label map updates.

Thus the codebook replenishment scheme described here can be said to adapt a codebook effectively for very little data and processing overhead.

4.8 Conclusions

VQ methods for coding image sequences can be split into those which do not update the contents of the codebook and those which do. This chapter has studied both of these areas and shown that codebook replenishment schemes are necessary for effective image sequence coding. It has also been shown that this can be attained with only a small computational and data overhead.

The author has presented two original label replenishment methods, which are unique in that they identify the areas of largest change before vector quantization rather than after. This gives a saving on the amount of processing that the vector quantizer must do. The most important result from the work on these methods was the unsuitability of a single codebook to code a full image sequence. This result was true even for heavily constrained sequences, and would become even more important for less artificial videotelephony applications.

Another problem with label replenishment which has been identified by the author is the actions needed to initiate a videotelephony link. Foremost amongst these problems is the generation of a codebook, which requires a great deal of processing. This would either take a considerable time to do, or would require a large over-capacity of processing power (compared to the amount needed by the coder itself). The other problems associated with startup are the construction of the image from a blank start, and transmission of the codebook, but these are shown to be less significant than the task of codebook generation.

The aim of codebook replenishment is to ensure that the codebook remains as close to optimal as possible as the sequence progresses. This must be true also when a complete scene change occurs. This scenario would occur when the image source were switched to another camera. For typical videotelephony applications there may be only one camera, but a similar problem could easily occur if the camera position were altered, or the image field was obscured by a foreground object.

The author has developed techniques for codebook replenishment which build on the techniques developed in the modified codebook design algorithm of section 3.5. Redundant codewords are identified through codebook usage statistics, deleted from the codebook, and replaced with more valuable codewords generated by a codeword splitting algorithm. The techniques used are designed to be simple to compute and are analysed for the benefit they offer.

The results show that considerable improvement can be gained with very few codeword updates. This compares with the other published research which does not present any analysis of the codebook update levels required for effective performance.

What the work in this chapter has failed to do is combine the label replenishment and codebook replenishment techniques into one algorithm. The codebook replenishment techniques were developed with full label updates used to isolate the codebook problem for analysis. Unfortunately, time did not permit the combination of the two sections of work.

Chapter 5

Kohonen's Self-Organising Feature Maps for Vector Quantization

The field of study generally known as *Neural Networks* uses models of the structures which make up the human brain to tackle the problems of pattern recognition and classification. The artificial neural network models use non-linear computation elements to represent *neurons*, with neurons having many inputs and a single output. Each input has associated with it a weight, which has the human *synapse* as its physiological equivalent. It is the values of the weights, and the network topology, which control the operation of the network. Weights are generally adjusted during a training period, when the network *learns* the operation it is to perform.

Many different network topologies and training schemes have been proposed, and an excellent review of these is given by Lippmann [116]. The application of artificial neural networks to the coding of images has seen the use of a few different networks [117-120], but the one topology which may have a role to play in the future of VQ is the Kohonen Self-Organising Feature Map (KSOFM). KSOFMs have been used as a method of generating vector quantizers and a number of factors mark them out as being well suited to this task.

The features of KSOFMs which distinguish it as applicable to VQ are :

- A topology which reduces the dimensionality of the input vector space into a smaller number of vectors. In other words, clustering is performed.

- The weight vectors associated with each neuron, which can be directly related to the codewords of a codebook, are structured in an array, and neighbouring weight vectors will be trained to be similar to each other. This may enable the development of reduced complexity search schemes.

- The training algorithms used with KSOFMs give the network an adaptive nature, whereby the weights are being constantly updated with the presentation of each training vector. In this way, adaptive vector quantizers may be a natural result of KSOFMs, and it was identified in the previous chapter that adaptive codebooks are certainly required to code image sequences.

Consideration of these facts led to the decision to study KSOFM for this thesis.

Section 5.1 introduces KSOFMs in more detail and also includes a review of the research work being done in this field. In section 5.2, the KSOFM method will be modelled and its performance evaluated. This analysis will include a qualitative assessment of how KSOFM clustering operates, which enables us to see some of the problems associated with this technique. It also allows us to speculate on some of the possible solutions to those problems encountered. In conclusion, the overall suitability of KSOFMs to VQ are discussed.

5.1 Introduction

Kohonen proposed his Self-Organising Feature Maps [121, 122] as a neural network model which extended those already in use, such as perceptrons [116], by considering the spatial order of the processing units. This factor, he felt, was missing from previous models, and may be important to the action of classification.

Structurally, KSOFM defines a matrix (usually 2-dimensional) containing N output units, or neurons, which is fully connected to a number, K, of input nodes. This means that every input is connected to every neuron in the array, and associated with each connection is a weight. Every neuron is defined by its weight vector, w_j, which is of dimension K. This is illustrated graphically in figure 5–1 which shows a very small network as an example.

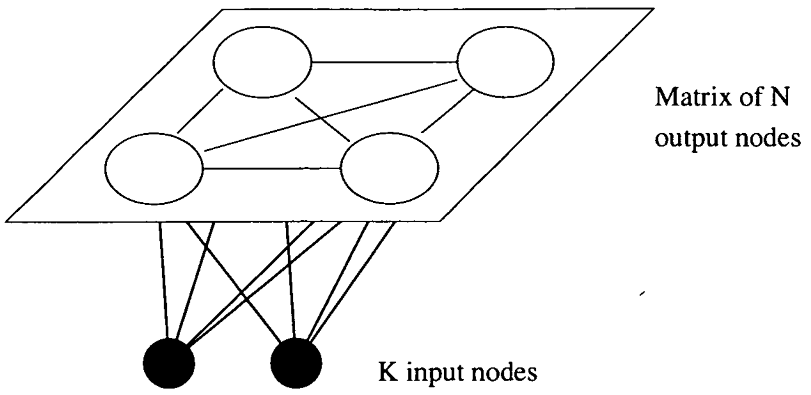


Figure 5–1: Kohonen's Self-Organising Feature Map, showing a 2-dimensional matrix of $N = 4$ output nodes, fully connected to $K = 2$ input nodes.

To train the network, input vectors from a training set are presented sequentially and the weight vectors are adjusted according to the algorithm described below. The weight vectors converge towards cluster centres after sufficient training time. Topological ordering occurs because each input vector adjusts not only one weight vector, but a *neighbourhood* of weight vectors.

The KSOFM belongs to the group of neural networks which is trained by *unsupervised learning*. Here, presentation of each training vector is followed by adjustment to the weight vectors according to the result of the classification that the network has made : the network in effect teaches itself. In contrast, *supervised learning* reinforces the action of a classification by adjusting the weights to reward a correct classification and penalise an incorrect one. This, however, requires that the result of each classification be known beforehand. With VQ, the classification results cannot be known before the quantizer has been trained, so it must train itself.

The KSOFM algorithm is defined thus :

Step 1. Initialise all weight vectors to random values. Set $t = 0$.

Step 2. Apply next input vector, x, from the training set.

Step 3. Calculate the Euclidian distance from x to all output nodes j

$$d_j = \sum_{i=0}^{K-1} (x_i - w_{ij})^2$$

where K is the dimensionality of the input vector.

Step 4. Select the output node with the smallest distance d_i and label it as the winning unit, j^*.

Step 5. Update weight vectors of all nodes in the matrix according to the equation

$$w_{ij} = w_{ij} + \eta(t, \mathcal{D})\alpha(t)(x_i - w_{ij})$$

$\eta(t, \mathcal{D})$ is the neighbourhood gain function, which defines a neighbourhood on the matrix round the winning neuron, decreasing exponentially with distance $\mathcal{D}$ from the winning neuron, and which shrinks over time.
$\alpha(t)$ is the adaption gain function, which decreases exponentially with time, and $0 \leq \alpha(0) \leq 1$.

Step 6. If there are any more vectors in the training set, return to **Step 2**.

Step 7. Increment t. If $t = e$ (where e is the number of *epochs* which the network is to be trained for) then stop. Otherwise, return to **Step 2**.

5.1.1 Published Research

The published research into the use of KSOFMs for VQ concentrates on their performance relative to LBG codebook generation and the advantages that can be gained in the reduction of search complexity due to the topological ordering of the codebook. In general, quoted results show that the performance of KSOFM is very similar to that of LBG. Search complexity reduction can be gained with a small increase in coding error.

McAuliffe *et al* [105] present a comparison of the standard forms of KSOFM and LBG codebook generation and find the performance to be very similar. Nasrabadi and Feng [123] have published similar work, obtaining good results with KSOFM by training for very long periods. Erickson and Thyagarajan [124] present results for KSOFM compared with a K-means nearest neighbour classifier and quote similar performances for both, but with shorter training times for the KSOFM classifier. Giusto and Vernaza [125] use a KSOFM to generate an initial codebook which is then presented to LBG : perhaps here they are tacitly suggesting that LBG is better, as long as it can be given a good initial codebook.

Other researchers recognise that KSOFM is limited and attempt to account for the problems by modifying the training algorithms or network structure. Hsu and Wu [126] use insert and delete functions to modify the matrix of output neurons according to their usage statistics. This has parallels with the techniques developed in section 3.5 for a modified LBG approach. They do not use their techniques to train with image data. Similar techniques are used by Lee and Peterson [127] to create an adaptive codebook which can grow or decay to follow the statistics of the input data.

Krishnamurthy *et al* [128] modify the training algorithm to create what they call Frequency Sensitive Competitive Learning (FSCL). This incorporates a *fairness function* which increases the distortion measure as the usage of the neuron increases. This makes it more difficult for a frequently used neuron to continue *winning* training vectors, resulting in more even utilisation of the neurons. Results show a limited improvement over standard KSOFM for VQ coding of images.

Genetic algorithms (GA) have been identified as an optimisation method that is well suited for large dimensional problems [129]. In this paper, Harp and Samad apply GA to optimise the various network parameters of KSOFM for the best clustering performance. This is an interesting idea but involves huge amounts of processing and memory, as an entire *population* of networks is kept alive, and the GA identifies the best of these and *breeds* from them at the end of every training epoch. For each network in the population, the full training sequence must be presented at each epoch.

Codebooks with the structure generated by KSOFMs can be used to reduce the search requirements during the quantization phase. Troung and Mersereau [130] create a 2-level tree structure from 2-dimensional and 4-dimensional Kohonen networks. Qualitative assessment of the results suggests that they are able to produce similar results to those using LBG. Luttrell presents several papers which concentrate on the same theme [131-135].

One of the other much-vaunted advantages of neural networks is their highly parallel nature which makes them suitable for real-time VLSI implementation. Fang *et al* [66, 136] present simulation results from a device which they claim to be a "trainable analog neural chip for image compression". What the chip architecture actually does is to identify the winning neuron at the presentation of each training vector. All network-level activities are performed digitally by the controlling processor. This architecture, along with others, will be considered in more detail later in section 6.2.

In summary, it can be said that that the published work in this field shows that KSOFMs can be used to vector quantize images, with very similar performance to the standard LBG method. It may offer the benefit of producing codebooks which naturally enable the utilisation of reduced complexity search methods without much further performance loss.

5.2 KSOFM for Codebook Generation

In the light of the work done in this field, this section investigates a standard KSOFM implementation for VQ of image data. This will include an assessment not only of how well it can perform, but also a discussion of the actual mechanism of its operation. This is in an endeavour to find out if there is anything that distinguishes it from other clustering techniques as more suited to this specific problem.

Firstly, the algorithm used will be detailed, followed by a description of the software which was written to test the algorithm. The experimental work presented after this shows the optimisation of the network for the coding of a specific image, and an analysis of the optimal operation is given. These network parameters are then used to train the KSOFM for two other images, and the operation of the network is again analysed to further an assessment of the results that are produced. The processing complexity of the KSOFM algorithm used is then evaluated, and compared with the modified LBG approach of section 3.5.

5.2.1 The Algorithm

The training algorithm outlined above in section 5.1 was used with the definitions of the adaption gain function, $\alpha(t)$, and the neighbourhood gain function, $\eta(t, \mathcal{D})$, given below. Equation 5.1 gives the adaption gain function as :

$$\alpha(t) = \alpha_{init} e^{-\frac{t}{\tau_G}} \tag{5.1}$$

where α_{init} is the initial adaption gain, and τ_G is the adaption gain decay constant. The neighbourhood gain function is most easily expressed if we define the neighbourhood size function, $\nu(t)$, first in equation 5.2 :

$$\nu(t) = \nu_{init} + (\nu_{init} - \nu_{final})e^{-\frac{t}{\tau_N}} \tag{5.2}$$

where ν_{init} and ν_{final} are the initial and final neighbourhood size, and τ_N is the neighbourhood size decay constant. Then we can define the neighbourhood gain function in equation 5.3 as :

$$\eta(t, \mathcal{D}) = e^{-\frac{\mathcal{D}}{\nu(t)}} \tag{5.3}$$

where $\mathcal{D}$ is the Euclidian distance on the neuron matrix from the winning neuron.

The product of the $\alpha(t)$ and $\eta(t, \mathcal{D})$ defines how much influence each training vector has on each neuron. $\alpha(t)$ decreases exponentially with time which gives more influence to early presentations of the training data. $\nu(t)$ describes a neighbourhood size which decreases exponentially with time. $\eta(t, \mathcal{D})$ is always 1 at the winning neuron ($\mathcal{D} = 0$), and decreases exponentially with time and distance from the winning neuron. In this way, neighbourhood effects diminish with time, giving the network more freedom later on in the training period.

The output neurons were arranged in a 2-dimensional matrix of a size which was defined by the user at the start of the training period, and remained fixed throughout the training.

5.2.2 The Software

An interactive SunWindows application was developed to implement the algorithm and provide insight into its operation. Textual input fields and control buttons allowed the user to control the training flow and training parameters. A graphical representation of the network allowed the user to monitor the clustering operation as it happened.

Two dimensions of the weight vectors (selectable from the control panel) were used as x and y coordinates to enable the structure of the network to be visualised, at least in part. Neurons were plotted at these coordinates as small squares and neighbouring neurons were connected by a straight line. This plot of the network was made on top of a graph representing the data in the training set. The same two dimensions of all the vectors in the training set were shown, with lighter blocks showing a higher density of data and black representing no data. An example of the output from this representation is shown in figure 5–2.

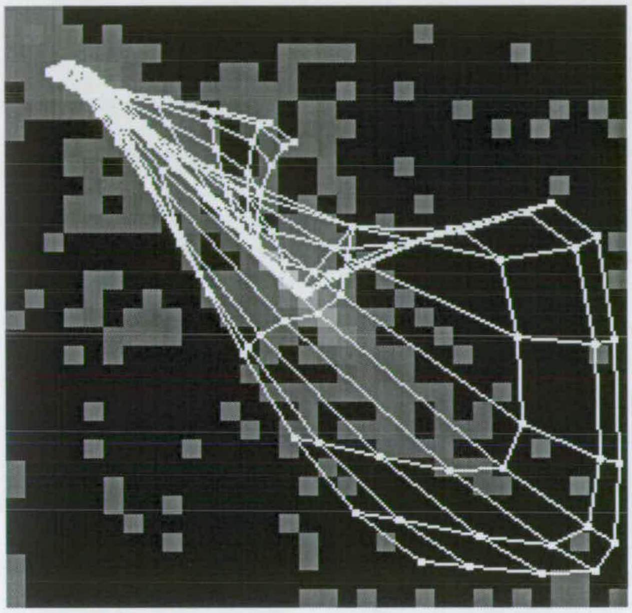


Figure 5–2: An example output from the clustering software showing a network of 16×16 neurons being trained on image data.

The software was written to enable multi-dimensional network topologies to be investigated. In the event, only 2-dimensional topologies were studied, but little work would be required to enable investigations to extend to higher dimensions.

Training sets were generated by dividing an image up into its constituent vectors, with the vector size specified by the user. The pixels of these vectors were converted from their 8-bit bytes into the floating-point numbers required by the KSOFM software and each vector output to a training file. The KSOFM software read in a full training set at start-up.

5.2.3 Results and analysis

The KSOFM sofware was used to create codebooks for the first image in the *Claire* sequence. The performance was tested across a spread of values for the training parameters. The results obtained were analysed to investigate the effect of these parameters on the quality and speed of convergence.

The training parameters selected as optimal for this image were then used again on this image and the clustering process was be analysed in more detail. This includes the presentation of the network and codebook as the training progresses.

The same training parameters were then used to train the network for two other images. These are the first images in the *Miss America* and *Salesman* sequences. Again the networks and codebooks were investigated in an attempt to understand the clustering operation being done by the KSOFM.

If we are to compare the operation of the KSOFM to LBG we must also compare the complexity of the algorithms. This is done in section 5.2.3.4.

5.2.3.1 Training for CLAIRE image

The network was trained to code the *Claire* image with a 16×16 matrix of neurons, each with 9 inputs. The 9 inputs were related to the 9 pixels in a 3×3 image vector. This structure produces a codebook of size $256 \times 3 \times 3$ when the weight vectors are converted into codewords at the end of training. This is a familiar size of codebook for these images which has been used throughout the thesis.

The network was trained with the following spread of training parameters :

$$\begin{aligned} \alpha_{init} &= 0.25,\ 0.50,\ 0.75 \\ \eta_{init} &= 10,\ 20,\ 50,\ 100 \\ \tau_N &= 0.5,\ 1.0,\ 2.0,\ 3.0 \end{aligned}$$

The other training parameters were fixed at the following values :

$$\begin{aligned} \eta_{final} &= 0.001 \\ \tau_G &= 20 \end{aligned}$$

The network was trained for 30 epochs for each of the 48 combinations of parameters. The weight vectors were output at the end of every 5 epochs, and these were converted to codebooks and the *Claire* image was coded so that the progress of the training could be assessed.

Quality of Codebook Generation

The metric used to equate to codebook *quality* here will once again be the PSNR of the coded image, using the codebook generated from the weights at the end of the 30th training epoch. The comparison can be made throughout this section with the result for a codebook generated by the modified LBG algorithm of section 3.5, which codes the *Claire* image with PSNR=36.70dB.

First we can analyse the results with respect to the initial adaption gain, α_{init}. Table 5–1 shows the results for the 3 different values of α_{init} with the final PSNR being averaged over the 16 combinations of neighbourhood parameters.

α_{init}	Average PSNR at epoch 30 (dB)
0.25	36.18
0.50	36.82
0.75	36.89

Table 5–1: Final coding performance against initial adaption gain.

This shows that for higher values of α_{init}, better performance is obtained. The improvement for $\alpha_{init} = 0.75$ over $\alpha_{init} = 0.5$ is very small, but there is a definite drop in performance for $\alpha_{init} = 0.25$. These results are exhibited in the whole sweep of experiments performed. This can be seen in table 5–2 where the average PSNR values at epoch 30 are shown for all values of α_{init} and τ_N.

This table of results also shows us that the size of τ_N has an effect on the quality of the clustering process. Results varied more across this variable than η_{init}, which is the other control we have over the influence of the neighbourhood. τ_N controls the length of time for which neighbourhood effects are significant. Reducing the size of τ_N from 3 to

	Average PSNR at epoch 30 (dB)		
τ_N	$\alpha_{init} = 0.25$	$\alpha_{init} = 0.50$	$\alpha_{init} = 0.75$
0.5	36.10	37.04	36.46
1.0	36.33	36.93	37.46
2.0	36.15	36.57	36.73
3.0	36.16	36.73	36.65

Table 5–2: Final coding performance against initial adaption gain and neighbourhood decay constant.

1 improves the results in all cases. Reducing it further to 0.5 gives a slight improvement for $\alpha_{init} = 0.5$, but degrades the performance in the other two cases.

Overall, we can say that the final clustering results are sensitive to the the initial adaption gain, α_{init}, and the neighbourhood decay constant, τ_N. Results are improved with larger values of α_{init} and smaller values of τ_N. However, the sensitivity to these values is not great, the spread of results in all cases covering less than a 1.5dB range. Results are similar, and in some cases better, than the benchmark modified LBG result.

Speed of convergence

It is also of considerable interest to investigate the speed of convergence as it is imperative that training is done in a short time. The KSOFM algorithm is quite complex and if it is to compete as a realistic alternative to LBG based methods, it must attain good results within only a few training epochs. Section 5.2.3.4 compares the computational requirements of the two methods.

Figure 5–3 shows the speed of convergence which is obtained for all combinations of neighbourhood parameters. In all cases shown the initial adaption gain is set at 0.75 (similar results are shown for 0.50 and 0.25 also). Each graph shows curves for the four values of neighbourhood decay constant, for a constant value of initial neighbourhood size.

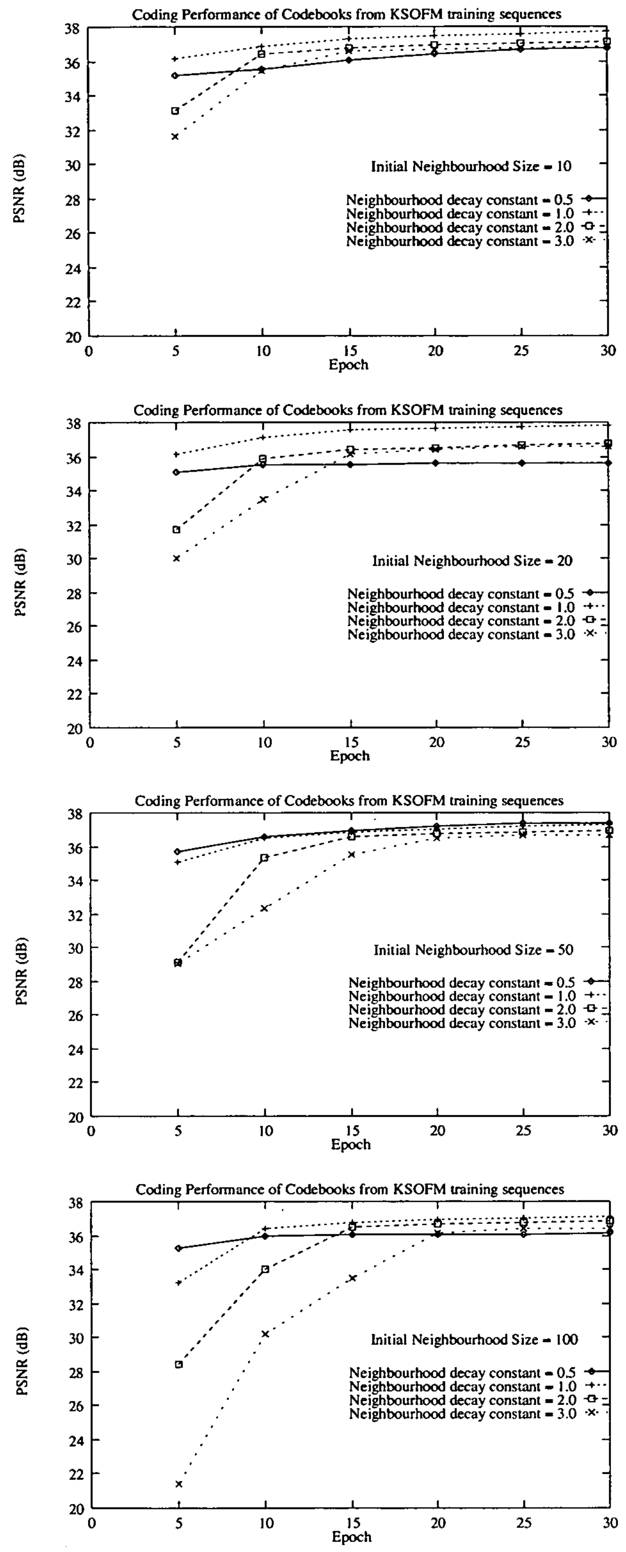


Figure 5–3: Coding performance for *Claire* against training time for all combinations of neighbourhood parameters. In all cases $\alpha_{init} = 0.75$.

We see demonstrated in each graph the same result : that small values of τ_N give the most rapid attainment of good results. When $\tau_N = 0.5$, the performance after 5 epochs is in all cases within 1.5dB of the final result. Indeed, for $\tau_N = 1.0$, similarly good performances are obtained in all cases except when the initial neighbourhood size is at its largest. What we can conclude from these results is that the quicker the influence of the neighbourhood function dies away, the quicker the network is able to converge to its final solution. There will be more discussion of this phenomenon in section 5.2.3.2.

Repeatability

We notice that if the same training set is clustered twice with exactly the same training parameters, we obtain coding results that are different. This is due to the combination of the network being started by randomising all of the weight vectors, and the sensitivity of the training scheme to the order of presentation of the training set. This phenomenon was discussed by Hsu and Wu [126]. What can actually be seen in these experiments is that the KSOFM can be orientated differently on the training map. Figure 5–4 shows two codebooks generated after 10 training epochs for two separate experiments with exactly the same training sequences and training parameters. The codebooks are seen to have largely similar content, although the organisation is certainly different.

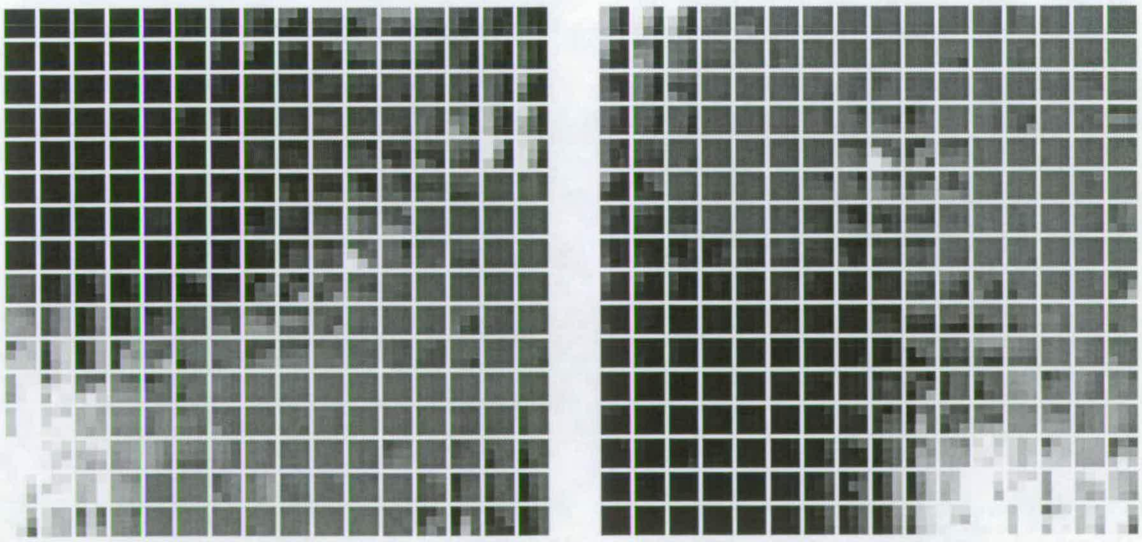


Figure 5–4: Two codebooks generated using exactly the same training parameters.

It was seen through experimentation with the KSOFM software that certain combinations of training parameters caused the results to be less repeatable than others. Generally, the results for short times of neighbourhood influence (low values of τ_N) were more susceptible to these problems. This is a phenomenon which warrants more research into its causes and possible solutions.

5.2.3.2 Analysis of KSOFM Operation

The results from the previous section give us some ideas about how the clustering is being done with KSOFM. A fuller investigation into a single training example was then required to understand the operation more fully. In this section, a set of training parameters which were shown to produce good results in the previous section, was used to train the network again and provide the basis for a more rigorous examination.

This choice of training parameters was :

$$\begin{aligned}\alpha_{init} &= 0.75\\ \eta_{init} &= 20\\ \tau_N &= 1.0\\ \eta_{final} &= 0.001\\ \tau_G &= 20\end{aligned}$$

The choice of $\tau_N = 1.0$ was shown to be desirable for rapid and good training, without being too small to impact seriously on the repeatability of the results. A high initial adaption gain value was also shown to be best for good training.

This time the training was done for 10 epochs only, and the weight vectors were exported at the end of each iteration. Figure 5–5 shows the PSNR results for coding the *Claire* image with the codebooks from each of these weight sets.

This graph shows that the vast majority of the training has been done within 5 epochs. The coding performance after 5 epochs is only 0.75dB down on the result after 10 epochs. To obtain a better understanding of the training process it helps us to look at the KSOFM network representations as the training progresses. These are shown in

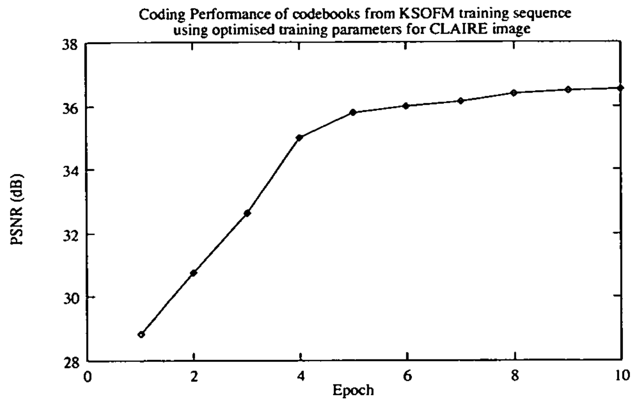


Figure 5–5: Coding performance for *Claire* against training time using optimised training parameters.

Appendix B with the codebook at the end of each epoch shown alongside the KSOFM network representation.

We can see that for the first two epochs, the neighbourhood effects have a strong influence on the development of the codebook. Here, neighbouring codes are very similar, and the codebook is adapting to spread the network over the full training set. From epoch 3 onwards, the network representations show that the structure of the network is being lost, and the codebooks show that this freedom has enabled the development of much more individual codewords. We can add to this analysis by looking at the values of the neighbourhood gain function for the immediate neighbourhood of the winning neuron, for the current training parameters. Figure 5–6 shows a small section of a 2-dimensional KSOFM network, and table 5–3 shows the values of $\eta(t, \mathcal{D})$ for the first 3 epochs.

This shows that from the third epoch, training set vectors will only have an effect on the winning neuron. This concurs with the results from the graphical KSOFM network representations which show the network losing its structural constraints on the third epoch.

From this analysis, and the fact that larger values of τ_N were shown earlier to slow down the training process, we can conclude that for an efficient clustering performance, the neighbourhood constraints inherent to KSOFM must be released. Once this has been

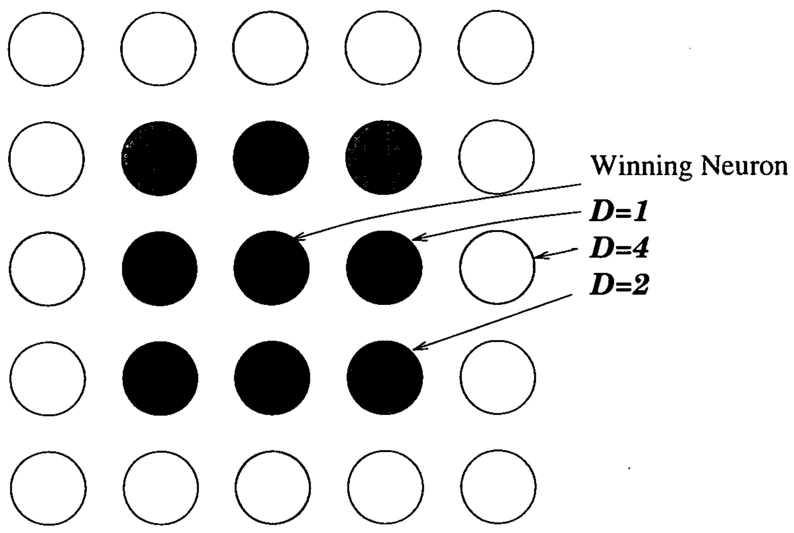


Figure 5–6: The immediate neighbourhood of the winning neuron on a 2-dimensional KSOFM, showing the winning neuron and distances, $\mathcal{D}$, to neighbouring neurons.

done, the weights are free to differentiate themselves much further, a step which is vital if an effective codebook is to be formed.

5.2.3.3 Tests with other images

The same network parameters identified above as being effective to train the KSOFM for the *Claire* image were then used to train a network for each of the *Miss America* and *Salesman* images. Figure 5–7 shows the training results for these two images over the first 10 epochs.

	$\eta(t,\mathcal{D})$		
t	$\mathcal{D}=1$	$\mathcal{D}=2$	$\mathcal{D}=4$
1	0.87	0.76	0.58
2	0.69	0.47	0.23
3	0.00	0.00	0.00

Table 5–3: Neighbourhood gain function values in the immediate neighbourhood of the winning neuron for the first 3 training epochs.

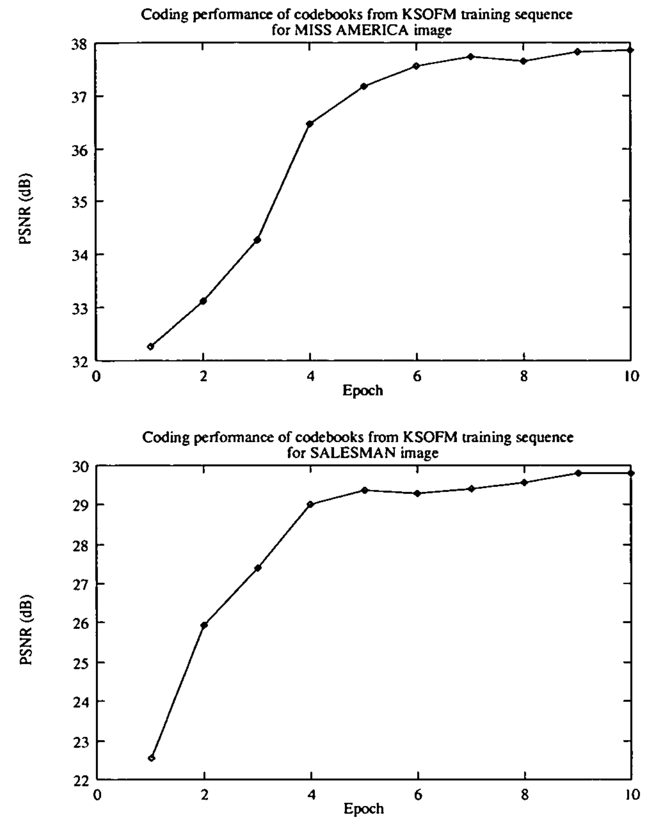


Figure 5–7: KSOFM training results for the *Miss America* and *Salesman* images.

As for the *Claire* image, the KSOFM has trained very rapidly, obtaining results after 5 or 6 epochs which are very close to the final results. For the *Claire* sequence, it was found that KSOFM could produce codebooks which matched or even bettered the performance obtained with the modified LBG codebook generator. However, these results could not be matched for the *Miss America* and *Salesman* images. The results presented above underperform the LBG based technique by more than 1dB for both images, and this numerical difference is clearly visible in a qualitative assessment of the coded images. Figure 5–8 shows the *Miss America* image coded with a codebook from the modified LBG generator and from the KSOFM codebook generator. On qualitative assessment, both the *Miss America* and *Salesman* images have been even more poorly coded than the numerical result suggests.

These results cannot be explained away as a freak result : the training was performed a number of times and consistently produced poor codebooks. The ability to view the KSOFM network and the codebooks produced gives us the insight into the reasons why

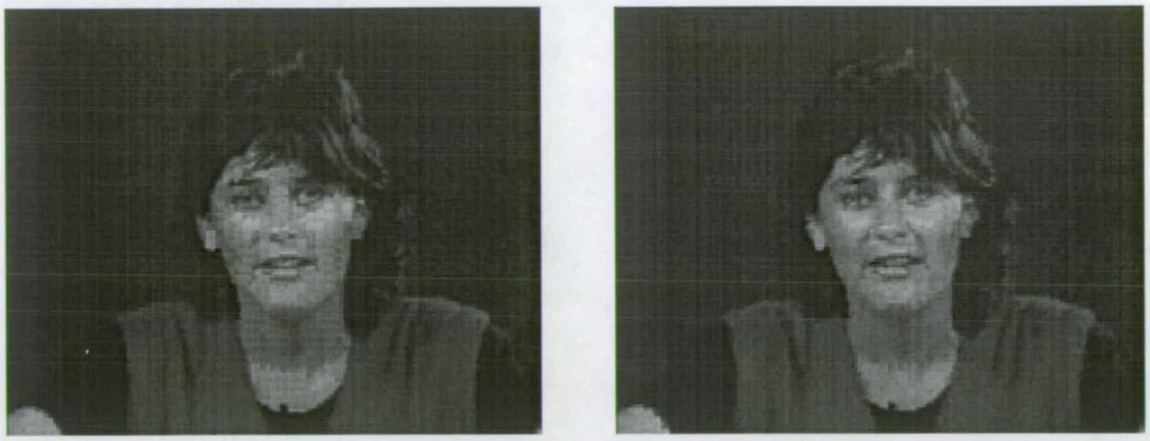
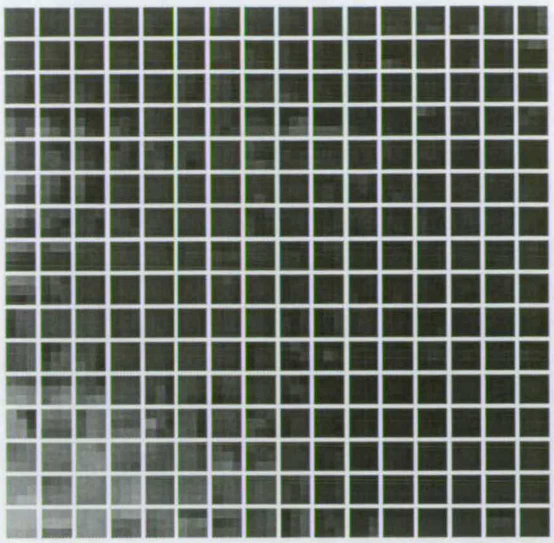


Figure 5–8: *Miss America* image coded using codebooks from the KSOFM method and the modified LBG method.

the coding has been poorly done in these cases. In figure 5–9 the codebook from the tenth training epoch of the KSOFM training sequence for the *Miss America* sequence is shown alongside the one generated for the same image with the modified LBG algorithm. It can clearly be seen that there are too few bright vectors in the KSOFM codebook. This fact is also clear when the images are examined carefully : the face area, which is the bright area of the image, is particularly badly coded.

This result is due to the behaviour of the KSOFM training algorithm when presented with training sets which contain large numbers of very similar input vectors. The number of neurons which are used to code these training vectors is very large because of the neighbourhood effects pulling many neurons towards the same area. After the release of the neighbourhood constraints, the few neurons which exist outside the concentrated cluster are used to code all the other training vectors. Figure 5–10 shows the KSOFM network for the *Miss America* training sequence after 2 and 10 epochs. It can be seen that the bulk of the neurons are clustered in one small area after 2 epochs, and that most of these remain close to this area as the training progresses, leaving the remainder of the network to cover a large area of the vector space.

With this understanding of the operation of the KSOFM process, it would be possible to develop strategies for overcoming this problem which is inherent to the sort of training sets that come from image data. One obvious method would be to pre-process the training


KSOFM codebook

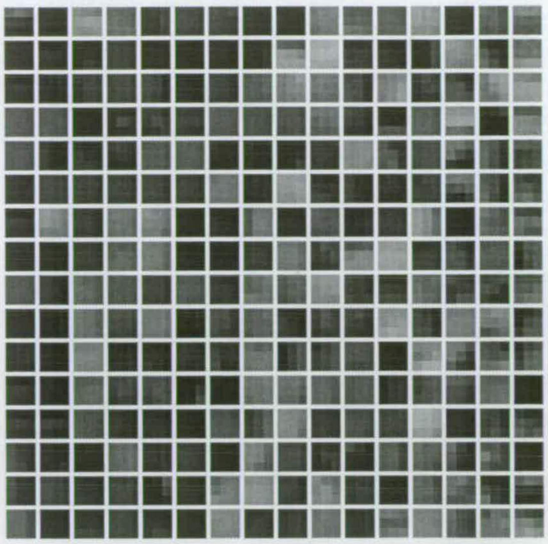

Modified LBG codebook

Figure 5–9: Codebooks generated for the *Miss America* image using the KSOFM method and the modified LBG method.

sets to remove the large number of very similar training vectors. The codebooks which have been generated here suggest that any improvements to the algorithm would be aimed at encouraging a more diverse population of codewords in the codebook, in much the same way as the methods developed to improve LBG codebook design. However, time did not permit any further development of the KSOFM algorithm in this thesis.

5.2.3.4 Processing complexity

From the experiments performed in the course of this chapter it was clear that the KSOFM algorithm as implemented was a very compute-intensive process. Since one of the aims of this thesis is to search for low-complexity methods it was of interest to compare this method to the modified LBG codebook generation method developed by the author in section 3.5.

The Unix **time** command was used to get an estimate of the complexity of the algorithms. Table 5–4 shows the timings for codebook generation for the *Claire*, *Miss America* and *Salesman* images. The timings shown include all the overheads associated with these processes, such as the reading in of training data from disk and the writing

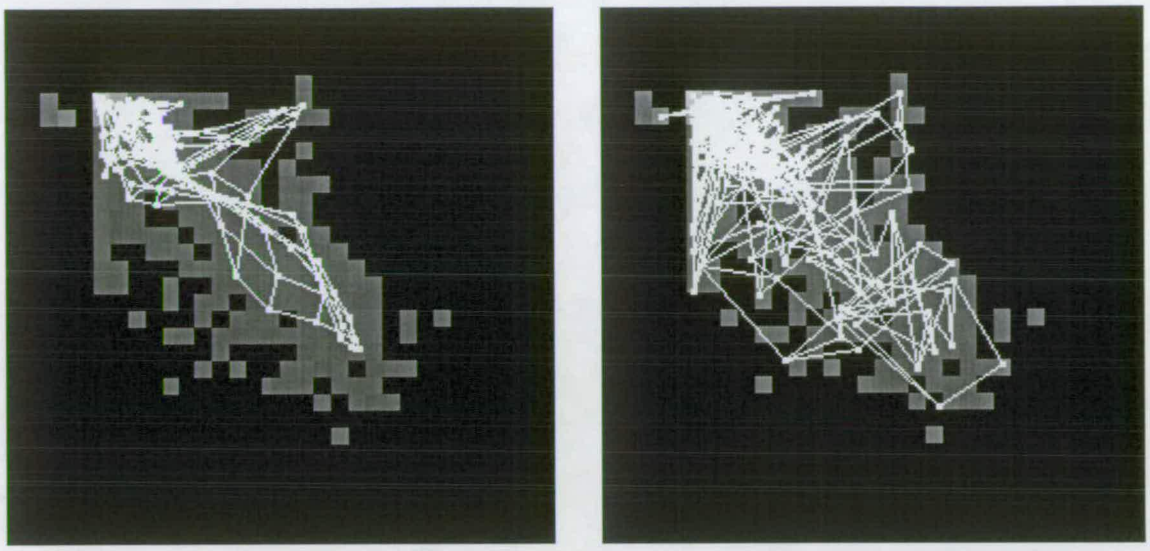


Figure 5–10: KSOFM network representation for training on the *Miss America* image at the end of epochs 2 and 10.

out of results. However, in both cases the bulk of the CPU time used would be for the core algorithm. The time for the KSOFM software was not distorted by any overhead for the graphical interface : the software was written so that the display could be turned off and the training controlled purely from command line arguments. Timings for the modified LBG algorithm represent those taken to iterate to a final solution. This takes a variable number of iterations, or presentations of the training data set, and this number is also shown in the table. For the KSOFM algorithm, the training parameters were set to be equal to those used in section 5.2.3.2. The training period was set to 5 epochs, which was shown to be sufficient to reach a solution close to the final level attained by the KSOFM process.

The results show that the KSOFM takes approximately 7 to 11 times longer than the modified LBG method. It is recognised that the implemented algorithm could be modified, particularly to include a much simpler neighbourhood function. However, it is unlikely that sufficient improvement could be made to bring the complexity down to the same level as for the modified LBG.

As with modified LBG, the closest neighbour to each training vector must be found, and this is the core of the processing that is done. With KSOFM, this calculation

	KSOFM	LBG	
Image	time(s)	time(s)	iterations
Claire	248.2	22.4	7
Miss America	247.4	27.0	8
Salesman	247.2	37.0	9

Table 5–4: CPU timings for codebook generation.

is done in floating-point arithmetic, whereas modified LBG uses integer arithmetic. This, though, is not a reason for the increased CPU times, as the Sun IPX computer used for these tests actually performed floating-point operations faster than integer operations. The reason, therefore, is due to the number of operations rather than the type of operations. With KSOFM, each presentation of a training vector results in the movement of the entire codebook. With modified LBG, the codewords are only moved once at the end of each presentation of the entire training set.

The results presented above show that in computational terms, KSOFM is much less efficient than modified LBG.

5.3 Codebook generation techniques applied to a face recognition system

The codebook generation techniques investigated in this thesis have been evaluated for their ability to produce codebooks for the vector quantization of images. There are other applications where codebook generation methods are required, where some form of clustering technique is necessary to reduce a data set. One such application, a face recognition system, was studied by the author in collaboration with other members of the Integrated Systems Group at Edinburgh University. The reader is referred to a paper published on this work, which is included in Appendix C of this thesis.

The face recognition developed by Sutherland operates on seven facial features which are extracted from an image of a face. These features are each eye, the nostrils, the bridge of the nose, the mouth, the chin, and the hair. In addition, a sub-sampled view of the whole face is treated as an eighth feature. For each of these eight features, a function analogous to a police *photofit* system is performed whereby the feature is replaced by the closest match from a codebook of possible features. A vector quantizer is used to find the nearest neighbour from the codebook.

The system is first trained to recognize a face by extracting the features from a number of training images and coding these features. The system then knows which feature codewords are commonly used to represent that face. A face image will be recognised correctly if the extracted features are coded with the same feature codewords as were used during training.

The feature codebooks in the original system were generated by extracting the features from one image of each person in the test population. There were forty members in the test population, so each of the feature codebooks had forty codewords. Each of the features had dimensions of 28 $\times$ 20 pixels (formed by sub-sampling in the case of the face, chin and hair features). Clearly, though, it is not possible to add another codeword to each codebook every time another person is added to the system. Clustering techniques were required to reduce the feature codebook sizes to maintain a system with realisable memory and processing requirements.

In the experiment presented in the included paper, the feature codebooks were reduced from forty vectors to twenty vectors in size. This reduction was done using the modified LBG algorithm and the KSOFM algorithm, both of which have been developed by the author in this thesis. The software was adapted to train on the very much larger vectors used in this application.

The results show that for this application the modified LBG algorithm performs much better than the KSOFM method. Recognition rates for the twenty vector codebooks generated from the modified LBG method were close to those attained for the full 40 vector codebooks. The KSOFM generated codebooks gave a markedly worse performance. It was proposed that this was due to the KSOFM method generating codebooks which failed to maintain a good spread of the vector space. The modified LBG algorithm, however, has been shown to be an effective method in an application very different to the one for which it was originally developed.

5.4 Conclusions

The aim of the research presented in this chapter was to gain an understanding of the operation of KSOFMs, and to ascertain whether they may provide the basis for improved VQ methods. Neural network publications suggest that they can offer comparable performance to existing methods, and that their structure offered inherent advantages due to the topological ordering of the codebook which occurs. Another suggested advantage was their "adaptive" nature, which may suit KSOFMs for interframe coding methods. A quantitative and qualitative assessment of KSOFM was given to investigate these claims.

The KSOFM algorithm implemented here was shown to produce codebooks which could equal or better those produced by the modified LBG algorithm in certain cases. In other cases, however, KSOFM was shown to produce very poor codebooks. These results showed problems related to those of LBG, where sufficient diversity was not maintained in the codebook. Training sets with large numbers of similar vectors resulted in codebooks with too many codewords dedicated to coding this part of the training set. It is recognised that an improved KSOFM training algorithm could be developed to circumvent these problems, but that development has not been done in this thesis.

A fundamental problem remains with KSOFM : it is considerably more compute-intensive than LBG-based methods. Analysis of compute times showed that the implemented KSOFM algorithm took between 7 and 11 times longer to converge close to its final performance than the modified LBG method. While the KSOFM algorithm could have been simplified, it is still true that KSOFM training does include many more processing steps. This factor was the main reason why KSOFM methods were not studied any further in this thesis.

The other potential gains offered by KSOFM were the reduction in search complexity and the adaptive nature of the training process. Codebooks generated using this method did indeed display considerable structure and it is easy to see how they would enable reduced search schemes to be used. However, it was seen that the release of the neighbourhood constraints, which created this structure, was vital for the production of effective codebooks. Improvements to the KSOFM algorithm for improved coding performance would almost certainly encourage further diversification of the codewords, reducing the codebook structure further.

As to the adaptive nature of KSOFM, it is no more adaptive than any LBG based method. All that is being done is that the codebook, in this case represented by the weight vectors, is iterating towards the best clustering performance it can provide. In an interframe coding scheme, any codebook updates would have to be sent to the receiver just as with any other codebook replenishment method. There is no reason to suggest that better codebook replenishment decisions could be made with a KSOFM based technique than an LBG based technique.

Chapter 6

Suggestions for further work and concluding remarks

This chapter will summarise the work reported in this thesis and attempt to evaluate the achievements with respect to the thesis aims and existing work in this field. From this evaluation a number of suggestions for future work will be proposed.

6.1 Thesis Summary

Codebook Generation

To code an image effectively with VQ it is vital that a good codebook is available. Codebooks are generated, ideally, by training on the image data that is to be coded. In this situation, an efficient codebook generation scheme should be able to create a close to optimal codebook.

The most common method for codebook design is the Linde-Buzo-Gray, or LBG, algorithm. This technique uses a normal vector quantizer to code the training data, and improves the codebook by moving it at the end of each presentation of the training set, to iterate towards an effective solution. Experimentation showed that this algorithm is sensitive to the initial codebook supplied to it, and to the training data. In cases where the training data included large areas of near-constant illumination, such as the videotelephony images used, the resultant codebooks contained a poor balance of codewords. In these cases there were too many codewords dedicated to the coding of the background of the image scene.

A novel codebook generation scheme was developed which ensured that an effective spread of codewords is obtained. The algorithm is a modified LBG technique which uses intelligent techniques to identify redundant codewords for deletion and create useful replacements to fill the resulting gaps. Redundant codewords are identified as those which are poorly utilised and are very similar to other, better used, members of the codebook. Replacement codewords are generated by a splitting algorithm which takes one valuable codeword and created two close copies of it.

This algorithm was demonstrated to produce improved codebooks for the images which had caused problems for the LBG algorithm. The numerical improvement in the examples chosen was more than 1dB for codewords of size 2×2 and 3×3 pixels. The qualitative improvement in the coded images was even more marked than the quantitative improvement, with facial detail and high contrast edges, to which the human eye is very sensitive, showing particular improvement.

Interframe Coding

The coding of image sequences relies upon the high correlation between successive frames, which uses the greater redundancy in the data to generate high compression ratios. This compression is effected by coding only those areas of the image which change between frames. With VQ, this sort of scheme is called label replenishment, where only those labels which have changed are sent to the receiver.

Two original forms of label replenishment are presented by the author. Rather than coding the full image and then identifying those labels which have changed, these algorithms first identify the areas of the image which have changed the most, then code these areas for transmission. The benefit here is the reduction in the computation load because the vector quantizer need only code the fraction of the image which will be transmitted.

For the image sequences under analysis, which had a rate of 10 frames per second, acceptable quality coded sequences were generated for a bit rate of 0.24 bits per pixel. This bit rate equated to the case where 400 of the total of 2832 vectors in the image were updated at each frame. At this update rate the worst of the effects of limiting the update rate were eliminated.

The limitations of label replenishment schemes were clearly demonstrated with the experiments performed here. In the case where all labels were updated at each frame, the effectiveness of the codebook was seen to diminish with the progression through the sequence. This was true even for heavily constrained videotelephony sequences where the image content changed very little. The need for techniques which adapt the codebook over time was obvious.

Codebook replenishment schemes attempt to maintain an effective codebook by updating the codebook at each frame of the image sequence. Published techniques update a number of vectors by moving them to their cluster centroids, deleting redundant codewords and generating new codewords to fill the gaps. These operations are analogous to those developed for the modified LBG codebook generation algorithm. What is missing from the published work is an analysis of the value of the different types of codeword updates and an estimate of the update levels required to gain an adequate performance enhancement.

Techniques for codeword replenishment were developed from those used in the modified LBG algorithm. Redundant codewords were identified as those which have been unused for the longest time and which have little dynamic range. These codewords were deleted and replaced with another which is spawned from a codeword which has been used many times and which has a large cluster volume. The replacement was generated by creating a copy and then perturbing each pixel by a small random value, leaving the original unmoved. A splitting algorithm like the one in the modified LBG algorithm, where the original was moved, was found to be destructive to the codebook. Also, a number of codewords were centred at each frame. The codewords chosen for centring are those with the largest cluster distortions. These identification and codeword movement techniques were simple to implement and require very little computation relatived to the amount performed by the vector quantizer.

It was shown that significant improvement can be gained with a very small number of codebook updates. Rapid adaption of a codebook which is initially very poor, which would occur at a scene change, was obtained. The delete/replace operation was shown to be vital for rapid adaption from an initially poor codebook. The optimal number of these operations was shown to be 1, as little gain is to be had by increasing this number when the codebook is poor, and the action is actually destructive when the codebook is already good.

Increasing the number of codewords to centre at each frame was shown to always give a coding improvement. However, the relative value of the codewords centred is very significant. The benefit gained from the first two codewords centred was much greater than that from further centring operations, especially in the cases where the original codebook is poor.

The techniques developed here showed that codebook adaption, which is necessary for effective image sequence coding, can be attained through simple techniques and with very low update rates. For the example configuration used in the experiments presented in this thesis, useful codebook adaption was achieved through one delete/replace and two centring operations per frame. The data overhead for this update level is less than 1% of the data for a full label map update.

KSOFMs for VQ

In the light of the requirement for adaptive codebook schemes, and the claims made in the neural network literature, the decision was made to study Kohonen's Self-Organising Feature Maps (KSOFMs) as a potential structure on which to base VQ methods. The KSOFM is a clustering technique which can be used to generate codebooks which are distinguished by the fact that they are topologically ordered : in other words, neighbouring vectors in the codebook will be similar. This result, caused by the imposition of a structure on the codebook, may enable the use of tree-search methods.

Software was developed to implement the algorithm and provide a graphical representation of the network. This software not only provided an assessment of how well the algorithm was able to perform, but also offered insight into the mechanism by which it was operating.

The KSOFM algorithm presented here was shown to produce codebooks which were of a similar quality to those produced by the modified LBG algorithm. For the training image initially studied an improvement was obtained, but training on other images indicated a sensitivity to the training set data. When there were a large number of similar vectors in the training set, the codebook generated by the KSOFM contained many vectors for coding these training vectors.

The clustering operation was investigated using the network display software. What was discovered was that the best, and fastest, training performance was obtained when the structure constraining the codebook was released as early as possible. This freedom allowed the diversification of codewords, a process which was seen as vital for the production of effective codebooks. For the KSOFM technique to be truly effective, it would be necessary to modify the training process to encourage further diversification of the codebook. Reduced search methods would clearly become less effective as a result of any such modification of the training algorithm which would cause more sub-optimal decisions to be made.

The problem which discouraged any development of improved KSOFM based methods was the high compute intensity of the training process. Analysis of compute times showed that the KSOFM algorithm implemented took between 7 and 11 times longer to complete than the modified LBG algorithm.

Furthermore, the KSOFM algorithm showed no greater suitablity to an adaptive VQ coding technique than the modified LBG technique. If one is to analyse what action is being performed, it is simply a clustering technique with a different training algorithm. Also, the nearest neighbour calculation used, far from being performed by a "neural computation unit", is simply the vector difference calculation used in the LBG and other clustering methods.

6.2 Suggestions for further work

The original aim of this thesis was the identification of image compression algorithms for VLSI implementation. The work presented here identifies VQ as suitable for this task and progresses to the development of improved techniques for codebook generation and codebook adaption. These techniques have been developed throughout with a requirement that they be as simple as possible whilst remaining effective. However, an actual implementation of these algorithms in hardware has not been realised or indeed studied.

Recommendations for future work can be split into two areas : continuing work on algorithms and VLSI realisations for VQ. These areas are considered separately below.

6.2.1 Algorithms

Algorithmic developments which are suggested directly from the work in this thesis are :

Combination of codebook replenishment and label replenishment techniques. These two facets of interframe coding were considered separately in this thesis. An alliance of the two could provide a complete interframe coding technique which combines high compression ratios with simplicity and high coding performance.

Simplification of codebook replenishment techniques. The codebook replenishment methods presented ranked the entire codebook according to certain measures to identify the codewords to be operated upon. Analysis later showed that only a small number of updates was required for effective adaption, and this could be exploited to simplify the above processes.

Modified LBG codebook design. The presented work on this topic isolated the need for improvement over the standard LBG method, and proposed techniques for doing so. In the light of the work done later on codebook replenishment, it is believed that further improvements could be made. In particular, the splitting algorithm used in the modified LBG method was later seen to have a destructive effect on the codebook replenishment process. An alternative method for the generation of replacement codewords was developed and something similar to this technique would probably improve the modified LBG codebook generator.

6.2.2 Hardware Implementation

The techniques developed in this thesis have demonstrated that effective image compression schemes are possible with simple modifications of the standard VQ process. Of particular interest, codebook replenishment techniques are shown to be possible through simple analysis of the codebook usage statistics for VQ coding of an image sequence. With these algorithms, the bulk of the required processing is that done by the vector quantizer itself to find the codeword label for each vector presented to it. If we look again at this method, we can discuss possible hardware realisations for VQ coding methods. A codebook replenishment coder is shown in figure 6–1.

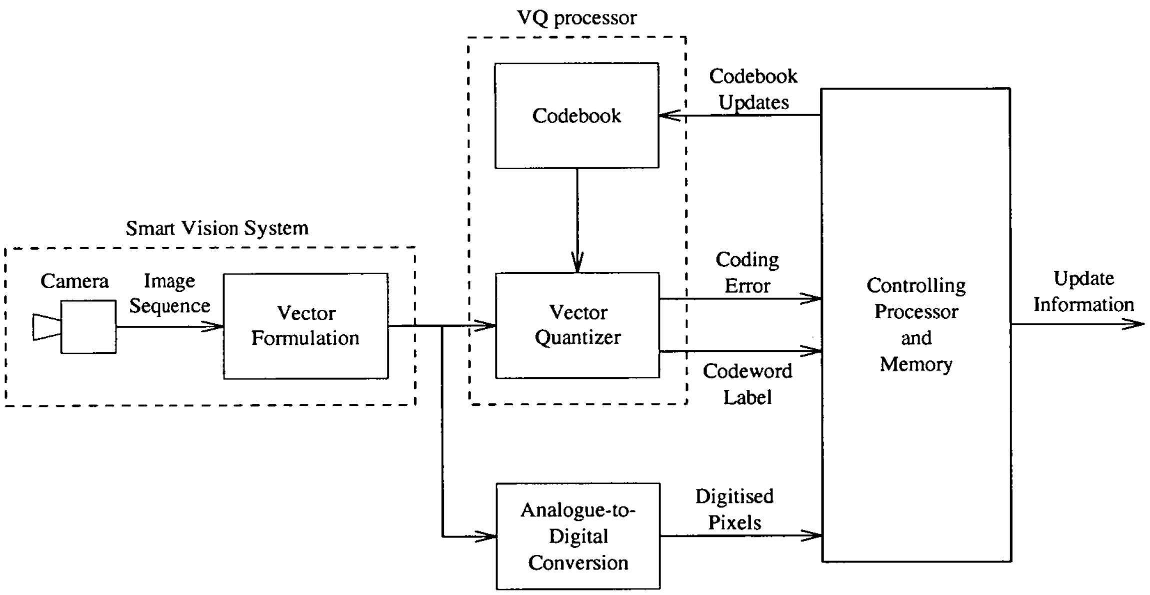


Figure 6–1: Possible hardware realisation for a codebook replenishment VQ.

This diagram is drawn in a way which shows how the design of a hardware implementation of this method could be approached. The VQ processor block operates on one vector of the image at a time, finding the closest match from the codebook, and must also output the coding error for this match. The implementation shown uses a

VQ processor with analogue vector inputs. A controlling processor collates the coding statistics and can be programmed to implement a codebook replenishment algorithm, perhaps combined with conditional label replenishment. Update information is generated to transmit to the receiver and update the local codebook. This realisation suggests that future study of the VLSI implementations of VQ codecs could look at the creation of smart vision systems and at VQ processors.

Smart vision systems

The Integrated Systems Group at the University of Edinburgh provides an excellent opportunity for interesting research in this field with their ASIS technology, which provides the basis for the development of *smart vision systems*. ASIS technology allows digital processing logic to be combined with an imaging array on a single substrate, so that low-level image processing functions can be performed on image data on a single chip [137].

While it would be desirable to create a smart vision system which performed the entire image compression process, generating compressed image data on the same device as the imaging array, this task is estimated to be well beyond the bounds of current technology. The added value which could be created with smart vision systems would be the integration of the vector formulation process with the imaging array. This would avoid the need to digitise, store, and then address the entire image. Related work was performed by Anderson [137] where image filtering was performed by acting on 3×3 pixel blocks of the array at a time.

VQ processors

Figure 6–2 shows the possible VLSI implementations of VQ processors.

Digital solutions split into 2 types : brute force application of high performance DSPs, and custom designs. Enormously high processing power is now available from DSP devices such as the Texas Instruments Multimedia Video Processor [138] and Adaptive Solutions' CNAPS chip [139]. These devices, and other similar designs [64,

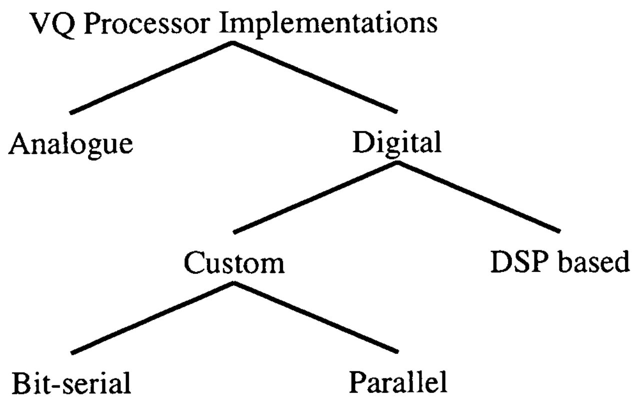


Figure 6–2: VQ processor realisations.

140-142], use multiple processors operating in parallel to deliver sufficient processing power for imaging applications. These devices offer the advantage that they are flexible and can be reprogrammed to perform a different task. However, the function of a VQ processor is clearly defined, and the penalty for this flexibilty is the enormous cost of these DSPs.

Dedicated digital solutions can be bit-serial or parallel implementations. Bit-serial processors use systolic arrays of processing units to pipeline the calculation of a vector difference. A number of different architectures have been proposed [54-64]. Recently, Israelsen [65] has produced results for a parallel implementation of a VQ processor which is capable of finding the best match of 16 codevectors in 38 nanoseconds. A number of processors can be cascaded in parallel to search larger codebooks.

A very interesting avenue of research would be the investigation of the viability of analogue VQ processors. Analogue processors could offer considerable benefits in terms of size and power consumption over digital solutions [143]. The building blocks needed are already available : what must be studied is the ability of these blocks to be combined and deliver sufficient reliability and accuracy. These building blocks are :

Vector difference formulation. Assuming that an analoque representation of the codebook is available, a vector difference could be formed by multiplying the difference between input and stored voltages for each pixel, and summing these results. Standard designs are already available for four-quadrant analogue multipliers [144].

Winner-take-all circuit. This block must find the largest of all the vector difference calculations, and encode the result. Fang *et al* [66, 136] have demonstrated a circuit designed for this task.

Analogue storage for codebook. The performance of this block is currently limited by the technology available. Today's solution would use capacitors to store voltages, and charge leakage dictates that refresh circuitry would need to be employed to update the codebook representation. Emerging technologies for analogue storage promise exciting opportunities for applications such as VQ processors. These new devices are based on floating gate (EEPROM), ferroelectric and amorphous silicon technologies [145, 146].

The feasibilty of analogue VQ processors would be investigated by modelling the behaviour of each of the blocks above and combining these models with a description of the whole processor. The design problem this processor poses is an extremely challenging one, particularly because its die size would be very large and this would pose all sorts of problems including those of matching and temperature effects. The design need not be constrained to ensuring the best match is always found, as sub-optimal search methods have been used before.

6.3 Concluding remarks

Communication by video will become a common occurrence within the next few years. Image compression methods facilitate this development by reducing the bandwidth and storage costs for image sequence data. International standards have recently been agreed for compression algorithms for videotelephony and image sequence storage, driving sales volumes up and prices down. There is now considerable momentum behind the use of these standards, and any alternative must offer some unique benefit to encourage its adoption for widespread use.

Transform coding methods provide the basis for the existing compression standards. These are mature techniques and it is widely acknowledged that they are neither simple to implement nor offer the best compression performance available. Vector Quantization has been studied in this thesis as it has been recognised that it is a method which possesses a structure suitable for VLSI implementation and could offer improved compression performance. This thesis has shown that VQ codecs can be kept very simple and still achieve excellent results.

Considerable investment has been made by industrial concerns in the development of the compression standards. It is difficult to see how a VQ based system, along the lines of those studied in this thesis, could compete in the markets for which these standards are designed. The adoption of new standards for these applications will only be made when new techniques demonstrate markedly improved performance above that obtained with VQ.

However, the study of alternative image compression methods must not be slowed by the dominance of the standard methods. Standard techniques are not always desirable or necessary. In certain applications, alternative methods will offer advantages which will favour their use. Low complexity VQ codecs using techniques developed in this thesis could be one such alternative. The development of analogue VQ processors could maximise the benefits of simplicity and low power, and I believe that this problem in particular warrants further study.

References

[1] M L Liou. 'Visual Telephony as an ISDN Application'. *IEEE Communs. Mag.*, 28(2):30–38, February 1990.

[2] A Netravali and J O Limb. 'Picture Coding : a Review'. *Proceedings of the IEEE*, 68(3):366–406, March 1980.

[3] L Wang and M Goldberg. 'Progressive Image Transmission Using Vector Quantization on Image in Pyramid Form'. *IEEE Trans. on Comms.*, COM-37(12):1339–1349, December 1989.

[4] A K Jain. 'Image Data Compression : a Review'. *Proceedings of the IEEE*, 69(3):349–389, March 1981.

[5] L R Yencharis. 'Video Compression Chips Now : More Hope or Less Hype?'. *Advanced Imaging*, 7(3):62–65, March 1992.

[6] R C Gonzalez and P Wintz. *Digital Image Processing*, pages 255–330. Addison-Wesley, Reading, MA, second edition, 1987.

[7] P J Burt and E H Adelson. 'The Laplacian Pyramid as a compact image code'. *IEEE Trans. on Comms.*, COM-31(4):532–540, April 1983.

[8] S-C Pei and I-I Yang. 'Hybrid Pyramid image coding and Data Compression'. In *Proc. Int. Symp. on Circuits and Systems (ISCAS-89)*, pages 1358–1361. IEEE, May 1989.

[9] M Ohta and S Nogaki. 'Hybrid Picture Coding with Wavelet Transform and Overlapped Motion-Compensated Interframe Prediction'. *IEEE Trans. on Signal Proc.*, SP-41(12):3416–3423, December 1993.

[10] A S Lewis and G Knowles. 'Image Compression using the 2-D Wavelet Transform'. *IEEE Trans. on Image Processing*, 1(2):244–250, April 1992.

[11] M Antonini, M Barlaud, P Mathieu, and I Daubechies. 'Image Coding using Wavelet Transform'. *IEEE Trans. on Image Processing*, 1(2):205–218, April 1992.

[12] M T Orchard and K Ramchandran. 'An Investigation of Wavelet Based Image Coding using an Entropy-Constrained Quantization Framework'. In *Proc. Data Compression Conf. (DCC-92)*, pages 341–350, March 1992.

[13] S Yao and R J Clarke. 'Motion-Compensated Wavelet Coding using Adaptive Vector Quantization'. In *Proc. IEE Colloquium on Applications of Wavelet Transforms in Image Processing*, pages 2/1–2/4. IEE, Jan 1993. Digest No : 1993/009.

[14] D R E Timson, B L Combridge, and M L Childerhouse. 'Interframe Laplacian Image Coding for Video Compression'. In *Proc. Int. Conf. on Image Processing and its Applications*, pages 37–40. IEE, Apr 1992.

[15] R M Gray. 'Vector Quantization'. *IEEE ASSP Mag.*, 1(2):4–29, April 1984.

[16] N M Nasrabadi. 'Use of Vector Quantizers in Image Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-85)*, pages 4.4.1–4.4.4. IEEE, March 1985.

[17] W J Welsh. 'Model-Based Coding of Video Images'. *Electronics and Communication Engineering Journal*, 3(1):29–36, February 1991.

[18] Y Nakaya, Y C Chuah, and H Harashima. 'Model-Based/Waveform Hybrid Coding for VideoTelephone Images'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-91)*, pages 2741–2744. IEEE, May 1991.

[19] F I Parke. 'Parameterized Models for Facial Animation'. *IEEE Comp. Graphics and Apps. Mag.*, 2(9):61–68, November 1982.

[20] K Aizawa and H Harashima. 'Model-Based Synthesis Image Coding System'. In *Proc. Global Telecomms. Conf.*, pages 132–135. IEEE, 1987.

[21] K Aizawa, H Harashima, and T Saito. 'Model-Based Analysis Synthesis Image Coding (MBASIC) System for a Person's Face'. *Signal Processing : Image Communication*, 1(2):139–152, 1989.

[22] R Forchheimer and T Kronander. 'Image Coding — From Waveforms to Animation'. *IEEE Trans. on Acoustics, Speech and Signal Proc.*, ASSP-37(12):2008–2023, December 1989.

[23] H Harashima and F Kishino. 'Intelligent Image Coding and Communications with Realistic Sensations — Recent Trends'. *IEICE Transactions*, E-74(6):1583–1592, June 1991.

[24] C S Choi, H Harashima, and T Takebe. 'Analysis and Synthesis of Facial Expressions in Knowledge-Based Coding of Facial Image Sequences'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-91)*, pages 2737–2740. IEEE, May 1991.

[25] W J Welsh, S Searby, and J B Waite. 'Model-Based Image Coding'. *Br. Telecom Technol. J.*, 8(3):94–106, July 1990.

[26] J B Waite and W J Welsh. 'Head Boundary Location Using Snakes'. *Br. Telecom Technol. J.*, 8(3):127–136, July 1990.

[27] M F Barnsley and A D Sloan. 'A Better Way to Compress Images'. *BYTE*, pages 215–223, January 1988.

[28] A E Jacquin. 'Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations'. *IEEE Trans. on Image Processing*, 1(1):18–30, January 1992.

[29] D Leakey. 'Integrated Services Digital Networks : Some Possible Ongoing Evolutionary Trends'. *Computer Networks and ISDN Systems*, 15:303–312, 1988.

[30] P J Lewis. 'Whatever happened to ISDN?'. *IEE Review*, 36(10):357–360, October 1990.

[31] P J Davidson. 'Review of the CCITT Recommendations for Integrated Services Digital Network'. *British Telecomms. Eng.*, 5:202–206, October 1986.

[32] K D Fogarty. 'Introduction to the CCITT I. Series Recommendations'. *Br. Telecom Technol. J.*, 6(1):5–13, January 1988.

[33] J F Marshall and F Welsby. 'British Telecom's ISDN Implementation'. *British Telecomms. Eng.*, 9(8):43–46, August 1990.

[34] D Pearson. 'Packet Video'. *IEE Review*, 36(8):315–318, August 1990.

[35] C J Hughes and A G Waters. 'Packet Power : B-ISDN and the Asynchronous Transfer Mode'. *IEE Review*, 37(10):357–360, October 1991.

[36] P E White. 'The Role of the Broadband Integrated Servies Digital Network'. *IEEE Communs. Mag.*, 29(3):116–119, March 1991.

[37] D Delisle and L Pelamourgues. 'B-ISDN and how it works'. *IEEE Spectrum*, 28(8):39–42, March 1991.

[38] T Koga, K Niwa, and Y Iijima K Iinuma. 'Low Bit Rate Motion Video Coder/Decoder for Teleconferencing'. *Optical Engineering*, 26(7):590–595, July 1987.

[39] A Zaccarin and B Liu. 'Transform Coding of Colour Images with Limited Palette Size'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-91)*, pages 2625–2628. IEEE, May 1991.

[40] S E Budge and R L Baker. 'Compression of Color Digital Images Using Vector Quantization in Product Codes'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-85)*, pages 4.5.1–4.5.4. IEEE, March 1985.

[41] M D Carr. 'New Video Coding Standard for the 1990's'. *Electronics and Communication Engineering Journal*, 2(3):119–124, June 1990.

[42] M D Carr. 'Video Codec Hardware to Realise a New World Standard'. *Br. Telecom Technol. J.*, 8(3):28–35, July 1990.

[43] E M Gold. 'New Video Standard Draws Mixed Reactions from Users'. *Networking Management*, 8(7):68–72, September 1990.

[44] J E Thompson. 'European Collaboration on Picture Coding Research for 2Mbit/s Transmission'. *IEEE Trans. on Comms.*, COM-29(12):2003–2004, December 1981.

[45] R C Nicol and N Mukawa. 'Motion Video Coding in CCITT SG XV — The Coded Picture Format'. In *Proc. Globecom 88*, pages 992–996. IEEE, November 1988.

[46] J Guichard, D Devimeux, T Koga, G Morrison, N Randall, and J Speidel. 'Motion Video Coding in CCITT SG XV — Hardware Trials'. In *Proc. Globecom 88*, pages 37–40. IEEE, November 1988.

[47] R Plompen, Y Hatori, W Geuen, J Guichard, M Guglielmo, and H Brusewitz. 'Motion Video Coding in CCITT SG XV — The Video Source Coding'. In *Proc. Globecom 88*, pages 997–1004. IEEE, November 1988.

[48] P H Ang, P A Ruetz, and D Auld. 'Video Compression Makes Big Gains'. *IEEE Spectrum*, 28(10):16–19, October 1991.

[49] G P Hudson, H Yasuda, and I Sebestyen. 'The International Standardisation of a Still Picture Compression Technique'. In *Proc. Globecom 88*, pages 1016–1021. IEEE, November 1988.

[50] G P Hudson. 'The International Standardisation of a Still Picture Compression Technique'. *Br. Telecom Technol. J.*, 7(3):41–47, July 1989.

[51] G Wallace, R Vivian, and H Poulsen. 'Subjective Testing Results for Still Picture Compression Algorithms for International Standardisation of a Still Picture Compression Technique'. In *Proc. Globecom 88*, pages 1022–1027. IEEE, November 1988.

[52] A Leger, J L Mitchell, and Y Yamazaki. 'Still Picture Compression Algorithms Evaluated for International Standardisation'. In *Proc. Globecom 88*, pages 1028–1032. IEEE, November 1988.

[53] S Okubo, T Omachi, and F Ono. 'International Standardization on Picture Coding'. *IEICE Transactions*, E-74(3):533–539, March 1991.

[54] B E Nelson and C J Read. 'A Bit-Serial VLSI Vector Quantizer'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-86)*, pages 2211–2214. IEEE, April 1986.

[55] P Cappello, G Davidson, A Gersho, C Koc, and V Somayazulu. 'A Systolic Vector Quantization Processor for Real-Time Speech Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-86)*, pages 2143–2146. IEEE, April 1986.

[56] P A Ramamoorthy and B Potu. 'Bit Serial Systolic Chip Set for Real-Time Image Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 787–790. IEEE, April 1987.

[57] G A Davidson, P R Cappello, and A Gersho. 'Systolic Architectures for Vector Quantization'. *IEEE Trans. on Acoustics, Speech and Signal Proc.*, ASSP-36(10):1651–1664, October 1988.

[58] P A Ramamoorthy, B Potu, and T Tran. 'Bit-Serial VLSI Implementation of Vector Quantizer for Real-Time Image Coding'. *IEEE Trans. on Circuits and Systems*, CAS-36(10):1281–1290, October 1989.

[59] K Dezhgosha, M M Jamali, and S C Kwatra. 'Real-Time VLSI Architecture for a VQ-Based High-Quality Image Coding System'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-89)*, pages 2425–2428. IEEE, May 1989.

[60] C-Y Lee and S-C Juan. 'An ASIC Architecture for Real-Time Image/Video Coding Based on Fixed-Basis-Distortion Vector Quantization'. In *Proc. Int. Symp. on Circuits and Systems (ISCAS-92)*, pages 1676–1679. IEEE, May 1992.

[61] M Yan, J V McCanny, and Y Hu. 'VLSI Architectures for Digital Image Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 913–916. IEEE, April 1990.

[62] T Komarek and P Pirsch. 'VLSI Architecttures for Block Matching Algorithms'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-89)*, pages 2457–2460. IEEE, May 1989.

[63] H Abut, B P M Tao, and J L Smith. 'Vector Quantizer Architechtures for Speech and Image Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 756–759. IEEE, April 1987.

[64] I Tamitani, H Harasaki, T Nishitani, Y Endo, M Yamashina, and T Enomoto. 'A Real-Time Video Signal Processor Suitable for Motion Picture Coding Applications'. *IEEE Trans. on Circuits and Systems*, CAS-36(10):1259–1266, October 1989.

[65] P Israelsen. 'VLSI Implementation of a Vector Quantization Processor'. In *Proc. Data Compression Conf. (DCC-91)*, page 463, April 1991. Poster Presentation.

[66] W-C Fang, B J Sheu, O T-C Chen, and J Choi. 'A VLSI Neural Processor for Image Data Compression Using Self-Organization Networks'. *IEEE Trans. on Neural Networks*, NN-3(3):506–518, May 1991.

[67] C E Shannon. 'A Mathematical Theory of Communication'. *Bell Systems Technical Journal*, 27:379–423, 1986.

[68] J Linde, A Buzo, and R M Gray. 'An Algorithm for Vector Quantizer Design'. *IEEE Trans. on Comms.*, COM-28(1):84–95, January 1980.

[69] S P Lloyd. 'Least Squares Quantization in PCM'. *IEEE Trans. on Information Theory*, IT-28(2):129–137, March 1982.

[70] N M Nasrabadi and R A King. 'Image Coding Using Vector Quantization : A Review'. *IEEE Trans. on Comms.*, COM-36(8):957–971, August 1988.

[71] J Makhoul, S Roucos, and H Gish. 'Vector Quantization in Speech Coding'. *Proceedings of the IEEE*, 73(11):1551–1588, November 1985.

[72] R M Gray and J Linde. 'Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources'. *IEEE Trans. on Comms.*, COM-30(2):381–389, February 1982.

[73] F Kossentini, M J T Smith, and C F Barnes. 'Large Block RVQ with Multipath Searching'. In *Proc. Int. Symp. on Circuits and Systems (ISCAS-92)*, pages 2276–2279. IEEE, May 1992.

[74] E A Riskin, T Lookabaugh, P A Chou, and R M Gray. 'Variable Rate Vector Quantization for Medical Image Processing'. *IEEE Trans. on Medical Imaging*, MI-9(3):290–298, September 1990.

[75] P A Chou, T Lookabaugh, and R M Gray. 'Entropy-Constrained Vector Quantization'. *IEEE Trans. on Acoustics, Speech and Signal Proc.*, ASSP-37(1):31–42, January 1989.

[76] C L Yeh. 'Color Image-Sequence Compression Using Adaptive Binary-Tree Vector Quantization with Codebook Replenishment'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 1059–1062. IEEE, April 1987.

[77] B Ramamurthi and A Gersho. 'Image Vector Quantization with a Perceptually-based Cell Classifier'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-84)*, pages 145–148. IEEE, March 1984.

[78] H Bheda and K S Thyagarajan abd H Abut. 'A Fast Matrix Quantizer for Image Encoding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-85)*, pages 4.7.1–4.7.4. IEEE, March 1985.

[79] A Madisetti, H Subramonian, and V R Algazi. 'A Radius-Bucketing Approach to Fast Vector Quantization Encoding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-89)*, pages 1767–1770. IEEE, May 1989.

[80] M C Rost and K Sayood. 'The Root Lattices as Low Bit-Rate Vector Quantizers'. *IEEE Trans. on Information Theory*, IT-34(5):1053–1058, September 1988.

[81] T R Fischer. 'Geometric Source Coding and Vector Quantization'. *IEEE Trans. on Information Theory*, IT-35(1):137–145, January 1989.

[82] T-C Chen. 'A Lattice Vector Quantization Using a Geometric Decomposition'. *IEEE Trans. on Comms.*, COM-38(5):704–714, May 1990.

[83] G H Freeman, I F Blake, and J W Mark. 'Trellis Source Code Design as an Optimization Problem'. *IEEE Trans. on Information Theory*, IT-34(5):1226–1241, September 1988.

[84] B Hammer, A v Brandt, and M Schielein. 'Hierarchical Encoding of Image Sequences Using Multi-Stage Vector Quantization'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 1055–1058. IEEE, April 1987.

[85] Y-S Ho and A Gersho. 'Variable-Rate Multi-Stage Vector Quantization for Image Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-88)*, pages 1156–1159. IEEE, April 1988.

[86] H J Lee and D T L Lee. 'A Gain-Shape Vector Quantizer for Image Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-86)*, pages 141–144. IEEE, April 1986.

[87] E Daly and T R Hsing. 'Variable Bit-Rate Vector Quantization of Video Images for Packet Switched Networks'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-88)*, pages 1160–1163. IEEE, April 1988.

[88] D J Vaisey and A Gersho. 'Variable Block-Size Image Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 1051–1054. IEEE, April 1987.

[89] I Dinstein, K Rose, and A Heiman. 'Variable Block-Size Transform Image Coder'. *IEEE Trans. on Comms.*, COM-38(11):2073–2078, November 1990.

[90] J L de Bougrenet de la Tocnaye and J F Cavassilas. 'Image Coding Using an Adaptive Sampling Technique'. *Signal Processing : Image Communication*, 1(1):75–80, 1989.

[91] P J Cordell and R J Clarke. 'An Interpolative Spatial Domain Technique for Coding Image Sequences'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-89)*, pages 1917–1920. IEEE, May 1989.

[92] L Wang and M Goldberg. 'Pyramid Transform Coding Using Vector Quantization'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-88)*, pages 812–815. IEEE, April 1988.

[93] J L Boxerman and H J Lee. 'Variable Block-Sized Vector Quantization of Grayscale Images with Unconstrained Tiling'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 2277–2280. IEEE, April 1990.

[94] L Corte-Real and A P Alves. 'Vector Quantization of Image Sequences using Variable Size and Variable Shape Blocks'. *Electronics Letters*, 26(18):1483–1484, August 1990.

[95] S S Dixit and Y Feng. 'Hierarchical Address Vector Quantization for Image Coding'. *CVGIP : Graphical Models and Image Processing*, 53(1):63–70, 1991.

[96] N M Nasrabadi, S E Lin, and Y Feng. 'Interframe Hierarchical Vector Quantization'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-89)*, pages 1739–1742. IEEE, May 1989.

[97] P Strobach, D Schütt, and W Tengler. 'Space-Variant Regular Decomposition Quadtrees in Adaptive Interframe Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-88)*, pages 1096–1099. IEEE, April 1988.

[98] R L Baker and H-S Shen. 'A Finite-State Vector Quantizer for Low-Rate Image Sequence Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 760–763. IEEE, April 1987.

[99] H-S Shen and R L Baker. 'A Finite State/Frame Difference Interpolative Vector Quantizer for Low Rate Image Sequence Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-88)*, pages 1188–1191. IEEE, April 1988.

[100] N M Nasrabadi and Y Feng. 'A Dynamic Finite-State Vector Quantization Scheme'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 2261–2264. IEEE, April 1990.

[101] N M Nasrabadi and Y Feng. 'Image Compression Using Address-Vector Quantization'. *IEEE Trans. on Comms.*, COM-38(12):2166–2173, December 1990.

[102] N M Nasrabadi and Y Feng. 'A Multilayer Address Vector Quantization Technique'. *IEEE Trans. on Circuits and Systems*, CAS-37(7):912–921, July 1990.

[103] Y Feng and N M Nasrabadi. 'A New Vector Quantization Scheme Using Inter-Block Correlation :Address Vector Quantizer'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-88)*, pages 755–758. IEEE, April 1988.

[104] Y Feng and N M Nasrabadi. 'A Dynamic Address-Vector Quantization Algorithm Based on Inter-Block and Inter-Color Correlation for Color Image Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-89)*, pages 1755–1758. IEEE, May 1989.

[105] J D McAuliffe, L E Atlas, and C Rivera. 'A Comparison of the LBG Algorithm and Kohonen Neural Network Paradigm for Image Vector Quantization'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 2293–2296. IEEE, April 1990.

[106] W Equitz. 'Fast Algorithms for Vector Quantization Picture Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 725–728. IEEE, April 1987.

[107] W H Equitz. 'A New Vector Quantization Clustering Algorithm'. *IEEE Trans. on Acoustics, Speech and Signal Proc.*, ASSP-37(10):1568–1575, October 1989.

[108] K Zeger and A Gersho. 'Stochastic Relaxation Algorithm for Improved Vector Quantizer Design'. *Electronics Letters*, 25(14):896–898, July 1989.

[109] D E Pearson and M W Whybray. 'Transform Coding of Images Using Interleaved Blocks'. *IEE Proceedings-F*, 133(5):466–472, August 1984.

[110] H F Sun and M Goldberg. 'Adaptive Vector Quantization for Image Sequence Coding'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-85)*, pages 10.1.1–10.1.4. IEEE, March 1985.

[111] M Goldberg and H F Sun. 'Image Sequence Coding Using Vector Quantization'. *IEEE Trans. on Comms.*, COM-34(7):703–710, July 1986.

[112] M Goldberg and H F Sun. 'Frame Adaptive Vector Quantization for Image Sequence Coding'. *IEEE Trans. on Comms.*, COM-36(5):629–635, May 1988.

[113] H F Sun and M Goldberg. 'Radiographic Image Sequence Coding Using Two-Stage Adaptive Vector Quantization'. *IEEE Trans. on Medical Imaging*, MI-7(2):118–126, June 1988.

[114] P Monet and C Labit. 'Codebook Replenishment in Classified Pruned Tree-Structured Vector Quantization of Image Sequences'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 2285–2288. IEEE, April 1990.

[115] A Gersho and M Yano. 'Adaptive Vector Quantization by Progressive Codevector Replacement'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-85)*, pages 4.6.1–4.6.4. IEEE, March 1985.

[116] R P Lippmann. 'An Introduction to Computing with Neural Nets'. *IEEE ASSP Mag.*, 4(2):4–22, April 1987.

[117] X Liu and J Herault. 'Colour Image Processing by a Neural Network Model'. In *Proc. Int. Neural Network Conf. (INNC-90)*, pages 3–6. IEEE, July 1990.

[118] M Mougeot and R Barrow. 'From Static to Dynamic Image Compression'. In *Proc. Int. Neural Network Conf. (INNC-90)*, pages 59–62. IEEE, July 1990.

[119] G L Sicuranza and G Ramponi. 'Artificial Neural Network for Image Compression'. *Electronics Letters*, 26(7):477–479, March 1990.

[120] C Manikopoulos, G Antoniou, and S Metzelopoulou. 'ANS Classification of Finite State Machine for High Compression Video Conference Coding'. In *Proc. Int. Neural Network Conf. (INNC-90)*, pages 1–55, July 1990.

[121] T Kohonen. 'Self-Organized Formation of Topologically Correct Feature Maps'. *Biological Cybernetics*, 43:59–69, 1982.

[122] T Kohonen. 'Clustering, Taxonomy, and Topological Maps of Patterns'. In *Proc. 6th Int. Conf. on Pattern Recognition*, pages 114–128. IEEE, October 1982.

[123] N M Nasrabadi and Y Feng. 'Vector Quantization of Images Based Upon the Kohonen Self-Organising Feature Maps'. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN-88)*, pages 101–108, July 1988.

[124] D S Erickson and K S Thyagarajan. 'A Neural Network Approach to Image Compression'. In *Proc. Int. Symp. on Circuits and Systems (ISCAS-92)*, pages 2921-2924. IEEE, May 1992.

[125] D D Giusto and G Vernazza. 'Color-Image Coding by an Advanced Vector-Quantizer'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 2265–2268. IEEE, April 1990.

[126] C-Y Hsu and H-E Wu. 'An Improved Algorithm for Kohonen's Self-Organizing Feature Maps'. In *Proc. Int. Symp. on Circuits and Systems (ISCAS-92)*, pages 328–331. IEEE, May 1992.

[127] T-C Lee and A M Peterson. 'Adaptive Vector Quantization Using a Self-Development Neural Network'. *IEEE J. on Selec. Areas in Commun.*, 8(8):1458–1471, October 1990.

[128] A K Krishnamurthy, S C Ahalt, D E Melton, and P Chen. 'Neural Networks for Vector Quantization of Speech and Images'. *IEEE J. Selec. Areas in Commun.*, SAC-8(8):1449–1456, October 1990.

[129] S A Harp and T Samad. 'Genetic Optimization of Self-Organizing Feature Maps'. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN-91)*, pages I–341–I–346, July 1991.

[130] K K Truong and R M Mercereau. 'Structral Image Codebooks and the Self-Organizing Feature Map Algorithm'. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 2289–2292. IEEE, April 1990.

[131] S P Luttrell. 'Self-Organising multilayer topographic mappings'. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN-88)*, pages 93–100, July 1988.

[132] S P Luttrell. 'Hierarchical Self-Organising Networks'. In *Proc. 1st IEE Conf. on Neural Networks*. IEE, 1989.

[133] S P Luttrell. 'Hierarchical Vector Quantization'. *IEE Proceedings-I*, 136(6):405–413, December 1989.

[134] S P Luttrell. 'Image Compression using a multilayer neural network'. *Pattern Recognition Letters*, 10:1–7, 1989.

[135] S P Luttrell. 'Self-Supervised training of Hierarchical Vector Quantizers'. In *Proc. 2nd IEE Conf. on Neural Networks*. IEE, 1991.

[136] C-F Chang, B J Sheu, W-C Fang, and J Choi. 'A Trainable Analog Neural Chip for Image Compression'. In *Proc. IEEE Custom Integrated Circuits Conf.*, pages 16.1.1–16.1.4. IEEE, May 1991.

[137] S Anderson. *A VLSI Smart Sensor-Processor for Fingerprint Comparison*, pages 80–84. PhD Thesis, University of Edinburgh, 1993.

[138] D Bursky. 'Parallelism pushes DSP throughput'. *Electronic Design*, pages 151–154, March 1994.

[139] T Skinner. 'Harness multiprocessing power for DSP systems'. *Electronic Design*, pages 55–72, February 1994.

[140] M Maruyama, H Nakahima, T Araki, S Sakiyama, Y Kitao, K Aono, and H Yamada. 'An Image Signal Multiprocessor on a Single Chip'. *IEEE J. of Solid-State Circuits*, 25(6):1476–1483, December 1990.

[141] T Murakami, K Kamizawa, M Kameyama, and S-I Nakagawa. 'A DSP Architectural Design for Low Bit-Rate Motion Video Codec'. *IEEE Trans. on Circuits and Systems*, CAS-36(10):1267–1274, October 1989.

[142] H Jeschke, K Gaetze, and P Pirsch. 'A VLSI Based Multiprocessor Architecture for Video Signal Processing'. In *Proc. Int. Symp. on Circuits and Systems (ISCAS-92)*, pages 1685–1688. IEEE, May 1992.

[143] M Ismail and T Fiez. *Analogue VLSI Signal and Information Processing*, pages 5–7. McGraw-Hill, New York, NY, first edition, 1994.

[144] C Toumazou, F J Lidgey, and D G Haigh. *Analogue IC design : the current mode approach*, pages 11–90. Peter Peregrinus Ltd., London, UK, first edition, 1990.

[145] M Bloom. 'A Memory to Remember'. *Electronic Systems Design Magazine*, pages 5–9, October 1989.

[146] A Wright. 'Analogue data storage – speaking of the future'. *Electronics World and Wireless World*, pages 110–113, February 1992.

Appendix A

Addressing Methods for Label Updates

In interframe coding, to update only part of a frame, addressing information must accompany the updates to indicate where in the image the updates are to go. To calculate bit-rate measures for the label replenishment schemes of chapter 4, it is necessary to estimate the amount of data required for this addressing information. This appendix introduces three ways that the addressing can be performed, and calculates the addressing requirements for a given example.

The example chosen is for the same configuration used for the experiments in chapter 4. Here the QCIF images, of 176×144 pixels, were coded using a codebook of size 3×3 pixels. The total number of vectors in this image is

$$(\mathrm{mod})\frac{176}{3} \times (\mathrm{mod})\frac{144}{3} = 2832, \tag{A.1}$$

where (mod) specifies the integer modulus of the division. The number of label updates used in these experiments was 100, 200, 400, 800 and 2832 labels per frame (for 2832 labels there is no addressing requirement as this signifies that all the labels are being updated at each frame). For each of the three addressing methods proposed, the data requirement for these update rates is calculated. In conclusion, the best method for each update rate is chosen, and the results are collated to give bit-rate calculations for the given example.

1. **Binary representation of label map.** The position of all labels which are to change can be identified by a binary representation of the label map, with updates indicated

by a 1 in the relevant position. Label updates then require to be sent sequentially after this label map. The size of the label map, in bits, is simply the number of vectors in the input image. For the given example, this is equal to 2832 bits. With this method the addressing data requirement is independent of the number of label updates.

2. **Identification of each label with its address.** To address each label separately with its position in the label map requires

$$\text{(mod) } log_2 \text{ (label map size)} \quad \text{bits} \tag{A.2}$$

For the given example, this is equal to 12 bits. For the update rates under consideration, the update requirements are shown in table A–1.

Label updates	Bit requirement
100	1200
200	2400
400	4800
800	9600

Table A–1: Bit requirement for addressing label updates by their unique address in the label map.

3. **Run length encoding of label map.** For certain update rates it will be more efficient to encode the binary label map before transmission. This could be done with run length encoding, which codes the label map as a set of integers representing the number of consecutive 0's or 1's. This method gives a variable amount of data for the coded label map depending on the distribution pattern of the label updates within the image. The following analysis calculates the maximum number of bits that would be required.

Consider coding each run with an integer represented by n bits. Integer values from 0 to $(2^n - 2)$ are used to encode run lengths from 1 to $(2^n - 1)$. The code $(2^n - 1)$ is reserved to indicate that the current run of 0's or 1's is longer than $(2^n - 1)$ bits long, and that the next integer encodes a run of the same value (0 or 1). Figure A–1 shows an example coding of a reduced size label map with $n = 2$.

0	0	0	1	1	0	1	0
0	0	1	1	1	1	0	1
0	0	0	0	1	1	1	0
0	0	0	0	1	1	0	0

Figure A–1: Run length encoding of a binary label map. With $n = 2$, the above label map would be encoded with the integer set (2,1,0,0,2,3,0,0,0,3,0,2,3,1,1,1).

The most difficult distribution of label updates for this method to encode would be where the labels were placed on every second position, creating the maximum number of minimum length runs. In this case, the number of runs would be given by

$$\text{runs} = 2U + \frac{(L - (2U))}{2^n - 1} \tag{A.3}$$

where U is the number of label updates, and L is the total number of labels in the frame. For the considered example, the number of bits required would thus be

$$\text{bits} = n(2U + \frac{(2832 - (2U))}{2^n - 1}) \tag{A.4}$$

Table A–2 shows the calculated values for the given label update rates, with $n = 2, \ldots 6$.

n	Number of label updates			
	100	200	400	800
2	2156	2422	**2956**	**4032**
3	1728	**2244**	3273	5328
4	1504	2252	3744	6732
5	**1425**	2395	4330	8200
6	1452	2634	4998	9720

Table A–2: Bit requirements for run length encoding of the binary label map. Bold numbers indicate the optimal value for each label update rate.

Summary

Table A–3 summarises the best addressing method and the related data requirement for each label update rate, for the example under consideration. Also shown is the total frame update rate, which is the data for the label updates plus the addressing requirement. This is expressed as bits per frame and as bits per pixel. The number of bits for each label update is 8, which corresponds to a codebook containing 256 codewords.

Label	Addressing	Update requirement in bits			Total
Updates	Method	Labels	Addressing	Total	(bpp)
100	2	800	1200	2000	0.09
200	3	1600	2244	3844	0.15
400	1	3200	2832	6032	0.24
800	1	6400	2832	9232	0.36
2382	(none)	22656	0	22656	0.89

Table A–3: Label update rates converted into frame update rates.

Appendix B

KSOFM Network Training Example

This appendix shows results from the experiment performed in section 5.2.3.2. In this experiment a KSOFM network was trained for 10 epochs with the *Claire* image. At each epoch the KSOFM neuron matrix, and the codebook generated from it, are shown.

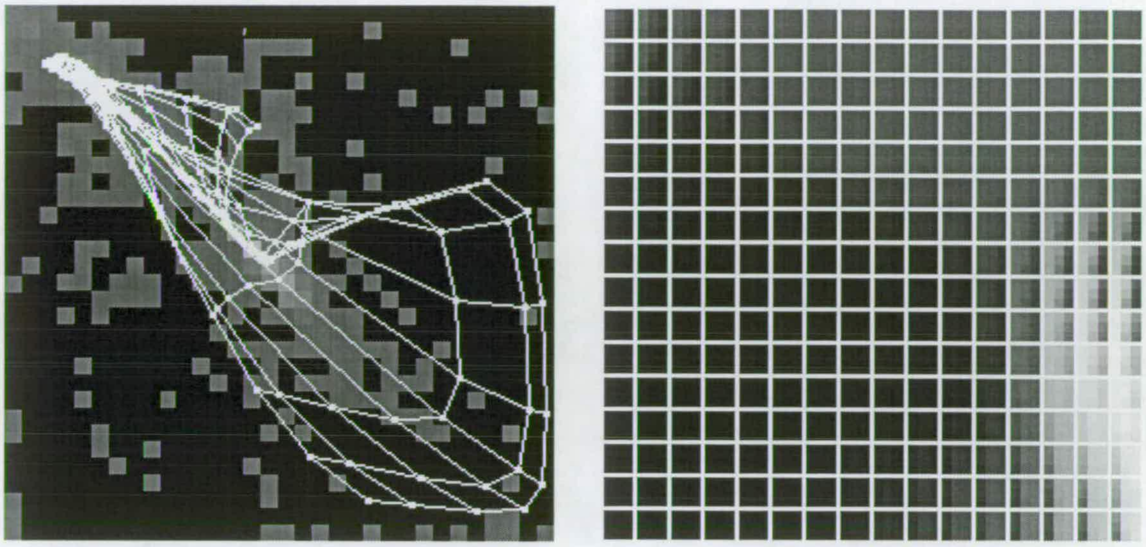


Figure B–1: KSOFM neural network and related codebook at training epoch 1.

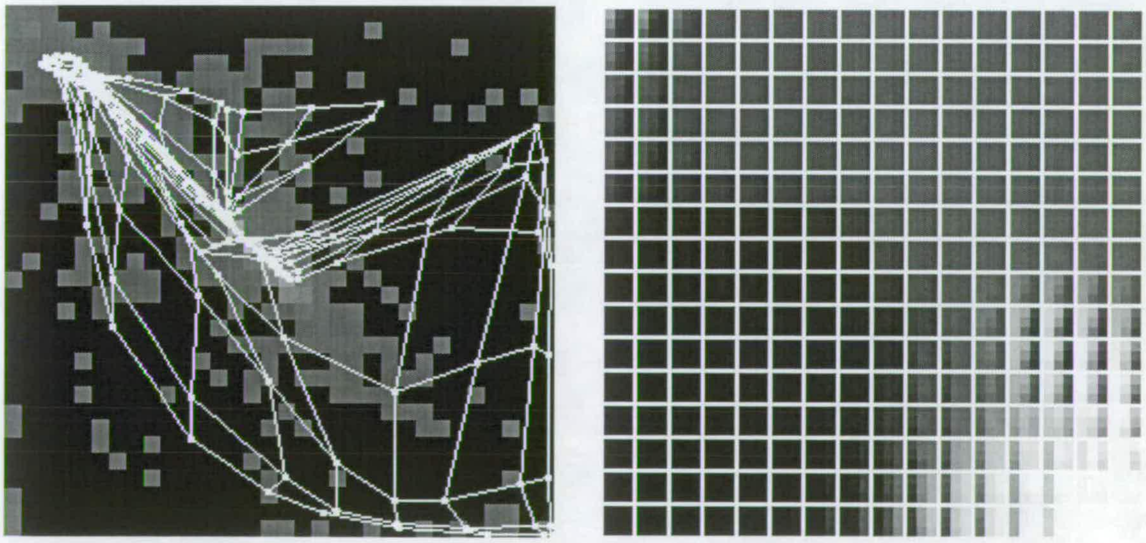


Figure B–2: KSOFM neural network and related codebook at training epoch 2.

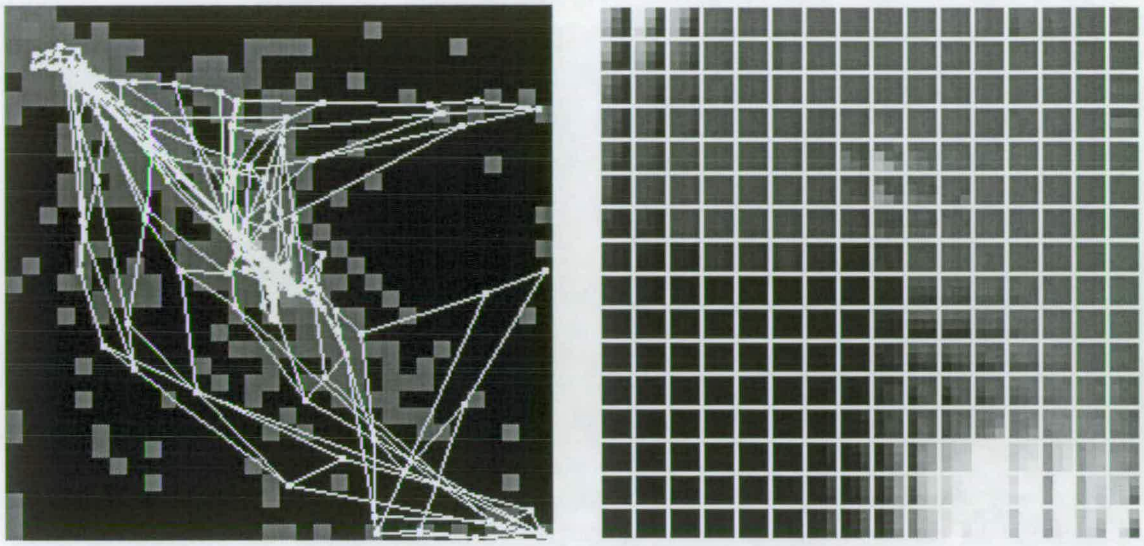


Figure B–3: KSOFM neural network and related codebook at training epoch 3.

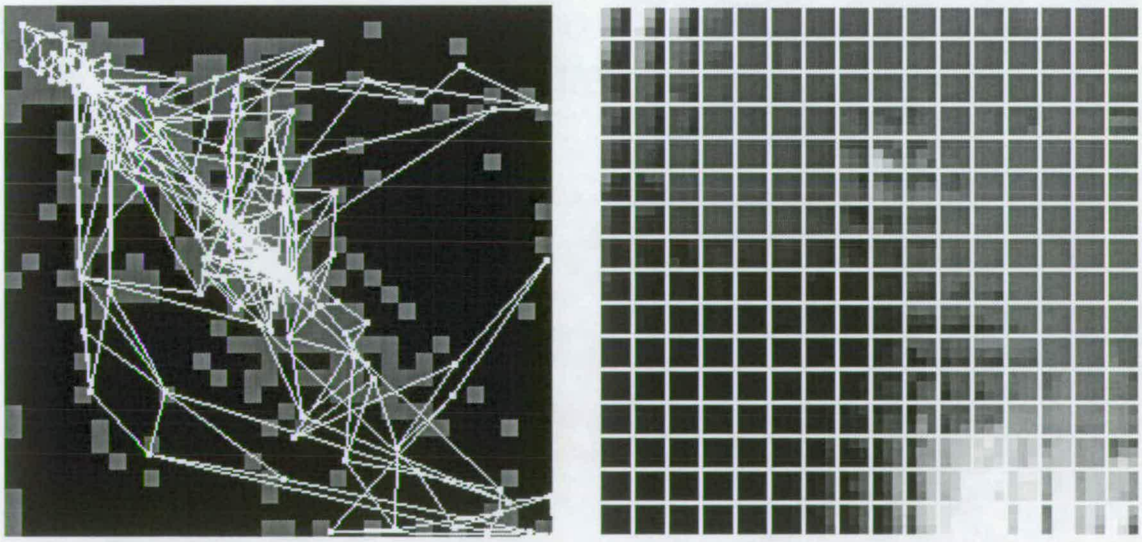


Figure B–4: KSOFM neural network and related codebook at training epoch 4.

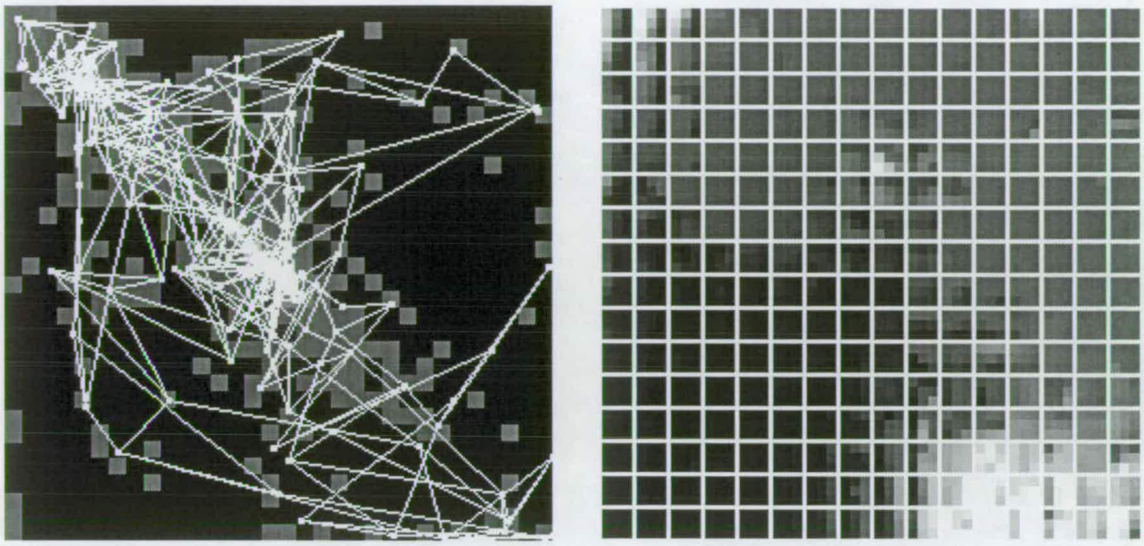


Figure B–5: KSOFM neural network and related codebook at training epoch 5.

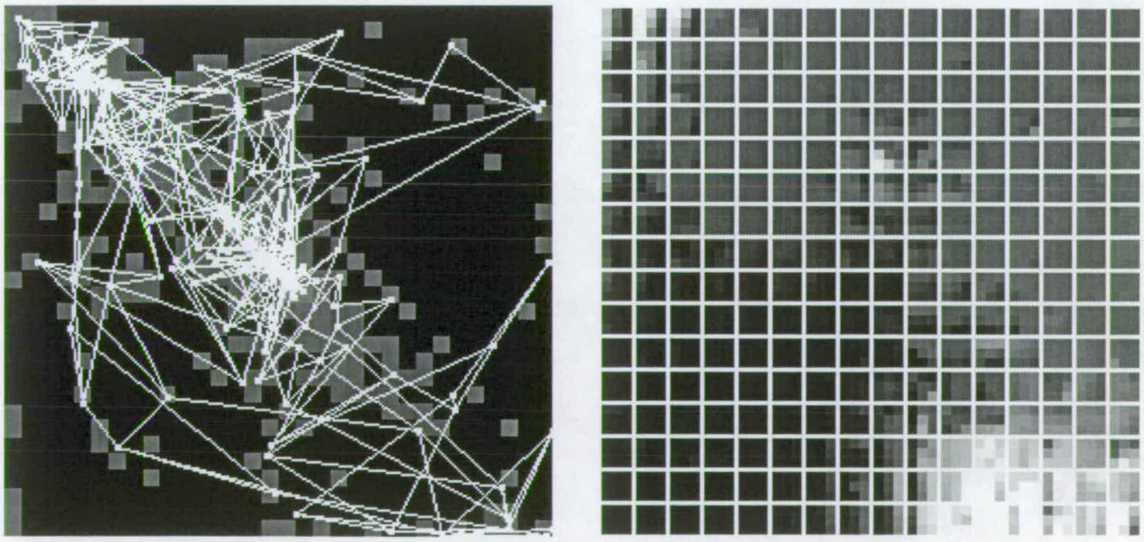


Figure B–6: KSOFM neural network and related codebook at training epoch 6.

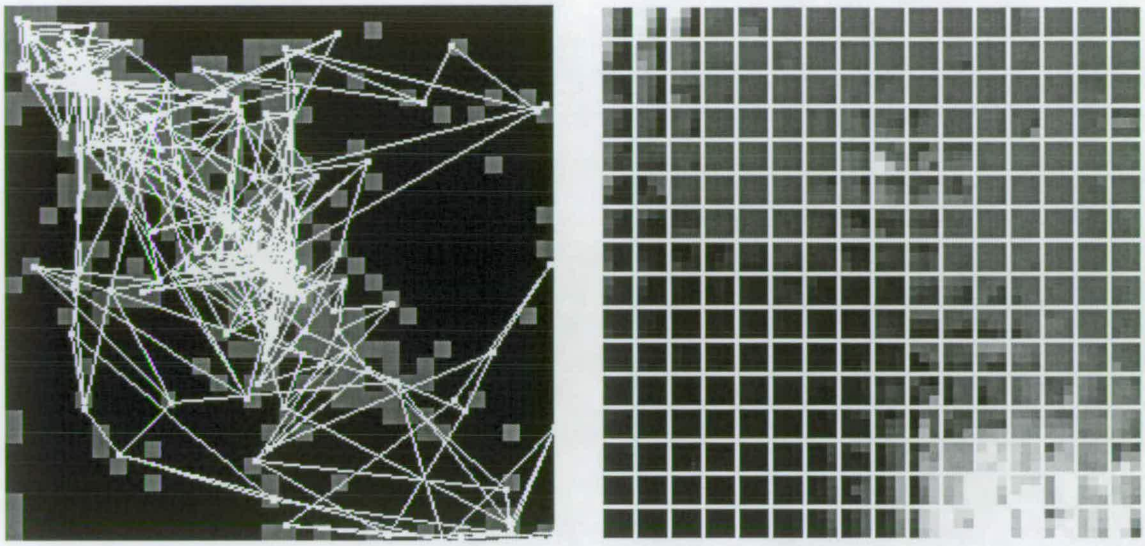


Figure B–7: KSOFM neural network and related codebook at training epoch 7.

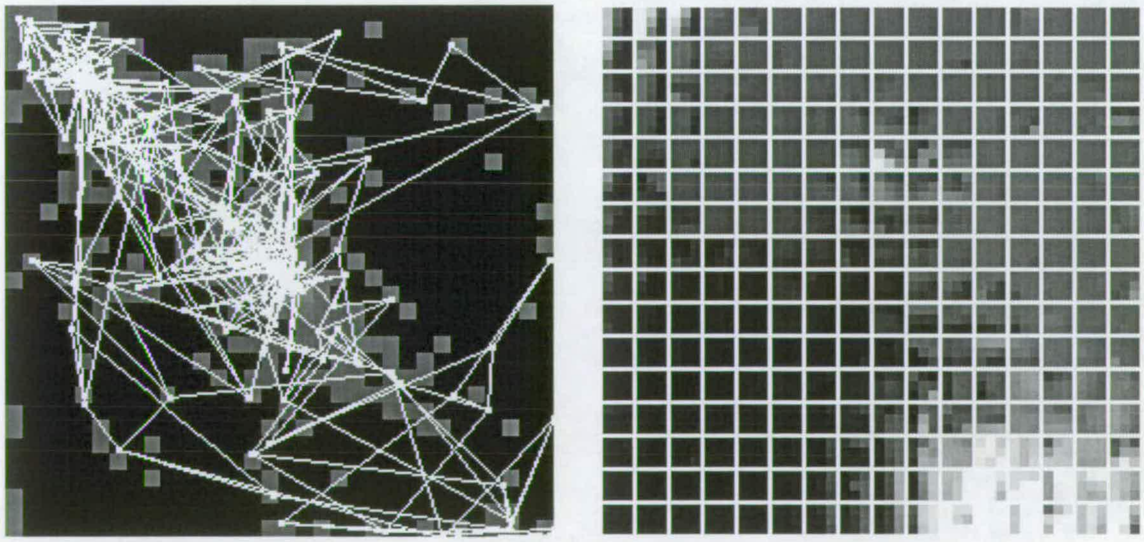


Figure B–8: KSOFM neural network and related codebook at training epoch 8.

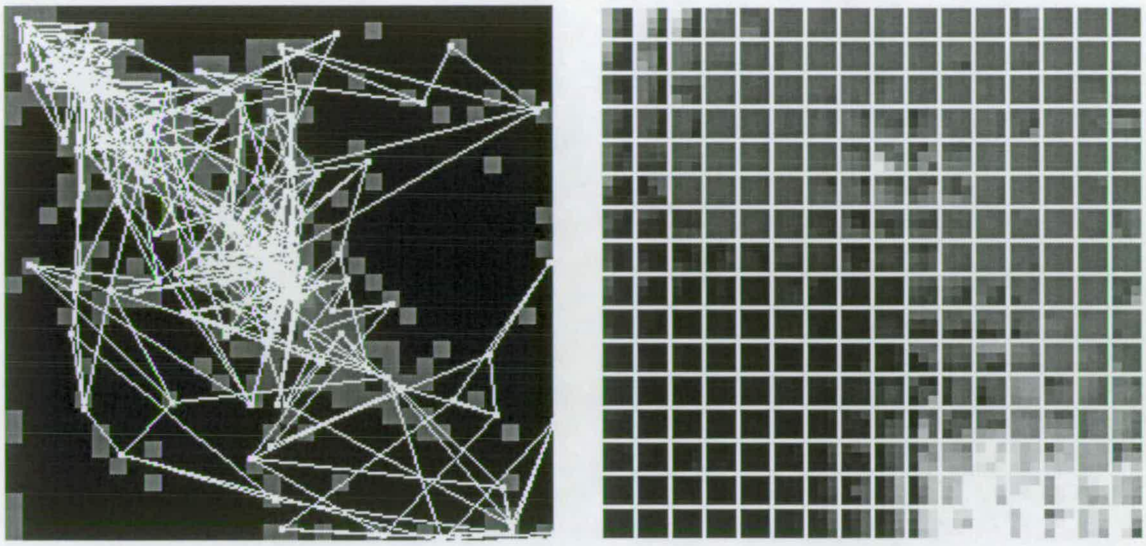


Figure B–9: KSOFM neural network and related codebook at training epoch 9.

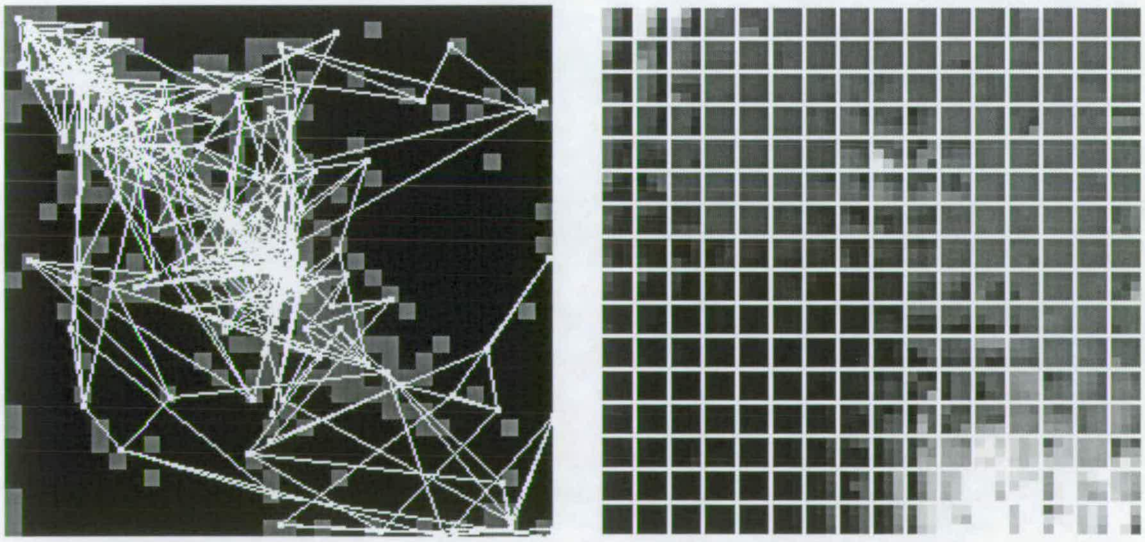


Figure B–10: KSOFM neural network and related codebook at training epoch 10.

Appendix C

Published Work

From the work presented in this thesis, a paper was presented by the author at the *British Machine Vision Conference* in Leeds, England, in September 1992. This paper is included here.

A Comparison of Vector Quantization Codebook Generation Algorithms Applied to Automatic Face Recognition.

C. S. Ramsay[†] K. Sutherland[†] D. Renshaw and P.B. Denyer
Integrated Systems Group, Electrical Engineering Department,
University of Edinburgh, Edinburgh, EH9 3JL.

Abstract

Automatic facial recognition is an attractive solution to the problem of computerised personal identification. In order to facilitate a cost effective solution, high levels of data reduction are required when storing the facial information. Vector Quantization has previously been used as a data reduction technique for the encoding of facial images.

This paper identifies the fundamental importance of the vector quantizer codebooks in the performance of the system. Two different algorithms – the Linde-Buzo-Gray algorithm and Kohonen's Self Organising Feature Map – have been used to obtain two sets of facial feature codebooks. For comparison, the system performance has also been analysed using a codebook dedicated to the test population. It has been shown that by using a *good* codebook generation algorithm it is possible to substantially reduce the dimensionality of the vector codebooks, with remarkably little degradation in system performance.

1 Introduction

Automatic facial recognition is now receiving an increasing amount of research interest [1], largely due to its possible applicability to the automatic personal identification task. As such, automatic face recognition is attempting to stake its claim among the other biometric systems available (notably fingerprint, hand geometry, dynamic signature matching and voice pattern recognition).

In order to establish biometric systems as likely tools for personal identification we must instill public confidence in the concept of biometric storage and retrieval. In this area automatic face recognition scores heavily over the other likely biometrics, as the least intrusive and the most *natural* personal identification process. It is also important that any prototype system has a demonstrably high accuracy. To obtain such high accuracy a likely advancement would be to incorporate the use of a number of different biometrics into one identification system, making the final response conditional on all the available information.

However, in practice, the overriding factor in achieving a *marketable* system will be the data reduction obtained, since this controls the speed at which multiple comparisons can be performed and, ultimately, the cost, as data storage

[†]These authors have SERC studentships.

requirement is a prime factor in this area. The available level of data reduction is now of particular importance given the increasing use of smart-card technology and the ever increasing likelihood that we will all eventually carry personal biometric information with us in this way. If a facial image is to be one of many pieces of biometric information stored on our smart-card, then accurate, high data reduction, parametrisation of the face is required.

The authors have previously introduced a Vector Quantization (VQ) based facial recognition technique and presented results for a recognition experiment using 30 individuals [2]. In this paper we would like, firstly, to present further results on a 33% larger data set and, secondly, to investigate the use of VQ codebook generation techniques to reduce further the data space required to store the intrinsics of the face (or the *facial signature*). The codebook generation techniques used here to reduce the overall number of vectors are Kohonen's neural Self-Organising Feature Map and the Linde-Buzo-Gray algorithm. However, firstly, a brief overview of the entire facial recognition algorithm will be presented.

2 System Overview

The intrinsic function of a pattern recognition process is that of parameterisation, *i.e.* the extraction of the fundamental characteristics of the object to be recognised. In [2] the authors presented an algorithm which distilled the face into a compact facial signature, while still maintaining much of the *recognisability* of the initial input face.

In brief, the system identifies seven fundamental features; each eye, the nostrils, the bridge of the nose, the mouth, the chin and the hair. In addition to storing these individual facial parts, the facial signature also incorporates a coarsely sub-sampled view of the entire face. The partitioning of the face used here is shown in Figure 1. These facial features are automatically located using a template matching algorithm based on Fischler and Elschalger's *feature embedding* approach [3]. Manual correction of the location failures was performed.

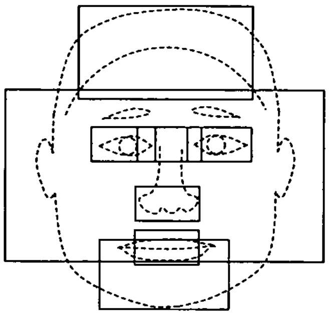


Figure 1: Facial Features Exploited to Differentiate between Individuals.

Having extracted these eight facial parts for storage and discarded the rest of the image data, there still remains the task of data reduction. The signal processing function of VQ has been chosen to provide the requisite reduction. The use of VQ for image coding is widespread [4]. However, recognition analysis using VQ is much more novel.

Applied to faces, VQ performs a function analogous to the police *photofit* system. In turn, each of the eight selected features is compared with a codebook of standard examples (or *vectors*) of only that feature; using the normalised euclidean distance as a metric, the most similar of these standard examples is chosen as the most likely match. Then to store that feature, we need only to keep the *index* of the standard feature and not all the data points which constitute that vector.

It is desirable to train a facial recognition device on a number of examples of each person it is expected to recognise, in order to allow it to construct internal representations of each person. To facilitate this, a system of *feature histograms* has been devised; these reflect the ways in which the subject's face has varied during training. To obtain the histograms we use the VQ technique described above to encode each feature, from each training image, then that *match* is recorded in the histogram. Thus, given 20 possible vectors and ten training images one particular *histogram* could look like this, Figure 2.

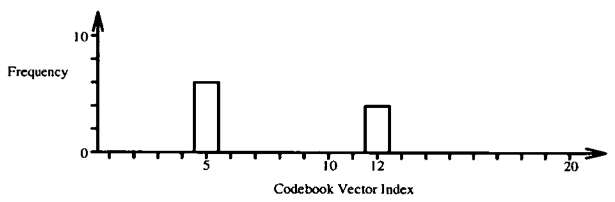


Figure 2: One Feature Histogram for One Person's Training Images. This feature has been mapped six times to codeword 5 and four times to codeword 12 during training.

In order to differentiate one person from another their respective histograms would have to show significant differences. If we look at the histograms of all eight features for two different people, Figure 3, it can be seen that there is only a low level of variability within training for each person, yet there is a substantial difference between these characteristics between these two people. Low *within person* and high *between person* variabilities are essential for a good recognition system.

By storing the histograms obtained in the manner described above it is argued that we have the requisite facial data to perform recognition. Thus, the feature histograms can be thought of as our facial signatures. The problem with this approach is that, to date, the VQ codebook entries have been extracted from a control image of each member of the test population. In this way the data storage requirement, as determined by the number of vectors, is dependent on the population size. Thus, if we were to enlarge the population there would be a consequent rise in data storage requirements. To combat this, the following section outlines the ways in which the codebook dimensionality can be reduced.

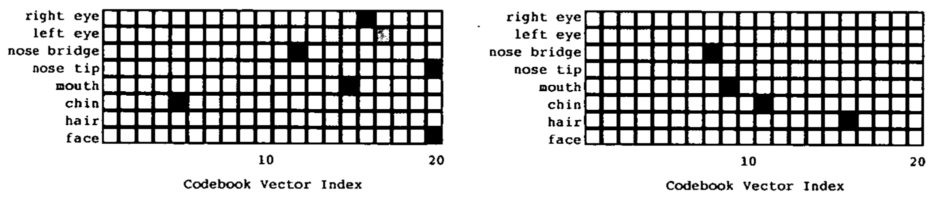


Figure 3: Feature Histograms for Two Population Members.
In this graphic all eight feature histograms are shown, with grey tone signifying the frequency of use of each codebook vector. *Black* represents a well used vector, with lighter tones indicating less use.

3 Codebook Generation

The process of VQ relies on having a good set of vectors in its codebook. For image compression, these vectors are chosen to minimise the overall pixel level error introduced. However, for face recognition, other considerations are also important. For example, distinctiveness may be more significant than pixel error when encoding facial features. Fortunately, there are many different algorithms available which perform codebook generation [5].

As an initial starting point, each feature codebook has been constructed using one sample vector drawn from a control image of each member of the test population, thus reflecting only the variation present within this test population. To obtain a reduction in data storage, the two most common VQ codebook generation techniques have been used to reduce the codebook dimensionality. In this study the test population contained 40 members and thus the initial codebooks (for each of the eight features) contained 40 vectors. A reduction to half this number has been chosen as a sufficiently demanding test of the codebook generation algorithms available. The task of the codebook generation technique is to perform the requisite dimensionality reduction while still containing the system's error rate within acceptable limits.

3.1 Kohonen's Self-Organising Feature Maps

Neural network clustering techniques have been employed in a variety of application areas, such as pattern recognition, optimisation and, notably, VQ codebook design for image compression [6, 7]. Kohonen developed his Self-Organising Feature Maps (KSOFM) in order to model the neural feature maps which are thought to form in the human brain [8]. KSOFMs result in a network where neighbouring output units have similar responses : *topological ordering* has occurred. This ordering can be used to reduce search requirements in VQ applications, though this is not an aspect of the KSOFM which has been exploited here. As a clustering algorithm, KSOFM allows a reduction in the dimensionality of the input vector space to a smaller number of reference vectors.

KSOFM defines a matrix (usually two dimensional) of output units, or neurons, each of which has a weight vector, w_j, associated with it. Input vectors from a training set are presented sequentially and the weight vectors are adjusted as described below. The weight vectors converge towards cluster centres after sufficient training time. The reason that topological ordering occurs is that each input vector adjusts not only one weight vector, but a *neighbourhood* of weight vectors.

The KSOFM algorithm is defined thus :

Step 1. Initialise all weight vectors to random values.

Step 2. Apply new input vector, $x(t)$.

Step 3. Calculate the Euclidean distance from $x(t)$ to all output nodes j

$$d_j = \sum_{i=0}^{N-1} (x_i(t) - w_{ij}(t))^2$$

where N is the dimensionality of the input vector.

Step 4. Select the output node with the smallest distance d_i and label it as the winning unit, j^*.

Step 5. Update weight vectors of all nodes in the matrix according to the equation

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t, \mathcal{D})\alpha(t)(x_i(t) - w_{ij}(t))$$

where $\eta(t, \mathcal{D})$ is the neighbourhood gain function, and $\alpha(t)$ is the adaption gain function.
$\eta(t, \mathcal{D})$ is a function which defines a neighbourhood on the matrix round the winning neuron, decreasing exponentially with distance $\mathcal{D}$ from the winning neuron, and which shrinks over time. $\alpha(t)$ decreases exponentially with time, and $0 \leq \alpha(0) \leq 1$.

Step 6. Repeat **Step 2** to **Step 5** until the entire training set has been presented e times. e is the number of *epochs*, which is set before training starts.

In this application the input vectors, x, come from the original feature codebooks containing 40 vectors, and the matrix was a 5×4 array of neurons producing a codebook of 20 vectors.

3.2 Linde-Buzo-Gray Algorithm

The Linde-Buzo-Gray (LBG) algorithm [9] is the most commonly used codebook design algorithm due to the fact that it was the earliest proposed method and consistently outperforms other methods in a variety of applications [10, 11]. It is an iterative technique which repeatedly moves codewords to cluster centroids in an effort to find a codebook which will display the lowest error when encoding the training data (*i.e.* a modified version of the K-Means clustering algorithm). The basic algorithm is as follows :

Step 1. Initialise the codebook.

Step 2. For each vector, x, in the training set, calculate the Euclidean distance from it to each vector v_j in the codebook.

$$d_j = \sum_{i=0}^{N-1} (x_i - v_{ij})^2$$

where N is the dimensionality of the vectors.
The minimum distance selects the closest vector, v_j^*, in the codebook. Assign x to the cluster around v_j^*.

Step 3. Replace each codeword with the centroid of the vectors in the training set that have been assigned to it. If any codewords are unused, they are discarded and replaced with new codewords which are more likely to be used in the next iteration. In this work, this has been done by taking the most commonly used codewords and splitting them to create two close copies of the original.

Step 4. If the total error in clustering the training data is still decreasing by a significant amount, return to **Step 2.** Otherwise, stop.

This algorithm should optimise the codebook so that the sum of the distances from each vector of the training set to its nearest codeword is a minimum. It is possible, however, that in certain conditions the algorithm will reach a local minimum rather than a global minimum. This can be seen to be true due to the fact that the final codebook will change if the initial codebook is different.
There are a number of recognised methods for generating an initial codebook, the most common of which is to populate the codebook with vectors chosen randomly from the training set. Another method is to use a codebook generated from other clustering techniques, such as the KSOFM technique described above, as the initial codebook. The method utilised here was to populate the codebook with 20 replicas of the centroid of the entire training set. In this way the codebook is gradually filled with useful codewords, as unused codewords are replaced in **Step 3**.

There are some similarities between the ways in which the two techniques outlined above perform dimensionality reduction. However, the codebooks produced can be significantly different. To illustrate this Figures 4 and 5 show the two 20 vector *face*[†] codebooks generated from the same original input of 40 vectors. It can be clearly seen that both codebooks contain composite faces formed by the merging of several of the input faces together, but that they are quite different from each other.

[†]There are parallel codebooks for each of the other seven features used in the recognition algorithm.

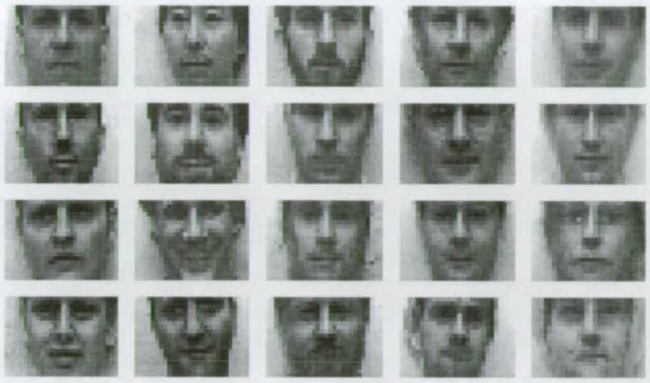


Figure 4: Codebook Generated using LBG.

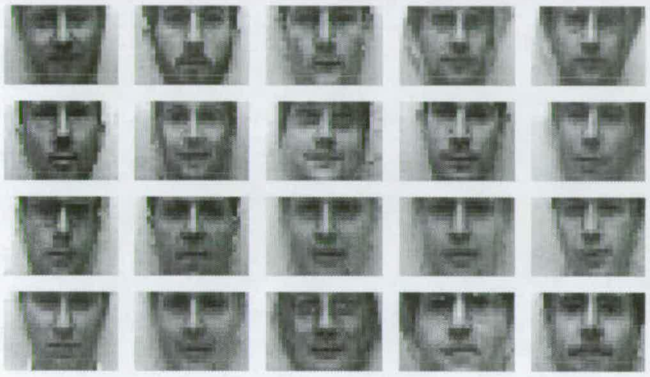


Figure 5: Codebook Generated using KSOFM.

4 Experimental Results

At present the algorithm for facial recognition is implemented as a suite of software programs. The system cannot yet function in real-time and thus the test images used are stored on disk for repetitive analysis. The images used were captured with a video camera under largely controlled conditions. However, the subjects were allowed to vary their expressions during the several days during which images were being captured. In this way, the results obtained for the system will be closer to a real-world implementation than some other, highly controlled, studies.

As mentioned earlier, a test population of 40 males was used, with ten images of each person used for training (*i.e.* the generation of the facial signature) and ten images kept for testing. The trial thus consisted of 400 test presentations. For each presentation, the output ranking of likelihood was recorded. From this data, and from knowing the correct response, it is possible to obtain a first place recognition rate (*i.e.* the proportion of tests in which the correct signature was selected as the most similar to the test stimulus). By analysing the output ordering of responses an average rank figure can also be obtained.

4.1 Dedicated codebook

To provide a benchmark, and to demonstrate the best possible performance, a set of dedicated codebooks were used. These codebooks were constructed using the same vectors as were used to train the other codebook generation algorithms. The feature histograms were thus 40 vectors wide for each of the eight features. For this experiment the average rank and the cumulative success rates of the first three output positions is given in Table 1.

Average Rank	1.37
1^{st} Place Recognition	88.5%
2^{nd} Place Recognition	92.75%
3^{rd} Place Recognition	95.5%

Table 1: Recognition rates for a 40 member population with 400 test presentations.

4.2 Dimensionality Reduction

Essentially, here, we are performing the same experiment as above, but using codebooks of half the size. Such a significant reduction would be expected to cause a substantial reduction in recognition performance unless, as described in section 3, the algorithms used to derive the new codebook entries reflect the initial characteristics of all 40 vectors. The experiment is thus a parallel comparison between the two different sets of codebook vectors when applied to the recognition function. Table 2 gives the relative performances between the two approaches.

	KSOFM	LBG
Average Rank	1.99	1.46
1^{st} Place Recognition	70.2%	83.3%
2^{nd} Place Recognition	81.5%	90.0%
3^{rd} Place Recognition	88.5%	94.7%

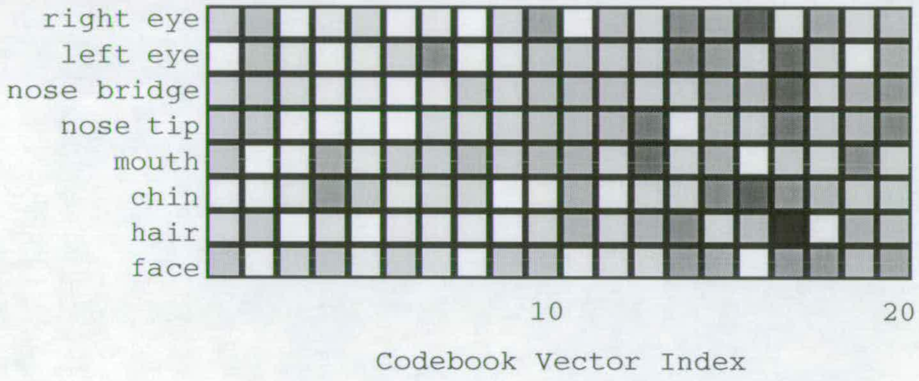
Table 2: Two methods of codebook design.

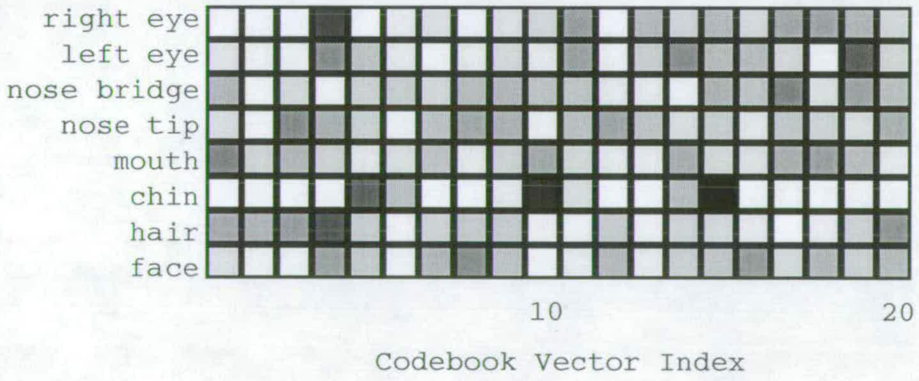
4.3 Discussion

If we consider the first experiment using a dedicated codebook, the performance results are very encouraging. The first place recognition rate of 88.5% does not yet rival the other biometric systems. However, the results reported here represent one of the very few significant population studies of a practical automatic face recognition system reported to date and, as such, give an important indication of the future viability of automatic face recognition.

Considering the dramatic reduction in codebook dimensionality, the second set of recognition results are remarkably good. The LBG approach performs significantly better than the KSOFM method. Its overall recognition rate is approaching that of the 40 vector system reported in section 4.1. To explain the variation in the performance of these two codebook generation algorithms more detailed consideration of their mechanics is required.

In general the codebook generation algorithms perform reduction by averaging the most similar vectors together, while still maintaining good coverage of the initial vector space. If the most similar vectors are also the most frequently used vectors, then the clustering process may reduce the good spread of vector choice required to maintain good differentiation when encoding the population. To investigate whether this is in fact the case feature histograms have been drawn up for the entire population.





Figures 4.3 and 4.3 illustrate the feature histograms for the LBG and K-SOFM algorithms respectively. The spread of vector choice is much more even for the LBG vector set. We believe this is the underlying factor which explains the relatively poor performance of the KSOFM approach. In effect, KSOFM has maintained *too good* coverage of the entire feature space at the expense of differentiation between the most common feature types.

5 Conclusions and Future Work

The results of the initial research into the use of VQ for automatic facial recognition are encouraging. This approach has been shown to work for facial recognition and undoubtedly has uses in other areas of image pattern recognition.

The dimensionality of the codebooks used by VQ based recognition has been identified as a potential limiting factor, however, this research has shown the important improvements which can be obtained in this area by using different codebook generation algorithms. However, we accept that further experimentation, with different populations, is required to validate the results reported here. It is further recommended that much larger populations are required to adequately test potential facial recognition systems.

References

[1] V Bruce and M Burton. Computer recognition of faces. In A W Young and H D Ellis, editors, *Handbook of Research of Face Processing*, pages 487–506. North-Holland, 1989.

[2] K Sutherland, D Renshaw, and P B Denyer. A novel automatic face recognition algorithm employing vector quantization. In *Digest of the IEE Colloquium on Facial Recognition and Storage*, London, January 1992.

[3] M A Fischler and R A Elschalger. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22:67–92, 1973.

[4] N M Nasrabadi and R A King. Image coding using vector quantization : A review. *IEEE Trans. on Comms.*, COM-36(8):957–971, August 1988.

[5] R M Gray. Vector quantization. *IEEE ASSP Mag.*, 1(2):4–29, April 1984.

[6] N M Nasrabadi and Y Feng. Vector quantization of images based upon the Kohonen self-organising feature maps. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN-88)*, pages 101–108, July 1988.

[7] T-C Lee and A M Peterson. Adaptive vector quantization using a self-development neural network. *IEEE J. on Selec. Areas in Commun.*, 8(8):1458–1471, October 1990.

[8] T Kohonen. Clustering, taxonomy, and topological maps of patterns. In *Proc. 6th Int. Conf. on Pattern Recognition*, pages 114–128. IEEE, October 1982.

[9] J Linde, A Buzo, and R M Gray. An algorithm for vector quantizer design. *IEEE Trans. on Comms.*, COM-28(1):84–95, January 1980.

[10] J D McAuliffe, L E Atlas, and C Rivera. A comparison of the LBG algorithm and Kohonen neural network paradigm for image vector quantization. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-90)*, pages 2293–2296. IEEE, April 1990.

[11] W Equitz. Fast algorithms for vector quantization picture coding. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP-87)*, pages 725–728. IEEE, April 1987.