

THE DISCRETE COSINE TRANSFORM

by

MYRON DALE FLICKNER

B. S., Kansas State University, 1980

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

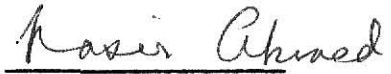
MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1982

Approved by:


Major Professor

Spec.
Coll
LD
2668
.T4
1982
F64
C.2

111202 248388

TABLE OF CONTENTS

	<u>Page</u>
Chapter I: Introduction	1
Chapter II: Definitions	3
Chapter III: Computational Algorithms	9
Chapter IV: Image Compression Applications	15
Chapter V: Conclusions	34
References	35
Acknowledgment	38
Appendix A: A Fast Discrete Cosine Routine	39
Appendix B: Simulation programs	43

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
1. Plot of w_m	8
2. In-place FDCT signal flow graph	13
3. Typical transform coding system	16
4. Plot of residual correlation	21
5. Transform basis planes	22
6. DCT and WHT comparison	25
7. DST and SCT comparison	26
8. Additional comparisons	29-30
9. Higher compression comparison	31-33

LIST OF TABLES

<u>Table</u>	<u>Page</u>
1. Endpoint error	24
2. Transformations of a Constant Matrix	28

Chapter I

Introduction

In recent years there has been extensive use of orthogonal transforms for image and speech compression, pattern recognition, noise cancellation, and Wiener filtering. This is a tutorial paper on the Discrete Cosine Transform (DCT), one of the most popular orthogonal transforms. The DCT was introduced by Ahmed et al. [1] in 1974. Since the DCT's inception, it has been primarily used for transform coding of images. Other applications include speech prediction [22-25] and noise cancellation [27-28].

The two-dimensional DCT is used in image transform coding to decorrelate picture data, and pack the image energy into a few transform components. For small block sizes, image data is statistically modeled as a first-order Markov random process. The DCT is used to reduce redundant information, thus allowing a digital picture to be represented with fewer bits.

For speech coding, the one-dimensional DCT is used to decorrelate speech data, and pack the speech signal energy in a few transform components. Speech data is treated as wide-sense stationary on a short term basis. The DCT is often combined with an adaptive encoding scheme. Such methods have made possible hardware implementation of a 9.6 kilo bits/second speech transmission system [26].

Keshaven [27-28] has used the DCT to assist in implementing a noise cancellation scheme. Here the DCT is used to convert a vector interpolation scheme into an implementable scalar scheme.

The DCT is derived in Chapter II. Chapter III shows how to implement the DCT efficiently. Chapter IV shows how the DCT can be used in image data compression. It also includes a comparison between the DCT and the newly introduced Symmetric Cosine Transform (SCT) [2], the Walsh Hadamard Transform (WHT) [3], and the fast Karhunen-Loeve Transform (DST) [4]. Conclusions are presented in Chapter V.

CHAPTER II

Definitions

The discrete cosine transform (DCT) of an N-point data sequence is defined as

$$F(k) = \sqrt{\frac{2}{N}} c(k) \sum_{m=0}^{N-1} x(m) \cos\left\{\frac{(2m+1)}{2N} k\pi\right\} ; \quad 0 \leq m, k \leq N-1,$$

and the inverse discrete cosine transform (IDCT) is defined as

$$x(m) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} c(k) F(k) \cos\left\{\frac{(2m+1)}{2N} k\pi\right\} ; \quad 0 \leq m, k \leq N-1$$

where

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } k = 0 \\ 1 & \text{for } k \neq 0 \end{cases} .$$

It can be easily shown the DCT of an N point data sequence can be represented in matrix form as

$$\overline{F} = \overline{A} \overline{X}$$

where \overline{X} is the N x 1 column vector containing the data sequence; \overline{A} is a N x N coefficient matrix whose elements, $a_{k,m}$, are computed as

$$a_{k+1,m+1} = c(k) \cos\left\{\frac{(2m+1)}{2N} k\pi\right\} ; \quad 0 \leq m, k \leq N-1$$

where

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & k = 0 \\ 1 & k \neq 0 \end{cases} ;$$

\overline{F} is the resulting N x 1 column vector containing the DCT coefficients.

Higher dimensional DCT definitions can be arrived at by extending the one-dimensional DCT definition. For example, the two-dimensional DCT is defined as

$$F(k, \ell) = \frac{2}{N} c(k) c(\ell) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m, n) \cos\left\{\frac{(2m+1)}{2N} k\pi\right\} \cos\left\{\frac{(2n+1)}{2N} \ell\pi\right\}.$$

Similarly the two-dimensional IDCT is defined as

$$x(m, n) = \frac{2}{N} \sum_{k=0}^{N-1} \sum_{\ell=0}^{N-1} c(k) c(\ell) F(k, \ell) \cos\left\{\frac{(2m+1)}{2N} k\pi\right\} \cos\left\{\frac{(2n+1)}{2N} \ell\pi\right\}$$

where

$$c(k) = \begin{cases} \sqrt{\frac{1}{2}} & k = 0 \\ 1 & k \neq 0 \end{cases}.$$

The DCT can be shown to be a separable transform. To illustrate, let us write the two-dimensional DCT definition as follows:

$$F(k, \ell) = \sqrt{\frac{2}{N}} c(k) \sum_{m=0}^{N-1} \left\{ \sqrt{\frac{2}{N}} c(\ell) \sum_{n=0}^{N-1} x(m, n) \cos\left[\frac{(2n+1)}{2N} k\pi\right] \right\} \cos\left[\frac{(2m+1)}{2N} \ell\pi\right]$$

It can be seen that the term in curly brackets is the one-dimensional DCT of $x(m, n)$ by columns. If the bracketed term is replaced by $\hat{X}(m, \ell)$, then the resulting expression

$$F(k, \ell) = \sqrt{\frac{2}{N}} c(k) \sum_{m=0}^{N-1} \hat{X}(m, \ell) \cos\left[\frac{(2m+1)}{2N} k\pi\right]$$

is the one-dimensional DCT of $\hat{X}(m, \ell)$ by rows. Thus, the two-dimensional DCT can be computed by taking the one-dimensional DCT by columns and then taking the one-dimensional DCT of the result by rows. Note that the order of processing could be changed by first computing the one-

dimensional DCT by rows and then computing the one-dimensional DCT by columns. In matrix form, separability can be represented as

$$\bar{F} = \bar{A} \bar{X} \bar{A}'$$

where \bar{F} is the $N \times N$ matrix containing the DCT coefficients, \bar{A} is the $N \times N$ coefficient matrix, and \bar{X} is the $N \times N$ data matrix.

The DCT has several interesting properties. Since it is an orthogonal transform, it satisfies the relation

$$\sum_{m=0}^{N-1} (x(m))^2 = \sum_{k=0}^{N-1} (F(k))^2$$

which is called Parseval's Theorem. Another interesting point is the fact that the basis vector of the DCT, defined as the $N \times 1$ column vector $B(k)$ whose elements are computed as

$$b_m = \sqrt{\frac{2}{N}} c(k) \cos \left[\left(\frac{2m+1}{2N} \right) k\pi \right], \quad 0 \leq k, m \leq N-1$$

are actually the roots of discrete Chebyshev polynomials [1]. The basis vector interpretation is similar to a Fourier series expansion. Recall that any periodic function $f(x)$ can be represented as an infinite sum of sine and cosine terms. Since the sine and cosine can be written in terms of complex exponentials we have

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{-jn\omega_0 t}$$

where $j = \sqrt{-1}$. Similarly, we can express any finite data sequence as a finite weighted sum of DCT basis vectors

$$x(m) = \sum_{k=0}^{N-1} F(k) \bar{B}(k) .$$

For example, with $N=4$ the DCT basis vectors are shown below:

$$B(0) = \begin{bmatrix} .500 \\ .500 \\ .500 \\ .500 \end{bmatrix} \quad B(1) = \begin{bmatrix} .653 \\ .271 \\ -.271 \\ -.653 \end{bmatrix} \quad B(2) = \begin{bmatrix} .500 \\ -.500 \\ -.500 \\ .500 \end{bmatrix} \quad B(3) = \begin{bmatrix} .271 \\ -.653 \\ .653 \\ -.271 \end{bmatrix} .$$

To represent the data sequence $\bar{X}' = [1, 2, 3, 4]$ as a DCT expansion the following computation is made:

$$\bar{X} = F(0) \bar{B}(0) + F(1) \bar{B}(1) + F(2) \bar{B}(2) + F(3) \bar{B}(3)$$

or

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} = 5.0 \begin{bmatrix} .500 \\ .500 \\ .500 \\ .500 \end{bmatrix} - 2.23 \begin{bmatrix} .653 \\ .271 \\ -.271 \\ -.653 \end{bmatrix} + 0.0 \begin{bmatrix} .500 \\ -.500 \\ -.500 \\ .500 \end{bmatrix} - .159 \begin{bmatrix} .271 \\ -.653 \\ .653 \\ -.271 \end{bmatrix}$$

Basis planes, sometimes called basis images, are defined as the outer product of the above basis vectors. This definition is used to form the two-dimensional DCT basis functions.

It has been observed that the DCT is the best suboptimal first transform for a first order Markov random process. Ray and Driver [5] have shown the Karhunen Loeve Transform (KLT), the transform that minimizes the mean-squared-error, for a first order Markov random process is given by

$$F(k) = \sum_{m=0}^{N-1} a_m x(m) \sin \left[\omega_m \left(k - \frac{(N+1)}{2} \right) + (m+1) \frac{\pi}{2} \right] \quad 0 < m, k < N-1. \quad (1)$$

a_m , a normalization constant, is given by

$$a_m = \sqrt{\frac{2}{N+\lambda_m}}$$

where

$$\lambda_m = \frac{1 - \rho^2}{1 - 2\rho \cos \omega_m + \rho^2} .$$

Furthermore, $\{\omega_m\}$ is the set of positive roots of the following transcendental equation:

$$\tan(N\omega_m) = \frac{(\rho^2 - 1) \sin \omega_m}{\cos \omega_m - 2\rho + \rho^2 \cos \omega_m} . \quad (2)$$

Here ρ is defined as the statistical correlation between pixels.

Solving (2) as $\rho \rightarrow 1$ we get

$$\tan(N\omega_m) = 0$$

or

$$\omega_m = \frac{m\pi}{N} .$$

Substituting this result back into (1) and simplifying we get

$$F(k) = \sum_{m=0}^{N-1} a_m x(m) \cos\left(\frac{k\pi}{2N} (2m+1)\right) .$$

The kernel in the above transform is easily recognized as the DCT kernel. Thus, the DCT is the optimal transform for a first order Markov process as $\rho \rightarrow 1$ [29].

By looking at the transcendental equation (2) we can see that ω_m is a function of ρ . Unfortunately (2) does not yield a simple closed form solution. However, numerical techniques, such as Newton iteration, can solve (2) for fixed N . This solution for $N=8$ is shown in Figure 1. Observe that ω_m is almost a linear function of ρ . Therefore, the DCT is good approximation to the optimum transform for a first order Markov process for ρ near 1.

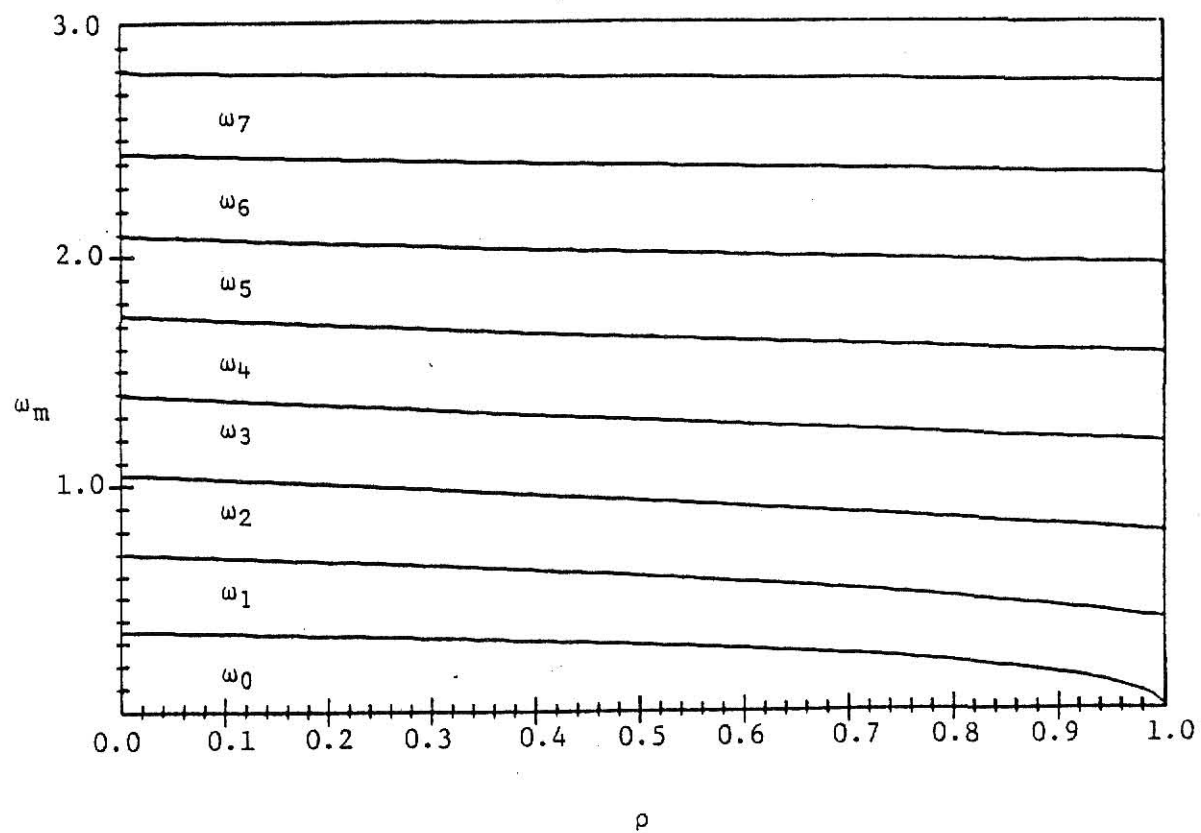


FIGURE 1
The roots ω_m for $N=8$

CHAPTER III

Computational Algorithms

When the DCT was introduced by Ahmed et al. [1] an algorithm to compute it using a $2N$ -point fast Fourier transform (FFT) was given. This algorithm can be derived by examining the definition of the DCT; i.e.,

$$F(k) = \sqrt{\frac{2}{N}} c(k) \sum_{m=0}^{N-1} x(m) \cos \left\{ \frac{(2m+1)}{2N} k\pi \right\}.$$

Representing the cosine term as a complex exponential results in

$$F(k) = \sqrt{\frac{2}{N}} c(k) \sum_{m=0}^{N-1} x(m) \operatorname{Real} \left\{ e^{-i \left(\frac{2m+1}{2N} \right) k\pi} \right\}.$$

Performing some simple algebra gives

$$F(k) = \operatorname{Real} \left\{ \sqrt{\frac{2}{N}} c(k) e^{-i \frac{k\pi}{2N}} \sum_{m=0}^{N-1} x(m) e^{-i \frac{2m}{2N} k\pi} \right\}.$$

Now, let $\hat{x}(m) = [x(m) \text{ appended with } N \text{ zeros}]$ so that

$$F(k) = \operatorname{Real} \left\{ \sqrt{\frac{2}{N}} c(k) e^{-i \frac{k\pi}{2N}} \left[\sum_{m=0}^{2N-1} \hat{x}(m) e^{-i \frac{2m}{2N} k\pi} \right] \right\}.$$

The term in square brackets is recognized as the discrete Fourier transform (DFT) of $\hat{x}(m)$. It can be computed via a fast Fourier transform (FFT) when N is a power of 2. Thus, the DCT can be computed by multiplying the DFT of $\hat{x}(m)$ by a complex number, and then taking the real part of the result. Although this algorithm's efficiency is better than computing the DFT by matrix multiplication, it requires $2N$ complex storage locations. This storage requirement can be reduced by utilizing

the fact that the input data sequence to the DFT is real. For a real input data sequence the resulting DFT is Hermitian. Thus, only $N + 1$ complex storage locations are needed. Many other methods of computing the DCT have been published since the DCT's inception. In 1976 Haralick [6,7] presented a method for computing the DCT using two N -point FFT's. A year later, an algorithm for computing the DCT was presented by Chen et al. [8] which yields a computation savings of one half over the conventional two N -point real FFT method. Chen's method is accomplished by factoring the DCT into sparse matrices and is amenable to hardware implementation. It requires only N real storage locations and does not use complex arithmetic. Unfortunately, it is difficult to generalize for different values of N . Chen's algorithm is the most common algorithm used for hardware implementation of fixed size DCT's.

A method of computing the DCT using a single N -point FFT of a recorded input sequence was given by Narasimha and Peterson [9] in 1978. In 1980, Wagh and Ganesh [10] presented a method for computing the DCT using a matrix partitioning approach. This technique shows that the partitioned submatrices are equivalent to group tables of Abelian groups. Wagh's algorithm has an advantage of working efficiently for an arbitrary number of points and, more importantly, when N is prime. Also in 1980, Dyer et al. [11] presented a technique for implementing the DCT in hardware via an arcsine transform. Dyer's method requires only additions and table-lookup operations. A fast algorithm for computing the two-dimensional DCT was presented by Kamanger and Rao [12] in 1980. Makhoul [13] generalized the algorithm developed by Narasimha and Peterson [9] to work with sequences containing an odd number of points. He also extended the same to compute the DCT of two-dimensional data.

The one-dimensional version of this algorithm is as efficient as Chen's [8] algorithm for values of N that are integer powers of 2. Finally, Nussbaumer [14] developed a method for computing fast multidimensional cosine transforms. This method is based on polynomial transforms.

The algorithm presented by Narasimha and Peterson [9] and modified by Makhoul [13] will be referred to as the fast discrete cosine transform (FDCT) and is described below.

First, we reorder the sequence $x(m)$ to get the sequence $v(m)$ according to the following rule:

$$v(m) = \begin{cases} x(2m) & 0 \leq m \leq \left\lfloor \frac{N-1}{2} \right\rfloor \\ x(2m-1) & \left\lfloor \frac{N+1}{2} \right\rfloor \leq m \leq N-1 \end{cases}$$

Taking the DFT of $v(m)$, we obtain $V(k)$. This can be done using an $N/2$ point complex FFT algorithm. Next, the resulting $V(k)$'s are multiplied by $2 \exp(-j\pi k/2N)$. The desired DCT coefficients are then obtained as

$$\left. \begin{aligned} F(k) &= \sqrt{\frac{2}{N}} c(k) \operatorname{Real}\{V(k)\} \\ F(N-k) &= -\sqrt{\frac{2}{N}} c(k) \operatorname{Imag}\{V(k)\} \end{aligned} \right\} 0 \leq k \leq \left\lfloor \frac{N}{2} \right\rfloor$$

where

$$c(k) = \begin{cases} \sqrt{\frac{1}{2}} & k = 0 \\ 1 & k \neq 0 \end{cases} .$$

We note that the above expression gives the value of $F(N) = 0$ even though this point is not needed. The above expression can be shown to be equivalent to a shuffle on the real sequence $\hat{V}(k)$ formed by

$$\begin{aligned} \hat{V}(2K) &= \operatorname{Real}\{V(k)\} \\ \hat{V}(2K+1) &= -\operatorname{Imag}\{V(k)\} \end{aligned} \quad 0 \leq k \leq \left\lfloor \frac{N}{2} \right\rfloor .$$

The shuffle is done as follows:

$$F(k) = \begin{cases} \hat{V}(2k) & 0 \leq k \leq \left\lfloor \frac{N-1}{2} \right\rfloor \\ \hat{V}(2N-2k+1) & \left\lfloor \frac{N+1}{2} \right\rfloor \leq k \leq N-1 \end{cases}$$

The FDCT algorithm requires $2N + 2$ real storage locations since the shuffle operations are not pairwise. An in-place shuffle algorithm can be derived by looking at a method to form bit-reversed sequences.

Recall that for $N=2^M$ the bit-reversed sequence can be arrived at in M operations as shown below for $M = 3$.

Step 1. Let the N numbers be arranged in index ascending order.

0 1 2 3 4 5 6 7 8

Step 2. Form two sequences by taking alternate elements.

0 2 4 6 1 3 5 7

Step 3. Form four sequences by taking alternate elements of the two $N/2$ point sequences.

0 4 2 6 1 5 3 7

This result is the bit-reversed sequence.

It is interesting to note that the sequence resulting in step 2 is very similar to the shuffle sequence for the DCT, which is

0 2 4 6 7 5 3 1.

Therefore, it appears the DCT shuffle can be done in place as a two step operation.

Step 1. Bit reverse the input data

Step 2. Apply a pairwise shuffle

The Step 2 shuffle has not been generalized or proven to be pairwise at this point. Examples of the inplace shuffles are illustrated in Figure 2 in the form of signal flow graphs.

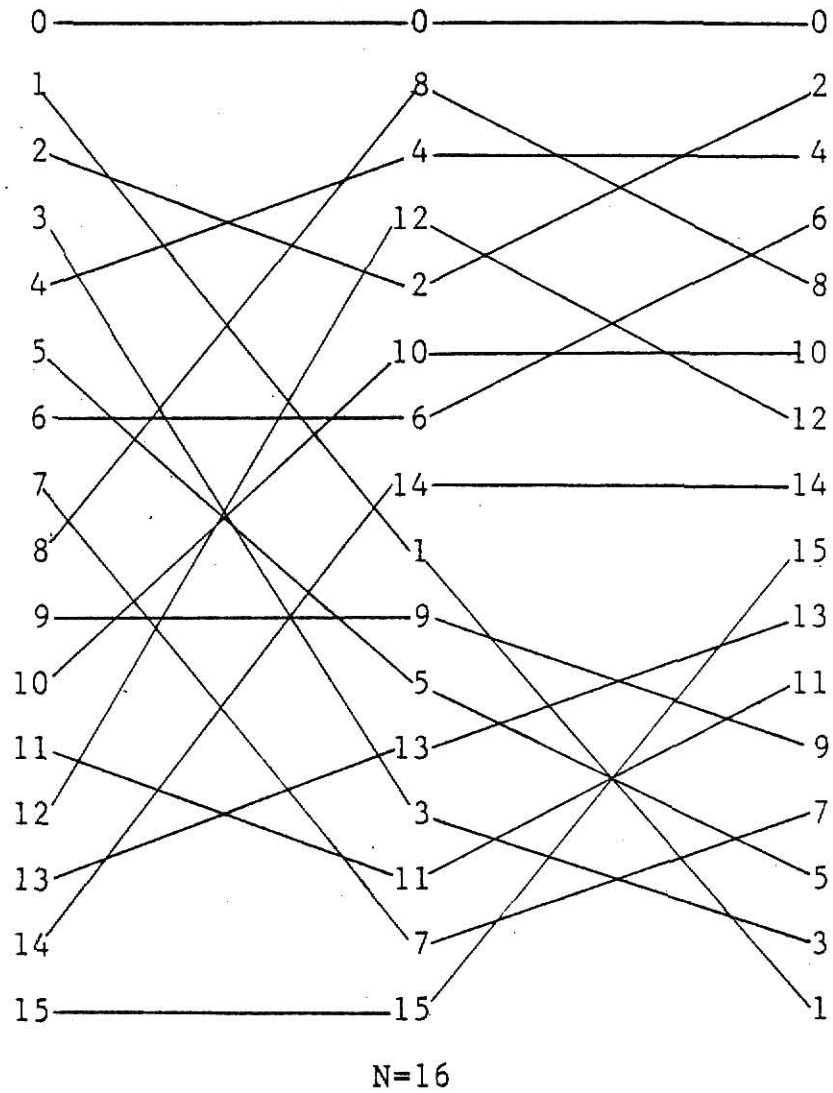


Figure 2

Signal flowgraph for in-place FDCT shuffle

The FDCT algorithm allows one to use existing software to compute the FFT instead of using a specialized algorithm. Appendix A contains a general FDCT routine.

Chapter IV

Image Compression Applications

Recently, many orthogonal transforms have been used in applications of image data compression, often called image transform coding. The object of transform coding is to represent a signal with fewer samples. Many different transforms (Fourier, Walsh, Hadamard, Karhune Loeve, Haar, Slant, Cosine, Sine, etc.) have been used for transform coding with varying degrees of success [15-17]. Kekre and Solenki [15] suggested that the DCT is the best transform for image transform coding. A typical image compression system is illustrated in Figure 3. The image is first blocked into $N \times N$ sub-blocks where N is typically 8 or 16. A two-dimensional transform is then applied to each sub-block. The resulting transform data is then quantized and coded. The coded information is transmitted to a receiver where the inverse process is performed. For example, consider the 4×4 matrix:

$$\begin{bmatrix} 83.0 & 38.0 & 37.0 & 31.0 \\ 47.0 & 22.0 & 13.0 & 11.0 \\ 49.0 & 11.0 & 4.0 & 2.0 \\ 37.0 & 11.0 & 2.0 & 0.0 \end{bmatrix} .$$

Taking the two-dimensional DCT results in the following matrix

$$\begin{bmatrix} 99.5 & 59.7 & 30.5 & 14.8 \\ 49.0 & 3.2 & 3.1 & 4.9 \\ 20.0 & 1.1 & 1.0 & 2.8 \\ 10.0 & 6.4 & 6.3 & 5.3 \end{bmatrix} .$$

By retaining only the first row and column of the transformed coefficients one can try to approximate the original data since the remaining coefficients are relatively small. Replacing the other points with zero, a known value which would thus not be transmitted, results in

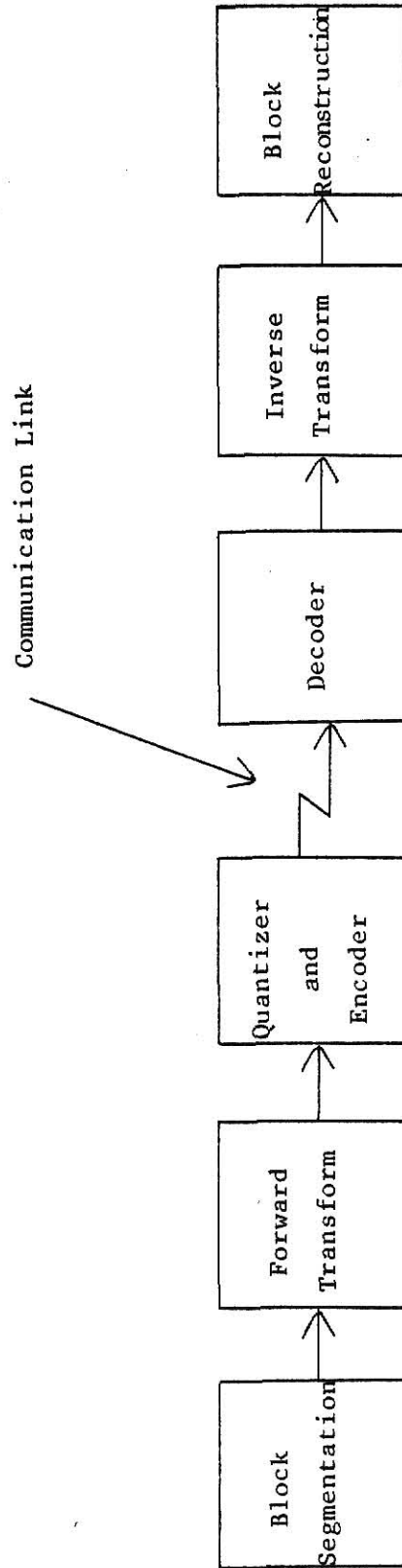


FIGURE 3

Block diagram of a typical transform coding system

$$\begin{bmatrix} 99.5 & 59.7 & 30.5 & 14.8 \\ 49.0 & 0 & 0 & 0 \\ 20.0 & 0 & 0 & 0 \\ 10.0 & 0 & 0 & 0 \end{bmatrix} .$$

The nonzero elements would then be quantized typically using Max' [18] quantizers, coded and finally transmitted. The receiver would decode the data and, perform an inverse transform. If perfect quantizers and coders were used the result would be

$$\begin{bmatrix} 84.0 & 38.1 & 37.2 & 31.4 \\ 47.0 & 22.3 & 13.7 & 11.0 \\ 48.6 & 11.0 & 4.0 & 2.4 \\ 36.9 & 11.4 & 2.3 & -1.3 \end{bmatrix}$$

It is obvious that the above image is a fair approximation to the original image, yet only one-fourth the information was transmitted. Typical quantizers and coders do not increase the error significantly, and they can increase the compression rate. The root-mean-squared error introduced in the given example is .50. Obviously the error would change if the data or the transform changed. Thus, different transforms perform differently for different types of data. As a result, images are statistically modeled to determine the best transform. It can be shown that the transform that minimizes the least mean squared error is the Karhunen Loeve Transform (KLT) [19]. The KLT is the transform that diagonalizes (perfectly decorrelates) the data domain covariance matrix, \bar{C}_z , defined as

$$\bar{C}_z = E\left\{(Z - \bar{Z})(Z - \bar{Z})'\right\}$$

where $E\{\cdot\}$ denotes statistical expectation.

Clearly, the KLT is composed of the augmented eigenvectors of the data domain covariance matrix. For a block size of 16, the covariance matrix would be 256 x 256. Computing the eigenvectors of such a matrix is indeed a complex computational task. Also, once the eigenvectors are

computed, it is doubtful that the resulting transform has a fast computational algorithm. Thus, the KLT has little hope for hardware implementation. Furthermore, it is not clear that the mean-squared error is the best measure of visual differences. Still, the KLT is used as an optimal reference point for transform consideration since it yields the smallest mean squared error when the transformed image is reconstructed.

Most images are statistically modeled as a first-order Markov random process within a small ($N < 16$) block. As a result the data covariance matrix

$$\bar{C}_x = \begin{bmatrix} 1 & \rho & \rho^2 & . & . & \rho^N \\ \rho & 1 & \rho & . & . & \rho^{N-1} \\ \rho^2 & \rho & 1 & . & . & . \\ . & . & . & . & . & . \\ \rho^N & . & . & . & . & 1 \end{bmatrix}$$

is a Toeplitz matrix. Here ρ is the correlation coefficient between pixels. The best approximation of the KLT for the first-order Markov model is one criterion for transform selection. Another transform selection tool is the variance criterion. The diagonal terms of the transform domain covariance matrix represent the variance (energy) of the transformed components. By treating the rows and columns independently, the statistical model can be reduced from a $N^2 \times N^2$ dimension to a $N \times N$ dimension. Thus, to approximate the variance of the $N \times N$ block compute the following matrix:

$$\begin{bmatrix} \sigma_{11}^2 & \sigma_{11}\sigma_{22} & . & \sigma_{NN}\sigma_{11} \\ \sigma_{11}\sigma_{22} & \sigma_{22}^2 & . & . \\ . & . & . & . \\ \sigma_{NN}\sigma_{11} & . & . & \sigma_{NN}^2 \end{bmatrix}$$

where the σ_{ii} 's are the diagonal terms of the transform domain covariance matrix. To minimize the energy loss, retain the N values with the highest variances. The transform that results in a variance distribution that has relatively few large variances, with the remaining variances very small, will minimize energy loss. It can be shown that the KLT is also optimal in this sense [20].

Recently, Kitajima [2] introduced the Symmetric Cosine Transform (SCT) defined in matrix form as \bar{A} whose elements are given by

$$a_{k+1,m+1} = \sqrt{\frac{2}{N-1}} u(k) u(m) \cos\left(\frac{km}{N-1} \pi\right), \quad 0 \leq k, m \leq N-1$$

where

$$u(k) = \begin{cases} \sqrt{\frac{1}{2}} & k = 0, \quad k = N-1 \\ 1 & \text{elsewhere} \end{cases}$$

Since the A matrix is symmetrical, the inverse SCT is identical to the forward transform. Kitajima developed a simple data-dependent windowing function W defined as

$$\bar{W} = \text{diag} \left[\frac{1}{\sqrt{1+\rho^2}}, 1, 1, \dots, \frac{1}{\sqrt{1+\rho^2}} \right]$$

The window function was used to produce a new vector $\bar{y} = W\bar{x}$. The SCT could then be taken of the \bar{y} vector. Note, the windowing process is identical to defining a new orthogonal transform $\bar{T} = \bar{A}\bar{W}$, where A is the SCT matrix. The windowed SCT can be shown to perform better than the DCT and Discrete Sine Transform (DST) for $N \gg \frac{\rho+1}{\rho-1}$.

A simulation was run to compare the DCT, Discrete Sine Transform (FKLT, DST) [4], SCT and the Walsh Hadamard Transform (WHT) [3].

The first part consisted of measuring the transform's decorrelation properties. An 8 x 8 Toeplitz matrix \bar{C}_x was transformed in two dimensions to produce \bar{C}_z . The residual correlation was then computed as follows:

$$r = \frac{1}{N} \frac{1}{(N-1)} \sum_{\substack{i=0 \\ i \neq j}}^{N-1} \sum_{j=0}^{N-1} [C_z(i,j)]^2.$$

The value of ρ was varied from 0 to 1.0. The results are shown in Figure 4. The result shows that both the DCT and WHT approach perfect decorrelation as ρ approaches one. The residual correlation for the SCT and DST grows rapidly for $\rho > .5$. This fact can be explained by examining the basis planes of the transforms (see Figure 5). Both the DCT and WHT have a constant (DC) basis plane. As $\rho \rightarrow 1$, the covariance matrix, \bar{C}_x , becomes a constant matrix. Then for the Walsh and the DCT, \bar{C}_z contains a single nonzero term in the DC position. The DST and SCT do not have constant basis planes. Thus as $\rho \rightarrow 1$ several nonzero terms exist in C_z which results in a large residual correlation. A windowed SCT [2] was also tested with the results shown in Figure 4. Unfortunately, the results show that windowing decreased the SCT performance. This is due to the fact that the condition

$$N \gg \frac{\rho+1}{\rho-1}$$

does not hold for $N=8$ and $0 < \rho < 1.0$.

The second part of the simulation addressed the question of how the different transforms perform with actual image data. Several 512 x 512 images quantized to 8 bits were blocked into 8 x 8 blocks. A two dimensional transform was then applied to each block. The transformed image was then multiplied point by point by a mask matrix. The mask

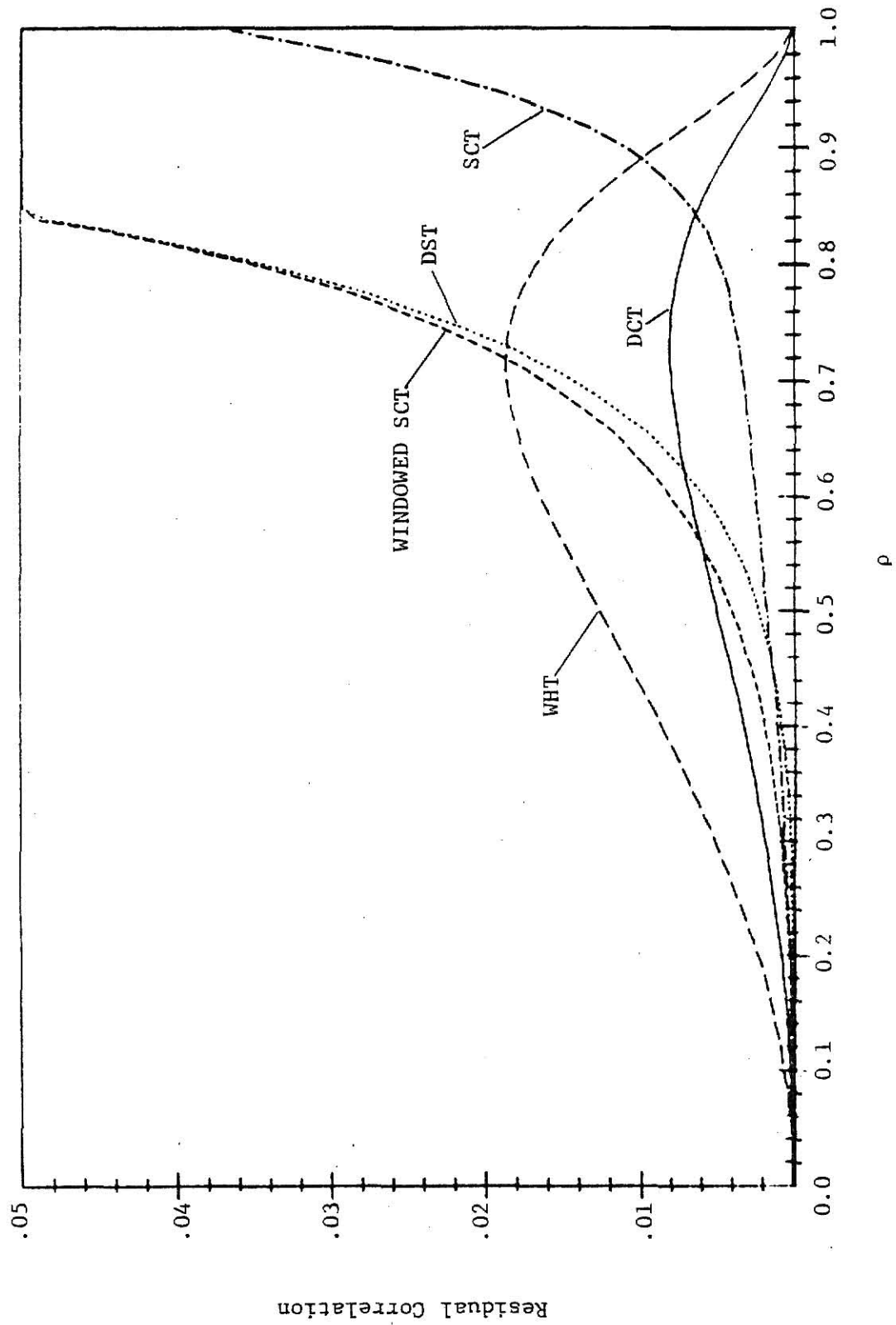
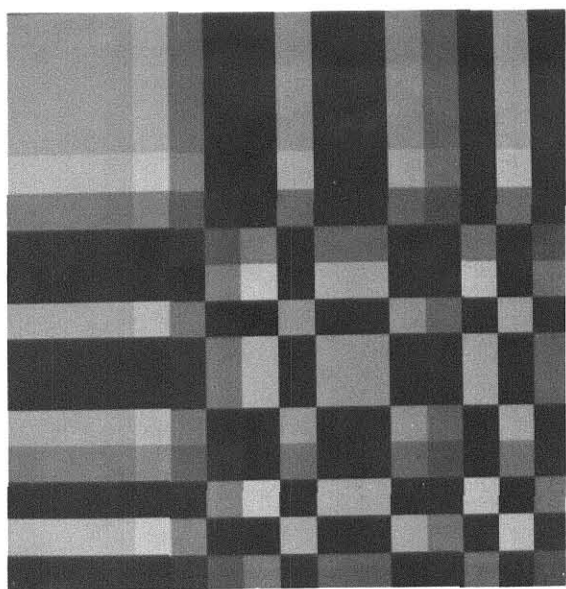
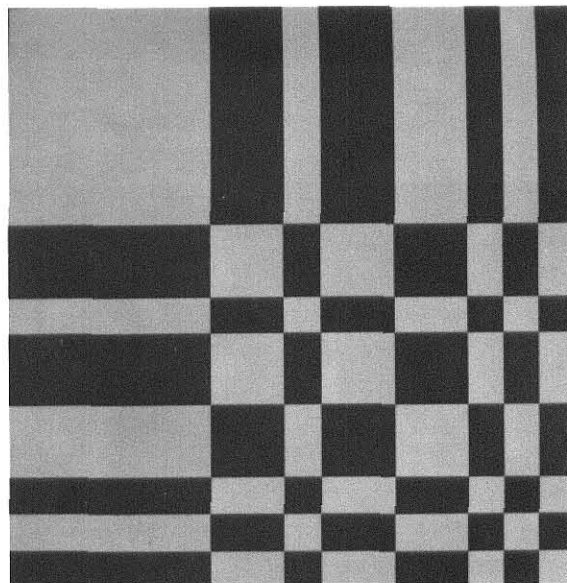


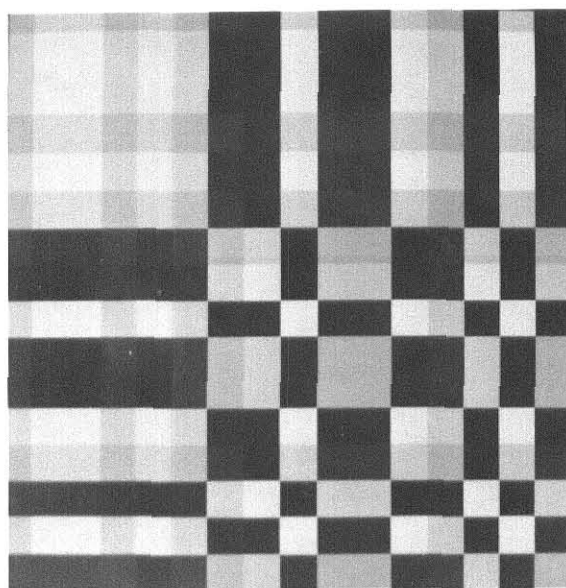
FIGURE 4.



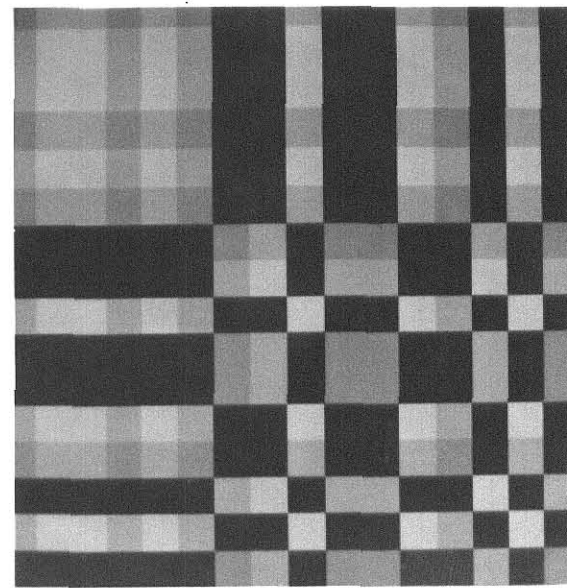
DCT basis planes



WHT basis planes



DST basis planes



SCT basis planes

FIGURE 5
Basis Planes

matrix contained either 0's or 1's. The position of the 1's was selected using the variance criterion using a Toeplitz model with $\rho=.9$. The positions with the highest M variances are the retained coefficients. The masked transform matrix was then inverse-transformed to reconstruct the data. Values larger than 255 were set to 255 and values smaller than 0 were set to 0. All transform calculations were performed in floating point representation to minimize roundoff errors. The results, along with the RMS error, computed as

$$e = \frac{1}{N^2} \sqrt{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} [f(x,y) - g(x,y)]^2}$$

where $f(x,y)$ is the original image and $g(x,y)$ is the reconstructed image, are in Figure 6-10. By looking at the results for $M = 16$ one can see the DST and SCT have larger errors.

Tasto and Wintz [21] have suggested that the largest errors in block coding occur at the boundry. A simulation was used to verify this suggestion. An 8 x 8 Toeplitz matrix with $\rho = .9$ was transformed. The inverse transform was performed using 16 of the 64 coefficients which implies a 4:1 compression. The percent error was then calculated on a point by point basis. This result is shown in Table 1. Clearly the DCT has the lowest endpoint error (15%). The SCT and the Walsh come next with a 24% endpoint error. Note, however, that the SCT endpoint error is very abrupt, whereas the Walsh endpoint error is gradual. The endpoint effect is visible on the SCT reconstructed picture but not on the WHT reconstruction as seen in Figure 6 and 7. The endpoint error for the DST is 46% and is clearly visible in Figure 7. The visual endpoint effect can be reduced for the DST and SCT by subtracting the block mean

11	- 1	- 8	-10	- 8	- 2	6	15
- 1	7	- 1	- 5	- 5	- 3	1	6
- 8	- 1	10	3	- 1	- 3	- 3	- 2
-10	- 5	3	13	5	- 1	- 5	- 8
- 8	- 5	- 1	5	13	3	- 5	-10
- 2	- 3	- 3	- 1	3	10	- 1	- 8
6	1	- 3	- 5	- 5	- 1	7	- 1
15	6	- 2	- 8	-10	- 8	- 1	11

DCT % error $\rho = .9$ M=16

24	2	- 6	- 9	- 7	- 3	1	- 1
2	2	- 5	- 7	- 3	4	12	1
- 6	- 5	6	1	- 0	1	4	- 3
- 9	- 7	1	11	5	- 0	- 3	- 7
- 7	- 3	- 0	5	11	1	- 7	- 9
- 3	4	1	- 0	1	6	- 5	- 6
1	12	4	- 3	- 7	- 5	2	2
- 1	1	- 3	- 7	- 9	- 6	2	24

SCT % error $\rho = .9$ M=16

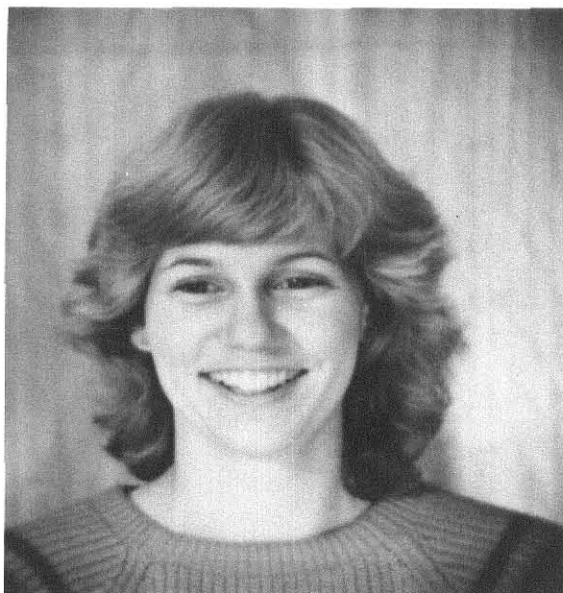
24	10	- 4	-18	1	- 7	-12	-13
10	14	0	-13	4	- 4	-10	-12
- 4	0	7	- 6	9	2	- 4	- 7
-18	-13	- 6	3	16	9	4	1
1	4	9	16	3	- 6	-13	-18
- 7	- 4	2	9	- 6	7	0	- 4
-12	-10	- 4	4	-13	0	14	10
-13	-12	- 7	1	-18	- 4	10	24

WHT % error $\rho = .9$ M=16

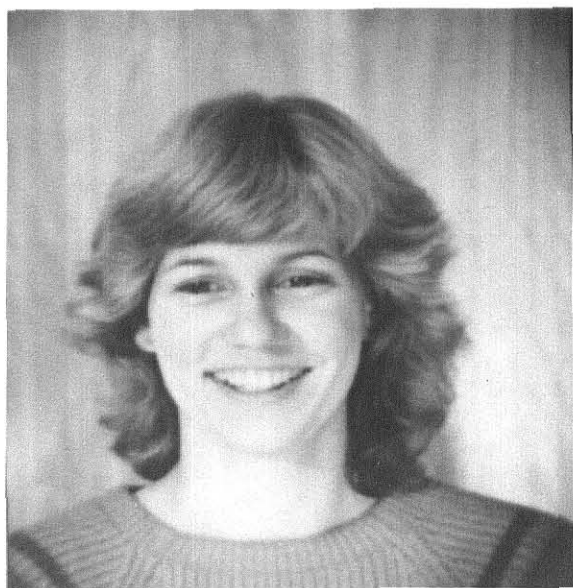
46	16	- 6	-17	-17	- 7	7	22
16	8	- 8	-12	- 5	6	12	7
- 6	- 8	1	- 2	2	7	6	- 7
-17	-12	- 2	11	6	2	- 5	-17
-17	- 5	2	6	11	- 2	-12	-17
- 7	6	7	2	- 2	1	- 8	- 6
7	12	6	- 5	-12	- 8	8	16
22	7	- 7	-17	-17	- 6	16	46

DST % error $\rho = .9$ M=16

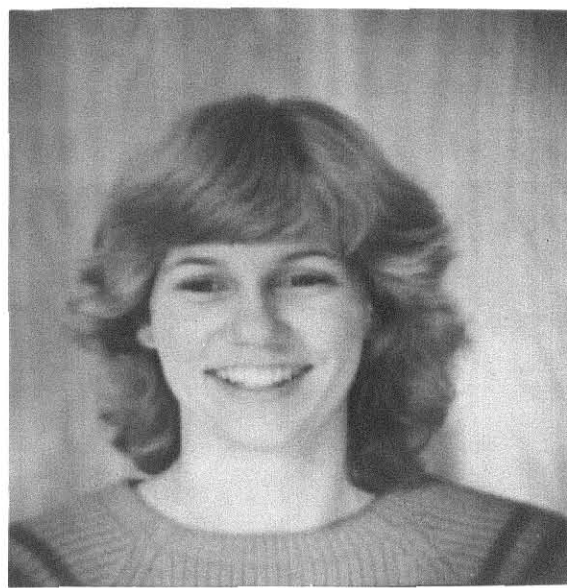
Table 1



Original

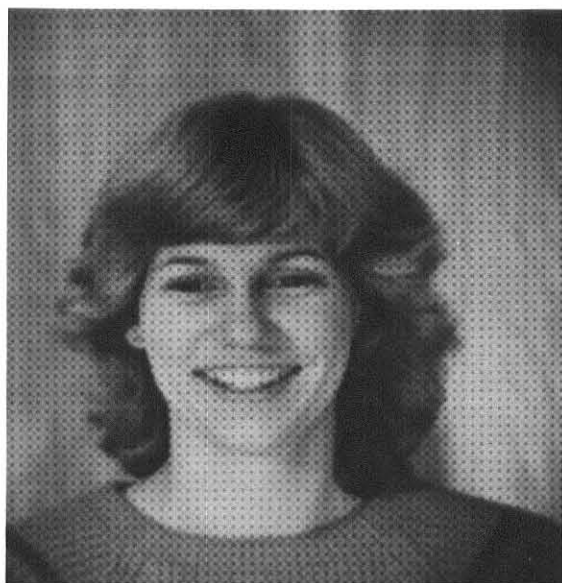


Transformation: DCT
Coefficients retained: 16
RMS error: 3.0

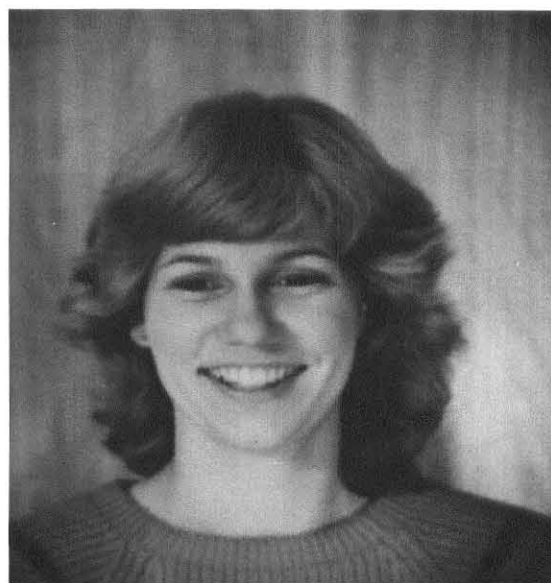


Transformation: WHT
Coefficients retained: 16
RMS error: 3.2

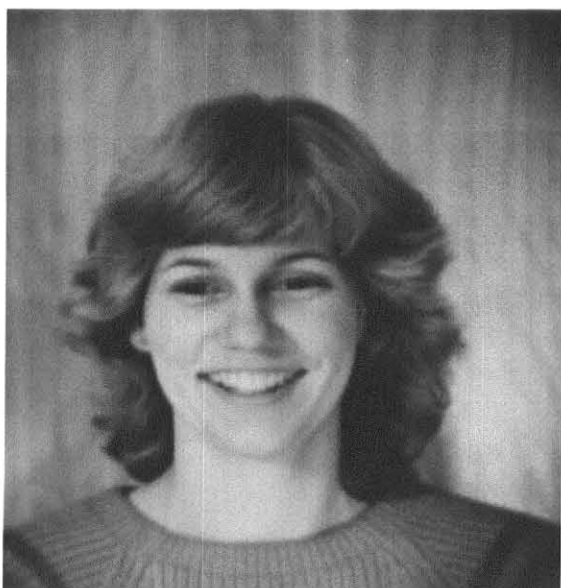
FIGURE 6; 4:1 compression.



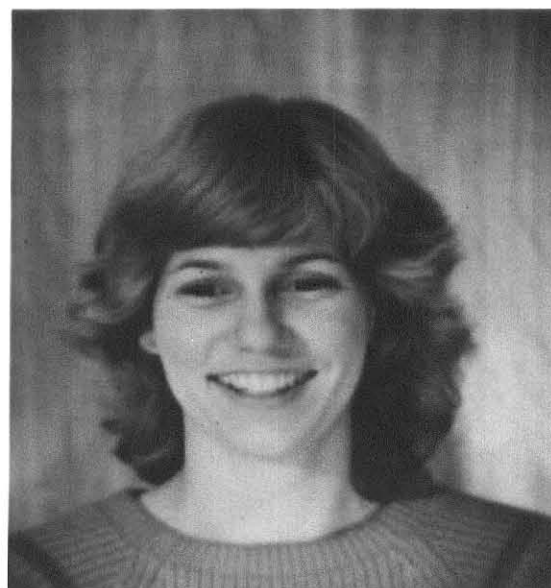
Transformation: DST
Coefficients retained: 16
RMS error: 18.7



Transformation: DST
Coefficients retained: 16 + mean
RMS error: 3.9



Transformation: SCT
Coefficients retained: 16
RMS error: 4.6



Transformation: SCT
Coefficients retained: 16 + mean
RMS error: 3.3

FIGURE 7; 4:1 compression.

before transforming. The mean can be added back after reconstruction. Note, that this requires more computation and reduces the compression rate.

The large endpoint errors for the DST and SCT can be explained by examining the transformation of a constant matrix. The two dimensional transformations of an 8×8 constant matrix are shown in Table 2. Note that the DCT and the Walsh transformation consists of a single nonzero value, whereas the SCT and the DST consist of several nonzero values. This means that the DCT and the WHT have the single basis plane needed to reconstruct a constant matrix, whereas the SCT and the DST require a summation of several basis planes to reconstruct the matrix. When only the lowest frequency basis planes are used to reconstruct the constant image, the DCT and WHT can reconstruct it exactly. The SCT and DST can only approximate the constant plane. By examining the lowest frequency basis planes (Figure 5) of the DST and the SCT we can see that they differ significantly from the constant matrix at the endpoints. It is easily shown that the coefficient for a constant basis plane is the mean of the image. The SCT and DST are distributing the DC energy throughout several basis planes. By computing and retaining the mean, we can reduce the DC energy loss by processing the resulting zero mean blocks.

In the example of Figure 6 and 7, the relatively small RMS errors indicate the correlation between adjacent pixels is large; i.e. $\rho > .9$. Input data in which $\rho > .9$ result in larger RMS errors. Examples of this are shown in Figure 8.

Higher compression rates are easily obtainable if larger errors are acceptable. Simulation results with $M=8$ are shown in Figure 9. The mean information was retained in the SCT and DCT processing.

$$\begin{bmatrix} 8.0 & 0.0 & . & . & 0.0 \\ 0.0 & 0.0 & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0.0 & . & . & . & 0.0 \end{bmatrix}$$

DCT and WHT

$$\begin{bmatrix} 7.9 & 0.0 & .62 & 0.0 & .62 & 0.0 & .62 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ .62 & 0.0 & .05 & 0.0 & .05 & 0.0 & .05 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ .62 & 0.0 & .05 & 0.0 & .05 & 0.0 & .05 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ .62 & 0.0 & .05 & 0.0 & .05 & 0.0 & .05 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

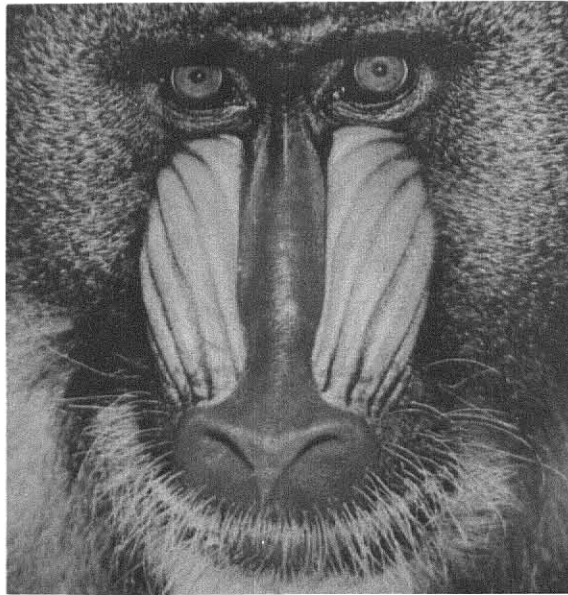
SCT

$$\begin{bmatrix} 7.1 & 0.0 & 2.2 & 0.0 & 1.1 & 0.0 & .46 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 2.2 & 0.0 & .67 & 0.0 & .32 & 0.0 & .14 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 1.1 & 0.0 & .32 & 0.0 & .16 & 0.0 & .07 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ .46 & 0.0 & .14 & 0.0 & .07 & 0.0 & .03 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

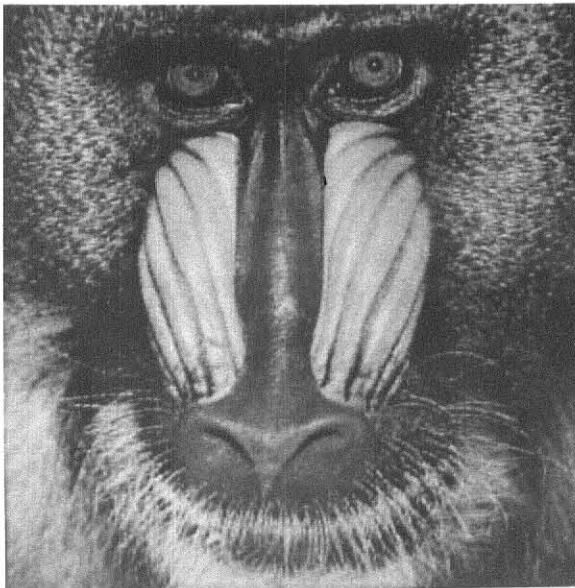
DST

Table 2

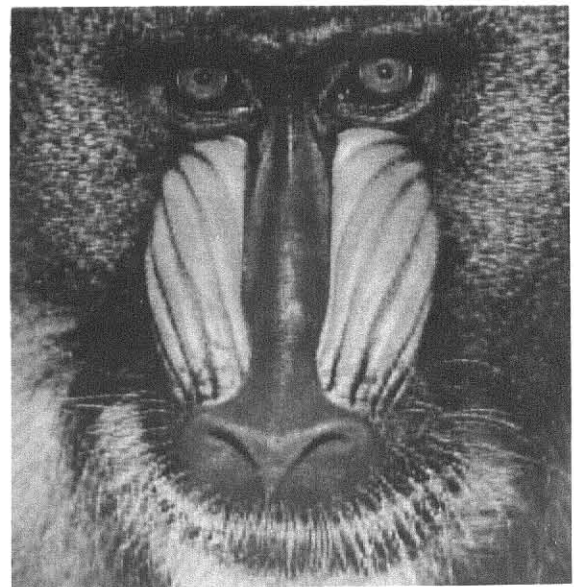
Transformation of a Constant Matrix



Original

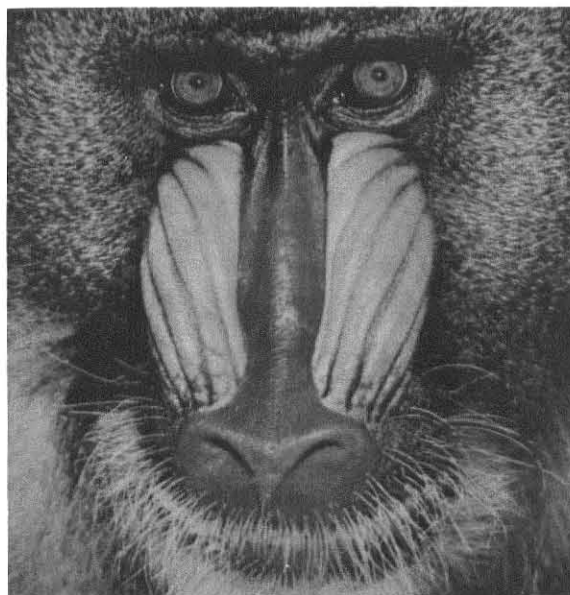


Transformation: DCT
Coefficients retained: 16
RMS error: 15.4

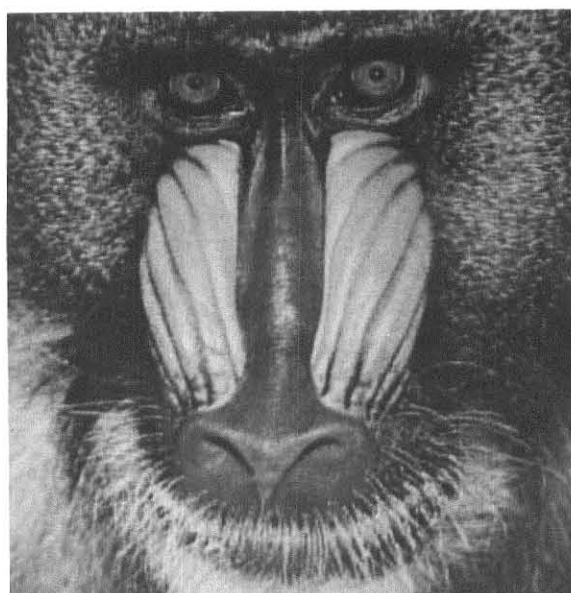


Transformation: WHT
Coefficients retained: 16
RMS error: 17.7

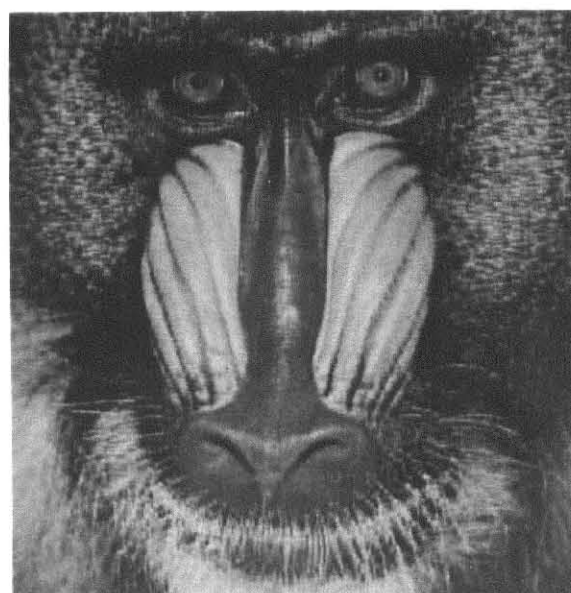
FIGURE 8; 4:1 compression.



Original

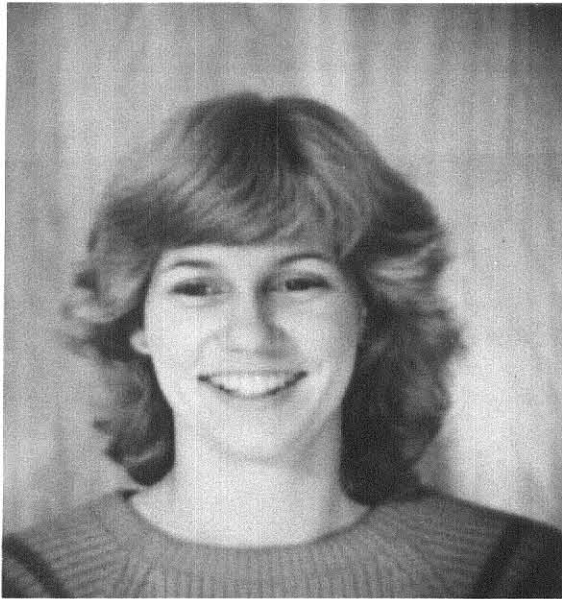


Transformation: DST
Coefficients retained: 16 + mean
RMS error: 16.4

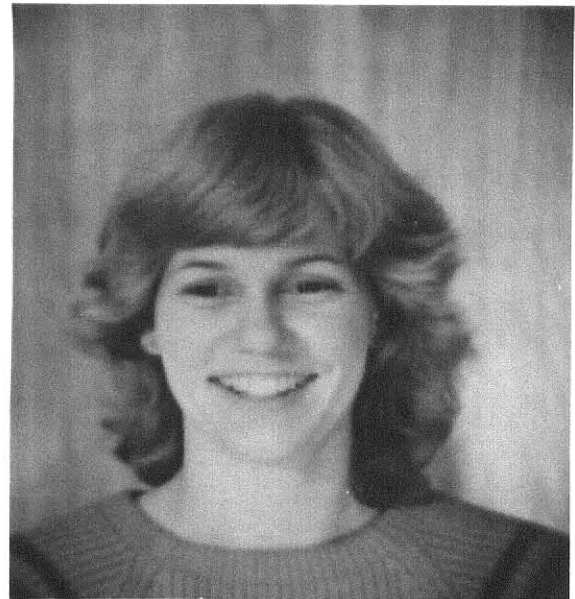


Transformation: SCT
Coefficients retained: 16 + mean
RMS error: 17.4

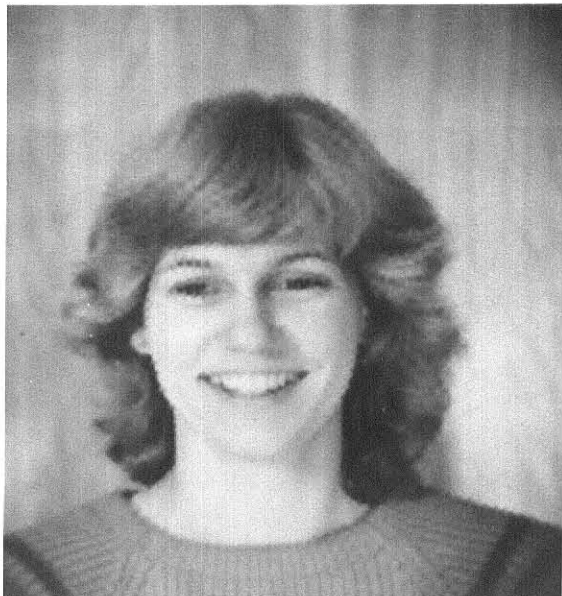
FIGURE 8 continued; 4:1 compression.



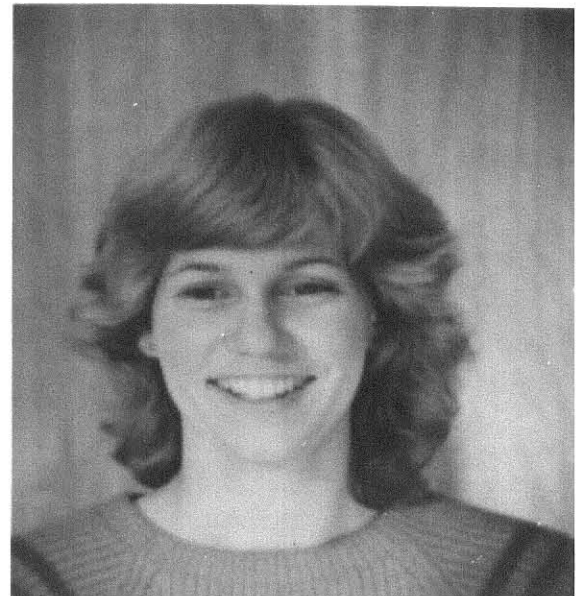
Transformation: DCT
Coefficients retained: 8
RMS error: 4.5



Transformation: WHT
Coefficients retained: 8
RMS error: 6.0

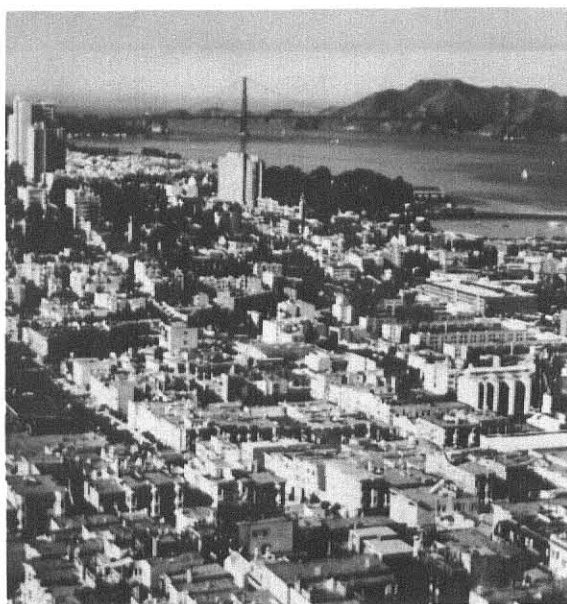


Transformation: DST
Coefficients retained: 8 + mean
RMS error: 5.9

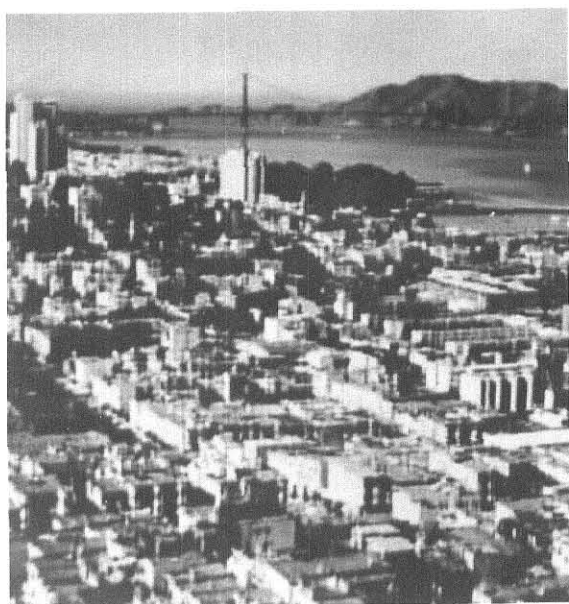


Transformation: SCT
Coefficients retained: 8 + mean
RMS error: 4.7

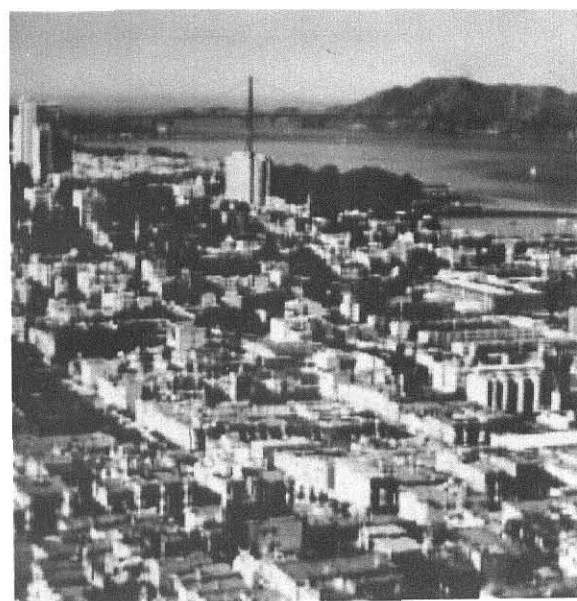
FIGURE 9; 8:1 compression.



Original

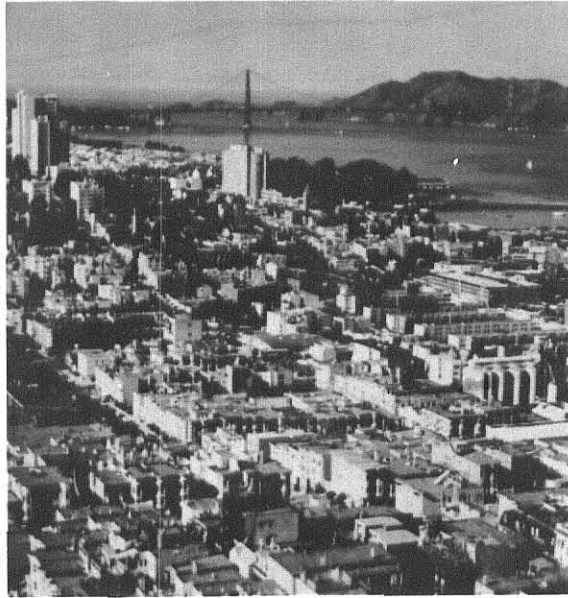


Transformation: DCT
Coefficients retained: 8
RMS error: 26.0

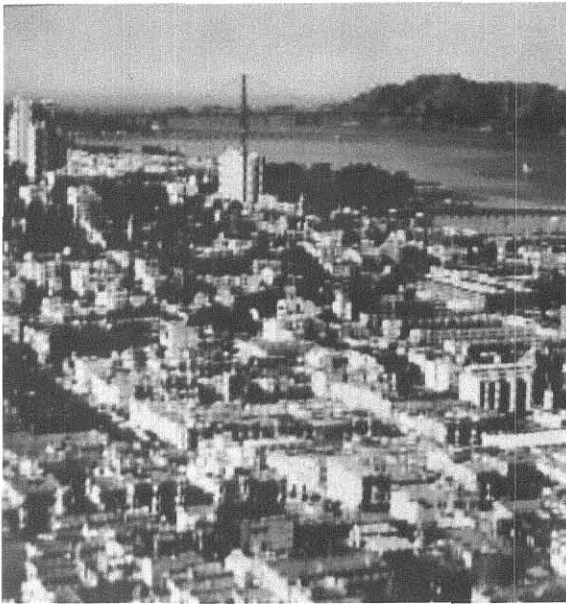


Transformation: WHT
Coefficients retained: 8
RMS error: 28.2

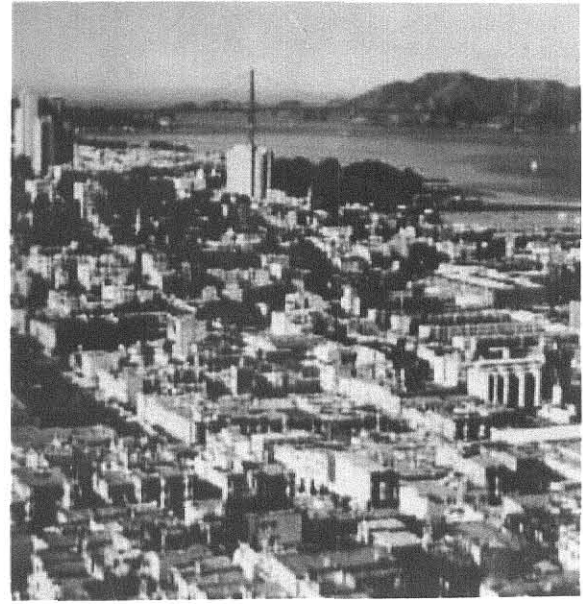
FIGURE 9 continued; 8:1 compression.



Original



Transformation: DST
Coefficients retained: 8 + mean
RMS error: 28.4



Transformation: SCT
Coefficients retained: 8 + mean
RMS error: 26.3

FIGURE 9 continued; 8:1 compression.

Chapter V

Conclusions

Recently, there has been extensive use of orthogonal transforms for many signal processing applications. This is a tutorial paper on the Discrete Cosine Transform [1]. Many of published uses of the DCT are presented. Different algorithms for computing the DCT have been examined. A comparison of different orthogonal transformations when used for image compression shows the DCT to perform better than the Discrete Sine Transform [4], the Symmetric Cosine Transform [2], and the Walsh Hadamard Transform. This result, along with the comparison conducted by Kerke and Solenki [15], suggest that the DCT is currently the best fast transform for the transform coding of images.

References

- [1] N. Ahmed, T. Natarajan, and K. Rao, "Discrete Cosine Transform," IEEE Trans. Comput. (Corresp.), Vol. C-23, pp. 90-93, Jan 1974.
- [2] H. Kitajima, "A Symmetric Cosine Transform," IEEE Trans. Comput., vol. C-29, pp. 317-323, April 1980.
- [3] N. Ahmed, K. Rao, Orthogonal Transforms for Digital Signal Processing, New York: Springer Verlag, 1975.
- [4] A. Jain, "A Fast Karhunen-Loeve Transform for a Class of Random Processes," IEEE Trans. Commun., vol. COM-24, pp. 1023-1029, Sept. 1976.
- [5] W. D. Ray and R. M. Driver, "Further Decomposition of the Karhunen-Loeve Series Representation of a Stationary Random Process," IEEE Trans. Information Theory, vol IT-16, pp. 663-668, November 1970.
- [6] R. Haralick, "A Storage Efficient Way to Implement the Discrete Cosine Transform," IEEE Trans. Comput., vol. C-25, pp. 764-765, July 1976.
- [7] B. Tseng and W. Miller, "On Computing the Discrete Cosine Transform," IEEE Trans. Commun., vol. C-27, pp. 966-968, October 1978.
- [8] W. Chen, C. Harrison and S. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," IEEE Trans. Commun., vol. COM-25, pp. 1004-1009, Sept. 1977.
- [9] M. Narasimha and A. Peterson, "On the Computation of the Discrete Cosine Transform," IEEE Trans. Commun., vol. COM-26, pp. 934-936, June 1978.
- [10] M. Wagh and H. Ganesh, "A New Algorithm for the Discrete Cosine Transform of Arbitrary Number of Points," IEEE Trans. Comput., vol. C-29, pp. 269-277, April 1980.
- [11] S. Dyer, N. Ahmed, and D. Hummels, "Computation of the Discrete Cosine Transform via the Arcsine Transform," IEEE Conference on Acoustics, Speech and Signal Processing, Part I, pp. 321-234, April 1980.
- [12] F. Kamanger and K. Rao, "Fast Algorithm for the 2d-Discrete Cosine Transform," IEEE Conference on Acoustics, Speech and Signal Processing, Part I, pp. 206-209, April 1980.
- [13] J. Makhoul, "A Fast Cosine Transform in One and Two Dimensions," IEEE Trans. Acoustics, Speech, Signal Processing, vol. ASSP-28, pp. 27-34, February 1980.

- [14] H. Nussbaumer, "Fast Multidimensional Discrete Cosine Transform," IBM Technical Disclosure Bulletin, vol. 23, pp. 1976-1981, October 1980.
- [15] H. Kekre and J. Solenki, "Comparative Performance of Various Trigonometric Unitary Transform for Transform Image Coding," Int. J. Electronics, vol. 44, pp. 305-315, March 1978.
- [16] T. Natarajan and N. Ahmed, "Performance Evaluation for Transform Coding Using a Nonseparable Covariance Model," IEEE Trans. Commun., vol. COM-26, pp. 310-312, Feb. 1978.
- [17] N. Griswold and R. Haralick, "A Critical Comparison of Fast Transforms for Image Data Compression," Proceeding of the SPIE Advances in Image Transmission Techniques, vol. 87, 1976.
- [18] J. Max, "Quantizing for Minimum Distortion," IRE Trans. Info. Theory, vol. IT-6, pp. 7-12, 1960.
- [19] R. Gonzales and P. Wintz, Digital Image Processing, Massachusetts: Addison Wesley, 1977.
- [20] W. Pratt, Digital Image Processing, New York: Wiley and Sons, 1978.
- [21] M. Tasto and P. Wintz, "Note on the Error Signal of Block Quantizers," IEEE Trans. Commun., vol. COM-21, March 1973.
- [22] P. Noll, "Transform Coding of Speech," Conf. Rec. Int. Conf. Commun., ICC '77, vol. 1, p. 306-309, 1977.
- [23] R. Zelinski and P. Noll, "Adaptive Transform Coding of Speech Signals," IEEE Trans. Acoustic Signal Processing, vol. ASSP-25, pp. 299-309, August 1977.
- [24] M. Berouti and J. Makoul, "An Embedded-Code Multirate Speech Transform Coder," Rec. IEEE Int. Conf. Acoustics, Speech Signal Process., ICASSP 80, Proc., vol. 2, pp. 356-359, 1980.
- [25] M. Ashoui, "Linear Prediction of Transformed Speech," IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 77-80, 1979.
- [26] L. Bergeron, A. Goldberg, S. Kwon and M. Miller, "A Robust, Adaptive Transform Coder for 9.6 kb/s Speech Transmission," ICASSP 80 Proceedings, IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 344-347, Part II, 1980.
- [27] H. Keshwen and M. Srinath, "Two Dimensional Interpolative Models in Enhancement of Noisy Images," NTC '77 Conference Record, pp. 49:4/1-5, 1977.
- [28] H. Keshaven and M. Srinath, "Enhancement of Noisy Images Using an Interpolative Model in Two Dimensions," IEEE Trans. Systems, Man, and Cybernetics, vol. SMC-8, pp. 247-257, April 1978.

- [29] M. Flickner and N. Ahmed, "A Derivation of the Discrete Cosine Transform", submitted for publication, IEEE Proceedings.

Acknowledgment

I wish to thank my friends and family for their support. I also wish to express my appreciation to Dr. Virgil Wallentine and Dr. James Tracey for being members of my graduate committee. I particularly want to thank my major advisor, Dr. Nasir Ahmed, for his invaluable support and guidance. Finally I wish to thank Peggy Grosh for allowing me to use her picture as data for my thesis.

Appendix A

A Fast Discrete Cosine Routine

```

C*****40
C
C      FAST DISCRETE COSINE TRANSFORMATION
C
C      DG FORTRAN 5 SOURCE FILENAME:          FDCT.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              MAY 07, 1981    MYRON FLICKNER
C      01.0              NOV 06, 1981    MYRON FLICKNER
C
C*****
C
C      CALLING SEQUENCE
C
C          CALL FDCT ( X , N, INV )
C
C      PURPOSE
C
C          THE ROUTINE IMPLEMENTS A FAST DISCRETE COSINE
C          TRANSFORMATION USING AN ALGORITHM DEFINED BY
C          JOHN MAKHOUL IN THE FEBRUARY 1980 ASSP TRANSACTION
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          FFS      -      IEEE SIGNAL PROCESSING ROUTINE
C          FFA      -      IEEE SIGNAL PROCESSING ROUTINE
C
C      ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C          X      -      VECTOR TO BE TRANSFORMED
C          N      -      NUMBER OF ELEMENTS TO BE TRANSFORMED
C                      (POWER OF 2)
C
C      ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C          X      -      TRANSFORMED VECTOR
C
C*****
C
C      NOTE 1: This subroutine makes no checks on the validity
C              of the data supplied by the calling routine.
C
C      The subroutine FFS and FFA are subroutines from
C      the Programs for Digital Signal Processing, New York:
C      IEEE Press.
C
C*****
C
C      SUBROUTINE FDCT( X, N, INV )
C      REAL X(N), VR(2050), PI, PI2, RN, THETA
C      COMPLEX V(1025), TEMP1, TEMP2, W
C      EQUIVALENCE ( VR(1), V(1) )
C      DATA PI/3.141592654/, PI2/6.283185307/
C      IF( N .GT. 2048 ) STOP ' N TOO LARGE FOR FDCT '

```

```

RN = FLOAT( N )
N2 = N/2
N4 = N/4
I = 1
M = N
IF( INV .NE. 0 ) GO TO 30
C = SQRT( 2.0/RN )

C
C REORDER DATA
C
C
C      DO 10 K=1,N2
C      VR(K) = X(I)
C      VR(K+N2) = X(M)
C      I = I + 2
C      M = M - 2
10  CONTINUE
C
C DO N POINT REAL FFT USING N/2 POINT COMPLEX FFT
C THIS IS DONE BY FFA. FFA IS IN THE IEEE PROGRAMS FOR DIGITAL
C SIGNAL PROCESSING.
C
C CALL FFA( V, N )
C
C NOW FFT IS DONE. NOTE WE ONLY CALCULATED THE FIRST N2+1
C POINTS. THE DFT OF A REAL SEQUENCE IS HERMITIAN SO WE DO NOT
C NEED THE OTHER POINTS.
C
C NOW GET THE DCT COEFFICIENTS
C
C X(1) = VR(1)/SQRT( RN )
C
C      DO 20 K=1,N2
C      W = CEXP( CMPLX( 0.0, -FLOAT(K)*PI/(2.0*RN) ) )
C      V(K+1) = V(K+1)*W
C      X(K+1) = C*REAL( V(K+1) )
C      X(N-K+1) = -C*AIMAG( V(K+1) )
20  CONTINUE
C
C RETURN
C
C 30 CONTINUE
C
C DO THE INVERSE DCT
C
C C = 1.0/SQRT( 2.0/RN )
C V(1) = CMPLX( X(1)*SQRT(RN), 0.0 )
C
C      DO 35 K=1,N2
C      W = CEXP( CMPLX( 0.0, FLOAT(K)*PI/(2.0*RN) ) )
C      V(K+1) = W*CMPLX( C*X(K+1), -C*X(N-K+1) )
35  CONTINUE
C
C USE N/2+1 POINT COMPLEX IFFT TO GET N POINT REAL
C IFFT. THIS IS DONE IN FFS. FFS IS ALSO A IEEE PROGRAM.
C
C CALL FFS( V, N )

```


C
C
C

REORDER DATA

DO 45 K=1,N2
X(I) = VR(K)
X(M) = VR(K+N2)
I = I + 2
M = M - 2
CONTINUE

45
C

RETURN
END

Appendix B

Programs Used for Simulation

```

C*****
C
C      DCT MATRIX GENERATION
C
C      DG FORTRAN 5 SOURCE FILENAME:          DCTMAT.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              JAN 20, 1982    MYRON FLICKNER
C
C*****
C
C      .PURPOSE
C
C          THE ROUTINE GENERATES AN N X N DCT MATRIX
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          WRITR
C          CHECK
C          OPENW
C
C*****
C
C      use this space for added information unique to this routine
C
C*****
C
C      DOUBLE PRECISION  A, THETA, PI, TEMP
C      REAL X(64,64)
C      ACCEPT ' ENTER VALUE OF N ? ', N
C      CALL OPENW( 0, ' OUTPUT FILENAME ? ', N*4, F )
C      A = DSQRT( 1.0/DFLOAT( N ) )
C      PI = DATAN(1.0)*4.0
C
C          DO 30 J=1,N
C
C              DO 20 I=1,N
C                  THETA = DFLOAT( PI/DFLOAT(2*N)*
1                  DFLOAT(J-1)*DFLOAT(2*(I-1)+1) )
C                  TEMP = A*DCOS(THETA)
C                  X(I,J) = SNGL( TEMP )
20                  CONTINUE
C
C                  A = DSQRT( 2.0/DFLOAT( N ) )
C                  CALL WRITR( 0, J, X(1,J), 1, IERR )
C                  CALL CHECK( IERR )
30                  CONTINUE
C
C      STOP
C      END

```

```

C*****
C
C      ENDPOINT ERROR SIMULATION
C
C      DG FORTRAN 5 SOURCE FILENAME:          ENDPOINT.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              JAN 28, 1982    MYRON FLICKNER
C
C*****
C
C      PURPOSE
C
C          THE ROUTINE COMPUTES THE PERCENT ENDPOINT
C          ERROR FOR ATRANSFORM.  INPUTS ARE THE MASK FILE,
C          AND THE PARAMETER RHO FOR THE TOEPLIZT MODEL.
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          ASK
C          OPENR
C          READR
C          TPLZMAT
C          T88
C          RESET
C
C*****
C
C      use this space for added information unique to this routine
C
C*****
C
C      REAL X(8,8), Y(8,8), DIFF(8), MASK(8,8), T(8,8)
C      LOGICAL YESNO
5      CONTINUE
      ITTI = 11
      LPT = 10
      ITTO = 10
      CALL ASK( ' OUTPUT TO LINE PRINTER ? ', YESNO )
      IF( YESNO ) LPT = 12
      CALL READT( 0, T, 8 )
      CALL OPENR( 0, ' MASK FILENAME ? ', 256, F )
      CALL READR( 0, 1, MASK, 1, ICNT, IERR )
C
C      ACCEPT ' ENTER THE VALUE OF RHO ? ', RHO
      CALL TPLZMAT( X, RHO, 8 )
      CALL TPLZMAT( Y, RHO, 8 )
      CALL T88( X, T, 0 )
C
C          DO 15 I=1,8
C          DO 15 J=1,8
C          X(J,I) = X(J,I)*MASK(J,I)
15      CONTINUE
C

```

CALL T88(X, T, 1)

C

DO 25 I=1,8

DO 20 J=1,8

DIFF(J) = (Y(J,I) - X(J,I))/Y(J,I)

20

CONTINUE

WRITE(LPT, 1) (DIFF(J), J=1,8)

1

FORMAT(' ', 8F9.2)

25

CONTINUE

C

CALL RESET

CALL ASK(' RE-EXECUTE ? ', YESNO)

IF(YESNO) GO TO 5

STOP

END

```

C*****
C
C      READ IN AN N X N TRANSFORM MATRIX FILE
C
C      DG FORTRAN 5 SOURCE FILENAME:          READT.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              JAN 28, 1982    MYRON FLICKNER
C
C*****
C
C      CALLING SEQUENCE
C
C          CALL READT( ICHAN, T, N )
C
C      PURPOSE
C
C          THE ROUTINE OPENS AND READS IN AN N X N
C          TRANSFORM MATRIX FILE.
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          OPENR
C          READR
C          CHECK
C
C      ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C          ICHAN  -      UNUSED LOGICAL CHANNEL
C          N      -      SIZE OF TRANSFORM
C
C      ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C          T      -      THE N X N TRANSFORM MATRIX
C
C*****
C
C      NOTE 1: This subroutine makes no checks on the validity
C              of the data supplied by the calling routine.
C
C      NOTE 2: Argument(s) supplied by the calling routine are
C              not modified by this subroutine.
C
C*****
C
C      SUBROUTINE READT( ICHAN, T, N )
C      REAL T(N,N)
C
C      CALL OPENR( ICHAN, ' TRANSFORM MATRIX FILENAME ? ', N*N*4, F )
C      CALL READR( ICHAN, 1, T, 1, ICNT, IERR )
C      CALL CHECK( IERR )
C      CLOSE ICHAN
C      RETURN
C      END

```

```

C*****
C
C      COMPUTE THE RESIDUAL CORRELATION AND ENTROPY
C
C      DG FORTRAN 5 SOURCE FILENAME:          RESID.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              JAN 28, 1982    MYRON FLICKNER
C
C*****
C
C      PURPOSE
C
C          THE ROUTINE COMPUTES THE RESIDUAL CORRELATION
C          AND ENTROPY FOR A TOEPLITZ MATRIX FOR A GIVEN
C          COMPRESSION RATE.  THE PARAMETER RHO IS SWEEPED
C          THROUGH A RANGE OF VALUES SPECIFIED BY THE USER.
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          READT
C          OPENW
C          RUNTIME
C          TPLZMAT
C          T88
C          WRITR
C
C*****
C
C      use this space for added information unique to this routine
C
C*****
C
C      PARAMETER N = 8
C      REAL X(N,N), T(N,N)
C      DOUBLE PRECISION SUM, ENTROPY
C
C      CALL READT( 0, T, N )
C      ACCEPT ' STARTING RHO ? ', RHO1
C      ACCEPT ' ENDING RHO ? ', RHO2
C      ACCEPT ' NUMBER OF POINTS ? ', NPTS
C      CALL OPENW( 1, ' RESIDUAL FILENAME ? ', 4, F )
C      CALL OPENW( 2, ' ENTROPY FILENAME ? ', 4, F )
C
C      CALL RUNTIME
C      DELTA = (RHO2 - RHO1)/ FLOAT(NPTS)
C      RHO = RHO1
C
C          DO 40 I1=1,NPTS
C              CALL TPLZMAT( X, RHO, N )
C              CALL T88( X, T, 0 )
C              SUM = 0.0
C
C          DO 20 J=1,N

```

```

C
DO 10 I=1,N
IF(I.EQ.J) GO TO 10
SUM = SUM + X(I,J)*X(I,J)
10 CONTINUE
C
CONTINUE
C
SUM = SUM/(FLOAT(N)*FLOAT(N-1) )
ENTROPY = 0.0
C
DO 30 I=1,N
ENTROPY = ENTROPY + ALOG( X(I,I) )
30 CONTINUE
C
CALL WRITR( 1, I1, SUM, 1, IERR )
CALL WRITR( 2, I1, ENTROPY, 1, IERR )
RHO = RHO + DELTA
40 CONTINUE
C
CALL RUNTIME
STOP
END

```



```

C*****
C
C      SCT MATRIX GENERATION
C
C      DG FORTRAN 5 SOURCE FILENAME:          SCTMAT.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          -
C      00.0              JAN 20, 1982    MYRON FLICKNER
C
C*****
C
C      PURPOSE
C
C          THE ROUTINE GENERATES AN N X N SCT MATRIX
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          WRITR
C          CHECK
C          OPENW
C
C*****
C
C      use this space for added information unique to this routine
C
C*****
C
C      DOUBLE PRECISION  A, THETA, PI, TEMP
C      REAL X(64,64)
C      ACCEPT ' ENTER VALUE OF N ? ', N
C      CALL OPENW( 0, ' OUTPUT FILENAME ? ', N*4, F )
C      A = DSQRT( 2.0/DFLOAT( N-1 ) )
C      PI = DATAN(1.0)*4.0
C
C          DO 30 J=1,N
C
C              DO 20 I=1,N
C                  THETA = DFLOAT( (I-1)*(J-1) )*PI/DFLOAT(N-1)
C                  TEMP = A*DCOS(THETA)
C                  IF( J .EQ. 1 .OR. J .EQ. N ) TEMP = TEMP*DSQRT(.5)
C                  IF( I .EQ. 1 .OR. I .EQ. N ) TEMP = TEMP*DSQRT(.5)
C                  X(I,J) = SNGL( TEMP )
C
C20             CONTINUE
C
C                  CALL WRITR( 0, J, X(1,J), 1, IERR )
C                  CALL CHECK( IERR )
C
C30             CONTINUE
C
C      STOP
C      END

```



```
                SUM=0.0
C
                DO 10 J=1,8
                SUM=X(J)*T(J,I)+SUM
10             CONTINUE
C
                Y(I)=SUM
20             CONTINUE
C
                GO TO 60
C
                compute inverse transform
C
30             CONTINUE
C
                DO 50 I=1,8
                SUM=0.0
C
                DO 40 J=1,8
                SUM=X(J)*T(I,J)+SUM
40             CONTINUE
C
                Y(I)=SUM
50             CONTINUE
C
60             CONTINUE
C
                DO 70 I=1,8
                X(I)=Y(I)
70             CONTINUE
C
                RETURN
                END
```

```

C*****
C
C      TWO-DIMENSIONAL TRANSFORMATIONS OF AN 8 X 8 MATRIX
C
C      DG FORTRAN 5 SOURCE FILENAME:          T88.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              JAN 28, 1982    MYRON FLICKNER
C
C*****
C
C      CALLING SEQUENCE
C
C          CALL T88( X, T, INV )
C
C      PURPOSE
C
C          THE ROUTINE COMPUTES THE 2-DIMENSIONAL
C          TRANSFORMATION T OF AN 8 X 8 MATRIX
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          T8
C
C      ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C          X          -          THE DATA MATRIX TO BE TRANSFORMED
C          T          -          THE TRANSFORMATION MATRIX
C          INV        -          FORWARD INVERSE FLAG
C                               INV .EQ. 0 -> FORWARD TRANSFORM
C                               INV .NE. 0 -> INVERSE TRANSFORM
C
C      ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C          X          -          THE TRANSFORMED MATRIX
C
C*****
C
C      NOTE 1: This subroutine makes no checks on the validity
C              of the data supplied by the calling routine.
C
C*****
C
C      SUBROUTINE T88( X, T, INV )
C      REAL X(8,8), T(8,8), Y(8)
C
C      do transform by image rows
C
C          DO 10 I=1,8
C          CALL T8( X(1,I), T, INV )
10      CONTINUE
C
C      do transform by image columns

```

```
C
      DO 40 I=1,8
C
      DO 20 J=1,8
      Y(J) = X(I,J)
20    CONTINUE
C
      CALL T8( Y, T, INV )
C
      DO 30 J=1,8
      X(I,J) = Y(J)
30    CONTINUE
C
40    CONTINUE
C
      RETURN
      END
```

```

C*****
C
C      TOEPLIZT MODEL VARIANCE CRITERIA
C
C      DG FORTRAN 5 SOURCE FILENAME:          TOEPLIZT.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              JAN 28, 1982    MYRON FLICKNER
C
C*****
C
C      PURPOSE
C
C          THE ROUTINE ALLOWS FOR THE CALCULATION OF THE
C          MATRIX POSITIONS TO SAVE FOR A GIVEN NUMBER OF
C          RETAINED COEFFICIENTS.
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          READT
C          TPLZMAT
C          OPENW
C          WRITR
C          RESET
C          ASK
C
C*****
C
C      use this space for added information unique to this routine
C
C*****
C
C      PARAMETER LPT=12,ITTI=11,ITTO=10
C      REAL X(8,8), Y(8,8), T(8,8)
C      INTEGER ROW(64), COLUMN(64)
C      LOGICAL YESNO
C
C
C100  CONTINUE
C      CALL READT( 0, T, 8 )
C      ACCEPT ' ENTER THE VALUE OF RHO ? ',RHO
C      CALL TPLZMAT( X, RHO, 8 )
C      WRITE(LPT,3)
C3    FORMAT(////,'1',T50,'THE TOEPLIZT MATRIX',///)
C
C          DO 1020 J=1,8
C          WRITE(LPT,1) (X(J,K) K=1,8)
C          FORMAT(' ',8(G15.7))
C1    1020 CONTINUE
C
C      CALL T88( X, T, 0 )
C      WRITE(LPT,4)
C4    FORMAT(//////////,' ',T50,'THE TRANSFORMED MATRIX '///)
C
C          DO 1030 J=1,8

```

```

WRITE(LPT,1) (X(J,K) K=1,8)
1030 CONTINUE
C
WRITE(LPT,5)
5 FORMAT(//////////, " ", T50, "THE VARIANCE MATRIX "////)
C
DO 1040 J=1,8
C
DO 1040 K=1,8
Y(J,K)=X(J,J)*X(K,K)
1040 CONTINUE
C
DO 1050 J=1,8
WRITE(LPT,1) (Y(J,K) K=1,8)
1050 CONTINUE
C
CALL OPENW( 1, ' OUTPUT MASK FILE ? ', 256, F )
ACCEPT ' NUMBER OF COEFFICIENT TO RETAIN ? ', NUMBER
C
C sort the variances
C
DO 70 K=1,NUMBER
YMAX = 0.0
C
DO 60 J=1,8
DO 60 I=1,8
IF( YMAX .GT. Y(I,J) ) GO TO 55
YMAX = Y(I,J)
ROW(K) = I
COLUMN(K) = J
55 CONTINUE
60 CONTINUE
C
Y( ROW(K), COLUMN(K) ) = 0.0
70 CONTINUE
C
DO 80 J=1,8
DO 80 I=1,8
Y(I,J) = 0.0
80 CONTINUE
C
C save the largest 'NUMBER' of variances
C
DO 90 K=1,NUMBER
Y( ROW(K), COLUMN(K) ) = 1.0
90 CONTINUE
C
CALL WRITR( 1, 1, Y, 1, IERR )
CALL RESET
CALL ASK( ' RE-EXECUTE TOEPLIZT ? ', YESNO )
IF( YESNO ) GO TO 100
STOP
END

```

```

C*****
C
C      GENERATE A TOEPLITZ MATRIX
C
C      DG FORTRAN 5 SOURCE FILENAME:          TPLZMAT.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              JAN 28, 1982    MYRON FLICKNER
C
C*****
C
C      CALLING SEQUENCE
C
C          CALL TPLZMAT( X, RHO, L )
C
C      PURPOSE
C
C          THE ROUTINE GENERATES A FIRST-ORDER MARKOV
C          COVARIANCE MATRIX OF SIZE L X L AND PARAMETER
C          RHO.
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          NONE
C
C      ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
C
C          RHO      -      PARAMETER OF GENERATION
C          L        -      SIZE OF MATRIX
C
C      ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE
C
C          X        -      GENERATED MATRIX
C
C*****
C
C      NOTE 1: This subroutine makes no checks on the validity
C              of the data supplied by the calling routine.
C
C      NOTE 2: Argument(s) supplied by the calling routine are
C              not modified by this subroutine.
C
C*****
C
C      SUBROUTINE TPLZMAT( X, RHO, L )
C      REAL X(L,L), RHO
C
C          DO 20 J=1,L
C              N=1-J
C                  DO 10 K=1,L
C                      X(J,K)=RHO**ABS(N)
C                      N=N+1
C                  CONTINUE
10
C

```


20
C

CONTINUE

RETURN
END

```

C*****
C
C      IMAGE COMPRESSION TRANSFORMATIONS
C
C      DG FORTRAN 5 SOURCE FILENAME:          TRANSFORM.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              DEC 12, 1980    MYRON FLICKNER
C      01.0              JAN 28, 1982    MYRON FLICKNER
C
C*****
C
C      PURPOSE
C
C      THE ROUTINE COMPUTES THE 2-DIMENSIONAL TRANSFORMATION
C      OF THE 8 X 8 BLOCKS OF AN IMAGE.  THE RESULTING TRANSFORM
C      COEFFICIENTS ARE THEN MASKED BY A MASK MATRIX.  THE INVERSE
C      TRANSFORM IS THEN COMPUTED TO RECONSTRUCT THE DATA.
C      THIS IS FOR A DATA COMPRESSION STUDY.  THE TRANSFORM
C      MATRIX FILENAME IS READ FROM DISK.
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C      OPENR
C      OPENW
C      READR
C      RUNTIME
C      READT
C      T88
C      T8
C      WRITR
C      RESET
C
C*****
C
C      use this space for added information unique to this routine
C
C*****
C
C      PARAMETER ITTO=10,ITTI=11
C      INTEGER IDATA(256,8)
C      REAL RMASK(64), BLOCK(64), T(8,8)
C
C      CALL READT( 0, T, 8 )
C
C      CALL OPENR(0," INPUT FILENAME ? ",512,FSIZE)
C      CALL OPENW(1," OUTPUT FILENAME ? ",512,FSIZE)
C      CALL OPENR(2," MATRIX MASK INPUT FILENAME ? ",256,FSIZE)
C      CALL READR(2,1,RMASK,1,ICNT,IERR)
C      CLOSE(2)
C      CALL RUNTIME
C
C      DO 40 I=1,512,8

```

```

CALL READR(0,I,IDATA,8,ICNT,IERR)
TYPE ' CHUNK STARTING AT RECORD ',I,' HAS BEEN READ '
C
      DO 30 N=1,256,4
      N3=N+3
      L=1
C
      DO 10 J=1,8
      DO 10 K=N,N3
      BLOCK(L) = FLOAT( BYTE(IDATA(K,J),1) )
      BLOCK(L+1) = FLOAT( BYTE(IDATA(K,J),2) )
      L=L+2
10    CONTINUE
C
      CALL T88( BLOCK, T, 0 )
C
      DO 15 L=1,64
      BLOCK(L) = RMASK(L)*BLOCK(L)
15    CONTINUE
C
      CALL T88( BLOCK, T, 1 )
C
      DO 20 L=1,64
      IF(BLOCK(L) .LT. 0.0 ) BLOCK(L) = 0.0
      IF(BLOCK(L) .GT. 255.0 ) BLOCK(L) = 255.0
20    CONTINUE
C
      L=1
      DO 25 J=1,8
      DO 25 K=N,N3
      BYTE( IDATA(K,J),1 ) = IFIX( BLOCK(L) + .5 )
      BYTE( IDATA(K,J),2 ) = IFIX( BLOCK(L+1) + .5 )
      L=L+2
25    CONTINUE
C
      CONTINUE
30
C
      CALL WRITR(1,I,IDATA,8,IERR)
40    CONTINUE
C
CALL RESET
CALL RUNTIME
STOP
END

```

C*****

C

C IMAGE COMPRESSION TRANSFORMATIONS

C

C DG FORTRAN 5 SOURCE FILENAME: TRANSZMEAN.FR

C

C DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

C

C REVISION DATE PROGRAMMER

C

C 00.0 DEC 12, 1980 MYRON FLICKNER

C

C 01.0 JAN 28, 1982 MYRON FLICKNER

C

C*****

C

C PURPOSE

C

C THE ROUTINE COMPUTES THE 2-DIMENSIONAL TRANSFORMATION
C OF THE 8 X 8 BLOCKS OF AN IMAGE. THE RESULTING TRANSFORM
C COEFFICIENTS ARE THEN MASKED BY A MASK MATRIX. THE INVERSE
C TRANSFORM IS THEN COMPUTED TO RECONSTRUCT THE DATA.
C THIS IS FOR A DATA COMPRESSION STUDY. THE TRANSFORM
C MATRIX FILENAME IS READ FROM DISK.
C THE MEAN OF THE BLOCK IS SUBTRACTED OFF BEFORE TRANSFORMATION.
C THE MEAN IS ADDED BACK AFTER RECONSTRUCTION.

C

C

C ROUTINE(S) CALLED BY THIS ROUTINE

C

C OPENR

C

C OPENW

C

C READR

C

C T8

C

C T88

C

C READT

C

C RUNTIME

C

C WRITR

C

C RESET

C

C*****

C

C use this space for added information unique to this routine.

C

C*****

C

C PARAMETER ITTO=10, ITTI=11

C

C INTEGER IDATA(256,8)

C

C REAL RMASK(64), BLOCK(64), T(8,8), AVERAGE

C

C CALL READT(0, T, 8)

C

C CALL OPENR(0, ' INPUT FILENAME ? ', 512, FSIZE)

C

C CALL OPENW(1, ' OUTPUT FILENAME ? ', 512, FSIZE)

C

C CALL OPENR(2, ' MATRIX MASK INPUT FILENAME ? ', 256, FSIZE)

C

C CALL READR(2, 1, RMASK, 1, ICNT, IERR)

C

C CLOSE(2)

C

C CALL RUNTIME

```

C
DO 40 I=1,512,8
CALL READR(0,I,IDATA,8,ICNT,IERR)
TYPE ' CHUNK STARTING AT RECORD ',I,' HAS BEEN READ '
C
DO 30 N=1,256,4
N3=N+3
L=1
AVERAGE = 0.0
C
DO 10 J=1,8
DO 10 K=N,N3
BLOCK(L) = FLOAT( BYTE(IDATA(K,J),1) )
BLOCK(L+1) = FLOAT( BYTE(IDATA(K,J),2) )
AVERAGE = AVERAGE + BLOCK(L) + BLOCK(L+1)
L=L+2
10 CONTINUE
C
AVERAGE = AVERAGE/64.0
C
DO 11 J=1,64
BLOCK(J) = BLOCK(J) - AVERAGE
11 CONTINUE
C
CALL T88( BLOCK, T, 0 )
C
DO 15 L=1,64
BLOCK(L) = RMASK(L)*BLOCK(L)
15 CONTINUE
C
CALL T88( BLOCK, T, 1 )
C
DO 20 L=1,64
BLOCK(L) = BLOCK(L) + AVERAGE
IF(BLOCK(L) .LT. 0.0 ) BLOCK(L) = 0.0
IF(BLOCK(L) .GT. 255.0 ) BLOCK(L) = 255.0
20 CONTINUE
C
L=1
DO 25 J=1,8
DO 25 K=N,N3
BYTE( IDATA(K,J),1 ) = IFIX( BLOCK(L) + .5 )
BYTE( IDATA(K,J),2 ) = IFIX( BLOCK(L+1) + .5 )
L=L+2
25 CONTINUE
C
30 CONTINUE
C
CALL WRITR(1,I,IDATA,8,IERR)
40 CONTINUE
C
CALL RESET
CALL RUNTIME
STOP
END

```

```

C*****
C
C      WHT MATRIX GENERATION
C
C      DG FORTRAN 5 SOURCE FILENAME:          WHTMAT.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              JAN 20, 1982    MYRON FLICKNER
C
C*****
C
C      PURPOSE
C
C          THE ROUTINE GENERATES AN N X N WHT MATRIX
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          WRITR
C          CHECK
C          OPENW
C
C*****
C
C      THE ROUTINE USES THE SIGN OF THE SCT TO GET THE WALSH
C      HADAMARD MATRIX
C
C*****
C
C      DOUBLE PRECISION  A, THETA, PI, TEMP
C      REAL X(64,64)
C      ACCEPT ' ENTER VALUE OF N ? ', N
C      CALL OPENW( 0, ' OUTPUT FILENAME ? ', N*4, F )
C      A = DSQRT( 1.0/DFLOAT( N ) )
C      PI = DATAN(1.0)*4.0
C
C          DO 30 J=1,N
C
C              DO 20 I=1,N
C                  THETA = DFLOAT( (I-1)*(J-1) )*PI/DFLOAT(N-1)
C                  TEMP = DCOS(THETA)
C                  X(I,J) = SIGN( A, SNGL( TEMP ) )
C
20          CONTINUE
C
C          CALL WRITR( 0, J, X(1,J), 1, IERR )
C          CALL CHECK( IERR )
C
30          CONTINUE
C
C      STOP
C      END

```

```

C*****
C
C      ROOT MEAN SQUARE ERROR
C
C      DG FORTRAN 5 SOURCE FILENAME:          RMSERROR.FR
C
C      DEPARTMENT OF ELECTRICAL ENGINEERING    KANSAS STATE UNIVERSITY
C
C      REVISION          DATE          PROGRAMMER
C      -----          ----          -
C      00.0              DEC 12, 1981    MYRON FLICKNER
C
C*****
C
C      PURPOSE
C
C          THE ROUTINE CALCULATES THE RMS ERROR BETWEEN
C          TWO BYTE OR FLOATING POINT IMAGE FILES.
C
C      ROUTINE(S) CALLED BY THIS ROUTINE
C
C          ASK
C
C*****
C
C      use this space for added information unique to this routine
C
C*****
C
C      DOUBLE PRECISION ERROR, DIFF
C      LOGICAL FPOINT
C      REAL LINE1(512), LINE2(512)
C
C      CALL ASK( ' FLOATING POINT INPUT FILE (Y,N=(CR)) ? ', FPOINT )
C      ACCEPT ' NUMBER OF ELEMENTS PER LINE --> ', NELEM
C      ACCEPT ' NUMBER OF LINES PER IMAGE --> ', NLINE
C      NBYTE = 1
C      IF( FPOINT ) NBYTE=4
C
C      CONTINUE
C      ERROR = 0.0
C      CALL OPENR( 1, ' INPUT FILE # 1 ? ', NELEM*NBYTE, F )
C      CALL OPENR( 2, ' INPUT FILE # 2 ? ', NELEM*NBYTE, F )
C
C          DO 25 I=1,NLINE
C              CALL READR( 1, I, LINE1, 1, ICNT, IERR )
C              CALL READR( 2, I, LINE2, 1, ICNT, IERR )
C              IF( FPOINT ) GO TO 15
C
C                  DO 10 J=1,NELEM
C                      DIFF = DBLE( BYTE( LINE1,J ) - BYTE( LINE2,J ) )
C                      ERROR = DIFF*DIFF + ERROR
C                  CONTINUE
C
C          GO TO 25
C
C      CONTINUE
C      DO 20 J=1,NELEM

```

```
                DIFF = DBLE( LINE1(J) - LINE2(J) )
                ERROR = DIFF*DIFF + ERROR
20             CONTINUE
C
25             CONTINUE
C
                DIFF = DFLOAT( NELEM )*DFLOAT( NLINE )
                ERROR = DSQRT( ERROR/DIFF )
                TYPE ' THE RMS ERROR IS ', ERROR
                STOP
                END
```


THE DISCRETE COSINE TRANSFORM

by

MYRON DALE FLICKNER

B. S., Kansas State University, 1980

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1982

Abstract

This is a tutorial paper on the Discrete Cosine Transform. Published uses of the DCT are presented. Different algorithms to compute the DCT are examined. Finally, a comparison of different orthogonal transforms used for image transform coding shows the DCT to perform better than the Discrete Sine Transform (also called the fast Karhuene Loeve Transform), the newly introduced symmetric Cosine Transform, and the Walsh Hadamard Transform.