CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche 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 film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

AVIS

La qualité de cette microfiche 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, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

Adaptive Vector Quantization of Pictorial Data

by

Paul Boucher

A thesis
presented to the School of Graduate Studies and Research
of the University of Ottawa
in partial fulfillment of the
requirements for the degree of
Master of Applied Science
in
Electrical Engineering

OTTAWA, Ontario, 1984

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

The University of Ottawa requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

CONTENTS

ABSTRACT

This report investigates the use of adaptive vector quantization in compressing monochrome and colour static pictorial data. Adaptive refers to coding an image with an adapted quantizer, as opposed to previous schemes using a single quantizer for coding any type of imagery.

Vector quantization is used to spatially decorrelate image data. An image is decomposed into small non-overlapping blocks, and the pixel grey level values are aligned into a vector. The vector sample space is partitioned into clusters of vectors. Quantization is performed by replacing each sample vector by the cluster representative vector. The representative vectors are used to form a sequence of scalar numbers that define each image block in terms of the representative vectors.

Decoding is performed via look-up tables. This enables video-rate decoding. In turn this permits the use of a small refresh memory.

An image decomposition technique permitting progressive image reconstruction is also proposed. The K-means clustering algorithm is fully investigated for pictorial data distributions, with particular emphasis on a practical implementation.

Various coder organizations are proposed for monochrome picture coding. Preprocessing and local coding is studied for reducing coding time. The coding schemes are analysed according to rate-distortion performance curves, computing time requirement and photographic interpretation. Acceptable subjective results are obtained for data rates as low as 0.7 bits per pixel. Local coding requires only minutes of CPU time.

Spatial and spectral decorrelation of colour pictures is also investigated. A Y,I,Q coordinate conversion is performed; the I and Q planes are then spatially reduced in size. Each plane is then coded as if it were a monochrome picture. Another coding scheme is also proposed. The Y plane is decomposed into blocks ; the components of a block from the Y and subsampled I and Q planes are associated to form a vector. The sequence of vectors are then quantized using the K-means algorithm. This scheme outperforms Y,I,Q plane coding with the added advantage of producing a single sequence of codewords instead of three. Data rates as low as 2.8 bits per pixel for colour pictures, are obtained for acceptable subjective results.

ACKNOWLEDGEMENTS

I wish to acknowledge the following people and organizations who have supported the effort in the completion of this report. Special thanks go to Dr. Morris Goldberg, professor at the Ottawa University, for his continuous support and especially in making my Master's work a wonderful social, cultural and scientific experience. Also I wish to thank the "Ecole Nationale Superieure des Telecommunications" of Paris, France, for receiving me in their "Laboratoire Image" and particularly H. Maitre and F. Schmitt. I thank A. Clainchard and P. Fressard of the "Laboratoire Image", the "Electronique et Physique" and the "Systemes et Communications" groups for providing the computer simulation support. I am also indebted to my technical advisor Dr. Seymour Shlien, presently at the Communications Research Center, Departement of Communications in Ottawa. Finally words of appreciation for my sponsor the National Science and Engineering Research Center of Canada.

This work was supported in part by a NATO fellowship and monitored by the Communications Research Center of Ottawa.

LIST OF SYMBOLS

a	Horizontal block dimension
AD	Absolute distance measure
b	Number of bits in an image
B	Bandwidth
B	Blue component of R,G,B coordinate system
B_o	Number of bits for a codebook
B_w^h	Number of bits per codeword (using a h-tuple)
BAT	CCITT "BAT" picture
BR	Bit rate for an image
c	Multiplicative factor
$c(i)$	Coefficient
C	Compression ratio
$C(i)$	Quantization decision region or cluster set
CBD	City block distance
$d(X,R)$	Distance between X and R
D	Dimensionality of a vector
DPCM	Differential-pulse-code-modulation
e	Perturbation vector
ϵ	Symbol for "element of"
$f(x,y)$	Monochrome image matrix
$f(x,y,z)$	Colour image matrix

F	Number of bits per vector component
F(V)	Histogram frequency count of vector V
FRA	CCITT "FRA" test picture
g	$\log_2 L$
g(x,y)	Image approximation of f(x,y)
G	Green component of R,G,B coordinate system
G(X,V)	Gradient measure between vector X and V
H(V)	Zero-th entropy of process V
H(V(i)/V(i-1))	First order entropy
Hz	Hertz
I	I component of Y,I,Q coordinate system
INT	Interval
K	Number of representative vectors
Kb/s	Kilobits per second
L	Number of grey levels
L'	Number of patterns in image
M	Number of lines in an image
m	$\log_2 M$
Mb/s	Megabits per second
MCCD	Maximum cluster center difference criterion
MSE	Mean-square-error
MSS	Multispectral Scanner
N	Number of columns in an image
n	$\log_2 N$
NC(i)	Number of samples in a cluster set C(i)
nm	Nanometer

NMSE	Normalized mean-square-error
NP	Number of pixels represented by a vector
NT	Number of training samples for clustering
NV	Number of vectors extracted from an image
p(V)	Probability density function of process V
P(V)	Probability distribution function of process V
PCM	Pulse-code-modulation
q	Number of pixels greater than the mean value
Q	Number of quantization levels
q	$\log_2 Q$
QED	Quantization error difference convergence criterion
R	Red component of R,G,B coordinate system
R(i)	Reconstruction or representative vector
r(i,j)	j component of R(i) vector
s	Number of spectral components of an image
SED	Squared Euclidian distance measure
SUBINTV	Sub-interval
T(i)	Decision threshold for quantization
TAB	CCITT "TAB" test
u	Mean value
U	U component of the Y,U,V coordinate system
V	V component of Y,U,V coordinate system
V(i)	Sample vector
v(i,j)	j-th component of vector V(i)
V^D	D-dimensional vector space
w(i)	Codeword (integer label)

X(i)	Training sample for clustering
Y	Luminance component
Z(i)	Initial cluster center
σ^2	Variance or energy measurement

INTRODUCTION

1. OBJECTIVE OF DIGITAL IMAGE COMPRESSION

Digital processing of signals, as opposed to analog processing, is increasingly becoming ubiquitous. The advantages are multiple : greater flexibility, random access in storage, possibility of signal regeneration, ease of multiplexing and encryption, etc. However, one has to pay the price in terms of increase in bandwidth. An analog signal of bandwidth B Hz sampled at the Nyquist rate with 8 bits per sample requires 8*B Hz when transmitted using a digital modulation technique such as phase-shift-keying (PSK), requiring 1 Hz for 2 bits per second.

Data compression provides the opportunity for significantly decreasing transmission costs. These costs can be quite large; in comparaison with a digitized speech signal at 64 kb/s, straight forward digitization of a broadcast television signal requires approximately 60Mb/s.

Digital data compression is the process of reducing the bandwidth or the number of information carrying units used to represent the data, while maintaining acceptable fidelity. The large memory and/or channel capacity requirements for digital image transmission and storage make it mandatory to consider data compression techniques.

Typically compressed data when decoded to it's original form is accompanied by distortion. The efficiency of a compression technique over any other, is measured by: it's data compressing ability for a given distortion (rate-distortion function), its flexibility, robustness and computing complexity.

The objective of this dissertation is to study and extend one relatively new data coding technique, vector quantization for digital image compression. Simulations of various vector quantization compression algorithms are investigated with emphasis given to pratical data system considerations.

2. ORGANIZATION OF DISSERTATION

The dissertation is divided into six chapters. Chapter one reviews the fundamentals of digital image coding, for simple and compact image representation. Chapter two introduces the concept of image coding by vector quantization. Image decomposition, quantizer design, coding and decoding of vector- quantized data, are discussed. A review of prior efforts in vector quantization for compression of speech, multispectral scanner data and pictorial data, is also presented. Chapter three describes the test images and the quantizer design algorithm, used in chapters four and five for simulating image vector quantization.

Experimental results obtained from pictorial data compression using vector quantization, are presented in chapter four for monochrome pictures and in chapter five for colour pictures. The performance of the various coding strategies are compared using rate-distortion curves and photographs of the decoded images. Chapter six concludes the dissertation with a summary and recommendations for future research.

Chapter I

IMAGE CODING

Digital image processing is based on the conversion of a continuous image field (i.e. analog signal) to an equivalent digital form. The following first section briefly describes the mathematical characterization of digital images. Image concepts and some of the mathematical notations used throughout the dissertation are presented. Image digitization, sampling and quantization are discussed.

The high data rate resulting from image digitization makes it mandatory to seek methods for more compact digital representation. The concepts and uses of coding for compression are introduced in section two as techniques for reducing digital image storage or transmission requirements. Image compression techniques are classified in two categories : information preserving and non-information preserving. Emphasis is put on non-information preserving techniques because of their practicability and high compression rate capability. Rate-distortion concepts are considered for analysis of these techniques.

Section three contains a review of non-preserving digital image compression techniques, and section four contains a

summary of the choice of digital image compression techniques.

1.1 DIGITAL IMAGE REPRESENTATION

This section describes the discrete representation of monochrome, i.e. black and white, and colour imagery. Image sampling and quantization fundamentals are discussed.

1.1.1 Discrete Monochrome and Colour Image Representation

The term monochrome image, as used in this dissertation, refers to a two dimensional light intensity function, denoted by $f(x,y)$. The value of $f(x,y)$ at spatial coordinates (x,y) gives the brightness of the image at that point.

In order to be in a form suitable for computer processing, the image function $f(x,y)$ must be digitized both spatially and in amplitude. The result is referred to as a digital image. Spatial digitization corresponds to image sampling, while amplitude digitization to grey level quantization.

Monochrome images contain no colour (or spectral) information, it is the total reflected visual spectrum energy that is measured, also called luminance.

Colour is produced by the mixture of light of different wavelengths reflected by an object. It has been determined [1] that certain wavelengths of red (R), green (G) and blue

(B), when combined with each other in various proportions (intensities) will produce a wider range of colours than any other combination of three colours. These colours are thus referred to as primary colours of light.

For the purpose of standardization, the CIE (Commission Internationale de l'Eclairage) designated in 1931 the following specific wavelength values to the three primary colours : blue = 435.8 nm , green = 546.1 nm , and red = 700 nm. These discrete values are called spectral light bands.

Colour imagery can be regarded as data in the form of several two-dimensional images, each image taken through different spectral windows or bands.

1.1.2 Image Sampling

In digital processing systems, one deals with arrays of numbers obtained by spatially sampling points of a physical source. Image samples nominally represent a physical measurement of a continuous image field, for example, measurements of the light intensity or photographic density. The image sampling rate can be analytically determined using the convolution theorem on band limited data [1]. This is referred to as the Nyquist rate. It is a sufficient but not necessary condition.

The band limitation on imagery depends on its resolution (i.e. the degree of discernable detail) measured by way of

the Fourier transform and known human psycho-visual limitations [1]. The basic aim of image digitization is to obtain a simple and standard representation for random and simple access of the data, thus an equally spaced array of of M by N samples is commonly used.

$$f(x,y) = \begin{vmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & & & \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{vmatrix} \qquad (1.1.2\text{-}1)$$

Each matrix element is referred to as an image element, pixel or pel. It is common practice to let M and N be integer powers of two, because of the binary nature of computer systems.

$$N = 2^n \qquad (1.1.2\text{-}2)$$

$$M = 2^m \qquad (1.1.2\text{-}3)$$

As a rule, a minimum system for general purpose image processing, should be able to display an image size of at least 256 by 256 pixels.

1.1.3 Scalar Quantization

Each image sample f(x,y), is a light intensity measurement. These must also be discretized, by using quantization. In the quantization process the amplitude of a sample pixel is compared to a set of decision levels and replaced by a discrete value called a grey level.

Mathematically, a continuous variable V is mapped into a discrete variable R(i), belonging to a finite set R(i), i = 0 , ... , L-1 of numbers. This process is referred to as scalar quantization. The mapping is generally a staircase function.

Define T(j) , j = 0 , ... , L as a set of increasing transition or decision levels. If V lies in the interval [T(j) , T(j+1)] , then V is mapped to R(j) , $j \in [0,L-1]$. The quantity R(j) , called the reconstruction level, is the quantized value of V and also lies in the interval [T(j) , T(j+1)].

The number of grey levels L is commonly chosen as an integer power of two for full use of binary registers.

$$L = 2^{g} \qquad (1.1.3\text{-}1)$$

The quantizer design problem is to determine the optimum decision and reconstruction levels, given the probability density and an optimization criterion. Since the quantizer

mapping is irreversible, the quantizer introduces distortion which any reasonable quantizer design must attempt to minimize. The optimal quantizer is known as the Max-Llyod quantizer [2]. It is not used for digitization because of complex image probability density function formulation due to the non-stationarity [1] property of images and complex calculation of Max quantization levels. There are several other scalar quantizer designs available that offer various tradeoffs between simplicity and performance.

Typically 64 equally spaced grey levels are sufficient for faithful reproduction of picture type images [3]. Generally, 256 grey levels are used per pixel, for programming reasons, since most computers operate using eight bit bytes.

1.2 COMPACT DIGITAL IMAGE REPRESENTATION

A digital image represented by a simple matrix of quantized grey level values, is commonly referred to as a PCM (Pulse-Code-Modulation) image. The total number of bits , b , required to represent such an image is given by :

$$b = M * N * g * s \qquad (1.2\text{-}1)$$

where s is the number of spectral bands, i.e. s = 1 for monochrome images and s = 3 for colour images.

A typical high quality colour television image has a spatial resolution of 512 by 512 pixels with 256 grey levels

per spectral channel. This translates into 6.3 Megabits or approximately the content of a floppy disk. If a large number of such pictures are to be stored (tens of thousands) a data mountain problem occurs. This last example clearly proves the need for more compact image representation.

1.2.1 Practical Data Systems Considerations

The aim of digitization is to obtain a simple and standard image representation. There is nothing sacred about digitization. In most cases, there is still considerable extraneous, or redundant data remaining ; because of this, image compaction is possible and necessary in many cases where bandwidth, storage or computations are expensive.

The end requirements of the application guide the choice of the compact image representation to use. For live broadcast, compaction or compression techniques are greatly constrained by real time and on-line considerations. For storage applications, encoder requirements are less stringent because much of the compression can be done off line. However, the retrieval (decoding or decompression) should be quick and efficient to minimize turn around or respone time.

Image storage is required [4] for educational and business documents, medical images, engineering drawings, and other types of imagery. Image transmission applications [4] are in broadcast television, remote sensing, teleconferenc-

ing, computer communications, facsimile transmission, and other applications.

In pratice transmission channels are frequently prone to errors [4]. Ideally one would like to jointly solve the problems of compression and error protection but a "catch 22" of coding occurs: when a signal is represented more efficiently, the effect of an error can become far more serious. Thus it is frequently necessary to add a controlled form of redundancy back into the signal (channel coding) for error protection. In this dissertation we constrain ourselves to redundancy removal or source coding, although channel noise immunity is an important advantage.

The application of digital image compression to transmission channels is an economic tradeoff in system design, balancing:

1 - encoder and/or decoder hardware complexity and size

2 - bit rate

3 - channel error performance

4 - flexibility

In the case of broadcast systems, if the decoding is simple, the computational costs of coding are of secondary importance, since they can be amortized over a large number of decoders.

In storage applications, the system error rate encountered is many orders of magnitude lower than the design error rate for a digital channel. As a result, one can consider more complicated, hence more efficient encoding algorithms with less concern about noise sensitivity.

A compression algorithm must be flexible enough to handle both monochrome and colour imagery ; different types of images : pictures, medical data, and others. Benefits would also result if an image could be reconstructed step by step from a coarse to a finer approximation [5]. This would permit rapid image browsing (searching) and especially a choice of image-matrix size (or spatial resolution); thus a low resolution terminal could stop the reconstruction process after its full resolution is reached.

1.2.2 Classification of Compression Approaches

There are two classes of image compression schemes : information-preserving and non-information preserving. Here, information preserving is the synonym of distortion-free or lossless coding.

1.2.2.1 Information Preserving Coding Techniques

The goal of these techniques is the exact reproduction of the original or raw data by using less information carrying units. Raw image data rates do not necessarily represent information rates [6], as discussed earlier. For example,

the typical raw rate for a monochrome image is 8 bits per pixel, whereas the average information rate is given by the entropy, usually measured in bits,

$$H(V) = -\sum_{i=1}^{L} p(i) \log_2 p(i) \qquad (1.2.2.1\text{-}1)$$

where p(i) is the probability that a quantized sample V takes the value R(i), say, from a set of $L = 2^{\mathcal{B}}$ values. This is called the zeroth-order entropy since no consideration is given to the fact that a given sample may have statistical dependence on its neighbours. For monochrome images the zeroth-order entropy is generally around 4 to 6 bits per pixel. The first-order entropy is defined as

$$H(V(k)|V(k-1)) = -\sum_{i=1}^{L}\sum_{j=1}^{L} p(i,j) \log_2(p(i,j)/p(j)) \qquad (1.2.2.1\text{-}2)$$

where p(i,j) is the probability of (V(k) = R(i) and V(k-1) = R(j) ; V(k-1) is a pixel "previous" to V(k) and p(i), p(j) are the marginal probabilities of V(k) and V(k-1), respectively. This is considered as the average information content of V(k) if the state of V(k-1) is known. Second and higher order entropies can be defined similarly. Practical calculation of third and higher order entropies become unfeasible because of excessive computational requirements. For six-bit raw monochrome images, Schreiber [7] has esti-

mated the zeroth-, first- and second-order entropies to be 4.4, 1.9 and 1.5 bits per pixel respectively.

Since according to Shannon's noiseless coding theorem [6], it is possible to code without distortion, a source of entropy H bits per sample using H + e bits per sample where e is an arbitrarily small positive quantity, the maximum achievable compression C defined by

$$C = \frac{\text{Average bit rate of the original data}}{\text{Average bit rate of the encoded data}} = \frac{BR}{BR_e} \qquad (1.2.2.1\text{-}3)$$

is BR / (H + e) = BR / H. Computation of such a compression ratio for images is impractical if not impossible. For example an N by M digital image with BR bits per pixel is one of

$$L' = 2^{BR*M*N} \qquad (1.2.2.1\text{-}4)$$

possible image patterns that could occur. Thus, if p(i), the probability of the i-th image pattern were known, one could compute the entropy, i.e., the information rate for BR bits per pixel N by M images. Then, one could store all L' possible image patterns and encode the image by its address using an entropy coding scheme. However, even for relatively small N and M , L is prohibitively large; e.g., for BR=8

and N=M=3, $L' = 2^{72} > 10^{20}$. Although it is physically impractical to measure p(i), it is believed that the entropy of such an ensemble of images is likely to be very low since only a few of the L' images are likely to occur often [4].

Still entropy coding can be used to remove some image redundancy. These are purely statistical schemes. Each pattern is assigned a variable length code word based on its frequency of occurence. An optimum statistical coding procedure is known as the Huffman code. A pattern may be a single pixel or the difference between neighbouring pixels. These techniques are on the whole rather unattractive, because pictures tend to have non-uniform information content. The coded representation must be buffered in order to be transmitted at a constant rate. Furthermore special precautions must be taken to avoid loss of synchronization due to channel errors. Other techniques, permitting a small controlled amount of degradation can achieve greater compression ratios ; these are known as non-information preserving coding techniques.

1.2.2.2 Non-information Preserving Coding Techniques

At the expense of misrepresenting slightly an image, high compression ratios can be achieved. By exploiting the psychophysical limitations of the human observer the degradation might even improve contrast or other image features and give a more intelligible or pleasant end result.

A multitude of non-information preserving coding schemes exist. Typically these schemes are extensions of three basic techniques : predictive coding, transform coding and block quantization. The performance of these techniques is measured in terms of the data rate versus decoded image quality. There are two subdivisions in image quality : image fidelity and image intelligibility. Image fidelity characterizes the departure of a processed image from a standard image, while image intelligibility denotes the ability of a man or machine to extract relevant information from an image. In other words image fidelity is concerned with small scale differences and image intelligibility involves gross differences between the processed and standard image.

Clearly it is desirable to formulate quantitative measures of image fidelity and intelligibility as a basis for the design and evaluation of imagery systems. A quantitative measure of fidelity should correlate well with subjective testing for a broad class of imagery and be reasonably calculable. It is also highly desirable that the measure be analytic so that it can be used as an objective performance function in the optimization or parametric design of image compression systems. Such quantitative measures were proposed [8] but still have not been proven trustworthy because of a poor understanding of the human visual system. Still a commonly used quantitative error measurement is the

suboptimal, but analytically tractable, mean-square-error measure. Let f(x,y) represent the original M by N pixel image data, and let g(x,y) represent the reconstructed approximation of the same image data after compression. Then the sample average mean-square-error is defined as

$$MSE = \frac{1}{M * N} \sum_{x=1}^{N} \sum_{y=1}^{M} (f(x,y) - g(x,y))^2 \qquad (1.2.2.2\text{-}1)$$

The average energy " σ^2 " in the original image f(x,y) is defined

$$\sigma^2 = \frac{1}{M * N} \sum_{x=1}^{N} \sum_{y=1}^{M} (f(x,y) - u)^2 \qquad (1.2.2.2\text{-}2)$$

where " σ^2 " is in fact the variance and "u" the mean grey level of f(x,y). Thus the percent normalized mean-square-error is defined by

$$NMSE = \frac{MSE * 100}{\sigma^2} \qquad (1.2.2.2\text{-}3)$$

100 percent NMSE would correspond to an image with no contrast, i.e. the coded version of the image would be of uniform luminosity with an intensity equal to "u".

The mean-square-error is not generally meaningful for measuring data quality in an absolute sense. For example,

it would not be meaningful comparing data quality across different compression techniques or across different test images. This is because the observer does not necessarily sum errors over the whole picture as one does with the mean-square-error. The observer bases his estimate on the worst few local areas, especially uniform luminosity areas. However the mean-square-error is very useful for comparing performances from options of the same compression technique and the same test data.

Subjective rating of images, although a non-quantitative measure, is the common evaluation technique. There are two common types of subjective evaluation : absolute and relative. In the former case, observers are shown an image on a good quality television monitor, and are asked to judge its quality according to some predefined rating scale. Comparative or relative evaluation involves the observer ranking of a set of images from the best to worst in a particular group of images. This is the most common evaluation scheme. A commonly used rating scale is the "group goodness" [1] scale, in which an image is numerically rated using the categories shown in Table 1.2.2.2-1 .

TABLE 1.2.2.2-1 Group goodness scale

7. Best in group
6. Well above average for this group
5. Slightly above average for this group
4. Average for this group
3. Slightly below average for this group
2. Well below average for this group
1. Worst in group

It should be emphasized that the results of subjective testing are influenced by viewing conditions : television monitor quality, monitor calibrations, observer's mood, etc.

A good data compressor, emphasizes flexibility : i.e. allows a high performance rate versus quality trade off over wide ranges of compression ratios and applications. This stems from the fact that even for dedicated applications, the user definition of adequate quality is an uncertainty [4] which suggests a need for capability for change.

Interactive redundancy removal, would further aid the user in attempting to reach a lower bound relative to the rate-distortion criteria, based on subjective quality. A

review of non-information preserving coding techniques follow in the next section.

1.3 REVIEW OF NON-INFORMATION PRESERVING CODING TECHNIQUES

Commonly used non-information preserving data compression schemes are examined in this section.

1.3.1 Predictive Coding

Predictive coding is a coding method utilizing point processing, i.e. pixel by pixel redundancy removal [1]. It is used in applications where the cost and speed of the decoder are of primary importance. Its compression efficiency is between 1.5 3 bits per pixel and for monochrome images and its susceptibility to channel noise is high.

A well known predictive coding scheme is DPCM [1], differential pulse-code-modulation. The basic coder consists of a predictor and a quantizer. The predictor uses the previously quantized pixel grey level to predict the next grey level value. The difference between the actual value and the predicted value is quantized using a small number of levels, since the error signal is very small for good predictor models [1]. The decoder uses the coded differences to correct the predicted grey levels based on the previous computed grey levels.

Letting V(i-j) denote the grey level of the j-th previous neighbouring pixels, the n-th order predictor is of the form

$$X(i) = \sum_{j=1}^{n} c(j) * V(i-j) \qquad (1.3.1\text{-}1)$$

The predictor coefficients c(j) are chosen to minimize a quantization error measure, such as the mean-square-error [1]. If the neighbouring pixels originate from the current and previous line, the coding scheme is called two-dimensional DPCM.

The typical subjective errors encountered in DPCM is noise in dark uniform regions (granular distortion), and blurring of sharp edges (slope overload) [1].

The efficiency of DPCM can be increased by updating the quantization levels and prediction coefficients, to match the local statistics of the image. This is referred to as adaptive DPCM coding [9].

The sensitivity of DPCM to channel noise is one of its most serious drawbacks [10]. The errors appear as highly visible streaks because of the inter-dependency of DPCM output coefficients. The streaks are objectionable even at low error rates (0.000001). Techniques reducing the impact of channel noise, reduce the coding efficiency by either adding redundancy [11], or introducing a leak factor into the predictor [Connor].

In summary, though DPCM is a simple coding scheme, the necessary channel noise protection makes it as complex as other coding techniques.

1.3.2 Transform Coding

Transform coding [13] is commonly accepted to be the most efficient coding scheme, with a compression efficiency between 0.5 and 1.5 bits per pixel, and the least sensitive to noise [1]. The drawback is computational complexity for coding and decoding.

The common scheme to compress images using transform techniques, is to divide an image into blocks, typically 16 by 16 pixels. The transform decorrelates the spatial information in the block, concentrating most of the information in a few first coefficients. The transform coefficients are quantized independently according to their energy, and transmitted to the decoder. Some coefficients are simply eliminated while others are quantized with a small number of quantization levels.

Letting f(x,y) represent a block of pixels, the transformation consists of two matrix multiplications

$$f(u,v) = u(x,y)\, f(x,y)\, v(x,y)^t \qquad (1.3.2\text{-}1)$$

where $u(x,y)$ and $v(x,y)$ are unitary matrices which act on the row and columns of $f(x,y)$. It is commonly agreed upon that the best unitary transform for decorrelation is the Karhunen-Loeve expansion [1]. This technique is computationally too expensive for both the coder and the decoder. Unitary transform with known fast computing algorithms for N by N blocks (N a power of two) are used : Hadamard [1], Fourier [15], Cosine [16], etc. The Cosine transform is considered the closest approximation to the Karhunen-Loeve expansion [1]

Although not optimal for decorrelation, small blocks of pixels are used to minimize the number of multiplications per pixel, the required arithmetic precision and the buffering problems caused by variable-length codewords [1].

A bit map, which allocates the number of bits to each frequency component, and the quantized frequency coefficients are transmitted. The allocation of bits is based on the statistics of the picture, assigning more bits to the components with the greatest energy [1]. The method is made adpative [16] by defining "n" bit maps and assigning a particular block of pixel to one of "n" maps based on the statistics of the block.

The coding scheme is reasonably insensitive to channel noise [11,16,17]. A bit error is confined to one specific block and its effect is spread over the entire block. As a result, error rates of 0.0001 are barely visible. The major objection to this scheme is the heavy computer requirements. The ensuing hardware and firmware costs are not trivial even with today's technology. Buffering problems are also non-trivial since variable-length coding is implicit with transform coding : each coefficient is quantized using a different number of levels.

A hybrid technique [18], in which an image is transform coded in the horizontal direction and DPCM coded in the vertical direction, produces low bit rates and requires less computation than two-dimensional transform coding; but still channel noise sensitivity is high.

1.3.3 Block Quantization

Block quantization [19,20,21] divides an image into small blocks (e.g. 4 by 4 pixels) and PCM codes each independently. Block Truncation Coding (BTC) [19,20] efficiently codes monochrome images at bit rates in the order of 1.5 bits per pixel, with good channel immunity. In BTC, a bilevel quantizer is designed for each block, preserving both the sample mean and variance. The reconstruction levels are calculated as follows

$$R(1) = u - \sigma \left[\frac{q}{D - q} \right]^{1/2} \qquad (1.3.3\text{-}1)$$

$$R(2) = u + \sigma \left[\frac{D - q}{q} \right]^{1/2} \qquad (1.3.3\text{-}2)$$

where "u" and " σ " are the sample mean and standard deviation of a block, "D" the number of pixels in a block and "q" the number of pixels greater than "u". The coder transmits a label plane consisting of ones and zeros, the sample mean "u" and standard deviation " σ ". The receiver reconstructs the two-tone blocks from (1.3.3-1) and (1.3.3-2).

Two major artifacts occur in BTC [22] : false contouring, due to only two grey levels in each block, and misrepresentation of some mid-range points due to assignments to either a high or a low value. In summary this technique is more suited to coding multilevel graphics [20].

1.4 SUMMARY

Many image processing applications would have tremendous benefits from compact image representation. Many coding techniques have been developed for redundancy removal. Information preserving techniques have the advantage of exactly reproducing an image at a lower bit rate. Non-information preserving coding techniques introduce a controlable amount of errors with the advantage of high compression.

The choice of one coding technique over another, is made according to the importance of each of the four points discussed in section 1.2.1. In the following chapters a very recent addition to image coding techniques is fully described and analysed: vector quantization.

Chapter II

VECTOR QUANTIZATION

Any analog signal that is to be processed by a digital processor must be discretized : i.e. sampled and quantized. Scalar quantization , which consists in quantizing amplitude samples of an analog signal, is ubiquitous and well understood. Scalar quantization , a special case of vector quantization, corresponds to quantization of uni-dimensional data. The following sections describe the general concepts and use of vector quantization for image compression.

2.1 GENERAL CONCEPTS OF VECTOR QUANTIZATION

In scalar quantization a sequence of continuous amplitude samples is normally quantized on an individual basis. Each member of the sequence is treated as a scalar variable , quantized separately according to some staircase function and assigned a binary code group for storage or transmission. It is possible , however, to reduce the quantization error by jointly quantizing and reconstructing the elements of the sequence.

Consider the D element signal vector V , which is assumed to be a sample of a vector random process, with known n-th order probability density function

$$V = (v(1) , v(2) , \ldots , v(D)) \qquad (2.1\text{-}1)$$

$$p(V) = p(v(1) , v(2) , \ldots , v(D)) \qquad (2.1\text{-}2)$$

Vector quantization involves subdivision of a D dimensional vector space into K non-overlapping decision regions C(k): k = 1,..., K , and determining for each C(k) a reconstruction vector: If V^D represents the vector space, then

$$V^D = C(1) \cup C(2) \cup \ldots C(K) \qquad (2.1\text{-}3)$$

is a possible but not necessary condition. What is necessary is that

$$C(i) \text{ is not empty} \qquad (2.1\text{-}4)$$

$$C(i) \cap C(j) = 0 \quad , i \text{ not equal to } j . \qquad (2.1\text{-}5)$$

each C(i), or decision region, must contain at least one vector sample, and no two C(i) intersect in vector space. Figure 2.1-1 presents an illustration of vector quantization for one-, two- and three dimensional space.

The sample vector V(i) is quantized to the reconstruction vector R(k) if V(i) lies in the decision region C(k).

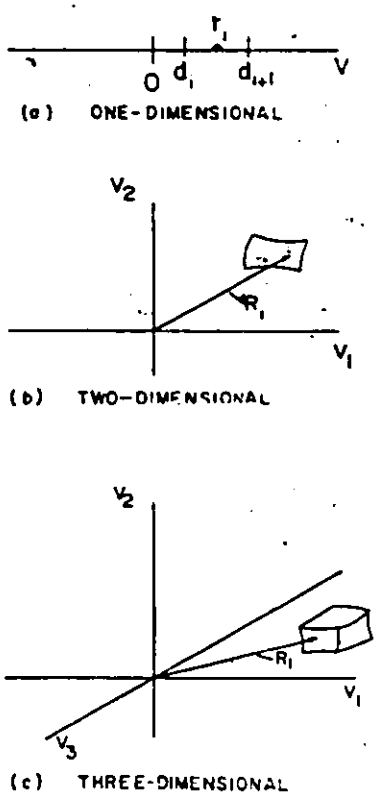


Figure 2.1-1 Vector quantization decision regions.

In this general formulation of vector quantization, a vector V(i) is mapped into a reproduction alphabet

$$R(i) : i=1,2,..,K \qquad (2.1\text{-}6)$$

where

$$R = (r(1) , r(2) , \ldots , r(D)) \qquad (2.1\text{-}7)$$

The individual elements $r(j) : j = 1,...,D$ are not necessarily individually quantized over a set of decision levels : they may be real values.

2.2 IMAGE DECOMPOSITION FOR VECTOR QUANTIZATION

The first step involved in vector quantization of imagery, is decomposing a discrete image of M by N pixels, denoted by $f(x,y)$, into a vector list

$$V(i) : i = 1,..,NV \qquad (2.2\text{-}1)$$

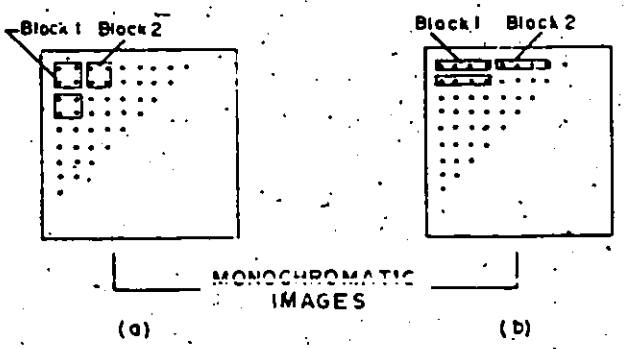
where $V(i)$ is a D dimensional vector (2.1-1) and NV represents the number of vectors generated by the image decomposition process. If each vector component corresponds to a pixel , then

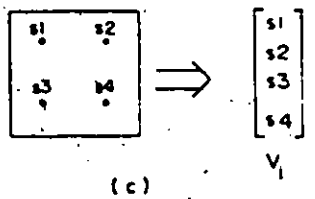
$$NV = \frac{M * N}{D} \qquad (2.2\text{-}2)$$

2.2.1 Block Decompositon

A simple image decomposition scheme partitions a line into sequences of pixels. Each sequence is called a cell or a block. The pixels of each sequence are aligned into a vector. An alternative is to use contiguous pixel blocks : i.e. two neighbouring pixels from the current line and two

from the next [23]. This decomposition scheme will be referred to as block decomposition. Since each block is quantized, this decomposition leading to quantization is closely related to block quantization described in section 1.3.3.





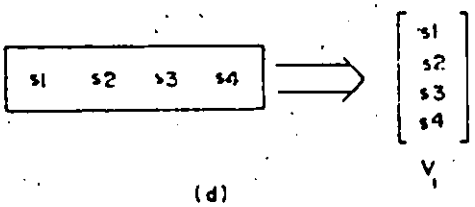


Figure 2.2.1-1. Monochrome image decomposition into blocks of pixels. a) Two-dimensional (D=2*2) and b) one-dimensional (D=3) blocks of pixels. Alignement of pixel grey level values into vectors for c) two- dimensional and d) one-dimensional blocks.

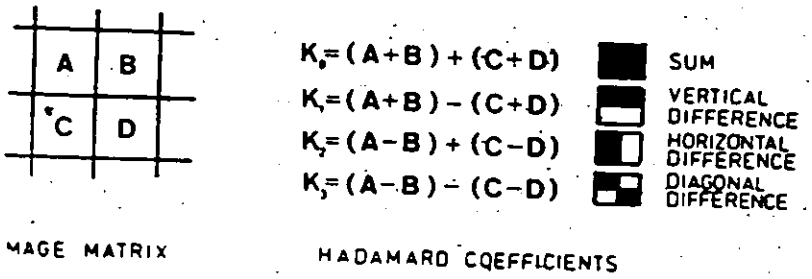
Block decomposition has the disadvantage of permitting only sequential reconstruction of the coded image, i.e. line by line display. It is a disadvantage when a compressed im-

age must be transmitted over a low bit-rate channel; in such a case the image will appear very slowly on the screen. It would be a great advantage to be able to interpret the information well before the complete image is received. As an example, image browsing (searching), frequently performed on information retrieval systems, requires rapid information assimilation. A progressive information display scheme is preferred, where the image is reconstructed, step by step from a coarse approximation to a finer and finer approximation.

2.2.2 The S-transform Image Decomposition Scheme

Step by step reconstruction of an image requires a complex image decomposition scheme. The S-Transform [24], an hierarchical approach to the description of images, is such a technique. It was originally developed for picture archiving. A derivative of the Hadamard transform, it decomposes an image matrix step-wise, into a final coarse picture matrix and an ordered set of contrast detailed information matrices.

The Hadamard transform is applied to non-overlapping blocks of two by two pixels. This results in a sum coefficient (corresponding to the luminosity mean of the block) and coefficients for the vertical, horizontal and diagonal differences.



$$\mathcal{H} = H\begin{bmatrix} A & B \\ C & D \end{bmatrix}H \qquad H=\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

HADAMARD TRANSFORM

Figure 2.2.2-1 The Hadamard transform of a two by two pixel block; the sum coefficient represent the overall block luminosity, the difference coefficients correspond to contrast measurements.

The assembly of the sum coefficients (K(0)), from each pixel block into a two-dimensional matrix, represents exactly the original picture at a quarter of its spatial resolution. In subsequent steps the matrix of sum coefficients is decomposed using the same algorithm.

After S steps of decomposition, the original image of 2^m by 2^m pixels is represented by an image of 2^{m-s} by 2^{m-s} sum coefficients (grey level values) and a hierarchical set of S = (n-m) matrices of difference coefficients. At the last step a single sum coefficient remains, corresponding to the overall luminosity mean of the original picture.

The difference components from each block, at each step, are simply vector-quantized and stored. For two by two pixel blocks, three-dimensional vectors would be produced.

The reconstruction process is the reverse. It is started by first retrieving the coarse image of the last step of decomposition and then the corresponding transform difference coefficients. The four coefficients of a block are inversely Hadamard transformed to obtain a finer resolution picture. The algorithm is repeated until the full resolution image is obtained. Thus a coarse image representation is instantaneously received with gradually added details, permitting the user to abort transmission at any desired detail level. This allows for browsing rapidly through stacks of high resolution pictures.

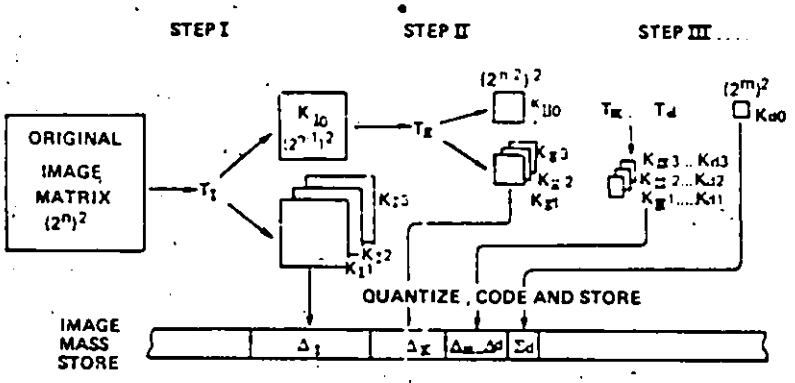


Figure 2.2.2-2 The S-transform decomposition process. (Figure extracted from [24]).

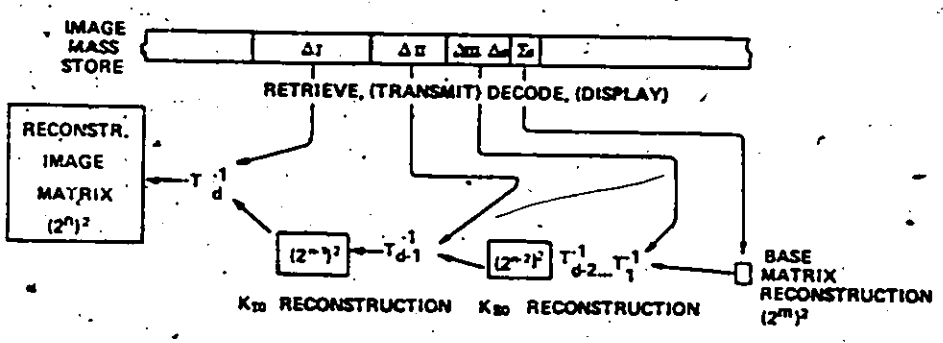


Figure 2.2.2-3 The S-transform reconstruction process. (Figure extracted from [24]).

It is not neccessary to completely decompose an image for transmission. Only a few steps of decomposition can be performed without loss of information.

This technique is best adapted to both the data flow needs of the human observer and low resolution image display terminals. It is thus useful even for high data rate systems where a multitude of different resolution receiver terminals exist.

The Hadamard transform has the advantage of a simple implementation. It only requires : add, subtract and shift operations.

2.3 VECTOR QUANTIZER DESIGN

Vector quantizer design, for a vector random process (2.1-2), is the determination of a set of reconstruction vectors $R(i)$, $i=1,..,K$, minimizing a quantization error measure.

In scalar quantization, the reconstruction values are the centers of quantization intervals. The quantization problem arises in optimally segmenting the sample space into quantization intervals and determining the interval's center.

It is possible to derive analytically the expression of the optimum reconstruction vectors $R(i)$, for a fixed decision region $C(i)$, for a specified quantization error and known probability density function [1]. Although possible, the evaluation problem is overwhelming. Firstly, no quantization error formulation is well related to a subjective error measure of visual data; secondly, the joint probability density function $p(V)$ (2.1-2) is usually not available; thirdly, the simultaneous evaluation of the quantization region boundaries and region centers [1,2] is formidable for non-trivial $p(V)$.

Basically, a quantizer design algorithm requiring no "a priori" knowledge of $p(V)$ is needed for vector space partitioning. Such algortihms are used in pattern recognition, a field of data analysis [25]. These algorithms "train" on an input sequence (vector list), representative of the vector

random process, and generate a set of useful measurements, among other representative vectors.

An important class of pattern recognition algorithms, are clustering techniques. The process of clustering is best explained by representing the training data set in Cartesian space. Vectors of grey level values, are represented by a vector of integer numbers. The D-dimensional vectors can be plotted as points in a Cartesian space. Pattern vectors emanating from a near-constant lumihosity image region, produce groups of vectors closely assembled. These separable groups of vectors will hereafter be called "clusters".

Since in most natural images, there is high redundancy, a large number of clusters is expected.

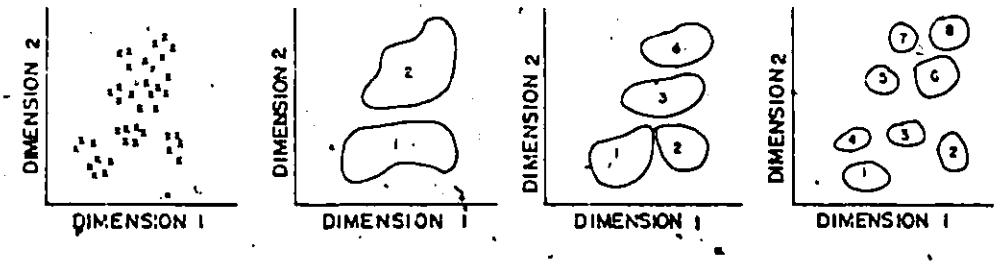


Figure 2.3-1 Clusters of vector points in Cartesian space. a) Measurement vectors { V(i) : i=1,...,NV}. b) Two clusters. c) Four clusters. d) Eight clusters.

Clustering algorithms use this property of cluster separability to extract features from the clusters. The natural boundaries between clusters can be used as thresholds for quantization. Unsupervised clustering algorithms automatically adapt themselves to a training sequence; hence, each training sequence produces a different set of clusters.

Clustering algorithms have been extensively studied for Multispectral Scanner (MSS) imagery classification and combined classification/compression [26,27]. Classification is the process of extracting from a data set, a usually small number (20 to 30) of relevant features and ignoring isolated vectors . Our concern is a partitioning scheme optimal for quantization, not classification accuracy.

Clustering schemes can either be parametric or non-parametric. An example of a non-parametric scheme is Graph-Theoretic clustering [28]. This class of clustering algorithms identify generally shaped clusters of points in metric space. They seek the modes and the valleys of a histogram distribution, assigning valleys as boundaries between clusters. Any cluster shape is thus possible, clustering being totally dependent on the frequency of occurence of the training samples. This is a disadvantage for quantization purposes. The aim in quantization, is to minimize a quantization error measure; cluster separation is not a necessity, only a practical method to generate quantization regions.

For example, in many cases, clusters produced by graph-theoretic clustering, will have very large variances. This results in heavy false contouring in the image reproduction; a major objectionable artifact in image quantization, especially for low luminosity regions. This clustering technique is more applicable for automatic interpretation applications, where cluster seeking and separability is the objective.

Parametric clustering is based upon the minimization or maximization of a performance criterion. By its aim, parametric clustering is more amenable to quantization. One common type of criterion is a measure of similarity of vectors, making use of a distance measure. The separation of space clusters is determined, by minimizing intra-cluster distance and maximizing inter-cluster distance. Algorithms developed using these concepts are the LBG vector quantization algorithm [29] and the Hilbert cluster/compression algorithm [26]. Both use in principle the K-Means clustering algorithm [25].

2.3.1 K-means Clustering

This clustering algorithm [25] is an iterative process minimizing a distance measure between a sample vector and a cluster center. The clustering algorithm is as follows:

Step 1) Choose K initial cluster centers : $R_1(1)$, .., $R_1(K)$ where $R_t(k) = r_t(k,1)$; .., $r_t(k,D)^1$, and "t" is the iteration number.

Step 2) At the t-th iterative step, distribute the training samples:

$$X(i) : i=1,2,..,NT \qquad (2.3.1\text{-}1)$$

among the K cluster domains using the relation :

$$X(i) \in C_t(k) \quad \text{if}$$

$$d(X(i), R_t(k)) < d(X(i), R_t(j)) \qquad (2.3.1\text{-}2)$$

for all $k = 1,2,..,K$; $j = 1,2,..,K$ where k not equal j

$C_t(k)$ denotes a cluster set, $R_t(k)$ the cluster center and $d(X(i), R_t(k))$ the distance between $X(i)$ and and $R_t(k)$.

The Euclidian distance is chosen if the mean-square quantization error criterion is to be minimized. Ties in (2.3.1-2) are resolved arbitrarily.

Step 3) Calculate the K new cluster centers : $R_{t+1}(k)$, $k=1,..,K$, based on the statistics of $C_t(k)$. If the mean-square quantization error criterion is to be minimized, the optimal cluster centers are calculated using the sample mean.

$$R_{t+1}(k) = \frac{1}{NC(k)} \sum_{X(i) \in C_t(k)} X(i) \qquad (2.3.1\text{-}3)$$

$$k = 1,2,..,K$$

where NC(k) is the number of samples in $C_t(k)$.

Step 4) Reiterate to step two until the process converges.

Convergence is obtained when the cluster centers from the last iteration are exactly the same as the current iteration. After convergence the last cluster centers are called reconstruction vectors and used for quantization.

The behaviour of the K-means algorithm is influenced by the number of cluster centers specified, the choice of initial cluster centers, the distance measure used and of course, the geometrical properties of the vector space data. Although no general proof of convergence exists for this algorithm, it can be expected to yield acceptable results when the data exhibits characteristic clusters which are relatively far from each other. This algorithm produces a quantizer meeting necessary but not sufficient conditions for optimality.

Improved clustering is possible by using intra-cluster and inter-cluster distance measures for splitting, merging and combining of clusters, at each iteration. A clustering technique using these principles is ISODATA [25].

The disadvantage of such a scheme is the necessary heuristics for decisions on splitting, merging and combining of clusters. Furthermore, extra computation effort is required for calculation of the inter and intra-cluster distances.

This type of clustering technique is more suited to applications allowing a variable number of clusters, such as in automatic interpretation, and where computational complexity is of secondary importance.

2.4 QUANTIZATION AND CODING OPERATION

It was proposed in the last section to use the cluster centers, from the last iteration of the K-Means algortihm, as representative vectors. The components of each of these vectors are real values. If the coded image is to be displayed, then the vector components must be quantized to integer values. The usual dynamic range of grey level values is 256, eight bits per vector component.

Once the reconstruction vectors are determined and quantized, the vector list generated by the image decomposition process must be vector quantized. Each vector V(i), is assigned to the closest R(k), using the same distance measure d(V(i),R(k)) as in the quantizer design algorithm.

Efficient coding is achieved by assigning an integer number or label, 1 to K, to each R(k). Each vector V(i) is coded not by its representative vector R(k), but by a binary codeword, which can simply be the quantization region label 1 to K. More efficient coding is obtained by entropy codes, such as a Huffman code [30], but at the expense of variable length coding, producing non-trivial buffering problems, for transmission.

Thus, a vector quantized image is coded by a reproduction alphabet R(k) : k=1,2,..,K and a list of integer labels, or codewords

$$w(i) : i=1,2,..,NV \qquad (2.4\text{-}1)$$

where w(i) is an integer number from the set 1,2,..,K .

The reproduction alphabet is also called the codebook, the dictionnary or the overhead.

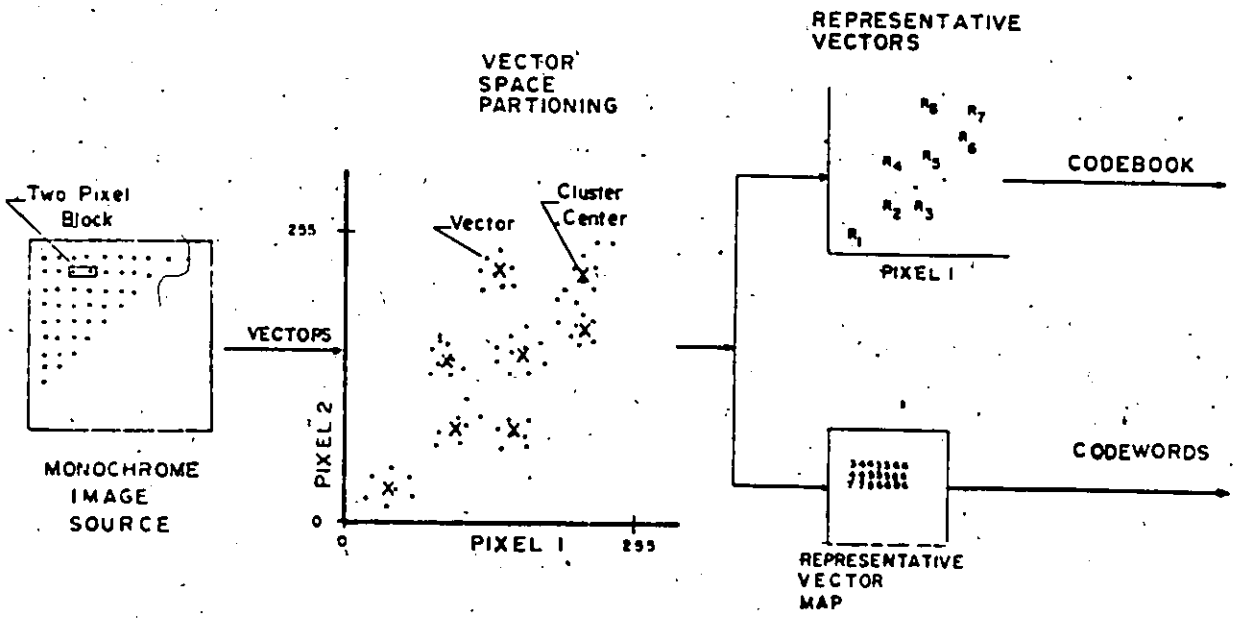


Figure 2.4-1. Vector quantization coding process.

2.5 VECTOR-QUANTIZED DATA DECODING

Vector quantization offers implicitly, simple decoding. Each codeword w(i) is used to look-up the corresponding reconstruction vector R(k), from a known codebook. For block decomposed images each R(k) is then simply restructured into displayable form.

Look-up table decoding has an added advantage : real time decoding implementation. In a conventional image display system, a large refresh memory (e.g. 6 Megabits for a high quality colour image) contains a full PCM image representation ready for digital to analog conversion.

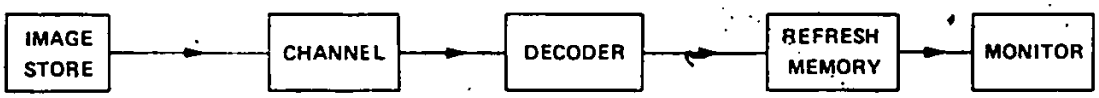


Figure 2.5-1 Conventional image decoder with monitor.

Vector quantization decoding is possible at video rates, since only memory look-ups are performed. Video-rate decoding does not require a full size refresh memory. The refresh memory must only be large enough to contain the coded representation of an image.

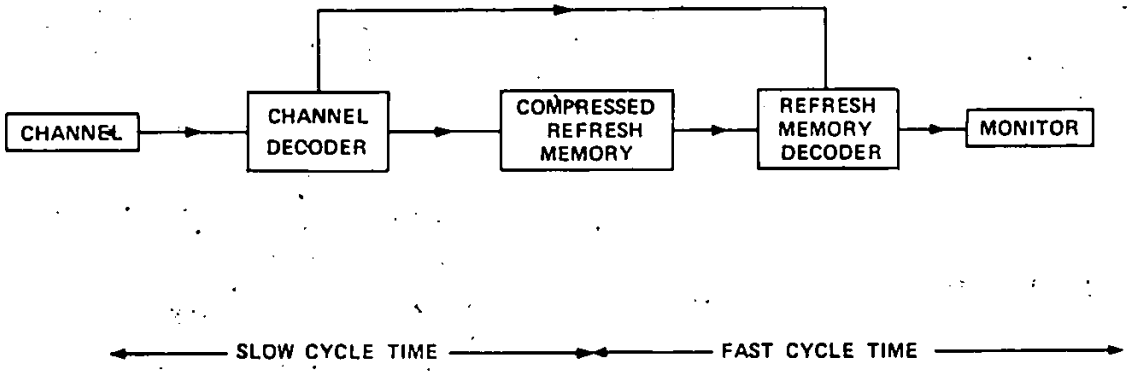


Figure 2.5-2 Image decoder block architecture with a compressed refresh memory.

The decoder is composed of two parts. The first part operates at low speed and interfaces to the channel. The second part operates at the raster refresh rate and is situated between the small refresh memory and the monitor. The decoder, for vector-quantized data, is simply look-up tables containing the reconstruction vectors (2.1-6).

A more detailed block diagram of the compressed refresh memory vector quantizer decoder is illustrated in figure 2.5-3.

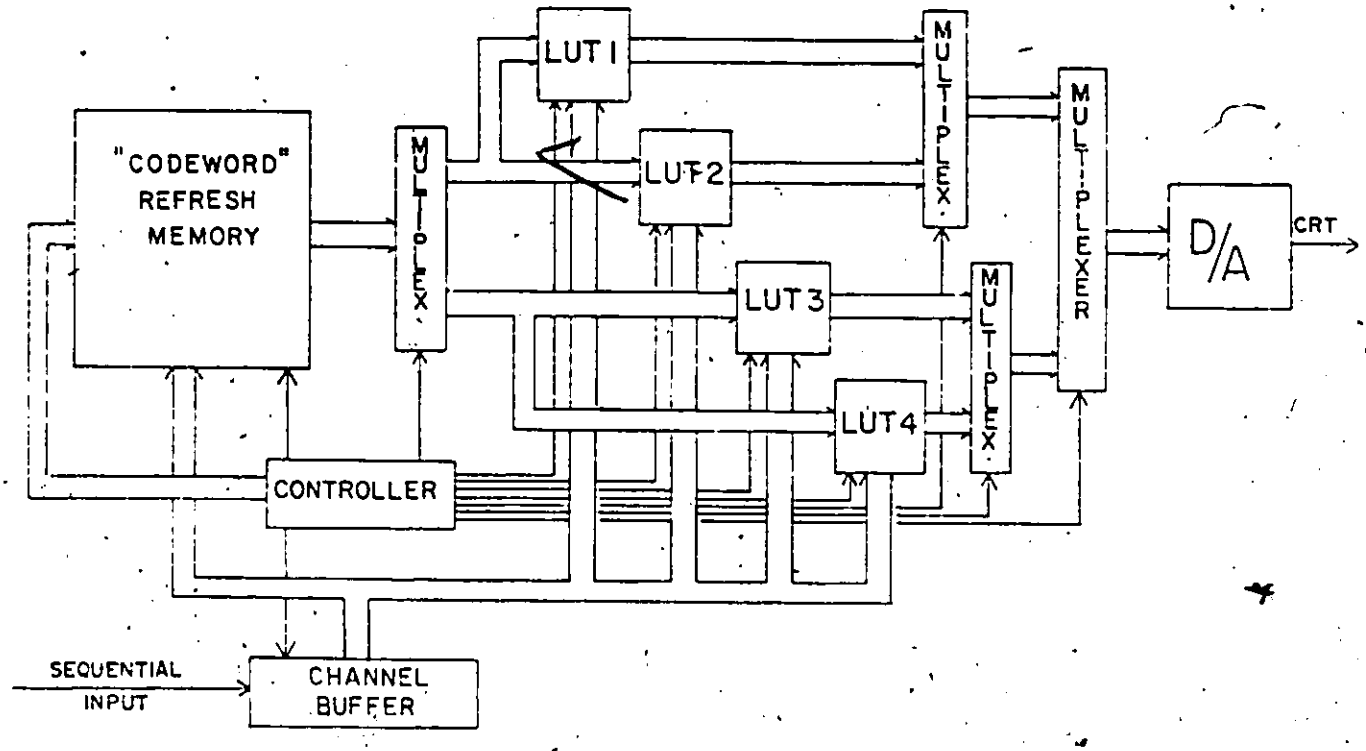


Figure 2.5-3 Detailed video rate decoder architecture for vector quantized data.

As can be observed a simple and low cost decoder architecture is possible. In this example, the picture is reconstructed using two by two pixel blocks. Each line of the codeword refresh memory is read twice to generate two lines of picture grey level values. Each codeword is transmitted to two LUTs (Look-Up Tables) to generate two neighboring pixels, which are automatically fed to the D/A (Digital to Analog) converter and the CRT (Cathode Ray Tube).

With this architecture the refresh memory size is reduced by a factor of 4 (the number of pixels represented by a block). For example, when coding with four by four pixel blocks, the refresh memory size is reduced by a factor of at least 16, if K is smaller or equal to the number of grey levels in the original image.

2.6 PRIOR STUDIES ON VECTOR QUANTIZATION

Two types of digital signals have been compressed successfully using vector quantization : speech and imagery.

2.6.1 Speech Compression

Linear predictive coding (LPC) [31] is one of the methods currently used to compress speech [29,31,32,33]. In this technique short segments of speech, typically 20 milli-seconds, are treated one at a time. The first step in coding involves finding an all pole filter GM(z) which best de-

scribes the segment. The next step concerns the compression or quantization of the coefficients describing the filter. The traditional method quantizes these coefficients one at a time, hence the term scalar quantization.

Recently, there has been interest in quantizing the D coefficients "en bloc" ; that is, by treating them as vector values in a higher dimensional space. NV segments of speech are first analyzed to give NV D-dimensional vectors. Using a clustering algorithm such as K-means, K clusters which best represent the NV vectors are found. For each cluster the mean value is then usually taken as the most representative value of the cluster. A codebook or dictionary of these representative values is then transmitted. As each segment of speech is analyzed, the closest cluster is found and the corresponding cluster label is transmitted.

Vector quantization of the coefficients can yield significant savings in the transmission rate when compared to scalar quantization. Reductions of 2 to 1 are quoted in the literature.

2.6.2 Image Compression

Image compression using vector quantization, was first investigated on Multispectral Scanner (MSS) data [26,27], under the name of the Cluster/Compression algorithm. Compression algorithms developed for MSS data can be used ef-

fectively in most cases on colour pictures [26], for removal of redundancy between the spectral channels.

Monochrome image compression [23] was recently studied with the Linde Buzo and Gray vector quantization algorithm [29].

2.6.2.1 Multispectral Image and Colour Picture Compression

i) The Cluster-Compression Algorithm

Hilbert [26,27] introduced in 1975, a joint feature extraction/compression scheme for Multispectral Scanner(MSS) data for on-board (i.e., satellite, plane, etc.) applications. The compression algorithm is as follow$:

1) divide a MSS frame into equal size subimages, typically 16 by 16 pixels.
2) use the s-measurements (spectral reflectance values) of each pixel as a vector, and determine K representative vectors, using the K-means algorithm cluster centers.
3) replace each pixel-vector by a label number referencing one of the K representative vectors.
4) transmit the K representative vectors, and the label list for each subimage.

Decoding is performed by table-look-up as described in section 2.5.

Subimages, instead of a complete MSS frame, are used for coding for faster processing time. This is because fewer representative vectors are required for small image sources, and the K-means processing time is proportional to K. Optimum subimage sizes are 576 pixels for bit rates greater than 1.2 bits per pixel per band, and 256 pixels for lower bit rates.

Compression ratios of 7 to 1 are obtained for colour pictures : i.e. 3.75 instead of 24 bits per pixel for the "Kodak girl" picture. The algorithm performs better than hybrid coding [34] or adaptive Fourier or Hadamard coders [26] using zonal quantization.

Adaptive subimage coding produces a further gain in compression of 0.75 bits per pixel for the "Kodak girl" [26], at the expense of increased coder/decoder complexity. The improvement results from coding each subimage with a different number of representative vectors, chosen according to subimage statistics.

Postprocessing further decreases the bit-rate. Entropy coding results in a 20 percent gain [26]. Simple label plane subsampling, results in better performance than adaptive cluster/compression coding. Cascaded clustering (cascaded vector quantization) reduces the correlation between the codebook of neighbouring subimages. The technique consists in vector quantizing the representative vectors from a

sequence of subimages, into a smaller set, and using it as a codebook for the subimages of the sequence. Hilbert affirms this postprocessing technique to be non-efficient.

ii) A Practical Implementation of the Cluster/compression Algorithm

Lowitz [35,36,37] adapted Hilbert's algorithm [26] for a practical implementation. For pratical buffering considerations, a line of Multispectral Scanner (MSS) data is used as a subimage. Each pixel (D-dimensional vector) is first Karhunen-Loeve transformed. Only the two to four first transform coefficients for each pixel are conserved. 16 representative vectors are determined, for each line, using the K-means algorithm.

Once 1024 representative vectors are accumulated : i.e. after 64 lines, these are reduced to 128 by cascaded clustering. Hilbert's misfortune with cascaded clustering is quoted [37] to be eliminated by using only distinct representative vectors, for clustering. After 8 groups of 64 lines are processed, cascaded clustering is reapplied to generate less than 256 clusters. Typically 11 minutes are required, using a IBM 370/158, to code a 512 by 512 pixel image.

Lowitz affirms that adaptive coding, by varying the number of representative vectors for each subimage, results in

minimal bandwidth saving for an increase in system complexity, due to the problem in handling variable-length codewords.

Compared to DPCM based compression strategies, the cluster compression scheme is relatively noise immune. Only the codebook, of representative vectors, has to be protected. This amounts to 5 to 6 percent [36] of the total number of bits to transmit. If the representative vector lists are not compressed by cascaded clustering, then a simple error correction scheme is sufficient: if a codebook is known to have been altered, then the preceding codebook (of the previous line) can be used without great damage, because of the great redundancy between successive lines.

iii) Improved Cluster/Compression Algorithm

MacCalla and Chang [38] improved Hilbert's adaptive Cluster Coding scheme, using a modified multi-dimensional extension of a mode-seeking estimator due to Koontz et al. [28] followed by a weighted minimum distance classifier [25]. The estimator is applied iteratively to 16 by 16 pixel local sources of Landsat Multispectral Scanner data. It selects the number and relative positions of the modes of the data based on the local source variance. Usually 3 iterations are required. The pixels of the subimage are then assigned to the closest mode.

The technique outperforms Hilbert's algorithm by an average of 2dB (signal to noise ratio). The authors affirm that the improvement results from Hilbert's algorithm's improper initial choice of cluster centers and inability to select isolated points as clusters, using an Euclidian metric.

iv) Colour Picture Compression

Vector quantization of colour pictures is often called colour quantization, when the vectors extracted from an image correspond to the R,G,B (or other) coordinates of a pixel. Heckbert [39] studied colour quantization, for reduction of the refresh memory size. A typical colour image requires 24 bits per pixel; 8 bits per colour coordinate. Heckert was able to obtain a reduction of 3 to 1, by using only 256 colours as representative vectors. A suboptimal quantizer design was used to obtain fast coding. This is well below Hilbert's compression result. Heckbert also proves the viability of using dithering for compression, on colour imagery, to obtain compression ratios of 10 to 1. Dithering is the process of first poorly quantizing an image, and then adding noise to smooth out false contours.

2.6.2.2 Monochrome Picture Compression

Gersho et al. [23], investigated monochrome picture vector quantization using blocks of a by b pixels, as a*b-dimensional vectors. The set of representative vectors

is generated using the algorithm proposed by Linde et al. [29].

The technique assumes that a codebook of K representative vectors or templates, already exists. An image is partioned into NV blocks of a by b pixels. Each is vector-quantized by searching for a best match in the codebook, according to the mean-square-error criterion. The codeword identifying the best match is transmitted.

The codebook is generated using a training set, statistically representative of the images to be quantized. It is generated once and used to code any image.

Compression ratios of 10 are quoted, but the decoded pictures are of questionable subjective reproduction quality.

The most noticeable coding artifact is a "staircase effect" on the edges, following the block contour ; the higher the block dimensionality the worse the artifact. Instead of using a more subjectively optimal and more complex distance measure for the LBG [29] algorithm, a segmented codebook approach was investigated to improve subjective quality.

Each vector is classified as an edge or smooth block by measuring the intra-block contrast:

$$\frac{(\ IMAX\ -\ IMIN\ )}{IMAX} \qquad (2.6.2.3\text{-}1)$$

IMAX and IMIN referring to the maximum and minimum grey level value. A contrast threshold of 40 percent, is used for classifying patterns as edge or non-edge. Then a LBG quantizer [29] is designed for each block type, the edge codebook containing 75 percent of the total representative vectors. The final codebook is a concatenation of the codebooks. Subjective quality is quoted to improve slightly.

Two very recent papers [40,41], extend the Gersho coding algorithm, by preprocessing schemes. The Cabrera et al. [40] algorithm preprocesses the image vector list, by reducing the correlation between neighbouring blocks using predictive coding. A current vector is compared to a predicted vector and the error vector is quantized using Gersho's technique. Very poor subjective quality is obtained, producing streaking errors similar to DPCM data corrupted by channel noise.

Baker et al. [41] preprocessed each vector to remove internal block correlation. The block mean grey level is subtracted from each pixel, and the resulting "difference vector" is used for quantization. The codewords and the mean of each block are transmitted. This differential vector quantization technique is quoted to slightly outperform Gersho's grey level vector quantization technique.

2.7 SUMMARY OF VECTOR QUANTIZATION CODING

The vector quantization process can be modelled as a sequence of five operations, as shown in the following figure.

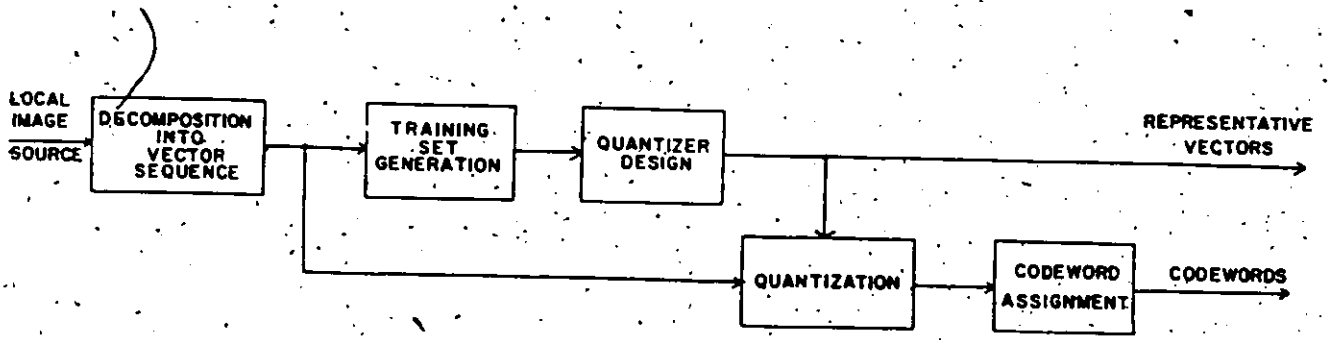


Figure 2.7-1. Block diagram of vector quantization operations.

The decomposition operation maps the input data, an image source, from the pixel domain to a vector domain. The decomposition technique is chosen according to the type of image reconstructrion required : line by line, progressive reconstruction, etc.

The second operation maps the vector list into another domain (e.g. transform domain to remove redundancy), to generate a training set, which can be more efficiently used for quantizer design.

The quantizer design operation generates a reproduction alphabet, or codebook, based on the training set. The quantization operation rounds off the reconstruction vectors and maps each datum to one of the possible reconstruction vector values. The coder assigns a codeword to each quantizer out-

put, for example a Huffman code, to efficiently represent the quantized data.

Decoding can be performed at video rates, only table look-up decoding is required, for block decomposed images. This effectively reduces the refresh memory size by a factor proportional to the block size (in pixels) used for decomposition; thus vector quantization is especially suited for coding systems requiring cheap decoders.

2.7.1 Adaptive vs. Non-adaptive Quantization

Two vector quantization strategies are possible : adaptive and non-adaptive quantization.

In non-adaptive quantization a single reproduction alphabet (2.1-6) is generated and used for all random data sequences to be quantized. The generation of a representative training set, of a vector random process (2.1-1), for quantizer design is as difficult as determining its probability density function (2.1-2) [1].

An example of the use of non-adaptive vector quantization is speech digitization in telephone systems. All voices source (any caller) are quantized using the same scalar quantizer (compander). In non-adaptive quantization, although the data are represented by a list of codewords (2.4-1) and a reproduction alphabet (2.1-6), only the codeword list is used for transmission; the receiver is assumed to already have a copy of the reproduction alphabet.

The determination of a representative training sequence (2.3.1-1), or a probability density function (2.1-2), assumes stationary data statistics, which is not the case for imagery [42] or speech [31]. Gersho [23] studied non-adaptive quantization of imagery. The results although remarquable for this simple coding procedure, are subjectively questionable. Furthermore the coding efficiency is not assured, because of the non-stationary statistics of imagery [23].

The problem is overcome by generating a reproduction alphabet, for each image source to be quantized. This is referred to as adaptive quantization. An "optimal" quantizer is designed to automatically match the histogram distribution of the input data. Hilbert's compression algorithm [26] used this principle for colour image coding.

The disadvantage of adaptive quantization, is increased coder complexity. A quantizer must be designed for each image source. The decoding process is the same for adaptive or non-adaptive quantization.

In the following chapters, only adaptive vector quantization is studied, since the best practical quantizer is always assured.

Chapter III

QUANTIZER DESIGN ALGORITHM

In the second chapter the K-means [25] clustering algorithm was described and proposed for quantizer design. The vector space partitioning produced by this algorithm depends on : 1) the vector data distribution, 2) the choice of initial cluster centers, 3) the choice of the distance measure to minimize, and 4) the number of required clusters K. In this chapter initialization techniques and the metric impact are investigated. Measures of convergence are also defined. The influence of the data distribution and the number of specified clusters, on data quality, is investigated in the following chapters. A description of the test pictures, used in this and the following chapters, is first given.

3.1 TEST PICTURES

Three R,G,B (CIE standard colours) colour pictures are used for testing adaptive vector quantization. These are CCITT standard test pictures, digitized by the CCETT (Centre Commun d'Etudes des Telematiques et des Telecommunications) of France. They were graciously made available, by the Laboratoire Image of the Ecole Nationale Superieure des Telecommunications of Paris, France.

The CCETT pictures are digitized to 604 columns by 562 lines, and 24 bits per pixel, i.e. 8 bits for each of the colour coordinates. The monochrome version of each picture is generated using the Y linear luminance transformation [1].

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \qquad (3.1\text{-}1)$$

Portions of these picture, 256 by 256 pixels, are used for testing the adaptive vector quantization algorithm. A face type picture, designated "FRA", was chosen because of its ubiquity in the field of image coding. Another picture, denominated "TAB", was chosen because of its large smooth surfaces and straight edges, similar to graphic type imagery. A picture containing a larger amount of detail was also chosen : "BAT". The monochrome pictures are shown in figure 3.1-1.

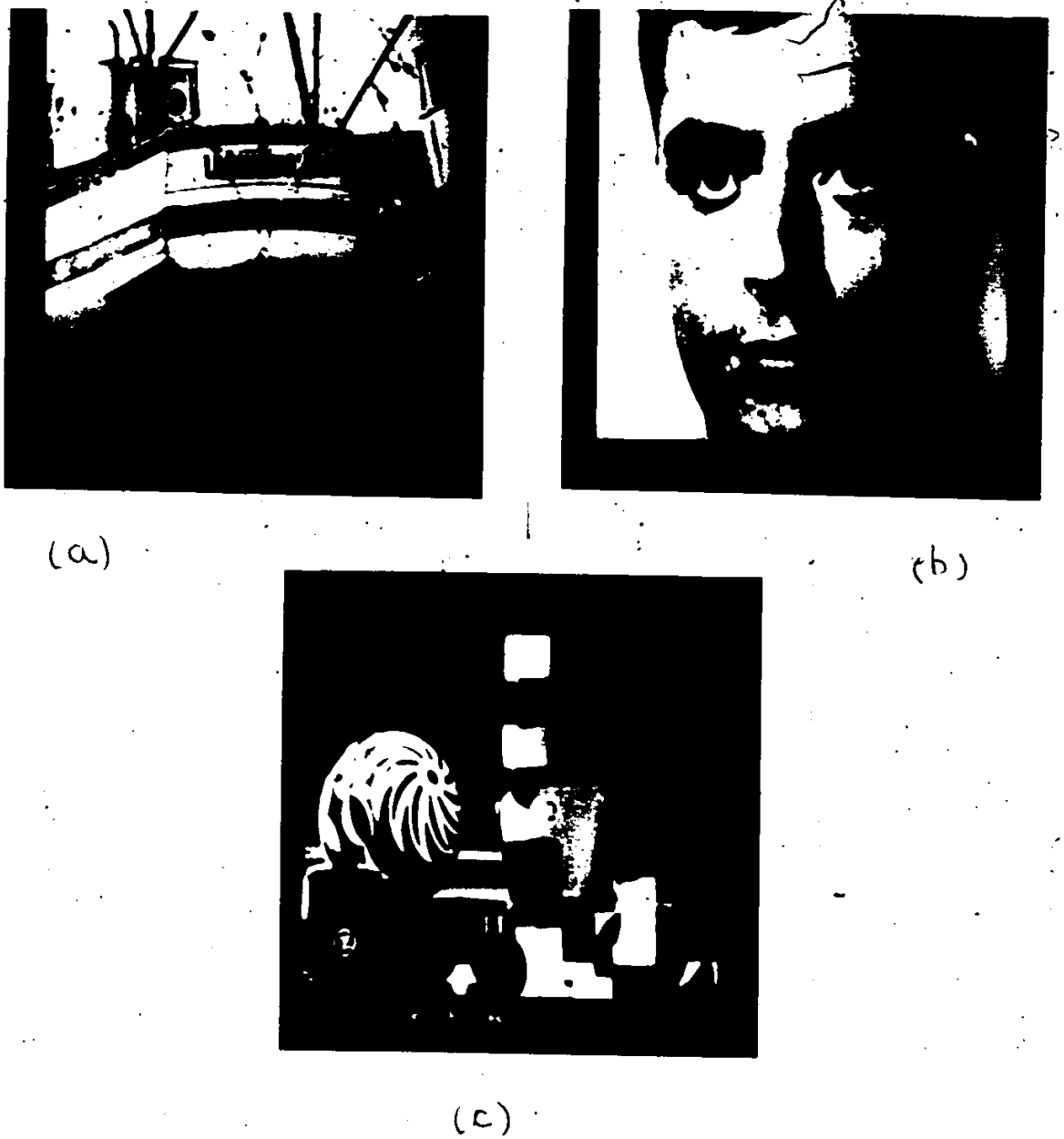


Figure 3.1-1 Monochrome test pictures, composed of 256 by 256 pixels PCM coded at 8 bits per pixel. a) BAT , b) FRA , c) TAB.

3.2 VECTOR SPACE PARTITIONING USING THE K-MEANS ALGORITHM

The K-means clustering algorithm is an iterative process minimizing a distance measure between a sample vector and a cluster center. It was chosen for quantizer design because it does not require any knowledge of the probability density function of the data and since it minimizes a criterion that can be related to quantization error.

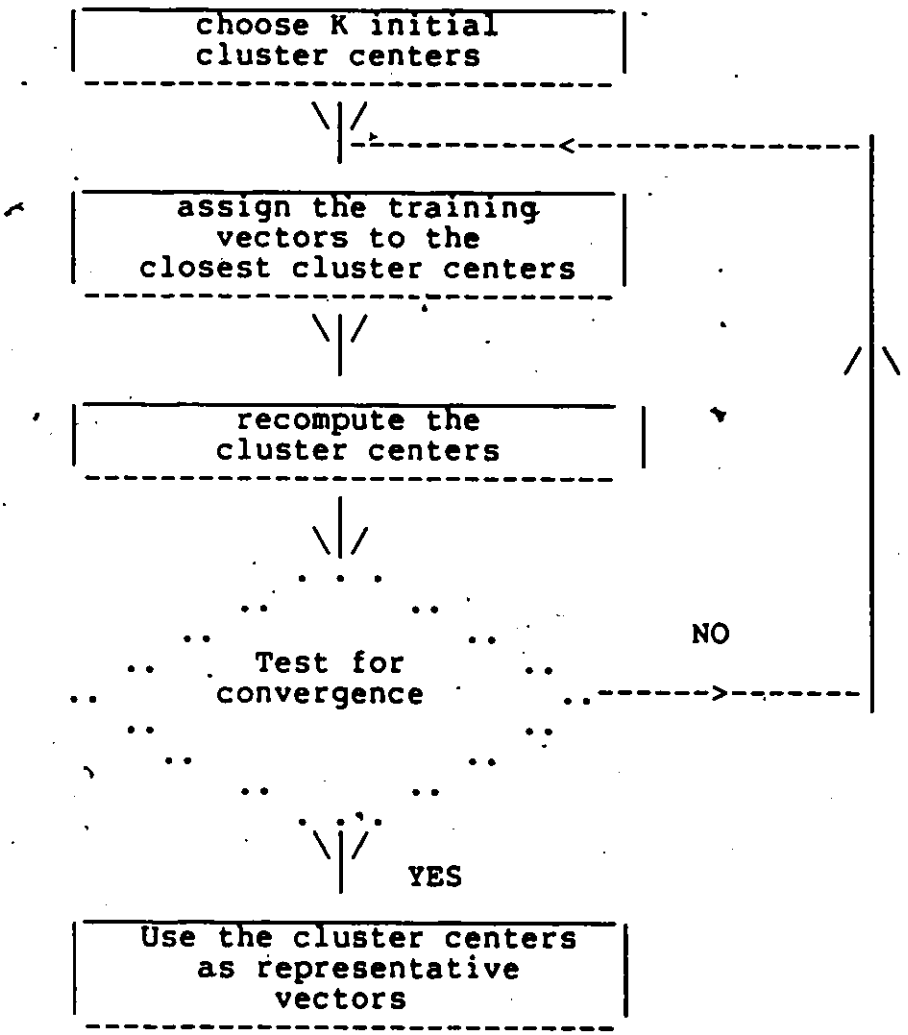


Figure 3.2-1 The K-means iterative clustering algorithm.

There exists no general proof of convergence of the K-means algorithm [29]; hence no optimal vector space partitioning is assured, only a local optimum. The achievable local optimum depends on the initial choice of the cluster centers [25]. Furthermore the speed of convergence depends on the initialization. For the same reasons, no optimal initialization technique exists.

In the following subsections, two initialization techniques, two distance measures and the convergence rate are investigated. Heavy emphasis is put on simplicity for the choice of the clustering parameters for a practical implementation. The simulations were performed using three training sets, each generated from the pictures of figure 3.1-1. The pictures are decomposed into vectors, using the grey level values from two by two pixel blocks as vector components. All the vectors extracted from an image represent a training set.

The quantization error is measured, using the normalized mean-square-error criterion (1.2.2.2.-3), between the original grey level values and the cluster center components coded linearly to eight bits.

If at some point during the iterative clustering process, a cluster becomes empty, the algorithm creates two new cluster centers. These are : u + e and u - e, where u is the mean vector of the most populous cluster and e a controlable

amount of noise added to each mean component. "e" is usually ten percent of the dynamic range of the coefficients (3 for 256 resolution coefficients). One of the new cluster centers replaces the cluster center value of the most populous cluster, while the other is used for the empty cluster. Although this technique is far from being optimal, it is simple and found to work satisfactorily. Furthermore it was observed that empty clusters occur rarely when the clustering algorithm is properly initialized.

3.2.1 Initialization of the K-means Algorithm

There are several ways to choose the initial cluster centers. One common method, is to choose the first K vectors in the training sequence [25]. This approach is dismissed, since one would like the initial cluster centers to be well-seperated, and K consecutive samples may not be.

Another scheme is to generate a uniform quantizer : i.e. a D-dimensional uniform quantizer is used on the D-dimensional Euclidian hyperspace, including all or most of the points in the training sequence [29]. Again this technique is dismissed because most of the sample points lie along a space diagonal, because the correlation is high between vector components generated from neighboring pixels.

A good initialization technique, is the "splitting" method [29]. Here Q level quantizers are generated, with $Q=2^q$

, q=1,2,.. , until a K level quantizer is obtained. Two initial cluster centers, Z(1) and Z(2), are first determined using the mean vector of the sample distribution, plus or minus one standard deviations. The K-means process is then used to obtain 2 representative vectors. Each representative vector is then split in two cluster centers, according to Z(i) - e , and Z(i) + e , where e is a fixed perturbation vector. These 4 vectors are used for initialization and the process is reiterated to generate 4 representative vectors. The process is stopped once K cluster centers are determined. This scheme is time consuming; by limiting the number of iterations at two for each step, the computing time would still be equivalent to approximetaly two iterations of the K-means algorithm for K levels.

Two fast initialization techniques are prefered : the traditional parametric method, or diagonal seeding, [26], and the mode-seeking approach [38].

3.2.1.1 Parametric initialization

The parametric initialization approach generally gives acceptable results on condition that the data is correlated in all its dimensions. This is the case for monochrome pictures, where vectors components correspond to grey level values from a contiguous region. K vectors are generated as follows

1)calculate the means "u" and standard deviations "σ" for each vector dimension.

2)Generate the intervals

$$INT(d) = [\, u(d) - c^* \sigma(d) \,,\, u(d) + c^* \sigma(d) \,] \quad (3.2.1.1\text{-}1)$$

for d = 1 to D

where "c" is a multiplicative factor, and d the vector dimension.

3)divide the D intervals into K sub-intervals (K is the desired number of clusters).Let a sub-interval length be

$$SUBINTV(d) = \frac{2 * c * \sigma(d)}{K} \quad (3.2.1.1\text{-}2)$$

for d = 1 to D

4) calculate the seed values for each sub-interval :

$$Z(d,k) = (u(d) - c^* \sigma(d)) + (k-0.5)*SUBINTV(d) \quad (3.2.1.1\text{-}3)$$

for d = 1 to D and k = 1 to K

5) regroup the seeds Z(d,k), d=1,..,D to form K vectors, and use these as initial cluster centers.

The vectors generated by this scheme are points, in vector space, lying along the diagonal of positive correlation through the hyper-rectangle, enclosed in the interval [u(d) - c* σ(d), u(d) + c* σ(d)].

The choice of the multiplicative factor "c" depends on the sample distribution.

3.2.1.2 Mode-seeking Initialization

A better alternative to initialization is to use the modes of the sample histogram space as initial cluster centers. MacCalla et al. [38] quotes that by properly choosing the modes, and assigning the sample vectors to the closest mode, better clustering results are obtained compared to K-means clustering initialized by the parametric scheme. The Koontz et al. [28] algorithm for mode-seeking is iteratively applied on small N by N MSS subimages. N by N vector space distance measures, comparisons, and gradient operations are required. The process converges, on the average, after three iterations [38]. The number of operations is thus $3*3*(N*N)^2$. The K-means clustering performs K*N*N operations for each iteration; thus the computing time of MacCalla et al. algorithm, for M and N and K equal to 16, corresponds to 144 iterations of K-means clustering.

The technique is still viable by using Narendra's et al. [43] mode-seeking algorithm. The scheme consists of two

steps. A histogram of the sample data is generated and used as a probability-density estimate. The modes of the histogram are determined using a graph-theoretic clustering approach based on Koontz et al. algorithm [28].

A histogram consists of a list of distinct vectors and their frequency of occurence. The histogram is computed by partitioning linearly the D-dimensional data space into discrete cells with equal volume and counting the number of vectors which occur in each cell. For pictures originally PCM coded with 256 grey levels, a good probability-density estimate is obtained by using only 64 grey levels; thus a histogram cell will contain up to 4^D full resolution distinct vectors.

The histogram vectors are then linked to parents. A parent of a vector is defined as the neighbor with the highest gradient. The gradient is measured as follows.

$$G(V(i), V(j)) = \frac{F(V(i)) - F(V(j))}{CBD(V(i), V(j))} \qquad (3.2.1.2\text{-}1)$$

F(V(i)) is the frequency count of a histogram vector V, CBD(V(i),V(j)) is the city block distance between a vector and a neighbor [43]. If no parent exists, the vector is called a mode. The clusters are formed by grouping all the vectors linked to a same mode, through parents, grand-pa-

rents, etc. The cluster space is thus partitioned according to the valleys between the modes. Only neighbors at less than a Euclidian distance of two are used : i.e. city block distances of one to four.

The population (using the frequency counts) of each cluster is computed, and the clusters with the greatest population are used as initial cluster centers for the K-means algorithm. If less then K + K/4 modes (since many isolated points are modes) are determined, the process is reiterated using a smaller neighborhood, until at least K modes are determined.

Fast algorithms exist for the histogram and neighborhood list generation [44]. The computing time (seconds), is linearly proportional to the number of histogram vectors. The following flow-chart summarizes the process.

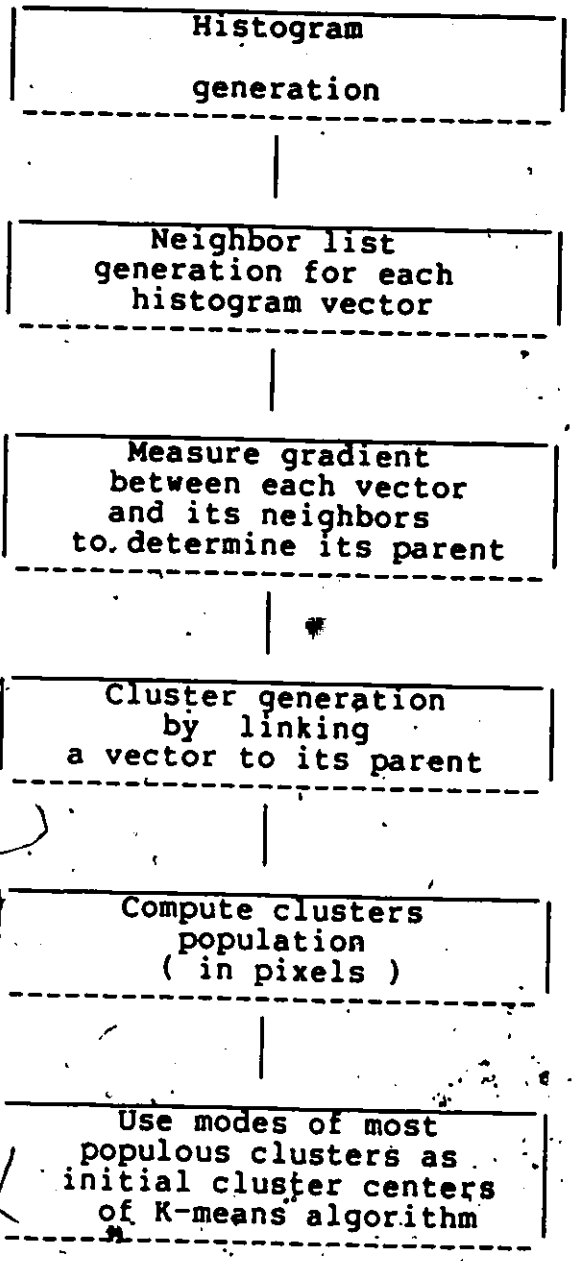


Figure 3.2.1.2-1. Mode-seeking initialization algorithm.

3.2.1.3 Comparison of the Initialization Techniques

Parametric initialization and mode-seeking initialization are compared using the three training sets described in section 3.2. The K-means algorithm is run on each of the training sets, using the Euclidian metric and cluster means as cluster centers.

Clustering is allowed to iterate until convergence. The normalized mean-square quantization error is measured after each iteration, between the new cluster centers and the original data. Table 3.2.1.3-1 presents the results obtained using the "FRA" training set picture ; similar results are obtained for the other training sets.

Number of clusters	Number of iterations to reach convergence			NMSE		
(K)	Mode-seeking	Parametric c = 1	c = 2	Mode-seeking	Parametric c = 1	c = 2
2	11	6	9	34.63	34.63	34.63
4	22	10	19	7.67	7.67	7.67
8	30	*	49	2.65	*	2.66
16	40	*	*	1.49	*	*

Table 3.2.1.3-1 Number of iterations to convergence for parametric and mode-seeking initialization. "*" indicates a very slow convergence, requiring more than 50 iterations.

Although the mode-seeking initialization technique produces slower convergence for $K < 8$, the end result is the same as the best parametric initialization scheme. For op-

timal results, "c" should be set equal to one standard deviation for K < 8 , and greater or equal to two for K > or = to 8. Mode-seeking initialization consistently outperforms parametric initialization for K > 4, for all training sets.

The computation time for mode-seeking, is negligible (seconds) on the VAX-750. The mode-seeking technique for fast computability, requires the use of very large memory space for storing the neighborhood list for each histogram vector : for D-dimensional vectors, 3^D * NT table entries NT being the number of histogram vectors [43]. This problem can be overcome on small computers, using fast disk handlers.

3.2.2 Metric Impact on the K-means Algorithm

The quantization error is represented by the distance measure used in the K-means clustering algorithm. Many different distance measures exist. General measures using the sample covariance of each cluster in addition to the cluster center are not used because of their high computing time requirement and since no satisfactory measure well related to the visual process exists [45].

Only simple metrics are considered for vector quantization. The absolute (Minkowski) metric produces cubical clusters, and the squared Euclidian metric produces spherical clusters. The absolute distance is formulated as follows

$$AD = |V_i - R_k| = \sum_{j=1}^{D} |v_j - r_j| \qquad (3.2.2\text{-}1)$$

The squared Euclidian distance is formulated as follows

$$SED = ||V_i - R_k||^2 = \sum_{J=1}^{D} (v_j - r_j)^t (v_j - r_j) \qquad (3.2.2\text{-}2)$$

The use of the squared Euclidian distance produces the same clustering result as the Euclidian distance since the relative distance value is used for decision. Henceforth, the squared Euclidian distance measure will be designated Euclidian distance.

It is expected that the absolute distance will produce poorer results than the Euclidian distance, according to the mean-square-error criterion, since the Euclidian metric is optimal in a mean-square-error sense. Nonetheless, the mean-square-error criterion can give an indication of the sensitivity of the clustering algorithm performance to the choice of the distance measure. This is especially expected for a large number of clusters where the assignment function is more dependent on the distance measure.

Simulation results comparing the absolute distance to the square Euclidian distance using the "FRA" picture, for K = 2, 4, 8, 16, 32, 64, and mode-seeking initialization are presented.

In all cases (even for K = 8) the Euclidian outperforms the absolute distance. Although the absolute distance produces slower convergence, the results show very small degradation even for large values of K. This indicates that the clustering performance is quite insensitive to the distance measure used in the assignment process. The results of table 3.2.2-1 are typical of other training sets.

Number of clusters (K)	Number of iterations to reach convergence		%MSE	
	Absolute distance	Euclidian distance	Absolute distance	Euclidian distance
2	11	11	34.63	34.63
4	23	22	7.684	7.667
8	24	30	2.693	2.653
* 16	20	20	1.563	1.491
* 32	20	20	1.036	0.993
* 64	20	20	0.687	0.681

Table 3.2.2-1. Comparison of the Euclidian and absolute distance measures, for the "FRA" training set. In the cases preceded by an "*", the iterative process was stopped after 20 iterations and the quantization error measured. Initialization was performed using the mode-seeking approach, in all cases. The sample mean of each cluster was used as cluster center.

The computing time saving using the absolute distance is minimal; nonetheless, for a low cost coding implementation, the absolute distance measure is practical since it does not require any floating point operations.

3.2.3 Convergence Rate for Iterative Clustering

Each iteration of the K-means algorithm consists of an assignment of all training vectors to the nearest cluster center, followed by the recalculation of the cluster centers. Convergence of the clustering algorithm occurs when none of the vectors change assignment, since the cluster centers do not change from one iteration to the next.

The clustering computations are linearly related to the number of iterations. It is thus necessary to investigate the relationship between clustering performance and the number of iterations.

Intuitively one would anticipate that many of the last iterations, before convergence, are just reassigning a small number of vectors which are located nearly equidistant from more than one cluster center.

The final assignment of such vectors will likely have little impact on the clustering performance. Simulations were conducted to determine the impact on the mean-square-error performance due to constraining the number of clustering iterations.

The results of these simulations on the "FRA" training set are shown in figure 3.2.3-1 where the normalized mean-square-error is plotted as a function of the number of iteration allowed for various values of K. In all cases mode-seeking initialization is performed.

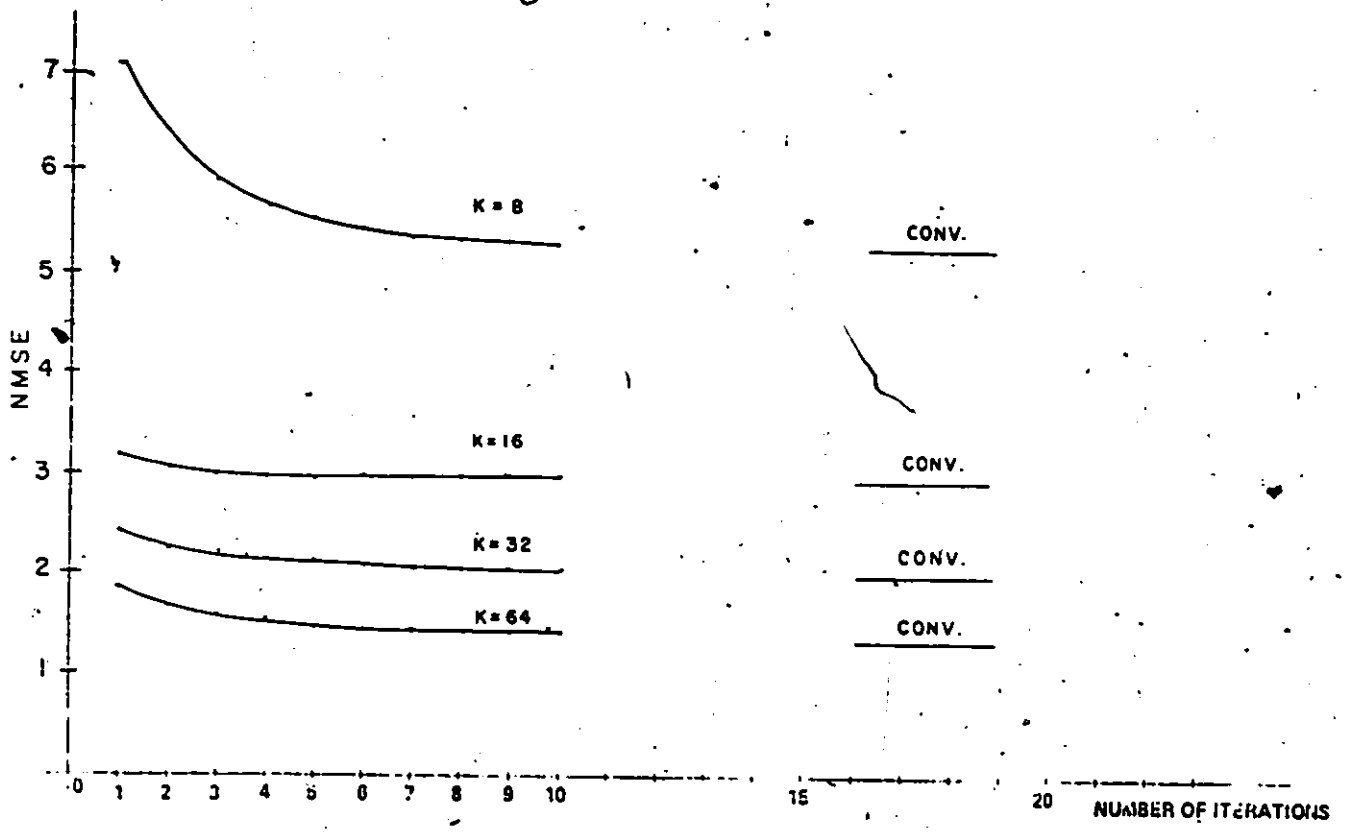


Figure 3.2.3-1 Normalized mean-square-error vs. number of iteration allowed for the "FRA" training set, for various values of "K". In all cases mode-seeking initialization was performed.

The results, typical of all training sets, indicate that with respect to the normalized mean-square-error performance, the number of clustering iterations can be limited to a maximum of about 5, with little performance loss. This is a significant result in terms of practical implementation consideration. A convergence measure is of secondary importance in a pratical implementation, since only a small number of iterations is required.

Convergence measures are still investigated and used in the following simulations, to make sure that in all cases a result close to convergence is obtained. A convergence measure is used to stop the recursive clustering process, instead of simply limiting the number of iterations. A simple technique is to measure the cluster centers drift from one iteration to the next. This convergence measure will be labelled MCCD : i.e maximum cluster center difference criterion. The vector difference between the previous and new cluster centers is measured. If any of the components of the K difference vectors is larger than MCCD, then the process continues.

Another scheme measures the quantization distortion [29], in our case the mean-square-error, between two successive iterations. This technique labelled QED, uses the following measure

$$QED = \frac{(MSE_{t-1} - MSE_{t})}{MSE_{t}} \qquad (3.2.3\text{-}1)$$

t-1 and t refer to the last and current iteration respectively. The quantization error is measured using the cluster center as representative vector. If QED is smaller than "e", the iterative process is stopped. For the three training sets described in section 3.2, and for any K, a QED smaller than 0.004 and a MCCD equal to 1.0 (one grey level), will stop the iterative process at around 10 iterations.

Although the QED and MCCD convergence criteria are sub-optimal, they are simple and produce satisfactory results. The MCCD is slightly erratic compared to the QED criterion, but has the advantage of requiring only K*D difference operations, while the QED criterion requires NT*D difference and squaring operations.

3.3 SUMMARY

Vector quantization of a training set generated from an image was shown to require many iterations to reach convergence. In cases involving large values of "K", the number of iterations is particularly large. In all cases only about 5 iterations is required to obtain a very satisfactory result. Vector quantization even in this simple form requires a large amount of computing time. Nonetheless two recommendations can be made, one for a low cost coder implementation, and another for a practical but near-optimum implementation.

A simple coder, should use the parametric initialization technique, setting "c" equal to one for $K < 8$, and to two for K equal or greater than 8. The absolute distance should be used, to eliminate floating point operations and the number of iterations limited to 5 or even less.

A more optimum coder, would use mode-seeking initialization, the Euclidian distance and a QED (quantization error

difference) of 0.004 or lower, as a measure of convergence. This last algorithm is used in the following chapter for coding the monochrome pictures of figure 3.1-1.

Chapter IV

MONOCHROME IMAGE CODING

In this chapter, computer simulations are used to provide comparative performance results for the various vector quantization strategies for coding monochrome imagery. The performance measures used to evaluate the various coding strategies are described in section one.

The coding philosophy, in this dissertation, is adaptive vector quantization : i.e. image quantizers are individually designed for each image source. The iterative K-means clustering algorithm, with the Euclidian distance, is used for the simulations. Unless otherwise stated, initialization of the algorithm is performed by the parametric scheme setting "c" equal to two, and the algorithm is allowed to iterate until the QED (3.2.3-1) convergence criterion is less than 0.0005.

The decoded image is obtained by substituting the closest cluster mean of the last iteration, according to the Euclidian metric, for each vector extracted from the image. The components of each cluster mean are linearly quantized to eight bits.

The first step in vector quantization is to decompose the image into a list of vectors. Block decomposition is investigated in section two. Results are presented for various compression ratios by varying the number of quantization levels : i.e. K.

In section three, vector list preprocessing is investigated, for reducing the large training time associated with K-Means clustering. In section four, local image source quantization is simulated, to further reduce training time and improve coding performance.

Another image decompositon technique is investigated in section five : the association of the S-transform with vector quantization. This scheme permits a progressive reconstruction of the decoded image. Section six discusses post-processing for data rate reduction. Finally section seven presents a performance comparison of the best vector quantization schemes to other non-information preserving coding techniques.

4.1 CODING PERFORMANCE MEASURES

The performance of a non-information preserving compression algorithm is measured in terms of the data rate vs. data quality and computing time.

The various quantization strategies are simulated using the test pictures of figure 3.1-1.

Data quality is measured in terms of percent mean-square-error and subjective appearance. The Percent Mean-Square-Error between the original and reconstructed image is calculated using 1.2.2.2-1 through 1.2.2.2-3 .

Photographic results are presented for subjective evaluation, but one must keep in mind that they are taken from a television monitor, and that the reproduction process introduces some artifacts.

The simulation software package was not optimized for speed but for flexibility. For this reason, relative computing time units are used, to present computing time performance from one scheme to another.

A computing time unit, for the VAX-750, depends upon : the number of available memory pages, input/output processing speed, type of array access, type of software languange used; etc. As a guideline, one computing time unit corresponds to one hour, for the present hardware-software simulation package.

4.1.1 Data Rate of a Vector-quantized Image

The total data rate of a compressed image is expressed in bits per pixel. In adaptive vector-quantized imagery both the codebook (2.1-6) and the codeword list (2.4-1), are included in the bit rate.

The contributing bit rate of the codeword list is

$$B_W^h = \frac{1}{h} \left(\left| \log_2 K^h \right| \right) \quad \text{bits per codeword} \qquad (4.1.1\text{-}1)$$

where $|\ x\ |$ = smallest integer >= x

This representation of the codeword list is wasteful if K, the number of quantization levels, is not a power of two (e.g., the use of 3 bits to specify one of 5 integers for K = 5). This representation can be improved by using simple fixed rate natural coding on the h-th extension (h-tuples) of the codeword list. For h > 1, the total bit rate is never significantly greater than for h = 1 [26]. For simplicity, the number of quantization regions will always be chosen to be power of two, and h set to one for optimal coding.

The codebook contribution to the data rate is calculated using

$$B_O = K * D * F \quad \text{bits} \qquad (4.1.1\text{-}2)$$

where each reconstruction vector component, r(i,k), is scalar number quantized using F bits.

The total bit rate for an adaptively vector-quantized image will be:

$$BR = \left(B_O + \left(B_W * \frac{M * N}{NP} \right) \right) * \frac{1}{M * N} \quad \text{bits per pixel} \qquad (4.1.1\text{-}3)$$

where NP refers to the number of pixels represented by a vector, and M * N the number of pixels in the image source.

4.1.2 Example of Bit Rate Calculation

Consider for example, adaptive vector quantization of a 512 by 512 pixel monochrome image. M =512 and N = 512. Assume that the image was originally coded using 8 bits per pixel (PCM coded).

Block decomposition is chosen : i.e. blocks of contiguous pixels are aligned into a vector. The block size will be four by four pixels. Thus : NP = 16 and D = 16.

Assume only 128 representative vectors are required to code this image, then K = 128. Each reconstruction vector dimension is quantized linearly to six bits, thus from (4.1.1-2) we obtain

$$B_O = 128 * 16 * 6 = 12288 \quad \text{bits}$$

This is the overhead bit rate contribution. ($\log_2 128$) bits are required for labeling each of the 128 quantization regions, hence using (4.1.1-1)

$$B_W^1 = 7 \text{ bits per codeword}$$

The total data rate for transmitting the above compressed image would be, using (4.1.1-3)

$$BR = \left(12288 + \left(7 * \frac{512 * 512}{16} \right) \right) * \frac{1}{512 * 512}$$

$$= 0.484 \text{ bits per pixel}$$

or a compression factor greater than 16, since the image was originaly coded using 8 bits per pixel. The corresponding refresh memory at the decoder would also be 16 times smaller than necessary, if real-time decoding is opted for.

4.2 IMAGE DECOMPOSITION INTO VECTORS

The first step in vector quantization of imagery is to decompose the image into a list of vectors. In section 2.2, two decomposition techniques were explained. Block decomposition, the simplest, partitions the image into blocks of pixels, each block representing a vector and each pixel grey level value corresponding to a vector component. The image is decoded block by block, producing a line by line type of reconstruction.

The computing time associated with K-Means iterative clustering is formulated as follows

$$\text{COMPUTING TIME} = \$ * NT * D * K \qquad (4.2\text{-}1)$$

where $ is a proportionality constant, NT is the number of training samples (vectors), D the vector dimensionality, and K the number of quantization decision regions. In the following simulations, the complete vector list extracted from an image, is used as a training set for the K-means clustering algorithm ; thus NV (2.2-1), the number of vectors generated by the image decomposition process, equals NT the number of training samples. Since the complete vector list is used as a training set, the computing time of 4.2-1, is for quantizer design, quantization and coding. The K-means process calculates a cluster center at each iteration, and associates implicitly each training vector to a cluster center. At the end of the clustering process, all vectors are

labelled (coding) by an integer number specifying to which cluster or representative vector it is associated. The required memory space for computation, for this case, is thus proportional to the image size.

Since the complete vector list is used as a training set, NT * D corresponds to the number of image pixels, thus is a constant for any block configurations. Only K influences the computing time : thus K should be kept as small as possible.

Block structure and size are two parameters influencing the compression ratio. The optimal block configuration, should maximize the correlation between vector components so as to obtain the densest possible vector space for quantizer design. The block configuration should thus be related to the image auto-correlation function.

The density of the vector space data distribution varies inversely with block size, since the correlation between vector components diminishes as the block size increases. Intuitively it is thus expected that vectors of high dimensionality (D) will require more quantization regions to produce equivalent reproduction quality. Furthermore it is expected that an upper limit in block size, will make vector quantization impractical.

Image redundancy extends in both spatial dimensions. Two-dimensional blocks are the logical choice for decomposition, since they maximize the use of intra-image redundancy.

The following block configurations are simulated : 1 by 4, 2 by 2, 2 by 3, 3 by 3, 3 by 4, 4 by 4.

Computing time (4.2-1) and memory space requirements are proportional to M*N. For practical reasons the input image source size, to the K-means processor, will be limited to square subpictures of 128*128 pixels. In other words, the test pictures of figure 3.1-1, are partitioned into four subpictures and each is processed as an small image source of 128*128 pixels. Quantizers are designed independently for each subpicture, called local sources, for a fixed value of K.

Results are presented using the "FRA" picture (figure 3.1-1); equivalent results were obtained from the other test pictures.

The purpose of these simulations is to obtain a measure of data quality vs. data rate and computing time requirements, as a function of block configuration. The data rate, or compression ratio, is varied using different values of K. K is varied discretly as a power of two, from 8 to 256 so as to obtain good to excellent image reproduction, and a smooth curve is drawn between the measurements. K is chosen as a power of two for full use of binary registers.

An important observation, from the following results, is that the Percent Mean-Square-Error correlates well with subjective error. Hence the following interpretations can be made directly from figure 4.2-3 :

- 0.2 bit per pixel is gained by using two-dimensional blocks instead of uni-dimensional block, for D equal to four.
- block sizes larger than three by three pixels do not produce any gain.
- for data rates below 1.5 bits per pixel, a gain of 0.1 bit per pixel is obtained by using three by three blocks instead of two by two blocks.
- above data rates of 1.5 bits per pixel the performance of larger blocks start to degrade, relatively to small blocks.

It was also observed that uni-dimensional blocks produce a more annoying artifact than square blocks, especially at low data rates.

The codeword, the overhead and the total bit rate are shown in figure 4.2-1, 4.2-2 and 4.2-3 respectively.

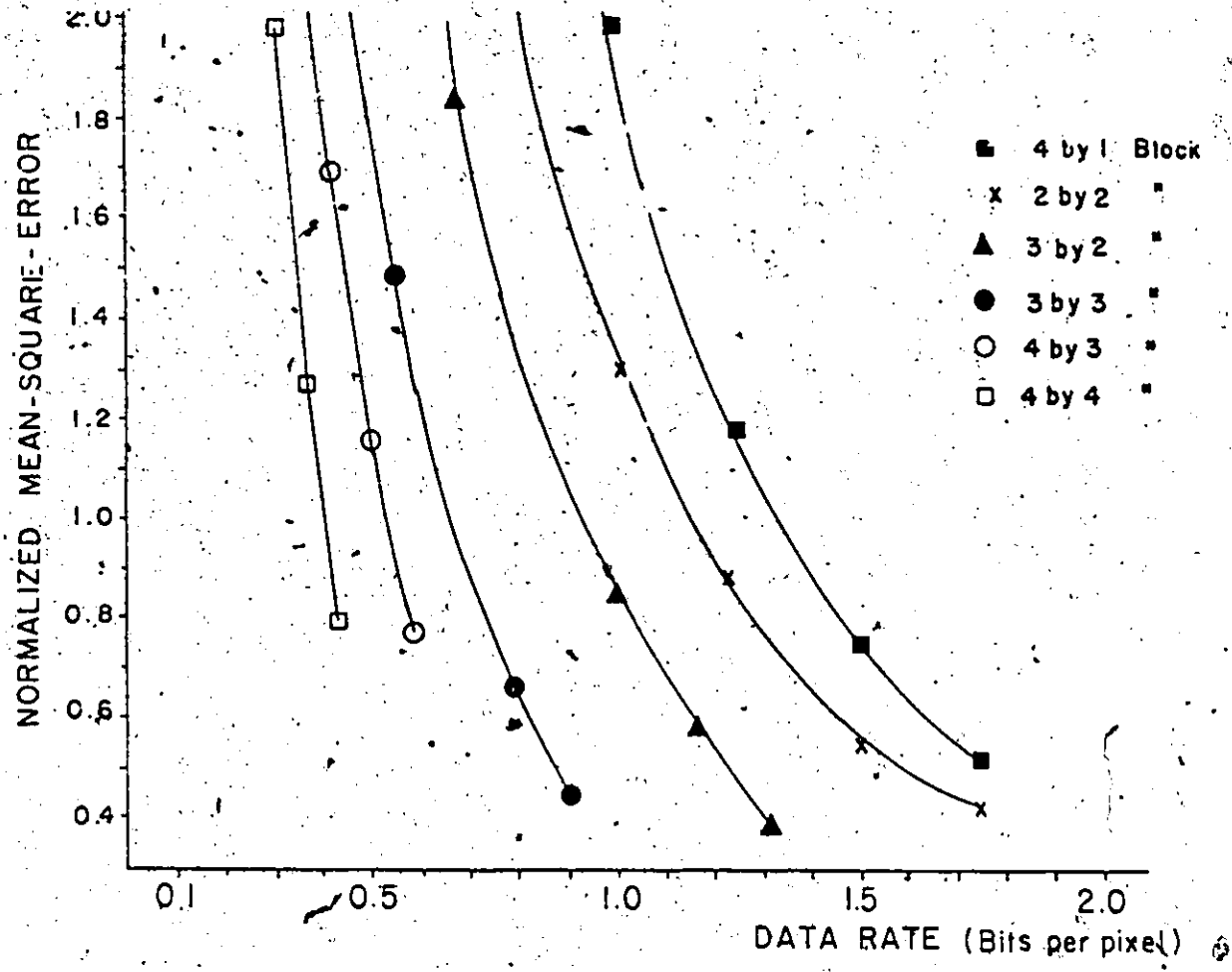


Figure 4.2-1 Percent mean-square-error vs. Data rate for various block configurations, for the "FRA" training set. Here the data rate only includes the codeword contribution.

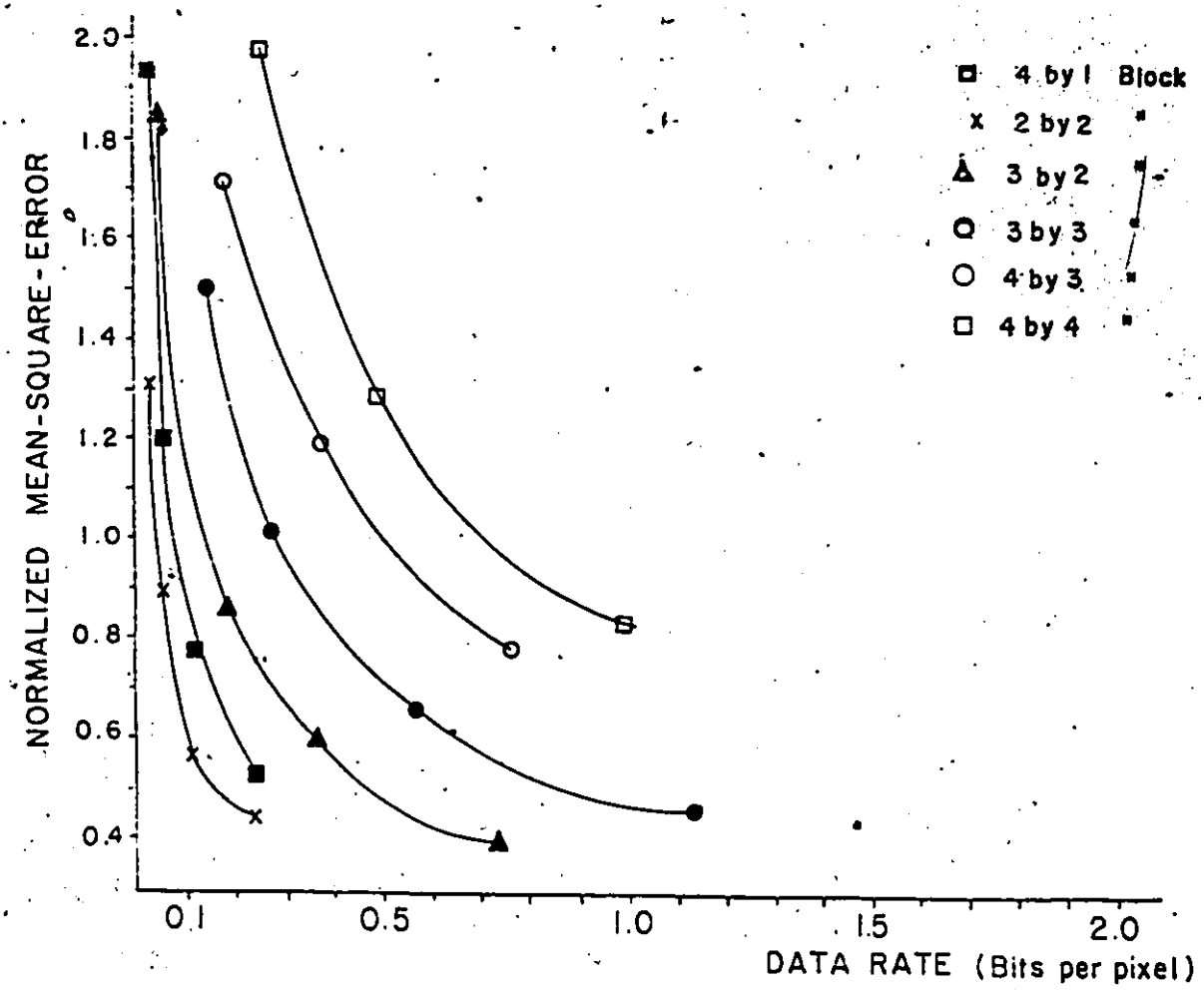


Figure 4.2-2 Percent mean-square-error vs. Data rate for various block configurations, for the "FRA" training set. Here the data rate only includes the codebook contribution.

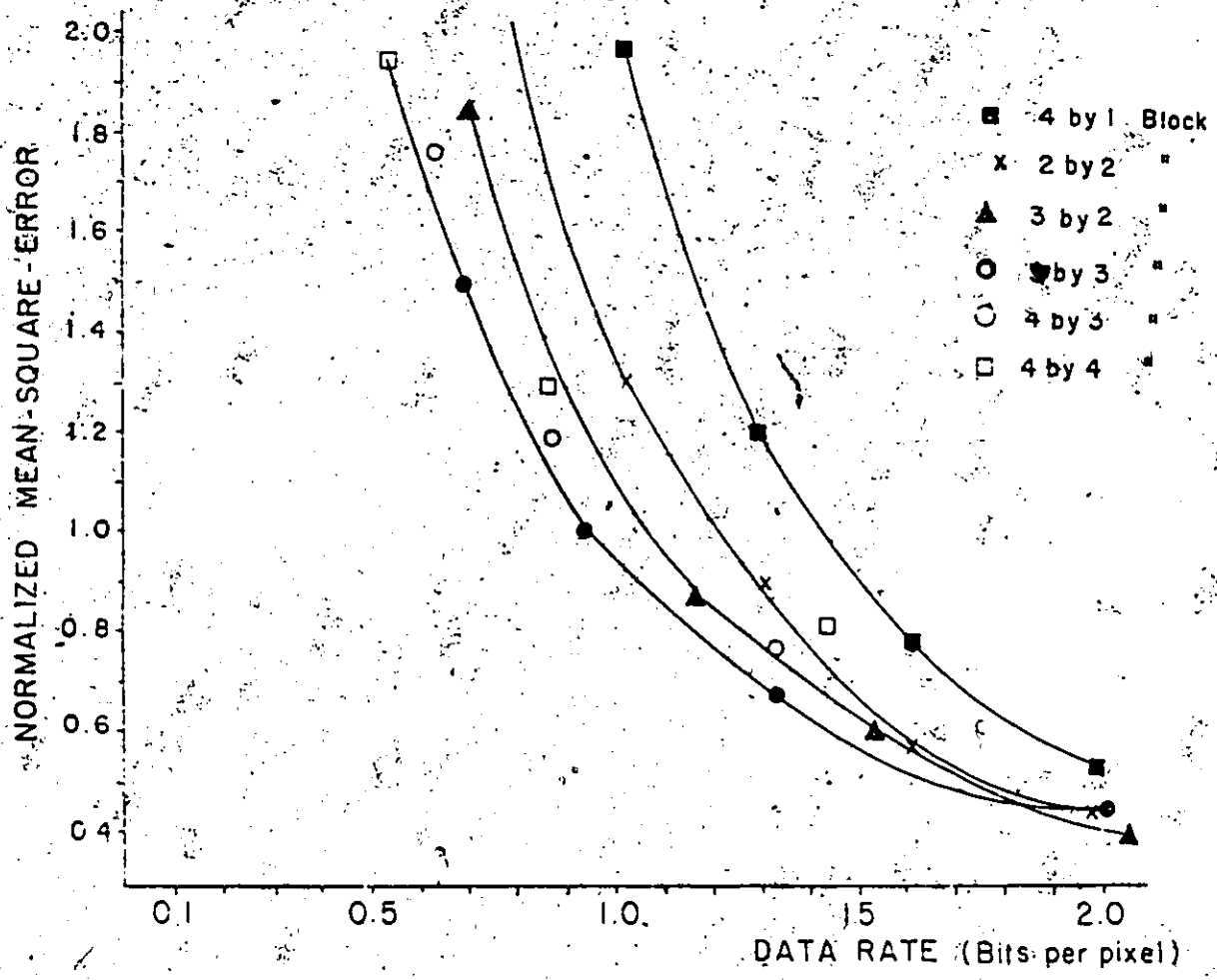


Figure 4.2-3 Percent mean-square-error vs. Total data rate for various block configurations, for the "FRA" training set. Here the data rate includes the overhead and the codeword contribution.

4.2.1 Photographic Interpretation

The following figures present photographic results, of various block configurations at various bit rates. It is observed that the main coding artifacts, produced by vector quantization are false-contouring, most visible in dark regions, and a slight staircase effect at edges (e.g. figure 4.2.1-2.a, the eyes). The staircase-effect is defined as diagonal edges being degraded to jagged ones, the contour of the staircase following the block structure. This type of artifact is not properly measured by the mean-square-error.

At low data rates, e.g. figure 4.2.1-1.c, block sizes of four by four pixels, produce a "blotchy" effect, while small blocks, e.g. figure 4.2.1-1.a, produce heavy false contouring.

Data rates of 1.3 bits per pixel (figure 4.2.1-2) produce very good picture approximations ; three by three pixel blocks being the optimal choice. At around 1.7 bits per pixel (figure 4.2.1-3), two by two pixel blocks give the best approximation, since a staircase-effect is still visible for larger blocks.

Some pictures decomposed using large blocks were coded using only six bits per representative vector component, to reduce the large overhead. This results in a slight contrast reduction.

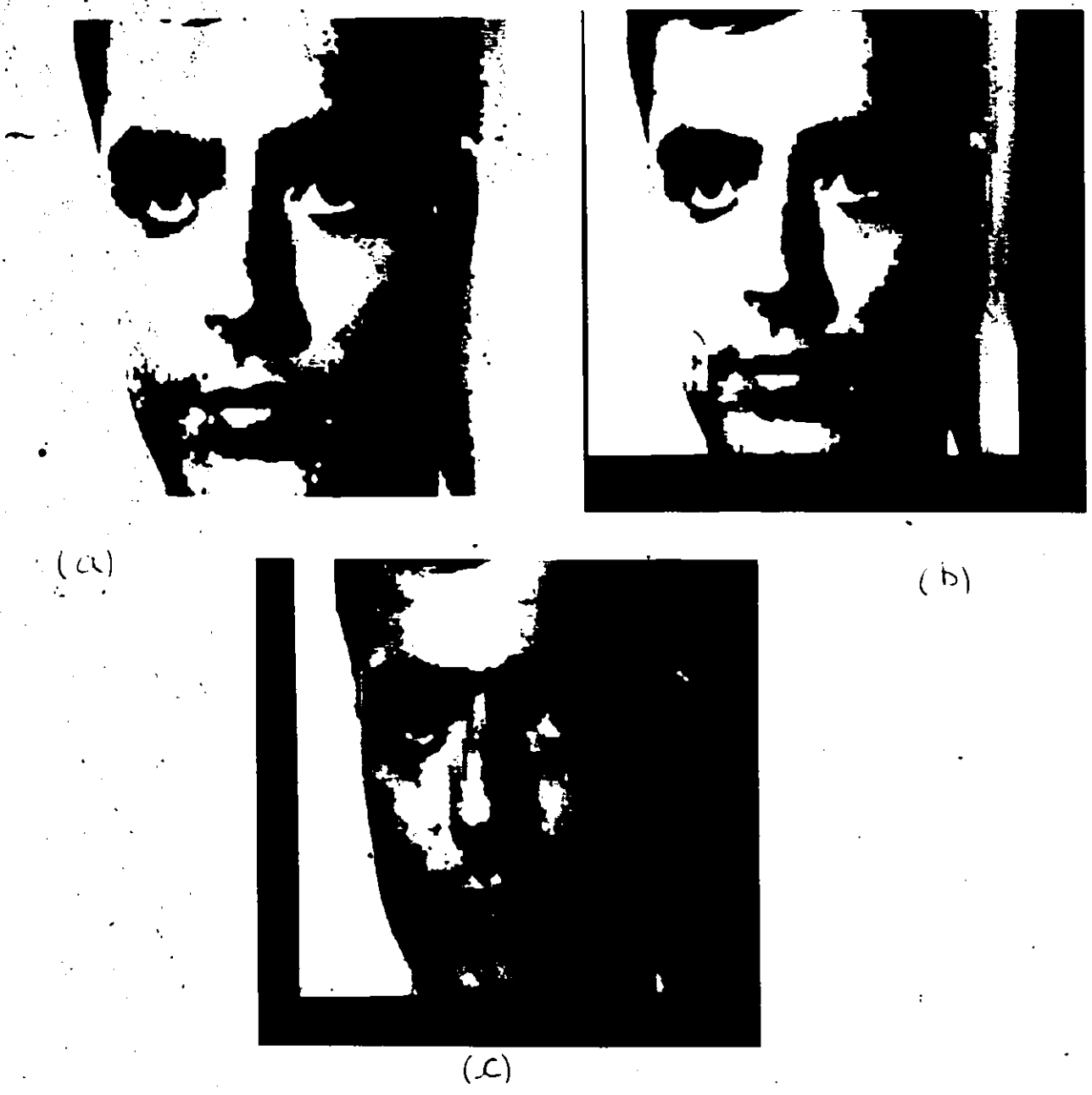
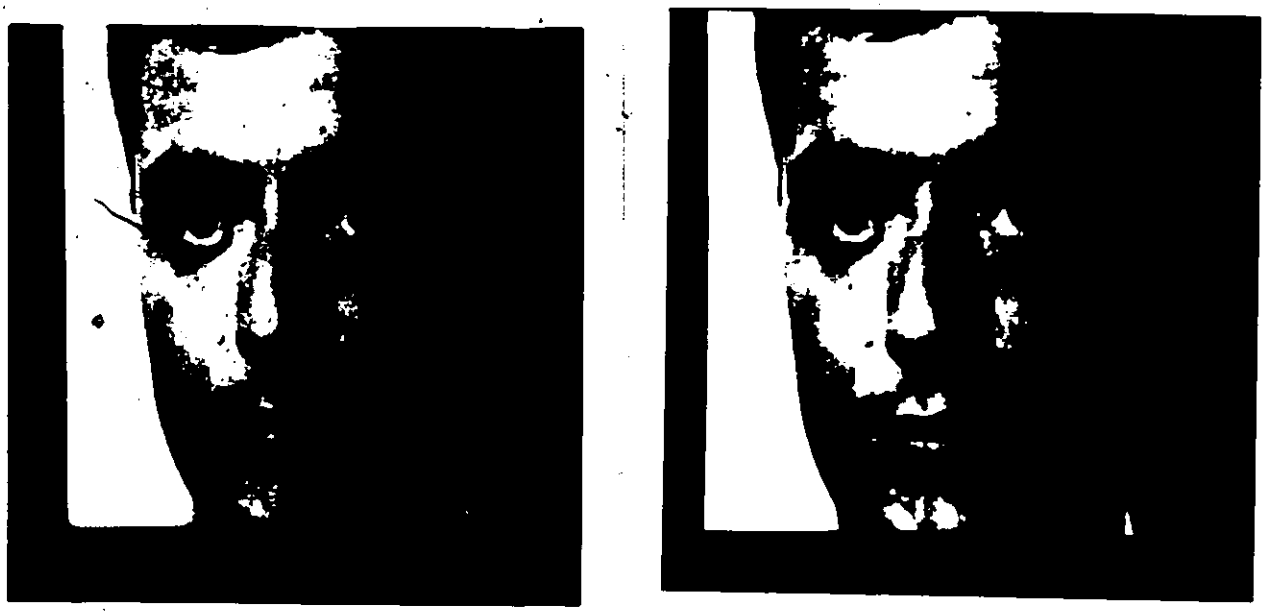
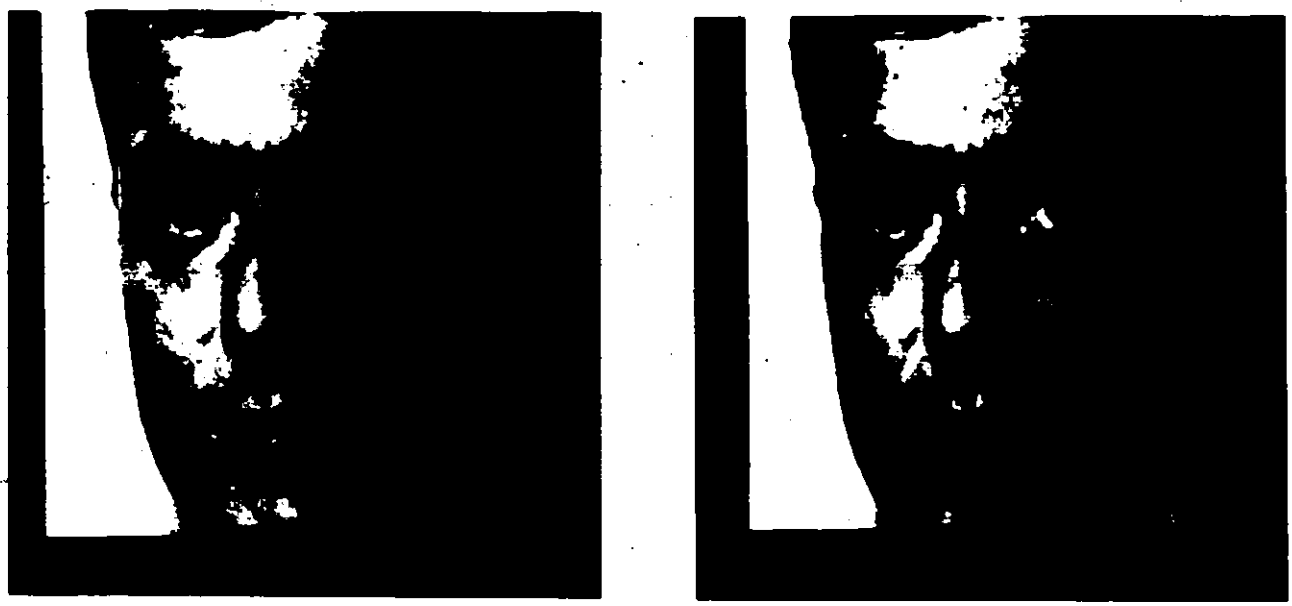


Figure 4.2.1-1 Example of block vector-quantizing the "FRA" picture using approximately 0.75 bit/pixel, with various block sizes. The picture is processed as four non-overlapping square 128*128 pixel sources, each coded using K reconstruction vectors. a) 2 by 2 blocks, K = 8 , 8 bits per vector component. NMSE : 2.37 . Total data rate : 0.77 bits per pixel b) 3 by 3 blocks, K = 32, 8 bits per vector component. NMSE : 1.53 . Total data rate : 0.70 bits/pixel. c) 4 by 4 blocks, K = 64, 6 bits per vector component. NMSE : 1.3 . Total data rate : 0.75 bits/pixel



(a) (b)

Figure 4.2.1-2 Example of block vector-quantizing the "FRA" picture using approximately 1.3 bits/pixel, with various block sizes. The picture is processed as four non-overlapping square 128*128 pixel sources, each coded using K reconstruction vectors. a) 2 by 2 blocks, K = 32, 8 bits/vector component. NMSE : 0.84 . Total data rate : 1.31 bits per pixel b) 3 by 3 blocks, K = 128, 6 bits/vector component. NMSE : 0.68 . Total data rate : 1.21 bits/pixel.



(a) (b)

Figure 4.2.1-3 Example of block vector-quantizing the "FRA" picture using approximately 1.7 bits/pixel, with various block sizes. The picture is processed as four non-overlapping square independent 128*128 pixel sources, each coded using K reconstruction vectors. a) 2 by 2 blocks, K = 64 , 8 bits/vector component. NMSE : 0.57 . Total data rate : 1.63 bits per pixel b) 3 by 3 blocks, K = 256, 6 bits/vector component. NMSE : 0.55 . Total data rate : 1.73 bits/pixel.

4.2.2 Computing Time for Quantizer Design

The computing time required to generate a quantizer is proportional to K (4.2-1). It was observed in figures 4.2-1 to 3 that large block sizes require a larger K for a constant quantization error ; thus large block sizes require greater computing time. Computing time requirements are presented in figure 4.2.2-1 for various block sizes and mean-square-error values.

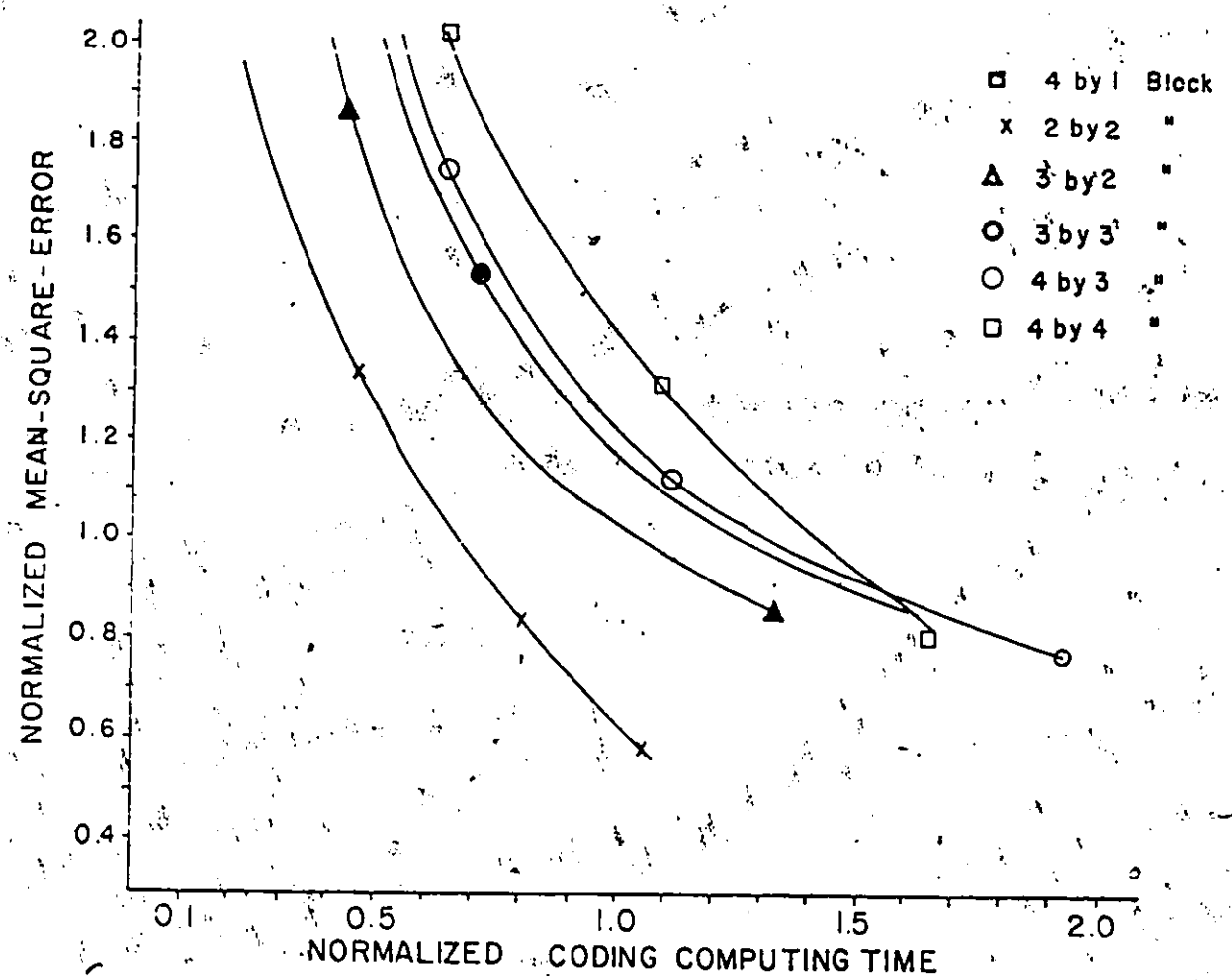


Figure 4.2.2-1 Percent mean-square-error vs. Training time for various block configurations. In each case the value of K was varied to obtain bit rates from 0.5 to 2.0 bits per pixel. The "FRA" training set was used.

Figure 4.2.2-1 shows the computer time variation for various error rates, not data rates, since picture quality is of greater importance. For a fixed error rate, the data rate varies as shown in figure 4.2-3.

The improvement brought by large blocks is costly in computing time ; for blocks size up to nine pixels, the next larger block size requires 0.2 computing time units more, for the same mean-square-error (but lower bit rate).

Simple schemes can reduce training time ; for example the average number of iterations to produce the pictures of figure 4.2.1-1 to 3 was 20 (for a QED (3.2.3-1) of 0.0005). By limiting the number of iterations to 10, each curve of figure 4.2.2-1 will move down approximately to the next smaller block configuration curve, with minimal image degradation. Other techniques to further reduce training time are discussed in sections 4.3 and 4.4.

4.2.3 Summary

The coding time as defined in (4.2-1) includes quantizer design, quantization and coding. Decoding complexity is independent of the block size. However, the coding time is relatively large and proportionnal to the block size.

If coding time is of no concern, the optimal block configuration is three by three pixel blocks. Otherwise the use of two by two pixel blocks reduces the computing time at

the expense of a loss in compression (up to 0.2 bits/pixel). For the same bit rate, two by two cells require a value of K less than half that of three by three cells (see figure 4.2.1-2), thus the computing time is reduced by the same amount, but at the expense of some subjective distortion.

4.3 TRAINING SET COMPRESSION

Considering the conceptual simplicity of vector quantization and the rather low data rates obtained in the last section, the performance achieved is quite remarkable. The only disadvantage is the high computing time necessary for quantizer training. It is possible to obtain a 50 percent computing time decrease by limiting the number of iterations; but even this is insufficient. A coding time of minutes is aimed for, to be able to compare vector quantization to transform techniques.

In this section, other schemes to obtain faster training time, with minimal image degradation are investigated: decorrelation of the vector luminosity components and redundant vector elimination by way of histogram techniques.

4.3.1 Intra-vector Decorrelation

The computing time of the K-Means iterative training algorithm, is proportional to NT * D * K (4.2-1). K sets the compression ratio, chosen as a function of D and the image source size (figure 4.2-3); hence only the factor, NT * D ,

is independent of the compression ratio and can be modified to obtain a faster training time.

Correlation exists at two levels in the vector list, between vector components and between the vectors themselves. Correlation in pixel blocks runs very high, because of the typical high redundancy existing in imagery [45]. Many decorrelation functions exist. A well known group is unitary transforms: Hadamard [14], Fourier [15], Cosine [16], etc. Unitary tranforms and clustering using an unweighted distance measure, are linear operations : hence training on tranformed data would not modify clustering results, provided that initialization is exactly the same for correlated and uncorrelated data.

The advantage of decorrelating the data, comes from the property of unitary transforms of compacting the energy of a vector in the first coefficients. The least important coefficients can be simply ignored for training, thus effectively reducing vector dimensionality and training time. For example, a 50 percent training time saving is obtained by using only half of the transform coefficients generated from 16-dimensional luminosity vectors. The number of coefficients to be used for training depends on the original vector dimensionality.

The Hadamard transform is chosen as a decorrelation function because of its computing simplicity, requiring only in-

teger add, subtract and shift operations. Unitary transforms order the transform coefficients by order of frequency, or sequency in the case of the Hadamard transform, corresponding approximately to the energy content of the coefficients. Thus the last coefficients are of small energy content, hence of little importance.

Clustering can further be improved by using a weighted distance measure : i.e. by weighting each transform coefficient according to its importance. A weighted distance measure requires D supplementary operations (divisions) for each vector, at each iteration. An alternate way is to weight each vector component, before clustering, by reducing the transform coefficient's dynamic range according to their importance. The high energy coefficients will conserve their full resolution, while the low energy coefficients a lower resolution. The use of an unweighted sum type distance measure, during K-means clustering, will automatically take into account each vector component's importance. For example, a distance of four in a sample space divided into 256 grey levels, corresponds to a distance of 1 for the same space at 64 resolution.

Although only a certain number of transform coefficients can be used for training, all of the coefficients are used for calculation of the representative vector, at the last training iteration.

Grey level representative vectors are regenerated using the inverse Hadamard transform. The Hadamard transform coefficients are only used for faster processing and not for tranmission.

Results, for a two by two block configuration, are presented in the next section, where only three out of four Hadamard coefficients are used for training. The resulting degradation is minimal. The results also include the following preprocessing scheme :redundant vector elimination by histogram techniques.

4.3.2 Redundant Vector Elimination

It is typical to observe large smooth areas (near constant grey level values) in natural imagery. Vector quantization tends to smooth out these areas and produce slight false contouring (figure 4.2.1-1 to 3). Since these vectors are smoothed out implicitly, they contribute very little to quantizer design. Thus a non-negligible amount of redundant vectors exists in natural imagery. Depending on the block size, more or less redundant vectors exist.

By way of histogram techniques, redundant vectors can be replaced by a frequency count. The number of training samples, NT, is effectively reduced without any loss of information and accordingly clustering time is also reduced. Histogram techniques can also quantify data. This is possi-

ble by first lowering the data resolution (reducing the number of grey levels), and classifying each vector in one of L^D histogram regions or bins, L corresponding to the number of grey levels. This corresponds to partitioning the vector space into L^D equivalent adjacent bins. All the vectors in a bin are replaced by a vector whose components correspond to the most significant bits of each original vector component. The number of vectors in each bin is represented by a frequency count. For example, if instead of using 256 grey levels per vector component, 64 are used, then 64^D classifying regions are produced, instead of the possible 256^D. 64 grey levels is sufficient to represent the vast majority of pictures [30] with minimal degradation. Correspondingly this pre-quantization scheme introduces negligible degradation. It is only at resolution below 64, that errors introduced by the histogram preprocessing begin to be too important.

The following graph shows the average number of different vectors, for various subpicture sizes, each vector component quantized to 64 grey levels.

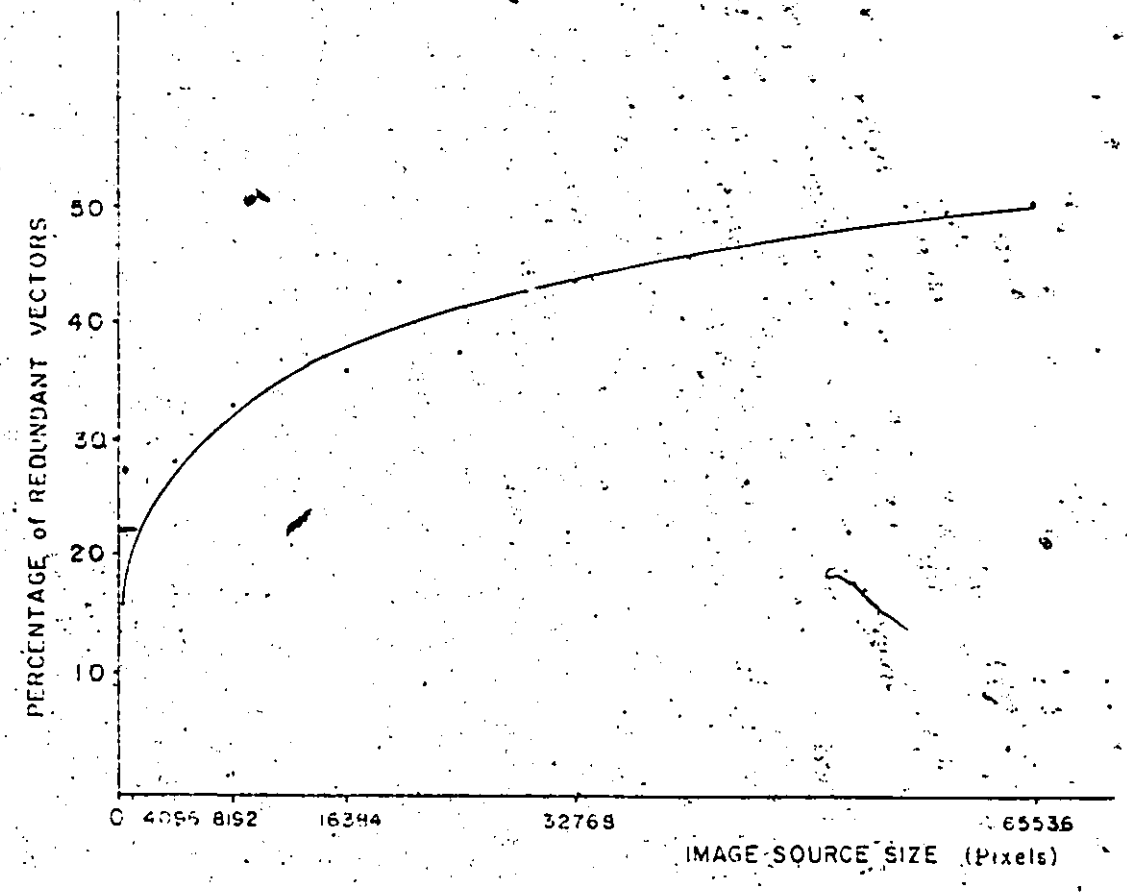


Figure 4.3.2-1 Average percentage of redundant vectors (64 grey level resolution) for different local source sizes.

It is observed that for an image of 256 by 256 pixels, an average reduction of 51 percent in number of training samples is obtained.

This preprocessing technique is useful only if the histogram generation time is small. A fast histogram generation algorithm is hashing. The algorithm proposed by Narendra et al. [43] is used for the simulation. The computing time is negligible.

If the preprocessing technique of section 4.3.1 is combined with histogram preprocessing, much higher savings are obtained. This comes from using only a fraction of the transform coefficients, resulting in even fewer independent histogram vectors. For example, if coding is performed on a 128 by 128 pixels local source with two by two pixel blocks, a 36 percent saving is obtained by histogram preprocessing. A further 13 percent saving is obtained if the histogram preprocessing is performed on only three out of four transform coefficients. A further implicit reduction of 14 percent results because training is performed on only three out of four coefficients, hence resulting in a grand total of a 63 percent computing time saving. This training time reduction is obtained with minimal image degradation, as shown in the following graph.

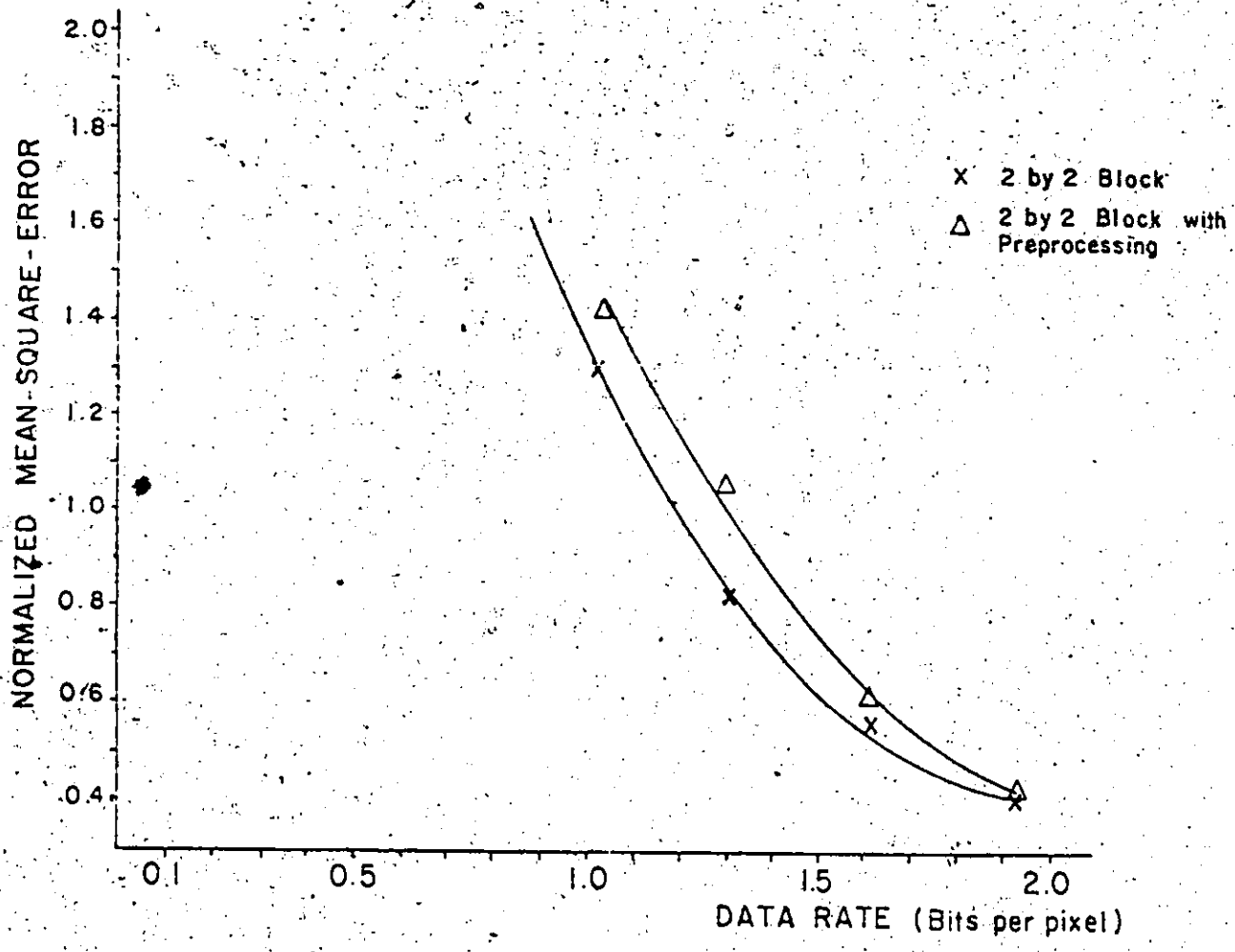


Figure 4.3.2-2 Degradation brought by histogram preprocessing and Decorrelation of the training samples, using only three out of four transform coefficients for training. The image used is 256 by 256 pixels quantized using two by two pixel blocks.

Image redundancy is present at two levels in the vector list, as discussed earlier. The redundancy existing in the vector list does not vary from one block size to another. What varies is the distribution of redundancy at each level;

i.e. higher dimensionality vectors will produce less redundant vectors but contain more intra-vector redundancy. Thus training time saving would not vary from one block size to another.

It has been shown that large training time reduction is obtained by way of redundancy removal : i.e. preprocessing, with minimal image degradation. The redundancy removed resulted in reducing the NT * D factor of equation 4.2-1. Ways to further decrease training time are presented in the next section, where the variable K of equation 4.2-1, is effectively reduced for a fixed data rate.

4.4 LOCAL SOURCE QUANTIZATION

It was previously observed that at least 64 representative vectors are required to obtain a good quality/compression ratio, for 256 by 256 pictures. The associated computing time is relatively large, especially for large blocks. Although preprocessing techniques can reduce the computing time by more than 60 percent, it is still impracticaly large. In this section the computing time is further reduced by modifying the variable K of 4.2-1. This is achieved by quantizing an image at the local level. In other words, the image is partitioned into small non-overlapping subpictures and each is processed as a small independent source. This scheme also reduces memory requirements, since only small image sources are processed at one time.

The rationale behind this scheme is that if K quantization regions are required to represent a complete image, then a smaller number of regions, K', is required to code a small subpicture extracted from the original image.

Again two approaches can be taken in coding each local source : adaptive or non-adaptive quantization. Non-adaptive quantization is still rejected for the reasons discussed in section 2.7.1.

If the image is subdivided into NS non-overlapping local sources and each source individually quantized using a fixed number of levels, K', then the total number of different re-

construction vectors representing the global image is NS * K'.

The total image training time, excluding initialization, is now equal to

$$\text{COMPUTING TIME} = \$ * NS * NT' * D * K' \qquad (4.4\text{-}1)$$

where $ is a constant of proportionality, and NT' the average number of training samples per local source. If all vectors generated by a decomposition process are used as training samples, then the factor NS * NT' is equal to NT, the total number of training samples used when globally vector-quantizing a picture.

If histogram preprocessing is performed, then NS * NT' will be slightly larger than NT, because local processing does not eliminate inter-subpicture redundancy. Local processing is thus expected to require a factor NS * K', larger than K, for a same data quality. Still the compression ratio is expected to be higher for a locally quantized image because the codeword length, $\log_2 K'$, will be smaller than $\log_2 K$, even if the overhead NS * K' is larger than K. In any case, K' is smaller than K, and the training time is reduced by the factor K/K' approximately. Other advantages of processing on such a small scale are discussed below.

Define PG(V) as a global sample distribution function in intensity space obtained by sampling pixel blocks over some spatially large source g, for example a picture of 1024 by 1024 pixels. PG(V) can be expressed as a mixture of local sample distribution functions PL(V) obtained from sampling pixel blocks within small spatially local sources (e.g., 16 by 16 pixels) giving

$$PG(V) = \frac{NVL}{NVG} \sum_{L_i \in g} PL_i(V) \qquad (4.4\text{-}1)$$

where NVL and NVG are the number of vectors in L and G respectively. Sources G and L are depicted in figure 4.4-1(a) and typical distributions for PG(V) and PL(V) are shown in figure 4.4-1(b).

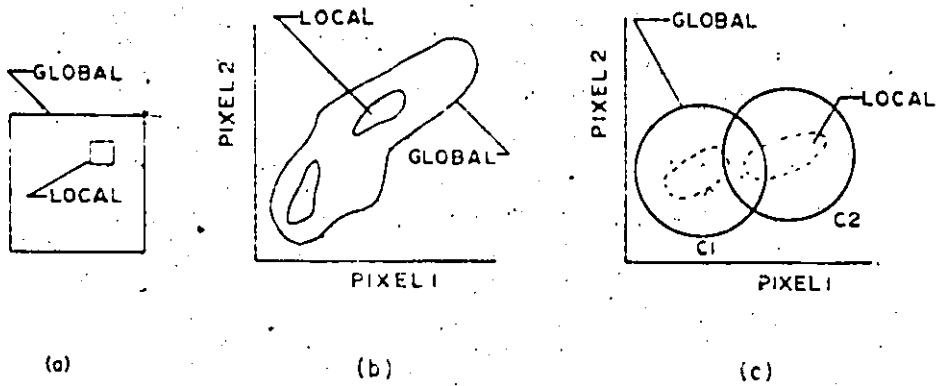


Figure 4.4-1 Global vs. Local sources. (a) Spatially (for two pixel blocks), (b) Distributions, (c) Conditional Distributions.

Observe that it is reasonable to expect a low average number of clusters of interest per local source relative to the number of clusters within the global source. Thus, clustering at the local source level is initially easier to implement because there are fewer clusters and fewer training samples to cluster simultaneously.

Consider some local source L' with clusters C1 and C2. Typical PG(V/C1), PG(V/C2), PL'(V/C1) and PL'(V/C2) are shown in figure 4.4-1(c). The larger spread in each global conditional distribution is due to averaging over many local distributions, each with corresponding various perturbations due to cluster variance. Therefore, at the local scale the training samples of a given cluster tend to form tighter groups which are more separable.

The greater separability of clusters within a local source would permit faster convergence, or a more practical advantage, permit the use of simpler clustering algorithms while still preserving separability. Thus clustering at a spatially local source level is not only important for practical implementation reasons, but it is also an advantage from the standpoint of clustering quality, although it should be clear that small local sources will not support a larger number of clusters for the same estimate of accuracy.

Figure 4.4-2 shows the average number of iterations required for convergence, for the "FRA" picture (figure

3.1-1.b), as a function of local source size in pixels and K'. As expected, the number of iterations increase with increasing source size and K'. Of course the number of iterations for convergence will fluctuate around the average for different data sets.

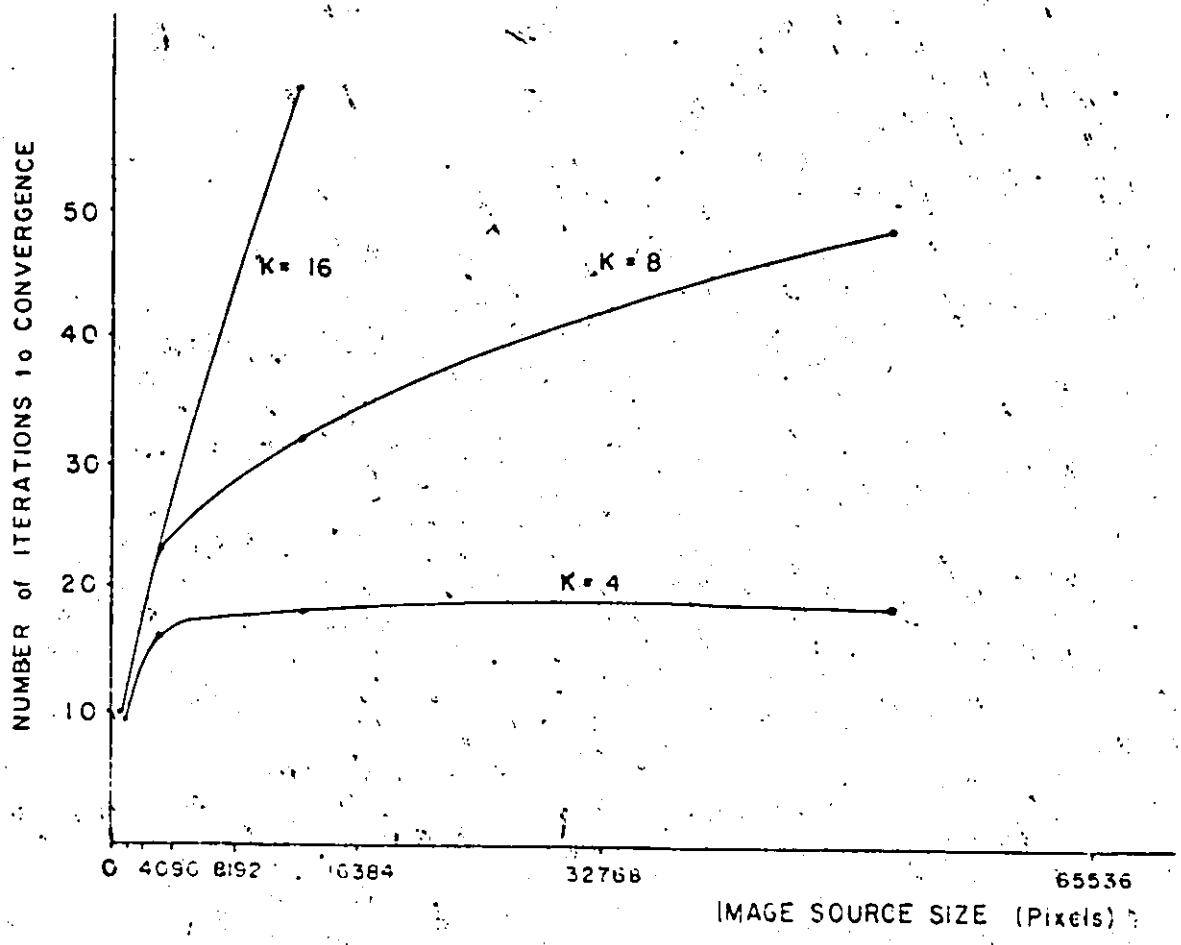


Figure 4.4-2 Number of iterations required to reach convergence for various square local source and various values of K'. The "FRA" picture was decomposed using two by two blocks. Parametric initialization was used in all cases with "c" equal to two (see section 3.2.1.1). The other test pictures produce equivalent results.

Although it was discovered in section 3.2.1.3 that mode-seeking initialization is best for local sources of 256 by 256 pixels, parametric initialization with "c" set to two standard deviations produces slightly faster convergence for local sources of 32 by 32 to 128*128 pixels, decomposed in two by two blocks. The superiority of parametric over mode-seeking initialization is attributed to the poor probability-density estimate produced with small training sets. A smoothing of the histogram would be necessary to produce a better estimate at the expense of more computing time ; thus the simpler parametric initialization scheme is used for local vector quatization.

4.4.1 Optimal Local Source Size

The best subpicture structure is next analysed for two by two blocks, in terms of data quality. The local source size was varied from 256 pixels to 65536 pixels. It was observed that a square or rectangular subpicture structure is unimportant for local sources larger than or equal to 128*128 pixels. The optimal structure for smaller subpictures, is a square configuration. The results are presented using three categories of local source sizes:

1) large : greater or equal to 128*128 pixels

2) medium : less than 8192 and greater than 512 pixels

3) small : less or equal to 512 pixels

The same quantization algorithm , K-means clustering, was used in all cases, with the Euclidian distance measure and parametric initialization, with "c" set to two standard deviations.

Local quantization is expected to yield better results subjectively since a different quantizer is implicitly designed for each local source.

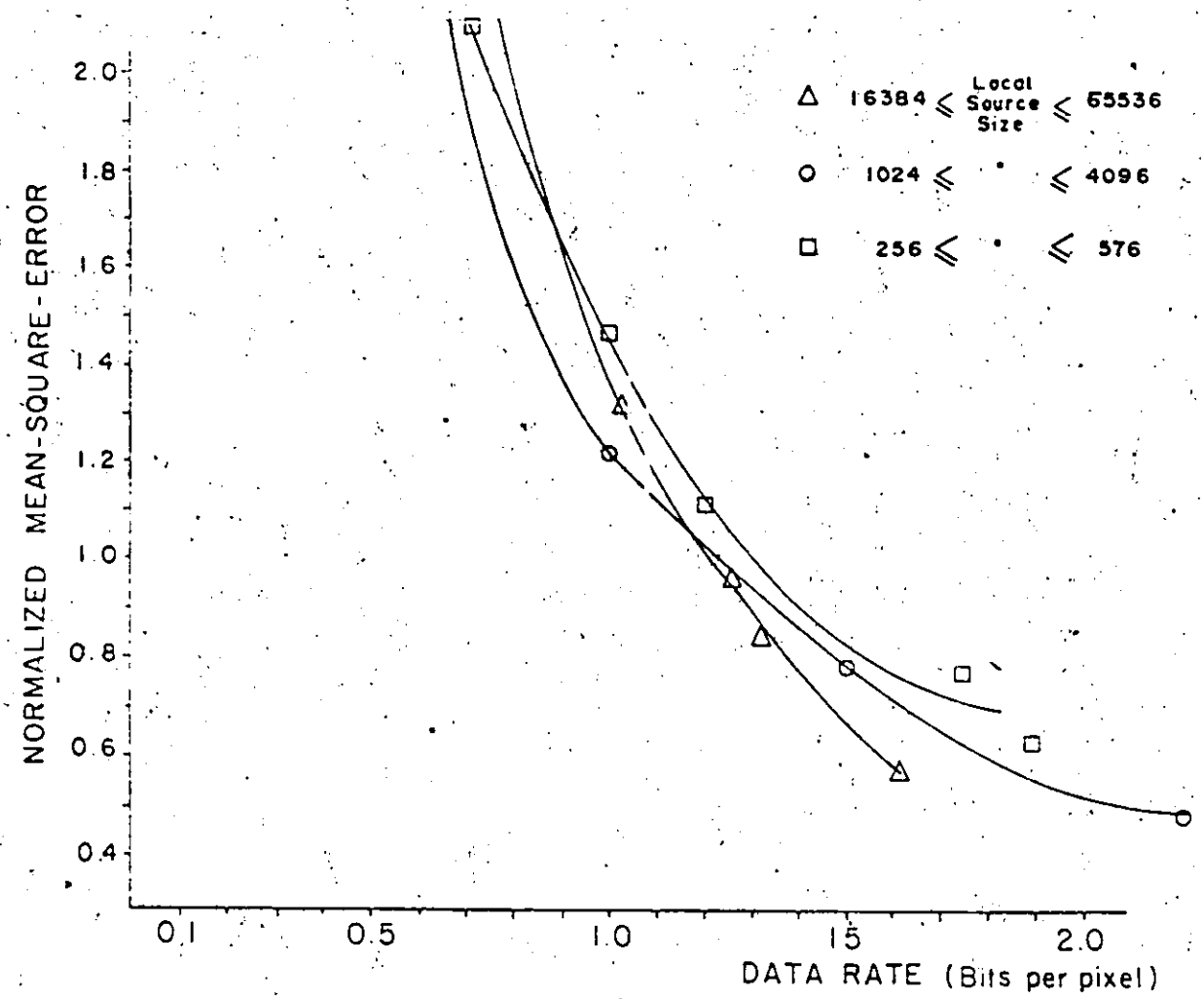


Figure 4.4.1-1 Typical normalized mean-square-error vs. data rate for various local source sizes and various K'. The overhead or codebook, is included in the data rate. Two by two blocks were used for decomposition.

If an image is divided into NS equal-sized subpictures and each is quantized using the same value of K', then BR (4.1.1-3) calculated for one local source corresponds to the BR of the entire image.

As can be observed from figure 4.1.1-1, large local sources are optimal for coding at data rates above 1.4 bits per pixel, while medium local sources are best adapted for data rates below 1.4 bits per pixel. Small local sources are inefficient for any data rates, because the overhead is relatively too important. Furthermore, at medium data rates a slight streaking effect, not measurable by the mean-square-error criterion, is visible.

The saving from one local source category to another is not very significant, 0.1 bits per pixel. The main advantage of using small local sources comes from the faster convergence and the small K'/K factor which greatly diminishes the total training time; small local sources require four times less computation time than large local sources ; as can be observed in figure 4.4.1-2.

By combining the preprocessing schemes explained in the section 4.3 with local source coding, only minutes of computing time are required to code a 256 by 256 pixel image using local vector quantization.

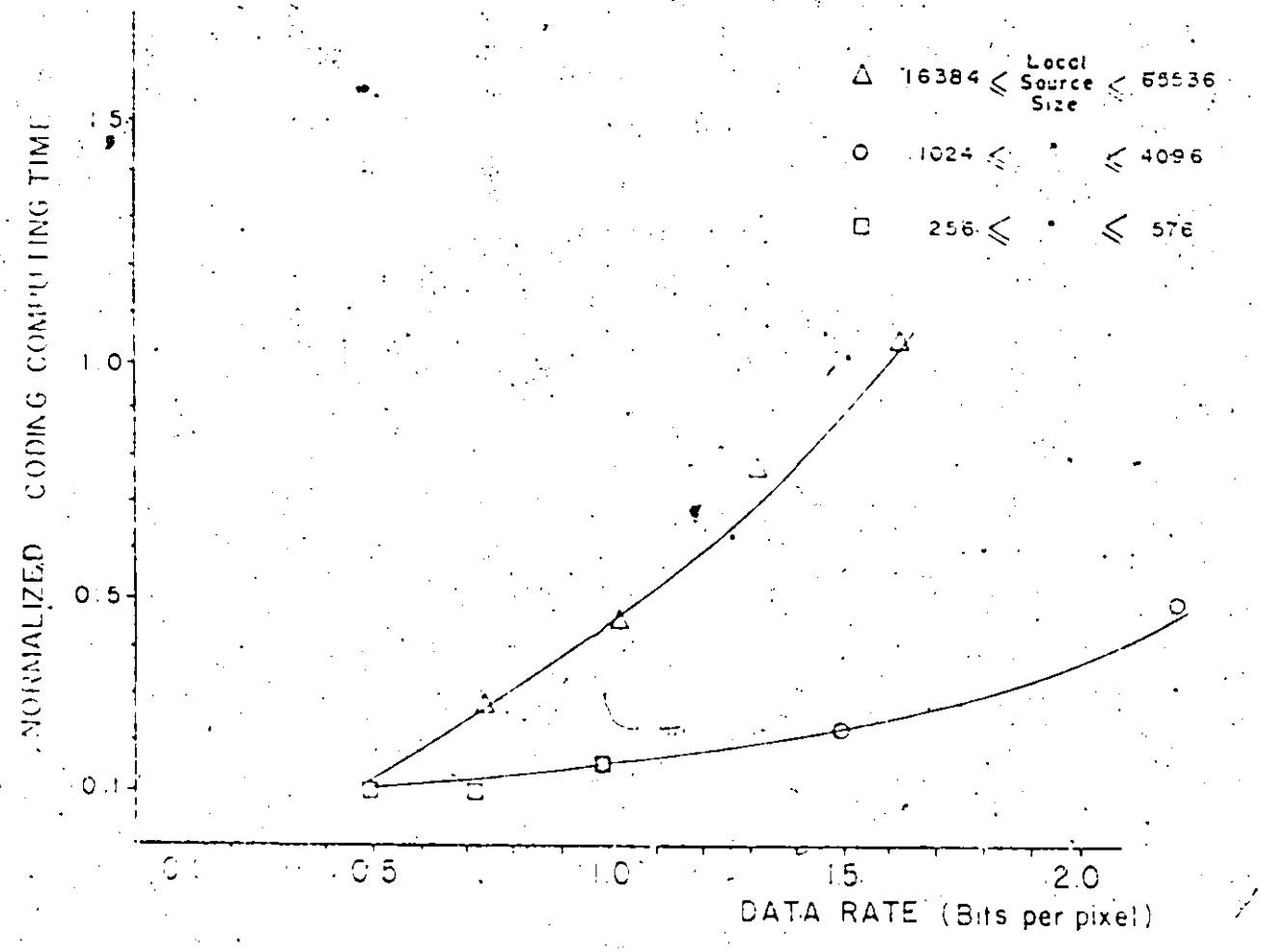


Figure 4.4.1-2 Computing time requirements for various local source sizes, for various values of K'. Two by two blocks were used for the decomposition process.

4.4.1.1 Photographic Interpretation of Results

Photographs of locally vector-quantized pictures are shown in Figure 4.4.1-3 and 4.' By comparing figure 4.4.1-3 to figure 4.2.1-2, one notices very little additional sub-

jective apperance degradation due to using small local sources. Furthermore no subpicture blocking-effect is visible even at bit rates of 1 bit per pixel.

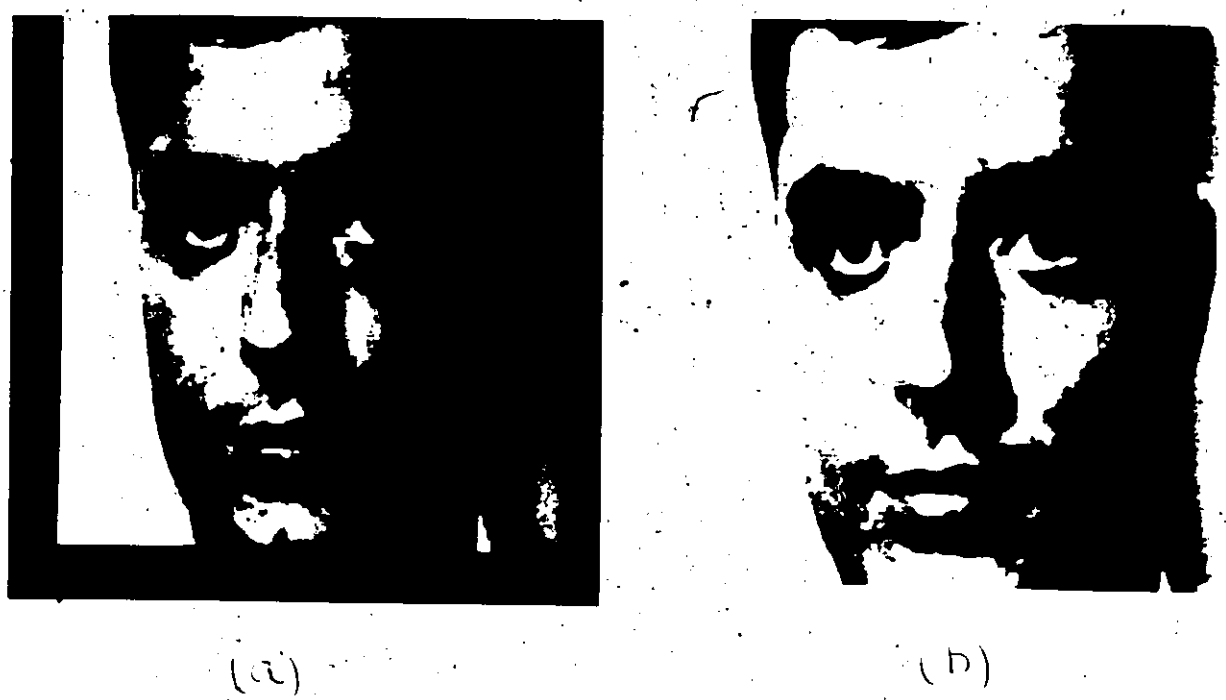
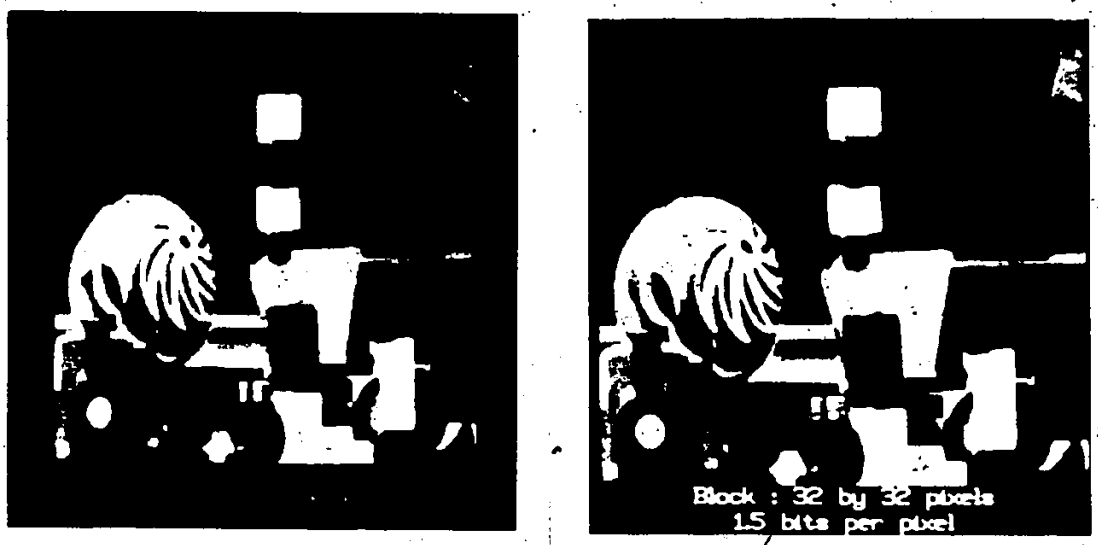


Figure 4.4.1-3 Example of local vector quantization of the "FRA" picture at various bit rates. The picture is processed as 64 non-overlapping square 32 by 32 pixel sources, each coded using K' reconstruction vectors. a) K' = 8 , 8 bits/vector component. NMSE : 1.22. Total data rate : 1.0 bits per pixel b) K' = 16, 8 bits/vector component. NMSE : 0.762. Total data rate : 1.5 bits/pixel.



(a) (b)

Figure 4.4.1-4 Example of local vector quantization of the "TAB" picture at various bit rates. The picture is processed as 64 non-overlapping square 32 by 32 pixel sources, each coded using K' reconstruction vectors. a) K' = 8 , 8 bits/vector component. NMSE : 2.85. Total data rate : 1.0 bits per pixel b) K' = 16, 8 bits/vector component. NMSE : 1.48. Total data rate : 1.5 bits/pixel.

The investigation of local source size was limited to the use of two by two blocks, to obtain a minimal overhead.

Larger blocks (e.g. three by three) produce a poorer distortion/compression ratio relative to two by two blocks, because of their non-trivial overhead when using small local sources.

4.4.2 Adaptive Local Source Quantization

In quantization of local sources the number of clusters K' was a constant throughout the entire image. However, since most image data are highly nonstationary [45], it is desirable to have a variable number of representative vectors per local source adaptively determined to meet the quality requirements for the data of each specific local source. For example, one local source may consist of data for which only two clusters are necessary to meet the quality requirements, while another local source may require sixteen clusters. Rather than use sixteen clusters for each local source, it is preferable to use a minimum of clusters for each local source. By adaptively setting K' for each local source, a lower average of K' can be used and thereby reduce the total data rate.

An adaptive clustering algorithm could be used for adaptively quantizing each local source. A well known adaptive clustering technique is the ISODATA algorithm [25]. This clustering algorithm deletes, splits or combines clusters at each iteration according to a user's specifications. This results in a variable number of clusters, K', according to

the local source statistics. The user supervision involves determining the lower and upper limits, on the number of clusters generated per local source, and the thresholds for deleting, splitting and merging of clusters. These thresholds and limits ideally allow the user to define the quality requirements in terms of 1) a minimum number of elements required for distinct representation, 2) maximum intracluster variance, 3) minimum intercluster distance, and 4) the minimum and maximum number of clusters per local source. These user specifications are constants and need to be specified only once for the entire image.

Extra complexity is introduced by adaptive local vector quantization; it comes from two sources, 1) the added complexity of the adaptive clustering algorithm, 2) the increased data system complexity resulting from the variable codeword ($\log_2 K'$) and codebook size from one local source to another and 3) the heuristics involved in determining the clustering parameters for each image type. Increased data system complexity is necessary, to ensure that the long-range average bit rate equals the channel capacity, and that at no time the buffer overflows or underflows. A controller is thus needed which given the constraints of buffer size and output bit stream rate, will allow more or less bits to be allocated to any given subimage depending on subimage complexity.

For the simulation of adaptive local quantization, the simple K-Means clustering scheme, with parametric intialization (c=2) is used. It is preceded by a measure of local source "busyness" (density of significant scene details) to determine, based on heuristics, the proper number of clusters required to encode the subimage. Once the appropriate number K' for a local source is determined, it is used by the K-Means clustering algorithm to produce K' quantization levels. K' values are limited to powers of two, for full use of binary registers.

The heuristics involved, are determined as follows : a large number of sample local sources are extracted from the test images of figure 3.1-1. The subimages are analysed subjectively to determine the appropriate number of clusters to encode each local source. Next the statistics, or "busyness", of the sample local sources are correlated to determine a set of thresholds. The set of thresholds are then used in classifying a local source into one of C "busyness" classes. The measurements are picture dependent, thus in this case only optimal for the pictures of figure 3.1-1. If the compression rate must be variable, then the repartition of the K' per local source will be made according to the specified bit rate.

The "busyness" or activity index can be measured in many different ways. Many non-information preserving compression

techniques [16,46,47,48,49,50] have proven the viability of adaptive coding. They used an activity index based on: energy measurement, variance, mean luminosity value of the local source, number of independent vectors or some other complicated rule, such as directionality and fineness [50].

The statistics used for activity index measurements are : the mean luminosity, variance and number of independent vectors generated at a resolution of 64 grey levels. The following set of classification thresholds were subjectively determined for 32 by 32 local sources extracted from the "FRA" picture.

Activity Class Number	K'	Luminosity Variance	Luminosity Mean	Number of histogram vectors
1	2	< 30		
2	4	> 30 AND < 85	> 70	
3	8	> 30 AND < 85	< 70	
		> 85 AND < 400	> 65	
		> 400 AND < 3000	< 70	
4	16	> 85 AND < 400	< 65	> 70
		> 400 AND < 3000	> 65	
5	32	> 3000		

Table 4.4.2-1 Decision thresholds for classifying a local 32 by 32 pixel source, from the "FRA" picture, into one of 5 "busyness" class.

Simulation results prove that slightly better quality reproduction is obtained under the mean-square-error criterion, for any data rate. The improvement, is of the order of 0.1 bit per pixel. The following graph presents the results.

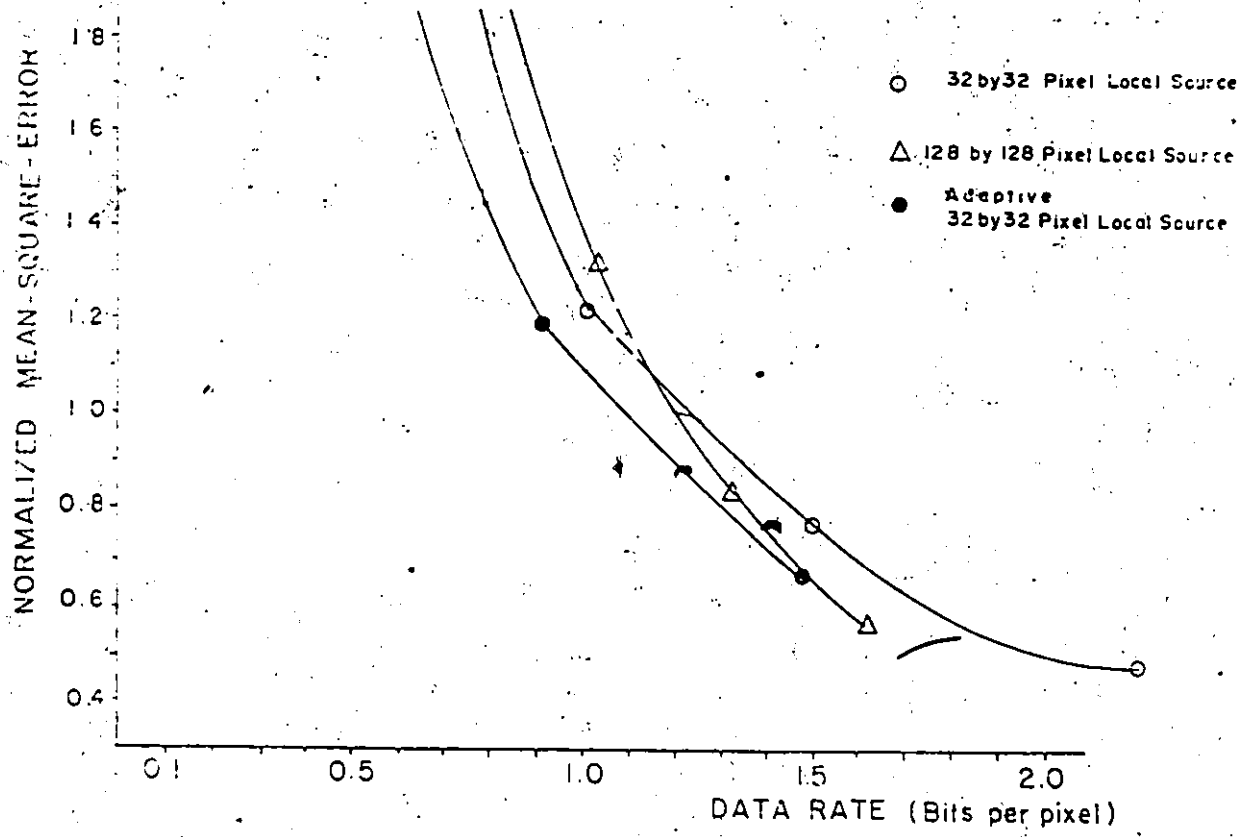


Figure 4.4.2-1 Percent Mean-Square-Error vs. Data Rate for adaptively varying K' for each local source of the FRA picture. 32 by 32 local sources and two by two pixel blocks are used. The results are compared to coding of 32 by 32 and 128 by 128 local sources, with a fixed value for K'.

Even if the activity classification process is optimized and automatized for any image type, this order of improvement is considered not to be worthwhile for the inherent buffering problems associated with adaptive encoding of lo-

cal sources ; thus no further investigation is undertaken on this coding option.

4.5 THE S-TRANSFORM

A powerful image decomposition technique is obtained by combining the S-transform [27] with vector quantization.

The S-transform decomposes an image in a step-like manner. At each step the image is partitioned into blocks of a*b pixels. Each block is then Hadamard transformed. The mean grey level values of each block are then assembled to form a copy of the original picture at lower spatial resolution. The difference coefficients from each block are aligned into a (a by b - 1) dimensional vector. The decomposition process is reiterated on the lower resolution image. The process stops when only a single mean grey level remains, corresponding to the overall mean grey level of the original image.

Decoding reconstructs the image, producing first a coarse approximation then in a step-like manner a finer and finer approximation. This reconstruction scheme allows rapid browsing through stacks of high resolution pictures on slow display systems.

At each step of the decomposition process, a list of difference-vectors and a matrix of grey level values is generated. The matrix of grey level values is a coarse represen-

tation of the picture decomposed. It is used for further decomposition. The list of vectors is compressed by vector quantization

The coarse picture obtained after a decomposition step, still contains most of the visual information of the original picture, but much less redundancy since it is a by b smaller. The decrease in redundancy is demonstrated by the variance of the difference coefficients of each step. The results of the following table were obtained from the S-tranform decomposition of the "TAB" picture, of figure 3.1-1.c, using two by two blocks.

Picture matrix size (pixels)	Decomp. step	Number of vectors	Variance of difference coefficients		
			Ver.	Hor.	Dia.
256 by 256	1	16384	24.5	60.1	3.0
128 by 128	2	4096	64.4	121.5	11.8
64 by 64	3	1024	100.4	134.6	27.6
32 by 32	4	256	155.7	218.0	37.8
16 by 16	5	64	143.4	194.6	48.3
8 by 8	6	16	181.8	141.9	99.1
4 by 4	7	4	85.5	13.0	11.6

Table 4.5-1 Variance of the Hadamard difference coefficients of the successive steps in the decomposition of the "TAB" picture of figure 3.1-1.c. Two by two blocks were used for decompositon and the Hadamard transformation produced a sum coefficient (mean grey level) and three difference coefficients.

The variance of the difference coefficients increases up to the last step where only 4 vectors are produced,

It was shown in section 4.4.2, that the variance was a fairly good measure of "busyness" : i.e. vector space density ; thus each successive step of decomposition would require more representative vectors for quantization than the preceding, but since the next step of decomposition produces four times less vectors than its preceding step, the increase in number of representative vector, K, should not be exponential.

The required number of representative vectors to code each step can not be determined by any simple means (section 4.4.2). The compression effect of each step is studied independently. The three test pictures of figure 3.1-1, were coded using two by two blocks (for fast coding time), and vector quantized by the K-means algorithm, using the Euclidian distance and mode-seeking initialization (since the tranform coefficients are uncorrelated). Only four steps of the S-tranform were performed. No worthwhile gain is obtained by vector quantizing further steps, because of their low vector count.

The following observations were made :

- Step 1 Fine image detail. Few representative vectors are necessary for this step, 8 to 16 is satisfactory,
- Step 2 Coarse image detail. The vector set from this step must be well representated; 32 to 64

vectors are necessary.

- Step 3 This step is of major importance. A patchwork-effect occurs if less than 64 representative vectors are used.
- Step 4 Same problems as in step 3 occur. To eliminate the blocking-effect, 64 to 128 clusters are required.

The number of representsative vectors assigned to each step of the S-tranform, will follow the previous worst-case observations.

4.5.1 Bit-rate of a Vector-quantized S-transfomed Picture

The bit rate of a vector-quantized S-transform decomposed image, must include the codebook from each step. An example of the bit-rate calculation is given in table 4.5.1-1.

Step	Number of representative vectors	Number of bits per pixel
1	16	1.00585
2	64	0.39844
3	128	0.15625
4	128	+ 0.07422
		1.63476
Coarse image matrix		+ 0.03125
		1.666 bits per pixel

Number of representative vectors	Codeword contribution (bits)	Codebook (bits)
16	128 * 128 * 4	16 * 3 * 8
64	64 * 64 * 6	64 * 3 * 8
128	32 * 32 * 7	128 * 3 * 8
128	16 * 16 * 7	128 * 3 * 8
Coarse image matrix	16* 16 * 8	
TOTAL1	101 120	+ 8064
TOTAL 2	109 184 bits	

$$\text{Number of bits per pixel} \quad \frac{109\ 184}{256 * 256} = 1.666 \text{ Bits per pixel}$$

Table 4.5.1-1 Example of the bit rate calculation of 256 by 256 picture decomposed using the S-transform on two by two blocks, and vector quantization on the difference vectors obtained at each step of the transformation.

The dynamic range of Hadamard transform coefficients, in this case, quadruples after each tranformation step. Only eight bits per representative vector coefficient produced negligible image degradation, since the Hadamard transform is very robust to round-off errors [51].

Although at each step of transformation, the number of representative vectors vary, the maximum and minimum number can be fixed. This sets an upper bound and a lower bound on the output bit rate which can be used for a more effective control of system output buffering.

4.5.2 Transmission Time

The transmission time on a 1200 baud telephone line, for each step of the S-tranform image of table 4.5.1-1, is as follows :

Image	Number of bits	Transmission time in minutes
PCM image	524 288	7:16.91
S-tranform image		
Base matrix	2 048	0:01.71
Step 1	65 920	0:54.93
Step 2	26 112	0:21.76
Step 3	10 240	0:08.53
Step 4	4 864	0:04.05
Total transmission time		1:30.98

Table 4.5.2-1 Transmission time calculation for a 256 by 256 pixel image decomposed using 2 by 2 blocks and the S-transform , followed by vector quantization. 16,64,128 and 128 representative vectors were used respectively per decomposition step.

Only 13 seconds (base matrix + step 4 + step 3) of transmission time is required to obtain the picture matrix of step three. The picture at this step of reconstruction is already clearly intelligible.

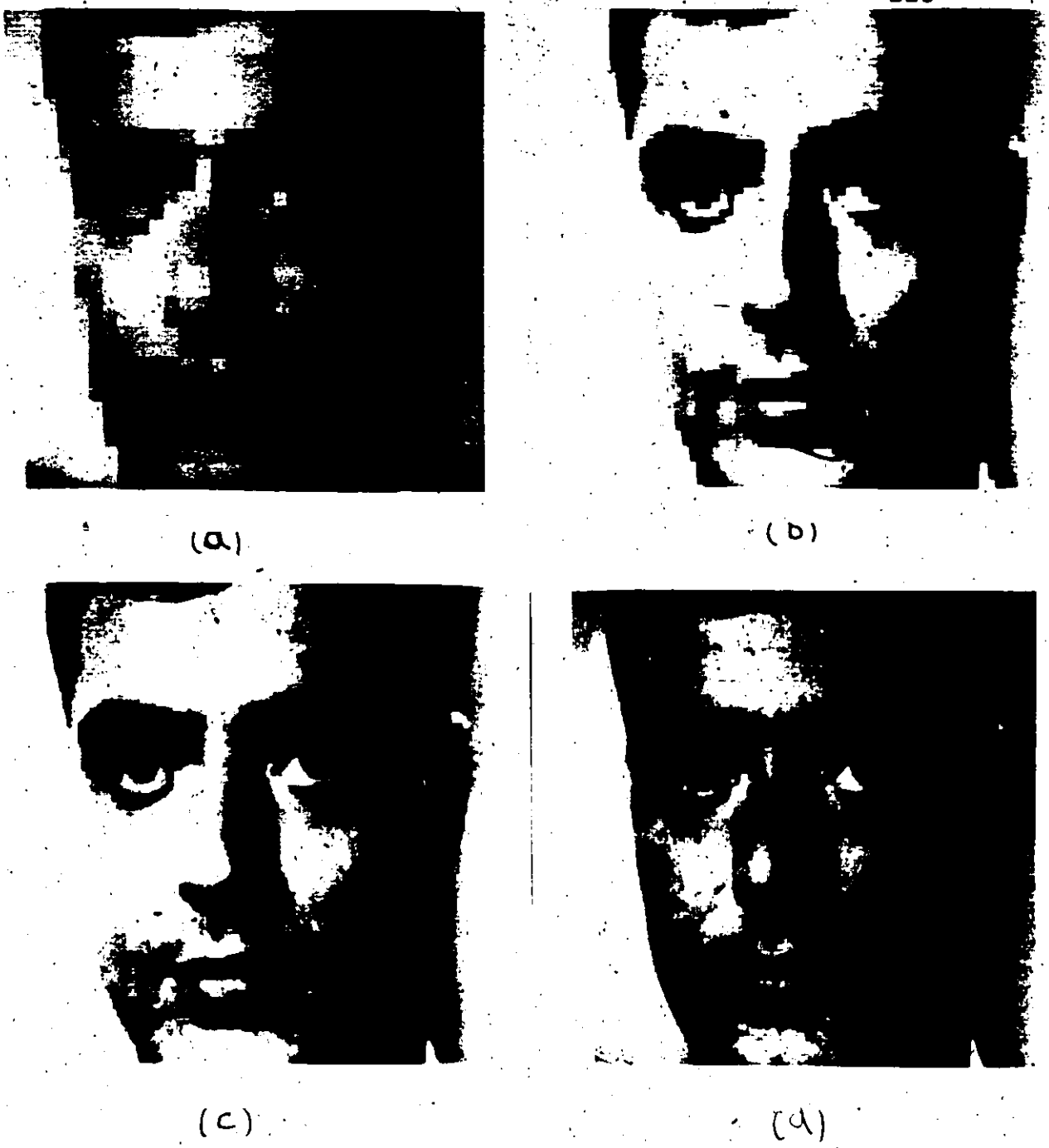


Figure 4.5.2-1 Displayable pictures after each step of the reconstruction process, using a total bit/pixel assignment of 1.67 bits/pixel. a) step four , b) step three , c) step two , d) step one : full resolution picture (only 1.67 bits/pixel used).

4.5.3 Photographic Interpretation

The S-tranforms decompositon of the "FRA" and "TAB" pictures (figure 3.1-1 b and c) were vector-quantized, using different representative vector assignment.

The artifacts produced by the quantization of tranform coefficient vectors, is especially visible in figure 4.5.3-1.c : diagonal edges are poorly represented. This is a by-product of poorly quantizing the difference vectors of step two and three. One can notice that no false-contouring problem occurs at 1 bit per pixel. This is explained by the masking effect of one step over another; a dithered type image is produced. The quality of the reconstructed picture is not as good as those produced in figure 4.2.1, for the same bit rate. The compression is not as efficient since only three instead of four-dimensional vectors are processed; i.e. a codeword represents in the above simulations three pixels, instead of four.

Better compression results are expected if larger blocks (four by four) are used for decompositon, but this at the expense of more computing time for coding.

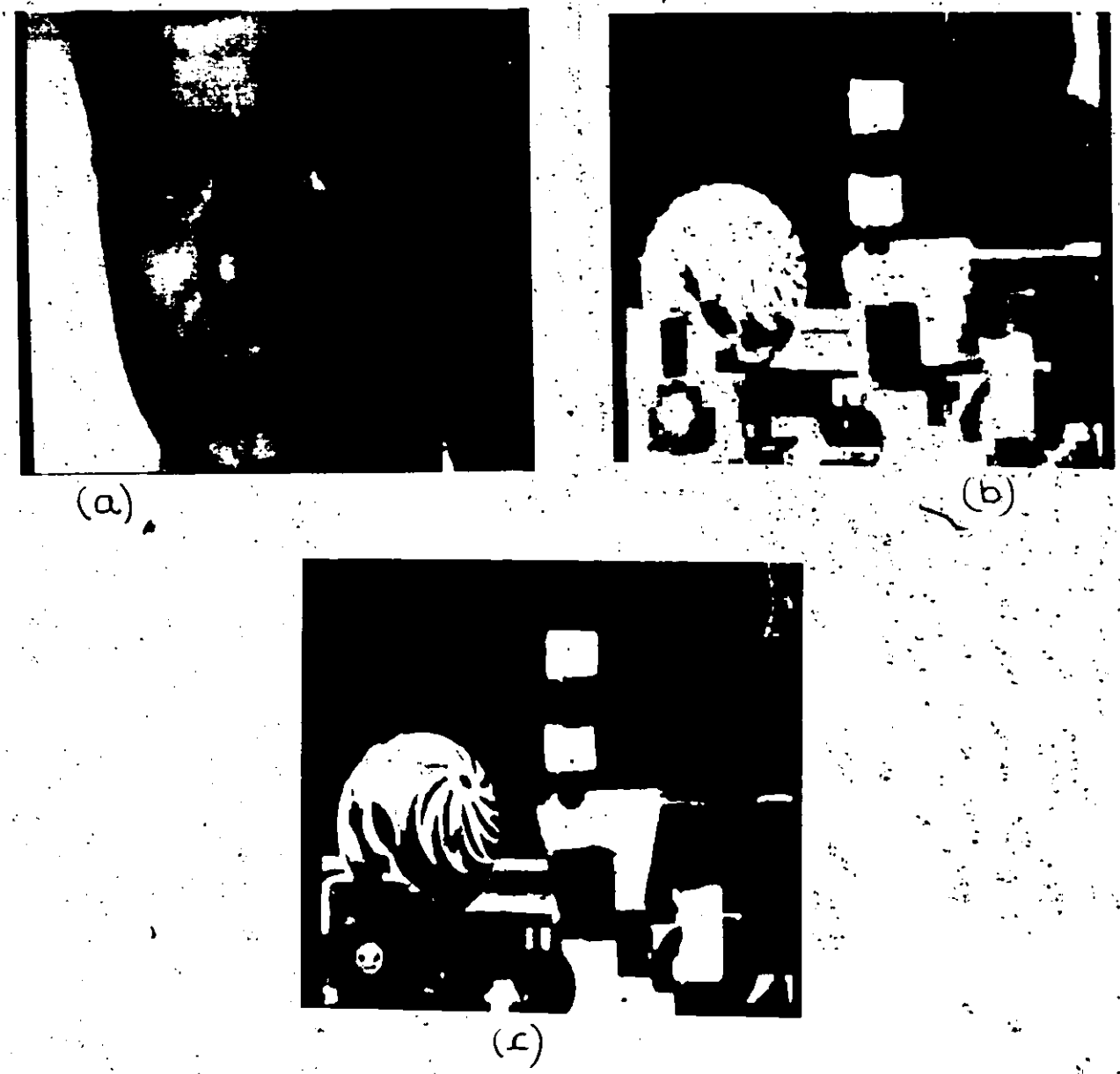


Figure 4.5.3-1 Decoded pictures resulting from four steps of S-tranform decomposition on two by two blocks, and vector quantization of the difference vectors of each step. a) "FRA" picture, coded using only 1.05 bits/pixel : i.e. 4, 32, 64, 128 representative difference vectors respectively for step one through four. b) "TAB" picture, coded using only 1.05 bits/pixel : i.e. 4, 32, 64, 128 representative difference vectors respectively for step one through four. C) "TAB" picture, coded using only 1.67 bits/pixel : i.e. 16, 64, 128, 128 representative difference vectors respectively for step one through four.

4.6 PERFORMANCE COMPARISON TO OTHER COMPRESSION TECHNIQUES

Two vector quantization strategies can be recommended. A fast coding scheme would use two by two pixel blocks for decomposition and 32 by 32 pixel local sources. Histogram preprocessing and limiting the number of iterations to five assures fast computability, e.g. around one minute of computing time to code a 256 by 256 pixel picture on a Vax-750.

A more efficient scheme would use three by three blocks for decomposition with decorrelation and histogram preprocessing, on 128 by 128 subpictures. Computing time would be 10 times greater than the previous scheme, but with a gain in compression of up to 0.3 bits per pixel.

Progressive transmission is also possible by using a combination of the S-tranform decomposition technique and vector-quantization.

Test were performed to compare the normalized mean-square-error (1.2.2.2-3) performance of vector quantization with some well-known compression techniques. The techniques used for comparaison are block Hadamard [14] and cosine [16] transform coding (section 1.3.2) and adaptive DPCM (section 1.3.1). The simulations were carried out using the monochrome test pictures of figure 3.1-1.

Block transform coding involved taking the two-dimensional transform of 16 by 16 pixel blocks. The transform fre-

quency coefficients are Max-quantized [2], using a Gaussian distribution model. The coefficients are quantized using a different number of levels according to their variance, and according to the required data rate [16]. The mean component is quantized linearly to 8 bits.

The adaptive DPCM technique is also performed using a*a blocks. A predictor is adaptively designed for each block according to the local statistics. 1 bit per pixel is used in coding the predictor error signal and some overhead for the predictor coefficients. The block size was varied from 8 by 8 to 32 by 32 pixels, to obtain various bit rates.

The normalized mean-square-error vs. total data rate curves comparing these techniques to vector quantization are shown in figure 4.6-1. The mean-square-error correlates well with subjective quality in this case, apart from some artifacts which are worth noting. It is observed that DPCM performs as well or slightly better than vector quantization. Cosine transform coding outperforms all other techniques except at around 1.5 bits per pixel where vector quantization using three by three pixel blocks is best. An annoying blocking-effect is visible in tranform-coded pictures, especially at data rates above 1.5 bits per pixel, while a muddled-effect appears at low data rates. Vector-quantized data produces a less noisy picture than block transform coding, though false-contouring becomes annoying

at data rates below 1 bit per pixel. The same type of behaviour is obtained for the "BAT" picture.

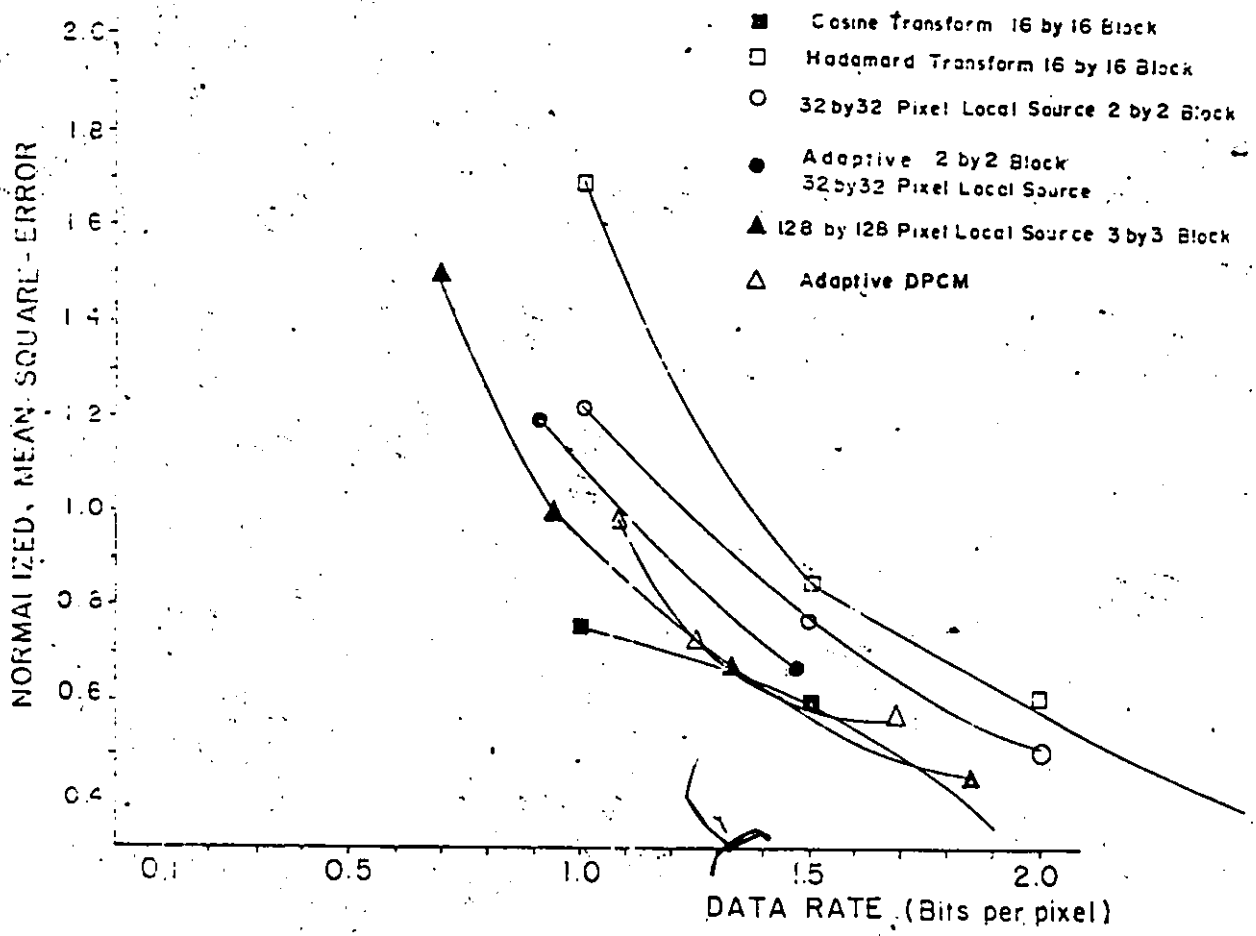


Figure 4.6-1 Perfomance comparaison of block transform coding, adaptive DPCM coding and vector quantization using the "FRA" training set.

Although cosine tranform coding outperforms vector quantization, it has the implicit disadvantage of requiring variable-length codewords for efficient coding (as does the Hadamard transform coding scheme).

If the decoder is assumed to be able to handle variable-length codewords, then one might as well use postprocessing to further reduce the data rate using entropy coding. Hilbert [26] proved that statistical coding of the codeword sequence generated from a vector-quantized colour picture (the three colour components of a pixel are used as a vector), produced a saving of 0.3 bits per pixel. Although the statistics of the codeword list generated from colour vector quantization is different from spatial vector quantization, the same order of saving is expected.

Hence vector quantization fares very well compared to both adaptive DPCM and block tranform coding, without the noise sensitivity of DPCM and the complexity of transform decoding.

Chapter V

COLOUR IMAGE CODING

Digital coding of colour video signals has received less attention than the coding of monochrome video signals. However, given the current widespread proliferation of colour television systems and the general preference for colour pictures versus monochrome, it is obvious that the efficient coding of colour signals is of prime importance.

5.1 CODING OF COLOUR TELEVISION IMAGES

A poor understanding of the human visual process has not prevented the use of coding in broadcast colour television systems [52]. Efficient use is made of analog bandwidth, to accomodate the increased information content of colour signals. A colour television camera provides simultaneous red, green and blue line scans of a scene, according to the National Television System Commission (NTSC) receiver phosphor primary system.

These are combined to form the scene luminance (brightness) according to equation 3.1.1.

The luminance signal is transmitted along with two chrominance signals I , Q that are linear functions of the col-

our difference signals R-Y and B-Y [1]. The colour differences are normalized to form the signals

$$U = \frac{R - Y}{1.14} = 0.877 * (R - Y) \qquad (5.1\text{-}1)$$

$$V = \frac{B - Y}{2.03} = 0.493 * (B - Y) \qquad (5.1\text{-}2)$$

The normalization, which is standard for the NTSC, PAL and SECAM systems [1], was chosen to limit the maximum excursion of the composite colour television signal to the arbitrary value of 1.33 times the excursion of the monochrome television signal. In the NTSC system the I and Q chrominance signals are formed by

$$I = U * \cos(33^\circ) - V * \sin(33^\circ) \qquad (5.1\text{-}3)$$

$$Q = U * \cos(33^\circ) + V * \sin(33^\circ) \qquad (5.1\text{-}4)$$

Alternatively the I and Q signals can be directly related to the R, G, B camera signals by

$$I = 0.596 * R - 0.274 * G - 0.322 * B \qquad (5.1\text{-}5)$$

$$Q = 0.211 * R - 0.523 * G + 0.312 * B \qquad (5.1\text{-}6)$$

For transmission the Y,I,Q components are low-pass filtered uni-dimensionaly (along the scan line). The luminance signal is limited to a bandwidth of about 4.0 MHz, while the I and Q signals are bandlimited to about 1.3 and 0.5 MHz,

respectively. This severe bandlimiting of I and Q at the transmitter affects the frequency spectrum of the reconstructed R,G,B signals at the receiver, of course, but the degradation to a human viewer has proved minimal because of the poor spatial frequency response of the human visual system to coloured light [1].

The R,G,B drive signals are reconverted for display according to the relations

$$R = 1.000 * Y + 0.956 * I + 0.621 * Q \qquad (5.1\text{-}7)$$

$$G = 1.000 * Y - 0.272 * I - 0.647 * Q \qquad (5.1\text{-}8)$$

$$B = 1.000 * Y - 1.106 * I + 1.703 * Q \qquad (5.1\text{-}9)$$

The coordinate conversion from the R,G,B colour space to the Y,I,Q colour space can be considered as a three-dimensional transformation. The conversion provides an energy compaction between colour planes, making bandlimiting possible. Adopting this philosophy, the near-optimal three-dimensional transform would be a Karhunen-Loeve transformation which completely decorrelates the 3*N*M colour image components. It has been shown [53] that the Y,I,Q coordinate conversion provides almost as high an energy compaction for colour images as does the Karhunen-Loeve colour coordinate conversion, with the advantage of simpler computability. The energy compaction of the Y,I,Q coordinate system is illustrated in Table 5.1-1.

Test Image	Coordinate System	Percentage energy in first component	Percentage energy in second component	Percentage energy in third component
BAT	RGB	28.53	36.93	34.54
	YIQ	92.70	4.63	2.67
FRA	RGB	31.56	34.37	34.07
	YIQ	93.50	5.23	1.27

Table 5.1-1 Energy compaction produced by coordinate conversion from R,G,B to Y,I,Q systems.

The R,G,B components of a natural picture, are highly inter-correlated [1]. In the Y,I,Q representation, the luminance plane contains most of the picture's energy and detail, while the energy in the I and Q plane is rather sparce [1]. Correlation between the Y,I,Q components still exists. Even by using the Karhunen-Loeve transform, correlation between the planes will still exist, because of the non-Gaussian nature of picture statistics [53].

In the following simulations, the Y,I,Q coordinate system is used for decorrelation of the R,G,B components, because of its ubiquity in television broadcast systems.

5.2 VECTOR QUANTIZATION OF COLOUR IMAGERY

Earlier attempts in coding digital colour pictures used unitary transforms for spatial decorrelation of the Y,I,Q planes [53,16]. Compression is achieved by either eliminating some transform coefficients (brick-wall filtering) or quantizing each coefficient to different resolutions, according to its variance (exponential filtering). This type of compression corresponds to two-dimensional filtering instead of uni-dimensional filtering, used in broadcast television.

Colour imagery can be vector quantized, in a similar manner. The image is first spectrally decorrelated by way of the Y,I,Q coordinate systems. Each Y,I,Q plane is vector quantized independently, as if it were a monochrome image. The luminance plane is exactly that: a monochrome image, hence the results from chapter four are directly applicable. The I and Q planes can be coarsely quantized, because of their low energy content.

Another vector quantization strategy is to quantize the colours directly. Each colour image pixel is composed of three components. Each triplet corresponds to a CIE space colour. A pixel can be processed as a three dimensional vector, hence the name colour quantization. Two authors [26,39] studied this coding strategy. Although Hilbert [26] achieved high compression rates, the spatial redundancy ex-

isting in each plane, is not efficiently reduced. Hilbert [26] by using statistical coding, further increased the compression ratio at the expense of variable length coding. Advantages of colour quantization are that a single codeword map is required (instead of three as in Y,I,Q plane coding), and the use of decorrelation transform functions (e.g. Y,I,Q coordinate system)is not necessary.

This coding scheme can be improved, by vector quantizing both the colour and spatial information simultaneously : i.e: by combining in a single vector, the R, G and B component intensity values from an a*b pixel block. For a two by two spatial block, twelve-dimensional data are produced, four intensities from the R plane, four from the G plane and so on. This high dimensionality requires excessive computing requirements (section 4.2.2).

In television broadcasting, the I and Q signals are severely bandlimited because of their low energy content. The subjective error produced by the bandlimiting is small [1]. A similar "trick" can be used for reducing the coding time associated with vector quantization. The data dimensionality is reduced, by first transforming the data into the Y,I,Q coordinate system, then lowering by four the spatial resolution of the low energy I and Q planes, hence M/2 by N/2 instead of M by N pixel planes are processed. The dimensionality is thus effectively reduced to six for two by two

pixel blocks : four values from the Y plane, one from I, and another from the Q plane. "D" in (4.2-1) is reduced by half, correspondingly the processing requirements are reduced by half, and an initial compression ratio of two is obtained. The decoder generates the R,G,B components by first blowing-up the I and Q planes, before inverse transformation.

The quantization error produced by lowering the spatial resolution of the I and Q planes, is masked out by the other components and by the compressed version of the planes themselves. The subjective error is expected to be lower than in the case of monochrome picture coding, because of the spatial frequency limitations of the human visual system, to colour [1]. Figure 5.2-1 illustrates the block diagram of the proposed preprocessing for vector quantization of colour imagery.

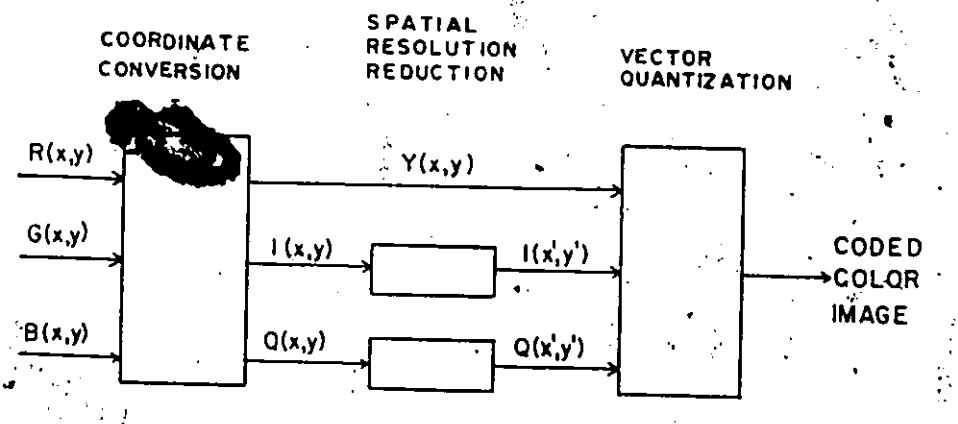


Figure 5.2-1 Preprocessing of the R,G,B tristimulus values for vector quantization

The proposed preprocessing scheme introduces quantization error in the subsampling process. The Y,I,Q coordinate conversion produces "real" numbers. Subsampling of a plane is performed by replacing each two by two pixel block by its average component value. The subsampling quantization error is barely noticeable subjectively.

Two colour vector quantization techniques are simulated in the following subsections : independent Y,I,Q plane coding and combined Y,I,Q coding. Both strategies use the I and Q planes at quarter spatial resolution, reducing both computing requirements and the data rate by half.

The colour test pictures are the colour version of the picture of figure 3.1-1. The iterative K-Means clustering algorithm is used for vector space partitioning, initialized by the mode-seeking scheme since the Y,I,Q components are uncorrelated.

The performance of the two coding strategies are measured in terms of data quality and data rate. The data quality is observed subjectively, and measured using the relative mean-square-error, of formula 1.2.2.2-1 through 1.2.2.2-3, between the original R,G,B tristimulus planes and the R,G,B planes of the image approximation.

The data rate is calculated according to 4.1-1 to 3, for each plane. The I and Q planes are at quarter spatial reso-

lution, thus the bit rate calculated by 4.1-3 must be divided by four.

In the following simulation results, the representative vector components corresponding to the luminance and chrominance components are quantized using 256 (8 bits) linear levels.

The coding time for Y,I,Q imagery is approximately 1.5 times that of a monochrome imagery, since the I and Q planes have four times less vectors to quantize than a monochrome image, and require fewer quantization levels because of their low energy content.

5.2.1 Y,I,Q Plane Coding

In this section, each luminance-chrominance plane is coded as if it were a monochrome image. This coding scheme is referred to as plane coding. The data distribution of a local source determines the number of quantization levels necessary for compression. The I and Q planes, of low energy require fewer quantization levels than the Y plane. Each plane is decomposed using two by two pixel blocks, and 32 by 32 pixel local sources for coding, for fast coding time.

Proper analytical rules for determining the number of representative vectors for each plane cannot be formulated, since the human visual process of colour vision is poorly understood [53]. The Y plane being the monochrome version

of a colour image, is known to require (section 4.4.1) 16 quantization levels per local source of 32 by 32 pixels, or 1.5 bits per pixel, to achieve good subjective reproduction. The I and Q planes require much less quantization levels, but since their spatial resolution is reduced by four, around 8 quantization levels are used per local source, for a bit rate of 0.25 bits per pixel per plane. In this study the chrominance planes are coded using the same number of representative vectors, although it is common practice to bandlimit the Q plane more than the I plane. Any further reduction in the bit rate of the Q plane is negligible, since its contribution to the total bit rate is so small.

Mean-square-error measures for the following representative vector assignment are presented : 32-8-8 , 8-8-8 and 8-2-2 for the Y, I and Q planes respectively.

NUMBER OF REPRESENTATIVE VECTORS			BIT RATE bits per pixel	% MEAN-SQUARE-ERROR		
Y	I	Q		R	G	B
8	2	2	1.15	8.1	4.1	14.0
8	8	8	1.50	5.9	3.7	12.0
32	8	8	2.75	5.1	2.2	9.8

Table 5.2.1-1 Y,I,Q plane coding by vector quantization, of the colour "BAT" picture. Two by two blocks were used for decomposition, 32 by 32 local sources for coding. The chrominance planes were spatially reduced by four before processing.

The use of the Y,I,Q coordinate system produces lower mean-square-error for the green plane, since it is well represented by the luminance component which is always efficiently coded.

Photographic results are not presented, since the colour prints do not properly reproduce displayable results. Subjectively, it was observed that an acceptable colour picture was obtained with only 2.75 bits per pixel, for a compression ratio of 8.7 . The compression ratio obtained is higher than for monochrome images, because of the subsampling process introducing a compression ratio of two. Thus the effective compression ratio of vector quantization of colour pictures is the same as for monochrome imagery (approximately 6.7). Approximately 1.3 bits per pixel is required to transmit the luminance information, and a further 1.5 bits per pixel for the chrominance information of a picture.

5.2.2 Combined Coding of the Y,I,Q Components

Combined coding of the Y,I,Q components involves decomposing the luminance-chrominance picture into blocks of two by two pixels. The chrominance planes are spatially reduced by four, thus a chrominance coefficient is associated to each two by two pixel block of the luminance plane. A vector is formed of four luminance components and two chrominance components, one from the I and Q planes respectively.

This six-dimensional vector is quantized using the K-means algorithm with mode-seeking initialization. By varying K, the number of representative vectors, various bit rates are obtained.

Coding is performed using 32 by 32 local sources, for fast processing time. Better results are obtained by using larger local sources but at the expense of a large increase in computing time. Performance results can be judged from the following table.

NUMBER OF REPRESENTATIVE VECTORS	BIT RATE bits per pixel	% MEAN-SQUARE-ERROR		
		R	G	B
8	1.13	6.8	3.9	8.8
16	1.75	5.4	2.9	7.4
32	2.75	4.6	2.2	6.4

Table 5.2.2-1 Y,I,Q combined coding by vector quantization, of the colour "BAT" picture. The chrominance planes were spatially reduced by four before processing. Vectors are formed of four luminance and two chrominance coefficients, from a two by two pixel block. 32 by 32 local sources are used for coding.

Combined Y,I,Q component coding slightly outperforms plane coding, according to the mean-square-error criterion, with the added advantage of producing a single codeword list, instead of three as in plane coding. This results in

a simpler decoder architecture. The slight improvement in mean-square-error results from minimizing this criterion in a multispectral sense, instead of independently for each plane.

5.3 SUMMARY

Two colour picture vector quantization schemes removing spatial redundancy were described. These make use of the Y,I,Q coordinate system. One technique involves coding each Y,I and Q plane as if it were a monochrome image. The second technique uses a vector formed of Y,I and Q components, from a two by two pixel block. In both cases the I and Q planes are spatially reduced by four because of their low information content. Combined coding of the Y,I,Q luminance-chrominance components from pixel blocks slightly outperformed spatial plane coding. Furthermore, combined coding of the Y,I,Q components produces a single plane of codewords, thus a simpler decoder architecture. Compression ratios of more than 8 were obtained with only limited degradation in both cases.

Vector quantization of colour pictures compares favorably to block-transform coding. Cosine transform coding of colour pictures was performed by coding the Y and subsampled I and Q planes as if they were monochrome pictures. The monochrome transform coding scheme was described in section 4.6. Performance results can be judged from the following table.

BIT RATE bits per pixel	% MEAN-SQUARE-ERROR		
	R	G	B
1.00	8.0	4.4	10.2
1.50	8.4	3.8	10.2
2.00	7.2	3.4	9.1

Table 5.3-1 Cosine transform coding of the colour "BAT" picture. The chrominance planes were spatially reduced by four before processing. 16 by 16 local sources are used for coding.

Subjective comparison of the two colour picture vector quantization techniques with the cosine transform technique, shows the vector-quantized colour picture to be again "clearer" in appearance compared to the transform picture, although some false-contouring occurs at low bit rates (1.5 bits per pixel). Vector quantization produces a less noisy picture since the quantizer is designed to minimize the average error in a multispectral sense, rather than trying to minimize the average error of each plane independently. Futhermore no blocking-effect comparable to block-transform coding occurs in the vector-quantized picture.

Chapter VI

SUMMARY AND RECOMMENDATIONS FOR FURTHER RESEARCH

In this dissertation, adaptive vector quantization is investigated for simple and efficient compression of pictorial data. Adaptive vector quantization consists in designing a quantizer for each image source. Clustering is used for quantizer design. The K-means clustering algorithm is extensively investigated and simulated to ascertain its use in adaptive vector quantization.

Bit-rate reductions by a factor of more than seven and ten for monochrome and colour pictures respectively, produce only limited degradation. The combination of progressive transmission and vector quantization also provides good compression results.

Although decoding of vector-quantized data is simplistic and makes use of a very small refresh memory, coding of a 256 by 256 pixel image requires minutes of computing time on a VAX-750.

Vector quantization is limited for the moment to off-line coding applications. An important field of application for vector quantization is broadcast systems where the decoder must be of low cost. Videotex systems are one good example.

A fast quantizer design algorithm is required for practical and real-time vector quantization. A classification scheme based on the Hadamard transform is proposed. An image is decomposed into four by four pixel blocks. Each block of grey level values is first Hadamard transformed, in either two or three dimensions for a static or a sequence of images respectively. The sum coefficient, or mean value is extracted and quantized to as few as four bits [41]. The Hadamard difference coefficients corresponding to contrast measures, are weighted according to the mean luminosity of the pixel block, to take into account Weber's effect [1]. Only the first, most important difference coefficients are used for classification, to reduce vector dimensionality and thus coding time. Classification is performed in two steps. The first step consists in replacing each coefficient by a label specifying which type of contrast it represents : i.e. high positive contrast, low positive contrast, no contrast, etc. This first step is performed using simple thresholding. The second step is the classification of the pattern vectors. Each vector of transform coefficients, represented by a contrast pattern vector of integer labels, is classified using simple table look-up techniques. Each pattern vector is then replaced by its contrast pattern class number.

The coded output consists of a sum coefficient for each pixel block and a contrast pattern label number. If one

wishes progressive transmission, then the S-transform technique is used on the sum coefficients.

Decoding is performed via look-up tables and inverse Hadamard transformation, requiring only add, subtract and shift operations.

REFERENCES

[1] Pratt W. K., "Digital Image Processing". A Wiley-Interscience Publication, John Wiley and sons. New-York 1978.

[2] Max J. , "Quantizing for Minimum Distortion". IRE Trans. Information Theory, vol.IT-6, March 1960,pp.7-12.

[3] Gonzalez R.C., Wintz P. , "Digital Image Processing". Addison-Wesley Publishing Company, Inc. Don Mills, Ontario, 1977.

[4] Jain A.K. , "Image Data Compression: A Review". Proceedings of the IEEE, vol.69, no.3, March 1981, p.349.

[5] Knowlton K. , "Progressive Transmission of Grey-scale and Binary Pictures by Simple, Efficient, and Lossless Encoding Schemes". Proceedings of the IEEE, vol.68 no.7, July 1980, pp.885-896.

[6] Shannon C.E., "The Mathematical Theory of Communication". Urbana Ill. : University of Illinois Press, 1949.

[7] Schreiber W.F. , "Picture Coding". Proceedings of the IEEE, vol.55, no.3 March 1967, pp.320-330.

[8] Sakrinson D.J., "On the role of the Observer and a Distortion Measure in Image-Transmission". IEEE Tran. on Comm., Vol COM-25 #11, November 1977, pp. 1251-1267.

[9] Habibi A. , "Survey of Adaptive Image Coding Techniques". IEEE Transactions on Communications, vol.COM-25, no.11, Nov. 1977, pp.1275-84.

[10] Arguello R. J., Sellner H. R. and A.A. Stuller. "The effect of Channel Errors in the DPCM transmission of Sample Imagery". IEEE Trans. on Comm. Technology, Vol. COM-19, pp 926-933, Dec. 1971.

[11] Modestimo J. W., Daut D.J. "Combined Source-channel Coding of Images". IEEE Trans. on Comm., Vol. COM-27 #1, pp. 1644-1659.

[12] Connor D.J. "Techniques for Reducing the Visibility of Transmission Errors in Digitally Encoded Video Signals". IEEE Trans. on Comm., Vol. COM-21 #6, pp. 695-706, June 1973.

[13] Wintz P.A. , "Transform Picture Coding". Proceedings of the IEEE vol.60, no.7, July 1972, pp.809-820.

[14] Pratt W.K. , Kane J. , Andrews H.C. , "Hadamard Transform Image Coding". Proceedings of the IEEE, January 1969, pp.58-68.

[15] Anderson G.B., Huang T.S., "Piecewise Fourier Transformation for Picture Bandwidth Compression". IEEE Trans. On Communication Technology, vol.COM-19, no.2, April 1971, pp.133-140.

[16] Chen W-H. and Smith H. , "Adaptive Coding of Monochrome and Color Images". IEEE Transactions on Communications, vol.COM-25, no.11, November 1977, pp.1285-1292.

[17] Mitchell O.R. ,and Tabatabai A. , "Adaptive Transform Image Coding Error recovery". 9.3, pp.92-3.

[18] Habibi A. , "Hybrid Coding of Pictorial Data". IEEE Trans. on Communications, May 1974, pp.240-250.

[19] Delp E.J. , Mitchell O.R. , "Image Compression Using Block Truncation Coding". IEEE Trans. on Communications vol. COM-27 no.9 September 1979 pp.1335-1342.

[20] Mitchell O.R. ,and Delp E. , "Multilevel Graphics Representation Using Block Truncation Coding". Proceedings of the IEEE vol.68 no.7, July 1980 pp.868-873.

[21] Goeddel T.W., Bass S.C., "A Two-dimensional Quantizer for Coding of Digital Imagery". IEEE Trans. on Communications, Vol. COM-29 #1, January 1981.

[22] Mitchell O.R. , Bass S.C. , Edwards J.D. , "Image Coding for Photoanalysis". Proceedings of the SID, vol.21/3, 1980 pp.279-292.

[23] Gersho A. , Ramamurthi B. , "Image Coding Using Vector Quantization". IEEE ICASSP May 1982, pp.428-431.

[24] Wendler T.H. , Meyer-Ebrecht D. , "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 for Medical Applications, 1982, pp.298-305.

[25] Tou J.T. , Gonzalez R.C. , "Pattern Recognition Principles". Addison-Wesley Publishing Company, 1974.

[26] Hilbert E.E, "Cluster Compression Algorithm a Joint Clustering/Data Compression Concept". Jet Propulsion Laboratory Publication 77-43, 143 pages.

[27] Hilbert E.E. , "Joint Pattern Recognition/Data Compression Concept for ERTS Multispectral Data". SPIE vol.66, 'Efficient Transmission of Pictorial Information', August 1975.

[28] Koontz, W.L., Narendra P.M. and Fukunaga K. "A Graph-Theoritic Approach to Non-parametric Cluster Analysis". IEEE Trans. on Computers, Vol. C-25 #9, September 1976, pp. 936-945.

[29] Linde Y. , Buzo A. , Gray R.M. , "An Algorithm for Vector Quantizer Design". IEEE Trans. on Communications, vol.COM-28 no.1 January 1980, pp.84-95.

[30] Gonzalez R.C., Wintz P., "Digital Image Processing". Addison-Wesley Publishing Company. Reading Massachusetts. 1977.

[31] Buzo A. , Gray A.H. Jr. , Gray R.M. , Markel J.D. , "Speech coding Based Upon Vector Quantization". IEEE Trans. on Acoustics, Speech and Signal Processing vol.ASSP-28 no.5 Oct.1980, pp.562-574.

[32] Abut H.,Gray R.M.,Rebolledo G.,"Vector Quantization of Speech and Speech-like Waveforms". IEEE Trans. on ASSP vol.ASSP-30 no.3, June 1982.

[33] Gersho A. , "On the Structure of Vector Quantizers". IEEE Trans. on Information Theory vol. IT-28 no.2 March 1982, pp.157-166.

[34] Habibi A. , and Samulon A.S. , "Bandwidth Compression of Multispectral Data". SPIE vol.66 pp.23-34, August 1975, Efficient Transmission of Pictorial Information.

[35] Lowitz G.E. , "Compression des donnees images par reconnaissance des formes et 'clustering'". Proceedings of an International Conference on Earth Observation from Space and Management of Planetary Resources; Toulouse, March 1978, pp.243-250.

[36] Lowitz G.E. , "Image Data Reduction". Proc. International Conf. on Spacecraft On-board Data Management; Nice, Oct.1978 pp.291-4.

[37] Lowitz G.E. , Vivier J.M. , "Cascaded Clustering of Compressed Multispectral Image Data". European Space Agency CR(P)-1387, Data Rate Reduction - Vol 3. MATRA, France, Jul. 1980.

[38] MacCalla J.R. , Chang M.V. , "Multispectral Data Compression Using Hybrid Cluster Coding". American Institute of Aeronautics and Astronautics Communications Satellite. Systems Conference, 1980 pp.404-7.

[39] Heckbert P. , "Color Image Quantization for Frame Buffer Display". ACM Computer Graphics, volume 16, no. 3. July 1982, pp. 297-307.

[40] Baker R.L., Gray R.M., "Image Compression using Non-adaptive Spatial Vector Quantization". IEEE Conference on Circuits Systems amd Computers, 1983.

[41] Cabrera S., Landau I.D. "codage Differentiel Adaptatif d'images avec Quantification vectorielle". Neuvieme colloque sur le traitement du signal et ses applications. GRETSI conference, Nice, France, 1983, pp. 571-576.

[42] Hunt B.R. , "Nonstationary Statistical Image Models (and their Application to Image Data Compression)". Computer Graphics and Image Processing 12, 1980 pp.173-186.

[43] Narendra P.M. ,and Goldberg M. , "A Non-Parametric Clustering Scheme for Landsat". Pattern Recognition. Vol no.9 1977, pp.207-215.

[44] Warton Stephen W., "A Generalized Histogram Clustering Scheme for Multidimensional Images Data". Pattern Recognition Vol. 16 #2 pp. 193-199, 1983.

[45] Netravali A.N. ,and Limb J.O. , "Picture Coding: A Review". Proceedings of the IEEE, vol.68, no.3,March 1980, pp.366-407.

[46] Gimlett J.I. , "Use of 'Activity' Classes in Adaptive Transform Image Coding". IEEE Transactions on Communications, July 1975, pp.785-6.

[47] Tasto M. , Wintz P.A. , "Image Coding by Adaptive Block Quantization". IEEE Trans. on Communications Technology vol.COM-19 no.6, December 1981, pp.957-972.

[48] Burge R.E. , WU J.K. , "An Adaptive Transform Image Data Compression Scheme Incorporating Pattern Recognition Procedures". IEEE PRIP 1981, pp.230-336.

[49] Habibi A. , Wintz P.A. , "Image Coding by Linear Transformation and Block Quantization". IEEE Transactions on Communication Technology, vol.COM-19, no.1, Feb.1971, pp.50-62.

[50] Wu J.K. , Burge R.E. , "Adaptive Bit Allocation for Image Compression". Computer Graphics and Image Processing, no.19, 1982, pp.392-400.

[51] Knab J.J. , "Effects of Round-off Noise on Hadamard Transformed Imagery". IEEE Trans. on Communications. Vol COM-25, No. 11, pp. 1292-1294, Nov 1977.

[52] Limb J.O. , Rubinstein C.B. , Thompson J.E. , "Digital Coding of Color Video Signals : A Review". IEEE Trans. on Communications vol.COM-25 no.11 Nov.1977, pp.1349-1385.

[53] Pratt W.K. , "Spatial Transform Coding of Color Images". IEEE Trans. on Communication Technology, vol.COM-19 no.6, Dec. 1971, pp. 980-992.