**EXP 1a:**

**Read, Access, and Display digital image using MATLAB or SCI Lab**

**Aim:**

 To read, access, and display digital images using MATLAB or Scilab.

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Show the image
6. Stop.

**Program:**

a=imread('F:\lily.jpg');

//figure;

subplot(1,2,1);

title('original image');

imshow(a);

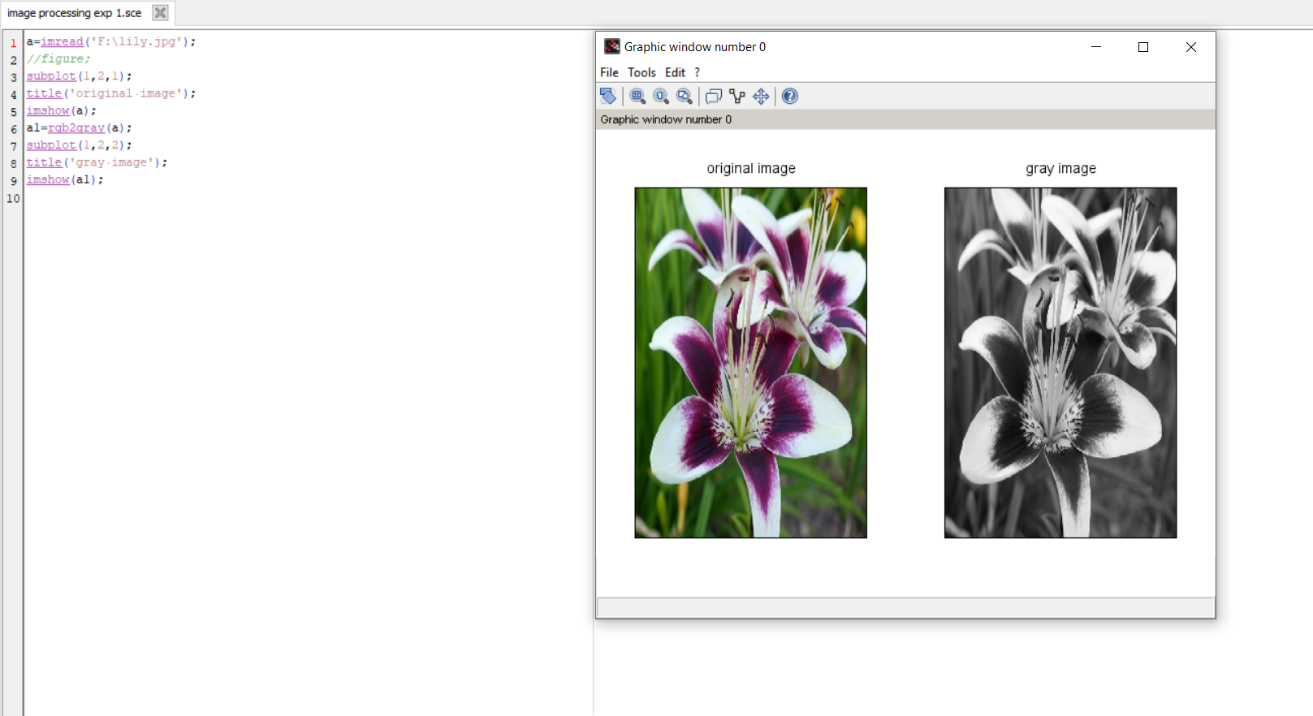
a1=rgb2gray(a);

subplot(1,2,2);

title('gray image');

imshow(a1);

**Output:**

****

**Result:**

The program to read, access, and display digital images using MATLAB or Scilab was executed successfully.

**EXP 1b:**

**Add, Subtract, Multiply and Divide two images using SCI Lab**

**Aim:**

 To add, subtract, multiply and divide two images using Scilab.

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input images from your desktop
5. Add, subtract, multiply and divide the images
6. Show the image
7. Stop.

**Program:**

a=imread('F:\lily.jpg');

a=rgb2gray(a);

a=imresize(a,[100 100]);

subplot(2,3,1)

title('original image of a');

imshow(a);

b=imread('F:\doggo.jpeg');

b=rgb2gray(b);

b=imresize(b,[100 100]);

subplot(2,3,2)

title('original image of b');

imshow(b);

c=imadd(a,b);

subplot(2,3,3)

title('image addition');

imshow(c);

d=imsubtract(a,b);

subplot(2,3,4)

title('subtraction');

imshow(d);

e=immultiply(b,b);

subplot(2,3,5)

title('multiplication');

imshow(uint8(e));

f=imdivide(a,b);

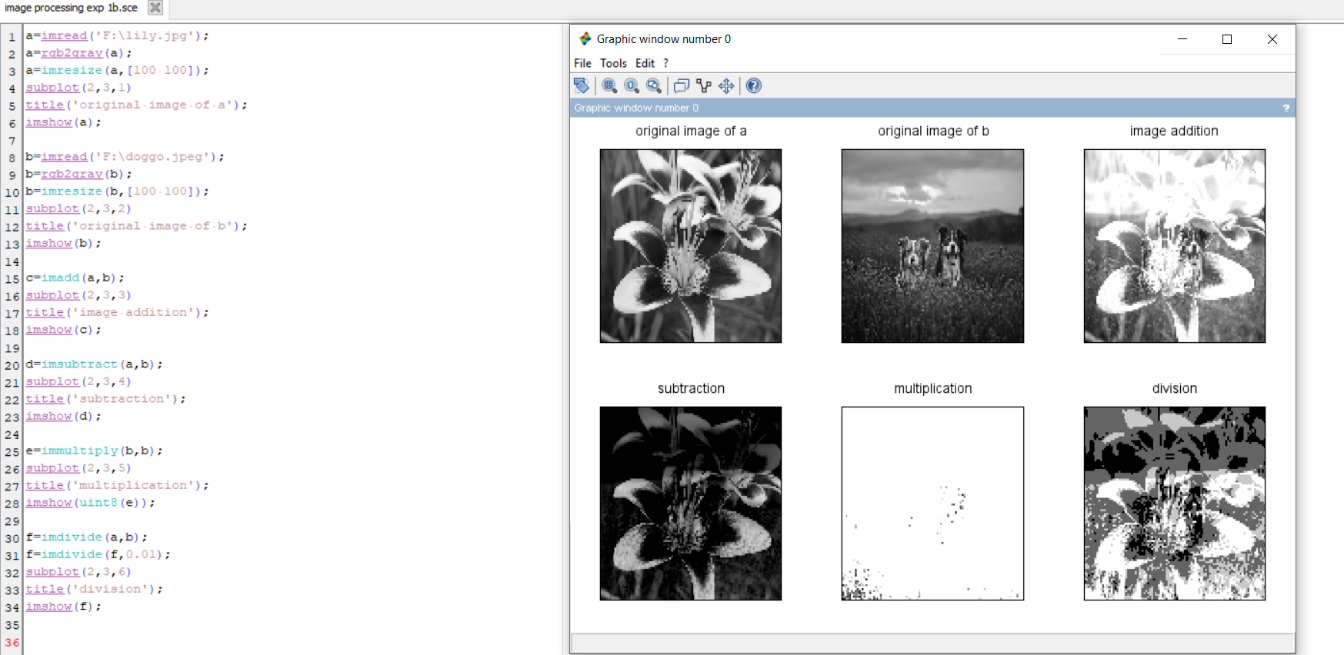
f=imdivide(f,0.01);

subplot(2,3,6)

title('division');

imshow(f);

**Output:**

****

**Result:**

The program to add, subtract, multiply and divide two images using scilab was executed successfully.

**EXP 2:**

**Sampling**

**Aim:**

 To write a program to implement sampling of an image

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Sample the image using various values
6. Show the image
7. Stop.

**Program:**

clc;

n=2;

img = rgb2gray(imread('F:\lily.jpg'));

a=size(img);

w=a(2);

h=a(1);

im=zeros(100);

for i=1:n:h

for j=1:n:w

for k=0:n-1

for l=0:n-1

im(i+k,j+l)=img(i,j);

end

end

end

end

n1=4;

im1=zeros(100);

for i=1:n1:h

for j=1:n1:w

for k=0:n1-1

for l=0:n1-1

im1(i+k,j+l)=img(i,j);

end

end

end

end

n2=8;

im2=zeros(100);

for i=1:n2:h

for j=1:n2:w

for k=0:n2-1

for l=0:n2-1

im2(i+k,j+l)=img(i,j);

end

end

end

end

n3=12;

im3=zeros(100);

for i=1:n3:h

for j=1:n3:w

for k=0:n3-1

for l=0:n3-1

im3(i+k,j+l)=img(i,j);

end

end

end

end

subplot(2,3,1);

imshow(uint8(img));

title('Original Image');

subplot(2,3,2);

imshow(uint8(im));

title('Sampled Image - 2');

subplot(2,3,3);

imshow(uint8(im1));

title('Sampled Image - 4');

subplot(2,3,4);

imshow(uint8(im2));

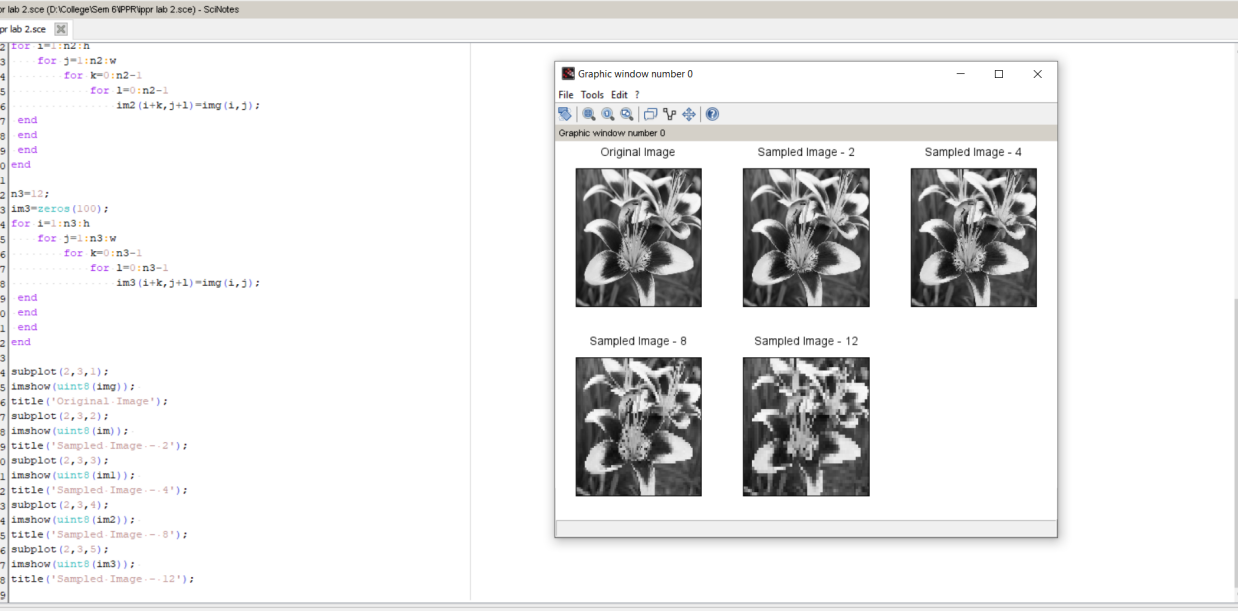
title('Sampled Image - 8');

subplot(2,3,5);

imshow(uint8(im3));

title('Sampled Image - 12');

**Output:**

****

**Result:**

 The program to implement sampling of an image was executed successfully.

**EXP 3a:**

**Neighborhood Metrics**

**Aim:**

 To write a program to learn neighborhood metrics

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement neighborhood metrics
6. Show the image
7. Stop.

**Program:**

img = rgb2gray(imread('F:\lily.jpg'));

imgnoise = imnoise(img, 'salt and pepper');

subplot(2,3,1)

title('Original Image');

imshow(img);

subplot(2,3,2)

title('Noisy Image');

imshow(imgnoise);

flinear1=1/25\*ones(5,5);

imglinear1=imfilter(imgnoise,flinear1);

subplot(2,3,3)

title('Linear Average Filtered');

imshow(imglinear1);

hsize=[5,5];

sigma=1;

flinear2=fspecial('gaussian',hsize,sigma);

imglinear2=imfilter(imgnoise,flinear2);

subplot(2,3,4)

title('Linear Gaussian Filtered');

imshow(imglinear2);

fnonlinear=[3,3];

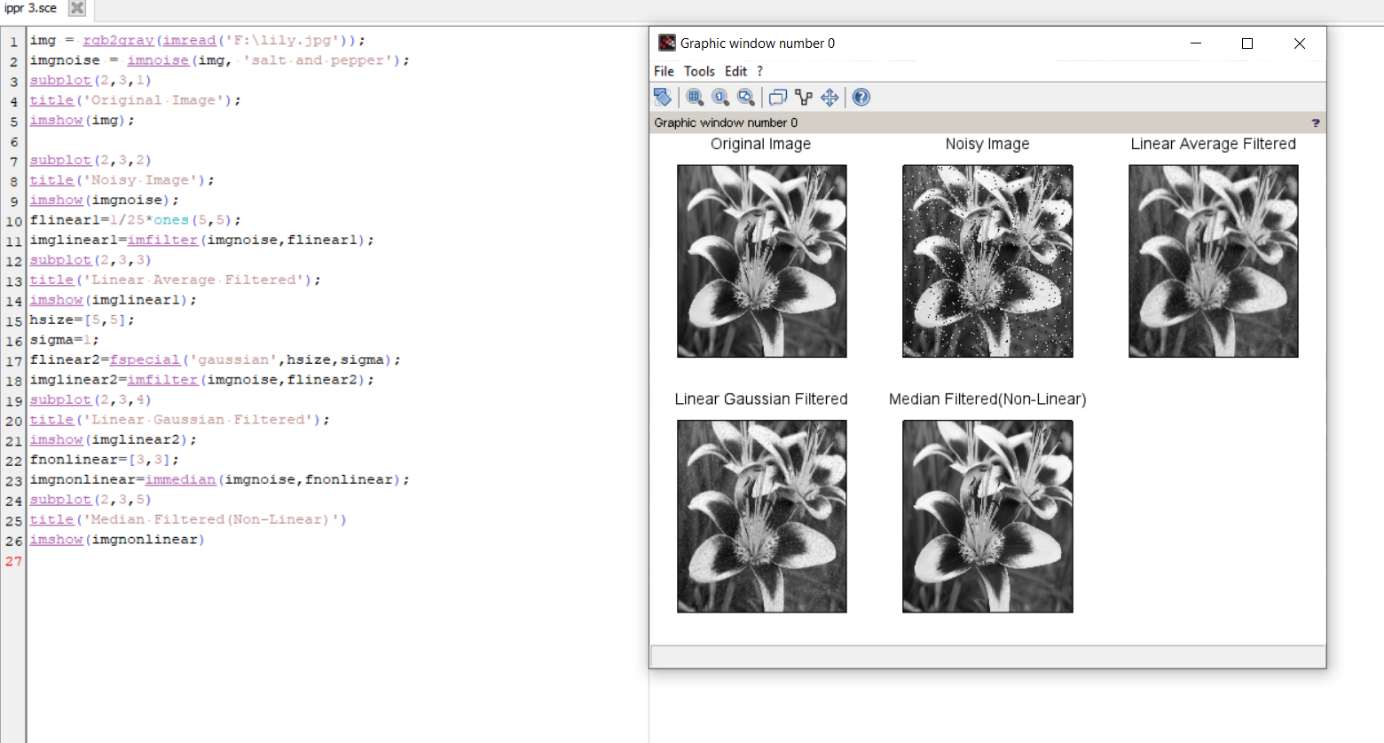
imgnonlinear=immedian(imgnoise,fnonlinear);

subplot(2,3,5)

title('Median Filtered(Non-Linear)')

imshow(imgnonlinear)

**Output:**



**Result:**

Neighborhood metrics program has been executed successfully.

**EXP 3b:**

**Effect of Size of Neighborhood**

**Aim:**

To study the effect of the size of neighborhood on the result of processing

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement neighborhood metrics
6. Show the image
7. Stop.

**Program:**

clc;

clear all;

close all;

I = rgb2gray (imread("F:\lily.jpg"));

I\_noise=imnoise (I, 'salt & pepper');

FilterSize = [3 3];

I\_3x3 = immedian (I\_noise, FilterSize ) ;

I\_5x5 = immedian ( I\_noise, [5 5]);

I\_7x7 = immedian ( I\_noise, [7 7]);

I\_9x9 = immedian ( I\_noise, [9 9]);

subplot (2,3,1);

title ('original image')

imshow (I)

subplot (2,3,2);

title ('noisy-image')

imshow (I\_noise)

subplot (2,3,3);

title('Filter-size-3x3')

imshow (I\_3x3)

subplot (2,3,4);

title('Filter-size-5x5')

imshow (I\_5x5)

subplot (2,3,5);

title('Filter size 7x7')

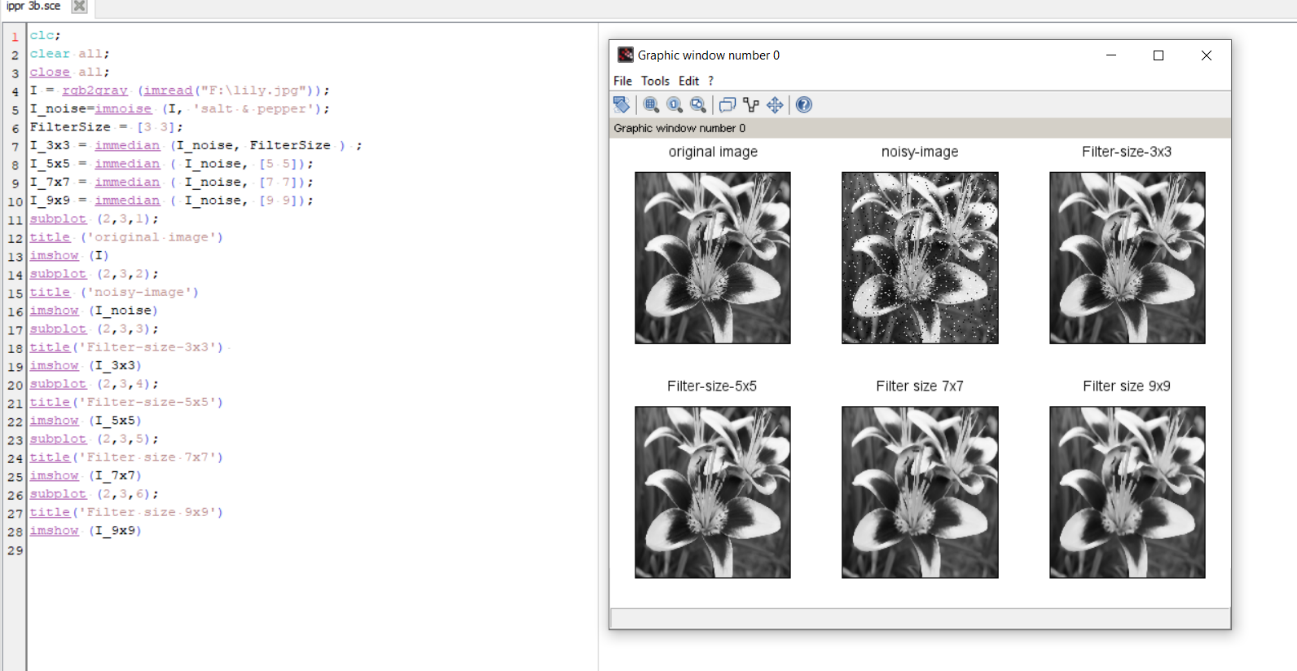
imshow (I\_7x7)

subplot (2,3,6);

title('Filter size 9x9')

imshow (I\_9x9)

**Output:**

****

**Result:**

The program to study the effect of the size of neighborhood on the result of processing

has been executed successfully.

**EXP 4:**

**Image Enhancement**

**Aim:**

 To write a program to enhance the image using scilab through image negative and gamma transformation enhancement.

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement image enhancement processes
6. Show the image
7. Stop.

**Program:**

clc;

clear;

close;

img= imread ("F:\lily.jpg") ;

img=rgb2gray(img) ;

I =im2double(img);

J= imcomplement(I) ;

subplot(2, 3, 1);

title('Original-Image');

imshow (img) ;

subplot(2, 3, 2) ;

title('Image -Negative');

imshow(J) ;

gamma=1.5

k=I.^gamma;

subplot(2, 3, 3) ;

title('Garma-transformation');

imshow(k) ;

contrastl=1./(1+(0.2./(I+%eps)).^4);

contrast2=1./(1+ (0.5./(I+%eps)).^5);

contrast3=1./(1+(0.7./(I+%eps)).^10);

subplot(2, 3, 4) , imshow (contrastl) ;

title ('Contrast enhancement -0.2') ;

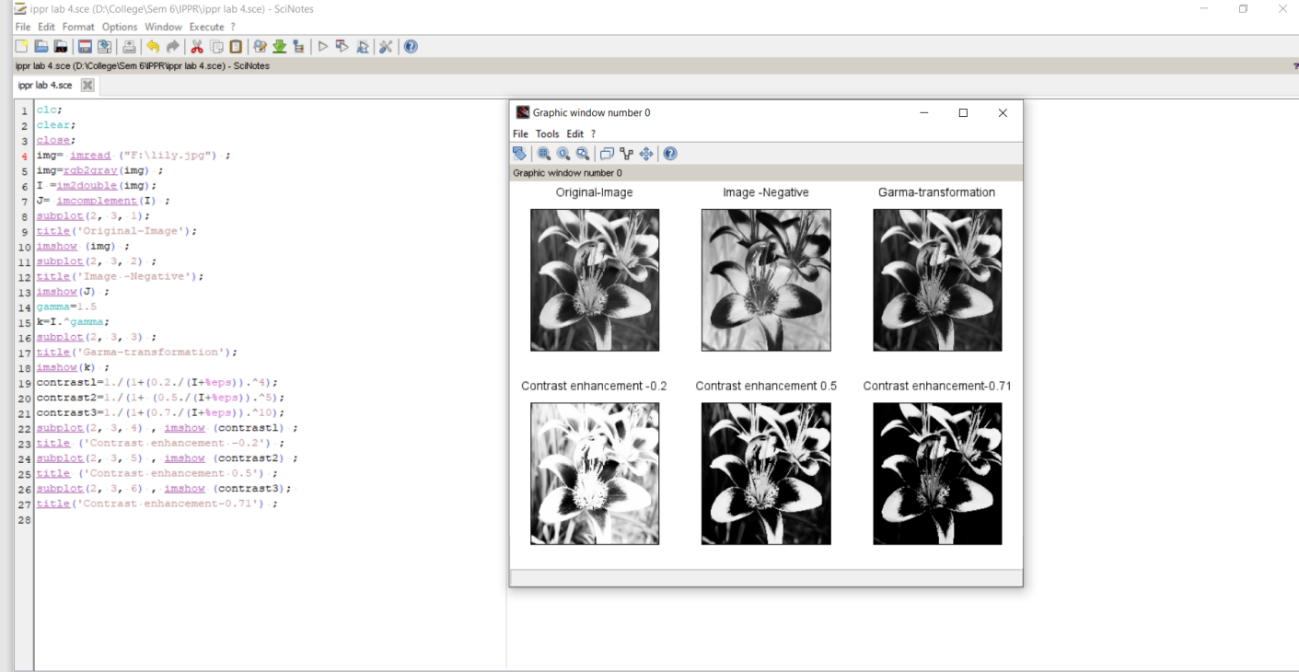
subplot(2, 3, 5) , imshow (contrast2) ;

title ('Contrast enhancement 0.5') ;

subplot(2, 3, 6) , imshow (contrast3);

title('Contrast enhancement-0.71') ;

**Output:**

****

**Result:**

Image enhancement program has been executed successfully

**EXP 5:**

**Histogram**

**Aim:**

To plot histogram,  histogram equalized image and correlate between histogram and histogram equalized of original image.

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement histogram equalization
6. Implement correlation
7. Show the image
8. Stop.

**Program:**

**For Histogram Equalization:**

clc ;

clear;

close;

img=imread ('F:\lily.jpg');

img=rgb2gray(img)

[count, cells ]= imhist (img);

subplot (2,2,1);

title('Original-image');

imshow (img)

subplot (2,2,2);

plot2d3 ('gnn', cells, count)

title('Histogram-plot-for-original-image!');

Iheq = imhistequal (img);

[count, cells ]=imhist (Iheq);

subplot (2,2,3);

title('Histogram-Equalized-image');

imshow (Iheq);

subplot (2,2,4);

plot2d3 ('gnn', cells, count);

title ('Histogram-plot-for-histogram-equalized-image');

**For Historgram Correlation:**

clc ;

clear;

close;

img=imread ('F:\lily.jpg');

img=rgb2gray(img);

[count, cells ]= imhist (img);

subplot (2,2,1)

title('Original-image');

imshow (img);

subplot (2,2,2);

plot2d3 ('gnn', cells, count);

title('Histogram-plot-for-original-image!');

Iheq = imhistequal (img);

[count, cells ]=imhist (Iheq);

subplot (2,2,3);

title('Histogram-Equalized-image');

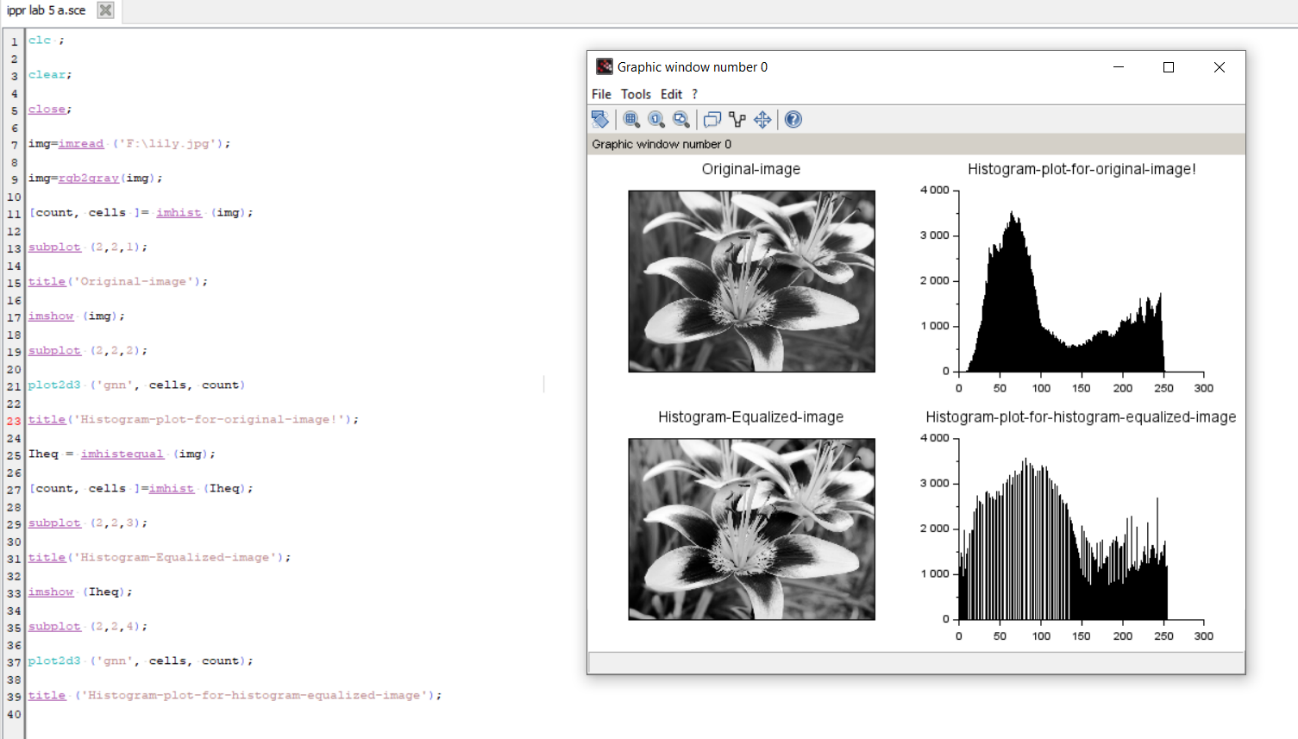
imshow (Iheq)

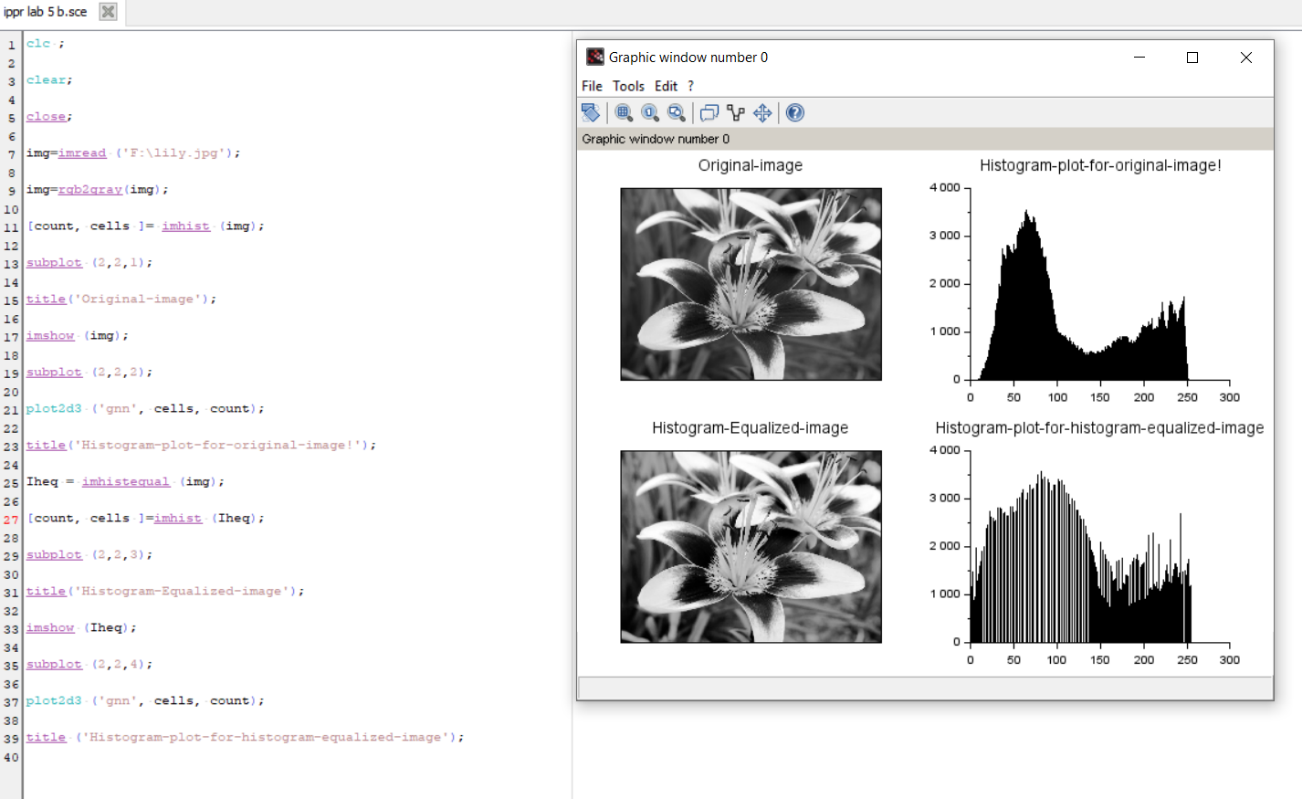
subplot (2,2,4);

plot2d3 ('gnn', cells, count);

title ('Histogram-plot-for-histogram-equalized-image');

**Output:**

****

****

**Result:**

The program to plot histogram, histogram equalized image and correlate between histogram and histogram equalized of original image was executed successfully.

**EXP 6:**

**Smoothing**

**Aim:**

 To implement smoothing in scilab

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement smoothing
6. Show the image
7. Stop.

**Program:**

clc ;

clear ;

close;

I= imread ('F:\lily.jpg');

Iblur1 = imfilter(I,2);

Iblur2 = imfilter(I,4);

Iblur3 = imfilter(I,8);

subplot(2,4,1)

imshow(I)

title('Original image')

subplot(2,4,2)

imshow(Iblur1)

title('Smoothed image, \sigma = 2')

subplot(2,4,3)

imshow(Iblur2)

title('Smoothed image, \sigma = 4')

subplot(2,4,4)

imshow(Iblur3)

title('Smoothed image, \sigma = 8')

IblurX1 = imfilter(I,[4 1]);

IblurX2 = imfilter(I,[8 1]);

IblurY1 = imfilter(I,[1 4]);

IblurY2 = imfilter(I,[1 8]);

subplot(2,4,5)

imshow(IblurX1)

title('Smoothed image, \sigma\_x = 4, \sigma\_y = 1')

subplot(2,4,6)

imshow(IblurX2)

title('Smoothed image, \sigma\_x = 8, \sigma\_y = 1')

subplot(2,4,7)

imshow(IblurY1)

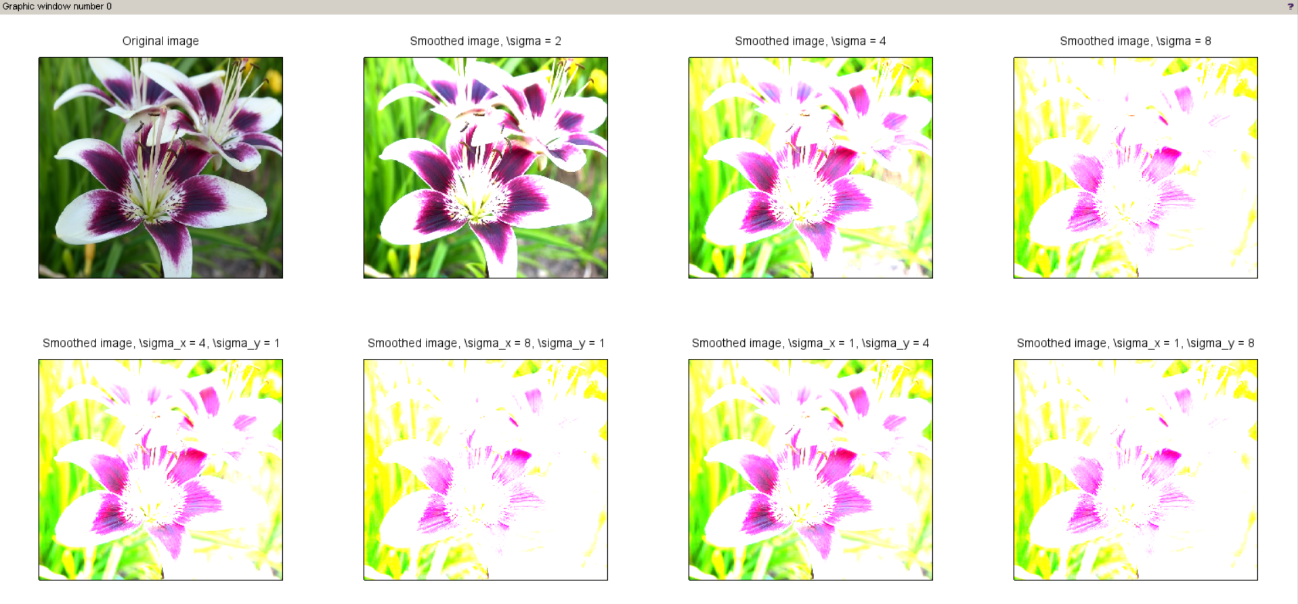
title('Smoothed image, \sigma\_x = 1, \sigma\_y = 4')

subplot(2,4,8)

imshow(IblurY2)

title('Smoothed image, \sigma\_x = 1, \sigma\_y = 8')

**Output:**

****

**Result:**

Implementation of smoothing in scilab done successfully

**EXP 7:**

**Thresholding**

**Aim:**

 To implement thresholding in scilab.

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement thresholding
6. Show the image
7. Stop.

**Program:**

img= imread("F:\lily.jpg");

img=rgb2gray(img);

threshold =0.2;

imagen1=~im2bw(img, threshold);

subplot(2,2,1)

title ('threshold=0.1');

imshow (imagen1)

threshold =0.3;

imagen2=~im2bw(img,threshold);

subplot(2,2,2)

title('threshold = 0.2');

imshow(imagen2)

threshold =0.4,

imagen3=~im2bw(img ,threshold);

subplot(2,2,3)

title('threshold=0.3');

imshow(imagen3)

threshold=0.5;

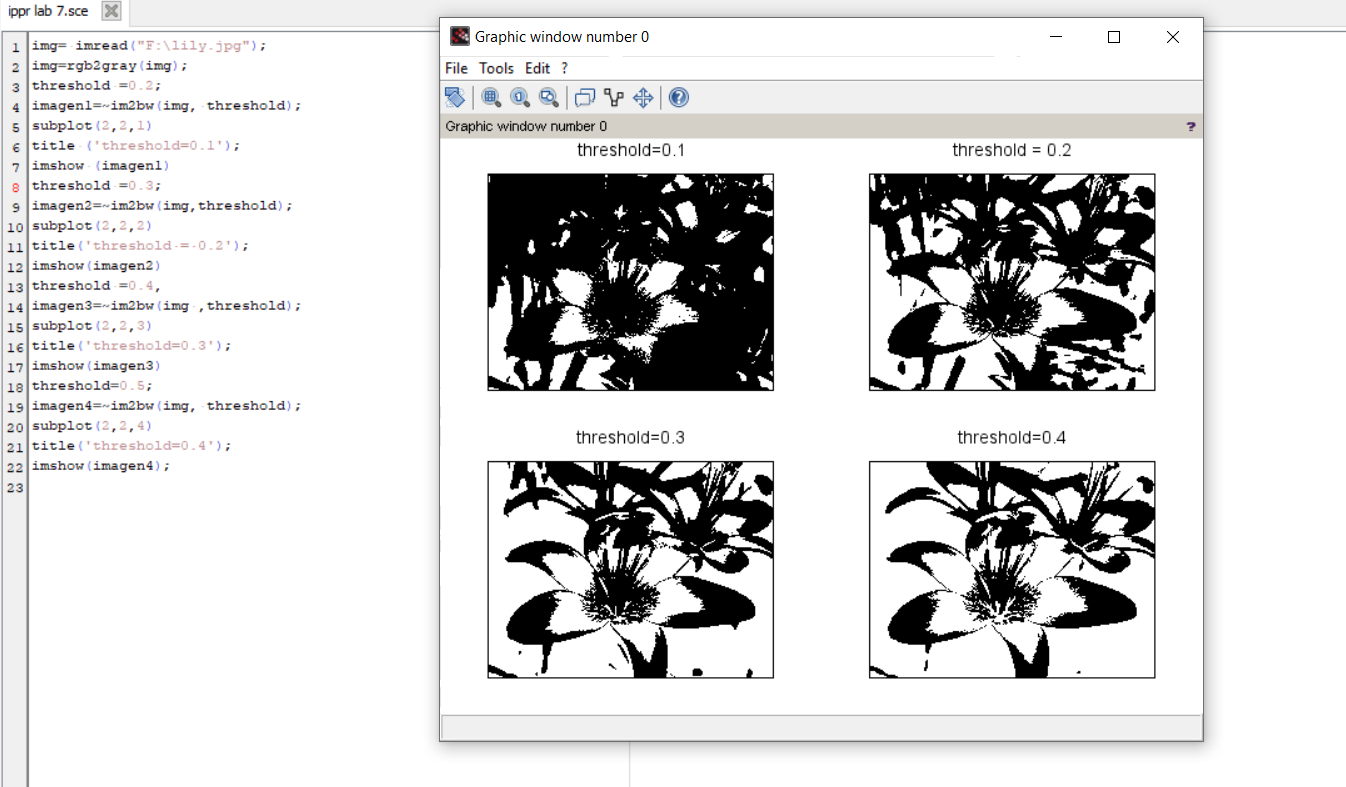
imagen4=~im2bw(img, threshold);

subplot(2,2,4)

title('threshold=0.4');

imshow(imagen

**Output:**

****

**Result:**

Implementation of thresholding in scilab done successfully.

**EXP 8:**

**Edge Detection**

**Aim:**

 To implement edge detection in scilab

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement edge detection
6. Show the image
7. Stop.

**Program:**

//edge detection

img= imread('F:\lily.jpg');

img=rgb2gray( img );

subplot(2,3,1)

imshow (img);

e=edge (img );

//Opens new figure

subplot(2,3,2)

imshow (e)

e=edge (img, 'prewitt');

subplot(2,3,3)

imshow (e)

e=edge (img,'canny',[0.06 0.2]);

// Applies canny e detection method

subplot(2,3,4)

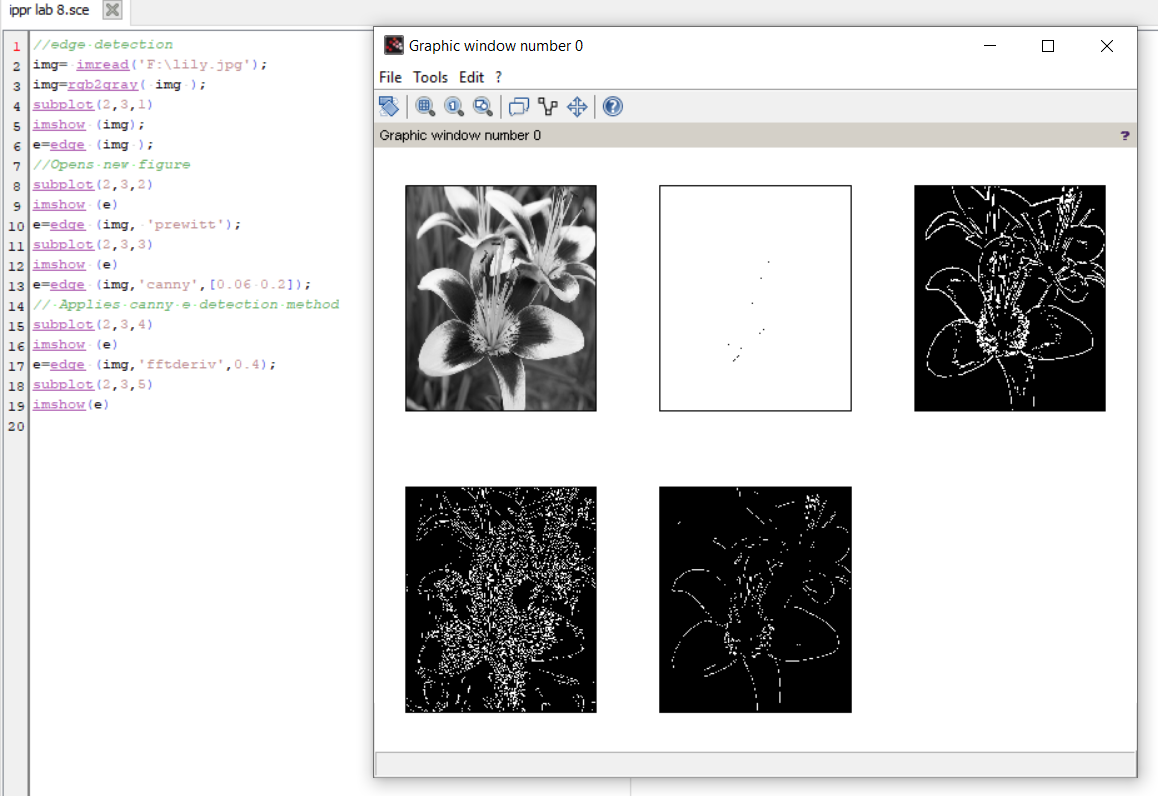
imshow (e)

e=edge (img,'fftderiv',0.4);

subplot(2,3,5)

imshow(e)

**Output:**

****

**Result:**

Implementation of edge detection in scilab done successfully

**EXP 9:**

**Hough Transformation**

**Aim:**

 To implement Hough transformation in scilab

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement Hough transformation
6. Show the image
7. Stop.

**Program:**

img = imread ('F:\lily.jpg');

img = rgb2gray (img);

subplot(2,2,1);

imshow ( img ) ;

e = edge ( img ,'canny', [0.06 0.2]) ;

// A p p l i e s canny e dg e d e t e c t i o n method

subplot(2,2,2);

imshow ( e)

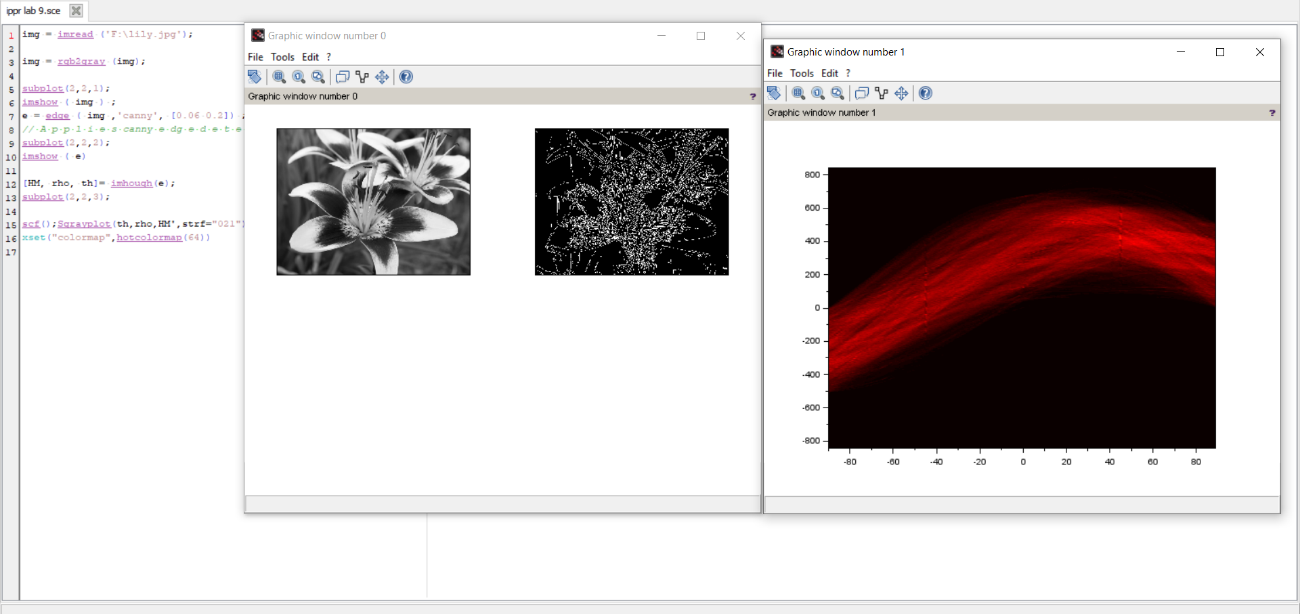
[HM, rho, th]= imhough(e);

subplot(2,2,3);

scf();Sgrayplot(th,rho,HM',strf="021");

xset("colormap",hotcolormap(64))

**Output:**

****

**Result:**

 Implementation of Hough transformation in scilab done successfully

**EXP 10:**

**Feature Extraction**

**Aim:**

 To implement feature extraction in scilab

**Procedure:**

1. Start
2. Open scilab
3. Open a new console window
4. Get the input image from your desktop
5. Implement feature extraction
6. Show the image
7. Stop.

**Program:**

S= imread ('F:\lily.jpg' );

S = rgb2gray (S) ;

S2 = imrotate(S,45);

// Use the ORB to detect features

f1 = imdetect\_ORB(S)

f2 = imdetect\_ORB(S2)

// Extract the descriptor

d1 = imextract\_DescriptorORB(S,f1);

d2 = imextract\_DescriptorORB(S2,f2);

// Feature matching

m = immatch\_BruteForce(d1,d2,4)

// Find the 10 best matches

[fout1,fout2,mout] = imbestmatches(f1,f2,m,10);

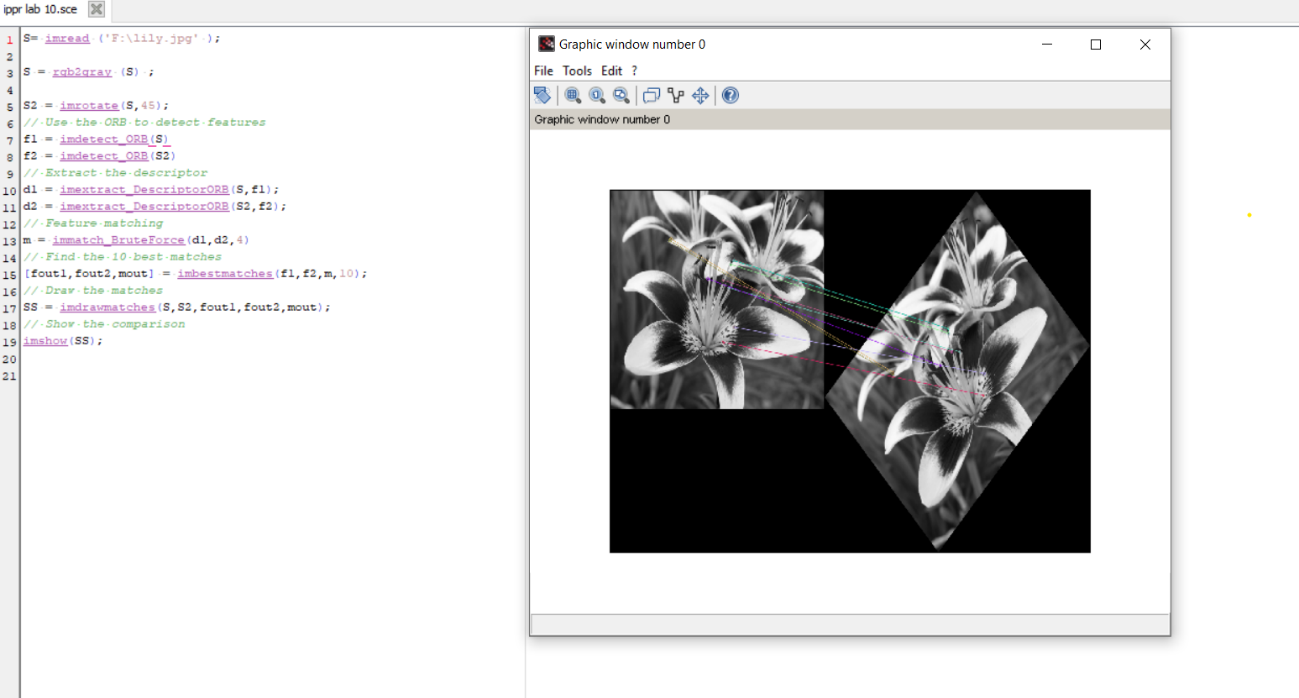
// Draw the matches

SS = imdrawmatches(S,S2,fout1,fout2,mout);

// Show the comparison

imshow(SS);

**Output:**

****

**Result:**

 Implementation of feature extraction in scilab done successfully