1987

Image Reconstruction After Transform Coding Using Relative Entropy and Maximum Entropy

John S. Bodenschatz
University of Central Florida

IMAGE RECONSTRUCTION AFTER TRANSFORM CODING USING RELATIVE ENTROPY AND MAXIMUM ENTROPY

BY

JOHN S. BODENSCHATZ, II
B.S.E.E., University of Virginia, 1986

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program
of the College of Engineering
University of Central Florida
Orlando, Florida

Fall Term
1987

ABSTRACT

Presented are two new methods based on entropy for reconstructing images compressed with the Discrete Cosine transform. One method is based upon a sequential implementation of the Minimum Relative Entropy Principle; the other is based upon the Maximum Entropy Principle. These will be compared with each other and with the conventional method employing the Inverse Discrete Cosine transform.

Chapter 2 describes the traditional use of the Discrete Cosine transform for image compression. Chapter 3 explains the theory and implementation of the entropy-based reconstructions. It introduces a fast algorithm for the Maximum Entropy Principle. Chapter 4 compares the numerical performance of the three reconstruction methods. Chapter 5 shows the theoretical convergence limit of the iterative implementation of the Minimum Relative Entropy Principle to equal the limit of the convergence of the Maximum Relative Entropy method.

Preliminary results of this thesis were presented at Southeastcon '87 in Tampa. Final results will be presented at the Annual Meeting of the American Optical Society in Rochester on October 19, 1987.

ACKNOWLEDGEMENTS

I would like to thank my committee for their help and patience during the past year, especially my committee chairman Dr. N. S. Tzannes, upon whose ideas this work is based; and Dr. Belkerdid, who acted as my advisor in Dr. Tzannes' absence. Dr. Myler gave me much assistance as I was using the Gould computer.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

Symbol	Description
C(m)	m-th one-dimensional transform coefficient
C(m,n)	the value of a transform coefficient in a two-dimensional matrix
C_k	the k-th coefficient, same as C(m,n), except the frequency is denoted by the subscript (see g_k). For example: C_0 = C(0,0) C_1 = C(0,1) C_9 = C(1,1) for 8x8 matrix
DC	Discrete Cosine
DCT	Discrete Cosine Transform
DCT{X}(m,n) or DCT{X}	2-dimensional DCT of vector X; identical to C(m,n), except the latter does not specify the data-domain vector or the type of transform
FT	Fourier Transform
f1(x)	Any one-dimensional function
f2(x,y)	Any two-dimensional function
g(i,m)	A single value of a transform vector for a one-dimensional transform; i refers to the corresponding pixel; m refers to the coefficient
g(i,j,m,n)	A single value of a transform vector for a two-dimensional transform; i and j refer to the corresponding pixel; m and n refer to the coefficient

Symbol	Description
$g_k(i,j)$	The k-th DCT vector, same as g(i,j,m,n), except the frequency is denoted by the subscript (see C_k). For example: $g_0(i,j) = g(i,j,0,0)$ $g_1(i,j) = g(i,j,0,1)$ $g_9(i,j) = g(i,j,1,1)$ for 8x8
IDCT	Inverse Discrete Cosine Transform
IDCT{f1(x)}(i) or IDCT{f1}(i)	Value of a pixel resulting from the use of the one-dimensional IDCT upon the vector f1(x)
IDCT{f2(x,y)}(i,j) or IDCT{f2}(i,j)	Value of a data-domain pixel resulting from the use of the two-dimensional IDCT upon the vector f2(x,y).
i and j	Designate the pixel in a data-domain matrix
K	Number of coefficients retained less one
KLT	Karhunen-Loeve Transform
k	Integer increment used for looping through the frequencies or coefficients of the DCT
MEP	Maximum Entropy Principle; a reconstruction based on the MEP
MREP	Minimum Relative Entropy Principle; a reconstruction based on the MREP
mse	1. Mean squared error, MEP mse would be mse between the MEP reconstruction and the original image 2. The function that is minimized in the implementation of the MEP algorithm
m and n	Designate the coefficient in a frequency-domain matrix
N	Number of pixels along one edge of a square matrix

Symbol	Description
p	Pixel value in a uniform prior
p( )	An array used as a prior
pmf	Probability mass function
q	Integer increment used to designate Lagrangian multipliers
r x r	Size of matrix of retained coefficients
RE	Relative Entropy (same as cross-entropy)
SQDCF	Square DC function, a new mapping function (see Appendix A)
X(i) or X(i,j)	One- or two-dimensional vector of Lagrangian multipliers
x(i) or x(i,j)	Pixel value of a one or two-dimensional signal
x and y	Integers

CHAPTER 1

INTRODUCTION

This thesis will evaluate three methods of reconstruction for images coded with the Discrete Cosine Transform (DCT): the Inverse Discrete Cosine Transform (IDCT), the sequential implementation of the Minimum Relative Entropy Principle (MREP), and the Maximum Entropy Principle (MEP). Several factors will be compared: accuracy of reconstruction, computational speed, and the "characteristics" of the reconstruction. "Characteristics" refers to probing the nature of the reconstructed image, in hopes of answering such questions as "Does the MREP converge with several passes through the coefficients?" and "Is the MREP reconstruction closest to the MEP reconstruction, the IDCT reconstruction, or to the original?" The analysis will be performed on artificially created binary 16x16 images, as well as on a conventional 512x152, 256 grayness-level picture.

Binary 16x16 images will hopefully provide insights that would be harder to see with more complex images. The initial binary value of the pixels is represented by either a one or a two. Reconstruction produces images with pixel values that are real numbers. Two numerical distance measures, mean squared error and relative entropy, will be employed to find the distance between the reconstruction and the original. To obtain a binary image, the real numbered images are processed by a thresholding routine that sets the pixels to either one or two (one if below threshold, two if above). The analysis will be primarily numerical, because of the large number of reconstructed versions images.

The Gould IP8000 image processor will be used to manipulate and display the pictures. The hard copies will be obtained on a HP LaserJet printer. The reconstructions will be compared via aforesaid numerical measures of distance.

The second chapter, "Background," defines the problem of image compression, showing the advantages of the DCT method. The usual method of reconstruction using the IDCT is discussed.

The third chapter introduces an alternative method of reconstruction using entropic methods. It includes a discussion of the theoretical basis and the algorithm for the sequential implementation of the MREP and the MEP.

The fourth chapter tabulates the numerical results of this thesis. The work with the binary images is covered first; then, the work with the picture of the Golden Gate bridge is covered. The distances of between the reconstructions and the originals are measured to evaluate the reconstruction methods.

The fifth chapter is entitled "Theoretical Analysis of the MREP Convergence." It shows the iterative MREP to converge to the MEP with repeated passes through coefficients; although, this work does not include a proof that the convergence actually occurs.

CHAPTER 2

BACKGROUND

Introduction

This chapter provides background information. It begins by stating the problem of image reconstruction after transform coding. It provides the definition of a transform and enumerates the advantages of using the Discrete Cosine Transform for image compression. The implementation of the conventional reconstruction method, the Inverse Discrete Cosine Transform, is described. The chapter concludes by highlighting the advantages and disadvantages of this method.

Statement of the Problem

The general area of this thesis is reconstruction of images that have been shortened in some way. More specifically, this thesis focuses on images compressed by a transform. The Discrete Cosine Transform was chosen for this work.

The following diagram shows the process.

```
**************                 ***************        Step 2:
*            *                 *             *          Drop
*  Original  *   Step 1:       * Coefficient * >>>> High-Freq.
*   Image    * >>>IDCT>>>      *   Matrix    *      Coefficients
*   Matrix   *                 *             *           v
**************                 ***************           v
                                                         v
                                                         v
*************                  Step 3:           ***************
*           *                Reconstruct         *             *
*  Recon-   *                   with-            *  Retained   *
* struction * <<<<<<<<<<       IDCT   <<<<<<<<   * Coefficients *
*           *                  or MEP            *             *
*************                  or MREP           ***************
```

Figure 1. Discrete Cosine Transform Image Compression and Reconstruction Flowchart.

It is desired that the reconstruction be close to the original image. To achieve this, the three different methods are used in Step 3. The object of this thesis is to find the optimal method.

Transforms

Transform coding, a frequency-domain technique, is the representation of a signal in terms of pre-determined basis functions. The signal is mapped onto a corresponding set of coefficients. This section defines a transform and its inverse in terms of the basis vectors. The properties of a basis vector are discussed, and several popular transforms are mentioned (Shore 1984).

This discussion will use a one-dimensional transform for simplicity. The coefficients are obtained by taking the dot product, the original signal f(i) and the basis vector $g_m(i)$. Since the basis vector is two-dimensional, it will be written as g(i,m). The output transform coefficients are the inner product of the basis vector and the data. This constitutes the forward transform:

$$C(m) = \sum_{i=0}^{N-1} g(m,i) \cdot f(i) \tag{2.1}$$

where the original signal f(i) is N components long, and C(m) is defined for m = 0, 1, 2, ..., N-1 (N = blocksize).

There is a one-to-one correspondence between the original signal and the transform. The signal can be reconstructed by taking a weighted sum of the basis functions; this function weighting is determined by the corresponding coefficient (Tzannes 1985). The inverse transform follows:

$$f(i) = \sum_{i=0}^{N-1} g(m,i) \cdot C(m) \tag{2.2}$$

The previous two equations make up a transform pair.

The transform is defined by its basis vector, which must be orthonormal; each component must be orthogonal with every other component:

$$0 = \sum_{i=0}^{N-1} g(m,i) \cdot g(n,i) \tag{2.3}$$

for all m,n when m <> n, and they must be normal (Tzannes 1985):

$$1 = \sum_{i=0}^{N-1} g^2(m,i) \quad \text{for all } m \tag{2.4}$$

The two-dimensional forward transform is:

$$C(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n) \cdot f(i,j) \tag{2.5}$$

where:

$C(m,n)$ is defined for $m,n = 0, 1, 2, \ldots, N-1$

$f(i,j)$ is defined for $i,j = 0, 1, 2, \ldots, N-1$

$j = 0, 1, 2, \ldots, N-1$

$N \times N$ is the blocksize of $f(i,j)$

The two-dimensional inverse transform is:

$$f(i,j) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n) \cdot C(m,n) \tag{2.6}$$

There are many types of transforms used in signal processing: Slant, Walsh-Hadamard, Hadamard-Walsh, Harr, Discrete Fourier, Discrete Cosine, and Karhunen-Loeve. The amount of data compression possible is dependent upon the properties of the transform, the statistics of the data, and the quality required in the reconstructed image.

DCT and Data Reduction

Introduction

This section begins by defining the DCT. Then, its advantages for image compression are discussed. Finally, the method for reducing the data by dropping high order coefficients is explained.

Definition of DCT

All transforms can be defined in terms of the basis vectors. The DCT basis vectors will be written as g(i,m) and g(i,j,m,n) for the one-dimensional and two-dimensional cases respectively, and the coefficients as C(m) and C(m,n). C(m) corresponds to Clarke's C(p) with m equalling p (Clarke 1984). The two-dimensional transformations in this work are all square, although the results would apply to other shapes. The one- and two-dimensional transforms are given by equations (2.1), (2.2), (2.5), and (2.6) using the following definitions for the basis vectors.

$$g(m) = \frac{c(m)}{\sqrt{N}} \cdot \cos \frac{(2i + 1)m\pi}{2n} \qquad (2.7)$$

$$g(i,j,m,n) = \frac{c(m)\ c(n)}{N} \ \cos \frac{(2i + 1)m\pi}{2N} \cos \frac{(2j + 1)m\pi}{2N} \qquad (2.8)$$

$$c(l) = 1 \qquad l = 0$$

$$= 2 \qquad l <> 0$$

This is a common definition used by A. K. Jain, F. A. Kamangar and K. R. Rao, etc. (Jain 1979, Kamangar 1982). It satisfies the criteria for orthogonality as given by equations (2.3) and (2.4). There are some variations in use, some of them do not divide the vector by the square root of the size in the reverse transform and divide by the size (not its root) in the forward direction (Ahmed 1974). The reconstruction methods in this paper could be performed with these different versions, or they could be performed with any other transform whose basis vectors are known.

Advantages

There were several reasons that the DCT was chosen for these studies. Its performance with conventional image data having high inter-element correlation is virtually identical to that of the Karhunen-Loeve (KLT). The DCT basis vectors are quite similar to those of the KLT for a data correlation of 0.91. (see figures 2 and 3). Energy packing efficiency was measured as the energy in the largest N/2 coefficients. The DCT had the highest. This efficiency is desirable as it represents the effectiveness of the possible data reduction (see Figure 4) (Clarke 1984). The symmetry of the DCT transform has permitted the development of many fast algorithms for its computation (Haque 1985; Ahmed, Natarajan, and Rao 1974; Karmangar and Rao 1982). However, the KLT is very slow, as the basis vectors need to be recalculated for each image. Thus, the DCT is widely used in image processing (Shore 1984).

Choosing the Coefficients to be Retained

There are two main ways to choose the coefficients to be saved during the compression. They can be chosen by magnitude (the n largest coefficients) or position (the n lowest frequency coefficients). The latter method requires that information about the position must be sent. This position information takes up a significant amount of transmission data, and choosing by magnitude does not seem to be significantly better than by position (Mailaender 1985). This work will use the method of choosing coefficients by position.

Usual Method of Reconstruction (IDCT)

The usual way to reconstruct a transform-code image is to use the corresponding inverse transform, setting the unknown

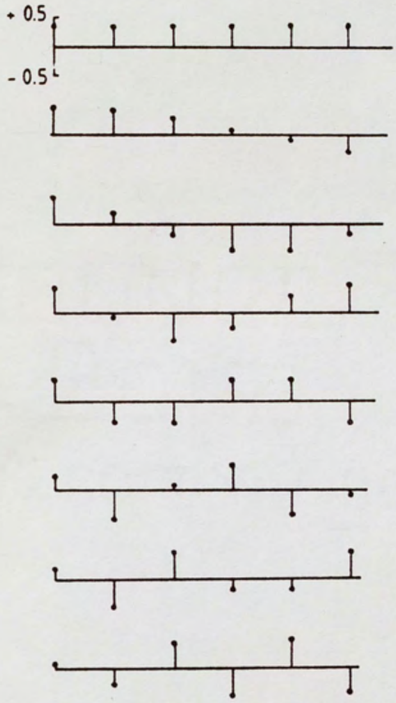


Figure 2. Discrete Cosine Transform Basis Vectors; N = 8; p Denotes Coefficient Order.

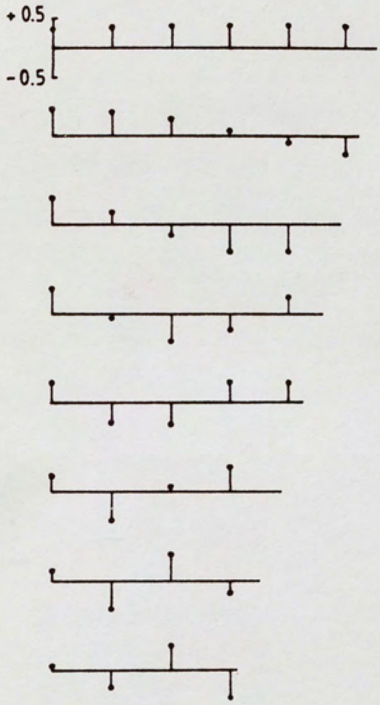


Figure 3. Karhunen-Loeve Transform Basis Vectors; N = 8; p = 0.91; p Denotes Coefficient Order.

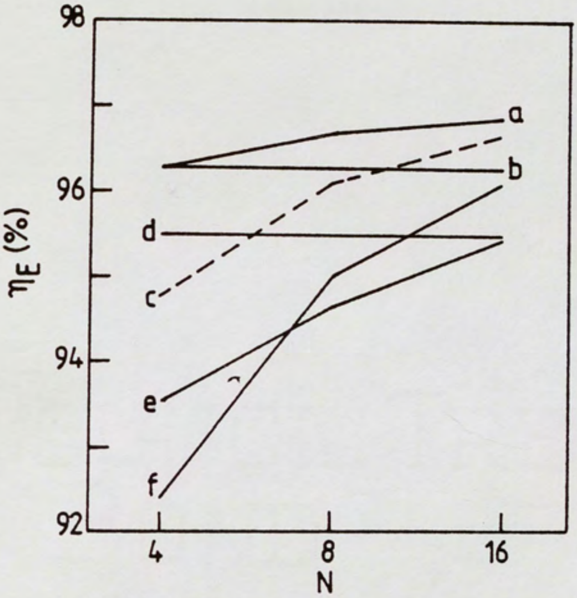


Figure 4. Energy "Packing" Efficiency η_E as a Function of Transform Block Size, $p = 0.91$. (a) DCT and KLT (0.91), (b) Slant Transform, (c) KLT (0.36), (d) WHT and Haar Transforms, (e) DFT, and (f) DST.

coefficients to zero. The formula for the inverse transform was given by equation (2.2) for the one-dimensional (1D) version and equation (2.6) for the two-dimensional (2D) version. The basis vector was given by equation (2.7) for the 1D transform and equation (2.8) for the 2D transform. The IDCT can be performed very quickly due to the symmetry of the transform (Haque 1985).

Conclusion

Transform coding is useful for compressing data before transmission or storage. The DCT is employed in this thesis, as it is very popular for data reduction (Clarke 1984). Many coefficients can be dropped without losing much information. The IDCT is the normal method of image reconstruction after DCT coding.

CHAPTER 3

ENTROPIC METHODS FOR RECONSTRUCTION OF TRANSFORM CODED IMAGES

Introduction

This chapter discusses a new method based on probability theory for reconstructing transform-coded images. The second section shows that the functions representing the images can be viewed as pmfs. The third describes Relative Entropy as a measure of distance for pmfs. The next applies the MEP and develops a fast implementation. The fifth illustrates the use of the MREP to satisfy the coefficients sequentially and a variation that makes additional passes through the sequence of coefficients.

Treatment of Images as Pmfs

The image data will be treated as a probability mass function (pmf). There are two main restrictions on pmfs; they must be positive everywhere, and the sum of the element values must be unity. The images used in this work were positive everywhere. The second restriction is not numerically necessary for the algorithms in this work, if the images have the same grayness level.

Relative Entropy

Relative Entropy has many other names: cross-entropy, discrimination information, directed divergence, and Kullback-Leibler number. It is a means of measuring the distance of two probability mass functions (pmfs). If two signals are identical, the RE between them is zero. Relative Entropy between two pmfs f(i) and p(i) is defined as:

$$RE = \int f(x) \log \frac{f(x)}{p(x)} dx \tag{3.1}$$

Applying this to discrete signals yields

$$RE = \sum_{i=0}^{N-1} f(i) \log \frac{f(i)}{p(i)} \tag{3.2}$$

the two-dimensional version is simply:

$$RE = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) \log \frac{f(i,j)}{p(i,j)} \tag{3.3}$$

This gives a working definition of RE (Shore 1984).

This measure of distance was chosen to be extremized because it satisfies four criteria: uniqueness, invariance, system independence, and subset independence (Shore 1984).

Entropy itself is defined as:

$$H = - \sum_{n=0}^{N-1} p(n) \log p(n) \qquad (3.4)$$

(units are in bits if the logarithm base is two)

This is a measure of the information in the realization of a random variable described by the pmf p(n). Because of the negative sign, maximizing this entropy corresponds to minimizing RE with respect to a uniform prior.

MEP

Introduction

The following subsection discusses the theoretical basis for the reconstruction after transform coding based upon the Maximum Entropy Principle (MEP). The simultaneous equations to implement this reconstruction are given. The third subsection discusses a new (as far as research shows) algorithm that greatly simplifies the calculations to solve these simultaneous equations. This algorithm results in a function to be driven to zero and the value of each partial derivative. This function is written in terms of the unknowns of the simultaneous equations; when it is zeroed, the solution is found. The last subsection gives a brief description of the numerical method used to zero a function when each partial derivative is known. In summary, the second section introduces a set of simultaneous equations; the third uses them to produce a function, and the fourth zeroes the function.

Theory and Formulas of MEP

The MEP technique produces an image that has maximum entropy (equation 3.4) which is equivalent to minimum relative entropy (equation 3.3) with a uniform prior. All the known constraints are simultaneously satisfied. Nothing is assumed about the unknown coefficients, unlike the inverse transform method.

These constraints are the coefficients C(m,n). Each constraint corresponds to an equation that must be satisfied. The equation is given by equation (2.5). It can be rewritten:

$$0 = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n)\, f(i,j) - C(m,n) \tag{3.5}$$

where:

N x N is the size of the block that was coded;

pixel locations are i,j = 0, 1, 2, ..., N-1

before data reduction m,n = 0, 1, 2, ..., N-1

after data reduction m,n = 0, 1, 2, ..., r

an r x r block of coefficients is retained

Equation (3.5) must be satisfied for each known coefficient.

The mathematics of extremization yield a solution:

$$f(i,j) = p \cdot \exp\left[\sum_{m=0}^{r-1} \sum_{n=0}^{r-1} X(m,n)\, g(i,j,m,n) \right] \tag{3.6}$$

where:

X is the Lagrangian multiplier

p is the value of a pixel in a uniform prior

The Lagrangian multiplier is normally written as a lowercase lambda.

The Xs are found by the simultaneous solution of:

$$C(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\ C(i,j,m,n) \tag{3.7}$$

for each known C(m,n).

Consider that a low frequency square of coefficients has been retained. The square contains r*r coefficients (each side is r long). Substitution of equation (3.6) into equation (3.5) yields:

$$C(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n) \cdot p \cdot \exp\left[\sum_{mm}^{r-1} \sum_{nn}^{r-1} x(mm,nn) \cdot g(i,j,mm,nn) \right] \tag{3.8}$$

for each known C(m,n).

For the RE to hold, the gray level of the reconstruction must equal the gray level of the prior. This gray level is given by the lowest-order coefficient. Assuming a uniform prior:

$$p = C(0,0)/N \tag{3.9}$$

For a non-uniform prior, this value would be a function p(i,j) in the above equations.

Thus, equations (3.6) and (3.8) are used for image reconstruction after transform coding using MEP.

Fast MEP for Transform-Coded Images

This section introduces a fast method of implementing MEP for transform-coded images; equations (3.6) and (3.8) are solved using a minimum of calculation. This section was written using the DCT; however, the solution is more general. Any transform whose basis vectors are known could be substituted for the DCT. The IDCT, which is used in the fast MEP reconstruction, would obviously be replaced by the inverse of the transform used for compression. The Square Discrete Cosine mapping Function (SQDCF), which is introduced in this subsection, could be replaced by another function defined in terms of the basis vectors of the compression transform (see Appendix A).

Basically, equations (3.6) and (3.8) will be rewritten in terms of the DCT, the IDCT and the SQDCF. Because of their symmetry, many fast algorithms are available for implementation.

The formula for the image, once the Lagrangian multipliers are known, is given by equation (3.6). Note that the exponent X(m,n)*g(m,n,i,j) is an IDCT. If we can replace the equation with:

$$f(i,j) = p \cdot \exp[\mathrm{IDCT}\ \{X\}\ (i,j)] \tag{3.10}$$

if $X(m,n) = 0$ for $m,n \geq r$. IDCT $\{X\}$ (i,j) is the two-dimensional coefficient vector. It is written this way to indicate it is a transform of $X(m,n)$, and its members are designated by (i,j). Note that the IDCT only needs to be evaluated once per image block.

The method for calculating the Lagrangian multipliers involves similar substitutions. These multipliers come from the simultaneous solution of equation (3.8) for each known $C(m,n)$. Call the right-hand side of this equation $F\{X\}(m,n)$, to indicate that it is a function of all of the known Lagrangian multipliers $X(mm,nn)$ and that it is two-dimensional. It can be considered a mapping, analogous to the DCT; both are written the same manner in this paper. Thus:

$$F\{X\}(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n) \cdot p \cdot \exp\left[\sum_{mm}^{r-1} \sum_{nn}^{r-1} X(mm,nn) \cdot g(i,j,mm,nn)\right] \tag{3.11}$$

$$C(m,n) = F\{X\}(m,n) \quad \text{for each } m,n = 0, 1, 2, \ldots, r-1 \tag{3.12}$$

Consider the standard distance measure mse:

$$\mathrm{mse} = \sum_{m=0}^{r-1} \sum_{n=0}^{r-1} [C(m,n) - F\{X\}(m,n)]^2 \tag{3.13}$$

If the mse is zero, the two vectors are identical. Thus, it suffices to drive the mse to zero to solve equation (3.8). This corresponds to (Shore 1984):

$$\sum_{r=0}^{M} a_r \left[\int s_r(x)\, q^t(x)\, dx - \overline{s}_r\right]^2 \leq \varepsilon^2 \tag{3.14}$$

where:

s_r is the basis vectors

$\overline{s}_r$ is the constraints

q^t is the unknown pmf

The method employed for finding the zeros of a function requires the value of that function and every first order partial derivative. Take the derivative of mse (equation 3.12) with respect to F{X}(m,n) and call this derivative dmse:

$$dmse(m,n) = 2 \cdot \sum_{m=0}^{r-1} \sum_{n=0}^{r-1} [C(m,n) - F\{X\}(m,n)] \cdot dF\{X\}(m,n) \tag{3.15}$$

where dF{X}(m,n) is the derivative of F{X}(m,n).

Take the derivative of F{X}(m,n) (from equation 3.11) with respect to X(m,n):

$$dF\{X\}(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n)^2 \cdot p \cdot$$

$$\exp[\sum_{mm}^{r-1} \sum_{nn}^{r-1} X(mm,nn) \cdot g(i,j,mm,nn)] \cdot dX(m,n) \qquad (3.16)$$

for each m,n = 0, 1, 2, ..., r-1.

The exponent in equations (3.11) and (3.16) can be replaced with an IDCT, after setting X(m,n) = 0 for n or m > r-1:

$$F\{X\}(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n) \cdot p \cdot \exp\{IDCT\{X\}(i,j)] \qquad (3.17)$$

$$dF\{X\}(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n)^2 \cdot p \cdot \exp[IDCT\{X\}(i,j)] \cdot dX(m,n) \qquad (3.18)$$

Take the constant prior outside of the summation:

$$F\{X\}(m,n) = p \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n) \cdot \exp[IDCT\{X\}(i,j)] \qquad (3.19)$$

$$dF\{X\}(m,n) = p \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n)^2 \cdot \exp[IDCT\{X\}(i,j)] \cdot dX(m,n) \qquad (3.20)$$

The double summation in the equation for F{X}(m,n) is an IDCT. The equation for dF{X}(m,n) can be modified in a similar manner. A new mapping function SQDCF is defined. It is similar to the DCT. The only difference is that its basis vector is the square of the basis vector of the DCT. The coefficients are given by:

$$\mathrm{SQDCF}\{f\}(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} g(i,j,m,n)^2 \cdot f(i,j) \qquad (3.21)$$

Much of the symmetry employed for fast evaluation of the DCT also applies to this function. This SQDCF mapping function will be investigated in Appendix A. The author has not seen the SQDCF in literature. Thus, equations (3.19) and (3.20) become:

$$F\{X\}(m,n) = p \cdot \mathrm{DCF}\{\exp[\mathrm{IDCT}\{X\}]\}(m,n) \cdot dX(m,n) \qquad (3.22)$$

$$dF\{X\}(m,n) = p \cdot \mathrm{SQDCF}\{\exp[\mathrm{IDCT}\{X\}]\}(m,n) \cdot dX(m,n) \qquad (3.23)$$

Note that the i and j of DCT{X} are no longer present, because of the outer mapping function and that nothing needs to be set to zero before taking the DCT or the SQDCF.

The term exp[IDCT{X}] is the same in both equations and only needs to be evaluated once per block (not once for every m,n as it is not a function of m or n). The DCT and the SQDCF only need to be performed once per block to find F{X}(m,n) and dF{X}(m,n). Look back at mse and dmse (equations 3.13 and 3.15), the only other calculations per block required are taking the exponent of the IDCT{X} (N*N exponential calculations), the subtraction of [C(m,n) -F{X}(m,n)] (N*N subtractions), the multiplications in the above formula for dmse and multiplying by prior (2*N*N multiplications) for finding dmse and mse. These are not much overhead. Thus, each step in the iterative solution for the

Lagrangian multipliers takes little more time than calculating the DCT thrice. As N becomes larger, the time becomes equal to the time required to take the DCT thrice. This calculation is necessary for each step in the convergence of the nonlinear solution. The procedure that solves the simultaneous equations is covered in Appendix A. This fast MEP algorithm could be applied to any transform with a known transform vector.

Zeroing a Function with Known Partial Derivatives

This section gives a quick description of a method to zero an equation when all of the partial derivatives are known. This method is used to calculate the numerical solution to equation (3.13) with the partial derivatives given by equation (3.15). It is a very sophisticated technique that the author wrote without consulting a reference.

It is easiest to view this problem spacially, where the solution is a point in a space where every axis corresponds to one unknown. In this application, the initial guess was zero. The second guess is related to the first by a step. The direction of the step is the direction in which the value of the function is diminishing the fastest. The size of the first step is an arbitrary value. The second step is 1.25 times the first step, as long as the value of the function at the second point is lower than the value of the function at the first. If the second function value is higher, the stepsize is cut in half and the algorithm backtracks to the first point.

The algorithm can also be view in terms of a man walking to the bottom of a hill. He steps in the downhill direction. If he is lower than he was before, he takes a larger step downhill; if he is higher than he was before, he goes back to his previous position and takes a smaller step downhill.

There is one additional point to make. The above stepsize is only a factor in the true stepsize and is not used directly. The true stepsize also takes into account the slope and height of the hill at the current location.

The algorithm is resilient. Its method will not diverge, as a standard 1D Newton method can. It can become stuck at a local minimum or at an inflection point; however, neither occur during the MEP calculation.

MREP

Introduction

The following subsection justifies the use of Minimum Relative Entropy (MREP) and discusses the implementation to reconstruct images coded with the DCT. The third subsection explains the iterative variation that is computationally more complex.

Shore and Johnson originally proposed the MREP as a means of reconstructing a function. The method combines a previous estimate of the signal with additional information to obtain a second estimate. This additional information (prior knowledge) comes from the coefficients of the DCT. The prior knowledge is given by equations (2.1) and (2.5). Thus, MREP is applied to reconstruction after transform coding.

Previously, N.S. Tzannes and M.S. Tzannes (1986) introduced a new universal method of decoding transform-coded images using the principle of Minimum Relative Entropy (MREP). In this paper, we examine the MREP's iterative convergence properties by applying MREP to image data compressed by the Discrete Cosine Transform (DCT) and running the iterative algorithm until it stabilizes.

The minimization of a two-dimensional function subject to many constraints, and the usual lack of a prior guess makes the use of the regular MREP quite difficult to implement in practice. However, it can be done if one assumes a uniform prior; then, it is identical to the MEP. To alleviate these problems, the following sequential MREP algorithm was suggested in "Reconstruction of Transform-Coded Data by RE Minimization" (Tzannes and Tzannes 1986).

1. It is desirable to treat the image as a probability mass function (pmf) and there are physical justifications for this treatment (Shore 1984). This could be accomplished by normalizing the image so that its pixel values add up to unity. However, this normalization is not numerically necessary, for this algorithm. If each image in a group has the same sum of pixel values (same dc level), the images can be treated as pmfs, for this work.

2. Assume $p(j,k)$ is a uniform image and maximize the entropy of $f(j,k)$ subject to a single constraint that is specified by the first retained coefficient. This results in a first estimate of the reconstructed image. Maximization of the entropy of the image refers to maximizing RE of $f(i,j)$ when $p(i,j)$ is uniform. This maximization of the entropy reduces to merely producing a uniform image with the same grayness level as the original. The IDCT given the first coefficient and the rest set to zero would produce an identical image. Thus:

$$f_0(i,j) = C(0,0)/N \tag{3.24}$$

for $i,j = 0, 1, 2, \ldots, N-1$. This is analogous to equation (3.9) for the MEP prior.

3. The first estimate obtained by the previous step is now used as the prior $p(j,k)$ in the MREP with the second retained coefficient as a constraint. The minimization is simple since only one constraint is used and one Lagrangian multiplier needs to be evaluated. The result is the second estimate of the reconstruction.

4. Repeat step 3 using as a prior the previous estimate and, as a constraint, the next coefficient until all retained coefficients are used. The n'th estimate is given by:

$$f_k(i,j) = f_{k-1}(i,j)e^{-\lambda_k g_k(i,j)} \tag{3.25}$$

where:

f_{k-1} is prior

X is the Lagrangian multiplier

$g_k(\ )$ is the transform basis vector

for the one-dimensional case. The X is found by applying the constraint:

$$C_k = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_k(i,j)\ g_k(i,j) \tag{3.26}$$

where the subscript on the "f" refers to the k'th estimate, and the subscript on the "C" designates that the k'th coefficient is being used.

Substitution yields (Tzannes and Tzannes 1986):

$$C_k = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{k-1}(i,j) e^{-X_k g_k(i,j)} \tag{3.27}$$

where $g_k(j,k)$ is the transform basis element whose average over the image represents the k'th coefficient.

This differs slightly from Shore's MREP extremization (1984):

$$q(X) = p(X)\ \exp[-\ \tau - \sum_{r=0}^{M} \beta_r\ s_r(X)] \tag{3.28}$$

where the β_r and τ are Lagrangian multipliers. β is determined by the restriction that the dc level of the prior equal the dc level of the original. is determined by the new information (the constraint) (Shore 1984).

In this implementation, the dc level restriction is relaxed. It is the same as Shore's when $\tau = 0$. This reduces the system to have one equation and one unknown. Subsequently, each pixel is multiplied by a constant, to normalize the dc level. The difference in dc level is less than one percent, for the binary images. Thus, while there is a theoretical difference, the methods are numerically close enough for practical purposes.

This method will be referred to as the one-pass MREP or the sequential MREP, as all of the constraints are passed through one time. Preliminary results presented by N.S. Tzannes and M.A. Tzannes (1986) showed that this algorithm performs well, often better than the IDCT under the same compression ratios. Its use can lead to greater compression of an image for transmission or storage.

Iterative MREP

The above algorithm suffers from one fundamental theoretical deficiency which is investigated in this paper. The MEP and MREP, originally proposed for estimating pmfs or spectra demand that the extremization of the functionals be performed under under the simultaneous satisfaction of all prior knowledge (in the form of constraints). The progressive algorithm discussed above does not do this; it utilizes one constraint at a time and uses the solution obtained in each step as a prior for the next step. Each recursive solution obtained in this manner satisfies only the most recent constraint and not all the previous ones, and thus the last solution satisfies only the last constraint and not all of them. Of course, it is obvious that the information contained in each constraint is not lost -- it gets incorporated in the resultant estimate which is the prior for the next solution. Therefore, the last solution is not only the last constraint, it is also weighted by the prior which embodies in it information about all the previous constraints. Thinking of the Relative Entropy as a distance, the last solution is closest to the prior (which satisfies the previous constraint), which is closest to the previous prior (which satisfies the constraint before), etc. Thus, philosophically, at least, the information in all the constraints is carried on to the last solution. Mathematically, however, the final reconstruction only satisfies the last constraint with no guarantee that all the other constraints are satisfied.

There are three methods to studying this problem.

1. Theoretically

2. Practically, by comparing the results of the progressive algorithm to the regular approach, the IDCT, and seeing which performs better under the same conditions.

3. Practically, by using the last reconstruction as a prior, and running through the constraints a second, a third time, etc., until the solution stabilizes. These will be referred to as additional passes through the constraints. This will be referred to as the iterative MREP, when more than one pass through the coefficients is used.

All three methods are investigated. Methods 2 and 3 are covered in Chapter 4. Method 1 is the subject of Chapter 5.

Conclusion

Relative entropy provides alternative methods for reconstructing transform-coded images. If the images are treated as pmfs, the principles of Maximum and Minimum Entropy can be applied. Two algorithms for entropic reconstruction have been presented: one based upon Maximum Entropy and one based upon Minimum Relative Entropy. The former was optimized for speed.

passes is increased. In the sixth, the MEP mse is compared to the IDCT mse. The seventh measures MREP mse relative to the MEP mse to compare the methods and to test for MREP convergence. Finally, in the last subsection, the distance between all of the reconstructions themselves is measured.

Advantages of 16x16 Binary Images

The size of the binary images yielded two advantages. Firstly, it facilitated program development; secondly, it allowed very long calculations to be performed. An example of a long calculation is the MREP reconstruction of a 16 x 16 image, using 1 through 10 pass, for 1x1, 2x2, 3x3, ..., 16x16 coefficients retained.

Test Images

Five images were used: A5, E, EO, NewO, and the O (see Appendix C). The EO is sometimes referred to as the OE. The NewO is a symmetric "o"; the other one is non-symmetric. The first four were used in all of the studies (see Appendix C).

CHAPTER 4

APPLICATION

Introduction

The three methods of reconstruction, the IDCT, the MREP, and the MEP, will be applied to two types of images: binary images and a black-and-white picture of the Golden Gate Bridge. The second section describes the work with the binary image; the third describes the work with the bridge picture. The fourth section discusses the error of the numerical solution to the non-linear equations.

Binary Images

Introduction

This section focuses on the methodology and results from the 16x16 binary images. The next subsection explains why the 16x16 binary images were studied. In the following subsection, the test images are described. The fourth analyzes the mse distance between the reconstructions and the original to judge MREP performance relative to the IDCT. The fifth subsection notes the convergence of the iterative MREP by checking the mse and RE between the reconstruction and the original as the number of

MREP mse vs. IDCT mse

TABLE 1

MREP mse vs. IDCT mse

RANGE OF COEFFICIENTS WHERE MREP MSE WAS LESS THAN IDCT MSE	NUMBER OF PASSES	IMAGE USED
3x3, 5x5 - 9x9	1	A5
3x3 - 15x15	>1	A5
5x5 - 8x8	1,2	E
5x5 - 8x8, 13x13	>2	E
4x4, 7x7 - 12x12, 14x14	1	E0
3x3 - 4x4, 6x6 - 15x15	2	E0
3x3 - 15x15	>2	E0
3x3 - 13x13	1	New0
3x3 - 15x15	>1	New0

One of the largest differences in mse occurred with the image New0. With 13x13 coefficients retained, the MREP 3-pass had an mse that was 20% less than the mse of the IDCT.

MREP Convergence

This data shows that the reconstruction improves markedly with the second pass and some with the third. Additional passes are not very significant.

Examination of the tables reveals that the RE between the reconstruction and the original, H(recn, org), is monotonically decreasing. The mse between the reconstruction and the original decreases except for infrequent small jumps. These jumps in mse do correspond to reductions in RE. For instance, with the image A5, for 6x6 coefficients retained, between the 4th and the 5th pass, the mse goes up from 5.53993154 to 5.53993470; the RE declines from 10.47380396 to 10.47380261. Another important observation is the behavior when all 16x16 coefficients are retained; MREP converges to zero mse and zero RE, as the number of passes increases.

MEP mse vs. IDCT mse

The ratio of the MEP mse over the IDCT mse was plotted for images A5, E, E0, and New0 for a convergence limit of $1/10^3$. The other 0 was tabulated, but not plotted. The A5 ratio was also plotted for a convergence of 1/10. The following table (Table 2) will be made by just reading the tables where the graphs do not have sufficient resolution.

TABLE 2

MEP mse vs. IDCT mse

RANGE OF COEFFICIENTS WHERE MEP MSE WAS LESS THAN IDCT MSE	IMAGE USED
3x3 - 15x15	A5
5x5 - 8x8, 13x13	E
3x3 - 15x15	E0
3x3 - 15x15	New0
Convergence limit was $1/10^3$	

Thus, the MEP performance exceed the IDCT in the same cases as the MREP with 3 or more passes.

MREP mse vs. MEP mse

A different approach was tried. The ratio of MEP mse to MREP mse was plotted as a function of the number of retained coefficients. This allowed the MREP 1-pass /MEP to be on a different scale than the MREP 2-pass /MEP. The graphs were plotted for 1-, 2-, 3-, 4-, and 10-pass MREP. Thus, the convergence could be viewed from graph to graph.

An improvement on the graphs was achieved by only to plotting up to 13x13 or 14x14 retained coefficients. The graphs tend toward extremes at the ends, and this can make the scale so large that one cannot see what is going on over the majority of the coefficient range.

The results illustrated convergence very well. The following discussion of this ratio range is valid for 2x2 to 13x13 coefficients retained. For one and two passes, the MREP performed worse than the MEP; the ratio was greater than one. The mse ratio after one pass was as high as 10 (approximately). The two-pass ratio was less than 1.1. After three passes, the MREP mse was within 1% of the MEP mse (and within 0.1% for two of the pictures). Skipping to the 10-pass MREP, the difference in mse between the two methods was less than 0.01%.

A similar procedure was used to see if the MREP mse or the MREP reconverged to that of the IDCT. The ratio of the MREP re to the IDCT was plotted as a function of coefficients retained. Each different graph corresponded to a different number of passes. This procedure made the convergence of the MREP to the MEP obvious; the lack of convergence of the MREP to the IDCT was also obvious.

Distance Measures Between the Reconstructions Themselves

At this time, it was decided to directly measure distance between the reconstructed images themselves. This is the subject of Appendix E. A quick look shows that MREP converges to MEP.

Picture of Golden Gate Bridge

Introduction

The picture of the Golden Gate Bridge (see Appendix C) was included to test the performance of the algorithms upon realistic data. The picture is composed of 512x512 pixels with 256 greyness levels. The information content in bits per pixel was 6.28 for the pixels taken one at a time (equation 3.4).

The second subsection discusses the blocksizes used in the calculations; the third states the number of coefficients retained. In the fourth, the performance of the MEP is compared to that of the IDCT under the mse criteria. The fifth describes the mse of the MREP. The amount of calculation for the MEP relative to the DCT is tabulated in the sixth subsection.

Blocksize

The 512x512 picture is broken up into smaller blocks for processing. The compression and reconstruction are then performed upon each separately. This is much faster than working with the picture as a whole. The block sizes are 4x4, 8x8, and 16x16. Blocking in these sizes is common in video processing (Haque 1985 and Mailaender 1985).

Number of Retained Coefficients

The three methods of reconstruction were used: IDCT, MREP, and MEP. For each blocksize 1x1, 2x2, 3x3, and 4x4 coefficients are retained.

MEP mse vs. IDCT mse

The mse between the reconstructions and the original were measured for the picture of the Golden Gate Bridge. The MEP was superior to the IDCT for five out of nine cases (Appendix H).

Mse of MREP

The second pass of the MREP can decrease the mse by over 1%. The MREP 2-pass method produces a mse that was within 0.1% of the mse of the MEP. Generally, the third pass of the MREP does little to change the mse. The only exception to the above observations occurred in the case of a 4x4 blocksize and 4x4 coefficients retained; the mse decreases from 176.7 with one pass, to 20.4 with two passes, and to 4.6 with 3 passes. This implies a convergence limit of 0.0 mse, which is reasonable.

Number of MEP Iterations

The average number of iterations per block were calculated for the MEP algorithm. The image was the picture of the Golden Gate Bridge; the blocksize was 4x4.

For 3x3 coefficients retained, the MEP required 4.4 (approximately) iterations per block. With each iteration taking about three times as long as an IDCT, the time for reconstruction was about 13.2 times the time required for an IDCT.

For a blocksize of 8x8, Table 4 is obtained.

TABLE 3

MEP SPEED WITH A 4X4 BLOCKSIZE

AVERAGE ITERATIONS PER BLOCK	COEFFICIENTS RETAINED
1	1x1
3.77338	2x2
4.41498	3x3
4.75409	4x4

TABLE 4

MEP SPEED WITH AN 8X8 BLOCKSIZE

AVERAGE ITERATIONS PER BLOCK	COEFFICIENTS RETAINED
1	1x1
4.94165	2x2
5.63916	3x3
5.95435	4x4
6.16553	5x5

Error Allowed in Numerical Solution of Non-linear Equations

There are two functions that are iteratively driven to zero to find the Lagrangian multipliers. These two functions are equation (3.22) for the MEP and equation (3.28) for the MREP. The numerical solution necessitates an error value; if the absolute value of the function is less than the error value, the iteration stops.

Error of Non-linear Solution

Two limits of error for the numerical non-linear equation solution algorithm were widely used: $1/10^6$ and $1/10^3$. The former was used for the comparisons presented in this thesis (unless otherwise noted), although there was only a small difference in performance. A limit of 1/100 increases the mse of the reconstruction. From a standpoint of computational efficiency, $1/10^3$ is optimal.

Conclusion

The work with the binary images illustrated the general superiority of MEP and the MREP 3-pass to the IDCT. The reconstructions of the Golden Gate Bridge picture showed the MEP and MREP 3-pass were only slightly superior to the IDCT. Unfortunately, one grey-level picture is not sufficient to judge the overall performance of these reconstruction methods on pictorial data. The studies found the MREP to converge to the MEP, as the number of passes increased.

More work needs to be done to perform quantitative and qualitative analysis of the reconstructions after coding real picture data. One could test the affect of the order of the coefficient sequence is important for the MREP technique. Also, one could employ additional prior knowledge in the reconstruction. For example, "extremize the entropy to produce an image that satisfies the knowledge we have about the coefficients (include quantization error) and has integer pixel values between 1 and 256, inclusive."

CHAPTER 5

THEORETICAL ANALYSIS OF MREP CONVERGENCE

Introduction

This chapter shows that the limit of the convergence of the MREP is the same as that of the MEP, as the set of equations in the MREP are the same as the equations in the MEP. No attempt will be made to prove that this convergence does actually occur.

The next section illustrates the convergence limit in a simple one-dimensional example where two coefficients are retained and gives some more information about the implementation of the MREP algorithm. The third section presents a slightly more formal proof.

An Illustration

MREP, looping once through all the known, was said to only satisfy the last coefficient, but contain information from the previous constraints. It was shown numerically to converge for one constraint at a time. When one is solved for one variable, the solution is effectively substituted into the next equation. This constitutes a set of nonlinear equations.

MREP assumes a uniform prior and satisfies the first unknown. The one-dimensional version of the satisfaction of the known constraint is given by equation (3.28); the resulting solution is given by equation (3.26). Solving these for the first constraint (with k=1):

$$C_1 = \sum_{i=0}^{N-1} f_0(i)\, e^{-X_1 g_1(i)}\, g_2(i) \tag{5.1}$$

$$f_1(i) = f_0(i)\, e^{-X_1 g_1(i)} \tag{5.2}$$

with k = 2 for the 2nd constraint.

$$C_2 = \sum_{i=0}^{N-1} f_1(i)\, e^{-X_2 g_2(i)}\, g_2(i) \tag{5.3}$$

Substituting in solution for f_1:

$$C_2 = \sum_{i=0}^{N-1} f_0(i)\, e^{-X_1 g_1(i) - X_2 g_2(i)}\, g_2(i) \tag{5.4}$$

simplifying:

$$C_2 = \sum_{i=0}^{N-1} f_0(i)\, e^{-X_1 g_1(i) - X_2 g_2(i)}\, g_2(i) \tag{5.5}$$

If there were only two coefficients (not including the zero-order coefficient) retained in this one-dimensional matrix, the above equations represent one pass.

For a second pass, the solution for X_2 would be put into:

$$C_1 = \sum_{i=0}^{N-1} f_0(i)\, e^{-(X_3 + X_1)g_1(i) - X_2 g_2(i)}\, g_1(i) \tag{5.6}$$

and X_3 would be the unknown. This can be written as solving for X_1, where X_1 = sum of all prior solutions = $X_3 + X_1$.

$$C_1 = \sum_{i=0}^{N-1} f_0(i)\, e^{-X_2 g_2 V(i)}\, g_1(i) \tag{5.7}$$

Then X_2 would be solved for as the unknown, with X_1 constant.

$$C_2 = \sum_{i=0}^{N-1} f_0(i)\, e^{-X_1 g_1(i) - X_2 g_2(i)}\, g_2(i) \tag{5.8}$$

Additional passes would iterate through the previous two equations.

These two equations, that are being driven to zero, are the same as the equation (3.6) for the MEP. Technically, the MEP would also be solving for X_0, by the MREP notation. This is also solved by the MREP, via the normalization after finding each

Lagrangian multiplier. Thus, MREP solves the same simultaneous equation as the MEP.

The zero-order coefficient could be looped through once per pass, instead of being satisfied after every non-linear solution; this method would probably converge. There are two main advantages in performing the normalization each time.

Firstly, it is very quick, as it simply entails summing the pixels and multiplying them by a factor. Actually, it would probably suffice to add the same constant to each pixel. Since addition is faster than multiplication, it would probably be a bit faster. This was not investigated numerically.

Secondly, the definition of relative entropy requires that the two images have the same dc level; that is the basis of the MREP equation. Although, the success of the "simultaneous" method implies that the normalization might only be necessary once per pass.

Non-rigorous Proof

This proof assumes that a system of simultaneous equations (specifically the MEP equation, 3.7) can be solved with the following algorithm. Assume all unknowns, save one, are set to an initial value and solve one equation for that unknown. Substitute this solution into the next equation and solve for another unknown, setting the other unknowns equal to the initial value. Repeat until all the equations are used. (The one-pass MREP is an implementation of this method upon the MEP equation.) This loop through the equations is repeated until the values obtained for the unknowns cease to change (converge). If this happens, the solution to the set of simultaneous equations has been found; there is no proof that this will occur for an arbitrary set of equations.

The main task is to show that the iterative MREP equations reduce to the MEP set of simultaneous equations. This is straightforward. Two "compressions" are involved. First, an expression for the solution after the one-pass MREP is obtained from the separate equations. Secondly, this expression is "compressed" to show the result after several passes of the MREP. The generalized form will correspond to the MEP equation.

The main problem lies in the notation. Many different subscripts and increments will be used and it may become confusing. A quick overview will be helpful. The data-domain increments are "i" and "j." The frequency-domain increments are "m" and "n;" however, they are not used because each known frequency coefficient corresponds to a separate non-linear equation. The equations are more easily visualized sequentially; thus, the subscript "k" will designate the frequency. There are K known coefficients. The range of "k" will be 1 to K. If $k = 0$, the prior for a given pass is indicated. The Lagrangian multipliers, "X," will be superscripted by a "q,r." The frequency is designated by "q" ($q = 0$ means zero frequency) and "r" is used to designate the pass ($r = 1$ means the first pass).

The expression for the solution after one pass is:

$$f_{K,1}(i,j) = f_{0,1}(i,j)\, e^{\sum_{q=0}^{k} X_{q,1}\, g_q(i,j)} \tag{5.9}$$

The ",1" means this is the first pass. Each $X_{q,1}$ came from:

$$C_K = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{0,1}(i,j)\, e^{\sum_{q=0}^{k} X_{q,1}\, g_q(i,j)} \tag{5.10}$$

for $k = 1, 2, 3, \ldots, K$.

The above form represents K separate equations. It is solved for $k = 1, 2, \ldots, K$ in sequence. The unknown Lagrangian multipliers are assumed to be zero. (The fact that $X_{1,1}$ represents the greyness level is not important. It will be treated as any other Lagrangian multiplier.) After the first multiplier, $X_{1,1}$, is found, k is set to 2 to find the second. The estimate of $X_{1,1}$ is used to find $X_{2,1}$. The estimates of $X_{1,1}$ and $X_{2,1}$ are used to find $X_{3,1}$. This process continues until the first pass is complete, using all the known coefficients.

Next, the generalized form of the MREP solution after several passes will be developed. The solution after the second pass of the MREP is:

$$f_{K,2}(i,j) = f_{0,2}\, e^{\sum_{q=0}^{k} X_{q,2}\, g_q(i,j)} \tag{5.11}$$

Substituting in the value of $f_{0,2}(i,j)$ yields:

$$f_{K,2}(i,j) = f_{0,2}\, e^{\sum_{q=0}^{k} X_{q,1}\, g_q(i,j)} \cdot e^{\sum_{q=0}^{k} X_{q,2}\, g_q(i,j)} \tag{5.12}$$

where $f_{0,1}$ is the uniform prior of the first pass.

Note that:

$$f_{0,2}(i,j) = f_{K,1}(i,j) \tag{5.13}$$

as the prior of the second pass equals the value of the solution after the first pass. More generally, with "r" for the number of the pass:

$$f_{0,r}(i,j) = f_{K,r-1}(i,j) \tag{5.14}$$

The Lagrangian multipliers for equation (5.12) are obtained from:

$$C_K = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{0,2}(i,j)\; e^{\sum_{q=0}^{k} X_{q,2}\, g_q(i,j)} \; g_K(i,j) \qquad (5.15)$$

for $k = 1, 2, 3, \ldots, K$.

This is identical to equation (5.10) except for the change of the second subscript of the f( ) and the X( ) from a "1" to a "2."

The value of prior $f_{0,2}(i,j)$ can be substituted in to yield:

$$C_K = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{0,1}(i,j) e^{\sum_{q=0}^{k} X_{q,1}\, g_q(i,j)} \cdot e^{\sum_{q=0}^{k} X_{q,2}\, g_q(i,j)} \cdot g_K(i,j)$$

$$(5.16)$$

for $k = 1, 2, 3, \ldots, K$.

This was the same substitution that was performed between equations (5.11) and (5.12). Equation (5.12) gives the image after the second pass; equation (5.16) gives the Lagrangian multipliers after the second pass. A more general solution is desired. Next, the equations for a third pass will be written to clarify the pattern.

$$(5.17)$$

$$f_{K,3}(i,j) = f_{0,1} e^{\sum_{q=1}^{k} X_{q,1}\, g_q(i,j)} \cdot e^{\sum_{q=1}^{k} X_{q,2}\, g_q(i,j)} \cdot e^{\sum_{q=1}^{k} X_{q,3}\, g_q(i,j)}$$

where $f_{0,1}$ is the uniform prior of the first pass.

$$C_K = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{0,1}(i,j) e^{\sum_{q=1}^{k} X_{q,1} g_q(i,j)} . e^{\sum_{q=1}^{k} X_{q,2} g_p(i,j)} . e^{\sum_{q=1}^{k} X_{q,3} g_q(i,j)} \tag{5.18}$$

for k = 1, 2, 3, ..., K.

The previous two equations have three arrays of Lagrangian multipliers. The solution of the each non-linear equation comes up with a new value for each Lagrangian multiplier. This new solution is in effect merely added to the old solution. This is the way the algorithm is implemented.

There is another way of viewing this process that is numerically equivalent. The old solution of a Lagrangian multiplier can be considered discarded when it is solved for again. It does not affect the new solution (except for the addition). The three arrays of unknowns can be algebraically combined into one. This new array will be called Y( ). Rewriting the two MREP equations yields:

$$f(i,j) = f_{0,1} \, e^{\sum_{q=0}^{k} Y_q \, g_q(i,j)} \tag{5.19}$$

where $f_{0,1}$ is the uniform prior of the first pass.

$$C_K = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{0,1}(i,j)\, e^{\sum_{q=1}^{k} Y_q\, g_q(i,j)} \cdot g_K(i,j) \qquad (5.20)$$

for k = 1, 2, 3, ..., K.

At every pass, the Y( )s are solved for again. The old value corresponding to the one being solved for is only used as a prior, it is otherwise discarded. The other old values are used in the equation. There is now only one array of unknowns, no matter how many passes are made. Equation (5.20) represents one set of simultaneous equations derived from the sequence of equations from the iterative MREP. With the use of "m,n" in lieu of "q" to designate the frequence, equations (5.19) and (5.20) are identical to equations (3.6) and (3.8) of the MEP solution.

Conclusion

Thus, if the iterative MREP algorithm does converge, it is mathematically equivalent to MEP. However, no attempt has been made to prove that this convergence actually does occur.

APPENDICES

APPENDIX A

SQDCF, A MAPPING FUNCTION

This appendix contains a short description of a mapping function called Squared Discrete Cosine Function and the calculation of the SQDCF via the DCT. This unusual function is of interest as it is used in the fast MEP method.

The difference between this function and the DCT is that the "basis vectors" of the SQDF are the DCT vectors squared. As these vectors are not orthogonal, it is not a transform, only a mapping function. The mapping of a one-dimensional vector, X, of length N is:

$$\mathrm{SQDCF}\{X\}(m) = \sum_{i=0}^{N-1} X(i) * \cos^2 \frac{(2i + 1)m\pi}{2N} \tag{A.1}$$

The two-dimensional version is:

$$\mathrm{SQDCF}\{X\}(m,n) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} X(i,j) * \cos^2 \frac{(2i+1)m\pi}{2N} * \cos^2 \frac{(2j+1)n\pi}{2N} \tag{A.2}$$

The next part shows the calculation of the SQDCF via the DCT. By using the simple substitution, equation (A.1) can be written:

$$\mathrm{SQDCF}\{X\}(m) = \sum_{i=0}^{N-1} X(i) * \frac{1}{2} * \{1 + \cos \frac{(2i + 1)m\pi}{N}\} \qquad (A.3)$$

$$\mathrm{SQDCF}\{X\}(m) = \frac{1}{2} * \sum_{i=0}^{N-1} X(i) + \frac{1}{2} \sum_{i=0}^{N-1} X(i) * \cos \frac{(2i + 1)m\pi}{N} \qquad (A.4)$$

The calculation of equation (A.4) is only slightly slower than the calculation of the standard DCT.

APPENDIX B

COMPUTER PROGRAMS

The following program performs reconstruction of a 512x512 picture using MEP and IDCT.

```
c        program main
c          tues 7/14
c         su 7/12 removed calc of g( ,,,,)
c               removed extra writes
c               made in terms of 16 for subsititution(routine def_g.)
c               changed loop thru blocks to stop at a
c                 function of block size (not 31 but 512/siz-1)
c                in re changed output filenames
c        It will take the 2d-dct w/o using 1d-dct's.
c        The idct is done in a similar manner.
c        the non-integer variables are : double precision
c                  the simultaneous will converge to
c                  1/10**3 in lieu of 1/10*3
c                  the iterative will comverge to
c                  1/10**3 in lieu of 1/10*3
c
c
c        su11m0s6    mrep not used sim to 10**-6
c        su11m0s20    mrep not used sim to 10**-20
c               wk
c          1110    remove -x  used x
c          1100    speed drtni
c          1050 wk
c        su1050    sim to 1/10**12
c        su850     removed some extra writes
c          845 wk
c        su845     the simultaneous will converge to
c                  1/10**7 in lieu of 1/10*3
c                  the iterative will comverge to
c                  1/10**7 in lieu of 1/10*3
c          650 wk
c        su650     1-10 passes thru coef
c          640 wk
c        su640     correctly labels the output files (no of iter)
c        su630 wk  is fastest   , the mrep uses previous output as
c                  prior
c        s500 wk
c        shifts simultaneous satifaction output
c        uses coef 1x1-16x16
c
c
c
c
c
c
```

```
c
c
c
c
c
c
      double precision g(16,16,16,16) ,g1(16,16),gsq(16,16)
c     g(16,16,16,16) only used in re.f but left to keep proper
c        spacing for common block
      common g ,g1,gsq
      call defg1sq
      call menu
      stop
      end
c
      subroutine defg1sq
      integer i,m
      double precision g(16,16,16,16), g1(16,16),gsq(16,16), pie,c,y
      common g, g1 ,gsq
      y = -1.0
      pie = dacos(y)
      do 10 m = 1 ,16
        if (m.eq.1) then
                  c = 1.0
                else
                   c= 2.0**0.5
                     end if
        do 20 i = 1,16
      g1(i,m) = c/((16.0)**0.5)*dcos ((2.0*i-1.0)*(m-1.0)*pie/2.0/16.0)
      gsq(i,m)= g1(i,m)**2.0
c          write    (*,*) 'dfg1: ', g1(i,m)
  20    continue
  10  continue
      end
c
c      character tpl1*80,tpl2*80,tpl3*80,tpl4*80,tpl5*80,tpout20*20
c      double precision tp(0:15,0:15)
c
c      double precision x
c      character rl1*80,rl2*80,rl3*80,rl4*80,rl5*80
c      character tl1*80,tl2*80,tl3*80,tl4*80,tl5*80
c      character ol1*80,ol2*80,ol3*80,ol4*80,ol5*80
c
      subroutine menu
      integer c,fifteen,iter,i,j,vblockno,hblockno
c      c = number of non-zero dct coefficients
      integer*2 bigorg(0:511,0:511),bigidct(0:511,0:511)
      integer*2 bigsim(0:511,0:511)
      integer retain
      double precision dcta(0:15,0:15),orga(0:15,0:15),idcta(0:15,0:15)
```

```
      double precision rela(0:15,0:15),temp(0:15,0:15),sima(0:15,0:15)
      character name*20
      read (*,*) retain
c      retain =3
      call redar(bigorg)
c     loop thru ea 16x16 block
      do 10 vblockno=0,(512/16)-1
      do 11 hblockno=0,(512/16)-1
      do 20 j=0,15
      do 30 i=0,15
       orga(i,j) = float(bigorg( i+vblockno*16,j+hblockno*16))
   30 continue
   20 continue
      call cmtdct(orga,dcta,1)
      call afilter(dcta,retain)
      call d2mm(retain,dcta,sima)
      do 40 j=0,15
      do 50 i=0,15
       bigsim( i+vblockno*16,j+hblockno*16)= int(sima(i,j)+0.5)
   50 continue
   40 continue
c
c       write (*,*)  'did a block !!!  ',hblockno,'vbk= ',vblockno
   11 continue
c       write (*,*) ' Fortran: vblockno= ', vblockno,' ************'
   10 continue
      name = 'me'
      call printa(bigsim,name)
      end
c
c
      subroutine d2mm(p,coef,outa)
      double precision outa(1:16,1:16)
c
      double precision coef(16,16)
      integer p
c
      integer i,siz
      integer ier,iend
      double precision prior,xst,eps,rsiz
      double precision dexpo(16,16),x(16,16)
      double precision g2(16,16,16,16)
      common g2
      logical zeros
      siz = 16
      rsiz = 16.0
c       if your going to subroutine ascale
      prior = coef(1,1)/rsiz
      write (*,*) 'd2m: cof(11) ',coef(1,1),' prior ',prior,' siz',siz
      eps = 1.0/10.0**3.0
```

```
      iend =500
      xst =0.0
      call m2drtni(x,xst,eps,iend,ier,coef,zeros,p)
      if (zeros) then
         write (*,*) ' ********* zero slope **'
      end if
c       do 10 ii=1,siz
c       do 20 jj=1,siz
c            dexpo(ii,jj)= 0
c            do 30 mm=1,  p
c            do 40 nn=1,  p
c              if (((mm.eq.1).and.(nn.eq.1)).eq..false.)then
c                 dexpo(ii,jj)=dexpo(ii,jj)+x(mm,nn)*g2(ii,jj,mm,nn)
c              end if
c 40         continue
c 30         continue
c            dexpo(ii,jj) = dexp(dexpo(ii,jj))
c 20    continue
c 10    continue
c       x(1,1) = 0.0
      call afilter(x,p)
      call cmtdct(x,dexpo,2)
      do 50 i=1,siz
      do 60 j=1,siz
c            write (*,*) '284  prior = ',prior,' dexp= ', dexpo(i,j)
        outa(i,j) = prior*dexp(dexpo(i,j))
 60   continue
 50   continue
      call  shift(coef,outa,dexpo)
      call       copy (dexpo,outa)
      end
c
      subroutine m2drtni(x,xst,eps,iend,ier,coef,zeros,p)
      double precision x(16,16),xst,eps,coef(16,16)
      integer iend,ier,p
      logical zeros
c
      integer m,n,irept
      double precision  stepfactor,oldf,f,derfsum ,stepsize
      double precision  oldx(16,16), derf(16,16)
      isiz =16
      siz =16.0
      ier= 0
      oldf=10000
      stepfactor = 2.0
       call set(x,xst)
c       x(1,1)=0
       call m2fct(x,f,derf,coef,p)
      irept=0
 30   continue
```

```
      zeros= .true.
      derfsum=0.0
      do 40 m=1,p
      do 50 n = 1,p
c       if (((m.eq.1).and.(n.eq.1)).eq..false.) then
        derfsum=derfsum+(derf(m,n))**2
c       end if
 50   continue
 40   continue
      derfsum= derfsum**0.5
      stepsize = f/derfsum*stepfactor
      do  60 m=1 , p
      do 70  n =  1 , p
c       if (((m.eq.1).and.(n.eq.1)).eq..false.) then
        oldx(m,n) = x(m,n)
        if ((derf(m,n).eq.0).eq..false.) then
              zeros=.false.
              x(m,n)=x(m,n)-stepsize*derf(m,n)/derfsum
        end if
c       end if
  70  continue
  60  continue
      oldf=f
      call m2fct(x,f,derf,coef,p)
      if ((oldf.lt.f) .and. (stepfactor.gt. 0.0001)) then
c         do 75 i = 1,p
c         do 80 j = 1,p
c            x(i,j) = oldx(i,j)
c   80    continue
c   75    continue
         call copy(oldx,x)
         f = oldf
         stepfactor = stepfactor/2
         write    (*,*) 'm2drtni: -- stepfctr= ',stepfactor
       else
         if (stepfactor.gt.0.0001) then
           stepfactor=stepfactor*1.25
           write    (*,*) 'm2drtni: ++ stepfactor= ',stepfactor
         else
           stepfactor=10
         end if
        end if
      if ((irept.lt.iend).and.((zeros).eq..false.)
     *       .and.(dabs(f).gt.eps)) goto 30
      write  (*,*) 'irept =',irept
      write (*,*) 'zeros = ',zeros
      write (*,*) ' f =  ',f
      end
      subroutine m2fct (x,f,derf,coef,p)
       integer ii,jj,m,n,isiz,p
```

```
      double precision sum1(16,16),sum2(16,16),dexpo(16,16)
      double precision g2(16,16,16,16),derf(16,16)
      double precision x(16,16),coef(16,16) ,fac
      common g2
      double precision, siz,f
      siz = 16.0
      isiz =16
c      do 10 ii=1,isiz
c        do 20 jj=1,isiz
c         dexpo(ii,jj) =0.0
c         do  30 mm=1,p
c         do 40  nn=1,p
c           dexpo(ii,jj)= dexpo(ii,jj)+x(mm,nn)*g2(ii,jj,mm,nn)
c 40      continue
c 30      continue
c         dexpo(ii,jj)= dexp(dexpo(ii,jj))
c 20     continue
c 10   continue
        call afilter(x,p)
        call cmtdct(x,dexpo,2)
        fac = coef (1,1)/siz
        do 11 ii=1,isiz
        do 12 jj=1,isiz
          dexpo(ii,jj)= fac*dexp(dexpo(ii,jj))
 12     continue
 11     continue
c       write (*,*) 'finished dexp loop'
      f=0.0
c      do 100 m=1,p
c      do 110 n=1,p
c      sum1(m,n) = 0.0
c      sum2(m,n) = 0.0
c        do 120 i=1,isiz
c        do 130 j=1,isiz
c          sum1(m,n)=sum1(m,n)+  g2(i,j,m,n) *dexpo(i,j)
c          sum2(m,n)=sum2(m,n)+g2(i,j,m,n)**2*dexpo(i,j)
c 130    continue
c 120    continue
c 110  continue
c 100  continue
c     take dct of dexpo put into sum1
c      unneeded sum1's will go unused
      call cmtdct(dexpo,sum1,1)
      call cmtdct(dexpo,sum2,3)
      do 200 m=1,p
      do 210 n=1,p
c      if (((n.eq.1).and.(m.eq.1)).eq..false. ) then
        f=f+( sum1(m,n)-coef(m,n))**2
        derf(m,n)=sum2(m,n)*2*(sum1(m,n) - coef(m,n))
c      end if
```

```
  210  continue
  200  continue
       end
c
c
      subroutine redar (ireconi)
      integer*2 ireconi(512,512)
      integer  i,j
      character chr*20
      chr = 'inp'
199   format(a20)
      open(15, file=chr ,status='old')
      rewind 15
      do 100 i=1,512
       do 200 j=1,512
                read (15,*) ireconi(i,j)
 200   continue
c       write (*,*) 'redar: i= ',i
 100  continue
      close (15)
      return
      end
c
      subroutine printa (ireconi,chr)
      integer*2 ireconi(512,512)
      integer  i,j
      character chr*20
199   format(a20)
      open(15, file=chr ,status='new')
      do 100 i=1,512
       do 200 j=1,512
                write (15,*) ireconi(i,j)
 200   continue
c       write (*,*) 'printa: i= ',i,' chr= ',chr
 100  continue
      close (15)
      return
      end
c
c
      subroutine ascale(dcta,rela,thres,l3)
      double precision dcta(16,16),rela(16,16)
      double precision thres(16,16),f,sum
      character l3*80,t3*45,t4*11
      call sumbl(rela,sum)
      f= dcta(1,1)*16/sum
c      write (*,*) 'ascale: factor = ',f
      call multbl(rela,f,thres)
c      t3= 'rel ent w/ 1 -pass.       normalized '
c      t4= '               factor = '
```

```
c       write (13,47) t3,t4,f
c 47    format (a41,a22,f8.4)
      return
      end
c
      subroutine shift(dcta,rela,thres)
      integer i,j
      double precision dcta(16,16),rela(16,16)
      double precision thres(16,16),f,sum
      call sumbl (rela,sum)
      f= (dcta(1,1)*16.0-sum ) /16.0/16.0
      do 10 i=1,16
       do 20 j=1,16
         thres(i,j) =rela(i,j)+f
 20    continue
 10   continue
      return
      end
c
      subroutine sumbl(rela,sum)
      double precision rela(0:15,0:15),sum
      integer i,j
      sum = 0.0
      do 12 i=0,15
       do 22 j=0,15
        sum  =rela(i,j)+ sum
 22    continue
 12   continue
      return
      end
c
      subroutine multbl(rela,f,thres)
      double precision rela(0:15,0:15),thres(0:15,0:15),f
      integer i,j
      do 10 i=0,15
       do 20 j=0,15
         thres(i,j) =rela(i,j)*f
 20    continue
 10   continue
      return
      end
c
c
      subroutine afilter(orga,j)
      double precision orga(0:15,0:15)
      integer j,l,n
        do 120 n=0,15
         do 140 l=j,15
           orga(n,l)= 0
 140       continue
```

```
 120        continue
          do 35 n=0,15
           do 45 l=j,15
             orga(l,n)= 0
 45         continue
 35        continue
        end
c
         subroutine cmtdct(in,out,q)
         double precision in(0:15,0:15),out(0:15,0:15),temp(0:15,0:15)
         double precision x(0:15),y(0:15)
         integer i,q,j
c         write (*,*) 'cmtdct'
         do 10 i=0,15
            do 20 j=0,15
               x(j)=in(i,j)
 20        continue
          if (q.eq.1) then
            call mtdct(x,y)
          end if
          if (q.eq.2) then
            call mtidct(x,y)
          end if
          if (q.eq.3) then
            call mtsqdct(x,y)
          end if
          do 25 j=0,15
             temp(i,j)=y(j)
 25         continue
 10        continue
c
c         write    (*,*) 'cmtdct temp(3,7) = ',temp(3,7)
         do 40 i= 0,15
           do 50 j=0,15
            x(j)= temp(j,i)
 50        continue
c           call mtdct(x,y)
          if (q.eq.1) then
            call mtdct(x,y)
          end if
          if (q.eq.2) then
            call mtidct(x,y)
          end if
          if (q.eq.3) then
            call mtsqdct(x,y)
          end if
           do 60 j=0,15
              out(j,i)=y(j)
 60        continue
 40      continue
```

```
c        write    (*,*) 'cmtdct out(3,7) = ',out(3,7)
c
       return
       end
c
       subroutine mtdct(x,y)
        integer i,k
          double precision x(0:15),y(0:15)
          double precision  g2(16,16,16,16), g1(0:15,0:15)
          common g2,g1
           do 20 k=0,15
              y(k)=0.0
              do 10 i=0,15
                  y(k)= y(k) + x(i)*g1(i,k)
 10       continue
 20       continue
c        write    (*,*) 'mtdct y(3) = ',y(3)
           return
          end
        subroutine mtidct(x,y)
         integer i,k
          double precision x(16),y(16)
          double precision g2(16,16,16,16), g1(16,16)
          common g2,g1
           do 20 i=1,16
              y(i) = 0.0
              do 10 k=1,16
                  y(i) = y(i) +x(k)*g1(i,k)
 10       continue
 20       continue
          return
          end
c
        subroutine mtsqdct(x,y)
         integer i,k
          double precision x(16),y(16)
          double precision g2(16,16,16,16), g1(16,16), gsq(16,16)
          common g2,g1,gsq
           do 20 i=1,16
              y(i) = 0.0
              do 10 k=1,16
                  y(i) = y(i) +x(k)*gsq(k,i)
 10       continue
 20       continue
          return
          end
c
      subroutine set(orga,r)
c     sets orga = r
      double precision orga(16,16),r
```

```
      integer i,j
      do 10 i=1,16
         do 20 j=1,16
           orga(i,j) = r
 20      continue
 10   continue
       return
       end
      subroutine copy(ina,outa)
      double precision ina(16,16),outa(16,16)
      integer i,j
      do 10 i=1,16
      do 20 j=1,16
         outa(i,j) = ina(i,j)
 20   continue
 10   continue
      return
      end
c      subroutine xpobl(rela,thres)
c      double precision rela(0:15,0:15),thres(0:15,0:15)
c      integer i,j
c      do 10 i=0,15
c       do 20 j=0,15
c         thres(i,j) = dexp(rela(i,j))
c 20    continue
c 10   continue
c      return
c      end
```

The following program performs 512x512 picture reconstruction with MREP and IDCT.

```
c        program main
c         Sun 7./12  removed extra writes
c                    made equ in terms of 16  for substitution
c                    chnaged big loop (thru blocks) to
c                    be a function of block size
c                    changed output filenames to
c                         r1p  r2p  r3p
c        It will take the 2d-dct w/o using 1d-dct's.
c        The idct is done in a similar manner.
c        the non-integer variables are : double precision
c                  the simultaneous will converge to
c                  1/10**3 in lieu of 1/10*3
c                  the iterative will comverge to
c                  1/10**3 in lieu of 1/10*3
```

```
c
c
c        su11m0s6    mrep not used sim to 10**-6
c        su11m0s20    mrep not used sim to 10**-20
c               wk
c          1110     remove -x  used x
c          1100     speed drtni
c          1050 wk
c        su1050     sim to 1/10**12
c        su850      removed some extra writes
c          845 wk
c        su845      the simultaneous will converge to
c                   1/10**7 in lieu of 1/10*3
c                   the iterative will comverge to
c                   1/10**7 in lieu of 1/10*3
c          650 wk
c        su650      1-10 passes thru coef
c          640 wk
c        su640      correctly labels the output files (no of iter)
c        su630 wk   is fastest   , the mrep uses previous output as
c                   prior
c        s500 wk
c        shifts simultaneous satifaction output
c        uses coef 1x1-16x16
c
c
c
c
c
c
c
c
c
c
c
c
      double precision g(16,16,16,16) ,g1(16,16),gsq(16,16)
      common g ,g1,gsq
      call defineg
      call defg1sq
      call menu
      stop
      end
c
      subroutine defineg
      integer i,j,m,n
      double precision pie,y ,s,c,d
      double precision g(16,16,16,16)
      common g
      y= -1.0
```

```
      s = 16.0
      pie = dacos(y)
      do 10 i=1,16
        do 20 j=1,16
           do 30 m=1,16
              if (m.eq.1) then
                 c=1
                else
                  c= 2.0**0.5
                end if
             do 40 n=1,16
              if (n.eq.1) then
                 d=1.0
                else
                  d= 2.0**0.5
                end if
              g(i,j,m,n)=c*d/s*dcos(((2*i-1)*(m-1)*pie)/(2*s))*
     1                           dcos(((2*j-1)*(n-1)*pie)/(2*s))
  40         continue
  30       continue
  20    continue
  10  continue
      end
c
      subroutine defg1sq
      integer i,m
      double precision g(16,16,16,16), g1(16,16),gsq(16,16), pie,c,y
      double precision s,sq
      common g, g1 ,gsq
      s = 16.0
      sq = s**0.5
      y = -1.0
      pie = dacos(y)
      do 10 m = 1 ,16
        if (m.eq.1) then
                 c = 1.0
               else
                  c= 2.0**0.5
                    end if
        do 20 i = 1,16
             g1(i,m) = c/sq*dcos ((2.0*i-1.0)*(m-1.0)*pie/2.0/s)
             gsq(i,m)= g1(i,m)**2.0
c             write    (*,*) 'dfg1: ', g1(i,m)
  20    continue
  10  continue
      end
c
c      character tpl1*80,tpl2*80,tpl3*80,tpl4*80,tpl5*80,tpout20*20
c      double precision tp(0:15,0:15)
c
```

```
c       double precision x
c       character rl1*80,rl2*80,rl3*80,rl4*80,rl5*80
c       character tl1*80,tl2*80,tl3*80,tl4*80,tl5*80
c       character ol1*80,ol2*80,ol3*80,ol4*80,ol5*80
c
      subroutine menu
      integer c,fifteen,iter,i,j,vblockno,hblockno
c       c = number of non-zero dct coefficients
      integer*2 bigorg(0:511,0:511)
      integer*2 bigrel1(0:511,0:511)
      integer*2 bigrel2(0:511,0:511),bigrel3(0:511,0:511)
      integer retain
      double precision dcta(0:15,0:15),orga(0:15,0:15),idcta(0:15,0:15)
      double precision rela(0:15,0:15),temp(0:15,0:15),sima(0:15,0:15)
      character name*20
      read (*,*) retain
c       retain =3
      call redar(bigorg)
c     loop thru ea 16x16 block
      do 10 vblockno=0,(512/16)-1
      do 11 hblockno=0,(512/16)-1
      do 20 j=0,15
      do 30 i=0,15
       orga(i,j) = float(bigorg(i+vblockno*16,j+hblockno*16))
c       write (*,*) 'orga= ',orga(i,j)
   30 continue
   20 continue
      call cmtdct(orga,dcta,1)

      iter =1
      call mrereco(retain,dcta,rela,iter)
      do 80 j=0,15
      do 90 i=0,15
       bigrel1(i+vblockno*16,j+hblockno*16)= int(rela(i,j)+0.5)
   90 continue
   80 continue
      iter =2
      call mrereco(retain,dcta,rela,iter)
      do 81 j=0,15
      do 91 i=0,15
       bigrel2(i+vblockno*16,j+hblockno*16)= int(rela(i,j)+0.5)
   91 continue
   81 continue
      iter =3
      call mrereco(retain,dcta,rela,iter)
      do 82 j=0,15
      do 92 i=0,15
       bigrel3(i+vblockno*16,j+hblockno*16)= int(rela(i,j)+0.5)
   92 continue
   82 continue
```

```
c        write (*,*)  'did a block !!!  ',hblockno,'vbk= ',vblockno
  11  continue
c        write (*,*) ' Fortran: vblockno= ', vblockno,' ************'
  10  continue
      name = 'r1'
      call printa(bigrel1,name)
      name = 'r2'
      call printa(bigrel2,name)
      name = 'r3'
      call printa(bigrel3,name)
      end
c
c
c
      subroutine redar (ireconi)
      integer*2 ireconi(512,512)
      integer  i,j
      character chr*20
      chr = 'inp'
 199  format(a20)
      open(14, file=chr ,status='old')
      rewind 14
      do 100 i=1,512
       do 200 j=1,512
                read (14,*) ireconi(i,j)
 200   continue
 100  continue
      close (14)
      return
      end
c
      subroutine printa (ireconi,chr)
      integer*2 ireconi(512,512)
      integer  i,j
      character chr*20
c         199   format(1x,i3)
      open(14, file=chr ,status='new')
      do 100 i=1,512
       do 200 j=1,512
                write (14,*) ireconi(i,j)
 200   continue
 100  continue
      close (14)
      return
      end
c
c
      subroutine ascale(dcta,rela,thres,l3)
      double precision dcta(16,16),rela(16,16)
      double precision thres(16,16),f,sum
```

```
      character l3*80,t3*45,t4*11
      call sumbl(rela,sum)
      f= dcta(1,1)*16/sum
      call multbl(rela,f,thres)
      return
      end
c
      subroutine sumbl(rela,sum)
      double precision rela(0:15,0:15),sum
      integer i,j
      sum = 0.0
      do 12 i=0,15
       do 22 j=0,15
        sum  =rela(i,j)+ sum
 22    continue
 12   continue
      return
      end
c
      subroutine multbl(rela,f,thres)
      double precision rela(0:15,0:15),thres(0:15,0:15),f
      integer i,j
      do 10 i=0,15
       do 20 j=0,15
         thres(i,j) =rela(i,j)*f
 20    continue
 10   continue
      return
      end
c
       subroutine cmtdct(in,out,q)
       double precision in(0:15,0:15),out(0:15,0:15),temp(0:15,0:15)
       double precision x(0:15),y(0:15)
       integer i,q,j
c       write (*,*) 'cmtdct'
       do 10 i=0,15
          do 20 j=0,15
             x(j)=in(i,j)
 20       continue
        if (q.eq.1) then
          call mtdct(x,y)
        end if
        if (q.eq.2) then
          call mtidct(x,y)
        end if
        if (q.eq.3) then
          call mtsqdct(x,y)
        end if
        do 25 j=0,15
           temp(i,j)=y(j)
```

```
 25        continue
 10       continue
c
c        write    (*,*) 'cmtdct temp(3,7) = ',temp(3,7)
        do 40 i= 0,15
          do 50 j=0,15
           x(j)= temp(j,i)
 50       continue
         if (q.eq.1) then
           call mtdct(x,y)
         end if
         if (q.eq.2) then
           call mtidct(x,y)
         end if
         if (q.eq.3) then
           call mtsqdct(x,y)
         end if
          do 60 j=0,15
             out(j,i)=y(j)
 60       continue
 40     continue
c        write    (*,*) 'cmtdct out(3,7) = ',out(3,7)
c
        return
        end
c
        subroutine mtdct(x,y)
         integer i,k
           double precision x(0:15),y(0:15)
           double precision  g2(16,16,16,16), g1(0:15,0:15)
           common g2,g1
            do 20 k=0,15
               y(k)=0.0
               do 10 i=0,15
                   y(k)= y(k) + x(i)*g1(i,k)
  10       continue
  20       continue
c         write    (*,*) 'mtdct y(3) = ',y(3)
            return
           end
         subroutine mtidct(x,y)
          integer i,k
           double precision x(16),y(16)
           double precision g2(16,16,16,16), g1(16,16)
           common g2,g1
            do 20 i=1,16
               y(i) = 0.0
               do 10 k=1,16
                   y(i) = y(i) +x(k)*g1(i,k)
  10       continue
```

```
20      continue
        return
        end
c
       subroutine mtsqdct(x,y)
        integer i,k
        double precision x(16),y(16)
        double precision g2(16,16,16,16), g1(16,16), gsq(16,16)
        common g2,g1,gsq
         do 20 i=1,16
            y(i) = 0.0
            do 10 k=1,16
                y(i) = y(i) +x(k)*gsq(k,i)
10      continue
20      continue
        return
        end
c
      subroutine mrereco(p,y,old,iter)
c
c     p= no of coef to reconstruct with
c     y= input array of coef.
c     reconsu= output recon array
c     n= ibl= image block length
c
c     procedure for image reconstruction using iterative minimum
c     cross entropy me
c
      double precision f,derf,x,xst,eps,s,y(16,16)
      integer p,k,n,i,ii,iend,iter
      double precision ascin(16,16),old(16,16),g(16,16,16,16)
      common g
      character ch80*80
      n=16
      xst=0.0
      iend=500
c      eps=0.001
      eps = 1.0/10.0**6.0
c      write (*,*) 'reconstructing with ',p,' coef'
      k=1
      s=float(n)
c
       if (iter.eq.1) then
         fac = y(1,1)/s
         call   set(old,fac)
       end if
c         do 12 ii=1,iter
      do 14 mm=1,p
c          write (*,*) ' next row  mm= ',mm
         do 89 nn=1,p
```

```
          if ( ( not( mm.eq.1) ).or. ( not (nn.eq.1)) ) then
            call drtni(x,f,derf,xst,eps,iend,ier,mm,nn,old,y,n)
            if (ier.ne.0) print*,'ier=',ier
            do 35 i=1,n
                do 45 j=1,n
                 ascin(i,j)= old(i,j)*dexp(x*g(i,j,mm,nn))
 45             continue
 35         continue
             call ascale(y,ascin,old,ch80)
           end if
 89      continue
 14   continue
 12      continue
c
c        write (*,*) 'ne mre fini: old(1,1) =  ',old(1,1)
      return
      end
c
c     subroutine   drtni
c
c     purpose
c         to solve general nonlinear equations of the form  f(x)=0
c         by means of newton-s iteration method.
c
c     usage
c         call drtni (x,f,derf,fct,xst,eps,iend,ier,mm,nn)
c         parameter fct requires an external statement.
c
c
c     description of parameters
c         x   -double precision resultant root of equ f(x)=0
c         f   -double precision resultant function value at root x
c         derf-double precision resultant value of derivative :aa root x
c         fct -name of the external subroutine used. it computes
c              to given argument x function value f and derivative
c              derf. its parameter list must include x,f,derf, where
c              all parameters are double precision.
c         xst -double precision input value which specifies the
c              intitail guess of the root x.
c         eps -single precision input value which specifies the
c              upper bound of the error of result x.
c         iend-max. nu. of iteration steps specified.
c         ier -resultant error parameter coded as follows
c                 ier=0 - no error
c                 ier=1 - no convergence after iend iteration steps
c                 ier=2 - at any iteration step derivative derf was
c                         equal to zero.
c
c     remarks
c         the procedure is bypassed and gives the error message ier =2
```

```
c          if at any iteration step derivative of f(x) is equal to 0.
c          possibly the procedure would be successful if it is started
c          once more with another intial guess xst.
c
c      subroutines and function subprograms required
c          the external subroutine  fct(x,f,derf,?,?) must be furnished
c          by the user.
c
c      method
c          solution of equation f(x)=0 is done by means of newton-s
c          iteration method, which starts at the initial guess xst of
c          a root x. convergence is quadratic if the derivative of
c          f(x) at root x is not equal to zero. one iteration step
c          requires one evalution of f(x) and one evaluation of the
c          derivative of f(x). for test on satisfactory accuracy see
c          formula (2) of mathematical description.
c          for reference, see r. zurmuehl, praktische mathematik fuer
c          ingenieure und physiker, springer, berlin/goettingen/
c          heidelberg, 1963, pp. 12-17.
c
c
      subroutine drtni(x,f,derf, xst,eps,iend,ier,mm,nn,sb,y,n)
c
      double precision x,f,derf,xst,eps,dx
c      double precision b(430,16,16), y(16,16),sb(16,16)
      double precision  y(16,16),sb(16,16)
      integer i,n,iend,ier,mm,nn
c
c      prepare iteration
      ier =0
      x=xst
      call fct(x, f,derf,mm,nn,sb,y,n)
c
c      start iteration loop
      i = 0
      if (f) 1,7,1
c      equation is not satisfied by x
    1 if (derf) 2,8,2
c
c      iteration is possible
    2 x=x+f/derf
      i= i+1
c       x=x-dx
      call fct(x,f,derf,mm,nn,sb,y,n)
c      test on satisficatory accuracy
      if (i.gt.iend)  goto 20
      if (dabs(f).gt.eps)  goto  1
c      end of iteration loop
      return
c
```

```
c     no convergence after iend iteration steps. error return.
   20 continue
      write (*,*) 'drtni: failed to converge in ',i,' steps'
      ier=1
    7 return
c
c     error return in case of zero divisor
    8 ier=2
      return
      end
c
      subroutine fct(x,f,derf,mm,nn,sb,y,n)
      double precision x,f,derf,sum1,sum2
      double precision sb(16,16), y(16,16),g(16,16,16,16)
      double precision new1(16,16),sb(16,16)
      common g
      integer n
      sum1=0.0
      sum2=0.0
      do 10 i=1,n
          do 20 j=1,n
           new1(i,j)= sb(i,j)*dexp(x*g(i,j,mm,nn))*g(i,j,mm,nn)
           sum1=sum1+new1(i,j)
           sum2=sum2+new1(i,j)*g(i,j,mm,nn)
  20      continue
  10  continue
      f=sum1-y(mm,nn)
      derf=-sum2
      return
      end
c
      subroutine set(orga,r)
c     sets orga = r
      double precision orga(16,16),r
      integer i,j
      do 10 i=1,16
         do 20 j=1,16
           orga(i,j) = r
  20     continue
  10  continue
       return
       end
```

The following program works with binary images performing the MREP, MEP, and IDCT reconstructions. It was used to generate Appendix F.

```
c         the non-integer variables are : double precision
c                   the simultaneous will converge to
c                   1/10**6
c                   the iterative will comverge to
c                   1/10**6
c
c
c
      double precision g(16,16,16,16) ,g1(16,16),gsq(16,16)
      common g ,g1,gsq
      call defineg
      call defg1sq
      call menu
      stop
      end
c
      subroutine defineg
      integer i,j,m,n
      double precision pie,y ,s,c,d
      double precision g(16,16,16,16)
      common g
      y= -1.0
      s = 16.0
      pie = dacos(y)
      do 10 i=1,16
        do 20 j=1,16
           do 30 m=1,16
              if (m.eq.1) then
                 c=1
                else
                 c= 2.0**0.5
                end if
             do 40 n=1,16
              if (n.eq.1) then
                 d=1.0
                else
                 d= 2.0**0.5
                end if
              g(i,j,m,n)=c*d/s*dcos(((2*i-1)*(m-1)*pie)/(2*s))*
     1                        dcos(((2*j-1)*(n-1)*pie)/(2*s))
  40         continue
  30       continue
  20    continue
  10  continue
      end
c
      subroutine defg1sq
      integer i,m
      double precision g(16,16,16,16), g1(16,16),gsq(16,16), pie,c,y
```

```
      common g, g1 ,gsq
      y = -1.0
      pie = dacos(y)
      do 10 m = 1 ,16
        if (m.eq.1) then
                 c = 1.0
               else
                  c= 2.0**0.5
                    end if
        do 20 i = 1,16
             g1(i,m) = c/4.0*dcos ((2.0*i-1.0)*(m-1.0)*pie/32.0)
             gsq(i,m)= g1(i,m)**2.0
c             write    (*,*) 'dfg1: ', g1(i,m)
 20     continue
 10   continue
      end
c
      subroutine menu
      integer c,k,fifteen,iter
c      c = number of non-zero dct coefficients
      double precision dcta(0:15,0:15),orga(0:15,0:15),idcta(0:15,0:15)
      double precision rela(0:15,0:15),temp(0:15,0:15),thres(0:15,0:15)
      double precision mepa(0:15,0:15)
      character tpl1*80,tpl2*80,tpl3*80,tpl4*80,tpl5*80,tpout20*20
      double precision tp(0:15,0:15)
      double precision x
      character rl1*80,rl2*80,rl3*80,rl4*80,rl5*80
      character tl1*80,tl2*80,tl3*80,tl4*80,tl5*80
      character ol1*80,ol2*80,ol3*80,ol4*80,ol5*80
      character dl1*80,dl2*80,dl3*80,dl4*80,dl5*80
      character il1*80,il2*80,il3*80,il4*80,il5*80
c      character tempil3*80
      character str0*3,str1*3,str2*3,out20*20
c      do 946 incc=(incb+3),(incb+3)
c
      fifteen = 15
      c= 16
c      write    (*,*) ' what do you want with the arrrays'
         call redar(orga,ol2,ol3)
      call cmtdct(orga,dcta,1)
c
c     controls the number of pixels used in reconstruction
c
c
      il5=' METHOD     MSE          RE1'
      rl5='           R2'
      write (*,288)  il5, rl5
 288  format (a30,a30)

      do  129 inc=0,14
```

```
      c= 16 -inc
c
      call afilter (dcta,c)
      call cmtdct(dcta,idcta,2)
      call mse(idcta,orga,i15)
      write (*,245) '   idct/org ',i15
 245  format (a12,a62)
c
        call d2mm(c,dcta,mepa)
        call mse(mepa,orga,t15)
      write (*,245) '   mep/org  ',t15
        call mse(mepa,idcta,t15)
      write (*,245) '   mep/idct ',t15
          m=16
          do 283, iter = 1,10
          call mrereco(c,dcta,rela,m,iter)
           call ascale(dcta,rela,thres,r13)
           call copy(thres,rela)
      call mse (rela,orga,r15)
      write (*,853) ' rel ',iter,' /org',r15
      call mse (rela,idcta,r15)
      write (*,853) ' rel ',iter,' /idct ',r15
      call mse (rela,mepa,r15)
      write (*,853) ' rel ',iter,' /mep  ',r15
c      write (r13,397) 'iterative min relative entropy: ',iter,' passes'
c      397  format (a34,i2,a8)
c       282   format (a1,i2)
 853   format(a5,i2,a5,a60)
 283   continue
c
c       end iterative loop    begin simulateous
c        call label (str0,str1,n1,str2,n12,out20)
 129  continue
c
c
      end
c
c
c
c
      subroutine d2mm(p,coef,outa)
      double precision outa(1:16,1:16)
c
      double precision coef(16,16)
      integer p
c
      integer i,siz
      integer ier,iend
      double precision prior,xst,eps,rsiz
      double precision dexpo(16,16),x(16,16)
```

```
      double precision g2(16,16,16,16)
      common g2
      logical zeros
      siz = 16
      rsiz = 16.0
c       if your going to subroutine ascale
      prior = coef(1,1)/rsiz
c      write (*,*) 'd2m: cof(11) ',coef(1,1),' prior ',prior,' siz',siz
      eps = 1.0/10.0**9.0
      iend =500
      xst =0.0
      call m2drtni(x,xst,eps,iend,ier,coef,zeros,p)
      if (zeros) then
         write (*,*) ' ********* zero slope **'
      end if
c      do 10 ii=1,siz
c      do 20 jj=1,siz
c           dexpo(ii,jj)= 0
c           do 30 mm=1,  p
c           do 40 nn=1,  p
c             if (((mm.eq.1).and.(nn.eq.1)).eq..false.)then
c                dexpo(ii,jj)=dexpo(ii,jj)+x(mm,nn)*g2(ii,jj,mm,nn)
c             end if
c 40        continue
c 30        continue
c           dexpo(ii,jj) = dexp(dexpo(ii,jj))
c 20   continue
c 10   continue
c      x(1,1) = 0.0
      call afilter(x,p)
      call cmtdct(x,dexpo,2)
      do 50 i=1,siz
      do 60 j=1,siz
c           write (*,*) '284  prior = ',prior,' dexp= ', dexpo(i,j)
        outa(i,j) = prior*dexp(dexpo(i,j))
 60   continue
 50   continue
      call  shift(coef,outa,dexpo)
      call       copy (dexpo,outa)
      end
c
      subroutine m2drtni(x,xst,eps,iend,ier,coef,zeros,p)
      double precision x(16,16),xst,eps,coef(16,16)
      integer iend,ier,p
      logical zeros
c
      integer m,n,irept
      double precision  stepfactor,oldf,f,derfsum ,stepsize
      double precision  oldx(16,16), derf(16,16)
      isiz =16
```

```
      siz =16.0
      ier= 0
      oldf=10000
      stepfactor = 2.0
       call set(x,xst)
c        x(1,1)=0
       call m2fct(x,f,derf,coef,p)
      irept=0
 30   continue
      zeros= .true.
      derfsum=0.0
      do 40 m=1,p
      do 50 n = 1,p
c       if (((m.eq.1).and.(n.eq.1)).eq..false.) then
        derfsum=derfsum+(derf(m,n))**2
c       end if
 50   continue
 40   continue
      derfsum= derfsum**0.5
      stepsize = f/derfsum*stepfactor
      do  60 m=1 , p
      do 70  n =  1 , p
c       if (((m.eq.1).and.(n.eq.1)).eq..false.) then
        oldx(m,n) = x(m,n)
        if ((derf(m,n).eq.0).eq..false.) then
              zeros=.false.
              x(m,n)=x(m,n)-stepsize*derf(m,n)/derfsum
        end if
c       end if
  70  continue
  60  continue
      oldf=f
      call m2fct(x,f,derf,coef,p)
      if ((oldf.lt.f) .and. (stepfactor.gt. 0.0001)) then
c         do 75 i = 1,p
c         do 80 j = 1,p
c            x(i,j) = oldx(i,j)
c   80    continue
c   75    continue
         call copy(oldx,x)
         f = oldf
         stepfactor = stepfactor/2
c         write    (*,*) 'm2drtni: -- stepfctr= ',stepfactor
       else
         if (stepfactor.gt.0.0001) then
           stepfactor=stepfactor*1.25
c            write    (*,*) 'm2drtni: ++ stepfactor= ',stepfactor
         else
           stepfactor=10
         end if
```

```
         end if
       if ((irept.lt.iend).and.((zeros).eq..false.)
      *      .and.(dabs(f).gt.eps)) goto 30
c       write  (*,*) 'irept =',irept
c       write (*,*) 'zeros = ',zeros
c       write (*,*) ' f =  ',f
       end
       subroutine m2fct (x,f,derf,coef,p)
        integer ii,jj,m,n,isiz,p
        double precision sum1(16,16),sum2(16,16),dexpo(16,16)
        double precision g2(16,16,16,16),derf(16,16)
        double precision x(16,16),coef(16,16) ,fac
        common g2
        double precision, siz,f
        siz = 16.0
        isiz =16
c        do 10 ii=1,isiz
c          do 20 jj=1,isiz
c           dexpo(ii,jj) =0.0
c           do  30 mm=1,p
c           do 40  nn=1,p
c             dexpo(ii,jj)= dexpo(ii,jj)+x(mm,nn)*g2(ii,jj,mm,nn)
c  40       continue
c  30       continue
c           dexpo(ii,jj)= dexp(dexpo(ii,jj))
c  20      continue
c  10    continue
          call afilter(x,p)
          call cmtdct(x,dexpo,2)
          fac = coef (1,1)/siz
          do 11 ii=1,isiz
          do 12 jj=1,isiz
            dexpo(ii,jj)= fac*dexp(dexpo(ii,jj))
  12      continue
  11      continue
c         write (*,*) 'finished dexp loop'
        f=0.0
c        do 100 m=1,p
c        do 110 n=1,p
c        sum1(m,n) = 0.0
c        sum2(m,n) = 0.0
c          do 120 i=1,isiz
c          do 130 j=1,isiz
c            sum1(m,n)=sum1(m,n)+  g2(i,j,m,n) *dexpo(i,j)
c            sum2(m,n)=sum2(m,n)+g2(i,j,m,n)**2*dexpo(i,j)
c  130     continue
c  120     continue
c  110   continue
c  100   continue
c       take dct of dexpo put into sum1
```

```
c        unneeded sum1's will go unused
        call cmtdct(dexpo,sum1,1)
        call cmtdct(dexpo,sum2,3)
        do 200 m=1,p
        do 210 n=1,p
c        if (((n.eq.1).and.(m.eq.1)).eq..false. ) then
          f=f+( sum1(m,n)-coef(m,n))**2
          derf(m,n)=sum2(m,n)*2*(sum1(m,n) - coef(m,n))
c        end if
  210   continue
  200   continue
        end
c
c
c
c
       subroutine redar (ireconi,l2,l3)
       double precision ireconi(16, 16)
       integer  i,j
       character chr*20,l2*80,l3*80
        chr = 'inp'
199    format(a20)
       open(15, file=chr ,status='old')
        rewind 15
       do 100 i=1,16
        do 200 j=1,16
                 read (15,*) ireconi(i,j)
 200    continue
 100   continue
c        l1 = ' '
       close (15)
       return
       end
c
       subroutine mse(orga,temp,l5)
       character l5*80
       double precision orga(16,16 ),temp(16,16)
       integer i,j
       double precision x,y,z
       x=0.0
       do 10 i=1,16
        do 20 j=1,16
         x=x + (orga(i,j)-temp(i,j))**2
 20     continue
 10    continue
       y=x**0.5
        call relent (orga,temp,x)
        call relent (temp,orga,z)
       write (l5,47) '  ',y,'     ',x,'     ',z
 47    format (1x,a2,f12.8,a5,f12.8,a5,f12.8)
```

```
      end
c
      subroutine ascale(dcta,rela,thres,l3)
      double precision dcta(16,16),rela(16,16)
      double precision thres(16,16),f,sum
      character l3*80,t3*45,t4*11
      call sumbl(rela,sum)
      f= dcta(1,1)*16/sum
c      write (*,*) 'ascale: factor = ',f
      call multbl(rela,f,thres)
      t3= 'rel ent w/ 1 -pass.         normalized '
      t4= '              factor = '
      write (l3,47) t3,t4,f
 47   format (a41,a22,f8.4)
      return
      end
c
      subroutine shift(dcta,rela,thres)
      integer i,j
      double precision dcta(16,16),rela(16,16)
      double precision thres(16,16),f,sum
      call sumbl (rela,sum)
      f= (dcta(1,1)*16.0-sum ) /256.0
      do 10 i=1,16
       do 20 j=1,16
         thres(i,j) =rela(i,j)+f
 20    continue
 10   continue
      return
      end
c
      subroutine sumbl(rela,sum)
      double precision rela(0:15,0:15),sum
      integer i,j
      sum = 0.0
      do 12 i=0,15
       do 22 j=0,15
        sum  =rela(i,j)+ sum
 22    continue
 12   continue
      return
      end
c
      subroutine multbl(rela,f,thres)
      double precision rela(0:15,0:15),thres(0:15,0:15),f
      integer i,j
      do 10 i=0,15
       do 20 j=0,15
         thres(i,j) =rela(i,j)*f
 20    continue
```

```
 10   continue
      return
      end
      subroutine athreshold(orga,temp,l3,x)
      character l3*80
      double precision orga(0:15, 0:15),temp(0:15, 0:15)
      integer i,j
      double precision x
      write (l3,47) 'after thresholding with ',x
 47   format (a24,f8.4)
      do 10 i=0,15
         do 20 j=0,15
            if (orga(i,j).ge.x) then
               temp(i,j)=2.0
             else
               temp(i,j)=1.0
             end if
 20      continue
 10   continue
      end
c
c
      subroutine afileout(orga,l1,l2,l3,l4,l5,c,k,ch8,
     *   tp,tpl1,tpl2,tpl3,tpl4,tpl5,a,b,tpout20)
      double precision orga(0:15,0:15)
      integer i,j,k,c,a,b
      character ch8*20,l1*80,l2*80,l3*80,l4*80,l5*80,ch(0:15)*1
      character tpl1*80,tpl2*80,tpl3*80,tpl4*80,tpl5*80,tpout20*20
      double precision tp(0:15,0:15)
c      write (*,*) 'afilout: opening ch8= ',ch8,' k= ',k
c      write (*,*) 'l3 = ',l3
      open ( 12,file=ch8, status='new')
c      write (*,*) '1= fast, 2=one digit,3=reg 4=2dig,5=x-'
      do 110 i=0,15
       if (k.eq.1) then
           write (12,20) (orga(i,j),j=0,15)
  20       format (1x,16f3.0)
       end if
       if (k.eq.2) then
           write (12,120) (orga(i,j),j=0,15)
  120      format (1x,16f5.0)
       end if
       if (k.eq.3) then
           write (12,25) (orga(i,j),j=0,15)
   25      format (1x,16f5.2)
       end if
      if (k.eq.4) then
           write (12,121) (orga(i,j),j=0,15)
  121      format (1x,16f4.1)
      end  if
```

```
  110 continue
      if (k.eq.5) then
      do 112 i=0,15
         do 132 j=0,15
              if (orga(i,j).eq.0 ) then
                   ch(j)=' '
                end if
              if (orga(i,j).eq.1 ) then
                   ch(j)='-'
                end if
              if (orga(i,j).eq.2 ) then
                   ch(j)='x'
                end if
  132       continue
            write (12,122) (ch(j),j=0,15)
  122       format (1x,16a1)
  112    continue
      end   if
      write (12,*) ' '
      write (12,*) l1
      write (12,*) l2
      write (12,*) l3
      write (12,*) l4
      if (k.eq.5)      write (12,911) l5
 911   format (1x,1a17)
      if ( (k.eq.5).eq..false. )      write (12,*) l5
      write (12,*)
      write (12,*) 'an n*n coefficient matrix was used with n= ',c
c
c        if k=5 print out the next one
c
      if (k.eq.5) then
      do 512 i=0,15
         do 532 j=0,15
              if (orga(i,j).lt.0.5 ) then
                   ch(j)=' '
                end if
              if (tp(i,j).eq.1 ) then
                   ch(j)='-'
                end if
              if (tp(i,j).eq.2 ) then
                   ch(j)='x'
                end if
  532       continue
            write (12,522) (ch(j),j=0,15)
  522       format (1x,16a1)
  512    continue
      write (12,*) ' '
      write (12,*) tpl1
      write (12,*) tpl2
```

```
      write (12,*) tpl3
      write (12,*) tpl4
      if (k.eq.5)       write (12,511) tpl5
 511   format (1x,1a17)
      if ((k.eq.5).eq..false. )      write (12,*) tpl5
      write (12,*)
      write (12,*) 'an n*n coefficient matrix was used with n= ',c
      close (12)
      end if
      end
c
c
      subroutine afilter(orga,j)
      double precision orga(0:15,0:15)
      integer j,l,n
        do 120 n=0,15
         do 140 l=j,15
           orga(n,l)= 0
 140       continue
 120      continue
        do 35 n=0,15
         do 45 l=j,15
           orga(l,n)= 0
 45       continue
 35      continue
      end
c
       subroutine cmtdct(in,out,q)
       double precision in(0:15,0:15),out(0:15,0:15),temp(0:15,0:15)
       double precision x(0:15),y(0:15)
       integer i,q,j
c       write (*,*) 'cmtdct'
       do 10 i=0,15
          do 20 j=0,15
             x(j)=in(i,j)
 20        continue
        if (q.eq.1) then
          call mtdct(x,y)
        end if
        if (q.eq.2) then
          call mtidct(x,y)
        end if
        if (q.eq.3) then
          call mtsqdct(x,y)
        end if
        do 25 j=0,15
           temp(i,j)=y(j)
 25       continue
 10      continue
c
```

```
c        write    (*,*) 'cmtdct temp(3,7) = ',temp(3,7)
       do 40 i= 0,15
         do 50 j=0,15
          x(j)= temp(j,i)
 50      continue
c         call mtdct(x,y)
        if (q.eq.1) then
          call mtdct(x,y)
        end if
        if (q.eq.2) then
          call mtidct(x,y)
        end if
        if (q.eq.3) then
          call mtsqdct(x,y)
        end if
         do 60 j=0,15
            out(j,i)=y(j)
 60      continue
 40    continue
c       write    (*,*) 'cmtdct out(3,7) = ',out(3,7)
c
       return
       end
c
       subroutine mtdct(x,y)
        integer i,k
          double precision x(0:15),y(0:15)
          double precision  g2(16,16,16,16), g1(0:15,0:15)
          common g2,g1
           do 20 k=0,15
              y(k)=0.0
              do 10 i=0,15
                  y(k)= y(k) + x(i)*g1(i,k)
  10      continue
  20      continue
c         write    (*,*) 'mtdct y(3) = ',y(3)
           return
          end
        subroutine mtidct(x,y)
         integer i,k
          double precision x(16),y(16)
          double precision g2(16,16,16,16), g1(16,16)
          common g2,g1
           do 20 i=1,16
              y(i) = 0.0
              do 10 k=1,16
                  y(i) = y(i) +x(k)*g1(i,k)
  10      continue
  20      continue
          return
```

```
          end
c
        subroutine mtsqdct(x,y)
         integer i,k
          double precision x(16),y(16)
          double precision g2(16,16,16,16), g1(16,16), gsq(16,16)
          common g2,g1,gsq
           do 20 i=1,16
              y(i) = 0.0
              do 10 k=1,16
                  y(i) = y(i) +x(k)*gsq(k,i)
 10       continue
 20       continue
          return
          end
c
c
          subroutine relent (in1,in2,out)
c
c         finds relative entropy between 2  blocks
c
          double precision in1(16,16), in2(16,16),out
          integer i,j
c
         out=0.0
          do 10 i=1,16
             do 20 j=1,16
             if(in1(i,j).lt.0.00001) then
               print *,in1(i,j)
                else
                out=out+in1(i,j)*log(in1(i,j)/in2(i,j))
                end if
 20          continue
 10       continue
c
          return
          end
c
c     from file ddrtni.ftn
c
      subroutine mrereco(p,y,old,n,iter)
c
c     p= no of coef to reconstruct with
c     sub= input array of coef.
c     reconsu= output recon array
c     n= ibl= image block length (either 4 or 8)
c
c     procedure for image reconstruction using iterative minimum
c     cross entropy me
c
```

```
      double precision f,derf,x,xst,eps,s,y(16,16)
      integer p,k,n,i,ii,iend,iter
      double precision ascin(16,16),old(16,16),g(16,16,16,16)
      common g
      character ch80*80
c      write (*,*)  'p= ',p
      xst=0.0
      iend=500
      eps = 1.0/10.0**6.0
      pie=3.141592654
c      write (*,*) 'reconstructing with ',p,' coef'
      k=1
      s=float(n)
c
       if (iter.eq.1) then
         fac = y(1,1)/s
         call   set(old,fac)
       end if
c         do 12 ii=1,iter
      do 15 mm=1,p
c         write (*,*) ' next row  mm= ',mm
          do 16 nn=1,p
            if ( ( not( mm.eq.1) ).or. ( not (nn.eq.1)) ) then
              call drtni(x,f,derf,xst,eps,iend,ier,mm,nn,old,y,n)
c              write (*,*) 'nn= ',nn,' x= ',x
              if (ier.ne.0) print*,'ier=',ier
              do 35 i=1,n
                  do 45 j=1,n
                   ascin(i,j)= old(i,j)*dexp(x*g(i,j,mm,nn))
 45               continue
 35           continue
               call ascale(y,ascin,old,ch80)
             end if
 16       continue
 15   continue
12       continue
c
c       write (*,*) 'ne mre fini: old(1,1) =  ',old(1,1)
      return
      end
c
c     subroutine   drtni
c
c     purpose
c         to solve general nonlinear equations of the form  f(x)=0
c         by means of newton-s iteration method.
c
c     usage
c         call drtni (x,f,derf,fct,xst,eps,iend,ier,mm,nn)
c         parameter fct requires an external statement.
```

```
c
c
c     description of parameters
c         x   -double precision resultant root of equ f(x)=0
c         f   -double precision resultant function value at root x
c         derf-double precision resultant value of derivative :aa root x
c         fct -name of the external subroutine used. it computes
c              to given argument x function value f and derivative
c              derf. its parameter list must include x,f,derf, where
c              all parameters are double precision.
c         xst -double precision input value which specifies the
c              intitail guess of the root x.
c         eps -single precision input value which specifies the
c              upper bound of the error of result x.
c         iend-max. nu. of iteration steps specified.
c         ier -resultant error parameter coded as follows
c                 ier=0 - no error
c                 ier=1 - no convergence after iend iteration steps
c                 ier=2 - at any iteration step derivative derf was
c                         equal to zero.
c
c     remarks
c         the procedure is bypassed and gives the error message ier =2
c         if at any iteration step derivative of f(x) is equal to 0.
c         possibly the procedure would be successful if it is started
c         once more with another intial guess xst.
c
c     subroutines and function subprograms required
c         the external subroutine  fct(x,f,derf,?,?) must be furnished
c         by the user.
c
c     method
c         solution of equation f(x)=0 is done by means of newton-s
c         iteration method, which starts at the initial guess xst of
c         a root x. convergence is quadratic if the derivative of
c         f(x) at root x is not equal to zero. one iteration step
c         requires one evalution of f(x) and one evaluation of the
c         derivative of f(x). for test on satisfactory accuracy see
c         formula (2) of mathematical description.
c         for reference, see r. zurmuehl, praktische mathematik fuer
c         ingenieure und physiker, springer, berlin/goettingen/
c         heidelberg, 1963, pp. 12-17.
c
c
      subroutine drtni(x,f,derf, xst,eps,iend,ier,mm,nn,sb,y,n)
c
      double precision x,f,derf,xst,eps,dx
c      double precision b(430,16,16), y(16,16),sb(16,16)
      double precision  y(16,16),sb(16,16)
      integer i,n,iend,ier,mm,nn
```

```
c
c     prepare iteration
      ier =0
      x=xst
      call fct(x, f,derf,mm,nn,sb,y,n)
c
c     start iteration loop
      i = 0
      if (f) 1,7,1
c     equation is not satisfied by x
    1 if (derf) 2,8,2
c
c     iteration is possible
    2 x=x+f/derf
      i= i+1
c      x=x-dx
      call fct(x,f,derf,mm,nn,sb,y,n)
c     test on satisficatory accuracy
      if (i.gt.iend)  goto 20
      if (dabs(f).gt.eps)  goto  1
c     end of iteration loop
      return
c
c     no convergence after iend iteration steps. error return.
   20 continue
      write (*,*) 'drtni: failed to converge in ',i,' steps'
      ier=1
    7 return
c
c     error return in case of zero divisor
    8 ier=2
      return
      end
c
      subroutine fct(x,f,derf,mm,nn,sb,y,n)
      double precision x,f,derf,sum1,sum2
      double precision sb(16,16), y(16,16),g(16,16,16,16)
      double precision new1(16,16),sb(16,16)
      common g
      integer n
      sum1=0.0
      sum2=0.0
      do 10 i=1,n
          do 20 j=1,n
           new1(i,j)= sb(i,j)*dexp(x*g(i,j,mm,nn))*g(i,j,mm,nn)
           sum1=sum1+new1(i,j)
           sum2=sum2+new1(i,j)*g(i,j,mm,nn)
 20       continue
 10   continue
      f=sum1-y(mm,nn)
```

```
      derf=-sum2
      return
      end
c
      subroutine label(str3n0,str3n1,n1,str3n2,n2,out20)
      character  str3n0*3,str3n1*3,str3n2*3,out20*20
      integer n1
      write (out20,47)  'd',str3n0,'_',str3n1,'_',n1,'_',str3n2,'_',n2
 47   format (a1,a3,a1,a3,a1,i2,a1,a3,a1,i4)
      return
      end
c
      subroutine combine(st1,st2,out)
      character  st1*20,st2*20,out*40
      write (out,47)  st1,st2
 47   format (a20,a20)
      return
      end
c
      subroutine drawrect(orga,x1,y1,x2,y2,r)
c
c     makes a x1,y1 to x2,y2 rect of orga = to r
c
      double precision orga(0:15,0:15),r
      integer i,j,x1,y1,x2,y2
      do 10 i=x1,x2
       do 20 j=y1,y2
          orga(i,j)=r
 20    continue
 10   continue
      return
      end
c
      subroutine set(orga,r)
c     sets orga = r
      double precision orga(16,16),r
      integer i,j
      do 10 i=1,16
         do 20 j=1,16
           orga(i,j) = r
  20     continue
  10  continue
       return
       end
      subroutine copy(ina,outa)
      double precision ina(16,16),outa(16,16)
      integer i,j
      do 10 i=1,16
      do 20 j=1,16
         outa(i,j) = ina(i,j)
```

```
   20  continue
   10  continue
       return
       end
c       subroutine xpobl(rela,thres)
c       double precision rela(0:15,0:15),thres(0:15,0:15)
c       integer i,j
c       do 10 i=0,15
c        do 20 j=0,15
c          thres(i,j) = dexp(rela(i,j))
c 20     continue
c 10    continue
c       return
c       end
```

APPENDIX C

BINARY TEST IMAGES AND THE GOLDEN GATE BRIDGE

<u>Binary Test Images</u>

Image A5:

```
----------------
----X-----------
---XXX----------
--XX-XX---------
-XX---XX--XXXXX-
-XX---XX--XXXXX-
-XXXXXXX--XX----
-XXXXXXX--XX----
-XX---XX--XXXX--
-XX---XX--XXXXX-
-------------XX-
-------------XX-
------------XX--
-----------XXX--
-----------XX---
----------------
```

Image E:

```
----------------
----------------
---XXXXXXXXX----
---XXXXXXXXX----
---XXXXXXXXX----
---XXX----------
---XXX----------
---XXXXXXXXX----
---XXXXXXXXX----
---XXX----------
---XXX----------
---XXXXXXXXX----
---XXXXXXXXX----
---XXXXXXXXX----
----------------
----------------
```

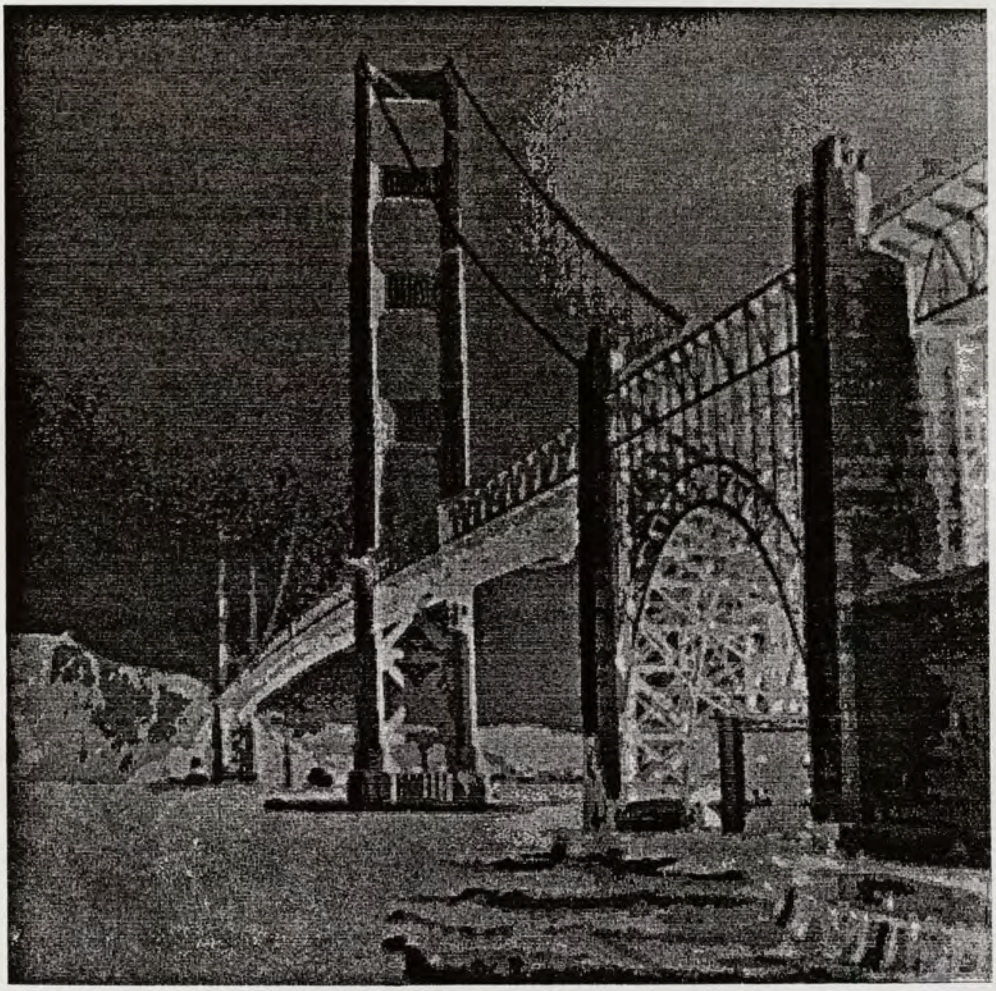
Image E0:

```
----------------
----------XXXXX-
----------XXXXX-
----------XX----
----------XX----
----------XXXXX-
----X-----XXXXX-
---XXX----XX----
--XX-XX---XX----
--X---X---XXXXX-
-XX---XX--XXXXX-
--X---X---------
--XX-XX---------
---XXX----------
----X-----------
----------------
```

Image 0:

```
----------------
----------------
----------------
----------------
----------------
----------------
----------------
----------XX----
--------XXXXXX--
-------XX----XX-
-------X------X-
-------X------X-
-------XX----XX-
--------XXXXXX--
----------XX----
----------------
```

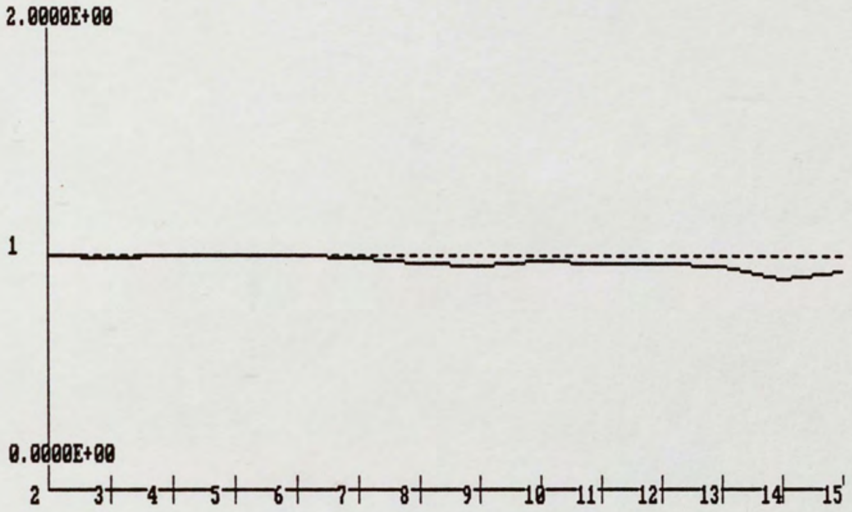
Picture of the Golden Gate Bridge



APPENDIX D

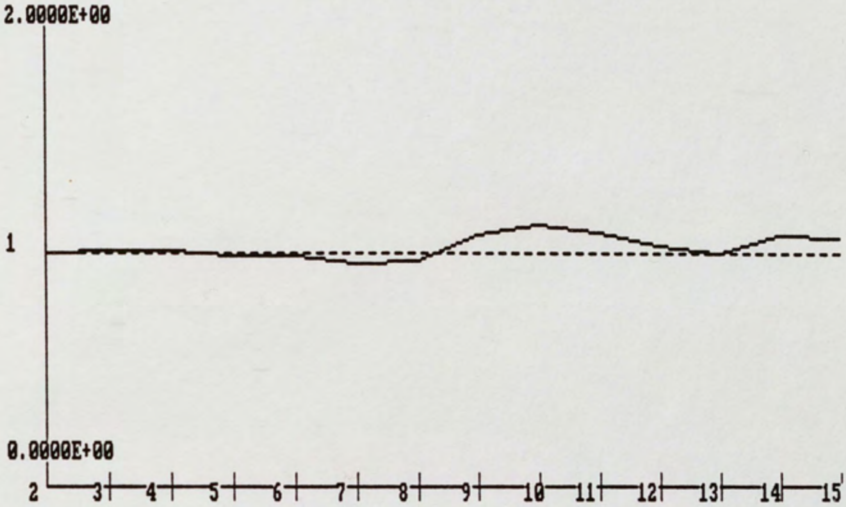
GRAPHS AND TABLES OF ERROR BETWEEN RECONSTRUCTION AND ORIGINAL FOR MEP AND IDCT FOR ALL 4 BINARY IMAGES

Note: The "NewO" is simply the "O". The convergence limit is the limit of the error as the Newton non-linear solution method converges to zero.



MSE
Ratio of MEP mse / IDCT mse for image: A5
X-value is number of coeffficients retained

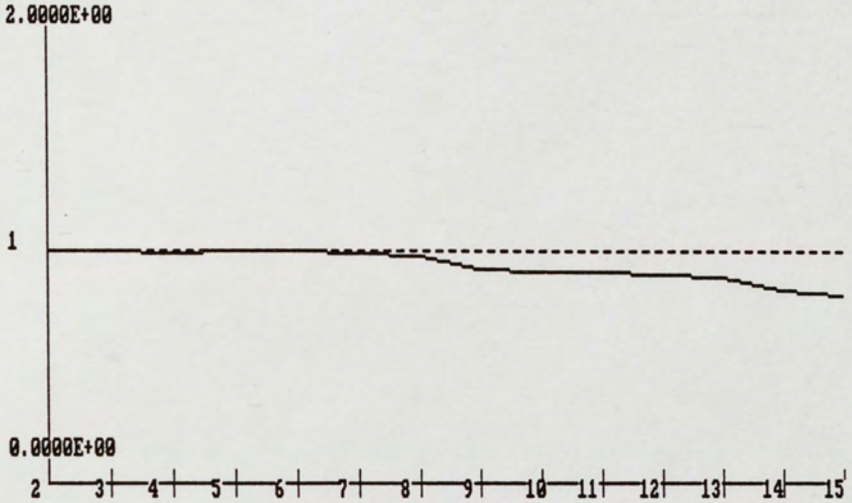
Convergence Limit: 1/10**3
Variables: double precision



MSE

Ratio of MEP mse / IDCT mse for image: E
X value is the number of coefficients retained

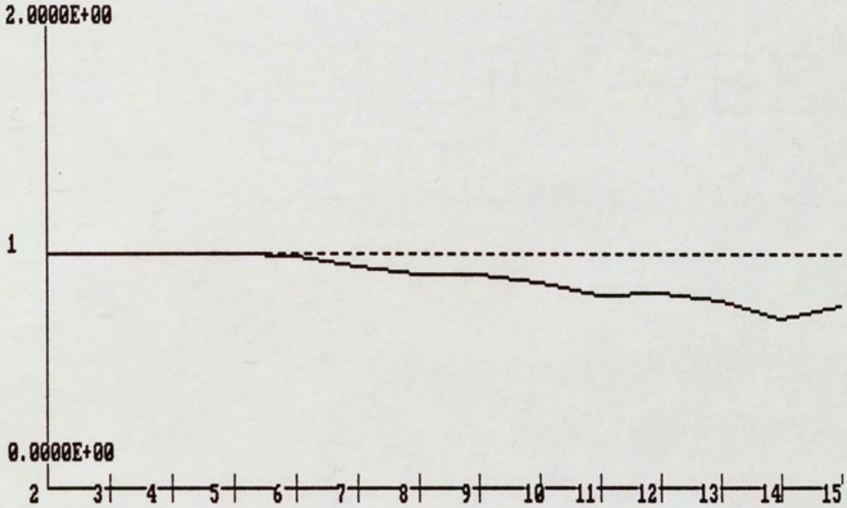
Convergence Limit: 1/10**3
Variables: double precision



MSE

Ratio of MEP mse / IDCT mse for image: EO
X value is the number of coefficients retained

Convergence Limit: 1/10**3
Variables: double precision



MSE

Ratio or MEP mse / IDCT mse for image: New O
X value is the number of coefficients retained

Convergence Limit: 1/10**3
Variables: double precision

the non-integer variables are : double precision
the MEP will converge to
1/10**6

this includes a shift to normalize greyness
level after reconstruction.

Image used:

```
----------------
----x-----------
---xxx----------
--xx-xx---------
-xx---xx--xxxxx-
-xx---xx--xxxxx-
-xxxxxxx--xx----
-xxxxxxx--xx----
-xx---xx--xxxx--
-xx---xx--xxxxx-
-------------xx-
-------------xx-
------------xx--
-----------xxx--
-----------xx---
----------------
```

Number
of Coef.
Retained
NxN

image: A5

N	Method	MSE	RE: recn x org	org x recn
2	MEP	7.03829980	17.55254011	16.69795912
2	idct	7.02090915	17.46160240	16.61680268
3	MEP	6.32082376	13.97969985	13.87475769
3	idct	6.36202443	14.20157969	14.10041438
4	MEP	6.10283319	12.84424271	12.96620230
4	idct	6.10544076	12.88445234	13.01326134
5	MEP	5.74751512	11.31845586	11.62258414
5	idct	5.76460163	11.42050742	11.74303887
6	MEP	5.53994606	10.47380389	10.88036800
6	idct	5.56642803	10.61233813	11.02811602
7	MEP	5.14235868	9.06846993	9.45839005
7	idct	5.21114184	9.38685482	9.81343355
8	MEP	3.56399912	4.45828612	4.49391813
8	idct	3.68085725	4.97073488	5.08986580
9	MEP	3.04125694	3.25320072	3.29729913

Number
of Coef.
Retained
NxN

image: A5

N	Method	MSE	RE: recn x org	org x recn
9	idct	3.17117118	3.79406342	3.87543085
10	MEP	2.58887596	2.31295558	2.32669551
10	idct	2.66266656	2.78152339	2.77142393
11	MEP	2.08976767	1.51229971	1.53419167
11	idct	2.17505877	1.87922527	1.89026210
12	MEP	1.68077699	0.98254266	0.98713372
12	idct	1.74112547	1.18964096	1.18847816
13	MEP	1.47311918	0.76261123	0.76289828
13	idct	1.53459202	0.92121287	0.91844357
14	MEP	1.19701624	0.54595815	0.54613127
14	idct	1.33268600	0.71199815	0.71072539
15	MEP	0.81654311	0.24653073	0.24660839
15	idct	0.87922835	0.29691391	0.29694794
16	MEP	0.00060147	0.00000011	0.00000011
16	idct	0.00000651	0.00000000	0.00000000

the non-integer variables are : double precision
the MEP will converge to
1/10**6

this includes a shift to normalize greyness
level after reconstruction.

Image used:

```
----------------
----------------
---xxxxxxxxx----
---xxxxxxxxx----
---xxxxxxxxx----
---xxx----------
---xxx----------
---xxxxxxxxx----
---xxxxxxxxx----
---xxx----------
---xxx----------
---xxxxxxxxx----
---xxxxxxxxx----
---xxxxxxxxx----
----------------
----------------
```

(@egin table header
Number
of Coef. image: E
Retained
NxN

N	Method	MSE	RE: recn x org	org x recn
2	MEP	7.42647217	19.41535038	18.75045136
2	idct	7.41426495	19.34781578	18.68809214
3	MEP	6.24295589	13.17777392	13.58772884
3	idct	6.19361329	13.01751550	13.45375296
4	MEP	6.12206731	12.62029543	12.91067104
4	idct	6.02158157	12.22882584	12.53840961
5	MEP	5.08777726	8.91046625	9.12265575
5	idct	5.13099455	9.13802366	9.40274754
6	MEP	4.95422327	8.49376116	8.71753859
6	idct	5.03315263	8.85517599	9.14595402
7	MEP	3.52056939	4.40051993	4.44646187
7	idct	3.68530018	5.07184480	5.18721664
8	MEP	3.46991543	4.25643046	4.26109111
8	idct	3.59124497	4.77561921	4.82475678
9	MEP	2.75246215	2.37331856	2.38960168

9	idct	2.54684581	2.18686439	2.21766550
10	MEP	2.65265636	2.10727950	2.12197931
10	idct	2.35564268	1.84137701	1.85665896
11	MEP	2.38662019	1.78299641	1.79715849
11	idct	2.18544627	1.56552610	1.57714640
12	MEP	2.23384062	1.65980487	1.67251826
12	idct	2.16548059	1.60193204	1.61282084
13	MEP	1.79555222	1.10852040	1.11308786
13	idct	1.80048554	1.15481731	1.15841623
14	MEP	1.68070209	0.90047981	0.90600541
14	idct	1.55860038	0.78998025	0.79487563
15	MEP	0.88111794	0.24609731	0.24621556
15	idct	0.82383574	0.23621649	0.23623482
16	MEP	0.00068268	0.00000015	0.00000015
16	idct	0.00000566	0.00000000	0.00000000

the non-integer variables are : double precision
the MEP will converge to
1/10**6

this includes a shift to normalize greyness
level after reconstruction.

Image used:

```
----------------
----------xxxxx-
----------xxxxx-
----------xx----
----------xx----
----------xxxxx-
----x-----xxxxx-
---xxx----xx----
--xx-xx---xx----
--x---x---xxxxx-
-xx---xx--xxxxx-
--x---x---------
--xx-xx---------
---xxx----------
----x-----------
----------------
```

Number
of Coef.
Retained
NxN

image: EO

N	Method	MSE	RE: recn x org	org x recn
2	MEP	6.53362235	15.18025775	14.33138122
2	idct	6.51773649	15.11009328	14.27189510
3	MEP	6.26661886	13.82616926	13.39508712
3	idct	6.26824483	13.84590611	13.41555343
4	MEP	5.99221136	12.62895147	12.41981804
4	idct	6.03331887	12.83409744	12.61611859
5	MEP	5.46855199	10.29525531	10.40675203
5	idct	5.46868138	10.34249012	10.46485075
6	MEP	5.26218285	9.50412345	9.69595757
6	idct	5.27142147	9.57383575	9.78700400
7	MEP	4.83970506	8.08462452	8.23188319
7	idct	4.88013297	8.28491321	8.48429076
8	MEP	4.32590819	6.56474977	6.70376881
8	idct	4.44467868	7.00792825	7.20123445
9	MEP	3.05199053	3.39905102	3.50567841

Number
of Coef.
Retained
NxN

image: EO

N	Method	MSE	RE: recn x org	org x recn
9	idct	3.29917863	4.15522499	4.27066004
10	MEP	2.59946370	2.50060912	2.58259891
10	idct	2.86602679	3.21878528	3.31853367
11	MEP	2.19884963	1.78938969	1.83514347
11	idct	2.42849988	2.35734738	2.41158263
12	MEP	1.78998806	1.19386106	1.21747120
12	idct	1.99396027	1.62459983	1.65523208
13	MEP	1.64603270	1.02621158	1.03881964
13	idct	1.85723033	1.44881061	1.45326012
14	MEP	1.42568738	0.82996977	0.84293201
14	idct	1.71120174	1.25510071	1.26439450
15	MEP	1.07012494	0.48566309	0.48892441
15	idct	1.32209837	0.75993542	0.76331346
16	MEP	0.00071602	0.00000016	0.00000016
16	idct	0.00000634	0.00000000	0.00000000

the non-integer variables are : double precision
the MEP will converge to
1/10**6

this includes a shift to normalize greyness
level after reconstruction.

Image used:

```
----------------
----------------
----------------
----------------
----------------
----------------
----------------
----------xx----
--------xxxxxx--
-------xx----xx-
-------x------x-
-------x------x-
-------xx----xx-
--------xxxxxx--
----------xx----
----------------
```

Number
of Coef.
Retained
NxN

image: NewO

N	Method	MSE	RE: recn x org	org x recn
2	MEP	4.61570972	7.68271919	7.02134043
2	idct	4.59666398	7.61426789	6.96593476
3	MEP	4.45878877	7.15044939	6.68926407
3	idct	4.47877368	7.23118480	6.75682492
4	MEP	4.31022707	6.56972896	6.31289914
4	idct	4.31262126	6.58345621	6.32421455
5	MEP	4.11673428	5.93538239	5.83127486
5	idct	4.12675073	5.99244636	5.88027436
6	MEP	4.04312391	5.78573834	5.68628590
6	idct	4.09590764	5.96288290	5.85228764
7	MEP	3.21032272	3.71663728	3.82409522
7	idct	3.39773091	4.22096877	4.32565759
8	MEP	2.75799320	2.83530838	2.92334164
8	idct	3.02608006	3.49304366	3.57760520
9	MEP	2.75670635	2.83436082	2.92274056

Number
of Coef.
Retained
NxN

image: NewO

N	Method	MSE	RE: recn x org	org x recn
9	idct	3.02608006	3.49304366	3.57760520
10	MEP	2.28969895	2.03126447	2.12097045
10	idct	2.62445554	2.68400454	2.80450370
11	MEP	1.65508925	1.13067732	1.15712482
11	idct	2.00816666	1.70314425	1.73972000
12	MEP	1.64865002	1.11332245	1.13856183
12	idct	1.98458733	1.65447229	1.68694463
13	MEP	1.35157099	0.79083811	0.79716533
13	idct	1.70321000	1.27702900	1.28587638
14	MEP	0.54470436	0.14302950	0.14287747
14	idct	0.75788227	0.27840163	0.27760304
15	MEP	0.39325096	0.06899315	0.06900538
15	idct	0.50762974	0.11626281	0.11617604
16	MEP	0.00043888	0.00000006	0.00000006
16	idct	0.00000471	0.00000000	0.00000000

APPENDIX E

MEASUREMENT OF THE DISTANCES BETWEEN RECONSTRUCTIONS FOR THE IMAGE A5

rel 3 refers to the 3-pass MREP

RE1 of IDCT/ORG means $\quad \mathrm{IDCT}(i) \quad \ln \dfrac{\mathrm{IDCT}(i)}{\mathrm{ORG}(i)} \; i$

RE2 of IDCT/ORG = RE1 of ORG/IDCT

Image: A5
Convergence limit = $1/10^9$ for MEP and $1/10^6$ for MREP

METHOD	MSE	RE1	RE2
IDCT/ORG	0.00000651	0.00000000	0.00000000
MEP/ORG	0.00001473	0.00000000	0.00000000
MEP/IDCT	0.00001890	0.00000000	0.00000000
rel 1/ORG	1.03763134	0.40061775	0.39760002
rel 1/IDC	1.03763340	0.40061888	0.39760118
rel 1/MEP	1.03762859	0.40061571	0.39759796
rel 2/ORG	0.21942138	0.01650705	0.01652786
rel 2/IDC	0.21942515	0.01650760	0.01652840
rel 2/MEP	0.21941921	0.01650676	0.01652757
rel 3/ORG	0.05070269	0.00110973	0.00111136
rel 3/IDC	0.05070276	0.00110974	0.00111137
rel 3/MEP	0.05070291	0.00110975	0.00111138
rel 4/ORG	0.01269510	0.00006108	0.00006109
rel 4/IDC	0.01269267	0.00006106	0.00006107
rel 4/MEP	0.01269635	0.00006109	0.00006110
rel 5/ORG	0.00275554	0.00000322	0.00000322
rel 5/IDC	0.00275322	0.00000322	0.00000322
rel 5/MEP	0.00275694	0.00000322	0.00000322
rel 6/ORG	0.00077066	0.00000026	0.00000026
rel 6/IDC	0.00077059	0.00000026	0.00000026
rel 6/MEP	0.00077166	0.00000026	0.00000026
rel 7/ORG	0.00021295	0.00000002	0.00000002
rel 7/IDC	0.00021484	0.00000002	0.00000002
rel 7/MEP	0.00021306	0.00000002	0.00000002

rel 8/ORG	0.00005453	0.00000000	0.00000000
rel 8/IDC	0.00005676	0.00000000	0.00000000
rel 8/MEP	0.00005519	0.00005519	0.00000000
rel 9/ORG	0.00001588	0.00000000	0.00000000
rel 9/IDC	0.00001725	0.00000000	0.00000000
rel 9/MEP	0.00002082	0.00000000	0.00000000
rel 10/ORG	0.00000453	0.00000000	0.00000000
rel 10/IDC	0.00000708	0.00000000	0.00000000
rel 10/MEP	0.00001543	0.00000000	0.00000000
IDCT/ORG	0.87922835	0.29694794	0.29691391
MEP/ORG	0.81651727	0.24660878	0.24652897
MEP/IDCT	0.12979615	0.00764820	0.00764471
rel	0.12979615	0.00764820	0.00764471
rel 1/ORG	1.29840862	0.62333789	0.61757791
rel 1/IDC	1.01651895	0.38111109	0.37869482
rel 1/MEP	1.00803508	0.37376299	0.37104740
rel 2/ORG	0.83825212	0.26097953	0.26116281
rel 2/IDC	0.24181764	0.02222242	0.02227911
rel 2/MEP	0.20328875	0.01461039	0.01463393
rel 3/ORG	0.81701481	0.81701481	0.24753373
rel 3/IDC	0.13952738	0.00866355	0.00865944
rel 3/MEP	0.04867307	0.00101334	0.00101474
rel 4/ORG	0.81672295	0.24665722	0.24657988
rel 4/IDC	0.13033496	0.00770181	0.00769566
rel 4/MEP	0.01162809	0.00005095	0.00005096
rel 5/ORG	0.81659099	0.24661375	0.24653157
rel 5/IDC	0.12978652	0.00765132	0.00764737
rel 5/MEP	0.00247682	0.00000265	0.00000265
rel 6/ORG	0.81652090	0.24660935	0.24652913
rel 6/IDC	0.12979264	0.00764836	0.00764493
rel 6/MEP	0.00069369	0.00000021	0.00000021
rel 7/ORG	0.81651299	0.24660868	0.24652894
rel 7/IDC	0.12979684	0.00764818	0.00764474
rel 7/MEP	0.00018128	0.00000001	0.00000001
rel 8/ORG	0.81651531	0.24660867	0.24652892
rel 8/IDC	0.12979650	0.00764820	0.00764473
rel 8/MEP	0.00004700	0.00000000	0.00000000
rel 9/ORG	0.81651629	0.24660867	0.24652892
rel 9/IDC	0.12979614	0.00764821	0.00764473
rel 9/MEP	0.00002052	0.00000000	0.00000000
rel 10/ORG	0.81651636	0.24660867	0.24652892
rel 10/IDC	0.12979609	0.00764821	0.00764473
rel 10/MEP	0.00001668	0.00000000	0.00000000
IDCT/ORG	1.33268600	0.71072539	0.71199815
MEP/ORG	1.19699510	0.54613265	0.54595441
MEP/IDCT	0.24030941	0.02568248	0.02561445
rel 1/ORG	1.50905440	0.88073178	0.86845742
rel 1/IDC	0.96548334	0.34856069	0.34811867
rel 1/MEP	0.93350907	0.32362576	0.32250158

rel 2/ORG	1.20093295	0.55540888	0.55695840
rel 2/IDC	0.30603676	0.03654588	0.03661904
rel 2/MEP	0.17563390	0.01099327	0.01100470
rel 3/ORG	1.19685483	0.54637599	0.54664116
rel 3/IDC	0.24470610	0.02635486	0.02630138
rel 3/MEP	0.04087351	0.00068639	0.00068697
rel 4/ORG	1.19733956	0.54613499	0.54598154
rel 4/IDC	0.24013321	0.02571239	0.02564173
rel 4/MEP	0.00847850	0.00002726	0.00002726
rel 5/ORG	1.19708184	0.54614332	0.54595535
rel 5/IDC	0.24022140	0.02568437	0.02561555
rel 5/MEP	0.00156517	0.00000107	0.00000107
rel 6/ORG	1.19699543	0.54613527	0.54595435
rel 6/IDC	0.24030696	0.02568261	0.02561455
rel 6/MEP	0.00039963	0.00000007	0.00000007
rel 7/ORG	1.19698913	0.54613221	0.54595428
rel 7/IDC	0.24031355	0.02568246	0.02561449
rel 7/MEP	0.00009486	0.00000000	0.00000000
rel 8/ORG	1.19699191	0.54613193	0.54595428
rel 8/IDC	0.24031095	0.02568248	0.02561448
rel 8/MEP	0.00003041	0.00000000	0.00000000
rel 9/ORG	1.19699286	0.54613200	0.54595428
rel 9/IDC	0.24031000	0.02568249	0.02561448
rel 9/MEP	0.00002462	0.00000000	0.00000000
rel 10/ORG	1.19699292	0.54613203	0.54595428
rel 10/IDC	0.24030993	0.02568249	0.02561448
rel 10/MEP	0.00002432	0.00000000	0.00000000
IDCT/ORG	1.53459202	0.91844357	0.92121287
MEP/ORG	1.47308789	0.76289587	0.76260831
MEP/IDCT	0.35111643	0.04900938	0.04870938
rel 1/ORG	1.72610107	1.08547402	1.06771285
rel 1/IDC	0.98131218	0.35442341	0.35381503
rel 1/MEP	0.90953643	0.30585226	0.30510443
rel 2/ORG	1.47641425	0.77114591	0.77265722
rel 2/IDC	0.39866770	0.05887913	0.05875872
rel 2/MEP	0.16918666	0.01004538	0.01004942
rel 3/ORG	1.47243826	0.76284795	0.76322586
rel 3/IDC	0.35391766	0.04959869	0.04932697
rel 3/MEP	0.03936991	0.03936991	0.00061719
rel 4/ORG	1.47327085	0.76284723	0.76263223
rel 4/IDC	0.35096703	0.04903331	0.04873332
rel 4/MEP	0.00797966	0.00002394	0.00002394
rel 5/ORG	1.47314812	0.76290439	0.76260915
rel 5/IDC	0.35105470	0.04901089	0.04871026
rel 5/MEP	0.00140899	0.00000086	0.00000086
rel 6/ORG	1.47309075	0.76290036	0.76260834
rel 6/IDC	0.35111287	0.04900952	0.04870945
rel 6/MEP	0.00035122	0.00000005	0.00000005
rel 7/ORG	1.47308471	0.76289604	0.76260829

rel 7/IDC	0.35111946	0.04900934	0.04870940
rel 7/MEP	0.00007689	0.00000000	0.00000000
rel 8/ORG	1.47308599	0.76289530	0.76260829
rel 8/IDC	0.35111825	0.04900934	0.04870940
rel 8/MEP	0.00002210	0.00000000	0.00000000
rel 9/ORG	1.47308655	0.76289533	0.76260829
rel 9/IDC	0.35111765	0.04900934	0.04870940
rel 9/MEP	0.00001461	0.00000000	0.00000000
rel 10/ORG	1.47308660	0.76289537	0.76260829
rel 10/IDC	0.35111758	0.04900934	0.04870940
rel 10/MEP	0.00001364	0.00000000	0.00000000
IDCT/ORG	1.74112547	1.18847816	1.18964096
MEP/ORG	1.68074670	0.98712890	0.98253943
MEP/IDCT	0.41612845	0.07038479	0.06972954
rel 1/ORG	1.89243231	1.29438568	1.26428790
rel 1/IDC	0.97438525	0.35216393	0.35147928
rel 1/MEP	0.87159773	0.28189370	0.28174653
rel 2/ORG	1.68236810	0.99407223	0.99071977
rel 2/IDC	0.45012960	0.07827378	0.07791045
rel 2/MEP	0.15180280	0.00817620	0.00818103
rel 3/ORG	1.68008965	0.98694000	0.98297481
rel 3/IDC	0.41751437	0.07080120	0.07016514
rel 3/MEP	0.03331223	0.00043546	0.00043564
rel 4/ORG	1.68090569	0.98704267	0.98255458
rel 4/IDC	0.41601567	0.07040393	0.06974488
rel 4/MEP	0.00626106	0.00001527	0.00001527
rel 5/ORG	1.68078905	0.98712709	0.98253979
rel 5/IDC	0.41610882	0.07038575	0.06973011
rel 5/MEP	0.00107057	0.00000048	0.00000048
rel 6/ORG	1.68074902	0.98713231	0.98253934
rel 6/IDC	0.41612847	0.07038473	0.06972966
rel 6/MEP	0.00023986	0.00000002	0.00000002
rel 7/ORG	1.68074399	0.98712915	0.98253932
rel 7/IDC	0.41612933	0.07038475	0.06972963
rel 7/MEP	0.00005672	0.00000000	0.00000000
rel 8/ORG	1.68074476	0.98712825	0.98253931
rel 8/IDC	0.41612874	0.07038479	0.06972963
rel 8/MEP	0.00002471	0.00000000	0.00000000
rel 9/ORG	1.68074513	0.98712819	0.98253931
rel 9/IDC	0.41612860	0.07038479	0.06972963
rel 9/MEP	0.00002184	0.00000000	0.00000000
rel 10/ORG	1.68074517	0.98712821	0.98253931
rel 10/IDC	0.41612860	0.07038479	0.06972963
rel 10/MEP	0.00002167	0.00000000	0.00000000
IDCT/ORG	2.17505877	1.89026210	1.87922527
MEP/ORG	2.08974301	1.53418394	1.51229428
MEP/IDCT	0.55463822	0.12546744	0.12336016
rel 1/ORG	2.23185020	1.81134653	1.75381757
rel 1/IDC	0.98027819	0.36803651	0.36488448

rel 7/ORG	2.58883987	2.32668406	2.31295071
rel 7/IDC	0.63618408	0.17424921	0.17010502
rel 7/MEP	0.00002717	0.00000000	0.00000000
rel 8/ORG	2.58883980	2.32668358	2.31295071
rel 8/IDC	0.63618376	0.17424916	0.17010502
rel 8/MEP	0.00002300	0.00000000	0.00000000
rel 9/ORG	2.58883974	2.32668350	2.31295071
rel 9/IDC	0.63618378	0.17424914	0.17010502
rel 9/MEP	0.00002308	0.00000000	0.00000000
rel 10/ORG	2.58883973	2.32668350	2.31295071
rel 10/IDC	0.63618379	0.17424914	0.17010502
rel 10/MEP	0.00002312	0.00000000	0.00000000
IDCT/ORG	3.17117118	3.87543085	3.79406342
MEP/ORG	3.04123094	3.29728492	3.25319297
MEP/IDCT	0.63617124	0.16048044	0.15810993
rel 1/ORG	3.11974020	3.51409872	3.42563743
rel 1/IDC	0.94300255	0.33348927	0.33055524
rel 1/MEP	0.66698487	0.17237829	0.17244344
rel 2/ORG	3.04164789	3.30271603	3.25663954
rel 2/IDC	0.65220548	0.16387511	0.16155691
rel 2/MEP	0.10111241	0.00344737	0.00344679
rel 3/ORG	3.04123518	3.29725921	3.25329982
rel 3/IDC	0.63584194	0.16062756	0.15821698
rel 3/MEP	0.01627981	0.00010702	0.00010704
rel 4/ORG	3.04121057	3.29717925	3.25319476
rel 4/IDC	0.63609370	0.16047722	0.15811192
rel 4/MEP	0.00220610	0.00000192	0.00000192
rel 5/ORG	3.04122373	3.29726926	3.25319290
rel 5/IDC	0.63617193	0.16047869	0.15811006
rel 5/MEP	0.00032292	0.00000005	0.00000005
rel 6/ORG	3.04122867	3.29728417	3.25319286
rel 6/IDC	0.63617525	0.16048028	0.15811002
rel 6/MEP	0.00006489	0.00000000	0.00000000
rel 7/ORG	3.04122869	3.29728451	3.25319285
rel 7/IDC	0.63617397	0.16048041	0.15811001
rel 7/MEP	0.00002973	0.00000000	0.00000000
rel 8/ORG	3.04122863	3.29728416	3.25319285
rel 8/IDC	0.63617367	0.16048040	0.15811001
rel 8/MEP	0.00002924	0.00000000	0.00000000
rel 9/ORG	3.04122862	3.29728409	3.25319285
rel 9/IDC	0.63617366	0.16048039	0.15811001
rel 9/MEP	0.00002926	0.00000000	0.00000000
rel 10/ORG	3.04122861	3.29728409	3.25319285
rel 10/IDC	0.63617366	0.16048039	0.15811001
rel 10/MEP	0.00002925	0.00000000	0.00000000
IDCT/ORG	3.68085725	5.08986580	4.97073488
MEP/ORG	3.56396930	4.49390578	4.45827851
MEP/IDCT	0.64810045	0.15764811	0.15689488
rel 1/ORG	3.66091645	4.72323491	4.62941239

rel 1/IDC	0.96333998	0.33204187	0.32802938
rel 1/MEP	0.68673106	0.17216526	0.17113301
rel 2/ORG	3.56237217	4.49755327	4.46133204
rel 2/IDC	0.66327465	0.16057026	0.15994876
rel 2/MEP	0.09332134	0.00305181	0.00305402
rel 3/ORG	3.56344370	4.49340908	4.45836425
rel 3/IDC	0.64837869	0.15766943	0.15698078
rel 3/MEP	0.01453231	0.00008586	0.00008590
rel 4/ORG	3.56393328	4.49378621	4.45827975
rel 4/IDC	0.64799337	0.15764311	0.15689627
rel 4/MEP	0.00189874	0.00000132	0.00000132
rel 5/ORG	3.56397455	4.49390234	4.45827845
rel 5/IDC	0.64808781	0.15764869	0.15689498
rel 5/MEP	0.00022035	0.00000002	0.00000002
rel 6/ORG	3.56397027	4.49390795	4.45827843
rel 6/IDC	0.64810364	0.15764828	0.15689495
rel 6/MEP	0.00003852	0.00000000	0.00000000
rel 7/ORG	3.56396852	4.49390594	4.45827843
rel 7/IDC	0.64810356	0.15764809	0.15689495
rel 7 MEP	0.00002002	0.00000000	0.00000000
rel 8/ORG	3.56396840	4.49390566	4.45827843
rel 8/IDC	0.64810327	0.15764808	0.15689495
rel 8/MEP	0.00001859	0.00000000	0.00000000
rel 9/ORG	3.56396842	4.49390567	4.45827843
rel 9/IDC	0.64810323	0.15764808	0.15689495
rel 9/MEP	0.00001821	0.00000000	0.00000000
rel 10/ORG	3.56396843	4.49390568	4.45827843
rel 10/IDC	0.64810324	0.15764808	0.15689495
rel 10/MEP	0.00001819	0.00000000	0.00000000
IDCT/ORG	5.21114184	9.81343355	9.38685482
MEP/ORG	5.14234834	9.45834966	9.06846657
MEP/IDCT	0.37767784	0.05917869	0.05879711
rel 1/ORG	5.16699009	9.53435462	9.12250212
rel 1/IDC	0.52063656	0.11416658	0.11283262
rel 1/MEP	0.37020363	0.05436192	0.05403359
rel 2/ORG	5.14230749	9.46509084	9.06929391
rel 2/IDC	0.38384706	0.05999590	0.05962461
rel 2/MEP	0.04668706	0.00082714	0.00082753
rel 3/ORG	5.14219136	9.45845803	9.06848449
rel 3/IDC	0.37797761	0.05918233	0.05881513
rel 3/MEP	0.00637613	0.00001804	0.00001804
rel 4/ORG	5.14233071	9.45830509	9.06846687
rel 4/IDC	0.37765879	0.05917835	0.05879750
rel 4/MEP	0.00093526	0.00000037	0.00000037
rel 5/ORG	5.14234725	9.45834607	9.06846650
rel 5/IDC	0.37767294	0.05917881	0.05879714
rel 5/MEP	0.00010490	0.00000000	0.00000000
rel 6/ORG	5.14234787	9.45835149	9.06846650
rel 6/IDC	0.37767856	0.05917872	0.05879713

rel 6/MEP	0.00001830	0.00000000	0.00000000
rel 7/ORG	5.14234766	9.45835155	9.06846650
rel 7/IDC	0.37767914	0.05917869	0.05879713
rel 7/MEP	0.00001706	0.00000000	0.00000000
rel 8/ORG	5.14234763	9.45835148	9.06846650
rel 8/IDC	0.37767912	0.05917869	0.05879713
rel 8/MEP	0.00001766	0.00000000	0.00000000
rel 9/ORG	5.14234762	9.45835146	9.06846650
rel 9/IDC	0.37767911	0.05917869	0.05879713
rel 9/MEP	0.00001773	0.00000000	0.00000000
rel 10/ORG	5.14234762	9.45835146	9.06846650
rel 10/IDC	0.37767911	0.05917869	0.05879713
rel 10/MEP	0.00001773	0.00000000	0.00000000
IDCT/ORG	5.56642803	11.02811602	10.61233813
MEP/ORG	5.53993731	10.88033247	10.47380261
MEP/IDCT	0.24699515	0.02471669	0.02467061
rel 1/ORG	5.54656090	10.89048553	10.49762167
rel 1/IDC	0.33959755	0.04860747	0.04848951
rel 1/MEP	0.23866196	0.02387087	0.02381792
rel 2/ORG	5.54009548	10.88592757	10.47411522
rel 2/IDC	0.24997580	0.02503159	0.02498328
rel 2/MEP	0.02770889	0.00031249	0.00031259
rel 3/ORG	5.53993154	10.88065839	10.47380396
rel 3/IDC	0.24703560	0.02471748	0.02467198
rel 3/MEP	0.00176711	0.00000137	0.00000137
rel 4/ORG	5.53993470	10.88034016	10.47380261
rel 4/IDC	0.24699779	0.02471674	0.02467063
rel 4/MEP	0.00020401	0.00000002	0.00000002
rel 5/ORG	5.53993676	10.88033310	10.47380259
rel 5/IDC	0.24699586	0.02471669	0.02467061
rel 5/MEP	0.00002905	0.00000000	0.00000000
rel 6/ORG	5.53993694	10.88033425	10.47380259
rel 6/IDC	0.24699606	0.02471669	0.02467061
rel 6/MEP	0.00001432	0.00000000	0.00000000
rel 7/ORG	5.53993694	10.88033443	10.47380259
rel 7/IDC	0.24699612	0.02471669	0.02467061
rel 7/MEP	0.00001402	0.00000000	0.00000000
rel 8/ORG	5.53993694	10.88033445	10.47380259
rel 8/IDC	0.24699613	0.02471669	0.02467061
rel 8/MEP	0.00001402	0.00000000	0.00000000
rel 9/ORG	5.53993694	10.88033445	10.47380259
rel 9/IDC	0.24699613	0.02471669	0.02467061
rel 9/MEP	0.00001402	0.00000000	0.00000000
rel 10/ORG	5.53993694	10.88033445	10.47380259
rel 10/IDC	0.24699613	0.02471669	0.02467061
rel 10/MEP	0.00001402	0.00000000	0.00000000
IDCT/ORG	5.76460163	11.74303887	11.42050742
MEP/ORG	5.74750597	11.62276946	11.31845460
MEP/IDCT	0.26472093	0.02780918	0.02777577

rel 1/ORG	5.75566913	11.62887343	11.33818607
rel 1/IDC	0.34137343	0.04764255	0.04750711
rel 1/MEP	0.22178003	0.01974736	0.01973153
rel 2/ORG	5.74746231	11.62532604	11.31855510
rel 2/IDC	0.26564762	0.02790963	0.02787629
rel 2/MEP	0.01547255	0.00010049	0.00010050
rel 3/ORG	5.74750095	11.62288178	11.31845536
rel 3/IDC	0.26475436	0.02780985	0.02777653
rel 3/MEP	0.00134868	0.00000076	0.00000076
rel 4/ORG	5.74750443	11.62276160	11.31845460
rel 4/IDC	0.26472051	0.02780918	0.02777578
rel 4/MEP	0.00011926	0.00000001	0.00000001
rel 5/ORG	5.74750595	11.62276780	11.31845459
rel 5/IDC	0.26472074	0.02780918	0.02777577
rel 5/MEP	0.00001072	0.00000000	0.00000000
rel 6/ORG	5.74750600	11.62276923	11.31845459
rel 6/IDC	0.26472093	0.02780918	0.02777577
rel 6/MEP	0.00000253	0.00000000	0.00000000
rel 7/ORG	5.74750599	11.62276930	11.31845459
rel 7/IDC	0.26472094	0.02780918	0.02777577
rel 7/MEP	0.00000213	0.00000000	0.00000000
rel 8/ORG	5.74750599	11.62276930	11.31845459
rel 8/IDC	0.26472094	0.02780918	0.02777577
rel 8/MEP	0.00000211	0.00000000	0.00000000
rel 9/ORG	5.74750599	11.62276930	11.31845459
rel 9/IDC	0.26472094	0.02780918	0.02777577
rel 9/MEP	0.00000211	0.00000000	0.00000000
rel 10/ORG	5.74750599	11.62276930	11.31845459
rel 10/IDC	0.26472094	0.02780918	0.02777577
rel 10/MEP	0.00000211	0.00000000	0.00000000
IDCT/ORG	6.10544076	13.01326134	12.88445234
MEP/ORG	6.10283924	12.96622734	12.84424307
MEP/IDCT	0.21854120	0.01949673	0.01946911
rel 1/ORG	6.10816076	12.94902322	12.86104487
rel 1/IDC	0.29278188	0.03649500	0.03627065
rel 1/MEP	0.20251676	0.01680982	0.01680190
rel 2/ORG	6.10288836	12.96629615	12.84426084
rel 2/IDC	0.21857600	0.01951869	0.01948682
rel 2/MEP	0.00648485	0.00001776	0.00001776
rel 3/ORG	6.10283940	12.96627092	12.84424315
rel 3/IDC	0.21854537	0.01949678	0.01946913
rel 3/MEP	0.00038444	0.00000006	0.00000006
rel 4/ORG	6.10283943	12.96623518	12.84424309
rel 4/IDC	0.21854373	0.01949669	0.01946907
rel 4/MEP	0.00002673	0.00000000	0.00000000
rel 5/ORG	6.10283942	12.96623397	12.84424309
rel 5/IDC	0.21854367	0.01949669	0.01946907
rel 5/MEP	0.00002369	0.00000000	0.00000000
rel 6/ORG	6.10283942	12.96623397	12.84424309

rel 6/IDC	0.21854367	0.01949669	0.01946907
rel 6/MEP	0.00002376	0.00000000	0.00000000
rel 7/ORG	6.10283942	12.96623397	12.84424309
rel 7/IDC	0.21854367	0.01949669	0.01946907
rel 7/MEP	0.00002376	0.00000000	0.00000000
rel 8/ORG	6.10283942	12.96623397	12.84424309
rel 8/IDC	0.21854367	0.01949669	0.01946907
rel 8/MEP	0.00002376	0.00000000	0.00000000
rel 9/ORG	6.10283942	12.96623397	12.84424309
rel 9/IDC	0.21854367	0.01949669	0.01946907
rel 9/MEP	0.00002376	0.00000000	0.00000000
rel 10/ORG	6.10283942	12.96623397	12.84424309
rel 10/IDC	0.21854367	0.01949669	0.01946907
rel 10/MEP	0.00002376	0.00000000	0.00000000
IDCT/ORG	6.36202443	14.10041438	14.20157969
MEP/ORG	6.32082559	13.87477387	13.97970140
MEP/IDCT	0.23906526	0.02478753	0.02473879
rel 1/ORG	6.32795006	13.84475586	13.99627013
rel 1/IDC	0.29027932	0.04179758	0.04130707
rel 1/MEP	0.19181175	0.01663347	0.01656810
rel 2/ORG	6.32101931	13.87306095	13.97976706
rel 2/IDC	0.23870557	0.02486837	0.02480430
rel 2/MEP	0.01216930	0.00006554	0.00006553
rel 3/ORG	6.32083210	13.87466969	13.97970182
rel 3/IDC	0.23903829	0.02478866	0.02473906
rel 3/MEP	0.00084153	0.00000030	0.00000030
rel 4/ORG	6.32082582	13.87477098	13.97970151
rel 4/IDC	0.23906488	0.02478754	0.02473876
rel 4/MEP	0.00004645	0.00000000	0.00000000
rel 5/ORG	6.32082543	13.87477660	13.97970151
rel 5/IDC	0.23906649	0.02478749	0.02473876
rel 5/MEP	0.00001287	0.00000000	0.00000000
rel 6/ORG	6.32082541	13.87477693	13.97970151
rel 6/IDC	0.23906658	0.02478749	0.02473876
rel 6/MEP	0.00001292	0.00000000	0.00000000
rel 7/ORG	6.32082541	13.87477695	13.97970151
rel 7/IDC	0.23906659	0.02478749	0.02473876
rel 7/MEP	7.00001294	0.00000000	0.00000000
rel 8/ORG	6.32082541	13.87477695	13.97970151
rel 8/IDC	0.23906659	0.02478749	0.02473876
rel 8/MEP	0.00001294	0.00000000	0.00000000
rel 9/ORG	6.32082541	13.87477695	13.97970151
rel 9/IDC	0.23906659	0.02478749	0.02473876
rel 9/MEP	0.00001294	0.00000000	0.00000000
rel 10/ORG	6.32082541	13.87477695	13.97970151
rel 10/IDC	0.23906659	0.02478749	0.02473876
rel 10/MEP	0.00001294	0.00000000	0.00000000
IDCT/ORG	7.02090915	16.61680268	17.46160240
MEP/ORG	7.03829741	16.69795548	17.55254213

MEP/IDCT	0.06893337	0.00192879	0.00192650
rel 1/ORG	7.03832156	16.69339138	17.55366342
rel 1/IDC	0.08516566	0.00305704	0.00304777
rel 1/MEP	0.05232799	0.00112153	0.00112132
rel 2/ORG	7.03829957	16.69785573	17.55254313
rel 2/IDC	0.06891978	0.00192993	0.00192750
rel 2/MEP	0.00158706	0.00000100	0.00000100
rel 3/ORG	7.03829729	16.69795292	17.55254213
rel 3/IDC	0.06893255	0.00192880	0.00192650
rel 3/MEP	0.00001704	0.00000000	0.00000000
rel 4/ORG	7.03829727	16.69795405	17.55254213
rel 4/IDC	0.06893291	0.00192879	0.00192650
rel 4/MEP	0.00000589	0.00000000	0.00000000
rel 5/ORG	7.03829727	16.69795406	17.55254213
rel 5/IDC	0.06893291	0.00192879	0.00192650
rel 5/MEP	0.00000592	0.00000000	0.00000000
rel 6/ORG	7.03829727	16.69795406	17.55254213
rel 6/IDC	0.06893291	0.00192879	0.00192650
rel 6/MEP	0.00000592	0.00000000	0.00000000
rel 7/ORG	7.03829727	16.69795406	17.55254213
rel 7/IDC	0.06893291	0.00192879	0.00192650
rel 7/MEP	0.00000592	0.00000000	0.00000000
rel 8/ORG	7.03829727	16.69795406	17.55254213
rel 8/IDC	0.06893291	0.00192879	0.00192650
rel 8/MEP	0.00000592	0.00000000	0.00000000
rel 9/ORG	7.03829727	16.69795406	17.55254213
rel 9/IDC	0.06893291	0.00192879	0.00192650
rel 9/MEP	0.00000592	0.00000000	0.00000000
rel 10/ORG	7.03829727	16.69795406	17.55254213
rel 10/IDC	0.06893291	0.00192879	0.00192650
rel 10/MEP	0.00000592	0.00000000	0.00000000

REFERENCES

Ahmed, N., and Roa, K.P. Orthogonal Transforms for Digital Signal Processing. New York: Springer-Verlag, 1975.

Ahmed, N.; Natarjan, T.; and Roa, K.R. "Discrete Cosine Transform." IEEE Transactions on Computers (January 1974): 90-93.

Clarke, R.J. Transform Coding of Images. New York: Academic Press, 1985.

Healy, D.J., and Michell, R.O. "Video Bandwidth Compression Using Block Truncation Coding." IEEE International Conference on Communications (June 1981): 14-15.

Jain, A.K. "A Sinusoidal Family of Unitary Transforms." IEEE Transactions on Pattern Analysis and Machine Intelligence. (October 1979): 356-365.

Jaynes, E.T. "Prior Probabilities." IEEE Transactions on Systems Science and Cybernetics SSC-4 (September 1968): 227-241.

Karmangar, F.A., and Rao, K.R. "Fast Algorithms for the 2-D Discrete Cosine Transform." IEEE Transactions on Computers. (September 1982): 899-906.

Kernighan, Brian W., and Ritchie, Dennis M. The C Programming Language. Englewood Cliffs, NJ: Prentice-Hall, 1978.

Mailaender, Laurence E. "Transform Domain Vector Quantization for Low-Rate Image Coding." Master's thesis, University of Virginia, May 1985.

Mailaender, Laurence E. Interview by author, January 1986, at Stanford Telecommunications, McLean, VA.

Oppenheim, A.V., and Schafer, R.W. Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1975.

Shore, J.E. "Inversion as Logical Inference-Theory and Applications of Maximum Entropy and Minimum Cross-Entropy." SIAMA-AMS Proceedings 14, 1984.

Shore, J.E. "Minimum Cross-Entropy Spectral Analysis." IEEE Transactions Acoustics, Speech and Signal Processing ASSP-29 (April 1981).

Tzannes, Michael A. "Transform Coded Data Reconstruction Using the Principles of Maximum Entropy and Minimum Relative Entropy." Senior project, University of Michigan-Ann Arbor, 1985.

Tzannes, N.S. Communication and Radar Systems. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1985.

Tzannes, N.S., and Bodenschatz, J.S. "Iterative Reconstruction of Transform-Coded Images Using Relative Entropy." In Proceedings of the IEEE Southeastcon '87, Tampa, FL, April 5-8, 1987.

Tzannes, N.S.; Thacker, J.A.; and Tzannes, M.A. "Entropy-Preserving Block Truncation Coding." In Proceedings of the IEEE Southeastcon '86.

Tzannes, N.S., and Tzannes, M.A. "Reconstruction of Transform-Coded Data by RE Minimization." In Proceedings of the 12th Annual Conference on Information Sciences, Princeton University, March 1986.