**IPML PRACTICALS:**

**Exp 1.1: A. To study different basic image processing operations.**

**B. To perform different geometric transformation on an image. (Rotation,**

**shearing using affine transform)**

**Code:**

clc

clear all

close all

img=imread('C:\Users\micromax\Desktop\311824.jpg');

imshow(img);

resize=imresize(img,[256,256]);

figure(1)

subplot(3,3,1)

imshow(resize)

title('resized')

grayimg= rgb2gray(img);

subplot(3,3,2)

imshow(grayimg)

title('gray level image')

binary= im2bw(img,0.5);

subplot(3,3,3)

imshow(binary)

title('binary image')

red=img(:,:,1);

subplot(3,3,4)

imshow(red)

title('red layer')

blue=img(:,:,2);

subplot(3,3,5)

imshow(blue)

title('blue layer')

green=img(:,:,3);

subplot(3,3,6)

imshow(green)

title('green layer')

rgbimage=cat(3,red,blue,green);

subplot(3,3,7)

imshow(rgbimage)

title('RGB Image')

rotate= imrotate(img,-45);

subplot(3,3,8)

imshow(rotate)

title('rotated image')

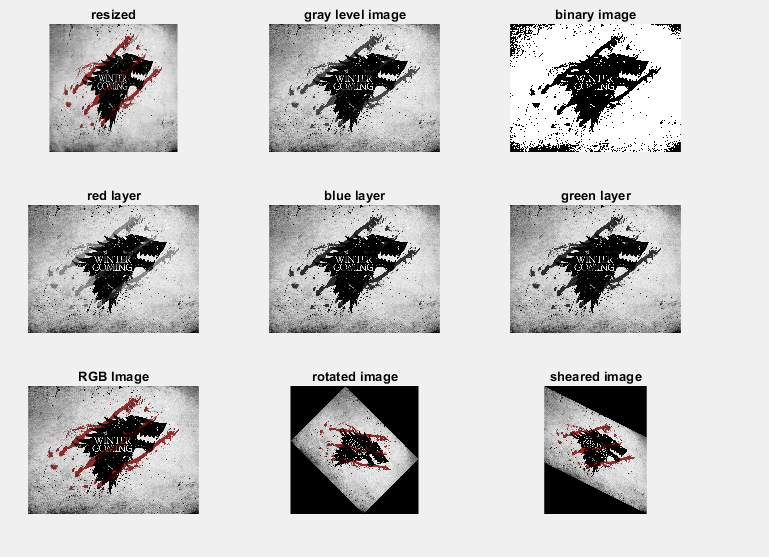
tform = maketform('affine',[1 0.5 0; 0 1 0; 0 0 1]);

shear = imtransform(img,tform);

subplot(3,3,9)

imshow(shear)

title('sheared image')

**Output:**

**Exp 1 To perform different arithmetic.2: and logical operations on given image. (Addition, Subtraction, Multiplication , And, Nand, Or, Nor, Not, XOR, XNOR.**

**Code:**

clc

close all

clear all

img1=imread('C:\Users\micromax\Desktop\311824.jpg');

img2=imread('C:\Users\micromax\Desktop\678317.jpg');

img3=imread('C:\Users\micromax\Desktop\508286.jpg');

img1=imresize(img1,[256,256]);

img2=imresize(img2,[256,256]);

img3=imresize(img3,[256,256]);

sub=imsubtract(img2,img1);

add=imadd(img3,sub) ;

figure(1)

sub1=img2-img1;

add1=img3+sub1;

subplot(3,3,1)

imshow(img1)

title('Image 1')

subplot(3,3,2)

imshow(img2)

title('Image 2')

subplot(3,3,3)

imshow(img3)

title('Image 3')

subplot(3,3,5)

imshow(sub)

title('Subtraction using imsub')

subplot(3,3,6)

imshow(add)

title('Addition using imadd')

subplot(3,3,8)

imshow(sub1)

title('Subtraction using -')

subplot(3,3,9)

imshow(add1)

title('Addition using +')

subplot(3,3,4)

imshow(img1+50)

title('Image 1 + 50')

subplot(3,3,7)

imshow(img1-80)

title('Image 1 - 80')

p=ones(3,3);

q=zeros(3,3);

A=[p q p q ; p q p q; p q p q; p q p q];

B=A';

AND=A&B;

OR=A|B;

XOR=xor(A,B);

NOT=~XOR;

figure(2)

subplot(3,2,1)

imshow(A)

title('Image A')

subplot(3,2,2)

imshow(B)

title('Image B')

subplot(3,2,3)

imshow(AND)

title('AND Operation')

subplot(3,2,4)

imshow(OR)

title('OR Operation')

subplot(3,2,5)

imshow(XOR)

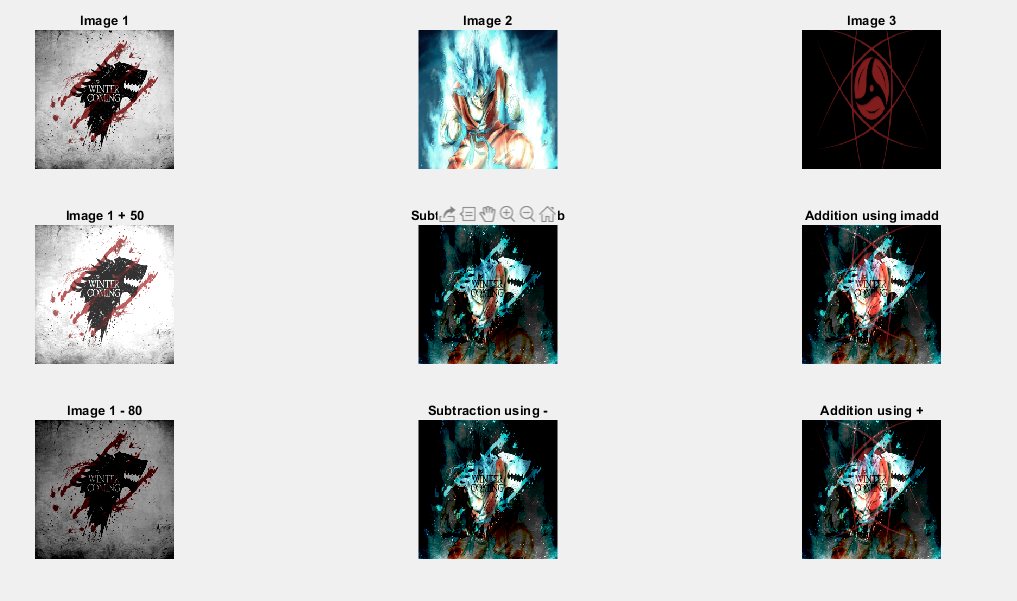
title('XOR Operation')

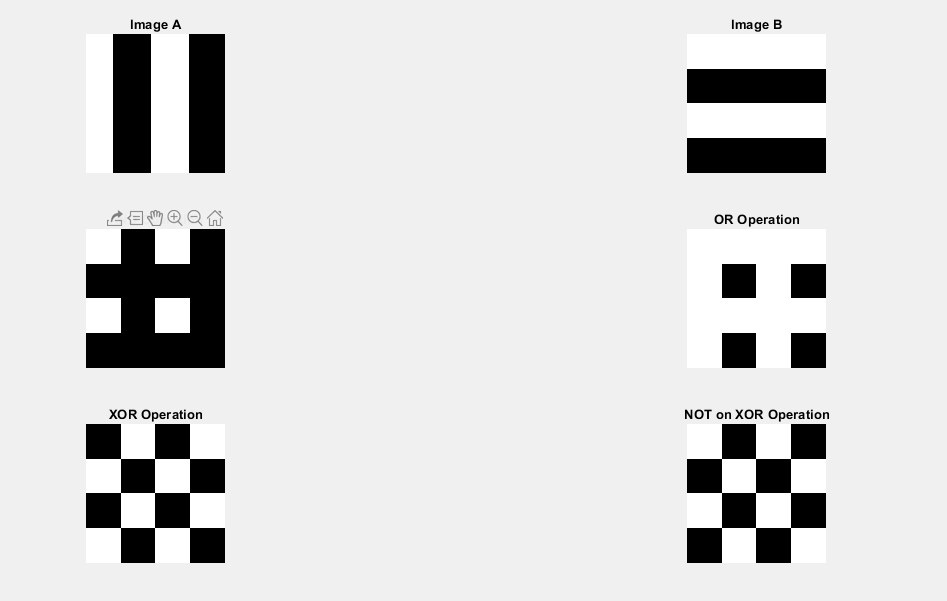
subplot(3,2,6)

imshow(NOT)

title('NOT on XOR Operation')

**Output:**





**Exp 2: To perform Spatial Domain Image Enhancement using different Point Processing techniques.**

* **Negative**
* **Thresholding**
* **Intensity Slicing/ gray level slicing**
* **Log Transform**
* **Power law Transform**

**Code:**

clc;

clear all;

close all;

a=imread('C:\Users\micromax\Desktop\311824.jpg');

subplot(5,4,1);

imshow(a)

title('original image');

A=imresize(a,[246 300]);

x=rgb2gray(A)

subplot(5,4,2);

imshow(x);

title('RGB to gray image');

b=255-x;

subplot(5,4,3);

imshow(b)

title('Negative image');

subplot(5,4,4);

stem(x,b);

title('Negative graph')

a1=x;

for i=1:246

for j=1:300

if (a1(i,j)<=125)

a1(i,j)=0;

else

a1(i,j)=255;

end

end

end

subplot(5,4,5)

imshow(a1)

title('thresholding');

subplot(5,4,6)

stem(x,a1);

title('Thresholding graph');

for i=1:246

for j=1:300

if (x(i,j)>=125 && x(i,j)<180)

a2(i,j)=255;

else

a2(i,j)=0;

end

end

end

subplot(5,4,7)

imshow(a2)

title('gray level slicing with background');

subplot(5,4,8)

stem(x,a2);

title('gray level slicing graph');

for i=1:246

for j=1:300

if (x(i,j)>=125 && x(i,j)<180)

a3(i,j)=255;

else

a3(i,j)=x(i,j);

end

end

end

subplot(5,4,9)

imshow(a3)

title('gray slicing level without background');

subplot(5,4,10)

stem(x,a3);

title('gray level slicing graph');

b=im2double(x);

s=3\*(log(b+1));

subplot(5,4,11);

imshow(s)

title('log transform');

subplot(5,4,12);

stem(x,s)

title('log transform graph');

c=im2double(x);

s=3\*(log(b+1));

subplot(5,4,13);

imshow(s)

title('power law');

subplot(5,4,14);

stem(x,s)

title('New image vs og image');

c=im2double(x);

y=3\*(c.^1.5);

subplot(5,4,15);

imshow(y)

title('power law transform gamma=1.5');

subplot(5,4,16);

stem(x,y)

title('power law transform ');

z=0.6\*(c.^0.5);

subplot(5,4,17);

imshow(z)

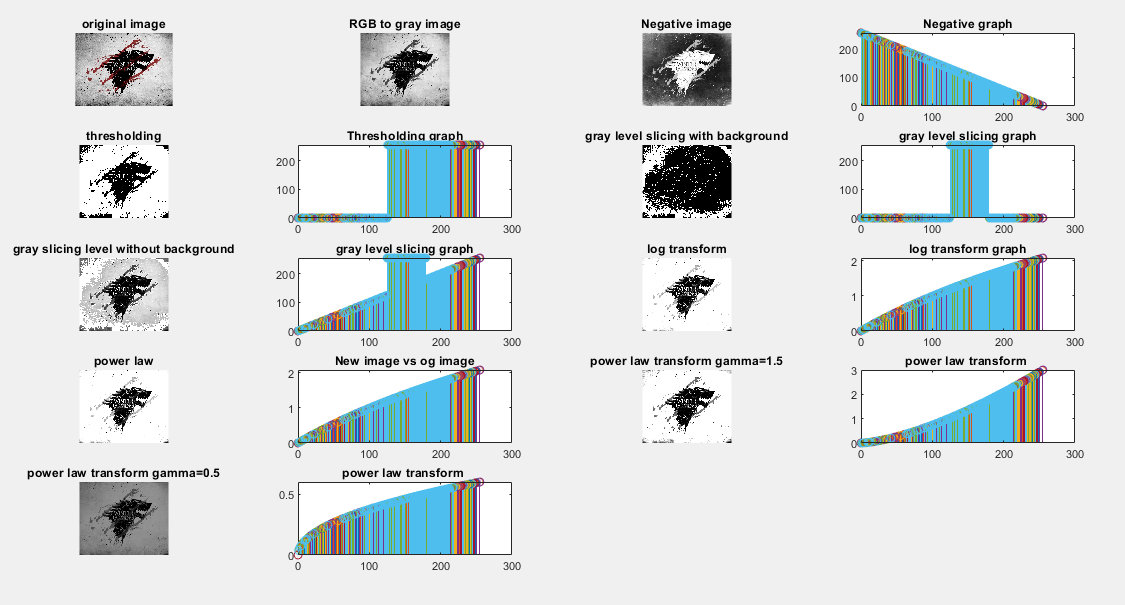
title('power law transform gamma=0.5');

subplot(5,4,18);

stem(x,z)

title('power law transform ');

**Output:**

****

**Exp 3: To perform spatial domain image enhancement using different neighbourhood processing techniques.**

**Code:**

clc

clear all

close all

a1=imread('C:\Users\micromax\Desktop\311824.jpg');

subplot(4,4,1);

imshow(a1)

title('256x256 original image')

a = double(a1)

t1 = (1/9)\*[1 1 1; 1 1 1; 1 1 1]; %3x3 averaging mask

t2 = (1/25)\*[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]; %5x5 averaging mask

t3 = (1/16)\*[1 2 1;2 4 2;1 2 1];% 3x3 weighted averagfe mask

t4 = (1/9)\*[-1 -1 -1; -1 8 -1; -1 -1 -1]; %3x3 sharpening mask

t5=[0 -1 0; -1 4 -1; 0 -1 0]; %3x3 laplacian mask

t6=[-1 -2 -1; -2 12 -2; -1 -2 -1]; %3x3 weighted mask

t7=[-1 -1 -1; 0 0 0; 1 1 1]; %3x3 horizontal mask

t8=[-1 0 1; -1 0 1; -1 0 1]; %3x3 vertical mask

t9=[-1 -1 -1; -1 8.9 -1; -1 -1 -1]; %High boost mask

for i=2:255

for j=2:255

x1(i,j)=sum(sum(t1.\*a(i-1:i+1,j-1:j+1)));

end

end

subplot(4,4,2);

imshow(uint8(x1));

title('3x3 averaging mask')

for i=3:254

for j=3:254

x2(i,j)=sum(sum(t2.\*a(i-2:i+2,j-2:j+2)));

end

end

subplot(4,4,3);

imshow(uint8(x2));

title('5x5 averaging mask')

for i=2:255

for j=2:255

x3(i,j)=sum(sum(t3.\*a(i-1:i+1,j-1:j+1)));

end

end

subplot(4,4,4);

imshow(uint8(x3));

title('3x3 weighted averaging mask')

for i=2:255

for j=2:255

x4(i,j)=sum(sum(t4.\*a(i-1:i+1,j-1:j+1)));

end

end

subplot(4,4,5);

imshow(uint8(x4));

title('3x3 sharpening mask')

for i=2:255

for j=2:255

x5(i,j)=sum(sum(t5.\*a(i-1:i+1,j-1:j+1)));

end

end

subplot(4,4,6);

imshow(uint8(x5));

title('3x3 Laplacian mask')

for i=2:255

for j=2:255

x6(i,j)=sum(sum(t6.\*a(i-1:i+1,j-1:j+1)));

end

end

subplot(4,4,7);

imshow(uint8(x6));

title('3x3 wieghted mask')

for i=2:255

for j=2:255

x7(i,j)=sum(sum(t7.\*a(i-1:i+1,j-1:j+1)));

end

end

subplot(4,4,8);

imshow(uint8(x7));

title('Horizontal mask')

for i=2:255

for j=2:255

x8(i,j)=sum(sum(t8.\*a(i-1:i+1,j-1:j+1)));

end

end

subplot(4,4,9);

imshow(uint8(x8));

title('Vertical mask')

for i=2:255

for j=2:255

x9(i,j)=sum(sum(t9.\*a(i-1:i+1,j-1:j+1)));

end

end

subplot(4,4,10);

imshow(uint8(x9));

title('High Boost mask')

d = 0.05;

n = imnoise(a1,'salt & pepper',d);

subplot(4,4,11);

imshow(n)

title('median mask')

for i=2:255

for j=2:255

x10(i,j)=median(median(n(i-1:i+1,j-1:j+1)));

end

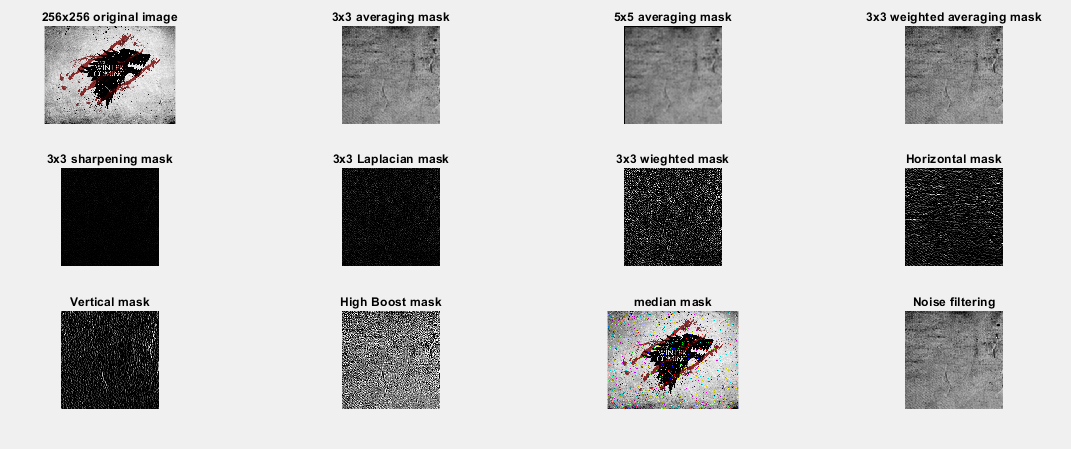
end

subplot(4,4,12);

imshow(uint8(x10));

title('Noise filtering')

**Output:**

****

**Exp 4: To perform frequency domain image enhancement techniques.**

**Code:**

clc

clear all

close all

img=imread('C:\Users\micromax\Desktop\311824.jpg');

img=imresize(img,[256,256]);

figure(1)

subplot(2,2,1)

imshow(img);

title('Original Image')

double\_img=im2double(img);

fft\_image=fft2(double\_img);

subplot(2,2,2)

imshow((fft\_image));

title('FFT of Original Image')

for i=1:256

for j=1:256

translated(i,j)=img(i,j)\*((-1)^(i+j));

end

end

subplot(2,2,3)

imshow(translated);

title('Translated Image')

subplot(2,2,4)

fft\_translated=fft2(translated);

imshow(fft\_translated);

title('FFT of Translated Image')

%Low pass Filtering

D0 = 20;

D0\_high = 20

n=2;

for u=1:256

for v=1:256

D(u,v)=sqrt((u-128)^2 + (v-128)^2 );

if D(u,v)<=D0

H1(u,v)=1; %Ideal Low Pass

H2(u,v)=1/(1+(D(u,v)/D0)^(2\*n)); %Butterworth Filter

H3(u,v)=exp(-((D(u,v))^2)/(2\*D0\*D0)); %Gaussian Filter

else

H1(u,v)=0;

H2(u,v)=0;

H3(u,v)=0;

end

end

end

%high pass Filtering

for u=1:256

for v=1:256

D(u,v)=sqrt((u-128)^2 + (v-128)^2 );

if D(u,v)>D0\_high

H1\_high(u,v)=1; %Ideal High Pass

H2\_high(u,v)=1/(1+(D(u,v)/D0\_high)^(2\*n)); %Butterworth Filter

H3\_high(u,v)=exp(-((D(u,v))^2)/(2\*D0\_high\*D0\_high)); %Gaussian Filter

else

H1\_high(u,v)=0;

H2\_high(u,v)=0;

H3\_high(u,v)=0;

end

end

end

%Ideal LPF

figure(2)

subplot(2,2,1)

imshow(H1);

title('Transfer Function - Ideal LPF')

lpf\_img=fft\_translated.\*H1;

ifft\_lpf\_img=ifft2(lpf\_img);

for i=1:256

for j=1:256

lpf\_filtered(i,j)=ifft\_lpf\_img(i,j)\*((-1)^(i+j));

end

end

subplot(2,2,2)

imshow(uint8(lpf\_filtered));

title('Ideal Low pass filtered image')

%Ideal HPF

subplot(2,2,3)

imshow(H1\_high);

title('Transfer Function - Ideal HPF')

hpf\_img=fft\_translated.\*H1\_high;

ifft\_hpf\_img=ifft2(hpf\_img);

for i=1:256

for j=1:256

hpf\_filtered(i,j)=ifft\_hpf\_img(i,j)\*((-1)^(i+j));

end

end

subplot(2,2,4)

imshow(uint8(hpf\_filtered));

title('Ideal High pass filtered image')

%%Butterworth Filtering

%LPF

figure(3)

subplot(2,2,1)

imshow(H2);

title('Transfer Function - Butterworth Low')

butt\_img=fft\_translated.\*H2;

ifft\_butt\_img=ifft2(butt\_img);

for i=1:256

for j=1:256

butt\_filtered(i,j)=ifft\_butt\_img(i,j)\*((-1)^(i+j));

end

end

subplot(2,2,2)

imshow(uint8(butt\_filtered));

title('Butterworth low filtered image')

%HPF

figure(3)

subplot(2,2,3)

imshow(H2\_high);

title('Transfer Function - Butterworth High')

butt\_high\_img=fft\_translated.\*H2\_high;

ifft\_butt\_high\_img=ifft2(butt\_high\_img);

for i=1:256

for j=1:256

butt\_high\_filtered(i,j)=ifft\_butt\_high\_img(i,j)\*((-1)^(i+j));

end

end

subplot(2,2,4)

imshow(uint8(butt\_high\_filtered));

title('Butterworth high filtered image')

%%Gaussian Filtering

%LPF

figure(4)

subplot(2,2,1)

imshow(H3);

title('Transfer Function - Gaussian Low')

gaus\_img=fft\_translated.\*H3;

ifft\_gaus\_img=ifft2(gaus\_img);

for i=1:256

for j=1:256

gaus\_filtered(i,j)=ifft\_gaus\_img(i,j)\*((-1)^(i+j));

end

end

subplot(2,2,2)

imshow(uint8(gaus\_filtered));

title('Gaussian low filtered image')

%HPF

subplot(2,2,3)

imshow(H3\_high);

title('Transfer Function - Gaussian High')

gaus\_high\_img=fft\_translated.\*H3\_high;

ifft\_gaus\_high\_img=ifft2(gaus\_high\_img);

for i=1:256

for j=1:256

gaus\_high\_filtered(i,j)=ifft\_gaus\_high\_img(i,j)\*((-1)^(i+j));

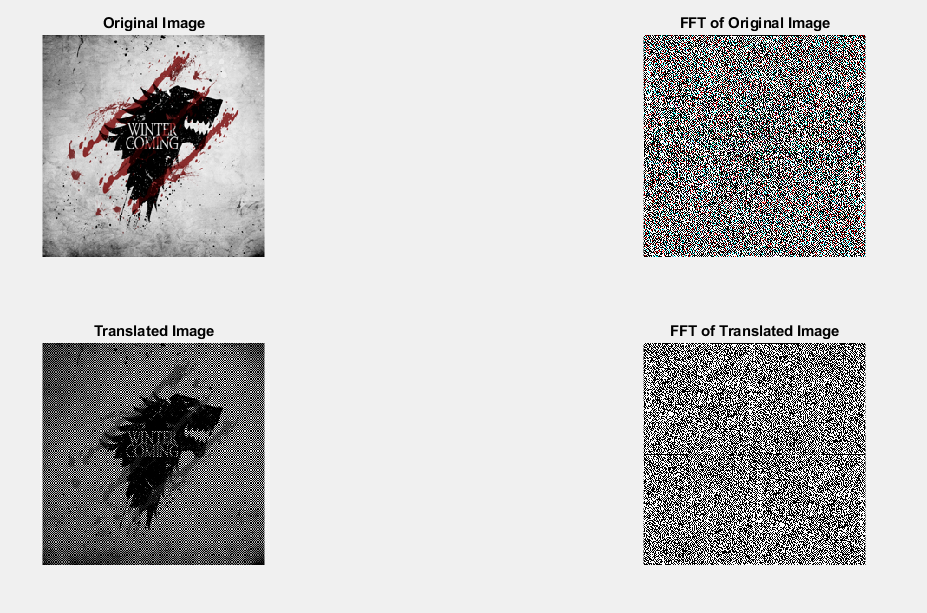
end

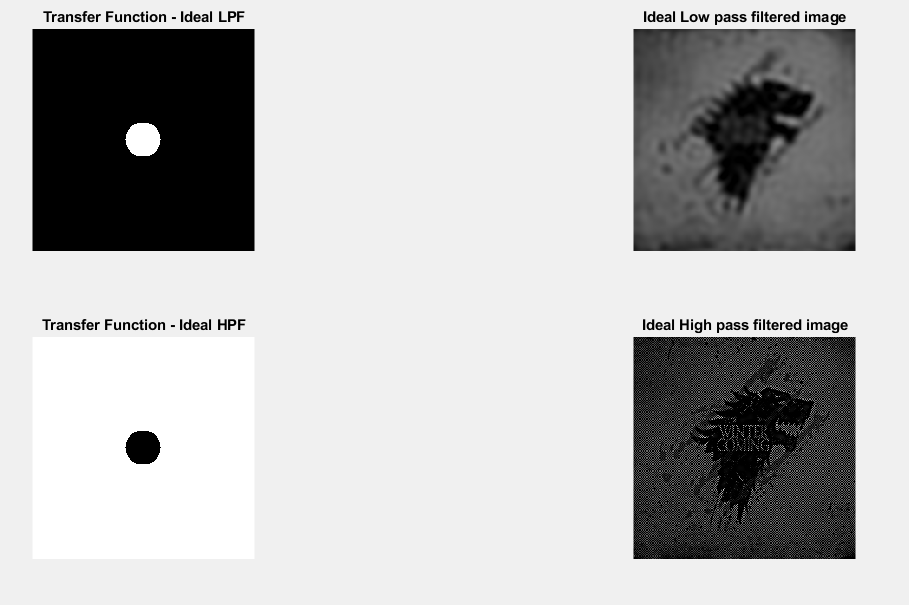
end

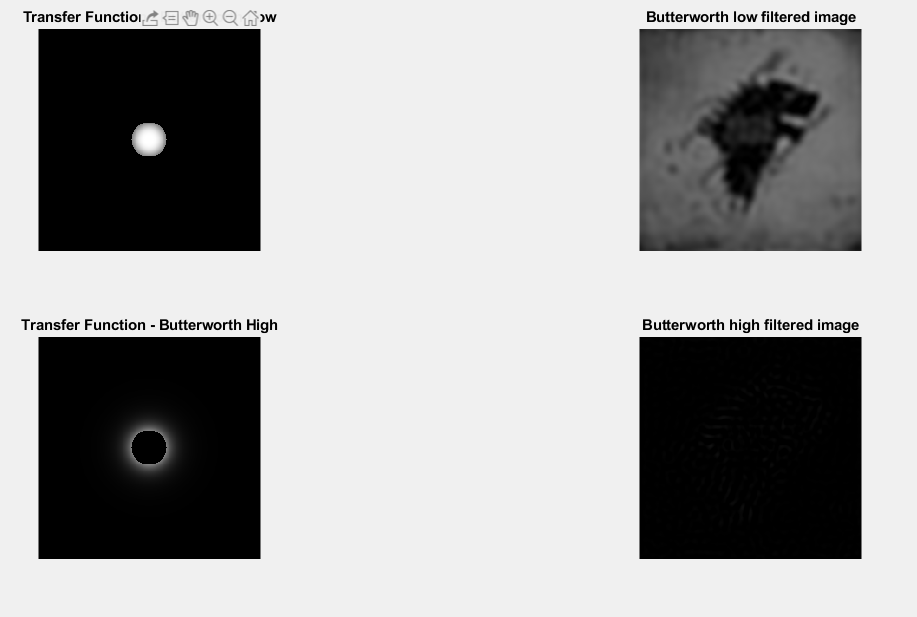
subplot(2,2,4)

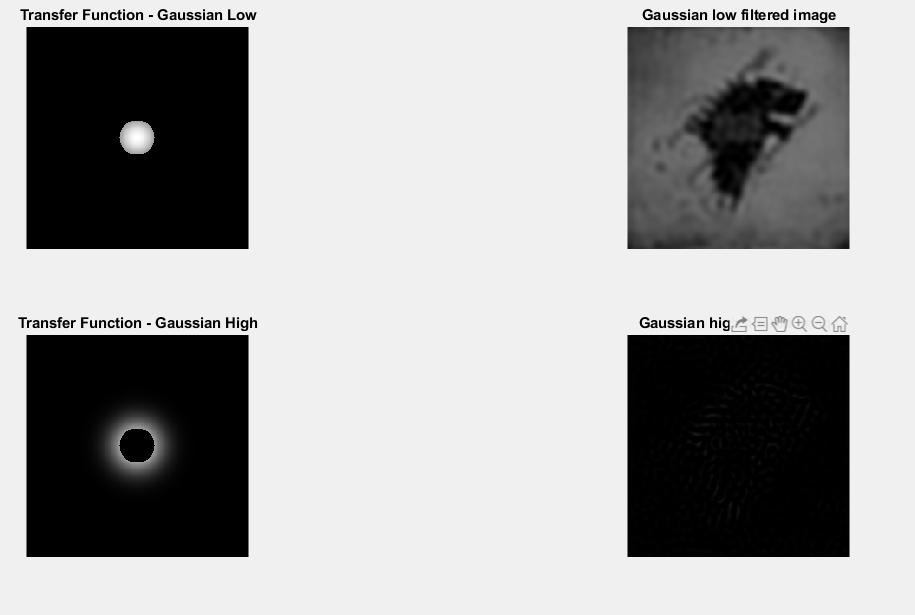
imshow(uint8(gaus\_high\_filtered));

title('Gaussian high filtered image')

**Output:**

****

****

****

**Exp 5: To perform image morphological operations.**

**Code:**

img = imread('C:\Users\micromax\Desktop\311824.jpg');

figure(1)

subplot(3,2,1)

imshow(img)

title('Original Image')

bw=im2bw(img);

subplot(3,2,2)

imshow(bw)

title('Black and White Image')

SE=[1 1 1; 1 1 1; 1 1 1];

d=imdilate(bw,SE);

e=imerode(bw,SE);

subplot(3,2,3)

imshow(d)

title('Dilation using inbuilt fn')

subplot(3,2,4)

imshow(e)

title('Erosion using inbuilt fn')

%Dilation and Erosion using Code

for i=2:255

for j=2:255

xyz=bw(i-1: i+1, j-1:j+1).\*SE;

d\_code(i,j)=max(max(xyz));

e=isequal(xyz,SE);

if e==0

e\_code(i,j)=0;

else

e\_code(i,j)=1;

end

end

end

subplot(3,2,5)

imshow(d\_code)

title('Dilation using Code')

subplot(3,2,6)

imshow(e\_code)

title('Erosion using Code')

%Part 2

SE2 = strel('disk',1);

img2 = imread('C:\Users\micromax\Desktop\678317.jpg');

%img2=imresize(img2,[256 256]);

figure(2)

subplot(3,2,1)

imshow(img2);

title('Resized Image')

bw2=im2bw(img2);

subplot(3,2,2)

imshow(bw2)

title('Black and White Image')

d2=imdilate(bw2,SE2);

e2=imerode(bw2,SE2);

subplot(3,2,3)

imshow(d2)

title('Dilation using STREL SE')

subplot(3,2,4)

imshow(e2)

title('Erosion using STREL SE')

o = imopen(bw2,SE2);

subplot(3,2,5)

imshow(o)

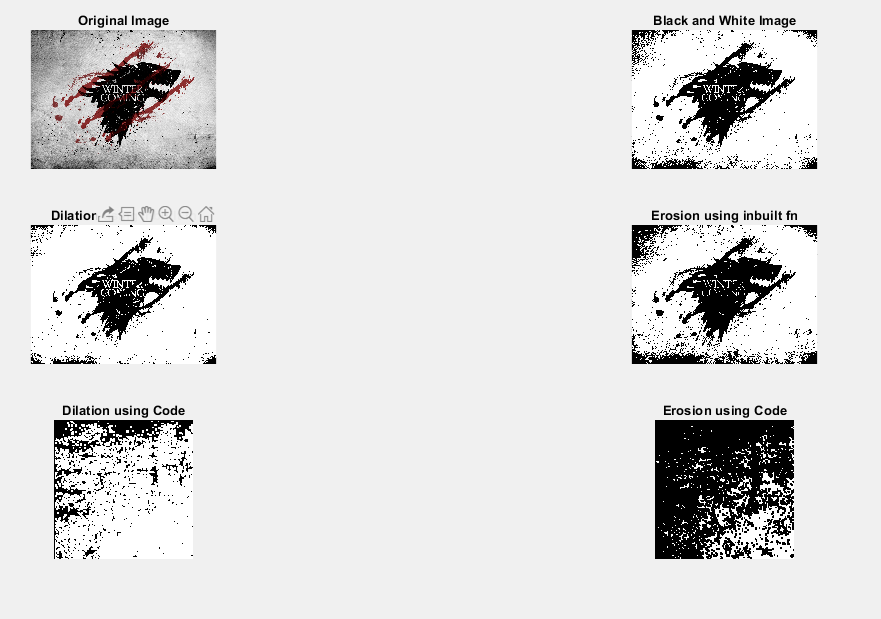
title('Opening using Matlab fn')

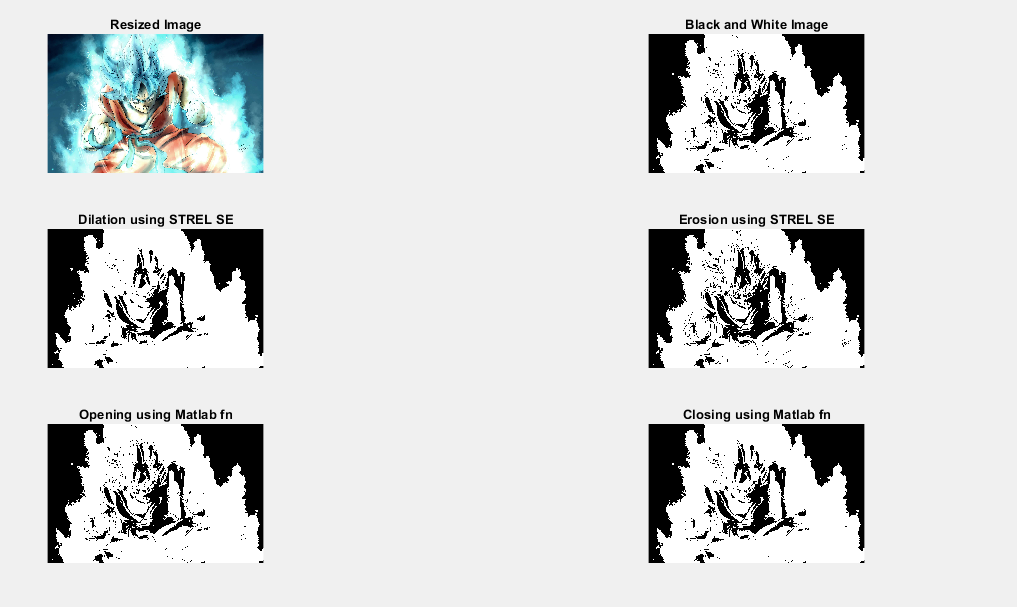
c = imclose(bw2,SE2);

subplot(3,2,6)

imshow(c)

title('Closing using Matlab fn')

**Output:**

****

**Exp 6: To perform image segmentation operations.**

**Code:**

**Without inbuilt commands**

clc;

clear all;

close all;

C=imread('C:\Users\micromax\Desktop\678317.jpg');

A=imresize(C, [256 256]);

A=rgb2gray(A);

subplot(2,4,1);

imshow(A);

title('Gray scale image');

a=im2double(A);

m1=[1 1;-1 -1];

for i=2:255

for j=2:255

x(i,j)=sum(sum(a(i-1:i,j-1:j).\*m1));

end

end

subplot(2,4,2);

imshow(x);

title('Roberts Mask');

m2=[-1 -1 -1;0 0 0;1 1 1];

for i=2:255

for j=2:255

s(i,j)=sum(sum(a(i-1:i+1,j-1:j+1).\*m2));

end

end

subplot(2,4,3);

imshow(s);

title('Prewitts H Mask');

m3=[-1 0 1;-1 0 1;-1 0 1];

for i=2:255

for j=2:255

t(i,j)=sum(sum(a(i-1:i+1,j-1:j+1).\*m3));

end

end

subplot(2,4,4);

imshow(t);

title('Prewitts V Mask');

m4=[-2 -1 0;-1 0 1;0 1 2];

for i=2:255

for j=2:255

u(i,j)=sum(sum(a(i-1:i+1,j-1:j+1).\*m4));

end

end

subplot(2,4,5);

imshow(u);

title('Prewitts Full Mask');

m5=[-2 -2 0;-2 0 2;0 2 2];

for i=2:255

for j=2:255

v(i,j)=sum(sum(a(i-1:i+1,j-1:j+1).\*m5));

end

end

subplot(2,4,6);

imshow (v);

title('Sobel Mask');

m6=[0 1 0;1 -4 1;0 1 0];

for i=2:255

for j=2:255

w(i,j)=sum(sum(a(i-1:i+1,j-1:j+1).\*m6));

end

end

subplot(2,4,7);

imshow(w);

title('Laplacian Mask');

**With inbuilt functions:**

clc;

clear all;

close all;

C=imread('C:\Users\micromax\Desktop\311824.jpg');

A=imresize(C, [256 256]);

A=rgb2gray(A);

subplot(2,4,1);

imshow(A);

title('Gray scale image');

t=edge(A,'roberts');

subplot(2,4,2);

imshow(t);

title('Roberts Mask');

u=edge(A,'prewitt');

subplot(2,4,3);

imshow(u);

title('Prewitts Mask');

v=edge(A,'sobel');

subplot(2,4,4);

imshow(v);

title('Sobel Mask');

w=edge(A,'zerocross');

subplot(2,4,5);

imshow(v);

title('Laplacian Mask');

x=edge(A,'log');

subplot(2,4,6);

imshow(x);

title('Marr and Hildreth Edge Detection');

y=edge(A,'canny');

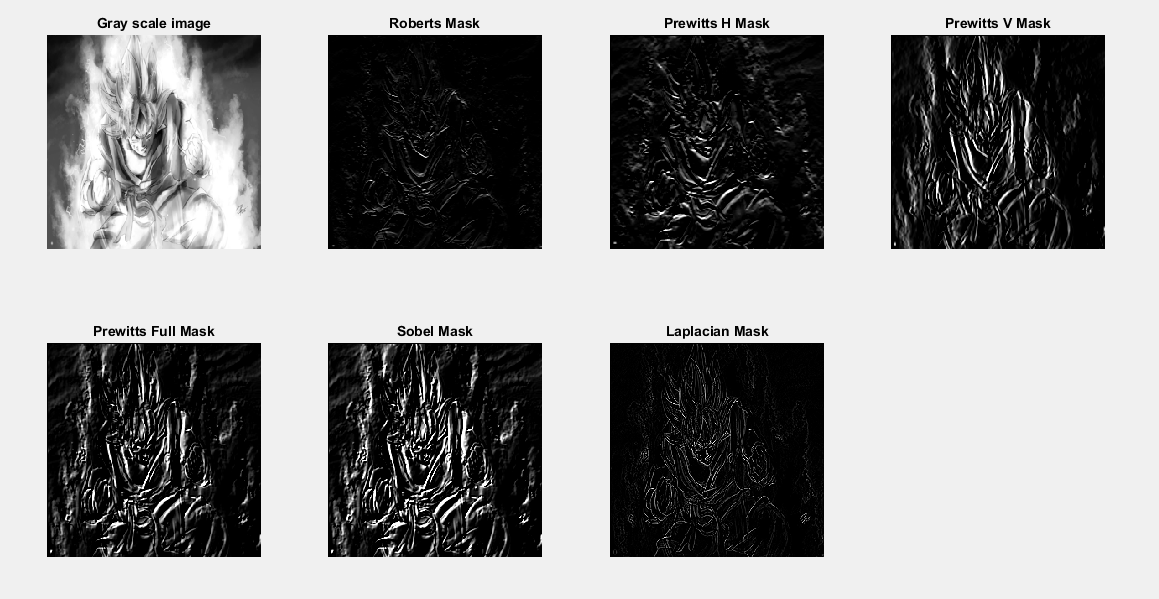
subplot(2,4,7);

imshow(y);

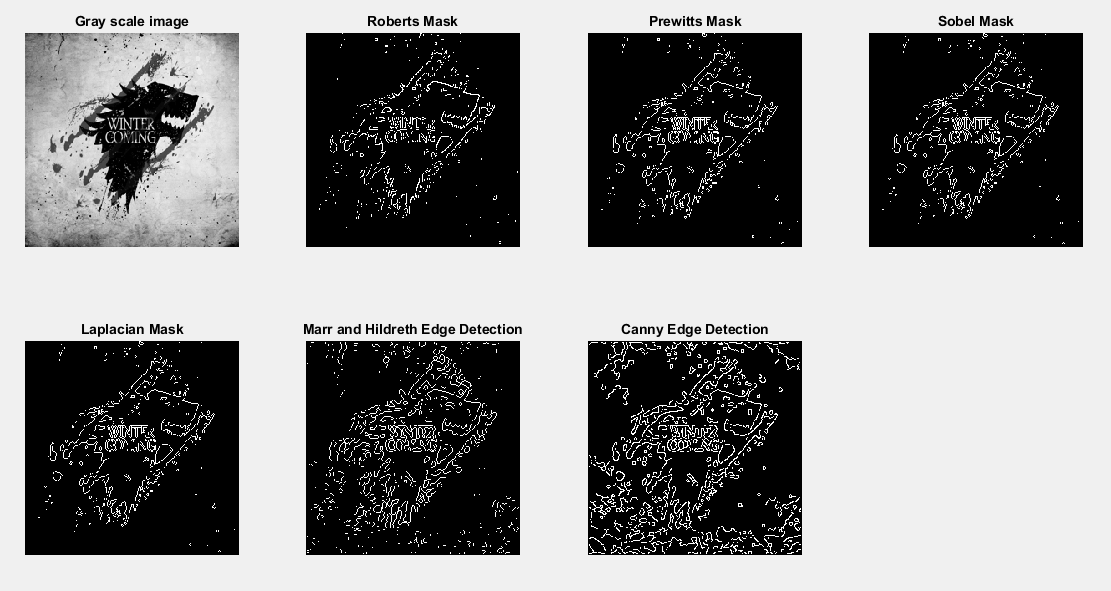
title('Canny Edge Detection');

**Output:**

**Without inbuilt functions**

****

**With inbuilt functions:**

****

**Exp 7: To perform image restoration.**

**Code:**

clc;

close all;

clear all;

a=imread('C:\Users\micromax\Desktop\311824.jpg');

a=im2double(a);

a=imresize(a,[256 256]);

subplot(3,4,1);

imshow(a);

title('original image');

j2=fspecial('motion',21,11);

subplot(3,4,2);

imshow(100\*j2);

title('motion image');

b=imfilter(a,j2);

subplot(3,4,3);

imshow(b);

title('filter image');

c=imfilter(a,j2,'conv','circular');

subplot(3,4,4);

imshow(c);

title('filter2 image');

j4=deconvwnr(c,j2,0);

subplot(3,4,5);

imshow(j4);

title('deconvolution image');

e=imnoise(c,'salt & pepper',0.1);

subplot(3,4,6);

imshow(e);

title('salt and pepper noise image');

f=imnoise(c,'gaussian',0.1);

subplot(3,4,7);

imshow(f);

title('gaussian noise image');

j1=deconvwnr(e,j2,0.8);

subplot(3,4,8);

imshow(j1);

title('deconvolution of salt&pepper image');

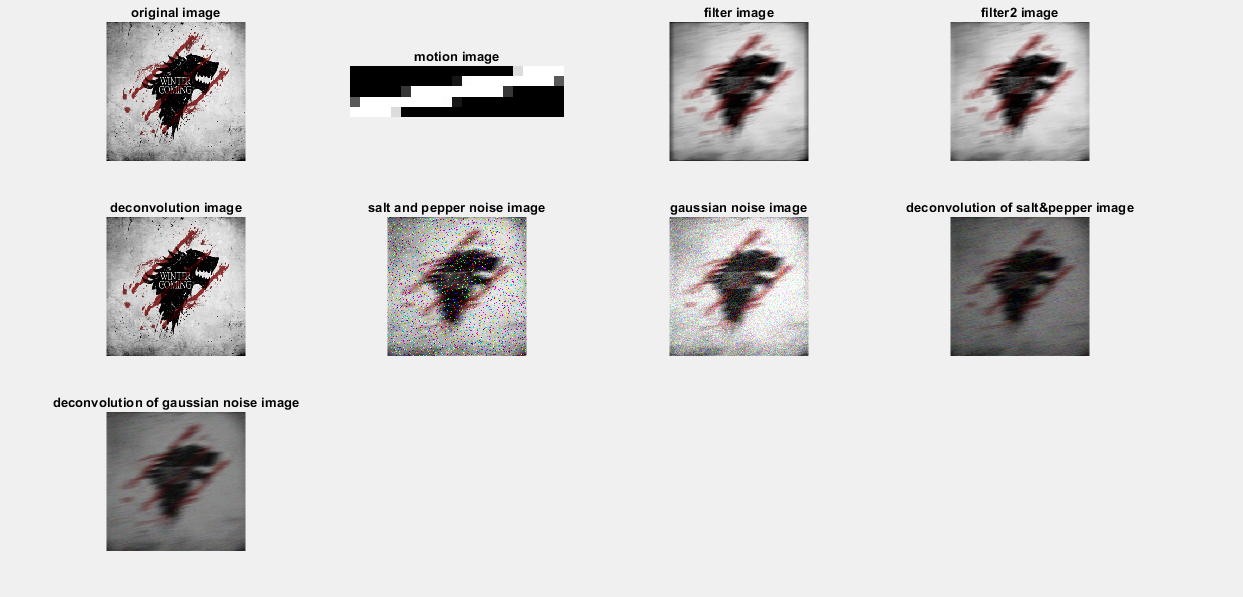
j2=deconvwnr(f,j2,0.8);

subplot(3,4,9);

imshow(j2);

title('deconvolution of gaussian noise image');

**Output:**

****

**Exp 8: To perform Binary classification using support vector machine.**

**Code:**

clc

clear all

close all

a=xlsread('D:\Bkp 07.02.19\Desktop\mu sem 6\ipml\1.xlsx');

means=xlsread('D:\Bkp 07.02.19\Desktop\mu sem 6\ipml\3.xlsx');

sp=xlsread('D:\Bkp 07.02.19\Desktop\mu sem 6\ipml\2.xlsx');

x=a(1:100,:);

y=sp(1:100);

rand\_num=randperm(100);

x\_train=x(rand\_num(1:80),:);

y\_train=y(rand\_num(1:80),:);

x\_test=x(rand\_num(81:end),:);

y\_test=y(rand\_num(81:end),:);

SVMModel=fitcsvm(x\_train,y\_train);

[labels]=predict(SVMModel,x\_test);

sv=SVMModel.SupportVectors;

figure

gscatter(x(:,1),x(:,2),y)

hold on

plot(sv(:,1),sv(:,2),'ko','Markersize',10)

legend('setosa','versicolor','Support Vector')

hold off

idx=(y\_test()==1);

idx1=(y\_test()==2);

p=length(y\_test(idx));

n=length(y\_test(idx1));

N=p+n;

tp=sum(y\_test(idx)==labels(idx));

tn=sum(y\_test(idx1)==labels(idx1));

fp=n-tn;

fn=p-tp;

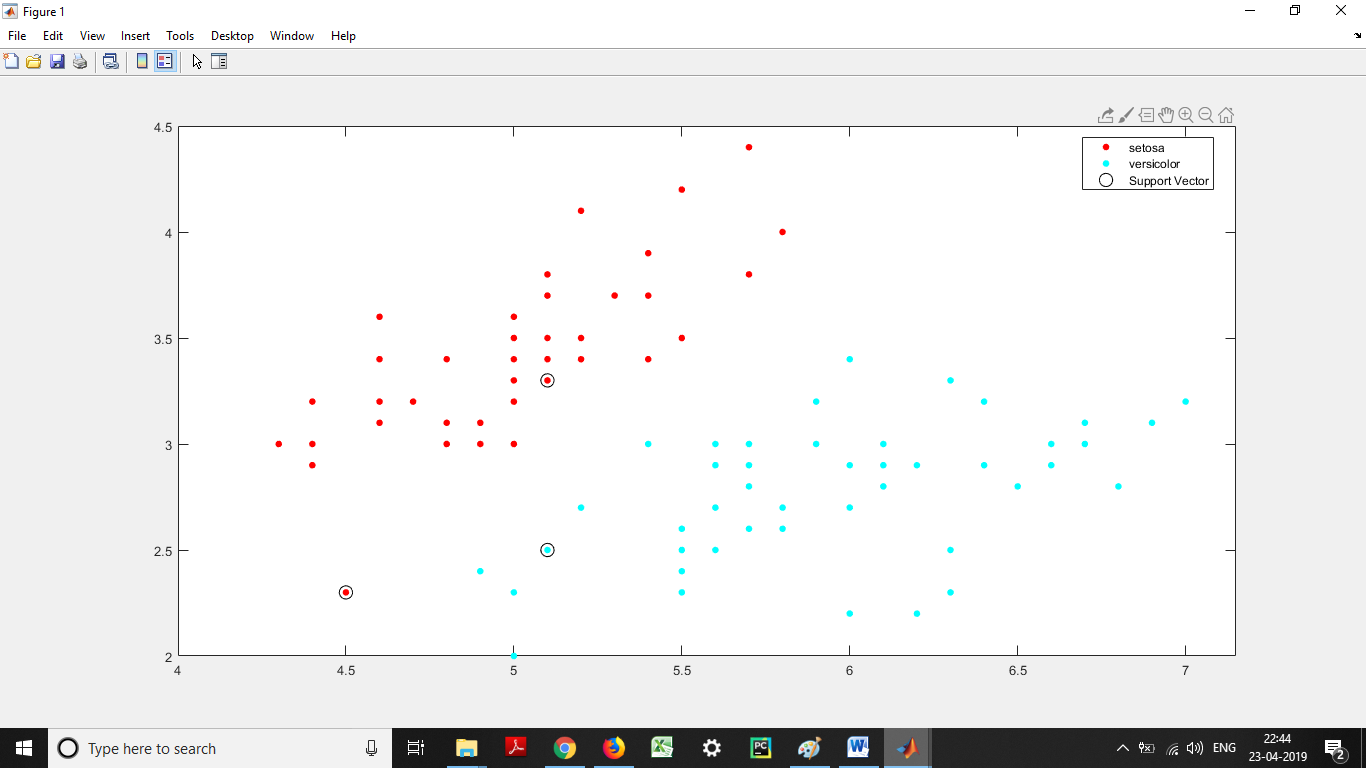
tp\_rate=tp/p;

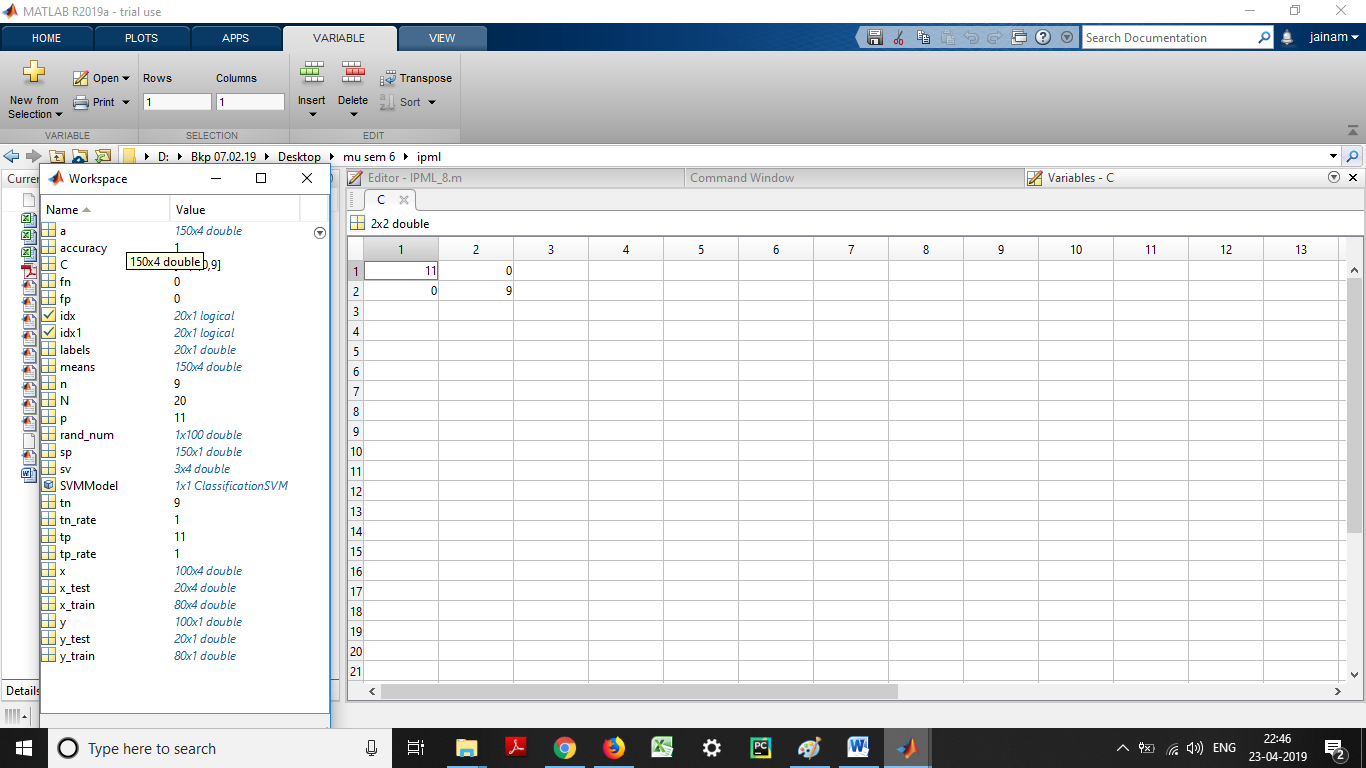
tn\_rate=tn/n;

accuracy=(tp+tn)/N;

C=confusionmat(y\_test,labels);

**Output:**





**Exp 9: To perform clustering using K-means algorithm.**

**Code:**

clc

close all

clear all

data=xlsread('D:\Bkp 07.02.19\Desktop\mu sem 6\ipml\3.xlsx');

data=data(:,3:4);

[ind c]=kmeans(data,3);

figure

for i=1:length(ind)

if ind(i)==1

plot(data(i,1),data(i,2),'k^')

end

if ind(i)==2

plot(data(i,1),data(i,2),'k\*')

end

if ind(i)==3

plot(data(i,1),data(i,2),'k+')

end

hold on

end

plot(c(:,1),c(:,2),'ro')

**Output:**

