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:

[Signature]
Major Professor
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter I: Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Chapter II: Definitions</td>
<td>3</td>
</tr>
<tr>
<td>Chapter III: Computational Algorithms</td>
<td>9</td>
</tr>
<tr>
<td>Chapter IV: Image Compression Applications</td>
<td>15</td>
</tr>
<tr>
<td>Chapter V: Conclusions</td>
<td>34</td>
</tr>
<tr>
<td>References</td>
<td>35</td>
</tr>
<tr>
<td>Acknowledgment</td>
<td>38</td>
</tr>
<tr>
<td>Appendix A: A Fast Discrete Cosine Routine</td>
<td>39</td>
</tr>
<tr>
<td>Appendix B: Simulation programs</td>
<td>43</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

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

LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Endpoint error</td>
<td>24</td>
</tr>
<tr>
<td>2. Transformations of a Constant Matrix</td>
<td>28</td>
</tr>
</tbody>
</table>
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 kilobits/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

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

where $\bar{X}$ is the $N \times 1$ column vector containing the data sequence; $\bar{A}$ is a $N \times N$ 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

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

$\bar{F}$ is the resulting $N \times 1$ 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

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

Similarly the two-dimensional IDCT is defined as

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

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:

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

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

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

is the one-dimensional DCT of $x(m,\ell)$ 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

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

where $\mathbf{F}$ is the $N \times N$ matrix containing the DCT coefficients, $\mathbf{A}$ is the $N \times N$ coefficient matrix, and $\mathbf{X}$ 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 $\mathbf{b}(k)$ whose elements are computed as

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

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

$$x(m) = \sum_{k=0}^{N-1} F(k) \mathbf{b}(k).$$
For example, with N=4 the DCT basis vectors are shown below:

\[
\begin{bmatrix}
.500 \\
.500 \\
.500 \\
.500
\end{bmatrix} \quad \begin{bmatrix}
.653 \\
.271 \\
-.271 \\
-.653
\end{bmatrix} \quad \begin{bmatrix}
.500 \\
-.500 \\
-.500 \\
.500
\end{bmatrix} \quad \begin{bmatrix}
.271 \\
-.653 \\
.653 \\
-.271
\end{bmatrix}
\]

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

\[
\begin{bmatrix}
1 \\
2 \\
3 \\
4
\end{bmatrix} = 5.0 \begin{bmatrix}
.500 \\
.500 \\
.500 \\
.500
\end{bmatrix} - 2.23 \begin{bmatrix}
.653 \\
.271 \\
-.271 \\
-.653
\end{bmatrix} + 0.0 \begin{bmatrix}
.500 \\
-.500 \\
-.500 \\
.500
\end{bmatrix} - 0.19 \begin{bmatrix}
.271 \\
-.653 \\
.653 \\
-.271
\end{bmatrix}
\]

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

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

\( a_m \), 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:

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

Here \( \rho \) is defined as the statistical correlation between pixels.

Solving (2) as \( \rho \to 1 \) 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 \( \rho \to 1 \) [29].

By looking at the transcendental equation (2) we can see that \( \omega_m \) is a function of \( \rho \). 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 \( \omega_m \) is almost a linear function of \( \rho \). Therefore, the DCT is a good approximation to the optimum transform for a first order Markov process for \( \rho \) near 1.
FIGURE 1

The roots $\omega_m$ 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

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

Performing some simple algebra gives

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

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

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

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 arbit-
rary 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 Kämanger 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:

\[
v(m) = \begin{cases} 
  x(2m) & 0 \leq m < \left[ \frac{N-1}{2} \right] \\
  x(2m-1) & \left[ \frac{N+1}{2} \right] \leq m \leq N-1 
\end{cases}
\]

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

\[
F(k) = \sqrt{\frac{2}{N}} \begin{cases} 
  c(k) \text{ Real} \{V(k)\} & 0 \leq k < \left[ \frac{N}{2} \right] \\
  -c(k) \text{ Imag} \{V(k)\}
\end{cases}
\]

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

\[
\hat{V}(2K) = \text{Real} \{V(k)\} \quad 0 \leq k < \left[ \frac{N}{2} \right]
\]

\[
\hat{V}(2k+1) = -\text{Imag} \{V(k)\} 
\]
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 inplace 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 \( 4 \times 4 \) 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

\[
\begin{bmatrix}
38.4 & 38.1 & 37.2 & 31.4 \\
47.0 & 22.3 & 13.7 & 11.0 \\
48.6 & 11.0 & 4.0 & 2.4 \\
36.9 & 11.4 & 2.3 & -1.3
\end{bmatrix}
\]

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, \( \overline{C}_x \), defined as

\[
\overline{C}_x = \text{E}\left\{ (Z - \overline{Z})(Z - \overline{Z})' \right\}
\]

where \( \text{E}\{\cdot\} \) 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 com-
putational 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 ran-
dom process within a small \( N<16 \) block. As a result the data covari-
ance matrix

\[
\mathbf{C}_x = \begin{bmatrix}
1 & \rho & \rho^2 & \cdots & \rho^N \\
\rho & 1 & \rho & \cdots & \rho^{N-1} \\
\rho^2 & \rho & 1 & \cdots & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\rho^N & \cdots & \cdots & \cdots & 1
\end{bmatrix}
\]

is a Toeplitz matrix. Here \( \rho \) 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:

\[
\begin{bmatrix}
\sigma_{11}^2 & \sigma_{11}\sigma_{22} & \cdots & \sigma_{NN}\sigma_{11} \\
\sigma_{11}\sigma_{22} & \sigma_{22}^2 & \cdots & \cdots \\
\vdots & \vdots & \ddots & \vdots \\
\sigma_{NN}\sigma_{11} & \cdots & \cdots & \sigma_{NN}^2
\end{bmatrix}
\]
where the $c_{ij}$'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

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

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

$$\bar{W} = \text{diag} \left( \frac{1}{\sqrt{1+p^2}}, 1, 1, \ldots, \frac{1}{\sqrt{1+p^2}} \right).$$

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

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 transforms' decorrelation properties. An 8 x 8 Toeplitz matrix \( \overline{C}_x \) was transformed in two dimensions to produce \( \overline{C}_z \). The residual correlation was then computed as follows:

\[
\rho = \frac{1}{N} \frac{1}{(N-1)} \sum_{i=0}^{N-1} \sum_{j=0, i \neq j}^{N-1} \left[ C_z(i,j) \right]^2.
\]

The value of \( \rho \) 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 \( \rho \) approaches one. The residual correlation for the SCT and DST grows rapidly for \( \rho > 0.5 \). 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, \( \overline{C}_x \), becomes a constant matrix. Then for the Walsh and the DCT, \( \overline{C}_z \) 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 \( \overline{C}_z \) 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
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 boundary. A simulation was used to verify this suggestion. An $8 \times 8$ 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.
<table>
<thead>
<tr>
<th>DCT</th>
<th>SCT</th>
<th>WHT</th>
<th>DST</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Table" /></td>
<td><img src="image" alt="Table" /></td>
<td><img src="image" alt="Table" /></td>
<td><img src="image" alt="Table" /></td>
</tr>
</tbody>
</table>

Table 1
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 x 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.
\[
\begin{bmatrix}
8.0 & 0.0 & \cdots & \cdots & 0.0 \\
0.0 & 0.0 & \cdots & \cdots & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots \\
0.0 & \cdots & \cdots & \cdots & 0.0
\end{bmatrix}
\]

DCT and WHT

\[
\begin{bmatrix}
7.9 & 0.0 & 0.62 & 0.0 & 0.62 & 0.0 & 0.62 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.62 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.62 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0
\end{bmatrix}
\]

SCT

\[
\begin{bmatrix}
7.1 & 0.0 & 2.2 & 0.0 & 1.1 & 0.0 & 0.0 & .46 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
2.2 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
1.1 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
.46 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\
0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0
\end{bmatrix}
\]

DST

Table 2
Transformation of a Constant Matrix
Transformation: DCT
Coefficients retained: 16
RMS error: 15.4

Transformation: WHT
Coefficients retained: 16
RMS error: 17.7

FIGURE 8; 4:1 compression.
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.
Transformation: DCT
Coefficients retained: 8
RMS error: 26.0

Transformation: WHT
Coefficients retained: 8
RMS error: 28.2

FIGURE 9 continued; 8:1 compression.
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


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** FAST DISCRETE COSINE TRANSFORMATION **C

DG FORTRAN 5 SOURCE FILENAME: FDCT.FR

DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

REVISION DATE PROGRAMMER
----- ----- ------
00.0 MAY 07, 1981 MYRON FICKNER
01.0 NOV 06, 1981 MYRON FICKNER

CALLING SEQUENCE

CALL FDCT ( X, N, INV )

PURPOSE

THE ROUTINE IMPLEMENTS A FAST DISCRETE COSINE
TRANSFORMATION USING AN ALGORITHM DEFINED BY
JOHN MAKHOUL IN THE FEBRUARY 1980 ASSP TRANSACTION

ROUTINE(S) CALLED BY THIS ROUTINE

FFS - IEEE SIGNAL PROCESSING ROUTINE
FFA - IEEE SIGNAL PROCESSING ROUTINE

ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE

X - VECTOR TO BE TRANSFORMED
N - NUMBER OF ELEMENTS TO BE TRANSFORMED (POWER OF 2)

ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE

X - TRANSFORMED VECTOR

NOTE 1: This subroutine makes no checks on the validity
of the data supplied by the calling routine.

The subroutine FFS and FFA are subroutines from
the Programs for Digital Signal Processing, New York!
IEEE Press.

SUBROUTINE FDCT( X, N, INV )

REAL X(N), VR(2050), PI, PI2, RN, THETA
COMPLEX V(1025), TEMPI, TEMP2, W
EQUIVALENCE ( VR(1), V(1) )
DATA PI/3.141592654/, PI2/6.283185307/
IF( N .GT. 2048 ) STOP ' N TO LARGE FOR FDCT '

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

REORDER DATA

DO 10 K=1,N2
   VR(K) = X(I)
   VR(K+N2) = X(M)
   I = I + 2
   M = M - 2
   CONTINUE

DO N POINT REAL FFT USING N/2 POINT COMPLEX FFT
THIS IS DONE BY FFA. FFA IS IN THE IEEE PROGRAMS FOR DIGITAL
SIGNAL PROCESSING.

CALL FFA(V, N)

NOW FFT IS DONE. NOTE WE ONLY CALCULATED THE FIRST N2+1
POINTS. THE DFT OF A REAL SEQUENCE IS HERMITIAN SO WE DO NOT
NEED THE OTHER POINTS.

NOW GET THE DCT COEFFICIENTS

X(1) = VR(1)/SQRT(RN)

DO 20 K=1,N2
   W = CEXP(CMPLX(0.0, -FLOAT(K)*PI/(2.0*RN)))
   V(K+1) = V(K+1)*W
   X(K+1) = C*REAL(V(K+1))
   X(N-K+1) = -C*AIMAG(V(K+1))
20 CONTINUE

RETURN

CONTINUE

DO THE INVERSE DCT

C = 1.0/SQRT(2.0/RN)
V(1) = CMPLX(X(1)*SQRT(RN), 0.0)

DO 35 K=1,N2
   W = CEXP(CMPLX(0.0, FLOAT(K)*PI/(2.0*RN)))
   V(K+1) = W*CMPLX(C*X(K+1), -C*X(N-K+1))
35 CONTINUE

USE N/2+1 POINT COMPLEX IFFT TO GET N POINT REAL
IFFT. THIS IS DONE IN FFS. FFS IS ALSO A IEEE PROGRAM.

CALL FFS(V, N)
C REORDER DATA
C
DO 45 K=1,N2
X(I) = VR(K)
X(H) = VR(K+N2)
I = I + 2
M = M - 2
CONTINUE

45 C
RETURN
END
Appendix B

Programs Used for Simulation
C** DCT MATRIX GENERATION
C
DG FORTRAN 5 SOURCE FILENAME: DCTMAT.FR
C
DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY
C
REVISION DATE PROGRAMMER
----- ---- ---------
00.0 JAN 20, 1982 MYRON FLICKNER
C

** PURPOSE
THE ROUTINE GENERATES AN N X N DCT MATRIX

** ROUTINE(S) CALLED BY THIS ROUTINE
WRITR
CHECK
OPENW

** use this space for added information unique to this routine

** DOUBLE PRECISION A, THETA, PI, TEMP
REAL X(64,64)
ACCEPT ' ENTER VALUE OF N ? ', N
CALL OPENW( 0, ' OUTPUT FILENAME ? ', N+4, F )
A = DSORT( 1.0/DFLOAT( N ) )
PI = DATAN(1.0)*4.0

DO 30 J=1,N

DO 20 I=1,N
THETA = DFLOAT( PI/DFLOAT(2*N)*1
DFLOAT(J-1)*DFLOAT(2*(I-1)+1) )
TEMP = A*DCOS(THETA)
X(I,J) = SNGL( TEMP )
20 CONTINUE

A = DSORT( 2.0/DFLOAT( N ) )
CALL WRITR( 0, J, X(1,J), 1, IERR )
CALL CHECK( IERR )
30 CONTINUE

STOP
END
ENDPOINT ERROR SIMULATION

DG FORTRAN 5 SOURCE FILENAME: ENDPOINT.FOR

DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

REVISION DATE PROGRAMMER
-------- ---- ------------------
00.0 JAN 28, 1982 MYRON FLICKNER

PURPOSE

THE ROUTINE COMPUTES THE PERCENT ENDPOINT
ERROR FOR A TRANSFORM. INPUTS ARE THE MASK FILE,
AND THE PARAMTER RHO FOR THE TOEPPLIZT MODEL.

ROUTINE(S) CALLED BY THIS ROUTINE

ASK
OPENR
READR
TPLZMAT
TBB
RESET

USE THIS SPACE FOR ADDED INFORMATION UNIQUE TO THIS ROUTINE

REAL X(8,8), Y(8,8), DIFF(8), MASK(8,8), T(8,8)
LOGICAL YESNO
CONTINUE
ITII = 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 )

ACCEPT ' ENTER THE VALUE OF RHO ?', RHO
CALL TPLZMAT( X, RHO, 8 )
CALL TPLZMAT( Y, RHO, 8 )
CALL T88( X, T, 0 )

DO 15 I=1,8
DO 15 J=1,8
X(J,I) = X(J,I)*MASK(J,I)
15 CONTINUE
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
CALLING SEQUENCE

CALL READT(ICHAN, T, N)

PURPOSE

THE ROUTINE OPENS AND READS IN AN N X N TRANSFORM MATRIX FILE.

ROUTINE(S) CALLED BY THIS ROUTINE

OPENR
READR
CHECK

ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE

ICHAN - UNUSED LOGICAL CHANNEL
N - SIZE OF TRANSFORM

ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE

T - THE N X N TRANSFORM MATRIX

NOTE 1: This subroutine makes no checks on the validity of the data supplied by the calling routine.

NOTE 2: Argument(s) supplied by the calling routine are not modified by this subroutine.

SUBROUTINE READT(ICHAN, T, N)

REAL T(N,N)

CALL OPENR(ICHAN, 'TRANSFORM MATRIX FILENAME ? ', N*N*4, F)
CALL READR(ICHAN, I, T, I, ICNT, IERR)
CALL CHECK(IERR)
CLOSE ICHAN
RETURN
END
COMPUTE THE RESIDUAL CORRELATION AND ENTROPY

DG FORTRAN 5 SOURCE FILENAME: RESID.FR

DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

REVISION DATE PROGRAMMER
------- ---- -------
00.0 JAN 29, 1982 MYRON FLICKNER

PURPOSE

THE ROUTINE COMPUTES THE RESIDUAL CORRELATION
AND ENTROPY FOR A TOEPLITZ MATRIX FOR A GIVEN
COMPRESSION RATE. THE PARAMETER RHO IS SWEP'T
THOUGH A RANGE OF VALUES SPECIFIED BY THE USER.

ROUTINE(S) CALLED BY THIS ROUTINE

READT
OPENW
RUNTIME
TPLZMAT
T88
WRITR

use this space for added information unique to this routine

PARAMETER N = 9
REAL X(N,N), T(N,N)
DOUBLE PRECISION SUM, ENTROPY

CALL READT( 0, T, N )
ACCEPT ' STARTING RHO ? ', RHO1
ACCEPT ' ENDING RHO ? ', RHO2
ACCEPT ' NUMBER OF POINTS ? ', NPTS
CALL OPENW( 1, ' RESIDUAL FILENAME ? ', 4, F )
CALL OPENW( 2, ' ENTROPY FILENAME ? ', 4, F )

CALL RUNTIME
DELTA = (RHO2 - RHO1) / FLOAT(NPTS)
RHO = RHO1

DO 40 I1=1,NPTS
CALL TPLZMAT( X, RHO, N )
CALL T88( X, T, 0 )
SUM = 0.0

DO 20 J=1,N

DO 40
DO 10 I=1,N
IF(I.EQ.J) GO TO 10
SUM = SUM + X(I,J)*X(I,J)
CONTINUE

C
20 C
SUM = SUM/(FLOAT(N)*FLOAT(N-1))
ENTROPY = 0.0
C
DO 30 I=1,N
ENTROPY = ENTROPY + ALOG( X(I,I) )
CONTINUE
C
30 CALL WRITR( 1, I1, SUM, 1, IERR )
CALL WRITR( 2, I1, ENTROPY, 1, IERR )
RHO = RHO + DELTA
CONTINUE
C
40 CALL RUNTIME
STOP
END
SCT MATRIX GENERATION

DG FORTRAN S SOURCE FILENAME: SCTHAT.FR
DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

REVISION DATE PROGRAMMER
--------- ---- ------------
00.0 JAN 20, 1982 MYRON FLICKNER

PURPOSE

THE ROUTINE GENERATES AN N X N SCT MATRIX

ROUTINE(S) CALLED BY THIS ROUTINE

WRITR
CHECK
OPENW

use this space for added information unique to this routine

DOUBLE PRECISION A, THETA, PI, TEMP
REAL X(64,64)
ACCEPT ' ENTER VALUE OF N ? ', N
CALL OPENW( 0, 'OUTPUT FILENAME ? ', N*4, F)
A = D SQRT ( 2.0/DFLOAT( N-1 ) )
PI = DATAN(1.0)*4.0

DO 30 J=1,N

DO 20 I=1,N
THETA = D FLOAT ( (I-1)*(J-1) )*PI/DFLOAT(N-1)
TEMP = A*DCOS(THETA)
IF( J, .EQ. 1, .OR. J, .EQ. N ) TEMP = TEMP*D SQRT (.5)
IF( I, .EQ. 1, .OR. I, .EQ. N ) TEMP = TEMP*D SQRT (.5)
X(I,J) = SNGL(T EMP )
20 CONTINUE

CALL WRITR( 0, J, X(I,J), 1, IERR )
CALL CHECK( IERR )
30 CONTINUE

STOP
END
CALL T8( X, T, INV )

PURPOSE

THE ROUTINE COMPUTE THE TRANSFORM T OF THE 8 POINT DATA VECTOR X. IF INV IS 0 THE FORWARD TRANSFORM IS COMPUTED, OTHERWISE THE INVERSE TRANSFORM IS COMPUTED.

ROUTINE(S) CALLED BY THIS ROUTINE

NONE

ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE

X - THE 8 POINT VECTOR TO BE TRANSFORMED
T - THE 8 X 8 ORTHOGONAL TRANSFORM MATRIX
INV - FORWARD OR INVERSE FLAG
INV .EQ. 0 -> FORWARD TRANSFORM
INV .NE. 0 -> INVERSE TRANSFORM

ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE

X - THE 8 POINT TRANSFORMED VECTOR

NOTE 1: This subroutine makes no checks on the validity of the data supplied by the calling routine.
SUM=0.0
C
   DO 10 J=1,8
       SUM=X(J)*T(J,I)+SUM
   CONTINUE
C
   Y(I)=SUM
   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
       CONTINUE
C
   Y(I)=SUM
   CONTINUE
C
50 CONTINUE
C
60 CONTINUE
C
   DO 70 I=1,8
       X(I)=Y(I)
    70 CONTINUE
C
RETURN
END
C*****************************************************************************
C TWO-DIMENSIONAL TRANSFORMATIONS OF AN 8 X 8 MATRIX
C
C FORTRAN 5 SOURCE FILENAME: TBB.FR
C
C DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY
C
C REVISION DATE PROGRAMMER
C -------- ---- ----------------
C 00.0 JAN 28, 1982 HYRON FLICKNER
C
C*****************************************************************************
C
C CALLING SEQUENCE
C
C CALL TBB( 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 TO
C
C ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE
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
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 TBB( 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 TB( X(I,I), T, INV )
C 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)
CONTINUE
C
CALL TB(Y, T, INV)
C
DO 30 J=1,8
X(I,J) = Y(J)
CONTINUE
C
40 CONTINUE
C
RETURN
END
TOEPLIZT MODEL VARIANCE CRITERIA

DG FORTRAN 5 SOURCE FILENAME: TOEPLIZT.FR

DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

REVISION DATE PROGRAMMER
------- ---- ---------------
00.0 JAN 28, 1982 MYRON FLICKNER

PURPOSE

THE ROUTINE ALLOWS FOR THE CALCULATION OF THE
MATRIX POSITIONS TO SAVE FOR A GIVEN NUMBER OF
RETAINED COEFFICIENTS.

ROUTINE(S) CALLED BY THIS ROUTINE

READT
TPLZMAT
OPENW
WRITR
RESET
ASK

use this space for added information unique to this routine

PARAMETER LPT=12,ITTI=11,ITTD=10
REAL X(9,8), Y(8,8), T(3,8)
INTEGER ROW(64), COLUMN(64)
LOGICAL YESNO

100 CONTINUE
CALL READT (0, T, 8)
ACCEPT * ENTER THE VALUE OF RHO ? ', RHO
CALL TPLZMAT (X, RHO, 8)
WRITE (LPT, 3)
FORMAT (///, '1', T50, 'THE TOEPLIZT MATRIX', ///)

DO 1020 J=1,8
WRITE (LPT, 1) (X(J, K) K=1, 8)
1 FORMAT (* ',8(G15.7))
1020 CONTINUE

CALL T88 (X, T, 0)
WRITE (LPT, 4)
FORMAT (///, '1', T50, 'THE TRANSFORMED MATRIX', ///)

DO 1030 J=1,8
WRITE(LPT,1) (X(J,K) K=1,8)
CONTINUE
C
WRITE(LPT,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)
CONTINUE
1040
C
DO 1050 J=1,8
WRITE(LPT,1) (Y(J,K) K=1,8)
CONTINUE
1050
C
CALL OPENW (1, 'OUTPUT MASK FILE ?', 256, F)
ACCEPT 'NUMBER OF COEFFICIENT TO RETAIN ?', NUMBER
C
sort the variances

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
60
CONTINUE
C
Y(ROW(K), COLUMN(K)) = 0.0
CONTINUE
70
C
DO 80 J=1,8
DO 80 I=1,8
Y(I,J) = 0.0
80
CONTINUE
C
save the largest 'NUMBER' of variances

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
**GENERATE A TOEPLIZT MATRIX**

**DG FORTRAN 5 SOURCE FILENAME:** TPLZMAT.FR

**DEPARTMENT OF ELECTRICAL ENGINEERING** KANSAS STATE UNIVERSITY

**REVISION DATE PROGRAMMER**

00.0 JAN 28, 1982 MYRON FLICKNER

**CALLING SEQUENCE**

CALL TPLZMAT( X, RHO, L )

**PURPOSE**

THE ROUTINE GENERATES A FIRST-ORDER MARKOV COVARIANCE MATRIX OF SIZE L X L AND PARAMETER RHO.

**ROUTINE(S) CALLED BY THIS ROUTINE**

NONE

**ARGUMENT(S) REQUIRED FROM THE CALLING ROUTINE**

RHO - PARAMETER OF GENERATION
L - SIZE OF MATRIX

**ARGUMENT(S) SUPPLIED TO THE CALLING ROUTINE**

X - GENERATED MATRIX

**NOTE 1:** This subroutine makes no checks on the validity of the data supplied by the calling routine.

**NOTE 2:** Argument(s) supplied by the calling routine are not modified by this subroutine.

**SUBROUTINE TPLZMAT( X, RHO, L )**

REAL X(L,L), RHO

```
  DO 20 J=1,L
      N=1-J
      DO 10 K=1,L
          X(J,K)=RHO**ABS(N)
          N=N+1
  10 CONTINUE
```
CONTINUE

RETURN

END
IMAGE COMPRESSION TRANSFORMATIONS

DG FORTRAN 5 SOURCE FILENAME: TRANSFORM.FR

DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

REVISION DATE PROGRAMMER
--------- -------- ------------
00.0 DEC 12, 1980 MYRON FLICKNER
01.0 JAN 28, 1992 MYRON FLICKNER

PURPOSE

THE ROUTINE COMPUTES THE 2-DIMENSIONAL TRANSFORMATION OF THE 8 X 8 BLOCKS OF AN IMAGE. THE RESULTING TRANSFORM COEFFICIENTS ARE THEN MASKED BY A MASK MATRIX. THE INVERSE TRANSFORM IS THEN COMPUTED TO RECONSTRUCT THE DATA. THIS IS FOR A DATA COMPRESSION STUDY. THE TRANSFORM MATRIX FILENAME IS READ FROM DISK.

ROUTINE(S) CALLED BY THIS ROUTINE

OPENR
OPENW
READR
RUNTIME
READT
T88
T8
WRITR
RESET

PARAMETER ITTO=10, ITTI=11
INTEGER IDATA(256,8)
REAL RMASK(64), BLOCK(64), T(8,8)

CALL READT(0, T, 8)

CALL OPENR(0,' INPUT FILENAME ? ',512,FSIZE)
CALL OPENW(1,' OUTPUT FILENAME ? ',512,FSIZE)
CALL OPENR(2,' MATRIX MASK INPUT FILENAME ? ',256,FSIZE)
CALL READR(2,1,RMASK,1,ICNT,IERR)
CLOSE(2)
CALL RUNTIME

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
  M3=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

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

30 CONTINUE

C
CALL WRTR(L,I,IData,0,IERR)
40 CONTINUE

C
CALL RESET
CALL RUTIME
STOP
END
IMAGE COMPRESSION TRANSFORMATIONS

DG FORTRAN 5 SOURCE FILENAME: TRANSZMEAN.FR

DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

REVISION DATE PROGRAMMER
------ ----- -------
00.0 DEC 12, 1980 MYRON FLICKNER
01.0 JAN 28, 1982 MYRON FLICKNER

POURICE

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

ROUTINE(S) CALLED BY THIS ROUTINE

OPENR
OPENW
READR
T8
T88
READT
RUNTIME
WRITR
RESET

use this space for added information unique to this routine.

PARAMETER ITD=10, ITI=11
INTEGER IDATA(256,8)
REAL RMASK(64), BLOCK(64), T(8,8), AVERAGE

CALL READT(0, T, 8)

CALL OPENR(0,' INPUT FILENAME ?,512,FSIZE)
CALL OPENW(1,' OUTPUT FILENAME ?,512,FSIZE)
CALL OPENR(2,' MATRIX MASK INPUT FILENAME ?,256,FSIZE)
CALL READR(2,1,RMASK,1,ICNT,IERR)
CLOSE(2)
CALL RUNTIME
DO 40 I=1,512,8
CALL READR(0,I,IDATA,8,ICNT,IERR)
TYPE ' CHUNK STARTING AT RECORD 'I' HAS BEEN READ'

DO 30 N=1,256,4
N3=N+3
L=1
AVERAGE = 0.0

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
CONTINUE

AVERAGE = AVERAGE/64.0

DO 11 J=1,64
BLOCK(J) = BLOCK(J) - AVERAGE
CONTINUE

CALL T88( BLOCK, T, 0 )

DO 15 L=1,64
BLOCK(L) = RMASK(L)*BLOCK(L)
CONTINUE

CALL T88( BLOCK, T, 1 )

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
CONTINUE

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
CONTINUE

CALL WRITR(1,I,IDATA,8,IERR)
CONTINUE

CALL RESET
CALL RUNTIME
STOP
END
**WHT MATRIX GENERATION**

**DG FORTRAN 5 SOURCE FILENAME:** WHTMAT.F

**DEPARTMENT OF ELECTRICAL ENGINEERING** KANSAS STATE UNIVERSITY

**REVISION** **DATE** **PROGRAMMER**

00.0 JAN 20, 1982 MYRON FLICKNER

**PURPOSE**

THE ROUTINE GENERATES AN N X N WHT MATRIX

**ROUTINE(S) CALLED BY THIS ROUTINE**

WRTR
CHECK
OPENW

**THE ROUTINE USES THE SIGN OF THE SCT TO GET THE WALSH HADAMARD MATRIX**

**DOUBLE PRECISION** A, THETA, PI, TEMP
**REAL X(64,64)**

ACCEPT ' ENTER VALUE OF N ? ', N
CALL OPENW( 0, ' OUTPUT FILENAME ? ', N*4, F )
A = DSORT( 1.0/DFLOAT( N ) )
PI = DATAN(1.0)*4.0

DO 30 J=1,N

    DO 20 I=1,N
        THETA = DFLOAT( (I-1)*(J-1) )*PI/DFLOAT(N-1)
        TEMP = DCOS(THETA)
        X(I,J) = SIGN( A, SMUL( TEMP ) )
        CONTINUE

20    CALL WRTR( 0, J, X(I,J), 1, IERR )
CALL CHECK( IERR )

30    CONTINUE

STOP
END
ROOT MEAN SQUARE ERROR

DG FORTRAN 5 SOURCE FILENAME: RMSERROR.F

DEPARTMENT OF ELECTRICAL ENGINEERING KANSAS STATE UNIVERSITY

REVISION DATE PROGRAMMER
------- ---- -------
00.0 DEC 12, 1981 MYRON FLICKNER

PURPOSE

THE ROUTINE CALCULATES THE RMS ERROR BETWEEN TWO BYTE OR FLOATING POINT IMAGE FILES.

ROUTINE(S) CALLED BY THIS ROUTINE

ASK

DOUBLE PRECISION ERROR, DIFF
LOGICAL FPOINT
REAL LINE1(512), LINE2(512)

CALL ASK(' FLOATING POINT INPUT FILE (Y,N=(CR)) ? ', FPOINT)
ACCEPT ' NUMBER OF ELEMENTS PER LINE --> ', NELEM
ACCEPT ' NUMBER OF LINES PER IMAGE --> ', NLINE
NBYTE = 1
IF( FPOINT ) NBYTE=4

CONTINUE
ERROR = 0.0
CALL OPENR( 1, ' INPUT FILE # 1 ?', NELEM*NBYTE, F )
CALL OPENR( 2, ' INPUT FILE # 2 ?', NELEM*NBYTE, F )

DO 25 I=1,NLINE
   CALL READR( 1, I, LINE1, 1, ICNT, IERR )
   CALL READR( 2, I, LINE2, 1, ICNT, IERR )
   IF( FPOINT ) GO TO 15

25      DO 10 J=1,NELEM
       DIFF = DBLE( BYTE(LINE1,J) - BYTE(LINE2,J) )
       ERROR = DIFF*DIFF + ERROR
       CONTINUE

   GO TO 25

15 CONTINUE
   DO 20 J=1,NELEM
DIFF = DBLE( LINE1(J) - LINE2(J) )
ERROR = DIFF*DIFF + ERROR
CONTINUE

C

CONTINUE

C

DIFF = DFLOAT( NELEM )*DFLOAT( NLINE )
ERROR = DSORT( 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.