9-30-1989

Adaptive transform coding of video with motion compensation and vector quantization

Jung-Hui Chien
New Jersey Institute of Technology

Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

Title of Thesis: Adaptive Transform Coding of Video with Motion Compensation and Vector Quantization

Jung-Hui Chien, Master of Science in Electrical Engineering, 1989

Thesis directed by: Assistant Professor Dr. Ali N. Akansu

This thesis proposes an efficient transform coding scheme for the low bit rate video sequence. The redundancy within adjacent video frame is exploited by motion compensated interframe prediction using the efficient independent orthogonal search technique, and the motion compensated frame difference (MCFD) signal is transform coding by 1-D DCT, 1-D HDCT and 2-D DCT, the coefficients are zonal masked, and vector quantizatied adaptively. The encoded MCFD as well as the motion information and variance grouping (side informations) are transmitted to the receiver to reconstruct the frame. The transform MCFD signal is classified into the four groups according to the coefficient band variance. A multi-codebook vector quantizer is designed based on all groups. The overall average $\overline{\text{PSNR}}$ for 1-D DCT, 1-D HDCT and 2-D DCT are 36.14dB, 36.26dB, 36.30dB respectively. The results are obtained for the "CINDY" video sequences with the average bit rate 0.3318, 0.3397, 0.2655 bits/pixel, over 39 frames. It is shown that the vector codebook on the set of training sequence generated by the MCFD signal is more independent, general than the original video signal. It is also shown that the performance of 1-D HDCT technique for MCFD signal coding is as good as 2-D DCT technique. Scene change detector algorithm based on cross correlation criterion is proposed in this thesis for the "MIX" video sequence.

Adaptive Transform Coding of Video with Motion Compensation and Vector Quantization

by
Jung-Hui Chien

Thesis submitted to the Faculty of the Graduate School of
the New Jersey Institute of Technology in partial fulfillment of
the requirements for the degree of
Master of Science in Electrical Engineering
1989

APPROVAL SHEET

Title of Thesis : Adaptive Transform Coding of Video with Motion Compensation and Vector Quantization

Name of Candidate : Jung-Hui Chien

Master of Science in Electrical Engineering, 1989

Thesis and Abstract Approved : _____ 9/30/89
Date

Dr. Ali N. Akansu, Advisor
Assistant Professor
Department of Electrical Engineering

8/30/89
Date

Dr. Chang Lu
Assistant Professor
Department of Electrical Engineering

8/31/89
Date

Dr. Nirwan Ansari
Assistant Professor
Department of Electrical Engineering

New Jersey Institute of Technology

VITA

Name: Jung-Hui Chien

Temporary address:

Permanent address:

Degree and date to be conferred: Master of Science in Electrical Engineering, 1989

Date of birth:

Place of birth:

Secondary education: Pien Chiao High School, Pien Chiao, Taiwan, R.O.C. 1976

Collegiate institutions attended	Dates	Degree	Date of Degree
New Jersey Institute of Technology	9/87-10/89	M.S.E.E	Oct, 1989
Fu Jen Catholic University	9/77-6/81	B.S.E.E	June, 1981

Major: Electrical Engineering

Position held:

Sept/87-Oct/89 Research Assistant
Center for Communications and Signal
Processing Research, NJIT

Dec/83-July/87 Assistant Engineer
VIDAR-SMS CO., LTD. Tamsui, Taiwan, R.O.C.

To my parents

ACKNOWLEDGEMENTS

I am deeply indebted to my advisor, Dr. Ali N. Akansu, for the valuable guidance and advice in this research. His constant encouragement and constructive criticism contributed to the successful completion of this work, and in the furthering of my academic and professional career. Working with him was a unforgettable experience.

I am grateful to Dr. Chang Lu and Dr. Nirwan Ansari for their help and time in reviewing this work.

Thanks and regards to my parents and Pearl Kuo for their moral support and love. Thanks also to following people Kadur, Chris, Paresh, MingDeh, Ben, Steve, Shiue, for their supports and encouragements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In digital image processing, "Data Compression" is still desirable in order to meet an operational requirement under an existing system performance constraint, such as limited bandwidth or to realize a cost saving in the design of a new system. Therefore all researcher efforts are toward finding efficient image coding methods and the implementation of these coding algorithms with relatively inexpensive, compact, and high speed very large scale integrated (VLSI) circuits.

A digital version of the standard NTSC (National Television Systems Committee) of monochrome signals, have spatial resolution of approximately 512×512 pels per frame. At 8-bits per pixel intensity resolution and 30 frames per second, this translates into a rate of nearly 60×10^6 bits/s. Depending on the application digital image raw data rates typically vary from 10^5 bits/s to 10^8 bits/s. The large memory and channel capacity requirements for digital image transmission and storage make it mandatory to consider data compression technique.

Image transmission and storage applications are in teleconferencing, video telephone, broad television and satellite image transmission, etc.. Because of their wide applications, data compression and coding schemes have been of great importance in digital image processing.

Image data compression methods can be classified in two basically different categories [1]. In the first category are those methods which exploit *redundancy* in the data. The second category, compression is achieved by an energy preserving transformation of the given image into another array such that maximum information is packed into a minimum number of samples. These two methods are known as redundancy reduction and entropy reduction techniques respectively. Other image data compression algorithms exist which use a combination of these two methods.

Among the various compression techniques, Transform coding is known to be generally superior moreover, with the evolution and widespread use of VLSI, transform coders offer practical means of data compression.

In transform coding, the input signals are mapping into a new coordinate system called the transform domain, where the coefficients are much less correlated. These transforms, being unitary, conserve the signal energy in the transform domain, typically most of this energy is concentrated in low "frequency" samples. Compression is achieved by considering these high energy samples to be sufficient for reconstruction subsequent to transmission, storage, or processing.

For video conferencing, motion in the entire scene is usually low. Therefore interframe coding techniques can reduce the information redundancy in video sequences. Combined with intraframe coding, a high compression ratio can be achieved by incorporating motion detection techniques and then sending only the motion information through the communication channel. In this thesis a free error channel is assumed and only the source encoding is considered.

For interframe coding, the method of predicting the motion compensated interframe is employed by block match algorithm (BMA) with independent orthogonal search technique, therefore the redundancy within adjacent video frame could be ex-

ploited. The block motion compensated coding schemes are encoding only the block differences in the moving areas between the current frame and the prediction based on the motion compensation. The difference frame is called Motion Compensated Frame Difference signal (MCFD).

By Shannon's rate-distortion theory [3] [4], It shows that the vector quantization performs more efficiently than the scalar quantization [5]. Vector Quantization in its simple form is a mapping of a vector of signal samples into a vector that is selected from a fixed and finite set of vectors. The efficiency of VQ comes from its role as a pattern-matching technique. The vectors of samples is a pattern that will be approximated by one of the finite set of prototype patterns. This pattern is described by identifying the address of the pattern in the dictionary of standard patterns that "best" approximates it. This dictionary is called the codebook; the patterns in the codebook are called codevectors; the addresses in the dictionary are called codewords.

There have been several encoding techniques reported in the literature to encode the MCFD signal [14][24][26][26][27]. This study presents a new transform coding; *Hadamard matrix decomposition Discrete Cosine Transform,* which performed with one dimension on MCFD signal. The transform coefficients are zonal masked and vector quantization applied adaptively according to the band variances. It is clear that 1-D transform is simpler and requires less number of operations than 2-D transforms. It also provides more dense structure for parallel processing which saves operation delay times.

The scene change problem is considered, whenever abrupt scene change occurred the new frame is encoded by still image coding mode. Two criteria of scene change detector, *Frame difference variance* and *Cross correlation coefficient,* are proposed in this study as well.

The thesis is organized as follows. Chapter 2 explains the theory of Discrete Cosine Transform coding with zonal masking. Chapter 3 discusses motion compensation of interframe images with an efficient block matching algorithm. Chapter 4 discusses the adaptive vector quantization. LBG algorithm [6] which is used for the generation of codebooks is given. In Chapter 5, a scene change detector algorithm is proposed. In chapter 6, the simulation results are given and finally, chapter 7 is the conclusion.

Chapter 2

Transform Coding (TC)

Transform coding involves linear transformation in which the signal source, a block of N input samples $x(n)$ are linearly transformed into a set of N transform coefficients $\theta(n)$. The coefficients are then vector quantized for transmission, or stored, and the reconstruction of $x(n)$ is obtained at the receiver performing an inverse transform operation on quantized coefficients. The transform coding scheme diagram is shown in Figure 2.1.

TC technique has been found theoretically and experimentally to have relatively good capability of bit rate reduction. There are two main reasons for this. First, packs the signal energy into a minimum number of coefficients, and thus not all of the transform domain coefficients need to be transmitted in order to maintain a good image quality. Secondly, generate a set of uncorrelated coefficients so that coding can be done more efficiently.

2.1 Discrete Linear Orthogonal Transforms

An orthogonal $N \times N$ matrix A has the following property:

$$AA^{-1} = I$$

where I is the $N \times N$ identity matrix and A^{-1} called inverse of A.

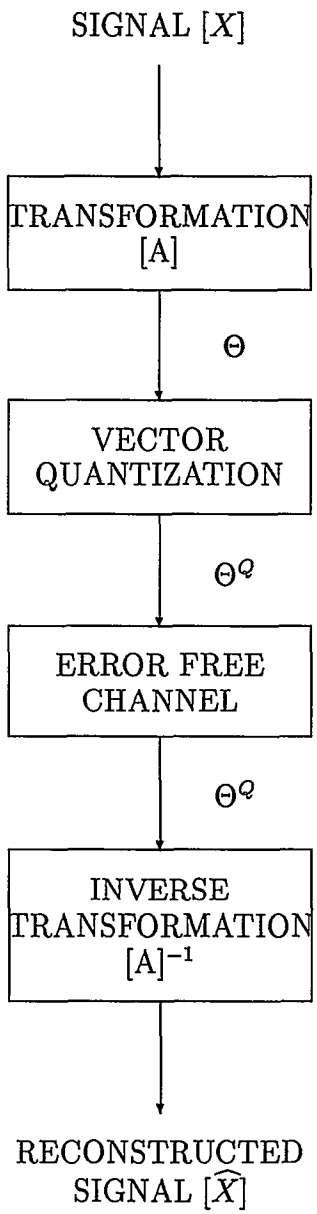


Figure 2.1: Transform Coding

In the complex case, matrix A is called unitary matrix is defined as below:

$$AA^H = I, \quad \text{and} \quad A^H A = I, \quad \text{and} \quad A^H = A^{-1}.$$

The matrix A^H is referred as "A Hermitian".

2.1.1 One Dimensional Transform Coding

The one-dimensional linear transform of N_{th} order sequence

$$\{x(n)\}; \quad \text{for} \quad n = 0, 1, \ldots, N-1$$

is given as

$$\theta(k) = \sum_{n=0}^{N-1} x(n)a(k,n) \quad \text{for} \quad k = 0, 1, \ldots, N-1$$

where $a(k,n)$ is the forward transformation kernel, and the $\theta(k)$ are the transform coefficients. The transform coefficients are quantized. The inverse transformation that reconstruct the input sequence is given as

$$\widehat{x}(n) = \sum_{k=0}^{N-1} \theta^Q(k)b(k,n) \quad \text{for} \quad n = 0, 1, \ldots, N-1$$

where $b(k,n)$ is the inverse transformation kernel, and $\theta^Q(k)$ is the quantized coefficients.

The matrix representation is given as

$$[\Theta] = [A][X] \tag{2.1}$$

$$[\Theta^Q] = Q[\Theta]$$

and

$$[\widehat{X}] = [A]^{-1}[\Theta^Q] \tag{2.2}$$

where $A^{-1} = A^T$ for orthonormal transforms.

2.1.2 Two Dimensional Transform Coding

Two dimensional forward and inverse transforms are given as

$$\theta(k,l) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} x(m,n)a(k,l,m,n);$$

and

$$\widehat{x}(m,n) = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} \theta^{Q}(k,l)b(k,l,m,n)$$

The equations describe the transform of $N \times N$ data array with the forward transform kernel $a(\bullet)$ and the inverse transform kernel $b(\bullet)$. If the 2-D transform is separable, the kernels $a(\bullet)$ and $b(\bullet)$ are separable into the row and column operations as shown in Figure 2.2

$$\begin{aligned} a(k,l,m,n) &= a_v(k,m)a_h(l,n) \\ b(k,l,m,n) &= b_v(k,m)b_h(l,n) \end{aligned}$$

Hence, in order to obtain $\theta(k,l)$, it can be conveniently performed in two steps. Each one involves a one dimensional transform operation. The first step uses $a_h(\bullet)$ for operation on row m to prevail the l_{th} transformed coefficient, $\theta(m,l)$. While the second step utilizes $a_v(\bullet)$ to provide the one-dimensional transform of column l;

$$\begin{aligned} \theta(k,l) &= \sum_{m=0}^{N-1} a_v(k,m) \sum_{n=0}^{N-1} x(m,n)a_h(l,n) \\ &= \sum_{m=0}^{N-1} a_v(k,m)\theta(m,l) \end{aligned}$$

x and θ are arrays which have their elements $x(m,n)$ and $\theta(k,l)$ respectively. We may write the matrix notation of the two-dimensional transform as

$$[\Theta] = [A_v][X][A_h]^T$$

for the symmetric kernels:

$$A_v = A_h = A;$$

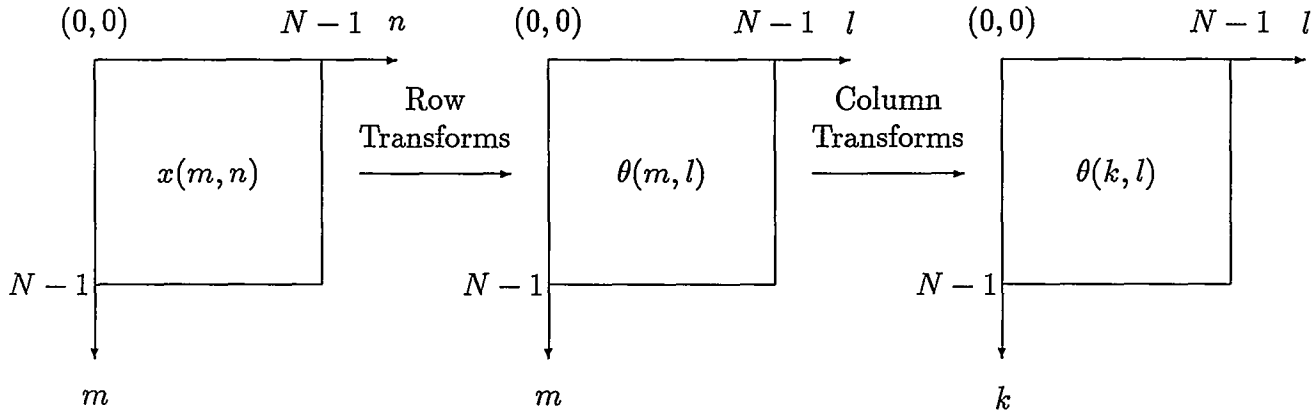


Figure 2.2: Computation of the 2-D transform as a series of 1-D transform

$$[\Theta] = [A][X][A]^T \tag{2.3}$$

and

$$[\widehat{X}] = [A]^T[\Theta^Q][A] \tag{2.4}$$

where $A^{-1} = A^T$

2.2 Properties of Transform Coding

Two important features of image transform coding employing the orthogonal matrix (unitary matrix) are highlighted as;

- The average energy of the transform coefficients is equal to the average energy of the input signal; that is, orthogonal transforms are variance preserving. If A is an orthogonal matrix ($A^{-1} = A^T$), the average sum of variances $E[\theta^2(k)] = \sigma_k^2$

of the transform coefficients $\theta(k)$ equals the variance of the input;

$$\begin{aligned}\frac{1}{N}\sum_{k=0}^{N-1}\sigma_k^2 &= \frac{1}{N}\sum_{k=0}^{N-1}E[\theta^2(k)] = \frac{1}{N}E[\Theta^T\Theta] = \frac{1}{N}E[X^TA^TAX] \\ &= \frac{1}{N}E[X^TX] = \frac{1}{N}\sum_{k=0}^{N-1}E[x^2(k)] = \frac{1}{N}\sum_{k=0}^{N-1}\sigma_x^2 = \sigma_x^2 \qquad (2.5)\end{aligned}$$

When the input signal has zero mean.

- The average reconstruction error variance is equal to the average quantization error variance in the transformation domain. The total reconstruction error between input image X and reconstruction image $\widehat{X}$ introduced by the individual quantizer Q_i is given by

$$\sigma_{rk}^2 = E[(x(k) - \hat{x}(k))^2] \quad \text{for} \quad k = 0, 1, \ldots, N-1$$

where σ_{rk}^2 is the *mean square error (MSE)* of the element k of X.

The average reconstruction error variance is

$$\sigma_r^2 = \frac{1}{N}\sum_{k=0}^{N-1}\sigma_{rk}^2 = \frac{1}{N}E[(X - \widehat{X})^T(X - \widehat{X})] \qquad (2.6)$$

Clearly, $r = X - \widehat{X}$, is the N block of error samples and σ_r^2 is the variance average over that block. In keeping with $X = A^{-1}\Theta$ for unquantized quantities, and $A^{-1} = A^T$, *Eq.*(2.6) can be rewritten as

$$\sigma_r^2 = \frac{1}{N}E[(\Theta - \Theta^Q)^T(\Theta - \Theta^Q)] = \frac{1}{N}E[Q^TQ] = \frac{1}{N}\sum_{k=0}^{N-1}\sigma_{qk}^2 = \sigma_q^2 \qquad (2.7)$$

where $q = \Theta - \Theta^Q$ is the block of quantization error.

The result in *Eq.*(2.5) and (2.7) can be straight forwardly extended to include complex entries in A and for two dimensional signals generalize to averages divided by N^2 elements.

2.3 Suboptimum Transform

Many different types of unitary transforms, including the Karhunen-Loève (KL), Fourier, Discrete Sines, Discrete Cosine, Walsh-Hadamard, Slant, Harr, Modified Hermite transforms, have been presented for signal coding [7][8][9][10][11]. The KL transform has been considered as an optimum transformation. However, for the reasons of computation and source data dependency, it is impractical. One of the most attractive from the stand point of coding performance and implementation simplicity is the **Discrete Cosine Transform (DCT)** [10], that the theoretical mean square error coding performance equivalent to that obtained by the optimum KL transform [9].

2.3.1 Discrete Cosine Transform (DCT)

The primary purpose of DCT is to convert statistically dependent signal samples into somewhat independent coefficients. The DCT coefficients are corresponding to the real and symmetric terms of the discrete Fourier transform (DFT) coefficients of digital arrays. The DCT was proposed by Ahmed, Natarajan and Rao in 1974 [10], the one-dimensional length, N discrete cosine transform (1-D DCT) is given as

$$\theta(k) = \sqrt{\frac{2}{N}} C(k) \sum_{n=0}^{N-1} x(n) \cos \frac{(2n+1)k\pi}{2N}$$

$$\text{for} \quad k = 0, 1, \ldots, N-1$$

where

$$C(k) = \frac{1}{\sqrt{2}} \quad k = 0$$

$$C(k) = 1 \quad k > 0$$

and the inverse DCT is defined as

$$\widehat{x}(n) = \sqrt{\frac{2}{N}} C(k) \sum_{k=0}^{N-1} \theta^{Q}(k) \cos \frac{(2n+1)k\pi}{2N}$$

$$\text{for} \quad n = 0, 1, \ldots, N-1$$

As mentioned earlier, the principal advantage of the separability property is that two dimensional discrete cosine transform (2-D DCT) can be obtained in two steps by consecutive applications of the 1-D DCT.

A Hadamard matrix Discrete Cosine transformation (HDCT) is proposed in this study. The $2N \times 2N$ Hadamard matrix decomposition obtained from $N \times N$ seed matrix is given by

$$A_{2N} = \frac{1}{\sqrt{2}} \begin{pmatrix} A_N & A_N \\ A_N & -A_N \end{pmatrix}$$

where $A_{2N} A_{2N}^{-1} = I$, and $A_{2N}^{-1} = A_{2N}^{T}$.

The base matrix of size 32×32 is obtained by employing 8×8 modules in this study.

2.3.2 Subpicture Size

Transform coding takes advantage of the statistical dependence of picture elements for entropy reduction. The MSE performance of TC should improve while increasing the size of the block K, since the number of correlations taken into account increase with K [23]. Most pictures contain significant correlations between pixels for only about 20 adjacent pixels, although this number is strongly dependent on the amount of detail in the picture. However, the improvement is not significant as subpicture size is increased beyond 16×16 [12].

In this study for 2-D DCT 8×8 subpicture size, and for 1-D DCT, 1-D HDCT on 32 sample arrays (scan from 4×8 subpicture size) are used.

2.4 Zonal Sampling (Zonal Mask)

As mentioned earlier, TC packs the maximum amount of variance in the few coefficients, thus we can apply a zonal mask on the transform coefficients and quantize only the significant elements for transmission or storage, moreover it reduces the bit rate as well. This method is also called zonal coding [1].

The zonal mask is chosen based on transform coefficient variances. In 1-D DCT and 1-D HDCT, we select 16 dominant coefficients out of 32 sample array, and 25 coefficients out of 64 for 2-D case. The detail of zonal masks for 1-D DCT, 1-D HDCT and 2-D DCT will be discussed in chapter 6.

2.5 Fidelity Measure

A standard objective measure of image coding quality is the peak to peak signal to noise ratio, which is widely used in literature as the performance criterion, is defined as

$$\text{PSNR(dB)} = 10 \log_{10} \left(\frac{255^2}{\sigma_{mse}^2} \right) \tag{2.8}$$

where σ_{mse}^2 is the average mean square error (MSE) between the original image X_{mn}, and the reconstructed image $\widehat{X}_{mn}$.

Chapter 3

Motion Compensated Video Coding

The frame to frame redundancy of video sequences is exploited in interframe coding. Some video coding technique employ the frame difference coding but more sophisticated ones include the motion compensation and motion compensated frame difference rather than frame difference is encoded. It has been found that the human observer can tolerate poorer spatial resolution in the moving areas of video and a poorer temporal (frame-to-frame) resolution in the nonmoving areas. Due to this fact, we can use fewer bits per pixel, or even subsample the pixels in the moving areas (also called dot interlacing) and skip frame-to-frame pixels for the nonmoving areas to keep the overall bit rate down. There are two types of interframe coders, called (1) frame replenishment, and (2) motion compensation.

1. **Frame Replenishment** [13] In this type of interframe coding, pixels are differentiated between those that change and those that do not change from frame to frame. If a pixel does not change, it is repeated in the next frame at the receiving side. If a pixel does change, it is replenished by the amount it changes. An important trade-off exits between spatial and temporal resolution in this type of interframe coding.

2. **Motion Compensation** [13] In this interframe coding technique, the image is assumed as a superposition of moving pixels on a stationary image. The challenge is to find the ways of describing the trajectories of the moving pixels so that each moving pixel can then be propagated along with its trajectory at the reconstruction. Only the first frame and the trajectory information could be sufficient for reconstruction of a video sequence, if motion is estimated perfectly.

3.1 Block Motion Tracking Algorithm [14]

One popular technique to estimate the motion displacement is the *block matching approach*, which is used in this study. In this technique each video frame is divided into fixed size subblocks, typically 8×8, and the detection of motion is then performed on each subblock, by motion detector, if subblock is moving, then the displacement estimation is performed to find the displacement vector (motion information) by a block matching algorithm.

Several efficient block matching algorithms have been proposed, such as Exhaustive Search (Brute Force Search) which requires extensive computations, 2-D logarithmic Search, 3-Step Direction Search, Modified version of the former, Conjugated Direction Search (CDS), and One-at-a-Time Search (OTS) [15][16][17][18].

An efficient search technique called *Independent Orthogonal Search* [14] is used in this study which minimizes the computations. The tests performed on the video sequence "CINDY" shown that the performance of the orthogonal search is practically the same as the brute force search algorithm.

A motion compensated video coding system typically consists of three stages:

1. A motion detector which detects the moving blocks.

2. A displacement estimator, which estimates the displacement vector of moving

blocks.

3. A data compression algorithm which encodes the interframe differences, motion compensated frame difference, after motion compensation.

3.2 Motion Detector

A motion detector is first used to classify a block to be moving or non-moving, before any motion estimation employed for the block. It screens the original image blocks for motion so that the displacement estimator, which usually requires much more calculations than the motion detector, works only on a smaller number of blocks which are actually moving.

It also helps in reducing the blocks that would be erroneously classified as moving by the displacement estimator due to uncertainty, when blocks are in low detail region and/or have random camera noise. The motion detector is thus necessary to ensure a zero displacement for the unchanged region. The motion detector compares each pixel of a predefined subblock in the present frame K, with the corresponding pixel value of the previous frame (reconstructed version) $K-1$. A pixel at location (x,y) in frame K is said to be moving if

$$| F_K(x,y) - F_{K-1}(x,y) | \geq T_0 \tag{3.1}$$

where pixels $F_K(\cdots)$ and $F_{K-1}(\cdots)$ are in frames K and $K-1$ respectively. A block is said to be moving if the number of moving pixels in that block is greater than or equal to N_0. In this study the parameters, T_0 =3 and N_0 =10, are found suitable for blocks of size 8×8 and 8-bit pixels. These thresholds were also reported as the suboptimums [14].

3.3 Displacement Estimator

The goal of the displacement estimator is to find the best match of a reference block from frame K, in a suitable area of *Search Region (SR)* in the previous frame (reconstruct frame) $K-1$. The detectable displacement is assumed to be less than $\pm p$ pixel, along the horizontal and vertical axes. If the size of a reference block is $M \times N$, then the search area is $(M+2p) \times (N+2p)$.

The problem consists of seeking a best match location by evaluating a matching criterion at every point in the *SR* of $(2p+1)^2$ points. Each point in the *SR* is a candidate for being the correct displacement vector and will be called a *search point.*

The best match is based on some specified criteria such as the minimum mean square error *(MSE)*, number of thresholded absolute differences *(NTAD)*, and minimum mean absolute frame difference *(MAD)*. The *MAD* is used as the matching criterion in this study since its performance on the frame differential entropy reduction is about the same as the more involved *MSE* criterion.

The *MAD* objective function $D(\bullet)$ at any search point (i,j), in the search region is defined as

$$D(i,j) = \frac{1}{64} \sum_{x=1}^{8} \sum_{y=1}^{8} \mid F_K(x,y) - F_{K-1}(x+i\,,\, y+j) \mid \qquad -p \leq i,j \leq +p \quad (3.2)$$

The search point $D(i,j)_{\min}$ which has $\min\{D(i,j)\}$ defines the motion information for that subblock.

We also assume image data satisfies the *quadrant-monotonic* model proposed by Jain and Jain [36] and the orthogonal search technique is employed to decrease the computational burden.

Quadrant-Monotonic Model definition [14]

Suppose $O = (x_O, y_O)$ is the optimum search point, and $A = (x_A, y_A)$ is any other

point in SR. A function $D = (x, y)$ is called *quadrant monotonic*, if $D(B) < D(A)$ for any point $B = (x_B, y_B) \in SR$ that satisfies the following conditions:

1. B and A belong to the same quadrant with respect to O, that is, $(x_B - x_O)$ and $(y_B - y_O)$ has the same sign as $(x_A - x_O)$ and $(y_A - y_O)$, and

2. either

   $| x_B - x_O | < | x_A - x_O |$ and $| y_B - y_O | \leq | y_A - y_O |$

   or

   $| x_B - x_O | \leq | x_A - x_O |$ and $| y_B - y_O | < | y_A - y_O |$

The following properties are derived based on the above quadrant-monotonic model assuming that (1) $O = (x_O, y_O)$ is the optimum (minimum) points in the SR, and (2) two distinct search points $A = (x_A, y_A)$ and $B = (x_B, y_B)$ have been placed in SR, and $D(A) > D(B)$.

- Property 1 If $y_A = y_B$ and $x_A > x_B$, then O can*NOT* exit in the half plane defined by $\{ (x, y) \in SR \mid x \geq x_A \}$.

- Property 2 If $x_A > x_B$ and $y_A > y_B$, then O can*NOT* exit in the quadrant defined by $\{(x, y) \in SR \mid x > x_A \text{ and } y > y_A\}$.

The above properties were shown in Figures 3.1 (a) and (b). The rules are still valid for three search points or more. By examining every pair of search points, we can eliminate sequentially the regions that prohibit the optimum point. this technique, known as (interval) elimination in optimization theory [46], is legitimate here because $D(\bullet)$ is quadrant-monotonic. The remaining area, which contains the optimum point, will be called *region of uncertainty (RUC).*

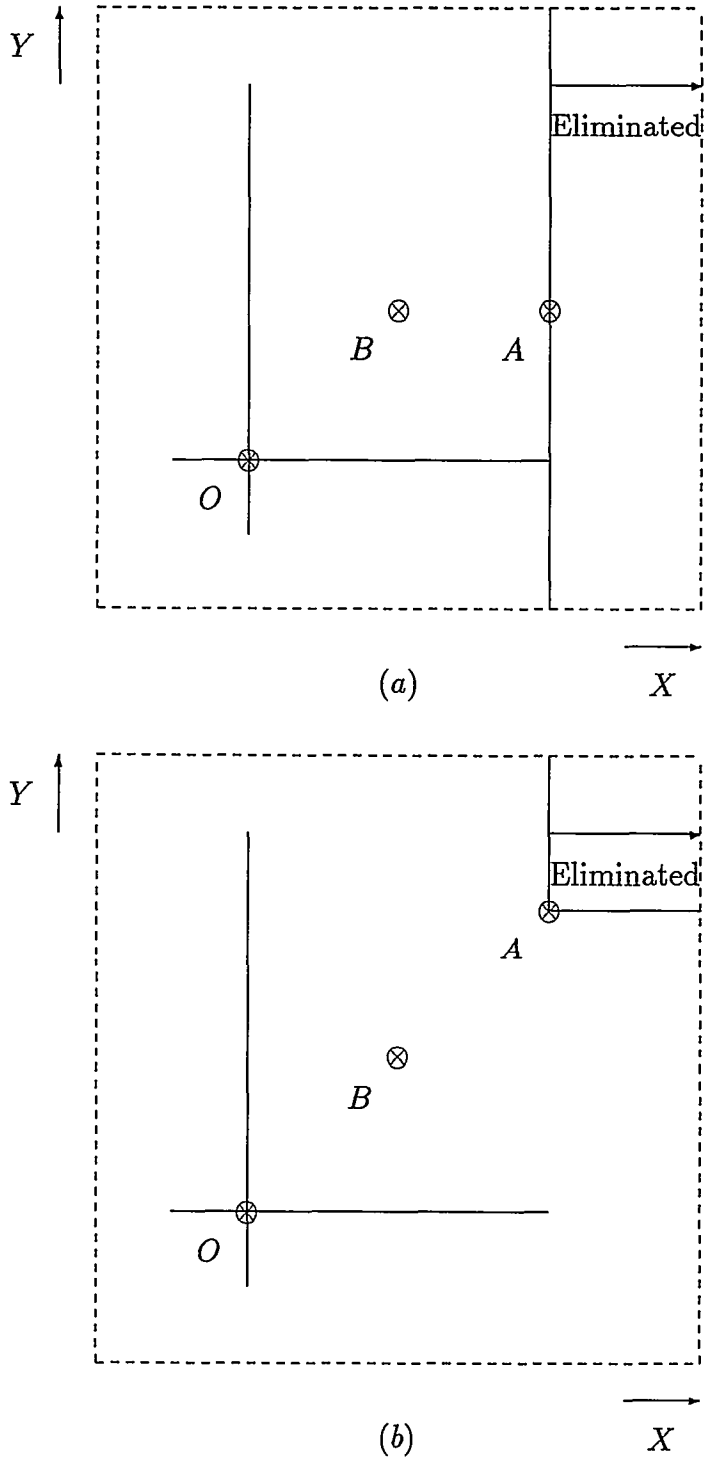


Figure 3.1: Properties of Quadrant-Monotonic Model

3.3.1 Orthogonal Search Algorithm

A set of requirements desired for a good search algorithm, even though any realistic algorithm cannot achieve all the goals simultaneously because of the inherent conflicts among them, for instance, convergence, fewer search points , fewer search steps, noise immunity, hardware complexity. The primary goal of the *independent orthogonal search algorithm* is to minimize the total number of search points in the worst case, though, we are also concerned with the average behavior when an algorithm is applied to real image data.

Algorithm:

Initialization:

Search region. $SR = (2p+1) \times (2p+1)$

Step Number i=1

Initial Step Size $k = \left[\frac{p}{2}\right]$

$k = \left[\frac{p}{2}\right]$ denotes the smallest integer that is greater than $\frac{p}{2}$.

a. **Step 1:** Three search points are placed horizontally in the center of SR. The distance between every two neighboring points equals to the Step Size as shown in Figure 3.2. The minimum point will be selected as the center for the next step. Step Number $i \leftarrow (i+1)$.

b. **Vertical Step:** Two more search points are placed vertically around the minimum from the previous step. The distance between every two neighboring points equals to the Step Size as shown in Figure. 3.2. The minimum point will be the center at the next step.

c. Stopping Rule: The remaining region of uncertainty now has an area $4(k-1) \times (k-1)$. If $k = 1$, stop. Otherwise, $k \leftarrow \left[\frac{k}{2}\right]$, $i \leftarrow (i+1)$, and continue.

d. Horizontal Step: Two more search points are placed horizontally around the minimum from the Vertical Step. The minimum point will be the center for the next step. $i \leftarrow (i+1)$, and go back to **b.**

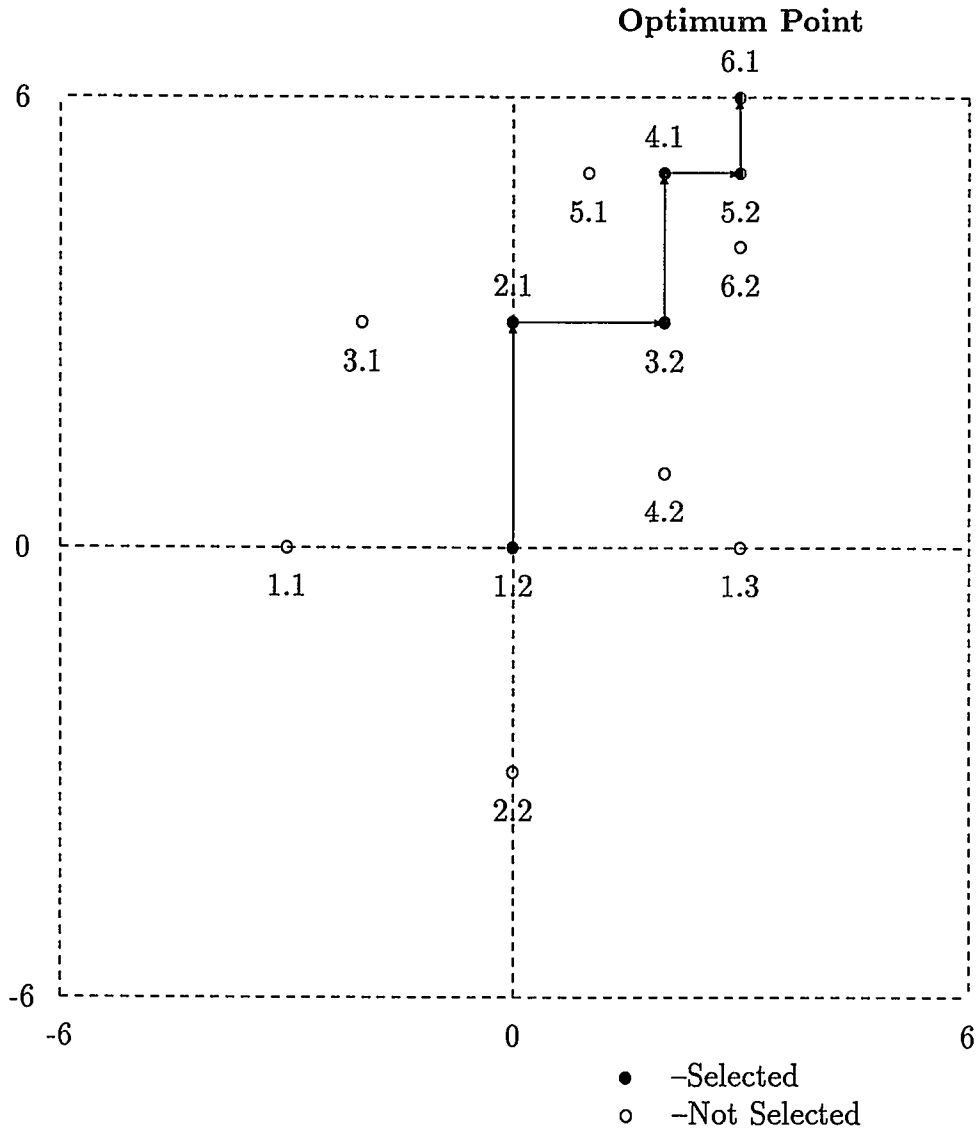


Figure 3.2: Orthogonal Independent Search

Chapter 4

Vector Quantization

It has been stated that quantization is the most common form of data compression. There are two basic approaches to design quantizer : (1) We are given a fixed number of quantization levels and we wish to minimize the "quantization noise" (or distortion). (2) We are given a fixed quantization noise or distortion, and asked to minimize the average number of bits per sample that will meet the distortion specification. There are basically three forms of quantization :

1. Scalar quantization
2. Vector quantization
3. Sequential quantization

According to Shannon's rate distortion theory [3][4], although the data source is memoryless, better performance can be achieved by coding vectors rather than scalar. So, instead of quantizing each sample individually with the same quantizing function, a sequence or block of N samples is quantized together. The representing vector or sequence is the closest one in the set of quantizer vectors, vector codebook. A distortion measure is used for the purpose. The vector quantization algorithms for reducing the transmission or the storage bit rate has been adopted for speech and image signal coding [5][6][19].

There are several vector codebook design algorithms, which are based on the simple observation that Lloyd's [20] basic development is also valid for vectors, with known distribution and a variety of distortion measures. An efficient algorithm developed by Linde, Buzo and Gray [6] known as the LBG algorithm is used to design the vector codebooks in this thesis.

The algorithm is based on Lloyd's algorithm, it is not a variational technique, and involves no differentiation; hence it works well even when the distribution has discrete components, as is the case when a sample distribution from a training sequence is used. The algorithm produces a quantizer meeting necessary but not sufficient conditions for optimality.

4.1 Definition

An N-level k-dimensional quantizer is mapping, q, that assigns to each input vector, $\mathbf{x} = (x_0, \ldots, x_{k-1})$, a code vector, $\hat{\mathbf{x}} = q(\mathbf{x})$, drawn from a finite codebook, $\widehat{\mathbf{C}} = \{y_i;\ i = 1, ..., N\}$. The quantizer q is completely described by the codebook $\widehat{\mathbf{C}}$ together with the partitions, $S = \{S_i;\ i = 1, \ldots, N\}$, of the input vector space into the sets $S_i = \{\mathbf{x} : q(\mathbf{x}) = y_i\}$ of input vectors mapping into the i^{th} codevector (or codeword, channel index). The quantizer can also be seen as the combination of two functions: an *encoder*, which maps the input vector and generate the index of the codevector specified by $q(\mathbf{x})$ and a *decoder*, which uses this index to produce the codevector $\hat{\mathbf{x}}$.

If $\widehat{\mathbf{C}}$ has N Levels, than $R = \log_2 N$ is called the rate of the quantizer in bits per vector and $r = \frac{R}{k}$ is the rate in bits per sample. Unlike the scalar quantization, general VQ permits fractional rates in bits per pixel. The goal of the quantization system is to produce the "best" possible reproduction sequence for a given rate R.

To quantify this and to define the performance of a quantizer, a distortion measure should be defined.

4.2 Distortion Measure

The distortion between an input vector $\mathbf{x}$ and a reproduction vector $\hat{\mathbf{x}}$ is given by a non-negative distortion measure $d(\mathbf{x}, \hat{\mathbf{x}})$. Given such a distortion measure, we can quantify the performance of a system by an average distortion $E\{d(\mathbf{x}, \hat{\mathbf{x}})\}$ between the input and the reproduction vectors. A system is good if it yields a small average distortion. In practice, the important average is the long term sample average or time average

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n-1} d(\mathbf{x}_i, \hat{\mathbf{x}}_i)$$

provided the limit makes sense. If the vectors process is stationary and ergodic, then with probability one, the limit exists and equals $E\{d(\mathbf{x}, \hat{\mathbf{x}})\}$.

Many distortion measures have been proposed in the literature. The most common and simple distortion measure for image coding, is the squared error distortion measure for reason of mathematical convenience, is given as

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \| \mathbf{x} - \hat{\mathbf{x}} \|^2 = \sum_{i=0}^{k-1} (x_i - \hat{x}_i)^2$$

the input vector $\mathbf{x}$ and the reproduction vector $\hat{\mathbf{x}}$ are in a k- dimensional Euclidean space, and $d(\mathbf{x}, \hat{\mathbf{x}})$ is the square of the Euclidean distance between the vectors. For the squared-error distortion it is common practice to measure the performance of a system by the signal-to-noise ratio (or signal-to-quantization-noise-ratio).

4.3 Encoder

In the absence of transmission errors, the encoder and decoder operations are exactly identical to the quantizer mapping $Q(\bullet)$ shown in Fig 4.1 The input image after the transformation is partitioned at the encoder into small, contiguous, nonoverlapping blocks (vectors) of pixels, and each block is independently quantized as mentioned before. The vector dimension is given by the number of significant coefficients in the block. The codebook $\mathbf{C}$ is composed of vectors $\{y_1, y_2, \ldots, y_n\}$. For any block, its vector can be expressed as $\mathbf{x}_n = (x_1, x_2, \ldots, x_k)$. According to the distortion measure, the channel symbol k can be chosen by

$$d(\mathbf{x}_n, y_k) \leq d(\mathbf{x}_n, y_i) \qquad i \neq k$$

4.4 Decoder

The decoder will have the same codebooks used at the encoder. Using the codebooks as lookup table for VQ decoding, the decoder will extract a particular codevector, addressed by the channel index k_n, as the reproduction vector $\{\hat{\mathbf{x}}_n\}$. Its operation can be written as

$$\hat{\mathbf{x}}_n = \text{codebook}\{y_{k_n}\}$$

the VQ encoder and decoder diagram is shown in figure 4.2

4.5 LBG VQ Design Algorithm [21]

The design algorithm of optimal vector quantizers from empirical data (training sets) were proposed by Linde, Buzo, and Gray [6] using a clustering approach. This algorithm is now commonly referred to as the **LBG** algorithm.

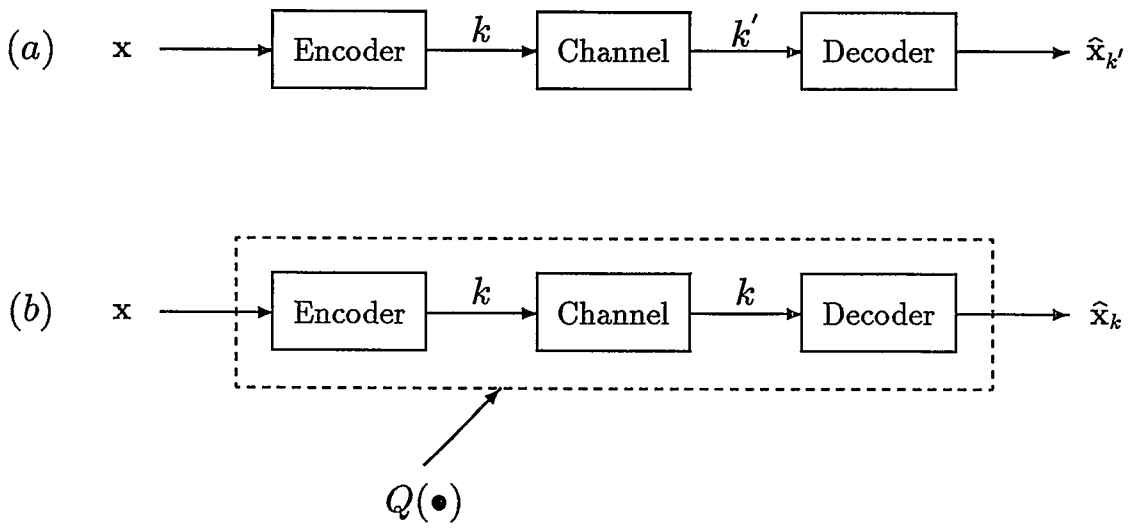


Figure 4.1: Digital transmision (a) with and (b) without transmission errors [22]

The goal in designing an optimal vector quantizer is to obtain a quantizer consisting of N reproduction vectors, such that it minimizes the expected distortion. Lloyd [20] proposed an iterative nonvariational technique known as his "Method I" for the design of scalar quantizers. Recently, Linde *et al*[6]. extended Lloyds' basic approach to the general case of vector quantizers.

Let the expected distortion be approximated by the time-averaged square error distortion given by the expression

$$D(x, q(x)) = \frac{1}{N} \sum_{i=0}^{N-1} d(x_i, \hat{x}_i)$$

The algorithm for an unknown distribution training sequence is given as below.

(1) Let N = number of levels; distortion threshold $\epsilon \geq 0$. Assume an initial N level reproduction alphabet $\widehat{\mathbf{A}}_0$, and a training sequence $(x_j;\ j = 0, 1, \ldots, n-1)$, and m= number of iterations, set to zero.

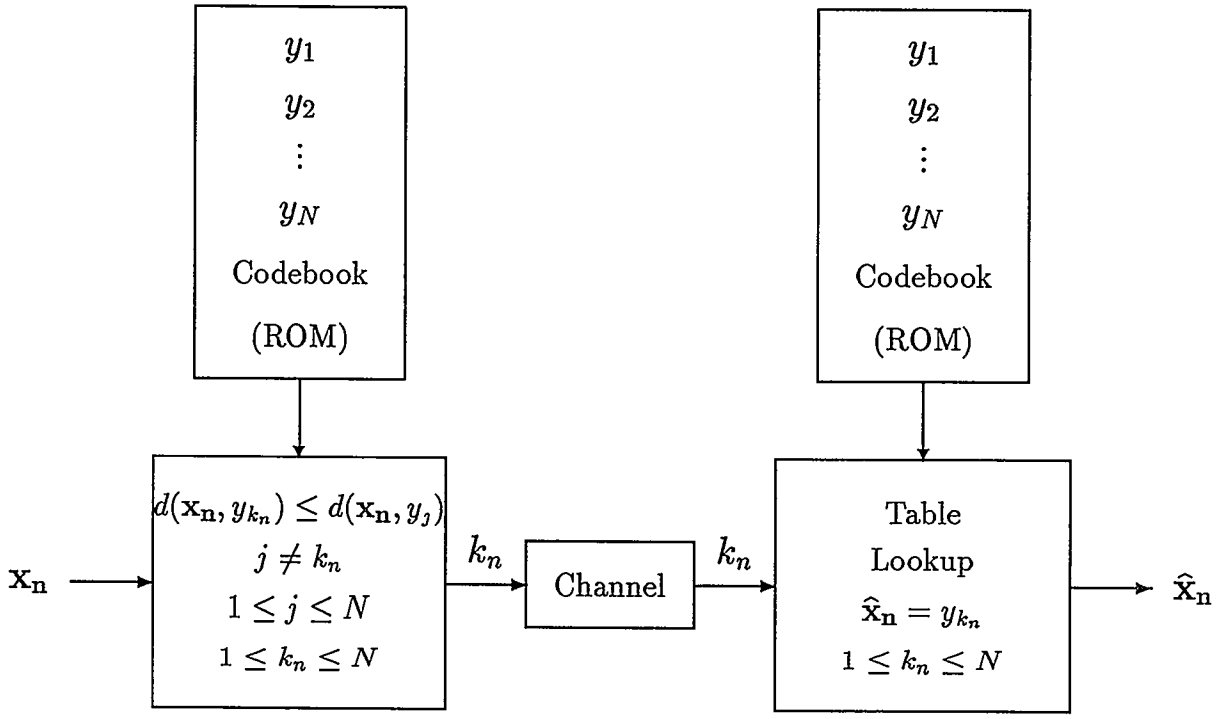


Figure 4.2: VQ Encoder and Decoder

(2) Given $\widehat{\mathbf{A}}_m = (y_i;\ i = 1, \ldots, N)$, find the minimum distortion partition $P(\widehat{A}_m) = (S_i;\ i = 1, \ldots..., N)$ of the training sequence; $x_j \epsilon S_i$ if $d(x_j, y_i) \leq d(S_j, y_l)$, for all l. Compute the average distortion

$$D_m = D[(\widehat{A}_m, P(\widehat{A}_m))] = n - 1 \sum_{j=0}^{n-1} \min_{y \epsilon \widehat{A}_m} d(x_j, y)$$

(3) If $\frac{D_{m-1} - D_m}{D_m} \leq \epsilon$, stop the iteration with $\widehat{\mathbf{A}}_m$ as the final reproduction alphabet; otherwise continue.

(4) Find the optimal reproduction alphabet $\widehat{x}(P(\widehat{A}_m)) = (\widehat{x}(S_i);\ i = 1, \ldots, N)$ for $P(\widehat{A}_m)$ where

$$\widehat{x}(S_i) = \frac{1}{\| S_i \|} \sum_{j: x_j \epsilon S_i}^{m} x_j$$

(5) Set $\widehat{A}_{m+1} = \widehat{x}(S_i)$, increment m to $m + 1$ and goto **(2)**.

In above iterative algorithm an initial reproduction alphabet $\widehat{A}_0$ was assumed in order to start the algorithm. There are a number of techniques to obtain the initial codebook. The simplest technique is to use the first widely spaced words from the training sequence. Linde *et al.* [6] used a splitting technique where the centroid for the training sequence was calculated and split into two close vectors. The centroid or the reproduction vectors for the two partitions were then calculated. Each resulting vector was then split into two vectors and the above procedure was repeated until an N level initial reproduction vector was created. Splitting was performed by adding a fixed perturbation vector ϵ to each vector y_i producing two vectors $y_i + \epsilon$, $y_i - \epsilon$. Another approach for designing initial codebooks is to employ product code techniques [21]. In this approach, a scalar quantizer is used k times successively to yield a k-dimensional vector quantizer.

Chapter 5

Scene Change Algorithm

Many image transmission and storage applications, contain images of moving objects. The motion captured in such a multiframe sequence of images includes translation and rotation of objects with respect to the camera or moving camera and moving objects in some cases. It is easier to deal with the fixed camera and the moving objects case. This problem could be achieved by interframe image coding if the trajectories of various objects were known.

At the present time, the scene change problem is excluded and motion compensation in the same scene with motion is employed. As expected, in real life video applications, abrupt scene change occur. The scene change problem is being studied here, whenever a scene change occurred the encode switch to the still frame coding mode, from the interframe coding mode. Still frame coding performance of the encoder effects the interframe coding performance during the following frames of a new scene. Any good video coding technique should perform well in interframe coding mode as well as the still frame coding mode. The algorithm of scene change is shown in Figure 5.1.

The technique to estimate the scene change employs 5 small picture blocks in each video frame (original frame) and the detection is then performed on each block, if 3

out of 5 blocks are greater/smaller than the threshold, then it is said scene change has occurred. The 5 blocks considered in each frame are fixed and shown in Figure 5.2. The threshold criterion is define in the following section.

5.1 Frame Difference Variance Criterion

The purpose of the criterion is to find the variances of the frame difference, in a reference block between frame K and the previous frame $K-1$. It is assumed that the scene change occurred for blocks A in frame K and $K-1$ if

$$\text{Var}(\Delta A) \geq Vo$$

where ΔA is the block frame difference between frame K and $K-1$ of reference block A. In this study the parameters, $Vo = 200$, found suitable for "MIX", "MIX5", "MIX60" video sequence. Since the technique is image dependent, so we present a unique method in the following section.

5.2 Cross Correlation Criterion

This method is more general, it is more robust for the scene change detection, the goal of the method is to find the cross correlation from frame K to the previous frame (original frame) $K-1$, a scene change for block A is detected if

$$\rho(A_K, A_{K-1}) = \frac{\text{Cov}(A_K, A_{K-1})}{\sigma_{A_K}\sigma_{A_{K-1}}} \leq C_o$$

where $\text{Cov}(A_K, A_{K-1})$ is the covariance of block A_K and A_{K-1} is defined by

$$\text{Cov}(A_K, A_{K-1}) = E(A_K A_{K-1}) - \mu_{A_K}\mu_{A_{K-1}}$$

σ_{A_K} and $\sigma_{A_{K-1}}$ are standard deviations of blocks A_K and A_{K-1} respectively.

The cross correlation ρ is dimensionless, C_o is set to 0.2 in this study.

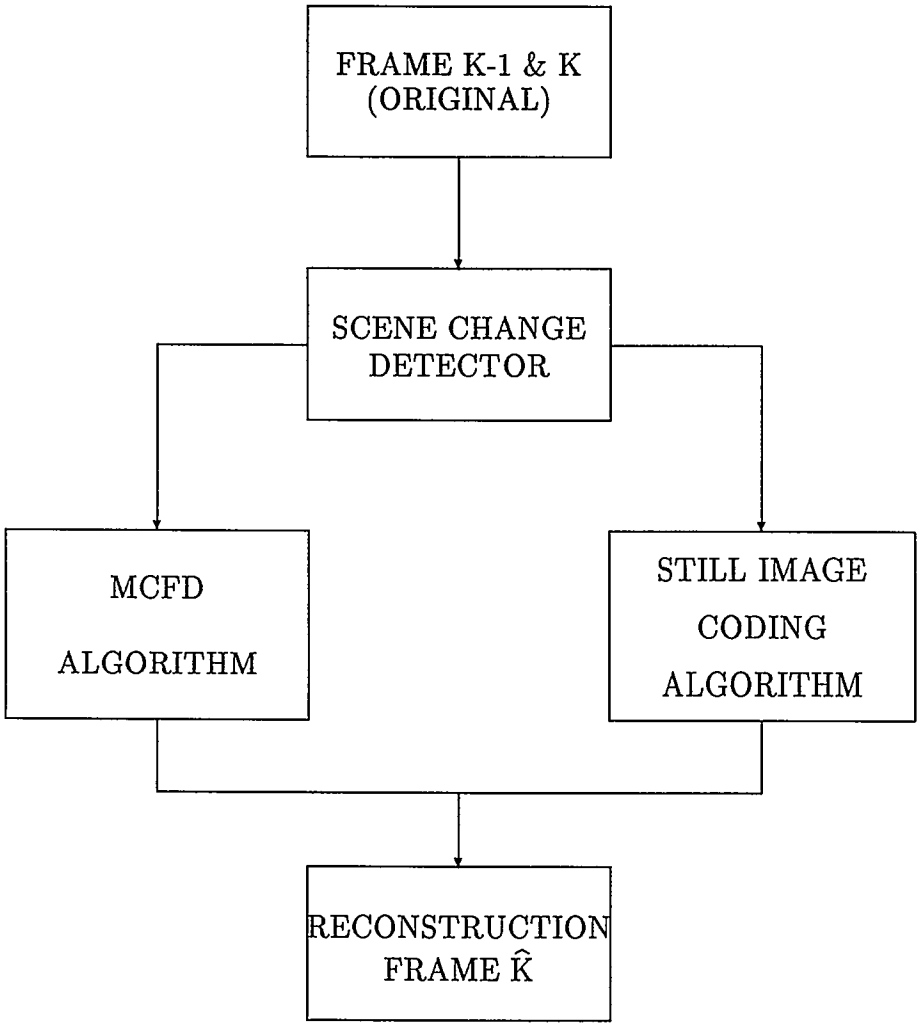


Figure 5.1: Scene Change Algorithm

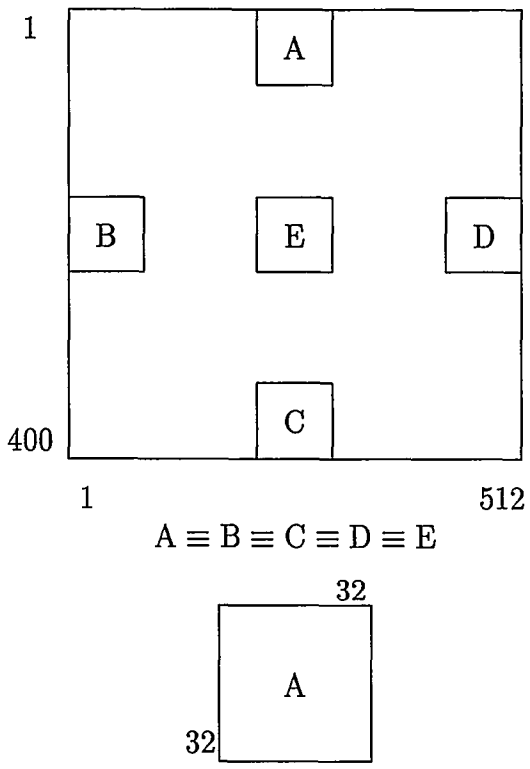


Figure 5.2: The 5 blocks considered in each frame

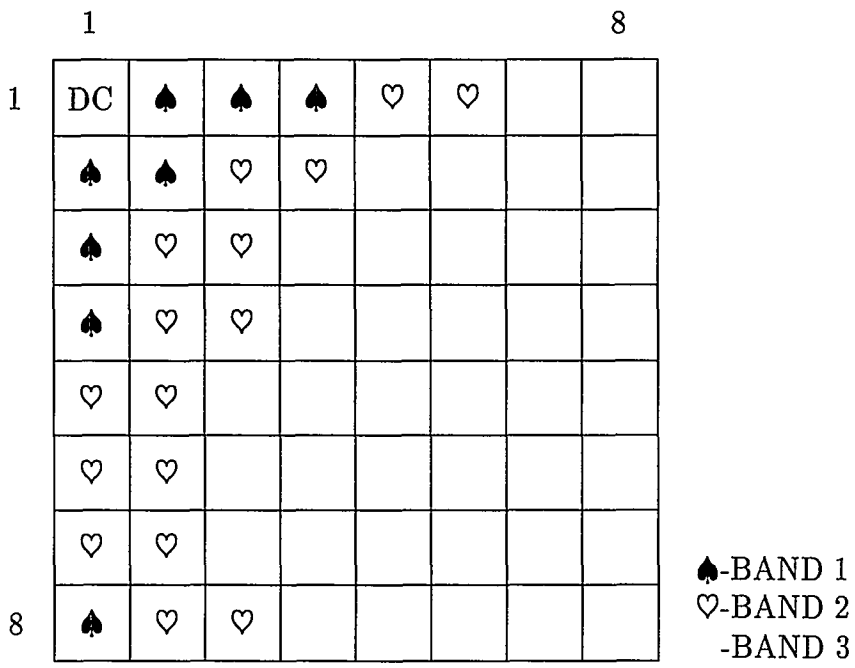


Figure 5.3: Zonal mask of 2-D 8×8 DCT on still image

5.3 Transform Coding of Still Images

The still images are divided into 8×8 blocks and each block undergoes a 2-D 8×8 DCT transformation, then using the zonal mask shown in Figure 5.3, create 3 bands, 8 coefficients in band 1, 16 coefficients in band 2, 39 coefficients in band 3. The DC coefficients kept as it is for the moment. The band 3 is discarded, then Vector Quantization of the two bands performed independently. The simulation results will be presented in chapter 6.

Chapter 6

Simulation Results

A series of computer simulations have been conducted to evaluate the performance of the Adaptive DCT techniques. The software developed was written in Sun-workstation/UNIX/Fortran 77 environment. The test image used in this thesis is 40 frames of highly active monochrome video sequence **"CINDY"** of size 512×400 and 8 bits per pixel. The performance of scene change algorithm is also studied. The image used in scene change algorithm is the 40 frames of monochrome **"MIX"** of size 512×400 and 8 bits per pixel, which consists of 10 frames from each sequence"CINDY", "MONO", "DUO", "QUARTET" respectively. The structure is shown in figure 6.1.

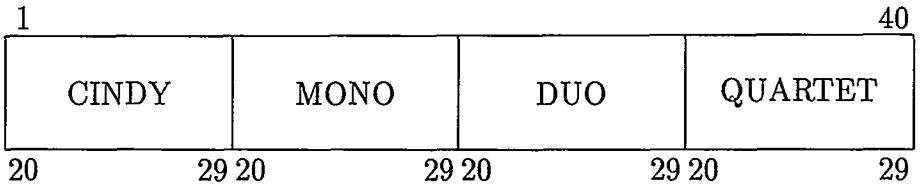


Figure 6.1: "MIX" video sequence structure

6.1 MCFD Signal

The efficient independent orthogonal search algorithm which minimizes the computations is used for block prediction. From Figure 6.2, it is seen that the performance of the independent orthogonal search technique is very close to the brute force search technique . For a typical search where the maximum displacement is limited to ± 6 pels between adjacent frames, the independent orthogonal search requires the least number of search points in the worst case: 13 search points as compared to 169 search points required in the case of the brute force search technique. One good feature of the independent orthogonal search algorithm is its regularity , it requires the same number of search points and search steps no matter where the optimum point is located. The block diagram of motion compensated video coding structure is shown in figure 6.3.

It is found that motion compensation performs quite better than the direct Frame Difference (FD) technique.

The experiment consists of various measured parameters including a histogram, three different entropies, variances and autocorrelation coefficients for the MCFD, direct FD, and "CINDY" signals. The MCFD signal is based on the perfect reconstruction of the previous frame. The histogram of the MCFD signal is shown in figure 6.4. It looks like a Laplacian pdf with most values around zero. The performance results are summarized below.

6.1.1 Entropy

The entropy of the MCFD signal is smaller than the original video by as much as 46% . Also it is less than the direct FD signal by about 0.5 bit on the average as shown in table 6.1. The entropies vs frame index, of MCFD and direct FD for "CINDY"

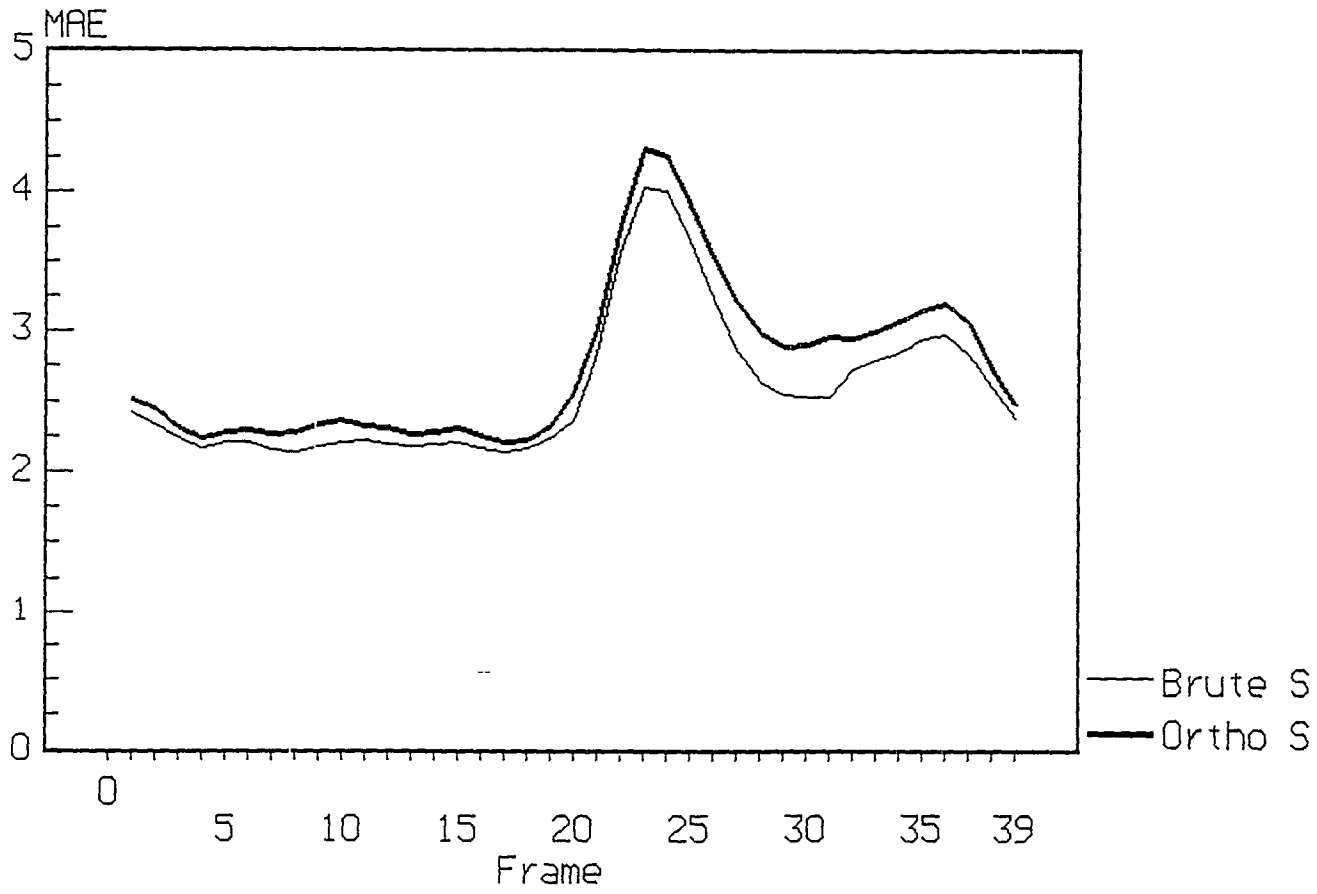


Figure 6.2: Comparison of Independent Orthogonal Search and Brute force technique for MCFD signal with Mean Absolute Difference/pixel as criterion.

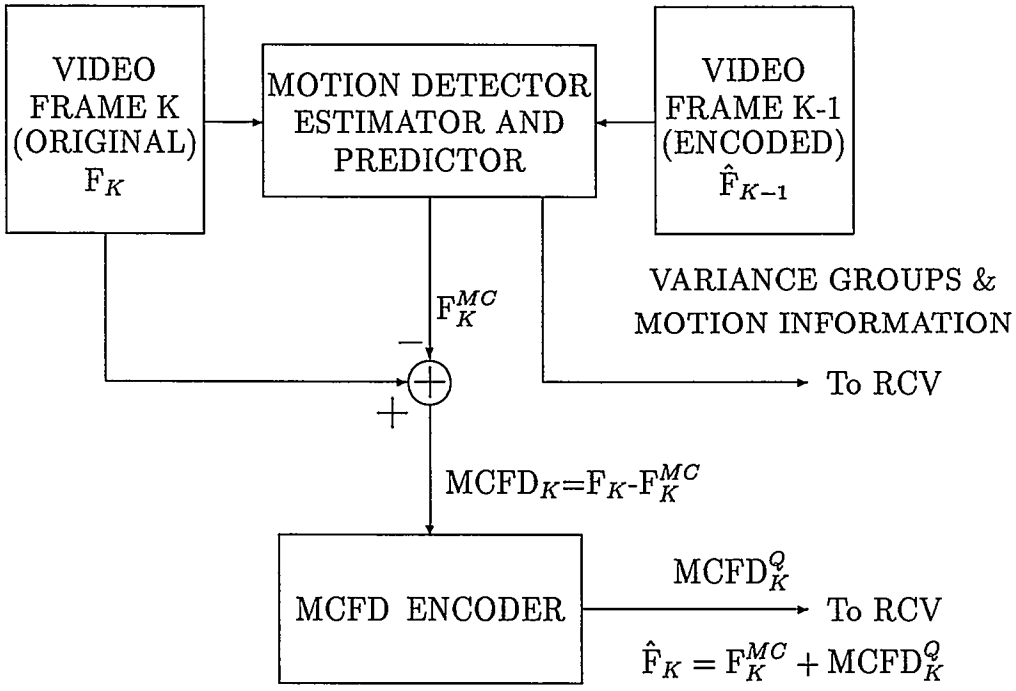


Figure 6.3: Block Diagram of Motion Compensated Video Coding Structure

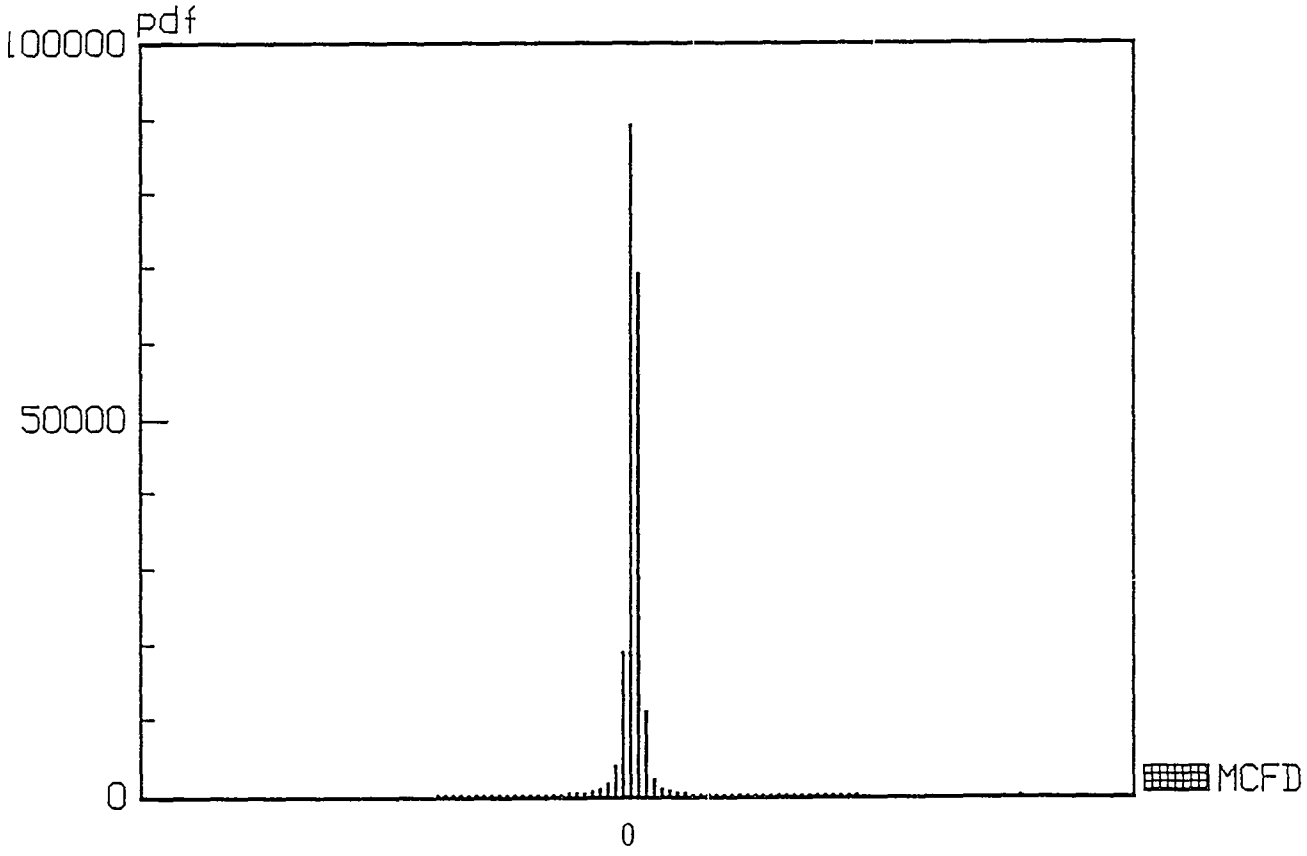


Figure 6.4: Histogram of MCFD signal for "CINDY"

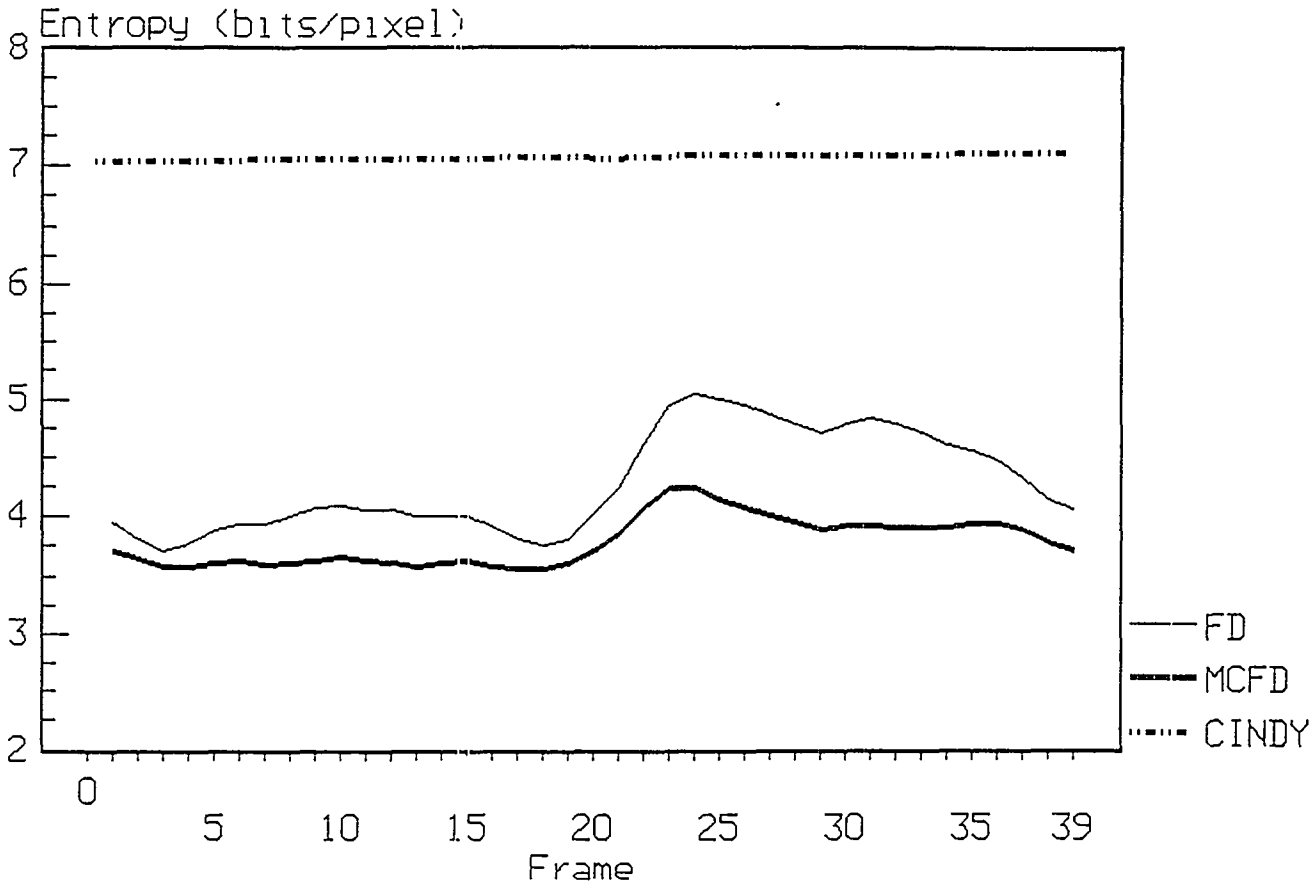


Figure 6.5: Entropy of MCFD, direct FD for "CINDY" sequence

sequence are shown in figure 6.5. The entropy values are based on integer rounded-off values of MCFD and FD.

6.1.2 Variance

The variance of the original signal, "CINDY", ranges around 145, whereas the MCFD variance is around 19 and 103. The variance of the MCFD signal is less than direct FD signal as shown in figure 6.6. This again illustrates that motion compensation works well.

Signal	Entropy (bit/pixel)
CINDY	7.07
MCFD	3.78
direct FD	4.28

Table 6.1: Average entropies over 40 frames of "cindy"

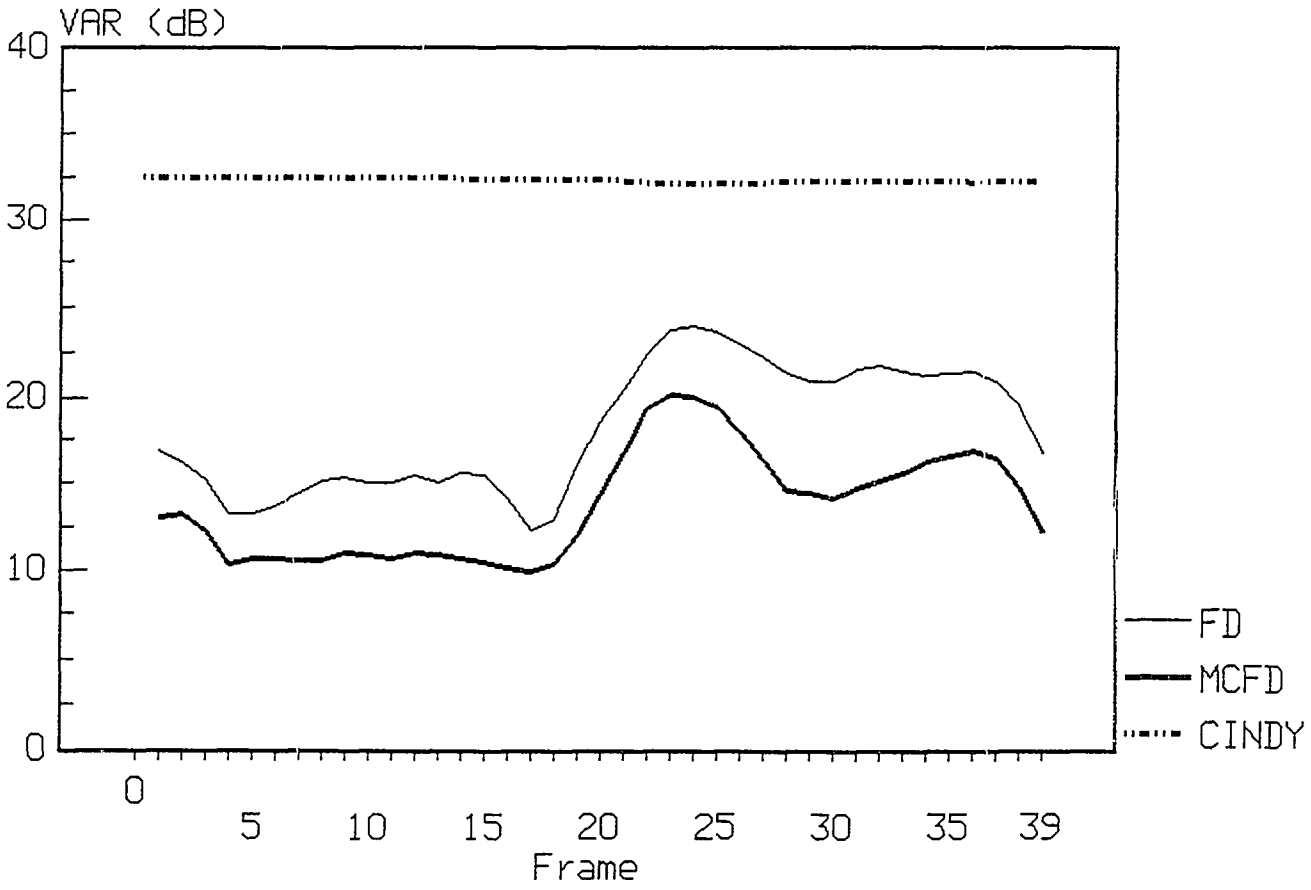


Figure 6.6: Variance of MCFD, direct FD for "CINDY" sequence

Figure 6.7: Horizontal, Vertical Autocorrelation Coefficient of "CINDY" and MCFD Signals

6.1.3 Autocorrelation Coefficient

Auto-regressive, order 1, AR(1), source model is a good first approximation to the speech and image signals [22]. The horizontal and vertical first order autocorrelation coefficients, for the "CINDY" signal is in the range of 0.92 and 0.95. But for the MCFD signal vary in the range of 0.24 and 0.50. The plot of horizontal and vertical auto correlation coefficients against the frame index is shown for original "CINDY" and MCFD signals in figure 6.7.

6.2 Transform Coding of MCFD signal

The MCFD signal is divided into small blocks, and each block employs a transformation to produce the transform coefficients. The transform coefficients are zonal masked and vector quantized adaptively.

Three cases of transformations are used in this study:

(1) 1-D Discrete Cosine Transform

(2) 1-D Hadamard matrix decomposition Discrete Cosine Transform

(3) 2-D Discrete Cosine Transform

The 1-D DCT and 1-D HDCT of MCFD are 32 sample arrays, scan from 4×8 block. The performance is compare with 2-D 8×8 DCT. The results show that 1-D case and 2-D case give about the same transform performance.

6.2.1 Gain of TC [22]

With optimum bit allocation, the gain of one and two dimensional transform coding over PCM has been shown to be the ratio of the arithmetic mean of the transform coefficient variances to the geometric mean as

$$1\text{D } G_{TC} = \frac{\frac{1}{N}\sum_{i=1}^{N}\sigma^2(i)}{\left[\prod_{i=1}^{N}\sigma^2(i)\right]^{\frac{1}{N}}}$$

and

$$2\text{D } G_{TC} = \frac{\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sigma^2(i,j)}{\left[\prod_{i,j=1}^{N}\sigma^2(i,j)\right]^{\frac{1}{N^2}}}$$

This measure shows the compactness power of any transform using a theoretical

Transform	N	G_{TC}
1-D DCT	32	1.963
1-D HDCT	32	2.337
2-D DCT	8	2.407

Table 6.2: Gain of TC

performance analysis technique of the transforms[62], The G_{TC} of 1-D DCT, 1-D HDCT and 2-D DCT cases are found as in the table 6.2. The results indicate that the 1-D 32 sized HDCT and 2-D 8×8 DCT give about the same transform performance for MCFD signal. The 1-D HDCT requires less number of operations and more dense for parallel processing which mean less delay.

6.2.2 Entropy

The average entropies of the transform coefficients MCFD signal with 1-D DCT, 1-D HDCT, and 2-D DCT techniques are close to one another and shown in table 6.3.

6.2.3 Zonal Mask

The fixed zonal coefficient masks of the 3 different transform cases are shown in figures 6.8, 6.9, 6.10. For 1-D case, 2 coefficient bands are used, band 1 and band 2, each band contains 16 coefficients. The band 2 is discarded. For 2-D case, 3 bands are created, 9 coefficients in band 1, 16 coefficients in band 2, 39 coefficients in band 3. The band 3 is discarded.

6.2.4 Generating Training Sequence

The vector codebooks are generated using large set of training vectors known as the training sequences. The coefficient band signals are classified into the four groups according to their variances. The variance grouping and quantizer levels for this

Transform	Entropy (bits/pixel)
1-D DCT	3.62
1-D HDCT	3.57
2-D DCT	3.77

Table 6.3: Entropy of transform MCFD

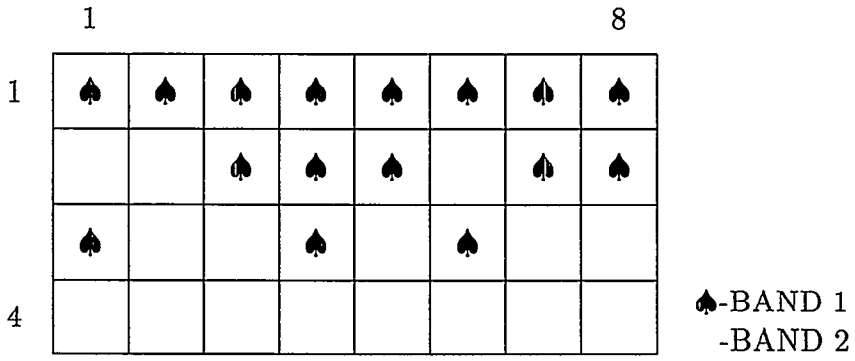


Figure 6.8: 1-D DCT coefficients zonal mask

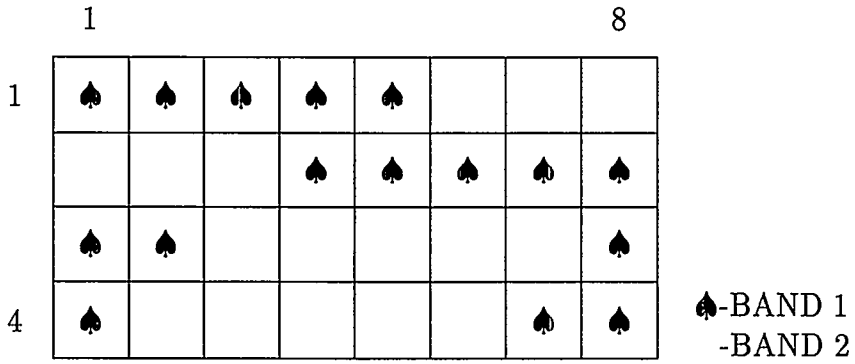


Figure 6.9: 1-D HDCT coefficients zonal mask

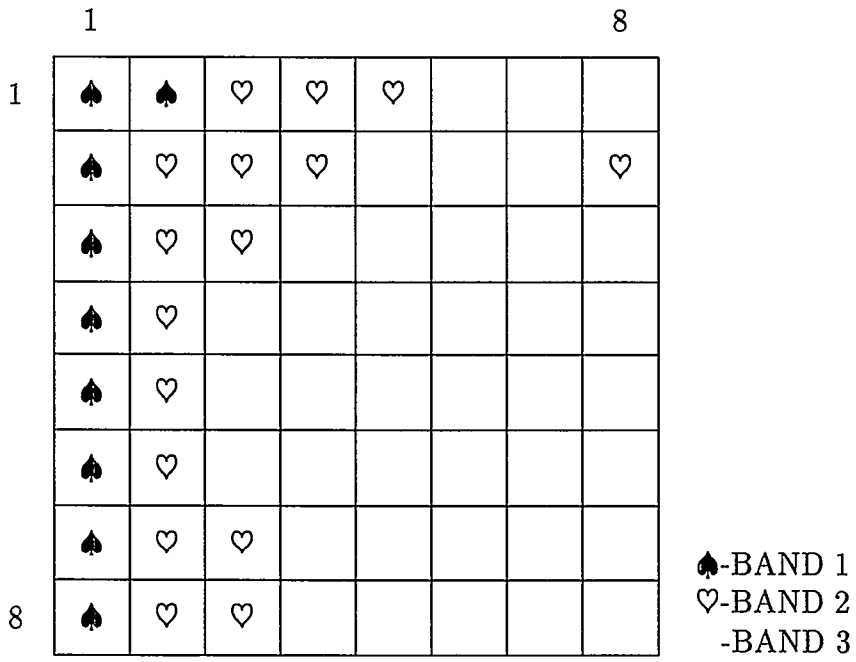


Figure 6.10: 2-D DCT coefficients zonal mask

Group	0	1	2	3
Band variance	0~5	5~10	10~30	>30
Codebook size	discard	1024	1024	1024

Table 6.4: Variance groups and quantizer levels

study are illustrated in table 6.4. The training sequences are generated by coefficient band variance grouping. A multi codebook vector quantizer is designed by employing LBG algorithm [6].

6.2.5 Adaptive Vector Quantization

In adaptive VQ, the coefficient band signals are quantized by tailor made quantizer, according to the variance grouping. The adaptive vector quantization employed here can be summarized as;

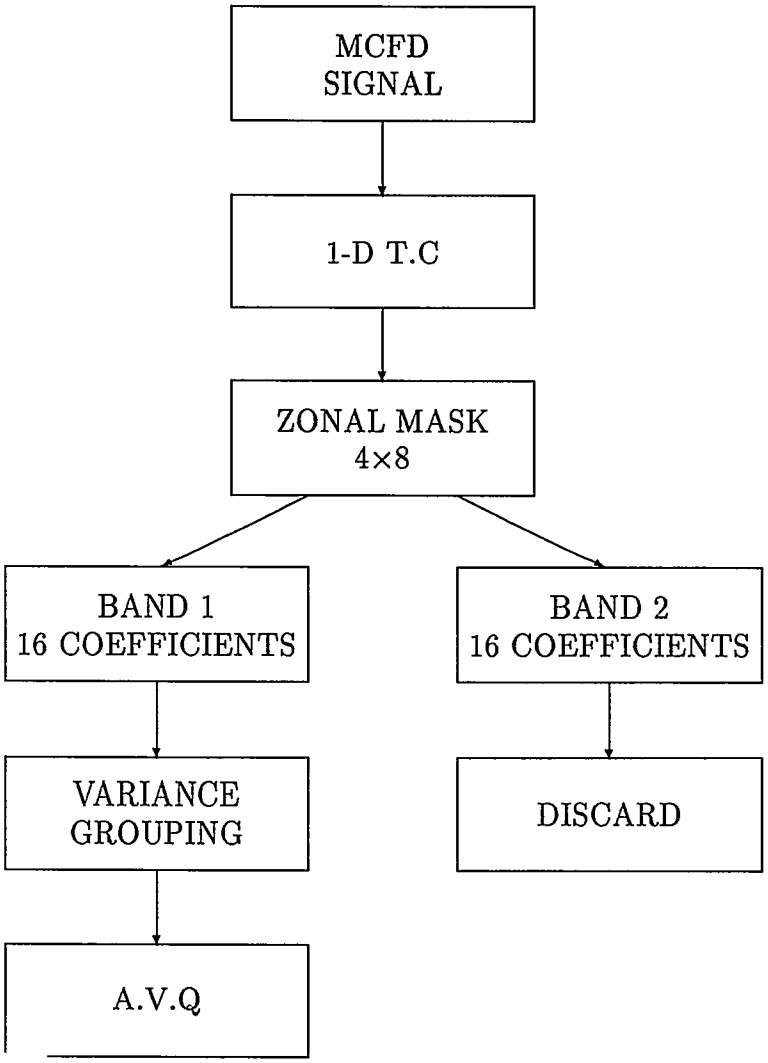
(1) If the variance of a coefficient band is within the variance interval of the group 0, then discard this band, means assume the band has value of zero.

(2) If the variance of a coefficient band is within the variance interval of any group, then use the corresponding vector codebook.

The block diagram of the adaptive VQ of 1-D and 2-D cases are shown in figures 6.11, 6.12. The band signals are encoded, consequently the vector index as well as motion information, variance grouping information are required at the receiver to reconstruct the frame.

6.2.6 Bit Rate

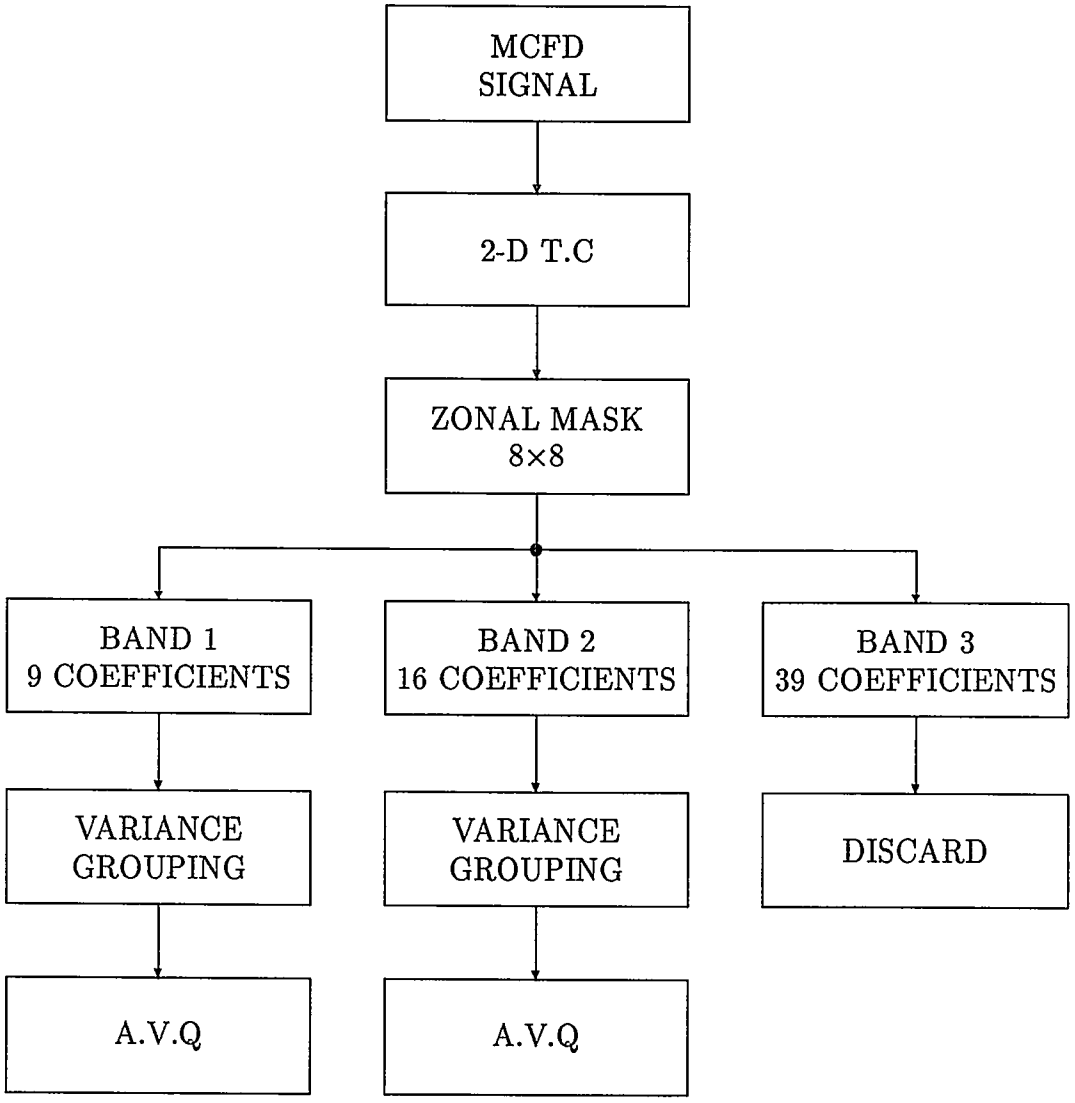
The total bit rate for adaptive transform coding with MC and VQ technique can be written as:



zonal mask refers to figures 6.8, 6.9

variance grouping refers to table 6.4

Figure 6.11: Block diagram for the encoder of 1-D A.V.Q scheme



zonal mask refers to figure 6.10

variance grouping refers to table 6.4

Figure 6.12: Block diagram for the encoder of 2-D A.V.Q scheme

$$\textbf{Bit-Rate} = \mathbf{B_m} + \mathbf{B_{mcfd}} + \mathbf{B_{vg}}$$

where

$\mathbf{B_m}$: Average bits/pixels used for the transmission of motion information. It is fixed and $B_m = 0.125$ bits/pixel in this study.

$\mathbf{B_{mcfd}}$: average bits/pixel used for encoding of MCFD signal. It is variable from frame to frame.

$\mathbf{B_{vg}}$: Average bits/pixel used for the transmission of the variance grouping. $B_{vg} = 0.0625$ bits/pixel and fixed in this study.

The performance criterion PSNR used here is the peak to peak signal noise ratio as defined in equation (2.8)

It was observed that the overall $\overline{\text{PSNR}}$ of the adaptive VQ is more than 1.5 dB better than non-adaptive VQ at the same bit rate.

It was also observed that the overall $\overline{\text{PSNR}}$ performance of 2-D case is little better than the others at the same bit rate.

The PSNR as a function of the frame index of three cases are given in figures 6.13, 6.14, 6.15 and the average Bit-Rate as a function of the frame index of three cases are given in figures 6.16, 6.17, 6.18.

The overall average Bit-Rate and $\overline{\text{PSNR}}$ of three cases are tabulated in table 6.5.

Technique	Bit-Rate (bits/pixel)	$\overline{\text{PSNR}}$(dB)
1-D DCT	0.3318	36.14
1-D HDCT	0.3245	35.77
2-D DCT	0.2655	36.30

Table 6.5: Overall average Bit-Rate and $\overline{\text{PSNR}}$

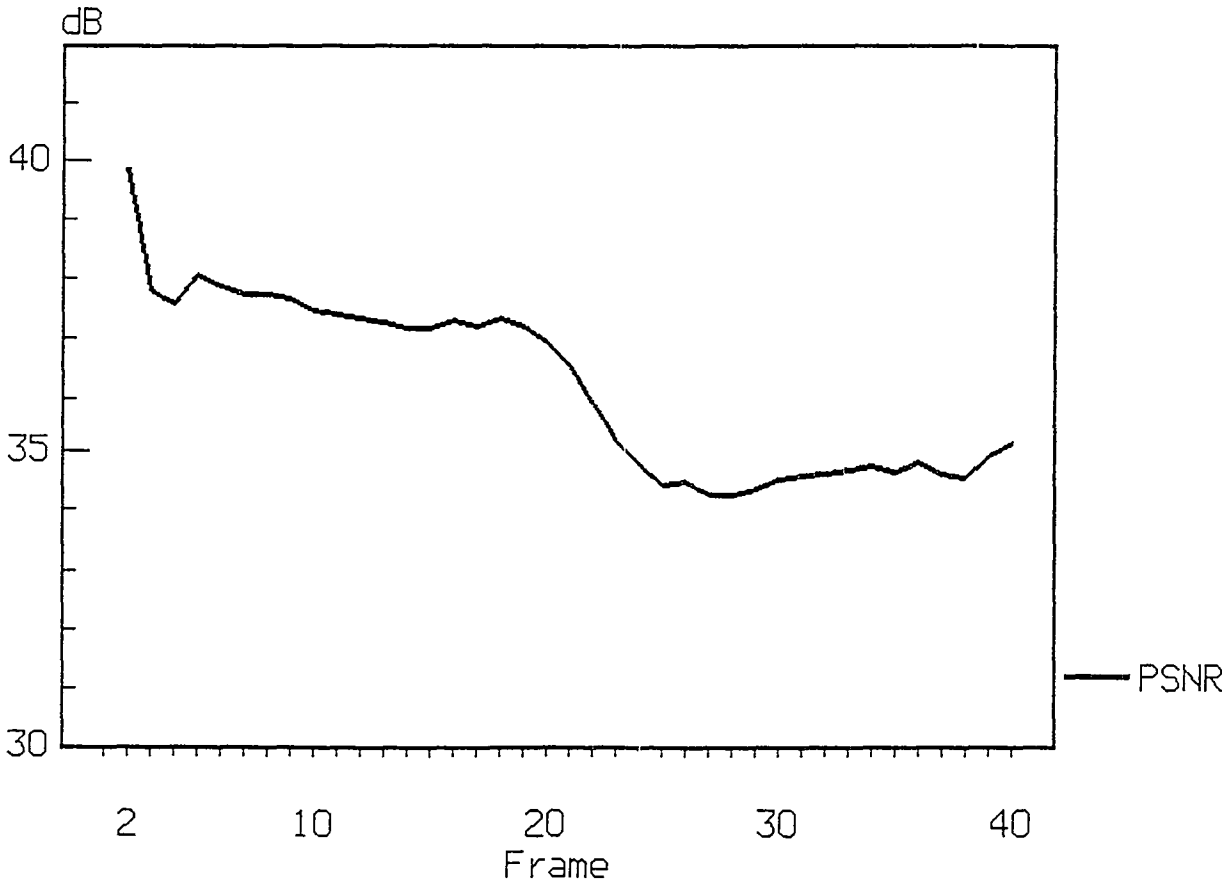


Figure 6.13: PSNR vs frame index of 1-D DCT technique for "CINDY"

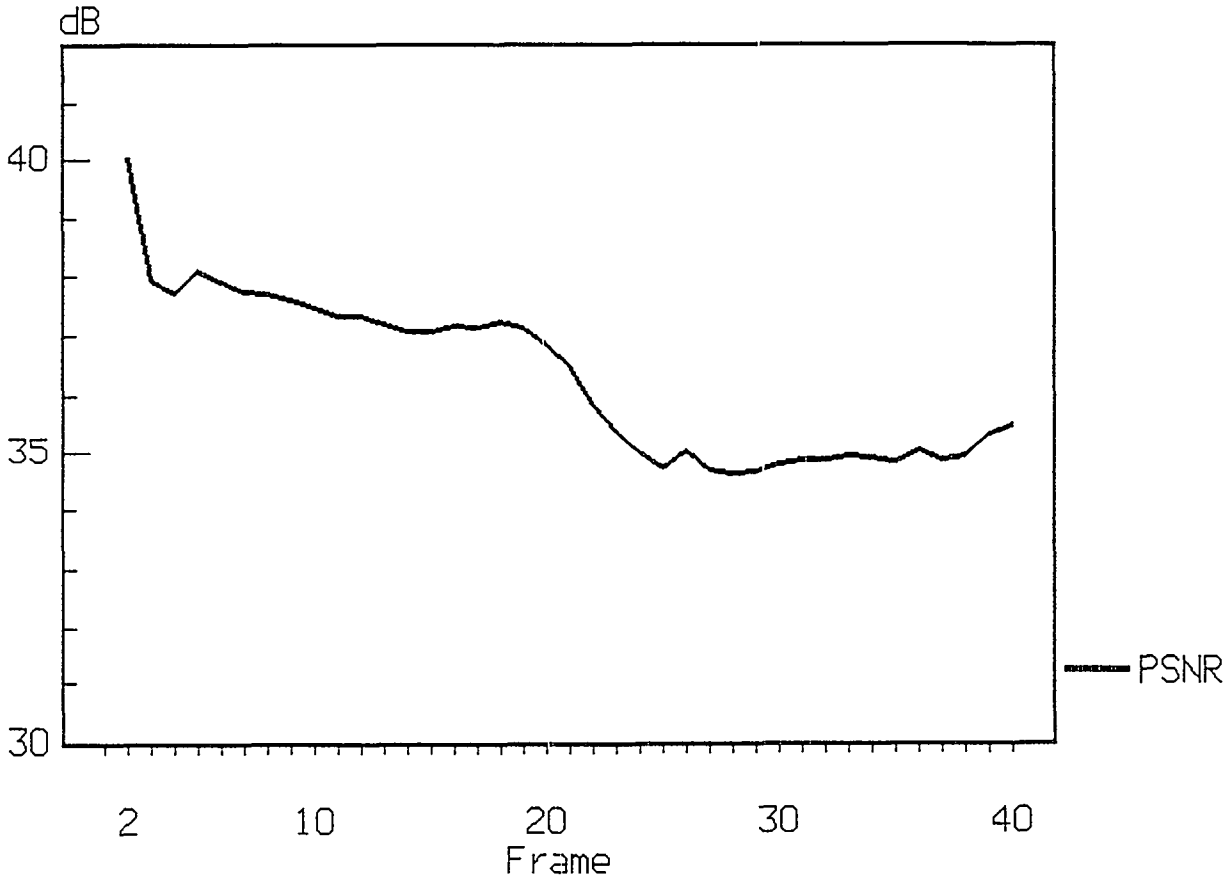


Figure 6.14: PSNR vs frame index of 1-D HDCT technique for "CINDY"

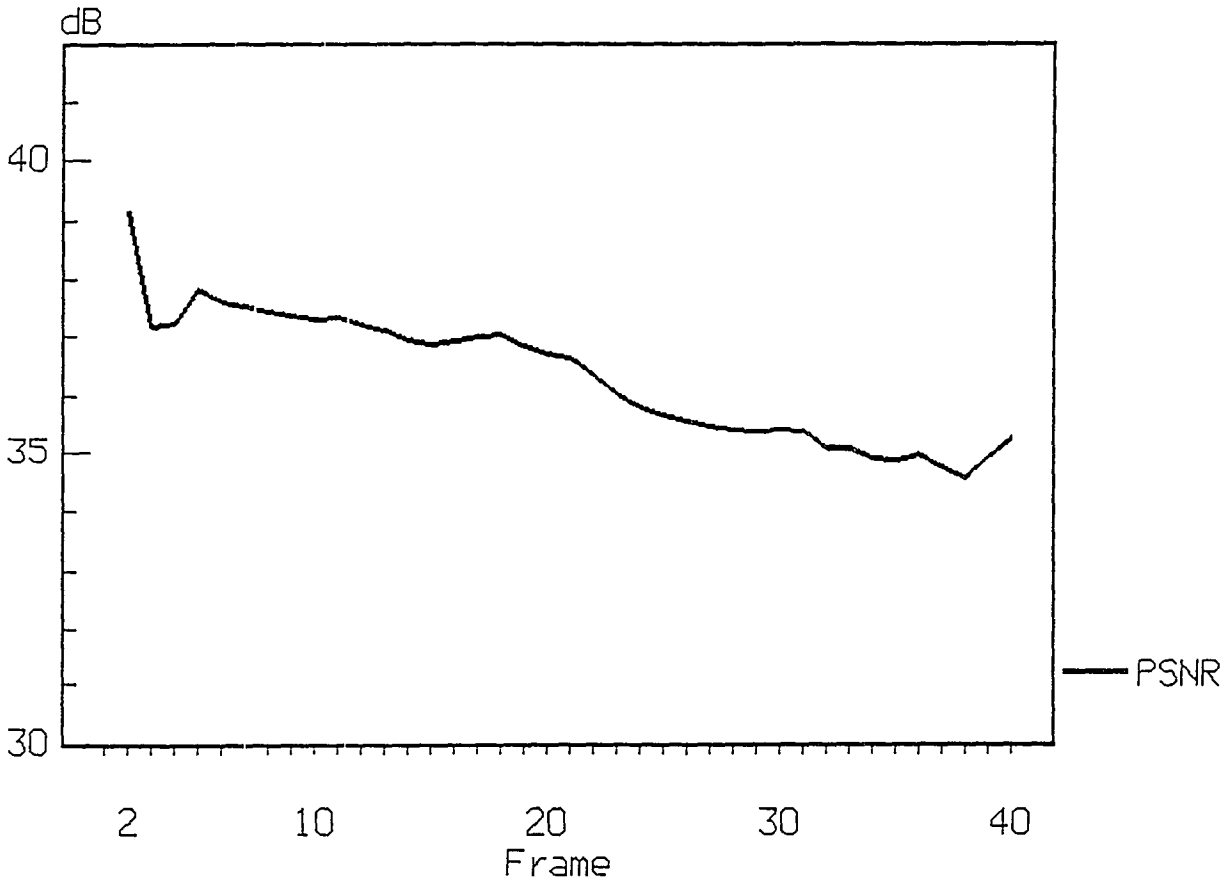


Figure 6.15: PSNR vs frame index of 2-D DCT technique for "CINDY"

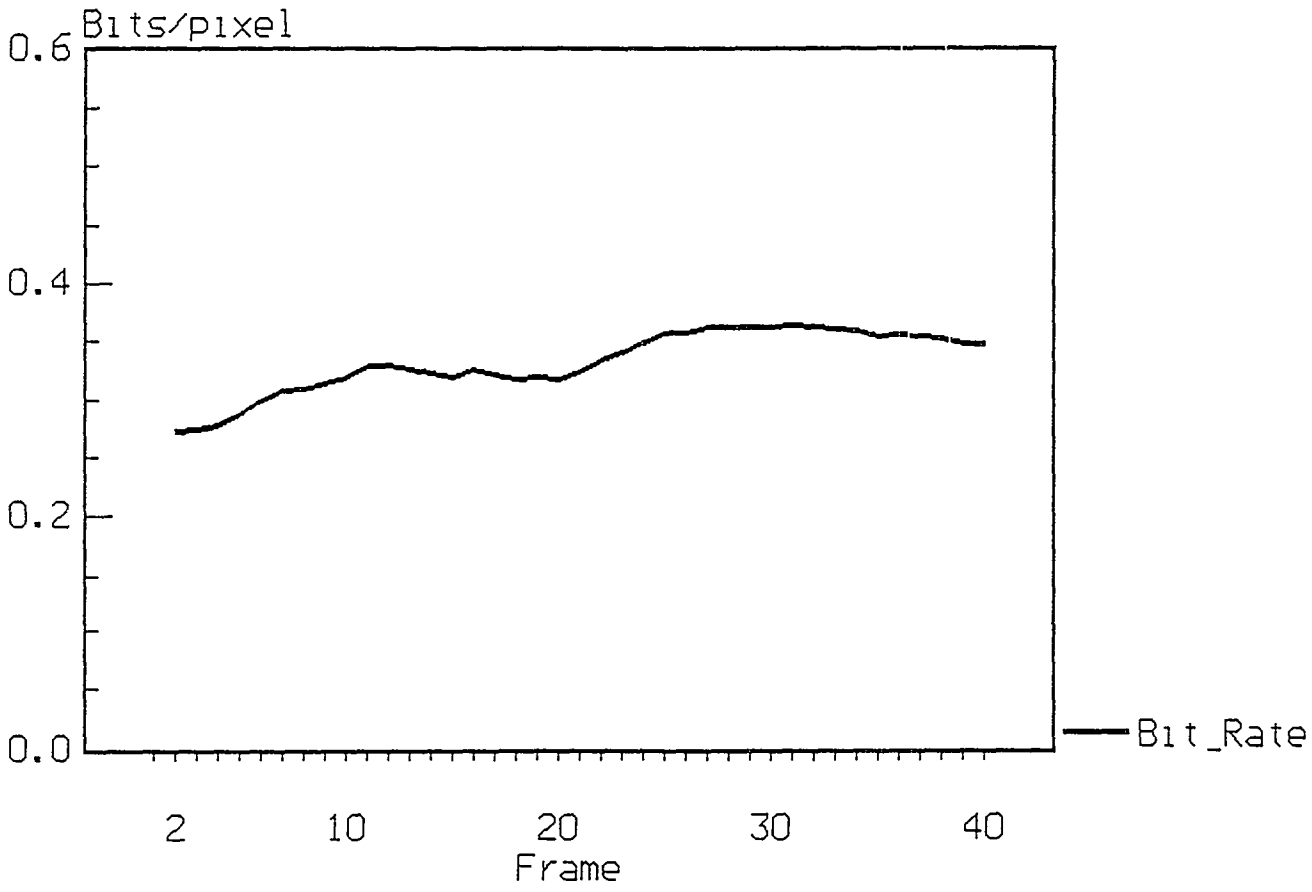


Figure 6.16: Average Bit-Rate for 1-D DCT technique

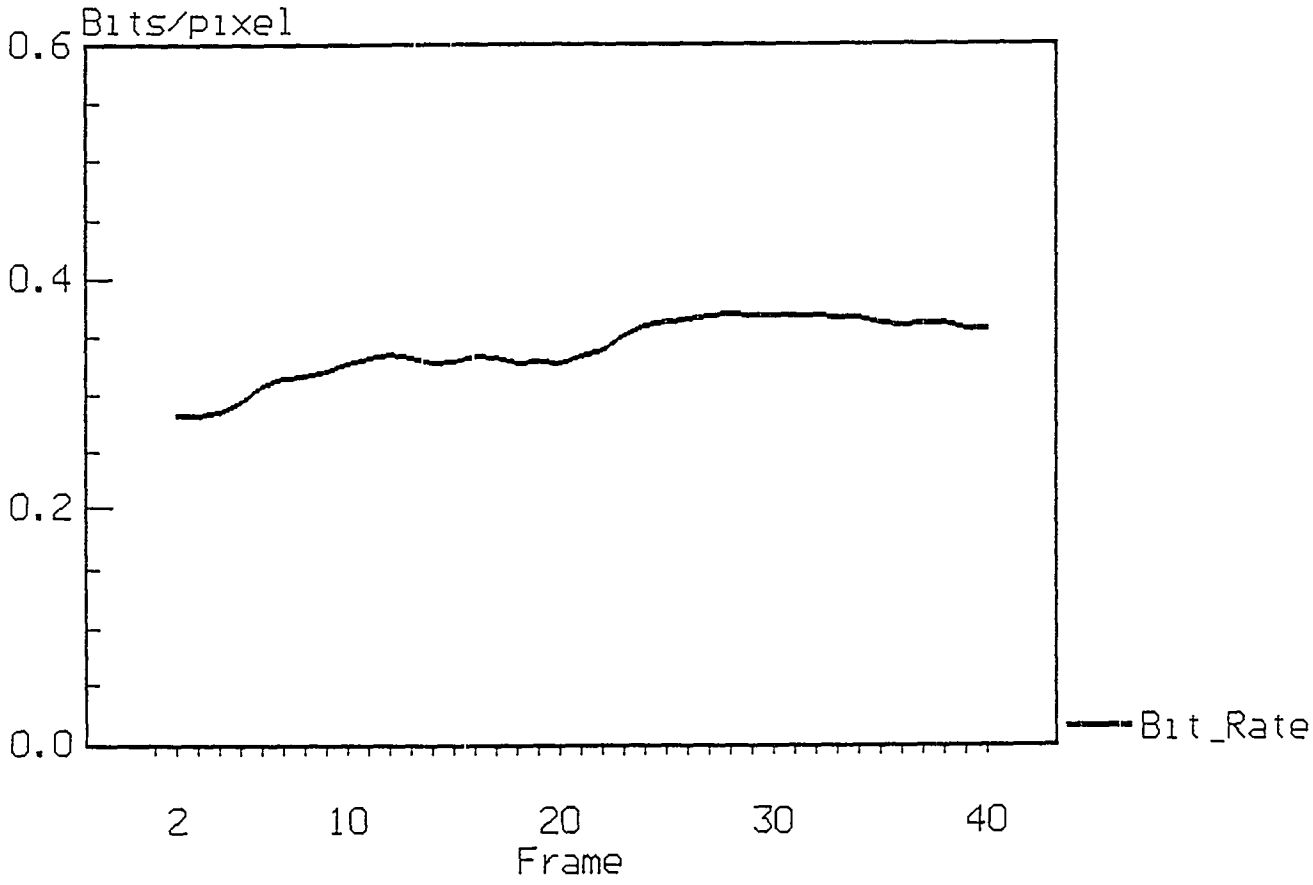


Figure 6.17: Average Bit-Rate for 1-D HDCT technique

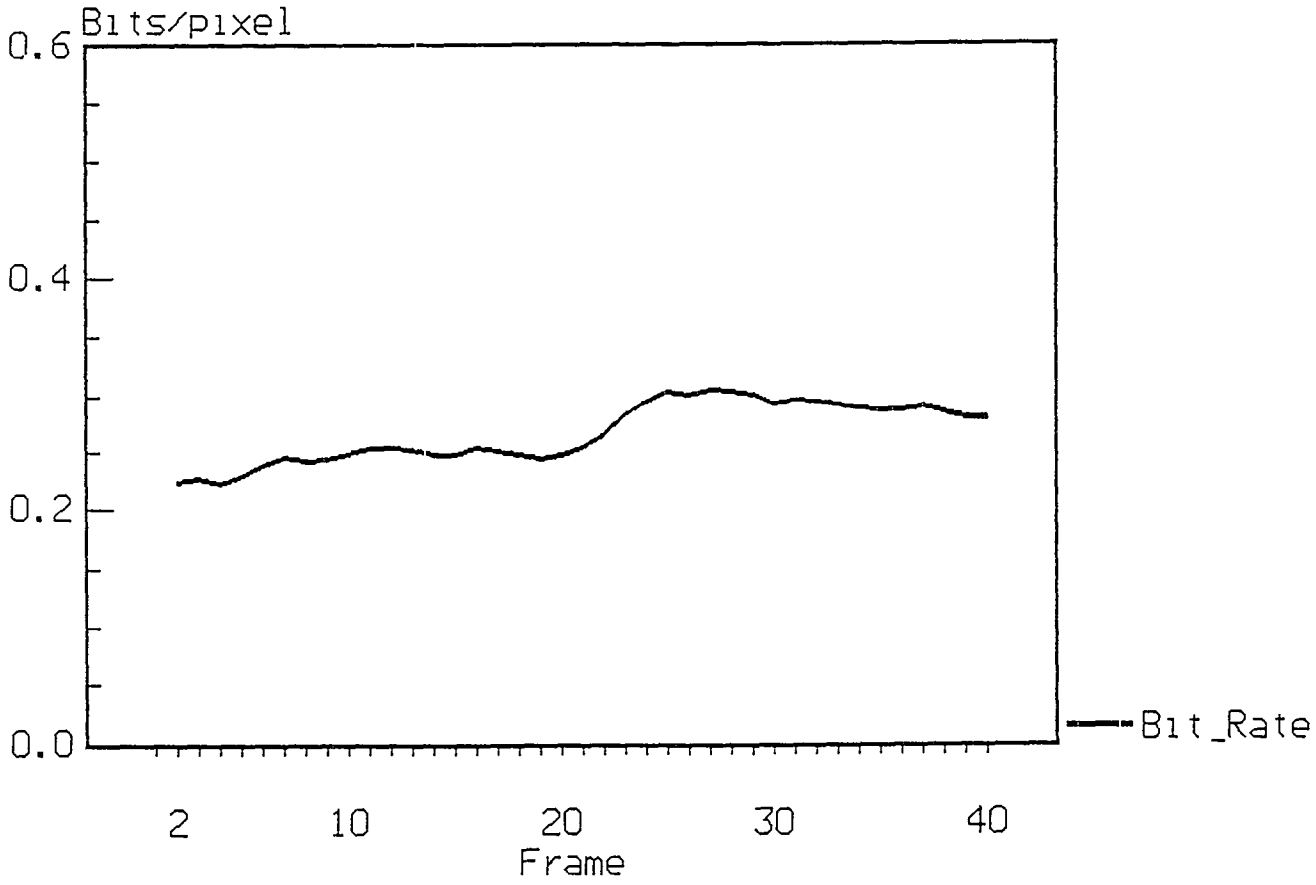


Figure 6.18: Average Bit-Rate for 2-D DCT technique

Technique	Bit-Rate (bits/pixel)	$\overline{\text{PSNR}}$(dB)
2-D	0.375	34.41
8×8	0.406	35.41
DCT	0.438	36.33

Table 6.6: Overall average Bit-Rate and $\overline{\text{PSNR}}$ of still image coding for "CINDY" sequence

6.3 Scene Change Detector

The scene change detector is based on the cross correlation criterion as mentioned in chapter 5. The cross correlation as a function of the frame index for "MIX" sequence on 5 small picture blocks are shown in figures 6.19, 6.20.

6.3.1 Bit-Rate

Whenever a scene change occurred, the encoder switches to the still frame coding mode. The coding method employed on still image is designated in chapter 5.3. The total bit rate of still image coding is fixed and can be defined as:

Bit-Rate : average bits/pixel according to the level of quantizer. It is set to 0.4375 bits/pixel in this study.

The overall average Bit-Rate and $\overline{\text{PSNR}}$ of 2-D 8×8 DCT still image coding on "CINDY" sequence is tabulated in table 6.6, The interframe coding mode in scene change algorithm is employed 1-D DCT case. The result is tabulated in table 6.7. The PSNR and average Bit-Rate as a function of the frame index with scene change algorithm for "MIX" sequence are shown in figures 6.21, 6.22.

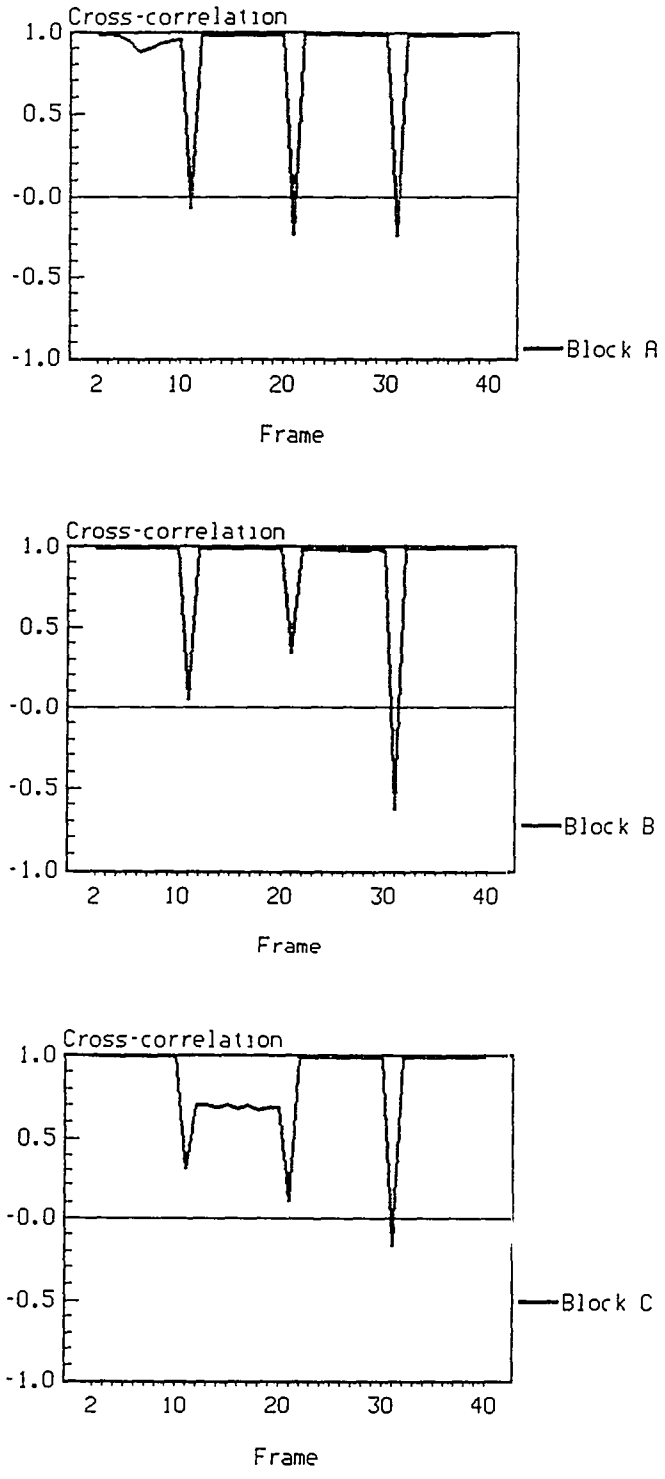


Figure 6.19: Cross correlation of "MIX" sequence on blocks A, B, C

Bit-Rate (bits/pixel)	$\overline{\text{PSNR}}$(dB)
0.3346	36.03

Table 6.7: Overall average Bit-Rate and $\overline{\text{PSNR}}$ with scene change algorithm for "MIX" sequence

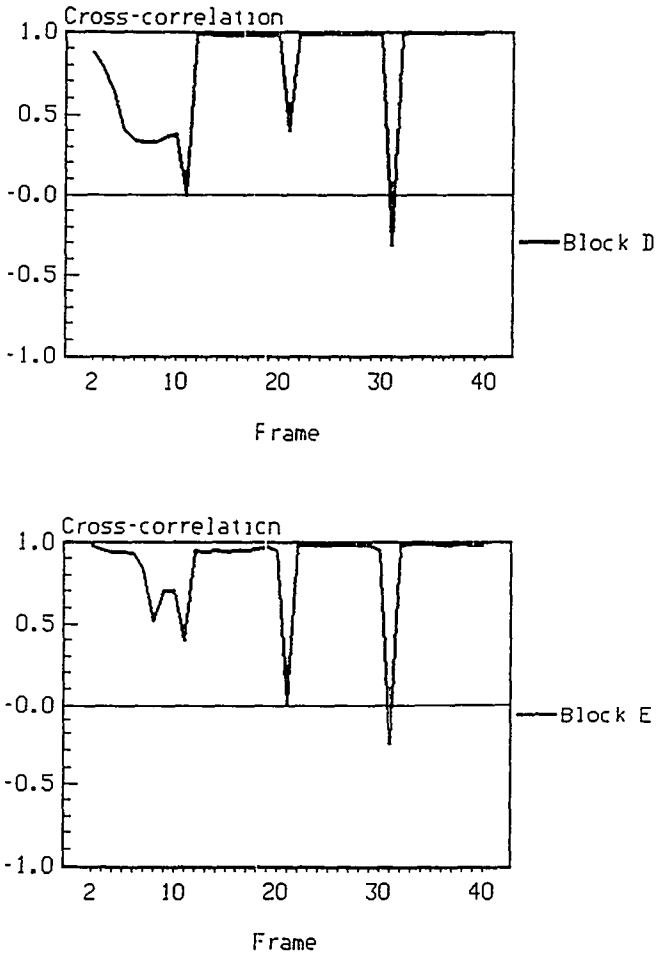


Figure 6.20: Cross correlation of "MIX" sequence on blocks D, E

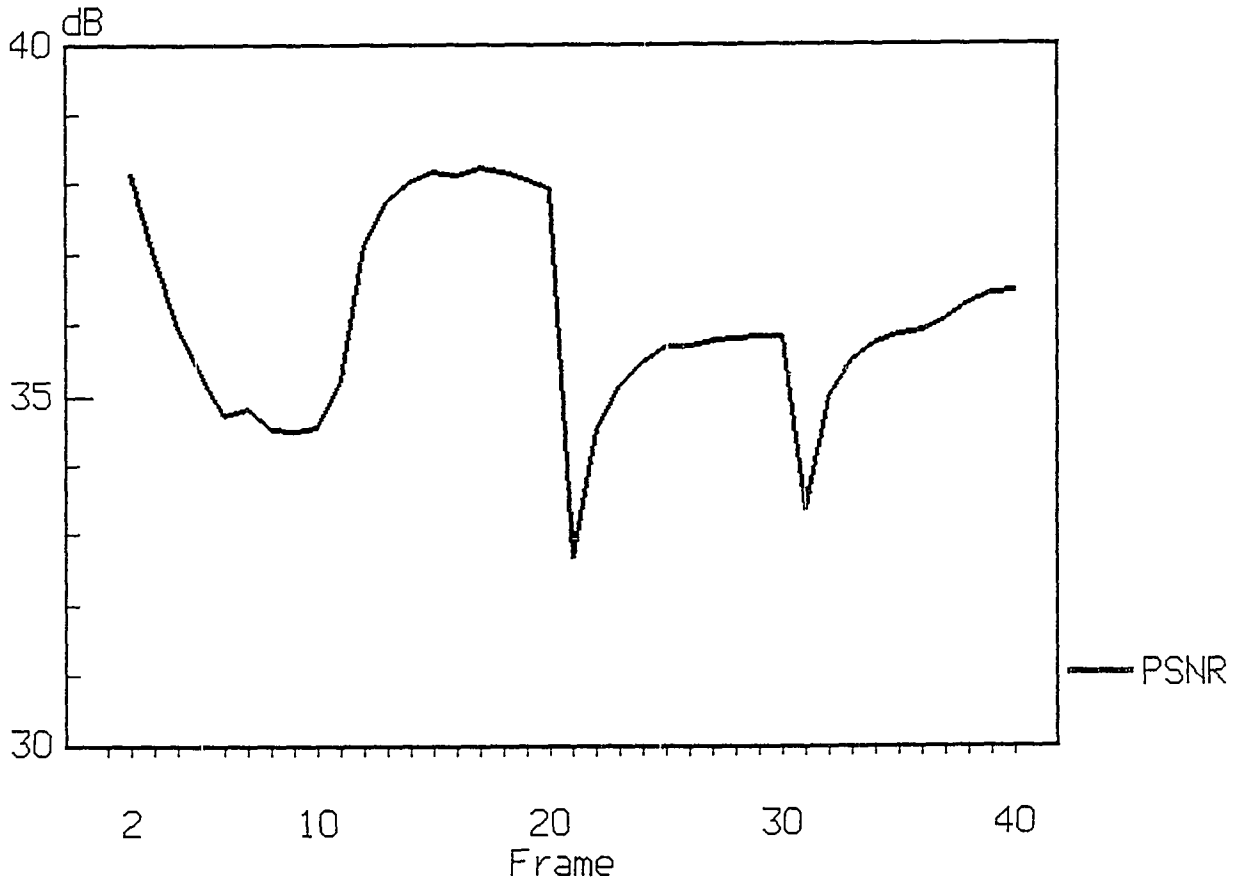


Figure 6.21: PSNR with scene change algorithm for "MIX" sequence

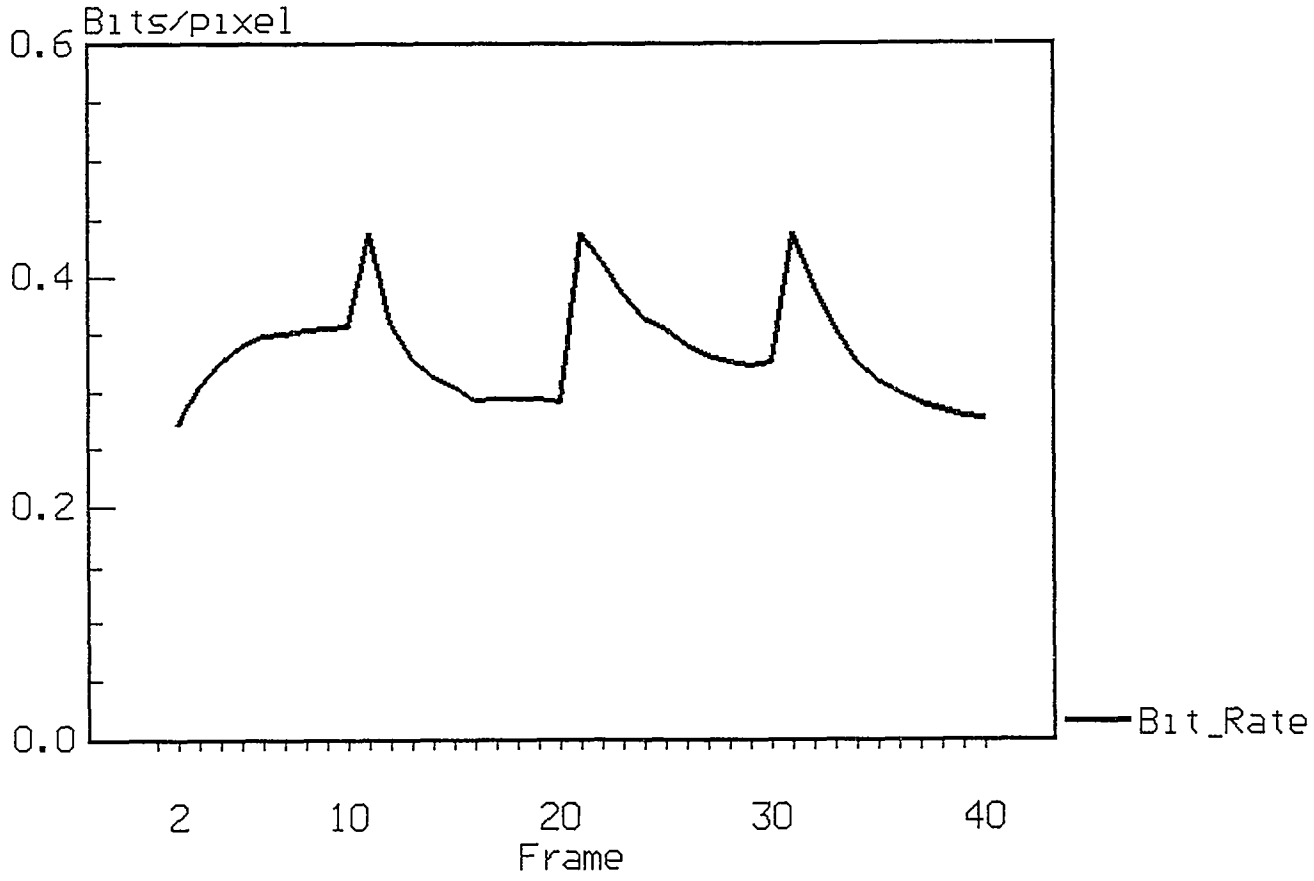


Figure 6.22: Average Bit-Rate with scene change algorithm for "MIX" sequence

Chapter 7

Conclusion

This thesis proposes an efficient transform coding scheme with motion compensation and adaptive vector quantization for the low bit rate video sequence. The bit rate is achieved around 0.3318, 0.3245 and 0.2655 bits/pixel for 1-D DCT, 1-D HDCT and 2-D DCT respectively. It has been shown that the quality of reconstructed image employed by 1-D HDCT is as good as 1-D DCT and 2-D DCT techniques.

The redundancy within adjacent video frames is exploited by motion compensated interframe prediction using the efficient independent orthogonal search technique. The MCFD signal is transform coded with zonal masking. The coefficient bands are encoded for the purpose of transmission with adaptive VQ. It is shown that the vector codebooks on the set of training sequences generated by the MCFD signal is more independent, robust, general than the original video signal.

It was shown that the 1-D HDCT performs practically as well as 1-D DCT and 2-D DCT techniques. The 1-D HDCT is simpler and requires 50% less number of operations for transformations than 2-D DCT. It also provides modularity and dense structure for parallel processing which minimizes processing delay. It may also employ length 8 DCT chips available in the market. This demonstrates that 1-D HDCT with MC and adaptive VQ should be considered as an attractive and efficient scheme for

video coding.

We also presented a cross correlation technique for scene change detection. It has been shown that the detection technique performs well for testing image sequences "MIX", "MIX5", and "MIX60". But the vector codebooks have strong dependency on the training sequences for still image coding.

The bit rate in this study is still considered too high for practical application. The schemes presented in this thesis should be further refined to reduce the bit rate. For example, we have not encoded the side informations. The side informations contain substantial redundancy, which can certainly be reduced by lossless compression techniques. As second example, the current study used fixed zonal masks. The zonal masks could be made adaptive to improve image quality.

Appendix A

Vector codebook generating program

```
c
c       create codebook subroutine
c       dimension (16,1024)
c       input file : group1.tst,group2.tst,group3.tst,group4.tst
c       output file: gp1256.vtr,gp2256.vtr,gp3256.vtr,gp4256.vtr
c       written by : Jung-Hui Chien
c       date   : Sep. 5, 1989
c
        parameter(nclus=1024,pvv=4,phh=4)
        parameter(k1=16954,k2=4085,k3=2319,k4=2242)
        dimension ivec1(16,k1),rmeans(16,nclus)
        dimension ivec2(16,k2)
        dimension ivec3(16,k3)
        dimension ivec4(16,k4)
        real  ivec1,ivec2,ivec3,ivec4
        common  /a/ rmeans
        open(1,file='group1.tst')
        open(2,file='group2.tst')
        open(3,file='group3.tst')
        open(4,file='group4.tst')
        open(5,file='gp11024.vtr')
        open(6,file='gp21024.vtr')
        open(7,file='gp31024.vtr')
        open(8,file='gp41024.vtr')
c
c       read training sequences
c
        read(1,*)((ivec1(i,j),j=1,k1),i=1,16)
        close(1)

        m=k1
        call codebook(ivec1,pvv,phh,m,nclus)
```

```
c
c       writing the codebook into file
c
        write(5,*)((rmeans(i,j),j=1,nclus),i=1,16)
        close(5)

        read(2,*)((ivec2(i,j),j=1,k2),i=1,16)
        close(2)
        m=k2
        call codebook(ivec2,pvv,phh,m,nclus)
        write(6,*)((rmeans(i,j),j=1,nclus),i=1,16)
        close(6)

        read(3,*)((ivec3(i,j),j=1,k3),i=1,16)
        close(3)
        m=k3
        call codebook(ivec3,pvv,phh,m,nclus)
        write(7,*)((rmeans(i,j),j=1,nclus),i=1,16)
        close(7)

        read(4,*)((ivec4(i,j),j=1,k4),i=1,16)
        close(4)
        m=k4
        call codebook(ivec4,pvv,phh,m,nclus)
        write(8,*)((rmeans(i,j),j=1,nclus),i=1,16)
        close(8)
        stop
        end
c
c       codebook subroutine
c
        subroutine codebook(ivec,pvv,phh,m,nclus)
c       this subroutine designs vector code-book
c       pvv---number of rows in a sub-block
c       phh---number of columns in a sub-block
c       m---number of vectors in the training set
c       ivec(l,m)---training set
c       l---vector dimension
```

```
c       rmeans(l,n)---designed vector code-book
c       n---number of vectors in the vector code-book
c       nlus=n---vector code-book size
c       laba(m)---label of code word
        parameter(nx=16,ny=1024,nbig=16954)
        dimension ivec(nx,nbig)
        dimension laba(nbig)
        dimension pixel(ny),vctr(ny)
        real rmse(nx),rmeans(nx,ny),rmd(ny)
        real reans(nx,ny),rvd(ny),rssm(ny)
        real ivec
        integer pixel,clnum
        common /a/ rmeans
c
c       initial values
c
        nb=pvv*phh
        do 20 l=1,nb
        rmeans(l,1)=float(ivec(l,1))
 20     continue
        ik=2
        i=2
        nr=0
 25     rsh=12000.0-(nr*25.0)+0.5
c
c       rsh---threshold value to select initial seeds
c
        do 30 k=1,ik-1
        rvd(k)=0
        do 40 l=1,nb
        rvd(k)=rvd(k)+(rmeans(l,k)-float(ivec(l,i)))**2
 40     continue
        if(rvd(k).le.rsh) goto 55
 30     continue
        do 50 l=1,nb
        rmeans(l,ik)=float(ivec(l,i))
 50     continue
        ik=ik+1
```

```
 55     i=i+1
        if(i.ge.m) goto 75
        if(ik.lt.nclus+1) goto 25
        goto 77
 75     nr=nr+1
        ik=2
        i=2
        goto 25
 77     do 60 k=1,nclus
c
c       using initial seeds to cluster the training set
c
        do 65 l=1,nb
        reans(l,k)=0
 65     continue
        pixel(k)=0
 60     continue
        do 510 i=1,m
        do 520 ii=1,nb
        vctr(ii)=ivec(ii,i)
 520    continue
        rdmin=1.e20
        clnum=0
        do 530 k=1,nclus
        r=0.0
        do 540 l=1,nb
        rtmp=rmeans(l,k)-float(vctr(l))
        r=r+rtmp*rtmp
 540    continue
        if(r.ge.rdmin) goto 530
        rdmin=r
        clnum=k
 530    continue
        pixel(clnum)=pixel(clnum)+1
        do 550 l=1,nb
        reans(l,clnum)=reans(l,clnum)+float(vctr(l))
 550    continue
        laba(i)=clnum
```

```
 510   continue
c
c      calculate real means after all records processed
c
       do 560 k=1,nclus
       do 570 l=1,nb
       if(pixel(k).ne.0) goto 575
       goto 560
 575   rmeans(l,k)=reans(l,k)/float(pixel(k))
 570   continue
 560   continue
       do 600 i=1,nb
       rmse(i)=0
 600   continue
       rgg1=1.0e20
       rgg2=0.0
       do 610 i=1,m
       do 620 ii=1,nb
       vctr(ii)=ivec(ii,i)
 620   continue
       do 640 ii=1,nb
       rmse(ii)=rmse(ii)+(rmeans(ii,laba(i))-float(vctr(ii)))**2
 640   continue
 610   continue
       do 650 ii=1,nb
       rgg2=rgg2+rmse(ii)
 650   continue
       rgg=(abs(rgg1-rgg2))/(rgg2+0.0001)
       if(rgg.le.0.00001) goto 999
       iter=0
 97    rgg1=rgg2
       rgg2=0
       iter=iter+1
       do 660 k=1,nclus
       do 665 l=1,nb
       reans(l,k)=0
 665   continue
       rssm(k)=0
```

```
      rmd(k)=0
      pixel(k)=0
660   continue
      do 700 i=1,m
      do 710 ii=1,nb
      vctr(ii)=ivec(ii,i)
710   continue
      rdmin=1.e20
      clnum=0
      do 720 k=1,nclus
      r=0.0
      do 730 l=1,nb
      rtmp=rmeans(l,k)-float(vctr(l))
      r=r+rtmp*rtmp
730   continue
      if(r.ge.rdmin) goto 720
      rdmin=r
      clnum=k
720   continue
      pixel(clnum)=pixel(clnum)+1
      do 740 l=1,nb
      reans(l,clnum)=reans(l,clnum)+float(vctr(l))
740   continue
      laba(i)=clnum
700   continue
      do 750 k=1,nclus
      do 760 l=1,nb
      if(pixel(k).ne.0) goto 759
      goto 750
759   rmeans(l,k)=reans(l,k)/float(pixel(k))
760   continue
750   continue
      do 800 i=1,nb
      rmse(i)=0
800   continue
      do 810 i=1,m
      do 820 ii=1,nb
      vctr(ii)=ivec(ii,i)
```

```
820   continue
      do 830 ii=1,nb
      rmse(ii)=rmse(ii)+(rmeans(ii,laba(i))-float(vctr(ii)))**2
830   continue
810   continue
      do 840 ii=1,nb
      rgg2=rgg2+rmse(ii)
840   continue
      rgg=(abs(rgg1-rgg2))/(rgg2+0.0001)
      if(rgg.gt.0.0001) goto 97
999   return
      end
```

Appendix B

Main program

```
c
c       adaptive transform coding of video with motion
c       compensation and vector quantization
c       written by: Jung-Hui Chien
c       date  : Sep. 5, 1989
c
        parameter(nx=400,ny=512)
        dimension pdf16(4),pgp16(4)
        real frame1(nx,ny),frame2(nx,ny),pics(416,528),
     + recon2(nx,ny),frm1msk(8,8),frm2msk(8,8),err1(nx,ny),
     + errtemp(nx,ny),pframe(nx,ny)
        integer ifrm1(nx,ny),ifrm2(nx,ny),
     + motionvec(20,20)
        character*1 pim(nx*ny)
        character*1 pim1(nx*ny)

        common /A/ searg(24,24),mask(8,8)
        common /E/ motionvec

        open(10,file='thesis.dat')
        open(12,file='thesis.pdf')

c       print *,'Enter number of fields to be read'
c       read *,mfld
        mfld =40
c       print *, 'Starting field number'
c       read *,ifld
        ifld = 1
c       call read12(ifld,pim)
        call read13(ifld,pim)

        do 5 i = 1,20
```

```
      do 5 j = 1,20
      motionvec(i,j) = 0
5     continue

      do 10 i=1,nx
      k=(i-1)*ny
      l=k+1
      ll=l+ny-1
      mm=0
      do 10 m=l,ll
      mm=mm+1
      ifrm1(i,mm)=ichar(pim(m))
      if(ifrm1(i,mm).lt.0) ifrm1(i,mm)=256+ifrm1(i,mm)
      frame1(i,mm)=float(ifrm1(i,mm))
10    continue

      do 20 i=1,nx
      do 20 j=1,ny
      pframe(i,j)=frame1(i,j)
20    continue

      imcfd=0
      mavg=0
      asnr=0.
      asnr1=0.

      do 30 i=1,4
      pdf16(i)=0.
30    continue

6000  ifld = ifld+1
      print *,'process : ',ifld-1
c     call read12(ifld,pim1)
      call read13(ifld,pim1)
      do 40 i=1,4
      pgp16(i)=0.
40    continue
```

```
      do 50 i=1,nx
      k=(i-1)*ny
      l=k+1
      ll=l+ny-1
      mm=0
      do 50 m=l,ll
      mm=mm+1
      ifrm2(i,mm)=ichar(pim1(m))
      if(ifrm2(i,mm).lt.0) ifrm2(i,mm)=256+ifrm2(i,mm)
      frame2(i,mm)=float(ifrm2(i,mm))
 50   continue

      print *,'Frame',ifld-1
      call autocor(frame1,nx,ny)
      print *,'Frame',ifld
      call autocor(frame2,nx,ny)
c
c     Scene change detector
c
      call scene(pframe,frame2,isc)

      if(isc.eq.1)then
      do 60 i=1,nx
      do 60 j=1,ny
      pframe(i,j)=frame2(i,j)
 60   continue
c
c     still image coding 2-D 8*8 DCT algorithm
c
      call still(frame2,snr,snr1)
      do 70 i=1,nx
      do 70 j=1,ny
      frame1(i,j)=frame2(i,j)
 70   continue
      asnr=asnr+snr
      asnr1=asnr1+snr1
      else
c
```

```
c       generating MCFD signal algorithm
c
        imcfd=imcfd+1
c       print *,'Enter displacement'
c       read *,ip
        ip=6

c       print *,'Enter the mask block size'
c       read *,ibs
        ibs=8

c       Print *,'Enter 1 for Brute-force search'
c       Print *,'      2 for Orthogonal independent search'
c       read *,imthd
        imthd=2

c       print *,'Enter 1 if motion-detection is required'
c       read *,imdetect
        imdetect=1
c
c       extend border
c
        do 80 i=1,nx+2*ip
        do 80 j=1,ny+2*ip
        pics(i,j)=0.0
 80     continue

        do 90 i=1,nx
        do 90 j=1,ny
        pics(i+ip,j+ip)=frame1(i,j)
 90     continue

        do 100 i=1,ip
        do 100 j=1,ny
        pics(i,j)=frame1(i,j)
        pics(i+nx+ip,j)=frame1(i+nx-ip,j)
 100    continue
```

```
      do 110 i=1,nx
      do 110 j=1,ip
      pics(i,j)=frame1(i,j)
      pics(i,j+ny+ip)=frame1(i,j+ny-ip)
 110  continue

      do 120 i4=1,nx
      do 120 j4=1,ny
      recon2(i4,j4)=0.0
 120  continue

      mcount=0
      do 200 i=1,nx/ibs
      do 200 j=1,ny/ibs
      iact=(i-1)*ibs+1
      jact=(j-1)*ibs+1
      if (imdetect .eq. 1) then
      do 130 k=1,ibs
      do 130 l=1,ibs
      frm1msk(k,l)=frame1(iact-1+k,jact-1+l)
      frm2msk(k,l)=frame2(iact-1+k,jact-1+l)
 130  continue
c
c     motion detection
c

      call motiondetect(frm1msk,frm2msk,ibs,indx)
c
c     define search region
c

      if (indx .eq. 1) then
      do 135 i1=1,ibs+ip*2
      do 135 j1=1,ibs+ip*2
      searg(i1,j1)=pics(i1+iact-1+ip-ip,j1+jact-1+ip-ip)
 135  continue

      do 140 i2=1,ibs
```

```
      do 140 j2=1,ibs
      mask(i2,j2)=frame2(iact-1+i2,jact-1+j2)
 140  continue
c
c     block matching algorithm
c
      call matct(ip,ibs,imthd,n,nn)
c
c     motion blocks counter
c
      nnnd=(n-1)*13+nn
      if(nnnd.ne.85)then
      mcount=mcount+1
      endif

      do 150 i3=1,ibs
      do 150 j3=1,ibs
      recon2(iact-1+i3,jact-1+j3)=pics(iact+ip-1+(n-ip)-1+i3,
     + jact+ip-1+(nn-ip)-1+j3)
 150  continue

      else
c
c     no motion
c
c     print *,85
      do 160 k1=1,ibs
      do 160 l1=1,ibs
      recon2(iact-1+k1,jact-1+l1)=frame1(iact-1+k1,jact-1+l1)
 160  continue
      endif

      else

      do 170 i1=1,ibs+ip*2
      do 170 j1=1,ibs+ip*2
      searg(i1,j1)=pics(i1+iact-1+ip-ip,j1+jact-1+ip-ip)
 170  continue
```

```
      do 180 i2=1,ibs
      do 180 j2=1,ibs
      mask(i2,j2)=frame2(iact-1+i2,jact-1+j2)
180   continue

      call matct(ip,ibs,imthd,n,nn)

      do 190 i3=1,ibs
      do 190 j3=1,ibs
      recon2(iact-1+i3,jact-1+j3)=pics(iact+ip-1+(n-ip)-1+i3,
     + jact+ip-1+(nn-ip)-1+j3)
190   continue
      endif
200   continue

      mavg=mavg+mcount
      err=0.0
      do 210 i=1,nx
      do 210 j=1,ny
      err=(err+abs(frame2(i,j)-recon2(i,j)))
      err1(i,j)=frame2(i,j)-recon2(i,j)
210   continue

      print *,'MAE between frame & MC frame ',err

      if (imdetect .eq. 1) then
      print *,'Number of blocks moved in this frame',ifld,'=',mcount
      endif

      print *,'Error Signal between frame & MC frame before VQ'
      call autocor(err1,nx,ny)
      print *,'Motion compensated singal'
      call autocor(recon2,nx,ny)

      do 220 i = 1,nx
      do 220 j = 1,ny
      errtemp(i,j)=err1(i,j)
```

```
      pframe(i,j)=frame2(i,j)
 220  continue
c
c     encoding MCFD signal
c
      call hdct1(err1,pgp16)

      do 230 i=1,4
      pdf16(i)=pdf16(i)+pgp16(i)
 230  continue
      print *,'Error Signal after VQ'
      call autocor(err1,nx,ny)
      vecmean = 0.0
      vecvar = 0.0
      do 240 i = 1,nx
      do 240 j = 1,ny
      vecmean = vecmean + (errtemp(i,j) - err1(i,j))
      vecvar = vecvar + (errtemp(i,j) - err1(i,j))**2
 240  continue
      vecmean = vecmean/(nx*ny)
      vecvar = vecvar/(nx*ny)
      vecvar = vecvar - vecmean**2
      print *, 'Variance of error signal before VQ - after VQ=',
     + vecvar

      omean=0.
      rmean=0.
      ovar=0.
      rvar=0.
      t=0.
      to=0.
      too=0.
      errmean = 0.0
      do 250 i=1,nx
      do 250 j=1,ny
      frame1(i,j)=recon2(i,j)+err1(i,j)
      if(frame1(i,j).le.0.)frame1(i,j)=0.
      if(frame1(i,j).ge.255.)frame1(i,j)=255.
```

```
      omean=omean+frame2(i,j)
      rmean=rmean+frame1(i,j)
      ovar=ovar+frame2(i,j)*frame2(i,j)
      rvar=rvar+frame1(i,j)*frame1(i,j)
      too = too + abs(frame2(i,j)-frame1(i,j))
      to = to + (frame2(i,j)-frame1(i,j))**2
      t=t+frame2(i,j)**2
      errmean = errmean + (frame2(i,j)-frame1(i,j))
250   continue

      errmean = errmean/(nx*ny)
      omean=omean/(nx*ny)
      rmean=rmean/(nx*ny)
      ovar=ovar/(nx*ny)-omean*omean
      rvar=rvar/(nx*ny)-rmean*rmean
      fmse=to/(nx*ny)
      fmae=too/(nx*ny)
      errvar=fmse-errmean**2
      ratio=(255.*255.)/fmse
      t=t/(nx*ny)
      snr=10*alog10(ratio)
      snr1=10*alog10(t/fmse)
      write(10,*)ifld,omean,rmean,ovar,rvar,to,fmse,too,
     + fmae,errvar,ratio,snr,snr1
      write(12,*)ifld,(pgp16(i),i=1,4)

      asnr=asnr+snr
      asnr1=asnr1+snr1
      endif
      if (ifld .lt. mfld) goto 6000

      avg=float(mavg)/imcfd
      asnr=asnr/(ifld-1)
      asnr1=asnr1/(ifld-1)
      do 260 i=1,4
      pdf16(i)=pdf16(i)/imcfd
260   continue
      write(10,*)'average of SNR (255*255) = ',asnr
```

```
      write(10,*)'average of SNR (original pixel **2)=',asnr1
      write(12,*)' 4 varinace groups PDF 1,2,3,4 =',(pdf16(i),i=1,4)
      print*,'Average number of blocks moved = ',avg
      stop
      end

      subroutine matct(ip,ibs,imthd,n,nn)
      common /A/ searg(24,24),mask(8,8)
      common /E/ motionvec
      real test(13,13)
      integer motionvec(20,20)

      do 50 i=1,2*ip+1
      do 50 j=1,2*ip+1
      test(i,j)=0.0
      do 50 ii=1,ibs
      do 50 jj=1,ibs
      test(i,j)=abs(mask(ii,jj)-searg(i+ii-1,j+jj-1))+test(i,j)
 50   continue
c
c     brute force search
c
      if (imthd .eq. 1 ) then
      tmin=1.0e20
      do 100 i=1,2*ip+1
      do 100 k=1,2*ip+1
      if(test(i,k) .lt. tmin) then
      tmin=test(i,k)
      n=i
      nn=k
      endif
 100  continue
      Num=(n-1)*(ip*2+1)+nn
c     print*,Num
      else
c
c     orthogonal independent search
c
```

```
      call ortho(test,ip,ibs,icent,jcent)
      n=icent
      nn=jcent
      motionvec(n,nn) = motionvec(n,nn) + 1
      Num=(n-1)*(ip*2+1)+nn
c     print*,Num
      endif
      return
      end

      subroutine ortho(test,ip,ibs,icent,jcent)
      eal test(ip*2+1,ip*2+1)
      icent=ip+1
      jcent=ip+1
      l=ip/2.+.5
      istep=0
10    if ((test(icent,jcent) .lt. test(icent,jcent-l)) .and.
     + (test(icent,jcent) .lt. test(icent,jcent+l)))  then
      icent=icent
      jcent=jcent
      else if ((test(icent,jcent-l) .lt. test(icent,jcent)) .and.
     + (test(icent,jcent-l) .lt. test(icent,jcent+l))) then
      icent=icent
      jcent=jcent-l
      else if ((test(icent,jcent+l) .lt. test(icent,jcent)) .and.
     + (test(icent,jcent+l) .lt. test(icent,jcent-l))) then
      icent=icent
      jcent=jcent+l
      endif

      istep=istep+1

      if ((test(icent,jcent) .lt. test(icent-l,jcent)) .and.
     + (test(icent,jcent) .lt. test(icent+l,jcent)))  then
      icent=icent
      jcent=jcent
      else if ((test(icent-l,jcent) .lt. test(icent,jcent)) .and.
     + (test(icent-l,jcent) .lt. test(icent+l,jcent))) then
```

```
      icent=icent-l
      jcent=jcent
      else if ((test(icent+l,jcent) .lt. test(icent,jcent)) .and.
     + (test(icent+l,jcent) .lt. test(icent-l,jcent))) then
      icent=icent+l
      jcent=jcent
      endif

      istep=istep+1

      if (l .ne. 1) then
      l=(l/2.0+.5)
      go to 10
      endif
      return
      end

      subroutine motiondetect(frm1msk,frm2msk,ibs,indx)
      real frm1msk(ibs,ibs),frm2msk(ibs,ibs)
      kount=0
      do 10 i=1,ibs
      do 10 j=1,ibs
      thrsh=abs(frm1msk(i,j)-frm2msk(i,j))
      if (thrsh .gt. 3.0) kount=kount+1
10    continue
      if (kount .gt. 10) then
      indx=1
      else
      indx=0
      endif
      return
      end

      subroutine autocor(frame,nx,ny)
      real frame(nx,ny)
      real autoc(400),autoc1(512)

      rautoc=0.0
```

```
      do 11 k=1,nx
      autoc(k)=0.0
      do 12 l=1,ny-1
      autoc(k)=autoc(k)+frame(k,l)*frame(k,l+1)
12    continue
      rautoc=rautoc+autoc(k)/ny
11    continue

      rautoc1=0.0
      do 13 l=1,ny
      autoc1(l)=0.0
      do 14 k=1,nx-1
      autoc1(l)=autoc1(l)+frame(k,l)*frame(k+1,l)
14    continue
      rautoc1=rautoc1+autoc1(l)/nx
13    continue

      rac=0.0
      rmean=0.0
      do 23 l=1,ny
      do 24 k=1,nx
      rac=rac+frame(k,l)*frame(k,l)
      rmean=rmean+frame(k,l)
24    continue
23    continue
      rmean=rmean/(nx*ny)
      var=rac/(nx*ny)-rmean*rmean
      rautoc=rautoc/nx-rmean*rmean
      rautoc1=rautoc1/ny-rmean*rmean
      print *,'     Mean         Variance'
      print *,rmean,var
      rh=rautoc/var
      rv=rautoc1/var
      print *,'  Autocor-H      Autocor-V'
      print *,rh,rv
      return
      end
```

```
      subroutine read13(ifrmno,pim1)
      dimension pic(400,512),ilady(400,512),picl1(400,512)
c
c     gets any frame out of VIDEO sequence (Cindy)
c
      character*1 pim(8192000),pim1(204800)
      integer ilady,picl1,l4,l5
      real pic
      open(1,file='/images/cindy',access='direct',form='unformatted',
     + recl=8192000)
      read(1,rec=1) (pim(j),j=1,8192000)
      l4=ifrmno
      l5=204800*(l4-1)
      do 20 i=1,200
      k=(i-1)*512
      l=k+1
      ll=l+511
      mm=0
      do 21 m=l,ll
      mm=mm+1
      jj=(i-1)*2+1
      ilady(jj,mm)=ichar(pim(m+l5))
      if(ilady(jj,mm).lt.0) ilady(jj,mm)=256+ilady(jj,mm)
      pic(jj,mm)=float(ilady(jj,mm))
 21   continue
 20   continue
      do 30 i=1,200
      k=(i-1)*512
      l=k+1
      ll=l+511
      mm=0
      do 31 m=l,ll
      mmm=m+102400
      mm=mm+1
      jj=(i-1)*2+2
      ilady(jj,mm)=ichar(pim(mmm+l5))
      if(ilady(jj,mm).lt.0) ilady(jj,mm)=256+ilady(jj,mm)
      pic(jj,mm)=float(ilady(jj,mm))
```

```
 31     continue
 30     continue
        do 70 i=1,400
        do 71 k=1,512
        picl1(i,k)=pic(i,k)
 71     continue
 70     continue
        do 80 i=1,400
        ii=(i-1)*512
        l=ii+0
        do 81 k=1,512
        if(picl1(i,k).gt.128) picl1(i,k)=picl1(i,k)-256
        pim1(l+k)=char(picl1(i,k))
 81     continue
 80     continue
        return
        end

        subroutine read12(ifrmno,pim1)
        dimension pic(400,512),ilady(400,512),picl1(400,512)
c
c       gets any frame out of VIDEO sequence (mix,mix5,mix60,keilharbor)
c
        character*1 pim(8192000)
        character*1 pim1(204800)
        integer ilady,picl1,l4,l5
        real pic
        open(1,file='/images/mix',access='direct',form='unformatted',
     + recl=8192000)
        read(1,rec=1) (pim(j),j=1,8192000)
        l4=ifrmno
        l5=204800*(l4-1)
        do 20 i=1,400
        k=(i-1)*512
        l=k+1
        ll=l+511
        mm=0
        do 21 m=l,ll
```

```
      mm=mm+1
      jj=i
      ilady(jj,mm)=ichar(pim(m+15))
      if(ilady(jj,mm).lt.0) ilady(jj,mm)=256+ilady(jj,mm)
      pic(jj,mm)=float(ilady(jj,mm))
 21   continue
 20   continue
      do 70 i=1,400
      do 71 k=1,512
      picl1(i,k)=pic(i,k)
 71   continue
 70   continue
      do 80 i=1,400
      ii=(i-1)*512
      l=ii+0
      do 81 k=1,512
      if(picl1(i,k).gt.128) picl1(i,k)=picl1(i,k)-256
      pim1(l+k)=char(picl1(i,k))
 81   continue
 80   continue
      return
      end

      subroutine vcb(nclus,iml,cvc)
      dimension iml(16,1),cvc(16,nclus),temp(16,1)
      real  min,iml,cvc,temp
c
c     choose optimum codeword
c
      min=10.**20
      do 345 jj=1,nclus
      sum=0.
      do 365 i=1,16
      sum=sum+(iml(i,1)-cvc(i,jj))**2
 365  continue
      if(sum.ge.min) goto 345
      min=sum
      do 385 i=1,16
```

```
      temp(i,1)=cvc(i,jj)
 385  continue
 345  continue
      do 425 i=1,16
      iml(i,1)=temp(i,1)
 425  continue
      return
      end

      subroutine scene(frame1,frame2,isc)
      parameter(nbk=5,nx=400,ny=512)
      dimension rows(nbk),rowe(nbk),cols(nbk),cole(nbk)
      dimension frame1(nx,ny),frame2(nx,ny)
      dimension e(nbk),v(nbk)
      dimension om(nbk),rm(nbk),ov(nbk),rv(nbk),cv(nbk),p(nbk)
      integer  rows,rowe,cols,cole
c
c     define small block A,B,C,D,E 32*32 size
c
      rows(1)=1
      rowe(1)=32
      cols(1)=240
      cole(1)=271
      rows(2)=184
      rowe(2)=215
      cols(2)=1
      cole(2)=32
      rows(3)=369
      rowe(3)=400
      cols(3)=240
      cole(3)=271
      rows(4)=184
      rowe(4)=215
      cols(4)=481
      cole(4)=512
      rows(5)=184
      rowe(5)=215
      cols(5)=240
```

```
      cole(5)=271
      do  5 i=1,nbk
      e(i)=0.
      v(i)=0.
      om(i)=0.
      rm(i)=0.
      ov(i)=0.
      rv(i)=0.
      cv(i)=0.
      p(i)=0.
 5    continue
c
c     cross correlation criterion
c
      do 10 k=1,nbk
      do 10 i=rows(k),rowe(k)
      do 10 j=cols(k),cole(k)
      e(k)=e(k)+(frame1(i,j)-frame2(i,j))
      v(k)=v(k)+(frame1(i,j)-frame2(i,j))**2
      om(k)=om(k)+frame1(i,j)
      rm(k)=rm(k)+frame2(i,j)
      ov(k)=ov(k)+frame1(i,j)**2
      rv(k)=rv(k)+frame2(i,j)**2
      cv(k)=cv(k)+frame1(i,j)*frame2(i,j)
10    continue
      icount=0
      do 15 i=1,nbk
      e(i)=e(i)/(32.*32.)
      v(i)=v(i)/(32.*32.)-e(i)**2
      om(i)=om(i)/(32.*32.)
      rm(i)=rm(i)/(32.*32.)
      ov(i)=ov(i)/(32.*32.)-(om(i)**2)
      rv(i)=rv(i)/(32.*32.)-(rm(i)**2)
      cv(i)=(cv(i)/(32.*32.))-(om(i)*rm(i))
      p(i)=cv(i)/sqrt(ov(i)*rv(i))
      if(p(i).le.0.2)icount=icount+1
 15   continue
      isc=0
```

```
      if(icount.ge.3)isc=1
      write(10,*)'DF variances=',(v(i),i=1,nbk)
      write(10,*)'cross correlation = ',(p(i),i=1,nbk)
      return
      end

c
c     Still Image Coding
c
      subroutine  still(err1,snr,snr1)
      dimension   adct(8,8),adcti(8,8),rpic(400,512),err1(400,512),
     +   pic(400,512),im(8,8),pp(8,8),ot(8,8),
     +   iml(8,8),ppl(8,8),ot1(400,512),vc1(1,3200),
     +   vc2(8,3200),vc3(16,3200),
     +   cvc2(8,1024),cvc3(16,1024),temp(16,3200),tmp(8,3200)
      real    pic,im,pp,ot,iml,ppl,ot1,adct,adcti,pi,tt0,min,
     +   mean1,mean2,van1,van2,mse,mae
      integer    y
c
c     read codebooks
c
      open(2,file='81024.vtr')
      open(3,file='161024.vtr')
      do 20 i=1,400
      do 20 j=1,512
      pic(i,j)=err1(i,j)
 20   continue
c
c     Define 2D 8*8 DCT
c
      pi=3.1415927
      do 5001 k=1,8
      adct(1,k)=0.35355
 5001 continue
      tt0=((1./4.)**(.5))
      do 5002 k=2,8
      do 5003 n=1,8
      adct(k,n)=tt0*cos((2*(n-1)+1)*(k-1)*pi/16.0)
```

```
 5003  continue
 5002  continue
c
c      define inverse D.C.T
c
       do 5005 i=1,8
       do 5006 k=1,8
       adcti(i,k)=adct(k,i)
 5006  continue
 5005  continue

       y=1
       do 30 i=1,50
       do 31 k=1,64
       m=(i-1)*8+1
       mm=m+7
       l=(k-1)*8+1
       ll=l+7
       do 33 n=m,mm
       do 34 j=l,ll
       pp(n-m+1,j-l+1)=pic(n,j)
 34    continue
 33    continue
c
c      D.C.T
c      F=A*f*A~
c
       do 36 n=1,8
       do 37 j=1,8
       t0=0.0
       do 38 nn=1,8
       t0=adct(n,nn)*pp(nn,j)+t0
 38    continue
       im(n,j)=t0
 37    continue
 36    continue
       do 40 n=1,8
       do 41 j=1,8
```

```
      t1=0.0
      do 42 nn=1,8
      t1=im(n,nn)*adcti(nn,j)+t1
 42   continue
      ot(n,j)=t1
 41   continue
 40   continue
c
c     zonal masking
c     create 3 coefficients bands
c
      vc1(1,y)=ot(1,1)

      vc2(1,y)=ot(2,1)
      vc2(2,y)=ot(1,2)
      vc2(3,y)=ot(1,3)
      vc2(4,y)=ot(3,1)
      vc2(5,y)=ot(8,1)
      vc2(6,y)=ot(1,4)
      vc2(7,y)=ot(2,2)
      vc2(8,y)=ot(4,1)

      vc3(1,y)=ot(3,2)
      vc3(2,y)=ot(5,1)
      vc3(3,y)=ot(1,5)
      vc3(4,y)=ot(7,1)
      vc3(5,y)=ot(2,3)
      vc3(6,y)=ot(6,1)
      vc3(7,y)=ot(4,2)
      vc3(8,y)=ot(3,3)
      vc3(9,y)=ot(8,2)
      vc3(10,y)=ot(2,4)
      vc3(11,y)=ot(1,6)
      vc3(12,y)=ot(5,2)
      vc3(13,y)=ot(4,3)
      vc3(14,y)=ot(7,2)
      vc3(15,y)=ot(8,3)
      vc3(16,y)=ot(6,2)
```

```
      y=y+1
 31   continue
 30   continue
c
c     read codebook 81024.vtr
c
      read(2,*)((cvc2(i,j),j=1,1024),i=1,8)
      close(2)
c
c     choose optimum codeword
c
      do 320 j=1,3200
      min=10.**20
      do 340 jj=1,1024
      sum=0.
      do 360 i=1,8
      sum=sum+(vc2(i,j)-cvc2(i,jj))**2
 360  continue
      if(sum.ge.min) goto 340
      min=sum
      do 380 i=1,8
      tmp(i,j)=cvc2(i,jj)
 380  continue
 340  continue
      do 420 i=1,8
      vc2(i,j)=tmp(i,j)
 420  continue
 320  continue
c
c     read codebook 161024.vtr
c
      read(3,*)((cvc3(i,j),j=1,1024),i=1,16)
      close(3)
c
c     choose optimum codeword
c
      do 325 j=1,3200
      min=10.**20
```

```
      do 345 jj=1,1024
      sum=0.
      do 365 i=1,16
      sum=sum+(vc3(i,j)-cvc3(i,jj))**2
 365  continue
      if(sum.ge.min) goto 345
      min=sum

      do 385 i=1,16
      temp(i,j)=cvc3(i,jj)
 385  continue
 345  continue
      do 425 i=1,16
      vc3(i,j)=temp(i,j)
 425  continue
 325  continue
c
c     clear ot(i,j) matrix
c
      do 666 i=1,8
      do 666 j=1,8
      ot(i,j)=0.
 666  continue

      y=1
      do 3000 i=1,50
      do 3000 k=1,64
      m=(i-1)*8+1
      mm=m+7
      l=(k-1)*8+1
      ll=l+7
      ot(1,1)=vc1(1,y)

      ot(2,1)=vc2(1,y)
      ot(1,2)=vc2(2,y)
      ot(1,3)=vc2(3,y)
      ot(3,1)=vc2(4,y)
      ot(8,1)=vc2(5,y)
```

```
      ot(1,4)=vc2(6,y)
      ot(2,2)=vc2(7,y)
      ot(4,1)=vc2(8,y)

      ot(3,2)=vc3(1,y)
      ot(5,1)=vc3(2,y)
      ot(1,5)=vc3(3,y)
      ot(7,1)=vc3(4,y)
      ot(2,3)=vc3(5,y)
      ot(6,1)=vc3(6,y)
      ot(4,2)=vc3(7,y)
      ot(3,3)=vc3(8,y)
      ot(8,2)=vc3(9,y)
      ot(2,4)=vc3(10,y)
      ot(1,6)=vc3(11,y)
      ot(5,2)=vc3(12,y)
      ot(4,3)=vc3(13,y)
      ot(7,2)=vc3(14,y)
      ot(8,3)=vc3(15,y)
      ot(6,2)=vc3(16,y)

      do 3100 n=m,mm
      do 3150 j=l,ll
      ot1(n,j)=ot(n-m+1,j-l+1)
3150  continue
3100  continue
      y=y+1
3000  continue

      do 300 i=1,50
      do 305 k=1,64
      m=(i-1)*8+1
      mm=m+7
      l=(k-1)*8+1
      ll=l+7
      do 310 n=m,mm
      do 315 j=l,ll
      ot(n-m+1,j-l+1)=ot1(n,j)
```

```
 315   continue
 310   continue
c
c      inverse D.C.T
c      f=A~*F*A
c
       do 60 n=1,8
       do 61 j=1,8
       t2=0.0
       do 62 nn=1,8
       t2=adcti(n,nn)*ot(nn,j)+t2
 62    continue
       iml(n,j)=t2
 61    continue
 60    continue

       do 65 n=1,8
       do 66 j=1,8
       t4=0.0
       do 67 nn=1,8
       t4=iml(n,nn)*adct(nn,j)+t4
 67    continue
       ppl(n,j)=t4
 66    continue
 65    continue

       do 400 n=m,mm
       do 410 j=1,ll
       rpic(n,j)=ppl(n-m+1,j-l+1)
       if(rpic(n,j).lt.0.)rpic(n,j)=0.
       if(rpic(n,j).gt.255.)rpic(n,j)=255.
 410   continue
 400   continue
 305   continue
 300   continue

       do 499 i=1,400
       do 499 j=1,512
```

```
      err1(i,j)=rpic(i,j)
499   continue
      total=0.
      do 520 i=1,400
      do 520 j=1,512
      total=total+(pic(i,j)-rpic(i,j))**2/(400.*512.)
520   continue
      ratio=(255.*255.)/total
      snr=10*alog10((255.*255.)/total)

      t=0.
      do 521 i=1,400
      do 521 j=1,512
      t=t+(pic(i,j)**2/(400.*512.))
521   continue
      snr1=10*alog10(t/total)

      mean1=0.
      mean2=0.
      do 522 i=1,400
      do 522 j=1,512
      mean1=pic(i,j)+mean1
      mean2=rpic(i,j)+mean2
522   continue

      van1=0.
      van2=0.
      do 523 i=1,400
      do 523 j=1,512
      van1=pic(i,j)*pic(i,j)+van1
      van2=rpic(i,j)*rpic(i,j)+van2
523   continue

      mean1=mean1/(400.*512.)
      mean2=mean2/(400.*512.)
      van1=van1/(400.*512.)-mean1*mean1
      van2=van2/(400.*512.)-mean2*mean2
```

```
      to=0.
      do 525 i=1,400
      do 525 j=1,512
      to=to+(pic(i,j)-rpic(i,j))**2
525   continue
      mse=to/(400.*512.)

      too=0.
      errmean=0.
      do 526 i=1,400
      do 526 j=1,512
      too=too+abs(pic(i,j)-rpic(i,j))
      errmean=errmean+(pic(i,j)-rpic(i,j))
526   continue
      mae=too/(400.*512.)
      errmean=errmean/(400.*512.)
      errvan=mse-errmean**2
      write(10,*)'***** scene change *****'
      write(10,*)mean1,mean2,van1,van2,to,mse,too,mae,
     + errvan,ratio,snr,snr1
      return
      end

c
c     encoding MCFD signal with 1-D HDCT technique
c
      subroutine hdct1(err1,pgp)
      parameter(nt=32,dis=5.)
      parameter(nu=400,nv=512,ix=4,iy=8)
      parameter(nclus2=1024,nclus3=1024,nclus4=1024)
      dimension  adct(32,32),adcti(32,32),err1(nu,nv),
     + pic(nu,nv),im(32,1),pp(ix,iy),vs16(100,64),
     + iml(16,1),ppl(ix,iy),ppp(32,1),
     + cvc2(16,nclus2),cvc3(16,nclus3),cvc4(16,nclus4),
     + pgp(4),igp(4)
      real    pic,im,pp,iml,ppl,adct,adcti,pi,ppp

      open(21,file='gp21024.vtr')
```

```
      open(31,file='gp31024.vtr')
      open(41,file='gp41024.vtr')
c
c     read codbooks
c
      read(21,*)((cvc2(i,j),j=1,nclus2),i=1,16)
      close(21)
      read(31,*)((cvc3(i,j),j=1,nclus3),i=1,16)
      close(31)
      read(41,*)((cvc4(i,j),j=1,nclus4),i=1,16)
      close(41)

      do 20 i=1,nu
      do 20 j=1,nv
      pic(i,j)=err1(i,j)
 20   continue
c
c     define DCT
c
      pi=3.1415927
      do 5001 k=1,8
      adct(1,k)=0.35355
 5001 continue
      tt0=((1./4.)**(.5))
      do 5002 k=2,8
      do 5003 n=1,8
      adct(k,n)=tt0*cos((2*(n-1)+1)*(k-1)*pi/16.0)
 5003 continue
 5002 continue
c
c     extend DCT(8,8) to DCT(32,32)
c     take Hadamard type scan
c
      ntt=nt/4
      do 5008 i=1,ntt
      do 5008 j=1,ntt
      adct(i,j+ntt)=adct(i,j)
      adct(i+ntt,j)=adct(i,j)
```

```
      adct(i+ntt,j+ntt)=-adct(i,j)
 5008 continue
      nnt=nt/2
      do 5009 i=1,nnt
      do 5009 j=1,nnt
      adct(i,j+nnt)=adct(i,j)
      adct(i+nnt,j)=adct(i,j)
      adct(i+nnt,j+nnt)=-adct(i,j)
 5009 continue
      do 5010 i=1,nt
      do 5010 j=1,nt
      adct(i,j)=adct(i,j)/2
 5010 continue
c
c     define inverse D.C.T
c
      do 5005 i=1,nt
      do 5006 k=1,nt
      adcti(i,k)=adct(k,i)
 5006 continue
 5005 continue
c
c     variance grouping
c
      esum=0.
      vsum=0.
      do 230 i=1,(nu/ix)
      do 231 k=1,(nv/iy)
      m=(i-1)*ix+1
      mm=m+(ix-1)
      l=(k-1)*iy+1
      ll=l+(iy-1)
      do 233 n=m,mm
      do 234 j=l,ll
      pp(n-m+1,j-l+1)=pic(n,j)
 234  continue
 233  continue
```

```
      ppp(1,1)=pp(1,1)
      ppp(2,1)=pp(1,2)
      ppp(3,1)=pp(1,3)
      ppp(4,1)=pp(1,4)
      ppp(5,1)=pp(1,5)
      ppp(6,1)=pp(1,6)
      ppp(7,1)=pp(1,7)
      ppp(8,1)=pp(1,8)
      ppp(9,1)=pp(2,8)
      ppp(10,1)=pp(2,7)
      ppp(11,1)=pp(2,6)
      ppp(12,1)=pp(2,5)
      ppp(13,1)=pp(2,4)
      ppp(14,1)=pp(2,3)
      ppp(15,1)=pp(2,2)
      ppp(16,1)=pp(2,1)
      ppp(17,1)=pp(3,1)
      ppp(18,1)=pp(3,2)
      ppp(19,1)=pp(3,3)
      ppp(20,1)=pp(3,4)
      ppp(21,1)=pp(3,5)
      ppp(22,1)=pp(3,6)
      ppp(23,1)=pp(3,7)
      ppp(24,1)=pp(3,8)
      ppp(25,1)=pp(4,8)
      ppp(26,1)=pp(4,7)
      ppp(27,1)=pp(4,6)
      ppp(28,1)=pp(4,5)
      ppp(29,1)=pp(4,4)
      ppp(30,1)=pp(4,3)
      ppp(31,1)=pp(4,2)
      ppp(32,1)=pp(4,1)
c
c     D.C.T
c     F=A*f
c
      do 236 n=1,(ix*iy)
      t0=0.0
```

```
      do 238 nn=1,(ix*iy)
      t0=adct(n,nn)*ppp(nn,1)+t0
 238  continue
      im(n,1)=t0
 236  continue
c
c     zonal masking
c
      iml(1,1)=im(1,1)
      iml(2,1)=im(2,1)
      iml(3,1)=im(3,1)
      iml(4,1)=im(4,1)
      iml(5,1)=im(5,1)
      iml(6,1)=im(9,1)
      iml(7,1)=im(10,1)
      iml(8,1)=im(11,1)
      iml(9,1)=im(12,1)
      iml(10,1)=im(13,1)
      iml(11,1)=im(17,1)
      iml(12,1)=im(18,1)
      iml(13,1)=im(24,1)
      iml(14,1)=im(25,1)
      iml(15,1)=im(26,1)
      iml(16,1)=im(32,1)

      do 225 jc=1,(ix*iy)
      esum=esum+im(jc,1)/(ix*iy)
      vsum=vsum+im(jc,1)**2/ (ix*iy)
 225  continue
      vsm=0.
      do 226 jc=1,16
      vsm=vsm+iml(jc,1)**2/(ix*iy)
 226  continue
      vs16(i,k)=vsm
 231  continue
 230  continue
      esum=esum/((nu/ix)*(nv/iy))
      vsum=vsum/((nu/ix)*(nv/iy))-esum**2
```

```
      do 229 i=1,(nu/ix)
      do 229 j=1,(nv/iy)
      vs16(i,j)=vs16(i,j)-esum**2
229   continue

      do 222 i=1,4
      igp(i)=0
      pgp(i)=0.
222   continue
      do 30 i=1,nu/ix
      do 31 k=1,nv/iy
      m=(i-1)*ix+1
      mm=m+(ix-1)
      l=(k-1)*iy+1
      ll=l+(iy-1)
      do 33 n=m,mm
      do 34 j=l,ll
      pp(n-m+1,j-l+1)=pic(n,j)
34    continue
33    continue

      ppp(1,1)=pp(1,1)
      ppp(2,1)=pp(1,2)
      ppp(3,1)=pp(1,3)
      ppp(4,1)=pp(1,4)
      ppp(5,1)=pp(1,5)
      ppp(6,1)=pp(1,6)
      ppp(7,1)=pp(1,7)
      ppp(8,1)=pp(1,8)
      ppp(9,1)=pp(2,8)
      ppp(10,1)=pp(2,7)
      ppp(11,1)=pp(2,6)
      ppp(12,1)=pp(2,5)
      ppp(13,1)=pp(2,4)
      ppp(14,1)=pp(2,3)
      ppp(15,1)=pp(2,2)
      ppp(16,1)=pp(2,1)
      ppp(17,1)=pp(3,1)
```

```
      ppp(18,1)=pp(3,2)
      ppp(19,1)=pp(3,3)
      ppp(20,1)=pp(3,4)
      ppp(21,1)=pp(3,5)
      ppp(22,1)=pp(3,6)
      ppp(23,1)=pp(3,7)
      ppp(24,1)=pp(3,8)
      ppp(25,1)=pp(4,8)
      ppp(26,1)=pp(4,7)
      ppp(27,1)=pp(4,6)
      ppp(28,1)=pp(4,5)
      ppp(29,1)=pp(4,4)
      ppp(30,1)=pp(4,3)
      ppp(31,1)=pp(4,2)
      ppp(32,1)=pp(4,1)
c
c     D.C.T
c     F=A*f
c
      do 36 n=1,(ix*iy)
      t0=0.0
      do 38 nn=1,(ix*iy)
      t0=adct(n,nn)*ppp(nn,1)+t0
 38   continue
      im(n,1)=t0
 36   continue

      iml(1,1)=im(1,1)
      iml(2,1)=im(2,1)
      iml(3,1)=im(3,1)
      iml(4,1)=im(4,1)
      iml(5,1)=im(5,1)
      iml(6,1)=im(9,1)
      iml(7,1)=im(10,1)
      iml(8,1)=im(11,1)
      iml(9,1)=im(12,1)
      iml(10,1)=im(13,1)
      iml(11,1)=im(17,1)
```

```
      iml(12,1)=im(18,1)
      iml(13,1)=im(24,1)
      iml(14,1)=im(25,1)
      iml(15,1)=im(26,1)
      iml(16,1)=im(32,1)
c
c     adaptive vector quantization band signal
c     based on band variance grouping
c
      if(vs16(i,k).le.dis)then
      igp(1)=igp(1)+1
      do 127 ig=1,16
      iml(ig,1)=0.
 127  continue
      else if(vs16(i,k).le.10.)then
      igp(2)=igp(2)+1
      call vcb(nclus2,iml,cvc2)
      else if(vs16(i,k).le.30.)then
      igp(3)=igp(3)+1
      call vcb(nclus3,iml,cvc3)
      else if(vs16(i,k).gt.30.)then
      igp(4)=igp(4)+1
      call vcb(nclus4,iml,cvc4)
      endif

      do 126 jc=1,(ix*iy)
      im(jc,1)=0.
 126  continue
      im(1,1)=iml(1,1)
      im(2,1)=iml(2,1)
      im(3,1)=iml(3,1)
      im(4,1)=iml(4,1)
      im(5,1)=iml(5,1)
      im(9,1)=iml(6,1)
      im(10,1)=iml(7,1)
      im(11,1)=iml(8,1)
      im(12,1)=iml(9,1)
      im(13,1)=iml(10,1)
```

```
      im(17,1)=iml(11,1)
      im(18,1)=iml(12,1)
      im(24,1)=iml(13,1)
      im(25,1)=iml(14,1)
      im(26,1)=iml(15,1)
      im(32,1)=iml(16,1)
c
c     inverse DCT
c
      do 60 n=1,(ix*iy)
      t2=0.0
      do 62 nn=1,(ix*iy)
      t2=adcti(n,nn)*im(nn,1)+t2
 62   continue
      ppp(n,1)=t2
 60   continue

      ppl(1,1)=ppp(1,1)
      ppl(1,2)=ppp(2,1)
      ppl(1,3)=ppp(3,1)
      ppl(1,4)=ppp(4,1)
      ppl(1,5)=ppp(5,1)
      ppl(1,6)=ppp(6,1)
      ppl(1,7)=ppp(7,1)
      ppl(1,8)=ppp(8,1)
      ppl(2,8)=ppp(9,1)
      ppl(2,7)=ppp(10,1)
      ppl(2,6)=ppp(11,1)
      ppl(2,5)=ppp(12,1)
      ppl(2,4)=ppp(13,1)
      ppl(2,3)=ppp(14,1)
      ppl(2,2)=ppp(15,1)
      ppl(2,1)=ppp(16,1)
      ppl(3,1)=ppp(17,1)
      ppl(3,2)=ppp(18,1)
      ppl(3,3)=ppp(19,1)
      ppl(3,4)=ppp(20,1)
      ppl(3,5)=ppp(21,1)
```

```
      ppl(3,6)=ppp(22,1)
      ppl(3,7)=ppp(23,1)
      ppl(3,8)=ppp(24,1)
      ppl(4,8)=ppp(25,1)
      ppl(4,7)=ppp(26,1)
      ppl(4,6)=ppp(27,1)
      ppl(4,5)=ppp(28,1)
      ppl(4,4)=ppp(29,1)
      ppl(4,3)=ppp(30,1)
      ppl(4,2)=ppp(31,1)
      ppl(4,1)=ppp(32,1)

      do 133 n=m,mm
      do 134 j=l,ll
      err1(n,j)=ppl(n-m+1,j-l+1)
134   continue
133   continue
31    continue
30    continue
      do 223 i=1,4
      pgp(i)=float(igp(i))/((nu/ix)*(nv/iy))
223   continue
      return
      end
```

Bibliography

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

[2] To R. Hsing and Kuo-Hu Tzou, "Video compression techniques: A review," *IEEE Global Telecomm. Conf.,* Atlenta, Georgia, pp. 1521-1526, Nov. 26-29, 1984.

[3] C. E. Shannon, "A mathematical theory of communication," Bell syst. tech. J., vol. 27, pp. 379-423, 623-659, 1948.

[4] C. E. Shannon, "Coding theorems for a discrete source with a fidelity criterion," *IRE National Convention Record,* part 4, pp, 142-163, 1959.

[5] R. M. Gray, "Vector quantization," *IEEE ASSP Mag.*, pp. 4-29, Apr. 1984.

[6] Y. Linde, A. Buzo and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Comm.,* vol. COM-28, no. 1, pp. 84-95, Jan. 1980.

[7] N. Ahmed and K. Rao, "Orthogonal for Digital Signal Processing," New York : Springer-Verlag, 1975.

[8] H. Andrew, "Computer Techniques in Image Processing," New York : Academic Press, 1970.

[9] R. Zelinski and P. Noll, "Adaptive Transform Coding of Speech Signals", *IEEE Trans. on ASSP.,* vol. ASSP-25, no.4, pp 299-309, Aug. 1977.

[10] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform", *IEEE Trans. comput.,* vol C-23, pp 90-93. Jan. 1974.

[11] R. A. Haddad and A. N. Akansu, "A new orthogonal transform for signal coding," *IEEE Trans. ASSP,* vol. ASSP-36, pp. 1404-1411, Sept. 1988.

[12] A. Hetravali, and J. Limb, "Picture Coding : A Review," *Proc. IEEE,* vol. 68, pp 366-406, Mar. 1980.

[13] Thomas J. Lynch, "Data compression techniques and applications," Van Nostrand Reinhold, N. Y., 1985.

[14] A. Puri, H. M. Hang and D. L. Schiling, "An efficient block-matching algorithm for motion compensated coding," *Proc. IEEE ICASSP,* Apr. 1987.

[15] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Comm.* vol. COM-29, pp. 1799-1808, Dec. 1981.

[16] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Comm.* vol. COM-33, pp. 888-896, Aug. 1985.

[17] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat. Telecomm. Conf.,* New Orleans, L.A., pp. G 5.3.1-G 5.3.5, Nov. 29-Dec. 3, 1981.

[18] S. Kappagantula and K. R. Rao, "Motion predictive interframe coding," *IEEE Trans. Comm.* vol. COM-33, pp. 1011-1015, Sept. 1985.

[19] A. Gersho and V. Cuperman, "Vector quantization: A pattern-matching technique for speech coding," *IEEE Comm. Mag.,* pp. 15-21, Dec. 1983.

[20] S. P. Lloyd, "Least-squares Quantization in PCM," *IEEE Trans. on Inform. Theory,* vol. IT-28, pp. 129-137, Mar. 1982.

[21] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. on Comm.,* vol. COM-36, no. 8, pp. 957-971, Aug. 1988.

[22] N. S. Jayant and P. Noll, "Digital Coding of Waveforms," N.J., Prentice-Hall, 1984.

[23] P. Winza, "Transform Picture Coding", *Proc. IEEE,* vol. 60, pp 809-820, July 1972.

[24] A. N. Akansu and M. S. Kadur, "DWHT-VQ for motion compensated frame difference signal coding," *Proc. IEEE ICASSP,* Glasgow, pp. 1862-1865, 1989.

[25] A. N. Akansu and M. S. Kadur, "Adaptive vector quantization of video signals with motion compensation and spatial masking," *Proc. IEEE ISCAS,* Portland, pp. 1378-1381, 1989.

[26] Y. Kota, N. Mukawa, S. Okubo, H. Hashimoto, H. Yosuda, "DCT coding for video conferencing using vector and scalar quantization," *Proc. IEEE Globecom'86,* pp. 266-270.

[27] Y. Kota, N.Mukawa, S. Okubo, "A motion picture coding algorithm using adaptive DCT encoding based on coefficient power distribution classification," *IEEE JSAC,* vol. SAC-5, no. 7, pp. 1090-1099, Aug. 1987.