DELHI TECHNOLOGICAL UNIVERSITY

Computer Vision Assignment

EC - 357

Anish Sachdeva

DTU/2K16/MC/013

Haar Transform

```
im = imread(test-image.jpeg');
imagesc(im)
```


```
[a2,h2,v2,d2] = haart2(im,2);
imagesc(a2)
```

Slant Transform

getSlantTransform.m

```
function t=getSlantTransform(im,N)
s=sltmtx(log2(N));
t=s*im*s';
```

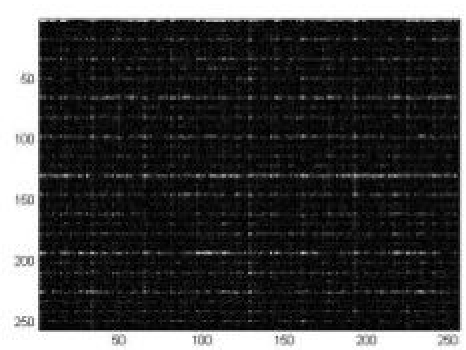
Sltmtx.m

```
function T = sltmtx(L)
% sltmtx   slantlet matrix.
%
% T = sltmtx(L) is the slantlet matrix of size 2^L by 2^L.
%
% See also slantlt, islantlt, sislet, isislet.
%
% % example
% l = 4;
% x = sin(sin([1:2^l]/3));
% T = sltmtx(l);
% q = T*x(:);
% s = slantlt(x);
% max(abs(q-s(:)))

% Ivan Selesnick, 1997
% subprograms: getg.m, gethf.m


m = 2^L;
T = zeros(m);

[a0,a1,b0,b1,c0,c1,d0,d1] = gethf(L);
h = [a0+a1*(0:m-1), b0+b1*(0:m-1)];
f = [c0+c1*(0:m-1), d0+d1*(0:m-1)];

T(1,1:m) = h(1:m) + h(m+1:2*m);
T(2,1:m) = f(1:m) + f(m+1:2*m);

for i = L-1:-1:1
 for k = 1:2^(L-i-1)
   m = 2^i;
   [a0,a1,b0,b1] = getg(i);
   g = [a0+a1*(0:m-1), b0+b1*(0:m-1)];
   gr = g(2*m:-1:1);
le = 2^(i+1);
q = 2^(L-i)+2*(k-1)+1;
   T(q,[1:le]+le*(k-1)) = g;
   T(q+1,[1:le]+le*(k-1)) = gr;
 end
```

```
end
```

slantImage.m

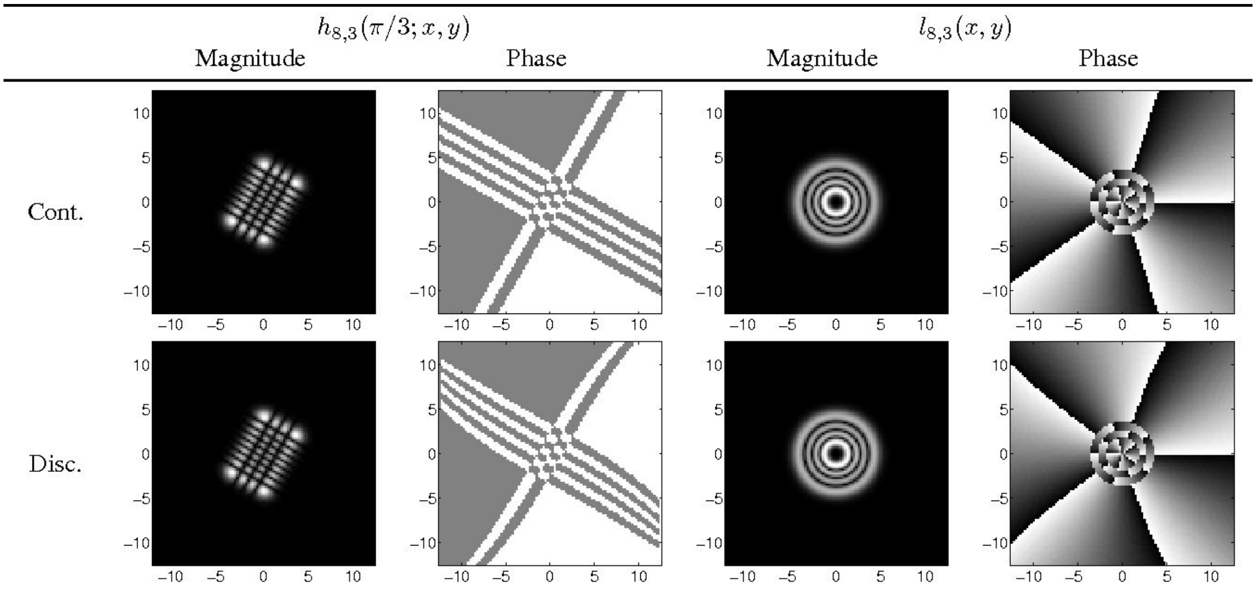
```
clc;
clear all;
close all;
A=imread('cameraman.tif');
figure,imshow(uint8(A))
title('Original Image');
A=double(A);
[s1 s2]=size(A);
% bs=input('Enter the block sizes for division of the image: '); % Block Size
bs=256;

% Slant
temp=double(zeros(size(A)));
for y=1:bs:s1-bs+1
  for x=1:bs:s2-bs+1
    croppedImage = A((y:y+bs-1),(x:x+bs-1));
    t=getSlantTransform(croppedImage,bs);
    temp((y:y+bs-1),(x:x+bs-1))=t;
  end
end
figure,imshow(uint8(temp))

% Inverse Slant
temp1=double(zeros(size(A)));
for y=1:bs:s1-bs+1
  for x=1:bs:s2-bs+1
    croppedImage = temp((y:y+bs-1),(x:x+bs-1));
    t=getInvSlantTransform(croppedImage,bs);
    temp1((y:y+bs-1),(x:x+bs-1))=t;
  end
end
figure,imshow(uint8(temp1))
```



Sine Transform

```
function c = dstmtx(n)
%DSTMTX Discrete sine transform matrix.
%   D = DSTMTX(N) returns the N-by-N DST transform matrix.  D*A is the DST
%   of the columns of A and D'*A is the inverse DST of the columns of A
%   (when A is N-by-N).
%
%   If A is square, the two-dimensional DST of A can be computed as D*A*D'.
%   This computation is sometimes faster than using DSTN, especially if you
%   are computing large number of small DST's, because D needs to be
%   determined only once.
%
%   Class Support
%   -------------
%   N is an integer scalar of class double. D is returned as a matrix of
%   class double.
%
%   Example
%   -------
%      A = im2double(imread('rice.png'));
%      D = dstmtx(size(A,1));
%      dst = D*A*D';
%      figure, imshow(dst)
%
%   See also DSTN, IDSTN, DCTMTX
%

%   I/O Spec
%      N - input must be double
%      D - output DCT transform matrix is double

iptchecknargin(1,1,nargin,mfilename);
iptcheckinput(n,{'double'},{'integer' 'scalar'},mfilename,'n',1);

[cc,rr] = meshgrid(0:n-1);
```

```
c = sqrt(2 / n) * sin(pi * (2*cc + 1) .* (rr + 1) / (2 * n));
c(n,:) = c(n,:) / sqrt(2);
```



Cosine Transform

Application: Removing High frequencies in images using DCT

This example shows how to remove high frequencies from an image using the two-dimensional discrete cosine transfer (DCT).

Read an image into the workspace, then convert the image to grayscale

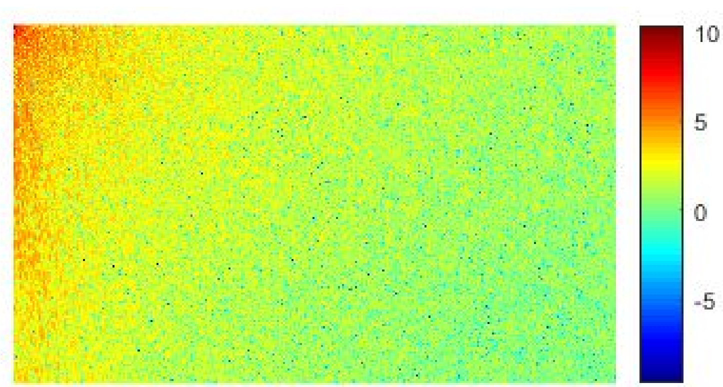
```
RGB = imread('autumn.tif');
I = rgb2gray(RGB);
```

Perform a 2-D DCT of the grayscale image using the dct2 function.

```
J = dct2(I);
```

Display the transformed image using a logarithmic scale. Notice that most of the energy is in the upper left corner.

```
figure
imshow(log(abs(J)),[])
colormap(gca,jet(64))
colorbar
```



Set values less than magnitude 10 in the DCT matrix to zero.

```
J(abs(J) < 10) = 0;
```

Reconstruct the image using the inverse DCT function `idct2`.

```
K = idct2(J);
```

Display the original grayscale image alongside the processed image.

```
figure
imshowpair(I,K,'montage')
title('Original Grayscale Image (Left) and Processed Image
(Right)');
```

.Discrete Cosine Transform

```
I = imread(test-image.jpeg');

I = im2double(I);

T = dctmtx(8);

dct = @(block_struct) T * block_struct.data * T';

B = blockproc(I,[8 8],dct);

mask = [1   1   1   1   0   0   0   0

        1   1   1   0   0   0   0   0

        1   1   0   0   0   0   0   0

        1   0   0   0   0   0   0   0

        0   0   0   0   0   0   0   0

        0   0   0   0   0   0   0   0

        0   0   0   0   0   0   0   0

        0   0   0   0   0   0   0   0];

B2 = blockproc(B,[8 8],@(block_struct) mask .* block_struct.data)

invdct = @(block_struct) T' * block_struct.data * T;
```

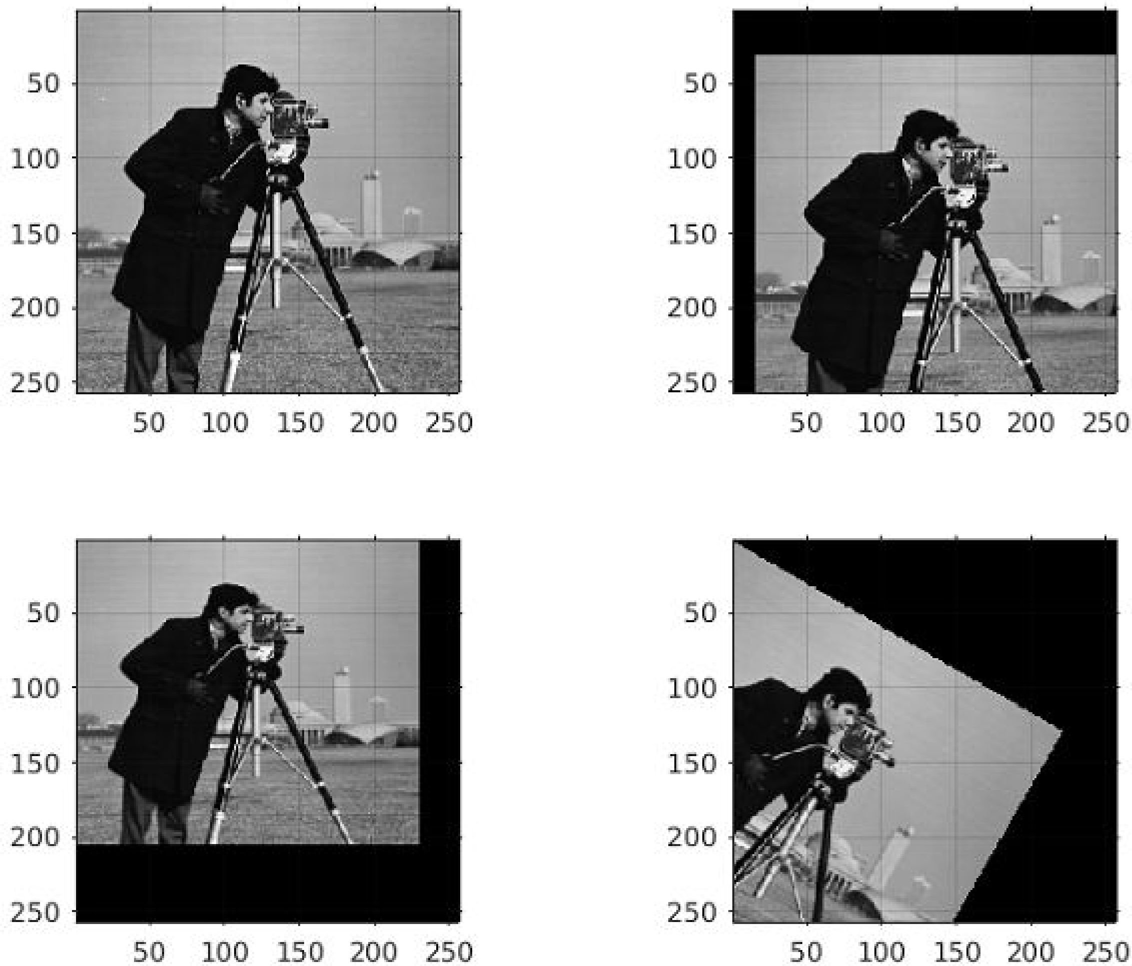
```
I2 = blockproc(B2,[8 8],invdct);
imshow(I)
figure
imshow(I2)
```

Applying Geometric Transform, Scaling, Rotation etc.

Code

```
I1 = imread(test-image.jpeg');
I1_ref = imref2d(size(I1));

T = [1 0 0; 0 1 0; 15 30 1];
[I2, I2_ref] = imwarp(I1, affine2d(T), 'OutputView', I1_ref);

T = [0.9 0 0; 0 0.8 0; 0 0 1];
[I3, I3_ref] = imwarp(I1, affine2d(T), 'OutputView', I1_ref);

x = pi/6;
T = [cos(x) sin(x) 0; -sin(x) cos(x) 0; 0 0 1];
[I4, I4_ref] = imwarp(I1, affine2d(T), 'OutputView', I1_ref);

figure;
subplot(2, 2, 1);
imshow(I1, I1_ref);

subplot(2, 2, 2);
imshow(I2, I2_ref);

subplot(2, 2, 3);
imshow(I3, I3_ref);
subplot(2, 2, 4);
imshow(I4, I4_ref);
```

Output



50
100
150
200
250
50 100 150 200 250
50
100
150
200
250
50 100 150 200 250
50
100
150
200
250
50 100 150 200 250
50
100
150
200
250
50 100 150 200 250

Wiener Filter

Code

```
I = imread('cameraman.tif');

PSF = fspecial('motion', 21, 11);
Idouble = im2double(I);
I1 = imfilter(Idouble, PSF, 'conv', 'circular');
subplot(1, 2, 1);
imshow(I1);
title('Blurred Image');

I2 = deconvwnr(I1, PSF);
subplot(1, 2, 2);
imshow(I2);
title('Restored Blurred Image');
```

Output