A NEW CLASS OF OPTIMAL UNITARY TRANSFORMS FOR IMAGE PROCESSING

by

PHILIPPE MICHEL CASSEREAU

Ingénieur de l'école supérieure d'electricité (1984)
Gif-sur-Yvette, FRANCE

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS OF THE

DEGREES OF

ELECTRICAL ENGINEER

and

MASTER OF SCIENCE

IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1985

Signature of Author

Department of Electrical Engineering and Computer Science
May 10,1985

Certified by

David H. Staelin
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

A NEW CLASS OF OPTIMAL UNITARY TRANSFORMS FOR IMAGE PROCESSING

by

Philippe Michel Cassereau

Submitted to the Department of Electrical Engineering
and Computer Science, May 1985
in partial fulfillment of the requirements for the degrees of
Master of Science and of Electrical Engineer.

ABSTRACT

Unitary transform image coding systems have been shown to be a successful approach to achieve image data compression. However, block transform image coding systems generate artifacts which degrade low bit-rate coded images. The discrete cosine transform (DCT), for example, generates "blocking effects". Alternatively, the short-space Fourier transform (SSFT) generates "ringing effects". To reduce these artifacts, a new class of unitary transformations, defined as lapped orthogonal transforms (LOT), has been investigated. The basis functions upon which the signal is projected are overlapped by a fixed non-zero number of samples. The transform must remain orthogonal to avoid redundancy of the image representation in the transform domain. An example of a LOT optimized in terms of energy compaction was numerically derived on a digital computer, using an augmented Lagrangian optimization algorithm.

Intraframe zonal transform coding experiments were performed at bit-rates ranging from 0.1 to 0.5 bit per pixel. An hybrid transform/DPCM interframe coding system was computer-simulated. Such a system, including a motion-compensated predictor, was also experimentally tested. Using these two architectures, interframe coding experiments were performed at a constant data-rate of 56 kilobits per second. For both intraframe and interframe encoders, the LOT, by reducing blocking and ringing effects, improved the coded image subjective quality over the DCT and the SSFT.

Thesis Supervisor: David H. Staelin

Title: Professor of Electrical Engineering

To Monique,
Marc, and
Kristen

Common sense, do what it will, cannot avoid being surprised occasionally. The object of science is to spare it this emotion and create mental habits which shall be in such close accord with the habits of the world as to secure that nothing shall be unexpected.

B.R.

A. E. Van Vogt
The World of Null-A

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Professor David H. Staelin for his suggestions, supervision, support and dedication of time.

I would like to extend my thanks to everyone in the Research Group. There are so many who I wish to acknowledge that I am sure I may inadvertantly leave out some names. With that in mind, I would like to express my gratitude:

To Brian Hinman and Jeff Bernstein, who introduced me to image processing and whose valuable suggestions got me started on this research.

To Henrique Malvar, for providing the software necessary to use the video system. Despite the fact he usually calls me "Felipe", I always found his comments useful.

To Dr. Philip Rosenkranz and Mark Colavita, for helping me when the computer system was mad at me (I always wondered why?).

To Vivek Dhawan, for his artistry which appears in some of the figures of this thesis.

To Ashok Popat, for helping me produce the photographs in this thesis.

To Alain and Maria Briancon, for their friendship and support at times of need. Alain provided numerous suggestions and helped me with the computer-generated graphs. Maria edited parts of the text.

My very special thanks to Kristen Kaliski. She gave me love and moral support and helped me keep what little sanity I had left during the last few months. Her dedication to this thesis was probably as high as my own. She did an outstanding job editing and typing the text.

Last but not least, I would like to express my heartfelt gratitude to my parents. Their commitment to the education of their children has always been as strong as their will to see them succeed. Without them, none of this would have been possible.

This work was supported by grants from DARPA and the Advanced Television Research Project.

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

Data communication technology has been marked by extraordinary growth in the past decade. Data communication systems usually involve either data storage or data transmission, and often both. As the technology and performance of digital communication systems improve, the demand for systems able to store, process, and communicate large amounts of complex data, such as images, becomes stronger. To satisfy these needs, various methods relative to image processing have been developed.

In digital image processing, an analog monochrome video signal is sampled. The total number of samples taken from one image determines the resolution of the digital image. The luminance value of the image sample (also called picture element or pixel) is quantized with a fixed number of bits. This number of bits, along with the resolution, characterizes the quality of the digital image [8], [26]. However, images in digital form may represent considerable amounts of binary data. Consider, for example, a monochrome video signal recorded at 30 frames per second, 512 x 512 pixels per frame, and 8 bits per pixel. This signal corresponds to a data rate of about 63 megabits per second.

Although images in digital form can be processed with much more flexibility than images in analog form, the bandwidth requirements of the digital image transmission systems are increased. To minimize the cost of storage and transmission of digital images, memory and data compression techniques must be considered [17], [24].

The basic idea is to use the processing flexibility of images in digital form to implement data compression algorithms. It is important to notice that significant compression cannot be achieved without some distortion of the reconstructed image. Therefore, for any image compression algorithm, three qualities are always considered to measure its efficiency: the data compression ability, the resulting distortion, and the complexity of possible hardware implementation.

Transform image coding is known as one of the most popular and successful methods in image data compression. In traditional image coding systems, such as PCM, predictive, and interpolative coding, the image samples are coded directly in their original representation [8], [26]. In transform image coding systems, the approach is radically different because the coding process is indirect. An energy-preserving, or unitary, transformation is applied to the image, and the resulting transform coefficients are quantized, coded and transmitted [26], [12], [41].

The concept of transform coding was first introduced by Andrews and Pratt in 1968 [2]. They used the Fourier transform as the energy-preserving transformation. Numerous transforms were presented afterwards. The main concerns were to find unitary transforms with concurrent low mean-square error coding performance and reduced computational requirements [1], [14], [16], [28] - [30].

Transform coding systems are successful because the representation of the signal in the transform domain can be coded more efficiently than the one in the spacial domain. Because of the typically high correlation between pixels of natural monochrome images, the signal energy in the transform domain tends to be clustered into a small number of transform coefficients. Thus, considerable data compression can be achieved without significant distortion of the reconstructed coded image by discarding or coarsely quantizing the low-energy coefficients. However, some coding artifacts, typical of transform coding systems, distort images coded at a very low bit-rate (normally less than 0.5 bit per pixel). For example, blocking effects are generated by transforms such as the discrete cosine transform (DCT) [23]. Alternatively, the short-space Fourier transform (SSFT) avoids blocking effects, but generates ringing effects [14].

A new class of unitary transforms which reduces these artifacts is developed in this thesis. Definitions and principles of transform coding systems are reviewed first. The basis functions of a transform are defined, and the important concepts of decorrelation of image samples and energy compaction are described. The coding artifacts generated by the DCT and the SSFT are analyzed. A new type of unitary transform for image coding is then introduced. For a specific example of such a transform, optimized in terms of

energy compaction, an algorithm which yields a numerical model for the new transform is derived. For the experimental part, an intraframe transform coding system (coding of still images) is described. Both adaptive and non-adaptive zonal coding schemes are presented. Hybrid transform/DPCM and motion-compensated interframe coding system (coding of sequences of images) architectures are computer-simulated. The new transform and the DCT are tested in both interframe and intraframe coding experiments; the SSFT is tested in intraframe coding experiments alone. Finally, the performances of the various transforms and systems are evaluated and compared.

CHAPTER TWO

DEFINITION AND EXAMPLES OF TRANSFORM IMAGE CODING SYSTEMS

2.1. PRINCIPLES OF TRANSFORM CODING SYSTEMS

2.1.1. Definition

An image in digital form can be represented as an array or matrix F of size R x R, where R is the resolution of the image. Each image sample is an element of this matrix. In addition, each sample is quantized and assigned a fixed-length code word [8], [26].

Suppose a sequence of digital images has to be transmitted over a communication channel. One possibility is to scan each image following a predetermined pattern and to send the code word of each pixel through the channel. However, for real-time applications, this technique involves a data rate exceeding the capacity of the large majority of channels currently used [17].

Alternatively, in a transform coding system, an image is represented in a space called the transform domain. The domain which has the original representation of the image is called the space domain, in contrast to that of the transform. The image samples in a transform coding system are quantized and coded in the transform domain, rather than in the space domain [2], [8], [17], [24], [26] - [30], [41].

The new representation of the image is obtained by applying an energy-preserving linear transformation to the array F [16]. If the transformation is invertible, the representation in the transform domain is non-redundant and thus is an array F' of the same size as F. Therefore, F' is obtained by applying a unitary transform T to F:

$$F'(k,l) = \sum_{i=1}^{R} \sum_{j=1}^{R} F(i,j)T(i,j,k,l) \quad k,l = 1,\ldots,R \tag{2.1}$$

The transform kernel T is usually separable. In actuality, T is expressed as the product of two operators: one for the rows, and one for the columns:

$$T(i,j,k,l) = T_c(i,k)T_r(j,l) \tag{2.2}$$

Following this, F' is expressed as the product of three matrices:

$$F' = T_c \; F \; T_r \tag{2.3}$$

For reasons of symmetry, transforms used in image coding usually have identical row and column operators:

$$T_r^{\,t} = T_c = T \tag{2.4}$$

Accordingly, the two-dimensional computation of F' reduces to:

$$F' = T\,F\,T^t \qquad (2.5)$$

or:

$$F' = [T(T\,F)^t]^t \qquad (2.6)$$

The R x R unitary matrix T ($T^t = T^{-1}$) is denoted as the one-dimensional transform kernel. Consequently, the two-dimensional transform F' of F can be computed by first taking a one-dimensional transform along each row, and by then taking another one-dimensional transform along each resulting column.

Once F' has been derived, each transform coefficient, or element of F', is quantized and assigned a code word. These coefficients are scanned and the corresponding code words are transmitted. At the receiver, the transform coefficients are decoded. The inverse transform is then taken to acquire the reconstructed image [41].

2.1.2. The basis functions

Applying a unitary transform T to the image F can be interpreted as projecting F onto a set of two-dimensional basis functions. The unitarity condition implies that the basis functions are orthonormal. Hence, F' is the set of coordinates of F in the new basis. Since only separable transforms are considered, the two-dimensional basis functions are derived from the one-dimensional basis functions. Computing the two-dimensional transform of F can be done by taking one-dimensional transforms along rows and columns. Consider a vector $\underline{f}$ of size R which could be a line of pixels from the original image F. The one-dimensional transform of $\underline{f}$ is a vector $\underline{f}'$ of size R. This is derived as such:

$$\underline{f}' = T\,\underline{f} \tag{2.7}$$

where T is the R x R unitary matrix defined by equation (2.4).

In most transform coding systems the digitized image is divided into subimages called blocks [8], [12], [17], [41]. The process applied to each data block is identical. Suppose F is divided into blocks of size N by N pixels. Knowing that the resolution of the original image is equal to R, the total number of data blocks is K^2, where K is equal to R/N.

The transform vector $\underline{f}'$ is obtained by projecting $\underline{f}$ upon each of the R one-dimensional basis functions, which are the rows of the matrix T. To express the block structure of the transform process, $\underline{f}$ and $\underline{f}'$ are divided into K non-overlapping segments of N samples. Even in the one-dimensional case, each segment is referred to as a "block" to correspond to the two-dimensional case. The basis functions can be written as K x N vectors of size R $\underline{b}_{i+kN}$ for $i = 1,\ldots,N$ and $k = 0,\ldots,K-1$. Using this notation, the projection of $\underline{f}$ upon the N basis functions $\underline{b}_{i+kN}$, $i = 1,\ldots,N$, (k is fixed) yields the N coefficients of the (k+1)-th data block of $\underline{f}'$. The block structure of the process is determined by the fact that for a given index i, $i = 1,\ldots,N$, the basis function

$\underline{b}_{i+k'N}$ is equal to the basis function $\underline{b}_{i+kN}$ linearly shifted by (k'-k)N samples (assuming that the basis functions are extended to infinite length sequences). This N-sample shift invariance is illustrated in Figure 1, for N = 10.

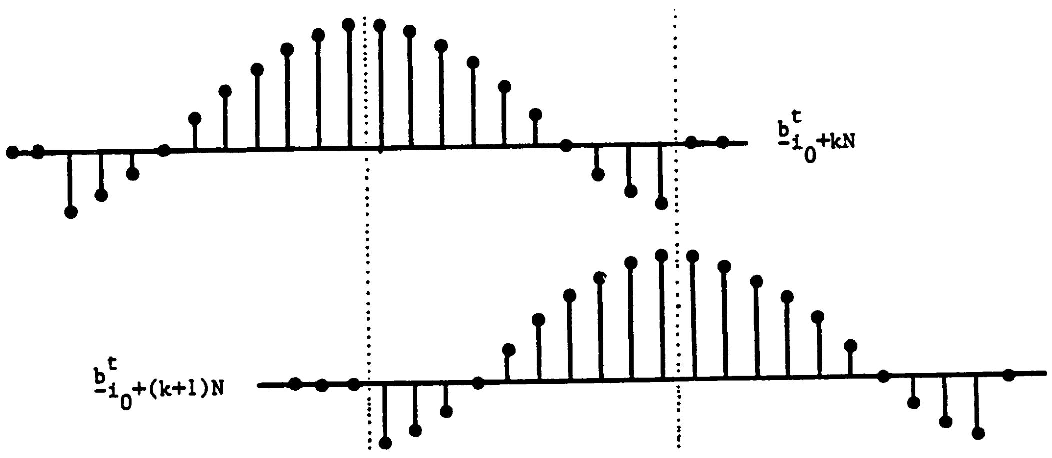


Figure 1 - N-sample shift invariance of the basis functions for a given index i_0.

For transforms traditionally used in image coding, only the N samples of $\underline{b}_{i+kN}$, which correspond to the locations of the N image samples of the (k+1)-th block of $\underline{f}$, have non-zero value. Consequently, the processing of one data block is completely independent from the processing of another. On the other hand, the non-zero coefficients of $\underline{b}_{i+kN}$ need not be limited to the locations of the (k+1)-th block samples. The overlap L of the transform T is defined as the number of non-zero coefficients of $\underline{b}_{i+kN}$ which correspond to image samples extending outside the (k+1)-th block. For reasons of symmetry, the total number of non-zero coefficients of the $\underline{b}_{i+kN}$ basis functions is N + 2L. If L is not zero, the transform representation of a block includes not only the image data of that block, but also the image data from neighboring blocks. In the example given in Figure 1, L is equal to N/2. Notice that the basis functions corresponding to adjacent blocks actually overlap by 2L samples.

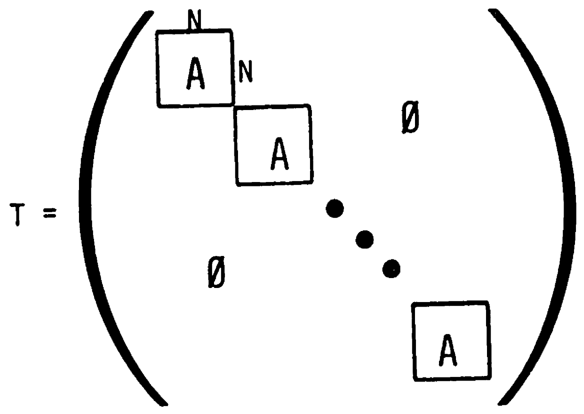
Define for a given i the vector $\underline{a}_i$ as the representation of the non-zero coefficients of the $\underline{b}_{i+kN}$ basis functions for $k = 0,\dots,K-1$. The length of the $\underline{a}_i$ vectors, $i = 1,\dots,N$, is N + 2L. The $\underline{b}_{i+kN}$ basis is obtained by shifting $\underline{a}_i$ by KN samples and by setting the remaining samples to zero. The $\underline{b}_{i+kN}$ basis functions represent the rows of the matrix T:

$$T = [\underline{b}_{i+kN}{}^t] \qquad i = 1,\dots,N;\ k = 0,\dots,K-1 \qquad (2.8)$$

Similarly, the one-dimensional block transform kernel A is described as:

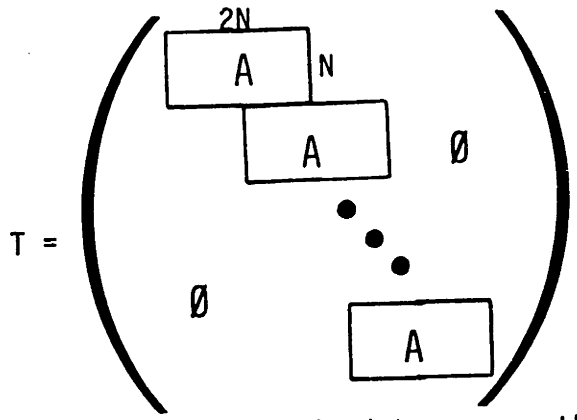
$$A = [\underline{a}_i{}^t] \qquad i = 1,\dots,N \qquad (2.9)$$

The operator A is a N x (N+2L) matrix. The transform kernel T can be expressed in terms of the block transform kernel A. For example, when L is equal to zero, T has the following block structure:



(2.10)

When L is equal to N/2, T takes the following form:



(2.11)

The matrix A is constrained to ensure that the matrix T is unitary. When there is no overlap, the block structure of T, as described by equation (2.10), produces the condition that A itself must be unitary. In other words, taking the transform of the image F is equivalent to taking a transform of size N x N of each block of F. Such transforms have been widely used and remain the most popular ones used in transform image coding.

2.1.3. Decorrelation of the image samples

In this section and in the following one, it is assumed that the transform has no overlap (L=0). Consequently, a unitary transform of each block of size N x N is taken (A is the one-dimensional block transform kernel).

Stationary assumptions are not valid for typical images because the image statistics may vary widely between different regions. However, when an image is divided into blocks of N x N pixels (usually N is equal to 8 or 16), each block of the image can be viewed as a sample of a discrete two-dimensional stationary random process. Images are commonly described as first order Markov processes [8], [26]. Consider a line $\underline{x}$ of N pixels. The covariance matrix Λ_x of the vector $\underline{x}$ is defined as:

$$\Lambda_x = E[(\underline{x}-E(\underline{x}))(\underline{x}-E(\underline{x}))^t] \tag{2.12}$$

The stationary assumption implies that the variance of each coefficient of $\underline{x}$ is constant, as such:

$$\underline{x} = [x_i] \qquad \sigma^2(x_i) = \sigma^2 \qquad i = 1,\dots,N \tag{2.13}$$

The Markov model yields a covariance matrix of the form:

$$\Lambda_x = \sigma^2 R \tag{2.14}$$

where R is the N x N matrix defined by:

$$R(i,j) = \rho^{|i-j|} \qquad i,j = 1,\dots,N \tag{2.15}$$

and ρ is the correlation factor between adjacent pixels.

For typical images, each pixel is strongly correlated with its neighbors. Sample values used for the correlation factor ρ range from 0.9 to 1 [26]. Due to this high correlation, the representation of the image information content in the space domain is redundant. Moreover, since the image samples have the same variance, a fixed-length code word encryption scheme must be used in the space domain. The interest in using a unitary transform for image data compression has evolved because the transform can be chosen to decorrelate optimally the image samples. Consequently, the representation of the image information content in the transform domain is less redundant. The coding efficiency can thus be improved in the transform domain.

Suppose the $\underline{a}_i$ vectors, as defined by equations (2.9) and (2.10), are chosen to be the eigenvectors of the covariance matrix Λ_x associated respectively with the eigenvalues λ_i. In this instance, the following holds true:

$$\Lambda_x \underline{a}_i = \lambda_i \underline{a}_i \tag{2.16}$$

Define $\underline{x}'$, the transform vector of $\underline{x}$ as:

$$\underline{x}' = A\,\underline{x} \qquad \underline{x}' = [\underline{a}_i^t \underline{x}] \qquad i = 1,\ldots,N \tag{2.17}$$

The covariance matrix of $\underline{x}'$ is given by:

$$\Lambda_{x'} = A\,\Lambda_x\,A^t \tag{2.18}$$

However, A is the unitary matrix that diagonalizes Λ_x (suppose the

$\underline{a}_i$ vectors are normalized to 1). Namely:

$$\Lambda_{x'} = \begin{pmatrix} \lambda_1 & & \emptyset \\ & \ddots & \\ \emptyset & & \lambda_N \end{pmatrix} \tag{2.19}$$

Consequently, in this optimal case, the components of the $\underline{x}'$ vector are uncorrelated with each other.

This optimal transform is known as the Hotelling or the Karhunen-Loeve transform (KLT) [8], [26]. However, such a transform cannot be used easily in practice because, for a given image, the actual covariance matrix Λ_x is not known exactly. Alternative models for Λ_x, such as the one defined by equations (2.14) and (2.15), are used. Even though perfect decorrelation cannot be achieved in practice, all practical transforms are chosen for maximal performance towards decorrelation.

2.1.4. Energy compaction

In section 2.1.3, the reasons why A should be chosen to decorrelate the image samples as much as possible were given. Another interpretation can be made to provide some insight to the problem of using unitary transforms for image data compression.

The covariance matrix of the $\underline{x}'$ vector, $\Lambda_{x'}$, is given by equation (2.18). More particularly, the variance (or expected energy) of the i-th component of $\underline{x}'$, x'_i, which is the projection of $\underline{x}$ onto the i-th basis function $\underline{a}_i$, is given by the following equation:

$$\sigma^2(x'_i) = \underline{a}_i^t \Lambda_x \underline{a}_i \qquad (2.20)$$

The concept of energy compaction refers to the idea that the transform should be chosen to pack a maximum of information into a minimum of transform samples where energy and information are presumed to be proportional [17]. If the image information content is concentrated in the fewest possible coefficients, these transform coefficients then have different variances. Variable-length code word encryption schemes would be used for transmission. If the transform is optimized in terms of energy compaction, a large number of the transform coefficients have small variances. In this case, they contain little information about the image, and therefore are quantized coarsely, allowing for large data compression.

A transform optimized in terms of energy compaction is such that $\sigma^2(x'_i)$ is maximized for any i, $i = 1,\dots,N$. Normalizing $\sigma^2(x'_i)$ by the squared norm of $\underline{a}_i$ yields the Rayleigh's quotient of Λ_x. According to Rayleigh's principle, this quotient is maximized by the eigenvector of Λ_x corresponding to the largest eigenvalue. Therefore, the solution to this problem is also the KLT. The concepts of decorrelation and energy compaction are equivalent.

Using the Markov model for Λ_x, the variances $\sigma^2(x'_i)$, $i = 1,\dots,N$, can be normalized by σ^2 and the squared norm of $\underline{a}_i$.

Thus, the energy compaction of the $\underline{a}_i$ basis function can be defined as the Rayleigh's quotient of the matrix R defined by equation (2.15):

$$E_i = \frac{\underline{a}_i^t \, R \, \underline{a}_i}{\underline{a}_i^t \, \underline{a}_i} \qquad (2.21)$$

A transform is optimal in terms of energy compaction if E_i is maximized for any i, $i = 1,\ldots,N$. The vectors $\underline{a}_i$, $i = 1,\ldots,N$, normally are arranged by decreasing order of the respective ratios E_i.

2.2. THE DISCRETE COSINE TRANSFORM

2.2.1. Definition

The transform optimal both in terms of energy compaction and decorrelation of the image samples is the KLT as previously established. When the image is modelled as a Markov process, the $\underline{a}_i$ basis functions of the KLT are computable. Unfortunately, no fast algorithm is available to compute the KLT of an image. The discrete cosine transform (DCT) produces results close to those of the KLT and is known to have computationally fast algorithms [1], [5], [33], [39].

The DCT is a transform with no overlap (L=0). Thus, K^2 two-dimensional DCT's of size N x N of each block are taken to compute the DCT of an image of resolution R (R=KN). The block structure of the one-dimensional transform kernel T is given in equation (2.10). The $\underline{a}_i$ vectors, which are defined by the next equation, are of size N:

$$\underline{a}_i^t = [a_i(n)] \qquad n = 0,\ldots,N-1$$

$$a_i(n) = \frac{2C(i)}{N} \cos\left[\frac{(2n+1)\, i\, \pi}{2N}\right] \quad i,n = 0,\ldots,N-1 \qquad (2.22)$$

with:

$$C(i) = \begin{cases} 1 & \text{for } i \neq 0 \\ 1/\sqrt{2} & \text{for } i = 0 \end{cases}$$

Note that the $\underline{a}_i$ basis functions, defined by (2.22), are not normalized to one. Thus, the basis functions used to calculate the inverse DCT are the $\underline{a}_i$ vectors scaled by N/2. The DCT, due to both its performance in energy compaction and its fast algorithms, has been widely used in transform image coding systems.

2.2.2. Coding artifacts generated by the DCT

A block transform coding system using the DCT takes the DCT of each block, independently quantizes each transform sample, and transmits the resulting coded coefficients over the communication channel. Using these coefficients, the receiver reconstructs the image by taking the inverse DCT of each block. One consequence of this procedure is that the quantization noise is uncorrelated mainly from block to block. The resulting effect is that the blocks may be visible, especially for images coded at low data rates. In such a system, the original image is divided into subimages. The subimages are transform coded separately as independent images. Then, the complete image is reconstructed by the juxtaposition of each block. This procedure however, may generate mismatches between adjacent blocks. Experiments performed with the DCT have shown that these so-called "blocking effects" may damage significantly the subjective quality of the reconstructed image [3], [23].

Actually, blocking effects are not generated solely by the DCT. Indeed, these artifacts appear with any coarsely quantized transforms having no overlap (L=0). Blocking effects are linked directly to the typical block structure of the transform kernel T, given in equation (2.10). This problem can also be explained by the discontinuities of the $\underline{b}_{0+kN}$ basis functions. Consider the first basis function $\underline{a}_0$ of the DCT. This basis captures the DC component of the signal. All the components of the $\underline{a}_0$ vector are constant, being equal to $\sqrt{2}/N$. Consequently, the corresponding $\underline{b}_{0+kN}$ basis functions present discontinuities at the edges of the blocks.

Until now, three methods have been proposed to reduce blocking effects due to quantization noise. The first method, known as both two-component source coding and pinned transform coding, separates the image into a stationary field coded in the space domain, and a non-stationary field, coded in the transform domain [21], [22], [35], [42]. Usually, the stationary field is a low-pass version of the original image. Since blocking effects are mainly a result of quantization errors which affect the low-frequency components of the image, the two-component source coding procedure reduces their visibility. A second method is to apply a low-pass filter over the boundary regions between the blocks [31]. This also reduces the visibility of the blocking effects. However, when an edge matches into a block boundary, a blurring of the edge occurs. The last method is based upon an overlapping of the blocks by one pixel before transform coding [31]. The average intensity over the pixels from overlapping blocks is computed to reconstruct more accurately the boundary regions between blocks. The main disadvantage of this method is the redundancy of the transform samples. Since the image representation in the transform domain is redundant, a loss of data compression ability results. Thus, none of the methods described here sufficiently solve the problem.

2.3. THE SHORT-SPACE FOURIER TRANSFORM

2.3.1. Definition

Hinman and Bernstein introduced the Short-Space Fourier Transform (SSFT) as an alternative to the DCT to avoid blocking effects [3], [14]. The SSFT is a multidimensional extension of the short-time Fourier transform which was developed for one-dimensional infinite-length signals such as speech.

The image, which is a finite length signal, is first reflectively extended periodically to get an infinite-length signal. An infinite-length window is applied to this signal. Finally, the SSFT is obtained by taking a two-dimensional Fourier transform of the resulting windowed signal. The applied window is located at the center of each block and extends over the entire signal. An interesting feature of the SSFT is that it is computed using all the image data, but still provides local spectral characteristics. In that manner, the basis functions of the SSFT completely overlap. The overlap L is equal to infinity.

2.3.2. Coding artifacts generated by the SSFT

The SSFT has been used for image transform coding. As expected because the value of L is equal to infinity, blocking effects are totally avoided when the SSFT is used. Since the window covers the entire signal, quantization noise generated in one part of the image spreads everywhere. Around sharp edges, the high-frequency components of the signal are large. Quantization of the transform coefficients, especially at low bit-rate, results in a low-pass effect on the image. Using the SSFT, this noise is spread because of the infinite length of the basis functions. This generates "ringing effects," especially noticeable around the edges. In coding of single images (intraframe coding) the SSFT has been shown to provide better results than the DCT, mainly because of the elimination of blocking effects. However, for coding of sequences of images involving motion (interframe coding), the SSFT fails to provide better results than the DCT. Actually, the blocking effects of the DCT were preferred to the ringing effects of the SSFT in interframe coding experiments [3].

CHAPTER THREE

DEFINITION OF A LAPPED ORTHOGONAL TRANSFORM

In the previous chapter, two transforms used in image coding were presented. When the DCT ($L=0$) and the SSFT ($L=\infty$) are used in image coding, they generate different types of artifacts: blocking effects by the DCT and ringing effects by the SSFT. Image coding experiments illustrating these effects are presented in chapter 5. This thesis introduces a new class of unitary transforms with non-zero valued finite overlap L to reduce simultaneously the blocking effects of the DCT and the ringing effects of the SSFT.

3.1. DEFINITION AND PROPERTIES

In this research only real separable unitary transforms have been considered. Thus, the computation of the transform F' of the array F of size R x R ($R=KN$) reduces to:

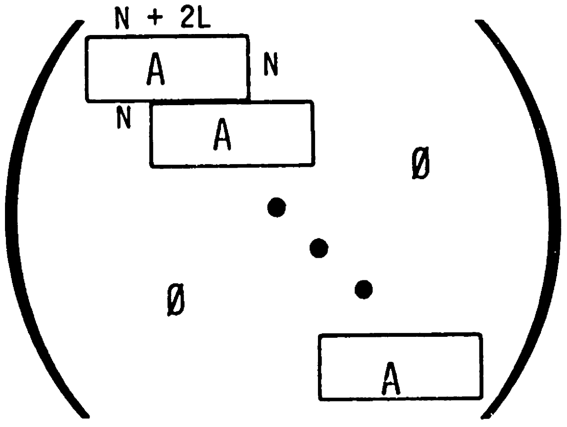
$$F' = T\ F\ T^t \tag{3.1}$$

where T is the one-dimensional transform kernel. Since the transform is unitary, the matrix T is a real unitary matrix. That is:

$$T\,T^t = T^t\,T = I \tag{3.2}$$

A lapped orthogonal transform (LOT) is a real transform satisfying both properties of separability and unitarity described by equations (3.1) and (3.2). The one-dimensional transform kernel T is constrained, moreover, to having the following standard block structure introduced in section 2.1.2:

T =



(3.3)

The matrix A is of size N x (N+2L). The overlap L, defined in section 2.1.2, is a non-zero valued integer. The block size is the integer N. Therefore, the block size N and the overlap L completely describe the matrix T and hence a LOT.

The choice of the block size N has been discussed in section 2.1.3. It must be chosen so that the assumption about the stationarity of the "block" random process is valid. Typical values for N are 8 and 16.

The structure of T given in (3.3) should reduce "mismatch effects" between blocks since the transform process is not independent from block to block. However, the matrix T still has a block structure which may produce discontinuities. If the value of L is not chosen carefully, multiple boundary effects may result. Consequently, the overlap L must be chosen so that these eventual boundary effects merge into a square grid of size N. Given the structure of T, this clearly implies that L must be selected so that N+2L is a multiple of N. That is, L must be a multiple of N/2 (assuming N is even):

$$L = k\ N/2 \tag{3.4}$$

where k is a non-zero positive integer.

Given the block size N and the overlap L, the matrix A must be derived so that the unitarity condition on T is satisfied. Thus, the transform process yields a non-redundant representation of the digitized image.

This definition of a lapped orthogonal transform defines a class of unitary transforms. Different values of L may be considered. The unitarity condition on T specifies the constraints on A. On the other hand, A can be optimized in terms of energy compaction. An additional constraint may be a fast computational algorithm for the calculation of the transform.

3.2 EXAMPLE OF A LOT

An example of a lapped orthogonal transform is defined in this section. The problem of finding this transform, referred to in the following sections and chapters as the LOT, is formulated as a non-linear optimization problem. The optimization technique used to solve this problem will be developed in chapter 4. The applications of the LOT to image coding will be presented in chapters 5 and 6.

3.2.1. Definition

In order to define a LOT, the value of the overlap L and how the one-dimensional block transform kernel A is to be optimized need specification. The overlap L is chosen here to be equal to N/2. The block transform kernel A is optimized here in terms of energy compaction, according to the definition given in section 2.1.4. In this manner, the LOT to be developed in chapter 4 will be equivalent to the KLT when L is equal to 0. Since the $\underline{a}_i$ vectors alone determine the $\underline{b}_{i+kn}$ basis functions defined in section 2.1.2, they are also referred to as the LOT basis functions. Since L is equal to N/2, the basis functions are 2N points long and the block transform kernel A is a N x 2N matrix. All the fundamental properties of the LOT are summarized in Table 1.

3.2.2. Constraints on the basis functions

The constraints on the $\underline{a}_i$ basis functions are determined by the unitarity condition on T. The basis functions $\underline{a}_i$ can be written:

Table 1 - Properties of the LOT

* Separability of the 2-D transform kernel:

$$F' = T\ F\ T^t$$

F : image representation in the space domain

F': image representation in the transform domain

* 1-D transform kernel:

T is a real and unitary matrix:

$$T\ T^t = T^t\ T = I$$

* Block size: N

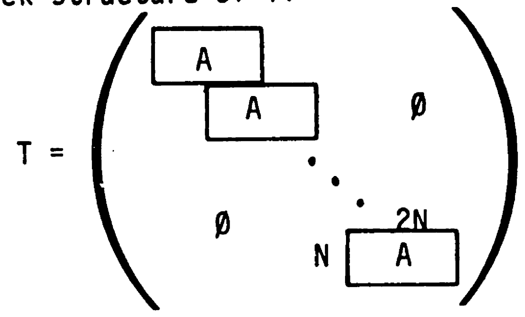
* Overlap: $L = N/2$

* 1-D block transform kernel:

$$A = [\underline{a}_i^t] \qquad i = 1,\dots,N$$

A is a N x 2N matrix

* Block structure of T:



* Optimization: the LOT is optimized in terms of energy compaction:

For any $i = 1,\dots,N$; $\underline{a}_i$ maximizes E_i, where:

$$E_i = \frac{\underline{a}_i^t\ R\ \underline{a}_i}{\underline{a}_i^t\ \underline{a}_i}$$

with: $R = [R(k,l)]$, 2N x 2N matrix defined by:

$$R(k,l) = \rho^{|k-l|} \qquad k,l = 1,\dots,2N$$

where ρ is the correlation factor between adjacent pixels.

$$\underline{a}_i^t = (\underline{x}_i^t, \underline{y}_i^t) \qquad i = 1,\dots,N \tag{3.5}$$

where $\underline{x}_i$ and $\underline{y}_i$ are two vectors of size N respectively representing the first and the last N elements of the basis function $\underline{a}_i$. In that manner, the N x 2N matrix A is composed of two block matrices X and Y of size N x N:

$$A = (X \mid Y) \tag{3.6}$$

The one-dimensional transform kernel T can then be rewritten:

$$T = \begin{pmatrix} \ddots & & & & \\ & \boxed{X} & \boxed{Y} & & \emptyset \\ & & \boxed{X} & \boxed{Y} & \\ \emptyset & & & \boxed{X} & \boxed{Y} \\ & & & & \ddots \end{pmatrix} \tag{3.7}$$

The unitarity of T can be expressed in terms of the rows of T, namely the $\underline{b}_{i+kN}$, i = 1,...,N, k = 0,...,K-1, basis functions (cf section 2.1.2):

$$\underline{b}_{i+kN}^t \, \underline{b}_{j+lN} = \delta(i-j, k-l) \tag{3.8}$$

for i,j = 1,...,N; k,l = 0,...,K-1

* If $|k-l| > 2$, the constraint (3.8) is always satisfied because of the block structure of T.

* If $|k-l| = 1$, consider the two cases:

  . $k = l - 1$, then the constraint (3.8) implies:

$$\underline{y}_i^t \underline{x}_j = 0 \qquad i,j = 1,\ldots,N \tag{3.9}$$

  . $k = l + 1$, then the constraint (3.8) implies:

$$\underline{y}_j^t \underline{x}_i = 0 \qquad i,j = 1,\ldots,N \tag{3.10}$$

* If $k = l$, then the constraint (3.8) reduces to:

$$\underline{a}_i^t \underline{a}_j = \delta(i-j) \tag{3.11}$$

or:

$$\underline{x}_i^t \underline{x}_j + \underline{y}_i^t \underline{y}_j = \delta(i-j) \tag{3.12}$$

Considering only the orthogonality constraints (the basis functions can always be normalized later to one), the equations (3.9), (3.10), and (3.12) define the following set of constraints on the $\underline{a}_i$ basis functions:

$$\underline{x}_i^t \underline{y}_i = 0 \qquad i = 1,\ldots,N \tag{3.13}$$

$$\left.\begin{array}{ll} \underline{x}_i^t \underline{x}_j + \underline{y}_i^t \underline{y}_j = 0 & (3.14a) \\ \underline{x}_i^t \underline{y}_j = 0 & (3.14b) \\ \underline{y}_i^t \underline{x}_j = 0 & (3.14c) \end{array}\right\} \quad \begin{cases} i,j = 1,\ldots,N \\ i \neq j \end{cases}$$

3.2.3. Symmetry properties of the basis functions

The even/odd symmetry properties of the $\underline{a}_i$ basis functions are introduced in this section. Other transforms, such as the

KLT or the DCT, comparable to the LOT for L equal to zero, present an even/odd symmetry property of their basis functions with respect to the center of each block. This symmetry appears to be correct intuitively since the whole block transform process is symmetric at the center of each block. For the same reason, similar properties can be expected and assumed for the LOT. However, a more powerful justification for the symmetry properties can be given. Assuming the $\underline{a}_i$ basis functions satisfy these symmetry properties, the problem defined by equations (3.13), (3.14a, b, c), admits a non-empty set of feasible solutions.

The symmetry properties of the basis functions of the LOT can be expressed in the following manner:

$$\underline{y}_i = (-1)^{i+1} H \underline{x}_i \qquad i = 1,\dots,N \tag{3.15}$$

where $\underline{x}_i$ and $\underline{y}_i$ are the vector components of the basis $\underline{a}_i$ defined by equation (3.5) and H is the following N x N unitary matrix:

$$H = \begin{pmatrix} & & 1 \\ & \emptyset \; \diagup & \\ 1 & & \emptyset \end{pmatrix} \tag{3.16}$$

Note that $H^t = H$ and $H^2 = I$.

For $i = 1,3,\dots,N/2-1$, $\underline{y}_i$ is equal to $H \underline{x}_i$. Thus, the basis $\underline{a}_i$ is symmetric. For $i = 2,4,\dots,N/2$, $\underline{y}_i$ is equal to $-H \underline{x}_i$. The basis $\underline{a}_i$ is therefore antisymmetric. Using the symmetry properties from equation (3.15) the constraints on the basis functions can be revised. The constraints (3.14b) and (3.14c) become identical, and combined with the constraint (3.13), yield the following equation:

$$\underline{x}_i^t H \underline{x}_j = 0 \qquad i,j = 1,\ldots,N \qquad (3.17)$$

By replacing $\underline{y}_i$ and $\underline{y}_j$ with their expression in terms of $\underline{x}_i$ and $\underline{x}_j$ respectively, the constraint (3.14a) becomes:

$$(1+(-1)^{i+j})\ \underline{x}_i^t \underline{x}_j = 0 \qquad i,j = 1,\ldots,N;\ i \neq j \qquad (3.18)$$

This constraint is obsolete if $\underline{a}_i$ and $\underline{a}_j$ have different types of symmetry (i+j is an odd number). If $\underline{a}_i$ and $\underline{a}_j$ are either both symmetric or antisymmetric, then $\underline{x}_i^t \underline{x}_j$ has to be equal to zero (i+j is an even number).

Define $\{\underline{s}_m\}$, $m = 1,\ldots,N/2$, as the set of the N/2 $\underline{x}_i$ vectors corresponding to the symmetric $\underline{a}_i$ basis functions. Define $\{\bar{\underline{s}}_m\}$, $m = 1,\ldots,N/2$, as the set of the N/2 $\underline{x}_i$ vectors corresponding to the antisymmetric $\underline{a}_i$ basis functions. Denote S, $\bar{S}$, S_H, $\bar{S}_H$ as the subspaces of $\mathbb{R}^N$ spanned by the sets $\{\underline{s}_m\}$, $\{\bar{\underline{s}}_m\}$, $\{H\underline{s}_m\}$, $\{H\bar{\underline{s}}_m\}$, $m = 1,\ldots,N/2$, respectively. The constraint (3.17) implies that the subspaces S and S_H, S and $\bar{S}_H$, $\bar{S}$ and S_H, $\bar{S}$ and $\bar{S}_H$ must be orthogonal:

$$S \perp S_H;\ S \perp \bar{S}_H;\ \bar{S} \perp S_H;\ \bar{S} \perp \bar{S}_H \qquad (3.19)$$

The constraint (3.18) dictates that both $\{\underline{s}_m\}$ and $\{\bar{\underline{s}}_m\}$, $m = 1,\ldots,N/2$, must be orthogonal sets. Since H is a unitary matrix, $\{H\underline{s}_m\}$ and $\{H\bar{\underline{s}}_m\}$, $m = 1,\ldots,N/2$, are also orthogonal sets. Therefore, the subspaces S, $\bar{S}$, S_H, and $\bar{S}_H$ all must have a dimension equal to N/2. Combining these results with the orthogonality constraints given in (3.19) yields the following necessary condition:

$$S = \bar{S} \tag{3.20}$$

Thus S_H is also equal to $\bar{S}_H$.

It is interesting to notice that one consequence of (3.20) is that the set $\{\bar{\underline{s}}_m\}$ (respectively $\{H\bar{\underline{s}}_m\}$) can be obtained from the set $\{\underline{s}_m\}$ (respectively $\{H\underline{s}_m\}$) by applying a unitary transform. Finally, all the constraints given in (3.19) reduce to the single constraint:

$$S \perp S_H \tag{3.21}$$

Since $\{\underline{s}_m\}$ and $\{H\underline{s}_m\}$ are orthogonal sets, the combination of these two sets form an orthogonal basis of IR^N if the condition (3.21) is satisfied. Therefore, finding a feasible solution for the problem of the LOT is reduced to finding an orthogonal basis of IR^N such that N/2 of the basis vectors is obtained by applying the operator H to the remaining N/2 basis vectors. A simple example can be given to prove the feasibility of the problem.

Consider the following orthogonal basis of $\mathbb{R}^N$, given in matrix notation:

$$B = \left(\begin{array}{c|c} C & \emptyset \\ \hline \emptyset & H\,C \end{array} \right) \tag{3.22}$$

where C is a N/2 x N/2 orthogonal matrix. Clearly, B defines an orthogonal basis whose last N/2 basis bectors are obtained by applying H to the first N/2 basis vectors. Define the N x N/2 matrix ($\frac{C}{0}$) as equal to the set $\{\underline{s}_m\}$, m = 1,...,N/2. The condition (3.21) is satisfied. Moreover, if the set $\{\bar{\underline{s}}_m\}$ is determined by applying a unitary transform to $\{\underline{s}_m\}$, then all the remaining constraints are satisfied. This yields a feasible X, A, and consequently, a unitary one-dimensional transform kernel T.

Using the symmetry property defined by equation (3.15), the LOT is fully specified by the matrix X of reduced size N x N. All the constraints of the problem have been expressed in terms of the $\underline{x}_i$ vectors (cf (3.17) and (3.18)).

The optimization criterion as a function of the $\underline{x}_i$ vector can be expressed similarly. The energy compaction of the i-th basis function $\underline{a}_i$ becomes:

$$E_i = \frac{\underline{x}_i^{\,t}\, R_i'\, \underline{x}_i}{\underline{x}_i^{\,t}\, \underline{x}_i} \tag{3.23}$$

where R'_i is the following N x N matrix:

$$R'_i = [R'_i(k,l)]$$

with:

$$R'_i(k,l) = \rho^{|k-l|} + (-1)^{i+1}\rho^{2N+1-k-l} \qquad (3.24)$$

$$k,l = 1,\dots,N$$

where ρ is the correlation factor between adjacent pixels. The detailed derivations of this result are given in Appendix A.

A complete formulation of the optimization problem is given in Table 2 .

3.3. COMPUTATION OF THE TRANSFORM AND THE INVERSE TRANSFORM

In consequence of the separability property, the computation of the two-dimensional forward and backward transforms reduces to the computation of a series of one-dimensional forward and backward transforms. Namely, one-dimensional transforms (or inverse transforms) of each column of the original array are taken. The resulting array is then transposed and the process is repeated. This property has been mathematically described in equation (2.6).

In this section, the computations of the one-dimensional LOT and the inverse LOT are described. The extension to the two-dimensional case is straightforward.

Table 2 - Properties of the 1-D block transform kernel A

* $A = [\underline{a}_i^t]$ $\qquad i = 1,\dots,N$

* $\underline{a}_i^t = (\underline{x}_i^t, \underline{y}_i^t)$ $\qquad i = 1,\dots,N$

* $\underline{y}_i = (-1)^{i+1} H \, \underline{x}_i$ $\qquad i = 1,\dots,N$

with:

$$H = \begin{pmatrix} 0 & & 1 \\ & \iddots & \\ 1 & & 0 \end{pmatrix}$$

* Constraints:

. $\underline{x}_i^t \, H \, \underline{x}_j = 0$ $\qquad i,j = 1,\dots,N$

. $(1 + (-1)^{i+j}) \, \underline{x}_i^t \underline{x}_j = 0$ $\qquad i,j = 1,\dots,N;\ i \neq j$

* Optimization:

$\max\limits_{\underline{x}_i} E_i$ $\qquad i = 1,\dots,N$

with:

. $E_i = \dfrac{\underline{x}_i^t \, R'_i \, \underline{x}_i}{\underline{x}_i^t \underline{x}_i}$

. $R'_i = [R'_i(k,l)]$

. $R'_i(k,l) = \rho^{|k-l|} + (-1)^{i+1} \rho^{2N+1-k-l}$

$k,l = 1,\dots,N$

3.3.1. Computation of the LOT

Consider $\underline{f}$ a column of pixels extracted from the original image F of resolution R. The vector $\underline{f}$ of size R is divided into K blocks of N pixels, which can be written as K vectors of size N: $\underline{b}_k$, k = 1,...,K. Because of the overlap of N/2 samples, any block $\underline{b}_k$ is reconstructed from the transform blocks $\underline{b}'_{k-1}$, $\underline{b}'_k$ and $\underline{b}'_{k+1}$. Details on this will follow in section 3.3.2. However, it is clear that the exact reconstruction of the boundary blocks $\underline{b}_1$ and $\underline{b}_K$ requires additional boundary data. Consequently, the original signal $\underline{f}$, of size R = KN, is extended to get the signal $\bar{\underline{f}}$, of size R + 2(N+L) = (K+3)N, by reflecting N+L = 3N/2 samples at each boundary of the signal.

Divide each block $\underline{b}_k$ into two half-blocks $\underline{c}_k$ and $\underline{d}_k$ of size N/2:

$$\underline{b}_k^t = (\underline{c}_k^t, \underline{d}_k^t) \qquad k = 1,\ldots,K \tag{3.25}$$

The original signal $\underline{f}$ can be written:

$$\underline{f}^t = (\underline{c}_1^t, \underline{d}_1^t, \underline{c}_2^t, \ldots, \underline{d}_{K-1}^t, \underline{c}_K^t, \underline{d}_K^t) \tag{3.26}$$

The extended signal $\bar{\underline{f}}$ can be written:

$$\bar{\underline{f}}^t = (\underline{d}_{-1}^t, \underline{c}_0^t, \underline{d}_0^t, \underline{c}_1^t, \underline{d}_1^t, \underline{c}_2^t, \ldots, \underline{d}_{K-1}^t, \underline{c}_K^t, \underline{d}_K^t, \underline{c}_{K+1}^t, \underline{d}_{K+1}^t, \underline{c}_{K+2}^t) \tag{3.27}$$

with:

$$\underline{d}_0 = H\,\underline{c}_1 \qquad \underline{c}_{K+1} = H\,\underline{d}_K$$

$$\underline{c}_0 = H\,\underline{d}_1 \qquad \underline{d}_{K+1} = H\,\underline{c}_K$$

$$\underline{d}_{-1} = H\,\underline{c}_2 \qquad \underline{c}_{K+2} = H\,\underline{d}_{K-1}$$

and where H is the N/2 x N/2 matrix $\begin{pmatrix} \emptyset & & 1 \\ & \diagup & \\ 1 & & \emptyset \end{pmatrix}$

This extension through reflection of the original signal is illustrated in Figure 2 for N = 4.

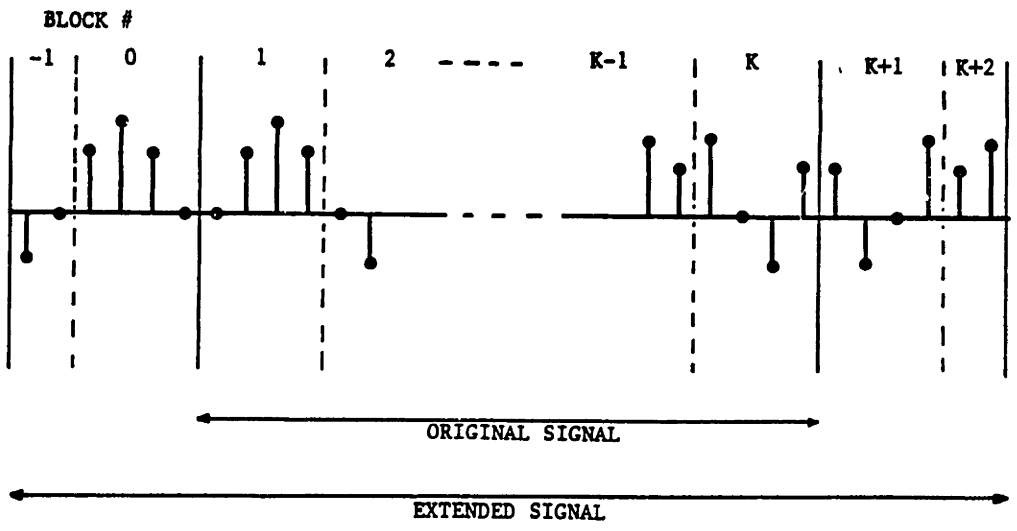


Figure 2 - Extension of the original signal by reflecting 1.5 block at each boundary of the signal.

Since L is equal to N/2, each coefficient of the transform block $\underline{b}'_k$ of any block $\underline{b}_k$ is a weighted sum of not only the N samples of $\underline{b}_k$, but also of the last N/2 samples of the preceding block $\underline{b}_{k-1}$ and the first N/2 samples of the following block $\underline{b}_{k-1}$

(respectively $\underline{d}_{k-1}$ and $\underline{c}_{k+1}$). Thus, one defines the extended block $\bar{\underline{b}}_k$:

$$\bar{\underline{b}}_k^{\,t} = (\underline{d}_{k-1}^{\,t}, \underline{b}_k^{\,t}, \underline{c}_{k+1}^{\,t}) = (\underline{d}_{k-1}^{\,t}, \underline{c}_k^{\,t}, \underline{d}_k^{\,t}, \underline{c}_{k+1}^{\,t}) \qquad (3.28)$$

$$k = 0,\ldots,K+1$$

These K+2 extended blocks of size 2N, when projected upon the N $\underline{a}_i$ basis functions, yield the K+2 transform blocks $\underline{b}'_k$ of size N, k = 0,...,K+1:

$$\underline{b}'_k = [b'_k(i)] \qquad\qquad i = 1,\ldots,N$$

with:

$$b'_k(i) = \underline{a}_i^{\,t}\bar{\underline{b}}_k \qquad (3.29)$$

The transform signal is composed of K+2 data blocks as opposed to the original signal which is composed of K data blocks. However, the coefficients of the extraneous blocks $\underline{b}'_0$ and $\underline{b}'_{K+1}$ are equal within an alternation of sign to the coefficients of $\underline{b}'_1$ and $\underline{b}'_K$ respectively. This result is due to the symmetric and antisymmetric properties of the $\underline{a}_i$ basis functions and the fact that the extended signal $\bar{\underline{f}}$ is obtained from the original signal $\underline{f}$ by reflective symmetry. Therefore, the blocks $\underline{b}'_0$ and $\underline{b}'_{K+1}$ only provide redundant information, yet they must be derived in order to have exact reconstructions of the blocks $\underline{b}_1$ and $\underline{b}_K$. In this manner, the non-redundant transform representation of $\underline{f}$ is of the same size as the original signal. This is expected since the transform is constrained to be unitary.

3.3.2. Computation of the inverse LOT

Suppose the one-dimensional LOT of $\underline{f}$ has been computed. Thus, the K transform blocks $\underline{b}'_k$ of size N, specified by equations (3.28) and (3.29), are given. Assume, as well, that the two additional transform blocks $\underline{b}'_0$ and $\underline{b}'_{K+1}$ have been derived.

An original block $\underline{b}_k$ can be expressed as the sum of the basis functions weighted by the transform coefficients. However, since the basis functions extend beyond the boundaries of the block, the reconstruction of this block also depends on the transform coefficients of the two neighboring blocks and is expressed in terms of truncated $\underline{a}_i$ basis functions. More precisely, the block $\underline{b}_k$ can be reconstructed as a summation of the tails of the $\underline{a}_i$ basis functions weighted by the coefficients of $\underline{b}'_{k-1}$, and of the middle of the basis functions weighted by the coefficients of $\underline{b}'_k$, and also of the heads of the basis functions weighted by the coefficients of $\underline{b}'_{k+1}$. This is illustrated in Figure 3 for one basis function.

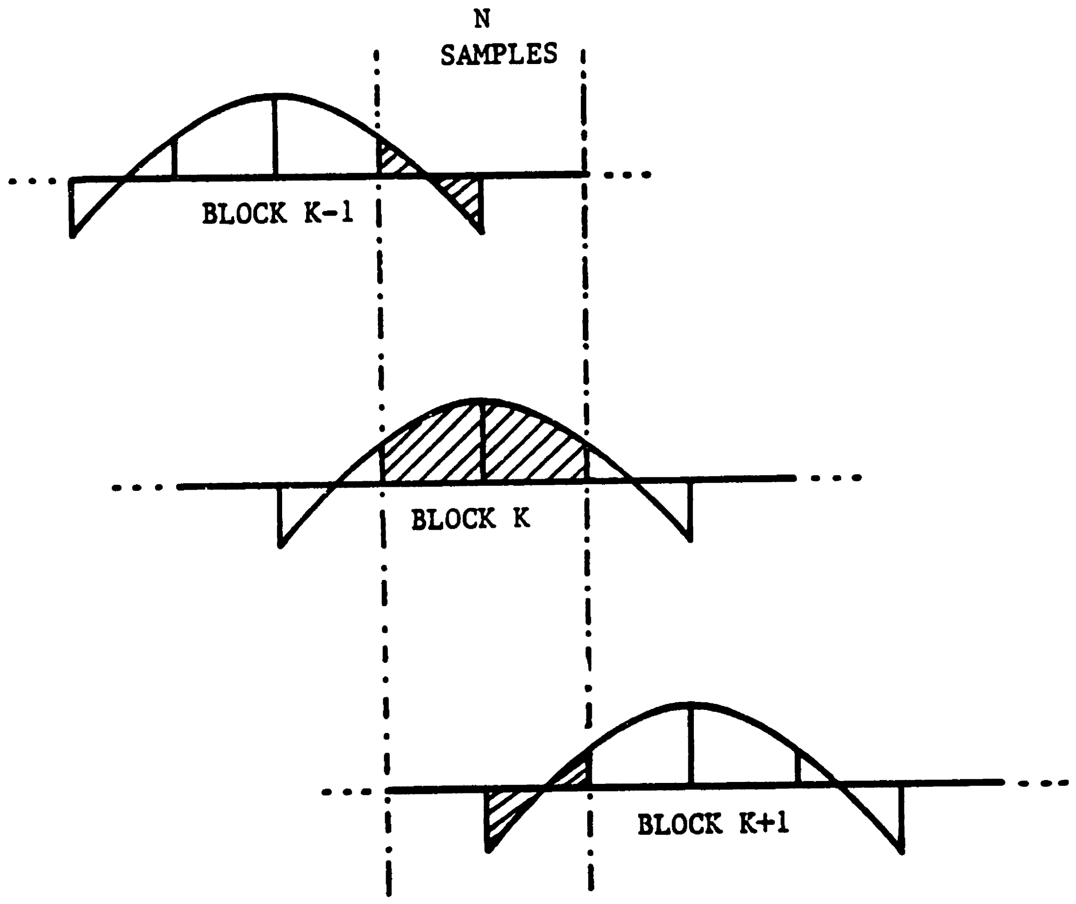


Figure 3 - Contributions of the transform blocks K-1, K, and K+1, to the reconstruction of the K-th data block.

Any $\underline{a}_i$ basis function can be divided into four segments of length N/2: $\underline{\alpha}_i$, $\underline{\beta}_i$, $\underline{\gamma}_i$, and $\underline{\delta}_i$:

$$\underline{a}_i^t = (\underline{\alpha}_i^t, \underline{\beta}_i^t, \underline{\gamma}_i^t, \underline{\delta}_i^t) \qquad i = 1,\dots,N \tag{3.30}$$

Define three vectors $\underline{\varepsilon}_{1i}$, $\underline{\varepsilon}_{2i}$, and $\underline{\varepsilon}_{3i}$, of length N, in the following way:

$$\begin{aligned} \underline{\varepsilon}_{1i}^t &= (\underline{\delta}_i^t, 0) \\ \underline{\varepsilon}_{2i}^t &= (\underline{\beta}_i^t, \underline{\gamma}_i^t) \qquad i = 1,\dots,N \\ \underline{\varepsilon}_{3i}^t &= (0, \underline{\alpha}_i^t) \end{aligned} \tag{3.31}$$

The reconstructed block $\underline{b}_k$ is derived as the sum of the vectors $\underline{\varepsilon}_{1i}$, $\underline{\varepsilon}_{2i}$, and $\underline{\varepsilon}_{3i}$, weighted by the transform coefficients of $\underline{b}'_{k-1}$, $\underline{b}'_k$ and $\underline{b}'_{k+1}$ respectively. This is represented as:

$$\underline{b}_k = \sum_{i=1}^{N} (b'_{k-1}(i)\underline{\varepsilon}_{1i} + b'_k(i)\underline{\varepsilon}_{2i} + b'_{k+1}(i)\underline{\varepsilon}_{3i}) \tag{3.32}$$

$$k = 1,\dots,K$$

From the K blocks $\underline{b}_k$, the original signal $\underline{f}$ can then be reconstructed. This completes the computation of the inverse LOT.

CHAPTER FOUR

DERIVATION OF THE BASIS FUNCTIONS

In this chapter, the algorithm which provided a numerical model to the LOT defined in section 3.2 is presented. The problem of finding the LOT is formulated first as a sequence of non-linear optimization problems which are solved iteratively. The optimization method which has been used, the augmented Lagrangian method, is described in section 4.2. The algorithm applied to the LOT is derived in section 4.3. Finally, the experimental results are provided in section 4.4.

4.1. RECURSIVE FORMULATION OF THE PROBLEM

In Chapter 3, it was proven that the mere knowledge of the N vectors $\underline{x}_i$ of size N is sufficient to determine completely the LOT. The constraints which must be satisfied by the $\underline{x}_i$ vectors are:

$$\underline{x}_i^t \, H \, \underline{x}_j = 0 \qquad i,j = 1,\dots,N \qquad (4.1)$$

$$(1+(-1)^{i+j}) \, \underline{x}_i^t \underline{x}_j = 0 \qquad i,j = 1,\dots,N; \; i \neq j \qquad (4.2)$$

Moreover, the $\underline{x}_i$ vectors are optimized in terms of energy compaction:

$$\max_{\underline{x}_i} \frac{\underline{x}_i^t \, R'_i \, \underline{x}_i}{\underline{x}_i^t \, \underline{x}_i} \qquad i = 1,\ldots,N \qquad (4.3)$$

Table 2 in section 3.2.3 provides a summary of the LOT properties.

The algorithm used to develop the LOT is a recursive procedure similar to the Gram-Schmidt procedure for the orthogonalization of a basis. The $\underline{x}_i$ vectors are determined sequentially. For each of the vectors, the objective function to be maximized is the energy compaction. However, due to the recursive procedure, the number of constraints increases as the order of the function increases.

Suppose the first i-1 basis functions have been determined previously. That is, the vectors $\underline{x}_j$, $j = 1,\ldots,i-1$, are known and have been normalized to one. At this stage, it is necessary to define the set of constraints which must be satisfied by the i-th vector $\underline{x}_i$. The constraint (4.1) yields the following set of constraints (after normalization):

$$\frac{\underline{x}_i^t \, H \, \underline{x}_i}{\underline{x}_i^t \, \underline{x}_i} = 0 \qquad (4.4)$$

and

$$\frac{(H \, \underline{x}_j)^t \, \underline{x}_i}{(\underline{x}_i^t \, \underline{x}_i)^{1/2}} = 0 \qquad j = 1,\ldots,i-1 \qquad (4.5)$$

The constraint (4.2) generates a constraint on $\underline{x}_i$ for only the values of the index j such that i+j is an even number. This, in turn, implies the following set of constraints:

$$\frac{\underline{x}_j^t \underline{x}_i}{(\underline{x}_i^t \underline{x}_i)^{1/2}} = 0 \qquad j = 1,\ldots,i-1;\ i+j \text{ even} \qquad (4.6)$$

Define the vector of constraints on $\underline{x}_i$: $\underline{c}(\underline{x}_i)$, which specifies the complete set of the constraints which must be satisfied by $\underline{x}_i$ (cf equations (4.4), (4.5), and (4.6)):

$$\underline{c}(\underline{x}_i) = \begin{bmatrix} \dfrac{\underline{x}_i^t H \underline{x}_i}{\underline{x}_i^t \underline{x}_i} \\ \left[\dfrac{(H \underline{x}_j)^t \underline{x}_i}{(\underline{x}_i^t \underline{x}_i)^{1/2}} \right]_{j = 1,\ldots,i-1} \\ \left[\dfrac{\underline{x}_j^t \underline{x}_i}{(\underline{x}_i^t \underline{x}_i)^{1/2}} \right]_{\substack{j = 1,\ldots,i-1 \\ i+j \text{ even}}} \end{bmatrix} \qquad (4.7)$$

The feasibility condition for the vector $\underline{x}_i$ is:

$$\underline{c}(\underline{x}_i) = 0 \qquad (4.8)$$

Define the objective function for the $\underline{x}_i$ vectors $f(\underline{x}_i)$ as being $-E_i$. That is:

$$f(\underline{x}_i) = - \frac{\underline{x}_i^t R'_i \underline{x}_i}{\underline{x}_i^t \underline{x}_i} \qquad (4.9)$$

where R' is the N x N matrix defined in section 3.2.3 (equation (3.24)). The optimality condition for the vector $\underline{x}_i$ is:

$$f(\underline{x}_i) = \min_{\underline{x}} f(\underline{x}) \tag{4.10}$$

Consequently, the problem of finding the LOT is formulated as a sequence of N non-linear optimization programs of the general standard form:

$$\min_{\underline{x}} f(\underline{x}) \text{ subject to } \underline{c}(\underline{x}) = 0 \tag{4.11}$$

In the formulation of the LOT, the programs are non-linear because of the constraint (4.4). Due to this constraint, the feasible sets of the programs are also non-convex. Hence, non-linear optimization techniques must be investigated to solve the N programs. The recursive formulation of the problem of finding the LOT is summarized in Table 3.

4.2. THE AUGMENTED LAGRANGIAN METHOD

In this section the optimization technique, which has been used to solve the N non-linear programs described in the previous section, is presented.

Consider the following general non-linear optimization problem in standard form (4.11):

$$\min_{\underline{x}} f(\underline{x}) \text{ subject to } \underline{c}(\underline{x}) = 0 \tag{4.11}$$

Table 3 - Recursive formulation of the problem of defining the LOT

Find recursively the N vectors $\underline{x}_i$ of size N. For any vector $\underline{x}_i$, i = 1,...,N, solve the following program:

$$\min_{\underline{x}_i} f(\underline{x}_i) \text{ subject to } \underline{c}(\underline{x}_i) = 0$$

where:

$$f(\underline{x}_i) = - \frac{\underline{x}_i^t R'_i \underline{x}_i}{\underline{x}_i^t \underline{x}_i}$$

with:

$$R'_i = [R'_i(k,l)]$$

$$R'_i(k,l) = \rho^{|k-l|} + (-1)^{i+1} \rho^{2N+1-k-l} \qquad k,l = 1,\ldots,N$$

and:

$$\underline{c}(\underline{x}_i) = \begin{bmatrix} \dfrac{\underline{x}_i^t H \underline{x}_i}{\underline{x}_i^t \underline{x}_i} \\[2ex] \left[\dfrac{(H \underline{x}_j)^t \underline{x}_i}{(\underline{x}_i^t \underline{x}_i)^{1/2}} \right]_{j = 1,\ldots,i-1} \\[2ex] \left[\dfrac{\underline{x}_j^t \underline{x}_i}{(\underline{x}_i^t \underline{x}_i)^{1/2}} \right]_{\substack{j = 1,\ldots,i-1 \\ i+j \text{ even}}} \end{bmatrix}$$

where $f(\underline{x})$ is the objective function and $\underline{c}(\underline{x})$ is the vector of constraints.

The Lagrangian function of the problem is defined as:

$$L(\underline{x},\underline{\lambda}) = f(\underline{x}) - \underline{\lambda}^t \underline{c}(\underline{x}) \tag{4.12}$$

where $\underline{\lambda}$ is the Lagrange multipliers' vector [4], [25]. Suppose $(\underline{x}^*,\underline{\lambda}^*)$ is the optimum solution of the problem (4.11). The first-order conditions state that $(\underline{x}^*, \underline{\lambda}^*)$ must be a stationary point of the Lagrangian function [4]. That is, the first-order derivatives of $L(\underline{x},\underline{\lambda})$ with respect to $\underline{x}$ and $\underline{\lambda}$, evaluated at $(\underline{x}^*,\underline{\lambda}^*)$, must be equal to zero:

$$\nabla_{\underline{x}} L(\underline{x},\underline{\lambda})\Big|_{(\underline{x}^*,\underline{\lambda}^*)} = \nabla_{\underline{\lambda}} L(\underline{x},\underline{\lambda})\Big|_{(\underline{x}^*,\underline{\lambda}^*)} = 0 \tag{4.13}$$

The symbol $\nabla_{\underline{x}}$ refers to the derivative with respect to $\underline{x}$, that is, the gradient with respect to $\underline{x}$. The second order condition states that the Lagrangian function must have non-negative curvature at $(\underline{x}^*,\underline{\lambda}^*)$ [4]. The conditions (4.13) yield a set of equations which must be satisfied by the optimum solution $(\underline{x}^*,\underline{\lambda}^*)$:

$$\nabla_{\underline{x}} f(\underline{x}) = \nabla_{\underline{x}} (\underline{c}^t (\underline{x}))\, \underline{\lambda}$$

$$\underline{c}(\underline{x}) = 0 \tag{4.14}$$

where $\nabla_{\underline{x}} f(\underline{x})$ is the gradient of $f(\underline{x})$ with respect to $\underline{x}$, and $\nabla_{\underline{x}}(\underline{c}^t(\underline{x}))$ is the Jacobian matrix of $\underline{c}(\underline{x})$ with respect to $\underline{x}$ (the matrix whose columns are the gradients of each one of the components of the row vector $\underline{c}^t(\underline{x})$). The solution of the set of equations (4.14) satisfies the first-order conditions of optimality. This method, known as the Lagrange multipliers' method, requires the solving of the equations (4.14). Unfortunately, in the case of a non-linear problem, an exact closed-form solution may be impossible to determine. Thus, iterative methods must be considered.

The augmented Lagrangian method is a sequential multiplier method [25]. The augmented Lagrangian function is defined as the Lagrangian function plus an added penalty function. The penalty function basically is an increasing function of the constraints' vector $\underline{c}(\underline{x})$. For the problem of finding the LOT, a quadratic penalty function was chosen. In this case, the augmented Lagrangian function $\phi(\underline{x},\underline{\lambda},\sigma)$ is defined as follows:

$$\phi(\underline{x},\underline{\lambda},\sigma) = L(\underline{x},\underline{\lambda}) + {}^1/_2\ \sigma\ \underline{c}^t(\underline{x})\underline{c}(\underline{x}) \tag{4.15}$$

This can also be stated as:

$$\phi(\underline{x},\underline{\lambda},\sigma) = f(\underline{x}) - \underline{\lambda}^t\underline{c}(\underline{x}) + {}^1/_2\ \sigma\underline{c}^t(\underline{x})\underline{c}(\underline{x}) \tag{4.16}$$

where σ is a fixed parameter. The value of $\underline{\lambda}$ for which $\underline{x}^*$ mininizes $\phi(\underline{x},\underline{\lambda},\sigma)$ is $\underline{\lambda}^*$. For a fixed parameter σ, the algorithm works as follows [25]:

1. Find a sequence $\{\underline{\lambda}_n\}$ such that $\lim_{n \to \infty} \underline{\lambda}_n = \underline{\lambda}^*$
2. For each $\underline{\lambda}_n$, find the local minimizer $\underline{x}(\underline{\lambda}_n)$ to $\phi(\underline{x},\underline{\lambda}_n,\sigma)$.

3. Stop the iterations on $\underline{\lambda}$ when $\underline{c}(\underline{x}(\underline{\lambda}_n))$ is approximately equal to zero.

The two main features of this algorithm are: the optimum $\underline{\lambda}^*$ is determined iteratively; and the solution is not forced to satisfy the constraints exactly, but instead, is forced to satisfy them with a variable precision.

The key in the augmented Lagrangian method is the iterative rule to find the sequence $\{\underline{\lambda}_n\}$. Consider the following function $\psi(\underline{\lambda})$:

$$\psi(\underline{\lambda}) = \phi(\underline{x}(\underline{\lambda}),\underline{\lambda},\sigma) \tag{4.17}$$

where $\underline{x}(\underline{\lambda})$ minimizes $\phi(\underline{x},\underline{\lambda},\sigma)$.

The main property of the augmented Lagrangian function, stated earlier, implies that:

$$\lim_{\underline{\lambda} \to \underline{\lambda}^*} \underline{x}(\underline{\lambda}) = \underline{x}(\underline{\lambda}^*) = \underline{x}^* \tag{4.18}$$

Since $\underline{x}(\underline{\lambda})$ is the minimum of the function $\phi(\underline{x},\underline{\lambda},\sigma)$, the following inequality is true for any $\underline{\lambda}$:

$$\phi(\underline{x}(\underline{\lambda}),\underline{\lambda},\sigma) < \phi(\underline{x}^*,\underline{\lambda},\sigma)$$

consequently:

$$\psi(\underline{\lambda}) < \phi(\underline{x}^*,\underline{\lambda},\sigma) \tag{4.19}$$

For the optimum $\underline{x}^*$, $\underline{c}(\underline{x}^*)$ is equal to zero. Therefore, $\phi(\underline{x}^*,\underline{\lambda},\sigma)$ is independent of $\underline{\lambda}$:

$$\phi(\underline{x}^*,\underline{\lambda},\sigma) = f(\underline{x}^*) \tag{4.20}$$

Additionally:

$$f(\underline{x}^*) = \phi(\underline{x}^*,\underline{\lambda}^*,\sigma) \tag{4.21}$$

Combining the results expressed in equations (4.18), (4.20), and (4.21) yields:

$$\phi(\underline{x}^*,\underline{\lambda},\sigma) = \phi(\underline{x}(\underline{\lambda}^*),\underline{\lambda}^*,\sigma) = \psi(\underline{\lambda}^*) \tag{4.22}$$

Equations (4.19) and (4.22) imply that $\psi(\underline{\lambda}) < \psi(\underline{\lambda}^*)$ for any $\underline{\lambda}$. Consequently, it has been demonstrated that $\underline{\lambda}^*$ is a maximizer of the function $\psi(\underline{\lambda})$ defined by equation (4.17).

The main consequence of this property is that an unconstrained minimization method can be applied to $-\psi(\underline{\lambda})$ to define the sequence $\{\underline{\lambda}_n\}$. The Newton's method yields the following iteration rule [25]:

$$\underline{\lambda}_{n+1} = \underline{\lambda}_n - (J^t H^{-1} J)^{-1} \underline{c}(\underline{x}(\underline{\lambda}_n)) \tag{4.23}$$

where J is the Jacobian matrix of $\underline{c}(\underline{x})$: $J = \nabla_{\underline{x}}(\underline{c}^t(\underline{x}))$, and H is the Hessian matrix of $\phi(\underline{x},\underline{\lambda},\sigma)$: $H = \nabla_{\underline{x}}^2(\phi(\underline{x},\underline{\lambda},\sigma))$. With the iteration rule (4.23), the convergence of the sequence $\{\underline{\lambda}_n\}$ to $\underline{\lambda}^*$ is quadratic. However, the algorithm was developed using a linear approximation to this rule, which is easier to implement [25]. The iteration then reduces to:

$$\underline{\lambda}_{n+1} = \underline{\lambda}_n - \sigma\underline{c}(\underline{x}(\underline{\lambda}_n)) \tag{4.24}$$

The parameter σ takes an a-priori fixed value. It can, however, be augmented during the iteration to increase the speed of convergence.

Because of the non-convexity of the feasible set, analyzing the global optimality is extremely difficult. However, because of the convexity of the objective function along every dimension, the solution of the Lagrange's equations (4.14) must be a local minimum. In the augmented Lagrangian method, the intermediate computed points are not forced to stay in the non-convex feasible set. Therefore, the global minimum is more likely to be obtained with such a sequential method.

4.3. DERIVATION OF THE ALGORITHM FOR THE LOT

The algorithm which has been used for the LOT to solve the problem of the minimization of $\phi(\underline{x},\underline{\lambda}_n,\sigma)$ for a given $\underline{\lambda}_n$, is presented in this section. In the general formulation of the augmented Lagrangian problem, this aspect is generally not discussed. The method used to solve this problem is independent of the augmented Lagrangian algorithm, and depends only on the specific applications when the algorithm is applied.

For the problem of finding the LOT, an adaptive step gradient search method has been used to solve the unconstrained minimization problem corresponding to step 2 of the augmented Lagrangian method.

For a given $\underline{\lambda}$, one has to find $\underline{x}(\underline{\lambda})$ such that $\underline{x}(\underline{\lambda})$ minimizes $\phi(\underline{x},\underline{\lambda},\sigma)$. Gradient search methods are iterative procedures. That is, a sequence $\{\underline{x}_n(\lambda)\}$ is derived such that $\lim_{n \to \infty} \underline{x}_n(\underline{\lambda}) = \underline{x}(\underline{\lambda})$.
For an adaptive step gradient search (also known as steepest descent)

method, the iteration rule is given by:

$$\underline{x}_{n+1}(\underline{\lambda}) = \underline{x}_n(\underline{\lambda}) - \varepsilon(n)\, \nabla_{\underline{x}}\phi(\underline{x},\underline{\lambda},\sigma)\Big|_{\underline{x} = \underline{x}_n(\underline{\lambda})} \tag{4.25}$$

where $\varepsilon(n)$ is the step size, which can be modified at each step, and where $\nabla_{\underline{x}}\, \phi(\underline{x},\underline{\lambda},\sigma)$ is the gradient with respect to $\underline{x}$ of the objective $\phi(\underline{x},\underline{\lambda},\sigma)$. In the implementation of the algorithm for the LOT, $\varepsilon(n)$ was updated according to the following simple rule: as long as the objective value $\phi(\underline{x},\underline{\lambda},\sigma)$ is decreasing, the step size $\varepsilon(n)$ is increased by a fixed factor at each iteration; whenever $\phi(\underline{x},\underline{\lambda},\sigma)$ starts increasing, the step size is decreased by a fixed factor and $\underline{x}$ is reinitialized to the value having yielded the smallest previous value of the objective function. This adaptive rule has the advantage of being easy to implement and significantly increases the convergence speed of the algorithm.

A critical aspect to any gradient search method is the precise evaluation of the gradient of the objective function. However, what makes gradient search methods worthwhile for the augmented Lagrangian method applied to the LOT is that $\nabla_{\underline{x}}\phi(\underline{x},\underline{\lambda},\sigma)$ has a closed-form expression and thus can be computed precisely. The expression of $\phi(\underline{x},\underline{\lambda},\sigma)$ is given by equation (4.16). Taking the gradient of this expression with respect to $\underline{x}$ produces:

$$\nabla_{\underline{x}}\phi(\underline{x},\underline{\lambda},\sigma) = \nabla_{\underline{x}}f(\underline{x}) - \nabla_{\underline{x}}(\underline{c}^t(\underline{x}))\underline{\lambda} + \sigma\nabla_{\underline{x}}(\underline{c}^t(\underline{x}))\underline{c}(\underline{x}) \tag{4.26}$$

The optimization problem structure, formulated for the LOT in section 4.1, is such that closed-form solutions for both the gradient of $f(\underline{x})$, $\nabla_{\underline{x}} f(\underline{x})$, and the Jacobian matrix of $\underline{c}(\underline{x})$, $\nabla_{\underline{x}}(\underline{c}^t(\underline{x}))$, exist. The exact value of $\nabla_{\underline{x}}\phi(\underline{x},\underline{\lambda},\sigma)$ can therefore be computed at each step of the algorithm. The detailed derivations of $\nabla_{\underline{x}} f(\underline{x})$ and $\nabla_{\underline{x}}(\underline{c}^t(\underline{x}))$ for the LOT are given in Appendix B.

4.4. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The algorithm described in sections 4.2 and 4.3 was implemented on a digital computer to obtain a numerical model for the LOT for two different block sizes: $N = 8$ and $N = 16$. In both cases, N programs for the N $\underline{x}_i$ vectors were run successively according to the recursive formulation given in Table 3 (section 4.1).

The augmented Lagrangian algorithm was used in each program. The chosen value of the correlation factor ρ was 0.9.

The iterations of the Lagrange multipliers' vector were computed according to equation (4.24). A value of the parameter σ, chosen between 10 and 50, yielded an acceptable convergence behavior for all of the basis functions.

The minimization of $\phi(\underline{x},\underline{\lambda}_n,\sigma)$ with respect to $\underline{x}$, for a given $\underline{\lambda}_n$, was performed using the adaptive step gradient search algorithm described by equation (4.25). The initial step size $\varepsilon(0)$ was of the order of 10^{-5}. Then, if the objective function was decreasing, the step size was increased by a factor of 2 at each iteration. If the objective function was increasing, the step size was decreased by a factor of 10. It was discovered experimentally that between 50 and 100 iterations were sufficient to obtain the optimal value of $\underline{x}$, $\underline{x}(\underline{\lambda}_n)$.

The overall algorithm was discontinued when no further improvements of the objective function and constraints' vector values could be obtained. The number of iterations of $\underline{\lambda}$ necessary to achieve this result varied widely between the basis functions. Typically, the low-order basis functions required between 20 and 30 iterations of $\underline{\lambda}$. The higher-order basis functions required up to 100 iterations of $\underline{\lambda}$. Because of the finite precision of the computations, the constraints were also satisfied with a finite precision. Moreover, because of the recursive procedure, the precision, by which the constraints were satisfied, worsened as the order of the basis function increased. For the first N/2 basis functions, the precision achieved was of the order of 10^{-9}. For N = 8, the precision of the last N/2 basis function was of the order of 10^{-5}. For N = 16, this precision was only of the order of 10^{-3}. Because of this finite precision, it is conceivable to believe that the transform obtained from the numerical model is not perfectly invertible. However, experiments showed that this is not the case. When the transform and the inverse transform of an image were computed (see description in section 3.3), the reconstructed image had a high signal to noise ratio (SNR > 50dB) for N = 16, and was exactly identical to the original image for N = 8. Therefore, the finite precision of the numerical model did not affect the precision of the integer reconstruction of the image. The numerical values of the LOT basis function coefficients are given in Appendix C, for N = 8 and 16.

In Figure 4, the basis functions of the LOT for N = 8 are represented as discrete sequences. In Figure 5, the basis function of the LOT for N = 16, interpolated with a (sin x)/x window, are represented as continuous waveforms. In both figures the complete $\underline{a}_i$ basis functions are represented (2N points long). As might be expected, the order of harmonics increases with the order of the basis functions. Additionally, the typical structure of the LOT basis functions looks like that of modulated sinewaves.

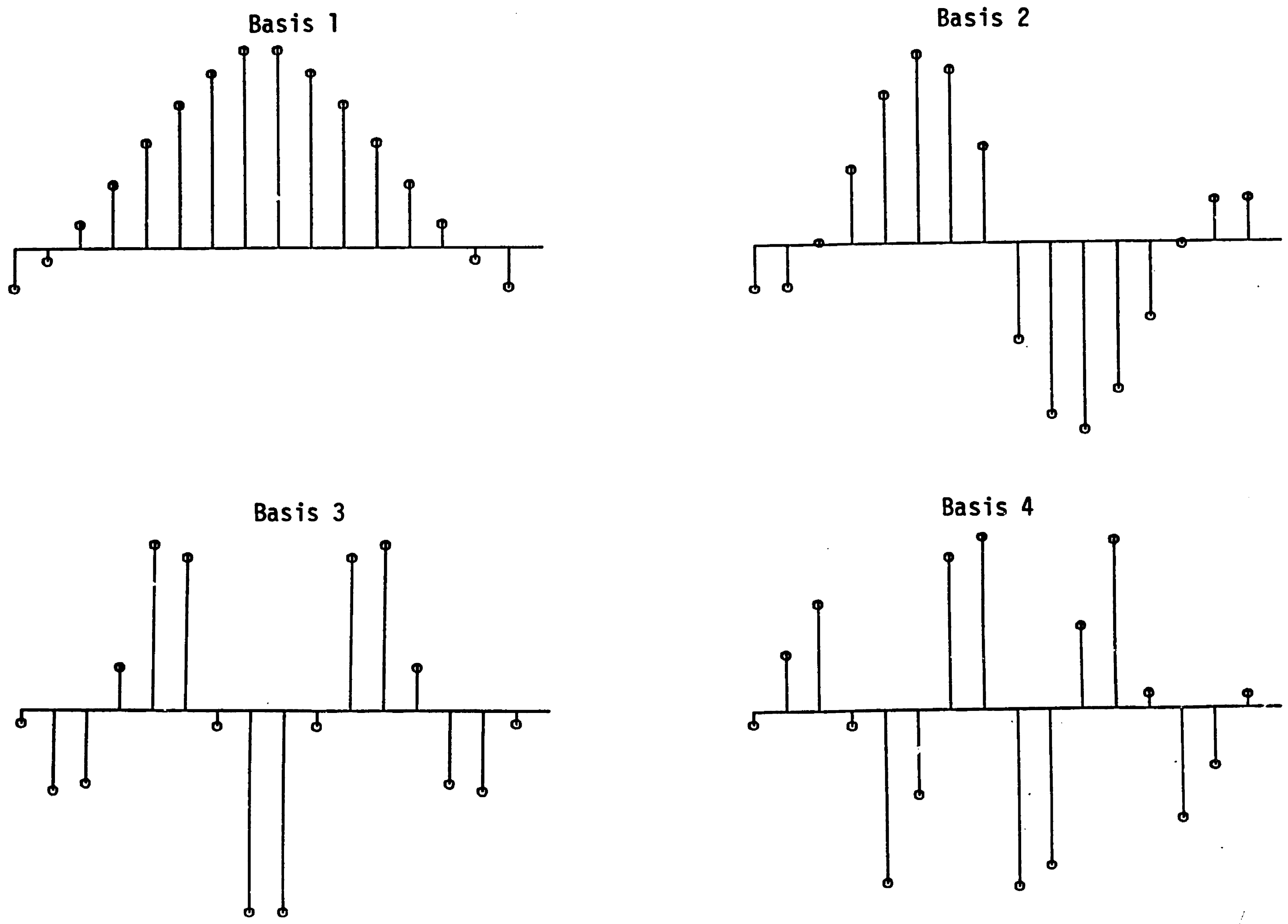


Figure 4 - LOT basis functions # 1-4 for N=8 and L=4.

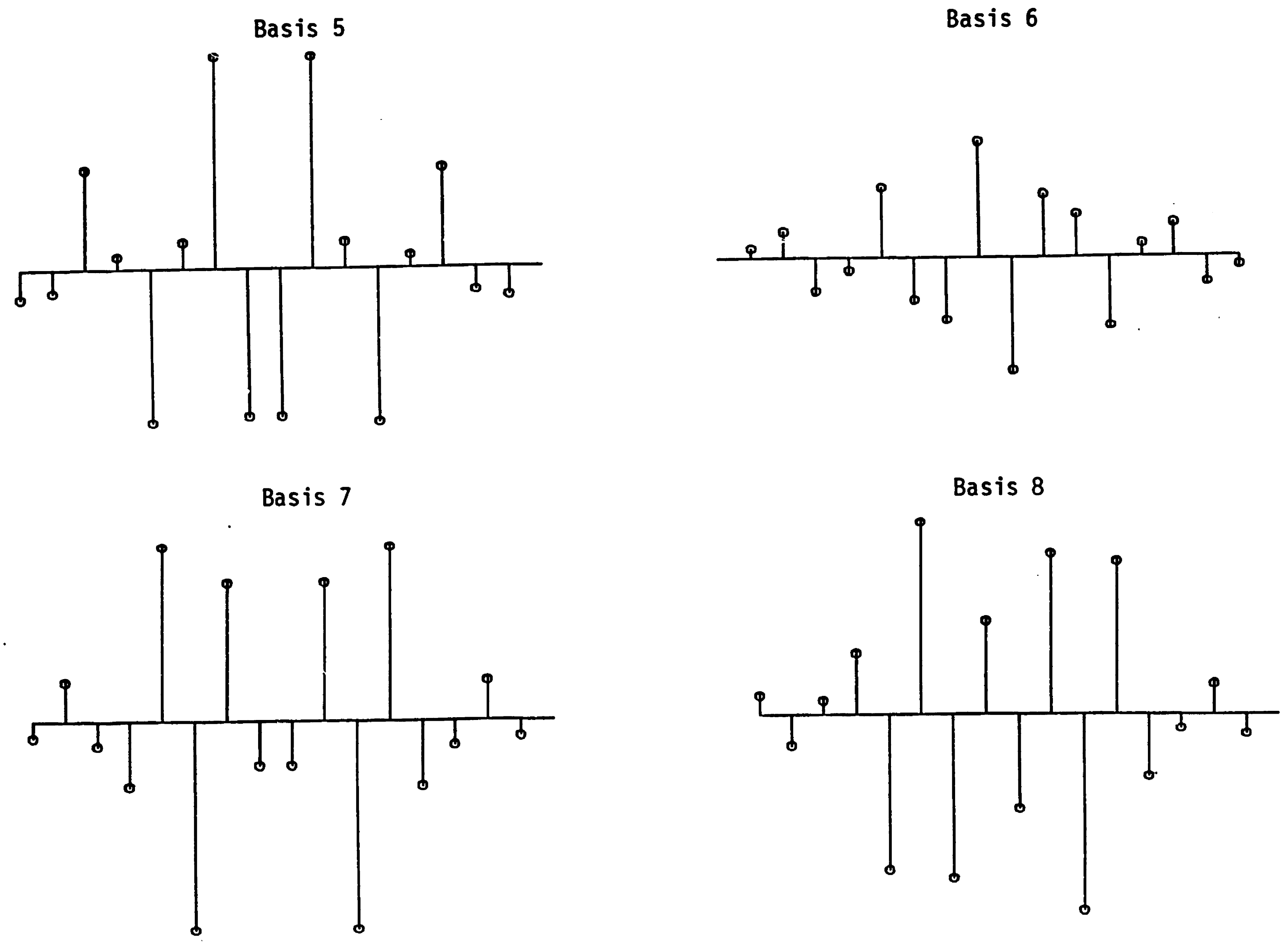
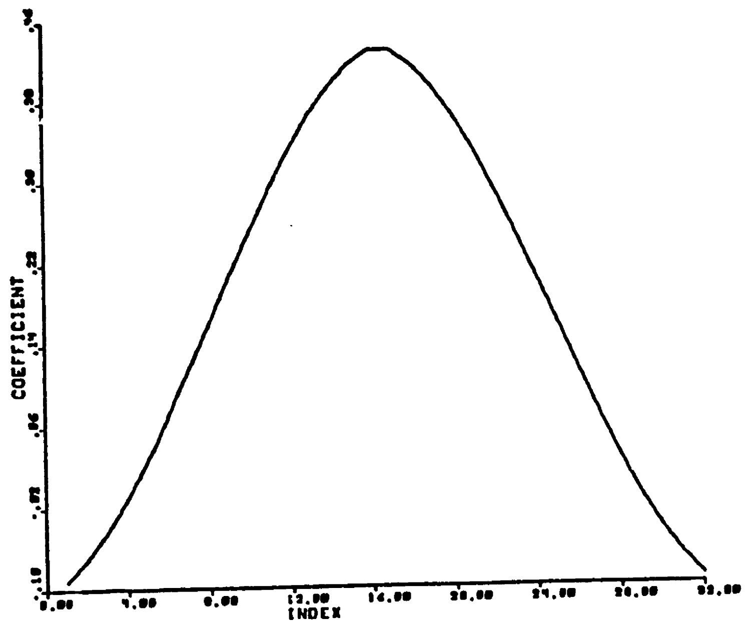
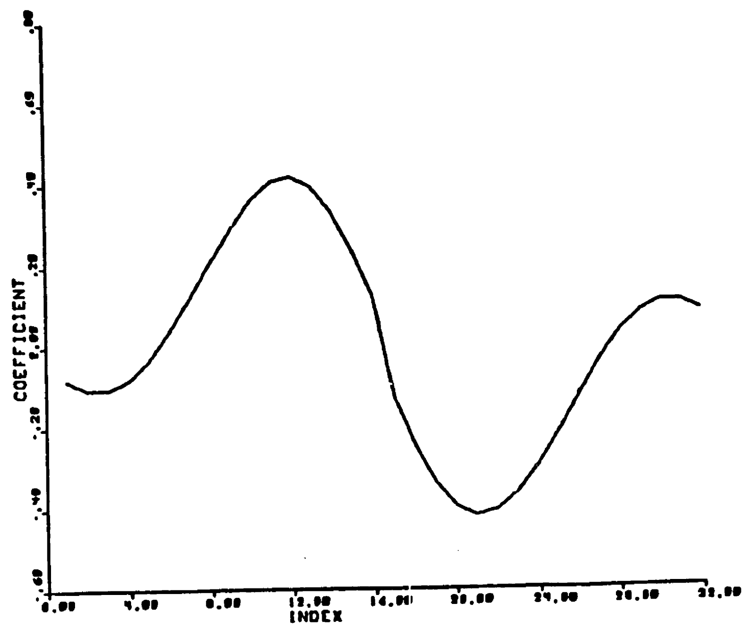


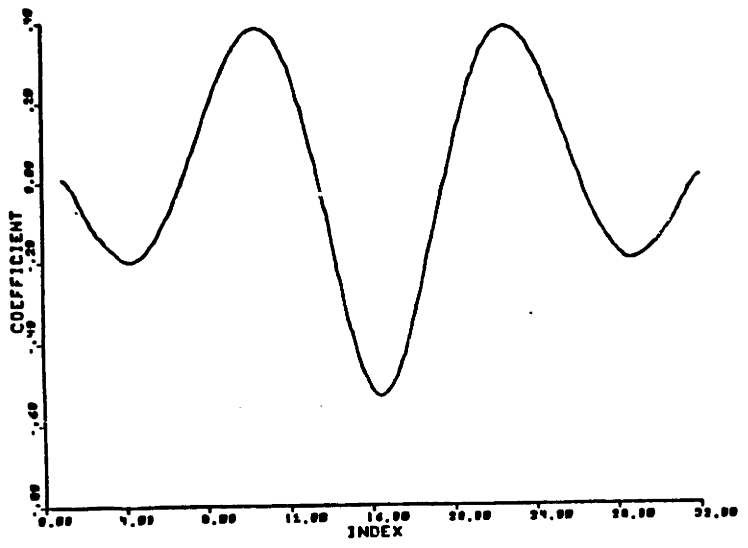
Figure 4 (cont.) - LOT basis functions # 5-8 for N=8 and L=4.



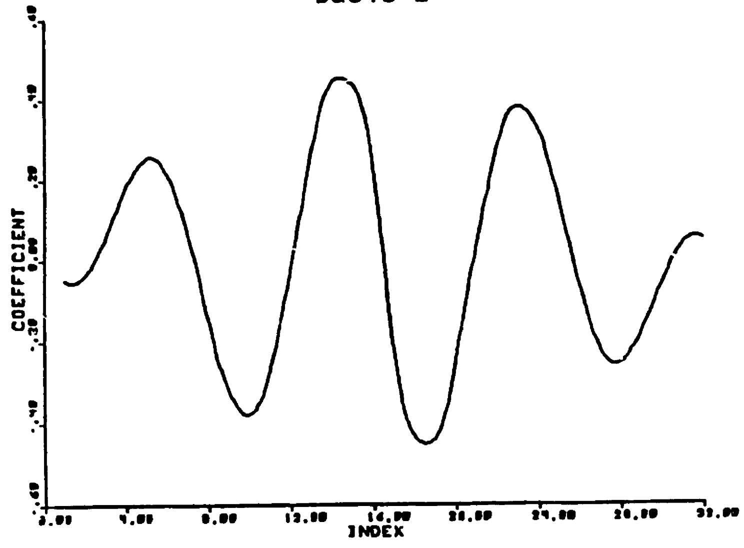
Basis 1



Basis 2

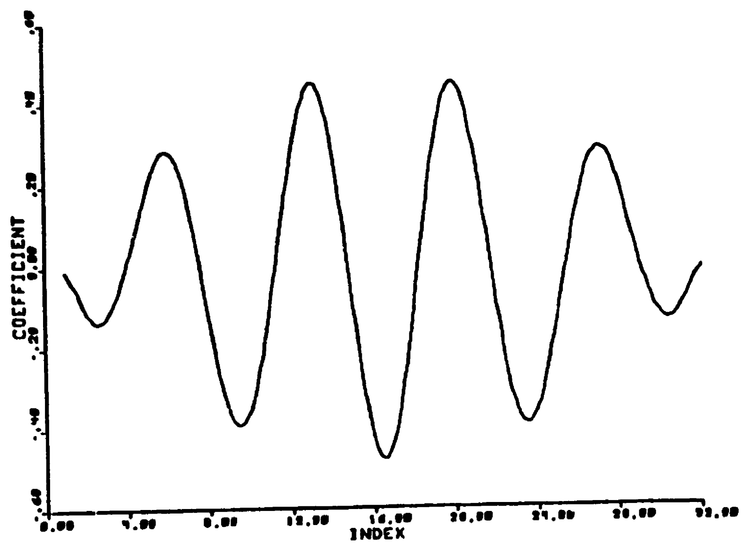


Basis 3

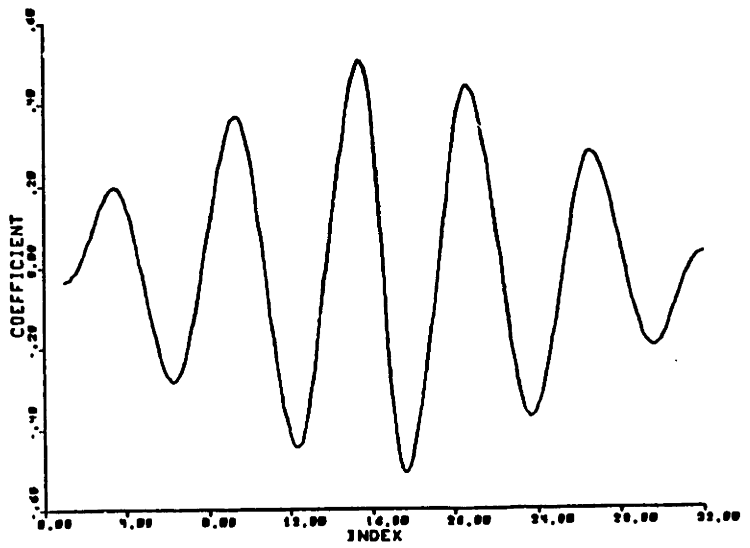


Basis 4

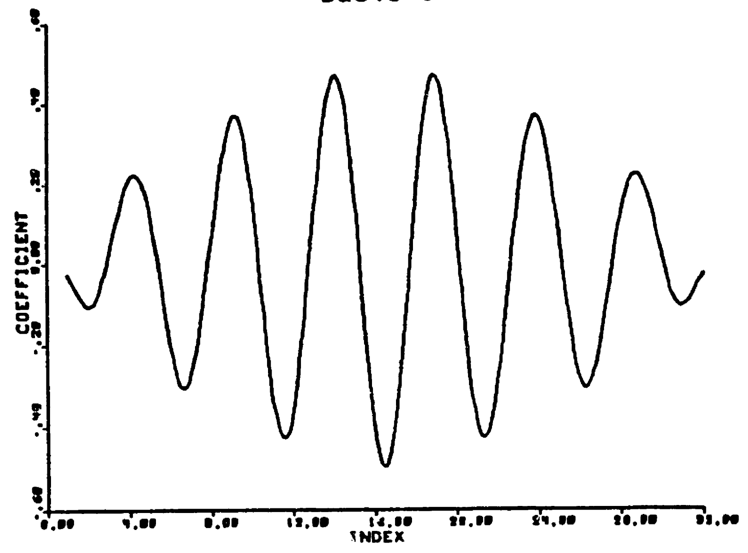
Figure 5 - LOT basis functions # 1-4, interpolated with a (sin x)/x window, for N=16 and L=8.



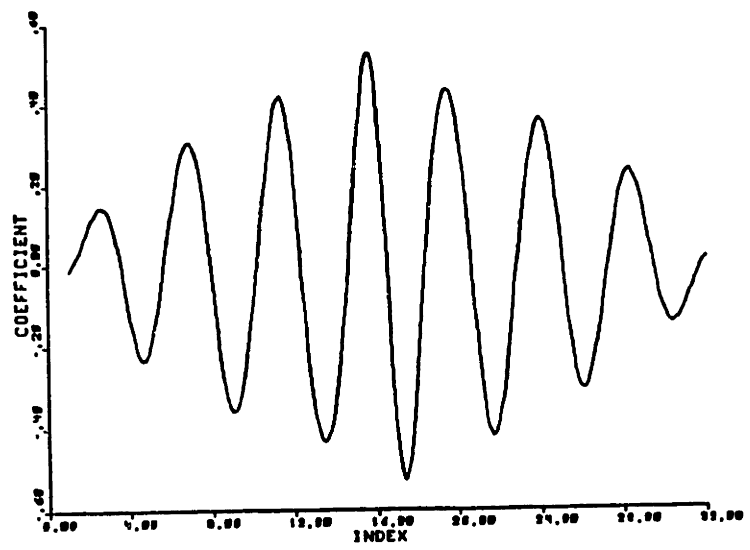
Basis 5



Basis 6

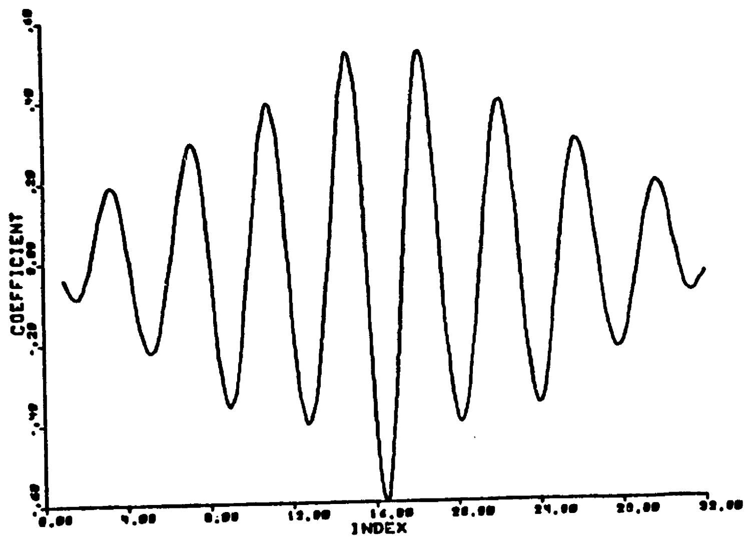


Basis 7

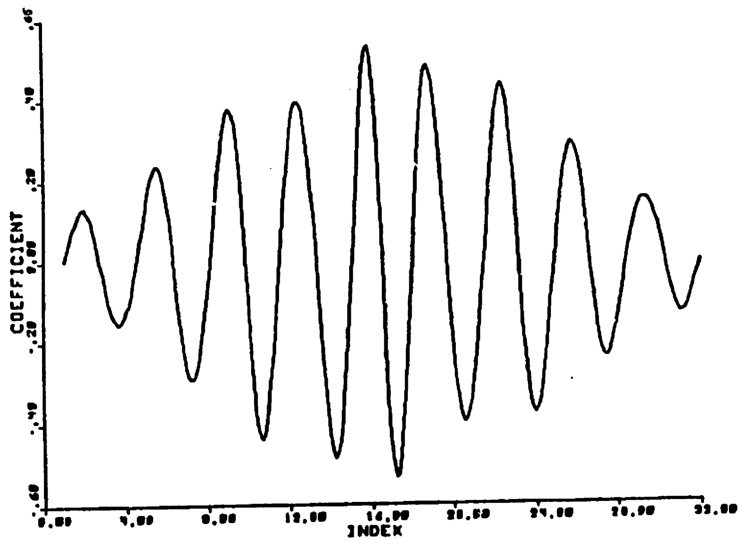


Basis 8

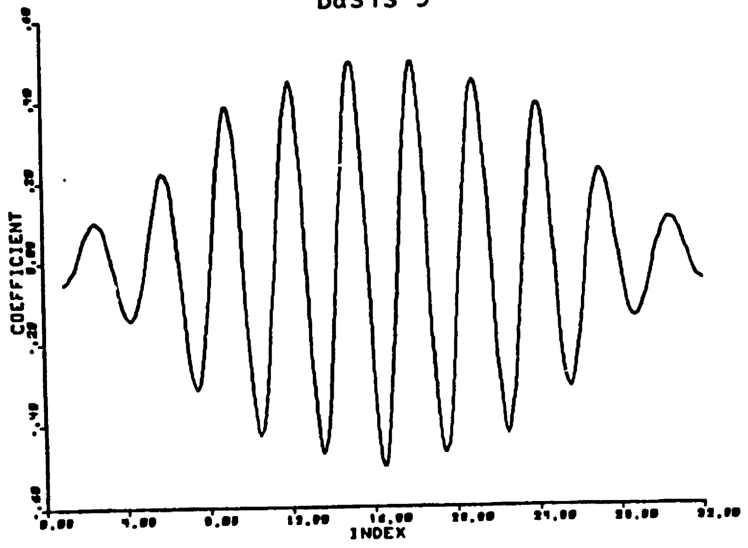
Figure 5 (cont.) - LOT basis functions # 5-8, interpolated with a (sin x)/x window, for N=16 and L=8.



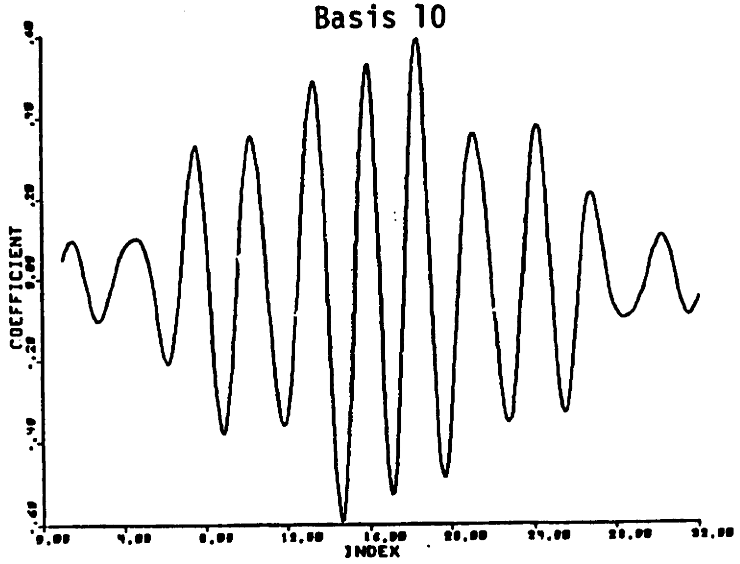
Basis 9



Basis 10

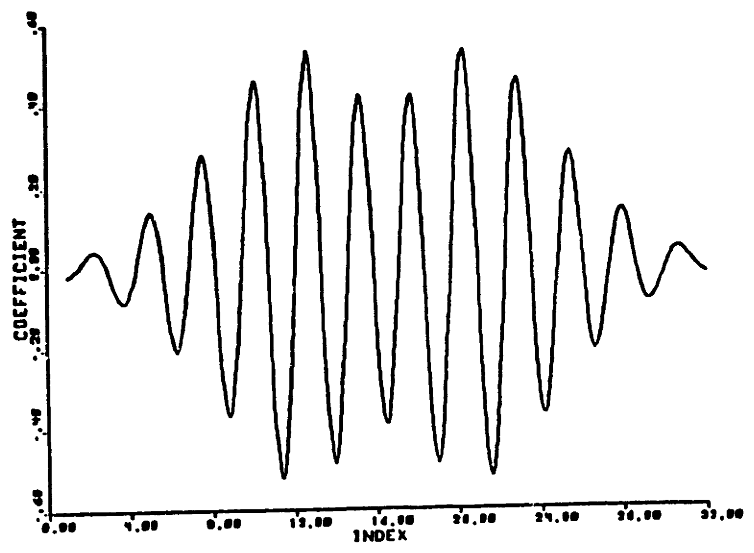


Basis 11

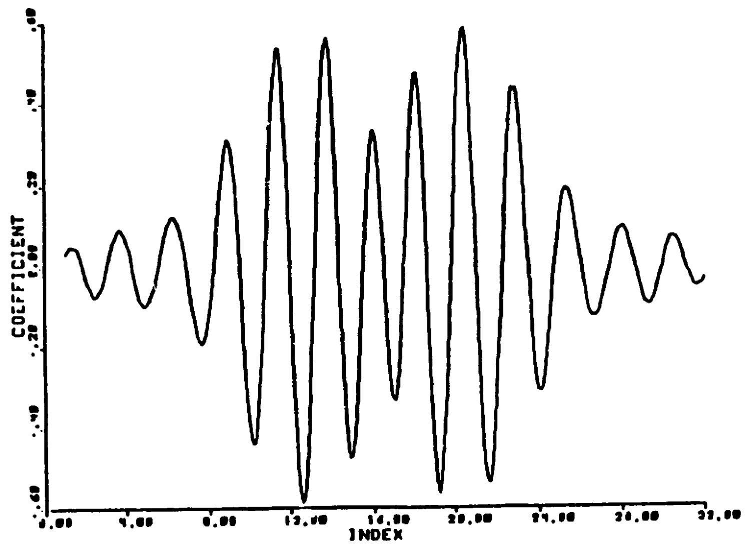


Basis 12

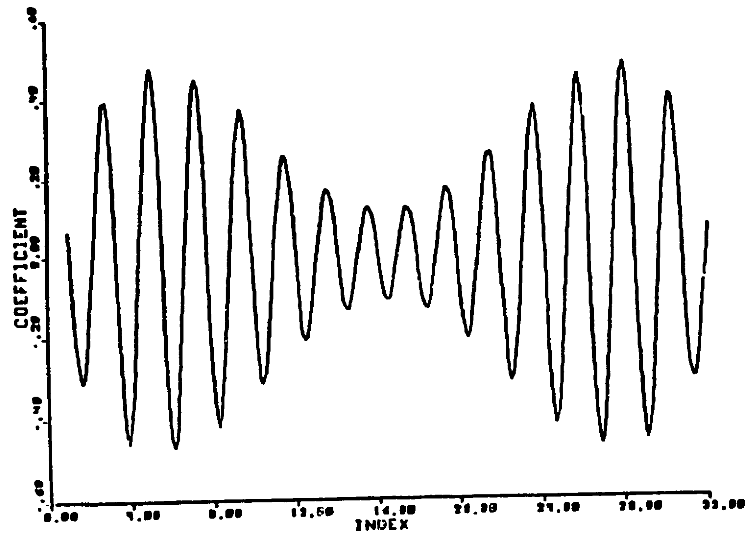
Figure 5 (cont.) - LOT basis functions # 9-12, interpolated with a (sin x)/x window, for N=16 and L=8.



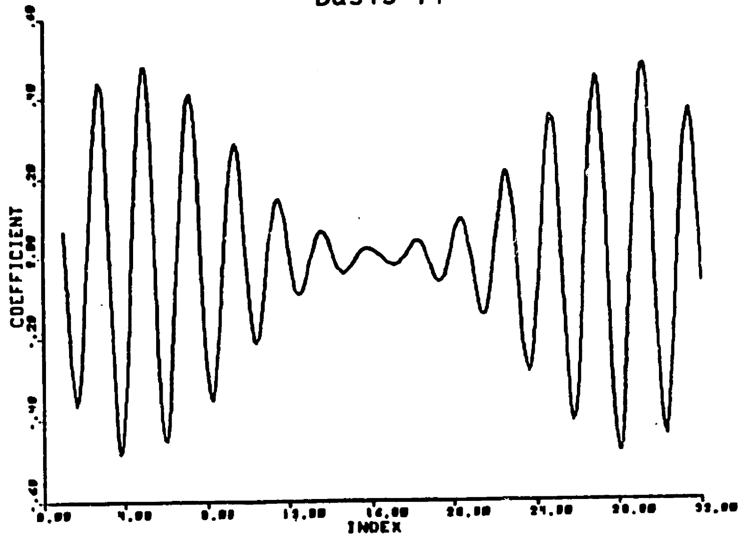
Basis 13



Basis 14



Basis 15



Basis 16

Figure 5 (cont.) - LOT basis functions # 13-16, interpolated with a (sin x)/x window, for N=16 and L=8.

CHAPTER FIVE

APPLICATION OF THE LOT TO INTRAFRAME CODING

Intraframe coding refers to the coding of single still images. In this chapter, the architecture of a typical zonal transform coding system is described [12], [26]. Both non-adaptive and adaptive coding schemes are investigated. Low bit-rate transform coding of single images was performed using the DCT, the LOT and the SSFT. The results of these experiments are presented in section 5.4.

5.1. DESCRIPTION OF THE SIMULATED SYSTEM ARCHITECTURE

Consider a monochrome digitalized image F of resolution R. The image F can be described as an R x R array of R^2 integer numbers. Each number represents the luminance value of the corresponding pixel, quantized with a fixed-length code word encryption scheme. The use of transform coding techniques may achieve data compression of the image F. In order to quantify the data compression ability of a system, the data rate d of a coded image is defined as the average number of bits per pixel used to code this image. For the original image F, d is simply equal to the number of bits assigned to the luminance value of one pixel.

The architecture of a transform monochrome image coding system is illustrated in Figure 6 [26], [41].

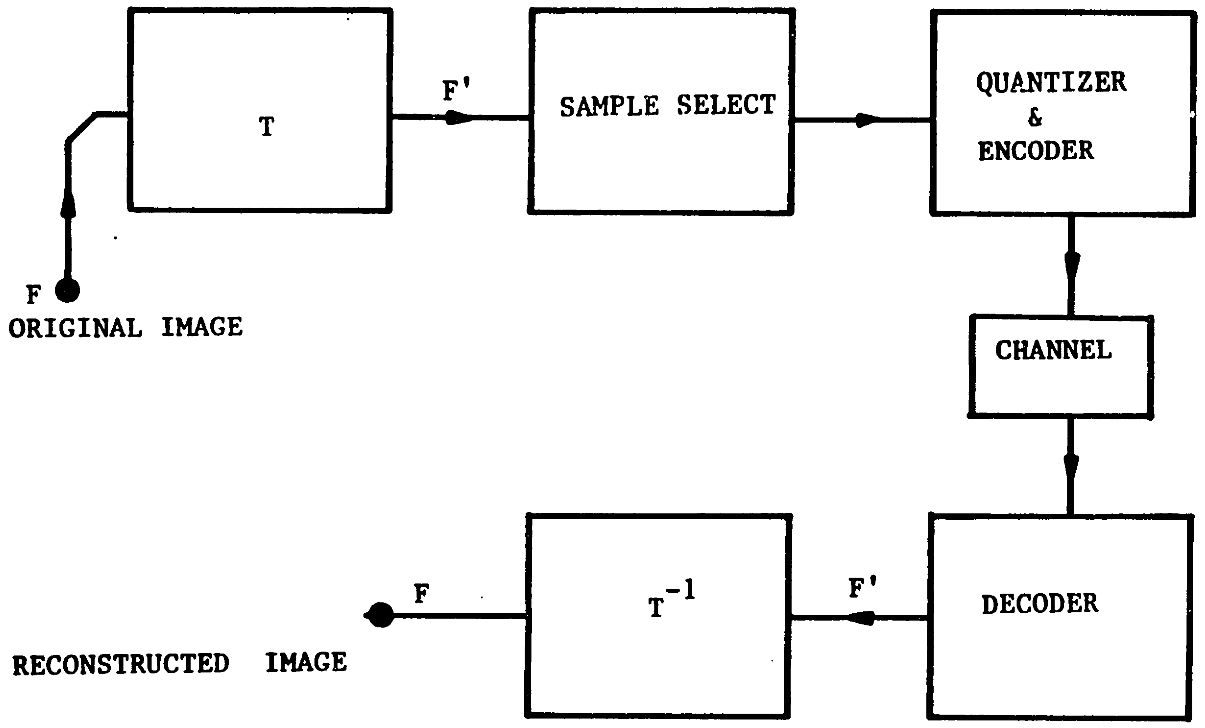


Figure 6 - A transform monochrome image coding system.

In Figure 6, T represents the forward transform operator, and T^{-1} is the inverse transform operator. Since T is a unitary transform, the size of the array F' is the same as the size of F. The channel is assumed to be a digital link.

Data compression can be achieved with a transform coding system because not all the transform coefficients of F' are transmitted. Typically, only coefficients with large variances are quantized and coded. The purpose of the sample selector is to decide which samples are to be transmitted. For a digital link, these transmitted coefficients are quantized, coded, and transmitted in binary form. At the receiver, the data is decoded and the inverse transform is taken to reconstruct the coded image F.

Two different types of sampling procedures are used in transform image coding: zonal sampling and threshold sampling [26]. In zonal sampling, only the transform coefficients lying in an a-priori fixed geographic region of the block are transmitted. These coefficients are typically the low-frequency coefficients. In threshold sampling, only the transform coefficients whose magnitudes are larger than a fixed threshold are transmitted. Threshold sampling implies that the coding scheme varies from block to block, as opposed to zonal sampling. In this manner, threshold sampling makes the coding procedure adaptive. The coefficients which are to be transmitted depend on only the block which is to be quantized, and are directly related to the corresponding local structure of the image (background, edge, detailed region, etc.). Threshold sampling is superior to zonal sampling for adaptation. However, with threshold sampling, the receiver has no a-priori knowledge of which coefficients are transmitted. Therefore, the position of each transmitted coefficient must be coded and communicated to the receiver, which usually is done using a run-length coding scheme [5], [26]. This means that more data has to be transmitted and makes the coder more complex.

In the intraframe coding experiments, which are described in the following sections, zonal sampling is used. Since the receiver has an a-priori knowledge of the positions of the transmitted coefficients, only the binary code words of these coefficients need to be communicated. This procedure, however, is not as adaptive as threshold coding. A non-adaptive zonal coding scheme is described in section 5.2. To improve the performance of the coder, adaptive zonal coding schemes can be used [6], [10], [37]. Each block is assigned a category. The zonal coding scheme, which is used for a block, depends upon which category the block has been assigned. This adaptive coding scheme, with four categories, is described in section 5.3.

5.2. NON-ADAPTIVE CODING SCHEME

Consider a block B, of size N x N, of transform coefficients:

$$B = [B(i,j)] \qquad i,j = 1,\dots,N \qquad (5.1)$$

The variance of each transform coefficient B(i,j) can be computed with a set of typical blocks. This yields a variance matrix V:

$$V = [\sigma^2(i,j)] \qquad i,j = 1,\dots,N \qquad (5.2)$$

where $\sigma^2(i,j)$ is the variance of B(i,j).

In a zonal coding scheme, a bit pattern N is generated and transmitted to the receiver as overhead data [26]. In this case, the receiver knows the variance and bit patterns V and N before the actual image data is communicated. In a non-adaptive coding scheme, the bit pattern N is identical for all blocks. This bit pattern specifies the number of bits to be assigned to the codeword of each coefficient B(i.j). The bit pattern N is the bit matrix:

$$N = [N(i,j)] \qquad i,j = 1,\ldots,N \qquad (5.3)$$

where N(i,j) is the positive integer number which specifies the number of bits to be used in coding the transform coefficient B(i,j). Thus, the number of quantization levels to be used to quantize B(i,j) is $2^{N(i,j)}$.

If N(i,j) is equal to zero, the corresponding coefficient B(i,j) is not transmitted. Therefore, the bit pattern also specifies which coefficients are to be transmitted.

Given the bit assignment N(i,j), there are $2^{N(i,j)}$ quantization levels for B(i,j). The optimal placement of the quantization decision and reconstruction levels in order to minimize the mean-square reconstruction error is given by the Max quantizer [20]. The use of a Max quantizer requires a model for the probability density of each coefficient. For the LOT and the DCT, the probability density of the DC term (coefficient B(1,1)) was modeled as the uniform density. The probability density of the remaining coefficients was modeled as the Gaussian density [26]. For the SSFT, the probability density of the DC term was also modeled as the uniform density. For the remaining coefficients, the Rayleigh density provided a model for the magnitude, and the uniform density for the phase [14]. When $N(i,j)$ is different from zero, the appropriate Max quantizer, with $2^{N(i,j)}$ quantization levels, is used to quantize the coefficient $B(i,j)$.

When $N(i,j)$ is equal to zero, the corresponding coefficient $B(i,j)$ is simply discarded, and set by the receiver either to zero, or to the mean value if the mean is known. Consequently, for transform zonal coding, the mean-square error between the original and the coded images is equal to the sum of the mean-square quantization error on the transmitted coefficients and the mean-square energy of the discarded coefficients. This explains why the transform should be able to pack a maximum amount of the image energy into a minimum number of transform coefficients.

If d is the desired data rate for the coded image, the total number of bits available to code the image is $d\ R^2$. The total number of transform blocks is K^2 ($R = KN$). For a non-adaptive coding scheme, the total number of bits assigned to a block, N_B, is constant:

$$N_B = \frac{d\ R^2}{K^2} = d\ N^2 \qquad (5.4)$$

This yields the following constraint:

$$N_B = \sum_{i=1}^{N} \sum_{j=1}^{N} N(i,j) \qquad (5.5)$$

Subject to the constraint (5.5), the bit pattern N can be chosen to minimize the mean-square quantization error. For all the coding experiments described herein, the bit assignment N was set according to the log-variance relation suggested by Wintz [8], [12], [15], [36]:

$$N(i,j) = \frac{N_B}{N^2} + 2\log_{10}(\sigma^2(i,j)) - \frac{2}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\log_{10}(\sigma^2(i,j)) \quad (5.6)$$

Experimentally, N(i,j) is rounded off to the nearest integer value, or set to zero if the expression (5.6) is negative. Figure 7 illustrates a typical bit pattern for the LOT.

6	4	3	2	0	0	0	0
4	3	2	1	0	0	0	0
3	2	1	0	0	0	0	0
2	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 7 - Typical bit pattern for the LOT for the non-adaptive intraframe coding scheme.
Block size : 8; Data rate : 0.5 bpp.

5.3. ADAPTIVE CODING SCHEME

An adaptive coding scheme is presented in this section. Typical images usually can be divided into several local regions where the characteristics of the image content differ significantly (e.g., background, edges, high-detailed regions). Because of this non-stationarity of the image, the total number of bits available to code the image should not be assigned uniformly over the image [6], [10]. For example, fewer bits should be assigned to code a background region than a highly detailed region. This is especially important in the case of low bit-rate image coding where the total amount of data to be transmitted is limited.

In a transform coding system, each block in the transform domain reflects the local structure of the image in the space domain. This property makes block transform processes viable for adaptive image coding [37]. In the following adaptive transform coding scheme, each block is assigned to one class, from four possibilities, before coding. Within a class, or so-called category, the block quantizer is the same for each block.

Given a block B of N x N transform coefficients, the spectral energies which are contained in four fixed geometric regions of the block, are computed. The four regions which have been used, R_i, $i = 1,..,4$, are described in Figure 8.

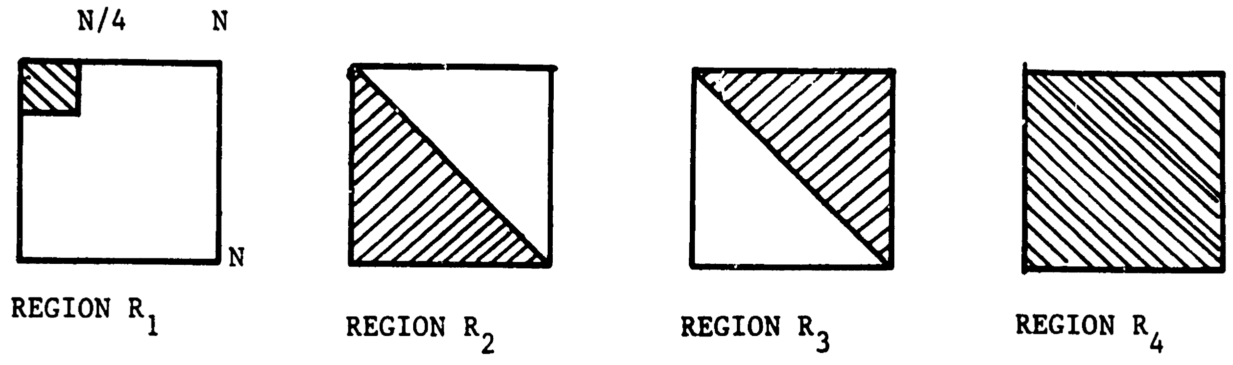


Figure 8 - Geometric description of the four regions used for the adaptive coding scheme.

The corresponding four spectral energies E_i, $i = 1,..,4$, are computed as follows:

$$E_i = \sum_{R_i} B^2(i,j) \tag{5.7}$$

To decide to which category the block B should be assigned, the two following ratios are calculated: E_1/E_4, and $|E_2 - E_3|/E_4$. The assignment procedure works in this order:

- Step 1: if $E_1/E_4 > \eta$, the block is assigned to category 1. Otherwise, move to step 2.
- Step 2: if $|E_2 - E_3|/E_4 < \gamma$, the block is assigned to category 4. Otherwise, see step 3.
- Step 3: if $E_2 > E_3$, the block is assigned to category 2. If $E_2 < E_3$, the block is assigned to category 3.

The parameters of η and γ are fixed for the image. These two parameters determine the repartition of the blocks per category for a given image. They are chosen experimentally so that the block repartition per category yields the best coded image quality.

For the standard "head and shoulder" images, which were used in the coding experiments (see section 5.4), about thirty percent of the blocks fell in category 1, fifteen percent in each of categories 2 and 3, and forty percent in category 4. This repartition closely parallels the nature of the image: typically, blocks in category 1 correspond to background regions, blocks in categories 2 and 3 to edge regions, and blocks in category 4 to highly detailed regions (facial features).

For each category, a variance matrix is computed, and the sum of the log-variance values is determined. Given these statistics and the repartition per category of the blocks, the bit assignment per category is determined using the log-variance rule extended to data blocks. Although the total number of bits assigned to the coding of a block is constant within each category, it varies from one category to another. Fewer bits are assigned to a block in category 1 than to a block in one of the other categories. About the same number of bits is assigned to a block in categories 2, 3, and 4. For each category, the bit pattern is determined using the log-variance rule (5.6). The coding scheme is identical to the non-adaptive scheme described in section 5.2 for each category. For each block, a code word identifying the category is sent ahead to the receiver before the transmission of the coded transform coefficients. Figure 9 illustrates typical LOT derived bit patterns for a block in each one of the four categories.

3	2	1	0	0	0	0	0
2	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Category 1

4	2	1	0	0	0	0	0
4	3	1	0	0	0	0	0
4	3	2	0	0	0	0	0
4	4	1	0	0	0	0	0
2	2	1	0	0	0	0	0
2	2	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Category 2

4	5	4	4	3	2	1	1
1	3	3	3	1	1	0	0
0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Category 3

5	3	2	1	0	0	0	0
3	3	3	1	1	0	0	0
2	2	2	1	1	0	0	0
2	3	2	1	0	0	0	0
1	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Category 4

Figure 9 - Typical bit patterns for the LOT for the adaptive intraframe coding scheme.
Block size : 8; Data rate : 0.5 bpp.

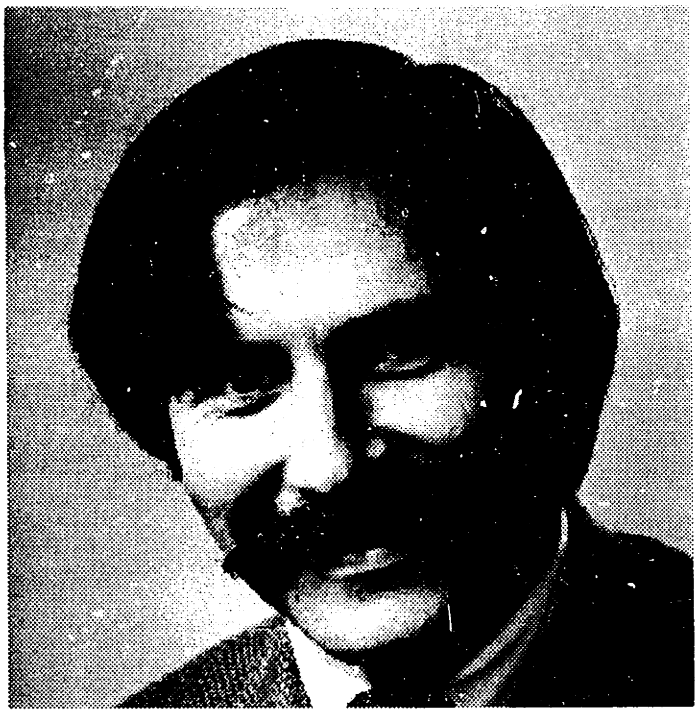
5.4. EXPERIMENTAL RESULTS

Two images were used for the intraframe coding experiments: "HIRES" and "BISWEX." Both images are of resolution 256 x 256 pixels. They are monochrome (black and white), and are originally coded at a data rate of 8 bits per pixel (8 bpp). That is, 256 gray levels are used to quantize the luminance value of each pixel. "HIRES" is a typical "head and shoulders" image. "BISWEX" represents about one half of a face; it was extracted from an image of higher resolution. Figure 10 illustrates these two original images.

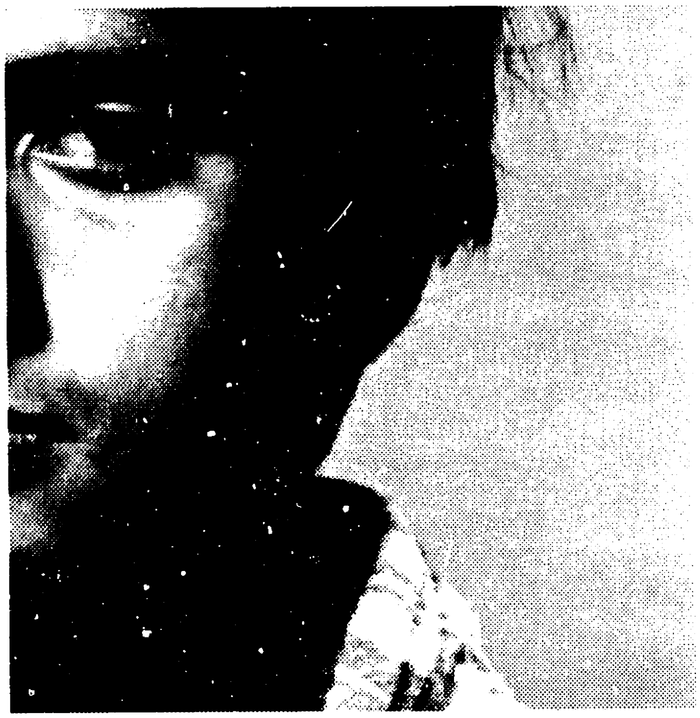
These two images were coded at low bit-rates, by applying the DCT, the LOT, and the SSFT, and by using the non-adaptive and adaptive coding schemes described in sections 5.3 and 5.4. For all these transforms, the block size was equal to 16 pixels. For specifically the LOT, the number of overlapping samples L was equal to 8. In other words, a block of 16 x 16 LOT coefficients was computed from a block of 32 x 32 data points.

The computation of the normalized signal-to-noise ratio (SNR) between the original and coded images measured the quality of the coded image. The SNR is defined as follows:

$$\text{SNR} = -10 \log_{10} \frac{\sum_{i=1}^{R} \sum_{j=1}^{R} (F(i,j) - \hat{F}(i,j))^2}{\sum_{i=1}^{R} \sum_{j=1}^{R} F^2(i,j)} \quad (\text{dB}) \qquad (5.8)$$



Original "HIRES"



Original "BISWEX"

Figure 10 - The original images of resolution 256 x 256, 8 bpp.

where F is the original array of resolution R x R, and $\hat{F}$ is the coded array.

The adaptive coding scheme performed better than the non-adaptive one in terms of the resulting SNR of the coded image. Also, the objective quality of the coded image was improved when the adaptive coding scheme was used. Table 4 compares the signal-to-noise ratios of the image "HIRES" transform coded with the DCT, LOT, and SSFT, using both adaptive and non-adaptive coding schemes, at a data rate of 0.32 bpp.

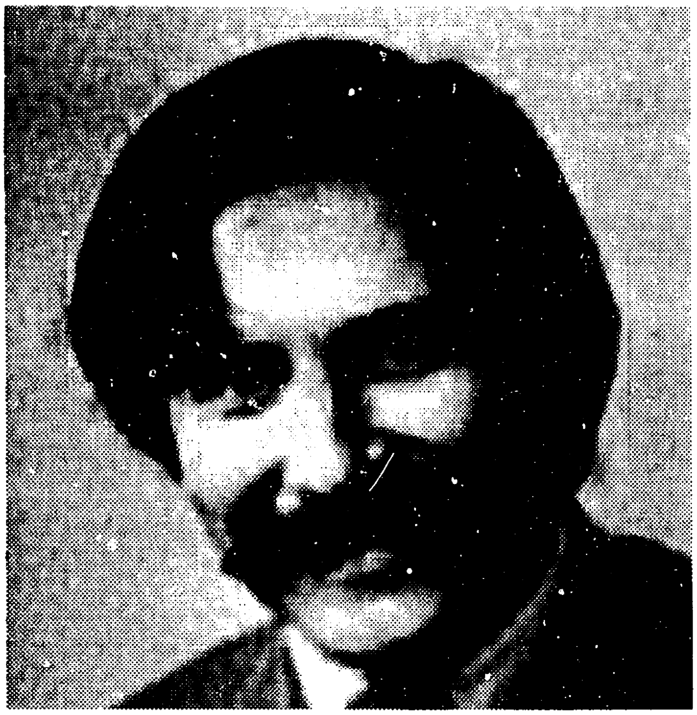
Table 4: SNR (in dB) of "HIRES" coded at a data rate d = 0.32 bpp

coding scheme / transform	non-adaptive	adaptive
DCT	20.8	22.5
LOT	20.9	22.5
SSFT	21.6	21.8

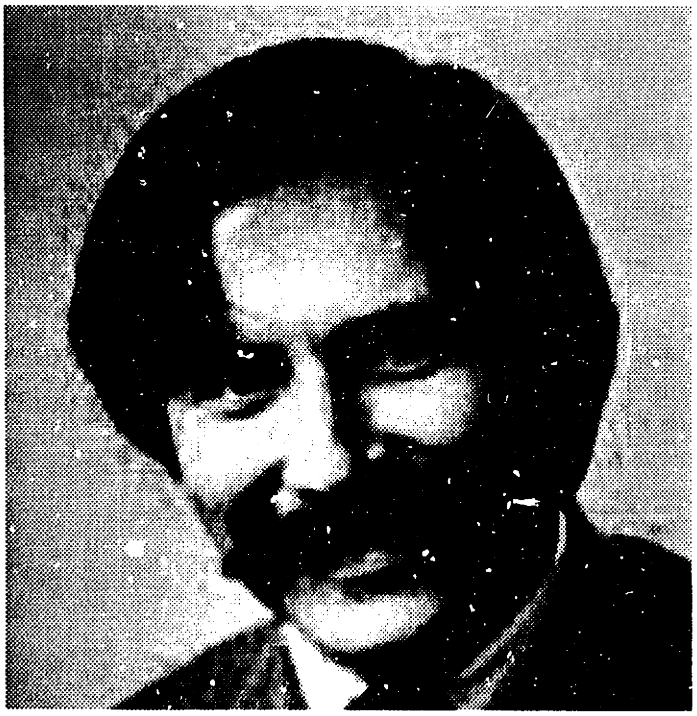
Figure 11 illustrates the improvement of the subjective quality of the coded image when the adaptive scheme is used.

All the remaining coded images in this section have been coded with the exclusive use of the adaptive scheme. Figures 12 and 13 compare the performances of the DCT and the LOT, while Figures 14 and 15 compare the performances of the SSFT and the LOT.

Indeed, the LOT reduces the blocking effects generated by the DCT. In particular, mismatches that appear with the DCT between

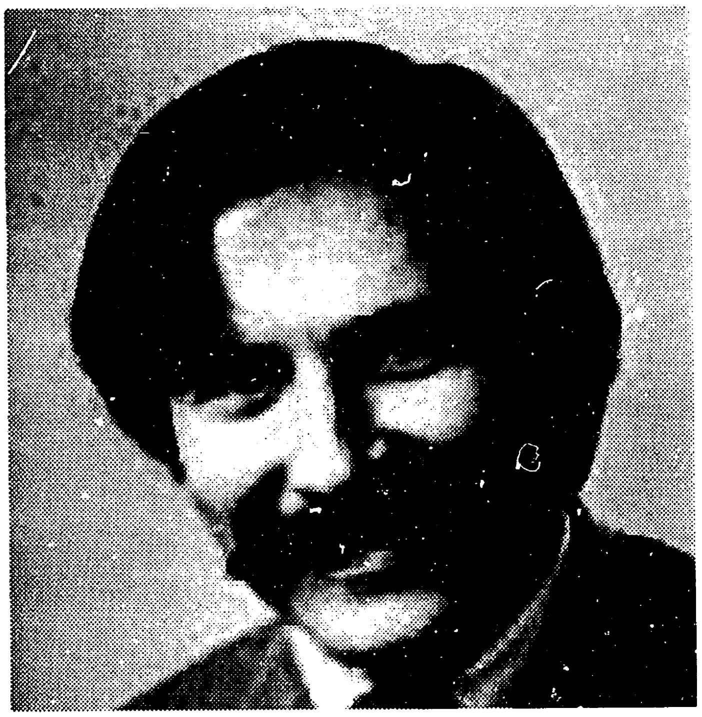


Non-adaptive coding scheme SNR = 20.9 dB



Adaptive coding scheme SNR = 22.5 dB

Figure 11 - "HIRES" coded at 0.32 bpp using the LOT (original: 256 x 256, 8 bpp).


DCT, SNR = 22.5 dB

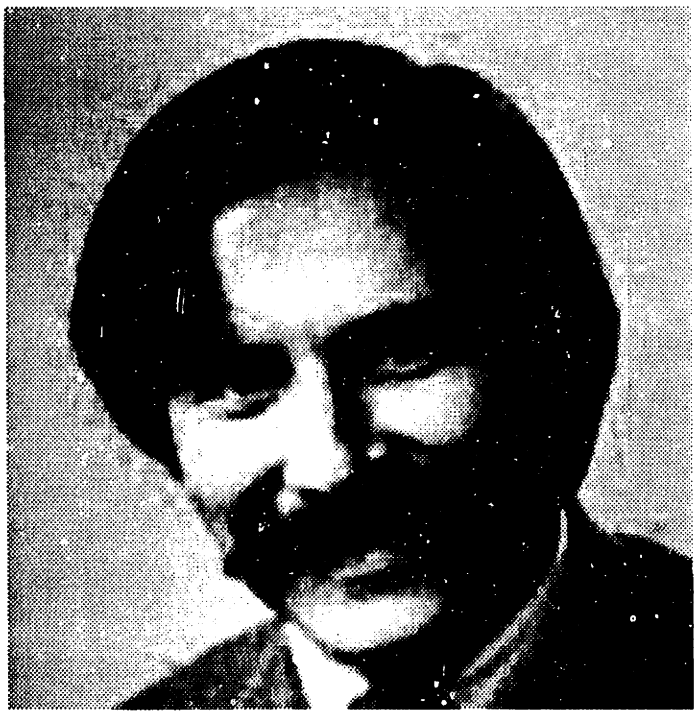
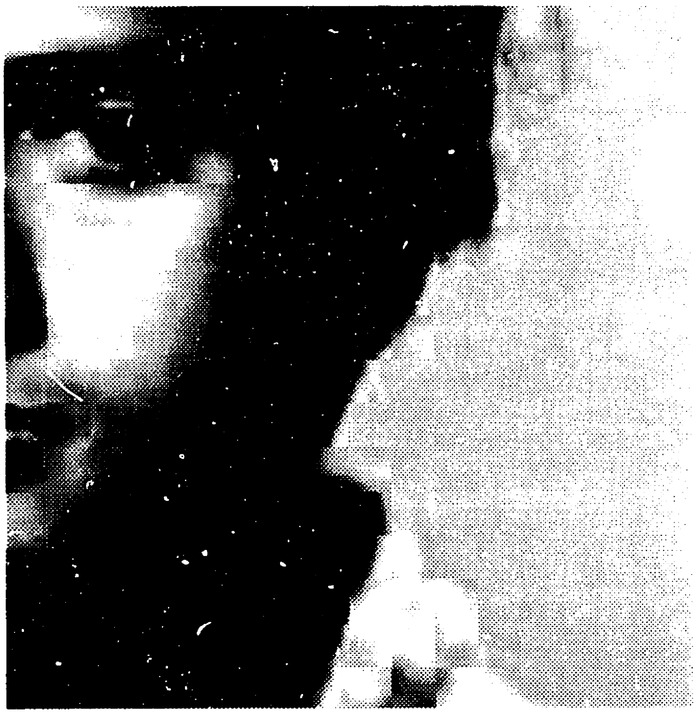
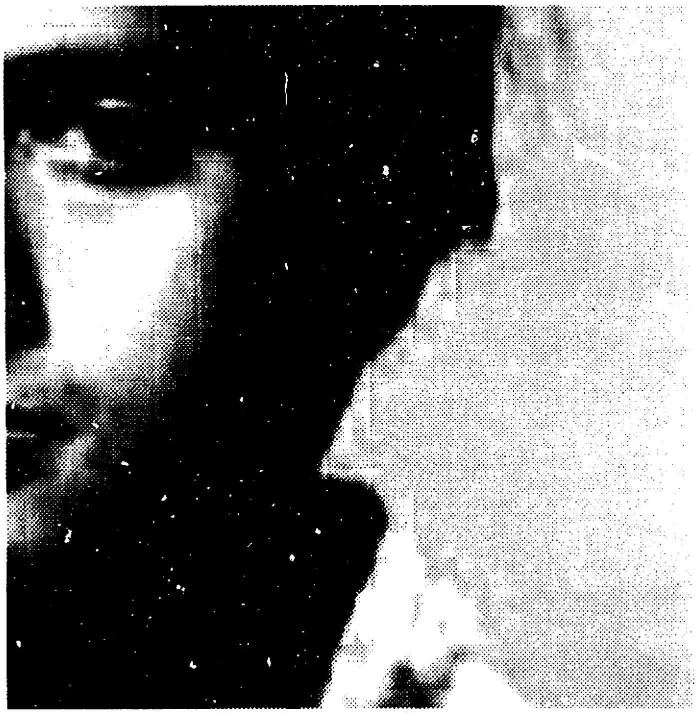

LOT, SNR = 22.5 dB

Figure 12 - "HIRES" adaptively coded at 0.32 bpp (original: 256 x 256, 8 bpp).

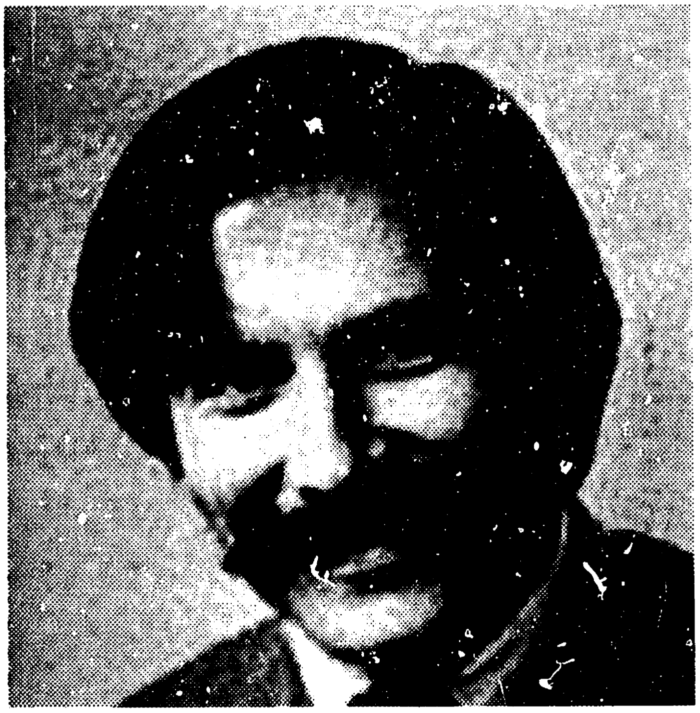


DCT, SNR = 23.8 dB



LOT, SNR = 24.3 dB

Figure 13 - "BISWEX" adaptively coded at 0.10 bpp (original: 256 x 256, 8 bpp).


SSFT, SNR = 21.8 dB

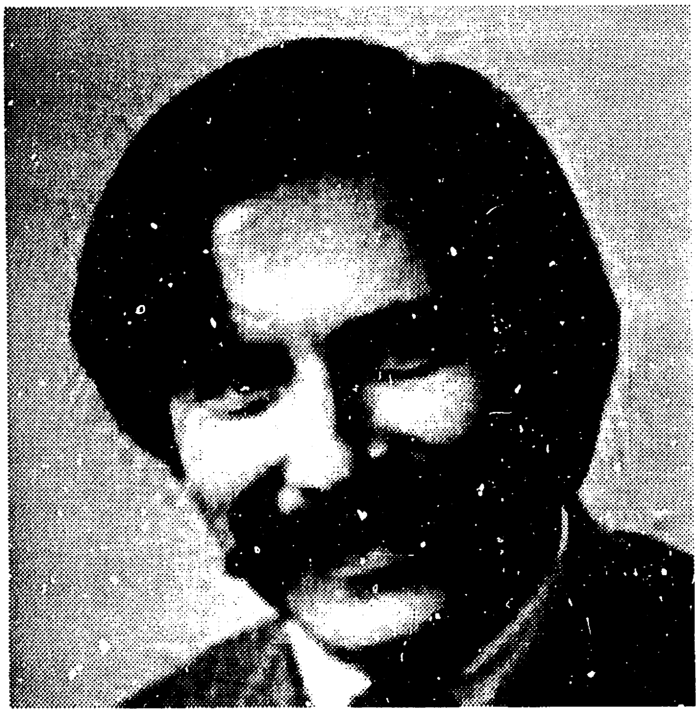
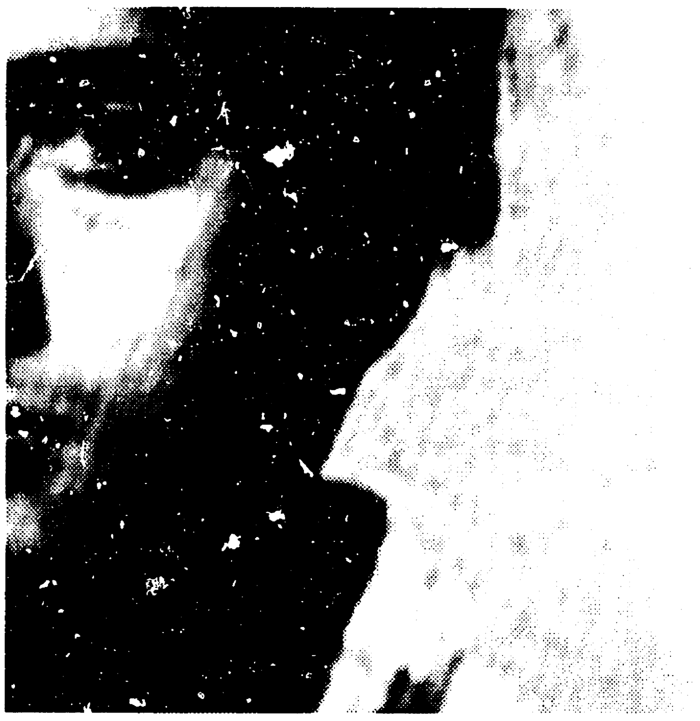
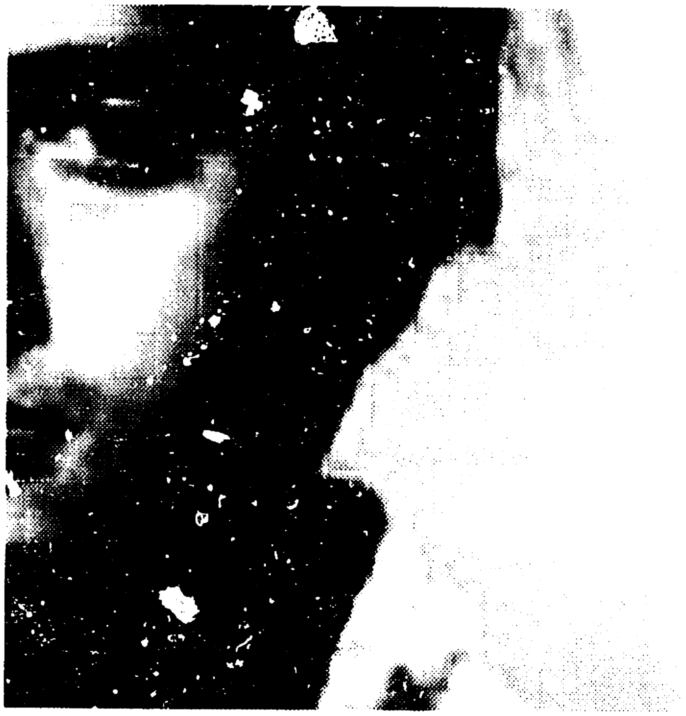

LOT, SNR = 22.5 dB

Figure 14 - "HIRES" adaptively coded at 0.32 bpp (original:256 x 256, 8 bpp).



SSFT, SNR = 23.4 dB



LOT, SNR = 24.3 dB

Figure 15 - "BISWEX" adaptively coded at 0.10 bpp (original: 256 x 256, 8 bpp).

adjacent blocks in highly detailed regions disappear with the LOT. However, the boundaries between blocks are still visible with the LOT. This can be explained by the boundary discontinuities of the low-order LOT basis functions. These discontinuities generate a boundary effect which makes the blocks visible, but this artifact is different from the DCT's blocking effects. The LOT does not produce any mismatches between adjacent blocks. Since blocking effects are only local artifacts and do not increase the background noise level, the DCT and the LOT perform equally well in terms of signal-to-noise ratios of the coded images.

Additionally, the LOT reduces the SSFT's ringing effects. The ringing effects generated by the LOT around an edge spread over only half a block (about 8 pixels), which happens to correspond to the value of the overlap L. The LOT also performs better than the SSFT in terms of signal-to-noise ratio of the coded images because the ringing effects generated by the SSFT are global artifacts which affect every region of the image. Therefore, these increase the background noise level. No blocking effects, however, appear with the SSFT.

In summary, ringing effects, which are not generated by the DCT, spread over half of a block with the LOT, and over the entire image with the SSFT. While the latter does not produce any blocking effects, those generated by the DCT are highly visible. The LOT is able to reduce blocking effects to boundary effects. In any case, though, the LOT improved the subjective quality of the coded images.

Images coded at the same data rate with the DCT, the LOT, and the SSFT, were shown to several groups of viewers (usually 2 or 3 persons at a time). Most viewers agreed that the LOT coded images had the best overall subjective quality. To all viewers, the LOT coded images appeared less noisy than the SSFT coded ones. They also recognized that the LOT's blocking effects were less visible and preferrable to those of the DCT. However, one of five viewers thoughtthat the quality of the DCT and LOT coded images appeared close.

CHAPTER SIX

APPLICATION OF THE LOT TO INTERFRAME CODING

6.1. INTRODUCTION

Interframe coding, which refers to coding sequences of closely related images, can also be viewed as the coding of three-dimensional signals. A third dimension, the time axis, is added to the two dimensions of the space domain. For typical video sequences, the temporal correlation between successive frames is strong because these frames are often similar. Because of the high correlation in both the spatial and temporal domains, efficient interframe coding can be achieved [5], [26], [41].

Since the efficiency of transform coding has been demonstrated for intraframe coding, one approach to interframe coding is to extend the block transform coding techniques to three-dimensional signals. This scheme was experimented using the DCT, and it provided excellent coding efficiency [33]. However, this technique is not practical for two reasons. First, a block transform coding system involves the coding of three-dimensional blocks, which means that all the frames corresponding to the width of one block in the time dimension must be stored simultaneously. Therefore, the system must have large storage capabilities which increase its cost. Second, the transmission of the coded signal is likely to introduce a delay which is not acceptable for real-time applications (e.g., videophone systems).

However, hybrid transform/DPCM coding systems provide an alternative solution to the interframe coding problem. The idea is to design a hybrid system by combining two different coding schemes: transform coding and DPCM coding. Transform coding is used in the spacial domain, and DPCM coding in the temporal domain [5], [32], [34], [43]. Since a DPCM coding scheme is used along the time axis, the coding of each frame requires only the knowledge of the previous frame. Consequently, just one frame at a time needs to be stored, and both transmission delays and storage requirements are decreased significantly. A hybrid transform/DPCM coder involves a linear predictor which estimates each frame based on its predecessor [19], [32]. The coder described in section 6.2 involves a fixed predictor equal to the identity. In other words, the predictor assumes there is no motion between successive frames. The encoder described in section 6.3 includes the motion estimation between successive frames. Thus, a motion compensated predictor provides a much more accurate estimate of each frame [13], [18].

6.2 HYBRID TRANSFORM/DPCM CODING SCHEME

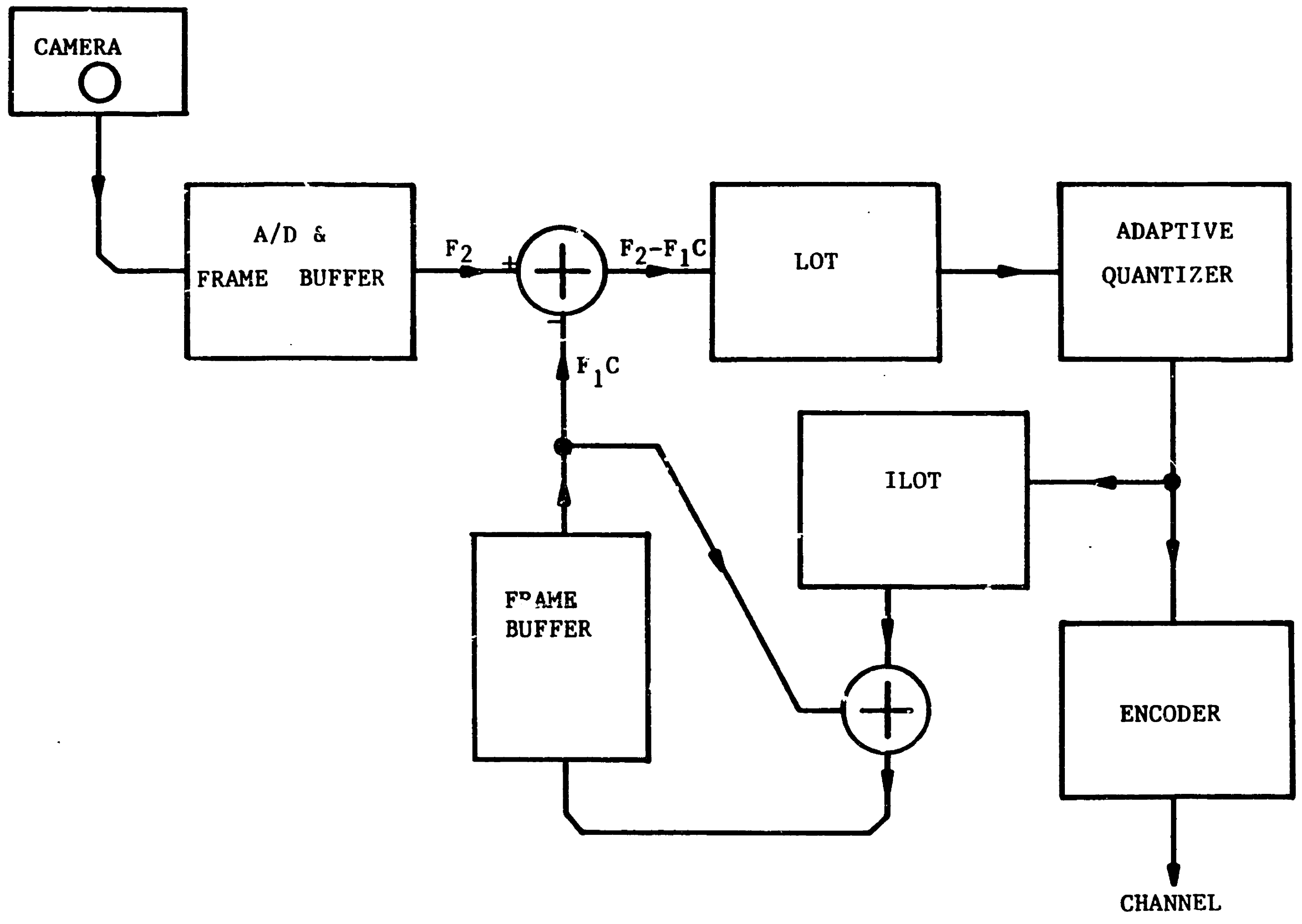
A simple hybrid transform/DPCM coding system is described in this section. In the following section, a higher performance system will be presented. Because of the similarities between the two systems, the same notations are to be assumed. The frame previously coded and transmitted is called F_1 and its coded version F_{1c}. The next frame to be transmitted is called F_2 and its coded version F_{2c}. In a hybrid transform/DPCM coding system, a linear predictor provides an estimate of F_2 called $\hat{F}_2$. The frame $\hat{F}_2$ is estimated from the frame F_{1c}. Since the necessary data F_{1c} is available to the transmitter and to the receiver, the estimation process can be carried out by the pair.

Block diagrams for the transmitter and the receiver of the transform/DPCM coding system are given in Figures 16 and 17. For this system, $\hat{F}_2$ is simply equal to the previously transmitted frame F_{1c}. Thus, the DPCM signal $F_2 - \hat{F}_2$ reduces to the difference between the two successive frames $F_2 - F_{1c}$. This DPCM signal, also called the error signal, is transform coded. The transforms used for the experiments were the LOT and the DCT, with a block size equal to 8 pixels (the sequence was of resolution 128 x 128 pixels). Since the SSFT has been shown to perform poorly in interframe coding, it has not been tested in these experiments. At the receiver, the inverse transform is taken to obtain the reconstructed DPCM signal. This is then added to the previously transmitted frame F_{1c} to generate the new coded frame F_{2c}.

To achieve low data-rate interframe coding with a hybrid

FIGURE 16

THE TRANSMITTER OF THE HYBRID LOT/DPCM CODING SYSTEM



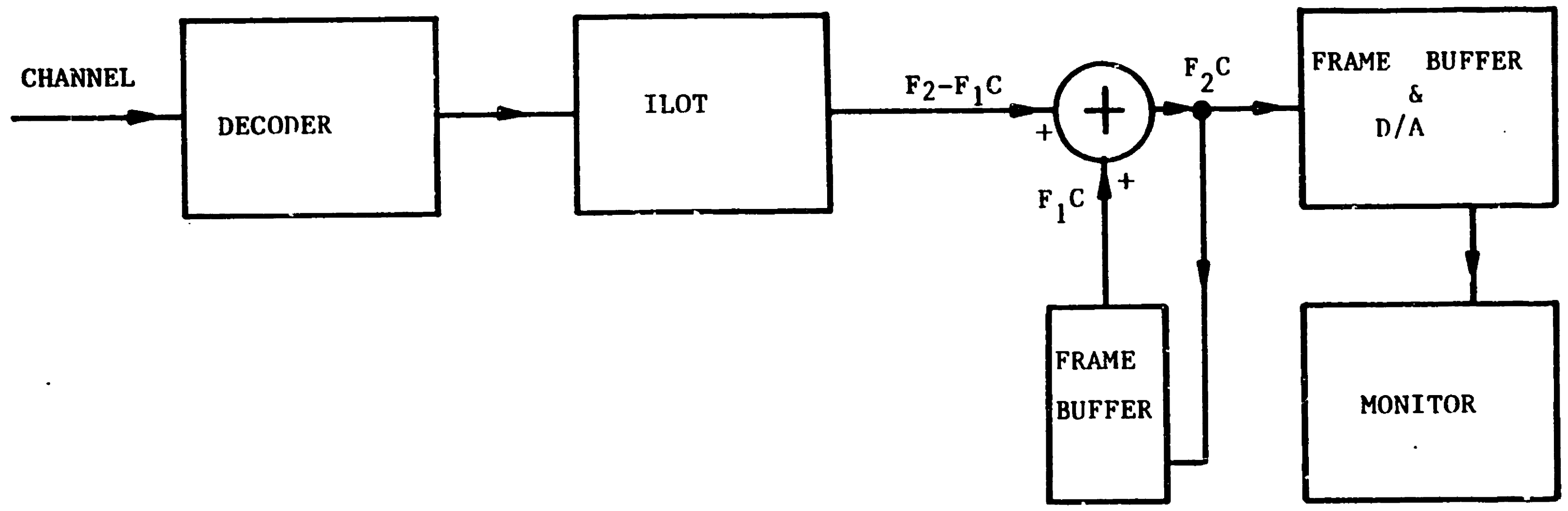


FIGURE 17

THE RECEIVER OF THE HYBRID LOT/DPCM CODING SYSTEM transform/DPCM coding system, not all the transform blocks of the DPCM signal should be communicated to the receiver. In actuality, since successive frames often change only slightly in some local regions of the image, some transform blocks need not be transmitted. Depending on the desired number of transmitted blocks, a threshold is computed for each error signal. The error energy inside each block is computed and compared to this threshold. Blocks which correspond to an error energy larger than the threshold are coded and transmitted; blocks which correspond to an error energy smaller than the threshold are not. A code word identifying the non-transmitted blocks is sent to the receiver which assumes zero error for these blocks. The transmitted blocks are quantized adaptively. The adaptive coding scheme used is identical to the one described in section 5.3. The transmitted blocks are assigned to one of four categories. For each category, a bit pattern is derived using the log-variance rule (5.6). Each transform coefficient is quantized using a Gaussian Max quantizer.

For the interframe coding experiments described in section 6.4, it was desired to maintain a constant data-rate. Consequently, a fixed percentage of blocks for each DPCM signal are transmitted. The repartition of these blocks is fixed for each category. First, each block is assigned to its respective category. Then, for each category, a threshold is computed so that the total number of blocks having error energy greater than the threshold corresponds to the a-priori fixed repartition for the category. Within each category, only the blocks with energy above the category threshold are transmitted.

Because the total error energy may vary widely between different parts of the sequence, transform coefficients of each DPCM signal must be scaled so that their variances are uniform from frame to frame. The average error energy is computed over each DPCM signal. Each transform coefficient is scaled by the squared root of the average energy. This number is coded for each frame, and sent to the receiver as overhead data so that the decoder can recover the original transform coefficients.

6.3. MOTION COMPENSATED CODING SCHEME

A description of a transform/motion compensation coding system is given in this section. This system produces an accurate estimation of F_2 by using motion compensation. Thus, the energy in the DPCM signal is much lower than the error energy in the system described in section 6.2. Consequently, the coding of the DPCM signal is far more efficient.

The block diagram of the transmitter and the receiver for a transform/motion compensation coding system are given in Figures 18 and 19. The coding procedure of the DPCM signal $F_2 - \hat{F}_2$ is identical to the one used in the hybrid transform/DPCM system. The main difference between these two systems lies in the predictor. The motion estimation procedure used was developed by Hinman [13].

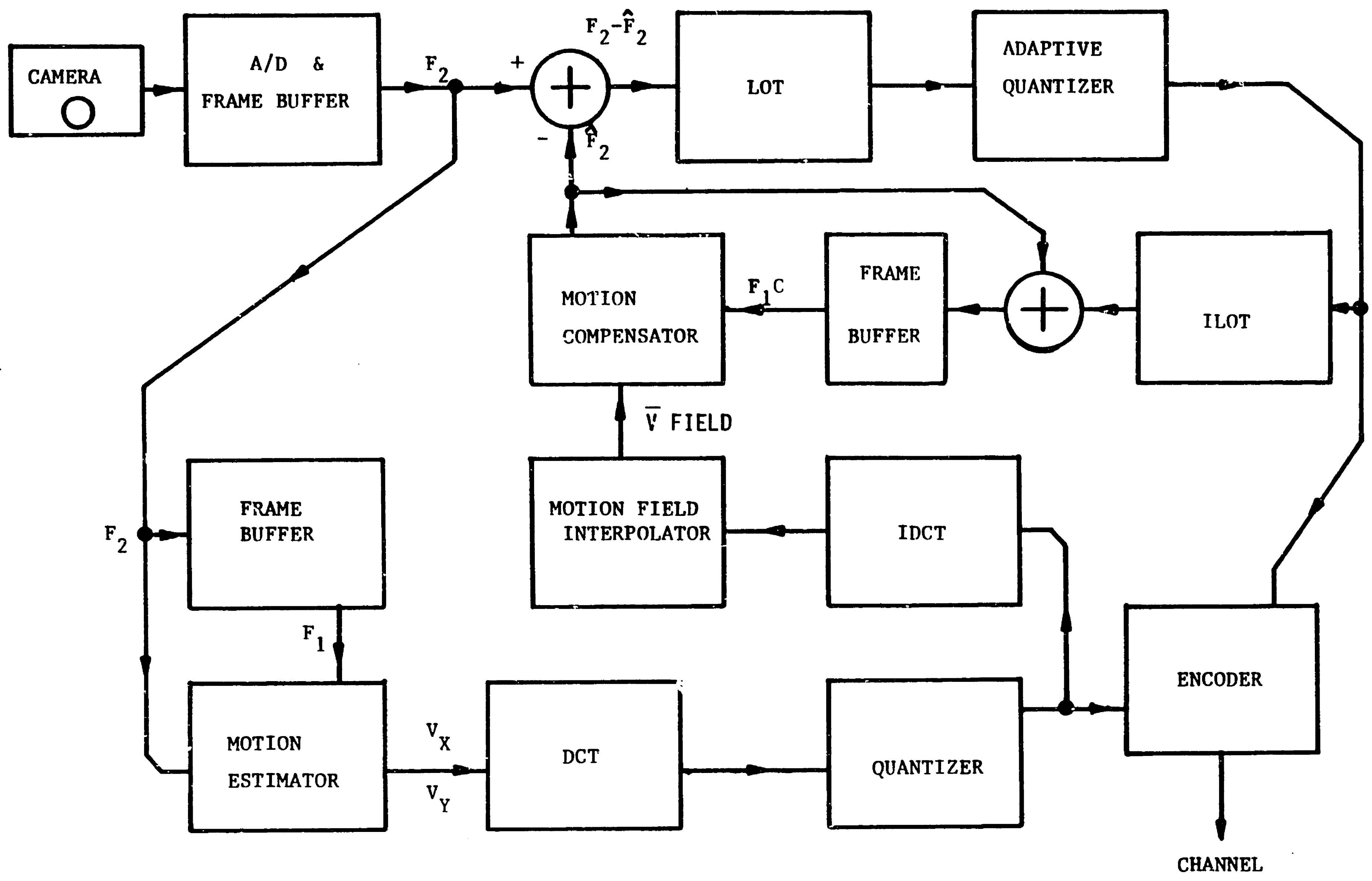


FIGURE 18

THE TRANSMITTER OF THE LOT/MOTION COMPENSATION CODING SYSTEM

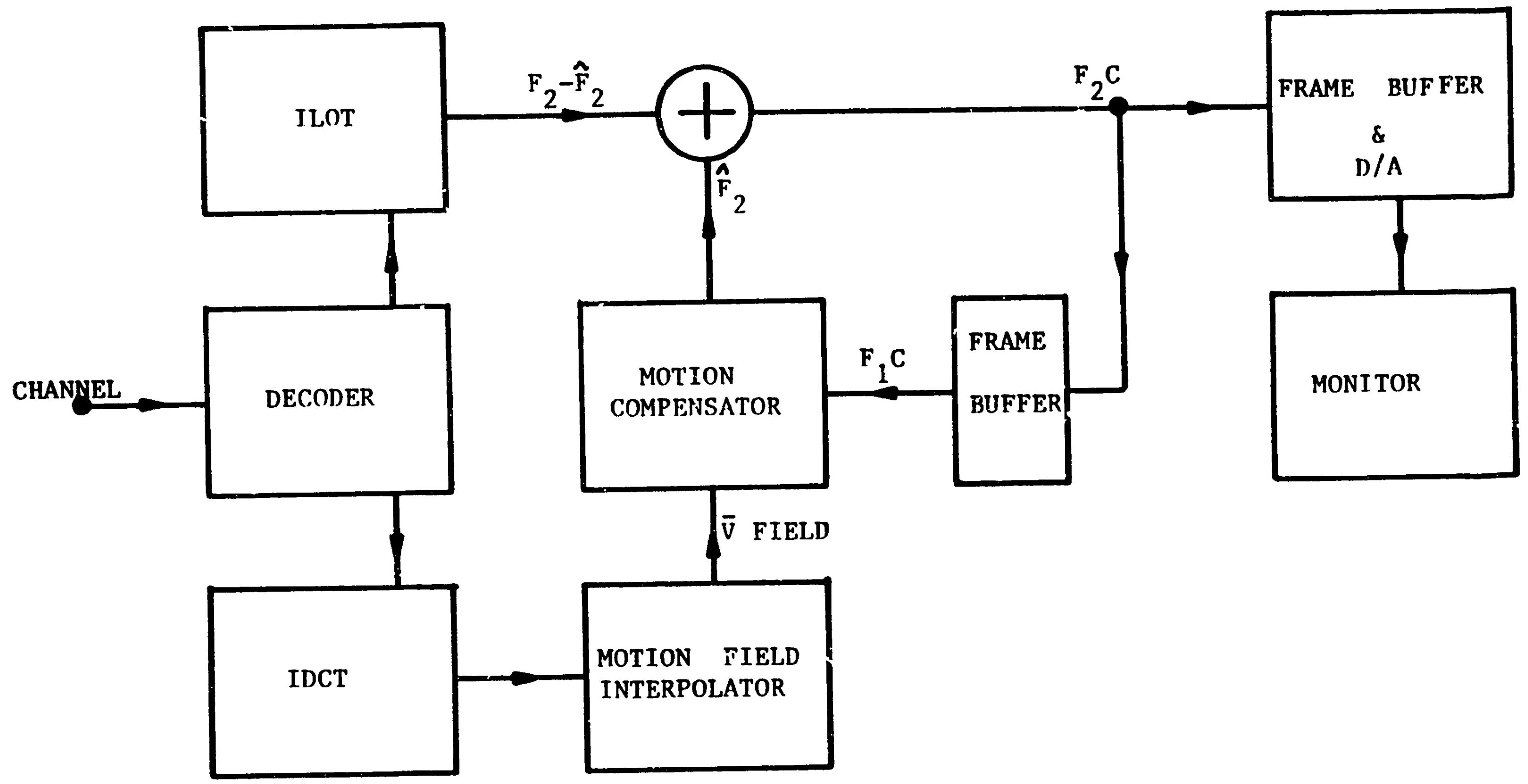


FIGURE 19

THE RECEIVER OF THE LOT / MOTION COMPENSATION SYSTEM

The two original frames F_1 and F_2 are divided into blocks (of size 4 x 4 for images of resolution 128 x 128). For each block, a motion vector with two components is estimated. A subsampled motion vector field, whose two components are V_x and V_y, is generated for the entire image. Using a raised-cosine interpolation filter, the full-size motion vector field $\bar{V}$ FIELD is acquired [13]. The motion-compensated estimate $\hat{F}_2$ is obtained by projecting each pixel of F_1 in the direction of motion. The motion vector field is estimated so that the mean-square error between F_2 and $\hat{F}_2$ is minimal. The adaptive steepest-descent minimization algorithm developed by Hinman was used to estimate the motion vector field [13]. The predictor in this system is a motion compensator. The components V_x and V_y are estimated from the original data F_1 and F_2. Using motion compensation, the estimate $\hat{F}_2$ is produced from the previously transmitted frame F_{1c}.

The receiver must be able to estimate F_2 also. Therefore, the motion vector field components V_x and V_y must be quantized, coded and communicated to the receiver. The receiver structure includes the motion compensator which produces $\hat{F}_2$ (cf. Figure 19).

Because motion vectors from adjacent blocks are highly correlated, a transform coding scheme can be used to efficiently encode the motion vector field [13]. The two components V_x and V_y are transform coded using the DCT (of block size 8 x 8). Two bit arrangements are used. Each transform block is assigned to either a high-energy category or a low-energy category. Half of the blocks are assigned to each class. The quantized motion vector field is coded and transmitted to the receiver as overhead information.

Although the system is more complex than the hybrid transform/DPCM coding system, the transform/motion compensation coding system is more efficient since it uses optimally the available information previously communicated to the receiver. Natural images closely parallel the motion compensation model; therefore, the expense of sending the motion vectors is outweighed by the improved performance of the DPCM coder.

6.4. EXPERIMENTAL RESULTS

The "MAN" video sequence served as test data for the interframe coding experiments. "MAN" is a "head and shoulder" type of sequence of 150 frames of resolution 128 x 128 pixels, 8 bits per pixel (luminance), originally recorded at 15 frames per second. This sequence was coded at a constant data rate of 56 kilobits per second (kbps) using the following four simulated architectures:

1. DCT/DPCM coding system
2. LOT/DPCM coding system
3. DCT/MC coding system (motion compensated)
4. LOT/MC coding system

The early portion of the "MAN" sequence contains mostly facial activity. The latter portion contains a large amount of head motion.

As with all architectures, only alternate frames were coded and transmitted. Thus, the actual frame rate was reduced to 7.5 frames per second. To increase the frame rate to the original 15 frames per second, the intermediate frames are interpolated at the receiver. For a DPCM coder, they are reconstructed using linear interpolation between two successfully transmitted frames. For a motion compensated coder, these frames are reconstructed using motion interpolation. Namely, each pixel of the previously transmitted frame is projected forward halfway in the direction of motion. In this manner, the receiver performs smooth frame interpolation by using the transmitted motion vector field information. Motion interpolation gives better results than linear interpolation because it avoids the "doubling" effects of the latter.

For a sequence coded at 56 kbps and at a frame rate of 7.5 frames per second, a total of 7465 bits is assigned to one coded frame (0.46 bit per pixel). For the DCT-LOT/DPCM encoders, all the available data is used for the coding of the DPCM signal. For the DCT-LOT/MC encoders, a constant rate of twenty-two percent of the available data was used for the coding of the motion vector field.

In the coding of the DPCM signal, fifty percent of the blocks were quantized and transmitted. Among these updated blocks, fifty percent were assigned to category 1, fifteen percent to category 2 and 3, and twenty percent to category 4. The category assignment was completed as described in section 6.2. The total number of bits assigned to a block in one of the categories 2, 3, and 4, was about three times the number of bits assigned to category 1. Figure 20 illustrates the bit pattern for each category for the LOT/DPCM coding system (at 56 kbps).

In the coding of the motion vector field (cf. section 6.3), half of the blocks are assigned to both the low-energy and high-energy categories. Three times as many bits were assigned to a high-energy block than to a low-energy one. Those various category and bit assignments were derived from the sequence characteristics, and yielded satisfactory experimental results. However, no extensive study of the various possible coding procedures was made. Rather, a reasonable scheme was adopted. The DCT and the LOT were compared under exactly identical coding procedures.

The signal-to-noise ratios of the DCT/DPCM and LOT/DPCM coded sequences are plotted in Figures 21 and 22 as a function of the frame number. These two figures illustrate the almost identical performance of the DCT and the LOT in terms of mean-square quantization error. Sharp variations of the SNR (occuring around frames 10, 70, and 110) are due to large variations in the amount of motion. To quantify the amount of motion in the sequence, the energy of the difference signal (DPCM) between successive frames was computed. Figure 23 illustrates the variations in the amount of motion in the "MAN" sequence. Whenever there is a sharp increase

5	4	3	1	0	0	0	0
4	3	2	1	0	0	0	0
3	2	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Category 1

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

Category 2

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

Category 3

6	4	3	2	1	1	1	1
4	4	4	2	1	1	0	1
3	3	3	2	2	0	1	1
3	4	3	2	1	1	1	0
2	2	2	2	1	1	0	0
2	1	1	1	0	0	0	0
1	1	1	0	0	0	0	0
0	1	1	0	0	0	0	0

Category 4

Figure 20 - Bit patterns used for the LOT/DPCM coding system at 56 kbps.

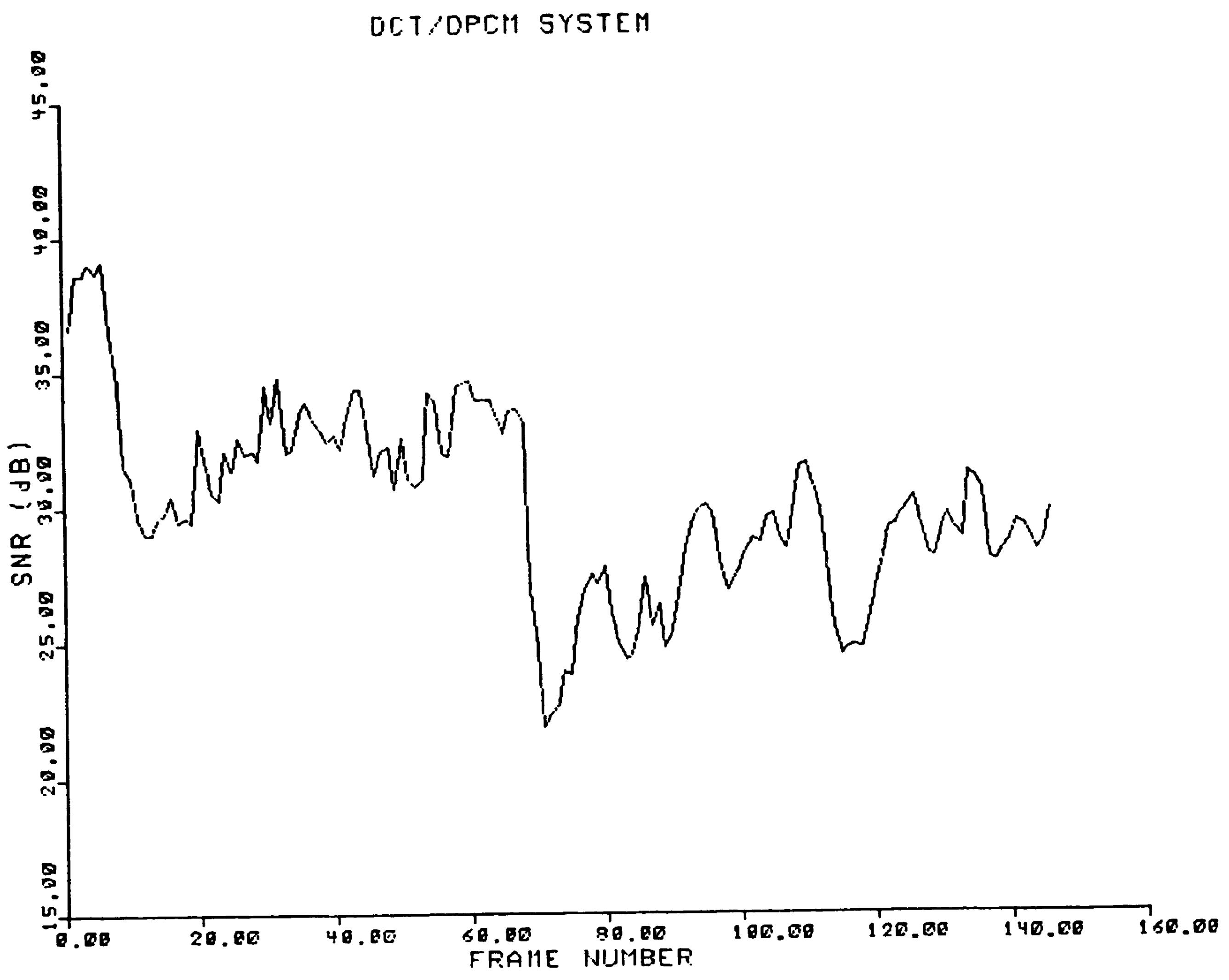


Figure 21 - SNR versus frame number for the DCT/DPCM coded sequence.

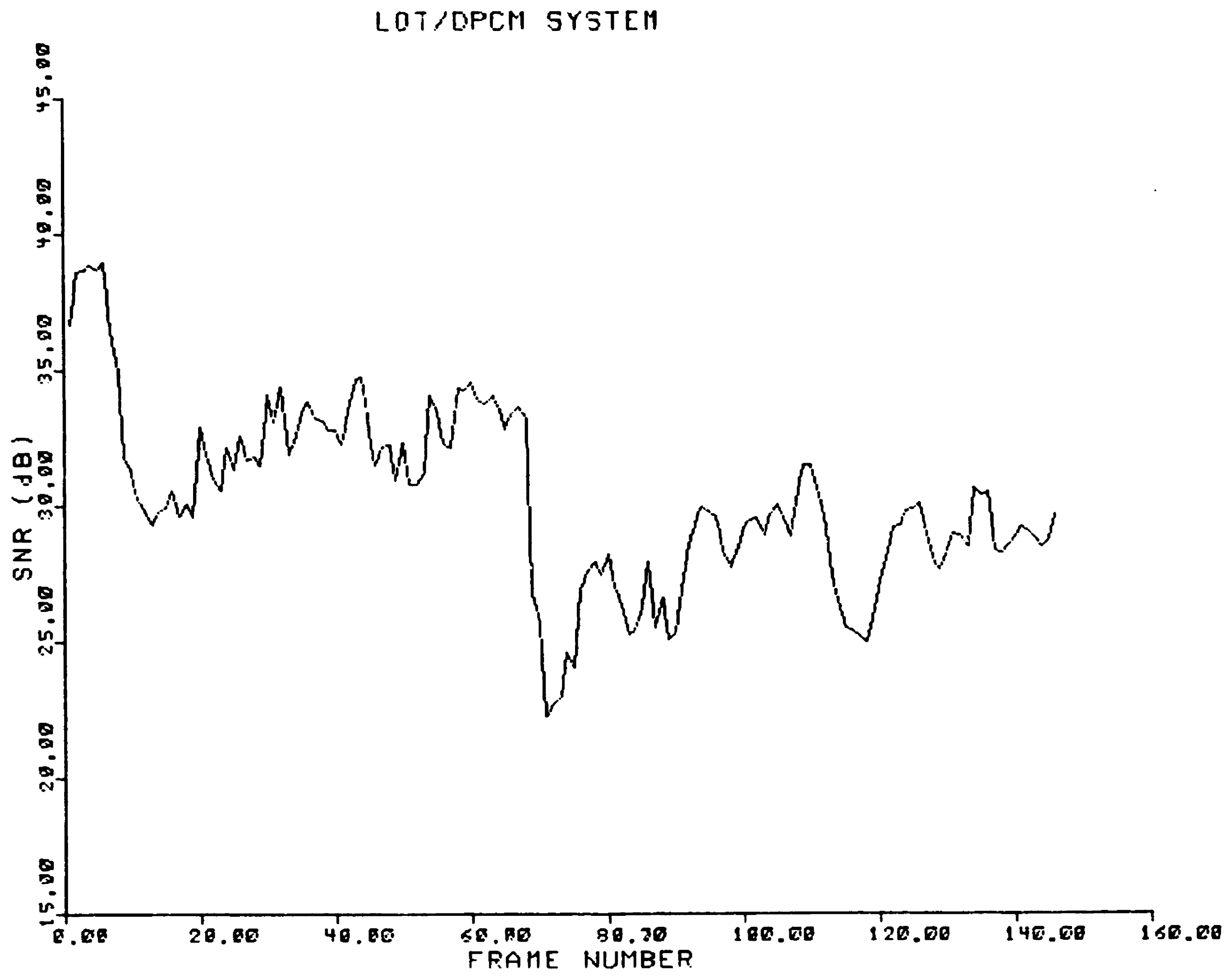


Figure 22 - SNR versus frame number for the LOT/DPCM coded sequence.

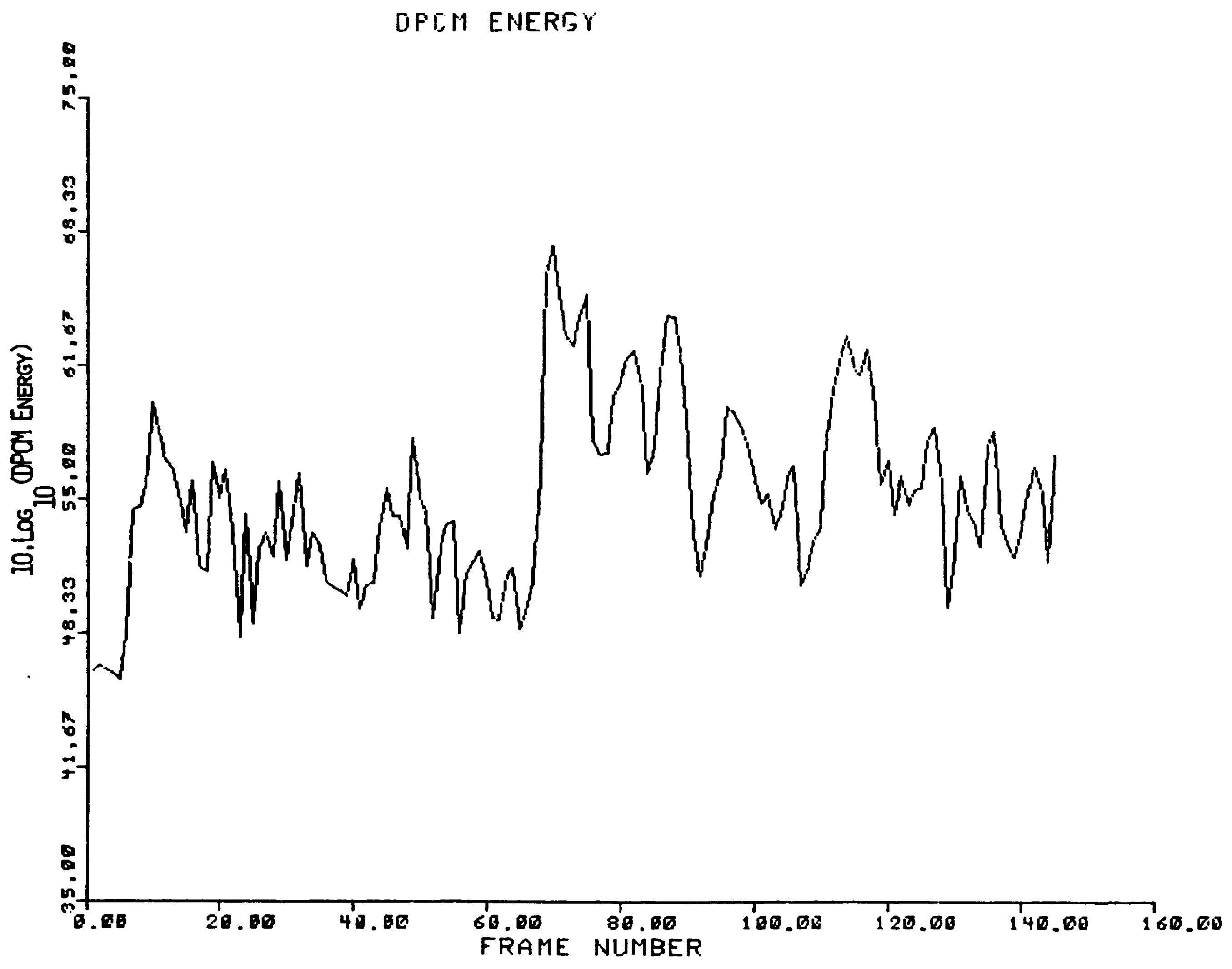


Figure 23 - The DPCM log-energy of the "MAN" sequence versus frame number.

in the amount of motion, there is a corresponding decrease in SNR. A comparison between SNR and DPCM energy is given in Figure 24. The higher the DPCM energy is, the lower the signal-to-noise ratio is. A least-square fit to a straight line was computed from the data. The resulting line has been added to Figure 24.

The signal-to-noise ratios of the DCT/MC and LOT/MC coded sequences are plotted in Figures 25 and 26 as a function of the frame number. As for the DPCM systems, the mean-square error performances of these two systems are almost identical. Figure 27 compares the SNR's between the LOT/MC and DCT/MC coded sequences. All the data points are clustered along the diagonal axis. Only a slightly better performance by the LOT is noticeable. Specifically, the least-square fit to the data shows that the LOT yields an SNR about 0.5 dB higher, on the average, than the DCT.

Figure 28 compares the SNR's between the DCT/MC and the DCT/DPCM coded sequences. This figure illustrates the mean-square error performance improvement when motion compensation is used. For high signal to noise ratios (SNR > 34 dB), DPCM and motion compensated systems perform equally well. In these cases, there is little motion between successive frames which renders the use of motion compensation unnecessary. Except for this special case, the improvement due to motion compensation is very significant, reaching about 9 dB for several frames. The least-square fit to a straight line, along with the average improvements in SNR, have been added to Figure 28 (the diagonal $y = x$ represents the equi-performance line).

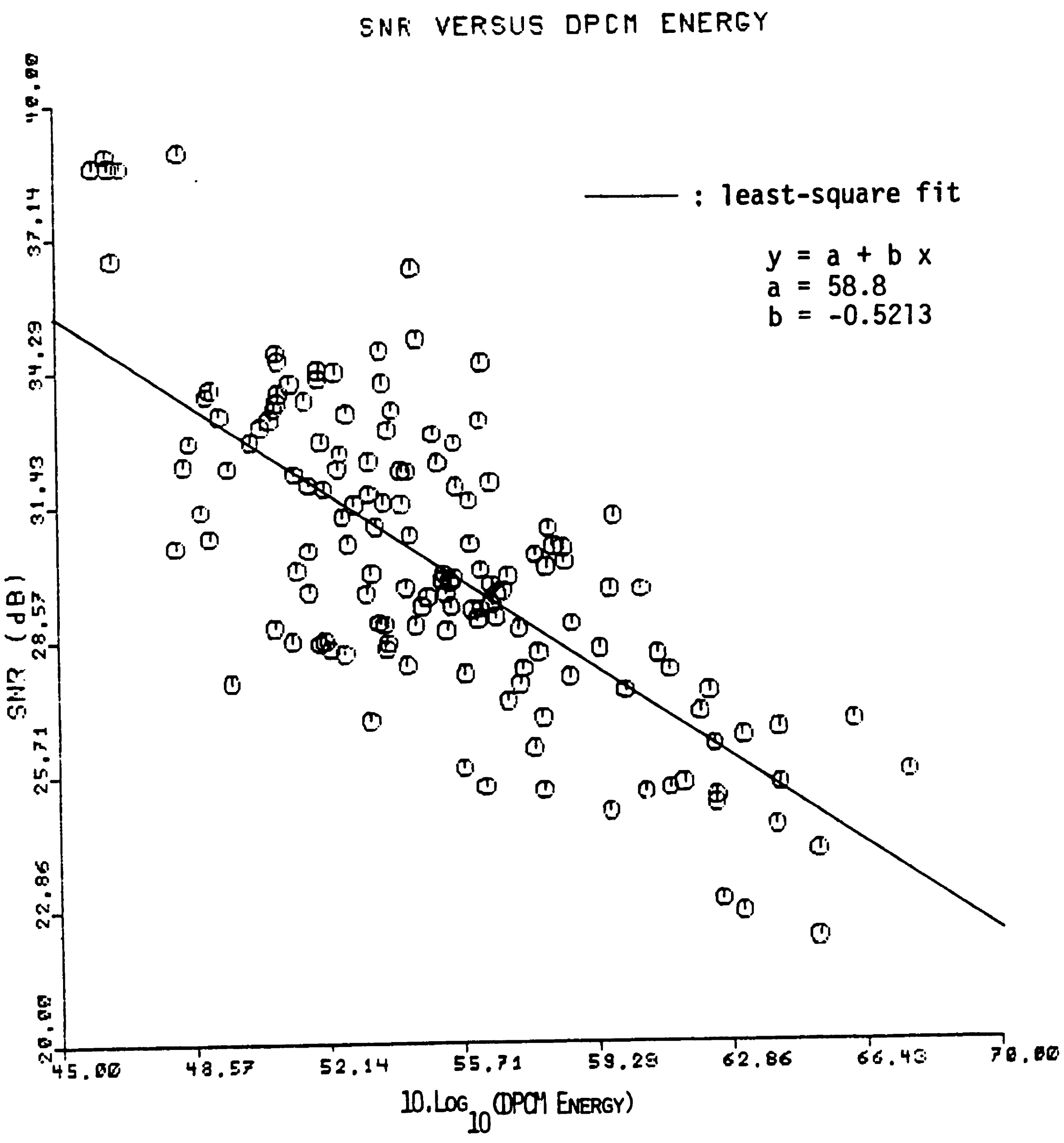


Figure 24 - Comparison of SNR versus DPCM log-energy for the LOT/DPCM coded sequence.

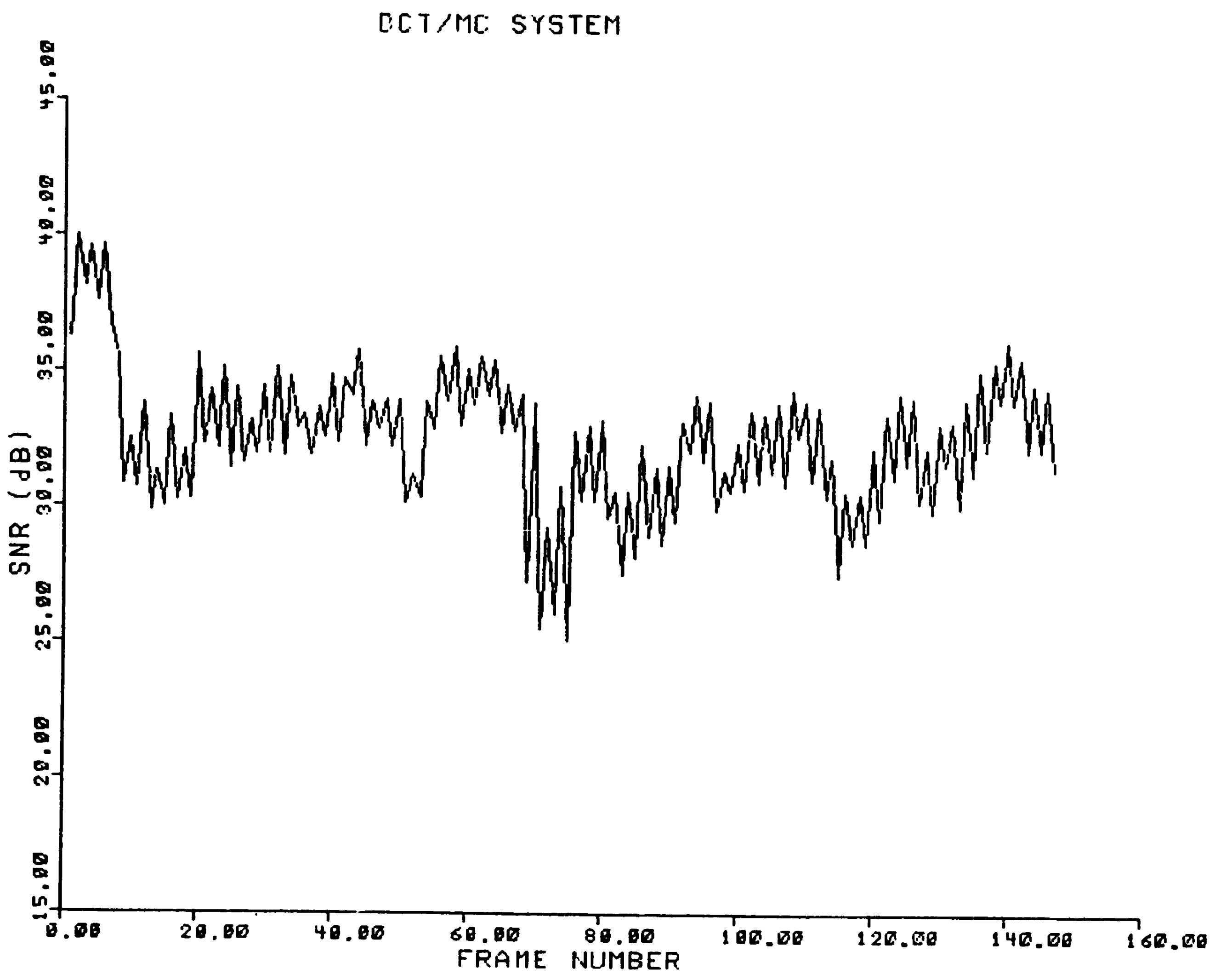


Figure 25 - SNR versus frame number for the DCT/MC coded sequence.

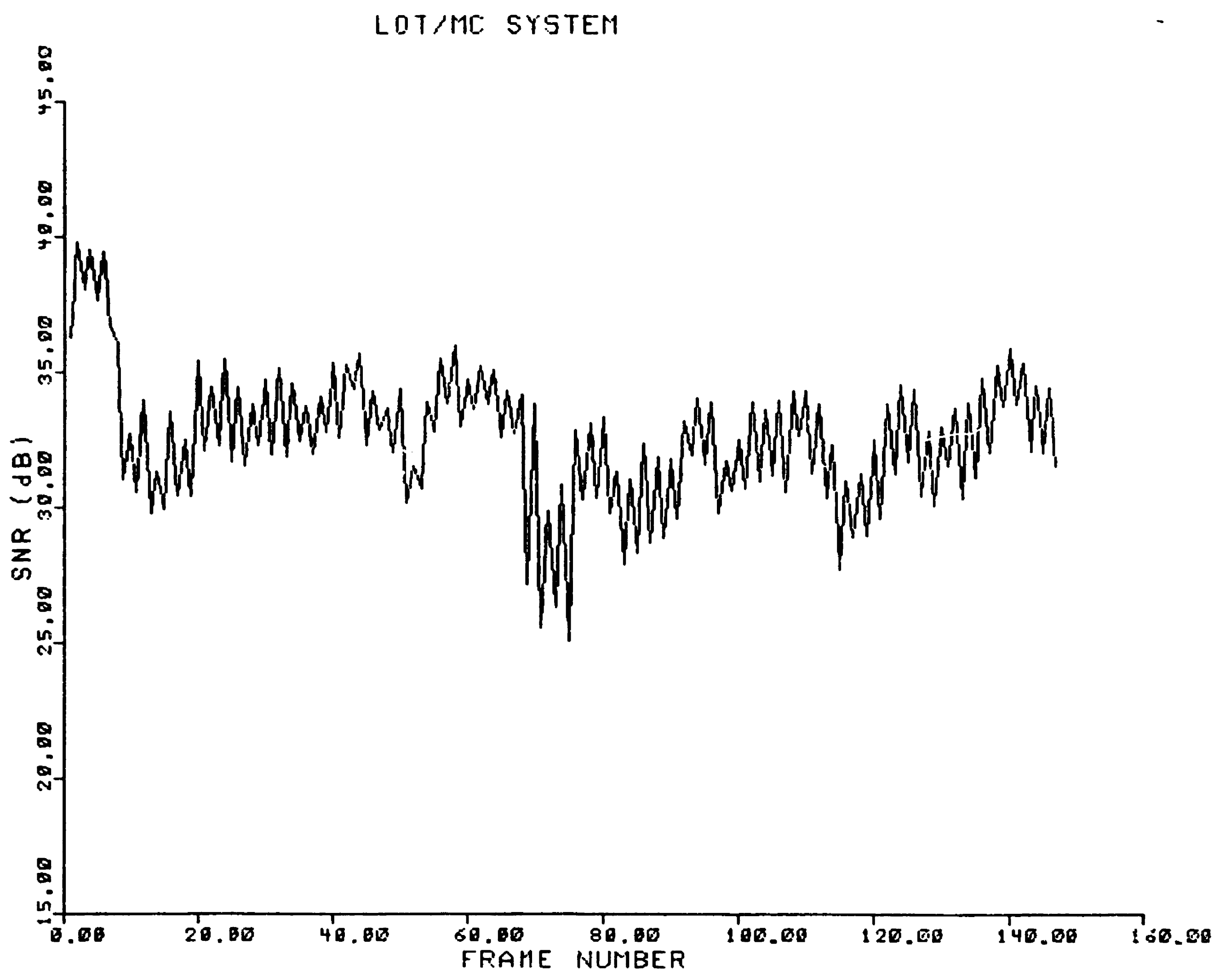


Figure 26 - SNR versus frame number for the LOT/MC coded sequence.

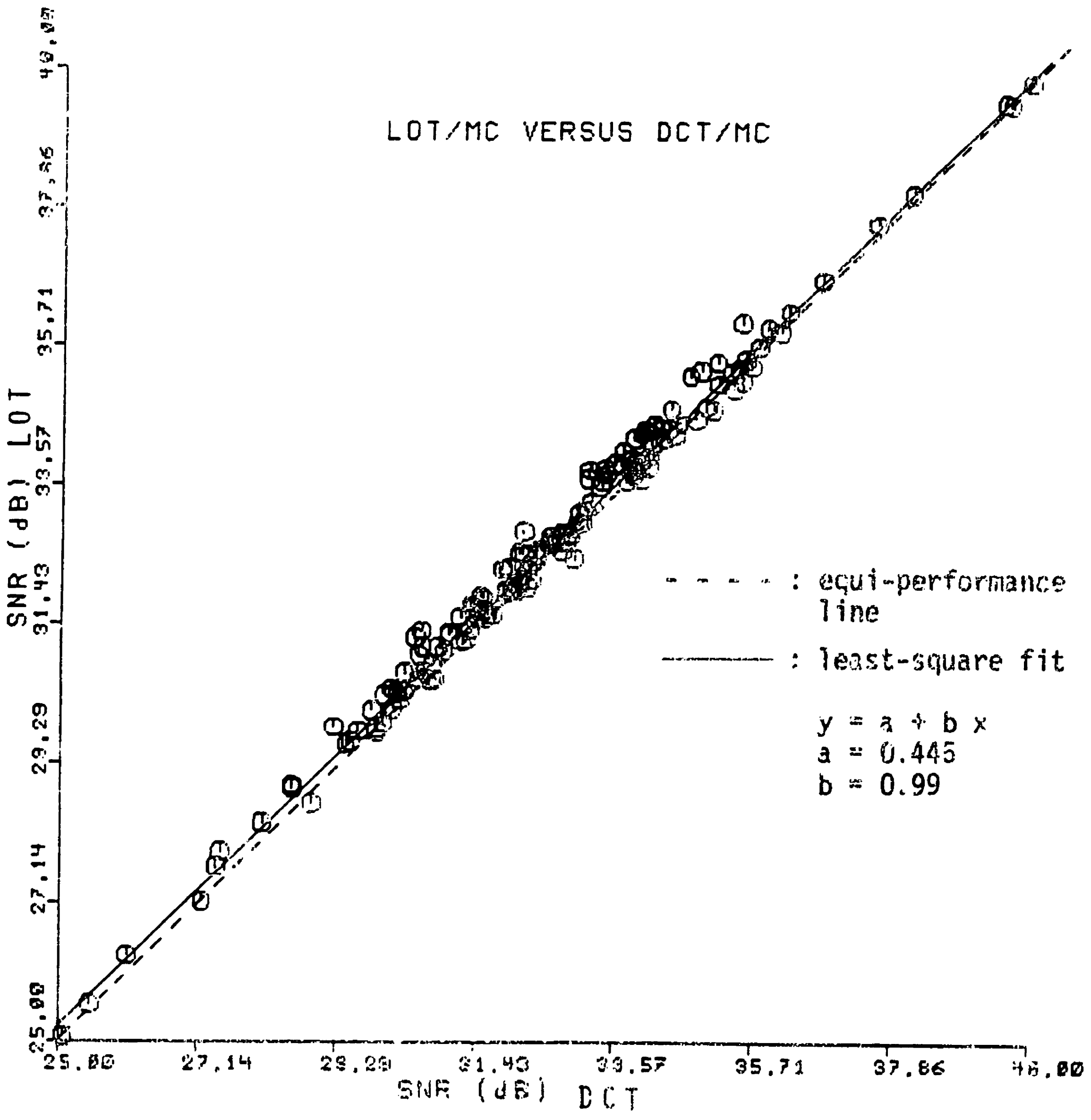


Figure 27 - Comparison of frame SNR's between the LOT/MC coded sequence and the DCT/MC coded sequence.

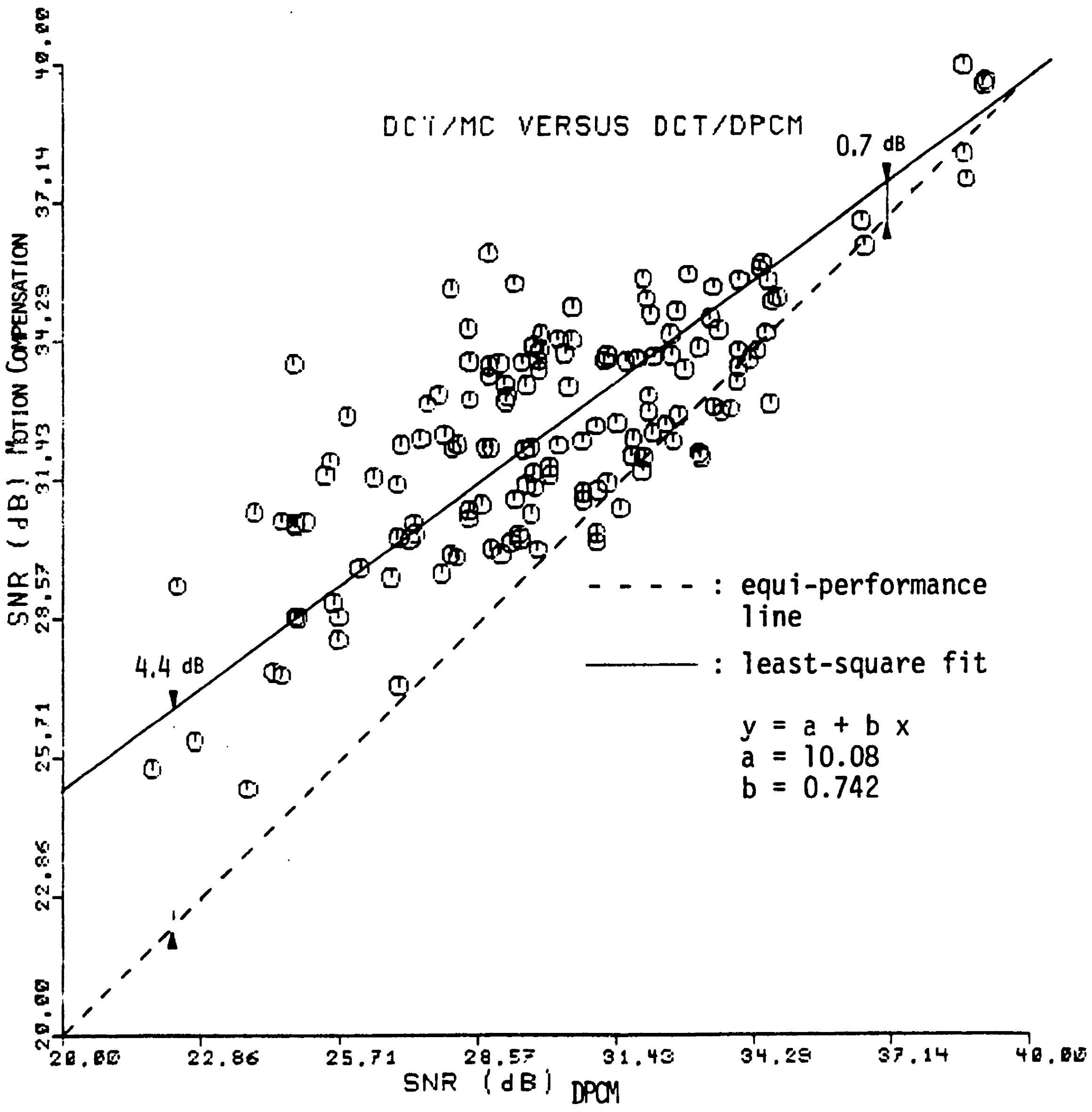


Figure 28 - Comparison of frame SNR's between the DCT/MC coded sequence and the DCT/DPCM coded sequence.

Figures 29 - 32 show four frames extracted from the four coded sequences. For frames 19 and 63, the performances by the four sequences are close because little motion is involved. On the other hand, frames 73 and 119 correspond to section of the sequence which contain a lot of head motion. Figures 31 and 32 demonstrate the dramatic improvement in image quality when a motion-compensated predictor is included in the coding system. For these two frames, the DCT/DPCM coded images are very noisy and show considerable blocking effects. Although the LOT/DPCM coded frames are as noisy, the blocking effects are much less visible. Similarly, remaining blocking effects in the DCT/MC coded frames disappear in the LOT/MC coded frames. In actuality, the LOT/MC coded sequence was virtually free of these artifacts. Figures 33 and 34 provide close-ups of the DCT/MC and the LOT/MC coded frame 119. The blocking effects, especially visible around the subject's right eye in the DCT/MC coded image, are avoided completely with the LOT/MC coding scheme.

In summary, the interframe coding experiments described in this chapter demonstrate the clear superiority of a motion-compensated system over a simple hybrid transform/DPCM system. In terms of both image quality and mean-square error, the performance of the motion-compensated systems was much better than the one of transform/DPCM systems. The DCT and the LOT perform almost identically in terms of mean-square error. However, the LOT reduces the DCT's blocking effects in the DPCM system, and

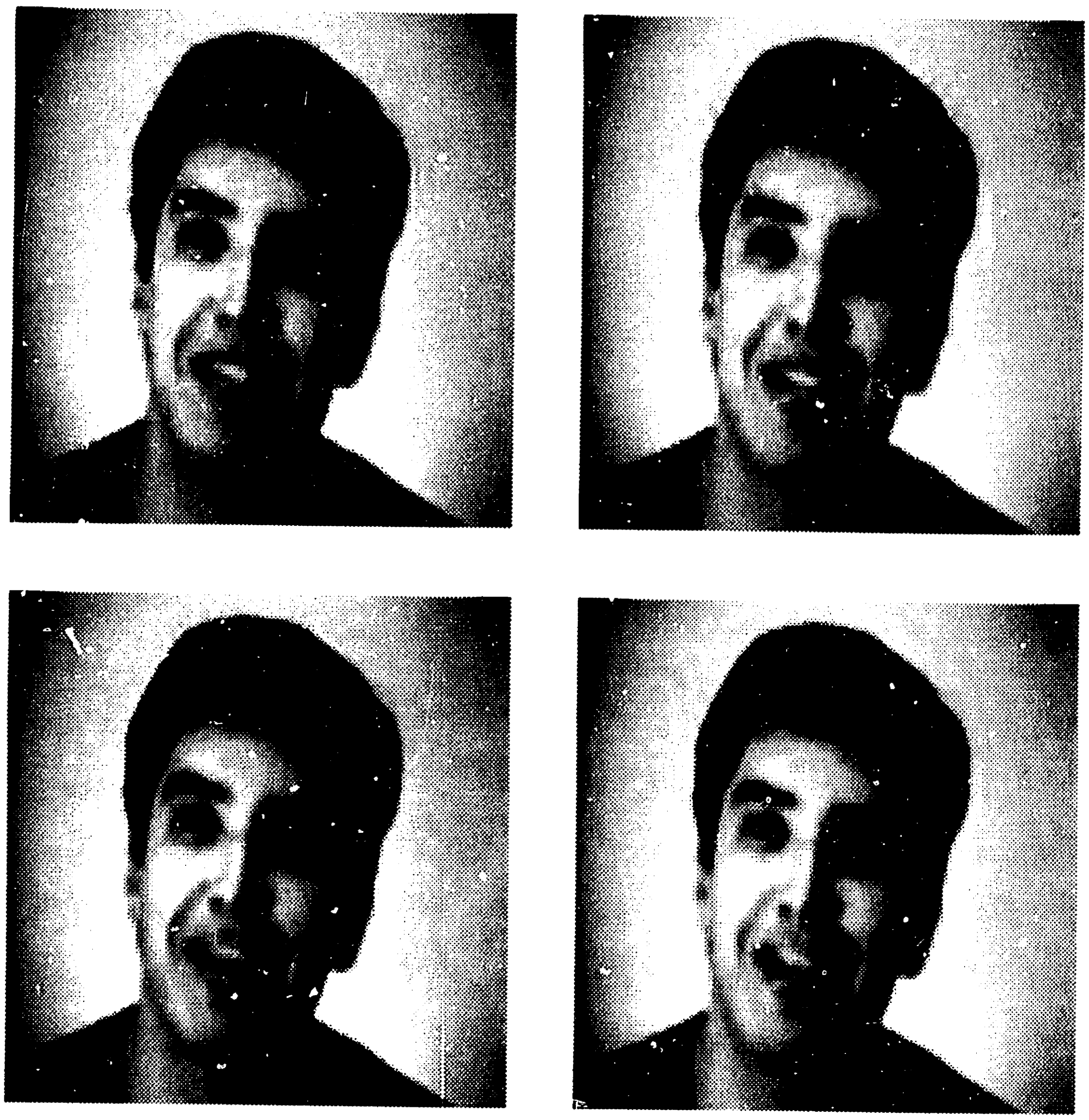


Figure 29 - Frame number 19 from the "MAN" sequence.

Upper left : DCT/DPCM coded sequence
Upper right: LOT/DPCM coded sequence
Lower left : DCT/MC coded sequence
Lower right: LOT/MC coded sequence
(original: 128 x 128, 8 bpp)

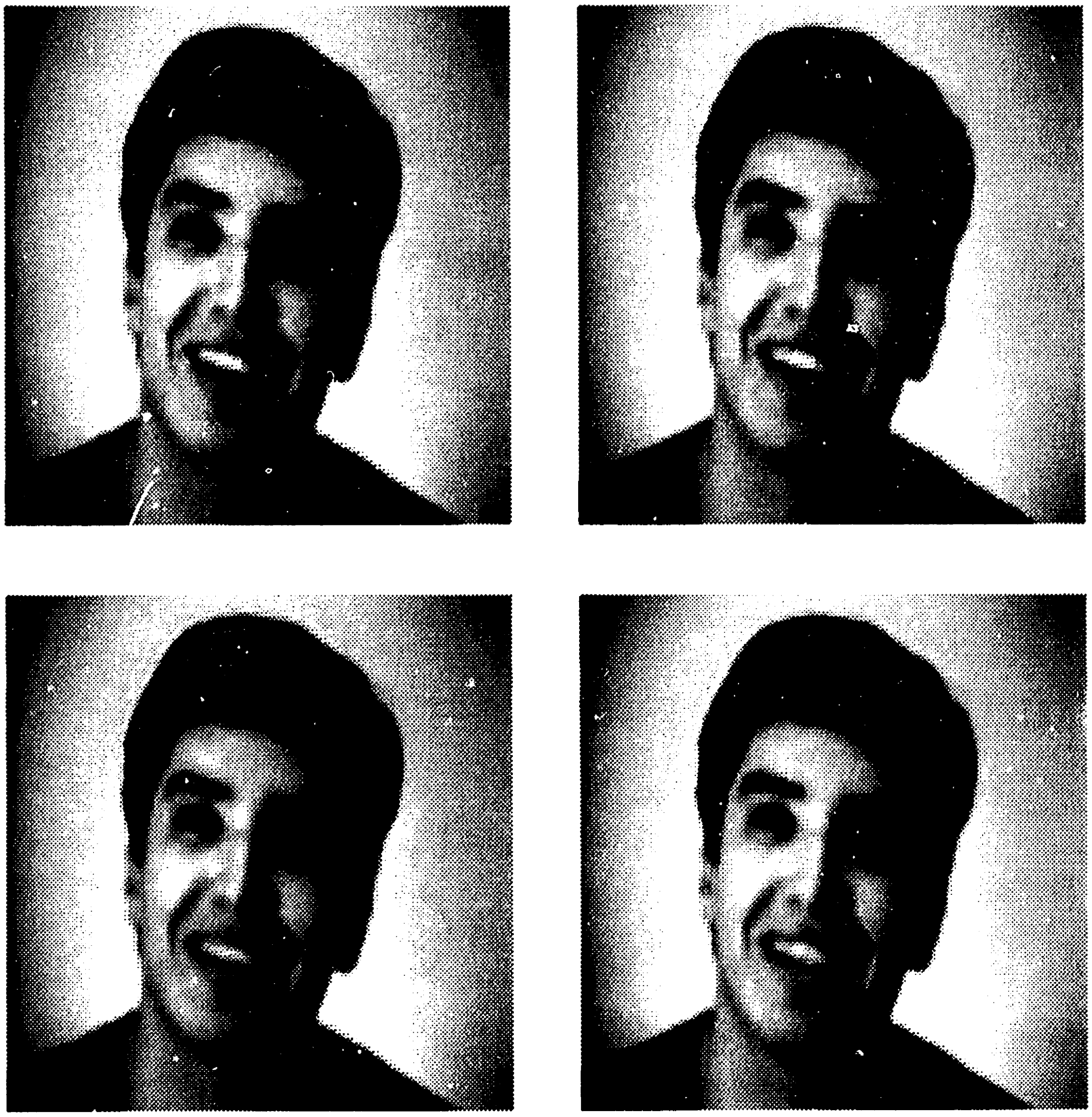


Figure 30 - Frame number 63 from the "MAN" sequence.

Upper left : DCT/DPCM coded sequence
Upper right: LOT/DPCM coded sequence
Lower left : DCT/MC coded sequence
Lower right: LOT/MC coded sequence
(original: 128 x 128, 8 bpp)

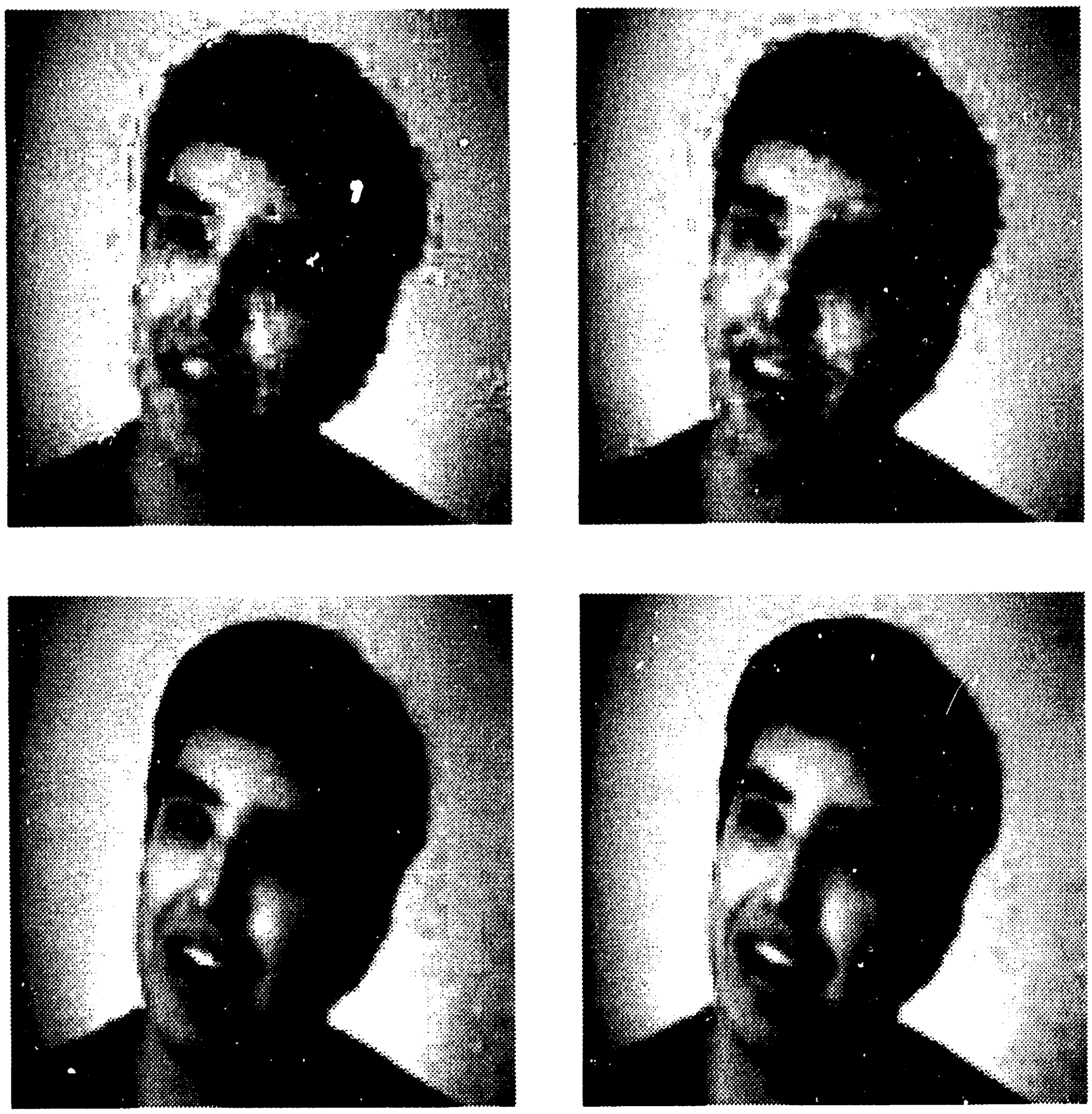


Figure 31 - Frame number 73 from the "MAN" sequence.

Upper left : DCT/DPCM coded sequence
Upper right: LOT/DPCM coded sequence
Lower left : DCT/MC coded sequence
Lower right: LOT/MC coded sequence
(original: 128 x 128, 8 bpp)

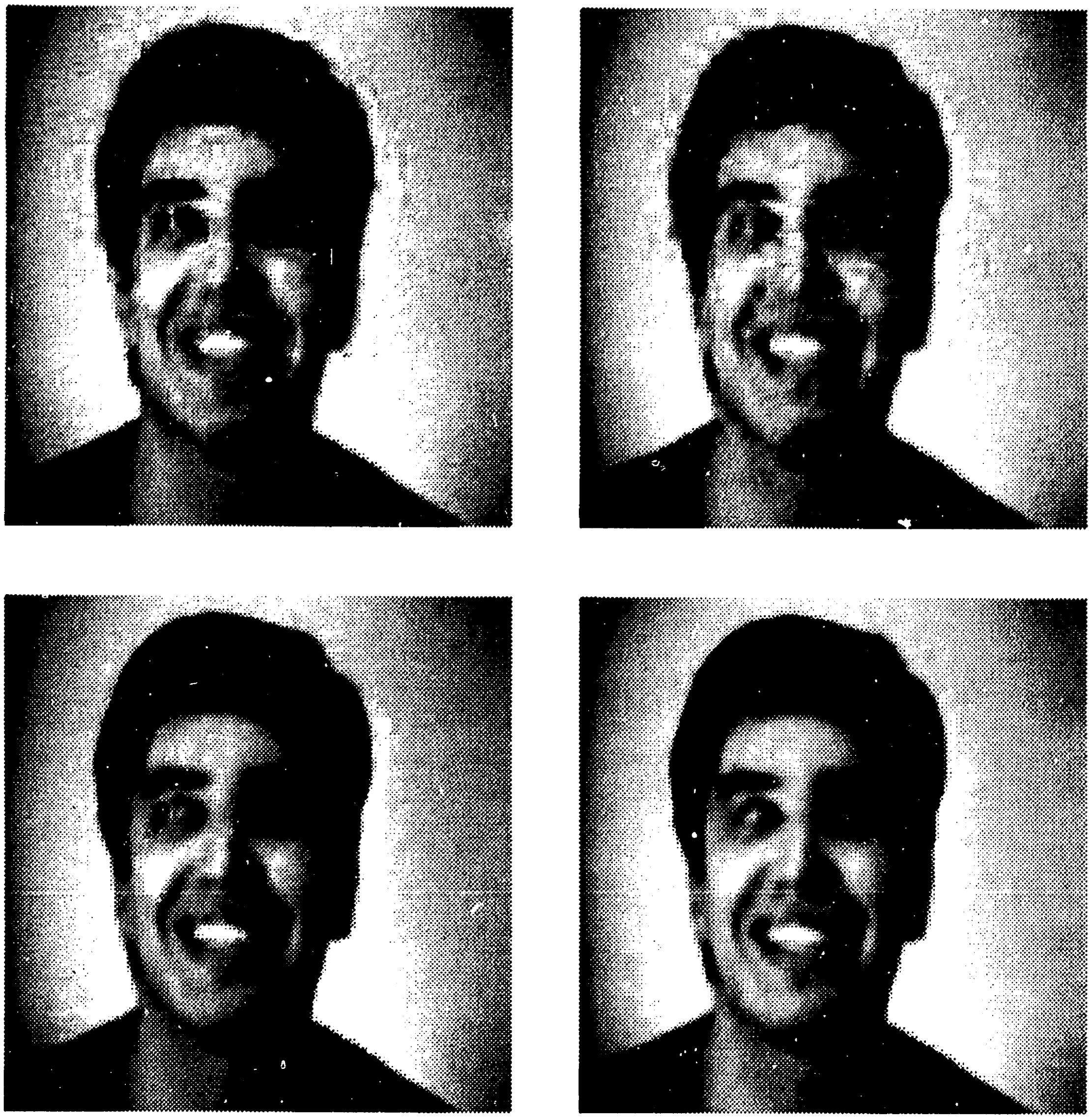


Figure 32 - Frame number 119 from the "MAN" sequence.

Upper left : DCT/DPCM coded sequence
Upper right: LOT/DPCM coded sequence
Lower left : DCT/MC coded sequence
Lower right: LOT/MC coded sequence
(original: 128 x 128, 8 bpp)

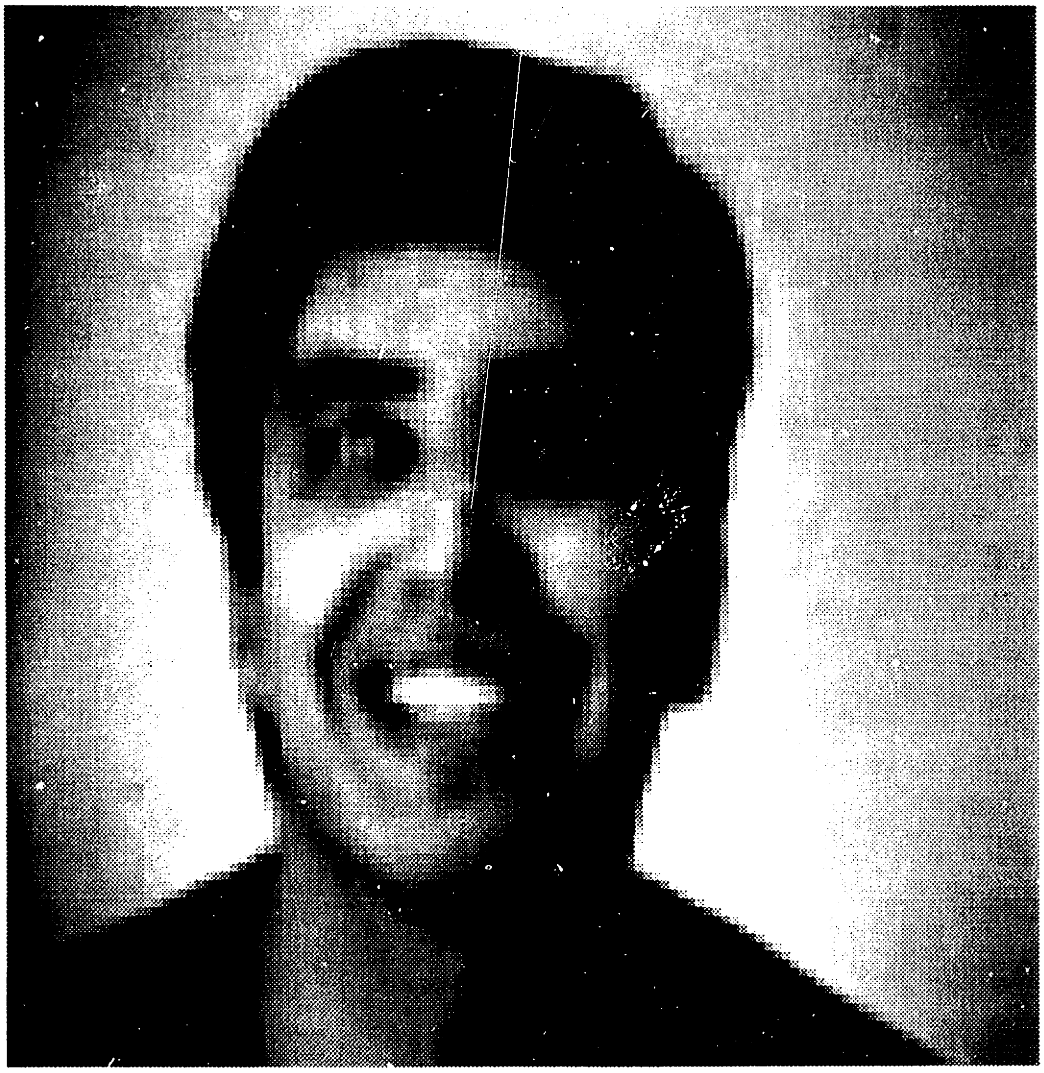


Figure 33 - Close-up of frame 119 from the DCT/MC coded sequence.

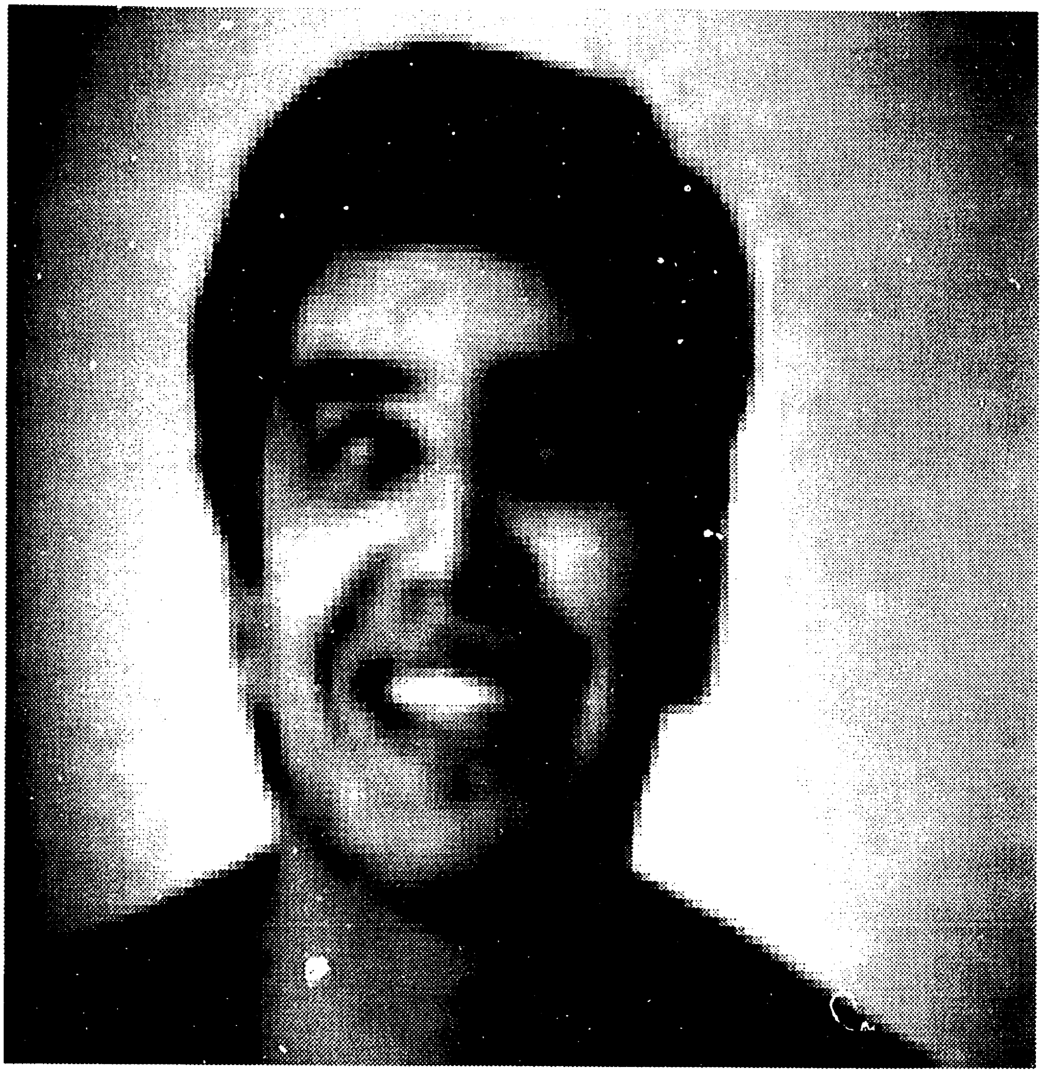


Figure 34 - Close-up of frame 119 from the LOT/MC coded sequence.

almost avoids them entirely in the motion compensated one (at 56 kbps). All viewers (the coded sequences were shown to half a dozen people familiar with image processing) recognized that the overall quality of the LOT coded sequences was better than the quality of DCT coded sequences at a constant data rate of 56 kbps.

CHAPTER SEVEN

CONCLUSIONS

A new type of unitary transform for image data compression has been introduced in this thesis. Only real separable unitary transforms were considered. The most important result of this research is that the unitarity property of the transform can be maintained with non-zero overlap between the basis functions. By overlapping the basis functions, the block transform process is no longer independent from block to block. By keeping the overall transform unitary, the image representation in the transform domain is non-redundant.

A specific example of a lapped orthogonal transform (LOT) was developed. The number of overlapping samples between adjacent block basis functions was equal to half the number of samples in a block. Furthermore, the LOT was optimized in terms of energy compaction. By the use of a non-linear optimization algorithm (the augmented Lagrangian method), a numerical model for the LOT was obtained.

Intraframe and interframe coding experiments were performed. The mean-square error coding performance of the DCT and the LOT were almost identical. However, in both intraframe and interframe coding experiments, eighty percent of the viewers preferred the quality of the LOT coded images to the quality of the images coded with the DCT and the SSFT. The remaining twenty percent of the viewers found the performance improvement of the LOT over the DCT not as significant. An important conclusion to these experiments is that the signal-to-noise ratio of a coded image does not account for local coding artifacts such as blocking effects. The blocking effects generated by the LOT are much less visible than those generated by the DCT. The ringing effects, generated by the SSFT, virtually disappear with the LOT.

Although the LOT's performance is better than the DCT's, local boundary effects between blocks still remain and sometimes are visible. This is due to the discontinuities of the LOT basis functions. Future research on lapped orthogonal transforms should try to eliminate completely these remaining artifacts. The basis functions could, for example, be constrained for a smooth decrease to zero at the boundaries. This would, however, worsen the energy compaction performance. A larger number of overlapping samples could also be considered.

Last, but not least, a transform can be used in practice only if its computational requirements are small enough to compete with other transforms such as the DCT. Although the superiority of the LOT over other transforms was proven, the possibility of developing a fast LOT, without sacrificing its performance in image coding, remains to be demonstrated.

APPENDIX A

The energy compaction of the $\underline{a}_i$ LOT basis function is defined as:

$$E_i = \frac{\underline{a}_i^t \, R \, \underline{a}_i}{\underline{a}_i^t \, \underline{a}_i} \tag{A.1}$$

where R = [R(k,l)] is the 2N x 2N matrix defined by:

$$R(k,l) = \rho^{|k-l|} \qquad k,l = 1,\ldots,2N \tag{A.2}$$

and where ρ is the correlation factor between adjacent pixels. According to equations (3.5) and (3.15), the $\underline{a}_i$ vector can be written:

$$\underline{a}_i^t = (\underline{x}_i^t, \ (-1)^{i+1} \ \underline{x}_i^t \ H) \tag{A.3}$$

where H is the N x N matrix:

$$H = \begin{pmatrix} 0 & & 1 \\ & \diagup & \\ 1 & & 0 \end{pmatrix} \tag{A.4}$$

Using a matrix notation, $\underline{a}_i$ can be written:

$$\underline{a}_i^t = \underline{x}_i^t \ (I \ \big| \ (-1)^{i+1} \ H) \tag{A.5}$$

The 2N x 2N matrix R has the following block structure:

$$R = \left(\begin{array}{c|c} R_1 & R_2 \\ \hline R_2^t & R_1 \end{array}\right) \tag{A.6}$$

where R_1 and R_2 are two N x N matrices respectively defined by:

$$R_1(k,l) = \rho^{|k-l|} \qquad k,l = 1,\dots,N \tag{A.7}$$

and

$$R_2(k,l) = \rho^{N-k+l} \qquad k,l = 1,\dots,N \tag{A.8}$$

Consequently, $\underline{a}_i^t \, R \, \underline{a}_i$ can be expressed in the following way:

$$\underline{a}_i^t \, R \, \underline{a}_i = \underline{x}_i^t \, (I \;|\; (-1)^{i+1} H) \left(\begin{array}{c|c} R_1 & R_2 \\ \hline R_2^t & R_1 \end{array}\right) \left(\begin{array}{c} I \\ \hline (-1)^{i+1} H \end{array}\right) \underline{x}_i \tag{A.9}$$

that is:

$$\underline{a}_i^t \, R \, \underline{a}_i = \underline{x}_i^t \, (R_1 + (-1)^{i+1} R_2 H + (-1)^{i+1} H R_2^t + H R_1 H) \, \underline{x}_i \tag{A.10}$$

It can be shown easily that because of the definition of H and the Toeplitz nature of R_1 and R_2, the following properties are derived:

$$H R_1 H = R_1 \tag{A.11}$$

and

$$R_2 H = H R_2^t \tag{A.12}$$

Thus, equation (A.10) reduces to:

$$\underline{a}_i^t \, R \, \underline{a}_i = 2 \, \underline{x}_i^t \, (R_1 + (-1)^{i+1} R_2 H) \, \underline{x}_i \tag{A.13}$$

Next, $\underline{a}_i^t \, \underline{a}_i$ can be expressed in terms of $\underline{x}_i$, following an analog procedure:

$$\underline{a}_i^t \, \underline{a}_i = \underline{x}_i^t \, (I \;|\; (-1)^{i+1} H) \left(\begin{array}{c} I \\ \hline (-1)^{i+1} H \end{array}\right) \underline{x}_i \tag{A.14}$$

This equation reduces to:

$$\underline{a}_i^t \, \underline{a}_i = 2 \, \underline{x}_i^t \, \underline{x}_i \tag{A.15}$$

Consequently, the energy compaction E_i, using equations (A.1), (A.13), and (A.15), is expressed as:

$$E_i = \frac{\underline{x}_i^t \, (R_1 + (-1)^{i+1} \, R_2 \, H) \, \underline{x}_i}{\underline{x}_i^t \, \underline{x}_i} \tag{A.16}$$

Define the N x N matrix R'_i:

$$R'_i = R_1 + (-1)^{i+1} \, R_2 \, H \tag{A.17}$$

$R'_i(k,l)$ can be expressed as follows:

$$R'_i(k,l) = \rho^{|k-l|} + (-1)^{i+1} \, \rho^{2N+1-k-l} \qquad k,l = 1,\dots,N \tag{A.18}$$

Therefore, the expression of E_i reduces to:

$$E_i = \frac{\underline{x}_i^t \, R'_i \, \underline{x}_i}{\underline{x}_i^t \, \underline{x}_i} \tag{A.19}$$

APPENDIX B

B.1 Computation of the gradient of $f(\underline{x})$

According to equation (4.9), the objective function $f(\underline{x})$ for the optimization problem of any of the LOT basis functions has the following form:

$$f(\underline{x}) = - \frac{\underline{x}^t R' \underline{x}}{\underline{x}^t \underline{x}} \tag{B.1}$$

Taking the gradient of $f(\underline{x})$ with respect to $\underline{x}$ yields:

$$\nabla_{\underline{x}} f(\underline{x}) = \frac{1}{(\underline{x}^t \underline{x})^2} \cdot \left(- (\underline{x}^t \underline{x}) \cdot \nabla_{\underline{x}} (\underline{x}^t R' \underline{x}) + (\underline{x}^t R' \underline{x}) \cdot \nabla_{\underline{x}} (\underline{x}^t \underline{x}) \right) \tag{B.2}$$

Additionally:

$$\nabla_{\underline{x}} (\underline{x}^t R' \underline{x}) = (R' + R'^t) \underline{x} = 2R' \underline{x} \tag{B.3}$$

and

$$\nabla_{\underline{x}} (\underline{x}^t \underline{x}) = 2\underline{x} \tag{B.4}$$

Using the results of equations (B.1), (B.3), and (B.4) in equation (B.2) yields the following expression of the gradient of $f(\underline{x})$:

$$\nabla_{\underline{x}} f(\underline{x}) = - \frac{2}{\underline{x}^t \underline{x}} \cdot (f(\underline{x}) \cdot \underline{x} + R' \underline{x}) \tag{B.5}$$

B.2 Computation of the Jacobian matrix of $\underline{c}(\underline{x})$

The constraints' vector $\underline{c}(\underline{x})$, defined by equation (4.7), can be written:

$$\underline{c}^t(\underline{x}) = \left(\gamma_1(\underline{x}),\ [\gamma_{2_j}\ (\underline{x})],\ [\gamma_{3_j}\ (\underline{x})]\right) \tag{B.6}$$

with:

$$\gamma_1(\underline{x}) = \frac{\underline{x}^t\ H\ \underline{x}}{\underline{x}^t\ \underline{x}} \tag{B.7}$$

$$\gamma_{2_j}(\underline{x}) = \frac{(H\ \underline{x}_j)^t\ \underline{x}}{(\underline{x}^t\ \underline{x})^{1/2}} \tag{B.8}$$

$$\gamma_{3_j}(\underline{x}) = \frac{\underline{x}_j^{\ t}\ \underline{x}}{(\underline{x}^t\ \underline{x})^{1/2}} \tag{B.9}$$

The Jacobian matrix of $\underline{c}(\underline{x})$ is defined as follows:

$$\nabla_{\underline{x}}(\underline{c}^t(\underline{x})) = \left(\nabla_{\underline{x}}\ \gamma_1(\underline{x}),\ [\nabla_{\underline{x}}\ \gamma_{2_j}\ (\underline{x})],\ [\nabla_{\underline{x}}\ \gamma_{3_j}\ (\underline{x})]\right) \tag{B.10}$$

where $\nabla_{\underline{x}}\ \gamma_1\ (\underline{x})$, $\nabla_{\underline{x}}\ \gamma_{2_j}\ (\underline{x})$, and $\nabla_{\underline{x}}\ \gamma_{3_j}\ (\underline{x})$ are the gradients with respect to $\underline{x}$ of $\gamma_1(\underline{x})$, $\gamma_{2_j}(\underline{x})$, and $\gamma_{3_j}(\underline{x})$, in that order. Taking the gradient of $\gamma_1\ (\underline{x})$ yields:

$$\nabla_{\underline{x}}\ \gamma_1(\underline{x}) = \frac{1}{(\underline{x}^t\underline{x})^2} \cdot((\underline{x}^t\underline{x})\cdot\ \nabla_{\underline{x}}\ (\underline{x}^t\ H\ \underline{x}) - (\underline{x}^t H\underline{x})\ \cdot\nabla_{\underline{x}}(\underline{x}^t\underline{x})) \tag{B.11}$$

Taking the gradient of $\gamma_{2_j}(\underline{x})$ yields:

$$\nabla_{\underline{x}}\ \gamma_{2_j}(\underline{x}) = \frac{1}{\underline{x}^t\ \underline{x}} \cdot\ ((\underline{x}^t\underline{x})^{1/2}\cdot\ \nabla_{\underline{x}}((H\underline{x}_j)^t\underline{x}) - ((H\underline{x}_j)^t\underline{x})\cdot\nabla_{\underline{x}}((\underline{x}^t\underline{x})^{1/2})) \tag{B.12}$$

Taking the gradient of $\gamma_{3_j}(\underline{x})$ yields:

$$\nabla_{\underline{x}}\, \gamma_{3_j}(\underline{x}) = \frac{1}{\underline{x}^t\, \underline{x}} \cdot ((\underline{x}^t\, \underline{x})^{1/2} \cdot \nabla_{\underline{x}}(\underline{x}_j{}^t \underline{x}) - (\underline{x}_j{}^t \underline{x}) \cdot \nabla_{\underline{x}}((\underline{x}^t \underline{x})^{1/2})) \quad \text{(B.13)}$$

Following this, the next equations are easily derived:

$$\nabla_{\underline{x}}\, (\underline{x}^t\, H\, \underline{x}) = 2H\, \underline{x} \quad \text{(B.14)}$$

$$\nabla_{\underline{x}}\, (\underline{x}^t\, \underline{x}) = 2\underline{x} \quad \text{(B.15)}$$

$$\nabla_{\underline{x}}\, ((H\, \underline{x}_j)^t\, \underline{x}) = H\, \underline{x}_j \quad \text{(B.16)}$$

$$\nabla_{\underline{x}}\, (\underline{x}_j{}^t\, \underline{x}) = \underline{x}_j \quad \text{(B.17)}$$

$$\nabla_{\underline{x}}\, ((\underline{x}^t\, \underline{x})^{1/2}) = \frac{1}{(\underline{x}^t\, \underline{x})^{1/2}} \cdot \underline{x} \quad \text{(B.18)}$$

Combining the results given by equations (B.14) - (B.18), and the definitions given by equations (B.7) - (B.9), with equations (B.11) - (B.13) yields:

$$\nabla_{\underline{x}}\gamma_1\, (\underline{x}) = \frac{2}{\underline{x}^t \underline{x}} \cdot (H\, \underline{x} - \gamma_1(\underline{x}) \cdot \underline{x}) \quad \text{(B.19)}$$

$$\nabla_{\underline{x}}\gamma_{2_j}(\underline{x}) = \frac{1}{(\underline{x}^t \underline{x})^{1/2}} \cdot (H\, \underline{x}_j - \frac{\gamma_{2_j}(\underline{x})}{(\underline{x}^t \underline{x})^{1/2}} \cdot \underline{x}) \quad \text{(B.20)}$$

$$\nabla_{\underline{x}}\gamma_{3_j}(\underline{x}) = \frac{1}{(\underline{x}^t \underline{x})^{1/2}} \cdot (\underline{x}_j - \frac{\gamma_{3_j}(\underline{x})}{(\underline{x}^t \underline{x})^{1/2}} \cdot \underline{x}) \quad \text{(B.21)}$$

Replacing the expressions of the gradients of $\gamma_1(\underline{x})$, $\gamma_{2_j}(\underline{x})$ and $\gamma_{3_j}(\underline{x})$, (B.19) - (B.21), in equation (B.10) results in a closed-form formulation of the Jacobian matrix of $\underline{c}(\underline{x})$.

APPENDIX C

C.1 The coefficients of the LOT basis functions for a block size equal to 8 (overlap equal to 4)

Basis 1	Basis 2	Basis 3	Basis 4
-0.85022E -1	-0.94311E -1	-0.28200E -1	-0.27498E -1
-0.25800E -1	-0.90764E -1	-0.17315E 0	0.12354E 0
0.50189E -1	0.44394E -2	-0.15729E 0	0.23669E 0
0.13631E 0	0.16241E 0	0.95968E -1	-0.32202E -1
0.22490E 0	0.32116E 0	0.36463E 0	-0.37045E 0
0.30792E 0	0.40951E 0	0.33735E 0	-0.18149E 0
0.37772E 0	0.37587E 0	-0.33854E -1	0.33563E 0
0.42772E 0	0.21059E 0	-0.43286E 0	0.37960E 0
0.42772E 0	-0.21059E 0	-0.43286E 0	-0.37960E 0
0.37772E 0	-0.37587E 0	-0.33854E -1	-0.33563E 0
0.30792E 0	-0.40951E 0	0.33735E 0	0.18149E 0
0.22490E 0	-0.32116E 0	0.36463E 0	0.37045E 0
0.13631E 0	-0.16241E 0	0.95968E -1	0.32202E -1
0.50189E -1	-0.44394E -2	-0.15729E 0	-0.23669E 0
-0.25800E -1	0.90764E -1	-0.17315E 0	-0.12354E 0
-0.85022E -1	0.94311E -1	-0.28200E -1	0.27498E -1

Basis 5	Basis 6	Basis 7	Basis 8
-0.63229E -1	0.42286E -1	-0.35594E -1	0.43009E -1
-0.51062E -1	0.11491E 0	0.89607E -1	-0.67578E -1
0.21991E 0	-0.14696E 0	-0.54318E -1	0.31287E -1
0.28055E -1	-0.56691E -1	-0.14414E 0	0.13690E 0
-0.33969E 0	0.30653E 0	0.38952E 0	-0.34102E 0
0.58039E -1	-0.18817E 0	-0.45766E 0	0.43051E 0
0.46776E 0	-0.27526E 0	0.30999E 0	-0.35921E 0
-0.32664E 0	0.50504E 0	-0.98540E -1	0.20785E 0
-0.32664E 0	-0.50504E 0	-0.98540E -1	-0.20785E 0
0.46776E 0	0.27526E 0	0.30999E 0	0.35921E 0
0.58039E -1	0.18817E 0	-0.45766E 0	-0.43051E 0
-0.33969E 0	-0.30653E 0	0.38952E 0	0.34102E 0
0.28055E -1	0.56691E -1	-0.14414E 0	-0.13690E 0
0.21991E 0	0.14696E 0	-0.54318E -1	-0.31287E -1
-0.51062E -1	-0.11491E 0	0.89607E -1	0.67578E -1
-0.63229E -1	-0.42286E -1	-0.35594E -1	-0.43009E -1

C.2 The coefficients of the LOT basis functions for a block size equal to 16 (overlap equal to 8)

-0.65329E -1	-0.58681E -1	0.72729E -2	-0.35039E -1
-0.50285E -1	-0.74737E -1	-0.50451E -1	-0.26299E -1
-0.30797E -1	-0.75075E -1	-0.10813E 0	0.41422E -1
-0.73557E -2	-0.57837E -1	-0.13818E 0	0.12851E 0
0.19430E -1	-0.23391E -1	-0.12571E 0	0.18022E 0
0.48850E -1	0.25635E -1	-0.69272E -1	0.15496E 0
0.80114E -1	0.84588E -1	0.20237E -1	0.47925E -1
0.11237E 0	0.14726E 0	0.12258E 0	-0.10230E 0
0.14473E 0	0.20655E 0	0.21259E 0	-0.22886E 0
0.17630E 0	0.25525E 0	0.26544E 0	-0.26755E 0
0.20618E 0	0.28683E 0	0.26210E 0	-0.18907E 0
0.23354E 0	0.29624E 0	0.19419E 0	-0.17016E -1
0.25759E 0	0.28047E 0	0.67758E -1	0.17836E 0
0.27762E 0	0.23901E 0	-0.94835E -1	0.30916E 0
0.29303E 0	0.17392E 0	-0.25502E 0	0.30925E 0
0.30335E 0	0.89777E -1	-0.36041E 0	0.16604E 0
0.30335E 0	-0.89777E -1	-0.36041E 0	-0.16604E 0
0.29303E 0	-0.17392E 0	-0.25502E 0	-0.30925E 0
0.27762E 0	-0.23901E 0	-0.94835E -1	-0.30916E 0
0.25759E 0	-0.28047E 0	0.67758E -1	-0.17836E 0
0.23354E 0	-0.29624E 0	0.19419E 0	0.17016E -1
0.20618E 0	-0.28683E 0	0.26210E 0	0.18907E 0
0.17630E 0	-0.25525E 0	0.26544E 0	0.26755E 0
0.14473E 0	-0.20655E 0	0.21259E 0	0.22886E 0
0.11237E 0	-0.14726E 0	0.12258E 0	0.10230E 0
0.80114E -1	-0.84588E -1	0.20237E -1	-0.47925E -1
0.48850E -1	-0.25635E -1	-0.69272E -1	-0.15496E 0
0.19430E -1	0.23391E -1	-0.12571E 0	-0.18022E 0
-0.73557E -2	0.57837E -1	-0.13818E 0	-0.12851E 0
-0.30797E -1	0.75075E -1	-0.10813E 0	-0.41422E -1
-0.50285E -1	0.74737E -1	-0.50451E -1	0.26299E -1
-0.65329E -1	0.58681E -1	0.72729E -2	0.35039E -1
Basis 1	Basis 2	Basis 3	Basis 4

-0.72912E -2	-0.24180E -1	-0.19006E -1	-0.89802E -2
-0.76501E -1	0.29073E -1	-0.75269E -1	0.70105E -1
-0.87328E -1	0.12434E 0	0.10048E -1	0.87746E -1
0.51573E -2	0.10870E 0	0.14906E 0	-0.86586E -1
0.14098E 0	-0.43563E -1	0.90066E -1	-0.14643E 0
0.20117E 0	-0.18514E 0	-0.13635E 0	0.77252E -1
0.11055E 0	-0.14706E 0	-0.19998E 0	0.21654E 0
-0.88048E -1	0.64334E -1	0.35779E -1	-0.16559E -1
-0.25334E 0	0.24515E 0	0.25568E 0	-0.25478E 0
-0.25001E 0	0.19279E 0	0.11915E 0	-0.62066E -1
-0.59627E -1	-0.70123E -1	-0.21435E 0	0.26550E 0
0.19141E 0	-0.29106E 0	-0.26702E 0	0.15060E 0
0.31663E 0	-0.23263E 0	0.68851E -1	-0.24656E 0
0.20997E 0	0.70840E -1	0.33079E 0	-0.24152E 0
-0.63380E -1	0.32103E 0	0.12467E 0	0.18916E 0
-0.30147E 0	0.24548E 0	-0.27777E 0	0.29730E 0
-0.30147E 0	-0.24548E 0	-0.27777E 0	-0.29730E 0
-0.63380E -1	-0.32103E 0	0.12467E 0	-0.18916E 0
0.20997E 0	-0.70840E -1	0.33079E 0	0.24152E 0
0.31663E 0	0.23263E 0	0.68851E -1	0.24656E 0
0.19141E 0	0.29106E 0	-0.26702E 0	-0.15060E 0
-0.59627E -1	0.70123E -1	-0.21435E 0	-0.26550E 0
-0.25001E 0	-0.19279E 0	0.11915E 0	0.62066E -1
-0.25334E 0	-0.24515E 0	0.25568E 0	0.25478E 0
-0.88048E -1	-0.64334E -1	0.35779E -1	0.16559E -1
0.11055E 0	0.14706E 0	-0.19998E 0	-0.21654E 0
0.20117E 0	0.18514E 0	-0.13635E 0	-0.77252E -1
0.14098E 0	0.43563E -1	0.90066E -1	0.14643E 0
0.51573E -2	-0.10870E 0	0.14906E 0	0.86586E -1
-0.87328E -1	-0.12434E 0	0.10048E -1	-0.87746E -1
-0.76501E -1	-0.29073E -1	-0.75269E -1	-0.70105E -1
-0.72912E -2	0.24180E -1	-0.19006E -1	0.89802E -2
Basis 5	Basis 6	Basis 7	Basis 8

-0.28755E -1	0.28481E -2	-0.37357E -1	0.34022E -1
-0.45148E -1	0.92651E -1	0.38207E -1	0.16318E -1
0.11799E 0	-0.51676E -1	0.45648E -1	-0.63171E -1
0.38548E -1	-0.92427E -1	-0.91325E -1	0.51597E -1
-0.15170E 0	0.10303E 0	-0.38934E -2	0.56348E -1
-0.38643E -1	0.11008E 0	0.15496E 0	-0.13285E 0
0.20071E 0	-0.18470E 0	-0.14206E 0	0.86423E -1
0.19706E -1	-0.46762E -1	-0.83763E -1	0.98233E -1
-0.25435E 0	0.26649E 0	0.27424E 0	-0.25368E 0
0.38465E -1	-0.93378E -1	-0.16840E 0	0.21853E 0
0.26994E 0	-0.24891E 0	-0.14715E 0	0.10747E -1
-0.86729E -1	0.22629E 0	0.31651E 0	-0.24429E 0
-0.27441E 0	0.14622E 0	-0.13756E 0	0.25184E 0
0.11090E 0	-0.29959E 0	-0.20142E 0	-0.95500E -2
0.33942E 0	-0.39196E -1	0.34954E 0	-0.29214E 0
-0.25919E 0	0.34000E 0	-0.16860E 0	0.36499E 0
-0.25919E 0	-0.34000E 0	-0.16860E 0	-0.36499E 0
0.33942E 0	0.39196E -1	0.34954E 0	0.29214E 0
0.11090E 0	0.29959E 0	-0.20142E 0	0.95500E -2
-0.27441E 0	-0.14622E 0	-0.13756E 0	-0.25184E 0
-0.86729E -1	-0.22629E 0	0.31651E 0	0.24429E 0
0.26994E 0	0.24891E 0	-0.14715E 0	-0.10747E -1
0.38465E -1	0.93378E -1	-0.16840E 0	-0.21853E 0
-0.25435E 0	-0.26649E 0	0.27424E 0	0.25368E 0
0.19706E -1	0.46762E -1	-0.83763E -1	-0.98233E -1
0.20071E 0	0.18470E 0	-0.14206E 0	-0.86423E -1
-0.38643E -1	-0.11008E 0	0.15496E 0	0.13285E 0
-0.15170E 0	-0.10303E 0	-0.38934E -2	-0.56348E -1
0.38548E -1	0.92427E -1	-0.91325E -1	-0.51597E -1
0.11799E 0	0.51676E -1	0.45648E -1	0.63171E -1
-0.45148E -1	-0.92651E -1	0.38207E -1	-0.16318E -1
-0.28755E -1	-0.28481E -2	-0.37357E -1	-0.34022E -1
Basis 9	Basis 10	Basis 11	Basis 12

```
-0.12490E -1    0.23669E -1    0.46286E -1    0.47395E -1
 0.25251E -1   -0.13657E -1   -0.12781E  0   -0.12594E  0
-0.65244E -2   -0.10743E -1    0.22782E  0    0.22663E  0
-0.47359E -1    0.49933E -1   -0.31026E  0   -0.31025E  0
 0.10030E  0   -0.68996E -1    0.33664E  0    0.34088E  0
-0.10271E  0    0.70379E -1   -0.31468E  0   -0.31795E  0
 0.33829E -1   -0.20367E -1    0.25925E  0    0.26133E  0
 0.96108E -1   -0.84837E -1   -0.19014E  0   -0.18596E  0
-0.24297E  0    0.21723E  0    0.11136E  0    0.10341E  0
 0.31377E  0   -0.28299E  0   -0.42466E -1   -0.31914E -1
-0.20972E  0    0.20871E  0   -0.60542E -2   -0.10734E -1
-0.45508E -1   -0.15861E -1    0.28391E -1    0.25021E -1
 0.28403E  0   -0.19722E  0   -0.36019E -1   -0.24057E -1
-0.34500E  0    0.32227E  0    0.30290E -1    0.22258E -1
 0.23798E  0   -0.33071E  0   -0.18694E -1   -0.19931E -1
-0.77794E -1    0.23688E  0    0.60906E -2    0.13457E -1
-0.77794E -1   -0.23688E  0    0.60906E -2   -0.13457E -1
 0.23798E  0    0.33071E  0   -0.18694E -1    0.19931E -1
-0.34500E  0   -0.32227E  0    0.30290E -1   -0.22258E -1
 0.28403E  0    0.19722E  0   -0.36019E -1    0.24057E -1
-0.45508E -1    0.15861E -1    0.28391E -1   -0.25021E -1
-0.20972E  0   -0.20871E  0   -0.60542E -2    0.10734E -1
 0.31377E  0    0.28299E  0   -0.42466E -1    0.31914E -1
-0.24297E  0   -0.21723E  0    0.11136E  0   -0.10341E  0
 0.96108E -1    0.84837E -1   -0.19014E  0    0.18596E  0
 0.33829E -1    0.20367E -1    0.25925E  0   -0.26133E  0
-0.10271E  0   -0.70379E -1   -0.31468E  0    0.31795E  0
 0.10030E  0    0.68996E -1    0.33664E  0   -0.34088E  0
-0.47359E -1   -0.49933E -1   -0.31026E  0    0.31025E  0
-0.65244E -2    0.10743E -1    0.22782E  0   -0.22663E  0
 0.25251E -1    0.13657E -1   -0.12781E  0    0.12594E  0
-0.12490E -1   -0.23669E -1    0.46286E -1   -0.47395E -1
```

Basis 13 Basis 14 Basis 15 Basis 16

APPENDIX D

Standard FORTRAN program for the computation of the LOT basis functions

```
C
C                       BASISn.FR
C
C               * = n + Int((n-1)/2)
C               ** = Int((n-1)/2)
C
      DOUBLE PRECISION RAU,R(8,8),ISTEP,ISTEPO,ACCF,DECF
      DOUBLE PRECISION CXMAX,CXMIN,TCX
      DOUBLE PRECISION NORMX,OBJ,PHI,NORMXOP,OBJOP,PHIOP,SIGMA
      DOUBLE PRECISION X(8),XOP(8),XCX(8),DXOBJ(8),DXPHI(8)
      DOUBLE PRECISION LAMBDA(*),CX(*),CXOP(*),CXCX(*),LAMBDACX(*),JACOB(*,8)
      DOUBLE PRECISION MA(n-1,8),A(8)
      ACCEPT "Enter correlation factor:    ",RAU
      ACCEPT "Enter initial step size:     ",ISTEPO
      ACCEPT "Enter acceleration factor:   ",ACCF
      ACCEPT "Enter decceleration factor:  ",DECF
      ACCEPT "Enter precision:             ",TCX
      ACCEPT "Enter number of iterations:  ",NITER
      ACCEPT "Enter sigma:                 ",SIGMA
      ACCEPT "Is this your first run (Yes=1,No=0) ?",M
      CXMIN=1.0D10
      IF(M.EQ.0) GOTO 30
      DO 10 I=1,*
      LAMBDA(I)=1.0D0
10    CONTINUE
      DO 20 I=1,8
      X(I)=1.0D0
20    CONTINUE
      GOTO 40
30    CALL FOPEN(25,"LAMBn")
      READ BINARY(25) LAMBDA
      CALL FCLOS(25)
      CALL FOPEN(25,"BASnP")
      READ BINARY(25) X
      CALL FCLOS(25)
40    DO 60 I=1,8
      DO 50 J=1,8
      R(I,J)=RAU**(DBLE(ABS(FLOAT(I-J))))+
             ((-1)**(n+1))*RAU**(DBLE(FLOAT(17-I-J)))
50    CONTINUE
60    CONTINUE
C
C     Repeat for i=1,n-1
C
      CALL FOPEN(20,"BASi")
      READ BINARY(20) A
      CALL FCLOS(20)
      DO 7i I=1,8
      MA(i,I)=A(I)
7i    CONTINUE
1000  N=0
      PHIOP=1.0D30
      ISTEP=ISTEPO
2000  N=N+1
```

```
        NORMX=0.0D0
        DO 80 I=1,8
        NORMX=NORMX+X(I)*X(I)
80      CONTINUE
        OBJ=0.0D0
        DO 100 J=1,8
        DO 90 I=1,8
        OBJ=OBJ+X(J)*R(I,J)*X(I)
90      CONTINUE
100     CONTINUE
        OBJ=-OBJ/NORMX
        DO 110 I=1,*
        CX(I)=0.0D0
110     CONTINUE
        DO 120 I=1,8
        CX(1)=CX(1)+X(I)*X(9-I)
120     CONTINUE
        DO 140 K=1,n-1
        DO 130 I=1,8
        CX(K+1)=CX(K+1)+MA(K,9-I)*X(I)
130     CONTINUE
140     CONTINUE
        DO 160 L=0,**-1      if n odd                   (1)
               L=1,**        if n even                  (2)
        DO 150 I=1,8
        CX(n+1+L)=CX(n+1+L)+MA(2*L+1,I)*X(I)            (1)
        CX(n+L)=CX(n+L)+MA(2*L,I)*X(I)                  (2)
150     CONTINUE
160     CONTINUE
        CX(1)=CX(1)/NORMX
        DO 170 I=2,*
        CX(I)=CX(I)/DSQRT(NORMX)
170     CONTINUE
        CXMAX=0.0D0
        DO 180 I=1,*
        IF(DABS(CX(I)).GT.CXMAX) CXMAX=DABS(CX(I))
180     CONTINUE
        TYPE "CXMAX: ",CXMAX
        IF(CXMAX.GT.CXMIN) GOTO 210
        CXMIN=CXMAX
        DO 190 I=1,8
        XCX(I)=X(I)
190     CONTINUE
        DO 200 I=1,*
        CXCX(I)=CX(I)
        LAMBDACX(I)=LAMBDA(I)
200     CONTINUE
210     TYPE "CXMIN: ",CXMIN
        PHI=OBJ
        DO 220 I=1,*
        PHI=PHI-LAMBDA(I)*CX(I)+0.5D0*SIGMA*CX(I)*CX(I)
220     CONTINUE
        IF(PHI.GT.PHIOP) GOTO 250
        PHIOP=PHI
```

```
       OBJOP=OBJ
       NORMXOP=NORMX
       DO 230 I=1,*
       CXOP(I)=CX(I)
230    CONTINUE
       DO 240 I=1,8
       XOP(I)=X(I)
240    CONTINUE
       ISTEP=ISTEP*ACCF
       GOTO 280
250    DO 260 I=1,*
       CX(I)=CXOP(I)
260    CONTINUE
       DO 270 I=1,8
       X(I)=XOP(I)
270    CONTINUE
       OBJ=OBJOP
       NORMX=NORMXOP
       ISTEP=ISTEP/DECF
280    DO 290 I=1,8
       DXOBJ(I)=0.0D0
290    CONTINUE
       DO 310 J=1,8
       DO 300 I=1,8
       DXOBJ(J)=DXOBJ(J)+R(I,J)*X(I)
300    CONTINUE
       DXOBJ(J)=DXOBJ(J)+OBJ*X(J)
       DXOBJ(J)=(-2.0D0*DXOBJ(J))/NORMX
310    CONTINUE
       DO 320 I=1,8
       JACOB(1,I)=(2.0D0*(X(9-I)-CX(1)*X(I)))/NORMX
320    CONTINUE
       DO 340 K=1,n-1
       DO 330 I=1,8
       JACOB(K+1,I)=MA(K,9-I)/DSQRT(NORMX)-(CX(K+1)*X(I))/NORMX
330    CONTINUE
340    CONTINUE
       DO 360 L=0,**-1                                    (1)
              L=1,**                                      (2)
       DO 350 I=1,8
       JACOB(n+1+L,I)=MA(2*L+1,I)/DSQRT(NORMX)-(CX(n+1+L)*X(I))/NORMX   (1)
       JACOB(n+L,I)=MA(2*L,I)/DSQRT(NORMX)-(CX(n+L)*X(I))/NORMX         (2)
350    CONTINUE
360    CONTINUE
       DO 380 J=1,8
       DXPHI(J)=DXOBJ(J)
       DO 370 I=1,*
       DXPHI(J)=DXPHI(J)-LAMBDA(I)*JACOB(I,J)+SIGMA*CX(I)*JACOB(I,J)
370    CONTINUE
380    CONTINUE
       DO 390 I=1,8
       X(I)=X(I)-ISTEP*DXPHI(I)
390    CONTINUE
       DO 400 I=1,8
```

```
        TYPE X(I)
 400    CONTINUE
        TYPE "LAGRANGIAN VALUE:",PHI
        TYPE "OBJECTIVE VALUE: ",OBJ
        IF(N.GT.NITER) GOTO 3000
        WRITE(10,1)
        GOTO 2000
 3000   IF(CXMIN.LT.TCX) GOTO 4000
        WRITE(10,1)
        GOTO 5000
 4000   ACCEPT "Do you want to continue (Yes=1,No=0) ?",M
        IF(M.EQ.0) GOTO 6000
        ACCEPT "Enter new precision: ",TCX
        WRITE(10,2)
 5000   CALL FOPEN(25,"LAMBn")
        WRITE BINARY(25) LAMBDACX
        CALL FCLOS(25)
        CALL FOPEN(25,"BASnP")
        WRITE BINARY(25) XCX
        CALL FCLOS(25)
        DO 410 I=1,*
        LAMBDA(I)=LAMBDA(I)-SIGMA*CX(I)
 410    CONTINUE
        GOTO 1000
 6000   ACCEPT "Do you want last X (0) or best CX (1) ?",M
        IF (M.EQ.0) GOTO 7000
        DO 420 I=1,*
        TYPE "CX: ",CXCX(I)
 420    CONTINUE
        DO 430 I=1,8
        X(I)=XCX(I)
 430    CONTINUE
        GOTO 8000
 7000   DO 440 I=1,*
        TYPE "CX: ",CX(I)
 440    CONTINUE
 8000   NORMX=0.0D0
        DO 450 I=1,8
        NORMX=NORMX+X(I)*X(I)
 450    CONTINUE
        NORMX=DSQRT(NORMX)
        DO 460 I=1,8
        X(I)=X(I)/NORMX
 460    CONTINUE
        CALL FOPEN(20,"BASn")
        WRITE BINARY(20) X
        CALL FCLOS(20)
 1      FORMAT(1X,13("<13>"))
 2      FORMAT(1X,15("<13>"))
        STOP
        END

 C      JANUARY 22,1985
```

REFERENCES

[1] N. Ahmed, T. Natarjan, and K. R. Rao, "Discrete cosine transform," IEEE Trans. Comput., vol. C-23, pp. 90-93, Jan. 1974.

[2] H. C. Andrews and W. K. Pratt, "Fourier transform coding of images," Proc. Hawaii Int. Conf. Syst. Sci., Jan. 1968, pp. 677-678.

[3] J. G. Bernstein, "Properties and applications of the short-space Fourier transform," MSEE Thesis, MIT, May 1984.

[4] D. P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods, Computer Science and Applied Mathematics, Academic Press, 1982.

[5] W. H. Chen and W. K. Pratt, "Scene adaptive coder," IEEE Trans. Commun., vol. COM-32, No. 3, Mar. 1984, pp. 225-232.

[6] W. H. Chen and C. H. Smith, "Adaptive coding of monochrome and color images," IEEE Trans. Commun., vol. COM-25, pp. 1285-1292, Nov. 1977.

[7] D. E. Dudgeon, R. M. Mersereau, Multi-Dimensional Digital Signal Processing, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[8] R. C. Gonzalez, P. Wintz, Digital Image Processing, Addison-Wesley, 1977.

[9] A. Habibi, "Hybrid coding of picturial data," IEEE Trans. Commun., Vol. COM-22, pp. 614-624, May 1974.

[10] A. Habibi, "Survey of adaptive image coding techniques," IEEE Trans. Commun., vol. COM-25, pp. 1275-1284, 1977.

[11] A. Habibi, "An adaptive strategy for hybrid image coding," IEEE Trans. Commun., Vol. COM-29, pp. 1736-1740, Dec. 1981.

[12] A. Habibi and P. A. Wintz, "Image coding by linear transformation and block quantization," IEEE Trans. Commun. Technol., vol. COM-19, pp. 50-62, Feb. 1971.

[13] B. L. Hinman, "Theory and applications of image motion estimation," MSEE Thesis, MIT, May 1984.

[14] B. L. Hinman, J. G. Bernstein, and D. H. Staelin, "Short-space Fourier transform image processing," Proc. IEEE ICASSP (San Diego, CA), Mar. 1984

[15] J. J. Y. Huang and P. M. Schultheiss, "Block quantization of correlated Gaussian random variables," IEEE Trans. Commun. Syst., vol. CS-11, pp. 289-296, Sept. 1963.

[16] A. K. Jain, "A sinusoidal family of unitary transforms," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-1, Oct. 1979.

[17] A. K. Jain, "Image data compression: A review," Proc. IEEE, vol. 69, pp. 349-389, Mar. 1981.

[18] J. R. Jain, A. K. Jain, "Displacement measurement and its application to interframe image coding," IEEE Trans. Commun., Vol. COM-29, pp. 1799-1808, Dec. 1981.

[19] P. A. Maragos, R. W. Schafer, and R. M. Mersereau, "Two-dimensional linear prediction and its application to adaptive predictive coding of images," IEEE Trans. ASSP, Vol. ASSP-32, pp. 1213-1229, Dec. 1984.

[20] J. Max, "Quantizing for minimum distortion," IRE Trans. Inform. Theory, vol. IT-6, pp. 7-12, Mar. 1960.

[21] A. Z. Meiri, "The pinned Karhunen-Loeve transform of a two-dimensional Gauss-Markov field," Proc. SPIE Conf. on Image Processing, San Diego, CA, 1976.

[22] A. Z. Meiri, E. Yudilevich, "A pinned sine transform image coder," IEEE Trans. Commun., Vol. COM-29, pp. 1728-1735, Dec. 1981.

[23] M. Miyahara, K. Kotani, "Block distortion in orthogonal transform coding - Analysis, minimization and distortion measure," IEEE Trans. Commun., Vol. COM-33, pp. 90-96, Jan. 1985

[24] A. N. Netravali, J. O. Limb, "Picture coding: A review," Proc. IEEE, Vol. 68, pp. 336-406, Mar. 1980.

[25] M. J. D. Powell, Nonlinear Optimization, 1981, New-York, Academic Press and NATO Scientific Affairs Division, 1982.

[26] W. K. Pratt, Digital Image Processing, Wiley, New York, 1977.

[27] W. K. Pratt, "Spacial transform coding of color images," IEEE Trans. Commun. Technol., vol. COM-19, pp. 980-992, Dec. 1971.

[28] W. K. Pratt, W. H. Chen, and R. Welch, "Slant transform image coding," IEEE Trans. Commun., vol. COM-22, pp. 1075-1093, Aug. 1974.

[29] W. K. Pratt, J. Kane, and H. C. Andrews, "Hadamard transform image coding," Proc. IEEE, vol. 57, pp. 58-68, Jan. 1969.

[30] K. R. Rao, M. A. Narasimhan, and K. Revuluri, "Image data processing by Hadamard-Haar transform," IEEE Trans. Comput., vol. C-24, pp. 888-896, Sept. 1975.

[31] R. C. Reeve, J. S. Lim, "Reduction of blocking effects in image coding," Proc. IEEE ICASSP (Boston, MA), 1983.

[32] J. A. Roese, et. al., "Interframe transform coding and predictive coding methods," Proc. International Conf. on Commun., (San Francisco, CA), Vol. 2, pp. 17-21, June 1975.

[33] J. A. Roese, W. K. Pratt, and G. S. Robinson, "Interframe cosine transform image coding," IEEE Trans. Commun., pp. 1329-1339, Nov. 1977.

[34] J. A. Roese, G. S. Robinson, "Combined spatial and temporal coding of digital image sequences," Proc. SPIE, Vol. 66, pp. 172-180, Aug. 1975.

[35] J. M. Schumpert, R. J. Jenkins, "A two-component image coding scheme based on two-dimensional interpolation and the discrete cosine transform," Proc. IEEE ICASSP (Boston, MA), 1983.

[36] A. Segall, "Bit allocation and encoding of vector sources," IEEE Trans. Inform. Theory, Vol. IT-22, pp. 162-169, Mar. 1976.

[37] M. Tasto and P. A. Wintz, "Image coding by adaptive block quantization," IEEE Trans. Commun. Technol., vol. COM-19, pp. 956-972, 1971.

[38] A. Tescher, "A dual transform coding algorithm," Proc. Nat. Telecommun. Conf., 1980.

[39] Z. Wang and B. R. Hunt, "The discrete cosine transform - A new version," Proc. IEEE ICASSP (Boston, MA), 1983.

[40] R. Wilson, H. E. Knutsson, and G. H. Granlund, "Anisotropic nonstationary image estimation and its applications: Part II-Predictive image coding," IEEE Trans. Commun., vol. COM-31, pp. 398-406, Mar. 1983.

[41] P. A. Wintz, "Transform picture coding," Proc. of IEEE, vol. 60, No. 7, July 1972, pp. 802-820.

[42] J. K. Yan, D. J. Sakrison, "Encoding of images based on a two-component source model," IEEE Trans. Commun., Vol. COM-25, pp. 1315-1322, Nov. 1977.

[43] L. H. Zetterberg, S. Ericsson, and H. Brusewitz, "Interframe DPCM with adaptive quantization and entropy coding," IEEE Trans. Commun., Vol. COM-30, pp. 1888-1899, Aug. 1982.