ADAPTIVE HYBRID (MOTION COMPENSATED INTERFRAME TRANSFORM) CODING TECHNIQUE FOR MULTIFRAME IMAGE DATA

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree of
Master of Science

by

Minsok Peck
August, 1987
ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. Joseph E. Essman for his supervision and guidance during the course of this thesis. I am also grateful to Mr. Kou Hu Tzou of GTE, and Mr. Billy Bryant of VICOM Inc. for their assistance in preparing and processing the image data used in this thesis.

Finally, I am most thankful for my parents, and my wife and son for their financial and emotional support throughout my graduate study.
# TABLE OF CONTENTS

<p>| ACKNOWLEDGEMENTS                                      | iii          |
| LIST OF FIGURES                                      | vi           |
| LIST OF TABLES                                       | vii          |
| CHAPTER I. INTRODUCTION                              | 1           |
| 1. 1. Digital Image Transmission Problem             | 2           |
| 1. 2. Current Multiframe Image Coding Techniques     | 8           |
| 1.2.1. Interframe Coding Techniques                 | 9           |
| 1.2.2. Transform Coding Techniques                  | 15          |
| 1.2.3. Hybrid Coding Techniques                     | 20          |
| 1. 3. Thesis Objectives and Organizations            | 22          |
| 1. 4. Description of Experimental Data Set           | 25          |
| CHAPTER II. MOTION COMPENSATED INTERFRAME PREDICTION TECHNIQUES | 28          |
| 2. 1. Temporal Characteristics of Interframe Motion  | 31          |
| 2. 2. Interframe Motion Estimation                   | 37          |
| 2. 3. Results of Motion Estimation                   | 48          |
| CHAPTER III. 2-D TRANSFORM ADAPTIVE CODING           | 59          |
| 3. 1. Statistical Properties of Unitary Transform Samples | 60          |
| 3. 2. 2-D Unitary Transform Adaptive Coding          | 63          |
| 3.2.1. Thresholding and Normalization                | 66          |</p>
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.2.</td>
<td>Quantization</td>
<td>69</td>
<td></td>
</tr>
<tr>
<td>3.2.3.</td>
<td>Coding for Chain Enclosed Coefficients</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>3.3.</td>
<td>Chain Coding</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>3.3.1.</td>
<td>Boundary Tracing Algorithm</td>
<td>73</td>
<td></td>
</tr>
<tr>
<td>3.3.2.</td>
<td>Boundary Encoding</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>CHAPTER IV.</td>
<td>ADAPTIVE HYBRID CODING TECHNIQUE</td>
<td>94</td>
<td></td>
</tr>
<tr>
<td>4.1.</td>
<td>Adaptive Interframe Transform Coding</td>
<td>95</td>
<td></td>
</tr>
<tr>
<td>4.1.1.</td>
<td>Motion Compensated Interframe Prediction</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>4.1.2.</td>
<td>Adaptive Transform Coding Operation</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>4.1.3.</td>
<td>Error Considerations</td>
<td>106</td>
<td></td>
</tr>
<tr>
<td>4.2.</td>
<td>Simulation Results</td>
<td>108</td>
<td></td>
</tr>
<tr>
<td>CHAPTER V.</td>
<td>SUMMARY AND CONCLUSIONS</td>
<td>124</td>
<td></td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>128</td>
<td></td>
</tr>
<tr>
<td>APPENDICES</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Appendix A</td>
<td>VICOM DIGITAL IMAGE PROCESSOR</td>
<td>134</td>
<td></td>
</tr>
<tr>
<td>Appendix B</td>
<td>2-D HADAMARD TRANSFORM</td>
<td>141</td>
<td></td>
</tr>
<tr>
<td>Appendix C</td>
<td>SIMULATION PROGRAM FOR ADAPTIVE HYBRID CODING</td>
<td>151</td>
<td></td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1.</td>
<td>Block Diagram for Digital Image Transmission and Reproduction System</td>
</tr>
<tr>
<td>1-2.</td>
<td>Block Diagram of a Simple Transform Encoding/Decoding Scheme</td>
</tr>
<tr>
<td>1-3.</td>
<td>Some Frames of Original Mike Image Data Set</td>
</tr>
<tr>
<td>2-1.</td>
<td>Block Diagram of Basic Interframe Coding/Decoding Scheme</td>
</tr>
<tr>
<td>2-2.</td>
<td>Geometry for Manipulation in (M x M) Present Sub-block U with Previous Sub-block U_R of Size (M + 2P) x (M + 2P)</td>
</tr>
<tr>
<td>2-3.</td>
<td>The 2-D Logarithmic Search Algorithm</td>
</tr>
<tr>
<td>2-4.</td>
<td>The 2-D Directional Search Procedure</td>
</tr>
<tr>
<td>2-5.</td>
<td>Block Diagram of Motion Compensated Interframe Coding/Decoding Scheme</td>
</tr>
<tr>
<td>2-6.</td>
<td>The Effect of Motion Compensation in Interframe Difference between Present and Previous Frames</td>
</tr>
<tr>
<td>3-1.</td>
<td>Freeman's Chain Code</td>
</tr>
<tr>
<td>3-2.</td>
<td>2-D Unitary Transform Adaptive Coding/Decoding Scheme</td>
</tr>
<tr>
<td>3-3.</td>
<td>Histogram and Cumulative Density Function of the 2-D Ordered Hadamard Transform Coefficients of MIKE12</td>
</tr>
<tr>
<td>3-4.</td>
<td>N-Neighbors with respect to the Point P</td>
</tr>
<tr>
<td>3-5.</td>
<td>Geometry for Manipulation in Transform Sub-block H with Buffer W</td>
</tr>
<tr>
<td>3-6.</td>
<td>Flow Graph of Chain Coding Procedure</td>
</tr>
<tr>
<td>3-7.</td>
<td>The Order of Examination for Finding Next Search Direction</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>3-8.</td>
<td>Right-Neighbor Points for Evaluation of Next Direction</td>
</tr>
<tr>
<td>3-9.</td>
<td>Possible Chain Link Contact Configuration between Present Chain Link (Solid line) and Previous Chain Link (Broken line)</td>
</tr>
<tr>
<td>4-1.</td>
<td>Adaptive Hybrid Coding/Decoding Scheme</td>
</tr>
<tr>
<td>4-2.</td>
<td>Histogram and Cumulative Density Function of the 2-D Ordered Hadamard Transform Coefficients for the Difference Values between MIKE15 and MIKE16</td>
</tr>
<tr>
<td>4-3.</td>
<td>Reconstructed Image by using Transform Adaptive Coding Scheme of Fig. 3-2</td>
</tr>
<tr>
<td>4-4.</td>
<td>Reconstructed Image by using Adaptive Hybrid Coding Scheme of Fig. 4-1</td>
</tr>
<tr>
<td>5-1.</td>
<td>Plot of the Mean-Square Error Performance for both Adaptive Hybrid and Transform adaptive Techniques</td>
</tr>
<tr>
<td>B-1.</td>
<td>Flow Graph for Forward (or Inverse) 1-D Ordered Hadamard Transform Computation</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1.</td>
<td>The Effect of Motion Compensation in terms of Interframe Variance Reduction for Each Sub-block of Size 16 x 16. MIKE12 was used as Reference Frame.</td>
<td>51</td>
</tr>
<tr>
<td>2-2.</td>
<td>The Effect of Motion Compensation in terms of Interframe Variance Reduction for Each Sub-block of Size 16 x 16. MIKE15 was used as Reference Frame.</td>
<td>52</td>
</tr>
<tr>
<td>2-3.</td>
<td>Image Motion Displacement Vector for Each Sub-block of Size 16 x 16</td>
<td>53</td>
</tr>
<tr>
<td>2-4.</td>
<td>The Mean-Square Distortions for All Possible Search Direction. Sub-block Size = 16 x 16 and P=7 were used.</td>
<td>55</td>
</tr>
<tr>
<td>2-5.</td>
<td>The Effect of Motion Compensation in terms of Interframe Variance Reduction in Interframe Coding</td>
<td>56</td>
</tr>
<tr>
<td>3-1.</td>
<td>Huffman Code Table for Transform Coefficient Amplitude in Absolute Value</td>
<td>71</td>
</tr>
<tr>
<td>3-2.</td>
<td>Chain Coding Processes of A Typical Transform Sub-block</td>
<td>88</td>
</tr>
<tr>
<td>3-3.</td>
<td>ASGN Process for Table 3-2 (b) after Chain Coding of Each Boundary</td>
<td>89</td>
</tr>
<tr>
<td>3-4.</td>
<td>Encoding of the Chain Boundaries shown in (b) of Table 3-2.</td>
<td>91</td>
</tr>
<tr>
<td>3-5.</td>
<td>Variable Length Instantaneous Code for Differential Chain Code</td>
<td>93</td>
</tr>
<tr>
<td>4-1.</td>
<td>Intensity Values of a Typical Sub-block in the Present and the Corresponding Sub-block in the Previous Frame</td>
<td>110</td>
</tr>
<tr>
<td>4-2.</td>
<td>Intensity Values of the Motion Compensated Sub-block for the Prediction of the Present Sub-block of Table 4-1</td>
<td>111</td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>4-3. Motion Compensated Prediction Error Values for the Present Sub-block in Table 4-1 and Its 2-D Ordered Hadamard Transform Coefficients</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>4-4. Thresholding, Normalization, and Quantization Processes for the Transform Sub-block in Table 4-3</td>
<td>114</td>
<td></td>
</tr>
<tr>
<td>4-5. Huffman Code Table for the Transform Coefficient Amplitude (in Absolute Value) of the Motion Compensated Prediction Error</td>
<td>115</td>
<td></td>
</tr>
<tr>
<td>4-6. Average Mean-Square Error Performance of the 2-D Transform Adaptive Coding</td>
<td>116</td>
<td></td>
</tr>
<tr>
<td>4-7. Average Mean-Square Error Performance of the Adaptive Hybrid Coding</td>
<td>118</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER I

INTRODUCTION

Digital encoding of image data has gained widespread interest due to the recent progress in digital computers and the expansion of the television signal transmission services such as video-teleconferencing, video-telephone, and satellite image transmission. This is because of the greater flexibility in the nature and scope of the operations which can be carried out on the data. Digital transmission can be considered as a promising and powerful means from the standpoint of channel capacity, particularly when combined with an efficient encoding technique.

An analog image is a function of two spatial coordinates. In order to achieve digital representation of an image, the image must be digitized both spatially and in amplitude without removing much statistical or perceptual redundancy from the image signal. The digitization process of the spatial coordinates is called image sampling. The sampling theory provides the most important step towards data reduction of digital information required to represent a continuous signal. This theory states that any band-limited signal, sampled at a rate faster than twice its highest frequency component, can be reproduced without introducing any distortion. This minimum sampling rate is
known as the Nyquist rate. The amplitude digitization process is then performed on each digitized sample, called a pel or pixel. This process converts image brightness at each sample point to a sufficient fixed number of bits so that the distortion due to this process is not easily visible. The amplitude digitization is called gray-level quantization.

In this chapter the statement of the digital image data compression problem is formulated, and some of the recent image data compression techniques are briefly outlined with emphasis on the multiframe image coding techniques.

1.1 Digital Image Transmission Problem

Digital treatment of analog images has advantages over its analog form in terms of processing flexibility, easy or random access in storage, higher signal-to-noise ratio, and possibility of errorless transmission. However, the enormous data rates generated by the multiframe image signal (e.g., broadcast television) would result in a large memory and/or channel capacity requirement and would introduce quite large transmission costs. For instance, for a typical 4 MHz television signal sampled at the Nyquist sampling rate with an intensity resolution of 8 bits/sample, the data rate for simple PCM transmission of images generated by the multiframe image is approximately sixty million bits per second. The conventional PCM transmission techniques require
a very high data rate (i.e., large bandwidth). Thus, there is great need for reducing the data rate for a given image quality to yield a reduction in transmission costs. A block diagram of a basic digital image transmission and reconstruction system is illustrated in Fig. 1-1.

During the past decade most efforts in this field have been focused on bandwidth compression by removing the redundancy inherent in an image or sequence of images both in spatial and temporal dimensions. Numerous bandwidth compression techniques have been proposed in response to the continually increasing requirements for transmission and/or storage. Most of these techniques are based upon transform and linear predictive coding techniques. For more efficient coding, the two techniques can be combined resulting in hybrid coding, which combines the more attractive features
of both. A comprehensive review of various coding techniques is given by J. O. Limb et al. [1] and A. K. Jain [2].

Both predictive and transform coding techniques are statistical in nature. The basic statistical property related to image compression techniques is inter-pixel correlation, within both a single frame and the frame-to-frame time dimension. Simple statistical or visual analysis of the image raw data reveals that there is a very strong correlation between adjacent pixels, both intraframe and interframe. This high correlation results in significant redundancy in the original image data. Removal of some of this redundancy would reduce the memory size and/or channel capacity required for storage and digital image transmission. The basic problem of data compression is to exploit this redundancy efficiently to reduce data rates.

For the compression of image data, various transform and predictive coding techniques have been investigated independently by many researchers. These coding techniques can be categorized as intraframe and interframe coding techniques. The interframe technique, which has the highest reduction rate for multiframe image data, utilizes the redundancy in the temporal dimension [14 - 22]. On the other hand, the intraframe coding technique, which is developed for a single image frame, is based on the exploitation of spatial redundancy. Among the various intraframe coding techniques, transform coding is well known for its robust performance and high reduction rate [7 - 13]. The efficiency
of these coding techniques can be measured by their ability of data compaction, degree of distortion, and implementation complexity.

Basically, digitized image raw data can be compressed without introducing any further distortion, except for the degradation introduced by digitization process. However, such techniques do not yield large enough data compression ratios for transmission and/or storage. In image transmission systems such as broadcasting television, some errors in the reconstructed image can be tolerated. Thus, it is possible to achieve a much larger compression by introducing an acceptable amount of degradation in the original image data. Therefore, we should provide some fidelity measurement techniques for the degradation of the reconstructed image. The problem of the image data compression can now be thought of as the problem of the minimization of data rates within a given distortion range or the minimization of distortion for a given data rate. The amount of distortion that may be tolerated will depend upon the overall quality that is required for the specific application.
Fidelity Measurement

Measurement techniques of the image quality are categorized as a subjective measurement or an objective measurement. Since the reconstructed images are usually viewed by a human observer, it is more appropriate to use a subjective technique corresponding to how good the image looks to the human observer. Some evaluation methods of this type are described in the references [1], [28], and [29]. A simple subjective measure is to use the pair-comparison between original and encoded images. However, this subjective evaluation is very time consuming and difficult to incorporate in the data compression algorithm due to the lack of knowledge pertaining to the human visual system. Therefore, the objective measurement technique is commonly employed for evaluating the reconstructed image fidelity of data compression techniques.

The most widely used objective measure for the image data compression algorithms is the mean-square error (MSE) criterion. Let \( U(x,y) \) be an intensity sample of the original image array of size \( N \times N \) and \( \hat{U}(x,y) \) be its reconstructed value after data compression. The error between input sample and the corresponding output value is

\[
e(x,y) = U(x,y) - \hat{U}(x,y)
\]

The mean-square error (MSE) due to the data compression is defined as
Another common discrete image fidelity measure is the peak-mean-square error (PMSE) defined in [29] as

\[
PMSE = \frac{MSE}{A^2}
\]  \hspace{1cm} (1-3)

where \( A \) represents the maximum intensity value of \( U(x,y) \).

Since the reconstructed value of the original image can be considered as an input signal plus noise, the mean-square error (MSE) is also represented by the signal-to-noise ratio (SNR), in decibels (dB), as

\[
SNR = -10 \log_{10} (PMSE) \quad (dB)
\]  \hspace{1cm} (1-4)

However, if we are dealing with an image which should be evaluated visually, the mean-square error (MSE) fidelity criterion is not always an accurate measure of visual fidelity. The chief appeal of the mean-square error criterion is its simplicity.
1.2 Current Multiframe Image Coding Techniques

A multiframe image can be viewed as a sequence of frames of a still image, where each frame corresponds to an equi-interval of time or other variables, which depend on the particular application. In the case of a multispectral satellite image, the third dimension could be the path connecting the same spatial elements on successive frames. Although the same principles hold for time and other variables, the inter-scan correlation behavior of other variables is quite different from that of the sequence of images in time dimension. For the application of motion image transmission, such as broadcast television, videophone, and teleconferencing, the frames are presented at frequent intervals. There are well known restrictions on the allowable time interval between successive frames of a moving sequence so that the human visual system can perceive continuous smooth motion without distortion. In the standards of the National Television System Commission (NTSC), motion image frames are presented at a rate of thirty frames per second in U.S. commercial T.V. broadcasts. Since, in practice, image sequence is presented at this high frame rate, there is considerable correlation between successive frames. To reduce the transmission bandwidth and/or storage requirement of image data, much research has been done to investigate an efficient coding scheme.
1.2.1 Interframe Coding Techniques

Since the original image data is assumed to be highly correlated, the image pixels lying in the same neighborhood will tend on average to have similar amplitude values. In image predictive coding a prediction of the intensity value of a current pixel in a raster scanned uses of one or more earlier pixels, which have been previously scanned and coded in the same line, or in a previous line or lines, or frame. Due to the Markovian nature of the image data, the neighborhood pixels used for the prediction are limited to a small set of pixels in the previous and present frames. Differential Pulse Code Modulation (DPCM), interframe, and other predictive coding techniques belong to this category.

Although DPCM coding was originally an intraframe coding technique which utilizes inter-pixel correlation within a single frame, its concept can be easily extended to frame-to-frame DPCM with the knowledge that the majority of the pixels in a present frame do not vary considerably from the corresponding pixels of the previous frame. In frame-to-frame DPCM, which can be referred to as interframe DPCM, only the difference between consecutive frames is transmitted. For multiframe image data usually generated by a broadcast television or a video camera, interframe coding is more efficient than intraframe DPCM coding.

In [16], F. W. Mounts uses the technique that takes advantage of frame-to-frame correlation to reduce the
transmission bandwidth for television signals. Since only the changed part between frames is coded and transmitted, this technique is called conditional replenishment. The basic idea of the conditional replenishment system is that, given a reference frame memory stored at both transmitter and receiver, a comparison is made with the next frame to be coded and then only those elements whose absolute magnitude of prediction error is larger than a threshold are quantized and coded together with the addressing information of the pixel.

In the paper of J. C. Candy et al. [17], the idea of the pixel-by-pixel basis operation in the conditional replenishment technique of F. W. Mounts [16] has been improved with the knowledge that the significant interframe differences tend to appear in clusters crowded near brightness edges along the time dimension. Additional improvement in efficiency has also been obtained by employing a subsampling method, which takes advantage of the fact that reduced resolution is tolerable in changing scenes. Another improvement is gained by transmitting the beginning and terminating address information of a cluster. With the increased complexity in hardware, a more enhanced result than that of F. W. Mounts [16] has been obtained by J. C. Candy et al. [17]. The reconstructed image has good quality for still and slowly moving scenes. However, there is noticeable degradation of rapidly moving edges. Since the previous frame is repeated whenever the prediction signal is
not replenished, most degradation is in the temporal direction.

The coding techniques described by F. W. Mounts [16] and J. C. Candy et al. [17] are designed at a rate of approximately one bit per pixel and are well suited to operate under the conditions that the video camera is not moved very often and that scenes consist of stationary background areas with moving objects in the foreground. With the conditional replenishment techniques of F. W. Mounts [16] and J. C. Candy et al. [17], only the information pertaining to the moving areas of the image frame is transmitted. Addressing information of each pixel transmitted is efficiently coded, using the segmentation method. This method also called clustering, divides each frame into moving and stationary areas instead of pointwise coding of F. W. Mounts [16]. The simulations of these coders are carried out on data sampled at about $2 \times 10^6$ samples/sec (e.g., low resolution T.V. signal). However, in many applications high resolution with data sampled at about $8 \times 10^6$ samples/sec is needed. An interframe coder of 525 lines monochrome television is implemented by B. G. Haskell et al. [18], which compresses the data rate to 0.19 bits/pixel. This adaptive interframe coding scheme uses the conditional replenishment of F. W. Mounts [16] together with a higher-order predictor, moving area segmenter, temporal filter, and elastic buffer. Using the basic assumption that the camera is largely stationary and moving objects within an image
frame do not move rapidly, acceptable quality for the possible applications to video-teleconferencing or videotelephone has been obtained at the transmission bit rate of 1.5 Mbits/sec.

In the interframe coding techniques of F. W. Mounts [16], J. C. Candy et al. [17], and B. G. Haskell et al. [18], the motion prediction has been made by the intensity difference between the present and previous pixel on the same position. For scenes with small motion and low detail conventional interframe coding techniques produce a small amount of information to be transmitted, and therefore it is possible to reduce the coding data rate greatly. However, there is a disadvantage when the objects' motion on the scene is increased further between frames; the amplitude of the frame difference signal is increased, and the correlation between the present pixel and previous pixel is reduced accordingly. Thus, the image pixel on the same position in the previous frame is no longer a good sample for prediction. Rather, the image pixel on the same position of the moving object in the previous frame will be a better sample for prediction. Using an estimate of the objects' motion is in the field of view of a video camera, more efficient interframe coding can be performed by taking differences of pixels with respect to pixels in the previous frame that are appropriately translated spatially. Thus, the nature and direction of the interframe motion in multiframe images should be analyzed to eliminate the disadvantage due
to the objects' movement. Such an adaptive coding prediction technique has been called motion compensated prediction.

Interframe motion can be thought as the intensity difference between consecutive frames. This frame difference signal includes translation and rotation of the objects with respect to the camera. The obvious difficulty which accompanies any attempt to code moving detail is to identify properties inherent to motion which may be used to enhance the performance of a multiframe image coding system. Because of the dimensionality and computational problems, the motion of interframe images is usually assumed to be a simple translational shift in location; i.e., the objects move in an arbitrary direction within the frame, but without rotation or change in their shape or size.

Cafforio and Rocca[22] have investigated the techniques for estimating small displacements and segmenting an image frame into fixed and moving parts for different movements of a single, moving object with a stationary background. The technique is generalized to more than one moving object. By assuming the pure translation of the interframe motion, simple formulas are derived where image motion can be estimated from frame difference and horizontal and vertical differences. Since the information related to the segmentation needs to be coded, this technique becomes complex as the number of moving objects increases and the size of the image grows larger.

Another approach to motion prediction and compensation
method is a technique by Netravali and Robbins [14], called the "Pel recursive method", which recursively adjusts the motion estimate at each pixel by using its neighboring pixels, which have already been coded. The iterations of this algorithm continue until the motion estimates converge to the true displacement. Because the motion vector for a pixel displacement can be calculated at the receiver, this technique needs no information related to the motion vector to be transmitted. However, such an approach involves computational complexity and implementational difficulty.

A much simpler method of interframe motion prediction and compensation, called the "Block matching method", particularly with regard to an algorithm for searching for the direction of displacement, has been considered by Jain and Jain [15]. The searching procedure used in this method has been improved by Kappagantula and Rao [21]. In these prediction techniques, the image frame is first segmented into small, fixed sized sub-blocks to avoid the transmission of segmentation information, and the movement in each sub-block is assumed to be a linear translational shift. Motion compensation is done by shifting each sub-block independently. At low data rates, the addition of motion compensation is desirable and will improve efficiency by approximately 50% or more without any visible degradation [2]. During the past few years the techniques of motion prediction and compensation have become increasingly important in the areas of coding for multiframe image data.
1.2.2 Transform Coding Techniques

The unitary transform coding method on two dimensional image data is a highly effective means of removing the redundancy in original time or space dependent image data. This coding technique can be applied to the coding of moving image sequences without taking into account the relationship which naturally exists between consecutive frames.

The transform operation, which may be seen as producing approximately uncorrelated spectral coefficients, results in drastic redistribution of image energy into relatively few, low-order coefficients. The ability to pack the image signal energy into few transform coefficients depends on the choice of the transformation. An optimum transform for random image data would result in the best reconstruction image quality using the fewest bits, but the quantitative description of this criterion is difficult to specify clearly.

Alternatively, the optimum transform can be specified by one that produces statistically independent coefficients. However, because such a transformation requires higher order statistics of images, it can not be generally determined [1]. The only transform that generates a set of uncorrelated coefficients is known as the Karhunen-Loeve transform (or Hotelling transform). A detailed discussion of 1-D and 2-D Karhunen-Loeve transforms can be found in the references [10] and [11]. Although the Karhunen-Loeve transform is explicitly known to be the best in the mean-square error
sense, it is not commonly used in practice. Other suboptimum transforms, which results in approximately uncorrelated coefficients, are usually employed due to the computational simplicity and the ease of implementations. Many different types of suboptimum unitary transforms—Fourier, cosine, sine, Hadamard, slant transforms, and others—have been investigated for transform image coding. Basic theories and computation techniques of various unitary transforms are given, in detail, by R. C. Gonzalez and P. A. Wintz [28], W. K. Pratt [29], and R. J. Clarke [30], and the alternative interpretation for the correlation reducing property of the transform operation is described by Wintz[10] and Habibi[4].

In basic transform coding, sometimes called block quantization, the image data is divided into sub-blocks of smaller size, and then each sub-block is transformed into a set of statistically more independent coefficients. Bandwidth compression is achieved by transmitting only a subset of these coefficients, suitably quantized and coded. At the receiver the incoming data is decoded, and an inverse transform is performed to reconstruct the original image.

Generally, there are two strategies—zonal sampling and threshold sampling—for selecting the coefficients to be transmitted or stored. In zonal sampling, the coefficients lying within a prespecified zone of the transform coefficients array are selected for subsequent quantization and coding. On the other hand, in threshold sampling, the coefficients with amplitude above a
A predetermined threshold is retained for transmission. This technique is adaptive, because the number of coefficients transmitted is dependent upon the energy content of each sub-block. Although the threshold technique would give the better result for the transform coefficients selection, it has the disadvantage that the address of the transmitted coefficients must also be coded for each sub-block. The block diagram of a basic transform coder is described in Fig. 1-2.

Since the information content of the transform coefficients changes from one block to the other, non-adaptive coding schemes inevitably lead to increased error in image reconstruction. Various adaptive coding techniques have been investigated for coefficient selection, quantization, and bit assignment.
In 1976, Knauer[8] used the 3-D Hadamard transform adaptively to encode various degrees of motions in television signals. Four frames of video data are stored at an accuracy of 6 bits, and this block of four frames is divided into smaller sub-blocks of size 4 x 4 x 4. In the adaptive mode of operation, the degree of motion in each sub-block is monitored by the system and then by specifying the high spatial resolution for low motion areas and exchanging it with temporal resolution in rapid motion areas. This coding technique indicates that the bit rates can be considerably reduced by incorporating the correlation in the temporal direction. However, because of computational complexity and large amounts of storage requirement, the 3-D transform encoding systems have usually been avoided by many practical researchers.

A more practical and probably more effective method is to adapt the bit assignment to changes in sub-block statistics. An efficient adaptive transform coding technique of both monochrome and color image data has been developed by W. Chen and C. H. Smith [7]. In their coding technique, assuming Gaussian DC and AC coefficient probability density functions, the 2-D Fast Discrete Cosine Transform (FDCT) is applied on each sub-block of size 16 x 16. All the transform sub-blocks are classified into one of four classes on the basis of sub-block AC energy and are normalized by the normalization factor, which is simply the estimate of the standard deviation of the transform coefficients. The
normalized transform coefficients within each class are quantized using Max's optimum quantizer, and adaptive bit assignment is then performed using the bit allocation maps for each classes. For real time application this coding system consists of two passes. The first pass estimates the AC coefficients' energy to generate the classification maps to build the bit assignment matrices and normalization factor. The second pass performs normalization, quantization and code words assignment utilizing the information from the first pass. Excellent subjective results are obtained with this system. However, for the purpose of practical use, the difficulties remain with the memory requirement for the second pass and the complexity of the quantization algorithm.

Recently, the adaptive transform coding technique, which uses a single pass consisting of thresholding, normalization and a simple integer floating point round-off operation on the 2-D DCT transform coefficients, is reported by W. Chen and W. K. Pratt [12]. To code the coefficients within each sub-block, the Huffman code for AC coefficients and the block code for the DC coefficient are used. The addressing information of the coefficients retained for transmission is coded using runlength code. Excellent reconstruction of color images at an overall rate of 0.4 bit/pixel (i.e., 1.5 Mbits for 15 frames/sec.) is reported. For more efficient representation of the addressing information of the retained coefficients, a chain coding
algorithm, which can efficiently separate the nonzero coefficient regions from the zero region, has been introduced by J. A. Saghri et al. [13]. By using the chain coding algorithm, a 10 to 30 percent improvement, compared with runlength coding, is reported.

The transform coding technique is relatively complex compared with the predictive coding technique; however, recent advances in digital computer technology and the development of fast signal processing algorithms make it possible to transmit multiframe image data.

1.2.3 Hybrid Coding Techniques

A study of both transform and predictive coding techniques has indicated that each technique has comparative advantages and disadvantages. The predictive coding offers easy implementation and the possibility of the real time coding by minimizing the delay due to coding operation. The disadvantage of this coding technique is its sensitivity to image data statistics. On the other hand, transform coding techniques achieve a superior coding performance at lower bit rates and show less sensitivity to data statistics. However, a transform coder is usually much more complex to implement than a predictive coder. A hybrid coding technique that combines the attractive features of both transform coding and predictive coding has also been studied to realize an efficient technique for transmitting image data.
over a digital communication channel.

Habibi [3] proposed a hybrid transform-DPCM coding technique. In its operation, a 1-D transform is taken along each image line of the N x N image block. Transform coefficients then enter a bank of DPCM coders, which perform previous pixel predictive coding along the original columns, to produce the difference signals. The quantized difference signals from the bank of DPCM coders are then coded for transmission. Q. D. Hua[23] has proposed a intraframe hybrid (i.e., transform-DPCM and DPCM-transform) coding technique using both 1-D and 2-D unitary transformations. His results indicate that, subjectively, hybrid techniques are superior to 2-D transform coding techniques with the exception of the application of the Karhnen-Loeve transform.

For multiframe image data coding, the concept of the prediction operation of the intraframe hybrid coding method can be easily extended to the temporal direction between the corresponding blocks in successive frames. The processing load is significantly reduced, compared with that of 3-D transform coding. This coding technique is called interframe hybrid coding.

Based upon the approximation of Markovian image sources, experimental and theoretical performances have been obtained by Roese et al.[24] for transform coding using 3-D blocks and hybrid transform/DPCM along the temporal direction. They show that the hybrid coder is quite efficient and does as well as a 3-D transform coder. The
hybrid coding system can be made temporally adaptive by periodically optimizing the prediction and bit assignment procedure. Although this adaptation results in a significant improvement in performance when compared with non-adaptive implementations, it increases coder complexity greatly.

Recently, a multiframe image motion prediction technique has been considered by several researchers. Jain and Jain [15] have investigated a motion estimation algorithm, which is based on a 2-D search procedure, and the algorithm is applied to the interframe hybrid coding technique. Due to the use of motion prediction, significant improvements in interframe variance reduction and bit rates are reported. This coding technique can be classified as adaptive hybrid coding. Mathematical analysis and simulation results for the various fixed and adaptive prediction methods for hybrid predictive/transform coding are given by S. Ericsson [26].

1.3 Thesis Objectives and Organizations

The aim of this work is to investigate an efficient hybrid coding technique for transmitting multiframe moving image data by using both motion compensated interframe coding and adaptive transform coding. We start our research by analyzing the temporal characteristics of interframe motion in Chapter II. There, with the understanding of the statistical properties of the interframe motion, we propose
a motion vector estimation algorithm, which is based on block by block operation, for efficient interframe motion prediction. The experimental results for evaluating the performance of the proposed algorithm and the effect of motion compensated prediction are also described in Chapter II.

In Chapter III, we discuss statistical properties of the 2-D unitary transform coefficients and describe an adaptive transform coding technique based on the chain encoding of the transform coefficients. In this adaptive transform coding scheme we follow the similar procedure of W. Chen et al. [12], except for employing the chain coding algorithm of J. A. Saghri et al. [13] for encoding addressing information of the coefficients transmitted. Variable rate Huffman code is used to code the magnitude of the transform coefficients. For encoding the chain boundaries, which represent the addressing information of the selected coefficients to be transmitted, the instantaneous variable rate code is used.

To realize an efficient hybrid coding technique, the motion compensated interframe prediction scheme of Chapter II and adaptive transform coding scheme of Chapter III are combined in Chapter IV. Conceptually, two possible combinations of interframe and 2-D transform coding, i.e., (1) Interframe Transform and (2) Transform Interframe, can be realized for coding of multiframe image data. The theoretical study described by S. Ericsson [26] shows that
the performances of both (1) and (2) are equivalent. However, since motion estimation for interframe motion prediction is performed on the space domain of the image data, we used the hybrid coding scheme of (1) for our adaptive hybrid coding. The experimental results of the proposed coding technique are presented in Section 4.3 of Chapter IV. Here, the transform adaptive coding technique of Chapter III is also simulated in order to compare its performance with the proposed technique.

In Chapter V, the important results presented in this thesis are summarized, and conclusions are also given.

For the simulations of both adaptive hybrid and transform adaptive coding techniques a Hadamard transform is used because of its simplicity in implementation and high speed in computation. All the simulations are performed by using real image data given in the preceding Section. A VAX 11/750 computer system and a VICOM Image Processing System of the Department of Electrical and Computer Engineering at Ohio University are used for processing of the image data and displaying original or reconstructed image data. A brief description for the VICOM System is given in Appendix A and a VAX/VMS FORTRAN subroutine called RWIMAGE, which can convert VICOM format image data to readable real data format and vice versa, is also given in Appendix A. 2-D Hadamard transform representation, its statistical properties and the flow diagram of the fast 1-D ordered Hadamard transform algorithm are given in Appendix B. The simulation program
for the adaptive hybrid coding is presented in Appendix C.

Throughout this thesis, it is assumed that the original image data are the intensity samples of an analog image and that the channel is noiseless.

1.4 Description of Experimental Data Sets

The image data used for the experiments consist of multiframe video motion images obtained from the GTE Laboratories Incorporated, Waltham, Massachusetts. This data set is written in a line-scan fashion, starting from line 1 through line 256 consecutively, and has a data format of 256 x 256 pixels/frame. The original data for MIKE is shown in Fig. 1-3. A brief description of this data is given below.

MIKE: contains 30 sequential frames (1 sec.) of a motion image of a subject (Mike) against a stationary background, digitized to 256 x 256 pixel/frame, 8 bits/pixel.

Since each pixel is originally digitized to 8 bits/pixel, the intensity range of the original data is represented by an integer number between 0 and 255. The peak-to-peak value of the original image is 255. Eq.(1-3) thus becomes

\[
PMSE = \frac{(MSE)}{255^2} \quad (dB) \quad (1-5)
\]
Throughout this thesis we use the mean-square error (MSE) criterion of (1-2), (1-5), and the signal-to-noise ratio (SNR) of (1-4).
Fig. 1-3. Some Frames of the Original Mike Image Data Set
MOTION COMPENSATED INTERFRAME PREDICTION TECHNIQUES

It is well known that there is high correlation between adjacent pixels in the frame-to-frame time dimension. This strong correlation between consecutive frames results in significant redundancy in the original raw data. The interframe coding technique utilizes this redundancy. In the interframe coding technique the prediction of a pixel is the intensity value of the corresponding pixel in the preceding frame. If the absolute value of the prediction error is larger than a threshold, it is quantized and coded together with the address of the pixel. Otherwise, the value of the pixel in the preceding frame is repeated. A block diagram of the simple interframe coding scheme is shown in Fig. 2-1.

![Block Diagram of Basic Interframe Coding/Decoding Scheme](image)

**Fig. 2-1.** Block Diagram of Basic Interframe Coding/Decoding Scheme.
Let \( U(x, y, K) \) denote the intensity value of the pixel at the location \((x, y)\) in the \(K\)-th frame. The prediction signal \( U_p(x, y, K) \) is formed from the previously reconstructed image signal \( \hat{U}(x, y, K-1) \), which is also available at the receiver. That is,

\[
U_p(x, y, K) = \alpha \hat{U}(x, y, K-1) \tag{2-1}
\]

where \( \alpha \) is simple prediction coefficient. The interframe prediction error \( e(x, y, K) \) can be represented by

\[
e(x, y, K) = U(x, y, K) - U_p(x, y, K) \tag{2-2}
\]

Only the pixel difference, \( e(x, y, K) \), greater than a predetermined threshold, is coded for transmission; i.e.,

\[
\text{if } |e(x, y, K)| > T, \text{ then it is quantized and coded for transmission,}
\]

where \(|\cdot|\) denotes the absolute value, and \(T\) denotes the predetermined threshold.

At the receiver, a pixel is reconstructed either by repeating the value of the pixel in the preceding frame or by updating the preceding pixel value using the difference signal; i.e.,
if \(|e(x,y,K)| > T,\)

\[\hat{U}^*(x,y,K) = \hat{e}(x,y,K) + U_p^*(x,y,K)\]

otherwise,

\[\hat{U}^*(x,y,K) = U_p^*(x,y,K-1)\]

where \(U_p^*(x,y,K) = \alpha \hat{U}(x,y,K-1)\), and * represents the noise introduced by channel.

The reconstruction image quality for low bit rate interframe coding is very good for relatively still motion images. However, if the motion in the sequence of image data increases actively between frames, the prediction error increases rapidly. That is, when the interframe correlation of the moving images is decreased, the coding efficiency of the interframe coding system applied to moving images is decreased considerably. Thus, the pixel on the same position in the previous frame is no longer a good sample for motion prediction. If we know the trace information of the objects' motion in the sequence of multiframe images and we can utilize the knowledge of the displacement of pixels between frames, this problem can be eliminated and more bandwidth compression can be achieved by minimizing the prediction error signal. Utilization of the knowledge of displacement of pixels in a sequence of motion images is called motion compensation. In this chapter, we first analyze the temporal characteristics of the interframe motion and propose a new motion vector estimation method.
2.1. Temporal Characteristics of Interframe Motion

Let $U(x,y,t)$ denote the intensity of $(x,y)$ coordinates of an image at time $t$ and let it be a real, stationary random variable. Without loss of generality, it can be assumed that images have zero mean and unity variance. For estimating the temporal characteristics of interframe motion, we use the same method of Jain and Jain [15]. Let each image be a sample of a 2-D homogeneous stationary random process. The covariance is given by

$$E[U(m,n,t)U(x+m,y+n,t)] = \sigma^2 R(|x|,|y|) \quad (2-3)$$

where $\sigma^2$ is the variance of $U(x,y,t)$, $E[\cdot]$ denotes the expectation, $|\cdot|$ denotes the absolute value, and $R$ is the image autocorrelation function.

Let $(x+dx,y+dy,t+dt)$ be the new location of the pixel at time $t+dt$. Then the trajectory of the motion is given by

$$U(x,y,t) = U(x+dx,y+dy,t+dt)$$

$$= \text{Constant} \quad (2-4)$$

The observed value of $U(x,y,t)$ can be represented as

$$V(x,y,t) = U(x,y,t) + N(x,y,t) \quad (2-5)$$
where \( V(x, y, t) \) denotes the observed value, and \( N(x, y, t) \) is the observation noise, which is assumed to be white and independent of \( U \). Let \( N(x, y, t) \) have zero mean and variance \( \sigma_n^2 \). Let \( \hat{d}x \) and \( \hat{d}y \) be the estimates of \( dx \) and \( dy \), respectively, and \( dx' \) and \( dy' \) be the motion estimation error, i.e.

\[
\begin{align*}
    dx' &= dx - \hat{d}x \\
    dy' &= dy - \hat{d}y
\end{align*}
\]  

(2-6)

The motion compensated interframe estimate \( \hat{V} \) is

\[
\hat{V}(x+\hat{d}x, y+\hat{d}y, t+dt) = V(x, y, t) \]  

(2-7)

The temporal correlation after motion compensation is given by

\[
\begin{align*}
    \varphi &= \frac{E[\hat{V}(x+\hat{d}x, y+\hat{d}y, t+dt)\hat{V}(x+\hat{d}x, y+\hat{d}y, t+dt)]}{E[V^2(x+\hat{d}x, y+\hat{d}y, t+dt)]} \\
    &= \frac{E[V(x, y, t)\hat{V}(x+\hat{d}x, y+\hat{d}y, t+dt)]}{E[(U(x+\hat{d}x, y+\hat{d}y, t+dt)+N(x+\hat{d}x, y+\hat{d}y, t+dt))^2]} \\
    &= \frac{1}{\sigma^2 \cdot \sigma_n^2} E[U(x, y, t)U(x+\hat{d}x, y+\hat{d}y, t+dt)] \\
\end{align*}
\]  

(2-8)

Using (2-4) and (2-6) and assuming \( \hat{d}x \) and \( \hat{d}y \) to be independent random variables, (2-8) becomes
Therefore, the temporal correlation, $\phi$, can be obtained from the distribution of $dx'$ and $dy'$. This temporal correlation coefficient will be used for the motion compensated interframe prediction coefficient.

The motion compensated interframe variance is defined as

$$\beta = E[\{V(x+\hat{dx},y+\hat{dy},t+dt)-\hat{V}(x+\hat{dx},y+\hat{dy},t+dt)\}^2]$$

$$= E[V^2(x+\hat{dx},y+\hat{dy},t+dt)+\hat{V}^2(x+\hat{dx},y+\hat{dy},t+dt)$$

$$- 2V(x+\hat{dx},y+\hat{dy},t+dt)\hat{V}(x+\hat{dx},y+\hat{dy},t+dt)]$$

$$= E[V^2(x+\hat{dx},y+\hat{dy},t+dt)] + E[\hat{V}^2(x+\hat{dx},y+\hat{dy},t+dt)]$$

$$- 2E[V(x+\hat{dx},y+\hat{dy},t+dt)\hat{V}(x,y,t)]$$

$$= 2(\sigma^2 + \frac{\sigma_n^2}{\sigma}) - 2\sigma^2 R(|dx'|,|dy'|) \quad (2-10)$$

Using (2-9) in (2-10), the result is
If we assume that no observation noise is present, (2-9) and (2-11) then become

\[
\theta = R(|dx'|, |dy'|) \tag{2-12}
\]

\[
\beta = 2 \sigma^2 (1 - \theta) \tag{2-13}
\]

In the absence of motion compensation, (2-12) and (2-13) can be represented as

\[
\theta' = R(|dx|, |dy|) \tag{2-14}
\]

\[
\beta' = 2^2 \sigma (1 - \theta') \tag{2-15}
\]

where \(\theta'\) and \(\beta'\) denote the no-motion compensated interframe correlation coefficient and the variance, respectively.

In our simulation of image coding and future analysis we assume that images are noise-free; i.e.,

\[
\sigma_n^2 = 0,
\]

and use the approximation of (2-12) to calculate the temporal correlation coefficient, \(\theta\), which is used as the
prediction coefficient. For the optimum mean square estimate of \( U_p(x+dx, y+dy, t+dt) \), define the motion compensated predictor as

\[
U_p(x+dx, y+dy, t+dt) = \Theta U(x, y, t) \tag{2-16}
\]

where \( \Theta \) is the temporal correlation coefficient of (2-12).

In this case, the variance of the motion compensated prediction error is given by

\[
\Omega = E[(U(x+dx, y+dy, t+dt) - U_p(x+dx, y+dy, t+dt))^2]
\]

\[
= E[(U(x+dx, y+dy, t+dt) - \Theta U(x, y, t))^2]
\]

\[
= E[U^2(x+dx, y+dy, t+dt) + \Theta^2 U^2(x, y, t)
- 2\Theta U(x+dx, y+dy, t+dt)U(x, y, t)]
\]

\[
= E[U^2(x+dx, y+dy, t+dt)] + \Theta^2 E[U^2(x, y, t)]
- 2\Theta E[U(x+dx, y+dy, t+dt)U(x, y, t)]
\]

\[
= \Theta^2 + \Theta^2 \sigma^2 - 2\Theta \sigma^2
\]

\[
= \sigma^2 (1 - \Theta^2) \tag{2-17}
\]

Since \( \Theta \) is a correlation coefficient, \( |\Theta| \) is slightly less than or equal to 1. By comparing (2-17) with (2-13), one can conclude

\[
\Omega \leq \beta \tag{2-18}
\]
That is, the motion compensated interframe variance of (2-13) can be reduced by using the temporal correlation coefficient. This indicates that the motion compensated predictor of (2-16) will further improve the performance of the interframe prediction scheme, if $\theta$ is not very close to 1. The temporal correlation coefficient can be calculated, from (2-12), by assuming both the correlation model and the distribution of the motion estimation error of (2-6).
2.2 Interframe Motion Estimation

Let $U_R(x+i, y+j)$ be the position in the previous frame of $U(x,y)$ in the present frame. If we can obtain displacement $(i, j)$, the interframe prediction error between $U(x,y)$ and $U_R(x+i, y+j)$ gives a smaller difference than that between $U(x,y)$ and $U_R(x,y)$. If the displacement can be estimated exactly, it is possible to make a precise prediction.

Motion estimation techniques to improve the prediction efficiency can be broadly categorized as a pel recursive algorithm as described by A. N. Netravali [14] or a block matching algorithm described by Jain and Jain[15], and K. R. Rao and S. Kappagantula [21]. The former, although it can estimate motion better than the latter, is computationally much more complex. The block matching algorithm is much simpler than the pel recursive algorithm and gives very good performance under most circumstances for interframe image motion. In order to estimate the motion on a block-by-block basis, an image is divided into fixed size, small sub-blocks, and each of these sub-blocks is then assumed to undergo independent translation. If these blocks are small enough, the motion of the image objects can be approximated by piecewise linear translation of these sub-blocks.

Let $U$ be an $M \times M$ size sub-block in the present frame and $U_R$ be an $(M+2P) \times (M+2P)$ size sub-block in the previous frame, centered at the same spatial location as $U$. 
where $P$ is the maximum displacement allowed for one frame interval. The geometric configuration for both present and previous sub-block is shown in Fig.2-2.

Let us define a distortion function, $D(i,j)$, which represents the mismatch between $U$ and $U_R$, as

$$D(i,j) = \frac{1}{M^2} \sum_{m=1}^{M} \sum_{n=1}^{M} \{U(m,n) - U_R(m+i,n+j)\}^2$$

(2-19)

$$-P \leq i, j \leq P$$

The direction of minimum distortion is given by the displacement vector $(i,j)$, such that $D(i,j)$ is minimum. However, this simple matching technique requires a vast amount of computation; i.e., for the searching window of size $(M+2P) \times (M+2P)$, to find the direction of the minimum mean square distortion, the number of computations is given by $(2P+1) \times (2P+1)$. Therefore, efficient search procedures which will reduce the number of computations should be used. Jain and Jain [15] considered the problem of complexity of the motion prediction and compensation algorithm, especially with regard to the searching procedure for estimating the interframe displacement of a small block. In their search algorithm, the 2-D logarithmic search is performed along a virtual direction of minimum distortion (called DMD) on the data within the search window of the previous frame. Each step consists of searching five positions, which include the center and the four boundaries.
of the area along the coordinates passing through the center. First, the initial five locations are searched, and the direction of minimum distortion (DMD) is then determined through the computation of the distortion function (D) for each of five locations. This procedure continues until the search area is reduced to $3 \times 3$ size, and finally all nine locations of $3 \times 3$ grid are tested to find the true DMD. Thus, the total search requirement is reduced to approximately $15\%$ of the maximum possible number of searches. This logarithmic 2-D search procedure for $P=5$ is illustrated in Fig. 2-3.
The basic assumption made is that the distortion measure is a monotonically increasing function of distance from the location of the true minimum. That is, assuming

$$D_0(k,l) = \text{MIN}_{i,j}\{D(i,j)\},$$

then for \( m = i - k, n = j - l \),

$$D_1(|m|,|n|) = D(i,j) - D_0(k,l), \quad m \geq 0, n \geq 0$$

$$D_2(|m|,|n|) = D(i,j) - D_0(k,l), \quad m \geq 0, n \leq 0$$

$$D_3(|m|,|n|) = D(i,j) - D_0(k,l), \quad m \leq 0, n \geq 0$$

$$D_4(|m|,|n|) = D(i,j) - D_0(k,l), \quad m \leq 0, n \leq 0$$

These functions \( D_z(|m|,|n|) \), where \( 1 \leq z \leq 4 \), are non-decreasing functions of \(|m|\) and \(|n|\). That is, for \( 1 \leq z \leq 4 \),

$$D_z(|m|,|n|) \leq D_z(|m'|,|n'|) \quad (2-20)$$

if \(|m| \leq |m'|\) and \(|n| \leq |n'|\),

where \(|\cdot|\) denotes the absolute value. Eq.(2-20) confirms that the image covariance function is a decreasing function moving away from the DMD along any direction in each of four quadrants. In this technique, the search is made over horizontal and vertical displacements of up to \pm 5\ pixels.

To simplify the motion vector estimation technique, a technique called One at a Time Search (OTS) is suggested by S. Kappagantula and Rao [21]. This algorithm uses the NMSE.
criterion as the measure of the distortion function. In the initialization of this search algorithm, \( D(0,0) \), which represents the NMSE between the corresponding sub-blocks of adjacent frames, is computed and then compared with predetermined threshold \( (TH) \). If \( D(0,0) < TH \), then the block is classified as a stationary block, and the search terminates there. Otherwise, the first four search locations are searched, and the minimum from these searches is then compared with \( D(0,0) \). If the minimum \( > D(0,0) \), the search distance is reduced to half, and this step is repeated. If not, the minimum is compared with \( TH \). If the minimum \( < TH \), the search ends there. If not, the next two locations are searched. This algorithm uses the fixed value of \( P=7 \). Thus, the maximum search locations are 18 in this algorithm. Considerable reduction in searching time and the number of searches required is reported in [21].

With the same basic assumption as Jain and Jain [15], we introduce a 2-D directional search algorithm. The search is accomplished by progressively reducing the search distance by half or less. In the initialization step the same comparison as that of S. Kappagantula and Rao[21] is performed. If \( D(0,0) < TH \), the search ends. If not, a 2-D directional search is recursively performed by reducing the search distance at the end of each step, which consists of three search directions: horizontal, vertical, and diagonal. In each step, the horizontal and the vertical minimums are first calculated, and the minimum between
these two is then compared with $D(0,0)$. If the minimum is greater than $D(0,0)$, the search distance is reduced, and this step is repeated. If not, the diagonal direction is searched, and the minimum from the previous searches is compared with the distortion function of the diagonal direction. The location of the diagonal search direction is
found from the positions of both the horizontal and vertical minimums. The search terminates whenever the value of the distortion function is less than the predetermined threshold or the search distance reduces to 1. Since only one diagonal position is required to be searched, the maximum number of searches for the 2-D directional search algorithm is 15 for \( P=7 \). A detailed description of the algorithm is given below.
2-D Directional Search Procedure

The K-th sub-block in the present frame is identified by its upper left hand corner pixel (x,y).

Define:

ISD = Initial Search Distance
S = Search Distance
TH = Predetermined Threshold

(0) (Initialization)

Set: S = ISD
i = 0 and j = 0

The distortion function D between a sub-block in the present frame and the corresponding sub-block in the previous frame is compared with TH.

If D(0,0) < TH, the block is classified as an unchanged block. Go to (1).

Otherwise, go to (1).

(1) (Horizontal Search)

The positions to be searched are the locations (x,y-S) and (x,y+S). Find the horizontal minimum HM(k,l). (For instance, D(0,-S) < D(0,S), HM(k,l) = D(0,-S) and k=0, l=-S). Go to (2).
(2) (Vertical search)

Set: \( m = 0 \) and \( n = 0 \)

The positions to be searched are the locations \((x-S,y)\) and \((x+S,y)\). Find the vertical minimum \( VM(k',l') \) and compare it with \( HM(k,l) \).

If \( VM(k',l') < HM(k,l) \), \( DM(m,n) = VM(k',l') \)

\[ m = k' \text{ and } n = l' \]

If not, \( DM(m,n) = HM(k,l) \), \( m = k \), and \( n = l \).

After \( DM(m,n) \) is found, it is compared with \( D(i,j) \).

If \( DM(m,n) > D(i,j) \), set \( m = 0 \) and \( n = 0 \).

Go to (4).

If not, compare \( DM(m,n) \) with \( TH \).

If \( DM(m,n) < TH \), set \( D(i,j) = DM(m,n) \), and \( S = 1 \).

Go to (4).

Otherwise, go to (3).

(3) (Diagonal Search)

The next position of search can be found from the location of both \( HM \) and \( VM \). (For example, if \((x,y+S)\) is the location of \( HM \) and \((x+S,y)\) is the location of \( VM \), the diagonal search direction is \((x+S,y+S)\). In this case the value from the diagonal search is given by \( DS(m',n') \), where \( m' = S \) and \( n' = S \).)
If $DS(m',n') > DM(m,n)$, set $D(i,j) = DM(m,n)$
and go to (4).

If not, set $D(i,j) = DS(m',n')$ and $m = m'$, $n = n'$.

Compare $DS(m',n')$ with $TH$.

If $DS(m',n') < TH$, set $S = 1$.

Otherwise, set $D(m,n) = D(i',j')$.

Go to (4).

(4) The search distance is recursively replaced by
half or less.

Set : $i = i + m$ and $j = j + n$.

If $S = 1$, go to (5).

Otherwise, set : $S = \text{INT}\left[\frac{S}{2}\right]$,
where $\text{INT}[*]$ denotes the lower integer
truncation function.

Go to (1).

(5) Set Minimum Distortion $= D(i,j)$
Displacement Vector $= (i,j)$.

The search procedure terminates.

The direction of minimum mean square distortion is
given by the displacement vector $(i,j)$. The search procedure
for ISD $= 4$ is graphically depicted in Fig. 2-4.
Fig. 2-4. The 2-D Directional Search Procedure. HM, VM, and DS denote the horizontal, vertical, and diagonal search directions for a given search step, respectively. The subscript represents the search step. The horizontal and vertical minimums for each step are given by ▲. The optimum search direction for a given search step is represented by ★. ★ denotes the position of the minimum distortion for this search. In this case, the motion displacement vector \((i,j)\) is \((-7,5)\).
2.3 Results of Motion Estimation

In this section the 2-D directional search algorithm, presented in the previous section, was applied to the Mike sequences. The direction of the minimum mean square distortion was estimated, using the MSE of (2-19), to find the motion displacement vector between corresponding sub-blocks in the present and previous frames. While the interframe motion estimation algorithm gives better performance on a small block, it increases the overhead that has to be sent. Thus, the sub-block size has to be chosen as a compromise between the accuracy of the piecewise linear translational approximation of the interframe motion and the cost of transmitting motion displacement vectors. It has been reported by Jain and Jain [15] and S. Ericsson [26] that the sub-block size of 16 x 16 can be a good compromise.

![Block Diagram of Motion Compensated Interframe Coding/Decoding Scheme.](image)

**Fig. 2-5.** Block Diagram of Motion Compensated Interframe Coding/Decoding Scheme.
For our simulations, the sub-block size of 16x16 was used. In the OTS of S. Kappagantula and Rao [21], a maximum pixel displacement of $P = 7$ is used for both horizontal and vertical axes, and the threshold value of 4 is also used for additional saving in the number of computations required for the search. Compared with the results of Jain and Jain [15], the improved performance is demonstrated in [21]. A maximum displacement of $P = 7$ for both horizontal and vertical axes is adopted for our simulations. We assumed the initial search distance of ISD = 4 pixel/line in the one frame interval. In this case the maximum displacement of a pixel for one frame interval is 7 pixel/line. To evaluate the performance of the motion estimation algorithm of Section 2.2 and to gather the statistics, the motion compensated interframe coding scheme of Fig.2-5 was used without quantization. To evaluate the interframe motion the threshold value used by S. Kappagantula and Rao [21] was used.

After the motion displacement vector for each sub-block has been found, motion compensation is performed on the reference frame, which is defined as an adjacent frame relative to which motion measurement is done. If $(i,j)$ is the motion displacement vector for the sub-block, the motion compensated estimate $U_p(x,y)$, where $(x,y)$ represents the spatial coordinates of an image, is given by

$$U_p(x,y) = \hat{U}_R(x+i,y+j)$$ (2-22)
where $U_R$ denotes the reference frame, and $\theta$ is the interframe prediction coefficient defined in Section 2.1. For multiframe image data, the motion compensated interframe variance is defined as $D(i,j)$, and $D(0,0)$ is called the interframe variance without motion compensation. The results of motion measurement are shown in Table 2-1, Table 2-2, and Table 2-3. (A) and (B) of Table 2-1 and Table 2-2 show the interframe variances without and with motion compensation for each sub-block, respectively. There is a wide range of variation among the sub-blocks in (A). Since Mike sequences consist of only one moving object surrounded by the stationary background, the large magnitude interframe variances in the central portion are due to the object motion displacement, and the low magnitude interframe variances in the surrounding sub-blocks are due to the stationary background. After motion compensation is done in (B), the variation smoothes out, and the large magnitude interframe variances are lowered considerably. The motion displacement vectors for each sub-block of size $16 \times 16$ are shown in Table 2-3. Through the computer simulation of the 2-D directional search algorithm, it is found that the average number of searches is in the range of 5-7 locations for our experimental data set given in Section 1.4. This method reduces the computational requirements by approximately 60% as compared with 13-21 search of the 2-D logarithmic search algorithm described by Jain and Jain[15].
| 2.6 | 2.0 | 1.5 | 495.0 | 86.1 | 95.8 | 78.3 | 37.4 | 37.1 | 20.0 | 28.9 | 112.9 | 751.7 | 147.5 | 1.7 | 1.9 |
| 1.6 | 1.7 | 6.4 | 385.4 | 88.1 | 90.6 | 29.4 | 27.4 | 39.6 | 23.5 | 74.1 | 182.1 | 193.7 | 744.3 | 3.0 | 2.3 |
| 2.6 | 2.8 | 69.5 | 238.5 | 107.1 | 114.1 | 39.5 | 56.8 | 37.3 | 69.1 | 75.3 | 97.3 | 236.2 | 670.2 | 3.0 | 2.4 |
| 2.7 | 2.7 | 248.1 | 220.5 | 217.7 | 42.7 | 19.0 | 16.3 | 31.0 | 44.8 | 63.7 | 190.3 | 166.7 | 641.3 | 2.2 | 2.2 |
| 2.1 | 2.1 | 521.8 | 271.8 | 373.4 | 8.5 | 8.4 | 6.4 | 6.2 | 86.8 | 77.8 | 141.4 | 186.8 | 791.9 | 2.2 | 2.4 |
| 1.8 | 2.0 | 36.3 | 904.8 | 163.0 | 23.0 | 42.3 | 16.8 | 7.9 | 65.6 | 148.6 | 106.5 | 32.2 | 618.0 | 1.8 | 1.6 |
| 2.1 | 2.2 | 4.8 | 761.2 | 491.9 | 5.0 | 4.5 | 2.1 | 5.3 | 21.9 | 9.5 | 94.6 | 749.6 | 509.5 | 2.1 | 1.7 |
| 2.4 | 2.1 | 1.3 | 128.2 | 287.6 | 6.2 | 29.7 | 18.6 | 15.3 | 16.6 | 6.8 | 162.3 | 776.7 | 3.6 | 1.8 | 2.4 |
| 1.7 | 0.8 | 0.5 | 113.7 | 505.4 | 26.3 | 104.7 | 39.3 | 20.3 | 92.3 | 29.2 | 96.5 | 320.2 | 2.9 | 1.3 | 1.8 |
| 2.2 | 1.3 | 0.7 | 4.4 | 469.2 | 57.6 | 329.9 | 962.8 | 943.0 | 221.9 | 23.2 | 33.1 | 208.5 | 2.6 | 1.0 | 1.9 |
| 2.3 | 1.8 | 0.7 | 0.9 | 354.9 | 227.3 | 81.8 | 527.9 | 456.5 | 97.2 | 11.2 | 331.6 | 347.9 | 2.3 | 2.0 | 2.4 |
| 1.9 | 1.8 | 0.6 | 0.6 | 1.7 | 123.5 | 44.7 | 39.3 | 17.5 | 18.1 | 5.4 | 159.7 | 7.3 | 1.9 | 1.6 | 1.5 |
| 1.6 | 1.9 | 2.0 | 1.7 | 2.2 | 85.7 | 37.5 | 37.0 | 56.3 | 20.5 | 4.9 | 47.2 | 20.9 | 2.5 | 2.0 | 1.7 |
| 3.2 | 2.6 | 2.8 | 3.5 | 55.3 | 44.6 | 5.2 | 27.3 | 15.5 | 6.8 | 4.8 | 44.8 | 11.3 | 23.6 | 44.0 | 12.0 |
| 2.6 | 2.9 | 10.7 | 39.0 | 10.2 | 8.0 | 14.6 | 4.4 | 4.7 | 4.9 | 11.9 | 70.6 | 7.8 | 5.3 | 5.0 | 45.7 |
| 2.7 | 28.5 | 20.8 | 6.7 | 6.5 | 5.6 | 5.8 | 19.2 | 10.2 | 22.3 | 39.4 | 6.3 | 13.1 | 6.1 | 5.6 | 6.7 |

(A) NO MOTION COMPENSATED INTERFRAME VARIANCES (NIFV) FOR EACH SUBBLOCK OF MIKE13 (AVERAGE NIFV = 97.236)

(B) MOTION COMPENSATED INTERFRAME VARIANCES (MCIFV) FOR EACH SUBBLOCK OF MIKE13 (AVERAGE MCIFV = 25.579)

Table 2-1. THE EFFECTS OF MOTION COMPENSATION IN TERMS OF INTERFRAME VARIANCE REDUCTION FOR EACH SUBBLOCK OF SIZE 16 X 16. MIKE12 WAS USED AS REFERENCE FRAME.
<table>
<thead>
<tr>
<th></th>
<th>1.8</th>
<th>1.9</th>
<th>1.4</th>
<th>1.4</th>
<th>5.1</th>
<th>635.6</th>
<th>712.0</th>
<th>97.8</th>
<th>117.5</th>
<th>68.4</th>
<th>63.4</th>
<th>1063.2</th>
<th>395.6</th>
<th>2.0</th>
<th>1.7</th>
<th>1.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>1.8</td>
<td>2.4</td>
<td>2.8</td>
<td>1156.6</td>
<td>393.0</td>
<td>211.8</td>
<td>197.8</td>
<td>78.5</td>
<td>95.7</td>
<td>43.3</td>
<td>52.3</td>
<td>1865.4</td>
<td>2.0</td>
<td>1.6</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>2.6</td>
<td>2.8</td>
<td>46.0</td>
<td>1494.6</td>
<td>124.4</td>
<td>185.4</td>
<td>117.9</td>
<td>46.4</td>
<td>90.8</td>
<td>39.3</td>
<td>102.3</td>
<td>1695.4</td>
<td>522.6</td>
<td>3.1</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>2.9</td>
<td>2.8</td>
<td>1142.6</td>
<td>312.7</td>
<td>85.1</td>
<td>155.2</td>
<td>45.0</td>
<td>61.4</td>
<td>64.5</td>
<td>45.5</td>
<td>138.3</td>
<td>237.8</td>
<td>1710.9</td>
<td>3.4</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>1.9</td>
<td>2.4</td>
<td>1.9</td>
<td>1072.8</td>
<td>122.5</td>
<td>128.0</td>
<td>46.4</td>
<td>48.7</td>
<td>75.5</td>
<td>62.7</td>
<td>103.4</td>
<td>190.0</td>
<td>314.0</td>
<td>2020.9</td>
<td>3.8</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>1.6</td>
<td>73.2</td>
<td>875.7</td>
<td>135.4</td>
<td>204.8</td>
<td>66.0</td>
<td>76.0</td>
<td>106.6</td>
<td>132.6</td>
<td>215.0</td>
<td>121.1</td>
<td>326.6</td>
<td>1916.0</td>
<td>4.4</td>
<td>2.2</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>2.3</td>
<td>205.4</td>
<td>1297.3</td>
<td>424.1</td>
<td>271.6</td>
<td>21.4</td>
<td>27.2</td>
<td>50.7</td>
<td>120.6</td>
<td>211.3</td>
<td>284.3</td>
<td>298.3</td>
<td>1636.5</td>
<td>4.5</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>1.5</td>
<td>1.1</td>
<td>91.4</td>
<td>2580.7</td>
<td>460.7</td>
<td>350.6</td>
<td>131.3</td>
<td>87.5</td>
<td>13.7</td>
<td>223.5</td>
<td>431.6</td>
<td>218.4</td>
<td>259.9</td>
<td>2128.3</td>
<td>2.5</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td>1.6</td>
<td>1.8</td>
<td>5.8</td>
<td>2313.7</td>
<td>642.9</td>
<td>176.0</td>
<td>1377.0</td>
<td>148.6</td>
<td>46.6</td>
<td>404.4</td>
<td>741.8</td>
<td>122.7</td>
<td>34.8</td>
<td>1705.0</td>
<td>5.8</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>2.0</td>
<td>2.2</td>
<td>799.8</td>
<td>1980.7</td>
<td>11.6</td>
<td>29.9</td>
<td>28.1</td>
<td>48.3</td>
<td>378.3</td>
<td>296.0</td>
<td>80.7</td>
<td>792.1</td>
<td>2085.3</td>
<td>2.1</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>1.8</td>
<td>2.0</td>
<td>1.4</td>
<td>0.5</td>
<td>153.8</td>
<td>584.8</td>
<td>12.7</td>
<td>15.4</td>
<td>79.1</td>
<td>67.3</td>
<td>168.2</td>
<td>25.3</td>
<td>186.7</td>
<td>1913.2</td>
<td>258.4</td>
<td>1.5</td>
<td>2.0</td>
</tr>
<tr>
<td>2.1</td>
<td>1.5</td>
<td>0.9</td>
<td>148.8</td>
<td>1925.8</td>
<td>65.2</td>
<td>125.4</td>
<td>149.8</td>
<td>112.6</td>
<td>542.8</td>
<td>42.7</td>
<td>111.7</td>
<td>1112.0</td>
<td>9.0</td>
<td>1.8</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>2.1</td>
<td>1.7</td>
<td>2.1</td>
<td>1445.7</td>
<td>136.6</td>
<td>526.3</td>
<td>723.2</td>
<td>470.6</td>
<td>517.5</td>
<td>71.8</td>
<td>62.2</td>
<td>598.2</td>
<td>6.7</td>
<td>2.5</td>
<td>2.6</td>
<td></td>
</tr>
<tr>
<td>2.8</td>
<td>2.1</td>
<td>1.3</td>
<td>708.7</td>
<td>1233.1</td>
<td>74.6</td>
<td>198.2</td>
<td>231.4</td>
<td>106.4</td>
<td>20.3</td>
<td>360.9</td>
<td>1012.4</td>
<td>3.7</td>
<td>2.5</td>
<td>2.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>2.3</td>
<td>1.6</td>
<td>1.7</td>
<td>2.5</td>
<td>260.1</td>
<td>50.3</td>
<td>25.7</td>
<td>18.0</td>
<td>41.4</td>
<td>13.6</td>
<td>501.4</td>
<td>92.8</td>
<td>2.6</td>
<td>2.2</td>
<td>1.9</td>
<td></td>
</tr>
<tr>
<td>1.7</td>
<td>1.6</td>
<td>2.1</td>
<td>2.0</td>
<td>1.7</td>
<td>54.2</td>
<td>54.5</td>
<td>30.3</td>
<td>21.7</td>
<td>23.8</td>
<td>5.7</td>
<td>46.4</td>
<td>29.5</td>
<td>2.4</td>
<td>2.3</td>
<td>1.3</td>
<td></td>
</tr>
</tbody>
</table>

(A) NO MOTION COMPENSATED INTERFRAME VARIANCES (NIFV) FOR EACH SUBBLOCK OF MIKE16 (AVERAGE NIFV = 256.748)

(B) MOTION COMPENSATED INTERFRAME VARIANCES (MCIFV) FOR EACH SUBBLOCK OF MIKE16 (AVERAGE MCIFV = 27.537)

Table 2-2. THE EFFECTS OF MOTION COMPENSATION IN TERMS OF INTERFRAME VARIANCE REDUCTION FOR EACH SUBBLOCK OF SIZE 16 X 16. MIKE15 WAS USED AS REFERENCE FRAME.
<table>
<thead>
<tr>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
<th>(0, 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
<td>(0, 0)</td>
</tr>
</tbody>
</table>

(A) Displacement Vectors for Motion Compensation of MIKE15 (Average Number of Iterations = 5)

(B) Displacement Vectors for Motion Compensation of MIKE16 (Average Number of Iterations = 7)

Table 2-3. IMAGE MOTION DISPLACEMENT VECTOR FOR EACH SUBBLOCK OF SIZE 16 X 16.

REFERENCE FRAME FOR (A) AND (B) ARE MIKE14 AND MIKE15, RESPECTIVELY.
Although the maximum search requirements in the proposed method is reduced by approximately 18% as compared with the OTS of S. Kappagantula and K. R. Rao [21], almost the same performance was obtained. This is because the proposed method and that of [21] used different picture data for the simulation. In addition, the threshold comparison after the end of each step limits the required number of searches for the motion estimation of each sub-block.

The estimation of the displacement vector based on minimizing the distortion function is an optimization problem. In order to evaluate the accuracy of the 2-D directional search method, the distortion function for all search directions is studied. The maximum displacement of $P = 7$ is assumed. That is, the evaluation of $(2P+1)^2$ or 225 directions for each sub-block was made. Table 2-4 lists the mean square distortions of all directions for the chosen three sub-blocks. It is evident from the values in Table 2-4 that the image covariance function behaves approximately as a convex function. The magnitude of distortion increases away from the true minimum. The motion displacement vector and number of searches by the 2-D directional search method are also indicated in Table 2-4. From the study of the distortion function for various sub-blocks, it can be inferred that the 2-D directional search algorithm could well be applied for various motion image sequences.

The effect of motion compensation in the interframe
Table 2-4. THE MEAN SQUARE ERROR DISTORTIONS FOR ALL POSSIBLE SEARCH DIRECTIONS SUBBLOCK SIZE OF 16 x 16 AND P = 7 WERE USED.
prediction is shown by the improvement in terms of the signal to noise ratio (SNR), in Table 2-5. The interframe variances for one frame, both with and without motion compensation, can be computed by averaging them over all sub-blocks. It is shown in Table 2-5 that the improvement for Mike14 is very small, while the improvement for Mike16 is large. This is because the improvement depends upon the range of the variations between consecutive frames. The results in Table 2-5 indicate that the error motion prediction is almost identically distributed over different frames even though the motion itself is not. Thus, by assuming a constant interframe variance, we can use the fixed value of temporal correlation for all the pixels. Eq. (1-4) was used for the calculation of SNR. The error images for both cases of with and without motion compensation are shown in Fig.2-6.

<table>
<thead>
<tr>
<th>FRAME NO.</th>
<th>SNR(dB)</th>
<th>NO MOTION COMPENSATION</th>
<th>MOTION COMPENSATION</th>
<th>IMPROVEMENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIKE13</td>
<td>28.25</td>
<td>34.05</td>
<td></td>
<td>5.8</td>
</tr>
<tr>
<td>MIKE14</td>
<td>34.03</td>
<td>35.57</td>
<td></td>
<td>1.54</td>
</tr>
<tr>
<td>MIKE15</td>
<td>30.12</td>
<td>34.89</td>
<td></td>
<td>4.77</td>
</tr>
<tr>
<td>MIKE16</td>
<td>24.67</td>
<td>33.73</td>
<td></td>
<td>9.06</td>
</tr>
</tbody>
</table>

Table 2-5. The Effect of Motion Compensation in terms of Interframe Variance Reduction in Interframe Coding
Fig. 2-6. The Effect of Motion Compensation in Interframe Difference between Present Frame and Previous Frame. (i) represents the prediction error image between present frame and previous frame, and (ii) shows the motion compensated prediction error image. For interframe motion estimation & compensation of the image for (ii), the algorithm described in section 2.2 is used with $T = 16.0$ and $\sigma = 1.0$.

(Continued)
(c) Interframe Difference between Mikel4 and Mikel5

(ii)

(ii)

Fig. 10. The effect of Motion Compensation in Interframe Difference between Present Frame and Previous Frame. (i) represents the prediction error image between present frame and previous frame, and (ii) shows the motion compensated prediction error image. For interframe motion estimation & compensation of the image for (ii), the algorithm described in section ... is used with $T = 16.0$ and $\sigma = 1.0$. 
CHAPTER III

2-D TRANSFORM ADAPTIVE CODING

Transform coding, which has been proven to be an efficient approach to provide data compression in image transmission, is a method for accomplishing some aspects of both statistical and psychovisual coding [10]. The benefits of transform coding derive from the statistical characteristics of the input image data in that they allow us to reproduce the original image with an acceptable degree of fidelity by retaining only significant coefficients.

By using a nonadaptive transform coding technique it is possible to achieve a coding rate of approximately 1.5 bits/pixel with no visible degradation[12]. To achieve a lower coding rate without increasing coding error, the transform coding technique should be adapted to the statistics of the local image data.

In this chapter, we first discuss the statistical properties of the unitary transform coefficients and introduce an adaptive transform coding technique based on the chain coding. The adaptivity of this coding technique is applied in the threshold coding in the transform domain. This coding technique is presented in Section 3.2, and the chain coding algorithm is given in section 3.3.
3.1 Statistical Properties of Unitary Transform Samples

Let \( U(x,y) \) be an intensity sample of an original image array. The 2-D forward and inverse unitary transforms for a block of size \( M \times M \) can be represented by

\[
H(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} U(x,y) \cdot P(x,y,u,v)
\]

\[ \text{u, v = 0, 1, 2, \ldots, M-1}. \]  

\[
U(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{M-1} H(u,v) \cdot P(x,y,u,v)
\]

\[ \text{x, y = 0, 1, 2, \ldots, M-1}. \]  

where \( H(u,v) \) denotes the 2-D unitary transform coefficient, and \( P(x,y,u,v) \) is the 2-D unitary transform kernel. The mean value of the 2-D discrete image array is given by the spatial average of the luminance values of all the pixels within the block.

\[
E[U(x,y)] = \frac{1}{M^2} \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} U(x,y)
\]

\[ = m \]  

And the variance, which is the average value of the square of the difference between the values of an arbitrary image element and the mean value, is given by
Expanding (3-4), the result is,

\[ \sigma^2 = \frac{1}{M^2} \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} [U(x,y) - m]^2 \]  

(3-4)

Expanding (3-4), the result is,

\[ \sigma^2 = \frac{1}{M^2} \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} [U^2(x,y)] - m^2 \]

\[ = E[U^2(x,y)] - m^2 \]  

(3-5)

\( E[U^2(x,y)] \) is considered as the average total energy per pixel and \( m^2 \) is the mean energy per pixel. Thus Eq. (3-5) can be expressed by

\[ E[U^2(x,y)] = m^2 + \sigma^2 \]

or

\[ \text{TOTAL ENERGY} = \text{DC ENERGY} + \text{AC ENERGY}, \]  

(3-6)

where total energy \( = M^2 E[U^2(x,y)] \), DC energy \( = M^2 m^2 \), AC energy \( = M^2 \sigma^2 \), and \( M \) denotes the sub-block size. Since the DC and AC energies in the transform domain equal those of the original image, it is possible to process the image data by setting the whole image mean value to zero, in which case all transform domain AC coefficients remain unchanged but the magnitude of the DC coefficient now becomes zero. That
is, if the intensity value \( U(x,y) \) of the image pixel is subtracted by its mean value given in Eq. (3-3), then

\[
H(0,0) = 0
\]

and all transform AC coefficients will be unaffected. Thus, the variance of the transform coefficients can be calculated by

\[
\sigma^2(u,v) = E[H^2(u,v)], \quad u,v \neq 0.
\]

Since the DC coefficient represents the average brightness of the sub-block and there is only one DC coefficient within the transform sub-block, it is usual to quantize the DC coefficient accurately to avoid the contouring effect in the reconstructed image. Therefore, as far as transform coding is concerned, attention centers upon the AC transform coefficients because the whole gain of the transform coding results from the deletion, or coarse quantization, of the higher order AC coefficients.
3.2 2-D Unitary Transform Adaptive Coding

In this adaptive transform coding scheme, an input digitized image of size \( N \times N \) is first divided into small sub-blocks of size \( M \times M \). The 2-D unitary transform is applied separately and independently to each sub-block. All the transform coefficients, except for the DC coefficient, undergo a threshold process to retain only those coefficients with absolute amplitude above a predetermined threshold. Coefficients less than the threshold are set to zero. Because the whole gain of the transform coding results from the deletion of the low absolute magnitude AC transform coefficients, only the coefficients greater than the threshold are subtracted by the threshold and normalized by a scaling parameter. The quantization process is performed on these normalized coefficients. Then, the chain coding algorithm [13] is applied to the quantized 2-D transform sub-block so as to improve the coding efficiency of the zero coefficients region. Definition of the 8-directional chain code is shown in Fig. 3-1. All zero coefficients are chain coded within the transform block and removed prior to the two-to-one dimensional mapping process of the non-zero coefficients. The scanning of the non-zero coefficients to convert the 2-D data to a 1-D format is performed. Address information of non-zero coefficients is retained in the receiver by 2-D mapping of 1-D data according to the chain code. This adaptive coding scheme is depicted in Fig. 3-2.
Let $U(x,y)$ be a sample of the intensity value of the image array and $H(u,v)$ be an unitary transform coefficient. The 2-D forward and inverse unitary transform for the sub-block of size $M \times M$ is given by (3-1) and (3-2). In the transform domain, the sum of the squares of the AC coefficients for each sub-block is the same as the sum of the AC energies in the sub-block. Thus, the sum of the absolute values of the AC coefficients is also a measure of the activity. Fig. 3-1. Freeman's Chain Code levels of the images. Since we now wish to represent a good approximation to the original image with only a subset of the transform coefficients, we should choose those coefficients with the largest absolute values in order to minimize mean square error. To choose the largest absolute AC coefficients, the 2-D unitary transform coefficients are subjected to the thresholding process.
Fig. 3-2. 2-D Unitary Transform Adaptive Coding/Decoding Scheme
3.2.1 Thresholding and Normalization

All AC coefficients are compared with the predetermined threshold. Only those coefficients which have larger absolute magnitudes than the threshold are retained and then subtracted by the threshold. The AC coefficients below the threshold are set to zero. This process is defined as:

\[
H_T(u,v) = \begin{cases} 
H(u,v) - T, & \text{when } H(u,v) > 0 \\
H(u,v) + T, & \text{when } H(u,v) < 0
\end{cases}
\]

if \( T < |H(u,v)| \),

\[
H_T(u,v) = \begin{cases} 
H(u,v) - T, & \text{when } H(u,v) > 0 \\
H(u,v) + T, & \text{when } H(u,v) < 0
\end{cases}
\]

if \( T \geq |H(u,v)| \),

\[
H_T(u,v) = 0.
\]  

where \( H_T(u,v) \) denotes the 2-D unitary transform coefficient after the thresholding process and \( T \) denotes the properly chosen threshold which controls the achievable average bit rate. Histogram and cumulative distribution of the 2-D ordered Hadamard transform coefficients for MIKE12 and image data is shown in Fig. 3-3. Although the maximum Hadamard transform coefficient value could be as large as \( M \times A \), where \( M \) is the transform sub-block size and \( A \) is maximum intensity value in the original image, it is shown in Fig. 3-3 that over 90% of the coefficients have an absolute magnitude of less than 8. Therefore, the thresholding process of (3-9) will change the major portion of the
<table>
<thead>
<tr>
<th>INTERVAL</th>
<th>DISTRIBUTION OF OCCURANCES</th>
<th>NO. REL FREQ</th>
<th>CUMULATIVE PROBABILITY</th>
<th>PROB.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50I</td>
<td></td>
<td>24576</td>
<td>0.375I</td>
<td>10.3750</td>
</tr>
<tr>
<td>1.50I</td>
<td></td>
<td>14080</td>
<td>0.215I</td>
<td>10.5899</td>
</tr>
<tr>
<td>2.50I</td>
<td></td>
<td>8192</td>
<td>0.125I</td>
<td>10.7148</td>
</tr>
<tr>
<td>3.50I</td>
<td></td>
<td>8186</td>
<td>0.043I</td>
<td>10.7158</td>
</tr>
<tr>
<td>4.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.8125</td>
</tr>
<tr>
<td>5.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.8555</td>
</tr>
<tr>
<td>6.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.8789</td>
</tr>
<tr>
<td>7.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.8945</td>
</tr>
<tr>
<td>8.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9023</td>
</tr>
<tr>
<td>9.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9023</td>
</tr>
<tr>
<td>10.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9180</td>
</tr>
<tr>
<td>11.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9258</td>
</tr>
<tr>
<td>12.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9297</td>
</tr>
<tr>
<td>13.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9336</td>
</tr>
<tr>
<td>14.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9453</td>
</tr>
<tr>
<td>15.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9570</td>
</tr>
<tr>
<td>16.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9570</td>
</tr>
<tr>
<td>17.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9570</td>
</tr>
<tr>
<td>18.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9570</td>
</tr>
<tr>
<td>19.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9570</td>
</tr>
<tr>
<td>20.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9570</td>
</tr>
<tr>
<td>21.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9570</td>
</tr>
<tr>
<td>22.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9648</td>
</tr>
<tr>
<td>23.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>24.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>25.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>26.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>27.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>28.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>29.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>30.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>31.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>32.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>33.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>34.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>35.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>36.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>37.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>38.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>39.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>40.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>41.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>42.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>43.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
<tr>
<td>44.50I</td>
<td></td>
<td>8164</td>
<td>0.006I</td>
<td>10.9727</td>
</tr>
</tbody>
</table>

**Fig. 3-3.** Histogram and Cumulative Density Function of 2-D Ordered Hadamard Transform Coefficients of MIKE12
transform coefficients to zero. Since the 2-D unitary transform contains exactly the same amount of information as the input image data, omitting the small AC transform coefficients thus contributes to image degradation in the reconstructed image at the receiver. The mean square error induced by this thresholding process can be expressed as

\[ \text{ERR}_d = \frac{1}{M^2} \sum_{u=0}^{M-1} \sum_{v=0}^{M-1} E[(H(u,v,K) - \hat{H}(u,v,K))^2] \]  \quad (3-10)

where

\[ \hat{H}(u,v,K) = \frac{1}{M^2} \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} U_T(x,y,K) P(x,y,u,v) \]

and \( U_T(x,y,K) \) is the reconstructed version after the inverse threshold process of the \( H(u,v,K) \). The deletion error is related to the energy packing ability of the particular type of unitary transform used and is independent of other coding processes.

The retained threshold subtracted transform coefficients are normalized by a scaling parameter, i.e.,

\[ H_{TN}(u,v) = \frac{H_T(u,v)}{D} \]  \quad (3-11)

where \( H_{TN}(u,v) \) denotes the transform coefficient after the normalization process and \( D \) represents the suitably chosen normalization parameter. The normalization parameter should
be equal to or greater than unity to introduce meaningful transform coefficients to the coder. Since we use a fixed normalization parameter, the inverse normalization at the receiver can be carried out with no overhead from the coder.

3.2.2 Quantization

The quantization of the normalized coefficients is carried out by using an integer floating point round-off quantizer that performs the floating point to integer conversion. This process is defined as

\[
\hat{H}_{TN}(u,v) = \begin{cases} 
\text{Integer Part of } [H_{TN}(u,v) + 0.5], & H_{TN}(u,v) > 0 \\
\text{Integer Part of } [H_{TN}(u,v) - 0.5], & H_{TN}(u,v) < 0
\end{cases}
\]  

(3-12)

In addition to the effects of the deletion of the small magnitude AC transform coefficients, an additional source of error is introduced in the quantization process. The mean square quantization error can be written as

\[
\text{ERR}_q = \frac{1}{M^2} \sum_{u=0}^{M-1} \sum_{v=0}^{M-1} \left[ H_{TN}(u,v) - \hat{H}_{TN}(u,v) \right]^2 .
\]  

(3-13)
This error can be easily calculated by assuming the probability density function of the transform coefficients. W. Chen and W. K. Pratt[12] shows, by assuming the Laplacian density of the 2-D cosine transform coefficients, that the worst case of this mean square quantization error is \( \frac{M^2}{48} \), where \( M \) is the sub-block size. Since, for \( M = 16 \), this error corresponds to SNR of more than 40.86 dB, the degradation due to this quantization process is relatively insignificant in the reconstructed image at the receiver.

During this quantization process many fractional values will be changed to zero value due to the subtraction by the threshold and the division by the normalization parameter. Only significant coefficients will remain as non-zero coefficients. To retain the addressing information of the remaining coefficients, the chain coding algorithm given in Section 3.3 is applied to the transform sub-block, and 1-D mapping for the chain enclosed coefficients is then performed in a line by line scanning fashion.

3.2.3. Coding for Chain Enclosed Coefficients

Since the frequency of occurrence of the quantizer output values is not equally likely, variable rate coding is more efficient because it requires fewer bits on average to represent the same information. Huffman coding, whose average bit rate is very close to the entropy of the quantizer output, is used for coding the chain enclosed
Table 3-1. Huffman Code Table for the Transform Coefficient Amplitude in Absolute Value. EOB denotes the End of Block code.

<table>
<thead>
<tr>
<th>MESSAGE</th>
<th>CODE WORD</th>
<th>PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>0.3735</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>0.2140</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>0.1245</td>
</tr>
<tr>
<td>3</td>
<td>1101</td>
<td>0.0428</td>
</tr>
<tr>
<td>4</td>
<td>0111</td>
<td>0.0545</td>
</tr>
<tr>
<td>5</td>
<td>1110</td>
<td>0.0428</td>
</tr>
<tr>
<td>6</td>
<td>11001</td>
<td>0.0233</td>
</tr>
<tr>
<td>7</td>
<td>011000</td>
<td>0.0156</td>
</tr>
<tr>
<td>8</td>
<td>0110100</td>
<td>0.0078</td>
</tr>
<tr>
<td>9</td>
<td>011001</td>
<td>0.0156</td>
</tr>
<tr>
<td>10</td>
<td>0110101</td>
<td>0.0078</td>
</tr>
<tr>
<td>11</td>
<td>1100011</td>
<td>0.0042</td>
</tr>
<tr>
<td>12</td>
<td>11000100</td>
<td>0.0036</td>
</tr>
<tr>
<td>13</td>
<td>011011</td>
<td>0.0117</td>
</tr>
<tr>
<td>14</td>
<td>110000</td>
<td>0.0117</td>
</tr>
<tr>
<td>15</td>
<td>1111 + 9 bits</td>
<td>0.0427</td>
</tr>
<tr>
<td>EOB</td>
<td>11000101</td>
<td>0.0039</td>
</tr>
</tbody>
</table>

coefficients. The Huffman coding table can be derived from the Histogram of the transform coefficients. Table 3-1 shows the Huffman coding table used to code the amplitude of the transform coefficients. The EOB code is used to terminate
coding of the block. This coding table is derived from the histogram of Fig. 3-3. The method to construct the Huffman coding table can be found in the references [25] and [32]. The only exception is the DC coefficient.

For the DC coefficient, it represents the average brightness of the sub-block, and the human visual system can easily distinguish the average brightness difference between sub-blocks. Thus, the error in the DC coefficient will cause errors in the reconstructed image at frequencies of maximum subjective visual response. To avoid the contouring effect at sub-block boundaries in the reconstructed image, the DC coefficient is linearly quantized, and a fixed length code word is assigned. The number of bits for the fixed length code can be determined from the maximum possible value of the DC coefficient of the unitary transform used.
3.3 Chain Coding

Since the unitary transform coefficients usually decrease in absolute magnitude with increasing frequency, the limited significant coefficients, which are retained after the quantization process, will be clustered near the DC coefficient, and many other insignificant coefficients will be changed to zero during the preceding processes. This characteristic of the transform sub-block can be efficiently represented by a chain code. All zero coefficients are grouped by a boundary tracing algorithm and removed after coding the chain link. Because our purpose is to reduce the transmission bit rates by using the chain coding of the zero coefficients region, we need only find the approximate boundary between the nonzero and zero regions within the transform block. Thus, to isolate the zero coefficients region, an approximate boundary-tracing algorithm is sufficient. As a result, a few zero terms may be included in the group of the non-zero coefficients.

3.3.1 Boundary Tracing Algorithm

The boundary tracing algorithm presented here consists of three sub-algorithms: an SP algorithm for finding a new starting point, a TR algorithm for tracing the boundaries, and an ASGN algorithm for assigning the new value for the points within the boundary already generated.
We assume that all the paths between the center point and the $N$-neighbors have the same length $L$, which represents the number of points covered by a chain link. The size of $L$, which can be referred to the resolution of the chain, gives a degree of coarseness of the chain coding. The term $N$-neighbors, shown in Fig. 3-4, is defined as $0 \leq N \leq 7$.

Suppose that a buffer array of size $(M + 2L) \times (M + 2L)$ is filled with zeroes. The transform sub-block $H$ of size $M \times M$ is added to the center portion of the buffer array. All non-zero values are replaced by 1. Let $W(m,n)$ be a new buffer array of size $(M + 2L) \times (M + 2L)$. $W(m,n)$ is defined as

$$
W(m,n) = \begin{cases} 
1 & H(u,v) \neq 0 \\
0 & \text{otherwise}
\end{cases} \quad (3-14)
$$

if $L < m \leq M+L$ and $L < n \leq M+L$,

elsewhere, $W(m,n) = 0$.
where $H(u,v)$ is the value of the transform coefficient within the sub-block. The geometric configuration of the buffer $W$ is illustrated in Fig. 3-5. The Boundary tracing algorithm is applied on the buffer $W$ as follows:

Define:

$\text{LP} =$ Chain Code for Present Chain Link

$\text{LN} =$ Chain Code for Next Chain Link

$F(K) =$ Identifier for $K$-th Boundary and Counter for Points covered by $K$-th Boundary

$[\text{XS}(K),\text{YS}(K)] =$ XY-Coordinates of the Starting Point for the $K$-th Boundary.

$[\text{XP}(K),\text{YP}(K)] =$ XY-Coordinates of the Point corresponding to the Head of LP in the $K$-th boundary.

$\text{CH}(I,K) =$ Chain Code of $I$-th Link for the $K$-th Boundary.
Initialization

Set: \( I = 1, \ K = 1, \ LP = 0, \ R = 100000 \) and \( S = 100 \).

* Note: \( R \) and \( S \) are arbitrarily fixed values greater than 10000 and 100, respectively. \( S \) must be less than or equal to \( R + S \).

Step 1: The SP algorithm is applied to find the location of the \( K \)-th starting point (SP) of the \( K \)-th boundary. If \( K \)-th SP is found, go to Step 2. If not, chain coding is completed.

Step 2: Set \( F(K) = R \times K, \ XS(K) = X \) and \( YS(K) = Y \), where \( X \) and \( Y \) are the coordinates of the \( K \)-th SP. Go to Step 3.

Step 3: The TR algorithm is used to trace the \( K \)-th boundary. The TR algorithm traces the outer boundary of the largest connected set of the points of non-zero value and terminates back at the \( K \)-th SP. The chain codes of each \( CH(I,K) \) for new LP of the \( K \)-th boundary are recorded through the TR algorithm. In order to determine the completeness of the chain coding of the \( K \)-th boundary, \( XP(K) \) and \( YP(K) \) are always compared with the \( K \)-th SP.

If \( XP(K) = XS(K) \) and \( YP(K) = YS(K) \), go to Step 4.
If not, \( I = I + 1 \) and continue Step 3.
Step 4: The ASGN algorithm is used to replace the value of each point $W(m,n)$ within the $K$-th boundary to $F(K)$; i.e.,

$$W(m,n) = \begin{cases} F(K) & \text{if } W(m,n) \text{ is within the } K\text{-th boundary;} \\ W(m,n), & \text{otherwise.} \end{cases}$$

Step 5: Set $I = 1$ and $K = K + 1$

Go to Step 1.

This boundary tracing algorithm terminates when all the points of the transform sub-block $H$ are tested by the SP algorithm. A simple block diagram of the chain coding process is illustrated in Fig.3-6.

SP Algorithm

The search for SP starts with the point in the upper left hand corner of the transform sub-block $H$ and proceeds to the right and top-to-bottom to find the location of the new SP for the $K$-th boundary. Each coefficient within the transform sub-block $H$ is evaluated to determine whether or not it is a non-zero coefficient.

If $W(m,n) = 1$,

the upper left neighbor of the point under test is SP.
Fig. 3-6. Flow Graph of Chain Coding Procedure
The SP algorithm terminates when the search for SP is reached to the point in the lower right corner of the transform sub-block.

**TR Algorithm**

The TR algorithm traces the K-th boundary by deciding the next direction of travel out of N-neighbors (Fig. 3-4). The searching operation for finding the next direction is shown in Fig. 3-7.

![Diagram showing the order of examination for finding next search direction.](image)

- **a) Even LP (LP=0)**
- **b) Odd LP (LP=7)**

Fig. 3-7 The Order of Examination for finding Next Search Direction. The dashed line denotes present chain link (LP), and the broken line denotes the next chain link candidates (LN).

The search process is referred to the clockwise modulo 8 operation relative to the direction of the present link. Let P be a current point whose N-neighbors are examined and LN be a search direction in terms of common 8-directional chain code shown in Fig. 3-1. All mathematical operations on LN are assumed to be modulo 8.
(1) Set: \( P = [X_S(K), Y_S(K)] \). Go to (2).

(2) Search Direction is LN. Go to (3).

* Note: LN varies with respect to LP.

(3) Evaluate the right-neighbor points related to LN. This evaluation process for the cases of LN = 2 and LN = 5 is illustrated in Fig. 3-8.

a) If all points are zero, go to (4).

b) If any of the points is greater than or equal to \( F(K) \), go to (5).

c) If at least one of the points is nonzero, go to (6).

(4) Set: \( LN = LN - 1 \). Go to (3).

(5) Determine the chain link contact configuration.

a) If one of the configurations in (a) of Fig.3-9 has occurred, go to (6).

b) If one of the configurations in (b) of Fig.3-9 has occurred, go to (4).

(6) Set: \( LP = LN \) and \( CH(I,K) = LP \).

Go to (7).

(7) To denote the element of the I-th link of the K-th boundary, all values \( W(m,n) \) of the points covered by the chain link LP are replaced by \( F(K) = F(K) + 1 \), recursively. But, if any of the pixels covered by the I-th chain link
Fig. 3-8 Right-Neighbor Points for Evaluation of Next Direction. (Chain Link Size = 2 is used)

Fig. 3-9. Possible Chain Link Contact Configuration between Present Chain Link (Solid Line) and Previous Chain Link (Dashed Line).
is greater than \( R \), \( S \times I \) is then added to the value that was already assigned for that pixel. Go to (8).

(8) To determine the completeness of the \( K \)-th boundary, \( X_P(K) \) and \( Y_P(K) \) are compared with the location of the \( K \)-th SP.

If \( X_P(K) \neq X_S(K) \) or \( Y_P(K) \neq Y_S(K) \),

set : \( P = [X_P(K), Y_P(K)] \) and \( I = I + 1 \)
and go to (2).

If not, the \( K \)-th boundary is already completed.
The TR algorithm terminates.

**ASGN Algorithm**

This assignment algorithm replaces the values \( W(m,n) \) of all points inside the \( K \)-th boundary to \( F(K) \), which distinguishes all interior points of the \( K \)-th boundary from others. The search for determining interior points of the \( K \)-th boundary starts with the lower right-neighbor of SP and performs left-to-right and top-to-bottom scanning until no point greater than \( F(K) \) is detected.

Define :

\( RN, CN \) = Counters for checking row and column numbers for the point currently being examined.

\( NIK \) = Counter for the number of intersections of the \( K \)-th boundary with the current row.
L = Number of points covered by one chain link. This can be referred to as the resolution of the chain.

I = The I-th chain link of the K-th boundary.

J = The J-th element of the I-th link.

BN, IL = Dummy variable for I and J.

Initialization:

\[ [X S(K) + 1, Y S(K) + 1] = \text{Starting Point for ASGN algorithm.} \]

RN = XS(K) + 1, CN = YS(K) + 1.

NIK = 1.

(1) Set the present value \( W(RN, CN) = P(K) \).

(2) Move to the right neighbor.

Set: \( CN = CN + 1 \)

(3) Check the column number.

If \( CN = M + 2L \), go to (5).

If not, go to (7).

(4) The location of the point which is now the subject of the search is the end of the current row. If there is no point greater than RK, or this is the last row of the buffer array W, the K-th boundary is already filled completely by RK. Thus, the ASGN algorithm terminates. That is,

If \( NIK = 0 \) or \( RN = M + 2L \), ASGN Algorithm terminates.

If not, go to (6).
(5) Move to the next row and refresh the counter for column and intersection.
Set: RN = RN + 1,
NIK = 0,
CN = 1.
Go to (7).

(6) Determine whether or not the point is on the boundary.
If \( W(RN, CN) > RK \), go to (8).
If not, go to (17).

(7) Find I and J
a) Set: \( BN = W(RN, CN) - RK, \ IL = 1. \)
Go to b).
b) If \( BN > L \), \( BN = BN - L \) and go to c).
If not, go to d).
c) Set: \( IL = IL + 1. \)
Go to b).
d) Set: \( J = BN, \ I = IL. \)

Dummy variables, BN and IL, will be refreshed with zeroes. Go to (9).

(8) Determine the chain code of the present link.
(Refer to Fig. 3-1)
If \( CH(I, K) = 4 \), go to (10).
If \( CH(I, K) = 0 \), go to (11).
If \( CH(I, K) = 0, 4 \), go to (12).

(9) \( CH(I, K) \) is 4.
The position of the point under examination is always the head point of the chain link. The head point of the previous chain link always exists behind the present chain link, \( CH(I,K) = 4 \). Move to the head of the previous chain link.

Set: \( CN = CN + L \),
\[ I = I - 1. \]

* Note that \( J \) is the same because the head points of all the chain links have the same number \( L \).

Go to (9).

(10) \( CH(I,K) \) is 0.

The present point is always the tail of the chain link, and the direction of \( CH(I,K)=0 \) is right. Move to the right by \( L \) to determine the next 0 chain link.

a) Set: \( CN = CN + L \).

b) If \( W(RN,CN) > RK \), go to a).

If not, go to (17).

(11) Check the value \( J \) to determine whether or not that point is the head of the chain link.

If \( J = L \), go to (13).

If not, go to (16).

(12) To find the shape of the chain links, the current chain code is considered.

If \( CH(I,K) = 1, 2, \) or \( 3 \), go to (14).

If \( CH(I,K) = 5, 6, \) or \( 7 \), go to (15).
To determine whether the current point is an intersection or not, the next chain codes are examined in (14) and (15).

(13) If \( \text{CH}(I+1,K) = 5, 6, \) or 7, go to (3).
    If \( \text{CH}(I+1,K) = 0, \) go to (11).
    If \( \text{CH}(I+1,K) = 0, 5, 6, \) and 7, go to (16).

(14) If \( \text{CH}(I+1,K) = 1, 2, \) or 3, go to (3).
    If \( \text{CH}(I+1,K) = 0, \) go to (11).
    If \( \text{CH}(I+1,K) = 0, 1, 2, \) or 3, go to (16).

(15) Count the number of intersections.
    Set \( \text{NIK} = \text{NIK} + 1 \)
    Go to (3).

(16) Examine the number of intersections to find whether the point belongs to the \( K \)-th boundary or not.
    If \( \text{NIK} = \text{odd}, \) go to (18).
    If not, go to (3).

(17) Replace the value of the point inside the \( K \)-th boundary.
    Set \( \text{W(RN,CN)} = R \times K \)
    Go to (3).

After chain coding is done, the buffer \( \text{W} \) is used as a reference for a two-to-one dimensional mapping process of the chain enclosed coefficients within the transform sub-block \( H \), and all the chain links for each boundary are
removed for the subsequent encoding process. The actual chain coding process for the typical transform sub-block is presented in Table 3-2 and Table 3-3. In Table 3-2(b), we can see the actual manipulation process of the transform sub-block of (a) with the buffer array. The transform sub-block of (a) was superimposed on the center portion of the buffer array, and all the nonzero coefficients were replaced by 1. The SP and TR algorithms were applied on this buffer array to cluster the nonzero coefficients within the buffer. The resulting chain boundaries are also illustrated in Table 3-2(b). The chain link size of $L = 2$ was used to trace each boundary of the nonzero coefficients. After finding each chain boundary, the ASGN algorithm is applied to identify the nonzero coefficients inside the chain boundary just generated. This assignment process is depicted in Table 3-3.
(a) Transform coefficients of the (6, 8)th subblock of MIKE12 after quantization process.

(b) Buffer array of (6, 8)th subblock of MIKE12

Table 3-2. Chain coding process of a typical transform subblock. Manipulation of the transform subblock with buffer is shown in (b). Chain boundaries are illustrated in (b). Chain link size of \( L = 2 \) was used.
TABLE 3-3. ASGN PROCESSES FOR TABLE 3-2(b) AFTER CHAIN CODING OF EACH BOUNDARY. 100000 AND 200000 WERE USED TO IDENTIFY THE FIRST AND THE SECOND BOUNDARY, RESPECTIVELY (i.e., R=100000 and S=100).
3.3.2 Boundary Encoding

The regions for nonzero coefficients (or zero coefficients) were efficiently clustered by implementing the boundary-tracing algorithm, and one of the chain code numbers was assigned to represent each chain link of the boundary. Thus, the K-th boundary can be represented by using the K-th SP and the chain code for each chain link of the K-th boundary. The SP will be coded using a fixed length block code. As for the chain links, if we use fixed length block code, three bits per chain link is required. However, since there are usually not so many curvatures in the boundary, a more efficient code can be offered by the differential chain code, which is represented by the difference between adjacent pairs of the chain links. Since there is no possibility of a difference of ±4 in the boundary tracing algorithm described in the previous section, we have 7 values (0, ±1, ±2, ±3) of the differential chain code, which provides orientation independent expressions of the chain link. For smooth curves, the values 0, ±1, and ±2 will occur more frequently than others. Since the occurrence values of the differential chain code are not equally likely, a variable length code word is more suitable than block code.

For the differential chain coding of the K-th boundary, the chain links are scanned, and the difference between adjacent pairs of the chain links is coded using the
** BLOCK CODE(4) FOR BOUNDARY STARTING POINT**

<table>
<thead>
<tr>
<th>POSITION</th>
<th>CODE WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X = 2$</td>
<td>$0 0 0 0$</td>
</tr>
<tr>
<td>$Y = 2$</td>
<td>$0 0 0 0$</td>
</tr>
</tbody>
</table>

** CHAIN ENCODING FOR THE BOUNDARY NUMBER 1**

<table>
<thead>
<tr>
<th>LINK NUMBER</th>
<th>CHAIN CODE</th>
<th>DIFF. CODE</th>
<th>CODE WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>-1</td>
<td>010</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>-3</td>
<td>1100</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>-1</td>
<td>1011</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>-2</td>
<td>1000</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>1</td>
<td>0111</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>1</td>
<td>0111</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>-1</td>
<td>0101</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>1</td>
<td>0111</td>
</tr>
<tr>
<td>17</td>
<td>6</td>
<td>1</td>
<td>0111</td>
</tr>
<tr>
<td>18</td>
<td>6</td>
<td>-1</td>
<td>0111</td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td>1</td>
<td>0111</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>-1</td>
<td>0110</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>3</td>
<td>1101</td>
</tr>
<tr>
<td>22</td>
<td>7</td>
<td>-1</td>
<td>0110</td>
</tr>
<tr>
<td>23</td>
<td>4</td>
<td>-3</td>
<td>1100</td>
</tr>
<tr>
<td>24</td>
<td>5</td>
<td>-1</td>
<td>1110</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>-3</td>
<td>1100</td>
</tr>
<tr>
<td>26</td>
<td>6</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>27</td>
<td>2</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>28</td>
<td>3</td>
<td>-1</td>
<td>0010</td>
</tr>
<tr>
<td>29</td>
<td>2</td>
<td>-2</td>
<td>0101</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>0</td>
<td>0100</td>
</tr>
<tr>
<td>31</td>
<td>2</td>
<td>-2</td>
<td>1010</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>-2</td>
<td>0110</td>
</tr>
<tr>
<td>33</td>
<td>4</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>34</td>
<td>2</td>
<td>-2</td>
<td>1010</td>
</tr>
<tr>
<td>35</td>
<td>2</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>36</td>
<td>2</td>
<td>EOC</td>
<td>1110</td>
</tr>
</tbody>
</table>

** BLOCK CODE(4) FOR BOUNDARY STARTING POINT**

<table>
<thead>
<tr>
<th>POSITION</th>
<th>CODE WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X = 8$</td>
<td>$0 1 1 0$</td>
</tr>
<tr>
<td>$Y = 9$</td>
<td>$1 1 1 0$</td>
</tr>
</tbody>
</table>

** CHAIN ENCODING FOR THE BOUNDARY NUMBER 2**

<table>
<thead>
<tr>
<th>LINK NUMBER</th>
<th>CHAIN CODE</th>
<th>DIFF. CODE</th>
<th>CODE WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-2</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>-2</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>-2</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>EOCB</td>
<td>1111</td>
</tr>
</tbody>
</table>

Table 3-4. Encoding of the chain boundaries shown in (b) of Table 3-2. Chain code, differential chain code, and instantaneous code word are given in this table. EOC and EOCB denote the end of chain code and the end of chain block respectively.
instantaneous coding table shown in Table 3-5. This coding table includes "end of chain (EOC)" and "end of chain block (EOCB)". The EOC is used to terminate decoding of the chain link for a boundary, and the EOCB is used to distinguish the chain block from the amplitude code. The differential chain coding process is defined as

$$S_j = \begin{cases} L(I,K), & \text{if } I = 1 \\ L(I,K) - L(I-1,K), & \text{if } I \neq 1 \end{cases}$$

$$1 \leq j \leq 7,$$

where $S_j$ is the differential chain code, and $L(I,K)$ denotes the $I$-th chain link of the $K$-th boundary. The encoding process for each chain boundary of Table 3-3(b) is given in Table 3-4.
Table 3-5. Variable Length Instantaneous Code for Differential Chain Code. EOC and EOCB denote the end of chain code and end of chain block, respectively.

<table>
<thead>
<tr>
<th>CODE SYMBOL</th>
<th>BINARY CODE WORD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>-1</td>
<td>010</td>
</tr>
<tr>
<td>1</td>
<td>011</td>
</tr>
<tr>
<td>-2</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>101</td>
</tr>
<tr>
<td>-3</td>
<td>1100</td>
</tr>
<tr>
<td>3</td>
<td>1101</td>
</tr>
<tr>
<td>EOC</td>
<td>1110</td>
</tr>
<tr>
<td>EOCB</td>
<td>1111</td>
</tr>
</tbody>
</table>


CHAPTER VI

ADAPTIVE HYBRID CODING TECHNIQUE

Since the temporal characteristics of a sequence of moving image frames vary considerably, the predictive coding technique along the temporal direction is advantageous in reducing the bandwidth required to transmit multiframe image data. However, interframe prediction techniques are very sensitive to the variations which are introduced by motion in the successive frames. To minimize the variations due to the motion and improve the performance of interframe prediction, motion compensation could be made between frames. On the other hand, for spatial image data, the transform coding technique has proven to be more efficient over other intraframe coding techniques.

By combining 2-D transform and interframe coding techniques, hybrid coding has the advantages of the simplistic hardware configuration of a predictive coder and the robust performance of a transform coder. Moreover, motion compensation techniques can give drastic improvements in bit rate reduction for multiframe images and can be efficiently included in the hybrid coding scheme. Using hybrid coding, one can achieve more bandwidth compression over interframe and transform coding techniques.

In this chapter, in order to realize an efficient
adaptive hybrid coding technique, a motion compensated interframe prediction technique of Chapter II is combined with the transform coding technique of Chapter III, and the performance is evaluated using the computer simulation.

4.1 Adaptive Interframe Transform Coding Technique

In this adaptive hybrid coding technique, the image frame is divided into small sub-blocks of size $M \times M$ and subtracted, in the spatial domain, by the motion compensated previous frame. For this prediction operation, the storage of only the previous frame is required at both transmitter and receiver. The prediction error signal is minimized by using the block matching directional search method described in Section 2.2 of Chapter II. The adaptive transform coding technique of Chapter III is then applied to the prediction error signal of each sub-block. Since the transform adaptive coding technique depends only upon the contents of each transform sub-block, it is well adapted to the motion compensated interframe prediction scheme for a sequence of motion images. A reverse process at the receiver reproduces the approximation of the original image within an acceptable range of degradation. The system is made adaptive in the temporal direction by the motion compensated prediction of the present frame and in the spatial domain by the selection of the transform coefficients. Fig. 4-1 shows the block diagram of the adaptive interframe transform coding scheme.
4.1.1 Motion Compensated Interframe Prediction

Let $U(x,y,K)$ be an $M \times M$ sub-block of the $K$-th frame and let it represent the intensity value at the location $(x,y)$. If the sub-block size is small enough, motion introduced by larger objects can be closely approximated by piecewise linear translation of each sub-block. By assuming that image data is wide sense stationary, and each sub-block undergoes independent translation, the motion compensated interframe prediction error signal $e(x,y,K)$ for a sub-block at the $K$-th frame can be obtained by subtracting the motion compensated prediction signal from the actual pixel value $U(x,y,K)$; i.e.,

$$e(x,y,K) = U(x,y,K) - U_p(x,y,K) ,$$ \hfill (4-1)

where $U_p(x,y,K)$ represents the motion compensated prediction signal used as an estimate of the actual pixel value at location $(x,y)$ in the $K$-th frame. The motion compensated prediction signal, which is formed in the space domain by using the reconstructed intensity values of the previous frame, can be represented as

$$U_p(x,y,K) = \hat{\Phi}U(x+i,y+j,K-1) ,$$ \hfill (4-2)
Fig. 4-1. Adaptive Hybrid Coding/Decoding Scheme
where \( \hat{U}(x,y,K-1) \) denotes the reproduced value of \( U(x,y,K-1) \) after the quantization process; \( \theta \) denotes the temporal correlation coefficient, and \( (i,j) \) is the motion displacement vector. The motion displacement vector \( (i,j) \) can be found by using the 2-D directional search algorithm given in Section 2.2 of Chapter II. In this search algorithm, the direction of the minimum mean square distortion between a sub-block in the present frame and a corresponding area within a search window in the previous frame stored in the predictor, as a reference image for prediction, is taken as the motion displacement vector \( (i,j) \). Once the motion displacement vector \( (i,j) \) has been found, a motion compensated estimate of the sub-block is given by the area in the reference image displaced, according to the displacement vector \( (i,j) \). If \( e(x,y,K) \) is below the threshold, the block is classified as stationary block, i.e.,

\[
e(x,y,K) = 0 ,
\]

and only the motion displacement vector is transmitted for motion compensation of the corresponding previous sub-block at the receiver. Otherwise, \( e(x,y,K) \) of (4-1) is processed for transmission. The motion compensated reference image can be obtained by collecting all the motion compensated estimates from the reference image. The displacement vector \( (i,j) \) for a sub-block is coded and
transmitted as side information.

At the receiver, the reproduced motion compensated estimate \( U^*_p(x,y,K) \), which is the version of \( U(x,y,K) \) corrupted by the channel noise, can be expressed as

\[
U^*_p(x,y,K) = \hat{U}^*(x+i,y+j,K-1), \tag{4-4}
\]

where \( * \) denotes the signal corrupted by the channel noise and \( \hat{U}^*(x,y,K-1) \) is the reference image stored in the predictor memory at the receiver. The reproduced channel corrupted prediction error signal \( \hat{e}^*(x,y,K) \) is added to the motion compensated estimate, which is formed by shifting the sub-block in the reference image according to the decoded motion displacement vector \((i,j)\), to reconstruct the approximated version of the original image. The reproduced version of the original image can be represented by

\[
\hat{U}^*(x,y,K) = U^*_p(x,y,K) + \hat{e}^*(x,y,K) \tag{4-5}
\]

In the absence of the channel noise,

\[
\hat{e}^*(x,y,K) = \hat{e}(x,y,K) \tag{4-6}
\]

and (4-4) is same as (4-3). Where \( \hat{e}(x,y,K) \) is the reproduced version of the prediction error signal \( e(x,y,K) \) after quantization process. Thus, at the receiver, the reproduced...
image becomes

\[ \hat{U}(x, y, K) = \vec{U}(x, y, K), \quad (4-7) \]

The effect of the motion compensation in the prediction error signal is shown in Fig.2-6. For further improvement of the hybrid coding performance the temporal correlation coefficient \( \theta \), given by (2-12), is used as a prediction coefficient. In general, \( \theta \) is a variable. However, this variable rate \( \theta \) increases the complexity of the hybrid coder shown in Fig.4-1. To avoid this problem, a fixed value of \( \theta \) is used as a prediction coefficient.

4.1.2 Adaptive Transform Coding Operation

The adaptive transform coding technique, which operates on single image frame, is applied to the motion compensated prediction error signal \( e(x, y, K) \). The 2-D unitary transform coefficients for an \( M \times M \) size sub-block of \( e(x, y, K) \) is given by (3-1) as

\[ H(u, v, K) = \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} e(x, y, K)P(x, y, u, v), \quad (4-8) \]

where \( H(u, v, K) \) represents the 2-D unitary transform coefficient and \( P(x, y, u, v) \) is the 2-D unitary transform
kernel. The thresholding process is performed on the AC transform coefficients within the sub-block by using (3-9). The AC coefficients lower than the threshold value will be changed to zero, and a small portion of the large magnitude AC coefficients will be retained. That is;

\[
\begin{align*}
\text{if } |H(u,v,K)| &> T, \\
H_T(u,v,K) &= \begin{cases} 
H(u,v,K) - T, & \text{ when } H(u,v,K) > 0. \\
H(u,v,K) + T, & \text{ when } H(u,v,K) < 0.
\end{cases}
\end{align*}
\]

\[
\text{if } |H(u,v,K)| \leq T, \\
H_T(u,v,K) = 0,
\]

where \(T\) represents properly chosen threshold and \(\cdot\) denotes the absolute value. Through this deleting process of the lower AC energy coefficients, the transform coefficients of the largest absolute magnitude are retained. The retained coefficients are normalized by the normalization parameter \(D\) given by (3-10). All the AC coefficients with magnitude above the threshold are retained for subsequent quantization. The integer floating point round-off quantizer given by (3-11) is used for quantization. During this quantization process many fractional values due to the thresholding and normalization will be changed to zero. Only significant coefficients will remain as nonzero coefficients.

Since the mean value of the transform sub-block is
already subtracted in the prediction step, there is no longer any difference, in principle, between DC and AC coefficients. However, since the DC coefficient, i.e., the first component within the transform sub-block, represents the fractional value of the average brightness of the original image sub-block, an error in the DC coefficient will be more critical in the reconstructed image. Thus, care should be taken to avoid this critical error in the reproduced image at the receiver. It is excluded in the thresholding and normalization processes to avoid the error incurred by those processes.

To retain the addressing information of the remaining significant coefficients within the sub-block, the chain coding method described in Section 3.3 is applied to the quantized version of the transform sub-block. The regions of the nonzero coefficients are identified through the boundary tracing algorithm of Section 3.3.1, and then encoded with the instantaneous variable length coding table shown in Table 3-1. By transmitting this information as overhead, the addressing information can be efficiently retained at the receiver. Thus, only the coefficients inside the boundaries need to be coded. For encoding the coefficients within the boundaries, two-to-one dimensional mapping of the chain enclosed coefficients is performed on the transform sub-block, and the magnitudes are then coded with the Huffman coding table derived from a typical histogram of the transform AC coefficients. The DC coefficient and the
<table>
<thead>
<tr>
<th>INTERVAL</th>
<th>DISTRIBUTION OF OCCURANCES</th>
<th>NO. REL. FREQ</th>
<th>CUMULATIVE PROBABILITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50I</td>
<td>26479</td>
<td>0.4040</td>
<td>0.4040</td>
</tr>
<tr>
<td>1.50I</td>
<td>14437</td>
<td>0.2201</td>
<td>0.6240</td>
</tr>
<tr>
<td>2.50I</td>
<td>7498</td>
<td>0.1141</td>
<td>0.7391</td>
</tr>
<tr>
<td>3.50I</td>
<td>4391</td>
<td>0.0671</td>
<td>0.8065</td>
</tr>
<tr>
<td>4.50I</td>
<td>2852</td>
<td>0.0431</td>
<td>0.8494</td>
</tr>
<tr>
<td>5.50I</td>
<td>1974</td>
<td>0.0301</td>
<td>0.8795</td>
</tr>
<tr>
<td>6.50I</td>
<td>1528</td>
<td>0.0231</td>
<td>0.9036</td>
</tr>
<tr>
<td>7.50I</td>
<td>1092</td>
<td>0.0171</td>
<td>0.9207</td>
</tr>
<tr>
<td>8.50I</td>
<td>838</td>
<td>0.0131</td>
<td>0.9338</td>
</tr>
<tr>
<td>9.50I</td>
<td>684</td>
<td>0.0101</td>
<td>0.9436</td>
</tr>
<tr>
<td>10.50I</td>
<td>514</td>
<td>0.0081</td>
<td>0.9517</td>
</tr>
<tr>
<td>11.50I</td>
<td>455</td>
<td>0.0071</td>
<td>0.9578</td>
</tr>
<tr>
<td>12.50I</td>
<td>359</td>
<td>0.0051</td>
<td>0.9630</td>
</tr>
<tr>
<td>13.50I</td>
<td>333</td>
<td>0.0051</td>
<td>0.9681</td>
</tr>
<tr>
<td>14.50I</td>
<td>277</td>
<td>0.0041</td>
<td>0.9722</td>
</tr>
<tr>
<td>15.50I</td>
<td>215</td>
<td>0.0031</td>
<td>0.9753</td>
</tr>
<tr>
<td>16.50I</td>
<td>180</td>
<td>0.0021</td>
<td>0.9774</td>
</tr>
<tr>
<td>17.50I</td>
<td>163</td>
<td>0.0021</td>
<td>0.9815</td>
</tr>
<tr>
<td>18.50I</td>
<td>128</td>
<td>0.0021</td>
<td>0.9846</td>
</tr>
<tr>
<td>19.50I</td>
<td>115</td>
<td>0.0021</td>
<td>0.9867</td>
</tr>
<tr>
<td>20.50I</td>
<td>94</td>
<td>0.0011</td>
<td>0.9888</td>
</tr>
<tr>
<td>21.50I</td>
<td>80</td>
<td>0.0011</td>
<td>0.9899</td>
</tr>
<tr>
<td>22.50I</td>
<td>79</td>
<td>0.0011</td>
<td>0.9920</td>
</tr>
<tr>
<td>23.50I</td>
<td>64</td>
<td>0.0011</td>
<td>0.9931</td>
</tr>
<tr>
<td>24.50I</td>
<td>57</td>
<td>0.0011</td>
<td>0.9944</td>
</tr>
<tr>
<td>25.50I</td>
<td>51</td>
<td>0.0011</td>
<td>0.9955</td>
</tr>
<tr>
<td>26.50I</td>
<td>63</td>
<td>0.0011</td>
<td>0.9955</td>
</tr>
<tr>
<td>27.50I</td>
<td>36</td>
<td>0.0011</td>
<td>0.9966</td>
</tr>
<tr>
<td>28.50I</td>
<td>49</td>
<td>0.0011</td>
<td>0.9962</td>
</tr>
<tr>
<td>29.50I</td>
<td>34</td>
<td>0.0011</td>
<td>0.9994</td>
</tr>
<tr>
<td>30.50I</td>
<td>24</td>
<td>0.0001</td>
<td>0.9994</td>
</tr>
<tr>
<td>31.50I</td>
<td>34</td>
<td>0.0001</td>
<td>0.9995</td>
</tr>
<tr>
<td>32.50I</td>
<td>16</td>
<td>0.0001</td>
<td>0.9995</td>
</tr>
<tr>
<td>33.50I</td>
<td>22</td>
<td>0.0001</td>
<td>0.9995</td>
</tr>
<tr>
<td>34.50I</td>
<td>25</td>
<td>0.0001</td>
<td>0.9996</td>
</tr>
<tr>
<td>35.50I</td>
<td>23</td>
<td>0.0001</td>
<td>0.9996</td>
</tr>
<tr>
<td>36.50I</td>
<td>14</td>
<td>0.0001</td>
<td>0.9996</td>
</tr>
<tr>
<td>37.50I</td>
<td>20</td>
<td>0.0001</td>
<td>0.9997</td>
</tr>
<tr>
<td>38.50I</td>
<td>14</td>
<td>0.0001</td>
<td>0.9997</td>
</tr>
<tr>
<td>39.50I</td>
<td>17</td>
<td>0.0001</td>
<td>0.9997</td>
</tr>
<tr>
<td>40.50I</td>
<td>11</td>
<td>0.0001</td>
<td>0.9997</td>
</tr>
<tr>
<td>41.50I</td>
<td>9</td>
<td>0.0001</td>
<td>0.9997</td>
</tr>
<tr>
<td>42.50I</td>
<td>10</td>
<td>0.0001</td>
<td>0.9997</td>
</tr>
<tr>
<td>43.50I</td>
<td>9</td>
<td>0.0001</td>
<td>0.9998</td>
</tr>
<tr>
<td>44.50I</td>
<td>8</td>
<td>0.0001</td>
<td>0.9998</td>
</tr>
</tbody>
</table>

**HISTOGRAM INTERVAL = 1.000000**

**Fig. 4-2. HISTOGRAM AND CUMULATIVE DENSITY FUNCTION OF THE 2-D ORDERED HADAMARD TRANSFORM COEFFICIENTS FOR THE DIFFERENCE VALUES BETWEEN MIKE15 AND MIKE16.**
displacement vector \((i,j)\) are coded by using the fixed length block code. The histogram and cumulative probability density function for the 2-D Hadamard transform coefficients of the motion compensated prediction error signal is shown in Fig. 4-2.

In the feedback loop of Fig. 4-1, for the prediction operation of the next frame, the quantized coefficients are also fed through the inverse normalization, thresholding, and transform processes to produce an approximated version \(\hat{e}(x,y,K)\) of the prediction error signal \(e(x,y,K)\). The inverse normalization process is defined by

\[
\hat{H}_T(u,v,K) = D\hat{H}_{TN}(u,v,K), \tag{4-10}
\]

where \(D\) denotes the normalization factor and \(\hat{H}_{TN}(u,v,K)\) denotes the quantized version of the normalized transform coefficient. The threshold value is then added to \(\hat{H}_T(u,v,K)\); i.e.,

\[
\hat{H}(u,v,K) = \begin{cases} 
\hat{H}_T(u,v,K) + T, & \text{if } \hat{H}_T(u,v,K) > 0 \\
\hat{H}_T(u,v,K) - T, & \text{if } \hat{H}_T(u,v,K) < 0
\end{cases}
\]

if \(\hat{H}_T(u,v,K) \neq 0\), \(\hat{H}(u,v,K) \neq 0\), \(\hat{H}(u,v,K) = 0\), \(\hat{H}_T(u,v,K) = 0\), \(\hat{H}(u,v,K) = 0\), \tag{4-11}
where \(|\cdot|\) denotes the absolute value. To reproduce \(\hat{e}(x,y,K)\) the inverse transform is performed by using Eq. (3-2):

\[
\hat{e}(x,y,K) = \sum_{u=0}^{M-1} \sum_{v=0}^{M-1} H(u,v,K) P(x,y,u,v)
\]

The estimate \(U_p(x,y,K)\), used for prediction of the \(K\)-th frame of Eq. (4-3), is formed by taking a linear combination of previously generated motion compensated interframe error signals. That is,

\[
U_p(x,y,K) = \theta[U_p(x,y,K-1) + \hat{e}(x,y,K-1)]
\]

\[
= \theta^{K-1} U_p(x,y,1) + \sum_{q=1}^{K-1} \theta^q \hat{e}(x,y,q) .
\]

At the receiver, Eq. (4-4) will be

\[
U_p^*(x,y,K) = \theta^{K-1} U_p^*(x,y,K-1) + \sum_{q=1}^{K-1} \theta^q \hat{e}^*(x,y,q) .
\]

As shown in (2-18), the correlation coefficient \(\theta\) is chosen to minimize the variance of the motion compensated interframe error signal. If the threshold and quantization processes are placed outside the feedback loop, the summation of the previous frame estimate and the motion compensated interframe difference produces an exact replica of the actual pixel value of the corresponding position with
one frame delay. In the absence of channel noise, if there are no thresholding and quantization steps, the interframe motion compensated estimate at the receiver is identical to the estimate at the transmitter. Since thresholding and quantization processes cause errors in the reconstructed image, the errors due to those processes may accumulate because the transmitter and receiver will not be identical if the processes are performed outside the feedback loop. To eliminate the error accumulation at the receiver and thereby force the transmitter and receiver estimates to be identical, these processes are included in the prediction loop.

4.2. Error Consideration

In the adaptive hybrid coding technique proposed in Section 4.1, the total system mean square block distortion introduced by the coding operation can be expressed as

\[
\text{ERR}_t = \frac{1}{M^2} \sum_{x=0}^{M-1} \sum_{y=0}^{M-1} \mathbb{E}[(U(x,y,K) - \hat{U}(x,y,K))^2]. \tag{4-15}
\]

By the orthogonal property of the unitary transform, (4-15) can be alternatively written in the transform domain as

\[
\text{ERR}_t = \frac{1}{M^2} \sum_{u=0}^{M-1} \sum_{v=0}^{M-1} \mathbb{E}[(H(u,v,K) - \hat{H}(u,v,K))^2]. \tag{4-16}
\]
Since, in the absence of channel noise, the motion compensated interframe prediction operation of Section 4.1.1 involves no approximation process of the original image and is designed to be identical in both transmitter and receiver, the error during the coding operation is only given by the data compression abilities of the adaptive transform operation derived from the thresholding, normalization, and quantization of the coefficients in the transform domain.

The mean square deletion errors are given by (3-14) and the mean square quantization error can be calculated by assuming particular probability density models for the transform coefficients. In addition to the errors due to the deletion and quantization processes, the threshold subtracting operation and normalization process will increase the error in the reconstructed image by limiting the number of coefficients to be quantized. In this adaptive hybrid coding scheme, it is meaningless to determine each error separately because any further optimization may not be made by analyzing each error. The total system mean square error of (4-15) is calculated to evaluate the coding performance.
4.3 Simulation Results

A series of computer simulations was performed to evaluate the performance of the proposed adaptive hybrid coding technique of Section 4.1 in this chapter. For the transform operation, a 2-D ordered Hadamard transform is employed to exploit a spatial redundancy within an image frame. Since 2-D ordered Hadamard transform kernels are separable and identical, the 2-D ordered Hadamard transform coefficients can be generated by applying the 1-D transform algorithm twice. The 1-D ordered Hadamard transform algorithm used for our simulations and its computational flow diagram are given in Appendix B with a brief description of the statistical properties of the Hadamard transform coefficients. The same sub-block size 16 x 16 as that of Section 2.3 in Chapter II was used for our simulations. Since the interframe prediction technique requires the knowledge of the initial image frame, it is assumed as the initial condition that the first frame is available without any distortion at the receiver. The original image data used for computer simulations is described in Section 1.4.

In the simulation of the adaptive hybrid coding, the original image data is first divided into smaller sub-blocks and subtracted by the corresponding motion compensated previous sub-block in the spatial domain. The 2-D ordered Hadamard transform is applied to this prediction error
block. Table 4-1 shows the intensity values of typical sub-block in the previous and present frames and Table 4-2 shows the motion compensated sub-block in the previous frame. The motion compensated interframe prediction error and its 2-D ordered Hadamard transform coefficients are shown in Table 4-3. Because, as shown in Table 2-5, the motion compensated prediction operation minimizes the prediction error, only the fractional values of the original present image sub-block remain in the prediction error sub-block of (a) in Table 4-3. Since, by transmitting only the motion displacement vector, the motion compensated previous sub-block is already available at the receiver, great saving is achieved in this motion compensated prediction operation. After prediction, exactly the same processes as those of the transform adaptive coding of Fig. 3-2 are performed on this prediction error sub-block. In adaptive transform coding, the actual data compression is achieved by a thresholding process and additional compression is gained by following normalization and quantization operations. The resulting sub-blocks of the prediction error after these processes are given in Table 4-4(a-c), respectively. In these tables we clearly see the data compression operation of each step of thresholding, normalization, and quantization processes. After the quantization process, only a small number of significantly large magnitude transform coefficients is retained with considerably lower magnitude than the original. All the low magnitude coefficients are changed to
(a) INTENSITY VALUE OF (7, 9)th SUBBLOCK OF THE PREVIOUS FRAME

<table>
<thead>
<tr>
<th>101.0</th>
<th>101.0</th>
<th>107.0</th>
<th>103.0</th>
<th>91.0</th>
<th>95.0</th>
<th>101.0</th>
<th>99.0</th>
<th>95.0</th>
<th>87.0</th>
<th>83.0</th>
<th>85.0</th>
<th>89.0</th>
<th>89.0</th>
<th>83.0</th>
<th>75.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>103.0</td>
<td>99.0</td>
<td>105.0</td>
<td>105.0</td>
<td>95.0</td>
<td>93.0</td>
<td>101.0</td>
<td>99.0</td>
<td>91.0</td>
<td>97.0</td>
<td>90.0</td>
<td>95.0</td>
<td>87.0</td>
<td>85.0</td>
<td>87.0</td>
<td>87.0</td>
</tr>
<tr>
<td>101.0</td>
<td>107.0</td>
<td>109.0</td>
<td>99.0</td>
<td>87.0</td>
<td>89.0</td>
<td>95.0</td>
<td>101.0</td>
<td>99.0</td>
<td>89.0</td>
<td>95.0</td>
<td>87.0</td>
<td>85.0</td>
<td>87.0</td>
<td>87.0</td>
<td>83.0</td>
</tr>
<tr>
<td>103.0</td>
<td>103.0</td>
<td>107.0</td>
<td>101.0</td>
<td>91.0</td>
<td>87.0</td>
<td>93.0</td>
<td>99.0</td>
<td>97.0</td>
<td>87.0</td>
<td>83.0</td>
<td>85.0</td>
<td>85.0</td>
<td>87.0</td>
<td>87.0</td>
<td>79.0</td>
</tr>
<tr>
<td>109.0</td>
<td>107.0</td>
<td>105.0</td>
<td>103.0</td>
<td>99.0</td>
<td>87.0</td>
<td>89.0</td>
<td>97.0</td>
<td>101.0</td>
<td>99.0</td>
<td>89.0</td>
<td>95.0</td>
<td>87.0</td>
<td>85.0</td>
<td>87.0</td>
<td>87.0</td>
</tr>
<tr>
<td>101.0</td>
<td>105.0</td>
<td>105.0</td>
<td>99.0</td>
<td>87.0</td>
<td>87.0</td>
<td>93.0</td>
<td>99.0</td>
<td>97.0</td>
<td>89.0</td>
<td>85.0</td>
<td>85.0</td>
<td>87.0</td>
<td>87.0</td>
<td>87.0</td>
<td>87.0</td>
</tr>
<tr>
<td>107.0</td>
<td>107.0</td>
<td>111.0</td>
<td>97.0</td>
<td>79.0</td>
<td>89.0</td>
<td>99.0</td>
<td>103.0</td>
<td>95.0</td>
<td>89.0</td>
<td>87.0</td>
<td>87.0</td>
<td>89.0</td>
<td>89.0</td>
<td>89.0</td>
<td>91.0</td>
</tr>
<tr>
<td>107.0</td>
<td>105.0</td>
<td>111.0</td>
<td>99.0</td>
<td>83.0</td>
<td>87.0</td>
<td>97.0</td>
<td>103.0</td>
<td>95.0</td>
<td>87.0</td>
<td>85.0</td>
<td>87.0</td>
<td>87.0</td>
<td>87.0</td>
<td>89.0</td>
<td>89.0</td>
</tr>
<tr>
<td>109.0</td>
<td>103.0</td>
<td>103.0</td>
<td>95.0</td>
<td>79.0</td>
<td>85.0</td>
<td>97.0</td>
<td>99.0</td>
<td>97.0</td>
<td>89.0</td>
<td>85.0</td>
<td>87.0</td>
<td>87.0</td>
<td>87.0</td>
<td>89.0</td>
<td>91.0</td>
</tr>
<tr>
<td>105.0</td>
<td>105.0</td>
<td>105.0</td>
<td>95.0</td>
<td>81.0</td>
<td>83.0</td>
<td>95.0</td>
<td>99.0</td>
<td>89.0</td>
<td>83.0</td>
<td>85.0</td>
<td>87.0</td>
<td>89.0</td>
<td>89.0</td>
<td>91.0</td>
<td>91.0</td>
</tr>
<tr>
<td>107.0</td>
<td>97.0</td>
<td>97.0</td>
<td>87.0</td>
<td>81.0</td>
<td>93.0</td>
<td>97.0</td>
<td>91.0</td>
<td>83.0</td>
<td>87.0</td>
<td>91.0</td>
<td>89.0</td>
<td>95.0</td>
<td>95.0</td>
<td>95.0</td>
<td>95.0</td>
</tr>
<tr>
<td>109.0</td>
<td>103.0</td>
<td>97.0</td>
<td>97.0</td>
<td>83.0</td>
<td>81.0</td>
<td>95.0</td>
<td>95.0</td>
<td>87.0</td>
<td>81.0</td>
<td>87.0</td>
<td>90.0</td>
<td>91.0</td>
<td>93.0</td>
<td>93.0</td>
<td>93.0</td>
</tr>
<tr>
<td>107.0</td>
<td>101.0</td>
<td>103.0</td>
<td>91.0</td>
<td>87.0</td>
<td>97.0</td>
<td>89.0</td>
<td>83.0</td>
<td>89.0</td>
<td>91.0</td>
<td>89.0</td>
<td>91.0</td>
<td>95.0</td>
<td>95.0</td>
<td>95.0</td>
<td>95.0</td>
</tr>
<tr>
<td>107.0</td>
<td>103.0</td>
<td>99.0</td>
<td>99.0</td>
<td>89.0</td>
<td>85.0</td>
<td>95.0</td>
<td>97.0</td>
<td>87.0</td>
<td>83.0</td>
<td>87.0</td>
<td>91.0</td>
<td>89.0</td>
<td>93.0</td>
<td>93.0</td>
<td>93.0</td>
</tr>
<tr>
<td>107.0</td>
<td>105.0</td>
<td>109.0</td>
<td>95.0</td>
<td>91.0</td>
<td>99.0</td>
<td>95.0</td>
<td>89.0</td>
<td>87.0</td>
<td>89.0</td>
<td>91.0</td>
<td>91.0</td>
<td>91.0</td>
<td>95.0</td>
<td>95.0</td>
<td>95.0</td>
</tr>
<tr>
<td>109.0</td>
<td>107.0</td>
<td>103.0</td>
<td>105.0</td>
<td>91.0</td>
<td>91.0</td>
<td>99.0</td>
<td>93.0</td>
<td>87.0</td>
<td>83.0</td>
<td>87.0</td>
<td>89.0</td>
<td>91.0</td>
<td>91.0</td>
<td>93.0</td>
<td>93.0</td>
</tr>
</tbody>
</table>

(b) INTENSITY VALUE OF (7, 9)th SUBBLOCK OF THE PREVIOUS FRAME

TABLE 4-1. INTENSITY VALUE OF A TYPICAL SUBBLOCK IN THE PRESENT FRAME AND CORRESPONDING SUBBLOCK IN THE PREVIOUS FRAME. MIKE15 AND MIKE16 WERE USED AS PREVIOUS AND PRESENT FRAMES, RESPECTIVELY.
** FOR THE PREDICTION OF (7, 9)th SUBBLOCK OF MIKE16

DISPLACEMENT VECTOR: (-2, 1)

NUMBER OF ITERATION = 15

\[
\begin{array}{cccccccccccccccc}
103.0 & 101.0 & 109.0 & 105.0 & 91.0 & 89.0 & 93.0 & 99.0 & 97.0 & 87.0 & 85.0 & 87.0 & 89.0 & 87.0 & 83.0 \\
103.0 & 99.0 & 103.0 & 105.0 & 93.0 & 91.0 & 95.0 & 97.0 & 99.0 & 87.0 & 83.0 & 87.0 & 89.0 & 85.0 & 83.0 \\
105.0 & 105.0 & 107.0 & 103.0 & 89.0 & 85.0 & 93.0 & 99.0 & 99.0 & 89.0 & 85.0 & 85.0 & 87.0 & 87.0 & 87.0 \\
103.0 & 105.0 & 107.0 & 97.0 & 87.0 & 87.0 & 95.0 & 99.0 & 95.0 & 87.0 & 83.0 & 87.0 & 85.0 & 87.0 & 81.0 \\
109.0 & 107.0 & 109.0 & 107.0 & 87.0 & 81.0 & 97.0 & 101.0 & 97.0 & 89.0 & 87.0 & 85.0 & 87.0 & 87.0 & 87.0 \\
109.0 & 105.0 & 107.0 & 101.0 & 85.0 & 87.0 & 97.0 & 99.0 & 95.0 & 85.0 & 85.0 & 87.0 & 85.0 & 87.0 & 87.0 \\
105.0 & 105.0 & 109.0 & 105.0 & 87.0 & 81.0 & 93.0 & 101.0 & 93.0 & 87.0 & 87.0 & 87.0 & 89.0 & 89.0 & 89.0 \\
107.0 & 107.0 & 111.0 & 101.0 & 83.0 & 85.0 & 99.0 & 103.0 & 91.0 & 85.0 & 87.0 & 87.0 & 87.0 & 89.0 & 89.0 \\
107.0 & 107.0 & 99.0 & 101.0 & 89.0 & 79.0 & 91.0 & 95.0 & 89.0 & 85.0 & 85.0 & 89.0 & 89.0 & 91.0 & 93.0 \\
107.0 & 107.0 & 103.0 & 101.0 & 83.0 & 81.0 & 93.0 & 95.0 & 87.0 & 83.0 & 85.0 & 87.0 & 89.0 & 91.0 & 93.0 \\
107.0 & 101.0 & 99.0 & 99.0 & 87.0 & 85.0 & 97.0 & 91.0 & 83.0 & 85.0 & 89.0 & 89.0 & 91.0 & 93.0 & 93.0 \\
107.0 & 105.0 & 95.0 & 99.0 & 91.0 & 83.0 & 95.0 & 95.0 & 87.0 & 85.0 & 87.0 & 89.0 & 91.0 & 91.0 & 93.0 \\
109.0 & 107.0 & 103.0 & 107.0 & 91.0 & 91.0 & 99.0 & 91.0 & 85.0 & 85.0 & 89.0 & 91.0 & 91.0 & 91.0 & 93.0 \\
107.0 & 107.0 & 103.0 & 107.0 & 95.0 & 87.0 & 95.0 & 97.0 & 85.0 & 83.0 & 89.0 & 91.0 & 91.0 & 91.0 & 95.0 \\
107.0 & 105.0 & 103.0 & 107.0 & 97.0 & 95.0 & 99.0 & 93.0 & 89.0 & 89.0 & 89.0 & 91.0 & 91.0 & 91.0 & 93.0 \\
107.0 & 107.0 & 103.0 & 109.0 & 101.0 & 91.0 & 99.0 & 95.0 & 87.0 & 89.0 & 89.0 & 87.0 & 89.0 & 89.0 & 93.0 \\
\end{array}
\]

**TABLE 4-2. INTENSITY VALUES OF THE MOTION COMPENSATED SUBBLOCK FOR THE PREDICTION OF THE PRESENT (7,9)th SUBBLOCK OF TABLE 4-1.**
-2.0  0.0  -2.0  -2.0  0.0  6.0  8.0  0.0  -2.0  0.0  -2.0  2.0  0.0  -4.0  -8.0  
0.0  0.0  0.0  0.0  6.0  2.0  0.0  0.0  4.0  0.0  4.0  0.0  -2.0  -2.0  -2.0  0.0  -6.0  
0.0  -2.0  0.0  -4.0  -2.0  2.0  0.0  -2.0  0.0  2.0  0.0  0.0  -2.0  -2.0  -4.0  0.0  -4.0  
0.0  0.0  -4.0  -8.0  0.0  8.0  0.0  0.0  2.0  0.0  -2.0  4.0  -2.0  2.0  2.0  0.0  -2.0  
-4.0  2.0  2.0  -8.0  -8.0  8.0  6.0  2.0  2.0  2.0  0.0  0.0  0.0  -2.0  0.0  0.0  0.0  -2.0  
0.0  -2.0  0.0  -2.0  0.0  2.0  -2.0  0.0  0.0  4.0  2.0  -2.0  0.0  0.0  0.0  0.0  0.0  -2.0  
2.0  -2.0  0.0  -2.0  0.0  0.0  6.0  6.0  2.0  0.0  -2.0  0.0  0.0  -2.0  -2.0  0.0  0.0  -2.0  
0.0  -4.0  -4.0  -12.0  -6.0  8.0  0.0  0.0  0.0  2.0  0.0  2.0  0.0  0.0  4.0  2.0  -2.0  -2.0  
0.0  -2.0  0.0  -2.0  0.0  0.0  -2.0  8.0  0.0  -4.0  0.0  -2.0  0.0  0.0  0.0  0.0  -2.0  
-2.0  -6.0  0.0  -16.0  -4.0  6.0  -2.0  -2.0  -2.0  4.0  -2.0  -2.0  0.0  4.0  4.0  2.0  
0.0  -4.0  -4.0  -8.0  -6.0  -2.0  -2.0  0.0  0.0  0.0  2.0  -2.0  2.0  2.0  2.0  2.0  -2.0  
0.0  0.0  6.0  -12.0  -6.0  4.0  -4.0  -4.0  -2.0  0.0  -2.0  0.0  0.0  2.0  0.0  2.0  4.0  2.0  
2.0  0.0  0.0  -4.0  -10.0  0.0  0.0  -2.0  0.0  -2.0  2.0  0.0  0.0  2.0  0.0  2.0  0.0  2.0  

(a) PREDICTION ERROR VALUE OF (7, 9)th SUBBLOCK AFTER MOTION COMPENSATION  
MOTION DISPLACEMENT VECTOR IS (-2, 1) 

-5.5  -5.5  -9.0  -6.5  5.0  4.0  0.5  5.0  -13.5  -9.0  -0.5  4.5  12.5  14.0  0.5  1.5  
6.5  9.0  -4.5  0.5  -3.0  -4.5  -2.0  6.0  4.5  5.5  1.0  -1.5  -9.5  -6.5  -3.0  -2.5  
-2.3  0.8  2.8  -0.8  -3.3  0.8  0.8  1.3  -1.8  4.8  -3.3  0.8  -4.3  -0.8  1.3  1.3  
1.8  10.3  -2.8  1.3  -4.3  2.3  -6.8  -1.8  0.3  4.3  2.3  -0.3  -1.3  -3.3  -1.3  1.3  
1.5  7.5  -0.5  3.0  1.0  1.0  -3.0  -2.5  -1.0  -5.5  2.5  -0.9  0.0  1.5  -0.5  1.5  1.5  
-1.5  -1.0  -4.0  -2.0  2.0  1.5  -1.5  2.5  -3.0  3.0  1.0  2.5  0.0  1.0  1.0  0.5  0.5  
-1.8  -0.3  1.8  -0.3  -1.8  0.3  1.3  1.3  -0.3  -0.8  0.3  -4.3  0.3  -2.3  -0.3  2.3  0.3  
-0.8  0.8  -7.8  -3.8  0.3  -1.3  -1.8  2.3  1.3  1.3  1.8  -3.8  -3.3  -0.3  0.3  0.3  0.8  
-0.8  0.8  3.3  -0.8  -1.8  -0.3  -0.8  -1.8  2.8  -1.3  3.3  3.8  0.3  -1.8  -0.3  1.3  1.3  
-0.3  -2.3  -2.8  -2.3  -2.3  -1.3  1.3  3.8  3.3  -0.3  -1.8  1.3  -2.3  -0.8  0.8  -0.3  2.3  
-0.5  -1.0  2.0  -3.0  -1.0  0.5  -1.5  2.5  -2.5  0.5  1.5  0.0  0.5  -2.5  -0.5  -2.5  -1.0  
1.0  2.0  3.0  0.5  0.5  0.5  3.5  -3.0  1.0  -0.5  1.5  1.5  0.0  2.5  0.5  0.5  0.5  0.5  
1.0  0.3  0.3  0.3  0.3  3.3  -3.8  0.3  -0.3  -0.3  1.8  3.3  -1.8  -0.8  1.8  0.8  0.3  0.3  
-1.8  -0.8  1.3  -0.3  -2.8  -1.8  0.8  -0.8  -0.3  -1.3  -5.3  -1.3  -0.8  1.3  1.3  0.3  0.3  
-1.5  4.0  0.5  0.5  3.0  7.5  -1.0  -3.0  2.0  -6.0  0.5  5.0  -2.0  -1.0  -1.5  -1.0  
2.0  -1.0  3.5  -6.0  0.5  3.5  4.0  2.5  -11.5  -8.0  4.5  0.5  4.5  8.0  1.5  0.5  0.5  

(b) TRANSFORM COEFFICIENTS OF THE PREDICTION ERROR FOR (7, 9)th SUBBLOCK 

| TABLE 4-3 | MOTION COMPENSATED PREDICTION ERROR VALUES FOR THE PRESENT SUBBLOCK IN 
| TABLE 4-1 | AND ITS 2-D ORDERED HADAMARD TRANSFORM COEFFICIENTS. THRESHOLD OF 16 WAS 
| USED AS THRESHOLD FOR MOTION PREDICTION. (MAX. SEARCH DISTANCE P = 7) |
zero. Although, in this coding scheme, the DC coefficient no longer represents the average intensity level of the original image sub-block, we exclude it from thresholding and normalization processes because it still represents the fractional value of the average intensity value of the original image sub-block. To code the magnitude of the retained transform coefficients after quantization, 7 bits of fixed length block code is assigned for coding the DC coefficient, and the Huffman coding table shown in Table 4-5 is used for other coefficients. This Huffman coding table is derived according to the probability of each coefficient's magnitude in Fig. 4-2. Since the magnitudes of the coefficients equal to or greater than 15 rarely appeared in the quantized transform sub-block, in Table 4-5 the magnitudes of those coefficients are truncated to 15 and an additional 6 bits fixed length block code is assigned to code the magnitude of the coefficient. For transform adaptive coding a different Huffman coding table is derived from Fig.3-3. The Huffman coding table used for transform adaptive coding is shown in Table 3-1. The position of the nonzero coefficients are efficiently retained by using the chain coding of Section 3.3.1 in Chapter III and coded with the variable length instantaneous code word presented in Table 3-4. Since transform coefficients tend to cluster near the DC coefficient, the chain coding algorithm gives better performance than the conventional runlength coding. J. A. Saghri et al. [13] compared the chain coding technique with
Table 4-4. Thresholding, Normalization, and Quantization Processes for the Transform Subblock of (b) in Table 4-3.
Table 4-5. Huffman Code Table for the Transform Coefficient Amplitude of the Prediction Error

runlength coding, and their comparisons showed 10 - 30 % improvement in bit rate. As shown in the transform sub-block of the Table 4-3(b), the large magnitude coefficients tend to be scattered compared to the transform sub-block of the
original image. Since the chain coding algorithm traces the outer boundary of the non-zero coefficients within the transform sub-block, this effect will increase the overhead information to be transmitted. We found that the increased number of bits due to the scattering effect of the retained transform coefficients is negligible, especially when a low rate of transmission is desired. The chain coding processes of the actual transform sub-block are presented in Table 3-2, Table 3-3, and Table 3-4. It is found by J. A. Saghri et al. [13] that the chain link size of $L = 2$ can be a good compromise between the accuracy of the chain boundary and the amount of the overhead information to be transmitted. Thus, the chain link size of $L = 2$ was chosen for all of our simulations.

<table>
<thead>
<tr>
<th>BIT RATE</th>
<th>BITS/FRAME</th>
<th>PMSE (10^{-2}% )</th>
<th>SNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.75</td>
<td>115028</td>
<td>0.334</td>
<td>44.77</td>
</tr>
<tr>
<td>1.50</td>
<td>98993</td>
<td>0.469</td>
<td>43.29</td>
</tr>
<tr>
<td>1.25</td>
<td>82071</td>
<td>0.616</td>
<td>42.10</td>
</tr>
<tr>
<td>1.0</td>
<td>65579</td>
<td>0.923</td>
<td>40.35</td>
</tr>
<tr>
<td>0.75</td>
<td>49337</td>
<td>1.329</td>
<td>38.77</td>
</tr>
<tr>
<td>0.5</td>
<td>33023</td>
<td>2.623</td>
<td>35.81</td>
</tr>
<tr>
<td>0.25</td>
<td>16404</td>
<td>7.739</td>
<td>31.11</td>
</tr>
</tbody>
</table>

Table 4-6: Average Mean Square Error Performance of the 2-D Transform Adaptive Coding
Fig. 4-3(b - d) shows the results of the transform adaptive coding as applied to the MIKE12 image, where the coding rates are 1, 0.5, and 0.25 bits/pixel, respectively. Degradation is difficult to see at 1 bit/pixel. At 0.5 bits/pixel the degradation becomes more apparent. At 0.25 bits/pixel of (c) we can easily distinguish the sub-block structures from this image. The efficient overall MSE performance at each bit rate is given in Table 4-6. Compared with the adaptive transform coding of W. Chen and C. H. Smith [7] this coding scheme shows 6-7 dB improvements in signal to noise ratio (SNR). The reconstructed image for adaptive hybrid coding is given in Fig. 4-3(b - f). MIKE15 and MIKE16 are used as the previous and present frames, respectively. In Fig. 4-4, we cannot distinguish any degradation for rates greater than or equal to 0.5 bits/pixel. Even at the rate of 0.25 bits/pixel, quite good quality of reproduced image is presented. A poor result becomes evident only when the coding rate is reduced to 0.15 bits/pixel. The mean square performance is tabulated in Table 4-7. At the rate below 0.75 bits/pixel, the proposed adaptive hybrid coding technique shows much better performance than the transform adaptive technique.
<table>
<thead>
<tr>
<th>BIT RATE</th>
<th>BITS/FRAME</th>
<th>PMSE ($10^{-2}$)</th>
<th>SNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.75</td>
<td>114021</td>
<td>0.375</td>
<td>44.26</td>
</tr>
<tr>
<td>1.50</td>
<td>98107</td>
<td>0.509</td>
<td>42.93</td>
</tr>
<tr>
<td>1.25</td>
<td>81987</td>
<td>0.642</td>
<td>41.93</td>
</tr>
<tr>
<td>1.0</td>
<td>64035</td>
<td>1.018</td>
<td>39.92</td>
</tr>
<tr>
<td>0.75</td>
<td>48769</td>
<td>1.335</td>
<td>38.75</td>
</tr>
<tr>
<td>0.5</td>
<td>33680</td>
<td>1.959</td>
<td>37.08</td>
</tr>
<tr>
<td>0.25</td>
<td>16534</td>
<td>3.393</td>
<td>34.69</td>
</tr>
<tr>
<td>0.15</td>
<td>10008</td>
<td>4.169</td>
<td>33.80</td>
</tr>
</tbody>
</table>

Table 4-7. Average Mean Square Error Performance of the Adaptive Hybrid Coding
Fig. 4-3. Reconstructed Image by using Adaptive Transform Coding Scheme of Fig. 3-2. In (b-d), (i) shows the reconstructed image at a given bit rate and (ii) shows the error image between original and reconstructed images. Subblock of size 16 x 16 is used. Frame size = 256 x 256.

(Continued)
Fig. 4-3. Reconstructed Image by using Adaptive Transform Coding Scheme of Fig. 3-2. In (b-d), (i) shows the reconstructed image at a given bit rate and (ii) shows the error image between original and reconstructed images. Subblock of size 16 x 16 is used. Frame size = 256 x 256.
Fig. 4-4. Reconstructed Image by using Adaptive Hybrid Coding Scheme proposed in Chapter IV. In (b-f), (i) shows the reconstructed image at a given bit rate and (ii) shows the error image between present frame and reconstructed image frame. Subblock size of $16 \times 16$ and correlation coefficient $\phi = 0.96$ are used. $T = 16$ is also used as a threshold for motion estimation in the prediction loop of Fig. 4-1. (Continued)
(c) Bit Rate = 0.75 bit/pixel, PMSE = 1.335 x 10^{-2} %
SNR = 38.75 dB

(i) 
(ii)

(d) Bit Rate = 0.75 bit/pixel, PMSE = 1.959 x 10^{-2} %
SNR = 17.08 dB

(i) 
(ii)

Fig. 15: Reconstructed Image by using Adaptive Hybrid Encoder proposed in Chapter IV. In (b-f), (i) shows the original image at a given bit rate and (ii) shows the correlation between present frame and reconstructed image. An block size of 16 x 16 and correlation coefficient of 0.6 are used. T = 16 is also used as a threshold estimation in the prediction loop of Fig. 16. (Continued)
(e) Bit Rate = 0.25 bit/pixel, PMSE = 3.393 x 10^{-2} \% 
SNR = 34.69 dB

(f) Bit Rate = 0.15 bit/pixel, PMSE = 4.169 x 10^{-2} \% 
SNR = 33.80 dB

Fig. 4-1. reconstructed Image by using Adaptive Hybrid Coding as proposed in Chapter IV. In (b-f), (i) shows the reconstructed image at a given bit rate and (ii) shows the error image between present frame and reconstructed image with the block size of 16 x 16 and correlation coefficients of 0.96 are used. T = 16 is also used as a threshold for motion estimation in the prediction loop of Fig. 3-1.
CHAPTER V

SUMMARY AND CONCLUSIONS

In the interframe image sequence, image motion between frames consists of translation and rotation of objects against a fixed background in a scene, zooming and panning of the camera, and so forth. This complex interframe motion can be simplified by segmenting an image frame and by making hypothesis such that all pixels belonging to the moving area undergo uniform movement. Based on the above assumption the new algorithm, called 2-D directional search method, for interframe motion detection and motion compensation was investigated for data compression. Another statistically valid assumption made on this algorithm is that the image covariance function of images is a nonincreasing function away from the minimum distortion location. By studying the behavior of the distortion function, it is shown that this assumption is well suited for our test image data. The simulation results of using this algorithm on the real image data indicate that it gives very good estimates and contributes large redundancy reduction in the temporal dimension. It is demonstrated that this algorithm results in over 60% savings in computation as compared with the 2-D logarithmic search algorithm developed by Jain and Jain [15]. Similar performance was obtained as those using the
OTS of S. Kappagantula and K. R. Rao [21]. However, the 2-D directional method requires fewer search locations in the maximum number of searches for the motion estimation of a sub-block.

Another area of investigation in this thesis concerned the study of data compression within a single frame. Varying statistics in moving and background area indicate that an adaptive transform coding technique should be advantageous. Two approaches investigated by W. Chen and W. K. Pratt [12], and J. A. Saghri [13] were considered. The scene adaptive coder of W. K. Pratt [12] gives less complexity than other transform coding techniques, and performs quite well using mean-square error and subjective evaluation criteria. This technique uses the runlength coding for retaining the addressing information of the transform coefficients to be sent. In [13], the concept of the chain coding algorithm was applied to the adaptive transform coding technique by replacing previously employed runlength coding to the chain coding algorithm. A significant improvement in bit rate was obtained.

In this thesis we combined these two coding techniques by replacing the runlength coding of [12] with the chain coding algorithm of [13] and applied this coding method to the real image data. Excellent performance in terms of mean-square error and subjective evaluations is demonstrated through the simulations. These excellent results may be due to the optimum coefficient selection by threshold comparison
and the efficient representation of the addressing information of selected coefficients by chain coding. Comparing the performances of both the transform adaptive coding technique and the scheme of W. Chen and C. H. Smith [7], better performance in the mean-square error sense is obtained. Compared with the runlength coding, the chain coding gives considerable saving in bit rate, but it adds the complexity into the coder.

While the motion compensated interframe prediction technique results in a significant reduction in bit rate by reducing the temporal activity variations existing between frames, the 2-D transform adaptive coding operation depends only upon the information contents of a sub-block. Comparing those techniques, it is realized that the introduction of 2-D transform adaptive coding operation into the spatial dimension of the motion compensated interframe prediction technique produces a high compression efficiency by taking advantage of the redundancy in both temporal and spatial directions. A new adaptive hybrid coding is investigated in this thesis. The incorporation of motion compensated interframe prediction and transform adaptive coding techniques results in the significant improvement of performance. The plot of peak-mean-square error versus average code bits rate for both transform adaptive coding and adaptive hybrid coding is shown in Fig. 5-1. This figure shows almost equal performance up to 0.75 bits/pixel, but, at lower bit rates, the adaptive hybrid coding performs
significantly better than transform adaptive coding. This adaptive coding technique can be used for video-teleconferencing and television image.

![Plot of the Mean-Square Error Performances for both Adaptive Hybrid and Transform Adaptive Techniques.](image)

\[
\text{PMSE} = \frac{\text{MSE} \times 100}{255^2}
\]

**Fig. 5-1.** Plot of the Mean-Square Error Performances for both Adaptive Hybrid and Transform Adaptive Techniques.
REFERENCES
REFERENCES


APPENDICES
APPENDIX A

VICOM DIGITAL IMAGE PROCESSOR

The VICOM system is a digital image processing and display system. It performs four functions; alphanumeric display, raster graphic display, image display and storage, and real time image processing through a programmable table. For computer image processing, the VICOM system contains a 16 bit Motorola 68000 microprocessor, which performs dual functions: general purpose computation and operational control. In its control function, the microprocessor reads a command string entered into the VICOM and sends it to the real time image processing elements, which consist of a video controller, an image/graphic memory, a pipeline image processor, and a display controller. The Motorola 68000 microprocessor and its peripherals, connected together by a Motorola BERSAbus, directly control VICOM and implement general image processing tasks. The BERSAbus provides communication links between its internal microprocessor, an I/O controller connected to a host computer, and real time image processing elements. Through these routines, the user can operate the VICOM system from the host computer. External commands from a host computer are executed by a parallel port on the I/O controller or a serial port on the microprocessor connected via the VICOM alphanumeric
terminal. The parallel port is utilized for high speed Direct Memory Access (DMA) operation of image data and the serial port is used for transmitting commands entered from the host computer. Since VICOM uses a host computer to store or retrieve an image data file, the image data stored in the host computer can be manipulated under the user program.

The VICOM system in the ECE department at Ohio University uses a VAX/VMS 11/750 computer system as a host computer and is configured with a Firmware Operating System. One of the following modes can be assigned to operate the VICOM system.

1) VICOM Command Interpreter ON

   i) Local control mode

      The VICOM terminal prompt sign is *>. 

      The commands from VICOM alphanumeric terminal are interpreted.

   ii) Online control mode

      The VICOM terminal prompt sign is +>. 

      The commands from the host computer are interpreted.

2) VICOM Command Interpreter OFF

   i) Local control mode

      No prompt is displayed.

      No commands are interpreted.
ii) Online control mode

VICOM alphanumeric terminal acts as the host terminal

To change the mode, <ESC>0 and <ESC>I are used on the VICOM alphanumeric terminal; i.e.,

<ESC>0 : for ONLINE/OFFLINE with host computer.

<ESC>I : for Interpreter ON/OFF.

To support the communication between the VICOM system and the host computer, there is a VAX/VMS FORTRAN program called VAXHOST. This program allows the user to handle the VICOM system from the host computer by making the host computer terminal resemble VICOM's terminal. Since the VAX/VMS system assigns the serial port, which is connected to the VICOM, to TTB7, before executing the VAXHOST program, the host terminal must be allocated the TTB7 port. This can be achieved by running the command procedure called SETVICOM.COM.

After logging on the host terminal, the user must do the following steps to communicate with VICOM.

Step 1: Turn on the VICOM and the VICOM alphanumeric terminal powers.

Step 2: Make VICOM alphanumeric terminal to operate on the ONLINE control mode.
Step 3: In the host terminal, execute the following programs:

i) @SETVICOM ! To allocate host terminal TTB7 port

ii) RUN VAXHOST

** Note: The two files, SETVICOM.COM and VAXHOST.EXE, should be in your directory in order to execute these programs.

After running the VAXHOST program, the prompt sign in the host terminal is changed to an asterisk arrow (*>). With this prompt, the user can enter the VICOM commands from the host terminal. The following commands can be used to transfer an image file to the VICOM or vice versa.

A) To transfer an image data file from the host terminal to the VICOM

RIM A(file,a,b)

B) To receive an image data file from the VICOM memory

WIM A(file,a,b)

where A: image plane (1,2, or 3)

file: image file name

a: pixel size (a=2)

b: block size for data transfer (default = 4096).

Detailed descriptions of the VICOM system and the VICOM commands are given in the VICOM user's manual.
The VICOM system utilizes Two's Complement format for the definition of each pixel within an image and for all operations. Therefore, image data with decimal format or other format should be converted to the Two's Complement format before using the VICOM commands.

In this thesis, the original data set used for the simulations has the VICOM format. To process the original image data by using the simulation programs, the data is converted to the real type decimal format. The VAX/VMS FORTRAN subroutine RWIMAGE is used for this pixel format conversion operation. RWIMAGE accepts an image of size 512 x 512 or 256 x 256, and converts it from the VICOM format to the real type decimal format or vice versa. This subroutine is attached at the end of this appendix. We use the RIM command to display the original or the reconstructed image data, stored in the VAX/VMS directory, on the display monitor of the VICOM system. \( b = 16384 \) is used for fast image data transfer operation.
SUBROUTINE RWIMAGE(FILENAME, NUNIT, ARRAY, NDIM, RW)

*** ********************************************
* READ OR WRITE AN IMAGE (512x512 or 256x256) *
* FILENAME : Image File Name *
* ARRAY : 2-D array to hold Image data *
* NDIM : Dimension of the 2-D array *
* NUNIT : Unit Number for I/O *
* RW : Read/Write Control *
* 1 - Read, 0 - Write *
* *
* ********************************************

IMPLICIT NONE
INTEGER*2 IWORD
INTEGER*4 RW, NDIM, NUNIT, IR, IC, NRECL, I
CHARACTER*8 FILENAME
BYTE BUF1(256), BUF2(512), BBYTE(2)
REAL*4 ARRAY(NDIM, NDIM)
EQUIVALENCE (BYTE(1), IWORD)

NRECL = NDIM/4
IF (NDIM .EQ. 256) GO TO 100
IF (NDIM .EQ. 512) GO TO 200
PRINT *, ' *** NONSTANDARD IMAGE SIZE. CAN`T READ OR WRITE ***'
RETURN
100 IF (RW .EQ. 1) THEN
C ! Read a 256 x 256 Image Line-by-Line
C
OPEN(UNIT=NUNIT, FILE=FILENAME, RECORDTYPE=`FIXED`,
1     STATUS=`OLD`, RECL=NRECL, FORM=`UNFORMATTED`,
1     READONLY)
DO IR = 1, NDIM
READ(NUNIT) BUF1
DO IC = 1, NDIM
BBYTE(I) = BUF1(IC)
ARRAY(IR, IC) = FLOAT(IWORD)
END DO
END DO
ELSE
C ! Write a 256 x 256 Image Line-by-Line
C
OPEN(UNIT=NUNIT, FILE=FILENAME, RECORDTYPE=`FIXED`,
1     STATUS=`UNKNOWN`, RECL=NRECL, FORM=`UNFORMATTED`)
DO IR = 1, NDIM
DO IC = 1, NDIM
IWORD = IFIX(ARRAY(IR, IC)+0.5)
IF (IWORD .GT. 255) IWORD = 255
IF(IWORD .LT. 0)  IWORD = 0
   BUFI(IC) = BBYTE(1)
END DO
   WRITE(NUNIT) BUFI
END DO
   END IF
   CLOSE(UNIT=NUNIT)
   RETURN

200   IF (RW .EQ. 1) THEN
   C
   !! Read 512 x 512 Image Line-by-Line
   C
   OPEN(UNIT=NUNIT,FILE=FILENAME,RECORDTYPE='FIXED',
      STATUS='OLD',RECL=NRECL,FORM='UNFORMATTED',
      READONLY)
   DO IR = 1, NDIM
      READ(NUNIT) BUF2  !Read Image Buffer by Line
      DO IC = 1, NDIM
         IWORD = IFIX(ARRAY(IR,IC)+0.5)
         IF(IWORD .GT. 255) IWORD = 255
         IF(IWORD .LT. 0)  IWORD = 0
         BBYTE(1) = BBYTE(1)
      END DO
      WRITE(NUNIT) BUF2
   END DO
   ELSE
   !! Write a 512 x 512 Image Line-by-Line
   C
   OPEN(UNIT=NUNIT,FILE=FILENAME,RECORDTYPE='FIXED',
      STATUS='UNKNOWN',RECL=NRECL,FORM='UNFORMATTED')
   DO IR = 1, NDIM
      DO IC = 1, NDIM
         IWORD = IFIX(ARRAY(IR,IC)+0.5)
         IF(IWORD .GT. 255) IWORD = 255
         IF(IWORD .LT. 0)  IWORD = 0
         BBYTE(1) = BBYTE(1)
      END DO
      WRITE(NUNIT) BUF2
   END DO
   END IF
   CLOSE(UNIT=NUNIT)
   RETURN
   C
   END
APPENDIX B

2-D HADAMARD TRANSFORM

B.1 2-D Hadamard Transform Representation

The Hadamard transform is based on the Hadamard matrix, which is a square array of plus and minus ones whose rows and columns are orthogonal to one another. Let \([H]_N\) be the Hadamard matrix of size \(N \times N\). Then, the product of \([H]_N\) and its transpose is

\[
[H]_N[H]_N^T = N[I] \tag{B-1}
\]

where \([I]\) denotes the identity matrix. If \([H]\) is a symmetric Hadamard matrix, then (B-1) reduces to

\[
[H]_N[H]_N = N[I] \tag{B-2}
\]

The rows and columns of a Hadamard matrix can be exchanged each another without affecting the orthogonality properties of the matrix. The lowest order Hadamard matrix is the \(2 \times 2\) Hadamard matrix given by

\[
[H]_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{B-3}
\]
The order of a Hadamard matrix with order higher than 2 must be an integer multiple of 4. With one exception, at least one Hadamard matrix exists for all possible values of \( N \) up to 200. Certain Hadamard matrices of order \( N = 2^n \), where \( n \) is an integer, are related to Walsh functions, and the relationship is given by

\[
[H]_{2N} = \frac{1}{\sqrt{2N}} \begin{bmatrix}
H_N & H_N \\
H_N & -H_N
\end{bmatrix}
\]

where \([H]_{2N}\) is the Hadamard matrix of order \( 2N \) and \( H_N \) denotes the Hadamard matrix of order \( N \). Table B-1 shows Hadamard matrices, in natural form, of \( N = 4 \) and \( N = 8 \).

\[
H_4 = \frac{1}{2} \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 \\
1 & -1 & -1 & 1
\end{bmatrix}
\]

\[
H_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & -1 \\
1 & -1 & 1 & -1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1
\end{bmatrix}
\]

Table B-1. Hadamard Matrices in Natural Form for \( N = 4 \) and \( N = 8 \).
Along the each row of the Hadamard matrix the number of sign changes is often called the sequency of the row. The term "sequency" was suggested by Harmuth[29] as a generalized frequency. The row of the Hadamard matrix of (B-4) can be considered to be samples of rectangular waves with a subperiod of 1/N. These continuous functions are called Walsh functions.

Let array $U(x,y)$ be the intensity sample of an original image of size $N \times N$ and $H(u,v)$ be a 2-D Hadamard transform coefficient. Then, for symmetric Hadamard matrices of order $N = 2^n$ the 2-D Hadamard transform pair can be given by (3-1) and (3-2) as

$$H(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} U(x,y) P(x,y,u,v) \quad (B-5)$$

$$U(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} H(u,v) P(x,y,u,v) \quad (B-6)$$

where $P(x,y,u,v)$ is 2-D Hadamard transform kernel. As shown in (B-5) and (B-6) the forward and inverse transforms are identical, we can use the same algorithm used for computing $H(u,v)$ without modification to obtain $U(x,y)$, and vice versa. The 2-D Hadamard transform kernel $P(x,y,u,v)$ can be expressed as
where \( b_k(z) \) is the \( K \)-th bit in the binary expression of \( z \).

This representation of the Hadamard transform is for the Hadamard matrix in "natural" form shown in Table B-1.

Another representation exists for a Hadamard matrix in "ordered" form in which the sequency of each row is larger than the preceding row. The ordered 2-D Hadamard transform kernel is given by

\[
P(x, y, u, v) = \frac{1}{N} \sum_{i=0}^{n-1} (-1)^i \left[ b_i(x) b_i(y) + b_i(u) b_i(v) \right]
\]

(B-7)

where \( b_k(z) \) is the \( K \)-th bit in the binary expression of \( z \).

The expansion of (B-8) for the ordered Hadamard transform with \( N = 16 \) is shown in Table B-2. Since the 2-D Hadamard transform kernels are separable and symmetric, the 2-D
Table B-2. The Ordered Hadamard transform Matrix for $N = 16$. Scaling factor (1/4) is omitted in this representation. Number of sign changes are shown in right side.

| 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | 0 |
| 1 1 1 1 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 | 1 |
| 1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 1 1 | 2 |
| 1 1 -1 -1 -1 -1 1 1 1 1 1 -1 -1 -1 -1 | 3 |
| 1 -1 -1 -1 -1 1 1 1 -1 -1 -1 1 1 1 1 | 4 |
| 1 -1 -1 -1 -1 1 1 1 1 1 -1 1 1 1 1 | 5 |
| 1 1 -1 -1 -1 -1 1 1 1 -1 -1 -1 1 1 1 | 6 |
| 1 -1 -1 1 1 -1 -1 1 1 1 -1 -1 1 1 1 | 7 |
| 1 -1 -1 1 1 -1 -1 1 1 1 -1 -1 1 1 1 | 8 |
| 1 -1 1 1 1 -1 1 1 1 1 1 1 1 1 1 | 9 |
| 1 -1 -1 1 -1 1 1 1 1 1 1 1 1 1 1 | 10 |
| 1 -1 -1 1 -1 1 1 1 1 1 1 1 1 1 1 | 11 |
| 1 -1 -1 -1 -1 1 1 1 1 1 1 1 1 1 1 | 12 |
| 1 -1 1 -1 -1 1 -1 1 1 1 1 1 1 1 1 | 13 |
| 1 -1 1 -1 1 1 1 1 1 1 1 1 1 1 1 | 14 |
| 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 1 -1 | 15 |

forward and inverse transform can be obtained by successive applications of 1-D Hadamard transform algorithm. Although the Hadamard ordering has disadvantages in terms of successive doubling implementation, it gives to a simple recursive relationship for generating transform matrices required for implementation.

B.2 Statistical Properties of Hadamard Transform Samples

Let $U(x,y)$ be an intensity sample of an original image array of size $N \times N$ and $H(u,v)$ be its 2-D Hadamard transform coefficient. If the image array is wide-sense stationary, then the mean value of the image array is constant. In the
transform domain, the mean of the Hadamard transform coefficients can be represented, by using (B-5), as

\[ E[H(u,v)] = E[U(x,y)] \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P(x,y,u,v) \quad (B-9) \]

Because the Hadamard transform kernel is an orthogonal function, (B-9) becomes

\[ E[H(0,0)] = m \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P(x,y,0,0) \quad (B-10) \]

and

\[ E[H(u,v)] = 0 \quad u,v = 0 \quad (B-11) \]

where \( m \) is the mean value of the original image array. The double summation equates to \( N \) for the unitary transform processing a constant basis function. The zero sequency term (i.e., DC coefficient) can be represented as

\[ H(0,0) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} U(x,y)P(x,y,0,0) \]

\[ = mN \quad (B-12) \]

Thus, the transform domain DC energy is \( m^2N^2 \). And, for AC coefficients, transform domain AC energy is given by
\[
\text{AC energy } = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} [H^2(u,v)] - m^2 N^2
\]

\[
= N^2 \left[ E[H^2(u,v)] - m^2 \right]
\]

\[
= N^2 \sigma^2(u,v)
\]

where \( H^2(u,v) = \left[ \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} U(x,y) P(x,y,u,v) \right]^2 \)

and \( P(x,y,u,v) \) is the Hadamard transform kernel. From (B-13) we can calculate the variance of the Hadamard transform coefficients. That is,

\[
\sigma^2(0,0) = E[H^2(0,0)] - m^2 \quad \text{(B-14)}
\]

and

\[
\sigma^2(u,v) = E[H^2(u,v)], \quad u,v = 0 \quad \text{(B-15)}
\]

The transform operation consists of pointwise multiplication of a set of image values, which will take on any value between zero and the maximum allowable value. The dynamic range of the transform coefficients can be easily calculated by considering DC and AC coefficients separately. Let the maximum allowable value of \( U(x,y) \) be \( A \). Using (B-12), we can
calculate the maximum possible value for DC coefficient.

\[ H_{\text{max}}(0,0) = AN \]  \hspace{1cm} (B-16)

And the dynamic range of all transform AC coefficients remains in the range of

\[ \frac{-NA}{2} \leq H(u,v) \leq \frac{+NA}{2} \]  \hspace{1cm} (B-15)

where \( u, v = 0 \).

B.3 Computational Algorithm

In the general case unitary transform of an \( N \times N \) sub-image into an \( N \times N \) array of transform coefficients requires on the order of \( N^4 \) computational operations (multiplications and additions). For a large size sub-image, the number of calculations may be considerably larger. In order to save computation time, efficient algorithms have been developed for many unitary transforms. The key of the fast transform algorithm is its ability to subdivide the total computational task into a series of computational steps, so that the partial results from the initial step can be used repeatedly in subsequent steps.

Since 2-D Hadamard transform can be readily computed by a series of applications of the 1-D transform algorithm, we
use the 1-D ordered Hadamard transform algorithm[27], which can generate forward and inverse transform samples in "ordered" form, for our transform coder simulation. The number of input data for this algorithm should be integer power of 2. While direct computation of the forward and inverse 1-D Hadamard transform with N elements requires N(N-1) computational operations, which is counted for each plus or minus sign in expression, the algorithm used for our simulation needs only N(log₂ N) computations. The flow chart for computation of the 1-D ordered Hadamard transform is depicted in Fig. B-1 for N = 16.
Fig. B-1. Flow Graph for Forward (or Inverse) 1-D Ordered Hadamard Transform Computation for N=16. Solid line denotes the addition and dashed line denotes the subtraction.
APPENDIX C

SIMULATION PROGRAM FOR ADAPTIVE HYBRID CODING
THIS FORTRAN PROGRAM PERFORMS THE ADAPTIVE HYBRID CODING AND DECODING PROCESSES DESCRIBED IN FIG. 4 OF CHAPTER IV IN THIS THESIS. IN THIS PROGRAM, IMAGE FRAME IS FIRST SUBDIVIDED INTO SMALLER SIZE OF 16 x 16 AND THEN EACH SUB-BLOCK IS PROCESSED INDEPENDENTLY. THE VARIABLES USED IN THIS PROGRAM ARE DEFINED BELOW.

FRAME(NT,NT) = PRESENT IMAGE FRAME
PFRM(NT,NT) = PREVIOUS IMAGE FRAME
IMAGE(NT,NT) = REFERENCE IMAGE AT THE RECEIVER
PIM(NT,NT) = RECONSTRUCTED IMAGE AT THE RECEIVER
(XBN,YBN) = IMAGE SUB-BLOCK COORDINATES
SD = INITIAL SEARCH DISTANCE FOR INTERFRAME MOTION ESTIMATION
(XX,YY) = MOTION DISPLACEMENT VECTOR
CH(I,K) = CHAIN CODE FOR THE K-th CHAIN LINK OF THE I-th BOUNDARY
HV(I,K) = CHAIN CODE FOR THE K-th CHAIN LINK OF THE I-th BOUNDARY AT RECEIVER

PARAMETER (NT=256,N=256)
REAL*4 FRAME(NT,NT), IMAGE(NT,NT), PIM(NT,NT)
REAL*4 PFRM(NT,NT), IMG(16,16), RMG(NT,NT)
INTEGER X, Y, XL, YL, XBN, YBN, IUNIT, UNIT
INTEGER SD, XX, YY, IMG(16,16), tbnp, NF, tnp
INTEGER BP, F(256), CH(16,256), NC(256), C, B(600000)
INTEGER NZ(256), PIC(16,16), Z, HV(16,256), N0C(256)
INTEGER SX(16), SY(16), IX(16), IY(16), IV, JU, px, py
REAL*4 XMG(16,16), XTH, THR, PTH, ALPA
REAL*4 BER, INP, RNP, TB(16,16), TC(16,16)
CHARACTER*8 INFILE, OUTFILE
logical ok
COMMON M, NP, XBN, YBN
COMMON /BLK1/XX, YY, SD, PTH, ALPA, MCH
COMMON /BLK2/X, Y, XL, YL
common /CHK/INFILE, OUTFILE
common /res/NSA, NFA, NSB, NFB
common /prn/pr, px, py
common /mpn/ok

** INITIALIZATION PROCESS**

M = 16 ! Set Subblock Size to 16
LK = 2 ! Set chain link size
call initial(NOCN, BER, XTH, SD, PTH, ALPA)
MCH = 1 ! Choice of predictor
IUNIT = 7 ! INPUT UNIT NUMBER
OUNIT = 9 ! OUTPUT UNIT NUMBER
NP = M ** 2 ! Number of Point in the Subblock
NM = N/M
ntop = N ** 2 ! Total number of points
A = XTH
PRINT *, 'Type total frame number to be processed :'
READ *, NF
inum = (px-1)*NM + py
DO NFN = 1, NF
   BP = 0
   IF(NFN .LT. 1) THEN
      PRINT *, 'Reference image is ',INFILE
      PRINT *, 'What is the OUTPUT image file name ?'
      READ '(A)', OUTFIE
   END IF
   PRINT *, 'What is the image file name to be read ?'
   READ '(A)', INFILE
C
** READ THE INTENSITY VALUE OF THE PRESENT IMAGE FRAME
C
CALL RNIMAGE(INFILE,UNIT,FRAME,NT,1)
PRINT *, ' *** Completion of Reading Process ***'
IF(NFN .LT. 1) THEN
   IDN = 0
   DO XBN = NSA, NFA
      X = M * (XBN-1) + 1
      XL = M * XBN
      DO YBN = NSB, NFB
         Y = M * (YBN-1) + 1
         YL = M * YBN
         IDN = IDN + 1
         ok = IDN.eq.inum
         if(ok) then
            open(unit=7,file='hdat1',status='new')
            write(6,100)
            write(7,101)
            do i = X, XL
               write(7,1) (PFRM(i,j),j=Y,YL)
            end do
            write(7,11) XBN,YBN
            do i = X, XL
               write(7,1) (FRAME(i,j),j=Y,YL)
            end do
            write(7,11) XBN,YBN
            close(unit=7)
         end if
C
** INTERFRAME MOTION PREDICTION OPERATION FOR MOTION COMPENSATION
C
CALL PREDICTOR(MG,FRAME,PFRM,NT,N)
INP = A
C
** THE 2-D TRANSFORM OPERATION OF THE SUB-BLOCK
C
CALL XFORM(MG,XMG)
if(ok) then
  write(6,100)
  open(unit=7,file='hdat3',status='new')
  do i = 1, M
    write(7,1) (MG(i,j),j=1,M)
  end do
  write(7,33) XBN,YBN,XV,YV
  do i = 1, M
    write(7,1) (XMG(i,j),j=1,M)
  end do
  write(7,44) XBN,YBN
  close(unit=7)
end if

isum = 0
DO I = 1, M
  DO J = 1, M
    isum = isum + 1
    TA = XMG(I,J)
    IF(isum .EQ. 1) THEN
      TC(I,J) = TA
      IF(ok) TB(I,J) = TA
    ELSE
      TB(I,J) = THRESHOLD(TA,XTH,1) ! THRESHOLD PROCESS
      TC(I,J) = TB(I,J)/INP ! NORMALIZATION PROCESS
    END IF
    IMG(I,J) = NINT(TC(I,J)) ! QUANTIZATION PROCESS
  END DO
END DO
END if

if(ok) then
  write(6,100)
  open(unit=7,file='hdat4',status='new')
  do i = 1, M
    write(7,1) (TB(i,j),j=1,M)
  end do
  write(7,55) XTH
  do i = 1, M
    write(7,1) (TC(i,j),j=1,M)
  end do
  write(7,66) INP
  do i = 1, M
    write(7,2) (IMG(i,j),j=1,M)
  end do
  write(7,77)
  close(unit=7)
end if

** FEEDBACK OPERATION TO FORM THE REFERENCE IMAGE FOR THE INTERFRAME
** MOTION PREDICTION OF THE NEXT FRAME.

CALL FBACK(IMG,PFRM,RMG,NT,INP,XTH,XV,YV,ALPA,IDN,1)
** CHAIN CODING AND 1-D MAPPING PROCESSES OF THE NON-ZERO TRANSFORM COEFFICIENTS WITHIN THE SUB-BLOCK

CALL CHAIN(IMG,F,C,NC,NNB,LK,CH,IX,IY,1)
    if(ok) write(6,100)

** ENCODING AND MULTIFLEXING PROCESSES

CALL CODE(F,LK,C,B,CH,NC,NNB,BP,IX,IY,XV,YV,M,NP,1)

END DO

print *, ** CODING COMPLETED **

TBP = float(BP)/float(tnop)

tnbp = BP

if(NOCN .eq. 1) call CHANNEL(B,BP,BER)

ISM = 0

I = 0

write(6,100)

print *, ** START DECODING **

DO WHILE(ISM .LT. BP)

I = I + 1

ok = I.eq.inum

** DEMULTIFLEXING AND DECODING PROCESSES

CALL CODE(NZ,LK,2,B,HV,NOC,NOB,ISM,SX,SY,IV,JV,M,NP,0)

RNP = A

** CHAIN DECODING AND 2-D MAPPING PROCESSES TO RECONSTRUCT THE TRANSFORM SUB-BLOCK.

CALL CHAIN(PIC,NZ,2,NOC,NOB,LK,HV,SX,SY,0)

** IMAGE RECONSTRUCTION PROCESS

CALL FBACK(PIC,IMAGE,PIM,NT,RNP,XTH,IV,JV,ALPA,I,0)

END DO

print *, ** DECODING COMPLETED **

** THE RECONSTRUCTED IMAGE WRITING PROCESS FOR VICOM FORMAT

CALL RWIMAGE(OUTFILE,OUNIT,PIM,NT,0)

print *, ** END OF OUTPUT IMAGE WRITING PROCESS **

sum = 0.

do ia = 1, N
    do ib = 1, N
        PFEM(ia,ib) = RMG(ia,ib)
        IMAGE(ia,ib) = PIM(ia,ib)
    end do
end do
write(6,88) tnbp
write(6,99) TBP
print *, "OUTPUT IMAGE FILE NAME : '',OUTFILE"
write(6,100)
ELSE
  do ia = 1, NT
    do ib = 1, NT
      PFRM(ia,ib) = FRAME(ia,ib)
      IMAGE(ia,ib) = PFRM(ia,ib)
    end do
  end do
END IF
END DO
C
1  format(5x,16f6.1)
2  format(5x,16i6)
3  format(5x,i3,3x,16f6.1)
11 format(/,5x, ' ** INTENSITY VALUE OF (''i2'', ''i2'')th ','
  1 ' SUBBLOCK OF THE PREVIOUS FRAME',///)
33 format(/,5x, ' ** PREDICTION ERROR VALUE OF (''i2'', ''i2'')th ','
  1 'SUBBLOCK AFTER MOTION COMPENSATION',//,9x,'MOTION ','
  1 'DISPLACEMENT VECTOR IS (''i2'', ''i2''),///)
44 format(/,5x, ' ** TRANSFORM COEFFICIENTS OF THE PREDICTION ','
  1 'ERROR FOR (''i2'', ''i2'')th SUBBLOCK',///)
55 format(/,5x, ' ** TRANSFORM COEFFICIENTS AFTER THRESHOLDING ','
  1 'PROCESS',//,9x,'THRESHOLD VALUE = '',f4.1,///)
66 format(/,5x, ' ** TRANSFORM COEFFICIENTS AFTER NORMALIZATION ','
  1 'PROCESS',//,9x,'NORMALIZATION FACTOR = '',f4.1,///)
77 format(/,5x, ' ** TRANSFORM COEFFICIENTS AFTER QUANTIZATION ','
  1 'PROCESS')
88 format(10x, 'TOTAL NUMBER OF BITS :',i8)
99 format(10x, 'TRANSMISSION BIT RATE :',f8.2)
100 format(1h1)
101 format(/,///)
C
END
SUBROUTINE PREDICTOR(U,UR,PFRM,NT,NS)

*******************************************************************************
** This FORTRAN Program performs motion compensated prediction operation by using the two subroutine MATCH_NEW and MATCH_OLD. The prediction errors are returned to the calling program by using U(i,j). **
** ** CHOICE OF THE SUBROUTINE **
** CH = 1 : MATCH_NEW **
** CH = 2 : MATCH_OLD **
** ** UR(M,M) = PRESENT IMAGE SUBBLOCK **
** PFRM(N,N) = PREVIOUS IMAGE **
** N = IMAGE FRAME SIZE **
** M = SUBBLOCK SIZE **
** P = SEARCH DISTANCE FOR SUBBLOCK **
** TH = PREDETERMINED THRESHOLD **
** (IX,IY) = DISPLACEMENT VECTOR FOR K-th SUBBLOCK **
*******************************************************************************

INTEGER P,CH,AX,AY,BX,BY,XBN,YBN
REAL*4 U(MS,MS),UR(NT,NT),PFRM(NT,NT)
REAL*4 nifv,TH,a,pf(16,16)
LOGICAL yes
COMMON MS,NN,XBN,YBN
COMMON /BLK1/IDX,IDY,P,TH,ALPA,CH
COMMON /BLK2/AX,AY,BX,BY
COMMON /mpn/yes

SUM = 0.
DO I = AX, BX
   DO J = AY, BY
      a=UR(I,J)-PFRM(I,J)
      SUM = SUM + a**2
   END DO
END DO
DM = SUM/FLOAT(NN)
nifv = DM
IX = 0
IY = 0
IC = 1
IF(DM.GE.TH) THEN
   IF(CH.EQ.1) THEN
      CALL MATCH_NEW(UR,PFRM,NT,NS,P,TH,DM,IX,IY,IC)
   ELSE
      CALL MATCH_OLD(UR,PFRM,NT,NS,P,TH,DM,IX,IY,IC)
   END IF
END IF
END IF
if(yes) then
  write(6,100)
  write(6,1) XBN,YBN
  print *, ** NO MOTION COMPENSATED INTERFRAME VARIANCE : ',nifv
  print *, ** MOTION COMPENSATED INTERFRAME VARIANCE : ',DM
  write(6,2) IX,IY,IC
end if

tsx = 0.
IDX = IX
IDY = IY
DO I = AX, BX
   II = I - AX + 1
   NX = I + IDX
   DO J = AY, BY
      JJ = J - AY + 1
      NY = J + IDY
      pf(II, JJ) = ALPA*PFRM(NX,NY)
      U(II, JJ) = UR(I, J) - pf(II, JJ)
      tsx = tsx + U(II, JJ)**2
   END DO
END DO
amce = tsx/float(NN)
if(amce.lt.TH) then
   do i = 1, MS
      do j = 1, MS
         U(i, j) = 0.
      end do
   end do
end if
if(yes) then
   do i = 1, MS
      write(6,3) (pf(i, j), j=1, MS)
   end do
   write(6,33) XBN,YBN
   do i = AX, BX
      write(6,3) (UR(i, j), j=AY, BY)
   end do
   write(6,44) XBN,YBN
end if

format(5x, ** FOR THE PREDICTION OF ('',i2,'','\text{'}), 'i2, \text{"}th SUBBLOCK',/)
format(/,5x, ** DISPLAEMENT VECTOR : ('',i2,',$\text{"}','i2,'$\text{"}'),/)
format(5x,1d16.1)
format(/,5x, ** MOTION COMPENSATED PREVIOUS SUBBLOCK FOR',)
format(/,5x, ** THE ('',i2,'','\text{'}), 'i2,'$\text{"}$ \text{'}\text{'},'i2,'$\text{"}'),/)
format(/,5x, ** INTENSITY VALUES OF THE ('',i2,'','\text{'}), 'i2,'$\text{"}$ \text{'}\text{'},'i2,'$\text{"}'),/)
format(1h1/)
END
SUBROUTINE MATCH_NEIL(UR,PFRM,NP,TH,DM,DUX,DVY,NC)

******************************************************************************
* This FORTRAN subroutine performs new 2-D search method *
* called "2-D Directional Search (DS)," which find the *
* direction of the minimum difference between present *
* and previous subblock by using directional search *
* algorithm. *
* UR(N,N) = PRESENT IMAGE FRAME *
* PFRM(N,N) = PREVIOUS IMAGE FRAME *
* TH = PREDETERMINED THRESHOLD *
******************************************************************************

INTEGER S,P,X,Y,DUX,DVY,PX,PY,XB,YB
REAL*4 UR(NT,NT),PFRM(NT,NT),MDM,MDV
COMMON M,NP,XB,YB
COMMON /BLK2/X,Y,PX,PY
LOGICAL COND1,COND2

S = 2 * P
SPS = FLOAT(P)
SP = ALOG10(SPS)/ALOG10(2.0)
ITS = INT(SP)
LS = ITS + 1
NS = 0
L = 0
DO WHILE (L .LT. LS)
  L = L + 1
  INS = 3 * L
  S = S/2
  DO WHILE (NS .LT. INS)
    NS = NS + 1
    MG = INS - NS
    IF (MG .NE. 0) THEN
      NSP = 2
    ELSE
      NSP = 1
    END IF
    DO NL = 1, NSP
      NZI = S*(-1)**MOD(NL,2)
      IF (MG .EQ. 2) THEN
        JX = 0
        JY = NZI
      ELSE IF (MG .EQ. 1) THEN
        JX = NZI
        JY = 0
      ELSE
        JX = MI
        JY = MJ
      END IF
      JXI = JX + DUX
      JYI = JY + DVY
IF (N .EQ. NT) THEN
  IPX = X + JXI
  IPY = Y + JYI
  ITX = PX + JXI
  ITY = PY + JYI
  COND1 = (IPX .LT. 1) .OR. (ITX .GT. NT)
  COND2 = (IPY .LT. 1) .OR. (ITY .GT. NT)
  IF (COND1) JXI = DVX
  IF (COND2) JYI = DVY
END IF
SUM = 0.0
DO I = X, PX
  LX = I + JXI
  DO J = Y, PY
    LY = J + JYI
    MMD = UR(I,J) - PFRM(LX,LY)
    SUM = SUM + MMD**2
  END DO
END DO
SUM = SUM/FLOAT(NP)
NC = NC + 1
IF (NL .EQ. 1) THEN
  MDV = SUM
  MX = JX
  MY = JY
ELSE
  IF (MDV .GT. SUM) THEN
    MDV = SUM
    MX = JX
    MY = JY
  END IF
END IF
END DO
IF (MG .EQ. 2) THEN
  GG = MDV
  MXI = MX
  MYI = MY
ELSE IF (MG .EQ. 1) THEN
  IF (GG .LT. MDV) THEN
    IF (GG .LT. DM) THEN
      DM = GG
      KX = MXI
      KY = MYI
      IF (GG .LT. TH) THEN
        NS = INS
        L = LS
      ELSE
        MI = MX
        MJ = MYI
      END IF
    END IF
  END IF
END IF
ELSE
    NS = INS
    KX = 0
    KY = 0
END IF
ELSE
    IF (MDV .LT. DM) THEN
        DM = MDV
        KX = MX
        KY = MY
        IF (DM .LT. TH) THEN
            NS = INS
            L = LS
        ELSE
            MI = MX
            MJ = MYI
        END IF
    ELSE
        NS = INS
        KX = 0
        KY = 0
    END IF
END IF
ELSE
    IF (MDV .LT. DM) THEN
        DM = MDV
        KX = MX
        KY = MY
        IF (DM .LT. TH) L = LS
    END IF
END IF
END DO
DVX = DVX + KX
DVY = DVY + KY
END DO
C
END
SUBROUTINE MATCH_OLD(UR,PFRM,NFS,N,P,TH,DM,DVX,DVY,NC)

******************************************************************************
* ** This FORTRAN subroutine performs 2-D direct search called "One at a Time Search", which *
* is to adjust one variable at a time for *
* minimum distortion. This procedure repeated *
* until no further improvement can be achieved. *
* *
* ** UR(N,N) = Present image frame *
* PFRM(N,N) = Previous image frame memory *
* S = Initial Search Distance *
* P = Max. Search Distance *
******************************************************************************

INTEGER S,P,X,Y,DVX,DVY,PX,Py
REAL*4 UR(N,N),PFRM(N,N),MMD,MDV
LOGICAL COND1,COND2
COMMON M,NP,XB,YB
COMMON /BLK2/X,Y,PX,Py

SPS = FLOAT(P)
SP = ALOG10(SPS)/ALOG10(2.0)
ITS = INT(SP)
LS = ITS + 1
S = 2*P
KX = 0
KY = 0
NS = 0
L = 0
DO WHILE (L.LT.LS)
  L = L + 1
  INS = 2*L
  S = S/2
  DO WHILE (NS.LT.INS)
    NS = NS + 1
    MG = INS - NS
    NSP = 2 ** (MG + 1)
    DO NL = 1, NSP
      NC1 = NL + 1
      NC2 = MOD(NL,2)
      NC3 = MOD(NC1,2)
      NCZ = S * (-1) ** MOD(NL,3)
      IF(MG.NE.0) THEN
        JX = NC3 * NCZ
        JY = NC2 * NCZ
      ELSE
        IF(KX.EQ.0) THEN
          JX = NCZ
          JY = KY
        ELSE
          JX = KX
        END IF
      END IF
    END DO
  END DO
END DO
JY = NCZ
END IF
END IF
JXI = JX + DVX
JYI = JY + DVY
IF (N .EQ. NFS) THEN
  IPX = X + JXI
  IPY = Y + JYI
  ITX = PX + JXI
  ITY = PY + JYI
  COND1 = (IPX .LT. 1) .OR. (ITX .GT. N)
  COND2 = (IPY .LT. 1) .OR. (ITY .GT. N)
  IF (COND1) JXI = DVX
  IF (COND2) JYI = DVY
END IF
SUM = 0.0
DO I = X, PX
  LX = I + JXI
  DO J = Y, PY
    LY = J + JYI
    MMD = UR(I,J) - PFRM(LX,LY)
    SUM = SUM + MMD**2
  END DO
END DO
SUM = SUM/FLOAT(NP)
NC = NC + 1
IF (NL .EQ. 1) THEN
  MDV = SUM
  MX = JX
  MY = JY
ELSE
  IF (SUM .LT. MDV) THEN
    MDV = SUM
    MX = JX
    MY = JY
  END IF
END IF
END IF
END DO
IF (DM .LT. MDV) THEN
  NS = INS
ELSE
  DM = MDV
  KX = MX
  KY = MY
  IF (DM .LT. TH) THEN
    NS = INS
    L = LS
  END IF
END IF
END IF
END DO
DVX = DVX + KX
DVY = DVY + KY
END DO
END
SUBROUTINE XFORM(V,VH)

* *
* This FORTRAN subroutine calculates 2-D HADAMARD transform coefficients *
* by using the subroutine FWHT which can generate 1-D ordered HADAMARD transform *
* coefficients. The 1-D mapping of the 2-D image data is performed within this subroutine. *
* For 1-D case, the choice of forward and inverse transformation is given by CHT, i.e., *
* CHT = 1 : FORWARD TRANSFORM *
* CHT = 2 : INVERSE TRANSFORM *
* *
**CHT******************

INTEGER XB,YB,CHT
REAL*4 V(M,M),VH(M,M),H(5,256)
COMMON M,NP,XB,YB

CHT = 1
SP = FLOAT(M)
SL = ALOG10(SP)/ALOG10(2.)
M1 = NINT(SL)
MM = M1 + 1
ISUM = 0
DO I=1,M
   DO J=1,M
      ISUM = ISUM + 1
      H(1,ISUM) = V(I,J)
   END DO
   CALL FWHT(H,MM,M,I,CHT)
END DO
ISUM = 0
DO K=1,M
   DO L = K, NP, M
      ISUM = ISUM + 1
      H(1,ISUM) = H(MM,L)
   END DO
END DO
ISUM = 0
DO J=1,M
   CALL FWHT(H,MM,M,J,CHT)
   DO I=1,M
      ISUM = ISUM + 1
      VH(I,J) = H(MM,ISUM)/FLOAT(M)
   END DO
END DO
SUBROUTINE FWHT(IM, MS, NP, IS, ICH)

* This FORTRAN subroutine performs ordered HADAMARD transform and inverse transform, and produces the transform coefficients in the order of the increasing sequence of the WLSH functions represented by the row of a symmetric HADAMARD matrix for which the sequency of each row is larger than the sequency of the proceeding row. In this subroutine input data are read by IM(1,J) and output data are returned to IM(MS,J).

* NP = 2 ** MP : Number of Sample Points
* L = Index of Array used to compute values for Array L2.
* NCG = Number of computation groups for Array being computed.
* J = Index of Point in Array being computed.

REAL IM(MS,NP*IS)

NN = NP/2
MP = MS - 1
N1 = NP * (IS - 1)
DO L = 1, MP
   L1 = L - 1
   L2 = L + 1
   IT = N1
   NR = 2 ** L1
   NC = 2 * NR
   NCG = NP/NC
   DO K = 1, NCG
      IP = IT + 1
      IT = IT + NR
      IZI = NC * (K - 1) + N1
      IZJ = IZI + NC + 1
      DO J = IP, IT
         IZI = IZI + 1
         J1 = J + NN
         IM(L2,IZI) = IM(L,J) + IM(L,J1)
         IZJ = IZJ - 1
         IF (ICH .EQ. 2) THEN
            IM(L2,IZJ) = ABS(IM(L,J) - IM(L,J1))
         ELSE
            IM(L2,IZJ) = IM(L,J) - IM(L,J1)
         END IF
      END DO
   END DO
END DO
END DO
END
SUBROUTINE chain(A,F,ISUM,NC,NB,L,CH,IX,IY,CHOICE)

********************************************************************
* ** chain coding for the transform coefficients **
* * A = transform coefficient H(u,v) *
* MS = buffer size [M + 2(L+1)] *
* L = resolution of the chain link *
* W = buffer array for chain coding *
* NC = number of chain code for K-th boundary *
* NN = number of points within the subblock *
* CH(K,I) = chain code for the I-th link of *
* the K-th boundary *
* (MX,MY) = xy-coordinates of the upper left corner pixel for the subblock *
* *
********************************************************************

INTEGER A(M,M),W(20,20),F(NN),CHOICE
INTEGER NC(NN),CH(M,NN),IX(M),IY(M),XB,YB
INTEGER FINDX,FINDY
character*8 in,out
logical yes
COMMON M,NX,NX,YB
COMMON /CHK1/MS,ICR,ICR1
common /CHK/in,out
common /mpn/yes

!! forming the buffer array

ICR = 100000
ICR1 = 100
MS = M+2*L
DO I = 1, MS
   DO J = 1, MS
      W(I,J) = 0
   END DO
END DO
IF(CHOICE.EQ.1) THEN
   DO I = 1, M
      II = I + L
      DO J = 1, M
         JJ = J + L
         IF(A(I,J).NE.0) W(II,JJ)=1
      END DO
   END DO
ELSE IF(yes) THEN
   write(6,99)
   write(6,98)
   do i = 1, MS
      write(6,1) (W(i,j),j=1,MS)
   end do
write(6,11) XB,YB,in
end if
CALL STP(W,L,NB,NC,IX,IY,CH)
else
if(NB .NE. 0) then
   do k = 1, NB
      idb = icr * k
      kf = idb
      nx = ix(k)
      ny = iy(k)
      jx = nx
      jy = ny
      ncc = nc(k)
      do i = 1, ncc
         lp = ch(k,i)
         jsum = 0
         do while(jsum .lt. l)
            jsum = jsum + 1
            kx = findx(jx,lp,jsum)
            ky = findy(jy,lp,jsum)
            kf = kf + 1
            iab = w(kx,ky)
            if(iab .gt. idb) then
               ikf = (kf-idb) * icri
               w(kx,ky) = iab + ikf
            else
               w(kx,ky) = kf
            end if
         end do
         jx = kx
         jy = ky
      end do
      call asgn(w,ch,l,k,kf,ncc,nx,ny)
   end if
end if
 Call map(A,F,W,ISUM,L,CHOICE)
format(/11/)
98 format(1h1)
1 format(5x,20i3)
2 format(5x,20a8)
11 format(/5x,** BUFFER ARRAY OF (',i2,','i2,')th SUBBLOCK',
     'OF',a8//)
22 format(/5x,** BUFFER ARRAY OF THE RECONSTRUCTED',
     'SUBBLOCK OF',a8)
end
SUBROUTINE STP(V,L,NB,NC,IS,JS,CH)

**************************************************************************
* * ** V(MW,MW) = BUFFER ARRAY CONTAINING X-FORM SUBBLOCK H(U,V) *
* MB = SIZE OF THE X-FORM SUBBLOCK *
* L = RESOLUTION OF THE CHAIN LINK *
* NB ; DETERMINES THE NUMBER OF THE BOUNDARY *
* CH(K,I) IS THE CHAIN CODE FOR THE I-th LINK OF THE K-th BOUNDARY. *
**************************************************************************

INTEGER V(MW,MW), XB,YB
INTEGER*4 CH(MB,NN), NC(NN), IS(MB), JS(MB)
character*8 in, out
LOGICAL FOUND, yes
COMMON MB, NN, XB,YB
COMMON /CHK1/MW, ICR, ICR1
common /CHK/in, out
common /mpn/yes

K = 0
MT = MB + L
I = L
DO WHILE (I .LE. MT)
   J = I + 1
   DO WHILE (J .LE. MT)
      IG = V(I,J)
      FOUND = (IG .EQ. 1)
      IF (FOUND) THEN
         K = K + 1
         IF (L .EQ. 1) THEN
            IS(K) = I - 1
            JS(K) = J
         ELSE
            IS(K) = I - 1
            JS(K) = J - 1
         END IF
      END IF
      ISI = IS(K)
      JSJ = JS(K)
      IDK = ICR * K
      IDB = IDK
      CALL TRACER(V, CH, L, K, IDB, ISI, JSJ)
   END DO
   I = I + 1
END DO

* Calculate the number of chain code for the K-th boundary
NCC = (IDB-IDK)/L
NC(K) = NCC
CALL ASGN(V,CH,L,K,IDB,NCC,ISI,JSJ)
if(yes) then
  write(6,100)
  do ia = 1, MW
    write(6,1) (V(ia,ib),ib=1,MW)
  end do
  write(6,2) XB,YB
end if
END IF
END DO
END DO
NB = K
1  format(3x,20i6)
2  format(6x,5x,'** ASGN PROCESS FOR (',i2,',',i2,')th'
     1     ' SUBBLOCK',///)
100 format(1h1,///)
C
END
SUBROUTINE TRACER(A, CL, LI, KF, IX, IY)

******************************************************************************
*     BOUNDARY TRACING ALGORITHM FOR CHAIN CODE                           *
******************************************************************************
* ** A(MS, MS) = BUFFER ARRAY                                              *
* CL(K, MS) = CHAIN CODES FOR THE K-th                                    *
* LI = RESOLUTION OF THE CHAIN LINK                                       *
* KIDF = IDENTIFIER OF THE K-th BOUNDARY                                  *
* (IX, IY) = STARTING POINT OF THE K-th                                    *
******************************************************************************

INTEGER A(MS, MS), XB, YB
INTEGER CL(MB, NN), EVT, FINDX, FINDY, SEARCHX, SEARCHY
INTEGER CHECK
LOGICAL NOK1, NOK2
COMMON MB, NN, XB, YB
COMMON /CHK1/ MS, ICR, ICR1

ID = 0
LP = 0
IDF = KF
JX = IX
JY = IY - LI
DO WHILE (.NOT.(JX .NE. IX) .OR. (JY .NE. IY))
   ID = ID + 1
   IL = LP + 5 ! CALCULATE THE LAST CHAIN CODE
   EVT = MOD(IL, 8) ! RELATED TO THE NEXT CHAIN LINK
   IL1 = MOD(LP, 2)
   IF (IL1 .EQ. 0) THEN
      IE = LP + 2 ! Find the FIRST NEXT CHAIN
   ELSE
      IE = LP + 3 ! LINK candidate to determine
   END IF
   LN = MOD(IE, 8)
   EVT = EVT - 1
   IF (EVT .LT. 0) EVT = 8 + EVT
   DO WHILE (LN .NE. EVT)
      IF (ID .EQ. 1) THEN
         KX = JX
         KY = JY
      ELSE
         KX = FINDX(JX, LN, LI)
         KY = FINDY(JY, LN, LI)
      END IF
      DO WHILE ((KX .LE. 0) .OR. (KY .LE. 0)
      OR. (KX .GT. MS) .OR. (KY .GT. MS))
LN = LN - 1
IF (LN .LT. 0) LN = LN + LN
KX = FINDX(JX, LN, LI)
KY = FINDY(JY, LN, LI)
END DO
II = 1
I = 0
IF,ID ,NE. 1) THEN
IF(KX .EQ. IX .AND. KY .EQ. IY) THEN
ICK = 2
I = LI
END IF
END IF
DO WHILE (I .LT. LI)
I = I + 1
J = 0
DO WHILE (J .LT. I)
J = J + 1
IEX = SEARCHX(JX, LN, I, J)
IEY = SEARCHY(JY, LN, I, J)
NOK1 = (IEX .GT. MS) .OR. (IEY .GT. MS)
NOK2 = (IEX .LE. 0) .OR. (IEY .LE. 0)
IF(NOK1 .OR. NOK2) THEN
ICK = 1
J = LI
I = LI
ELSE
IAB = A(IEX, IEY)
IF(IAB .GT. IDF) THEN
IF(J .EQ. LI) THEN
ICK = II
ELSE
LCK = MOD(LN, 2)
IBK = CHECK(IAB, ICR, ICR1, MB, NN, LN, LI, CL)
IF(IBK .LE. 2) THEN
ICK = 1
IF(LCK .EQ. 0 .AND. IBK .EQ. 1) ICK = 0
J = LI
I = LI
ELSE
ICK = II + 1
II = ICK
END IF
END IF
ELSE IF(IAB .EQ. 1) THEN
ICK = II + 1
II = ICK
ELSE
ICK = II
END IF
END IF
END DO
END DO
IF(ICK .EQ. 0) THEN
    LN = LN - 2
    IF(LN .LT. 0) LN = 8 + LN
ELSE IF(ICK .EQ. 1) THEN
    LN = LN - 1
    IF(LN .LT. 0) LN = 8 + LN
ELSE
    CL(K,ID) = LN
    LN = EVT
END IF
END DO
LP = CL(K,ID)

C ** Identification of the Present CHAIN LINK C
ISUM = 0
DO WHILE (ISUM .LT. LI)
    ISUM = ISUM + 1
    KX = FINDX(JX,LP,ISUM)
    KY = FINDY(JY,LP,ISUM)
    KF = KF + 1
    IAB = A(KX,KY)
    IF(IAB .GT. IDF) THEN
        IKF = (KF-IDF)*ICR1
        A(KX,KY) = IAB + IKF
    ELSE
        A(KX,KY) = KF
    END IF
END DO
JX = KX
JY = KY
END DO
C
END
FUNCTION SEARCHX(IP,LS,I,J)
C
IMPLICIT NONE
INTEGER LV1,LV2,IP,LS,IA,I,J,SEARCHX
C
LV1 = MOD(LS,4)
LV2 = 4 - LS
C
IF (LV1 .GT. 1) THEN
  IF (LV2 .LT. 0) THEN
    IA = I
  ELSE
    IA = -I
  END IF
ELSE IF (LV1 .EQ. 0) THEN
  IF (LV2 .GT. 0) THEN
    IA = J
  ELSE
    IA = -J
  END IF
ELSE
  IF (LV2 .LT. 0) THEN
    IA = I - J
  ELSE
    IA = J - I
  END IF
END IF
C
SEARCHX = IP + IA
C
END
FUNCTION SEARCHY(IP,LS,I,J)

IMPLICIT NONE
INTEGER LV1,LV2,IP,LS,IA,I,J,SEARCHY

LV1 = MOD(LS,4)
LV2 = 4 - LS

IF (LV1 .LE. 1) THEN
  IF (LV2 .GT. 0) THEN
    IA = I
  ELSE
    IA = -I
  END IF
ELSE IF (LV1 .EQ. 2) THEN
  IF (LV2 .GT. 0) THEN
    IA = J
  ELSE
    IA = -J
  END IF
ELSE
  IF (LV2 .LT. 0) THEN
    IA = I - J
  ELSE
    IA = J - I
  END IF
END IF

SEARCHY = IP + IA

END
FUNCTION FINDX(XS,LP,L)

IMPLICIT NONE
INTEGER XS,LP,L,FINDX
LOGICAL COND

COND = (LP .EQ. 0 .OR. LP .EQ. 4)

IF (COND) THEN
   FINDX = XS
ELSE IF (LP .LT. 4) THEN
   FINDX = XS - L
ELSE
   FINDX = XS + L
END IF

END

FUNCTION FINDY(YS,LP,L)

IMPLICIT NONE
INTEGER YS,LP,L,FINDY
LOGICAL TEST1, TEST2

TEST1 = (LP .EQ. 2 .OR. LP .EQ. 6)
TEST2 = (LP .GT. 2 .AND. LP .LT. 6)

IF (TEST1) THEN
   FINDY = YS
ELSE IF (TEST2) THEN
   FINDY = YS - L
ELSE
   FINDY = YS + L
END IF

END
FUNCTION CHECK(IAP, ICR, ICR1, MB, NN, LN, LINK, CL)

C
IMPLICIT NONE
INTEGER IAP, ICR, ICR1, MB, NN, MG, K, I, J, CNT
INTEGER LINK, LN, CL(MB,NN), LP, CHECK
REAL*4 CK

C
MG = MOD(IAP, ICR)
K = (IAP-MG)/ICR
IF(MG.GT. ICR1) THEN
   CK = FLOAT(MG/ICR1)
   MG = INT(CK)
END IF
J = MOD(MG, LINK)
IF(J.NE. 0) THEN
   CK = FLOAT(MG/LINK)
   I = INT(CK) + 1
ELSE
   I = MG/LINK
END IF
LP = CL(K, I)
CNT = LP - LN
IF(CNT.LE. 0) CNT=8+CNT
CHECK = CNT

C
END
SUBROUTINE ASGN(A,CL,L,K,RK,NN,XI,YI)

******************************************************************************
*
* ** ASSIGNMENT ALGORITHM FOR THE COEFFICIENTS *
* ** INSIDE THE K-th BOUNDARY JUST CHAIN CODED *
* ** *
* ** THIS FORTRAN PROGRAM REPLACES THE ORIGINAL *
* ** VALUE OF ALL POINTS INSIDE THE BOUNDARY *
* ** JUST GENERATED TO ( R * K ). *
* ** *
* ** A(N,N) = BUFFER ARRAY CONTAINING X-FORM *
* ** SUBBLOCK *
* ** CL(K,NN) = CHAIN CODES FOR K-th BOUNDARY *
* ** RK = IDENTIFIER OF THE K-th BOUNDARY *
* ** L = RESOLUTION OF THE CHAIN LINK *
* ** (IX, IY) = STARTING POINT FOR ASSIGNMENT *
* ** *
******************************************************************************

INTEGER A(N,N), RK, CL(M,NP), XB, YB
LOGICAL COND, TEST1, TEST2, TEST3, TEST4
LOGICAL TEST5, TEST6
COMMON M,NP,XB,YB
COMMON /CHK1/N,ICR,ICRI

IDF = RK - NN*L
IDF1 = IDF + ICR1
ISUM = 0
IX = XI + 1
DO WHILE (IX .LT. N)
  IX = IX + 1
  IY = 0
  NIS = 0
  ISUM = ISUM + 1
  IF(ISUM .EQ. 1) IY=IY-1
  DO WHILE (IY .LT. N)
    IY = IY + 1
    NPK = MOD(NIS,2)
    IVP = A(IX,IY)
    IF(IVP .GT. IDF) THEN
      NCL = MOD(IVP,ICR)
      NCP = NCL
      JSUM = 0
      DO WHILE(JSUM .LT. 2)
        JSUM = JSUM + 1
        IF(NCP .GT. ICR1) THEN
          CK = FLOAT(NCP/ICR1)
          NCL = INT(CK)
          IF(JSUM .NE. 1) THEN
            NCL = NCP-NCL*ICR1
          END IF
        END IF
      END DO
      NCP = NCL
      JSUM = JSUM + 1
      IF(NCP .GT. ICR1) THEN
        CK = FLOAT(NCP/ICR1)
        NCL = INT(CK)
        IF(JSUM .NE. 1) THEN
          NCL = NCP-NCL*ICR1
        END IF
      END IF
    END IF
  END DO
END DO
END SUBROUTINE ASGN
ELSE
    JSUM = JSUM + 1
END IF
J = MOD(NCL,L)
IF(J .EQ. 0) THEN
    IB = NCL/L
    II = IB + 1
    IF(II .GT. NN) II=II-NN
    IVC = CL(K,IB)
    IVC1 = CL(K,II)
    DO WHILE(IVC .EQ. 4)
        IY = IY + L
        IB = IB - 1
        IVC = CL(K,IB)
        IVP = A(IX,IY)
        IF(IVP .GT. IDF1) THEN
            JSUM = JSUM - 1
            NCP = MOD(IVP,ICR)
        END IF
    END DO
    DO WHILE(IVC1 .EQ. 0)
        IY = IY + L
        II = II + 1
        IVC1 = CL(K,II)
        IVP = A(IX,IY)
        IF(IVP .GT. IDF1) THEN
            JSUM = 0
            NCP = MOD(IVP,ICR)
        END IF
    END DO
    TEST1 = (IVC .GT. 0 .AND. IVC .LT. 4)
    TEST2 = (IVC .GT. 4 .AND. IVC .LT. 8)
    TEST3 = (IVC1 .GT. 0 .AND. IVC1 .LT. 4)
    TEST4 = (IVC1 .GT. 4 .AND. IVC1 .LT. 8)
    TEST5 = (TEST1 .AND. TEST3)
    TEST6 = (TEST2 .AND. TEST4)
    IF(TEST5 .OR. TEST6) NIS = NIS + 1
ELSE
    NIS = NIS + 1
END IF
END DO
ELSE
    IF(NPK .EQ. 1) A(IX,IY)=IDF
END IF
IF(JSUM .NE. 1) THEN
    IF(IY .EQ. N .AND. NIS .EQ. 0) IX=N
END IF
END DO
END
subroutine map(a,b,w,n,1,choice)

implicit none
integer m,np,a(m,m),b(np),n,choice,x,y
integer ms,icr,icr1,mi(ms,ms),ck,isum
integer i,j,ii,jj,l
common m,np,x,y
common /CHK1/ms,icr,icr1

isum = 0
do i = 1, m
   ii = i + 1
   do j = 1, m
      jj = j + 1
      if(w(ii,jj).ne.0) then
         ck = mod(w(ii,jj),icr)
         if(ck.eq.0) then
            isum = isum + 1
            if(choice.eq.1) then
               b(isum) = a(i,j)
            else
               a(i,j) = b(isum)
            end if
         else
            if(choice.eq.0) a(i,j)=0
         end if
      else
         if(choice.eq.0) a(i,j)=0
      end if
   end do
end do
if(choice.eq.1) n=isum
end
Subroutine initial(a,b,c,SD,PTH,PC)
real*4 b,c,PTH,PC
integer nx,nfx,ny,nfy,a,SD,CH,pX,pY
common /res/nx,nfx,ny,nfy
common /prn/pX,pY
print *, 'Enter the Number for Choice of the channel :'
print *, '   1 : For noise channel'
print *, '   0 : For errorless channel'
read *, a
if(a .eq. 1) then
   print *, 'Enter the BER(real) :'
   read *, b
end if
print *, 'Enter the first X block number :'
read *, nx
print *, 'Enter the last X block number :
read *, nfx
print *, 'Enter the first Y block number :
read *, ny
print *, 'Enter the last Y block number :
read *, nfy
print *, 'Enter the initial distance for prediction :
read *, SD
print *, 'Enter the threshold for prediction :
read *, PTH
print *, 'Enter the correlation coefficient :
read *, PC
print *, 'Enter the threshold value of transform coefficients:
read *, C
print *, 'Do you want to see hybrid coding process of the block ?'
print *, '   If yes, type 1'
print *, '   If no, type 2'
read *, CH
if(CH .eq. 1) then
   print *, 'Enter the X block number for printing out :
read *, PX
   print *, 'Enter the Y block number for printing out :
read *, PY
else
   PX = 0
   PY = 0
end if
end
REAL FUNCTION THRESHOLD(X,T,CHOICE)

INTEGER CHOICE
REAL*4 X, T, VALUE

VALUE = ABS(X)
IF (CHOICE .EQ. 1) THEN
  IF (VALUE .GT. T) THEN
    IF (X .GT. 0.) THEN
      THRESHOLD = X - T
    ELSE
      THRESHOLD = X + T
    END IF
  ELSE
    THRESHOLD = 0.
  END IF
ELSE
  IF (VALUE .NE. 0.) THEN
    IF (X .GT. 0.) THEN
      THRESHOLD = X + T
    ELSE
      THRESHOLD = X - T
    END IF
  ELSE
    THRESHOLD = 0.
  END IF
END IF

END

SUBROUTINE CHANNEL(B,NB,BER)

INTEGER B(NB),BIT,BIR
REAL*4 BER,THS
INTEGER Z

THS = 1 - BER
Z = 2 ** 20 +1
DO I = 1, NB
  BIT = B(I)
  IF(BIT .EQ. 0) BIR=1
  IF(BIT .EQ. 1) BIR=0
  Y = RAN(Z)
  IF(Y .GT. THS) B(I)=BIR
END DO

END
SUBROUTINE FBACK(MG,PF,F,NT,INP,TH,XV,YV,AP,KK,CHOICE)

IMPLICIT NONE
INTEGER M,MG(M,M,X,Y,XL,YL,XV,YV,KK
INTEGER ISUM,i,j,II,JJ,IX,IY,NT,INP,CHOICE
integer nx,nfx,ny,nfy,py,x1,y1,iiy
REAL*4 F(NT,NT),PF(NT,NT),A(16,16),B(16,16),px
REAL*4 GA,GB(16,16),GC,TH,AP,INP,THRESHOLD
logical yes
common M,NP,x1,y1
common /res/nx,nfx,ny,nfy
common /mpn/yes

iiy = nfx-ny+1
px = float(kk)/float(iiy)
x1 = nx+int(px)
py = mod(kk,iiy)
if(py .eq. 0) then
  x1 = x1 - 1
  py = iiy
end if
y1 = ny + px - 1
X = M * (x1-1) + 1
XL = M * x1
Y = M * (y1-1) + 1
YL = M * y1
ISUM = 0
DO i = 1, M
  DO j = 1, M
    ISUM = ISUM + 1
    GA = FLOAT(MG(i,j))
    IF(ISUM .EQ. 1) THEN
      GC = GA
      GB(i,j) = GA
    ELSE
      GB(i,j) = GA * INP
      GC = THRESHOLD(GB(i,j),TH,0)
    END IF
    A(i,j) = GC
  END DO
END DO
if(yes) then
  write(6,100)
  do i = 1, M
    write(6,1) (GB(i,j),j=1,M)
  end do
  write(6,11) x1,y1,INP
  do i = 1, M
    write(6,1) (A(i,j),j=1,M)
  end do
  write(6,22) x1,y1,TH
END IF

end if
CALL XFORM(A,B)
DO i = X, XL
   II = i - X + 1
   IX = i + XY
   DO j = Y, YL
      JJ = j - Y + 1
      IY = j + YV
      IF(CHOICE .EQ. 1) THEN
         F(i,j) = B(II, JJ) + PF(IX, IY)
      ELSE
         F(i,j) = B(II, JJ) + AP * PF(IX, IY)
      END IF
   END DO
END DO
END IF
if(yes) then
   write(6,100)
   do i = 1, M
      write(6,1) (B(i,j), j=1,M)
   end do
   write(6,33) x1,y1
   do i = X, XL
      write(6,1) (F(i,j), j=Y, YL)
   end do
   write(6,44) x1,y1
end if

1     format(5x,16f6.1)
11    format(/,5x,** TRANSFORM COEFFICIENTS OF (',i2,','i2,')th
     1    ' SUBBLOCK AFTER INVERSE NORMALIZATION',/,'9x,
     1    'NORMALIZATION FACTOR = ',f4.1,///)
22    format(/,5x,** TRANSFORM COEFFICIENTS OF (',i2,','i2,')th
     1    ' SUBBLOCK AFTER ADDING THRESHOLD',/,'9x,'THRESHOLD',
     1    'VALUE = ',f4.1)
33    format(/,5x,** RECONSTRUCTED VALUES OF THE PREDICTION ERROR OF',
     1    '(',',i2,','i2,')th SUBBLOCK',///)
44    format(/,5x,** RECONSTRUCTED INTENSITY VALUES OF (',i2,','
     1    ',i2,')th SUBBLOCK')
100   format(1h1///)

END
**THIS FORTRAN SUBROUTINE PERFORMS THE ENCODING (ENCODING) AND MULTIFLEXING (DEMULTIFLEXING) OPERATIONS. VARIABLES USED IN THIS SUBROUTINE ARE DEFINED AS FOLLOWS:**

- **A(NP)** = TRANSFORM COEFFICIENT
- **L** = CHAIN LINK SIZE
- **CN** = NUMBER OF CHAIN ENCLOSED COEFFICIENTS
- **C(M,NP)** = CHAIN CODE
- **NC(M)** = NUMBER OF CHAIN LINKS FOR THE BOUNDARY
- **NB** = NUMBER OF BOUNDARY WITHIN THE SUB-BLOCK
- **BPF** = BIT COUNTER
- **(IX, IY)** = STARTING POINT OF THE BOUNDARY
- **(CX, CY)** = MOTION DISPLACEMENT VECTOR FOR SUB-BLOCK
- **M** = SUB-BLOCK SIZE
- **NP** = NUMBER OF POINTS WITHIN THE SUB-BLOCK

**THE CHOICE OF THE ENCODING OR DECODING PROCESS IS GIVEN BY:**

- **CHOICE = 1** ! ENCODING PROCESS
- **CHOICE = 0** ! DECODING PROCESS

**SUBROUTINE CODE(A, L, CN, B, C, NC, NB, BPF, IX, IY, CX, CY, M, NP, CHOICE)**

**INTEGER** ECBC, ECC, EBC, CODEA, CODEB, CODEC, NTC, CEV
**INTEGER** BPF, CC, NC(M), CHOICE, DIFF, CX, CY, CN, DCC
**INTEGER** A(NP), B(600000), C(M, NP), IX(M), IY(M)
**logical** yes
**common** /mpn/yes

**NFXB = 4** ! Set the number of bits for starting point.
**NDCB = 8** ! Set the number of bits for DC coefficients.
**MDBP = 3** ! Set the number of bits for displacement vector.

**** BLOCK ENCODING & DECODING OF THE MOTION ****
**** DISPLACEMENT VECTOR FOR THE SUBBLOCK ****

**DO KS = 1, 2**
  **ibp = BPF**
  **BPF = BPF + 1**
  **IF(CHOICE .EQ. 1) THEN**
    **B(BPF) = 0**
    **IF(KS .EQ. 1) THEN**
      **IV = IABS(CX)***
      **IF(CX .LT. 0) B(BPF) = 1**
    **ELSE**
      **IV = IABS(CY)***
      **IF(CY .LT. 0) B(BPF) = 1**
    **END IF**
  **ELSE**
    **IF(B(BPF) .EQ. 0) IAD = 1**
    **IF(B(BPF) .EQ. 1) IAD = -1**
  **END IF**
CALL BLOCK(MDBP,BPF,IV,B,CHOICE)
IF(CHOICE .EQ. 0) THEN
  IF(KS .EQ. 1) THEN
    XV = IV * IAD
  ELSE
    YV = IV * IAD
  END IF
END IF
END IF
if(yes) then
  if(KS.eq.1) then
    print *, ** CODING/DECODING MOTION DISPLACEMENT VECTOR **'
    print *,
    write(6,1) XV,(B(iw),iw=ibp+1,BPF)
  else
    write(6,1) YV,(B(iw),iw=ibp+1,BPF)
  end if
end if
END DO
BPF = BPF + 1
IF(CHOICE .EQ. 1) THEN
IF(NB .NE. 0) THEN
  B(BPF) = 0
  if(yes) ibp = BPF
  BPF = BPF + 1
  IDA = IABS(A(1))
  B(BPF) = 0
  if(A(1).lt.0) B(BPF)=1
  IDC = 2**NDCB - 1
  if(IDA.gt.IDC) IDA=IDC
  call BLOCK(NDCB,BPF,IDA,B,1) ! Encoding DC coefficient
  if(yes) then
    write(6,100)
    print *, ** ENCODING DC COEFFICIENT **'
    write(6,2) A(1),(B(iw),iw=ibp+1,BPF)
  end if
END IF
C ** CHAIN ENCODING for TRANSFORM BLOCK **

DO I = 1, NB
  NX = IX(I)-L
  NY = IY(I)-L
  if(yes) then
    ibp = BPF
    write(6,100)
    print *, ** BLOCK ENCODING FOR BOUNDARY STARTING POINT **'
    print *,
  end if
  CALL BLOCK(NFXB,BPF,NX,B,1)
  if(yes) then
    write(6,1) IX(I),(B(iw),iw=ibp+1,BPF)
    ibp = BPF
  end if
CALL BLOCK(NFXB,BPF,NY,B,1)
if(yes) then
    write(6,1) IY(I),(B(iw),iw=ibp+1,BPF)
    write(6,11) I
end if
NCC = NC(I)

DO J = 1, NCC+1
  IF(J .EQ. NCC+1) THEN
    IF(I .EQ. NB) THEN
      DCC = 4    ! END BLOCK CODE
    ELSE
      DCC = -4  ! END CHAIN CODE
      END IF
    ELSE
      CC = C(I,J)
      DCC = DIFF(CC,C,I,J,M,NP,1)
      END IF
  IF(yes) ibp = BPF
  CALL C_CODE(DCC,B,BPF,1)
  IF(yes) then
    write(6,3) J,C(I,J),DCC,(B(iw),iw=ibp+1,BPF)
  end if
END DO
END DO

C ** HUFFMAN ENCODING for TRANSFORM COEFFICIENTS **

if(yes) then
  write(6,100)
  print *,' ** HUFFMAN ENCODING FOR AC COEFFICIENT **'
  print *,'
  print *,'
  print *,'   AC CODE'
  print *,'
  print *,'   HUFFMAN COEFFICIENT WORD'
end if

DO I = 2, CN+1
  IF(I .EQ. CN+1) THEN
    NTC = 16   ! END BLOCK CODE
    IVL = 0
    if(yes) A(I) = 444
  ELSE
    CEV = A(I)
    IVL = IABS(CEV)
    IF(IVL .GE. 15) THEN
      IF(CEV .GT. 0) NTC=15
      IF(CEV .LT. 0) NTC=-15
    ELSE
      NTC = CEV
      IVL = 0
    END IF
  END IF
end if
if(yes) ibp = BPF
CALL HUFFMAN(NTC,IVL,BPF,B,1)  ! FOR AC COEFFICIENTS
if(yes) then
    write(6,4) 1-1,NTC,A(I),B(iw),iw=ibp+1,BPF
end if
END IF
ELSE
B(BPF) = 1
END IF
ELSE
IF(B(BPF) .EQ. 0) THEN
    if(yes) ibp=BPF
    BPF = BPF + 1
    ick = 1
    if(B(BPF) .EQ. 1) ick=-1
    call BLOCK(NDCB,BPF,IDA,B,0) ! Decoding of DC coefficient
    A(1) = IDA * ick
    if(yes) then
        write(6,100)
        print *, ' ** DECODING DC COEFFICIENT **'
        write(6,2) A(1),B(iw),iw=ibp+1,BPF
    end if
    EBC = 16 ! END BLOCK CODE
    ECBC = 4 ! END CHAIN BLOCK CODE
    ECC = -4 ! END CHAIN CODE
C ** CHAIN DECODING for TRANSFORM BLOCK **
CODEA = 0
K = 0
DO WHILE(CODEA .NE. ECBC)
    K = K + 1 ! Boundary Number
    if(yes) then
        write(6,100)
        print *, ' ** BLOCK DECODING FOR BOUNDARY STARTING POINT **'
        print *
        ibp = BPF
    end if
    CALL BLOCK(NFXB,BPF,NX,B,0)
    if(yes) jbp=BPF
    CALL BLOCK(NFXB,BPF,NY,B,0)
    IX(K) = NX+L
    IY(K) = NY+L
    if(yes) then
        write(6,1) IX(K),B(iw),iw=ibp+1,jbp
        write(6,1) IY(K),B(iw),iw=jbp+1,BPF
        write(6,22) K
    end if
    CODEB = 0
    J = 0
DO WHILE(CODEB .NE. ECC)
    J = J + 1  ! Chain Link number
    if(yes) ibp = BPF
    CALL C_CODE(CODEB,B,BPF,0)
    IF(ABS(CODEB) .EQ. 4) THEN
        if(yes) then
            print *, 'END CODE =', CODEB
        end if
        IF(CODEB .EQ. 4) THEN
            CODEA = CODEB
            CODEB = ECC
        END IF
    ELSE
        C(K,J) = DIFF(CODEB,C,K,J,M,NP,0)
        if(yes) then
            write(6,3) J,C(K,J),CODEB,(B(iw),iw=ibp+1,BPF)
        end if
    END IF
END DO
NC(K) = J - 1  ! Number of Chain Link for K-th Boundary
END DO
NB = K  ! Number of Boundaries for the K-th Block

C ** DECODING for TRANSFORM COEFFICIENTS **

if(yes) then
    write(6,100)
    print *, ** HUFFMAN DECODING FOR AC COEFFICIENTS **
    print *, **
    print *, AC CODE'
    print *, # HUFFMAN COEFFICIENT WORD'
    end if
CODEC = 0
I = 1
DO WHILE(CODEC .NE. EBC)
    I = I + 1
    if(yes) ibp=BPF
    CALL HUFFMAN(NTC,IVL,BPF,B,0)
    CODEC = NTC
    IF(CODEC .NE. EBC) THEN
        IF(IVL .EQ. 0) THEN
            A(I) = NTC
        ELSE
            A(I) = IVL
        END IF
    else
        if(yes) then
            write(6,4) I-1,NTC,A(I),(B(iw),iw=ibp+1,BPF)
        end if
    END IF
    if(yes) then
        if(CODEC.eq.EBC) print *, 'END CODE =', CODEC
    end if
END IF
if(yes) then
    if(CODEC.eq.EBC) print *, 'END CODE =', CODEC
end if
END DO
DN = I - 1
ELSE
NB = 0
ENDIF

END IF
1  FORMAT(5x, i5, 8x, 4i1)
2  FORMAT(/5x, i5, 8x, 12i1)
3  FORMAT(6x, i3, 5x, i5, 5x, i5, 7x, 4i1)
4  FORMAT(6x, i3, 5x, i5, 7x, i5, 7x, 20i1)
11 FORMAT(/3x, '** CHAIN ENCODING FOR THE BOUNDARY NUMBER',
    1  i2, //, 7x, 'LINK', 5x, 'CHAIN', 6x, 'DIFF.', 4x, 'CODE', /,
    1  6x, 'NUMBER', 4x, 'CODE', 7x, 'CODE', 5x, 'WORD', /)
22 FORMAT(/3x, '** CHAIN DECODING FOR THE BOUNDARY NUMBER',
    1  i2, //, 7x, 'LINK', 5x, 'CHAIN', 6x, 'DIFF.', 4x, 'CODE', /,
    1  6x, 'NUMBER', 4x, 'CODE', 7x, 'CODE', 5x, 'WORD', /)
100 FORMAT(/)

END
SUBROUTINE C_CODE(A,B,I,CHOICE)

IMPLICIT NONE
INTEGER I,A,B(I+4),CHOICE,X

I = I + 1
IF(CHOICE .EQ. 1) THEN
  X = ABS(A)
  IF(X .LT. 2) THEN
    B(I) = 0
    I = I + 1
    IF(A .EQ. 0) B(I)=0
    IF(X .EQ. 1) THEN
      B(I) = 1
      I = I + 1
      IF(A .LT. 0) B(I)=0
      IF(A .GT. 0) B(I)=1
    END IF
  ELSE
    B(I) = 1
    I = I + 1
    IF(X .LT. 3) THEN
      B(I) = 0
      I = I + 1
      IF(A .LT. 0) B(I)=0
      IF(A .GT. 0) B(I)=1
    ELSE
      B(I) = 1
      I = I + 1
      IF(X .LT. 4) THEN
        B(I) = 0
        I = I + 1
        IF(A .LT. 0) B(I)=0
        IF(A .GT. 0) B(I)=1
      ELSE
        B(I) = 1
        I = I + 1
        IF(A .LT. 0) B(I)=0
        IF(A .GT. 0) B(I)=1
      END IF
    END IF
  END IF
END IF
ELSE
  IF(B(I) .EQ. 0) THEN
    I = I + 1
    IF(B(I) .EQ. 0) A=0
    IF(B(I) .EQ. 1) THEN
      I = I + 1
      IF(B(I) .EQ. 0) A=-1
      IF(B(I) .EQ. 1) A=1
    END IF
  END IF
END IF
ELSE
    I = I + 1
    IF(B(I) .EQ. 0) THEN
        I = I + 1
        IF(B(I) .EQ. 0) A=-2
        IF(B(I) .EQ. 1) A=2
    ELSE
        I = I + 1
        IF(B(I) .EQ. 0) THEN
            I = I + 1
            IF(B(I) .EQ. 0) A=-3
            IF(B(I) .EQ. 1) A=3
        ELSE
            I = I + 1
            IF(B(I) .EQ. 0) A=-4
            IF(B(I) .EQ. 1) A=4
        END IF
    END IF
END IF
END IF
END IF

END

SUBROUTINE BLOCK(FC,BPF,X,B,CHOICE)
INTEGER FC,FB,BPF,X,B(BPF+FC),CHOICE

FB = BPF + FC
BPF = BPF + 1
IF(CHOICE .EQ. 1) THEN
    DO I = BPF, FB
        B(I) = MOD(X,2)
        X = (X-B(I))/2
    END DO
ELSE
    X = 0
    DO I = BPF, FB
        NUM = 2**(I-BPF)
        ISUM = B(I)*NUM
        X = X + ISUM
    END DO
END IF
BPF = FB
END
FUNCTION DIFF(A,CH,K,I,M,NP,CHOICE)

IMPLICIT NONE
INTEGER M,NP,A,K,I,CH(M,NP),CC
INTEGER DD,NX,NY,DIFF,CHOICE

IF(I.EQ.1) CC=A
IF(CHOICE.EQ.1) THEN
  IF(I.GT.1) CC = A-CH(K,I-1)
  IF(CC.GT.4) THEN
    DD = CC - 8
  ELSE IF(CC.LT.-4) THEN
    DD = 8 + CC
  ELSE
    DD = CC
  END IF
ELSE
  IF(I.GT.1) CC = CH(K,I-1)+A
  IF(CC.GT.7) THEN
    DD = CC - 8
  ELSE IF(CC.LT.0) THEN
    DD = 8 + CC
  ELSE
    DD = CC
  END IF
END IF
END IF
DIFF = DD

END
subroutine huffman(a,cev,n,b,choice)

implicit none
integer a,n,b(n+15),iv1,fct,c
integer pct,choice,nbcl,cev,acv
logical test1,test2,test3,test4
logical test5,test6,test7,test8

nbcl = 6
pct = 2**nbcl
n = n + 1
if(choice .eq. 1) then
  iv1 = iabs(a)
  b(n) = 0
  if(a .lt. 0) b(n)=1
  n = n + 1
  if(iv1.eq.0) then
    b(n) = 1
  else
    b(n) = 0
    n = n + 1
    test1 = iv1.eq.1
    test2 = iv1.eq.5 .or. iv1.eq.8
    test3 = iv1.eq.9 .or. iv1.eq.10
    test4 = iv1.eq.11 .or. iv1.eq.12
    test5 = iv1.eq.13 .or. iv1.eq.14
    test6 = iv1.eq.2 .or. iv1.eq.16
    test7 = test1.or.test2.or.test4
    test8 = test3.or.test5.or.test6
    if(test7.or.test8) then
      b(n) = 0
      n = n + 1
      if(test7) then
        b(n) = 0
        n = n + 1
        if(iv1 .eq. 1) then
          b(n) = 0
        else
          b(n) = 1
          n = n + 1
        if(iv1.eq.5) then
          b(n) = 1
        else
          b(n) = 0
          n = n + 1
        if(iv1.eq.8) then
          b(n) = 1
        else
          b(n) = 0
          n = n + 1
        if(iv1.eq.11) b(n)=0
        if(iv1.eq.12) b(n)=1
        end if
      end if
end if
end if
else
b(n) = 1
n = n + 1
if(iul.eq.2) then
  b(n) = 1
else
  b(n) = 0
  n = n + 1
  if(iul.eq.9.or.iul.eq.16) then
    b(n) = 0
    n = n + 1
    if(iul.eq.9) b(n)=1
    if(iul.eq.16) b(n)=0
  else
    b(n) = 1
    n = n + 1
    if(iul.eq.10) then
      b(n) = 1
    else
      b(n) = 0
      n = n + 1
      if(iul.eq.13) b(n)=1
      if(iul.eq.14) b(n)=0
    end if
  end if
end if
end if
else
b(n) = 1
n = n + 1
if(iul.eq.3.or.iul.eq.15) then
  b(n) = 0
  n = n + 1
  if(iul.eq.3) b(n)=1
  if(iul.eq.15) then
    b(n) = 0
    cev = cev-iul
    if(cev.ge.pct) cev=pct-1
    call block(nbc1,n,cev,b,i)
  end if
else
  b(n) = 1
  n = n + 1
  if(iul.eq.4) then
    b(n) = 1
else
  b(n) = 0
  n = n + 1
  if(iul.eq.6) b(n)=0
  if(iul.eq.7) b(n)=1
end if
end if

end if

end if

else

c = 0

if(b(n) .eq. 1) then
  n = n + 1
end if

n = n + 1

if(b(n) .eq. 0) then
  n = n + 1
end if

else

c = 1

n = n + 1

if(b(n) .eq. 1) then
  c = 5
end if

n = n + 1

if(b(n) .eq. 0) then
  c = 8
end if

else

c = 0

n = n + 1

if(b(n) .eq. 1) then
  c = 5
end if

n = n + 1

if(b(n) .eq. 0) then
  c = 8
end if

else

c = 0

n = n + 1

if(b(n) .eq. 1) then
  c = 10
end if

else

c = 0

n = n + 1

if(b(n) .eq. 1) then
  c = 13
end if

end if
end if
end if
end if
else
n = n + 1
if(b(n) .eq. 0) then
  n = n + 1
  if(b(n) .eq. 1) then
    c = 3
  else
    c = 15
    call block(nbc1,n,acv,b,0)
    cev = acv + c
  end if
else
  n = n + 1
  if(b(n) .eq. 1) then
    c = 4
  else
    n = n + 1
    if(b(n) .eq. 0) c=6
    if(b(n) .eq. 1) c=7
  end if
end if
end if
cev = fct * cev
a = fct * c
end if
end