NASA Technical Memorandum 106183

Subband/Transform Functions for Image Processing

Daniel Glover
Lewis Research Center
Cleveland, Ohio

June 1993

NASA

ERRATA

NASA Technical Memorandum 106183

Daniel R. Glover
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

Replace existing pages 16, 28, and 29.

SUBBAND/TRANSFORM FUNCTIONS FOR IMAGE PROCESSING

Daniel Glover
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

ABSTRACT

Functions for image data processing written for use with the MATLAB software package are presented. These functions provide the capability to transform image data with block transformations (such as the Walsh Hadamard) and to produce spatial frequency subbands of the transformed data. Block transforms are equivalent to simple subband systems. The transform coefficients are reordered using a simple permutation to give subbands. The low frequency subband is a low resolution version of the original image, while the higher frequency subbands contain edge information.

The transform functions can be cascaded to provide further decomposition into more subbands. If the cascade is applied to all four of the first stage subbands (in the case of a four band decomposition), then a uniform structure of sixteen bands is obtained. If the cascade is applied only to the low frequency subband, an octave structure of seven bands results. Functions for the inverse transforms are also given.

These functions can be used for image data compression systems. The transforms do not in themselves produce data compression, but prepare the data for quantization and compression. Sample quantization functions for subbands are also given. A typical compression approach is to subband the image data, quantize it, then use statistical coding (e.g., run-length coding followed by Huffman coding) for compression. Contour plots of image data and subbanded data are shown.

INTRODUCTION

Subband coding is a data processing technique which transforms the original signal into several frequency bands. The classic method uses a bank of digital filters to provide the frequency decomposition followed by decimators to reduce the total number of samples in all bands to the same (or nearly the same) as the original signal. Subbanding does not result in any data compression by itself (in fact, it results in a little data expansion), but it prepares the data for lossy compression using quantizers and statistical coders. For image data, a one dimensional filter bank is used on the data in two directions (horizontal and vertical) to give a separable approximation to two dimensional filtering.

Block transform coding is a related technique that predates subband coding. A transform matrix is applied to the image data one block at a time. The block size corresponds to the transform matrix size. A separable transform is used for image data in essence applying the transform in both the horizontal and vertical directions. Block transforms can be used to produce subbands equivalent to those obtained with simple filters.

TWO-DIMENSIONAL, SEPARABLE SUBBAND/TRANSFORM SYSTEMS

The simplest perfect reconstruction subband/transform system uses the identity matrix as the transform:

$$\boldsymbol{I}(2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{1}$$

This trivial case is equivalent to a subbanding system using the following filters:

$$\begin{aligned} H_0(z) &= 1 \\ H_1(z) &= z^{-1} \end{aligned} \tag{2}$$

Although this system splits the original image into four subbands (when used separably), the subbands are all-pass and do not provide any useful properties for compression. The term "perfect reconstruction" refers to the property of obtaining a reconstructed image identical to the original image if no loss has occurred between the forward and inverse transformation stages.

The well-known Walsh-Hadamard Transform (WHT, also known as the Discrete Hadamard Transform) makes use of the Hadamard matrix (**H**, not to be confused with the H(z) notation common for subband analysis filters) for 2x2 blocks [1] (ignoring a scaling factor of $\frac{1}{\sqrt{2}}$ for simplicity):

$$\boldsymbol{H}(1) = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3}$$

This is equivalent to a subbanding system using the following analysis filters (again ignoring the scaling factor):

$$\begin{aligned} H_0(z) &= 1 + z^{-1} \\ H_1(z) &= 1 - z^{-1} \end{aligned} \tag{4}$$

These methods both offer the possibility of perfect reconstruction of the original signal if no information is lost during subsequent coding or transmission. The WHT is attractive because it can be implemented without multipliers, only addition and sign complement functions are needed.

To derive two-dimensional, separable subband filters from the one-dimensional case, construct a row vector from the coefficients of each filter:

$$\begin{aligned} R_1 &= [1 \quad 1] \\ R_2 &= [1 \quad -1] \end{aligned} \tag{5}$$

The two-dimensional filters are obtained by taking the outer product of these vectors: $\mathbf{R_1}'\mathbf{R_1}$, $\mathbf{R_1}'\mathbf{R_2}$, $\mathbf{R_2}'\mathbf{R_1}$, and $\mathbf{R_2}'\mathbf{R_2}$. This is equivalent to filtering horizontally with the first filter and vertically with the second. These two-dimensional filters are obtained from the same process that basis pictures are constructed from a transform matrix (using row basis vectors) [1]. Thus, the two-dimensional subband filters are the basis pictures of the equivalent transform. The two-dimensional filters derived from (4) are the same as the basis pictures formed from the outer product of the row basis vectors of (3), namely:

$$B_1 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}; \; B_2 = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}; \; B_3 = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}; \; B_4 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{6}$$

The subbands are just separate collections of the four transform coefficients. The result of subbanding can be obtained by reordering the WHT coefficients or vice versa. Thus the two-dimensional subband filters of (6) are the same as a block WHT with the outputs (transform coefficients) reordered.

The values for each of the subbands of a block of data, **D**, is the inner product (or element-by-element product) of the basis picture (or 2-D filter) for that subband with the data; for example the low band value, s_1, is given by $s_1 = \mathbf{B_1} \cdot \mathbf{D}$, where the inner product $(\cdot)$ is the sum of the products of corresponding terms in the two matrices. For the WHT, the four subband values (transform coefficients) for a block of data $\mathbf{D} = \begin{bmatrix} d_1 & d_2 \\ d_3 & d_4 \end{bmatrix}$ would be (not including the scaling factor):

$$\begin{aligned} s_1 &= d_1 + d_2 + d_3 + d_4 \\ s_2 &= d_1 + d_2 - d_3 - d_4 \\ s_3 &= d_1 - d_2 + d_3 - d_4 \\ s_4 &= d_1 - d_2 - d_3 + d_4 \end{aligned}$$

After these subband values are calculated, a new block of data is read in. (The subband values should be scaled by dividing by 4 and shifted to positive values if it is desired to view them.)

Since the inverse transform matrix is the same as the forward transform matrix (in this orthogonal case), the reconstruction filters are the same. The scaling factor of ¼ can be included in either set of filters or divided between them. For both the forward and inverse transforms to be identical, the scaling factor should be the same for the transform matrices in the forward transform equation (7) and in the inverse transform equation. For

convenience the scaling factor can be combined and put in one place.

The reconstruction filters (with the scaling factor included) are:

$$\hat{d}_1 = \tfrac{1}{4}(s_1 + s_2 + s_3 + s_4)$$
$$\hat{d}_2 = \tfrac{1}{4}(s_1 + s_2 - s_3 - s_4)$$
$$\hat{d}_3 = \tfrac{1}{4}(s_1 - s_2 + s_3 - s_4)$$
$$\hat{d}_4 = \tfrac{1}{4}(s_1 - s_2 - s_3 + s_4)$$

An implementation of the WHT in 2x2 blocks can also be viewed as a subband analysis bank using separable, two-dimensional filters. If all the low frequency terms (DC coefficients) are collected in one group, the low-horizontal/low-vertical frequency subband is formed. The other subbands are likewise formed by grouping results by frequency band. Collecting one result from each of the four outputs and grouping them together (maintaining the block organization) is equivalent to a block transform. One advantage to organizing the results as subbands is that the low band is a good low resolution representation of the original image. The higher frequency bands contain edge information, as can be expected by looking at $\mathbf{B}_2$, $\mathbf{B}_3$, and $\mathbf{B}_4$, in as image operators.

The decimation that follows subband filtering is achieved by the block organization of the data. The values that would have been decimated are simply not calculated. Since the operations that produce the subbands are additions and subtractions, the increase in the size of the output over the input sample size is only 2 bits. This makes maintaining accuracy for perfect reconstruction relatively easy.

The following relationships between subband filter bank matrix formulations and block transforms is summarized from the pertinent theory given in [2] and [3]. A block transform system can be thought of as a subband system where the subband filter length is equal to the decimation factor. Let a general forward transform matrix, **T**, have a corresponding inverse transform matrix, **t** (for the orthogonal Hadamard matrix, $\mathbf{T} = \mathbf{t} = \mathbf{H}$). The transform matrix, **T**, is related to the subband analysis filters by:

$$T(2) = [R_1 \; R_2]' \tag{7}$$

where $\mathbf{R}_1$ is the vector of coefficients from $H_1(z)$ (for the WHT, given by $\mathbf{R}_1=[1\ 1]$), and $\mathbf{R}_2$ is the vector of coefficients from $H_2(z)$ (e.g., given by $\mathbf{R}_2=[1\ -1]$). More generally, the transform matrix is given by:

$$T(N) = [R_1 \; R_2 \; \ldots \; R_N]' \tag{8}$$

If the filter bank has no filter length greater than the decimation factor, then the transform matrix will be a traditional block transform.

The inverse transform matrix, **t**, is the inverse transpose of the forward matrix. It is also related to the synthesis filters in an equivalent simple subband filter bank. The inverse transpose matrix is made up of row vectors consisting of the coefficients of the subband

synthesis filters, but in reverse order by powers of z compared to the forward transform matrix. The basis images of the inverse transform matrix (t') using row basis vectors can be thought of as two-dimensional synthesis filters. Equivalently, the basis images can be found using column basis vectors (instead of row basis vectors) on the inverse matrix t directly.

In the WHT case the inverse transform matrix is the same as the transform matrix since the Hadamard matrix (H) is unitary and symmetric, but that is not true in general. For perfect reconstruction it is only necessary that t = inv(T)' [2].

A typical two-dimensional block transform would operate on a block (matrix) of data values, D, with a transform matrix, T, to give coefficients, C, as follows [1]:

$$C = T*D*T' \tag{9}$$

where T' is the transpose of T. The inverse transform is:

$$D = t'*C*t \tag{10}$$

Subbanding is equivalent to the traditional block transform when the highest number of filter taps in any channel is the same as the decimation factor. For example, the WHT using Hadamard(1) is the same as subbanding using the two tap filters of (4) separably and decimating by two each time or using the two-dimensional four tap filters derived from (6) and decimating by 4.

Cascading the 2x2 subbanding in a uniform band tree structure is also equivalent to performing larger size block transforms using Kronecker product expansions of the matrix. For example, a 2x2 WHT of data that has already been processed by a 2x2 WHT and organized into subbands is equivalent to performing a 4x4 WHT on the original image. This can easily be shown by comparing the 16 permutations of the Kronecker tensor product of the basis pictures of the 2x2 WHT with the basis pictures of the 4x4 WHT. If the first stage transform coefficients are not organized as subbands, but are left in the same block structure, applying the WHT again results in the original data (because the transform matrix is the same for both the forward and inverse case).

The Kronecker product (also direct product or tensor product) is the operation which creates a larger matrix from two smaller matrices by using the product of the components of one matrix with the other matrix as submatrices. For example, for the Hadamard matrix [1]:

$$H(1) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{11}$$

$$H(2) = \frac{1}{\sqrt{2}} \begin{bmatrix} H(1) & H(1) \\ H(1) & -H(1) \end{bmatrix} \tag{12}$$

giving the 4x4 matrix:

$$H(2) = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \tag{13}$$

The matrix in (13) can be obtained using the MATLAB function:

kron(hadamard(1),hadamard(1))

Cascading the 2x2 transform results in an operation wherein each of the subbands (produced by each of the four basis pictures) is again processed by each of the four basis pictures. This is equivalent to processing the original data in 4x4 blocks with the sixteen possible permutations of Kronecker products of the four 2x2 basis pictures. These are the same sixteen basis pictures that are obtained from the 4x4 transform matrix (which is the Kronecker product of the 2x2 transform matrix with itself). Cascading a simple 2x2 transform to provide 4x4, 8x8, or 16x16 size blocks can provide performance gains in hardware implementations. The 2x2 block circuit can simply be replicated and cascaded to provide a larger block size operation. The cascading operation naturally allows parallel processing which can provide performance gains. The 2x2 transform can also be cascaded in an octave-band tree structure by repeatedly subbanding only the low frequency band (also known as a simple wavelet decomposition).

The order of the subbands in a uniform cascade is not the same as a typical subband system might give, but these can be easily rearranged into whatever order is useful. For example, the LLHH band would normally be located diagonally next to the LLLL band in a subband system, but this position is occupied by the HHLL band using the subband/transform functions herein.

USING THE IMAGE PROCESSING FUNCTIONS

Some functions for transforming image data into subbands and processing the data for lossy compression are given in Appendix A. The functions are:

SUBG (A)	Produces four subbands using a non-orthogonal transform
SYNG (A)	Reconstructs image data from subbands produced by SUBG
OVERFLOW (A)	Finds out-of-range reconstructed values and sets them to 0-255
SUBH (A)	Produces four subbands using Walsh-Hadamard transform
SYNH (A)	Reconstructs image data from subbands produced by SUBH
SUBI (A)	Produces sixteen subbands using Integer Cosine Transform
SYNI (A)	Reconstructs image data from subbands produced by SUBI
DDPCM (A)	Two-dimensional differential PCM coder
DQ (A)	Quantizer for DDPCM
UNDD (A)	Two-dimensional differential PCM decoder
QFINE (A)	Example of a fine quantizer
QMID (A)	Example of a medium-coarse quantizer
QCOARSE (A)	Example of a coarse quantizer
QFINEBIN (A)	Example of a fine quantizer, returns bin number not value
STATS	Returns statistics on reconstructed image quality
SUB3 (A)	Produces a nine band uniform subband decomposition
SYN3 (A)	Reconstructs image data from subbands produced by SUB3
SUBH10 (A)	Produces a ten band octave subband (wavelet) decomposition
SYNH10 (A)	Reconstructs image data from subbands produced by SUBH10
XFORM (A,T)	Performs block transform of image data
IFORM (A,T)	Performs inverse block transform from XFORM coefficients
PERM (A)	Rearranges block transform coefficients into subbands
UNPERM (A)	Rearranges subbanded data into blocks of coefficients
BASIS (A)	Finds the basis images of a matrix using row vectors

The functions are meant to operate on even sized images (e.g., 512 x 512). For an odd size either a row or column of zeros can be added to even up the matrix, or a row or column can be dropped if the data is not critical. The SUB3 and SYN3 functions work on image sizes that are a multiple of 3 only. Other matrix sizes need to be augmented or cropped.

The SUB (subband) and SYN (synthesis) functions for the transformation and rearrangement of coefficients into subbands are explicitly coded for particular transforms. A general method of generating subbands with any transform can be accomplished with XFORM and PERM. The two-dimensional DPCM coder is for compressing the lowest band. The predictor is the same as used in [4]. Example quantization functions are given for the higher bands. Quantizer design is the key to compression performance. Simple, fixed quantizers are not going to give the best performance and some sophisticated bit

allocation is needed to squeeze the most out of an image. To apply compression coding to the quantized data, it may be desirable to assign bin numbers instead of actual bin values to the quantized data (see QFINEBIN for an example).

The subbanded data can be coarsely viewed with the contour plotting function. To view the subbands as an image, it is necessary to shift and scale the values to match the range of the output device, typically 0 to 255.

The BASIS function can be used to explicitly code a particular transform by giving the equations for each transform coefficient. The basis images are returned as submatrices (the same size as the transform matrix) in one large matrix. For the inverse transform equations, take the BASIS of the inverse of the transform matrix (e.g., BASIS(inv(T))) or equivalently, the BASIS of the transpose of the inverse matrix (e.g., BASIS(t')). The inverse basis images are really the basis images using the column vectors of the inverse transform matrix t. A column vector basis function can be obtained from BASIS.M by deleting the line "D=A'" and replacing "D" with "A" in the rest of the function.

An example script for processing a 512 x 512 image matrix stored in the variable named "image" is:

```
subbands=SUBH(image);
m=1:256;n=257:512;
B0=DDPCM(subbands(m,m));
B1=QMID(subbands(m,n));
B2=QMID(subbands(n,m));
B3=QCOARSE(subbands(n,n));
B0=UNDD(B0);
subbands=[B0 B1;B2 B3];
reconstruction=SYNH(subbands);
dif=image-reconstruction;
STATS
```

A cascade of the four band decomposition produces sixteen uniform bands and is equivalent to using the 4x4 Hadamard matrix:

```
subbands=SUBH(SUBH(image));
```

The synthesis function is likewise cascaded:

```
reconstruction=SYNH(SYNH(subbands));
```

If any loss has taken place between the analysis and synthesis stages, some of the reconstructed values may lie outside the allowable range. The synthesis functions replace negative values by zero and values greater than 255 by 255.

Another type of cascade is the octave structure which operates on only the lowest band for the second stage to produce a total of seven bands:

```
subbands=SUBH(image);
m=1:256;
subbands(m,m)=SUBH(subbands(m,m));
```

An example of using the XFORM functions to perform a 4x4 Integer Cosine Transform [5] and inverse transform is:

```
T=[1 1 1 1;2 1 -1 -2;1 -1 -1 1;1 -2 2 -1];
coefficients=XFORM(image,T);
reconstruction=IFORM(coefficients,T);
```

Because the transform matrix is 4x4, the image has to contain an integer number of 4x4 submatrices. The inverse transform matrix is: t=inv(T'); "t" is calculated in the IFORM function, so use T in the call. To get subbands use:

```
subbands=PERM(coefficients)
```

Figure 1 shows a contour plot of a monochrome, 400x512 image of Io. The image appears oblong because the pixels are not square. An odd number of bins were used in the contour plot to obtain a bin around zero in the subband plots. Figure 2 shows a contour plot of the subbands created by SUBG. Figure 3 shows the subbands created by SUBI. Figure 4 shows the subbands created by SUB3 after cropping the image to 399x510. To display the subbands as an image requires shifting the data to all positive values (by adding the most negative value possible) then scaling the result to a range of 0 to 255. The plots merely show the organization of the subbands and do not represent the signal energy in each band.

CONCLUDING REMARKS

Some functions for processing image data were presented for use in MATLAB. These functions are intended to simulate subband/transform image data compression systems. The simulation allows the evaluation of various quantization and bit allocation schemes. Actual compression is accomplished with C programs external to MATLAB.

ACKNOWLEDGEMENTS

Some of these functions were written as part of research at the University of Toledo in partial fulfillment of the Doctor of Philosophy degree [6].

REFERENCES

[1] Clarke, R. J.: Transform Coding of Images. London: Academic Press, 1985.

[2] Simoncelli, Eero P. and Edward H. Adelson: "Subband Transforms." In: Subband Image Coding, John W. Woods (ed.), Boston: Kluwer, 1991.

[3] Vetterli, Martin: "Multirate Filter Banks for Subband Coding." In: Subband Image Coding, John W. Woods (ed.), Boston: Kluwer, 1991.

[4] Gharavi, H. and Ali Tabatabai: "Sub-band Coding of Digital Images Using Two-Dimensional Quadrature Mirror Filtering." SPIE Vol 707 Visual Communications and Image Processing, 1986, pp. 51-61.

[5] Cheung, Kar-Ming, and Kevin Tong: "Proposed Data Compression Schemes for the Galileo S-Band Contingency Mission." In: Proc. 1993 Space and Earth Science Data Compression Workshop, held at Snowbird, Utah, April 2, 1993, NASA CP 3191, pp. 99-109.

[6] Glover, Daniel: Subband/Transform Image Coding for Lossy and Lossless Compression. Ph.D. dissertation, The University of Toledo, December, 1992.

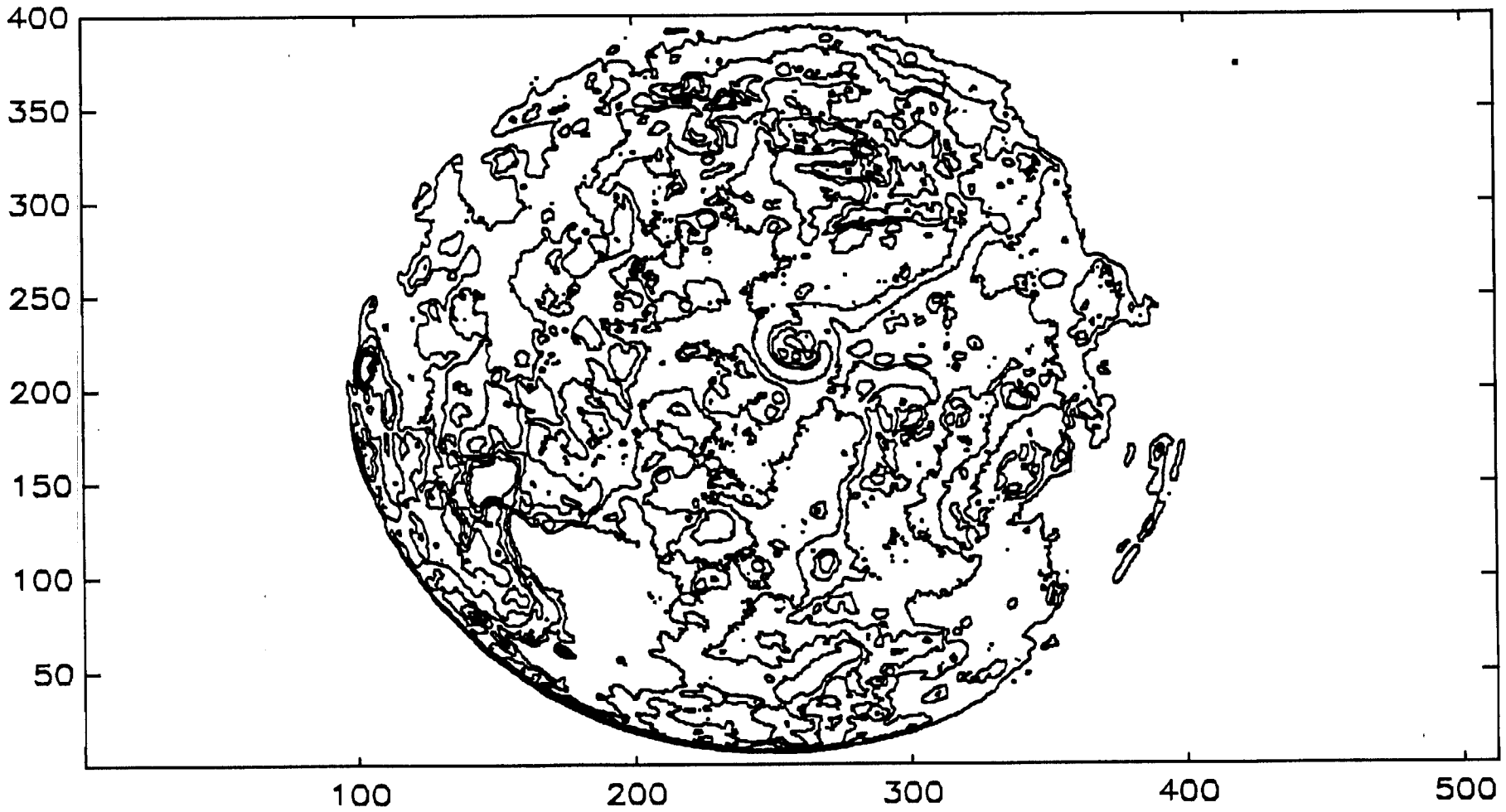

FIGURE 1
Contour Plot of Original Image "Io"

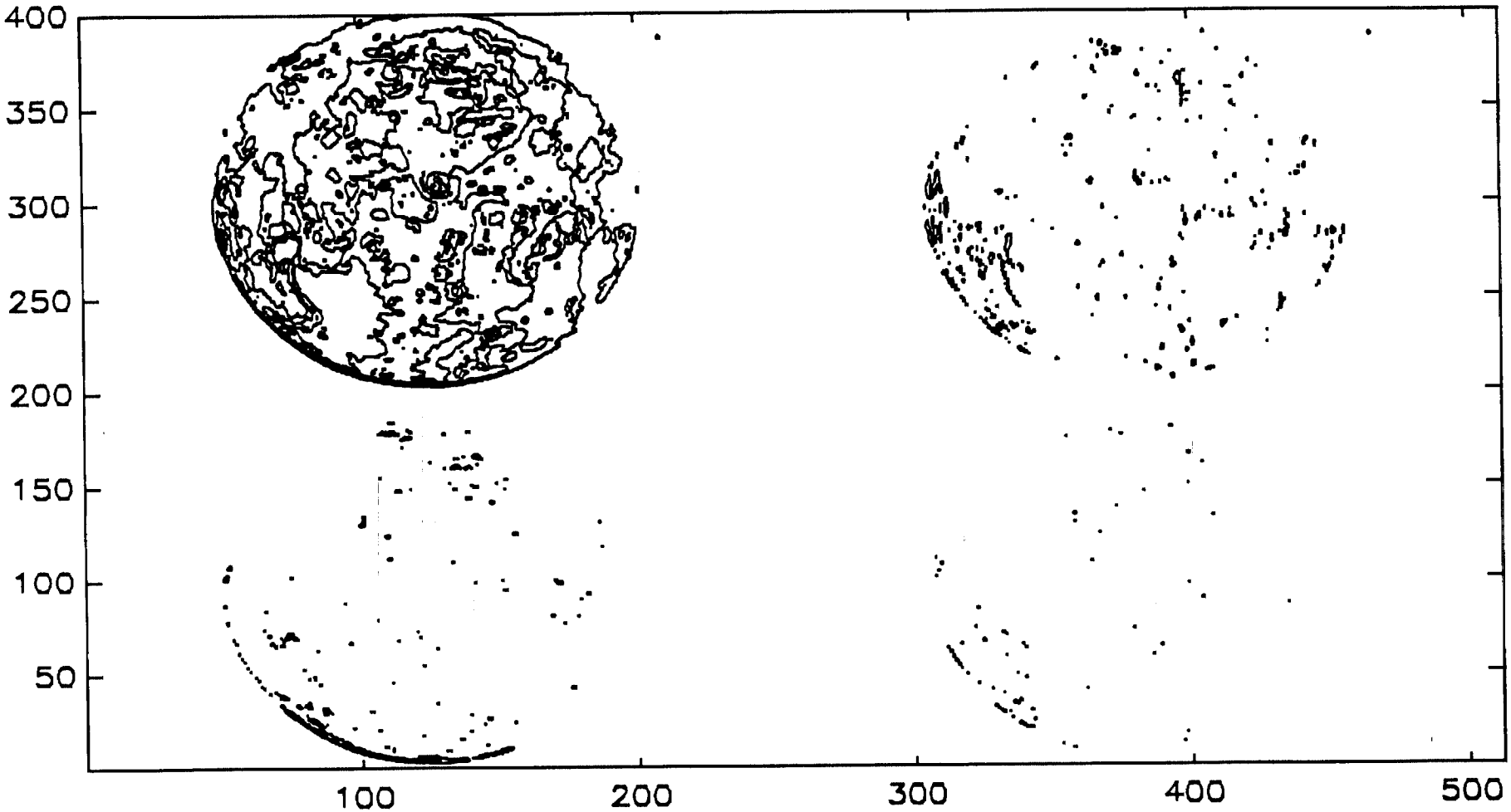

FIGURE 2
Contour Plot of Four Subbands Created by SUBG

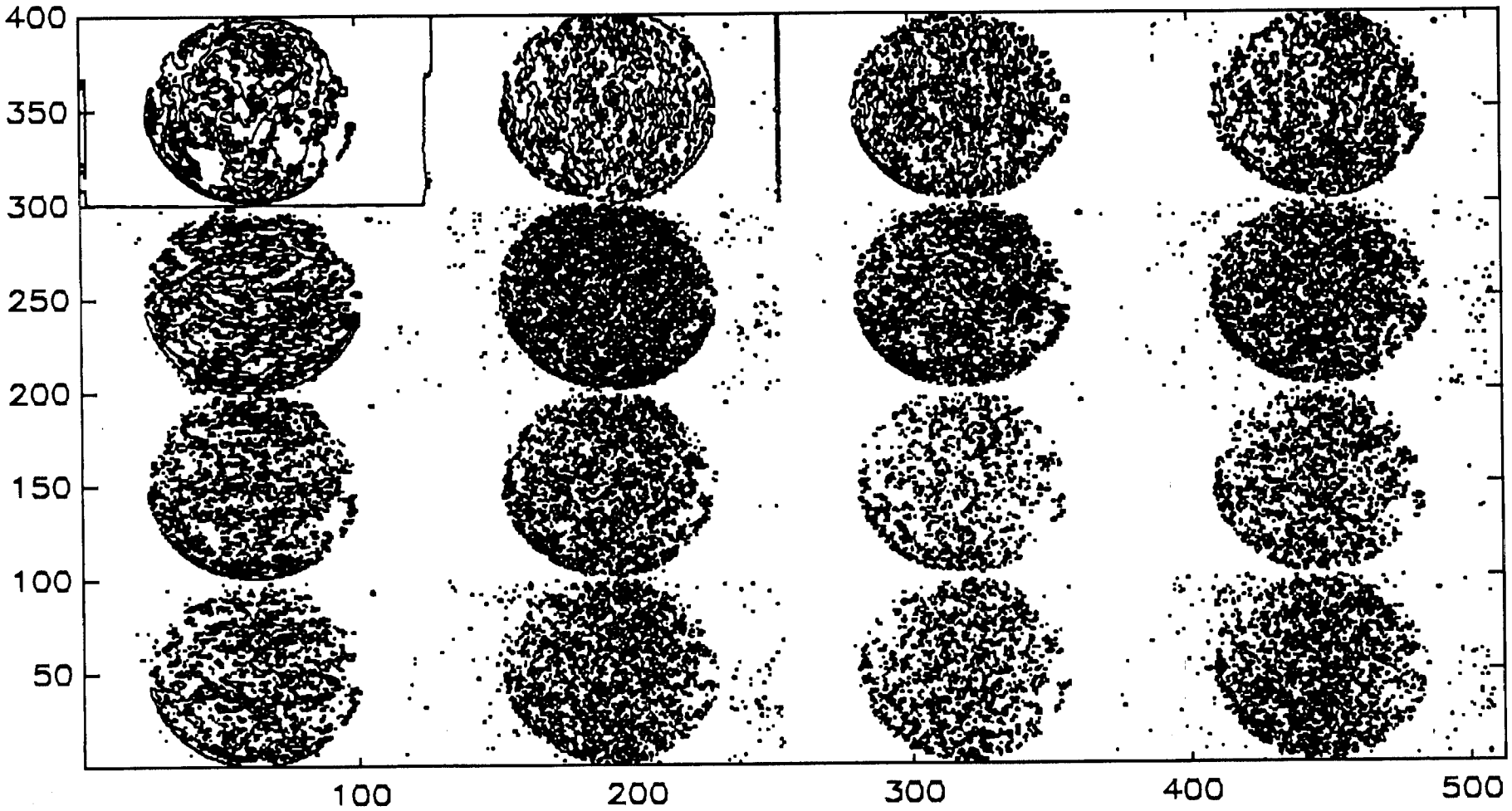

FIGURE 3
Contour Plot of Sixteen Subbands Created by SUBI

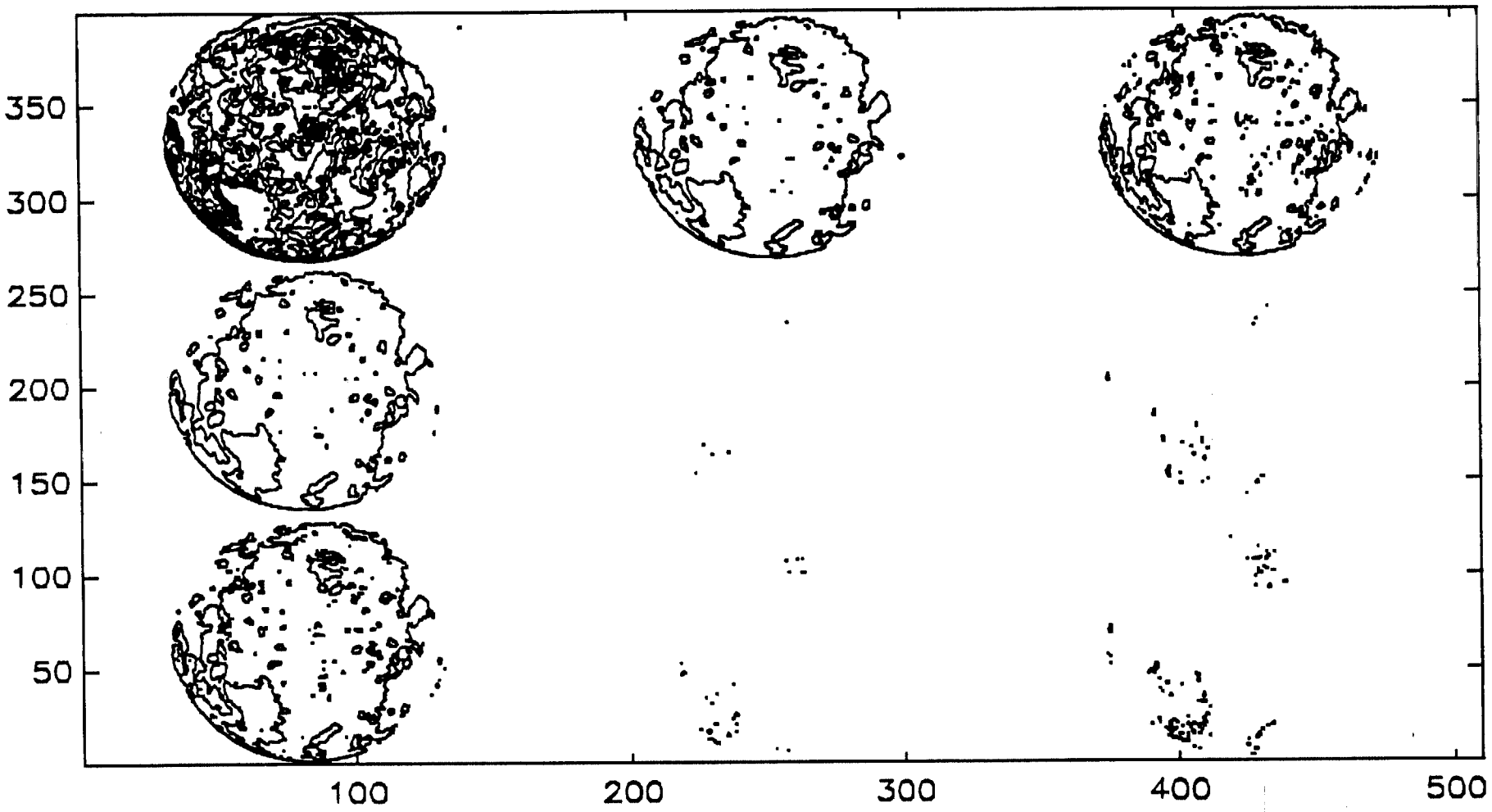

FIGURE 4
Contour Plot of Nine Subbands Created by SUB3

APPENDIX A

LISTINGS OF MATLAB FUNCTIONS FOR IMAGE PROCESSING

Listing of SUBG.M

```
% SUBG.M
% by Daniel Glover
% FINDS THE 2x2 NON-ORTHOGONAL GLOVER TRANSFORM OF A MATRIX A
% PLACES TRANSFORM COEFFICIENTS IN SUBBAND (TILED) FORMAT
%
function X = subg(A)
[m n]=size(A);
%
% setup indices
%
r1=1:2:m;
r2=2:2:m;
c1=1:2:n;
c2=2:2:n;
%
% compute transformed values, place in subbands
%
LL=(A(r2,c2));
HL=(A(r2,c1)-A(r2,c2));
LH=(A(r1,c2)-A(r2,c2));
HH=(A(r1,c1)-A(r1,c2)-A(r2,c1)+A(r2,c2));
%
% place subbands in tiled format
%
X=[LL HL;LH HH];
```

Listing of SYNG.M

```
% SYNG.M
% by Daniel Glover
% SYNTHESIZE NON-ORTHOGONAL TRANSFORM
% OF A SUBBANDED MATRIX, A
% IN SUBBAND (TILED) FORMAT
%
function X = syng(A)
[m n]=size(A);
%
% setup indices
%
a=m/2; b=n/2;
r1=1:1:a;
r2=(a+1):1:m;
c1=1:1:b;
c2=(b+1):1:n;
R1=1:2:m;
R2=2:2:m;
C1=1:2:n;
C2=2:2:n;
%
% compute reconstructed pixels
%
X(R2,C2)=A(r1,c1);
X(R1,C2)=(A(r2,c1)+A(r1,c1));
X(R2,C1)=(A(r1,c2)+A(r1,c1));
X(R1,C1)=(A(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2));
%
X=overflow(X);
```

Listing of OVERFLOW.M

```
% OVERFLOW.M by Daniel Glover
%
% THIS FUNCTION HANDLES OUT OF RANGE CASES AFTER A RECONSTRUCTION
% FUNCTION IS USED.  IT SHOWS UP IN SYN.M FUNCTIONS, UNDD.M, AND IFORM.M.
%
%
% set overflow values to 255
%
```

Listing of SYNG.M

```
% SYNG.M
% by Daniel Glover
% SYNTHESIZE NON-ORTHOGONAL TRANSFORM
% OF A SUBBANDED MATRIX, A
% IN SUBBAND (TILED) FORMAT
%
function X = syng(A)
[m n]=size(A);
%
% setup indices
%
a=m/2; b=n/2;
r1=1:1:a;
r2=(a+1):1:m;
c1=1:1:b;
c2=(b+1):1:n;
R1=1:2:m;
R2=2:2:m;
C1=1:2:n;
C2=2:2:n;
%
% compute reconstructed pixels
%
X(R2,C2)=A(r1,c1);
X(R1,C2)=(A(r2,c1)+A(r1,c1));
X(R2,C1)=(A(r1,c2)+A(r1,c1));
X(R1,C1)=(A(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2));
%
X=overflow(X);
```

Listing of OVERFLOW.M

```
% OVERFLOW.M by Daniel Glover
%
% THIS FUNCTION HANDLES OUT OF RANGE CASES AFTER A RECONSTRUCTION
% FUNCTION IS USED.  IT SHOWS UP IN SYN.M FUNCTIONS, UNDD.M, AND IFORM.M.
% THIS FUNCTION WAS WRITTEN FOR MATLAB386 AND WILL PROBABLY HAVE
% A SIMPLER IMPLEMENTATION IN MATLAB 4.0.
%
% set overflow values to 255
```

```
%
o=find(X>255);
over=length(o)
%
if over==1,
        X(o)=255;
else
        X(o)=255*ones(o);
end
%
% set underflow values to 0
%
u=find(X<0);
under=length(u)
if under==1,
    X(u)=0;
else
    X(u)=zeros(u);
end
```

Listing of SUBH.M

```
% SUBH.M by Daniel Glover
% FINDS 2x2 WALSH-HADAMARD TRANSFORM OF A MATRIX, A,
% PLACES TRANSFORM COEFFICIENTS IN SUBBAND (TILED) FORMAT
% INCLUDES SCALING FACTOR OF 1/4
%
function X = subh(A)
[m n]=size(A);
r1=1:2:m;
r2=2:2:m;
c1=1:2:n;
c2=2:2:n;
LL=(A(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2))/2;
LH=(A(r1,c1)+A(r1,c2)-A(r2,c1)-A(r2,c2))/2;
HL=(A(r1,c1)-A(r1,c2)+A(r2,c1)-A(r2,c2))/2;
HH=(A(r1,c1)-A(r1,c2)-A(r2,c1)+A(r2,c2))/2;
X=[LL HL;LH HH];
```

Listing of SYNH.M

```
% SYNH.M by Daniel Glover
% SYNTHESIZE (RECONSTRUCT) USING WHT ON A MATRIX, A, OF
% TRANSFORMED DATA IN SUBBAND (TILED) FORMAT
%
function X = synh(A)
[m n]=size(A);
a=m/2;
b=n/2;
r1=1:1:a;
r2=(a+1):1:m;
c1=1:1:b;
c2=(b+1):1:n;
R1=1:2:m;
R2=2:2:m;
C1=1:2:n;
C2=2:2:n;
%
X(R1,C1)=(A(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2))/2;
X(R2,C1)=(A(r1,c1)+A(r1,c2)-A(r2,c1)-A(r2,c2))/2;
X(R1,C2)=(A(r1,c1)-A(r1,c2)+A(r2,c1)-A(r2,c2))/2;
X(R2,C2)=(A(r1,c1)-A(r1,c2)-A(r2,c1)+A(r2,c2))/2;
%
X=overflow(X);
```

Listing of SUBI.M

```
% SUBI.M by Daniel Glover
% Finds 4x4 ICT OF A MATRIX, A
% WHERE T=[1 1 1 1;2 1 -1 -2;1 -1 -1 1;1 -2 2 -1]
%
function X = subI(A)
[m n]=size(A);
%
r1=1:4:m;
r2=2:4:m;
r3=3:4:m;
r4=4:4:m;
%
c1=1:4:n;
c2=2:4:n;
c3=3:4:n;
c4=4:4:n;
%
a=m/4;
b=(2*m)/4;
c=(3*m)/4;
d=m;
%
%
aa=n/4;
bb=(2*n)/4;
cc=(3*n)/4;
dd=n;
%
%
R1=1:1:a;
R2=(a+1):1:b;
R3=(b+1):1:c;
R4=(c+1):1:d;
%
%
C1=1:1:aa;
C2=(aa+1):1:bb;
C3=(bb+1):1:cc;
C4=(cc+1):1:dd;
%
%
%
%
%
%
```

```
a=A(r1,c1)+A(r2,c1)+A(r3,c1)+A(r4,c1);
b=A(r1,c2)+A(r2,c2)+A(r3,c2)+A(r4,c2);
c=A(r1,c3)+A(r2,c3)+A(r3,c3)+A(r4,c3);
d=A(r1,c4)+A(r2,c4)+A(r3,c4)+A(r4,c4);
%
X(R1,C1)=a+b+c+d;
X(R1,C2)=2*a+b-c-2*d;
X(R1,C3)=a-b-c+d;
X(R1,C4)=a-2*b+2*c-d;
%
a=2*A(r1,c1)+A(r2,c1)-A(r3,c1)-2*A(r4,c1);
b=2*A(r1,c2)+A(r2,c2)-A(r3,c2)-2*A(r4,c2);
c=2*A(r1,c3)+A(r2,c3)-A(r3,c3)-2*A(r4,c3);
d=2*A(r1,c4)+A(r2,c4)-A(r3,c4)-2*A(r4,c4);
%
X(R2,C1)=a+b+c+d;
X(R2,C2)=2*a+b-c-2*d;
X(R2,C3)=a-b-c+d;
X(R2,C4)=a-2*b+2*c-d;
%
a=A(r1,c1)-A(r2,c1)-A(r3,c1)+A(r4,c1);
b=A(r1,c2)-A(r2,c2)-A(r3,c2)+A(r4,c2);
c=A(r1,c3)-A(r2,c3)-A(r3,c3)+A(r4,c3);
d=A(r1,c4)-A(r2,c4)-A(r3,c4)+A(r4,c4);
%
X(R3,C1)=a+b+c+d;
X(R3,C2)=2*a+b-c-2*d;
X(R3,C3)=a-b-c+d;
X(R3,C4)=a-2*b+2*c-d;
%
a=A(r1,c1)-2*A(r2,c1)+2*A(r3,c1)-A(r4,c1);
b=A(r1,c2)-2*A(r2,c2)+2*A(r3,c2)-A(r4,c2);
c=A(r1,c3)-2*A(r2,c3)+2*A(r3,c3)-A(r4,c3);
d=A(r1,c4)-2*A(r2,c4)+2*A(r3,c4)-A(r4,c4);
%
X(R4,C1)=a+b+c+d;
X(R4,C2)=2*a+b-c-2*d;
X(R4,C3)=a-b-c+d;
X(R4,C4)=a-2*b+2*c-d;
```

Listing of SYNI.M

```
% SYNI.M by Daniel Glover
% SYNTHESIZE IMAGE DATA FROM ICT DATA MATRIX, A
%
function X = syni(A)
[m n] = size(A);
%
a=m/4;
b=m/2;
c=(3*m)/4;
d=m;
%
aa=n/4;
bb=n/2;
cc=(3*n)/4;
dd=n;
%
r1=1:1:a;
r2=(a+1):1:b;
r3=(b+1):1:c;
r4=(c+1):1:d;
%
c1=1:1:aa;
c2=(aa+1):1:bb;
c3=(bb+1):1:cc;
c4=(cc+1):1:dd;
%
R1=1:4:m;
R2=2:4:m;
R3=3:4:m;
R4=4:4:m;
%
C1=1:4:n;
C2=2:4:n;
C3=3:4:n;
C4=4:4:n;
%
a=.0625*A(r1,c1)+.05*A(r2,c1)+.0625*A(r3,c1)+.025*A(r4,c1);
b=.05*A(r1,c2)+.04*A(r2,c2)+.05*A(r3,c2)+.02*A(r4,c2);
c=.0625*A(r1,c3)+.05*A(r2,c3)+.0625*A(r3,c3)+.025*A(r4,c3);
d=.025*A(r1,c4)+.02*A(r2,c4)+.025*A(r3,c4)+.01*A(r4,c4);
%
X(R1,C1)=(a+b+c+d);
X(R1,C2)=a+b/2-c-2*d;
X(R1,C3)=a-b/2-c+2*d;
X(R1,C4)=a-b+c-d;
```

```
%
a=.0625*A(r1,c1)+.025*A(r2,c1)-.0625*A(r3,c1)-.05*A(r4,c1);
b=.05*A(r1,c2)+.02*A(r2,c2)-.05*A(r3,c2)-.04*A(r4,c2);
c=.0625*A(r1,c3)+.025*A(r2,c3)-.0625*A(r3,c3)-.05*A(r4,c3);
d=.025*A(r1,c4)+.01*A(r2,c4)-.025*A(r3,c4)-.02*A(r4,c4);
%
X(R2,C1)=a+b+c+d;
X(R2,C2)=a+b/2-c-2*d;
X(R2,C3)=a-b/2-c+2*d;
X(R2,C4)=a-b+c-d;
%
a=.0625*A(r1,c1)-.025*A(r2,c1)-.0625*A(r3,c1)+.05*A(r4,c1);
b=.05*A(r1,c2)-.02*A(r2,c2)-.05*A(r3,c2)+.04*A(r4,c2);
c=.0625*A(r1,c3)-.025*A(r2,c3)-.0625*A(r3,c3)+.05*A(r4,c3);
d=.025*A(r1,c4)-.01*A(r2,c4)-.025*A(r3,c4)+.02*A(r4,c4);
%
X(R3,C1)=a+b+c+d;
X(R3,C2)=a+b/2-c-2*d;
X(R3,C3)=a-b/2-c+2*d;
X(R3,C4)=a-b+c-d;
%
a=.0625*A(r1,c1)-.05*A(r2,c1)+.0625*A(r3,c1)-.025*A(r4,c1);
b=.05*A(r1,c2)-.04*A(r2,c2)+.05*A(r3,c2)-.02*A(r4,c2);
c=.0625*A(r1,c3)-.05*A(r2,c3)+.0625*A(r3,c3)-.025*A(r4,c3);
d=.025*A(r1,c4)-.02*A(r2,c4)+.025*A(r3,c4)-.01*A(r4,c4);
%
X(R4,C1)=(a+b+c+d);
X(R4,C2)=a+b/2-c-2*d;
X(R4,C3)=a-b/2-c+2*d;
X(R4,C4)=a-b+c-d;
%
X=overflow(X);
```

Listing of DDPCM.M

```
% DDPCM.M by Daniel Glover
% TWO DIMENSIONAL DPCM OF AN m x n MATRIX,
% Prediction=.5H+.25V+.25D from Gharavi and Tabatabai
% THE FIRST COLUMN AND TOP ROW IS 1-D DPCM ONLY,
% THE LAST COLUMN IS SECOND ORDER 2-D DPCM ONLY
% This function would probably be faster if implemented in C
% since it operates on individual values in a matrix
% This function uses a quantizer called DQ.M
%
function X=ddpcm(A)
[m n]=size(A);
%
% X is the dpcm result, D is the difference between the original value (A)
% and the reconstructed values (R), Q is the quantized difference
% The first column and the first row is 1-D dpcm only
%
X(1,1)=A(1,1);
R(1,1)=A(1,1);
%
% Initialization complete; first column 1-D DPCM
%
for i=2:m
X(i,1)=dq(A(i,1)-R(i-1,1));
R(i,1)=R(i-1,1)+X(i,1);
end
%
% First row 1-D DPCM
%
for j=2:n
X(1,j)=dq(A(1,j)-R(1,j-1));
R(1,j)=X(1,j)+R(1,j-1);
end
%
% Calculate the basic differences, actual minus predicted value
%
for i=2:m
for j=2:n-1
P(i,j)=R(i,j-1)/2+R(i-1,j)/4+R(i-1,j+1)/4;
X(i,j)=dq(A(i,j)-P(i,j));
R(i,j)=X(i,j)+P(i,j);
end
% The last column is second order 2-D DPCM
X(i,n)=dq( A(i,n)-0.5*(R(i,n-1)+R(i-1,n)) );
R(i,n)=X(i,n)+0.5*(R(i,n-1)+R(i-1,n));
end
```

Listing of QD.M

```
% QD.M by Daniel Glover
% FINE QUANTIZER FOR DDPCM AND SUBBANDS
% 31 bins
%
function x=qd(A)
[m n]=size(A);
%
x=17*round(A/17);
%
f=find(abs(A)>194);
x(f)=sign(A(f))*212;
%
f=find(abs(A)>229);
x(f)=sign(A(f))*255;
%
f=find(abs(A)<13);
x(f)=5*round(A(f)/5);
```

Listing of UNDD.M

```
% UNDD.M by Daniel Glover
% DECODE TWO-D DPCM OF AN m x n MATRIX
%
function X=undd(A)
[m n]=size(A);
%
% first row is 1-D DPCM
%
X(1,1)=A(1,1);
for j=2:n
X(1,j)=A(1,j)+X(1,j-1);
end
%
% first column is 1-D DPCM
%
for i=2:m
X(i,1)=A(i,1)+X(i-1,1);
        %
        % reconstruct third order DPCM
        %
```

```
        for j=2:n-1
        X(i,j) = A(i,j) + 0.5*X(i,j-1) + 0.25*X(i-1,j) + 0.25*X(i-1,j+1);
        end
%
% last column is second order 2-D DPCM
%
X(i,n)=A(i,n)+(X(i,n-1)+X(i-1,n))*0.5;
end
%
% round result to integer value
%
X=round(X);
%
X=overflow(X);
```

Listing of QCOARSE.M

```
% QCOARSE.M by Daniel Glover
% COARSE QUANTIZATION FOR HIGH BANDS
% 7 levels from -255 to 255
%
function X=qcoarse(A)
[m n]=size(A);
%
X=127*round(A/127);
%
f=find(abs(A)<64);
X(f)=sign(A(f))*20;
f=find(abs(A)<8);
X(f)=zeros(f);
%
% OPTIONAL HIGH BINS
%       f=find(abs(A)>247),
%       X(f)=sign(A(f))*248;
```

Listing of QMID.M

```
% QMID.M by Daniel Glover
% QUANTIZATION FOR MID BANDS
% 15 levels from -255 to 255
%
function X=qmid(A)
[m n]=size(A);
%
X=41*round(A/41);
%
f=find(abs(A)<32);
X(f)=sign(A(f))*20;
%
f=find(abs(A)<8);
X(f)=zeros(f);
% OPTIONAL HIGH BINS
%       f=find(abs(A)>247);
%       X(f)=sign(A(f))*248;
```

Listing of QFINE.M

```
% QFINE.M by Daniel Glover
% FINE QUANTIZATION FOR SUBBANDS
% 63 levels
%
function X=qfine(A)
[m n]=size(A);
%
X=5*round(A/5);
%
f=find(abs(A)<4);
X(f)=zeros(f);
%
f=find(abs(A)>31);
X(f)=9*round(A(f)/9);
```

Listing of QFINEBIN.M

```
% QFINEBIN.M by Daniel Glover
% RETURNS BIN NUMBER INSTEAD OF BIN VALUE
% BIN #0 IS DEADBAND BIN
% 63 levels
function X=qfinebin(A)
[m n]=size(A);
X=round(A/5);
%
f=find(abs(A)<4);
X(f)=zeros(f);
%
f=find(abs(A>31);
X(f)=round(A(f)/9)+(sign(A(f))*4);
%
f=find(X==0);
X=X+32;
X(f)=zeros(f);
```

Listing of STATS.M

```
% STATS.M by Daniel Glover
%
% SCRIPT FINDS STD. DEV., SNR, AND MSE FROM ERROR MATRIX
% CALCULATE ERROR MATRIX BEFORE USING THIS SCRIPT
% REQUIRES VARIABLE: dif=(original_image_matrix)-(reconstructed_image_matrix);
%
% CALCULATE STANDARD DEVIATION OF ERROR MATRIX
%
s=std(dif(:))
%
% CALCULATE Peak Signal-to-Noise Ratio (PSNR) IN dB
%
PSNR=(10*log(255^2/s^2))/log(10);
%
% CALCULATE Mean Square Error (mse),
% difsq is matrix of square of error matrix elements, mse is mean of difsq
%
difsq=dif.^2;
mse=mean(difsq(:))
```

Listing of SUB3.M

```
% SUB3.M by Daniel Glover
% PERFORMS NINE BAND UNIFORM DECOMPOSITION
% OF A MATRIX AND ARRANGES IN SUBBANDS
% TRANSFORM MATRIX IS:
%
%              1  2  1
%             -1  1 -1
%              1 -1 -1
%
function X = subh(A)
[m n]=size(A);
r1=1:3:m;
r2=2:3:m;
r3=3:3:m;
c1=1:3:n;
c2=2:3:n;
c3=3:3:n;
%
a=A(r1,c1)+2*A(r2,c1)+A(r3,c1);
b=A(r1,c2)+2*A(r2,c2)+A(r3,c2);
c=A(r1,c3)+2*A(r2,c3)+A(r3,c3);
B0=a+2*b+c;
B1=-a+b-c;
B2=-a-b+c;
%
a=-A(r1,c1)+A(r2,c1)-A(r3,c1);
b=-A(r1,c2)+A(r2,c2)-A(r3,c2);
c=-A(r1,c3)+A(r2,c3)-A(r3,c3);
B3=a+2*b+c;
B4=-a+b-c;
B5=-a-b+c;
%
a=-A(r1,c1)-A(r2,c1)+A(r3,c1);
b=-A(r1,c2)-A(r2,c2)+A(r3,c2);
c=-A(r1,c3)-A(r2,c3)+A(r3,c3);
B6=a+2*b+c;
B7=-a+b-c;
B8=-a-b+c;
%
X=[B0 B1 B2;B3 B4 B5;B6 B7 B8];
```

Listing of SUB3.M

```
% SUB3.M by Daniel Glover
% PERFORMS NINE BAND UNIFORM DECOMPOSITION
% OF A MATRIX AND ARRANGES IN SUBBANDS
% TRANSFORM MATRIX IS:
%
%              1  2  1
%             -1  1 -1
%             -1 -1  1
%
% IMAGE DIMENSIONS MUST BE A FACTOR OF THREE
%
function X = subh(A)
[m n]=size(A);
r1=1:3:m;
r2=2:3:m;
r3=3:3:m;
c1=1:3:n;
c2=2:3:n;
c3=3:3:n;
%
a=A(r1,c1)+2*A(r2,c1)+A(r3,c1);
b=A(r1,c2)+2*A(r2,c2)+A(r3,c2);
c=A(r1,c3)+2*A(r2,c3)+A(r3,c3);
B0=a+2*b+c;
B1=-a+b-c;
B2=-a-b+c;
%
a=-A(r1,c1)+A(r2,c1)-A(r3,c1);
b=-A(r1,c2)+A(r2,c2)-A(r3,c2);
c=-A(r1,c3)+A(r2,c3)-A(r3,c3);
B3=a+2*b+c;
B4=-a+b-c;
B5=-a-b+c;
%
a=-A(r1,c1)-A(r2,c1)+A(r3,c1);
b=-A(r1,c2)-A(r2,c2)+A(r3,c2);
c=-A(r1,c3)-A(r2,c3)+A(r3,c3);
B6=a+2*b+c;
B7=-a+b-c;
B8=-a-b+c;
%
X=[B0 B1 B2;B3 B4 B5;B6 B7 B8];
```

Listing of SYN3.M

```
% SYN3.M by Daniel Glover
% RECONSTRUCT IMAGE DATA FROM NINE SUBBANDS
% INVERSE TRANSFORM MATRIX IS:
%
%        0   1/3   1/3
%      -1/2  1/3  -1/6
%      -1/2   0    1/2
%
function X = syn3(A)
[m n]=size(A);
m3=m/3;
n3=n/3;
r1=1:1:m3;
r2=(m3+1):1:2*m3;
r3=(2*m3+1):1:m;
c1=1:1:n3;
c2=(n3+1):1:2*n3;
c3=(2*n3+1):1:n;
R1=1:3:m;
R2=2:3:m;
R3=3:3:m;
C1=1:3:n;
C2=2:3:n;
C3=3:3:n;
X(R1,C1)=(A(r2,c2)+A(r2,c3)+A(r3,c2)+A(r3,c3))/4;
X(R2,C1)=(-A(r1,c2)-A(r1,c3)-A(r2,c2)-A(r2,c3))/6;
X(R3,C1)=(-(A(r1,c2)+A(r1,c3))/6) + ((A(r2,c2)+A(r2,c3))/12) - ((A(r3,c2)+A(r3,c3))/4);
%
X(R1,C2)=-(A(r2,c1)+A(r2,c2)+A(r3,c1)+A(r3,c2))/6;
X(R2,C2)=(A(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2))/9;
X(R3,C2)=X(R2,C2)+(-A(r2,c1)-A(r2,c2)+A(r3,c1)+A(r3,c2))/6;
%
X(R1,C3)=(-A(r2,c1)-A(r3,c1))/6 + (A(r2,c2)+A(r3,c2))/12 - (A(r2,c3)+A(r3,c3))/4;
X(R2,C3)=(A(r1,c1)+A(r2,c1))/9 - (A(r1,c2)+A(r2,c2))/18 + (A(r1,c3)+A(r2,c3))/6;
X(R3,C3)=A(r1,c1)/9-A(r2,c1)/18+A(r3,c1)/6-A(r1,c2)/18+A(r2,c2)/36
         -A(r3,c2)/12+A(r1,c3)/6-A(r2,c3)/12+A(r3,c3)/4;
%
X=overflow(X);
```

Listing of SYN3.M

```
% SYN3.M by Daniel Glover
% RECONSTRUCT IMAGE DATA FROM NINE SUBBANDS
% INVERSE TRANSFORM MATRIX IS:
%
%       1/3  1/3    0
%      -1/6  1/3  -1/2
%       1/2   0   -1/2
%
function X = syn3(A)
[m n]=size(A);
m3=m/3;
n3=n/3;
r1=1:1:m3;
r2=(m3+1):1:2*m3;
r3=(2*m3+1):1:m;
c1=1:1:n3;
c2=(n3+1):1:2*n3;
c3=(2*n3+1):1:n;
R1=1:3:m;
R2=2:3:m;
R3=3:3:m;
C1=1:3:n;
C2=2:3:n;
C3=3:3:n;
X(R1,C1)=(A(r2,c2)+A(r2,c3)+A(r3,c2)+A(r3,c3))/4;
X(R2,C1)=(-A(r1,c2)-A(r1,c3)-A(r2,c2)-A(r2,c3))/6;
X(R3,C1)=(-(A(r1,c2)+A(r1,c3))/6) + ((A(r2,c2)+A(r2,c3))/12) - ((A(r3,c2)+A(r3,c3))/4);
%
X(R1,C2)=-(A(r2,c1)+A(r2,c2)+A(r3,c1)+A(r3,c2))/6;
X(R2,C2)=(A(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2))/9;
X(R3,C2)=X(R2,C2)+(-A(r2,c1)-A(r2,c2)+A(r3,c1)+A(r3,c2))/6;
%
X(R1,C3)=(-A(r2,c1)-A(r3,c1))/6 + (A(r2,c2)+A(r3,c2))/12 - (A(r2,c3)+A(r3,c3))/4;
X(R2,C3)=(A(r1,c1)+A(r2,c1))/9 - (A(r1,c2)+A(r2,c2))/18 + (A(r1,c3)+A(r2,c3))/6;
X(R3,C3)=A(r1,c1)/9-A(r2,c1)/18+A(r3,c1)/6-A(r1,c2)/18+A(r2,c2)/36
A(r3,c2)/12+A(r1,c3)/6-A(r2,c3)/12+A(r3,c3)/4;
%
X=overflow(X);
```

Listing of SUBH10.M

```
% SUBH10.M by Daniel Glover
% FINDS 10 BAND (OCTAVE SPLIT) WALSH-HADAMARD (SIMPLE WAVELET)
% TRANSFORM OF A MATRIX, A
function X = subh10(A)
[m n]=size(A);
r1=1:2:m;
r2=2:2:m;
c1=1:2:n;
c2=2:2:n;
LL=(A(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2))/2;
LH=(A(r1,c1)+A(r1,c2)-A(r2,c1)-A(r2,c2))/2;
HL=(A(r1,c1)-A(r1,c2)+A(r2,c1)-A(r2,c2))/2;
HH=(A(r1,c1)-A(r1,c2)-A(r2,c1)+A(r2,c2))/2;
%
r1=1:2:m/2;
r2=2:2:m/2;
c1=1:2:n/2;
c2=2:2:n/2;
LLLL=(LL(r1,c1)+LL(r1,c2)+LL(r2,c1)+LL(r2,c2))/2;
LLLH=(LL(r1,c1)+LL(r1,c2)-LL(r2,c1)-LL(r2,c2))/2;
LLHL=(LL(r1,c1)-LL(r1,c2)+LL(r2,c1)-LL(r2,c2))/2;
LLHH=(LL(r1,c1)-LL(r1,c2)-LL(r2,c1)+LL(r2,c2))/2;
%
r1=1:2:m/4;
r2=2:2:m/4;
c1=1:2:n/4;
c2=2:2:n/4;
LLLLLL=(LLLL(r1,c1)+LLLL(r1,c2)+LLLL(r2,c1)+LLLL(r2,c2))/2;
LLLLLH=(LLLL(r1,c1)+LLLL(r1,c2)-LLLL(r2,c1)-LLLL(r2,c2))/2;
LLLLHL=(LLLL(r1,c1)-LLLL(r1,c2)+LLLL(r2,c1)-LLLL(r2,c2))/2;
LLLLHH=(LLLL(r1,c1)-LLLL(r1,c2)-LLLL(r2,c1)+LLLL(r2,c2))/2;
%
LLLL=[LLLLLL LLLLHL;LLLLLH LLLLHH];
LL=[LLLL LLHL;LLLH LLHH];
X=[LL HL;LH HH];
```

Listing of SYNH10.M

```
% SYNH10.M by Daniel Glover
% SYNTHESIZE FROM A 10 BAND, OCTAVE, WALSH-HADAMARD (SIMPLE
% WAVELET) TRANSFORMED MATRIX, A
%
function X = synh10(A)
[m n]=size(A);
%
%
%
a=m/8;
b=n/8;
r1=1:1:a;
r2=(a+1):1:m/4;
c1=1:1:b;
c2=(b+1):1:n/4;
R1=1:2:m/4;
R2=2:2:m/4;
C1=1:2:n/4;
C2=2:2:n/4;
%
X(R1,C1)=(A(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2))/2;
X(R2,C1)=(A(r1,c1)+A(r1,c2)-A(r2,c1)-A(r2,c2))/2;
X(R1,C2)=(A(r1,c1)-A(r1,c2)+A(r2,c1)-A(r2,c2))/2;
X(R2,C2)=(A(r1,c1)-A(r1,c2)-A(r2,c1)+A(r2,c2))/2;
%
%
%
%
a=m/4;
b=n/4;
r1=1:1:a;
r2=(a+1):1:m/2;
c1=1:1:b;
c2=(b+1):1:n/2;
R1=1:2:m/2;
R2=2:2:m/2;
C1=1:2:n/2;
C2=2:2:n/2;
%
Y(R1,C1)=(X(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2))/2;
Y(R2,C1)=(X(r1,c1)+A(r1,c2)-A(r2,c1)-A(r2,c2))/2;
Y(R1,C2)=(X(r1,c1)-A(r1,c2)+A(r2,c1)-A(r2,c2))/2;
Y(R2,C2)=(X(r1,c1)-A(r1,c2)-A(r2,c1)+A(r2,c2))/2;
%
%
```

```
%
%
a=m/2;
b=n/2;
r1=1:1:a;
r2=(a+1):1:m;
c1=1:1:b;
c2=(b+1):1:n;
R1=1:2:m;
R2=2:2:m;
C1=1:2:n;
C2=2:2:n;
%
X(R1,C1)=(Y(r1,c1)+A(r1,c2)+A(r2,c1)+A(r2,c2))/2;
X(R2,C1)=(Y(r1,c1)+A(r1,c2)-A(r2,c1)-A(r2,c2))/2;
X(R1,C2)=(Y(r1,c1)-A(r1,c2)+A(r2,c1)-A(r2,c2))/2;
X(R2,C2)=(Y(r1,c1)-A(r1,c2)-A(r2,c1)+A(r2,c2))/2;
%
X=overflow(X);
```

Listing of XFORM.M

```
% XFORM.M by Daniel Glover
% TWO-DIMENSIONAL, SEPARABLE, FORWARD BLOCK TRANSFORM OF A
% MATRIX, A, BY TRANSFORM MATRIX, T
%
function  X = xform(A,T)
[m,n]=size(A);
[u,v]=size(T);
a=1:u;b=1:v;
for i=1:m/u;
        for j=1:n/v;
        X(a,b)=T*A(a,b)*T';
        b=b+v;
        end
a=a+u;b=1:v;
end
```

Listing of IFORM.M

```
% IFORM.M by Daniel Glover
% TWO-DIMENSIONAL INVERSE TRANSFORM OF A MATRIX, A,
% BY TRANSFORM MATRIX t WHICH IS CALCULATED FROM THE FORWARD
% TRANSFORM MATRIX, T, BY t=inv(T')
%
function  X = iform(A,T)
t=inv(T')
[m,n]=size(A);
[u,v]=size(t);
a=1:u;b=1:v;
for i=1:m/u;
        for j=1:n/v;
        X(a,b)=t'*A(a,b)*t;
        b=b+v;
        end
a=a+u;b=1:v;
end
% check for out of range
%
X=overflow(X);
```

Listing of PERM.M

```
% PERM.M by Daniel Glover
% PERMUTATION USED TO PLACE TRANSFORM COEFFICIENTS INTO
% SUBBANDS
% WORKS ON EVEN SIZED BLOCKS, MOVES ODD NUMBERED
% ELEMENTS TO LEFT AND TOP, EVEN NUMBERED ELEMENTS TO RIGHT
% AND BOTTOM OF MATRIX
%
function X = perm(A)
[m n]=size(A);
i=1:2:m;
B((i+1)/2,:)=A(i,:);
%
i=2:2:m;
B((m/2)+i/2,:)=A(i,:);
%
j=1:2:n;
X(:,(j+1)/2)=B(:,j);
%
j=2:2:n;
X(:,(n/2)+j/2)=B(:,j);
```

Listing of UNPERM.M

```
% UNPERM.M by Daniel Glover
% PERMUTATION TO PUT SUBBAND VALUES INTO BLOCKS, RESTORES BLOCK
% TRANSFORM ORDER FROM SUBBAND ORDER OBTAINED BY PERM.M
%
function X = unperm(A)
[m n]=size(A);
i=1:2:m;
B(i,:)=A((i+1)/2,:);
%
i=2:2:m;
B(i,:)=A((m/2)+i/2,:);
%
j=1:2:n;
X(:,j)=B(:,(j+1)/2);
%
j=2:2:n;
X(:,j)=B(:,(n/2)+j/2);
```

Listing of BASIS.M

```
% BASIS.M by Daniel Glover
%
% BASIS(A) FINDS THE BASIS IMAGES OF A MATRIX, A, FROM THE ROW BASIS
% VECTORS AND RETURNS THEM AS A SINGLE MATRIX MADE UP OF
% SUBMATRICES OF ALL THE BASIS IMAGES,
% THE BASIS IMAGES ARE THE SAME SIZE AS THE ORIGINAL MATRIX, A.
%
% TO CONVERT TO A FUNCTION USING THE COLUMN VECTORS, DELETE
% THE LINE "D=A'" AND CHANGE ALL "D"s TO "A"s AND RENAME THE FILE
% (e.g., CBASIS.M). THIS MAY BE USEFUL FOR OBTAINING THE BASIS
% PICTURES OF AN INVERSE TRANSFORM FROM THE INVERSE MATRIX
% TO AVOID CONFUSION.
%
function X = basis(A)
[m,n]=size(A);
D=A';
a=1:m;
b=1:n;
for i=1:n;
        for j=1:n;
        X(a,b) = D(:,i) * D(:,j)';
        b=b+n;
        end
        a=a+n;b=1:n;
end
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1993	Technical Memorandum

4. TITLE AND SUBTITLE

Subband/Transform Functions for Image Processing

5. FUNDING NUMBERS

WU-144-10-10

6. AUTHOR(S)

Daniel Glover

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER

E-7888

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM-106183

11. SUPPLEMENTARY NOTES

Responsible person, Daniel Glover, (216) 433-2847.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 61

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

Functions for image data processing written for use with the MATLAB™ software package are presented. These functions provide the capability to transform image data with block transformations (such as the Walsh Hadamard) and to produce spatial frequency subbands of the transformed data. Block transforms are equivalent to simple subband systems. The transform coefficients are reordered using a simple permutation to give subbands. The low frequency subband is a low resolution version of the original image, while the higher frequency subbands contain edge information. The transform functions can be cascaded to provide further decomposition into more subbands. If the cascade is applied to all four of the first stage subbands (in the case of a four band decomposition), then a uniform structure of sixteen bands is obtained. If the cascade is applied only to the low frequency subband, an octave structure of seven bands results. Functions for the inverse transforms are also given. These functions can be used for image data compression systems. The transforms do not in themselves produce data compression, but prepare the data for quantization and compression. Sample quantization functions for subbands are also given. A typical compression approach is to subband the image data, quantize it, then use statistical coding (e.g., run-length coding followed by Huffman coding) for compression. Contour plots of image data and subbanded data are shown.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Data compression; Software; Image processing; Coding	36
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	