LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.06**

**A.1 Aim:**

Write a program to apply various (DCT, DST, Hartley and Slant) transforms on an image and compare the results.

**A.2 Prerequisite:**

1 Matlab programming syntax (Refer the Matlab manual).

2. Knowledge of DCT, DST, Hartley and slant Transform.

2. Availability of Soft copy of your 3 Photographs with different background (i.e. Plane, scenery,

etc.) for experiment.

**A.3 Outcome:**

**After successful completion of this experiment students will be able to**

1. Understand the fundamentals of DCT, DST, Hartley and Slant Transforms and its effects on digital images.
2. Appreciate different properties of the DCT, DST, Hartley and Slant transform.
3. Apply and verify the correctness of DCT, DST, Hartley and Slant transforms on images.
4. Identify applications of transforms studied.

**A.4 Theory:**

**A.4.1. Discrete Cosine Transform (DCT)**

The DCT is similar to the discrete Fourier transform. It transforms a signal or image from the spatial domain to the frequency domain.

The NxN cosine transform matrix C={c(k,n)},also called as discrete cosine transform (DCT), is defined as

Properties of the cosine transform

1. The cosine transform is real and orthogonal, that is

C=C\* where C\* is Complex Conjugate of C.

1. C-1 = CT orthogonality property.
2. The cosine transform is a fast transform as it is real.
3. The cosine transform has excellent energy compaction for highly correlated data.

**A.4.2. Discrete Sine Transform(DST)**

The NxN sine transform matrix also called the discrete sine transform (DST), is defined as

Properties of the sine transform

1. The sine transform is real, symmetric and orthogonal, that is

Thus, the forward and inverse sine transforms are identical.

1. The sign transform is not the imaginary part of the unitary DFT.

The sine transform is the fast transform

**A.4.3. Hartley Transform**

The Normalized Hartley Transform matrix of size can be obtained by the given equation (3.14) below.

Properties of Hartley Transform:

1. It is real. H=H\*
2. It is orthogonal. HHt = I, Ht = H-1 .
3. It is symmetric. H=Ht ; H=H-1

**A.4.3. Slant Transform**

The slant transform has its first basis function as constant and second basis function as linear. The Slant vector is a discrete saw tooth waveform decreasing in uniform steps over its length. It has been seen that Slant vectors are suitable for efficiently representing gradual brightness change in an image line.

**I. Slant Matrix Construction:**

If S(n) denotes the NxN Slant matrix(N=2n  ),then

 (4)

The Slant matrix for N=4 can be written as

 (5)

where a and b are real constants to be determined subject to the following conditions:

1. step size must be uniform
2. S(2) must be orthogonal

The value of a and b are as follows

 and 

So normalized matrix S(2) becomes

Kamal1 (6)

It is observed that S(2) posses the sequency property like Walsh Hadamard transform**.**

Now S(2) can be represented in terms of S(1) as

 (7)

**where**   and 

Equation (4) can be generalized to give the Slant matrix of order n in terms of Slant matrix of order N/2 as follows

****

(8)

The coefficients (aN  and bN) can be calculated using the following relation





Where N =4,8,16… etc.

The properties of Slant transform are as follows:

1. The Slant transform is real and orthogonal



1. It has fast algorithm which is of order



1. It is very good in energy compaction for television pictures.

**A.5 Procedure/Algorithm:**

**A.5.1:**

**TASK 1:**

1. Read the i/p image

2. Resize the image to convert it into square matrix

3. Generate all mentioned transform matrices of size equivalent to the size

of image.

4. Apply these Transforms on the all images separately.

5. Display the transformed images w.r.t. particular transform applied.

6. Regenerate and display the original image back

7. Compare the input and output images for each transforms applied w.r.t. its

matrix content and visibility on the screen.

8. Add the original image with the transformed output image of each

Transformation function applied separately and observe the result.

8. Save and close the file and name it as **EX5\_Task1\_your Roll no.m**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.: N049 | Name: Tarun Tanmay |
| Class: MBATech CE | Batch: B3 |
| Date of Experiment: 27/08/2020 | Date of Submission |
| Grade: | Time of Submission: |
| Date of Grading: |  |

**B.1 Software Code written by student:**

***DCT, DST, and Hartley:***

clear all;

clc;

a=imread("/Users/tjrox0825/Desktop/InShot\_20190509\_015525989.jpg");

ag=rgb2gray(a);

ar=imresize(ag,[128,128]);

ad=double(ar);

N=128;

[row,col]=size(ar);

%DCT

c4=[];

for k=0:3

for n=0:3

if k==0

c4(k+1,n+1)=1/(4^0.5);

else

c4(k+1,n+1)=((2/4)^0.5)\*cos((pi\*((2\*n)+1)\*k)/(2\*4));

end

end

end

c128=[];

for k=0:127

for n=0:127

if k==0

c128(k+1,n+1)=1/(N^0.5);

else

c128(k+1,n+1)=((2/N)^0.5)\*cos((pi\*((2\*n)+1)\*k)/(2\*N));

end

end

end

% disp(c128);

dctimg=c128\*ad\*c128;

invdct=(c128\*dctimg\*c128');

figure('name','Figures');

subplot(3,3,1);

imshow(ar);

title('Original image');

dct=uint8(dctimg);

subplot(3,3,2);

imshow(dct);

title('DCT image');

idct=uint8(invdct);

subplot(3,3,3);

imshow(idct);

title('Inverse image');

%DST

dst128=[];

for k=0:127

for n=0:127

dst128(k+1,n+1)=((2/(N+1))^0.5)\*sin((pi\*(n+1)\*(k+1))/(N+1));

end

end

% disp(dst128);

dstimg=dst128\*ad\*dst128;

invdst=(dst128\*dstimg\*dst128');

subplot(3,3,4);

imshow(ar);

title('Original image');

dst=uint8(dstimg);

subplot(3,3,5);

imshow(dst);

title('DST image');

idst=uint8(invdst);

subplot(3,3,6);

imshow(idst);

title('Inverse image');

%Hartley

h128=[];

for k=1:128

for n=1:128

h128(k,n)=(1/(N^0.5))\*(cos((2\*(k-1)\*(n-1)\*pi)/N)+sin((2\*(k-1)\*(n-1)\*pi)/N));

end

end

% disp(dst128);

himg=h128\*ad\*h128;

invh=(h128\*himg\*h128');

subplot(3,3,7);

imshow(ar);

title('Original image');

h=uint8(himg);

subplot(3,3,8);

imshow(h);

title('Hartley image');

ih=uint8(invh);

subplot(3,3,9);

imshow(ih);

title('Inverse image');

***SLANT:***

clear all;

clc;

a=imread("/Users/tjrox0825/Desktop/TJ.jpg");

ag=rgb2gray(a);

ar=imresize(ag,[128,128]);

ad=double(ar);

N=128;

[row,col]=size(ad);

figure('name','Figures');

subplot(1,3,1);

imshow(ar);

title('Original image');

temp=double(zeros(size(ad)));

for y=1:N:row-N+1

for x=1:N:col-N+1

croppedImage = ad((y:y+N-1),(x:x+N-1));

t=getSlantTransform(croppedImage,N);

temp((y:y+N-1),(x:x+N-1))=t;

end

end

st=uint8(temp);

subplot(1,3,2);

imshow(st);

title('Slant image');

% Inverse Slant

temp1=double(zeros(size(ad)));

for y=1:N:row-N+1

for x=1:N:col-N+1

croppedImage = temp((y:y+N-1),(x:x+N-1));

t=getInvSlantTransform(croppedImage,N);

temp1((y:y+N-1),(x:x+N-1))=t;

end

end

invst=uint8(temp1);

subplot(1,3,3);

imshow(invst);

title('Inverse Slant image');

function t=getSlantTransform(im,N)

s=sltmtx(log2(N));

t=s\*im\*s';

end

function t=getInvSlantTransform(im,N)

s=sltmtx(log2(N));

t=s'\*im\*s;

end

function T = sltmtx(L)

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

end

function [a0,a1,b0,b1,c0,c1,d0,d1] = gethf(l)

m = 2^l;

u = 1/sqrt(m);

v = sqrt((2\* m^2+1)/3);

a0 = u\*(v+1)/(2\*m) ;

b0 = u\*(2\*m-v-1)/(2\*m) ;

a1 = u/m ;

b1 = -a1 ;

r = -sqrt((3\*m-sqrt(6\*m^2+3))/(3\*m+sqrt(6\*m^2+3)));

c0 = r\*a0;

c1 = r\*a1;

d0 = -b0/r;

d1 = -b1/r;

end

function [a0,a1,b0,b1,a0r,a1r,b0r,b1r] = getg(i)

m = 2^i;

s1 = 6\*sqrt( m / ( (m^2-1)\*(4\*m^2-1)) );

t1 = 2\*sqrt( 3/ (m\*(m^2-1)));

s0 = -s1 \*(m-1) / 2;

t0 = ( (m+1)\* s1/3 - m\* t1 ) \*(m-1) /(2\*m);

a0 = (s0 + t0)/2;

b0 = (s0 - t0)/2;

a1 = (s1 + t1)/2;

b1 = (s1 - t1)/2;

% time reversed version

a0r = b0+b1\*(m-1);

a1r = -b1;

b0r = a0+a1\*(m-1);

b1r = -a1;

end

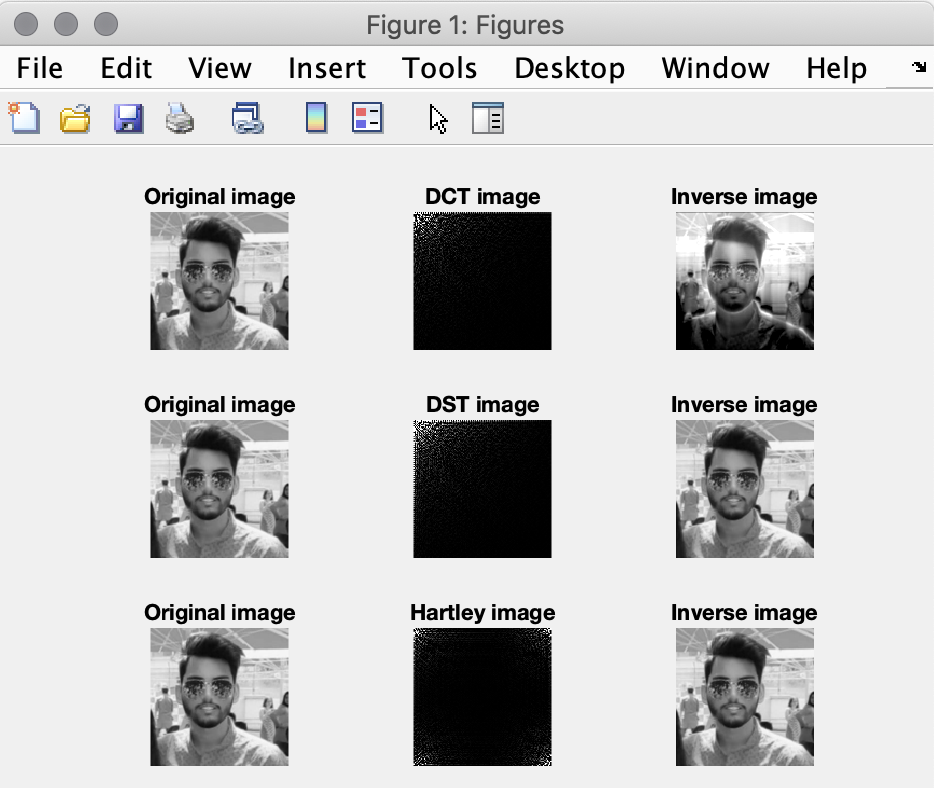
**B.2 Input and Output:**

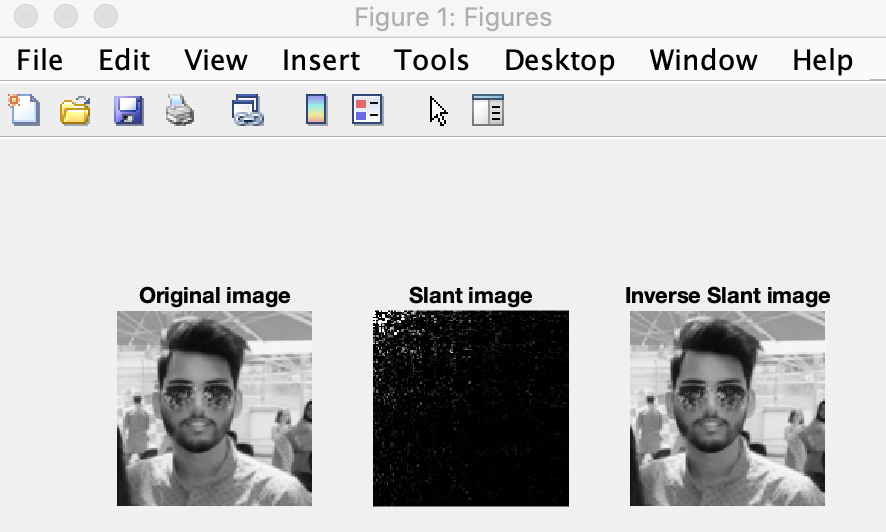
**Input Images:**

****

**Output Images:**

1. **For each Transform as per the procedure discussed in section A.5**



****

**B.3 Observations and learning:**

When working with images, sometimes operations in spatial domain do not have the desired output so the image is converted to transform domain, operations related to the domain are performed and converted back to spatial domain as the output of transform domain cannot be understood easily. Some operations in transform domain are Discrete Sine Transform, Discrete Cosine Transform, Hartley Transform and Slant Transform. All these transformed images will have to be converted back to spatial domain using inverse operation.

**B.4 Conclusion:**

I have understood the concept of Discrete Sine Transform, Discrete Cosine Transform, Hartley Transform and Slant Transform as well as their applications. I have also implemented them and understood their effect on images.

**B.5 Question of Curiosity**

***(To be answered by student based on the practical performed and learning/observations)***

Will be shared during the practical session.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*