FUNDAMENTALS OF IMAGE PROCESSING

(2181102)

B.E. 8th SEMESTER

LABORATORY MANUAL

2021

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that Mr. **Prashant Ranpura** Enrolment No. **170280111081** has satisfactorily completed course in **Fundamentals of Image Processing** at L. D. College of Engineering, Ahmedabad – 380015.

Date of Submission : 28/04/2021
Staff in-charge :
HOD :

List of Experiments

Sr. No.	Name of Experiment
1.	Write program to read and display digital image using MATLAB i. Become familiar with MATLAB Basic commands Read and display image in MATLAB ii. Resize given image iii. Convert given color image into gray-scale image iv. Convert given color/gray-scale image into black & white image Draw image profile v. Separate color image in three R G & B planes vi. Create color image using R, G and B three separate planes Flow control and LOOP vii. Write given 2-D data in image file
2.	To write and execute image processing programs using point processing method i. Obtain Negative image ii. Obtain Flip image iii. Thresholding iv. Contrast stretching
3.	To write and execute programs for image arithmetic operations i. Addition of two images ii. Subtract one image from other image iii. Calculate mean value of image iv. Different Brightness by changing mean value
4.	To write and execute programs for image logical operations i. AND operation between two images ii. OR operation between two images iii. Calculate intersection of two images
5.	To write a program for histogram calculation and equalization i. Standard MATLAB function ii. Program without using standard MATLAB functions iii. Use Simulink to plot histogram of color image
6.	To write and execute program for geometric transformation of image i. Translation ii. Scaling iii. Rotation iv. Shrinking v. Zooming

7.	To understand various image noise models and to write programs for image restoration i. Remove Salt and Pepper Noise ii. Minimize Gaussian noise iii. Median filter and Weiner filter
8.	Write and execute programs to remove noise using spatial filters i. Understand 1-D and 2-D convolution process ii. Use 3x3 Mask for low pass filter and high pass filter
9.	Write and execute programs for image frequency domain filtering i. Apply FFT on given image ii. Perform low pass and high pass filtering in frequency domain iii. Apply IFFT to reconstruct image
10.	Write a program in MATLAB for edge detection using different edge detection mask
11.	Write and execute program for image morphological operations Erosion and dilation.
12.	To write and execute program for wavelet transform on given image and perform inverse wavelet transform to reconstruct image.

EXPERIMENT NO. 1

AIM: Write program to read and display digital image using MATLAB**.**

Reading an Image

To import an image from any supported graphics image file format, in any of the supported bit depths, use the imread function.

Syntax A = imread(filename,fmt)

Description

A = imread(filename,fmt) reads a greyscale or color image from the file specified by the string
filename, where the string fmt specifies the format of the file. If the file is not in the current
directory or in a directory in the MATLAB path, specify the full pathname of the location on
your
system.

>> b= imread('C:\Documents and Settings\bond\My Documents\ld1 (8 x 8).jpg')

Display An Image

To display iamge, use the imshow function.

Syntax imshow(A)

Description

imshow(A) displays the image stored in array A.

Writing Image Data/ Writing Image to Disk

Imwrite Write image to graphics file

Syntax

imwrite(A,filename,fmt)

Description

It writes the image to filename, inferring the format to use from the filename's extension. The extension must be one of the values.

>>imwrite(a,'b1.jpg','jpeg')
>>imshow('b1.jpg')

How to get no. of rows and columns of image

Syntax [r,c]=size(a)

Description Function **size** gives the rows and columns dimension of image

Accessing the Pixel data

There is a one-to-one correspondence between pixel coordinates and the coordinates MATLAB uses for matrix subscripting. This correspondence makes the relationship between an image's data matrix and the way the image is displayed easy to understand. For example, the data for the pixel in the fifth row, second column is stored in the matrix element (5,2). You use normal MATLAB matrix subscripting to access values of individual pixels.

For example, MATLAB Code A(2,15) returns the value of the pixel at row 2, column 15 of the image A.

How to get Image Information

Syntax info = imfinfo(filename)

Description

It returns a structure, info, whose fields contain information about an image in a graphics file. filename is a string that specifies the name of the graphics file.The file must be in the current directory or in a directory on the MATLAB path.

```
>> imfinfo('C:\Documents and Settings\bond\My Documents\ld1 (8 x 8).jpg')
```

Convertion of Image

Syntax Im2unit8

Description Convert image to 8-bit unsigned integers I1 = im2uint8(I)

Syntax Im2bw

Description Convert image to binary image, based on threshold.

BW = im2bw(I, level)

Syntax Im2double

Description Convert image to double type image.

I2=im2double(I)

Displaying gray Convertion of Image

Syntax Im2unit8

Description Convert image to 8-bit unsigned integers I1 = im2uint8(I)

Syntax Im2bw

Description Convert image to binary image, based on threshold.

BW = im2bw(I, level)

Syntax Im2double

Description Convert image to double type image.

I2=im2double(I)

Displaying gray scale image
Syntax imtool(I,[low high])
Desription
It displays the grayscale image I, specifying the display range for I in the vector [low high]. The value low (and any value less than low) is displayed as black, the value high (and any value greater
than high) is displayed as white.

Changing the size of Image
Syntax B = imresize(A, scale)
Description
It returns image B that is scale times the size of A. The input image A can be a grayscale, RGB, or binary image. If scale is between 0 and 1.0, B is smaller than A. If scale is greater than 1.0, B is larger than A.
Example B=imresize(a,0.5) making image B half of the image size of A.
B=imresize(a,[512 512]) making image b with size of 512X512.

Rotating the image
Syntax B = imrotate(A,angle)
Description
It rotates image A by angle degrees in a counter clockwise direction around its center point. To rotate the image clockwise, specify a negative value for angle. imrotate makes the output image B large enough to contain the entire rotated image. imrotate uses nearest neighbour interpolation, setting the values of pixels in B that are outside the rotated image to 0 (zero).

Reading the color value of an Image
Syntax P = impixel(I)
Description
impixel returns the red, green, and blue color values of specified image pixels. In the syntax, impixel displays the input image and waits for you to specify the pixels with the mouse.

Converting color Image to Gray scale Image
Syntax I = rgb2gray(RGB)
Description
It converts the truecolor image RGB to the grayscale intensity image I. rgb2gray converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

Detecting edges of an Image

Syntax BW = edge(I)

Description It takes a grayscale or a binary image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere.

EXPERIMENT NO. 2

AIM: To write and execute image processing programs using point processing methods

i. Obtain Negative image
ii. Obtain Flip image
iii. Threshold operation
iv. Contrast Stretching

Introduction:

Image enhancement can be done in two domain: [1] Spatial Domain and [2] Transform domain.

In Spatial domain Image processing, there are two ways:

Point processing (Single pixel is processed at a time)

Neighborhood processing (Mask processing) Convolution of 3x3 or 5x5 or other size of mask with image

In point processing, Grey level transformation can be given by following equation

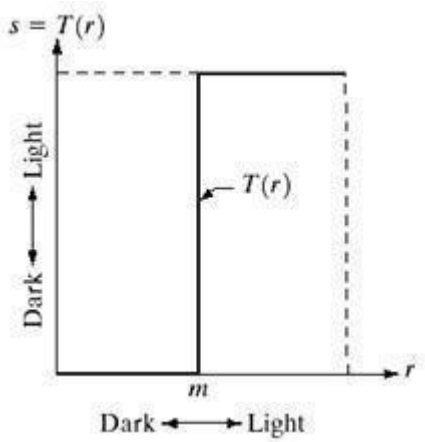
- $g(x,y)=T(f(x,y)) \rightarrow s=T(r)$

Where, f(x,y) is original image, g(x,y) is transformed image

s = gray level of transformed image

r = gray level of original image

Threshold operation:



In threshold operation, Pixel value greater than threshold value is made white and pixel value less than threshold value is made black. If we consider image having gray levels ***r*** and if grey levels in output image is ***s*** than,

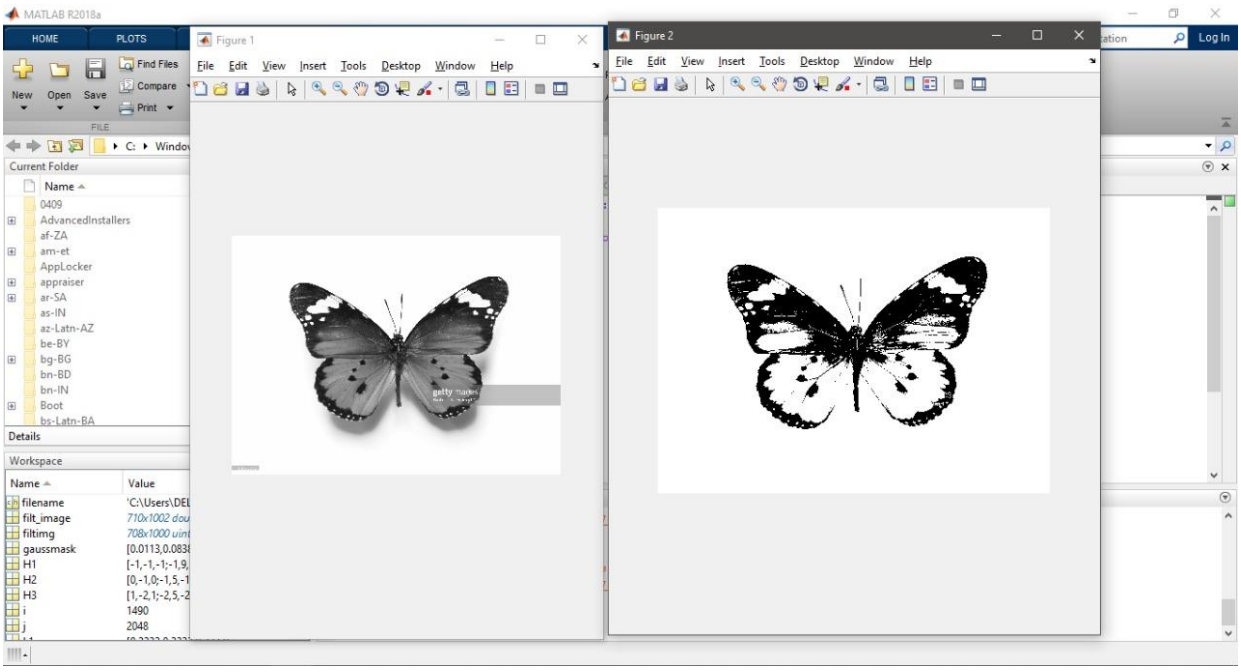
$$s = 0 \quad \text{if } r \leq m$$

$$s = 255 \quad \text{if } r > m$$

Where m = threshold

MATLAB Program for Thresholding:

```
% Experiment No. 2 Thresholding (Extreme contrast stretching)
% Scan any document and use this method to make it clean
% Example: scan your own signature and make it clean with
thresholding
filename=input('Enter file name: ','s');
y=imread(filename);
T=input('Enter threshold value between 0 to 255: ');
if(ndims(y)==3)
         y=rgb2gray(y);
end
[m,n]=size(y);
imshow(y); for
i=1:m
     for j=1:n
          if(y(i,j)>T)
               y(i,j)=255;
          else
               y(i,j)=0;
          end
     end
end figure;
imshow(y);
```

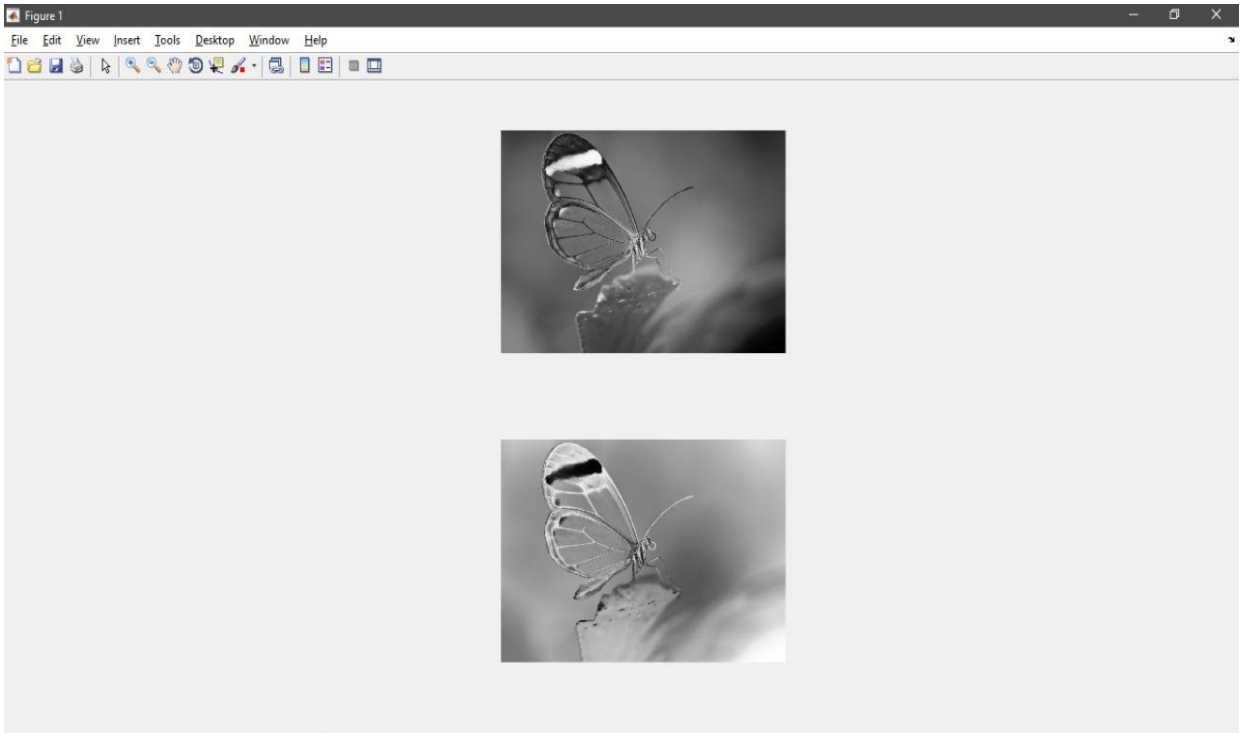


This is of 100 threshold.

Run above program for different threshold values and find out optimum threshold value for which you are getting better result.

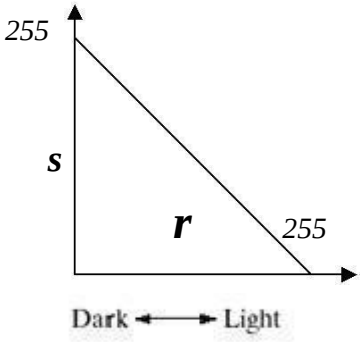
Horizontal flipping:

```
% Experiment 2: Flip given image horizontally
% Read in an image
filename=input('Enter file name:
','s'); imagedata = imread(filename);
if(ndims(imagedata)==3)
         imagedata=rgb2gray(imagedata);
end
%Determine the size of the image
[rows,cols] = size(imagedata);
%Declare a new matrix to store the newly created flipped
image FlippedImageData = zeros(rows,cols);
%Generate the flipped image
for r = 1:rows
  for c = 1:cols
                FlippedImageData(r,cols+1-c) = imagedata(r,c);
  end
end
%Display the original image and flipped image
subplot(2,1,1); imshow(imagedata) ;
subplot(2,1,2); imshow(FlippedImageData,[]);
%Write flipped image to a file
imwrite(mat2gray(FlippedImageData),'FlipTest.jpg','quality',99);
```



Negative Image:

Negative Image can be obtained by subtracting each pixel value from 255.



$$s = 255-r$$

MATLAB Program to obtain Negative Image:

```
% Experiment No. 2
% To obtain Negative Image
 clc;
close;
%Original Image
[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.jpg;*.gif','IMAG E
Files   (*.bmp,*.tif,*.jpg,*.gif)'},'Choose   GrayScale   Image');
img=imread(strcat(pathname,namefile));
if(size(img,3)==3)
      img=rgb2gray(img);
end
[rows,cols]=size(img);
neg_img=zeros(rows,cols);
for r = 1:rows
  for c = 1:cols
        neg_img(r,c) = 255-img(r,c);  %Calculate negative image
  end
end
%Display original and negative
images subplot(2,1,1); imshow(img);
subplot(2,1,2); imshow(neg_img,[]);
```

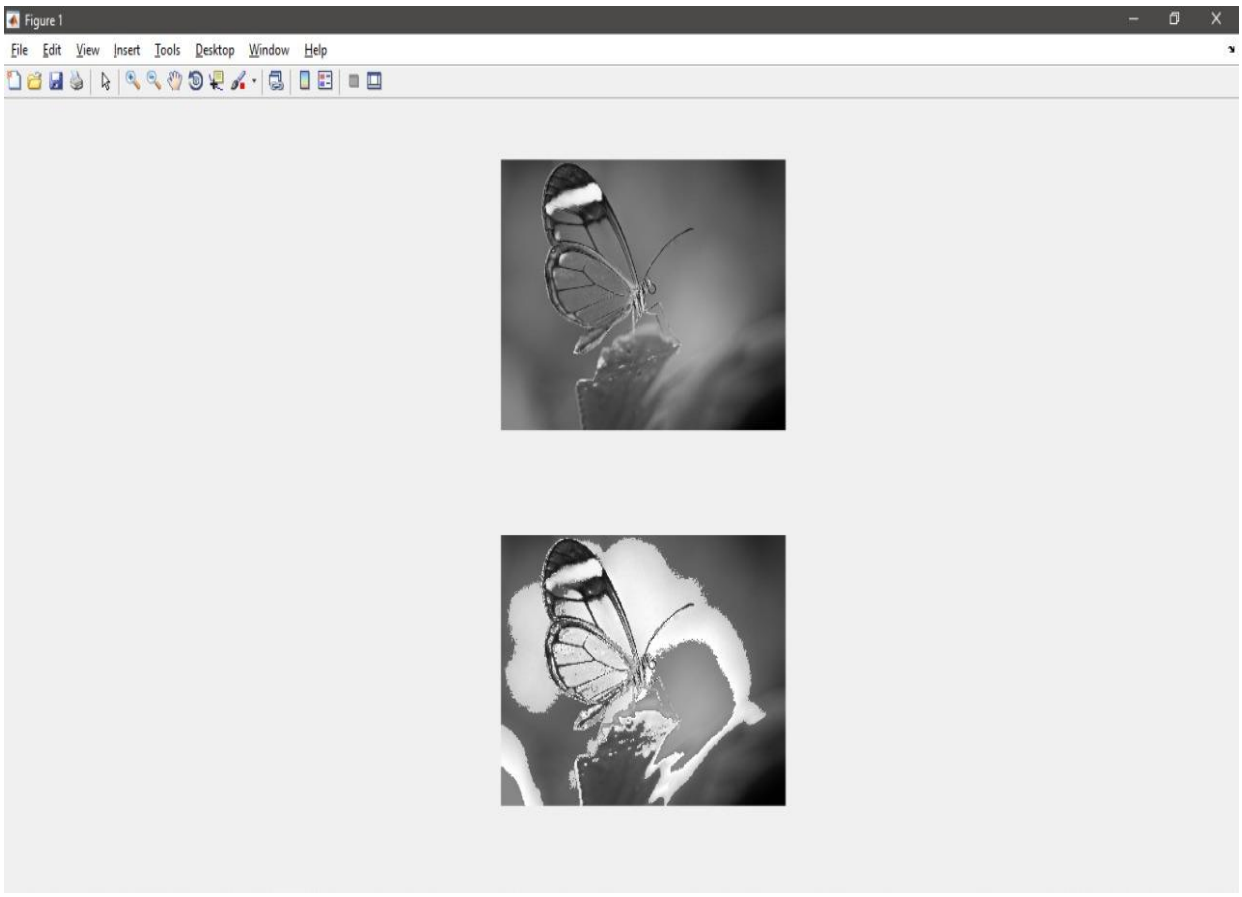
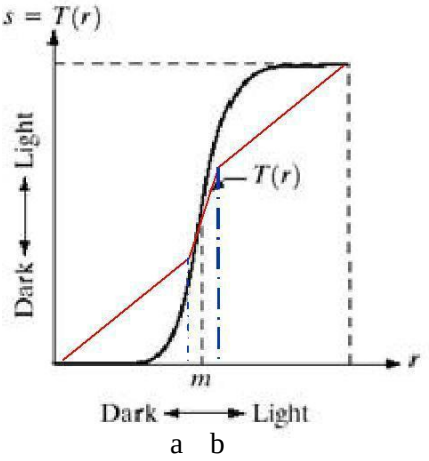


Figure 1
File Edit View Insert Tools Desktop Window Help

Contrast Stretching:

Contrast Stretching means Darkening level below threshold value **m** and whitening level above threshold value **m**. This technique will enhance contrast of given image. Thresholding is example of extreme constrast stretching.



Practically contrast stretching is done by piecewise linear approximation of graph shown in left by following equations:

$$s = x.r \qquad 0 \le r < a$$

$$s = y.r \qquad a \le r < b$$

$$s = z.r \qquad b \le r < 255$$

Where x,y and z are different slopes

RED line shows piecewise approximation

MATLAB Program for contrast stretching:

```
% Experiment No. 2 Contrast stretching using three slopes
% and two threshold values a & b
clc;
clear all;
[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpe
g;*.gif','IMAGE Files
(*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'},'Chose GrayScale
Image');
img=imread(strcat(pathname,namefile));
if(size(img,3)==3)
      img=rgb2gray(img);
end
[row, col]=size(img);
newimg=zeros(row,col);
a=input('Enter  threshold  value  a:
');  b=input('Enter  threshold  value
b: '); for i=1:row
    for j=1:col
        if(img(i,j)<=a)
            newimg(i,j)=img(i,j);
        end
        if (img(i,j)>a && img(i,j)<=b)
                newimg(i,j)=2*img(i,j);
        end
        if(img(i,j)>b)
                newimg(i,j)=img(i,j);
        end
    end
end
subplot(2,1,1); imshow(uint8(img));
subplot(2,1,2); imshow(uint8(newimg));
```

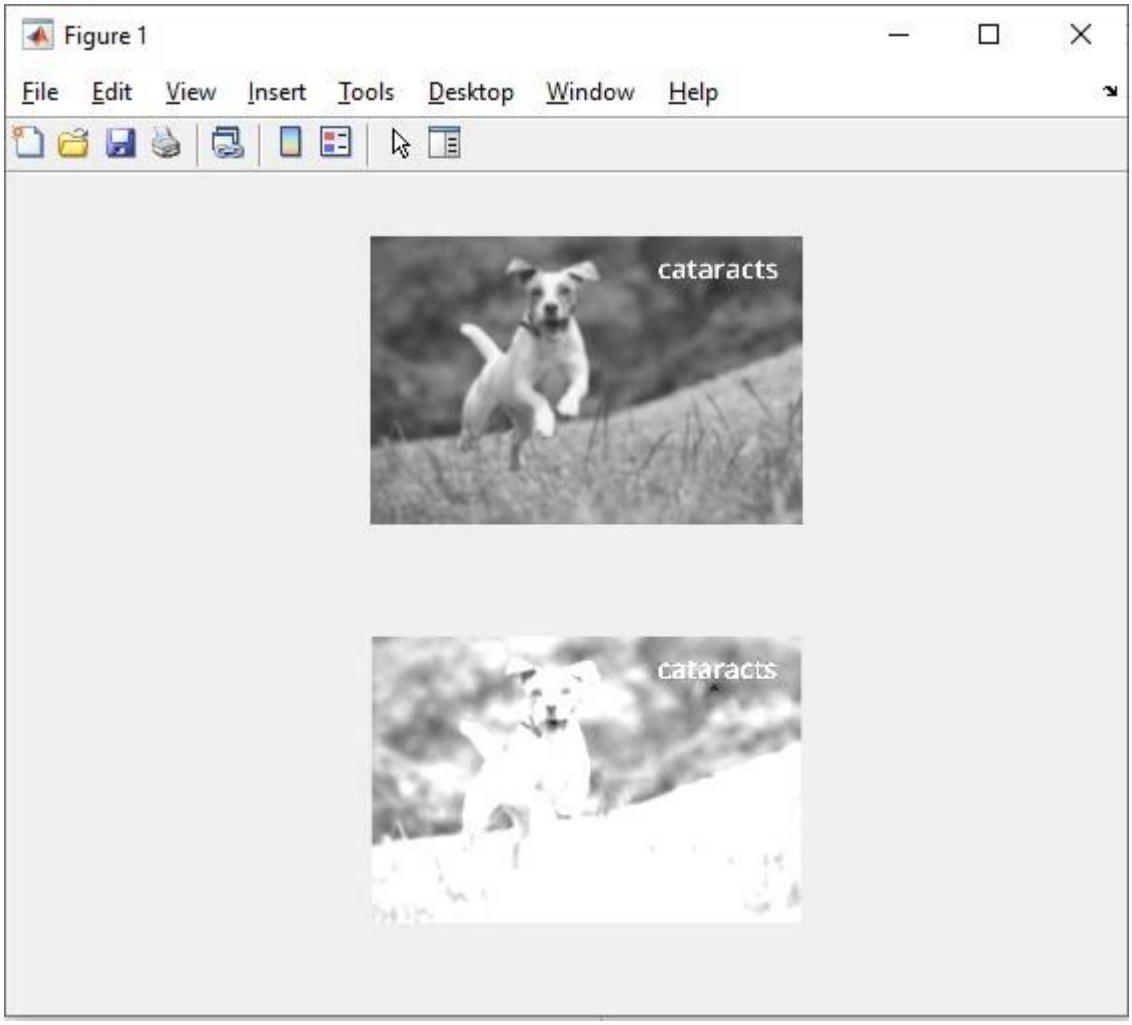


Figure 1
File Edit View Insert Tools Desktop Window Help
cataracts
cataracts

EXPERIMENT NO. 3

AIM: To write and execute programs for image arithmetic operations

Introduction: Arithmetic operations like image addition, subtraction, multiplication and division is possible. If there is two images I1 and I2 then addition of image can be given by:

$$I(x,y) = I1(x,y) + I2(x,y)$$

Where I(x,y) is resultant image due to addition of two images. x and y are coordinates of image. Image addition is pixel to pixel. Value of pixel should not cross maximum allowed value that is 255 for 8 bit grey scale image. When it exceeds value 255, it should be clipped to 255.

To increase overall brightness of the image, we can add some constant value depending on brightness required. In example program we will add value 50 to the image and compare brightness of original and modified image.

To decrease brightness of image, we can subtract constant value. In example program, value 100 is subtracted for every pixel. Care should be taken that pixel value should not less than 0 (minus value). Subtract operation can be used to obtain complement (negative) image in which every pixel is subtracted from maximum value i.e. 255 for 8 bit greyscale image.

Multiplication operation can be used to mask the image for obtaining region of interest. Black and white mask (binary image) containing 1s and 0s is multiplied with image to get resultant image. To obtain binary image function im2bw() can be used. Multiplication operation can also be used to increase brightness of the image

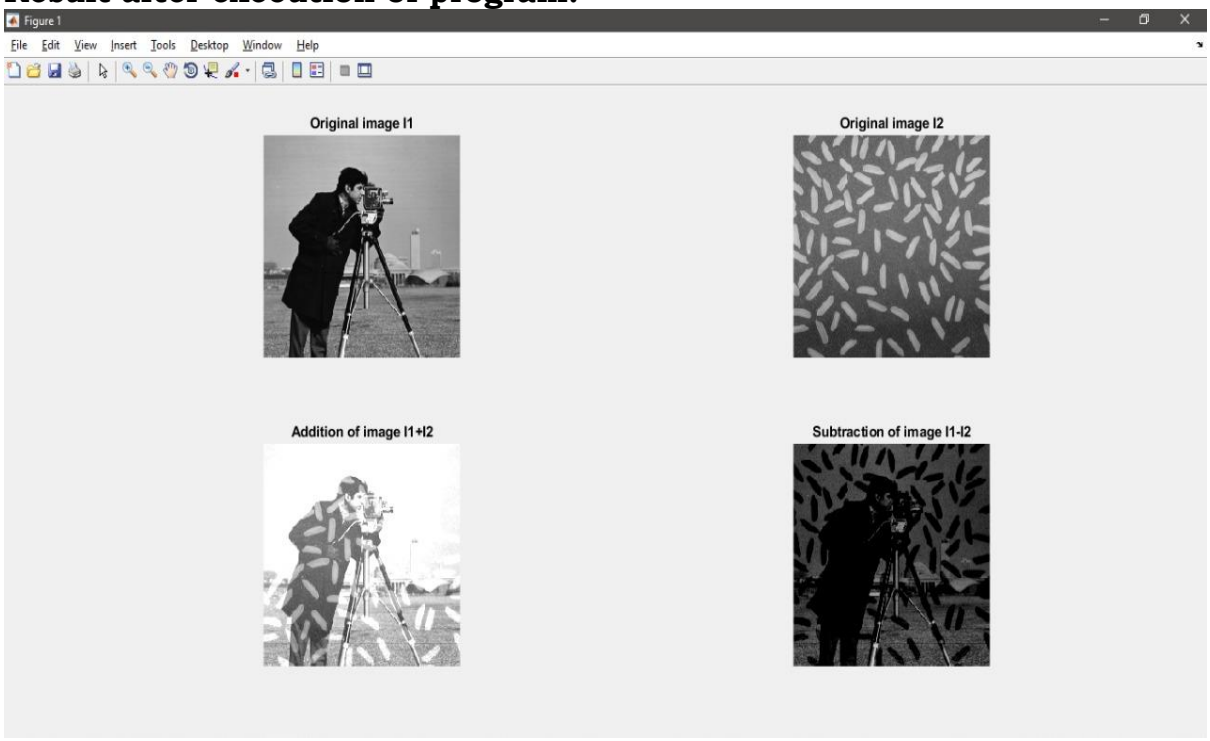
Division operation results into floating point numbers. So data type required is floating point numbers. It can be converted into integer data type while storing the image. Division can be used to decrease the brightness of the image. It can also used to detect change in image.

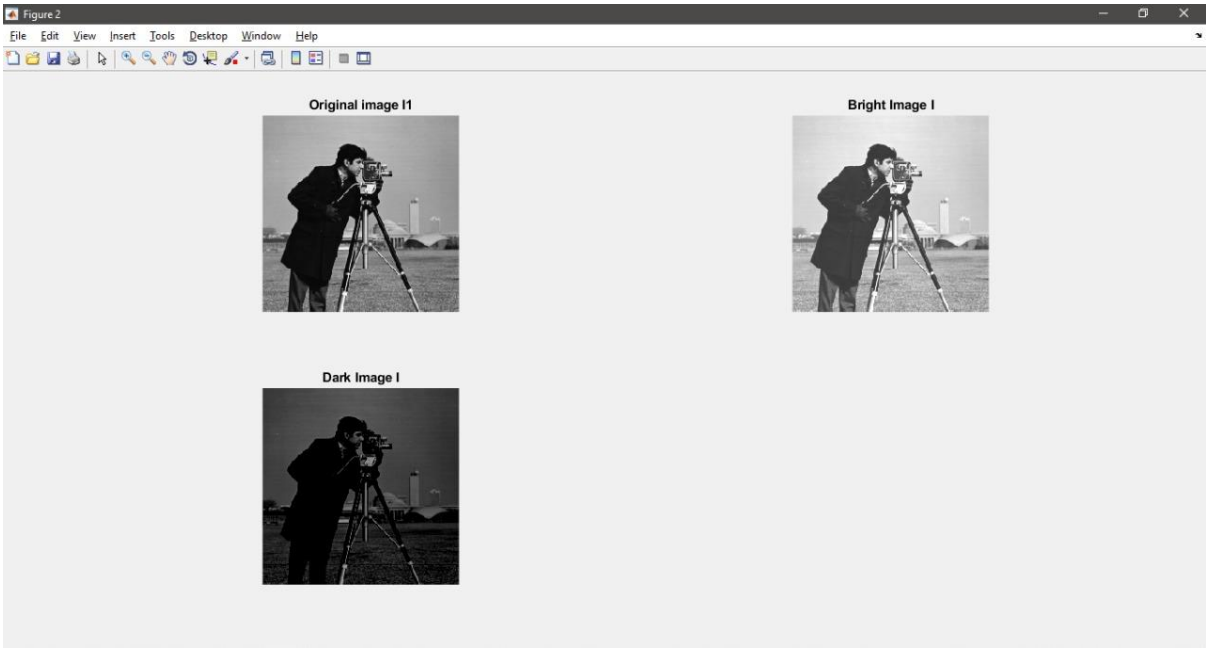
Execute following MATLAB program:

```
%Experiment No. 3 Image Arithmetic Operations
%Image addition is done between two similar size of image, so image
resize %function is used to make size of both image same.
%I=I1+I2
clc close
all
I1 = imread('cameraman.tif');
I2 = imread('rice.png');
```

```
subplot(2, 2, 1);imshow(I1);title('Original image
I1'); subplot(2, 2, 2);imshow(I2);title('Original
image I2'); I=I1+I2; % Addition of two images
subplot(2, 2, 3);imshow(I);title('Addition of image
I1+I2'); I=I1-I2; %Subtraction of two images
subplot(2, 2, 4);imshow(I);title('Subtraction of image I1-I2');
figure;
subplot(2, 2, 1);imshow(I1);title('Original image
I1'); I=I1+50;
subplot(2, 2, 2);imshow(I);title('Bright Image
I'); I=I1-100;
subplot(2, 2, 3);imshow(I);title('Dark
Image I'); M=imread('Mask.tif');
M=im2bw(M) % Converts into binary image having 0s and 1s
I=uint8(I1).*uint8(M); %Type casting before multiplication
subplot(2, 2, 4);imshow(I);title('Masked Image I');
```

Result after execution of program:



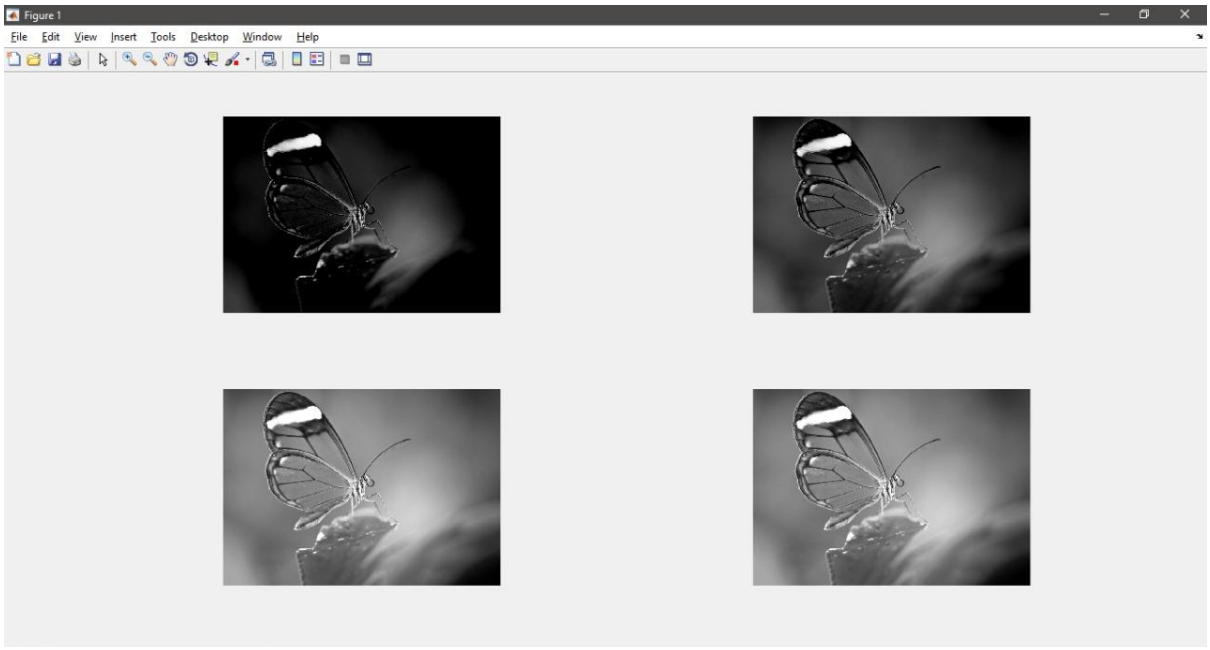


Display image with different brightness:

```
close all;
clear all;
[filename,pathname]=uigetfile({'*.bmp;*.jpg;*.gif','Choose
Image File'});
 myimage=imread(filename);
if(size(myimage,3)==3)
        myimage =rgb2gray(myimage);
end
[Rows, Cols] = size(myimage)
newimage=zeros(Rows,Cols);
k=1;
while k<5, for
i = 1:Rows
      for j = 1:Cols
            if k==1
                newimage(i,j)=myimage(i,j)-100;
            end
            if k==2 newimage(i,j)=myimage(i,j)-
                50;
            end
            if k==3
                newimage(i,j)=myimage(i,j)+50;
            end
            if k==4
                newimage(i,j)=myimage(i,j)+50;
            end

      end
end
subplot(2,2,k);
imshow(newimage,[]); k=k+1;
```

```
end
```

Result:



Calculate mean value:

Mean value can be calculated by MATLAB function mean()

Alternately following code can be used to calculate mean value:

```
[Rows, Cols] = size(myimage)
newimage=zeros(Rows,Cols);
total = 0;
for i = 1:Rows
      for j = 1:Cols
            total = total + myimage(i,j);
      end
end
average=total/(Rows*Cols)
```

EXPERIMENT NO. 4

AIM: To write and execute programs for image logical operations

Introduction: Bitwise logical operations can be performed between pixels of one or more than one image.

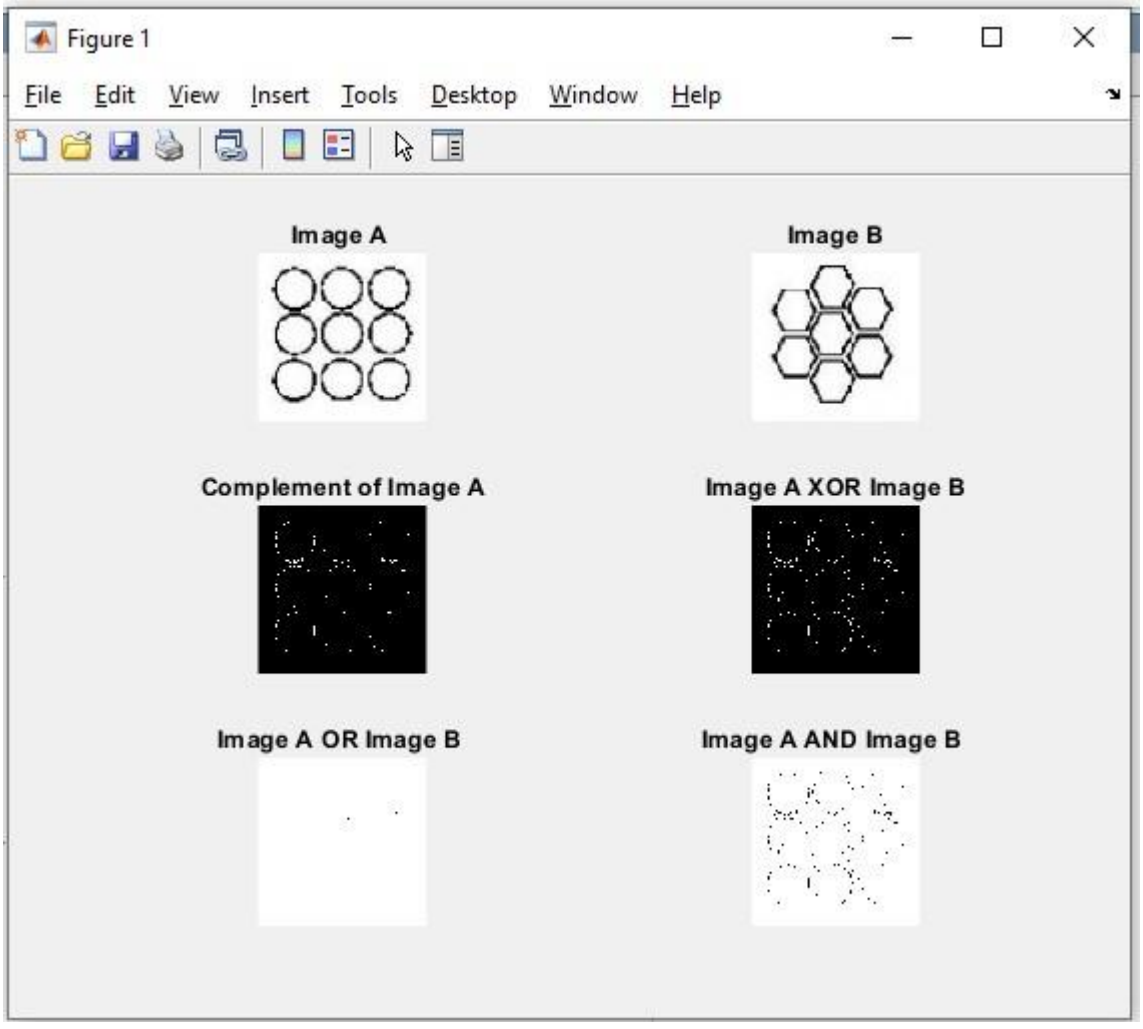
i. AND/NAND Logical operations can be used for following applications:
   a. Compute intersection of the images
   b. Design of filter masks
ii. Slicing of gray scale images
iii. OR/NOR logical operations can be used for following applications:
iv. Merging of two images
v. XOR/XNOR operations can be used for following applications:
   a. To detect change in gray level in the
   b. image Check similarity of two images
vi. NOT operation is used for:
vii. To obtain negative image
viii. Making some features clear

Program:

```
Experiment No. 4 Image Logical Operations
%Logical operations between two image circle and pentagon is
shown clc
close
all clc
close all
myimageA=imread('circle.jpg');
myimageA=rgb2gray(myimageA);
myimageB=imread('pentagon.jpg');
myimageB=rgb2gray(myimageB);
subplot(3,2,1)
imshow(myimageA),title('Image A ');

% Display the Original Image B
subplot(3,2,2)
imshow(myimageB),title('Image B');

% Take a complement of Image A and Display
it subplot(3,2,3)
cimageA= ~myimageA ;
imshow(cimageA), title('Complement of Image A');

% Take a Ex-OR of Image A and Image B and Display
it subplot(3,2,4)
xorimage= xor(myimageA,myimageB);
imshow(xorimage), title('Image A XOR Image
B');
```

```
% Take OR of Image A and Image B and Display
it subplot(3,2,5)
orimage= myimageA | myimageB;
imshow(orimage), title('Image A OR Image B
');

% Take AND of Image A and Image B and Display
it subplot(3,2,6)
andimage= myimageA & myimageB;
imshow(andimage), title('Image A AND Image B
');
```

Result after execution of program:



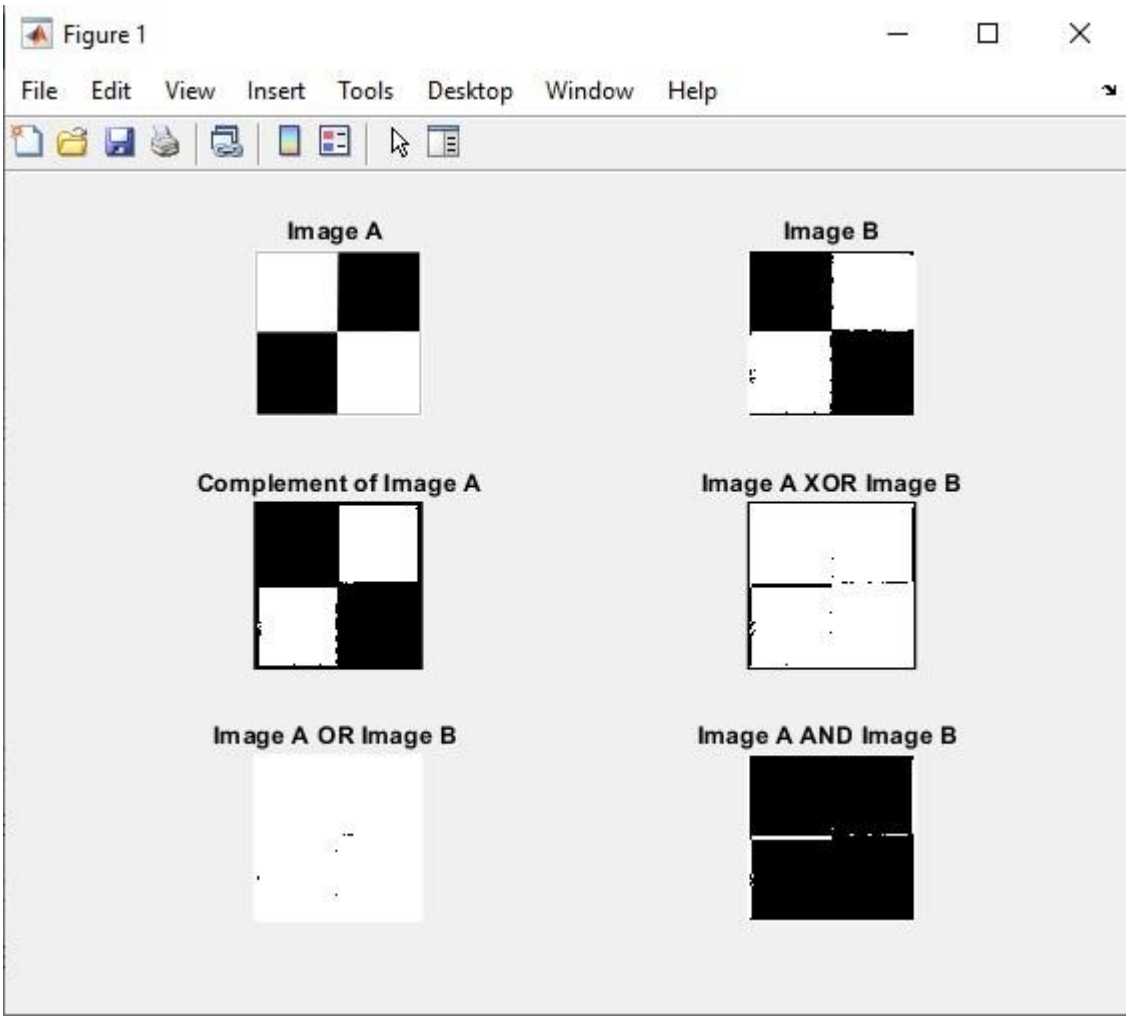


Figure 1
File Edit View Insert Tools Desktop Window Help
Image A
Image B
Complement of Image A
Image A XOR Image B
Image A OR Image B
Image A AND Image B

EXPERIMENT NO. 5

AIM: To write a program for histogram calculation and equalization

a. Standard MATLAB function
   i. Program without using standard MATLAB functions C Program
b. Use Simulink to plot histogram of colour image

Introduction:

- Histogram is bar-graph used to profile the occurrence of each gray level in the image
- Mathematically it is represented as $\mathbf{h(r_k)=n_k}$
  Where r_k is k_{th} grey level and n_k is number of pixel having that grey level
- Histogram can tell us whether image was scanned properly or not. It gives us idea about tonal distribution in the image.
- Histogram equalization can be applied to improve appearance of the image
- Histogram also tells us about objects in the image. Object in an image have similar gray levels so histogram helps us to select threshold value for object detection.
- Histogram can be used for image segmentation.

Pseudo code:

Calculate histogram:

```
Loop over ROWS of the image              i = 1 to rows
        Loop over COLS of then image     j = 1 to cols
                        Pixel value       k=data(i,j)
                        hist(k)=hist(k)+1
            End COLS loop
End ROWS loop
```

Calculate sum of the hist:

```
Loop over gray levels i = 0 to no. of gray levels
                        sum=sum+hist(i);
                        sum_of_hist(i)=sum;
End loop
```

Area of image = rows*cols
Dm=Number of gray levels in the output image

```
Loop over ROWS Loop
        over COLS
                k = data(i,j)
                data(i,j) = (Dm/area)*sum_of_hist(k)
        End COLS oop
End ROWS Loop
```

Standard MATLAB function for histogram and histogram equalization:

[1] imhist function

It computes histogram of given image and plot Example**:**

```
myimage=imread(„tier.jpg")
imhist(myimage);
```

[1] histeq function

It computes histogram and equalize it. Example**:**

```
myimage = imread(„rice.bmp");
newimage= histeq(myimage);
```

Sample program using standard functions:

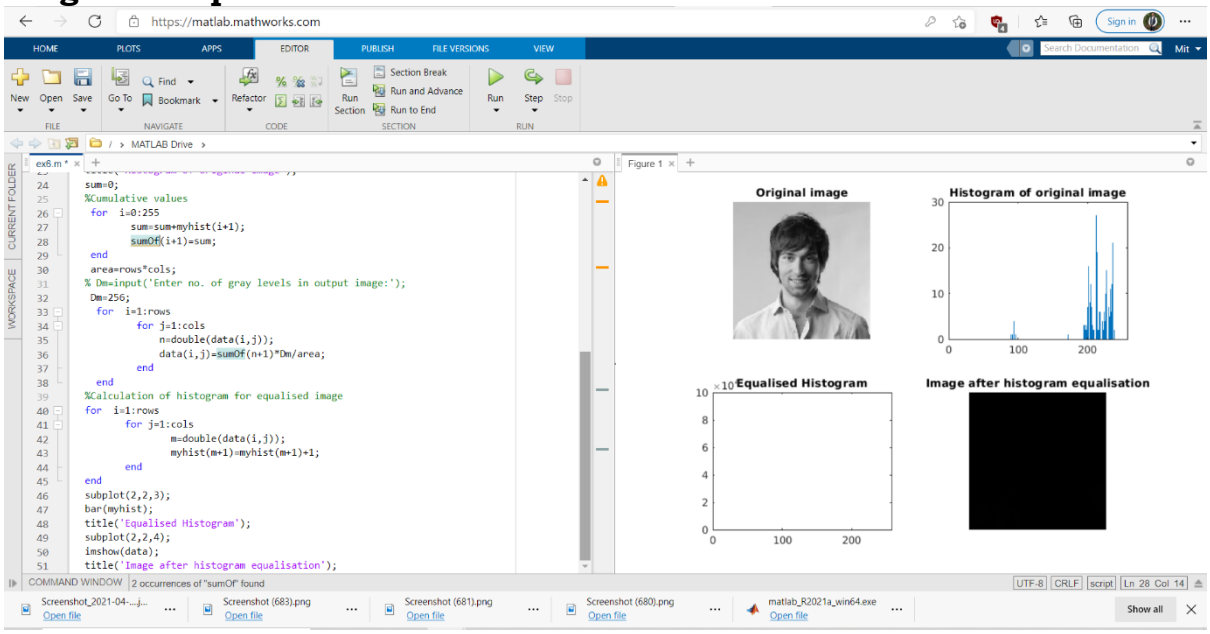
```
close all;
clear all;
[filename,pathname]=uigetfile({'*.bmp;*.jpg;*.gif','Choose
Poorly scanned Image'});
 myimage=imread(filename);
if(size(myimage,3)==3)
        myimage =rgb2gray(myimage);
end
imhist(myimage);
newimage=
histeq(myimage); figure;
subplot(2,2,1);imshow(myimage); title('Original image');
subplot(2,2,2);imshow(newimage); title('Histogram equalized
mage');
```

Sample Program in MATLAB (without using standard function):

```
% Experiment No. 5
%Program for calculation and equalisation of the
histogram %  close all;
clear all;
[filename,pathname]=uigetfile({'*.bmp;*.jpg;*.gif','Choose
Poorly scanned Image'});

data=imread(filename);

if(size(data,3)==3)
data=rgb2gray(data);
end
```

```
subplot(2,2,1);imshow(data); title('Original
image'); [rows cols]=size(data)
myhist=zeros(1,256);
% Calculation of
histogram for i=1:rows
   for j=1:cols m=double(data(i,j));
    myhist(m+1)=myhist(m+1)+1;
    end

end
subplot(2,2,2);bar(myhist); title('Histogram of original
image'); sum=0;
%Cumulative values
for i=0:255

    sum=sum+myhist(i+1);
   sum_of_hist(i+1)=sum;
end
area=rows*cols;
%Dm=input('Enter no. of gray levels in output image:
'); Dm=256;
for i=1:rows
   for j=1:cols
      n=double(data(i,j));
       data(i,j)=sum_of_hist(n+1)*Dm/area;
    end
 end
%Calculation of histogram for equalised image
for i=1:rows
   for j=1:cols
      m=double(data(i,j));
         myhist(m+1)=myhist(m+1)+1;
      end

end

subplot(2,2,3);bar(myhist);title('Equalised Histogram');

subplot(2,2,4);imshow(data); title('Image after histogram equalisation');
```

Program output:



Sample Program in C language:

Requirements:

Add member function histogram() in Image class for histogram calculation and equalisation

Add member function print_histogram() in Image class to display histogram of an image.

Implement this member function in implementation program You can use 10 10 sample image to observe results

```
void image::histogram(char filename[])
 {
 int i,j,m;
 FILE *f1;
 int sum_of_hist[100];
 f1=fopen(filename,"rb");
 int k=rows*cols; for(i=0;
 i<gray_levels;i++)
 {
 hist[i]=0;
 }
 fread(data,sizeof(int),k,f1);
 for(i=1; i<rows;i++)
 {
        for(j=1; j<cols; j++)
        {
        m=data[i][j];
        hist[m]=hist[m]+1;
        }
 }
 fclose(f1);
 print_histogram();
```

```
 getch();
  /* Histogram Equalisation*/
int sum=0;
int Dm;
for(i=0; i<gray_levels; i++)
{
        sum=sum+hist[i];
        sum_of_hist[i]=sum;
}
int area=k;
printf("Enter no. of gray levels in output
image:"); scanf("%d", &Dm);

for(i=0;i<rows;i++)
{
        for(j=0;j<cols;j++)
        {
                int n=data[i][j];
                data[i][j]=sum_of_hist[n]*Dm/area;
        }
}

for(i=0;i<rows;i++)
{
        for(j=0;j<cols;j++)
        {
                int n=data[i][j];
                hist[n]=hist[n]+1;
        }
}
printf("Equalised histogram:\n");
print_histogram();
getch();
}

 void image::print_histogram()
 {
 int i,j;
 printf("\n\n");
 printf("  Occurence of gray levels ---->\n");
 printf("        ");
 for(i=0;i<=25;i++)printf("_");
 printf("\n");
 for(i=0; i<16; i++)
 {rintf("%2d |",i);
 for(j=1; j<=hist[i];j++)
 printf("*"); printf("\n");
 }
 printf("\n\n");
 }
```

EXPERIMENT NO. 6

AIM: To write and execute program for geometric transformation of image

Introduction: We will perform following geometric transformations on the image in this experiment

Translation: Translation is movement of image to new position. Mathematically translation is represented as:

$$x' = x + x \text{ and } y' = y + y$$

In matrix form translation is represented by:

$$[x' \; y' \; 1] = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} x \\ y \\ 1 \end{matrix}$$

Scaling: Scaling means enlarging or shrinking. Mathematically scaling can be represented by:

$$x' = x \times S_X \text{ and } y' = y \times S_Y$$

In matrix form scaling is represented by:

$$[x' \; y' \; 1] = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} x \\ y \\ 1 \end{matrix}$$

Rotation: Image can be rotated by an angle , in matrix form it can be represented as:

$$x' = x\cos - y\sin \text{ and } y' = x\sin + y\cos$$

In matrix form rotation is represented by:

$$[x' \; y' \; 1] = \begin{bmatrix} \cos & \sin & 0 \\ \sin & \cos & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{matrix} x \\ y \\ 1 \end{matrix}$$

If is substituted with - , this matrix rotates the image in clockwise direction.

Shearing: Image can be distorted (sheared) either in x direction or y direction. Shearing can be represented as:

$$x' = sh_X \times y \quad y' = y$$

In matrix form rotation is represented by:

Shearing in Y direction can be given by:

$$x' = x \quad y' = y \times shy$$

$$Y_{shear} = \begin{matrix} 1 & sh_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \begin{matrix} x \\ y \\ 1 \end{matrix}$$

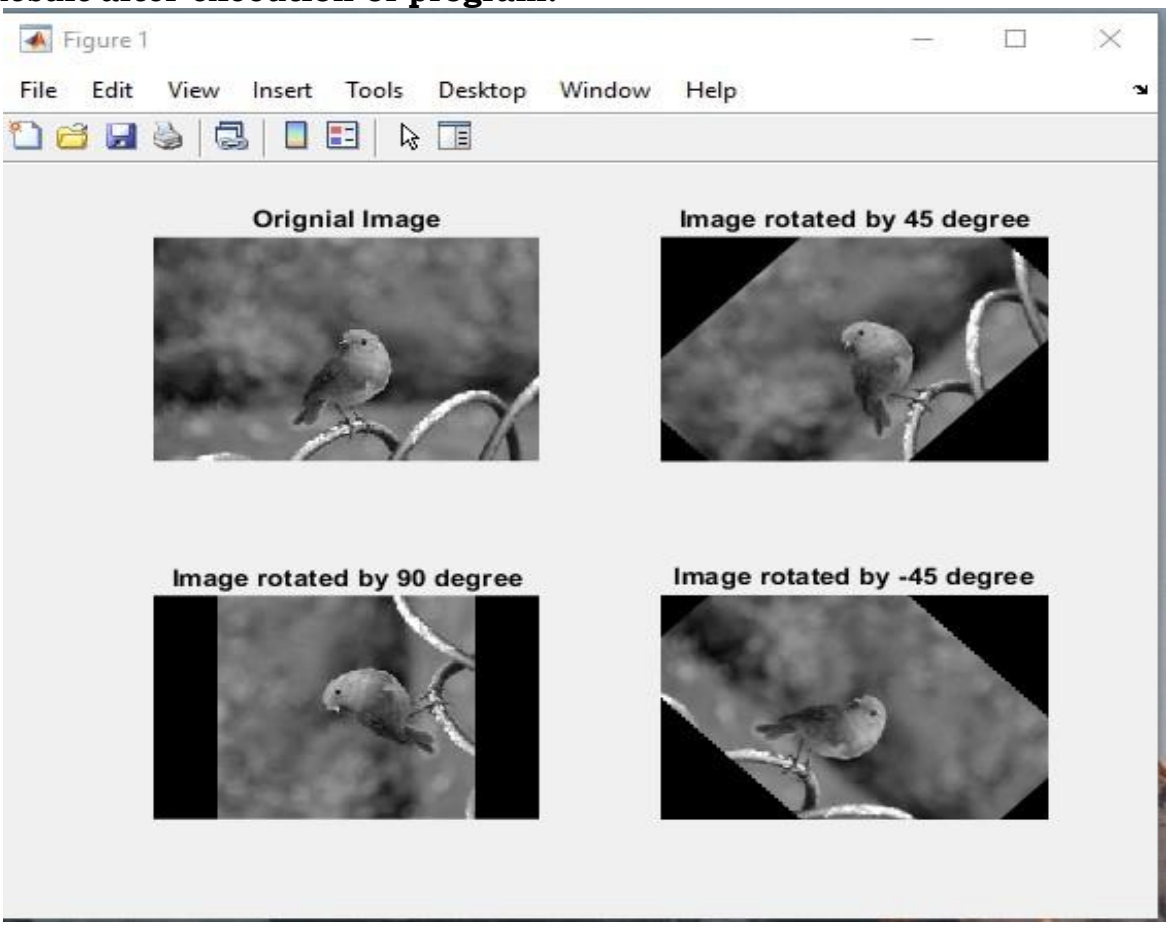
Zooming : zooming of image can be done by process called pixel replication or interpolation. Linear interpolation or some non-linear interpolation like cubic interpolation can be performed for better result.

Program:

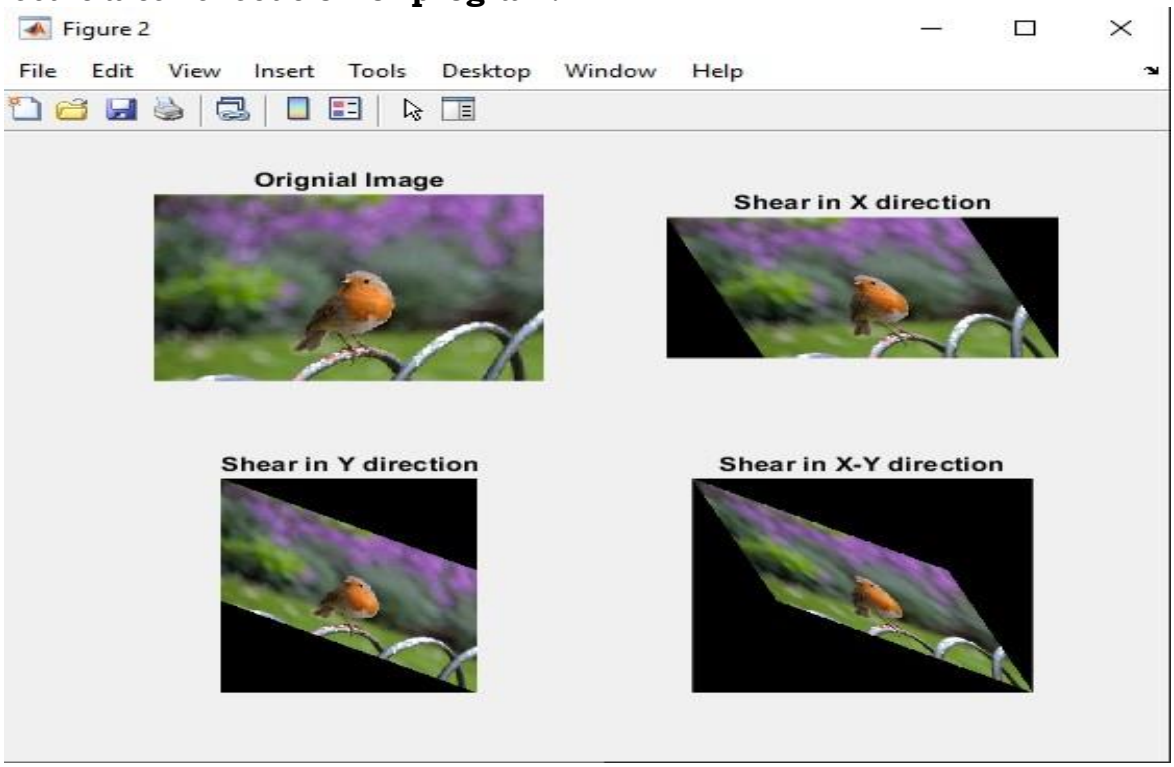
```
%Experiment No. 4 Image Geometric transformations
%To rotate given image using standard Matlab function
imrotate() %To transform image using standard Matlab function
imtransform() % using shearing matrix.
clc close all
filename=input('Enter File Name:','s');
x=imread(filename);
x=rgb2gray(x);
subplot(2,2,1); imshow(x); title('Orignial Image');
y=imrotate(x,45,'bilinear','crop');
subplot(2,2,2); imshow(y); title('Image rotated by 45
degree'); y=imrotate(x,90,'bilinear','crop');
subplot(2,2,3); imshow(y); title('Image rotated by 90
degree'); y=imrotate(x,-45,'bilinear','crop');
subplot(2,2,4); imshow(y); title('Image rotated by -45
degree'); x = imread('cameraman.tif');
 tform = maketform('affine',[1 0 0; .5 1 0; 0 0
 1]); y = imtransform(x,tform);
 figure;
 subplot(2,2,1); imshow(x); title('Orignial Image');
 subplot(2,2,2); imshow(y); title('Shear in X direction');
 tform = maketform('affine',[1 0.5 0; 0 1 0; 0 0 1]); y =
 imtransform(x,tform);
 subplot(2,2,3); imshow(y); title('Shear in Y direction');
 tform = maketform('affine',[1 0.5 0; 0.5 1 0; 0 0 1]);
```

```
y = imtransform(x,tform);
 subplot(2,2,4); imshow(y); title('Shear in X-Y direction');
```

Result after execution of program:



Result after execution of program:



EXPERIMENT NO. 7

AIM: To understand various image noise models and to write programs for image restoration

Introduction:

Image restoration is the process of removing or minimizing known degradations in the given image. No imaging system gives perfect quality of recorded images due to various reasons.

Image restoration is used to improve quality of image by various methods in which it will try to decrease degradations & noise.

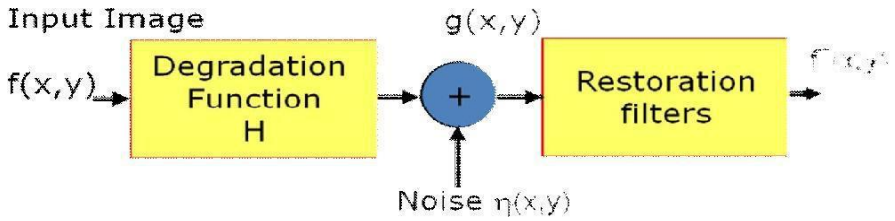
Degradation of images can occur due to many reasons. Some of them are as under:

- Poor Image sensors
- Defects of optical lenses in camera Non-linearity of the electro-optical sensor;
- Graininess of the film material which is utilised to store the image
- Relative motion between an object and camera
- Wrong focus of camera
- Atmospheric turbulence in remote sensing or astronomy
- Degradation due to temperature sensitive sensors like CCD
- Poor light levels

Degradation of images causes:

- Radiometric degradations;
- Geometric distortions;
- Spatial degradations.

Model of Image Degradation/restoration Process is shown in the following figure.



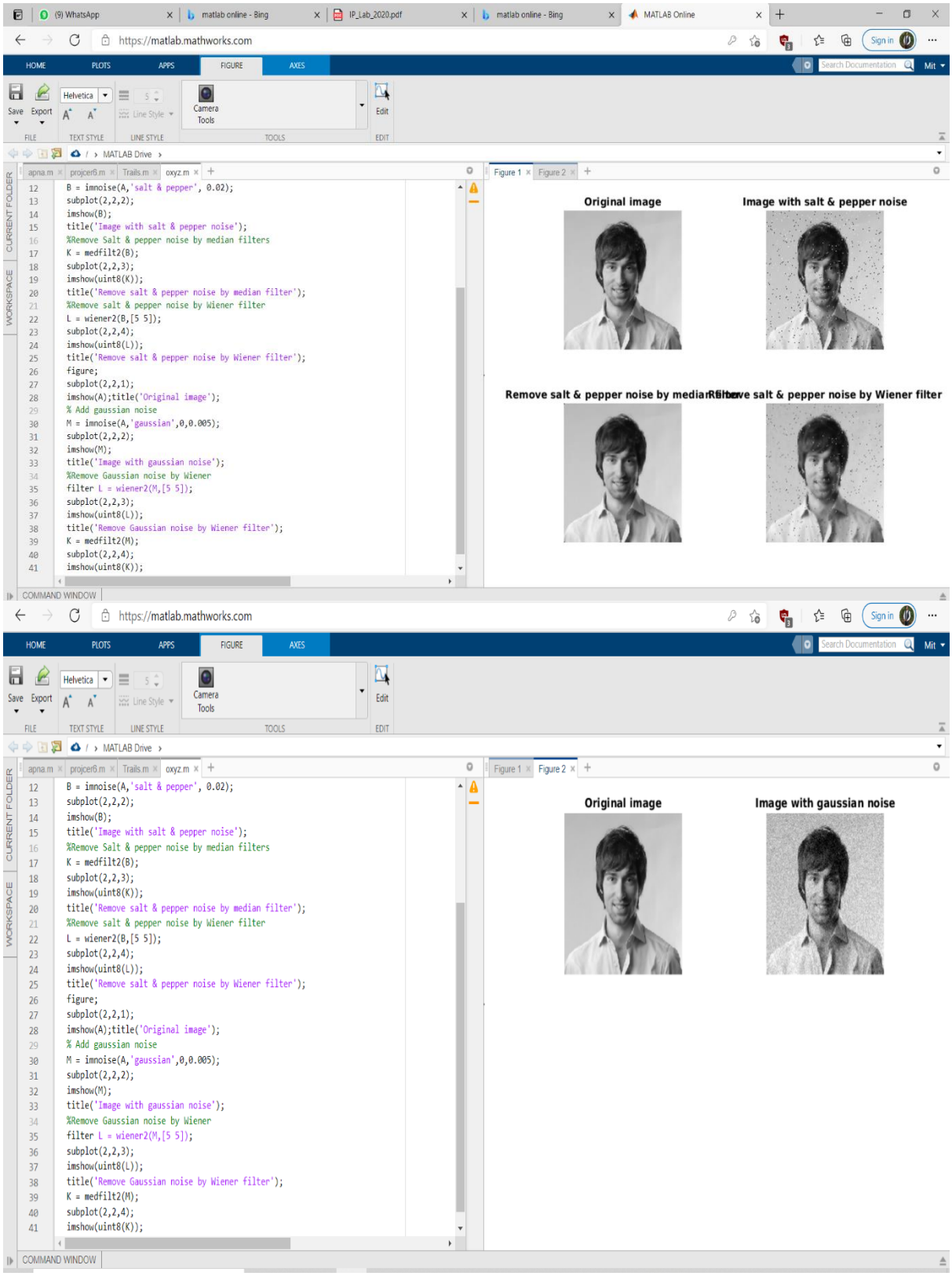
Spatial domain: $g(x,y)=h(x,y)*f(x,y) + (x,y)$

Frequency domain: $G(u,v)=H(u,v)F(u,v)+N(u,v)$

G(x,y) is spatial domain representation of degraded image and G(u,v) is frequency domain representation of degraded image. Image restoration applies different restoration filters to reconstruct image to remove or minimize degradations.

Program:

```
%Experiment No. 7
%To add noise in the image and apply image restoration
%technique using Wiener filter and median filter
clear all;
close all;
[filename,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif','IMAGE
Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'},'Chose Image File'); A=
imread(cat(2,pathname,filename));
if(size(A,3)==3)
A=rgb2gray(A);
end
subplot(2,2,1); imshow(A);title('Original
image'); % Add salt & pepper noise
B = imnoise(A,'salt & pepper', 0.02);
subplot(2,2,2); imshow(B);title('Image with salt & pepper noise');
%Remove Salt & pepper noise by median filters
K = medfilt2(B);
subplot(2,2,3); imshow(uint8(K)); title('Remove salt & pepper noise by median filter');

%Remove salt & pepper noise by Wiener filter
L = wiener2(B,[5 5]);
subplot(2,2,4); imshow(uint8(L)); title('Remove salt & pepper noise by Wiener filter');
figure;
subplot(2,2,1); imshow(A);title('Original
image'); % Add gaussian noise
M = imnoise(A,'gaussian',0,0.005);
subplot(2,2,2); imshow(M); title('Image with gaussian noise');

%Remove Gaussian noise by Wiener
filter L = wiener2(M,[5 5]);
subplot(2,2,3); imshow(uint8(L));title('Remove Gaussian noise by
Wiener filter'); K = medfilt2(M);
subplot(2,2,4); imshow(uint8(K)); title('Remove Gaussian noise by median filter');
```


https://matlab.mathworks.com
HOME
PLOTS
APPS
FIGURE
AXES
MATLAB Drive
apna.m
projcer6.m
Trails.m
oxyz.m
Figure 1
Figure 2
Original image
Image with salt & pepper noise
Remove salt & pepper noise by median filter
Remove salt & pepper noise by Wiener filter
Image with gaussian noise
COMMAND WINDOW

EXPERIMENT NO. 8

AIM: Write and execute programs to remove noise using spatial filters
Understand 1-D and 2-D convolution process
Use 3x3 Mask for low pass filter and high pass filter

Introduction:

Spatial Filtering is sometimes also known as neighborhood processing. Neighborhood processing is an appropriate name because you define a center point and perform an operation (or apply a filter) to only those pixels in predetermined neighborhood of that center point. The result of the operation is one value, which becomes the value at the center point's location in the modified image. Each point in the image is processed with its neighbors. The general idea is shown below as a "sliding filter" that moves throughout the image to calculate the value at the center location.

In spatial filtering, we perform convolution of data with filter coefficients. In image processing, we perform convolution of 3x3 filter coefficients with 2-D image data. In signal processing, we perform convolution of 1-D data with set of filter coefficients.

Program for 1D convolution (Useful for 1-D Signal Processing):

```
clear;
clc;
x=input("Enter value of x: ")
y=input("Enter value of y: ")

n=length(x)
k=length(y)

for z=n+1:n+k-1
   x(z)=0;
end
for u=k+1:n+k-1
   y(u)=0;
end

for i=1:n+k-1
   s(i)=0
   for j=1:i s(i)=(x(j)*y(i-
      j+1))+s(i)
   end
end
subplot(3,1,1)
plot2d3(x)
```

```
subplot(3,1,2)
plot2d3(y)
subplot(3,1,3)
plot2d3(s)
```

Take value of x: [1 2 3 4 5 6 7 8 9 0 10 11 12 13 14 15]
Y: [-1 1]

Program for 2D convolution:

```
clear;
clc;
x=input("Enter value of x in matrix form: ")
y=input("Enter value of y in matrix form: ")
[xrows,xcols]=size(x);
[yrows,ycols]=size(y);
result=zeros(xrows+yrows,xcols+ycols)
for r = 1:xrows-1
  for c = 1:xcols-1
                sum=0;
                for a=0:yrows
                    for b=0:ycols
                        sum=sum+x(r+a,c+b)*y(a+1,b+1);
                    end
                end
result(r,c)=sum;

  end

end
```

Enter following 2D matrix for x:

```
10  10  10  10  10  10 10 10
10  10  10  10  10  10 10 10
10  10  10  10  10  10 10 10
10  10  10  10  10  10 10 10
10  10  10  10  10  10 10 10
10  10  10  10  10  10 10 10
10  10  10  10  10  10 10 10
10  10  10  10  10  10 10 10
```

Enter following 2D matrix for y:

```
1  1  1
 0   0  0
 1   1  1
```

Program:

```
%Experiment No. 8 Spatial filtering using standard MATLAB function
%To apply spatial filters on given image
clc; close
all; clear
all;
%Define spatial filter masks
L1=[1 1 1;1 1 1;1 1 1]; L2=[0
1 0;1 2 1;0 1 0]; L3=[1 2 1;2
4 2;1 2 1]; H1=[-1 -1 -1;-1
9 -1;-1 -1 -1]; H2=[0 -1 0;-1
5 - 1;-0 -1 0]; H3=[1 -2 1;-2
5 - 2;1 -2 1];

% Read the test image and display it
[filename,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif','IMAGE
Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'},'Chose Image File');
myimage = imread(cat(2,pathname,filename));
if(size(myimage,3)==3)
myimage=rgb2gray(myimage);
end
subplot(3,2,1);
imshow(myimage); title('Original Image');

L1 = L1/sum(L1);
filt_image=
conv2(double(myimage),double(L1));
subplot(3,2,2);
imshow(filt_image,[]); title('Filtered
image with mask L1');

L2 = L2/sum(L2);
filt_image=
conv2(double(myimage),double(L2));
subplot(3,2,3);
imshow(filt_image,[]); title('Filtered
image with mask L2');

L3 = L3/sum(L3);
filt_image=
conv2(double(myimage),double(L3));
subplot(3,2,4);
imshow(filt_image,[]); title('Filtered
image with mask L3');

filt_image= conv2(double(myimage),H1);
subplot(3,2,5);
imshow(filt_image,[]); title('Filtered
image with mask H1');
```

```
filt_image= conv2(double(myimage),H2);
subplot(3,2,6);
imshow(filt_image,[]); title('Filtered
image with mask H1');

figure;
subplot(2,2,1);
imshow(myimage); title('Original Image');
%The command fspecial() is used to create mask
%The command imfilter() is used to apply the gaussian filter mask to the image
%Create a Gaussian low pass filter of size 3

gaussmask = fspecial('gaussian',3);
filtimg = imfilter(myimage,gaussmask);
subplot(2,2,2);
imshow(filtimg,[]),title('Output of Gaussian filter 3 X 3');

%Generate a lowpass filter of size 7 X 7
%The command conv2 is used the apply the filter
%This is another way of using the filter

avgfilt = [ 1 1 1 1 1 1 1;
        1 1 1 1 1 1 1; 1
        1 1 1 1 1 1; 1 1
        1 1 1 1 1; 1 1 1
        1 1 1 1; 1 1 1 1
        1 1 1; 1 1 1 1 1
        1 1];

avgfiltmask = avgfilt/sum(avgfilt);
convimage= conv2(double(myimage),double(avgfiltmask));

subplot(2,2,3);
imshow(convimage,[]);
title('Average filter with conv2()');

filt_image= conv2(double(myimage),H3);
subplot(3,2,6);
imshow(filt_image,[]); title('Filtered
image with mask H3');
```

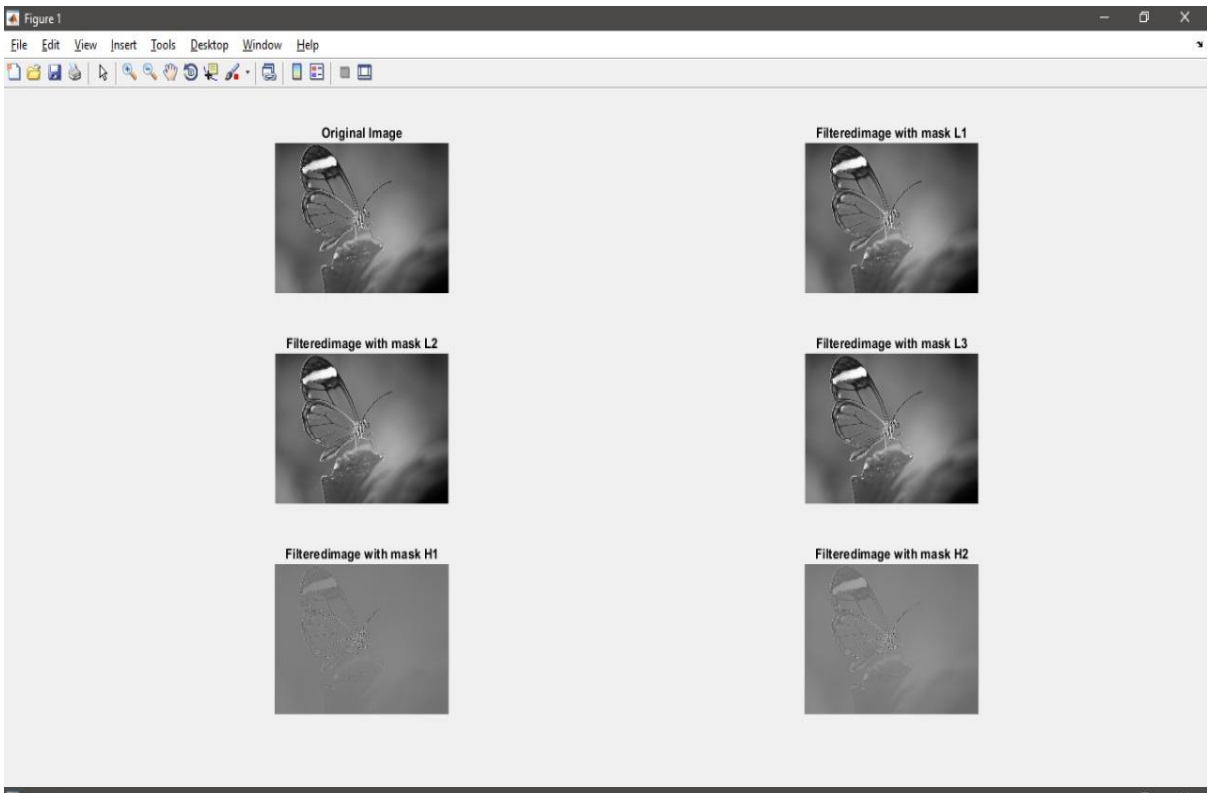


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
Filteredimage with mask L1
Filteredimage with mask L2
Filteredimage with mask L3
Filteredimage with mask H1
Filteredimage with mask H2

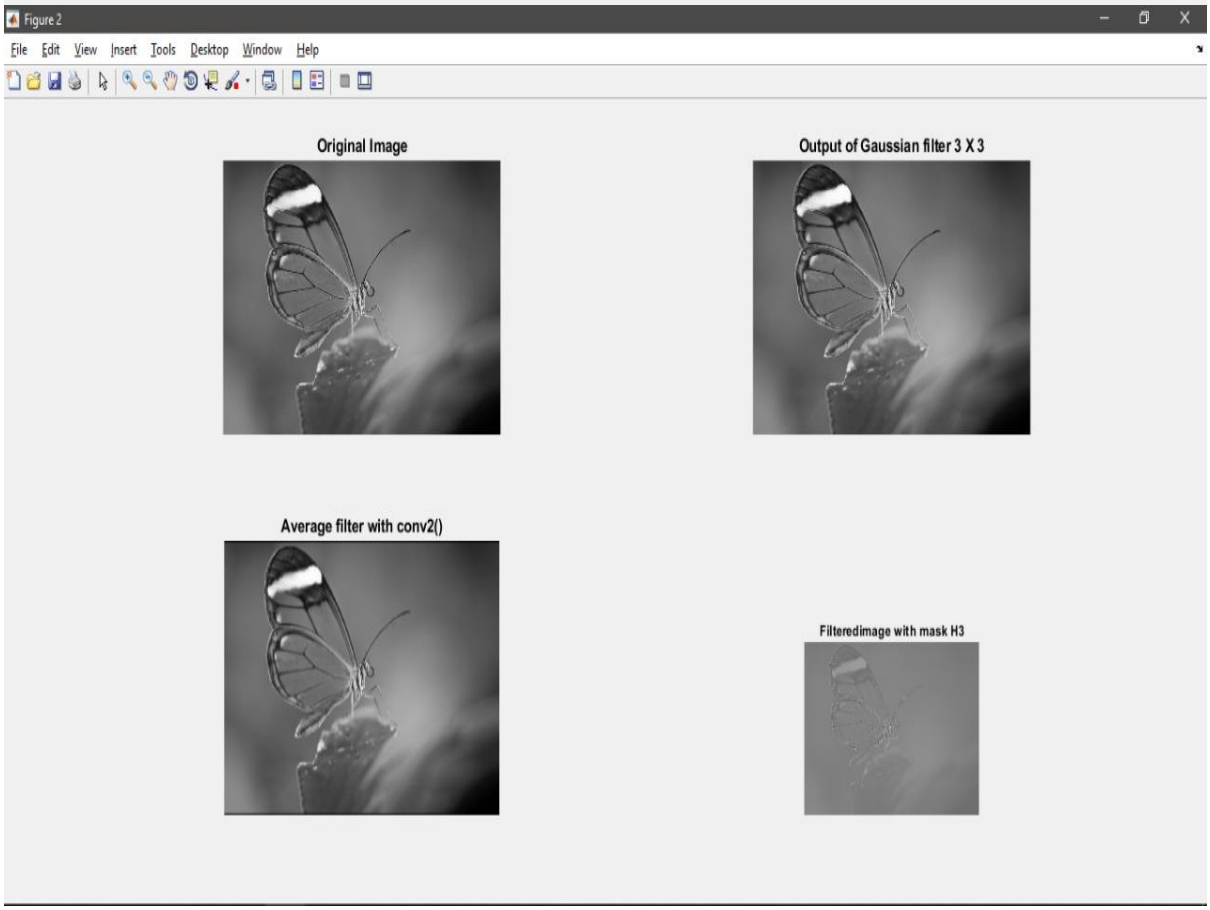


Figure 2
File Edit View Insert Tools Desktop Window Help
Original Image
Output of Gaussian filter 3 X 3
Average filter with conv2()
Filteredimage with mask H3

EXPERIMENT NO. 9

AIM: Write and execute programs for image frequency domain filtering

Introduction:
In spatial domain, we perform convolution of filter mask with image data. In frequency domain we perform multiplication of Fourier transform of image data with filter transfer function.

Fourier transform of image f(x,y) of size MxN can be given by:

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

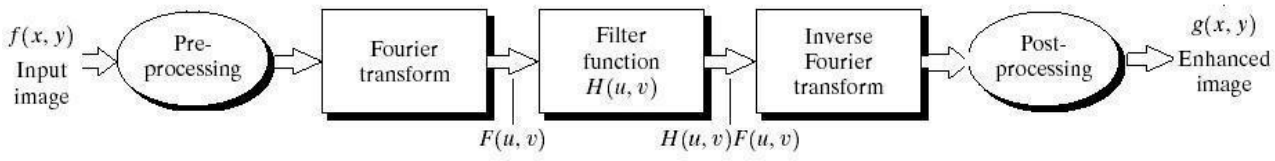
Where, u = 0,1,2 ……. M-1 and v = 0,1,2……N-1

Inverse Fourier transform is given by:

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

Where, x = 0,1,2 ……. M-1 and y = 0,1,2……N-1

Basic steps for filtering in frequency domain:



Pre-processing: Multiply input image f(x,y) by $(-1)^{x+y}$ to center the transform
Computer Discrete Fourier Transform F(u,v) of input image f(x,y)
Multiply F(u,v) by filter function H(u,v) Result: H(u,v)F(u,v)
Computer inverse DFT of the
result Obtain real part of the result Post-

Processing: Multiply the result by $(-1)^{x+y}$

Program:

```
%Experiment 9 Program for frequency domain filtering
clc;
 close all;
 clear all;
%Read the image, resize it to 256 x 256
%Convert it to grey image and display it
[filename,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif','IMAGE
Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'},'Chose Image File');
myimg=imread(cat(2,pathname,filename));
if(size(myimg,3)==3)
myimg=rgb2gray(myimg);
end
myimg = imresize(myimg,[256
256]); myimg=double(myimg);
subplot(2,2,1);
imshow(myimg,[]),title('Original Image');
[M,N] = size(myimg); % Find size
%Preprocessing of the image
 for x=1:M
    for y=1:N
       myimg1(x,y)=myimg(x,y)*((-
    1)^(x+y)); end
end

%Find FFT of the image myfftimage =
fft2(myimg1); subplot(2,2,2);
imshow(myfftimage,[]); title('FFT
Image');


%Define cut off frequency
low = 30;
band1 = 20;
band2 = 50;
%Define Filter Mask
mylowpassmask = ones(M,N);
mybandpassmask = ones(M,N);
%Generate values for ifilter pass mask
for u = 1:M
  for v = 1:N
      tmp = ((u-(M/2))^2 +(v-
      (N/2))^2)^0.5; if tmp > low
        mylowpassmask(u,v)
      = 0; end
      if tmp > band2 || tmp < band1;
        mybandpassmask(u,v) = 0;
```

```
    end
  end
end
%Apply the filter H to the FFT of the Image
resimage1  =  myfftimage.*mylowpassmask;
resimage3 = myfftimage.*mybandpassmask;

%Apply the Inverse FFT to the filtered image
%Display the low pass filtered image
r1 = abs(ifft2(resimage1));
subplot(2,2,3);
imshow(r1,[]),title('Low Pass filtered image');

%Display the band pass filtered image r3
= abs(ifft2(resimage3)); subplot(2,2,4);
imshow(r3,[]),title('Band Pass filtered
image'); figure;
subplot(2,1,1);imshow(mylowpassmask);
subplot(2,1,2);imshow(mybandpassmask);
```

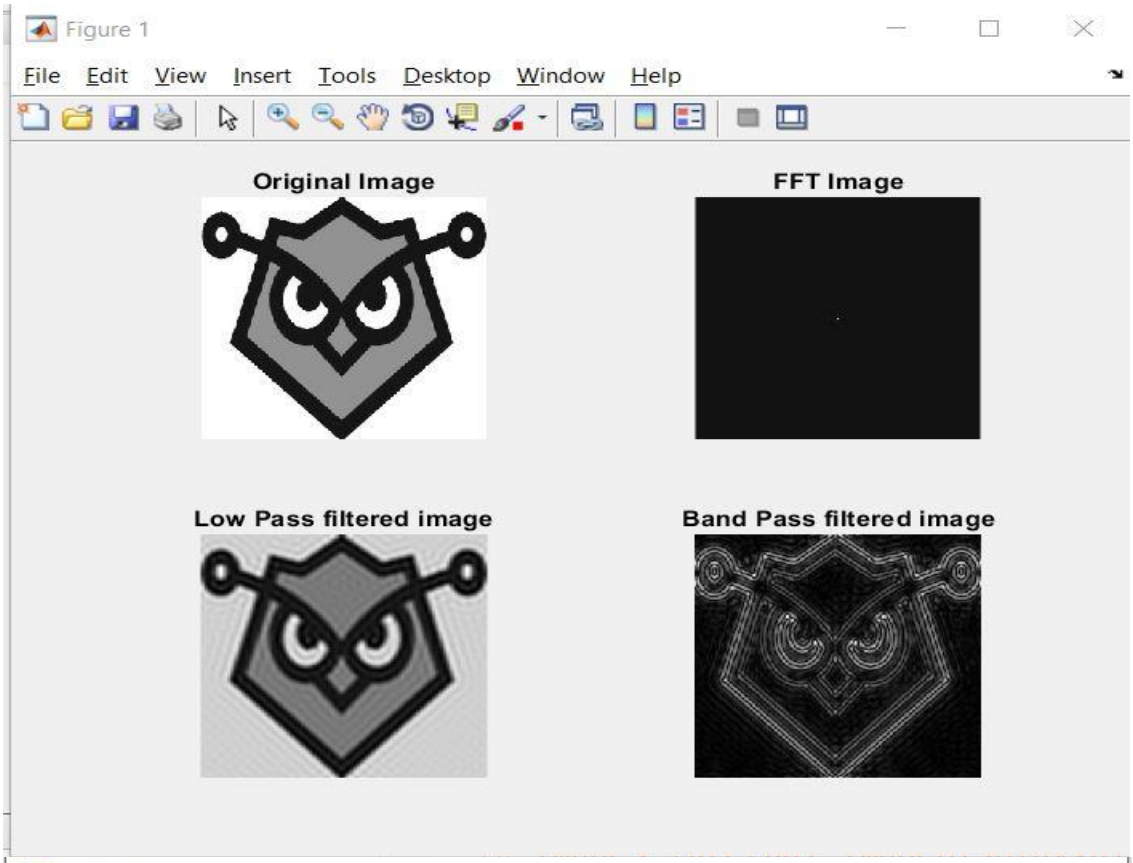


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
FFT Image
Low Pass filtered image
Band Pass filtered image

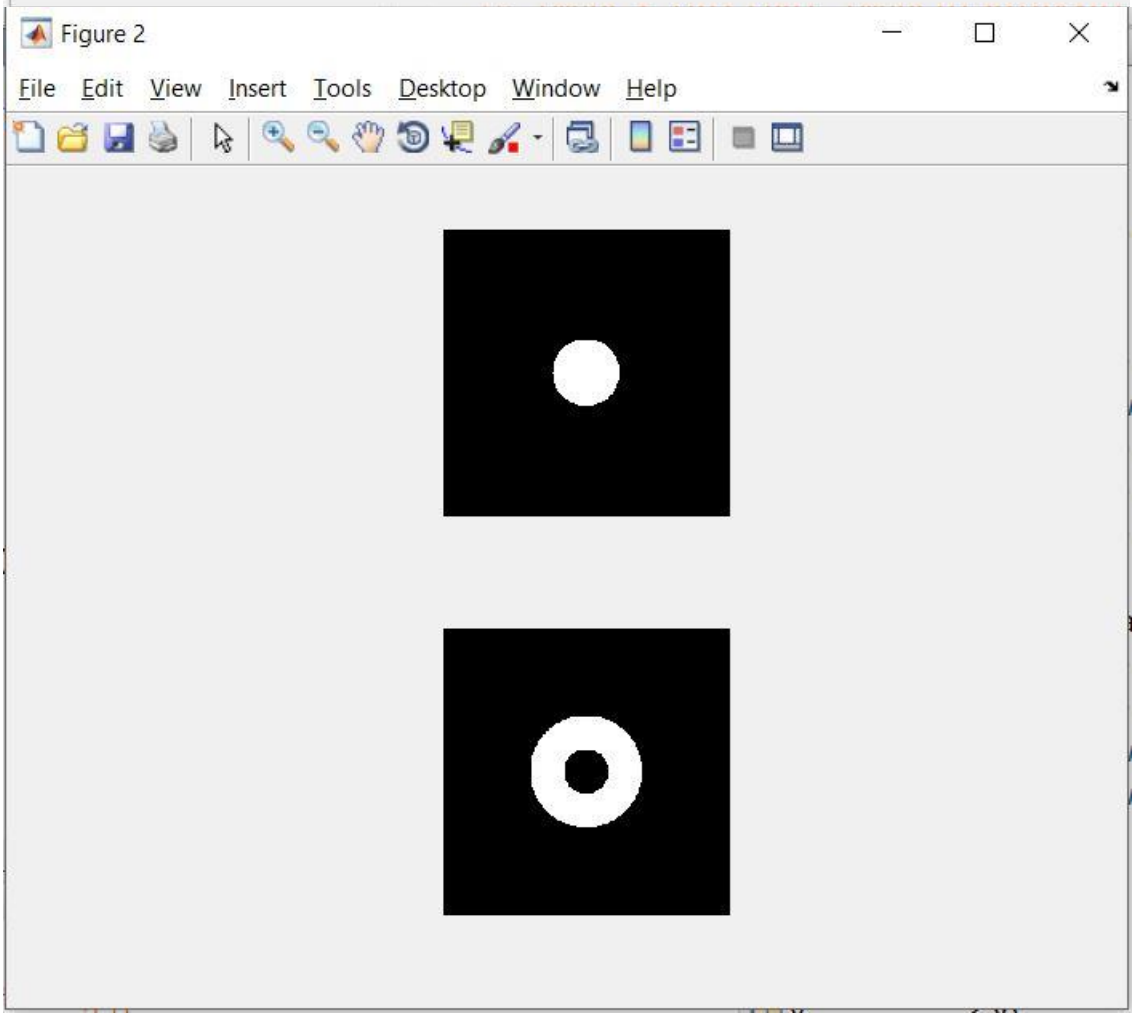


Figure 2
File Edit View Insert Tools Desktop Window Help

EXPERIMENT NO. 10

AIM: Write a program in MATLAB for edge detection using different edge detection mask.

Introduction:

Image segmentation is to subdivide an image into its component regions or objects. Segmentation should stop when the objects of interest in an application have been isolated. Basic purpose of segmentation is to partition an image into meaningful regions for particular application The segmentation is based on measurements taken from the image and might be grey level, colour, texture, depth or motion.

There are basically two types of image segmentation approaches:

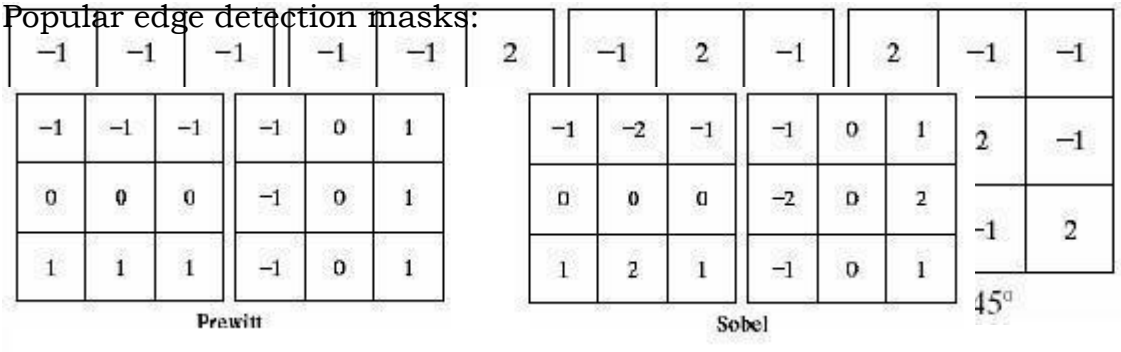
[1] Discontinuity based: Identification of isolated points, lines or edges
[2] Similarity based: Group pixels which has similar characteristics by thresholding, region growing, region splitting and merging

Edge detection is discontinuity based image segmentation approach. Edges play a very important role in many image processing applications. It provides outline of an object. In physical plane, edges are corresponding to changes in material properties, intensity variations, discontinuity in depth. Pixels on the edges are called edge points. Edge detection techniques basically try to find out grey level transitions.

Edge detection can be done by first order derivative and second order derivative operators.

First order line detection 3x3 mask are:

Popular edge detection masks:



- Sobel operator performs well for image with noise compared to Prewitt operator because Sobel operator performs averaging along with edge detection.

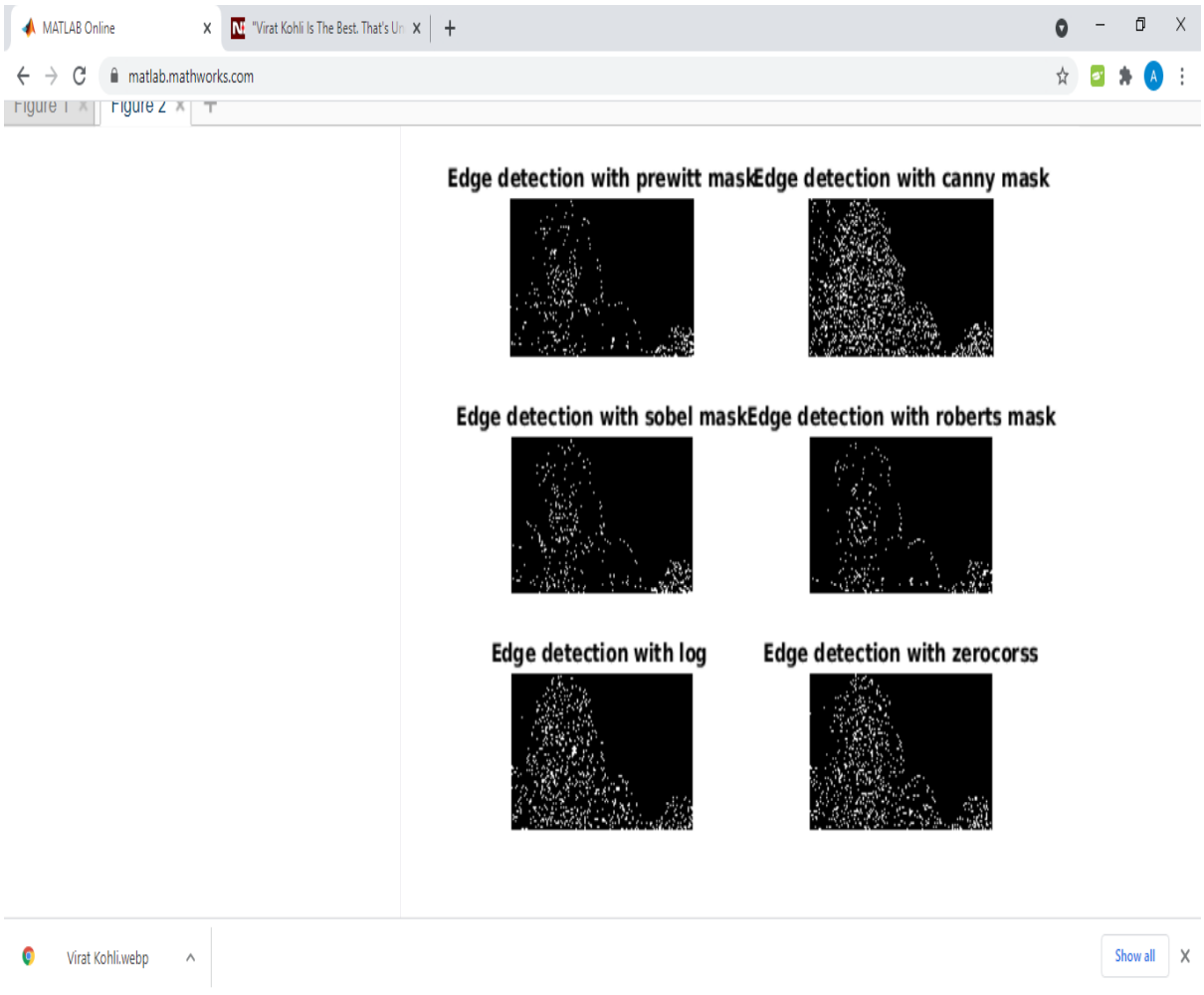
- Because Sobel operator gives smoothing effect, spurious edges will not be detected by it.

Second derivative operators are sensitive to the noise present in the image so it is not directly used to detect edge but it can be used to extract secondary information like …

- Used to find whether point is on darker side or white side depending on sign of the result
- Zero crossing can be used to identify exact location of edge whenever there is gradual transitions in the image

MATLAB Code using standard function:

```
%Experiment 10
%Program for edge detection using standard masks
 clear all;
[filename,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif','IMAGE
Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'},'Choose Image');
A=imread(filename);


if(size(A,3)==
3)
A=rgb2gray(A
); end
imshow(A);
figure;
BW = edge(A,'prewitt');
subplot(3,2,1); imshow(BW);title('Edge detection with prewitt mask');
BW = edge(A,'canny');
subplot(3,2,2); imshow(BW);;title('Edge detection with canny mask');
BW = edge(A,'sobel');
subplot(3,2,3); imshow(BW);;title('Edge detection with sobel mask');
BW = edge(A,'roberts');
subplot(3,2,4); imshow(BW);;title('Edge detection with roberts mask');
BW = edge(A,'log');
subplot(3,2,5); imshow(BW);;title('Edge detection with
log '); BW = edge(A,'zerocross');
subplot(3,2,6); imshow(BW);;title('Edge detection with zerocorss');
```


MATLAB Online
"Virat Kohli Is The Best. That's Un
matlab.mathworks.com
Figure 1
Figure 2
Edge detection with prewitt mask
Edge detection with canny mask
Edge detection with sobel mask
Edge detection with roberts mask
Edge detection with log
Edge detection with zerocorss
Virat Kohli.webp
Show all

MATLAB Code for edge detection using convolution in spatial domain

```
%Experiment 10
%Program to demonstrate various point and edge detection mask
clear all;
clc;
while 1
K= menu('Choose mask','Select Image File','Point Detect','Horizontal line
detect','Vertical line detect','+45 Detect','-45 Detect','ractangle Detect','exit')
M=[-1 0 -1; 0 4 0; -1 0 -1;] % Default mask
switch K
   case 1,
   [namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif','IMAGE Files
(*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'},'Chose
   GrayScale Image');
   data=imread(strcat(pathname,namefile));
   %data=rgb2gray(data);
   imshow(data);
   case 2,
      M=[-1 -1 -1;-1 8 -1;-1 -1 -1]; % Mask for
   point detection case 3,
      M=[-1 -1 -1; 2 2 2; -1 -1 -1]; % Mask for
   horizontal edges case 4,
      M=[-1 2 -1; -1 2 -1; -1 2 -1]; % Mask for
   vertical edges case 5,
      M=[-1 -1 2; -1 2 -1; 2 -1 -1]; % Mask for 45 degree diagonal
   line case 6,
      M=[2 -1 -1;-1 2 -1; -1 -1 2]; % Mask for -45 degree diagonal
   line case 7,
      M=[-1 -1 -1;-1 8 -1;-1 -1
   - 1]; case 8,
        break;
   otherwise,
      msgbox('Select proper mask');
end

outimage=conv2(double(data),double(M));
figure;
imshow(outimage);
end
close all %Write an image to a file
imwrite(mat2gray(outimage),'outimage.jpg','quality',99);
```

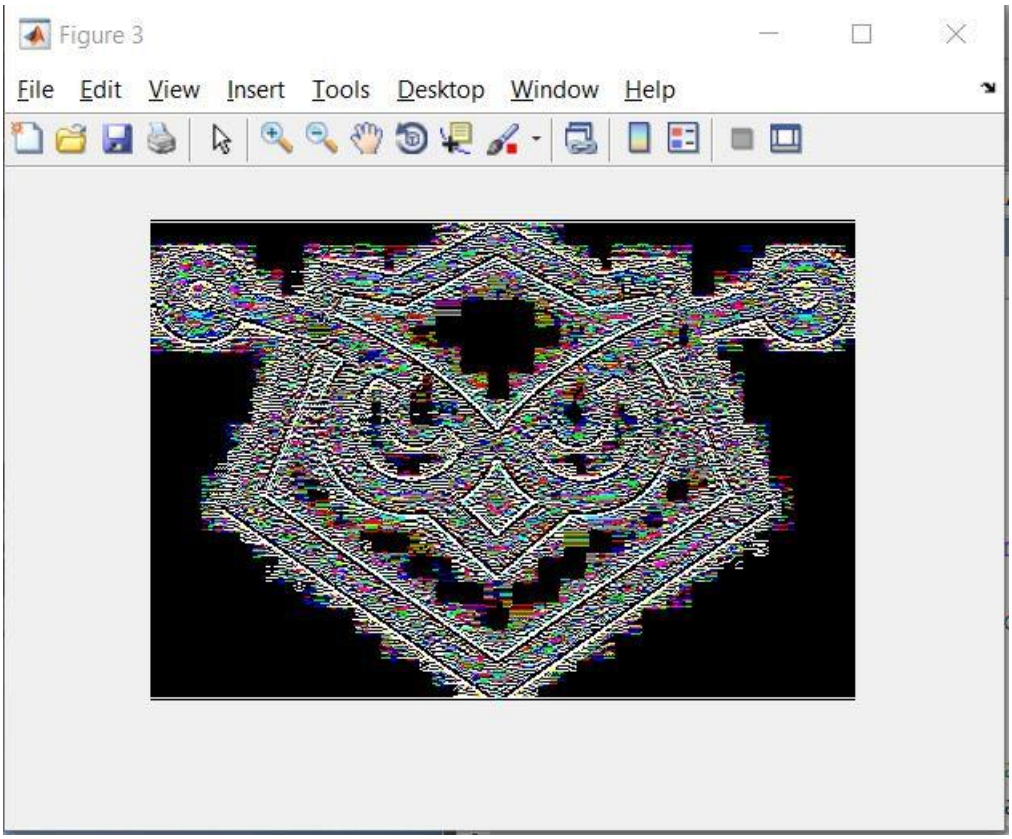


Figure 3
File Edit View Insert Tools Desktop Window Help

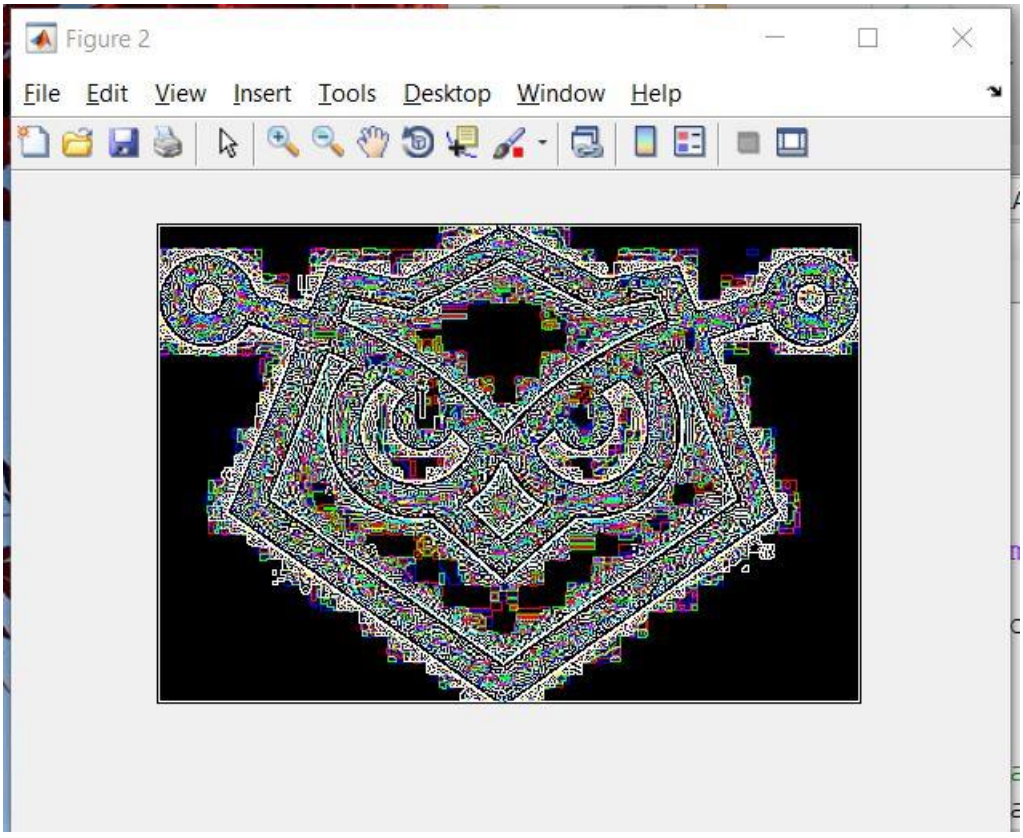


Figure 2
File Edit View Insert Tools Desktop Window Help

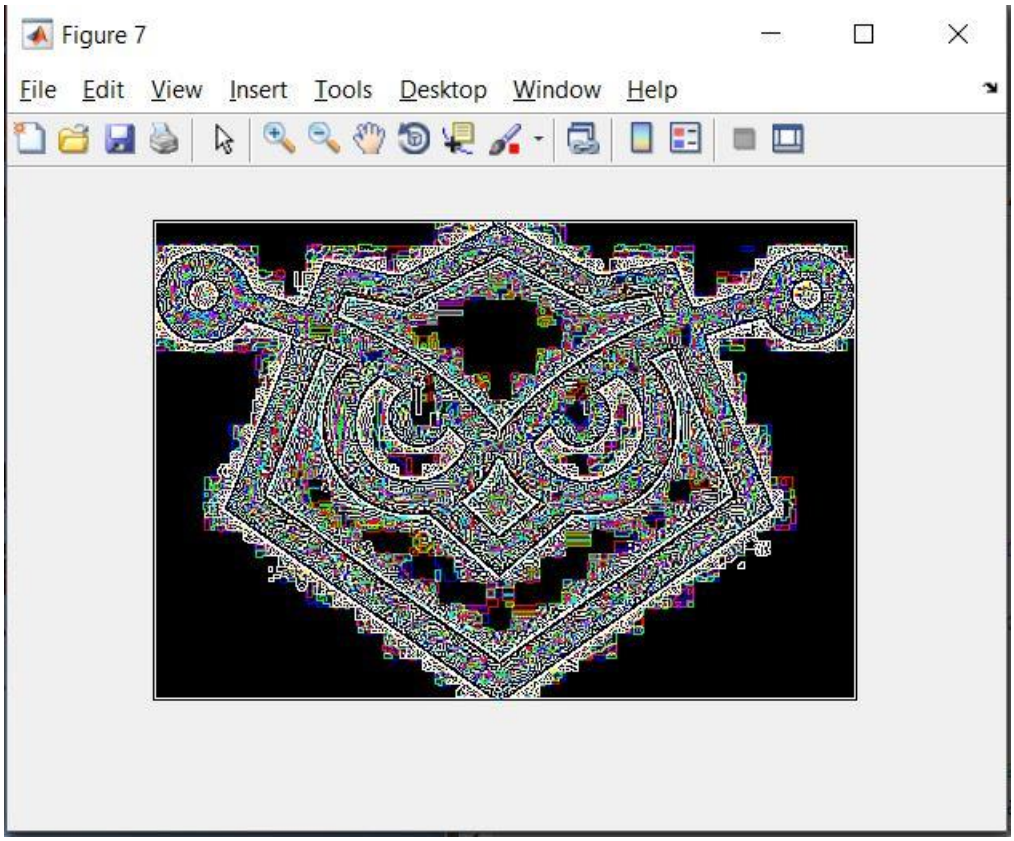


Figure 7
File Edit View Insert Tools Desktop Window Help

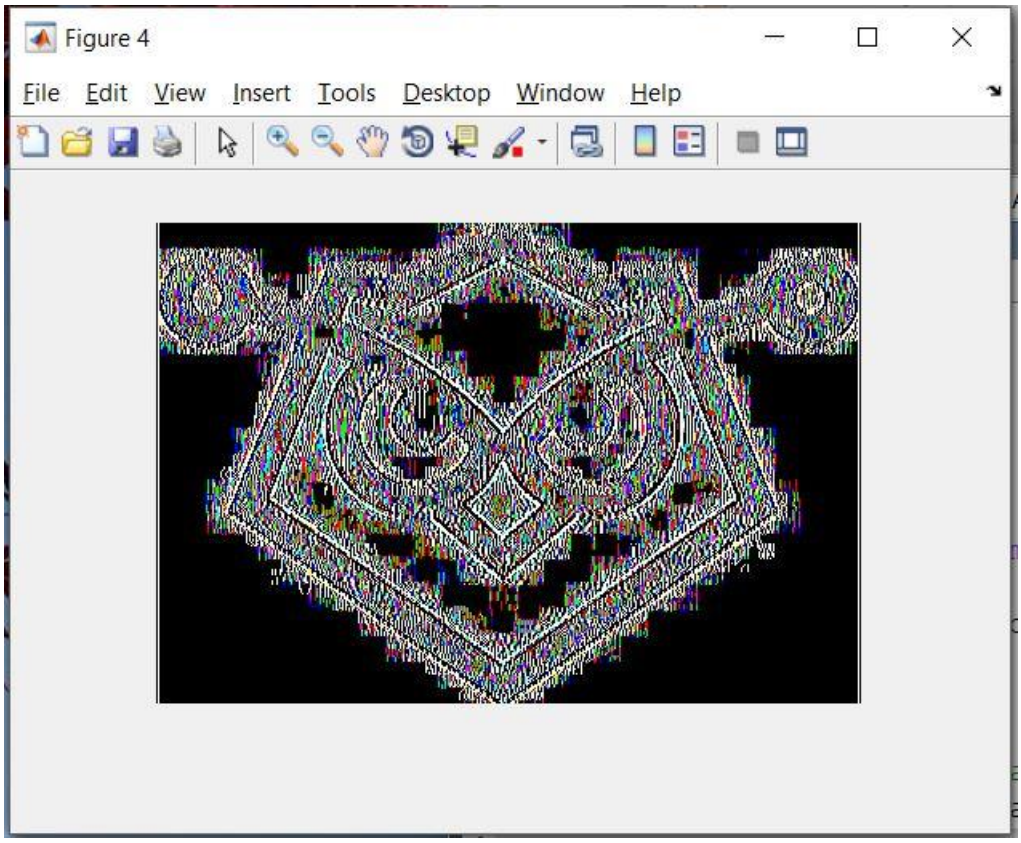


Figure 4
File Edit View Insert Tools Desktop Window Help

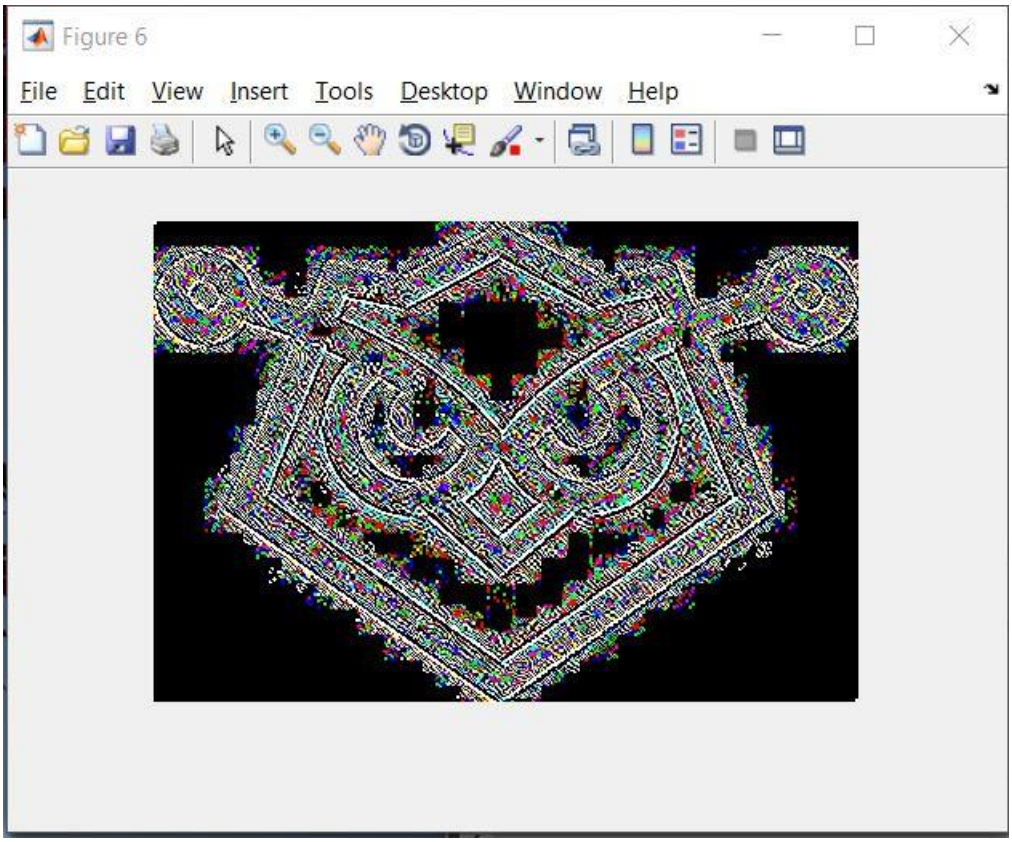


Figure 6
File Edit View Insert Tools Desktop Window Help

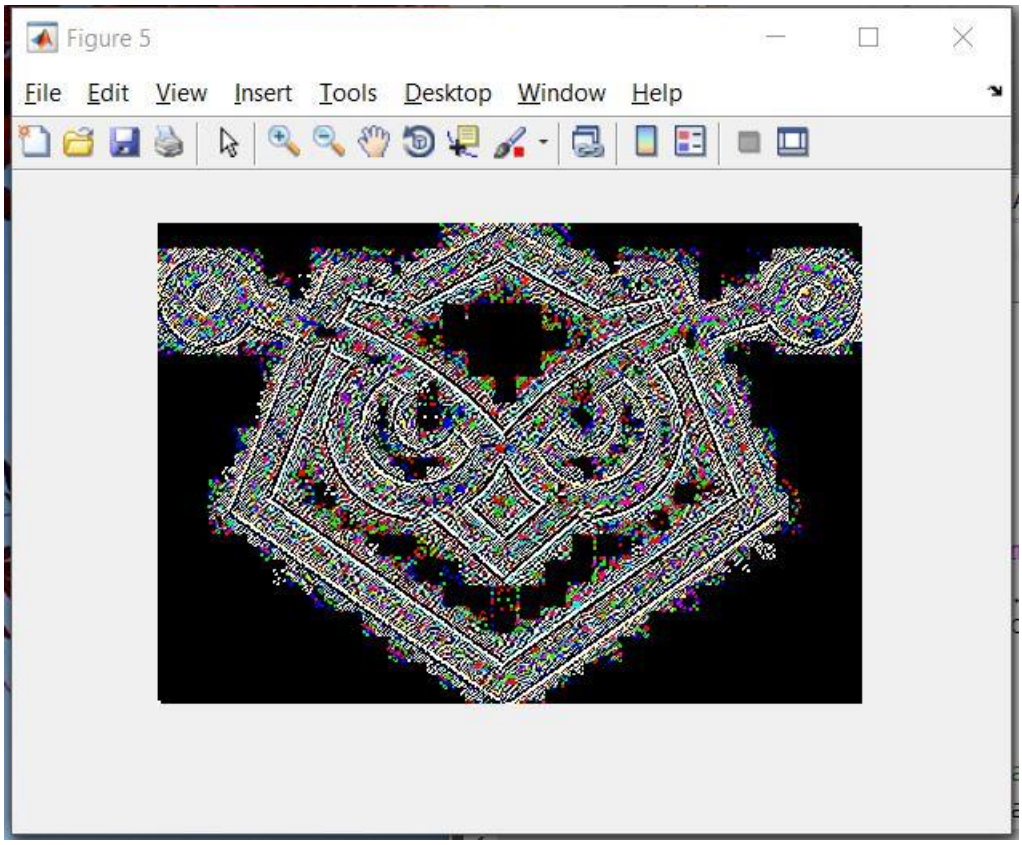


Figure 5
File Edit View Insert Tools Desktop Window Help

EXPERIMENT NO. 11

AIM: Write and execute program for image morphological operations: erosion and dilation

Introduction:

- Morphology is a branch of biology that deals with form and structure of animal and plant
- In image processing, we use mathematical morphology to extract image components which are useful in representation and description of region shape such as … Boundaries, Skeletons, Convex hull, Thinning, Pruning etc.
- Two Fundamental morphological operations are: Erosion and dilation
- Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries.

Erosion and dilation are two fundamental image morphological operations.

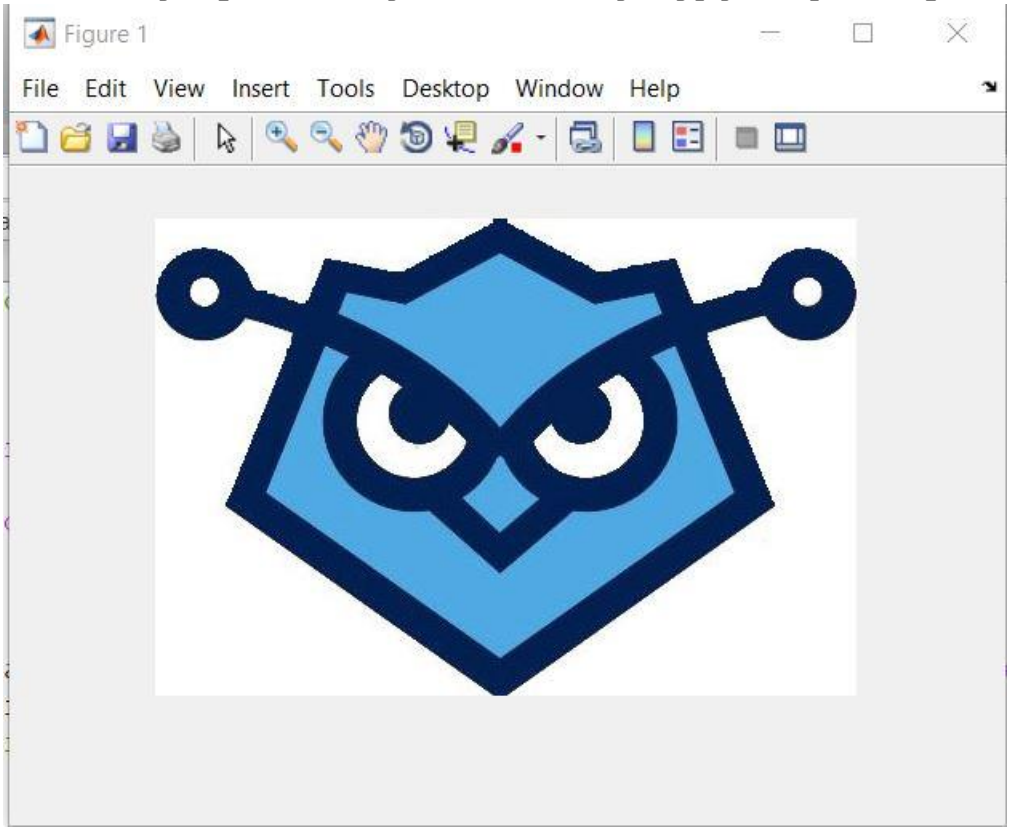
- Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries.
- Dilation operation: The value of the output pixel is the maximum value of all the pixels in the input pixel"s neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.
- Erosion operation: The value of the output pixel is the minimum value of all the pixels in the input pixel"s neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.
- Opening and closing operations can be done by combination of erosion and dilation in different sequence.

Program:

```
% Program to demonstrate Erosion and Dilation
clear all;
clc;
while 1
K = menu('Erosion and dilation demo','Choose
Image','Choose 3x3 Mask','Choose 5x5 Mask','Choose
Structure Image','Erosion','Dilation','Opening',
'Closing','EXIT')
%B=[1 1 1;1 1 1;1 1 1;];
```

```
B = strel('disk', 9);
%B = strel('disk', 5);
switch K
    case 1,
[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*. jpg;
*.jpeg;*.gif','IMAGE Files (*.bmp,*.tif,*.tiff,*.jpg,
*.jpeg,*.gif)'},'Chose GrayScale Image');
     A=imread(strcat(pathname,namefile));
    %data=rgb2gray(data);
    imshow(A);
    case 2,
         B=[1 1  1;1 1 1;1  1 1;];
    case 3,
  B=[1 1 1  1 1;1  1 1 1  1;1 1  1 1 1;1 1 1 1 1;1 1 1 1 1;];
    case 4,
[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*. jpg;
*.jpeg;*.gif','IMAGE Files (*.bmp,*.tif,*.tiff,*.jpg,
*.jpeg,*.gif)'},'Chose GrayScale Image');
     B=imread(strcat(pathname,namefile));
    %data=rgb2gray(data);
    figure;
    imshow(B);
    case 5,
        C=imerode(A,B);
        figure;
        imshow(C);
    case 6,
        C=imdilate(A,B);
        figure;
        imshow(C)
    case 7,
        C=imdilate(A,B);
        D=imerode(C,B);
        figure;
        imshow(D)
    case 8,
        C=imerode(A,B);
        D=imdilate(C,B);
        figure;
        imshow(D)
    case 9,
        break;
    otherwise,
```

```
        msgbox('Select proper mask');
end
end
close all
%Write an image to a file
imwrite(mat2gray(outimage),'outimage.jpg','quality',99);
```



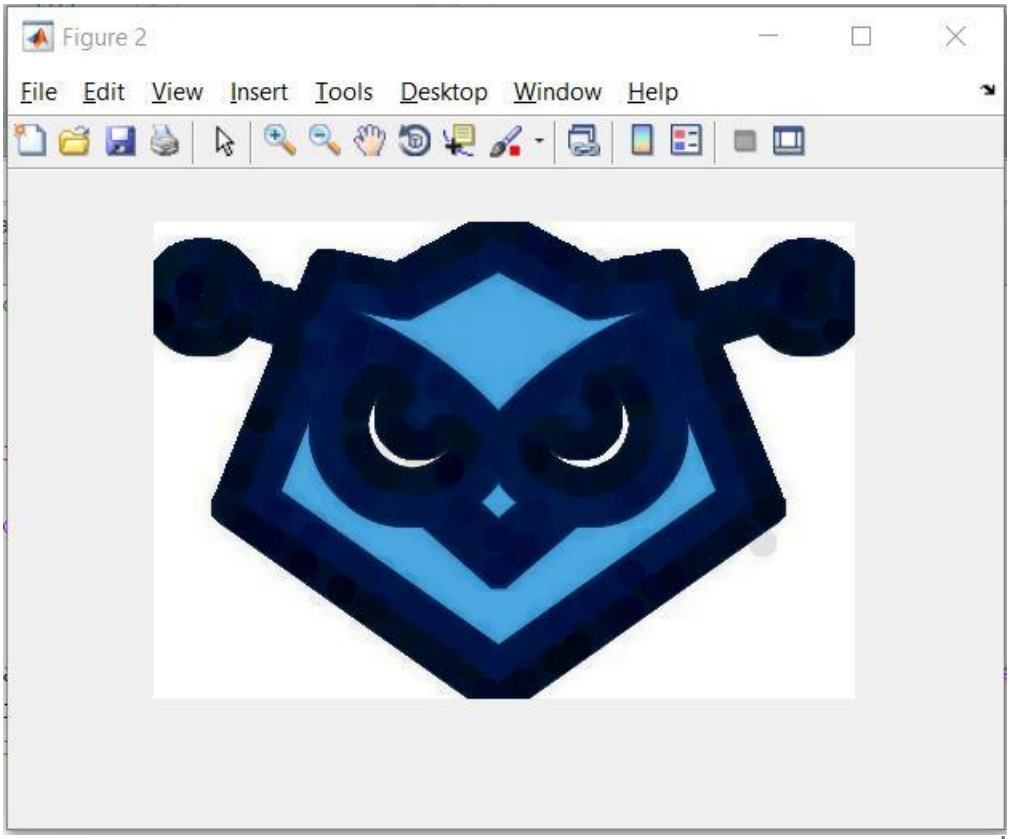


Figure 2
File Edit View Insert Tools Desktop Window Help

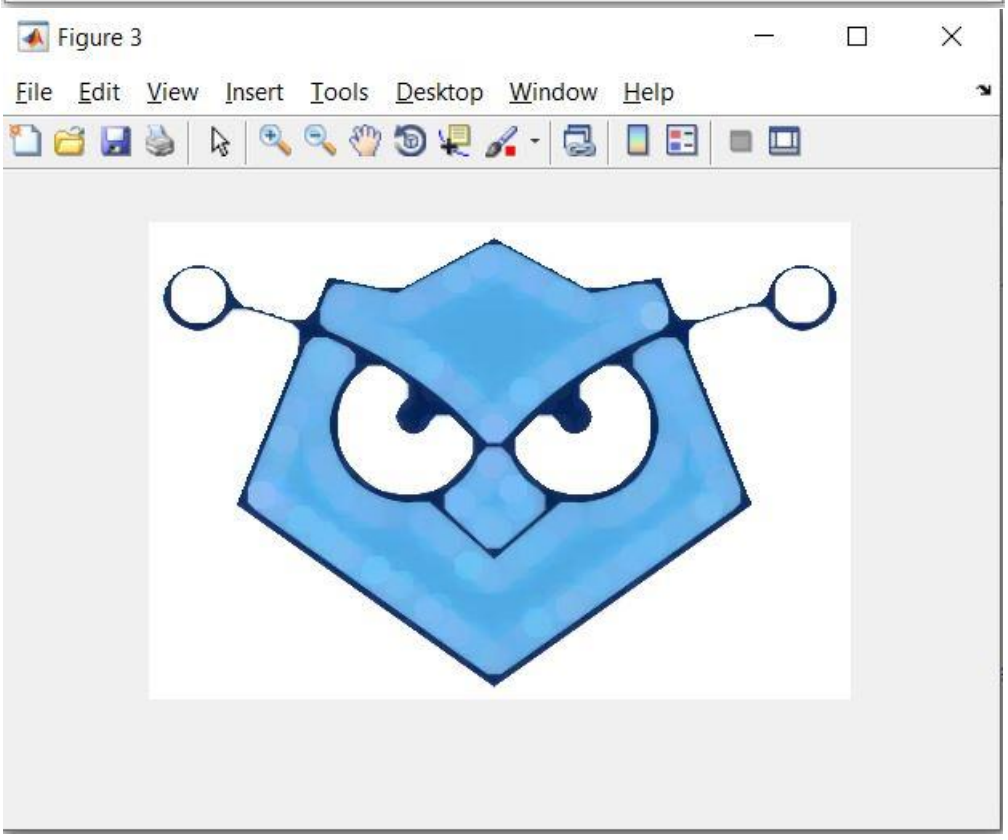


Figure 3
File Edit View Insert Tools Desktop Window Help

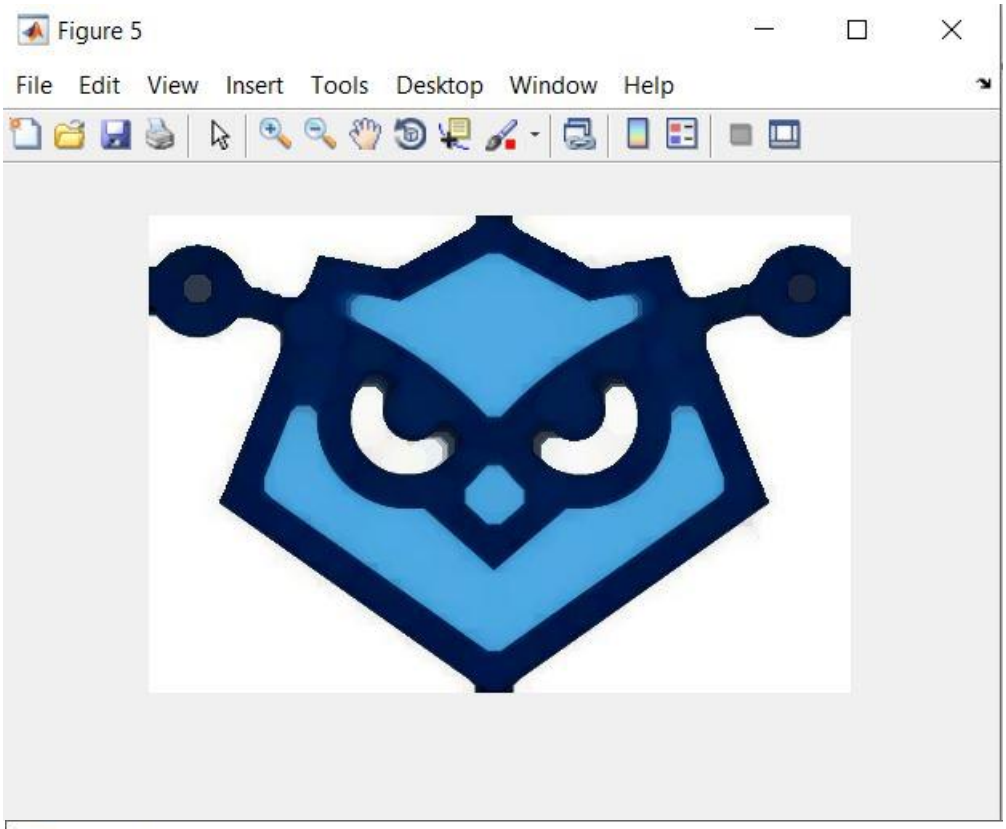


Figure 5
File Edit View Insert Tools Desktop Window Help

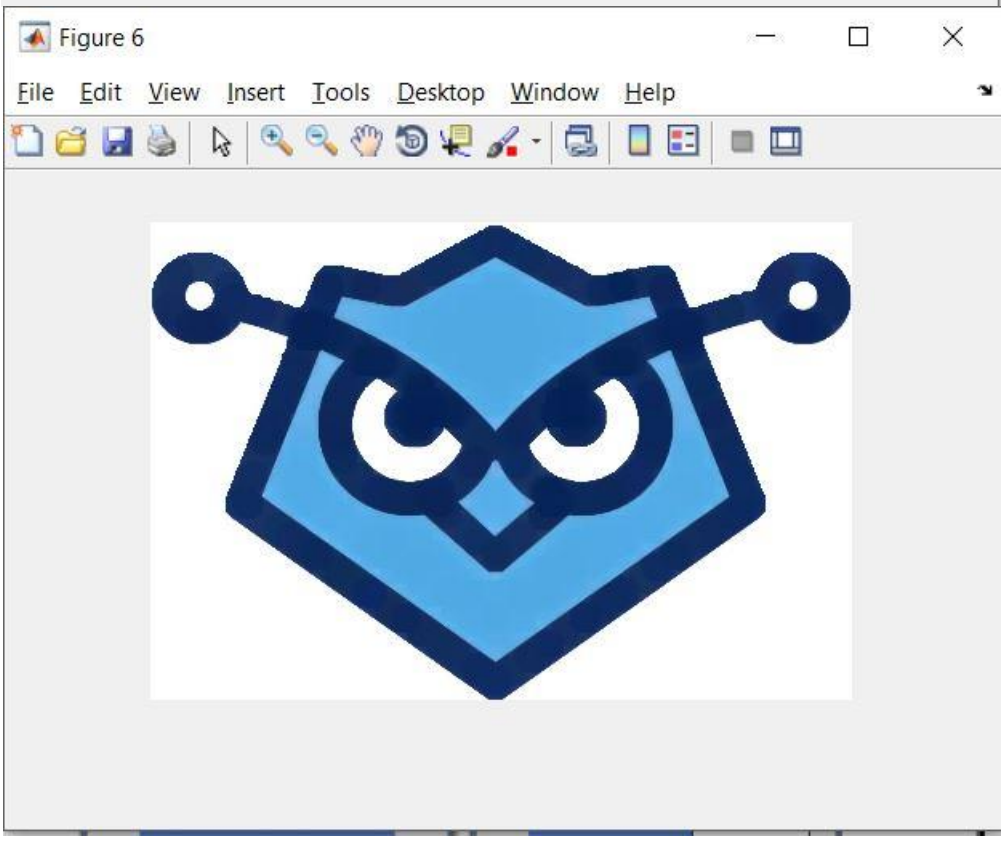


Figure 6
File Edit View Insert Tools Desktop Window Help

EXPERIMENT NO. 12

AIM:

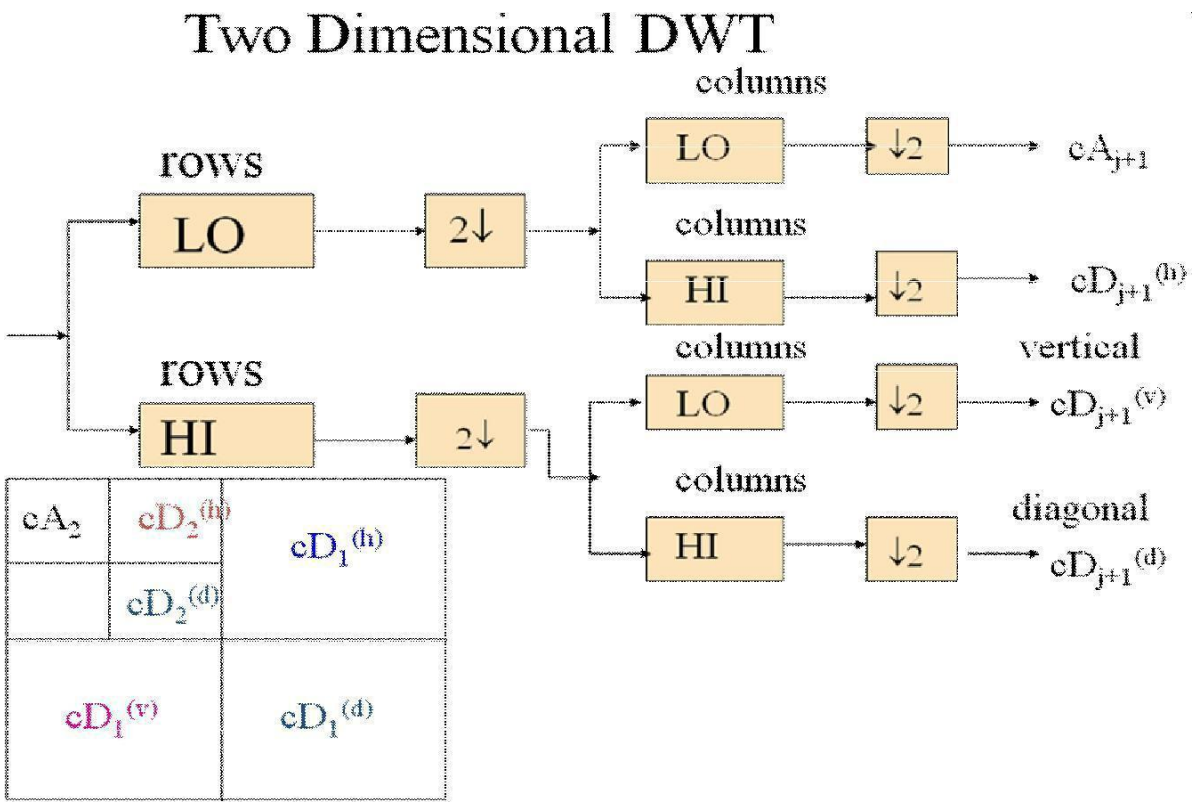
To write and execute program for wavelet transform on given image and perform inverse wavelet transform to reconstruct image.

Introduction:

Wavelet transform is relatively recent signal and image processing tool which has many applications. Basis functions of Fourier transform is sinusoids while basis functions of wavelet transform is wavelets.

- Wavelets are oscillatory functions vanishing outside the small interval hence, they are called wavelets. wave-let (small fraction of wave).
- Wavelets are building blocks of the signal.
- Wavelets are the functions which are well suited to the expansion of real time non-stationary signals.
- Wavelets can be used to de-correlate the correlations present in real time signal such as speech/audio, video, biomedical, seismic etc.

General block diagram of two dimensional wavelet transform:



Following standard matlab functions are used in this experiment.

[1] wavedec2 : Function for multi-level decomposition of 2D data using wavelet transform.

[C,S] = WAVEDEC2(X,N,'wname') returns the wavelet decomposition of the matrix X at level N, using the wavelet named in string 'wname' (wavelet name can be Daubechies wavelet db1, db2, db3 or any other wavelet).

Outputs are the decomposition vector C and the corresponding bookkeeping matrix S. N is level of decomposition which must be positive integer.

[2] appcoef2: This function utilize tree developed by wavelet decomposition function and constructs approximate coefficients for 2D data.

A = APPCOEF2(C,S,'wname',N) computes the approximation coefficients at level N using the wavelet decomposition structure [C,S] which is generated by function wavedec2.

[3] detcoef2: This function utilize tree develet by wavelet decomposition function and generates detail coefficients for 2D data.

D = DETCOEF2(O,C,S,N) extracts from the wavelet decomposition structure [C,S], the horizontal, vertical or diagonal detail coefficients for O = 'h' or 'v' or 'd', for horizontal, vertical and diagonal detail coefficients respectively.

[4] waverec2: Multilevel wavelet 2D reconstruction (Inverse wavelet transform)

WAVEREC2 performs a multilevel 2-D wavelet reconstruction using wavelet named in string 'wname' (wavelet name can be Daubechies wavelet db1, db2, db3 or any other wavelet).

X = WAVEREC2(C,S,'wname') reconstructs the matrix X based on the multi-level wavelet decomposition structure [C,S] which is generated by wavedec2

Program:

```
clc;
close;
[namefile,pathname]=uigetfile({'*.bmp;*.tif;*.tiff;*.jpg;*.jpeg;*.gif','IMAGE
Files (*.bmp,*.tif,*.tiff,*.jpg,*.jpeg,*.gif)'},'Chose GrayScale Image');
X=imread(strcat(pathname,namefile));
if(size(X,3)==3)
X=rgb2gray(X);
end imshow(X);

% Perform wavelet decomposition at level 2.
[c,s] = wavedec2(X,2,'db1');
figure;
imshow(c,[]); title('Wavelet decomposition data generated by wavedec2');
figure;
%Calculate first level approx. and detail components ca1 =
appcoef2(c,s,'db1',1); subplot(2,2,1);imshow(ca1,[]);title('First
level approx'); ch1 = detcoef2('h',c,s,1);
subplot(2,2,2);imshow(ch1,[]);title('First level horixontal detail');
cv1 = detcoef2('v',c,s,1); subplot(2,2,3);imshow(cv1,[]);title('First
level vertical detail'); cd1 = detcoef2('d',c,s,1);
subplot(2,2,4);imshow(cd1,[]);title('First level diagonal detail');
%Calculate second level approx. and detail components figure;


ca2 = appcoef2(c,s,'db1',2);
subplot(2,2,1);imshow(ca2,[]);title('Second level approx');
ch2 = detcoef2('h',c,s,2);
subplot(2,2,2);imshow(ch2,[]);title('Second level horizontal
detail'); cv2 = detcoef2('v',c,s,2);
subplot(2,2,3);imshow(cv2,[]);title('Second level vertical detail');
cd2 = detcoef2('d',c,s,2);
subplot(2,2,4);imshow(cd2,[]);title('Second level diagonal detail');
figure;
a0 = waverec2(c,s,'db1');
imshow(a0,[]);title('Reconstructed image using inverse wavelet transform');
```

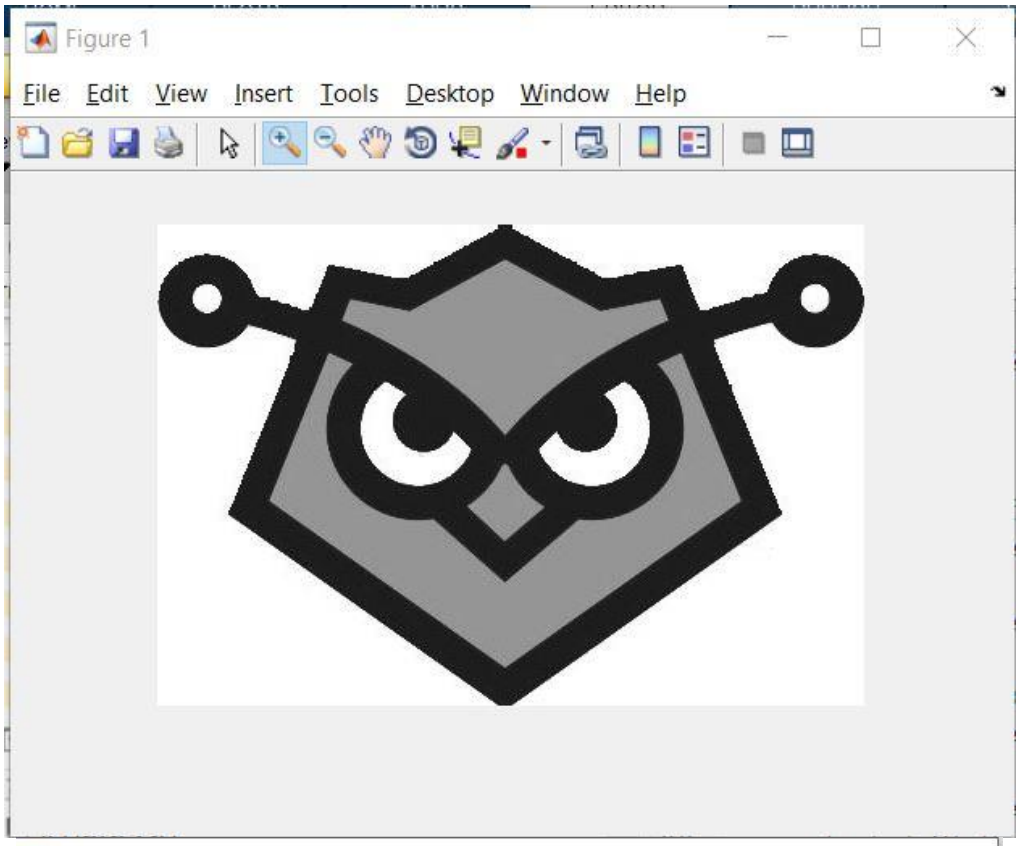


Figure 1
File Edit View Insert Tools Desktop Window Help

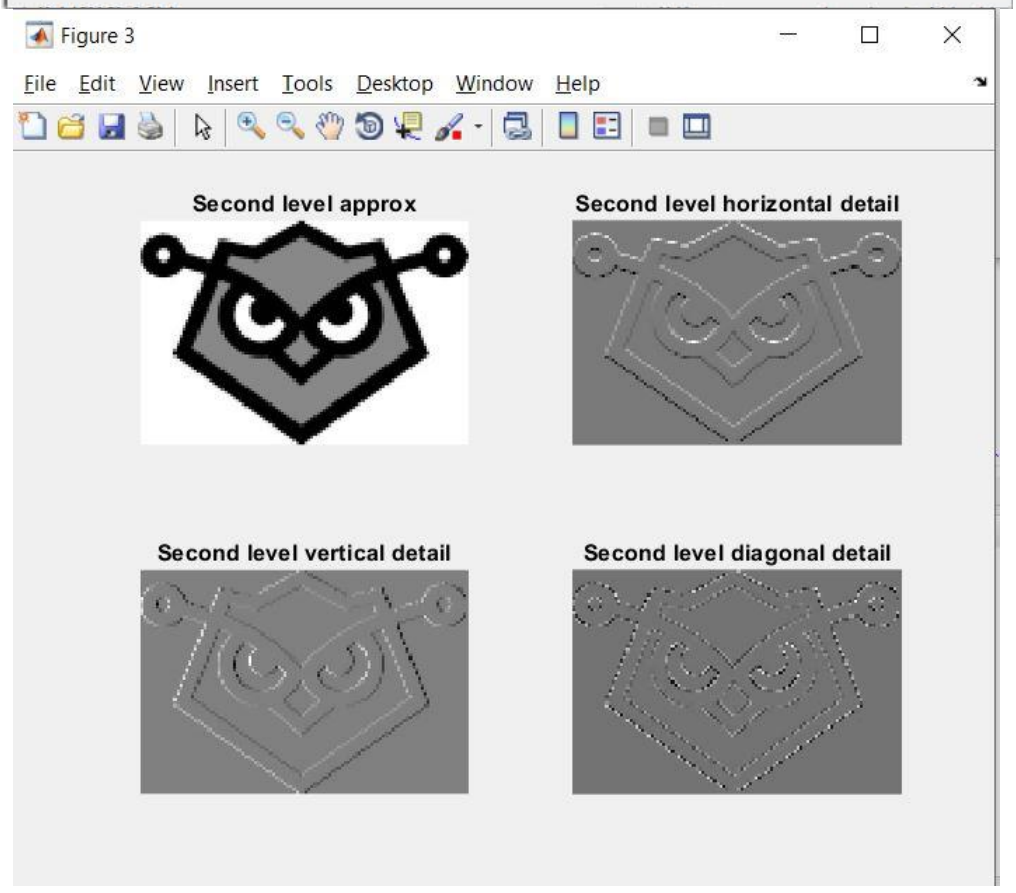


Figure 3
File Edit View Insert Tools Desktop Window Help
Second level approx
Second level horizontal detail
Second level vertical detail
Second level diagonal detail

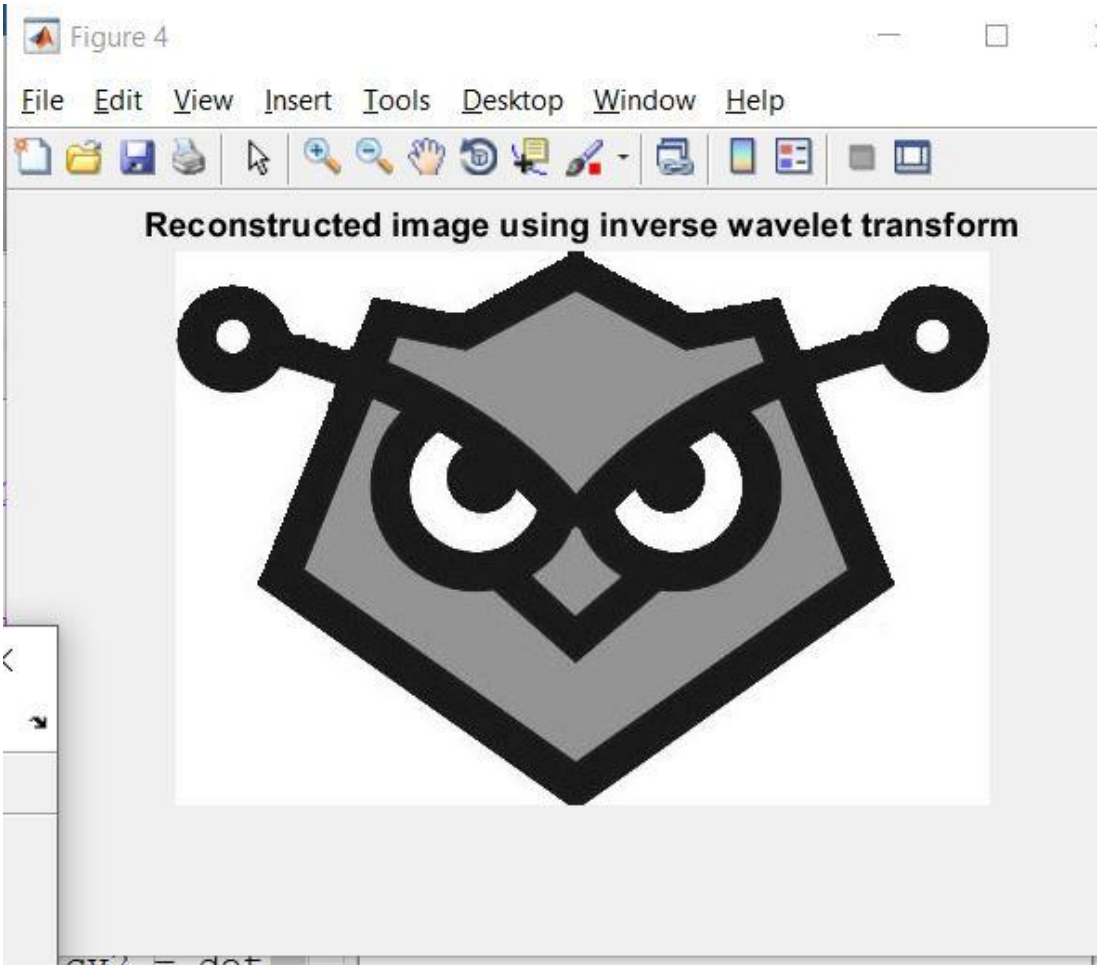


Figure 4
File Edit View Insert Tools Desktop Window Help
Reconstructed image using inverse wavelet transform