LOW RATE IMAGE CODING USING VECTOR QUANTIZATION

Thesis by

Anamitra Makur

In Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1990

(Submitted October 10, 1989)

ACKNOWLEDGEMENT

It is quite impossible to properly acknowledge every one of those people with a kind heart, a sharp mind, and both, who helped me, knowingly or unknowingly, in the course of this research. Following are the names of only a few of those to whom I am indebted.

Professor Edward C. Posner is the foremost person I should mention. His immensely helpful advice was not, thanks to him, limited to mere research topics. I was introduced to the field of image coding by him, and his constant encouragement, as well as his profound insight, helped me get through the darker days.

Dr. Jerry E. Solomon and others at the MIPL facility of the Jet Propulsion Laboratory have not only let me use their well-equipped laboratory, they were patient enough to teach me how to use those apparatus. I am deeply indebted to them. Special thanks are due to Professor P. P. Vaidyanathan, whose discussions with me were very helpful.

I consider it my duty to extend my deepest gratitude to all of my friends, the graduate students of the Systems Group, Caltech. The help I received from them at every stage is invaluable. I specially thank Dr. Truong Q. Nguyen, Amir F. Atiya, and Dr. Eric E. Majani, among others, for their technical assistance.

While I was busy writing this thesis, my wife Anindita was more understanding and supportive than I deserved. Her contributions are in no way to be disregarded.

I also thank Pacific Bell for financing this research.

ABSTRACT

This thesis deals with the development and analysis of a computationally simple vector quantization image compression system for coding monochrome images at low bit rate. Vector quantization has been known to be an effective compression scheme when a low bit rate is desirable, but the intensive computation required in a vector quantization encoder has been a handicap in using it for low rate image coding. The present work shows that, without substantially increasing the coder complexity, it is indeed possible to achieve acceptable picture quality while attaining a high compression ratio.

Several modifications to the conventional vector quantization coder are proposed in the thesis. These modifications are shown to offer better subjective quality when compared to the basic coder. Distributed blocks are used instead of spatial blocks to construct the input vectors. A class of input-dependent weighted distortion functions is used to incorporate psychovisual characteristics in the distortion measure. Computationally simple filtering techniques are applied to further improve the decoded image quality. Finally, unique designs of the vector quantization coder using electronic neural networks are described, so that the coding delay is reduced considerably.

Except for the basics of the vector quantization described in the first chapter, each chapter is independent from the others because each chapter deals with a separate aspect of the coder. Therefore, each chapter beyond the first can be read separately.

CONTENTS

LIST OF FIGURES

Chapter I
INTRODUCTION

Almost from the time television images started to be transmitted, the quest for an efficient way to represent an image began. In image data compression, which is more popularly known as *image coding*, one is concerned with converting an analog picture to the smallest set of binary digits such that this set of binary digits can be used to reconstruct a replica of the original image good enough for human visual perception. The need for such a representation arises during storage and transmission of images. A digitized picture is a step towards this goal because it is a time-discrete, amplitude-discrete representation of an analog image. A large coding efficiency may be achievable for pictures that are to be viewed by humans, taking into account the properties of the human visual system and the statistical and picture-dependent properties of digitized pictures. The problem is different if human viewing were not the primary objective. The human vision aspect has thus influenced image transmission and coding since the beginning.

1.1 History and Overview of Image Coding

A typical image possibly has more redundancy than any other type of natural signal. In the early fifties the autocorrelation of an image was measured and a high degree of local correlation was observed. Many types of these observed redundancies can be exploited. Typically horizontal or vertical correlation of a natural image is very high for adjacent *pixels* (or, *pels*, i.e., the samples of the digitized image). This implies that the overwhelming portion of spatial power in a video signal is in the lower frequencies, although the high-frequency part is responsible for the ultimate quality of an image. Most coding techniques take advantage of this fact. A large number of such techniques have been reported over almost four decades. Perhaps the vast diversity among these schemes is due to the fact that there could be a lot of difference in the coding objectives.

Image compression is not limited to the pixel-sampled images. It has been attempted on analog TV signals, too [1]. Our discussion, however, is concerned with the *digital* image compression/coding only. The compression techniques tend to vary for different types of images, such as gray-level and color, two-tone, printed material, or line-drawing images. We shall limit our discussion only to gray-level and color images.

Any data compression scheme can be broadly categorized into two classes, *lossless* and *lossy.* These classes are also differentiated as reversible and irreversible, or as noiseless and noisy coding. A lossless coder operating on a digital image achieves compression without losing any of the digital source information, hence the decoded image is no different from the original one. A lossy coder reduces data rate by sacrificing some information which is, or is thought to be, of not much relevance to the viewer. As a result, the decoded image from a lossy coder shows some *coding noise*, or *error*, when compared to the original image.

Lossless image coders exploit the strong spatial source correlation of an image. In general, truly lossless techniques do not provide much in the way of compression. Run-length coding, arithmetic coding, and Huffman coding are a few techniques followed in lossless image coders. Some such coders are described in [2–5].

Though there exists a large number of lossy image coders in the literature, almost all of them could be classified into one of the following categories: *predictive*, *block*, and *hybrid* coding. These will be discussed now.

In a predictive image coder, a prediction for the value of the next scanned pixel to be encoded is made based on the past encoded pixels, some of which may not be on the same scan line. The difference between this prediction and the actual value is encoded and sent to the decoder. The entropy of the adjacent-pixel difference signal of a 6 bit gray-level picture is found to be only about 2.6 bits per pixel. Therefore, traditional coding schemes such as delta modulation [6–8] and differential pulse code modulation (DPCM) [9–13], in simple and adaptive forms, are often used on imagery. Improved DPCM coders use more than one past pixel for prediction. Because the local statistics of a typical image vary to quite an extent, the need for a predictor that is adaptive was soon realized [14–20]. In case of video data, the prediction operates both in spatial (within a frame, or *intraframe*) and in temporal (between frames, or *interframe*) domains [21–26].

The prediction difference signal for the above-mentioned class of coders is quantized to achieve digital transmission or storage with compression. Several methods for optimizing this quantizer, or making it adaptive to local image statistics, have been suggested [27–31]. Keeping in mind the human audience, psychovisual properties have also been used to design these quantizers [32–37].

Linear prediction is used in some image coders. The encoders do not send all pixel values, instead the missing ones are computed by the decoder using interpolation. Some such coders for still and video images are described in [38–41]. These coders often use DPCM along with interpolation to achieve higher compression.

A strong similarity exists between the successive frames of a video signal. In fact, in a new frame, pixels from only a few areas change. Therefore, temporal prediction could be modified to become simply detecting and coding of the moving areas of a frame. This simple yet efficient technique, called *conditional replenishment*, was developed by Mounts in 1969 [42]. This technique produces variable rates for different amounts of motion. Improved designs have been offered characterizing buffer control, efficient addressing of the changed area, or spatial prediction [43–48].

Some interframe coders use a more sophisticated temporal prediction by estimating and compensating for the changes due to motion [49–62]. The encoder of such a *motion compensation* scheme determines the magnitude and direction of motion, usually simplified as a piecewise linear translation, and sends them to the decoder. Most improvements are done on these coders either by implementing an efficient search algorithm to determine the motion vector, or by combining motion estimation with other kinds of predictions and making the predictor adaptive to local image characteristics.

The next type of image coding is called block coding because of the fact that a section of neighboring pixels, or a block, is encoded together instead of encoding individual pixels separately. The strong two-dimensional correlation between the pixels of a block is thus exploited. Block coders can achieve impressive coding gains. In general, block coders perform comparatively better when the required compression ratio is high. However, these coders usually are more complex than predictive coders. Some block coding techniques will now be mentioned.

One very popular block coding technique is known as *transform coding.* The input block to a transform coder is a two-dimensional spatial block of square shape. The intensity values of this block are treated as a matrix with correlated coefficients. The encoder maps this matrix to a transformed domain in which the coefficients are expected to have reduced correlation. The transformed coefficients then require fewer bits to be transmitted than the original intensity values do. The decoder performs an inverse transform to return to the intensity domain.

A number of orthogonal or near-orthogonal transforms, such as Karhunen-

Loeve, Cosine, Hadamard, Sine, Fourier, Slant, Haar, etc., have been attempted on still images [63–78]. The discrete cosine transform, or DCT, seems to be the most popular among these transforms. Improved transform coders use adaptive bit assignment for transmitting the transform coefficients. Interframe transform coders use three-dimensional input blocks (spatial and temporal) while performing similar operations [79–81].

Another widely investigated class of block coders perform *vector quantization* to encode the input blocks [82, 83]. Each input block to a vector quantization encoder is treated as a multidimensional vector for quantization purposes. The large number of possible input vectors is mapped to only a few probable quantization regions. Labeling of the region to which the input block was assigned could be done using very few bits. The decoder substitutes the input block with the typical block of the assigned region. This type of coder will be discussed in detail in the next section.

The above two types of block coders have been combined in an attempt to keep the advantages of both. In transform-domain vector quantization coders, quantization is done on the transform coefficients instead of the intensity values [84–87]. These coders have added complexity while offering higher compression ratios.

Various other block coders have also been published. *Block truncation coding* encodes blocks using a bi-level non-parametric quantizer that can adapt to local image properties. Some parametric values of the block are also transmitted to preserve the lower-order moments of the square block. As has transform coding, two- and three-dimensional blocks have been encoded using block truncation coding [88–91]. *Singular value decomposition* has been applied for spatial energy compaction of image blocks in a way similar to orthogonal transform coding [92, 93].

Predictive coding is easy to implement, while block coding is robust and gives a lower bit rate. In an attempt to combine these advantages, a class of hybrid coders has evolved. In some hybrid encoders, a DPCM encoder follows the transform encoder to exploit the similarity between the spatial blocks [94–101]. DPCM can also be performed before the transform coding [101–103]. Vector quantization is used in conjunction with DPCM, too [104]. Conditional replenishment can be used with any intraframe block coder to design an efficient interframe coder [105]. Lastly, motion compensation is also done in transform domain [101, 106–108] and with vector quantization [109].

Apart from the three general types of coders described here, a few other image coding techniques have also been reported. Attempts have been made to consider source coding and channel coding at the same time. Combined *source-channel coding* has the advantage of some error correction capability within a low data rate. These coders usually are superior to others in the presence of transmission noise. Tree encoding [110–112] is one example of such a scheme. The effect of adding some error correction without increasing the bit rate has been investigated in [113–115] for DPCM and DCT coders. *Sub-band coding* techniques are also used for image compression. These coders split the spatial frequency band of the image signal and code each band separately using an appropriate coder and rate. Some such coders are described in [116–119]. Note that it is possible to classify a source-channel coder, or a sub-band coder, into one of the three broad groups discussed earlier, depending on which particular coding scheme is being used. The *fractal compression* scheme uses a completely different concept by tolerating encoding distortion not only in the intensity values of the pixel, but also in the spatial position of the pixel itself. Fractal image coding is claimed to have achieved very high compression ratios at the expense of enormous computing [120].

Tutorial and review papers have been published on general or specific image coding techniques from time to time. Refer to [121–127] for some of these papers.

1.2 Vector Quantization

Vector quantization (VQ) is a source coding technique that has existed in theory for a long time, and has been extensively used for speech for some years, but for images only more recently. A simple consequence of Shannon's rate distortion theory is that better performance can always be theoretically achieved by coding vectors instead of scalars, even if the data source is memoryless. Therefore pixel-wise compression schemes are, in a Shannon sense, inherently sub-optimal, because quantization is accomplished only on scalars. This fact naturally leads to designing a VQ coder where vectors instead of scalars are quantized to achieve better compression. Input samples from the source, typically speech or image, are bunched together as *input vectors*. A pre-computed set of representing vectors, or a *codebook*, is then searched through to find out the member (a *codeword*) which, according to some criterion, best matches the input vector to be encoded. For the purpose of deciding the best match, the closeness between the input vector and any codeword is measured by a vector *distortion function* associated with the VQ coder. The index, or a binary *label*, identifying the selected best-matching codeword is then assigned to the input vector and transmitted. The decoder uses this label to address the same codebook, each entry of which contains a precise digital representation of the corresponding codeword. The job of the decoder is to simply substitute the label by the codeword.

The *dimensionality* of a VQ image coder refers to the number of dimensions, or pixels, an input vector has. For such a K-dimensional VQ image coder, the source image is partitioned into typically rectangular blocks of $p \times q$ pixels (usually $p = q$). The intensity values of such a block is put into a typically row-first single-file ordering, regarding them as a K-dimensional input vector

$$X = (x_1, x_2, \ldots, x_K), \tag{1.1}$$

where $K = p \times q$ and x_k's are the individual pixel values. This VQ coder has a codebook $\mathcal{C}$ consisting of N pre-computed codewords,

$$\mathcal{C} = \{C^{(1)}, C^{(2)}, \ldots, C^{(N)}\}, \tag{1.2}$$

where N is the cardinality of $\mathcal{C}$, usually some integer power of 2. Each of these N codewords is also K-dimensional: $C^{(n)} = (c_1^{(n)}, c_2^{(n)}, \ldots, c_K^{(n)})$. Then, for an input vector X, the encoder determines $C^{(m)}$ such that

$$d(X, C^{(m)}) \leq d(X, C^{(n)}) \qquad \text{for } 1 \leq n \leq N. \tag{1.3}$$

Here $d(X, C)$ is the distortion function associated with the VQ coder. The distortion function almost universally used in image coding has been the mean square error (MSE) distortion,

$$d(X, C) = \frac{1}{K} \sum_{k=1}^{K} (x_k - c_k)^2. \tag{1.4}$$

In case of MSE distortion, the closeness is thus measured by the Euclidean distance between the two points in a K-dimensional Cartesian coordinate system.

To transmit the index m to the decoder, log$N$ bits are required. The decoder replaces m by $C^{(m)}$. Therefore, the coder achieves a compression ratio of $(Ks/\log N)$, where s is the number of bits per pixel (typically 8) in the original input. The minimum distortion, as computed by the encoder, is the resulting quantization noise in the decoded data.

The concept of quantizing vectors was applied to speech coding before it was extended to images. Quantization in multidimensions can, however, be found in some earlier image coding schemes. In one such coder [128], though VQ has not been explicitly mentioned, a vector codebook consisting of some typical codewords was searched to find the best match for the input block. Some early VQ image coders have already been referred to in the previous section. A detailed discussion of vector quantization is given in [129]. A brief review paper on VQ image coders appears in [130].

Some basic modifications of the VQ coder will now be mentioned. The dimensionality, and consequently the codebook size, of a VQ image coder is usually higher than those of a VQ speech coder. Therefore, image coding faces the disadvantages of a large codebook, which are large storage requirement and longer search time. In order to reduce the codebook size, *mean residual VQ* is used in some coders [131–133]. In such a coder, the mean value of the vector is subtracted from the input vector before encoding. The mean value is scalar quantized and sent along with the label. In order not to have a very large codebook, the block size of a VQ coder is kept small. This, however, does not allow one to exploit all the redundancy of an image. *Finite state VQ* uses the information from the previous blocks to decide which sub-codebook will most likely have the best match, thus exploiting some of the spatial [134] or temporal [133, 135] inter-block redundancy. Finite state VQ has also been used adaptively [136].

The MSE distortion function does not represent human viewing discomfort too well. Therefore, human visual properties have been incorporated in VQ image coders to achieve better subjective quality [137, 138]. The basic VQ coder puts equal emphasis in encoding uniform blocks and high-detailed blocks. The high-detailed blocks are not so well encoded as a result. To remedy this problem, a variable block size generating a variable bit rate has been used in some coders [139, 140]. An alternative approach is taken in [141, 142] by classifying the vectors into different 'shade' and 'edge' classes according to inner details, and by designing separate codebooks for these classes. Then the final codebook is a conglomeration of these codebooks with a desired emphasis on each class, such that a more balanced encoding would result. For the same purpose, multistage VQ image coders have also been used [140, 143].

Encoding of color images using a VQ technique offers some choices. Because the three color elements, whether composite or component, are highly correlated, vector quantization could be performed on three-dimensional input vectors containing the three elements of an individual pixel [144, 145]. This approach is quite common in the encoding of multispectral imagery, although the spatial correlation is disregarded here. A more popular approach is to use three independent spatial VQ coders for the three color elements [131, 132, 138, 146], which, however, fails to utilize the redundancy along the color axis. The best results have been achieved by considering a three-dimensional block (two spatial axes, one color axis) as the input vector to the VQ coder [146–148].

Encoding of video images with a VQ coder has been done by using some interframe techniques along with the basic VQ structure. *Label replenishment* is used as an extension of conditional replenishment in some schemes [105, 132, 149]. A label replenishment coder encodes each input block and compares the label with the label of the previous temporal block. The present label is transmitted only if the two labels differ. Motion compensation is also used in conjunction with vector quantization [109, 133, 150].

For a VQ image coder to perform efficiently, the codebook has to represent the image data very well. It has been observed that codebooks generated from some types of image data typically don't perform well on unforeseen images. Thus, *robustness* of a codebook towards changing image statistics is very desirable in a VQ. Some schemes use an adaptive codebook by periodically updating the codebook, i.e., replacing the relatively unused codewords by splitting the more likely clusters [105, 132, 149, 151]. Re-encoding the encoding error of the initial VQ by a second-stage VQ has been suggested in [104] to make the coder more robust.

Generating a good codebook requires a huge amount of computation. The amount of computation could be reduced by using less data, but it would also result in a less robust codebook. In order to circumvent this computation burden while not sacrificing robustness, some fast codebook generation algorithms have been suggested [152, 153]. Attempt has also been made to design a better codebook without increasing the amount of computation [154].

Once the codebook is generated, the encoding search still requires a lot of computing. The basic VQ algorithm performs a *full search* through the codebook (FSVQ), which is computationally intense. In [155], a faster search algorithm has been suggested which is still optimal. *Tree search* attempts to utilize some characteristics of the codebook to reduce the search time [153, 156]. Alternatively, various structured codebooks have been designed so that encoding becomes much simpler. One has to note that these structured codebooks are usually sub-optimal. Binary (or other) *tree-structured* VQ, first mentioned in [157], takes only logarithmic encoding time when compared to FSVQ. Similar savings in encoding computation is achieved by *lattice* VQ [158], which is used in some schemes [104]. *Pyramid* VQ is another example of such a structured VQ [159].

1.3 Brief Outline of the Chapters

In Chapter II we suggest a novel approach to extract input vectors for a vector quantization coder. We point out in Section 2.1 the motivation for doing so, and the advantages and disadvantages of this distributed-block VQ coder. For almost no added complexity, our coder achieves a lower bit rate for same distortion when compared to a similar conventional VQ image coder with small dimensionality. Moreover, the new coder offers more flexibility in altering bit rate, and needs less encoding computation than the standard coder. The remaining part of Section 2.1 describes the scheme in general terms. In Section 2.2 we show that the distributed-block VQ coder outperforms, in theory, the basic VQ coder in some cases for large enough codebook. Our simulation results in Section 2.3 show that a specific example of this coder, applied to low rate image coding, performs better than the basic VQ coder.

In order to improve the decoded image quality, we make some effective yet simple-to-implement modifications to the VQ image coder in Chapter III. A general class of distortion measure is proposed in Section 3.1. Our proposed distortion function introduces activity classes and a geometric emphasis. In accordance with the activity, or inner detail, of an input vector, each vector is assigned to an activity class with a penalty factor for the quantization distortion. In Section 3.2 we investigate the use of activity classes in VQ image coding and find that this scheme indeed improves the subjective quality of the decoded pictures in our simulations. The geometric emphasis part is used to selectively reduce the distortion at the block boundaries. In Section 3.3 we show some more results for a VQ coder with selective emphasis, achieving lower quantization distortion and better subjective quality when compared to the basic VQ coder results.

Chapter IV deals with filtering techniques for a block image coder, with special emphasis on a VQ coder. A brief introduction to incorporating filtering techniques in image coders is given in Section 4.1. By exploiting the spectral characteristics of the quantization error, we design and compare some simple spatial filter in Section 4.2 to smooth away the block discontinuities in the decoded image. In Section 4.3 we suggest a scheme using prefiltering and postfiltering to improve the decoded image quality, and show that our scheme performs better than the regular preprocessing scheme.

Chapter V describes two schemes we have designed to implement a VQ image coder using an electronic neural network. An introductory overview of using neural networks in vector quantization is given in Section 5.1. Our first scheme, described in Section 5.2, uses simple networks to realize a binary tree-structured VQ encoder, and will have a very short encoding time. The second scheme, described in Section 5.3, realizes the full-search VQ encoder using fewer synapses, but suffers a longer encoding delay due to the presence of feedback.

The thesis is summarized and conclusions drawn in Chapter VI. The VQ image coder has until now been perceived to be somewhat impractical due to its excessive encoding computation. However, from the results of our work, implementation of an affordable real-time low rate video coder using vector quantization appears plausible even with today's technology. We show that the subjective quality of the decoded picture can be enhanced without increasing the encoder computation. Also, electronic neural networks appear to be a good alternative for implementing a VQ image coder, especially when the system requires a small encoding delay.

Chapter II
USE OF THE DISTRIBUTED BLOCKS

In a vector quantization coder, the input vectors are usually formed from the source using spatial pixel blocks for images, or a set of neighboring samples (temporal blocks) for speech. In this chapter we present a different technique for extracting vectors from the source. The new vector quantization coder, as described in Section 2.1, uses distributed blocks as input vectors. The clustering property required during codebook generation from these distributed vectors is expected to be less favorable than for the conventional spatial vectors. On the other hand, the quantization noise with our technique is distributed, which is more acceptable for human viewing. In addition, owing to the correlation between the distributed vectors, the encoding search and the bit rate for the new scheme will be smaller than for the conventional coder. In Section 2.2, this coder is shown to asymptotically achieve lower rate for small dimensionality over the general coder producing the same SNR. Not only does the new technique not increase the coder complexity, it also is much more flexible for rate versus SNR compromises than other coders. Some experimental results are given in Section 2.3 comparing our technique with the spatial vector quantization coder for low rate image coding.

2.1 Distributed-Block Vector Quantization

Traditionally the input vectors for a VQ encoder have been extracted from the source by choosing K samples (K = dimensionality of the coder) that are immediate neighbors of each other (in a line for one-dimensional sources, in a square or rectangular block for two-dimensional images). This is only natural since immediate neighbors show the highest correlation, therefore the vectors tend to cluster together, which is desirable in a VQ coder. In the *distributed-block VQ* scheme the source sequence is decimated by a factor of d before constructing the K-dimensional vectors. Note that the normal spatial-block vector is a special case where $d = 1$. For $d > 1$, the source is split into d decimated sequences in a form similar to sub-band coding [160]. No information is lost in the process because each decimator receives a different delayed version of the source sequence. Each of these sub-sequences is encoded and decoded by a separate VQ coder. The decoded sequences at the receiver are put together to get back the source sequence with possible quantization error. Figure 2.1 shows the schematic diagram of the coder, features of which will now be described.

Because these d decimated sequences have similar probability distributions, it is necessary to design and store only one codebook. Coders of all channels use this same codebook. The codebook, $\mathcal{C}$ in (1.2), is assumed to be ordered in the following fashion:

$$f(C^{(1)}) \leq f(C^{(2)}) \leq \ldots \leq f(C^{(N)}), \qquad (2.1)$$

where $f(\cdot)$ is a similarity function on the codewords; codewords similar in intensity values and in patterns should have closer $f(C)$, and vice versa. The objective of this ordering is to have similar codewords in neighboring regions within the codebook.

Ideally, the similarity function should represent the similarity for human viewing. In practice, however, one could use some distortion measure, or correlation, between two codewords to determine their similarity and place them accordingly in the codebook. However, this may not result in a unique ordering. A simpler way

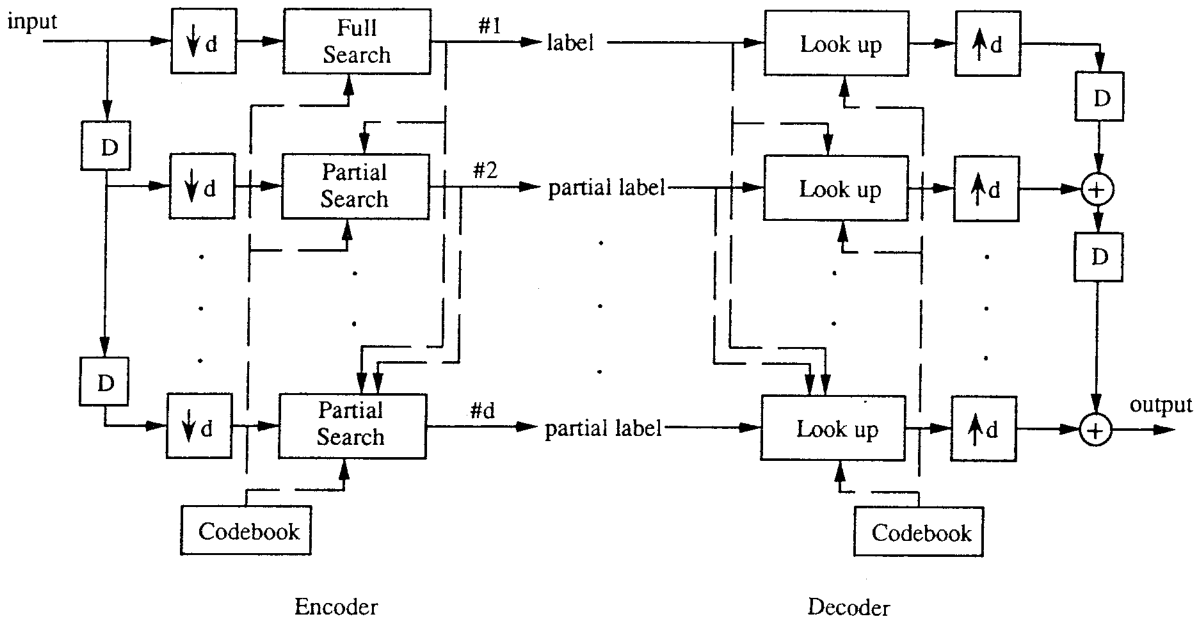


Figure 2.1: Distributed-Block Vector Quantization Coder

of sorting the codebook is using the average intensity,

$$f(C) = \frac{1}{K}\sum_{k=1}^{K} c_k, \tag{2.2}$$

which, however, is pattern-invariant. There is no penalty in making the similarity function computationally complex, because this part of the computation is to be done only initially and is not required in the encoding process.

Because the d simultaneous input vectors to the coder are interlaced, they are much correlated. The need to exploit spatial redundancy over a larger region than just the block size has been recognized before [134]. A finite state VQ, for example, utilizes the correlation between the adjacent spatial vectors. The correlation between the distributed vectors is, however, much stronger than the correlation between vectors in case of the traditional scheme. This correlation is exploited in the coder to reduce both the search time and the bit rate. The individual VQ encoders of the d channels do not exactly operate in parallel. Instead, the coding is done in sequence through the channels, and each channel uses the information from other channels above it in the coding process, conducting only partial search. The partial search for the i-th channel is done over some localized region within the codebook,

$$\{C^{(b_i)}, C^{(b_i+1)}, \ldots, C^{(e_i)}\}, \tag{2.3}$$

where b_i and e_i denotes the beginning and the end of this search region.

To begin with, full search is performed in the initial (topmost) channel, which means $b_1 = 1$ and $e_1 = N$. Once this search is completed, the index, $m^{(1)}$, is known, where the superscript indicates the channel number. The next channel then performs its search through the part of the codebook localized around $m^{(1)}$,

$$b_2 = b_2(m^{(1)}) \qquad \text{and} \qquad e_2 = e_2(m^{(1)}). \tag{2.4}$$

The functions $b_i(\cdot)$ and $e_i(\cdot)$ specify the range of the partial search where the closest codeword for the input vector of the i-th channel will most likely lie, and could be of the form

$$b_i(m) = m - \frac{R_i}{2} \qquad \text{and} \qquad e_i(m) = m + \frac{R_i}{2} - 1, \tag{2.5}$$

where R_i is the partial range for the i-th channel. Typically R_i will be much less than N for $i > 1$, therefore the partial search will result in a partial label, $m^{(i)}$, which needs fewer bits than $m^{(1)}$ to be specified. Consequent channels will further reduce their search area using information from all the labels computed before, for example

$$b_3 = b_3(m^{(1)}, m^{(2)}) \tag{2.6}$$

and so on. In order to keep the coder fixed-rate, these functions should be such that R_i remains the same in any case.

Though it is likely that the codeword with the least distortion for some input vector will lie within the partial range, with a certain probability it will not belong to the search neighborhood. In that case the selected codeword, though optimal within the partial range, will be sub-optimal when compared to full search. An appropriately chosen similarity function will result in a small probability of sub-optimal decision even for a fractional search area. The decoder constructs the labels from the received partial labels using information already available. Therefore, no overhead is required.

The bit rate of the coder is given by

$$\frac{1}{Kd}\sum_{i=1}^{d} \lceil \log(e_i - b_i + 1) \rceil \qquad \text{bits per sample.} \tag{2.7}$$

This rate will be smaller than the bit rate for a full-search VQ coder, $\lceil \log N \rceil / K$, for any reasonable choice of search ranges. The average number of times the distortion function has to be computed for one K-dimensional vector is

$$\frac{1}{d}\sum_{i=1}^{d}(e_i - b_i + 1). \tag{2.8}$$

For a full-search coder the corresponding number is N. Thus, the encoding computation is substantially less for the distributed-block scheme. The storage requirement remains the same in our scheme. Because of the sequential encoding of the channels, it is possible to implement the coder using same amount of hardware as the full-search VQ, which is not obvious from the diagram. The complexity of this coder is marginally higher than that of the conventional coder due to the variable partial search range, but this is negligible compared to the full coder complexity.

Consider a spatial vector quantization coder with dimensionality K and N codewords. This coder has a compression ratio of $(Ks/\log N)$, where s is the number of bits per sample (or, pixel) of the source, and 'log' refers to logarithm base 2. Some applications of this type of coder demand the bit rate, i.e., the compression ratio, to be variable while compromising the quantization distortion. In general a larger block dimension results in a higher compression ratio for a VQ coder. However, the computational burden increases linearly with the number of dimensions. It is easy to realize that once implemented, the dimensionality K of a coder cannot be altered trivially. For fixed K the rate could be reduced while sacrificing quality by reducing N. A factor of 2 reduction in N will decrease the bit rate by $\frac{1}{K}$ bit per sample. In case of the tree-structured VQ, this reduction of codebook size amounts to reduction in depth of the tree, and could be achieved easily. However, for other VQ coders, this is not at all trivial. If a pre-computed fixed codebook is to be used, for most typical sources there is a need to adapt the coder to source statistics. These facts motivate one to look for a coder which is capable of changing the rate easily at smaller steps even for fixed codebooks. The distributed-block scheme not only fulfills this goal, computer simulations show images with better quality using this scheme compared to the conventional VQ coder.

The distribution of blocks has a desirable effect in case of image coding. Block coding such as VQ at low rate results in *blockiness*, or visibly annoying blocks of quantization noise, in the decoded picture. In the new coder, as d increases, an encoded block appears more and more diffused in the decoded image, and because of the integrating effect of human eye the blockiness becomes less annoying. However, as d increases, the correlation between the components within a vector becomes less. As a result the clusters become more random, and larger codebook size is required to achieve the same quality as before. Thus, the decoded image quality is affected both positively and negatively for increasing d. The optimum value of d seems to depend mostly on the type of source. As our results will suggest, for low rate coding of gray-level images, $d = 2$ performs better than $d = 1$.

2.2 Performance Analysis

The following section considers a theoretical approach to the distributed-block vector quantization (DVQ) coder performance when compared to the convectional scheme (VQ).

Consider the source to be one-dimensional, emitting real numbers $x_1, x_2, \ldots,$ $x_i, \ldots$ where $-\infty < x_i < \infty$. For simplification we will assume these outcomes to form a Markov chain. Thus,

$$p(x_i/x_{i-1}, x_{i-2}, \ldots) = p(x_i/x_{i-1}). \tag{2.9}$$

We choose the symmetric Gaussian function to be this differential probability density,

$$p(x_i/x_{i-1}) = g_\sigma(x_i - x_{i-1}), \tag{2.10a}$$

where

$$g_\sigma(x) \stackrel{\triangle}{=} \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}, \tag{2.10b}$$

the normal density with zero mean. Assume the probability density of the first element, x_1, to be $q(x_1)$.

A conventional VQ coder of dimensionality K and with N codewords is applied to this source. The encoder takes K-tuples from the source to form input vectors of the form $X = (x_1, x_2, \ldots, x_K)$. The joint probability density function for X is

$$p(X) = p(x_1, x_2, \ldots, x_K). \tag{2.11}$$

No known result exists for the average distortion for arbitrary N; however, the minimum achievable average distortion for such a block quantizer for asymptotically large N is known [161]. If D_{VQ} denotes this minimum average distortion, then

$$D_{VQ} = \frac{C(K,r)}{N^{r/K}} \parallel p(X) \parallel_{\frac{K}{K+r}}, \tag{2.12a}$$

where the distortion measure is the l_r-norm, and

$$\parallel p(X) \parallel_\alpha \triangleq \left[\int [p(X)]^\alpha dX \right]^{\frac{1}{\alpha}}. \tag{2.12b}$$

Here $C(K, r)$ is a function independent of the source statistics, and only the upper and lower bounds are known for $K \geq 3$. The joint density, $p(X)$ in (2.11), could be expressed as

$$\begin{aligned} p(X) &= p(x_1)p(x_2/x_1)p(x_3/x_1,x_2)\ldots p(x_K/x_1,x_2,\ldots,x_{K-1}) \\ &= p(x_1)p(x_2/x_1)p(x_3/x_2)\ldots p(x_K/x_{K-1}) \qquad \text{[from (2.9)]} \\ &= q(x_1)g_\sigma(x_2-x_1)g_\sigma(x_3-x_2)\ldots g_\sigma(x_K-x_{K-1}) \quad \text{[from (2.10)]}. \end{aligned} \tag{2.13}$$

Now we will consider the distributed-block scheme. For some integer $d \geq 1$, every d-th outcome from the source is chosen to construct vectors of the form

$$Y = (y_1, y_2, \ldots, y_K) = (x_1, x_{d+1}, \ldots, x_{(K-1)d+1}). \tag{2.14}$$

Here $p(y_1) = p(x_1) = q(y_1)$, and

$$\begin{aligned} p(y_{i+1}/y_i) &= p(x_{i+d}/x_i) \\ &= \int \cdots \int p(x_{i+d}/x_{i+d-1}) \ldots p(x_{i+1}/x_i) dx_{i+d-1} \ldots dx_{i+1} \\ &= \int \cdots \int g_\sigma(x_{i+d} - x_{i+d-1}) \ldots g_\sigma(x_{i+1} - x_i) dx_{i+d-1} \ldots dx_{i+1} \\ &= \underbrace{g_\sigma * g_\sigma * \ldots * g_\sigma}_{d \text{ times}}(y_{i+1} - y_i) \end{aligned} \tag{2.15}$$

where $*$ means convolution. But $g_\sigma * g_\sigma * \ldots * g_\sigma = g_\xi$, where g_ξ is the normal density with zero mean and variance $\xi^2 = d\sigma^2$. Therefore, Y is a vector having a similar density function to that of X, except that the functions $g_\sigma(\cdot)$ are to be replaced by $g_\xi(\cdot)$. This is expected because the correlation between data points are reduced in a distributed-block VQ. Therefore, from (2.14) and (2.15),

$$p(Y) = q(y_1)g_\xi(y_2 - y_1)g_\xi(y_3 - y_2) \ldots g_\xi(y_K - y_{K-1}). \tag{2.16}$$

The minimum average distortion for this coder is given by

$$D_{DVQ} = \frac{C(K, r)}{N^{r/K}} \parallel p(Y) \parallel_{\frac{K}{K+r}} . \tag{2.17}$$

Therefore, keeping all other parameters the same, the ratio of the average distortion for a VQ to a DVQ coder in (2.12) and (2.17) turns out to be

$$\frac{D_{VQ}}{D_{DVQ}} = \frac{\parallel p(X) \parallel_\alpha}{\parallel p(Y) \parallel_\alpha}, \tag{2.18}$$

where $\alpha = K/(K + r)$, and $r = 2$ because we are using MSE distortion. Now from (2.13),

$$\begin{aligned} \int p^\alpha(X)dX &= \int \cdots \int q^\alpha(x_1)g_\sigma^\alpha(x_2 - x_1) \ldots g_\sigma^\alpha(x_K - x_{K-1})dx_1 \ldots dx_K \\ &= \int f(x_1, \ldots, x_{K-1})dx_1 \ldots dx_{K-1} \int_{-\infty}^{\infty} g_\sigma^\alpha(x_K - x_{K-1})dx_K \\ &= C_\sigma \int f(x_1, \ldots, x_{K-1})dx_1 \ldots dx_{K-1} \end{aligned} \tag{2.19a}$$

where

$$C_\sigma = (2\pi\sigma^2)^{\frac{1-\alpha}{2}}\alpha^{-\frac{1}{2}}. \tag{2.19b}$$

Proceeding this way, one gets

$$\int p^\alpha(X)dX = C'_q C_\sigma^{K-1}, \tag{2.19c}$$

where

$$C'_q = \int_{-\infty}^{\infty} q^\alpha(x)dx, \tag{2.19d}$$

and

$$\int p^\alpha(Y)dY = C'_q C_\xi^{K-1} \tag{2.20}$$

in a similar fashion. Then from (2.18), (2.19), and (2.20),

$$\begin{aligned}\frac{D_{VQ}}{D_{DVQ}} &= \left(\frac{C_\sigma^{K-1}}{C_\xi^{K-1}}\right)^{\frac{1}{\alpha}} \\ &= d^{\frac{1-K}{K}}.\end{aligned} \tag{2.21}$$

Note that for $K \gg 1$, the ratio approaches $\frac{1}{d}$. This means the DVQ scheme results in a factor of d more average distortion than a comparable VQ scheme. This is expected since use of distributed blocks makes the vectors effectively less correlated. However, one should keep in mind that MSE distortion is not a true measure of human visual discomfort, and that the quantization error for the DVQ coder is more distributed, as opposed to localized errors for the VQ coder. Subjective quality of a distributed-block image might be almost as good with a substantial saving in bit rate.

Because the d vectors of a DVQ encoder are highly correlated among themselves, the bit rate of the DVQ scheme is less than that of a similar VQ coder. We will now compute this bit rate advantage of a DVQ coder over a VQ counterpart. Let us rename the outcomes of our Markov chain source as

$$y_1^{(1)}, y_1^{(2)}, .., y_1^{(d)}, y_2^{(1)}, y_2^{(2)}, .., y_2^{(d)}, \ldots, y_K^{(1)}, y_K^{(2)}, .., y_K^{(d)}, \ldots \tag{2.22a}$$

and the DVQ input vectors of (2.14) as

$$
\begin{aligned}
Y^{(1)} &= (y_1^{(1)}, y_2^{(1)}, .., y_K^{(1)}) \\
&\vdots \\
Y^{(d)} &= (y_1^{(d)}, y_2^{(d)}, .., y_K^{(d)}).
\end{aligned} \tag{2.22b}
$$

The joint density function in (2.16) could also be expressed as

$$
p(Y) = q(y_1)g_\xi(\Delta y_2)g_\xi(\Delta y_3)\dots g_\xi(\Delta y_K), \tag{2.23}
$$

where $\Delta y_i = y_i - y_{i-1}$. Then, for $d \geq 2$, $i < d$,

$$
\begin{aligned}
&p(Y^{(i+1)}/Y^{(i)}) \\
&\quad = p(y_1^{(i+1)}, y_2^{(i+1)}, .., y_K^{(i+1)}/y_1^{(i)}, y_2^{(i)}, .., y_K^{(i)}) \\
&\quad = p(y_1^{(i+1)}/Y^{(i)})p(y_2^{(i+1)}/y_1^{(i+1)}, Y^{(i)})\dots p(y_K^{(i+1)}/y_1^{(i+1)}, .., y_{K-1}^{(i+1)}, Y^{(i)}) \\
&\quad = p(y_1^{(i+1)}/y_1^{(i)})p(y_2^{(i+1)}/y_2^{(i)})\dots p(y_K^{(i+1)}/y_K^{(i)}) \qquad \text{[from (2.9),(2.22)]} \\
&\quad = g_\sigma(\Delta y_1^{(i+1)})g_\sigma(\Delta y_2^{(i+1)})\dots g_\sigma(\Delta y_K^{(i+1)}),
\end{aligned} \tag{2.24}
$$

where $\Delta y_j^{(i)} = y_j^{(i)} - y_j^{(i-1)}$. Because of symmetry reasons, $p(Y^{(2)}/Y^{(1)}) = p(Y^{(3)}/Y^{(2)}) = \dots = p(Y^{(d)}/Y^{(d-1)})$. The entropy of these d vectors $Y^{(1)}, Y^{(2)}, \dots, Y^{(d)}$ is given by

$$
\begin{aligned}
H(Y^{(1)}, Y^{(2)}, \dots, Y^{(d)}) &= H(Y^{(1)}) + H(Y^{(2)}) + \dots + H(Y^{(d)}) - I(Y^{(1)}; Y^{(2)}) \\
&\qquad - I(Y^{(2)}; Y^{(3)}) - \dots - I(Y^{(d-1)}; Y^{(d)}) \\
&= dH(Y) - (d-1)I(Y^{(i)}; Y^{(i+1)}).
\end{aligned} \tag{2.25}
$$

For the non-discrete vector variables $Y^{(i)}$ and $Y^{(i+1)}$, the mutual information, $I(Y^{(i)}; Y^{(i+1)})$, is given by [162] as

$$
I(Y^{(i)}; Y^{(i+1)}) = h(Y^{(i+1)}) - h(Y^{(i+1)}/Y^{(i)}) \tag{2.26a}
$$

with the assumption that the following integrals,

$$h(Y^{(i+1)}) = h(Y) = \int p(Y) \log \frac{1}{p(Y)} dY \tag{2.26b}$$

and

$$h(Y^{(i+1)}/Y^{(i)}) = \int p(Y^{(i+1)}, Y^{(i)}) \log \frac{1}{p(Y^{(i+1)}/Y^{(i)})} dY^{(i+1)} dY^{(i)} \tag{2.26c}$$

both exist.

Consider $h(Y)$ first. From (2.23),

$$\begin{aligned} h(Y) = \int \cdots \int q(y_1) g_\xi(\Delta y_2) \ldots g_\xi(\Delta y_K) \Bigg[-\log q(y_1) \\ - \sum_{i=2}^{K} \log g_\xi(\Delta y_i) \Bigg] dy_1 d\Delta y_2 \ldots d\Delta y_K. \end{aligned} \tag{2.27a}$$

But $-\log g_\xi(y) = \frac{1}{2}\log(2\pi\xi^2) + \frac{y^2}{2\xi^2}$, and $\int_{-\infty}^{\infty} \frac{y^2}{2\xi^2} g_\xi(y) dy = \frac{1}{2}$. Therefore,

$$\begin{aligned} h(Y) &= \int q(y_1) \log \frac{1}{q(y_1)} dy_1 + \frac{K-1}{2} \log(2\pi e \xi^2) \\ &= h(y_1) + \frac{K-1}{2} \log(2\pi e d \sigma^2). \end{aligned} \tag{2.27b}$$

Here it is assumed that $h(y_1)$ exists. Now, consider the second integral. From (2.24),

$$\begin{aligned} h(Y^{(i+1)}/Y^{(i)}) &= \int p(Y^{(i)}) p(Y^{(i+1)}/Y^{(i)}) \log \frac{1}{p(Y^{(i+1)}/Y^{(i)})} dY^{(i+1)} dY^{(i)} \\ &= \frac{K}{2} \log(2\pi e \sigma^2) \end{aligned} \tag{2.28}$$

following a similar way. Hence, (2.26) becomes

$$I(Y^{(i)}; Y^{(i+1)}) = h(y_1) + \frac{1}{2} \log \left(\frac{d^{K-1}}{2\pi e \sigma^2} \right). \tag{2.29}$$

In case of the VQ scheme, the vectors X are assumed independent. Hence, the absolute entropy of each vector would be $H(X)$. In the DVQ case, the average entropy of one vector is $H(Y) - \frac{d-1}{d} I(Y^{(i)}; Y^{(i+1)})$. If we assume $H(X) \approx H(Y)$, then the DVQ coder has a bit rate saving of $\frac{d-1}{Kd} I(Y^{(i)}; Y^{(i+1)})$ bits per sample over the VQ coder. The above assumption may seem loose, because one might expect $H(Y)$ to be larger than $H(X)$. However, both absolute entropies are infinite here, and the mutual information between consecutive vectors is an appropriate measure of the maximum bit rate saving. Note that we do not compress the labels further by entropy coding in either scheme, therefore the quantity $H(Y) - H(X)$ is not important in this discussion. Thus, bit rate advantage per sample of the DVQ scheme over the VQ scheme is given by

$$\frac{d-1}{Kd} \left[h(y_1) + \frac{1}{2} \log \left(\frac{d^{K-1}}{2\pi e \sigma^2} \right) \right]. \tag{2.30}$$

Consider a VQ coder with $\mathcal{N}$ codewords, where $\mathcal{N}$ is less than N. The asymptotic average distortion of this coder, from (2.12), will be

$$\left(\frac{N}{\mathcal{N}} \right)^{\frac{2}{K}} D_{VQ}. \tag{2.31}$$

Equating this distortion to D_{DVQ} in (2.21), we get the following relation:

$$\frac{N}{\mathcal{N}} = d^{\frac{K-1}{2}}. \tag{2.32}$$

Assuming N and $\mathcal{N}$ to be integer powers of 2, the bit rate for the smaller VQ coder is $\frac{1}{K} \log_2 \frac{N}{\mathcal{N}}$ bit per sample less than for the original VQ coder. Comparing this bit rate with the bit rate in (2.30), one finds that the DVQ has a lower rate than the VQ for equal average distortion so long as the following condition is satisfied,

$$K < 1 + \frac{(d-1)\log 2 \left[2h(y_1) - 1 - \log(2\pi\sigma^2) \right]}{\log d \left[d(1 - \log 2) + \log 2 \right]} \tag{2.33}$$

where log refers to the natural logarithm.

Substituting $d = 2, K = 64$ in (2.21), we find the SNR resulting from the DVQ coder to be almost 3 dB below that of a VQ coder with same parameters. If

$q(x)$ is taken to be normal density with a variance of $\hat{\sigma}$, then $h(y_1) - \frac{1}{2}\log 2\pi e\sigma^2$ in (2.30) becomes $\log(\hat{\sigma}/\sigma)$. For gray-level image the value of $(\hat{\sigma}/\sigma)$ in one dimension ranges from 10 to 50. Using this value in (2.30), the DVQ coder will have a bit rate advantage of nearly 0.18 bit per sample over the VQ coder. These figures, however, are only true for asymptotically large N. For $N = 1024$ with same values of d and K, our simulation results show 1.3 dB reduction in SNR with 0.03 bit per pixel reduction in bit rate for the two-dimensional DVQ coder [Figure 2.4(a)].

2.3 Experimental Results

The DVQ coder has been applied to monochromatic images and compared with results obtained from a VQ coder. Extension of the distributed-block scheme into two dimensions is automatic. Figure 2.2 shows the formation of distributed blocks from a two-dimensional pixel matrix for $d = 2$. The coder will have d^2 channels and the interdependence between the channels will also be two-dimensional. A square block of size 8×8 is used, and the number of codewords, N, is taken to be 1024 or less. For reason of comparison, the training sequence and other parameters are kept exactly the same for both the coders.

The similarity function, $f(C)$, used in the DVQ coder is the average intensity function, as described by (2.2). We have not tried to optimize this function. The codebook has been sorted using this similarity function. The coder has four channels: (1,1), (1,2), (2,1), and (2,2). Let us rank these as channels 1, 2, 3, and 4, and call the corresponding inputs $X^{(1)}$, $X^{(2)}$, $X^{(3)}$, and $X^{(4)}$. In the first encoder, $b_1 = 1$ and $e_1 = N$, or full search is performed for $X^{(1)}$. Because $X^{(4)}$ is least

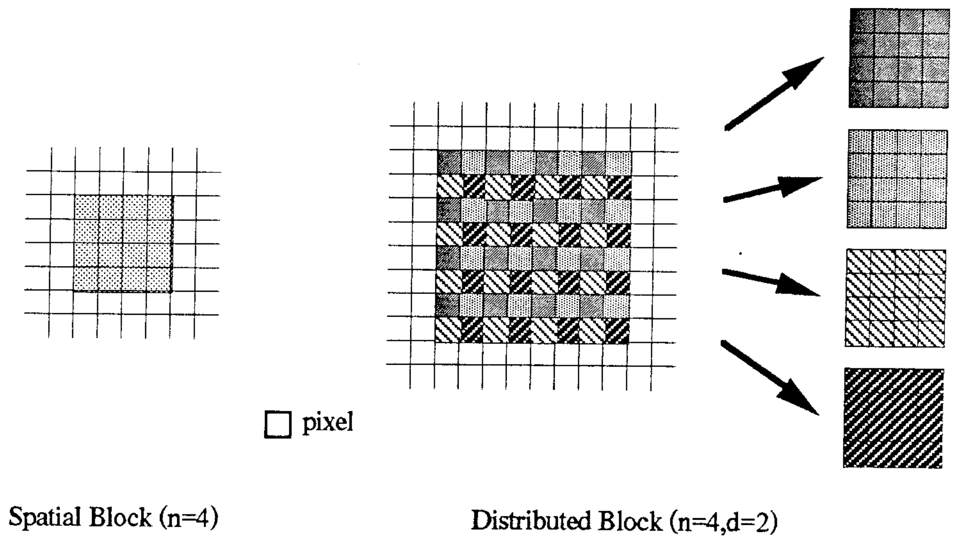


Figure 2.2: Extraction of the Distributed Blocks for Image

correlated to $X^{(1)}$, it is encoded next using

$$b_4(m^{(1)}) = \max\{1, m^{(1)} - \frac{N}{2^{r_4+1}}\}, \qquad e_4 = b_4 - 1 + \frac{N}{2^{r_4}}. \tag{2.34}$$

Here the partial search range is $R_4 = (N/2^{r_4})$, which saves r_4 bits to encode $X^{(4)}$. The strategy of encoding the two least correlated vectors first works best here, because it determines the possible range for the remaining labels. $X^{(2)}$ and $X^{(3)}$ are encoded next using

$$b_2(m^{(1)}, m^{(4)}) = \max\{1, \frac{m^{(1)} + m^{(4)}}{2} - \frac{N}{2^{r_2+1}}\}, \qquad e_2 = b_2 - 1 + \frac{N}{2^{r_2}}, \tag{2.35}$$

and similar functions for b_3 and e_3. The average saving in bits per pixel for this encoder is then $(r_2 + r_3 + r_4)/K$. Figure 2.3 shows the percentage of times the decision made by the encoder is sub-optimal for different values of r_2 or r_3, r_4 and N. In general a search region twice as long is required for $X^{(4)}$ to have a similar percentage as for $X^{(2)}$ or $X^{(3)}$. One should also remember that, even if the best match belongs outside the search range, the sub-optimal match may generate only marginally more distortion. The advantage of this scheme over entropy-coded partial label transmission is that the later scheme has a variable rate and does not save computation.

Figure 2.4(a) shows the resulting signal-to-noise ratio (SNR) versus bit rate for the DVQ and VQ coders. For our simulation, the well-known 256-level picture of Leena was taken. To evaluate the coder performance, peak SNR is used as given below:

$$\text{SNR} = 10 \log_{10} \frac{255^2}{E\left\{[x_{decoded}(i,j) - x_{original}(i,j)]^2\right\}}. \tag{2.36}$$

Here $x(i,j)$ is the intensity value of the (i,j)-th pixel. Our DVQ scheme had $N = 1024$ codewords. The results show the DVQ to be superior in quality to the VQ for bit rates lower than about 0.13 bits/pixel. As has been pointed out before,

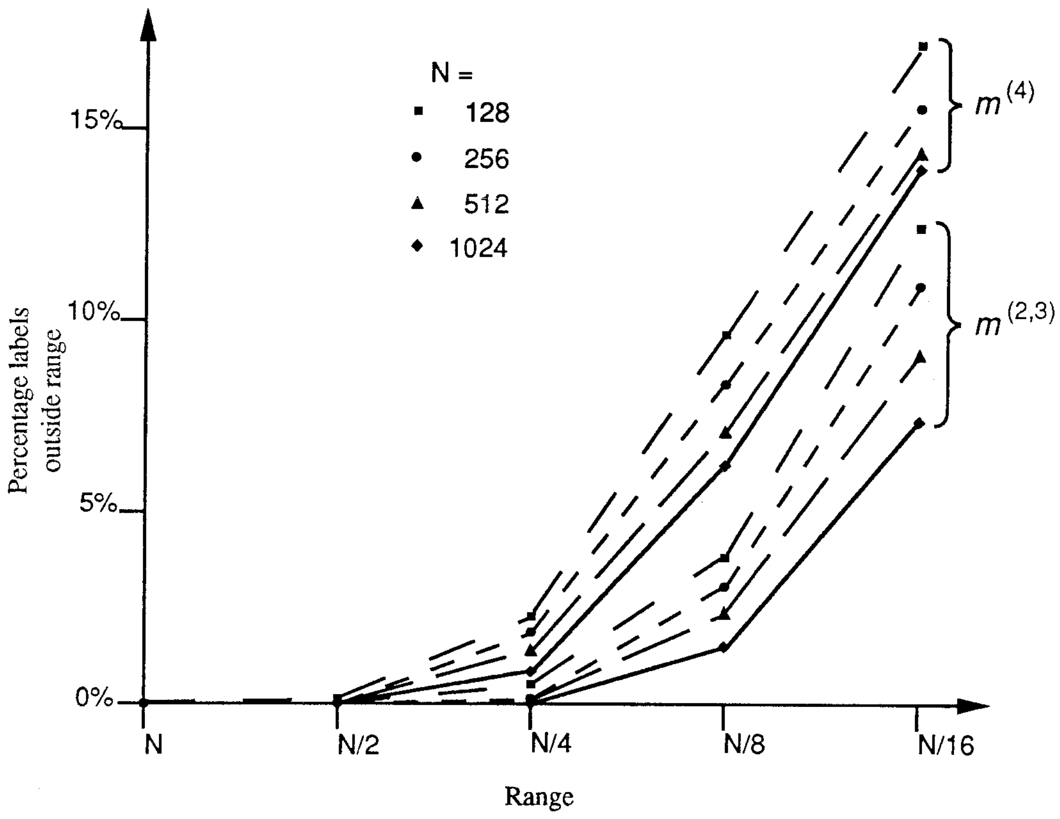


Figure 2.3: Partial Range Coverage for the DVQ Image Coder

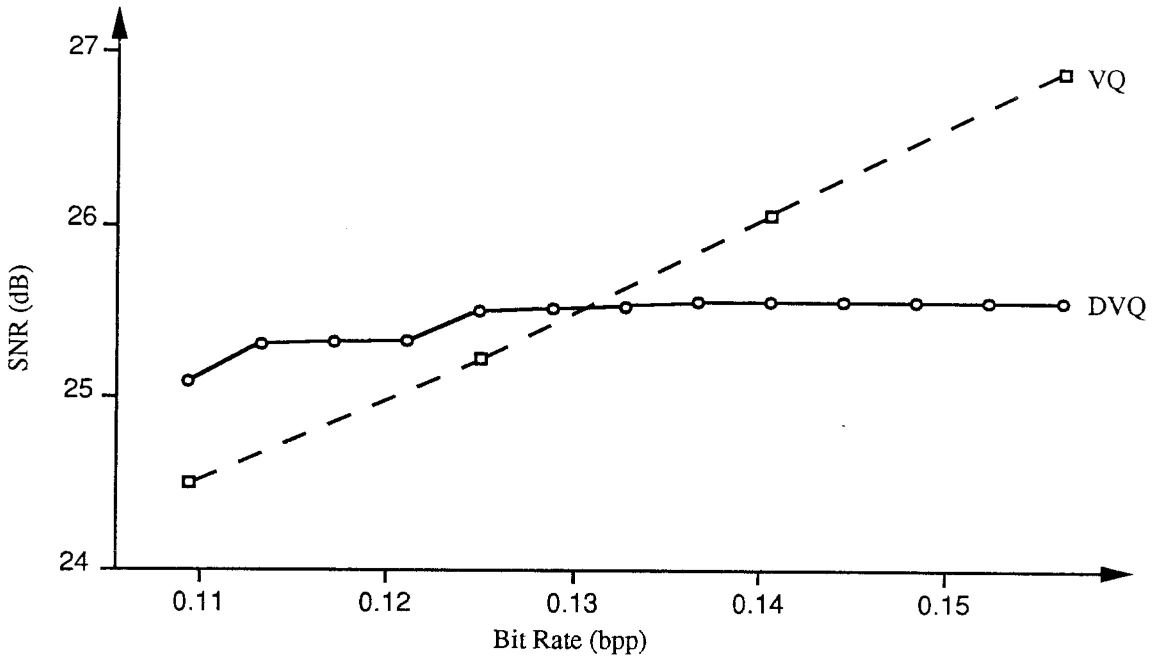


Figure 2.4(a): SNR Comparison of the DVQ and the VQ Coder

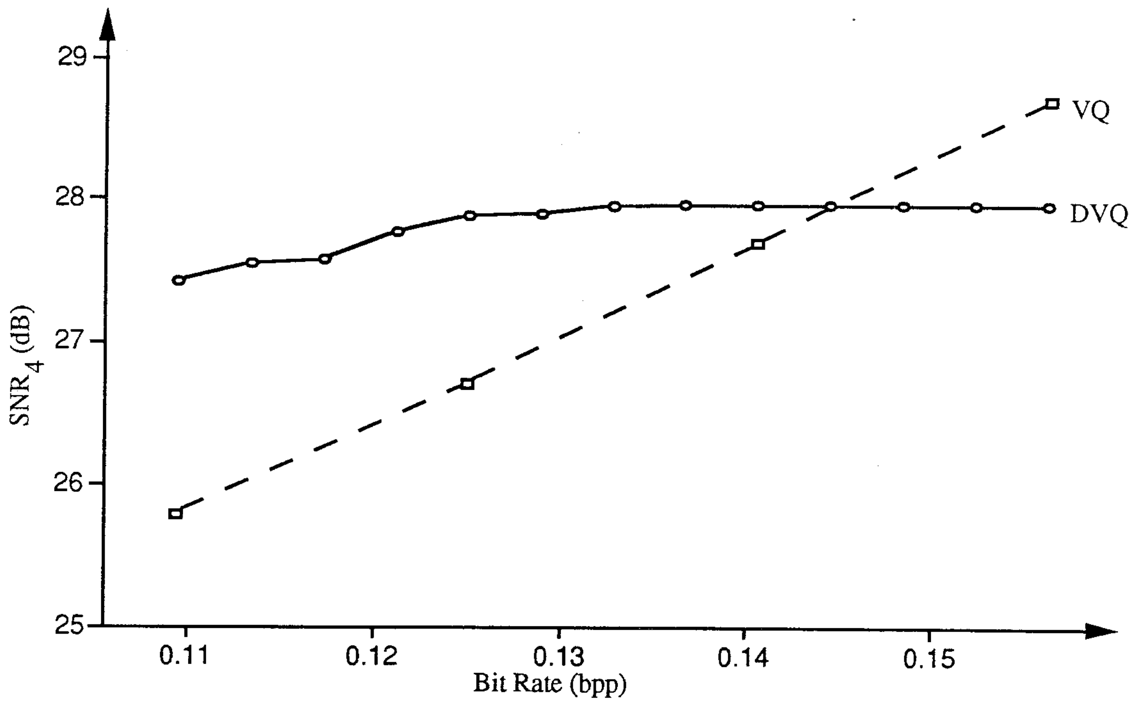


Figure 2.4(b): Modified SNR Comparison of the DVQ and the VQ Coder

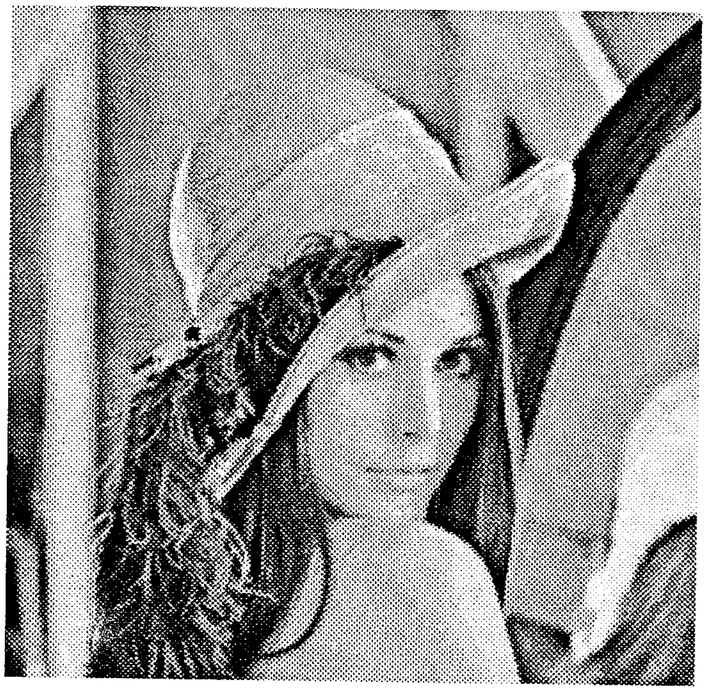
the quantization errors in the DVQ are distributed which is less annoying to eye. The MSE distortion is unable to display this fact.

In an attempt to account for this, the SNR expression in (2.36) could be re-written using a vector-MSE distortion as

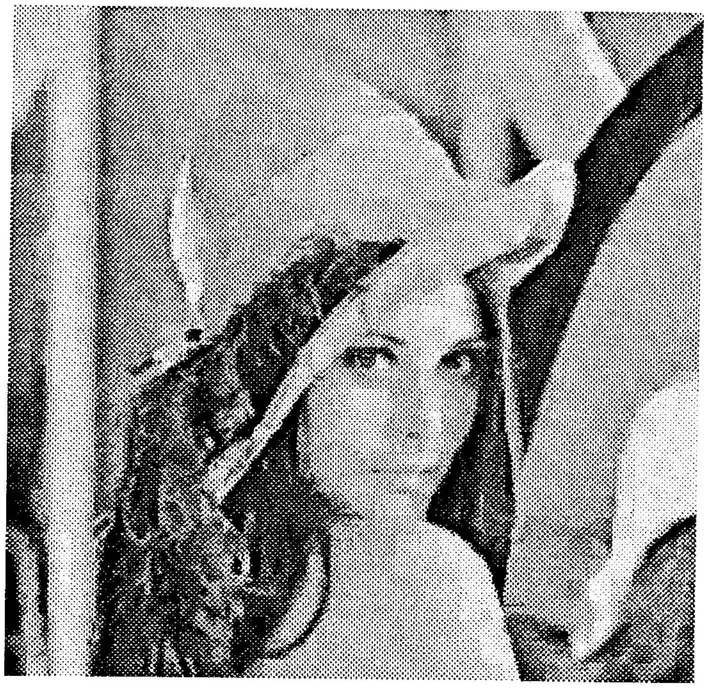
$$\mathrm{SNR}_{d^2} = 10\log_{10}\frac{255^2}{E\left\{\left(\frac{1}{d^2}\sum_{i=1}^{d}\sum_{j=1}^{d}[x_{decoded}(i,j) - x_{original}(i,j)]\right)^2\right\}}. \qquad (2.37)$$

Figure 2.4(b) re-draws the previous results using the modified SNR for $d = 2$. As can be seen, the DVQ coder outperforms the VQ for bit rates lower than about 0.145 bits/pixel now.

Figure 2.5 shows the coding results for the same image. While figure 2.5(a) shows the original version at 8 bits/pixel, 2.5(b) is the decoded image using the DVQ coder at 0.125 bits/pixel. Figures 2.5(c) and 2.5(d) are decoded images using the VQ coder for bit rates of 0.156 and 0.125 bits/pixel respectively. All of the decoded images show degradation in the areas of high details, for example the feathers of the hat. The overall quality of figure 2.5(c) is better than 2.5(b) or 2.5(d) because of its higher bit rate. Comparing for the same rate, figure 2.5(b) represents the high-contrast edges better than 2.5(d), for example the shoulder. The low-contrast edges are not so well represented by the DVQ coder, but being the background, this quality is tolerable.

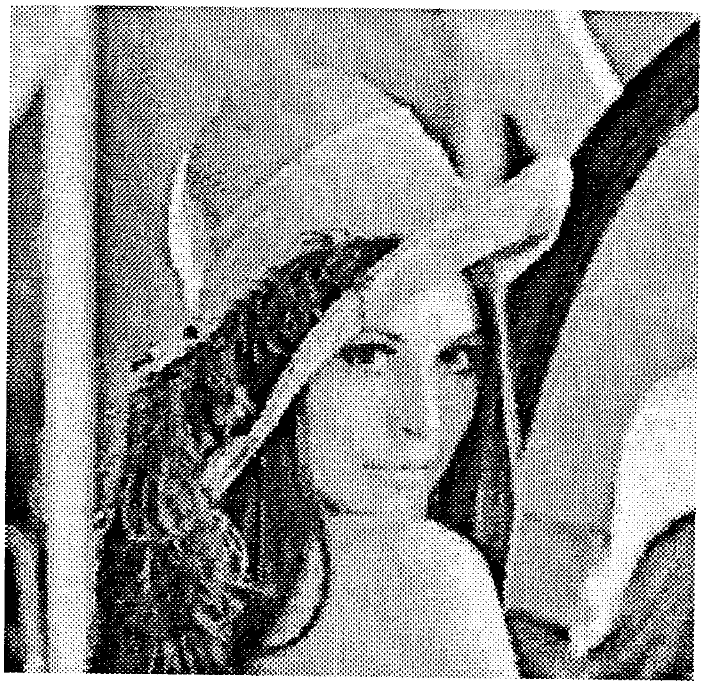


(a)

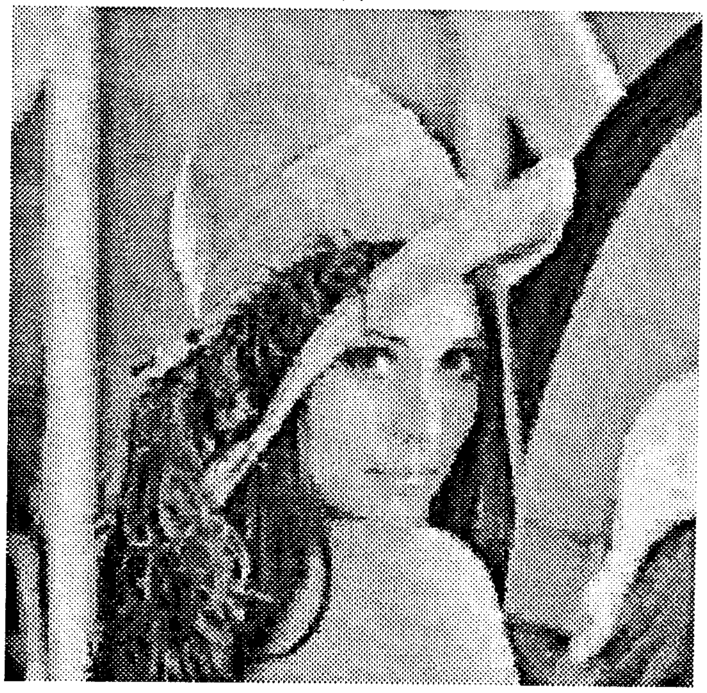


(b)

Figure 2.5: Coding Results with the Distributed-Block VQ: (a) Original Image of Leena at 8 bpp; (b) Decoded Image using the DVQ Coder at 0.125 bpp, SNR 25.5 dB;



(c)



(d)

Figure 2.5: (Continued):
(c) Decoded Image using the VQ Coder at 0.156 bpp, SNR 26.9 dB;
(d) Decoded Image using the VQ Coder at 0.125 bpp, SNR 25.2 dB.

Chapter III
IMPROVED DISTORTION MEASURES

The role played by the distortion measure in an image VQ encoder is very important. The mathematical function used for computing the distortion is nothing but an attempted quantification of human visual discomfort with quantization errors in the decoded picture. Unlike speech coding, where more involved distortion measures such as the Itakura-Saito distortion function have been used alongside the MSE distortion, there have been very few investigations on a more appropriate function than the squared-error distortion in VQ image coding. In the following discussion we look into a simple class of distortion functions, the input-dependent weighted squared-error distortion, which takes into account some psychovisual characteristics. The performance of the codebooks generated using these distortion functions are compared to that of the conventional MSE codebook. This simple yet efficient distortion measure suggested by us, while keeping the coder complexity the same or at worst marginally higher, is found to provide better subjective quality than the MSE distortion.

3.1 Input-Dependent Weighted Squared-Error Distortion Function

For a quantizer to perform well, the quantizer has to represent the source efficiently. Given a specific source and a fixed number of quantization levels, N, an optimum quantizer is defined as the quantizer which can encode the source with minimum average distortion. To define this optimality, a penalty for the distortion or quantization errors has to be assumed. The distortion function of a VQ coder mentioned in Section 1.2 is a computation of this penalty.

For scalars, Lloyd's algorithm can design an optimum quantizer when the source statistics and some distortion measure are specified [163]. Linde, Buzo and Gray extended Lloyd's algorithm to designing vector quantizers in a way similar to the clustering algorithms in pattern recognition [164]. Their algorithm is known as the *LBG algorithm* in VQ coding. Because no appropriately representing source statistics may be available for typical VQ sources such as image and speech, the LBG algorithm designs a vector quantizer optimal for a typical set of input vectors, known as the *training sequence.* The training sequence should be long enough and be chosen in such a way so that it represents the source statistically. The iterative LBG algorithm may, however, converge to a local instead of the global optimum.

The LBG algorithm will not be described here, but we shall mention some of its requirements. When the distortion function is specified, this algorithm is intended to design a codebook with the minimum expected distortion. Naturally, the distortion function $d(X, C)$, as in equation (1.3), has to exist. The algorithm works for any such distortion function as long as the *centroid* of a set of vectors using the distortion function exists, too.

The centroid for a cluster or a set of input vectors is defined to be the vector (not necessarily unique) having a minimum average distortion between it and any other member of the set. Thus, for such a set B, the centroid C is given by

$$C = \begin{matrix} \min \\ Y \end{matrix} \begin{bmatrix} \mathrm{E}\{ & d(X,Y)\} \\ X \in B & \end{bmatrix} . \tag{3.1}$$

In other words, a centroid is an optimal representation of a set of vectors for a specified distortion function. In case of the squared-error and the MSE distortions (they are identical except for a constant scaling factor), the distortion value is proportional to the square of the Euclidean distance between the two vectors, and the centroid happens to be the mean of the set.

In general, it is difficult to incorporate psychovisual effects in a computably simple distortion function. Attempts have been made to get around this problem. In [137] a $\frac{1}{3}$ power-law device followed by a psychovisually modeled filter precedes the VQ coder. Reverse operations are performed at the other end. Thus, while using MSE distortion in the VQ part, the effective distortion measure of the coder is more human-like.

Another class of distortion measures meeting the LBG requirements is the weighted mean square error (WMSE) distortion. If the input vector X and the codeword C are viewed as K-dimensional column vectors, then the WMSE distortion can be expressed as

$$d(X,C) = \frac{1}{K}[X-C]^t \cdot W \cdot [X-C], \tag{3.2}$$

where W is the weight matrix of size $K \times K$. The MSE distortion is a special case when $W = I$, the identity matrix. The WMSE distortion has been used for image VQ coding in a simple way by taking $W_i = w_i I$, where w_i is a scalar and i is the sub-class index [134, 142].

We suggest a general class of distortion measure by extending the WMSE function such that the weight matrix depends on the input vector. Because the function is not normalized, it is named *input-dependent weighted squared-error* (IDWSE) distortion function. Following the former notation, the IDWSE function is given by

$$d(X,C) = [X-C]^t \cdot W_X \cdot [X-C]. \tag{3.3}$$

The above function can easily be computed. In order for the centroid using the IDWSE distortion to exist too, we add a constraint that all non-diagonal elements of the weight matrix have to be zero. For such a diagonal weight matrix W_X, the centroid of a cluster is [Appendix I]:

$$C = \frac{\mathrm{E}\{W_X \cdot X\}}{\mathrm{E}\{W_X \cdot [\mathit{1}]\}}, \tag{3.4}$$

where $[\mathit{1}]$ is the unit column vector. Incorporating the above result in the LBG algorithm, it is then possible to design a VQ codebook optimum in the IDWSE distortion measure.

The following sections illustrate two ways of using the IDWSE distortion function to take into consideration human visual characteristics, and compare the IDWSE codebooks with the conventional MSE codebook for low bit rate image coding.

3.2 Use of the Activity Classes

In this section we propose using the IDWSE distortion function such that the weight matrix of equation (3.3) is of the form

$$W_X = a(X) \cdot I, \tag{3.5}$$

where $a(X)$ is a scalar function of the input vector, X, and is called the *activity index* of the vector. For a spatial block of image, the activity can be defined as the amount of detail, or changes in intensity, present in that block. A possible way to measure this activity is to compute the 'ac' energy of the vector. The local activity in a typical picture changes rather rapidly. The high-contrast edges and the high-detailed areas of an image have much higher activity than the uniform or the slowly changing areas have. In a typical image, the edges and the detailed parts are responsible for the quality of the image, and are usually more noticeable. Thus, a measure of the local activity gives us a chance to segment the image into different classes and to encode each class differently.

The concept of using the image activity to treat different parts of an image separately is not new. In scalar image coding, a visibility function has been used to measure the activity, thereby classifying the pixels and using separate quantizers for each class [32]. For block coding, the 'level of activity,' or the 'ac' energy, of the input blocks has been used to classify the blocks before performing transform coding [66]. The 'activity index' has also been computed in the transform domain by taking the weighted sum of either the absolute or the squared values of the transform coefficients [65].

In VQ image coding, a codebook designed using the widely used squared-error distortion has been found to fail to satisfactorily reproduce the edges of an image. Therefore, to achieve better performance, one has to make sure that enough edge-type codewords are there in the codebook. One suggestion to get this desirable property in a codebook involves the training sequence being segmented into two separate classes, 'edge' and 'shade', using an edge-detector or an edge-classifier [82, 141, 142]. Separate sub-codebooks are designed for each class, and the final codebook is a concatenation of the sub-codebooks. The number of distinct classes may be more than two.

The codebook resulted from the above scheme is no longer optimal in the MSE or any other sense. For this coder, if the encoding search is kept limited to the particular sub-codebook of the class to which the input vector belongs, the equivalent distortion measure will be a special case of the WMSE distortion of equation (3.2):

$$W = wI, \qquad w = \begin{cases} 1, & \text{if } X \text{ and } C \text{ belongs to the same class,} \\ \infty, & \text{if } X \text{ and } C \text{ belongs to a different class.} \end{cases} \tag{3.6}$$

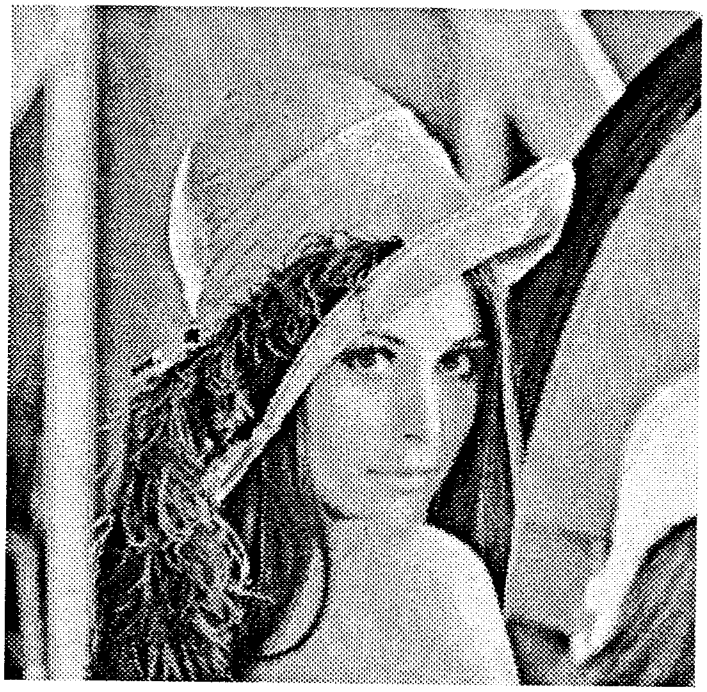
Using this distortion measure, the average distortion for any point becomes infinite, and the desired centroid cannot be determined. Therefore, it is not possible to design the entire codebook as a unit. So, given the size of the final codebook, the cardinality of each sub-codebook has to be determined. It is possible to analytically find an expression for asymptotically optimal allocation of the sub-codebooks, but the probability density of each class has to be known [142]. For a typical training sequence whose probability density function is unknown, and for a finite codebook size, the sub-codebook cardinalities have to be decided arbitrarily.

The IDWSE distortion can be used to achieve a similar effect. An input vector is assigned to some class depending on its activity. The objective of this classification is to separate out the more visible parts of an image from the background having little or no variation. Note that the measure of activity could be scalar or vector. It is therefore possible to use virtually any kind of classifier. An activity index which reflects the relative importance or visibility of the class is assigned to each such activity class. Thus an input block with an edge should have a higher activity index than a low-variation block. Here the activity index is a positive scalar quantity. Negative or zero value is not allowed because then the distortion measure would be inconsistent.

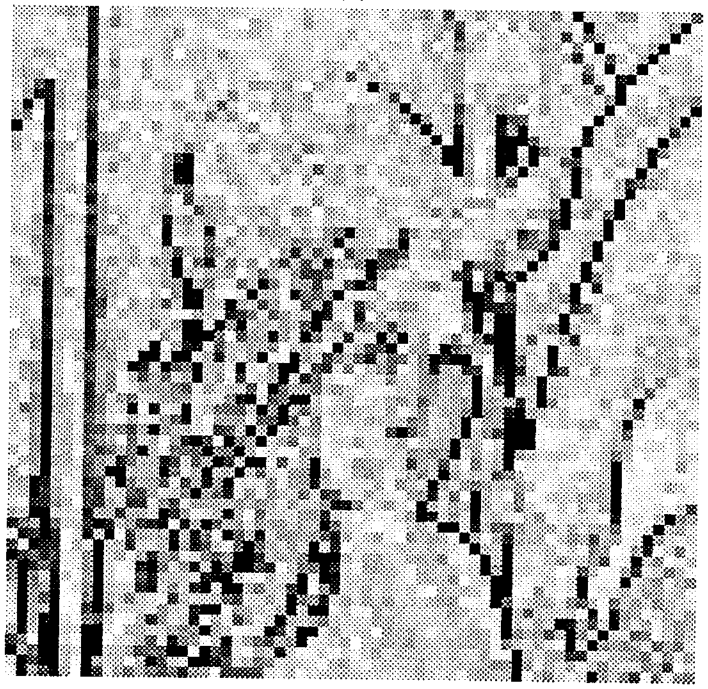
The codebook can be generated on the complete training sequence using this distortion function. The encoder in this case will be a full-search encoder. It is possible to emphasize, or de-emphasize, any particular class in the codebook by simply raising, or lowering, the activity index attached to it. Yet another flexibility in our scheme is that there exists no constraint regarding the number of classes. In fact, there could be infinitely many classes, e.g., a monotone continuous activity index. Our scheme has two distinct advantages over the sub-codebook method. Firstly, the modified distortion measure is used only during the codebook designing. The encoding process is exactly the same as for an MSE-distortion VQ coder. Hence, any conventional VQ coder can be used. Also, the codebook in our case remains well defined because it is optimum for the modified distortion measure.

Now we present our experimental system illustrating the use of activity index for low rate image coding. All of the previous examples of quantifying activity of an image block needed some amount of computation involving multiplication. Instead, we propose a computationally easy way to determine the activity value of a given vector, which does not need any multiplication. Consider a pair of pixels in general position in an image vector. The difference in value of the pixels will be zero or small if there is little or no intensity variation in the vector. On the other hand, if the pixels fall on either side of an edge, or on different intensity segments of a high-detail block, then the difference will be high. To capture edges of various orientations, more than one such pair of pixels is required. The magnitude of the total difference of all such pairs will, therefore, give some measure of the local activity. The more random these pixels are positioned in a vector, the more efficient the scheme will be in determining the activity for a fixed number of such pixel pairs.

We have tried several arrangements and found that pairs of pixels with two-dimensional random distribution perform better than any regular, cyclic or one-dimensional patterns in determining the activity of a square image block with varying edge orientation. In case of 8×8 square blocks, we found that 8 pairs of pixels are necessary to satisfactorily compute the activity. Thus, merely 15 additions are required to measure the local activity of an input block this way. If R denotes a column vector of K elements such that it contains '1' in eight random positions, '-1' in eight other random positions, and '0' elsewhere, then the activity of an input vector X is simply $|R^t \cdot X|$. Figure 3.1 shows the local activity of 8×8 blocks of Leena (a darker block implies higher activity) along with the original version. Note how the edges, the high-detailed parts of her eyes, and the feather of her hat are detected by our simple measure of activity. Observe that the activity in this case, which is a non-negative integer, has a possible range from 0 to 2040.



(a)



(b)

Figure 3.1: Local Activity:
(a) Original Image of Leena;
(b) Local Activity of Leena.

Figure 3.2 shows the distribution function of activity for the entire training sequence. We have defined four activity classes by dividing the activity range into four equiprobable regions, shown in the same diagram. In general, m such classes can be defined by using some increasing threshold values:

$$\begin{aligned} a(X) &= a_1, \text{ if } |R^t \cdot X| < t_1 \\ &= a_2, \text{ if } t_1 \leq |R^t \cdot X| < t_2 \\ &\vdots \\ &= a_m, \text{ if } t_{m-1} \leq |R^t \cdot X|. \end{aligned} \tag{3.7}$$

Here $t_1, t_2, \cdots, t_{m-1}$ are the thresholds or class boundaries, and $a_1, a_2, \cdots, a_m$ are the activity indices, typically monotonically increasing, of the m classes. The four activity classes in our system could be subjectively described as the non-varying background, the low-detailed area, the higher details and low-contrast edges, and the high-contrast edges.

We have tried several sets of values as the activity indices for the four classes. Some codebooks have been designed using $a_i = 1 + (i-1)\delta$ (linear increment) for different positive values of δ. We have also tried $a_i = (1+\delta)^{i-1}$ (exponential increment) for different positive values of δ. Note that $\delta = 0$ means the basic squared-error distortion, which we take to be the standard case for comparison. The training sequence and other parameters have been kept identical so that the results can be compared. We found the performance of the activity indexed codebooks to be as expected. With increasing value of δ, the edges and the contrasting blocks are encoded better.

For comparison, the image of Leena is again selected. Figure 3.3 shows the MSE distortion of the decoded image of Leena for some of the values of δ we have attempted. The bit rate for each case is 0.156 bit/pixel, with $N = 1024$ and $K = 64$. Observe that better signal-to-noise ratio has indeed been achieved for

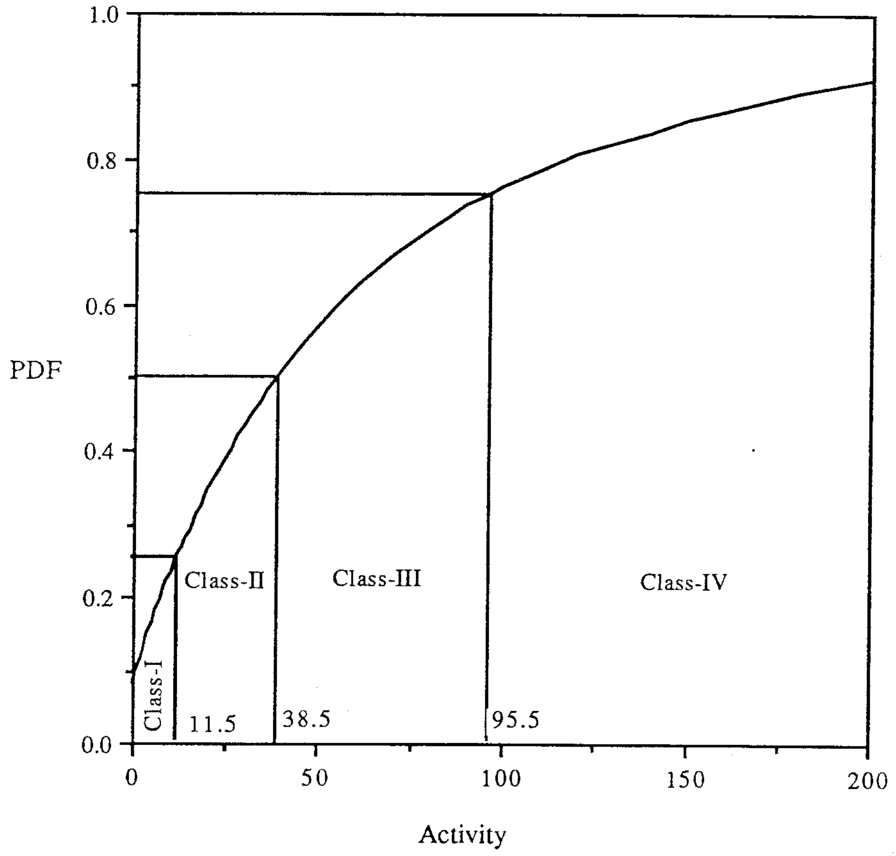


Figure 3.2: Distribution Function of Activity

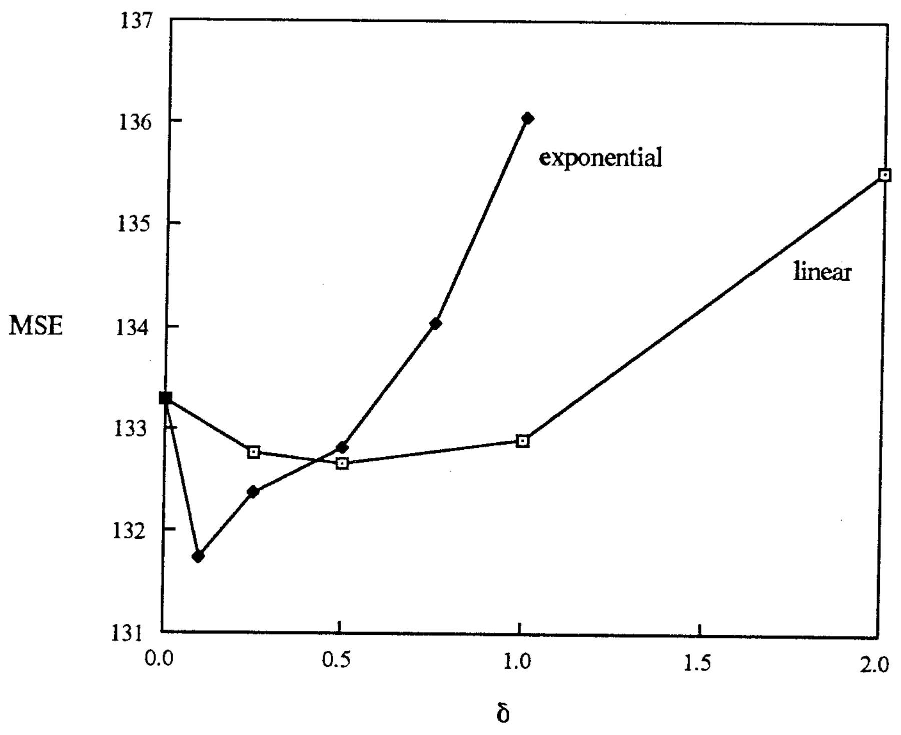


Figure 3.3: MSE Distortion versus Activity Index

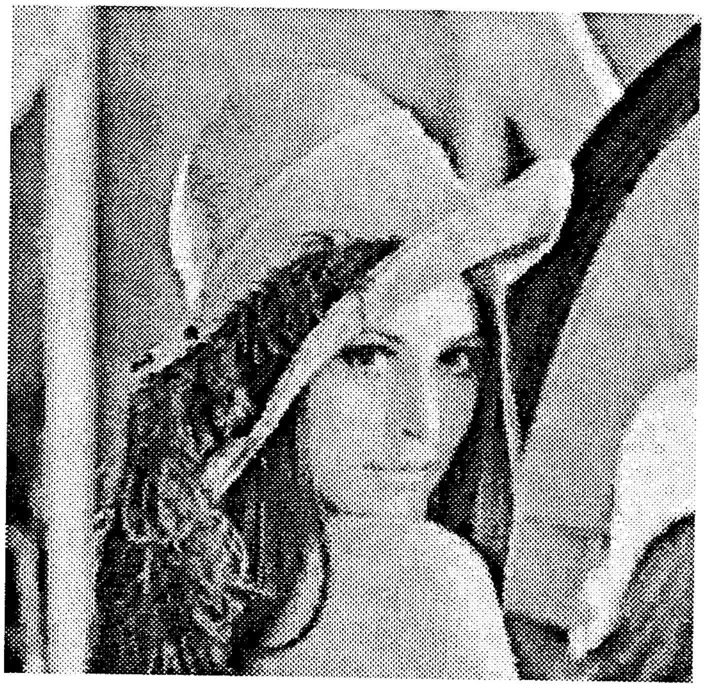
small δ. However, this curve does not compare the subjective quality of the images. Improved subjective quality can be observed for higher values of δ. Figure 3.4 compares the decoded images for δ values of 0 and 2 (linear increment). Figures 3.4(a) and 3.4(b) show the complete image, while figures 3.4(c) and 3.4(d) are the enlarged version of the back of Leena's hat. Note the better representation of the edges in figure 3.4(d), which shows the effect of activity classes on image VQ coders.

Therefore, incorporating the activity classes in the distortion measure improves the representation of edges in an image without changing the coder complexity at all. Using the proposed algorithm it is possible, by proper selection of indices, to design a codebook with just enough edges representing a set of pictures efficiently. Any other modified VQ scheme, such as the distributed-block VQ, can be used with the activity classes to retain advantages of both schemes.

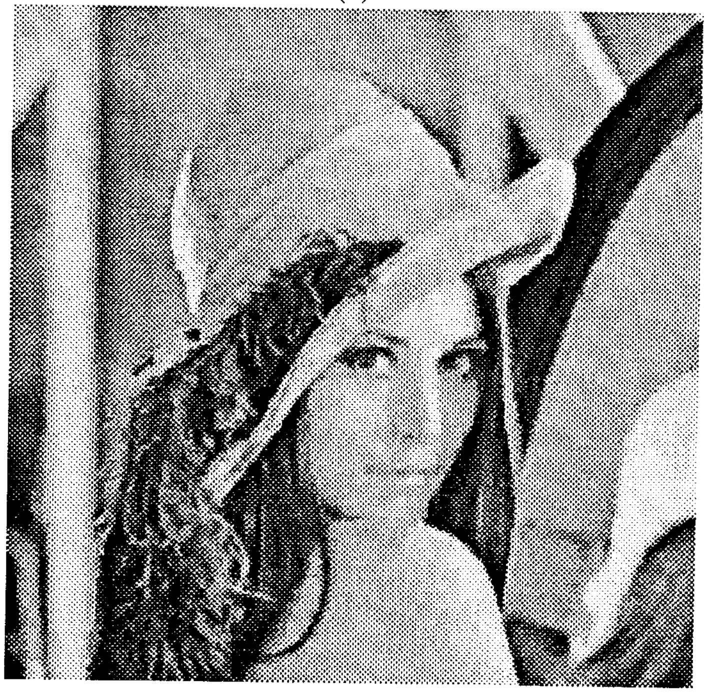
3.3 Emphasis on the Block Boundaries

A common problem in the low bit rate block image coding is the blockiness, described in Section 2.1. This degradation, occurring in the geometrical boundaries of the blocks of a decoded image, is usually the most noticeable and most annoying quantization error. This problem is common to both transform coding and vector quantization. There is thus great motivation to find some means of subduing, or at least reducing, the blockiness in low bit rate block coding.

Some modifications to the block coder has been suggested to get rid of the blockiness. In [69] a pinned orthogonal transform coder transmits some information regarding the block boundaries along with the transform coefficients in order to encode the boundaries better. As a result, this scheme requires some extra bits to

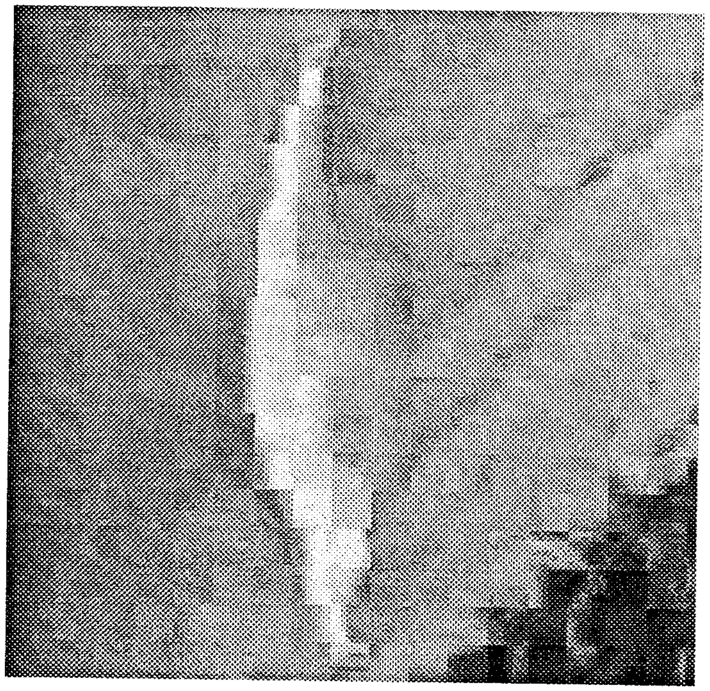


(a)

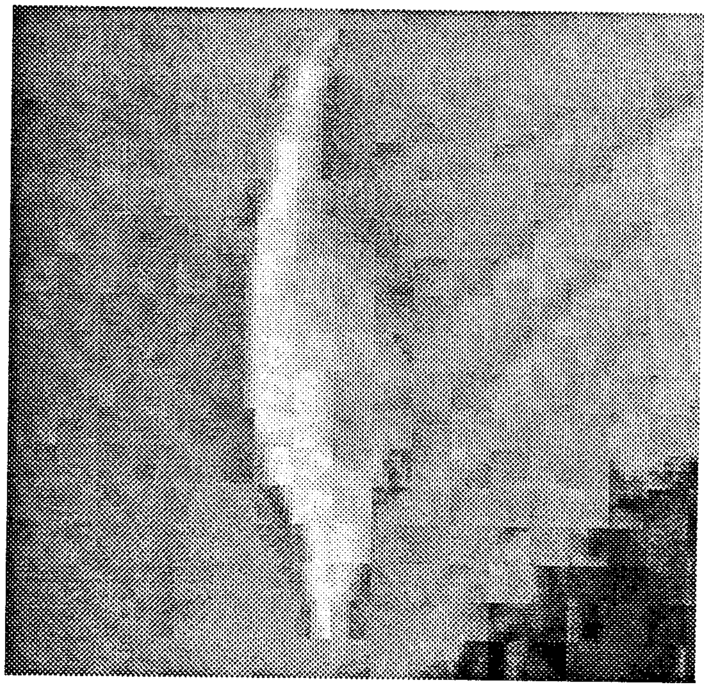


(b)

Figure 3.4: Coding Results with Activity Index:
(a) Decoded Image of Leena for $\delta=0$ at 0.156 bpp, SNR 26.88 dB;
(b) Decoded Image of Leena for $\delta=2$ at 0.156 bpp, SNR 26.81 dB;



(c)



(d)

Figure 3.4: (Continued):
(c) Enlarged Version of a Part of (a);
(d) Enlarged Version of a Part of (b).

achieve better quality.

In the MSE VQ coding, each pixel of the input block, or each dimension of the input vector, is encoded with equal effort. However, it is intuitively clear that, in order to preserve the continuity of an image across the blocks, the peripheral pixels are needed to be encoded more carefully. A quantization error in the interior pixels will do less harm. Therefore, one possible approach to reduce blockiness in a VQ coder is to assign a higher penalty in encoding the pixels at the geometrical boundary of the block than in encoding the interior pixels.

The IDWSE distortion measure can be used to achieve this effect. In this case the weight matrix, W_X in equation (3.3), becomes

$$W_X = \mathit{\Delta} = \{d_{ii}\}, \tag{3.8}$$

where $\mathit{\Delta}$ is a diagonal matrix independent of the input vector, X. Here the diagonal elements of $\mathit{\Delta}$, d_{ii}, are what we call the *emphasis factors* for each dimension, or the *dimensional weights.* For the dimensions i corresponding to the interior pixels, $d_{ii} = 1$. The remaining diagonal elements corresponding to the pixels at the block boundaries are set equal to p here, except for the four vertices of the block. For the vertex pixels, $d_{ii} = q$. Because a vertex pixel has neighboring pixels from three other blocks, while an edge pixel has only one neighboring block, a different (typically higher) emphasis factor is assigned to the vertices. In order to represent the boundary pixels better, the values of p and q should be greater than 1. The dimensional weights we see represent the relative importance of a dimension with respect to other dimensions.

Because the weight matrix $\mathit{\Delta}$ is independent of X, the centroid in equation (3.4) becomes the arithmetic mean of the cluster, which is very easy to compute. However, the modified distortion measure in this block boundary emphasis scheme increases both the codebook designing and the encoding computation. The number of elements not equal to 0 or 1 in Δ is $4\sqrt{K}-4$. Therefore, an additional $4N(\sqrt{K}-1)$ multiplications are required in encoding one input vector in our scheme. Taking p and q to be integer powers of 2, the additional multiplications needed could be replaced by bit-shifting operations in hardware implementations. However, for the neural implementations of the VQ encoder proposed in Chapter V, neither the encoding time nor the complexity is altered by this scheme.

Simulation results of emphasizing the block boundaries for low rate image coding will now be given, comparing the performance with that of the MSE VQ. Keeping all other parameters the same, we have designed several codebooks for different values of $p = q = \delta$. The basic MSE codebook is a special case when $\delta = 1$. Figure 3.5 shows the MSE distortion of the decoded image of Leena for some of these codebooks with increasing values of the emphasis factor, δ. Though for smaller emphases our scheme has a lower conventional distortion than the MSE VQ, the distortion is higher for larger values of δ. The desired subjective improvement is, however, noticeable only for the higher values of δ.

The codebook generated by our scheme has another advantage over the MSE codebook. Ideally, one would like all codewords of a VQ codebook to be equiprobable so that maximum compression can be achieved from the entropy point of view. Not only does the LBG algorithm not guarantee that the codewords are utilized equally often, another problem associated with that algorithm is the empty clusters or the unutilized codewords. The presence of unutilized codewords decreases the efficiency of the coder. In order to measure the uniformity of the codebook, we computed the variance (σ^2) of the codeword utilization (number of times a codeword is used for the training sequence). Figure 3.6 plots the variance versus the emphasis factor, δ, for some of these codebooks. Observe that the variance of a VQ codebook reduces when the emphasis factor goes up. Therefore, a relatively well utilized codebook resulted from our scheme.

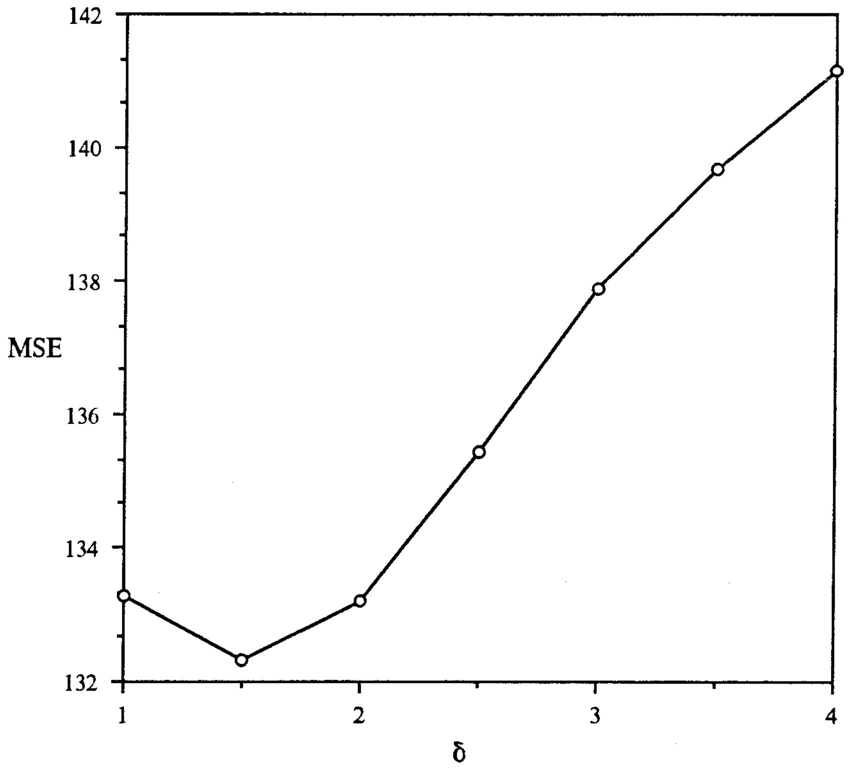


Figure 3.5: MSE Distortion versus Emphasis Factor

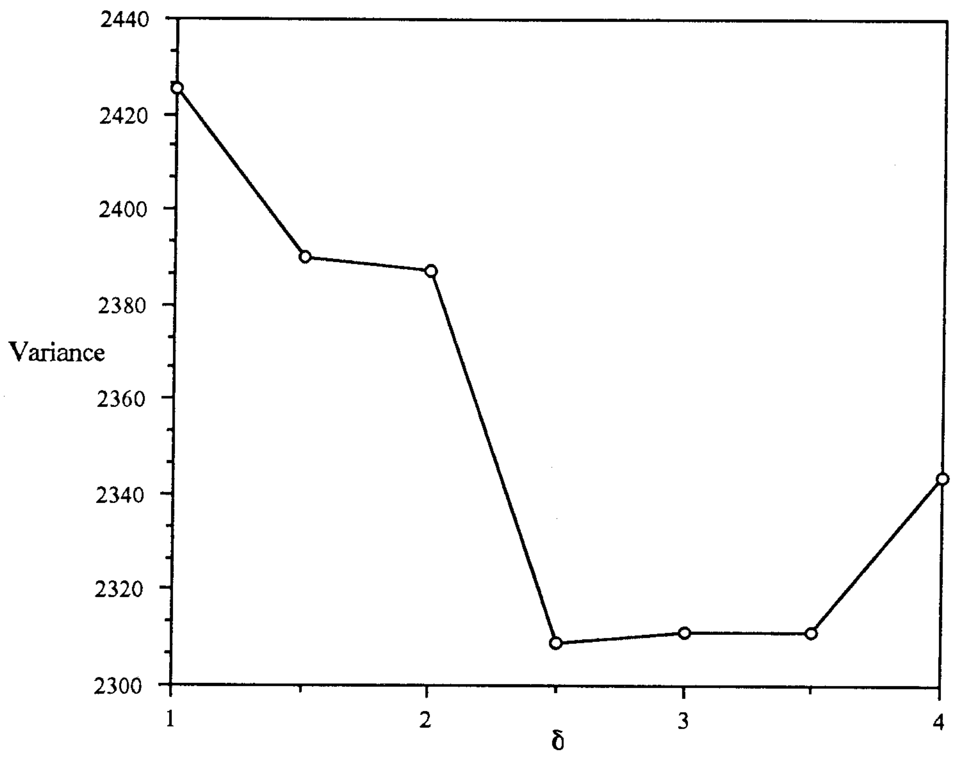
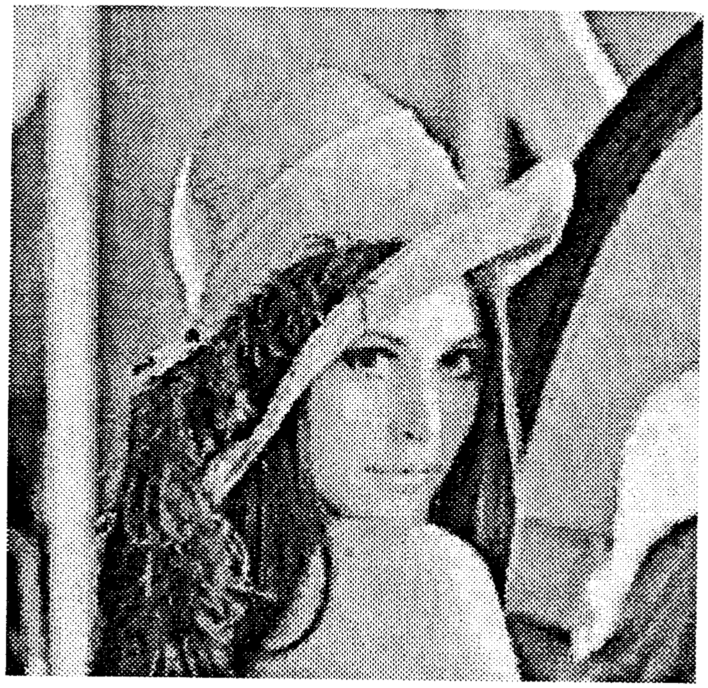


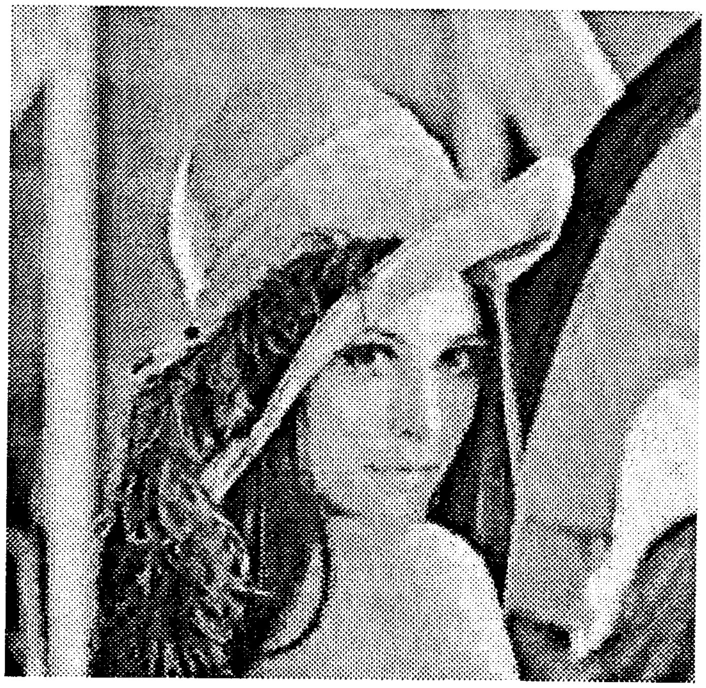
Figure 3.6: Variance of Codeword Utilization versus Emphasis Factor

Figure 3.7 shows some simulation results. Figures 3.7(a) and 3.7(b) are the complete decoded images of Leena using codebooks with δ values of 1 and 2.5 respectively. Figures 3.7(c) and 3.7(d) are enlarged versions of a part of these images. The blockiness of the bottom image has been considerably reduced. Moreover, the edges of the decoded image using the new codebook are better represented.

Thus, introducing some positive emphasis at the block boundaries produces a decoded image with less blockiness, and also improves the edge representation, both of which are desirable in image VQ coding. When using the block boundary emphasis distortion function, the encoder computation is marginally raised, though the neural implementation is not affected. This blockiness-reducing scheme can also be used along with other modifications to the basic VQ coder.

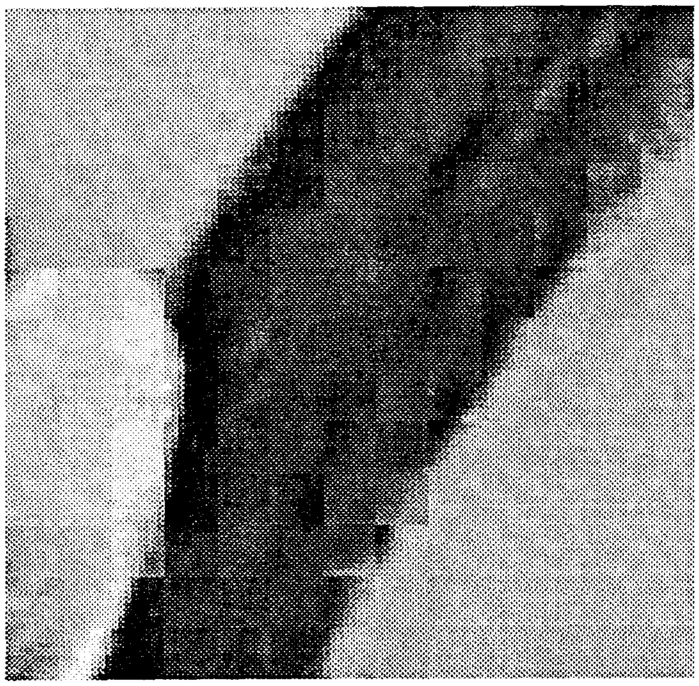


(a)

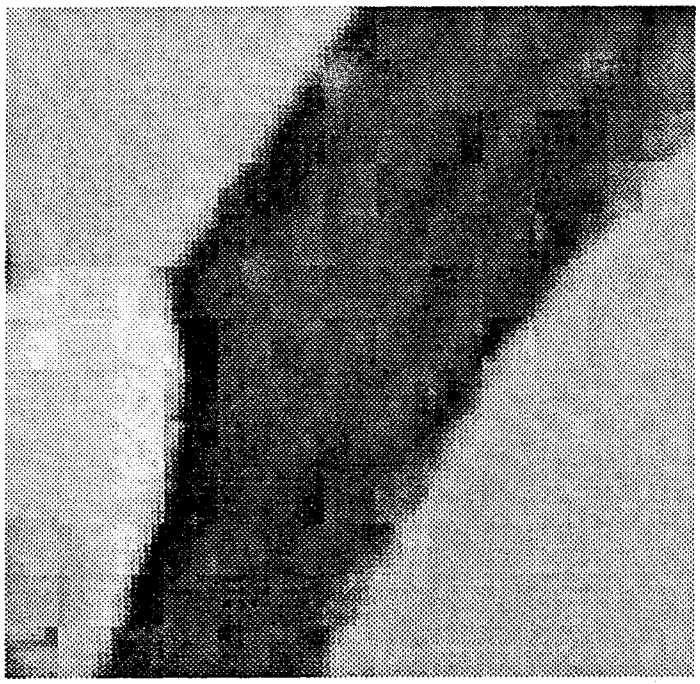


(b)

Figure 3.7: Coding Results with Block Boundary Emphasis:
(a) Decoded Image of Leena for δ=1 at 0.156 bpp, SNR 26.88 dB;
(b) Decoded Image of Leena for δ=2.5 at 0.156 bpp, SNR 26.81 dB;



(c)



(d)

Figure 3.7: (Continued):
(c) Enlarged Version of a Part of (a);
(d) Enlarged Version of a Part of (b).

Chapter IV
FILTERING TECHNIQUES

Lossy image coding methods are known to produce annoying coding errors or artifacts in the decoded image. Apart from trying to diminish these degradations in the coding algorithm itself, filtering techniques are also used to improve the subjective quality of the decoded image. Naturally, the type of this filter depends entirely upon the type of coding scheme used. In case of low rate block encodings such as VQ, transform coding, block truncation coding, etc., blockiness is one of the major problems. In this chapter we attempt to design some simple filters in order to reduce blockiness and other coding errors, with special consideration to the VQ image coder. We suggest a simple two-dimensional filter which is shown to produce subjectively better images. We also suggest the use of a prefiltered codebook before filtering to further improve the image quality.

4.1 Introduction

An inherent disadvantage of very low bit rate VQ image coding is the annoying presence of quantization noise in the decoded picture. Due to the fact that often the coding noise is statistically quite different from the input image, simple image enhancement techniques such as filtering can be applied to the decoded image to get rid of the obvious errors in the decoded version of the picture. The quality improvement of the enhancement technique might however be limited by the available processing time. The process of enhancing the image quality is referred to as *postprocessing* or *postfiltering* when it is performed at the decoder, usually after the decoding is complete. If the source image is filtered before encoding, the process is named *preprocessing* or *prefiltering.*

Blockiness is the most prominent coding error in case of the VQ image coding. Because of the mismatch between the luminance values of two spatially adjacent codewords, discontinuity is developed at the block boundaries and a staircase effect shows up at the edges. These errors contribute to the higher side of the frequency spectrum. Because most of the spectral energy of a typical image lie in the low-frequency region, a low-pass postprocessing filter may be the first linear candidate in eliminating the block discontinuities.

For a typical monochrome image, we have found that a 6 dB attenuation from the passband to the stopband is adequate. Because the signal and noise are both present in between, a smooth transition would do equally well. This kind of filter can be designed with very small length. However, the processed image quality using such a low-pass filter is found to be unsatisfactory. The attenuation cannot be increased much due to the presence of original image information in the high-frequency part of the spectrum. In fact, the relatively low high-frequency spectral components are responsible for the subjective quality of an image. Perhaps this is the reason why a simple low-pass filtering is not enough to reduce blockiness. In the next section we take a more involved approach for the same purpose.

4.2 Selective Spectral Attenuation Filtering

Because we are mainly concerned with the noise at the block boundaries, we may take advantage of the known positions of the noise. For a given spatial block size of $m \times m$, the errors are localized at a regular interval of m pixels in both horizontal and vertical directions. Due to this periodic nature, the error spectrum is expected to have amplitude peaks at horizontal and vertical frequencies of π/m, $3\pi/m$, $5\pi/m$, and so on. A *selective spectral attenuation* filter will selectively attenuate these frequency components. The amount of attenuation and the width of the stopbands can be specified.

For 8×8 block size, we need to attenuate four odd multiples of $\pi/8$. Even for a small attenuation of 3 to 5 dB, the one-dimensional filter length will be large if we specify narrow stopbands and transition bands. Instead, we decided to use very smooth transition between the 'hills' and the 'valleys.' In that case, a one-dimensional FIR filter of length 9 with real coefficients can be designed with the desirable property. Figure 4.1 shows the magnitude response of such a filter, whose transfer function is of the form

$$\boldsymbol{H}(z) = \prod_i \left[1 - 2r_i \cos\theta_i z^{-1} + r_i^2 z^{-2}\right]. \tag{4.1}$$

For each frequency θ to be attenuated, a pair of complex conjugate zeros of the form $re^{j\theta}$, $re^{-j\theta}$ have been inserted into the transfer function, where the value of r ($\neq 1$) depends on the attenuation required.

Our image signal is inherently two-dimensional. Therefore, consider a two-dimensional frequency plane, (ω_h, ω_v), where the subscripts to the frequency variables denote the horizontal and vertical directions. What is actually desirable is local attenuation at some points, such as $(\pi/8, 0)$, $(3\pi/8, 0)$, $(0, \pi/8)$, etc.

However, it turns out that designing a two-dimensional transfer function from this type of specified zeros is not a straightforward task, unlike the one-dimensional case. A one-dimensional transfer function can always be expressed as a product of first-order factors. Therefore, such a transfer function can be specified by the

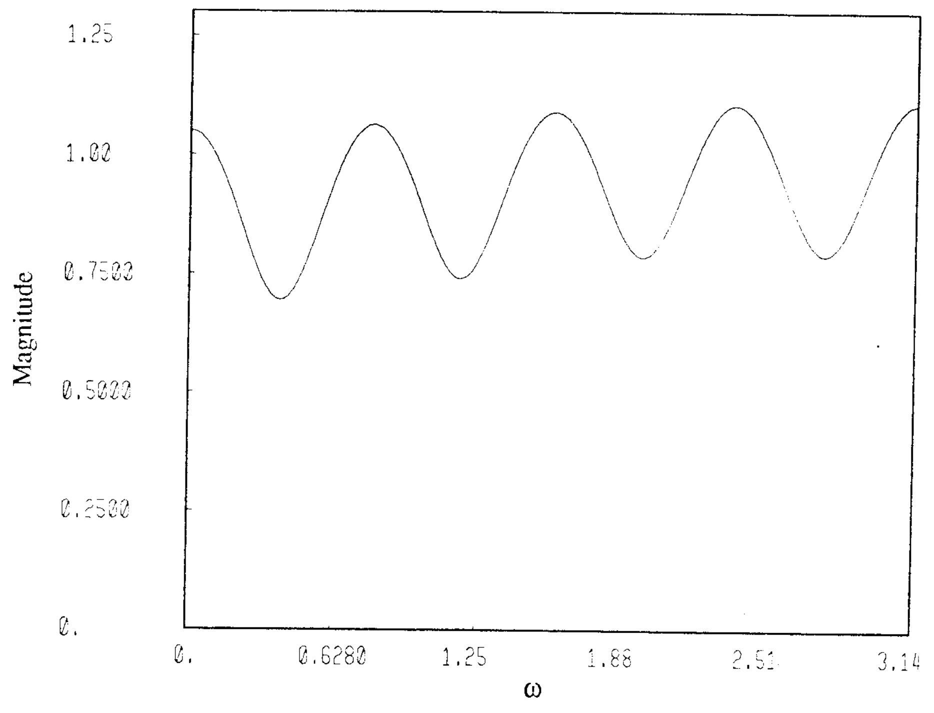


Figure 4.1: Magnitude Response of a Selective Spectral Attenuation Filter

location of its zeros (and poles, if rational). In general, a function with two or more variables cannot be factored into linear factors only.

Alternatively, some attenuation at other places may be tolerable in order to keep the filter transfer function simple. We mention three ways to design such a function. A two-dimensional filter with a transfer function $\boldsymbol{H}(z_h, z_v)$ is said to be *separable* [165, p.441] if the transfer function can be expressed as a product of one-dimensional functions,

$$\boldsymbol{H}(z_h, z_v) = \boldsymbol{G}_1(z_h)\boldsymbol{G}_2(z_v). \tag{4.2}$$

If the above relation does not hold, the filter is *nonseparable.*

It is possible to design a separable two-dimensional filter by simply taking the product of a one-dimensional filter with itself ($\boldsymbol{G}_1 = \boldsymbol{G}_2$). We used the one-dimensional FIR filter described in (4.1) to be the basic filter. The attenuation valleys in the magnitude response of this filter run along lines parallel to the axes. Therefore this filter is called the *rectangular* filter. The resulting FIR filter is of length 9 × 9, and the transfer function is of the form

$$\begin{aligned}\boldsymbol{H}_r(z_h, z_v) = \prod_i \Big[1 &- 2r_i \cos\theta_i (z_h^{-1} + z_v^{-1}) + r_i^2 (z_h^{-2} + z_v^{-2}) + 4r_i^2 \cos^2\theta_i z_h^{-1} z_v^{-1} \\ &- 2r_i^3 \cos\theta_i (z_h^{-1} z_v^{-2} + z_h^{-2} z_v^{-1}) + r_i^4 z_h^{-2} z_v^{-2} \Big].\end{aligned} \tag{4.3}$$

Note that if the VQ blocks are not square but rectangular, this type of filter can still be designed. The only difference is that the two one-dimensional components will be different in that case. This flexibility is not there in the next two filters, as we shall shortly see.

Using the *McClellan transformation* [165, pp. 472–478], a one-dimensional FIR filter can be transformed into a two-dimensional nonseparable FIR filter. The following mapping in frequency from one to two dimensions is used:

$$\cos\omega = \frac{1}{2}\cos\omega_h + \frac{1}{2}\cos\omega_v + \frac{1}{2}\cos(\omega_h\omega_v) + \frac{1}{2}. \tag{4.4}$$

It is required that the one-dimensional filter be linear-phase, so that the transfer function can be expressed as a function of $\cos\omega$ alone. Once this is done, substituting relation (4.4) gives the two-dimensional transfer function. Following the mapping in (4.4), the curves of constant ω run approximately circularly around the origin in the two-dimensional frequency plane. Therefore, the attenuation valleys are also circular, and the filter is called a *circular* filter. A linear-phase basic filter is required in order to design a circular filter for our purpose, to which we now turn our attention.

The basic filter designed in (4.1) can be converted to a linear-phase one by adding an additional pair of zeros $\frac{1}{r}e^{j\theta}$, $\frac{1}{r}e^{-j\theta}$ for each pair $re^{j\theta}$, $re^{-j\theta}$ already present. Thus, the two-dimensional circular filter will have a transfer function of the following form:

$$\begin{aligned}
\boldsymbol{H}_c(z_h, z_v) = \prod_i \Bigg[& \frac{1}{16}(1 + z_h^{-4} + z_v^{-4} + z_h^{-4}z_v^{-4}) + \frac{1}{4}(z_h^{-1} + z_v^{-1} + z_h^{-3} + z_v^{-3} \\
& + z_h^{-1}z_v^{-4} + z_h^{-4}z_v^{-1} + z_h^{-3}z_v^{-4} + z_h^{-4}z_v^{-3}) \\
& - \frac{1}{2}(r_i + \frac{1}{r_i})\cos\theta_i(z_h^{-1}z_v^{-1} + z_h^{-1}z_v^{-3} + z_h^{-3}z_v^{-1} + z_h^{-3}z_v^{-3}) \\
& + \frac{3}{8}(z_h^{-2} + z_v^{-2} + z_h^{-2}z_v^{-4} + z_h^{-4}z_v^{-2}) \\
& - \left(\frac{1}{2} + (r_i + \frac{1}{r_i})\cos\theta_i\right)(z_h^{-1}z_v^{-2} + z_h^{-2}z_v^{-1} + z_h^{-2}z_v^{-3} + z_h^{-3}z_v^{-2}) \\
& + \left(\frac{1}{4} + r_i^2 + \frac{1}{r_i^2} + 4\cos^2\theta_i + 2(r_i + \frac{1}{r_i})\cos\theta_i\right) z_h^{-2}z_v^{-2}\Bigg]. \qquad (4.5)
\end{aligned}$$

The two-dimensional linear-phase filter has a length of 17×17. Note that to get the same amount of attenuation, the values of r_i in (4.5) will not be the same as the values of r_i in (4.3).

We suggest instead another simple mapping in frequency from one to two dimensions:

$$\omega = \omega_h + \omega_v. \qquad (4.6)$$

In this mapping the lines of constant ω are straight lines in the two-dimensional frequency plane with a slope of -1, which preserves our desirable zero positioning along both axes. The attenuation valleys run in straight lines parallel to the above slope. Therefore, we call it a *striped* filter. Except for trivial cases, this filter is nonseparable. No linear phase requirement exists because the transformation from one to two dimensions is so simple. Using the basic filter of (4.1), the resulting filter length becomes 9×9, and the transfer function is

$$\boldsymbol{H}_s(z_h, z_v) = \prod_i \left[1 - 2r_i \cos\theta_i z_h^{-1} z_v^{-1} + r_i^2 z_h^{-2} z_v^{-2}\right]. \tag{4.7}$$

Each term of the form $z_h^{-i} z_v^{-k}$, $i \neq k$, in the above transfer function multiplied out is zero. So if the transfer function coefficients are arranged in a matrix, it will be a diagonal matrix. Thus, an $n \times n$ striped filter has only n non-zero coefficients, as opposed to n^2 non-zero coefficients in a general rectangular or circular filter. The circular filter can, however, be implemented using only $\mathcal{O}(n)$ multiplications [166].

Figure 4.2 shows the magnitude response of the above three types of filters, namely, the rectangular, circular, and striped filter. We have applied these filters on VQ decoded images to compare their performances. For equal amount of attenuation, these filters behave quite similarly. The circular filter is found to be marginally better against blockiness. Apart from reducing blockiness, the filtering also affects sharp edges in the picture, creating mild shadows. The circular filter creates shadows in all directions. The rectangular filter creates shadows in horizontal and vertical directions only. The striped filter does it along a line of slope -1 only. Therefore, the least number of sharp edges are affected by the striped filter. Moreover, the striped filter has the least number of non-zero coefficients in its transfer function, and can be implemented more easily than either of the other two filters.

Figure 4.3 shows the filtering results with the striped two-dimensional selective

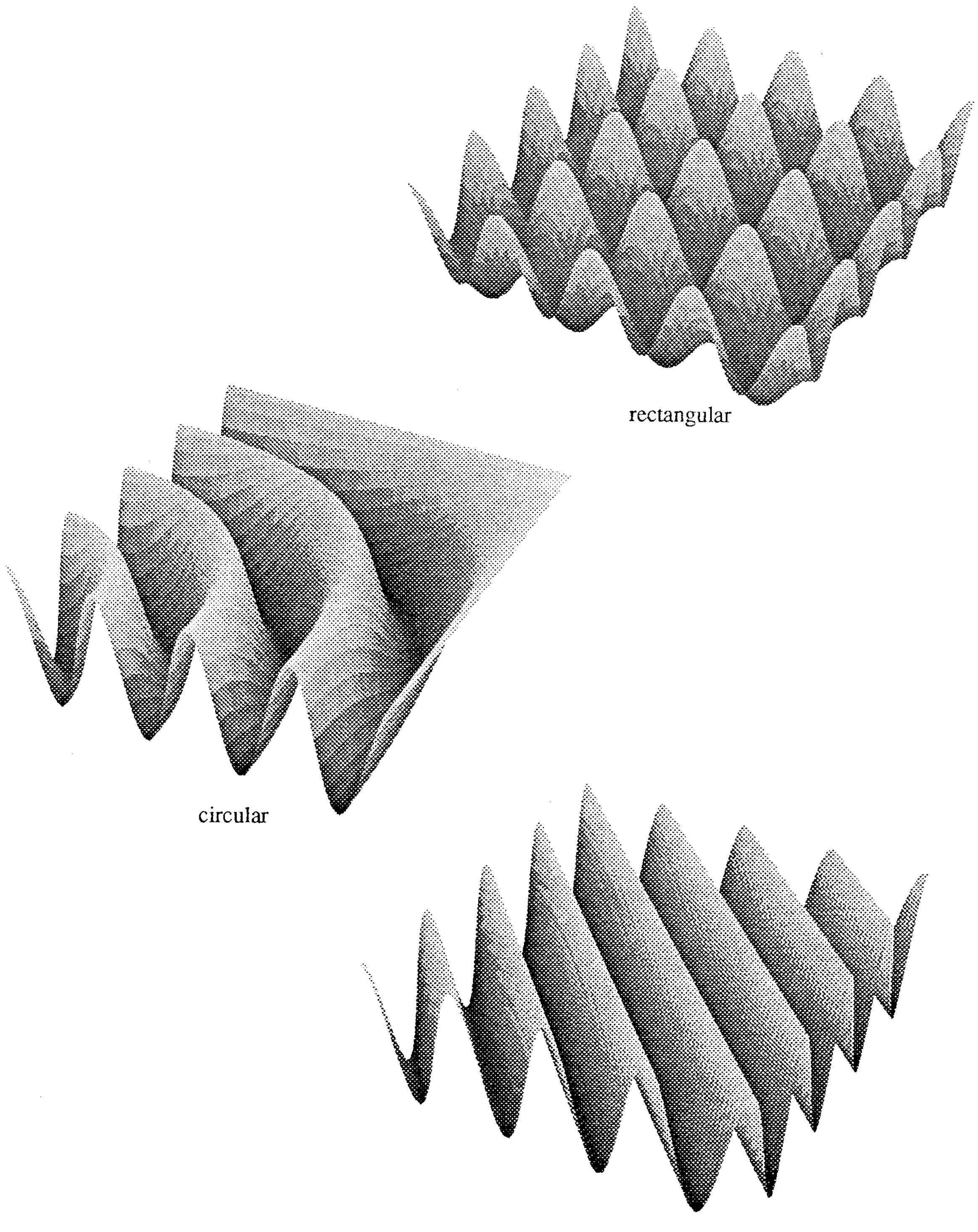
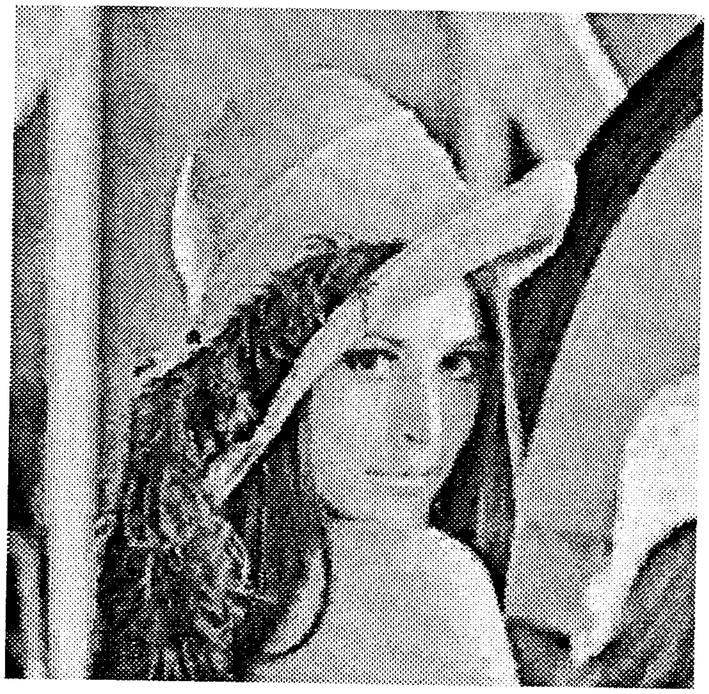
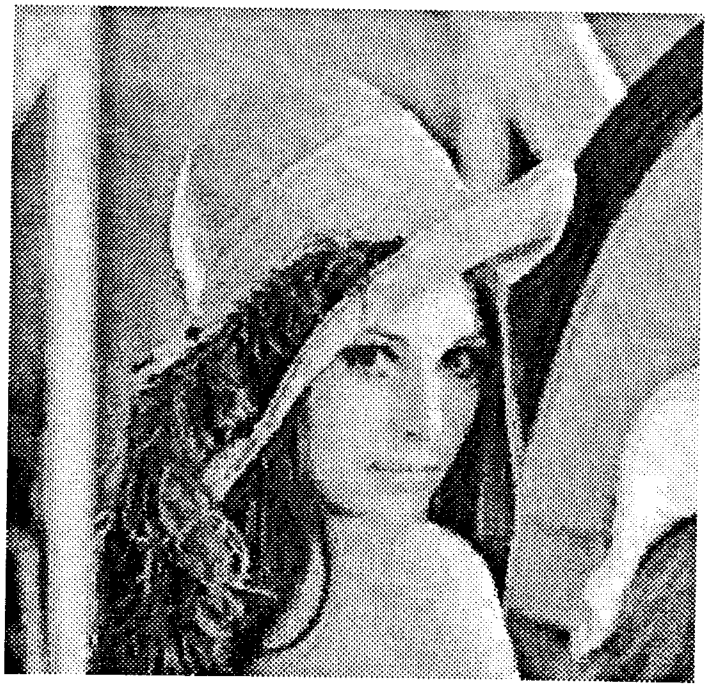


Figure 4.2: Magnitude Response of Two-Dimensional Filters

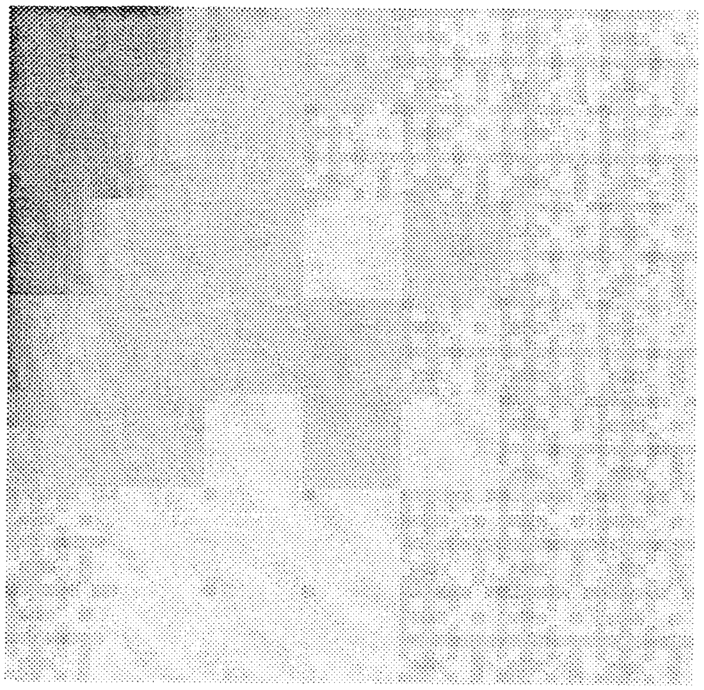


(a)



(b)

Figure 4.3: Filtering Results:
(a) Decoded Image before Postfiltering;
(b) Decoded Image after Postfiltering;


(c)

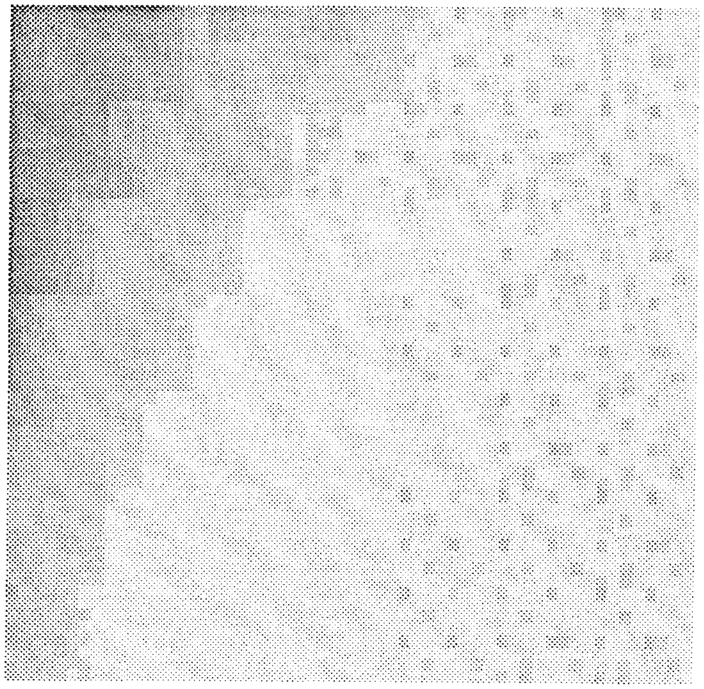

(d)

Figure 4.3: (Continued):
(c) Enlarged Version of a Part of (a);
(d) Enlarged Version of a Part of (b).

spectral attenuation filter. While figure 4.3(a) is the decoded image from a low rate VQ coder without any processing, figure 4.3(b) is the filtered version. The blockiness in the second image is visibly less. This effect is better seen in the enlarged part of the images, figures 4.3(c) and 4.3(d).

4.3 Use of the Prefiltered Codebook

Any kind of filtering, such as described in the previous section, has the disadvantage that it affects the original image to some extent. An ideal filter should attenuate the coding error but not any part of the original information. However, because of the overlap of image information and coding error in the spectral plane, it is not possible to separate them out with a linear filter. As a result, even without the presence of quantization noise, if an image is filtered by such a filter, the result will generally not be the same.

In order to compensate to some extent for this fact, we propose the use of a prefilter in the encoder. The idea is that the source image is prefiltered by a filter, $\boldsymbol{H}_e(z)$, before encoding. The decoded image is then postfiltered as in Section 2.2 by the decoder filter, $\boldsymbol{H}_d(z)$. The encoder filter is an inverse filter of the decoder filter, i.e., the product of these two filters is (or, resembles) an all pass filter. Thus, the postfiltered image is expected to be identical to the source image when no coding error has occurred. The sacrifice made here is that each frame needs to be filtered twice, once in the encoder and once in the decoder.

Because we are interested in fast coding time, an alternative to a second filtering has been sought. We suggest that the encoder codebook be prefiltered instead. This *prefiltered codebook* is stored in the decoder in place of the original codebook. The decoded image is reconstructed from the prefiltered codebook before it is postfiltered. Thus, the requirement of prefiltering the source image has been abolished. If a single full-search encoder-decoder unit is to be implemented in the conventional way, our scheme does require some extra memory to store the prefiltered codebook. However, for the designs where the encoder and decoder codebooks are implemented separately, such as the tree-structured encoder or the neural design suggested in the next chapter, no extra memory will be required. Prefiltering the codebook can be done off the line.

The two-dimensional prefilter has to be designed to fit the postfilter we have designed in the previous section. However, we are required to design only the basic one-dimensional encoder filter from the decoder filter transfer function. Extending it to two dimensions can be done as before.

Given an FIR selective spectral attenuation postfilter with a transfer function $\boldsymbol{H}_d(z)$, we desire to construct the encoder prefilter $\boldsymbol{H}_e(z)$ such that the following condition is approximately satisfied:

$$|\boldsymbol{H}_e(e^{j\omega})\boldsymbol{H}_d(e^{j\omega})| = 1. \tag{4.8}$$

An exact solution is possible if we allow the encoder filter to be IIR, in which case $\boldsymbol{H}_e(z) = 1/\boldsymbol{H}_d(z)$. This filter will be stable only if all the zeros of $\boldsymbol{H}_d(z)$ lie within the unit circle, $|z| = 1$. That is the case when $\boldsymbol{H}_d(z)$ is a minimum-phase filter, such as in (4.1) if r_i is less than 1 for all i. But if the decoder is a linear-phase filter, as is required for the McClellan transformation, some zeros lie outside the unit circle, and the IIR encoder filter will be unstable.

Due to the difficulty in implementing two-dimensional IIR filters, it is desirable that the decoded filter has a finite impulse response. We show that, because the magnitude of the decoder filter is close to 1, it is indeed possible to design an FIR encoder filter [Appendix II] such that

$$|\boldsymbol{H}_e(e^{j\omega})|^2 + |\boldsymbol{H}_d(e^{j\omega})|^2 = 2, \tag{4.9}$$

which is an approximation to the property of (4.8). This design procedure works for both minimum- and linear-phase filters. Figure 4.4 shows the magnitude response of such a minimum-phase FIR encoder filter, and compares it with the ideal IIR magnitude response.

Our simulations show that for the same two-dimensional filter, a prefiltered, decoded, and postfiltered image actually offers less subjective improvement over an image decoded using the prefiltered codebook and then postfiltered. Suppression of blockiness is better with the prefiltered codebook scheme than with the image prefiltering scheme. Moreover, the edges are free of any staircase effect when the prefiltered codebook is used. Therefore, our scheme of using the prefiltered codebook is subjectively superior than the image prefiltering scheme.

Figure 4.5 shows the results of using a prefiltered codebook. As earlier, the striped filter offered comparatively better quality because of less shadowing at the edges. While figure 4.5(a) is the postfiltered image from a normal codebook, figure 4.5(b) is the postfiltered image from the prefiltered codebook. Figures 4.5(c) and 4.5(d) are enlarged versions of the shoulder of Leena to realize the clear improvement offered by the prefiltered codebook. The filter used here is the two-dimensional striped selective spectral attenuation filter.

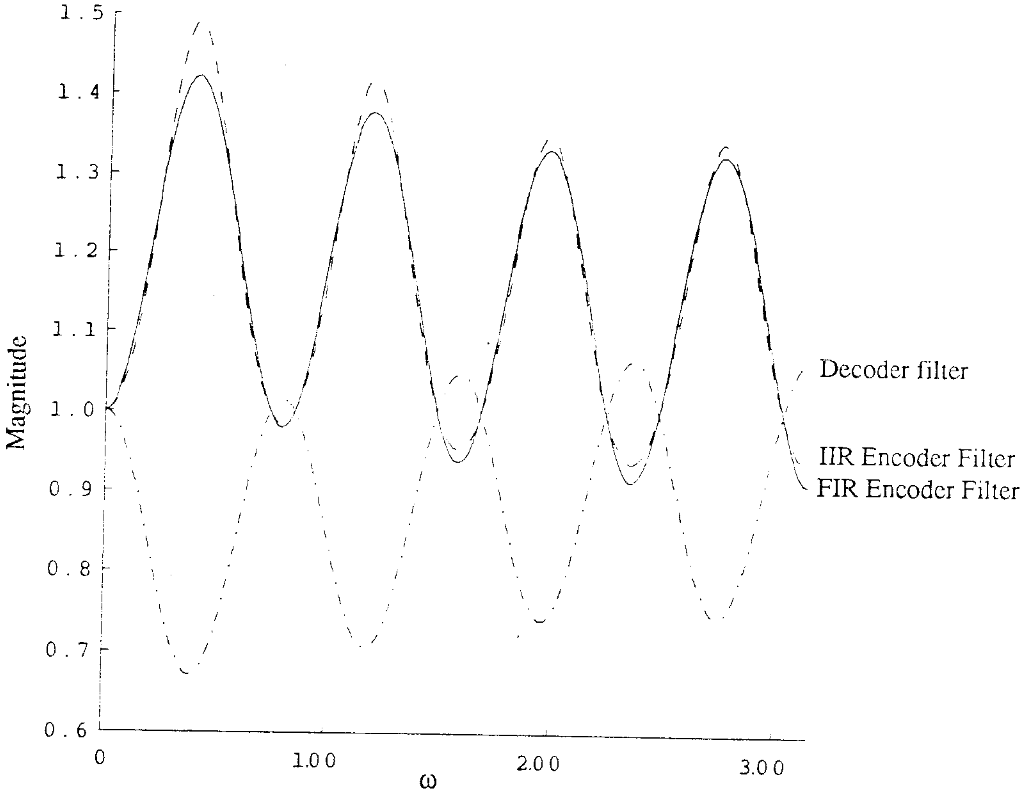
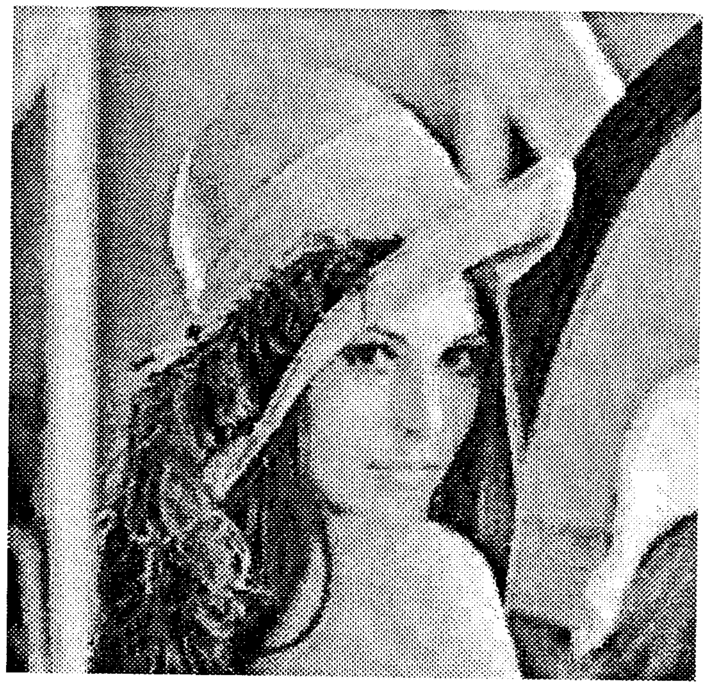
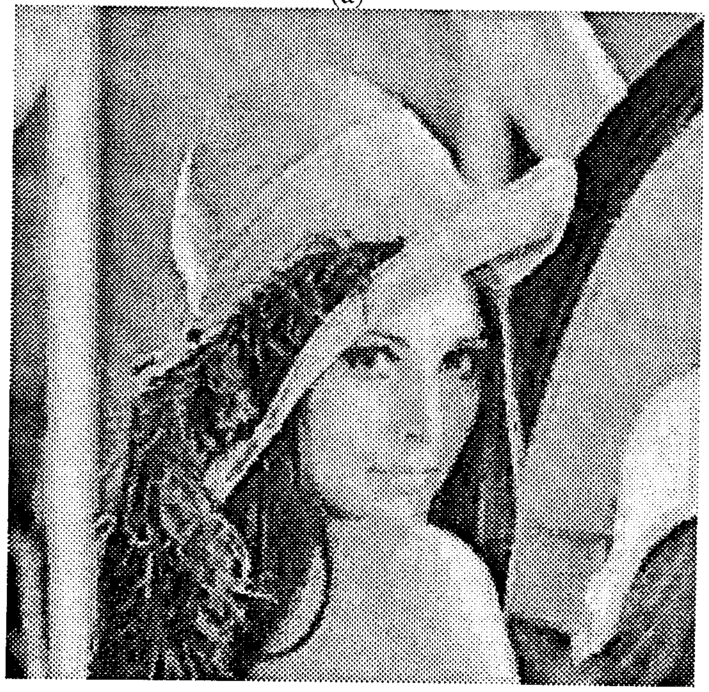


Figure 4.4: Magnitude Response of Encoder Filters

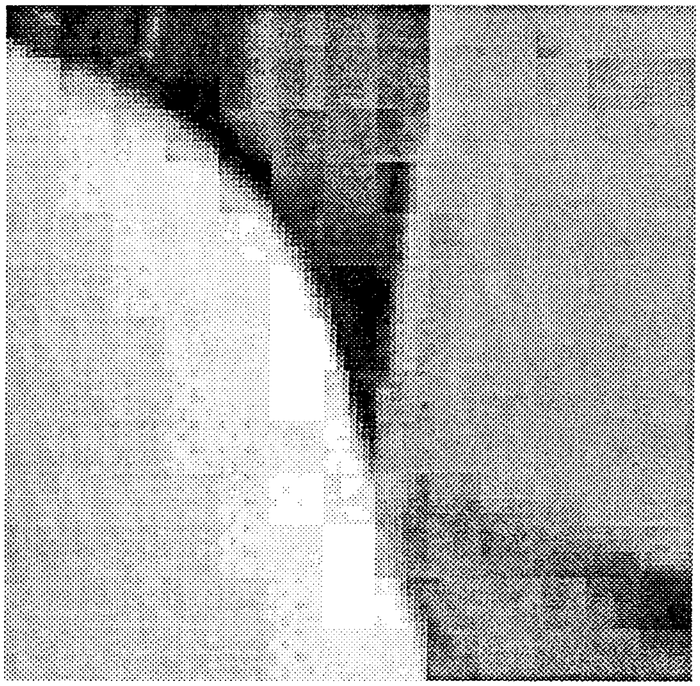


(a)



(b)

Figure 4.5: Filtering Results with Prefiltered Codebook:
(a) Postfiltered Image using Normal Codebook;
(b) Postfiltered Image using Prefiltered Codebook;


(c)

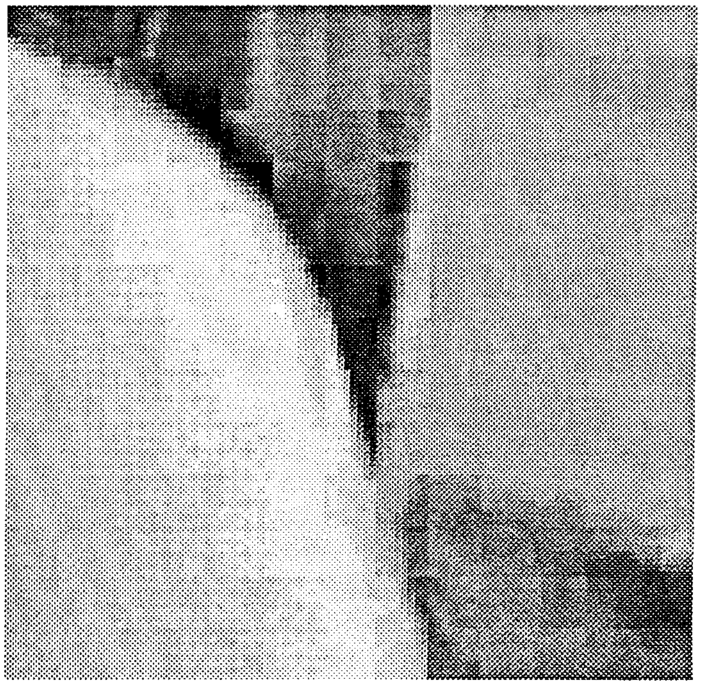

(d)

Figure 4.5: (Continued):
(c) Enlarged Version of a Part of (a);
(d) Enlarged Version of a Part of (b).

Chapter V
IMPLEMENTATIONS USING NEURAL NETWORKS

In this chapter we shall look into the implementation issue of a vector quantization coder. Traditional vector quantization encoders for image coding are computationally very intense. In order to implement a real-time encoder for this purpose, we consider a novel approach by investigating the use of neural networks. The pattern-matching task of a VQ encoder seems suitable for a neural circuit to solve, as we show in more detail later. A 3-layer feed-forward network has been suggested in the next section, which can realize a binary tree-structured VQ encoder using $\mathcal{O}(N)$ neurons (N = number of codewords) and $\mathcal{O}(N(\log N + K))$ synapses (K = number of dimensions). Also, a feedback network, which requires $\mathcal{O}(N)$ neurons and $\mathcal{O}(NK)$ synapses, is described to implement a full-search VQ encoding scheme.

5.1 Neural Networks in Vector Quantization

An electronic, or artificial, *neural network* is a massively parallel interconnected network of simple processing elements, or *neurons*, that can carry out information processing using simple operations on numerous inputs as biological nervous systems do. Although one of the most important features of some neural networks is their learning ability, there exists another category of time-invariant (non-learning) neural networks which are becoming popular as efficient parallel computing machines. A neural network designed to solve certain problem can be viewed as a custom-made, therefore very efficient, parallel-architecture computer.

The self-organizing ability found in an adaptive neural network is extensively used in different kinds of learning problems. *Clustering* or unsupervised learning is an extensively studied example. The objective in this class of problem is to divide the training patterns into a number of geometrical clusters. During the learning process, no information regarding the cluster membership of the training patterns is provided, either because it is not known or because of the high cost involved in doing so. The process of generating a codebook from the training sequence in VQ is an example of a clustering problem. The popular LBG algorithm [164] used for designing a VQ codebook is functionally same as the *isodata* algorithm [167]. Isodata is an iterative version of the original recursive c-means (also known as k-means) algorithm for clustering, long known in pattern recognition.

The application of neural unsupervised learning algorithms for vector quantization has been suggested, explicitly or implicitly, in a number of neural network articles. Recently several such attempts have been reported. In [168] three different neural unsupervised learning techniques have been compared with the LBG algorithm to design a speech VQ coder. Though some neural algorithms are found to generate codebooks having more balanced codeword utilization than the LBG codebook has, the LBG performance is never bettered by these codebooks. In [169] a similar comparison is reported between the LBG algorithm and an well-known neural learning scheme in case of images. For images outside the training sequence, the LBG codebook performed marginally better.

Apart from the fact that the neural learning schemes may not offer any improvement, it is not clear how such a system could be implemented in hardware. For a neural codebook to be adaptive, the weights of the network have to change. Moreover, because these weights may not directly represent the codewords, one has to find a way to compute the resulting changes in the codewords and transmit these changes to the decoder. If the learning is done off the line and the network is 'frozen' before using it, then it is inefficient because some part of the network is used only while learning, and remains unused in the encoding process.

However, the interconnectivity of a neural network can still be used to our advantage. Neural networks have been found to be suitable for pattern recognition applications. The vector quantization encoding process is nothing but an example of the classic *classification* problem, where the task is to map each input vector to one of N pre-defined classes. A neuron implements a linear discriminant function (LDF) capable of solving a binary classification problem when the decision boundary is a hyperplane. More complex decision boundaries and multi-class solutions can be achieved using a multi-layer network. Therefore, it seems natural to attempt to design a VQ encoder using a neural network.

There is another reason one should consider the neural implementation of a VQ coder. The major obstacle which has limited the use of vector quantization for practical purpose is the difficulty in implementing the computationally demanding codebook-search algorithm in the encoder in order to find out the best match. Considering an encoder for K-dimensional vectors and N codewords, for an exhaustive search one has to compute N distortion functions, each of which needs K multiplications and $2K-1$ additions for squared-error distortion measure. This means that for every K-pixel input vector, the arithmetic task consists of NK multiplications, $N(2K-1)$ additions and $N-1$ comparisons. The computation required is thus $\mathcal{O}(NK)$ here. It is easy to realize that, in order to achieve real-time encoding, one has to use parallel processors. Fortunately the VQ encoding algorithm is inherently parallel. Therefore, a neural network can indeed be a solution.

In general, it is easier to design a VQ encoder for speech coding. The image coding problem is considered more difficult because of the larger block size, K, as well as a larger codebook, N. However, the resolution $r = (\log N)/K$ required for an image is smaller than that required for speech, so the compression ratio can be higher.[1]

Attempts have been made to implement a real-time VQ encoder for image coding in hardware. One such VLSI implementation, reported in [170], uses parallelism and pipelining to perform vector operations, thus requiring only $\mathcal{O}(N)$ time for a full search. Note that performing the distortion computations in parallel is not easy to achieve, because it demands simultaneous memory addressing, which in turn requires either extremely fast memory or an enormous amount of storage.

Recently, a forward-only counterpropagation neural network has been proposed to implement a VQ encoder [171]. The network has three layers. The input layer contains K fanout neurons that simply multiplex the input signal. The middle layer consisting of N neurons is a portion of the self-organizing feature map of Kohonen [172]. The final layer is the outstar structure of Grossberg with N neurons [173]. The Kohonen units need to be fully interconnected with feedback to realize the encoder. This network is trained using supervised learning, which needs a pre-designed VQ codebook. After the training, the network would be capable to perform as well as a full-search VQ encoder. This network uses $\mathcal{O}(N+K)$ neurons and $\mathcal{O}(N(N+K))$ synapses.

5.2 Realization of the Tree-Structured Encoder

Consider a VQ codebook with N codewords, $C^{(1)}$, $C^{(2)}$, ..., $C^{(N)}$, each of dimensionality K. Consider a hypothetical case where an input vector X is to be mapped to either of the two codewords, $C^{(m)}$ or $C^{(n)}$, and $m \neq n$. For the

[1] Throughout this chapter $\lceil \log_2 N \rceil$ is abbreviated as $\log N$.

squared-error distortion measure, from equation (1.4),

$$d(X, C^{(m)}) = \sum_{k=1}^{K} (x_k - c_k^{(m)})^2 \quad \text{and} \quad d(X, C^{(n)}) = \sum_{k=1}^{K} (x_k - c_k^{(n)})^2. \tag{5.1}$$

We will map X to $C^{(m)}$ only if

$$\begin{aligned} d(X, C^{(m)}) &< d(X, C^{(n)}) \\ \Rightarrow d(X, C^{(m)}) - d(X, C^{(n)}) &< 0 \\ \Rightarrow \sum_{k=1}^{K} \left[(x_k - c_k^{(m)})^2 - (x_k - c_k^{(n)})^2 \right] &< 0 \\ \Rightarrow \sum_{k=1}^{K} \left[-2x_k c_k^{(m)} + c_k^{(m)^2} + 2x_k c_k^{(n)} - c_k^{(n)^2} \right] &< 0 \\ \Rightarrow \sum_{k=1}^{K} x_k (2c_k^{(n)} - 2c_k^{(m)}) + \sum_{k=1}^{K} (c_k^{(m)^2} - c_k^{(n)^2}) &< 0 \\ \Rightarrow \sum_{k=1}^{K} w_k x_k + w_o &< 0, \end{aligned} \tag{5.2}$$

where $w_k = 2c_k^{(n)} - 2c_k^{(m)}$, for $k = 1, \ldots, K$, and $w_o = \sum_k (c_k^{(m)^2} - c_k^{(n)^2})$.

Therefore, thinking of this as a classification problem with two classes represented by $C^{(m)}$ and $C^{(n)}$, the decision rule is

$$\begin{aligned} &\text{if } \sum_k w_k x_k + w_o < 0 \implies X \text{ mapped to } C^{(m)} \\ &\text{if } \sum_k w_k x_k + w_o \geq 0 \implies X \text{ mapped to } C^{(n)}. \end{aligned} \tag{5.3}$$

As has been mentioned before, this type of binary classification problem for a K-dimensional vector X can be solved using a neuron as shown in Figure 5.1. The neuron has K inputs $x_1, \ldots, x_K$, K weights $w_1, \ldots, w_K$, an adder and a monotonic (sigmoid or binary) function $f_S(\cdot)$ with threshold w_o. A positive output represents $C^{(n)}$ and a negative output represents $C^{(m)}$. In case of image coding, the input values, X_k, could be analog (before digitization), and the weights, w_k, can take discrete values within a fixed dynamic range.

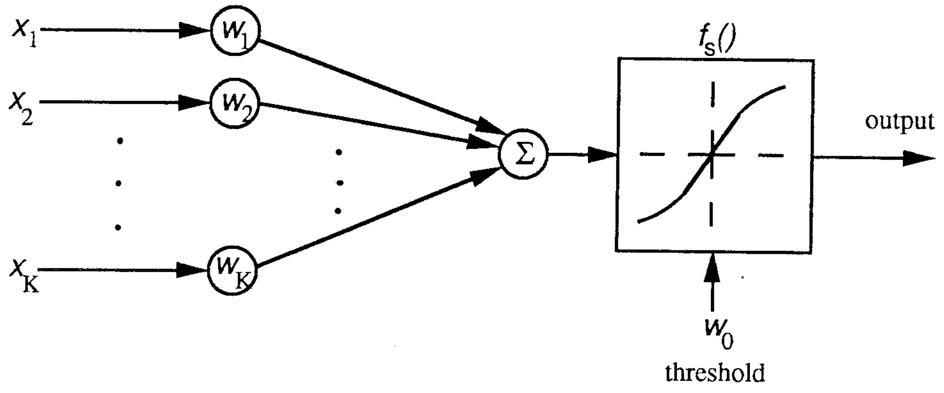


Figure 5.1: Neuron

The encoding procedure for a binary tree-structured VQ codebook is a repetitive binary classification process. Each node of the search-tree consists of two intermediate codewords, one of which has to be selected. We can implement such a binary classifier using one K-dimensional neuron (in other words, one neuron with K *synapses*). The final codewords, i.e., the leaves of the tree, are not needed during encoding. A binary tree of any type with N leaves has $N-1$ nodes. The neurons corresponding to these nodes will be arranged in parallel, unlike the tree. The control flow along the levels of the tree will be implemented in the second layer. The nonlinearity of the neurons will be a binary threshold function, such that the outputs take one of only two possible values. Thus, the first layer of the encoder will consist of $N-1$ classifier neurons and $(N-1)K$ synapses.

An index-generator circuit will follow the first layer of neurons to produce the $\log N$-bit index from the decision outputs of these neurons. Because the outputs from the previous layer are binary, a digital circuit with $\mathcal{O}(N \log N)$ two-input boolean gates can be designed to implement this index-generator. In Figure 5.2 we demonstrate another way of realizing the generator for a complete binary tree using a two-layer neural network. The tree hierarchy is implemented in the second layer, such that the outputs of the neurons belonging to a given level are enabled or disabled by the outputs from the higher levels. The individual bits of the index will be generated in the third layer as shown. Alternatively, it is possible to generate the analog value of the index using only one neuron in the third layer, in which case an analog-to-digital converter may follow to generate the bits. However, the alternative design uses the same number of synapses as before, and does not offer any advantage. For a complete tree, this network uses $N + \log N - 3$ neurons. The dimensionality of these neurons varies as opposed to the first layer, the maximum dimension being $\frac{N}{2}$. A total of $N \log N - 2$ synapses are required, which is of the same order as that of the boolean gates mentioned above.

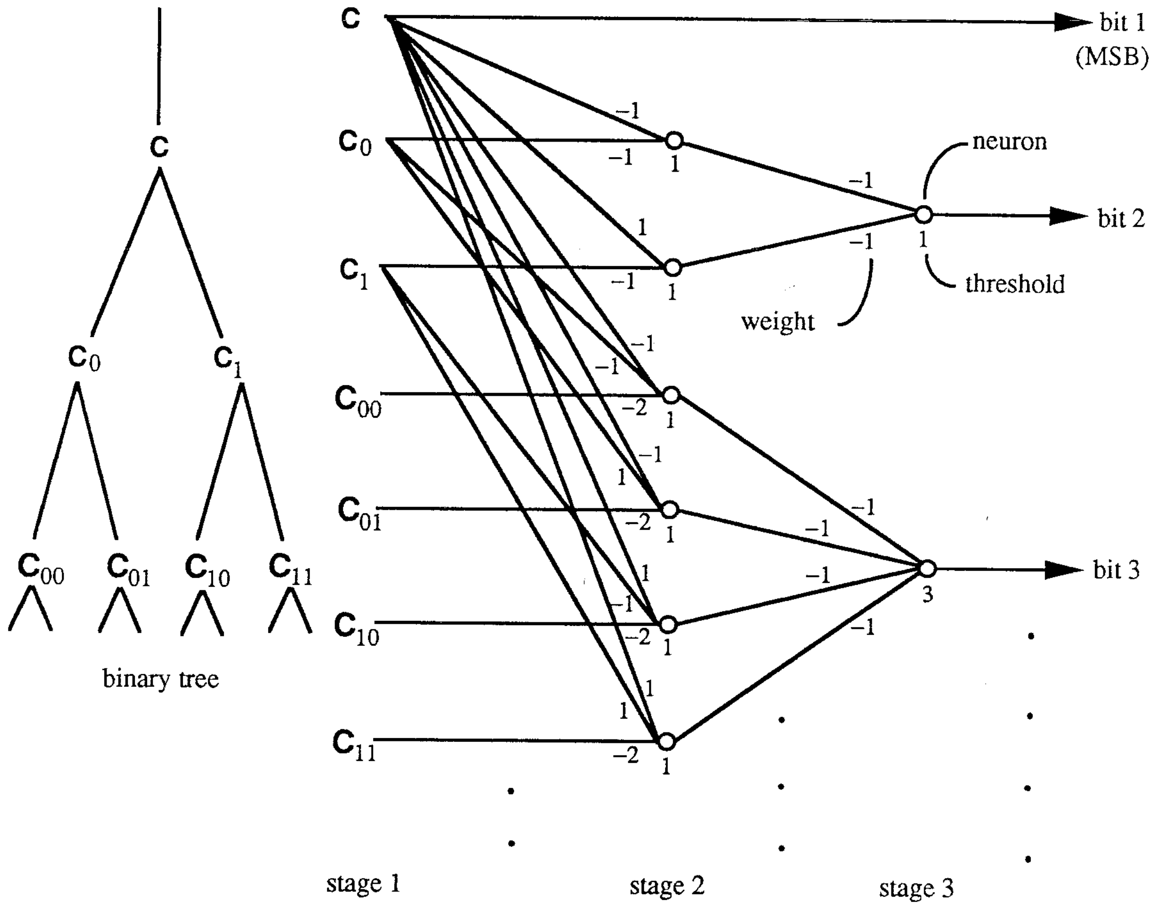


Figure 5.2: Index-Generation from Classification Results

The schematic diagram of the complete encoder is shown in Figure 5.3 when the tree is complete. The design uses a three-layer forward-only network to generate the index from the input vector. The initial layer consists of the classifier neurons. The remaining two layers compute the index from the classification results. The neuron count of this network is $\mathcal{O}(N)$, and the synapse count is $\mathcal{O}(N(\log N + K))$.

The most obvious advantage of our neural design is that the encoding time for this scheme is roughly the sum of the time delays of the three layers, i.e., the settling time of three neurons. This is possible because no feedback is involved, either from a layer to another, or within a layer itself. Thus we could safely assume that the encoding time, to first order, does not depend on N or K. There may, however, be some small dependency, because the delay in a neuron may to some extent depend on its dimensionality.

The weights, w_k, and the threshold value, w_o, of the first-layer neurons can be found in two ways. Using the analytical results given in (5.2), these values can be determined from a pre-designed codebook. This fact allows us to choose any appropriate clustering algorithm to design the codebook. On the other hand, the weights and thresholds could be trained using any unsupervised learning technique, either the whole network together or one node at a time in a hierarchical manner, in a way similar to the schemes suggested in [168]. It is not possible to make these weights adaptive because the codewords cannot be exactly determined from the weights.

A tree-structured codebook is usually sub-optimal in performance. However, due to the non-uniform distribution of the vectors, a variable-depth tree-structured VQ has been found to outperform even a full-search VQ in case of speech coding [174]. In conventional implementations a variable-depth tree requires a variable amount of search, which results in a variable encoding time. In our neural scheme

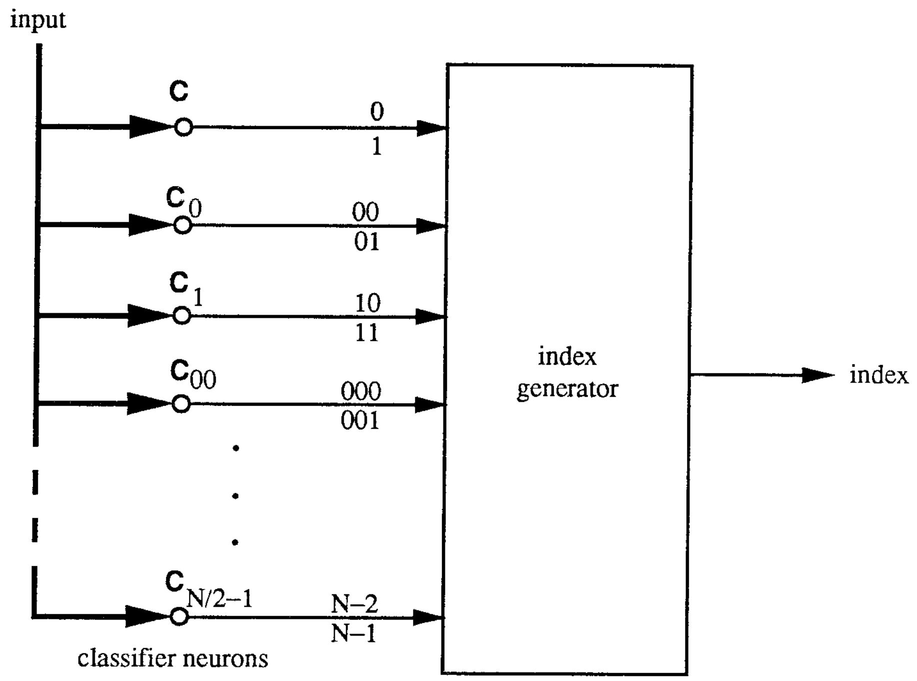


Figure 5.3: Binary Tree-Structured VQ Encoder

the encoding time is fixed, and a variable-depth tree could be realized with equal ease as a fixed-depth tree.

It is worthy noting that among the various kinds of VQ tree-encoders, the binary tree appears to be always optimal for neural implementation. To implement a non-binary tree, each first-layer node requires more than one neuron, which in turn increases the required number of neurons and synapses in the following layers proportionately. This is in contrast to the fact that, as long as storage has a nonzero cost, a quad tree is always superior to a binary tree for conventional hardware implementations. In conventional case, for a complete m-ary tree of depth d ($N = m^d$), the encoder must compute md distortion functions and store $\sum_{i=1}^{d} m^i$ intermediate codewords. When N is an integer power of 2, $m = 2$ and 4 yields the minimum number of distortion computations. But when $m = 4$, only two-thirds the storage is necessary compared to the binary tree encoder.

5.3 Realization of the Full-Search Encoder

Consider a codebook with N codewords $C^{(1)}, C^{(2)}, \ldots, C^{(N)}$ and a full-search encoder. For an input vector X, the encoder first computes the N distortion measures,

$$d^{(n)} = d(X, C^{(n)}), \qquad n = 1, \ldots, N. \tag{5.4}$$

Then the encoder finds out the minimum of these distortions (any one of the minima in case of a tie),

$$d_{min} = \min_{n} d^{(n)} = d^{(\hat{n})}, \tag{5.5}$$

and assigns $C^{(\hat{n})}$ to X, transmitting the label $\hat{n}$.

For K-dimensional vectors, we have

$$d^{(n)} = d(X, C^{(n)}) = \sum_{k=1}^{K}(x_k - c_k^{(n)})^2 = \sum_{k=1}^{K} x_k^2 + \sum_{k=1}^{K} c_k^{(n)^2} - 2\sum_{k=1}^{K} x_k c_k^{(n)}. \qquad (5.6)$$

Define

$$\tilde{d}^{(n)} = -\sum_{k=1}^{K} c_k^{(n)^2} + \sum_{k=1}^{K} x_k(2c_k^{(n)}) = \sum_{k=1}^{K} x_k(2c_k^{(n)}) + t^{(n)}, \qquad (5.7)$$

where $t^{(n)} = -\sum_k c_k^{(n)^2}$. It is obvious that finding the maximum among the $\tilde{d}^{(n)}$'s is the same as finding the minimum among the $d^{(n)}$'s, because both methods will select the same index, $\hat{n}$.

Observe that $\tilde{d}^{(n)}$ is a familiar form which could be computed by a neuron with K inputs $x_1, \ldots, x_K$, K weights $2c_1^{(n)}, \ldots, 2c_K^{(n)}$, a sum and a sigmoid nonlinearity $f_S(\cdot)$, so long as the input $\tilde{d}^{(n)}$ remains in the linear region of the sigmoid. Here $t^{(n)}$ would be the threshold to the neuron (Figure 5.1). With N such neurons in the initial layer, we could compute the modified distortions $\tilde{d}^{(n)}$ for every codeword paired with the input vector. Thus the initial layer consists of N neurons and NK synapses.

After the distortion values have been computed, the maximum of these have to be selected. For this purpose we suggest the use of a *winner-take-all* network. This type of network uses heavy lateral inhibition to hold a competition among the units, and the one with the strongest input wins. Other designs to pick the maximum from a set of inputs are also available [175], but typically require more layers or neurons. Since we are only interested in the index, the choice seems appropriate.

Winner-take-all networks have been developed and analyzed using different types of neural network models. Grossberg showed how a feedback competitive network behaves as a winner-take-all if the nonlinearity grows faster than a linear function [176]. A typical winner-take-all network with N external inputs has N neurons. Once the network is initialized by applying the inputs to the corresponding neurons, the external input is then withdrawn and the network is left to settle down to a stable output configuration. The output is also an N-dimensional vector. The lateral inhibition is generated by feeding back the output of every other neuron to the input of each neuron with a negative weight of ϵ. There also exists some excitatory feedback from each neuron to itself with a scaling factor of δ. The threshold of each neuron is set to zero. Lippmann calls this network a *maxnet* (Figure 5.4) when $\delta = 1$ and $\epsilon < \frac{1}{N}$ [175]. He also shows that, when the maxnet is allowed to iterate after the initialization until the output of only one neuron is positive (or, high), it will always converge and the only positive neuron is the one with the maximum input. In the VQ encoder, this is the neuron $\hat{n}$ corresponding to the maximum modified distortion $\tilde{d}^{(\hat{n})}$ or $I_{\hat{n}}$ at $t = 0$.

In case of two (or more) maximum inputs, the two corresponding output nodes will have a common positive value which is smaller than the output in the unique convergence case. However, in case of VQ-encoding it is highly unlikely that two maxima occurs simultaneously, i.e., then the integer input vector would lie on one of the Voronoi partitions.

The above configuration of maxnet requires $\mathcal{O}(N^2)$ synapses. Recently some groups have proposed equivalent networks requiring only $\mathcal{O}(N)$ synapses [177, 178]. In [177] an alternative configuration of a more general *k-winners-take-all* network (k=1 for our application) uses only $4N$ synapses and $N + 1$ neurons. In this alternative design $|\delta|$ is less than 1 (δ close to $+1$ for faster convergence) and ϵ is equal to 1. The first N nodes will initially receive external inputs. Each of these nodes will have a self-connection of weight $1 + \delta$. The sum of their outputs will be computed by the remaining node, whose output is fed back to the other N nodes of the maxnet through a weight of -1. The threshold value of each neuron is set to 0 except for the summing node, where it is set to $N - 2$. After the maxnet has been initialized with the distortion values through a set of switches, the switches

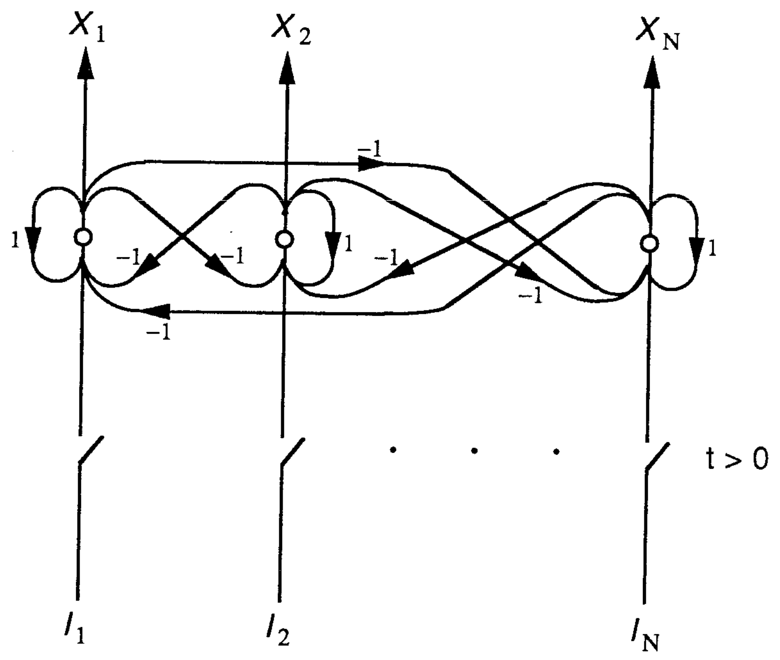


Figure 5.4: Maxnet

are opened and the network converges after a few iterations.

Because the maxnet outputs are not binary, a set of N neurons is used in the third layer as hard-decision thresholds. Only one of the N outputs from this layer will have a positive value, the rest all being zero. Once the set of binary outputs are available, the individual bits of the index can be generated from them using either a boolean circuit containing $\mathcal{O}(N \log N)$ two-input logic gates, or by an equivalent neural circuit having the same order of synapses. However, it is possible to generate the analog value of the index from the N outputs of the third layer by using only $\mathcal{O}(N)$ synapses, as shown in Figure 5.5. The fourth layer will then be succeeded by an analog-to-digital converter to digitize the selected index. Thus, the index-generation part of the network will require $N + 1$ neurons and $2N$ synapses.

The complete encoder, schematically drawn in Figure 5.6, consists of four layers of neurons. The initial layer of neurons computes the distortions. The second layer is a maxnet, which picks the unit with maximum input with the help of feedback. The last two layers generate the index to be transmitted. This scheme requires $\mathcal{O}(N)$ neurons and $\mathcal{O}(NK)$ synapses, which is an improvement over the scheme reported in [171]. By using the more efficient design of [177] in the Kohonen layer, it is possible to implement the encoder of [171] with only $\mathcal{O}(N(\log N + K))$ synapses, which is however more than what our scheme requires.

The encoding time of the above realization is the sum of three neuron response times (first, third and fourth layers) and the maxnet settling time. The last time contribution would probably increase as N goes up. Unlike the tree-structured encoder, this encoder has a layer with feedback connections, hence it may not have a negligible encoding delay. However, we expect this delay with neural implementation to be appreciably smaller when compared to the conventional hardware scheme. Fully interconnected maxnets with $N = 100$ have been observed to require typically

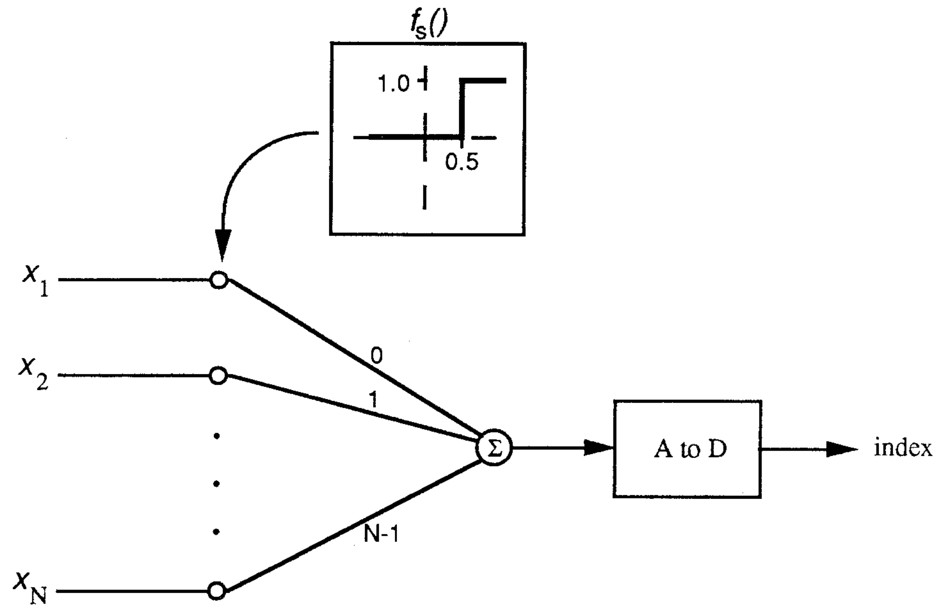


Figure 5.5: Index-Generation from Maxnet Outputs

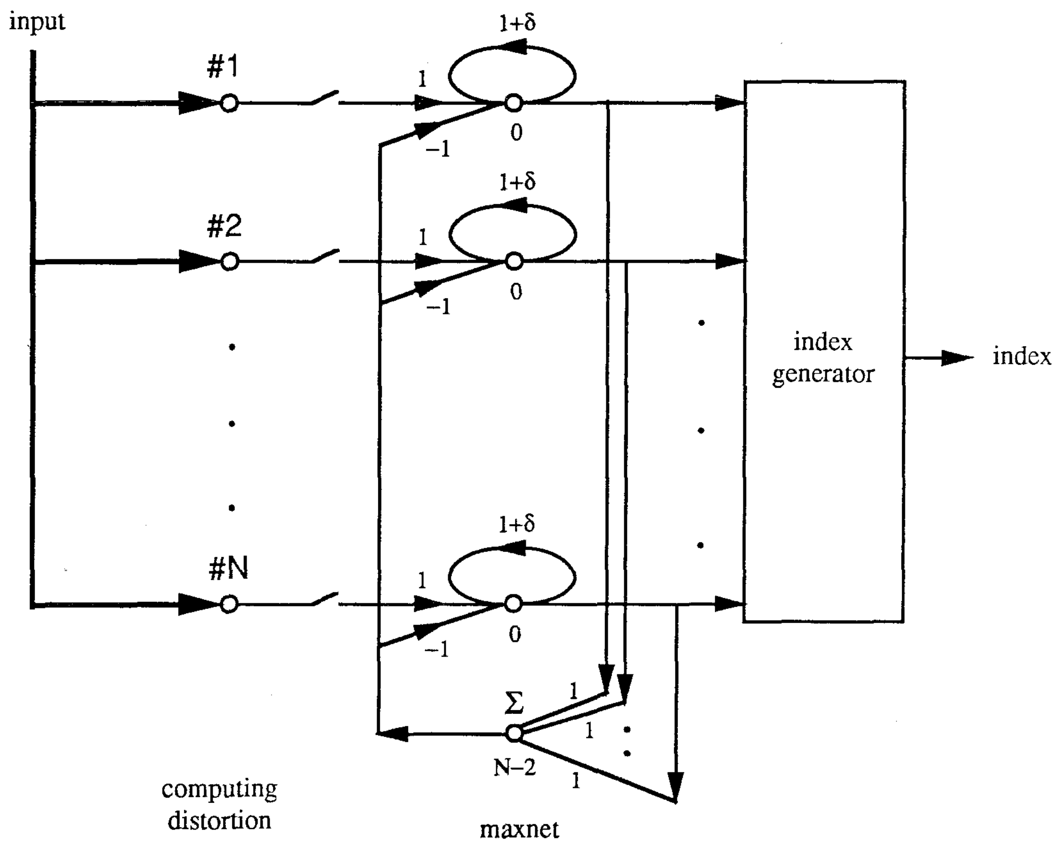


Figure 5.6: Full-Search VQ Encoder

three discrete iterations to distinctly identify the maximum input, and to converge in fewer than ten iterations [175].

The weights and the thresholds of the distortion-computing neurons of the initial layer can again be determined either analytically using (5.7) from a pre-designed codebook, or through unsupervised training. The weights of the other layers are pre-determined. Unlike the case of the tree-structured encoder, the first-layer weights (effectively, the codewords) in this case can be made adaptive and the information can be sent to the decoder with ease.

Because the typical number of codewords and the dimensionality in a VQ image coder is quite large, one is required to pack between 50,000 to 100,000 synapses into a single package in order to implement a high-compression full-search VQ image encoder. Due to the interconnectivity required, it is not plausible to distribute this circuit to more than one package. This task seems formidable with present-day technology. Therefore, some compromise may be in order here. [171] suggests the use of multi-stage VQ. In a similar fashion, an M-ary tree-structured VQ could be used. The codebook size of the individual VQ stages in the first case, or M in the second case, could be as large as 100. The resulting VQ coders in these cases may, however, be sub-optimal.

An alternative is to implement a tree-search VQ encoder [156] on the full-search codebook. For such a tree-search process, which preserves the optimality of the codebook, the decision function at each node is $C^T X + d$, from which we could design a neuron in the following way:

$$w_k = c_k \text{ for } k = 1, \ldots, K, \qquad w_o = d. \tag{5.8}$$

The tree part can thus be realized in a way similar to the tree-structured encoder scheme.

The leaves of a tree-search VQ are usually buckets consisting of a few codewords, through which a full search has to be performed. A separate full-search encoder, as described in this section, can be used for each leaf. The final index is a concatenation of the two indices generated from the tree and the chosen bucket. Due to the fact that the cardinality of a typical bucket is much less than N, this coder will need less encoding time than the full-search coder. The amount of saving will, however, entirely depend on the depth and the shape of the tree.

Chapter VI
SUMMARY AND CONCLUSIONS

In this thesis, different parts of a new type of vector quantization image compression scheme for low rate image coding have been discussed. Improvements have been achieved by using a new technique for extracting input vectors, a new class of distortion measures, some unique filtering techniques, and neural implementations in a VQ image coder.

In Chapter II, a new vector quantization technique has been described that uses distributed blocks instead of spatial blocks for encoding. The coder exploits the strong correlation between the distributed input blocks to reduce both the rate and the computation required. By making the partial ranges adaptive, it is easy to achieve a coder that is capable of adjusting its rate within a certain range while minimally compromising quality. It is shown that this coder outperforms the conventional VQ coder when coding gray-level images at low bit rate. Also, this coder has the effect of distributing the annoying quantization errors, thereby making them less visible to human eye. The coder complexity is almost the same as for the conventional scheme, and it needs no extra storage. Whether this coder will perform better than the conventional VQ coder is much dependent on the source statistics. The asymptotic results show that the new coder with two channels performs better than the VQ coder only if the vector dimensionality is small. However, it is shown to produce better quality even for a large dimensionality with a small codebook size when the source has enough redundancy.

In Chapter III an input-dependent distortion measure has been suggested which is capable of taking into account human psychovisual characteristics. Using this distortion measure, images with better subjective quality have been achieved when compared to the basic VQ coding results. The edge representation in a codebook has been improved, while still keeping the codebook optimal. The blockiness in the decoded image has been reduced as well. Moreover, the codewords are more uniformly utilized when the new distortion measure is used. This change of distortion function mostly affects the codebook design computation. When this distortion measure is incorporated in a VQ image coder, the encoding process remains the same in the neural design, or requires marginally more computation in the traditional design.

In Chapter IV some simple filtering techniques have been proposed in order to further improve the decoded image quality. Using a simple mapping of frequency from one to two dimensions, a two-dimensional filter has been designed which produces the same improvement as or better improvement than other standard filters requiring more computation. Also, prefiltering the codebook instead of the image has been suggested, and is shown to be not only computationally easy but also subjectively superior. However, to implement the above suggestions requires some extra computation, and may require some additional memory storage at the decoder.

In Chapter V, two neural schemes have been described to implement the binary tree-structured VQ encoder and the full-search VQ encoder. The first scheme requires synapses of linear order in the dimensionality, and of linear-times-logarithmic order in the codebook size. The flexibility of this design has been shown, compared with the conventional implementation. The encoding time is very small, and is mostly independent of the encoder parameters such as the codebook size, the vector dimensionality, or the depth of the tree. Also the design is quite flexible. For example, only the first-layer neurons have to be altered if the dimensionality is changed. The second scheme implements the full-search VQ encoder using synapses of linear order in both the dimensionality and the codebook size. This encoder, too, is shown to carry similar advantages over conventional coders. Altogether, a neural implementation is shown to be quite appropriate for image VQ encoding. It offers the advantage of using analog inputs while producing digital outputs. The storage being inherent within the neurons, substantial space is saved.

All of the suggested techniques can be thought of as parts of a more refined, better performing image compression system. These techniques can be implemented together, or along with other improved VQ techniques offering better coding gain. Throughout this discussion, the bit rate has been kept low while the computation has been kept within the implementation capability. Therefore, the final conclusion is that vector quantization is an appropriate and plausible technique for low rate image compression. Applying electronic neural networks in VQ coding looks promising, and is not outside the grasp of present-day technology.

APPENDIX

Appendix I: Centroid for the IDWSE distortion function

For a set of vectors with the input-dependent weighted squared-error distortion function when the weight matrix is diagonal, the centroid of the set is given by the ratio of the weighted mean to the mean weight of the set.

Proof: From equation (3.3), the IDWSE distortion function in each dimension i can be written as

$$d_i(X,C) = w_{ii}(X)(x_i - c_i)^2, \tag{a.1}$$

where w_{ii} is the i-th diagonal element of the weight matrix, W_X.

Because the right-hand side of the expression $(a.1)$ has no other component of C than c_i, it is possible to minimize the average distortion $d(X,C) = \sum d_i(X,C)$ by minimizing each dimensional component independently.

For a set of M vectors $X^{(1)}, X^{(2)}, \cdots, X^{(M)}$, the average distortion in the i-th dimension for the centroid C is

$$\mathrm{E}\{d_i\} = \frac{1}{M}\sum_{j=1}^{M} w_{ii}(X^{(j)})(x_i^{(j)} - c_i)^2. \tag{a.2}$$

This distortion will be minimized when the following is true:

$$\frac{1}{M}\sum_{j=1}^{M} 2w_{ii}(X^{(j)})(c_i - x_i^{(j)}) = 0. \tag{a.3}$$

Thus, c_i becomes

$$c_i = \frac{\sum_{j=1}^{M} w_{ii}(X^{(j)})x_i^{(j)}}{\sum_{j=1}^{M} w_{ii}(X^{(j)})}. \tag{a.4}$$

Therefore, the centroid for the set can be expressed as

$$C = \frac{\mathrm{E}\{W_X \cdot X\}}{\mathrm{E}\{W_X \cdot [1]\}}. \quad \blacksquare \tag{a.5}$$

Appendix II: FIR Encoder Filter Design

Given the decoder filter $\boldsymbol{H}_d(z)$, observe that

$$|\boldsymbol{H}_d(e^{j\omega})|^2 = 1 - \epsilon \simeq 1. \tag{a.6}$$

Or,

$$\frac{1}{|\boldsymbol{H}_d(e^{j\omega})|^2} \simeq 1 + \epsilon = 2 - |\boldsymbol{H}_d(e^{j\omega})|^2. \tag{a.7}$$

Therefore, relation (4.8) can be approximated as

$$|\boldsymbol{H}_e(e^{j\omega})|^2 \simeq 2 - |\boldsymbol{H}_d(e^{j\omega})|^2. \tag{a.8}$$

Or,

$$\boldsymbol{H}_e(z)\boldsymbol{H}_e(z^{-1}) + \boldsymbol{H}_d(z)\boldsymbol{H}_d(z^{-1}) = 2. \tag{a.9}$$

Using the power-complimentary relation of (a.9), we can design the encoder filter if the decoder filter is given. The algorithm is described below.

1) Take the minimum-phase decoder filter, having sets of zeros of the form $\{re^{j\theta}, re^{-j\theta}\}$, where $r < 1$. Replace $\boldsymbol{H}_d(z)$ in (a.9) by the minimum-phase transfer function. The zeros of $\boldsymbol{H}_d(z^{-1})$ will be the inverse of the zeros of the decoder filter, i.e., of the form $\{\frac{1}{r}e^{j\theta}, \frac{1}{r}e^{-j\theta}\}$. If the decoder filter is linear-phase, simply replace $\boldsymbol{H}_d(z)\boldsymbol{H}_d(z^{-1})$ in (a.9) by the linear-phase transfer function.

2) Following the power-complimentary relation, find $\boldsymbol{H}_e(z)\boldsymbol{H}_e(z^{-1})$. Using any standard algorithm, compute the zeros of this sequence. These zeros should be in sets of the form $\{se^{j\phi}, \frac{1}{s}e^{j\phi}, se^{-j\phi}, \frac{1}{s}e^{-j\phi}\}$.

3) Choose only the zeros inside (and, on) the unit circle to form the minimum-phase encoder transfer function. Only sets of the form $\{se^{j\phi}, se^{-j\phi}\}$ with $s < 1$ will be chosen. To construct the linear-phase encoder transfer function, simply use $\boldsymbol{H}_e(z)\boldsymbol{H}_e(z^{-1})$ from (a.9); computing the zeros is not necessary.

REFERENCES

[1] M. P. Beddoes, T. Chu, "A Simple Nonstatistical Television Compression System," *IEEE Trans. Info. Th.*, vol. IT-19 (1973), pp. 648–652.

[2] R. F. Rice, "Practical Universal Noiseless Coding," *1979 SPIE Sympo. Proc.*, vol. 207 (1979), pp. 247–267.

[3] H. Gharavi, "Conditional Variable-Length Coding for Gray-Level Pictures," *AT&T Bell Lab. Tech. Jrnl.*, vol. 63 (1984), pp. 249–260.

[4] I. H. Witten, R. M. Neal, J. G. Cleary, "Arithmetic Coding for Data Compression," *Comm. of the ACM*, vol. 30 (1987), pp. 520–540.

[5] K. Sayood, M. C. Rost, "A Robust Compression System for Low Bit Rate Telemetry – Test Results with Lunar Data," *Proc. of the Scientific Data Compression Workshop*, NASA Conference Publication 3025 (1988), pp. 237–250.

[6] T. R. Lie, N. Scheinberg, D. L. Schilling, "Adaptive Delta Modulation Systems for Video Encoding," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1302–1314.

[7] D. L. Schilling, N. Scheiberg, J. Garodnick, "Video Encoding using Adaptive Delta Modulation," *IEEE Trans. Comm.*, vol. COM-26 (1978), pp. 1682–1689.

[8] J. Barba, N. Scheinberg, D. L. Schilling, J. Garodnick, S. Davidovici, "A Modified Adaptive Delta Modulator," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1767–1785.

[9] J. P. Agrawal, J. B. O'Neal, "Low Bit Rate Differential PCM for Monochrome Television Signals," *IEEE Trans. Comm.*, vol. COM-21 (1973), pp. 706–714.

[10] J. O. Limb, "Picture Coding : The Use of a Viewer Model in Source Encoding," *Bell Sys. Tech. Jrnl.*, vol. 52 (1973), pp. 1271–1302.

[11] A. K. Jain, "Image Coding Via a Nearest Neighbors Image Model," *IEEE Trans. Comm.*, vol. COM-23 (1975), pp. 318–331.

[12] E. G. Bowen, J. O. Limb, "Subjective Effect of Substituting Lines in a Video -Telephone Signal," *IEEE Trans. Comm.*, vol. COM-24 (1976), pp. 1208–1212.

[13] A. N. Netravali, E. G. Bowen, "Improved Reconstruction of DPCM-Coded Pictures," *Bell Sys. Tech. Jrnl.*, vol. 61 (1982), pp. 969–979.

[14] W. Zschunke, "DPCM Picture Coding with Adaptive Prediction," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1295–1302.

[15] K. Sawada, H. Kotera, "A 32 Mbit/s Component Separation DPCM Coding System for NTSC Color TV," *IEEE Trans. Comm.*, vol. COM-26 (1978), pp. 458–465.

[16] K. Sawada, H. Kotera, "32 Mbit/s Transmission of NTSC Color TV Signals by Composite DPCM Coding," *IEEE Trans. Comm.*, vol. COM-26 (1978), pp. 1432–1439.

[17] N. F. Maxemchuk, J. A. Stuller, "An Adaptive Intraframe DPCM Codec Based upon Nonstationary Image Model," *Bell Sys. Tech. Jrnl.*, vol. 58 (1979), pp. 1395–1412.

[18] N. F. Maxemchuk, J. A. Stuller, "Reduction of Transmission Error Propagation in Adaptively Predicted, DPCM Encoded Pictures," *Bell Sys. Tech. Jrnl.*, vol. 58 (1979), pp. 1413–1423.

[19] V. Devarajan, K. R. Rao, "DPCM Coders with Adaptive Prediction for NTSC Composite TV Signals," *IEEE Trans. Comm.*, vol. COM-28 (1980), pp.

1079–1084.

[20] D. G. Daut, R. W. Fries, J. W. Modestino, "Two-Dimensional DPCM Image Coding Based on an Assumed Stochastic Image Model," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1365–1374.

[21] B. G. Haskell, R. L. Schmidt, "A Low-Bit-Rate Interframe Coder for Videotelephone," *Bell Sys. Tech. Jrnl.*, vol. 54 (1975), pp. 1475–1495.

[22] B. G. Haskell, P. L. Gordon, R. L. Schmidt, J. V. Scattaglia, "Interframe Coding of 525-Line, Monochrome Television at 1.5 Mbits/s," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1339–1348.

[23] I. J. Dukhovich, J. B. O'Neal, "A Three-Dimensional Spatial Non-Linear Predictor for Television," *IEEE Trans. Comm.*, vol. COM-26 (1978), pp. 578–583.

[24] P. Pirsch, "Adaptive Intra-Interframe DPCM Coder," *Bell Sys. Tech. Jrnl.*, vol. 61 (1982), pp. 747–764.

[25] L. H. Zetterberg, S. Ericsson, H. Brusewitz, "Interframe DPCM with Adaptive Quantization and Entropy Coding," *IEEE Trans. Comm.*, vol. COM-30 (1982), pp. 1888–1899.

[26] N. Mukawa, H. Kuroda, T. Matsuoka, "An Interframe Coding System for Video Teleconferencing Signal Transmission at a 1.5 Mbit/s Rate," *IEEE Trans. Comm.*, vol. COM-32 (1984), pp. 280–287.

[27] H. H. Bauch, H. Haberle, H. G. Musmann, H. Ohnsorge, G. A. Wengenroth, H. J. Woite, "Picture Coding," *IEEE Trans. Comm.*, vol. COM-22 (1974), pp. 1158–1167.

[28] F. Kretz, "Subjectively Optimal Quantization of Pictures," *IEEE Trans. Comm.*, vol. COM-23 (1975), pp. 1288–1292.

[29] B. Prasada, F. W. Mounts, A. N. Netravali, "Level Reassignment : A Technique for Bit Rate Reduction," *Bell Sys. Tech. Jrnl.*, vol. 57 (1978), pp. 61–73.

[30] B. Prasada, A. Netravali, A. Kobran, "Adaptive Companding of Picture Signals in a Predictive Coder," *IEEE Trans. Comm.*, vol. COM-26 (1978), pp. 161–164.

[31] D. E. Troxel, "Application of Pseudorandom Noise to DPCM," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1763–1767.

[32] A. N. Netravali, B. Prasada, "Adaptive Quantization of Picture Signals using Spatial Masking," *Proc. IEEE*, vol. 65 (1977), pp. 536–548.

[33] A. N. Netravali, C. B. Rubinstein, "Quantization of Color Signals," *Proc. IEEE*, vol. 65 (1977), pp. 1177–1187.

[34] D. K. Sharma, A. N. Netravali, "Design of Quantizers for DPCM Coding of Picture Signals," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1267–1274.

[35] C. B. Rubinstein, J. O. Limb, "On the Design of Quantizers for DPCM Coders: Influence of the Subjective Testing Methodology," *IEEE Trans. Comm.*, vol. COM-26 (1978), pp. 565–572.

[36] J. O. Limb, C. B. Rubinstein, "On the Design of Quantizers for DPCM Coders: A Functional Relationship between Visibility, Probability and Masking," *IEEE Trans. Comm.*, vol. COM-26 (1978), pp. 573–578.

[37] P. Pirsch, "Design of DPCM Quantizers for Video Signals using Subjective Tests," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 990–1000.

[38] B. G. Haskell, F. W. Mounts, J. C. Candy, "Interframe Coding of Videotelephone Pictures," *Proc. IEEE*, vol. 60 (1972), pp. 792–800.

[39] J. O. Limb, "A Picture-Coding Algorithm for the Merli Scan," *IEEE Trans. Comm.*, vol. COM-21 (1973), pp. 300–305.

[40] A. N. Netravali, "Interpolative Picture Coding using a Subjective Criterion," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 503–508.

[41] K. Takikawa, "Simplified 6.3 Mbit/s Codec for Video Conferencing," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1877–1882.

[42] F. W. Mounts, "A Video Encoding System with Conditional Picture-Element Replenishment," *Bell Sys. Tech. Jrnl.*, vol. 48 (1969), pp. 2545–2554.

[43] D. J. Connor, B. G. Haskell, F. W. Mounts, "A Frame-to-Frame Picturephone Coder for Signals Containing Differential Quantizing Noise," *Bell Sys. Tech. Jrnl.*, vol. 52 (1973), pp. 35–51.

[44] J. O. Limb, R. F. W. Pease, K. A. Walsh, "Combining Interframe and Frame-to-Frame Coding for Television," *Bell Sys. Tech. Jrnl.*, vol. 53 (1974), pp. 1137–1173.

[45] K. Iinuma, Y. Iijimi, T. Ishiguro, H. Kaneko, S. Shigaki, "Interframe Coding for 4-MHz Color Television Signals," *IEEE Trans. Comm.*, vol. COM-23 (1975), pp. 1461–1465.

[46] B. G. Haskell, "Differential Addressing of Clusters of Changed Picture Elements for Interframe Coding of Videotelephone Signals," *IEEE Trans. Comm.*, vol. COM-24 (1976), pp. 140–144.

[47] H. Yasuda, F. Kanaya, H. Kawanishi, "1.544-Mbits/s Transmission of TV Signals by Interframe Coding System," *IEEE Trans. Comm.*, vol. COM-24 (1976), pp. 1175–1180.

[48] H. Yasuda, H. Kuroda, H. Kawanishi, F. Kanaya, H. Hashimoto, "Transmitting 4-MHz TV Signals by Combinational Difference Coding," *IEEE Trans.*

Comm., vol. COM-25 (1977), pp. 508–516.

[49] J. O. Limb, J. A. Murphy, "Measuring the Speed of Moving Objects from Television Signals," *IEEE Trans. Comm.*, vol. COM-23 (1975), pp. 474–478.

[50] F. Giorda, A. Racciu, "Bandwidth Reduction of Video Signals via Shift Vector Transmission," *IEEE Trans. Comm.*, vol. COM-23 (1975), pp. 1002–1004.

[51] S. Brofferio, F. Rocca, "Interframe Redundancy Reduction of Video Signals Generated by Translating Objects," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 448–455.

[52] A. N. Netravali, J. D. Robbins, "Motion-Compensated Television Coding : Part I," *Bell Sys. Tech. Jrnl.*, vol. 58 (1979), pp. 631–670.

[53] J. A. Stuller, A. N. Netravali, J. D. Robbins, "Interframe Television Coding using Gain and Displacement Compensation," *Bell Sys. Tech. Jrnl.*, vol. 59 (1980), pp. 1227–1240.

[54] J. R. Jain, A. K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1799–1808.

[55] K. A. Prabhu, A. N. Netravali, "Motion Compensated Component Color Coding," *IEEE Trans. Comm.*, vol. COM-30 (1982), pp. 2519–2527.

[56] K. A. Prabhu, A. N. Netravali, "Motion Compensated Composite Color Coding," *IEEE Trans. Comm.*, vol. COM-31 (1983), pp. 216–223.

[57] Y. Ninomiya, Y. Ohtsuka, "A Motion-Compensated Interframe Coding Scheme for NTSC Color Television Signals," *IEEE Trans. Comm.*, vol. COM-32 (1984), pp. 328–334.

[58] S. Sabri, "Movement Compensated Interframe Prediction for NTSC Color TV Signals," *IEEE Trans. Comm.*, vol. COM-32 (1984), pp. 954–968.

[59] D. R. Walker, K. R. Rao, "Improved Pel-Recursive Motion Compensation," *IEEE Trans. Comm.*, vol. COM-32 (1984), pp. 1128–1134.

[60] C. D. Bowling, R. A. Jones, "Motion Compensated Image Coding with a Combined Maximum A Posteriori and Regression Algorithm," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 844–857.

[61] R. Srinivasan, K. R. Rao, "Predictive Coding Based on Efficient Motion Estimation," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 888–896.

[62] S. Kappagantula, K. R. Rao, "Motion Compensated Interframe Image Prediction," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 1011–1015.

[63] T. Fukinuki, M. Miyata, "Intraframe Image Coding by Cascaded Hadamard Transforms," *IEEE Trans. Comm.*, vol. COM-21 (1973), pp. 175–180.

[64] P. J. Ready, P. A. Wintz, "Information Extraction, SNR Improvement, and Data Compression in Multispectral Imagery," *IEEE Trans. Comm.*, vol. COM-21 (1973), pp. 1123–1131.

[65] J. I. Gimlett, "Use of 'Activity' Classes in Adaptive Transform Image Coding," *IEEE Trans. Comm.*, vol. COM-23 (1975), pp. 785–786.

[66] W. Chen, C. H. Smith, "Adaptive Coding of Monochrome and Color Images," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1285–1292.

[67] J. J. Knab, "Effects of Round-Off Noise on Hadamard Transformed Imagery," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1292–1294.

[68] T. Ohira, M. Hayakawa, K. Matsumoto, "Orthogonal Transform Coding System for NTSC Color Television Signals," *IEEE Trans. Comm.*, vol. COM-26 (1978), pp. 1454–1463.

[69] A. Z. Meiri, E. Yudilevich, "A Pinned Sine Transform Image Coder," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1728–1735.

[70] H. Murakami, Y. Hatori, H. Yamamoto, "Comparison between DPCM and Hadamard Transform Coding in the Composite Coding of the NTSC Color TV Signal," *IEEE Trans. Comm.*, vol. COM-30 (1982), pp. 469–479.

[71] R. C. Reininger, J. D. Gibson, "Soft Decision Demodulation and Transform Coding of Images," *IEEE Trans. Comm.*, vol. COM-31 (1983), pp. 572–577.

[72] L. T. Watson, R. M. Haralick, O. A. Zuniga, "Constrained Transform Coding and Surface Fitting," *IEEE Trans. Comm.*, vol. COM-31 (1983), pp. 717–726.

[73] K. Takikawa, "Fast Progressive Reconstruction of a Transformed Image," *IEEE Trans. Info. Th.*, vol. IT-30 (1984), pp. 111–117.

[74] W. Chen, W. K. Pratt, "Scene Adaptive Coder," *IEEE Trans. Comm.*, vol. COM-32 (1984), pp. 225–232.

[75] M. Miyahara, K. Kotani, "Block Distribution in Orthogonal Transform Coding – Analysis, Minimization and Distortion Measure," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 90–96.

[76] J. W. Modestino, N. Farvardin, M. A. Ogrine, "Performance of Block Cosine Image Coding with Adaptive Quantization," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 210–217.

[77] N. B. Nill, "A Visual Model Weighted Cosine Transform for Image Compression and Quality Assessment," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 551–557.

[78] E. Dubois, J. L. Moncet, "Encoding and Progressive Transmission of Still Pictures in NTSC Composite Format using Transform Domain Methods," *IEEE Trans. Comm.*, vol. COM-34 (1986), pp. 310–319.

[79] S. C. Knaner, "Real-Time Video Compression Algorithm for Hadamard Transform Processing," *Proc. SPIE*, August 1975, pp. 58–69.

[80] T. R. Natarajan, N. Ahmed, "On Interframe Transform Coding," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1323–1329.

[81] J. A. Roese, W. K. Pratt, G. S. Robinson, "Interframe Cosine Transform Image Coding," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1329–1339.

[82] A. Gersho, B. Ramamurthi, "Image Coding Using Vector Quantization," *Proc. Intl. Conf. ASSP*, 1982, pp. 428–431.

[83] H. Hang, J. W. Woods, "Predictive Vector Quantization of Images," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 1208–1219.

[84] N. M. Nasrabadi, R. A. King, "A New Image Coding Technique using Transforms Vector Quantization," *Proc. Intl. Conf. ASSP*, 1984, pp. 29.9.1–29.9.4.

[85] T. Saito, H. Takeo, K. Aizawa, H. Harashima, H. Miyakawa, "Adaptive Discrete Cosine Transform Image Coding using Gain/Shape Vector Quantizers," *Proc. Intl. Conf. ASSP*, 1986, pp. 129–132.

[86] K. Aizawa, H. Harashima, H. Miyakawa, "Adaptive Discrete Cosine Transform Coding with Vector Quantization for Color Images," *Proc. Intl. Conf. ASSP*, 1986, pp. 985–988.

[87] M. E. Blain, T. R. Fischer, "Vector Quantizer Transform Coding of Imagery," *TCSP Research Report No. 87-021*, Telecomm., Control and Signal Processing Research Center, Texas A&M University, December 1987.

[88] E. J. Delp, O. R. Mitchell, "Image Compression using Block Truncation Coding," *IEEE Trans. Comm.*, vol. COM-27 (1979), pp. 1335–1342.

[89] D. J. Healy, O. R. Mitchell, "Digital Video Bandwidth Compression using

Block Truncation Coding," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1809–1817.

[90] G. R. Arce, N. C. Gallagher, "BTC Image Coding using Median Filter Roots," *IEEE Trans. Comm.*, vol. COM-31 (1983), pp. 784–793.

[91] M. D. Lema, O. R. Mitchell, "Absolute Moment Block Truncation Coding and its Application to Color Images," *IEEE Trans. Comm.*, vol. COM-32 (1984), pp. 1148–1157.

[92] H. C. Andrews, C. L. Patterson, "Singular Value Decomposition (SVD) Image Coding," *IEEE Trans. Comm.*, vol. COM-24 (1976), pp. 425–432.

[93] N. Garguir, "Comparative Performance of SVD and Adaptive Cosine Transform in Coding Images," *IEEE Trans. Comm.*, vol. COM-27 (1979), pp. 1230–1234.

[94] A. Habibi, "Hybrid Coding of Pictorial Data," *IEEE Trans. Comm.*, vol. COM-22 (1974), pp. 614–624.

[95] F. W. Mounts, A. N. Netravali, B. Prasada, "Design of Quantizers for Real-Time Hadamard Transform Coding of Pictures," *Bell Sys. Tech. Jrnl.*, vol. 56 (1977), pp. 21–48.

[96] A. N. Netravali, B. Prasada, F. W. Mounts, "Some Experiments in Adaptive and Predictive Hadamard Transform Coding of Pictures," *Bell Sys. Tech. Jrnl.*, vol. 56 (1977), pp. 1531–1547.

[97] A. Habibi, "An Adaptive Strategy for Hybrid Image Coding," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1736–1740.

[98] F. A. Kamangar, K. R. Rao, "Interfield Hybrid Coding of Component Color Television Signals," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1740–1753.

[99] A. Ploysongsang, K. R. Rao, "DCT/DPCM Processing of NTSC Composite Video Signal," *IEEE Trans. Comm.*, vol. COM-30 (1982), pp. 541–549.

[100] T. O. Tam, J. A. Stuller, "Line-Adaptive Hybrid Coding of Images," *IEEE Trans. Comm.*, vol. COM-31 (1983), pp. 445–450.

[101] S. Ericsson, "Fixed and Adaptive Predictors for Hybrid Predictive/Transform Coding," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 1291–1302.

[102] B. G. Haskell, "Frame-to-Frame Coding of Television Pictures using Two-Dimensional Fourier Transform," *IEEE Trans. Info. Th.*, vol. IT-20 (1974), pp. 119–120.

[103] R. Wilson, H. E. Knutsson, G. H. Granlund, "Anisotropic Nonstationary Image Estimation and its Applications: Part II–Predictive Image Coding," *IEEE Trans. Comm.*, vol. COM-31 (1983), pp. 398–406.

[104] M. J. Bage, "Interframe Predictive Coding of Images using Hybrid Vector Quantization," *IEEE Trans. Comm.*, vol. COM-34 (1986), pp. 411–415.

[105] M. Goldberg, H. Sun, "Image Sequence Coding using Vector Quantization," *IEEE Trans. Comm.*, vol. COM-34 (1986), pp. 703–710.

[106] A. J. Stuller, A. N. Netravali, "Transform Domain Motion Estimation," *Bell Sys. Tech. Jrnl.*, vol. 58 (1979), pp. 1673–1702.

[107] A. N. Netravali, J. A. Stuller, "Motion-Compensated Transform Coding," *Bell Sys. Tech. Jrnl.*, vol. 58 (1979), pp. 1703–1718.

[108] A. N. Netravali, J. D. Robbins, "Motion-Compensated Coding: Some New Results," *Bell Sys. Tech. Jrnl.*, vol. 59 (1980), pp. 1735–1745.

[109] R. R. Furner, R. W. Christiansen, D. M. Chabries, "Motion Compensated Vector Quantization," *Proc. Intl. Conf. ASSP*, 1986, pp. 989–992.

[110] J. W. Modestino, V. Bhaskaran, J. B. Anderson, "Tree Encoding of Images in the Presence of Channel Errors," *IEEE Trans. Info. Th.*, vol. IT-27 (1981), pp. 677–697.

[111] J. W. Modestino, V. Bhaskaran, "Robust Two-Dimensional Tree Encoding of Images," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1786–1798.

[112] J. W. Modestino, V. Bhaskaran, "Adaptive Two-Dimensional Tree Encoding of Images using Spatial Masking," *IEEE Trans. Comm.*, vol. COM-32 (1984), pp. 177–189.

[113] J. W. Modestino, D. G. Daut, "Combined Source-Channel Coding of Images," *IEEE Trans. Comm.*, vol. COM-27 (1979), pp. 1644–1659.

[114] J. W. Modestino, D. G. Daut, A. L. Vickers, "Combined Source-Channel Coding of Images using the Block Cosine Transform," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 1261–1274.

[115] D. G. Daut, J. W. Modestino, "Two-Dimensional DPCM Image Transmission over Fading Channels," *IEEE Trans. Comm.*, vol. COM-31 (1983), pp. 315–328.

[116] J. K. Yan, D. J. Sakrison, "Encoding of Images based on a Two-Component Source Model," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1315–1322.

[117] D. E. Troxel, W. F. Schreiber, P. Curlander, A. Gilkes, R. Grass, G. Hoover, "Image Enhancement/Coding Systems using Pseudorandom Noise Processing," *Proc. IEEE*, vol. 67 (1979), pp. 972–973.

[118] P. H. Westerink, J. Biemond, D. E. Boekee, "Sub-Band Coding of Images using Predictive Vector Quantization," *Proc. Intl. Conf. ASSP*, 1987, pp. 1378–1381.

[119] P. H. Westerink, D. E. Boekee, J. Biemond, J. W. Woods, "Subband Coding

of Images using Vector Quantization," *IEEE Trans. Comm.*, vol. COM-36 (1988), pp. 713–719.

[120] M. F. Barnsley, A. D. Sloan, "A Better Way to Compress Images," *Byte*, January 1988, pp. 215–223.

[121] D. J. Connor, R. C. Brainard, J. O. Limb, "Intraframe Coding for Picture Transmission," *Proc. IEEE*, vol. 60 (1972), pp. 779–791.

[122] P. A. Wintz, "Transform Picture Coding," *Proc. IEEE*, vol. 60 (1972), pp. 809–820.

[123] A. Habibi, "Survey of Adaptive Image Coding Techniques," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1275–1284.

[124] J. O. Limb, C. B. Rubinstein, J. E. Thompson, "Digital Coding of Color Video Signals–A Review," *IEEE Trans. Comm.*, vol. COM-25 (1977), pp. 1349–1385.

[125] A. N. Netravali, J. O. Limb, "Picture Coding: A Review," *Proc. IEEE*, vol. 68 (1980), pp. 366–406.

[126] B. G. Haskell, R. Steele, "Audio and Video Bit-Rate Reduction," *Proc. IEEE*, vol. 69 (1981), pp. 252–262.

[127] A. K. Jain, "Image Data Compression: A Review," *Proc. IEEE*, vol. 69 (1981), pp. 349–389.

[128] T. W. Goeddel, S. C. Bass, "A Two-Dimensional Quantizer for Coding of Digital Imagery," *IEEE Trans. Comm.*, vol. COM-29 (1981), pp. 60–67.

[129] R. M. Gray, "Vector Quantization," *IEEE ASSP Mag.*, vol. 1 (1984), no. 2 (April), pp. 4–29.

[130] N. M. Nasrabadi, "Use of Vector Quantizers in Image Coding," *Proc. Intl.*

Conf. ASSP, 1985, pp. 125–128.

[131] S. E. Budge, R. L. Baker, "Compression of Color Digital Images using Vector Quantization in Product Codes," *Proc. Intl. Conf. ASSP*, 1985, pp. 129–132.

[132] C. Yeh, "Color Image-Sequence Compression using Adaptive Binary-Tree Vector Quantization with Codebook Replenishment," *Proc. Intl. Conf. ASSP*, 1987, pp. 1059–1062.

[133] H. Shen, R. L. Baker, "A Finite State/Frame Difference Interpolative Vector Quantizer for Low Rate Image Sequence Coding," *Proc. Intl. Conf. ASSP*, 1988, pp. 1188–1191.

[134] R. Aravind, A. Gersho, "Low-Rate Image Coding with Finite-State Vector Quantization," *Proc. Intl. Conf. ASSP*, 1986, pp. 137–140.

[135] R. L. Baker, H. Shen, "A Finite-State Vector Quantizer for Low-Rate Image Sequence Coding," *Proc. Intl. Conf. ASSP*, 1987, pp. 760–763.

[136] V. Ramamoorthy, N. S. Jayant, "High Quality Image Coding with a Model-Testing Vector Quantizer and a Human Visual System Model," *Proc. Intl. Conf. ASSP*, 1988, pp. 1164–1167.

[137] K. S. Thyagarajan, S. Parthasarathy, H. Abut, "A Matrix Quantizer Incorporating the Human Visual Model," *Proc. Intl. Conf. ASSP*, 1985, pp. 141–144.

[138] S. E. Budge, T. G. Stockham, D. M. Chabries, R. W. Christiansen, "Vector Quantization of Color Digital Images within a Human Visual Model," *Proc. Intl. Conf. ASSP*, 1988, pp. 816–819.

[139] E. Daly, T. R. Hsing, "Variable Bit Rate Vector Quantization of Video Images for Packet-Switched Networks," *Proc. Intl. Conf. ASSP*, 1988, pp. 1160–1163.

[140] Y. Ho, A. Gersho, "Variable-Rate Multi-Stage Vector Quantization for Image Coding," *Proc. Intl. Conf. ASSP*, 1988, pp. 1156–1159.

[141] B. Ramamurthi, A. Gersho, "Image Vector Quantization with a Perceptually-based Cell Classifier," *Proc. Intl. Conf. ASSP*, 1984, pp. 32.10.1–32.10.4.

[142] B. Ramamurthi, A. Gersho, "Classified Vector Quantization of Images," *IEEE Trans. Comm.*, vol. COM-34 (1986), pp. 1105–1115.

[143] B. Hammer, A. v. Brandt, M. Schielein, "Hierarchical Encoding of Image Sequences using Multistage Vector Quantization," *Proc. Intl. Conf. ASSP*, 1987, pp. 1055–1058.

[144] H. Yamaguchi, "Efficient Encoding of Colored Pictures in R, G, B Components," *IEEE Trans. Comm.*, vol. COM-32 (1984), pp. 1201–1209.

[145] H. Yamaguchi, "Vector Quantization of the Differential Luminance and Chrominance Signals," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 457–464.

[146] T. Murakami, K. Asai, A. Itoh, "Vector Quantization of Color Images," *Proc. Intl. Conf. ASSP*, 1986, pp. 133–136.

[147] P. Boucher, M. Goldberg, "Color Image Compression by Adaptive Vector Quantization," *Proc. Intl. Conf. ASSP*, 1984, pp. 29.6.1–29.6.4.

[148] J. Barrilleaux, R. Hinkle, S. Wells, "Efficient Vector Quantization for Color Image Encoding," *Proc. Intl. Conf. ASSP*, 1987, pp. 740–743.

[149] H. F. Sun, M. Goldberg, "Adaptive Vector Quantization for Image Sequence Coding," *Proc. Intl. Conf. ASSP*, 1985, pp. 339–342.

[150] R. L. Baker, J. L. Salinas, "A Motion Compensated Vector Quantizer with Filtered Prediction," *Proc. Intl. Conf. ASSP*, 1988, pp. 1324–1327.

[151] A. Gersho, M. Yano, "Adaptive Vector Quantization by Progressive Codevector Replacement," *Proc. Intl. Conf. ASSP*, 1985, pp. 133–136.

[152] F. Oliveri, G. Conte, M. Gugleilmo, "A Technique using a One-Dimensional Mapping for Vector Quantisation of Images," *Proc. Intl. Conf. ASSP*, 1986, pp. 149–152.

[153] W. Equitz, "Fast Algorithms for Vector Quantization Picture Coding," *Proc. Intl. Conf. ASSP*, 1987, pp. 725–728.

[154] J. Vaisey, A. Gersho, "Simulated Annealing and Codebook Design," *Proc. Intl. Conf. ASSP*, 1988, pp. 1176–1179.

[155] C. Bei, R. M. Gray, "An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization," *IEEE Trans. Comm.*, vol. COM-33 (1985), pp. 1132–1133.

[156] D. Cheng, A. Gersho, "A Fast Codebook Search Algorithm for Nearest-Neighbor Pattern Matching," *Proc. Intl. Conf. ASSP*, 1986, pp. 265–268.

[157] A. Buzo, A. H. Gray, R. M. Gray, J. D. Markel, "Speech Coding based upon Vector Quantization," *IEEE Trans. ASSP*, vol. ASSP-28 (1980), pp. 562–574.

[158] K. Sayood, S. J. Blankenau, "A Fast Quantization Algorithm for Lattice Quantizer Design," *Proc. Intl. Conf. ASSP*, 1988, pp. 1168–1171.

[159] T. R. Fischer, "A Pyramid Vector Quantizer," *IEEE Trans. Info. Th.*, vol. IT-32 (1986), pp. 568–583.

[160] R. E. Crochiere, S. A. Webber, J. L. Flanagan, "Digital Coding of Speech in Sub-bands," *Bell Sys. Tech. Jrnl.*, vol. 55 (1976), pp. 1069–1085.

[161] A. Gersho, "Asymptotically Optimal Block Quantization," *IEEE Trans. Info. Th.*, vol. IT-25 (1979), pp. 373–380.

[162] R. J. McEliece, "*The Theory of Information and Coding*: Volume 3 of *Encyclopedia of Mathematics and its Applications*," 1977, Addison-Wesley, pp. 35–37.

[163] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE Trans. Info. Th.*, vol. 28 (1982), pp. 129–136.

[164] Y. Linde, A. Buzo, R. M. Gray, "An Algorithm for Vector Quantizer Design," *IEEE Trans. Comm.*, vol. COM-28 (1980), pp. 84–95.

[165] L. R. Rabiner, B. Gold, "*Theory and Application of Digital Signal Processing*," Englewood Cliffs (NJ) : Prentice-Hall Inc., 1975.

[166] W. F. G. Mecklenbräuker, R. M. Mersereau, "McClellan Transformation for Two-Dimensional Digital Filtering: II – Implementation," *IEEE Trans. Circuits and Systems*, vol. CAS-23 (1976), pp. 414–422.

[167] P. A. Devijver, J. Kittler, "*Pattern Recognition : A Statistical Approach*," London : Prentice/Hall Intl., 1982, pp. 409.

[168] A. Krishnamurthy, S. C. Ahalt, D. Melton, P. Chen, "A New Neural Network Learning Algorithm for Vector Quantization," *Abstracts & Papers of the 6th IEEE Intl. Workshop on Microelectronics and Photonics in Comm.*, June 1989, paper no. III.3.

[169] N. M. Nasrabadi, Y. Feng, "Vector Quantization of Images based upon the Kohonen Self-Organizing Feature Maps," *IEEE Intl. Conference on Neural Networks*, 1988, pp. I-101 – I-108.

[170] R. Dianysian, R. L. Baker, "A VLSI Chip Set for Real Time Vector Quantization of Image Sequences," *Proc. of the Intl. Sympo. on Circuits and Systems*, May 1987, pp. 221–224.

[171] R. Hecht-Nielsen, "Applications of Counterpropagation Networks," *Neural Networks*, vol. 1, no. 2, 1988, pp. 131–139.

[172] T. Kohonen, "*Self-Organization and Associative Memory*," Berlin : Springer-Verlag, 1984, pp. 125–161.

[173] S. Grossberg, "*Studies of Mind and Brain*," Dordrecht (Holland) : D Reidel Publ., 1982, pp. 79–88.

[174] P. A. Chou, T. Lookabaugh, R. M. Gray, "Optimal Pruning with Applications to Tree-Structured Source Coding and Modeling," Report, Dept. of Electrical Engg., Stanford University, May 1987.

[175] R. P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Mag.*, April 1987, pp. 4–22.

[176] S. Grossberg, "Contour Enhancement, Short Term Memory, and Constancies in Reverberating Neural Networks," *Studies in Appl. Math.*, vol. LII (52), no. 3, September 1973, pp. 213–257.

[177] E. Majani, R. Erlanson, Y. Abu-Mostafa, "On the K-Winners-Take-All Network," *Advances in Neural Information Processing Systems I*, edited by D. S. Touretzky, San Mateo (CA) : Morgan Kaufmann Publ., 1989, pp. 634–642.

[178] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, C. A. Mead, "Winner-Take-All Networks of *O(N)* Complexity," *Advances in Neural Information Processing Systems I*, edited by D. S. Touretzky, San Mateo (CA) : Morgan Kaufmann Publ., 1989, pp. 703–711.