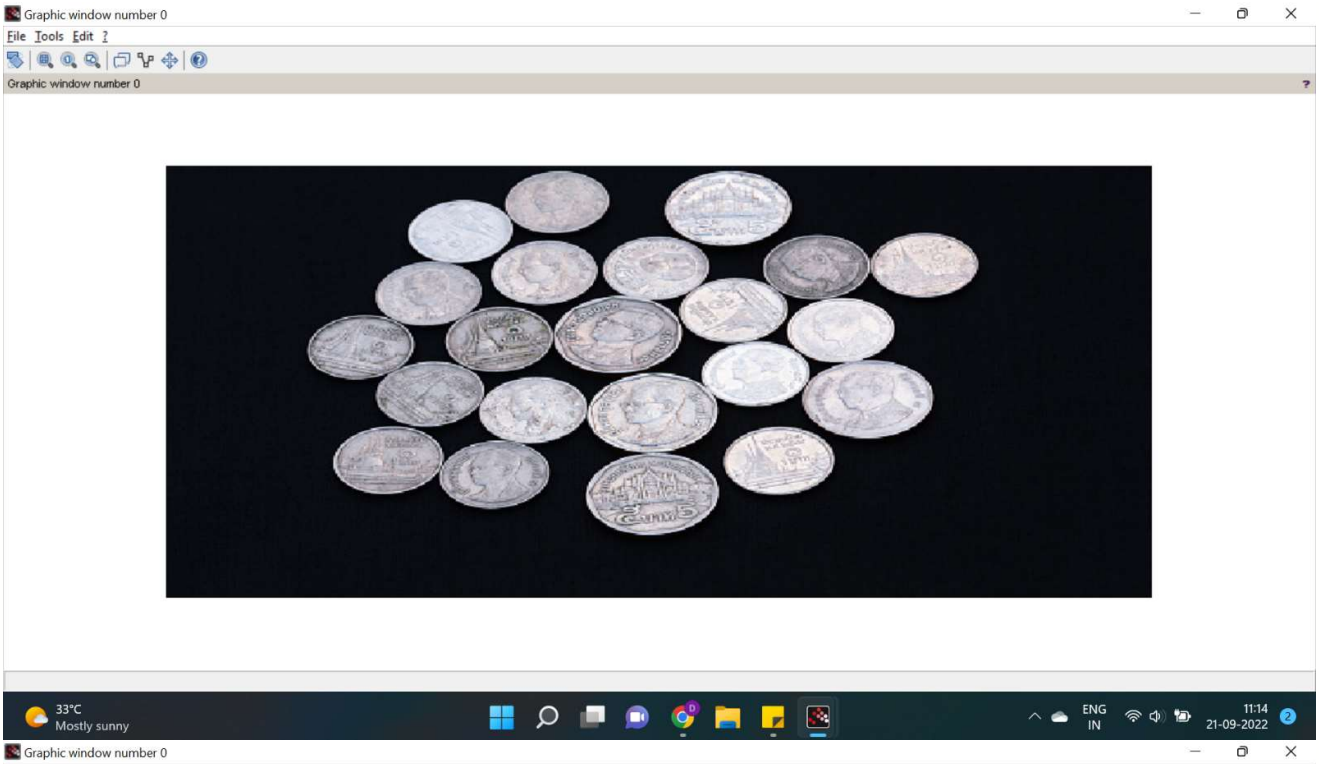
LAB-8

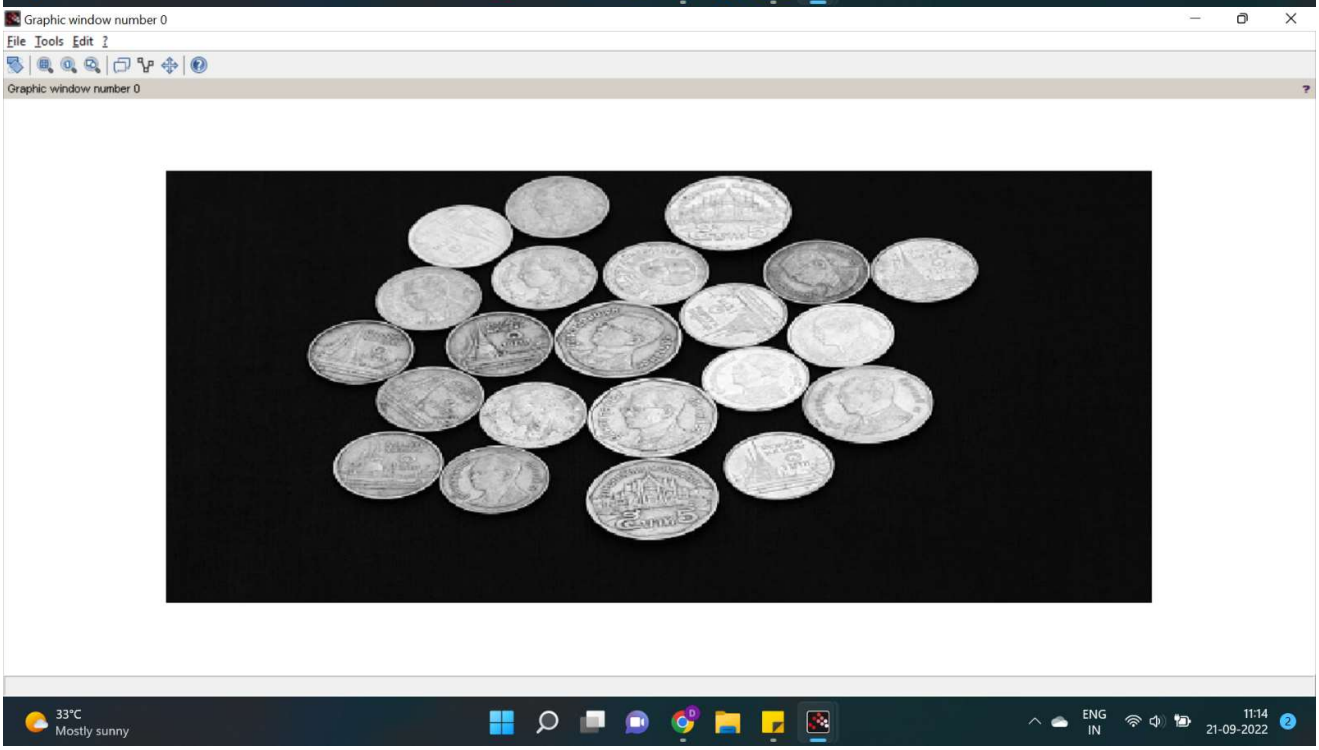
Aim: To perform the following operations in an image. (a) erosion, (b) dilation,

Code:

```
--> S = imread("C:\Users\Diksha Nasa\Desktop\Study Material\IT Workshop using scilab\coins.jpg");
-->imshow(S);
-->Sgray = rgb2gray(S);
-->imshow(Sgray);
-->Sbin = im2bw(Sgray,0.5);
-->imshow(Sbin);
-->imhist(Sgray,[],1);
-->Sbin = im2bw(Sgray,0.353);
-->imshow(Sbin);
-->th = imgraythresh(Sgray)
th  =


  0.4140625


-->Sbin = im2bw(Sgray,th);
-->imshow(Sbin);
--> Sd = imdilate(Sbin,se);
-->imshow(Sd);
--> Se = imerode(Sbin,se);


-->scf();imshow(Se);
```

Output:

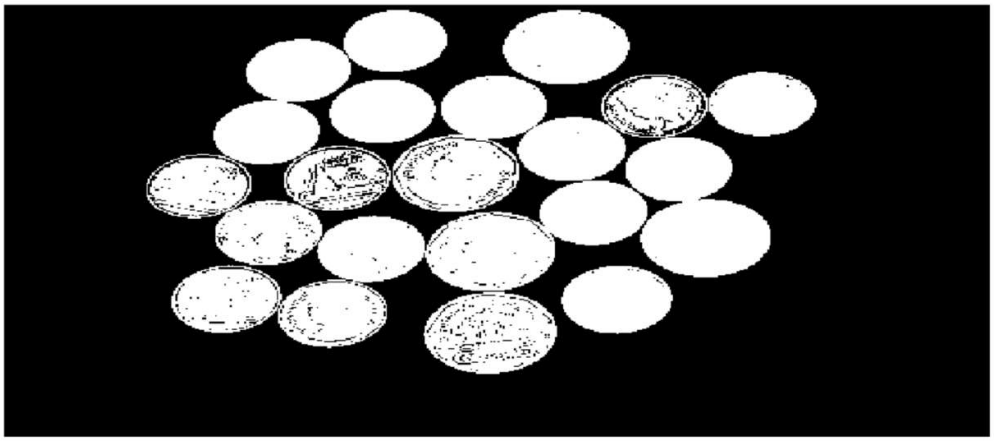

Graphic window number 0
File Tools Edit ?
Graphic window number 0
33°C
Mostly sunny
ENG
IN
11:14
21-09-2022


Graphic window number 0
File Tools Edit ?
Graphic window number 0
33°C
Mostly sunny
ENG
IN
11:14
21-09-2022

Graphic window number 0

File Tools Edit ?

Graphic window number 0



33°C
Mostly sunny

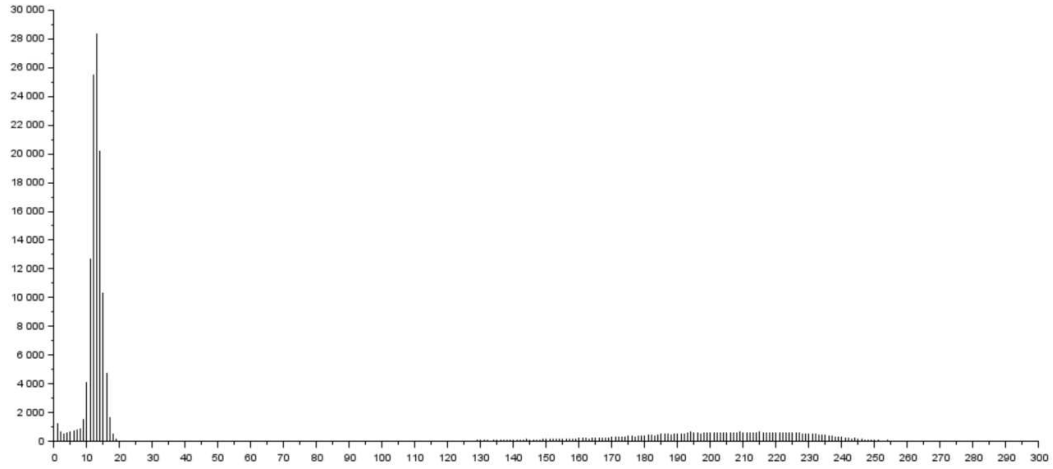
ENG
IN

11:16
21-09-2022

Graphic window number 0

File Tools Edit ?

Graphic window number 0

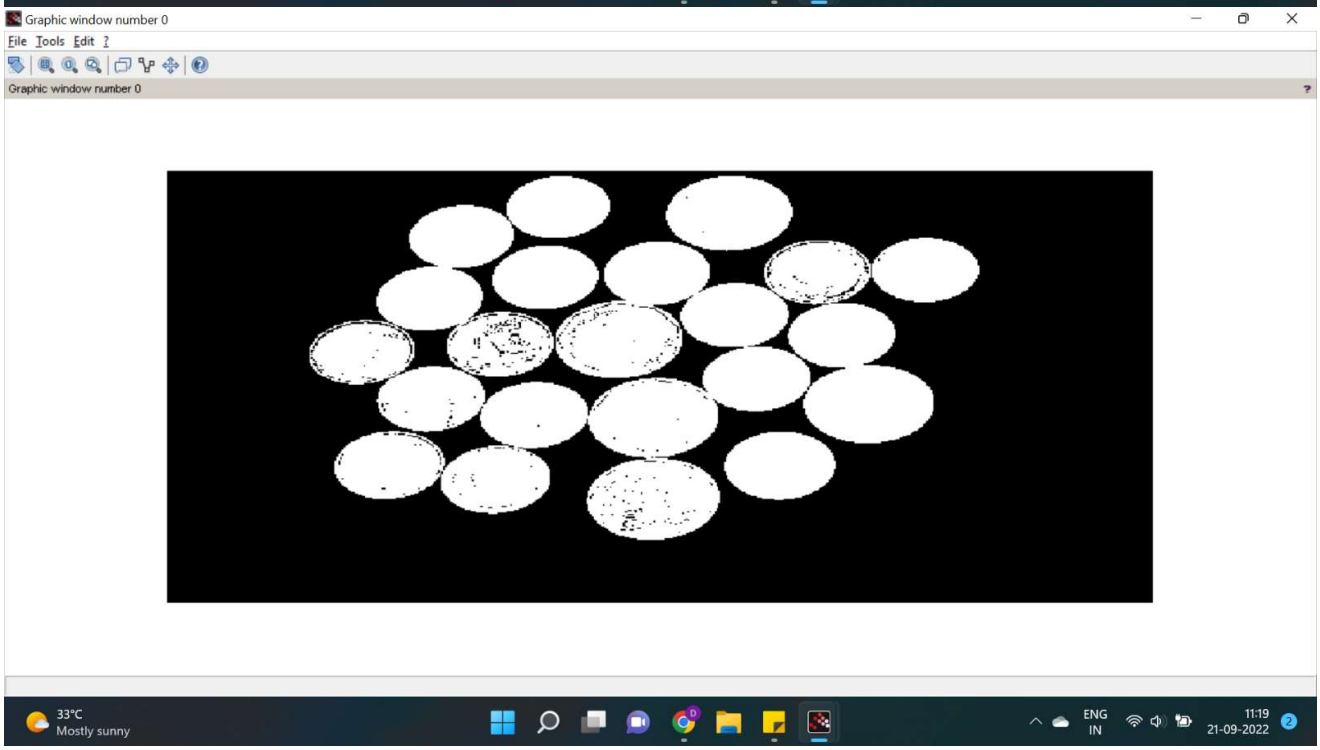


33°C
Mostly sunny

ENG
IN

11:16
21-09-2022


Graphic window number 0
File Tools Edit ?
Graphic window number 0
33°C
Mostly sunny
ENG
IN
11:19
21-09-2022


Graphic window number 0
File Tools Edit ?
Graphic window number 0
33°C
Mostly sunny
ENG
IN
11:19
21-09-2022


Graphic window number 0
File Tools Edit ?
Graphic window number 0
33°C
Mostly sunny
ENG
IN
11:20
21-09-2022


Graphic window number 0
File Tools Edit ?
Graphic window number 0
33°C
Mostly sunny
ENG
IN
11:21
21-09-2022

Experiment 5: Perform perform the Linear filtering using convolution in an image

AIM:

To Perform the Linear filtering using convolution in an image.

SOURCE CODE:

```
S = imread('balloons_noise.png');
imshow(S);
h = 1/25.*ones(5,5);
S2 = imfilter(S,h);
imshow(S2);

S = imread('text.png');
imshow(S);
S2 = imcomplement(S);
imshow(S2);

h = imread('a.png');
h2 = imcomplement(h);
imshow(h2);

S3 = im2double(S2);
h3 = im2double(h2);
S4 = imfilter(S3,h3);
S5 = imnorm(S4);
imshow(S5);

a_loc = S5 > 0.9;
[rows, cols] = find(a_loc);
imshow(S);
sz = size(S);
plot(cols,sz(1)-rows,'r.');
```

OUTPUT:

RESULT:

We have successfully performed the Linear filtering using convolution in an image.

2D FOURIER TRANFORMATION IN AN IMAGE

Exp: 4b

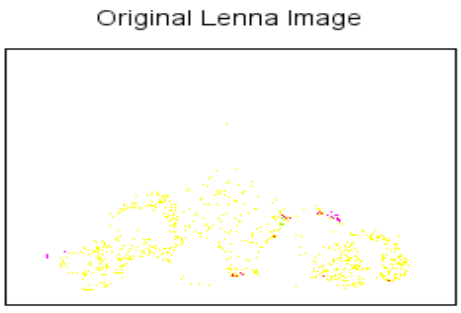
Objective:

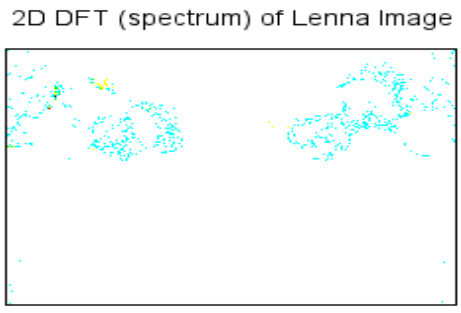
To perform the two dimensional Fourier transform operation in an image using SCILAB.

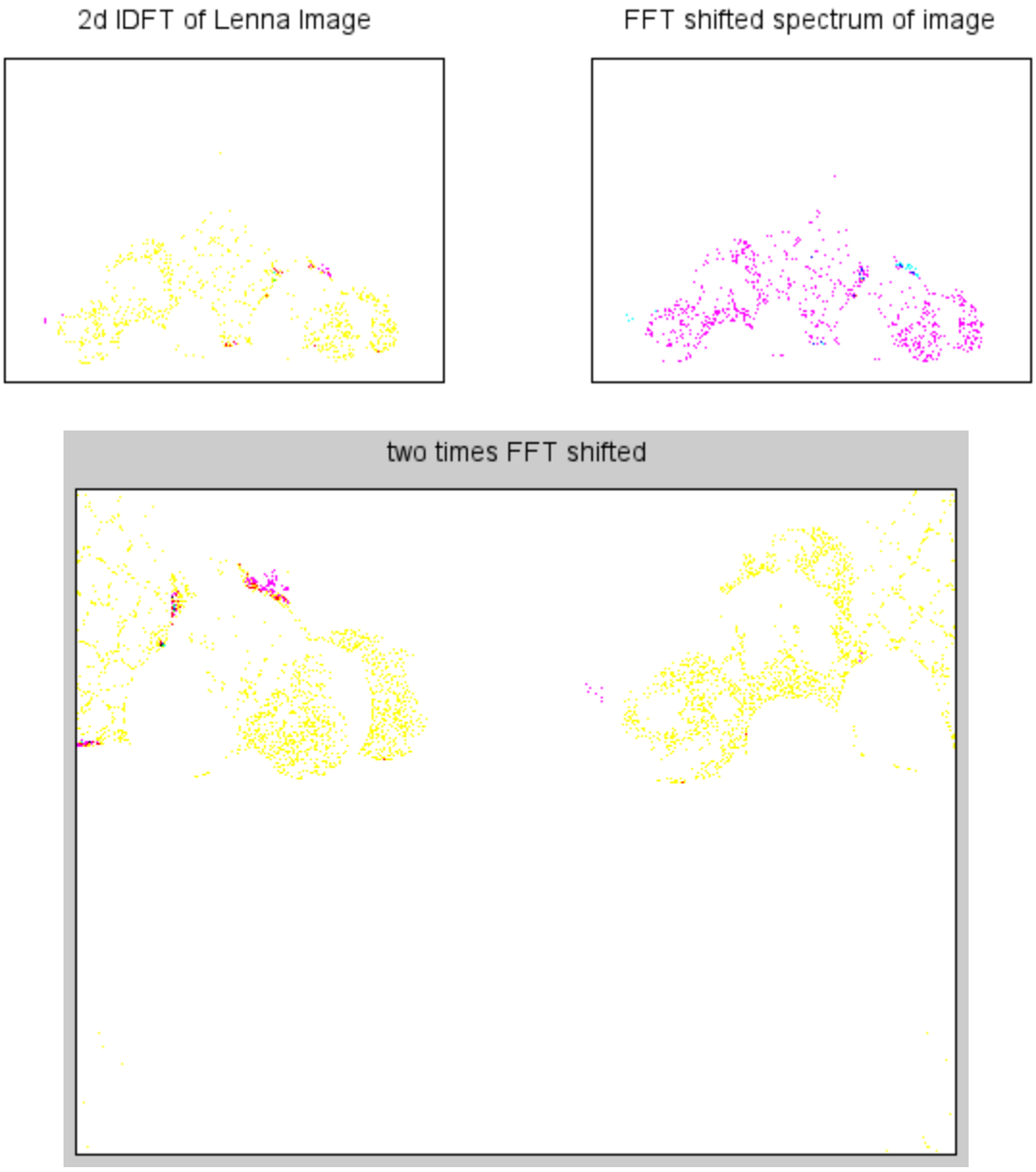
Source Code:

```
clc;
clear all;
close all;
I = imread('Fruit.jpg');
// [ 1 ] . 2D-DFT and i t s I n v e r s e 2D-DFT
I = double (I);
J = fftshift(I);
K = real(ifftshift(J));
subplot(2,2,1);
imshow(I);
title('Original Lenna Image');
subplot(2,2,2);
imshow(abs(J));
title('2D DFT (spectrum) of Lenna Image');
subplot(2,2,3);
imshow(K);
title('2d IDFT of Lenna Image');
L = fftshift (J);
M = fftshift (L);
subplot(2,2,4);
imshow(abs(L));
title('FFT shifted spectrum of image');
figure,
imshow(abs(M));
title('two times FFT shifted');
```

OUTPUT:







RESULT:

Thus, we have executed the basic two dimensional Fourier Transform operation in an image using Scilab successfully.

AFFINE TRANSFORMATION

EXP. NO: 3.

OBJECTIVE:

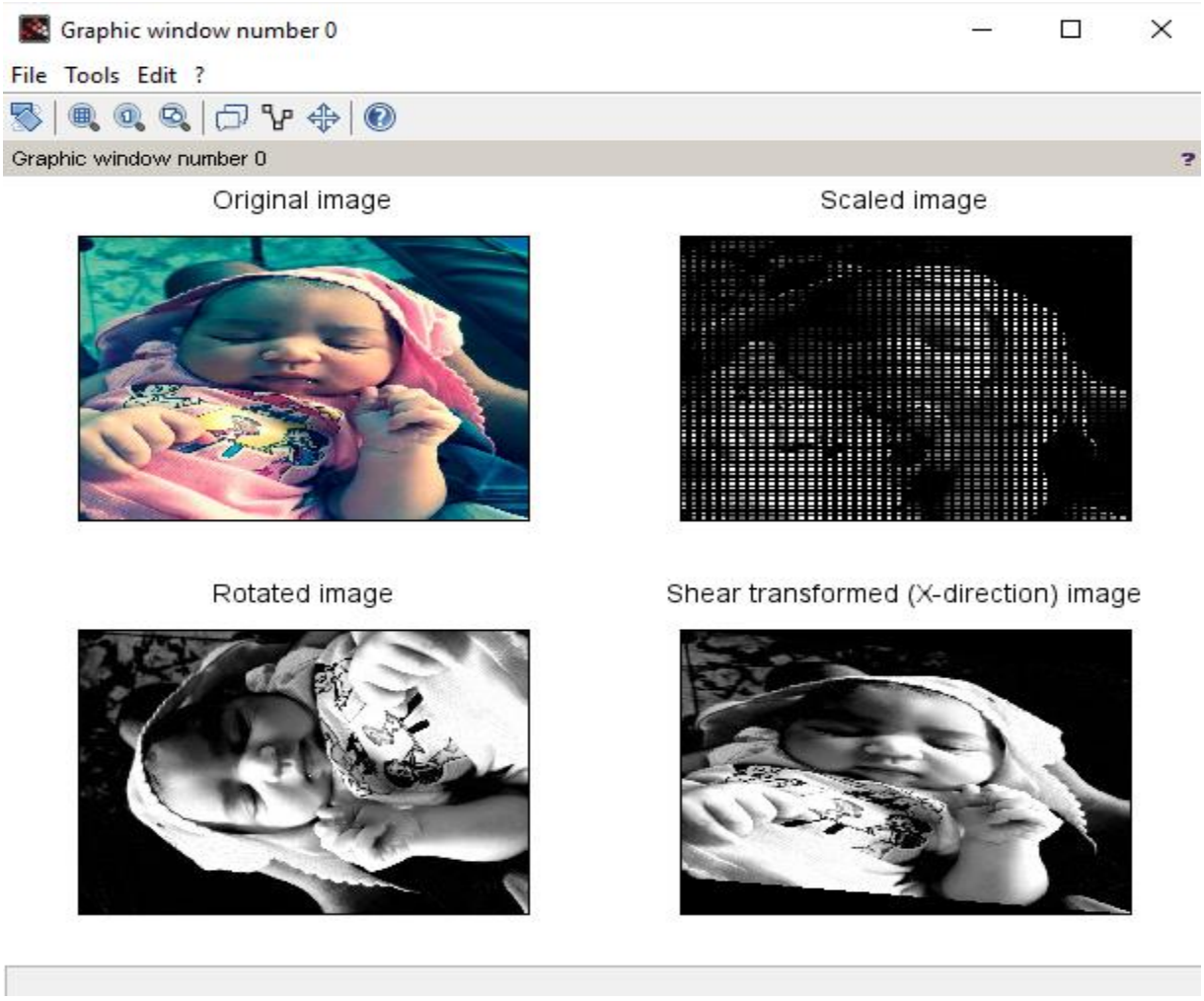
To learn basic image rotation, scaling and transformation using geometric transformations in Scilab.

CODE:

```
clc ;
clear; close;
I =imread('Baby.jpg');
[m,n] = size (I);
for i = 1:m
for j =1: n
// S c a l i n g
J(2*i,2*j) = I(i,j);
// Ro t a t i on
p = i*cos(%pi/2)+j*sin(%pi/2);
q = -i*sin(%pi/2)+j*cos(%pi/2);
p = ceil(abs(p)+0.0001);
q = ceil(abs(q)+0.0001);
K(p,q)= I(i,j);
// s h e a r t r a n s f o rma t i o n
u = i +0.2*j;
v = j;
L(u,v)= I(i,j);
end
end
subplot(2,2,1);
title('Original image');
imshow(I);
subplot(2,2,2);
title('Scaled image');
imshow(J);
subplot(2,2,3);
```

```
title('Rotated image');
imshow(K);
subplot(2,2,4);
title('Shear transformed (X-direction) image');
imshow(L);
```

SAMPLE OUTPUT:



RESULT:

Thus, the basic image rotation, scaling and transformation are executed successfully.

IMAGE HISTOGRAM USING SCILAB

EXP. NO: 2a.

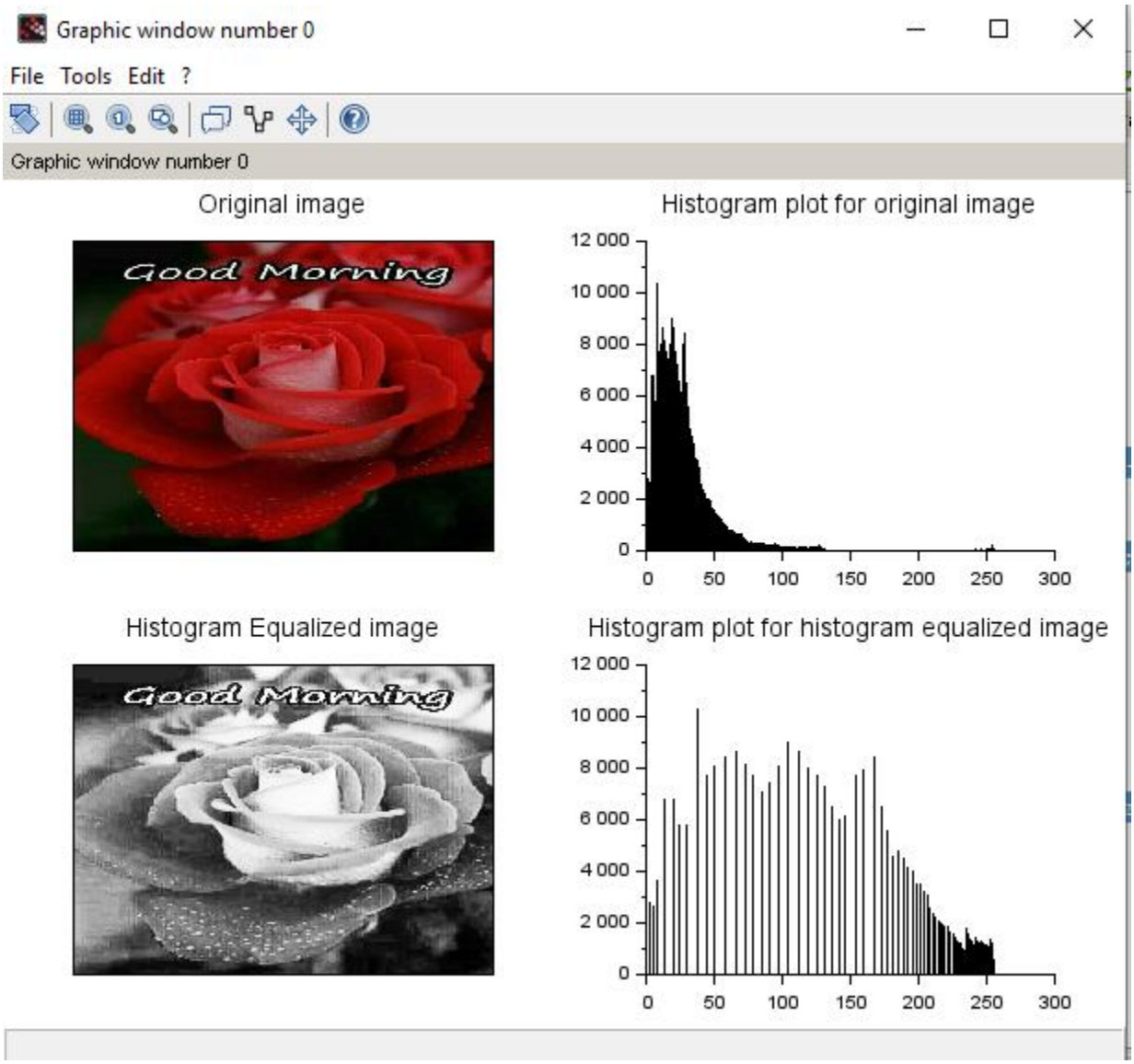
OBJECTIVE:

To understand how frequency distribution can be used to represent an image.

CODE:

```
clc ;
clear;
close;
img= imread ('rose.jpeg');
subplot(2,2,1);
title('Original image');
imshow(img);
img=rgb2gray(img);
[count,cells ]= imhist (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');
```

SAMPLE OUTPUT:



Result:

Thus the frequency distribution and correlation between the images using histogram has been executed successfully.

IMAGE THRESHOLDING

EXP. NO: 2b.

OBJECTIVE:

To find the threshold of an image and segment it.

CODE:

```
RGB = imread ("Cars.jpg");
Image = rgb2gray(RGB);
InvertedImage = uint8(255 * ones(size(Image,1), size(Image,2))) - Image;
Histogram=imhist(InvertedImage);
figure();plot(0:255, Histogram')
xgrid(color('black'),1,8)
LogicalImage = im2bw(InvertedImage, 100/255);
f1=scf(1);f1.name='Original Image';
imshow(Image);
f2=scf(2);f2.name='Inverted Image';
imshow(InvertedImage);
f3=scf(3);f3.name='Result of Thresholding';
imshow(LogicalImage);
```

RESULT:

Image was segmented successfully according to threshold value.

STUDY OF BASIC SCILAB COMMANDS

EXP. NO: 1

i. OBJECTIVE: Practicing MATLAB environment with simple exercises to familiarize Command Window, History, Workspace, Current Directory, Figure window, Edit window, Shortcuts, Help files.

ii. SOURCE CODE :

Working on general commands on scilab environment

Help – detailed help menu for scilab commands
Who – list all the variables from the variable browser window
Whos - list all the variables with byte size, variable type etc.
Clc – Clears command window
Clear – Removes the variable from memory
Quit – to close the session
Pwd – present working directory
Ls – list the files
Ls -ltr – list the detailed view on the files
Cd - to change the directory
Mkdir – to create a new directory

To work on Special variables / Pre-Defined Variables
%pi – 3.14
Ans
% e = 2.718
%eps – epsilon
%inf – infinity

Basic Scalar & Vector Operations

Creation of Scalar Elements
Y = [1 4 6] ➔ Declares a row vetor
yT = [1; 4; 6] ➔ Declares a Column vector
Creation of Vector Elements
Y = [1 4 6; 2 7 3; 4 1 1]
➔ Creates a 3*3 matrix

To determine the size / order of the vectors.
Size(y)

To change the elements in the given vector
vector(i,j) = value

Performing element by element operations using dot operator
.*, ./, .^,

Linspace[a,b,N] ➔ Vector can be created by evenly spaced points. From a to b the vector is created by 'N' evenly spaced points

Eg: linspace[0,1,5] → 0 0.25 0.5 0.75 1

Transpose of a matrix – y'

ans =

1. 2. 4.
4. 7. 1.
6. 3. 1.

RESULT

Study of basic SCILab commands are worked and executed

Experiment: 2

Generation of Various Signals & Sequences (Periodic/Aperiodic), such as Unit Impulse, Unit Step, Square, Sawtooth, Triangular

Scilab code Solution 2.1 Generation Of Unit Impulse and Unit Step Signal and Sequences

```
1 //Experiment Number:2.1
2 //Write a program to generate unit impulse and unit
     step Signals and Sequences
3 //Basic Simulation Laboratory
4 //B.Tech II Year I Sem
5 //Studdent Name:                    Enrolement Number:
6 // Course Instructor: Dr.Kantipudi MVV Prasad,
7 // Sreyas Institute Of Engineering & Technlogy,
     Hyderabad.
8 //
```

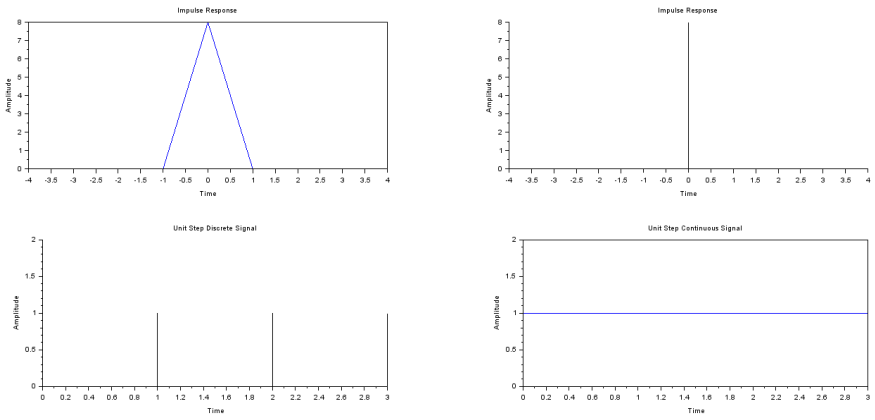


Figure 2.1: Generation Of Unit Impulse and Unit Step Signal and Sequences

```
9
10 // OS : Windows 10.1
11 // Scilab 6.0.2
12
13
14 clc;
15 close
16 clear ;
17
18 // Unit Impulse Signal and Sequence
19
20 t=-4:1:4;
21 a=[zeros(1,4) 1 zeros(1,4)];
22 k=input("Enter the Amplitude : "); // reading
      amplitude value from keyboard
23 b=k*a;
24
25 subplot(2,2,1);
26 plot(t,b);
27 xlabel("Time");
28 ylabel("Amplitude");
```

```
29 title("Impulse Response");
30
31 subplot(2,2,2);
32 plot2d3(t,b);
33 xlabel("Time");
34 ylabel("Amplitude");
35 title("Impulse Response");
36
37 // Unit Step Signal and Sequence:
38
39 // Discrete Signal
40
41 t=0:3;
42 y=ones(1,4);
43
44 subplot(2,2,3);
45 plot2d3 (t,y);
46 xlabel('Time');
47 ylabel('Amplitude');
48 title('Unit Step Discrete Signal');
49
50 // Continuous Signal
51
52 subplot(2,2,4);
53 plot(t,y);
54 xlabel('Time');
55 ylabel('Amplitude');
56 title('Unit Step Continuous Signal');
57
58 // Enter the Amplitude : 8
```

Scilab code Solution 2.2 Generation Of Square Wave and Sawtooth Wave Signals and Sequences

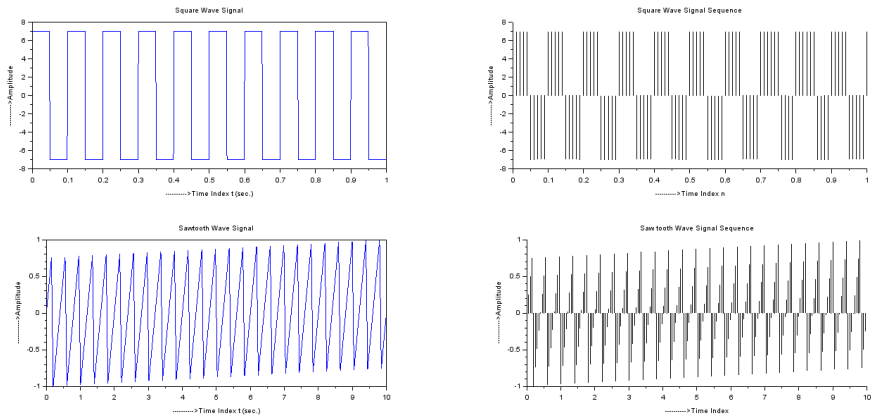


Figure 2.2: Generation Of Square Wave and Sawtooth Wave Signals and Sequences

```
1  //Experiment Number:2.2
2  //Write a program to generate square wave and
       sawtooth wave Signals and Sequences
3  //Basic Simulation Laboratory
4  //B.Tech II Year I Sem
5  //Studdent Name:                    Enrolement Number:
6  // Course Instructor: Dr.Kantipudi MVV Prasad,
7  // Sreyas Institute Of Engineering & Technlogy,
       Hyderabad.
8  //
       ____________________________________________________________________________
9
10 // OS : Windows 10.1
11 // Scilab 6.0.2
12
13
14 clc;
15 close;
16 clear ;
17
18
```

```
19 // continuous square wave Signal:
20
21 a=input('Enter Amplitude : ');
22 t=0:0.001:1;
23 d=a*squarewave(2*%pi*10*t);
24
25 subplot(2,2,1);
26 plot(t,d);
27 xlabel ("---------->Time Index t (sec.)");
28 ylabel ("---------->Amplitude");
29 title (" Square Wave Signal ");
30
31 // discrete square wave signal
32
33 //a=input('Enter amplitude');
34 n=0 : 0.01 :1;
35 d=a*squarewave(2*%pi*10*n);
36
37 subplot(2,2,2);
38 plot2d3(n,d);
39 xlabel ("---------->Time Index n");
40 ylabel ("---------->Amplitude");
41 title ("Square Wave Signal Sequence");
42
43 // Sawtooth Wave Signal
44
45 Fs = 20; // samples per second
46 t_total = 10; // seconds
47 n_samples = Fs * t_total;
48 t = linspace(0, t_total, n_samples);
49 f=500; // sound frequency
50
51 saw_wave=2*(f*t-floor(0.5+f*t));
52
53 subplot(2,2,3);
54 plot(t,saw_wave);
55 xlabel ("---------->Time Index t (sec.)");
56 ylabel ("---------->Amplitude");
```

```
57 title (" Sawtooth Wave Signal ");
58
59 // sawtooth wave sequence
60
61 Fs = 20; // samples per second
62 t_total = 10; // seconds
63 n_samples = Fs * t_total;
64 n = linspace(0, t_total, n_samples);
65 f=500; // sound frequency
66
67 saw_wave=2*(f*n-floor(0.5+f*n));
68
69 subplot(2,2,4);
70
71 plot2d3(n,saw_wave);
72 xlabel ("---------->Time Index ");
73 ylabel ("---------->Amplitude");
74 title ("Saw tooth Wave Signal Sequence");
75
76
77 // Input Parameters
78 //  Enter Amplitude : 7
```

Scilab code Solution 2.3 Generation Of Triangular and Sinusoidal Signal and Sequences

```
1 //Experiment Number:2.3
2 //Write a program to generate Triangular and
     Sinusoidal Signals and Sequences
3 //Basic Simulation Laboratory
4 //B.Tech II Year I Sem
5 //Studdent Name:                Enrolement Number:
6 // Course Instructor: Dr.Kantipudi MVV Prasad,
```

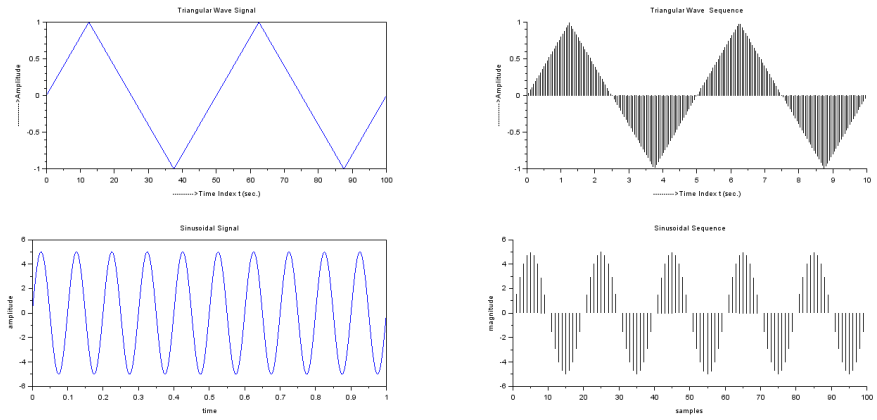


Figure 2.3: Generation Of Triangular and Sinusoidal Signal and Sequences

```
7  // Sreyas Institute Of Engineering & Technlogy,
       Hyderabad.
8  //
   ____________________________________________________________________________


9
10
11 // OS : Windows 10.1
12 // Scilab 6.0.2
13
14
15 clc;
16 close;
17 clear ;
18
19 // Triangular Wave Signal
20
21 Fs = 20; // samples per second
22 t_total = 100; // seconds
23 n_samples = Fs * t_total;
24 t = linspace(0, t_total, n_samples);
25 f=40; // sound frequency
26
```

```
27 tri_wave=(2/%pi)*asin(sin(2*%pi*f*t));
28
29 subplot(2,2,1);
30
31 plot(t,tri_wave);
32 xlabel ('---------->Time Index t (sec.)');
33 ylabel ('---------->Amplitude');
34 title ('Triangular Wave Signal ');
35
36 // traiangular wave sequence
37
38 Fs = 20; // samples per second
39 t_total = 10; // seconds
40 n_samples = Fs * t_total;
41 n = linspace(0, t_total, n_samples);
42 f=40; // sound frequency
43
44 tri_wave=(2/%pi)*asin(sin(2*%pi*f*n));
45
46 subplot(2,2,2);
47 plot2d3(n,tri_wave);
48 xlabel ('---------->Time Index t (sec.)');
49 ylabel ('---------->Amplitude');
50 title ('Triangular Wave  Sequence');
51
52
53 // continuous Sinusoidal Signal
54
55 a=input('Enter amplitude for Sinusoidal Signal: ');
56 t=0:0.001:1;
57 p=a*sin(2*%pi*10*t);
58
59 subplot(2,2,3);
60 plot(t,p);
61 title('Sinusoidal Signal');
62 xlabel('time');
63 ylabel('amplitude');
64
```

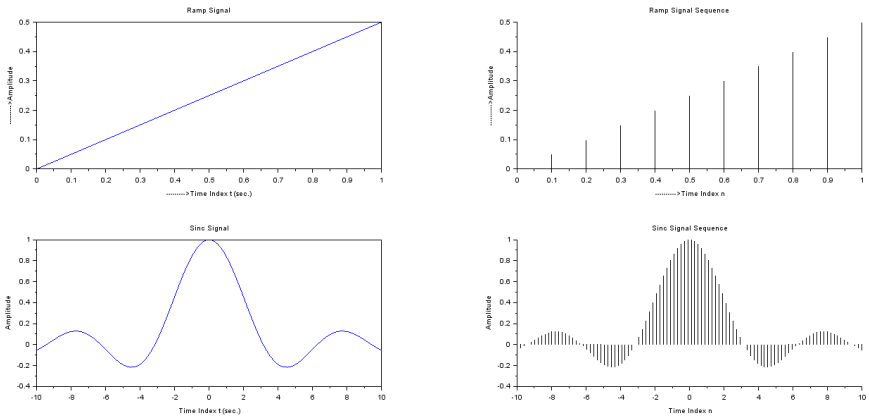


Figure 2.4: Generation Of Ramp and Sinc Signals and Sequences

```
65 // discrete sinuoidal signal
66
67 //a=input('Enter magnitude');
68 n = 0:100;
69 x =a*sin(((2*0.05)*%pi)*n);
70
71 subplot(2,2,4);
72 plot2d3(n,x);
73 title("Sinusoidal Sequence");
74 xlabel("samples");
75 ylabel("magnitude");
76
77 // After Getting Trainagular wave output ,vist the
   command window to enter Input Parameters
78 //  Enter amplitude for Sinusoidal Signal: 5
```

Scilab code Solution 2.4 Generation Of Ramp and Sinc Signals and Sequences

```
1  //Experiment Number:2.4
2  //Write a program to generate ramp and sinc Signals
       and Sequences
3  //Basic Simulation Laboratory
4  //B.Tech II Year I Sem
5  //Studdent Name:                    Enrolement Number:
6  // Course Instructor: Dr.Kantipudi MVV Prasad,
7  // Sreyas Institute Of Engineering & Technlogy,
       Hyderabad.
8  //
       ------------------------------------------------------------------------

9
10 // OS : Windows 10.1
11 // Scilab 6.0.2
12
13
14 clc;
15 close
16 clear ;
17
18 //continuous ramp signal
19
20 t = 0 : 0.001 : 1;
21 y = 0.5 * t;
22
23 subplot(2,2,1);
24 plot( t , y );
25 xlabel ('---------->Time Index t (sec.)');
26 ylabel ('---------->Amplitude');
27 title ('Ramp Signal');
28
29 //discrete ramp signal
30
31 n = 0 : 0.1 : 1;
32 y = 0.5 * n;
33
34 subplot(2,2,2);
```

```
35 plot2d3(n,y);
36 xlabel ('---------->Time Index n');
37 ylabel ('---------->Amplitude');
38 title ('Ramp Signal Sequence');
39
40 //continuous sinc signal
41
42 t=linspace(-10 , 10);
43 y=sinc(t);
44
45 subplot(2,2,3);
46 plot(t,y);
47 xlabel("Time Index t (sec.)");
48 ylabel("Amplitude");
49 title("Sinc Signal ");
50
51 //discrete sinc signal
52
53 n =linspace(-10 , 10);
54 y =sinc(n);
55
56 subplot(2,2,4);
57 plot2d3(n,y);
58 xlabel("Time Index n");
59 ylabel("Amplitude");
60 title("Sinc Signal Sequence");
```

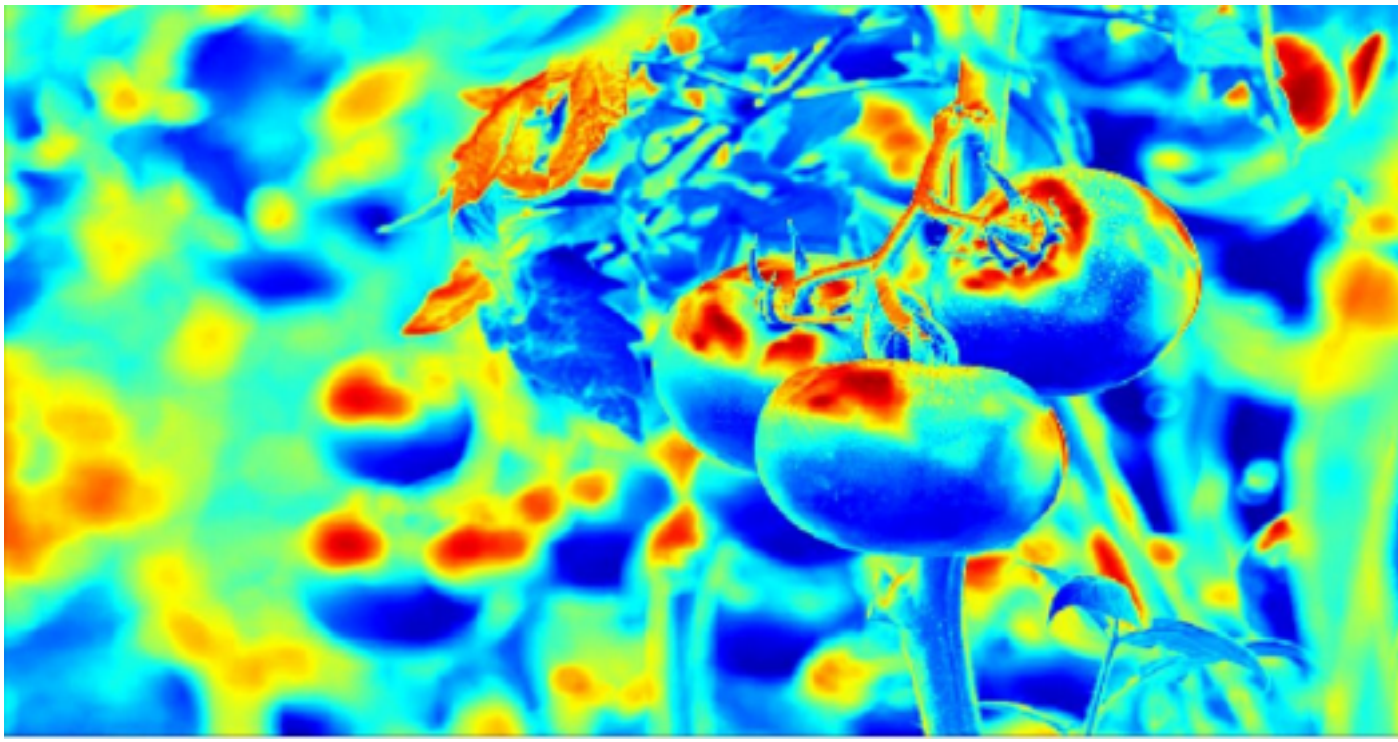
LAB-10

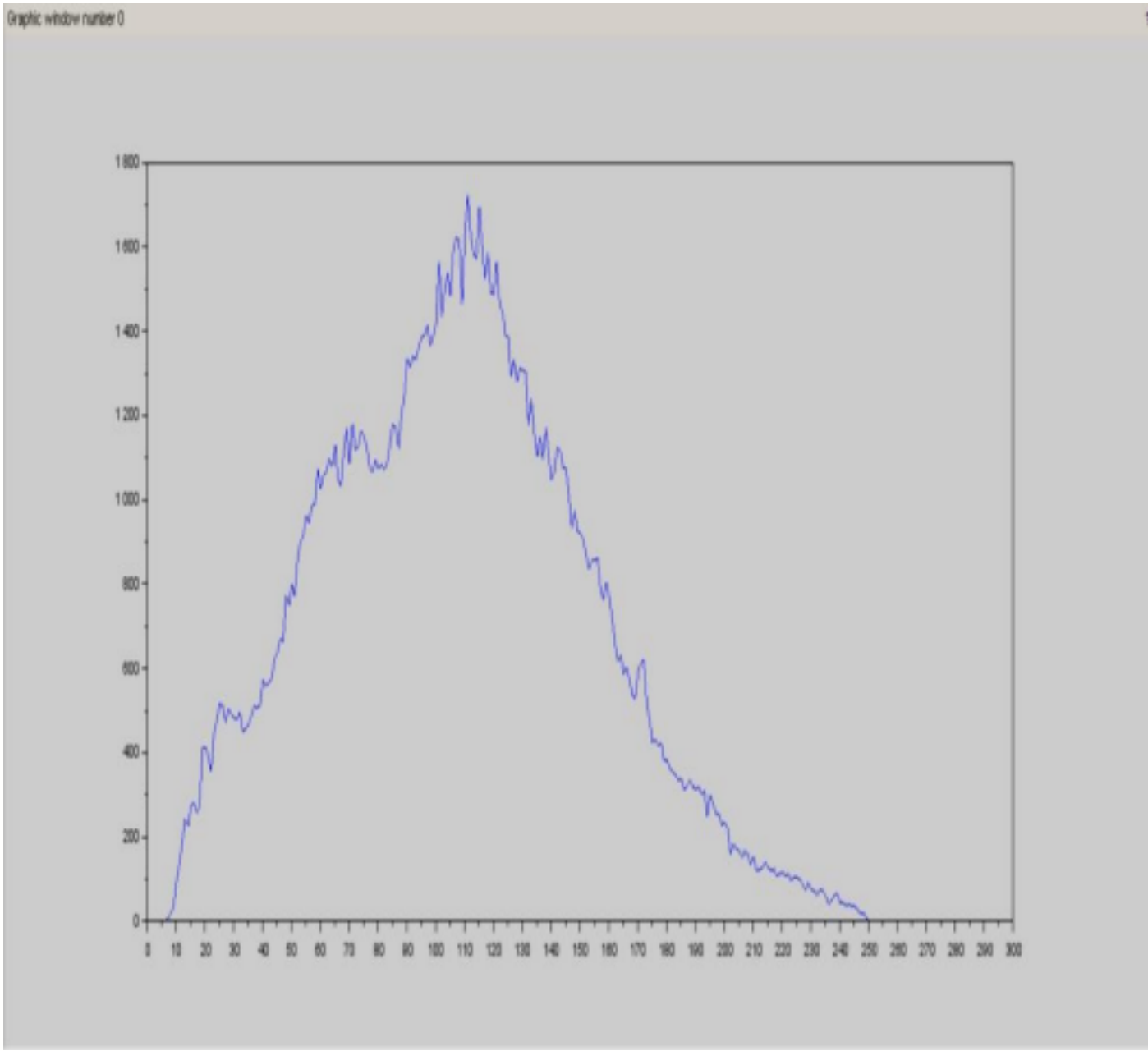
Aim: Apply Colour Image segmentation algorithm

Code:

```
--> RGB = imread('tomatoes.jpg');

--> imshow(RGB) = gcf();f.name='Color Image';

Unexpected redefinition of Scilab function.
--> Image = rgb2gray(RGB);

--> imshow(Image);

--> f=gcf();f.name='Gray Level Image';

--> imshow(Image,jetcolormap(256))

--> f=gcf();

--> f.name='Pseudo Color Image';

--> imshow(Image);

--> imshow(Image)

--> imshow(Image,jetcolormap(256))

--> Histogram=imhist(Image);

--> figure();plot(0:255, Histogram')

--> xgrid(color('black'),1,8)
```

Output:




Graphic window number 0
1800
1600
1400
1200
1000
800
600
400
200
0
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280 290 300

LAB-9

Aim: Image filtering in spatial and frequency domain

Code:

Low Pass Filter (Frequency domain):

```
--> xdel (winsid());
Warning: Feature xdel(…) is obsolete and will be permanently removed in Scilab 6.2
Warning: Please use close(…) instead.

--> fc = input ("Enter Analog cutoff freq. in Hz=") //250
Enter Analog cutoff freq. in Hz=250

 fc =

   250.

--> fs = input (" Enter Analog sampling freq. in Hz=")
 Enter Analog sampling freq. in Hz=2000

 fs  =

   2000.

--> M = input ("Enter order of filter =")//4
Enter order of filter =4

 M =

   4.

--> w = (2* %pi)*(fc/fs);

--> disp (w, ' Digital cutoff frequency in radians. cycles/samples');

   0.7853982

  " Digital cutoff frequency in radians. cycles/samples"
```

```
--> wc = w/%pi;

--> disp (wc, 'Normalized digital cutoff frequency in cycles /samples');

  0.25

 "Normalized digital cutoff frequency in cycles /samples"

--> [wft,wfm,fr]=wfir('lp',M+1,[wc/2,0],'re' , [0,0]);

--> disp(wft, 'Impulse Response of LPF FIR Filter:h[n} =');

  0.1591549  0.2250791  0.25  0.2250791  0.1591549

 "Impulse Response of LPF FIR Filter:h[n} ="

--> //Plotting the Magnitude Response of LPF FIR Filter

--> subplot (2,1,1)

--> plot (2*fr, wfm)

--> xlabel ('Normalized Digital Frequency W -------->')

--> ylabel (' Magnitude | H(w) |=')

--> title ( 'Magnitude Response of FIR LPF')

--> xgrid (1)

--> subplot (2,1,2)

--> plot (fr*fs,wfm)

--> xlabel ('Analog Frequency in Hz f - >')

--> ylabel ('Magnitude |H(w) |=')

--> title ('Magnitude Response of FIR LPF')

--> xgrid (1)
```

Spatial domain: Linear filtering & Non-Linear Filtering

```
--> clear

--> clear;

--> close;

--> I=imread("C:\Users\Diksha Nasa\Desktop\Study Material\IT Workshop using
scilab\saltpepperlenna.png");

--> I_noise=imnoise(I,"salt & pepper");

--> figure
 ans  =

Handle of type "Figure" with properties:
=======================================
children: "Axes"
figure_position = [200,200]
figure_size = [626,587]
axes_size = [610,460]
auto_resize = "on"
viewport = [0,0]
figure_name = "Graphic window number %d"
figure_id = 0
info_message = ""
color_map = matrix 33x3
pixel_drawing_mode = "copy"
anti_aliasing = "off"
immediate_drawing = "on"
background =  33
visible = "on"
rotation_style = "unary"
event_handler = ""
event_handler_enable = "off"
user_data = []
resizefcn = ""
closerequestfcn = ""
resize = "on"
toolbar = "figure"
toolbar_visible = "on"
menubar = "figure"
menubar_visible = "on"
infobar_visible = "on"
```

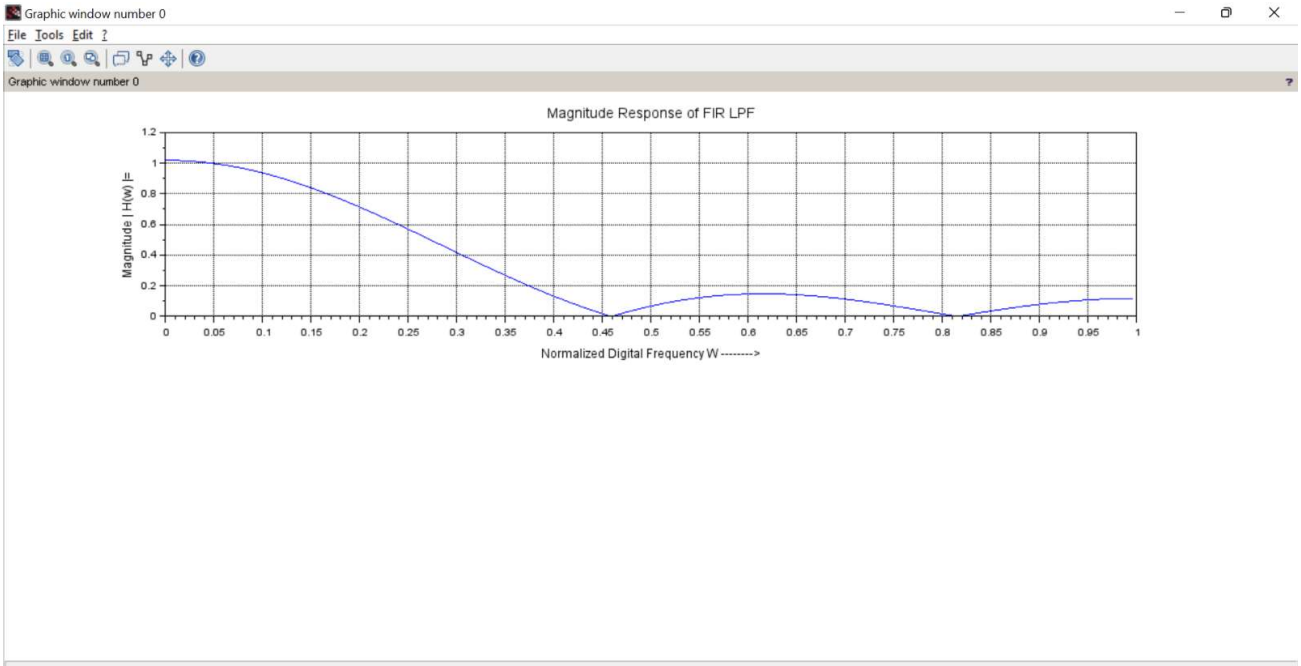
```
dockable = "on"
layout = "none"
layout_options = "OptNoLayout"
default_axes = "on"
icon = ""
tag = ""

--> imshow(I)

--> figure;

--> imshow(I_noise);

--> F_Linear1=1/25*ones(5,5);


--> I_linear1=imfilter(I_noise,F_Linear1);

--> figure
 ans  =

Handle of type "Figure" with properties:
========================================
children: "Axes"
figure_position = [200,200]
figure_size = [626,587]
axes_size = [610,460]
auto_resize = "on"
viewport = [0,0]
figure_name = "Graphic window number %d"
figure_id = 1
info_message = ""
color_map = matrix 33x3
pixel_drawing_mode = "copy"
anti_aliasing = "off"
immediate_drawing = "on"
background =  33
visible = "on"
rotation_style = "unary"
event_handler = ""
event_handler_enable = "off"
user_data = []
resizefcn = ""
closerequestfcn = ""
resize = "on"
```

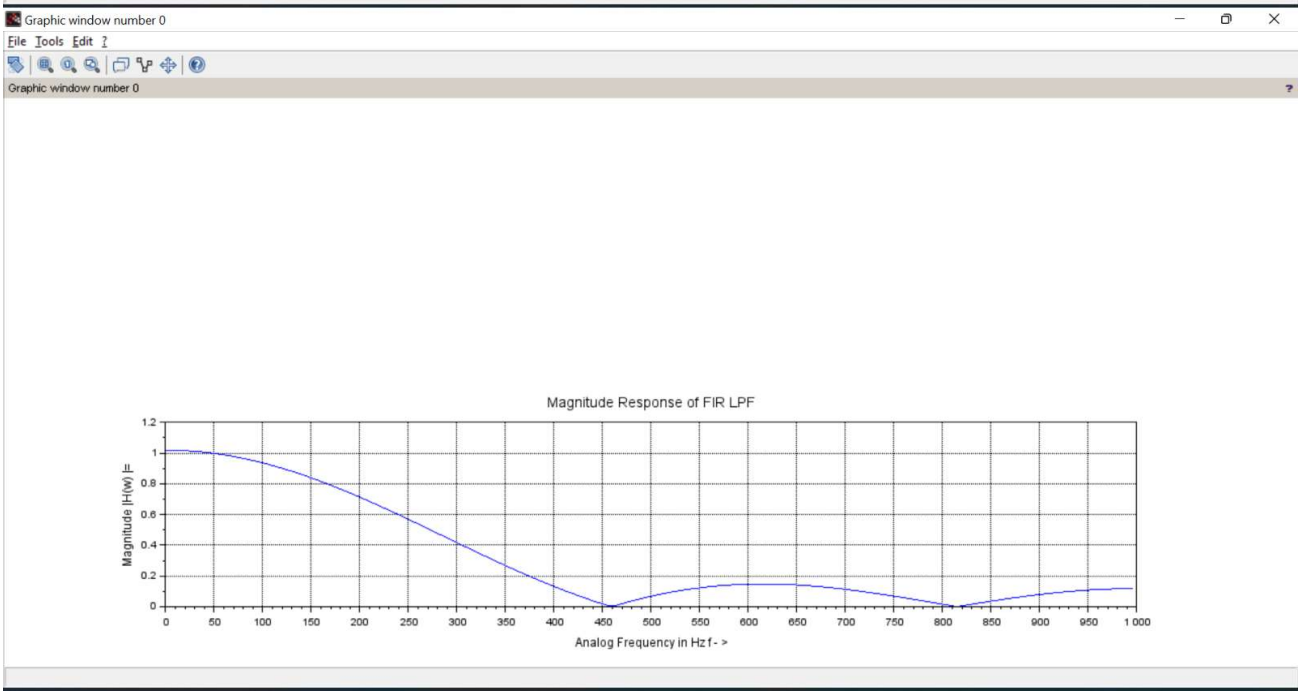
```
toolbar = "figure"
toolbar_visible = "on"
menubar = "figure"
menubar_visible = "on"
infobar_visible = "on"
dockable = "on"
layout = "none"
layout_options = "OptNoLayout"
default_axes = "on"
icon = ""
tag = ""

--> imshow(I_linear1);

--> hsize=[5,5];

--> sigma=1;

--> F_Linear2=fspecial('gaussian',hsize,sigma);

--> I_linear2=imfilter(I_noise,F_Linear2);

--> figure
 ans  =

Handle of type "Figure" with properties:
========================================
children: "Axes"
figure_position = [200,200]
figure_size = [626,587]
axes_size = [610,460]
auto_resize = "on"
viewport = [0,0]
figure_name = "Graphic window number %d"
figure_id = 2
info_message = ""
color_map = matrix 33x3
pixel_drawing_mode = "copy"
anti_aliasing = "off"
immediate_drawing = "on"
background =  33
visible = "on"
rotation_style = "unary"
event_handler = ""
event_handler_enable = "off"
```

```
user_data = []
resizefcn = ""
closerequestfcn = ""
resize = "on"
toolbar = "figure"
toolbar_visible = "on"
menubar = "figure"
menubar_visible = "on"
infobar_visible = "on"
dockable = "on"
layout = "none"
layout_options = "OptNoLayout"
default_axes = "on"
icon = ""
tag = ""

--> imshow(I_linear2);

--> F_NonLinear=[3,3];


--> [m,n]=size(I);


--> for i=2:m-1
 > for j=2:n-1
 > d(i,j)=median([I(i-1,j+1),I(i,j+1),I(i+1,j+1);I(i-1,j),I(i,j),I(i+1,j);I(i-1,j-1),I(i,j-1),I(i+1,j-1)]);
 > end
 > end

--> imshow(d)
```

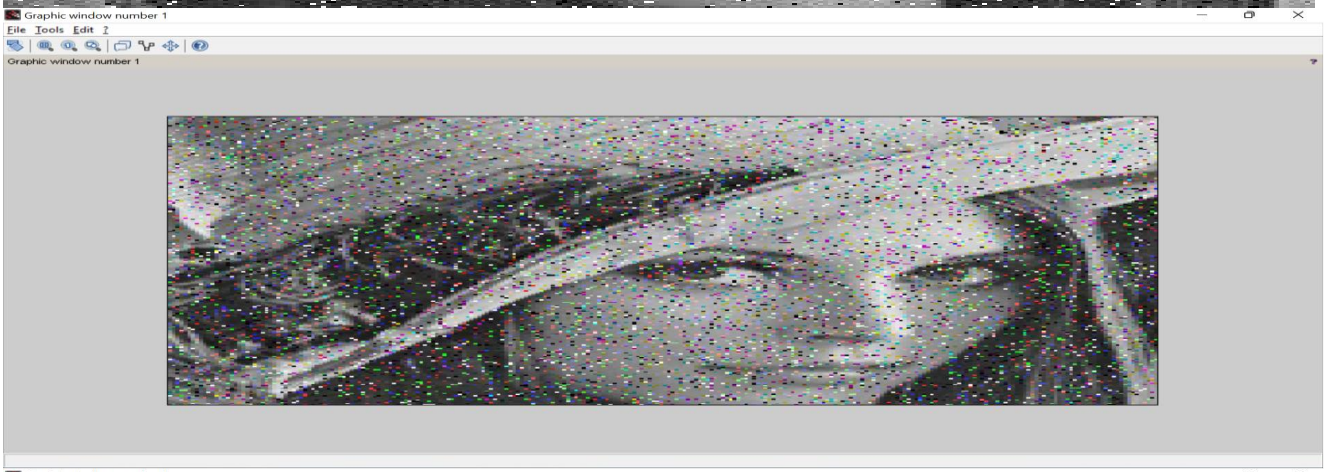
Output:

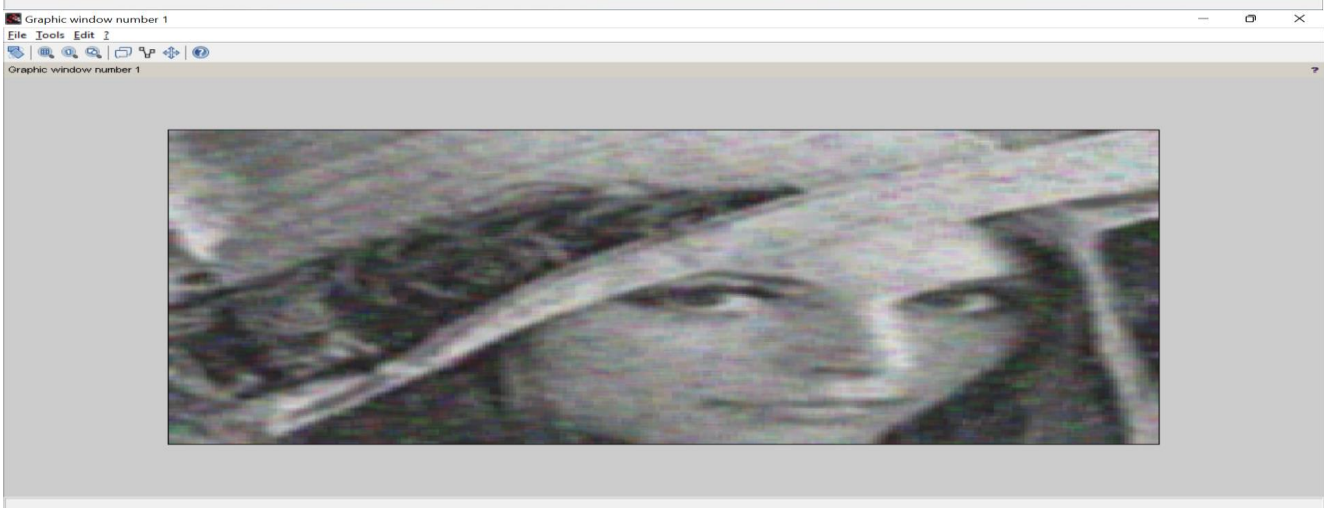
Low pass filter :





Spatial Domain Processing:





EXP. No: 6

Date :

IMAGE EDGE DETECTION USING SOBEL AND CANNEY FILTERING

Objective:

To perform the image edge detection using sobel and canney filtering using SCILAB.

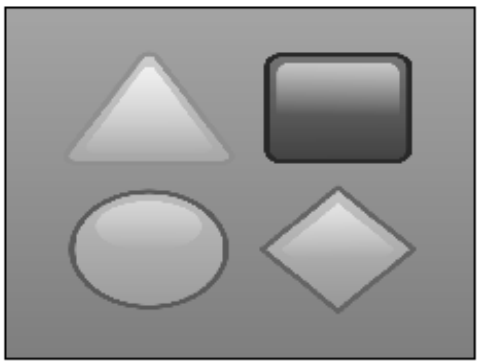
Code:

```
close ;
clear ;
clc ;
img = imread('shapes.jpg');
// Reads input image shapes.jpg
img = rgb2gray(img);
// Converts input image to gray scale
clf
// Clears figure handle
subplot(2,2,1);
imshow(img);
title('Gray scale image');
e=edge(img);//This performs edge detection operation with sobel, threshold =0.5
subplot(2,2,2);
imshow(e)
title('sobel filter');

e = edge(img, 'prewitt'); // threshold=0.5
// Applied prewitt edge detection method
subplot(2,2,3);;
imshow(e)
title('Prewitt image');
e = edge(img,'canny', [0.06 0.2]);
// Applies canny edge detection method
```

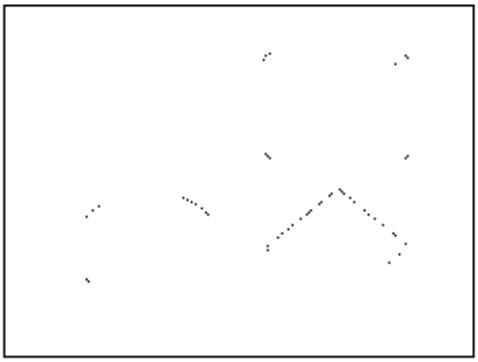
```
subplot(2,2,4);
imshow(e)
title('canney filter');
e = edge(img, 'fftderiv', 0.4) ;
// Applies FFT gradient method ; 0.4 threshold
figure;
imshow(e)
title('FFT image');
```

Output:

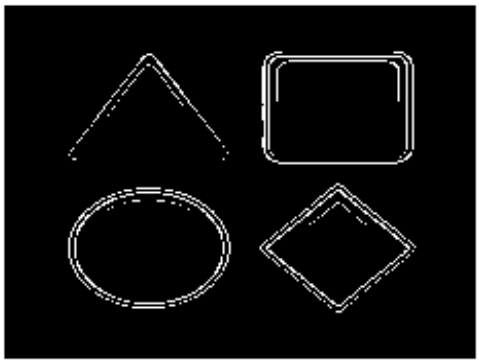
Gray scale image



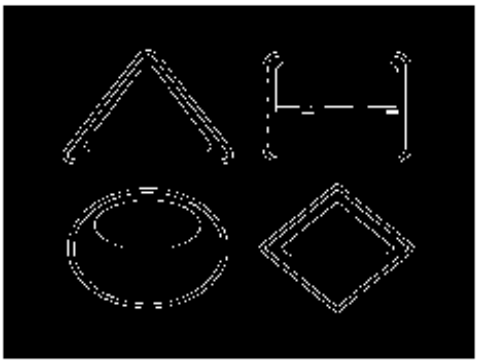
sobel filter

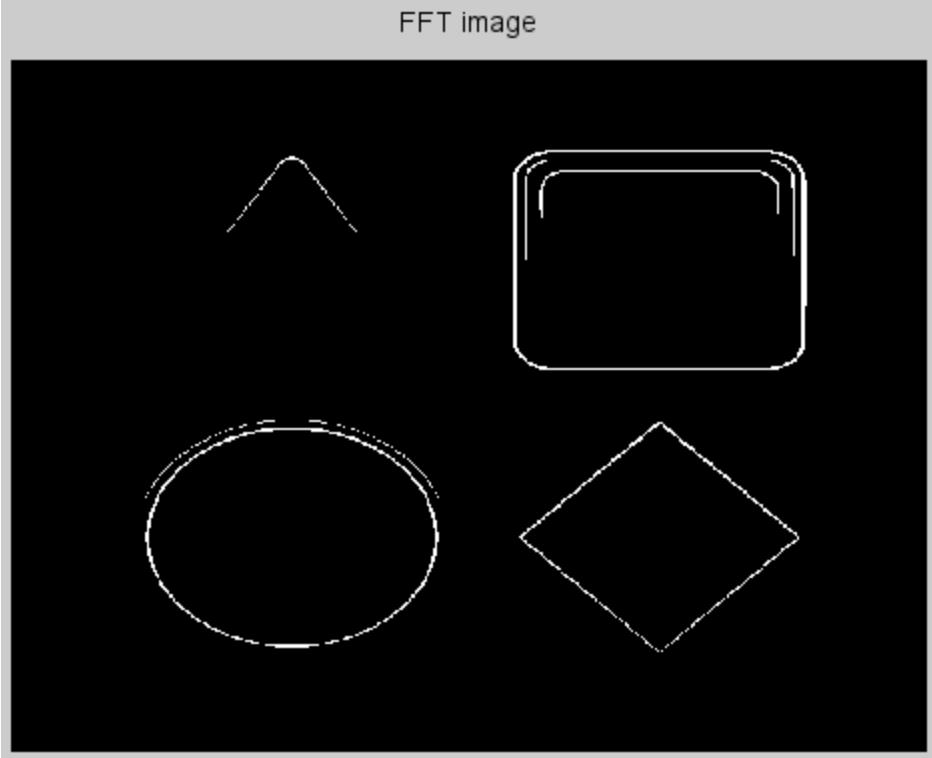


Prewitt image



canney filter





Result:

Thus, the image edge detection using Sobel and Canney filtering is verified successfully using SCILAB.