NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

Canada

PROGRESSIVE IMAGE TRANSMISSION

by
Limin Wang

A THESIS
submitted to the School of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of
DOCTOR OF PHILOSOPHY
in
Electrical Engineering

Ottawa-Carleton Institute for Electrical Engineering
Department of Electrical Engineering
Faculty of Engineering
University of Ottawa
OTTAWA, ONTARIO K1N 6N5

ISBN 0-315-46879-3

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

ABSTRACT

Progressive image transmission is receiving attention for application in interactive image communication over low-bandwidth channels. In progressive image transmission, a low-resolution image is first transmitted with as few bits as possible. Upon the reviewer's request, the image can progressively be improved with further transmission. Eventually, an exact replica of the original image should be reconstructed.

In this thesis, progressive image transmission techniques are first reviewed, following a summary of image coding. Four new progressive image transmission techniques are then presented. In the first two techniques, the residual errors due to the previous quantization are iteratively fedback and requantized either in the spatial domain or in the transform domain. In the third technique, vector quantization is applied to the transform coefficients in a multi-layer fashion so that the total number of bits allocated to each coefficient is proportional to the coefficient variance. In the fourth technique, a difference quadtree with an error-recovery property is formed and vector quantization is applied to the difference quadtree on a level-by-level basis.

The simulation results are reported in terms of rate distortion curves and in pictorial form. It is demonstrated that the first approximations provide the gross structure of the image at a very low bit rate and that the successive approximations converge rapidly to an excellent quality, both subjectively and objectively. The proposed techniques are also compared with other image coding techniques, particularly, single-stage vector quantization, transform coding and adaptive transform coding, and are seen in many cases to outperform these techniques.

ACKNOWLEDGMENT

I wish to express my gratitude and appreciation to my thesis supervisor, Dr. Morris Goldberg. No amount of words would suffice to thank him for his constant guidance, encouragement and support which he has generously given me during my research program.

I would also like to record my sincere thanks to the members of my advisory committee, Dr. Willem Steenaart, Dr. Seymour Shlien and Dr. David Falconer for their valuable guidance and encouragement.

My thanks are due to the Shandong University, China for giving me time on study leave. I would also like to thank the staff of the Department of Electrical Engineering, the University of Ottawa for their help and support.

I also wish to thank my colleagues and friends, Huifang Sun, Steve Reed and Sethuraman Panchanathan for their help and encouragement.

A special note of thanks goes to my family. It would not have been possible to finish this long-term work without their support.

Finally, I would like to acknowledge the financial support from NSERC grant.

LIST OF SYMBOLS

Chapter 1

RVQ-SPA – residual error vector quantization in spatial domain.

RQ-TRA – residual error quantization in transform domain.

MVQ-OBA – multistage vector quantization with optimal bit allocation planes.

VQ-QT – vector quantization on images in quadtree form.

Chapter 2

X – A digital image source.

X_{ij} – the (i,j)th pixel of X.

x – the specific image pattern.

x_{ij} – the (i,j)th pixel of x.

$H(X)$ – the entropy of X.

$p(x)$ – the probability of a specific image pattern x.

$H^1(X)$ – the first order entropy of X.

P_k – the probability of occurrence of the kth grey level value.

$\hat{X}$ – the coded image.

$\hat{X}_{ij}$ – the (i,j)th pixel of $\hat{X}$.

$I_{N\times N}(X,\hat{X})$ – the average mutual information.

$D = \frac{1}{N\times N}E[d(X,\hat{X})]$ – the average distortion per pixel.

$R_{N\times N}$ – the $N \times N$ – block rate distortion function

$R(D)$ – the rate distortion function.

NMSE – the normalized mean square error.

A ($A' = A^{-1}$) – the orthogonal transform.

Q – the transformed image.

Q_{ij} – the (i,j)th coefficient.

$\hat{Q}$ – the quantized transformed image.

$\hat{Q}_{ij}$ – the (i,j)th quantized coefficient.

σ_x^2 – the average variance of the image.

$\sigma_{x,ij}^2$ – the variance of pixel (i,j).

σ_r^2 – the reconstruction error variance.

σ_q^2 – the quantization error variance.

$\sigma_{r,ij}^2$ – the reconstruction error variance of pixel (i,j).

$\sigma_{q,ij}^2$ – the quantization error variance of coefficient (i,j).

ϵ_q^2 – the quantizer performance factor.

ϵ^2 – the variable correction factor.

ϵ_{ij}^2 – the correction factor for coefficient (i,j).

B – the average bit rate.

B_{ij} – the number of bits assigned to coefficient (i,j).

V – the input vector.

$\hat{V}$ – the output vector (codeword).

$\{\Pi_i\}$ – the partition of the input vector space.

D_v – the vector dimensionality.

N_c – the number of codewords.

C – the computational cost.

M – the storage cost.

R_l – the bit rate for the labels.

R_c – the bit rate for the codebook.

Chapter 3

X_k – the kth level of pyramid.

$X_{k,ij}$ – the (i,j)th node of the kth pyramid level.

G_l – the lth level of Gaussian pyramid.

L_l – the lth level of Laplacian pyramid.

C_l – the quantized version of L_l.

$\hat{G}_l$ – the reconstruction of G_l.

Chapter 4

E_k – the input image to the kth iterative stage of RVQ-SPA.

$\hat{E}_k$ - the output image to the kth iterative stage of RVQ-SPA.

$\hat{X}_k$ – the kth approximation.

$\hat{X}_{k,ij}$ – the (i,j)th pixel of the kth approximation.

σ_k^2 – the variance of the kth residual error image.

c_k – the coding factor for stage k.

NMSE(k) – the normalized mean square error of the kth approximation.

$R_p(k)$ – the total progressive bit rate up to stage k.

$R_t(k)$ – the total bit rate for lossless reproduction.

$H^t(k)$ – the total progressive information transmitted up to stage k.

$R^r(k)$ – the accumulated coding redundancy up to stage k.

$H_e(k)$ – the entropy of the kth residual error image.

$H_e^1(k)$ – the first order entropy of the kth residual error image.

ΔR_m – the bit rate required for transmission at stage m.

ΔH_m^t – the image information transmitted at stage m.

ΔR_m^r – the coding redundancy introduced at stage m.

$\Delta R_{t,m}$ – the variation of $R_t(k)$.

$\Delta H^r_{c,m}$ – the variation of $H^r_c(k)$.

Chapter 5

B_k – the average bit rate for stage k.

$B_{k,ij}$ – the number of bits assigned to coefficient (i,j) at stage k.

$\min(\sigma^2_{r,k+1})$ – the average reconstruction error variance up to stage k using RSQ-TRA.

$\min(\sigma^2_{q,k+1})$ – the average quantization error variance at stage k using RSQ-TRA.

ϵ^2_k – the correction factor for stage k.

$\epsilon^2_{m,ij}$ – the correction factor for coefficient (i,j) at stage k.

$\epsilon^2_{q,m,ij}$ – the quantizer performance factor for coefficient (i,j) at stage k.

c_m – the coding factor for stage m.

$\sigma^2_{vq,m}$ – the average quantization error variance at stage m using RVQ-TRA.

$R_{co}(k)$ – the bit rate for coding the coefficients at stage k using RSQ-TRA.

$R_o(k)$ – the bit rate for transmitting the bit allocation map at stage k.

Chapter 6

$\{B_{ij}\}$ – the bit allocation planes.

$B_{k,ij}$ – the number of bit assigned to coefficient (i,j) at bit allocation plane k.

T_k – the kth threshold.

$R_m(k)$ – the bit rate for indicating the position of the coefficient added at stage m.

Chapter 7

$X_{k,ij}$ – the (i,j)th pixel (node) of the kth level of the mean quadtree.

$D_{k,ij}$ – the (i,j)th pixel (node) of the kth level of the difference quadtree.

H^1_o – the first order entropy of the original image.

H_q^1 – the first order entropy of the difference quadtree.

H_{eq}^1 – the equivalent entropy.

$R_h(k)$ – the bit rate for Huffman coding of the difference quadtree stage level k.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1. INTRODUCTION

An *Image* refers to a two-dimensional light intensity function $F(x,y)$, where x and y denote spatial coordinates and the value of F at any point (x,y) is proportional to the brightness (or grey level) of the image at that point. The corresponding *digital image* $X(i,j)$ can be obtained by sampling the analog image $F(x,y)$ in the spatial coordinates and quantizing the brightness, that is,

$$X(i,j) = \hat{F}(x_0 + i\Delta x, y_0 + j\Delta y), \quad i,j = 0,1,...N$$

where $\hat{F}$ is the quantized value of F and Δx and Δy are the sampling intervals. A digital image can be considered as a matrix whose row and column indices identify a point in the image and the corresponding matrix element value identifies the grey level at that point. The elements of such a digital array are referred to as *image element*, *picture element*, or *pixel*.

The transmission of still images is an emerging field in image communication. There are many important applications, such as, teleconferencing, teleconsultation, still image broadcasting, remote plant surveillance, computer graphics communication, interactive visual search of large picture databases, etc.

With standard raster transmission techniques, a digital image is transmitted as a sequence of rows, or lines, of the image matrix, line by line from top to bottom. Hence, to have any notion of what the final image will be like, the viewer has to wait for a nearly complete transmission. Since the digital image data is often transmitted over relatively slow channels, it can take a long time to send a full-resolution image. For example, transmitting a image of size 256 by 256 with 8 bits/pixel over a 1200-baud line would require about 7.28 minutes.

Progressive image transmission [1-24] can alleviate this problem by first transmitting a low-resolution approximation image, instead of a full-resolution image, at a very low bit rate. Upon the viewer's request, the low-resolution image is then progressively improved with further transmission. Eventually, an exact replica of the original

image is reproduced. In other words, in progressive image transmission, successive approximation images are refinements of the earlier images and converge to the final, full resolution image.

In summary, progressive image transmission improves the viewer interaction with an image database by progressively transmitting a sequence of successive approximations with refined resolution. The primary advantage of progressive image transmission is that the gross structural information of the image appears immediately at the beginning of transmission. It is, therefore, possible for the viewer to make a decision whether further transmission is necessary, or not, in a shorter time, thus, reducing the transmission time.

The two basic requirements for progressive image transmission are, therefore, as follows:

1) The image to be transmitted is first reorganized in a form which is suited for progressive transmission;

2) The first approximations, transmitted at a very low bit rate, provide the gross structural information of the image so that the image can be recognized in a shorter time.

In addition, two other requirements are desirable:

1). Lossless reproduction;

2). A certain amount of compression.

In this thesis, four new progressive image transmission techniques are presented: 1) *Residual Error Vector Quantization In The Spatial Domain* (RVQ-SPA) [16,18,23]; 2) *Residual Error Quantization In The Transform Domain* (RQ-TRA) [17,19,21]; 3) *Multistage Vector Quantization With Optimum Bit Allocation Planes* (MVQ-OBA) [24]; and 4) *Vector Quantization on Images in Quadtree Form* (VQ-QT) [20,22]. All four techniques have a multi-stage coding structure. At each stage, a coding process

is involved, and a fraction of the image information is coded and transmitted to yield progressive image transmission. Finally, an entropy coder is used to encode the final residual error image, thus ensuring the lossless reproduction of the image.

In RVQ-SPA, the residual errors due to quantization are iteratively fedback and vector requantized in the spatial domain. An entropy coder is used to encode the final error image. At each stage, a fraction of the image information is transmitted and utilized in reconstructing successive approximations of the image. Iterative coding of the residual errors due to quantization makes it possible to transmit the image progressively. Compression is achieved by applying vector quantization at each stage. Lossless reproduction is guaranteed by an entropy coder at the final stage. The simulation results demonstrate that the proposed scheme achieves, simultaneously, lossless, progressive transmission of the image at an appreciable amount of data compression.

In RQ-TRA, the residual error quantization technique is applied to the transform domain. An image to be transmitted first undergoes an orthogonal transform and the resulting coefficients are then quantized (scalar or vector) before transmission. The residual error array due to quantization is iteratively fedback and requantized (scalar RSQ or vector RVQ); the coded residual error information is progressively transmitted and utilized in reconstructing the successive approximations. It is shown that the average reconstruction error variance converges to zero, as the number of iterative stages approaches infinity. In practice, lossless reproduction can be achieved with a small number of iterations by using an entropy coder on the final residual error image. It is shown experimentally that the total bit rate for lossless reproduction is less than the first order entropy of the original image; in other words, not only is progressive transmission achieved, but some compression as well. However, the simulation results demonstrate that there is no improvement using RVQ-TRA over RVQ-SPA. This is because the nonuniform distribution of the coefficient energies is not exploited.

In MVQ-OBA, we introduce a multistage vector quantization technique applied to the transform coefficients in such a way that the total number of bits allocated to each coefficient is proportional to the coefficient variance. An optimum bit allocation map is first found based on the variances of the transform coefficients for a fixed bit rate. The optimum bit allocation map is then sliced into a set of bit allocation planes by applying a set of the thresholds. With the set of bit allocation planes, the transformed image is vector quantized on a stage-by-stage basis where, at each stage, only the coefficients (or residual error coefficients) assigned the non-zero number of bits are taken as the components of vectors and vector quantized. In other words, the transform coefficients are first vector quantized with the first bit allocation plane; the residual error coefficients are then requantized with the second bit allocation plane. The same process is repeated until the last bit allocation plane is used. A separate codebook is generated for each plane by using the generalized Lloyd clustering algorithm. The experiments demonstrate that MVQ-OBA results in similar error variances in coefficient quantization at each stage. Since the technique operates in a multi-stages manner, it in fact results in progressive image transmission. It is shown experimentally that MVQ-OBA is outperforms RVQ-SPA.

In VQ-QT, vector quantization is applied to images represented by quadtree. A mean quadtree representation of an image is first built up by forming a sequence of reduced-size images by averaging over blocks of 2×2 pixels. A difference quadtree is then built up by taking the differences between successive levels in the mean quadtree. Progressive transmission is achieved by sending all the nodes in the difference quadtree starting from the top level and ending at the bottom level. The kth approximation image can be formed by adding the information of level k to the previously reproduced $(k-1)$th approximation. To gain efficiency, vector quantization is applied to the difference quadtree of the image on a level-by-level basis. The errors due to quantization at level k are fed forward and included in the next level, $k+1$. It is demonstrated that the original image can be losslessly reconstructed. It should be

emphasized that the error delivery procedure is the key for lossless reproduction of the original image, which makes it possible to reprocess, at lower levels, the information lost at higher levels. Finally, an entropy coder, such as Huffman coder, is used to losslessly encode the final residual error image, thus ensuring perfect reproduction of the original image. The experiments demonstrate that it is possible to achieve, simultaneously, lossless, progressive transmission, with compression. It is shown that, at the intermediate levels, the use of vector quantization results in a coding gain over using only an entropy coder. Furthermore, among all the four techniques (at a block size of 4×4), VQ-QT is seen to achieve the best performance at a lower bit rate ($<$ 0.3 bits/pixel whereas MVQ-OBA is best for bit rates of around 0.3-06 bits/pixel.

The rest of this thesis is organized as follows. Chapter 2 introduces the basic concepts of entropy and rate distortion function, and reviews image coding techniques. Since transform coding and vector quantization form the basis of the following chapters, they are analyzed and described in some detail. Chapter 3 gives a review of previous research contributions in progressive image transmission. Chapters 4, 5, 6 and 7 present, respectively, the four new progressive image transmission techniques. Chapter 8 briefly summarizes the main results, lists the current research work, and provides some suggestions for further investigation on progressive image transmission.

2. IMAGE CODING

In this chapter, the two basic concepts of information theory, *Entropy* and *Rate Distortion Function,* are first presented. Distortion measures used in image coding are then discussed. This is followed by a short review of *Lossless* and *Lossy* coding techniques, including *Huffman Coding, Run-length Coding, Predictive coding, Transform Coding* and *Hybrid Transform/Predictive Coding.* Finally, a review of *Vector Quantization* as applied to image coding is given.

2.1 ENTROPY AND RATE DISTORTION FUNCTION

Entropy and the rate distortion function are the two basic concepts in source coding. Information theory indicates [25-29] that if the source is stationary and ergodic, there exists a monotonically non-increasing rate distortion function which provides a lower bound on the average bit rate for a given distortion and hence an upper bound on the performance of practical source coders. The lower bound value for a distortion equal to zero corresponds to a quantity referred to as the source entropy. The entropy is a useful measure for source coding because it gives the minimum average bit rate for perfect reproduction, or lossless coding. The basic concepts of entropy and rate distortion function are described in the following.

2.1.1 ENTROPY

Consider a digital image source X of size $N \times N$, i.e.

$$X = \{X_{ij},\ i,j = 0,1,...N-1\}.$$

If each pixel is quantized to K grey levels, i.e. $\log_2 K$ bits/pixel, then, a total of $K^{N\times N}$ possible image patterns can be generated by the image source. Let $p(x)$ represent the probability of a specific image pattern,

$$x = \{x_{ij},\ i,j = 0,1,...N-1\},$$

where x_{ij} is the (i,j)th element of x. The average information of the source is specified

by its entropy defined as [43],

$$H(X) = -\frac{1}{N \times N} \sum_{\text{all } x} p(x) \log_2 p(x) \qquad bits/pixel. \tag{2.1.1}$$

It has been shown [47] that the source entropy $H(X)$ is lower bounded by 0 and upper bounded by $\log_2 K$, that is,

$$0 \leq H(X) \leq \log_2 K. \tag{2.1.2}$$

The left-side equality holds only if all the probabilities, $p(x)$, except one are zero, in which case the source is totally predictable. The right-side equality holds if, and only if, all the image patterns are equiprobable. The redundancy of the source has been defined as the difference between the quantity $\log_2 K$ and the source entropy [47],

$$redundancy = \log_2 K - H(X). \tag{2.1.3}$$

If the pixels of the image are statistically independent, then, the probability of a specific image pattern x can be expressed as

$$p(x) = \prod_{i,j=0}^{N-1} p_{ij}(x_{ij}). \tag{2.1.4}$$

Here, p_{ij} represents the probability that the (i,j)th element of the image source X, X_{ij}, has a value equal to x_{ij}. Substituting equation (2.1.4) into (2.1.1), we get,

$$\begin{aligned}
H(X) &= -\frac{1}{N \times N} \sum_{\text{all } x} \{ \prod_{i,j=0}^{N-1} p_{ij}(x_{ij}) \log_2 \prod_{i,j=0}^{N-1} p_{ij}(x_{ij}) \} \\
&= -\frac{1}{N \times N} \sum_{\text{all } x} \{ \prod_{i,j=0}^{N-1} p_{ij}(x_{ij}) \sum_{i,j=0}^{N-1} \log_2 p_{ij}(x_{ij}) \} \\
&= -\frac{1}{N \times N} \sum_{\text{all } x} \sum_{i,j=0}^{N-1} \prod_{u,v=0}^{N-1} p_{uv}(x_{uv}) \log_2 p_{ij}(x_{ij}) \\
&= -\frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sum_{\text{all } x} \prod_{u,v=0}^{N-1} p_{uv}(x_{uv}) \log_2 p_{ij}(x_{ij}) \\
&= -\frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sum_{\text{all } x_{ij}} p_{ij}(x_{ij}) \log_2 p_{ij}(x_{ij}).
\end{aligned} \tag{2.1.5}$$

Furthermore, if the pixels of the image are identically and independently distributed (i.i.d), that is,

$$p_{ij}(x_{ij}) = p_{uv}(x_{uv}), \qquad \text{for } x_{ij} = x_{uv}, \tag{2.1.6}$$

the entropy (2.1.5) can then be expressed as,

$$H(X) = -\sum_{k=0}^{N-1} P_k \log_2 P_k. \tag{2.1.7}$$

In this case, P_k is simply the probability of occurrence of the kth grey level value.

In practice, the statistical information of the image, $p(x)$, is, however, not easily measured or modeled and therefore, the true entropy of image is, in general, very difficult to obtain. In image processing, the true entropy is often replaced by the first order entropy, $H^1(X)$, which is measured on a pixel-by-pixel basis and defined as,

$$H^1(X) = -\sum_{k=0}^{N-1} P_k \log_2 P_k. \tag{2.1.8}$$

From equation (2.1.7) and (2.1.8), it can be seen that only when the image source is identically and independently distributed (i.i.d), does the first order entropy $H^1(X)$ equal to the true entropy $H(X)$.

2.1.2 RATE DISTORTION FUNCTION

Fig. 2.1.1 shows a general communication system. We let X and $\hat{X}$ refer, respectively, to the coder input (the source output) and the decoder output. The average information transmitted to the destination per pixel is given by the average mutual information $I_{N\times N}(X, \hat{X})$ [29],

$$I_{N\times N}(X, \hat{X}) = \frac{1}{N \times N} \sum_{\text{all } x,\hat{x}} p(x)p(\hat{x}/x) \log \frac{p(\hat{x}/x)}{\sum_{\text{all } x} p(x)p(\hat{x}/x)}. \tag{2.1.9}$$

This is a measure of the statistical dependence between the source output X and the decoder output $\hat{X}$. Since $p(x)p(\hat{x}/x) = p(x, \hat{x})$, it follows that

$$\sum_{\text{all } x} p(x)p(\hat{x}/x) = \sum_{\text{all } x} p(x, \hat{x}) = p(\hat{x}) \tag{2.1.10}$$

and therefore,

$$\begin{aligned} I_{N\times N} &= \frac{1}{N\times N}\sum_{\text{all } x,\hat{x}} p(x,\hat{x})\log_2\frac{p(\hat{x}/x)}{p(\hat{x})} \\ &= \frac{1}{N\times N}\sum_{\text{all } x,\hat{x}} p(x,\hat{x})\log_2\frac{p(x,\hat{x})}{p(x)p(\hat{x})} \\ &= \frac{1}{N\times N}\sum_{\text{all } x,\hat{x}} p(x,\hat{x})\log_2\frac{p(x/\hat{x})}{p(x)} \\ &= -\frac{1}{N\times N}\sum_{\text{all } x,\hat{x}} p(x,\hat{x})\log_2 p(x) + \frac{1}{N\times N}\sum_{\text{all } x,\hat{x}} p(x,\hat{x})\log_2 p(x/\hat{x}) \\ &= H(X) - H(X/\hat{X}). \end{aligned} \tag{2.1.11}$$

Here,

$$H(X/\hat{X}) = -\frac{1}{N\times N}\sum_{\text{all } x,\hat{x}} p(x,\hat{x})\log_2 p(x/\hat{x}) \tag{2.1.12}$$

is the entropy of the source X given the decoder output $\hat{X}$. Since the entropy is a concept of uncertainty, equation (2.1.11) can be understood as follows: the mutual information between X and $\hat{X}$ is equal to the uncertainty in the source X minus the uncertainty in source X given the decoder output $\hat{X}$. If the encoding is lossless, i.e. $\hat{x} = x$, then the uncertainty in the source output X given the decoder output $\hat{X}$ is zero. In other words, in the case $\hat{x} = x$,

$$p(x/\hat{x}) = \begin{cases} 1, & \text{if } \hat{x} = x \\ 0, & \text{if } \hat{x} \neq x \end{cases} \tag{2.1.13}$$

and therefore, the uncertainty in X given $\hat{X}$ is given by,

$$H(X/\hat{X}) = -\frac{1}{N\times N}\sum_{\text{all } x,\hat{x}} p(x,\hat{x})\log_2 p(x/\hat{x}) = 0 \tag{2.1.14}$$

Hence, the average mutual information is equal to

$$I_{N\times N} = H(X) - H(X/\hat{X}) = H(X). \tag{2.1.15}$$

This implies that lossless encoding requires at least a bit rate of $H(X)$. On the other hand, if the decoder output $\hat{X}$ contains no information about the source output X,

that is,

$$p(x/\hat{x}) = \frac{p(x,\hat{x})}{p(\hat{x})} = \frac{p(x)p(\hat{x})}{p(\hat{x})} = p(x), \tag{2.1.16}$$

the corresponding average mutual information is then equal to

$$\begin{aligned} H(X/\hat{X}) &= -\frac{1}{N \times N} \sum_{\text{all } x,\hat{x}} p(x,\hat{x}) \log_2 p(x/\hat{x}) \\ &= -\frac{1}{N \times N} \sum_{\text{all } x,\hat{x}} p(x,\hat{x}) \log_2 p(x) \\ &= H(X). \end{aligned} \tag{2.1.17}$$

This implies that no information is transmitted,

$$I_{N\times N} = H(X) - H(X/\hat{X}) = 0. \tag{2.1.18}$$

In the more general situation, since source encoding may introduce errors, the uncertainty $H(X/\hat{X})$ is a positive quantity less than the source entropy $H(X)$. The average mutual information is, therefore, lower bounded by 0 and upper bounded by the source entropy, i.e.

$$0 \leq I_{N\times N} \leq H(X). \tag{2.1.19}$$

If we now let $d(x,\hat{x})$ represent a measure of the *distortion,* or the *distance,* between x and $\hat{x}$, the average distortion per pixel is, therefore,

$$D = \frac{1}{N \times N} E[d(X,\hat{X})] = \frac{1}{N \times N} \sum_{\text{all } x,\hat{x}} d(x,\hat{x}) p(x) p(\hat{x}/x). \tag{2.1.20}$$

The $N \times N-$block rate distortion function $R_{N\times N}(D)$ [29] is defined mathematically as the minimum average mutual information between X and $\hat{X}$ subject to the constraint of a fixed average distortion D,

$$R_{N\times N}(D) = \inf_{\text{all } p(\hat{x}/x)} I_{N\times N}(X,\hat{X}). \tag{2.1.21}$$

Note that since the source is given, the minimization can only be over the conditional probability $p(\hat{x}/x)$. The rate distortion function $R(D)$ [29] is simply the limiting value

of the $N \times N-$block rate distortion function $R_{N\times N}(D)$ as the block size $N \times N \to \infty$,

$$R(D) = \lim_{N\times N\to\infty} R_{N\times N}(D). \tag{2.1.22}$$

In other words, the distortion D and the mutual information $I_{N\times N}(X,\hat{X})$ depend on the type of source coding. However, there is a minimum $I_{N\times N}(X,\hat{X})$ that is needed so that the average distortion of the reconstruction at the destination does not exceed the specified upper limit D. This minimum value of $I_{N\times N}(X,\hat{X})$ is $R(D)$.

A plot of a typical rate distortion function is shown in Fig. 2.1.2. It is seen to be monotonically non-increasing, which is reasonable in that the higher information-rate representations should lead to the smaller (strictly, non-increasing) average distortion. It is also clear that the distortion at rate $R = 0$ should be finite if the input variance is finite. For perfect reproduction ($D = 0$), the bit rate required is just equal to the source entropy or average self-information,

$$R(0) = H(X) \tag{2.1.23}$$

2.2 DISTORTION MEASURE

A distortion measure for a coding system is an assignment of a cost $d(x,\hat{x})$ for reproducing an input, x, by an output, $\hat{x}$. With such a distortion measure, the performance of a coding system can be evaluated using the average distortion $E[d(X,\hat{X})]$,

$$E[d(X,\hat{X})] = \sum_{\text{all } x,\hat{x}} d(x,\hat{x})p(x,\hat{x}). \tag{2.2.1}$$

To be of value for both design and comparison, a distortion measure should be:

1) Computable, so that it can be efficiently evaluated in real time;

2) Subjectively meaningful, so that it correlates well with human visual observations;

3) Tractable, so that it is amenable to mathematical analysis.

Perhaps, the most popular distortion measure used in image coding is the average mean square error (MSE) [43], defined as,

$$\text{MSE} = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} E[(X_{ij} - \hat{X}_{ij})^2] \qquad (2.2.2)$$

where X_{ij} and $\hat{X}_{ij}$ are, respectively, the (i,j)th element of the original image and the coded image. Experimentally, the average mean square error is often estimated by,

$$\text{MSE} = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} (X_{ij} - \hat{X}_{ij})^2. \qquad (2.2.3)$$

The mean square error is often normalized by the average energy of the original image (NMSE),

$$\text{NMSE} = \frac{\sum_{i,j=0}^{N-1} (X_{ij} - \hat{X}_{ij})^2}{\sum_{i,j=0}^{N-1} X_{ij}^2}. \qquad (2.2.4)$$

The major advantages of the normalized mean square error (NMSE) are its intuitive appeal (large errors are given more importance than small errors), its ease of computation, and its mathematical tractability. A variant form, more easily calculable, is the normalized absolute error between the original and coded images, defined as [43],

$$e = \frac{\sum_{i,j=0}^{N-1} | X_{ij} - \hat{X}_{ij} |}{\sum_{i,j=0}^{N-1} | X_{ij} |}. \qquad (2.2.5)$$

However, neither the mean square error nor the absolute error is found to correlate very well with subjective ratings obtained by presenting the coded images to the human observers. Two images having the same mean square error or absolute error may give very different subjective evaluations. The significant features of human visual system with regard to (monochrome) image coding evaluations are its logarithmic sensitivity to light intensity and its nonuniform response to spatial frequencies [43]. There has been considerable effort to yield a distortion measure that is better correlated with subjective results. One approach is to preprocess both the original and coded images prior to formation of the mean square error [43]. Examples

of preprocessing include point transformations, such as, logarithmic and power-law transformations, and spatial transformations, such as, Laplacian, gradient and convolution operations [43]. Another approach investigated [49,94] is to pass the error image between the original and coded images through an operator designed to model the processing steps for the human visual system

Although these preprocess steps do seem to improve the correlation with subjective rating [43], there is, up to now, still no accepted standard. In this thesis, the normalized mean square error (NMSE) between the original image X and the coded image $\hat{X}$, is adopted as the distortion measure because it is analytically tractable, computable in real time, and also subjectively relevant. It should be emphasized that even though the NMSE does not completely match with the subjective rating, small and large NMSE do correspond to good and bad subjective qualities, respectively [43]. In evaluating the performance of a coding system, the (bit) rate distortion (NMSE) curve [43] is used. To compare different coding schemes, the simulations in the following chapters are carried out on the same images.

2.3 LOSSLESS CODING

For most natural images, there is a strong dependence between neighbouring pixels, implying that there is a statistical redundancy in the images. Lossless coding removes, or at least reduces, the statistical redundancy of the image in such a way that redundancy reduction operation is a reversible process; i.e. the original image can be recovered exactly. Because of this property, there has been a great interest in lossless coding techniques. In this subsection, two lossless coding techniques, *Huffman Coding* and *Run-length Coding*, are reviewed.

2.3.1 HUFFMAN CODING

Shannon's first theorem, or lossless coding theorem, tells us that the average bit rate R for encoding a source can be made as small as, but no smaller than, the source

entropy H [30,31]. Huffman [32] provided a practical procedure for implementing the Shannon coding theorem for encoding a statistically independent source, which results in a minimum average bit rate. The Huffman coding procedure results in a variable length code in which the number of bits for a source sample varies approximately inversely with the corresponding source probability. This code has the property of instantaneous decodability and makes use of the concept of a coding tree. If the *efficiency* of the code is defined as [30,31],

$$efficiency = \frac{H}{R} \times 100\%, \tag{2.3.1}$$

the Huffman code reaches 100% efficiency when the source probabilities are negative powers of 2.

The procedure for constructing a Huffman code tree for a given source is illustrated in Fig. 2.3.1. First of all, the set of the source probabilities is arranged in decreasing order (0.4, 0.3, 0.1, 0.1, 0.06, 0.04). The two smallest probabilities (0.06, 0.04) are then combined to yield a new single value, resulting in a reduced set of probabilities. The reduced set of probabilities is once again re-arranged in decreasing order (0.4, 0.3, 0.1, 0.1, 0.1). This process is repeated until a set containing only one value (unity) is reached. Each node of the binary tree corresponds to the new probability formed, a binary number '0' is assigned to the upper member and '1' to the lower, as shown in Fig. 2.3.1. The codeword for a source sample is the sequence of 0's and 1's obtained by tracing the path from the leaf of the tree to the root of the tree.

The design of a Huffman code tree for a given source depends upon the source statistics. If source statistics change, then a new coding tree is required to ensure the minimum bit rate. In adaptive Huffman coding [33-36], the Huffman tree used to encode the current source sample is the Huffman tree calculated for the previous source samples. In other words, the current source sample, s_i, is coded by using a Huffman coder which is based upon frequency information derived from the most

most recent previous samples, $s_{i-1}, s_{i-2}, \ldots$. The frequency information is obtained by simply counting the previous occurrences of the sample values. If both the transmitter and receiver start with the same initial Huffman tree and use the same algorithm to modify the tree after each source sample is processed, no side information need be transmitted to the receiver.

2.3.2. RUN-LENGTH CODING

A second class of lossless coding widely used in image coding is run-length coding [37-40]. A run is defined as a sequence of consecutive pixels of identical values along a specified direction, for example, the horizontal scan line. If the runs are long, significant reduction in the average bit rate may be achieved by simply transmitting the start and the length of the run. Run length coding can be very efficient for binary images, consisting of graphics and text [42-47]. For detailed images, the runs tend to be very short and the required bit rate can even increase. However, it can be used to code the individual bit planes of an image. Fig. 2.3.2 shows decomposition of an 8-bit image into a set of bit planes where the ith significant bits are located in the ith bit plane. Spencer and Huang [37] reported an average bit rate of 2 to 4 bits/pixels for images of 256×256 with 6 bits/pixel.

Run-length coding can also be extended to two dimension, i.e. area coding. An area is defined as a connected, continuous group of pixels of identical value. To transmit an image, only the values which specify the area, such as, the set of boundary points and the intensity value of the area, are needed to be sent. In the simulation results reported in [38], the bit rate on 4-bit images is reduced to about 2.5 bits/pixel.

2.4 LOSSY CODING

In lossy coding, the objective is to reduce the transmission bit rate subject to some image quality constraint. Historically, lossy image coding techniques have generally followed two paths: predictive coding and transform coding. Both predictive and

transform coding possess relative advantages and limitations [42-49]. From an implementation point of view, predictive coding systems have much lower complexity both in terms of the memory requirements and the number of operations to be performed compared to transform coding systems. However, transform coding achieves a relatively higher image fidelity reconstruction at lower bit rates (less than 1 bits/pixel) while predictive coding results in superior coding performance at higher bit rates (around 2 bits/pixel) [43]. A second difference is that the errors due to quantization in transform coding are distributed over the entire image (during inverse transformation) whereas the errors in predictive coding are locally concentrated. Furthermore, predictive coding is more sensitive to changes in the statistics of the image than transform coding. Hybrid transform/predictive coding provides an effective compromise of the advantages and disadvantages of transform and predictive coding. Good results are reported at a moderate bit rate (around 1-2 bits/pixel) [43]. These coding techniques are reviewed below, with special emphasis on transform coding.

2.4.1 PREDICTIVE CODING

Predictive coding attempts to exploit the spatial redundancy in image. In predictive coding, the pixel value is predicted from the previous pixels. The predicted estimate is then subtracted from the actual pixel and the corresponding difference is quantized, coded, and transmitted. At the receiving end, the quantized difference is used to form a reconstruction of the image.

DPCM

Among predictive coding techniques, differential pulse-code modulation (DPCM) [51] can be considered as the basic system. A block diagram of a digital DPCM system is shown in Fig. 2.4.1. In linear predictive coding, for example, the predictive

value $\hat{x}_n$ is calculated for each input pixel, x_n, by using the following equation [47],

$$\hat{x}_n = \sum_i a_i x_{n-i} \tag{2.4.1}$$

The difference, e_n, between the true pixel value, x_n, and its predicted value, $\hat{x}_n$, is quantized, coded and transmitted. The predictor can be optimized in terms of the minimum mean-square error. In other words, the coefficients a_i can be chosen so that the variance of the predictive error $e_n = x_n - \hat{x}_n$ is minimized. The optimal set of coefficients can be obtained from the following equation,

$$\sum_i a_{i,opt} R(j-i) = R(j) \tag{2.4.2}$$

where $R(i)$ is the correlation function. As a special case, in *Delta Modulation* [50], the difference between the pixel x_n and its predicted value $\hat{x}_n$ is quantized into one of two levels, as shown in Fig. 2.4.2. That is, if the difference is positive, $+\Delta$ is coded, otherwise, $-\Delta$; in other words, only two possible levels are allowed.

Adaptive Predictive Coding

Predictive coding is local in structure and, therefore, quite sensitive to changes in the statistics of the data. When the input data is far from stationary, a fixed-design predictive coder will yield an inconsistent performance in terms of the rate distortion function. Adaptive techniques have been used to advantage in these cases. The approaches generally fall into one of two classes: an adaptive predictor with a fixed quantizer, or an adaptive quantizer with a fixed predictor [52]. In these adaptive approaches, the parameters involved in designing the predictor and the quantizer are required to keep up with the changing statistics of the input data. For example, in the case of the linear predictor, the correlations and the corresponding weighting coefficients are recomputed periodically [52].

2.4.2 TRANSFORM CODING

In transform coding [53-62], the input image X of $N \times N$ pixels first undergoes an *orthogonal transform* A ($A' = A^{-1}$), as shown in Fig. 2.4.3,

$$Q = AXA', \tag{2.4.3}$$

and the resulting decorrelated coefficients Q_{ij}, $i,j = 0,...,N-1$ of the transformed image are then *quantized* with a variable number of bits, the number being specified by the *bit allocation map*. At the receiver, the inverse transform is performed to recover the coded image $\hat{X}$, that is,

$$\hat{X} = A'\hat{Q}A, \tag{2.4.4}$$

where $\hat{Q}$ is the quantized transformed image.

Transformation

An optimum orthogonal transform should completely decorrelate the transform coefficients, Q_{ij}, $i,j = 0,1,...N-1$, and therefore, minimize the geometric mean of the variances of the transform coefficients [41,47]. It is shown below that minimizing the geometric mean of the variances of the transform coefficients corresponds to minimizing the reconstruction error variance in transform coding. The (discrete) Karhunen-Loeve transform is a well-known optimum orthogonal transform. However, due to the large amount of computation involved in implementing the Karhunen-Loeve transform, it is usually replaced by suboptimal transforms, such as, Fourier, sine, cosine, Hadamard, etc. Among these, the (discrete) cosine transform (DCT) has been shown to come closest to the Karhunen-Loeve transform in its energy compaction properties [46]. In this thesis, the DCT is the transform of choice.

Two useful properties of orthogonal transform coding are now highlighted [47]:

1) The average energy of the transformed image is equal to the average energy of the input image, that is,

$$\frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma_{ij}^2 = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma_{x,ij}^2 = \sigma_x^2. \tag{2.4.5}$$

Here $\sigma_{ij}^2 = E(Q_{ij}^2),\ i,j = 0,1,...N-1$ is the second moment of the (i,j)th transform coefficient and $\sigma_{x,ij}^2 = E(X_{ij}^2),\ i,j = 0,1,...N-1$ is the second moment of the (i,j)th input element.

2) The average reconstruction error variance is equal to the average quantization error variance in the transform domain, that is,

$$\sigma_r^2 = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma_{r,ij}^2 = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma_{q,ij}^2 = \sigma_q^2. \tag{2.4.6}$$

Here $\sigma_{r,ij}^2 = E((X_{ij} - \hat{X}_{ij})^2),\ i,j = 0,1,...N-1$, is the reconstruction error variance of the (i,j)th input element and $\sigma_{q,ij}^2 = E((Q_{ij} - \hat{Q}_{ij})^2)$, $i,j = 0,1,...N-1$, is the quantization error variance of the (i,j)th transform coefficient.

Equations (2.4.5) and (2.4.6) imply that the analysis in the spatial domain will be completely reflected in the transform domain in terms of the average value of the second moment.

Quantization

In terms of scalar quantization, the quantization error variance σ_q^2 can be related to the input signal variance σ_x^2 as follows [47],

$$\sigma_q^2 = \epsilon_q^2 \sigma_x^2 = \epsilon^2 2^{-2B} \sigma_x^2 \tag{2.4.7}$$

Here $\epsilon_q^2 = \epsilon^2 2^{-2B}$ is the quantizer performance factor which depends on the probability density function (PDF) of the input signal and on the quantizer characteristic, especially as regards the number of quantization levels. The quantity ϵ^2 can be considered as a variable correction factor that takes into account the performance of a practical quantizer and B is the number of bits assigned to the quantizer.

Bit Allocation

The bit allocation map indicates the number of bits to be assigned to each coefficient in the transform domain. The advantage of using an optimal bit allocation in the transform domain is now evaluated. It is first shown that there is little benefit obtained by transforming the input image without optimizing the bit allocation; in other words, if equal bit allocation is used. We then calculate the gain that can be achieved by using an optimal bit allocation subject to the minimization of the average coefficient quantization error variance for a fixed bit rate.

For the sake of the analysis, it is assumed that all elements in the spatial domain and transform domain have probability distribution functions of the same type but with different variances. An equal bit allocation scheme is assumed; that is, all the elements $X_{ij},\ i,j = 0,1,...N-1$ in the spatial domain and $Q_{ij},\ i,j = 0,1,...N-1$ in the transform domain are assigned the same number of bits B (the average bit rate). From (2.4.7), the quantization error variance for the element X_{ij} of the input image X can be written as,

$$\sigma^2_{q,SD,ij} = \epsilon^2_{SD,ij} 2^{-2B} \sigma^2_{x,ij} = \epsilon^2_{SD} 2^{-2B} \sigma^2_{x,ij}. \tag{2.4.8}$$

Here, $\epsilon^2_{SD,ij} = \epsilon^2_{SD}$ based on the assumption that all the elements in the spatial domain have probability distribution functions of the same type but different variances and that an equal bit allocation is used. The average quantization error variance in the spatial domain is, therefore,

$$\begin{aligned}
\sigma^2_{q,SD} &= \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{q,SD,ij} \\
&= \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \epsilon^2_{SD} 2^{-2B} \sigma^2_{x,ij} \\
&= \epsilon^2_{SD} 2^{-2B} \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{x,ij} \\
&= \epsilon^2_{SD} 2^{-2B} \sigma^2_x.
\end{aligned} \tag{2.4.9}$$

In a similar fashion, the quantization error variance for the transform coefficient Q_{ij}

can be written as,

$$\sigma^2_{q,TD,ij} = \epsilon^2_{TD,ij} 2^{-2B} \sigma^2_{ij} = \epsilon^2_{TD} 2^{-2B} \sigma^2_{ij}. \tag{2.4.10}$$

Once more $\epsilon^2_{TD,ij} = \epsilon^2_{TD}$ based on the same assumption as in the spatial case, and the average quantization error variance in the transform domain is, therefore,

$$\begin{aligned}
\sigma^2_{q,TD} &= \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{q,TD,ij} \\
&= \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \epsilon^2_{TD} 2^{-2B} \sigma^2_{ij} \\
&= \epsilon^2_{TD} 2^{-2B} \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{ij} \\
&= \epsilon^2_{TD} 2^{-2B} \sigma^2_x.
\end{aligned} \tag{2.4.11}$$

For both cases, the average quantization (reconstruction) error variances is proportional to the average energy of the input image (or the arithmetic mean of the transform coefficient variance). If logarithmic quantization is assumed in both the spatial and the transform domains,

$$\epsilon^2_{SD} = \epsilon^2_{TD}. \tag{2.4.12}$$

Note that logarithmic quantization is very insensitive to a mismatch relative to the PDF of the input sources [47]. Therefore, it is clear from (2.4.9) and (2.4.11) that ,

$$\sigma^2_{q,TD} = \sigma^2_{q,SD}. \tag{2.4.13}$$

In other words, there is little benefit obtained by transforming the input image and just using an equal bit allocation in transform domain (2.4.13).

We now calculate the gain that can be achieved by optimally distributing (allocating) the bits among the transform coefficients. Let B_{ij} be the number of bits allocated to the (i,j)th transform coefficient, then, B, the average bit rate is given by,

$$B = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} B_{ij}. \tag{2.4.14}$$

The average coefficient quantization error variance, σ_q^2, is

$$\sigma_q^2 = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma_{q,ij}^2 = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \epsilon_{ij}^2 2^{-2B_{ij}} \sigma_{ij}^2, \tag{2.4.15}$$

where

$$\sigma_{q,ij}^2 = \epsilon_{ij}^2 2^{-2B_{ij}} \sigma_{ij}^2. \tag{2.4.16}$$

The optimality is with regard to the minimization of σ_q^2 under the constraint of a fixed average bit rate B. The calculation of the gain follows that given in reference [47], where the ϵ_{ij}^2 $i,j = 0,1,...N-1$, are no longer constant assumed in [47]. In fact, ϵ_{ij}^2 $i,j = 0,1,...N-1$ depend upon both the number of bits assigned to and the PDF of the coefficients. Table 2.4.1 provides the numerical values of ϵ^2 for both PDF-optimized uniform quantization and PDF-optimized nonuniform quantization with four different inputs PDF's (U: Uniform, G: Gaussian, L: Laplacian and Γ : Gamma), derived from Tables 4.1 and 4.4 in [47]. As seen in Table 2.4.1, ϵ^2 varies with the number of quantization levels and the input PDF.

We calculate the gain using the Lagrange multiplier method,

$$\frac{\partial}{\partial B_{ij}} [\sigma_q^2 - \lambda (B - \frac{1}{N \times N} \sum_{i,j=0}^{N-1} B_{ij})] = 0, \qquad i,j = 0,1,...N-1. \tag{2.4.17}$$

An optimum bit allocation is then given by:

$$\begin{aligned} B_{ij} &= B + \frac{1}{2} \log_2 \epsilon_{ij}^2 \sigma_{ij}^2 - \frac{1}{N \times N} \sum_{k,l=0}^{N-1} \frac{1}{2} \log_2 \epsilon_{kl}^2 \sigma_{kl}^2 \\ &= B + \frac{1}{2} \log_2 \frac{\epsilon_{ij}^2 \sigma_{ij}^2}{[\prod_{k,l=0}^{N-1} \epsilon_{kl}^2 \sigma_{kl}^2]^{1/(N \times N)}}, \quad i,j = 0,1,...N-1. \end{aligned} \tag{2.4.18}$$

It should be observed that the number of quantizer levels $2^{B_{ij}}$ is proportional to both the coefficient variance, σ_{ij}^2, and the quantizer performance factor, ϵ_{ij}^2. The bit allocation formula (2.4.18) also implies identical error variances in the coefficient quantization:

$$\sigma_{q,ij}^2 = \epsilon_{ij}^2 2^{-2B_{ij}} \sigma_{ij}^2$$

$$= 2^{-2B}\Big(\prod_{k,l=0}^{N-1} \epsilon_{kl}^2\sigma_{kl}^2\Big)^{1/(N\times N)}$$

$$= \epsilon_0^2 2^{-2B}\Big(\prod_{k,l=0}^{N-1} \sigma_{kl}^2\Big)^{1/(N\times N)} \quad , \qquad i,j = 0,1,...N-1 \tag{2.4.19}$$

where

$$\epsilon_0^2 = \Big(\prod_{k,l=0}^{N-1} \epsilon_{kl}^2\Big)^{1/(N\times N)} \quad . \tag{2.4.20}$$

Therefore, the average reconstruction error variance is

$$\min(\sigma_r^2) = \min(\sigma_q^2) = \frac{1}{N\times N}\sum_{i,j=0}^{N-1}\sigma_{q,ij}^2 = \epsilon_0^2 2^{-2B}\Big(\prod_{k,l=0}^{N-1}\sigma_{kl}^2\Big)^{1/(N\times N)} \tag{2.4.21}.$$

Note that the average reconstruction error variance is proportional to the geometric mean of the coefficient variances.

The ratio of the distortion with the optimum bit allocation (2.4.21) to that with the equal bit allocation (2.4.11) in the transform domain is, therefore,

$$\frac{\min(\sigma_q^2)}{\sigma_{q,TD}^2} = \frac{\epsilon_0^2 2^{-2B}\big(\prod_{i,j=0}^{N-1}\sigma_{ij}^2\big)^{1/(N\times N)}}{\epsilon_{TD}^2 2^{-2B}\frac{1}{N\times N}\sum_{i,j=0}^{N-1}\sigma_{ij}^2} = \frac{\epsilon_0^2}{\epsilon_{TD}^2}\frac{\big(\prod_{i,j=0}^{N-1}\sigma_{ij}^2\big)^{1/(N\times N)}}{\frac{1}{N\times N}\sum_{i,j=0}^{N-1}\sigma_{ij}^2} = \alpha. \tag{2.4.22}$$

If

$$\frac{\epsilon_0^2}{\epsilon_{TD}^2} \approx 1 \tag{2.4.23}$$

which is a reasonable approximation in the case of logarithmic quantization [47], then,

$$\alpha \leq 1 \tag{2.4.24}$$

with equality if, and only if, all the variances of the coefficients are equal.

Gain of Transform Coding

Based on the above analysis, it can be concluded that,

1) The optimal transform minimizes the geometric mean of the transform coefficient variances and keeps the arithmetic mean constant (2.4.5),

2) The optimum bit allocation results in an average reconstruction error variance proportional to the geometric mean (2.4.21), instead of the arithmetic mean, of the transform coefficient variances.

The gain due to transform coding [47] is just the reciprocal of α, i.e. the ratio of the arithmetic mean to the geometric mean of coefficient variances.

Adaptive Transform Coding

Performance of transform coding may be improved substantially by adapting it to changes in image statistics [52]. One effective approach is to adapt the bit allocation map to changes in the statistics of the image. Adaption of this type has been considered by Chen and Smith [62]. In their technique [62], each image block is classified into one of several classes based upon its activity. A larger number of bits is then assigned to the blocks with higher activity and fewer bits to blocks with lower activity. The Chen and Smith technique is important because it is often used as a quasi-standard for comparing coding performances.

2.4.3 HYBRID TRANSFORM/PREDICTIVE CODING

The term of *Hybrid* refers to the combination of transform and predictive coding techniques [63]. A block diagram of a hybrid coder is presented in Fig. 2.4.4. A one-dimensional transform is first taken along rows of image and then, a one-dimensional predictive coding technique is applied to the transform coefficients in a columnwise fashion. In general, the performance characteristics of hybrid coding lies between that of transform and predictive coding.

2.5 VECTOR QUANTIZATION

Rate distortion theory [25-29] indicates that better performance can be achieved by coding vectors instead of scalars. However, this theory does not provide constructive design technique for vector coders. Vector quantization design techniques

and various vector quantization structures have only been developed in recent years [64-90]. A tutorial paper by Gersho and Cuperman [71] presents a review of vector quantization for speech coding. A broader and more comprehensive review of vector quantization is given in [72,75]. In this section, the basic concept of vector quantization is first presented, including a discussion on complexity and bit rates. Methods for reducing the complexity in vector quantization are then described and followed by a review of vector quantization for image coding.

2.5.1 BASIC CONCEPT OF VECTOR QUANTIZATION

Vector quantization concerns the joint quantization of a block of input data. In vector quantization [72], the input vector $V = \{V_i\}$ drawn from an M-dimensional Euclidean space is mapped into a finite set (codebook) of reproduction vectors (codewords) $\hat{V} = \{\hat{V}_i, \ i = 1, 2, ...N_c\}$ contained in the space . In other words, the input vector V is represented by $\hat{V}$, the vector quantized version of V. This can be written as,

$$q(V) = \hat{V} \tag{2.5.1}$$

where $q(.)$ is the quantization operation.

In designing a codebook with N_c levels, the M-dimensional Euclidean space of the input vector V is partitioned into N_c cells $\{\Pi_i, \ i = 1, 2, ...N_c\}$, where each cell Π_i is associated with a representative vector, or codeword, $\hat{V}_i$. The quantizer then assigns the codeword $\hat{V}_i$ if the input V is in cell Π_i,

$$q(V) = \hat{V}_i \qquad \text{if } V \in \Pi_i. \tag{2.5.2}$$

This implies that the vector mapping is completely characterized by the partition $\Pi = \{\Pi_i, \ i = 1, 2, ...N_c\}$ of the input space V, which assigns an input vector $V \in \Pi_i$ to the representative vector $\hat{V}_i$. An optimal quantizer should minimize the average coding distortion over all the other quantizers with N_c levels. If the distortion is

defined as follows,

$$D = E[d(V,\hat{V})] = \sum_{i=1}^{N_c} P(V \in \Pi_i) E[d(V,\hat{V}_i)|V \in \Pi_i]. \qquad (2.5.3)$$

then the optimal quantizer must simultaneously satisfy two necessary conditions [67]:

1) The optimal quantizer must use a minimum-distortion or nearest neighbor selection rule,

$$q(V) = \hat{V}_i, \qquad \text{iff } d(V,\hat{V}_i) < d(V,\hat{V}_j), \quad \text{for all } j \qquad (2.5.4)$$

where $q(.)$ is the quantization operator. In other words, the quantizer chooses the code vector that results in the minimum distortion with respect to V.

2) The codewords $\{\hat{V}_i\}$ are chosen so as to minimize the average distortion in each cell Π_i,

$$D_i = E[d(V,\hat{V})|V \in \Pi_i], \qquad i = 1,2,...N_c. \qquad (2.5.5)$$

In other words, the $\{\hat{V}_i\}$ are simply the centroids of the corresponding cells $\{\Pi_i\}$.

A procedure which has been applied successfully to codebook design is an iterative clustering algorithm, referred to as the LBG algorithm (Linde, Buzo and Gray [65]), or the generalized Lloyd algorithm [102]. This algorithm produces a quantizer such that the two necessary conditions for optimality are satisfied. The steps involved in this algorithm are as follows,

0) *Initialization:* Given a distortion threshold $\varepsilon \geq 0$, an initial codebook $\hat{V}_0 = \{\hat{V}_{i,0}; i = 1,2,...N_c\}$, and a training sequence of input vectors, $\{V_j, j = 1,2,...T\}$. Set $m = 0$ and $D_{-1} = \infty$.

1) *Partition:* Each training vector is assigned to the *nearest neighbor* output vector and a partition $\{\Pi_{i,m}\ i = 1,2,...N_c\}$ is obtained,

$$V_j \in \Pi_{i,m}, \qquad \text{if } d(V_j,\hat{V}_{i,m}) \leq d(V_j,\hat{V}_{l,m}) \quad \text{for all } l. \qquad (2.5.6)$$

2) *Codebook Updating:* The codewords of the $(m+1)$st codebook are formed,

$$\hat{V}_{i,m+1} = \frac{1}{T_{i,m}} \sum_{V \in \Pi_{i,m}} V, \qquad i = 1, 2, ...N_c. \tag{2.7.5}$$

Here, $T_{i,m}$ is the number of vectors in $\Pi_{i,m}$ and the new codewords are the centroids of the corresponding partition.

4) *Termination:* Compute the average distortion

$$D_m = \frac{1}{T} \sum_{j=1}^{T} \min_{\hat{V} \in \hat{V}_m} d(V_j, \hat{V}). \tag{2.5.8}$$

If $(D_{m-1} - D_m)/D_m \leq \varepsilon$, stop and let $\hat{V}_m$ be the final codebook. Otherwise, set $m = m + 1$ and go to step 1.

One important feature of this algorithm is that the codebook can be designed without requiring explicit knowledge of the source statistics. Furthermore, this algorithm has been shown to converge to a local optimum [65].

In general, however, the solution is not unique [75], but depends upon the initial codebook. There are two basic approaches to obtain the initial codebook [72]:

1) The first approach is to start with a codebook of the required size. An example is to take the first N_c vectors in the training sequence as the initial codewords [72]. A modification is to select suitably spaced vectors as the initial codewords from the training vectors [72].

2) The second approach is to start with a small codebook and recursively construct a larger one; an example is the *binary splitting* method [65]. In this method, the starting point is one codeword, the mean vector of the training vectors. The codeword is split by a fixed perturbation vector to form two codewords and the LBG algorithm is applied to yield a codebook of two levels. Each of the two codewords is then split and the LBG is again applied

to produce a codebook with four levels. The procedure continues in this way until the required number of codewords are generated.

2.5.2 COMPLEXITY AND BIT RATE CALCULATION

Once the codebook is generated, quantization is performed by computing the distortion between the input and each codeword in the codebook. The codeword which minimizes the distortion is chosen as the quantized version of the input. This type of quantization is known as a full search since each input vector is compared with all the codewords in the codebook. The computational cost for quantizing one input vector using the full search is proportional to the dimensionality of the vectors, D_v, and the number of codewords, N_c, [75],

$$C = k_c \times D_v \times N_c \tag{2.5.9}$$

where k_c is a constant. The storage cost for the codebook is given by,

$$M = k_m \times D_v \times N_c \tag{2.5.10}$$

where k_m is also a constant. If an equal length coder is used to encode the labels corresponding to the selected codewords, the average number of bits per dimension for the labels is given,

$$R_l = \frac{\log_2 N_c}{D_v} \qquad bits/dimension. \tag{2.5.11}$$

Substituting (2.5.11) in (2.5.9) and (2.5.10), the computational and storage costs can be, respectively, rewritten as,

$$C = k_c \times D_v \times 2^{D_v R_l}, \tag{2.5.9}'$$

$$M = k_m \times D_v \times 2^{D_v R_l}. \tag{2.5.10}'$$

Both the computational and storage costs are exponential in the dimensionality of the vectors D_v and the number of bits per dimension, R_l. Therefore, both high-rate

and large-dimensionality vector quantization can become impractical. It should be emphasized that the R_l defined in equation (2.5.11) is in fact the upper bound for encoding the labels. The use of a variable length encoder for the labels, such as Huffman coder, can result in certain reductions in the bit rate.

In image coding, the codebook can be adapted to each input image [S6,S8]. In other words, for each image, a *new* codebook is generated and is transmitted to the receiver. The corresponding bit rate for the codebook, or *overhead*, is given by,

$$R_c = \frac{D_v \times N_c \times B_c}{N \times N} = \frac{D_v \times 2^{D_v R_l} \times B_c}{N \times N} \quad bits/pixel, \qquad (2.5.12)$$

where B_c is the average number of bits per component of the codeword and $N \times N$ is the size of the image. The overhead for transmitting the codebook is seen to grow exponentially with the dimensionality of the vectors and the number of bits per dimension. This implies that at higher bit rates, a larger dimensionality leads to a considerable overhead for transmitting the codebook. Table 2.5.1 shows the overheads for transmitting the codebooks at different bit rates and vector dimensions. For example, if $D_v = 4 \times 4$, $N_c = 256$, $B_c = 8$, and $N \times N = 256 \times 256$, then, $R_l = \frac{\log_2 256}{4 \times 4} = 0.5$ bits/pixel and $R_c = \frac{4 \times 4 \times 256 \times 8}{256 \times 256} = 0.5$ bits/pixel. As seen, 50% of the bits are required for coding the codebook.

2.5.3 METHODS FOR REDUCING COMPLEXITY

Although vector quantization can achieve significant performance advantage over scalar quantization, the advantage is obtained at the expense of considerable computational and storage costs. For full search vector quantization, the computational and storage costs are exponentially proportional to the dimensionality of the vectors and the number of bits per dimension (2.5.9-2.5.10). Several techniques have been proposed to reduce the computational and the storage costs in a full search vector quantization.

Tree-structured Vector Quantization

Instead of using a full search, quantization can be operated on a binary tree codebook [66]. In binary tree-structured vector quantization (Fig. 2.5.1), the M-dimensional space is first divided into two regions, then each region is further divided into two subregions, and so on, until the space is divided into N_c regions, or cells. In quantizing each input vector, a decision is made between one of the two subregions at each layer of the binary tree and the representative codeword is found at the last layer, In other words, an input vector is quantized by traversing (searching) the tree along a path that gives the minimum distortion at each node in the path. The computational cost [75] is proportional to $2 \times D_v \times \log_2 N_c = 2 \times D_v \times (D_v \times R_l)$ and the total storage cost is proportional to $M = 2 \times D_c \times (N_c - 2)$. Note that the computational cost is now proportional to the square of the dimensionality of the vectors and the number of bits per dimension.

Multistage Vector Quantization

Multistage vector quantization was first proposed by Juang [69]. Fig. 2.5.2 shows a two-stage example of multistage vector quantization. The input X is first vector quantized using a quantizer with N_{c1} levels. The residual error, e, between the input X and its vector quantized version $\hat{X}$ is then used as the input to a second stage vector quantizer with N_{c2} levels. The final quantized version of the input X is simply the sum of the outputs of two stages. The process can be repeated by feeding the residual errors from the second stage into the third stage, and so on. In terms of the average bit rate for the labels (2.5.11), n-stage vector quantization, where stage i has N_{ci} codewords, is equivalent to a single stage vector quantization with $\prod_{i=1}^{n} N_{ci}$ codewords. However, the computational and storage costs for the n-stage vector quantization is proportional to $D_v \times \sum_{i=1}^{n} N_{ci}$ instead of $D_v \times \prod_{i=1}^{n} N_{ci}$ for a single stage vector quantization.

Product Vector Quantization

In product vector quantization [66,73,82], the Euclidean space of the input vector is first decomposed into two or more subsets, jointly representing all the points in the space. If we let D_{vi} and N_{ci} be the dimensionality and codebook size of subset i, then the dimensionality and codebook size of the product code are, respectively, given by,

$$D_v = \sum_i D_{vi} \tag{2.5.18}$$

$$N_c = \prod_i N_{ci}. \tag{2.5.19}$$

If the subsets are separately vector quantized with their respective codebooks, the corresponding computational and storage costs are proportional to $\sum_i D_{vi} \times N_{ci}$. The corresponding computational and storage costs for a single stage vector quantization with vector dimension D_v and codebook size N_c is proportional to $D_v \times N_c = (\sum_i D_{vi})(\prod_i N_{ci})$

An example of a product vector quantization is the *Gain/Shape* vector quantization [73]. In this technique, the input vector is first normalized, prior to encoding, by the removal of the *gain* defined as the energy of the vector. The resultant normalized input vector is referred to as the *shape* vector. The gain and shape are then separately quantized; in particular, the gain is scalar quantized and the shape vector quantized. If coding is operated at a total bit rate of $R = R_g + R_s$ where R_g is the bit rate for scalar quantizing the gain and R_s is the bit rate for vector quantizing the shape, the computational and storage for vector quantization cost are proportional to $C = D_v \times 2^{D_v R_s}$. This is to be compared with $D_v \times 2^{D_v R_i}$ for a single stage full search vector quantization at a bit rate of $R_i = R = R_g + R_s$.

Another example of product vector quantization is the *Mean/Shape* vector quantization [82]. The mean of each vector is subtracted to yield a new shape vector with zero mean. The shape vector is then vector quantized while the mean is separately scalar quantized.

2.5.4 A REVIEW OF VECTOR QUANTIZATION FOR IMAGE CODING

A block diagram of the various steps involved in implementing vector quantization as applied to image coding is provided in Fig. 2.5.3. The first step is to decompose the input image into a set of vectors. A subset of the vectors is then chosen as a training sequence from which a codebook is generated, normally with the use of the generalized Lloyd clustering algorithm [65]. Finally, for each input vector, the closest codeword is determined and the corresponding label of this codeword is transmitted. Vector quantization for image coding is now reviewed from the following two aspects: vector formation, and training sequence and codebook generation.

Vector Formation

Vector formation, or decomposition of image, is the first step in vector quantization. Many different approaches to vector formation have been proposed. The approaches can be classified into two categories: Direct and Indirect.

Direct Approach

In the direct approach, the vectors are formed using the original pixel values. For example, in an early vector quantization scheme for multi-spectral Landsat image coding [76], the vectors were formed for each pixel from four Landsat bands, thus, exploiting the spectral correlation. Lowitz [77] proposed a vector quantization scheme for a monochrome images where the image is divided into rectangular blocks and the pixels from a number of different blocks are combined into a vector. A more widely used method to vector formation is to decompose the input image into spatially contiguous, nonoverlapping blocks (vectors) of pixels [78-81]. Recent experiments on combining both spectral and spatial information were described by Goldberg, Boucher and Shlien [86].

Indirect Approach

A number of examples for the indirect approach are now described.

1. Orthogonal Transform Coefficients

In orthogonal block transform coding, the transform results in transform coefficients which are decorrelated and in general, concentrates the energy of image in the lower coefficients. The resulting coefficients can be used as the components of vectors in image coding. In [S3,S8], only the coefficients with the highest energies are combined into a vector. An alternative approach is proposed by Aizawa, etc. [S7] where each transform block (8×8) is partitioned into 14 vectors and each vector quantized separately.

2. LPC Parameter

Sun and Goldberg describe an adaptive linear predictive image coding using vector quantization [S4]. In this scheme, the image is decomposed into blocks. For each block, the LPC (linear predictive coding) parameters (linear predictor coefficients) are calculated and concatenated to form a vector.

3. Predicted Errors

Hang and Woods [S5] present an image coding scheme which combines the features of predictive coding and vector quantization. The basic idea of this scheme is to first use a predictor to remove the predictable redundancy in the image and then use a vector quantizer to further remove the residual redundancy. In this scheme, the vectors are formed from the predicted errors.

4. Mean/Residual Errors

In Mean/Shape vector quantization [S2], the mean of each block is extracted and a difference signal is formed by subtracting the mean from the image. The mean is encoded by a scalar quantizer and the difference signal by vector quantization.

Training Sequence and Codebook Generation

In vector quantization, an optimal codebook should completely reflect the statistics of the input vectors. A codebook generated by the generalized Lloyd algorithm is optimized for the particular training sequence, chosen as the representative of the input image source. We now discuss two methods for training sequence and codebook generation: universal and adaptive.

In the universal method of [78,79], the vectors for training sequence are drawn from many different kinds of images. The resultant universal codebook can then be fixed at both the transmitter and the receiver and therefore, there is no overhead required for transmitting the codebook. There are two problems with the universal method. First of all, images outside of the training sequence may not always be represented well. Secondly, to ensure a good fit for all the images, a codebook with many codewords is needed, thus, increasing the bit rate. One proposed modification is Classified Vector Quantization [89], where the vectors in a training sequence are divided into a finite set of classes (states) based upon some feature of the vectors, for example, the edge content. For each class, a separate codebook is generated based upon the vectors (the sub-training sequence) in the class. In quantization, the class for each input vector is first determined and the input vector is then encoded with only the codebook specifically designed for the class, thus, significantly reducing the searching complexity. In a recent paper [90], Aravind and Gersho exploit the memory of the input image to determine the state of the input vector. In other words, the state of each input vector is predicted from the previous reproduction vectors. As there is no side information required for indicating the choice of codebook, the bit rate can be reduced.

In the adaptive method of [86,88], the training sequence and the codebook are adapted to each input image. For each input image, a subset of vectors formed from the image is taken as the training sequence and a codebook is generated. One further refinement is to break up the input image into subimages, and design a local codebook

for each subimage [88]. The local codebooks better match the local statistics, thus, smaller codebook can be used. However, an overhead is required for transmitting the codebook for each input image.

2.6 SUMMARY

In this chapter, we have reviewed the following aspects related to image coding: the basic concepts of entropy and rate distortion function, distortion measure, lossless coding, lossy coding and vector quantization.

Entropy and Rate Distortion Function: For a given image source X, the entropy $H(X)$ indicates the minimum bit rate for lossless reproduction and the rate distortion function $R(D)$ provides the lower bound of bit rate for a fixed distortion D.

Distortion Measure: The normalized mean square error (NMSE) is widely used in evaluating the quality of the coded image, but, it is not always correlated well with subjective rating. There has been considerable effort to improve the correlation with subjective rating. So far, there is no accepted standard criterion. We adopt the NMSE as the distortion measure and use the (bit) rate distortion (NMSE) curve in evaluating the performance of a coding system. To compare the different coding schemes, the simulations in the following chapters are carried out on the same images.

Lossless Coding: Huffman coding provides an optimal procedure for encoding a statistically independent source. Run length coder is particularly useful for images with long runs. We describe new lossless coding methods which improve Huffman coding.

Lossy Coding: Predictive, transform and hybrid coding are reviewed, with special emphasis on transform coding as it forms the basis for the following chapters. Once an orthogonal transform has been chosen, optimizing a particular transform coding scheme reduces to finding an optimal bit allocation such that the average distortion is minimized for a fixed bit rate. The gain using an optimal bit allocation over an equal

bit allocation in transform domain is the ratio of arithmetic mean to the geometric mean of the coefficient variances.

Vector Quantization: Vector quantization is a relatively new coding technique which achieves significant performance advantage over scalar quantization. The complexity associated with vector quantization is shown to grow exponentially with the dimensionality of the vectors and the bit rate per dimension. Many methods for reducing the complexity have been proposed, such as, tree-search vector quantization, multistage vector quantization, and product vector quantization. The techniques described in chapters 4-7 present new ways of using vector quantization techniques for image coding.

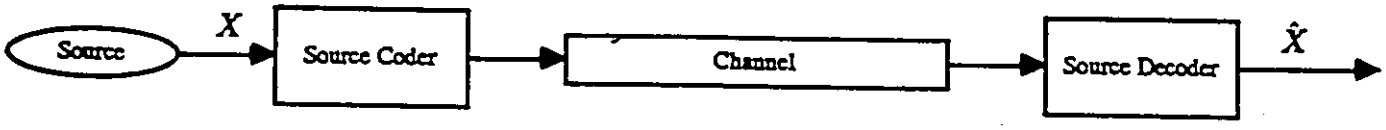


Fig. 2.1.1. A general communication system.

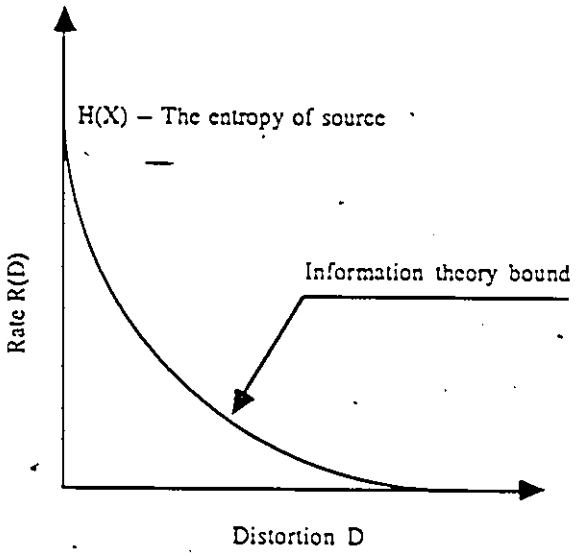
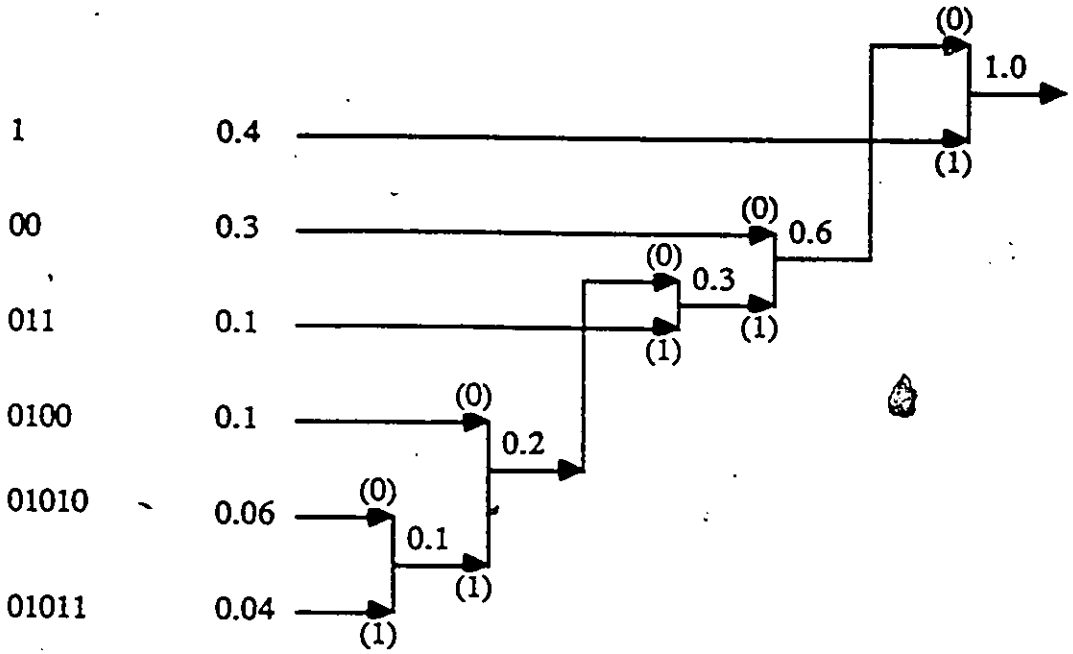


Fig. 2.1.2. An example of the rate distortion function for a discrete-amplitude source.



Entropy = 2.14 bits
Bit Rate = 2.20 bits
Efficiency = 97.3 %

Fig. 2.3.1. An example showing the construction of a Huffman coding tree.

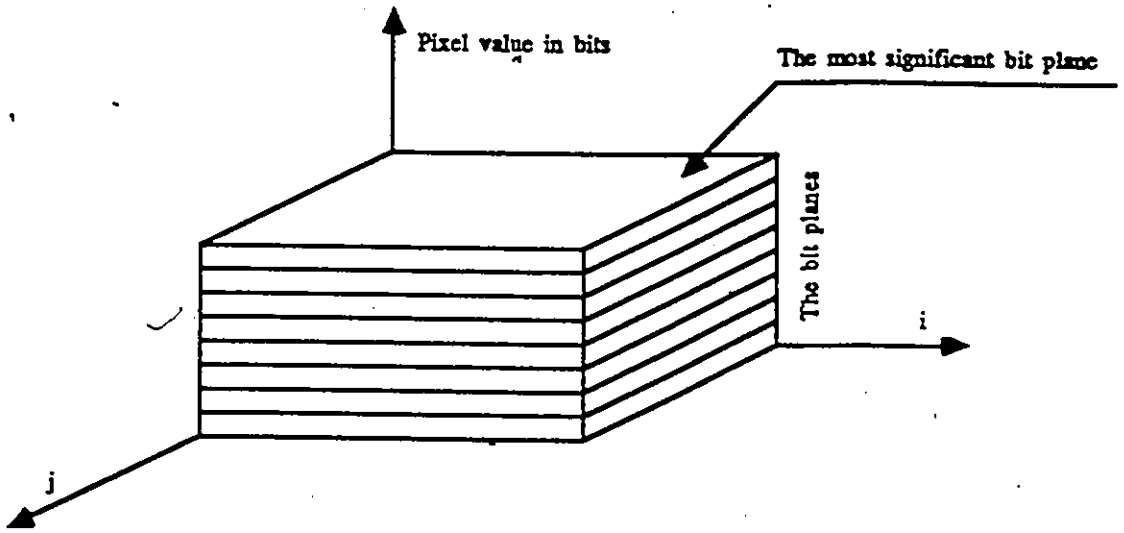


Fig. 2.3.2. A bit plane representation of an 8-bit image where the ith significant bits are allocated in the ith bit plane.

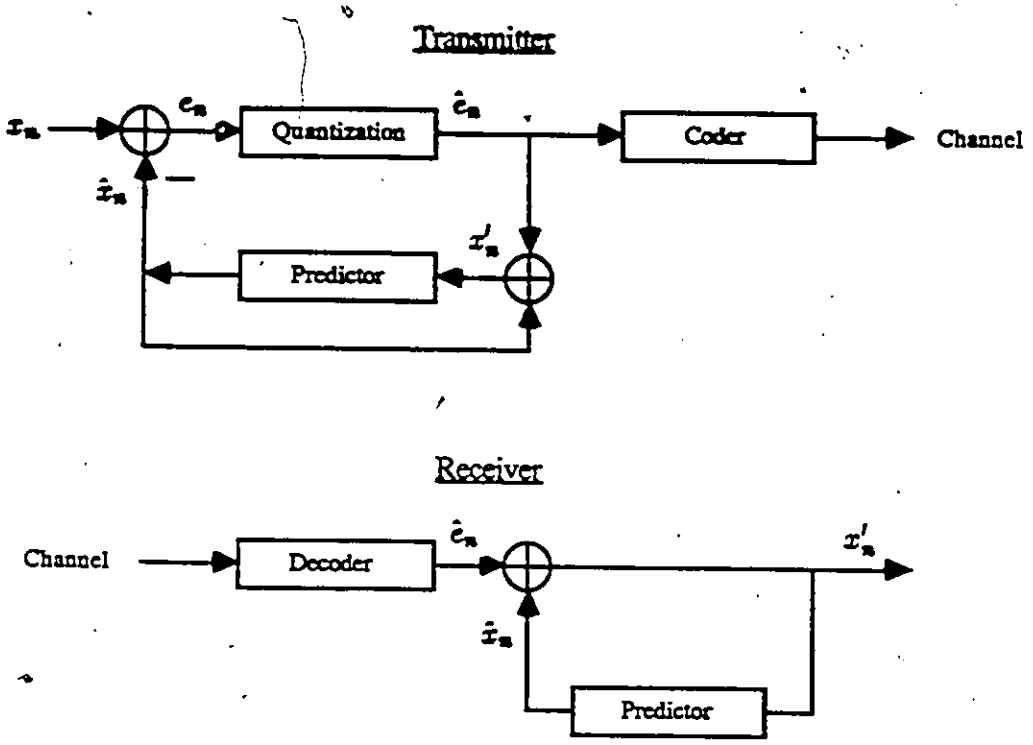


Fig. 2.4.1. Block diagram of the DPCM system.

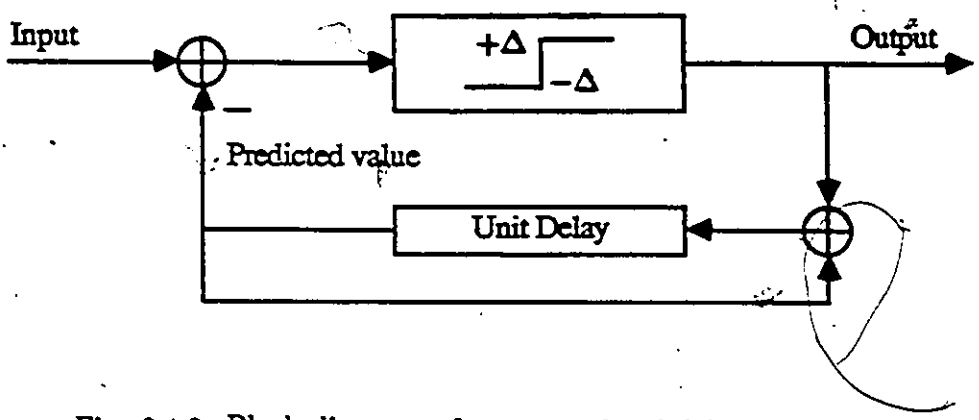


Fig. 2.4.2. Block diagram of a conventional delta modulator.

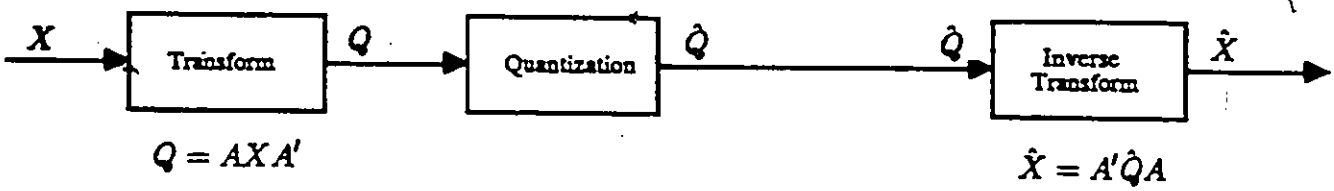


Fig. 2.4.3. Block diagram of transform coding. The input image X first undergoes an orthogonal transform A ($A' = A^{-1}$). The resulting transformed image Q is then quantized before transmission. At the receiving end, the inverse transform is performed to recover the coded image $\hat{X}$.

Table 2.4.1a Quantizer performance factor for PDF-optimized uniform quantizer

B (bits)	Quantizer Performance Factor $\epsilon^2 2^{-2B}$				Variable Correction Factor ϵ^2			
	U	G	L	Γ	U	G	L	Γ
1	0.250035	0.363078	0.500035	0.666807	1.0001	1.4523	2.0001	2.6672
2	0.062517	0.118850	0.196336	0.319890	1.0003	1.9016	3.1414	5.1182
3	0.015631	0.037411	0.071779	0.132434	1.0004	2.3943	4.5939	8.4758
4	0.003908	0.011535	0.025351	0.050119	1.0006	2.9528	6.4899	12.8304
5	0.000977	0.003491	0.008710	0.017824	1.0007	3.5752	8.9187	18.2516
6	0.000244	0.001040	0.002911	0.006081	1.0008	4.2595	11.9223	24.9092
7	0.000061	0.000307	0.000948	0.002000	1.0010	5.0283	15.5389	32.7657
8	0.000051	0.000092	0.000306	0.000647	0.9988	6.0601	20.0669	42.4111

Table 2.4.1b Quantizer performance factor for PDF-optimized nonuniform quantizer

B (bits)	Quantizer Performance Factor $\epsilon^2 2^{-2B}$				Variable Correction Factor ϵ^2			
	U	G	L	Γ	U	G	L	Γ
1	0.250035	0.363078	0.500035	0.666807	1.0001	1.4523	2.0001	2.6672
2	0.062517	0.117490	0.176198	0.231739	1.0003	1.8798	2.8192	3.7078
3	0.015631	0.034514	0.054450	0.070469	1.0004	2.2089	3.4848	4.5100
4	0.003908	0.009506	0.015382	0.019634	1.0006	2.4335	3.9377	5.0262
5	0.000977	0.002506	0.004102	0.005188	1.0007	2.5663	4.2005	5.3125
6	0.000244	0.000647	0.001062	0.001340	1.0008	2.6507	4.3487	5.4878
7	0.000061	0.000166	0.000270	0.000341	1.0010	2.7128	4.4200	5.5901

U: Uniform, G: Gaussian, L: Laplacian and Γ: Gamma

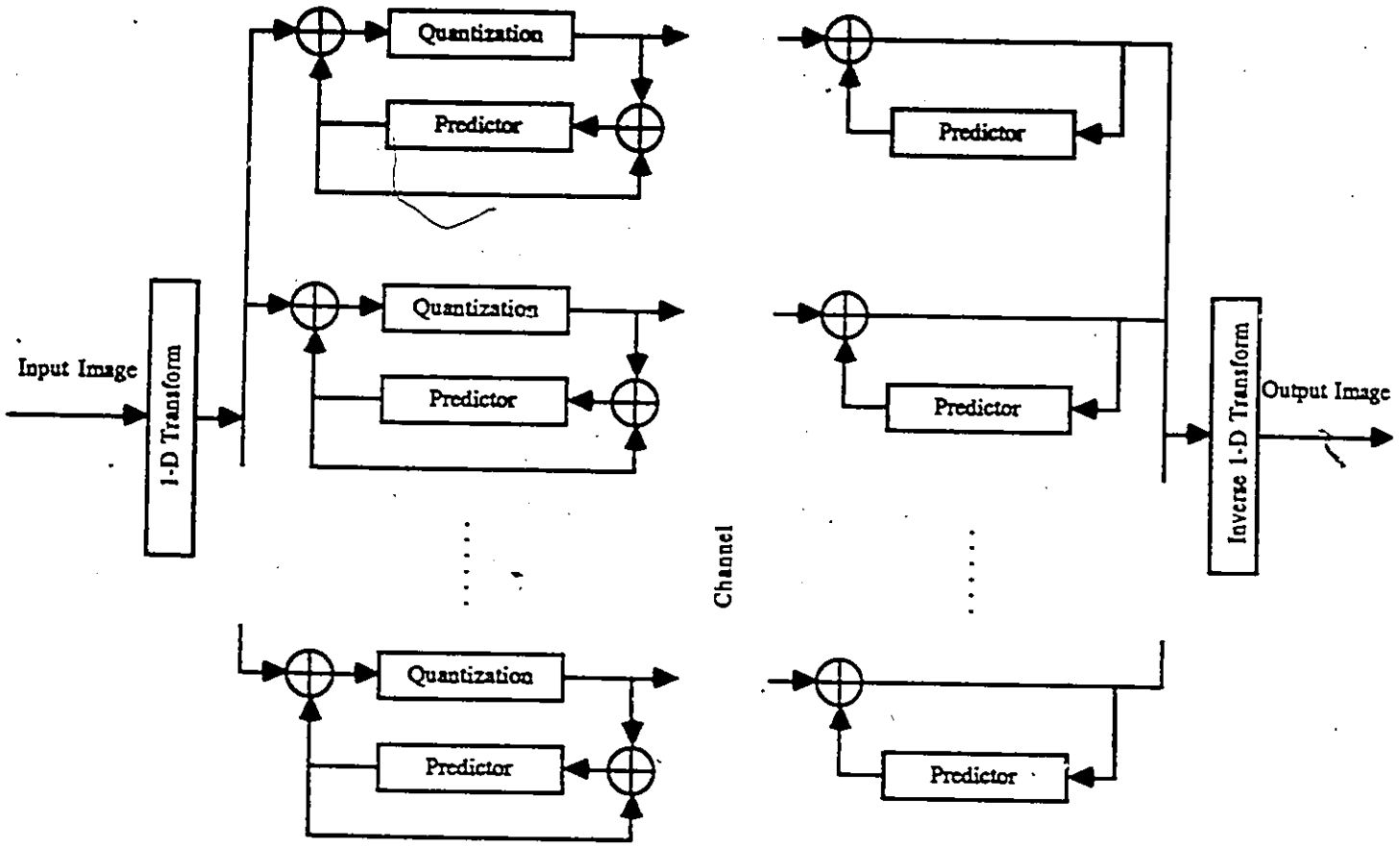


Fig. 2.4.4. Block diagram of a hybrid transform/DPCM coder.

Table 2.5.1 The bit rate distortion for the labels and the codebook at the different vector dimensions (N*N=256*256 and Bc=8 bits/pixel)

Dv	Nc	RI (bits/pixel)	Rc (bits/pixel)
2*2	4	0.5000	0.0019
2*2	8	0.7500	0.0039
2*2	16	1.0000	0.0078
2*2	32	1.2500	0.0156
2*2	64	1.5000	0.0312
4*4	16	0.2500	0.0312
4*4	32	0.3125	0.0625
4*4	64	0.3750	0.1250
4*4	128	0.4375	0.2500
4*4	256	0.5000	0.5000
4*4	512	0.5625	1.0000

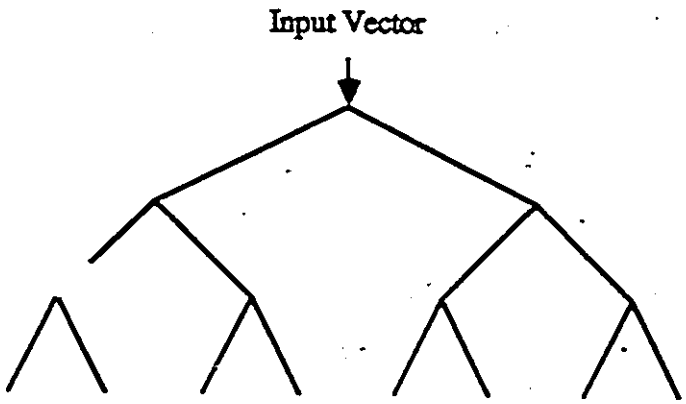


Fig. 2.5.1. Uniform tree for a binary-search vector quantization. In quantizing each input vector, a decision is made between one of the two subregions at each level of the binary tree, the representative codeword is found at the last level.

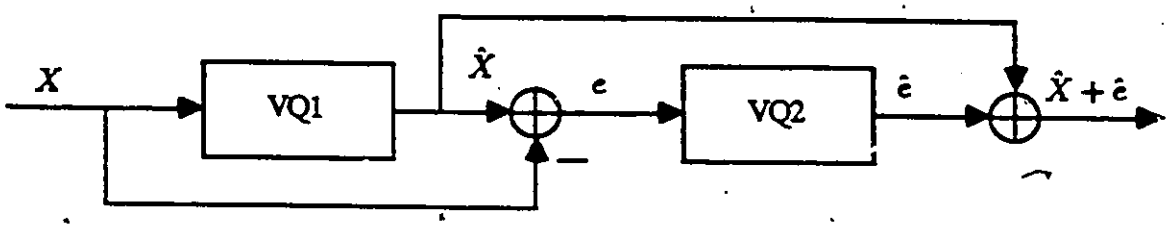


Fig. 2.5.2. A 2-stage vector quantizer. The input source is first vector quantized with quantizer VQ_1. The residual error between the original image and its vector quantized version is then used as the input to a second stage vector quantizer VQ_2. The final quantized version of the image is the sum of the outputs of two stages.

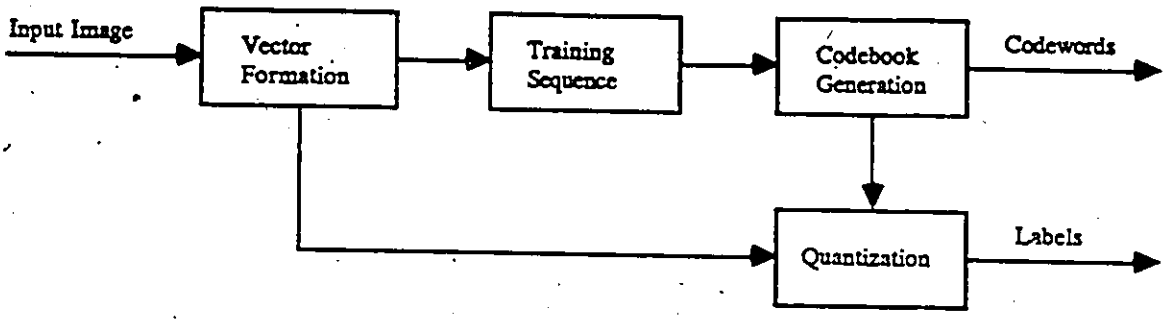


Fig. 2.5.3. Block diagram of vector quantization applied to image coding. The input image is first decomposed into a set of vectors. A subset of the vectors is then chosen as a training sequence from which a codebook is generated, normally by using the generalized Lloyd clustering algorithm. In quantization, for each input vector, the closest codeword is determined and the corresponding label of this codeword is transmitted.

3. A REVIEW OF PROGRESSIVE IMAGE TRANSMISSION

In progressive image transmission proposed [1-24], the image data is first reorganized and then transmitted progressively, where the reorganization is either in the spatial domain or in transform domain. In the spatial domain, progressive image transmission can take place either directly on the image pixels, such as in the bit plane and subsampling, or on the preprocessed data, such as in the pyramid technique [1-11]. The transform domain techniques can be classified into coefficient scanning [12,13], bit slicing [14,15] and S-transform [100,101]. These progressive image transmission techniques are now reviewed below.

3.1 DIRECT SPATIAL DOMAIN TECHNIQUES

A digital image can be divided into a set of bit planes as shown in Fig. 2.3.2, where the ith bit plane corresponds to the ith significant bits of the pixels in the original image. Progressive image transmission can be achieved by successively sending the bit planes starting with the most significant bit plane. The sets of possible reconstruction levels for successive approximations are, therefore, (0,128), (0,64,128,192), ... (0,1,...255) for an 8-bit image. This method is simple and needs no overhead, but, its performance is very poor. For example, the first approximation transmitted at a average bit rate of 1 bits/pixel is only a binary image.

Another simple method is to successively subsample the image pixels and progressively transmit them. At the receiving end, the missing pixels are defined by some interpolation scheme. The simplest approach is to repeat the received pixel values (zero-order interpolation). However, this method does not yield good intermediate approximations. More complex interpolation schemes have been proposed, including bicubic splines, cubic convolution and bilinear interpolation [6].

3.2 PYRAMIDAL TECHNIQUES

Pyramid data structures are a class of hierarchical representations of images.

This data structure has found an important role in the fields of image processing, computer graphics, geographic information systems, etc: Given an image X_n of size $2^n \times 2^n$, a pyramid can be defined as a sequence of matrices $\{X_k\}$ such that X_{k-1} is a reduced-resolution version of X_k. In the limit, X_0 is a single pixel (Fig. 3.1.1). For progressive image transmission, the pyramid levels $\{X_k\}$ should correspond to the approximations of the original image. Progressive image transmission is achieved by transmitting the pyramid starting from the top level. The top level can be used as the initial approximation and refined progressively by adding the information in the lower levels. Progressive image transmission techniques based upon the pyramid data structure are now reviewed.

Quadtree

Given an image of size $2^n \times 2^n$, a pyramid data structure can be formed by successively operating over 2×2 neighbouring pixels [1-3]. In other words, the value of a node at level k is a function of the values of 2×2 neighbouring nodes in level $k+1$. The value can be any function of the nodes in 2×2 window, such as Min, Max, Mean, Median, Mode, Sum, etc. [3]. As the nodes at each level $k-1$ are only associated with their four siblings at the lower level k, this pyramid data structure is also referred to as quadtree. A typical example is the mean quadtree [1-3], formed by (Fig. 3.1.2),

$$X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} = \frac{X_{k,i,j} + X_{k,i,j+1} + X_{k,i+1,j} + X_{k,i+1,j+1}}{4}, \qquad i,j = 0,1,...2^k - 1, \tag{3.1.1}$$

where $[\alpha]$ is the truncation of $\alpha + 0.5$. In this case, the top level of the quadtree is the mean of the original image and the intermediate levels of the quadtree correspond to the reduced-resolution approximations of the original image; for example, level $n-1$ corresponds to an image with $1/2$ the original resolution. However, there are two problems with this data structure. First of all, averages may lead to computational problems as more and more significant bits are needed to represent exactly the

averages. Secondly, since the number of nodes at level k is

$$\frac{2^n \times 2^n}{2^{n-k} \times 2^{n-k}}, = 2^k \times 2^k \qquad k = n, n-1, ...0, \tag{3.1.2}$$

the total number of nodes on the pyramid structure is equal to

$$\begin{aligned} \sum_{k=0}^{n} 2^k \times 2^k &= \frac{1 - 2^{n+1} \times 2^{n+1}}{1 - 2 \times 2} \\ &= \frac{4}{3}(2^n \times 2^n) - \frac{1}{3} \\ &\approx \frac{4}{3}(2^n \times 2^n). \end{aligned} \tag{3.1.3}$$

This implies that the number of nodes is about 33.3% more than the number of pixels in the original image. Thus in its primitive form, progressive image transmission using a quadtree structure results in a data expansion. The reason for the extra transmission time is that the information previously sent is not efficiently utilized in reconstructing successive approximations. In other words, redundancy or overhead is introduced during progressive transmission.

A number of methods have been proposed to improve performance. One method is to specify each node by a quadruple, namely, node value, spatial coordinates, and level number, and only transmit the quadruples of the nodes which differ from their predecessor [3]. In another method, only three of the sibling values are transmitted and the value of the fourth sibling is deduced from the relation between the sibling and parent nodes [3]. However, there still remains some redundant information associated with the transmitted nodes as there is a high degree of correlation between parent and sibling nodes.

Knowlton's Binary Tree

Knowlton [4] makes use of a binary tree version of a pyramid to progressively transmit grey-scale images. In essence, an image is repeatedly split into two halves alternating between the vertical and horizontal directions until the individual pixels

are reached, as shown in Fig. 3.1.3. Each pair of resulting pixels is expressed by two values using a look-up table (Fig. 3.1.4): the first is an approximate average of the two pixel values called a *composite value* (V_c) and the second is a *differentiator (d)* that enables the computation of the corresponding pixel values. For example (Fig. 3.1.4), a pair of pixel values, 3 and 4, is expressed by the composite value $V_c = 3$ and the differentiator $d = 7$. These two-pixel groups are recursively aggregated in groups of two to form a binary tree, as shown in Fig. 3.1.5. To achieve progressive transmission of the image, all that needs to be sent are the composite value of the top level and the successive sets of differentiators. It should be remarked that the coding is a 1:1 mapping that reformats an image into a data stream with the same number of bits. Increasing fractions of a complete transmission corresponds to a set of successive approximations which are refined progressively. Receipt of the entire bit stream permits an exact replication of the original image. Compression can be achieved by using a Huffman code to encode the differentiator values. Knowlton reported a 25% reduction in average bit rate for an 4-bit image. However, due to the limitation of the look-up table, the method is only suitable for low-resolution images. In Knowlton's example, each pixel was represented by four bits, hence the table was only 16 by 16. An extension of the Knowlton's method is presented in [5], which allows its application to images with any number of bits/pixel. In this method, the look-up table is replaced by a simple algorithm whose space complexity is independent of the number of bits/pixel.

Nonuniform Quadtree

In general, the information contained in an image is not uniformly distributed over the pixels; for example, there are regions which are uniform and other which are textured. Therefore, it is reasonable to transmit nonuniformly the pyramid data representing the image. In [7], a nonuniform decomposition technique is applied to the image to yield a set of nonuniform pyramids in quadtree form. A set of thresholds,

$\{P_i : P_i > P_{i+1}\}$, called *transmission pass,* is first given. The top left pixel of the image is taken as the root of the quadtree. The difference between the maximal and minimal pixel values of the image is compared with the first transmission pass, P_1. If the difference is less than P_1, no decomposition is necessary. If the difference is greater than P_1, the image is decomposed into four contiguous, nonoverlapping square subimages with equal size. The top left pixel values of the four subimages are taken as the node values at the second level of the pyramid. For each subimage, the difference between the maximal and minimal pixel values is once again compared with the first transmission pass, P_1. If the difference is less than P_1, no further decomposition. Otherwise, the subimage is further decomposed into four contiguous, nonoverlapping square subimages with equal size and the top left pixel values of the subimages are taken as the node values at the third level of the quadtree. The process is repeated until subimages with 1 pixel is reached. The quadtree obtained is nonuniform and corresponds to approximation of the image.

The same procedure can now be repeated for the other values in the transmission pass, P_2, P_3, ..., and a number of nonuniform quadtrees will be produced. Since the transmission pass with lower index is greater than that with higher index, i.e. $P_i > P_{i+1}$, the resolution of successive nonuniform quadtrees is improved. Furthermore, all the lower-index quadtrees are included in higher-index quadtrees.

Fig. 3.1.6 shows an example of the nonuniform decomposition for an image of size 8×8 with 4 bits/pixel, where Fig. 3.1.6a is the image, Fig. 3.1.6b is the nonuniform quadtree with a transmission pass of 8, and Fig. 3.1.6c is the nonuniform quadtree with a transmission pass of 4. Note that the second quadtree (Fig. 3.1.6c) has a higher resolution and includes all the node values of the first quadtree (Fig. 3.1.6b). Progressive image transmission is built up by transmitting the first nonuniform quadtree (to yield the first approximation of the image with a relatively low resolution) and then the nodes of the successive quadtrees which are different from the previous ones (to

improve the previous approximations). Note that the number of nodes transmitted is equal to the number of the pixels. To reduce the average bit rate, Huffman coding can be used to encode the node values of the set of nonuniform quadtrees. Since for each transmission pass, the approximation simply consists of the top left pixels of spatially nonoverlapping blocks with variable size (nonuniform), the scheme is in fact a nonuniform subsampling transmission technique.

The basic advantage of this method is that the detailed, or the higher information, areas appear first. An average reduction of 26.16% in the bit rate is reported for a sequence of test images of size 128×128 with 8 bits/pixel. However, the main problem is the overhead required for sending the information with regard to node selection for each quadtree, which is approximately 12% of the transmission bit rate. We note also that optimal pass parameter selection is still an open problem.

Filtered Pyramids

A more general method to form a pyramid data structure is proposed by Burt and Adelson [8]. In their method, the original image $X = G_0$ is first convolved with a Gaussian-like weighting function, producing a low-pass filtered image G_1. The low-pass filtered image G_1 is then subtracted from the original image G_0, giving a difference image L_0,

$$L_0 = G_0 - G_1. \tag{3.1.4}$$

Clearly, coding L_0 and G_1 is equivalent to directly coding the image itself, G_0. The following two observations should be noted:

1) As the predictor error L_0 is largely decorrelated, it may be represented pixel by pixel with fewer bits than G_0 [8].

2) As G_1 is low-pass filtered, it is of a lower bandwidth than the original image and so can be represented by fewer spatial samples. In Burt's example, the low-pass filtered image is subsampled by a factor of 2 in each direction.

By repeating the process, two sequences of two-dimensional images $G_0,\ G_1, \ldots$ and $L_0,\ L_1, \ldots$ are produced, referred to as the Gaussian and Laplacian pyramids, respectively, where each image is smaller than its predecessor by a factor 2 in each direction. The images of the Laplacian pyramid $L_0,\ L_1, \ldots$ are then quantized and transmitted in an inverse order of generation (i.e. $\ldots, L_1,\ L_0$). At the receiving end, each quantized error image is enlarged and added to the previous approximation. The basic steps involved in this scheme are summarized in Fig. 3.1.7. In the far left portion of the figure, the bottom-up construction of the Gaussian pyramid images, $G_0,\ G_1, \ldots$, is shown. The differences between successive Gaussian levels are taken as the Laplacian pyramid images $L_0,\ L_1, \ldots$, which are then quantized to yield a sequence of the compressed images represented by the pyramid of $C_l,\ l = 0, 1, \ldots$. Finally, the approximations are reconstructed by following an enlarge-and-sum procedure using the pyramid of C_l instead of $L_l,\ l = 0, 1, \ldots$. Note that in Burt and Adelson's method, all the computation must be done before the first approximation is sent.

A variant of the Burt and Adelson's method is proposed by W.D. Hofmann and D.E. Troxel [10]. Fig. 3.1.8 illustrates the block diagram of their method. By comparing Fig. 3.1.8 with Fig. 3.1.7, it can be observed that the structure used in [8] differs from the Burt and Adelson's in the following way. The successive predictive error images are generated by subtracting the previous approximations from the original image, instead of the next level of the pyramid (see Fig. 3.1.7). Thus, only the approximation being generated and the original image are required to be retained while in Burt and Adelson's, all the approximations and the original image have to be stored at the transmitter.

Another extension of the Burt and Adelson's method is to form a filtered pyramid by using a two dimensional quadrature mirror filter (QMF) [11]. To construct the 2-D QMF, a 1-D QMF is applied in the vertical and horizontal direction, respectively. The 1-D QMF applied to each line of the image saves the low-band outputs in the

left half of the line and the high-band outputs in the right half. Similarly, the 1-D QMF on each column saves the low-band outputs in the upper half of the column and the high-band outputs in the lower half. Consequently, the image output of the 2-D QMF is divided into four regions labeled LL, HL, LH, and HH, corresponding to low-band and high-band signals in the vertical and the horizontal directions, respectively, as shown in Fig. 3.1.9. The upper-left quadrant image (low-filtered) is referred to as the Gaussian image and the remaining three quadrants as the Laplacian images [11]. If the 2-D QMF is recursively applied to the low-pass filtered quadrant images (Gaussian), a filtered pyramid is obtained (Fig. 3.1.9). Each level consists of one Gaussian (LL) and three Laplacian (LH, HL, HH) images. The set of the low filtered quadrants forms the Gaussian pyramid and the sets of the other three quadrants form three Laplacian pyramids, respectively. At each level, the Gaussian and Laplacian quadrants can be combined to reconstruct the Gaussian quadrant at the next lower level. Therefore, only the Gaussian image at the top level and the Laplacian images need be transmitted to reproduce the original image.

To encode the Laplacian images, run-length coding and vector quantization are used [11]. All the pixels of the Laplacian images are first compared with a given threshold: if the pixel value is less than the threshold, its value is set to zero, otherwise, there is no change. Each Laplacian image is then decomposed into a set of spatially nonoverlapping blocks (vectors). If all the pixels in a block or a vector are zero, the vector is called a zero-vector. The locations of the zero-vectors are encoded by using the run-length coding. The non-zero vectors are vector quantized by using the generalized Lloyd algorithm. Since the images at different levels have rather different characteristics, a separate codebook is designed for each level.

3.3 TRANSFORM DOMAIN TECHNIQUES

In the transform coding mentioned in chapter 2.4, the image is transformed by an orthogonal transform. The image transformation can be considered as a decomposi-

tion of the image data into a generalized two dimensional spectrum. Each coefficient in the transform domain corresponds to the amount of energy of the *spectral* function within the image. Typically, the first (DC) coefficient represents the average brightness of the transform block. The AC coefficients indicate the amount of detail in the transform block. For most natural images, the energy is concentrated in the lower order coefficients. Therefore, a reasonable approximation of the image can be reconstructed by using only a few lower order coefficients and further refined by adding the higher order coefficients. This feature makes the orthogonal transform extremely attractive for progressive image transmission. A few progressive image transmission techniques have been proposed in the transform domain. They can be classified as, coefficient scanning [12,13], bit slicing [14,15] and S-transform [100,101].

Coefficient Scanning

In the coefficient scanning approach to progressive image transmission, the coefficients are transmitted in some order [12,13], usually from lower to higher order, because the lower order coefficients contain more amount of the image energy than the higher order coefficients. Clearly, this approach makes it possible to reconstruct rapidly a low-resolution approximation image by inverse transforming a few of the low order coefficients only. This low-resolution approximation is then improved progressively as more, and more, coefficients are received and decoded. However, it has to be decided in what order the transform coefficients are to be transmitted. One method [12] is to re-order all the AC coefficients in terms of the normalized mean square errors (NMSE) of their reconstructed images. The reconstructed image for each AC coefficient is formed by inverse transforming only the DC and the corresponding AC coefficient. Instead of using the NMSE of the coefficients as a measure for re-ordering, the coefficients can also be re-ordered according to their entropies [13]. Another method is to transmit the coefficients in a fixed order. A typical example of such a fixed order is the zig-zag scan of the transform block [12,13], as

shown in Fig. 3.2.1. Note that the energies of the coefficients along the zig-zag scan are basically monotonically decreasing. Examples of possible strategies [12,13] are shown in Fig. 3.2.2. Clearly, it is impossible for such fixed orders to achieve the optimal gradual improvements in the quality of progressively reconstructed images for all types of images. To achieve compression, the coefficients can be quantized before transmission.

Bit Slicing

Tzou and Elnahas [14] describe a progressive image transmission scheme where all the transform coefficients are scalar quantized in a number of stages by using a set of incremental bit assignment maps. Here, the total number of bits assigned to each coefficient is equal to the number of bits per pixel (in the spatial domain). An example of incremental bit assignment maps designed for the transform block of size 4×4 is shown in Fig. 3.2.3, where the incremental bit rate per stage is 1 bit/coefficient. The sequence of bits allocated to each coefficient can be read from the map at its corresponding location. For example, the bit assignment sequence is "51110000" for DC term and "31111100" for the AC term at (0,1). In fact, this scheme can be viewed as slicing the full bit assignment map (8 bits/coefficient in the Tzou and Elnahas's example) into layers of incremental bit assignment maps and sending a slice of information bits at each stage. There are two problems with this scheme. First of all, since the coefficients are eventually quantized at the same bit rate as the original image, no compression can be expected. Secondly, the total number of bits assigned to each coefficient are the same, i.e. equal bit allocation, which is not an optimal strategy in the transform domain. As demonstrated in chapter 2.4, for an optimal bit allocation, the number of bits assigned to each coefficient should be proportional to the coefficient variance.

S-Transform

The S-transform is a hierarchical approach to description of images, which has been used for progressive image transmission.[100,101]. An image to be transmitted is first decomposed into spatially nonoverlapping blocks of 2×2. Hadamard transform is then applied to the blocks of 2×2. The DC coefficients (the mean for each block) are assembled to form an approximation of the original image at 1/2 the original resolution. The same decomposition process is repeated on the lower resolution image to yield another approximation at 1/4 the resolution. The process is repeated until a single mean remains, corresponding to the overall mean of the original image. At the receiving end, a set of successive approximations is reconstructed in the reverse procedure of decomposition.

3.4 SUMMARY

We have reviewed progressive image transmission techniques: direct spatial domain techniques, pyramidal techniques and transform domain techniques.

The direct spatial domain techniques (bit plane and subsampling) are simple and need no overhead. However, these techniques cannot provide good intermediate approximations. Furthermore, no compression can be obtained for lossless reproduction.

In pyramidal techniques, the image is decomposed into a set of pyramid levels, either in tree form or in filtered form. Progressive image transmission is achieved by sending the set of levels starting from the top level. In general, the number of nodes of the pyramids is greater than the number of pixels, implying the extra transmission. The pyramid tree data structures proposed by Knowlton [4] and Drizen [7] are, however, 1:1 mappings that reformat the image into a bit stream with the same number of bits. Moreover, Drizen's nonuniform quadtree results in the more detailed areas of the image appearing first. To reduce the transmission bit rate, Huffman coding has been used to encode the nodes of the pyramid trees. In the case of filtered pyramid, (scalar) quantization is usually applied to the pyramids.

Transform domain techniques can be classified into coefficient scanning, bit slicing and S-transform. In coefficient scanning, the coefficients are transmitted in some order. A low resolution approximation is reconstructed by using only a few lower order coefficients and refined progressively by including more and more higher order coefficients. In the bit slicing technique, the full bit assignment map in the transform domain (the same as in the spatial domain) is sliced into a set of bit layers and the set of bit layers is progressively transmitted. Since all the coefficients are eventually quantized at the same bit rate, this technique is not optimal in the transform domain. In S-transform, a pyramid data structure is formed by successively Hadamard transforming the means of the 2×2 neighbouring blocks.

In chapters 4-7, we present four new image coding techniques. All the four techniques can achieve lossless progressive transmission with compression. To progressively transmit the image, the image is processed in a multistage fashion. To obtain compression, efficient coding techniques, such as transform coding and vector quantization, are used at each stage. To achieve lossless reproduction of the image, the residual errors introduced at each stage k are reprocessed at the next stage $k+1$, and an entropy coder is used to encode the final error image.

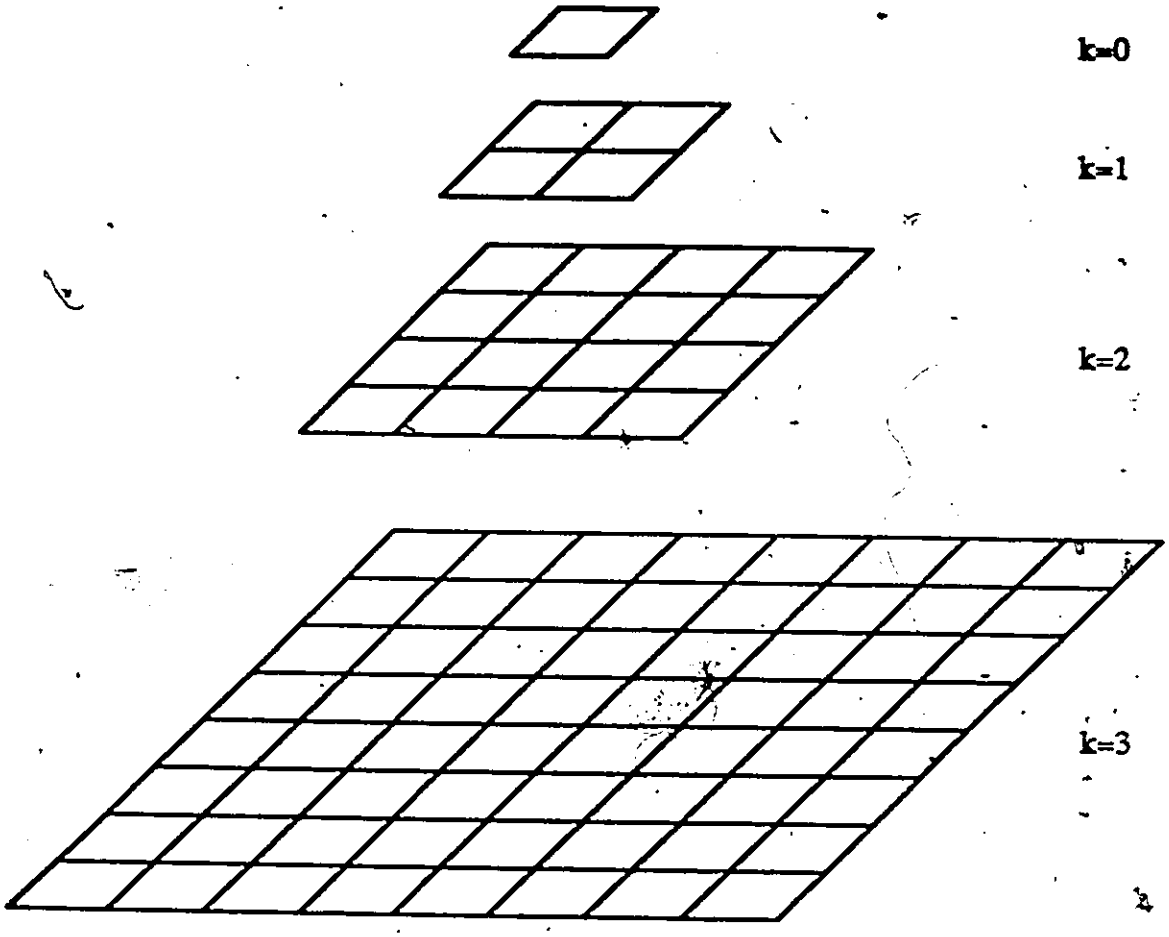


Fig. 3.1.1. An example of a pyramid structure with four levels.

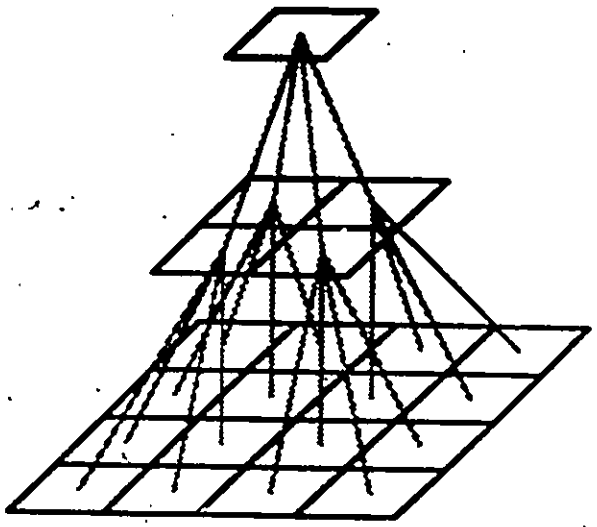


Fig. 3.1.2. An example of a quadtree representation of an image of size 4×4.

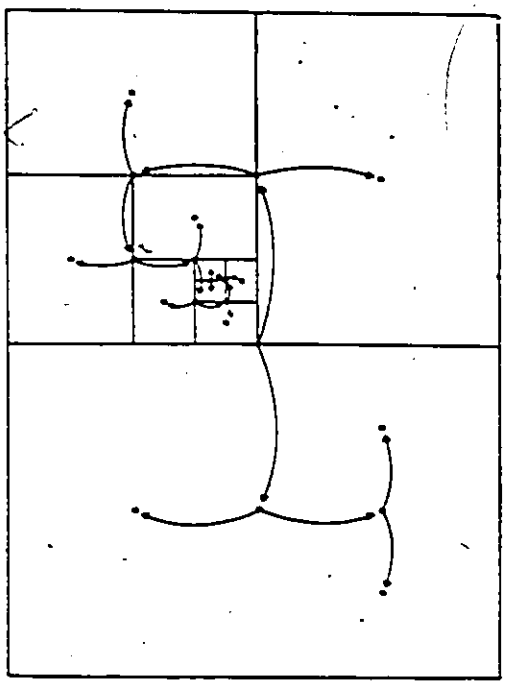


Fig. 3.1.3. Illustration of successive subdivision of the image (Ref. 4).

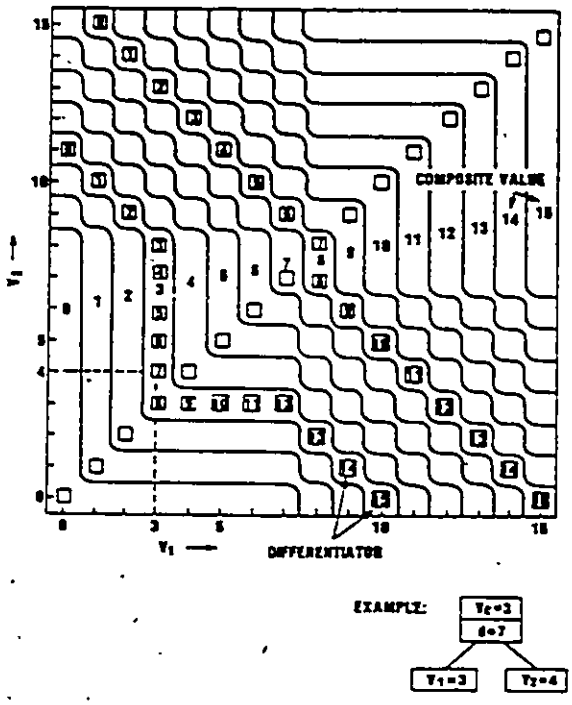


Fig. 3.1.4. A look-up table for determining composite value (V_c) and differentiator (d) from a pair of composite values at the lower level (Ref. 4).

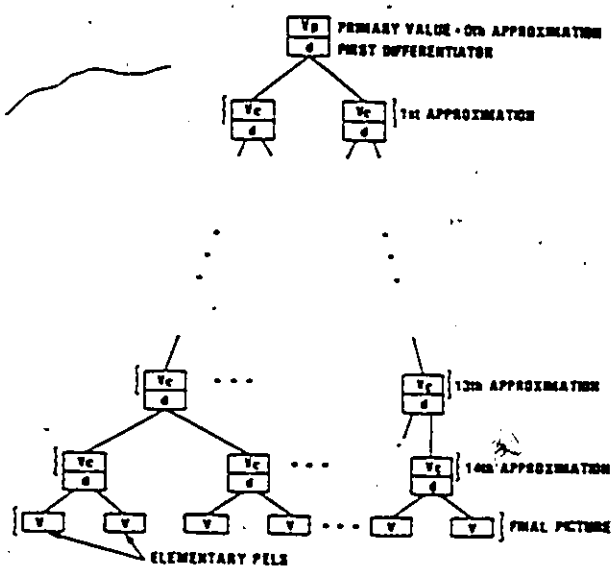


Fig. 3.1.5. Binary tree representation of the image where each node consists of two values: composite value and differentiator (Ref. 4).

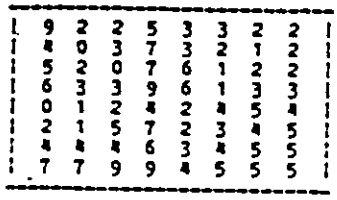


Fig. 3.1.6a. is the original image.

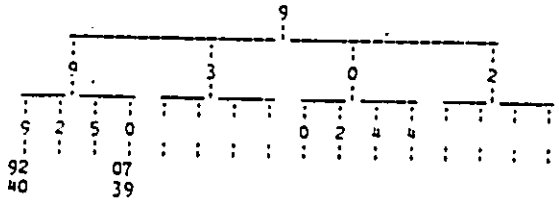


Fig. 3.1.6b. is the nonuniform quadtree with a transmission pass of 8.

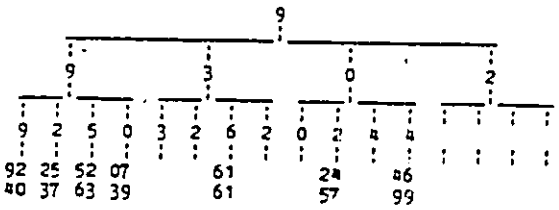


Fig. 3.1.6c. is the nonuniform quadtree with a transmission pass of 4.

Fig. 3.1.6. An example of nonuniform decomposition of the image (Ref. 7).

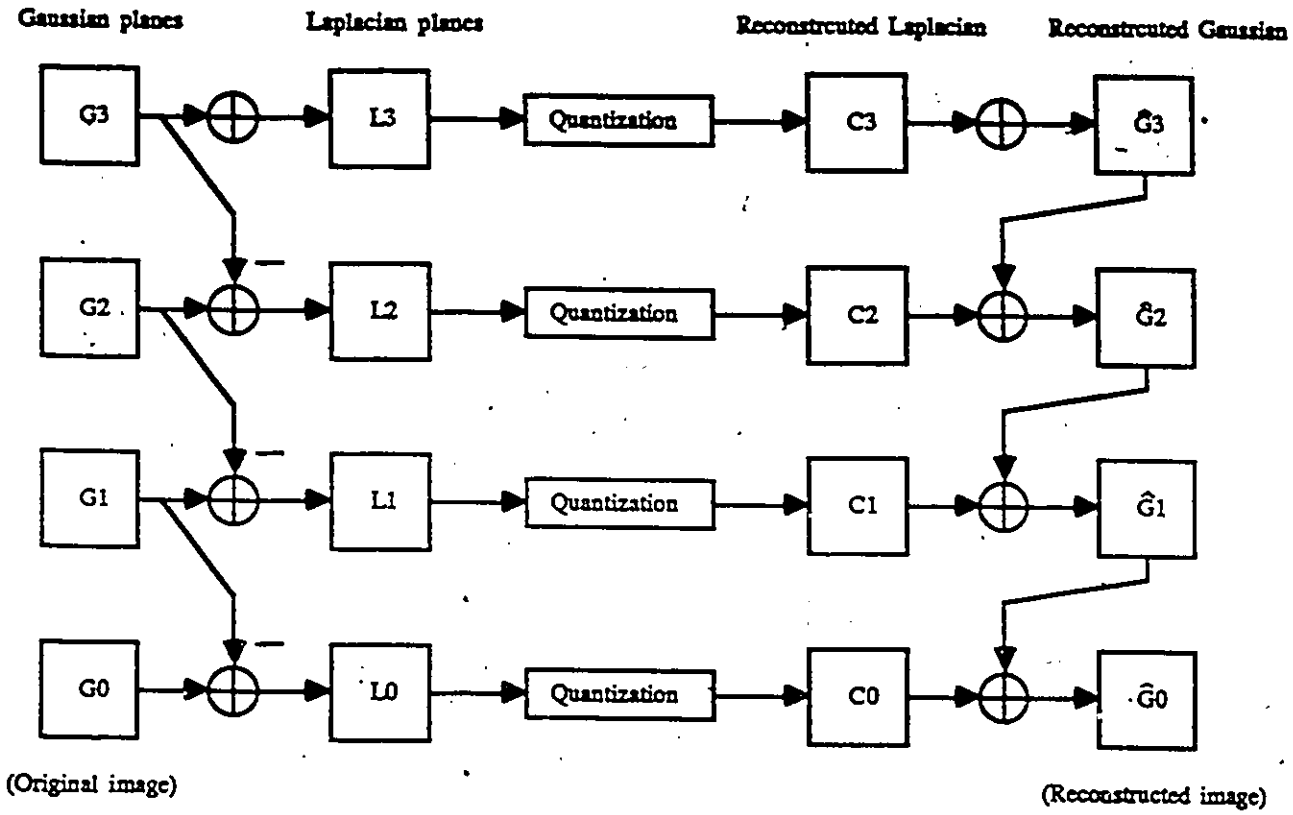


Fig. 3.1.7. A summary of the steps in Laplacian pyramid coding and decoding. A set of Gaussian pyramid levels $\{G_l\}$ is first generated from the original image G_0. A set of Laplacian pyramid levels $\{L_l\}$ is then obtained by computing the differences between successive Gaussian pyramid levels. The Laplacian pyramid is quantized to yield a sequence of compressed images of $\{C_l\}$. At the receiving end, the approximations are reconstructed by operating an enlarge-and-sum procedure on the pyramid of $\{C_l\}$.

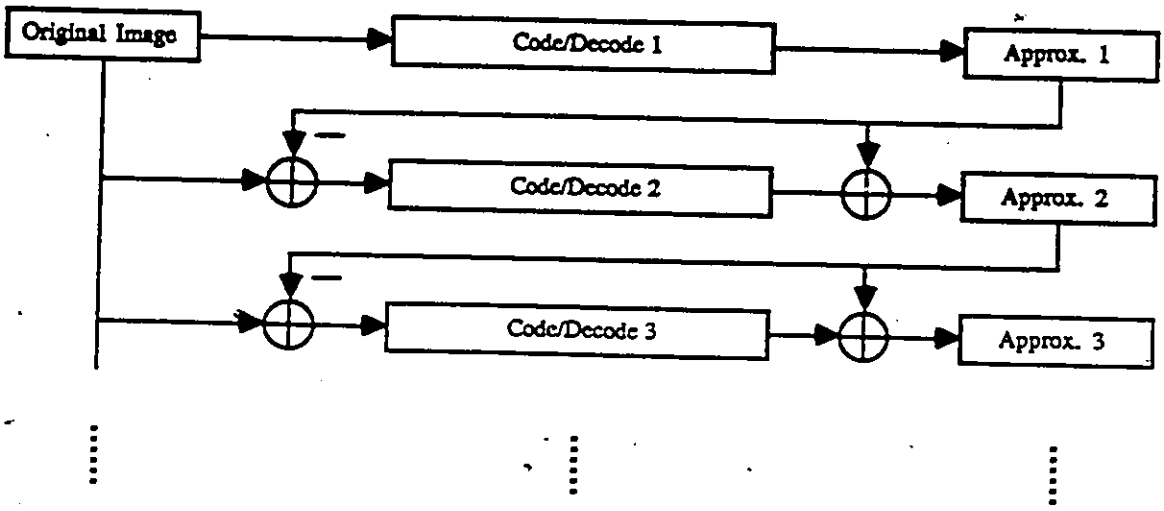


Fig. 3.1.8. Progressive transmission/receive structure.

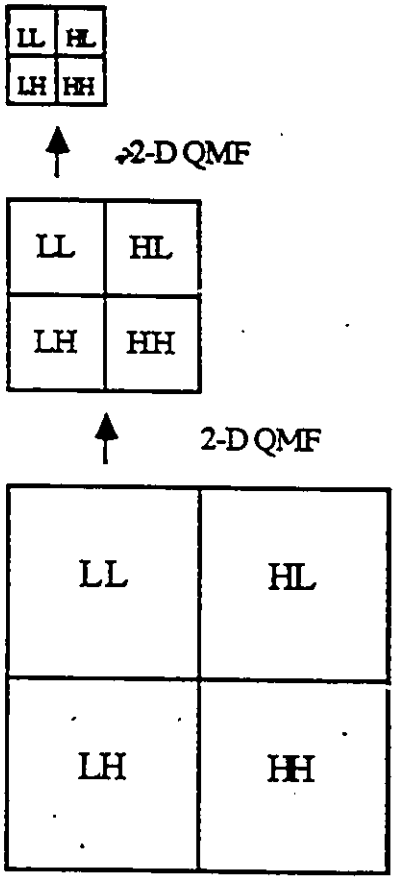


Fig. 3.1.9. Pyramid formation based upon 2-D quadrature mirror filters.

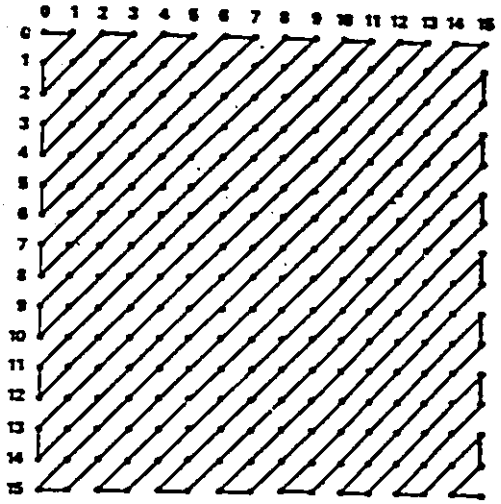
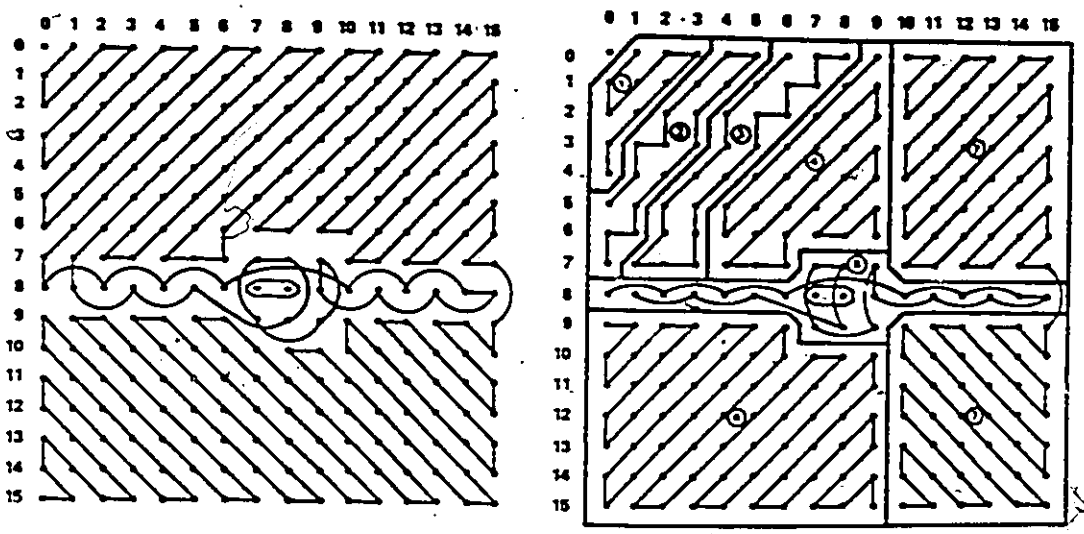


Fig. 3.2.1. The zig-zag scan (Ref. 13).



1	3	6	10	16	23	22	30
2	4	9	13	19	21	29	37
5	8	11	17	24	35	34	44
7	12	18	25	32	31	42	53
14	15	20	33	40	46	54	49
28	26	38	39	52	47	56	59
27	36	41	48	55	57	60	63
45	43	51	61	50	64	58	62

1	2	3	5	7	8	14	20
4	9	10	18	21	26	31	40
6	13	16	25	29	33	44	46
11	15	27	28	35	41	47	54
12	17	23	34	36	50	63	57
19	24	32	38	45	51	53	60
22	30	42	43	48	58	59	64
37	39	49	55	52	56	61	62

1	2	4	6	8	10	12	14
3	16	17	19	21	23	25	18
5	18	27	28	30	34	40	45
7	20	29	32	36	42	47	51
9	22	31	35	43	49	53	56
11	24	33	41	48	54	58	60
13	26	39	46	52	57	61	63
15	37	44	50	55	59	62	64

Fig. 3.2.2. Examples of possible scan orders (Ref. 12 and 13).

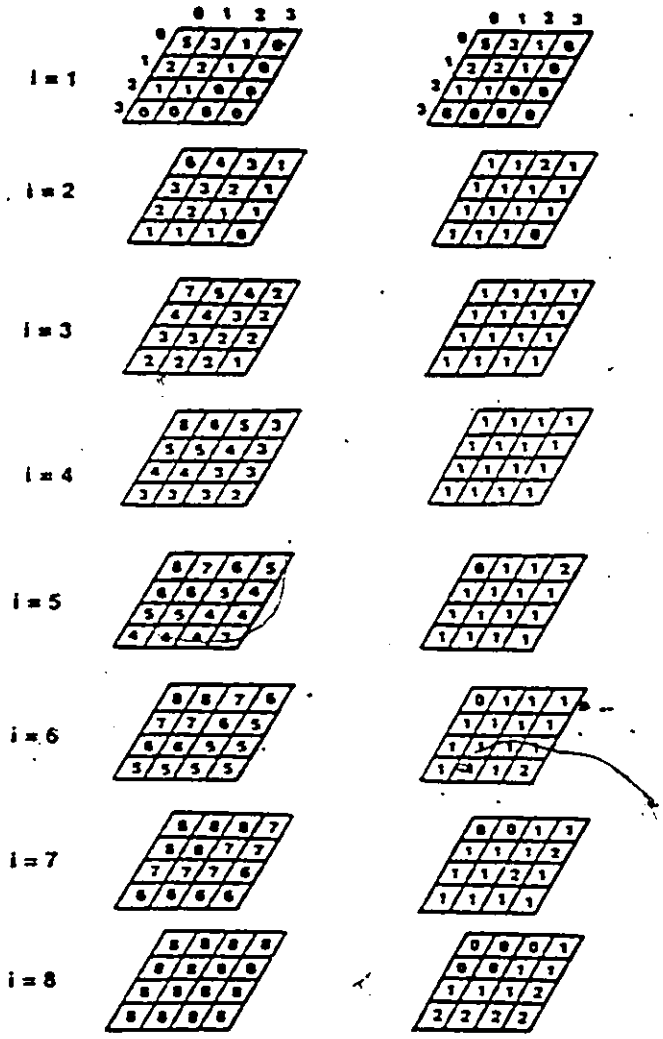


Fig. 3.2.3. Bit maps (left) and incremental bit maps (right) for a block size of 4×4 and an incremental rate of 1 bit/pixel (Ref. 14).

4. RESIDUAL ERROR VECTOR QUANTIZATION IN SPATIAL DOMAIN

This chapter presents a progressive image transmission scheme called *Residual Error Vector Quantization* (RVQ-SPA). In this scheme, the residual errors due to (vector) quantization are iteratively fedback and (vector) requantized. Coding takes place in the spatial domain and at each iterative stage, a fraction of the image information is transmitted and is utilized in reconstructing successive approximations to the original image. After a number of stages, an entropy coder, designed on a pixel-by-pixel basis, is used to losslessly encode the final residual error image.

In the following section, the details of RVQ-SPA are presented. Computer simulation results are reported in section 4.2. Single-stage and multi-stage coding are compared in section 4.3. This is followed by the conclusions in section 4.4.

4.1 RESIDUAL ERROR VECTOR QUANTIZATION

A progressive image transmission scheme based upon residual error vector quantization (RVQ-SPA) is now presented. The scheme has an iterative structure with an entropy coder being responsible for coding of the final residual error image, as shown in Fig. 4.1. The basic idea of this scheme is to apply iteratively vector quantization to the residual error images due to the previous quantization. At each stage, a fraction of the image information is transmitted and utilized in reconstructing successive approximations to the final image. If E_k and $\hat{E}_k$ represent, respectively, the input and output images of the coder at iterative stage k, the steps involved for each iterative stage are as follows (Fig. 4.1):

Vector Formation : The input image, E_k, $k = 0, 1, ...$ (original or residual error), is first decomposed into a set of vectors corresponding to spatially contiguous, nonoverlapping, blocks of 4×4 pixels;

Training Sequence Selection : All the vectors in the set are used to form the training sequence;

Codebook Generation : The codebook is generated by the generalized Lloyd clustering algorithm [65], where the normalized mean square error (NMSE) is used as the distortion measurement;

Codebook Encoding: The codewords are scalar quantized and encoded by a variable length coder based on the frequencies of occurrence of the values in the codebook;

Vector Quantization : Vector quantization involves finding the closest codeword in the codebook for each input vector; the corresponding label of the closest codeword is transmitted;

Feedback: The difference between the input and output images, $E_{k+1} = E_k - \hat{E}_k$, is then found, fedback and taken as the input image to the next stage;

Entropy Coding: A entropy coder, such as a Huffman coder [30-32], operating on a pixel-by-pixel basis is used to losslessly encode the final residual error image. The method of deciding when the system is to be switched to the entropy coder will be discussed below.

Decoding and Reconstruction: At the receiving end, the received labels are decoded by finding the corresponding codeword in the current codebook and the approximation, $\hat{X}_k$, is then reconstructed by adding the decoded information, $\hat{E}_k$, to the previous approximation, $\hat{X}_{k-1}$, i.e.

$$\begin{aligned} \hat{X}_k &= \hat{X}_{k-1} + \hat{E}_k \\ &= \hat{X}_{k-2} + \hat{E}_{k-1} + \hat{E}_k \\ &= \sum_{m=0}^{k} \hat{E}_m, \qquad k = 0, 1, \ldots \end{aligned} \tag{4.1.1}$$

which is the sum of the received information up to stage k.

We now analyze the performance of the system. From Fig. 4.1, the difference between the original image and the kth approximation is equal to the residual error

image after quantization at stage k (the difference between the input and output images at stage k), i.e.

$$\begin{aligned} X - \hat{X}_k &= X - \sum_{m=0}^{k} \hat{E}_m \\ &= E_0 - \hat{E}_0 - \hat{E}_1 - ... - \hat{E}_k \\ &= E_1 - \hat{E}_1 - ...\hat{E}_k \\ &= E_{k-1} - \hat{E}_{k-1} - \hat{E}_k \\ &= E_k - \hat{E}_k \\ &= E_{k+1}, \qquad k = 0, 1, ... \end{aligned} \tag{4.1.2}$$

where the 0'th input, E_0, is the original image, X. If a well-designed coder is assumed for each stage, the variance of the residual error image, $E_{k+1} = E_k - \hat{E}_k$ $(= X - \hat{X}_k)$, should be less than that of the input to the coder, E_k; that is,

$$\begin{aligned} \sigma_{k+1}^2 &= \mathrm{tr}(E(E'_{k+1}E_{k+1})) \\ &= \mathrm{tr}(E((E_k - \hat{E}_k)'(E_k - \hat{E}_k))) \\ &< \sigma_k^2 \\ &= \mathrm{tr}(E(E'_k E_k)). \qquad k = 0, 1, ... \end{aligned} \tag{4.1.3}$$

This implies that the energies of the residual error images keep decreasing with the number of iterations,

$$\sigma_x^2 = \sigma_0^2 > \sigma_1^2 > \sigma_2^2 > ... > \sigma_k^2 > ... \tag{4.1.4}$$

where $\sigma_x^2 = \sigma_0^2 = \mathrm{tr}(E(X'X))$ is the energy of the original image. Therefore, if the coding factor for each iterative stage is defined as,

$$c_m = \frac{\sigma_{m+1}^2}{\sigma_m^2} < c < 1, \qquad m = 0, 1, ... \tag{4.1.5}$$

then, the normalized mean square error (NMSE) of the approximation at stage k can be expressed as follows,

$$\mathrm{NMSE}(k) = \frac{\sigma_{k+1}^2}{\sigma_x^2} = \prod_{m=0}^{k} c_m, \qquad k = 0, 1, ... \tag{4.1.6}$$

It is clear that the NMSE as a function of the number of iterations is a decreasing function, that is,

$$\mathrm{NMSE}(k) < \mathrm{NMSE}(k-1) < \ldots < \mathrm{NMSE}(0), \tag{4.1.7}$$

or,

$$\prod_{m=0}^{k} c_m < \prod_{m=0}^{k-1} c_m < \ldots < \prod_{m=0}^{0} c_m. \tag{4.1.8}$$

Therefore, as the number of stages, k, approaches infinity, the NMSE converges to zero,

$$\begin{aligned} \mathrm{NMSE}(\infty) &= \lim_{k\to\infty} \prod_{m=0}^{k} c_m \\ &< \lim_{k\to\infty} \prod_{m=0}^{k} c \\ &= \lim_{k\to\infty} c^{k+1} \\ &= 0. \end{aligned} \tag{4.1.9}$$

A basic question is, when should the system be switched to the entropy coding? In order to answer this question, we now define a lossless coding rate at stage k, $R_t(k)$, as follows,

$$R_t(k) = R_p(k) + H_e(k), \tag{4.1.10}$$

where $R_p(k)$ is the total progressive bit rate required for transmission up to stage k, and $H_e(k)$ is the entropy of the residual error image at stage k. In other words, $R_t(k)$ is the total bit rate required for losslessly encoding the image using k stages of residual error vector quantization followed by an entropy coder. Theoretically, the system could be switched to the entropy coder at any time if the following two conditions hold:

1) No coding redundancy is introduced at any stage, implying that the bit rate required for transmission at stage m, ΔR_m, is exactly equal to the

image information transmitted at stage m, ΔH_m^t,

$$\Delta R_m = \Delta H_m^t, \qquad m = 0, 1, ... \tag{4.1.11}$$

2) The entropy coder can losslessly encode the residual errors (still correlated), at stage k, at a bit rate of $H_e(k)$.

The reason for this is that the total bit rate for losslessly encoding the image should in fact be a constant equal to the entropy of image, H,

$$R_t(k) = R_p(k) + H_e(k) = \sum_{m=0}^{k} \Delta R_m + H_e(k) = \sum_{m=0}^{k} \Delta H_m^t + H_e(k) = H^t(k) + H_e(k) \tag{4.1.12}$$

where $H^t(k)$ is the total progressive information transmitted.

Since an entropy coder operating on a pixel-by-pixel basis is proposed at the final stage, equation (4.1.10) is modified as follows,

$$R_t(k) = R_p(k) + H_e^1(k) \tag{4.1.13}$$

where $H_e(k)$ is replaced by $H_e^1(k)$ - the first order entropy of the residual error image at stage k. In the practical case, $R_t(k)$ is no longer a constant equal to the entropy of image because of the following two reasons:

1) Practical lossy coding systems (such as the vector quantization in Fig. 4.1) in general introduce some redundancy, and equation (4.1.11) has to be corrected as follows,

$$\Delta R_m = \Delta H_m^t + \Delta R_m^r, \qquad m = 0, 1, ... \tag{4.1.14}$$

where ΔR_m^r, is coding redundancy introduced at stage m. Therefore, the total progressive bit rate up to stage k, $R_p(k)$, will be larger than the total progressive information transmitted up to stage k, $H^t(k)$, due to the accumulated coding redundancy up to stage k, $R^r(k)$,

$$R_p(k) = \sum_{m=0}^{k} \Delta R_m = \sum_{m=0}^{k} \Delta H_m^t(k) + \sum_{m=0}^{k} \Delta R_m^r = H^t(k) + R^r(k) > H^t(k). \tag{4.1.15}$$

2) Since statistical dependency might still exists in the residual error image, the first order entropy of the residual error image, $H_e^1(k)$, is larger than, or equal to, the real entropy, $H_e(k)$,

$$H_e^1(k) = H_e(k) + H_e^r(k) \geq H_e(k), \tag{4.1.16}$$

where $H_e^r(k)$ is due to any statistical dependency of the residual errors.

We note, since each stage partially removes the statistical dependency of the image, the first order entropy of residual error image will tend to the real entropy, that is, $H_e^r(k)$ will tend to zero as k approaches infinity.

With these comments in mind, the total bit rate for losslessly encoding the image can be, therefore, re-expressed as followed,

$$\begin{aligned} R_t(k) &= R_p(k) + H_e^1(k) \\ &= H^t(k) + R^r(k) + H_e(k) + H_e^r(k) \\ &= H^t(k) + H_e(k) + R^r(k) + H_e^r(k) \\ &= H + (R^r(k) + H_e^r(k)), \end{aligned} \tag{4.1.17}$$

where H is the entropy of image and the redundancy is due to non-ideal encoding and the statistical dependency of the residual errors. It is clear that since H is a constant, the curve of $R_t(k)$ is only controlled by the last two terms: $R^r(k)$ and $H_e^r(k)$. Note that $R^r(k)$ and $H_e^r(k)$ are, respectively, monotonic-increasing and monotonic-decreasing functions of the number of stages, and consequently also with regard to the total progressive bit rate, $R_p(k)$.

The immediate question that arises is, does there exist a minimum value of $R_t(k)$? In the following, we demonstrate that the curve of $R_t(k)$ with respect to the total progressive bit rate, $R_p(k)$, is concave. To do this, we only need prove that the derivate of $R_t(k)$ with respect to $R_p(k)$ is a monotonic increasing curve crossing zero, i.e. an increasing curve from negative to positive value. From equations (4.1.13) and

(4.1.16),

$$\begin{aligned} R_t(k) &= R_p(k) + H_e^1(k) \\ &= R_p(k) + (H_e(k) + H_e^r(k)) \\ &= R_p(k) + (H - H^t(k)) + H_e^r(k) \end{aligned} \tag{4.1.18}$$

where $H_e(k) = H - H^t(k)$. Taking the derivative,

$$\frac{dR_t(k)}{dR_p(k)} = 1 - \frac{dH^t(k)}{dR_p(k)} + \frac{dH_e^r(k)}{dR_p(k)}. \tag{4.1.19}$$

In discrete form,

$$\frac{\Delta R_{t,m}}{\Delta R_m} = 1 - \frac{\Delta H_m^t}{\Delta R_m} + \frac{\Delta H_{e,m}^r}{\Delta R_m}. \tag{4.1.20}$$

where $\Delta R_{t,m}$ and $\Delta H_{e,m}^r$ represent, respectively, the variations of $R_t(k)$ and $H_e^r(k)$ at stage $k = m$. Since the first term in (4.1.20) is just a constant equal to 1, we need only analyze the last two terms. The following observations and assumptions are made:

(I) From equation (4.1.14), $\frac{\Delta H_m^t}{\Delta R_m} = \frac{\Delta H_m^t}{\Delta H_m^t + \Delta R_m^r}$. Therefore, this term is positive and less than 1.

(II) Since each stage partially removes the statistical dependency of the residual errors, $H_e^r(k)$ is a decreasing function of $R_p(k)$ and therefore $\frac{\Delta H_{e,m}^r}{\Delta R_m}$ is negative.

(III) Since the residual errors tend to become more decorrelated, it is reasonable to assume that the same coding technique (vector quantization) should code the residual errors less efficiently. This means that $\Delta R_m^r(k)$ increases and, therefore, $\frac{\Delta H_m^t}{\Delta R_m} = \frac{\Delta H_m^t}{\Delta H_m^t + \Delta R_m^r}$ decreases; while at the same time, $|\frac{\Delta H_{e,m}^r}{\Delta R_m}|$ decreases.

(IV) As the number of stages, or the total progressive bit rate of $R_p(k)$, increases, the residual errors tend to be statistically independent. Therefore, $H_e^r(k)$ approaches zero and

$$\frac{\Delta H_{e,m}^r}{\Delta R_m} \to 0,$$

which implies that,

$$\frac{\Delta R_{t,m}}{\Delta R_m} \rightarrow 1 - \frac{\Delta H_m^t}{\Delta H_m^t + \Delta R_m^r} > 0, \qquad (4.1.21)$$

i.e. the derivative (equation 4.1.20) is positive at some point.

(V) In addition, it is reasonable to assume that the coding system is very efficient for the original (highly correlated) image such that,

$$\frac{\Delta R_{t,0}}{\Delta R_0} < 0, \qquad (4.1.22)$$

that is, the derivative (equation 4.1.20) is initially less than zero.

Based upon the above analysis, it can be concluded that

$$\frac{\Delta R_{t,m}}{\Delta R_m} = 1 - \frac{\Delta H_m^t}{\Delta R_m} + \frac{\Delta H_{c,m}^r}{\Delta R_m} = 1 - (\frac{\Delta H_m^t}{\Delta R_m} + |\frac{\Delta H_{c,m}^r}{\Delta R_m}|)$$

is an increasing function of $R_p(k)$ which goes from negative to positive value. Thus, we have demonstrated that $R_t(k)$ is a concave curve as a function of $\tilde{R}_p(k)$. Consequently, the system should be switched to the entropy coder when the minimum value of $R_t(k)$ is achieved.

4.2 SIMULATIONS

The following computer simulations are carried out on two test images of size 256 × 256 pixels with 8 bits/pixel. The first, shown in Fig. 4.2, is a face image with relatively low overall detail, except in some areas, such as the eyes and the hair. The second, shown in Fig. 4.3, is a boat image, characterized by fine detail, notably many sharp edges. The first order entropies estimated from the histograms of the grey levels for both images are, respectively, 7.5178 bits/pixel for the face image and 7.5395 bits/pixel for the boat image.

The proposed progressive image transmission scheme (RVQ-SPA) is applied to these two test images with the vector dimension = 4 × 4. At each stage, the number of codewords can be independently chosen. A larger number of codewords at one stage

implies that more bits are allocated to that stage. Two distributions for the number of codewords per stage are tested, uniform (the same number at each stage) and nonuniform (increasing number per stage), with the latter giving the better results. The simulation results are shown in Tables 4.1 and 4.2 with,

$N_c(k)$ – the number of codewords at stage k;

$R_l(k)$ – the bit rate for transmitting the labels at stage k;

$R_c(k)$ – the bit rate for transmitting the codebook (overhead) at stage k;

$R_p(k) = \sum_{m=1}^{k}(R_l(m) + R_c(m))$ – the total progressive bit rate up to stage k;

$H_e^1(k)$ – the first order entropy of the residual error image at stage k;

$R_t(k)$ – the total bit rate for losslessly coding at stage k.

In evaluating the performance level of this scheme, the normalized mean square error (NMSE) between the original image X and the kth approximation X_k is used, defined as,

$$\text{NMSE}(k) = \frac{\sum_{i,j=0}^{255}(X_{ij} - X_{k,ij})^2}{\sum_{ij=0}^{255} X_{ij}^2}, \qquad k = 0, 1, ... \tag{4.2.1}$$

where X_{ij} and $X_{k,ij}$ represent the (i,j)th element of the original image and its kth approximation, respectively.

From Tables 4.1a, 4.1b, 4.2a and 4.2b, it is seen that as the number of iterative stages, k, increases, the total progressive bit rate, $R_p(k)$, also increases whereas the entropy (first order) of the residual error image, $H_e^1(k)$, and the NMSE(k) decrease (4.1.7). This behaviour is as expected, since at each stage a portion of the image information is transmitted. In other words, the larger the number of iterative stages, the greater the total progressive bit rate and the information transmitted, which implies less residual information and, therefore, less distortion between the original and the approximation images. The relation between $R_p(k)$ and NMSE(k) is shown

in graphical form in Fig. 4.4 and 4.5 as rate distortion curves. Note the better performance obtained when a non-uniform distribution of the codewords per stage is used (Tables 4.1a and 4.2a). For example, at stage 5, there is, respectively, a 10.48% ($\frac{0.0992-0.0888}{0.0992} \times 100\%$) improvement for the face image and 11.10% ($\frac{0.3575-0.3178}{0.3575} \times 100\%$) for the boat image.

The successive approximations to the original images corresponding to the results in Tables 4.1a and 4.2a are shown in Fig. 4.6 and 4.7: the zeroth, first, second, third and fourth approximations followed by the original images. Note that although the first two approximations are very rough, they have indeed provided the main structural information of the original images at a very low bit rate of about 0.064 bits/pixel and 0.190 bits/pixel, respectively. The following successive approximations rapidly converge to subjectively good quality. For example, for the face image, the details around the eyes and the hair are significantly improved from the second to third approximation and there is no apparent perceptible impairment by the fourth approximation.

The total bit rate $R_t(k)$ versus the total progressive bit rate $R_p(k)$ from Tables 4.1 and 4.2 is graphically shown in Fig. 4.8 and 4.9. We find, as demonstrated above, that $R_t(k)$ is a concave function of $R_p(k)$. The curve of $R_t(k)$ approaches rapidly (within 5 stages) a minimum value and then very slowly increases. An intuitive explanation for this behaviour from equation (4.1.17) is that the residual errors are gradually decorrelated within 5 stages such that $H_e^r(k)$ is reduced to a relative small value, while the coding redundancy, $R^r(k)$, progressively accumulated becomes large. The slow increase of $R_t(k)$ after stage 5 is thus mainly due to the accumulated coding redundancy. Note that the minimum value of the total bit rate, $R_t(k)$, reached by the proposed scheme is, respectively, 4.9721 bits/pixel for the face image and 5.7316 bits/pixel for the boat image, much lower than the corresponding first order entropies of the images (7.5178 bits/pixel for the face image and 7.5395 bits/pixel for the boat).

In terms of lossless coding, the minimum value of the total bit rate corresponds to the lowest bit rate for lossless reproduction of the image. Therefore, the system should be switched to the entropy coder at the minimum point of the total bit rate, $R_t(k)$.

4.3 DISCUSSIONS ON SINGLE-STAGE AND MULTI-STAGE CODING

An interesting question to be asked is, how does multi-stage coding compare with single-stage coding. The comparisons are made in terms of coding efficiency and the computational cost.

Information and Bit Rate:

As mentioned above, an ideal coding system transmits only the net image information and introduces no redundancy, i.e. at each stage;

$$\Delta R_m = \Delta H_m^t, \qquad m = 0, 1, ... \tag{4.1.11}$$

Therefore, the total progressive image information transmitted up to stage k, $H^t(k)$, should be equal to the total progressive bit rate up to stage k, $R_p(k)$,

$$\sum_{m=0}^{k} \Delta H_m^t = \sum_{m=0}^{k} \Delta R_m \tag{4.3.1}$$

or

$$H^t(k) = R_p(k). \tag{4.3.2}$$

From equation (4.1.11), it is also evident that single-stage coding at a bit rate of $R_s = R_p(k)$ can only transmit the same amount of the image information,

$$H_s^t = R_s = R_p(k) = H^t(k). \tag{4.3.3}$$

This implies that the performances for multi-stage and single-stage coding are equivalent under the assumption of an ideal coding system.

However, a practical coding system always introduces a certain amount of redundancy. For the multi-stage case, the appropriate equation is equation (4.1.15),

$$R_p(k) = H^t(k) + R_r(k) = \sum_{m=0}^{k} \Delta H_m^t + \sum_{m=0}^{k} \Delta R_m^r \tag{4.1.15}$$

where the second term is due to the redundancy introduced by non-ideal coding. For the single-stage case, the appropriate equation is

$$R_s = H_s^t + R_s^r \tag{4.3.4}$$

where the second term is the redundancy introduced. Thus, it can be concluded that, for a fixed bit rate, the performances of single-stage and multi-stage coding depend upon the redundancies R_s^r and $R^r(k)$. For comparative purposes, the simulation results using single-stage vector quantization are reported in Tables 4.1c and 4.2c. The corresponding rate distortion and total bit rate curves are also illustrated in Fig. 4.4, 4.5, 4.8 and 4.9. As we can see, multi-stage coding consistently outperforms single-stage coding. For example, at a bit rate of about 0.94 bits/pixel, there is an improvement over single-stage coding, respectively, of 28% ($\frac{0.14-0.10}{0.14} \times 100\%$) for the face image and 30% ($\frac{0.52-0.36}{0.52} \times 100\%$) for the boat image. The superior performance of multi-stage coding can be probably traced to the lower overhead required to transmit the codebooks ($R_c(k)$): 8% for the multi-stage case and 50% for the single-stage case.

Computational Cost:

The computational cost can be defined as follows (2.5.9),

$$C = k_c \times N_v \times D_v \times N_c, \tag{4.3.5}$$

where N_v is the number of input vectors, D_v is the vector dimension, N_c is the number of codewords in codebook and k_c is a constant of proportionality. For multi-stage coding, the cost is equal to the sum of the computational costs at different stages, i.e.

$$C_m = k_c \times \sum_m (N_{v,m} \times D_{v,m} \times N_{c,m}), \tag{4.3.6}$$

where m is the index of the stage. We now illustrate the difference in computational cost between single-stage and multi-stage coding with the following example. Let the

image of $N \times N = 256 \times 256$ be vector coded ($D_v = 4 \times 4$) at a bit rate of about 1.0 bits/pixel, implying that $N_v = \frac{N \times N}{D_v} = \frac{256 \times 256}{4 \times 4} = 4096$. From Tables 4.1c and 4.2c, 256 codewords ($N_c = 256$) are required in codebook at a bit rate of about 0.94 bits/pixel. The computational cost is proportional to

$$N_v \times D_v \times N_c = 4096 \times 16 \times 256 = 1.6777 \times 10^7$$

For the multi-stage case, there are 6 stages (Tables 4.1a, 4.1b, 4.2a and 4.2b). The corresponding total computational costs for uniform and non-uniform distributions of the number of codewords per stage are, respectively, proportional to,

$$\begin{aligned}\sum_{m=0}^{5}(N_{v,m} \times D_{v,m} \times N_{c,m}) &= N_v \times D_v \times \sum_{m=0}^{5} N_{c,m} \\ &= 4096 \times 16 \times (6 \times 8) \\ &= 3.1457 \times 10^6\end{aligned}$$

and

$$\begin{aligned}\sum_{m=0}^{5}(N_{v,m} \times D_{v,m} \times N_{c,m}) &= N_v \times D_v \times \sum_{m=0}^{5} N_{c,m} \\ &= 4096 \times 16 \times (2+4+8+3 \times 16) \\ &= 4.0632 \times 10^6\end{aligned}$$

As seen, there is a saving in computational cost of $75-81\%$ over single-stage at a bit rate of about 1.0 bits/pixel.

In summary, multi-stage vector quantization can achieve a performance level no worse than single-stage vector quantization under the assumption of an ideal coding system while the computational cost of multi-stage is much less than that of single-stage. It should be emphasized that single-stage vector quantization assigns a larger amount of the bit rate to transmit the codebook as overhead, compared to the multi-stage case.

4.4 CONCLUSIONS

In this chapter, a progressive image transmission scheme (RVQ-SPA) has been presented. The computer simulation results demonstrate that the proposed scheme achieves lossless progressive image transmission with compression. Here, the concept of iteratively coding the residual error images makes it possible to transmit the image progressively. Compression is achieved by applying vector quantization to each stage. The essential features of the images are captured at bit rates as low as 0.064-0.190 bits/pixel and excellent reproduction is achieved between 0.58-0.81 bits/pixel. Lossless coding is guaranteed by an entropy coder at the final stage. As a lossless coder, this scheme is superior to an entropy coder operating on the histogram of grey values of the image, by about 2.5457 (7.5178-4.9721) bits/pixel for the face image and 1.8079 (7.5395-5.7316) bits/pixel for the boat image.

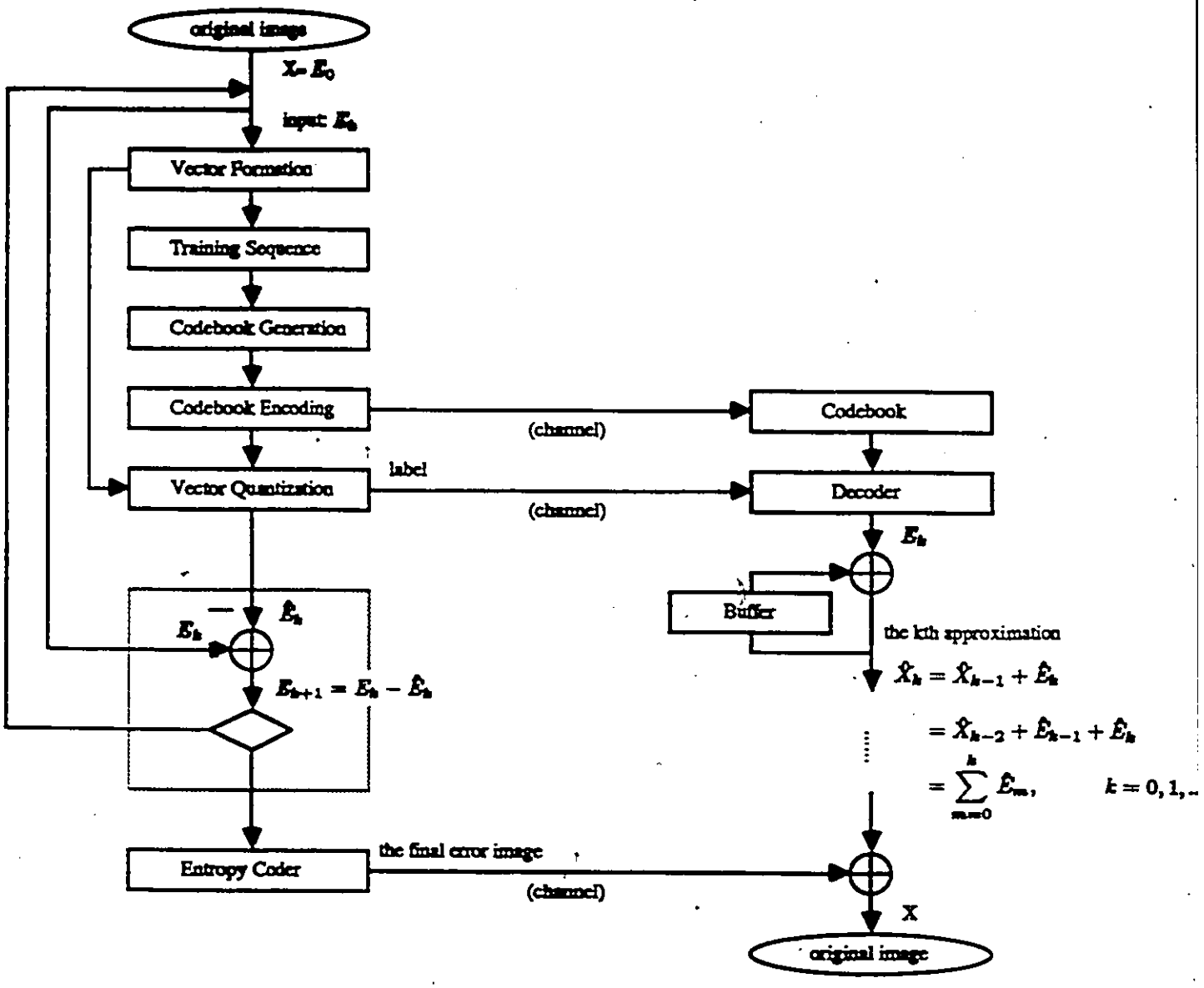


Fig. 4.1. Residual error vector quantization in the spatial domain. The image first undergoes a vector quantization. The residual errors due to quantization is iteratively fedback and requantized. At each iterative stage, a fraction of the image information is transmitted and utilized in reconstructing successive approximations of the image. Finally, an entropy coder is used to losslessly encode the final residual error image.

Fig. 4.2. The original face image of 256 × 256 pixels with 8 bits/pixel. The first entropy of the face image estimated from the histogram of grey levels is 7.5178 bits/pixel.

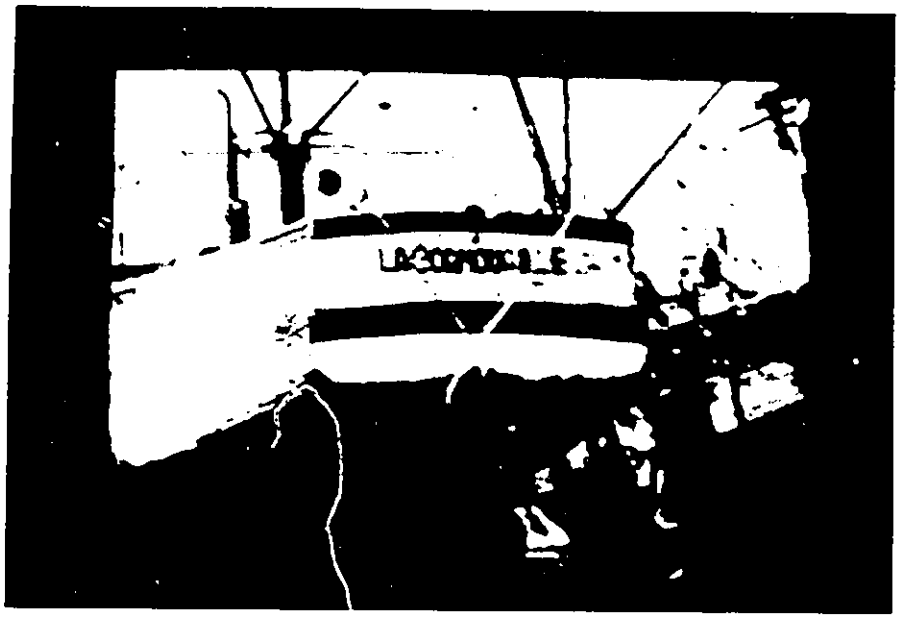


Fig. 4.3. The original boat image of 256 x 256 pixels with 8 bits/pixel. The first entropy of the boat image estimated from the histogram of grey levels is 7.5395 bits/pixel.

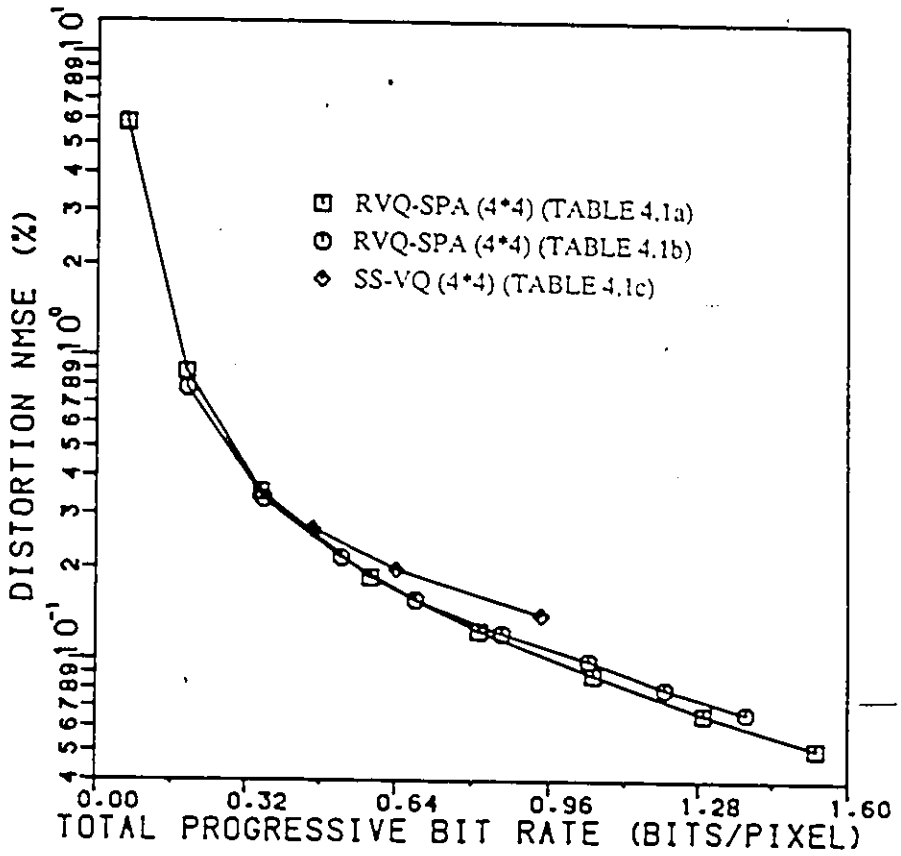


Fig. 4.4. The rate distortion curves obtained by using RVQ-SPA with a block size of 4×4 on the face image (from Table 4.1). It is clear that RVQ-SPA outperforms a single-stage vector quantization.

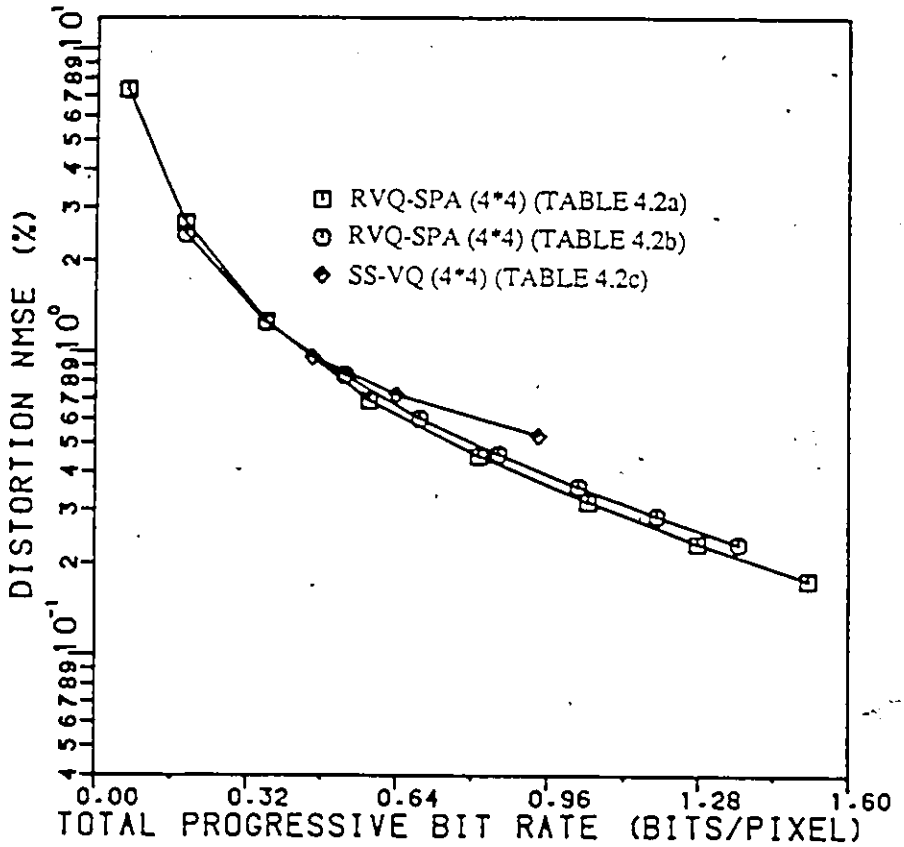


Fig. 4.5. The rate distortion curves obtained by using RVQ-SPA with a block size of 4×4 on the boat image (from Table 4.2). It is clear that RVQ-SPA outperforms a single-stage vector quantization.

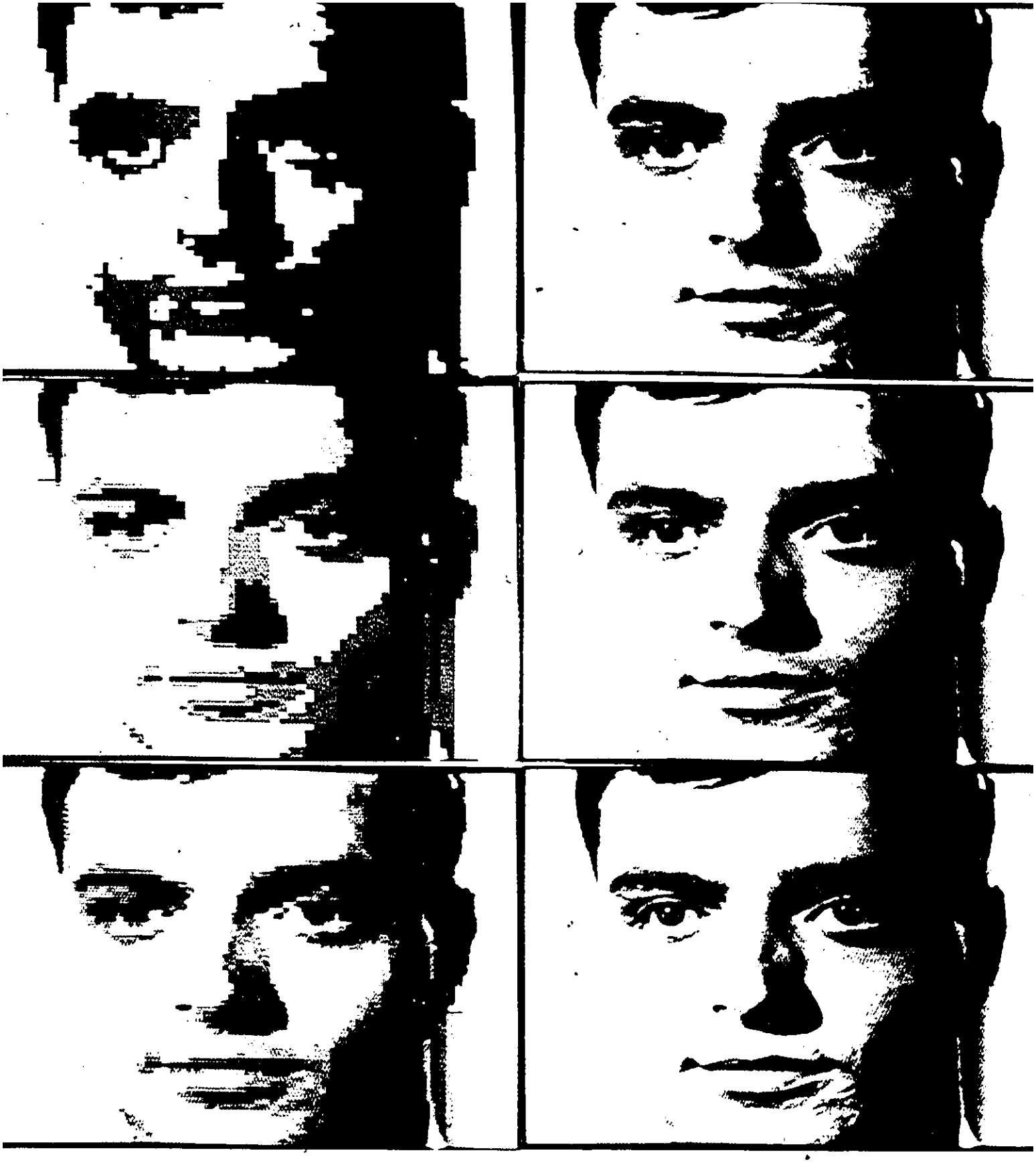


Fig. 4.6. Pictorial results for the face image (from Table 4.1a): the zeroth, first, second, third, and fourth followed by the original face image.

Fig. 4.7. Pictorial results for the boat image (from Table 4.2a): the zeroth, first, second, third, and fourth followed by the original boat image.

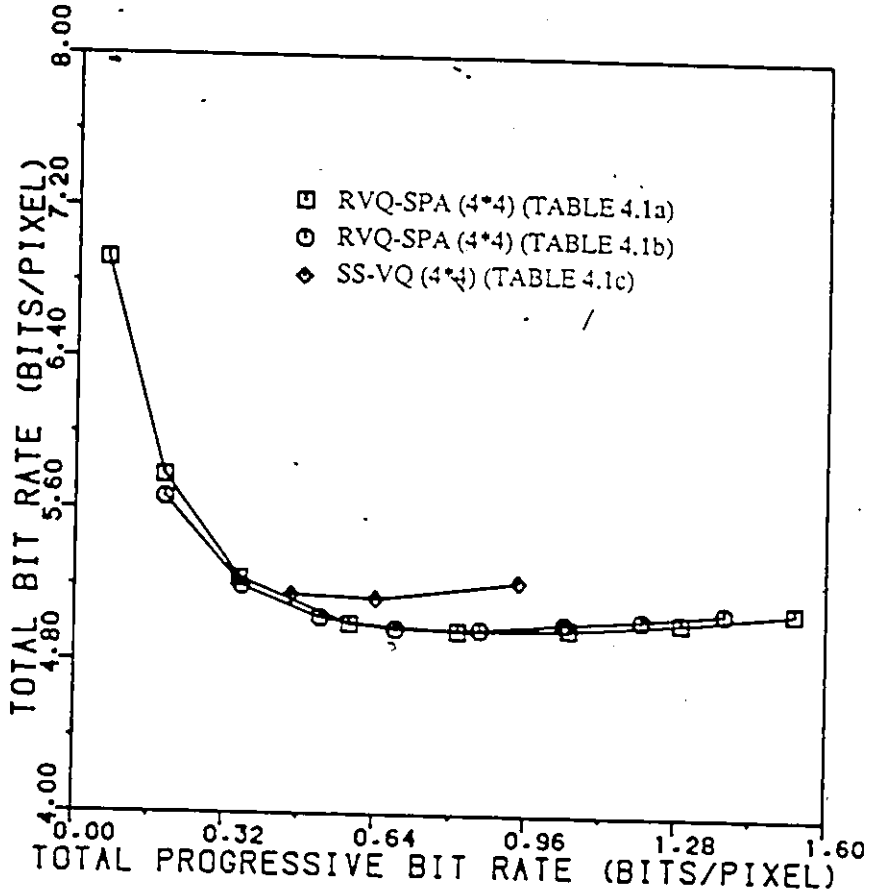


Fig. 4.8. Graph of the total bit rate $R_t(k)$ versus the total progressive bit rate $R_p(k)$ for the face image (from Table 4.1).

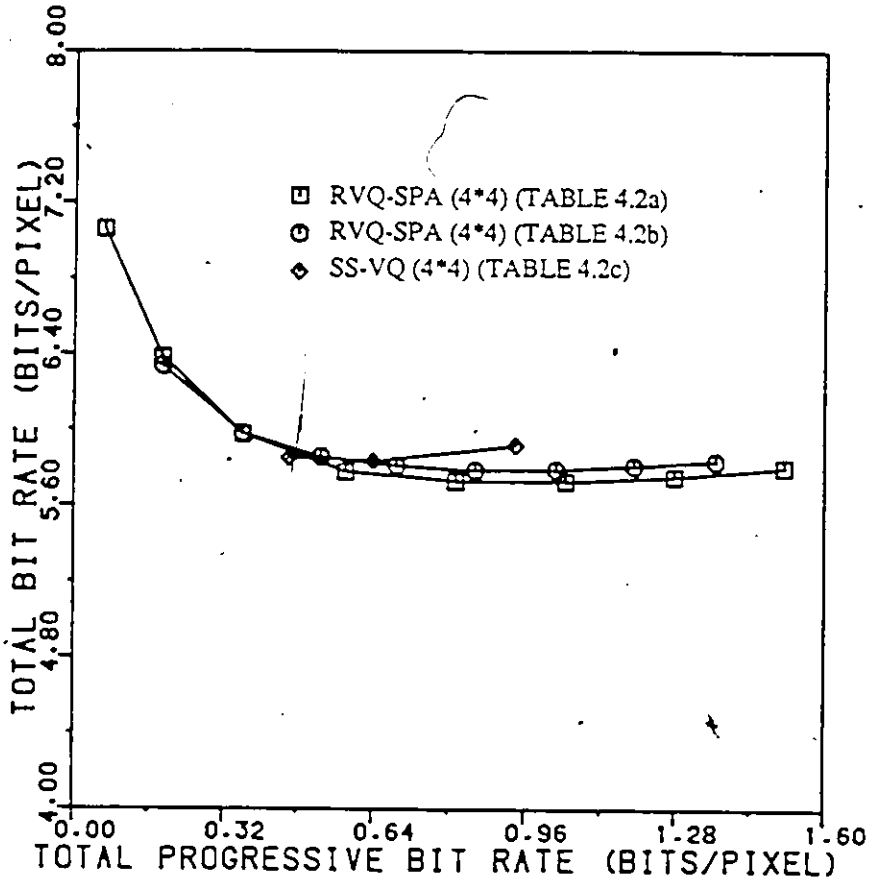


Fig. 4.9. Graph of the total bit rate $R_t(k)$ versus the total progressive bit rate $R_p(k)$ for the boat image (from Table 4.2).

Table 4.1a Computer simulation results for the face image by using RVQ-SPA with a block size of 4*4

Stage k	Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)	Rp(k) (bits/pixel)	Hc(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	2	0.0620	0.0011	0.0631	6.8575	6.9206	5.8241
1	4	0.1232	0.0034	0.1899	5.5888	5.7787	0.8807
2	8	0.1521	0.0105	0.3526	4.8850	5.2376	0.3545
3	16	0.2132	0.0200	0.5859	4.4182	5.0042	0.1860
4	16	0.2106	0.0180	0.8146	4.1575	4.9721	0.1240
5	16	0.2213	0.0150	1.0518	3.9343	4.9861	0.0888
6	16	0.2250	0.0145	1.2914	3.7362	5.0276	0.0661
7	16	0.2294	0.0125	1.5334	3.5578	5.0912	0.0512

Table 4.1b Computer simulation results for the face image by using RVQ-SPA with a block size of 4*4

Stage k	Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)	Rp(k) (bits/pixel)	Hc(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	8	0.1833	0.0089	0.1922	5.4717	5.6639	0.7772
1	8	0.1559	0.0104	0.3586	4.8420	5.2007	0.3369
2	8	0.1567	0.0090	0.5244	4.5141	5.0385	0.2168
3	8	0.1498	0.0085	0.6827	4.2989	4.9817	0.1572
4	8	0.1718	0.0075	0.8621	4.1190	4.9811	0.1216
5	8	0.1730	0.0065	1.0416	3.9714	5.0131	0.0992
6	8	0.1588	0.0070	1.2076	3.8308	5.0384	0.0803
7	8	0.1701	0.0060	1.3838	3.6979	5.0817	0.0670

Table 4.1c Computer simulation results for the face image by using a single-stage vector quantization with a block size of 4*4

Nc(k)	Rl (bits/pixel)	Rc (bits/pixel)	Rp (bits/pixel)	Hc (bits/pixel)	Rt (bits/pixel)	NMSE(%)
64	0.3470	0.1167	0.4637	4.6914	5.1552	0.2674
128	0.4025	0.2383	0.6409	4.4983	5.1392	0.1973
256	0.4608	0.4820	0.9429	4.2891	5.2320	0.1400

Table 4.2a Computer simulation results for the boat image by using RVQ-SPA with a block size of 4*4

Stage k	Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)	Rp(k) (bits/pixel)	Hc(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	2	0.0624	0.0015	0.0640	6.9967	7.0607	7.3013
1	4	0.1206	0.0038	0.1886	6.1958	6.3844	2.6615
2	8	0.1633	0.0105	0.3624	5.6143	5.9768	1.2540
3	16	0.1983	0.0219	0.5826	5.1976	5.7802	0.6870
4	16	0.2131	0.0195	0.8153	4.9162	5.7316	0.4505
5	16	0.2118	0.0181	1.0453	4.6871	5.7324	0.3178
6	16	0.2165	0.0166	1.2784	4.4781	5.7566	0.2322
7	16	0.2211	0.0155	1.5152	4.2900	5.8052	0.1760

Table 4.2b Computer simulation results for the boat image by using RVQ-SPA with a block size of 4*4

Stage k	Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)	Rp(k) (bits/pixel)	Hc(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	8	0.1782	0.0110	0.1893	6.1463	6.3357	2.4380
1	8	0.1610	0.0106	0.3610	5.6214	5.9825	1.2433
2	8	0.1591	0.0097	0.5300	5.3285	5.8586	0.8353
3	8	0.1519	0.0090	0.6910	5.1219	5.8129	0.6012
4	8	0.1562	0.0089	0.8561	4.9338	5.7900	0.4587
5	8	0.1603	0.0083	1.0248	4.7681	5.7929	0.3575
6	8	0.1582	0.0079	1.1909	4.6204	5.8114	0.2865
7	8	0.1678	0.0075	1.3663	4.4704	5.8367	0.2320

Table 4.2c Computer simulation results for the boat image by using a single-stage vector quantization with a block size of 4*4

Nc(k)	Rl (bits/pixel)	Rc (bits/pixel)	Rp (bits/pixel)	Hc (bits/pixel)	Rt (bits/pixel)	NMSE(%)
64	0.3447	0.1170	0.4617	5.3916	5.8533	0.9569
128	0.4019	0.2384	0.6404	5.1970	5.8375	0.7196
256	0.4587	0.4812	0.9400	4.9813	5.9214	0.5273

5. RESIDUAL ERROR QUANTIZATION IN TRANSFORM DOMAIN

In this chapter, we extend the residual error quantization technique presented in chapter 4 to the transform domain. An image to be transmitted first undergoes a general orthogonal transform and the resulting coefficients are then quantized before transmission. The residual error array due to quantization is iteratively fedback and requantized. Results using both scalar (RSQ-TRA) and vector quantization (RVQ-TRA) are presented, with superior performance achieved by vector quantization, where the vectors are formed from the block cosine transform coefficients. It is shown that the average reconstruction error variance converges to zero, as the number of iterative stages approaches infinity, i.e. the proposed scheme is theoretically lossless. In practice, lossless reproduction can be achieved with a small number of iterations by using an entropy coder on the final residual error image. It is shown experimentally that the total bit rate for lossless reproduction rapidly approaches a lower value than the first order entropy of the original image.

In section 5.1, residual error scalar quantization with optimum bit allocation is presented. Section 5.2 shows that better system performance levels can be achieved by iteratively applying vector quantization, instead of the optimum scalar quantization, to the transform coefficients. It is shown that, for both cases, the average reconstruction error variance converges to zero, as the number of iterative stages approaches infinity. The use of an entropy coder to encode the residual error image is described in section 5.3. Finally, computer simulation results are reported in section 5.4.

5.1 RESIDUAL ERROR SCALAR QUANTIZATION

Recall that the steps involved in transform coding (section 2.4.2) are *orthogonal transformation, quantization, and transmission.* It is clear that if the channel is noise-free, only the quantizer can introduce errors (quantization error), leading to information loss. Based on the idea of recoding the residual errors due to quantization,

a progressive image transmission scheme shown in Fig. 5.1 is proposed. In this scheme, an image to be transmitted first undergoes a general orthogonal transform and the transform coefficients are then quantized before transmission. The residual error array due to quantization is iteratively fedback, requantized and transmitted. At the receiving end, an approximation of the original image is reconstructed by progressively forming $\sum_{m=0}^{k} \hat{Q}_m$, which sums all the received information up to stage k and then performing the inverse transform.

If an optimum bit allocation (equation) is assumed for each stage, it can be shown that the average reconstruction error variance converges to zero, as the number of stages approaches infinity. In the proof, the following notation is adopted, †

X, Q_0 — the original and transformed images;

$\hat{X}_k$ — the kth approximation;

Q_k, $\hat{Q}_k$ — the residual error array at stage k in the transform domain and its quantized version;

$B_k = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} B_{k,ij}$ — the average bit rate for stage k;

$\sigma_{ij}^2 = \sigma_{0,ij}^2 = E(Q_{ij}^2) = E(Q_{0,ij}^2)$ — the variance of the (i,j)th transform coefficient;

$\sigma_{r,k+1,ij}^2 = E((X_{ij} - \hat{X}_{k,ij})^2)$ — the reconstruction error variance of the (i,j)th element of the image up to stage k;

$\sigma_{q,k+1,ij}^2 = E((Q_{k,ij} - \hat{Q}_{k,ij})^2)$ — the quantization error variance of the (i,j)th coefficient of the transformed image at stage k;

$\sigma_x^2 = \sigma_{q,0}^2 = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma_{ij}^2 = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma_{0,ij}^2$ — the average image energy;

$\sigma_{r,k+1}^2 = \mathrm{tr}(E((X - \hat{X}_k)'(X - \hat{X}_k)))/(N \times N)$ — the average reconstruction error variance up to stage k;

† The subscripts (r), (q), (k) and (i,j) represent reconstruction, quantization, stage k and element (i,j), respectively

$\sigma^2_{q,k+1} = \mathrm{tr}(E((Q_k - \hat{Q}_k)'(Q_k - \hat{Q}_k)))/(N \times N)$ — the average quantization error variance at stage k;

$\epsilon^2_k = (\prod^2_{ij} \epsilon^2_{k,ij})^{1/(N \times N)}$ — the correction factor defined in equation (2.4.20) for stage k.

Using the above notation and equation (2.4.21), the average quantization error variance at stage $k = 0$ can be expressed as follows,

$$\min(\sigma^2_{q,1}) = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{q,1,ij} = \epsilon^2_0 2^{-2B_0} \Big(\prod_{i,j=0}^{N-1} \sigma^2_{0,ij} \Big)^{1/(N \times N)} \tag{5.1.1}$$

or, alternatively as,

$$\min(\sigma^2_{q,1}) = \epsilon^2_0 2^{-2B_0} \frac{\left(\prod_{i,j=0}^{N-1} \sigma^2_{0,ij}\right)^{1/(N \times N)}}{\sigma^2_x} \sigma^2_x = \epsilon^2_0 2^{-2B_0} \frac{\left(\prod_{i,j=0}^{N-1} \sigma^2_{0,ij}\right)^{1/(N \times N)}}{\frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{0,ij}} \sigma^2_x. \tag{5.1.2}$$

In a similar manner, the average quantization error variance at subsequent stage $k = 1, 2, \ldots$, can be expressed as follows,

$$\begin{aligned}
\min(\sigma^2_{q,k+1}) &= \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{q,k+1,ij} \\
&= \sigma^2_{q,k+1,ij} \\
&= \epsilon^2_k 2^{-2B_k} \Big(\prod_{i,j=0}^{N-1} \sigma^2_{q,k,ij} \Big) \\
&= \epsilon^2_k 2^{-2B_k} \frac{\left(\prod_{i,j=0}^{N-1} \sigma^2_{q,k,ij}\right)^{1/(N \times N)}}{\min(\sigma^2_{q,k})} \min(\sigma^2_{q,k}) \\
&= \epsilon^2_k 2^{-2B_k} \frac{\left(\prod_{i,j=0}^{N-1} \sigma^2_{q,k,ij}\right)^{1/(N \times N)}}{\frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{q,k,ij}} \min(\sigma^2_{q,k}) \\
&= \Big(\prod_{m=0}^{k} \epsilon^2_m 2^{-2B_m} \frac{\left(\prod_{i,j=0}^{N-1} \sigma^2_{q,m,ij}\right)^{1/(N \times N)}}{\frac{1}{N \times N} \sum_{i,j=0}^{N-1} \sigma^2_{q,m,ij}} \Big) \sigma^2_x \\
&= \Big(\prod_{m=0}^{k} \frac{\min(\sigma^2_{q,m+1})}{\min(\sigma^2_{q,m})} \Big) \sigma^2_x.
\end{aligned} \tag{5.1.3}$$

A coding factor c_m for the mth stage is defined as the ratio of the average energy of the $(m+1)$st error array to that of mth error array, that is,

$$c_m = \frac{\min(\sigma^2_{q,m+1})}{\min(\sigma^2_{q,m})} = \epsilon^2_m 2^{-2B_m} \frac{(\prod_{i,j=0}^{N-1} \sigma^2_{q,m,ij})^{1/(N\times N)}}{\frac{1}{N\times N}\sum_{i,j=0}^{N-1} \sigma^2_{q,m,ij}}, \qquad m = 0,1,...k, \quad (5.1.4)$$

where

$$\epsilon^2_m 2^{-2B_m} = (\prod_{i,j=0}^{N-1} \epsilon^2_{m,ij})^{1/(N\times N)} \quad 2^{-2(\frac{1}{N\times N}\sum_{i,j=0}^{N-1} B_{m,ij})} = [\prod_{i,j=0}^{N-1} (\epsilon^2_{m,ij} 2^{-2B_{m,ij}})]^{1/(N\times N)}. \quad (5.1.5)$$

Note that

$$\frac{(\prod_{i,j=0}^{N-1} \sigma^2_{q,m,ij})^{1/(N\times N)}}{\frac{1}{N\times N}\sum_{i,j=0}^{N-1} \sigma^2_{q,m,ij}} \leq 1$$

with equality if, and only if, all the quantization error variances at stage m are equal. If the quantizers at stage $m = 0,1,...k$ are well designed such that (Table 2.4.1),

$$\epsilon^2_{m,ij} 2^{-2B_{m,ij}} < c < 1, \qquad i,j = 0,1,...N-1, \quad (5.1.6)$$

clearly, the coding factor from (5.1.4)

$$c_m < c < 1, \qquad m = 0,1,...k. \quad (5.1.7)$$

Using equations (5.1.3) to (5.1.7), it is now shown that the average reconstruction error variance converges to zero, as the number of stages, k, approaches infinity,

$$\begin{aligned}
\lim_{k\to\infty} \min(\sigma^2_{r,k+1}) &= \lim_{k\to\infty} \min(\sigma^2_{q,k+1}) \\
&= \lim_{k\to\infty} (\prod_{m=0}^{k} \frac{\min(\sigma^2_{q,m+1})}{\min(\sigma^2_{q,m})}) \sigma^2_x \\
&= \lim_{k\to\infty} (\prod_{m=0}^{k} c_m) \sigma^2_x \\
&< \sigma^2_x \lim_{k\to\infty} (\prod_{m=0}^{k} c) \\
&= \sigma^2_x \lim_{k\to\infty} c^{k+1} \\
&= 0. \qquad (5.1.8)
\end{aligned}$$

In other words, lossless transmission can be achieved.

Theoretically, for each stage except the 0th, the procedure of optimizing the bit allocation is not meaningful since the variances of the coefficients in the error array should be the same (2.4.19). However, practical quantizers for coefficient quantization are constrained to using nonnegative integers for the number of bits, $B_{m,ij}$, obtained in the bit allocation procedure (2.4.18). Particularly significant is the possibility of negative $B_{m,ij}$ values for coefficients whose variances are sufficiently smaller than the geometric mean, as given by (2.4.18). In this case, we set $B_{m,ij} = 0$, implying no transmission of the corresponding coefficient, and a corresponding quantization error variance, $\sigma^2_{q,m+1,ij} = \sigma^2_{q,m,ij}$. Equation (2.4.19) will not exactly be satisfied, namely, the distribution of the element variances in error array is nonuniform. Therefore, it is still necessary for practical system to optimize the bit allocation at each stage in order to code the error data more efficiently. The convergence in (5.1.8) can be also demonstrated for the practical case as follows. The quantizer performance factor for the (i,j)th coefficient of the error array at stage m can be written as (2.4.7),

$$\epsilon^2_{q,m,ij} = \frac{\sigma^2_{q,m+1,ij}}{\sigma^2_{q,m,ij}} = \begin{cases} \epsilon^2_{m,ij} 2^{-2B_{m,ij}}, & \text{if } B_{m,ij} > 0; \\ 0, & \text{if } B_{m,ij} = 0 \text{ and } \sigma^2_{m,ij} = 0; \\ 1, & \text{if } B_{m,ij} = 0 \text{ and } \sigma^2_{m,ij} \neq 0. \end{cases} \tag{5.1.9}$$

The quantization error variance for coefficient (i,j) at stage k can now be written in the form

$$\sigma^2_{q,k+1,ij} = \epsilon^2_{q,k,ij}\sigma^2_{q,k,ij} = \Big(\prod_{m=0}^{k} \epsilon^2_{q,m,ij}\Big)\sigma^2_{ij} \tag{5.1.10}$$

where

$$\prod_{m=0}^{k} \epsilon^2_{q,m,ij} = \prod_{\substack{m=0 \\ B_{m,ij}>0}}^{k} \epsilon^2_{m,ij} 2^{-2B_{m,ij}} \cdot \prod_{\substack{m=0 \\ B_{m,ij}=0 \\ \sigma^2_{m,ij}\neq 0}}^{k} 1 \cdot \prod_{\substack{m=0 \\ B_{m,ij}=0 \\ \sigma^2_{m,ij}=0}}^{k} 0. \tag{5.1.11}$$

It is clear that if the variance of coefficient (i,j) of the error array at any stage m, $\sigma^2_{m,ij}$, is equal to zero, then the corresponding coefficient variance at stage $k > m$ must also be equal to zero; that is,

$$\sigma^2_{k,ij} = 0, \qquad \text{for } k > m \qquad \text{if } \sigma^2_{m,ij} = 0, \tag{5.1.12}$$

Hence, for this case, as $k \rightarrow \infty$, we can conclude that,

$$\lim_{k\to\infty}(\prod_{m=0}^{k}\epsilon_{q,m,ij}^2) = 0. \tag{5.1.13}$$

If this situation does not arise within a finite number of stages for the (i,j)th coefficient, equation (5.1.11) can be rewritten as follows,

$$\prod_{m=0}^{k}\epsilon_{q,m,ij}^2 = \prod_{\substack{m=0\\B_{m,ij}>0}}^{k}\epsilon_{m,ij}^2 2^{-2B_{m,ij}} \cdot \prod_{\substack{m=0\\B_{m,ij}=0}}^{k} 1 = \prod_{\substack{m=0\\B_{m,ij}>0}}^{k}\epsilon_{m,ij}^2 2^{-2B_{m,ij}}. \tag{5.1.14}$$

For the case where $B_{m,ij} > 0$, that is, the number of bits assigned to coefficient (i,j) at stage m is a nonnegative integer, a well-designed quantizer can always make the quantizer performance factor (Table 2.4.1)

$$\epsilon_{m,ij}^2 2^{-2B_{m,ij}} < c < 1, \qquad m = 0, 1, ...k. \tag{5.1.15}$$

Hence, as $k \rightarrow \infty$, the product of the quantization performance factors approaches zero:

$$\lim_{k\to\infty}(\prod_{m=0}^{k}\epsilon_{q,m,ij}^2) = \lim_{k\to\infty}(\prod_{\substack{m=0\\B_{m,ij}>0}}^{k}\epsilon_{m,ij}^2 2^{-2B_{m,ij}}) < \lim_{k\to\infty}\prod_{\substack{m=0\\B_{m,ij}>0}}^{k} c = 0. \tag{5.1.16}$$

Based on the above analysis, the average reconstruction error variance converges to zero, as the number of iterative stages, k, approaches infinity,

$$\begin{aligned}
\lim_{k\to\infty}\sigma_{r,k+1}^2 &= \lim_{k\to\infty}\sigma_{q,k+1}^2 \\
&= \lim_{k\to\infty}\frac{1}{N\times N}\sum_{i,j=0}^{N-1}\sigma_{q,k+1,ij}^2 \\
&= \lim_{k\to\infty}\frac{1}{N\times N}\sum_{i,j=0}^{N-1}(\prod_{m=0}^{k}\epsilon_{q,m,ij}^2)\sigma_{ij}^2 \\
&= \frac{1}{N\times N}\sum_{i,j=0}^{N-1}\lim_{k\to\infty}(\prod_{m=0}^{k}\epsilon_{q,m,ij}^2)\sigma_{ij}^2 \\
&= \frac{1}{N\times N}\sum_{i,j=0}^{N-1} 0\cdot\sigma_{ij}^2 \\
&= 0.
\end{aligned} \tag{5.1.17}$$

5.2 RESIDUAL ERROR VECTOR QUANTIZATION

Rate distortion theory [27-31] indicates that better performance can be always achieved by coding vectors instead of scalars, even in the case of a memoryless sources. Furthermore, a well-defined source can be compressed arbitrarily close to the rate distortion function as the coding block size approaches infinity. This implies that the ratio of the distortion obtained by using vector quantization with sufficiently large block size to that by using scalar quantization with optimum bit allocation (2.4.19) is less than 1, i.e. ‡

$$\beta = \frac{\sigma_{vq}^2}{\min(\sigma_q^2)} < 1. \tag{5.2.1}$$

The gain due to using vector quantization instead of optimum scalar quantization is just the reciprocal of β.

Taking a Gaussian source X of size $N \times N$ as an example and the mean square error (MSE) as the distortion measure, after an optimum orthogonal transform, the transform coefficients Q_{ij}, $i,j = 0,1,...N-1$ are still Gaussian and statistically independent with variances $\sigma_{ij}^2 = E(Q_{ij}^2)$, $i,j = 0,1,...N-1$. In fact, σ_{ij}^2 is the (i,j)th eigenvalue of the Gaussian source of $N \times N$. For such a statistically independent Gaussian source X of size $N \times N$, it can be shown [28,47] that the $N \times N$ block distortion rate function is

$$D_G(\phi) = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \min(\phi, \sigma_{ij}^2), \tag{5.2.2}$$

$$R_G(\phi) = \frac{1}{N \times N} \sum_{i,j=0}^{N-1} \max(0, \frac{1}{2} \log_2 \frac{\sigma_{ij}^2}{\phi}) \tag{5.2.3}$$

where R_G can be interpreted as the average rate. For the case of small distortion, i.e, $\phi \leq \min(\sigma_{ij}^2)$, where the minimum is over all $i,j = 0,1,...N-1$, the $N \times N$ block distortion rate function for a given rate R_G is

$$D_G = 2^{-2R_G} \left(\prod_{i,j=0}^{N-1} \sigma_{ij}^2 \right)^{1/(N \times N)} \tag{5.2.4}$$

‡ The subscript (vq) represents vector quantization

From equations (2.4.19) and (5.2.4), the difference between the distortion by using scalar quantization with optimum bit allocation and the $N \times N$ block distortion rate function for a given bit rate B in the transform domain is,

$$\delta = (\epsilon^2 - 1)2^{-2B}\left(\prod_{i,j=0}^{N-1} \sigma_{ij}^2\right)^{1/(N\times N)} \tag{5.2.5}$$

As $\epsilon_{ij}^2 > 1, \quad i,j = 0,1,...N-1$ for a Gaussian source (Table 2.4.1).

$$\epsilon^2 = \left(\prod_{i,j=0}^{N-1} \epsilon_{ij}^2\right)^{1/(N\times N)} > 1. \tag{5.2.6}$$

We conclude that the difference δ is always greater than 0,

$$\delta > 0. \tag{5.2.7}$$

This implies that it is impossible to reach the distortion rate function by the optimum scalar quantization. The ratio of the distortion obtained by using vector quantization with sufficiently large block size to that obtained by using scalar quantization with an optimum bit allocation (2.4.19) is, therefore,

$$\frac{\sigma_{vq}^2}{\min(\sigma_q^2)} \approx \frac{D_G(B)}{\min(\sigma_q^2)} = \frac{2^{-2B}\left(\prod_{i,j=0}^{N-1} \sigma_{ij}^2\right)^{1/(N\times N)}}{\epsilon^2 2^{-2B}\left(\prod_{i,j=0}^{N-1} \sigma_{ij}^2\right)^{1/(N\times N)}} = \frac{1}{\epsilon^2} = \beta_G. \tag{5.2.8}$$

Since

$$\epsilon^2 > 1, \tag{5.2.9}$$

the ratio,

$$\beta_G < 1. \tag{5.2.10}$$

The gain due to using vector quantization instead of the optimum scalar quantization in the transform domain is approximately equal to the reciprocal of β_G. For non-Gaussian image sources, there are no explicit distortion rate functions but upper bounds are available, for a fixed rate R, in the form,

$$D(R) \leq D_G(R), \tag{5.2.11}$$

implying that the Gaussian image source is the most difficult to reproduce for a fixed second moment under the MSE criterion. This might lead to the intuitive conclusion that, for a rather broad class of distributions, more benefit can be expected using vector quantization as opposed to scalar quantization with optimum bit allocation in the transform domain.

Based on the above analysis, if the scalar quantization with optimum bit allocation in Fig. 5.1 is replaced by the vector quantization, further improvement in coding performance can be expected. It is now demonstrated that, as in scalar quantization case, the average reconstruction error variance converges to zero, as the number of iterative stages approaches infinity. Let the average error variance at stage k using vector quantization in the transform domain be as follows,

$$\sigma^2_{vq,k+1} = \mathrm{tr}(E((Q_k - \hat{Q}_k)'(Q_k - \hat{Q}_k)))/(N \times N), \qquad k = 0, 1, ... \tag{5.2.12}$$

and

$$\sigma^2_{vq,0} = \frac{1}{N \times N} \sum_{ij=0}^{N-1} \sigma^2_{0,ij} = \sigma^2_x \tag{5.2.13}$$

The coding factor at the mth stage is then,

$$c_m = \frac{\sigma^2_{vq,m+1}}{\sigma^2_{vq,m}} = \frac{\sigma^2_{vq,m+1}}{\min(\sigma^2_{q,m+1})} \frac{\min(\sigma^2_{q,m+1})}{\sigma^2_{vq,m}} = \beta_m \frac{\min(\sigma^2_{q,m+1})}{\sigma^2_{vq,m}}, \tag{5.2.14}$$

where β_m is the ratio of the distortion obtained by applying vector quantization to that obtained by applying scalar quantization with optimum bit allocation at stage m. In a similar fashion as in scalar quantization case, the average reconstruction error variance using vector quantization can be expressed as follows,

$$\begin{aligned} \sigma^2_{r,k+1} = \sigma^2_{vq,k+1} &= (\prod_{m=0}^{k} \frac{\sigma^2_{vq,m+1}}{\sigma^2_{vq,m}})\sigma^2_x \\ &= (\prod_{m=0}^{k} c_m)\sigma^2_x \\ &= (\prod_{m=0}^{k} \beta_m \frac{\min(\sigma^2_{q,m+1})}{\sigma^2_{vq,m}})\sigma^2_x. \end{aligned} \tag{5.2.15}$$

As, from equation (5.2.1),

$$\beta_m < 1, \qquad m = 0, 1, ...k, \tag{5.2.16}$$

and from equation (5.1.7), the ratio of the average distortion arising from scalar quantization (with optimum bit allocation) on the mth error array to the average variance of the mth error array,

$$\frac{\min(\sigma^2_{q,m+1})}{\sigma^2_{vq,m}} < c < 1, \qquad m = 0, 1, ...k \tag{5.2.17}$$

it can now be shown that the average reconstruction error variance (5.2.15) converges to zero, as $k \to \infty$,

$$\begin{aligned} \lim_{k\to\infty} \sigma^2_{r,k+1} &= \lim_{k\to\infty} \sigma^2_{vq,k+1} \\ &= \lim_{k\to\infty} (\prod_{m=0}^{k} c_m)\sigma^2_x \\ &< \sigma^2_x \lim_{k\to\infty} (\prod_{m=0}^{k} \beta_m c) \\ &= 0. \end{aligned} \tag{5.2.18}$$

It should be observed from (5.1.8) and (5.2.18) that the convergent speed using vector quantization is faster than that using scalar quantization because the gain β_m is less than 1. Note that, for both vector quantization and optimum scalar quantization, the distribution of the energies of the elements over the error array tends to become more uniform and the corresponding average energy decreases with increasing number of stages.

5.3 LOSSLESS PROGRESSIVE TRANSMISSION WITH FINITE ITERATIVE STAGES

It has been shown that the reconstruction error variance converges to zero, as the number of iterative stages approaches infinity. However, in practice, it is desired that the original image (digital) be reproduced with perfect fidelity with a finite number

of stages. The technique presented in the previous section is modified, as shown in Fig. 5.2; the coded information $\hat{Q}_k$, $k = 0, 1, \ldots$ are summed and stored at the transmitter. Upon the user's request, an approximation $\hat{X}_k$ can be reconstructed by inverse transforming the summed quantity $\sum_{m=0}^{k} \hat{Q}_k$ at both the transmitting and receiving ends, and rounded off to the same resolution as in the original image. The residual error image can now be losslessly coded by an entropy coder and the original image can be reproduced by adding the losslessly coded error image E_{k+1} to the approximation $\hat{X}_k$ at the receiver. Note that since each iterative stage partially removes the statistical redundancy of the original image, the elements of the residual error image tend to become decorrelated and the first order entropy of the residual error image tends to the entropy of the corresponding error image. Therefore, a Huffman coder operating on a pixel-by-pixel basis can be used to efficiently and losslessly encode the final error image.

5.4 SIMULATIONS

Computer simulations are carried out with the proposed progressive image transmission scheme on the face image shown in Fig. 5.3; the image is of size 256×256 pixels and quantized to 256 levels. The first order entropy estimated from the histogram of grey levels of the original face image is 7.5178 bits/pixel.

The original image first undergoes a two-dimensional discrete block cosine transform and then the transformed image is iteratively quantized, scalar or vector, respectively. Two block sizes are used, 4×4 and 8×8. In scalar quantization (RSQ-TRA), the bit allocation map is obtained from equation (2.4.18) and is transmitted as overhead at each stage. In vector quantization (RVQ-TRA), one vector is derived from each transform block, 4×4 or 8×8, respectively. At each stage, a codebook is generated from all possible input vectors using the generated Lloyd algorithm [65] and is transmitted as overhead.

The results of the computer simulations are listed in Table 5.1, where the com-

ponents shown are defined as follows,

RSQ-TRA:

$R_{co}(k)$ - the bit rate for coding the coefficients at stage k;

$R_o(k)$ - the bit rate for transmitting the bit allocation map at stage k, since the matrices are quite sparse, just the positions retained need to be indicated;

$R_p(k) = \sum_{m=0}^{k}(R_{co}(m) + R_o(m))$ - the total progressive bit rate up to stage k;

$H_e^1(k)$ - the first order entropy of the residual error image at stage k;

$R_t(k) = R_p(k) + H_e^1(k)$ - the total bit rate required for lossless reproduction at stage k (4.1.13).

RVQ-TRA:

$R_l(k)$ - the bit rate for coding the labels at stage k, where a variable length coder is used, dependent upon the frequency of occurrence of labels;

$R_c(k)$ - the bit rate for transmitting the codebook at stage k, where a variable length coder is also used;

$R_p(k) = \sum_{m=0}^{k}(R_l(m) + R_c(m))$ - the total progressive bit rate up to stage k;

$H_e^1(k)$ - the first order entropy of the residual error image at stage k (5.2);

$R_t(k) = R_p(k) + H_e^1(k)$ - the total bit rate required for lossless reproduction at stage k (4.1.13).

In the evaluation of system performance, the normalized mean square error (NMSE) is used, defined as follows,

$$\text{NMSE} = \frac{\sum_{i,j=0}^{255}(X_{ij} - \hat{X}_{k,ij})^2}{\sum_{i,j=0}^{255} X_{ij}^2}, \qquad k = 0, 1, \ldots \qquad (5.4.1)$$

where $X_{i,j}$ and $\hat{X}_{k,i,j}$ represent the (i,j)th element of the original image and the kth approximation, respectively.

Fig. 5.4 shows the rate distortion curves for both optimum scalar quantization and vector quantization (Table 5.1), where the increasing bit rates correspond to iterative/progressive quantization. It is clear that the best performance is achieved by using vector quantization on 8×8 pixel blocks. At 1.0 bits/pixel, there is an improvement of 22.7% ($\frac{0.1130-0.0873}{0.1130} \times 100\%$) over optimum scalar quantization on 8×8 pixel blocks and 19.3% ($\frac{0.1290-0.1041}{0.1290} \times 100\%$) on 4×4 pixel blocks, respectively. At 0.5 bits/pixel, the corresponding percentages are, respectively, 15.1.6% and 36.%.

Fig. 5.5 presents the zeroth, first, second, fourth, fifth and ninth approximation images by using vector quantization on 8×8 pixel block in the progressive sequence. Note that the first approximations in the sequence have already provided the main structural information of the original image at a very low bit rate and that by the fourth iteration, there has already been a dramatic improvement at a total progressive bit rate of 0.4954 bits/pixel.

Fig. 5.6 and Table 5.1 show that, with increasing number of iterations, the total bit rate curves, $R_t(k)$, required for lossless reproduction first approach a lower value than the first order entropy of the original image. For example, the minimum $R_t(k)$ reached at $k = 7$ is only 4.9685 bits/pixel. Note that, at bit rates between 0.4-1.0 bits/pixel, the total bit rate for optimum scalar quantization (8×8) and vector quantization (8×8) are almost identical, implying that either one can be used. However, at a higher bit rate (> 1.0 bits/pixel), the total bit rate for optimum scalar quantization is 5% higher than for vector quantization. The vector quantization is more efficient in removing the redundancy of the residual error images than optimum scalar quantization.

The simulation results using a single-stage transform coding are shown in Table 5.2. RQ-TRA is seen in many cases to outperform a single-stage transform coding.

Note the large bit rate required for transmitting the codebook for the (8×8) case. As mentioned in chapter 4, if multi-stage coding could be optimized at each stage (that is, only the net information of the image is transmitted at each stage), then the performance by iteratively encoding the transformed image should be no worse than that by single-stage transform coding.

We now turn to compare RVQ-TRA with RVQ-SPA (chapter 4). Fig. 5.7 shows the rate distortion curves drawn from Tables 4.1b and 5.1b for a block size $= 4 \times 4$. It can be noted that no advantage is obtained by using RVQ-TRA over RVQ-SPA. A possible explanation is that the vectors are simply formed from the block transform coefficients without regards to nonuniform energy distribution. In other words, the same number of bits are effectively allocated to each coefficient.

5.5 CONCLUSIONS

In this chapter, the residual error quantization technique is extended to the transform domain. The residual errors after transform coefficient coding are iteratively fedback and requantized (scalar RSQ or vector RVQ). Both the theoretical analysis and experimental results obtained demonstrate that lossless progressive transmission with a degree of compression is achieved. The reconstructed approximations are seen to rapidly converge to a good quality, both, subjectively and objectively. However, in comparing the performance of RVQ-TRA with RVQ-SPA's at a block size $= 4 \times 4$, we found that there is no improvement, as shown in Fig. 5.7. The reason could be that the energy compaction property in the transform domain is not exploited. In the following chapter, we will discuss how to optimally apply vector quantization to the transform coefficients.

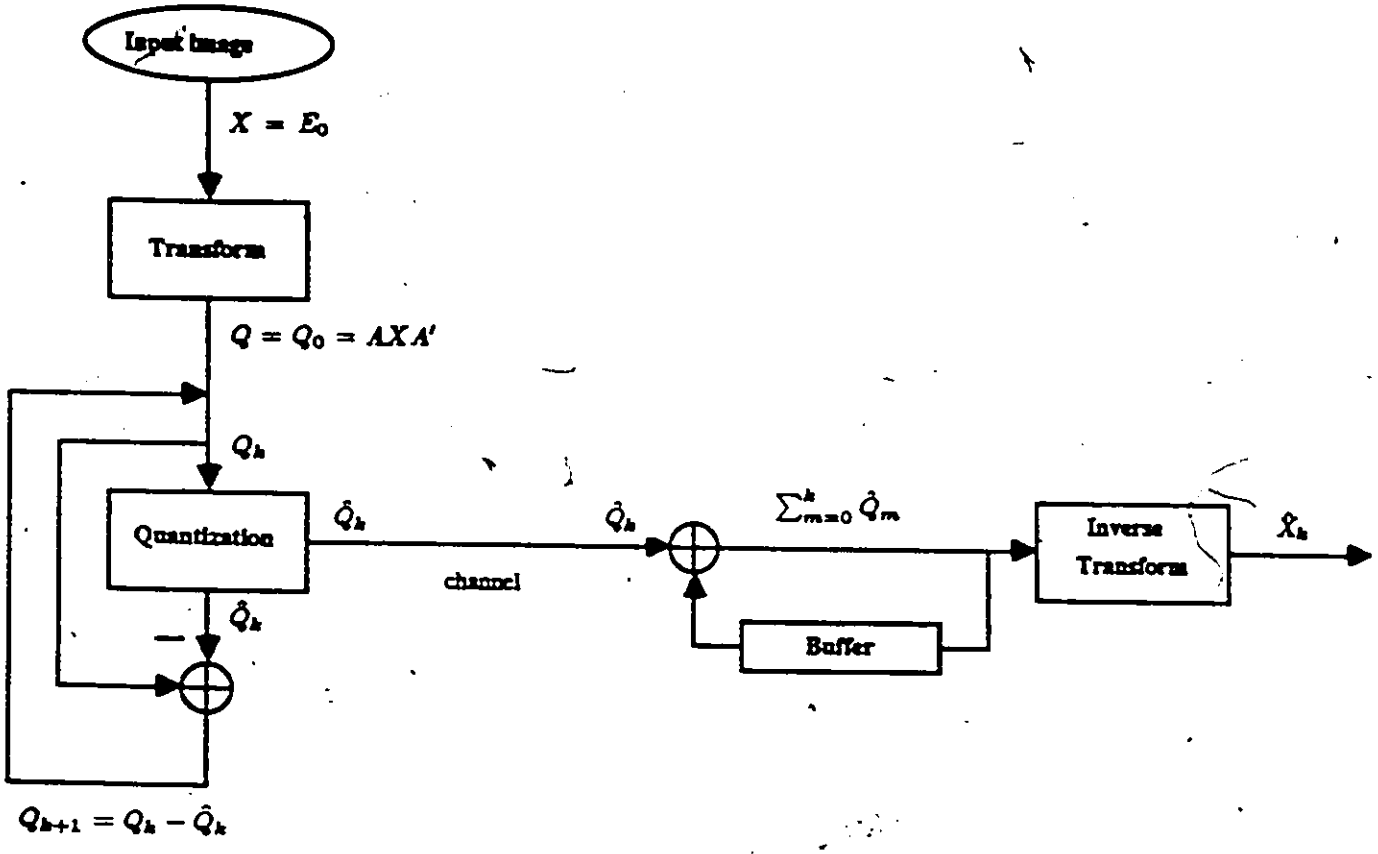


Fig. 5.1. Residual error quantization in the transform domain. The image to be transmitted first undergoes a general orthogonal transform and the transform coefficients are then quantized before transmission. The residual error array due to quantization is iteratively fedback, requantized and transmitted. At the receiving end, an approximation of the original image is reconstructed by progressively forming $\sum_{m=0}^{k} \hat{Q}_m$, which sums all the received information up to stage k and then performing the inverse transform.

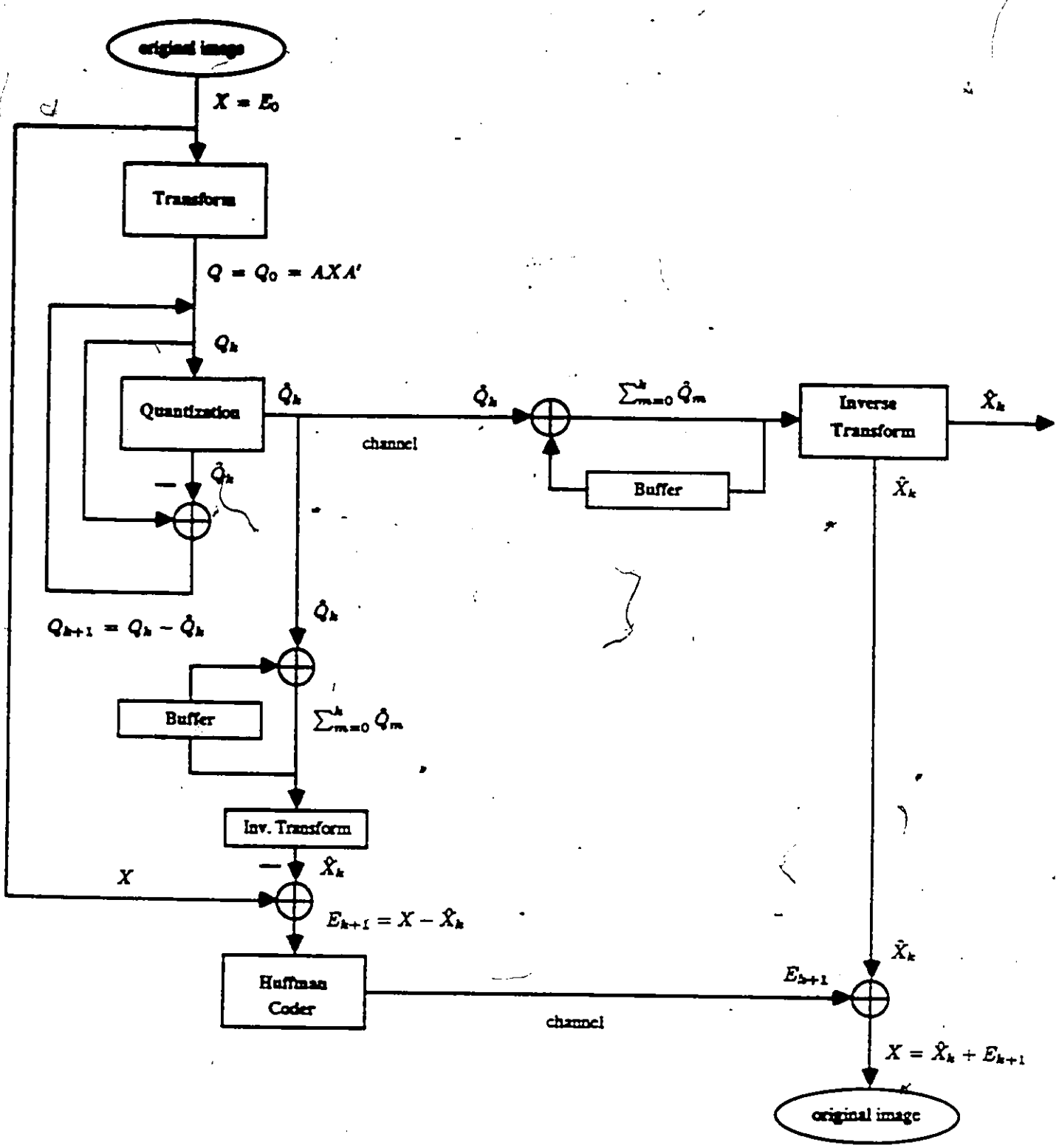


Fig. 5.2. Lossless progressive transmission with finite iterative stages. The coded information $\hat{Q}_k$, $k = 0, 1, \ldots$ are summed and stored at the transmitter. Upon user request, an approximation $\hat{X}_k$ can be reconstructed by inverse transforming the summed quantity $\sum_{m=0}^{k} \hat{Q}_k$ at both the transmitting and receiving ends, and rounded off to the same resolution as in the original image. The residual error image $E_{k+1} = X - \hat{X}_k$ can now be losslessly coded by entropy coding and the original image can be reproduced by adding the losslessly coded error image E_{k+1} to the approximation $\hat{X}_k$ at the receiver.

Fig. 5.3. The face image of 256 × 256 pixels with 8 bits/pixel.

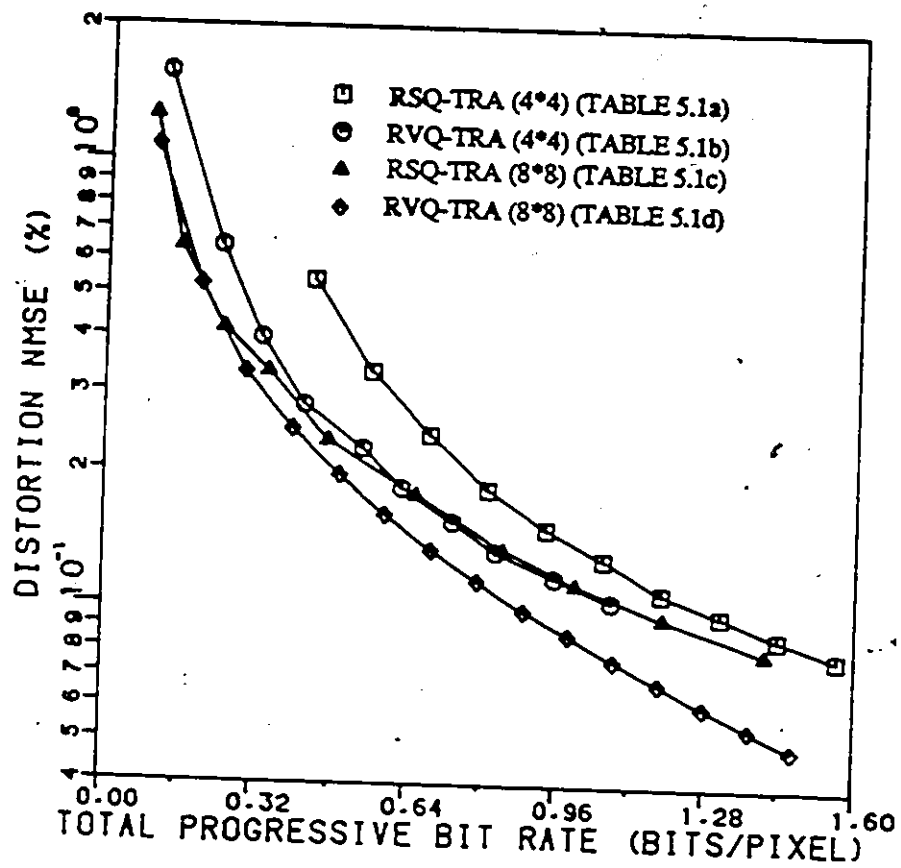


Fig. 5.4. The rate distortion curves for optimum scalar quantization and vector quantization on the block Cosine transformed images. The increasing bit rates correspond to iterative/progressive quantization. It is clear that the best performance is achieved by using vector quantization on 8×8 pixel blocks.

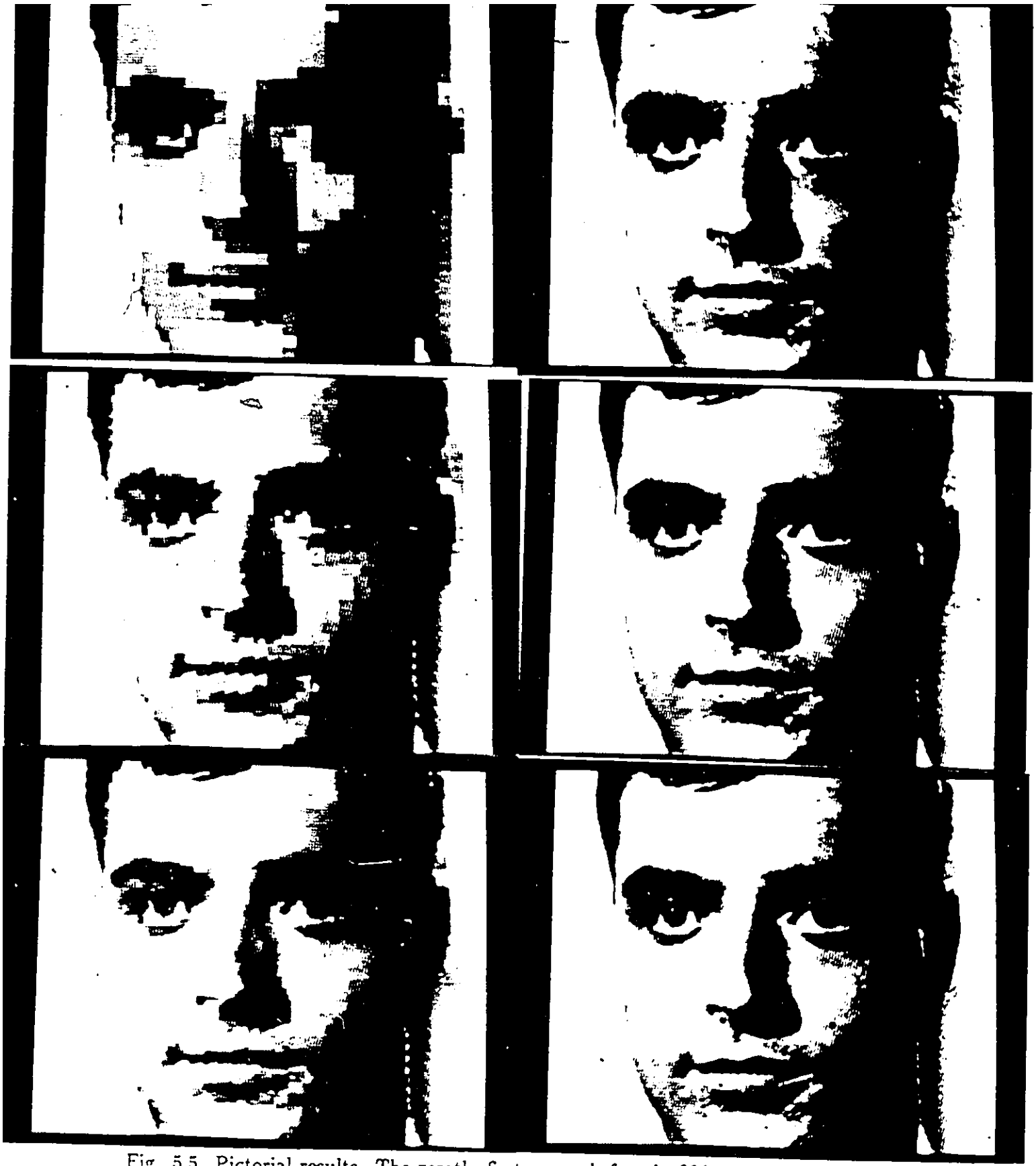


Fig. 5.5. Pictorial results. The zeroth, first, second, fourth, fifth and ninth approximation images using vector quantization on 8×8 pixel blocks are presented in the progressive sequence. Note that the first approximations in the sequence have already provide the main structural information of the original image at a very low bit rate and that, by the fourth iteration, there has already been a dramatic improvement at a total progressive bit rate of 0.4954 bits/pixel.

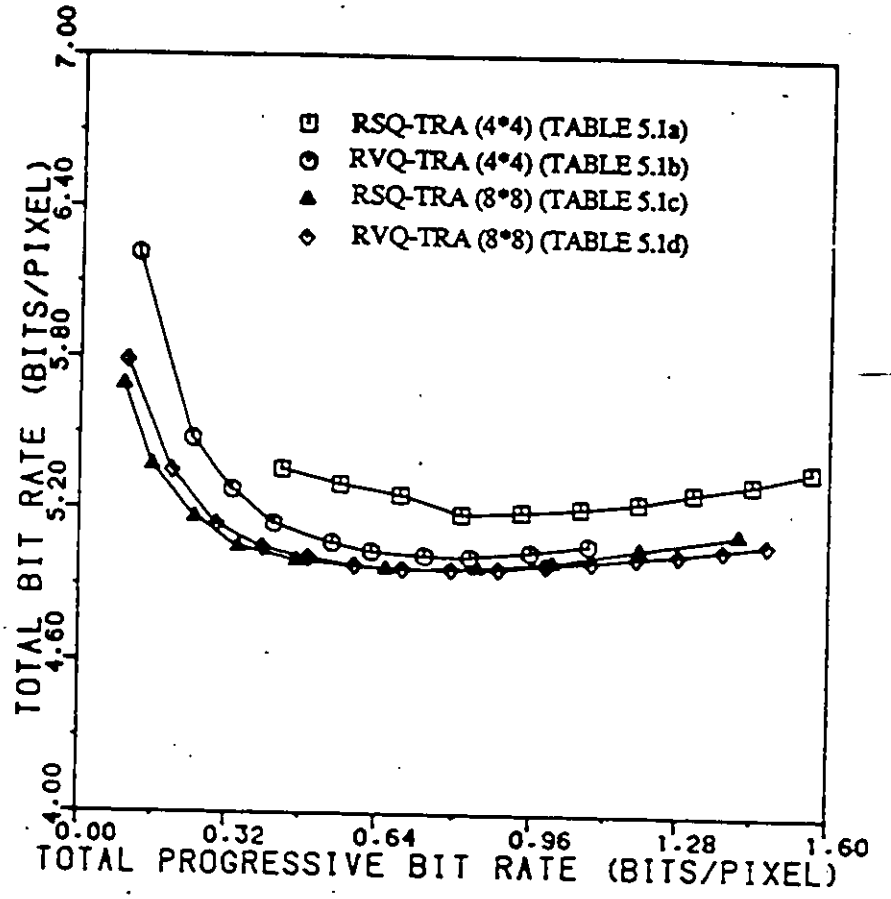


Fig. 5.6. Total bit rate curves. With increasing number of iterations, the total bit rate curves $R_t(k)$ required for lossless reproduction approach a lower value than the first order entropy of the original image. The minimum $R_t(k)$ reached at $k = 7$ is only 4.9685 bits/pixel, implying that, as a lossless coder, the proposed scheme is superior to Huffman coder operating on a pixel-by-pixel basis, by about 2.5493 (7.5178-4.9685) bits/pixel.

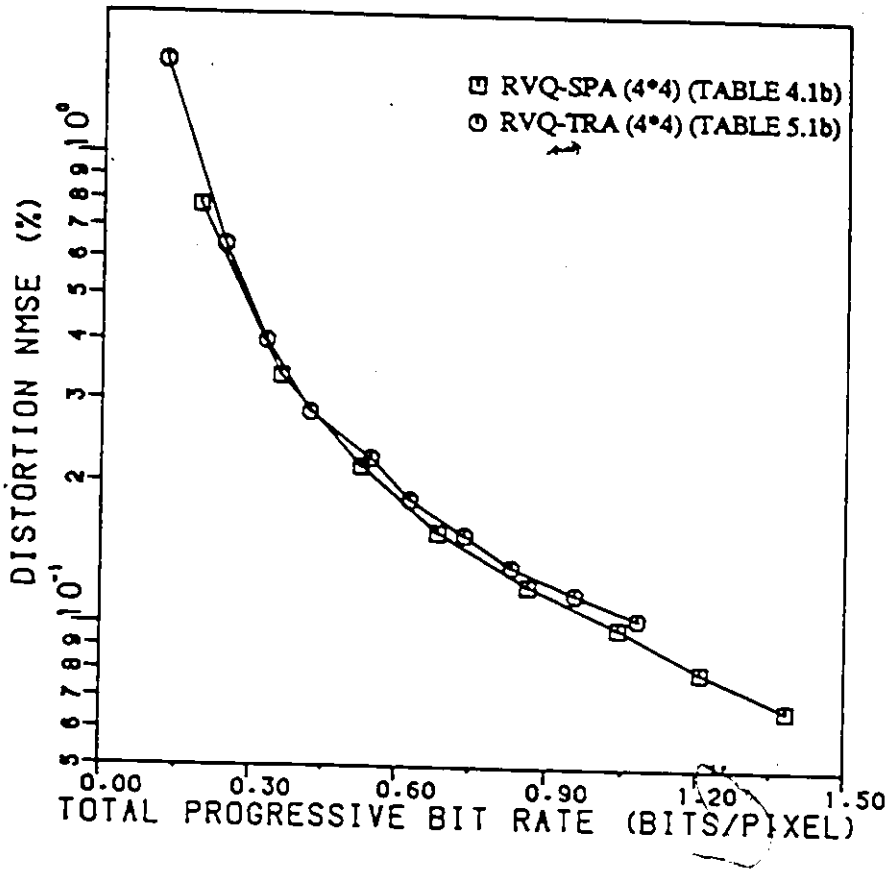


Fig. 5.7. The rate distortion curves for RVQ-TRA and RVQ-SPA with a block size of 4×4 (from Tables 5.1b and 4.1b).

Table 5.1a Computer simulation results for the face image by using RSQ-TRA with a block size of 4*4

k	Rco(k) (bits/pixel)	Ro(k) (bits/pixel)	Rp(k) (bits/pixel)	Hé(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	0.437500	0.000122	0.4376	4.9238	5.3614	0.5387
1	0.125000	0.000122	0.5627	4.7410	5.3037	0.3352
2	0.125000	0.000122	0.6878	4.5744	5.2622	0.2429
3	0.125000	0.000122	0.8129	4.3738	5.1867	0.1840
4	0.125000	0.000122	0.9381	4.2597	5.1978	0.1502
5	0.125000	0.000122	1.0632	4.1512	5.2144	0.1290
6	0.125000	0.000122	1.1883	4.0499	5.2382	0.1090
7	0.125000	0.000122	1.3134	3.9646	5.2780	0.0975
8	0.125000	0.000122	1.4386	3.8759	5.3145	0.0870
9	0.125000	0.000122	1.5637	3.8032	5.3669	0.0790

Table 5.1b Computer simulation results for the face image by using RVQ-TRA with a block size of 4*4 (the number of codewords per stage = 4)

k	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)	Rp(k) (bits/pixel)	Hé(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	0.121190	0.001536	0.1227	6.0897	6.2124	1.5706
1	0.119984	0.001358	0.2440	5.2356	5.4796	0.6402
2	0.083548	0.002233	0.3298	4.9498	5.2796	0.3997
3	0.088001	0.002089	0.4199	4.7249	5.1448	0.2822
4	0.123560	0.001787	0.5452	4.5247	5.0699	0.2258
5	0.079470	0.002383	0.6271	4.4107	5.0378	0.1844
6	0.107896	0.001843	0.7368	4.2850	5.0218	0.1553
7	0.090915	0.002190	0.8300	4.1866	5.0166	0.1334
8	0.123974	0.001971	0.9559	4.0811	5.0370	0.1172
9	0.122675	0.001962	1.0805	3.9857	5.0662	0.1041

Table 5.1c Computer simulation results for the face image by using RSQ-TRA with a block size of 8*8

k	Rco(k) (bits/pixel)	Ro(k) (bits/pixel)	Rp(k) (bits/pixel)	Hé(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	0.093750	0.000183	0.0939	5.5947	5.6886	1.2529
1	0.062500	0.000274	0.1567	5.2132	5.3699	0.6359
2	0.093750	0.000366	0.2508	4.9168	5.1676	0.4176
3	0.093750	0.000366	0.3449	4.7057	5.0506	0.3349
4	0.125000	0.000457	0.4704	4.5276	4.9980	0.2346
5	0.187500	0.000640	0.6585	4.3122	4.9707	0.1781
6	0.187500	0.000640	0.8466	4.1279	4.9745	0.1348
7	0.156250	0.000549	1.0034	3.9941	4.9975	0.1130
8	0.187500	0.000640	1.1916	3.8618	5.0534	0.0957
9	0.218750	0.000732	1.4111	3.7026	5.1137	0.0804

Table 5.1d Computer simulation results for the face image by using RVQ-TRA with a block size of 8*8 (the number of codewords per stage = 16)

k	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)	Rp(k) (bits/pixel)	Hé(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	0.054359	0.046468	0.1008	5.6846	5.7854	1.0728
1	0.052783	0.046736	0.2003	5.1504	5.3507	0.5239
2	0.053006	0.043770	0.2971	4.8437	5.1408	0.3336
3	0.053839	0.044804	0.3957	4.6534	5.0491	0.2487
4	0.050659	0.048990	0.4954	4.5167	5.0121	0.1965
5	0.052925	0.044111	0.5924	4.3904	4.9828	0.1604
6	0.056635	0.042422	0.6915	4.2775	4.9690	0.1341
7	0.057889	0.042031	0.7914	4.1771	4.9685	0.1146
8	0.056650	0.042198	0.8902	4.0794	4.9696	0.0986
9	0.058294	0.040150	0.9887	3.9993	4.9880	0.0873
10	0.058592	0.039652	1.0869	3.9131	5.0000	0.0767
11	0.059258	0.037825	1.1840	3.8344	5.0184	0.0683
12	0.060317	0.035568	1.2799	3.7523	5.0322	0.0607
13	0.059358	0.037650	1.3769	3.6766	5.0535	0.0544
14	0.059599	0.034227	1.4707	3.6049	5.0756	0.0490

Table 5.2a Computer simulation results for the face image
by using a single-stage transform coding with scalar quantization

Block Size	Rco (bits/pixel)	Ro (bits/pixel)	Rp (bits/pixel)	H_e' (bits/pixel)	Rt (bits/pixel)	NMSE(%)
4*4	1.000000	0.000335	1.000335	4.3700	5.3703	0.1755
4*4	1.500000	0.000549	1.500549	4.0915	5.5920	0.1113
8*8	1.031250	0.001541	1.032791	4.2009	5.2337	0.1495
8*8	1.421875	0.001846	1.423721	4.1641	5.5878	0.1390

Table 5.2b Computer simulation results for the face image
by using a single-stage transform coding with vector quantization

Block Size	Nc	Rl (bits/pixel)	Rc (bits/pixel)	Rp (bits/pixel)	H_e' (bits/pixel)	Rt (bits/pixel)	NMSE(%)
4*4	128	0.397508	0.136403	0.533912	4.5008	5.0347	0.1940
4*4	256	0.454878	0.301180	0.756058	4.2891	5.0451	0.1374
8*8	32	0.071024	0.103060	0.174085	5.3880	5.5621	0.7204
8*8	64	0.084240	0.240028	0.324268	5.1299	5.4542	0.4999
8*8	128	0.097987	0.528528	0.626515	4.8905	5.5170	0.3497
8*8	256	0.111631	1.071370	1.183002	4.5445	5.7275	0.2165

where Nc is the number of codewords.

6. MULTISTAGE VECTOR QUANTIZATION WITH OPTIMUM BIT ALLOCATION PLANES

In this chapter, we present a multistage vector quantization scheme (MVQ-OBA) in which the transform coefficients with higher energies are allocated more bits than the coefficients with lower energies, in other words, the number of bits assigned to each coefficient is proportional to the coefficient variance. An image to be coded first undergoes a block orthogonal transform. An optimal bit allocation map $\{B_{ij}\}$ is then found for a fixed bit rate (2.4.18), where the number of bits assigned to each coefficient is proportional to the coefficient variance. The optimal bit allocation map is then sliced into a set of "bit allocation planes", $\{B_{k,ij},\ k = 0, 1, ..., \}$ by applying a set of the thresholds, $\{T_k\}$. Here, $B_{k,ij}$ indicates the number of bits assigned to coefficient (i,j) at stage k. We note that $B_{k,ij}$ is not restricted to being an integer value. At each stage, all the components with a non-zero number of bits are combined into a vector and vector quantized. The residual error vectors are then calculated and fed to the next stage of the coder. In other words, the transform coefficients are first vector quantized with the first bit allocation plane, B_0; the residual error coefficients are then requantized with the second bit allocation plane, B_1; and the same process is repeated until the last bit allocation plane is used. A separate codebook is generated at each stage by using the generalized Lloyd clustering algorithm with the number of codewords equal to $2^{[\sum_{i,j} B_{k,ij}]}$ (where $[\alpha]$ means the truncation of $\alpha + 0.5$).

It is shown below that the effective number of bits allocated to each coefficient is proportional to the coefficient variance. Furthermore, as a variable number of coefficients are used in forming the vectors at each stage, the coefficients are effectively assigned a fractional number of bits if needed. An important property is that at each stage, the coefficient error variances due to quantization are similar. Since the coefficients are coded on a stage-by-stage basis, the information coded up to a certain stage in fact corresponds to an approximation of the image. This means that the

proposed technique results in progressive image transmission. To achieve lossless transmission, an entropy coder can be used to encode the final residual error image. The simulation results demonstrate that the proposed vector quantization scheme using the optimal bit allocation planes in the transform domain is superior to both scalar coefficient quantization [43,62] and direct vector quantization on the coefficients [83,88]. The superiority can be traced to coding vectors instead of scalars and using an optimal bit distribution strategy in vector quantization.

In the following section, the rationale for vector quantization in the transform domain are given. Section 6.2 describes the algorithm for slicing an optimal bit allocation map into a set of optimal bit allocation planes. In section 6.3, a multistage vector quantization technique with optimal bit allocation planes is presented. In section 6.4, a few properties of the optimal bit allocation planes are listed. The simulation results using the proposed technique are reported in section 6.5. Finally, the proposed transform vector quantization is compared with various other transform domain coding techniques in section 6.6.

6.1 RATIONALE

Transform coding with optimal bit allocation [43-49] is known to be very effective for image data compression. An image to be coded first undergoes an orthogonal transform, and the resulting coefficients are then scalar quantized with an optimal bit allocation map where the number of bits allocated is proportional to the coefficient variance. By using an optimal bit allocation scheme, identical (coefficient quantization) error variances result [47]. Such transform coding, however, is only suboptimal, as quantization is performed in a scalar manner. Information theory tells us that improved performance can be achieved if vector, as opposed to scalar, quantization is applied to the transform coefficients. However, in applying vector quantization in the transform domain, two issues must be addressed: how to form the vectors from the coefficients and how to optimally allocate the available bits.

A direct approach to vector formation is simply to use all the coefficients as the components of the vectors. In this case, the available bits are allocated relatively uniformly over the coefficients or components of the vectors. Such an approach is inefficient because the image information is nonuniformly distributed over the coefficients. As we demonstrate experimentally below, direct vector quantization in the transform domain does not result in identical error variances of the coefficient if the generalized Lloyd clustering algorithm is used in codebook generation. This means that the distribution of the coded bits is not proportional to the coefficient variances. A proper coding procedure should nonuniformly distribute the coding bits over the coefficients according to the coefficient variances.

Two more practical difficulties with direct vector quantization in the transform domain are a large overhead for transmitting the codebook and a large computational cost. For example, for a block size of 8×8, from equation (2.5.12), direct vector quantization of an image of size 256×256 at a bit rate of 0.5 bits/pixel (for the labels) would require an overhead for the codebook, $R_c = 8 \times 8 \times 2^{8 \times 8 \times 0.5} \times 8/(256 \times 256) = 2^{25}$ bits/pixel (assuming that the average number of bits per codeword component =8 bits/pixel), which is clearly impossible. The associated computational cost (equation (2.5.9)) is proportional to $8 \times 8 \times 2^{8 \times 8 \times 0.5} = 2.7487 \times 10^{11}$; which is not practical. To overcome these problems, a few alternative approaches to vector formation have been proposed [83,88,21,87]. In [83,88], the dimensionality of the vectors is reduced by using only the coefficients with the highest variances as the vector components. However, a basic problem is the optimal choice of the coefficients and the number of codewords in the codebook. Instead of being quantized in one stage, the coefficients are vector quantized on a stage-by-stage basis [21]. At each stage, a smaller codebook size can be used, thus reducing both overhead and computational cost. In [88], the transform block of size 8×8 is partitioned into 14 different low dimensional vectors which are separately (vector) quantized. Note that since vectors with small dimensionality are used, both the overhead for transmitting the codebooks and the computational cost

are reduced.

6.2 OPTIMAL BIT ALLOCATION PLANES

We now present a multistage vector quantization technique (MVQ-OBA) operating on the transform coefficients where the effective number of bits assigned to the individual coefficient is proportional to the coefficient variance. The block diagram of the proposed technique is shown in Fig. 6.1. An image to be coded first undergoes an orthogonal transform. An optimal bit allocation map, $\{B_{ij}\}$, is then found based upon the variances of the coefficients for a fixed bit rate from equation (2.4.18). Fig. 6.2 shows an example of a typical optimal bit allocation map. The numbers of bits assigned to the coefficients are seen to vary significantly from 5 to 0.375. In order to optimally distribute the bits over the coefficients when using vector quantization, a set of bit allocation planes, $\{B_{k,ij}\}$, is used instead of the optimal bit allocation map, $\{B_{ij}\}$. The set of bit allocation planes, $\{B_{k,ij},\ k = 0, 1, ...\}$ is obtained by slicing the optimal bit allocation map, $\{B_{ij}\}$, with a set of the thresholds $\{T_k\}$, (see Fig. 6.3), that is,

$$B_{0,ij} = \begin{cases} B_{ij} - T_0, & \text{if } B_{ij} > T_0 \\ 0, & \text{if } B_{ij} \le T_0 \end{cases}$$

$$B_{k,ij} = \begin{cases} T_{k-1} - T_k, & \text{if } B_{ij} > T_{k-1} \\ B_{ij} - T_k, & \text{if } T_{k-1} > B_{ij} > T_k \\ 0, & \text{if } T_k \ge B_{ij} \end{cases} \qquad k = 1, 2, ...M \tag{6.2.1}$$

where

$$T_0 > T_1 > ... > T_{k-1} > T_k > ... > T_M = 0$$

In other words, $B_{k,ij}$ indicates the number of bits assigned to coefficient (i, j) at stage k.

We can guarantee that the coefficients selected at each stage have the same number of bits assigned by the bit allocation plane, by appropriately choosing the thresholds as follows:

1) Re-order the values on the optimal bit allocation map, $\{B_{ij}\}$, from high to low, as $\{B_k\}$ where $B_k > B_{k+1}$. Repeated values are removed.

2) Choose the corresponding set of thresholds $\{T_k\}$ such that $T_k = B_{k+1}$.

This assures that the case of $T_{k-1} > B_{ij} > T_k$ in equation (6.2.1.) never happens and, therefore, guarantees that all the values on each bit allocation plane are the same. It should be emphasized that the same number of bits on each bit allocation plane implies that the corresponding chosen coefficients (or residual error coefficients) have the same variances at each stage. It is shown experimentally below that vector quantization on the coefficients with the similar variances results in approximately the same error variances in coefficient quantization. If there are no repeated values, then the number of bit allocation planes must be equal to the transform block size. Thus, in practice, the set of thresholds, $\{T_k\}$, should be chosen so that the numbers of bits on each bit plane are only almost the same rather than being exactly equal. A second constraint is that the sum of the bits on each bit plane k, $\sum_{i,j} B_{k,ij}$, should be reasonably small, otherwise the codebook size becomes too large.

Fig. 6.2 shows an ideal example of optimal bit allocation map where the values of $\{B_{ij}\}$ along the lines of equal sum of indices $(i+j)$ are the same. The following thresholds for the optimal bit allocation map in Fig. 6.2 are used (Fig. 6.3)

2.500, 1.250, 0.750, 0.375, 0.000.

The corresponding bit planes are shown in Fig. 6.4. The number of coefficients assigned a non-zero number of bits at each stage is, respectively, 1, 3, 6, 10. Note that the coefficients assigned a non-zero number of bits at lower planes are still assigned a non-zero number of bits at higher planes; implying that vector dimension increases progressively. For each bit plane, the numbers of bits are the same; that is, 2.5 at level 0, 1.25 at level 1, 0.75 at level 2 and 0.375 at level 3. The corresponding sums of bits on bit allocation planes are, respectively, 2.5 for level 0 and 3.75 for level 1 to 3 which translates into the following number of codewords, 8 at level 0 and 16 at

levels from 1 to 3.

6.3 VECTOR QUANTIZATION

With the set of bit allocation planes, the transform coefficients are vector quantized on a stage-by-stage basis. In other words, the transform coefficients $\{Q_{0,ij}\} = \{Q_{ij}\}$ are first vector quantized with the first bit allocation plane, B_o, the residual error coefficients $\{Q_{1,ij}\}$ are then fedback and vector quantized with the second bit allocation plane, B_1. The same process is repeated until the final bit allocation plane is used. The steps involved in coding the coefficients (or residual error coefficients) at stage k, $\{Q_{k,ij}\}$, are now detailed (see also Fig. 6.1)

1) **Vector Formation:** The vectors of dimensionality $D_v(k)$ are formed from the residual coefficients assigned a non-zero number of bits based upon the bit allocation plane, $B_{k,ij}$.

2). **Training Sequence Selection:** All the vectors formed above are chosen as part of the training sequence.

3). **Codebook Generation:** The codebook is generated by applying the generalized Lloyd clustering algorithm [65] on the training sequence selected in step 2, where the normalized mean square error (NMSE) is used as the criterion for distortion. The number of codewords is equal to $2^{[\sum_{i,j} B_{i,j}]}$, where $[\alpha]$ means the truncation of $\alpha + 0.5$, which implies that the average bit rate for coding a vector at stage k is bounded by

$$\max R_v(k) = \log_2 2^{[\sum_{i,j=0}^{N-1} B_{k,ij}]} = [\sum_{i,j=0}^{N-1} B_{k,ij}] \qquad (bits/vector)$$

if an equal length coder is used for the labels.

4). **Codebook Encoding:** The resulting codebook obtained is scalar quantized and encoded by a variable length coder exploiting the nonuniform distribution of the component values of the codewords;

5). **Vector Quantization:** For each vector, the closest codeword in the codebook is determined and the corresponding label of this codeword is then transmitted using a variable length coder.

6). **Feedback:** The $k+1$th residual coefficients, $\{Q_{k+1,ij}\}$, are formed by taking the difference between the input and output, $Q_{k,ij} - \hat{Q}_{k,ij}$ and fed to the next stage.

7). **Entropy Coding:** Finally, an entropy coder, such as Huffman coder, can be used to losslessly encode the final residual error image.

At the receiving end, the labels are decoded with the received codebooks and the compressed image is reconstructed by performing the inverse transform.

6.4 REMARKS

As mentioned above, the set of optimal bit allocation planes (6.2.1) is the key for the entire procedure. The following remarks with regard to the set of optimal bit allocation planes can be made:

1). For a fixed bit rate, there exists a set of optimal bit allocation planes (6.2.1) corresponding to an optimal bit allocation map where the number of bits assigned to each coefficient is proportional to the coefficient variance (2.4.18). The coefficients are quantized on a stage-by-stage basis by using the set of optimal bit allocation planes.

2). The bit allocation planes, $\{B_{k,ij}\}$, determine the number of bits assigned to the corresponding coefficients;

3). The thresholds, $\{T_k\}$, can be chosen so that the number of bits assigned at each stage are equal. This means that the coefficients (or residual error coefficients) chosen at each stage have similar variances. The simulation results shown below demonstrate that vector quantization of coefficients

(or residual error coefficients) with similar variances results in similar error variances in coefficient (or residual error coefficient) quantization.

4). The numbers of bits assigned to the coefficients on the bit allocation planes, $T_{k-1} - T_k$ (if $B_{ij} > T_{k-1}$) or $B_{ij} - T_k$ (if $T_{k-1} > B_{ij} \geq T_k$), can be any positive real values while in the case of scalar quantization, only positive integer figures are valid because of the constraint of practical quantizers for coefficient quantization.

5). The sum of the bit allocation planes is exactly equal to the optimal bit allocation map,

$$B_{ij} = \sum_k B_{k,ij} \qquad i,j = 0,1,...N-1 \qquad (6.3.1)$$

which can be shown as follows. For $B_{ij} > T_0$, from equation (6.2.1),

$$\begin{cases} B_{0,ij} = B_{ij} - T_0, \\ B_{k,ij} = T_{k-1} - T_k. \qquad k = 1,2,...M \end{cases}$$

Therefore,

$$\sum_{k=0}^{M} B_{k,ij} = B_{ij} - T_0 + T_0 -, ...T_{k-1} - T_k +, ... + T_M = B_{ij}.$$

For $T_{K-1} > B_{ij} > T_K$, from equation (6.2.1),

$$\begin{cases} B_{k,ij} = 0, \qquad k = 0,1,...K-1 \\ B_{k,ij} = B_{ij} - T_K, \\ B_{k,ij} = T_{k-1} - T_k. \qquad k = K+1, K+2, ...M \end{cases}$$

Therefore,

$$\sum_{k=0}^{M} B_{k,ij} = 0 + 0, ... + B_{ij} - T_K + T_K - T_{K+1} +, ... + T_M = B_{ij}.$$

For $B_{ij} = 0$, since, from equation (6.2.1),

$$B_{k,ij} = 0, \qquad k = 0, 1, ...M$$

$$\sum_{k=0}^{M} B_{k,ij} = 0.$$

This implies that the coefficients are, in fact, assigned the same number of bits as in the optimal bit allocation map (2.4.18). In other words, the set of optimal bit allocation planes (6.2.1) and the optimal bit allocation map (2.4.18) are equivalent in terms of bit distribution over the coefficients.

6). As the stage number increases, more and more of the higher order coefficients with lower variances are included as components of the vectors. This implies that the dimensionality of the vectors increases with the stage number, because all the previous coefficients are still included as components.

7). We note that the proposed technique is performed on the coefficients in a multi-stage manner. At each stage, the coefficients (or residual error coefficients), $\{Q_{k,ij}\}$, are quantized using a bit allocation plane and the quantized coefficients (or residual error coefficients), $\{\hat{Q}_{k,ij}\}$, is then transmitted. The information transmitted up to a certain stage k, $\sum_k \hat{Q}_{k,ij} = \hat{Q}_{ij}$, in fact corresponds to an approximation of the original image. Therefore, the proposed technique results in progressive transmission of the image.

6.5 SIMULATIONS

Computer simulations were carried out by using the multistage vector quantization with optimal bit allocation planes (MVQ-OBA) on the face image, shown in Fig. 6.5, of size 256 × 256 pixels with 8 bits/pixel.

The test face image is decomposed into a set of sub-blocks of sizes $N \times N$. (Two values of N are used: 4 and 8.) The discrete cosine transform is used as the orthogonal transform (Fig. 6.1) because it comes closest to the Karhunen-Loeve transform in

terms of energy compaction [46]. The variances of the coefficients are estimated by,

$$\begin{cases} \sigma_{ij}^2 = \dfrac{N \times N}{256 \times 256} \displaystyle\sum_{u,v=0}^{256/N} (Q_{ij}^{uv} - m_{ij})^2 \\ m_{ij} = \dfrac{N \times N}{25 \times 256} \displaystyle\sum_{u,v=0}^{256/N} Q_{ij}^{uv} \end{cases} \tag{6.5.2}$$

where N is the transform block size and u, v and i, j are, respectively, the indices of the blocks and the pixels within the block. The estimated variances for the two block sizes are shown in Fig. 6.6 and 6.7, respectively. The corresponding optimal bit allocation maps $\{B_{ij}\}$ are obtained from equation (2.4.18) and are shown in Fig. 6.8 and Fig. 6.9. Note that all the negative values on the optimal bit allocation map (2.4.18) are set to 0, implying no transmission of the corresponding coefficients.

The thresholds $\{T_k\}$ are chosen such that the sum of the bits, for each bit allocation plane, is between 2 to 7 which results in a reasonable codebook size for each stage, i.e. the number of codewords ranges from 4 to 128. The set of threshold values, $\{T_k\}$, is {3.32, 1.99, 1.20, 0.39, 0.00} for the 4×4 block case and {4.13, 2.50, 1.93, 1.47, 0.97, 0.69, 0.42, 0.23, 0.00} for the 8×8 block case. The corresponding set of optimal bit allocation planes is then obtained by slicing the optimal bit allocation maps shown in Fig. 6.8 and Fig. 6.9 with the sets of thresholds $\{T_k\}$, as illustrated in Fig. 6.10 and 6.11. As we can see, if the bit allocation planes (Fig. 6.10 or 6.11) are summed, coefficient by coefficient, then, the optimal bit allocation map (Fig. 6.8 or 6.9) results. We also note that the number of bits allocated to each coefficient at each plane is approximately equal and is always a positive real value. At each stage k, the coefficients (or residual error coefficients) assigned non-zero values are taken as the components of the vector and (vector) quantized. A separate codebook is generated by the generalized Lloyd clustering algorithm with the number of bits equal to the sum of the bits on the corresponding bit allocation plane. The quantization error variances of the coefficients (or residual error coefficients) at each stage, are, respectively, illustrated in Fig. 6.12 and 6.13 for the two block sizes.

From Fig. 6.10-6.13, it is seen that the coefficients (or residual error coefficients) chosen as the components of the vectors at each stage have similar variances, and vector quantization of these coefficients results in similar quantization error variances.

The simulation results are reported in Tables 6.1-and 6.2, where the components shown are as follows:

$D_v(k)$ – the vector dimension at stage k;

$N_c(k)$ – the number of codewords at stage k;

$R_l(k)$ – the bit rate for coding the labels at stage k, where a variable length coder is used, which depends upon the frequency of occurrence of the labels;

$R_c(k)$ – the bit rate for coding the codebook at stage k, where a variable length coder is also used;

$R_m(k)$ – the bit rate required for indicating the positions of the coefficients added at stage k;

$R_p(k) = \sum_{m=0}^{k}(R_l(m) + R_c(m) + R_m(m))$ – the total progressive bit rate up to stage k; $H_e^1(k)$ – the first order entropy of the residual error image at stage k.

$R_t(k) = R_p(K) + H_e^1(K)$ – the total bit rate for losslessly coding at stage k (4.K.13).

In evaluating the performance of the proposed scheme, the normalized mean square error (NMSE) between the original image X and the kth approximation $\hat{X}_k$ is employed,

$$\text{NMSE(k)} = \frac{\sum_{i,j=0}^{255}(X_{ij} - \hat{X}_{k,ij})^2}{\sum_{i,j=0}^{255} X_{ij}^2} \times 100\%, \qquad (6.5.1)$$

where X_{ij} and $\hat{X}_{k,ij}$ represent, respectively, the (i,j)th element of the original image and its kth approximation. Note that, for a block of size $= 8 \times 8$, as the progressive bit

rate increases, the overhead required for transmitting the codebooks goes up rapidly. For example, the overhead accounts for only 16% of bit rate at earlier stages (< 3) while it take up more than 57% of the bit rate at latter stages (> 8). An explanation for this behaviour is that at latter stages, the vector dimensions is much higher which leads to the larger number of bits required for transmitting the codebooks (see equation (2.5.12)). This implies that at higher stages, the bits are less efficiently used to transmit the image information.

The values of the progressive bit rate, $R_p(k)$, and the distortion, NMSE(k), shown in Tables 6.1 and 6.2, are plotted in Fig. 6.14. Note that the larger transform block size (8×8) achieves better performance than the smaller block size (4×4). However, at higher bit rate (> 0.8 bits/pixel), the two rate distortion curves begin to converge, this can probably be traced to the larger overhead required for codebook coding for the 8×8 block size.

The images reconstructed after the zeroth, first, third, fifth, seventh and eighth stages by using MVQ-OBA with a block size of (8×8) are shown in Fig. 6.15. The images are approximations, improving progressively with increasing stage number. Note that although the first two approximations are very rough, they have indeed provided the gross information of image at a very low bit rate of 0.0305 bits/pixel and 0.1115 bits/pixel, respectively. The successive approximations rapidly reach subjectively good quality. For example, the detail around the eyes and lip are significantly improved from the 3rd to 5th approximation and there is no apparent perceptible impairment by the 7th approximation at a bit rate of 0.7431 bits/pixel (see Fig. 6.15).

6.6 COMPARATIVE PERFORMANCES

Single Stage Coding

For comparative purpose, the following transform coding techniques with a block of size 8×8 have also been applied to the face image.

1) **Non-adaptive cosine transform coding (TC)** [42-49]: The coefficients are simply scalar quantized with the optimal bit allocation map from equation (2.4.18);

2) **Adaptive cosine transform coding (ATC)** [62]: The transform blocks are first partitioned into four classes according to their activity indices. For each class, an optimal bit allocation map is then found based upon the variances of the coefficients within the class. A larger number of bits is allocated to the blocks with the higher activity indices. The coefficients are then scalar quantized.

3) **Simple vector quantization (SVQ)**: The transform blocks are simply taken as the vectors and vector quantized with the number of codewords ranging from 32 to 256.

4) **Vector quantization with coefficient selection (CVQ)** [83,88]: The coefficients with the highest variances are chosen and taken as the components of the vectors. A codebook, with the number of codewords ranging from 64 to 512, is then found and used for vector quantization

It should be emphasized that for scalar quantization (TC and ATC), the optimal bit allocation map obtained by using equation (2.4.18) is rounded off, that is, the number of bits on the optimal bit allocation map are rounded into positive integer values. Thus, there is a loss of optimality. The simulation results are shown in Tables 6.3, 6.4, 6.5 and 6.6, where the following notation is adopted:

Scalar Quantization (TC and ATC):

R_{co} – the bit rate for coding the coefficients;

R_o, – the overhead including the bit rates for coding the bit allocation maps and the classification map (only for ATC);

$R_p = R_{co} + R_o$ is the total bit rate.

Vector Quantization (SVQ and CVQ):

D_v – the vector dimension (only for CVQ);

N_c – the number of codewords in the codebook;

R_l – the bit rate for encoding the labels where a variable length coder is used;

R_c – the bit rate for transmitting the codebook where a variable length coder is also used;

R_m – the bit rate for indicating the positions of the selected coefficients (only for CVQ);

$R_p = R_l + R_c + R_m$ is the total bit rate.

The rate distortion curves for the different techniques are shown in Fig. 6.16 and 6.17 for scalar and vector quantization, respectively. It is clear, from Fig. 6.16, that MVQ-OBA consistently outperforms scalar quantization, both TC and ATC. For example, at a bit rate of about 0.25 bits/pixel, there is an improvement using MVQ-OBA, respectively, of 21% ($\frac{0.3398-0.2684}{0.3398} \times 100\%$) for ATC and of 43% ($\frac{0.4732-0.2684}{0.4732} \times 100\%$) for TC. At a bit rate of 1.0 bits/pixel, the corresponding improvement over ATC and TC are, respectively, 32% ($\frac{0.0932-0.0630}{0.0932} \times 100\%$) and 64% ($\frac{0.1752-0.0630}{0.1752} \times 100\%$). Fig. 6.16 demonstrates the comparison of the three vector quantization techniques in the transform domain: SVQ, CVQ and MVQ-OBA. SVQ results in the poorest performance as the vectors are simply formed from the transform blocks and vector quantized. Vector quantization with coefficient selection (CVQ) outperforms SVQ, as a smaller overhead for transmitting the codebook is required, while MVQ-OBA achieves the best overall performance. Finally, we note that the error variances in coefficient quantization resulting from using the direct vector quantization techniques (SVQ with $N_c = 64$ and CVQ with $N_c = 128$) on the coefficients are quite dispersed (Fig. 6.18 and 6.19)

Progressive Image Transmission

MVQ-OBA is now compared with the following two progressive image transmission techniques. 1) PCT (8×8): The coefficients are progressively transmitted, one by one, from low to high order [12,13]. The first order entropies of the coefficients for the face image are shown in Fig. 6.20, which clearly demonstrates that the entropies of coefficients are monotonically decreasing from low to high order. 2) RVQ-TRA (8×8): Vector quantization is applied to the coefficients in a stage-by-stage manner [21]. Note that at each stage, all the coefficients are taken as the components of vectors and vector quantized. The resulting rate distortion curves are illustrated in Fig. 6.21, where it is seen that MVQ-OBA outperforms the other techniques.

The images reconstructed by using ATC with a block size of (8×8) at bit rates of 0.5006 bits/pixel and 1.0075 bits/pixel are provided in Fig. 6.22 for subjective comparison. From Fig. 6.15 and 6.22, it can be seen that at a bit rate of around 0.5 bits/pixel, MVQ-OBA achieves significantly better subjective image quality. At a bit rate of around 1.0 bits/pixel, both MVQ-OBA and ATC provide the reconstructed images with excellent subjective quality, however, MVQ-OBA gives rise to less block effect, especially, in the areas around the lip.

The error images by using MVQ-OBA and ATC with a block size of 8×8 are shown in Fig. 6.23, where the left column images are obtained by using MVQ-OBA and the right column by ATC, and the images at the top are coded at a bit rate of about 0.5 bits/pixel and at the bottom at about 1.0 bits/pixel. By comparing these error images, we find that the error images when using MVQ-OBA (the left column of Fig. 6.23) are almost uniform, while there are apparent block effects in some areas of the error images when using ATC (the right column of Fig. 6.23); such as, the areas around the lip, nose, eyes, and the portions on the edge of the face.

Before concluding this section, we compare MVQ-OBA with (RVQ-SPA) (chapter 4). Fig. 6.24 shows the rate distortion curves for both the techniques with a block

size of 4×4 (Tables 4.1b and 6.2). It is clear that the MVQ-OBA outperforms the RVQ-SPA.

6.7 CONCLUSIONS

We have presented a vector quantization technique in transform domain. The technique is performed on a stage-by-stage basis, using a set of optimal bit allocation planes where the effective number of bits assigned to each coefficient is proportional to the coefficient variance. At each stage, all the coefficients assigned a non-zero number of bits are combined into a vector and (vector) quantized. The computer simulation results demonstrate that the proposed transform vector quantization technique, as a coding technique, outperforms transform coding using scalar quantization and various forms of vector quantization. The scheme can be used for progressive transmission and is seen to outperform other progressive transmission techniques [12,13,18,21].

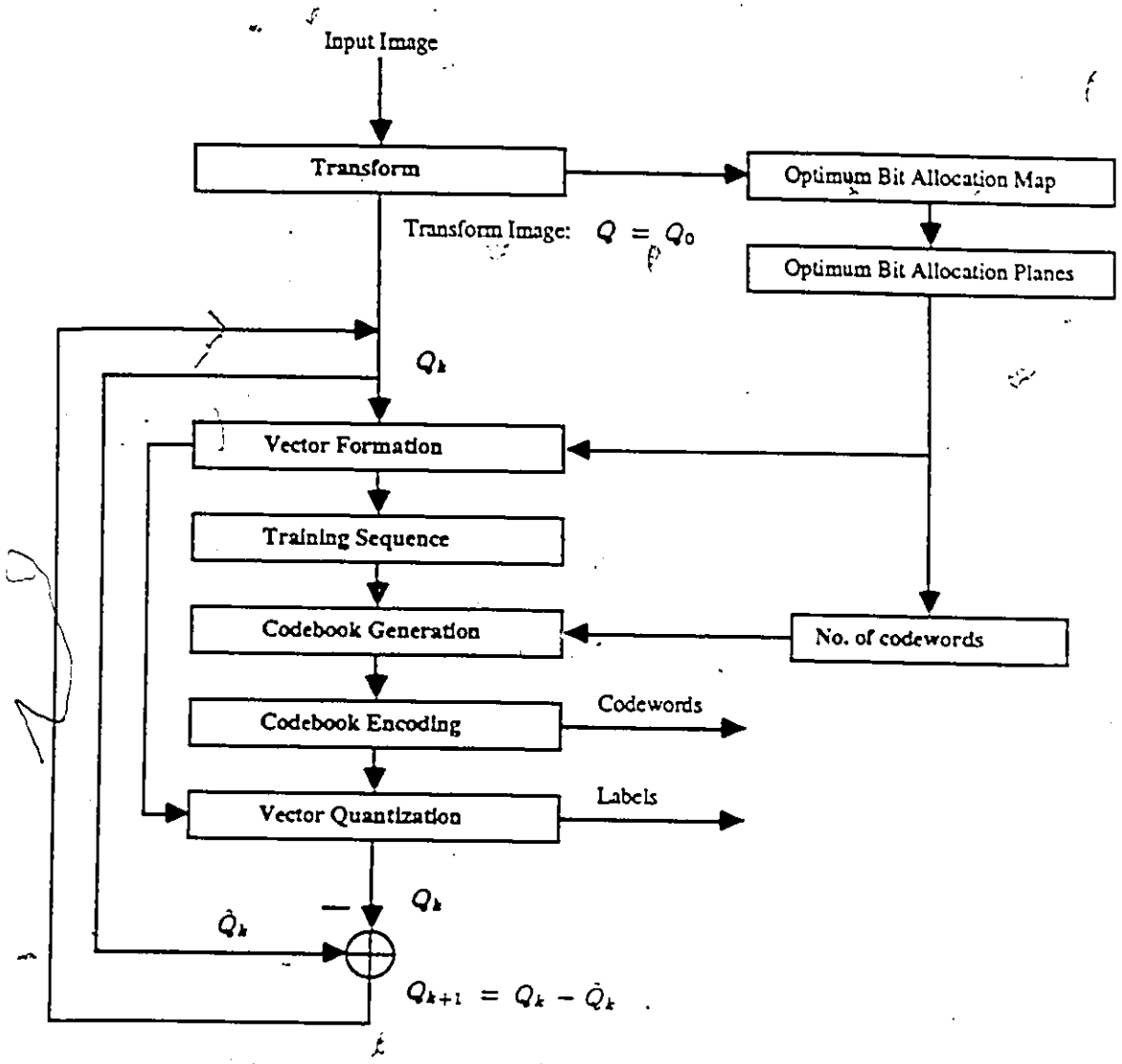


Fig. 6.1. Multistage vector quantization with optimum bit allocation planes. The input image first undergoes an orthogonal transform. An optimal bit allocation map is then found based upon the variances of the coefficients. A set of bit allocation planes are then obtained by applying a set of thresholds to the optimal bit allocation map. With the set of bit allocation planes, the coefficients are vector quantized on a stage-by-stage basis. In other words, the coefficients are first quantized with bit allocation plane 0; the residual error coefficients are then requantized with bit allocation plane 1. The same procedure is repeated until the last bit allocation plane is used.

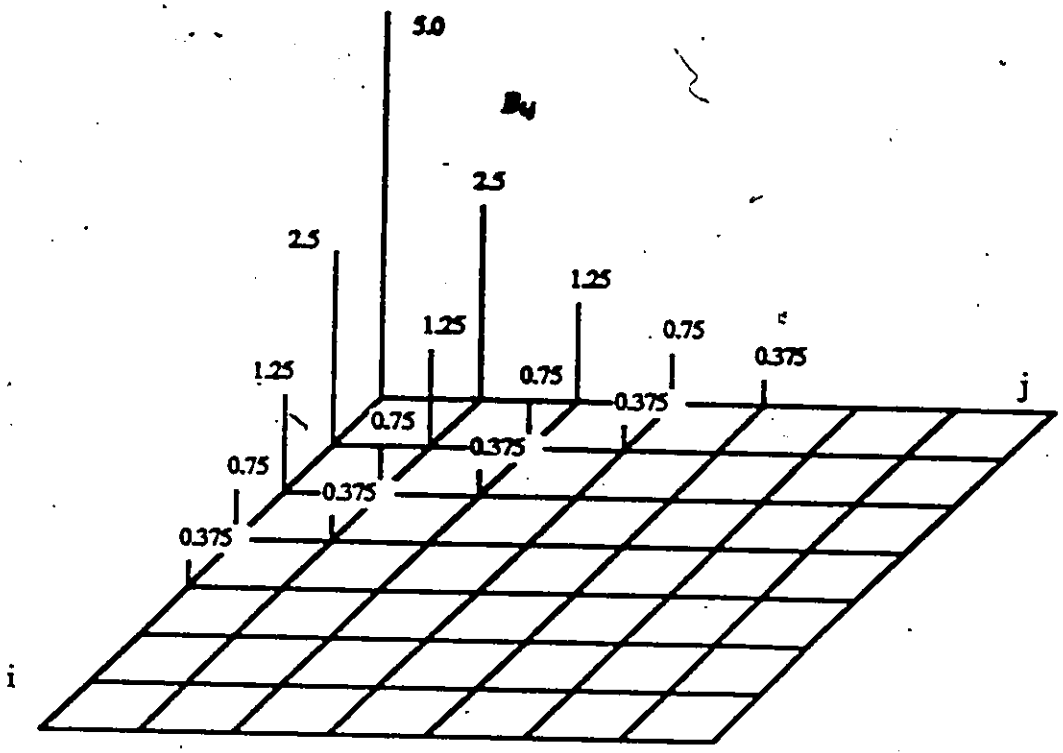


Fig. 6.2. An example of an optimal bit allocation map. Note that the number of bits assigned to the coefficients vary significantly.

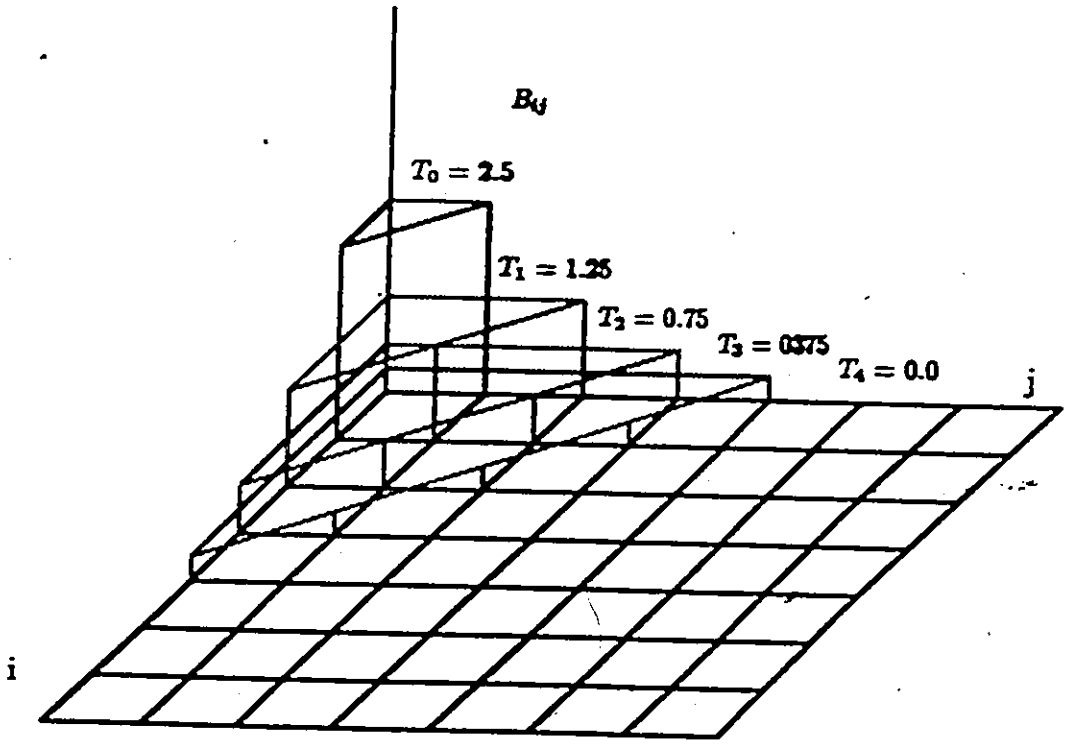


Fig. 6.3. The optimal bit allocation map is sliced by applying a set of thresholds (2.5, 1.25, 0.75, 0.375, 0.0) to yield a set of optimal bit allocation planes.

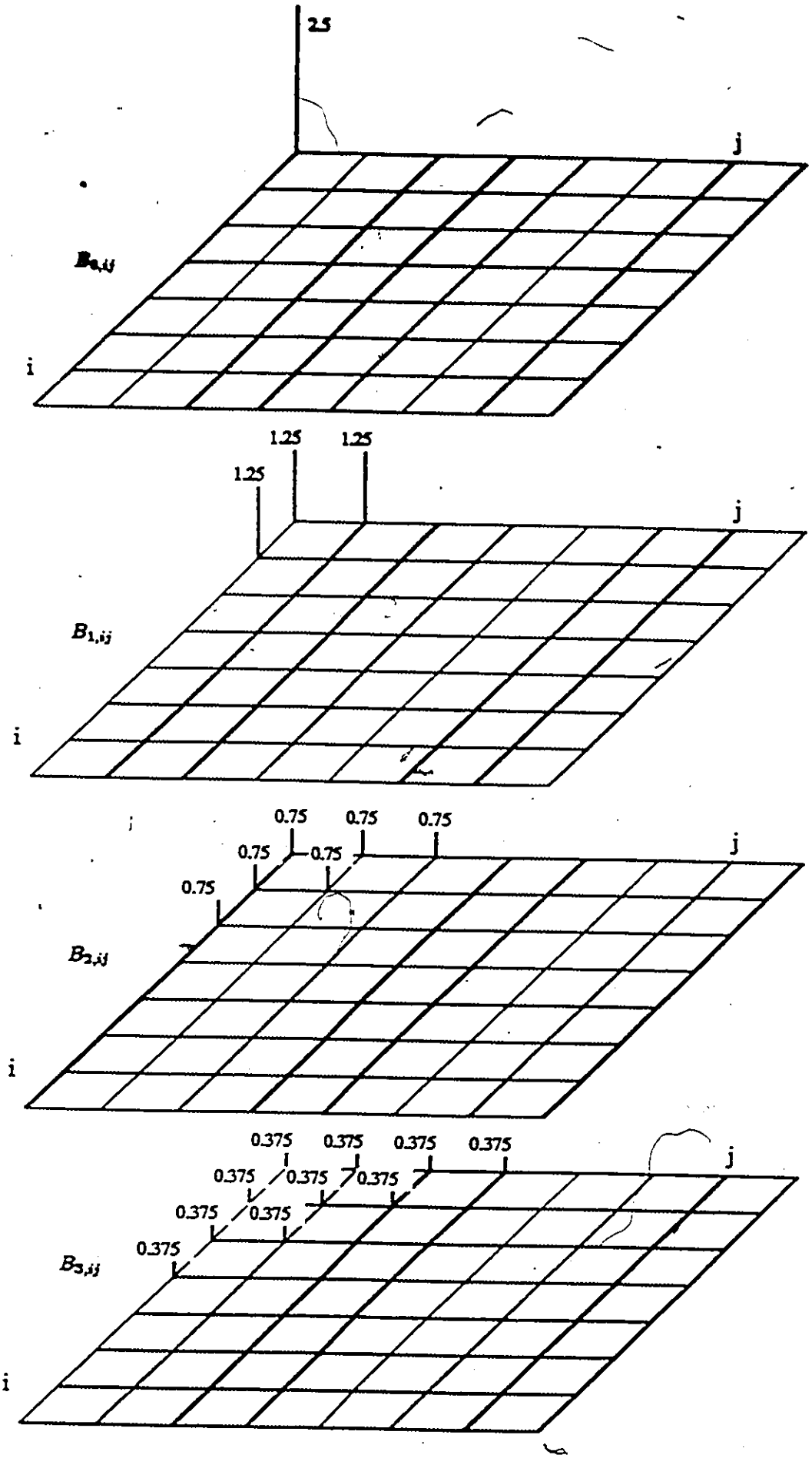


Fig. 6.4. The set of optimal bit allocation planes.

Fig. 6.5. The face image of 256 × 256 pixel with 8 bits/pixel.

46251.3	808.4	132.1	43.9
364.8	75.0	29.4	23.6
35.2	14.4	9.7	11.3
4.0	3.3	4.2	8.4

Fig. 6.6. The transform coefficient variances for a block size of 4 × 4.

176776.6	5947.6	1437.1	619.3	186.9	92.9	50.6	40.1
3307.7	603.8	281.5	122.4	70.4	57.2	34.7	26.3
512.8	185.9	118.0	58.4	40.4	31.8	26.9	27.2
149.2	74.0	38.4	24.2	16.7	17.0	17.0	19.4
32.9	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

Fig. 6.7. The transform coefficient variances for a block size of 8 × 8.

6.22	3.32	1.99	1.20
2.73	1.58	0.91	0.75
1.04	0.39	0.11	0.22
0.00	0.00	0.00	0.00

Fig. 6.8. An optimal bit allocation map for a block size of 4×4.

6.58	4.13	3.11	2.50	1.64	1.13	0.69	0.52
3.71	2.48	1.93	1.30	0.93	0.78	0.42	0.22
2.36	1.63	1.30	0.80	0.53	0.36	0.23	0.25
1.47	0.97	0.49	0.16	0.00	0.00	0.00	0.00
0.38	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Fig. 6.9. An optimal bit allocation map for a block size of 8×8.

2.92 0.00 0.00 0.00
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 $B_{0,ij}$
0.00 0.00 0.00 0.00

1.31 1.31 0.00 0.00
0.74 0.00 0.00 0.00 $B_{1,ij}$
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00

0.79 0.79 0.79 0.79
0.79 0.38 0.00 0.00 $B_{2,ij}$
0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00

0.81 0.81 0.81 0.81
0.81 0.81 0.52 0.36
0.65 0.00 0.00 0.00 $B_{3,ij}$
0.00 0.00 0.00 0.00

0.39 0.39 0.39 0.39
0.39 0.39 0.39 0.39
0.39 0.39 0.11 0.22 $B_{4,ij}$
0.00 0.00 0.00 0.00

Fig. 6.10. A set of optimal bit allocation planes for a block size of 4×4 obtained by applying a set of thresholds (3.32, 1.99, 1.20, 0.39, 0.00) to the optimal bit allocation map in Fig. 6.8.

$B_{0,ij}$

2.45 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$B_{1,ij}$

1.63 1.63 0.61 0.00 0.00 0.00 0.00 0.00
1.21 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$B_{2,ij}$

0.57 0.57 0.57 0.57 0.00 0.00 0.00 0.00
0.57 0.55 0.00 0.00 0.00 0.00 0.00 0.00
0.43 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$B_{3,ij}$

0.46 0.46 0.46 0.46 0.17 0.00 0.00 0.00
0.46 0.46 0.46 0.00 0.00 0.00 0.00 0.00
0.46 0.16 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$B_{4,ij}$

0.50 0.50 0.50 0.50 0.50 0.16 0.00 0.00
0.50 0.50 0.50 0.33 0.00 0.00 0.00 0.00
0.50 0.50 0.33 0.00 0.00 0.00 0.00 0.00
0.50 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$B_{5,ij}$

0.28 0.28 0.28 0.28 0.28 0.28 0.00 0.00
0.28 0.28 0.28 0.28 0.24 0.09 0.00 0.00
0.28 0.28 0.28 0.11 0.00 0.00 0.00 0.00
0.28 0.28 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$B_{6,ij}$

0.27 0.27 0.27 0.27 0.27 0.27 0.27 0.10
0.27 0.27 0.27 0.27 0.27 0.27 0.00 0.00
0.27 0.27 0.27 0.27 0.11 0.00 0.00 0.00
0.27 0.27 0.07 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$B_{7,ij}$

0.19 0.19 0.19 0.19 0.19 0.19 0.19 0.19
0 19 0.19 0.19 0.19 0.19 0.19 0.19 0.00
0.19 0.19 0.19 0.19 0.19 0.13 0.00 0.02
0.19 0.19 0.19 0.00 0.00 0.00 0.00 0.00
0.15 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

$B_{8,ij}$

0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.23
0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.22
0.23 0.23 0.23 0.23 0.23 0.23 0.23 0.23
0.23 0.23 0.23 0.16 0.00 0.00 0.00 0.00
0.23 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Fig. 6.11. A set of optimal bit allocation planes for a block size of 8×8 obtained by applying a set of thresholds (4.13, 2.50, 1.93, 1.47, 0.97, 0.69, 0.42, 0.23, 0.00) to the optimal bit allocation map in Fig. 6.9.

737.7	808.4	132.1	43.9	
364.8	75.0	29.4	23.6	
35.2	14.4	9.7	11.3	The variances of the coefficients after stage 0
4.0	3.3	4.2	8.4	
200.7	232.2	132.1	43.9	
148.6	75.0	29.4	23.6	
35.2	14.4	9.7	11.3	The variances of the coefficients after stage 1
4.0	3.3	4.2	8.4	
67.8	60.8	71.3	43.9	
67.8	71.2	29.4	23.6	
35.2	14.4	9.7	11.3	The variances of the coefficients after stage 2
4.0	3.3	4.2	8.4	
22.9	24.0	22.7	22.2	
24.1	24.7	22.6	16.8	
18.8	14.4	9.7	11.3	The variances of the coefficients after stage 3
4.0	3.3	4.2	8.4	
16.0	14.8	14.3	15.8	
15.2	15.0	16.7	14.5	
14.5	13.8	9.2	10.7	The variances of the coefficients after stage 4
4.0	3.3	4.2	8.4	

Fig. 6.12. The quantization error variances of the coefficients (or residual error coefficients) for a block size equal to 4 × 4 using the set of optimal bit allocation planes of Fig. 6.10.

The variances of the coefficients after stage 0.

11859.0	5947.6	1437.1	619.3	186.9	92.9	50.6	40.1
3307.7	603.8	281.5	122.4	70.4	57.2	34.7	26.3
512.8	185.9	118.0	58.4	40.4	31.8	26.9	27.2
149.2	74.0	38.4	24.2	16.7	17.0	17.0	19.4
32.9	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

The variances of the coefficients after stage 1.

696.5	855.1	621.6	619.3	186.9	92.9	50.6	40.1
922.1	603.8	281.5	122.4	70.4	57.2	34.7	26.3
512.8	185.9	118.0	58.4	40.4	31.8	26.9	27.2
149.2	74.0	38.4	24.2	16.7	17.0	17.0	19.4
32.9	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

The variances of the coefficients after stage 2.

357.1	334.0	305.0	459.0	186.9	92.9	50.6	40.1
388.5	359.3	281.5	122.4	70.4	57.2	34.7	26.3
270.4	185.9	118.0	58.4	40.4	31.8	26.9	27.2
149.2	74.0	38.4	24.2	16.7	17.0	17.0	19.4
32.9	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

The variances of the coefficients after stage 3.

221.3	180.9	205.0	188.7	163.8	92.9	50.6	40.1
195.7	211.0	223.7	122.4	70.4	57.2	34.7	26.3
201.6	157.3	118.0	58.4	40.4	31.8	26.9	27.2
149.2	74.0	38.4	24.2	16.7	17.0	17.0	19.4
32.9	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

The variances of the coefficients after stage 4.

97.3	87.6	88.0	80.4	76.6	62.9	50.6	40.1
90.9	79.2	95.2	77.0	70.4	57.2	34.7	26.3
79.7	84.6	64.0	58.4	40.4	31.8	26.9	27.2
81.3	74.0	38.4	24.2	16.7	17.0	17.0	19.4
32.9	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

Fig. 6.13. The quantization error variances of the coefficients (or residual error coefficients) for a block size equal to 8×8 using the set of optimal bit allocation planes of Fig. 6.11.

The variances of the coefficients after stage 5.

50.0	53.4	49.6	57.1	56.7	50.2	50.6	40.1
54.9	54.8	51.3	57.7	49.3	42.6	34.7	26.3
54.9	63.6	49.1	43.7	40.4	31.8	26.9	27.2
53.7	48.7	38.4	24.2	16.7	17.0	17.0	19.4
32.9	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

The variances of the coefficients after stage 6.

39.3	35.8	36.0	38.7	36.6	39.9	32.4	28.2
43.5	34.3	33.0	35.0	39.0	36.2	34.7	26.3
37.5	45.7	38.7	36.9	32.7	31.8	26.9	27.2
33.3	35.1	34.2	24.2	16.7	17.0	17.0	19.4
32.9	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

The variances of the coefficients after stage 7.

28.5	27.6	24.5	30.0	29.8	28.7	27.4	22.8
28.3	27.5	26.5	30.2	29.6	26.7	28.3	26.3
25.9	26.5	27.5	31.0	24.7	26.3	26.9	22.5
26.8	27.7	26.9	24.2	16.7	17.0	17.0	19.4
25.8	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

The variances of the coefficients after stage 8.

15.0	14.6	14.9	16.9	14.4	16.2	15.4	14.6
16.9	14.9	16.6	16.1	15.9	14.5	14.6	14.9
16.3	14.8	14.7	17.2	14.9	13.9	12.9	12.7
14.6	16.5	15.2	13.4	16.7	17.0	17.0	19.4
12.2	15.8	15.2	9.9	9.7	9.8	10.4	14.6
8.6	6.8	5.6	5.0	5.1	5.8	8.2	13.4
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

Fig. 6.13. The quantization error variances of the coefficients (or residual error coefficients) for a block size equal to 8×8 using the set of optimal bit allocation planes of Fig. 6.11.

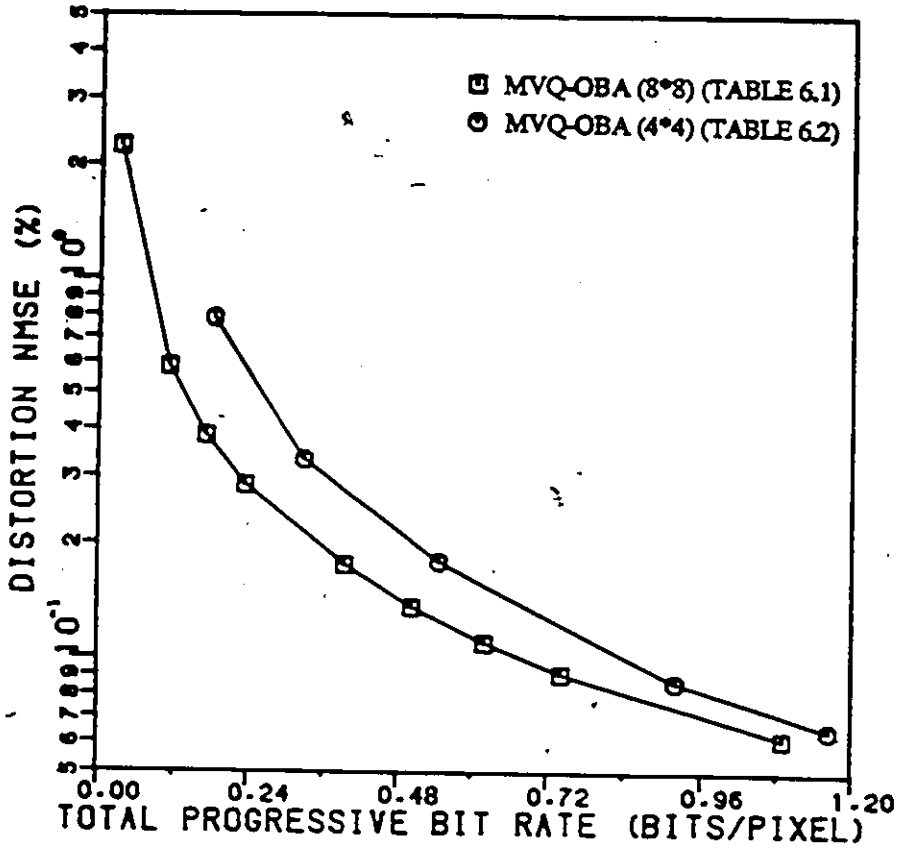


Fig. 6.14. The rate distortion curves using the multistage vector quantization with optimal bit allocation planes (MVQ-OBA). Note that the larger block size (8×8) achieves better performance than the smaller block size (4×4).

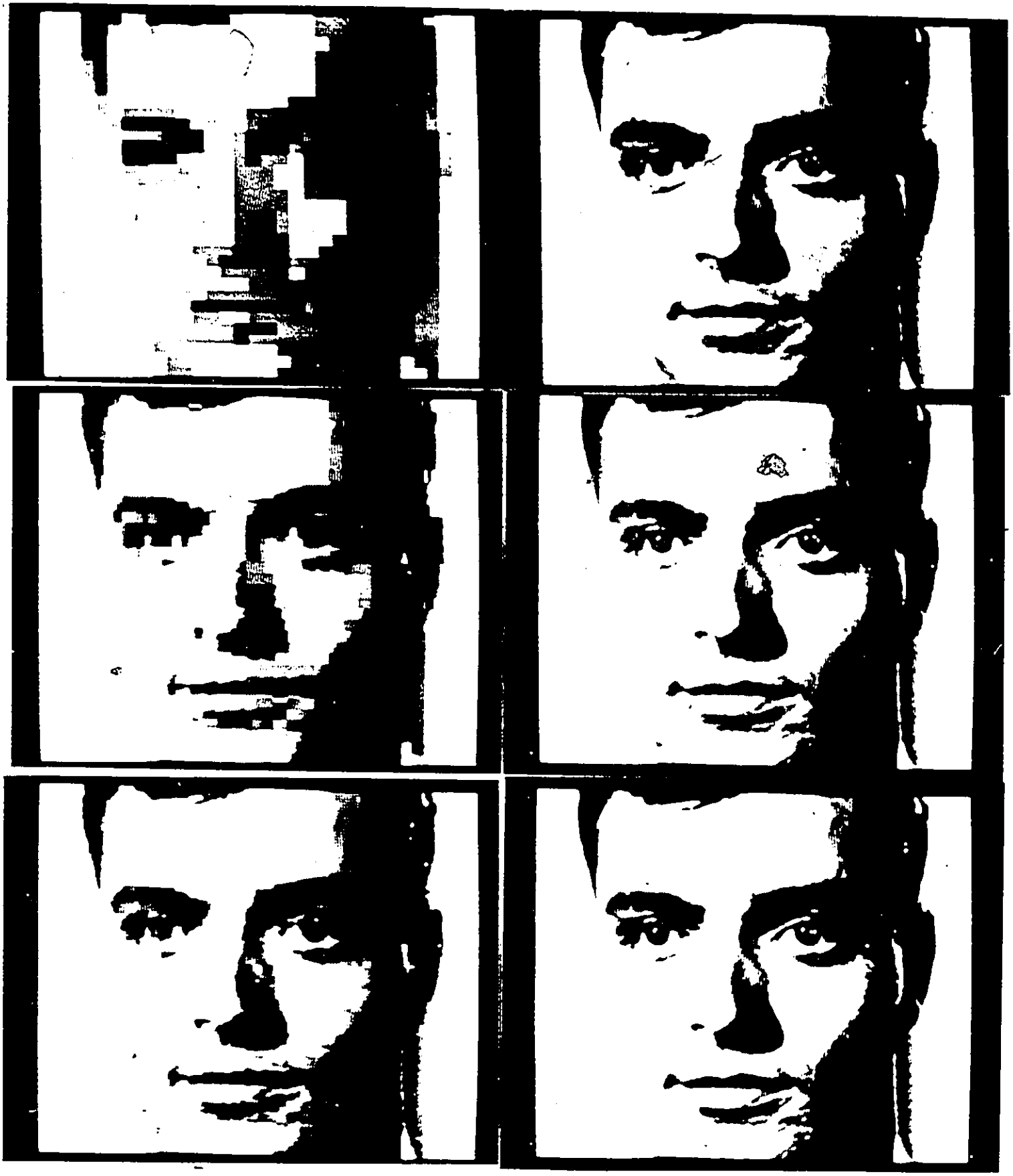


Fig. 6.15. The images reconstructed after stage zero, one, three, five, seven and eight by using MVQ-OBA with a block size of 8×8.

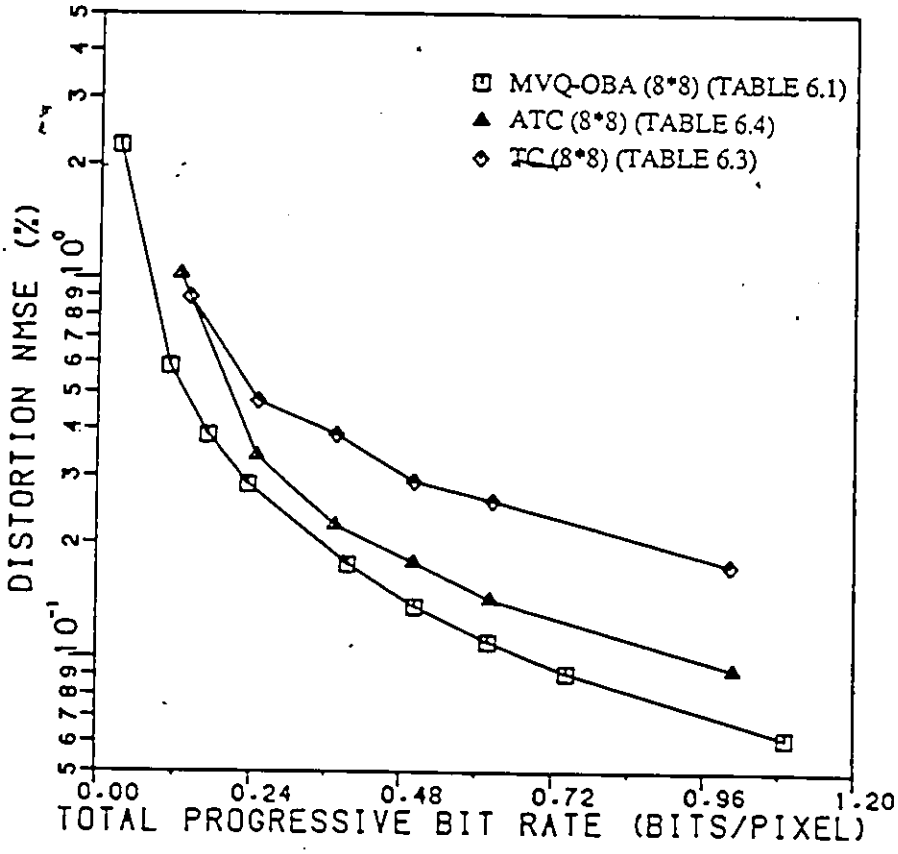


Fig. 6.16. The rate distortion curves for both non-adaptive (TC) and adaptive (ATC) transform coding and MVQ-OBA.

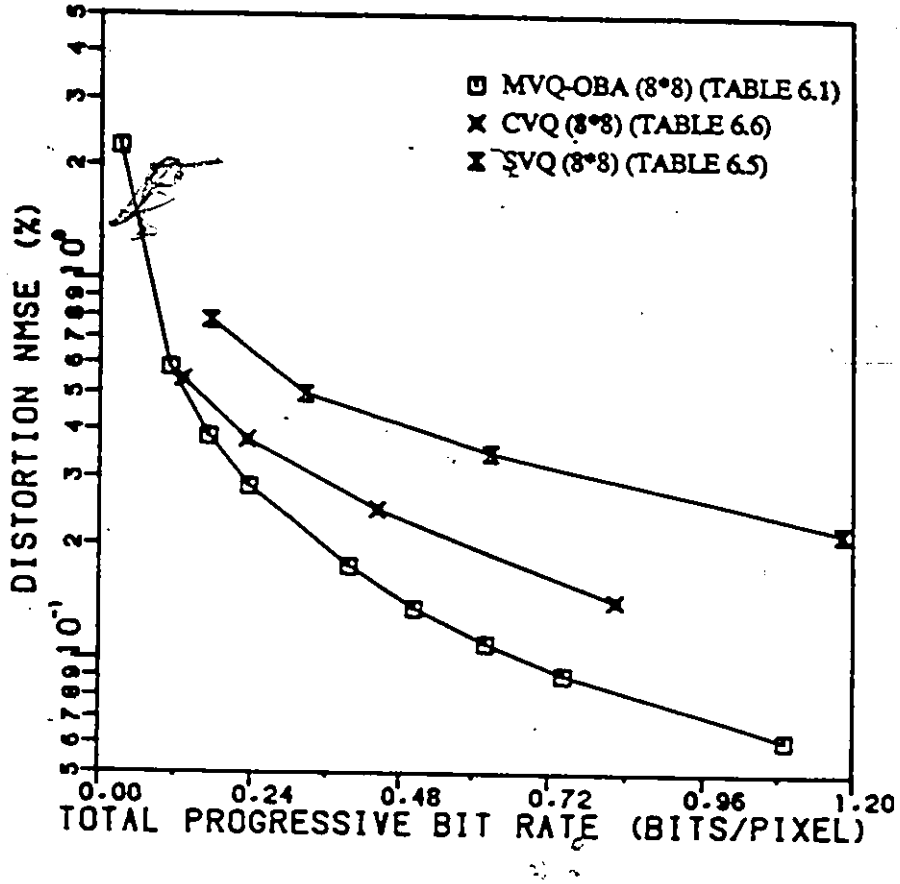


Fig. 6.17. The rate distortion curves for direct vector quantization in the transform domain and MVQ-OBA.

853.5	881.0	565.9	270.0	120.4	63.3	35.6	33.0
866.3	449.6	219.1	94.9	61.5	46.9	27.6	23.7
337.0	147.0	97.5	48.3	34.4	27.8	24.2	24.2
131.0	58.7	33.8	21.9	15.6	15.4	14.9	17.9
28.3	13.9	13.3	9.0	8.9	9.0	9.8	13.9
7.5	6.3	5.2	4.8	4.7	5.6	7.6	12.4
3.0	2.6	2.7	2.7	3.2	4.6	6.6	9.6
1.5	1.5	1.9	2.1	2.7	3.9	6.4	8.7

Fig. 6.18. The quantization error variances of the coefficient using SVQ with the number of codewords equal to 64.

530.6	502.4	328.7	176.6	77.6	93.6	50.8	40.1
548.5	341.3	153.6	122.4	70.4	57.2	34.7	26.3
268.9	120.3	118.0	58.5	40.4	31.9	26.9	27.2
149.2	74.0	38.5	24.2	16.7	17.0	17.0	19.5
32.9	15.8	15.2	9.9	9.8	9.8	10.4	14.7
8.6	6.8	5.6	5.0	5.1	5.9	8.2	13.5
3.3	2.8	2.9	2.8	3.4	4.8	7.0	10.3
1.6	1.6	2.0	2.2	2.8	4.1	6.8	9.2

Fig. 6.19. The quantization error variances of the coefficient using CVQ with the number of codewords equal to 128.

9.2	7.3	6.1	5.5	5.1	4.7	4.5	4.5
6.8	5.9	5.4	5.0	4.7	4.5	4.4	4.3
5.5	5.1	4.9	4.6	4.4	4.2	4.2	4.2
4.8	4.5	4.3	4.1	4.0	3.9	3.9	4.0
4.0	3.7	3.7	3.5	3.6	3.5	3.6	3.9
3.3	3.2	3.2	3.1	3.1	3.3	3.5	3.8
2.8	2.7	2.8	2.7	2.9	3.1	3.4	3.7
2.4	2.3	2.5	2.6	2.7	3.0	3.4	3.6

Fig. 6.20. The first order entropies of the transform coefficients for a block size of 8×8.

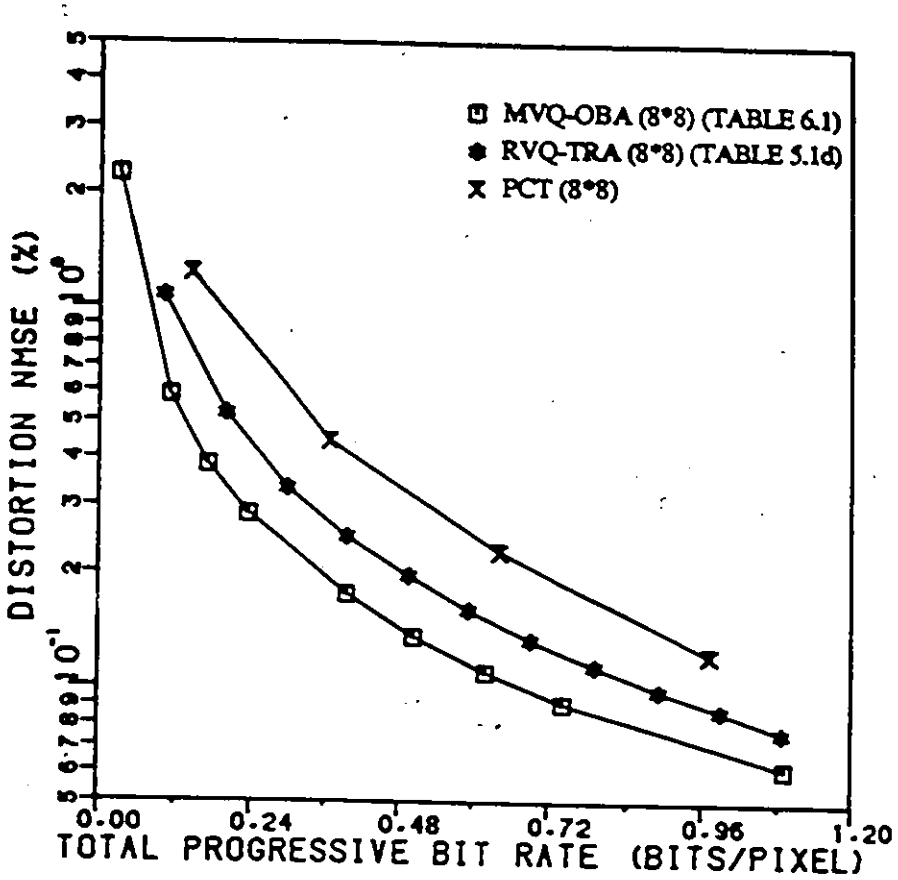


Fig. 6.21. The rate distortion curves for three different progressive image transmission techniques.

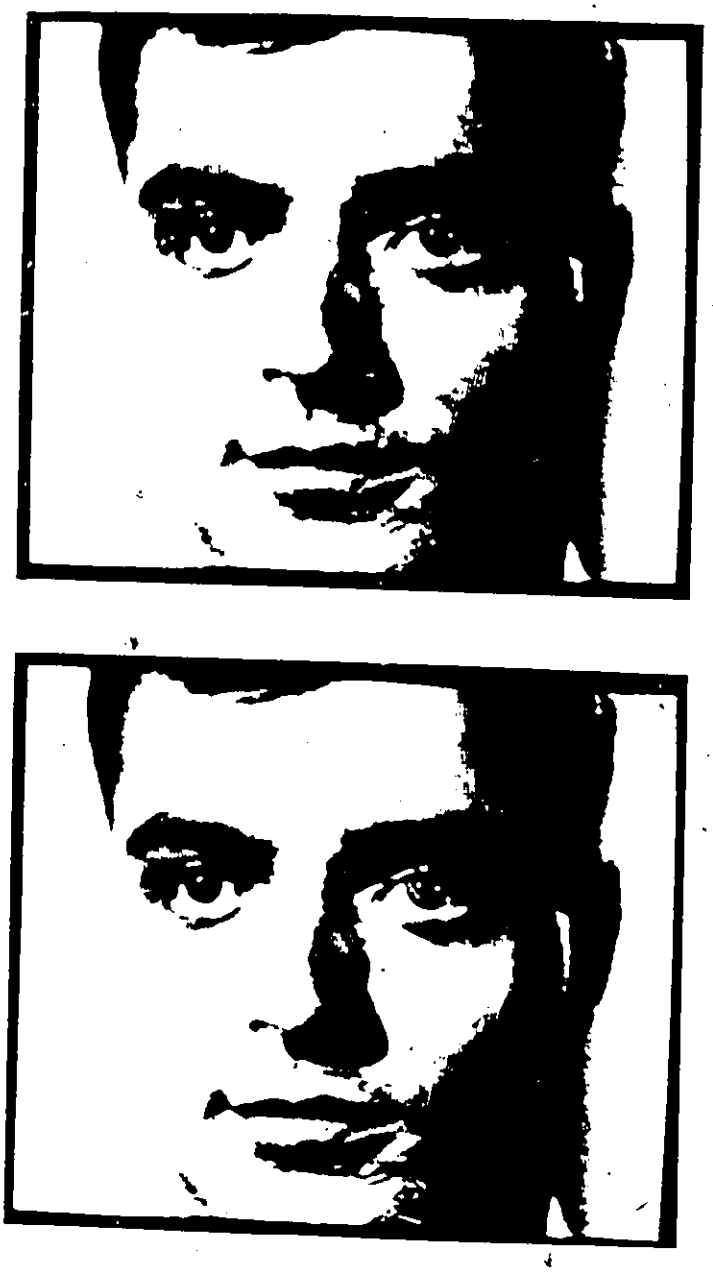


Fig. 6.22. The images reconstructed by using ATC with a block size of 8 × 8 at bit rates of 0.5006 bits/pixel (top) and 1.0075 bits/pixel (bottom).

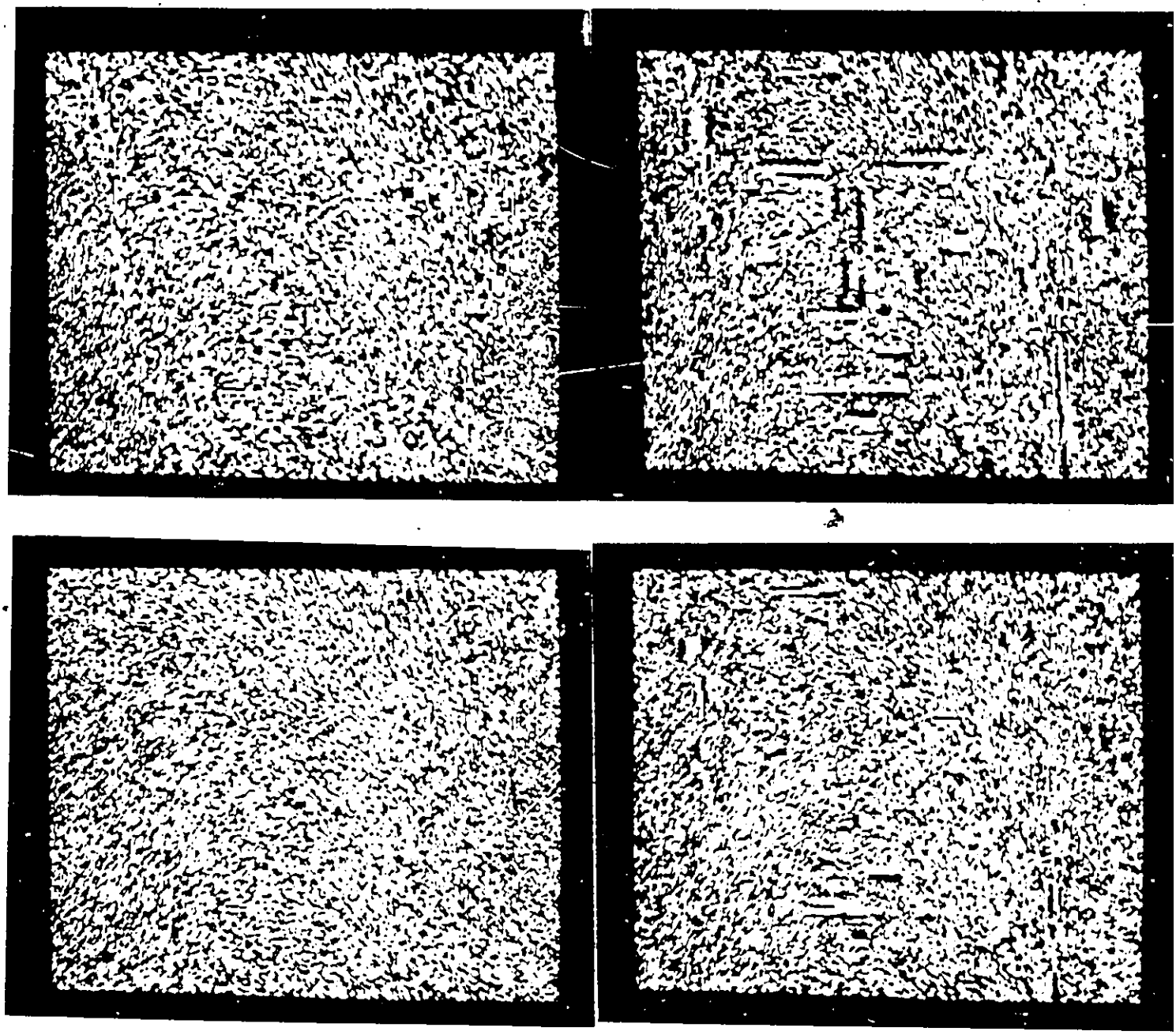


Fig. 6.23. The error images by using MVQ-OBA and ATC with a block size of 8×8, where the left column images are obtained by using MVQ-OBA and the right column by ATC, and the images at the top are coded at a bit rate of about 0.5 bits/pixel and at the bottom at about 1.0913 bits/pixel.

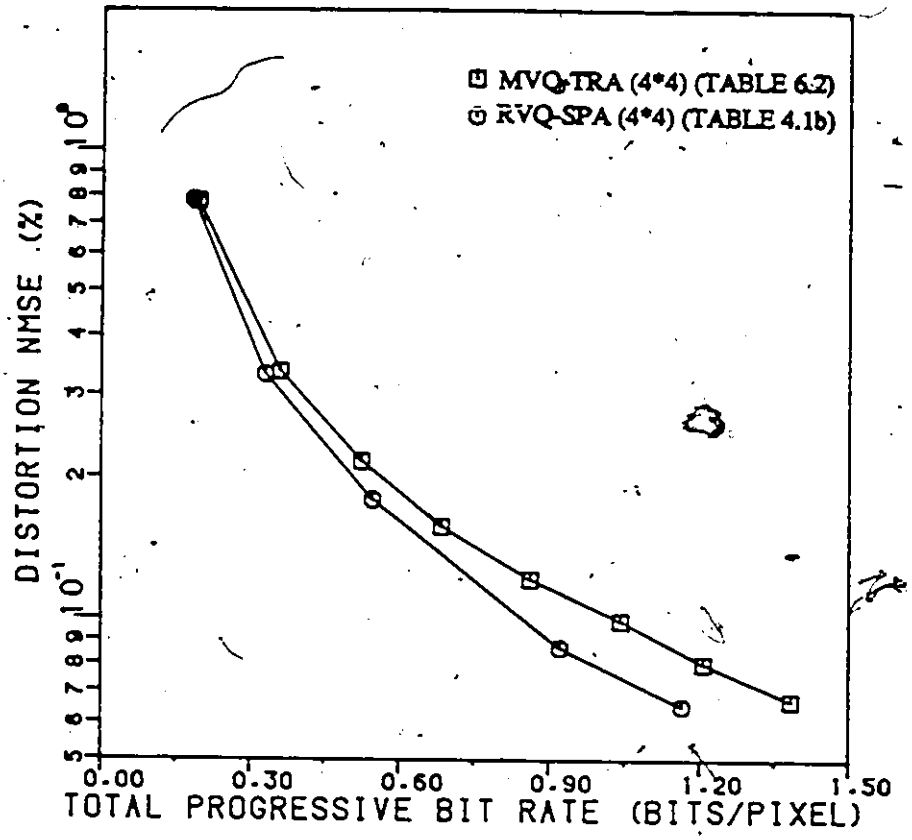


Fig. 6.24. The rate-distortion curves for MVQ-OBA and RVQ-SAP with a block size of 4×4 (from Tables 6.2 and 4.1b).

Table 6.1 Computer simulation results for the face image by using MVQ-OBA with a block size of 8*8

k	Dv(k)	Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)	Rm(k) (bits/pixel)	Rp(k) (bits/pixel)	Hc(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	1	4	0.030416	0.000122	0.000000	0.0305	6.3037	6.3342	2.2391
1	4	32	0.068415	0.012321	0.000275	0.1115	5.1908	5.3024	0.5847
2	7	16	0.050754	0.009306	0.000275	0.1719	4.9039	5.0758	0.3844
3	10	16	0.050217	0.013235	0.000275	0.2356	4.7170	4.9526	0.2846
4	14	64	0.078436	0.080900	0.000366	0.3954	4.4368	4.8321	0.1755
5	18	32	0.063979	0.043783	0.000366	0.5034	4.2816	4.7851	0.1359
6	22	32	0.069076	0.047497	0.000366	0.6204	4.1480	4.7684	0.1098
7	26	32	0.072027	0.050298	0.000366	0.7431	4.0244	4.7675	0.0910
8	29	128	0.101961	0.245951	0.000275	1.0913	3.7761	4.8674	0.0615

Table 6.2 Computer simulation results for the face image by using MVQ-OBA with a block size of 4*4

k	Dv(k)	Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)	Rm(k) (bits/pixel)	Rp(k) (bits/pixel)	Hc(k) (bits/pixel)	Rt(k) (bits/pixel)	NMSE(%)
0	1	8	0.183262	0.000366	0.000000	0.1836	5.4689	5.6525	0.7839
1	3	8	0.144046	0.001441	0.000122	0.3292	4.8512	5.1805	0.3322
2	5	16	0.212209	0.005771	0.000122	0.5473	4.3962	4.9435	0.1793
3	9	64	0.329368	0.044767	0.000244	0.9217	3.9469	4.8686	0.0867
4	12	16	0.232850	0.010063	0.000183	1.1648	3.7470	4.9118	0.0651

Table 6.3 Computer simulation results for the face image by using transform coding (TC) with a block size of 8*8

Rco (bits/pixel)	Ro (bits/pixel)	Rt (bits/pixel)	NMSE(%)
0.140625	0.000421	0.1410	0.8895
0.250000	0.000683	0.2507	0.4732
0.375000	0.000947	0.3759	0.3853
0.500000	0.001192	0.5012	0.2897
0.625000	0.001386	0.6264	0.2577
1.000000	0.001888	1.0019	0.1752

Table 6.4 Computer simulation results for the face image by using adaptive transform coding (ATC) with a block size of 8*8

Rco (bits/pixel)	Ro (bits/pixel)	Rt (bits/pixel)	NMSE(%)
0.093750	0.032255	0.1260	1.0237
0.218750	0.033699	0.2544	0.3391
0.339844	0.034706	0.3746	0.2222
0.464844	0.035754	0.5006	0.1779
0.585938	0.036599	0.6225	0.1424
0.968750	0.038699	1.0075	0.0932

Table 6.5 Computer simulation results for the face image by using simple vector quantization (SVQ) in the transform domain with a block size of 8*8

Nc	Rl (bits/pixel)	Rc (bits/pixel)	Rt (bits/pixel)	NMSE(%)
32	0.071024	0.103060	0.1741	0.7204
64	0.084240	0.240028	0.3242	0.4999
128	0.097987	0.528528	0.6265	0.3497
256	0.111631	1.071370	1.1830	0.2165

Table 6.6 Computer simulation results for the face image by using vector quantization with coefficient selection (CVQ) in the transform domain with a block size of 8*8

Dv	Nc	Rl (bits/pixel)	Rc (bits/pixel)	Rm (bits/pixel)	Rt (bits/pixel)	(NMSE(%)
7	64	0.084265	0.046526	0.000641	0.1314	0.5447
10	128	0.096756	0.134502	0.000916	0.2322	0.3780
14	256	0.112046	0.328468	0.001282	0.4418	0.2480
18	512	0.124768	0.696844	0.001648	0.8233	0.1425

Table 6.7 Computer simulation results for the face image by transmitting the coefficients from the low to high order with a block size of 8*8

Nco	R (bits/pixel)	NMSE(%)
1	0.1447	1.2314
3	0.3669	0.4440
6	0.6423	0.2274
10	0.9713	0.1225

7. VECTOR QUANTIZATION ON IMAGES IN QUADTREE FORM

In this chapter, we present a progressive image transmission scheme in which vector quantization is applied to images represented by quadtrees (VQ-QT). For a given image, a mean quadtree is first formed by successively averaging over 2×2 neighbouring pixels. The levels of the mean quadtree correspond to a sequence of reduced-size approximations of the original image. A difference quadtree is then formed by taking the differences between successive levels in the mean quadtree, i.e. between the parents and their siblings. The difference quadtree contains all the necessary information to reproduce successive reduced-size approximations of the image.

As shown below, there are two important features of the quadtree representation. First, as the difference quadtree is composed of a set of differences, the corresponding first order entropy is likely to be much less than the first order entropy of the original image. Secondly, if errors introduced at higher levels are properly delivered to the levels below, the original image can still be reconstructed. To achieve lossless progressive transmission of the original image, the difference quadtree is transmitted starting from the top level. The kth approximation image can formed by adding the information of the kth level to the previously reproduced $(k-1)$th approximation. Note that since the first order entropy of the difference quadtree is much less than that of the mean quadtree, a significant amount of savings can be expected by transmitting the differences, instead of the means [2], on a node-by-node basis. To gain efficiency, vector quantization is applied to the difference quadtree of the image on a level-by-level basis, where vectors are formed by partitioning the levels into spatially contiguous, nonoverlapping, blocks of $m \times m$ nodes. The errors due to quantization at level k are delivered to the next level, $k+1$; in other words, the information to be transmitted at level $k+1$ includes all the residual error components. It should be emphasized that the error delivery procedure is the key for lossless reproduction of

the original image. This makes it possible to reprocess, at lower levels, the information lost at higher levels. Finally, an entropy coder is used to losslessly encode the final residual error image, thus ensuring perfect reproduction of the original image. The experiments demonstrate that it is possible to achieve lossless progressive transmission with compression. Furthermore, at the intermediate levels, it is shown that use of vector quantization results in a coding gain over using only a Huffman coder [30-32].

The rest of this chapter is organized as follows. In section 2, an algorithm which reorganizes the image data into a quadtree form suitable for progressive image transmission is presented. This follows with the description of the progressive image coding technique, in particular, the application of vector quantization and the method of reprocessing the residual errors due to quantization in section 3. Finally, some simulation results on two test images are reported in terms of rate distortion curves and in pictorial form.

7.1 QUADTREE DECOMPOSITION OF IMAGES

A quadtree data structure (Fig. 7.1), which is suited for progressive image transmission by vector quantization, is now presented. An image of size $2^n \times 2^n$ is assumed and quadtree representation is built up as follows. The original image is decomposed into spatially contiguous, nonoverlapping, blocks of 2×2 pixels. Next, a new image at a reduced-resolution (1/2) is formed by finding the mean for each block of 2×2 pixels, and truncating the mean to a fixed number of bits. The same process is now repeated on the reduced-resolution image to yield a new image of (1/4) the resolution; If this process is repeated n times, an image of size 1×1 is produced. The sequence of reduced-resolution images is referred to as the mean quadtree of the image. Note that the top level of size 1×1 is in fact the mean of the original image. The corresponding difference quadtree is then formed by taking the differences between successive levels, i.e. between the parents and their siblings. In algorithmic form, the quadtree

representations of an image are formed as follows:

Formation of Mean Quadtree:

0) *Initialization:* Let $k = n$ and let level k of the mean quadtree be the original image;

1) *Level $k-1$ Formation:* For each spatially contiguous, nonoverlapping, block of 2×2 pixels (nodes) at level k, calculate the truncated mean (Fig. 7.2) using,

$$X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} = [(\frac{X_{k,i,j} + X_{k,i,j+1} + X_{k,i+1,j} + X_{k,i+1,j+1}}{4})] \qquad i,j = 1,3,...2^k - 1 \tag{7.1.1}$$

where $[\alpha]$ means truncation of $(\alpha + 0.5)$;

2) *Termination:* Let $k = k - 1$ and if $k \neq 0$, return to step 1; otherwise, stop.

Formation of Difference Quadtree:

The kth level of the difference quadtree is formed by taking the difference between the $(k-1)$th level and the kth level of the mean quadtree (Fig. 7.3),

$$\begin{cases} D_{k,i,j} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - X_{k,i,j} \\ D_{k,i,j+1} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - X_{k,i,j+1} & k = n, n-1, ...1 \\ D_{k,i+1,j} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - X_{k,i+1,j} & i,j = 1,3,...2^k - 1 \\ D_{k,i+1,j+1} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - X_{k,i+1,j+1}. \end{cases} \tag{7.1.2}$$

A detailed example of both the mean quadtree and difference quadtree formations for the case $n = 2$ is shown in Fig. 7.4.

Transmission and Reconstruction:

To achieve lossless progressive image transmission, the difference quadtree is transmitted starting from the top level. In other words, all that need be transmitted are the top most value of the difference quadtree, $X_{0,1,1}$, and the difference sets $\{D_{k,i,j}\}$. At the receiving end, the kth reduced-size approximation of the original image, corresponding to the kth level of the mean quadtree, is reconstructed by adding the kth level of the difference quadtree to the previous approximation of level $(k-1)$,

$$\begin{cases} X_{k,i,j} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - D_{k,i,j} \\ X_{k,i,j+1} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - D_{k,i,j+1} \\ X_{k,i+1,j} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - D_{k,i+1,j} \\ X_{k,i+1,j+1} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - D_{k,i+1,j+1} \end{cases} \qquad i,j = 1,3,...2^k - 1. \qquad (7.1.3)$$

Remarks:

We now highlight a few properties of the difference quadtree formed. First of all, as the neighboring pixels in most natural images tend to be highly correlated, the values of the differences $\{D_{k,i,j}\}$ will tend to concentrate around zero. Therefore, the first order entropy of the difference quadtree, H_q^1, is likely to be much less than the first order entropy of the original image, H_o^1, i.e.

$$H_q^1 < H_o^1. \qquad (7.1.4)$$

Here, H_q^1 and H_o^1 are defined, respectively, as follows,

$$H_q^1 = -\sum_i P_q(i) \log_2 P_q(i), \qquad (bits/node) \qquad (7.1.5)$$

$$H_o^1 = -\sum_i P_o(i) \log_2 P_o(i), \qquad (bits/pixel) \qquad (7.1.6)$$

where $P_q(i)$ and $P_o(i)$ are, respectively, the frequency of occurrence of value i in the difference quadtree and in the original image. Note that the first order entropy defined above is measured on a node-by-node or a pixel-by-pixel basis.

A second observation regards the number of nodes in the difference quadtree. Since the number of nodes at level k is

$$\frac{2^n \times 2^n}{2^{n-k} \times 2^{n-k}} = 2^k \times 2^k, \qquad k = 0, 1, ...n, \tag{7.1.7}$$

the total number of nodes on the quadtree data structure is equal to

$$\begin{aligned} \sum_{k=0}^{n} \frac{2^n \times 2^n}{2^{n-k} \times 2^{n-k}} &= \sum_{k=0}^{n} 2^k \times 2^k \\ &= \frac{1 - 2^{n+1} \times 2^{n+1}}{1 - 2 \times 2} \\ &= \frac{4}{3}(2^n \times 2^n) - \frac{1}{3} \\ &\approx \frac{4}{3}(2^n \times 2^n). \end{aligned} \tag{7.1.8}$$

This implies that the number of nodes on the quadtree is about one third more than the number of pixels on the original image. For comparative purposes, we therefore define an equivalent entropy, H^1_{eq}, measured on a pixel basis,

$$H^1_{eq} = \frac{\frac{4}{3}(2^n \times 2^n) - \frac{1}{3}}{(2^n \times 2^n)} H^1_q \approx \frac{4}{3} H^1_q. \quad (bits/pixel). \tag{7.1.9}$$

The entropies defined above are measured from two test images, a face image (Fig. 7.10) and a boat image (Fig. 7.11). The histograms of the face and boat images shown in, respectively, Fig. 7.5 and Fig. 7.6, are seen to be more uniformly distributed; whereas the histograms of the corresponding difference quadtrees tend to lie around zero. The entropies estimated from the histograms are as follows: $H^1_o = 7.5178$ bits/pixel, $H^1_q = 4.3861$ bits/node and $H^1_{eq} = 5.8481$ bits/pixel for the face image, and $H^1_o = 7.5395$ bits/pixel, $H^1_q = 5.1582$ bits/node and $H^1_{eq} = 6.8775$ bits/pixel for the boat image, respectively. Note that, for both two images,

$$H^1_{eq} < H^1_o. \tag{7.1.10}$$

Consequently, it should be possible to save a certain amount of bits by encoding the images in quadtree form, rather than the original image directly.

A third property of the particular quadtree representation chosen is that of error-recovery. In other words, if the errors introduced at higher levels are properly delivered to the lower levels, the original image can still be reproduced. The principle of error recovery is illustrated in Fig. 7.7, where an error Δ is introduced at node $(k-1, [\frac{i+1}{2}], [\frac{j+1}{2}])$. The error Δ is delivered to the next level, k, and added to the four siblings $\{D_{k,i+ii,j+jj},\ ii,jj = 0,1\}$ to yield $\{D'_{k,i+ii,j+jj},\ ii,jj = 0,1\}$ (Fig. 7.7). The corresponding pixel values $\{X'_{k,i+ii,j+jj},\ ii,jj = 0,1\}$ on the kth level of the mean quadtree are then evaluated from equation (7.1.3),

$$\begin{aligned} X'_{k,i+ii,j+jj} &= X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - D'_{k,i+ii,j+jj} \\ &= X_{k-2,[\frac{[\frac{i+1}{2}]+1}{2}],[\frac{[\frac{j+1}{2}]+1}{2}]} - D'_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - D'_{k,i+ii,j+jj} \\ &= X_{k-2,[\frac{[\frac{i+1}{2}]+1}{2}],[\frac{[\frac{j+1}{2}]+1}{2}]} - D_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} + \Delta - D_{k,i+ii,j+jj} - \Delta \\ &= X_{k-2,[\frac{[\frac{i+1}{2}]+1}{2}],[\frac{[\frac{j+1}{2}]+1}{2}]} - D_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} - D_{k,i+ii,j+jj} \\ &= X_{k,i+ii,j+jj}. \qquad ii,jj = 0,1. \end{aligned} \tag{7.1.11}$$

In other words, the error recovery property ensures that the correct original mean quadtree values are reconstructed. The error-recovery technique, in particular, provides the possibility of lossless encoding of the image, when coding of the difference quadtree introduces quantization errors.

Fig. 7.8 shows an example of error recovery effect. An error "e" is introduced at node (1,1,1) of the difference quadtree of Fig. 7.4. The error "e" is delivered to the next level, 2, and added to the four siblings (Fig. 7.8a). The 1st reduced-size approximation corresponding to level 1 of the mean quadtree is formed from level 0 and level 1 of the difference quadtree using equation (7.1.3). Note the presence of an error "e" in (1,1,1) of the mean quadtree (Fig. 7.8b). The 2nd reduced-size approximation corresponding to level 2 of the mean quadtree is formed from the previous approximation and level 2 of the difference quadtree with a modified top left quadrant (Fig. 7.8c). Note that the original image is correctly recovered.

7.2 VECTOR QUANTIZATION

To efficiently transmit the image in quadtree form, vector quantization can be applied to the difference quadtree on a level-by-level basis as follows. Level K of the difference quadtree is first vector quantized, where $0 < K \leq n$. Then, the errors due to quantization at level K are delivered to the next level $K+1$ and added to respective sibling node values in order to permit error-recovery. The process is repeated from higher levels to lower levels. Finally, an entropy coder is used to losslessly encode the final residual error image at the bottom most level, thus ensuring perfect reproduction of the original image.

The basic steps involved in applying vector quantization to the quadtree data structure of images and reprocessing the residual errors due to quantization are as follows (Fig. 7.9):

0) Let $k = K$, where $0 < K \leq \log_2 N$;

1) **Vector Formation:** Level k of the difference quadtree is decomposed into a set of vectors corresponding to spatially contiguous, nonoverlapping, square blocks of $m \times m$ nodes, $\{D_{k,i+ii,j+jj},\ \ ii,jj = 0,1,...m-1\ \}$ where $i,j = 1,m,...2^k - m + 1$;

2) **Training Sequence Selection:** All the vector formed in step 1 are chosen as the training sequence;

3) **Codebook Generation:** The codebook is generated by applying the generalized Lloyd clustering algorithm [65], where the normalized mean square error (NMSE) is used as the criterion for distortion;

4) **Codebook Encoding:** The codebook obtained is quantized to the same resolution as the difference quadtree and then encoded by a variable length coder based on the frequencies of occurrence of the values in the codebook;

5) **Vector Quantization:** For each vector at level k, the closest codeword in the codebook is determined and the corresponding label of this codeword transmitted;

6) If $k = n$, go to step 9 and otherwise, continue;

7) **Quantization Error Delivery:** In order to losslessly recover the original image, the residual errors $\{e_{k,[\frac{i+1}{2}],[\frac{j+1}{2}]}\}$ due to quantization at level k are delivered to the next level $k+1$ and added to the respective sibling nodes (Fig. 7.7) as follows,

$$\begin{cases} D'_{k+1,i,j} = e_{k,[\frac{i+1}{2}],[\frac{j+1}{2}]} + D_{k+1,i,j} \\ D'_{k+1,i,j+1} = e_{k,[\frac{i+1}{2}],[\frac{j+1}{2}]} + D_{k+1,i,j+1} \\ D'_{k+1,i+1,j} = e_{k,[\frac{i+1}{2}],[\frac{j+1}{2}]} + D_{k+1,i+1,j} \\ D'_{k+1,i+1,j+1} = e_{k,[\frac{i+1}{2}],[\frac{j+1}{2}]} + D_{k+1,i+1,j+1}. \end{cases} \qquad i,j = 1,3,...2^{k+1}-1 \tag{7.2.1}$$

In other words, the information to be transmitted at level $k+1$ is modified by the residual error $\{e_{k,[\frac{i+1}{2}],[\frac{j+1}{2}]}\}$ at level k;

8) Set $k = k+1$ and return to step 1;

9) **Entropy Coding:** Finally, an entropy coder, such as a Huffman coder operating on a pixel-by-pixel basis, is used to losslessly encode the residual error image at level $k = n$, thus ensuring perfect reproduction of the original image.

At the receiver, a sequence of reduced-size approximations to the original image can be obtained by using the error-recovery technique and equation (7.1.3) on the quantized difference values of equation (7.2.1). Note that error delivery in step 7 is the key to achieving lossless reproduction of the original image. This makes it possible to reprocess, at lower levels, the residual errors due to quantization at higher levels. In this way, all the residual errors due to quantization are passed down, from higher

levels to lower levels, so that the information to be transmitted at level $k+1$ actually contains all the residual error components.

7.3 SIMULATIONS

Computer simulations using the proposed progressive image transmission scheme (VQ-QT) on two test images are presented. Both test images are of size 256×256 and quantized to 256 levels. The first, shown in Fig. 7.10, is the face image. The second, shown in Fig. 7.11, is the boat image. The histograms of both the original images and the corresponding difference quadtree representations are illustrated in Fig. 7.5 and 6, respectively. The entropies estimated from the histograms are as follows: $H_o^1 = 7.5178$ bits/pixel, $H_q^1 = 4.3861$ bits/node, and $H_{cq}^1 = 5.8481$ bits/pixel for the face image; and $H_o^1 = 7.5395$ bits/pixel, $H_q^1 = 5.1582$ bits/node, and $H_{cq}^1 = 6.8775$ bits/pixel for the boat image, respectively.

The mean quadtree is first formed using (7.1.1) from the original images followed by difference quadtree formation (7.1.2). The quadtree representation contains nine levels; the number of nodes on each level is equal to $2^k \times 2^k$, $k = 0, 1, ...8$. The node values are represented only by integers due to truncation (7.1.1). For comparative purposes, a Huffman coder is used to code the images represented by the difference quadtrees and the results are reported in Tables 7.1a and 7.2a. The results on using vector quantization are tabulated in Tables 7.1b and 7.2b, and Tables 7.1c and 7.2c. In the implementation, it was found more efficient to code the first three levels of the difference quadtree with a Huffman coder. Vector quantization is applied only to levels 3 to 8 and the vectors are formed by partitioning the levels into spatially contiguous, nonoverlapping, blocks of 2×2 or 4×4 nodes. At each level (≥ 3), the codebook is first generated by the generalized Lloyd algorithm [65] and then quantized to the same resolution as the difference quadtree. The errors due to quantization are delivered and added to the next level (7.2.1). A set of reduced-size approximations is formed from the quantized-modified difference quadtree (7.1.3). The corresponding successive

approximations are obtained from the reduced-size approximations as follows

$$\hat{X}_{k,ii+2^{n-k}(i-1),jj+2^{n-k}(j-1)} = X_{k,i,j} \qquad (7.3.1)$$

where $i,j = 1,2,...2^k$ and $ii,jj = 1,2,...2^{n-k}$; in other words, the pixels are just repeated.

The results of the computer simulations are listed in Table 7.1 and 7.2, where the components shown are defined as follows,

$R_h(k)$ – the bit rate required for Huffman coding of the difference quadtree at level k;

$N_c(k)$ – the number of codewords in codebook at level k;

$R_l(k)$ – the bit rate for transmitting the labels at level k, where a variable length coder is used, dependent upon the frequency of occurrence of labels;

$R_c(k)$ – the bit rate for transmitting the codebook at level k, where a variable length coder is also used;

$R_p(k) = \sum_{m=0}^{k}(R_h(m) + R_l(m) + R_c(m))$ – the total progressive bit rate up to level k.

In the evaluation of system performance, the normalized mean square error (NMSE) between the original image X and the kth approximation $\hat{X}_k$ is used,

$$\text{NMSE} = \frac{\sum_{i,j=0}^{255}(X_{ij} - \hat{X}_{k,ij})^2}{\sum_{ij=0}^{255} X_{ij}^2}, \qquad k = 0,1,...8, \qquad (7.3.2)$$

where X_{ij} and $\hat{X}_{k,ij}$ represent the (i,j)th element of the original image and its kth approximation, respectively.

Fig. 7.12 and 7.13 show the rate distortion curves based on Table 7.1 and 7.2. It is clear that, for both test images, the performance by using vector quantization is much better than that by simply using an entropy coder, such as Huffman coder, on the difference quadtree. At NMSE = 0.17 %, there is a bit rate saving of about 0.882

bits/pixel, or a relative bit rate saving of about 60.0% ($\frac{1.462-0.580}{1.462} \times 100\%$) for the face image. At NMSE = 0.727 %, the corresponding savings are, respectively, 1.118 bits/pixel and 65% ($\frac{1.719-0.601}{1.719} \times 100\%$) for the boat image. Moreover, it is noted that the better performance is achieved with the larger vector dimension, as shown in Fig. 7.14 and 7.15. Finally, the original face image can be recovered at a bit rate of about 4.9680 bits/pixel and the original boat at a bit rate of about 5.8622 bits/pixel. This implies that, as a lossless coder, the proposed scheme is superior to the entropy coder operating on a pixel-by-pixel basis, by about 2.5498 (7.5178-4.9680) bits/pixel for the face image and 1.6773 (7.5395-5.8622) bits/pixel for the boat image, respectively.

Fig. 7.14 and 7.15 show two sets of successive approximations to the face and the boat images, by the VQ-QT scheme with the vector dimension of 4×4, in sequence. The sequences of successive approximations rapidly converge to a good quality both subjectively and objectively. Note that excellent quality reproduction is obtained by the 8th level at a bit rate of only about 0.6 bits/pixel.

Finally, we compare all the four presented techniques in terms of rate distortion curve. Fig. 7.16 shows the rate distortion curves obtained by using RVQ-SPA, RVQ-TRA, MVQ-OBA and VQ-QT with a block size of 4×4 (from Tables 4.1b, 5.1b, 6.2 and 7.1c). From Fig. 7.16, it is noted that VQ-QT achieves the best performance at low bit rates (< 0.3 bits/pixel) whereas MVQ-OBA is best at a higher bit rate ($>$ 0.3 bits/pixel).

7.4 CONCLUSIONS

In this chapter, a progressive image transmission scheme (VQ-QT) which utilizes the features of quadtree data structure of images and vector quantization has been presented. The experimental results demonstrate that the VQ-QT scheme achieves lossless progressive transmission with compression. The quadtree data structure proposed makes it possible to transmit the images progressively. Compression is obtained by using vector quantization on each level. Lossless reproduction is guaranteed by

delivering the residual errors due to quantization from higher levels to lower levels and using an entropy coder on the final residual error image. Among all the four presented techniques, VQ-QT is seen to achieve the best performance at a low bit rate.

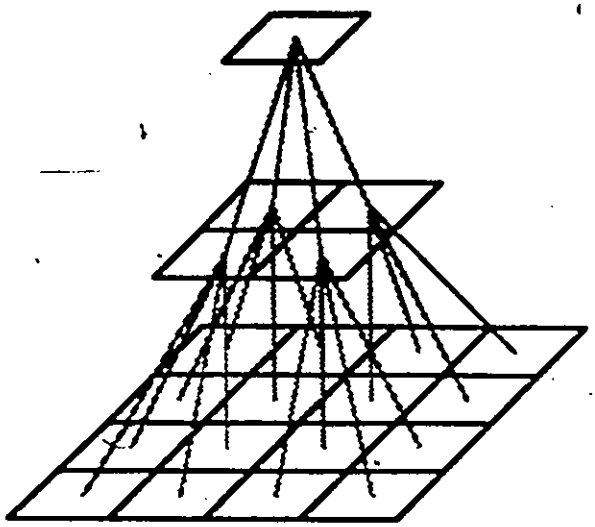


Fig. 7.1. An example of a quadtree representation of an image of size 4×4 pixels.

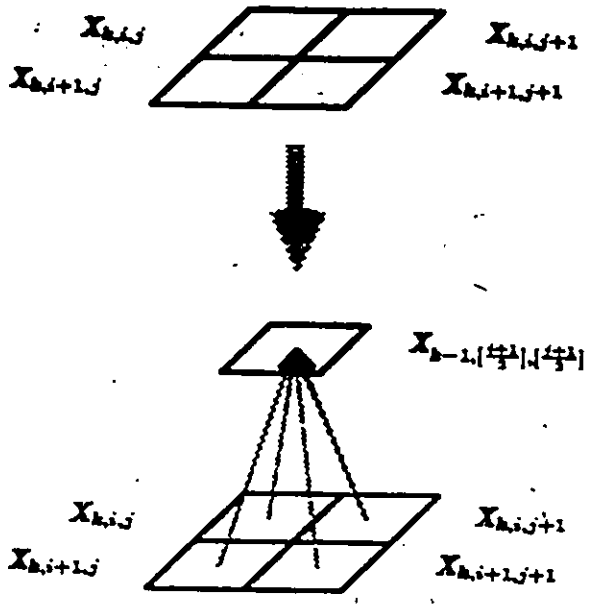


Fig. 7.2. Formation of mean quadtree. The pixel values at level $k-1$ of the mean quadtree are obtained by calculating the mean for spatial contiguous, nonoverlapping, block of size 2×2 at level k and then truncating the mean to a fixed number of bits.

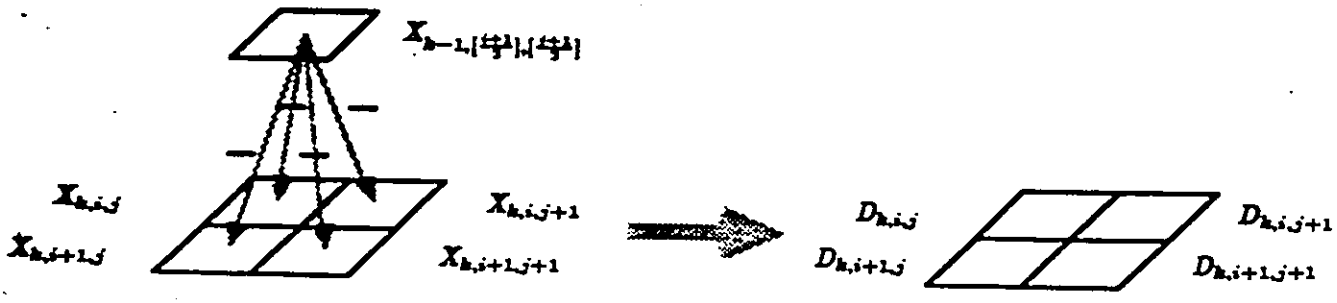


Fig. 7.3. Formation of difference quadtree. The node values at level k of the difference quadtree are formed by taking differences between successive levels in the mean quadtree, i.e. between the parents and their siblings.

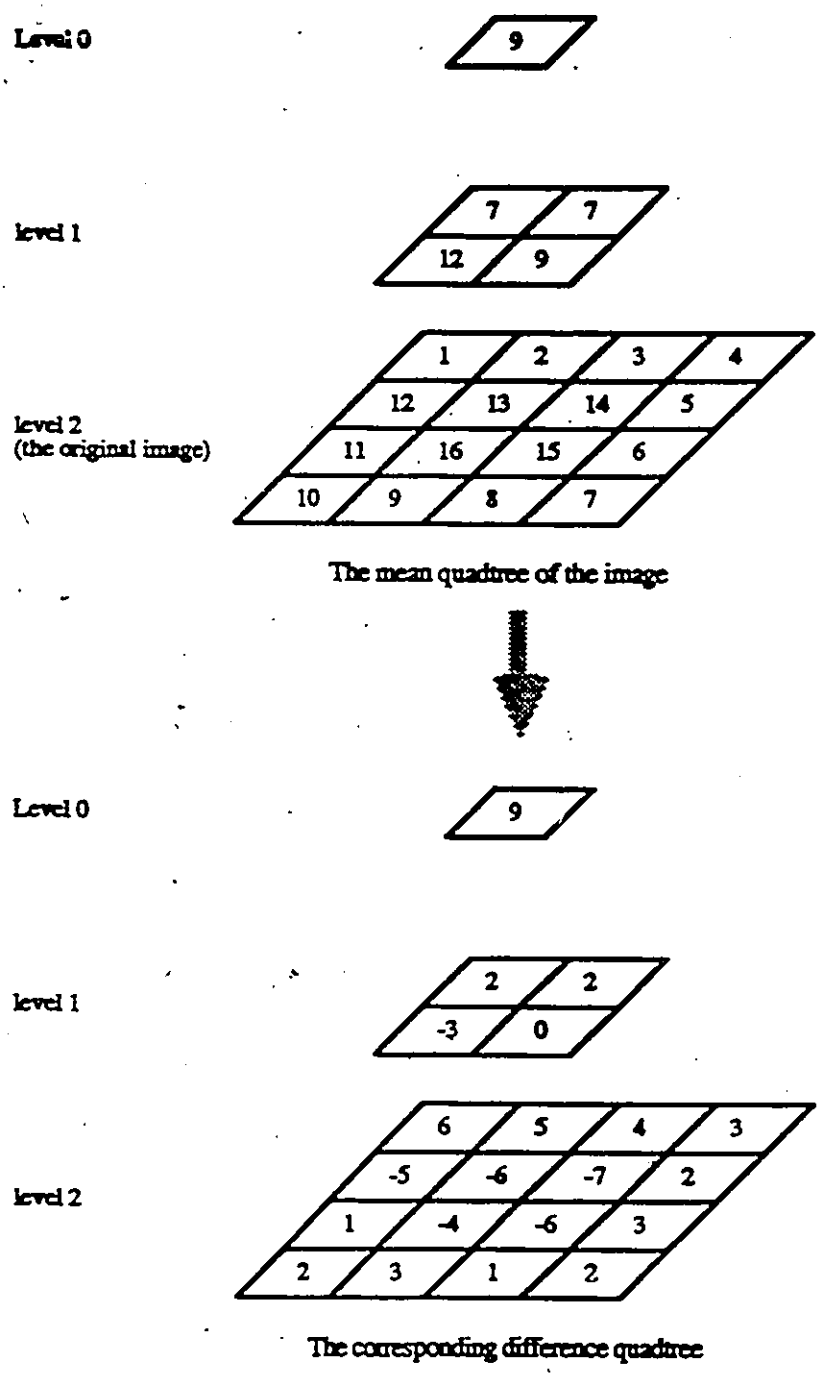


Fig. 7.4. An example of a mean quadtree followed by the corresponding difference quadtree for an image of size 4 × 4.

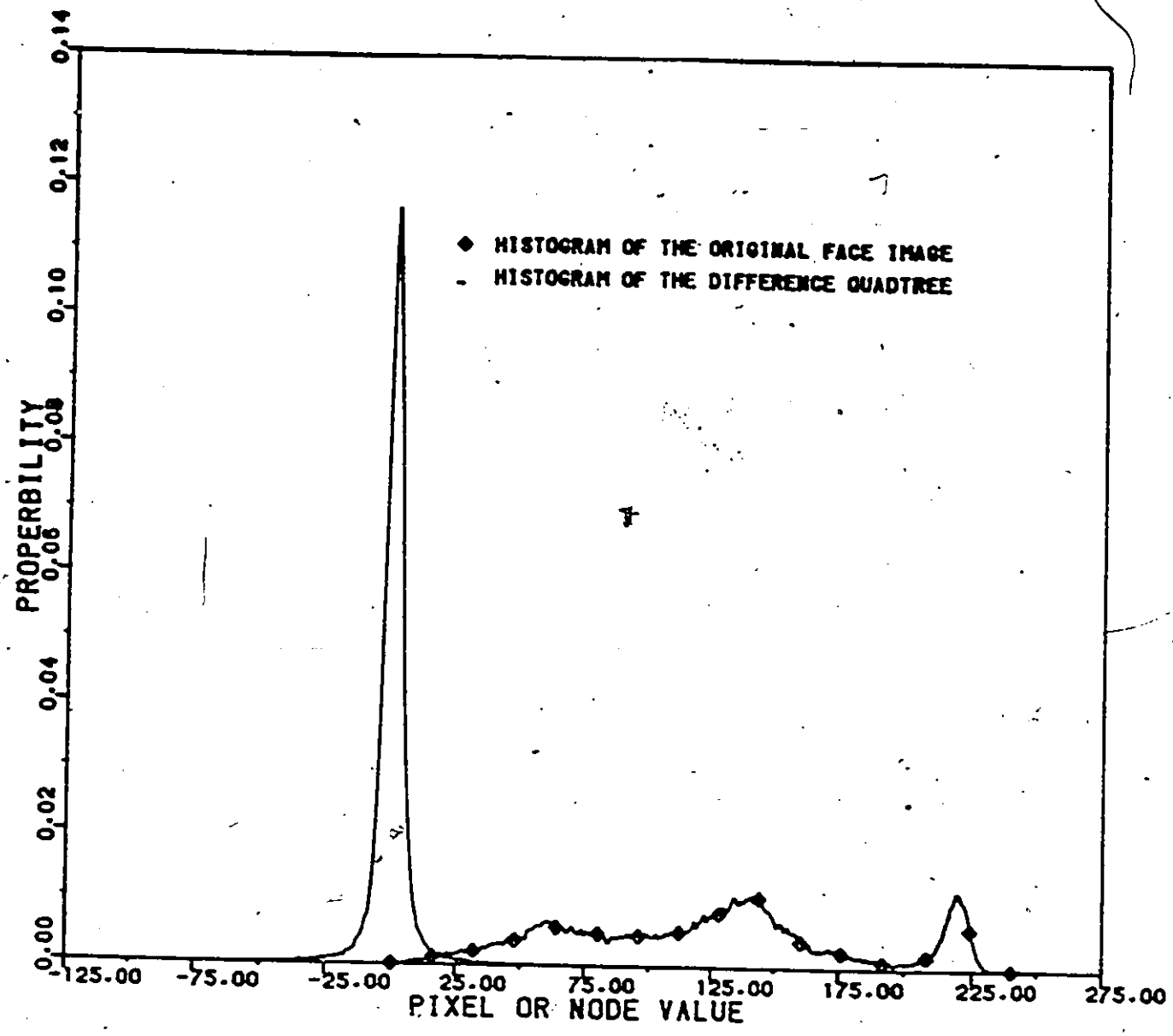


Fig. 7.5. Histograms for the original face image and the corresponding difference quadtree representation. The former is more uniformly distributed and the latter tends to lie around zero. The entropies estimated from the histograms are, respectively, as follows, $H_o^1 = 7.5178$ bits/pixel; $H_q^1 = 4.3861$ bits/node and $H_{cq}^1 = 5.8481$ bits/pixel.

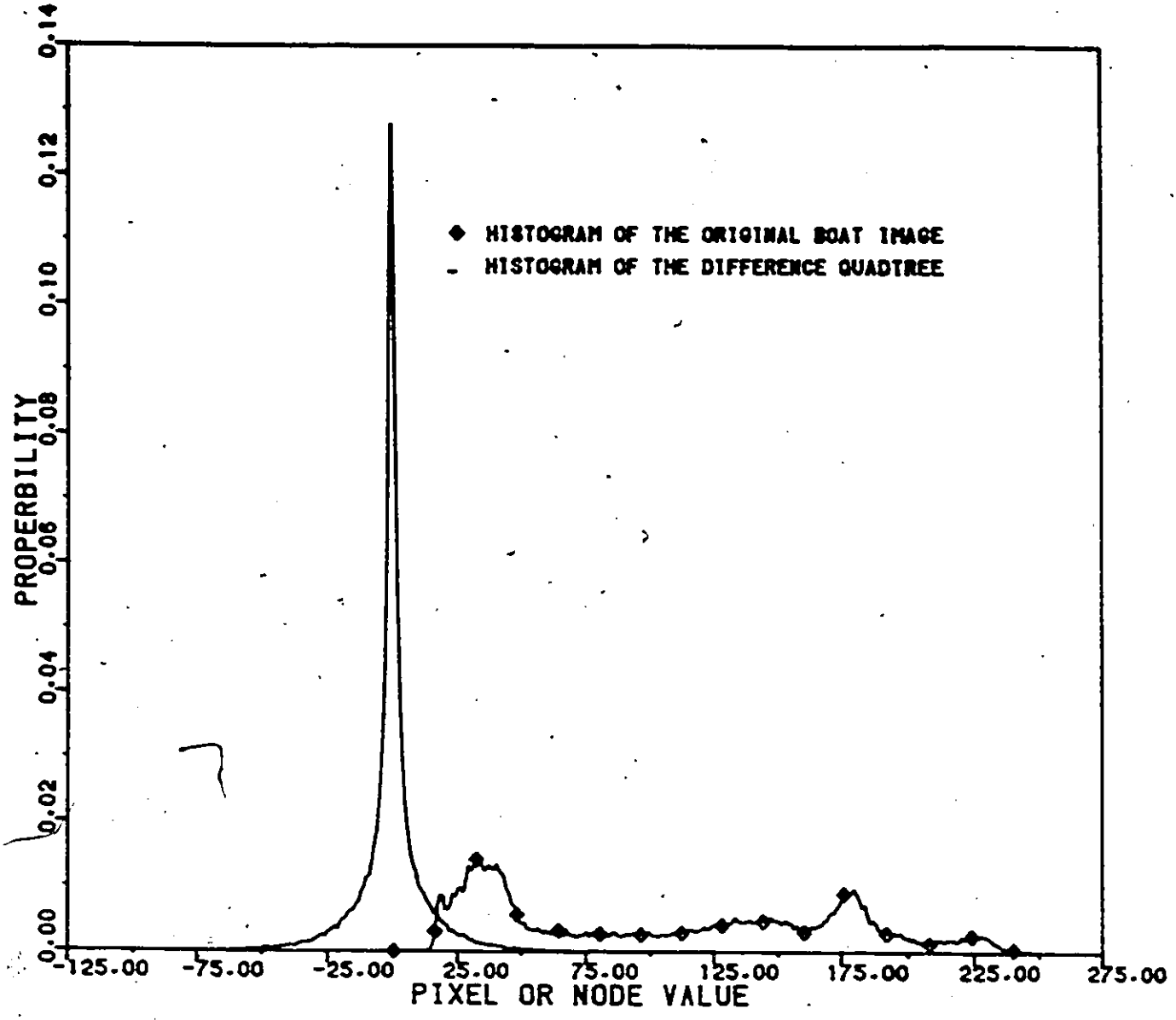


Fig. 7.6. Histograms for the original boat image and the corresponding difference quadtree representation. The former is more uniformly distributed and the latter tends to lie around zero. The entropies estimated from the histograms are, respectively, as follows, $H_o^1 = 7.5395$ bits/pixel; $H_q^1 = 5.1582$ bits/node and $H_{eq}^1 = 6.8775$ bits/pixel.

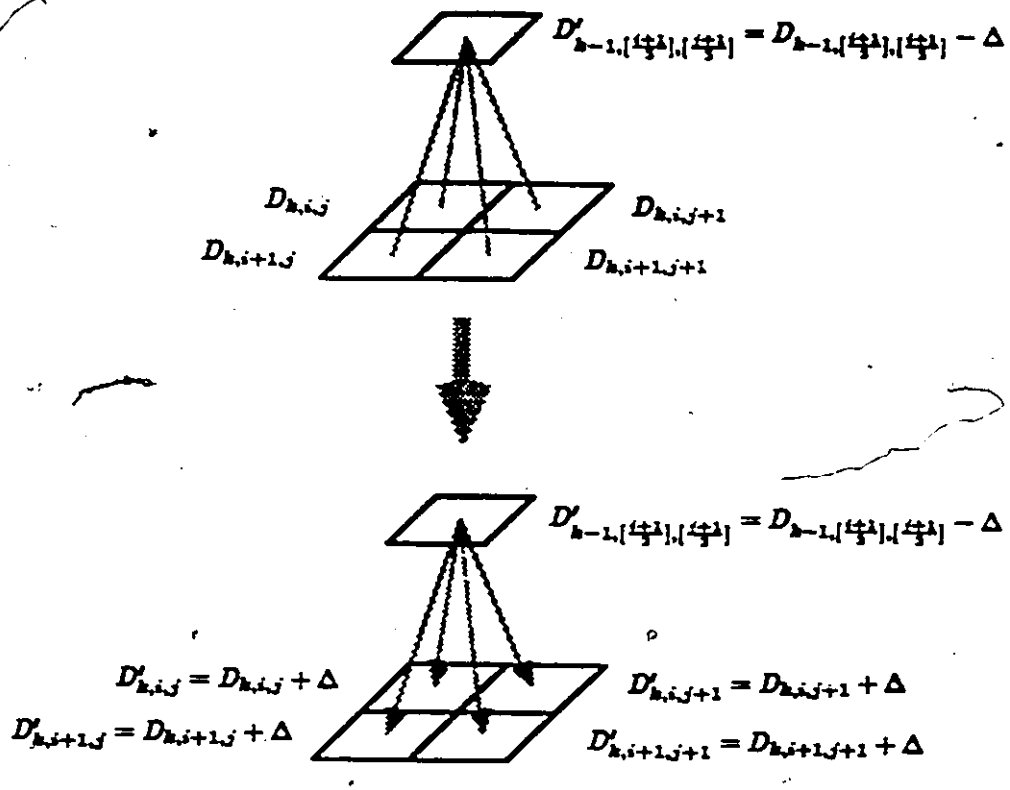


Fig. 7.7. Error delivery: An error Δ occurring at node $(k-1, [\frac{i+1}{2}], [\frac{j+1}{2}])$ is delivered to the next level k and added to the four siblings. In other words, the siblings are modified by the error introduced at their parent value.

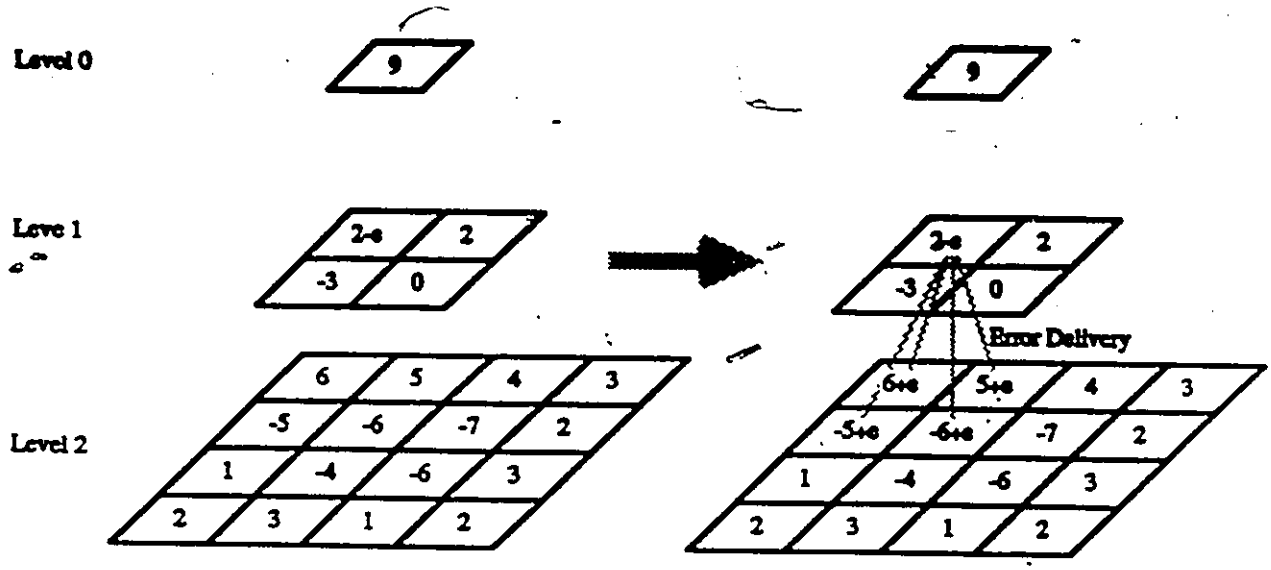


Fig. 7.8a. Given the difference quadtree from Fig. 4. An error "e" introduced at node (1,1,1) is delivered to the next level, 2, and added to the four siblings.

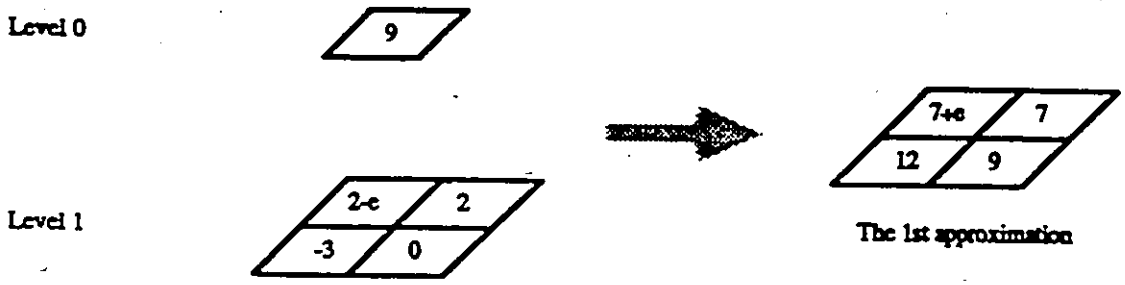


Fig. 7.8b. The 1st reduced-size approximation is formed from level 0 and level 1 of the difference quadtree. Note the presence of an error "e" in (1,1,1) of the mean quadtree.

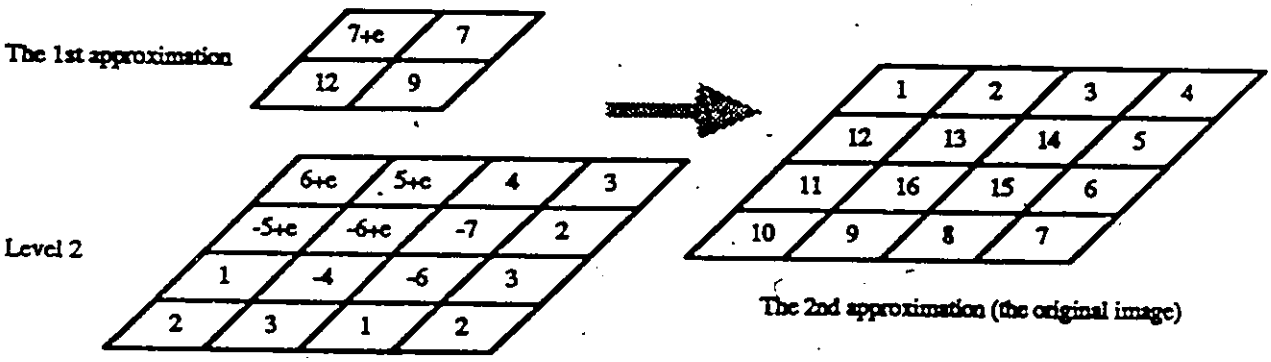


Fig. 7.8c. The 2nd reduced-size approximation corresponding to level 2 of the mean quadtree (the original image) is now formed from the 1th reduce-size approximation (1,1) and level 2 of the difference quadtree with a modified top left quadrant. Note that even though an error "e" is introduced at level 1 of the 1st reduced-size approximation, the 2nd approximation (the original image) is exactly recovered (Fig. 7.4).

Fig. 7.8. An example of error recovery effect.

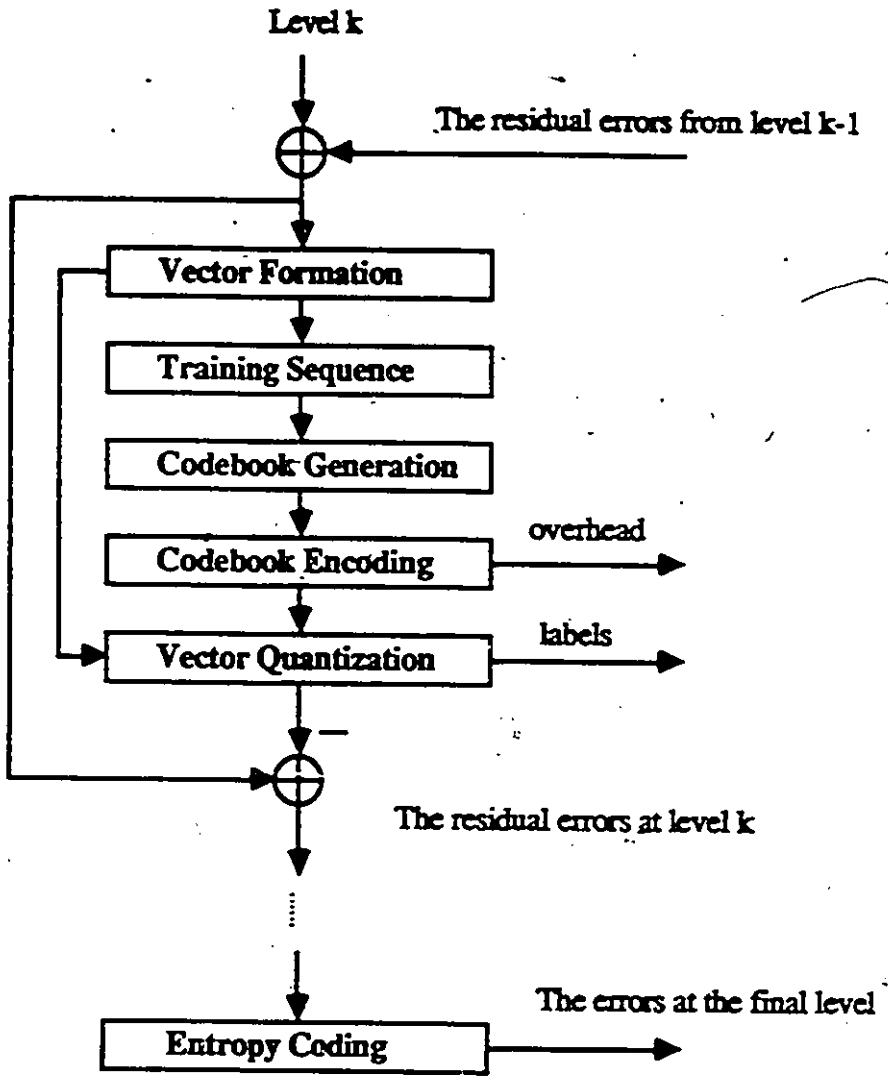


Fig. 7.9. The steps involved in applying vector quantization to the difference quadtree.

Fig. 7.10. The face image of 256 × 256 pixels with 8 bits/pixel.

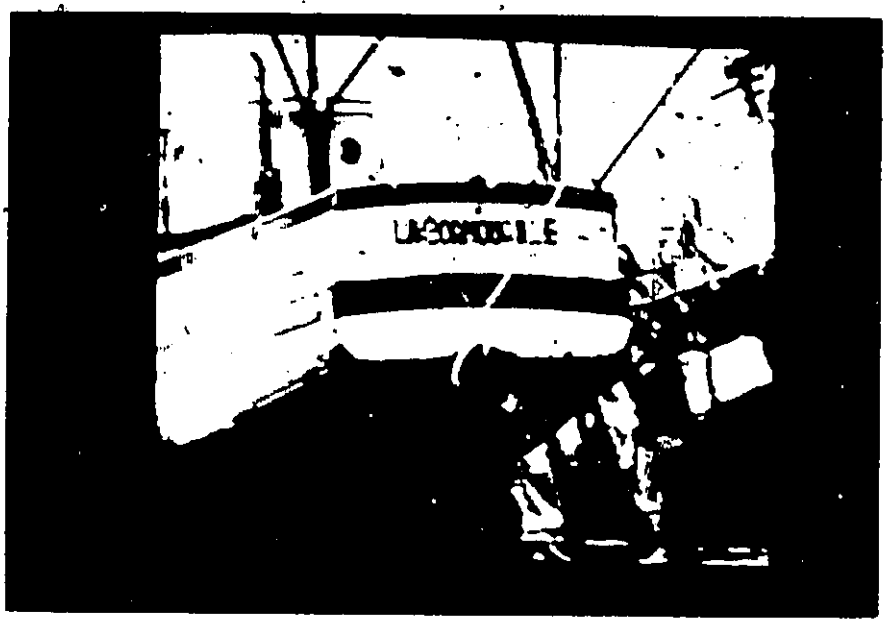


Fig. 7.11. The boat image of 256 × 256 pixels with 8 bits/pixel.

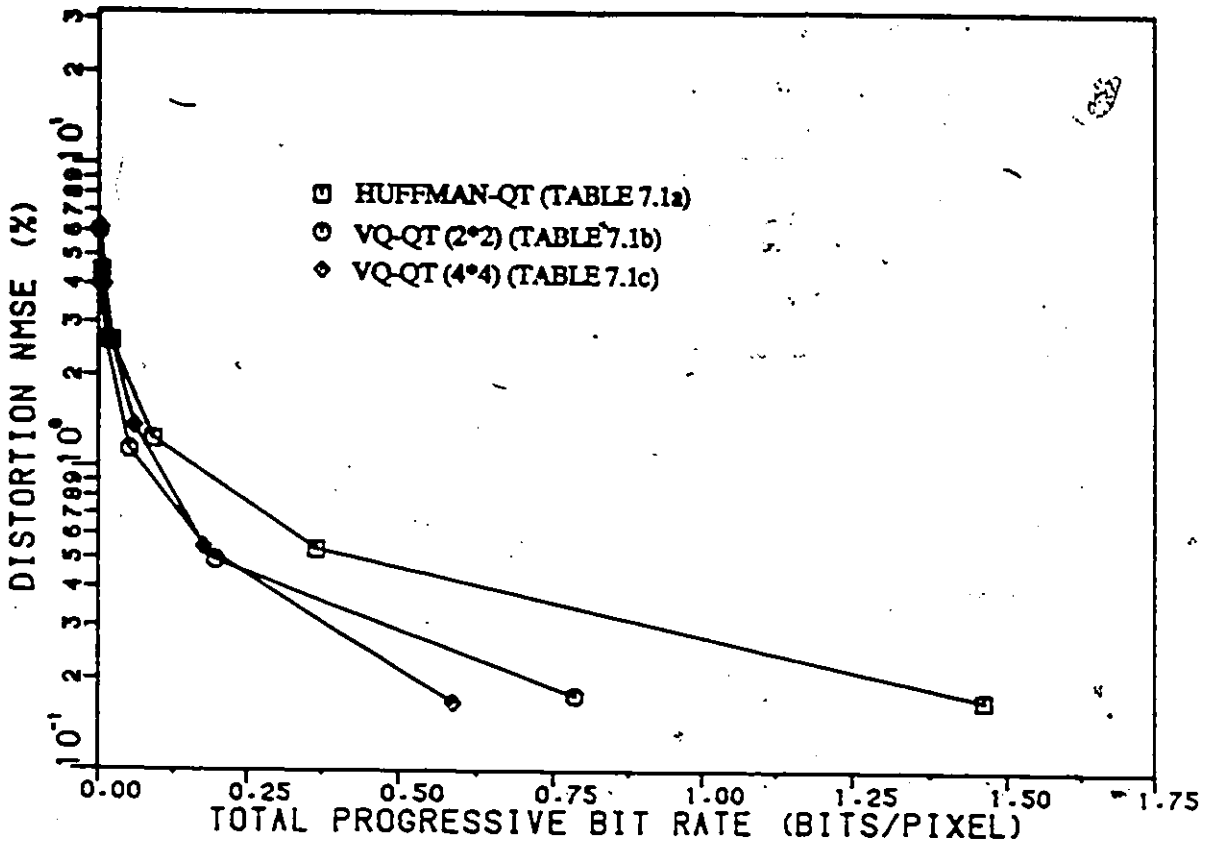


Fig. 7.12. The rate distortion curves for the face image in quadtree form using a Huffman coder and VQ-QT with the vector dimension 2×2 and 4×4 (Table 7.1). It is demonstrated that the use of vector quantization results in a coding gain over a Huffman coder.

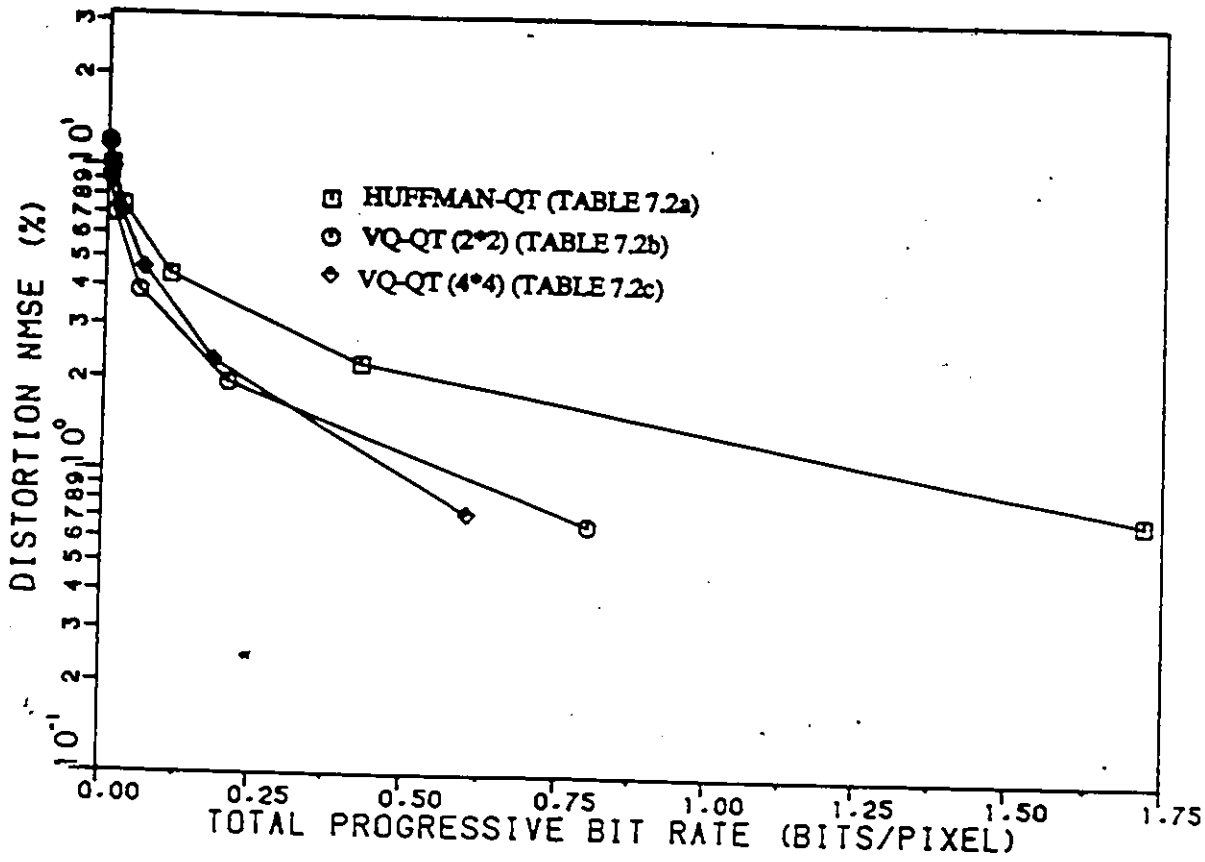


Fig. 7.13. The rate distortion curves for the boat image in quadtree form using a Huffman coder and VQ-QT with the vector dimension 2×2 and 4×4 (Table 7.2). It is demonstrated that the use of vector quantization results in a coding gain over a Huffman coder.

Fig. 7.14. The sequence of successive approximations (levels 4 through 9 of Table 1c) to the original face image (Fig. 7.11) with a vector dimension of 4×4. The original image has been reproduced as the last one in the sequence.

Fig. 7.15. The sequence of successive approximations (levels 4 through 9 of Table 2c) to the original boat image (Fig. 7.12) with a vector dimension of 4×4. The original image has been reproduced as the last one in the sequence.

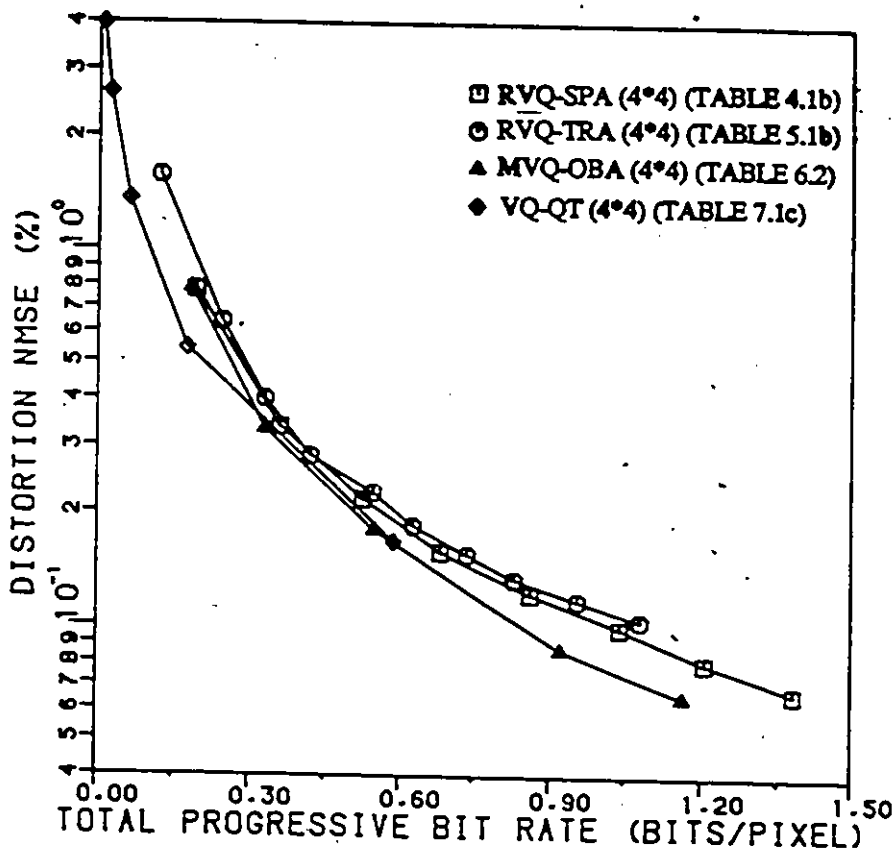


Fig. 7.16. The rate distortion curves obtained by using RVQ-SPA, RVQ-TRA, MVQ-OBA and VQ-QT with a block size of 4×4 (from Tables 4.1b, 5.1b, 6.2 and 7.1c).

Table 7.1a Computer simulation results for the face image by using Huffman coder on the difference quadtree

Level k	Rh(k) (bits/pixel)	Rp(k) (bits/pixel)	NMSE(%)
0	0.0000669	0.0000669	16.2379
1	0.0002677	0.0003346	11.8840
2	0.0010708	0.0014054	8.7855
3	0.0042833	0.0056888	4.4424
4	0.0171334	0.0228223	2.5904
5	0.0685338	0.0913561	1.2322
6	0.2741353	0.3654915	0.5334
7	1.0965415	1.4620330	0.1706
8	4.3861660	5.8481990	0.0000

Table 7.2a Computer simulation results for the boat image by using Huffman coder on the difference quadtree

Level k	Rh(k) (bits/pixel)	Rp(k) (bits/pixel)	NMSE(%)
0	0.0000787	0.0000787	27.8332
1	0.0003148	0.0003935	19.5735
2	0.0012593	0.0016528	13.8307
3	0.0050373	0.0066901	10.0295
4	0.0201492	0.0268393	7.3884
5	0.0805968	0.1074362	4.3708
6	0.3223875	0.4298238	2.2436
7	1.2895503	1.7193742	0.7282
8	5.1582012	6.8775754	0.0000

Table 7.1b Computer simulation results for the face image by using vector (2*2) quantization on the difference quadtree

Level k	Bit Rate Distribution at Level k				Rp(k) (bits/pixel)	NMSE(%)
	Rh(k) (bits/pixel)	Vector Quantization (2*2)				
		Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)		
0	0.0000669				0.0000669	16.2379
1	0.0002677				0.0003346	11.8840
2	0.0010708				0.0014054	8.7855
3		2	0.0002330	0.0003356	0.0019741	6.0095
4		4	0.0018607	0.0008544	0.0046894	4.0289
5		4	0.0074427	0.0009460	0.0130781	2.6188
6		8	0.0372137	0.0022583	0.0525502	1.1339
7		8	0.1408433	0.0021247	0.1955183	0.4913
8		8	0.5875044	0.0020751	0.7850980	0.1763
9	4.3086748				5.0937728	0.0000

Table 7.1c Computer simulation results for the face image by using vector (4*4) quantization on the difference quadtree

Level k	Bit Rate Distribution at Level k				Rp(k) (bits/pixel)	NMSE(%)
	Rh(k) (bits/pixel)	Vector Quantization (4*4)				
		Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)		
0	0.0000669				0.0000669	16.2379
1	0.0002677				0.0003346	11.8840
2	0.0010708				0.0014054	8.7855
3		2	0.0000610	0.0023498	0.0038163	6.1119
4		4	-0.0004827	0.0052565	0.0095555	3.9734
5		8	0.0027452	0.0110206	0.0233215	2.6082
6		16	0.0136678	0.0240743	0.0610637	1.3580
7		32	0.0645644	0.0502336	0.1758618	0.5424
8		64	0.3180145	0.0931685	0.5870449	0.1671
9	4.3810010				4.9680459	0.0000

Table 7.2b Computer simulation results for the boat image by using vector (2*2) quantization on the difference quadtree

Level k	Bit Rate Distribution at Level k				Rp(k) (bits/pixel)	NMSE(%)
	Rb(k) (bits/pixel)	Vector Quantization (2*2)				
		Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)		
0	0.0000787				0.0000787	27.8332
1	0.0003148				0.0003935	19.5735
2	0.0012593				0.0016528	13.8307
3		2	0.0002187	0.0003356	0.0022073	11.8961
4		4	0.0018109	0.0009460	0.0049643	9.1906
5		4	0.0074661	0.0009765	0.0134070	6.9191
6		8	0.0405449	0.0021972	0.0561492	3.8679
7		8	0.1504987	0.0022277	0.2088757	1.9493
8		8	0.5897188	0.0021667	0.8007613	0.6716
9	5.1581440				5.9589053	0.0000

Table 7.2c Computer simulation results for the boat image by using vector (4*4) quantization on the difference quadtree

Level k	Bit Rate Distribution at Level k				Rp(k) (bits/pixel)	NMSE(%)
	Rb(k) (bits/pixel)	Vector Quantization (4*4)				
		Nc(k)	Rl(k) (bits/pixel)	Rc(k) (bits/pixel)		
0	0.0000787				0.0000787	27.8332
1	0.0003148				0.0003935	19.5735
2	0.0012593				0.0016528	13.8307
3		2	0.0000495	0.0023193	0.0040217	11.8331
4		4	0.0004347	0.0052014	0.0096578	9.8389
5		8	0.0027468	0.0115774	0.0239822	6.9992
6		16	0.0143341	0.0243896	0.0627060	4.6032
7		32	0.0692045	0.0508756	0.1827862	2.2973
8		64	0.3161984	0.1030025	0.6019872	0.7270
9	5.2602968				5.8622840	0.0000

8. SUMMARY, CURRENT AND FUTURE WORK

8.1 SUMMARY

In this thesis, four progressive image transmission techniques: RVQ-SPA, RQ-TRA, MVQ-OBA and VQ-QT have been presented. All the four techniques operate in a multistage fashion, either in the spatial domain or in the transform domain. At each stage, a fraction of the image information is transmitted and utilized in reconstructing successive approximations. Both theoretical analysis and computer simulation demonstrate that it is possible for all the four techniques to achieve lossless progressive image transmission with compression. The first approximations provide the main structural information of the image at a very low bit rate and successive approximations converge to excellent quality, both objectively and subjectively. The main points of these four techniques are now summarized.

RVQ-SPA

The residual errors due to quantization is iteratively fedback and (vector) requantized in the spatial domain. To achieve lossless reproduction, an entropy coder is used to encode the final residual error image. It is shown that the total bit rate for lossless reproduction of the image with regard to the total progressive bit rate is a concave curve. This implies that there is a minimum point at which the system should be switched to the entropy coder in order to achieve the minimum total bit rate for lossless transmission of the image. Residual vector quantization is shown to be no worse than a signal-stage vector quantization both theoretically and experimentally.

RQ-TRA

Residual error quantization is extended to the transform domain where the transform coefficients are iteratively requantized, either in a scalar manner (RSQ) or in a vector manner (RVQ), with better performance achieved by vector quantization. It is

shown that the average reconstruction error variance converges to zero, as the number of stages approaches infinity. In comparing RVQ-TRA with RVQ-SPA, we found that no gain is obtained using RVQ-TRA over RVQ-SPA. Note that in RVQ-TRA, the transform blocks are simply taken as the vectors and vector quantized, implying that the nonuniform distribution of coefficient variances is not exploited.

MVQ-OBA

To exploit the nonuniform distribution of the coefficient variances, a multistage vector quantization is applied to the transform coefficients in such a way that the total number of bits assigned to each coefficient is proportional to the coefficient variance. An optimal bit allocation map is first found according to the coefficient variances. A set of bit allocation planes is then obtained by slicing an optimal bit allocation map. With the set of bit allocation planes, the coefficients are vector quantized in a multistage manner, which results in progressive image transmission. At each stage, only the coefficients assigned a non-zero number of bits are combined into a vector and vector quantized where the number of codewords in the codebook is determined from the bit allocation plane. An important property is that the values on each bit allocation plane can be almost equal, implying the corresponding coefficients (or residual error coefficients) have similar variances. The simulation results demonstrate that vector quantization of such coefficients with similar variances results in similar error variances in coefficient quantization. As a lossy coding technique, MVQ-OBA is shown experimentally to outperform both scalar quantization (non-adaptive and adaptive) and direct vector quantization of the transform coefficients. The superiority can be traced to combining vector quantization with optimal bit allocation.

VQ-QT

For a given image, a mean quadtree data structure of image is first built up by successively averaging over blocks of 2×2 pixels. A difference quadtree is then followed

by taking the differences between successive levels of the mean quadtree. Progressive transmission is achieved by sending the difference quadtree starting from the top level. Since the first order entropy of the difference quadtree is shown to be much less than that of the original image, it is possible to transmit the difference quadtree at a much lower bit rate than the original image. To improve the transmission performance, vector quantization is applied to the difference quadtree on a level-by-level basis. The residual errors due to quantization at the higher levels are delivered to the lower levels by an error deliver algorithm, which makes it possible to reprocess at the lower levels the information lost at the higher levels. Excellent quality of the approximation is reported at a bit rate of only 0.6 bits/pixel. The simulation results demonstrate that, of the four presented techniques with a block size of 4×4, VQ-QT achieves the best performance at very low bit rates (< 0.3 bits/pixel).

8.2 CURRENT RESEARCH WORK

Recent investigations shows that the quadtree [91-94] is a very promising technique for progressive image transmission. We briefly describe some of our current research.

Pyramid Transform Image Coding

Block transform coding is a very efficient coding technique because quantization operates on the relatively decorrelated transform coefficients. However, although the coefficients within blocks can be decorrelated by an appropriate transform procedure, there still exist some correlations between the neighboring blocks. Specifically, the pixels near the common boundary between blocks are correlated with one another. In a recent paper [95], the quadtree decomposition presented in chapter 7 is used to remove the correlation between blocks. An image of size $2^n \times 2^n$ first undergoes a two-dimensional discrete block cosine transform of size $2^m \times 2^m$ where $n > m$. The transform coefficients are then reorganized into $2^m \times 2^m$ transform subimages

according to the coefficient order. For each subimage, a mean quadtree is formed by successively averaging over 2×2 neighbouring values. A difference quadtree is then formed by taking the differences between successive levels in the mean quadtree. Like levels of the transform subimages can then be combined into a "concatenated" quadtree. Thus, the top level of the concatenated quadtree contains $2^m \times 2^m$ nodes; each one equal to the mean of the corresponding transform subimage. Progressive image transmission can be achieved by sending the concatenated difference quadtree starting from the top level. Successive approximations can be progressively built up by adding the information of current level to the previous levels. To gain efficiency, a multistage vector quantization presented in chapter 6 is applied to the difference quadtree of the image on a level-by-level basis. The errors due to quantization at the higher level can be delivered to and included in the lower level, as mentioned in chapter 7, which makes it possible to reprocess, at the lower levels, the information lost at the higher levels. Finally, an entropy coder is used to losslessly encode the final residual error image, thus ensuring perfect reproduction of the original image.

Reduced Quadtree

As mentioned, the number of nodes in the quadtree presented in chapter 7 is 33% more than the number of pixels of the original image. Thus, even though the transmission bit rate per node is significantly reduced because of much less entropy in the difference quadtree (Chap. 7), 33% more nodes values compared to transmission of the original image have to be sent. A *reduced* quadtree with the number of nodes equal to the number of pixels can be formed which preserve all the information. Given an image of size $2^n \times 2^n$, a reduced quadtree is formed as follows:

0) Initialization: Let $k = n$ and let level k be the original image.

1) Level $k - 1$ formation of quadtree: For each spatially contiguous,

nonoverlapping, block of 2×2 pixels at level k, find the truncated mean

$$X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} = [\frac{X_{k,i,j} + X_{k,i,j+1} + X_{k,i+1,j} + X_{k,i+1,j+1}}{4}] \qquad i,j = 1,3,...2^k - 1$$

and the three differences

$$\begin{cases} D_{k,i,j+1} = X_{k,i,j} - X_{k,i,j+1} \\ D_{k,i+1,j+1} = X_{k,i,j+1} - X_{k,i+1,j+1} \\ D_{k,i+1,j} = X_{k,i+1,j+1} - X_{k,i+1,j}. \end{cases} \quad i,j = 1,3,...2^k - 1$$

2) Let $k = k - 1$ and if $k \neq 0$, return to step 1.

3) Stop.

Note that only the top node value and the differences need be kept as these contain all the image information. To losslessly and progressively transmit the image, the quadtree is transmitted from the top to bottom level, in other words, all that need send are the top value of the quadtree, $X_{0,1,1}$, and the set of $\{D_{k,i,j}\}$. At the receiving end, the successive reduced-size approximations to the original image are reconstructed from the received data as follows,

$$\begin{cases} X_{k,i,j} = X_{k-1,[\frac{i+1}{2}],[\frac{j+1}{2}]} + [\frac{3D_{k,i,j+1} + 2D_{k,i+1,j+1} + D_{k,i+1,j}}{4}] \\ X_{k,i,j+1} = X_{k,i,j} - D_{k,i,j+1} \\ X_{k,i+1,j+1} = X_{k,i,j} - D_{k,i,j+1} - D_{k,i+1,j+1} \\ X_{k,i+1,j} = X_{k,i,j} - D_{k,i,j+1} - D_{k,i+1,j+1} - D_{k,i+1,j} \end{cases}$$

where $i,j = 1,3,...2^k - 1$.

Two remarks are in order. The first regards the node number of quadtree. Since the number of nodes at each level is given by,

$$1, \qquad k = 0$$

$$\frac{3}{4} \times 2^k \times 2^k, \qquad k = 1, 2, ...n$$

the total number of nodes is equal to

$$1 + \frac{3}{4} \sum_{k=1}^{n} 2^k \times 2^k = 1 + \frac{3}{4} \times 2 \times 2 \times \frac{1 - 2^n \times 2^n}{1 - 2 \times 2} = 2^n \times 2^n.$$

In other words, the number of nodes on the quadtree is exactly equal to the number of pixels of the original image, implying an 1:1 mapping. Secondly, the entropy of the quadtree is likely to be much less than that of the original image because it consists of a set of differences between the pixels. For example, the first order entropies of the quadtree, measured from two test images: face and boat images, are, respectively, 4.7986 bits/pixel and 5.5501 bits/pixel, while the corresponding first order entropies of the original images are, respectively, 7.5178 bits/pixel and 7.5395 bits/pixel. Note that the equivalent entropies of the difference quadtree (chapter 7) defined on a pixel basis for the two test images are, respectively, 5.8487 bits/pixel and 6.8775 bits/pixel.

Nonuniform Transmission Of Quadtree

Progressive image transmission in chapter 7 is achieved by sending the difference quadtree, level by level, starting from the top level. That is, the quadtree is *uniformly* transmitted. However, it is well-known that the image information is nonuniformly allocated over the pixels, implying that some areas may contain more information than others. Therefore, an optimal transmission strategy should send the nodes of quadtree according to their *information content*, that is, the nodes containing more information are sent first. In this way, more detail areas will first appear in the sequence of successive approximations. However, at each level, all the nodes to be sent have to be examined and selected based upon an information criterion. The node selection may results in a large amount of transmission overhead.

8.3 FUTURE RESEARCH WORK

Progressive image transmission is receiving attention for application in interactive

image communications over low bandwidth channels. This thesis provides only a few progressive image transmission techniques. There are still many interesting ways to transmit the image progressively. Some suggestions for future work are discussed below.

1) All the presented progressive image transmission techniques could be extended for encoding color images.

2) Interpolation could be used to improve the quality of approximations, especially the first approximations.

3) Other subjectively correlated distortion measures might be used as the distortion criterion in vector quantization so that the approximations could provide better subjective quality.

4) The optimal bit allocation map in the transform domain (chapter 5 and 6) could be calculated based upon the weighted coefficient variances, where the weighting function is related to the nonuniform response of human visual system to spatial frequencies.

5) Singular value decomposition (SVD) [96-99] may be a useful alternative to progressive image transmission where an image is decomposed into a set of eigenimages. The set of eigenimages in fact correspond to a set of approximations of the image.

6) Knowledge-based systems might be used in searching an optimal transmission path in the quadtree, thus, resulting in optimal nonuniform progressive transmission of the image.

REFERENCES

1. S.L. Tanimoto and T. Pavlidis, "A hierarchical data structure for picture processing", *Comput. Graphics and Image Processing*, vol. 4, pp. 104-119, 1975.
2. S.L. Tanimoto, "Image transmission with gross information first", *Comput. Graphics and Image Processing*, vol. 9, pp. 72-76, Jan. 1979.
3. K.R. Sloan, Jr., and S.L. Tanimoto, "Progressive refinement of raster images", *IEEE Trans. Comput*, vol. C-28, pp. 871-874, Nov. 1979.
4. K. Knowlton, "Progressive transmission of grey scale and binary pictures by simple, efficient, and lossless encoding scheme", *Proc. IEEE*, vol. 68, pp. 885-896, July. 1980.
5. F.S. Hill Jr., S. Walker Jr. and F. Gao, "Interactive image query system using progressive transmission", *Computer Graphics*, vol. 17, pp. 323-330, July. 1983.
6. A. Sanz, C. Munoz and N. Garcia, "Approximation quality improvement techniques in progressive image transmission", *IEEE Jou. on Selected Areas in Commun.*, vol. SAC-2, pp. 339-373, March. 1984.
7. H.M. Dreizen, "Content-driven progressive transmission of grey-scale images," *IEEE Trans. on Commun.*, vol. COM-35, pp. 289-296, March, 1987.
8. P.J. Burt and E.H. Adelson, "The Laplacian pyramid as a compact image code", *IEEE Trans. on Commun.*, vol. COM-31, pp. 532-540, April. 1983.
9. J. Naor and S. Peleg, "Hierarchical image representation for compression, filtering and normalization ", *Pattern Recognition Letters*, 2(1983) pp. 43-46, Oct. 1983.
10. W.D. Hofmann and D.E. Troxel, "Making progressive transmission adaptive", *IEEE Trans. on Commun.*, vol. COM-34, pp. 806-813, aug. 1986.
11. A. Tran, K.M. Liu, K.H. Tzou and E.B. Vogel, "An efficient pyramid image coding system", *IEEE Int. Conf. on ASSP'87*, Dallas, Texas, pp. 744-747, April, 1987.
12. K.N. Ngan, "Image display techniques using the Cosine transform", *IEEE Trans. on ASSP.*, vol. ASSP-32, pp. 173-177, Feb. 1984.
13. E. Dubois and J.L. Moncet, "Encoding and progressive transmission of still pictures in NTSC composite format using transform domain methods", *IEEE Trans. on Commun.*, vol. COM-34, pp. 310-319, March 1986.
14. K.H. Tzou and S.E. Elnahas, "An optimum progressive transmission and reconstruction scheme for transformed image", *Proc. IEEE Int. Conf. on Communications*, Toronto, Ont., pp. 413-418, June 1986
15. S.E. Elnahas, K.H. Tzou, J.R. Cox, R.L. Hill and R.G. Jost, "Progressive coding and transmission of digital diagnostic pictures," *IEEE Trans. on Medical Imaging*, vol. MI-5, pp. 73-83, June, 1986.
16. L. Wang and M. Goldberg, "Lossless progressive image transmission by residual vector quantization", *Proc. The Thirteenth Biennial Symposium on Communications*, pp. c.1.9-c.1.12, Kingston, Ont., June 1986.
17. L. Wang and M. Goldberg, "Progressive image transmission by multistage transform coefficient quantization", *Proc. IEEE Int. Conf. on Communications*, Toronto, Ont., pp. 419-423, June 1986.
18. L. Wang and M. Goldberg, "Lossless progressive image transmission by residual error vector

quantization", *Proc. IEEE Int. Montech'86 Conf. on Antennas and Communications*, pp. 173-176, Montreal, Quebec, Sept. 1986.

19. L. Wang and M. Goldberg, "Progressive image transmission by residual error vector quantization in transform domain", *Proc. IEEE Int. Phoenix Conf. on Computers and Communications*, pp. 178-182, Scottdale, Arizona, Fab. 1987.
20. L. Wang and M. Goldberg, "Progressive image transmission using vector quantization on images in quadtree form," *Proc. Int. Conf. on Digital Signal Processing*, Florence, Italy, Sept. 1987.
21. L. Wang and M. Goldberg, "Progressive image transmission by transform coefficient residual error quantization", accepted for publication in *IEEE Trans. on Communications*.
22. L. Wang and M. Goldberg, "Progressive image transmission using vector quantization on images in quadtree form," submitted to *IEEE Trans. on Communications*.
23. L. Wang and M. Goldberg, "Lossless progressive image transmission by residual error vector quantization," submitted to *IEE Proc. Communications, Radar and Signal Processing*.
24. L. Wang and M. Goldberg, "Transform image coding by vector quantization with optimal bit allocation planes", submitted to *IEEE Trans. on Communications*.
25. C.E. Shannon, "A mathematical theory of communication", *Bell Systems Technical Journal 27*, pp. 379-423, 623-656, 1948.
26. C.E. Shannon, "Coding theorems for a discrete source with a fidelity criterion", *IRE National Convention Record*, Part 4, pp. 142-163, 1959.
27. R.G. Gallager, *Information Theory and Reliable Communication*, John Wiley, New York, 1968.
28. T. Berger, *Rate Distortion Theory*, Prentice-Hall Inc., Englewood Cliffs, NJ, 1971.
29. L.D. Davisson, "Rate-Distortion Theory and Application", *Proc. IEEE*, pp. 156-164, July 1972.
30. Norman Abramson, *Information Theory and Coding*, McGraw-Hill, 1963.
31. R.W. Hamming, *Coding and Information Theory*, Prentice-Hall, Inc., 1980.
32. D.A. Huffman, "A method for the construction of minimum-redundancy codes", *Proc. I.R.E.*, pp. 1098-1101, Sept. 1952.
33. N. Faller, "An adaptive system for data compression", *Record of the 7th Asilomar Conf. on Circuits, Systems and Computers*, pp. 593-597, 1973.
34. R.G. Gallager, "Variations on a theme by Huffman", *IEEE Trans. on Inform. Theory*, vol. IT-24, pp.668-674, Nov. 1978.
35. G.V. Cormack and R.N. Horspool, "Algorithms for adaptive Huffman codes", *Information Processing Letters*, pp. 159-165, March, 1984.
36. D.E. Knuth, "Dynamic Huffman Coding" *Journal of Algorithms*, pp. 163-180, 1985.
37. D.R. Spencer and T. Huang, "Bit plane encoding of continuous-tone pictures", *Symp. on Comput. Process. in Commun.*, Polytechnic Inst. Brooklyn, New York, April, 1969.
38. W.F. Schreiber, T.S. Huang and O.J. Tretiak, "Contour coding of images", *Picture Bandwidth Compression*, T.S. Huang and O.J. Tretiak, Eds. pp. 443-448, Gordon and Breach, New York, 1972.
39. T.S. Huang, "Run-length coding and its extensions", *Picture Bandwidth Compression*, T.S. Huang and O.J Tretiak, Eds. pp. 221-266, Gordon and Breach, New York, 1972.
40. T.S. Huang, "Coding of two-tone images", *IEEE Trans. on Commun.*, vol. COM-25, pp. 1406-1424, Nov. 1977.

41. R. Bellman, *Introduction to Matrix Analysis*, MaGraw-Hill, New York, 1960.

42. R.C. Gonzalez and P. Wintz, *Digital Image Processing*, Addision-Wesley Publishing Company, Inc., 1977.

43. W.K. Pratt, *Digital Image Processing*, Wiley, New York, 1978.

44. W.K. Pratt, *Image Transmission Techniques*, Academic Press, New York, 1979.

45. E.L. Hall, *Computer Image Processing and Recognition*, Academic Press, New York, 1979.

46. A. Rosenfeld and A.C. Kak, *Digital Picture Processing*, Academic Press, New York, 1982.

47. N.S. Jayant and P. Noll, *Digital Coding of Waveform*, Prentice-Hall, 1984.

48. A.N. Netravali and J.O. Limb, "Picture coding: a review", *Proc. IEEE*, vol. 68, pp. 366-406, March 1980

49. A.K. Jain, "Image data compression: a review", *Proc. IEEE*, vol. 69, pp. 349-389, March 1981.

50. F. De Jager, "Deltamodulation: a method of PCM transmission using a one-unit code", *Philips Res. Rep.*, pp. 442-466, 1952.

51. C.C. Cutler, "Differential quantization of communication signals", USA Patent 2-605-361, July, 1952.

52. A. Habibi, "Survey of adaptive image coding techniques", *IEEE Trans. on Commun.*, vol. COM-25, pp. 1275-1284, Nov. 1977.

53. H.C. Andrews and W.K. Pratt, "Fourier transform coding of images", *Proc. Hawaii Int. Conf. System Science*, pp. 677-679, Jan. 1968.

54. W.K. Pratt, J. Kane and H.C. Andrews, "Hadamard transform image coding", *Proc. IEEE*, vol. 57, pp. 58-68, Jan. 1969.

55. H.C. Andrews and W.K. Pratt, "Transform image coding", *Proc. Comput. Processing Communications*, pp. 63-84, Polytechnic Press, New York, 1969.

56. A. Habibi and P.A. Wintz, "Image coding by linear transformation and block quantization", *IEEE Trans. on Commun. Tech.*, vol. COM-19, pp. 50-63, Feb. 1971.

57. W.K. Pratt and H.C. Andrews, "Application of Fourier-Hadamard transformation to bandwidth compression", *Picture Bandwidth Compression*, T.S. Huang and O.J. Tretiak, Eds., pp. 515-554, Gordon and Breach, New York, 1972.

58. J.W. Woods and T.S. Huang, "Picture bandwidth compression by linear transformation and block quantization", *Picture Bandwidth Compression*, T.S. Huang and O.J. Tretiak, Eds. pp. 555-573, Gordon and Breach, New York, 1972.

59. P.A. Wintz, "Transform picture coding", *Proc. IEEE*, vol. 60, pp. 809-823, July, 1972.

60. M. Tasto and P.A. Wintz, "Image coding by adaptive block quantization", *IEEE Trans. on Commun. Tech.*, vol. COM-19, pp. 956-972, Dec. 1971.

61. J.I. Gimlett, "Use of activity classes in adaptive transform image coding", *IEEE Trans. on Commun.*, vol. COM-23, pp. 785-786, July 1975.

62. W.H. Chen and C.H. Smith, "Adaptive coding of monochrome and color images", *IEEE Trans. on Commun.*, vol. COM-25, pp. 1285-1292, Nov. 1977.

63. A. Habibi, "Hybrid coding of pictorial data", *IEEE Trans. on Commun.*, vol. COM-22, pp. 614-624, May, 1974.

64. A. Gersho, "Asymptotically optimal block quantization", *IEEE Trans. on Inform. Theory*, vol.

IT-25, pp. 373-380, July, 1979.

65. Y. Linde, A. Buzo and R.M. Gray, "An algorithm for vector quantization design", *IEEE Trans. on Commun.*, vol. COM-28, pp. 84-95, Jan. 1980.

66. A Buzo, A.H. Gray, Jr., R.M. Gray and J.D. Markel, "Speech coding based upon vector quantization", *IEEE Trans. Acoustics, Speech, and Signal processing*, vol. ASSP-28, pp.562-574, Oct. 1980.

67. A. Gersho, "On the structure of vector quantizer", *IEEE Trans. on Inform. Theory*, vol. IT-28, pp. 157-166, March, 1982.

68. B.H. Juang, D.Y. Wong and A.H. Gray, Jr., "Distrtion performance of vector quantization for LPC voice coding", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-30, pp. 294-303, April 1982.

69. B.H. Juang and A.H. Gray, "Multiple stage vector quantization for speech coding", *Proc. IEEE Int. Conf. on ASSP*, pp. 597-600, April, 1982.

70. H. Abut, R.M. Gray and G. Rebolledo, "Vector quantization of speech and speech-like waveforms", it IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-30, pp.423-435, June 1982.

71. A. Gersho and V. Cuperman, "Vector quantization: a pattern-matching technique for speech coding", *IEEE Communications Magazine*, pp. 15-21, Dec. 1983.

72. R.M. Gray, "Vector quantization", *IEEE ASSP Magazine*, pp. 4-29, April. 1984.

73. M.J. Sabin and R.M. Gray, "Product code vector quantizers for waveform and voice coding", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. ASSP-32, pp. 474-488, June, 1984.

74. V. Cuperman and A. Gersho, "Vector predictive coding of speech at 16 kbits/s", *IEEE Trans. on Commun.*, vol. COM-33, pp. 685-696, July, 1985.

75. J. Makhoul and S. Roucos and H. Gish, "Vector quantization in speech coding", *Proc. IEEE*, vol. 73, pp. 1551-1588, Nov. 1985.

76. E.E. Hilbert, "Joint pattern recognition/data compression concept for ERTS multispectral data", *SPIE* vol. 66, "Efficient transmission of pictorial information", Aug. 1975.

77. G.E. Lowitz, "Image data reduction", *Proc. Int. Conf. on Spacecraft on-Board Data Management*, Nice, pp. 291-294, Oct. 1978.

78. Y. Yamada, K. Fujita and S. Tazaki, "Vector quantization of video signals", *Proc. Annu. Conf. IECE*, p. 1031, 1980.

79. A. Gersho and B. Ramamurthi, "Image coding using vector quantizaton", *Proc. IEEE Int. Conf. on ASSP*, pp. 428-431, May 1982.

80. R.L. Baker and R.M. Gray, "Image compression using non-adaptive spatial vector quantization", *Proc. IEEE Conf. Circuit, Syst., Comput.*, pp. 55-61, 1982.

81. T. Murakami, K. Asai and E. Yamazaki, "Vector quantizer of video signals", *Electron. Lett.*, pp. 1005-1006, Nov. 1982.

82. R.L. Baker and R.M. Gray, "Differential vector quantization of achromatic imagery", *Proc. Int. Picture Coding Symp.* March, 1983.

83. R.A. King and N.M Nasrabadi, "Image coding using vector quantization in the transform domain", *Pattern Recognition Letters*, vol. 1, pp.323-329, July 1983.

84. H.F. Sun and M. Goldberg, "Image coding using LPC with vector quantization", *Proc. Int.*

Conf. on Digital Signal Processing, pp. 5-8-512, Florence, Italy, Sept. 1984.

85. H.M. Hang and J.W. Woods, "Predictive vector quantization of images", *IEEE Trans. on Commun.*, vol. COM-33, pp. 1208-1219, Nov. 1985.

86. M. Goldberg, P.R. Boucher and S. Shlien, "Image compression using adaptive vector quantization", *IEEE Trans. on Commun.*, vol. COM-34, pp. 180-187, Feb. 1986.

87. K. Aizawa, H. Harashima and H. Miyakawa, "Adaptive discrete cosine transform coding with vector quantization for color images", *ICASSP'86*, pp. 985-988, Tokyo, April, 1986.

88. M. Goldberg and H. Sun, "Image sequence coding using vector quantization", *IEEE Trans. on Commun.*, vol. COM-34, pp. 703-710, July 1986.

89. B. Ramamurthi and A. Gersho, "Classified vector quantization of images", *IEEE Trans. on Commun.*, vol. COM-34, pp. 1105-1115, Nov. 1986.

90. R. Aravind and A Gersho, "Image compression based on vector quantization with finite memory", Optical Engineering, July, 1987.

91. H. Samet, "Region representation: quadtree from binary arrays", *Compter Graphics and Image Processing*, vol. 13, pp. 88-93, 1980.

92. H. Samet, "Data structure for quadtree approximation and compression", *Commun. of the ACM*, vol. 28, pp. 973-993, 1985.

93. H. Samet, "Using quadtree to represent spatial data", *NATO ASI Series*, vol. F18, pp. 229-247, 1985.

94. J.L. Mannos and D.J. Sakrison, "The effects of a visual fidelity criterion on the encoding of images", *IEEE Trans. on Inform. Theory*, vol. IT-20, pp. 525-536, July, 1974.

95. L. Wang and M. Goldberg, "Pyramid transform coding using vector quantization", to be presented in *IEEE Int. Conf. on ASSP*, New York, April, 1988

96. H.C. Andrews and C.L. Patterson, "Singular value decomposition (SVD) image coding", *IEEE Trans. on Commun.*, vol. COM-24, pp. 425-432, April, 1976.

97. N. Gargur, "Comparative performance of (SVD) and adaptive cosine transform in coding image", *IEEE Trans. on Commun.*, vol. COM-27, pp. 1230-1234, Aug. 1979.

98. S. Shlien, "A method for computing the partial singular value decomposition", *IEEE Trans. on Patt. Analy. and Mach. Intel.*, vol. PAMI-4, pp. 671-676, Nov. 1982.

99. J.C. Nash and S. Shlien, "Simple algorithm for the partial singular value decomposition", *The Computer Journal*, vol. 30, pp. 268-275, 1987.

100. T.H. Wendler and D. Meyer-Ebrecht, "Proposed standard for variable format picture processing and a codec approach to match diverse imaging devices", *SPIE*, vol. 318, (part 1), Picture archiving and communications systems (PACS) for medical applications, pp. 298-305, 1982.

101. P. Boucher and M. Goldberg, "Transform image coding by vector quantization", *Ninth Symp. on Sig. Proc. and App.*, pp. 629-633, Nice, France, May, 1983.

102. S.P. Lloyd, "Least squares quantization in PCM", *IEEE Trans. on Inform. Theory*, vol. IT-28, pp. 129-137, March, 1982.