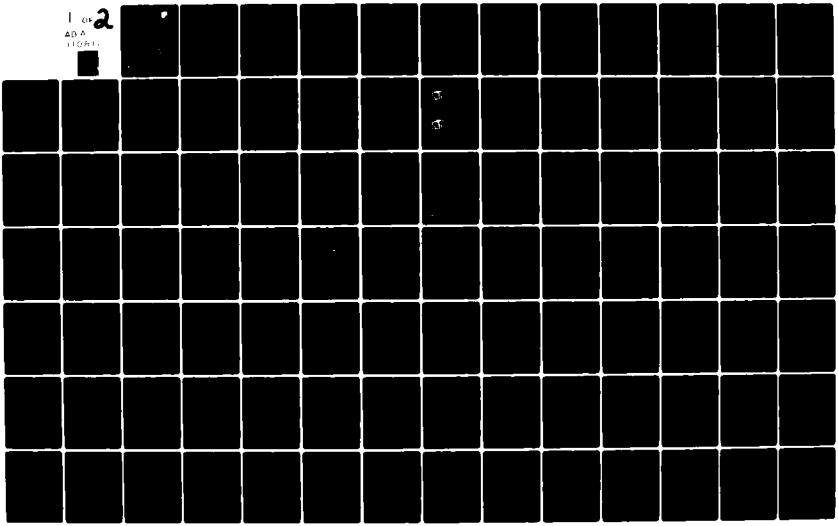
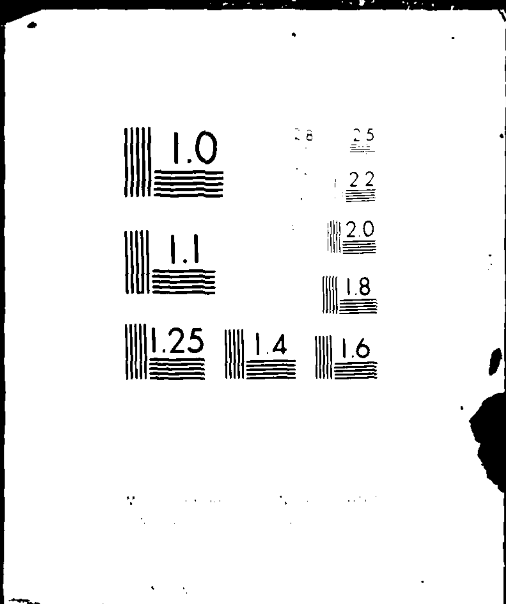
AD-A110 811　VERAC INC SAN DIEGO CA　F/G 5/8
IMAGE COMPRESSION RESEARCH.(U)
AUG 81　M S MURPHY　F30602-80-C-0168
UNCLASSIFIED　R-021-81　RADC-TR-81-223　NL

1 OF 2
AD A
110811



1 OF 2

AD A 110811



RADC-TR-81-223
Final Technical Report
August 1981

IMAGE COMPRESSION RESEARCH

VERAC Incorporated

Dr. Michael S. Murphy

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

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

RADC-TR-81-223 has been reviewed and is approved for publication.

APPROVED:

JOHN T. BOLAND
Project Engineer

APPROVED:

OWEN R. LAWTER, Colonel, USAF
Chief, Intelligence and Reconnaissance Division

FOR THE COMMANDER:

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRRE) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-223	2. GOVT ACCESSION NO. AD-A110811	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) IMAGE COMPRESSION RESEARCH		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report May 80 - June 81
		6. PERFORMING ORG. REPORT NUMBER R-021-81
7. AUTHOR(s) Dr. Michael S. Murphy		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0168
9. PERFORMING ORGANIZATION NAME AND ADDRESS VERAC Incorporated 10975 Torreyana Road, Suite 300 San Diego CA 92121		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62702F 45941832
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRRE) Griffiss AFB NY 13441		12. REPORT DATE August 1981
		13. NUMBER OF PAGES 198
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

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)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John T. Boland (IRRE)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image Coding
Image Compression
Transform Coding
Singular Valve Decomposition
Karhunen-Loeve Transform
Cosine Transform
Hadamard Transform

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The vast quantities and high generation rates of tactical imagery require very efficient data compression in order to conserve precious bandwidth for transmission and to limit the required storage volume for archiving. This report describes the results of efficiency and image quality comparisons for several transform image coding techniques. Specifically, the research effort was focused on developing the Singular Valve Decomposition (SVD) as an approach to image compression. Detailed comparisons were

DD FORM 1 JAN 73 1473A EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

to the two-dimensional cosine transform, Hadamard transform and Karhunen-Loeve techniques. High and low altitude aerial imagery, and IR and SAR imagery were included in this study. Images were coded with rates in the range .25 to 1.5 bits per pixel (bpp). (Original images were digitized at 8 bpp). The order of performance, as measured by rms error (reconstructed image versus original) versus bpp, as well as by visual image quality judgements, was first, cosine and Karahunen-Loeve (nearly the same), second, SVD, and third, Hadamard. Computational burdens, however, are least for the Hadamard, intermediate for cosine and Karhunen-Loeve and most for the SVD technique.

Table of Contents

Section		Page
1.0	INTRODUCTION	1-1
	1.1 Summary of Results	1-2
	1.2 Roadmap	1-3
2.0	PROJECT SCOPE AND OBJECTIVES	2-1
	2.1 Algorithms	2-4
	2.2 Imagery of Interest	2-4
	2.2.1 Post Processing for Blockiness Suppression	2-6
3.0	OVERVIEW OF TRANSFORM IMAGE CODING	3-1
	3.1 Image Coding Approaches	3-1
	3.2 Transform Image Coding	3-4
	3.3 Block Transformations	3-6
	3.4 Block Size	3-7
	3.5 Unitary Transforms	3-9
	3.6 Applicable Transformations	3-13
	3.6.1 Optimal Decorrelating Trasform	3-15
	3.6.2 Cosine Transform	3-16
	3.6.3 Hadamard Transform	3-16
	3.6.4 Singular Value Decomposition	3-17
	3.6.5 Focus of New Developments	3-18
4.0	CLASS-ADAPTIVE KARHUNEN-LOEVE TRANSFORM CODER	4-1
	4.1 Class-Adaptive Karhunen-Loeve Transformation	4-1
	4.1.1 The Separable Covariance Assumption	4-1
	4.1.2 KLT Definition	4-2

Table of Contents
(continued)

Section   Page

4.1.3 Class Adaptivity. 4-5
4.1.4 KLT Computational Algorithms. 4-7

4.2 KLT Preprocessing. 4-8

4.2.1 Classify Blocks 4-8
4.2.2 Compute KLT Operator Matrices and Predicted Statistics 4-10
4.2.3 Collect Empirical Statistics. 4-13
4.2.4 KLT Preprocessing Summary 4-15

4.3 KLT Image Coding 4-15

4.3.1 Overhead Encoding 4-18
4.3.2 KLT Coefficient Coding. 4-18

5.0 SINGULAR VALUE DECOMPOSITION TRANSFORM CODER. 5-1

5.1 SVD Computational Algorithms 5-2
5.2 Decorrelation and Energy Compaction of SVD Coefficients. 5-10

5.2.1 SVD Reordering. 5-16
5.2.2 Singular Vector Orthogonalization . . . 5-17
5.2.3 Repolarization. 5-20

5.3 Preprocessing. 5-21
5.4 Singular Value/Vector Coding 5-24

5.4.1 Bit Allocation. 5-25
5.4.2 Singular Value/Vector Coders. 5-28

Table of Contents
(continued)

Section		Page
	5.4.3 Coding the SVD Orthogonal Expansion Coefficients	5-31
6.0	RATE EQUILIZATION	6-1
7.0	ALGORITHM EVALUATION	7-1
	7.1 Preliminary Evaluation	7-2
	7.1.1 Optimal Class Boundaries	7-2
	7.1.2 Preliminary Algorithm Evaluation	7-4
	7.2 Comprehensive Evaluation	7-19
8.0	CONCLUSIONS	8-1
9.0	REFERENCES	9-1
APPENDICES		
A	DERIVATION OF THE SEPARABLE KARHUNEN-LOEVE TRANSFORMATION AND ASSOCIATED STATISTICS	A-1
B	HOMOGENIZING IN TRANSFORM COEFFICIENT SPACE	B-1
C	OPTIMAL CODER ASSIGNMENTS FOR KLT COEFFICIENTS	C-1

Table of Contents
(concluded)

Section		Page
D	HOMOGENIZING SVD'S OF PRETRANSFORMED BLOCKS	D-1
E	SVD ORTHOGONAL EXPANSION COEFFICIENT STATISTICS . . .	E-1
F	SVD COEFFICIENT CODING: BIT ALLOCATION	F-1

List of Figures

Figure		Page
2-1	Automated Compression/Decompression of Tactical Imagery	2-2
3-1	Transform Image Coding Chain	3-5
3-2	Unitary Block Transform	3-11
4-1	Comparison of Separable and Non-Separable Image Covariances	4-3
4-2	KLT Preprocessing	4-9
4-3	KLT Image Coding Chain	4-16
4-4	Example Bit Assignment Arrays	4-20
4-5	Example Three-Bit Max Quantizer for Gaussian PDF	4-23
5-1	SVD Coding Chain	5-3
5-2	Direct Pixel Block SVD Calculation	5-6
5-3	Indirect Pixel Block SVD Calculation	5-6
5-4	Finding the SVD via a Coefficient-Block SVD Calculation	5-8
5-5	Collapsing the Inverse 2D Pre-transfomation and the Forward 1D Singular Vector Transformation into the Identity	5-9

List of Figures
(continued)

Figure | Page

5-6 Combined Algorithm: Efficient Extraction of Both Singular Values and 1D Transform Coefficients of Singular Vectors 5-10

5-7 Example Singular Vector and its Cosine Transform 5-12

5-8 Example Singular Value Statistics 5-14

5-9 Two Impelementations for Extracting Singular Values and Singular Vector Transform Coefficients 5-15

5-10 Example of Migration of Energy Concentrated with Frequency 5-18

5-11 SVD Preprocessing 5-23

5-12 Example SVD Bit Assignment Arrays 5-29

5-13 Example Singular Value Statistics 5-30

6-1 Rate Equilization 6-2

7-1 Coding Algorithm Comparisons MSE versus Rate Airfield Image 7-6

7-2 Coding Algorithm Comparison MAE versus Rate Airfield Image 7-8

7-3 Original Image 7-9

List of Figures
(continued)

Figure		Page
7-4	Reconstructed Images	7-10
7-5	Reconstructed Images	7-11
7-6	Reconstructed Images	7-12
7-7	Reconstructed Images	7-13
7-8	Reconstructed Images	7-14
7-9	Error Images	7-15
7-10	Error Images	7-16
7-11	Original Airfield Image	7-22
7-12	Coded Airfield Images	7-23
7-13	Coded Airfield Images	7-24
7-14	Coded Airfield Images	7-25
7-15	Original Harbor Image	7-26
7-16	Coded Harbor Images	7-27
7-17	Coded Harbor Images	7-28
7-18	Coded Harbor Images	7-29

List of Figures
(continued)

Figure		Page
7-19	Original Infrared Image	7-30
7-20	Coded Infrared Images	7-31
7-21	Coded Infrared Images	7-32
7-22	Coded Infrared Images	7-33
7-23	Original SAR Image	7-34
7-24	Coded SAR Images	7-35
7-25	Coded SAR Images	7-36
7-26	Coded SAR Images	7-37
7-27	Coding Algorithm Comparisons; MSE versus Rate, Airfield Image	7-38
7-28	Coding Algorithm Comparisons; MSE versus Rate, Harbor Image	7-39
7-29	Coding Algorithm Comparisons; MSE versus Rate, IR Image	7-40
7-30	Coding Algorithm Comparisons; MSE versus Rate, SAR Image	7-41
7-31	Coding Algorithm Comparisons; MAE versus Rate, Airfield Image	7-42

List of Figures
(continued)

Figure		Page
7-32	Coding Algorithm Comparisons; MAE versus Rate, Harbor Image	7-43
7-33	Coding Algorithm Comparisons; MAE versus Rate, IR Image	7-44
7-34	Coding Algorithm Comparisons; MAE versus Rate, SAR Image	7-45

ACKNOWLEDGEMENTS

This report summarizes the results of a research effort conducted for Rome Air Development Center by VERAC, Incorporated, during the period May 1980 through June 1981. Work was performed under contract number F30602-80-C-0168 and was monitored by Mr. John Boland of RADC/IRRE.

A number of VERAC personnel participated in the effort. Dr. Harold J. (Pete) Payne served as program manager, Dr. Michael S. Murphy as principal investigator, and Messrs. Richard M. Crawford and Edwin H. Schnaath as programmer/analysts. Report preparation was handled by Ms. Denise Derringer-Straub, with assistance from Ms. Cheryl Ritter and artwork by Ms. Erika Pearl.

1.0 INTRODUCTION

This document presents the results of a twelve month, 6.1 research effort sponsored by RADC/IRRE and performed at VERAC, Incorporated in San Diego. The primary thrust of the study involved the development of image coding techniques based upon the singular value decomposition (SVD) operation, and intended for application to bandwidth compression of tactical imagery. An important aspect of the study was a thorough comparison of the new SVD approaches to other transform image coding schemes.

Compression algorithms based upon four distinct image transformations were examined:

- Singular Value Decomposition,
- Karhunen-Loeve,
- Cosine, and
- Hadamard.

The singular value decomposition coding algorithms were new, the Karhunen-Loeve coding algorithms were extensions of previous work, and the cosine and Hadamard coding algorithms were baselines representative of the current state of the art in transform image coding.

All algorithms were designed to be as similar as possible, both in philosophy and implementation. Differences were restricted entirely to the particular image transformation employed in each case. The result was a common framework in which the various transformations were evaluated for coding efficiency and image quality, without contamination by performance differences that can arise due to variations in other aspects of coder implementation. This was the first time, to our knowledge, that such a well-controlled environment was established for comparison of alternative transform image coders.

The study consisted of three efforts:

- Algorithm Design,
- Software Development, and
- Coder Evaluation.

This document is primarily concerned with describing the various algorithms developed under the study and summarizing their comparative performances both among themselves and with respect to the baseline algorithms. The software developed under the contract to implement the coding algorithms is described in a companion report, "Image Compression Software Documentation," VERAC Technical Report No. R-022-81.

1.1 Summary of Results

The singular value decomposition is the mathematical transformation which achieves maximum energy compaction into the fewest number of transform coefficients, called singular values in the case of the SVD. Thus, the SVD represents a potentially very useful operation for reducing the bandwidth required to encode image data, since a small number of singular values can be encoded in place of a larger number of pixels. The SVD achieves this efficient compaction by tailoring the transform operator -- called singular vectors for the SVD -- to the image data itself. The price for this tailoring is that the singular vectors must also be encoded along with the singular values to permit the decoder to perform image reconstruction.

A number of SVD-based image coding algorithms were developed. The variations were due to different approaches to efficiently coding singular vectors. The result was an assortment of SVD coding algorithms of varying complexity which were identified, implemented and evaluated.

In addition to the SVD algorithms, a class-adaptive Karhunen-Loeve transform (KLT) algorithm was also developed as a generalization of the

SVD approach. The basic idea is to replace the SVD's tailoring of transform operators to image data by the KLT's tailoring to average image characteristics. The result is a reduction in the number of different transform operators that must be encoded: instead of one for each block of imagery, one for each class of imagery is now required. The price for this improvement is a concomitant lessening of the high energy compaction produced by the SVD. Two versions of KLT code were developed, one depending upon explicit training on image data, and other computationally simpler but based upon an assumed image model.

The various SVD and KLT algorithms were evaluated against each other as well as against the baseline algorithms, which employed the fixed (not tailored) cosine and Hadamard transforms. Evaluations were performed over a range of coding rates, extending as low as 0.25 bits per pixel (bpp) and as high as 1.5 bpp. The best in each category were identified based upon a preliminary evaluation using a small set of test imagery. Next, four algorithms -- one SVD, one KLT and the cosine and Hadamard -- were comprehensively evaluated against a larger set of test imagery. This imagery included visible and IR aerial photographs and SAR imagery, all quantized to 8 bpp.

All four algorithms performed well on the test images at 1.5 bpp. The KLT and cosine algorithms had highest coding efficiency, whereas the Hadamard algorithm was most computationally efficient. Overall performance -- jointly considering both coding and computational efficiency -- was best for the cosine algorithm, which appeared to perform well all the way down to 0.5 and sometimes 0.25 bpp. Despite the intensive effort in developing the most efficient SVD coding algorithm possible, this approach was found to be inferior to the cosine transform coder.

1.2 Roadmap

The remainder of this report presents the algorithms developed under this study and the results of evaluations performed to compare these algorithms among themselves and against baseline algorithms. Section 2 begins by defining study objectives and scope. Section 3 presents an overview of the transform image coding approach employed by all the algorithms developed and tested. Section 4 then concentrates on the details of the KLT algorithm and Section 5 upon the SVD algorithms. Section 6 discusses the mechanism implemented to achieve rate equalization in all algorithms. Section 7 next presents evaluation results, Section 8 summarizes the study conclusions, and Section 9 lists references. A variety of technical details which support various aspects of coder algorithm development are presented in Appendices A through F.

2.0 PROJECT SCOPE AND OBJECTIVES

The intelligence community and the Air Force have, for several years, realized the important role that image compression will be required to play in various image exploitation and intelligence systems of the future. Full use of the potential of these systems implies a need to transmit and store enormous quantities of digital image data. As suggested in Figure 2-1, image compression (and associated decompression or reconstruction) will directly impact the utilization of these systems by bringing storage and transmission requirements within technologically feasible bounds of transmission and storage media.

The primary focus of this study was on the compression of single frame tactical imagery. Such imagery arises from a variety of imaging sensors, including those sensitive to visible, infrared, and microwave (radar) wavelengths. Applications typicaly include intelligence, reconnaisance, and strike assessment.

We differentiate the imagery for such applications from the TV-scan imagery normally associated with airborne scanners, trackers or target detectors/recognizers and used in weapon fire control. In our case, the imagery tends to be high resolution, with large area coverage, but with relatively long revisit times. This is in contrast to the TV-scan imagery which is typically of lower resolution and smaller field of view, but with revisits at video rates. The effect is that in this study we only exploited spatial information: temporal redundancy was not available for use in compression.

In the course of the study, we concentrated upon the image compression algorithms themselves, and not upon the particular implementation to specific transmission or storage applications. In particular, we did not take specific channel characteristics into consideration, but instead focused on the inherent performance properties of the various algorithms. We did, however, design and investigate algorithms for use over the range of compression ratios anticipated as characteristic of various transmission and storage channels of potential interest.

R02181

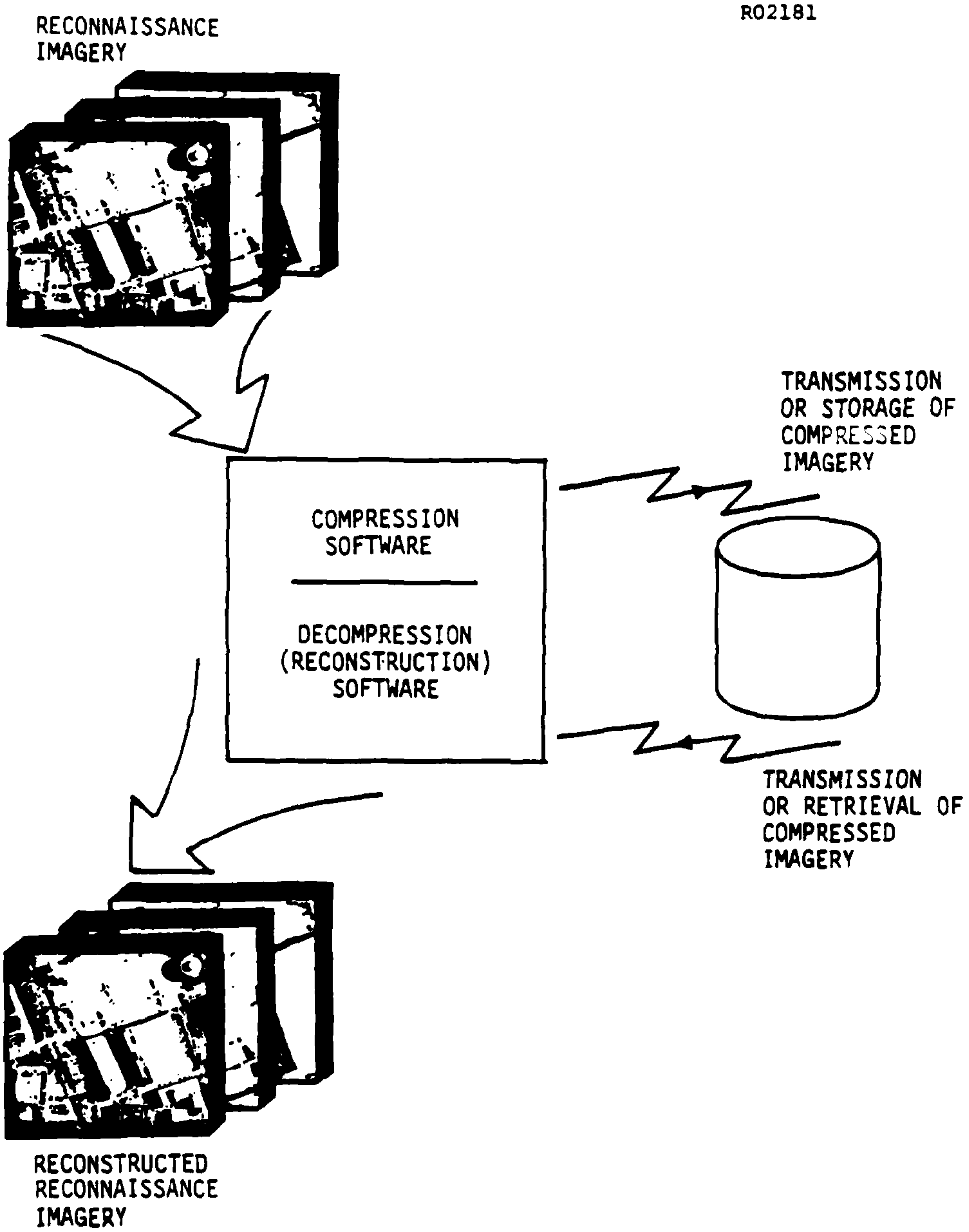


Figure 2-1. Automated Compression/Decompression of Tactical Imagery

Our primary evaluation tool involved rate distortion measures which describe coding efficiency in terms of the compression rates and image degradations that result from application of the various coders. A secondary measure was the computational efficiency associated with each approach.

Image compression can be veiwed as a coding process in which a compact representation of the image is extracted which is sufficient for subsequent viewing and analysis. The efficiency of the compression is defined by the amount of information (measured in number of bits) necessary for this purpose. One measure of this efficiency is the image <u>compression ratio</u>, defined as the ratio of the number of bits representing the original image to the number of bits in the coded representation. An alternative is the <u>compressed rate</u>, defined as the ratio of the number of bits in the coded representation to the number of pixels in the original image. These quantities are related as follows:

$$\text{compression ratio} = \frac{B_{orig}}{B_{coded}}$$

$$\text{compressed rate} = \frac{B_{coded}}{N \cdot M}$$

$$\text{orginal rate} = \frac{B_{orig}}{N \cdot M}$$

$$\text{compression ratio} = \frac{\text{orginal rate}}{\text{compressed rate}}$$

where

B_{orig} = number of bits in original image
B_{coded} = number of bits in coded representation
$N \cdot M$ = number of pixels in orginal

2.1 Algorithms

There are a variety of compression algorithms that can be applied to image data. In this study, attention was restricted to a class of particularly efficient techniques which involved use of two-dimensional linear transformations of image data prior to encoding. Prominent in this class are the well-known 2D cosine and Hadamard transforms which were included as baseline algorithms [1]. Image compression based upon these transforms is marked by both computational efficiency (due to the existence of specialized "fast" algorithms) and coding efficiency (due to good "energy compaction" properties). While the associated computational efficiency is a considerable advantage, the fact that these transforms are not specific to an image, or at least to a class of images, does suggest that these transforms produce less than optimal coding efficiency.

This study was concerned with developing and evaluating image transform coders employing transformations more tailored to image characteristics. Primary focus was on the singular value decomposition (SVD) operation, due to its known property of producing optimal energy compaction. Issues concerning both coding and computational efficiency were addressed and are reported in this document.

The price for the efficient energy compaction of the SVD is that not only the transform coefficients (singular values) themselves but also the transform operators (singular vectors) must be transmitted or stored in order to permit decoding. In order to reduce this load, averages over a number of similar images can be taken so that the operators are no longer image-specific, but rather class-specific. The result is the class-adaptive Karhunen-Loeve transform, which was also included in this study.

2.2 Imagery of Interest

The tactical image compression applications to which transform coders are targeted possess rather stringent compression requirements.

Overall compression ratios on the order of 30:1 or 60:1 are often necessitated. In order to achieve such ratios and still maintain useful image qauality in image regions of tactical significance, a selective compression algorithm is required. Such an algorithm employs priority designations of various image regions as, for example, "high interest", "low interest" or "background". Each such designation carries with it the requirement for a different level of compression. For example, "high interest" might require a compression ratio on the order of only 8:1, whereas "background" might have to be compressed down to 60:1 or so.

The idea here is that less important regions are assigned a greater share of the compression burden than are more important regions. The overall achievement of large compression ratios depends upon the predominance of less important ("background") regions within imagery. Fortunately, tactical imagery often has this characteristic [2].

In this study, we have concentrated on the more difficult to compress "high interest" regions of images. This is because it is on such data that transform approaches generally perform best, yielding the highest coding efficiency. Additionally, and perhaps more importantly, we focused on "high interest" image regions because it is the faithful rendition of such regions at the decoder that is the fundamental raison d'etre of tactical image collection, transmission and exploitation systems.

We have investigated the applicability of the various transform approaches to three types of imagery:

- Visible wavelength aerial photographs,
- Synthetic aperature radar imagery, and
- Infrared framing camera photographs.

All imagery was originally quantized to 8 bits per pixel. The range of compression ratios studied extended from 5:1 (1.5 bits per pixel) to 32:1 (0.25 bits per pixel). The nominal ratio used for comparison was 8:1 (1 bit per pixel).

2.2.1 Post Processing for Blockiness Suppression

A fundamental aspect of a transform coder is that it is applied to images in a block-by-block fashion. When such a coder is required to operate at high compression ratios, artifacts can appear at interblock boundaries. This blockiness occurs because the coder processes different blocks separately and because adjacent blocks often contain image data sufficiently different that when severe compression is applied, and these characteristics bloom out over the entire block, discontinuities are created at block edges.

This blockiness behavior is not restricted to transform coders, and in fact, has been observed in the operation of other compression algorithms as well. There are several fixes which are possible, all amounting to various restoration/enhancement schemes. For example, selective averaging across block edges can substantially reduce the visual impact of blockiness as well as the mean square degradation error [3]. Although developed for spatial domain implementation, such an approach also has an equivalent implementation in the transform domain, and could be integrated as a final post-processing step with any of the transform coders investigated under this study.

However, we have avoided such post-processing considerations, and have concentrated instead on the effect of coder algorithm operation alone. This permitted a cleaner assessment of coder performance, and enhanced our ability to isolate subtle image degradations introducted by various alterations in coder parameter values. Since such post-processing can always be added later, overall peformance of an eventual coder implementation based on these algorithms was not prematurely compromised. Introducing it at this early stage of algorithm development and evaluation, however, would have merely degraded our ability to assess algorithm performance.

3.0 OVERVIEW OF TRANSFORM IMAGE CODING

There are a number of extant approaches to compressing single frame imagery. Each approach represents a particular compromise among a set of conflicting goals, including:

- Maximize compression,
- Minimize degradation,
- Maximize adaptivity,
- Minimize encoder complexity, and
- Minimize decoder complexity.

3.1 Image Coding Approaches

Table 3-1 lists six categories of image coding approaches along with an example or two for each. The simplest is PCM (Pulse Code Modulation) which is simply a requantiziation of pixel intensities. Such an approach includes companding (COMpressing and exPANding), as well as adaptive versions that amount to digital automatic gain control. This approach is the least complicated to implement, and generally produces the least compression at a given level of distortion.

The next three categories -- predictive, transform, and interpolative/extrapolative -- attempt to exploit the spatial redundancies present in imagery. Predictive coding utilizes the observation that, in high resolution imagery, neighboring pixels tend to have similar intensity values. This information is used to encode only the differences between pixel values and estimates of these values predicted from previously encoded pixels. Since these differences tend to be smaller than the pixel values themselves, fewer bits are needed to encode them. A variety of versions, including schemes that are fixed

Table 3-1. Image Coding Approaches for Compressing of Single Frame Imagery

- PCM
  - Companding
- PREDICTIVE
  - DPCM
  - Delta Modulation
- TRANSFORM
  - Cosine
  - SVD
- INTERPOLATIVE/EXTRAPOLATIVE
  - Subsampling (e.g., MAPS)
  - Splines
- OTHERS
  - Contour
  - Bit Plane
- HYBRID
  - Cosine/DPCM

and adaptive and that are based on 1D and 2D prediction, are possible. This approach is computationally efficient and performs reasonably well on over-sampled digital imagery. However, it exploits only part of the spatial redundancy in the scene.

Transform approaches tend to perform best on high-resolution, moderate dynamic range, critically sampled imagery. This approach is based on dividing the image into blocks, performing a mathematical transform operation on each block, and encoding the resulting coefficients [1]. A number of transforms are available, including the Fourier, cosine, sine, Hadamard, Haar, slant, Karhunen-Loeve, and singular value decomposition. This set spans the spectrum of coding and computational efficiency. The fundamental idea involved in transform coding is to apply a transform which compresses the block information into a small number of coefficients which are then encoded in place of the larger number of pixel values themselves. This approach exploits 2D redundancy in the image, but only within the boundaries of individual blocks. Both fixed and adaptive versions are possible.

As an alternative to transform approaches, the interpolative/extrapolative approach attempts to fit curves to the two-dimensional surface defined by pixel intensity values. Then, only the parameters of the curves are coded. The simplest version uses piecewise constant curves, such as are generated by CDC's MAPS (Micro Adaptive Processing System) coder [4]. More ambitious approaches employ higher order splines [5]. The keys to the success of these types of scheme are their adaptivity to local image characteristics and their operation on imagery containing a high proportion of smooth areas, which thus permits parsimonious (low order and extensive in area) curve parameterizations.

The remaining approaches in the table are either specializations or combinations of the foregoing. For example, contour or bit plane coding is based on binary images, and the cosine/DPCM hybrid combines a 1D transform with a 1D predictive coder.

In this study, transform approaches were examined exclusively. New SVD and KLT algorithms were developed and compared with baseline cosine and Hadamard transform algorithms.

3.2 Transform Image Coding

Figure 3-1 illustrates the transform image coding chain used throughout the study. The first step involves extracting a block from the image, which accomplishes a reformating of the image from raster-scan into block ordering. Following this is the input intensity remapping step, which performs a memoryless transformation of the image to compensate for sensor and display system nonlinearites.

The next step is the application of the 2D transformation to the image block, creating an array of transform coefficients to replace the block of pixel values. This is where the different mathematical transforms are inserted into the chain.

After conversion to transform coefficients, actual encoding ensues. This is the step that performs the quantization and codeword assignment that constitutes the encoding of image information. It is the quantization part of this operation that is responsible for the deviations of a coded image from its original, by irreversibly degrading the image representation: the coarser the quantization, the greater the degradation (but the greater the compression). The trick is to perform this quantization efficiently, i.e., with the introduction of as little degradation as possible.

Next, the resulting codewords are reordered into 1D form and entered into the channel as a bit stream. Depending upon the application, the channel can take the form of a storage disk or magnetic tape, or a digital communication system for downlinking data from a sensor, for relaying to an exploitation center, or for dissemination to users.

B03181

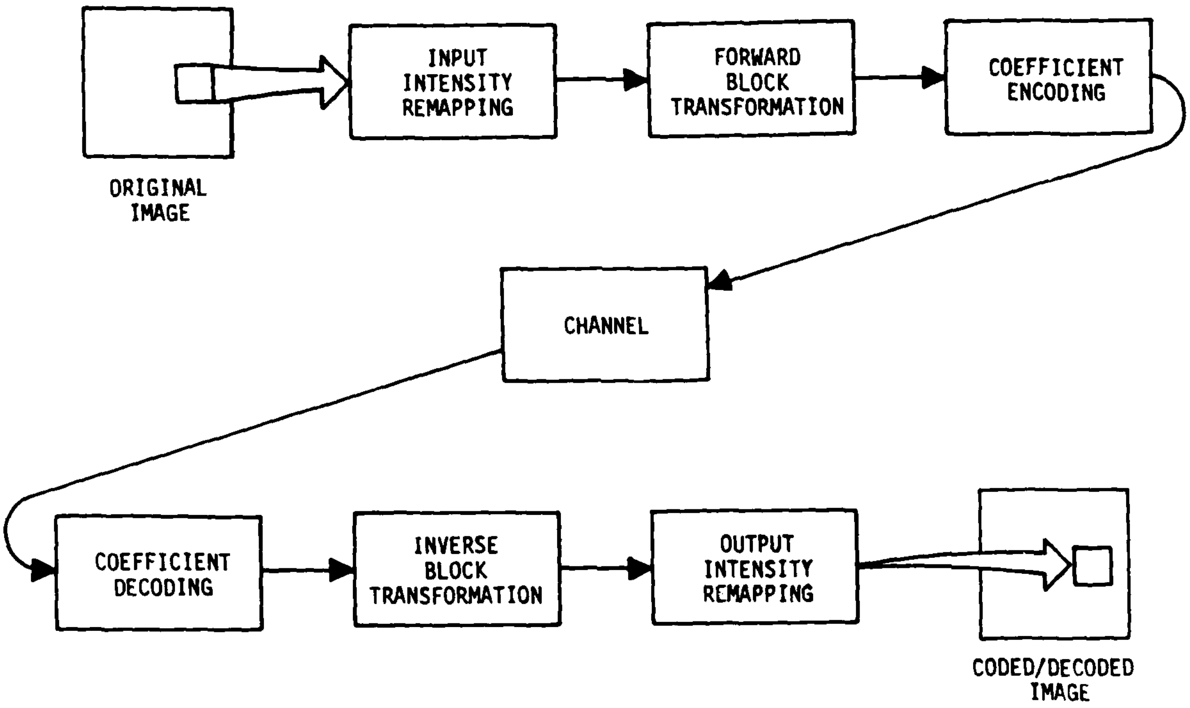


Figure 3-1. Transform Image Coding Chain

Whatever the application, it is the number of bits exiting the coefficient coder and entering the channel that describes the coder efficiency, either in terms of compression ratio, or, the measure preferred here, compressed rate (measured in bits per pixel).

Note that, invariably, the channel includes its own (channel) coder/decoder or modulator/demodulator (modem) which adds redundancy for error protection. Examples are the parity bits written onto tape or the burst error codes used in noisy communication systems. In any case, this redundancy is <u>excluded from the coder efficiency measures</u> employed in this report, i.e., we are describing <u>source coder performance</u> only. We are not concerned with channel coder performance, since the particular channel coder required in any situation is application-dependent.

The elements in the chain following the channel constitute the decoding operation and hence reverse the operation of the various steps applied before the channel. Coefficient decoding extracts the appropriate bit patterns from the bit stream, interprets them as codewords, and reconstructs the transform coefficients from the coded information. This reconstruction is not exact, however, due to the quantization error introduced during the the coefficient encoding operation. For this reason, the reconstructed coefficients are not identical to the original coefficients computed during encoding. They are, however, the best available estimates of these coefficients based on encoded data.

Next, the reconstructed coefficients are passed through the inverse transformation, producing reconstructed pixel values. Finally, an output intensity remapping is applied to match the gray scale output to the display system characteristics, and the block is re-inserted into the image in the appropriate location.

3.3 Block Transformations

There are two underlying reasons for applying a 2D transformation to a block of image data:

- To exploit spatial redundancy, and

- To concentrate information in a small number of coefficients.

The first of these reasons means that the correlation in intensity values of closely spaced pixels should be exploited. The objective is to generate a set of coefficients as uncorrelated as possible, with some of these describing gross image structure, some medium-sized features, and some fine detail. In this way, the degree of degradation introduced by quantization into any level can be accounted for separately. For example, since many images have many blocks with very little important fine detail, those coefficients can be neglected -- that is, not encoded -- with little loss of information. Additionally, the least mean square image degradation is produced in those cases where the coefficients are completely uncorrelated. This also motivates obtaining a transform which decorrelates pixels as much as possible prior to coefficient encoding.

The second objective concerns concentrating the block's energy into a small number of coefficients. In other words, the smaller the subset of coefficients that have appreciable size, the smaller the number of coefficients which must be coded for faithful image representation. But not only is the number of large coefficients important, so also is the consistency of their location within the coefficient array. Thus, transforms which consistently produce very small coefficient values in certain fixed locations permit having those coefficients consistently ignored by the coefficient coder.

3.4 Block Size

Image transform coding oeperates on images a block at a time, so that the question of appropriate block size immediately arises. There are several issues involved in selecting block size, since blocks with the following properties are required:

- Small enough for computational efficiency,

- Large enough for substantial decorrelation,

- Small enough for local adaptivity, and

- Power of two for "fast" algorithms.

The first of these objectives stipulates a reasonable block size for implementation. Both calculation time and storage space requirements grow with block dimensions. Consequently, it is necessary to keep these demands to a reasonable level. Based on experience with the 2D cosine transform, a maximum block size of 32 X 32 is indicated [6].

The objective of decorrelating pixels implies that blocks should be as large as possible, since a transform is only able to decorrelate pixels within a block. No decorrelation of pixels in distinct blocks is obtained. Based again on the 2D cosine transform, a minimum size of 8 X 8 is indicated for achieving appreciable decorrelation. (This finding is based on critically sampled imagery with a spatial correlation coefficient of approximately $\rho = 0.9$ [6].)

The third objective, for local adaptivity, implies that the block size should be small enough so that radically different image structure does not appear within the same block. The motivation for this requirement is based on cases where a small subregion of fine structure and, hence, high interest, is imbedded in an otherwise flat surround. If the busy subregion occupies too small a portion of the block, its effect on the transform coefficients is small with respect to that of the flat surround. Hence, the important coefficients are small and may therefore fail to be encoded accurately, if at all. Based again on the cosine transform and critically sampled imagery, objectives 2 and 3 -- for high decorrelation and local adaptivity -- balance each other out at a size of approximately 16 x 16 [6].

Since 16 is in fact a power of two, the block size used throughout the study for all algorithms developed and compared was 16 x 16. However, several notes are in order:

- Optimal block size may, in fact, vary with the particular transform employed. Sixteen by sixteen is indicated for the cosine transform, but was also adopted for the other transforms in order to provide a consistent basis for comparative performance evaluation.

- Optimal block size definitely depends upon the spatial sampling frequency. Sixteen by sixteen was predicated upon application to critically sampled raster imagery (i.e., at or near the Nyquist rate in each dimension). Significantly oversampled imagery would probably require larger block sizes to achieve the same degree of pixel decorrelation.

- There is no law requiring square blocks. In fact, past studies have indicated a degree of relative insensitivity to block aspect ratio, as long as the total number of pixels remains constant. Non-square blocks can arise naturally in imagery obtained from sensors utilizing non-square pixels (e.g., the common mod FLIR). We employed square blocks as a default, in the absense of reasons to adopt non-square blocks for the imagery of interest.

3.5 Unitary Transforms

Suppose we denote a block of image data by the symbol X. Based upon our adopted block size of 16 x 16, X represents a 16 x 16 matrix of pixel intensities. A linear transformation of X can be represented by:

$$Z = T(X)$$

where Z represents the transform coefficients collected into a second 16 x 16 matrix. The transformation T is linear, implying that

$$T(X_1 + X_2) = T(X_1) + T(X_2), \text{ and}$$
$$T(\alpha X) = \alpha T(X).$$

Linear transformations T are by far the most practical for image coding applications, due to their easy implementation with respect to general nonlinear transformations.

However, even restricting T to be linear does not guarantee a useful or easily implementable transformation. Further restricting T to be in the class of separable unitary transformations does, however. A separable, unitary transform has the following form:

$$Z = U^t X V$$

in which the coefficient array Z is obtained by premultiplication of the pixel array X by the matrix U^t, and postmultiplication by V. Furthermore, the transformation matrices U and V are unitary:

$$U^t U = U U^t = I$$

$$V^t V = V V^t = I$$

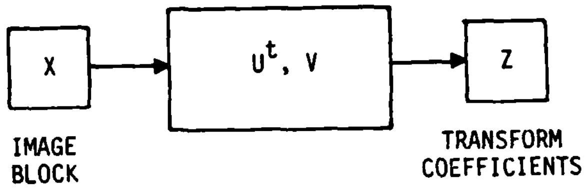
Figure 3-2 illustrates the structure of both the forward and inverse separable unitary transform.

The advantage of such a transformation is that it possesses the following characteristics:

- Column/row separable,
- Easy to invert, and
- Norm preserving.

Column/row separability obtains because the columns and rows of X are transformed separately: The U^t multiplication effects a column transformation while the V multiplication effects a row transformation. The result is that Z is obtained by applying $2n^3$ operations, where n

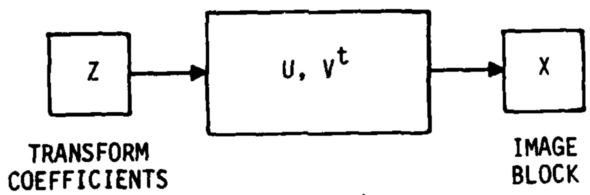
B03181



$$Z = U^tXV$$

U, V Orthogonal Matrices

Forward Transform



$$X = UZV^t$$

Because $U^{-1} = U^t$
$V^{-1} = V^t$

Inverse Transform

Figure 3-2. Unitary Block Transform

represents the sixe of X, i.e., X is n x n. This is a great savings over the n^4 operations required for a non-separable linear transformation.

Easy inversion obtains because both U and V are unitary and therefore their inverses are equal to their transposes:

$$\left(U^t\right)^{-1} = U$$

$$V^{-1} = V^t$$

Thus, both forward and inverse transformation entail the same amount of work and utilize the same operators U and V.

Norm preservation again is a consequence of the unitary character of U and V. What it implies is that energy calculations can be applied in either the pixel or coefficient domain. Specifically, if $Z = [z_{ij}]$ and $X = [x_{ij}]$, then:

$$\sum_{i,j=1}^{n} z_{ij}^2 = \sum_{i,j=1}^{n} x_{ij}^2 .$$

This property is extremely important in devising and analyzing coefficient coding schemes. For example, if z_{pq} is small and is neglected (i.e., not coded and then approximated by zero) the effect in the pixel domain can be predicted as a decrease in signal energy by z_{pq}^2.

The effect of a separable unitary transformation can best be explained by considering basis blocks. First adopt the notation:

$$X = [x_{ij}]$$

$$U = [\underline{u}_1\ \underline{u}_2\ \cdot\ \cdot\ \cdot\ \underline{u}_n]$$

$$V = [\underline{v}_1\ \underline{v}_2\ \cdot\ \cdot\ \cdot\ \underline{v}_n]$$

VERAC Incorporated

which depects the elements of the coefficient array X and the columns of the transformation matrices U and V. Then the inverse transform can be expanded as follows:

$$
\begin{aligned}
X &= UZV^t \\
&= \begin{bmatrix} \underline{u}_1 & \underline{u}_2 & \cdots & \underline{u}_n \end{bmatrix} \bullet \begin{bmatrix} z_{11} & z_{12} & \cdots & z_{1n} \\ z_{21} & z_{22} & \cdots & z_{2n} \\ \vdots & & & \\ z_{n1} & z_{n2} & \cdots & z_{nn} \end{bmatrix} \bullet \begin{bmatrix} \underline{v}_1^t \\ \underline{v}_2^t \\ \vdots \\ \underline{v}_n^t \end{bmatrix} \\
&= \sum_{i,j=1}^{n} z_{ij} \, \underline{u}_i \, \underline{v}_t^t
\end{aligned}
$$

Thus, the pixel block X is given as a weighted sum of rank one matrices $\underline{u}_i \underline{v}_j^t$.

Each rank one matrix $\underline{u}_i \underline{v}_j^t$ represents an elementary image block called a basis block. Together they constitute the fundamental components from which the overall X is constructed. In general, there are n^2 such basis blocks, which are weighted according to the corresponding coefficient values z_{ij} and combined to form X. The coefficient z_{ij} thus represents the strength of basis block $\underline{u}_i \underline{v}_j^t$ contained in X. If the basis blocks are known to the decoder, only the coefficient values z_{ij} need be encoded into the channel. The decoder can then reconstruct the image block X via an inverse transformation of Z via $X = UZV^t$.

3.6 Applicable Transformations

There are a number of separable unitary transformations which can be applied for image compression. These generally can be classed in one of three catagories:

- Fixed,
- Tailored to statistics, or
- Tailored to block itself.

Fixed transformations have received the greatest amount of attention for application to image coding. These comprise 2D extensions of familiar 1D unitary transformations and are characterized by fixed operators U and V. They include:

- Fourier,
- Sine,
- Cosine,
- Hadamard,
- Haar, and
- Slant.

The first three of these employ sinusoidal basis functions (i.e., the columns of U and V are sampled sinusoids), whereas the last three employ square wave, tertiary or triangular wave basis functions. A primary advantage of using the fixed type of transformation is its ease of implementation, often by a "fast" algorithm. The primary disadvantage is that these transformations are not sensitive to changes in local image characteristics, and so may work much better on some image blocks than on others.

The goal of adapting the transformation to local image characteristics motivates consideration of the remaining two tailored types of transformation. The first of these, which adjusts the operators U and V to local image statistics, is best represented by the Karhunen-Loeve transform, which is sensitive to second order block statistics. The second type of adaptive transform varies with the block data itself, and is best represented by the singular value decomposition, in which the U and V operators depend upon the image block X itself.

3.6.1 Optimal Decorrelating Transform

To better appreciate the interrelationships among these three transform types, a statistical viewpoint is helpful. In this viewpoint, an image can often be reasonably modeled as a sample from a spatially correlated, discrete random field. If the additional assumptions that the image is (spatially) stationary and Gaussian are included, Shannon theory indicates that optimal compression (least distortion for a given compression rate) can be achieved by first applying a decorrelating transform to convert the correlated pixels to a set of uncorrelated random variables, followed by encoding the resulting uncorrelated random variables with a memoryless coder.

The transform which is statistically optimal for decorrelating a block from a stationary image is the Hotelling, or discrete Karhunen-Loeve, transform. When the image has a separable covariance function, this transform takes the form

$$Z = U^t X V$$

where U and V are determined from the image covariance function. For this transform, Z is an array of completely uncorrelated random variables.

For two primary reasons, technical effort has historically been directed away from the optimal transform and focused instead on other transforms which only approximate the optimal decorrelating transform:

- No fast algorithm generally exists for performing the transform, and
- The procedure for deriving the Karhunen-Loeve transform involves potentially erroneous assumptions about the image model itself, resulting in difficulties with specific applications.

3.6.2 Cosine Transform

Historically, the first suboptimal transform to be considered was the discrete Fourier transform, in which U and V take the familar form of sampled complex sinusoids [7]. A prime motivation for using this transform is the fact that as the block size grows (again, under the stationarity assumption), the Fourier transform approaches the optimal transform in the mean-squared sense. Of more practical concern are the facts that the Fourier transform produces a (complex) coefficient array Z which is highly (though not perfectly) uncorrelated, and that a fast implementation (the FFT) exists.

However, a problem basic to use of this transform in coding is the Gibbs phenomenon, which results in severe artifacts near the edges of the compressed array, and thus introduces objectionable blocking in images that are block transformed. This latter problem can be eliminated by introducing a forced symmetry into the block, resulting in the cosine transform [8]. For this transform, U and V are sampled real sinusoids, and the coefficients Z are themselves all real. Because of its direct relationship to the Fourier transform, the cosine transform retains the Fourier transform's optimal asymptotic behavior, and is in fact superior to the Fourier transform for decorrelating smaller sized blocks. In addition, the FFT can still be used in actually executing the transformation.

A key property of the cosine transform which makes it particularly attractive for image compression is the energy compaction into the lower frequency coefficients that occurs for most images. Consequently, by concentrating on transmitting the larger magnitude, generally lower frequency coefficients, efficient coding with only slight loss of image energy is possible [9].

3.6.3 Hadamard Transform

The Hadamard transform is a binary approximation to the cosine transform that is characterized by unitary matrices U and V all of whose elements are either 1 or -1. An alternate interpretation is that the columns of U are sampled Walsh functions, so that this transform is also known as the Walsh transform.

The major advantage of the Hadamard transform is its ease of implementation. Not only are multiplications eliminated in forming the coefficient array Z, but a fast algorithm akin to the FFT also exists to speed execution. The price of this efficiency is a degradation in the decorrelational properties of the transform relative to the cosine transform. Even so, the transform does a fairly good job of decorrelating images and of compacting energy into the lower "sequency" coefficients of Z [10].

3.6.4 Singular Value Decomposition

Up to this point, the transforms discussed have been linear, separable and unitary, that is:

$$Z = U^t X V.$$

For stationary Gaussian images with separable covariance functions, theory indicates that this structure provides for efficient decorrelation of X into Z. However, for images which are nonstationary or non-Gaussian or which have nonseparable covariance functions, it is possible that a more general transform than that above could produce better results.

One such generalization is a nonlinear transform that is an image-adaptive version of those discussed above:

$$Z = U^t(X)\ X\ V(X)$$

where U and V are again unitary. Among transforms of this class, the best candidate in terms of energy compaction is the singular value decomposition (SVD):

$$Z = U^t X V$$

where

$$XX^tU = U\Lambda, \text{ and}$$

$$\Lambda \text{ diagonal}$$

$$X^tXV = V\Lambda.$$

In this case, $Z = \Lambda^{1/2}$ so that Z has at most n non-zero entries, in contrast to the n^2 entries of the optimal, 2D-cosine, and Hadamard transforms.

The major property of this transform relevant to image compression is that this choice of U and V yields a $Z(=\Lambda^{1/2})$ with maximum energy compaction. However, unlike the previous transforms, this U and V depend upon X, so that it is necessary to transmit not only $Z = \Lambda^{1/2}$, but also U and V. Consequently, it is perhaps better to represent this nonlinear image transformation as:

$$SVD(X) = (\Lambda, U, V).$$

Although there are altogether $2n^2 + n$ non-zero entries in the arrays Λ, U, and V, a degrees-of-freedon analysis indicates that a total of n^2 numbers--n for Λ and n^2-n for U and V together -- are sufficient to completely specify all three arrays.

3.6.5 Focus of New Developments

Primary attention in this study was aimed at further developing the SVD approach to image coding. A small amount of previous work using SVD's for image compression was reported in [11], but the results are preliminary and do not take into account the image statistics, the regularity of the singular vectors (columns of U and V), or the potential efficiencies that can be obtained by jointly considering the transform and memoryless coding processes. It was these aspects of SVD coding which were examined in the course of this study in developing an optimal SVD image coder.

In addition, recognition of the high overhead necessitated by singular vector coding prompted examination of a different approach which reduced this overhead by amortizing it over a number of image blocks. The mechanism for accomplishing this was the implementation of a coding scheme in which the U and V operators are specific to, instead of a single block of image data, a collection of such image blocks. Since this scheme amounts to a class-adaptive Karhunen-Loeve coder, such an algorithm was also developed for comparison.

4.0 CLASS-ADAPTIVE KARHUNEN-LOEVE TRANSFORM CODER

This section describes the class-adaptive Karhunen-Loeve transformation and the associated pre-processing and coefficient coding schemes employed with it under the study. Subsection 4.1 covers the transformation, Subsection 4.2 the preprocessing, and Subsection 4.3 the coefficient coder.

4.1 Class-Adaptive Karhunen-Loeve Transformation

The Karhunen-Loeve Transformation (KLT) is the method of expansion by principle statistical components. That is, it involves the representation of an image block X as a weighted sum of basis blocks B_{ij} which reflect statistically significant block characteristics. This representation takes the form

$$X = \sum_{ij} z_{ij} B_{ij}$$

where the z_{ij} constitute the KLT coefficient array Z. (This expression represents the inverse KLT operation.) The KLT coefficient array Z possesses two important properties:

- The elements z_{ij} of Z are uncorrelated, and
- The average energy compaction into the first few elements of Z is greater than that obtained from any other linear *transformation.*

4.1.1 The Separable Covariance Assumption

In order that the KLT be implementable as a separable operation on X, i.e.,

$$Z = U^t X V,$$

the basis blocks B_{ij} above must take the form of the outer product of two vectors, specifically:

$$B_{ij} = \underline{u}_i \underline{v}_j^t.$$

This situation obtains if it is assumed that the image covariance function is separable, i.e., if the correlation of the two pixels $x(i,j)$ and $x(i+\Delta i, j+\Delta j)$ depends not on the Euclidian separation $\sqrt{\Delta i^2 + \Delta j^2}$, but separately on the vertical separation Δi and the horizontal separation Δj. Mathematically, this can be written as

$$COV\ (\Delta i, \Delta j) = C_V(\Delta i) \quad C_H(\Delta j)$$

where C_V is the vertical image covariance and C_H is the horizontal covariance.

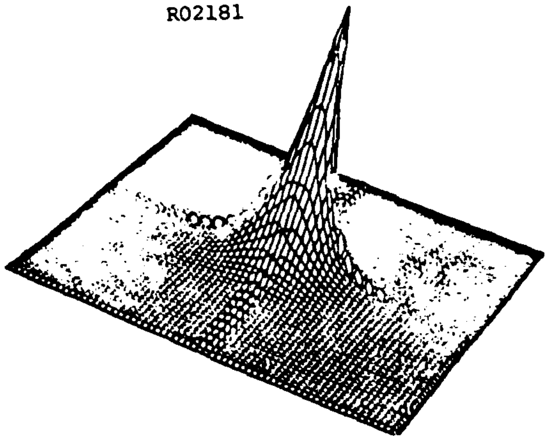
The significance of such an assumption is illustrated graphically in Figure 4-1, which shows a typical radially-symmetric image covariance function in part (a) and a separable approximation to it in part (b). The effect of the approximation is to over-accentuate image correlation vertically and horizontally and under-accentuate it at oblique angles. Thus, vertical and horizontal image structure can be expected to be retained somewhat more faithfully than oblique image structure when KLT coefficient coding is performed. However, the cost of implementing a KLT based on a non-separable image model is prohibitive (an order of magnitude more calculation). Consequently, we adopted the separable model for derivation of the KLT operators.

4.1.2 KLT Definition

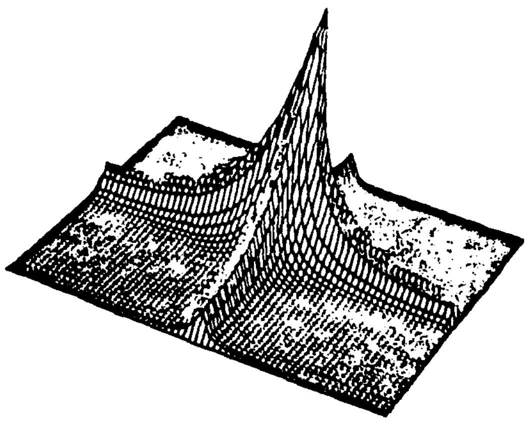
Based on the separable covariance assumption, it is shown in Appendix A that a separable, unitary KLT transformation takes the following form:

$$Z = U'XV$$

R02181



(a) Radially Symmetric Covariance



(b) Separable Covariance

Figure 4-1. Comparison of Separable and Non-Separable Image Covariances where the U and V operators are unitary matrices that depend upon vertical and horizontal pixel correlations, respectively. Specifically, the U and V matricies are pre-computed from pixel statistics according to the following pair of eigenvalue/eigenvector problems:

$$\left[\frac{1}{\sigma^2_n} C^{row}\right] U = U\Lambda^{row}, \text{ and}$$

$$\left[\frac{1}{\sigma^2_n} C^{col}\right] V = V\Lambda^{col}$$

where C^{row} and C^{col} are row and column covariance matrices, σ^2 is pixel variance, and Λ^{row} and Λ^{col} are diagonal matrices. Since the resulting U and V are unitary, $U^{-1}=U^t$ and $V^{-1}=V^t$ so that these problems can be rewritten as

$$U^t \left[\frac{1}{\sigma^2_n} C^{row}\right] U = \Lambda^{row}, \text{ and}$$

$$V^t \left[\frac{1}{\sigma^2_n} C^{col}\right] V = \Lambda^{col},$$

which shows that the effect of the operators U and V is to diagonalize the row and column covariance matrices. The result is that, in the KLT, U and V remove row and column correlations, respectively, from the pixel array X, producing an uncorrelated coefficient array Z. Maximum energy compaction into the z_{ij}'s with the smallest indices is achieved simply by ordering the columns of U and V so that the diagonal elements of Λ^{row} and Λ^{col} monotonically decrease from upper left to lower right.

4.1.3 Class Adaptivity

The success of the KLT depends upon having a good match between an image block X and its assumed statistics, summarized by the row and column covariance matrices. Since images are typically highly non-stationary, a multiple class KLT scheme was adopted here to better aid in employing the proper statistical assumptions at the proper time.

In this approach, a number of different pairs of row and column covariance matrices are included, each describing the statistical characteristics of a particular class of imagery. Then, whenever an image block of that class is to be transformed, the U and V matrices previously calculated from that class's statistics are employed in extracting the KLT coefficients.

Specifically, if a block X is determined to belong to class k, then the class k KLT is applied to X:

$$Z = U_k^t X V_k,$$

where U_k, V_k satisfy the following class k eigenvalue/eigenvector problems:

$$\left[\frac{1}{\sigma_k^2 n} C_k^{row}\right] U_k = U_k \Lambda_k^{row}, \text{ and}$$

$$\left[\frac{1}{\sigma_k^2 n} C_k^{col}\right] V_k = V_k \Lambda_k^{col} .$$

Because the inverse KLT is class-dependent,

$$X = U_k Z V_k^t,$$

it is necessary to encode not only the array Z, but also the class label k, so that the decoder can know how to properly inverse transform the coefficients Z it receives from the channel. For this reason, the number of classes is held to a reasonably small number, permitting efficient encoding of class information and resulting in near negligible overhead associated with block class encoding. In this study, eight classes were employed.

A key issue associated with class-adaptive coding is the mechanism for determining a block's class. Only if blocks can be easily and consistently separated into meaningfully distinct classes is the scheme useful. Properly, the problem of identifying meaningful classes and determining reasonable classification schemes is a problem in unsupervised pattern recognition.

Ideally, a number of block features would be examined to find the optimal class boundaries, and the feature extraction procedure and classification logic would be analyzed to determine the best tradeoff between accuracy of correct classification and computational expense. Instead, we adopted a block classifier based on the extraction of a single scalar feature known to be strongly correlated with the quantity of information contained in a block. We thus select our classes to roughly correspond to varying levels of block information content and, thus, difficulty of compression.

The feature employed in this study was block a.c. energy, defined as the mean square deviation of a block's pixel values from the average intensity value. That is:

$$\mu(x) = \frac{1}{n^2} \sum_{i,j=1}^{n} x_{ij}^2 - \left(\frac{1}{n^2} \sum_{i,j=1}^{n} x_{ij} \right)^2$$

The feature $\mu(x)$ is a good measure of bloc' "busyness" and for this reason provides a high correlation with block information content. In addition, since it is based upon nergy, and both U and V are unitar, μ can be calculated in either the pixel or transform coefficient domain.

Based upon this feature, a simple classifier of the following form was employed:

Block X is in class k IF $t_{k-1} \leq \mu(X) < t_k$

The decision points t_k were initially left unspecified, and an experiment was conducted to determine the best choice. During this experiment, which is detailed in Section 7 of this report, a uniform spacing of the t_k's in log (μ) space was indicated as best, and was adopted for all class-adaptive applications.

4.1.4 KLT Computational Algorithms

Three types of calculations are associated with the KLT:

- Determining transformation operators,
- Extracting KLT coefficients, and
- Reconstructing pixels from KLT coefficients.

The first type, involving construction of U_k and V_k for each class, amounts to the solution to 2k eigenvalue/eigenvector problems, where k represents the number of classes (8 in this study). Each of these problems entails the diagonalization of an nxn real, symmetric, positive semidefinite matrix. Since n=16 in this study, such problems can easily be solved by use of a conventional matrix calculation package such as LINPACK [12]. Since this calculation is off-line and precedes actual image coding, high efficiency is not required.

For the KLT, both forward and inverse transformations are performed by straightforward matrix multiplication:

$$Z = U^t X V \text{ and}$$

$$X = U Z V^t.$$

Thus, $2n^3$ multiplications and additions are required to extract KLT coefficients or to reconstruct pixels from coefficients.

In general, no "fast" KLT algorithm (akin to the FFT) exists, although under centain assumptions on the column and row covariance matrices, U's and V's corresponding to the sine transform can be generated. In this special case, the FFT can be used to effect both the forward and inverse transformations. However, the required assumptions to force this situation violate the philosophy of fitting the transformation to the naturally arising class statistics, which is the whole reason for including the KLT in this study. The cosine transform, which is very similar to, and, in fact, has been shown to be superior to the sine transform in a number of cases, is already included in the study for comparison, so including both would not illuminate any new performance possibilities.

4.2 KLT Preprocessing

KLT preprocessing entails the calculations of class-specific KLT operator matrices U_k and V_k and coefficient statistic matrices M_k and Σ_k from training data. The process is illustrated in Figure 4-2 and consists of three parts:

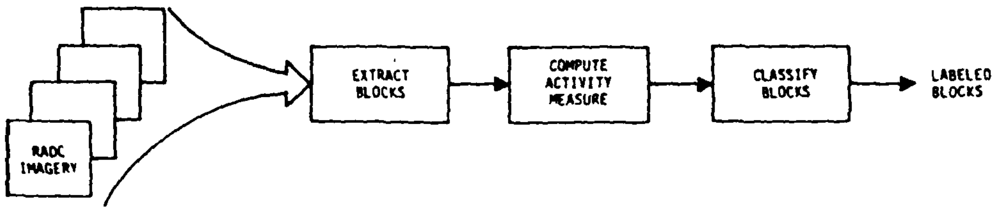
- Classify blocks of training imagery,
- Compute transform operator matrices and predicted statistics, and
- Collect empirical statistics.

4.2.1 Classify Blocks

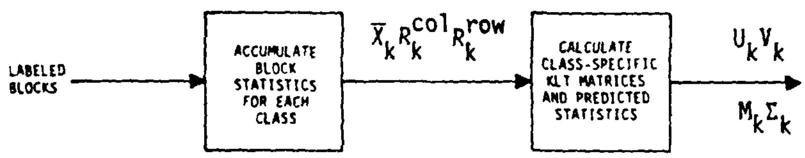
The block classification process involves the two steps discussed in subsection 4.1, namely:

- Compute activity measure $\mu(x)$, and
- Classify the block.

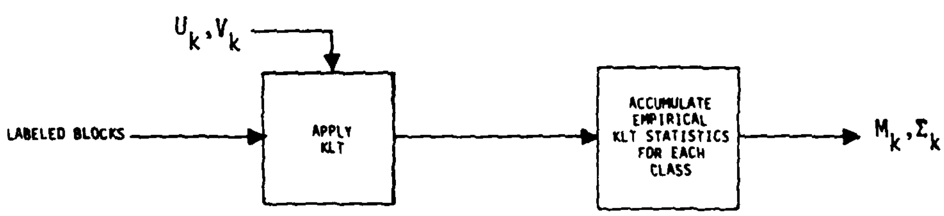
B03181



Classify Blocks of Training Imagery



Compute Transform Matrices and Predicted Statistics



Collect Emperical Statistics

Figure 4-2. KLT Preprocessing

The result is the appending of a label k to each block of training imagery. Since the KLT is class-adaptive, all further pre-processing to be discussed is class-specific, in the sense that all calculations are performed separately for all class 1 (k=1) blocks, all class 2 (k=2) blocks, etc.

4.2.2 Compute KLT Operator Matrices and Predicted Statistics

The class-specific U_k and V_k matrices are calculated from class-specific block statistics. Specifically, the following three block statistic matrices are computed for each clsss k:

- X_k = AVG [X in class k]
- R_k^{row} = AVG [XX^t in class k]
- R_k^{col} = AVG [X^tX in class k]

If these statistics are accumulated over a large number of blocks from a variety of imagery, they can be expected to converge to their proper values. However, whenever the training set if finite, residual structural artifacts may remain in the calculated statistics. To help smooth out these artifacts, the sample space of training imagery can be artifically expanded by the addition of new members synthesized from original members.

In particular, suppose $\mathcal{X}$ denotes the sample space of image blocks X obtained by partitioning the training imagery into nxn blocks. The set $\mathcal{X}$ can be expanded by any of the following schemes:

- Re-partition each image n^2 times, so that block boundaries shift around the image, causing a given pixel to occupy the various n^2 locations of a block exactly once. This eliminates artifacts due to block location within an image, and expands $\mathcal{X}$ by a factor of n^2.

- Flip each block vertically, horizontally and both. This eliminates certain artifacts due to the imaging system's orientation with respect to the scene, and expands $\mathcal{X}$ by a factor of four.

- Rotate each block 90°, then apply flips (good only for square blocks). This eliminates other artifacts due to the imaging system's orientation with respect to the scene, and expands $\mathcal{X}$ by a factor of four.

In this study, the second and third of these sample space enhancement schemes were employed for block statistics calculation. The first was omitted due to the extremely high computation and storage load associated with implementing it, and because the training set was reasonably large to begin with.

An additional structural artifact can be removed by introducing the homogenous mean assumption. That is, the block mean EX_k is assumed to be a matrix having all values equal to μ_k, i.e.:

$$EX_k = \begin{bmatrix} \mu_k & \mu_k & \cdots & \mu_k \\ \mu_k & \mu_k & \cdots & \mu_k \\ \mu_k & \mu_k & \cdots & \mu_k \end{bmatrix} = [\mu_k] = \mu_k \cdot [1]$$

where [1] is the nxn matrix all of whose elements are 1's. Since any deviation from this behavior is without physical justification, the assumption is introduced as a constraint to be satisfied during the sample mean calculation. This means that instead of determining $\bar{X}_k$ by elementwise averaging over the X's in class k, μ is calculated by averaging over all elements of all X's in class k:

$$\mu_k = \text{AVG}\left[\frac{1}{n^2}\sum_{i,j=1}^{n} x_{ij} \ : \ X \text{ in class } k\right]$$

Note: This can also be written as:

$$\mu_k = \frac{1}{n^2} \sum_{i,j=1}^{n} \bar{x}_{k,ij}$$

where $\bar{X}_k = [\bar{x}_{k,ij}]$, which is the way we actually implemented it.

Once $\bar{X}_r, R_k^{row}$ and R_k^{col} have been obtained, the required sample covariance matrices are computed from one of the following pairs of equations:

- Without homogenous mean constraint

$$C_k^{row} = R_k^{row} - \bar{X}_k \bar{X}_k^t$$

$$C_k^{col} = R_k^{col} - \bar{X}_k^t \bar{X}_k$$

- With homogenous mean constraint

$$C_k^{row} = R_k^{row} - \mu_k [1] \bar{X}_k^t - \mu_k \bar{X}_k [1] + \mu_k^2 [1][1]$$

$$C_k^{col} = R_k^{col} - \mu_k [1] \bar{X}_k - \mu_k \bar{X}_k^t [1] + \mu_k^2 [1][1]$$

From these matrices, the class-specific KLT operators U_k and V_k are obtained from (see Appendix A):

$$\left[\frac{1}{\sigma_k^2 n} C_k^{row}\right] U_k = U_k \Lambda_k^{row}, \text{ and}$$

$$\left[\frac{1}{\sigma_k^2 n} C_k^{col}\right] V_k = V_k \Lambda_k^{col}$$

where $\sigma_k^2 = \mathrm{tr}\ \Lambda_k^{row} = \mathrm{tr}\ \Lambda_k^{col}$.

For coefficient coding, the mean and standard deviation of the coefficients z_{ij} in the KLT array Z are required. In Appendix A, it is shown that under the assumptions employed in finding U_k and V_k, the class-specific coefficient statistics can be predicted as:

- Mean:

  -- without homogenous mean constraint

  $$M_k = U_k^t \bar{X}_k V_k$$

  -- with homogenous mean constraint

  $$M_k = \mu_k\ U_k^t\ [1]\ V_k$$

- Standard Deviation

  -- Variance

  $$S_k = \sigma_k^2 \underline{\lambda}_k^{row} \underline{\lambda}_k^{col^t}$$

  -- Standard Deviation

  $$\Sigma_k = [\sigma_{k,ij}]$$

  where $S_k = [\sigma^2_{k,ij}]$

4.2.3 Collect Empirical Statistics

Since it is recognized that the covariance separability assumption under which U_k and V_k are derived and coefficient statistics are predicted is not strictly valid, an alternative, empirical coefficient statistics calculation scheme is also employed. When the covariance separability assumption is valid, both approaches yield the same result. When the assumption is not valid, the second, empirical approach produces a more accurate estimate of the statistics of the coefficients actually produced by application of U_k and V_k.

Note that we are not talking about recalculating U_k and V_k under more general covariance assumptions. Rather, we are only dealing with obtaining a more accurate estimate of the average properties of the coefficients obtained by use of that U_k and V_k. The degree of disparity between the statistics calculated by the two methods indicates the degree to which the actual training data departs from the assumed separable covariance model.

Empirical statistics are obtained by applying the appropriate class-specific KLT to each block of training imagery, and accumulating statistics on the resulting coefficients. In particular, two items are required:

- Mean M_k, and
- Standard Deviation Σ_k

for each class k.

Calculation proceeds in two steps. First, the mean and mean square coefficient values are accumulated, then the standard deviations are derived from this data. The first step entails the following averages:

M_k = AVG ($U_k^t X V_k$: X is class k), and

$R_k = [r_{k,ij}]$

where $r_{k,ij}$ = AVG (z_{ij}^2: $Z = U_k^t X V_k$ and X is class k)

Next, the standard deviation array Σ_k is obtained by:

$$\Sigma_k = [\sigma_{k,ij}]$$

where $\sigma^2_{k,ij} = r^2_{k,ij} - m^2_{k,ij}$.

The various sample space enhancement techniques discussed under 4.1.2 are germane to empirical statistics calculation as well. However, rather than expand $\mathcal{X}$, the sample space of X's, directly, it is possible for the flip and rotation type of enhancements to expand $\mathcal{Z}$, the sample space of Z's, instead. This is of great practical benefit because of the large computational load associated with applying the KLT to so many blocks. Appendix B shows how the statistics of the expanded set can be calculated from the statistics of the original set $\mathcal{Z}$.

4.2.4 KLT Preprocessing Summary

To summarize, KLT preprocessing is a training procedure applied to a sample space of blocks obtained by appropriately partitioning a set of training imagery. The result is the generation of several class specific quantities:

- KLT operators U_k and V_k for each class,
- Predicted statistics of Z for each class, and
- Empirically collected statistics of Z for each class.

The U_k and V_k matrices are required to specify the class-adaptive KLT operation, while the coefficient statistics M_k and Σ_k are employed to efficiently code the coefficients produced by the KLT operator.

4.3 KLT Image Coding

Figure 4.3 depicts the KLT image coder employed in the study. The process begins by extracting an nxn block X from an image. The block is

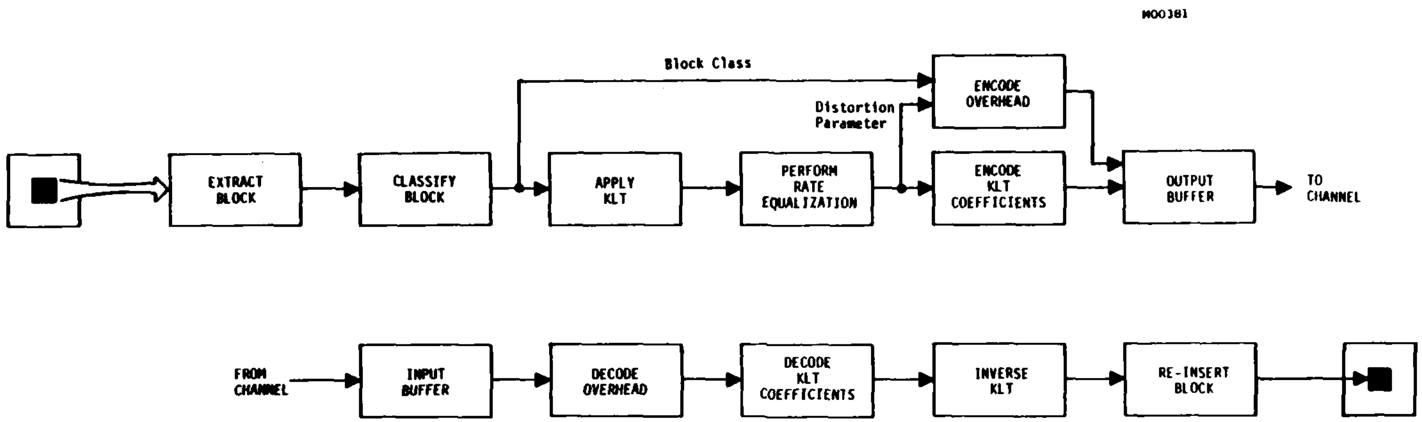


Figure 4-3. Image Coding Chain

classified by extracting the a.c. energy activity measure $\mu(X)$ and comparing the resulting value to a set of decision thresholds $\{t_k\}$. The result is a block label k. Based on this k, the proper KLT is applied to X using the appropriate U_k and V_k matrices.

The resulting KLT coefficients are then encoded, and placed into the output buffer for formation into a bit stream. Prior to this encoding, a rate equalization step occurs which is aimed at achieving a particular overall coding rate (e.g., 1 bit per pixel). This is achieved by computing a global distortion parameter D which serves to control coefficient coding by setting the fidelity level at which the coder is to operate. The rate equalization algorithm implemented for the KLT coder is essentially identical to that implemented for the SVD coder, and is discussed separately in Section 6.

In addition to the KLT operation itself, KLT coefficient coding is also a class-adaptive operation. This permits the allocation of relatively more channel bandwidth (number of bits) to high-information portions of the image than to low-information portions. This is implemented by generating more bits for "busy" (high activity measure μ) blocks than for "quiet" (low activity measure μ) blocks. The result is that bandwidth is adaptively allocated to the various blocks within an image. Class-adaptive KLT coefficient coding is disucssed in subsection 4.3.2.

The decoding operation is essentially the reverse of the encoding process. However, because the KLT operation is class-adaptive, the decoder must be provided with each block's class label in order to properly inverse-KLT the reconstructed coefficients into the reconstructed pixel block. Thus, the block labels k constitute overhead information which must be encoded and entered into the channel.

Similarly, the coder control parameter D must be available at the decoder in order for coefficient reconstruction to be properly performed. Thus, this parameter also constitutes overhead to be encoded and entered into the channel. Overhead coding is discussed next, in subsection 4.3.1.

4.3.1 Overhead Encoding

Since only a single D value need be specified for each image, and since only one of a small number of possible class labels need be specified for each block, overhead coding does not consume much channel bandwidth. Specifically, since, as indicated in Section 6, it is log(D) which controls coder fidelity, an 8-bit BCD log-quantizer was adopted for D. For k, which could assume one of eight values, a simple 3-bit BCD quantizer was employed.

The bandwidth resources consumed by encoding this overhead is slight. In particular, for 16 x 16 blocks and 256 X 256 images, the overhead is:

- Distortion parameter: 0.0001 bpp
- Class labels: 0.012 bpp.

Thus, total overhead to achieve both class adaptivity and rate equalization is slightly more than one hundredth of a bit per pixel. Since, for the high interest imagery under study here, overall coded rates on the order of one bit per pixel are of interest, the overhead associated with this scheme is, in fact, negligible.

4.3.2 KLT Coefficient Coding

The key aspect of the KLT coefficient coder is that it is class-adaptive. This adaptivity extends into two domains:

- Interblock adaptivity, and
- Intrablock adaptivity.

Interblock adaptivity refers to the distribution of total bandwidth among the various blocks in an image according to block class. High activity-index blocks contain more information and are thus allocated more bandwidth than are low activity-index blocks, which contain less information.

Intrablock adaptivity refers to the distribution of bandwidth among the various coefficients in a particular coefficient array Z according to their statistics. Coefficients with a high degree of predictability (e.g., usually small) are allocated less bandwidth than are coefficients with a low degree of predictability (i.e., can occur over a wide range of values).

Figure 4.4 illustrates an example of bit assignment arrays for two classes, one for low-activity blocks and the other for high-activity blocks. The arrays are to be interpreted as assigning the number of bits to be used in encoding the various $16^2 = 256$ coefficients within the array Z. Thus, the 3 in the $(i,j) = (3,2)$ position of the first array indicates that, for low-activity blocks, z_{32} is to be coded with a 3-bit quantizer.

The figure illustrates both types of adaptivity. Interblock adaptivity is indicated by the difference in the total number of bits allocated to all the n^2 coefficients, i.e., by the difference in the summations over all elements of each array. Intrablock adaptivity is illustrated by the preferential allocation of bits to those coefficients in the upper left hand corner of the arrays, corresponding to the coefficients which typically require the most dynamic range. Note that in both arrays a number of coefficients are allocated no bits at all, indicating they are to be ignored (not coded). These are the typically insignificant coefficients (approximated, for example, by zero).

Coefficient coding requires resolution of two issues:

- How to make coder assignments, and
- What quantizer to employ.

8	4	3	2	2	2	1	1	1	1	0	0	0	0	0	0
4	3	3	2	2	2	1	1	1	1	1	0	0	0	0	0
3	3	2	2	2	2	1	1	1	1	1	0	0	0	0	0
2	2	2	2	2	1	1	1	1	1	0	0	0	0	0	0
2	2	2	2	1	1	1	1	1	1	0	0	0	0	0	0
2	2	2	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Lowest Activity Class

8	8	8	5	6	4	4	2	3	1	2	0	0	0	0	0
8	7	6	5	5	4	3	2	2	1	1	1	0	0	0	0
8	6	6	5	5	4	4	3	2	1	1	0	1	0	0	0
5	5	5	4	3	4	3	3	3	2	2	0	1	0	0	0
6	5	5	3	3	3	3	2	2	2	2	2	0	0	0	0
4	4	4	4	3	3	1	2	2	2	1	1	1	0	0	0
4	3	4	3	3	1	2	1	1	0	1	0	0	1	0	0
2	2	3	3	2	2	1	2	1	0	1	0	0	0	0	0
3	2	2	3	2	2	1	1	0	0	0	1	0	0	0	0
1	1	1	2	2	2	0	0	0	0	0	0	0	0	0	0
2	1	1	2	2	1	1	1	0	0	0	0	0	0	0	0
0	1	0	0	2	1	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Highest Activity Class

Figure 4-4. Example Bit Assignment Arrays

4.3.2.1 Coder Assignment

Coder assignment amounts to constructing the bit assignment matrices shown in Figure 4-4. Since we are dealing with an eight-class situation, eight such matrices are required.

The criterion employed for coder assignment is the minimization of mean squared coding error at any given coding rate. Such a criterion results in a bit allocation rule which distributes the mean- squared error uniformly across all blocks, and, within a block, uniformly across all coefficients. To achieve such uniformity, such a rule must allocate more bits to high activity blocks than to low-activity blocks, and, within a block, more bits to strongly varying coefficients than to quiescent coefficients.

As shown in Appendix C, this criterion results in the following assignment rule:

$$B_{ij}(k) = INT \left| \log_2 \frac{\sigma_{ij}(k)}{D} \right|$$

where:

- $B_{ij}(k)$ = Number of bits allocated to the ij-th coefficient in class k blocks.
- $\sigma_{ij}(k)$ = Standard deviation of the ij-th coefficient in class k blocks. (This is the ij-th element of Σ_k the class-k coefficient standard deviation matrix. Either predicted or empirical values can be used.)
- D = Global distortion control parameter (determined to provide rate equalization).
- INT [.] = The integer part (required because we are using fixed rate quantizers).

To repeat, this rule produces adaptive bit allocation because it results in more bits being assigned to coefficients with high variability as measured by large σ_{ij}^2. Because high-activity blocks typically have many such coefficients, such blocks receive more bits in aggregate than do low-activity blocks.

4.3.2.2 Coefficient Quantization and Coding

Once a coefficient is allocated a number of bits for encoding, the next question is how to employ these bits in effectively encoding the coefficient. A number of possibilities exist, but the Max quantizer was selected here for its optimality properties. The key requirement for applying the Max quantizer is that the probability density functions of the z_{ij} be known.

The assumption applied is that all coefficients z_{ij} share the same form of probability density function, parameterized by mean $m_{ij}(k)$ and variance $\sigma_{ij}^2(k)$. Thus, the derived coefficients $(z_{ij}-m_{ij}(k))/\sigma_{ij}(k)$ all share the same zero-mean, unit-variance PDF $p(z)$.

For this study, we used a modified Gaussian function for $p(z)$. The Gaussian assumption is justifiable by the central limit theorem, and the modification, which slightly boosted up the tail of the distribution, was added to account for rare, but important events.

The Max quantizer consists of a set of quantizer decision thresholds and an associated set of reconstruction levels selected so that coding error is minimized on average. It results in a non-uniform quantization scheme that is tailored to the statistics of the coefficients. For example, the three-bit Max quantizer which minimizes mean squared coding error for a Gaussian PDF is shown in Figure 4-5. Max quantizers of 1,2, . . ., 8-bit were used in the study.

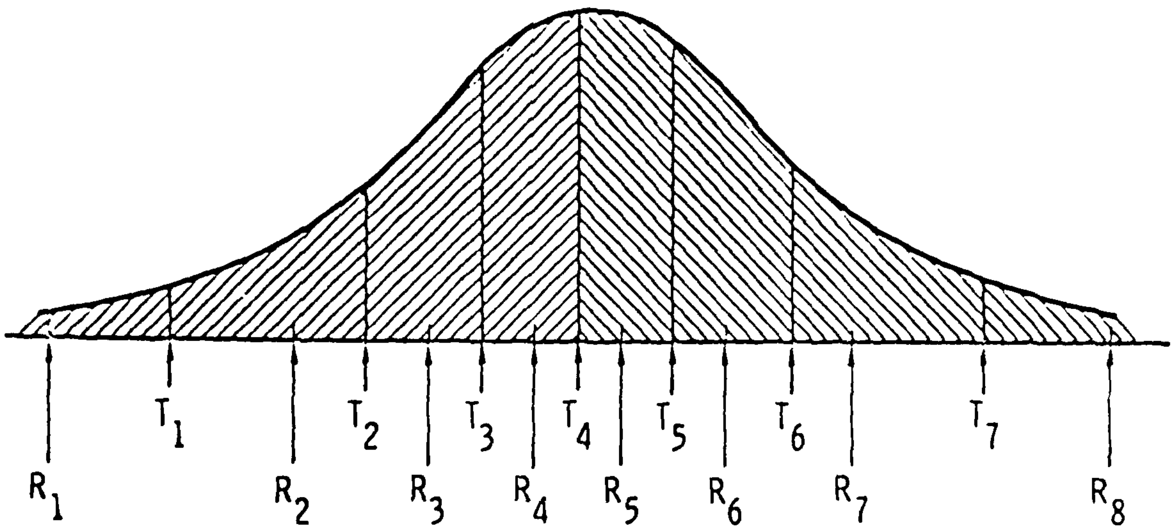


Figure 4-5. Example Three-Bit Max Quantizer for Gaussian PDF

Although a Max quantizer can be used on all coefficients, special treatment was provided the z_{11} coefficient. This is because the basis block $\underline{u}_1\underline{v}_1^t$ corresponding to this coefficient is invariably near constant in intensity over all pixels in the block, so that z_{11} is similar to the cosine or Hadamard "dc" coefficient.

The reason for special treatment for "dc" is that when using a Max quantizer -- even an 8-bit Max quantizer -- occasionally severe coding errors are committed. These errors arise where the coefficient deviates most from its assumed mean value, since there the Max quantizer bin-width is largest, and the potential difference between the actual coefficient value and its quantized (reconstructed) value is greatest.

Because "dc" errors are perceived as "blockiness" in the image, these errors are potentially more perceptually damaging than are similar errors encountered for a.c. coefficients. Thus, in place of a Max quantizer, a uniform quantizer was applied to the d.c. coefficient.

5.0 SINGULAR VALUE DECOMPOSITION TRANSFORM CODER

The singular value decomposition transform coder uses a transform constituting the method of principal deterministic components. In this method, each image block is again decomposed into a sum of unit norm basis blocks, but here the decomposition achieves the optimal energy compaction for each and every block, rather than merely on average as is the case for the KLT. This means that, here, the fewest number of coefficients of any decomposition is required for efficient image coding. However, in contrast with the statistical approach where the transform matrices are pre-computed and thus available to both coder and decoder, here the transform matrices themselves depend upon the image block and hence must themselves be coded along with the coefficients.

The transformation used in this approach is the singular value decomposition (SVD), given by:

$$S = U^t X V$$

where $XX^t U = U\Lambda$, where Λ is a diagonal array of non-negative elements and U is orthogonal;

and $X^t X V = V\Lambda$, where Λ is the same diagonal array of non-negative elements and V is orthogonal;

and where $S = (\Lambda)^{1/2}$.

The matrix S is diagonal and contains the singular values. The matrices U and V have as their columns the left and right singular vectors of X respectively. Because U and V depend upon X, all three matrices -- S, U, and V -- must be coded.

Several aspects of image coding using the SVD were explored:

- Computational algorithms,
- Further decorrelation and energy compaction, and

- Singular value/vector coding.

The overall coding chain is indicated in Figure 5-1. It is very similar to the KLT approach, with the major exceptions being:

- The SVD automatically adapts to the block X: it is not class-specific, and

- both singular values and singular vectors are coded.

As in the KLT approach, coefficients (singular values and vectors) are coded class-adaptively. Since the SVD is a unitary transformation, delaying the extraction of the activity measure $\mu(X)$ until after the forward SVD operation has no effect on the result of the classification process. In fact, the computational load of calculating $\mu(X)$ is less here due to the high energy compaction produced by the SVD transforms, as reflected in the diagonal structure of Z.

The remainder of this section presents the details of the SVD image coder. Subsection 5.1 discusses computational algorithms; subsection 5.2 summarizes additional steps potentially yielding further energy compaction or decorrelation of singular values/vectors; subsection 5.3 describes the preprocessing required to support SVD coding; and subsection 5.4 presents the new schemes developed for singular value/vector coding.

5.1 SVD Computational Algorithms

Several candidate algortithms for calculating the SVD were identified during the study. One is equally applicable for computing the KLT matrices during KLT preprocessing, and was, in fact, applied for that purpose. (Since KLT preprocessing is an initial, off-line training procedure, computational efficiency is not an issue there.)

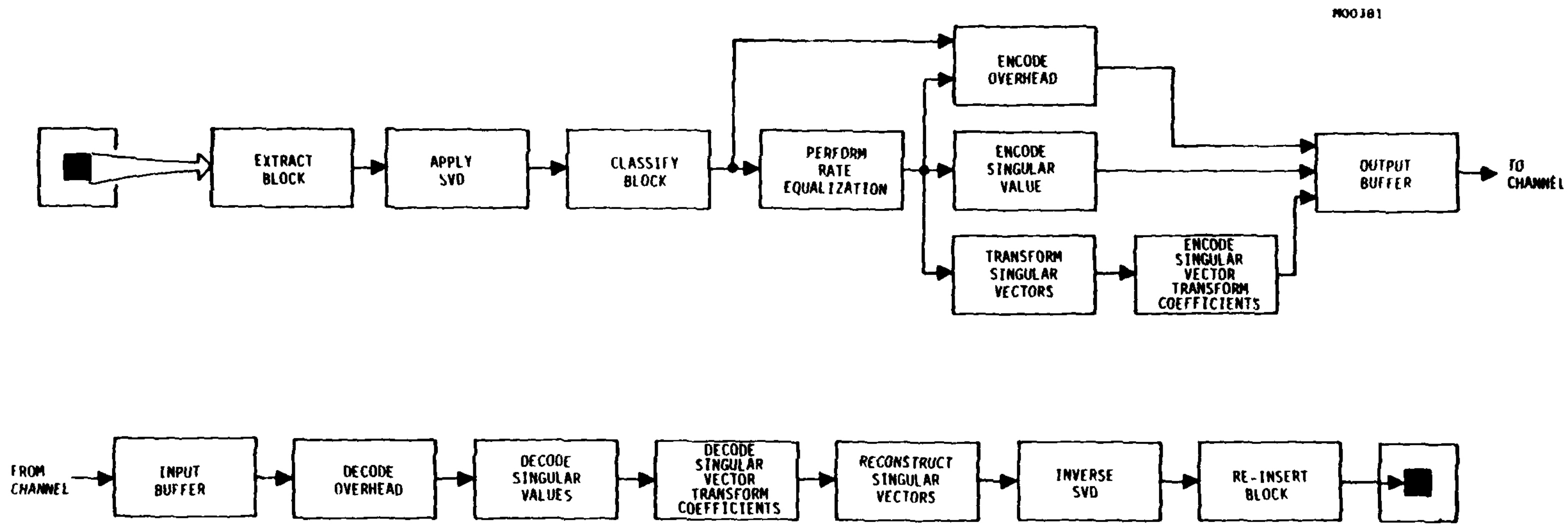


Figure 5-1. SVD Coding Chain

VERAC Incorporated

Two classes of SVD algorithms were identified, the direct and indirect methods. In the direct method, a block X is decomposed by directly searching for orthogonal matrices U and V such that

$U^tXV = S$, S diagonal and non-negative.

In the indirect method, the intermediate, symmetric positive semi-definite matrix XX^t or X^tX is first computed and its eignvalues and eignvectors calculated as:

$(X^tX)U = U\Lambda$, or

$(X^tX)V = V\Lambda$,

in which U and V are orthogonal and Λ is diagonal and positive semi-definite. In fact, $S = \Lambda^{1/2}$, i.e.,

$\Lambda = S^tS = SS^t$.

Whichever of U or V is calculated from the eigenvalue/eigenvector problem, the other is obtained directly from:

$V = X^tU\Lambda^{-1/2}$, or

$U = XV\Lambda^{-1/2}$,

in which $\Lambda^{-1/2}$ is a diagonal matrix having elements which are the reciprocal of the corresponding elements of Λ when non-zero, and zero otherwise. In this way, only those columns of V or U which correspond to non-zero singular values are obtained (they are the only ones needed).

Each type of SVD computation method, direct and indirect, can be tailored separately to two types of array X, a block of pixels and a block of transform coefficients, resulting in the four algorithms examined. (The indirect pixel block method is also used for computing the class adaptive KLT matrices by replacing X^tX and XX^t by $E(X^tX)$ and $E(XX^t)$.)

Pixel Block SVD Algorithms

The approach is illustrated in Figure 5-2 and consists of two steps:

- Apply a fixed number of Householder transformations U_i and V_j to render

  $$\tilde{X} = U_n^t \, U_{n-1}^t \cdots U_1^t \, XV_1V_2 \cdots V_n$$

  bi-diagonal, and then

- Iteratively apply a sequence of plane rotations, $\tilde{U}$ and $\tilde{V}$, to the rows and columns of X in order to implement the implicity shifted QR algorithm and to thus render the resulting

  $$\tilde{U}^t \, \tilde{X} \, \tilde{V}$$

  diagonal. This results in

  $$U = U_1U_2 \cdots U_n\tilde{U},$$

  $$V = V_1V_2 \cdots V_n\tilde{V},$$

  and $S = U^tXV$.

The indirect pixel block SVD algorithm is illustrated in Figure 5-3 and consists of an iterative application of Jacobi Transformations to diagonalize the symmetric matrix XX^t or X^tX. Suppose XX^t is to be diagonalized. Then a sequence of Jacobi Transformations U_i are applied to yield a diagonal matrix

$$\Lambda = U_L^t \cdots U_2^t(XX^t)U_1U_2 \cdots U_L$$

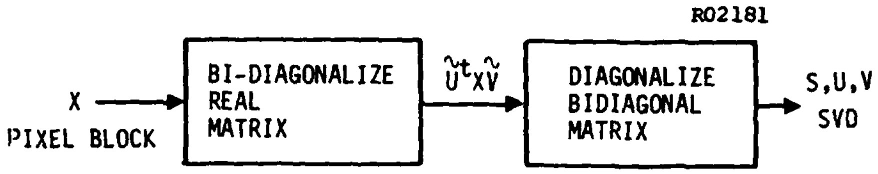


Figure 5-2. Direct Pixel Block SVD Calculation

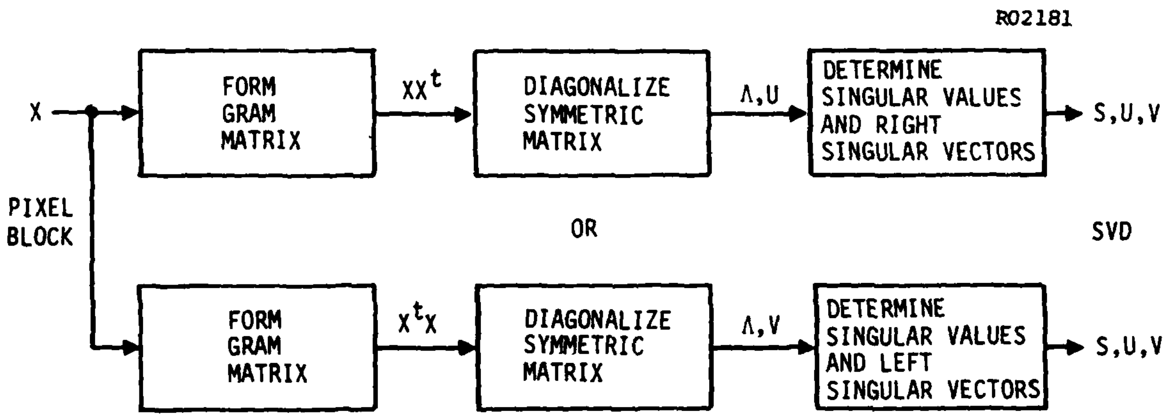


Figure 5-3. Indirect Pixel Block SVD Calculation

The transformation U_i is selected to reduce the largest off-diagonal elements remaining at the i-th iteration.

This results in

$$U = U_1 U_2 \cdots U_L,$$

$$S = \Lambda^{1/2}, \text{ and}$$

$$V = X^t U \Lambda^{-1/2}.$$

Both of these algorithms are generic in that they are applicable to extracting the SVD of any array X. They were included as baseline techniques against which to compare the more tailored coefficient block SVD algorithm and, for the indirect method, as a means of extracting the KLT matrices.

Coefficient Block SVD Algorithms

The coefficient block SVD approach attempts to exploit prior knowledge about typical image blocks. For example, from knowledge that pixels are non-negative and highly correlated arises the fact that one left and one right singular vector must be close to uniform (vector's of all 1's before normalization). Thus, pre-transforming by a $\tilde{U}$ and $\tilde{V}$ which each include such a column should render $\tilde{U}^t X \tilde{V}$ closer to diagonal.

In addition, the known regularity which often occurs in image block singular vectors can be anticipated by including appropriate columns in the pre-transforms $\tilde{U}$ and $\tilde{V}$. The result is a matrix $\tilde{X} = \tilde{U}^t X \tilde{V}$ which is more nearly diagonal than is X. This can be exploited in more easily completing the diagonalization process. The overall procedure is shown in Figure 5-4.

The pre-transforms employed here include the 2D cosine and Hadamard. Both include a uniform column in $\tilde{U} = \tilde{V}$ and both tend to mimic typical singular vector structure.

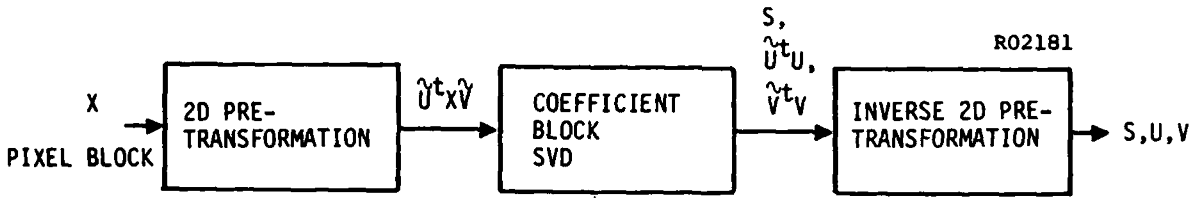


Figure 5-4. Finding the SVD via a Coefficient-Block SVD Calculation

Not only do such pre-transforms accelerate the diagonalization of X, but they also dovetail nicely with the singular vector coding approaches developed in the study. These approaches are discussed in detail elsewhere, but they amount to taking 1D correlating transforms (e.g., cosine or Hadamard) of the singular vectors of X and encoding the resulting coefficients. If such a singular vector coding approach is employed in conjunction with a pre-transform intended to ease SVD extraction, a particularly convenient synergism occurs. This is because the normally required steps of backing-out the pre-transform to find the singular vectors of X, followed by the application of a decorrelating 1D transform to these singular vectors to prepare for coding, can be eliminated. In particular, if the pre-transform is the 2D version of the 1D decorrelating transform (e.g., 2D cosine and 1D cosine) the combination of inverting the 2D pre-transform and applying the 1D transform cancel each other out. This is illustrated in Figure 5-5.

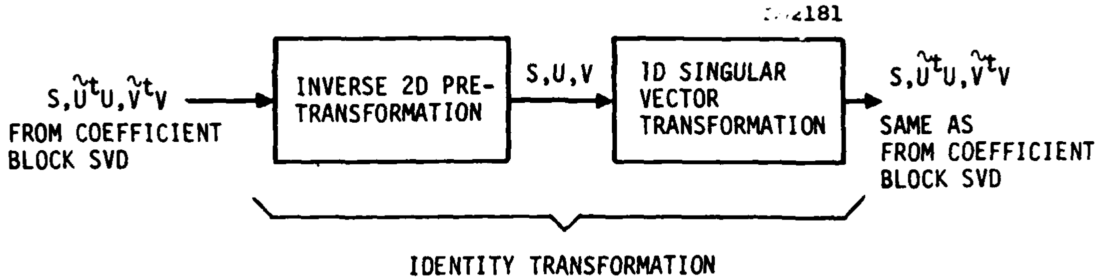


Figure 5-5. Collapsing the Inverse 2D Pre-transformation and the Forward 1D Singular Vector Transformation into the Identity

The result is an efficient algorithm for extracting both the singular values and the 1D transform of the right and left singular vectors of X. The process is illustrated in Figure 5-6. This was the procedure utilized during the evaluation phase of the study.

Both direct and indirect coefficient block SVD algorithms are possible and utilize a sequence of orthogonal transformations to diagonalize the appropriate matrix. In the direct case, transformations U_i and V_i, are applied until

$$U_L^t \cdots U_1^t \, (\tilde{U}^t X \tilde{V}) \, V_1 \cdots V_L$$

is approximately diagonal. In the indirect case, either the U_i's or V_i's are applied to diagonalize

$$U_L^t \cdots U_1^t \, (\tilde{U}^t X X^t \tilde{U}) \, U_1 \cdots U_L$$

or

$$V_L^t \cdots V_1^t \, (\tilde{V}^t X^t X \tilde{V}) \, V_1 \cdots V_L .$$

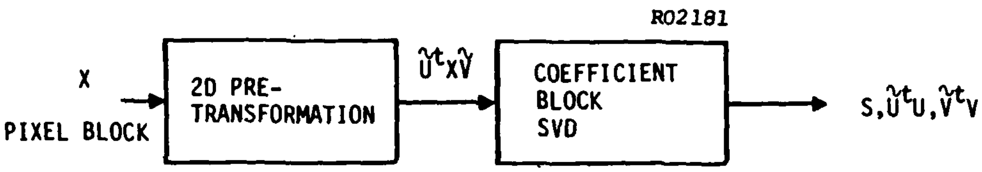


Figure 5-6. Combined Algorithm: Efficient Extraction of Both Singular Values and 1D Transform Coefficients of Singular Vectors

The other is obtained by substitution as in the indirect pixel block algorithm.

The indirect method was adopted here due to the availability of existing code to implement it, and, as discussed in Section 7, because SVD coder performance turned out not to be good enough to warrant a thorough investigation of the relative merits of the other SVD computational approaches.

5.2 Decorrelation and Energy Compaction of SVD Coefficients

Prior work utilizing the SVD transformation for image compression recognized the statistical correlations that typically occur within singular vectors [11]. In that work, a predictive coding scheme based on DPCM coding of singular vectors was employed to exploit the correlation. However, it is well known that such an approach only removes some of the correlation. Efficient encoding demands that as much correlation as possible be removed, which suggests applying a 1D transform to the singular vectors to decorrelate them.

The optimum transform for decorrelating a vector is the Karnunen-Loeve transform. However, for maximum adaptibility to non-stationarity, a class-aptitive, singular vector-specific 1D KLT is best. In such a scheme, the particular KLT operator applied would depend upon the block's class and the singular vector's index (location within U or V as appropriate). For our case, we have eight classes and sixteen left and sixteen right singular vectors requiring a total of 8• 16•16 = 2048 16x16 KLT operator matrices. Even in the case where statistical distinctions between left and right singular vectors are ignored, 8•16 = 128 such matrices are required.

Such a storage load, coupled with the computational load required to perform the KLT extraction via matrix multiplications, suggests that suboptimal transformations possessing a "fast" implementation be investigated. This was also indicated in order to efficiently combine the 2D pre-transformation discussed in Section 5.1 with the 1D singular vector transformation. Thus, we investigated two 1D transforms for singular vector decorrelation:

- Cosine, and
- Hadamard.

The first was included because of its known success at approximating the KLT's optimal decorrelating performance. The second was included due to its particular computational efficiency.

As an example of the effect of applying such a transform, Figure 5-7 shows some results for the cosine transform case. The first plot shows an example singular vector, in this case, the third left singular vector from a particular image block. The second plot shows the corresponding 1D cosine transform coefficients. Note the correlation from element to element in the singular vector and both the lack of such

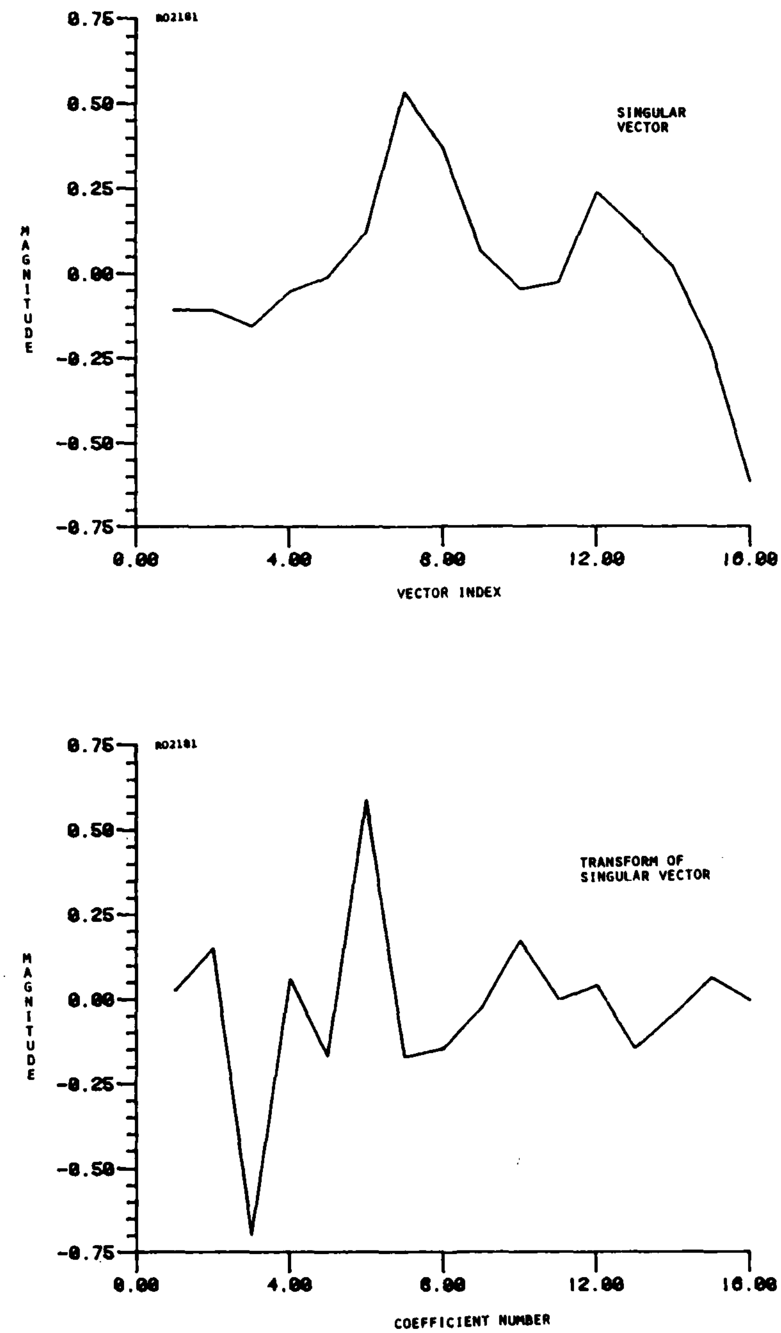


Figure 5-7. Example Singular Vector and its Cosine Transform

correlation and the occurrence of energy compaction in the transform as evidenced by the appearance of some coefficients with significantly larger magnitude than others.

That this behavior is even more striking on average is shown in Figure 5-8. In this figure, the first plot illustrates the mean and standard deviation obtained by accumulating over all third singular vectors. The corresponding quantities for transform coefficients are shown in the second plot. The marked peak in the second plot confirms the energy compacting property of the transform. When coding, coefficients corresponding to such peaks will be more accurately coded than will other, less important coefficients.

Figure 5-9 illustrates the two alternative implementations of the 1D singular vector transformations. The first approach is the straightforward one, in which the SVD is first calculated and the 1D transform of the resulting singular vector is then obtained. The second is the combined algorithm of Figure 5-6 which permits coordination with computation of the SVD itself. The equivalence is demonstrated by

$$S = U^t X V$$

$$\therefore X = U S V^t$$

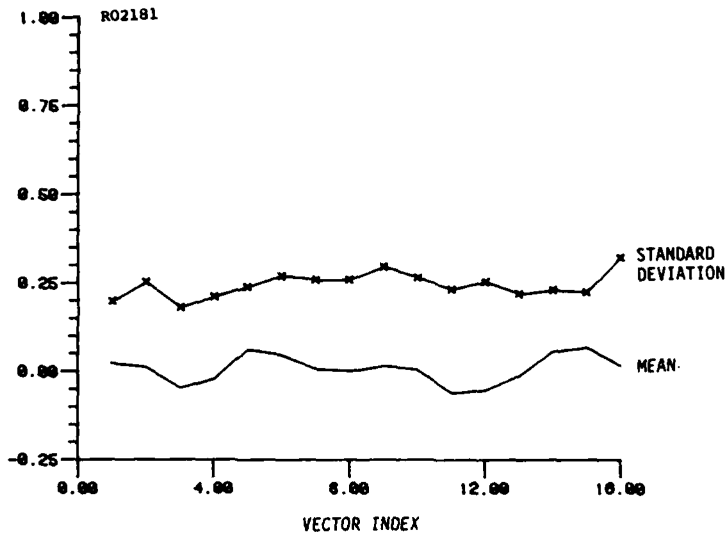
$$\tilde{U}^t X \tilde{V} = \tilde{U}^t U S V^t \tilde{V}$$

$$= (\tilde{U}^t U)\, S\, (\tilde{V}^t V)^t$$

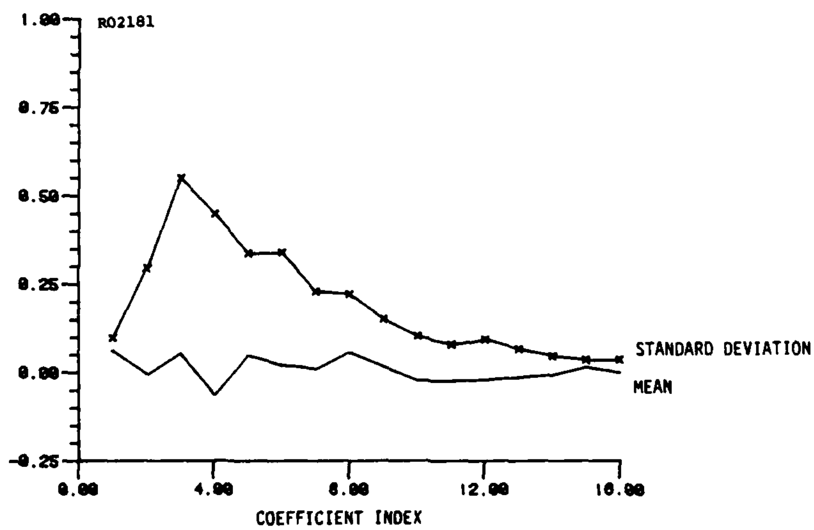
$$\therefore \quad S = (\tilde{U}^t U)^t\, (\tilde{U}^t X \tilde{V})\, (\tilde{V}^t V)$$

which shows that if $\{S,U,V\}$ constitute the SVD of X, then $\{S,\tilde{U}^t U,\tilde{V}^t V\}$ constitute the SVD of $\tilde{U}^t X \tilde{V}$. Thus, $\tilde{U}^t U$ and $\tilde{V}^t V$ can be obtained in either of two ways:

- Find SVD of X, then take the 1D transform of the columns of U and V, or



SINGULAR VECTOR STATISTICS



SINGULAR VECTOR TRANSFORM
COEFFICIENT STATISTICS

Figure 5-8. Example Singular Vector Statistics

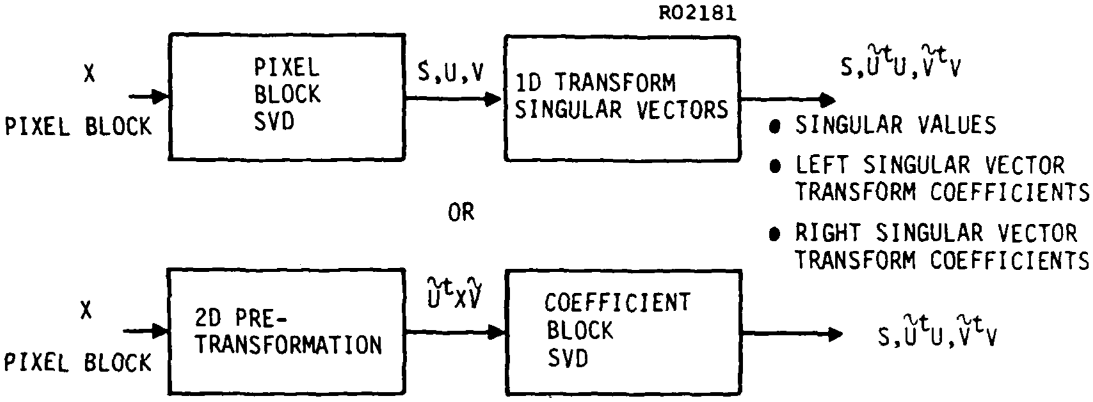


Figure 5-9. Two Implementations for Extracting
Singular Values and Singular Vector Transform Coefficients

- Find the SVD of $\tilde{U}^tX\tilde{V}$ directly.

In addition to the use of the 1D singular vector decorrelating transformation, three other techniques were identified as potentially useful for either additionally decorrelating the elements of S, U, and V, or for introducing further energy compaction. These techniques are:

- SVD reordering,

- Singular vector orthogonalization, and

- Repolarization of singular vectors.

Eacn of these techniques is described in turn.

5.2.1 SVD Reordering

The SVD forward transform takes the form

$$S = U^t X V,$$

wnere S is diagonal. The inverse transform takes the form

$$X = USV^t$$

which, because S is diagonal, can be rewritten as

$$X = \sum_{i=j}^{n} s_i \underline{u}_i \underline{v}_i^t$$

which expresses X as a weighted sum of n (not n^2) basis blocks $\underline{u}_i \underline{v}_i^t$.

There is no inherent ordering to the terms in this expression. In fact, permuting the ordering merely results in permuting the corresponding columns of U and V, and diagonal elements of S.

The normal default ordering is usually selected to result in s_i's with monotonically decreasing size, i.e., monotonically decreasing $|s_i|$. However, statistical analyses conducted under this study suggest that the singular vectors ordered in this way also typically have their strongest energy concentrated at monotonically increasing frequencies (or sequencies).

Figure 5-10 illustrates the migration of this energy concentration to higher frequencies for three singular vectors. The first singular vector has most of its energy concentrated at lowest frequencies. The third singular vector has most of its energy concentrated at somewhat higher frequencies. The twelfth singular vector has most of its energy at still higher frequencies.

This observation suggests that if ordering were performed. explicitly in terms of where singular vectors have peak energy instead of in terms of which singular values are largest, a very similar ordering would result. However, the ordering may be different often enough that when statistical averages are taken, even sharper peaking of average coefficient energy would result. This would then permit even more efficient encoding of singular vectors, since their energy would be, on average, more predictably concentrated is certain, known coefficients. There would be a concomitant decrease in the peakyness in the singular value statistics, but this would probably be more than compensated for by the increased peakyness of singular vector coefficient statistics.

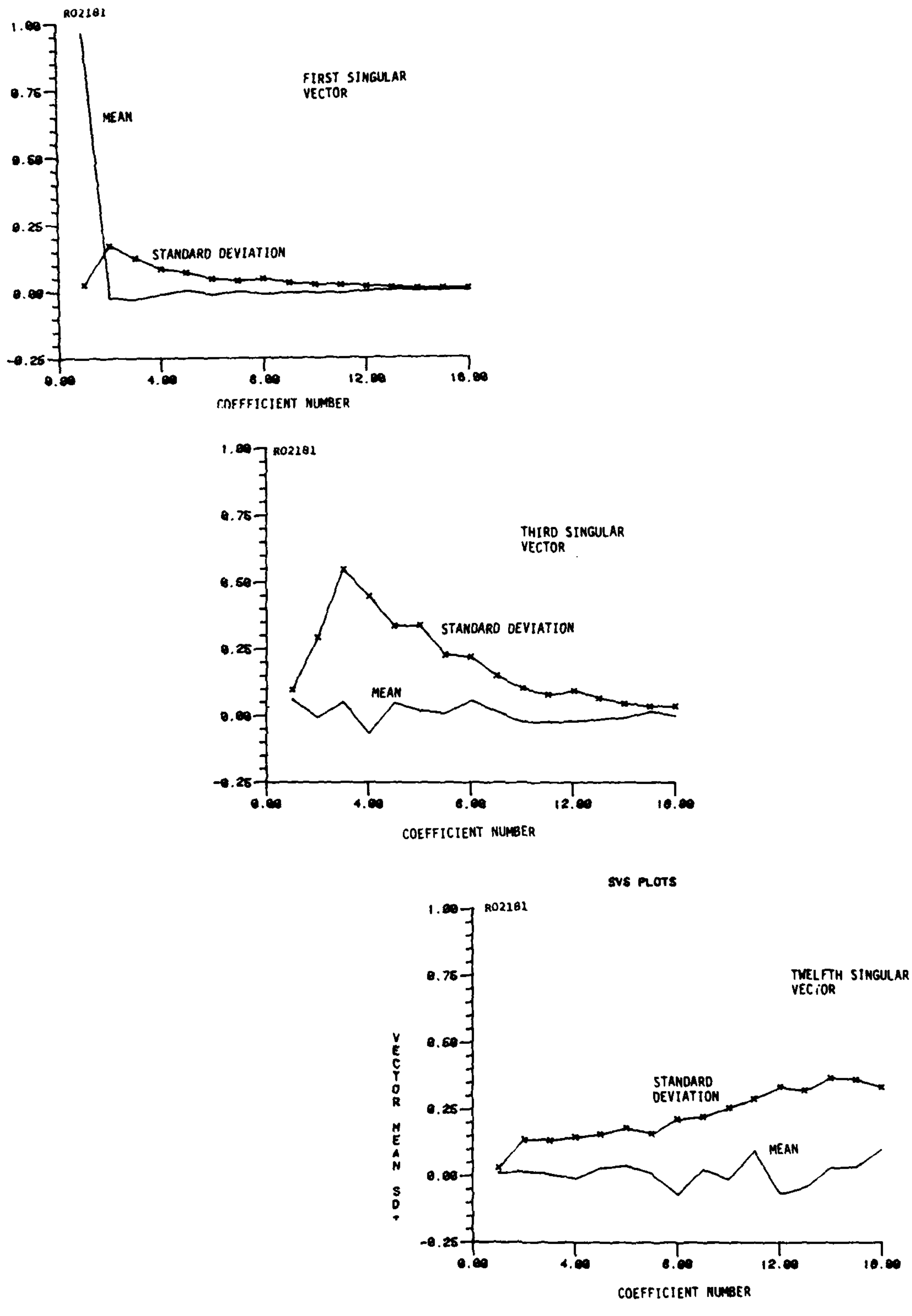
This reordering was implemented and evaluated against no reordering. The results are reported in Section 7.

5.2.2 Singular Vector Orthogonalization

This enhancement represents an attempt to exploit the known *orthogonal* structure of the singular vector arrays U and V. It results in a structure somewhat different from that otherwise applied for coding singular vectors.

Up to this point, singular vector coefficient coding was handled simultaneously: after the 1D transform was applied, all the coefficients were encoded at once. In the current enhancement, the structure is different: first, some coefficients are extracted, then they are coded, then other coefficients are extracted, and then they are coded. This process cycles until all coefficients are extracted and coded.

This enhancement is intended to exploit the known redundancy in the arrays U and V. Here we will focus on the left singular vector coefficient array $\tilde{U}^t U$. For notational simplicity, we will denote this array simply as U during the remainder of this discussion, although the process is applied to the coefficient array $\tilde{U}^t U$.



VERAC Incorporated

Figure 5-10. Example of Migration of Energy Concentrated with Frequency

First we note tnat the j-th column of U, $\underline{u}_j$, is orthogonal to all previous columns $\underline{u}_1$, $\underline{u}_2 \cdots$, $\underline{u}_{j-1}$. Suppose a change of basis is introduced, with the new basis being

$$B_j = [\underline{u}_1, u_2, \cdots, u_{j-1}, b_{jj} \cdots b_{nj}],$$

where $\{\underline{b}jj, \cdots, \underline{b}_{nj}\}$ simply completes the basis in R^n. The vector u_i can be represented by:

$$\underline{u}_j = \sum_{i=j}^{n} \alpha_{ij} \underline{b}_{ij},$$

since $\underline{u}_j$ is orthogonal to $\{u_1, u_2, \cdots, u_{j-1}\}$ so that $\underline{u}_j$ is linearly independent of $\{u_1, u_2, \cdots, u_{j-1}\}$. Thus, instead of having to transmit the n elements of $\underline{u}_j$ directly, only the $(n-j+1)$ coefficients $\{\alpha_{ij}, \cdots, \alpha_{nj}\}$ need be transmitted, as long as the $\{\underline{b}_{ij}\}$ are available to both transmitter and receiver. But the $\{\underline{b}_{ij}\}$ can be computed from the previously transmitted singular vectors $\{u_i\}_{i=1}^{j-1}$, so that the process is realizable.

When repeated for each i, this process results in an array of α's which can be collected into the following upper triangular form:

$$\begin{bmatrix} \alpha_{11} & & & \\ \alpha_{21} & \alpha_{22} & & 0 \\ \vdots & \vdots & \ddots & \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{bmatrix}$$

Thus, the n^2 elements of U can be completely represented by the $\frac{n(n-1)}{2}$ coefficients $[\alpha_{ij}]$.

The most obvious coding strategy based on this representation is to independently code the individual α_{ij}'s using statistics collected during a pre-processing statistical analysis. However, in this method, the reconstruction accuracy of later singular vectors is very sensitive to errors in earlier ones.

This can be seen by examining the reconstructed value of $\underline{u}_j$, $\hat{\underline{u}}_j$:

$$\hat{\underline{u}}_j = \sum_{i=j}^{n} \hat{\alpha}_{ij} \hat{\underline{b}}_{ij}$$

Here, not only is $\hat{\alpha}_{ij}$ an approximation to α_{ij}, but so is $\hat{\underline{b}}_{ij}$ an approximation to $\underline{b}_{ij}$. And although the error in $\hat{\alpha}_{ij}$ is independent of the error in all other α_{ij}'s, the error in $\hat{\underline{b}}_{ij}$ depend upon the previous reconstructed values $\{\hat{\underline{u}}_1, \hat{\underline{u}}_2, \cdots, \hat{\underline{u}}_{j-1}\}$ which themselves depend upon $\{\hat{\alpha}_{pq} : q \leq p,\ q<j\}$.

To aviod this dependency and to thereby reduce average coding errors, we use a different basis B_j:

$$B_j = [\hat{\underline{u}}_1, \hat{\underline{u}}_2, \cdots, \hat{\underline{u}}_{j-1}, \hat{\underline{b}}_{jj}, \cdots, \hat{\underline{b}}_{nj}]$$

where $\{\hat{\underline{b}}_{ij}, \cdots, \hat{\underline{b}}_{nj}\} = \{\hat{\underline{u}}_1, \hat{\underline{u}}_2, \cdots, \hat{\underline{u}}_{j-1}\}$. The price we pay is that the coefficient matrix of α_{ij}'s is no longer triangular -- it is in general full. However, the elements occuring in the upper triangle (the α_{ij}'s for $i<j$) will typically be small as long as $\hat{\underline{u}}_j$ is a reasonable approximation to $\underline{u}_j$. They can therefore either be neglected altogether, or, as we shall do, be more coarsely quantized tnan those α_{ij}'s in the lower triangle (α_{ij}'s for $j \leq i$).

5.2.3 Repolarization

In the SVD expansion

$$X = USV^t$$

$$= \sum_{i=1}^{n} s_i \underline{u}_i \underline{v}_i^t$$

there is a fundamental question of polarity of the various members of each term. In particular, the term $s_i \underline{u}_i \underline{v}_i^t$ has a definite sign, buc the individual members $s_i, \underline{u}_i$ and $\underline{v}_i^t$ do not, so long as their product works out to have the correct polarity.

Normal default in SVD is to choose $s_i \geq 0$. That still leaves $\underline{u}_i$ and $\underline{v}_i$'s polarities unspecified, but constrained so that their product is of the correct sign. The additional condition we impose is that the sum of the areas of the two vectors, $\sum_j u_{ij}+v_{ij}$, be non-negative. This condition is basically a default selected to help minimize the dispersion of singular vector coefficient statistics.

Repolarization is an alternate scheme intended to further reduce the dispersion of singular vector coefficient statistics. It is based upon the motive of providing a consistent sign for the largest energy component of eacn singular vector. The scheme consists of assigning signs to $\underline{u}_i$ and $\underline{v}_i$ that result in their both having their largest magnitude transform coefficient be positive. The sign of s_i is then adjusted to give the term $s_i \underline{u}_i \underline{v}_i^t$ the correct polarity.

The price for this repolarization is that singular values are no longer guaratneed to be non-negative and thus display increased dispersion in their statistics. However, having both polarization methoas available permits an evaluation of which effect dominates, the decrease in dispersion of singular vector coefficients, or the increase in dispersion of singular values.

5.3 Preprocessing

SVD preprocessing is required for the same reason KLT preprocessing is, as a training step. Since the singular values and the singular vector coefficients are coded using statistically-optimized coding schemes, the underlying statistics are required.

What are required are:

- Singular value statistics, and

- Singular vector statistics.

The procedure for obtaining this information is shown in Figure 5-11. This process constitutes an empirical SVD statistics calculation. It begins with the forward SVD transforming of the various blocks in the training imagery. Since coding is again class-adaptive, separate class-specific statistics are required. Also, since several different ordering, polarization methods are to be evaluated, several versions of the statistics are required. These include:

- Default singular value/vector ordering and polarization,
- Singular value/vector re-rodering,
- Singular value/vector re-polarization, and
- Both re-rodering and re-polarization.

In each case, the same SVD is applied, and the results simply reorganized as reqired. (As previously discussed in 5.1, the order of the SVD and transform operations can be interchanged.)

The statistics required are the first and second moments of the various entities to be coded. Specifically, let s_i denote the i-th singular value, and $\underline{u}_i$ and $\underline{v}_i$ the transform coefficient vectors for the corresponding singular vectors. Then the statistics calcualted are:

- Singular values

$$\bar{s}_{k,i} = \text{AVG}[s_i : x \text{ is class } k]$$

$$\bar{s}^2_{k,1} = \text{AVG}\ [s_i^2 : x \text{ is class } k]$$

- Singular vector coefficients

$$\bar{U}_k = \text{AVG}\ [U : x \text{ is class } k]$$

$$\bar{V}_k = \text{AVG}\ [V : x \text{ is class } k]$$

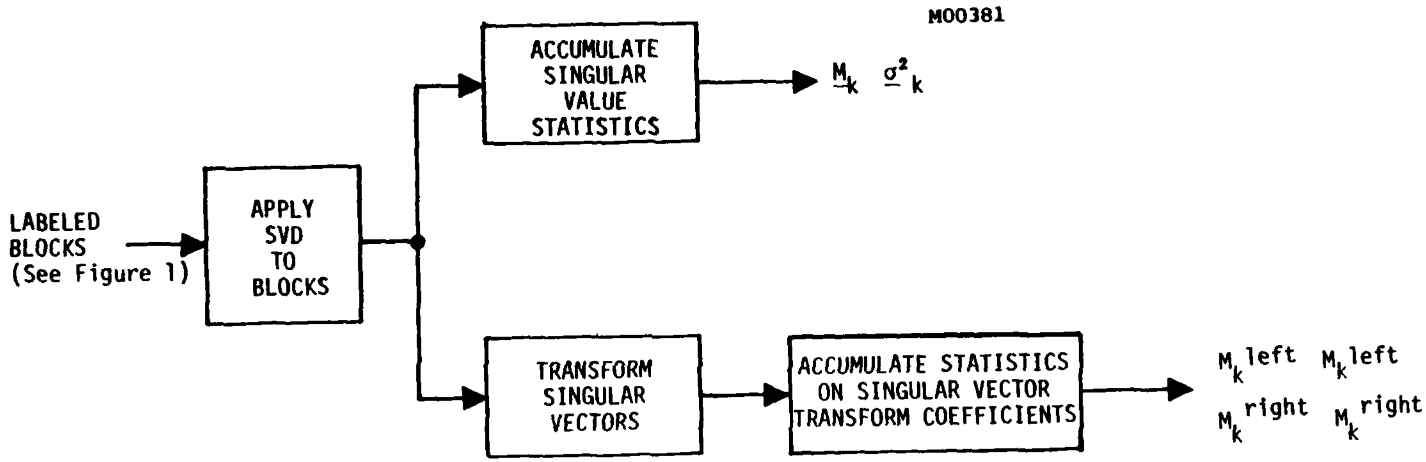


Figure 5-11. SVD Preprocessing

$$P_k^{left} = \left[r_{k,ij}^{left}\right] , \left(r_{k,ij}^{left}\right)^2 = \text{AVG} \left[u_{ij}^2 : \text{ X is class k}\right]$$

$$P_k^{right} = \left[r_{k,ij}^{right}\right] , \left(r_{k,ij}^{right}\right)^2 = \text{AVG} \left[v_{ij}^2 : \text{ X is class k}\right]$$

From these, the required standard devitions can be computed as:

$$\left(\sigma_i^s(k)\right)^2 = \overline{s_{k,i}^2} - \bar{s}_{k,i}^2$$

$$\Sigma_k^{left} = \left[\sigma_{k,ij}^{left}\right], \left(\sigma_{k,ij}^{left}\right)^2 = \left(r_{k,ij}^{left}\right)^2 - \bar{u}_{k,ij}^2$$

$$\Sigma_k^{right} = \left[\sigma_{k,ij}^{right}\right], \left(\sigma_{k,ij}^{right}\right)^2 = \left(r_{k,ij}^{right}\right)^2 - \bar{v}_{k,ij}^2$$

The efficient sample space enhancement techniques applied to smooth out structural artifacts in the KLT case can also be applied here. Specifically, both "flips" and "rotation" can be applied. Appendix D discusses how to implement these techniques on SVD's of pre-transformed data, which is the case of interest here.

In order to encode the coefficients resulting from the singuar vector orthogonal expansion enhancements discussed in 5.2.2, the first two moments of the orthogonal expansion coefficients are required. Appendix E computes expressions for these quantities which allow their calculation from the statistics of $\{s_i, \underline{u}_i$, and $\underline{v}_i\}$.

5.4 Singular Value/Vector Coding

To insure adaptibility to non-stationarity, a class-adaptive coding scheme is employed. Figure 5-1 illustrated the SVD coding chain and indicated the place of overhead, singular value, and singular vector coding in the overall arrangement.

As in the KLT case, overhead consists of the global rate distortion parameter D and each block's class label k. Both are required to permit correct singular value/vector reconstruction at the decoder. (Note that although the SVD transform is not class-dependent as was the KLT, the class labels are nontheless needed at the decoder for correct singular value/vector reconstruction.) SVD overhead coding is identical to KLT overhead encoding.

The overall singular value/singular vector coding problem constitutes a hierarchy of coding problems, which pose the following questions:

- How are bits allocated among blocks,
- How are bits allocated to terms in a particular block's SVD expansion,
- How are bits distributed among the singular value and two singular vectors in particular terms, and
- What coders are best for use on singular values and singular vectors?

5.4.1 Bit Allocation

To obtain solutions to these problems we again adopt the following global problem statement:

MINIMIZE : Total mean squared coding error

SUBJECT TO : Not exceeding a given coding rate

and specify the use of fixed rate coders (coders which produce codewords whose lengths do not depend upon the input values to be coded).

As demonstrated in Appendix C (in the context of KLT coding) the optimal solution dictates allocating bandwidth to achieve a uniform distribution of coding error over all blocks. This implies that busier blocks are encoded with more bits than are quieter blocks. Furthermore, Appendix C shows that each block's bit allocation problem can be separately solved. Appendix F addresses this problem (the second and third in the list), and shows that bandwidth should be allocated so that coding error is distributed uniformly over all terms $s_i \underline{u}_i \underline{v}_i^t$ in the SVD expansion

$$X = \sum_i s_i \underline{u}_i \underline{v}_i^t .$$

This means that those terms $s_i \underline{u}_i \underline{v}_i^t$ which have the most variation in energy will be allocated the most bits; those which are more predictable receive fewer bits.

Appendix F also shows that the bit allocation problem can be solved separately for each term, and that the optimal solution has the following features:

- the singular value s_i is allocated bits according to its variability, as given by its class k standard deviation, $\sigma_i^s(k)$,
- the singular vectors $\underline{u}_i, \underline{v}_i$ are allocated bits according to the average size of the corresponding singular value s_i, as given by its class k RMS value, $r_s(k)$, and
- bits are distributed among the 1D transform coefficients of the singular vectors to achieve uniform coding error in each coefficient.

The matnematical form of the allocation rule is as follows:

$$Bs_i = \text{INT}\left[\log_2 \frac{\sigma_i^s(k)}{D}\right]$$

$$Bu_{ij} = \text{INT}\left[\log_2 \frac{\left(\sigma_{k,ij}^{left}\right)\left(r_s(k)\right)}{D}\right]$$

$$Bv_{ij} = \text{INT}\left[\log_2 \frac{\left(\sigma_{k,ij}^{right}\right)\left(r_s(k)\right)}{D}\right]$$

where Bs_i = number of bits allocated to s_i,

Bu_{ij} = number of bits allocated to u_{ij},

Bv_{ij} = number of bits allocated to v_{ij},

$\sigma_i^s(k)$ = standard deviation of s_i in class k,

$r_s(k)$ = $\sum_i \bar{s}^2_{k,i}$,

$\sigma_{k,ij}^{left}$ = standard deviation of u_{ij} in class k,

$\sigma_{k,ij}^{right}$ = standard deviation of v_{ij} in class k, and

D = global distortion control parameter.

(This rule is an approximation based on a particular curve fit to the performance characteristics of the fixed rate coders used to quantize the singular values/vectors.)

As an example, the bit assignment matrices shown in Figure 5-12 were calculated from these rules. The figure illustrates the bit assignments for the singular values and left singular vector transform coefficients (right singular vector transform coefficients are similar) for two classes.

Both inter- and intrablock adaptivity are in evidence. The former obtains from the difference in total number of bits assigned, the latter from the selective allocation within the arrays. Note the larger allocations to the first few singular values/vectors, which are the ones with largest energy. Note also the preferential allocation within columns of the singular-vector-coefficient bit-allocation matrix, reflecting the energy compaction properties of the transform. Additionally, note the evidence of the centroid-of-energy migration from lower coefficient indices (top of column) to higher indices (botton of column), as reflected by the shifting bit allocation pattern as we move from the first few singular vectors (left side or array) to the last few singular vectors (right side of array). Finally, note that many singular value/vector combinations are not coded at all. This is a result of the highly efficient energy compaction into the first few terms in the SVD expansion provided by the SVD transform.

5.4.2 <u>Singular Value/Vector Coders</u>

Figure 5-13 illustrates the singular value statistics specific to a particular class of image blocks. Because each singular value extends over a fairly narrow range (approximately constant in log space, except for the last one), and because high fidelity singular value coding was desired (for the same reason high fidelity "dc" coding is in the KLT case), we selected a uniform quantizer for encoding singular values.

```
CLASS      1
Quantizer assignment matrix for singular values
 8  3  2  2  1  0  0  0  0  0  0  0  0  0  0  0
Quantizer assignment matrix for Left singular vectors
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 3  3  2  1  0  0  0  0  0  0  0  0  0  0  0  0
 2  3  2  1  0  0  0  0  0  0  0  0  0  0  0  0
 2  2  2  1  1  0  0  0  0  0  0  0  0  0  0  0
 2  2  1  1  1  0  0  0  0  0  0  0  0  0  0  0
 1  2  1  1  1  0  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
 1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Lowest Activity Class

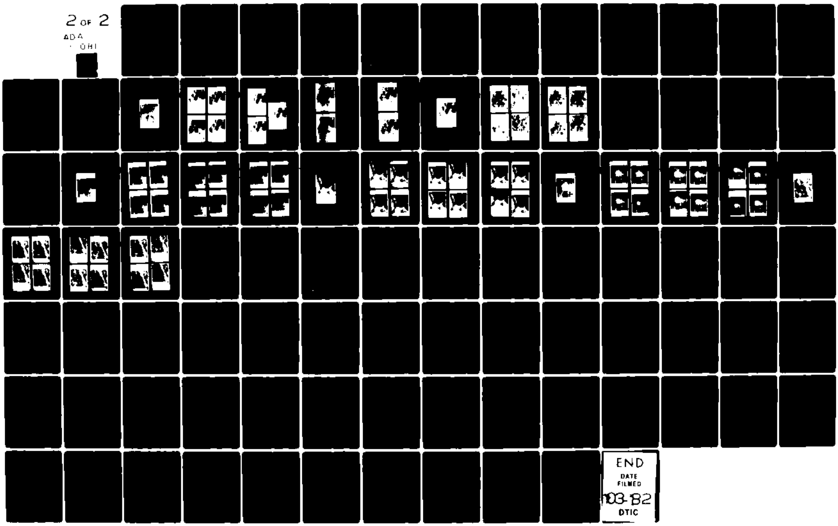
```
CLASS      8
Quantizer assignment matrix for singular values
 8  5  5  2  3  2  1  1  0  0  0  0  0  0  0  0
Quantizer assignment matrix for Left singular vectors
 5  4  3  0  1  0  0  0  0  0  0  0  0  0  0  0
 7  6  4  2  2  0  0  0  0  0  0  0  0  0  0  0
 6  6  6  3  2  0  0  0  0  0  0  0  0  0  0  0
 5  5  5  4  3  1  0  0  0  0  0  0  0  0  0  0
 4  4  4  3  3  2  0  0  0  0  0  0  0  0  0  0
 3  4  4  3  2  2  1  0  0  0  0  0  0  0  0  0
 3  3  4  3  2  1  2  0  0  0  0  0  0  0  0  0
 2  2  2  3  3  1  1  1  0  0  0  0  0  0  0  0
 2  2  2  2  2  1  1  0  0  0  0  0  0  0  0  0
 1  0  1  2  2  1  1  0  0  0  0  0  0  0  0  0
 1  0  1  0  2  1  1  0  0  0  0  0  0  0  0  0
 0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
 0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Highest Activity Class

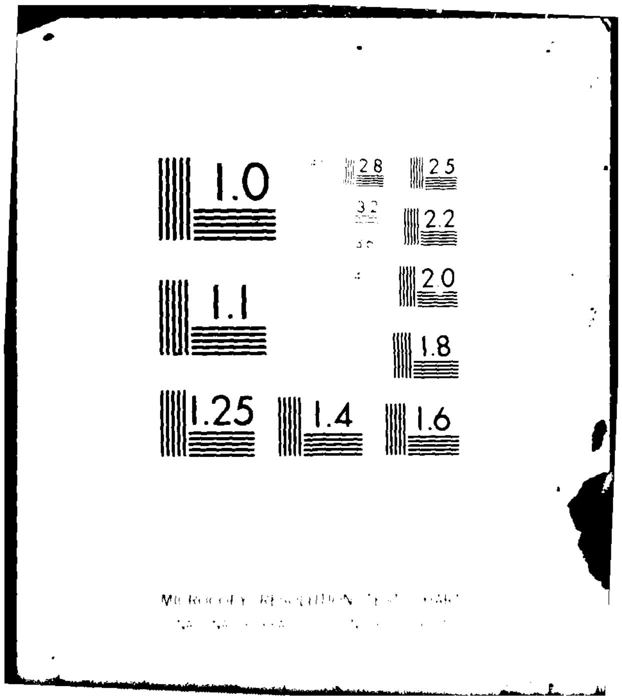
Figure 5-12. Example SVD Bit Assignment Arrays

AD-A110 811 VERAC INC SAN DIEGO CA F/G 5/8
IMAGE COMPRESSION RESEARCH.(U)
AUG 81 M S MURPHY F30602-80-C-0168
UNCLASSIFIED R-021-81 RADC-TR-81-223 NL

2 OF 2
ADA
0811



END
DATE
FILMED
03-82
DTIC


1.0
2.8
2.5
3.2
2.2
2.0
1.1
1.8
1.25
1.4
1.6

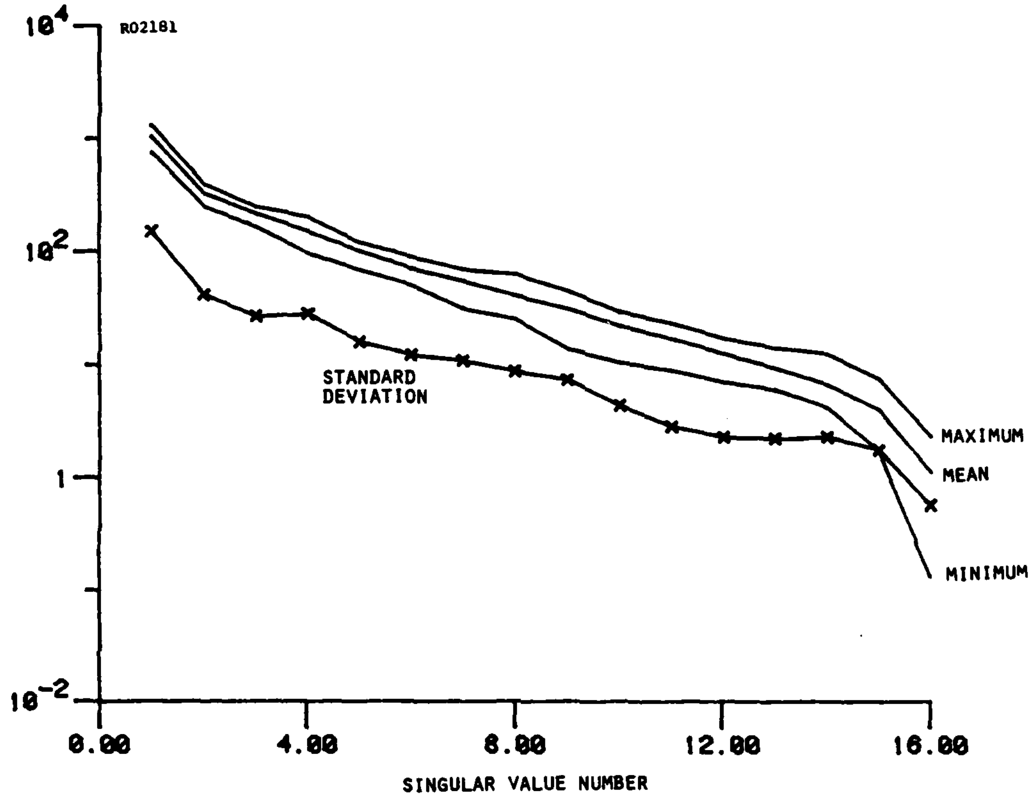


Figure 5-13. Example Singular Value Statistics

The procedure employed to encode singular values was:

- Subtract mean: $\tilde{s}_i = s_i - \bar{s}_i$
- Use a uniform quantizer of Bs_i bits, with lower end -D ; and upper end D, where

$$D = \sqrt{3 \ (\sigma_j^s)^2} = \sqrt{3}\ \sigma_i^s.$$

The value D describes the extent of a uniform pdf with standard deviation σ_i^s. Note: the s_1 case was handled slightly differently; the quantizer extent was stretched to exend down to zero instead of -D, if necessary. This is the precise analog of the special "dc" treatment included for the KLT, and is included to insure equally accurate quantization of average grey level for all blocks, and to thereby minimize blockiness.

As used in the KLT case for transform coefficient coding, a Max quantizer was applied to encode singular vector transform coefficients in the SVD case. A tail-modified Gaussian pdf was assumed, and Max quantizers of 1,2, ⋯, 8-bit length were used. The coding procedure was:

- Subtract mean from coefficients,
- Normalize by standard deviations of coefficients,
- Encode coefficients using BCD representation of the quantization levels obtained from Bu_{ij} or Bv_{ij} (as appropriate) bit Max quantizers.

5.4.3 Coding the SVD Orthogonal Expansion Coefficients

In this enhancement singular vectors are treated differently. The procedure is cyclic, and is repeated for each of the left and right singular vectors. For concreteness, suppose $\underline{u}_j$ (the vector of transform coefficients for the j-th left singular vector) is to be coded: A single cycle consists of the following steps.

- Find the basis matrix B_j,

- Find the coefficients $\underline{\alpha}_j = \begin{bmatrix} \alpha_{ij} \\ \alpha_{2j} \\ \vdots \\ \alpha_{nj} \end{bmatrix}$

- Quantize $\underline{\alpha}_j$ and output the resulting codeword to the buffer, and

- Reconstruct $\hat{\underline{\alpha}}_j$ values for next cycle.

A similar set of steps produces $\underline{\beta}_j$'s from the $\underline{V}_j$'s.

The theory to support this process was covered in Section 5.2.2. The topic here encompasses only the quantizers and bit assignments used for encoding the α_{ij}'s. For the quantizer, the same modified-Gaussian Max quantizer employed for the KLT and the other SVD algorithms is employed here. Coding is performed by the following procedure:

- Determine Bs_i as in § 5.4.1

- Determine Bu_{ij} and Bv_{ij} as in § 5.4.1, using statistics of α_{ij} in place of those of u_{ij} (and those of β_{ij} in place of those of v_{ij})

- Encode α_{ij} using a Bu_{ij}-bit quantizer, and β_{ij} using a Bv_{ij}-bit quantizer.

At the conclusion of this cycle, the next iteration is begun. This consists of incrementing j to j+1 and repeating the above process for $\underline{\alpha}_{j+1}$ and $\underline{\beta}_{j+1}$.

6.0 RATE EQUALIZATION

Rate equalization is the process of meeting a global target compressed rate (in bits per pixel). Since the coding algorithms employed in this study are class-adaptive and since it is generally not known ahead of time how many blocks of each class are present in a given image, it is necessary to regulate the coding process in order to adapt to changing class population from image to image [6].

The concept upon which rate equalization is based is illustrated in Figure 6-1 which shows the overall coder performance curve. The curve shows how the coder's output rate (average bits per pixel) depends upon a control parameter D. The parameter is a measure of the distortion added during coding: in order to achieve a small coded rate a large D is necessary; for larger coded rates, a smaller D will do. The task of rate equalization is to select the D that meets the target rate. Since the curve depends upon not only the coder, but also the image being encoded, the problem in non-trivial.

The method of rate equalization employed in the study is <u>predictive rate equalization</u>. This means that it is performed prior to coding the image. That is, no trial-and-error coding is required to meet the target rate. The correct value of D can be determined before any coding commences.

The determination of the correct global distortion parameter D is based on two types of information:

- class-specific transform coefficients statistics, and
- class-populations.

Thus, D only depends upon aggregate image information; it does not depend upon the actual image data (pixel values) themselves.

The rate equalization problem is solved by determining the value of D which satisfies the following condition:

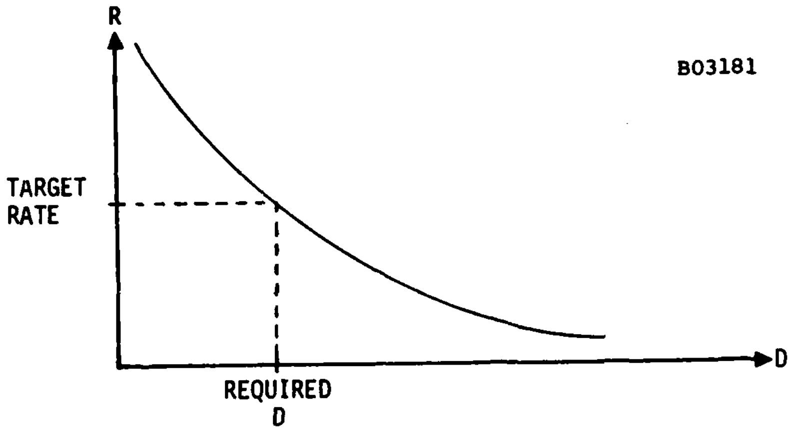


Figure 6-1. Rate Equalization

$$\text{Total bits for image} = \sum_{\substack{\text{class} \\ k}} N_k \cdot B_k(D) \tag{6.1}$$

where N_k = number of blocks of class k

$B_k(D)$ = number of bits allocated to class k blocks

The quantity $B_k(D)$ represents the total class k bit allocation and is obtained by summing over all elements of the class-k bit allocation array. For the KLT case, the expression for $B_k(D)$ is:

$$B_k(D) = \sum_{i,j}^{n} \log_2 \left[\frac{\sigma_{ij}(k)}{D}\right] \tag{6.2}$$

where $\sigma_{ij}(k)$ is the standard deviation of the ij-th element of the KLT coefficient array Z obtained from class k blocks. For the SVD case, the expression is more complicated; since singular values, left singular vectors and right singular vectors must all be coded:

$$B_k(D) = \sum_{j}^{n} \left\{ \log_2 \frac{\sigma_j^s(k)}{D} + \sum_{i}^{n} \log_2 \frac{\sigma_{ij}^{left}(k)\, r_j^s(k)}{D \cdot \gamma} + \sum_{i}^{n} \log_2 \frac{\sigma_{ij}^{right}(k)\, r_j^s(k)}{D \cdot \gamma} \right\} \tag{6.3}$$

where $\sigma_j^s(k)$ = standard deviation of j-th singular value for class k blocks

$r_j^s(k)$ = RMS value of j-th singular value for class k blocks

$\sigma_{ij}^{left}(k)$ = standard deviation of i-th transform coefficient of j-th left singular vector for class k blocks

$\sigma_{ij}^{right}(k)$ = standard deviation of i-th transform coefficient of j-th right singular vector for class k blocks

γ = a constant of proportionality.

The rate equalization process thus entails finding D to satisfy these conditions. For this study, the process was implemented by performing an iterative search of logD-space, relying upon the convexity of the R/D curve of Figure 6-1 it insure rapid convergence.

It is important to note that for each trial value of D, only a simple analytical expression (equation 6.2 or 6.3) need be computed and the result compared with the goal to see if (6.1) is satisfied. If not, a correction to D is applied and the procedure is repeated. Actual image coding is not necessary to find the correct D. In addition, experience indicates that convergence occurs usually in two iterations, but essentially always by the third iteration.

7.0 ALGORITHM EVALUATION

Algorithm evaluation was conducted in two phases:

- Preliminary, and

- Comprehensive.

The preliminary evaluation was aimed at investigating the relative merits of the various perturbations of the KLT and SVD algorithms developed under the effort, using a small set of test imagery. Based on this evaluation, the best members in each catagory were selected and more thoroughly exercised against a larger set of imagery to compare their performance with each other and with the baseline cosine and Hadamard algorithms.

All algorithms were essentially identical in all ways except for which transform applied. Thus, all had the following features:

- Class-adaptive coefficient coding,

- Empirical accumulation of class-specific coefficient statistics (except KLT/P),

- Special, error-free, "dc" coding,

- Same intensity mappings,

- Same blocks labels obtained from block classification,

- 16 X 16 block size,

- Same block boundaries,

- Same fixed rate quantizers (uniform and Max), and

- Same rate equalization techniques.

Algorithms were compared on several bases:

- Coding efficiency:

    - Mean square error versus coded rate
    - Mean absolute error versus coded rate
    - Subjective perception of distortion in reconstructed image versus coded rate
    - Subjective perception of information in error image versus coded rate

- Computation efficiency:

    - Execution time
    - Adaptability to hardware implementation

The remainder of this section is divided into two subsections, 7.1 which summarizes the findings of the preliminary evaluation, and 7.2 which presents the results of the comprehensive evaluation.

7.1 Preliminary Evaluation

Preliminary evaluation consisted of two parts:

- Determination of optimal class boundaries, and

- Algorithm evaluation.

7.1.1 Optimal Class Boundaries

All algorithms tested employed class-adaptive coefficient coding. An important aspect of such algorithms is determining good class definitions. In this study, a single scalar feature was extracted from each block and used to classify the block into one of eight classes, according to the value of that feature. Specifically, the feature used was block a.c. energy:

$$\mu(x) = \frac{1}{n^2} \sum_{ij} x_{ij}^2 - \left(\frac{1}{n^2} \sum_{ij} x_{ij}\right)^2 \quad ;$$

and the classifier took the form:

$$X \text{ is labled class } k \quad \text{IF} \quad t_{k-1} \leq \mu(X) < t_k.$$

This portion of preliminary evaluation dealt with determining the best values for $\{t_k\}$.

Five types of threshold settings were investigated:

(1) Uniform class population in test image,

(2) Uniform class population over many images,

(3) Uniform thresholds in μ,

(4) Uniform thresholds in $\sqrt{\mu}$,

(5) Uniform thresholds in $\log(\mu)$.

Evaluation consisted of exercising the baseline cosine coding algorithm on a particular GFE aerial image, for each of the threshold settings, over a range of compression rates.

Comparisons were based on mean square coding error (MSE), on mean absolute coding error (MAE), and on subjective comparisons of original, coded, and error images. The conclusion was that, based on MSE, (4) performed best with (1) a close second. Based on MAE, (4) again performed best, but this time both (1) and (2) were close. Subjectively, (4) was judged to produce the best results with (5) a close second.

Altogether, (4) was selected as best. Therefore, class boundaries uniform in $\sqrt{\mu}$ (uniform in block RMS value) were used for all algorithms during the remainder of the evaluations.

7.1.2 Preliminary Algorithm Evaluation

The following algorithms were compared under preliminary evaluation:

- COS : 2D cosine transform (baseline)
- HAD : 2D Hadamard transform (baseline)
- KLT/P : Class-adaptive KLT using predicted coefficient statistics
- KLT/E : Class-adaptive KLT using empirical coefficient statistics
- SVD/COS : SVD using 1D cosine transform of singular vectors
- SVD/HAD : SVD using 1D Hadamard transform of singular vectors
- SVD/COS/RO : Same as SVD/COS but with reordering enhancement
- SVD/HAD/RO : Same as SVD/HAD but with reordering enhancement
- SVD/COS/ORTH : Same as SVD/COS but with orthogonal expansion enhancement
- SVD/COS/RP : Same as SVD/COS but with repolarization enhancement

Each algorithm was applied to a GFE aerial image of an airfield at three coded rates, 0.5, 1.0, and 1.5 bits per pixel (bpp).

Mean Squared Error

The first eight algorithms were evaluated first. The last two were later enhancements subsequently evaluated. Figure 7-1 shows the mean square coding error plots for the first eight algorithms. Each curve is a piecewise linear fit to the three evaluation points (0.5, 1.0, and 1.5 bpp). Curves located towards the bottom of the plot indicate better coding efficiency than do curves located towards the top.

From this figure, the KLT/E and COS algorithms are seen to perform best; they add the least amount of mean square coding error of any algorithm. Since their curves essentially overlap, it is not possible to judge relative superiority of one of these over the other on the basis of MSE; however, both are markedly superior to the remaining six algorithms.

The figure also shows the poorest performance is obtained for algorithms employing the Hadamard transform. It is illuminating to compare the COS and HAD curves to see how much coding efficiency one gives up to gain the computational efficiency provided by the HAD algorithm. For example, the figure indicates that the COS performs as well at 0.5 bpp as the HAD does at twice that rate, 1.0 bpp. This same effect is in evidence in comparing the various SVD/COS algorithms with the various SVD/HAD algorithms.

Of particular relevance to this study, the figure shows that the various SVD algorithms perform significantly worse than the COS or KLT/E algorithms. The SVD/COS algorithms are superior to the baseline HAD algorithm but are nonetheless inferior to the baseline COS and the KLT/E.

Additionally, the figure also indicates that using empirically determined statistics in the KLT is superior to using predicted statistics based on a separable covariance model. This indicates that

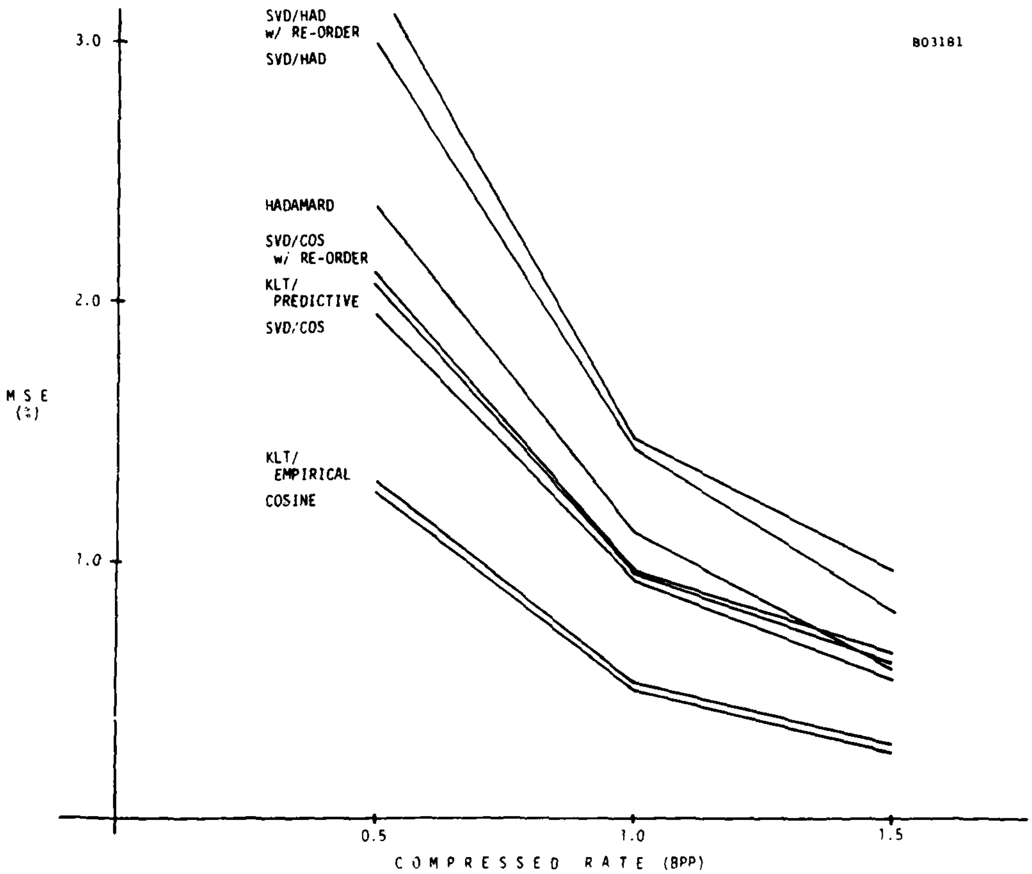


Figure 7-1. Coding Algorithm Comparisons MSE versus Rate Airfield Image

this model is not particularly good at characterizing image correlation, even though use of the KLT transform operators derived under this model do yield good results when empirical statistics are provided to the coder.

Finally, the singular value/vector reordering enhancement is seen to slightly degrade, rather than improve, both SVD/COS and SVD/HAD performance. This indicates that ordering on the basis of singular value size produces smaller dispersions in singular vector coefficient statistics than does ordering on the basis of singular vector frequency (or sequency) content.

<u>Mean Absolute Error</u>

Very similar relative algorithm performance is indicated by the mean absolute error curves of Figure 7-2. These curves show the intensity of the error image obtained at each experiment point. Since, on the whole, the curves occupy the same relative positions in Figure 7-2 as they do in Figure 7-1, similar conclusions on relative performance are drawn.

<u>Subjective Evaluation</u>

Figure 7-3 illustrates a GFE aerial photograph made available for algorithm testing. The 256 X 256 subset shown was extracted and used for preliminary evaluation. The results of Figures 7-1 and 7-2 were obtained by processing this subset. Additional, subjective comparisons of the algorithms were also performed.

Figures 7-4 through 7-8 show the reconstructed images obtained by applying the various algorithms at several bit rates. Figures 7-9 and 7-10 show error images for the eight algorithms operated at 1.0 bpp. Several observations were obtained by examining these pictures.

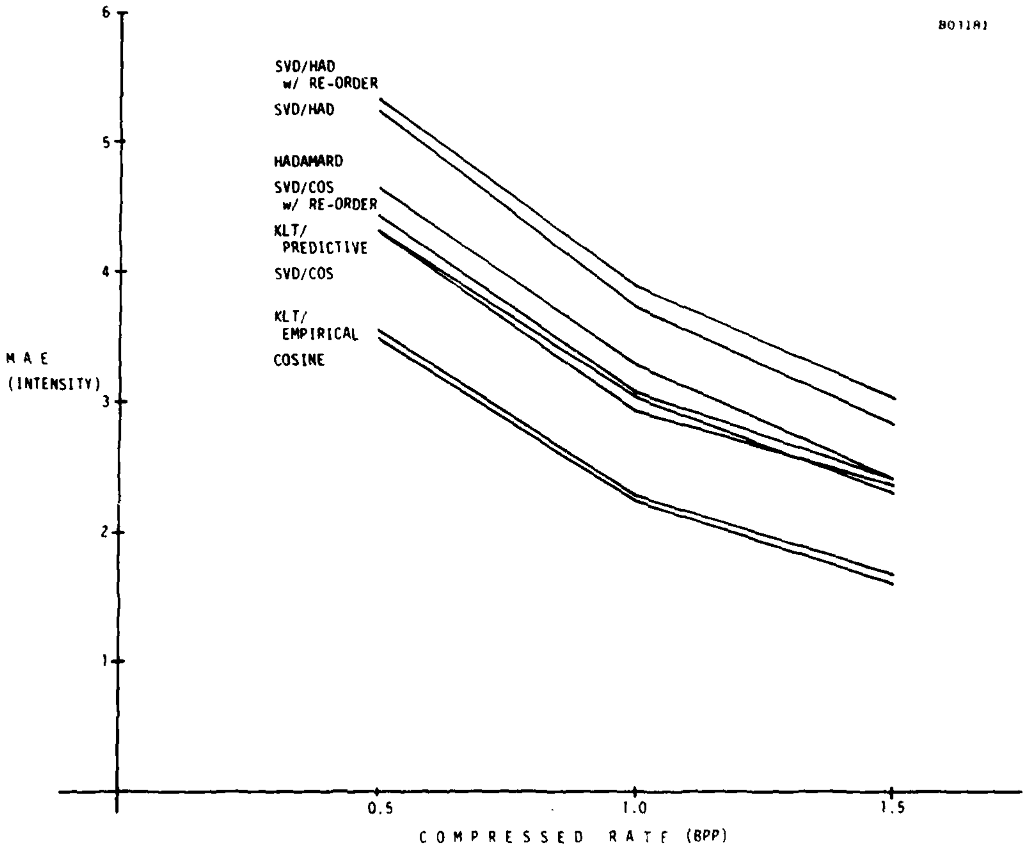
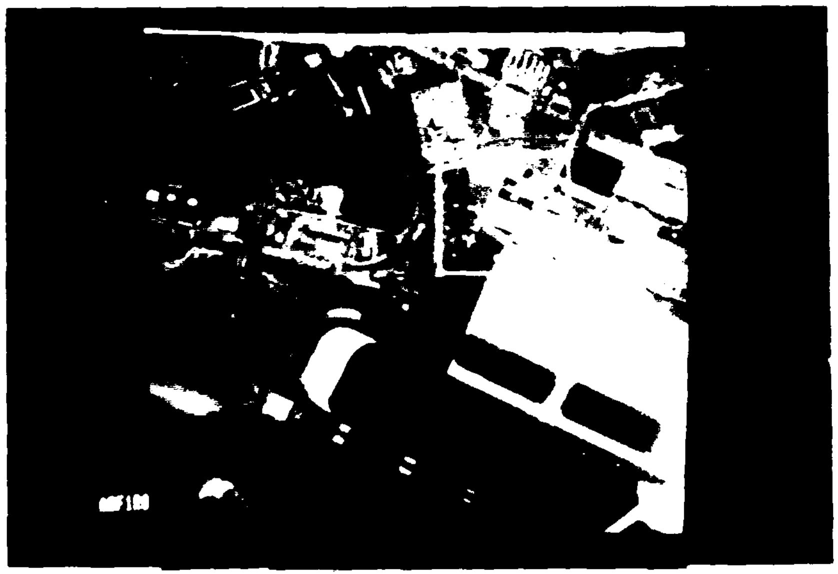
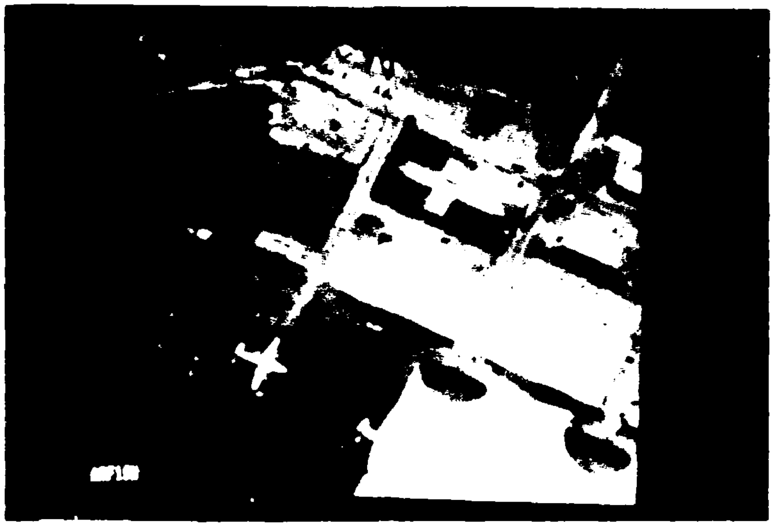


Figure 7-2. Coding Algorithm Comparison MAE versus Rate
Airfield Image

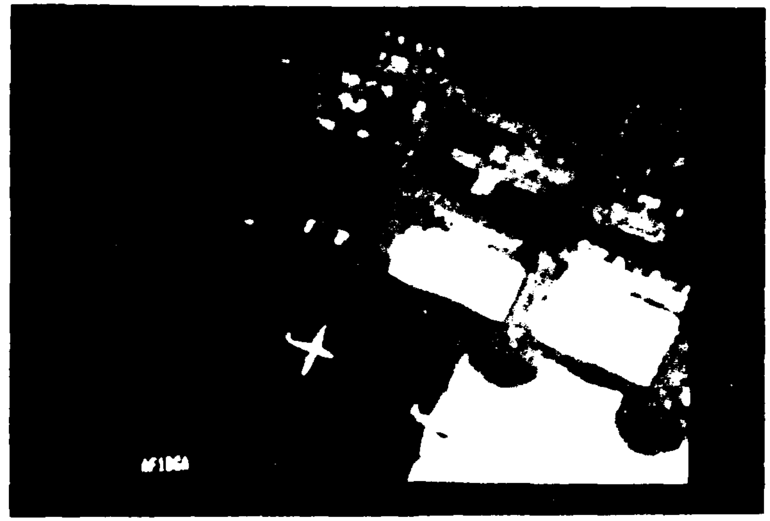


Test Image for Preliminary Evaluation

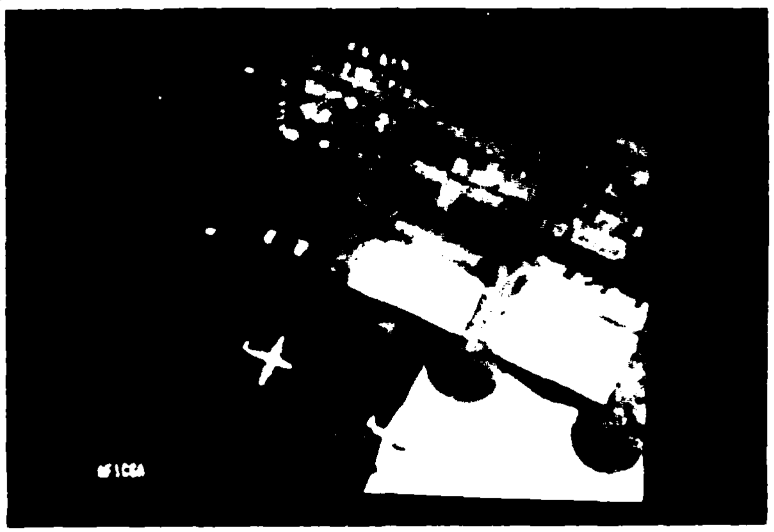
Figure 7.3



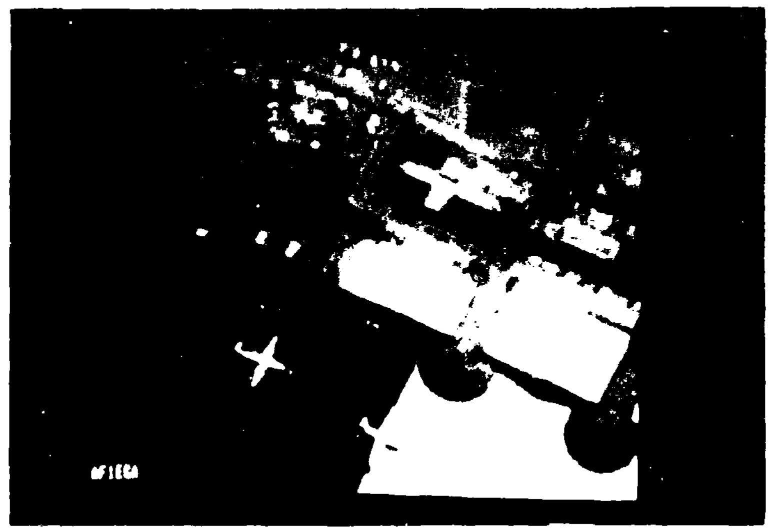
Original 8bpp



SVD/COS 0.5bpp 1.95%MSE

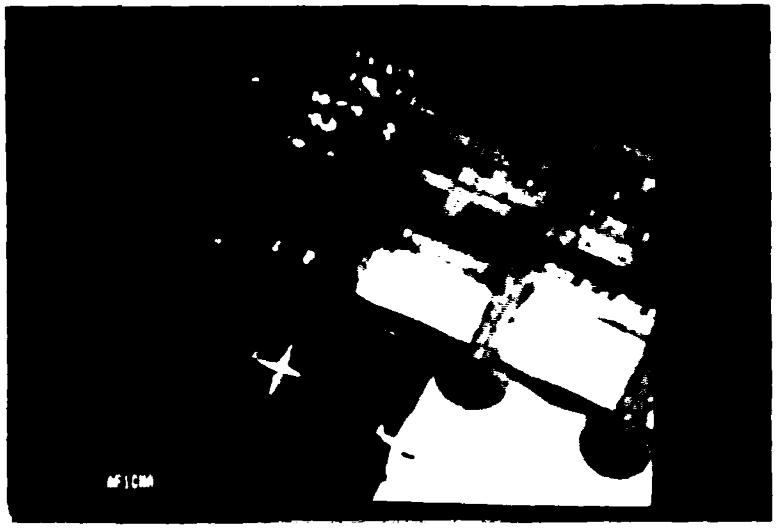


SVD/COS 1.0bpp 0.93%MSE

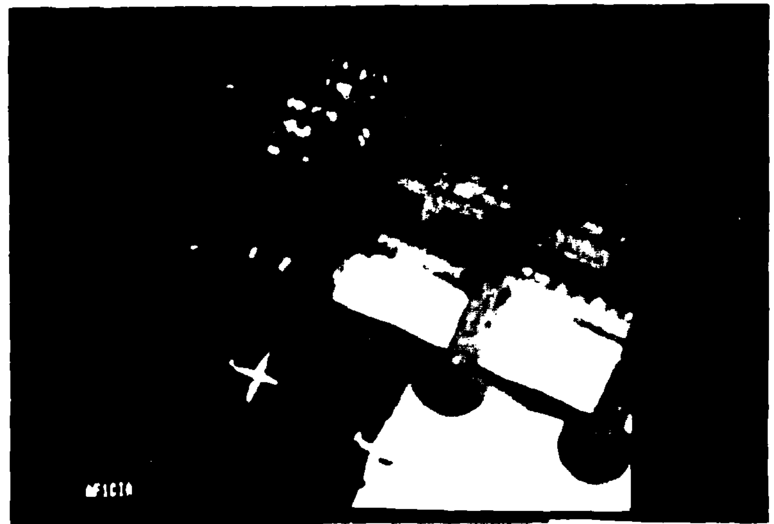


SVD/COS 1.5bpp 0.54%MSE

Figure 7.4



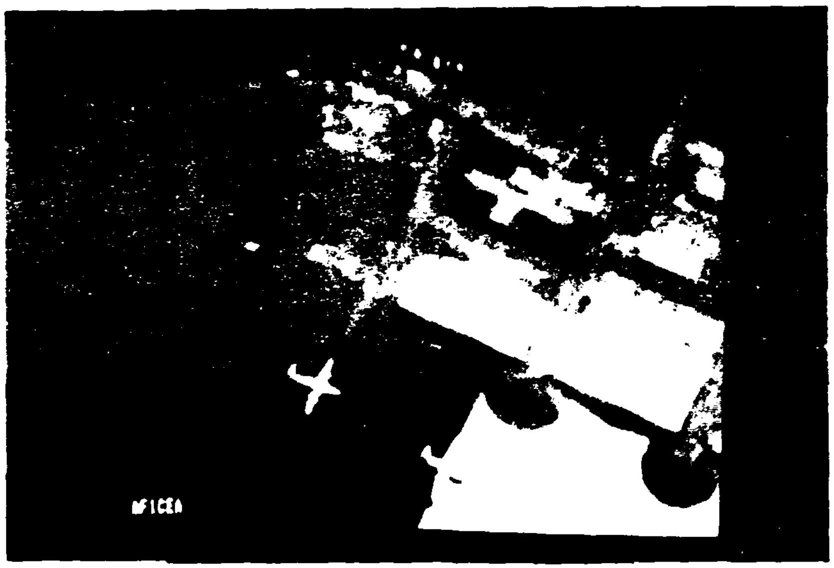
SVD/COS/RO 1.0bpp 0.96%MSE



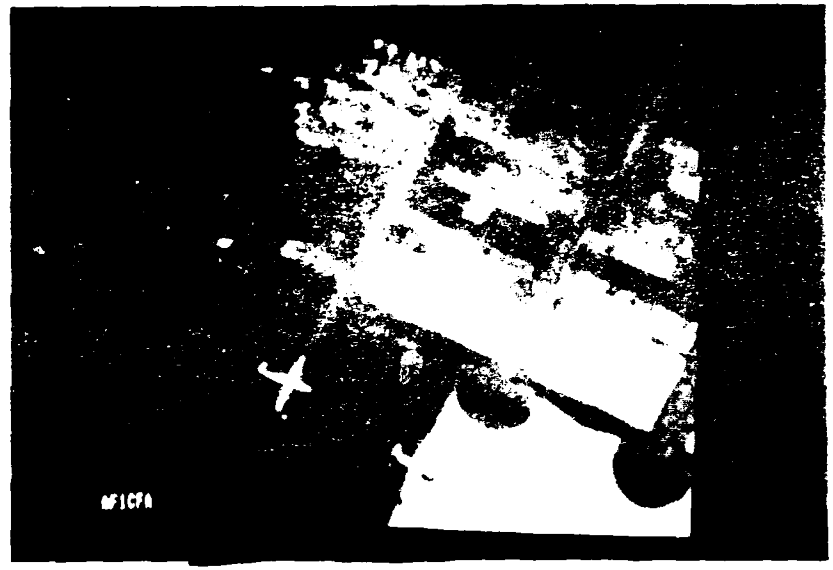
SVD/HAD 1.0bpp 1.4%MSE

SVD/HAD/RO 1.0bpp 1.6% MSE

Figure 7.5.

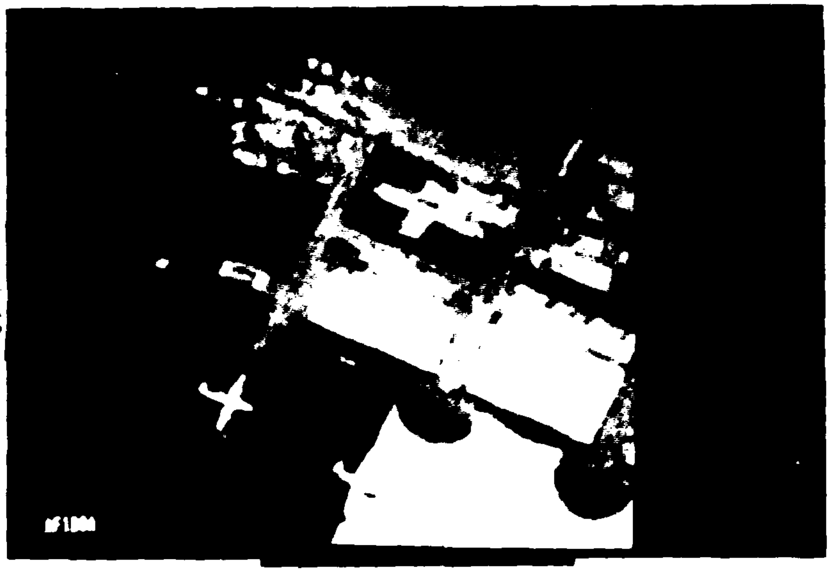


KLT Predictive 1.0bpp 0.96%MSE

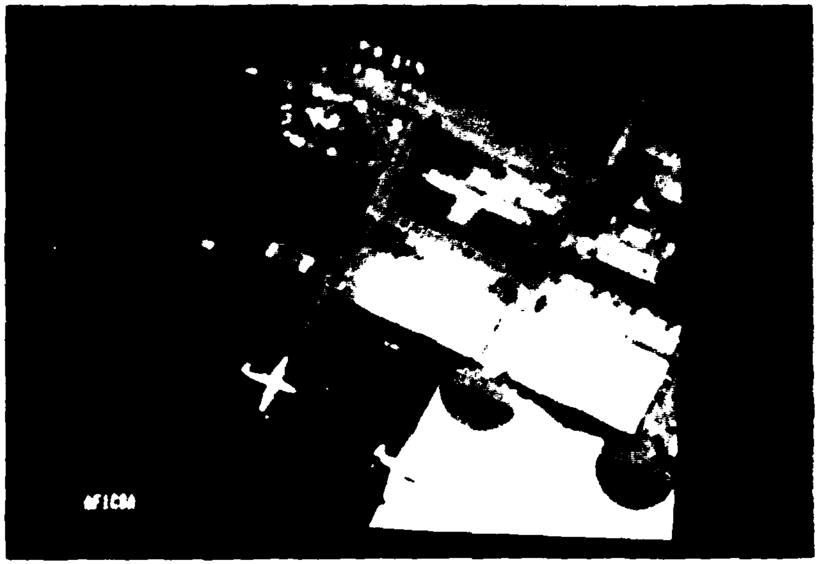


KLT Emperical 1.0bpp 0.52%MSE

Figure 7.6

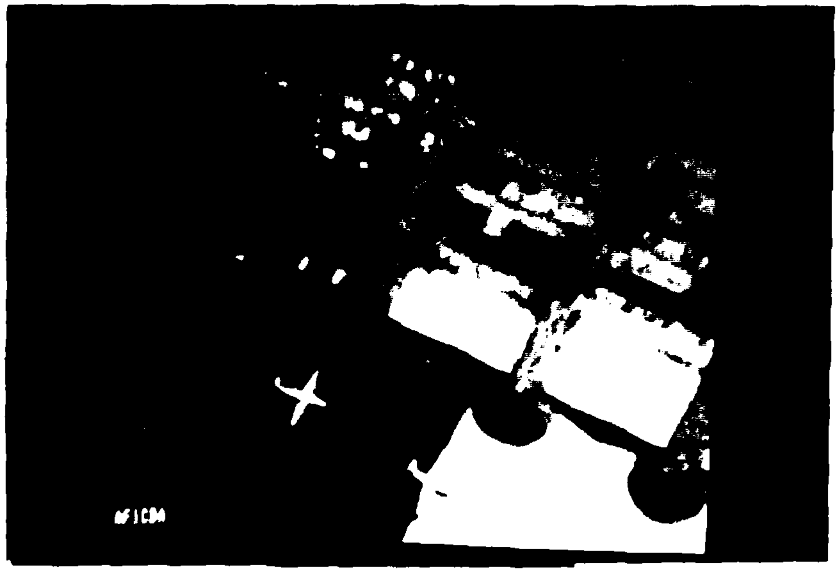


COS 0.56bpp 1.26%MSE



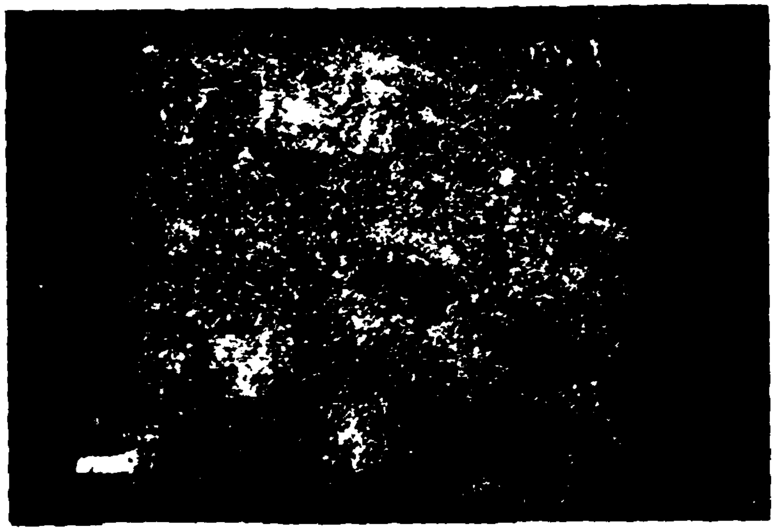
COS 1.0bpp 0.51%MSE

Figure 7.7

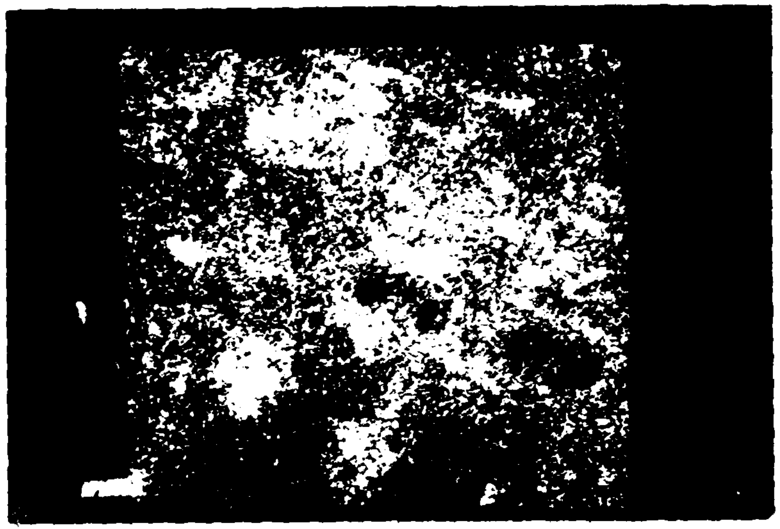


HAD 1.0bpp 1.1%MSE

Figure 7.8

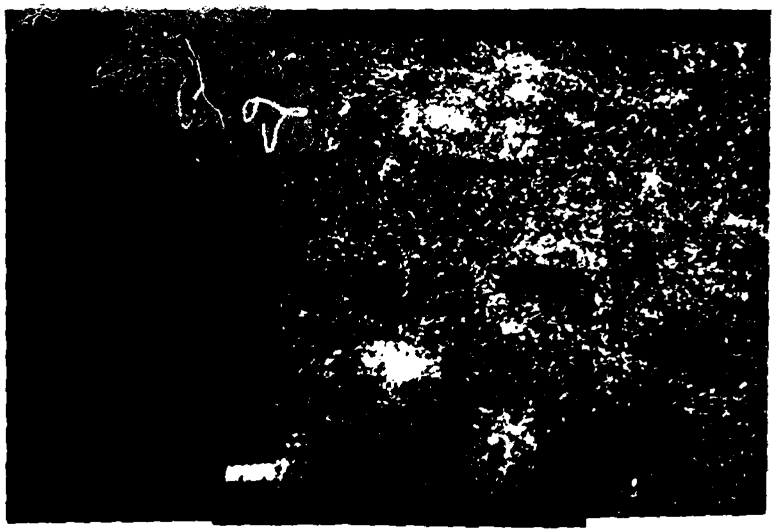


COS Error 1.0bpp



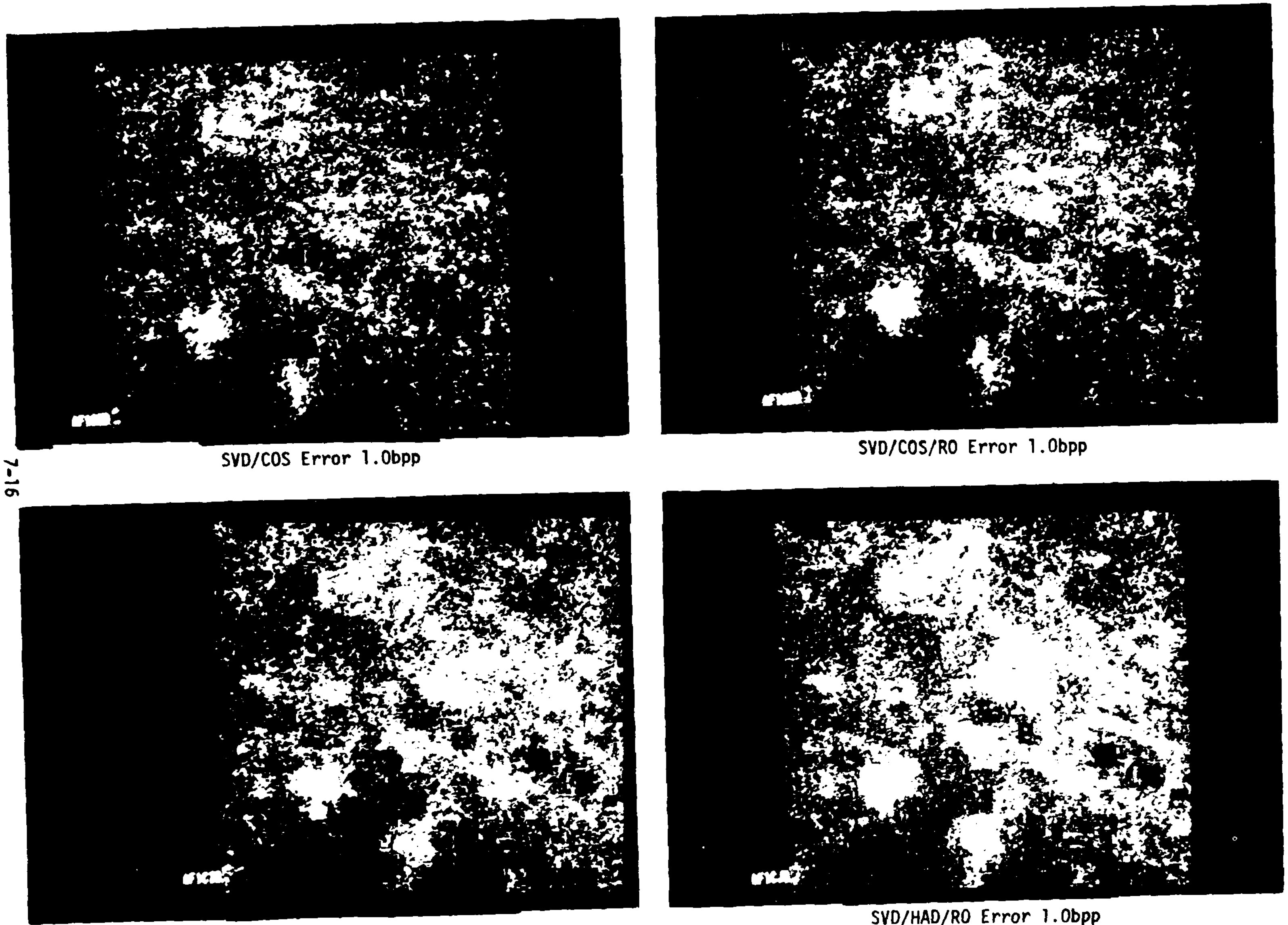
HAD Error 1.0bpp

KLT Predictive Error 1.0bpp



KLT Emperical Error 1.0bpp

Figure 7.9



SVD/COS Error 1.0bpp

SVD/COS/RO Error 1.0bpp

SVD/HAD Error 1.0bpp

SVD/HAD/RO Error 1.0bpp

Figure 7.10

First, the COS and KLT/E algorithms are subjectively best. They are indistinguishible in terms of subjective performance and yield the best reconstructed images, both in terms of edge crispness and continuity and faithful texture rendition. The COS and KLT/E produce the lowest brightness error images with the least structure in them. They also produce the smallest number of very bright error pixels. Reconstructed images retain all important detail at all three bit rates: 1.5, 1.0, and 0.5 bpp.

Second, the SVD/COS algorithm performs subjectively well. It is subjectively indistinguishable from the SVD/COS/RO algorithm. It renders detail well at 1.5 and 1.0 bpp, although it is inferior to both COS and KLT/E at these rates. This inferiority is evidenced in several catagories, including:

- Crispness of edges in reconstructed images,
- Rendition of texture in reconstructed images, and
- Intensity of error images.

On the other hand, the SVD/COS is approximately equivalent to COS and KLT/E in terms of the structure in the error images and the number of very bright error image pixels.

Third, the KLT/P is, subjectively, considerably inferior to all three of the COS, KLT/E, and the SVD/COS (and SVD/COS/RO) algorithms. This inferiority is consistent across all bit rates and is refelcted in all the subjective measures just discussed.

Last, the HAD, SVD/HAD and SVD/HAD/RO are worst in all catagories. Especially noticeable is the error image structure, which appears to accurately capture the essential structural information in the original image. Since good performance dictates having uncorrelated error and reconstructed imagery, this is an indication of poor coding performance.

Orthodonal Expansion and Repolarization Enhancements

Due to the apparent poor showing of the SVD-based algorithms with respect to the COS baseline, the orthogonal expansion and repolarization enhancements were developed as a final attempt at optimizing the SVD coder. These enhancements represented an additional effort to exploit the last possible sources of redundancy in the SVD decomposition in order to extract as high a degree of performance as possible.

Both enhancements were evaluated, and neither improved the SVD/COS performance markedly. The SVD/COS/ORTH was slightly better in terms of MSE and MAE, but the difference was similar to the small difference between the KLT/E and COS algorithms, and no subjective difference was apparent. The repolarization enhancement produced similar results, but in its case the objective performance was slightly poorer, while the subjective performance was indistinguishable.

Conclusions of Preliminary Algorithm Evaluation

The following points summarize the preliminary evaluation results:

- COS is superior to SVD/COS,
- KLT/E and COS are tied,
- SVD/COS is superior to SVD/HAD,
- KLT/E is superior to KLT/P,
- HAD performed worst,
- Reordering does not improve SVD coding,
- Orthogonal expansion does not markedly improve SVD coding, and
- Repolarization does not improve SVD coding.

Based on these findings, the best algorithm in each catagory can be identified:

- Baseline : COS
- KLT : KLT/E
- SVD : SVD/COS

7.2 Comprehensive Evaluation

This subsection reports on the results of the comprehensive algorithm evaluation. Four algorithms were applied:

- COS : cosine baseline,
- HAD : Hadamard baseline,
- SVD : SVD/COS algorithm, and
- KLT : KLT/E algorithm.

These algorithms were applied to four test images, each one a 256 X 256 subset of a larger GFE image. These images were:

- Visible airfield,
- Visible harbor scene,
- Infrared airfield, and
- SAR airfield.

Each algorithm was applied to each image at three different bit rates:

- Visible airfield : 0.5, 1.0, 1.5 bpp
- Visible harbor : 0.5, 1.0, 1.5 bpp,
- Infrared airfield : 0.25, 0.5, 1.0 bpp, and
- SAR airfield : 0.5, 1.0, 1.5 bpp.

The total number of image encodings/decodings was thus 48.

Figures 7-11 through 7-26 show the original and coded images involved in the evaluation. Figures 7-11 through 7-14 pertain to the visible airfield image, Figures 7-15 through 7-18 to the harbor scene, Figures 7-19 through 7-22 to the IR image, and Figures 7-23 through 7-26 to the SAR image.

Figures 7-27 through 7-34 give a summary of objective coding performance measures in terms of MSE and MAE versus coding rate.

These results can be summarized as follows:

- MSE, MAE, and subjective evaluation yield same conclusions,
- COS and KLT perform equally well and best,
- Next is SVD,
- Poorest is HAD.

In terms of computational load, the following rank ordering applied:

- HAD is lowest,
- COS is next,

- KLT is third, and

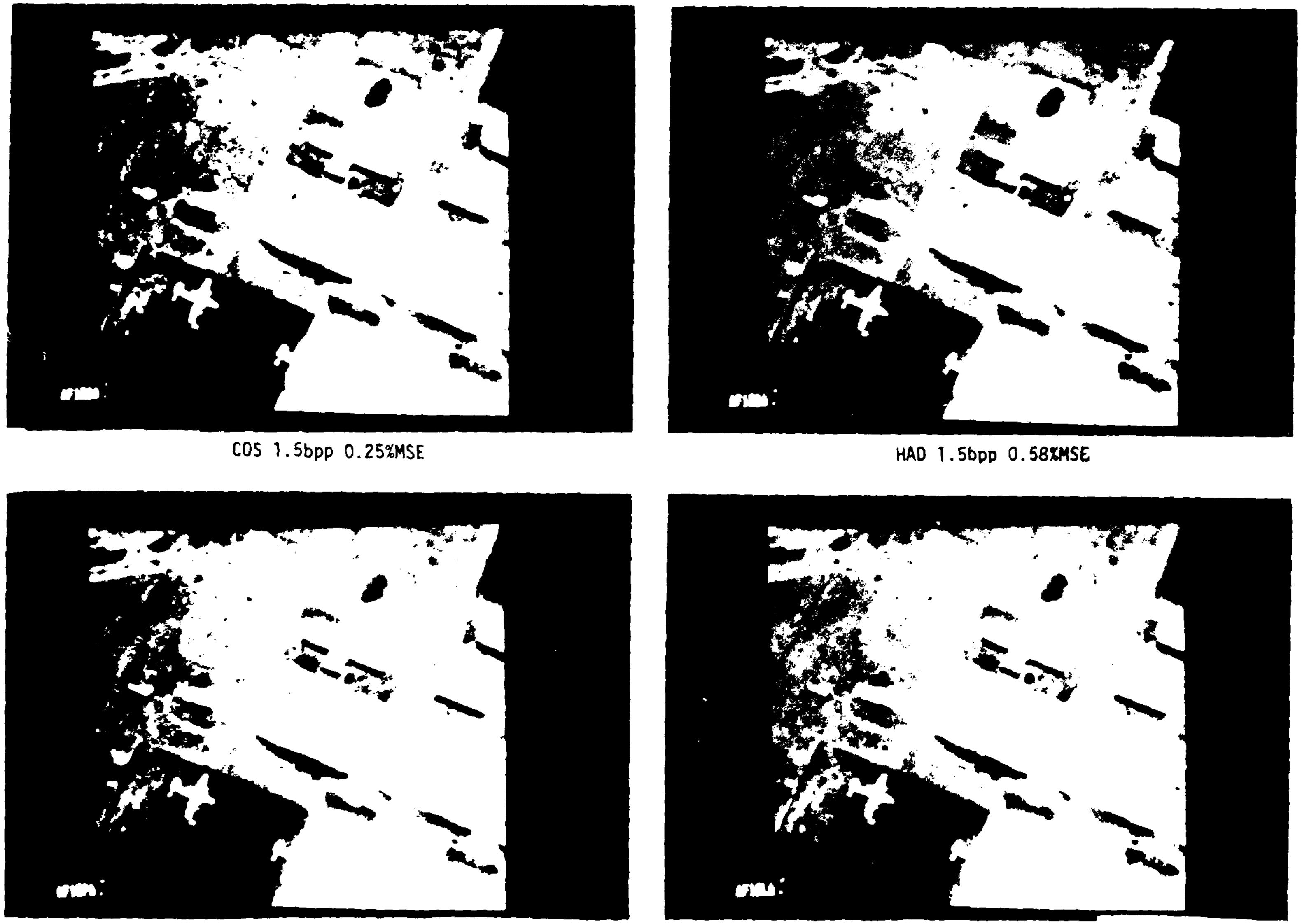
- SVD is significantly highest.

In terms of extendability to special purpose hardware, both the COS and HAD are good candidates owing to their "fast" algorithms. In fact, special purpose hardware for 1D versions of these transforms already exists. The KLT could be implemented in special purpose hardware, but it would be significantly more cumbersome due to the requirement to perform full matrix multiplications. A special purpose hardware implementation of the SVD is not so practical, owing to its reliance upon an iterative procedure which is not guaranteed to converge in a finite number of steps.

In summary, taken together, these observations point to the conclusion that the cosine transform coder is the best algorithm amongst those tested. In certain applications where computational efficiency is paramount, the Hadamard algorithm may be warranted. However, neither coding nor computational efficiency seems to favor the KLT or SVD in any case.

Visible Airfield 8bpp

Figure 7.11



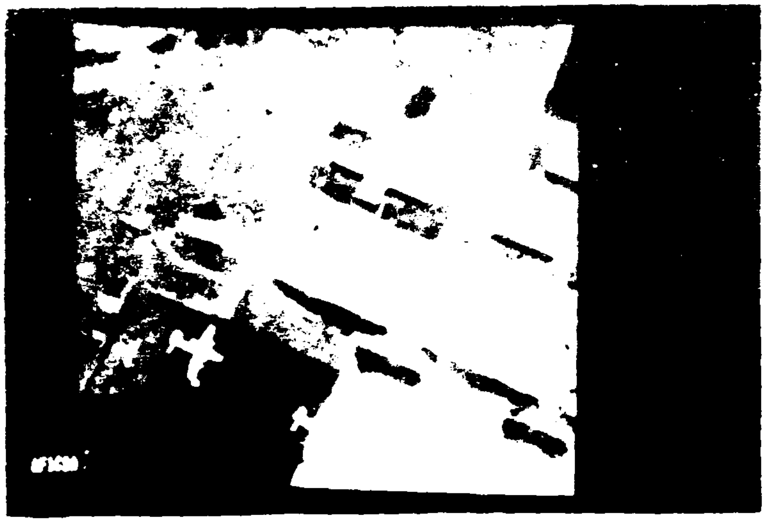
COS 1.5bpp 0.25%MSE

HAD 1.5bpp 0.58%MSE

KLT 1.5bpp 0.28%MSE

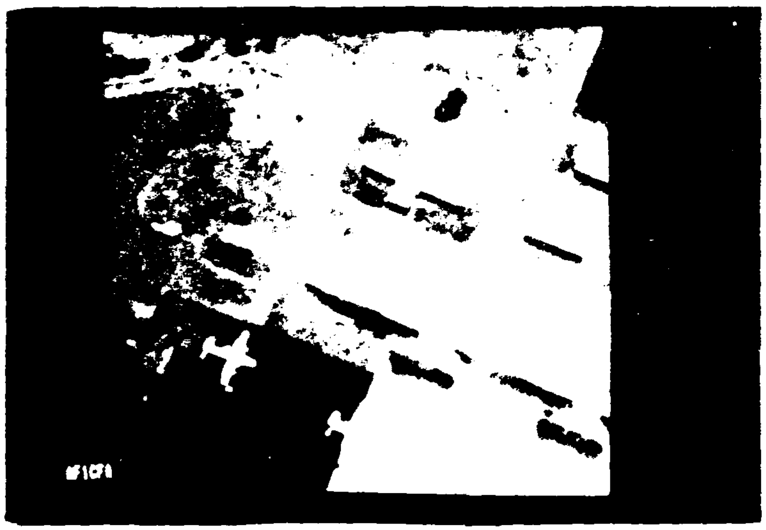
SVD 1.5bpp 0.54%MSE

Figure 7.12



COS 1.0bpp 0.50%MSE

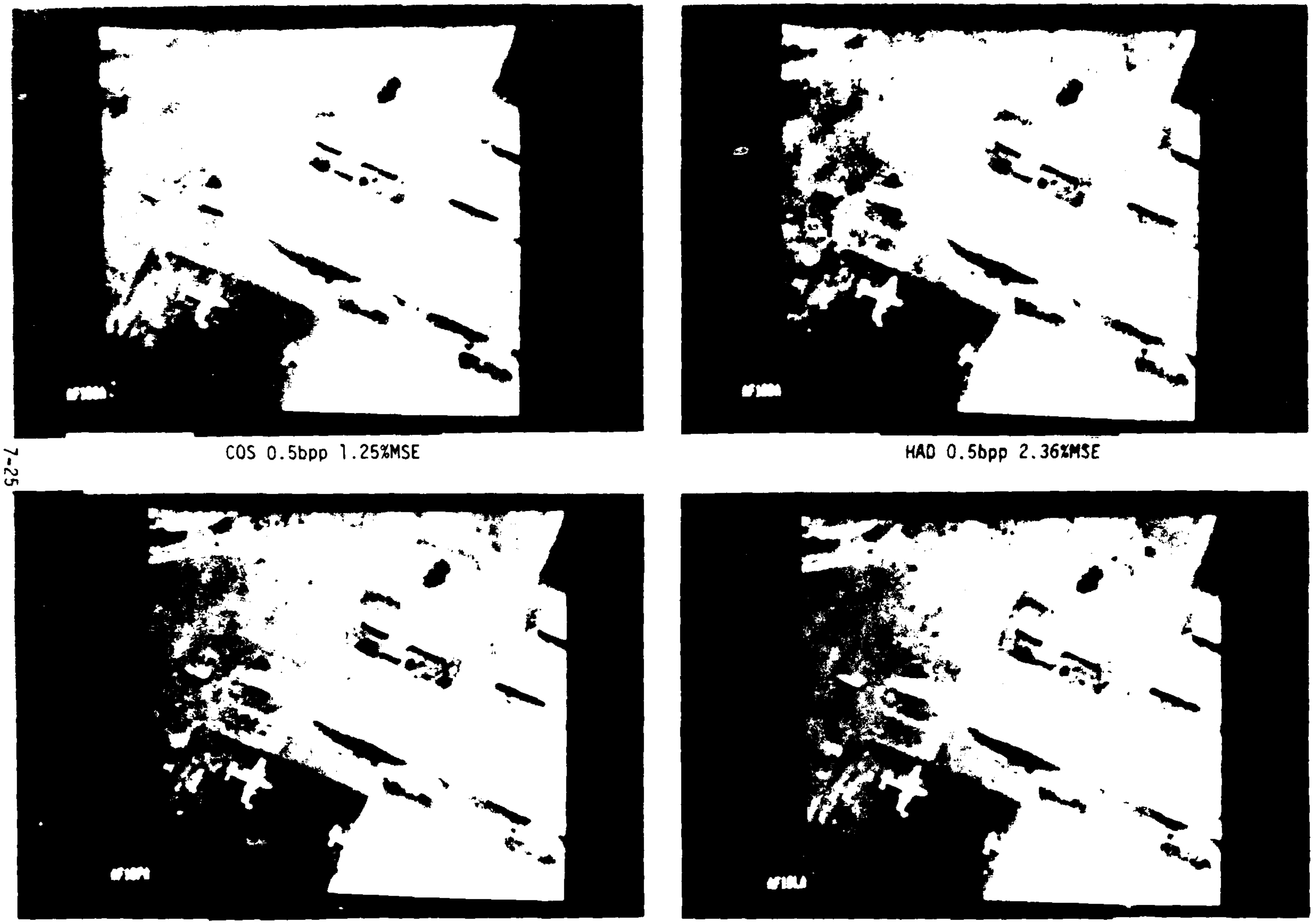
HAD 1.0bpp 1.10%MSE



KLT 1.0bpp 0.53%MSE

SVD 1.0bpp 0.93%MSE

Figure 7.13



COS 0.5bpp 1.25%MSE

HAD 0.5bpp 2.36%MSE

KLT 0.5bpp 1.30%MSE

SVD 0.5bpp 1.94%MSE

Figure 7.14

Visible Harbor 8bpp

Figure 7.15

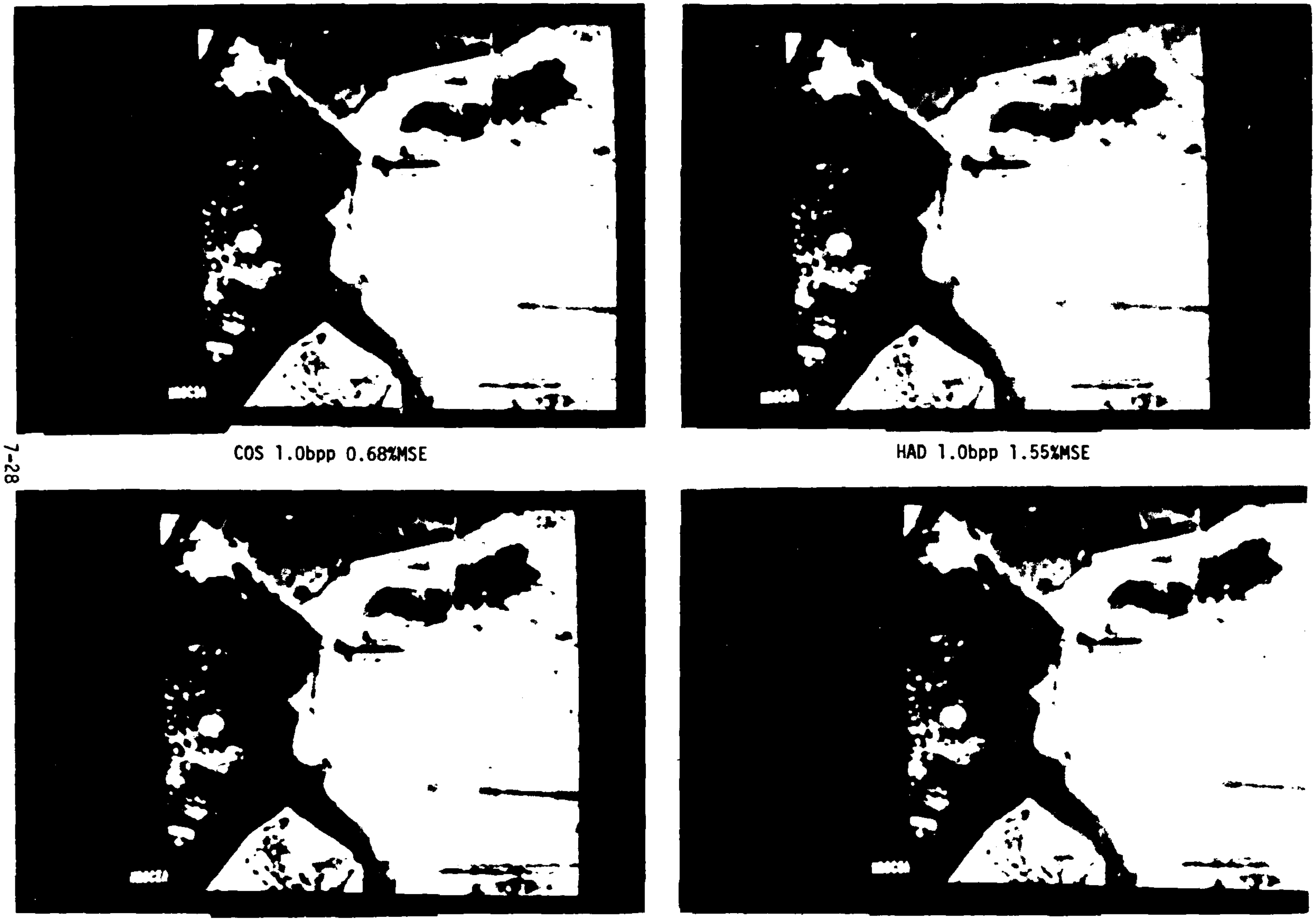
COS 1.5bpp 0.35%MSE

HAD 1.5bpp 0.79%MSE

KLT 1.5bpp 0.32%MSE

SVD 1.5bpp 0.77%MSE

Figure 7.16



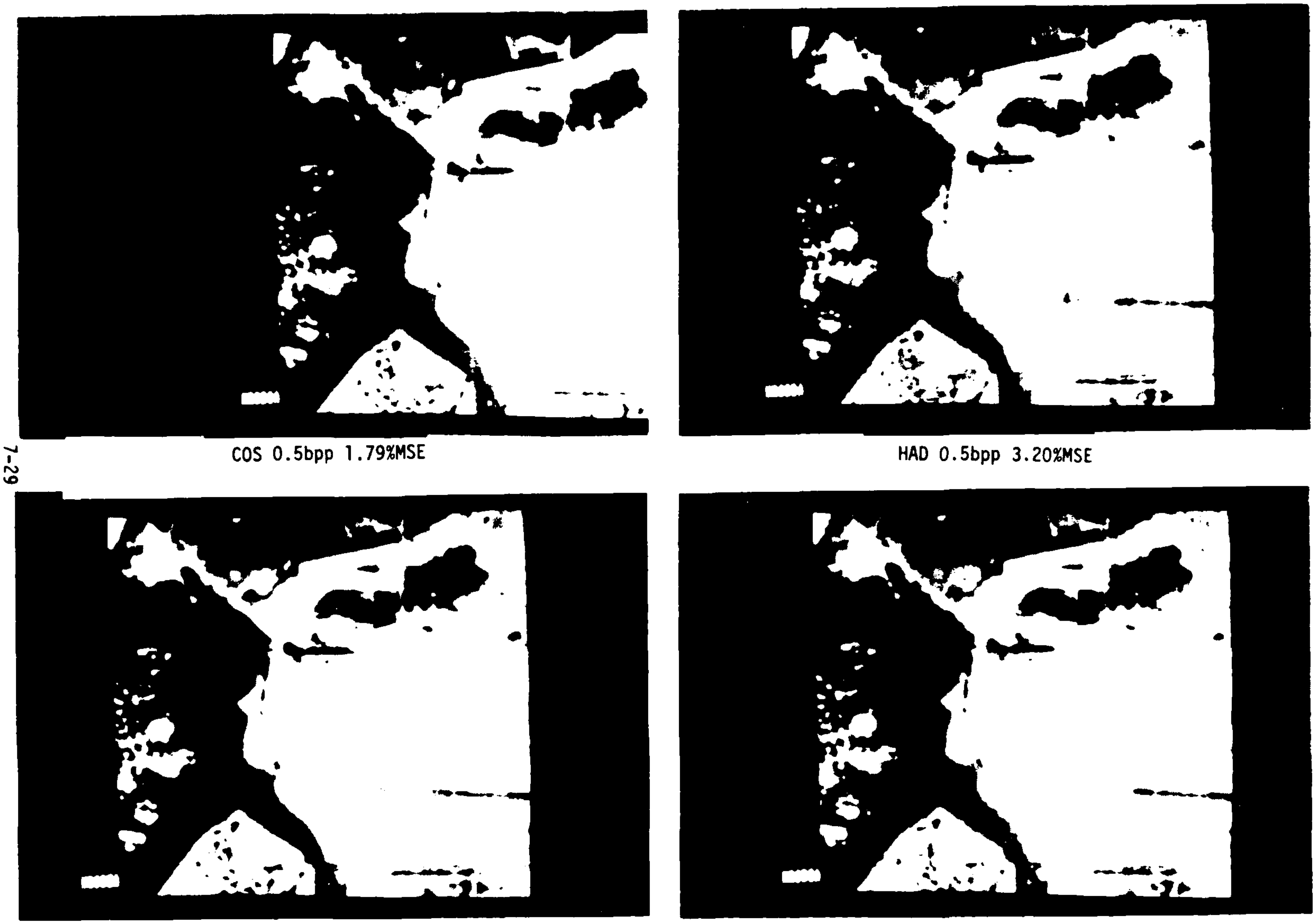
COS 1.0bpp 0.68%MSE

HAD 1.0bpp 1.55%MSE

KLT 1.0bpp 0.66%MSE

SVD 1.0bpp 1.33%MSE

Figure 7.17



COS 0.5bpp 1.79%MSE

HAD 0.5bpp 3.20%MSE

KLT 0.5bpp 1.80%MSE

SVD 0.5bpp 3.16%MSE

Figure 7.18

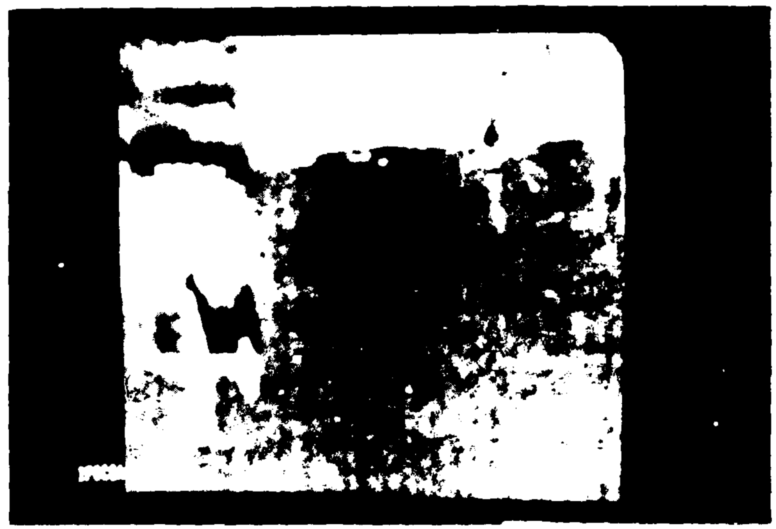
IR Airfield 8bpp

Figure 7.19

COS 1.0bpp 0.71%MSE

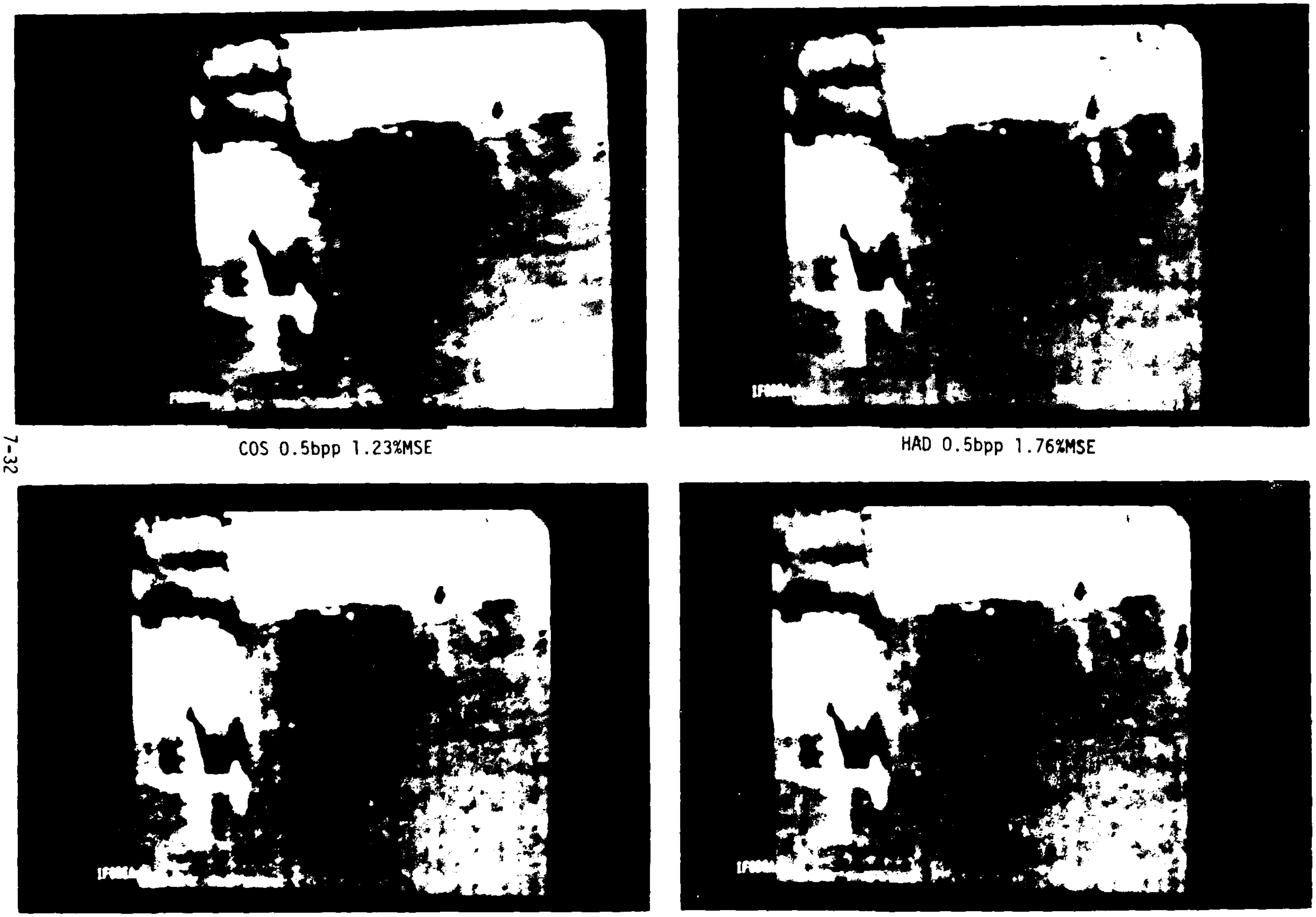
HAD 1.0bpp 1.00%MSE

KLT 1.0bpp 0.72%MSE



SVD 1.0bpp 1.06%MSE

Figure 7.20



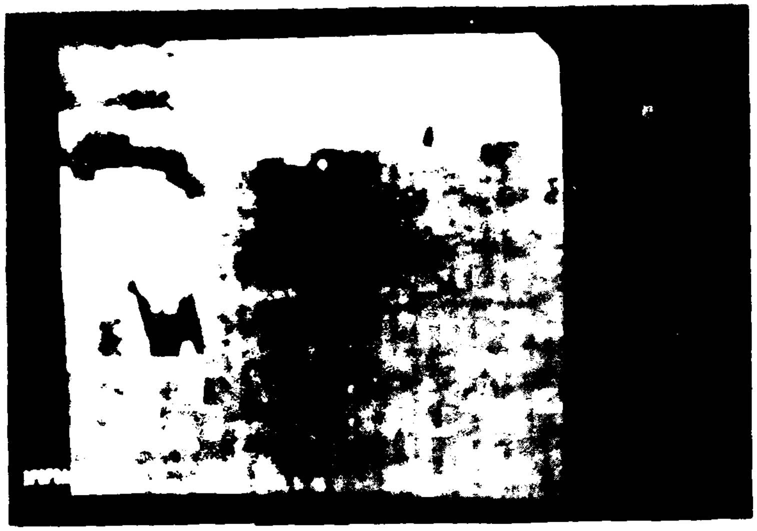
COS 0.5bpp 1.23%MSE

HAD 0.5bpp 1.76%MSE

KLT 0.5bpp 1.23%MSE

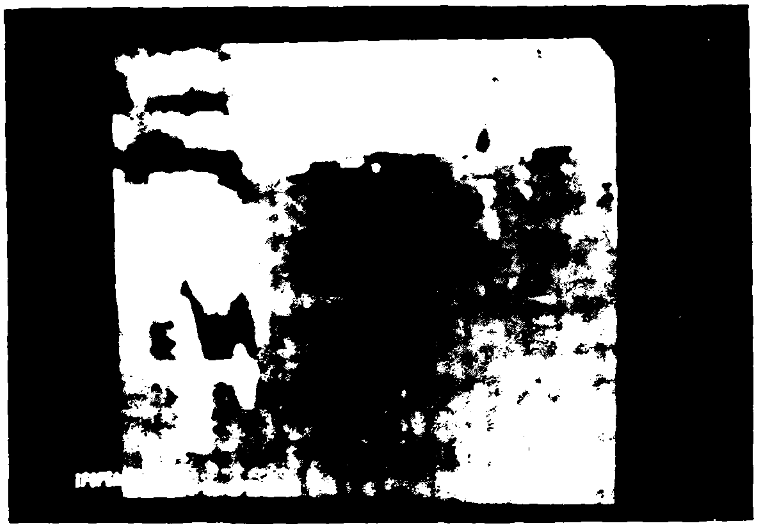
SVD 0.5bpp 1.64%MSE

Figure 7.21



COS .25bpp 1.89%MSE

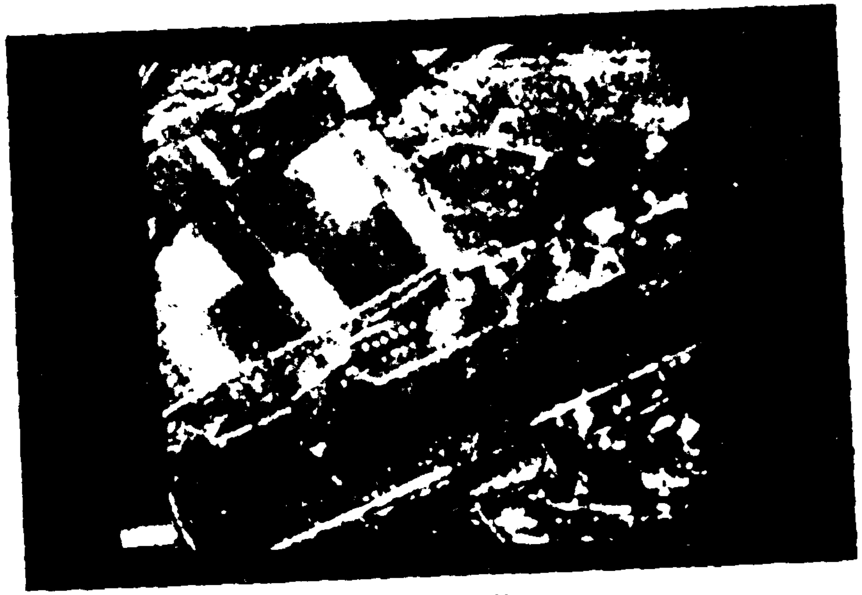
HAD .25bpp 2.60%MSE



KLT .25bpp 1.90%MSE

SVD .25bpp 2.34%MSE

Figure 7.22



SAR Imagery 8bpp

Figure 7.23

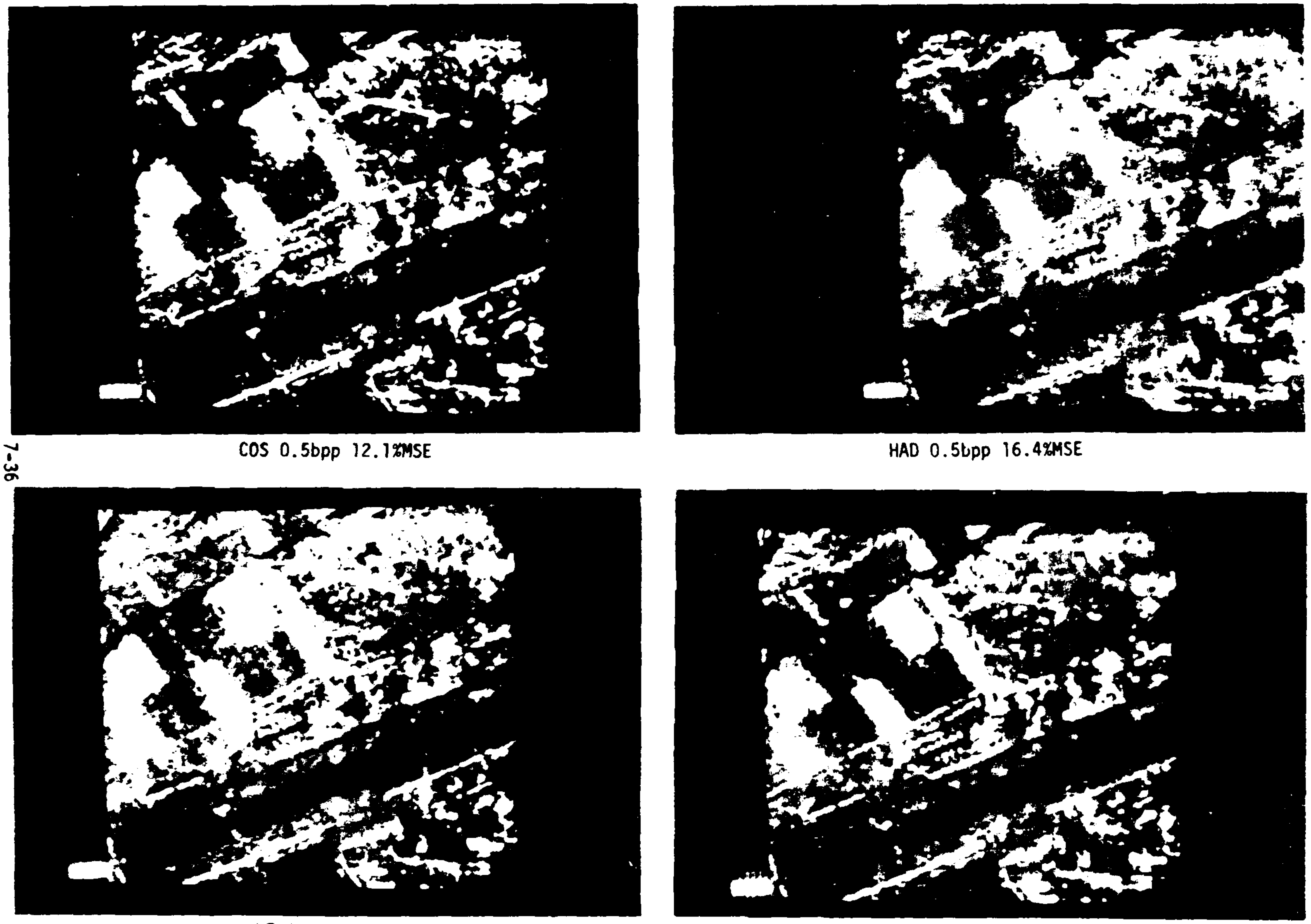
COS 1.0bpp 6.47%MSE

HAD 1.0bpp 8.82%MSE

KLT 1.0bpp 6.42%MSE

SVD 1.0bpp 9.73%MSE

Figure 7.24



COS 0.5bpp 12.1%MSE

HAD 0.5bpp 16.4%MSE

KLT 0.5bpp 13.0%MSE

SVD 0.5bpp 17.7%MSE

Figure 7.25

COS .25bpp 20.4%MSE

HAD .25bpp 26.3%MSE

KLT .25bpp 21.3%MSE

SVD .25bpp 29.2%MSE

Figure 7.26

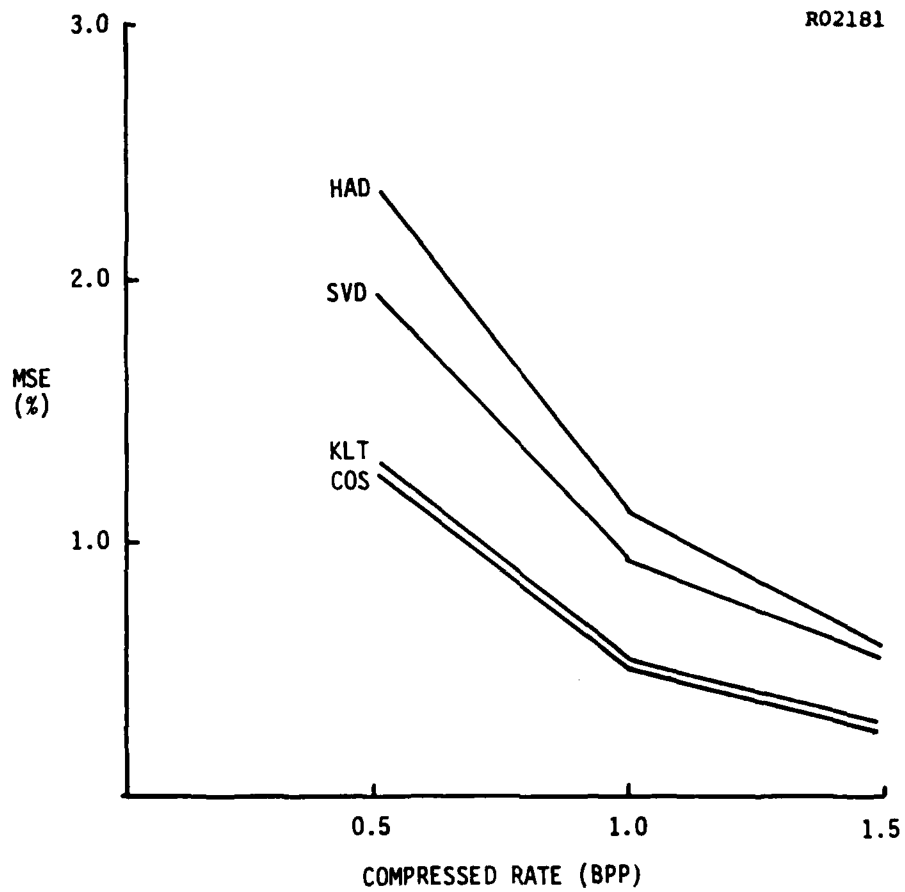
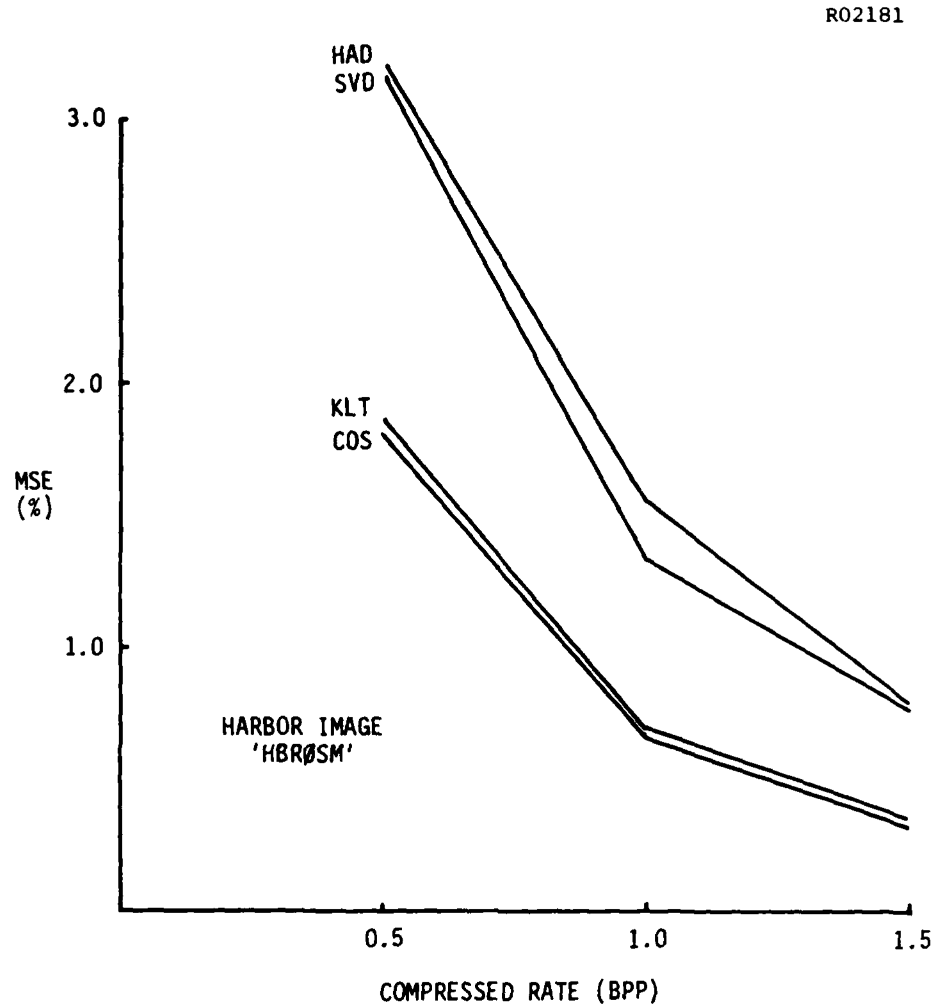


Figure 7-27. Coding Algorithm Comparisons;
MSE versus Rate, Airfield Image



Fugure 7-28. Coding Algorithm Comparisons;
MSE versus Rate, Harbor Image

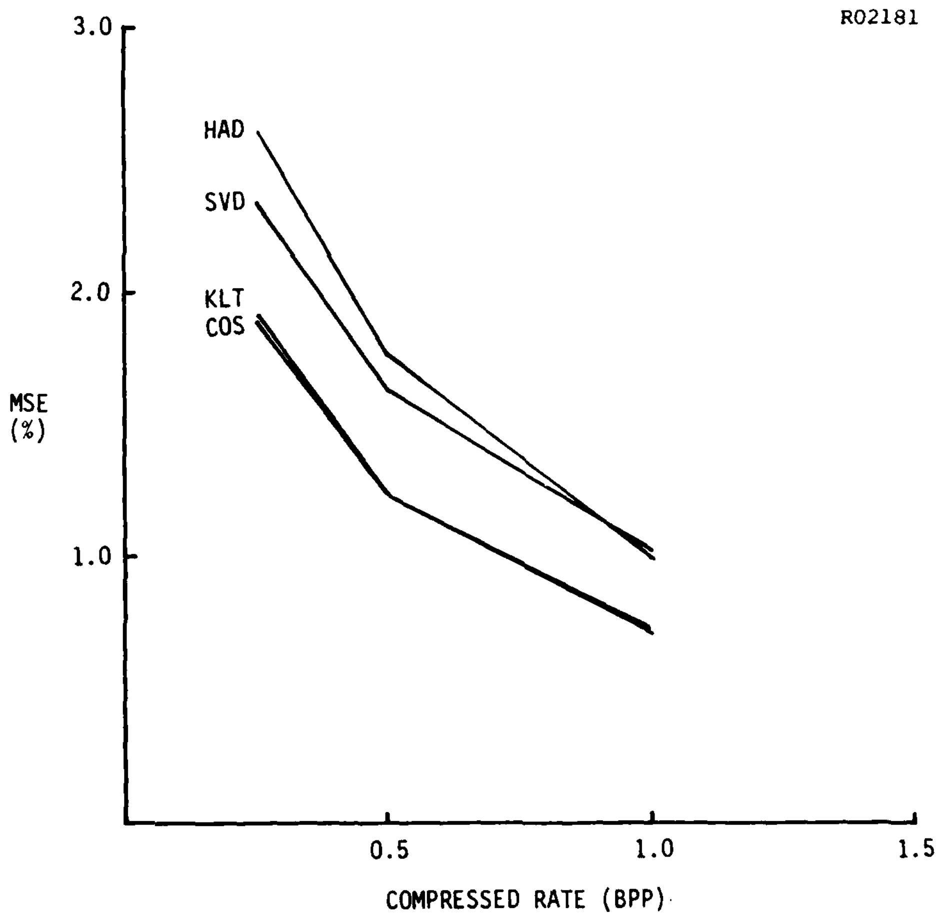


Figure 7-29. Coding Algorithm Comparisons;
MSE versus Rate, IR Image

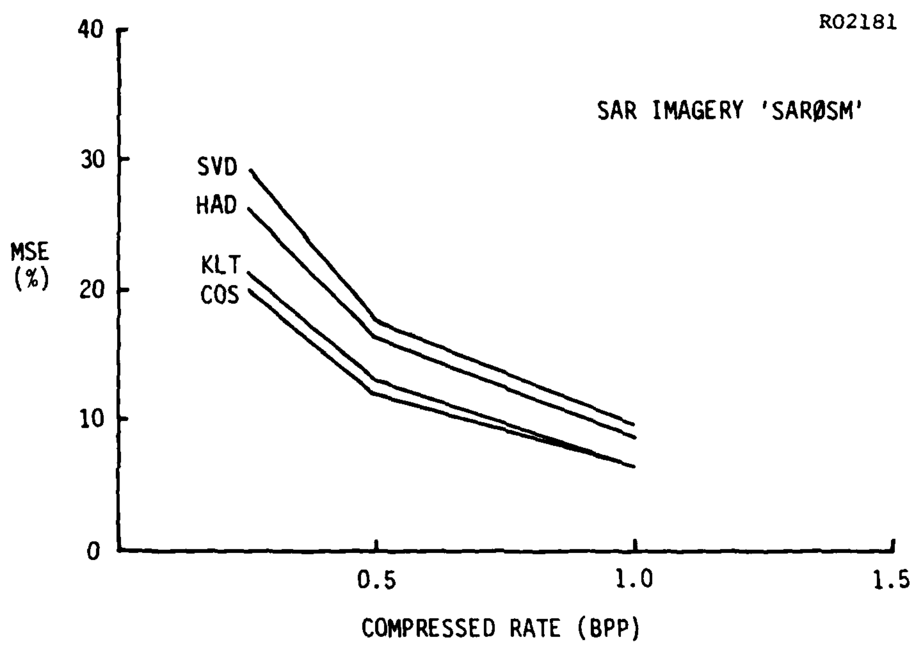


Figure 7-30. Coding Algorithm Comparisons;
MSE versus Rate, SAR Image

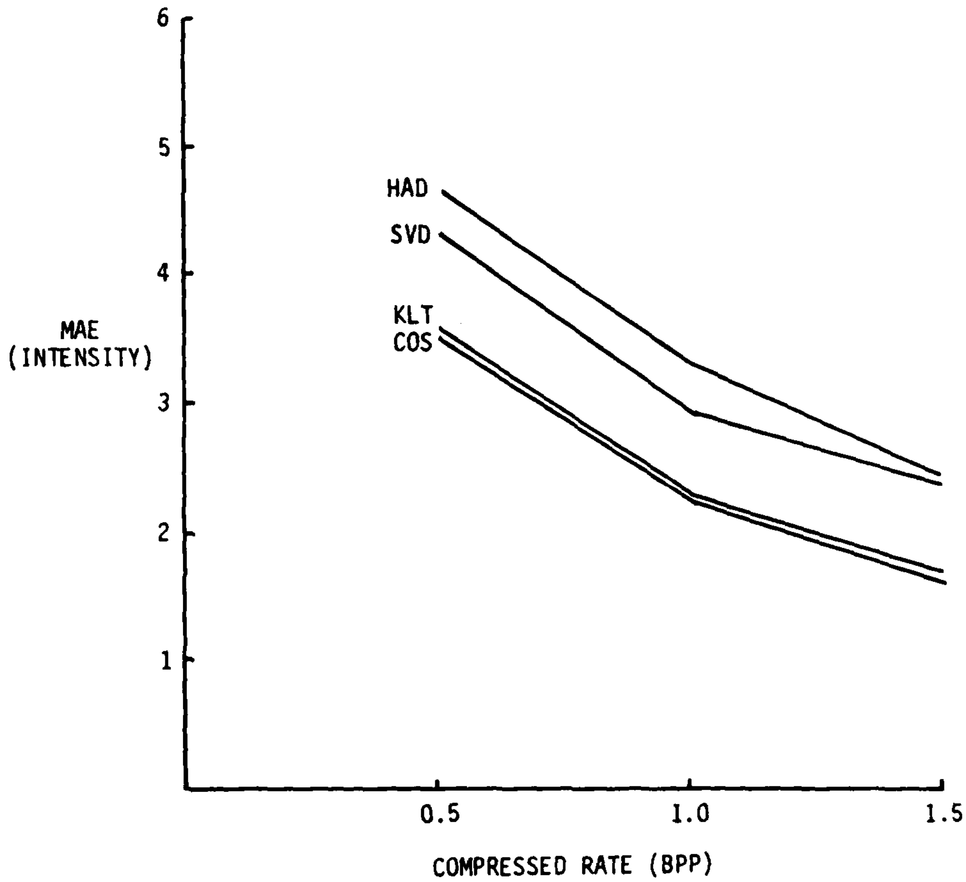


Figure 7-31. Coding Algorithm Comparisons;
MAE versus Rate, Airfield Image

R02181

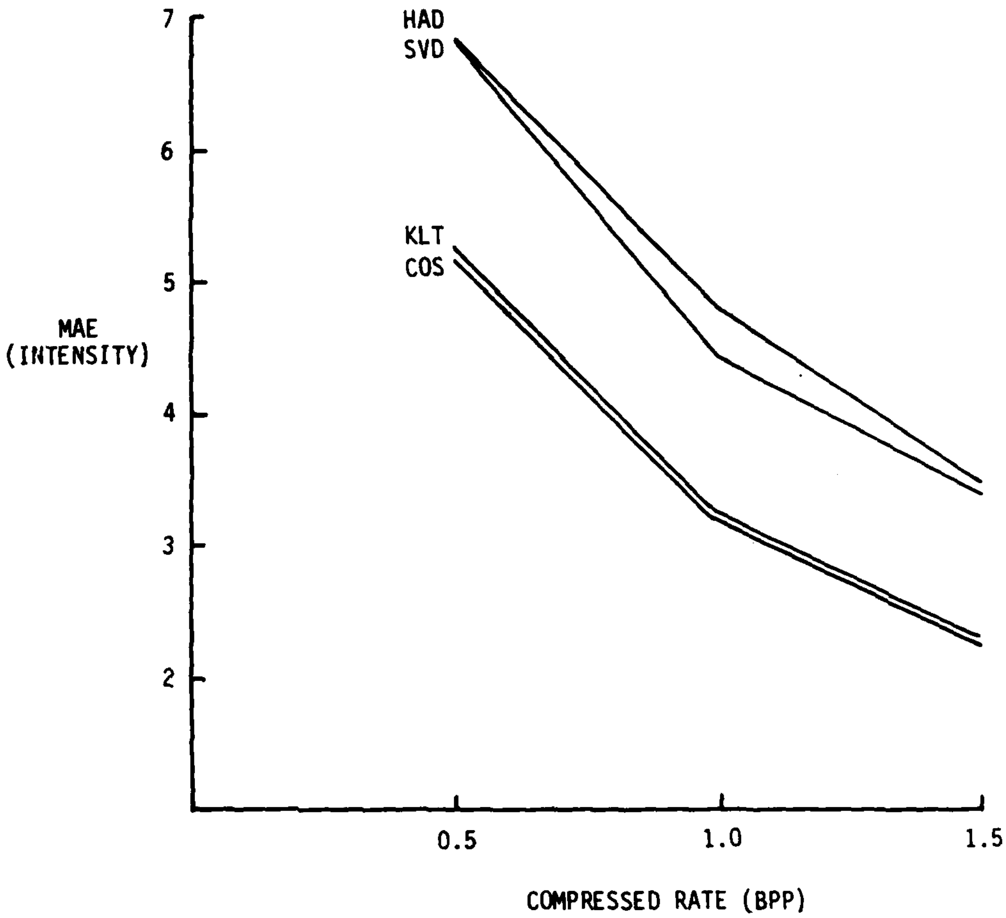


Figure 7-32. Coding Algorithm Comparisons;
MAE versus Rate, Harbor Image

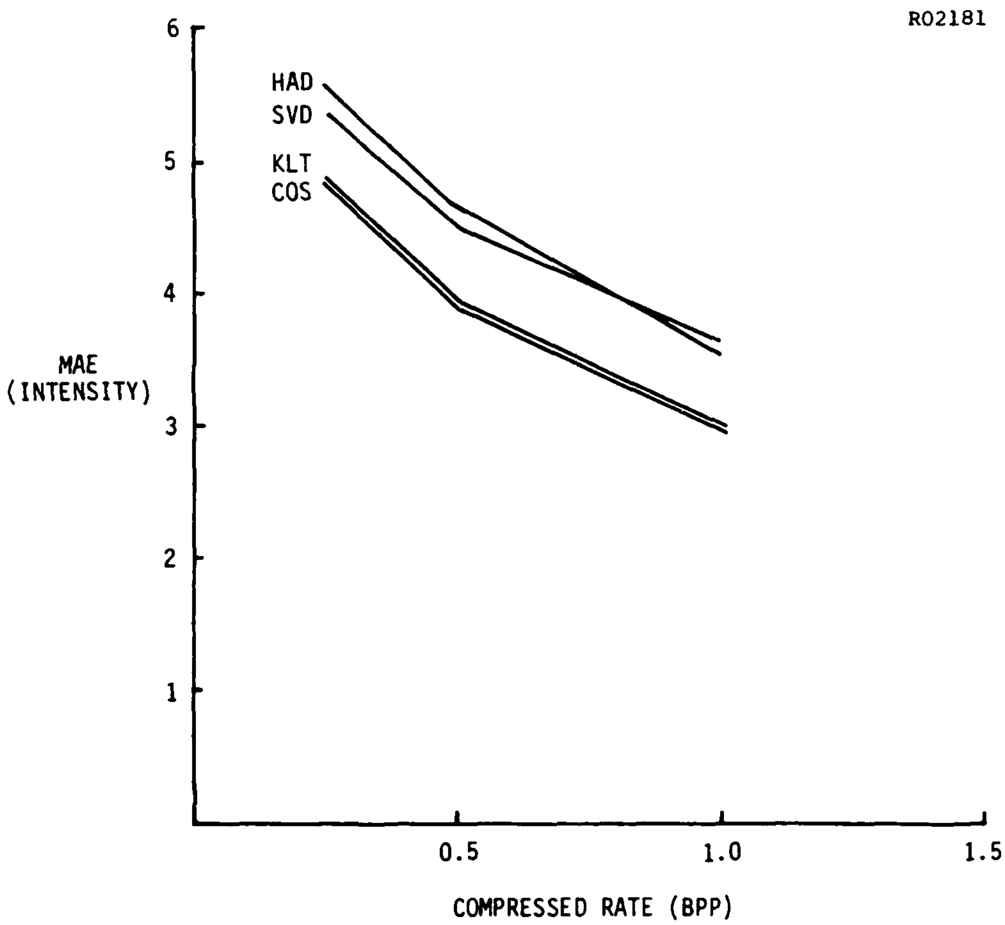


Figure 7-33. Coding Algorithm Comparisons;
MAE versus Rate, IR Image

R02181

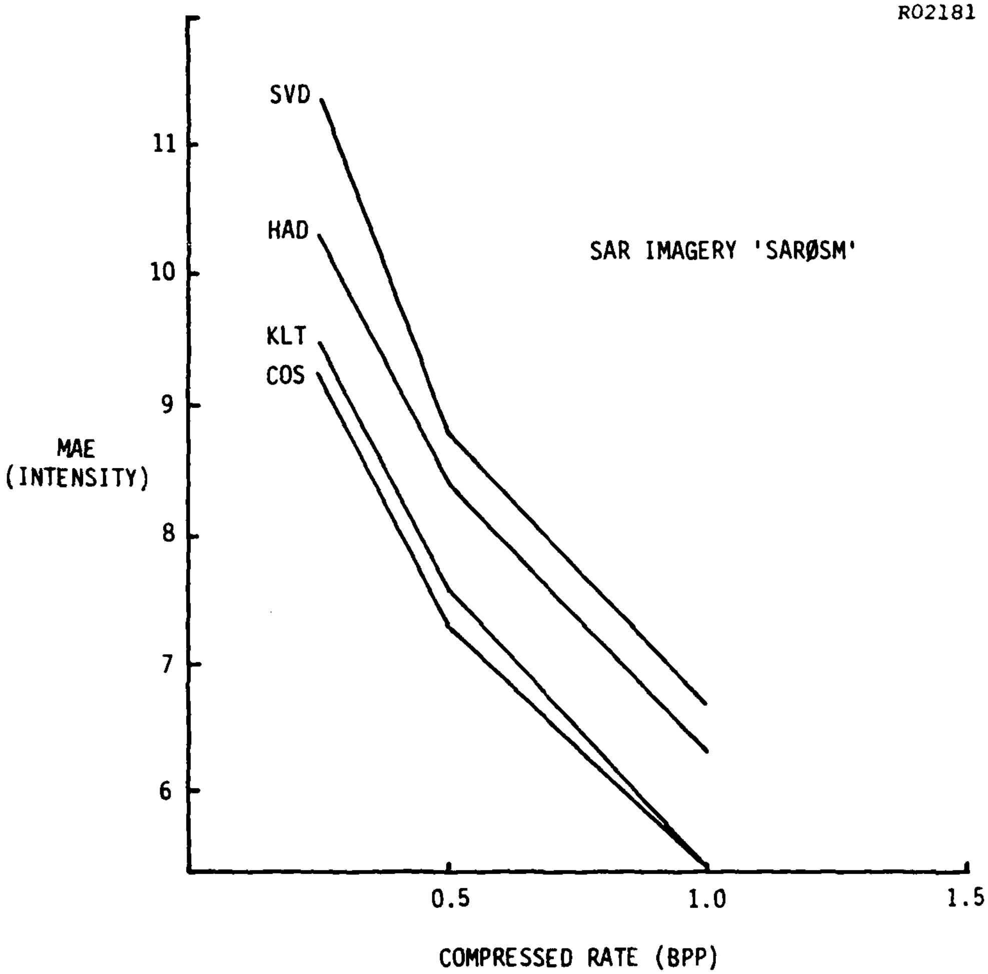


Figure 7-34. Coding Algorithm Comparisons;
MAE versus Rate, SAR Image

8.0 CONCLUSIONS

The principal conclusion of this study is that among the algorithms tested, the cosine transform algorithm appears to be the best performer in terms of coding efficiency. Computational efficiency points to the Hadamard algorithm as a better choice, but it suffers a significant performance degradation compared to cosine as a price. The SVD based algorithms displayed coding efficiency intermediate to the cosine and Hadamard, but computational efficiency worse than both. Because special purpose hardware can be used to implement it efficiently, the cosine approach appears best for applications requiring the highest degree of compression with the smallest coding distortion.

More generally, results point to the success of compressing various eight-bit images down to at least 1.0 bit per pixel using the better transform technques. In several cases, good performance down to 0.5 bits per pixel was also observed. All transform coders performed well at 1.5 bits per pixel.

Significantly, results demonstrated that although the singular value decomposition produces extremely high energy compaction into a small number of singular values by virtue of its being tailored to the image data, the price of also having to code singular vectors renders the approach less efficient overall than either the tailored-to-class KLT or the fixed cosine approaches. That this observation was constant over an assortment of techniques developed to minimize the bandwidth required for singular vector coding suggests that this conclusion is robust and that the SVD is inherently inferior for image coding applications. In addition, since the Karhunen-Loeve approach yielded performance results comparable to the cosine transform, it can be concluded that it is the tailoring of the coefficient coding process, and not the tailoring of the transform, which is most important in image coding.

All of the algorithms tested achieved tailoring of the coefficient coding process via class-adaptivivity, insuring the allocation of bandwidth to portions of an image where most required. In addition, these algorithms distribute bandwidth within blocks to the most important information that that block contains.

Such adaptivity ensures robustness and the capability to deal with highly non-stationary imagery. The price is that rate equalization is required to achieve target global compression rates. In this study, an approach was employed that guaranteed meeting the specified target rate through a process of predictive rate equalization, which was based on block class populations and class-specific statistics and which avoided trial and error coding.

9.0 REFERENCES

[1] Andrews, H.C., "Two-Dimensional Transforms," in Picture Processing and Digital Filtering (Topics in Applied Physics, T.S. Huang, ed.), Vol. 6, Springer-Verlag, New York, 1975.

[2] A.E. LaBonte and C.J. McCallum, "Image Compression Techniques," RADC Technical Report, RADC-TR-77-405, December, 1977.

[3] A.E. LaBonet and T.E. Rosenthal, "MAPS Image Compression," RADC Technical Report RADC-TR-80-173, May 1980.

[4] A.E. LaBonte, "Two Dimensional Image Coding by Micro-Adaptive Picture Sequencing," in Applications of Digital Image Processing, A.G. Tescher, Ed., SPIE Vol. No. 119, 1977.

[5] D.G. McCaughey and H.C. Andrews, "Image Approximation by Variable Knot Bicubic Splines," IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol PAMI-3, Number 3, May 1981, pp 299-310.

[6] A.G. Tescner, "Transform Image Coding", in Image Transmission Techniques, W.K. Pratt, Ed., Academic Press, New York, NY, 1979.

[7] Andrews, H.C. and W.K. Pratt, "Fourier Transform Coding of Images," Hawaii International Conference on System Science, 677-679, 1968.

[8] Ahmed, N., T. Natarajan, and K. Rao, "Discrete Cosine Transform," IEEE Trans. Comput., C-23, 90-93, 1974.

[9] Chen, W. and C.H. Smith, "Adaptive Coding of Color Images Using Cosine Transform," ICCC76, IEEE Catalog Number 76H1085-0 CSCB, 47-7 through 47-13, 1976.

[10] Pratt, W.K., J. Kane, and H.C. Andrews, "Hadamard Transform Image Coding," Proc. IEEE, 57, 58-68, 1969.

[11] Andrews, H.C. and C.L. Patterson, "Singular Value Decompositions and Digital Image Processing," IEEE Trans. Acoust., Speech, and Signal Process., ASSP-24, 26-53, 1976.

[12] J.J. Dongarra et al, LINPACK User's Guide, Society for Industrial and Applied Mathematics, Philadelphia, PA (1979).

APPENDIX A

DERIVATION OF THE SEPARABLE KARHUNEN-LOEVE TRANSFORMATION AND ASSOCIATED STATISTICS

In this appendix the separable KLT is derived and the predicted statistics of the resulting KLT coefficients are determined from the block mean and row and column covariance matrices.

A.1 Model

A block X is assumed to possess a separable covariance function. Such a situation can be modeled by assuming that block X is generated by an outer product matrix multiplication on a zero-mean, stationary white matrix:

$$X = HWG^t + \bar{X} \tag{A.1}$$

where

- H and G are normalized so that

$$\mathrm{tr}\, H^t H = n$$

$$\mathrm{tr}\, G^t G = n \tag{A.2}$$

- W is an n x n matrix of σ^2 variance, uncorrelated, zero mean random variables $W = [w_{ij}]$, i.e.

$$E\, w_{ij} = 0$$

$$E\, w_{ij}^2 = \sigma^2 \tag{A.3}$$

$$E\, w_{ij}\, w_{pq} = 0 \text{ for } (p,q) \neq (i,j), \text{ and}$$

- The n x n matrix $\bar{X}$ is the mean of the block X.

Figure A-1 illustrates the assumed model. That such a model results in a separable covariance can be verified by determining an expression for the covariance of pixels x_{ij} and x_{pq} in X.

$$c(i,j;\ p,q) = E(x_{ij} - \overline{x}_{ij})\ (x_{pq} - \overline{x}_{pq})$$

$$= E[\underline{e}_i^t(X - \overline{X})\ \underline{e}_j]\ [\underline{e}_p^t(X - \overline{X})\ \underline{e}_q]$$

(Here $\underline{e}_i$ denotes the unit vector with ith element 1 and the rest 0)

$$= E[\underline{e}_i^t\ HWG^t\ \underline{e}_j]\ [\underline{e}_q^t\ GW^t\ H^t\ \underline{e}_p]$$

$$= \underline{e}_i^t\ HE\ \{WG^t\ \underline{e}_j\ \underline{e}_q^t\ GW^t\}\ H^t\ \underline{e}_p$$

$$= \underline{e}_i^t\ H\ [\mathrm{Trace}\ (G^t\ \underline{e}_j\ \underline{e}_q^t\ G)\ \sigma^2\ I]\ H^t\ \underline{e}_p$$

$$= \underline{e}_i^t\ HH^t\ \underline{e}_p \cdot \sigma^2 \cdot \mathrm{Trace}(G^t\ \underline{e}_j\ \underline{e}_q^t\ G)$$

$$= \sigma^2\ (\underline{e}_i^t\ HH^t\ \underline{e}_p)\ (\underline{e}_j^t\ GG^t\ \underline{e}_q)$$

$$= C_V\ (i,p) \cdot C_H\ (j,q) \qquad \text{(A.4)}$$

where $C_V\ (i,p) = \sigma \cdot \underline{e}_i^t\ HH^t\underline{e}_p$, and

$C_H\ (j,q) = \sigma \cdot \underline{e}_j^t\ GG^t\ \underline{e}_q$.

Since c(i,j; p,q) can be written as the product of two functions each separately dependent upon vertical and horizontal pixel displacement, equation (A.1) is seen to model a block with a separable covariance function.

R02181

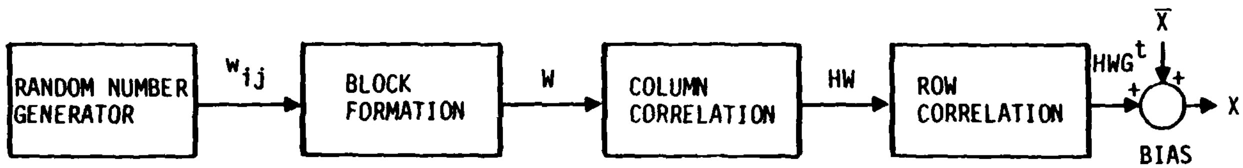


Figure A-1. Separable Covariance Block Model

A.2 KLT Derivation

The objective is to find a separable, unitary transformation from X into another array Z:

$$Z = U^t XV, \tag{A.5}$$

such that the z_{ij} in Z are uncorrelated and have maximum energy compaction into the upper left-hand corner of Z. Note that because U and V are unitary:

$$X = UZV^t, \tag{A.6}$$

i.e., X is recoverable from Z via the inverse unitary operation. Another way of writing (A.6) is:

$$X = \sum_{i,j=1}^{n} z_{ij}\, \underline{u}_i\, \underline{v}_j^t \tag{A.7}$$

where $\underline{u}_i$ is the ith column of U and $\underline{v}_j$ is the jth column of V. X is thus a weighted sum of basis blocks $\underline{u}_i\, \underline{v}_j^t$.

According to Shannon, optimum coding dictates selecting U and V such that the z_{ij} are uncorrelated. The coefficient z_{ij} can be expressed as:

$$z_{ij} = \underline{u}_i^t\, X\, \underline{v}_j \tag{A.8}$$

Therefore

$$E\, z_{ij} = \underline{u}_i^t\, \overline{X}\, \underline{v}_j \triangleq \overline{z}_{ij} \tag{A.9}$$

and

$$E(z_{ij} - \bar{z}_{ij})\,(z_{kl} - \bar{z}_{kl})$$

$$= E\,[\underline{u}_i^t\,(x - \bar{x})\,\underline{v}_j]\,[\underline{v}_l^t\,(x - \bar{x})^t\,\underline{u}_k]$$

$$= E\,[\underline{u}_i^t\,(HWG^t)\,\underline{v}_j]\,[\underline{v}_l^t\,GW^t\,H^t\,\underline{u}_k]$$

$$= \underline{u}_i^t\,HE\,\{WG^t\,\underline{v}_j\,\underline{v}_l^t\,GW^t\}\,H^t\,\underline{u}_k$$

$$= \underline{u}_i^t\,H\,[\text{Trace}\,(G^t\underline{v}_j\,\underline{v}_l^t\,G)\,\sigma^2\,I]\,H^t\,\underline{u}_k$$

$$= (\underline{u}_i^t\,HH^t\,\underline{u}_k)\cdot\sigma^2\cdot\text{Trace}\,(G^t\,\underline{v}_j\,\underline{v}_l^t\,G)$$

$$= \sigma^2\,(\underline{u}_i^t\,HH^t\,\underline{u}_k)\,(\underline{v}_j^t\,GG^t\,\underline{v}_l) \qquad (A.10)$$

Consequently, the z_{ij} will be uncorrelated if

$$\underline{u}_i^t\,HH^t\,\underline{u}_k = 0 \quad \text{for } i \neq k$$

$$\underline{v}_j^t\,GG^t\,\underline{v}_l = 0 \quad \text{for } j \neq l \qquad (A.11)$$

i.e., if U and V are the matrices that diagonalize HH^t and GG^t respectively. These latter matrices are related to the row and column correlations in the block X:

$$E\,\{(x - \bar{x})\,(x - \bar{x})^t\} = E\,\{HWG^t\,GW^t\,H^t\}$$

$$= HE\,\{WG^t\,GW\}\,H^t$$

$$= H\,\sigma^2\cdot\text{Trace}\,(G^t\,G)\,I\cdot H^t$$

$$= \sigma^2\cdot n\cdot HH^t$$

and

$$E \left| (x - M)^t \, (x - M) \right| = \sigma^2 \cdot n \cdot GG^t \tag{A.12}$$

Thus

$$HH^t = \frac{1}{\sigma^2 n} \, C^{row}$$

and

$$GG^t = \frac{1}{\sigma^2 n} \, C^{col} \tag{A.13}$$

where C^{row} and C^{col} are the row and column covariance matrices of X, $E(x - \overline{x}) \, (x - \overline{x})^t$ and $E \, (x - \overline{x})^t \, (x - \overline{x})$.

The U and V matrices can therefore be obtained by solving the following eigenvector/eigenvalue problems:

$$\left(\frac{1}{\sigma^2 n} \, C^{row} \right) U = U \, \Lambda^{row}$$

$$\left(\frac{1}{\sigma^2 n} \, C^{col} \right) V = V \, \Lambda^{col}$$

where

$$\Lambda^{row} = \text{Diag} \, (\lambda_1^R, \lambda_2^R, \ldots, \lambda_n^R), \; \lambda_i^R \geq 0$$

$$\Lambda^{col} = \text{Diag} \, (\lambda_1^c, \lambda_2^c, \ldots, \lambda_n^c), \; \lambda_j^c \geq 0$$

Because C^{row} and C^{col} are positive semi-definite matrices, both U and V can be found which are orthogonal.

A.3 Predicted KLT Coefficient Statistics

In order to code the z_{ij}, the corresponding mean and variance are required. Using the U and V calculated above, the mean is given by (A.9), and the variance by the specialization of (A.10) to the case where k = i and l = j:

$$E\,(z_{ij} - \bar{z}_{ij})^2 = \sigma^2\, [\underline{u}_i^t\, HH^t\, \underline{u}_i]\, [\underline{v}_j^t\, GG^t\, \underline{v}_j]$$

$$= \sigma^2\, [\underline{u}_i^t\, (\frac{1}{\sigma^2 n} C_{row})\, \underline{u}_i]\, [\underline{v}_j^t\, (\frac{1}{\sigma^2 n} C_{col})\, \underline{v}_j]$$

$$= \sigma^2\, \lambda_i^r\, \lambda_j^c \qquad \text{(A.13)}$$

For the greatest energy compaction into the z_{ij} with the smallest indices, we impose an ordering on the columns of U and V such that

$$\lambda_1^r \geq \lambda_2^r \geq \ldots \geq \lambda_n^r$$

<u>and</u>

$$\lambda_1^c \geq \lambda_2^c \geq \ldots \geq \lambda_n^c \qquad \text{(A.16)}$$

The mean and variance of the z_{ij} can be compactly summarized by a matrix form of equations (A.9) and (A.15):

$$\bar{Z} = [\bar{z}_{ij}] = [E\, z_{ij}] = U^t\, \bar{X} V \qquad \text{(A.17)}$$

and

$$S = [\sigma_{ij}^2] = [E(z_{ij} - \bar{z}_{ij})^2] = \sigma^2\, \underline{\lambda}^r\, \underline{\lambda}^{c^t} \qquad \text{(A.18)}$$

where

$$\underline{\lambda}^r = \begin{bmatrix} \lambda_1^r \\ \cdot \\ \cdot \\ \cdot \\ \lambda_n^r \end{bmatrix}, \quad \underline{\lambda}^c = \begin{bmatrix} \lambda_1^c \\ \cdot \\ \cdot \\ \cdot \\ \lambda_n^c \end{bmatrix}$$

APPENDIX B

HOMOGENIZING IN TRANSFORM COEFFICIENT SPACE

In order to smooth out structural artifacts induced in coefficient sample statistics due to too small a sample space, the sample space is artificially expanded by the addition of new members synthesized from original members.

In particular, statistics of the coefficient arrays

$$Z = U^t XV$$

are required, where X is an image block and U and V are specified unitary matrices. The sample space of Z's is

$$\mathcal{Z} = \{Z: Z = U^t XV, X \epsilon \mathcal{X}\}$$

where $\mathcal{X}$ is the sample space of image blocks X, consisting of all m x n blocks X obtained by partitioning the designated images (often $m = n$).

The sample space $\mathcal{Z}$ is expanded to $\mathcal{Z}'$ by expanding $\mathcal{X}$ to $\mathcal{X}'$. This latter expansion is obtained by including the following blocks in $\mathcal{X}'$:

For all X in $\mathcal{X}$:

- Original: $X = [x_{ij}]$
- Horizontal Flip: $X^H = [x_{i,n+1-j}]$
- Vertical Flip: $X^V = [x_{m+1-i,j}]$
- Double Flip: $X^{HV} = [x_{m+1-i,n+1-j}]$

For square case (m = n), also include:

- Transpose of X: $X^t = [x_{ji}]$
- Horizontal Flip of X^t: $X^{tH} = [x_{j,m+1-i}]$
- Vertical Flip of x^t: $x^{tV} = [x_{n+1-j,i}]$
- Double Flip of x^t: $X^{tHV} = [x_{n+1-j},\ x_{m+1-i}]$

B.1 Flips

Now, rather than expand $\mathcal{X}$ to $\mathcal{X}'$ and apply U^t and V to the resulting large set $\mathcal{X}'$, a more economical approach is to determine the elements of $\mathcal{Z}'$ from $\mathcal{Z}$ directly.

For this purpose, note that the flip operation is characterized by the operator

$$F = \begin{bmatrix} 0 & & 1 \\ & \cdot^{\cdot^{\cdot}} & \\ 1 & & 0 \end{bmatrix},$$

i.e., the 90° rotation of the identity matrix. Let F^n and F^m represent the n x n and m x m versions respectively. Then the blocks X^H, X^V and X^{HV} are related to X by:

$$X^H = XF^n$$

$$X^V = F^m X$$

$$X^{HV} = F^m XF^n$$

Also note that $F^t = F = F^{-1}$.

B.2 Assumption

We will assume that U and V have special symmetry properties:

$$F^m U = UJ^m, \quad F^n V = VJ^n$$

where J^n, J^m are square matrices of the form

$$J = \begin{bmatrix} 1 & & & & & & 0 \\ & -1 & & & & & \\ & & 1 & & & & \\ & & & -1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ 0 & & & & & & -1 \end{bmatrix}.$$

This property means that the 1st, 3rd, etc. columns of U are symmetric about their midpoint, and that the 2nd, 4th, etc. are anti-symmetric. The following transforms have this property:

- Cosine
- Sine
- Hadamard
- Slant
- Karhunen-Loeve when U and V are based on covariance matrices symmetric about the ortho-diagonal.

B.3 Z' from Z Directly

Based on all of this, the coefficient arrays Z^H, Z^V and Z^{HV} obtained from X^H, X^V and X^{HV} can be predicted from Z alone:

$$\begin{aligned} Z^H &= U^t X^H V = U^t (XF^n) V = U^t X (F^n V) = U^t \\ &= U^t X (VJ^n) = U^t XVJ^n \\ &= ZJ^n \end{aligned}$$

$$Z^V = U^t X^V V = U^t (F^m X) V = (U^t F^m) XV$$

$$= (U^t F^{mt}) XV = (F^m U)^t XV$$

$$= (UJ^m)^t XV = (J^{nt} U^t) XV$$

$$= J^m Z$$

$$Z^{HV} = J^m Z J^n$$

In the square case, $m = n$, the coefficient array Z^{Trans} resulting from $U^t X^t V$ is not necessarily the transpose of $Z = U^t XV$. In cases where it's not, the following coefficient arrays are added to $\mathcal{Z}'$:

- Z^{Trans}
- $Z^{Trans^H} = Z^{Trans} J$
- $Z^{Trans^V} = JZ^{Trans}$
- $Z^{Trans^{VH}} = JZ^{Trans} J$

In those square cases where $U = V$, the situation is simpler since $Z^{Trans} = Z^t$ and the following arrays -- all obtainable from Z directly -- are added to $\mathcal{Z}'$:

- Z^t
- $Z^t J$
- JZ^t
- $JZ^t J$

B.4 Statistics

Mean: The mean array $\overline{Z}$ obtained by averaging over all elements of $\mathcal{Z}'$ can be obtained by averaging the following over all members Z of $\mathcal{Z}$ (and for the second case Z^{Trans} of $\mathcal{Z}^{Trans}$):

- $m \neq n$:

$$\frac{1}{4}[Z + Z^{V} + Z^{H} + Z^{HV}]$$

$$= \frac{1}{4}[I + J]\ Z[I + J]$$

$$= \begin{pmatrix} {}^{1}{}_{0} & & & 0 \\ & {}^{1}{}_{0} & & \\ & & \ddots & \\ 0 & & & {}^{1}{}_{0} \end{pmatrix} Z \begin{pmatrix} {}^{1}{}_{0} & & & 0 \\ & {}^{1}{}_{0} & & \\ & & \ddots & \\ 0 & & & {}^{1}{}_{0} \end{pmatrix}$$

- $m = n;\ U \neq V$: $\frac{1}{2} \begin{pmatrix} {}^{1}{}_{0} & & 0 \\ & \ddots & \\ 0 & & {}^{1}{}_{0} \end{pmatrix} (Z + Z^{Trans}) \begin{pmatrix} {}^{1}{}_{0} & & 0 \\ & \ddots & \\ 0 & & {}^{1}{}_{0} \end{pmatrix}$

- $m = n;\ \ = V$: $\frac{1}{2} \begin{pmatrix} {}^{1}{}_{0} & & 0 \\ & \ddots & \\ 0 & & {}^{1}{}_{0} \end{pmatrix} (Z + Z^{t}) \begin{pmatrix} {}^{1}{}_{0} & & 0 \\ & \ddots & \\ 0 & & {}^{1}{}_{0} \end{pmatrix}$

Mean Square Value: The mean square value array of the set $\mathcal{Z}'$ can be directly obtained by element-wise mean square averaging of the following over all members Z of $\mathcal{Z}$ (and Z^{Trans} of $\mathcal{Z}^{Trans}$ for the 2nd case).

- $m \neq n$: Z
- $m = n,\ U \neq V$: $Z,\ Z^{Trans}$
- $m = n,\ U = V$: $Z,\ Z^{t}$

APPENDIX C

OPTIMAL CODER ASSIGNMENTS FOR KLT COEFFICIENTS

We pose the coder assignment problem as one of minimizing the total coding error Er_{TOT} subject to not exceeding a maximum bit allocation B_{MAX}. To be specific, suppose there are L blocks in an image. Then the total mean square error is the sum of each individual block's error:

$$Er_{TOT} = \sum_{l} E(l) \tag{C.1}$$

and the toal bit allocation B_{TOT} is equal to the sum of the individual blocks' bit allocations B(l):

$$B_{TOT} = \sum_{l} B(l) \tag{C.2}$$

The optimization problem can be posed as:

$$\begin{aligned} &\text{minimize } Er_{TOT} \\ &\text{subject to } B_{TOT} \le B_{MAX} \end{aligned} \tag{C.3}$$

This is most easily approached as a Lagrange multiplier problem in which the functional J is formed:

$$\begin{aligned} J &= Er_{TOT} + \lambda[B_{TOT} - B_{MAX}] \\ &= \sum_{l} Er(l) + \lambda[\sum_{l} B(l) - B_{MAX}] \end{aligned} \tag{C.4}$$

The optimum values of B(l) are found by taking partial derivatives of J and setting them to zero:

$$\frac{\partial J}{\partial B(l)} = \left(\frac{\partial Er(l)}{\partial B(l)}\right) + \lambda = 0 \text{ , } l = 1,\cdots,L \tag{C.5}$$

VERAC Incorporated

This yields

$$\frac{\partial Er(l)}{\partial B(l)} = -\lambda, \quad l = 1,\cdots L, \tag{C.6}$$

where the multiplier λ is given by:

$$\lambda = -\frac{\partial Er_{TOT}}{\partial B_{TOT}} , \tag{C.7}$$

which is the (negative) slope of the overall coding error/coding rate curve.

Now, since λ is global and does not depend upon the block index l, we see that if λ is known, we have L independent problems:

$$\frac{2\ Er(l)}{2\ B(l)} = -\lambda \tag{C.8}$$

which means that each block's bit allocation can be separately determined. The key point here is that λ provides global fidelity control: *specifying λ determines where on the coding error/coding rate curve we will operate.* Armed with that information, each block's allocation follows by solving (C.8) for the appropriate l.

The problem of specifying the correct λ to insure constraint satisfaction ($B_{TOT} \leq B_{MAX}$) is called rate equalization. It is treated in Section 6. For present purposes we consider λ given.

C.1 Single Block Problem

To solve (C.8), it is necessary to expand both Er(l) and B(l). Since each block is separately solved, we will drop the "l" argument for notational simplicity. Adopting the energy error measure, we have

$$Er = \sum_{i,j=1}^{n} E(x_{ij} - \hat{x}_{ij})^2 \tag{C.9}$$

where $\hat{x}_{ij}$ is the reconstructed version of pixel x_{ij}. Because the KLT is unitary, this last expression can be computed equally well in the transform domain, yielding

$$Er = \sum_{ij=1}^{n} E(z_{ij} - \hat{z}_{ij})^2 \tag{C.10}$$

where $\hat{z}_{ij}$ is the reconstructed version of coefficient z_{ij}.

Also, the total bit allocation for the block can be expanded in terms of the bit allocations for each coefficient, yielding

$$B = \sum_{ij=1}^{n} B_{ij} , \tag{C.11}$$

where B_{ij} is the bit allocation for z_{ij}. What we ultimately seek are the B_{ij}'s.

Now, we notice that (C.8) is one of the necessary conditions required to solve the single-block Lagrange multiplier problem,

$$J' = \sum_{ij} E(z_{ij} - \hat{z}_{ij})^2 + \lambda[\sum_{ij} B_{ij} - B] \tag{C.12}$$

which arises from wanting a solution to the following constrained minimization problem:

$$\begin{aligned} &\text{minimize } Er = \sum_{ij} E(z_{ij} - \hat{z}_{ij})^2 \\ &\text{subject to } B = \sum_{ij} B_{ij}. \end{aligned} \tag{C.13}$$

The remaining necessary conditions for the single-block Lagrange multiplier problem are obtained by setting to zero the partials of J' w.r.t. the B_{ij}:

$$\frac{\partial J'}{\partial B_{ij}} = \frac{\partial E(z_{ij} - \hat{z}_{ij})^2}{\partial B_{ij}} + \lambda = 0 \tag{C.14}$$

or:

$$\frac{\partial E(z_{ij} - \hat{z}_{ij})^2}{\partial B_{ij}} = -\lambda \tag{C.15}$$

Thus, again the problem reduces in scope: Now we need only solve bit allocations for a single coefficient at a time.

C.2 Single Coefficient Problem

For this we need a relationship connecting $E(z_{ij} - \hat{z}_{ij})^2$ and B_{ij}. We make the following assumptions:

- $E\ z_{ij} = \bar{z}_{ij} = E\ \hat{z}_{ij}$, i.e. the coder is unbiased.
- The random variables z_{ij} all share the same form of probability density function (pdf) with each parameterized by its mean $\bar{z}_{ij}$ and variance σ_{ij}^2.
- Quantization is performed by first subtracting $\bar{z}_{ij}$ from z_{ij}, then dividing by σ_{ij}, then finally passing the result through a B_{ij} - bit ($2^{B_{ij}}$ - level) unbiased quantizer. Reconstruction re-introduces the σ_{ij} factor and biases the result by $\bar{z}_{ij}$.

These assumptions are all in force for the coders used in this study. Under these conditions

$$E(z_{ij} - \hat{z}_{ij})^2 = \sigma_{ij}^2\ f(B_{ij}), \tag{C.16}$$

where $f(\cdot)$ is a monotonically decreasing positive function depending upon the assumed pdf and the type of quantizer.

The analytical form of f(B) is generally unwieldly, and common practice is to employ a simpler curve fit. Specifically, a fit of the following form is used here:

$$f(B) - b\,2^{-B/a} \tag{C.17}$$

where b and a are parameters tailored to various types of pdf's and quantizers. For example, for the case of Gaussian pdf's and Max quantizers -- which we use for KLT coefficient coding -- good upper-bound values are b = 2.2 and a = 0.5. Good lower-bound values are b = 1 and a = 0.5.

Given the fit (C.17) and the expression (C.16), the necessary condition (C.15) reduces to:

$$\sigma_{ij}^2\,(\ln 2)\,(b/a)\,2^{-B_{ij}/a} = \lambda\,, \tag{C.18}$$

which yields:

$$B_{ij} = a \log_2 \frac{\sigma_{ij}^2}{\left(\lambda \cdot \frac{a}{b \ln 2}\right)} \tag{C.19}$$

For our cases of interest, a = 0.5. We also denote

$$D = \left(\frac{\lambda\, a}{b \ln 2}\right)^{1/2} \tag{C.20}$$

to obtain

$$B_{ij} = \log_2 \frac{\sigma_{ij}}{D} \quad \text{(C.21,}$$

This last expression tells how to determine the bit allocation B_{ij} from the coefficient variances σ_{ij} and the global distortion control parameter D.

APPENDIX D

HOMOGENIZING SVD'S OF PRETRANSFORMED BLOCKS

We begin with $U^t XV = S$, where X is derived by applying either the cosine or Hadamard transform to the rows and columns of a pixel block x. We wish to find the singular values and left and right singular vectors for homogenized versions of X.

D.1 Flips

Corresponding to the set x, x^H, x^V, x^{HV} of flipped pixel blocks are the following transform blocks:

$$X,\ XJ^n,\ J^m X,\ J^m XJ^n$$

where J^n, J^m are the n x n and m x m versions of the matrix:

$$J = \begin{bmatrix} 1 & & & & & & 0 \\ & -1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ 0 & & & & & & -1 \end{bmatrix}$$

Note that this matrix has the property that:

$$J^{-1} = J^t = J.$$

Therefore, if $\quad U^t XV = S,$

$$U^t (XJ^n)(J^n V) = S$$

$$(J^m U)^t (J^m X) V = S$$

$$(J^m U)^t (J^m XJ^n)(J^n V) = S$$

VERAC Incorporated

Therefore, to homogenize the SVD wrt flips, average over the following:

- singular values: S only
- left singular vectors: U, $J^m U$
- right singular vectors: V, $J^n V$

D.2 Transpose (Rotation and Flip)

Since the transform of x is X, the following relationship:

$$U^t XV = S \Rightarrow V^t X^t U = S$$

tells us to homogenize wrt transposes by

- singular values: S only
- left singular vectors: U and V
- right singular vectors: U and V

VERAC Incorporated

APPENDIX E

SVD ORTHOGONAL EXPANSION COEFFICIENT STATISTICS

This appendix constitutes an analysis to determine the first two moments of the coefficients $\{\alpha_{ij}\}$ from the singular vector statistics $\{\overline{u_{ij}}\}$ and $\{\overline{u^2_{ij}}\}$.

E.1 Problem

At the jth step, the orthogonal singular vectors $\underline{u}_1,\cdots,\underline{u}_{j-1}$ have been quantized, transmitted, and decoded as $\hat{\underline{u}}_1,\cdots,\hat{\underline{u}}_{j-1}$. These vectors are then used to find an orthogonal basis for the jth step (assumes the $\hat{\underline{u}}_i$'s are orthogonal).

$$B_j = \left[\hat{\underline{u}}_1^N, \hat{\underline{u}}_2^N, \ldots, \hat{\underline{u}}_{j-1}^N, \underline{b}_j^N, \underline{b}_{j+1}^N, \ldots, \underline{b}_m^N\right]$$

where

$$\hat{u}_i^N = \frac{1}{|\hat{u}_i|} \hat{u}_i$$

$$\underline{b}_i = e_i - \sum_{l=1}^{j-1} (\underline{e}_i^t \hat{\underline{u}}_l^N) \hat{\underline{u}}_l^N - \sum_{l=j}^{i-1} (\underline{e}_i^t \underline{b}_l^N) \underline{b}_l^N$$

$$\underline{b}_i^N = \frac{1}{|\underline{b}_i|} \underline{b}_i .$$

Then express u_i as

$$u_i = \sum_{l=1}^{j-1} \alpha_{li} \hat{u}_i^N + \sum_{l=j}^{m} \alpha_{li} \underline{b}_l^N$$

We wish to find the statistics $\overline{\alpha_{li}}$ and $\overline{\alpha^2_{li}}$ of these coefficients in order to encode them.

E.2 Case 1

α_{ij} for $\boxed{i < j}$:

$$\alpha_{ij} = \underline{u}_j^t\, b_{ij}^N = \underline{u}_j^t\, \hat{\underline{u}}_i^N$$

$$= \underline{u}_j^t\, (\underline{u}_i - \tilde{\underline{u}}_i^N) \qquad\qquad \tilde{u} = u - \hat{u}$$

$$= -\underline{u}_j^t\, \tilde{\underline{u}}_i^N$$

$$E\alpha_{ij} = -E\,(\underline{u}_j^t)\; E\,(\tilde{\underline{u}}_i^N) = 0$$

$$E\alpha_{ij}^2 = E\,(\underline{u}_j^t\, \tilde{\underline{u}}_i^N)^2$$

$$= \mathrm{tr}\, E \left\{ \left(\tilde{\underline{u}}_i^N\, \tilde{\underline{u}}_i^{N\,t} \right) \left(\underline{u}_j\, \underline{u}_j^t \right) \right\}$$

$$= \mathrm{tr}\, E \left(\tilde{\underline{u}}_i^N\, \tilde{\underline{u}}_i^{N\,t} \right) \cdot E\,(\underline{u}_j\, \underline{u}_j^t)$$

$$= \mathrm{tr}\, \mathrm{diag} \left(\overline{\tilde{u}_{ki}^{N^2}} \right) \cdot \mathrm{diag} \left(\overline{u_{kj}^2} \right)$$

$$= \sum_{k=1}^{m} \overline{\tilde{u}_{ki}^{N^2}}\; \overline{u_{kj}^2}$$

$$= d_i \cdot \sum_{k=1}^{m} \overline{u_{kj}^2}$$

(This uses $\tilde{\underline{u}}_i$ in place of $\tilde{\underline{u}}_i^N$, i.e., uses $|\hat{\underline{u}}_i| = 1$ in order to obtain a linear-in-d expression for α_{ij}^2.)

$$= d_i$$

E.3 Case 2

α_{ij} for $\boxed{i \geq j}$

$$\underline{b}_{ij} = \underline{e}_i - \sum_{l=1}^{j-1} \hat{u}_{il}^N + \hat{\underline{u}}_l^N + \sum_{k=j}^{i-1} (\underline{b}_{ij}^N)_i \, \underline{b}_{ij}^N$$

$$\alpha_{ij} = \underline{u}_j^t \, \underline{b}_{ij}^N$$

$$= \frac{1}{|\underline{b}_{ij}|} \underline{u}_j^t \, \underline{b}_{ij}$$

$$= \frac{1}{|\underline{b}_{ij}|} \underline{u}_j^t \left[\underline{e}_i - \sum_{l=1}^{j-1} \hat{u}_{il}^N \, \hat{u}_{jl}^N + \sum_{k=j}^{i-1} (\underline{b}_{ij}^N)_i \, (\underline{b}_{ij}^N)_j \right]$$

$$= \frac{1}{|\underline{b}_{ij}|} \left[u_{ij} - \sum_{l=1}^{j-1} \hat{u}_{il}^N \quad \alpha_{lj} - \sum_{l=j}^{i-1} (\underline{b}_{lj}^N)_i \, \alpha_{lj} \right]$$

For simplification of notation, we will henceforth denote $\underline{b}_{ij}$ by $\underline{b}_i$, notationally supressing the dependence of $\underline{b}$'s on the stage j.

$$\underline{b}_i = \underline{e}_i - \sum_{l=1}^{j-1} \hat{u}_{il_1}^N \, \hat{\underline{u}}_{l_1}^N - \sum_{l_2=j}^{i-1} b_{il_2}^N \, \underline{b}_{l_2}^N \quad , \quad \underline{b}_i^N = \frac{1}{|\underline{b}_i|} \underline{b}_i$$

$$\therefore |\underline{b}_i|^2 = 1 - \sum_{l_1=1}^{j-1} \hat{u}_{il_1}^{N^2} - \sum_{l_2=j}^{i-1} b_{il_2}^{N^2}$$

$$\alpha_i = \frac{1}{|\underline{b}_i|} \left(u_{ij} - \sum_{l_1=1}^{j-1} \hat{u}_{il_1}^N \, \alpha_{l_1} - \sum_{l_2=1}^{j-1} b_{il_2}^N \quad \alpha_{l_2} \right)$$

$$E\alpha_i = 0 - 0 - 0 = 0$$

$$\alpha_i^2 = \frac{1}{|\underline{b}_i|^2} \left(u_{ij} - \sum_{l_1=1}^{j-1} \hat{u}_{il_1}^N \, \alpha_{l_1} - \sum_{l_2=1}^{j-1} b_{il_2}^N \quad \alpha_{l_2} \right)^2$$

$$\approx \left[1 + \sum_{l_1=1}^{j-1} \hat{u}^{N^2}_{il_1} + \sum_{l_2=j}^{i-1} b^{N^2}_{il_2}\right] \left[u^2_{ij} - 2u_{ij} \sum_{l_1=1}^{j-1} \hat{u}^N_{il_1} \alpha_{l_1}\right.$$

$$- 2u_{ij} \sum_{l_2=j}^{i-1} b^N_{il_2} \alpha_{l_2}$$

$$+ \sum_{l_1=1}^{j-1} \sum_{l_2=1}^{j-1} \hat{u}^N_{il_1} \hat{u}^N_{il_2} \alpha_{l_1} \alpha_{l_2}$$

$$+ \sum_{l_1=j}^{i-1} \sum_{l_2=j}^{i-1} b^N_{il_1} b^N_{il_2} \alpha_{l_1} \alpha_{l_2}$$

$$\left. + \sum_{l_1=1}^{j-1} \sum_{l_2=j}^{i-1} \hat{u}^N_{il_1} b^N_{il_2} \alpha_{l_1} \alpha_{l_2}\right]$$

$$E\alpha^2_i = \overline{u^2_{ij}} - 0 - 0 + \sum_{l=1}^{j-1} \overline{\hat{u}^{N^2}_{il}} \, \overline{\alpha^2_l} + \sum_{l=j}^{i-1} \overline{b^{N^2}_{il}} \, \overline{\alpha^2_l} + 0$$

$$+ \overline{u^2_{ij}} \sum_{l=1}^{j-1} \overline{\hat{u}^{N^2}_{il}} + \left(\sum_{l=1}^{j-1} \overline{\hat{u}^{N^2}_{il}}\right) \sum_{m=1}^{j-1} \overline{\hat{u}^{N^2}_{im}} \, \overline{\alpha^2_m}$$

$$+ 2 \sum_{l=1}^{j-1} \left(\overline{\hat{u}^N_{il}}\right)^2 \overline{\alpha^2_l} + \left(\sum_{l=1}^{j-1} \overline{\hat{u}^{N^2}_{il}}\right) \sum_{m=j}^{i-1} \overline{b^{N^2}_{im}} \, \overline{\alpha^2_m}$$

$$+ \overline{u^2_{ij}} \sum_{l=j}^{i-1} \overline{b^{N^2}_{il_2}} + \left(\sum_{l=j}^{i-1} \overline{b^{N^2}_{il}}\right) \sum_{m=1}^{j-1} \overline{\hat{u}^{N^2}_{im}} \, \overline{\alpha^2_m}$$

$$+\left(\sum_{l=j}^{i-1} \overline{b_{il}^{N^2}}\right) \sum_{l=j}^{i-1} \overline{b_{il}^{N^2}}\ \overline{\alpha_m^2} + 2 \sum_{l=j}^{i-1} \left(\overline{b_{il}^{N^2}}\right)^2 \overline{\alpha_l^2}$$

Neglecting all higher order terms, this can be simplified to:

$$E\alpha_i^2 = \overline{u_{ij}^2} + \overline{u_{ij}^2} \sum_{l=1}^{j-1} \overline{u_{il}^2}$$

$$= \overline{u_{ij}^2} \left[1 + \sum_{l=1}^{j-1} \overline{u_{il}^2}\right]$$

E.4 Summary

$$\overline{\alpha_{ij}} = 0$$

$$\overline{\alpha_{ij}^2} = \begin{cases} d_{u_i} = \sum_{l=1}^{n} \overline{u_{lj}^2} & , \ i < j \\ \overline{u_{ij}^2} \left[1 + \sum_{l=1}^{j-1} \overline{u_{il}^2}\right] & , \ i \geq j \end{cases}$$

VERAC incorporated

APPENDIX F

SVD COEFFICIENT CODING: BIT ALLOCATION

SVD block reconstruction error is given by:

$$
\begin{aligned}
\tilde{X} &= X - \hat{X} \\
&= \sum_l X_l - \sum_l \hat{X}_l \qquad (X = \textstyle\sum X_l \text{ is inverse SVD, } X_l = s_l\, B_l, \\
&\qquad\qquad\qquad\qquad\qquad \text{where } B_l = \underline{u}_l\, \underline{v}_l^t) \\
&= \sum_l (X_l - \hat{X}_l) \\
&= \sum_l (s_l\, B_l - \hat{s}_l\, \hat{B}_l) \\
&= \sum_l [s_l\, B_l - (s_l - \tilde{s}_l)(B_l - \tilde{B}_l)] \\
&= \sum_l [s_l\, B_l - s_l\, B_l + s_l\, \tilde{B}_l + \tilde{s}_l\, B_l - \tilde{s}_l\, \tilde{B}_l] \\
&= \sum_l [s_l\, \tilde{B}_l + \tilde{s}_l\, B_l - \tilde{s}_l\, \tilde{B}_l] \\
&\triangleq \sum_l \tilde{X}_l
\end{aligned}
$$

where

s_l = lth singular vector

B_l = $\underline{u}_l\, \underline{v}_l^t$ ... lth basis block

$\tilde{s}_l$ = quantization error in s_l

$\tilde{B}_l$ = quantization error in B_l

VERAC incorporated

The squared reconstruction error is then

$$\tilde{X}\tilde{X}^t = \left[\sum_l \tilde{X}_l\right] \left[\sum_k \tilde{X}_k\right]^t$$

$$= \left[\sum_l s_l \tilde{B}_l + \tilde{s}_l B_l - \tilde{s}_l \tilde{B}_l\right] \left[\sum_k s_k \tilde{B}_k + \tilde{s}_k B_k - \tilde{s}_k \tilde{B}_k\right]^t$$

$$\begin{aligned} = \sum_l \sum_k \Big[& s_l s_k \tilde{B}_l \tilde{B}_k^t + s_l \tilde{s}_k \tilde{B}_l B_k^t - s_l \tilde{s}_k \tilde{B}_l \tilde{B}_k^t \\ & + \tilde{s}_l s_k B_l \tilde{B}_k^t + \tilde{s}_l \tilde{s}_k B_l B_k^t - \tilde{s}_l \tilde{s}_k B_l \tilde{B}_k^t \\ & - \tilde{s}_l s_k \tilde{B}_l \tilde{B}_k^t - \tilde{s}_l \tilde{s}_k \tilde{B}_l B_k^t + \tilde{s}_l \tilde{s}_k \tilde{B}_l \tilde{B}_k^t \Big] \end{aligned} \tag{F.2}$$

Now assume:

- s_i, s_j uncorrelated for all $i \neq j$
- $\tilde{s}_i$, $\tilde{s}_j$ uncorrelated for all $i \neq j$
- $\tilde{s}_i$, s_j uncorrelated for all $i \neq j$
- $\tilde{B}_i$, B_j uncorrelated for all $i \neq j$

(F.3)

Then $\tilde{X}_i$ uncorrelated with $\tilde{X}_j$ for $i \neq j$ (assume Gaussian).

VERAC Incorporated

For uncorrelated $\tilde{X}_l$'s:

$$\mathrm{MSE} = \sum_{l} E \operatorname{tr} \tilde{X}_l \tilde{X}_l^t \tag{F.4}$$

$$\therefore \text{ let } e_l = \operatorname{tr} \tilde{X}_l \tilde{X}_l^t \tag{F.5}$$

Then, supressing subscripts:

$$\begin{aligned} e = \operatorname{tr} \{ & s^2 \tilde{B} \tilde{B}^t + s \tilde{s} \tilde{B} B^t - s \tilde{s} \tilde{B} \tilde{B}^t \\ & + \tilde{s} s B \tilde{B}^t + \tilde{s}^2 B B^t - \tilde{s} \tilde{s} B \tilde{B}^t \\ & - \tilde{s} s \tilde{B} \tilde{B}^t - \tilde{s}^2 \tilde{B} B^t + \tilde{s}^2 \tilde{B} \tilde{B}^t \} \end{aligned} \tag{F.6}$$

We next want to take the expected value. We will apply this theorem:

> Theorem X_i's zero mean, jointly Gaussian $\Rightarrow$
>
> $$\begin{aligned} E x_1 x_2 x_3 x_4 &= E x_1 x_2 E x_3 x_4 \\ &+ E x_1 x_3 E x_2 x_4 \\ &+ E x_1 x_4 E x_2 x_3 \end{aligned} \tag{F.7}$$

The form this takes for us is

$$E\, s_1\, s_2\, B_1\, B_2^t = E\, s_1\, s_2\, E\, B_1\, B_2^t + E\, s_1\, B_1\, E\, s_2\, B_2^t$$
$$+ E\, s_2\, B_1\, E\, s_1\, B_2^t \qquad \text{(F.8)}$$

Note: We are assuming that s_1, $\tilde{s}_1$, B_1, $\tilde{B}_1$ are jointly Gaussian.

Now, applying this to each term of the previous expression for e generates terms involving:

$$E\, s^2,\ \underline{E\, s\, \tilde{s}},\ E\, \tilde{s}^2$$
$$E\, B\, B^t,\ \underline{E\, B\, \tilde{B}^t},\ E\, \tilde{B}\, \tilde{B}^t$$
$$E\, s\, B,\ \underline{E\, \tilde{s}\, B},\ \underline{E\, s\, \tilde{B}} \text{ and } \underline{E\, \tilde{s}\, \tilde{B}} \qquad \text{(F.9)}$$

We assume those underlined to be zero. This results in:

$$Ee = tr \left\{ \overline{s^2}\ \overline{\tilde{B}\, \tilde{B}^t} + \overline{\tilde{s}^2}\ \overline{B\, B^t} + \overline{\tilde{s}^2}\ \overline{\tilde{B}\, \tilde{B}^t} \right\} \qquad \text{(F.10)}$$

where overbar indicates expected value.

$$\text{Now } \tilde{B} = B - \hat{B} = u\, v^t - \hat{u}\, \hat{v}^t$$
$$= u\, v^t - (u - \tilde{u})\, (v - \tilde{v})^t$$
$$= u\, v^t - [u\, v^t - \tilde{u}\, v^t - u\, \tilde{v}^t + \tilde{u}\, \tilde{v}]$$
$$= \tilde{u}\, v^t + u\, \tilde{v}^t - \tilde{u}\, \tilde{v}^t \qquad \text{(F.11)}$$

$$\therefore \tilde{B}\,\tilde{B}^t = [\tilde{u}\, v^t\, v\, \tilde{u}^t + \tilde{u}\, v^t\, \tilde{v}\, u^t - \tilde{u}\, v^t\, \tilde{v}\, \tilde{u}^t$$

$$+ u\, \tilde{v}^t\, v\, \tilde{u}^t + u\, \tilde{v}^t\, \tilde{v}\, u^t - u\, \tilde{v}^t\, \tilde{v}\, \tilde{u}^t$$

$$- \tilde{u}\, \tilde{v}^t\, v\, \tilde{u}^t - \tilde{u}\, \tilde{v}^t\, \tilde{v}\, u^t + \tilde{u}\, \tilde{v}^t\, \tilde{v}\, \tilde{u}^t] \qquad \text{(F.12)}$$

Now we want to take tr $\{E\,\{\cdot\}\,\}$. In doing so, we will assume:

- $\tilde{u}$, $\tilde{v}$ uncorrelated
- u, v uncorrelated
- $\tilde{u}$, u uncorrelated
- $\tilde{v}$, v uncorrelated (F.13)

Thus:

$$E\,\{\text{tr}\, \tilde{B}\,\tilde{B}^t\} = E\, \tilde{u}^t \tilde{u}\, E\, v^t\, v + E\, u^t\, \tilde{u}\, E\, v^t\, \tilde{v} - E\, \tilde{u}^t\, \tilde{u}\, E\, v^t\, \tilde{v}$$

$$+ E\, \tilde{u}^t\, u\, E\, \tilde{v}^t\, v + E\, u^t u\, E\, \tilde{v}^t\, \tilde{v} - E\, \tilde{u}^t\, u\, E\, \tilde{v}^t\, \tilde{v}$$

$$- E\, \tilde{u}^t\, \tilde{u}\, E\, \tilde{v}^t\, v - E\, u^t\, \tilde{u}\, E\, \tilde{v}^t\, v + E\, \tilde{u}^t\, \tilde{u}\, E\, \tilde{v}^t\, \tilde{v}$$

$$= E|\tilde{u}|^2\, E|v|^2 + E|u|^2\, E|\tilde{v}|^2 + E|\tilde{u}|^2\, E|\tilde{v}|^2 \qquad \text{(F.14)}$$

Also, $B\, B^t = (u\, v^t)\,(u\, v^t)^t = u\, v^t\, v\, u^t$

$$\text{tr}\, B\, B^t = u^t\, u\, v^t\, v$$

$$E\,\{\text{tr}\, B\, B^t\} = E|u|^2\, E|v|^2 \qquad \text{(F.15)}$$

Now:

$$E\,\tilde{s}^2 = (\sigma_s^2)\, f_s\,(n_s) \quad , \quad \sigma_s^2 = E\,s^2 - (Es)^2$$

$$E\,\tilde{u}_i^2 = \sigma_{u_i}^2 \quad f_u(n_{u_i}) \quad , \quad \sigma_{u_i}^2 = E\,u_i^2 - (Eu_i)^2$$

$$E\,\tilde{v}_i^2 = \sigma_{v_i}^2 \quad f_v\,(n_{v_i}) \quad , \quad \sigma_{v_i}^2 = E\,v_i^2 - (Ev_i)^2 \qquad (F.16)$$

Finally:

$$\begin{aligned}
Ee &= \overline{s^2}\ \mathrm{tr}\ \overline{\tilde{B}\tilde{B}^t} + \overline{\tilde{s}^2}\ \mathrm{tr}\ \overline{BB^t} + \overline{\tilde{s}^2}\ \mathrm{tr}\ \overline{\tilde{B}\tilde{B}^t} \\
&= (\sigma_s^2 + \bar{s}^2)\ \mathrm{tr}\ \overline{\tilde{B}\tilde{B}^t} + \left(\sigma_s^2\ f_s\ (n_s)\right)\ \mathrm{tr}\ \overline{BB^t} + \left(\sigma_s^2\ f_s\ (n_s)\right)\ \mathrm{tr}\ \overline{\tilde{B}\tilde{B}^t} \\
&= \left(\bar{s}^2 + \sigma_s^2\ \left(1 + f_s\ (n_s)\right)\right)\ \mathrm{tr}\ \overline{\tilde{B}\tilde{B}^t} + \sigma_s^2\ f_s\ (n_s)\ \mathrm{tr}\ \overline{BB^t} \\
&= \left[\bar{s}^2 + \sigma_s^2\ \left(1 + f_s\ (n_s)\right)\right] \cdot \left[\sum_i \sigma_{u_i}^2\ f_u\ (n_{u_i})\ \sum_j (\bar{v}_i^2 + \sigma_{v_i}^2)\right. \\
&\qquad + \sum_i \left(\bar{u}_i^2 + \sigma_{u_i}^2\right)\ \sum_j \sigma_{v_j}^2\ f_v\ (n_{u_j}) \\
&\qquad \left. + \sum_i \sigma_{u_i}^2\ f_u\ (n_{u_i})\ \sum_j \sigma_{v_j}^2\ f_v\ (n_{v_j})\right] \\
&\qquad + \sigma_s^2\ f_s\ (n_s)\ \sum_i (\bar{u}_i^2 + \sigma_{u_i}^2)\ \sum_j\ (\bar{v}_j + \sigma_{v_j}^2)
\end{aligned}$$

We will be interested in partial derivatives of this expression w.r.t. n_s, n_{u_i} and n_{v_j}.

$$\frac{\partial Ee}{\partial n_s} = \sigma_s^2 \left[\sum_i (\overline{u}_i^2 + \sigma_{u_i}^2) + \sum_i \sigma_{u_i}^2 f_u (n_{u_i})\right]$$

$$\cdot \left[\sum_j (\overline{v}_j^2 + \sigma_{v_j}^2) + \sum_j \sigma_{v_j}^2 f_v (n_{v_j})\right] \cdot \frac{\partial f_s(n_s)}{\partial n_s}$$

$$= \sigma_s^2 \left[\sum_i \left(\overline{u}_i^2 + \sigma_{u_i}^2 \left(1 + f_u (n_{u_i})\right)\right)\right]$$

$$\cdot \left[\sum_j \left(\overline{v}_j^2 + \sigma_{v_j}^2 \left(1 + f_v (n_{v_j})\right)\right)\right] \cdot \frac{\partial f_s(n_s)}{\partial n_s} \qquad \text{(F.18a)}$$

$$\frac{\partial Ee}{\partial n_{u_i}} = \left[\overline{s}^2 + \sigma_s^2 \left(1 + f_s (n_s)\right)\right] \left[\sum_j \left(\overline{v}_j^2 + \sigma_{v_j}^2 \left(1 + f_v (n_{v_j})\right)\right)\right] \sigma_{u_i}^2$$

$$\cdot \frac{\partial f_u(n_{u_i})}{\partial n_{u_i}} \qquad \text{(F.18b)}$$

$$\frac{\partial Ee}{\partial n_{v_j}} = \left[\overline{s}^2 + \sigma_s^2 \left(1 + f_s (n_s)\right)\right] \left[\sum_i \left(\overline{u}_i^2 + \sigma_{u_i}^2 \left(1 + f_u (n_{u_i})\right)\right)\right] \sigma_{u_j}^2$$

$$\cdot \frac{\partial f_v(n_{v_j})}{\partial n_{v_j}} \qquad \text{(F.18c)}$$

Now, using the fits:

$$f_s(n) = b_s\, 2^{-n/a_s}$$

$$f_u(n) = f_v(n) = b\, 2^{-n/a} \qquad (F.19)$$

we shall find explicit expressions for the remaining partials:

$$\frac{\partial f_s(n_s)}{\partial n_s} = -\left(\frac{b_s}{a_s}\right) 2^{-n_s/a_s} \ln 2$$

$$\frac{\partial f_u(n_{u_i})}{\partial n_u} = -\left(\frac{b}{a}\right) 2^{-n_u/a} \ln 2 \qquad (F.20)$$

$$\frac{\partial f_v(n_{v_j})}{\partial n_v} = -\left(\frac{b}{a}\right) 2^{-n_v/a} \ln 2$$

Now, to find expressions for bit allocations, we will solve the following problem:

minimize MSE

$$\text{subject to} \quad \sum_l \left\{ n_{s_l} + \sum_i n_{u_{il}} + \sum_j n_{v_{jl}} \right\} = N_T \qquad (F.21)$$

We will approach the problem via Lagrange multipliers. That is, we form the functional

$$J = MSE + \lambda\left[\sum_{l} n_{s_l} + \sum_{l}\sum_{i} n_{u_{il}} + \sum_{l}\sum_{j} n_{v_{jl}} - N_T\right] \qquad (F.22)$$

and set the derivatives $\frac{\partial J}{\partial n_{s_l}}$, $\frac{\partial J}{\partial n_{u_{il}}}$, $\frac{\partial J}{\partial n_{v_{jl}}}$ equal to 0.

From the previous expressions for Ee (one term of MSE $= \sum_{l} Ee_l$), we have:

$$\sigma_s^2\left[\sum_i \bar{u}_i^2 + \sigma_{u_i}^2\left(1 + b2^{-n_{u_i}/a}\right)\right]\cdot\left[\sum_j \bar{v}_j^2 + \sigma_{v_j}^2\left(1 + b2^{-n_{v_j}/a}\right)\right]$$

$$\cdot(\ln 2)\left(\frac{b_s}{a_s}\right)2^{-n_s/a_s} = \lambda \qquad (F.23a)$$

$$\left[\bar{s}^2 + \sigma_s^2\left(1 + b_s 2^{-n_s/a_s}\right)\right]\cdot\sigma_{u_i}^2\cdot\left[\sum_j \bar{v}_j^2 + \sigma_{v_j}^2\left(1 + b2^{-n_{v_j}/a}\right)\right]$$

$$\cdot(\ln 2)\left(\frac{b}{a}\right)2^{-n_{u_i}/a} = \lambda \qquad (F.23b)$$

$$\left[\bar{s}^2 + \sigma_s^2\left(1 + b_s 2^{-n_s/a_s}\right)\right]\cdot\left[\sum_i \bar{u}_i^2 + \sigma_{u_i}^2\left(1 + b2^{-n_{u_i}/a}\right)\right]$$

$$\cdot\sigma_{v_j}^2\;(\ln 2)\left(\frac{b}{a}\right)2^{-n_{v_j}/a} = \lambda \qquad (F.23c)$$

for every term Ee_l in MSE,

where λ is the value of the sensitivity of the total error to the total allocation:

$$\frac{\partial MSE}{\partial N_T} = -\lambda \tag{F.24}$$

To solve (F.23) we begin by noticing that the MSE in the various elements of $\underline{u}$ are equal, and similarly, that the MSE in the various elements of $\underline{v}$ are equal. We express this as:

$$\sigma^2_{u_i}\left(b2^{-n_{u_i}/a}\right) = d_u \qquad \forall \qquad i = 1,\ldots,m$$

$$\sigma^2_{vj}\left(b2^{-n_{v_j}/a}\right) = d_v \qquad \forall \qquad i = 1,\ldots,m. \tag{F.25}$$

The truth of these statements can be established by ratioing F.23b (and F.23c) for different i (and j).

We now simplify (F.23) by writing:

$$\mathscr{S} = \bar{s}^2 + \sigma^2_s\left(1 + b_s 2^{-n_s/a_s}\right)$$

$$= \bar{s}^2 + \sigma^2_s + \sigma^2_s\, b_s\, 2^{-n_s/a_s} \tag{F.26a}$$

$$= \overline{s^2} + d_s\,,$$ where d_s is defined as implied.

$$\mathcal{U} = \sum_i \overline{u_i^2} + \sigma_{u_i}^2 \left(1 + b2^{-n_{u_i}/a}\right)$$

$$= \sum_i \overline{u_i^2} + \sigma_{u_i}^2 + \sigma_{u_i}^2\, b2^{-n_{u_i}/a}$$

$$= \sum_i \left(\overline{u_i^2} + d_u\right) \tag{F.26b}$$

$$= \sum_i \overline{u_i^2} + md_u$$

$$= \overline{|\underline{u}|^2} + md_u$$

and similarly:

$$\mathcal{V} = \overline{|\underline{v}|^2} + nd_v. \tag{F.26c}$$

Now, we introduce first order approximations:

- $d_s \ll \overline{s^2} \quad : \mathcal{S} = \overline{s^2}$ (F.27a)
- $md_u \ll \overline{|\underline{u}|^2} \quad : \mathcal{U} = \overline{|\underline{u}^2|} = 1$ (F.27b)
- $nd_v \ll \overline{|\underline{v}|^2} \quad : \mathcal{V} = \overline{|\underline{v}^2|} = 1$ (f.27c)

Therefore, the equations (F.26) simplify to:

$$d_s = \frac{\lambda a_s}{\ln 2} \tag{F.28a}$$

$$d_u = \frac{\lambda a}{\ln 2} \cdot \frac{1}{s^2} \tag{F.28b}$$

$$d_v = \frac{\lambda a}{\ln 2} \cdot \frac{1}{s^2} \tag{F.28c}$$

In light of this, it is convenient to solve (F.25), and the definition of d_s (in F.26a), in terms of the bit allocations:

$$n_s = a_s \log_2 \frac{\sigma_s^2 \cdot b_s}{d_s} \tag{F.29a}$$

$$n_{u_i} = a \log_2 \frac{\sigma_{u_i}^2 \cdot b}{d_u} \tag{F.29b}$$

$$n_{v_i} = a \log_2 \frac{\sigma_{v_i}^2 \cdot b}{d_v} \tag{F.29c}$$

Finally, combining (F.28) and (F.29):

$$n_s = a_s \log_2 \frac{\sigma_s^2}{D^2}$$

$$n_{u_i} = a \log_2 \frac{\sigma_{u_i}^2 \quad \overline{s^2}}{D^2 \left(\dfrac{ab_s}{a_s b}\right)}$$

$$n_{v_i} = a \log_2 \frac{\sigma_{v_i}^2 \quad \overline{s^2}}{D^2 \left(\dfrac{ab_s}{a_s b}\right)}$$

where

$$D^2 = d_s/b_s = \frac{\lambda \ a_s}{\ln 2 \ b_s} \quad .$$

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

DATE
ILME