LAB - 1

Name: Siddharth Bose

Reg. Number: 19BEC1278

Date: 06/01/2022

Course Code: ECE3005

Slot: L23+L24

Faculty: Dr . Ranjeet Kumar

AIM: To read an image and display the red, green and blue values of the image individually.

TOOLS: MATLAB, Image Processing Toolbox

PROCEDURE:

1. Read an image in MATLAB using the command **imread**. This will upload the image along with its size to the Workspace.

2. To plot the image, use the command **imshow.** This will display the image. It functions like the **plot** command but for images.

3. To find the Red values of the image, we refer to the first column of the matrix of the image. We use the command **I(:,:,1)** where I is the variable assigned to the original image.

4. To find the Green values of the image, we refer to the second column of the matrix of the image. We use the command **I(:,:,2)** where I is the variable assigned to the original image.

5. To find the Blue values of the image, we refer to the third column of the matrix of the image. We use the command **I(:,:,3)** where I is the variable assigned to the original image.

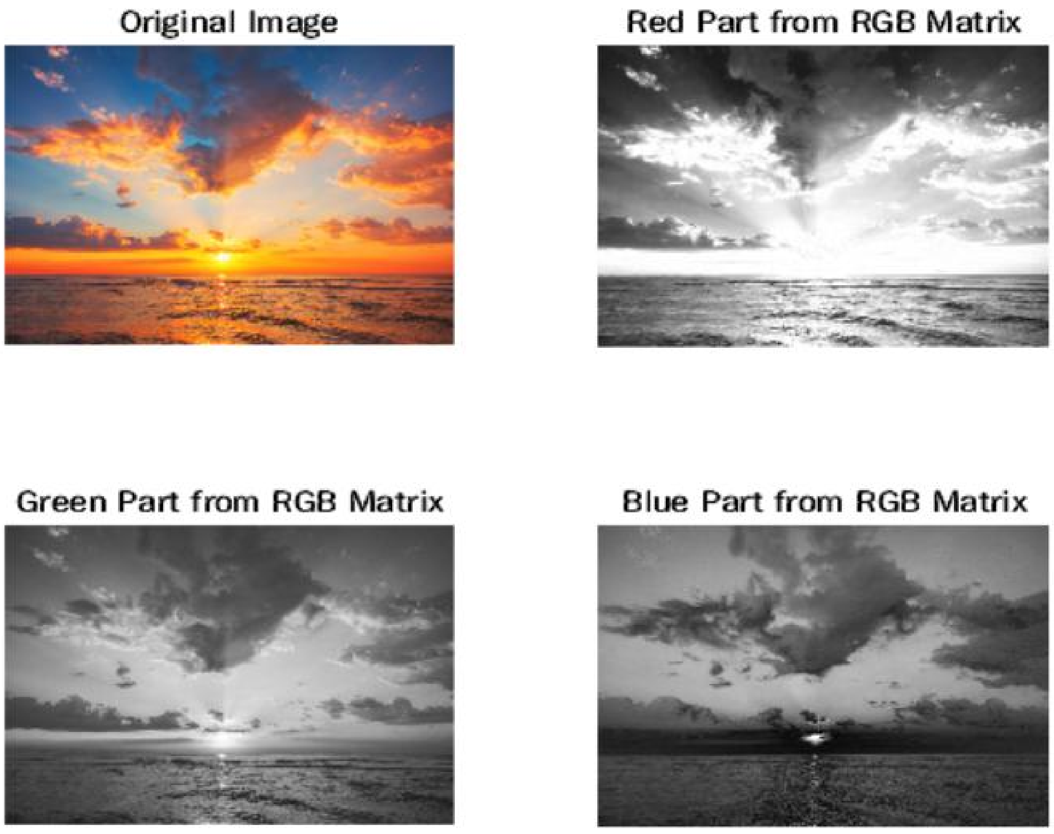
ALGORITHM/CODE:

```
I=imread("dipimage.jpg");

figure(1)

subplot(2,2,1)

imshow(I);

title('Original Image');

I_R=I(:,:,1);
```

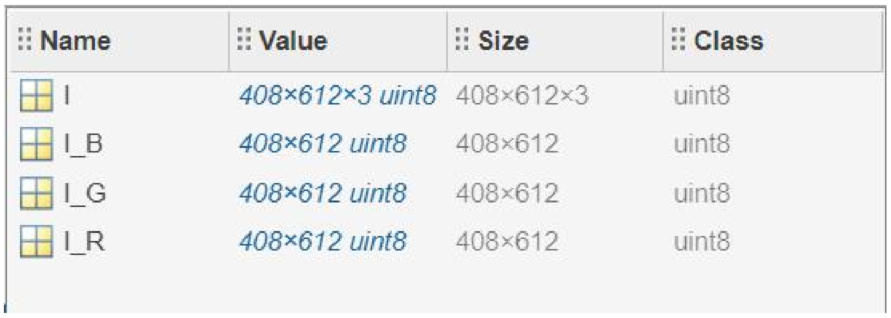
```
subplot(2,2,2)

imshow(I_R);

title('Red Part from RGB Matrix');

I_G=I(:,:,2);

subplot(2,2,3)

imshow(I_G);

title('Green Part from RGB Matrix');

I_B=I(:,:,3);

subplot(2,2,4)

imshow(I_B);

title('Blue Part from RGB Matrix');
```

OUTPUT:

A. Output Images



B. Workspace Window



Name	Value	Size	Class
I	408×612×3 uint8	408×612×3	uint8
I_B	408×612 uint8	408×612	uint8
I_G	408×612 uint8	408×612	uint8
I_R	408×612 uint8	408×612	uint8

INFERENCES:

1. The original image is of size 403×612 pixels. This image is stored in a matrix with 3 columns such that the Red, Green and Blue values of the image are each assigned a column.

2. The size of Red, Blue and Green matrices of the original image will have the same size.

RESULT: Simple commands to display an image and its Red, Green and Blue components were performed using MATLAB.

Experiment -2

Name: Siddharth Bose **Reg. Number:**19BEC1278 **Date:** 13/01/2022

Course Code: ECE3005 **Slot:** L23+L24 **Faculty:** Dr. Ranjeet Kumar

AIM: To perform the following transformations on an image and observe their effects:

A. Image Negation

B. Power Law Transformation

C. Log Transformation

TOOLS: MATLAB, Image Processing Toolbox

THEORY:

A. **Image Negation**

For an image with **L intensity levels**, the negative transformation is given by

$$s = L - 1 - r$$

This enhances white or gray details embedded in dark regions of an image.

B. Power Law Transformation

The general form of the power-law transformation is

$$s = cr^{\gamma}$$

where c and γ are positive constants. This transformation is also known as gamma correction. A variety of devices used for image capturing, printing and display respond according to power law. The optimal value for γ is device-dependent.

C. Log Transformation

The general form of the log transformation is

$$s = c \log(1 + r)$$

where c is a constant, and it is assumed that $r \geq 0$. This compresses the dynamic range of images with large variations in pixel values

PROCEDURE:

1. Insert an input image using the command **imread**. Convert the values of the image to double using **im2double.**

2. For Image Negation, find the R, G and B matrices of the image using the command **I_R=I(:,:,1), I_G=I(:,:,2) and I_B=I(;,:,3)**.

3. Perform intensity inversion for the R, G and B matrices by subtracting the values from 1 for i=m and j=n where m is the number of rows and n is the number of the columns of the image.

4. Assign the negative R, G and B matrices to another image and display it using **imshow**. This is the negative image of the original image.

5. Now, convert the original image to its GrayScale version using the command **rbg2gray**.

6. Initialise an array g for gamma values and assign it any 9 values. Assign c1=0.5 and c2=2.

7. Perform the Power Law Transformation using the formula. Display the images using **subplot** and **imshow.**

8. Perform Log Transformation for the same values of c1 and c2 using the formula given. Display the images using **imshow**.

ALGORITHM/CODE:

A. Image Negation

```
clc

clear all

format compact

I=imread("Image.jpg");

I=im2double(I);


%Find the dimensions of the image

%m = Number of Rows

%n = Number of Columns

%k = Number of Color Channels

[m,n,k] = size(I);
```

```
I_R = I(:,:,1); %Find red channel of the input image

I_G = I(:,:,2); %Find green channel of the input image

I_B = I(:,:,3); %Find blue channel of the input image

I_Rn = zeros(m,n);

I_Gn = zeros(m,n);

I_Bn = zeros(m,n);


%Intensity Inversion for RGB Matrices

for i=1:m

for j=1:n

I_Rn(i,j)=1-I_R(i,j);

I_Gn(i,j)=1-I_G(i,j);

I_Bn(i,j)=1-I_B(i,j);

end

end


%Image Negation

I_n=zeros(m,n,k);

I_n(:,:,1)=I_Rn;

I_n(:,:,2)=I_Gn;

I_n(:,:,3)=I_Bn;


figure(1)

subplot(1,2,1)

imshow(I);

title("Input Image");

subplot(1,2,2)

imshow(I_n);

title("Negative Image");
```

B. Power Law Transformation

```
%Power Law Transformation

G_I = rgb2gray(I); %Getting grayscale image from input image

G_I = im2double(G_I);

[m,n,k]=size(G_I);

c1=0.5; %Case 1: c=0.5

c2=2; %Case 2: c=2

g=[0.5 0.7 0.9 1 2 3 4 5 6];

for k=1:length(g)

for i=1:m

for j=1:n

P_I1(i,j)=c1*G_I(i,j).^g(k); %Performing Power Law Transformation

end

end

figure(2)

sgtitle('Power Law Transformation for c=0.5');

subplot(3,3,k)

imshow(P_I1);

xlabel(['Gamma: ',num2str(g(k))]);

end


for k=1:length(g)

for i=1:m

for j=1:n

P_I2(i,j)=c2*G_I(i,j).^g(k); %Performing Power Law Transformation

end

end

figure(3)

sgtitle('Power Law Transformation for c=2');

subplot(3,3,k)
```

```
imshow(P_I2);
xlabel(['Gamma: ',num2str(g(k))]);
end
```

C. Log Transformation

```
%Log Transformation
c1=0.5;
c2=2;
S1=c1*log(1+(G_I)); %Perform Log Transformation for c=0.5
S2=c2*log(1+(G_I)); %Perform Log Transformation for c=2
figure(4)
subplot(3,1,1)
imshow(I);
title('Original Image');
subplot(3,1,2)
imshow(G_I);
title('GrayScale Image');
subplot(3,1,3)
imshow(S1);
title('Log Transformed Image for c=0.5');
figure(5)
subplot(3,1,1)
imshow(I);
title('Original Image');
subplot(3,1,2)
imshow(G_I);
title('GrayScale Image');
subplot(3,1,3)
imshow(S2);
title('Log Transformed Image for c=2');
```

OUTPUT:

A. Image Negation

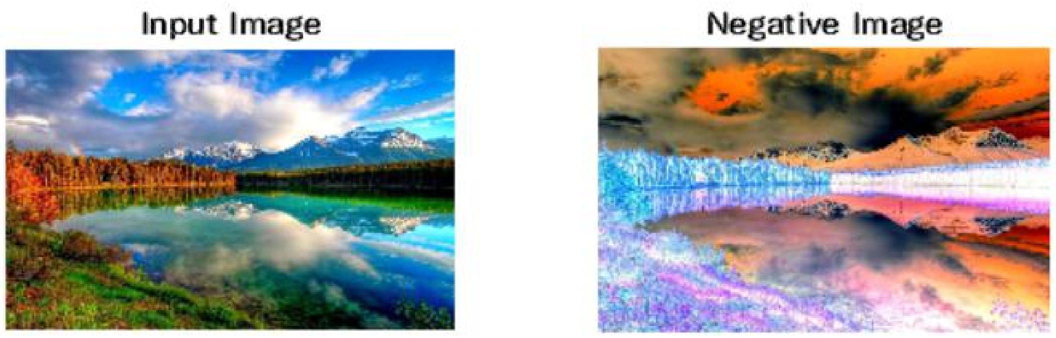


Fig 1 Image Negation of Input Image

B. Power Law Transformation

i. When c=0.5

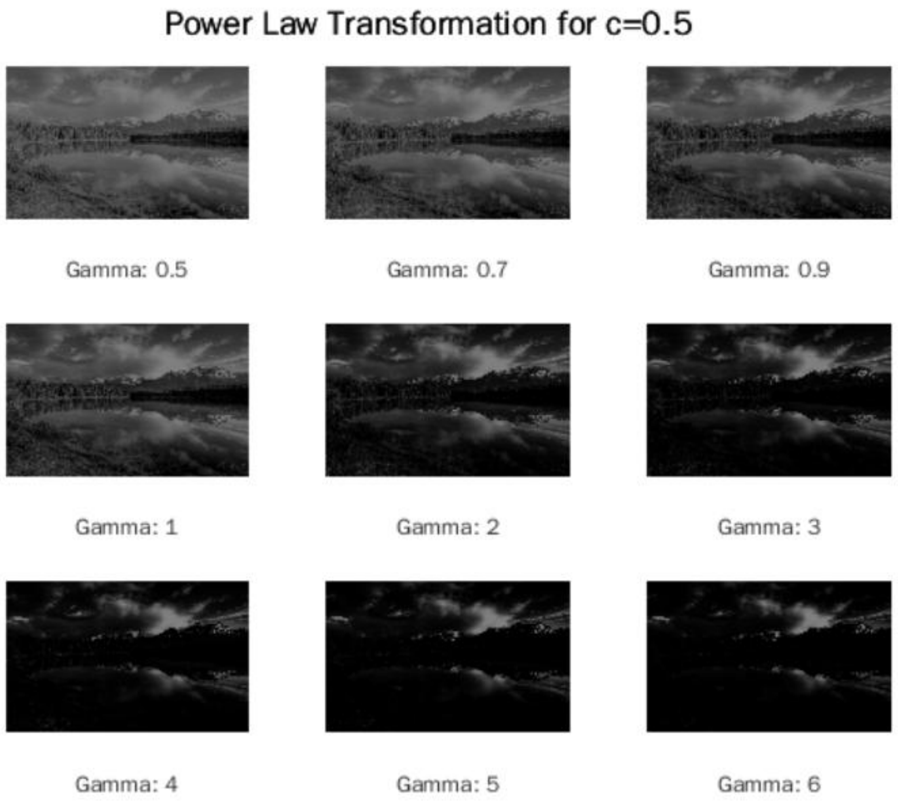


Fig 2 Power Law Tranformation when c=0.5 for Input Image

ii. When c=2

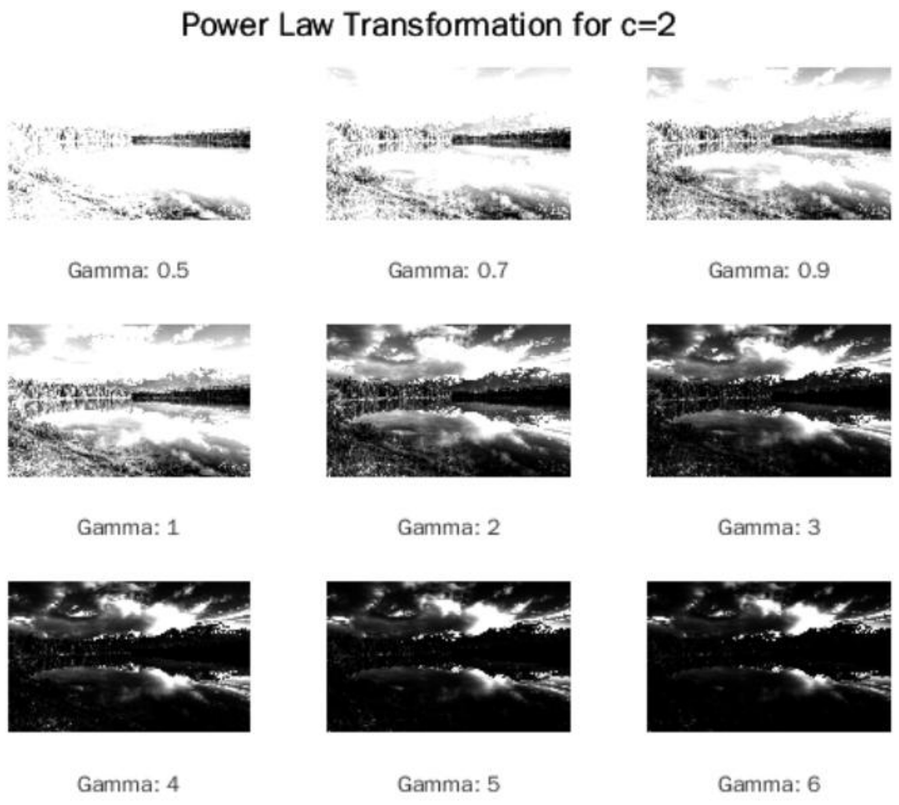


Fig 3 Power Law Transformation when c=2

C. Log Transformation

i. When c=0.5

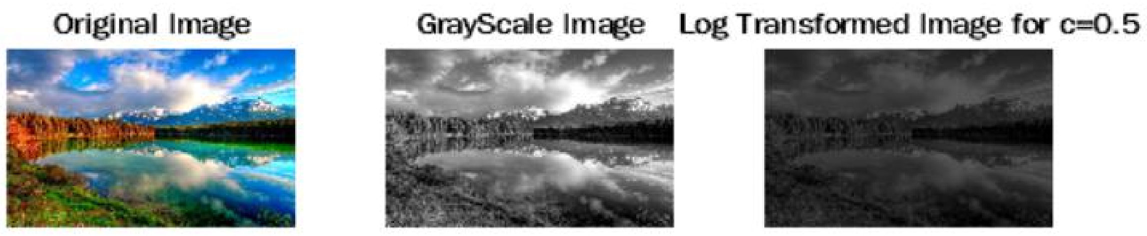


Fig 4 Log Transformation when c=0.5

ii. When c=2

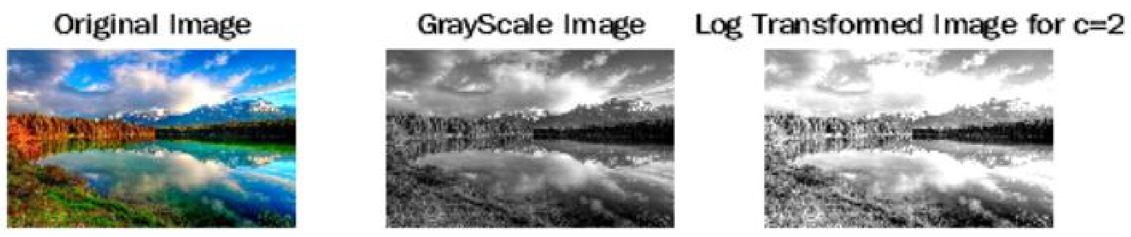


Fig 5 Log Transformation when c=2

INFERENCES:

1. In the negative of an image, the lightest parts appear darkest and the darkest parts appear lightest. This is because the image intensity at that pixel, is subtracted from the highest intensity possible for the image.

2. By varying values of γ, when it is lesser than 0, a narrow range of input values is mapped to a wider range of output values. The opposite occurws when it is greater than 0. Hence, by changing the value of γ and c, we can find a good power law transformed image for an input image.

3. Log transformation of an image means **replacing all pixel values**, present in the image, with its logarithmic values. Log transformation is used for image enhancement as it expands dark pixels of the image as compared to higher pixel values.

RESULT: Image Negation, Power Law Transformation and Log Transformation were performed on an input image using MATLAB and their effects were observed.

LAB - 3

Name: Siddharth Bose

Reg. Number:19BEC1278

Date: 24/02/2022

Course Code: ECE3005

Slot: L23+L24

Faculty: Dr. Ranjeet Kumar

AIM: To perform histogram equalisation on images using MATLAB

TOOLS: MATLAB, Image Processing Toolbox

THEORY:

Histogram Equalization is **a computer image processing technique used to improve contrast in images**. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image.

PROCEDURE:

1. Insert an input image using the command **imread**. Convert the image to its GrayScale using **rgb2gray.**
2. Find the histogram equalized images using **histeq, imadjust, adapthisteq**.
3. View the images using **imshow** and identify the differences.

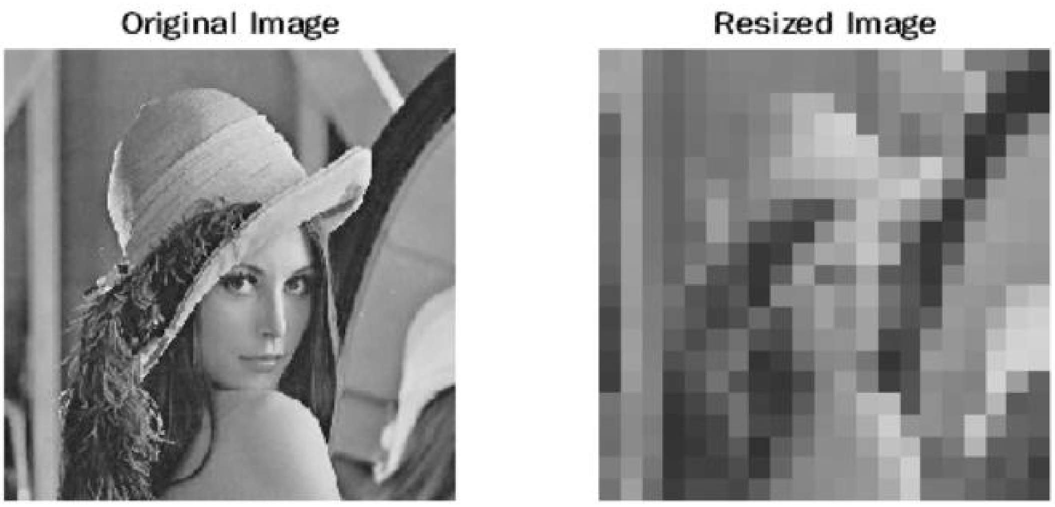
ALGORITHM/CODE:

```
clc

clear all

format compact

I=imread("dipimage1.jpg");

J=imresize(I, 0.08, "box");

I_H=histeq(I);

I_A=imadjust(I);

I_AH=adapthisteq(I);

figure(1)

subplot(2,1,1)

imshow(I);

title('Original Image');

subplot(2,1,2)

imshow(J);

title('Resized Image');

figure(2)

subplot(2,2,1)

imshow(I);

title('Original Image');

subplot(2,2,2)

imshow(I_H);

title('Enhanced Image');

subplot(2,2,3)

imhist(I);

title('Histogram of Original Image');

subplot(2,2,4)
```

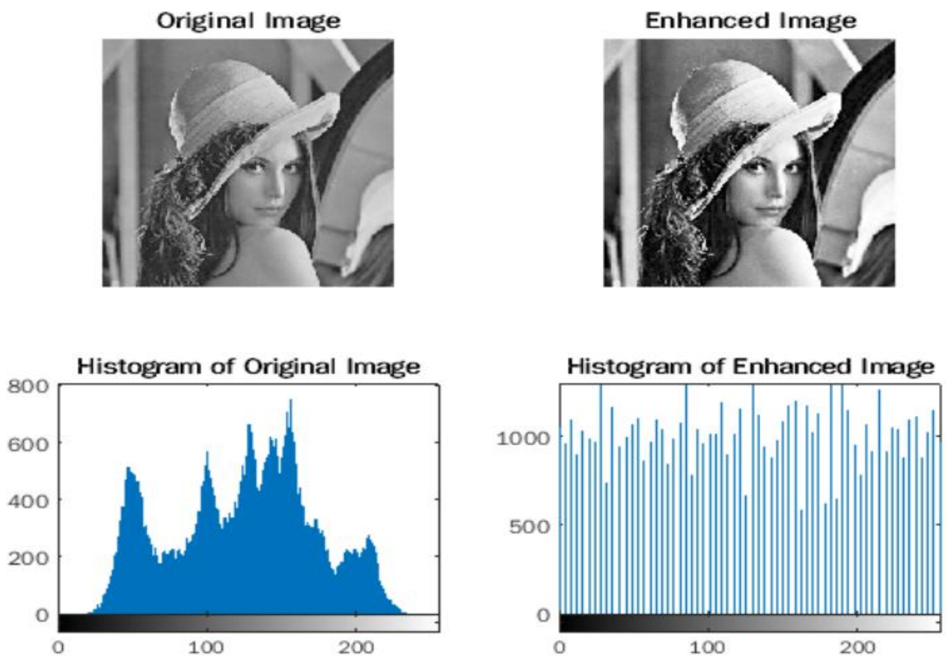
```
imhist(I_H);

title('Histogram of Enhanced Image');

figure(3)

montage({I,I_A,I_H,I_AH},"Size",[1 4])

title("Original Image and Enhanced Images using imadjust, histeq,
and adapthisteq")
```

OUTPUT:

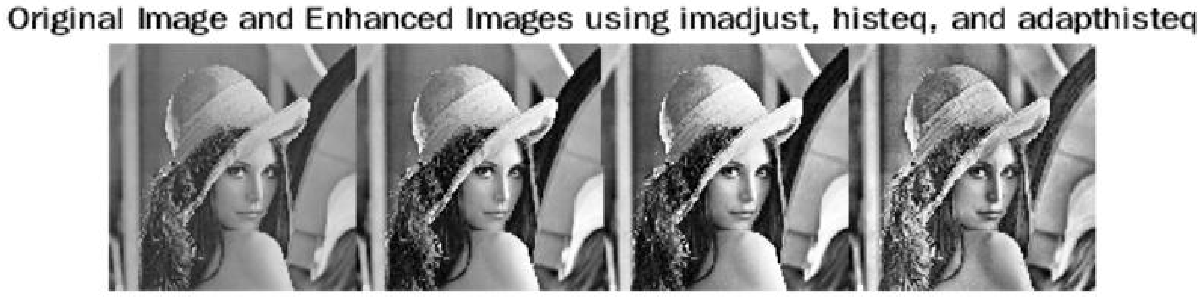
A. Resized Image



B. Original and Histogram Equalized Images



C. Histogram Equalized Images


Original Image and Enhanced Images using imadjust, histeq, and adapthisteq

INFERENCES:

1. The frequency intensity values are spread in the histogram equalised images.

RESULT:

Histogram Equalisation was done using MATLAB.

Experiment - 4

Name: Siddharth Bose **Reg. Number:**19BEC1278 **Date:** 03/03/2022

Course Code: ECE3005 **Slot:** L23+L24 **Faculty:** Dr. Ranjeet Kumar

AIM: To do noise addition to an image and apply the following filters:

A. Low Pass Filter for Image Smoothing

B. Median Pass Filter

C. High Pass Filter for Image Sharpening

TOOLS: MATLAB, Image Processing Toolbox

THEORY:

Image smoothing is a rapid process to soften edges and corners of the image. However, the image suffers from random noise. On the other hand, image sharpening refers to sharpen edges and correct the image even it has little defects. These operations will come under image enhancement.

PROCEDURE:

1. Insert an input image using the command **imread**. Convert the image to its GrayScale using **rgb2gray.**

2. Add salt and pepper noise to the image using the command **imnoise**.

3. Apply low pass filter, median filter and high pass filter. View the images using **imshow** and identify the differences.

ALGORITHM/CODE:

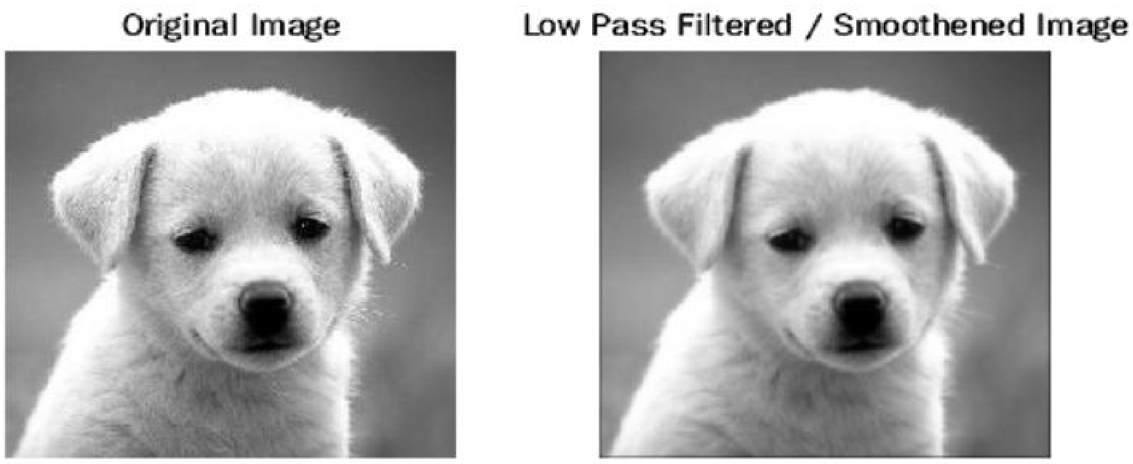
```
clc

clear all

close all

format compact
```

```
X=imread('gray.jpg');

X=rgb2gray(X);

J=imnoise(X,'salt & pepper',0.02);

k1=ones(3,3)/9;

k2=[0 -1/4 0;-1/4 2 -1/4;0 -1/4 0];

f1=imfilter(X,k1);

f2=imfilter(X,k2);

f3=medfilt2(J);

figure(1)

subplot(2,1,1)

imshow(X);

title('Original Image');

subplot(2,1,2)

imshow(f1);

title('Low Pass Filtered / Smoothened Image');

figure(2)

subplot(2,1,1)

imshow(X);

title('Original Image');

subplot(2,1,2)

imshow(f2);

title('High Pass Filtered / Sharpened Image');

figure(3)

subplot(2,1,1)

imshow(J);

title('Noisy Image');

subplot(2,1,2)
```

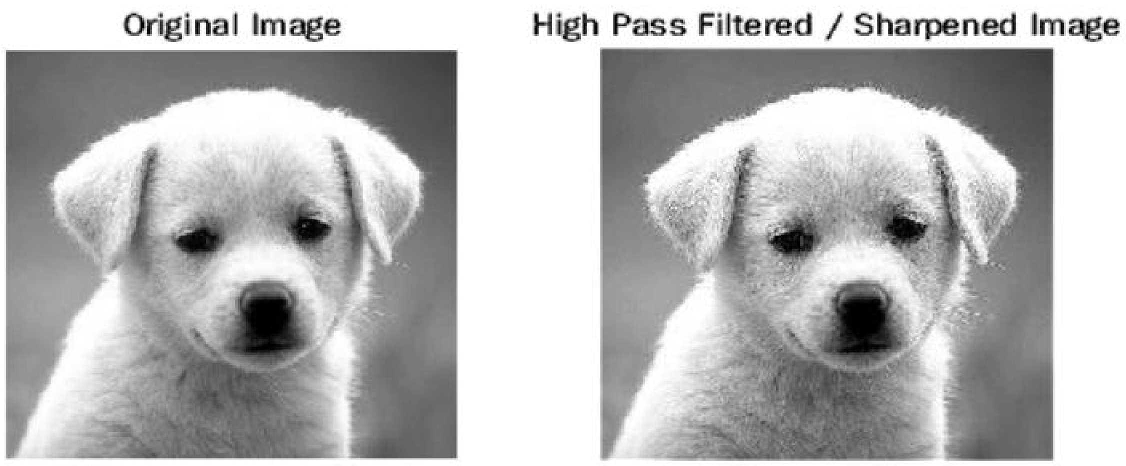
```
imshow(f3);

title('Median Filtered Image after Noise Addition');
```

OUTPUT:

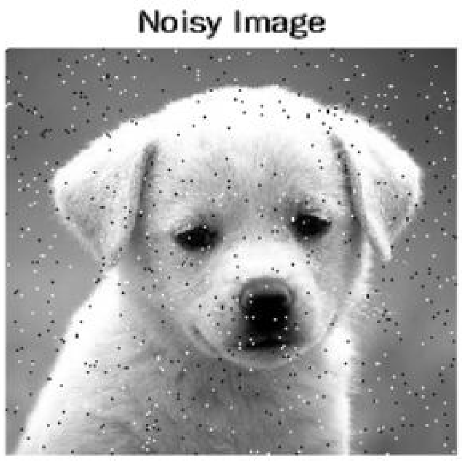
A. Low Pass Filtered / Smoothened Image

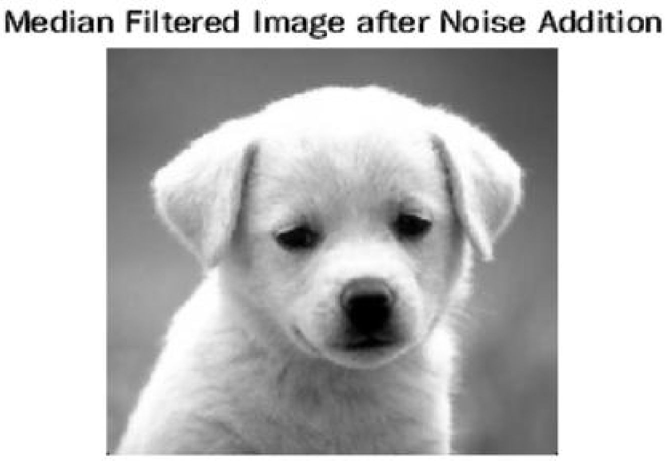


B. High Pass Filtered / Sharpened Image



C. Median Filtered Image for Noisy Image





INFERENCES:

1. The smoothened image has less noise. It is done to reduce noise in an image.

2. Sharpening enhances the definition of edges in an image. The dull images are those which are poor at the edges. There is not much difference in background and edges.

RESULT:

Image sharpening and smoothing was done using MATLAB.

Experiment - 5

Name: Siddharth Bose **Reg. Number:**19BEC1278 **Date:** 10/03/2022

Course Code: ECE3005 **Slot:** L23+L24 **Faculty:** Dr. Ranjeet Kumar

AIM: To perform arithmetic and logical operations on images using MATLAB

TOOLS: MATLAB, Image Processing Toolbox

THEORY:

Image arithmetic is the implementation of standard arithmetic operations, such as **addition, subtraction, multiplication, and division**, on images. Image arithmetic has many uses in image processing both as a preliminary step in more complex operations and by itself. Logical operators are **often used to combine two (mostly binary) images**. In the case of integer images, the logical operator is normally applied in a bitwise way.

PROCEDURE:

1. Insert input images using the command **imread**.
2. Perform the arithmetic and logical operations using MATLAB.
3. View the images using **imshow** and identify the differences.

ALGORITHM/CODE:

```
clc

clear all

close all

format compact

a=imread('dimage1.png');

b=imread('dimage2.png');

a=imresize(a,[512,512]);
```

```matlab
b=imresize(b,[512,512]);

%To perform arithmetic operations
%Addition, Subtraction, Multiplication, Division
I1=a+b;
figure(1)
subplot(3,1,1)
imshow(a)
title('Input Image 1');
subplot(3,1,2)
imshow(b)
title('Input Image 2');
subplot(3,1,3)
imshow(I1);
title('Image on Addition of Input Images');

I2=a-b;
figure(2)
subplot(3,1,1)
imshow(a)
title('Input Image 1');
subplot(3,1,2)
imshow(b)
title('Input Image 2');
subplot(3,1,3)
imshow(I2);
title('Image on Subtraction of Input Images');
```

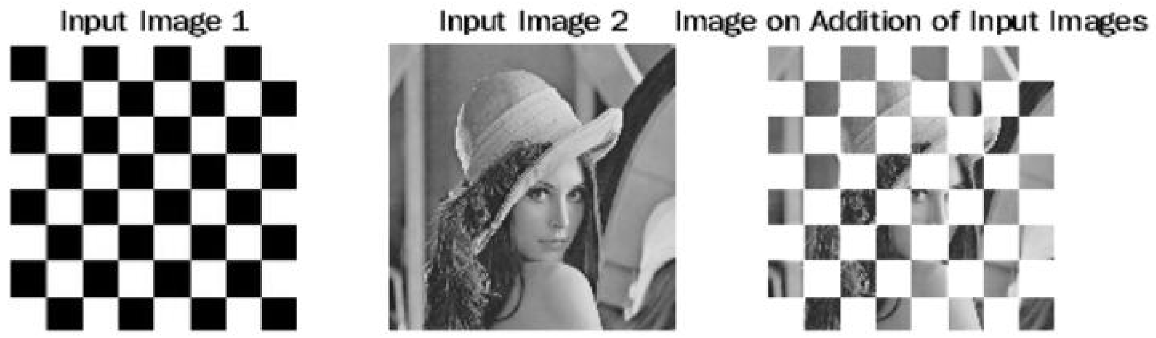
```
I3=a.*b;
figure(3)
subplot(3,1,1)
imshow(a)
title('Input Image 1');
subplot(3,1,2)
imshow(b)
title('Input Image 2');
subplot(3,1,3)
imshow(I3);
title('Image on Multiplication of Input Images');

I4=b./a;
figure(4)
subplot(3,1,1)
imshow(a)
title('Input Image 1');
subplot(3,1,2)
imshow(b)
title('Input Image 2');
subplot(3,1,3)
imshow(I4);
title('Image on Division of Input Images');

figure(5)
subplot(2,2,1)
```

```
imshow(I1)

title('Addition');

subplot(2,2,2)

imshow(I2)

title('Subtraction');

subplot(2,2,3)

imshow(I3)

title('Multiplication');

subplot(2,2,4)

imshow(I4)

title('Division');


I5=~a;

I6=~b;

figure(6)

subplot(2,2,1)

imshow(a)

title('Input Image 1');

subplot(2,2,2)

imshow(b)

title('Input Image 2');

subplot(2,2,3)

imshow(I5);

title('NOT(Input Image 1)');

subplot(2,2,4)

imshow(I6)

title('NOT(Input Image 2)');
```

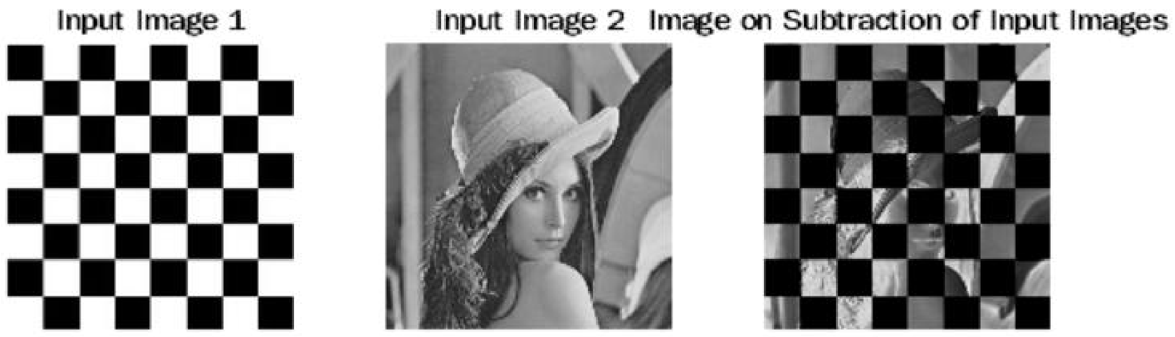
```
I7=a&b;

figure(7)

subplot(3,1,1)

imshow(a)

title('Input Image 1');

subplot(3,1,2)

imshow(b)

title('Input Image 2');

subplot(3,1,3)

imshow(I7)

title('AND Operation')


I8=a|b;

figure(8)

subplot(3,1,1)

imshow(a)

title('Input Image 1');

subplot(3,1,2)

imshow(b)

title('Input Image 2');

subplot(3,1,3)

imshow(I8)

title('OR Operation')
```

OUTPUT:

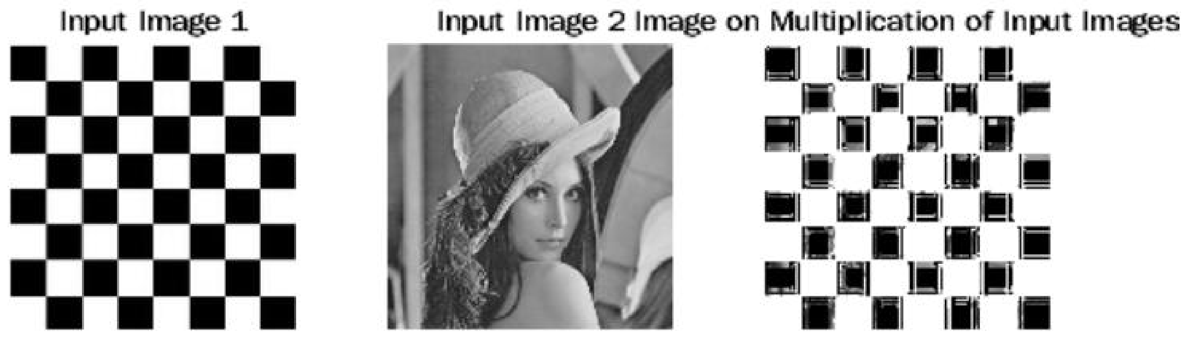
A. Addition



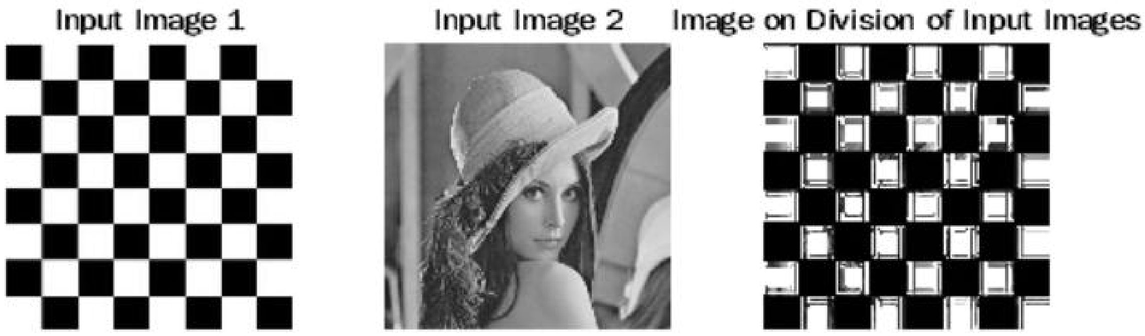
B. Subtraction



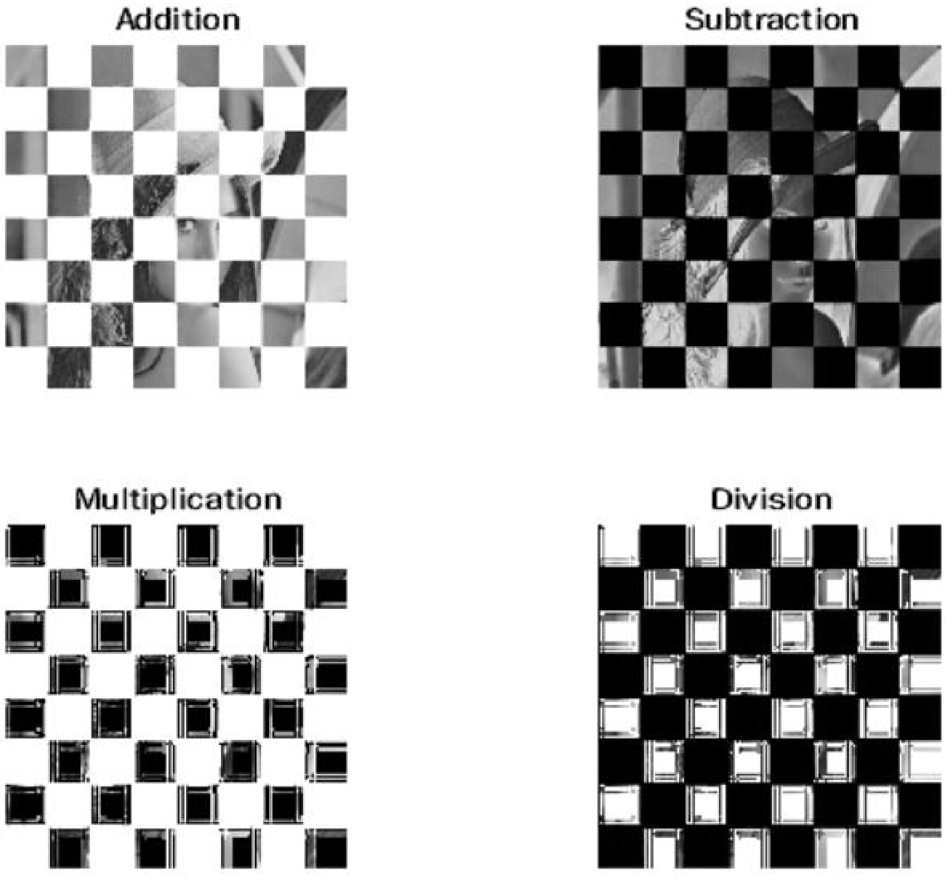
C. Multiplication



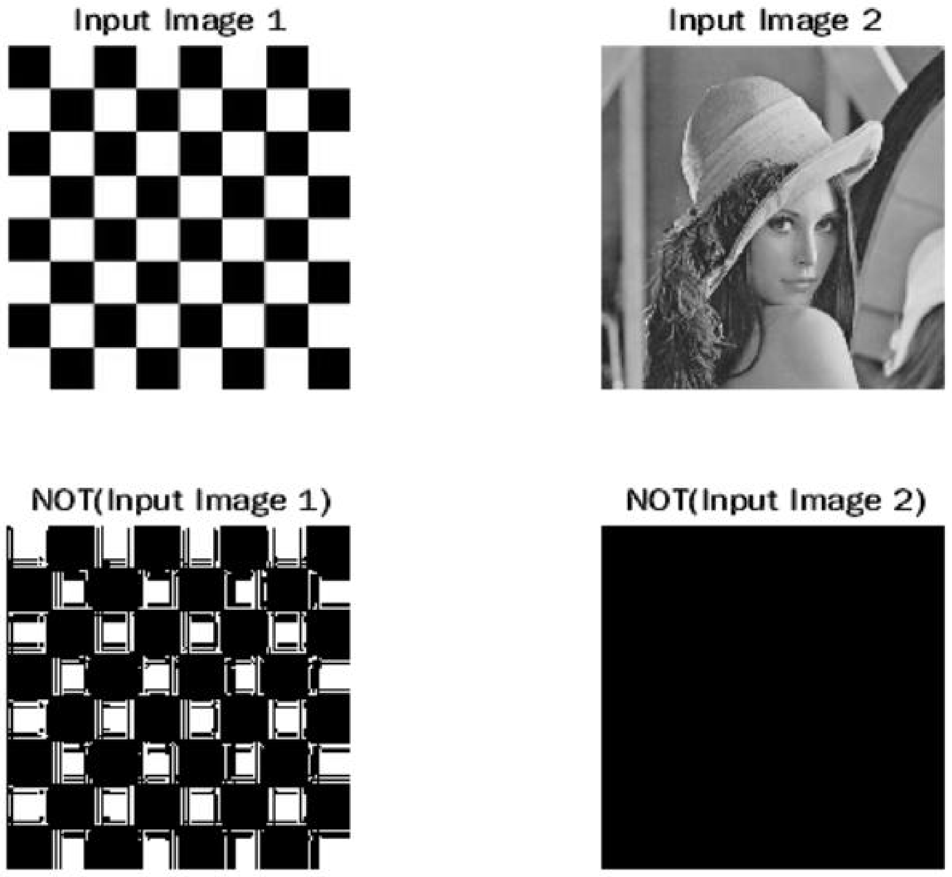
D. Division



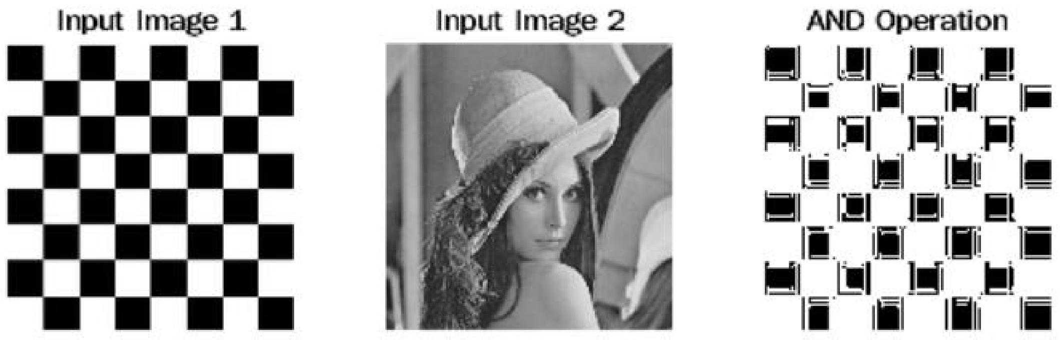
E. Arithmetic Operations



F. NOT Operation



G. AND Operation



H. OR Operation

INFERENCES & RESULT:

Arithmetic and Logical operations were performed on images using MATLAB.

Experiment -6

Name: Siddharth Bose **Reg. Number:** 19BEC1278 **Date:** 17/03/2022

Course Code: ECE3005 **Slot:** L23+L24 **Faculty:** Dr. Ranjeet Kumar

AIM: To remove noise in an image using frequency domain processing and measure the information loss.

TOOLS: MATLAB, Image Processing Toolbox

PROCEDURE:

1. Input an image on MATLAB.
2. Convert them to GrayScale using **rgb2gray**
3. Add Gaussian, Poisson and Salt & Pepper noise to the image.
4. Find the Fourier Transform.
5. Find the Inverse Fourier Transform of the images.
6. Print all the images.

CODE:

A. To add noise to image

```
clc
clear all
I=imread('dipimage.jpg');
I=rgb2gray(I);
I_N1=imnoise(I,'gaussian');
I_N2=imnoise(I,'poisson');
I_N3=imnoise(I,'salt & pepper');
```

B. To find the Fourier Transform of the noisy images

```
I_F1=fft(I_N1);
I_F2=fft(I_N2);
I_F3=fft(I_N3);
```

C. Magnification of Noisy Image

```
Fmag1=log(1+abs(I_F1));
Fmag2=log(1+abs(I_F2));
Fmag3=log(1+abs(I_F3));
figure(2)
subplot(3,1,1)
imshow(Fmag1,[ ],'Initialmagnification','fit');
title('Gaussian Noisy Image')
subplot(3,1,2)
imshow(Fmag2,[ ],'Initialmagnification','fit');
title('Poisson Noisy Image')
```

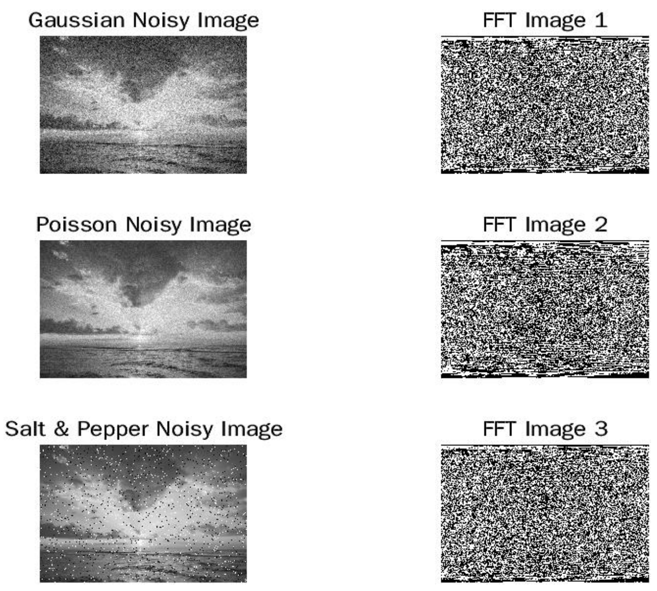
```
subplot(3,1,3)
imshow(Fmag3,[ ],'Initialmagnification','fit');
title('Salt & Pepper Noisy Image')
```

D. Arranging Frequency Components

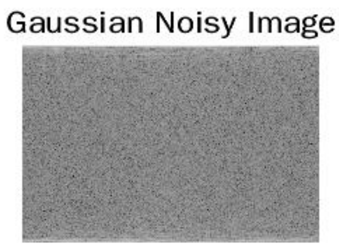
```
SF1=fftshift(I_F1);
SFmag1=log(1+abs(SF1));
SF2=fftshift(I_F2);
SFmag2=log(1+abs(SF2));
SF3=fftshift(I_F3);
SFmag3=log(1+abs(SF3));
figure(3)
subplot(3,1,1)
imshow(SFmag1,[ ],'Initialmagnification','fit');
title('Gaussian Noisy Image')
subplot(3,1,2)
imshow(SFmag2,[ ],'Initialmagnification','fit');
title('Poisson Noisy Image')
subplot(3,1,3)
imshow(SFmag3,[ ],'Initialmagnification','fit');
title('Salt & Pepper Noisy Image')
```

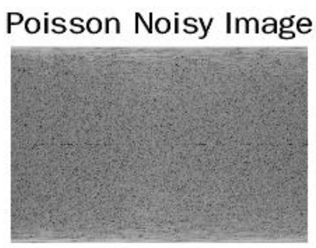
OUTPUT:

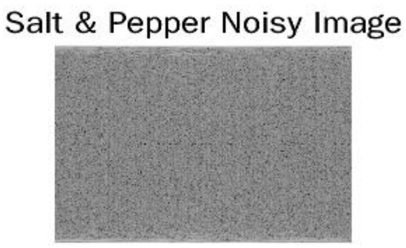
A. Noisy Images and their corresponding Fourier Transformed Images



B. Magnification of Noisy Image

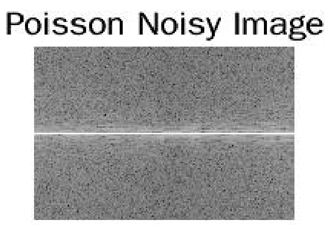


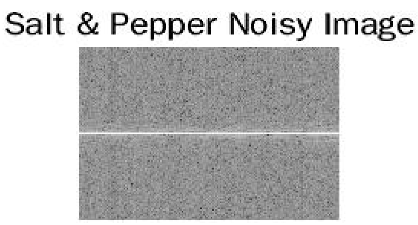




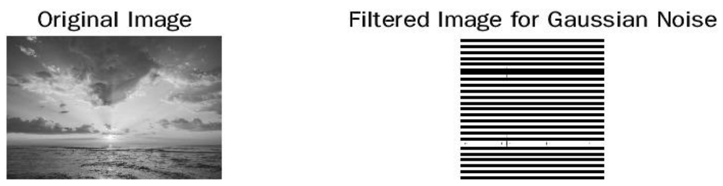
C. Arranging Frequency Components in Image

Gaussian Noisy Image





D. Inverse Fourier Transformed Images



INFERENCES:

1. The fourier transform of the noisy images were plotted.
2. The inverse fourier transform unveils that the images are not the same as the original images.

RESULT:

Fourier Transform was applied on input images with noise, using MATLAB.

LAB- 7

Name: Siddharth Bose

Reg. Number:19BEC1278

Date: 24/03/2022

Course Code: ECE3005

Slot: L23+L24

Faculty: Dr. Ranjeet Kumar

AIM: To compute the Fourier transform for given image and check the linearity property F[I1(x,y)+I2(x,y)]=F[I1(x,y)]+F[I2(x,y)]

TOOLS: MATLAB, Image Processing Toolbox

PROCEDURE:

1. Input two images on MATLAB.
2. Addition of the two images.
3. Performing fourier transform on the addition image.
4. Performing inverse fourier transform to get back the original image, thus performing the LHS of the Linearity property.
5. Performing fourier transform on the individual images and then adding them.
6. Performing inverse fourier transform on the addition image to get back the original image.
7. Print all the images.

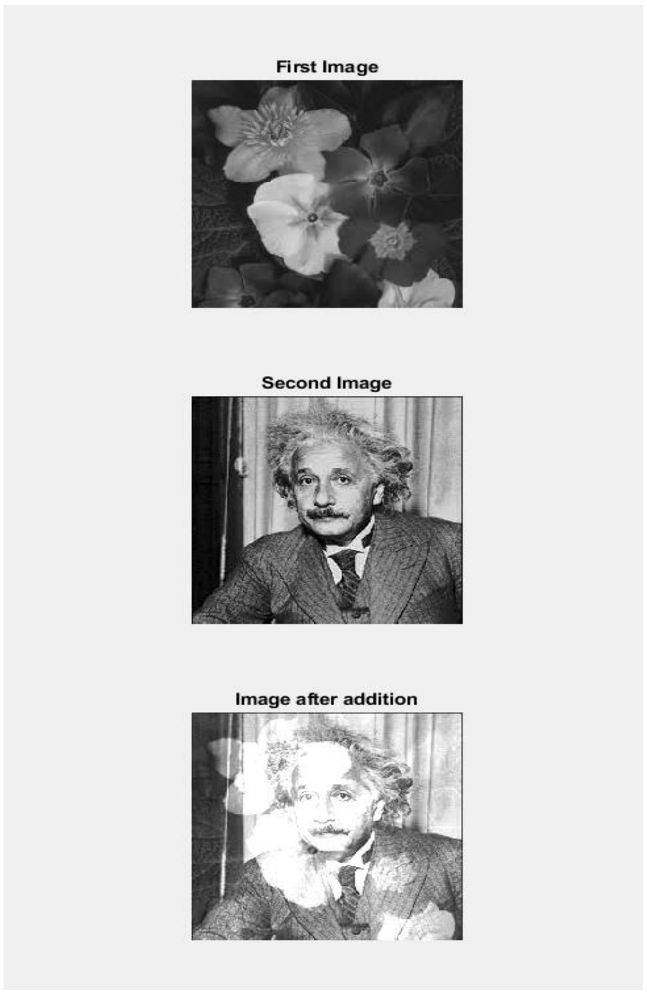
CODE:

```
clc;
clear all;

I1=imread("flower.jpg");
subplot(3,1,1);
imshow(I1);
title("First Image");

I2=imread("einstein.jpg");
subplot(3,1,2);
imshow(I2);
title("Second Image");

I=I1+I2;
subplot(3,1,3);
imshow(I);
```

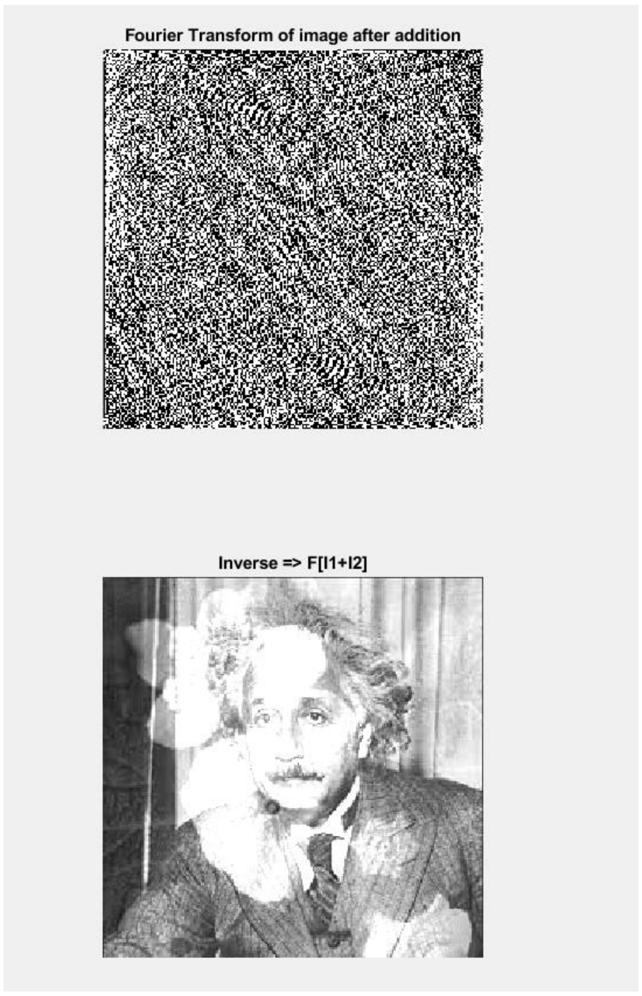
```
title("Image after addition");

%Fourier Transform of Addition Image
F1=fft2(I);
figure(2);
subplot(2,1,1);
imshow(F1);
title("Fourier Transform of image after addition");

%Inverse Fourier Transform of Addition image
IF1=ifft2(F1);
subplot(2,1,2);
imshow(uint8(IF1));
title("Inverse => F[I1+I2]");

F2=fft2(I1);
F3=fft2(I2);
F4=F2+F3;
IF2=ifft2(F4);
figure(3);
imshow(uint8(IF2));
title("Inverse => F[I1] + F[I2]");
```

OUTPUT:

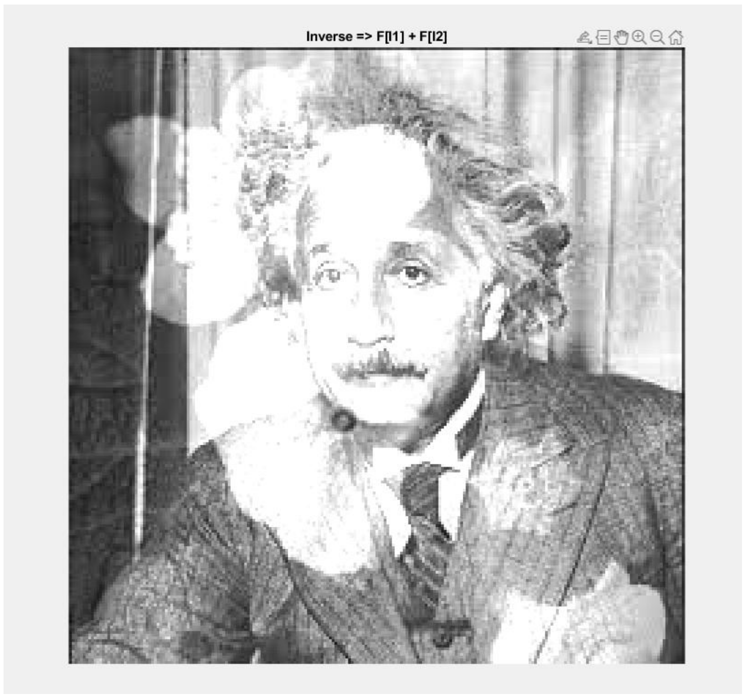
Performing basic image addition



Fourier transform and inverse fourier transform (LHS)



Inverse fourier transform showing RHS of Linearity property



INFERENCES:

- We’ve successfully performed fourier transform and inverse fourier transform on the sum of the 2 images .
- We’ve successfully performed fourier transform and inverse fourier transfrom on the individual images .
- After cross checking we can conclude that the linearity property has been proven using the code.

RESULT:

Linearity property has been proven using MATLAB

LAB- 8

Name: Siddharth Bose

Reg. Number:19BEC1278

Date: 07/04/2022

Course Code: ECE3005

Slot: L23+L24

Faculty: Dr. Ranjeet Kumar

AIM: To perform Discrete Cosine Transform on the given images using MATLAB

TOOLS: MATLAB, Image Processing Toolbox

THEORY:

A discrete cosine transform (DCT) expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. The DCT, first proposed by Nasir Ahmed in 1972, is a widely used transformation technique in signal processing and data compression. It is used in most digital media, including digital images (such as JPEG and HEIF, where small high-frequency components can be discarded), digital video (such as MPEG and H.26x), digital audio (such as Dolby Digital, MP3 and AAC), digital television (such as SDTV, HDTV and VOD), digital radio (such as AAC+ and DAB+), and speech coding (such as AAC-LD, Siren and Opus). DCTs are also important to numerous other applications in science and engineering, such as digital signal processing, telecommunication devices, reducing network bandwidth usage, and spectral methods for the numerical solution of partial differential equations.

PROCEDURE:

1. Insert input images using the command **imread**.

2. Perform discrete cosine transform using MATLAB.

3. Remove the high frequency components using had thresholding.

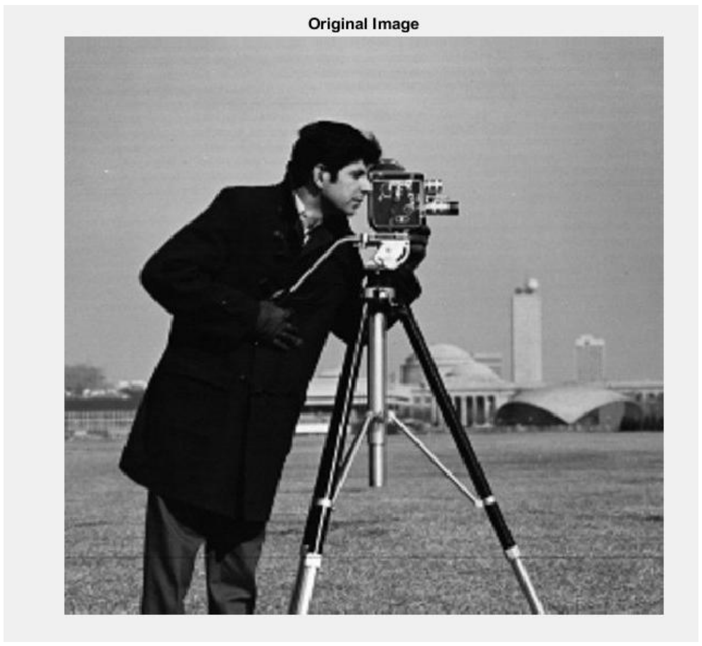
4. Perform reconstruction using inverse discrete cosine transform.

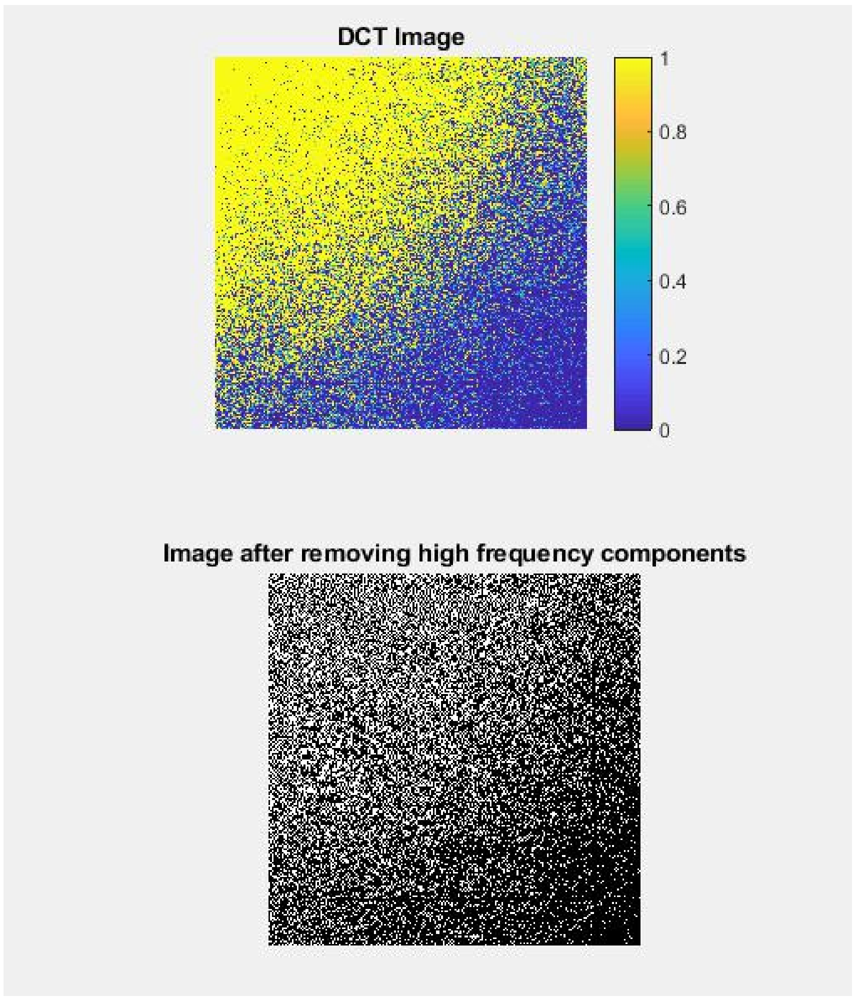
5. View the images using **imshow** and identify the differences.

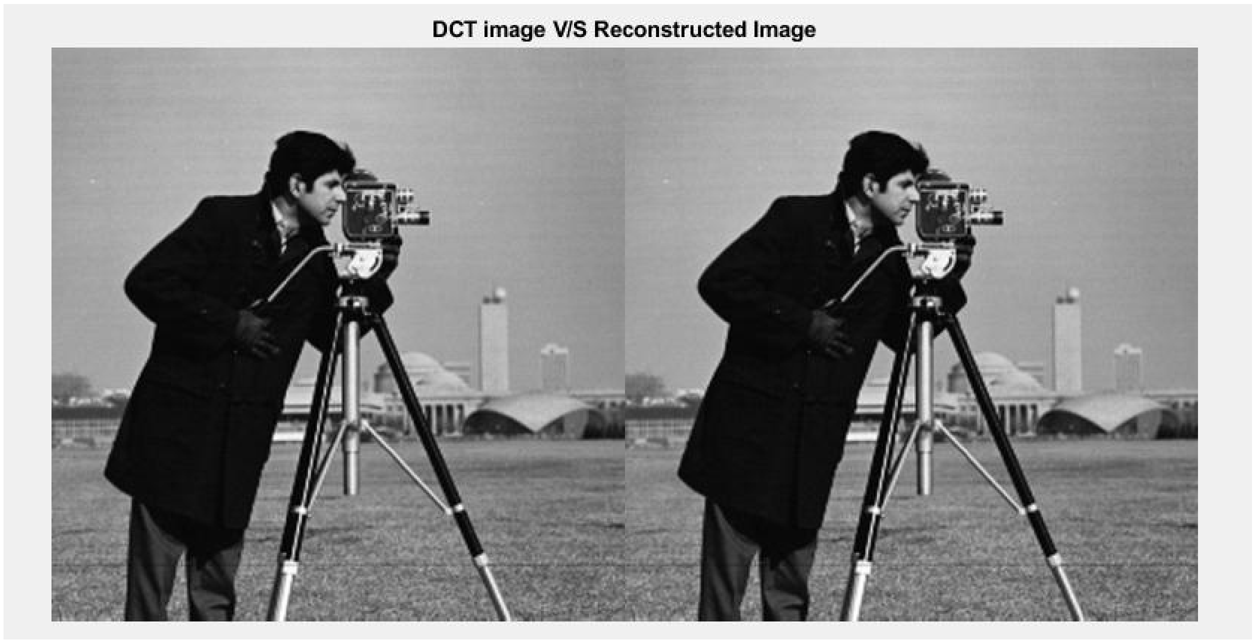
ALGORITHM/CODE:

```
%Aim:To perform Discrete Cosine Transform

clc;
clear all;

I= imread("C:\Users\student\Downloads\Gray.jpg");
figure(1);
imshow(I);
title("Original Image");

%Perfroming DCT
I1=dct2(I);
figure(2)
subplot(2,1,1);
imshow(log(abs(I1)));
title("DCT Image");
colormap parula
colorbar

%Removing High Frequency Component
% I2 = I1(abs(I1)<5 = 0);  %Hard thresholding
I1(abs(I1)<1) = 0;
subplot(2,1,2);
imshow(I1);
title("Image after removing high frequency components");

%Reconstruction
I2 = idct2(I1);
figure(3);
montage({I,uint8(I2)});
title("DCT image V/S Reconstructed Image")
```

OUTPUT:







INFERENCES & RESULT:

Discrete cosine transform was performed on images using MATLAB.

LAB- 9

Name: Siddharth Bose

Reg. Number:19BEC1278

Date: 07/04/2022

Course Code: ECE3005

Slot: L23+L24

Faculty: Dr. Ranjeet Kumar

AIM: To perform image analysis using Discrete Wavelet Transform and denoising the given images using MATLAB

TOOLS: MATLAB, Image Processing Toolbox

THEORY:

In numerical analysis and functional analysis, a discrete wavelet transform (DWT) is any wavelet transform for which the wavelets are discretely sampled. As with other wavelet transforms, a key advantage it has over Fourier transforms is temporal resolution: it captures both frequency and location information (location in time).

PROCEDURE:

1. Insert input images using the command **imread**.

2. Perform discrete wavelet transform using MATLAB.

3. Remove the high frequency components using had thresholding.

4. Perform reconstruction using inverse discrete wavelet transform.

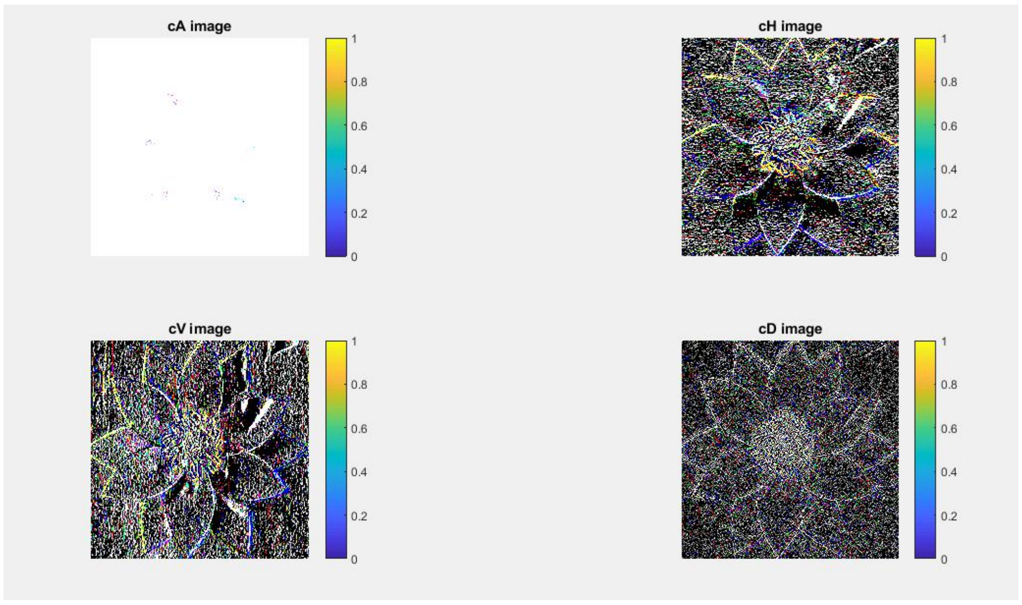
5. View the images using **imshow** and identify the differences.

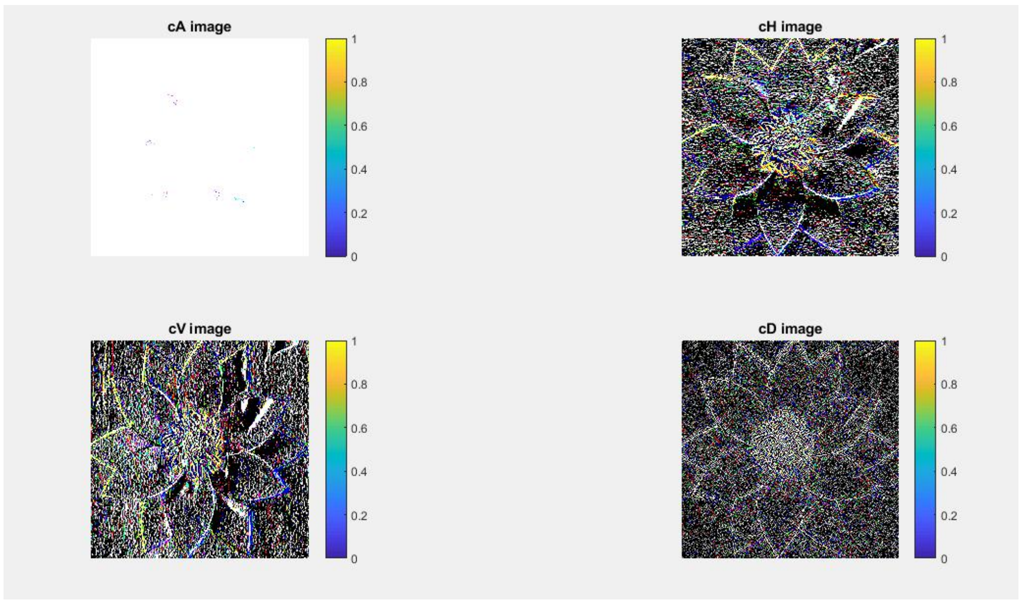
ALGORITHM/CODE:

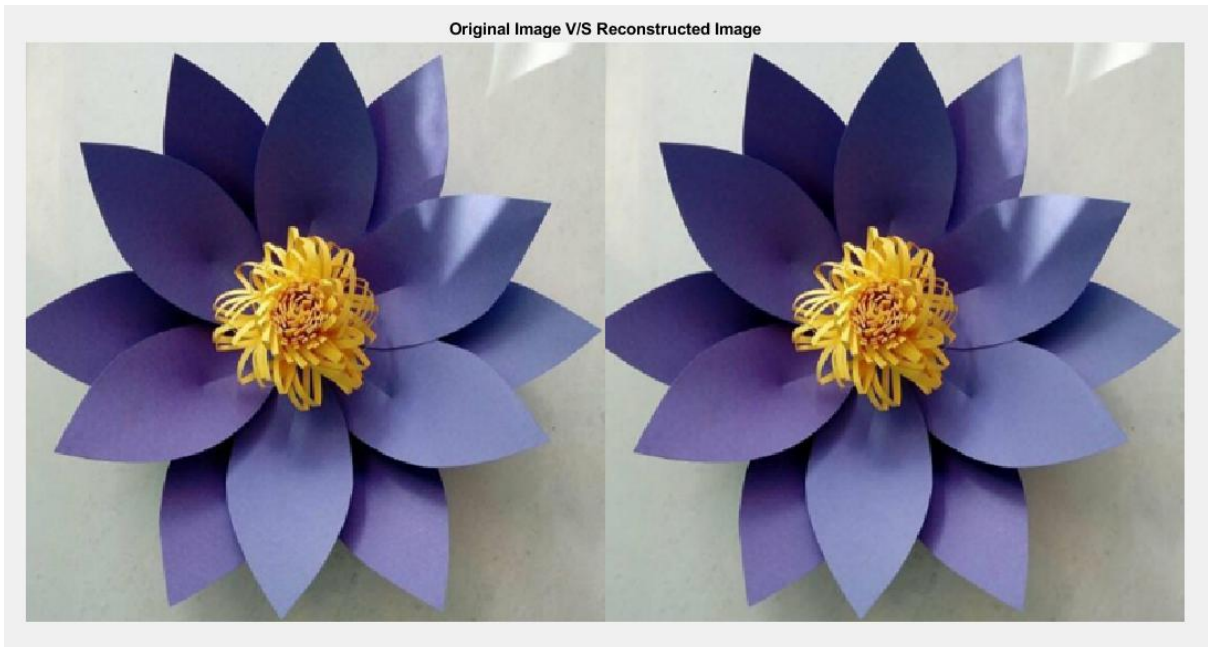
```
%Aim: Image analysis with DWT and denoising of image

clc;
clear all;

I= imread("C:\Users\student\Downloads\flower.jpg");
figure(1);
imshow(I);
title("Original Image");

%Perfroming DWT
[cA,cH,cV,cD] = dwt2(I,'haar')
figure(2);
subplot(2,2,1);
imshow(cA);
title("cA image");
colormap parula
colorbar
subplot(2,2,2);
imshow(cH);
title("cH image");
colormap parula
colorbar
subplot(2,2,3);
imshow(cV);
title("cV image");
colormap parula
colorbar
subplot(2,2,4);
imshow(cD);
title('cD image');
colormap parula
colorbar

%Thresholding
cA(abs(cA)<5) = 0;
cH(abs(cH)<5) = 0;
cV(abs(cV)<5) = 0;
cD(abs(cD)<5) = 0;
figure(3);
subplot(2,2,1);
imshow(cA);
title('cA image without high frequency components');
colormap parula
colorbar
subplot(2,2,2);
imshow(cH);
title('cH image without high frequency components');
colormap parula
colorbar
subplot(2,2,3);
imshow(cV);
title('cV image without high frequency components');
colormap parula
colorbar
subplot(2,2,4);
imshow(cD);
title('cD image without high frequency components');
```

```
colormap parula
colorbar

%Reconstuction
I1 = idwt2(cA,cH,cV,cD,'haar');
figure(4);
montage({I,uint8(I1)});
title('Original Image V/S Reconstructed Image');
```

OUTPUT:







INFERENCES & RESULT:

Discrete wavelet transform and denoising was performed on images using MATLAB.

LAB- 10

Name: Siddharth Bose

Reg. Number:19BEC1278

Date: 21/04/2022

Course Code: ECE3005

Slot: L23+L24

Faculty: Dr. Ranjeet Kumar

AIM: To perform Image Resolution Enhancement using Discrete Wavelet Transform in MATLAB

TOOLS: MATLAB, Image Processing Toolbox

THEORY:

In numerical analysis and functional analysis, a discrete wavelet transform (DWT) is any wavelet transform for which the wavelets are discretely sampled. As with other wavelet transforms, a key advantage it has over Fourier transforms is temporal resolution: it captures both frequency and location information (location in time).

PROCEDURE:

1. Insert input images using the command **imread**.

2. Perform Direct Resolution Enhancement.

3. View the images before and after enhancement using **imshow.**

4. Perform discrete wavelet transform using MATLAB.

5. Perform reconstruction using inverse discrete wavelet transform.

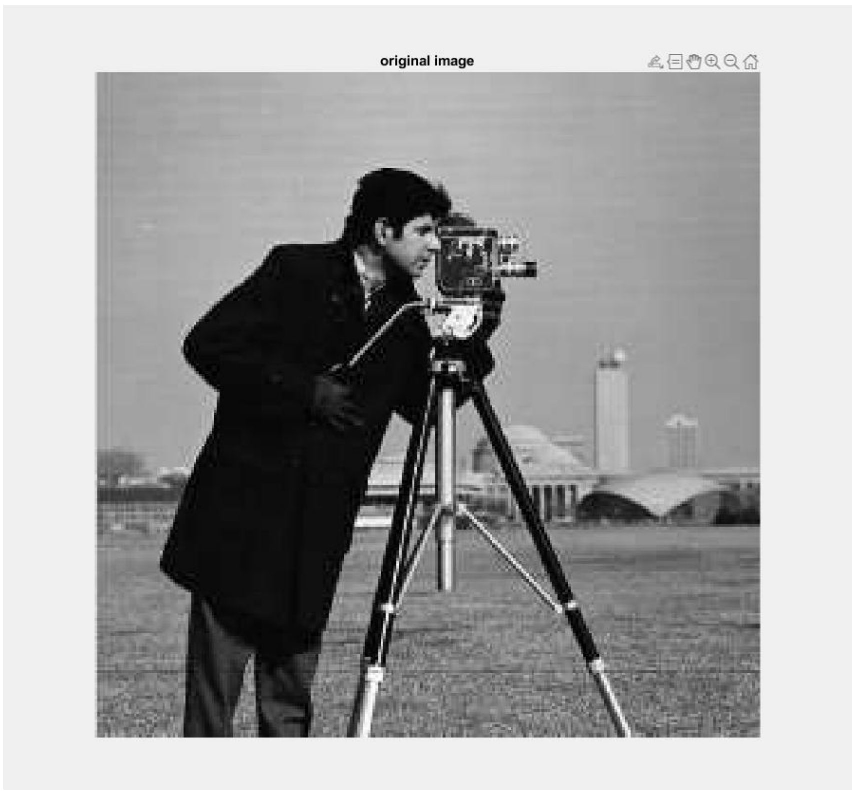
6. View the images using **imshow** and identify the differences between the Direct Resolution Enhancement method and DWT method.

ALGORITHM/CODE:

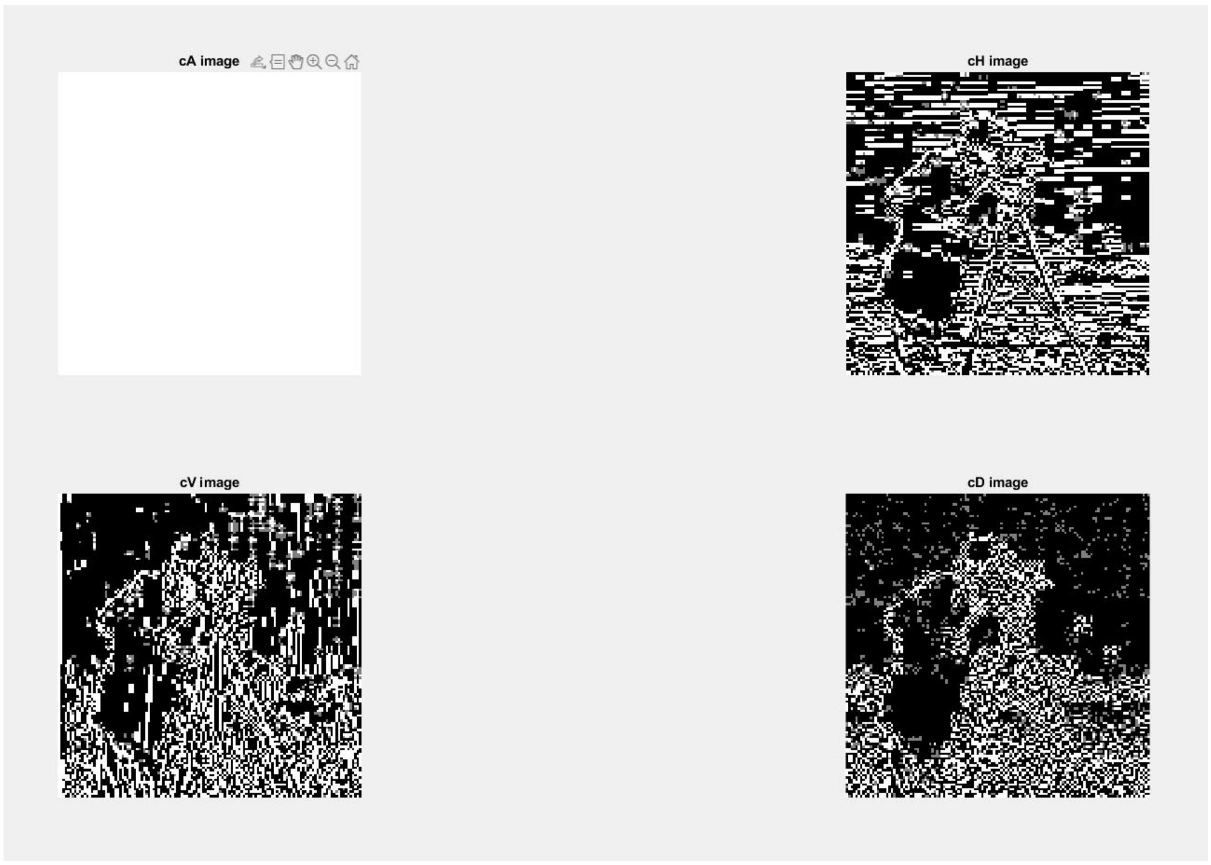
```
%lab 10 Image Resolution Enhancement using DWT (2 parts --> direct
resolution enhancement and DWT with comparison)
clc;
close all;

%Direct Resolution Enhancement
I=imread("CAMERA.jpg");
I1=imresize(I,2); %interpolation in time/space domain
figure(1);
imshow(I);
title('original image');
figure(2);
imshow(I1);
title('Direct Resolution Enhancement Image');


%DWT Enhancement
I2=imread("CAMERA.jpg");
[cA,cH,cV,cD] = dwt2(I2,'haar')
figure(3);
subplot(2,2,1);
imshow(cA);
title("cA image");
subplot(2,2,2);
imshow(cH);
title("cH image");
subplot(2,2,3);
imshow(cV);
title("cV image");
subplot(2,2,4);
imshow(cD);
title('cD image');

cA1=imresize(cA,2); %interpolation in wavelet domain
cH1=imresize(cH,2);
cV1=imresize(cV,2);
cD1=imresize(cD,2);
figure(4);
subplot(2,2,1);
imshow(cA1);
title("cA resized image");
subplot(2,2,2);
imshow(cH1);
title("cH resized image");
subplot(2,2,3);
imshow(cV1);
title("cV resized image");
subplot(2,2,4);
imshow(cD1);
title('cD resized image');
```

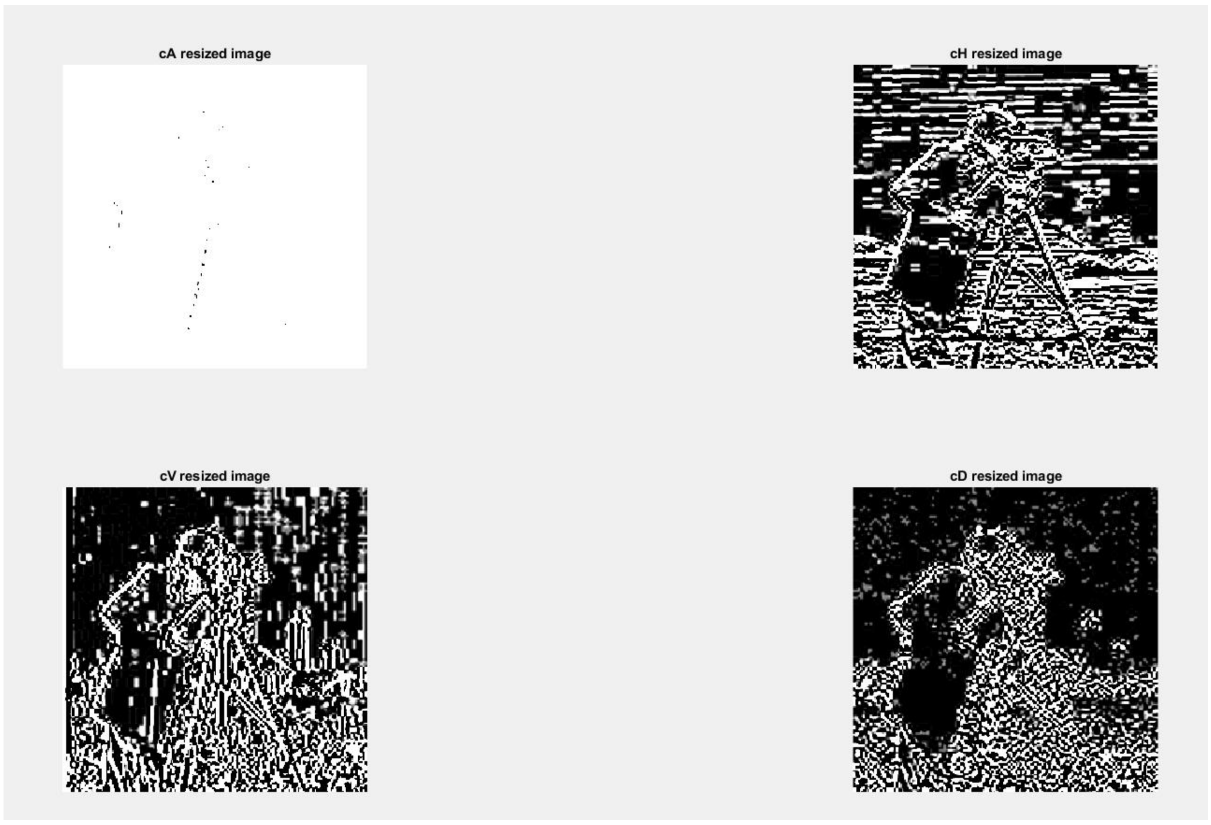
```
I3 = idwt2(cA1,cH1,cV1,cD1,'haar');
figure(5);
montage({I1,uint8(I3)});
title('DIRECT RESOLUTION ENHANCEMENT V/S  DWT ENHANCEMENT');
```

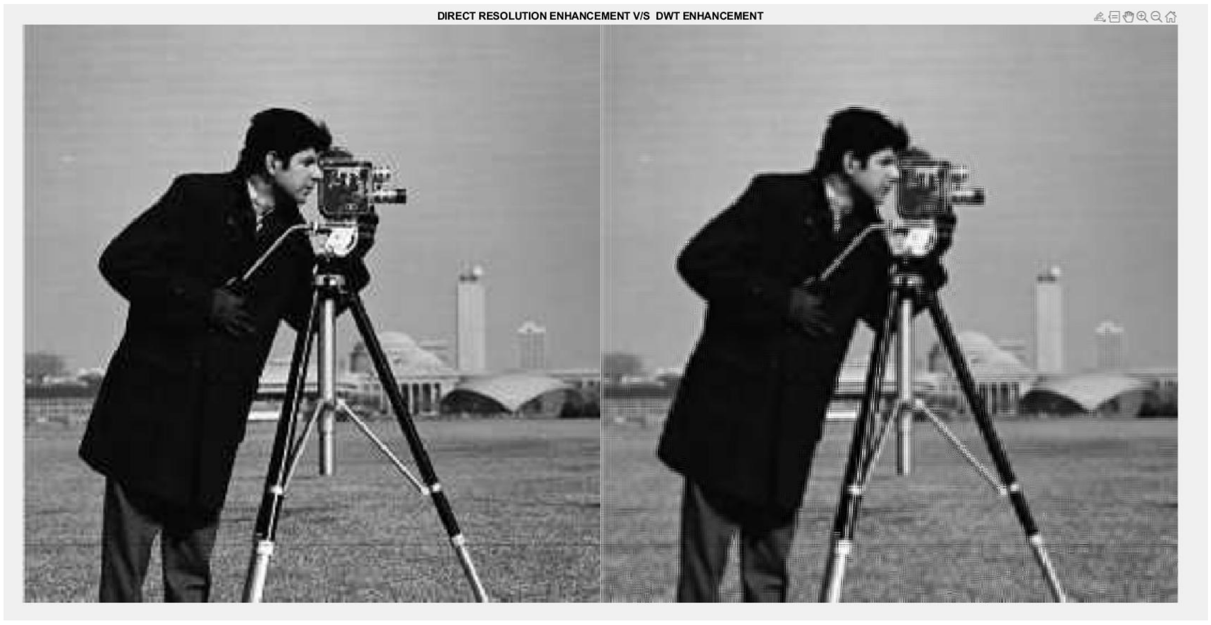
OUTPUT:



Direct Resolution Enhancement Image


cA image
cH image
cV image
cD image





INFERENCES & RESULT: MATLAB was used to perform discrete wavelet transform improvement on photographs.