

AD-A094 678

KANSAS UNIV/CENTER FOR RESEARCH INC LAWRENCE
A STUDY OF ADAPTIVE IMAGE COMPRESSION TECHNIQUES.(U)
FEB 80 R M HARALICK, R L KLEIN

F/G 17/2

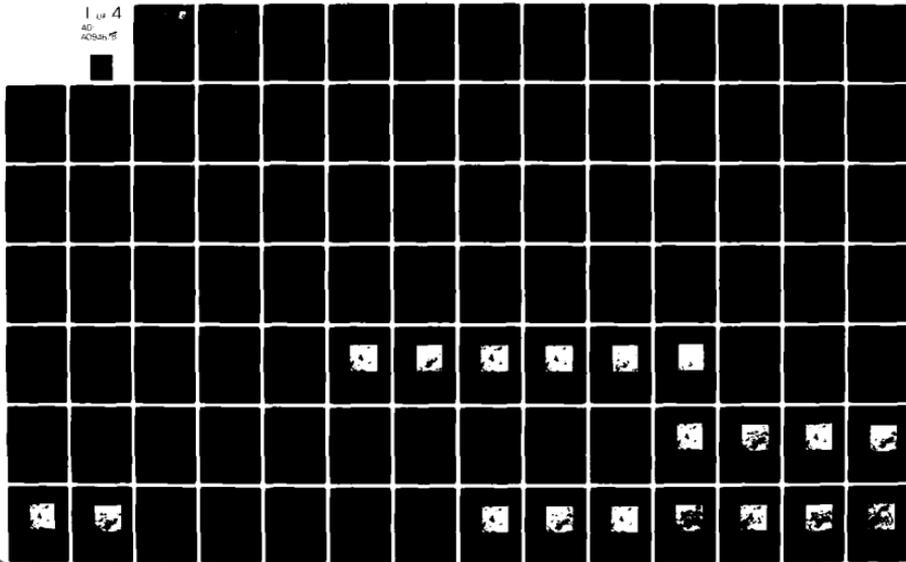
F33615-78-C-1545

UNCLASSIFIED

AFWAL-TR-80-1072

NL

1 of 4
40
AD-A094 678



AFWAL-TR-80-1072

LEVEL

2



A STUDY OF ADAPTIVE IMAGE COMPRESSION TECHNIQUES

THE UNIVERSITY OF KANSAS
CENTER FOR RESEARCH, INC.
2291 IRVING HILL DRIVE - CAMPUS WEST
LAWRENCE, KANSAS 66045

FEBRUARY 1980

TECHNICAL REPORT AFWAL-TR-80-1072
FINAL REPORT FOR PERIOD JUNE 1978 - JUNE 1979

DTIC
COLLECTED
FEB 6 1981
C

AD A094678

Approved for public release; distribution unlimited.

AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

FILE COPY

87 2 07 010

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

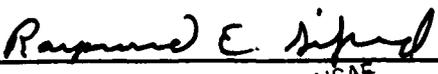


JOHN W. MAYHAN
Electronic Engineer
Data Link Technology Group, AAAD-2
Information Transmission Branch



CHARLES C. GAUDER
Chief, Information Transmission Branch
Avionics Laboratory

FOR THE COMMANDER



RAYMOND E. SIFERD, COL, USAF
Chief, System Avionics Division
Avionics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFWAL/AAAD, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFWAL-TR-80-1072	2. GOVT ACCESSION NO. AD-A094678	3. RECIPIENT'S CATALOG NUMBER 7
4. TITLE (and Subtitle) A STUDY OF ADAPTIVE IMAGE COMPRESSION TECHNIQUES.	5. TYPE OF REPORT & PERIOD COVERED Technical - Final 1 Jun 78 - 1 Jun 79	
7. AUTHOR(s) Robert M. Haralick Ronald L. Klein	6. PERFORMING ORG. REPORT NUMBER	
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Kansas Center for Research Inc. 229 Irving Hill Dr. - Campus West	8. CONTRACT OR GRANT NUMBER(s) F33615-78-C-1545	
11. CONTROLLING OFFICE NAME AND ADDRESS Avionics Laboratory (AFWAL/AAAD) Air Force Wright Aeronautical Laboratories, AFSC Wright-Patterson AFB OH 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 7662-04-31	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	12. REPORT DATE FEB 80	
	13. NUMBER OF PAGES 320	
	15. SECURITY CLASS. (of this report) Unclassified	
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for Public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Image Processing Adaptive Coding Image Compression Adaptive Image Compression Techniques		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This is a final report on the investigation of adaptive bit algorithm concepts for video image compression. Adaptive bit algorithms, using differential pulse code modulation (DPCM) and the discrete cosine transformation (DCT) with root mean squared (rms) error as a performance criterion have been derived using analytical and experimental procedures. The resulting algorithms have been implemented via software programs and their performance assessed via comparisons with fixed bit assignment procedures. Results are presented in quantitative form (i.e. rms error) as well as visual quality.		

FOREWORD

The problem of image data compression is to find a way of coding the graytone information in an image in as few bits as possible while maintaining a given image quality.

Many encoding techniques for achieving image compression have been used in the past, the most popular among these are Differential Pulse Code Modulation (DPCM) and Transform Coding.

This research was performed under U. S. Air Force Contract F33615-78-C-1545.

Accession For	
NPES	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Dist Special	
A	

TABLE OF CONTENTS

	PAGE
FOREWORD	iii
TABLE OF CONTENTS	v
LIST OF FIGURES AND TABLES - PART I	vii
LIST OF FIGURES AND TABLES - PART II	ix
PART I ADAPTIVE CODING OF IMAGES USING DIFFERENTIAL PULSE CODE MODULATION	1
1. PART 1 INTRODUCTION	1
2. CAUSAL AND NON-CAUSAL BIT ALLOCATION	3
2.1 Causal Bit Allocation	3
2.2 Non-Causal Bit Allocation	10
3. DPCM COMPRESSION TECHNIQUES	13
3.1 Closed Loop 2-D DPCM	14
3.2 Open Loop DPCM	15
3.3 Combined Open and Closed Loop DPCM	15
3.4 Models for Predictor Design	18
3.4.1 Least Squares Estimates	22
3.4.2 Minimum Variance Unbiased Estimates	22
3.4.3 Relationship Between Least Squares and Minimum Variance Unbiased Estimates	23
3.4.4 Predictor Design Solution for the Combined Open and Closed Loop DPCM	24
3.4.5 Predictor Design Solution for Closed Loop DPCM	36
4. PREPROCESSING AND POSTPROCESSING TECHNIQUES	39
5. EXPERIMENT DESIGN	42
6. EXPERIMENTAL RESULTS	50
7. CONCLUSIONS	92
8. REFERENCES	94
PART II ADAPTIVE CODING OF IMAGES USING TRANSFORM CODING TECHNIQUES	96
1. TRANSFORM CODING PROCEDURE: NON-CAUSAL CASE	97
2. ALLOCATION OF BITS	102
3. RESULTS: NON-CAUSAL ADAPTIVE ENCODING	107
4. TRANSFORM CODING: CAUSAL CASE	113
4.1 ARMA Model of Complexity of Subimages	113
4.2 Choosing an ARMA Model	114

4.3	Verification of an Identified ARMA Model for a Process	117
4.3.1	Diagnostic Tests	117
4.3.2	Model Overfitting	118
4.4	Causal Bit Allocation	120
4.5	Results of Causal Encoding Using ARMA Models for Error Prediction	125
4.6	Exponential Model of Complexity of Subimages	126
4.7	Power Series Model of Complexity of Subimages	127
4.8	Results and Conclusions: Causal Adaptive Encoding in the Transform Domain	128
5.	REFERENCES	135
APPENDIX A.1	Max Quantizer	137
APPENDIX A.2	Dynamic Programming Solution to the Optimal Non-Causal Bit Allocation Problem	141
APPENDIX A.3	Analytic Solution of the Optimal Bit Allocations Problem for the Optimal Non-Causal DPCM Quantization Problem	145
APPENDIX A.4	Computer Program Documentation	149
APPENDIX B	Key to "How to Interpret Names of Images"	204
APPENDIX C	Set of Photographs	208
APPENDIX D	Set of Graphs	223
APPENDIX E	Printouts 1 and 2	227
APPENDIX F	Technical Report - Causal DPCM Image Compression	237

LIST OF FIGURES - PART I

	PAGE
Figure 2.1 Causal Scheme	6
Figure 2.2 Possible Region for the Buffer State Given R Bits Buffer Size, K Blocks per Picture Frame and Channel Capacity c	8 - 9
Figure 2.3 Non-Causal Scheme	12
Figure 3.1 Closed Loop DPCM	16
Figure 3.2 Open Loop DPCM	17
Figure 3.3 Combined Open and Closed Loop DPCM	19
Figure 3.4 Data Set $y(r,c)$ for Combined Open and Closed Loop DPCM	25
Figure 3.5 Matrix of Correlations V for Data Set of Figure 3.4	26
Figure 3.6 Linear Prediction Mask for γ Open and Closed Loop DPCM Flat Model, Least Squares Estimates	29
Figure 3.7 Linear Prediction Mask for γ Open and Closed Loop DPCM, Flat Model, Minimum Variance Estimates	31
Figure 3.8 Linear Prediction Masks for p , Sloped Model, Least Squares Estimates, Open and Closed Loop DPCM	32
Figure 3.9 Linear Prediction Masks for p , Open and Closed Loop DPCM, Sloped Model, Minimum Variance Estimates	34
Figure 3.10 Masks for $y(0,0)$ in Combined DPCM	35
Figure 3.11 Masks for $y(0,0)$ in Closed Loop DPCM	37 - 38
Figure 5.1 Selection Made for Images Compressed at 2.0 b/p Under Non-Buffer Constrained Allocation. No Preprocessing	46
Figure 5.2 Selection Made for Images Compressed at 1.5 b/p Under Non-Buffer Constrained Allocation. No Preprocessing	47
Figure 5.3 Selection Made for Images Compressed Under Buffer Constrained Allocation	48
Figure 5.4 Selection Made for Images Compressed with Preprocessing Step	49
Figure 6.1 Original Pictures	51 - 52
Figure 6.2 Reconstructed Pictures Using Non-Buffer Constrained Allocation and Actual Errors	53 - 56
Figure 6.3 Block RMS Error vs. Bit Rate MD, SM, LS	57 - 59

Figure 6.4	Block RMS Error vs. Standard Deviation MD, SM, LS	60 - 62
Figure 6.5	Comparisons Among DPCM Predictors	63 - 68
Figure 6.6	Reconstructed Pictures Using Non-Buffer Constrained Allocation and Actual Errors	70 - 71
Figure 6.7	Reconstructed Pictures Using Buffer Constrained Allocation and Actual Errors	72 - 75
Figure 6.8	Buffer State vs. Number of Blocks Encoded MD, SM, LS	76 - 77
Figure 6.9	Buffer State vs. Number of Blocks Encoded MD, SM, LS	78 - 79
Figure 6.10	Postprocessing of the Reconstructed Pictures Obtained by Using Non-Buffer Constrained Allocation and Actual Errors	81 - 84
Figure 6.11	Preprocessing of the Original Pictures Using the Slope Facet Model	85 - 86
Figure 6.12	Reconstructed Pictures After Compressing the Preprocessed Pictures of Figure 6.11 Using Non-Buffer Constrained Allocation and Actual Errors	87 - 88
Figure 6.13	Postprocessing of the Pictures Shown in Figure 6.12	89 - 90

LIST OF TABLES - PART 1

Table 6.1	Total RMS Errors Obtained with a Fixed Bit Assignment Procedure	91
-----------	--	----

LIST OF FIGURES - PART II

	PAGE
Figure 1.1 Image Blocking and Block Sequence Specification	98
Figure 1.2 Truncated Transform Images of a Single Block	99
Figure 1.3 Error vs. Component and Error vs. Bit Curves	101
Figure 2.1 Sub-Partitioning of the Transform Image	103
Figure 4.1 Cumulative Normalized Periodogram	119

LIST OF TABLES - PART II

Table 3.1 Non-Causal KCALIF	108
Table 3.2 Non-Causal CCALIF	109
Table 3.3 Non-Causal Lady Image	110
Table 4.1 General Characteristics of Several Combinations of Time Series Classes	115
Table 4.2 RMS Error: Input-Output Images CCALIF	130
Table 4.3 RMS Error: Input-Output Images LADY3	131
Table 4.4 Parameters for Exponential Fit	132
Table 4.5 Parameters for Polynomial Fit	133

PART I

ADAPTIVE CODING OF IMAGES USING DIFFERENTIAL PULSE CODE MODULATION

1. Part I Introduction

To date most of these techniques have been utilized with fixed bit assignment procedures selected with respect to a particular encoding technique. In a fixed bit assignment procedure for a given compression ratio, each resolution cell in the image is assigned the same number of bits as any other. On the other hand, in an adaptive encoding scheme the number of bits allocated to different areas in the image changes according to area complexity. We expect that the performance improvement available through adaptive encoding will be greater than that possible by experimentally "fine tuning" a particular kind of encoding scheme and using a fixed bit allocation procedure.

We assume that the image under consideration is partitioned into a set of equal-sized, nonoverlapping blocks or subimages. The encoding of the image will then take place sequentially, block after block, and the encoding time for each block will be the same. We will also impose the constraint of a fixed number of bits per picture frame. The need for an adaptive bit allocation procedure arises for two reasons:

- (1) The statistical characterizations of the image data are not known in advance.
- (2) Some blocks of the image are more complex than others and require more encoded bits to maintain image quality.

Point 2 suggests that the bit rate generated by the allocation procedure should be variable, changing as the complexity of the blocks through the image changes. Best utilization of the channel, however, indicates that the channel capacity should be the desired long-range bit

transmission rate. This implies that a buffer is needed to accept a variable rate input bit stream and which produces a constant rate bit stream to dump into the channel.

To ensure the long-range average bit transmission rate equals the channel capacity and at no time does the buffer overflow or underflow, a controller is needed which, given the constraints of buffer size and output bit stream rate, will allow more or fewer bits to be allocated to any given block depending on block complexity. For this to be possible, the controller must have knowledge of buffer state; i.e., how full the buffer is, and the complexity of future blocks. In this paper, we use the RMS error versus bit rate function of a block as a measure of its complexity.

In order to evaluate the results of the causal rate buffer constrained bit allocation procedure, we performed experiments. Using RMS error as our criteria the experiments showed that the procedures can be rank ordered from best to worst by

- (1) Non-causal optimal bit allocation
- (2) Non-causal optimal bit allocation with rate buffer constraints
- (3) Causal adaptive bit allocation with rate buffer constraints
- (4) Non-adaptive bit allocation

Procedures (1) through (3) will be discussed in the next section. Experimental results and comparisons between (1) and (2) using several DPCM compression techniques will be shown in the last section. Section two describes the DPCM techniques used through the Part I experiments and section three describes the image preprocessing and postprocessing techniques used in Part I.

2. Causal and Non-Causal Bit Allocation

In this section we will formally define the causal and non-causal bit allocation problems and provide solutions for them. In these two problems the image to be compressed is assumed to be divided into K mutually exclusive and equal-sized blocks. We want to find out a way to allocate the available bits to these blocks so that the resulting quantization error is minimized. Bit allocation constraints imposed by the buffer size are considered.

2.1 Causal Bit Allocation

In causal bit allocation, exact knowledge of error versus bit rate is not available for future blocks, but summary information of past blocks is available. Causal bit allocation then employs a model to estimate the future error versus bit rate function using the past information and the buffer constraints. Bit allocation then proceeds using these estimates. Blocks with high estimated complexity get more bits than blocks with low estimated complexity. First we describe a causal bit allocation procedure which does not use any rate buffer constraints and then we give a modification which uses the rate buffer constraints.

Let there be K blocks which must be allocated bits and let $P = \{P_1, \dots, P_N\}$ be the N possible bit allocations which can be given to each block. Let e_p be the RMS error versus bit rate function for the present block, and let e_f be the average error per block we expect to make for future blocks after allocating bits. e_f and e_p will therefore map each element in the possible bit assignments set P into a real value representing the corresponding RMS error. For any number of bits

b in the set P , $e_f(b)$ has the meaning of the average error made on a future block upon allocating an average of b bits to each future block.

Assume blocks 1 through $t-1$ are the past blocks. Let b^t be the bits allocated to the t^{th} block and B^t be the number of bits available to allocate for future blocks t to K . Then optimal bit allocation chooses b^t to minimize

$$e_p^t(b^t) + (K-t) e_f^t\left(\frac{B^t - b^t}{K-t}\right) \quad (2.1)$$

After allocating b^t bits to the t^{th} block, there remains B^{t+1} bits where

$$B^{t+1} = B^t - b^t \quad (2.2)$$

The present error function can be used to update the expected future one. When $e_p^t(b^t) > e_f^t(b^t)$, the error that resulted from allocating b^t bits to the t^{th} block is worse than that expected for future blocks and

$$\frac{e_p^t(b^t) - e_f^t(b^t)}{e_f^t(b^t)}$$

is the relative amount of error more than expected. When this is greater than zero, it should tend to make the next estimate of expected future error larger. Hence a reasonable updating formula for e_f is:

$$e_f^{t+1} = \left[1 + \gamma \frac{e_p^t(b^t) - e_f^t(b^t)}{e_f^t(b^t)} \right] e_f^t \quad (2.3)$$

To take into account rate buffer constraints, we must not allocate a number of bits which makes the rate buffer over or underflow. Let r^t be the number of bits in the rate buffer just after block $t-1$ has been processed. Between time $t-1$ and t , the rate buffer will dump c bits onto the channel and accept b^t bits from the t^{th} block. Hence,

$$r^{t+1} = r^t - c + b^t$$

The number r^{t+1} is constrained by not under or overflowing. If the buffer has R bits capacity,

$$0 \leq r^{t+1} \leq R$$

This implies

$$c - r^t \leq b^t \leq R + c - r^t \quad (2.4)$$

One possible rate buffer constrained bit allocation procedure is to choose b^t which minimizes (2.1) under the constraint of (2.4). Another is to minimize (2.1) with a penalty added for filling up the buffer. That is, assuming the buffer is initially half full and under the constraint of (2.4) minimize

$$e_p^t(b^t) + (K-t) e_f^t \left(\frac{B^t - b^t}{K-t} \right) + \frac{r^t - c + b^t - R/2}{R/2} \kappa$$

The causal scheme is illustrated in Figure (2.1).

The buffer size problem can be stated as follows. Let R_0 be the initial state of the buffer, R the buffer size, c the channel capacity and K the number of blocks per picture frame. In order to prevent

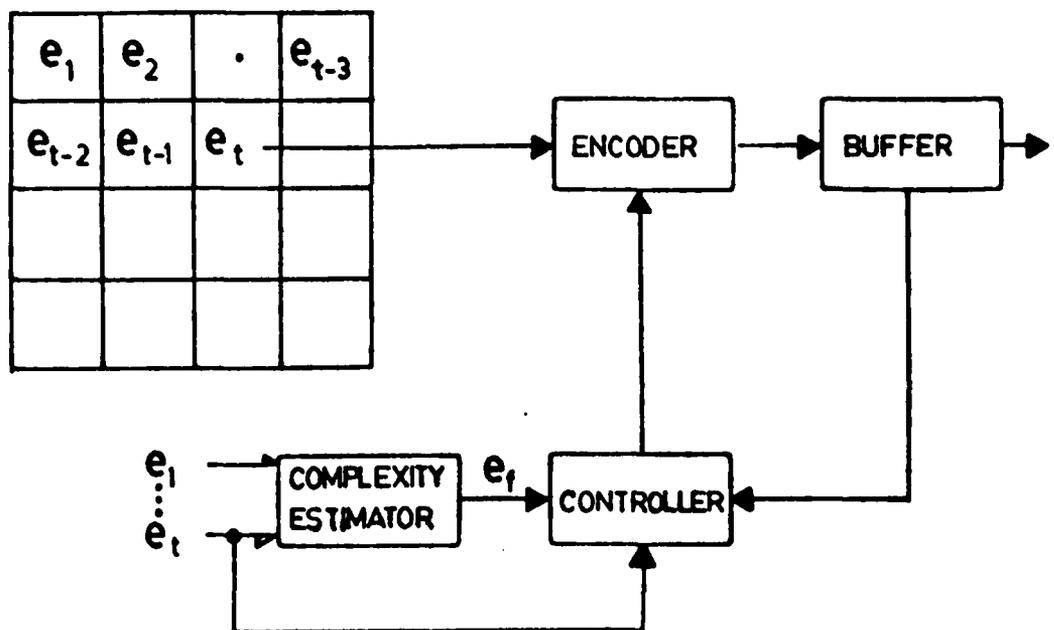


Figure 2.1 Causal Scheme

overflowing or underflowing the following relation must be satisfied for every L , $1 \leq L \leq K$.

$$-R_o \leq \sum_{\eta=1}^L b_{\eta} - Lc \leq R - R_o \quad (2.5)$$

where b_{η} represents the bits allocated to the n^{th} block. A judicious choice for R_o is

$$R_o = R/2$$

therefore (2.5) becomes

$$\left| \sum_{\eta=1}^L b_{\eta} - Lc \right| \leq R/2 \quad (2.6)$$

Letting the available number of bits per picture frame equal the amount that can be transmitted over the channel, the following relation must be satisfied for every L , $1 \leq L \leq K$.

$$\sum_{\eta=1}^L b_{\eta} \leq Kc \quad (2.7)$$

For large buffer sizes ($R > 2 K c$), only relation (2.7) imposes a constraint on the bit allocation. For smaller buffer sizes both relations (2.6) and (2.7) constraint the bit allocation. We refer to these two situations as the non-buffer constrained and buffer constrained bit allocation respectively. This is illustrated in Figure (2.2).

Assuming that the bit allocation procedure allocates bits b_1, \dots, b_k and produces a fixed error for each block, we want to choose the

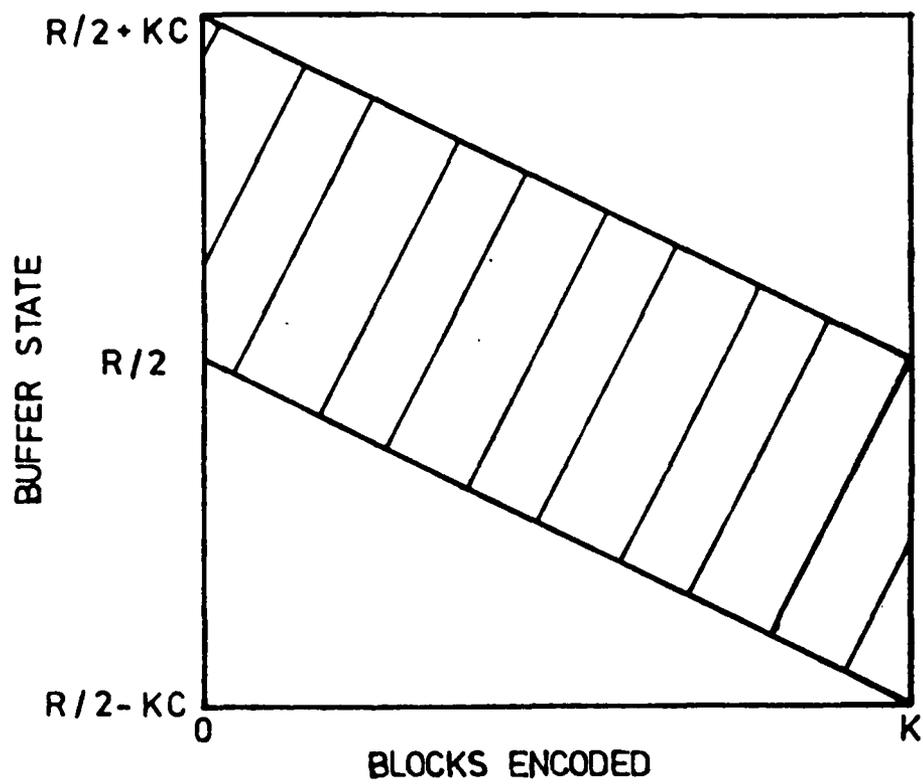
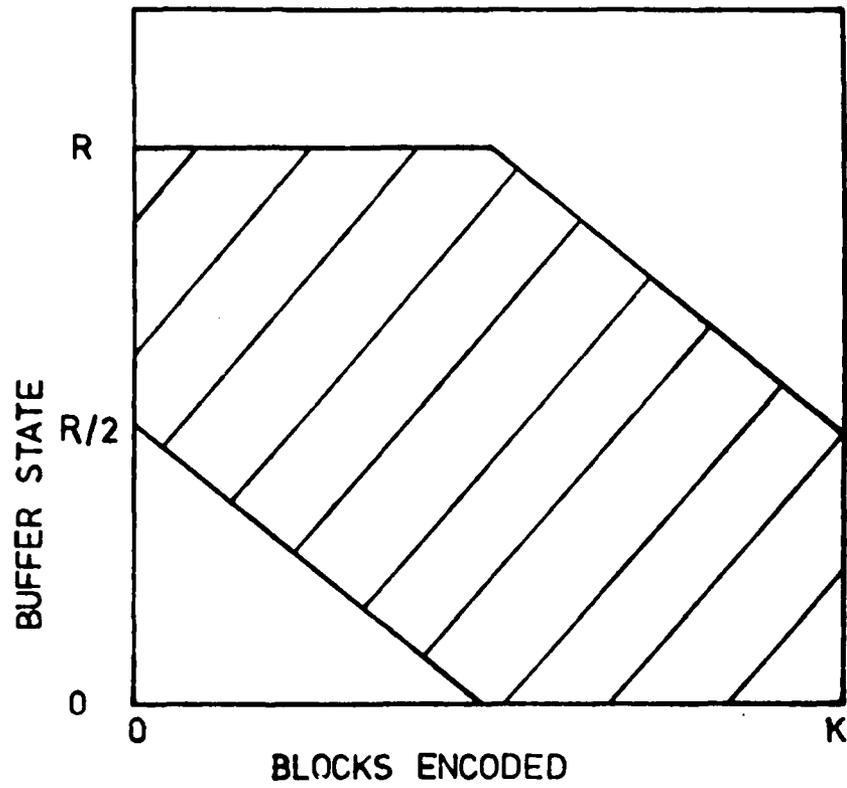


Figure 2.2 Possible Region for the Buffer State
 Given R Bits Buffer Size, K Blocks per
 Picture Frame and Channel Capacity c

(a) Non-Buffer Constrained Allocation
 ($R > 2Kc$)



(b) Buffer Constrained Allocation
($R < 2Kc$)

buffer size R so that R is the smallest size buffer satisfying that for every L , $1 \leq L \leq K$

$$P \left(\left| \sum_{\eta=1}^L b_{\eta} - Lc \right| \geq R/2 \right) \leq P_0$$

where P_0 is a given probability.

This size for R assures that by choosing a bit allocation that makes each block have the same error, the probability of buffer overflow is kept to less than probability P_0 .

2.2 Non-Causal Bit Allocation

In the non-causal bit allocation problem, the error versus bit rate functions for all blocks in the image are known before processing therefore an optimal bit allocation over these blocks can take place which will minimize the total RMS error under the constraint of a fixed number of bits per picture frame. The performance of the non-causal approach is therefore a least upper bound on any causal approach.

The optimal non-causal bit allocation problem can be stated as follows. Let there be K blocks which must be allocated bits and let $P = \{p_1, \dots, p_N\}$ be the set of N possible bit allocations which can be given to each block. The optimal non-causal bit allocation problem is then choosing b_1, \dots, b_K so that

$$\sum_{n=1}^K e_n(b_n) \quad (2.8)$$

is minimized under the constraint that

$$\sum_{n=1}^K b_n \leq B \quad (2.9)$$

where B equals the number of bits that can be transmitted over the channel.

For the non-causal buffer constrained bit allocation problem, in addition to (2.9), the following relation constraints (2.8) for all L , $1 \leq L \leq K$

$$\left| \sum_{n=1}^K b_n - Lc \right| \leq R/2 \quad (2.10)$$

where R is the buffer size and c the channel capacity.

Appendix A.2 provides a dynamic programming procedure that solves the non-causal bit allocation problem. Appendix A.3 provides an analytic solution to this problem. Figure 2.3 illustrates the non-causal scheme.

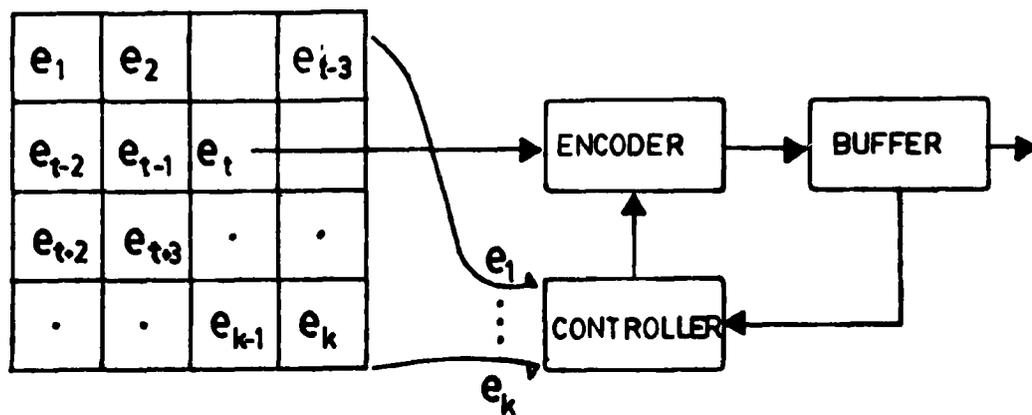


Figure 2.3 Non-Causal Scheme

3. DPCM Compression Techniques

Among the best known image compression techniques, are those based on Differential Pulse Code Modulation (DPCM). DPCM works in the following general way. Information which the receiver already has is used to predict or estimate the actual data value. This estimate is computed both at the transmitter and the receiver. The transmitter codes and sends the difference between the actual data value and the estimated data value. The receiver adds the received difference to its estimate to reconstruct the actual data value. With a good estimation scheme, the variance of the transmitted difference values is likely to be much smaller than the variance of the actual data values. In this way compression is achieved since fewer bits are required to encode the smaller variance difference values to be transmitted.

According to the spatial location of the information used in the estimation or prediction scheme, we can distinguish two main types of DPCM schemes; one-dimensional and two-dimensional DPCM. In the one-dimensional DPCM scheme values only on the current line, usually from the reconstructed image, are used to estimate the actual data value. In two-dimensional DPCM values from the current and immediate neighboring lines are used. Past reconstructed data values are used in classical 1-D and 2-D DPCM. For one-dimensional time signals, past means data values which precede the current data value in time. In classical 2-D DPCM image coding, when the image data values are generated in the usual

raster scan mode, past data values are those data values on lines above the current line or data values on the current line but to the left of the current data value.

To specify a 2-D DPCM procedure the quantizer, quantizer dither and predictor must be defined. The quantizer should be a Max quantizer based on the distribution of the image differences. A detailed information on the Max quantizer is provided in appendix A.1.

Before quantizing the image differences, a small amount of dither should be added to these differences and then subtracted from the quantized value to help eliminate contouring effects. The dither can be created from a uniformly distributed pseudo-random number generator having zero mean and a range proportional to the standard deviation of the image differences.

The predictor can be a linear combination of previously DPCMed neighboring values or values coming from a low-pass filtered image or a combination of both. This results in three types of DPCM techniques: closed loop DPCM, open loop DPCM, and combined open and closed loop DPCM. We will describe each of these techniques in sections 3.1, 3.2 and 3.3.

3.1 Closed Loop 2-D DPCM

This corresponds to the classical 2-D DPCM technique. It involves using a linear combination of the west, north-west, north, and north-east previous DPCMed values as an initial predictor, quantizing the difference between the original value and it, and forming a final predictor or reconstructed value by adding the quantized value to the

initial predictor. As mentioned before, a small amount of dither should be added to the differences before quantizing and subtracted afterwards. Figure 3.1 illustrates the closed loop DPCM concept.

3.2 Open Loop DPCM

In this DPCM technique, the current data value is estimated using some available rough estimates of data values in the neighborhood of the data value to be estimated. The algorithm is based on the fact that a low-pass filtered image makes a good estimate of the original image.

The image to be compressed is low-pass filtered and sampled taking every m th row and n th column. The sampled values are coded and sent to the receiver. The receiver estimates the image by interpolating these low-passed sampled values. The transmitter transmits the dithered quantized difference between the actual data value and the receiver's estimated value.

This open loop procedure has the following nice property: a channel error in the coded difference will affect only the data value in the resolution cell associated with the coded difference. A channel error in the PCM transmission of the low-pass filtered image will affect only the small neighborhood of resolution cells using the PCM value in the interpolation. Figure 3.2 illustrates this technique.

3.3 Combined Open and Closed Loop DPCM

This technique computes an initial predictor as a linear combination of the four DPCM values used in the closed loop technique plus values from the five remaining neighbors coming from the low-pass filtered image.

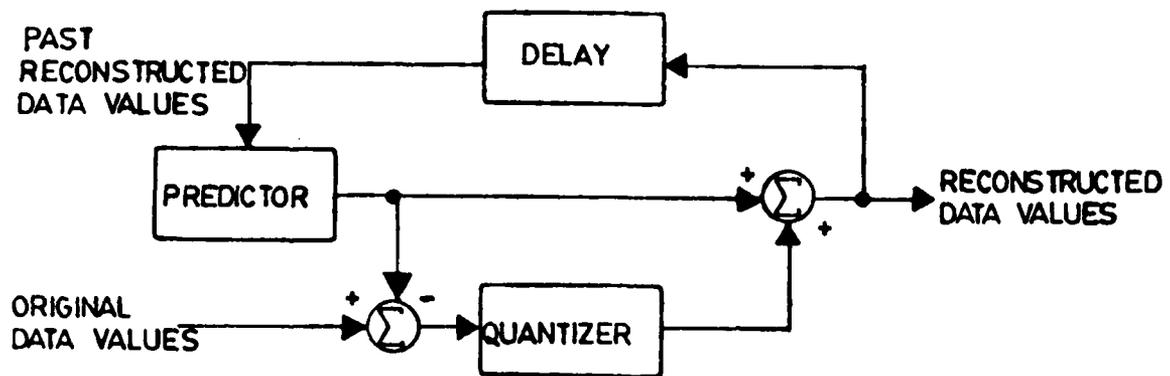


Figure 3.1 Closed Loop DPCM

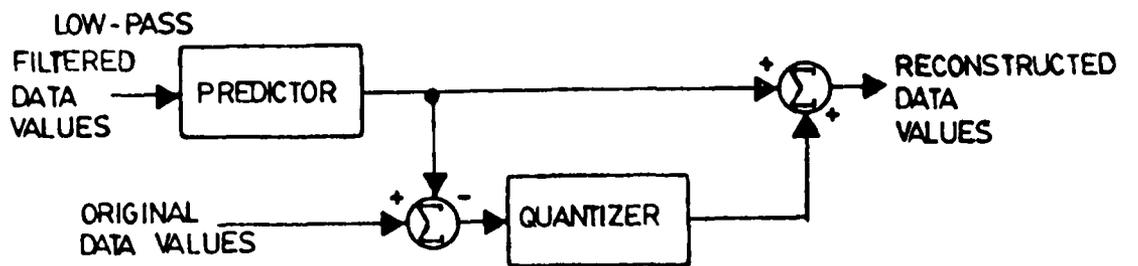


Figure 3.2 Open Loop DPCM

The difference between the original value and the predictor is determined and a small amount of dither coming from a uniform pseudo random generator is added to the difference. The result is quantized. The quantized result is added to the initial predictor and the dither is subtracted to form the final predictor. Thus to specify the combined method, the low-pass filtered image, the quantizer, the dither, and the coefficients of the predictor must be defined.

Fig. 3.3 illustrates the combined method.

3.4 Models for Predictor Design

We stated in the previous sections that the predictor can be computed as a linear combination of previous DPCM values and/or values coming from a low-pass filtered image. One question that now arises is how to determine the coefficients in the linear combination in order to have a good predictor. The answer to this question is of considerable importance since, as it was mentioned before, a good predictor scheme will result in a small variance for the image differences to be transmitted, which in turn accounts for the degree of compression achieved. We shall investigate this problem in this section and will provide several solutions each optimal under a different model.

We will approach the predictor design problem under the assumption that the resolution cells involved in the evaluation of the predictor all share a certain graytone property. In other words, our model assumes that the resolution cell whose graytone we want to estimate and those in the surrounding 3×3 neighborhood belong to a certain region within the spatial domain of the image, and therefore satisfy some relationship.

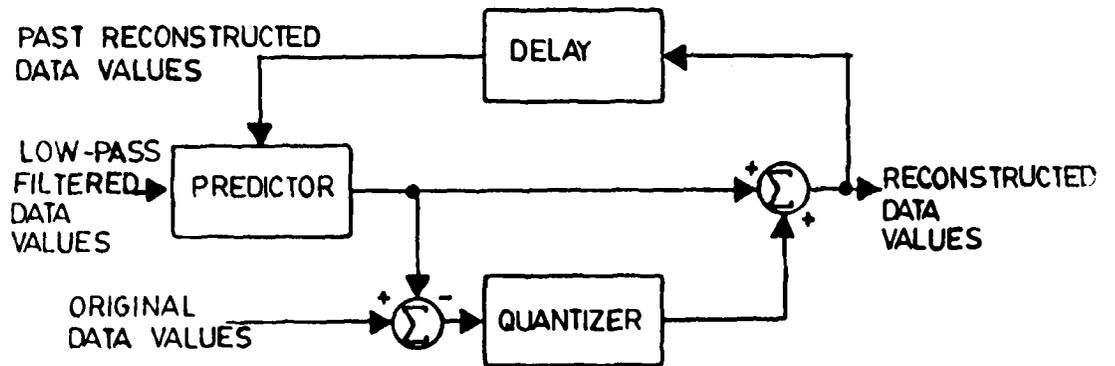


Figure 3.3 Combined Open and Closed Loop DPCM

This model will be an over-simplification of a more complete model for image data that will be described in section four, but will suffice for the purposes we are interested in this section.

Two specific sub-models that fall under the general context described above shall be investigated: the flat (horizontal) model and the sloped model. The flat model assumes that all resolution cells within a region have the same graytone value. The sloped model assumes that there is a linear relationship among row and column indexes and graytone values which is satisfied for all resolution cells within a region. If we let (r, c) be the row and column indexes pair identifying a resolution cell within a region and $I(r, c)$ the corresponding graytone value then we will have

$$I(r, c) = \gamma$$

for the flat model, and

$$I(r, c) = \alpha r + \beta c + \gamma$$

for the sloped model,

where α , β and γ are parameters that describe a particular region.

We may now take into account the effect of random noise added to the data. Assuming stationary noise with zero mean the real (noisy) image data can be described as:

$$y(r, c) = I(r, c) + z(r, c)$$

where

$$E[z(r, c)] = 0$$

$$E[z(r, c)z(r', c')] = K\sigma(r-r', c-c')$$

Let $\{y_1, \dots, y_n\}$ be the set of graytone values to be used in the estimation process, $\{r_1, \dots, r_n\}$ and $\{c_1, \dots, c_n\}$ be the corresponding sets of row and column indexes, and $\{z_1, \dots, z_n\}$ the associated random noise values; then the following linear relations result by applying the flat or sloped model.

$$\begin{bmatrix} y_1 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} = \begin{bmatrix} 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \end{bmatrix} \gamma + \begin{bmatrix} z_1 \\ \cdot \\ \cdot \\ \cdot \\ z_n \end{bmatrix}$$

for the flat model and

$$\begin{bmatrix} y_1 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} = \begin{bmatrix} r_1 & c_1 & 1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ r_n & c_n & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} + \begin{bmatrix} z_1 \\ \cdot \\ \cdot \\ \cdot \\ z_n \end{bmatrix}$$

for the sloped model.

Both relations can be expressed as

$$y = Ap + z$$

where y is a column vector representing the data, A is the design matrix, p the parameter vector and z the error vector which represents the random noise. The design matrix A can be also written in the form

$$A = [1]$$

for the flat model or

$$A = (r, c, 1)$$

for the sloped model,

where r is the column vector of row indexes, c the column vector of column indexes and 1 a column vector containing all ones.

The predictor design problem consists then in estimating the parameter vector ϕ . Two estimation schemes can be used for this purpose, least square error estimates and minimum variance unbiased estimates. Each of these two approaches is optimal as we shall see later depending on whether the noise is uncorrelated or correlated.

3.4.1 Least squares estimates

The "least squares" approach tells us to estimate p in such a way that the sum of squares of errors (noise value) is minimized. Our model is described by

$$y = Ap + z$$

Since $z'z$ is the desired sum of squares, we have

$$\begin{aligned} z &= y - Ap \\ z' &= y' - p'A' \\ z'z &= (y' - p'A')(y - Ap) \end{aligned}$$

Obtaining the derivative and setting the result to zero

$$\frac{\partial(z'z)}{\partial p} = -2A'(y - Ap) = 0$$

$$(A'A)p = A'y$$

Solving for p we obtain

$$p = (A'A)^{-1} A'y$$

3.4.2 Minimum variance unbiased estimates

The minimum variance unbiased estimate of p is obtained by the application of a very general form of Gauss Markov Theorem: Let

$$\begin{aligned} y &= Ap + z \\ E(z) &= 0 \\ \text{var}(y) &= \text{var}(z) = \sigma^2 V \end{aligned}$$

where V is a matrix (square, symmetric, non-singular) of order $(n \times n)$ with known elements. In other words the covariance matrix, i.e., the variances of $y(i)$ and covariances between $y(i)$ and $y(i)$ are known except for an arbitrary scalar multiplier applied to all of them. Then the best linear estimate of an arbitrary linear function $\ell'p$ is equal to $\ell'p^*$ where p^* minimizes the quadratic form

$$z'V^{-1}z$$

Notice that minimizing this expression is equivalent to minimizing the standard quadratic form due to error, that is

$$\frac{1}{\sigma^2} z'V^{-1}z = z'\Sigma^{-1}z$$

Obtaining the derivative and setting the result to zero

$$\begin{aligned} \frac{\partial}{\partial p} z'V^{-1}z &= \frac{\partial}{\partial p} (y - p'A')V^{-1}(y - Ap) \\ &= -2A'V^{-1}(y - Ap) = 0 \end{aligned}$$

or

$$A'V^{-1}Ap = A'V^{-1}y$$

therefore

$$p^* = (A'V^{-1}A)^{-1} A'V^{-1}y$$

3.4.3 Relationship between least squares and minimum variance unbiased estimates

We found out that the best estimate of the parameter vector p^* assuming random stationary noise with zero mean and covariance matrix proportional to a specified one, could be obtained by minimizing the standard quadratic form due to error, that is

$$z'\Sigma^{-1}z$$

where

$$E(z) = 0$$

$$\text{var}(z) = \sum = \sigma^2 V$$

If we now assume the noise to be uncorrelated, that is

$$\sum = \sigma^2 I$$

then the best estimate p^* of the parameter vector p is obtained by minimizing $z' I^{-1} z = z' z$. In other words least square estimation turns to be an optimal procedure and provides the same result as that obtained with a minimum variance unbiased estimate only if the noise is uncorrelated. Under the more general case of correlated noise least square estimation is a suboptimal procedure and a minimum variance unbiased estimate provides the optimal solution.

3.4.4 Predictor design solution for the combined open and closed loop DPCM

In this case the data set to be used in the estimation procedure comprises a 3×3 neighborhood formed by the four past DPCM data values and the five remaining neighbors coming from a lowpass filtered image. Using the set $\{-1, 0, 1\}$ for row and column indexes this data is represented in Fig. 3.4.

We will assume that the correlation between any two pixels depends only on the distance between those pixels and follows an exponential law. That is, the correlation between any two pixels K units apart is equal to p^K . The covariance matrix is then given by

$$\sum = \sigma^2 V$$

where σ^2 is the variance of any pixel and V , the matrix of correlations has the form showed in Fig. 3.5. If we let

$(-1,-1)$	$(-1,0)$	$(-1,1)$
$(0,-1)$	$(0,0)$	$(0,1)$
$(1,-1)$	$(1,0)$	$(1,1)$

Figure 3.4 Data Set $y(r,c)$ for Combined
Open and Closed Loop DPCM

	(-1, -1)	(-1, 0)	(-1, 1)	(0, -1)	(0, 0)	(0, 1)	(1, -1)	(1, 0)	(1, 1)
(-1, 1)	1	p	p ²	p	p ²	p ³	p ²	p ³	p ⁴
(-1, 0)	p	1	p	p ²	p	p ²	p ³	p ²	p ³
(-1, 1)	p ²	p	1	p ³	p ²	p	p ⁴	p ³	p ²
(0, -1)	p	p ²	p ³	1	p	p ²	p	p ²	p ³
(0, 0)	p ²	p	p ²	p	1	p	p ²	p	p ²
(0, 1)	p ³	p ²	p	p ²	p	1	p ³	p ²	p
(1, -1)	p ²	p ³	p ⁴	p	p ²	p ³	1	p	p ²
(1, 0)	p ³	p ²	p ³	p ²	p	p ²	p	1	p
(1, 1)	p ⁴	p ³	p ²	p ³	p ²	p	p ²	p	1

Figure 3.5 Matrix of Correlations V for Data Set of Figure 3.4

$$\Gamma = \begin{bmatrix} 1 & \rho & \rho^2 \\ \rho & 1 & \rho \\ \rho^2 & \rho & 1 \end{bmatrix}$$

then V can be written as

$$V = \begin{bmatrix} \Gamma & \rho\Gamma & \rho^2\Gamma \\ \rho\Gamma & \Gamma & \rho\Gamma \\ \rho^2\Gamma & \rho\Gamma & \Gamma \end{bmatrix}$$

and the inverse matrix V^{-1} becomes

$$V^{-1} = \frac{1}{1-\rho^2} \begin{bmatrix} \Gamma^{-1} & -\rho\Gamma^{-1} & 0 \\ -\rho\Gamma^{-1} & (1+\rho^2)\Gamma^{-1} & -\rho\Gamma^{-1} \\ 0 & -\rho\Gamma^{-1} & \Gamma^{-1} \end{bmatrix}$$

where Γ^{-1} is given by

$$\Gamma^{-1} = \frac{1}{1-\rho^2} \begin{bmatrix} 1 & -\rho & 0 \\ -\rho & 1+\rho^2 & -\rho \\ 0 & -\rho & 1 \end{bmatrix}$$

Any member $y(r, c)$ in the data set can be estimated as

$$y^*(r, c) = \gamma^*$$

for the flat model and

$$y^*(r, c) = \alpha^* r + \beta^* c + \gamma^*$$

for the sloped model.

Since we are interested in estimating $y(0, 0)$ both models yield

$$y^*(0, 0) = \gamma^*$$

We have two models: flat and sloped and on the other hand we have two estimation schemes: least squares and minimum variance unbiased

estimates. This result is four possible estimates for $y(0, 0)$. In the analysis that follow let the column vectors r , c and 1 be defined as follows:

$$r = \begin{bmatrix} -1 \\ -1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad c = \begin{bmatrix} -1 \\ 0 \\ 1 \\ -1 \\ 0 \\ 1 \\ 1 \\ -1 \\ 0 \\ 1 \end{bmatrix} \quad 1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

The vectors r and c represent respectively the row and column indexes for the data set $\{y(r, c)\}$ as we scan the spatial domain of the set in a left to right, top to bottom fashion.

1st Case. Flat Model; least squares estimates

For least square estimates,

$$p^* = (A'A)^{-1} A'y$$

where $p = \gamma$ and $A = 1$ (flat model)

then

$$\gamma^* = (1'1)^{-1} y$$

$$\gamma^* = \frac{1}{1'1} 1'y$$

so, in this case, the predictor is formed using a simple statistical mean.

The resulting linear prediction mask is shown in Fig. 3.6.

2nd Case. Flat Model; minimum variance unbiased estimates

In this case

$$p^* = (A'V^{-1}A)^{-1} A'V^{-1}y$$

As before $p = \gamma$ and $A = 1$

therefore

$$\gamma^* = (1'V^{-1}1)^{-1} 1'V^{-1}y$$

$$\gamma^* = \frac{1}{1'V^{-1}1} 1'V^{-1}y$$

The resulting linear prediction mask is shown in Fig. 3.7. As we expected for uncorrelated noise ($\rho = 0$) this mask reduces to the one found in the 1st case.

3rd Case. Sloped Model; least squares estimates

Here we have

$$p^* = (A'A)^{-1} A'y$$

where

$$A = [r \ c \ 1]$$

then

$$p^* = \left(\begin{bmatrix} r' \\ c' \\ 1' \end{bmatrix} [r \ c \ 1] \right)^{-1} \begin{bmatrix} r' \\ c' \\ 1' \end{bmatrix} y$$

$$p^* = \begin{bmatrix} r'r & r'c & r'l \\ c'r & c'c & c'l \\ 1'r & 1'c & 1'l \end{bmatrix}^{-1} \begin{bmatrix} r' \\ c' \\ 1' \end{bmatrix} y$$

$$p^* = \begin{bmatrix} 1/r'r & 0 & 0 \\ 0 & 1/c'c & 0 \\ 0 & 0 & 1/1'l \end{bmatrix} \begin{bmatrix} r' \\ c' \\ 1' \end{bmatrix} y$$

therefore,

$$\alpha^* = \frac{1}{r'r} r'y$$

$$\beta^* = \frac{1}{c'c} c'y$$

$$\gamma^* = \frac{1}{1'l} 1'y$$

The corresponding linear prediction masks are shown in Fig. 3.8.

4th Case. Sloped Model; minimum variance unbiased estimates

The estimate in this case is given by

$$p^* = (A'V^{-1}A)^{-1} A'V^{-1} y$$

where as before

$$A = [r \ c \ 1]$$

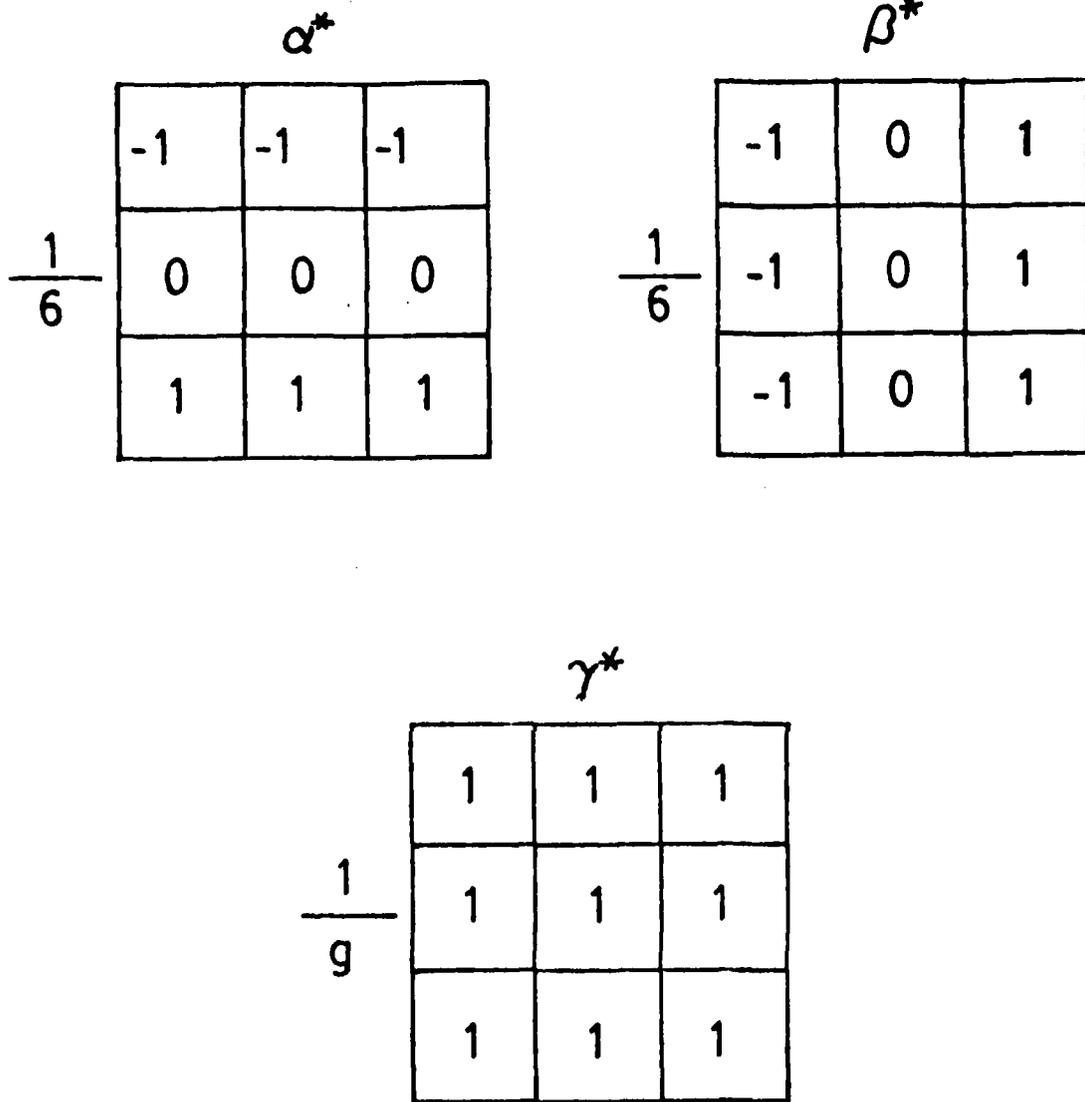


Figure 3.8 Linear Prediction Masks for p Sloped Model, Least Squares Estimates, Open and Closed Loop DPCM

then

$$\begin{aligned}
 p^* &= \left(\begin{bmatrix} r' \\ c' \\ l' \end{bmatrix} V^{-1} [r \ c \ l] \right)^{-1} \begin{bmatrix} r' \\ c' \\ l' \end{bmatrix} V^{-1} y \\
 p^* &= \begin{bmatrix} r'V^{-1}r & r'V^{-1}c & r'V^{-1}l \\ c'V^{-1}r & c'V^{-1}c & c'V^{-1}l \\ l'V^{-1}r & l'V^{-1}c & l'V^{-1}l \end{bmatrix}^{-1} \begin{bmatrix} r' \\ c' \\ l' \end{bmatrix} V^{-1} y \\
 p^* &= \begin{bmatrix} 1/r'V^{-1}r & 0 & 0 \\ 0 & 1/c'V^{-1}c & 0 \\ 0 & 0 & 1/l'V^{-1}l \end{bmatrix} \begin{bmatrix} r' \\ c' \\ l' \end{bmatrix} V^{-1} y
 \end{aligned}$$

therefore

$$\alpha^* = \frac{1}{r'V^{-1}r} r'V^{-1} y$$

$$\beta^* = \frac{1}{c'V^{-1}c} c'V^{-1} y$$

$$\gamma^* = \frac{1}{l'V^{-1}l} l'V^{-1} y$$

The corresponding linear predictor masks are shown in Fig. 3.9. Again, for uncorrelated noise these masks reduce to the ones shown in Fig. 3.8 corresponding to least squares estimates.

The masks corresponding to our desired estimate $\hat{y}(0, 0)$ for each of the cases analyzed are shown in Fig. 3.10. Notice there is no difference between the results obtained with the flat or sloped modes as far as the estimation of $y(0, 0)$ is concerned. This can be intuitively seen since the symmetry of the problem causes the horizontal plane associated with the flat model to pass through the center of the sloped fitting plane associated with the sloped model.

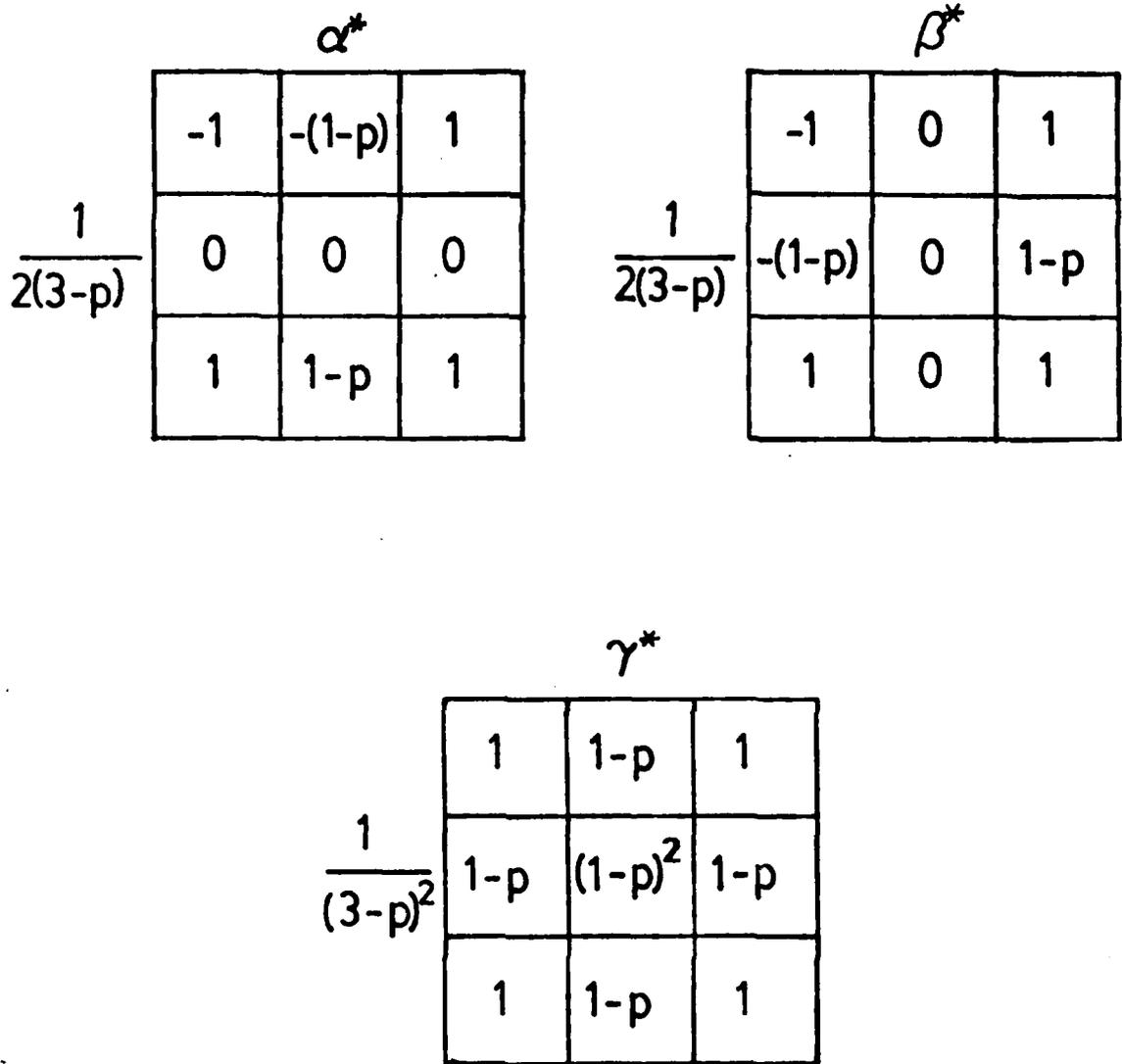


Figure 3.9 Linear Prediction Masks for p
 Open and Closed Loop DPCM,
 Sloped Model, Minimum Variance Estimates

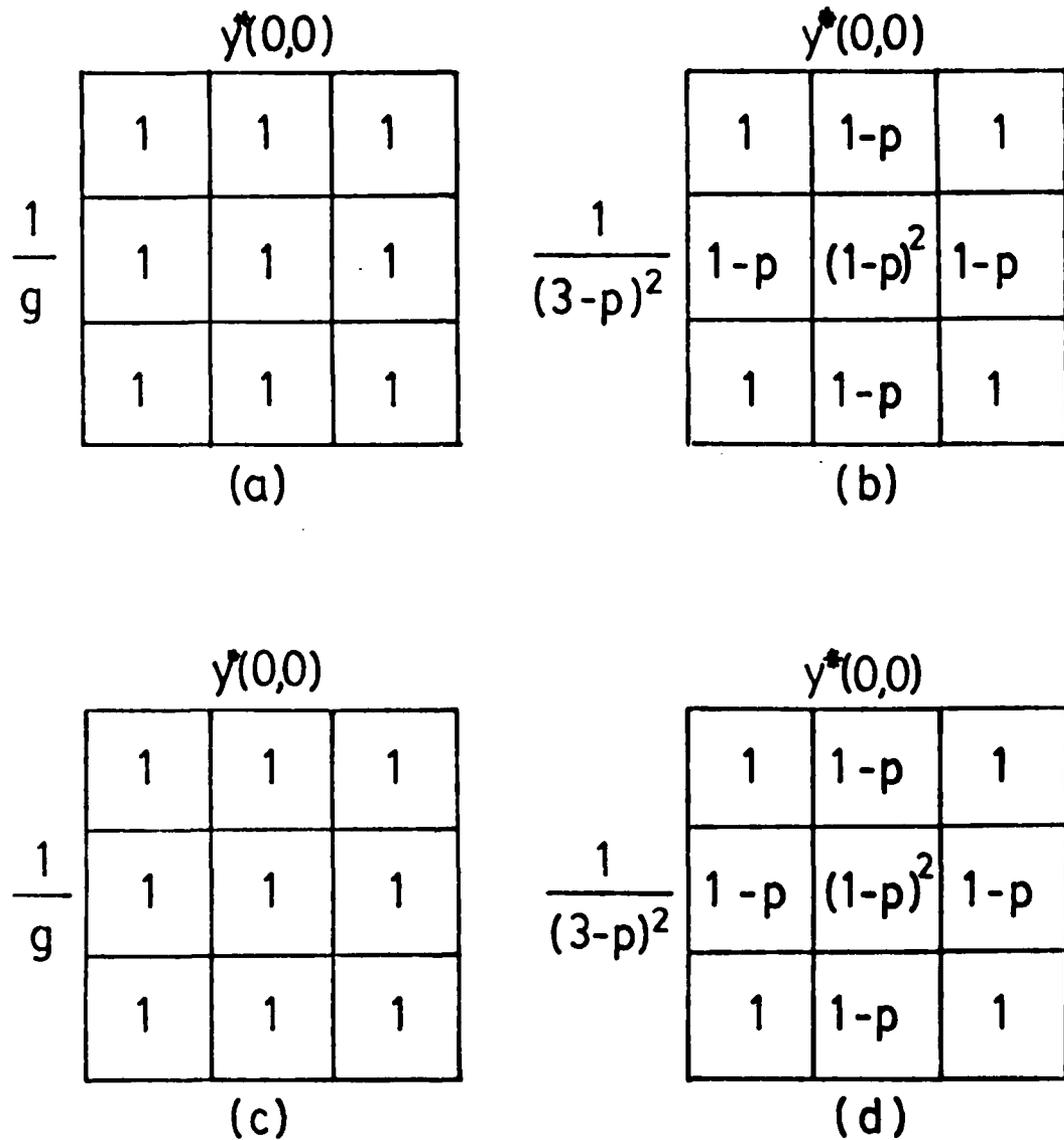


Figure 3.10 Masks for $y(0,0)$ in Combined DPCM

- (a) Flat, Least Squares
- (b) Flat, Minimum Variance
- (c) Sloped, Least Squares
- (d) Sloped, Minimum Variance

3.4.5 Predictor design solution for closed loop DPCM

As we know, the data set corresponding to this case is formed by the nearest four past DPCMed data values. The matrix of correlations V is given by

$$V = \begin{bmatrix} 1 & \rho & \rho^2 & \rho \\ \rho & 1 & \rho & \rho^2 \\ \rho^2 & \rho & 1 & \rho^3 \\ \rho & \rho^2 & \rho^3 & 1 \end{bmatrix}$$

and the inverse matrix V^{-1} is

$$V^{-1} = \frac{1}{1-\rho^2} \begin{bmatrix} 1+\rho^2 & -\rho & 0 & -\rho \\ -\rho & 1+\rho^2 & -\rho & 0 \\ 0 & -\rho & 1 & 0 \\ -\rho & 0 & 0 & 1 \end{bmatrix}$$

The analysis for the four possible estimates of $\hat{y}(0, 0)$ is carried out in exactly the same way as that described in the combined open and closed loop DPCM and we will only provide the final solutions here which are shown in Fig. 3.11.

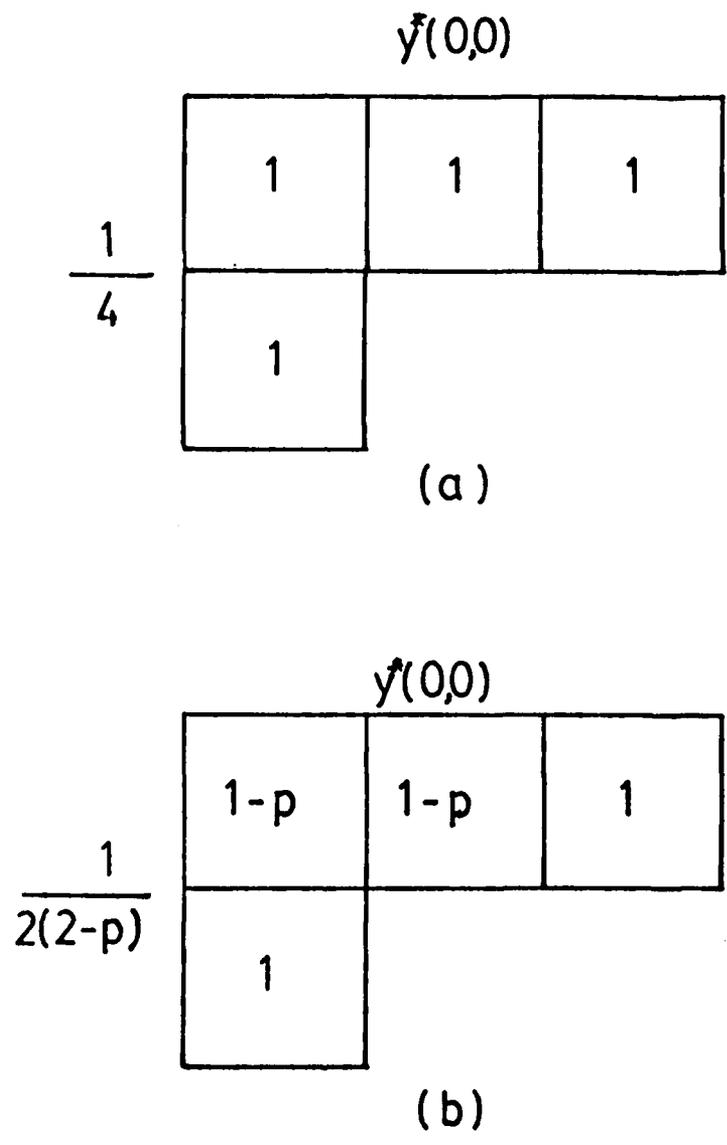


Figure 3.11 Masks for $y(0,0)$ in Closed Loop DPCM
 (a) Flat, Least Squares
 (b) Flat, Minimum Variance

$y^*(0,0)$

	-1	0	1
$\frac{1}{2}$	2		

(c)

$y^*(0,0)$

	$p^3 - 5p^2 + 9p - 3$	$2p$	$-p^3 + 5p^2 - 5p + 3$
$\frac{1}{p^4 - 5p^3 + 11p^2 - 7p + 6}$	$p^4 - 5p^3 + 11p^2 - 11p + 6$		

(d)

- (c) Sloped, Least Squares
- (d) Sloped, Minimum Variance

4. Preprocessing and Postprocessing Techniques

Two types of noise contribute to the deterioration of the information contained in the image data. The first one originates at recording time and is introduced by the remote sensors or the associated hardware; or is due to varying atmosphere variables present in the environment. The second type of "noise" accounts for the amount of information lost when compressing the data or is introduced by the channel during transmission. The effect of both types of noise can be kept to a minimum by noise filtering the data before and after transmission using one of several preprocessing or postprocessing techniques.

The technique we are going to describe in this section is based upon the facet model for image data proposed by Haralick (Reference 1) and which was briefly introduced in the previous section. The facet model assumes that the spatial domain of the image can be partitioned into regions having certain graytone and shape properties.

To assume smoothness of a region, the facet model assumes that for each image there exists a $K > 1$ such that each region in the image can be expressed as the union of $K \times K$ block of pixels. The value of K associated with an image means that the narrowest part of each of its regions is at least as large as a $K \times K$ block of pixels. Hence, images which can have large values of K have very smooth regions.

The flat (horizontal) model assumes that all pixels in the same region have the same graytone. The sloped facet model assumes that for each region there exists an affine relationship among the row, column, and graytone values which all pixels in the region satisfy.

To make these ideas precise, let Z_r and Z_c be the row and column index set for the spatial domain of an image. For any $(r, c) \in Z_r \times Z_c$, let $I(r, c)$ be the graytone value of resolution cell (r, c) and let $B(r, c)$ be the $K \times K$ block of resolution cells centered around resolution cell (r, c) . Let $A = \{A_{(n)}, \dots, A_{(n)}\}$ be a partition of $Z_r \times Z_c$ into its regions. In the facet model, for every resolution cell $(r, c) \in A_{(n)}$, there exists a resolution cell $(i, j) \in Z_r \times Z_c$ such that

$$(1) \quad (r, c) \in B(i, j) \subseteq A_{(n)}$$

$$(2) \quad I(r, c) = \gamma_{(n)} \quad (\text{flat model}) \text{ or,}$$

$$I(r, c) = \alpha_{(n)}r + \beta_{(n)}c + \gamma_{(n)} \quad (\text{sloped model})$$

Here (1) and (2) constitute respectively the shape and graytone constraints for region $A_{(n)}$.

As we have seen before the actual image differs from the ideal one by the addition of random stationary noise having zero mean and covariance matrix proportional to a specified one.

$$y(r, c) = I(r, c) + z(r, c)$$

Using the vectorial notation developed in section three:

$$y = Ap + z$$

where

$$E(z) = 0$$

$$\text{var}(z) = \sigma^2 V$$

The facet model suggests then the following simple non-linear filtering procedure. Each resolution cell is contained in K^2 different $K \times K$ blocks. The graytone distribution in each block can be fit by either a horizontal plane or a sloped plane. One of the K^2 blocks has smallest error of fit. Set the output graytone value to be the one fitted by the block having smallest error. The error of fit is given by

$z'z$ (uncorrelated noise) or,
 $z'V^{-1}z$ (correlated noise) .

The linear filter masks corresponding to the fitting parameters and for the flat and sloped model under the assumptions of uncorrelated or correlated noise were found in section three and are shown in Figs. 3.6, 3.7, 3.8 and 3.9.

5. Experiment Design

In this section we lay out the organization of the experiments done which apply the algorithms devised in section two to a number of real images using a non-causal bit allocation scheme with and without buffer size constraints.

The size of the experiments is determined by points I through VI below.

I. Images

Two original images with different degrees of complexity were used. Each of these images consist of 100 x 100 picture elements and were divided into 100 blocks of 10 x 10 picture elements before processing. Both images were quantized to 64 gray levels.

II. DPCM predictors

As mentioned in section three, eight kinds of DPCM predictors are possible, comprising all the possible combinations of

- (a) two DPCM compression techniques: Closed loop 2-D DPCM and Combined open and Closed loop DPCM
- (b) two image models: Flat model and sloped model
- (c) two estimation schemes: least squares estimates and minimum variance unbiased estimates

For minimum variance estimates a correlation coefficient of 0.3 was assumed for the noise. The notation used is:

CD - Closed loop 2-D DPCM
MD - Combined Open and Closed loop DPCM
FM - Flat model
SM - Sloped model
LS - Least squares estimates
MV - Minimum variance unbiased estimates

III. Buffer size

Two buffer sizes were selected, each one corresponding to the cases of non-buffer constrained and buffer constrained bit allocation respectively. For the case of non-buffer constrained allocation, the size of the buffer used was twice the amount of bits that can be transmitted over the channel per picture frame for a given compression ratio. This size was shown in section two to be the minimum size ensuring no constraints. For the case of buffer constrained allocation, the size of the buffer was set to 5% of the amount of bits transmitted over the channel per picture frame.

The following notation is used:

NB - Non-Buffer constrained allocation
BC - Buffer constrained allocation

IV. Compression ratio

Two compression ratios were selected; 2.0 and 1.5 bits per picture element[†]. The possible bit allocations to any block were 1, 2, 3, 4, or 5 bits per picture element.

[†]The DPCM was initialized at the beginning of each block by applying a PCM to the first line of the block with no compression. This accounts for effective compression ratios of 2.5 and 2.0 bits per picture element.

V. Error versus bit functions

We can use the actual error versus bit function for each block, or we can fit error versus block variance, for all possible bit rates and obtain the error versus bit rate functions for each block from these curves. The latter approach will tell us if the error versus bit functions can be parametrized successfully to reduce the computational burden. We chose a least squares fitting procedure that fits the data to a polynomial of degree 6. Also the variance of the difference between the actual values and the predicted values was used instead of the variance of the actual values, since it showed a better correlation with the RMS errors. We use the following notation:

AE - Actual error versus bit functions

FE - Fitted error versus bit functions

VI. Preprocessing and postprocessing techniques

The preprocessing and postprocessing techniques used correspond to the sloped facet model discussed in section four. The least squares estimation scheme was used. We use the following notation:

PR - preprocessing

NR - no preprocessing

PO - postprocessing

NP - no postprocessing

I through VI give us a total of 512 resulting images. A suitable choice of 720 images out of the total number was made to carry out the actual experiments. Figures 5.1 through 5.4 show the selection made. Still some savings result

from the fact that there are only 6 distinct predictors as observed in
figs. 3.10 and 3.11. We tried to obtain a reasonably complete set of
images using the Combined Open and Closed loop DPCM technique which
performs better than the Closed loop DPCM.

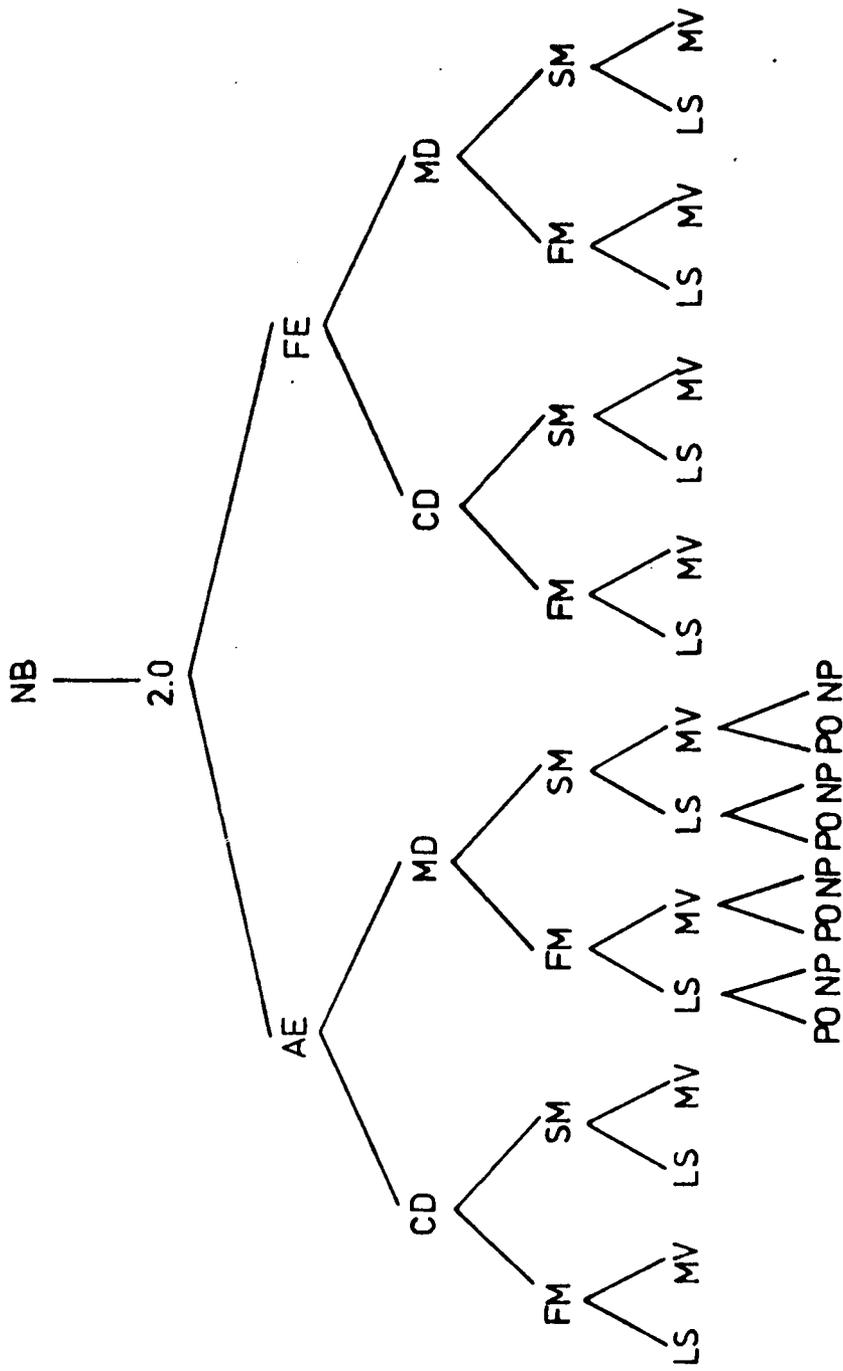


Figure 5.1 Selection Made for Images Compressed at 2.0 b/p Under Non-Buffer Constrained Allocation. No Preprocessing

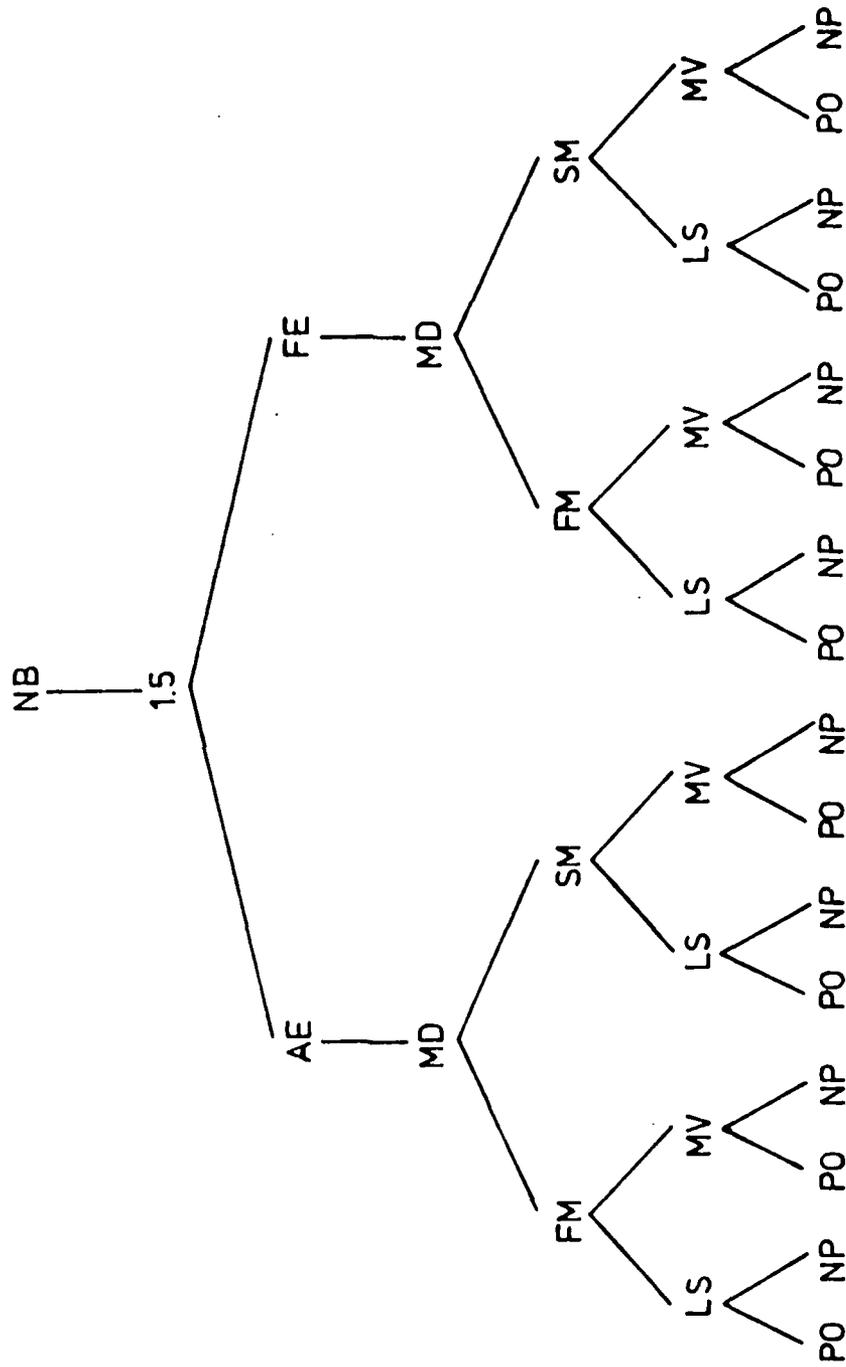


Figure 5.2 Selection Made for Images Compressed at 1.5 b/p Under Non-Buffer Constrained Allocation. No Preprocessing

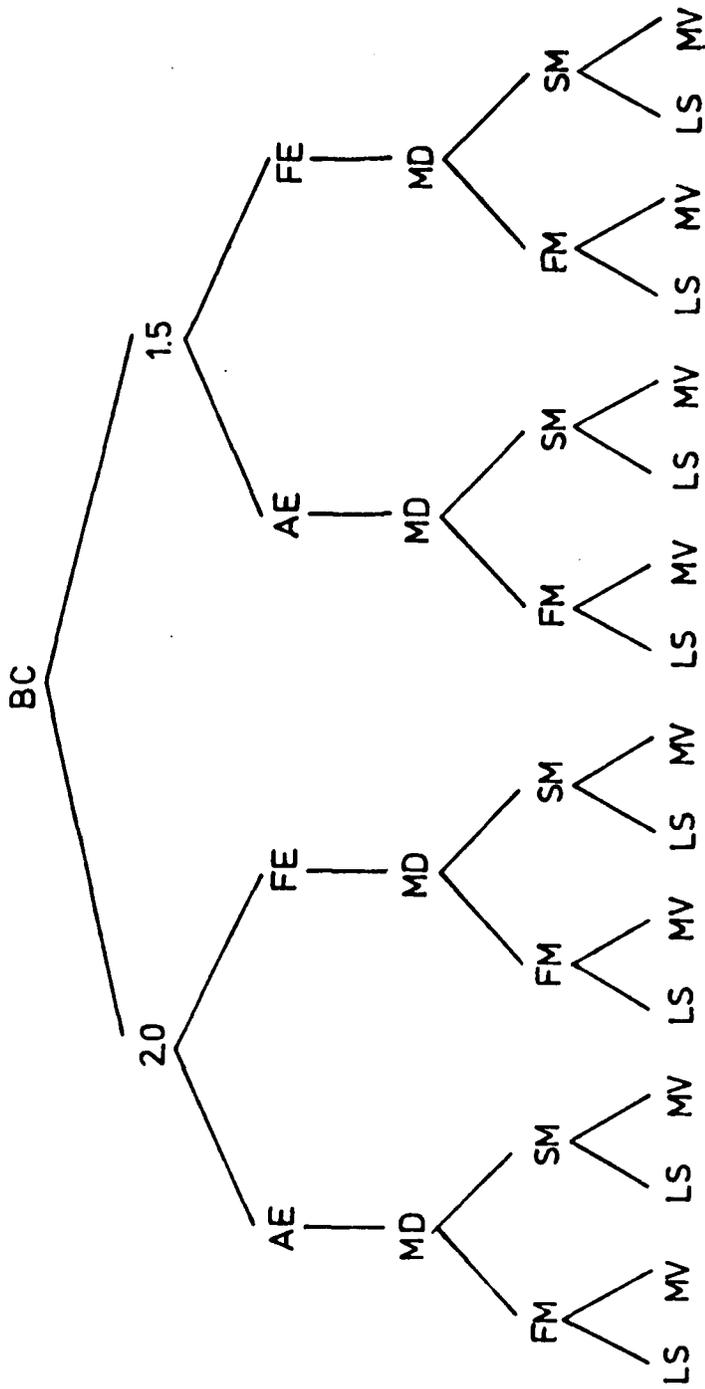


Figure 5.3 Selection Made for Images Compressed Under Buffer Constrained Allocation

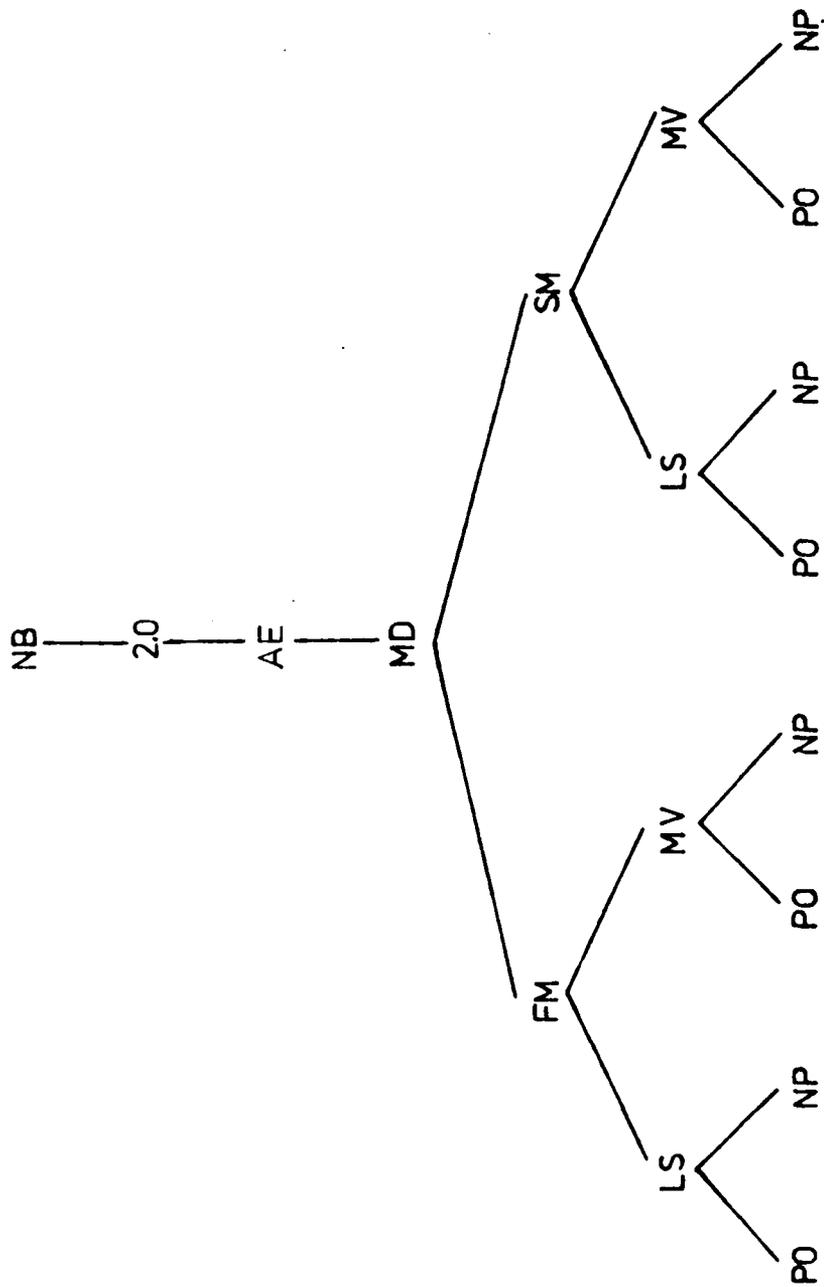


Figure 5.4 Selection Made for Images Compressed with Preprocessing Step

6. Experimental Results

We report in this section on the results obtained after carrying out the experiments laid out on the previous section. A non-causal procedure using the dynamic bit allocation algorithm described in Appendix A.2 was applied to the two LANDSAT images shown in Fig. 6.1. These images were quantized to 64 gray levels. Size, blocking and allowed bit allocations to any block are as established in section five.

I. Non-Buffer constrained allocation; Actual errors

DPCM compression procedures using each of the predictor masks developed in section three were applied to both LANDSAT images. Differences among the reconstructed images obtained depended mostly on the compression technique used, combined DPCM or closed loop DPCM and not so much on the image model or the estimation scheme used to form the predictor. Fig. 6.2 shows the reconstructed pictures compressed to 2.0 bits per picture element per two different predictors, one corresponding to the combined DPCM, sloped model, least squares estimates and the other to the closed loop DPCM, flat model, least squares estimates. RMS errors versus bit rate curves with variance as a parameter and error versus variance for several bit rates are shown in Figures 6.3 and 6.4 for the predictor corresponding to the Combined DPCM, sloped model, least squares estimates. As expected for a fixed number of bits, blocks with lower complexity (variance) have associated smaller RMS errors than those with greater complexity. Also for a fixed block complexity more encoded bits result in a smaller RMS error. A comparison of error curves among different predictors is shown in Fig. 6.5. We can observe that the RMS errors associated with the combined DPCM are significantly lower than those associated with the closed loop DPCM technique. There is no significant difference though in the



Figure 6.1 Original pictures; each picture consists of 100 x 100 picture elements quantized to 64 gray levels (6 b/p)

(a) First LANDSAT image



Figure 6.1 (continued)

(b) Second LANDSAT image

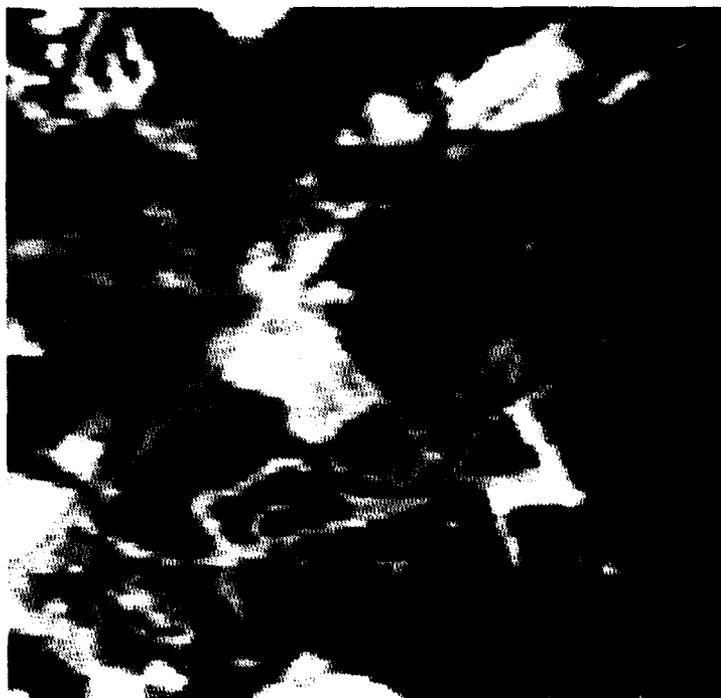


Figure 6.2 Reconstructed pictures using non-buffer constrained allocation and actual errors. Compression is 2.0 b/p.

(a) First LANDSAT image
Predictor: MD, SM, LS



Figure 6.2 (continued)

(b) First LANDSAT image
Predictor: CD, FM, LS



Figure 6.2 (continued)

(c) Second LANDSAT image
Predictor: MD, SM, LS



Figure 6.2 (concluded)

(d) Second LANDSAT image
Predictor: CD, FM, LS

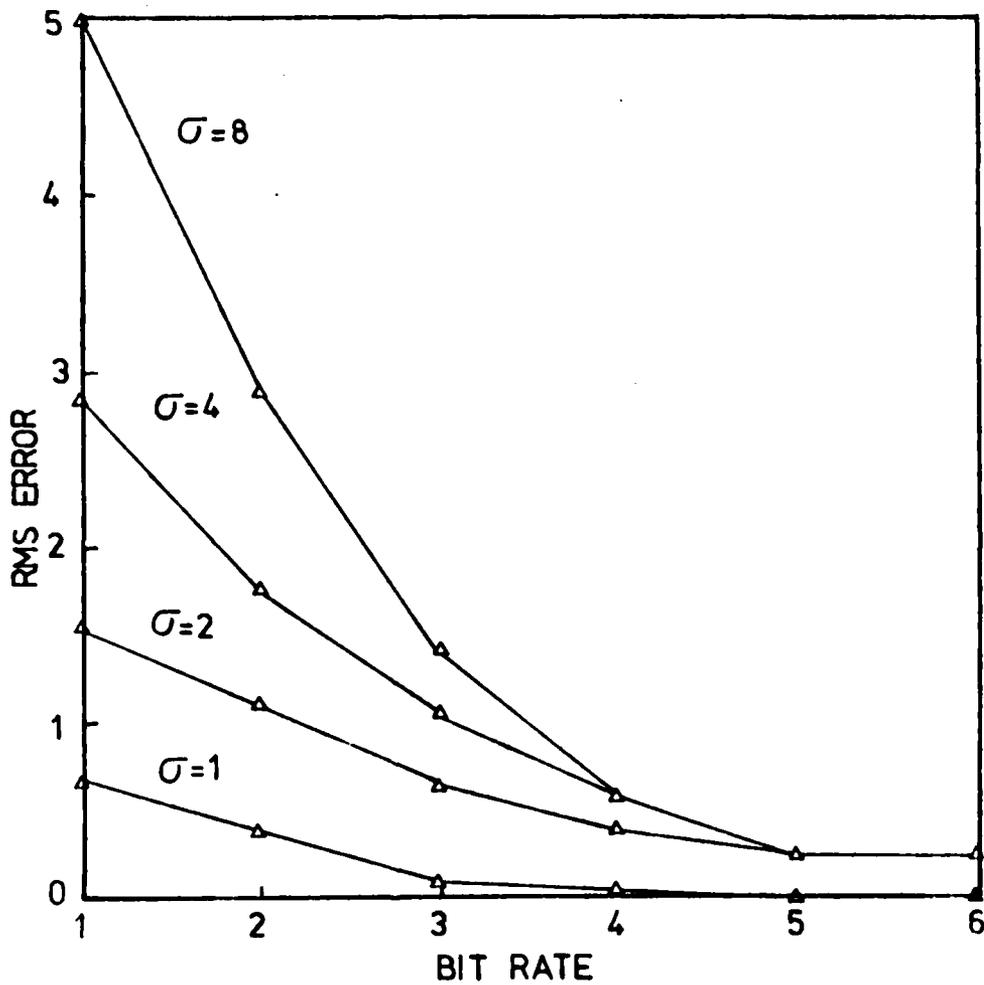
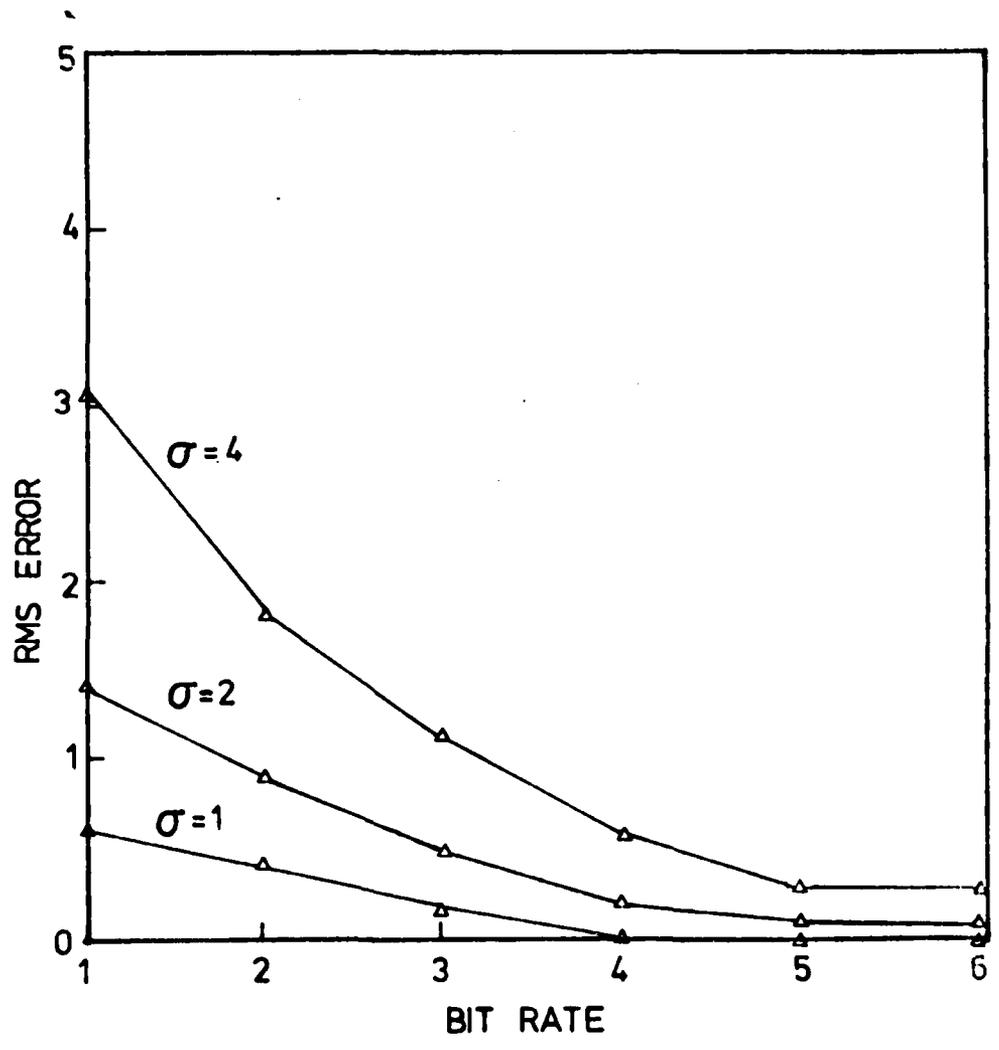
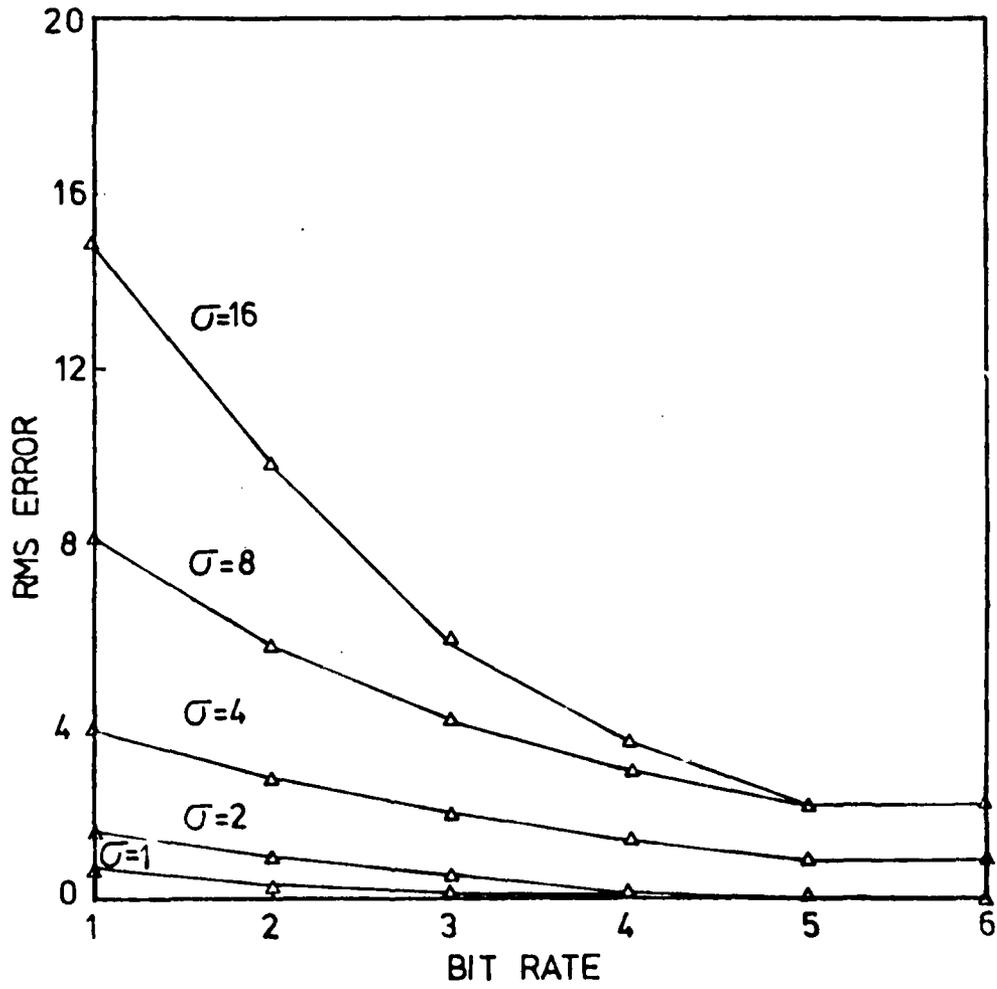


Figure 6.3 Block RMS Error vs. Bit Rate
 MD, SM, LS
 (a) First LANDSAT Image



(b) Second LANDSAT Image



(c) Face Image

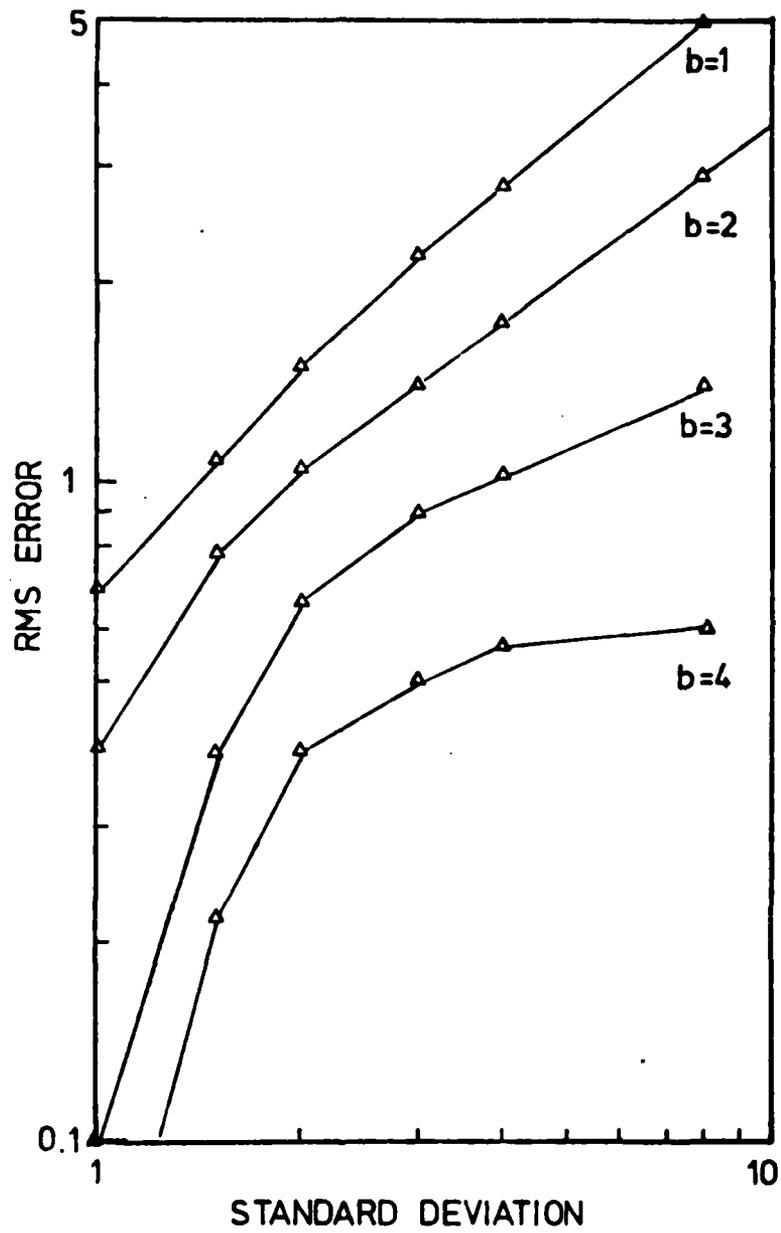
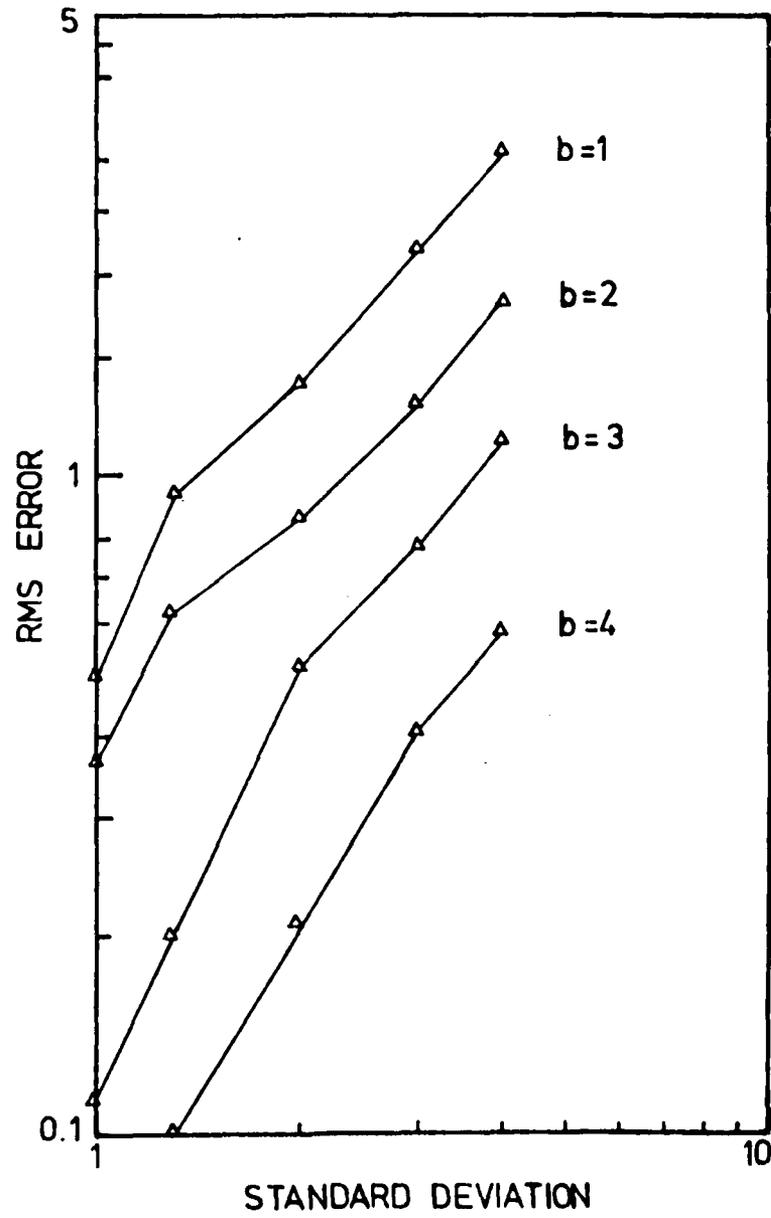
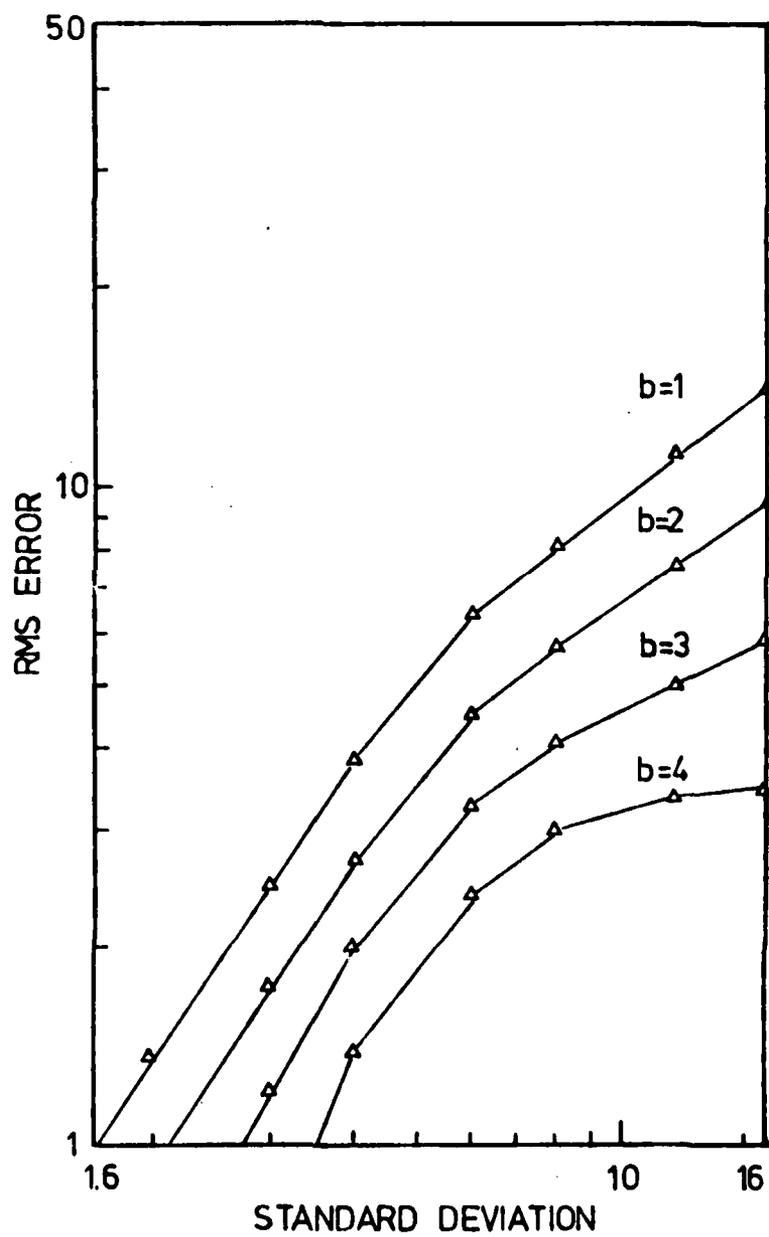


Figure 6.4 Block RMS Error vs Standard Deviation
 MD, SM, LS
 (a) First LANDSAT Image



(b) Second LANDSAT Image



(c) Face Image

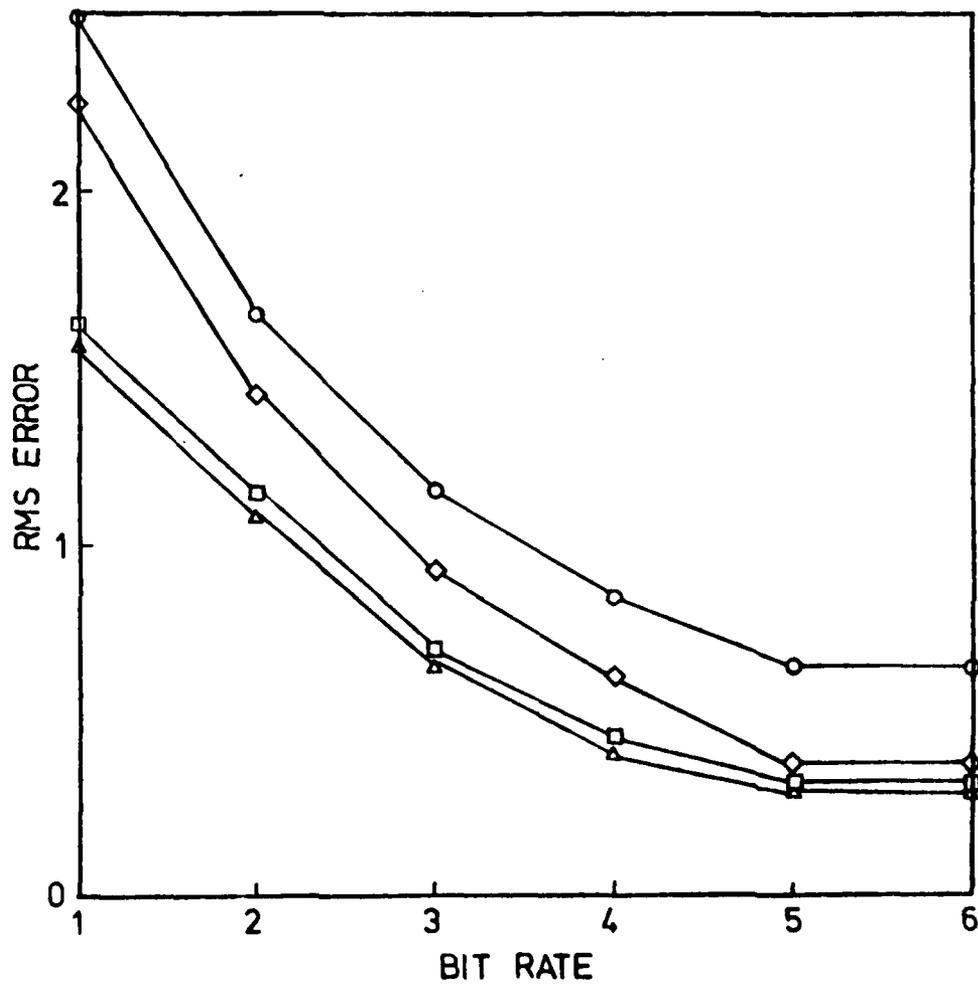


Figure 6.5 Comparisons Among DPCM Predictors ($\sigma = 2$)

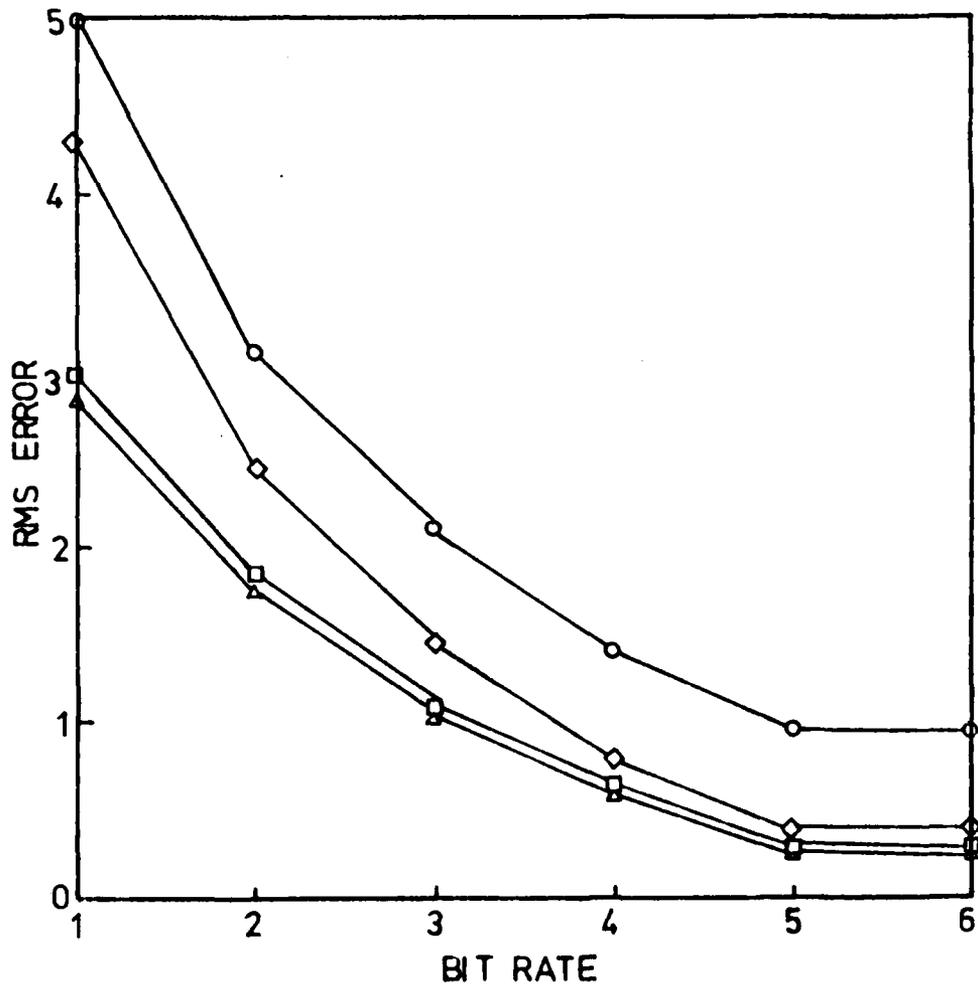
◇ CD, FM, MV or LS

○ CD, SM, MV, or LS

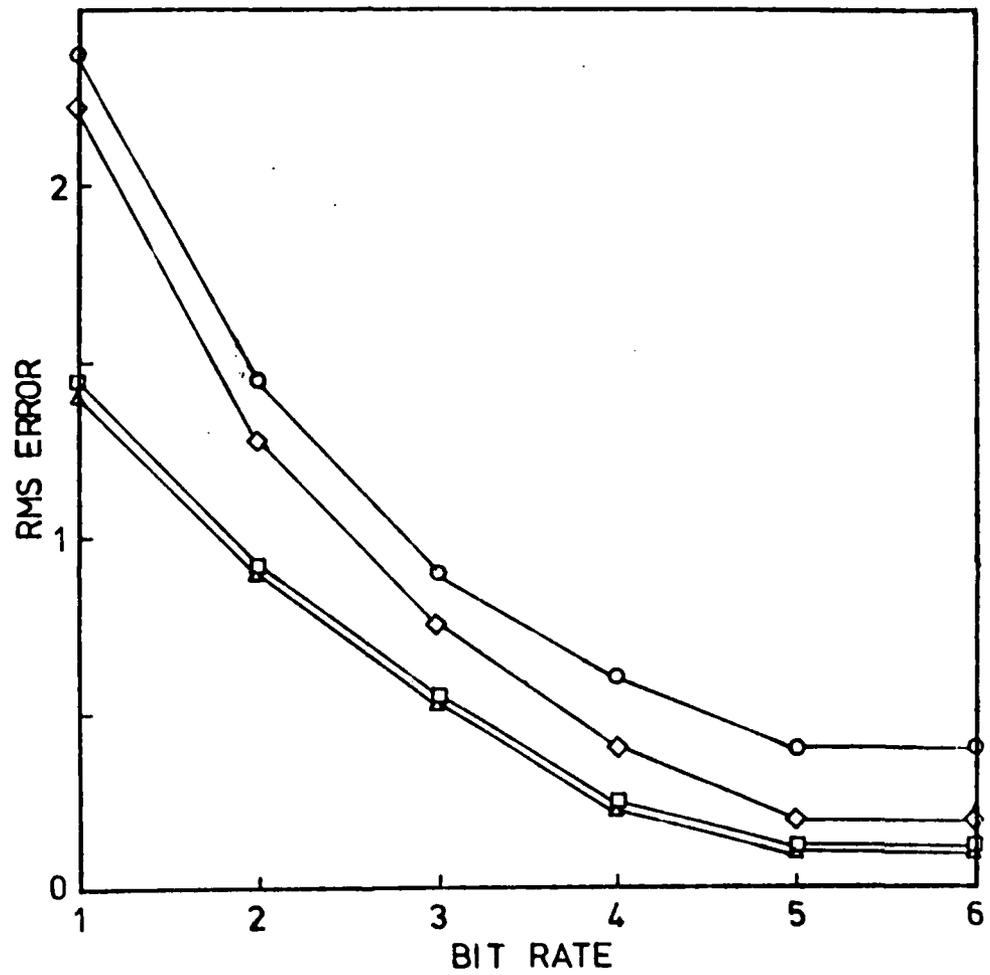
△ MD, FM or SM, LS

□ MD, FM or SM, MV

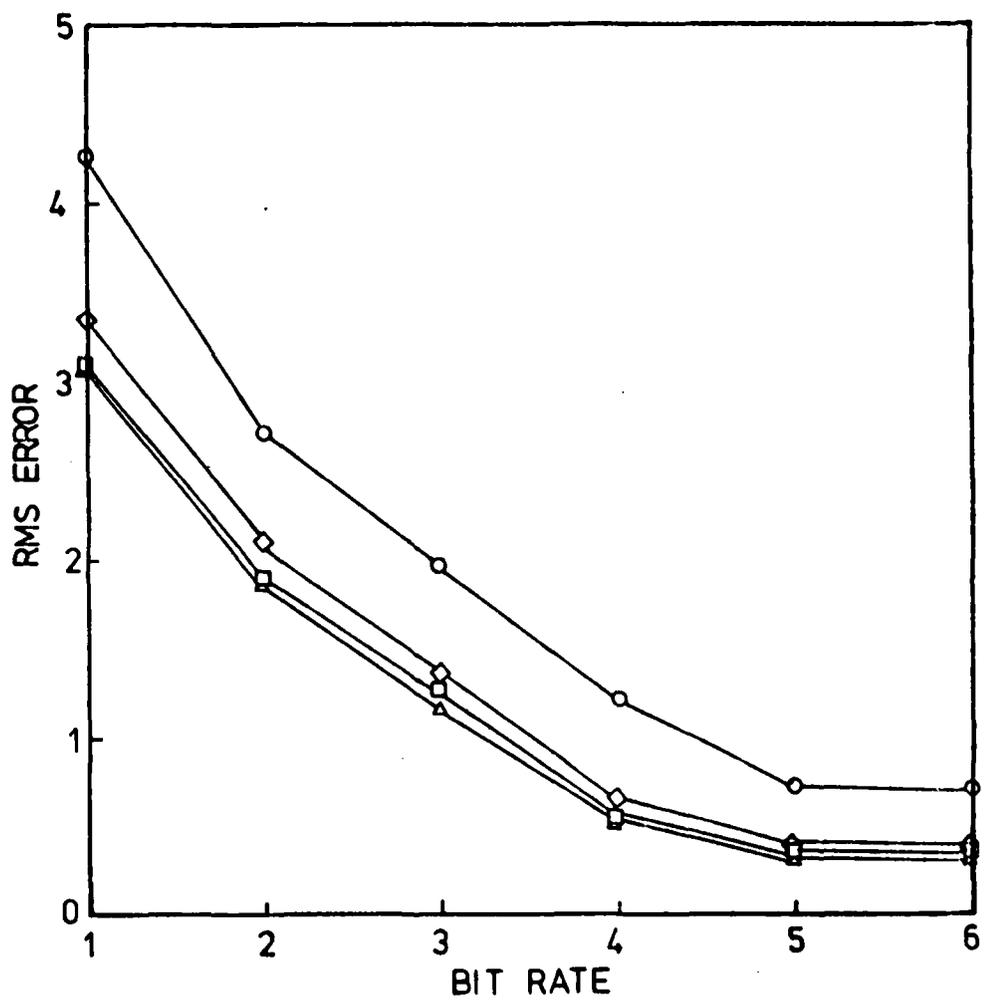
(a) First LANDSAT Image



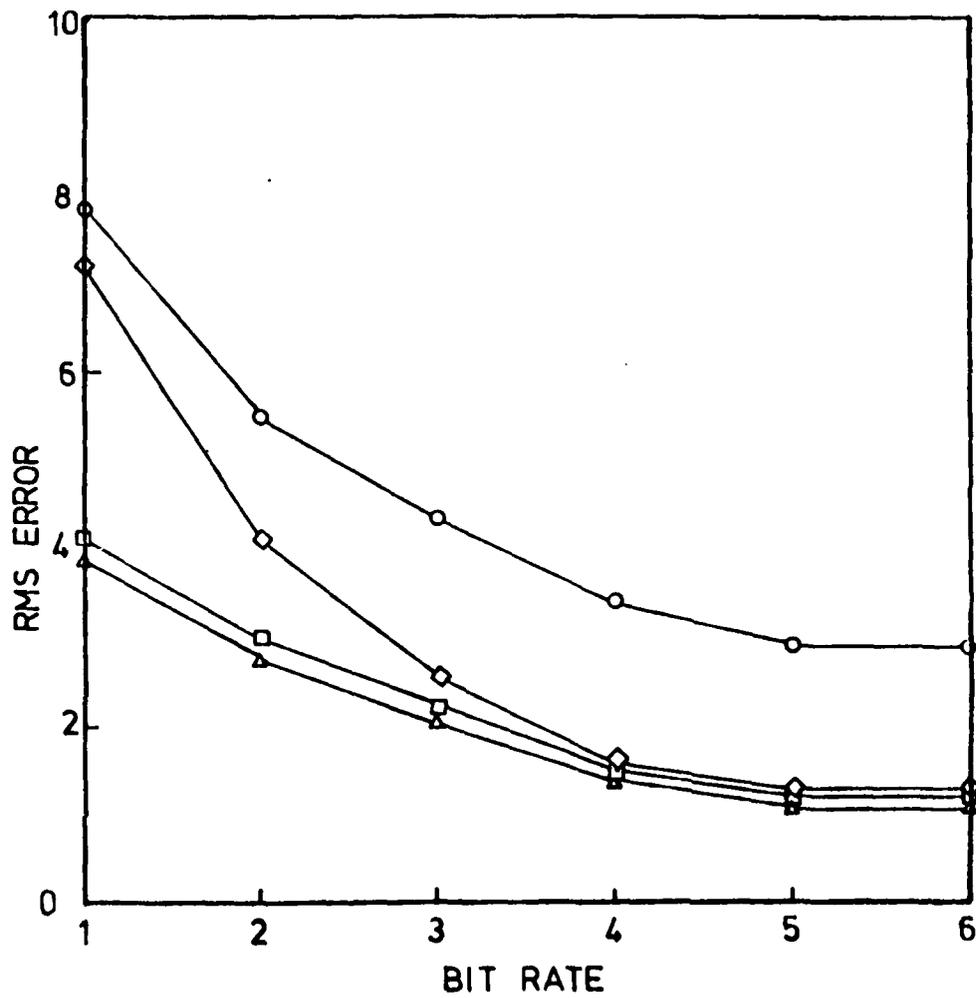
(b) First LANDSAT Image ($\sigma = 4$)



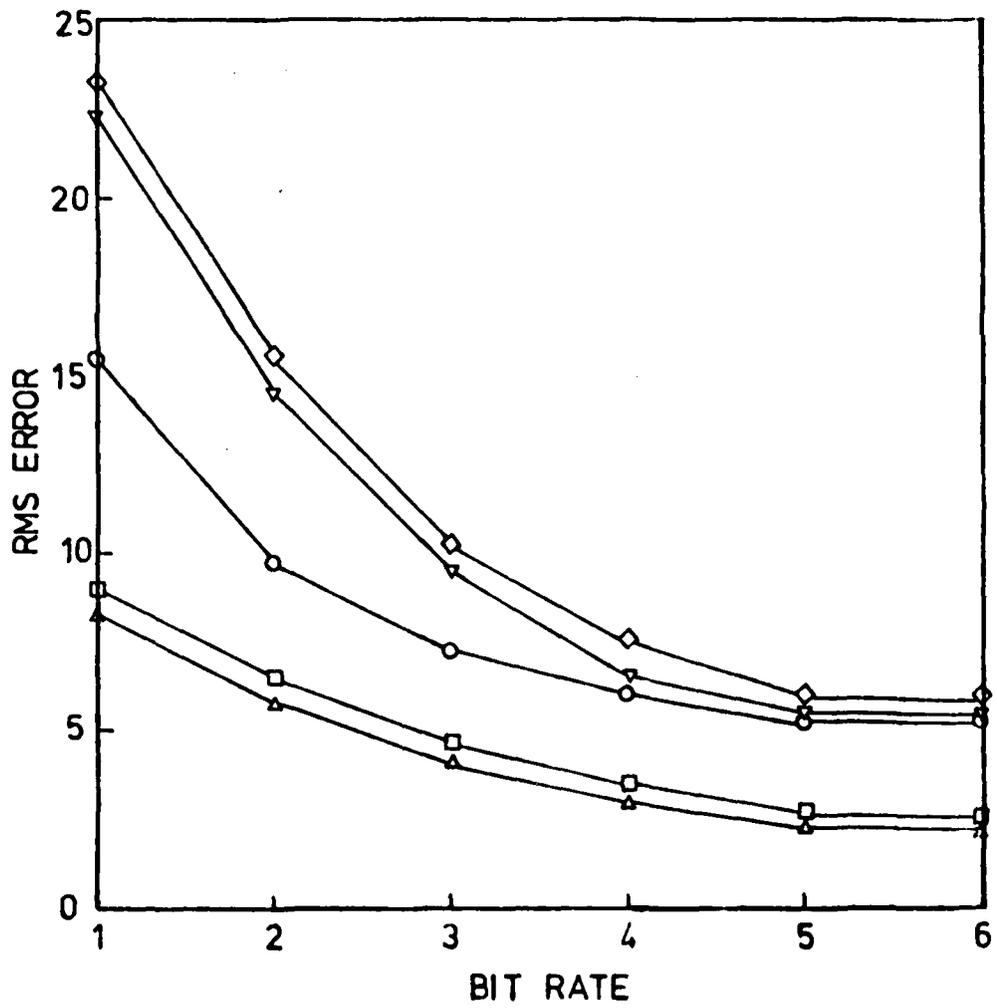
(c) Second LANDSAT Image ($\sigma = 2$)



(d) Second LANSAT Image ($\sigma = 4$)



(e) Face Image ($\sigma = 4$)



(f) Face Image ($\sigma = 8$)

◇ CD, FM, MV

▽ CD, FM, LS

RMS errors among predictors associated with the combined method or in the quality of the corresponding resulting pictures. The difference, however, may become noticeable with noisier images.

Fig. 6.6 shows the reconstructed pictures compressed to 1.5 bits per picture element for the predictor associated with the combined DPCM, sloped model, least squares estimates.

II. Buffer constrained allocation; Actual errors

Fig. 6.7 shows the reconstructed pictures for 2.0 and 1.5 bits per picture element using the predictor associated with the combined DPCM, sloped model, least squares estimates. Fig. 6.8 shows plots of the buffer state as the blocks in the image are allocated bits for the predictor associated with the combined DPCM, sloped model, least squares estimates. Both the non-buffer constrained and the buffer constrained case are shown along with the corresponding total RMS errors for all the blocks in the image. Notice that the reconstructed pictures with buffer constraints do not show a significant deterioration in quality as compared to the non-constrained case for the buffer size used.

III. Non-Buffer constrained allocation; Fitted errors

Images compressed at 2.0 and 1.5 bits per picture element using the fitted error versus bit rate functions for bit allocation did not show significant degradation in quality as compared to that obtained using the actual errors shown in Figs. 6.2 and 6.6, which indicate that a model relating the fitting parameters with the sample variance could be used successfully with a significant decrease in the computational burden. Fig. 6.9 shows plots of Buffer state versus blocks encoded for the predictor associated with the combined DPCM, sloped model, least squares



Figure 6.6 Reconstructed Pictures Using Non-Buffer
Constrained Allocation and Actual Errors.
Compression is 1.5 b/p

(a) First LANDSAT Image
Predictor: MD, SM, LS



Figure 6.6 (continued)

(b) Second LANDSAT Image
Predictor MD, SM, LS



Figure 6.7 Reconstructed Pictures Using Buffer Constrained Allocation and Actual Errors

(a) First LANDSAT Image
Predictor: MD, SM, LS
2.0 b/p



Figure 6.7 (continued)

(b) Second LANDSAT Image
Predictor: MD, SM, LS
2.0 b/p.



Figure 6.7 (continued)

(c) First LANDSAT Image
Predictor: MD, SM, LS
1.5 b/p



Figure 6.7 (continued)

(d) Second LANDSAT Image
Predictor: MD, SM, LS
1.5 b/p.

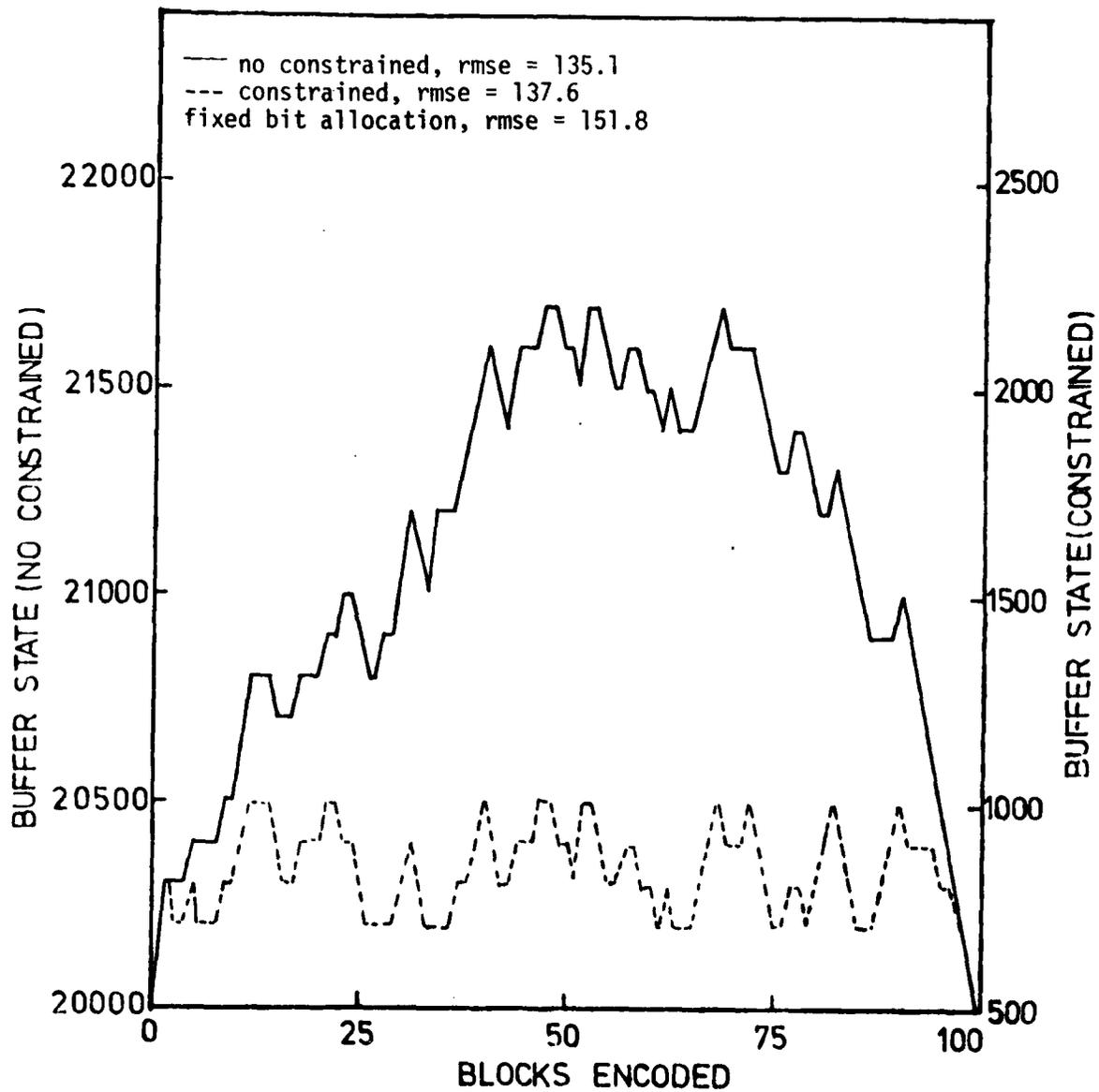
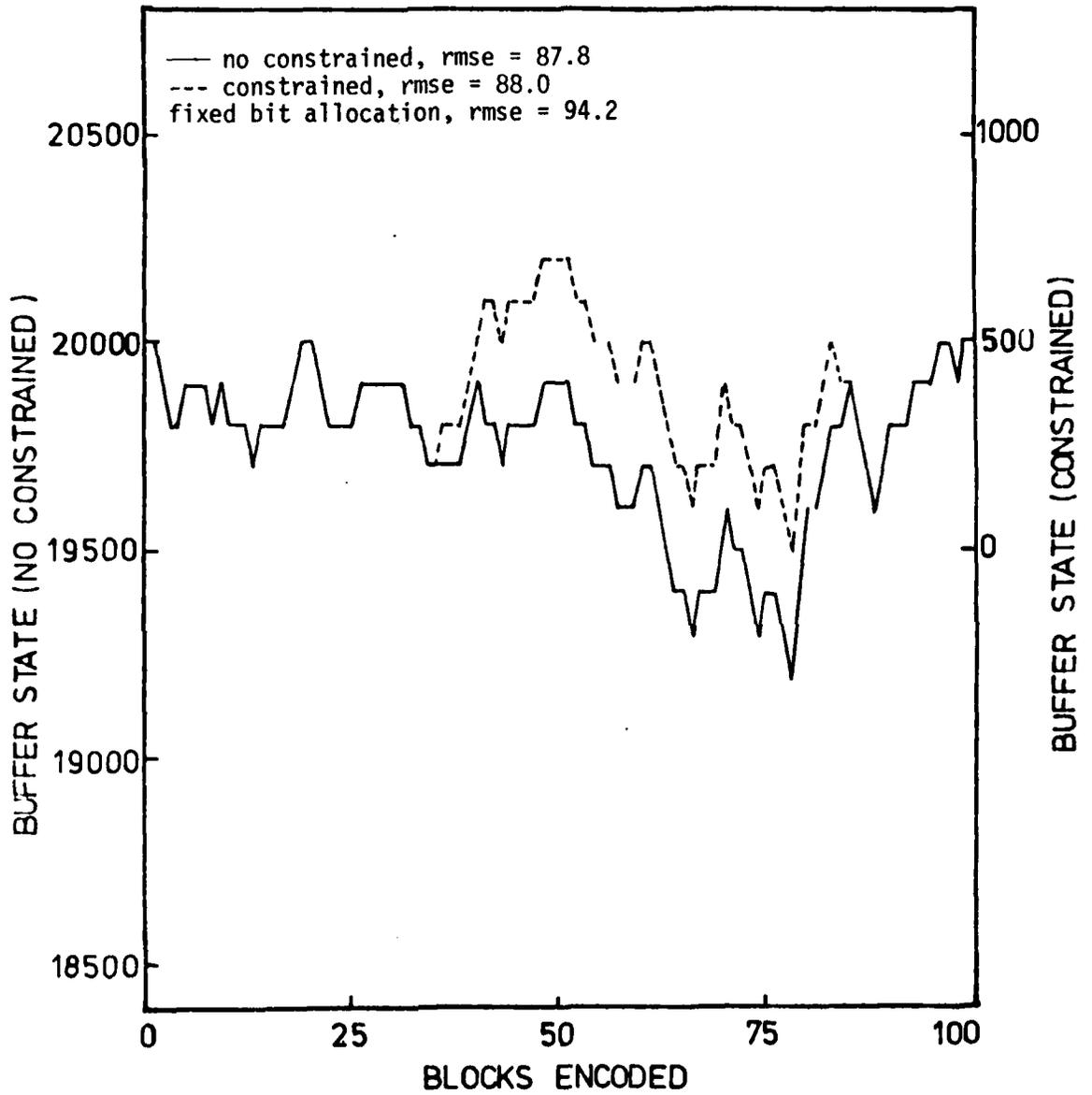


Figure 6.8 Buffer State vs Number of Blocks Uncoded
 MD, SM, LS
 (a) First LANDSAT Image



(b) Second LANDSAT Image

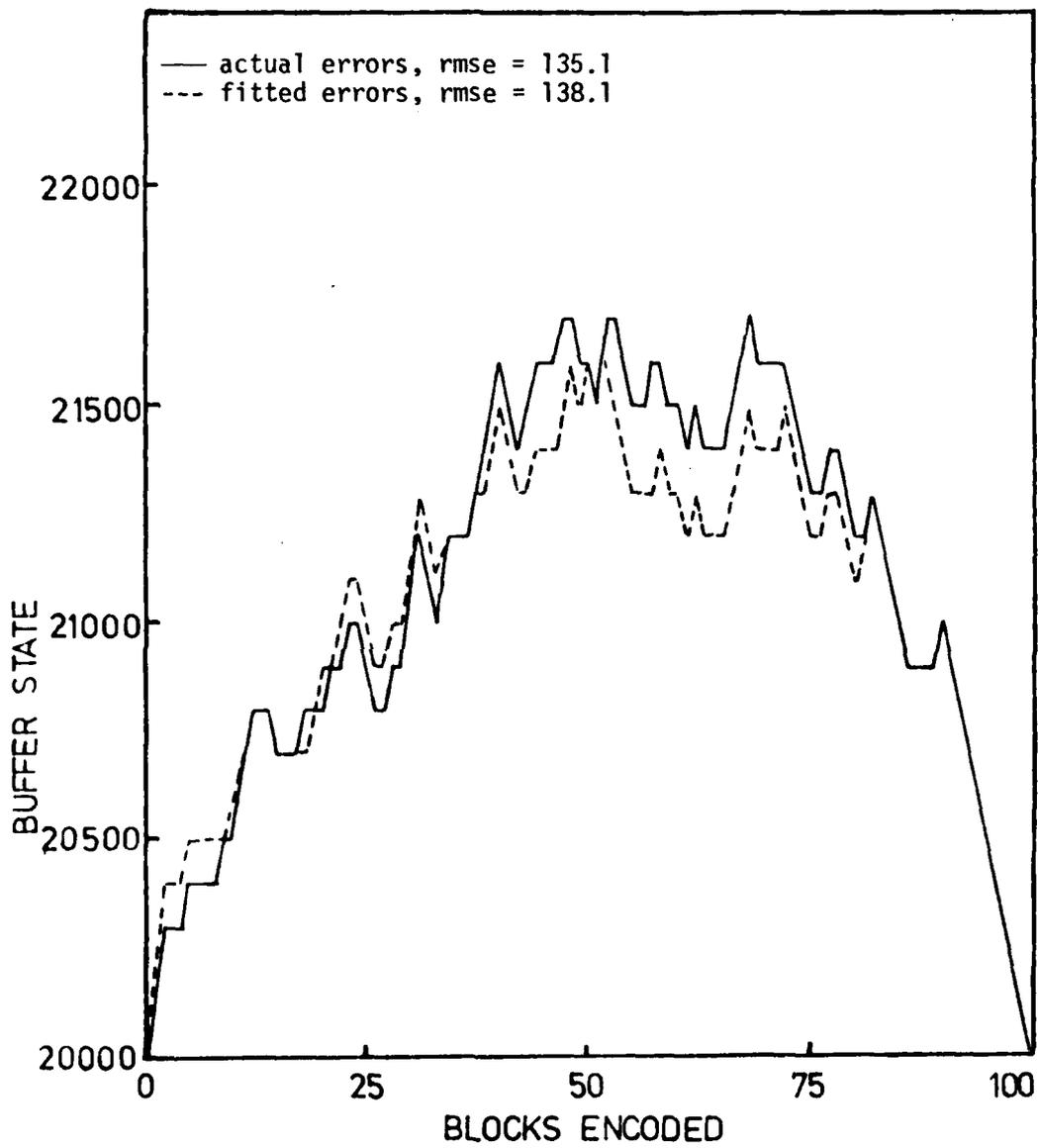
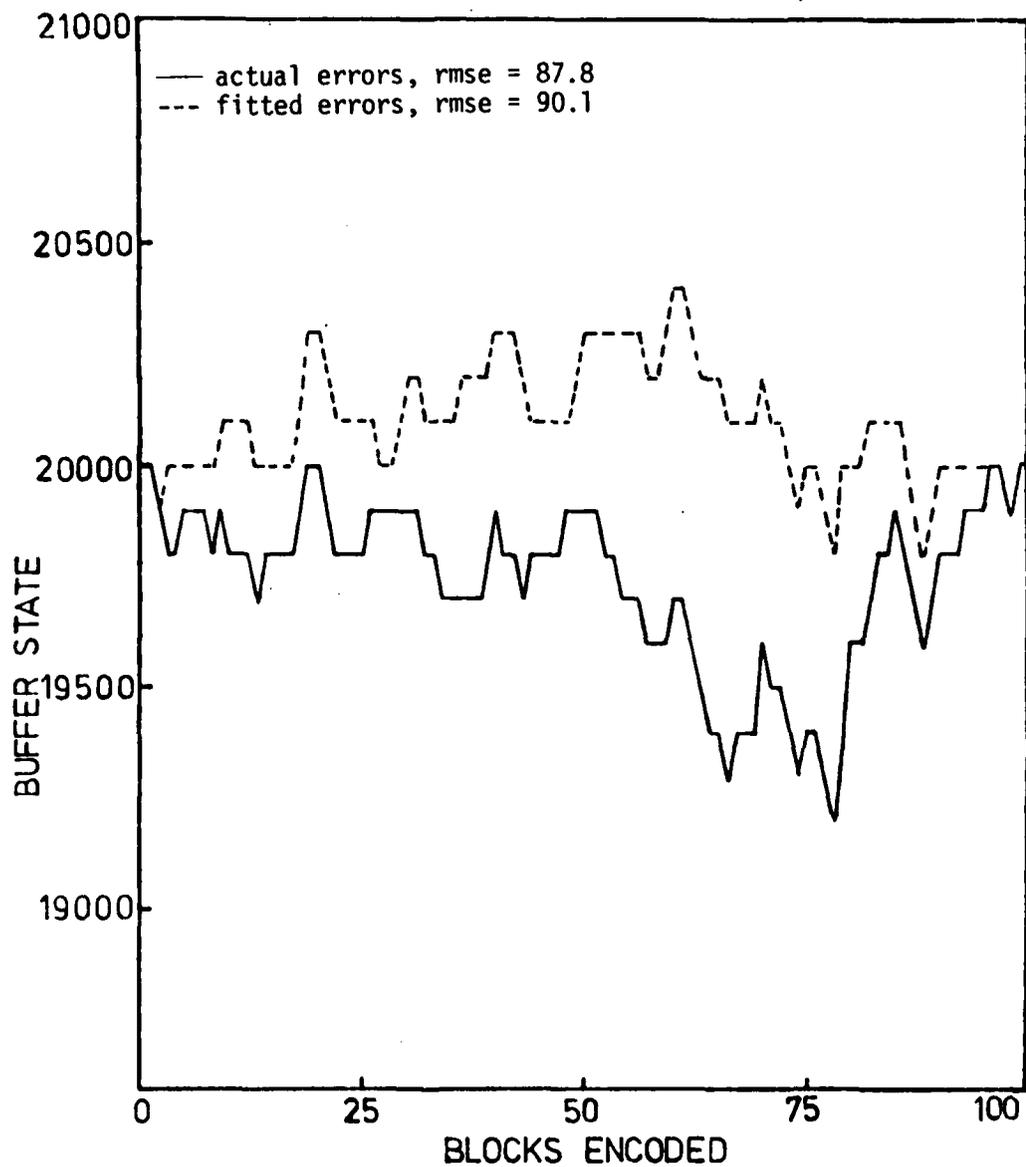


Figure 6.9 Buffer State vs Number of Blocks Encoded
 MD, SM, LS

(a) First LANDSAT Image



(b) Second LANDSAT Image

estimates. Both the actual error case and the fitted error case are shown along with the corresponding total RMS error.

IV. Buffer constrained allocation; Fitted errors

As in the previous case, the quality of the reconstructed images compressed at 2.0 and 1.5 bits per picture element was similar to that obtained using the actual error versus bit rate functions.

V. Preprocessing and Postprocessing

Fig. 6.10 shows the reconstructed pictures compressed to 2.0 and 1.5 bits per picture element with no buffer constraints; actual errors, combined DPCM, sloped model, least squares estimates, after being postprocessed.

Fig. 6.11 shows the original pictures after being preprocessed. Fig. 6.12 shows the reconstructed pictures compressed to 2.2 bits per picture element with no buffer constraints, actual errors, combined DPCM, sloped model, least squares estimates. The corresponding postprocessed pictures are shown in Fig. 6.13.

The effect of preprocessing or postprocessing could have been more noticeable with noisier images.

VI. Comparisons between variable and fixed bit allocation

Table 6.1 shows the total RMS errors obtained with a fixed bit assignment procedure for a compression ratio of 2.0 bits per picture element using the predictor associated with the combined DPCM, sloped model, least squares estimates as compared to those obtained using a variable bit assignment procedure with the same predictor for the cases of non-buffer constrained and buffer constrained bit allocation using the actual and fitted error versus bit rate functions. We can observe the improvement in RMS error obtained using the variable bit assignment procedure.



Figure 6.10 Postprocessing of the Reconstructed Pictures
Obtained by Using Non-Buffer Constrained
Allocation and Actual Errors.

(a) First LANDSAT Image
Predictor: MD, SM, LS
2.0 b/p



Figure 6.10 (continued)

Second LANDSAT Image
Predictor: MD, SM, LS
2.0 b/p.



Figure 6.10 (continued)

(c) First LANDSAT Image
Predictor: MD, SM, LS
1.5 b/p.



Figure 6.10 (continued)

(d) Second LANDSAT Image
Predictor: MD, SM, LS
1.5 b/p.

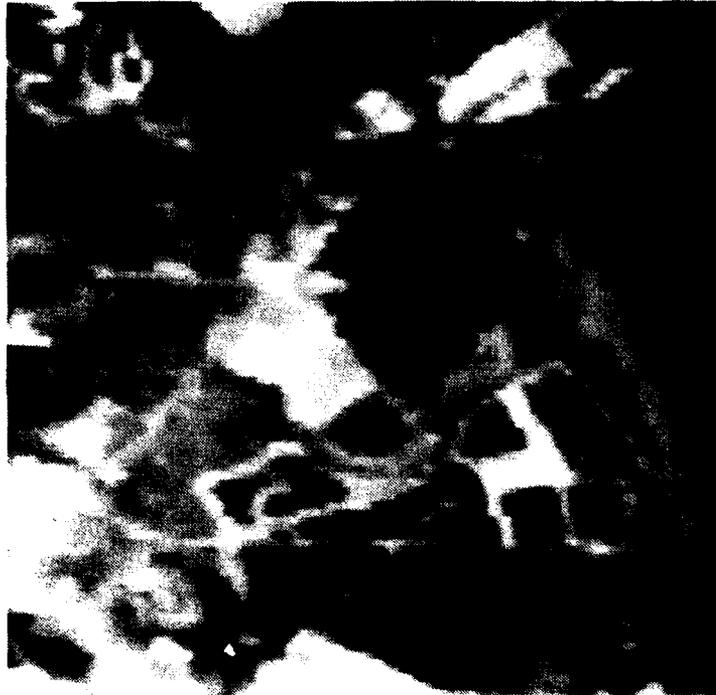


Figure 6.11 Preprocessing of the Original Pictures Using the Slope Facet Model.

(a) First LANDSAT Image.



Figure 6.11 (continued)

(b) Second LANDSAT Image.

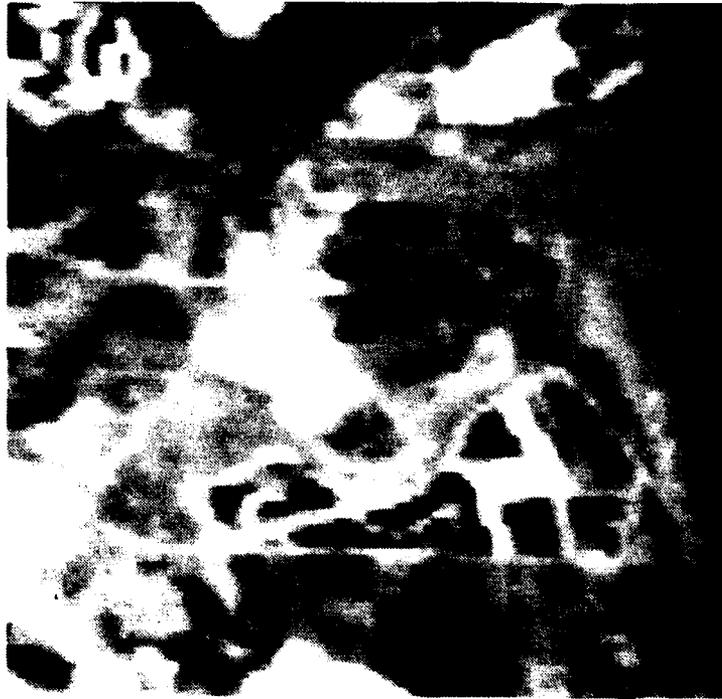


Figure 6.12 Reconstructed Pictures After Compressing the Preprocessed Pictures of Figure 6.11 Using Non-Buffer Constrained Allocation and Actual Errors. Compression is 2.0 b/p.

(a) First LANDSAT Image
Predictor: MD, SM, LS

AD-A094 678

KANSAS UNIV/CENTER FOR RESEARCH INC LAWRENCE
A STUDY OF ADAPTIVE IMAGE COMPRESSION TECHNIQUES, (U)
FEB 80 R M HARALICK, R L KLEIN

F/6 17/2

F33615-78-C-1545

UNCLASSIFIED

AFWAL-TR-80-1072

NL

2 of 4

AD
AC34678

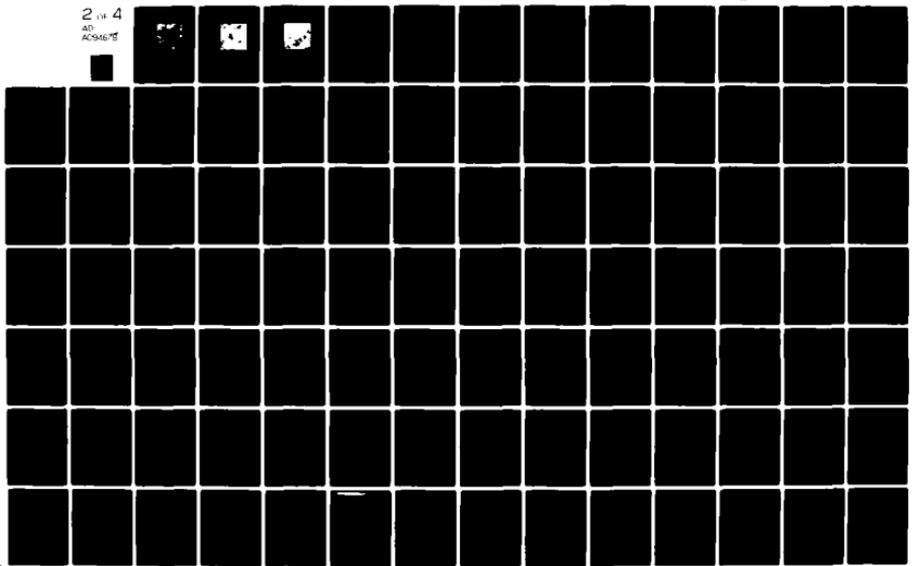




Figure 6.12 (continued)

(b) Second LANDSAT Image
Predictor: MD, SM, LS

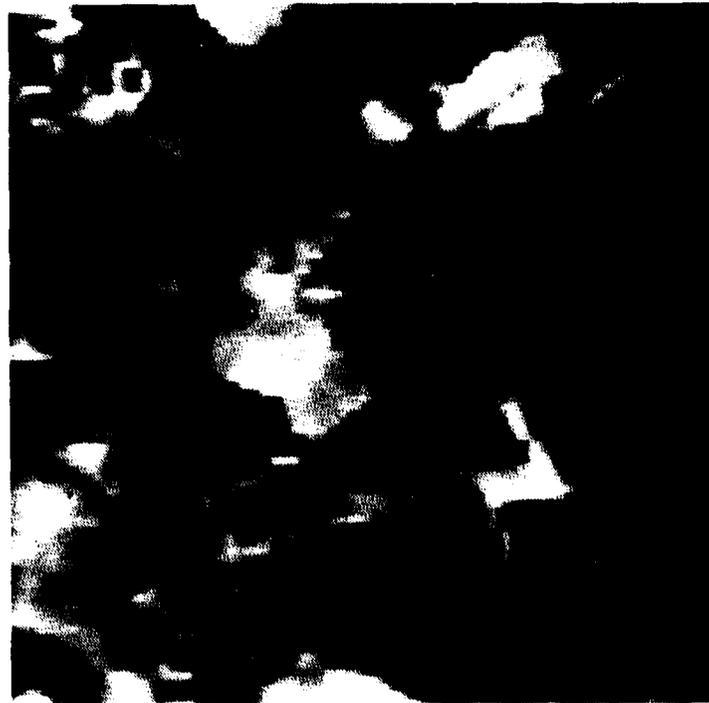


Figure 6.13 Postprocessing of the Pictures Shown in Figure 6.12.

(a) First LANDSAT Image



Figure 6.13 (continued)

(b) Second LANDSAT Image.

Table 6.1

Total RMS Errors Obtained with a Fixed Bit Assignment Procedure

	NB AE	BC AE	NB FE	BC FE	FIXED
FIRST LANDSAT	135.1	137.6	138.1	141.0	151.8
SECOND LANDSAT	87.8	88.0	90.1	90.6	94.2

Shows the total RMS errors obtained with a fixed bit assignment procedure for a compression ratio of 2.0 bits per picture element using the predictor associated with the combined DPCM, sloped model, least squares estimates as compared to those obtained using a variable bit assignment procedure with the same predictor for the cases of non-buffer constrained and buffer constrained bit allocation using the actual and fitted error versus bit rate functions.

NB Non-Buffer Constraints
 BC Buffer Constraints
 AE Adaptive Actual Error Curves
 FE Fitted Error Curves

7. Conclusions

The problem of Adaptive Image Data Compression has been discussed. Procedures for solving the causal non-buffer constrained and buffer constrained bit allocation problem have been suggested and experimental results for the optimal Non-causal bit allocation procedure using a number of DPCM compression techniques were presented for the cases of non-buffer and buffer constrained bit allocation. The performance of the optimal non-causal approach is a least upper bound on any causal approach and provides us with a way of comparing the performance by different causal procedures.

Several questions have been answered; others remain yet to be answered. It has been shown that the variable bit assignment scheme yields smaller RMS errors than those obtained with a fixed bit assignment scheme. It was also experimentally found that the buffer constrained non-causal scheme performs well even for small size buffers (2.5% of the minimum size that guarantees no constraints). One question to be answered is how small can we make the size of the buffer and still obtain a significantly better performance than that obtained using fixed bit assignment procedures. It is not hard to see that the smallest size we can allow if we are to avoid overflowing the buffer is that size necessary for holding the encoded bits for one block at the given compression ratio, in which case the bit allocation becomes fixed for all the blocks. It is a matter then of finding out an optimal trade off between buffer size and performance.

It has been found that enough correlation exists between RMS error and block variance to allow a model to be used to estimate the RMS

error versus bit functions instead of computing the actual errors with the corresponding savings in computational time. There is no significant degradation in the quality observed in the images that were compressed using the fitting functions instead of the actual ones.

With respect to the DPCM techniques, the combined open and closed loop DPCM performs significantly better than a simple closed loop 2-D DPCM. There was no observed significant difference in the performance among different predictors in the Combined method for the level of noise present in the original pictures. It must also be pointed out that an improvement in the initialization of the DPCM procedure can be carried out to achieve better effective compression ratios. Namely, the DPCM in any block can be initialized using the last line in the previously DPCM'ed north block.

8. REFERENCES

1. Haralick, R. M., and L. Watson, "A Facet Model for Image Data," Pattern Recognition Conference, August 1979, Chicago, Illinois.

PART II

PART II

Adaptive Coding of Images Using Transform Coding Techniques

Adaptive coding of images can be done in two general ways. The first is to perform adaptive operations on original image representations. The second is to perform adaptive operations on a transform representation. Part I has described adaptive methods applied to DPCM coding of the original image. Part II describes methods based on a transform representation of the image.

We begin this part with a description of how the transform of an image is taken.

1. Transform Coding Procedure: Non-Casual Case

An original image is subdivided into blocks. The block size chosen is 10×10 and the image size is 100×100 , which gives 100 blocks with 100 pixels in each block. All processing is done block by block, i.e., from block 1 to block 100. The blocks are numbered sequentially, column by column, as shown in Figure 1.1. The input image is real and all arithmetic is floating point.

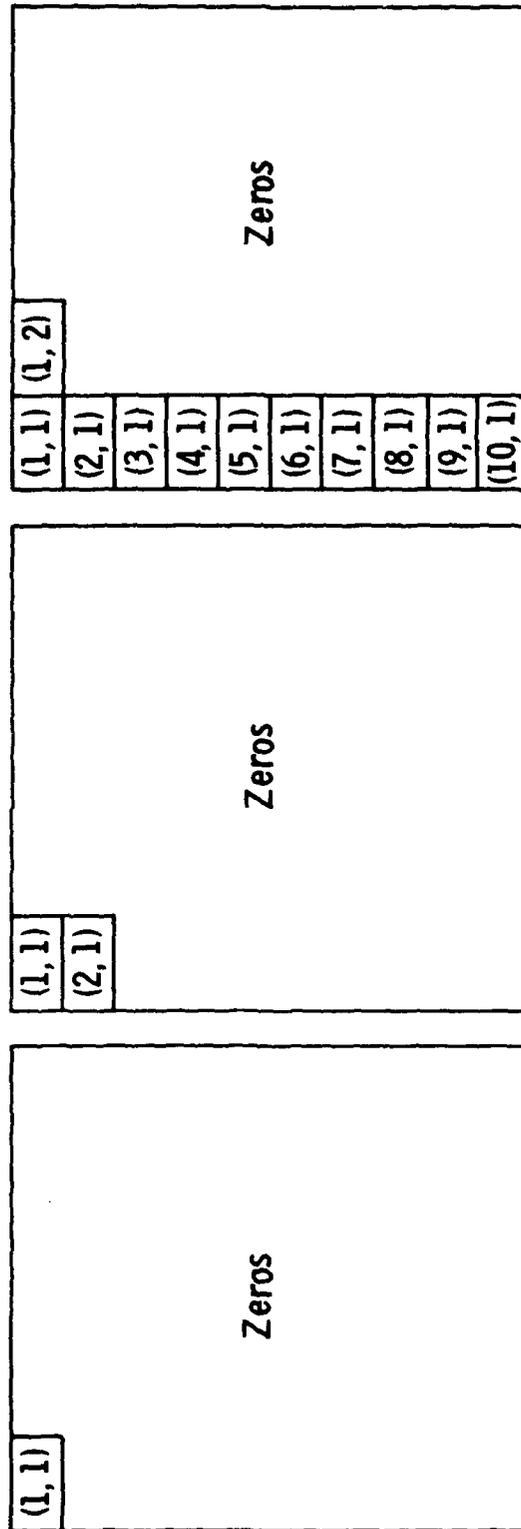
After a study of Griswold and Haralick [1], the Discrete Cosine Transform was selected since it had given best results.

A block of image data is read in and Discrete Cosine transformed using the fast transform technique described in [1]. At the same time the mean and variance of each pixel position in a block of the transformed image is updated. This yields two 10×10 arrays containing mean and variance information, e.g., after all 100 blocks have been processed the (1,1) position in the mean array will have the mean of all the pixels in the (1,1) position in each block and is similar for the variance array. This information will be later used by the Max quantizer. The pixels in the transformed block shall be referred to as components since they correspond to frequency components in a general transformed signal. Thus we have one hundred components per block.

Next, each component in a block is quantized using six bits (the upper limit of the max quantizer program). Simultaneous with the quantization, we calculate the error vs. component curve, i.e., what is the error if only one component is transmitted and the remainder are assumed zero; what is the error if two components are transmitted, etc.? This segment of "filtered images" is shown in Figure 1.2.

1	11	21		91
2	12	22		92
3	13	23		93
.
.
.
.
.
.
10	20	30		100

Figure 1.1 Image Blocking and Block Sequence Specification



an 11 component block

a two component block

a one component block

Figure 1.2 Truncated Transform Images of a Single Block

The components are taken in a specific columnwise, increasing-frequency order as shown in Figure 1.2. The first column continues to the top of the second column, etc.

So after this processing, we end up with 100 transformed and quantized blocks and corresponding to each of these an error vs. component curve. Since each component corresponds to six bits, the error vs. component curve can be alternatively represented as an error vs. bits curve as shown in Figure 1.3. The error here is defined to be RMS error.

Since storage of each of these error vs. bits curves requires a 100 x 100 matrix, it is desirable to find a simpler representation. One such representation is a polynomial. To obtain experimental data a sixth order polynomial was fitted to each of the curves, the "raw" one and the "fitted" one. (See Data Set 3).

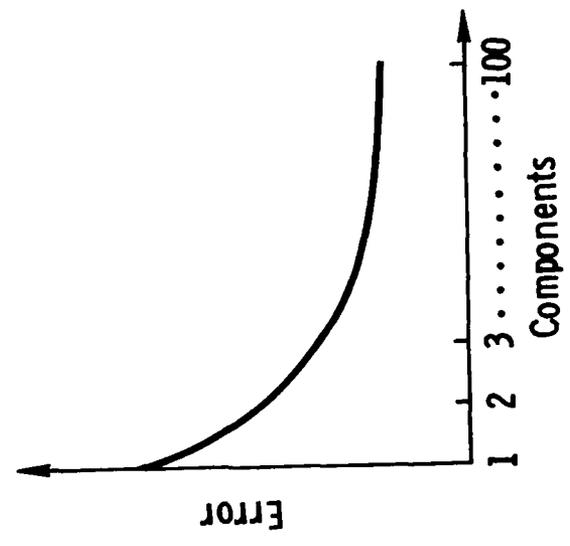
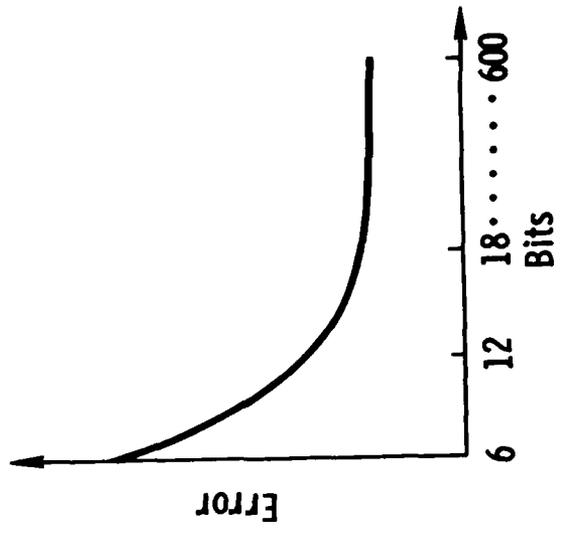


Figure 1.3 Error vs. Component and Error vs. Bit Curves

2. Allocation of Bits

Given the transformed images and the error vs. bits curves described earlier, we are ready to proceed with bit allocation. This can be done for transformed images in each of the four basic ways described in Part I, namely:

- (1) Non-Causal Optimal Allocation
- (2) Non-Causal Optimal Allocation with Buffer Constraints
- (3) Casual Adaptive Allocation with Buffer Constraints
- (4) Non Adaptive Allocation

Allocation methods (1) and (2) can be computed using the same software as was used for Part I. These are described in Appendices A.2 and A.3. Only one additional item of information is required in this transformed case which is the compression ratio desired. It determines the total number of bits to be allocated to the entire frame. (This number is defined as TTOTAL in the computer program.)

In concept we are now ready to proceed and obtain the computed optimal bit allocations for (1) and (2). Unfortunately, our storage requirement is much greater than is required for the DPCM case since an array of dimensions TTOTAL x No. of Blocks is necessary. To solve this impractical storage requirement on the PDP-15, the image was divided into four equal subparts as shown in Figure 2.1.

TTOTAL was equally allocated to each of the four parts, and similarly for the buffer. Now each quarter of the original (100 x 100) picture can be processed separately and the final bit allocations for the entire image are those calculated on the basis of four-part subpartitions. This is obviously not optimal with respect to the whole image but is very close when the number of partitions is small, e.g., four.

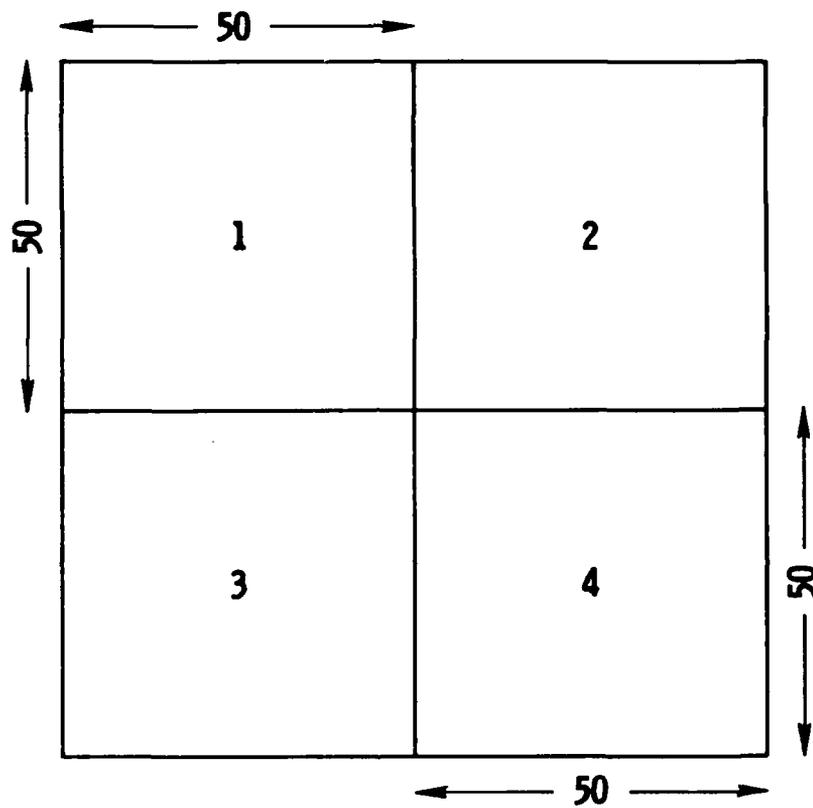


Figure 2.1 Sub-Partitioning of the Transform Image

Thus each block now has an optimum number of bits (components) allocated to it. This optimum number of components is retained and the others are replaced by zero.

The output image can now be obtained by taking the inverse discrete cosine transform of each block as described in [1].

Summarizing, all original images (100 x 100) were quantized at 8 bits/pixel. Our compression then consists of two stages:

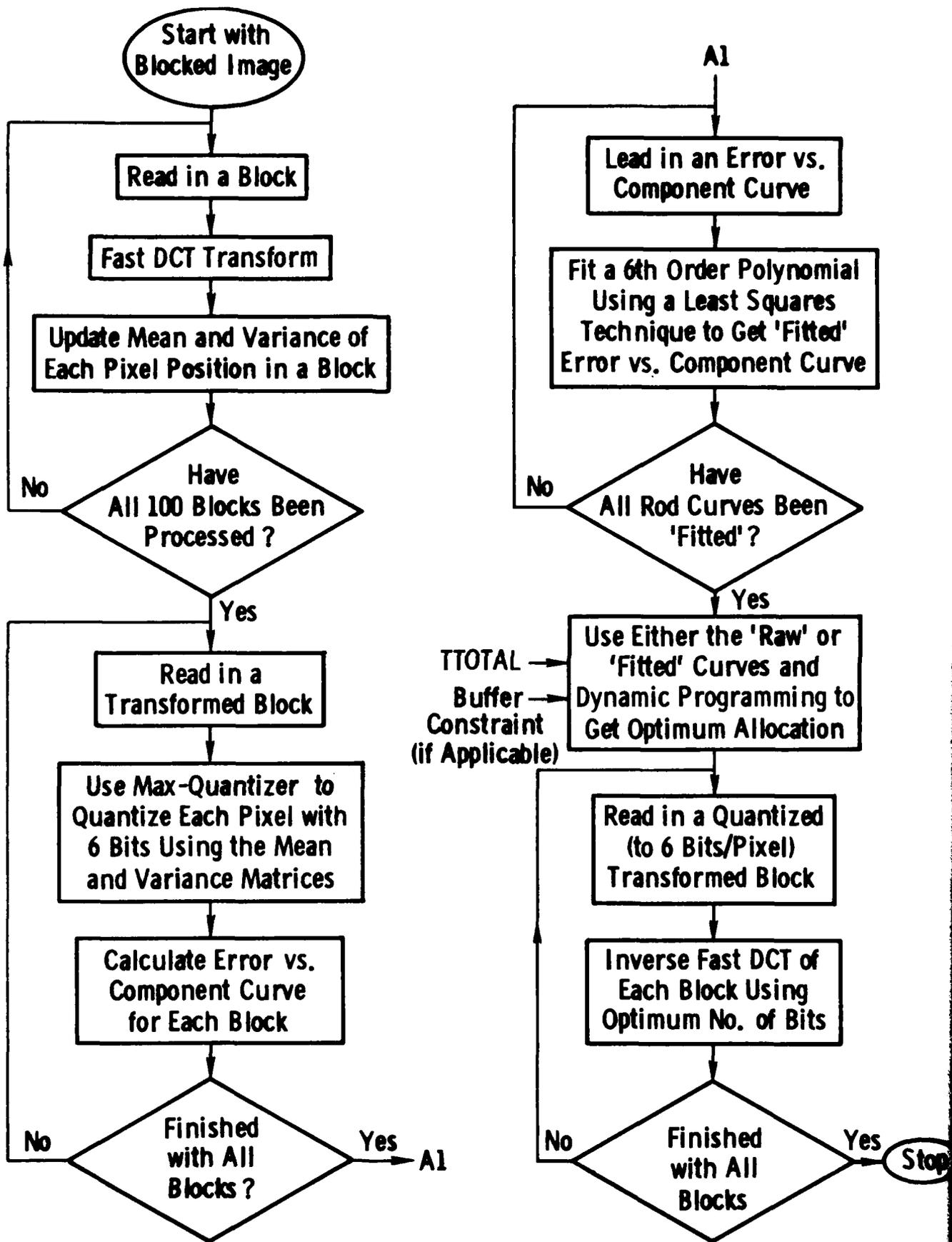
- (1) Using the Max quantizer to quantize each pixel using 6 bits followed by
- (2) A non-casual optimal adaptive allocation using dynamic programming both with and without buffer constraints.

The entire procedure is summarized in the flowchart on the following page.

EXPERIMENTS PERFORMED

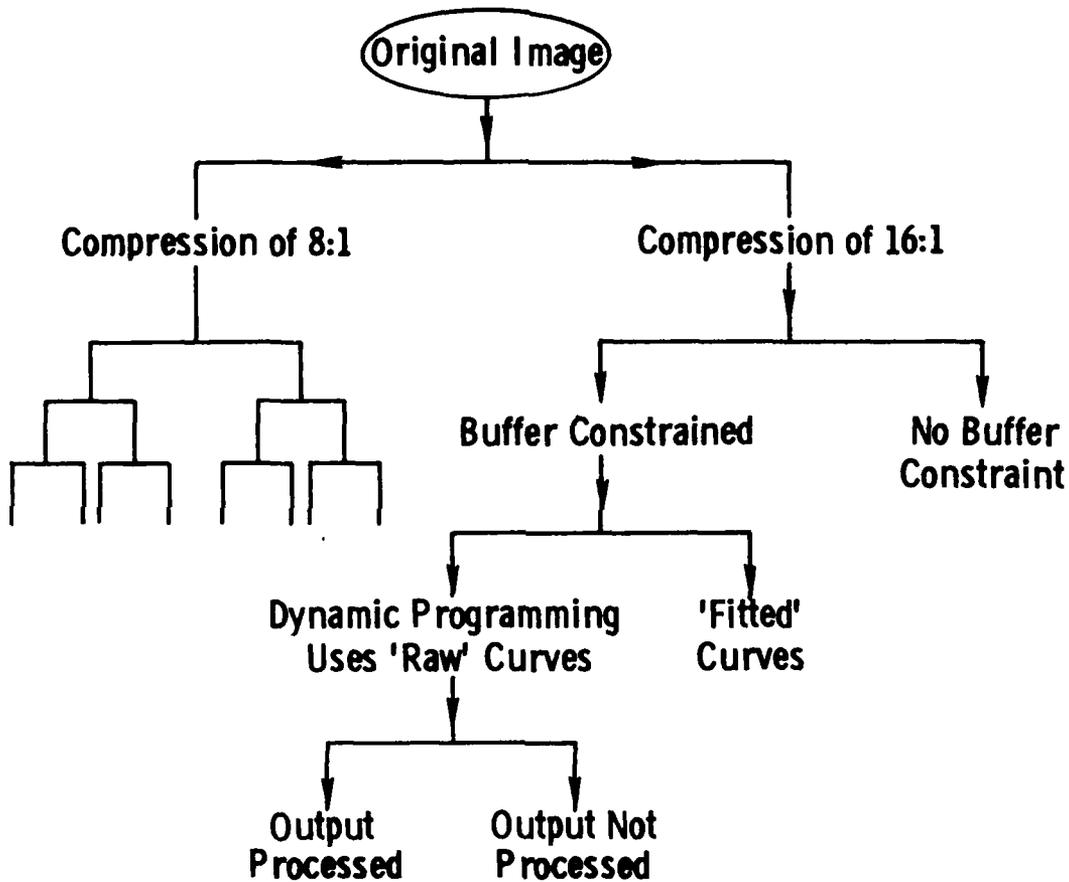
The following experiments were selected to test for potential compression improvements and to compare with non-adaptive results:

- (1) Adaptive encoding with or without a buffer constraint. When a buffer constraint was included it was taken to be 10% of TTOTAL.
- (2) Output image post processing (smoothing). The slope faceted algorithm (see Part I) was used to determine available improvements from smoothing the output to remove blur and computational discontinuities introduced by the optimization procedure.
- (3) Variable Compression Ratios: Two different compression ratios were selected, 8:1 and 16:1.
- (4) Representation of the error vs. bits curve: Use of both the "raw" and "fitted" error vs. components curve was selected to determine



if reduced representation (storage) of the image complexity could be gained without significant degradation in overall quality.

The following tree diagram summarizes the experiments performed using all the above options.



This gives a total of 16 images for original image. Three original images were used; they will be called KCALIF, CCALIF, and LADY3.

In addition to the above adaptive experiments, two images, at compression ratios 8:1, 16:1, were generated using equal allocation of bits to each block. These provide reference (non-adaptive) images for comparison with each of the adaptive results.

Thus we have a total of 18 output images for original input image.

3. Results: Non-Causal Adaptive Encoding

Two different criteria are used to judge quality. These are:

- (1) RMS error.
- (2) Visual quality.

Tables 3.1, 3.2, and 3.3 give the RMS error between the original and the output for the three different images KCALIF, CCALIF, and LADY3 respectively.

Certain representative pictures have been printed to provide the reader with his own reference for (2).

COMMENTS & CONCLUSIONS

(1) First, as is obvious, the images are better, both RMS error and visual qualitywise for 8:1 compression than for 16:1 compression. This can be seen visually by comparing KCALIF RFA and KCALIF RCA and CCALIF RCA and CCALIF RFA. On an average there is a 32.8% increase in RMS error as we go from 8:1 to 16:1 for KCALIF and 42.1% increase for CCALIF and a 75.2% increase for the LADY3 image.

(2) Output processing using the slope facet doesn't help much, as can be seen visually by comparing KCALIF RFA and KCALIO RFA. The change in RMS error is of the order of 10^{-1} .

(3) The dynamic programming version including buffer constraint (Appendix A.3) performs very well. There is no significant visual degradation. This is borne out by comparing visually:

- (a) LADY3 RFA and LADY3 RFB.
- (b) KCALIF RCA and KCALIF RCB.
- (c) CCALIF RCA and CCALIF RCB.

For KCALIF and CCALIF the change in the RMS is of the order of 10^{-1} and slightly higher for LADY3, but this small increase in RMS error is borne out visually. Thus buffer constraint is no impediment to this dynamic programming algorithm.

#	COMPRESSION RATIO? 8:1 or 16:1?	BUFFER CONSTRAINED? y or N?	OUTPUT PROCESSED y or N?	TYPE OF ERROR CURVE USED, RAW(R) OR FITTED BY A POLYNOMIAL(F)	RMS ERROR	NUMBER OF PIXELS THAT DIFFER	OTHER DESCRIPTION OF IMAGE?
1	8:1	N	N	Not Applicable	7.105		Images created by equal allocation of bits
2	16:1	N	N	-do-	8.258		(Non-adaptive)
3	8:1	N	N	R	4.0084	8425	Adaptive
4	8:1	N	Y	R	4.1058	8404	-do-
5	8:1	Y	N	R	4.0512	8410	-do-
6	8:1	Y	Y	R	4.2136	8416	-do-
7	8:1	N	N	F	4.3336	8422	-do-
8	8:1	N	Y	F	4.3960	8448	-do-
9	8:1	Y	N	F	4.3352	8425	-do-
10	8:1	Y	Y	F	4.4014	8410	-do-
11	16:1	N	N	R	5.5128	8798	-do-
12	16:1	N	Y	R	5.4903	8767	-do-
13	16:1	Y	N	R	5.5158	8798	-do-
14	16:1	Y	Y	R	5.4903	8767	-do-
15	16:1	N	N	F	5.9829	8885	-do-
16	16:1	N	Y	F	5.9634	8835	-do-
17	16:1	Y	N	F	5.9936	8892	-do-
18	16:1	Y	Y	F	5.9631	8821	-do-

Table 3.1 Non-Causal KCALIF

#	COMPRESSION RATIO: 8:1 or 16:1?	BUFFER CONSTRAINED? y or N?	OUTPUT PROCESSED y or N?	TYPE OF ERROR CURVE USED. RAW(R) OR FITTED BY A POLYNOMIAL(F)	RMS ERROR	NUMBER OF PIXELS THAT DIFFER	OTHER DESCRIPTION OF IMAGE?
1	3:1	N	N	Not Applicable	6.203		Images created by equal
2	16:1	N	N	-do-	5.349		allocation of bits (Non-adaptive)
3	8:1	N	N	R	2.3342	7850	Adaptive
4	8:1	N	y	R	2.393	7909	-do-
5	8:1	y	N	R	2.3612	7870	-do-
6	8:1	y	y	R	2.4171	7919	-do-
7	8:1	N	N	F	2.5443	7964	-do-
8	8:1	N	y	F	2.5931	8016	-do-
9	8:1	y	N	F	2.5403	7967	-do-
10	8:1	y	y	F	2.5869	8026	-do-
11	16:1	N	N	R	3.482	8565	-do-
12	16:1	N	y	R	3.4067	8542	-do-
13	16:1	y	N	R	3.482	8565	-do-
14	16:1	y	y	R	3.4067	8542	-do-
15	16:1	N	N	F	3.8032	8635	-do-
16	16:1	N	y	F	3.746	8554	-do-
17	16:1	y	N	F	3.8032	8635	-do-
18	16:1	y	y	F	3.7468	8554	-do-

Table 3.2 Non-Causal CCALIF

#	COMPRESSION RATIO? 8:1 or 16:1?	BUFFER CONSTRAINED? y or N?	OUTPUT PROCESSED y or N?	TYPE OF ERROR CURVE USED. RAW(R) OR FITTED BY A POLYNOMIAL(F)	RMS ERROR	NUMBER OF PIXELS THAT DIFFER	OTHER DESCRIPTION OF IMAGE?
1	8:1	N	N	Not Applicable	19.922	9401	Images created by equal allocation of bits
2	16:1	N	N	-do-	30.02	9706	(Non-adaptive)
3	8:1	N	N	R	13.609	9457	Adaptive
4	8:1	N	y	R	13.511	9477	-do-
5	8:1	y	N	R	13.978	9443	-do-
6	8:1	y	y	R	13.83	9422	-do-
7	8:1	N	N	F	15.324	9371	-do-
8	8:1	N	y	F	15.253	9362	-do-
9	8:1	y	N	F	15.4	9683	-do-
10	8:1	y	y	F	15.297	9394	-do-
11	16:1	N	N	R	23.658	9673	-do-
12	16:1	N	y	R	22.519	9599	-do-
13	16:1	y	N	R	24.421	9717	-do-
14	16:1	y	y	R	22.851	9661	-do-
15	16:1	N	N	F	27.711	9517	-do-
16	16:1	N	y	F	26.775	9517	-do-
17	16:1	y	N	F	28.135	9628	-do-
18	16:1	y	y	F	27.531	9597	-do-

Table 3.3 Non-Causal Lady Image

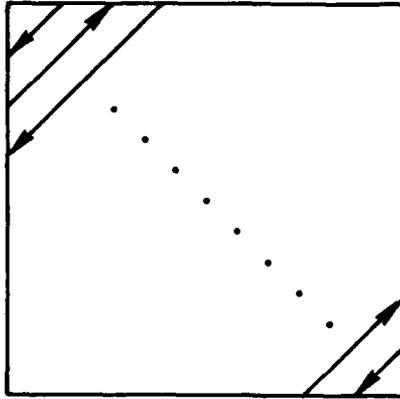
(4) The "fitted" error vs. bits curves caused a slight defocusing of the image. This can be seen best by comparing visually LADY3 RFA and LADY3 FFA. The defocusing is also accompanied by a 16% increase in RMS error. This effect is not as pronounced for KCALIF and CCALIF images in either visual or RMS error measures. The LADY3 image is more complex and, hence, is more sensitive to detail omitted in this approximation. It is important to point out the storage saving involved here, however. When a 6th order polynomial (characterized by 7 coefficients) is used instead of a 100 point error vs. bits curve for each block, we obtain a saving of approximately 190 out of 200 storage locations.

Clearly there is an important trade-off here. The decision as to whether to use the "raw" or "fitted" curves depends on how complex the images being transmitted are.

(5) Finally, all the "adaptive" images are significantly better, both RMS errorwise and visually than the "non-adaptive" equal allocation images. This can be seen by comparing LADE3 RET (8:1) and LADF3 RET (16:1) to any of the adaptive images like LADY3 RFA and LADY3 RFB. In fact, the LADF3 RET image is badly marred and degraded.

(6) One characteristic that is very apparent throughout is the "blockiness" evident in every image. This is due completely to the particular "component selection" scheme used, i.e., columnwise starting from the leftmost column.

This problem can be rectified by using the following scheme. Let (i,j) be the coordinates of a pixel in a block. Calculate $(i^2 + j^2)$ and accordingly arrange the components in ascending order and then select. This amounts to choosing the components diagonally as shown on the next page:



We start at the topmost diagonal and work our way downwards. This scheme weights equally all components at the same frequency. It also assumes a zero-mean image. Therefore, before taking the transform of the image one must subtract the mean and add it back at the end of the process.

Another obvious way to reduce the "blockiness" is to reduce the block size from (10 x 10) to (5 x 5) or (7 x 7).

(7) Instead of quantizing each pixel by 6 bits, 4 bits or 5 bits could be used together with further increases in the compression ratio.

(8) Three graphs P-1, P-2, P-3 have been appended. These are graphs of bits assigned by dynamic programming vs. block number. This clearly illustrates the adaptive nature of the algorithm. Complex blocks are assigned more bits than less complex ones.

4. Transform Coding: Causal Case

4.1 ARMA Model of Complexity of Subimages

We can, by dividing an image into blocks, improve the fidelity with a given number of bits (or bandwidths). We can do this by allocating more bits to where they are needed the most - the highest complexity blocks - and fewer bits to where they are needed less. Further, we can ask whether the complexity of a block itself is predictable from the complexities of neighboring blocks. This could occur, for example, in "busy" areas such as where edges are connected to each other or in smooth areas where background blocks are connected. If the complexity of subimage blocks is predictable, then further compression can be achieved by finding a model for the block complexities, considered as a stochastic series. Then the complexities of future blocks to be transmitted can be estimated from those already transmitted and the allocation of bits to the next block to be transmitted can be minimized based on not only its own complexity but the complexities of the "recent" blocks.

The initial model for the rate distortion was to describe each block's error vs. bits as a series, and to find the statistical properties of the series that allow it to be predicted. Box and Jenkins have done much analysis of time series, and their techniques were used to obtain the model of the series of block complexities.

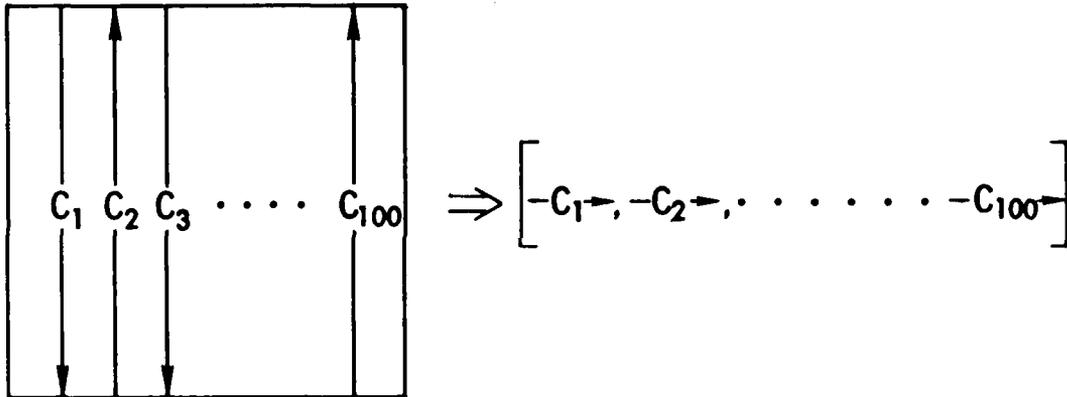
For this analysis, no assumptions were made, for example, as to the stationarity or non-stationarity of the series. We tried fitting models of different types and orders to the series, and used the statistics of the models compared to the original series to measure the goodness of fit of the models.

4.2 Choosing an ARMA Model

Each block is characterised by an error vs. bits curve. From these error curves we form error matrices:

$$\{e_{b_1}\}, \{e_{b_2}\}, \dots \{e_{b_n}\}$$

where $\{e_{b_i}\}$ is the two-dimensional error matrix created by finding the error at a fixed number of bits = b_i . Now our goal is to find an ARMA model for each of these error matrices. For this we used the Box & Jenkins time series package on the Honeywell 66/60. To use this we need to connect the two-dimensional arrays to one-dimensional arrays by concatenating the columns as shown below.



Step 1:

The first step in performing an identification of the type of time series which most closely represents the data is to compute the autocorrelation and partial autocorrelation of the data itself. In the present application the data is the series of error values at a single bit allocation value. The sequence of error values arises from the sequence of blocks.

Once these calculations have been completed, the autocorrelation and partial autocorrelation data are plotted to make patterns easily visible.

If no patterns are present, then the original data may not be a stationary stochastic process and it is necessary to difference the original data to remove the nonstationary part. Eventually after one or more differencing calculation stages almost all practical process data becomes stationary and corresponding autocorrelation and partial autocorrelation data sets, when graphed, exhibit certain patterns.

Table 4 summarizes the general characteristics exhibited by two graphs for several combinations of time series classes.

CLASS OF PROCESSES	AUTOCORRELATIONS	PARTIAL AUTOCORRELATIONS
Moving Average (MA)	Spikes at Lags 1 through Q, then cut off.	Tail Off
Auto Regressive (AR)	Tail Off	Spikes at Lags 1 through P, then cut off.
Mixed Auto Regressive Moving Average (ARMA)	Irregular Pattern at Lags 1 through Q, then tail off	Tail Off

Table 4.1

To illustrate the identification of process class some computer printout (PRINTOUT 1) corresponding to $\{e_{10}\}$ for KCALIF image has been appended. This printout presents IDEN runs for the:

- (1) raw data
- (2) first difference.
- (3) 2nd difference.
- (4) 3rd difference.
- (5) 4th difference.

We note that more structure is observed in the data as higher orders of differencing of the data are taken, but at the same time the raw data

itself is not without identifiable structure. By following the guidelines of Table 4 and some intuition we conclude that a reasonable hypothesis for the $\{e_{10}\}$ data is that it is an AR1 process.

In fact, this class was found to be the best class choice for all $\{e_{b_i}\}$ arrays for all three images. Further, no significant periodicity was observed for any of the $\{e_{b_i}\}$ s for any of the images.

Step 2:

The next step is to estimate the parameters of the AR1 process efficiently. This is accomplished by using maximum likelihood parameter estimation methods. The theory behind this is explained in detail in the book by Box & Jenkins [2] and need not be repeated here since these methods are widely known and utilized.

Step 3:

Having a set of parameter estimates, the next step is to perform tests to verify their quality. The Box & Jenkins software package provides the computational tools required to run several diagnostic checks.

4.3 Verification of an Identified ARMA Model for a Process

4.3.1 Diagnostic Tests.

One way of viewing the process of modeling time series is an attempt to find a transformation that reduces the observed data to random noise. If we have succeeded in this, we would expect to find that the residuals have the properties of random numbers - in particular not serially correlated.

a. A first check is, therefore, sample autocorrelations of the residuals. These autocorrelations should be significantly close to zero (independent of the number of lag steps).

b. Another important property of the residuals is that they be correlated in general with the current value of the observed data and some future values (depending on the order of the process) but not past-values. So as an additional check in the model, corresponding sample correlations between residuals and the observed data are computed.

c. Finally, it is often useful to simply inspect a graph of the residuals for evidence of model inadequacy. They should show no evidence of a particular pattern or trend and should be close to zero.

Other diagnostic tests discussed in Box & Jenkins [2] include:

d. The Dubin Watson statistic should be close to 2.0. A value between 1.8 - 2.2 is generally satisfactory.

e. Number of negative residuals should be approximately equal to the number of positive residuals.

f. The number of runs= R should not be too large or too small compared to the data dimension.

g. The z-statistic for the RMS Test should be between -1.96 and 1.96 for 95% confidence.

h. The power spectrum of white noise is a constant. Consequently, the cumulative spectrum for white noise

$$P(f) = \int_0^f p(g) dg$$

plotted against f is a straight line running from $(0,1)$ to $(0.5,1)$. For a white noise series, the "cumulative normalized periodogram" should be scattered about a straight line joining the points $(0,1)$ and $(0.5,0)$. Moreover, the periodogram should be within the 80% lower and upper Kolmogorov-Smirnoff limits as shown in Figure 4.1.

i. Estimates of the variance of the residuals provide another tool for discriminating among alternative models for a given series.

To illustrate the above computations, computer PRINTOUT 2 gives the IDEN and Estimation (ESTI) runs for $\{e_{30}\}$ of the KCALIF image. All diagnostic checks are made and we conclude that the following AR(1) model

$$e(n) = .3237 e(n-1) + 88.1869$$

for the $\{e_{30}\}$ sequence is valid.

The constant term (88.1869) adjusts for the nonzero mean of the working series.

4.3.2 Model Overfitting.

If our model is AR(1), we may ask if AR(2) might be a more appropriate model. Or should a moving average term be added to the model to make it an ARMA Model? The most obvious test of such hypotheses is by overfitting and testing the hypothesis that the added parameter is equal to zero.

Overfitting tests were carried out for the $\{e_{10}\}$ sequence of the KCALIF image.

An attempt was made to fit an AR(2) process to what was previously identified as an AR(1) process, that is to fit a model of the form

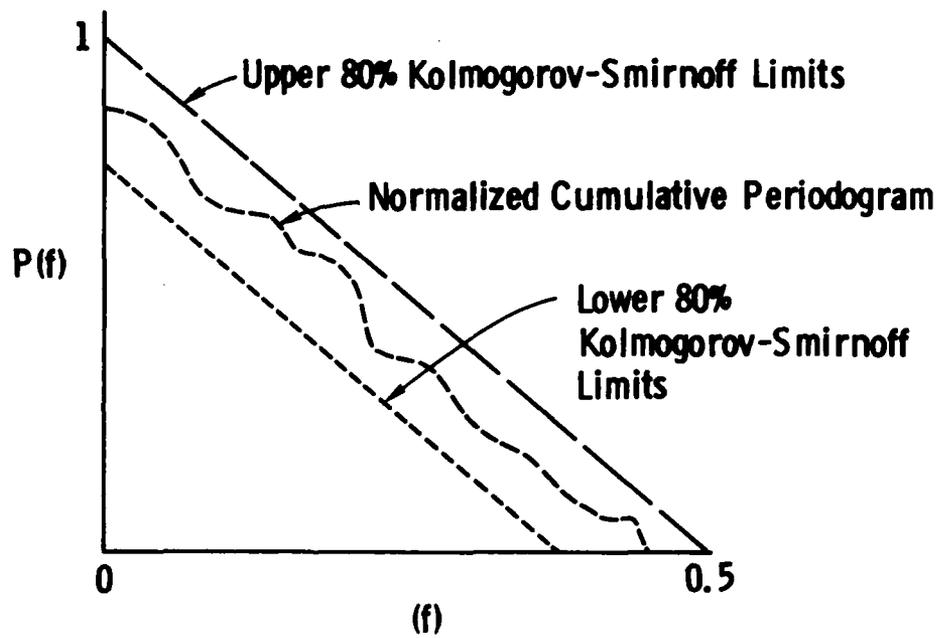


Figure 4.1 Cumulative Normalized Periodogram

$$e(n) = a_1 e(n-1) + a_2 e(n-2)$$

to the process previously hypothesized to be of the form

$$e(n) = a_1 e(n-1).$$

The parameter values obtained were

$$a_1 = .4359 \text{ with a standard error} = .1011$$

$$a_2 = .0108 \text{ with a standard error} = .1014.$$

Note that not only is $a_2 \ll a_1$ but the standard error for a_2 is much larger than the estimate of a_2 itself! This proves that a_2 is insignificant. Therefore, the process is AR(1) as previously hypothesized.

An attempt was also made to fit an AR(1) + MA(1) model to the above data. That is, a model of the form

$$e(n) = a_1 e(n-1) + a_2 u(n-1)$$

(where $u(n-1)$ is the noise term)

was hypothesized and fitted to the data. Resulting parameter estimates were

$$a_1 = .4675 \text{ with standard error} = .2044$$

$$a_2 = .0-31 \text{ with standard error} = .2298$$

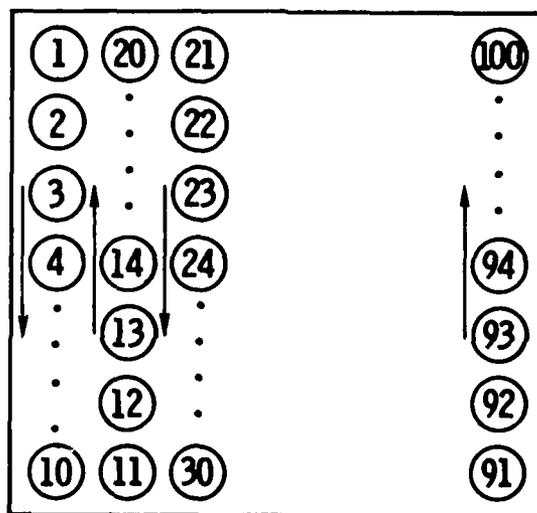
Once again $a_2 \ll a_1$ and a_2 is much greater than the estimate of a_2 itself, showing that a_2 is insignificant. We again conclude that the process is AR(1).

4.4 Causal Bit Allocation

Once the model for the rate distortion of blocks is known, it can be used in a predictor for causal bit allocation to blocks. In causal allocation, only the rate distortions of the past blocks (those already transmitted) are known; those of the future blocks are estimated so that the best allocation of bits to the current block relative to all the other blocks can be made. The better the estimate, the fewer bits will

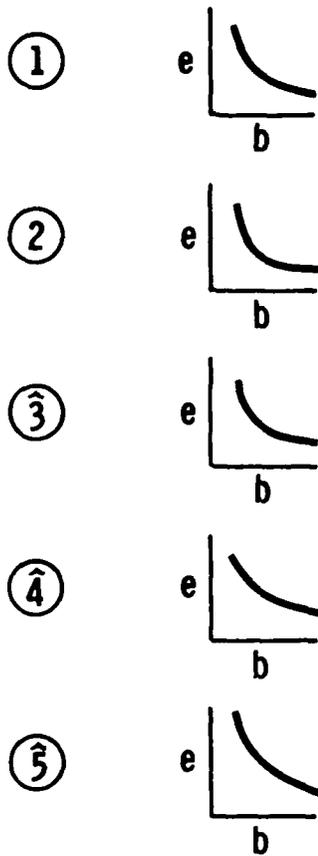
be wasted (that should be used on future blocks). What is wanted is a measure of the relative information content of the current block, compared to all past and future blocks.

Knowing the correlations between blocks as a function of the distance between them, future blocks can be predicted with minimum error. Using the model developed from the ARMA analysis, we scan through the blocks in a way such that the blocks form a series, each correlated with the previous one:



We scan down one column of the image and up the next column. Since the correlation decreases with distance (blocks of a large distance from other blocks have statistically independent rate distortions from them), we include only a small number of blocks in the allocation, estimating the rest by the mean rate distortion of all past blocks.

We take a sample of five blocks at a time, including two past blocks with known rate distortion, and three future blocks with estimated rate distortion.



We assume that this is the farthest we can estimate by correlation models. Having these estimates of the rate distortions of members in a group of blocks, we compare the bits needed for the group to the bits needed for all other blocks, under some constraint of total bits. As mentioned, the other blocks' rate distortion is estimated by the mean rate distortion of all known blocks. We then assign to the group of blocks a total number of bits, to be allocated among them by a technique depending on their specific cost vs. error curves (dynamic programming). Allocation is then made for the next group, an overlapping one with one new known to one new estimated block; e.g. 12345 then 23456.

The actual measure of information content of the group of blocks relative to all other blocks is:

$$\text{TOTAL}(\ell) = \frac{N}{M} \left[1 + w \sum_{b=6}^{600} \left(\frac{e_{\ell}(b) - \bar{e}(b)}{\sigma(b)} \right) \right]$$

TOTAL(ℓ) is the number of bits allocated to a particular group of five blocks (ℓ identifies which group in the series of blocks).

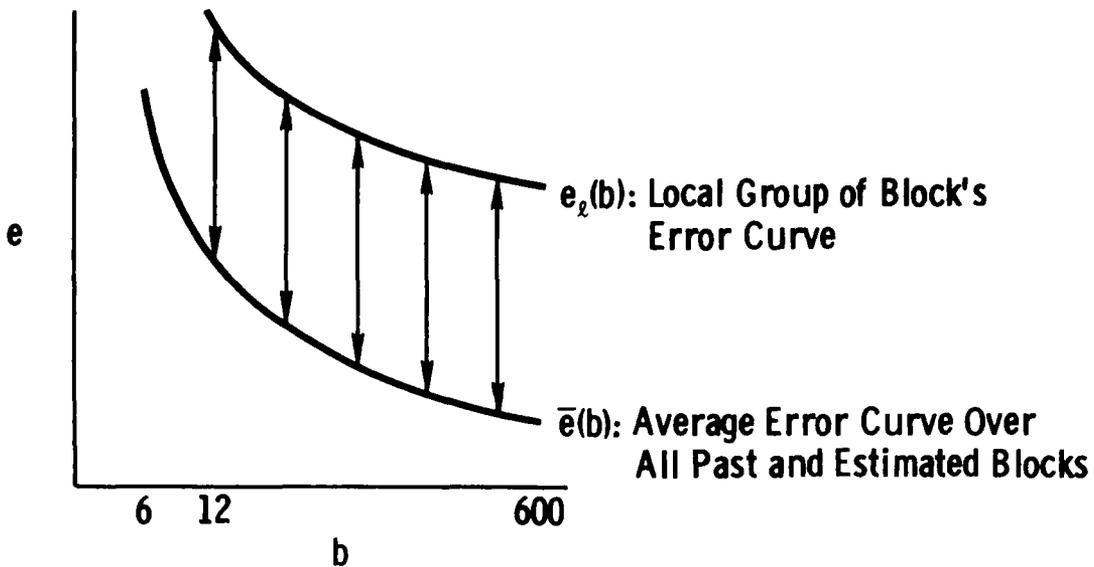
$\frac{N}{M}$ is the total number of bits for the image divided by the number of five-block groups; i.e., the average number of bits per group of five blocks.

$(e_{\ell}(b) - \bar{e}(b))$ is the difference of the error in the current group of blocks at a fixed number of bits (averaged over all blocks in the group) minus the average error at that number of bits overall known and estimated blocks.

$\sigma(b)$ is the standard deviation of error at that number of bits over all known and estimated blocks.

So $\left(\frac{e_{\ell}(b) - \bar{e}(b)}{\sigma(b)} \right)$ is the difference, at a fixed bits value, between

the current blocks' errors and the average error over all previous and estimated blocks, relative to the standard deviation from the average error over all past and estimated blocks. This measure of deviation between the local blocks' rate distortion, and all other blocks' rate distortion, is summed over all bits values. This gives an indication of the relative difference of rate distortion over the whole curve, as shown on the following page.



The "w" in the equation is a weighting factor to allow adjustment of the range of allocations among local groups of blocks. The higher w is, the more TOTAL(ℓ) can differ from the average group allocation $\frac{N}{M}$. If group ℓ has an average error curve exactly matching that of the average error curve for all past and estimated blocks, $(e_\ell(b) - \bar{e}(b))$ is 0, so TOTAL(ℓ) is $\frac{N}{M}$. If the local group of blocks has a higher information content than the average group of blocks (as measured by the difference of error vs. bits curves, as in Figure 1.2, more than the average number of bits is allocated to that group of blocks.

As more of the image is known, the estimate of the average error curve over all other blocks than the local group of blocks becomes more accurate. As this happens, the relative information of a current group to the rest of the image is better measured, so the adaptive allocation wastes fewer bits. But this kind of causal technique allows the best estimate of relative information of blocks at any point in the scanning of blocks, when any

fraction of the image is known and the rest unknown. Our optimizing of allocation over all such estimates is based on the image models developed from ARMA analysis.

4.5 Results of Causal Encoding Using ARMA Models for Error Prediction

The procedure described in the previous section was applied to all the $\{e_{b_i}\}$ arrays of the three different images KCALIF, CCALIF, and LADY3. The resulting error models obtained are:

(b=standard deviation of the residuals)

For KCALIF:

Model for $\{e_{10}\}$

$$e(n) = .4359 e(n-1) + 122.0354 \quad \sigma = 100.5122$$

Model for $\{e_{30}\}$

$$e(n) = .3237 e(n-1) + 88.1869 \quad \sigma = 68.7268$$

Model for $\{e_{70}\}$

$$e(n) = .1406 e(n-1) + 52.52 \quad \sigma = 40.8996$$

We observe that as b increases from 10 to 70 the dependence of $e(n)$ on $e(n-1)$ decreases.

For CCALIF:

Model for $\{e_{10}\}$

$$e(n) = .3229 e(n-1) + 127.369 \quad \sigma = 64.9558$$

Model for $\{e_{20}\}$

$$e(n) = .3397 e(n-1) + 89.7926 \quad \sigma = 52.389$$

Model for $\{e_{30}\}$

$$e(n) = .1934 e(n-1) + 79.623 \quad \sigma = 43.1174$$

Model for $\{e_{40}\}$

$$e(n) = .1914 e(n-1) + 62.5809 \quad \sigma = 39.182$$

Model for $\{e_{70}\}$

$$e(n) = .03 e(n-1) + 43.018 \quad \sigma = 33.1787$$

For LADY3:

Model for $\{e_{10}\}$

$$e(n) = .5433 e(n-1) + 61.2978 \quad \sigma = 70.7308$$

Model for $\{e_{30}\}$

$$e(n) = .4529 e(n-1) + 51.6741 \quad \sigma = 51.4638$$

Model for $\{e_{40}\}$

$$e(n) = .3664 e(n-1) + 48.2031 \quad \sigma = 45.3396$$

Model for $\{e_{70}\}$

$$e(n) = .0893 e(n-1) + 43.1763 \quad \sigma = 34.449$$

These models were successfully used in the CAUSAL transform coding program

4.6 Exponential Model of Complexity of Subimages

Another approach to modeling the error vs. bit curves is to note the general form of these graphs. They appear to have a strong decaying exponential form, therefore a decaying exponential representation

$$e(b) = \alpha e^{-\beta b} \quad \text{where } b = \text{bits}$$

is postulated. We attempt to use the least squares technique to find estimates for α and β for each block. Then we use the series of these estimates for α and β as raw data in the Box & Jenkins procedure to obtain ARMA models for these parameters of the exponentials fitted to the original error vs. bits data.

To obtain a least-squares estimate for α and β , we minimize

$$S = \sum_{i=1}^{100} (P_i - \alpha e^{\beta b_i})^2$$

But this gives rise to undesirable nonlinear equations. Therefore we reformulated first taking lag on both sides giving

$$\ln e = \ln \alpha + \beta b$$

let $\ln e = y$ and $\alpha_1 = \ln \alpha$.

$$y = \alpha_1 + \beta b$$

So now minimize:

$$S^1 = \sum_{i=1}^{100} (y_i - \alpha_1 - \beta b_i)^2$$

Table 4.4 lists the values of α , β , and the rms error for the blocks in quadrant of the LADY3 image which are typical.

Results:

Data Set 1 lists the values of α , β and the runs error for all blocks for the three images KCALIF, CCALIF, and LADY3.

The $\hat{\alpha}$ estimate sequence was easily modeled as an AR(1) process but the $\hat{\beta}$ sequence could not be successfully modelled as a time series. It apparently does not represent an independent parameter in the exponential representation not adequately accounted for by the α coefficient.

Without β , the exponential representation has only one parameter, α , which is not sufficiently flexible to make the exponential a practical representation.

4.7 Power Series Model of Complexity of Subimage

As an alternative parametric representation of the error vs. bits

data a 6th order polynomial of the form

$$e(b) = \alpha_0 + \alpha_1 b + \alpha_2 b^2 + \alpha_3 b^3 + \alpha_4 b^4 + \alpha_5 b^5 + \alpha_6 b^6$$

was considered. Following fitting of a polynomial of the above form to the data, the parameters $\alpha_0, \alpha_1, \dots, \alpha_6$ are each considered as series and modeled as an ARMA process.

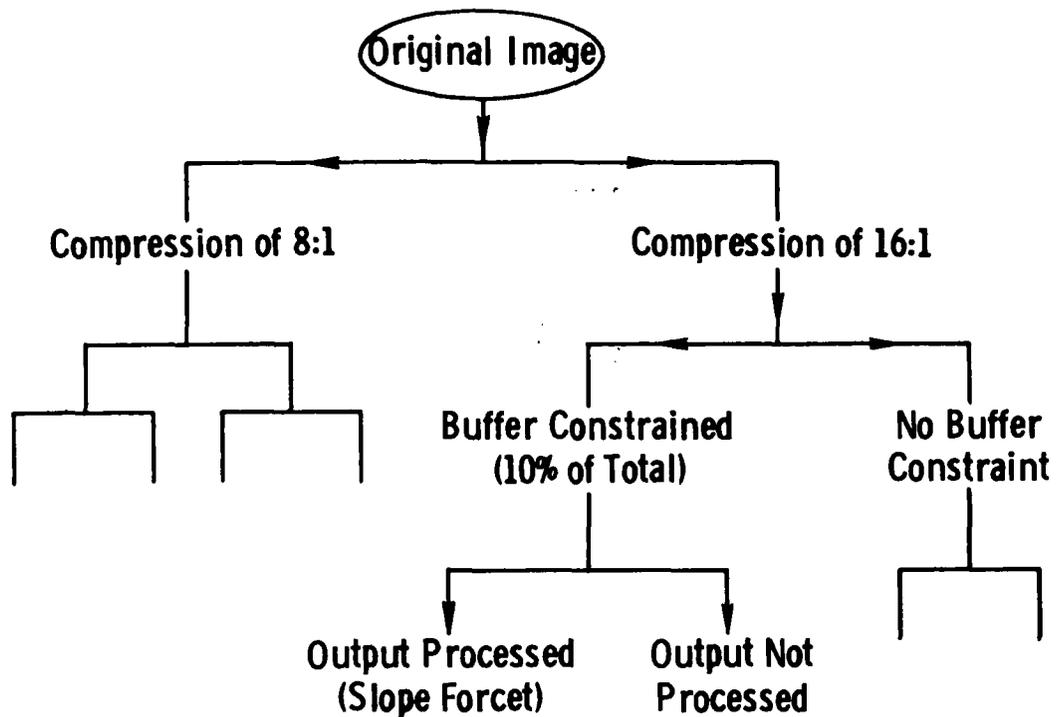
Table 4.5 gives the rms error between the polynomial fit and the raw data for the same set of LADY3 blocks used to generate Table 4.3. These are seen to be significantly less than those for the decaying exponential. This reflects the higher dimensionality of the polynomial model which should made the α_i 's easier to model.

AR(1) models were found to be the best ones. Limitations in project funds and time prevented a more complete analysis of this complexity representation. However, it is a method which appears to be feasible and it is an efficient method from its low storage requirements and should be studied further.

4.8 Results and Conclusions: Causal Adaptive Encoding in the Transform Domain

A series of experiments designed to evaluate the causal allocation system: modeling, prediction and allocation, is diagrammed below. The combinations of conditions are similar to those for the non-causal case. The main difference is that all allocations of bits were accomplished using the causal bit allocation method described in section 4.4 rather than the overall non-causal optimal methods of Part I.

Causal Transform Coding Experiments Performed



There are a total of eight output images per input image. These experiments were performed for two images, CCALIF and LADY3.

Results & Conclusions:

Tables 4.2 and 4.3 give the RMS error between the original image and the various output images.

(1) Once again, as expected, the 8:1 compression does better than 16:1. This can be seen visually by comparing LADYB RFA and LADYB RCA. This is also borne out by the RMS error table.

(2) Both RMS errorwise and visually the causal images are nearly as good as the non-causal ones, e.g., compare LADYB RFA and LADY3 RFA and note their close similarity.

No.	Compression Ratio 8:1 or 16:1	Buffer Constrained Y or N?	Output Processed Y or N?	RMS error	Points that differ out of 10,000	Any other description?
1	8:1	N	Y	2.4522	7973	CAUSAL, ADAPTIVE
2	8:1	N	N	2.3938	7852	-do-
3	8:1	Y	Y	2.7657	8127	-do-
4	8:1	Y	N	2.7510	8149	-do-
5	16:1	N	Y	3.2318	8457	-do-
6	16:1	N	N	3.2741	8412	-do-
7	16:1	Y	Y	3.8432	8863	-do-
8	16:1	Y	N	3.9143	8865	-do-

Table 4.2 RSM Error: Input-Output Images CCALIF

No.	Compression Ratio 8:1 or 16:1	Buffer Constrained Y or N?	Output Processed Y or N?	RMS error	Points that differ out of 10,000	Any other description?
1	8:1	N	Y	15.914	9453	CAUSAL, ADAPTIVE
2	8:1	N	N	15.994	9478	-do-
3	8:1	Y	Y	18,831	9602	-do-
4	8:1	Y	N	18.924	9479	-do-
5	16:1	N	Y	21.776	9594	-do-
6	16:1	N	N	22.003	9594	-do-
7	16:1	Y	Y	25.238	9812	-do-
8	16:1	Y	N	25.48	9812	-do-

Table 4.3 RMS Error: Input-Output Images LADY3

Table 4.4 Parameters for Exponential Fit

Block #	α	β	RMS Error
1	0.111	-0.611	0.821
2	0.122	-0.433	0.790
3	0.077	-0.917	0.905
4	0.082	-0.827	0.897
5	0.101	-0.516	0.846
6	0.114	-0.609	0.826
7	0.096	-0.716	0.859
8	0.422	-3.930	0.401
9	0.548	-2823	0.541
10	0.463	-2.128	0.906
11	0.428	-4.135	0.315
12	0.398	-2.629	0.474
13	0.296	-2.288	0.147
14	0.398	-2.532	0.216
15	0.322	-1.804	0.244
16	0.249	-3.361	0.380
17	0.387	-4.956	0.385
18	0.152	-2.207	0.726
19	0.312	-1.161	0.453
20	0.160	-2.381	0.651
21	0.133	-1.246	0.269
22	0.262	-3.945	0.528
23	0.153	-0.718	0.700
24	-.391	-5.280	0.232
25	0.349	-4.140	0.297

Table 4.5 Parameters for Polynomial Fit

Block #	RMS Error
1	0.255
2	0.207
3	0.255
4	0.256
5	0.218
6	0.254
7	0.260
8	0.277
9	0.210
10	0.290
11	0.155
12	0.201
13	0.086
14	0.150
15	0.115
16	0.121
17	0.214
18	0.209
19	0.119
20	0.237
21	0.073
22	0.272
23	0.282
24	0.179
25	0.219

(3) As regards buffer constraint, the conclusions are the same as that for the non-causal case.

(4) Once again we do better for the CCALIF image than for the LADY image. This can be confirmed by comparing the RMS errors.

(5) Output processing using slope facet caused no significant change visually or errorwise.

(6) We again have "blockiness" due to reasons explained before.

(7) By looking carefully at LADYB RFA we see that we may have done a better job on the right eye than on the left. This is because in the causal case, we gather more and more information as we go from left to right.

(8) In the causal case, if we assign TTOTAL bits for the whole image there is no guarantee the dynamic program will use exactly TTOTAL bits. Usually it ends up with $TTOTAL \pm 10\%$.

PART II

REFERENCES

- [1] Griswold, N.C., and Haralick, R.M., Video Bandwidth Compression, Final Tech Report, February 1977, AFAL-TR-76-102, Air Force Aviaries Lab.
- [2] Box, J.E.P., and Jenkins, G.M., Time Series Analysis Forecasting and Control, Holden Day, 1970.

APPENDIX A

APPENDIX A.1
MAX QUANTIZER

The main criteria used in designing a quantizer is the reduction of the quantization error. This is accomplished by adapting the structure of the quantizer to the signal to be processed. The problem of quantizing for minimum distortion for a signal of known probability density $p(x)$ was first considered in detail by Max [15] in 1960. We present here the basic formulation of this problem and its solution as developed by Max.

The digital transmission rate of any data-transmission system is finite. This means that a quantizer is needed which sorts the input signal into a finite number of ranges, N . For a given N , the system is described by specifying the end points, x_k , of the N input ranges, and an output level, y_k , corresponding to each input range. If the amplitude probability density of the signal which is the quantizer input is given, then the quantizer output is a quantity whose amplitude probability density may easily be determined as a function of the x_k 's and y_k 's. The distortion D , associated with the quantization process, is defined as the expected value of $f(\epsilon)$, where f is some function (differentiable), and ϵ is the quantization error. If we call the input amplitude probability density $p(x)$, then

$$\begin{aligned} D &= E[f(S_{in} - S_{out})] \\ &= \sum_{i=1}^N \int_{x_i}^{x_{i+1}} f(x - y_i) p(x) dx \end{aligned}$$

where $x_{N+1} = \infty$, $x_1 = -\infty$, and the convention is that an input between x_i and x_{i+1} has a corresponding output y_i .

If we wish to minimize D for fixed N , we get necessary conditions by differentiating D with respect to the x_i 's and y_i 's and setting derivatives equal to zero.

$$\frac{\partial D}{\partial x_j} = f(x_j - y_{j-1})p(x_j) - f(x_j - y_j)p(x_j) = 0 \quad (1)$$

$$j = 2, \dots, N$$

$$\frac{\partial D}{\partial y_j} = - \int_{x_j}^{x_{j+1}} f'(x - y_j)p(x)dx = 0 \quad (2)$$

$$j = 1, \dots, N$$

(1) becomes (for $p(x_j) \neq 0$)

$$f(x_j - y_{j-1}) = f(x_j - y_j) \quad j = 2, \dots, N \quad (3)$$

(2) becomes

$$\int_{x_j}^{x_{j+1}} f'(x - y_j)p(x)dx = 0 \quad j = 1, \dots, N \quad (4)$$

If all the second partial derivatives of D with respect to the x_i 's and y_i 's exist then the critical point determined by conditions (3) and (4) is a minimum if the matrix whose i^{th} row and j^{th} column element is

$$\left. \frac{\partial^2 D}{\partial p_i \partial p_j} \right|_{\text{critical point}}$$

where the p 's are the x 's and y 's, is positive definite.

If we let $f(x) = x^2$ then the distortion D is simply the mean square quantization error. In this case,

(3) implies

$$x_j = (y_j + y_{j-1})/2 \quad \text{or} \quad y_j = 2x_j - y_{j-1} \quad (5)$$

$$j = 2, \dots, N$$

(4) implies

$$\int_{x_j}^{x_{j+1}} (x - y_j)p(x)dx = 0 \quad j = 1, \dots, N \quad (6)$$

That is y_j is the centroid of the area of $p(x)$ between x_j and x_{j+1} .

Because of the complicated functional relationships which are likely to be induced by $p(x)$ in (6), this is not a set of simultaneous equations we can hope to solve with any ease. Note, however, that if we choose y_1 correctly we can generate the succeeding x_1 's and y_1 's by (5) and (6), the latter being an implicit equation for x_{j+1} in terms of x_j and y_j .

A method of solving (5) and (6) is to pick y_j , calculate the succeeding x_1 's and y_1 's by (5) and (6) and then if y_N is the centroid of the area of $p(x)$ between x_N and ∞ , y_j was chosen correctly. If y_N is not the appropriate centroid, then y_j must be chosen again. This search may be systematized so that it can be performed on a computer in quite a short time.

This procedure was carried out by Max for a normalized gaussian distribution, under the restriction that $x_{N/2+1} = 0$ for N even, and $y_{(N+1)/2} = 0$ for N odd. This procedure gives symmetric results, i.e., if a signal amplitude x is quantized as y_k , then $-x$ is quantized as $-y_k$. The answers appear in Table I.

TABLE I
PARAMETERS FOR THE OPTIMUM QUANTIZER

		N = 1		N = 2		N = 3	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3		—	0 0	0 0	0 7080	0 0	1 224
	Error	1.000		0 3434		0 1002	
	Entropy	0 0		1 000		1 536	
		N = 4		N = 5		N = 6	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3		0 0	0 4528	0 3823	0 0	0 0	0 3177
	Error	0 0816	1 510	1 244	0 7646	0 6580	1 070
	Entropy	0 0		1 724	1 447	1 804	
		N = 7		N = 8		N = 9	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3 4		0 2803	0 0	0 0	0 2451	0 2218	0 0
	Error	0 8744	0 5006	0 5006	0 7590	0 6812	0 4436
	Entropy	1 611	1 888	1 050	1 344	1 198	0 9188
		N = 10		N = 11		N = 12	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3 4 5 6		0 0	0 1008	0 1837	0 0	0 0	0 1684
	Error	0 4047	0 6090	0 5509	0 3673	0 3401	0 5110
	Entropy	0 8339	1 058	0 9656	0 7524	0 6043	0 8768
		N = 13		N = 14		N = 15	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3 4 5 6 7 8		0 1568	0 0	0 0	0 1457	0 1369	0 0
	Error	0 4760	0 3138	0 2035	0 4413	0 4143	0 3739
	Entropy	0 8126	0 6383	0 5059	0 7503	0 7030	0 5548
		N = 16		N = 17		N = 18	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3 4 5 6 7 8		0 0	0 1284	0 1215	0 0	0 0	0 1148
	Error	0 2582	0 3881	0 3670	0 2430	0 2308	0 3484
	Entropy	0 5224	0 6568	0 6201	0 4909	0 4853	0 5843
		N = 19		N = 20		N = 21	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3 4 5 6 7 8 9 10 11		0 1002	0 0	0 0	0 1036	0 00918	0 0
	Error	0 3284	0 2184	0 2083	0 3128	0 2969	0 1984
	Entropy	0 5551	0 4404	0 4107	0 5285	0 5027	0 3894
		N = 22		N = 23		N = 24	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3 4 5 6 7 8 9 10 11 12		0 0	0 09460	0 09095	0 0	0 0	0 06706
	Error	0 1400	0 2852	0 2736	0 1817	0 1748	0 2821
	Entropy	0 3822	0 4703	0 4574	0 3654	0 3510	0 4309
		N = 25		N = 26		N = 27	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3 4 5 6 7 8 9 10 11 12 13 14		0 0	0 0	0 0	0 08000	0 07779	0 0
	Error	0 2522	0 1676	0 1616	0 2425	0 2340	0 1586
	Entropy	0 4211	0 3368	0 3245	0 4006	0 3921	0 3124
		N = 28		N = 29		N = 30	
		x_i	y_i	x_i	y_i	x_i	y_i
j = 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15		0 0	0 07502	0 07257	0 0	0 0	0 07016
	Error	0 1503	0 2254	0 2182	0 1451	0 1408	0 2110
	Entropy	0 3018	0 3780	0 3655	0 2913	0 2821	0 3532

(cont'd next page)

APPENDIX A.2

Dynamic Programming Solution To The Optimal Non-Causal Bit Allocation Problem

First we describe a dynamic programming algorithm for solving the Optimal Non-Causal Bit Allocation problem with no buffer constraints and then provide a slight modification of it to include buffer constraints.

We assume there are K image blocks. Let $P = \{P_1, \dots, P_n\}$ be the set of N possible bit allocations which may be made to any one block. Let $\epsilon_n: P \rightarrow [0, \infty)$ be the error versus bit rate function for the n th block. Let B be the total number of bits to be allocated to the K blocks. For the optimal non-causal bit allocation, we wish to find any

$$b_1^*, \dots, b_k^* \in P, \sum_{k=1}^K b_k^* \leq B, \text{ satisfying}$$

$$\sum_{k=1}^K \epsilon_k(b_k^*) \leq \sum_{k=1}^K \epsilon_k(b_k) \text{ for every } b_1, \dots, b_k \in P$$

$$\text{and } \sum_{k=1}^K b_k \leq B. \text{ A brute force procedure}$$

would successively go through all N^k possible values b_1, \dots, b_k . Then for those satisfying the constraint $\sum_{k=1}^K b_k \leq B$, it would compute $\sum_{k=1}^K \epsilon_k(b_k)$ and remember the values b_1^*, \dots, b_k^* which gave the minimum. If we consider addition as the basic operation, such an inefficient procedure would take $2 K N^k$ operations.

Fortunately, a more efficient procedure is available. It is a specialized version of Bellman's dynamic programming. To illustrate this technique, we need the following definition. For any T , $1 \leq T \leq B$ and for any M , $1 \leq M \leq K$ define $f_M(T)$ by

$$f_M(T) = \min_{b_1, \dots, b_M \in P} \sum_{m=1}^M \epsilon_m(b_m)$$

$$\sum_{m=1}^M b_m \leq T$$

Then clearly

$$\sum_{k=1}^K \epsilon_k(b_k^*) = f_k(B)$$

Now notice that the f_M functions can be computed recursively since

$$f_M(T) = \min_{b_1, \dots, b_M \in P} \sum_{m=1}^M \epsilon_m(b_m)$$

$$\sum_{m=1}^M b_m \leq T$$

$$= \min_{b_M \in P} \min_{b_1, \dots, b_{M-1} \in P} \left\{ \epsilon_M(b_M) + \sum_{m=1}^{M-1} \epsilon_m(b_m) \right\}$$

$$\sum_{m=1}^{M-1} b_m \leq T - b_M$$

$$= \min_{b_M \in P} \left\{ \epsilon_M(b_M) + \min_{b_1, \dots, b_{M-1} \in P} \sum_{m=1}^{M-1} \epsilon_m(b_m) \right\}$$

$$\sum_{m=1}^{M-1} b_m \leq T - b_M$$

$$= \min_{b_M \in P} \left\{ \epsilon_M(b_M) + f_{M-1}(T - b_M) \right\}$$

Computing $f_K(B)$ by this recursive procedure allows a more efficient calculation since it requires $B \cdot N$ operations to compute the value of any f_M for all of its possible arguments. The values of the functions f_1, \dots, f_{K-1} have

to be computed for all of their arguments and f_K only has to be computed for the argument B. This takes a total $(K-1)B.N + 1$ operations.

These equations also suggest a quick way for determining the optimizing allocations b_1^*, \dots, b_k^* . For each T, $1 \leq T \leq B$ and M, $1 \leq M \leq K$, define $P_M(T)$ to be the smallest element of P satisfying

$$f_M(T) = \epsilon_M(P_M(T)) + f_{M-1}(T - P_M(T))$$

Then

$$b_K^* = P_K(B)$$

and for

$$M < K,$$

$$b_M^* = P_M\left(B - \sum_{m=M+1}^K b_m^*\right)$$

The solution for the Non-Causal buffer constrained bit allocation problem is carried out in identical form, but an additional constraint is put on the functions $f_M(T)$. That is,

$$f_M(T, r_1, r_2) \min_{b_1, \dots, b_M \in P} \sum_{m=1}^M \epsilon_m(b_m)$$

$$\sum_{m=1}^M b_m \leq T$$

$$-r_2 \leq \sum_{m=1}^M b_m - Mc \leq r_1$$

where c is the channel capacity and $r_1 = r_2 = R$ where R is the buffer size

$$\text{Let } r_1^1 = r_1 - (b_M - c)$$

$$\text{and } r_2^1 = r_2 + (b_M - c)$$

Then,

$$\begin{aligned}
 f_M(T, r_1, r_2) &= \min_{b_1, \dots, b_{M-1}} \min_{b_M \in P} \left\{ e_M(b_M) + \sum_{m=1}^{M-1} e_m(b_m) \right\} \\
 &\quad \sum_{m=1}^{M-1} b_m \leq T - b_M \\
 &\quad -r_2^1 - \sum_{m=1}^{M-1} b_m - (M-1)c \leq r_1^1 \\
 &= \min \left\{ e_M(b_M) + \min_{b_1, \dots, b_{M-1}} \min_{b_M \in P} \sum_{m=1}^{M-1} e_m(b_m) \right\} \\
 &\quad \sum_{m=1}^{M-1} b_m \leq T - b_M \\
 &\quad -r_2^1 - \sum_{m=1}^{M-1} b_m - (M-1)c \leq r_1^1 \\
 &= \min_{b_M \in P} e_M(b_M) + f_{M-1}(T - b_M, r_1^1, r_2^1)
 \end{aligned}$$

APPENDIX A.3

ANALYTIC SOLUTION OF THE OPTIMAL BIT ALLOCATIONS PROBLEM
FOR THE OPTIMAL NON-CAUSAL DPCM QUANTIZATION PROBLEM

Let x_1, \dots, x_N be random variables with variances $\sigma_1^2, \dots, \sigma_N^2$, respectively. Let q_n , $n = 1, \dots, N$ be quantizing functions. $q_n(x_n)$ is the mean of the interval to which x_n is quantized. For each number r of quantizing values for the n^{th} variable, we define the quantizing error to be

$$d_n(r_n) = \min_{q_n} E[(x_n - q_n(x_n))^2]$$

From rate distortion theory we can expect that

$$d_n(r_n) = \sigma_n^2 e^{-\alpha \log_2 r_n}$$

Setting $b_n = \log_2 r_n$, we can write the distortion as

$$d_n(b_n) = \sigma_n^2 e^{-\alpha b_n}$$

The bit assignment problem is to determine b_1, \dots, b_N satisfying $b_n \geq 0$ and

$$\sum_{n=1}^N b_n = B \text{ such that } \sum_{n=1}^N d_n(b_n) \text{ is minimized.}$$

One analytic way to solve this problem is to use the technique of Lagrange multipliers and assume that b_n can take any real value. This does not give us the precise answer we want, but it does give one close enough. Another technique is to use dynamic programming.

$$\begin{aligned} \text{Set } f(b_1, \dots, b_N) &= \sum_{n=1}^N d_n(b_n) + \lambda \left(\sum_{n=1}^N b_n - B \right) \\ &= \sum_{n=1}^N \sigma_n^2 e^{-\alpha b_n} + \lambda \left(\sum_{n=1}^N b_n - B \right) \end{aligned}$$

$$\frac{df}{db_k} = \sigma_k^2 (-\alpha) e^{-\alpha b_k} + \lambda$$

$$\text{Setting } \frac{df}{db_k} = 0, \text{ we have } \lambda = \sigma_k^2 \alpha e^{-\alpha b_k}, \quad k = 1, \dots, N.$$

Hence,

$$\frac{\lambda}{\alpha \sigma_k^2} = e^{-\alpha b_k}$$

$$\frac{\alpha \sigma_k^2}{\lambda} = e^{\alpha b_k}$$

$$\ln \frac{\alpha \sigma_k^2}{\lambda} = \alpha b_k$$

$$b_k = \frac{1}{\alpha} \ln \frac{\alpha \sigma_k^2}{\lambda}$$

Since $\sum_{n=1}^N b_n = B$ we may solve for λ .

$$B = \sum_{n=1}^N b_n = \sum_{n=1}^N \frac{1}{\alpha} \ln \frac{\alpha \sigma_n^2}{\lambda} = \frac{1}{\alpha} \sum_{n=1}^N \ln \sigma_n^2 + \frac{1}{\alpha} \sum_{n=1}^N \ln \frac{\alpha}{\lambda}$$

Multiplying by α and bringing the sums of the log of variances to the left, we have

$$\alpha B - \sum_{n=1}^N \ln \sigma_n^2 = \sum_{n=1}^N \ln \frac{\alpha}{\lambda} = N \ln \frac{\alpha}{\lambda}$$

$$\frac{1}{N}(\alpha B - \sum_{n=1}^N \ln \sigma_n^2) = \ln \alpha - \ln \lambda$$

Solving for λ , we obtain

$$\begin{aligned} \ln \lambda &= \ln \alpha - \frac{1}{N}(\alpha B - \sum_{n=1}^N \ln \sigma_n^2) \\ &= \ln \alpha + \frac{1}{N}(\sum_{n=1}^N \ln \sigma_n^2 - \alpha B) \end{aligned}$$

$$\lambda = e^{\sum_{n=1}^N \ln \alpha} + \frac{1}{N} \left(\sum_{n=1}^N \ln \sigma_n^2 - \alpha B \right)$$

$$= e^{\sum_{n=1}^N \ln \alpha} e^{\frac{1}{N} \sum_{n=1}^N \ln \sigma_n^2} e^{-\frac{\alpha}{N} B}$$

$$= \alpha e^{\frac{1}{N} \sum_{n=1}^N \ln \sigma_n^2} e^{-\frac{\alpha}{N} B}$$

$$= \alpha e^{\left(\frac{1}{N} \sum_{n=1}^N \ln \sigma_n^2 \right)} e^{-\alpha B/N}$$

$$= \alpha \left(\prod_{n=1}^N \sigma_n^2 \right)^{1/N} e^{-\alpha B/N}$$

Now substitute this value of λ in the expression for b_n .

$$b_n = \frac{1}{\alpha} \ln \frac{\alpha \sigma_n^2}{\left(\prod_{k=1}^N \sigma_k^2 \right)^{1/N}} e^{-\alpha B/N}$$

$$= \frac{1}{\alpha} \left(\ln \frac{\sigma_n^2}{\left(\prod_{k=1}^N \sigma_k^2 \right)^{1/N}} - \ln e^{-\alpha B/N} \right)$$

$$= \frac{1}{\alpha} \left(\ln \frac{\sigma_n^2}{\left(\prod_{k=1}^N \sigma_k^2 \right)^{1/N}} \right) - (-\alpha B/N)$$

$$= \frac{1}{\alpha} \left(\ln \frac{\sigma_n^2}{\left(\prod_{k=1}^N \sigma_k^2 \right)^{1/N}} + \frac{\alpha B}{N} \right)$$

$$= \frac{B}{N} + \frac{1}{\alpha} \ln \frac{\sigma_n^2}{\left(\prod_{k=1}^N \sigma_k^2 \right)^{1/N}}$$

$$= \frac{B}{N} + \frac{1}{\alpha} \left(\ln \sigma_n^2 - \ln \left(\prod_{k=1}^N \sigma_k^2 \right)^{1/N} \right)$$

$$b_n = \frac{B}{N} + \frac{1}{\alpha} \left(\ln \sigma_n^2 - \frac{1}{N} \sum_{k=1}^N \ln \sigma_k^2 \right)$$

APPENDIX A.4
COMPUTER PROGRAM DOCUMENTATION

I. GENERATION OF ERROR VS BIT RATE TABLES

THIS PACKAGE GENERATES THE DPCM ERROR VS BIT RATE TABLES TO BE USED SUBSEQUENTLY TO DO AN OPTIMAL BIT ALLCCATION AND STORES THEM IN A SEQUENTIAL FILE ON DISK.

DCERDV

DCMPIO

DCERNC

HVARDL

DPCHEE

QTZSAB

DPCH2L

DPCMIX

RMSERR

II. OPTIMAL BIT ALLOCATION AND DPCM COMPRESSION

THIS PACKAGE USES THE ERROR VS BIT RATE TABLES CREATED BY THE FIRST PACKAGE TO PERFORM AN OPTIMAL BIT ALLOCATION ON THE BLOCKS OF AN IMAGE AND THEN DPCM'S THOSE BLOCKS ACCORDINGLY.

DPCNDV

DCHPIO

DPCMNC

OBITAL

RESALL

FRSTCL

NEXTCL

HVARDL

DPCHER

QTZSAN

DPCM2L

DPCMIX

RMSERR

C--DCERDV DPCM ERROR VS BIT TABLES GENERATION DRIVER

C IDENTIFICATION

C PROGRAM TITLE DCERDV
C AUTHOR OSCAR A. ZUNIGA
C DATE NOVEMBER 9, 1978
C LANGUAGE FORTRAN IV (F44-RSX)
C SYSTEM PDP - 15
C SITE BSL - CRINC

C PURPOSE

C THIS IS THE DRIVER FOR THE DPCM ERROR VS BIT
C TABLES GENERATION PROGRAM.

C ENTRY POINT

C DCERDV (ALTRET)

C ARGUMENT LIST

C -NOTE- ALL ARGUMENTS PASSED THROUGH KANDIDATS
C LABELED COMMON AREAS

C ALTRET INT ALTERNATE RETURN TAKEN IN CASE OF ERROR

C SUBROUTINES CALLED

C KDPUSH PUSH ERROR PROCESSING STACK (KANDATS)
C DEVCHK CHECKS FOR PROPER DEVICES (KANDATS)
C RDKINL INITIALIZE 'SIF' FILE (KANDATS)
C WHBAND GET THE BAND (I/O) (KANDATS)
C COMTIN GET COMMENT DESCRIPTOR
C RECORDS (I/O) (KANDATS)
C CTRLT SET ADDRESS OF CONTROL T (SYSTEM)
C DCMPIO DPCM DATA COMPRESSION I/O (USER)
C DCERNC DPCM DATA COMP. NUMBER CRUNCHER (USER)
C KDPOP POP ERRCR PROCESSING STACK (USER)

C*****

C SUBROUTINE DCERDV (ALTRET)

C IMPLICIT INTEGER (A-Z)

C DOUBLE INTEGER INFIL (10), LPFIL (10), OTFIL (10), DUM
C DOUBLE INTEGER FILNM (2)
C REAL FRAC
C LOGICAL LFLAG, BRIEF, LONG, SHORT, RUN
C INTEGER IDENT (20), KIDENT (20)

C COMMON /IBCSIZ/ BRIEF, INTT, OTTT, LONG, SHORT, RUN,

```

1          TTYIN, TTYOT, LP, BUNDAT, EXPDAT, SCDEV1,
2          SCDEV2, SCDEV3, CDRDAT
COMMON /ERROR/ IEV
COMMON /COMAND/ IFC, OTDEVN, OTDEV, OTTYP, OTFIL, INDEVN,
1          INDEV, INTYP, INFIL, LPFIL, DUM(10),
2          LFLAG(26)
COMMON /DCEWKA/ WRKSZ, WORK (400)

C
EQUIVALENCE (NCOLS, IDENT(13)), (NROWS, IDENT(14))
EQUIVALENCE (NTBND, IDENT(17)), (NSBND, IDENT(18))
EQUIVALENCE (MODE, IDENT(19))

C
EQUIVALENCE (NTBND2, KIDENT(17)), (NSBND2, KIDENT(18))
EQUIVALENCE (MODE2, KIDENT(19))

C
C
DATA DEVMSK /#040000/
DATA NG /-1/

C
C
CALL KDPUSH ('DCERD', 'V')

C
WRKSZ = 400

C
C
CHECK FOR PROPER DEVICE

C
CALL DEVCHK (DEVMSK, 1, BIT, 69999)

C
C
CHECK INPUT AND OUTPUT .DAT SLOTS

C
IF ( INDEV.LT.1.OR.INDEVN.LT.1.OR.SCDEV1.LT.1 )
1 GO TO 9040

C
IF ( INDEV.EQ.INDEVN.OR.INDEV.EQ.SCDEV1.OR.
1 INDEVN.EQ.SCDEV1 ) GO TO 9050

C
C
SET UP INPUT FILE

C
CALL BDKINL ( INDEV, INFIL, IDENT, 1, IEV, 69999 )

C
CALL CLOSE ( INDEV )

C
C
CHECK THE INPUT FILE

C
IF ( MODE.NE.1.AND.MODE.NE.0 ) GO TO 9000
IF ( NTBND - NSBND .LE. 0) GO TO 9020

C
BLKSZE = NROWS*NCOLS
IF ( 3*BLKSZE.GT.WRKSZ ) GO TO 9010

C
C
GET INFORMATION FROM USER

C
1 CALL DCMPIO ( FILNM, BBFSZE, TOTAL, MXNBTS, SELECT,
FRAC )

```



```

C          NOT AN INTEGER FILE
C
C          IEV = -2012
C          GO TO 9998
C
C 9010    CONTINUE
C
C          NOT ENOUGH WORK SPACE
C
C          IEV = -5010
C          GO TO 9998
C
C 9020    CONTINUE
C
C          NO NUMERIC BANDS
C
C          IEV = -5018
C          GO TO 9998
C
C 9030    CONTINUE
C
C          INPUT FILES NOT THE SAME MODE
C
C          IEV = -5022
C          GO TO 9998
C
C 9040    CONTINUE
C
C          ILLEGAL .DAT SLOTS
C
C          IEV = -2001
C          GO TO 9998
C
C 9050    CONTINUE
C
C          .DAT SLOTS THE SAME
C
C          IEV = -5009
C          GO TO 9998
C
C 9998    CONTINUE
C
C          ERROR IN SUEPROGRAM
C
C          CALL CLOSE ( INDEV )
C          CALL CLOSE ( INDEVN )
C
C          ABNORMAL RETURN
C
C 9999    RETURN ALTRET
C          END

```



```

EQUIVALENCE (NCOLS,IDENT(13)),(NROWS,IDENT(14))
EQUIVALENCE (MODE,IDENT(19))

C
C
EQUIVALENCE (NPPL2,KDENT(6)),(NLIN2,KDENT(7))
EQUIVALENCE (NCCL2,KDENT(13)),(NROW2,KDENT(14))

C
C
CALL RDPUSH('DCERN','C')

C
C
CHECK .DAT SLOTS

C
1 IF (INDAT.EQ.OTDAT2.OR.INDAT2.EQ.OTDAT2.OR.
   OTDAT.EQ.OTDAT2) GO TO 9010

C
IF (OTDAT2.LT.1) GO TO 9050

C
C
OPEN INPUT FILE

C
CALL RDKINL(INDAT,INFIL,IDENT,1,IEV,&9998)

C
CALL CLOSE (INDAT)

C
C
OPEN LOW PASS FILTER FILE

C
CALL RDKINL (INDAT2, LPFIL, KDENT, 1, IEV, &9998)

C
C
CHECK THAT SIZE OF INPUT FILES
IS THE SAME

C
IF ( NPPL.NE.NPPL2.OR.NLIN.NE.NLIN2 ) GO TO 9030
IF ( NCOLS.NE.NCOL2.OR.NROWS.NE.NROW2 ) GO TO 9040

C
C
ACTUAL NUMBER CRUNCHING

C
C
COMPUTE THE NUMBER OF BLOCKS IN
INPUT FILE AND THE BLOCK SIZE

C
NBLOCKS = ICEIL ( NLIN, NROWS ) * ICEIL ( NPPL, NCOLS )
BLKSIZE = NROWS * NCOLS

C
C
COMPUTE POINTERS FOR WORK AREA

C
PT1 = 1
PT2 = PT1 + BLKSIZE
PT4 = PT2 + BLKSIZE

C
C
COMPUTE THE DPCM ERROR TABLE

C
DO 1015 I = 1, MXNBTS
   QTZBTS(I) = I
1015 CONTINUE

```



```
C          INPUT FILES NOT THE SAME BLOCK SIZE
C
          IEV = -5023
          GO TO 9999
C
9050      CONTINUE
C
C          ILLEGAL FILE CODE
C
          IEV=-2001
          GO TO 9999
9998      CONTINUE
C
C          READ OR WRITE ERROR
C
          CALL CLOSE ( INDAT )
          CALL CLOSE ( INDAT2 )
9999      RETURN ERRET
C
C          END
```



```

1          TTYIN, TTYOT, LP, RUNDAT, EXPDAT, SCDEV1,
2          SCDEV2, SCDEV3, CDRDAT
COMMON /ERROR/ IEV
COMMON /COMAND/ IPC, OTDEVN, OTDEV, OTTYP, OTFIL, INDEVN,
1          INDEV, INTYP, INFIL, LPFIL, DUM(10),
2          LFLAG(26)
COMMON /DCEWKA/ WRKSZ, WORK (400)

C
EQUIVALENCE (NCOLS, IDENT(13)), (NROWS, IDENT(14))
EQUIVALENCE (NTBND, IDENT(17)), (NSBND, IDENT(18))
EQUIVALENCE (MODE, IDENT(19))

C
EQUIVALENCE (NTBND2, KDENT(17)), (NSBND2, KDENT(18))
EQUIVALENCE (MODE2, KDENT(19))

C
C
DATA DEVMSK /#040000/
DATA NU /-1/

C
C
CALL KDPUSH ('DPCMD', 'V')

C
WRKSZ = 400

C
C
C          CHECK FOR PROPER DEVICE
CALL DEVCHK (DEVMSK, 1, BIT, 8999)

C
C
C          CHECK INPUT AND OUTPUT .DAT SLOTS
IF ( INDEV.LT.1.OR.INDEVN.LT.1.OR.OTDEV.LT.1 )
1      GO TO 9040

C
IF ( INDEV.EQ.INDEVN.OR.INDEV.EQ.OTDEV.OR.
1      INDEVN.EQ.OTDEV ) GO TO 9050

C
C
C          SET UP INPUT FILE
CALL RDKINL ( INDEV, INFIL, IDENT, 1, IEV, 89999 )
CALL CLOSE ( INDEV )

C
C
C          CHECK THE INPUT FILE
IF ( MODE.NE.1.AND.MODE.NE.0 ) GO TO 9000
IF ( NTBND - NSBND .LE. 0 ) GO TO 9020

C
BLKSIZE = NROWS*NCOLS
IF ( 3*BLKSIZE.GT.WRKSZ ) GO TO 9010

C
C
C          GET INFORMATION FROM USER
CALL DCNPIO ( FILNM, BBFSZE, TOTAL, MXNBTS, SELECT,
1          FRAC )

```



```

C
C
C          NOT AN INTEGER FILE
C
C          IEV = -2012
C          GO TO 9998
C
C 9010    CONTINUE
C
C
C          NOT ENOUGH WORK SPACE
C
C          IEV = -5010
C          GO TO 9998
C
C 9020    CONTINUE
C
C
C          NO NUMERIC BANDS
C
C          IEV = -5018
C          GO TO 9998
C
C 9030    CONTINUE
C
C
C          INPUT FILES NOT THE SAME MODE
C
C          IEV = -5022
C          GO TO 9998
C
C 9040    CONTINUE
C
C
C          ILLEGAL .DAT SLOTS
C
C          IEV = -2001
C          GO TO 9998
C
C 9050    CONTINUE
C
C
C          .DAT SLOTS THE SAME
C
C          IEV = -5009
C          GO TO 9998
C
C 9998    CONTINUE
C
C
C          ERROR IN SUBPROGRAM
C
C          CALL CLOSE ( INDEV )
C          CALL CLOSE ( INDEVN )
C          CALL CLOSE ( OTDEV )
C
C
C          ABNORMAL RETURN
C
C 9999    RETURN ALTRET
C          END

```

*--DCMPIO DPCM DATA COMPRESSION I/O

* IDENTIFICATION

* TITLE DPCM DATA COMPRESSION I/O
* AUTHOR OSCAR A. ZUNIGA
* DATE AUGUST 30, 1978
* LANGUAGE RATFOR (XVM/RATFOR V2A003)
* SYSTEM PDP - 15
* SITE BSL - CHINC
* UNIVERSITY OF KANSAS
* 2291 IRVING HILL DRIVE
* LAWRENCE, KANSAS 66045
* (913)-864-4836

* PURPOSE

* THIS SUBROUTINE OBTAIN FROM THE USER ALL THE INFOR-
* MATION REQUIRED TO PERFORM THE DPCM DATA COMPRESSION
* EXPERIMENTS

* ENTRY POINT

* CALL DCMPIO (FILNM, BBPSZE, TOTAL, MXNBTS, SELECT,
* FRAC)

* ARGUMENT LIST

* FILNM DINT SEQUENTIAL FILE NAME FOR
* DPCM ERROR TABLE
* BBPSZE INT THE BIT BUFFER SIZE
* TOTAL INT TOTAL NUMBER OF BITS CONSTRAINT
* MXNBTS INT THE MAXIMUM NUMBER OF BITS TO BE
* ASSIGNED TO ANY BLOCK (BETWEEN 0
* AND 6)
* SELECT INT ALLOWS THE USER TO SELECT A PARTI-
* CULAR DPCM TECHNIQUE
* (2) 2-D DPCM FLAT MODEL LSE
* (3) MOD. DPCM FLAT MODEL LSE
* (4) 2-D DPCM FLAT MODEL MVE
* (5) MOD. DPCM FLAT MODEL MVE
* (6) 2-D DPCM SLOPED MODEL LSE
* (7) MOD. DPCM SLOPED MODEL LSE
* (8) 2-D DPCM SLOPED MODEL MVE
* (9) MOD. DPCM SLOPED MODEL MVE
* FRAC REAL FRACTION OF STANDARD DEVIATION THAT
* RANGE OF DITHER SHOULD BE (TO BE
* USED ON THE PARTICULAR DPCM TECHNI-
* QUE)

* HARDWARE REQUIRED

* PDP - 15

*
* SUBROUTINES CALLED
*

* NONE
*
*

* SUBROUTINE DCMPIO (FILNM, BBFSZE, TOTAL, MINBTS, SELECT,
* FRAC)

* TYPE STATEMENTS
*

* IMPLICIT INTEGER (A - Z)
* DOUBLE INTEGER FILNM(2)
* REAL FRAC
* DATA INTT, OTTT / 12, 13 /

* READ SEQUENTIAL FILE NAME FOR
* DPCM ERROR TABLE

*
* WRITE (OTTT,6000)
6000 FORMAT (' ---ENTER SEQUENTIAL FILE NAME FOR DPCM ERROR',
* ' TABLE---A5, A4---')

*
* READ (INTT,6010) FILNM
6010 FORMAT (A5, A4)

* READ LOW PASS FILTER IMAGE OPTION

* READ TOTAL NUMBER OF BITS CCNSTRRAINT

*
* WRITE (OTTT,6050)
6050 FORMAT (' ----ENTER TOTAL NUMBER OF BITS CCNSTRRAINT--I3--')
* READ (INTT,6070) TOTAL

* READ BIT BUFFER SIZE

*
* WRITE (OTTT,6060)
6060 FORMAT (' ----ENTER BIT BUFFER SIZE---I3---')
* READ (INTT,6070) BBFSZE
6070 FORMAT (I3)

* READ MAXIMUM NUMBER OF BITS TO BE
* ASSIGNED TO ANY INDIVIDUAL BLCCK

*
* WRITE (OTTT,6080)
6080 FORMAT (' ----ENTER THE MAXIMUM NUMBER OF BITS TO BE',
* ' ASSIGNED TO ANY INDIVIDUAL BLCCK---I1---')
* READ (INTT,6090) MXNBTS

6090 FORMAT (I1)

*
*
*

READ SELECTION FOR DPCM TECHNIQUE

WRITE (OTTT,6100)

6100 FORMAT (' ----ENTER SELECTION FOR DPCM TECHNIQUE---I1---'//,
 ' (2) 2-D DPCM FLAT MODEL L.S.E.'//,
 ' (3) MODIF. DPCM FLAT MODEL L.S.E.'//,
 ' (4) 2-D DPCM FLAT MODEL MIN. VAR.'//,
 ' (5) MODIF. DPCM FLAT MODEL MIN. VAR.'//,
 ' (6) 2-D DPCM SLOPED MODEL L.S.E.'//,
 ' (7) MODIF. DPCM SLOPED MODEL L.S.E.'//,
 ' (8) 2-D DPCM SLOPED MODEL MIN. VAR.'//,
 ' (9) MODIF. DPCM SLOPED MODEL MIN. VAR.')

READ (INTT,6090) SELECT

*
*
*
*
*

READ FRACTION OF STANDARD DEVIATION
THAT RANGE OF DITHER SHOULD BE

WRITE (OTTT,6110)

6110 FORMAT (' ----ENTER FRACTION OF STANDARD DEVIATION THAT'//,
 ' RANGE OF DITHER SHOULD BE---F5.3---')

READ (INTT,6120) PRAC

6120 FORMAT (F5.3)

*
*

RETURN
END

C--DPCMNC NON-CAUSAL DPCM NUMBER CRUNCHER
 (USING DYNAMIC PROGRAMMING)

C IDENTIFICATION

C TITLE DPCMNC
C AUTHOR OSCAR A. ZUNIGA
C VERSION A.01
C DATE 11/04/78 06:33
C LANGUAGE FORTRAN IV (V44-RSX/MULTI-ACCESS)
C SYSTEM PDP-15
C SITE RSL-CRINC
C UNIVERSITY OF KANSAS,
C 2291 IRVING HILL DRIVE,
C LAWRENCE, KANSAS 66045.
C (913)-864-4836

C PURPOSE

C THIS IS THE NUMBER CRUNCHER FOR THE DPCM DATA COM-
C PRESSION PROGRAM.
C THIS ROUTINE DOES AN OPTIMAL BIT ALLOCATION TO THE
C BLOCKS OF AN INPUT IMAGE BY THE TECHNIQUE OF DYNAMIC
C PROGRAMMING USING A DPCM ERROR VS BIT RATE TABLE CRE-
C ATED PREVIOUSLY AND STORED IN A SEQUENTIAL FILE.
C BLOCKS ARE THEN READ FROM THE INPUT IMAGE AND A LOW
C PASS FILTERED VERSION OF IT AND A DPCM TECHNIQUE
C SELECTED BY THE USER IS SUBSEQUENTLY APPLIED TO EACH
C BLOCK USING AS MANY BITS AS INDICATED BY THE OPTIMAL
C BIT ALLOCATION ALGORITHM.

C ENTRY POINT

C DPCMNC (INDAT, INDAT2, INFIL, LPFIL, OTDAT,
C OTFIL, OTDAT2, FILNM, BND, BND2, WORK,
C WRKSIZ, BBFSIZE, TOTAL, MYNBTS, SELECT,
C FRAC, IEV, EBRET)

C ARGUMENT LISTING

C INDAT INT INPUT LOGICAL UNIT NUMBER
C INDAT2 INT INPUT LUN FOR LOW PASS FILTER FILE
C INFIL DINT INPUT FILE NAME
C LPFIL DINT LOW PASS FILTER FILE NAME
C OTDAT INT OUTPUT LOGICAL UNIT NUMBER
C OTFIL DINT OUTPUT FILE NAME
C OTDAT2 INT LUN FOR SEQUENTIAL FILE
C FILNM DINT SEQUENTIAL FILE NAME
C BND INT BAND OF INPUT IMAGE TO BE PROCESSED
C BND2 INT BAND OF LOW PASS FILTER IMAGE
C WORK INT WORK ARRAY TO HOLD INPUT AND OUTPUT LINES
C WRKSIZ INT SIZE OF THE WORK ARRAY
C BBFSIZE INT BIT BUFFER SIZE
C TOTAL INT TOTAL NUMBER OF BITS TO BE ALLOCATED
C TO ALL BLOCKS IN THE INPUT IMAGE

```

C      MXNBTS  INT      MAXIMUM NUMBER OF QUANTIZATION BITS
C      SELECT  INT      SELECTION NUMBER FOR DPCM TECHNIQUE
C      FRAC    REAL     FRACTION OF STANDARD DEVIATION THAT
C                               RANGE OF DITHER SHOULD BE
C      IEV     INT      INTEGER EVENT VARIABLE
C                               = -2001  ILLEGAL FILE CODE
C                               = -5009  LUNS ARE THE SAME
C      ERRET   INT      ERROR RETURN

```

HARDWARE REQUIRED

PDP - 15

PROGRAM ENVIRONMENT

DPCMNC SHOULD BE CALLED BY ITS DRIVER DPCMEV IN KANDIDATS

ROUTINES CALLED

```

C      KDPUSH  PUSH NAME ONTO ERROR STACK          (KANDIDATS)
C      RDKINL  INITIALIZES AND ACCESSES AN 'SIP'   (KANDIDATS)
C              FILE IN A1 FORMAT
C      RPYDSC  COPIES DESCRIPTOR RECORDS FROM INPUT (KANDIDATS)
C              TO OUTPUT FILE
C      NAMREC  WRITE NEW NAME RECORDS              (KANDIDATS)
C      RDSC20  WRITES PARAMETERS RECORDS          (KANDIDATS)
C      COMTWB  WRITES COMMENT DESCRIPTOR RECORDS  (KANDIDATS)
C      NRCHKO  CHECKS 'O' FLAG BEFORE GETTING NEW (KANDIDATS)
C              NUMBER OF RECORDS ( FUNCTION )
C      MVARDL  MEAN AND VARIANCE OF THE DIFF      ( USER )
C              ERENCE BETWEEN 2 LINES
C      DPCNER  DPCM DATA COMPRESSION ERROR       ( USER )
C      OBITAL  OPTINAL BIT ALLOCATION              ( USER )
C      BREAD   READS A BLOCK FROM AN 'SIP' IMAGE  (KANDIDATS)
C      RWRITE  WRITES A BLOCK ON AN 'SIP' FILE   (KANDIDATS)
C      KDPOP   POPS THE NAME OUT OF THE ERROR STACK (KANDIDATS)

```

```

C      SUBROUTINE DPCMNC ( INDAT, INDAT2, INFIL, LPFIL, OTDAT,
1      OTFIL, OTDAT2, FILNM, BND, BND2, WORK,
2      WRKSIZ, BBPSZE, TOTAL, MXNBTS, SELECT,
3      FRAC, IEV, ERRET )

```

IMPLICIT INTEGER (A-Z)

```

C      DOUBLE INTEGER INFIL(10),LPFIL(10),OTFIL(10),NAME(7)
C      DOUBLE INTEGER IST
C      DOUBLE INTEGER FILNM(2)

```


C
C

INPUT FILE AND THE BLOCK SIZE

NBLCKS = ICEIL (NLIN, NROWS) * ICEIL (NPPL, NCOLS)
BLKSIZE = NROWS * NCOLS

C
C
C

COMPUTE POINTERS FOR WORK AREA

PT1 = 1
PT2 = PT1 + BLKSIZE
PT4 = PT2 + BLKSIZE

C
C
C

FIND OPTIMAL BIT ALLOCATION

1 CALL OBITAL (OTDAT2, FILNM, BBFSZE, TOTAL, MNBTS,
NBLCKS, OBITAB)

C
C
C
C

CREATE OUTPUT FILE

DO 1110 I = 1, NBLCKS
1 CALL RREAD (INDAT, WORK(PT1), BND, I, 1, IDENT,
IEV, 89998)
1 CALL RREAD (INDAT2, WORK(PT2), BND2, I, 1, KIDENT,
IEV, 89998)
1 CALL MVARDL (WORK(PT1), WORK(PT2), DMEAN, DVAR,
BLKSIZE)
1 NBTS = OBITAB(I)
1 CALL DPCMER (WORK(PT1), WORK(PT4), WORK(PT2), DMEAN,
2 DVAR, BLKSIZE, NCOLS, NBTS, FRAC,
IST, SELECT, ERROR)
1 CALL RWRITE (OTDAT, WORK(PT4), 1, I, 1, JDENT,
IEV, 89998)

1110 CONTINUE

C

8000 CONTINUE

C

CALL CLOSE(INDAT)
CALL CLOSE(INDAT2)
CALL CLOSE(OTDAT)
CALL KDPOP
RETURN

C
C
C
C
C
C

ABNORMAL CONDITIONS

9010 CONTINUE

C

C

C

LUNS ARE THE SAME

IEV = -5009
GO TO 9999

C

```

9030 CONTINUE
C
C
C
      INPUT FILES NOT THE SAME SIZE
      IEV = -5021
      GO TO 9999
C
9040 CONTINUE
C
C
C
      INPUT FILES NOT THE SAME BLOCCK SIZE
      IEV = -5023
      GO TO 9999
C
9050 CONTINUE
C
C
C
      ILLEGAL FILE CODE
      IEV=-2001
      GO TO 9999
9998 CONTINUE
C
C
C
C
      READ OR WRITE ERROR
      CALL CLOSE ( INDAT )
      CALL CLOSE ( INDAT2 )
      CALL CLOSE ( OTDAT )
9999 RETURN ERRET
C
C
      END

```

*--MVARDL MEAN AND VARIANCE OF A LINE OF DIFFERENCES

* IDENTIFICATION

* TITLE MVARDL
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE AUGUST 31, 1978
* LANGUAGE RATFOR
* SYSTEM PDP-15

* PURPOSE

* THIS ROUTINE COMPUTES THE MEAN AND VARIANCE OF THE
* DIFFERENCE OF TWO LINES OF INTEGER DATA.

* ENTRY POINT

* MVARDL (LINE1, LINE2, DMEAN, DVAR, NUMPPL)

* ARGUMENT LISTING

* LINE1 INT FIRST LINE OF DATA
* LINE2 INT SECOND LINE OF DATA
* DMEAN REAL MEAN OF THE DIFFERENCE LINE
* DVAR REAL VARIANCE OF THE DIFFERENCE LINE
* NUMPPL INT NUMBER OF POINTS PER LINE

* ROUTINES CALLED .

* NONE

* SUBROUTINE MVARDL (LINE1, LINE2, DMEAN, DVAR, NUMPPL)

* INTEGER LINE1(NUMPPL), LINE2(NUMPPL)
* REAL DMEAN, DVAR, SUM, SUM2

* SUM = 0.
* SUM2 = 0.

* DO I = 1, NUMPPL
* SUM = SUM + LINE1(I) - LINE2(I)
* DMEAN = SUM/NUMPPL

* DO I = 1, NUMPPL
* SUM2 = SUM2 + (LINE1(I)-LINE2(I)-DMEAN)**2
* DVAR = SUM2 / NUMPPL

* RETURN
* END

PLARGE = 999999.

C

```
CALL FSTAT ( INDAT, FILM, I )
IF ( I.NE.0 ) GO TO 1000
WRITE (OTT,6000) FILM
6000  FORMAT ( 1X, A5, A4, ' FILE NOT FOUND' )
      RETURN
1000  CONTINUE
```

C

```
CALL SEEK ( INDAT, FILM )
READ (INDAT) NUMPOS, NSTAGE
BC = FLOAT(TTOTAL)/NSTAGE
CALL RESALL ( INDAT, OTDAT, F, P, PNEXT, PNEXT, TTOTAL,
1         RPOSS, RVALUE, NUMPOS, ALLCT, NSTAGE,
2         BBPSZE, BC )
WRITE (OTT2,6010) ( ALLCT(I), I = 1, NSTAGE )
6010  FORMAT ( 1X, 10I5 )
      CALL CLOSE (INDAT)

      RETURN
      END
```



```

C          NSTAGE INT      NUMBER OF STAGES
C          BBFSZE INT      BIT BUFFER SIZE
C          BC      INT      CHANNEL CAPACITY
C
C          HARDWARE REQUIRED
C
C          NOTHING SPECIAL
C
C          ROUTINES CALLED
C
C*****
C
C          SUBROUTINE RESALL(IN,OT,P,P,FNEXT,PNEXT,TOTAL,RPOSS,RVALUE,
1          NUMPOS,ALLCT,NSTAGE,BEFSZE,EC)
C
C          INTEGER TOTAL,P(TOTAL),PNEXT(TOTAL),RPOSS(NUMPOS)
C          INTEGER ALLCT(NSTAGE)
C          INTEGER TTOTAL,OT,THAX,TMIN,BBFSZE
C          REAL    FILNM(2),F(TOTAL),FNEXT(TOTAL),RVALUE(NUMPOS),BC
C          DATA FILNM/'TEMPR','FILE'/
C
C
C          DEFINE A TEMPORARY RANDCM FILE
C
C          CALL DEFINE(OT,TOTAL,NSTAGE,FILNM,IV,0,0,0,IEV)
C
C          READ IN THE ALLOCATION AND RESOURCE VALUE
C          TABLES FOR THIS STAGE
C
C          READ(IN) (RPOSS(L),L=1,NUMPOS)
C          READ(IN) (RVALUE(L),L=1,NUMPOS)
C
C          INITIALIZE THE FIRST COLUMN
C
C          CALL PRSTCL(F,P,TOTAL,RPOSS,RVALUE,NUMPOS)
C
C          WRITE THE FIRST COLUMN OUT
C
C          WRITE(OT #1) (P(L),L=1,TOTAL)
C
C          DO 2 N=2,NSTAGE
C
C          READ IN THE ALLOCATION AND RESOURCE VALUE
C          TABLES FOR THE NEXT STAGE
C
C          READ(IN) (RPOSS(L),L=1,NUMPCS)
C          READ(IN) (RVALUE(L),L=1,NUMPOS)
C
C          THAX = (BBFSZE+1)/2 + ( N - 1 ) * BC
C          TMIN = - (BBFSZE+1)/2 + ( N - 1 ) * BC + .9
C          THAX = MINO ( THAX, TCTAL )
C          TMIN = MAXO ( TMIN, 0 )
C

```



```

P(I)=0
1 CONTINUE
C
DO 2 I=1,NUMPOS
C
  FMIN=RVALUE(1)
  PHIN=RPOSS(1)
C
    DO 3 J=1,NUMPOS
      IF(RPOSS(J).GT.RPOSS(I)) GO TO 3
      IF(PHIN.LT.RVALUE(J)) GO TO 3
      FMIN=RVALUE(J)
      PHIN=RPOSS(J)
    3 CONTINUE
C
  II=RPOSS(I)
  F(II)=FMIN
  P(II)=PHIN
2 CONTINUE
C
RETURN
END

```

C--NEXTCL BUILD NEXT COLUMN

C IDENTIFICATION
C TITLE NEXTCL
C AUTHOR ROBERT M HARALICK
C VERSION 1
C DATE 10/13/78 13:49
C LANGUAGE FORTRAN IV (V44-RSX/MULTI-ACCESS)
C SYSTEM PDP-15
C SITE RSL-CRINC
C UNIVERSITY OF KANSAS,
C 2291 IRVING HILL DRIVE,
C LAWRENCE, KANSAS 66045.
C (913)-864-4836

C UPDATE # 1

C AUTHOR OSCAR A. ZUNIGA
C DATE DECEMBER 3, 1978
C VERSION B.01
C LANGUAGE FORTRAN IV
C PURPOSE MODIFIED TO ALLOW BUFFER HANDLING.
C TWO PARAMETERS HAVE BEEN ADDED TO
C THE ARGUMENT LIST: THE UPPER CONS-
C TRAIN (THAX) TO PREVENT BUFFER
C OVERFLOW AND THE LOWER CONSTRAINT
C (TMIN) TO PREVENT BUFFER UNDER-
C FLOW

C PURPOSE

C THIS SUBROUTINE TAKES IN THE PREVIOUS RESOURCE VALUE COLUMN
C AND THE ALLOCATION AND RESOURCE TABLES FOR THE CURRENT STAGE
C AND IT CREATES THE ALLOCATION AND RESOURCE VALUE COLUMNS
C FOR THE CURRENT STAGE.

C ENTRY POINT

C NEXTCL (F, TOTAL, RPOSS, RVALUE, NUMPOS, FNEXT, PNEXT,
C THAX, TMIN)

C ARGUMENT LISTING

C F REAL PREVIOUS RESOURCE VALUE COLUMN
C TOTAL INT NUMBER OF POSSIBLE ALLOCATIONS
C RPOSS INT TABLE OF POSSIBLE ALLOCATIONS FOR THE
C CURRENT STAGE
C RVALUE REAL CORRESPONDING TABLE OF VALUES RESULTING
C FROM RESOURCE ALLOCATIONS IN RPOSS
C NUMPOS INT NUMBER OF POSSIBLE ALLOCATIONS FOR THE
C CURRENT STAGE
C FNEXT REAL RESOURCE VALUE COLUMN CREATED AT THIS STAGE
C PNEXT INT RESOURCE ALLOCATION COLUMN CREATED AT THIS
C STAGE

```

C          TMAX      INT      UPPER CONSTRAINT; THE TOTAL MAXIMUM RESOURCES
C          TO BE ALLOCATED TO ALL THE STAGES PREVIOUS
C          TO THE CURRENT ONE
C          TMIN      INT      LOWER CONSTRAINT
C
C  HARDWARE REQUIRED
C
C          NOTHING SPECIAL
C
C  ALGORITHM
C
C          THIS SUBROUTINE DOES ONE STAGE OF DYNAMIC PROGRAMMING
C
C  ROUTINES CALLED
C
C*****
C
C          SUBROUTINE NEXTCL(F,TOTAL,RPOSS,RVALUE,NUMPOS,FNEXT,PNEXT,
1          TMAX,TMIN)
C
C          INTEGER TEST,PMIN,TOTAL,RPOSS(NUMPOS),PNEXT(NUMPOS)
C          INTEGER TMAX, TMIN
C          REAL F(TOTAL),RVALUE(NUMPOS),FNEXT(TOTAL)
C
C          COMMON /CONST/ FLARGE
C
C          DETERMINE EACH ENTRY IN THE FNEXT AND
C          PNEXT COLUMN OF THE ALLOCATION TABLE
C
C          DO 5 I=1,TOTAL
C          PMIN=FLARGE
C          PMIN=0
C
C          GO THROUGH ALL POSSIBLE ALLOCATIONS
C
C          DO 3 J=1,NUMPOS
C
C          IF AN ALLOCATION IS GREATER THAN THE
C          TOTAL ALLOWED, THERE IS NO HOPE
C
C          IF(RPOSS(J).GE.I) GO TO 3
C
C          WE TRY AN ALLOCATION OF RPOSS(J) TO
C          CURRENT STAGE
C
C          THIS LEAVES AN ALLOCATION OF I-RPOSS(J)
C          TO THE PREVIOUS STAGES
C
C          TEST=I-RPOSS(J)
C
C          IF ( TEST.GT.TMAX ) GO TO 3
C          IF ( TEST.LT.TMIN ) GO TO 3

```

```

C
      VALUEP=F (TEST)
C
C
C
C
C
      IF (VALUEP.EQ.FLARGE) GO TO 3
C
C
C
C
C
      ALLOCATION OF RPOSS (J) TO CURRENT STAGE
      AND TEST TO SUM OF ALL PREVIOUS STAGES
      IS OK. THE RESULTING VALUE OF THIS
      ALLOCATION IS ...
C
      GMIN=RVALUE (J) +VALUEP
C
C
C
      COMPARE THIS VALUE TO THE BEST
      PREVIOUS TRY
C
      IF (PMIN.LT.GMIN) GO TO 3
C
C
C
      IT IS BETTER
C
      PMIN=GMIN
      PMIN=RPOSS (J)
3      CONTINUE
C
      PNEXT (I) =PMIN
      PNEXT (I) =PMIN
5      CONTINUE
C
      RETURN
      END

```


*--DPCMER DPCM DATA COMPRESSION ERROR

* IDENTIFICATION

* TITLE DPCMER
* AUTHOR OSCAR A. ZUNIGA
* DATE AUGUST 15, 1978
* LANGUAGE RATFOR (XVM/RATFOR V2A003)
* SYSTEM PDP - 15
* SITE RSL - CRINC
* UNIVERSITY OF KANSAS
* 2291 IRVING HILL DRIVE
* LAWRENCE, KANSAS 66045
* (913)-864-4836

* PURPOSE

* THIS SUBROUTINE QUANTIZES A BLOCK OF DATA USING A
* MAX QUANTIZER AND GENERATES THE CORRESPONDING QUAN-
* TIZATION TABLE. A DPCM COMPRESSION IS THEN APPLIED
* TO THE INPUT BLOCK GENERATING AS A RESULT A BLOCK
* OF OUTPUT. FINALLY THE RMS ERROR BETWEEN INPUT AND
* OUTPUT BLOCKS IS COMPUTED.

* ENTRY POINT

* CALL DPCMER (LNIN, LNOUT, LNLPP, LNMEAN, LNVAR,
* NUMPPL, NCOLS, NBITS, FRAC, IST,
* SELECT, ERROR)

* ARGUMENT LIST

* LNIN INT 1-D ARRAY OF SIZE 'NUMPPL' TO
* STORE THE INPUT BLOCK OF DATA
* LNOUT INT 1-D ARRAY OF SIZE 'NUMPPL' TO
* STORE THE OUTPUT BLOCK OF DATA
* LNLPP INT 1-D ARRAY OF SIZE 'NUMPPL' TO
* STORE THE CONVOLUTED (LOW PASS FILTERED)
* BLOCK OF DATA USED AS A PREDICTOR FOR
* THE INPUT BLOCK.
* LNMEAN REAL THE MEAN OF THE DPCM CORRECTIONS
* LNVAR REAL THE VARIANCE OF THE DPCM CORRECTIONS
* NUMPPL INT THE SIZE OF THE BLOCKS
* NCOLS INT THE NUMBER OF COLUMNS IN A BLOCK
* OR SUBIMAGE
* NBITS INT THE NUMBER OF BITS USED FOR QUAN-
* TIZATION
* FRAC REAL THE FRACTION OF STANDARD DEVIATION
* THAT RANGE OF DITHER SHOULD BE (FOR
* 2-D DPCM)
* IST DINT THE SEED FOR RANDOM NUMBER GENERATION
* SELECT INT USED TO SELECT DPCM TECHNIQUE
* (2 - 9)
* ERROR INT THE RMS ERROR BETWEEN INPUT AND

OUTPUT BLOCKS

HARDWARE REQUIRED

PDP - 15

SUBROUTINES CALLED

QTZSAM	QUANTIZ. END POINTS & MEANS	USER LIBRARY
DPCH2L	2-DIMENSIONAL DPCM	USER LIBRARY
DPCHXY	MODIFIED DPCM (2-D DPCM USING FUTURE VALUES FROM CONVOLUTED IMAGE)	USER LIBRARY
RMSERR	RMS ERROR	USER LIBRARY

SUBROUTINE DPCMER (LNIN, LNOUT, LNLPF, LNMEAN, LNVAR,
NUMPPL, NCOLS, NBITS, FRAC, IST,
SELECT, ERROR)

TYPE STATEMENTS

IMPLICIT INTEGER (A - Z)
DOUBLE INTEGER IST
INTEGER LNIN(NUMPPL), LNOUT(NUMPPL), LNLPF(NUMPPL)
INTEGER QTABLE(512)
INTEGER WGTS2(4), WGTS4(4), WGTS6(4), WGTS8(4)
INTEGER WGTS5(3,3), WGTS7(3,3)
REAL QENDS(65), QMEANS(64), ERROR, LNMEAN, LNVAR
REAL RANGE, FRAC, SQRT
DATA MAX /255/
DATA WGTS2 /4*1/
DATA WGTS4 / 49, 49, 70, 70 /
DATA WGTS5 /100,70,100,70,49,70,100,70,100/
DATA WGTS6 / -1, 0, 1, 2 /
DATA WGTS7 / 9*1 /
DATA WGTS8 / -7, 6, 20, 35 /

INITIALIZATIONS

NLEVELS = 2**NBITS

COMPUTE THE END POINTS AND MEANS USING
MAX QUANTIZER

CALL QTZSAM (LNMEAN, LNVAR, NBITS, QENDS, QMEANS)

QENDS(1) = -MAX - 1
QENDS(NLEVELS+1) = MAX

COMPUTE SIZE OF QUANTIZING TABLE

NTABLE = 2*MAX + 1

COMPUTE THE QUANTIZING TABLE

```
FOR ( J = 1; J <= NLEVELS; J = J + 1 )
$(
  LOWEND = QENDS(J) + MAX + 2
  UPPEND = QENDS(J+1) + MAX + 1
  FOR ( I = LOWEND; I <= UPPEND; I = I + 1 )
  $(
    QTABLE(I) = IROUND(QMEANS(J))
  $)
$)
```

COMPUTE THE LINE OF OUTPUT

```
IF ( SELECT == 3 ) SELECT = 7
IF ( SELECT == 9 ) SELECT = 5
```

```
FOR ( I = 1; I <= NCOLS; I = I + 1 )
$(
  LNOUT(I) = LNIN(I)
$)
```

RANGE = FRAC*SQRT(LNVAR)

```
PRVLN = 1
NYTLN = 2*NCOLS + 1
PSTLN = NCOLS + 1
LSTLN = NUMPPL - NCOLS + 1
```

```
IF ( SELECT==2 | SELECT==4 | SELECT==6 | SELECT==8 )
FOR ( CURLN = PSTLN; CURLN <= LSTLN;
      CURLN = CURLN + NCOLS )
```

```
$(
  IF ( SELECT==2 )
    CALL DPCM2L ( LNIN(CURLN), NUMPPL, LNOUT(PRVLN),
                 LNOUT(CURLN), QTABLE, NTABLE, MAX,
                 RANGE, IST, WGTS2, NGEN, NUSED )
  IF ( SELECT==4 )
    CALL DPCM2L ( LNIN(CURLN), NUMPPL, LNOUT(PRVLN),
                 LNOUT(CURLN), QTABLE, NTABLE, MAX,
                 RANGE, IST, WGTS4, NGEN, NUSED )
  IF ( SELECT==6 )
    CALL DPCM2L ( LNIN(CURLN), NUMPPL, LNOUT(PRVLN),
                 LNOUT(CURLN), QTABLE, NTABLE, MAX,
                 RANGE, IST, WGTS6, NGEN, NUSED )
  IF ( SELECT==8 )
    CALL DPCM2L ( LNIN(CURLN), NUMPPL, LNOUT(PRVLN),
                 LNOUT(CURLN), QTABLE, NTABLE, MAX,
```



```
XSD=SQRT(XDVAR)
NN=2** (NBIT-1)
NNN=2*NN+1
NNX=FLOAT(2*NN)
STEP=6.0/NNX
```

C
C
C
C
C

PICK UP THE APPROPRIATE ENTRIES FROM THE
TABLES AND STICK THEM IN QENDS QND QMEANS
ADJUST FOR MEAN AND ATANDARD DEVIATION.

```
DO 10 I=1,NN
GO TO (1,2,3,4,5,6,7),NBIT
1  NNI=NN+I
   QENDS(NNI)=X1(I)*XSD+XDMEAN
   QMEANS(NNI)=Y1(I)*XSD+XDMEAN
   GO TO 9
2  NNI=NN+I
   QENDS(NNI)=X2(I)*XSD+XDMEAN
   QMEANS(NNI)=Y2(I)*XSD+XDMEAN
   GO TO 9
3  NNI=NN+I
   QENDS(NNI)=X3(I)*XSD+XDMEAN
   QMEANS(NNI)=Y3(I)*XSD+XDMEAN
   GO TO 9
4  NNI=NN+I
   QENDS(NNI)=X4(I)*XSD+XDMEAN
   QMEANS(NNI)=Y4(I)*XSD+XDMEAN
   GO TO 9
5  NNI=NN+I
   QENDS(NNI)=X5(I)*XSD+XDMEAN
   QMEANS(NNI)=Y5(I)*XSD+XDMEAN
   GO TO 9
6  NNI=NN+I
   QENDS(NNI)=(I-1)*STEP*XSD+XDMEAN
   QMEANS(NNI)=(I-0.5)*STEP*XSD+XDMEAN
   GO TO 9
7  NNI=NN+I
   QENDS(NNI)=X7(I)*XSD+XDMEAN
   QMEANS(NNI)=Y7(I)*XSD+XDMEAN
   GO TO 9
8  NNI=NN+I
   QENDS(NNI)=X8(I)*XSD+XDMEAN
   QMEANS(NNI)=Y8(I)*XSD+XDMEAN
9  CONTINUE
10 CONTINUE
```

C
C
C
C
C

SET UP END POINTS AND MEANS ON THE LOWER
SIDE OF THE MEAN.

```
NNN=NN-1
NNID=NN+1
```

```
DO 12 I=1,NNN
```

```

C
      N1=NMID+I
      N2=NMID-I
      QENDS(N2)=QENDS(N1)*(-1.0) + 2.0 * QENDS ( NMID )
C
12    CONTINUE
C
C          FIRST AND LAST END POINTS ARE - AND +
CI         INFINITY RESPECTIVELY)
C
      QENDS(1)=-0.1E+6
      QENDS(NNN)=0.1E+6
C
      DO 13 I=1,NN
C
      I1=NNN-I
      QMEANS(I)=QMEANS(I1)*(-1.0) + 2.0 * QENDS ( NMID )
C
13    CONTINUE
C
      RETURN
      END

```


NGEN=1
NUSED=1

C

RETURN
END


```

C           A B C D E F . . . ( PREVIOUS LINE )
C           P Q R S           ( CURRENT LINE, COL S )

```

```

C ROUTINES CALLED

```

```

C     MAXO    MAXIMUM OF A SET OF INTEGERS           ( SYSTEM )
C     IROUND  ROUND OFF TO NEAREST INTEGER          ( USER )
C     MINO    MINIMUM OF A SET OF INTEGERS         ( SYSTEM )
C     RCM     UNIFORM RANDOM NUMBER GENERATOR      ( SYSTEM )

```

```

C *****

```

```

C     2 SUBROUTINE DPCM2L( LNIN, NUMPPL, LNPOUT, LNOU, QTABLE,
C       NTABLE, MAX, RANGE, IST, WT, NGEN, NUSED )

```

```

C     DOUBLE INTEGER IST
C     INTEGER LNIN( NUMPPL ), LNPOUT( NUMPPL ), LNOU( NUMPPL )
C     INTEGER QTABLE( NTABLE ), WT( 4 ), DPCSUM

```

```

C     NRMLZF = WT( 1 ) + WT( 2 ) + WT( 3 ) + WT( 4 )
C     IRND = NRMLZF/2

```

```

C     MAX1 = 1 + MAX
C     LNOU( 1 ) = LNIN( 1 )

```

```

C     DO 1 I = 2, NUMPPL
C       DITHER = ( RCM( IST ) - .5 ) * RANGE
C       IDTHR = IROUND( DITHER )
C       J = MINO( I + 1, NUMPPL )
C       DPCSUM = LNPOUT( I - 1 ) * WT( 1 ) + LNPOUT( I ) * WT( 2 ) + LNPOUT( J ) * WT( 3 ) +
1         LNOUT( I - 1 ) * WT( 4 )
C       IPRED = ( DPCSUM + IRND ) / NRMLZF
C       IDIF = LNIN( I ) - IPRED + MAX1 + IDTHR
C       IDIF = MAXO( IDIF, 1 )
C       IDIF = MINO( IDIF, NTABLE )
C       LNOUT( I ) = IPRED + QTABLE( IDIF ) - IDTHR
C       LNOUT( I ) = MAXO( LNOUT( I ), 0 )

```

```

C     1 CONTINUE

```

```

C     NGEN = 1
C     NUSED = 1

```

```

C     RETURN
C     END

```

C--DPCMIX MODIFIED DPCM

C
C IDENTIFICATION

C TITLE DPCMIX
C AUTHOR GE MONAGHAN
C VERSION A.01
C DATE MARCH 31, 1977
C LANGUAGE FORTRAN
C SYSTEM PDP-15

C
C PURPOSE

C THIS SUBROUTINE FORMS A PREDICTOR FROM THE FOUR
C NEAREST PREVIOUS DPCM VALUES PLUS THE FIVE REM-
C AINING NEAREST LOW PASS FILTERED VALUES.

C
C ENTRY POINT

C DPCMIX (LNIN, NUMPPL, LNLPP, LNXTLP, LNPOUT, LNOUT,
C QTABLE, NTABLE, MAX, RANGE, IST, WT, NGEN,
C NUSED)

C
C ARGUMENT LISTING

C LNIN INT INPUT LINE OF DATA
C NUMPPL INT NUMBER OF POINT PER INPUT/OUTPUT LINE
C LNLPP INT CURRENT LOW PASS FILTERED LINE
C LNXTLP INT NEXT LOW PASS FILTERED LINE
C LNPOUT INT PREVIOUS OUTPUT LINE
C LNOUT INT CURRENT OUTPUT LINE
C QTABLE INT QUANTIZING TABLE
C QTABLE(IDIF) GIVES THE RECONSTRUCTED VALUE
C FOR ANY DIFFERENCE OF VALUE IDIF
C NTABLE INT DIMENSION OF QTABLE
C MAX INT MAXIMUM VALUE OF ANY INPUT LINE
C RANGE REAL RANGE OF DITHER
C IST DINT SEED OF RANDOM NUMBER GENERATOR
C WT INT 3X3 ARRAY OF WEIGHTS TO USE WHEN SUMMING
C PREVIOUS DPCM'D AND NEXT DATA POINTS WITH
C CURRENT DATA POINT
C NGEN INT NUMBER OF OUTPUT RECORDS GENERATED (=1)
C NUSED INT NUMBER OF INPUT RECORDS USED (=1)

C
C INTERNAL VARIABLES

C DPCSUM INT WEIGHTED SUM OF 3 NEAREST NEIGHBORS ON
C PREVIOUS DPCM LINE
C ITSUM INT WEIGHTED SUM OF LAST DPCM, CURRENT AND
C NEXT INPUT POINTS
C NITSUM INT WEIGHTED SUM OF 3 NEAREST NEIGHBORS ON
C NEXT INPUT LINE
C IPRED INT PREDICTED VALUE, COMPUTED FROM DPCSUM+
C ITSUM+NITSUM

```

C
C ROUTINES CALLED
C
C      MINO      MINIMUM OF A SET OF INTEGERS      (SYSTEM)
C      MAXO      MAXIMUM OF A SET OF INTEGERS      (SYSTEM)
C      IROUND    ROUND OFF TO NEAREST INTEGER      (USER)
C      RCM       UNIFORM RANDOM NUMBER GENERATOR   (SYSTEM)
C
C *****
C
C      SUBROUTINE DPCMHX ( LNIN, NUMPPL, LNLPP, LNXTLP, LNPOUT,
2          LNOUT, QTABLE, NTABLE, MAX, RANGE,
3          IST, WT, NGEN, NUSED )
C
C      DOUBLE INTEGER IST
C      INTEGER LNOUT (NUMPPL), QTABLE (NTABLE), LNLPP (NUMPPL)
C      INTEGER LNIN (NUMPPL), LNXTLP (NUMPPL), LNPOUT (NUMPPL)
C      INTEGER WT ( 3, 3 ), DPCSUM
C
C      MAXP1 = MAX + 1
C--     NMPLM1 = NUMPPL - 1
C      NRMLZF = 0
C
C      DO 1 I=1,3
C      DO 1 J=1,3
C      NRMLZF = NRMLZF + WT (I,J)
1      CONTINUE
C
C      IRND = NRMLZF/2
C
C      DO 1ST AND LAST DPCM'S SPECIALLY
C
C--     LNOUT (1) = LNIN (1)
C--     LNOUT (NUMPPL) = LNIN (NUMPPL)
C
C      NOW DO THE REST OF THE LINE
C
C      DO 1000 I=2,NUMPPL
C
C      IM1 = I-1
C      IP1 = MINO ( I+1, NUMPPL )
C
C      DITHER=(RCM (IST)-.5) *RANGE
C      IDTHR=IROUND (DITHER)
C
C      DPCSUM = LNPOUT (IM1) *WT (1,1) + LNPOUT (I) *WT (1,2) +
2          LNPOUT (IP1) *WT (1,3)
C      ITSUM = LNOUT (IM1) *WT (2,1) + LNLPP (I) *WT (2,2) +
2          LNLPP (IP1) *WT (2,3)
C      NXTSUM = LNXTLP (IM1) *WT (3,1) + LNXTLP (I) *WT (3,2) +
2          LNXTLP (IP1) *WT (3,3)

```

```
C      IPRED = (DPCSUM + ITSUM + NITSUM + IRND)/NRMLZF
      IDIF  = LMIN(I) - IPRED + MAXP1+IDTHR
      IDIF=MAXO(1,IDIF)
      IDIF=MINO(NTABLE,IDIF)
      LNOUT(I) = IPRED + QTABLE(IDIF)-IDTHR
      LNOUT(I) = MAXO(LNOUT(I),0)

C      1000  CONTINUE
C
      NGEN = 1
      NUSED = 1

C
      RETURN
      END
```

*--RMSERR ROOT MEAN SQUARE ERROR

* IDENTIFICATION

* TITLE RMSERR
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE SEPTEMBER 29, 1978
* LANGUAGE RATFOR
* SYSTEM PDP-15

* PURPOSE

* THIS ROUTINE COMPUTES THE RMS ERROR BETWEEN TWO LINES
* OF INTEGER INPUT DATA

* ENTRY POINT

* RMSERR (LNIN, LNOUT, NUMPPL, ERROR)

* ARGUMENT LISTING

* LNIN INT FIRST LINE OF DATA
* LNOUT INT SECOND LINE OF DATA
* NUMPPL INT NUMBER OF POINTS PER LINE
* ERROR REAL ROOT MEAN SQUARE ERROR

* ROUTINES CALLED

* NONE

* SUBROUTINE RMSERR (LNIN, LNOUT, NUMPPL, ERROR)

* TYPE STATEMENTS

* INTEGER LNIN(NUMPPL), LNOUT(NUMPPL)
* REAL ERROR, MSE, SUM2

* SUM2 = 0.

* COMPUTE SUM OF THE SQUARES

* FOR (I = 1; I <= NUMPPL; I = I + 1)
* \$(
* SUM2 = SUM2 + (LNIN(I) - LNOUT(I))**2
* \$)

* COMPUTE THE RMS ERROR

* MSE = SUM2 / NUMPPL

ERROR = SQRT (MSE)

RETURN

END

References

- [1] C. A. Andrews, J. M. Davies, and G. R. Schwarz, "Adaptive Data Compression," Proc. IEEE, vol. 55, pp. 267-277, March 1967.
- [2] A. V. BalaKreshnan, R. L. Kutz, and R. A. Stampfl, "Adaptive data compression for video signals," NASA Goddard Space Flight Center, Rept. X-730-66-110.
- [3] R. L. Kutz and J. A. Sciulli, "An adaptive image compression system and its performance in a noisy channel," presented at the Internatl. Information Theory Symp., 1967.
- [4] L. D. Davisson, "Theory of adaptive data compression," in Advances in Communications Systems, New York: Academic Press, 1966, pp. 173-192.
- [5] M. Tasto and P. A. Wintz, "Image Coding by Adaptive Block Quantization," IEEE Trans. Comm. Tech. vol. COM-19, pp. 957-972, 1971.
- [6] A. Habibi and G. S. Robinson, "A survey of digital picture coding," Comput., pp. 22-34, May 1974.
- [7] C. K. Chow, B. L. Deekshathln, and L. S. Loh, "Some computer experiments in picture processing for data compaction," Comput. Graphics and Image Processing, vol. 3, pp. 203-214, 1974.
- [8] R. M. Haralick, "A Facet Model for Image Data," unpublished paper.
- [9] R. M. Haralick and N. Kattiyakulwanich, "A fast two dimensional Karhunen-Loeve transform," SPIE, vol. 66, pp. 144-158, 1975.
- [10] C. C. Cutler, "Differential PCM," U.S. Patent 2 605 361, July 29, 1952.
- [11] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," IEEE Trans. Comput., vol. C-23, pp. 90-93, Jan. 1974.
- [12] A. Habibi and P. A. Wintz, "Hybrid Coding of Pictorial Data," IEEE Trans. Comm. Tech., vol. COM-22, no. 5, pp. 614-624, 1974.
- [13] J. A. Rose, W. K. Pratt, G. S. Robinson, and A. Habibi, "Interframe Transform Coding and predictive coding methods," in 1975 Proc. ICC, IEEE Catalog 75 CH 0971-2GSCB, pp. 23.17-23.21.
- [14] T. S. Huang and J. W. Woods, "Picture bandwidth compression by block quantization" presented at the 1969 Int. Symp. Information Theory, Ellenville, NY.
- [15] A. Habibi and P. A. Wintz, "Image Coding by Linear Transformation and Block Quantization," IEEE Trans. Comm. Tech., vol. COM-19, pp. 50-62.

- [16] J. Max, "Quantizing for minimum distortion," IRE Trans. Information Theory, vol. IT-6, pp. 7-12, March 1960.
- [17] R. E. Bellman, Dynamic Programming, Princeton University Press, Princeton, NJ, 1957.

APPENDIX B

APPENDIX B

KEY TO "HOW TO INTERPRET NAMES OF IMAGES"

(A) NON-CAUSAL, ADAPTIVE

K C A L I _ _ _ _ _
1 2 3 4 5 6 7 8 9

C C A L I _ _ _ _ _
1 2 3 4 5 6 7 8 9

L A D Y 3 _ _ _ _ _
1 2 3 4 5 6 7 8 9

<u>location #</u>	<u>Description</u>
6	An "0" in this position means output has been processed. Any other letter means no output processing.
7	An "R" in this position means the "raw" curves were used. An "F" means the fitted ones were used.
8	An "F" in this position means compression of 8:1. An "C" means 16:1 compression.
9	An "A" means no buffer constraint. A "B" means buffer has been constrained.

(B) NON-ADAPTIVE, EQUAL ALLOCATION

K C A L _ _ _ _ _
1 2 3 4 5 6 7 8 9

C C A L _ _ _ _ _
1 2 3 4 5 6 7 8 9

L A D 3 _ _ _ _ _
1 2 3 4 5 6 7 8 9

<u>location #</u>	<u>Description</u>
7, 8, 9	An RET in locations 7, 8, 9 means that the image is a non-adaptive, equal allocation image.
5*	An "E" in this location (for CCAL & KCAL) means 8:1 compression. An "F" means 16:1
4**	An "E" means 8:1. An "F" means 16:1

*for CCAL & KCAL only

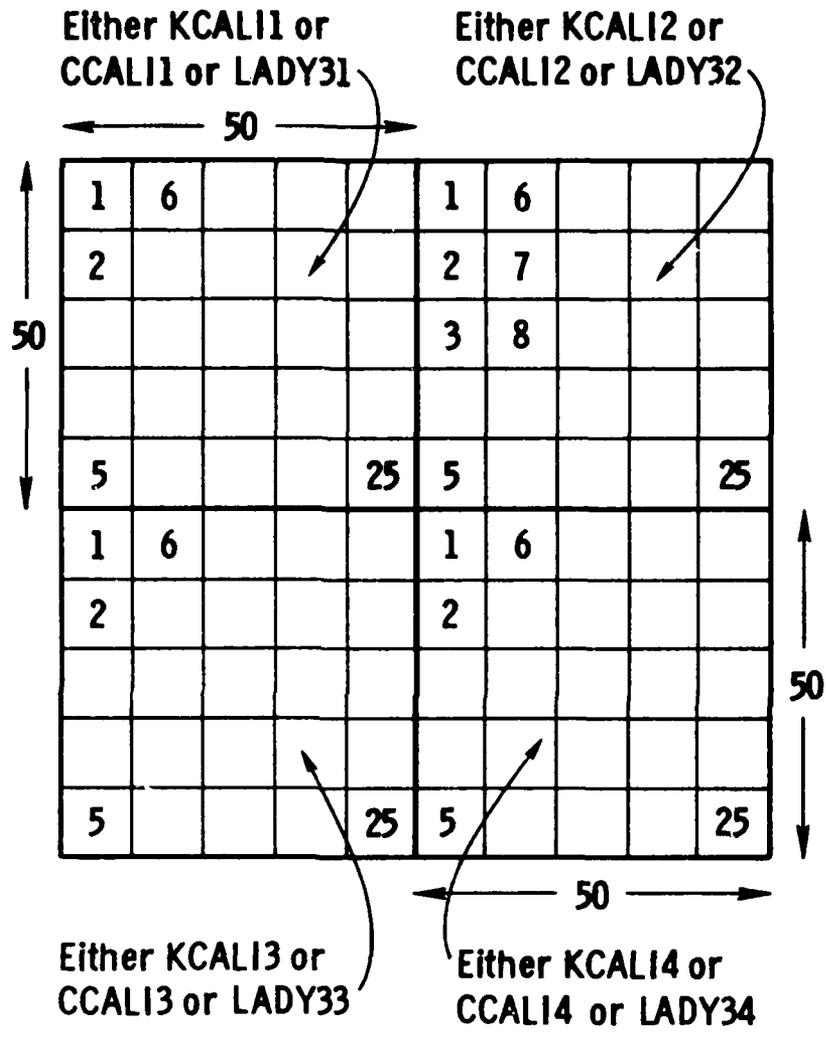
**for LADY image only

(C) ADAPTIVE CAUSAL

C C A L _ _ _ _ _
1 2 3 4 5 6 7 8 9

L A D Y _ _ _ _ _
1 2 3 4 5 6 7 8 9

<u>location #</u>	<u>Description</u>
1, 2, 3, 4, 5	If the letter "B" occurs in any one of these locations, it means the image is a "causal" one.
6, 8, 9	Same interpretation as in "non-causal" case.
7	As always, "R" doesn't mean anything.

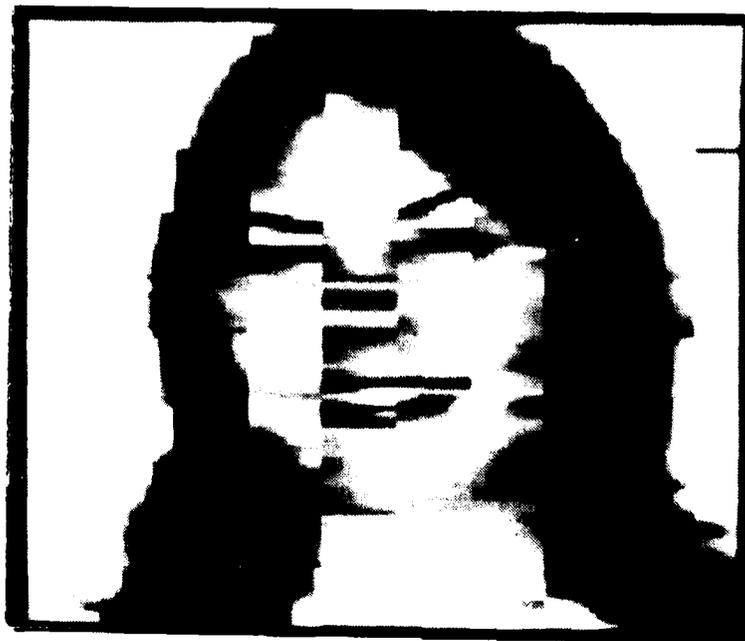


APPENDIX C

Set of Photographs



A) KCALIF EQU

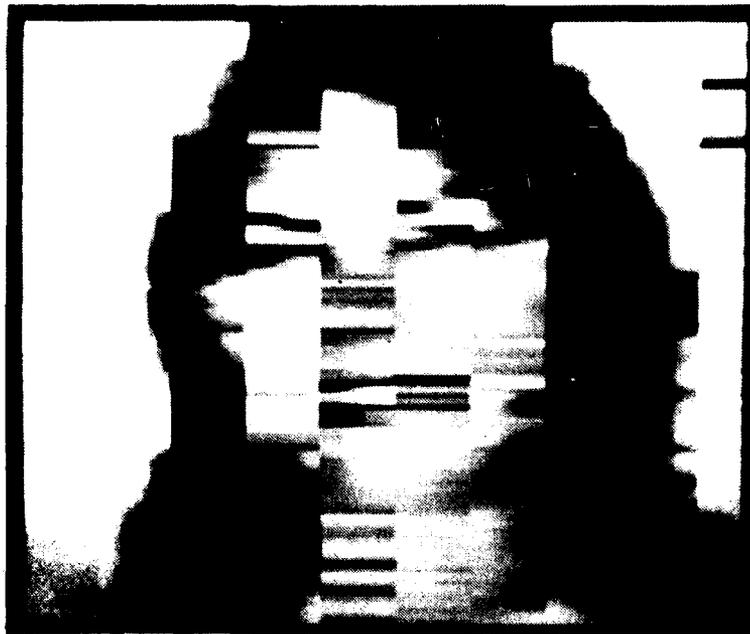


B) LADYB RFA

CAUSAL



C) LADYB RFB



D) LADYB RCA

CAUSAL



A) LADY3 QN3



B) LADY3 QN4

RICHARDS IMAGES

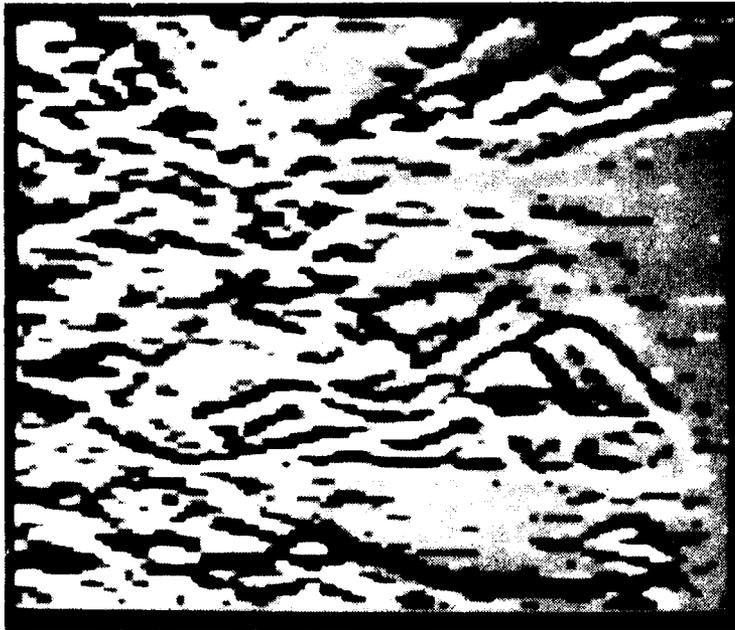


C) LADY3 QN5



D) KCALIF QN3

RICHARDS IMAGES



A) KCALIF QN4

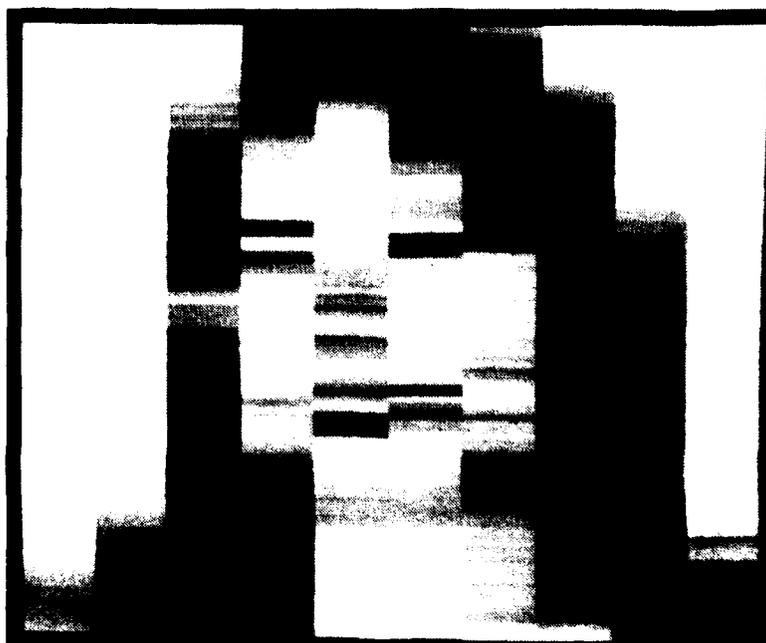


B) KCALIF QN5

RICHARDS IMAGES

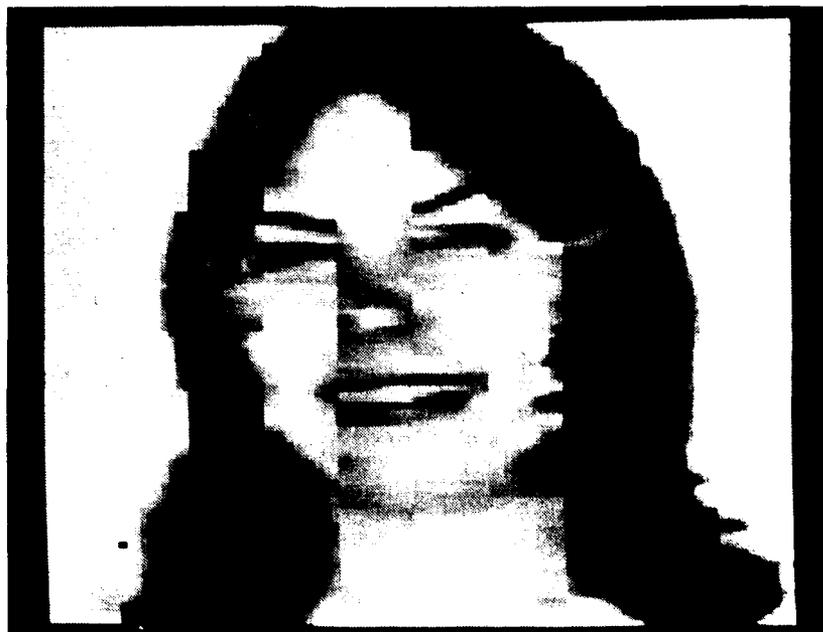


C) LADE3 RET



D) LADF3 RET

EQUAL ALLOCATION NON-ADAPTIVE



A) LADY3 RFA

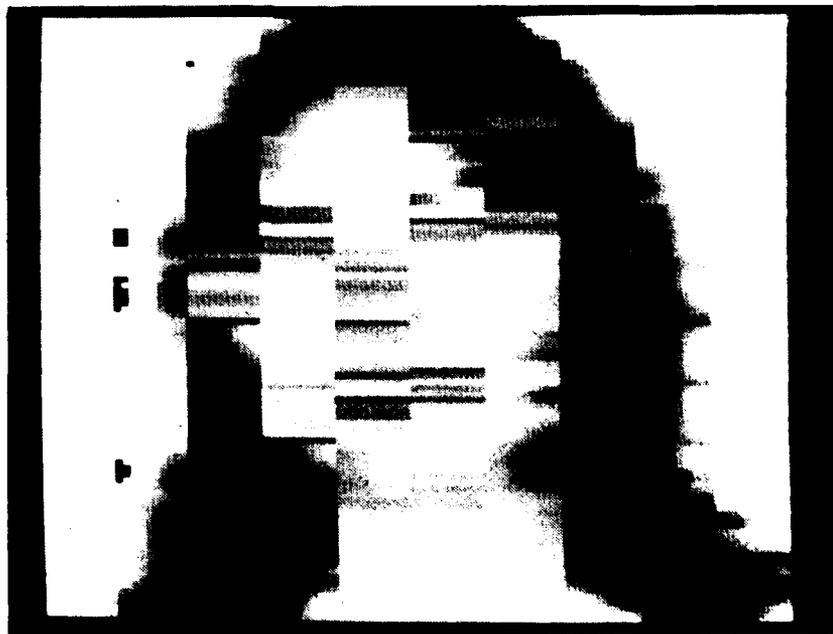


B) LADY3 RFB

NON-CAUSAL

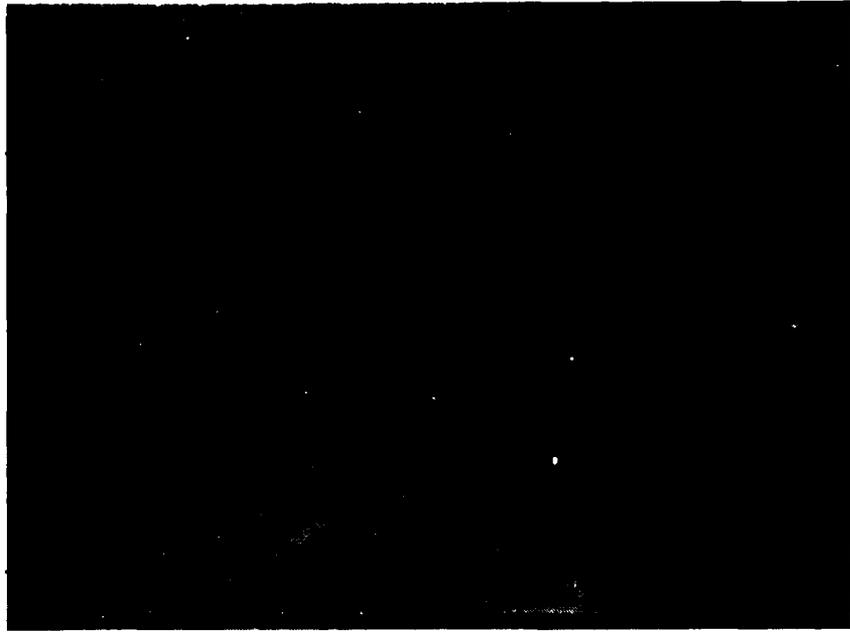


C) LADY3 FFA

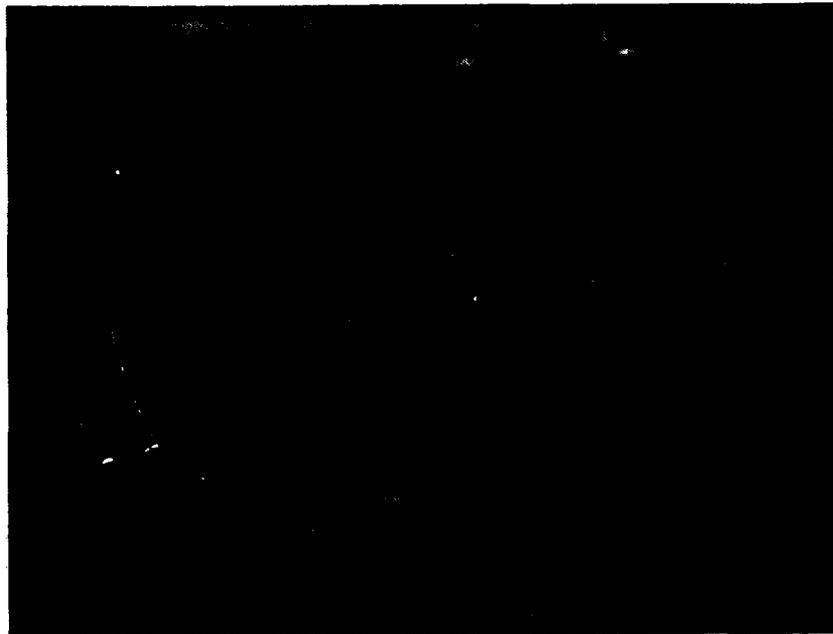


D) LADY3 RCB

NON-CAUSAL

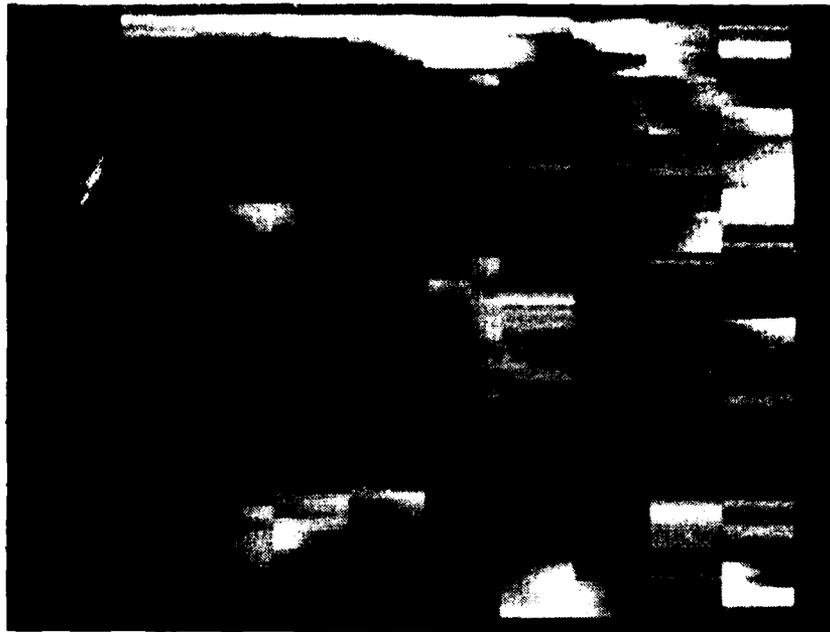


A) KCALIF RCA

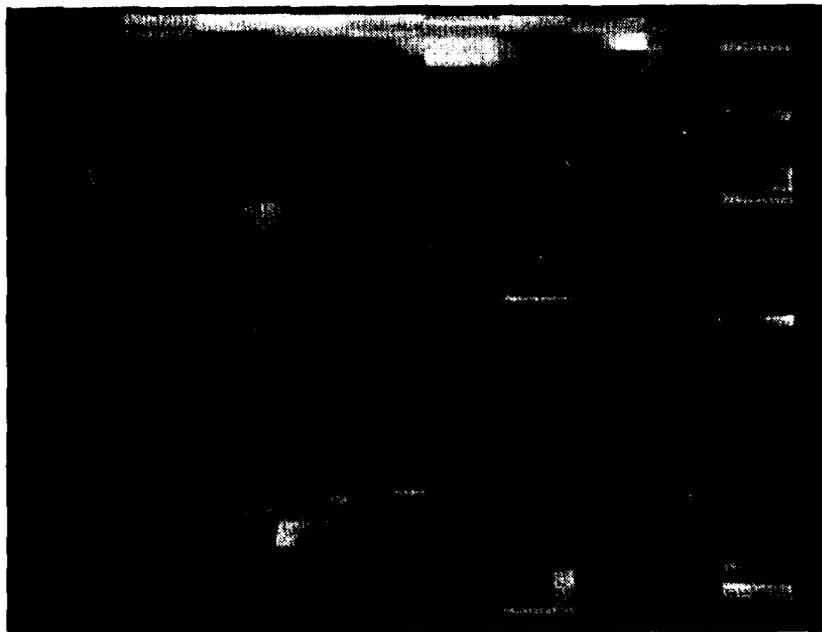


B) KCALIF RCB

NON-CAUSAL



C) KCALIF RFA

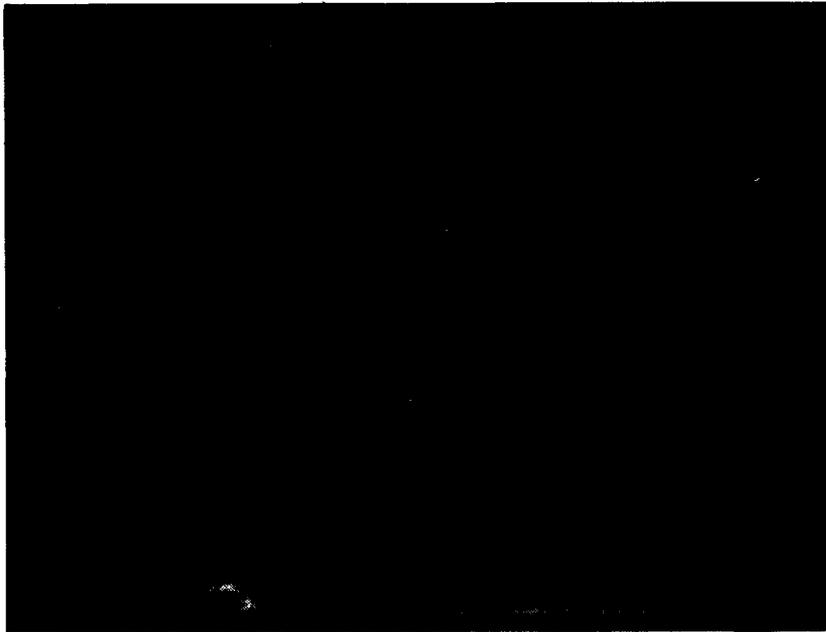


D) KCALIO RFA

NON-CAUSAL



A) KCALIO FFA

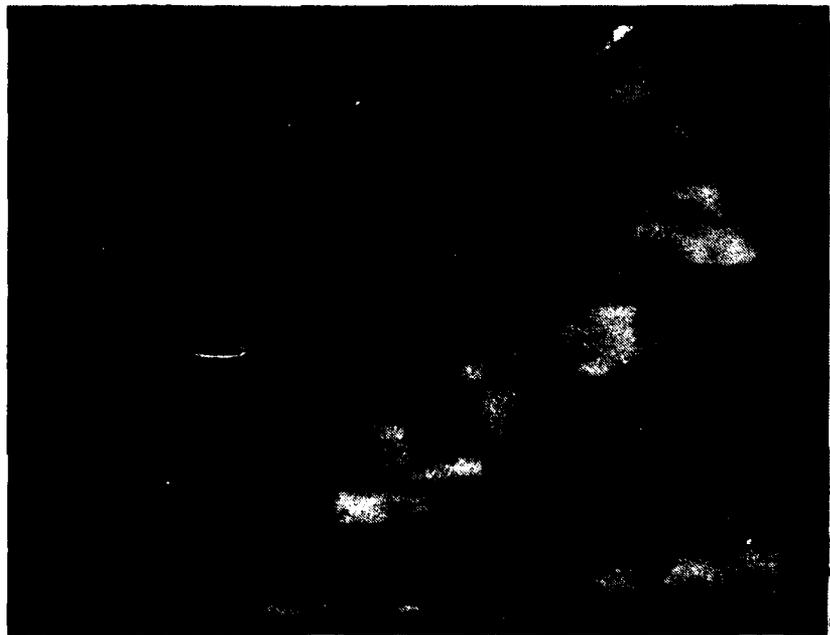


B) KCALIF FFB

NON-CAUSAL

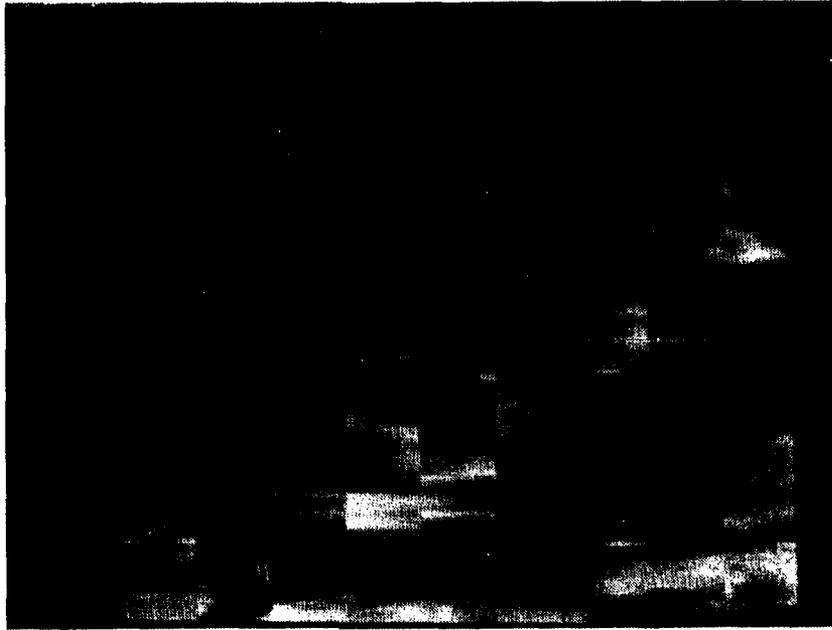


C) KCALIO FFB



D) CCALIF RFA

NON-CAUSAL



A) CCALIF RFB

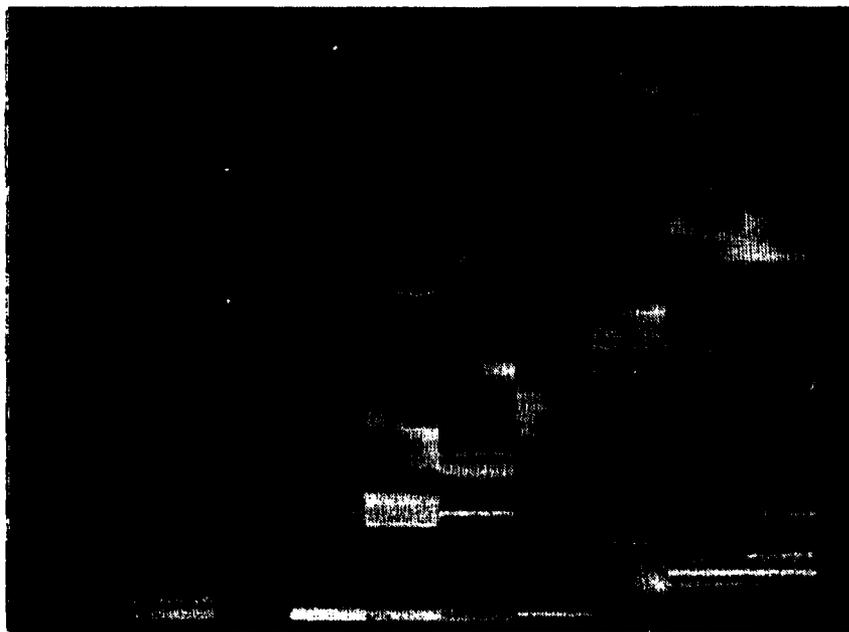


B) CCALIF FFA

NON-CAUSAL



C) CCALIF RCA

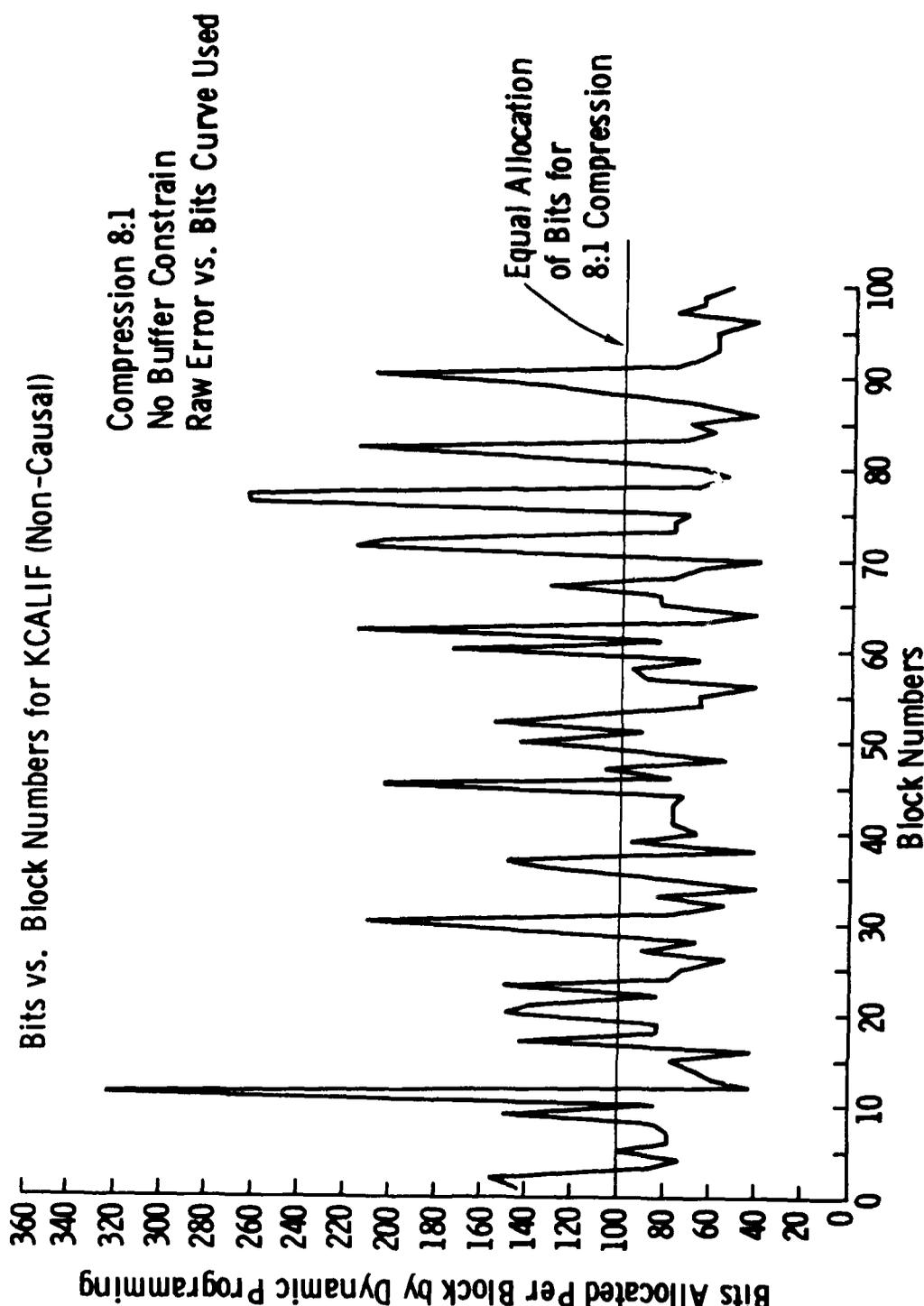


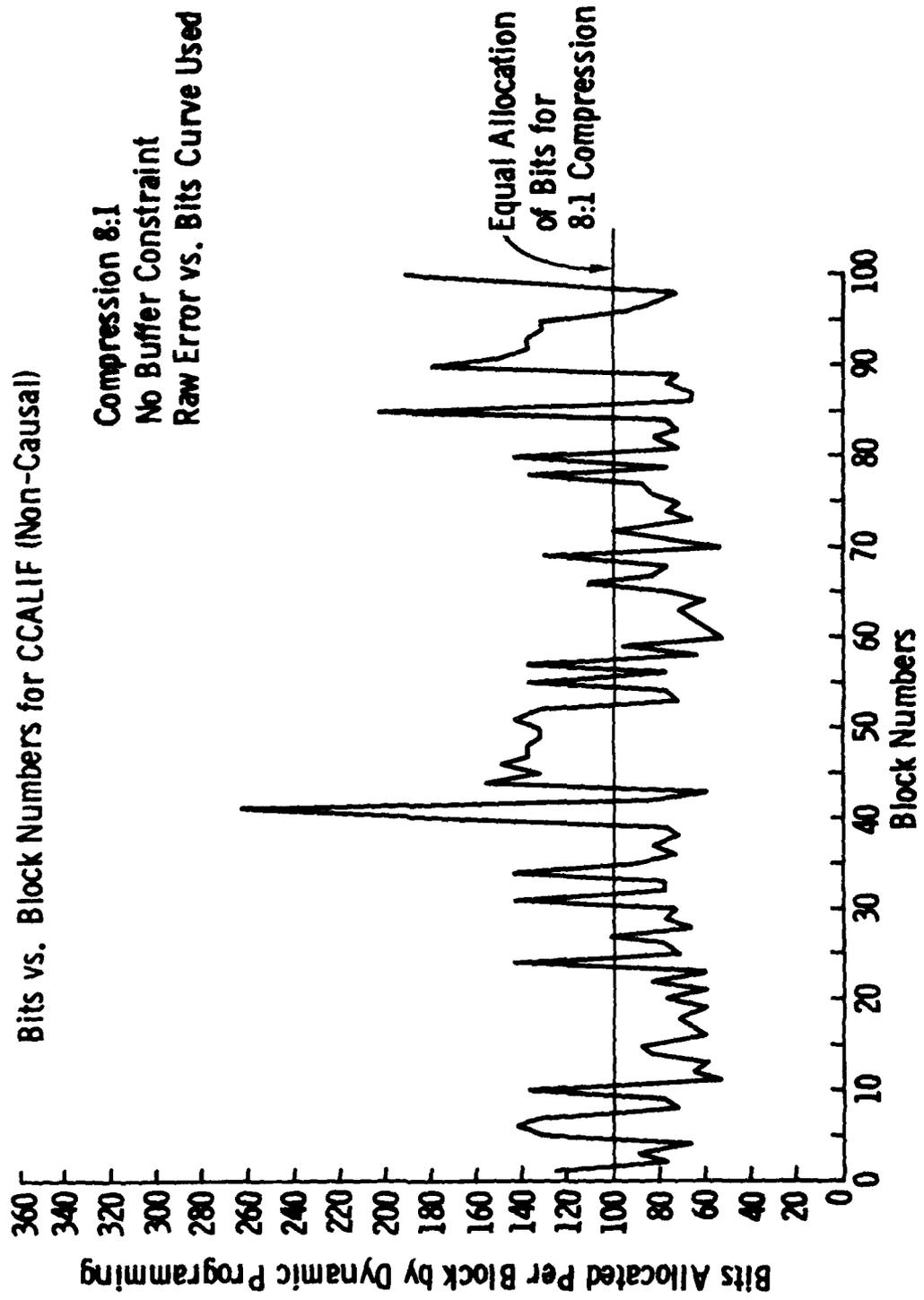
D) CCALIF RCB

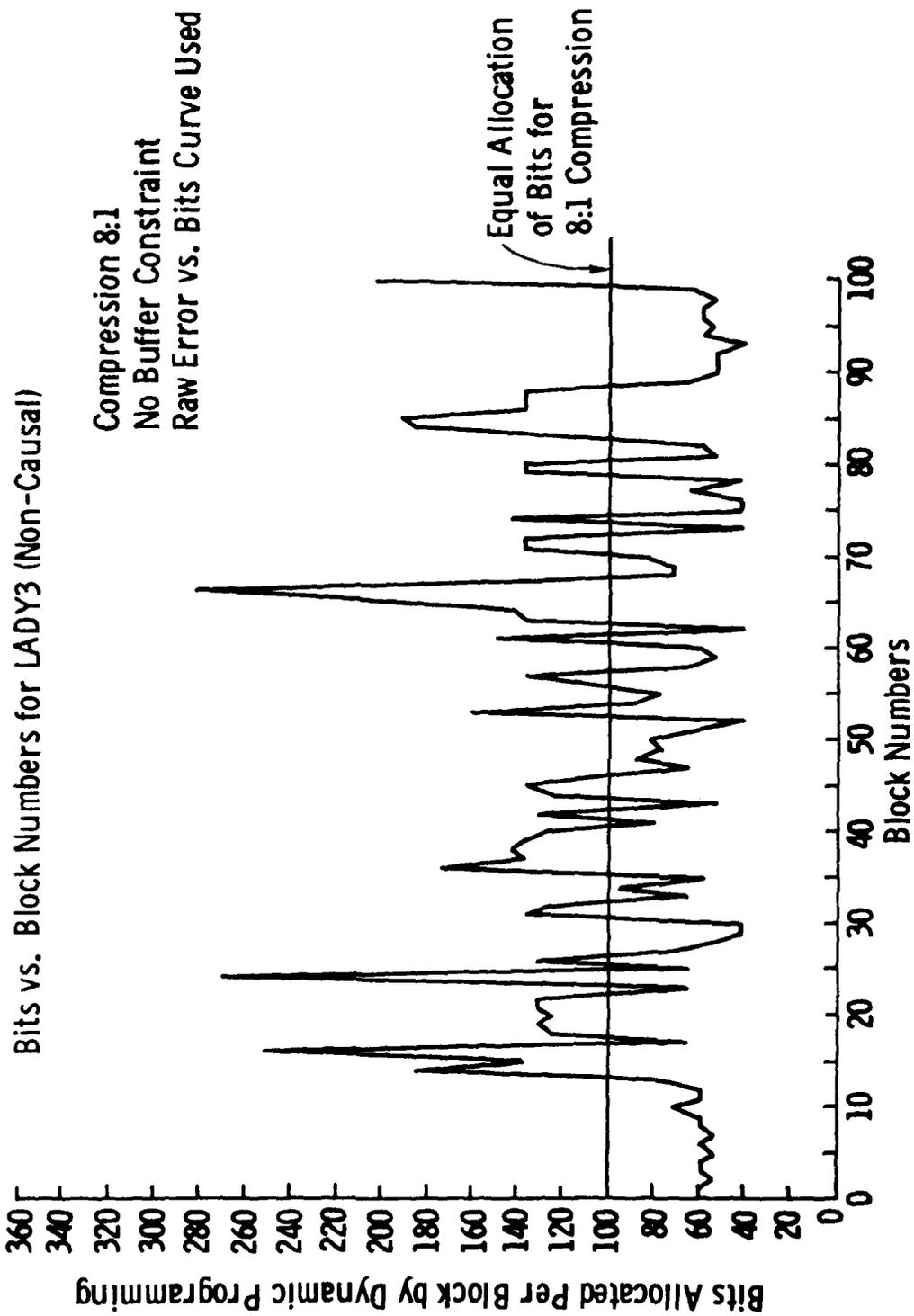
NON-CAUSAL

APPENDIX D

Set of Graphs





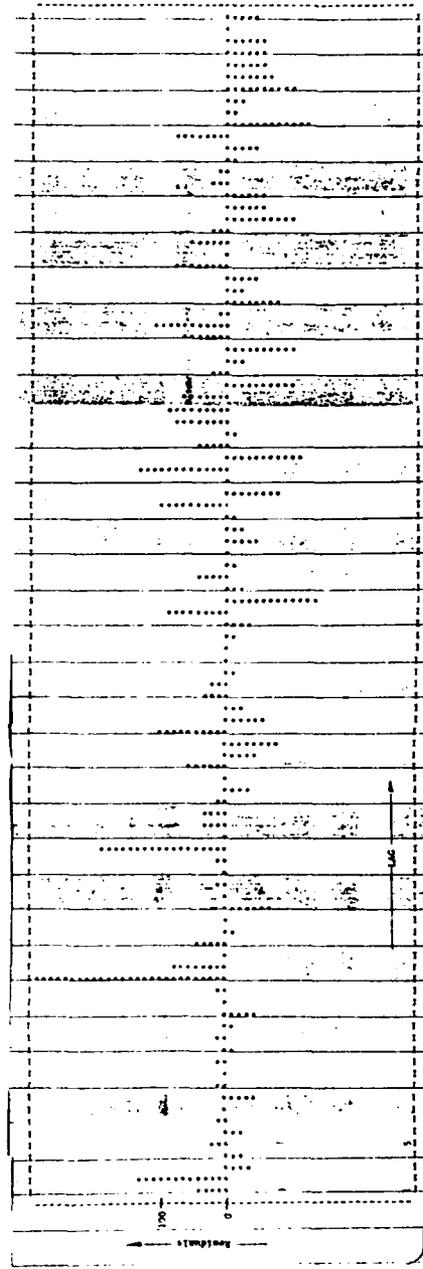


APPENDIX E

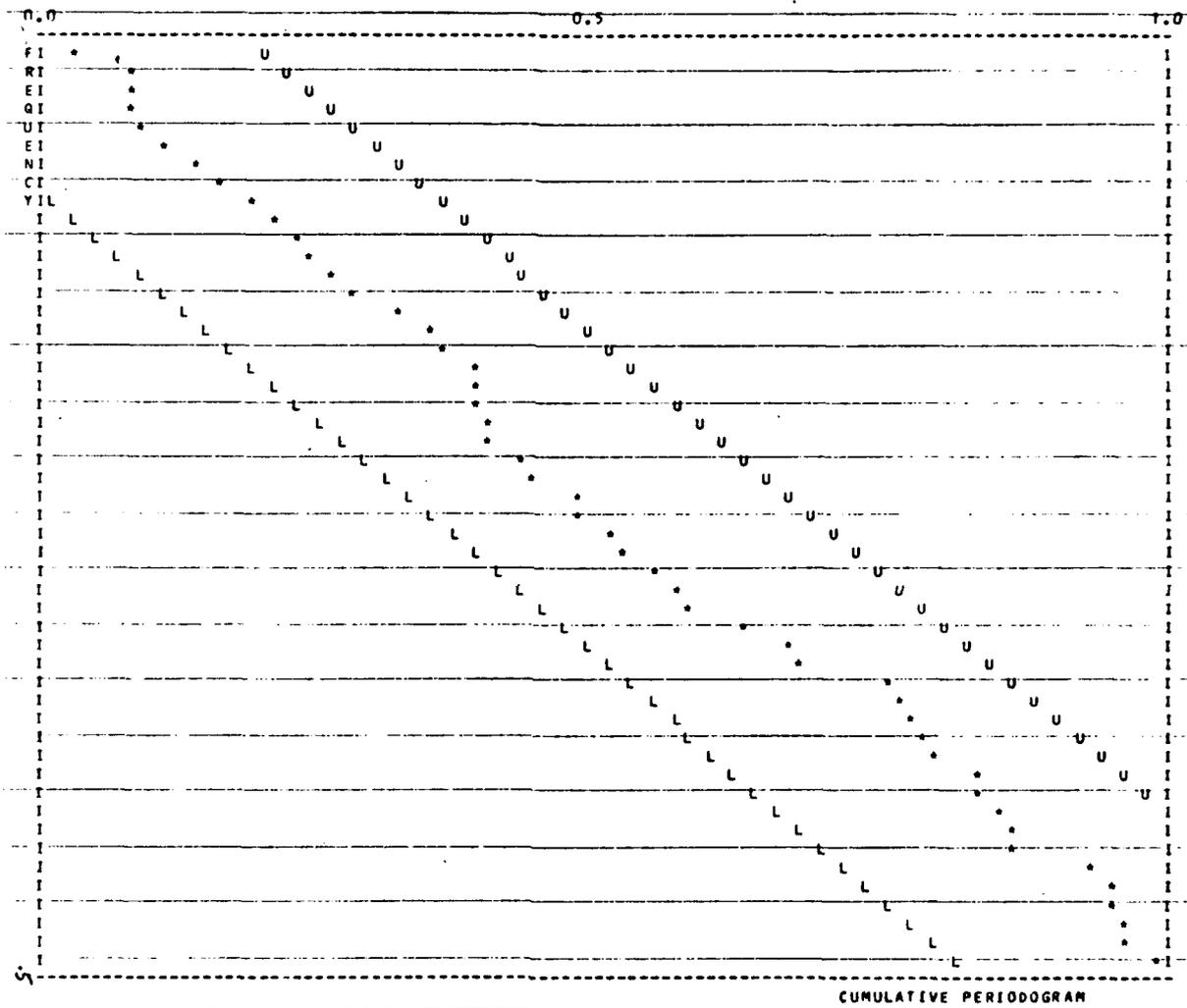
Printouts 1 & 2

Standard Deviation of the Residuals
(Not Variance)

Residual Std. Dev. = 68.7268
Durbin Watson Stat = 2.0019
Number of Negative Residuals = 53
Number of Positive Residuals = 47
Number of Runs = 48
Z Statistic for the Runs Test = -0.5689
(see page II-22)

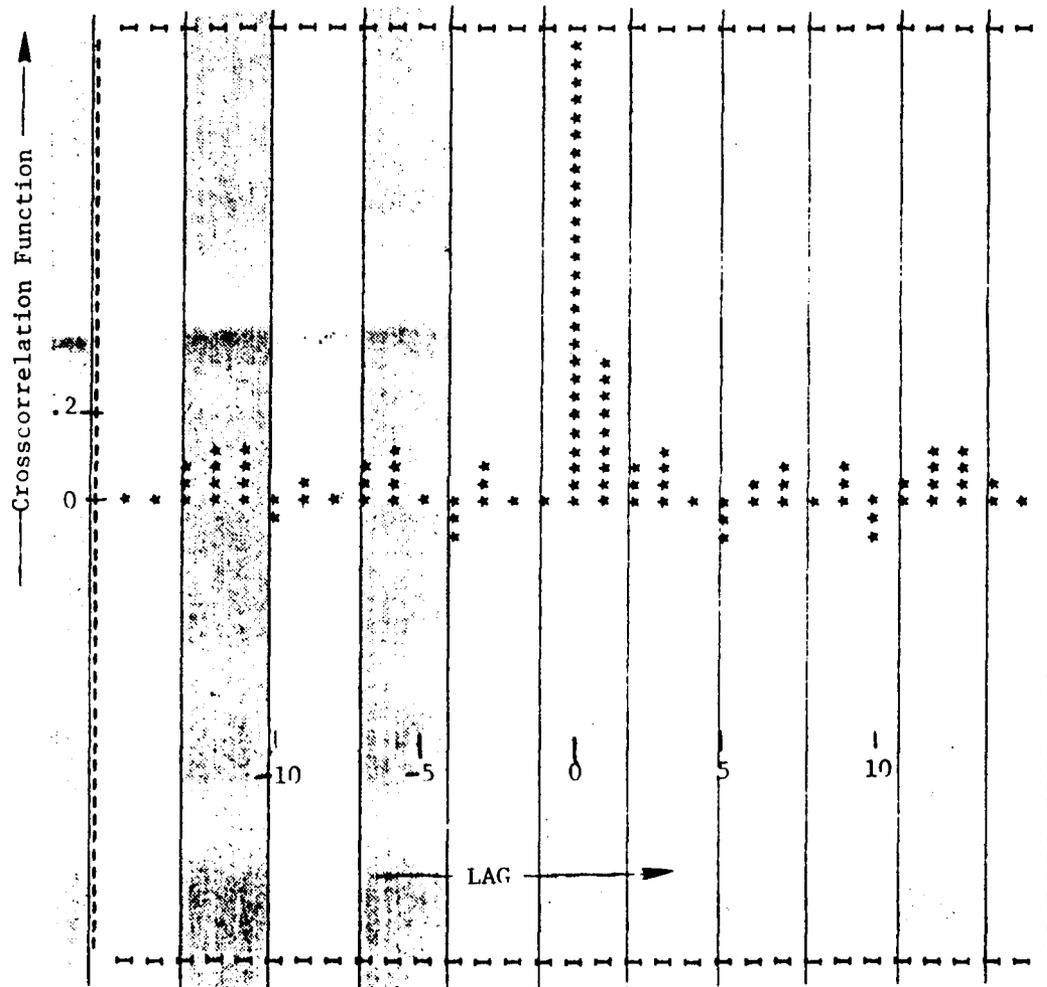


GRAPH OF RESIDUALS

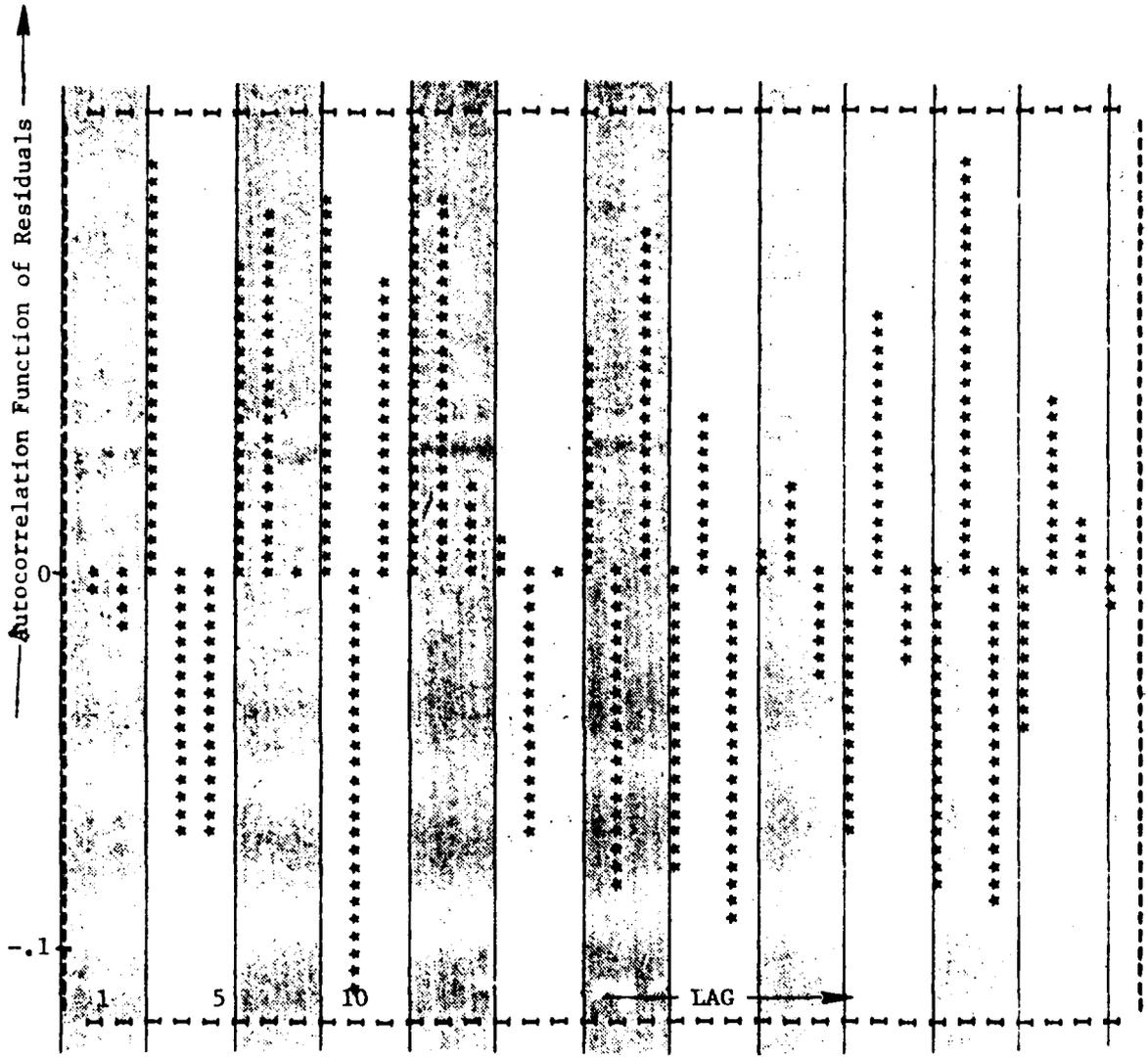


CUMULATIVE PERIODOGRAM

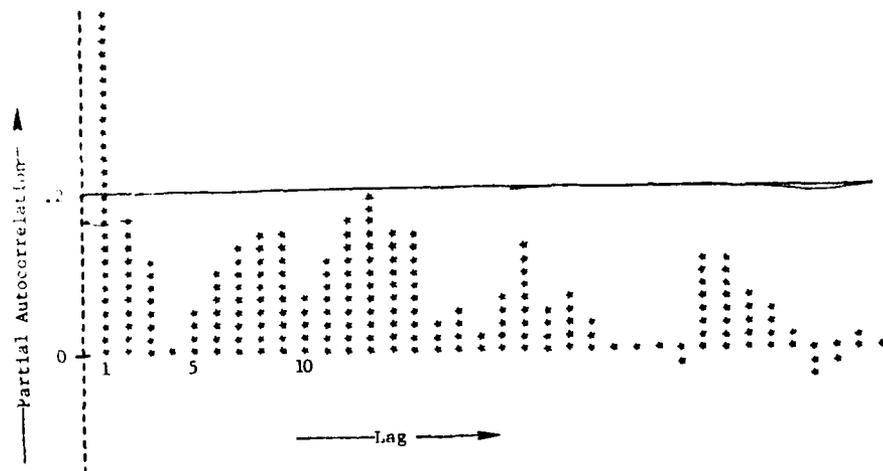
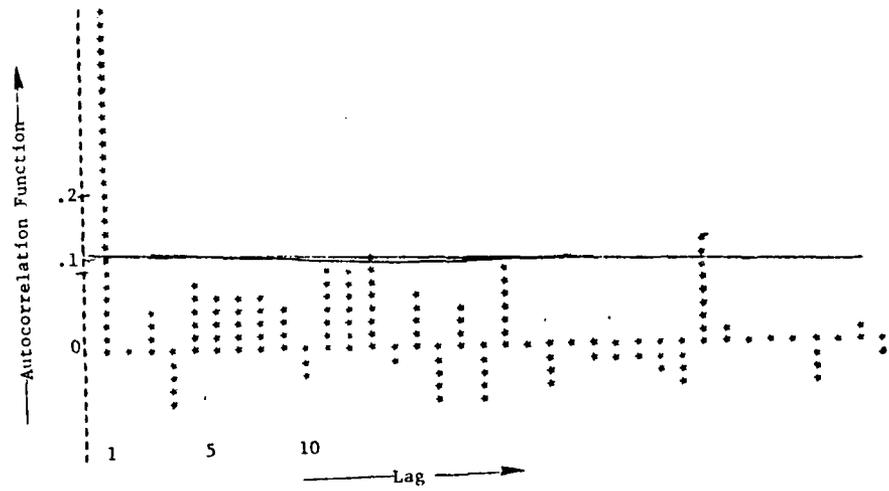
CUMULATIVE NORMALIZED PERIODOGRAM OF THE RESIDUALS



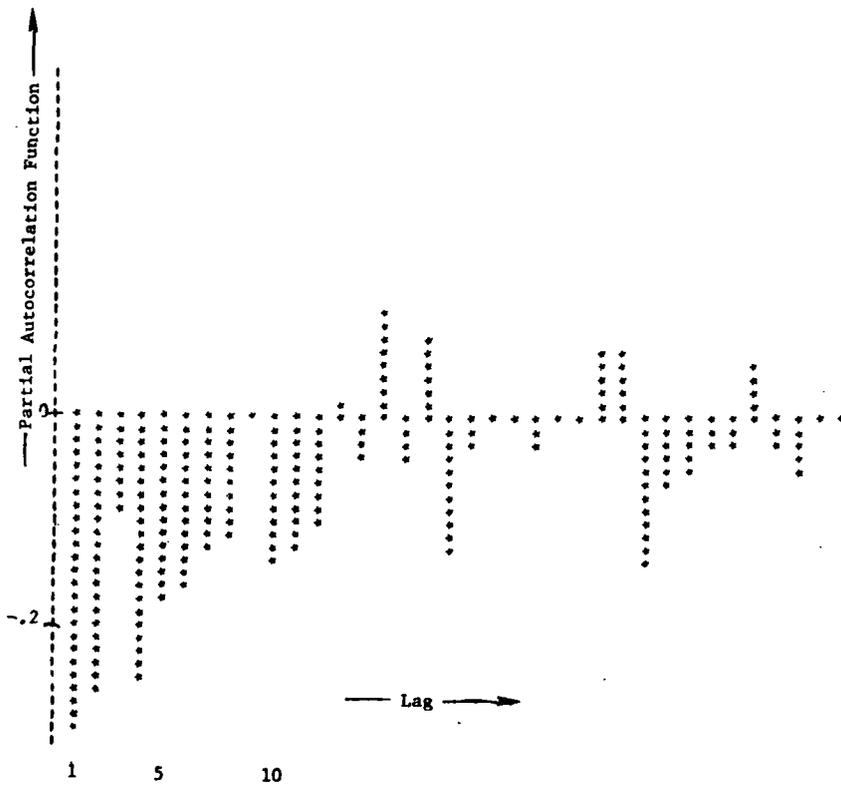
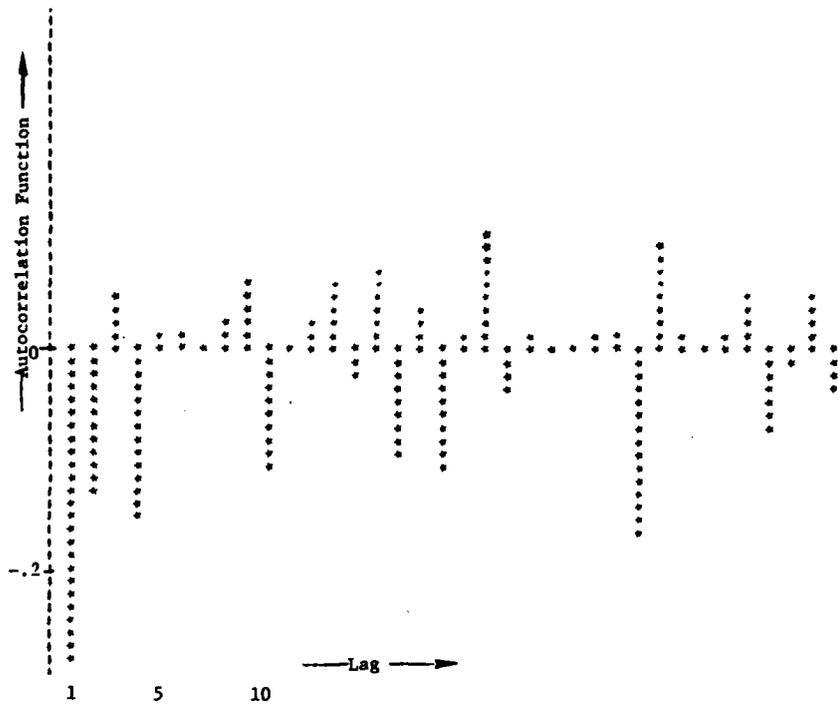
CROSSCORRELATION OF OBSERVED DATA
AND RESIDUALS



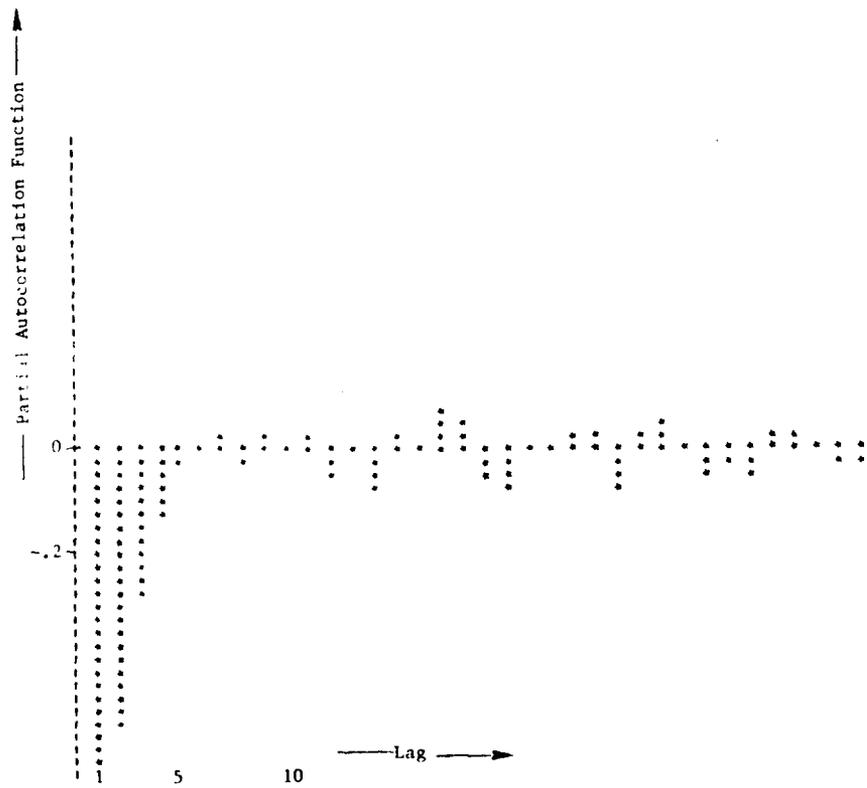
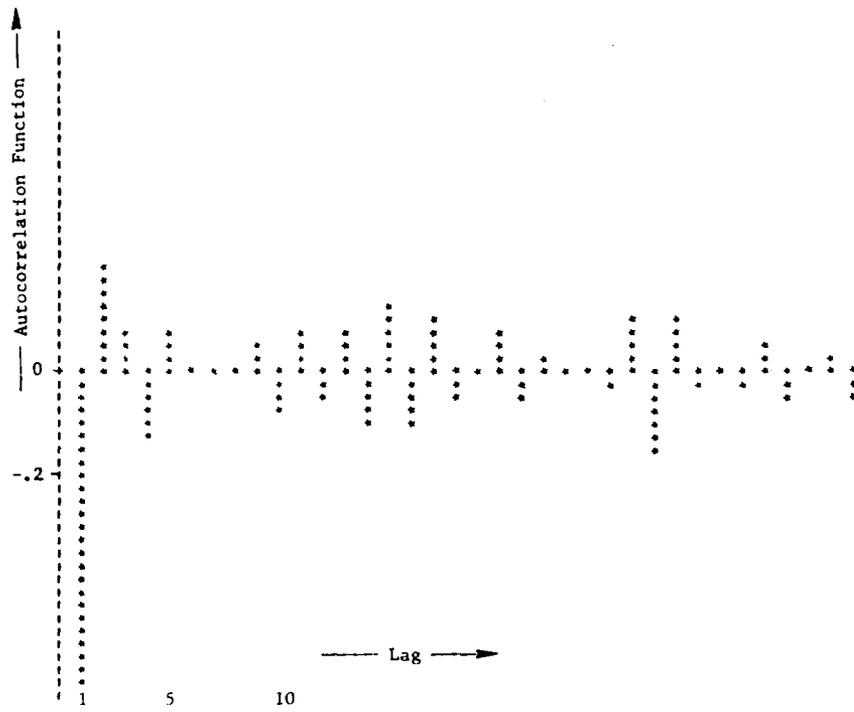
AUTOCORRELATION FUNCTION OF RESIDUALS



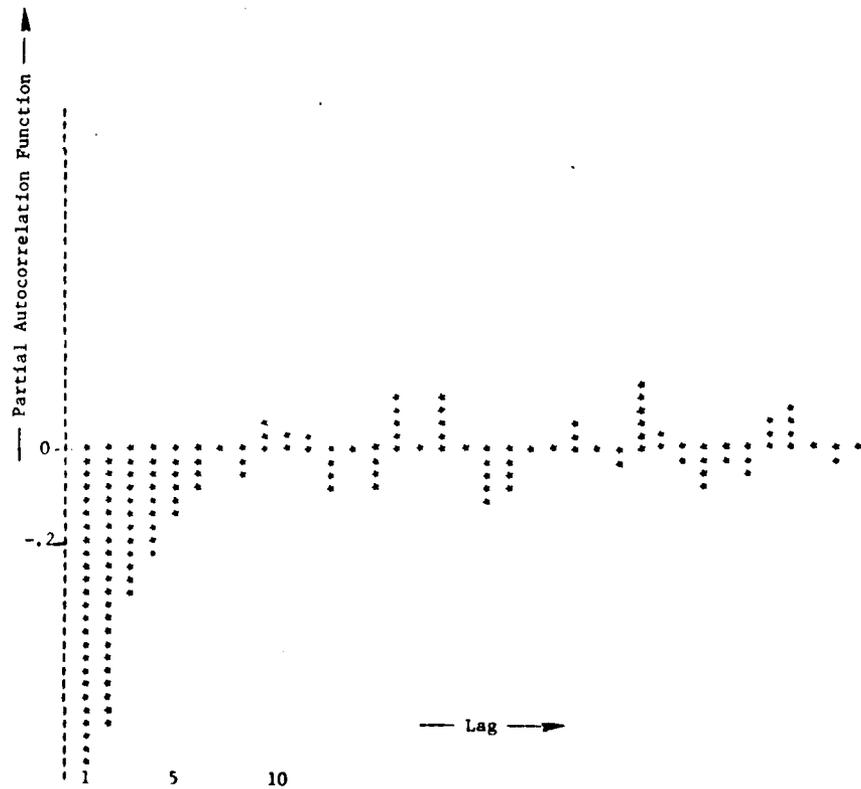
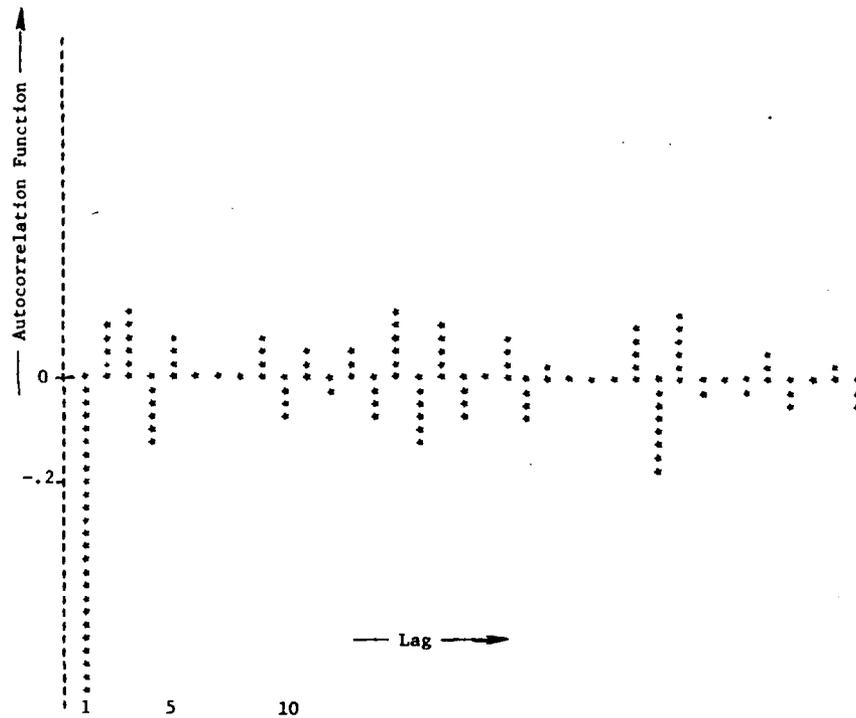
(1) Raw Data



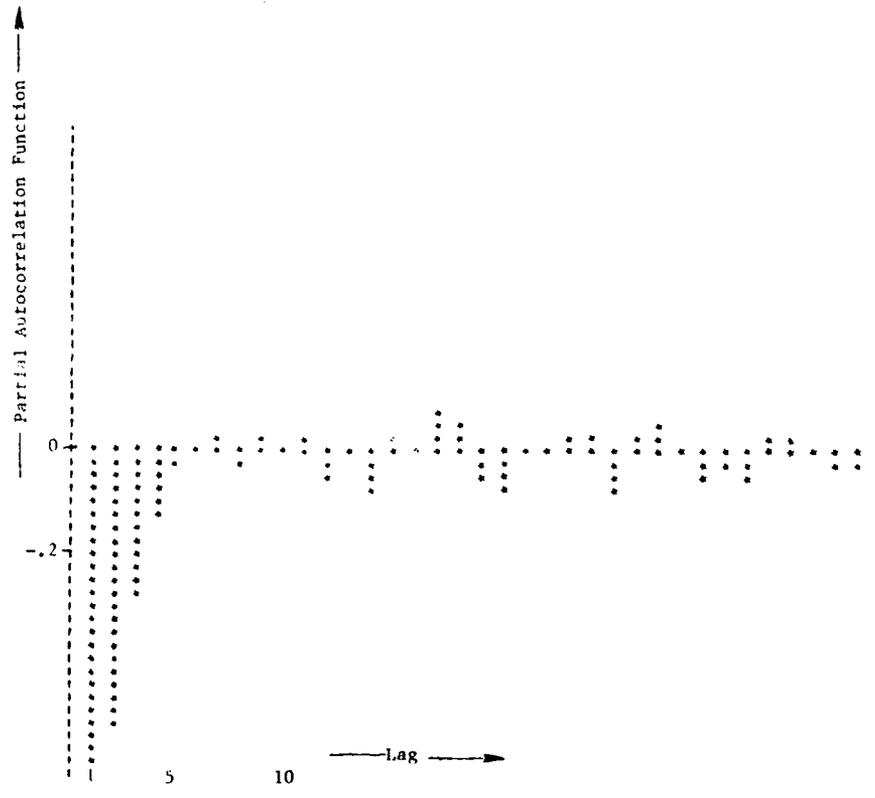
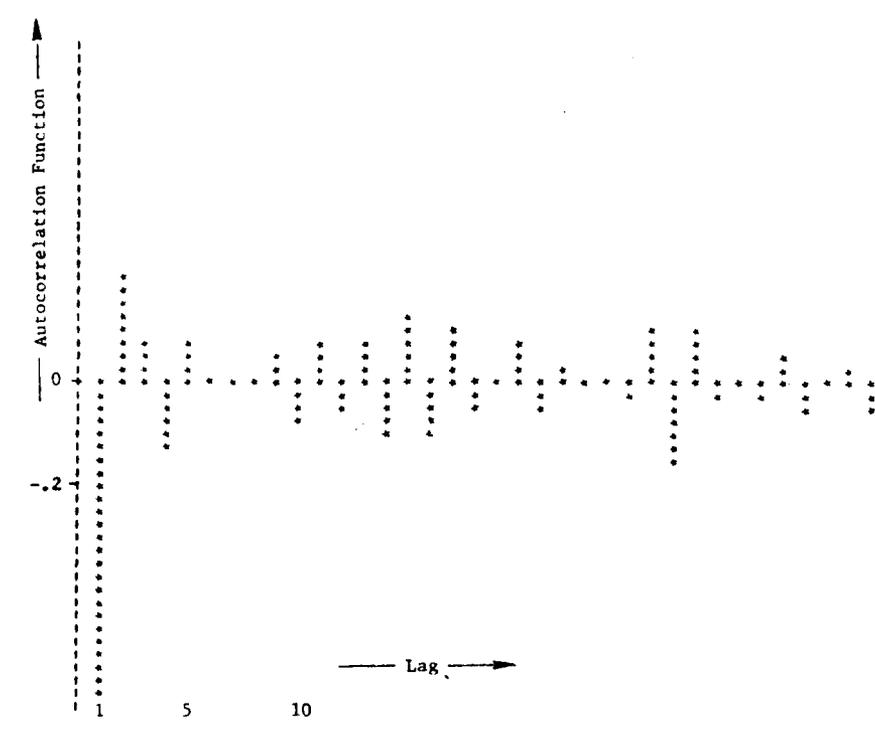
(2) First Difference



(3) Second Difference



(4) Third Difference



(5) Fourth Difference

APPENDIX F

Technical Report - Causal DPCM Image Compression

TECHNICAL REPORT
CAUSAL DPCM IMAGE COMPRESSION .

by

Oscar A. Zuniga

Virginia Polytechnic Institute and State University

TABLE OF CONTENTS

	<u>Page</u>
TITLE	i
TABLE OF CONTENTS	ii
LIST OF FIGURES	iii
LIST OF TABLES	iv
1. INTRODUCTION	1
2. CAUSAL DPCM IMAGE COMPRESSION	1
3. EXPERIMENTAL RESULTS	3
4. CONCLUSIONS	7
APPENDIX B.1 COMPUTER PROGRAM DOCUMENTATION	26

LIST OF FIGURES

Figure		Page
3.1	Original picture	11
3.2	Reconstructed pictures using a fixed bit allocation 2-D DPCM	12
3.3	Reconstructed pictures using a non-causal 2-D DPCM with no buffer constraints	14
3.4	Reconstructed pictures using a non-causal 2-D DPCM with buffer constraints	18
3.5	Reconstructed pictures using a causal 2-D DPCM with no buffer constraints	20
3.6	Reconstructed pictures using a causal 2-D DPCM with buffer constraints	24

LIST OF TABLES

Table		Page
3.1	Several error measures between original and reconstructed pictures	8

1. Introduction

We investigate here an approach for solving the causal DPCM image compression problem which arises naturally as an extension of the analytic solution of the non-causal problem under gaussian data assumptions. Experimental results are provided comparing the performance of the causal, non-causal, and fixed bit allocation DPCM compression.

2. Causal DPCM Image Compression

One of the important results obtained earlier in this report is the fact that the quantization mean square error versus bit rate functions for blocks in a large class of images match closely those predicted by rate distortion theory under gaussian data assumptions.

That is,

$$e^2 = \sigma^2 \exp(-\alpha b) \quad (2.1)$$

where σ^2 is the variance of the DPCM corrections, b is the bit rate; α is an image dependent constant, and e^2 the resulting quantization mean square error. Use of this model leads to an analytic solution of the non-causal problem. This solution is stated as follows: Let the image be partitioned into K blocks; let $\sigma_1^2, \dots, \sigma_K^2$ be the variances of the DPCM corrections in each of these blocks; let c be the desired channel capacity bit rate; and B the available number of bits for block 1 through K . The optimal bit allocations b_1, \dots, b_K are given by

$$b_k = c + \frac{1}{\alpha} (\ln \sigma_k^2 - \ln \sigma_f^2) \quad (2.2)$$

for $k = 1, \dots, K$

where,

$$\ln \sigma_f^2 = \frac{1}{K} \sum_{k=1}^K \ln \sigma_k^2 \quad ; \quad c = \frac{B}{K}$$

Here σ_f^2 is the geometric average variance of the image blocks.

This simple result states that blocks with variance larger than the average variance are allocated bits above the channel rate and accordingly blocks with variance smaller than the average variance are allocated bits below the channel rate.

Notice that the non-causal aspect of the allocation procedure of equation 2.2 reduces to the knowledge of the average variance σ_f^2 before any processing is done. This also suggests an attractive simple approach for solving the causal problem which involves an initial estimation and subsequent updating of σ_f^2 .

Assume blocks 1 through $t-1$ have been processed. Block t is the current block being processed. $(\sigma_p^2)^t$ is the variance of the DPCM corrections associated with block t , and $(\sigma_f^2)^t$ is the current estimate for σ_f^2 . Let B^t be the number of bits available for blocks t through K . Then allocate b^t bits to the current block in the following way,

$$b^t = \min \left\{ B^t, c + \frac{1}{\alpha} (\ln (\sigma_p^2)^t - \ln (\sigma_f^2)^t) \right\} \quad (2.3)$$

σ_f^2 can be updated as

$$\ln (\sigma_f^2)^{t+1} = (1 - \gamma) \ln (\sigma_f^2)^t + \gamma \ln (\sigma_p^2)^t \quad (2.4)$$

and

$$B^{t+1} = B^t - b^t$$

In order to make this procedure work better and to be able to attain higher compression ratios we must allow the assignment of zero bits to those blocks with very small variance as determined by (2.3). In this case the last row in the previously reconstructed north block and the last column in the previously reconstructed west block may be used to estimate the current block in the usual DPCM manner but no corrections are transmitted. The transmitter sends to the receiver the values of the mean and variance of the difference between the original block and the estimated one. These values are used by the receiver to improve its estimate of the current block by adding to it uniform noise of the same mean and variance. In order to avoid running out of bits before all blocks are processed the following modification can be made to (2.3).

$$\text{Let } b_o^t = c + \frac{1}{\alpha} (\ln (\sigma_p^2)^t - \ln (\sigma_f^2)^t)$$

then,

$$b^t = \begin{cases} \min \{ B^t, b_o^t \} & \text{if } B^t \geq (K - t + 1)c \text{ when } c < 1 \\ & \text{or } B^t \geq (K - t + 1) \text{ when } c \geq 1 \\ \min \{ 1, b_o^t \} & \text{otherwise} \end{cases} \quad (2.5)$$

An additional modification to (2.5) can be made to take into account buffer size constraints. Let R be the size of the bit buffer and let r^t be the state of the buffer right after processing the (t-1)th block. The following bit assignment will then prevent overflowing or underflowing the buffer.

If $r^t > c$

$$b^t = \begin{cases} \min \{B^t, b_0^t, R - r^t\} & \text{if } B^t \geq (K - t + 1)c \text{ when } c < 1 \\ & \text{or } B^t \geq (K - t + 1) \text{ when } c \geq 1 \\ \min \{1, b_0^t\} & \text{otherwise} \end{cases} \quad (2.6)$$

If $r^t \leq c$

$$b^t = \begin{cases} \min \{B^t, \max \{c, b_0^t\}\} & \text{if } B^t \geq (K - t + 1)c \text{ when } c < 1 \\ & \text{or } B^t \geq (K - t + 1) \text{ when } c \geq 1 \\ \min \{1, \max \{c, b_0^t\}\} & \text{otherwise} \end{cases}$$

3. Experimental Results

Several experiments were performed using the "girl" image shown in Figure 3.1. This image consists of 256 x 256 pixels and was blocked into 16 x 16 blocks before processing. This image was originally quantized to 256 levels. The compression technique used was a simple 2-D DPCM with an equally weighted predictor. The DPCM was initialized using the first row and first column in the image. The amount of dither used was 0.5 the standard deviation of the DPCM corrections. The corrections in each block were quantized using a max quantizer based on normal distribution with the same mean and variance as those of the corrections.

Fixed bit allocation procedures at bit rates of 1.0 and 2.0 b/p, and causal and non-causal procedures at 0.85, 1.0, 1.5 and 2.0 b/p were carried out. These bit rates do not take into account the amount of bits necessary for transmitting the first row and first column in the image without compression (9 b/p); the bits necessary to encode the mean and variance for each block (8 bits for the mean; 12 bits for the variance) and in the case of the non-causal and causal procedures the bits necessary to indicate the number of quantizing bits used for each pixel within a block (3 bits). After making the necessary corrections the effective bit rates correspond to 1.14 and 2.14 b/p for fixed bit allocation procedures and 1.0, 1.15, 1.65 and 2.15 b/p for causal and non-causal procedures.

Figure 3.2 shows the reconstructed pictures obtained by using the fixed bit allocation procedure. Figures 3.3 and 3.4 show the reconstructed pictures using non-causal procedures with buffer and no buffer constraints. Figures 3.5 and 3.6 show the corresponding results using a causal bit allocation procedure. We can observe the improvement in image quality obtained when using the

non-causal and causal bit allocation procedures over the fixed bit allocation procedures. This is also noticed by looking at Tables 3.1 (a)-(e) that show 3 measures of the error between the original picture and the reconstructed pictures. It is also observed that no additional degradation resulted when using the buffer constrained causal allocation as compared to the unconstrained case for the buffer size used.

4. Conclusions

We have shown that causal adaptive DPCM compression performs significantly better than fixed bit allocation DPCM compression, however the pixel by pixel nature of conventional DPCM puts a limit to the amount of compression we can obtain. Full advantage of the adaptive scheme we have developed is taken when the DPCM is carried out in a block by block fashion, that is instead of estimating the graytone value of a single pixel based on past reconstructed neighboring pixels we estimate an entire block of pixels based on past reconstructed neighboring blocks. The simplest such scheme would only make use of the last row in the top block and the last column in the left block. Such estimation scheme can be done in the form of a least squares fit and may be also constrained to minimize the errors in the top and left borders. A block in this case will be replaced by its fitting parameters. Blocks corresponding to regions in the image of low complexity will then require lower dimensional fitting than those blocks corresponding to regions of high complexity.

bit rate error	1.14	2.14
RMS	10.16	4.90
ABSE	5.93	2.92
MAXE	123	101

(a) Fixed bit allocation

bit rate error	1.0	1.15	1.65	2.15
RMS	8.29	7.18	4.70	3.44
ABSE	5.45	4.75	3.15	2.23
MAXE	112	104	100	90

(b) Non-causal; No buffer constraints

Table 3.1 Different error measures between original and reconstructed pictures; RMS is the root mean square error; ABSE is the mean absolute error (the mean of the absolute value of the difference image); MAXE is the maximum error (the maximum absolute value in the difference image).

bit rate error	1.0	1.15
RMS	8.58	7.22
ABSE	5.54	4.77
MAXE	115	104

(c) Non causal; Buffer size is 10% of minimum size that guarantees no constraints.

bit rate error	1.0	1.15	1.65	2.15
RMS	9.38	8.30	5.83	3.74
ABSE	5.82	5.13	3.45	2.34
MAXE	132	132	133	100

(d) Causal; No buffer constraints.

bit rate error	1.0	1.15
RMS	9.38	8.30
ABSE	5.82	5.13
MAXE	132	132

(e) Causal; Buffer size is 10% of minimum size that guarantees no constraints.



Figure 3.1 Original picture. This image consists of 256 x 256 picture elements quantized to 256 gray levels (8 b/p).



Figure 3.2 Reconstructed pictures using a fixed bit allocation 2-D DPCM.

(a) Compression: 1.14 b/p.



Figure 3.2 (continued)

(b) Compression: 2.14 b/p.



Figure 3.3 Reconstructed pictures using a non-causal 2-D DPCM with no buffer constraints.

(a) Compression: 1.0 b/p.



Figure 3.3 (continued)

(b) Compression: 1.15 b/p.



Figure 3.3 (continued)

(c) Compression: 1.65 b/p.



Figure 3.3 (continued)

(d) Compression: 2.15 b/p.



Figure 3.4 Reconstructed pictures using a non-causal 2-D DPCM with buffer constraints. Buffer size is 10% of minimum size that guarantees no constraints.

(a) Compression: 1.0 b/p.



Figure 3.4 (continued)

(b) Compression: 1.15 b/p.



Figure 3.5 Reconstructed pictures using a causal 2-D DPCM with no buffer constraints.

(a) Compression: 1.0 b/p.



Figure 3.5 (continued)

(b) Compression: 1.15 b/p.



Figure 3.5 (continued)

(c) Compression: 1.65 b/p.



Figure 3.5 (continued)

(d) Compression: 2.15 b/p.



Figure 3.6 Reconstructed picture using a causal 2-D DPCM with buffer constraints. Buffer size is 10% of minimum size that guarantees no constraint.

(a) Compression: 1.0 b/p.



Figure 3.6 (continued)

(b) Compression: 1.15 b/p.

APPENDIX B.1

DPCM DATA COMPRESSION PACKAGE

I. FIXED BIT ALLOCATION DPCM DATA COMPRESSION

FDPCHD

FDPCHI

FDPCHC

HVARDL

FIXBLK

GETEGS

BKDPCM

EGSINT

QTZSAM

DPCH2L

DPCMXX

MSERR

UPDEGS

II. CAUSAL AND NON-CAUSAL DPCM DATA COMPRESSION

DPCMCD

DPCMCI

DPCMDC

HVARDL

CBITAL

BITALO

WBITAL

BITALO

FIXBLK

GETEGS

BKDPCM

EGSINT

QTZSAN

DPCM2L

DPCMXX

MSERR

UPDEGS


```
INCLUDE GIPCOM
INCLUDE ERROR
INTEGER WORK ( .ARB )
INTEGER IDENT ( 20 ), IDNT2 ( 20 )
REAL FRAC
```

```
EQUIVALENCE (NPPL,IDENT(6)), (NLINS,IDENT(7))
EQUIVALENCE (NCOLS,IDENT(13)), (NROWS,IDENT(14))
EQUIVALENCE (NTBND,IDENT(17)), (NSBND,IDENT(18))
EQUIVALENCE (MODE,IDENT(19))
EQUIVALENCE (NTBND2,IDNT2(17)), (NSBND2,IDNT2(18))
EQUIVALENCE (MODE2,IDNT2(19))
```

```
CALL PPUSH ( 'FDPCHD' )
```

```
SET UP INPUT FILE
```

```
CALL RDKINL ( FDI1, IDENT, .OLD, IEV, %9999 )
CALL CLOSE ( FDI1 )
CALL RDKINL ( FDI2, IDNT2, .OLD, IEV, %9999 )
CALL CLOSE ( FDI2 )
```

```
CHECK INPUT FILE
```

```
IF ( MODE == 1 & MODE == 0 ) GO TO 9000
IF ( NTBND - NSBND <= 0 ) GO TO 9010
IF ( MODE2 == 1 & MODE2 == 0 ) GO TO 9000
IF ( NTBND2 - NSBND2 <= 0 ) GO TO 9010
```

```
GET USER INFORMATION
```

```
CALL FDPCHI ( FDI1, SELECT, FRAC, BTRATE, %9999 )
```

```
CHECK AVAILABLE SPACE
```

```
BLKSIZE = NROWS * NCOLS
WRKSIZE = 3*BLKSIZE + NPPL + NLINS
IF ( .OK == OSALOC ( WRKSIZE ) ) GO TO 9020
```

```
GET THE BAND TO USE
```

```
CALL WHBAND ( FDI1, IDENT, NU, BND, IEV, %9999 )
BND = 1
```

```
SET UP CTRL ADDRESS
```

```
CALL CTRL ( %8000 )
```

```
CALL THE NUMBER CRUNCHER
```

```
CALL FDPCHC ( FDI1, FDI2, FDO1, BND, WORK, WRKSIZE,
```

SELECT, FRAC, BTRATE, IEV, %9998)

*
CALL PPOP
RETURN

*
8000 CONTINUE

*
CONTROL T EXIT

*
CALL IGNORT
CALL CLOSE (FDI1)
CALL CLOSE (FDI2)
CALL CLOSE (PDO1)
RETURN

*
ABNORMAL CONDITIONS

*
9000 CONTINUE

*
NOT AN INTEGER FILE

*
IEV = -2012
GO TO 9998

*
9010 CONTINUE

*
NO NUMERIC BANDS

*
IEV = -5018
GO TO 9998

*
9020 CONTINUE

*
NOT ENOUGH WORK SPACE

*
IEV = -5010

*
9998 CONTINUE

*
ERROR IN SUBPROGRAM

*
CALL CLOSE (FDI1)
CALL CLOSE (FDI2)
CALL CLOSE (PDO1)

*
ABNORMAL RETURN

*
9999 RETURN 1
END


```

INCLUDE ERROR
INCLUDE TTCON

CHARACTER PD ( .PDLENGTH )
CHARACTER FDIO ( .FILENAMELENGTH )
CHARACTER ALT, BELLS
REAL PRAC
DATA ALT, BELLS / .ALTCHR, .BELLCHR /

CALL PPUSH ( 'FDPCHI' )

GET NAME INTO SYSTEM STANDARD FORMAT

IF ( .OK == OSGTNN ( PD, FDIO ) ) GO TO 9999

WRITE (RUNOT,6000) FDIO, BELLS, ALT
6000 FORMAT ( ' SELECT DPCH PREDICTOR FOR ', .FILENAMELENGTH A1,/,
           ' (2) 2-D DPCH   FLAT MODEL   LSE'//,
           ' (3) MOD DPCH   FLAT MODEL   LSE'//,
           ' (4) 2-D DPCH   FLAT MODEL   HVE'//,
           ' (5) MOD DPCH   FLAT MODEL   HVE'//,
           ' (6) 2-D DPCH   SLOPED MODEL  LSE'//,
           ' (7) MOD DPCH   SLOPED MODEL  LSE'//,
           ' (8) 2-D DPCH   SLOPED MODEL  HVE'//,
           ' (9) MOD DPCH   SLOPED MODEL  HVE'//,
           2A1 )
IF ( .OK == GETI ( PDRUNI, SELECT ) ) GO TO 9999

CHECK FOR LEGAL SELECTION NUMBER

WHILE ( SELECT < 2 | SELECT > 9 )
$(
WRITE (TTYOT,6000) FDIO, BELLS, ALT
IF ( .OK == GETI ( PTTYI, SELECT ) ) GO TO 9999
$)

WRITE (RUNOT,6010) BELLS, ALT
6010 FORMAT ( ' ENTER FRACTION OF STANDARD DEVIATION'//,
           ' THAT RANGE OF DITHER SHOULD BE --', 2A1 )
IF ( .OK == GETR ( PDRUNI, PRAC ) ) GO TO 9999

WRITE (RUNOT,6020) BELLS, ALT
6020 FORMAT ( ' ENTER BIT RATE --', 2A1 )
IF ( .OK == GETI ( PDRUNI, BTRATE ) ) GO TO 9999

CALL PPOP
RETURN

9999 CONTINUE

RETURN 1
END

```

* IDENTIFICATION

* TITLE FDPCHC
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE JULY 3, 1979
* LANGUAGE RATFOR
* SYSTEM IBM-370

* PURPOSE

* THIS ROUTINE READS BLOCKS FROM AN INPUT IMAGE IN A SCAN
* RASTER MODE AND PERFORMS A FIXED BIT ALLOCATION DPCM ON
* THOSE BLOCKS USING A NUMBER OF DPCM TECHNIQUES AS SEL-
* ECTED BY THE USER

* ENTRY POINT

* FDPCHC (FDI, FDI2, PDO, BND, WORK, WRKSIZ, SELECT,
* FRAC, BTRATE, IEV, ALTRET)

* ARGUMENT LISTING

* FDI INT FILE DESCRIPTOR FOR INPUT IMAGE
* FDI2 INT FILE DESCRIPTOR FOR LOW PASS FILTERED
* IMAGE
* PDO INT FILE DESCRIPTOR FOR OUTPUT IMAGE
* BND INT IMAGE BAND TO PROCESS
* WORK INT WORK ARRAY TO HOLD INPUT AND OUTPUT
* BUFFERS
* WRKSIZ INT SIZE OF WORK ARRAY
* SELECT INT SELECTION NUMBER FOR DPCM TECHNIQUE
* FRAC REAL FRACTION OF STANDARD DEVIATION THAT
* RANGE OF DITHER SHOULD BE
* BTRATE INT BIT RATE USED ON THE DPCM
* IEV INT INTEGER EVENT VARIABLE
* ALTRET INT ALTERNATE ERROR RETURN

* INCLUDE FILES/COMMONS

* MACA1 INCLUDE MACRO FILE FOR TOKEN DEFINITION

* PROGRAM ENVIRONMENT

* THE INPUT IMAGE SHOULD BE PREPROCESSED IN THE FOLLO-
* WING MANNER BEFORE BEING USED IN THIS PROGRAM:

- * (1) MOVE TOP LINE TO THE BOTTOM AND LEFT COLUMN TO
* THE RIGHT. THIS CAN BE DONE BY USING THE 'FLPEGS'
* COMMAND IN GIPSY
- * (2) BLOCK THE RESULTING IMAGE IN RECTANGULAR BLOCKS
* USING THE 'BLOCK' COMMAND IN GIPSY

*
* THE OUTPUT IMAGE SHOULD BE POSTPROCESSED IN THE
* FOLLOWING MANNER AFTER USING THIS PROGRAM:
*

- * (1) PUT THE IMAGE IN LINE FORMAT USING THE 'BLOCK'
* COMMAND IN GIPSY
- * (2) DELETE THE BOTTOM LINE AND THE RIGHTEST CO-
* LUMN IN THE IMAGE. ADD A NEW LINE AT THE TOP
* USING THE TOP LINE OF THE ORIGINAL IMAGE. ADD
* ALSO A NEW COLUMN AT THE LEFT USING THE LEFT
* COLUMN OF THE ORIGINAL IMAGE. THIS CAN BE DONE
* USING THE COMMAND 'RSTEGS' IN GIPSY.

*
* **ALGORITHM**
*

* THE FOLLOWING STEPS ARE FOLLOWED:
*

- * (1) READ A BLOCK FROM THE INPUT IMAGE
- * (2) READ A BLOCK FROM LOW PASS FILTERED IMAGE
- * (3) COMPUTE THE MEAN AND VARIANCE OF THE DIFFERENCE
* BLOCK.
- * (4) CREATE A QUANTIZATION TABLE USING A MAX QUANTIZER
* BASED ON A GAUSSIAN DISTRIBUTION BY USING THE MEAN
* AND VARIANCE COMPUTED IN STEP (3).
- * (5) PERFORM THE DPCM TECHNIQUE SELECTED BY THE USER ON
* THE INPUT BLOCK USING THE QUANTIZATION TABLE CREA-
* TED ON STEP (4).
- * (6) WRITE THE OUTPUT BLOCK ON THE OUTPUT IMAGE

* (NOTE): THE TOP LINE AND THE LEFTHOST COLUMN OF THE
* INPUT IMAGE ARE USED TO INITIALIZE THE DPCM PROCEDURE.
* SUBSEQUENTLY THE TOP AND LEFT NEIGHBORING EDGES FROM
* PREVIOUS BLOCKS ARE USED TO INITIALIZE THE DPCM IN THE
* CURRENT BLOCK
*

* **ROUTINES CALLED**
*

* PPOSH PUSHES PROGRAM NAME INTO ERROR STACK (PRIMITIVE)
* PPOP POPS PROGRAM NAME OUT OF ERROR STACK (PRIMITIVE)
* RDKINL INITIALIZES AND ACCESSES AN SIF FILE (PRIMITIVE)
* CLOSE CLOSES AN SIF FILE (PRIMITIVE)
* CPYIDR GET DESCRIPTOR RECORDS FROM INPUT FILE (PRIMITIVE)
* DSCNAM WRITE THE NAME DESCRIPTOR RECORD (PRIMITIVE)
* PDSCI WRITE INTEGER DESCRIPTOR RECORD (PRIMITIVE)
* PDSCR WRITE REAL DESCRIPTOR RECORD (PRIMITIVE)
* COPYDS COPY DESCRIPTOR RECORDS TO OUTPUT FILE (PRIMITIVE)
* RREAD READS FROM AN SIF FILE (PRIMITIVE)
* RWRITE WRITES TO AN SIF FILE (PRIMITIVE)
* FIXBLK FIXES BOTTOM AND/OR RIGHT EDGES OF THE (GIPSY)
* BOTTOM OR RIGHTEST BLOCKS OF THE
* IMAGE
* GETEGS GETS THE NEIGHBORING EDGES OF THE (GIPSY)
* BLOCKS AT THE TOP AND AT THE LEFT OF
* THE CURRENT BLOCK IN THE RECONSTRUCTED

```

*
*      IMAGE
*      UPDEGS  UPDATES THE BUFFER WHICH STORES ALL      (GIPSY)
*              TOP EDGES IN A ROW OF BLOCKS AND
*              ALL LEFT EDGES IN A COLUMN OF BLOCKS
*      HVARDL  COMPUTES MEAN AND VARIANCE OF THE DI-    (GIPSY)
*              FERENCE BETWEEN TWO BLOCKS
*      BKDPCH  CREATES QUANTIZATION TABLE AND PERFOR- (GIPSY)
*              MS DPCH ON INPUT BLOCKS
*
*
*

```

```

*****

```

```

*
*      SUBROUTINE FDPCHC ( FDI, FDI2, PDO, BND, WORK, WRKSIZ,
*                          SELECT, FRAC, BTRATE, IEV, * )
*

```

```

*
*      INCLUDE MACA1
*      IMPLICIT INTEGER ( A - Z )
*      CHARACTER FDI ( .FDLENGTH ), PDO ( .FDLENGTH )
*      CHARACTER FDI2 ( .FDLENGTH )
*

```

```

*
*      INTEGER IDENT ( 20 ), JDENT ( 20 )
*      INTEGER IDNT2 ( 20 )
*      INTEGER WORK ( WRKSIZ ), Z ( 33 )
*      REAL FRAC, ERROR, MEAN, VAR
*

```

```

*
*      EQUIVALENCE (NBITS,IDENT(5)), (NWDS,IDENT(12))
*      EQUIVALENCE (NPPL,IDENT(6)), (NLIN,IDENT(7))
*      EQUIVALENCE (MODE,IDENT(19))
*      EQUIVALENCE (NCOLS,IDENT(13)), (NROWS,IDENT(14))
*      EQUIVALENCE (NPPL2,IDNT2(6)), (NLIN2,IDNT2(7))
*      EQUIVALENCE (NCOL2,IDNT2(13)), (NROW2,IDNT2(14))
*

```

```

*
*      CALL PPUSH ( 'FDPCHC' )
*

```

```

*
*      IST = 359140257
*

```

```

*
*              OPEN INPUT FILE
*

```

```

*
*      CALL RDKINL ( FDI, IDENT, .OLD, IEV, %9998 )
*      CALL CLOSE ( FDI )
*      CALL RDKINL ( FDI2, IDNT2, .OLD, IEV, %9998 )
*      CALL CLOSE ( FDI2 )
*

```

```

*
*              CHECK IF SIZE OF INPUT FILES
*              IS THE SAME
*

```

```

*
*      IF ( NPPL $\neq$ NPPL2 | NLIN $\neq$ NLIN2 ) GO TO 9010
*      IF ( NCOLS $\neq$ NCOL2 | NROWS $\neq$ NROW2 ) GO TO 9020
*

```

```

*
*      DO I = 1, 20
*          JDENT(I) = 0
*

```

```

JDENT(5) = NBITS
JDENT(6) = NPPL
JDENT(7) = NLIN
JDENT(13) = NCOLS
JDENT(14) = NROWS
JDENT(17) = 1
JDENT(20) = 1

```

```

COPY DESCRIPTOR RECORDS FROM INPUT IMAGES,
ROUTINE NAME AND PARAMETERS TO TEMPORARY
SEQUENTIAL FILE

```

```

CALL CPYIDR ( FDI, IDENT, .OPWTHP, IEV, X9998 )
CALL CPYIDR ( FDI2, IDNT2, .NOOPWTHP, IEV, X9998 )

```

```

CALL DSCNAM ( 'PDPCHC', IEV, X9998 )

```

```

CALL PDSCI ( 'INPUT IMAGE BAND NUMBER.', BND, IEV, X9998 )
CALL PDSCI ( 'DPCM PREDICTOR (SELECT).', SELECT, IEV, X9998 )
CALL PDSCI ( 'BIT RATE', BTRATE, IEV, X9998 )
CALL PDSCB ( 'DITHER FRACTION', PRAC, IEV, X9998 )

```

```

OPEN OUTPUT IMAGE AND COPY TO IT THE
DESCRIPTOR RECORDS FROM TEMPORARY FILE

```

```

CALL COPYDS ( FDO, JDENT, IEV, X9998 )

```

```

ACTUAL NUMBER CRUNCHING

```

```

NBPC = ICEIL ( NLIN, NROWS )
NBPR = ICEIL ( NPPL, NCOLS )
NBLCKS = NBPC * NBPR
BLKSIZE = NROWS * NCOLS
ZLEN = NCOLS + NROWS + 1
PT1 = 1
PT2 = BLKSIZE + PT1
PT3 = NPPL + PT2
PT4 = NLIN + PT3
PT5 = BLKSIZE + PT4

```

```

GET TOP AND LEFT EDGES IN THE
ORIGINAL IMAGE

```

```

DO I = 1, NBPC
$(
  BLKNO = ( NBPR - 1 ) * NBPC + I
  CALL RREAD ( FDI, WORK(PT1), BND, BLKNO, IDENT,
              .WAIT, IEV, X9999 )
  DO K = 1, NROWS
    WORK(NROWS*(I-1)+PT3+K-1) = WORK(NROWS*(NCOLS-1)+K)
  $)
DO J = 1, NBPR
$(

```

```

BLKNO = J * NBPC
CALL RREAD ( FDI, WORK(PT1), BND, BLKNO, IDENT,
             .WAIT, IEV, %9999 )
DO K = 1, NCOLS
    WORK(NCOLS*(J-1)+PT2+K-1) = WORK(NROWS*K)
$)
CORNR = WORK(PT2+NPPL-1)

```

```

CORRECT EDGES

```

```

WORK(PT2+NPPL-1) = WORK(PT2+NPPL-2)
WORK(PT3+NLIN-1) = WORK(PT3+NLIN-2)

```

```

NOW READ EACH BLOCK IN THE IMAGE
IN A RASTER SCAN MODE AND PROCESS

```

```

DO I = 1, NBPC
$(
    NITCNR = WORK ( NROWS*I + PT3 - 1 )
    DO J = 1, NBPR
    $(
        BLKNO = NBPC * ( J - 1 ) + I
        CALL RREAD ( FDI, WORK(PT1), BND, BLKNO, IDENT,
                   .WAIT, IEV, %9999 )
        CALL FIXBLK ( WORK(PT1), I, J, NBPC, NBPR,
                    NROWS, NCOLS )
        CALL RREAD ( FDI2, WORK(PT5), BND, BLKNO, IDENT2,
                   .WAIT, IEV, %9999 )
        CALL HVARDL ( WORK(PT1), WORK(PT5), MEAN, VAR,
                    BLKSZE )
        CALL GETEGS ( WORK(PT2), WORK(PT3),
                    CORNR, NPPL, NLIN, I, J,
                    NROWS, NCOLS, Z, ZLEN )
        CALL BKDPCM ( WORK(PT1), WORK(PT4), WORK(PT5),
                    Z, ZLEN, MEAN, VAR, NROWS, NCOLS,
                    BTRATE, FRAC, IST, SELECT, ERROR )
        CALL RWRITE ( FDO, WORK(PT4), BND, BLKNO, IDENT,
                    .WAIT, IEV, %9999 )
        CALL UPDEGS ( WORK(PT4), WORK(PT2), WORK(PT3),
                    CORNR, NPPL, NLIN, I, J,
                    NROWS, NCOLS )
    $)
    CORNR = NITCNR
$)

```

```

8000 CONTINUE

```

```

CALL CLOSE ( FDI )
CALL CLOSE ( FDI2 )
CALL CLOSE ( FDO )
CALL PPOP
RETURN

```

```

ABNORMAL CONDITIONS

```

```

*
9010 CONTINUE
*
* INPUT FILES ARE NOT THE
* SAME SIZE
*
IEV = -5021
GO TO 9999
*
9020 CONTINUE
*
* INPUT FILES NOT THE SAME BLOCK
* SIZE
*
IEV = -5023
GO TO 9999
*
9998 CONTINUE
*
* READ OR WRITE ERROR
*
CALL CLOSE ( FDI )
CALL CLOSE ( FDI2 )
CALL CLOSE ( FDO )
9999 RETURN 1
*
END

```

*--DPCNCD DPCM DATA COMPRESSION DRIVER

HID

* IDENTIFICATION

* TITLE DPCNCD
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE JULY 16, 1979
* LANGUAGE RATFOR
* SYSTEM IBH-370

* PURPOSE

* THIS IS THE DRIVER FOR THE DPCM DATA COMPRESSION PACKAGE.

* ENTRY POINT

* DPCNCD (WORK, ALTRET)

* ARGUMENT LISTING

* WORK INT WORK ARRAY TO HOLD INPUT AND OUTPUT
* BUFFERS
* ALTRET INT ALTERNATE ERROR RETURN

* INCLUDE FILES/COMMONS

* HACA1 INCLUDE MACRO FILE FOR TOKEN DEFINITION
* GIPCOM INCLUDE INCLUDE FILE FOR GIPSY
* ERROR INCLUDE INCLUDE FILE FOR COMMON ERROR

* ROUTINES CALLED

* PPUSH PUSHES PROGRAM NAME INTO ERROR STACK (PRIMITIVE)
* PPOP POPS PROGRAM NAME FROM ERROR STACK (PRIMITIVE)
* RDKINL INITIALIZES AND ACCESSES AN SIF FILE (PRIMITIVE)
* CLOSE CLOSES AN SIF FILE (PRIMITIVE)
* CTRLT RETURNS ADDRESS OF CONTROL T (PRIMITIVE)
* IGNORT IGNORES CONTROL T (PRIMITIVE)
* OSALOC ALLOCATE DIN WORDS FOR DYNAMIC ARRAY (PRIMITIVE)
* DPCHCI ASKS USER FOR INPUT PARAMETERS (GIPSY)
* DPCMDC PERFORMS A NON-CAUSAL OR CAUSAL BIT (GIPSY)
* ALLOCATION DPCM ON AN INPUT IMAGE

* SUBROUTINE DPCNCD (WORK, *)

* INCLUDE HACA1
* IMPLICIT INTEGER (A - Z)

* INCLUDE GIPCOM


```
INCLUDE ERROR
INTEGER WORK ( .ARB )
INTEGER IDENT ( 20 ), IDNT2 ( 20 )
REAL FRAC, C, ALPHA, GAMMA
LOGICAL FLAG
```

```
EQUIVALENCE (NPPL,IDENT(6)), (NLINS,IDENT(7))
EQUIVALENCE (NCCOLS,IDENT(13)), (NROWS,IDENT(14))
EQUIVALENCE (NTBND,IDENT(17)), (NSBND,IDENT(18))
EQUIVALENCE (MODE,IDENT(19))
EQUIVALENCE (NTBND2,IDNT2(17)), (NSBND2,IDNT2(18))
EQUIVALENCE (MODE2,IDNT2(19))
```

```
CALL PPUSH ( 'DPCMCD' )
```

```
SET UP INPUT FILE
```

```
CALL RDKINL ( FDI1, IDENT, .OLD, IEV, %9999 )
CALL RDKINL ( FDI2, IDNT2, .OLD, IEV, %9999 )
```

```
CALL CLOSE ( FDI1 )
CALL CLOSE ( FDI2 )
```

```
CHECK INPUT FILE
```

```
IF ( MODE /= 1 & MODE /= 0 ) GO TO 9000
IF ( NTBND - NSBND <= 0 ) GO TO 9010
IF ( MODE2 /= 1 & MODE2 /= 0 ) GO TO 9000
IF ( NTBND2 - NSBND2 <= 0 ) GO TO 9010
```

```
GET USER INFORMATION
```

```
CALL DPCMCI ( FDI1, SELECT, FRAC, MXNBTS, MINBTS, C,
ALPHA, BUFSIZE, GAMMA, FLAG, %9999 )
```

```
CHECK AVAILABLE SPACE
```

```
BLKSIZE = NROWS * NCOLS
NBLCKS = ICEIL(NLINS,NROWS)*ICEIL(NPPL,NCOLS)
WRKSZ = 3*BLKSIZE + NPPL + NLINS + NBLCKS
IF ( .OK /= OSALOC ( WRKSZ ) ) GO TO 9020
```

```
GET THE BAND TO USE
```

```
BND = 1
```

```
SET UP INTERRUPT CONTROL T
```

```
CALL CTBLT ( %8000 )
```

```
CALL THE NUMBER CRUNCHER
```

```
CALL DPCMDC ( FDI1, FDI2, FDO1, BND, WORK, WRKSZ,  
SELECT, FRAC, C, ALPHA, GAMMA, BUFSZE,  
MXNBTS, MINBTS, FLAG, IEV, %9998 )
```

```
*  
CALL PPOP  
RETURN
```

```
*  
8000 CONTINUE
```

```
*  
CONTROL T EXIT
```

```
*  
CALL IGNORT  
CALL CLOSE ( FDI1 )  
CALL CLOSE ( FDI2 )  
CALL CLOSE ( FDO1 )  
RETURN
```

```
*  
ABNORMAL CONDITIONS
```

```
*  
9000 CONTINUE
```

```
*  
NOT AN INTEGER FILE
```

```
*  
IEV = -2012  
GO TO 9998
```

```
*  
9010 CONTINUE
```

```
*  
NO NUMERIC BANDS
```

```
*  
IEV = -5018  
GO TO 9998
```

```
*  
9020 CONTINUE
```

```
*  
NOT ENOUGH WORK SPACE
```

```
*  
IEV = -5010
```

```
*  
9998 CONTINUE
```

```
*  
ERROR IN SUBPROGRAM
```

```
*  
CALL CLOSE ( FDI1 )  
CALL CLOSE ( FDI2 )  
CALL CLOSE ( FDC1 )
```

```
*  
ABNORMAL RETURN
```

```
*  
9999 RETURN 1  
END
```

*--DPCMCI

DPCM DATA COMPRESSION USER INPUT

MID

* IDENTIFICATION

* TITLE DPCMCI
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE JULY 10, 1979
* LANGUAGE RATFOR
* SYSTEM IBM-370

* PURPOSE

* THIS ROUTINE ASKS THE USER THE INPUT PARAMETERS NEEDED
* TO PERFORM A CAUSAL OR A NON-CAUSAL DPCM COMPRESSION
* ON AN IMAGE.

* ENTRY POINT

* DPCMCI (FD, SELECT, FRAC, MXNBTS, MINBTS, C, ALPHA,
* BUFSZE, GAMMA, FLAG, ERRET)

* ARGUMENT LISTING

* FD CHRARRAY FILE DESCRIPTOR
* SELECT INT SELECTION NUMBER FOR DPCM TECHNIQUE
* FRAC REAL FRACTION OF STANDARD DEVIATION THAT
* RANGE OF DPCM DITHER SHOULD BE
* MINBTS INT THE ALLOWED MAXIMUM NUMBER OF BITS PER
* PIXEL FOR ANY BLOCK
* MINBTS INT THE ALLOWED MINIMUM NUMBER OF BITS PER
* PIXEL FOR ANY BLOCK
* C REAL THE CHANNEL CAPACITY IN BITS PER PIXEL
* ALPHA REAL THE DISTORTION MODEL CCNSTANT
* BUFSZE INT THE SIZE OF THE BIT BUFFER
* GAMMA REAL CONSTANT USED TO UPDATE THE EXPECTED
* AVERAGE VARIANCE IN CAUSAL BIT ALLOCA-
* TION.
* FLAG LOG TRUE FOR CAUSAL BIT ALLCCATION AND
* FALSE OTHERWISE
* ERRET ALTRET ALTERNATE RETURN

* INCLUDE FILES/COMMONS

* MACA1 INCLUDE MACRO FILE FOR TOKEN DEFINITION
* GIPCOM INCLUDE INCLUDE FILE FOR GIPSY
* ERROR INCLUDE INCLUDE FILE FOR COMMON ERFOR
* TTCOM INCLUDE INCLUDE FILE FOR TERMINAL & RUNFILE I/O

* ROUTINES CALLED

* PPUSH PUSH PROGRAM NAME ONTO ERROR STACK (PRIMITIVE)
* OSGTMM GET NAME STRING FROM FILE DESCRIPTOR (PRIMITIVE)
* GETI GET INTEGER INPUT (PRIMITIVE)

```

*      GETR      GET REAL INPUT                      (PRIMITIVE)
*      PPOP      POP PROGRAM NAME OFF ERROR STACK    (PRIMITIVE)
*
*
*****
*
SUBROUTINE DPCMCI ( FD, SELECT, FRAC, HXNBTS, MINBTS, C,
                  ALPHA, BUFSZE, GAMMA, FLAG, * )
*
INCLUDE MACA1
INPLICIT INTEGER ( A - Z )
*
INCLUDE GIPCOM
INCLUDE ERROR
INCLUDE TTCOM
*
CHARACTER FD ( .FDLENGTH )
CHARACTER FDIO ( .FILENAMELENGTH )
CHARACTER ALT, BELLS
LOGICAL FLAG
REAL FRAC, C, ALPHA, GAMMA
*
DATA ALT, BELLS / .ALTCHR, .BELLCHR /
*
CALL PPOP ( 'DPCMCI' )
*
*           GET NAME INTO SYSTEM STANDARD FORMAT
*
IF ( .OK == OSGTMM ( PD, FDIO ) ) GO TO 9999
*
WRITE (RUNOT,6000) FDIO, BELLS, ALT
6000  FORMAT ( ' SELECT DPCM PREDICTOR FOR ', .FILENAMELENGTH A1, //
            ' (2) 2-D DPCM   FLAT MODEL   LSE'//,
            ' (3) MOD DPCM   FLAT MODEL   LSE'//,
            ' (4) 2-D DPCM   FLAT MODEL   MVE'//,
            ' (5) MOD DPCM   FLAT MODEL   MVE'//,
            ' (6) 2-D DPCM   SLOPED MODEL  LSE'//,
            ' (7) MOD DPCM   SLOPED MODEL  LSE'//,
            ' (8) 2-D DPCM   SLOPED MODEL  MVE'//,
            ' (9) MOD DPCM   SLOPED MODEL  MVE'//,
            2A1 )
IF ( .OK == GETI ( FDRUNI, SELECT ) ) GO TO 9999
*
*           CHECK FOR LEGAL SELECTION NUMBER
*
WHILE ( SELECT < 2 | SELECT > 9 )
$(
  WRITE (TTYOT,6000) FDIO, BELLS, ALT
  IF ( .OK == GETI ( FDTTYI, SELECT ) ) GO TO 9999
$)
*
WRITE (RUNOT,6010) BELLS, ALT
6010  FORMAT ( ' ENTER FRACTION OF STANDARD DEVIATION'//,

```

```

          ' THAT RANGE OF DITHER SHOULD BE --', 2A1 )
IF ( .OK == GETR ( FDRUNI, FRAC ) ) GO TO 9999
*
6020  WRITE (RUNOT,6020) BELLS, ALT
      FORMAT ( ' ENTER MAX BITS --', 2A1 )
      IF ( .OK == GETI ( FDRUNI, MXNBTS ) ) GO TO 9999
*
6025  WRITE (RUNOT,6025) BELLS, ALT
      FORMAT ( ' ENTER MIN BITS --', 2A1 )
      IF ( .OK == GETI ( FDRUNI, MINBTS ) ) GO TO 9999
*
6030  WRITE (RUNOT, 6030) BELLS, ALT
      FORMAT ( ' ENTER CHANNEL CAPACITY--', 2A1 )
      IF ( .OK == GETR ( FDRUNI, C ) ) GO TO 9999
*
6040  WRITE (RUNOT,6040) BELLS, ALT
      FORMAT ( ' ENTER DISTORTION MODEL CONSTANT --',
              2A1 )
      IF ( .OK == GETR ( FDRUNI, ALPHA ) ) GO TO 9999
*
6050  WRITE (RUNOT,6050) BELLS, ALT
      FORMAT ( ' ENTER BUFFER SIZE --', 2A1 )
      IF ( .OK == GETI ( FDRUNI, BUFSIZE ) ) GO TO 9999
*
6060  WRITE (RUNOT,6060) BELLS, ALT
      FORMAT ( ' NON-CAUSAL (0)? OR CAUSAL (1)?--', 2A1 )
      IF ( .OK == GETI ( FDRUNI, KK ) ) GO TO 9999
      IF ( KK == 1 )
      $(
        FLAG = .TRUE.
        6070  WRITE (RUNOT,6070) BELLS, ALT
              FORMAT ( ' ENTER UPDATING CONSTANT --', 2A1 )
              IF ( .OK == GETR ( FDRUNI, GAMMA ) ) GO TO 9999
      $)
      ELSE  FLAG = .FALSE.
*
      CALL PPOP
      RETURN
*
9999  CONTINUE
*
      RETURN 1
      END

```

*--DPCMDC NON-CAUSAL AND CAUSAL DPCM DATA COMPRESSION MID

* IDENTIFICATION

* TITLE DPCMDC
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE JULY 16, 1979
* LANGUAGE RATFOR
* SYSTEM IBM-370

* PURPOSE

* THIS ROUTINE READS BLOCKS FROM AN INPUT IMAGE IN A SCAN
* RASTER MODE AND PERFORMS A NON-CAUSAL OR CAUSAL BIT ALL-
* OCATION DPCM ON THOSE BLOCKS USING A NUMBER OF DPCM TEC-
* HNIQUES AS SELECTED BY THE USER.

* ENTRY POINT

* DPCMDC (FDI, FDI2, FDO, BND, WORK, WRKSIZ, SELECT,
* FRAC, C, ALPHA, GAMMA, BUFSIZE, MXNBTS,
* MINBTS, FLAG, IEV, ALTRET)

* ARGUMENT LISTING

* FDI INT FILE DESCRIPTOR FOR INPUT IMAGE
* FDI2 INT FILE DESCRIPTOR FOR LOW PASS FILTERED
* IMAGE
* FDO INT FILE DESCRIPTOR FOR OUTPUT IMAGE
* BND INT IMAGE BAND TO PROCESS
* WORK INT WORK ARRAY TO HOLD INPUT AND OUTPUT
* BUFFERS
* WRKSIZ INT SIZE OF WORK ARRAY
* SELECT INT SELECTION NUMBER FOR DPCM TECHNIQUE
* FRAC REAL FRACTION OF STANDARD DEVIATION THAT
* RANGE OF DITHER SHOULD BE
* C REAL THE CHANNEL CAPACITY IN BITS PER PIXEL
* ALPHA REAL DISTORTION MODEL CONSTANT
* GAMMA REAL CONSTANT USED TO UPDATE THE EXPECTED
* AVERAGE VARIANCE IN CAUSAL BIT ALLOCA-
* TION DPCM
* BUFSIZE INT THE SIZE OF THE BIT BUFFER
* MXNBTS INT THE ALLOWED MAXIMUM NUMBER OF BITS PER
* PIXEL FOR ANY BLOCK
* MINBTS INT THE ALLOWED MINIMUM NUMBER OF BITS PER
* PIXEL FOR ANY BLOCK
* FLAG LOG TRUE FOR CAUSAL BIT ALLOCATION AND FALSE
* OTHERWISE.
* IEV INT INTEGER EVENT VARIABLE
* ALTRET INT ALTERNATE ERROR RETURN

* INCLUDE FILES/COMMONS

```

*      MACA1   INCLUDE      MACRO FILE FOR TOKEN DEFINITION
*
* PROGRAM ENVIRONMENT
*
* THE INPUT IMAGE SHOULD BE PREPROCESSED IN THE FOLLOW-
* ING MANNER BEFORE BEING USED IN THIS PROGRAM:
*
* (1) MOVE TOP LINE TO THE BOTTOM AND LEFT COLUMN TO THE
*     RIGHT. THIS CAN BE DONE BY USING THE 'FLPEGS' COM-
*     MAND IN GIPSY.
* (2) BLOCK THE RESULTING IMAGE IN RECTANGULAR BLOCKS USING
*     THE 'BLOCK' COMMAND IN GIPSY.
*
* THE OUTPUT IMAGE SHOULD BE POSTPROCESSED IN THE FOLLOWING
* MANNER AFTER USING THIS PROGRAM:
*
* (1) PUT THE IMAGE IN LINE FORMAT USING THE 'BLOCK' COMMAND
*     IN GIPSY.
* (2) DELETE THE BOTTOM LINE AND THE RIGHTMOST COLUMN IN THE
*     IMAGE. ADD A NEW LINE AT THE TOP USING THE TOP LINE OF
*     THE ORIGINAL IMAGE. ADD ALSO A NEW COLUMN AT THE LEFT
*     USING THE LEFT COLUMN OF THE ORIGINAL IMAGE. THIS CAN BE
*     DONE BY USING THE COMMAND 'RSTEGS' IN GIPSY.
*
* ALGORITHM
*
* REPER TO CHAPTER EIGHT IN THE PROJECT REPORT.
*
* ROUTINES CALLED
*
* PPUSH   PUSHES PROGRAM NAME INTO ERROR STACK      (PRIMITIVE)
* PPOP    POPS PROGRAM NAME FROM ERROR STACK        (PRIMITIVE)
* RDKINL  INITIALIZES AND ACCESSES AN SIF FILE      (PRIMITIVE)
* CLOSE   CLOSSES AN SIF FILE                       (PRIMITIVE)
* CPYIDR  GET DESCRIPTOR RECORDS FROM INPUT FILE    (PRIMITIVE)
* DSCNAM  WRITE THE NAME DESCRIPTOR RECORD          (PRIMITIVE)
* PDSCI   WRITE INTEGER DESCRIPTOR RECORD           (PRIMITIVE)
* PDSCR   WRITE REAL DESCRIPTOR RECORD              (PRIMITIVE)
* PDSCPS  WRITE PACKED STRING DESCRIPTOR RECORD    (PRIMITIVE)
* COPYDS  COPY DESCRIPTOR RECORDS TO OUTPUT FILE   (PRIMITIVE)
* RREAD   READS FROM AN SIF FILE                   (PRIMITIVE)
* RWRITE  WRITES TO AN SIF FILE                   (PRIMITIVE)
* FIXBLK  FIXES BCTTOM AND/OR RIGHT EDGES OF THE   (GIPSY)
*         BOTTOM OR RIGHTMOST BLOCKS OF THE IN-
*         PUT IMAGE.
*
* GETEGS  GETS THE NEIGHBORING EDGES OF THE BLOC-   (GIPSY)
*         KS AT THE TOP AND AT THE LEFT OF THE
*         CURRENT BLOCK IN THE RECONSTRUCTED I-
*         MAGE
*
* UPDEGS  UPDATES THE BUFFERS WHICH STORE ALL      (GIPSY)
*         TOP EDGES IN A ROW OF BLOCKS AND ALL
*         LEFT EDGES IN A COLUMN OF BLOCKS
*
* BKDPCH  CREATES QUANTIZATION TABLE AND PERFO-   (GIPSY)
*         RMS DPCM ON INPUT BLOCKS

```



```

JDENT(5) = NBITS
JDENT(6) = NPPL
JDENT(7) = NLIN
JDENT(13) = NCOLS
JDENT(14) = NROWS
JDENT(17) = 1
JDENT(20) = 1

```

```

COPY DESCRIPTOR RECORDS FROM INPUT IMAGES,
ROUTINE NAME AND PARAMETERS TO TEMPORARY
SEQUENTIAL FILE

```

```

CALL CPYIDR ( FDI, IDENT, .OPNTMP, IEV, %9998 )
CALL CPYIDR ( FDI2, IDNT2, .NOOPNTMP, IEV, %9998 )

```

```

CALL DSCNAM ( 'DPCMDC', IEV, %9998 )

```

```

CALL PDSCI ( 'INPUT IMAGE BAND NUMBER.', BND, IEV, %9998 )
CALL PDSCI ( 'DPCM PREDICTOR (SELECT).', SELECT, IEV, %9998 )
CALL PDSCI ( 'BIT BUFFER SIZE .', BUFSIZE, IEV, %9998 )
CALL PDSCI ( 'ALLOWED MAX #BITS/PEL .', MXNBTS, IEV, %9998 )
CALL PDSCI ( 'ALLOWED MIN #BITS/PEL .', MINBTS, IEV, %9998 )
CALL PDSCR ( 'DITHER FRACTION .', FRAC, IEV, %9998 )
CALL PDSCR ( 'CHANNEL CAPACITY .', C, IEV, %9998 )
CALL PDSCR ( 'DISTORTION MODEL CONST .', ALPHA, IEV, %9998 )

```

```

IF ( FLAG )

```

```

$(
CALL PDSCPS ( 'CAUSAL BIT ALLOCATION .', IEV, %9998 )
CALL PDSCR ( 'UPDATING CONSTANT .', GAMMA, IEV, %9998 )
$)

```

```

ELSE

```

```

CALL PDSCPS ( 'NON-CAUSAL BIT ALLOCAT.', IEV, %9998 )

```

```

OPEN OUTPUT IMAGE AND COPY TO IT THE
DESCRIPTOR RECORDS FROM TEMPORARY FILE

```

```

CALL COPYDS ( FDO, JDENT, IEV, %9998 )

```

```

ACTUAL NUMBER CRUNCHING

```

```

NBPC = ICEIL ( NLIN, NROWS )
NBPR = ICEIL ( NPPL, NCOLS )
NBLCKS = NBPC * NBPR
BLKSIZE = NROWS * NCOLS
ZLEN = NCOLS + NROWS + 1
PT1 = 1
PT2 = BLKSIZE + PT1
PT3 = NPPL + PT2
PT4 = NLIN + PT3
PT5 = BLKSIZE + PT4
PT6 = NBLCKS + PT5

```



```
*
      IEV = -5021
      GO TO 9999
*
9020  CONTINUE
*
*                                     INPUT FILES NOT THE SAME BLOCK
*                                     SIZE
*
      IEV = -5023
      GO TO 9999
*
*
9998  CONTINUE
*
*                                     READ OR WRITE ERROR
*
      CALL CLOSE ( FDI )
      CALL CLOSE ( FDI2 )
      CALL CLOSE ( FDO )
9999  RETURN 1
*
      END
```

*--MVARDL MEAN AND VARIANCE OF A LINE OF DIFFERENCES MID

* IDENTIFICATION

* TITLE MVARDL
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE AUGUST 31, 1978
* LANGUAGE RATFOR
* SYSTEM IBM-370

* PURPOSE

* THIS ROUTINE COMPUTES THE MEAN AND VARIANCE OF THE
* DIFFERENCE OF TWO LINES OF INTEGER DATA.

* ENTRY POINT

* MVARDL (LINE1, LINE2, DMEAN, DVAR, NUMPPL)

* ARGUMENT LISTING

* LINE1 INT FIRST LINE OF DATA
* LINE2 INT SECOND LINE OF DATA
* DMEAN REAL MEAN OF THE DIFFERENCE LINE
* DVAR REAL VARIANCE OF THE DIFFERENCE LINE
* NUMPPL INT NUMBER OF POINTS PER LINE

* ROUTINES CALLED

* NONE

* SUBROUTINE MVARDL (LINE1, LINE2, DMEAN, DVAR, NUMPPL)

* INTEGER LINE1(NUMPPL), LINE2(NUMPPL)
* REAL DMEAN, DVAR, SUM, SUM2

* SUM = 0.
* SUM2 = 0.

* DO I = 1, NUMPPL
* SUM = SUM + LINE1(I) - LINE2(I)
* DMEAN = SUM/NUMPPL

* DO I = 1, NUMPPL
* SUM2 = SUM2 + (LINE1(I) - LINE2(I) - DMEAN)**2
* DVAR = SUM2 / NUMPPL

* RETURN
* END

* IDENTIFICATION

* TITLE CBITAL
 * AUTHOR OSCAR A. ZUNIGA
 * VERSION A.01
 * DATE JULY 21, 1979
 * LANGUAGE RATFOR
 * SYSTEM IBH-370

* PURPOSE

* THIS ROUTINE PERFORMS A CAUSAL BIT ALLOCATION ON
 * THE BLOCKS OF AN IMAGE

* ENTRY POINT

* CBITAL (VARBUF, OBITAB, NBLCKS, BLKSZE, ALPHA,
 * GAMMA, C, MINBTS, MINBTS, BUFSIZE,
 * NBPC, NBPR)

* ARGUMENT LISTING

* VARBUF	REAL	1-D ARRAY CONTAINING THE VARIANCES OF
* OBITAB	INT	THE DPCM CORRECTIONS FOR EACH BLOCK
* NBLCKS	INT	1-D OUTPUT ARRAY CONTAINING THE BIT
* BLKSZE	INT	ALLOCATIONS FOR EACH BLOCK
* ALPHA	REAL	THE NUMBER OF BLOCKS IN THE IMAGE
* GAMMA	REAL	THE SIZE OF EACH BLOCK
		DISTORTION MODEL CONSTANT
		CONSTANT PARAMETER FOR UPDATING THE
		ESTIMATION OF THE AVERAGE OF
		THE FUTURE VARIANCE.
* C	REAL	THE CHANNEL CAPACITY IN BITS PER PIXEL
* MINBTS	INT	THE ALLOWED MAXIMUM NUMBER OF BITS PER
		PIXEL FOR ANY BLOCK
* MINBTS	INT	THE ALLOWED MINIMUM NUMBER OF BITS PER
		PIXEL FOR ANY BLOCK
* BUFSIZE	INT	THE SIZE OF THE BIT BUFFER
* NBPC	INT	NUMBER OF BLOCKS PER COLUMN
* NBPR	INT	NUMBER OF BLOCKS PER ROW

* INCLUDE FILES/COMMONS

* MACA1 INCLUDE MACRO FILE FOR TOKEN DEFINITION

* ALGORITHM

* THE ALGORITHM USED IN THIS ROUTINE IS BASED ON RATE DIS-
 * TORTION THEORY ASSUMPTIONS FOR GAUSSIAN DATA. UNDER THIS
 * ASSUMPTIONS THE RELATION BETWEEN MEAN SQUARE QUANTIZATION
 * ERROR AND BIT RATE FOR EACH BLOCK IS GIVEN BY :

```

*
*           MSE = VAR * EXP ( -ALPHA*NBITS )
*
* WHERE  MSE      MEAN SQUARE QUANTIZATION ERROR
*        VAR      VARIANCE ASSOCIATED WITH THE BLOCK
*        ALPHA    DISTORTION MODEL CONSTANT
*        NBITS    THE BIT RATE
*
* IF VAR1, VAR2,.....,VARK ARE THE VARIANCES OF THE DPCM
* CORRECTIONS FOR THE K BLOCKS IN THE IMAGE, C IS THE
* DESIRED CHANNEL CAPACITY, THEN IT CAN BE SHOWN THAT
* THE OPTIMAL BIT ALLOCATION FOR EACH BLOCK IS GIVEN BY:
*
*           NBITS(K) = C + (1/ALPHA) * ( LN(VARK) - LN(VARF) )
*
*           LN(VARF) = (1/K) *      SUM      LN(VARI)
*                        I = 1, K
*
* OBSERVATION OF THIS RELATION INDICATES THAT BIT ALLOCATION
* TO A PRESENT BLOCK DEPENDS ON THE ESTIMATION OF ONLY ONE
* FUTURE PARAMETER, IE, LN(VARF). THIS ROUTINE ESTIMATES
* INITIALLY LN(VARF) AS LN(VARI) FOR THE FIRST BLOCK AND SUB-
* SEQUENTLY UPDATES THIS ESTIMATION IN THE FOLLOWING FASHION:
*
*           LN(VARF;I+1) = (1-GAMMA)*LN(VARF;I) + GAMMA*LN(VARI)
*
* THE INDEXES I+1 AND I REFERS TO THE ESTIMATION FOR THE
* (I+1)TH AND ITH BLOCK RESPECTIVELY. VARI IS THE TRUE
* VARIANCE OF THE ITH BLOCK.
*
* FOR EXPLANATION OF BUFFER HANDLING REFER TO THE TECHNICAL
* REPORT, CHAPTER 8.
*
* ROUTINES CALLED
*
*           BITALO  BIT ALLOCATION OUTPUT              (GIPSY)
*
* *****
*
* SUBROUTINE CBITAL ( VARBUF, OBITAB, NBLCKS, BLKSZE,
*                   ALPHA, GAMMA, C, MNBTS, MINBTS,
*                   BUFSIZE, NBPC, NBPR )
*
* INCLUDE MACA1
* INTEGER OBITAB ( NBLCKS )
* INTEGER BLSLFT, BKSLEFT, BUFSIZE, BLKSZE
* INTEGER CBITS, BSTATE, BUPBTS, BLSOUT
* REAL VARBUF ( NBLCKS ), LNVARF, LNVARP
* LOGICAL FLAG
*
*           ESTIMATE AVERAGE VARIANCE AS THE
*           VARIANCE FOR THE FIRST BLOCK

```

```

VARF = VARBUF ( 1 )
LNVARF = ALOG ( VARF )
ALPHIV = 1. / ALPHA
BTSLFT = C * NBLCKS
BKSFLT = NBLCKS
IF ( C < 1. ) IBKSLF = C * BKSFLT
ELSE
    IBKSLF = BKSFLT
IF ( BTSLFT <= IBKSLF ) FLAG = .TRUE.
ELSE
    FLAG = .FALSE.

```

```

CBITS = C
IF ( C - CBITS > 0. ) CBITS = CBITS + 1
BTSOUT = IROUND ( C*BLKSIZE )

```

```

    ASSUME BUFFER INITIALLY HALF FULL

```

```

BSTATE = BUFSIZE / 2
BUFPTS = ( BUFSIZE + BTSOUT - BSTATE ) / BLKSIZE

```

```

    ALLOCATE BITS TO EACH BLOCK; UPDATE
    ESTIMATION OF AVERAGE VARIANCE AND
    PREVENT BUFFER OVERFLOW OR UNDERFLOW

```

```

DO K = 1, NBLCKS

```

```

$(
    LNVARP = ALOG ( VARBUF ( K ) )
    BTRATE = C + ALPHIV * ( LNVARP - LNVARF )
    NBITS = IROUND ( BTRATE )
    IF ( BSTATE <= BTSOUT )
    $(
        NBITS = MAXO ( NBITS, CBITS )
        NBITS = MINO ( NBITS, MXNBITS, BTSLFT )
    $)
    ELSE
    $(
        NBITS = MAXO ( NBITS, MINBTS )
        NBITS = MINO ( NBITS, MXNBITS, BTSLFT, BUFPTS )
    $)
    IF ( FLAG ) NBITS = MINO ( NBITS, 1 )
    OBITAB(K) = NBITS
    BTSLFT = BTSLFT - NBITS
    BKSFLT = BKSFLT - 1
    BSTATE = BSTATE + NBITS*BLKSIZE - BTSOUT
    BUFPTS = ( BUFSIZE + BTSOUT - BSTATE ) / BLKSIZE
    IF ( C < 1. ) IBKSLF = C * BKSFLT
    ELSE
        IBKSLF = BKSFLT
    IF ( BTSLFT <= IBKSLF ) FLAG = .TRUE.
    ELSE
        FLAG = .FALSE.
    LNVARP = (1.-GAMMA)*LNVARF + GAMMA*LNVARP

```

```

$)

```

```

IF ( BTSLFT > 0 )
    OBITAB ( NBLCKS ) = MINO ( BTSLFT, MINBTS )

```

CALL BITALO (OEITAB, NBPC, NBPR, K, BLSLFT, BKSLSFT)

RETURN
END

* IDENTIFICATION

* TITLE NBITAL
 * AUTHOR OSCAR A. ZUNIGA
 * VERSION A.01
 * DATE JULY 21, 1979
 * LANGUAGE RATFOR
 * SYSTEM IBM-370

* PURPOSE

* THIS ROUTINE PERFORMS A NON-CAUSAL BIT ALLOCATION ON
 * THE BLOCKS OF AN IMAGE

* ENTRY POINT

* NBITAL (VARBUF, OBITAB, NBLCKS, BLKSIZE, ALPHA,
 * C, MXNBTS, MINBTS, BUFSIZE, NBPC, NBR)

* ARGUMENT LISTING

* VARBUF REAL 1-D ARRAY CONTAINING THE VARIANCES OF
 * THE DPCM CORRECTIONS FOR EACH BLOCK
 * OBITAB INT 1-D OUTPUT ARRAY CONTAINING THE BIT
 * ALLOCATIONS FOR EACH BLOCK
 * NBLCKS INT THE NUMBER OF BLOCKS IN THE IMAGE
 * BLKSIZE INT THE SIZE OF EACH BLOCK
 * ALPHA REAL DISTORTION MODEL CONSTANT
 * C REAL THE CHANNEL CAPACITY IN BITS PER PIXEL
 * MXNBTS INT THE ALLOWED MAXIMUM NUMBER OF BITS PER
 * PIXEL FOR ANY BLOCK
 * MINBTS INT THE ALLOWED MINIMUM NUMBER OF BITS PER
 * PIXEL FOR ANY BLOCK
 * BUFSIZE INT THE SIZE OF THE BIT BUFFER
 * NBPC INT THE NUMBER OF BLOCKS PER COLUMN
 * NBR INT THE NUMBER OF BLOCKS PER ROW

* INCLUDE FILES/COMMONS

* MACA1 INCLUDE MACRO FILE FOR TOKEN DEFINITION

* ALGORITHM

* THE ALGORITHM USED IN THIS ROUTINE IS BASED ON RATE DIS-
 * TORTION THEORY ASSUMPTIONS FOR GAUSSIAN DATA. UNDER THIS
 * ASSUMPTIONS THE RELATION BETWEEN MEAN SQUARE QUANTIZATION
 * ERROR AND BIT RATE FOR EACH BLOCK IS GIVEN BY :

$$MSE = VAR * EXP (-ALPHA * NBITS)$$

* WHERE MSE MEAN SQUARE QUANTIZATION ERROR
 * VAR VARIANCE ASSOCIATED WITH THE BLOCK

```

*           ALPHA      DISTORTION MODEL CONSTANT
*           NBITS      THE BIT RATE

```

```

* IF VAR1, VAR2,.....,VARK ARE THE VARIANCES OF THE DPCM
* CORRECTIONS FOR THE K BLOCKS IN THE IMAGE, C IS THE
* DESIRED CHANNEL CAPACITY, THEN IT CAN BE SHOWN THAT
* THE OPTIMAL BIT ALLOCATION FOR EACH BLOCK IS GIVEN BY:

```

$$NBITS(K) = C + (1/ALPHA) * (LN(VARK) - LN(VARF))$$

$$LN(VARF) = (1/K) * \sum_{I=1, K} LN(VARI)$$

```

* FOR EXPLANATION OF BUFFER HANDLING REFER TO THE TECHNICAL
* REPORT, CHAPTER 8.

```

```

* ROUTINES CALLED

```

```

*           BITALO BIT ALLOCATION OUTPUT (GIPSY)

```

```

*****

```

```

* SUBROUTINE NBITAL ( VARBUF, OBITAB, NBLCKS, BLKSIZE,
*                   ALPHA, C, MXNBTS, MINBTS,
*                   BUFSIZE, NBPC, NBPR )

```

```

* INCLUDE MACA1
* INTEGER OBITAB(NBLCKS), BLSLFT, BKSFLT, BUFSIZE
* INTEGER CBITS, BSTATE, BUPBTS, BLSOUT, BLKSIZE
* REAL VARBUF(NBLCKS), LNVARF

```

```

*           COMPUTE THE LOG OF THE AVERAGE
*           VARIANCE

```

```

* SUM = 0.
* DO K = 1, NBLCKS
*   SUM = SUM + ALOG ( VARBUF(K) )

```

```

* LNVARF = SUM / NBLCKS
* ALPHIV = 1. / ALPHA
* BLSLFT = C * NBLCKS
* CBITS = C
* IF ( C - CBITS > 0. ) CBITS = CBITS + 1
* BLSOUT = IROUND ( C * BLKSIZE )

```

```

*           ASSUME BUFFER INITIALLY HALF FULL

```

```

* BSTATE = BUFSIZE / 2
* BUPBTS = ( BUFSIZE + BLSOUT - BSTATE ) / BLKSIZE

```

```

*           ALLOCATE BITS TO EACH BLOCK AND
*           PREVENT OVERFLOWING OR UNDERFLOWING

```

THE BUFFER

```

DO K = 1, NBLCKS
$(
  VAR = VARBUF(K)
  BTRATE = C + ALPHIV * ( ALOG(VAR) - LNVARF )
  NBITS = IROUND ( BTRATE )
  IF ( BSTATE <= BTSOUT )
  $(
    NBITS = MAXO ( NBITS, CBITS )
    NBITS = MINO ( NBITS, BLSLFT, MYNBTS )
  $)
  ELSE
  $(
    NBITS = MAXO ( NBITS, MINBTS )
    NBITS = MINO ( NBITS, BLSLFT, MYNBTS, BUFBTS )
  $)
  OBITAB(K) = NBITS
  BLSLFT = BLSLFT - NBITS
  BKSFLT = NBLCKS - K
  BSTATE = BSTATE + NBITS*BLKSIZE - BTSOUT
  BUFBTS = ( BUFSIZE + BTSOUT - BSTATE ) / BLKSIZE
  IF ( BLSLFT <= BKSFLT*MINETS ) BREAK
$)

LSTBLK = K
IF ( BKSFLT >= 1 )
$(
  KK = NBLCKS - BKSFLT + 1
  DO K = KK, NBLCKS
    OBITAB(K) = MINBTS
$)

CALL BITALO ( OBITAB, NBPC, NBPR, LSTBLK, BLSLFT,
              BKSFLT )

RETURN
END

```

* IDENTIFICATION

```

* TITLE BITALO
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE JULY 22, 1979
* LANGUAGE RATFOR
* SYSTEM IBM-370

```

* PURPOSE

```

* THIS ROUTINE WRITES TO THE USER'S TERMINAL INFORMATION
* CONCERNING THE RESULT OF A BIT ALLOCATION PROCESS.

```

* ENTRY POINT

```

* BITALO ( OBITAB, NBPC, NBPR, LSTBLK, BLSLFT )

```

* ARGUMENT LISTING

```

* OBITAB INT THE OUTPUT BIT ALLOCATION TABLE
* NBPC INT THE NUMBER OF BLOCKS PER COLUMN
* NBPR INT THE NUMBER OF BLOCKS PER ROW
* LSTBLK INT THE LAST BLOCK ALLOCATED NON ZERO BITS
* BLSLFT INT THE BITS LEFT IN THE BUFFER
* BKSLEFT INT THE BLOCKS LEFT (ALLOCATED ZERO BITS)

```

* INCLUDE FILES/COMMONS

```

* MACA1 INCLUDE MACRO FILE FOR TOKEN DEFINITION
* TTCOM INCLUDE INCLUDE FILE FOR TERMINAL & RUNFILE I/O

```

* ROUTINES CALLED

```

* NONE

```

```

*****

```

```

* SUBROUTINE BITALO ( OBITAB, NBPC, NBPR, LSTBLK, BLSLFT,
* BKSLEFT )

```

```

* INCLUDE MACA1
* IMPLICIT INTEGER ( A - Z )

```

```

* INCLUDE TTCOM
* INTEGER OBITAB ( NBPR, NBPC )

```

```

* WRITE ( TTYOT, 6000 ) LSTBLK, BLSLFT, BKSLEFT
6000 FORMAT ( /10X, 'LAST BLOCK BITS LEFT BLOCKS LEFT',///.

```

13X, I4, 8X, I4, 8X, I4)

```
*  
WRITE ( TTYOT, 6010 )  
6010 FORMAT ( // )  
*  
DO J = 1, NBPC  
  WRITE ( TTYOT, 6020 ) ( OBITAB(I,J), I = 1, NBPR )  
6020 FORMAT ( 10X, 16I2 )  
*  
RETURN  
END
```

*--FIXBLK FIX BOTTOM AND RIGHTMOST IMAGE BLOCKS MID

* IDENTIFICATION

* TITLE FIXBLK
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE MAY 15, 1979
* LANGUAGE RATFOR
* SYSTEM IBM-370

* PURPOSE

* THIS ROUTINE FIXES THE BOTTOM AND/OR RIGHT EDGES OF THE
* BOTTOM OR RIGHTMOST BLOCKS OF THE IMAGE.

* ENTRY POINT

* FIXBLK (INBLK, IB, JB, NBPC, NBPR, NR, NC)

* ARGUMENT LISTING

* INBLK	* INT	* THE INPUT BLOCK
* IB	* INT	* THE POSITION OF A BLOCK IN A
		* COLUMN OF BLOCKS
* JB	* INT	* THE POSITION OF A BLOCK IN A
		* ROW OF BLOCKS
* NBPC	* INT	* THE NUMBER OF BLOCKS PER COLUMN
* NBPR	* INT	* THE NUMBER OF BLOCKS PER ROW
* NR	* INT	* THE NUMBER OF ROWS IN A BLOCK
* NC	* INT	* THE NUMBER OF COLUMNS PER BLOCK

* ALGORITHM

* IF A BLOCK AT THE BOTTOM IS TO BE FIXED THE LAST
* ROW IN THE BLOCK IS REPLACED BY THE NEXT TO THE
* LAST ROW. IF A BLOCK AT THE RIGHT IS TO BE FIXED
* THE LAST COLUMN IN THE BLOCK IS REPLACED BY THE
* NEXT TO THE LAST COLUMN.

* ROUTINES CALLED

* NONE

* *****

* SUBROUTINE FIXBLK (INBLK, IB, JB, NBPC, NBPR,
* NR, NC)

* IMPLICIT INTEGER (A - Z)

* INTEGER INBLK (NR, NC)

*
*
*
*
CORRECT BLOCKS AT THE RIGHT
AND AT THE BOTTOM

IF (JB == NBPR)
DO K = 1, NR
INBLK (K, NC) = INBLK (K, NC-1)

*
IF (IB == NBPC)
DO K = 1, NC
INBLK (NR, K) = INBLK (NR-1, K)

*
RETURN
END

*--GETEGS GET NEIGHBORING EDGES FROM PREVIOUS BLOCKS MID

* IDENTIFICATION

* TITLE GETEGS
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE MAY 16, 1979
* LANGUAGE RATFOR
* SYSTEM IBM-370

* PURPOSE

* GIVEN THE POSITION OF A BLOCK IN AN IMAGE THIS ROUTINE
* OBTAIN THE NEIGHBORING EDGES COMING FROM PREVIOUS PRO-
* CESSSED TOP AND LEFT BLOCKS.

* ENTRY POINT

* GETEGS (TOPELG, LPTEDG, CORNB, NPPL, NLIN,
* IB, JB, NR, NC, Z, ZLEN)

* ARGUMENT LISTING

* TOPELG INT 1-D ARRAY OF SIZE NPPL CONTAINING
* THE NEIGHBORING TOP EDGES COMING
* FROM THE PREVIOUS PROCESSED ROW OF
* BLOCKS
* LPTEDG INT 1-D ARRAY OF SIZE NLIN CONTAINING
* THE NEIGHBORING LEFT EDGES COMING
* FROM THE PREVIOUS PROCESSED COLUMN
* OF BLOCKS
* CORNR INT THE PREVIOUS PROCESSED VALUE AT
* THE CORNER OF A GIVEN BLOCK
* NPPL INT THE NUMBER OF POINTS PER LINE
* NLIN INT THE NUMBER OF LINES IN THE
* IMAGE
* IB INT THE POSITION OF A BLOCK IN A
* COLUMN OF BLOCKS
* JB INT THE POSITION OF A BLOCK IN A
* ROW OF BLOCKS
* NR INT THE NUMBER OF ROWS PER BLOCK
* NC INT THE NUMBER OF COLUMNS PER BLOCK
* Z INT 1-D ARRAY CONTAINING THE EDGES
* NEIGHBORING TO A GIVEN BLOCK
* IN THE FOLLOWING ORDER: CCRNER,
* LEFT EDGE, TOP EDGE.
* ZLEN INT THE SIZE OF THE ARRAY Z

* ROUTINES CALLED

* NONE

```
*****  
*  
*  
SUBROUTINE GETEGS ( TOPEDG, LPTEDG, CORNR,  
NPPL, NLIN, IB, JB,  
NR, NC, Z, ZLEN )
```

```
*  
*  
IMPLICIT INTEGER ( A - Z )
```

```
*  
*  
INTEGER Z ( ZLEN )  
INTEGER TOPEDG ( NPPL ), LPTEDG ( NLIN )
```

```
*  
*  
GET THE EDGES
```

```
*  
*  
Z(1) = CORNR  
DO K = 1, NR  
  Z ( K + 1 ) = LPTEDG ( NR*(IB-1) + K )  
DO K = 1, NC  
  Z ( NR + K + 1 ) = TOPEDG ( NC*(JB-1) + K )
```

```
*  
*  
RETURN  
END
```

*--BKDPCM DPCM A BLOCK OF DATA

MID

*
* IDENTIFICATION
*

* TITLE BKDPCM
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE JULY 9, 1979
* LANGUAGE RATFOR
* SYSTEM IBM-370

*
* PURPOSE
*

* THIS SUBROUTINE QUANTIZES A BLOCK OF DATA USING A MAX
* QUANTIZER; DPCM THAT BLOCK USING ONE OF SEVERAL POSSIBLE
* DPCM TECHNIQUES; AND FINALLY COMPUTES THE MEAN SQUARE
* ERROR BETWEEN INPUT AND OUTPUT BLOCK.
*

*
* ENTRY POINT
*

* BKDPCM (INBLK, OTBLK, LFBLK, Z, ZLEN, MEAN,
* VAR, NR, NC, NBITS, FRAC, IST,
* SELECT, ERROR)
*

*
* ARGUMENT LISTING
*

* INBLK INT 2-D ARRAY TO HOLD THE INPUT BLOCK
* OTBLK INT 2-D ARRAY TO HOLD THE OUTPUT BLOCK
* LFBLK INT 2-D ARRAY TO HOLD A LOW PASS FILTERED
* VERSION OF THE INPUT BLOCK
* Z INT TOP AND LEFT NEIGHBORING EDGES OF THE
* INPUT BLOCK IN THE FOLLOWING ORDER:
* CORNER, LEFT EDGE, TOP EDGE.
* ZLEN INT THE LENGTH OF THE EDGES ARRAY Z
* MEAN REAL THE MEAN OF THE DPCM CORRECTIONS
* VAR REAL THE VARIANCE OF THE DPCM CORRECTIONS
* NR INT NUMBER OF ROWS IN INPUT BLOCK
* NC INT NUMBER OF COLUMNS IN INPUT BLOCK
* NBITS INT THE NUMBER OF QUANTIZING BITS
* FRAC REAL FRACTION OF STANDARD DEVIATION THAT
* RANGE OF DPCM DITHER SHOULD BE
* IST DINT SEED FOR UNIFORM RANDOM NUMBER GENERATOR
* SELECT INT SELECTION NUMBER FOR DPCM TECHNIQUES
* (2) 2-D DPCM FLAT MODEL LSE
* (3) MOD DPCM FLAT MODEL LSE
* (4) 2-D DPCM FLAT MODEL MVE
* (5) MOD DPCM FLAT MODEL MVE
* (6) 2-D DPCM SLOPED MODEL LSE
* (7) MOD DPCM SLOPED MODEL LSE
* (8) 2-D DPCM SLOPED MODEL MVE
* (9) MOD DPCM SLOPED MODEL MVE
* ERROR REAL THE MEAN SQUARE ERROR BETWEEN INPUT AND
* OUTPUT BLOCKS
*

```
* ROUTINES CALLED
```

```
*
* EGSINT 2-D INTERPOLATION BETWEEN EDGES (GIPSY)
* QTZSAM QUANTIZATION END POINTS AND MEANS (GIPSY)
* DPCM2L TWO DIMENSIONAL ( 2-D ) DPCM (GIPSY)
* DPCMIX MODIFIED 2-D DPCM ( USING LPP BLOCK ) (GIPSY)
* MSERR MEAN SQUARE ERROR (GIPSY)
*
```

```
*****
```

```
*
* SUBROUTINE BKDPCM ( INBLK, OTBLK, LPBLK, Z, ZLEN, MEAN,
* VAR, NR, NC, NBITS, FRAC, IST,
* SELECT, ERROR )
```

```
*
* TYPE STATEMENTS
```

```
*
* IMPLICIT INTEGER ( A - Z )
* INTEGER*4 IST
* INTEGER INBLK(NR,NC), OTBLK(NR,NC), LPBLK(NR,NC)
* INTEGER Z ( ZLEN )
* INTEGER LNIN(20), LNOUT(20), LNPOUT(20)
* INTEGER LNLPP(20), LNXTP(20)
* INTEGER QTABLE(512)
* INTEGER WGTS(16), WGTSX(27)
* REAL QENDS(65), QMEANS(64), ERROR, MEAN, VAR
* REAL RANGE, FRAC, SQRT
* DATA MAX /255/
* DATA WGTS / 4*1,
* 5, 5, 7, 7,
* -1, 0, 1, 2,
* -7, 6, 20, 35 /
* DATA WGTSX / 2, 2, 1, 2, 1, 1, 2, 1, 1,
* 10, 7, 10, 7, 5, 7, 10, 7, 10,
* 9*1 /
```

```
*
* INITIALIZATIONS
```

```
*
* NLEVELS = 2**NBITS
* NCP1 = NC + 1
* NCP2 = NC + 2
* BLKSIZE = NR*NC
* RANGE = FRAC*SQRT(VAR)
```

```
*
* TREAT NBITS = 0 SEPARATELY
```

```
*
* IF ( NBITS == 0 )
* $(
* CALL EGSINT ( Z, ZLEN, INBLK, OTBLK, NR, NC,
* MAX, IST )
* CALL MSERR ( INBLK, OTBLK, ERROR, NPPL )
* RETURN
* $)
```

```

*
*
*           COMPUTE THE END POINTS AND MEANS USING
*           MAX QUANTIZER
*
CALL QTZSAM ( MEAN, VAR, NEITS, QENDS, QMEANS )
*
*
QENDS(1) = -MAX - 1
QENDS(NLEVELS+1) = MAX
*
*           COMPUTE SIZE OF QUANTIZING TABLE
*
NTABLE = 2*MAX + 1
*
*           COMPUTE THE QUANTIZING TABLE
*
DO J = 1, NLEVELS
$(
  LOWEND = QENDS(J) + MAX + 2
  UPPEND = QENDS(J+1) + MAX + 1
  DO I = LOWEND, UPPEND
    QTABLE(I) = IROUND(QMEANS(J))
  $)
*
*           COMPUTE THE BLOCK OF OUTPUT
*
LNPOUT(1) = Z(1)
DO J = 1, NC
  LNPOUT(J+1) = Z(1+NR+J)
*
IF ( SELECT == 3 ) SELECT = 7
IF ( SELECT == 9 ) SELECT = 5
*
IF ( SELECT==2 | SELECT==4 | SELECT==6 | SELECT==8 )
$(
  SLCT = (SELECT-2)*2 + 1
  DO I = 1, NR
    $(
      LNIN(1) = Z(I+1)
      DO J = 1, NC
        LNIN(J+1) = INBLK(I,J)
        CALL DPCN2L ( LNIN, NCP1, LNPOUT, LNOUT,
                    QTABLE, NTABLE, MAX, RANGE,
                    IST, WGTS(SLCT), NGEN, NOSED )
      DO J = 1, NC
        $(
          OGTNE = MIN0 ( MAX, LNOUT(J+1) )
          OTBLK(I,J) = OGTNE
          LNPOUT(J+1) = OGTNE
        $)
      LNPOUT(1) = Z(I+1)
    $)
  $)
*)

```

```

IF ( SELECT==5 | SELECT==7 )
$(
  SLCT = (SELECT-3)*9/2 + 1
  DO I = 1, NR
  $(
    I1 = MINO ( I+1, NR )
    LNIN(1) = Z(I+1)
    DO J = 1, NC
    $(
      LNIN(J+1) = INBLK(I,J)
      LNLPP(J+1) = LFBLK(I,J)
      LNXTLP(J+1) = LFBLK(I1,J)
    $)
    LNXTLP(1) = LNXTLP(2)
    CALL DPCMX ( LNIN, NCP1, LNLPP, LNXTLP, LNPOUT,
                LNOUT, QTABLE, NTABLE, MAX, RANGE,
                IST, WGTSX(SLCT), NGEN, NUSED )
    DO J = 1, NC
    $(
      OGTNE = MINO ( MAX, LNOUT(J+1) )
      OTBLK(I,J) = OGTNE
      LNPOUT(J+1) = OGTNE
    $)
    LNPOUT(1) = Z(I+1)
  $)
$)
*
*
*
*
      COMPUTE THE RMS ERROR
CALL MSERR ( INBLK, OTBLK, ERROR, NPPL )

RETURN
END

```

*
* IDENTIFICATION

* TITLE EGSINT
* AUTHOR OSCAR A. ZUNIGA
* VERSION A.01
* DATE JULY 3, 1979
* LANGUAGE RATFOR
* SYSTEM IBM-370

* PURPOSE

* THIS ROUTINE PERFORMS A 2-D INTERPOLATION BETWEEN THE
* EDGES OF A BLOCK IN A DPCM LIKE FASHION, AND THEN
* CORRECTS THAT ESTIMATION USING THE MEAN AND VARIANCE
* OF THE DIFFERENCE BETWEEN INPUT BLOCK AND INTERPOLATED
* BLOCK

* ENTRY POINT

* EGSINT (Z, ZLEN, INBLK, OTBLK, NR, NC, MAX, IST)

* ARGUMENT LISTING

* Z INT THE TOP AND LEFT NEIGHBORING EDGES
* OF A BLOCK IN THE FOLLOWING ORDER:
* CORNER, LEFT EDGE, TOP EDGE.
* ZLEN INT SIZE OF THE EDGES ARRAY
* INBLK INT THE INPUT BLOCK
* OTBLK INT THE ESTIMATED OUTPUT BLOCK BASED ON
* THE NEIGHBORING EDGES OF THE INPUT
* BLOCK
* NR INT NUMBER OF ROWS IN THE BLOCKS
* NC INT NUMBER OF COLUMNS IN THE BLOCKS
* MAX INT THE MAXIMUM GRAYTONE IN A BLOCK
* IST DIN SEED FOR UNIFORM RANDOM NUMBER
* GENERATOR

* ALGORITHM

* THE INTERPOLATION IS DONE IN A DPCM LIKE FASHION BUT
* NO CORRECTIONS ARE ADDED. THE TOP AND LEFT NEIGHBORING
* EDGES COMING FROM PREVIOUS PROCESSED BLOCKS ARE USED
* TO INITIALIZE THE ALGORITHM. THE INTERPOLATED BLOCK IS
* CORRECTED BY ADDING UNIFORM NOISE WHOSE MEAN AND VARIANCE
* EQUAL THAT OF THE DIFFERENCE BETWEEN INPUT AND INTERPOLA-
* TED BLOCK

* ROUTINES CALLED

* MVARDL MEAN & VARIANCE OF A DIFFERENCE BLOCK (GIPSY)
* RCM UNIFORM RANDOM NUMBER GENERATOR (GIPSY)

```

*****
SUBROUTINE EGSINT ( Z, ZLEN, INBLK, OTBLK, NR, NC,
                  MAX, IST )

```

```

IMPLICIT INTEGER ( A - Z )
INTEGER Z ( ZLEN ), INBLK ( NR, NC ), OTBLK ( NR, NC )
INTEGER LNPOUT ( 20 ), LNOUT ( 20 )
REAL MEAN, VAR, STDEV, SQRT, RCM, DITHER

```

```

BLKSZE = NR * NC
NCP1 = NC + 1
LNPOUT(1) = Z(1)
DO K = 1, NC
  LNPOUT(K+1) = Z(1+NR+K)

```

```

DO I = 1, NR
$(
  LNOUT(1) = Z(I+1)
  DO J = 2, NCP1
$(
  J1 = MINO ( J+1, NCP1 )
  ISUM1 = LNPOUT(J-1) + LNPOUT(J)
  ISUM2 = LNPOUT(J1) + LNOUT(J-1)
  LNOUT(J) = ( ISUM1 + ISUM2 + 2 ) / 4
$)
  DO J = 1, NC
$(
  OTBLK(I,J) = LNOUT(J+1)
  LNPOUT(J+1) = LNOUT(J+1)
$)
  LNPOUT(1) = Z(I+1)
$)

```

```

CALL MVARDL ( INBLK, OTBLK, MEAN, VAR, BLKSZE )
STDEV = SQRT ( VAR )
DO I = 1, NR
  DO J = 1, NC
$(
  DITHER = MEAN + STDEV * ( RCM(IST)-0.5 )
  OGTNE = OTBLK(I,J) + IROUND ( DITHER )
  OGTNE = MAXO ( 0, OGTNE )
  OGTNE = MINO ( OGTNE, MAX )
  OTBLK(I,J) = OGTNE
$)

```

```

RETURN
END

```


C

COMPUTE THE STANDARD DEVIATION

```

XSD=SQRT (XDVAR)
NN=2** (NBIT-1)
MNN=2*NN+1
NNX=FLOAT (2*NN)
STEP=6.0/NNX

```

C
C
C
C
C

PICK UP THE APPROPRIATE ENTRIES FROM THE TABLES AND STICK THEM IN QENDS QND QMEANS ADJUST FOR MEAN AND ATANDARD DEVIATION.

```

DO 10 I=1, NN
GO TO (1,2,3,4,5,6,7), NBIT
1  NNI=NN+I
   QENDS (NNI) =X1 (I) *XSD+XDMEAN
   QMEANS (NNI) =Y1 (I) *XSD+XDMEAN
   GO TO 9
2  NNI=NN+I
   QENDS (NNI) =X2 (I) *XSD+XDMEAN
   QMEANS (NNI) =Y2 (I) *XSD+XDMEAN
   GO TO 9
3  NNI=NN+I
   QENDS (NNI) =X3 (I) *XSD+XDMEAN
   QMEANS (NNI) =Y3 (I) *XSD+XDMEAN
   GO TO 9
4  NNI=NN+I
   QENDS (NNI) =X4 (I) *XSD+XDMEAN
   QMEANS (NNI) =Y4 (I) *XSD+XDMEAN
   GO TO 9
5  NNI=NN+I
   QENDS (NNI) =X5 (I) *XSD+XDMEAN
   QMEANS (NNI) =Y5 (I) *XSD+XDMEAN
   GO TO 9
6  NNI=NN+I
   QENDS (NNI) = (I-1) *STEP*XSD+XDMEAN
   QMEANS (NNI) = (I-0.5) *STEP*XSD+XDMEAN
   GO TO 9
7  NNI=NN+I
   QENDS (NNI) =X7 (I) *XSD+XDMEAN
   QMEANS (NNI) =Y7 (I) *XSD+XDMEAN
   GO TO 9
8  NNI=NN+I
   QENDS (NNI) =X8 (I) *XSD+XDMEAN
   QMEANS (NNI) =Y8 (I) *XSD+XDMEAN
9  CONTINUE

```

C

10 CONTINUE

C
C
C
C

SET UP END POINTS AND MEANS ON THE LOWER SIDE OF THE MEAN.

```

MNN=NN-1
MND=NN+1

```

C

```

DO 12 I=1, NMH
C
  N1=NMID+I
  N2=NMID-I
  QENDS (N2)=QENDS (N1)*(-1.0) + 2.0 * QENDS ( NMID )
C
12  CONTINUE
C
C          FIRST AND LAST END POINTS ARE - AND +
C          INFINITY RESPECTIVELY)
C
QENDS (1)=-0.1E+6
QENDS (NMH)=0.1E+6
C
DO 13 I=1, NH
C
  I1=NMH-I
  QMEANS (I)=QMEANS (I1) *(-1.0) + 2.0 * QENDS ( NMID )
C
13  CONTINUE
C
RETURN
END

```

C
C IDENTIFICATION

TITLE	DPCM2L
AUTHOR	ROBERT M. HARALICK
VERSION	A.01
DATE	MARCH 1977
LANGUAGE	FORTRAN
SYSTEM	PDP-15

C
C UPDATE # 1

AUTHOR	OSCAR A. ZUNIGA
DATE	OCTOBER 10, 1978
VERSION	B.01
LANGUAGE	FORTRAN IV
PURPOSE	THE ARRAY WT HAS BEEN ADDED TO THE ARGUMENT LIST TO COMPUTE THE DPCM PREDICTOR USING A WEIGHTED AVERAGE

C
C PURPOSE

THIS SUBROUTINE IMPLEMENTS A SIMPLE TWO DIMENSIONAL
LINEAR FOURTH ORDER DPCM FOR A LINE OF DATA

C
C ENTRY POINT

DPCM2L (LMIN, NUMPPL, LNPOUT, LNOUT, QTABLE, NTABLE,
MAX, RANGE, IST, WT, NGEN, NUSED)

C
C ARGUMENT LISTING

LMIN	INT	INPUT LINE OF DATA
NUMPPL	INT	NUMBER OF VALUES IN INPUT LINE
LNPOUT	INT	PREVIOUS OUTPUT LINE
LNOUT	INT	CURRENT OUTPUT LINE
QTABLE	INT	QUANTIZING TABLE
		QTABLE(IDIF) GIVES THE RECONSTRUCTED VALUE FOR ANY DIFFERENCE OF VALUE IDIF
NTABLE	INT	LENGTH OF QUANTIZING TABLE
MAX	INT	MAXIMUM VALUE OF AN INPUT
RANGE	REAL	RANGE OF UNIFORMLY DISTRIBUTED DITHER
IST	DINT	RANDOM NUMBER GENERATOR SEED
WT	INT	WEIGHTS FOR COMPUTING PREDICTOR
NGEN	INT	NUMBER OF OUTPUT RECORDS GENERATED
NUSED	INT	NUMBER OF INPUT RECORDS USED

C
C INTERNAL VARIABLES

IPRED	INT	PREDICTED VALUE, COMPUTED FROM SUM OF THE FOUR NEAREST NEIGHBORS ABOVE AND TO THE LEFT OF THE CURRENT (C,D,E,R) :
-------	-----	---

```

C           A B C D E F . . . ( PREVIOUS LINE )
C           P Q R S           ( CURRENT LINE, COL S )
C
C ROUTINES CALLED
C
C     MAXO   MAXIMUM OF A SET OF INTEGERS           (SYSTEM ROUTINE)
C     IROUND ROUND OFF TO NEAREST INTEGER           (USER ROUTINE)
C     MINO   MINIMUM OF A SET OF INTEGERS           (SYSTEM ROUTINE)
C     RCM    UNIFORM RANDOM NUMBER GENERATOR        (SYSTEM ROUTINE)

```

```

C
C     SUBROUTINE DPCM2L( LNIN, NUMPPL, LNPOUT, LNCUT, QTABLE,
2     NTABLE, MAX,RANGE,IST, WT,NGEN, NUSED )
C
C     INTEGER*4 IST
C     INTEGER LNIN(NUMPPL), LNPOUT(NUMPPL), LNCUT(NUMPPL)
C     INTEGER QTABLE(NTABLE), WT(4), DPCSUM
C
C     NRMLZF = WT(1) + WT(2) + WT(3) + WT(4)
C     IRND = NRMLZF/2
C
C     MAX1=1+MAX
C     LNOUT(1)=LNIN(1)
C
C     DO 1 I=2, NUMPPL
C     DITHER=(RCM(IST)-.5)*RANGE
C     IDTHR=IROUND(DITHER)
C     J=MINO(I+1, NUMPPL)
C     DPCSUM=LNPOUT(I-1)*WT(1)+LNPOUT(I)*WT(2)+LNPOUT(J)*WT(3)+
1     LNOUT(I-1)*WT(4)
C     IPRED=(DPCSUM+IRND)/NRMLZF
C     IDIF=LNIN(I)-IPRED+MAX1+IDTHR
C     IDIF=MAXO(IDIF, 1)
C     IDIF=MINO(IDIF, NTABLE)
C     LNOUT(I)=IPRED+QTABLE(IDIF)-IDTHR
C     LNOUT(I)=MAXO(LNOUT(I), 0)
1 CONTINUE
C
C     NGEN = 1
C     NUSED = 1
C
C     RETURN
C     END

```



```

C
C ROUTINES CALLED
C
C      MINO    MINIMUM OF A SET OF INTEGERS          (SYSTEM)
C      MAXO    MAXIMUM OF A SET OF INTEGERS          (SYSTEM)
C      IROUND  ROUND OFF TO NEAREST INTEGER          (USER)
C      RCM     UNIFORM RANDOM NUMBER GENERATOR      (SYSTEM)
C
C*****
C
C      SUBROUTINE DPCMXX ( LNIN, NUMPPL, LNLPP, LNXTLP, LNPOUT,
2      LNOUT, QTABLE, NTABLE, MAX, RANGE,
3      IST, WT, NGEN, NUSED )
C
C      INTEGER*4 IST
C      INTEGER LNOUT(NUMPPL), QTABLE(NTABLE), LNLPP(NUMPPL)
C      INTEGER LNIN(NUMPPL), LNXTLP(NUMPPL), LNPOUT(NUMPPL)
C      INTEGER WT ( 3, 3 ), DPCSUM
C
C      MAXP1 = MAX + 1
C--      NMPPL1 = NUMPPL - 1
C      NRMLZF = 0
C
C      DO 1 I=1,3
C      DO 1 J=1,3
C      NRMLZF = NRMLZF + WT(I,J)
1      CONTINUE
C
C      IRND = NRMLZF/2
C
C      DO 1ST AND LAST DPCM'S SPECIALLY
C
C--      LNOUT(1) = LNIN(1)
C      LNOUT(NUMPPL) = LNIN(NUMPPL)
C
C      NOW DO THE REST OF THE LINE
C
C      DO 1000 I=2,NUMPPL
C
C      IM1 = I-1
C      IP1 = MINO ( I+1, NUMPPL )
C
C      DITHER=(RCM(IST) -.5) *RANGE
C      IDTBR=IROUND(DITHER)
C
C      DPCSUM = LNPOUT(IM1)*WT(1,1) + LNPOUT(I)*WT(1,2) +
2      LNPOUT(IP1)*WT(1,3)
C      ITSUM = LNOUT(IM1)*WT(2,1) + LNLPP(I)*WT(2,2) +
2      LNLPP(IP1)*WT(2,3)
C      NXTSUM = LNXTLP(IM1)*WT(3,1) + LNXTLP(I)*WT(3,2) +
2      LNXTLP(IP1)*WT(3,3)

```

```
C      IPRED = (DPCSUM + ITSUM + NITSUM + IRND)/NBMLZF
      IDIF = LNIN(I) - IPRED + MAXP1+IDTHR
      IDIF=MAXO(1,IDIF)
      IDIF=MINO(NTABLE,IDIF)
      LNOUT(I) = IPRED + QTABLE(IDIF) -IDTHR
      LNOUT(I) = MAXO(LNOUT(I),0)

C      1000  CONTINUE
C
      NGEN = 1
      NUSED = 1

C      RETURN
      END
```

#--MSERR MEAN SQUARE ERROR

MID

IDENTIFICATION

TITLE MSERR
AUTHOR OSCAR A. ZUNIGA
VERSION A.01
DATE JULY 6, 1979
LANGUAGE RATFOR
SYSTEM IBM-370

PURPOSE

THIS ROUTINE COMPUTES THE MEAN SQUARE ERROR BETWEEN TWO
ARRAYS OF INTEGER DATA.

ENTRY POINT

MSERR (LINE1, LINE2, MSE, NPPL)

ARGUMENT LISTING

LINE1 INT 1-D ARRAY OF DATA
LINE2 INT 1-D ARRAY OF DATA
MSE REAL THE MEAN SQUARE ERROR
NPPL INT THE SIZE OF THE INPUT ARRAYS

ROUTINES CALLED

NONE

SUBROUTINE MSERR (LINE1, LINE2, MSE, NPPL)

INTEGER LINE1 (NPPL), LINE2 (NPPL)
REAL MSE
MSE = 0.0

DO K = 1, NPPL
MSE = MSE + (LINE1(K) - LINE2(K)) ** 2
MSE = MSE / NPPL

RETURN
END

* IDENTIFICATION

* TITLE	UPDEGS
* AUTHOR	OSCAR A. ZUNIGA
* VERSION	A.01
* DATE	MAY 16, 1979
* LANGUAGE	RATFOR
* SYSTEM	IBM-370

* PURPOSE

* THIS ROUTINE UPDATES THE CONTENTS OF THE BUFFERS CONTAINING THE NEIGHBORING TOP EDGES AND THE NEIGHBORING LEFT EDGES WHENEVER A BLOCK HAS BEEN PROCESSED.

* ENTRY POINT

* UPDEGS (OTBLK, TOPEDG, LPTEDG, CORNR, NPPL, NLIN, IB, JB, NR, NC)

* ARGUMENT LISTING

* OTBLK	INT	THE PROCESSED OUTPUT BLOCK
* TOPEDG	INT	THE BUFFER THAT STORES ALL NEIGHBORING EDGES IN THE PREVIOUS PROCESSED ROW OF BLOCKS
* LPTEDG	INT	THE BUFFER THAT STORES ALL NEIGHBORING EDGES IN THE PREVIOUS PROCESSED COLUMN OF BLOCKS
* CORNR	INT	THE PREVIOUS PROCESSED VALUE AT THE CORNER OF A GIVEN BLOCK
* NPPL	INT	THE NUMBER OF POINTS PER LINE IN THE IMAGE
* NLIN	INT	THE NUMBER OF LINES IN THE IMAGE
* IB	INT	THE POSITION OF A BLOCK IN A COLUMN OF BLOCKS
* JB	INT	THE POSITION OF A BLOCK IN A ROW OF BLOCKS
* NR	INT	THE NUMBER OF ROWS PER BLOCK
* NC	INT	THE NUMBER OF COLUMNS PER BLOCK

* ROUTINES CALLED

* NONE

* SUBROUTINE UPDEGS (OTBLK, TOPEDG, LPTEDG, CORNR, NPPL, NLIN, IB, JB, NR, NC)

DATE
FILMED
— 8