

# **IMAGE PROCESSING**

**A PRACTICAL REPORT  
ON  
IMAGE PROCESSING**

**SUBMITTED BY  
Mr. RAHUL KEWAT  
Roll No: 22006**

**UNDER THE GUIDANCE OF  
PROF. CHIRAG DEORA**

**Submitted in fulfillment of the requirements for qualifying  
MSc. IT Part I Semester - II Examination 2022-2023**

**University of Mumbai  
Department of Information Technology**

**R.D. & S.H National College of Arts, Commerce &  
S.W.A. Science College Bandra (West), Mumbai – 400 050**



## **R. D. & S. H. National & S. W. A. Science College**

Bandra (W), Mumbai – 400050.

**Department of Information Technology  
M.Sc. (IT – SEMESTER II)**

### **Certificate**

This is to certify that Image Processing Practicals performed at R.D  
& S.H National & S.W.A. Science College by Mr. Rahul Kewat  
holding Seat No. \_\_\_\_\_ studying Master of Science in Information  
Technology Semester – II has been satisfactorily completed as  
prescribed by the University of Mumbai, during the year 2022– 2023.

**Subject In-Charge      Coordinator In-Charge      External Examiner**

**College Stamp**

# **INDEX**

<b>Sr. No</b>	<b>Date</b>	<b>Practical</b>	<b>Page No.</b>	<b>Sign</b>
1	15/3/23	Image Sampling	1	
2	29/3/23	Basic Intensity Transformation functions	8	
3	05/3/23	Filtering in Frequency Domain Program to apply Low pass and High pass filters in frequency domain.	23	
4	12/4/23	Image Denoising	34	
5	19/4/23	Colour Image Processing	38	
6	26/4/23	Fourier Related Transformation	44	
7	3/5/23	Morphological Image Processing	46	
8	10/5/23	Image Segmentation	51	

**Practical 1**

**Aim:** Program to calculate the number of samples required for an image.

**Writeup:**

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

**[A]:** Program to calculate number of samples required for an image.

**Code:**

```

clc;
clear;
fm = input('input signal frequency');
k = input('no. of cycles');
A = input('Enter amplitude signal');
tm = 0:1 / (fm*fm) .k/fm;
x = A*cos(2* %pi *fm*tm);
figure(1);

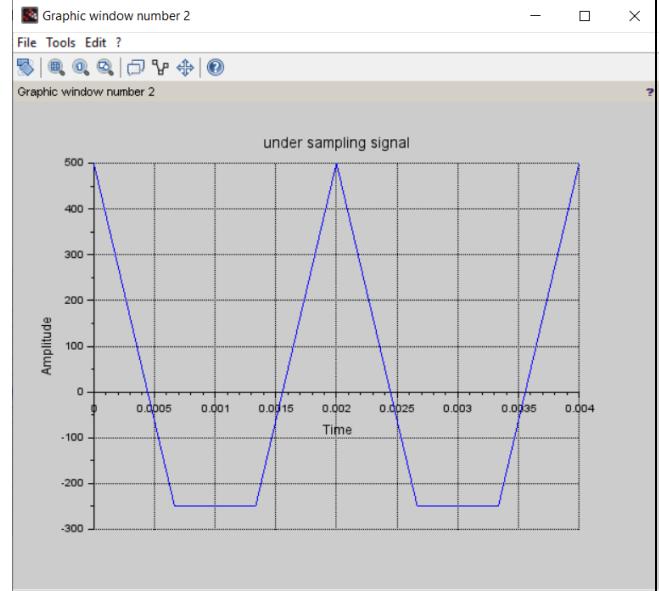
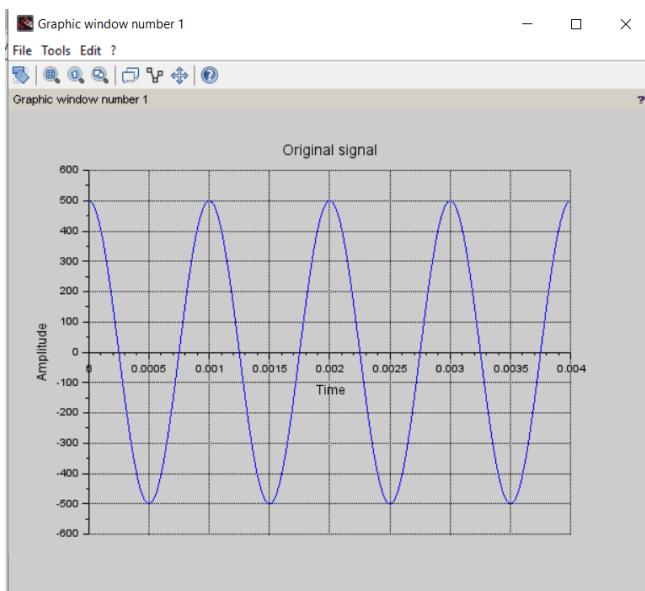
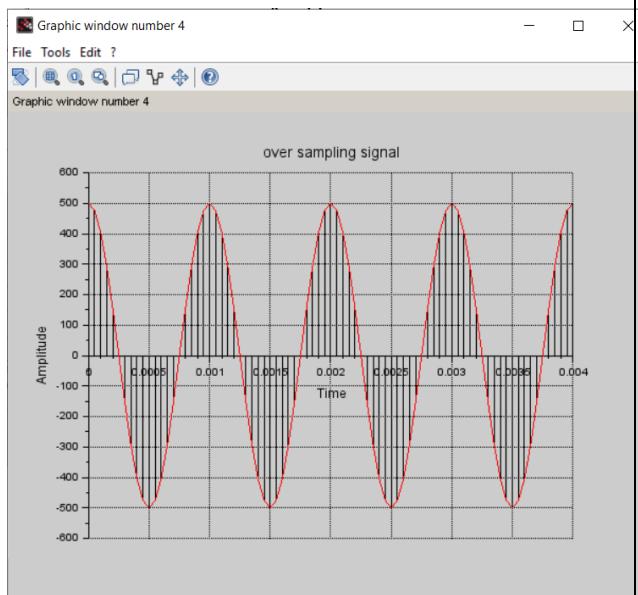
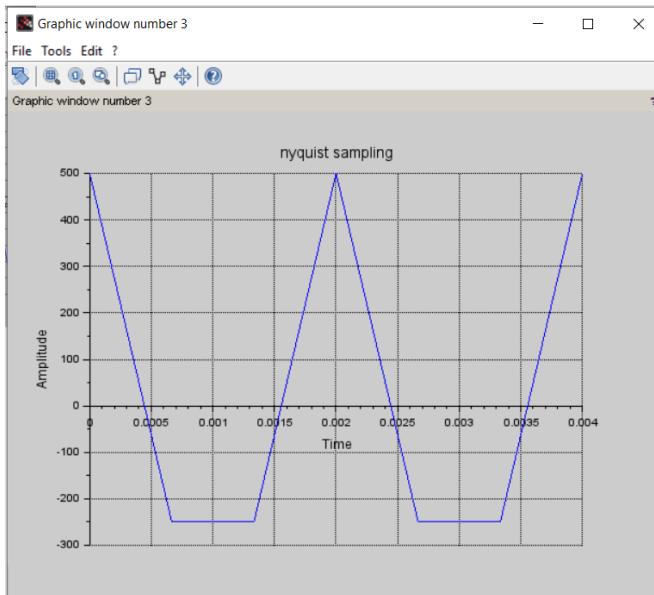
a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot(tm,x);
title('Original signal');
xlabel('Time');
ylabel('Amplitude');
xgrid(1)
fnyq = 2* fm;
fs = (3/4) * fnyq ;
n = 0:1 / fs. k / fm;
x = A * cos(2 * %pi *fm * n);
figure(2);

a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot(n,x);
title('under sampling signal');
xlabel('Time');
ylabel('Amplitude');
xgrid(1)
fnyq = 2* fm;
fo = (3/4) * fnyq ;
o = 0:1 / fo : k / fm;
x = A * cos(2 * %pi *fm * o);
figure(3);

a = gca();
a.x_location = "origin";
a.y_location = "origin";
plot(o,x);
title('nyquist sampling');
xlabel('Time');
ylabel('Amplitude');
xgrid(1)
fo = 10 * fnyq ;
o = 0:1 / fo : k / fm;
x = A * cos(2 * %pi *fm * o);
figure(4);
a = gca();

a.x_location = "origin";
a.y_location = "origin";
plot2d3('gnn',o,x);
plot(o,x,'r');
title('over sampling signal');
xlabel('Time');
ylabel('Amplitude');
xgrid(1)

```

**Output:****Graph 1****Graph 2****Graph 3****Graph 4**

**[B]: Program to study the effects of reducing the spatial resolution of a digital images.**

**Code:**

```

clc;
clear all;
Img = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\lena.png');
subplot(2,2,1),imshow(Img),title('Og image 512*512');

Samp=zeros(256);
for i=1:1:512
    for j=1:1:512
        if modulo(i,2)==0
            m=i/2;
            if modulo(j,2)==0
                n=j/2;
                Samp(i-m,j-n)=Img(i,j);
            else
                n=0;
            end
        else
            m=0;
        end
    end
end

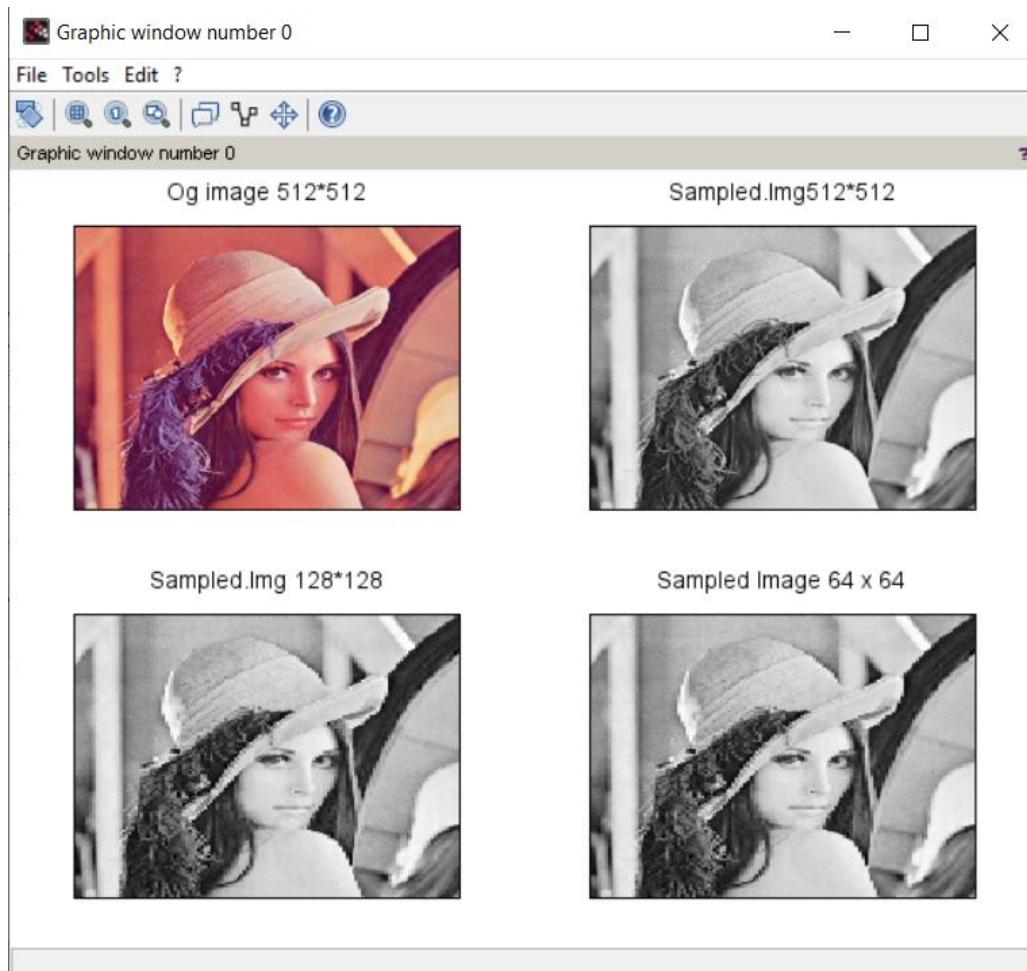
SampImg256=mat2gray(Samp);
subplot(2,2,2),imshow(SampImg256),title('Sampled.Img 512*512')
Samp=zeros(128)
for i=1:1:512
    for j=1:1:512
        if modulo(i,4)==0
            m=i/4*3;
            if modulo(j,4)==0
                n=j/4*3;
                Samp(i-m,j-n)=Img(i,j);
            else
                n=0;
            end
        else
            m=0;
        end
    end
end

SampImg128=mat2gray(Samp);
subplot(2,2,3),imshow(SampImg128),title('Sampled.Img 128*128')
Samp=zeros(64)
for i=1:1:512
    for j=1:1:512
        if modulo(i,8)==0
            m=i/8*7;
            if modulo(j,8)==0
                n=j/8*7;
                Samp(i-m,j-n)=Img(i,j);
            else
                n=0;
            end
        else
            m=0;
        end
    end
end

```

```
SampImg64=mat2gray(Samp);
subplot(2,2,4),imshow(SampImg128),title('Sampled Image 64 x 64');

Samp=zeros(32);
for i=1:1:512
    for j=1:1:512
        if modulo(i,16)==0
            m=i/16*4;
        if modulo(j,16)==0
            n=j/16*4;
            Samp(i-m,j-n)=Img(i,j);
        else
            n=0;
        end
    else
        m=0;
    end
end
end
```

**Output:**

[C]: Program to study the effects of varying the number of intensity levels in a digital image.

Code:

```
clc;
clear all;
figure(1)
/// checker board

subplot(3,3,1);
i=imread('C:\Users\DELL\Downloads\csv\xmlhorus\2018_Audi_R8_Coupe_V10_plus_Front.jpg');
imshow(i);
title('original image');
subplot(3,3,2);

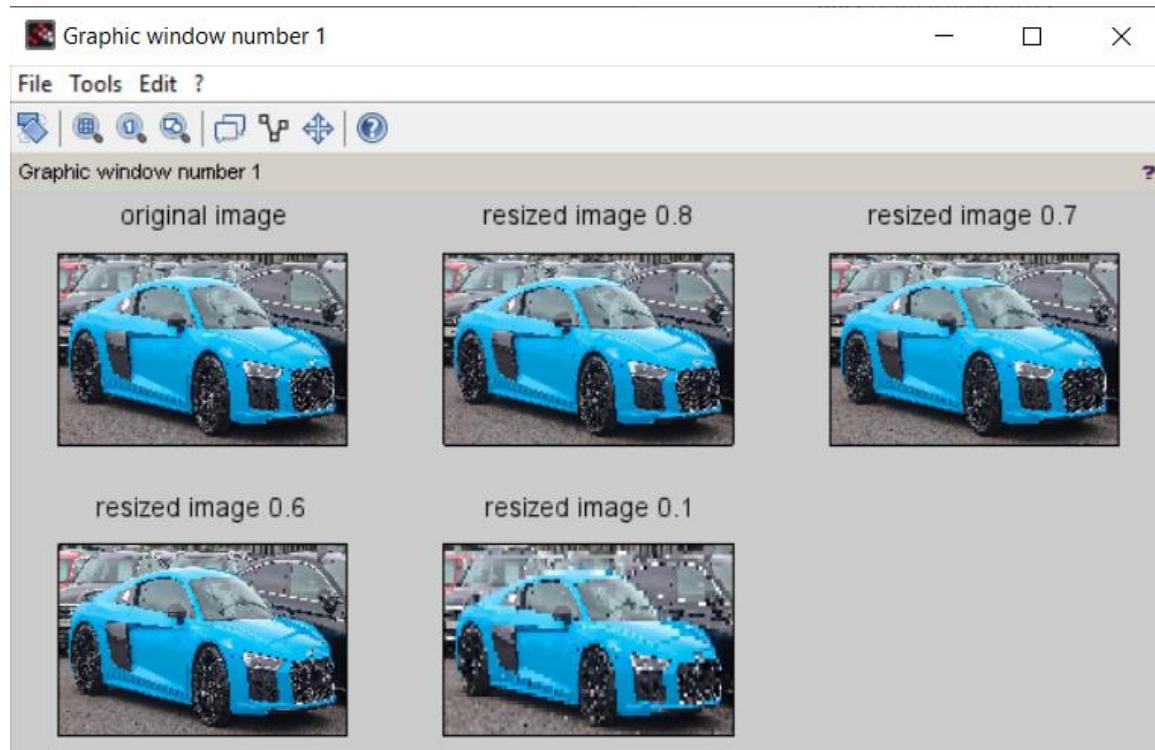
j=imresize(i,0.8);
imshow(j);
title('resized image 0.8');

subplot(3,3,3);
j=imresize(i,0.7);
imshow(j);
title('resized image 0.7');

subplot(3,3,4);
j=imresize(i,0.6);
imshow(j);
title('resized image 0.6');

subplot(3,3,5);
j=imresize(i,0.1);
imshow(j);
title('resized image 0.1');
```

Output:



[D]: Program to perform image averaging (image addition) for noise reduction:

**Code:**

```
clc;
clear all;

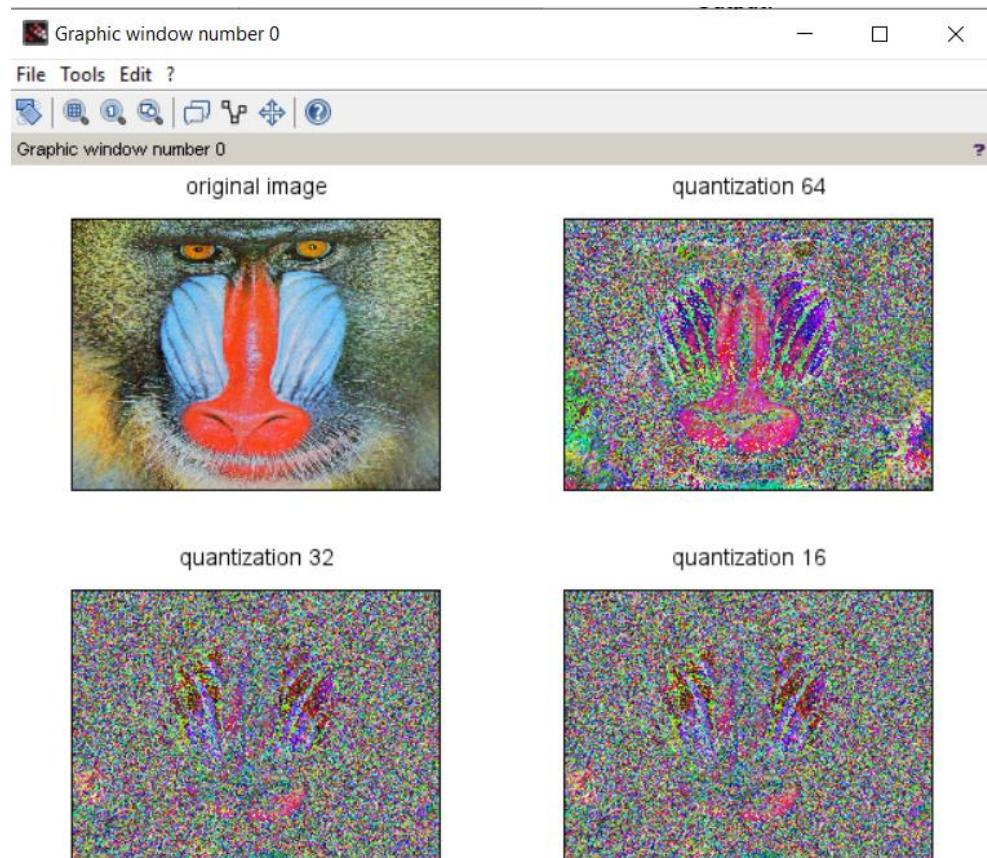
subplot(2,2,1);
i=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\baboon.png');
i=double(i);
imshow(uint8(i));
title('original image')
k=floor(i*256)/64;

subplot(2,2,2);
imshow(uint8(k));
title('quantization 64')
k=floor(i*255)/32;

subplot(2,2,3);
imshow(uint8(k));
title('quantization 32')
k=floor(i*255)/32;

subplot(2,2,4);
imshow(uint8(k));
title('quantization 16')
k=floor(i*255)/16;
```

**Output:**



**Practical 2**

**Aim:** Intensity transformation and Spatial Filtering.

**Writeup:**

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

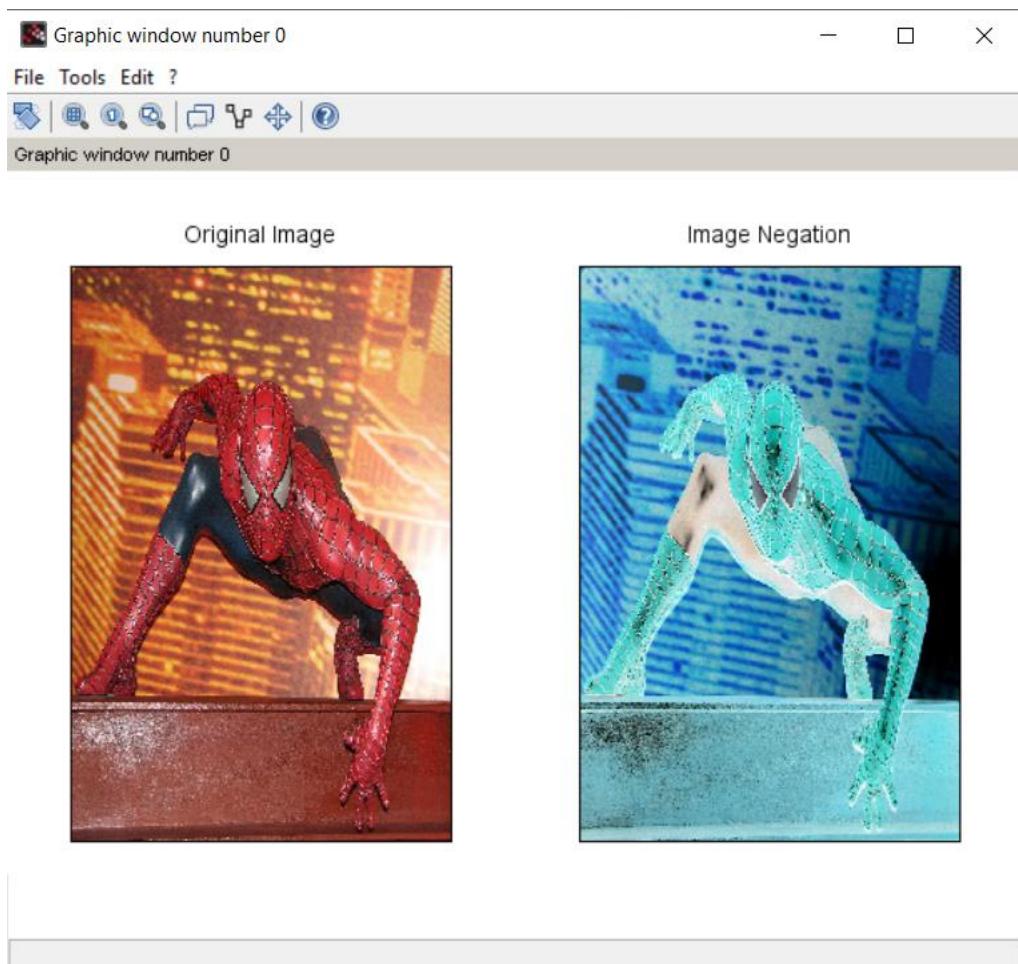
**[A]: Basic Intensity Transformation functions****1:Program to perform Image negation****Code:**

```
clc;
clear all;

i=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\Spiderman.JPG');
disp("Image Negation");
a=double(i);
c=255;
b=c-a;

subplot(1,2,1);
imshow(uint8(a));
title('Original Image');

subplot(1,2,2);
imshow(uint8(b));
title('Image Negation');
```

**Output:**

## 2: Program to perform threshold on an image

### Code:

```

clc;
clear all;

a=imread('C:\Users\OneDrive\Documents\Rahul Kewat\IPCV\images\675px-A_sunflower.jpg');
b=double(a)

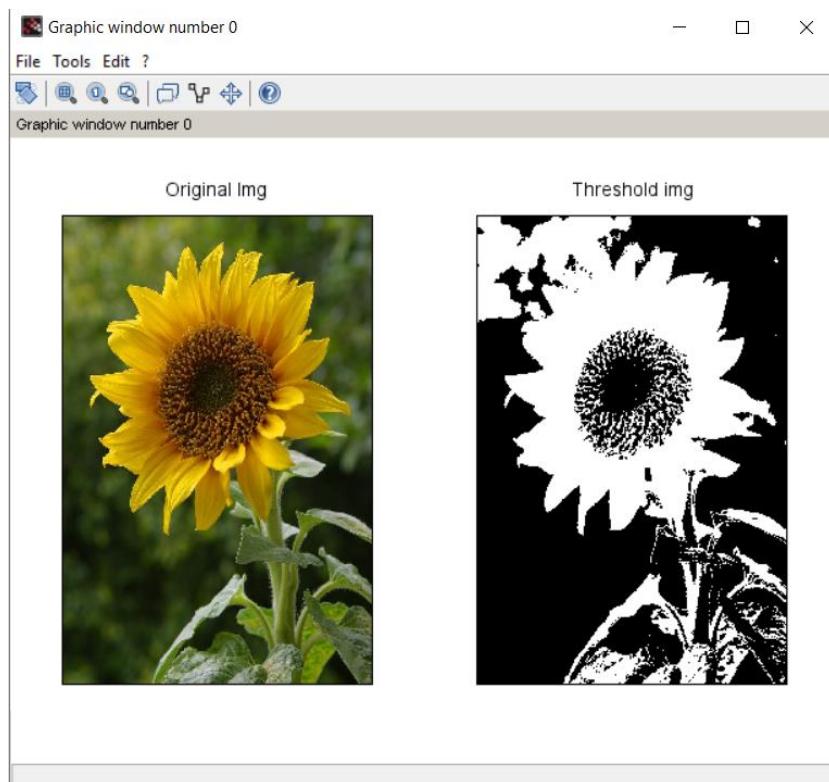
subplot(1,2,1);
imshow(a);
title('Original Img');

t=100;
[m,n]=size(b);
for i=1:m
    for j=1:n
        if(b(i,j)<t)
            c(i,j)=0;
        else
            c(i,j)=255;
        end
    end
end

subplot(1,2,2);
imshow(c);
title('Threshold img');

```

### Output:

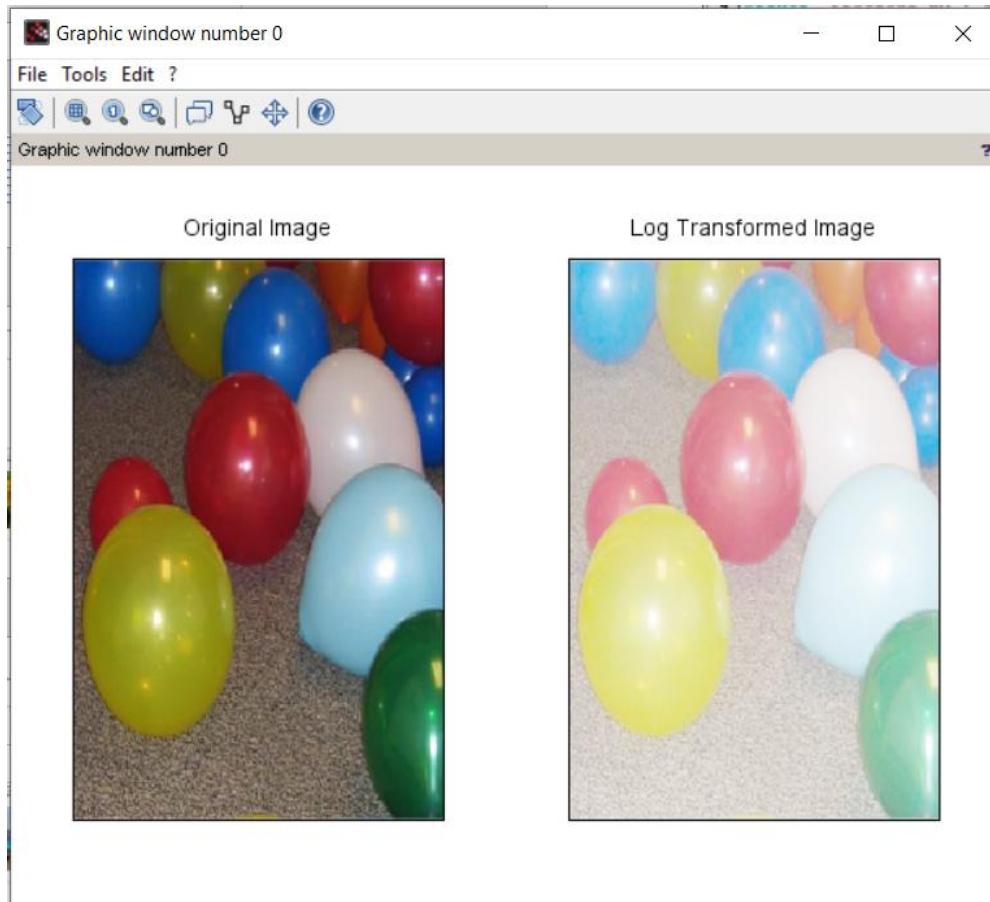


### 3: Program to perform Log transformation

#### Code:

```
clc;
clear;
Img2=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\balloons.png');
printf "Prctical 2A : 3 Log Transformation"
L=255;
C=L/log(1+L);
S=C*log(1+double(Img2));
subplot(1,2,1),
imshow(Img2),
title('Original Image');
subplot(1,2,2);
imshow(uint8(S));
title('Log Transformed Image');
```

#### Output:

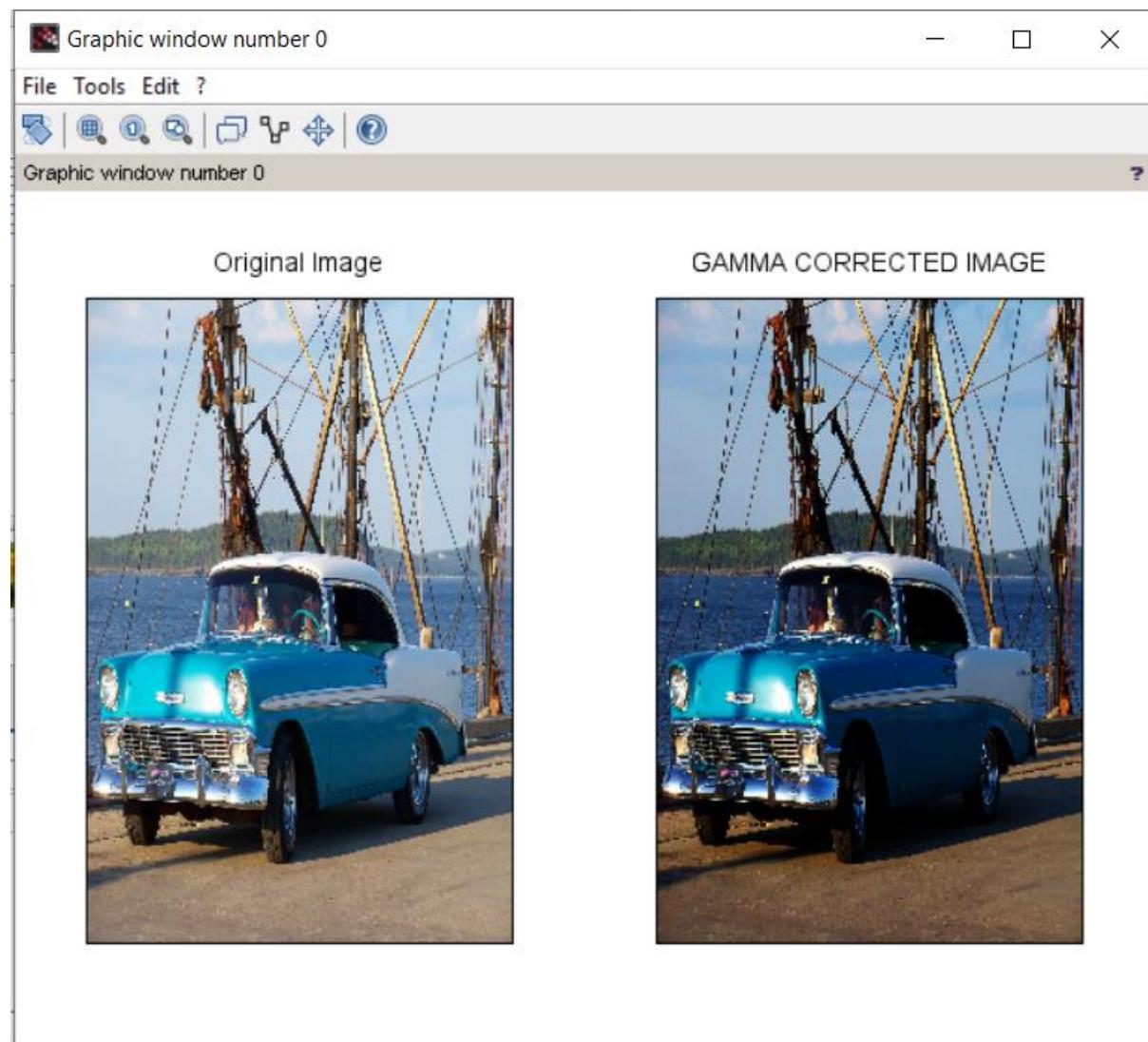


#### 4: Power-law transformations

**Code:**

```
clc;
clear;
Img3=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\vintage_car_cars_classic_car.jpg');
printf "Practical 2A: 4 Power Law Transformation"
r=double(Img3)/255;
c=1;
gamma=1.8;
s=c.*(r).^gamma;
subplot(1,2,1),
imshow(Img3);
title('Original Image');
subplot(1,2,2);
imshow(s);
title('GAMMA CORRECTED IMAGE');
```

**Output:**

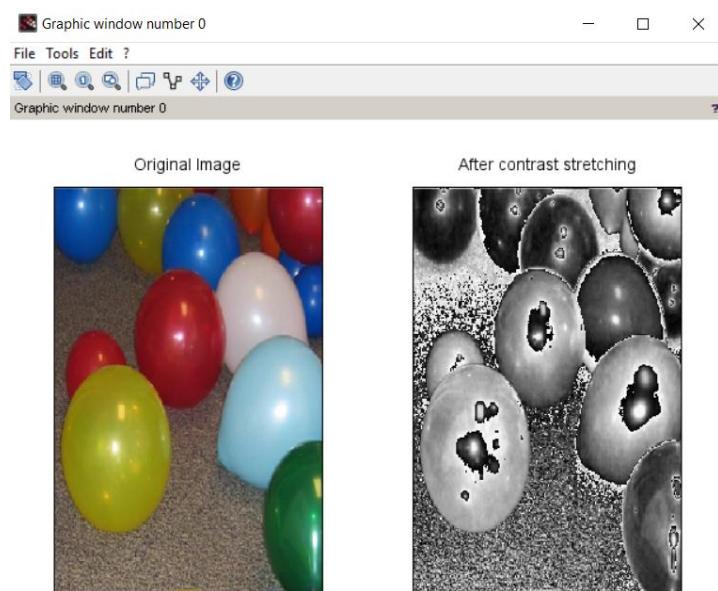


**A5:Piecewise liner transformations****[i]: Contrast Stretching****Code:**

```

clc;
clear;
disp("Contrast Stretching")
a=uigetfile('C:\Users\OneDrive\Documents\Rahul Kewat\IPCV\images\balloons.png');
a=imread(a);
a1=min(min(a));
b=max(max(a));
l=0.5;
n=0.5;
m=3;
v=l*a1;
w=v+(m*(b-a1));
[r,c]=size(a);
for i=1:r
for j=1:c
if (a(i,j)<a1)
new(i,j)=i*a(i,j);
else if(a(i,j)>=a1 & a(i,j)<b)
new(i,j)=(m*(a(i,j)-a1))+v;
else
new(i,j)=(n*(a(i,j)-b))+w;
end
end
end
subplot(1,2,1);
imshow(a);
title("Original Image");
subplot(1,2,2);new=double(new);imshow(uint8(new));title("After contrast stretching");
end

```

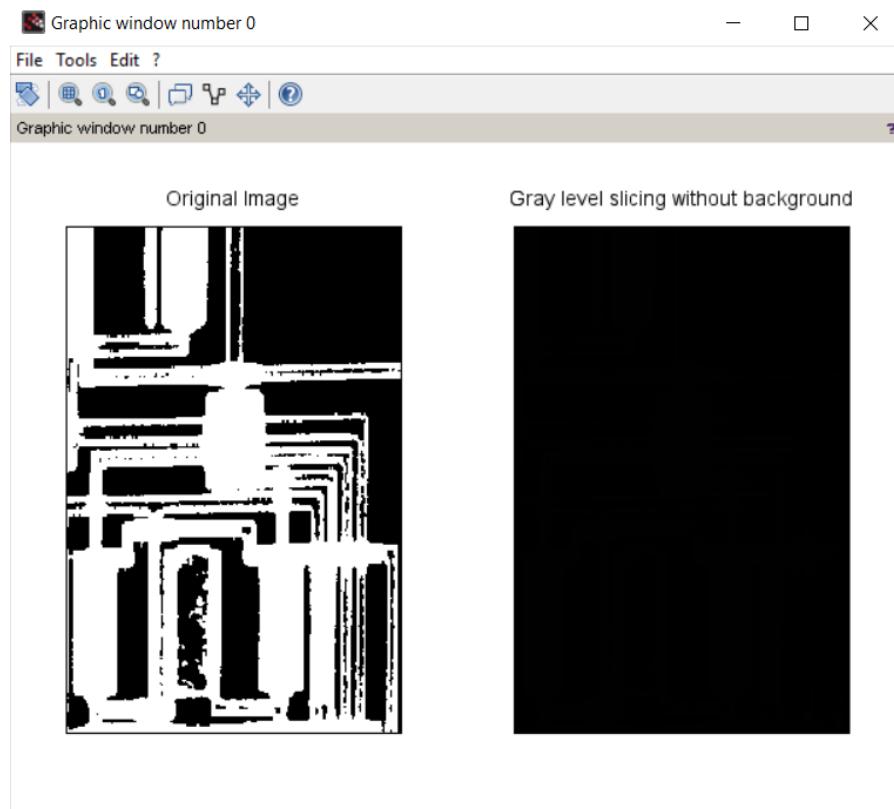
**Output:**

**[ii]: Gray-level slicing with and without background**

**Code:**

```
clc;
clear all;
printf("Practical 2A:5.2 Gray Level Slicing Without Background"
p=imread('C:\Users\OneDrive\Documents\Rahul Kewat\IPCV\images\circuit_bw.png');
z=double(p);
[row col]=size(p);
for i=1:1:row
for j=1:1:col
if(z(i,j)>50)&&(z(i,j)<150)
z(i,j)=255;
else
z(i,j)=p(i,j);
end
end
end
subplot(1,2,1);imshow(p);title('Original Image');
subplot(1,2,2);imshow(uint8(z));title('Gray level slicing without background');
```

**Output:**



**[iii] : Bit-plane slicing**

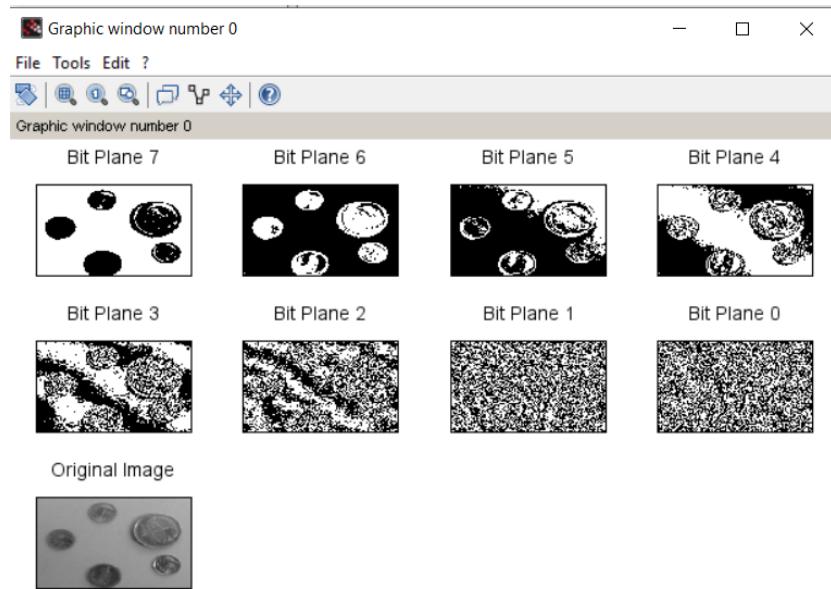
**Code:**

```
clc;
clear;
B=imread('C:\Users\OneDrive\Documents\Rahul Kewat\IPCV\images\coins_gray.jpg');
[r,c]=size(double(B));
disp("Practical 2A 5C")
for i=1:r
```

```

for j=1:c
MSB(i,j)=bitand(B(i,j),bin2dec('10000000'));
LSB(i,j)=bitand(B(i,j),bin2dec('00000001'));
Second(i,j)=bitand(B(i,j),bin2dec('01000000'));
Third(i,j)=bitand(B(i,j),bin2dec('00100000'));
Fourth(i,j)=bitand(B(i,j),bin2dec('00010000'));
Fifth(i,j)=bitand(B(i,j),bin2dec('00001000'));
Sixth(i,j)=bitand(B(i,j),bin2dec('00000100'));
Seventh(i,j)=bitand(B(i,j),bin2dec('00000010'));
end
end
subplot(4,4,1);imshow(MSB);title("Bit Plane 7");
subplot(4,4,2);imshow(Second);title("Bit Plane 6");
subplot(4,4,3);imshow(Third);title("Bit Plane 5");
subplot(4,4,4);imshow(Fourth);title("Bit Plane 4");
subplot(4,4,5);imshow(Fifth);title("Bit Plane 3");
subplot(4,4,6);imshow(Sixth);title("Bit Plane 2");
subplot(4,4,7);imshow(Seventh);title("Bit Plane 1");
subplot(4,4,8);imshow(LSB);title("Bit Plane 0");
subplot(4,4,9);imshow(B);title("Original Image");

```

**Output:**

### [B]: Histogram Processing

#### [1]: Program to plot the histogram of an image and categorise

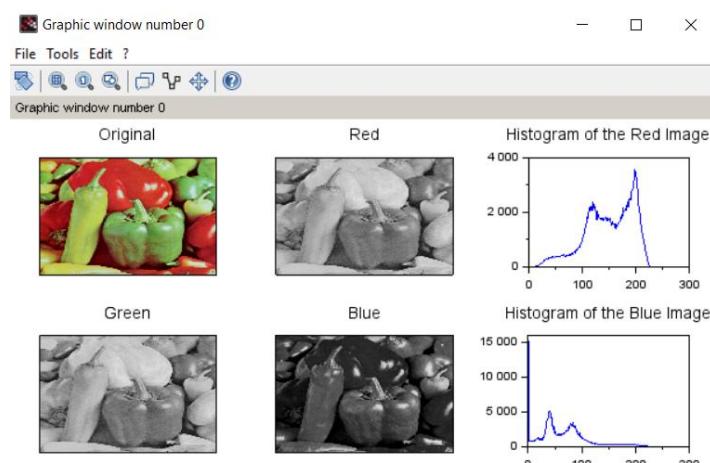
##### Code:

```

clc;
close;
original=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\peppers.png');
im_red=original(:,:,1);
im_green=original(:,:,1);
im_blue=original(:,:,3);
subplot(3,1),imshow(original),title('Original');
subplot(3,2),imshow(im_red),title('Red');
r=size(im_red,1); c=size(im_red,2); h=zeros(1,256);
for i=1:r
for j=1:c
if (im_red(i,j)==0)
im_red(i,j)=1;
end
k=im_red(i,j);
h(k)=h(k)+1;
end
end
subplot(3,3),plot(h); title('Histogram of the Red Image');
subplot(3,4),imshow(im_green),title('Green');
subplot(3,5),imshow(im_blue),title('Blue');
a=size(im_blue,1); b=size(im_blue,2); x=zeros(1,256);
for i=1:a
for j=1:b
if (im_blue(i,j)==0)
im_blue(i,j)=1;
end
y=im_blue(i,j);
x(y)=x(y)+1;
end
end
subplot(3,6),plot(x); title('Histogram of the Blue Image');

```

##### Output:



[C]: Write a program to perform convolution and correlation

**Code:**

```
clc; disp("Practical 2C 1 Convolution")
x = [4,5,6;7,8,9]; h = [1;1;1]; y = conv2(x,h); disp(y);
disp("Practical 2C 2DConvolution")
a = [1,2,3;4,5,6;7,8,9]; b = [1,1;1,1;1,1]; c = convol2d(a,b); disp(c);
disp("Practical 2C Correlation")
x = [1,2,3;4,5,6;7,8,9];
h = [1,1;1,1;1,1];
y = corr2(x,h);
disp(y);
```

**Output:**

Scilab 6.1.1 Console

```
"Practical 2C 1 Convolution"

4.      5.      6.
11.     13.     15.
11.     13.     15.
7.      8.      9.

"Practical 2C 2DConvolution"

1.      3.      5.      3.
5.      12.     16.     9.
12.     27.     33.     18.
11.     24.     28.     15.
7.      15.     17.     9.

"Practical 2C Correlation"

21.     21.     27.
27.     27.     33.
33.     33.     39.

-->
```

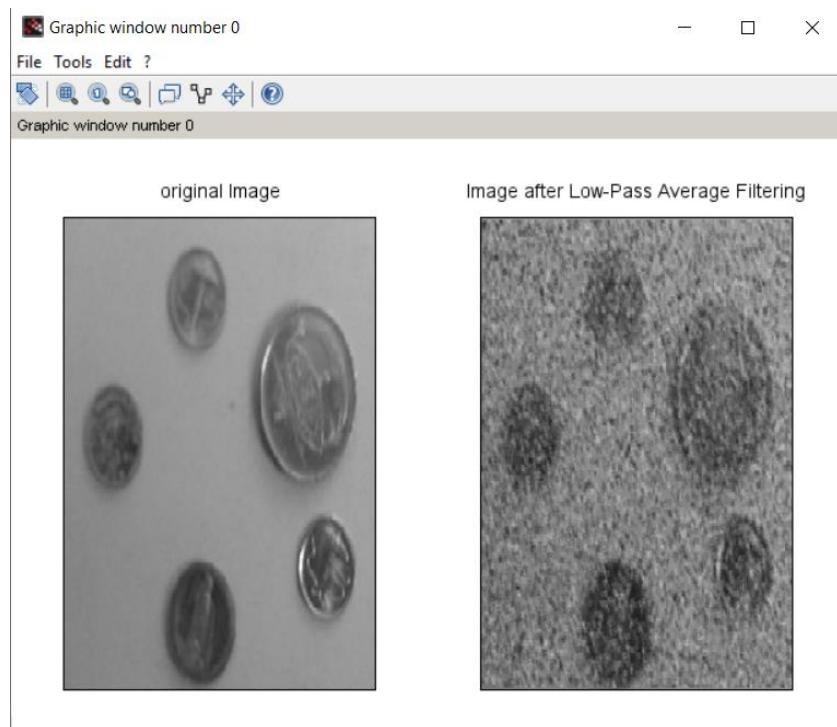
**D:** Write a program to apply smoothing and sharpening filters on grayscale and colour Images

### [i]: Low pass average Filtering

#### Code:

```
clc;
clear;
disp("Practical 2D A Low Pass Average Filtering")
a=uigetfile('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\coins_gray.jpg');
a=imread(a);
b=double(a);
c=imnoise(a,'salt & pepper',0.2);
d=double(c);
m=(1/9)*(ones(3,3));
[r1,c1]=size(a);
for i=1:r1
for j=1:c1
new(i,j)=a(i,j);
end
end
for i=2:r1-1
for j=2:c1-1
new(i,j)=(m(1)*d(i-1,j-1))+(m(2)*d(i-1,j))+(m(4)*d(i,j-1))+(m(5)*d(i,j))+(m(6)*d(i,j+1))+(m(7)*d(i+1,j-1))+(m(8)*d(i+1,j))+(m(9)*d(i+1,j+1));
end
end
subplot(1,2,1); imshow(a); title("original Image");
subplot(1,2,2); imshow(uint8(new));
title("Image after Low-Pass Average Filtering")
```

#### Output:

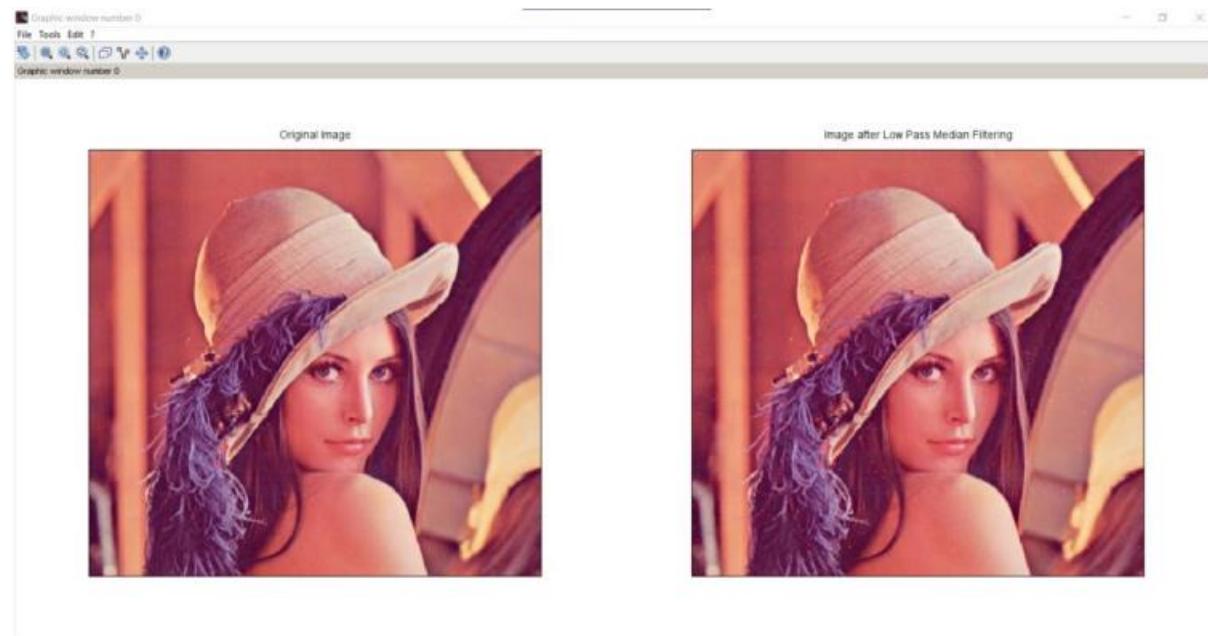


**[ii]: Low Pass Median Filtering****Code:**

```

clc;
clear;
disp("Practical 2D[i] Low Pass Median Filtering")
a = uigetfile("C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\lena.png");
a = imread(a);
b = double(a);
c = imnoise(a,'salt & pepper',0.2);
d = double(c);
m = (1/9) * (ones(3,3));
[r1,c1] = size(a);
for i=2:r1-1
for j=2:c1-1
a1 = [d(i-1,j-1) d(i-1,j) d(i-1,j+1) d(i,j-1) d(i,j) d(i,j+1) d(i+1,j-1) d(i+1,j) d(i+1,j+1)];
a2 = gsort(a1);
med = a2(4);
b(i,j) = med;
end
end
subplot(1,2,1);
imshow(a);
title("Original Image");
subplot(1,2,2);
imshow(uint8(b));
title("Image after Low Pass Median Filtering");

```

**Output:**

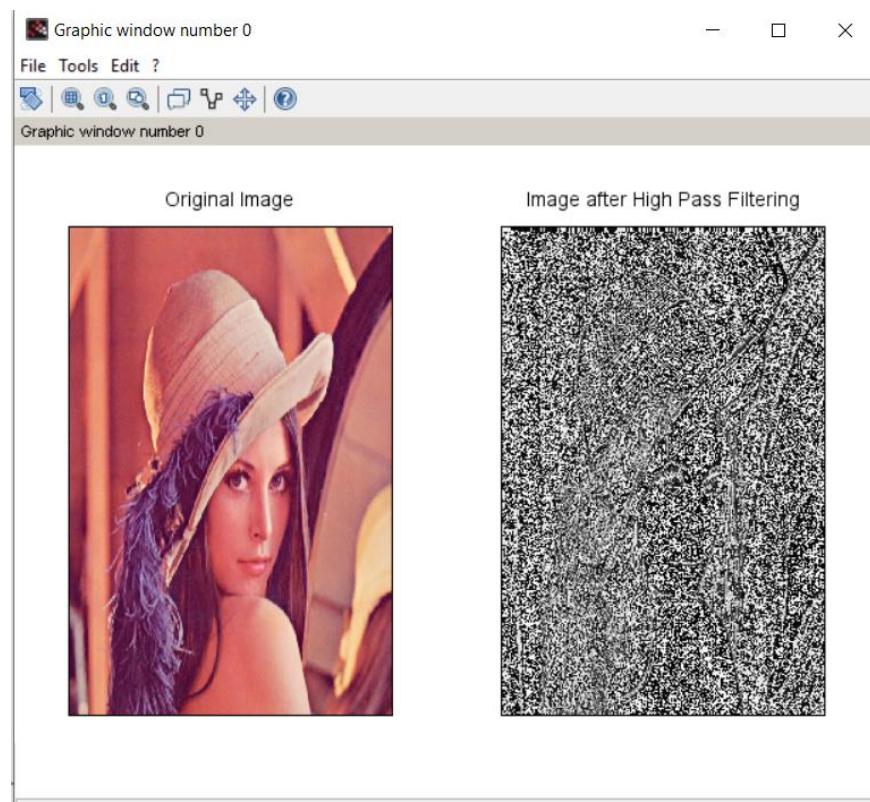
**[iii]: High Pass Filtering****Code:**

```

clc;
clear all;
close;
img = imread('C:\Users\DELL\OneDrive\Documents\Rahul Kewat\IPCV\images\lena.png');
f = double(img);
[m,n] = size(f);
w = [-1 -1 -1;-1 8 -1;-1 -1 -1];
for i = 2:m-1
for j = 2:n-1
b(i,j) = (w(1)*f(i-1,j+1)+w(2)*f(i,j+1)+w(3)*f(i+1,j+1)+w(4)*f(i-1,j)+w(5)*f(i,j)+w(6)*f(i+1,j)+w(7)*f(i-1,j-1)+w(8)*f(i,j-1)+w(9)*f(i+1,j-1));
end
end
subplot(1,2,1),
imshow(img),
title('Original Image');

subplot(1,2,2),
imshow(uint8(b)),
title('Image after High Pass Filtering');

```

**Output:**

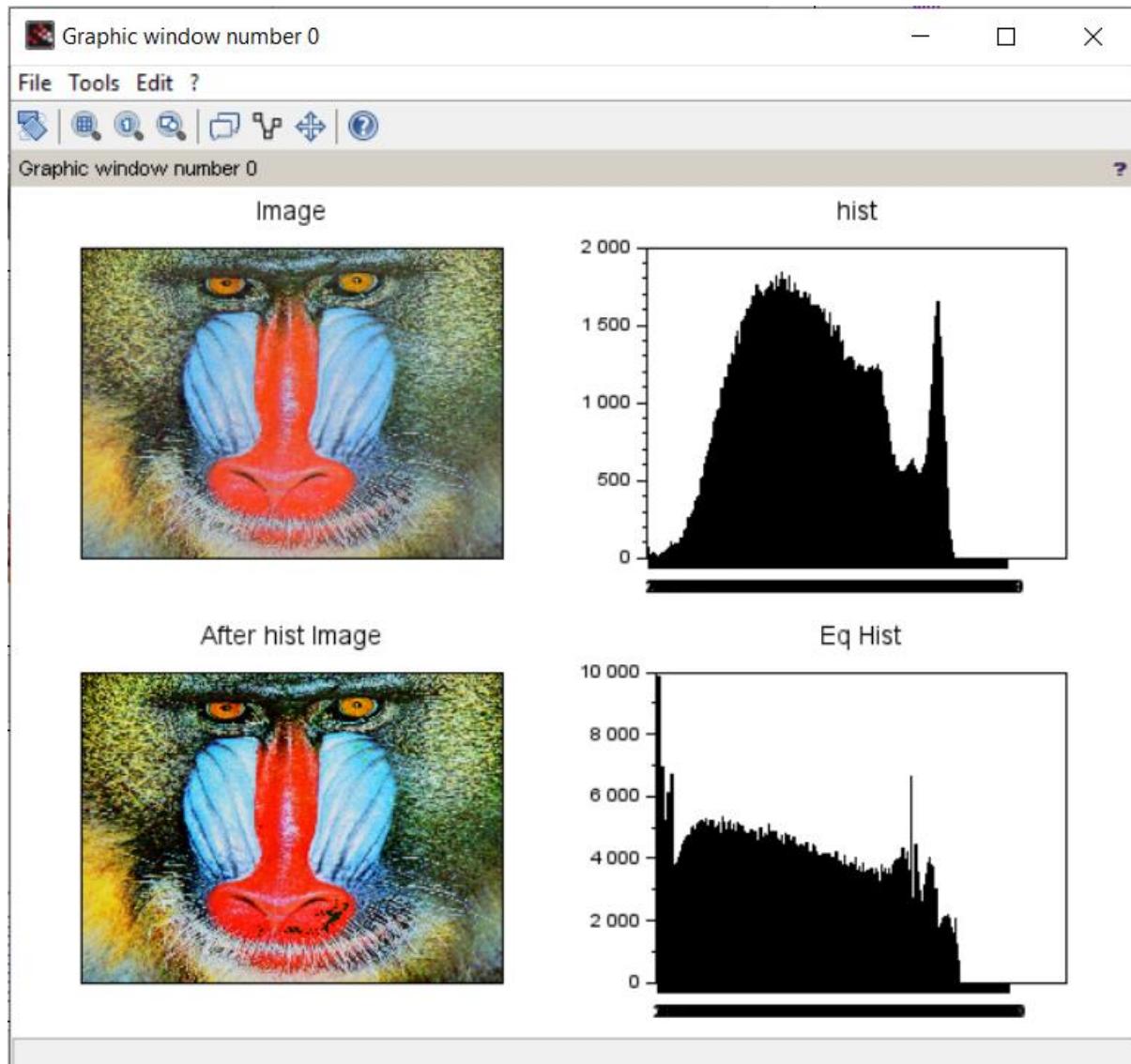
**[E]: Histogram Equalization****Code:**

```

a = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\baboon.png');
a = double(a);
big = 256;
[row col d] = size(a);
c = row*col;
h = zeros(1,300);
z = zeros(1,300);
for e = 1:1:d
    for n = 1:1:row
        for m = 1:1:col
            if a(n,m,e) == 0
                a(n,m,e) = 1;
            end
        end
    end
end
for n = 1:1:row
    for m = 1:1:col
        t = a(n,m);
        h(t) = h(t)+1;
    end
end
pdf = h/c;
cdf(1) = pdf(1);
for x = 2:1:big
    cdf(x) = pdf(x) + cdf(x-1);
end
new = round (cdf*big);
new = new + 1;
for r = 1:1:d
    for p = 1:1:row
        for q = 1:1:col
            temp = a(p,q,r);
            b(p,q,r) = new(temp);
            t = b(p,q,r);
            z(t) = z(t) + 1;
        end
    end
end
b = b-1;
subplot(2,2,1);imshow(uint8(a));title('Image');
subplot(2,2,2);bar(h);title('hist');subplot(2,2,3);imshow(uint8(b));
title('After hist Image');subplot(2,2,4);bar(z);title('Eq Hist');

```

**Output:**



**Practical 3**

**Aim:** : Filtering in Frequency Domain Program to apply Low pass and High pass filters in frequency domain.

**Writeup:**

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

**[A]:** Program to apply Discrete Fourier Transform on an image.

**Code:**

```
clc;
clear all;
a = zeros(30,30);

figure, subplot(2,2,1);
imshow(a);
title("Black Image");

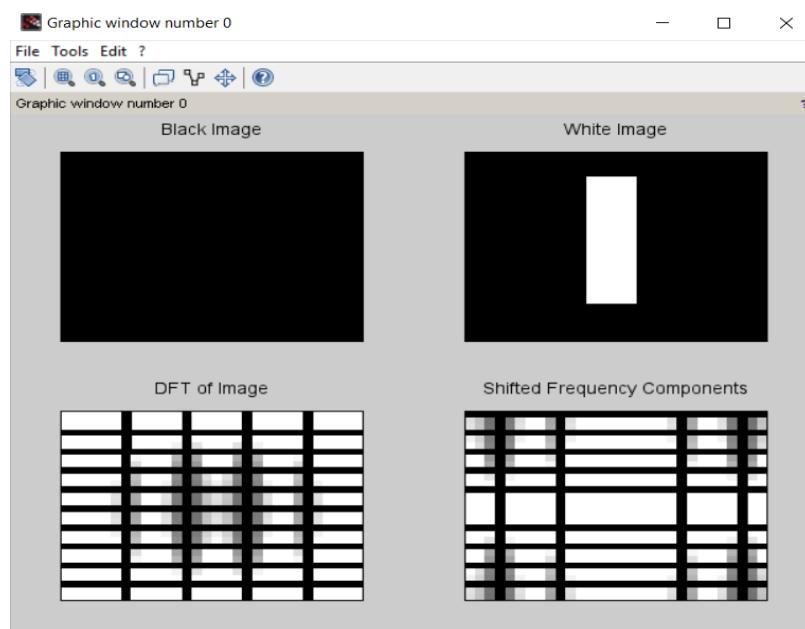
a(5:24,13:17) = 1;

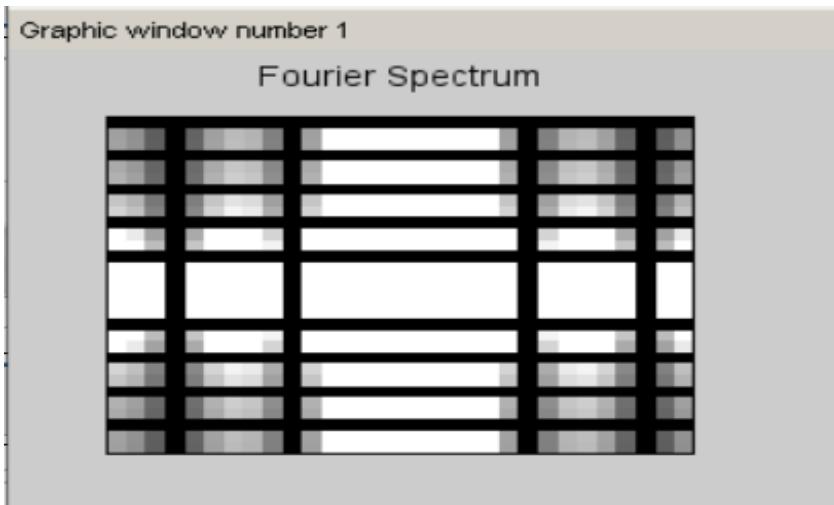
subplot(2,2,2);
imshow(a);
title("White Image");
F = fft2(a,256,256);
F2 = abs(F);

subplot(2,2,3);
imshow(F2);
title("DFT of Image");
F3 = fftshift(F);
F3 = abs(F3);

subplot(2,2,4);
imshow(F3);
title("Shifted Frequency Components");
F4 = log(1 + F3);
figure, subplot(2,2,1);
imshow(F4);
title("Fourier Spectrum");
```

**Output:**





### [B]: Ideal Low Pass Filter, Gaussian Low Pass Filter and High Pass Filter.

#### [i] Ideal Low Pass Filter:

**Code:**

```

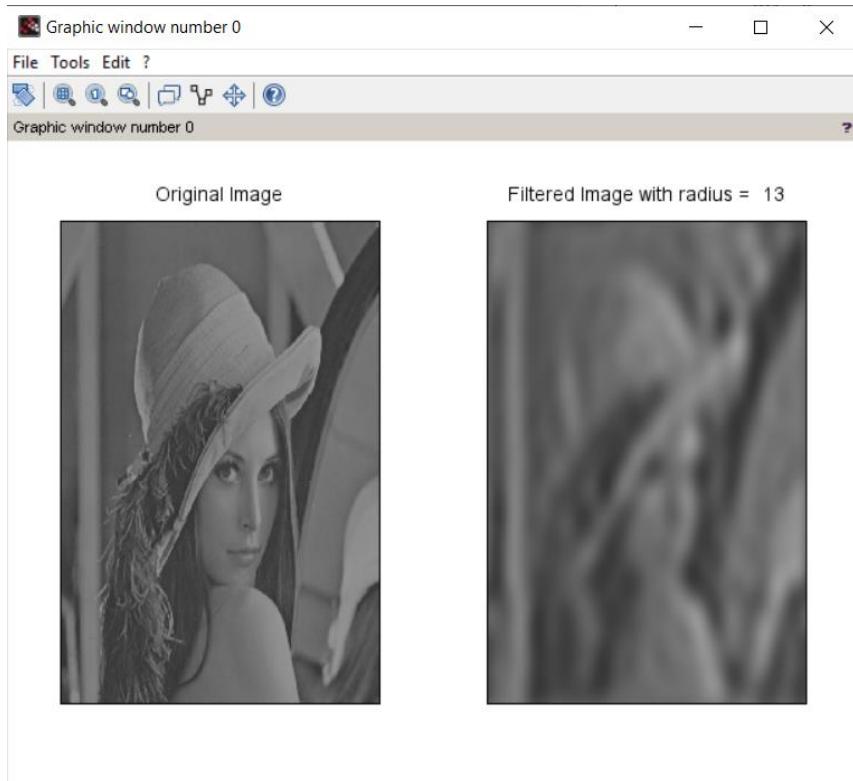
clear all;
clc;
img = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\Lena_dark.png');
a = double(img);
r = size(a,1);
c = size(a,2);
d0 = input("Enter the cutoff frequency (Radius) = ");

for u = 1:r
for v = 1:c
d = (((u-(r/2))^2) + ((v-(c/2))^2))^(0.5);
if d <= d0
h(u,v) = 1;
else
h(u,v) = 0;
end
end
end

b = fft2(a);
p_original = (abs(b))^2 + (atan(imag(b),real(b)))^2;
p_original = sum(sum(p_original));
c = fftshift(b);
c1 = uint16(c);
new = c.*h;
new2 = uint8(new);
new1 = abs(ifft(new));

subplot(1,2,1);
imshow(img);
title("Original Image");
subplot(1,2,2);
imshow(uint8(new1));
title(['Filtered Image with radius = ' string(d0)]);

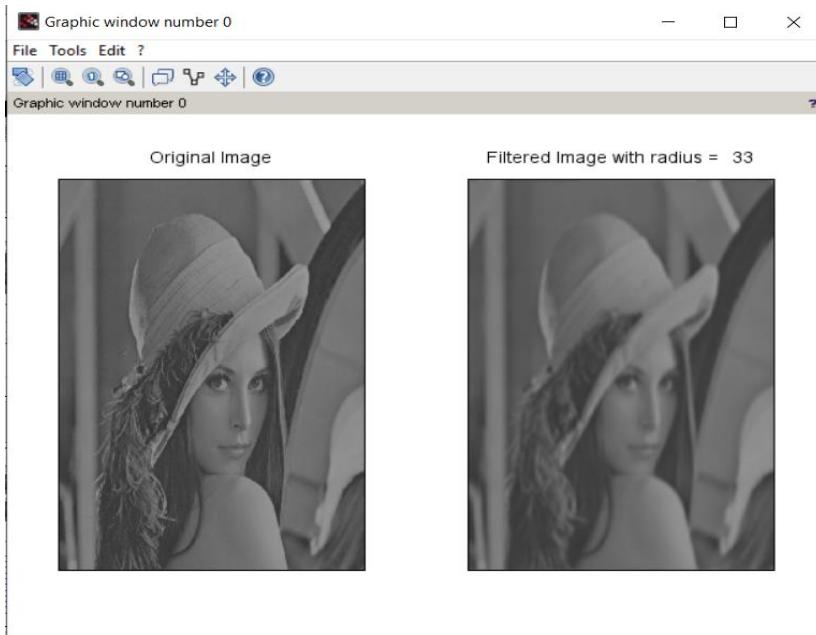
```

**Output:****[ii]: Gaussian Low Pass Filter :****Code:**

```

clear all;
clc;
img = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\Lena_dark.png');
a = double(img);
r = size(a,1);
c = size(a,2);
d0 = input("Enter the cutoff frequency (Radius) = ");
for u = 1:r
    for v = 1:c
        d = (((u - (r/2))^2) + ((v - (c/2))^2))^(0.5);
        dd = d*d;
        h(u,v) = exp(-dd/ (2*d0*d0));
    end
end
b = fft2(a);
c = fftshift(b);
c1 = uint16(c);
new = c.*h;
new2 = uint8(new);
new1 = abs(ifft(new));
subplot(1,2,1);
imshow(img);
title("Original Image");
subplot(1,2,2);
imshow(uint8(new1));
title(['Filtered Image with radius = ' string(d0)]);

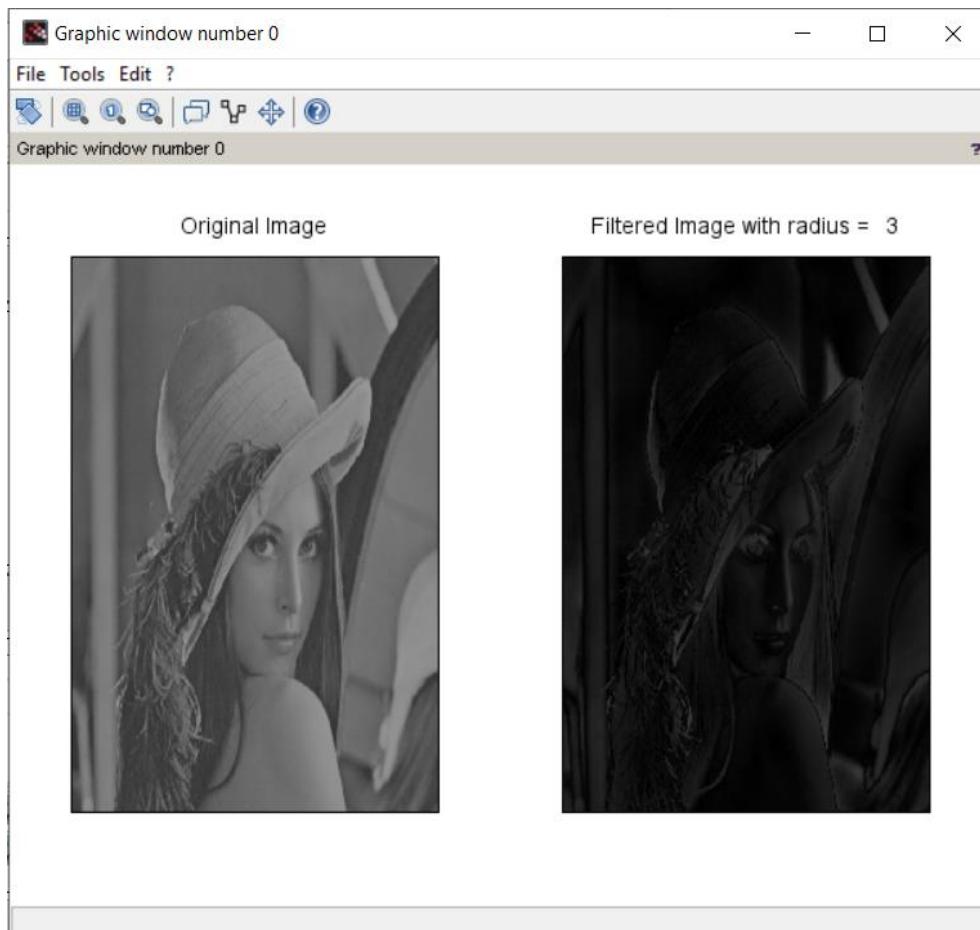
```

**Output:****[iii]: High Pass Filter****Code:**

```

clear all;
clc;
a = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\Lena_dark.png');
r = size(a,1);
c = size(a,2);
d0 = input("Enter the cutoff frequency (Radius) = ");
for u = 1:r
for v = 1:c
d = (((u-(r/2))^2) + ((v-(c/2))^2))^(0.5);
if d <= d0
h(u,v) = 0;
else
h(u,v) = 1;
end
end
b = fft2(double(a));
c = fftshift(b);
c1 = uint16(c);
new = c.*h;
new2 = uint8(new);
new1 = abs(ifft(new));
subplot(1,2,1);
imshow(img);
title("Original Image");
subplot(1,2,2);
imshow(uint8(new1));
title(['Filtered Image with radius = ' string(d0)]);

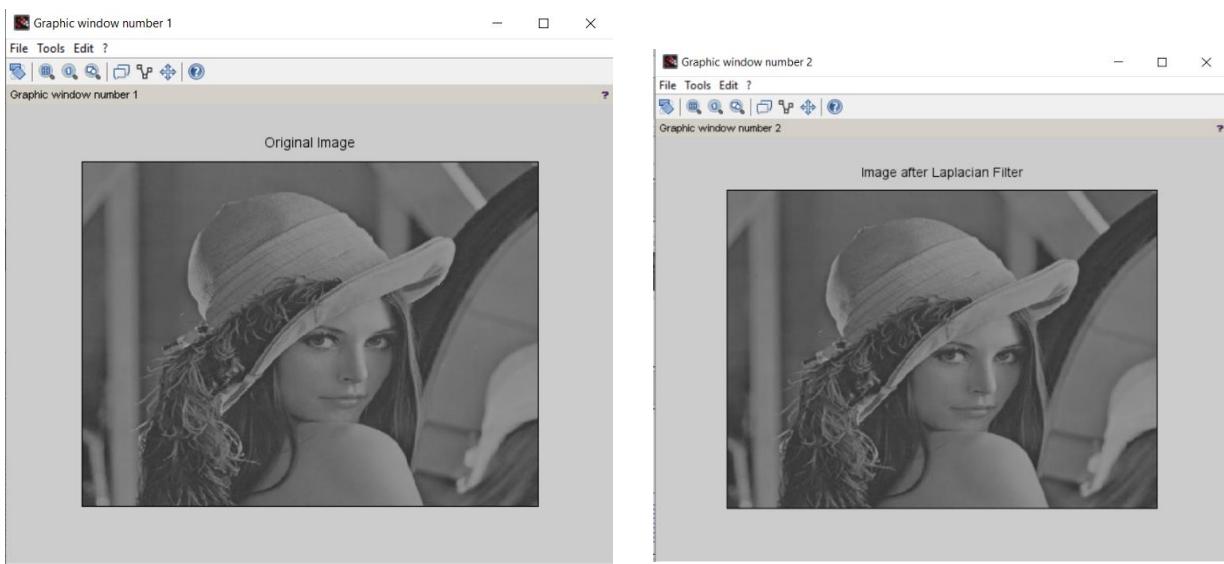
```

**Output:**

### [C]: Program to apply Laplacian filter in frequency domain

**Code:**

```
I = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\Lena_dark.png');
figure(1);
imshow(I);
title("Original Image");
H = fspecial('laplacian',0.2);
lap = imfilter(I,H,"replicate");
figure(2);
imshow(lap);
title("Image after Laplacian Filter");
```



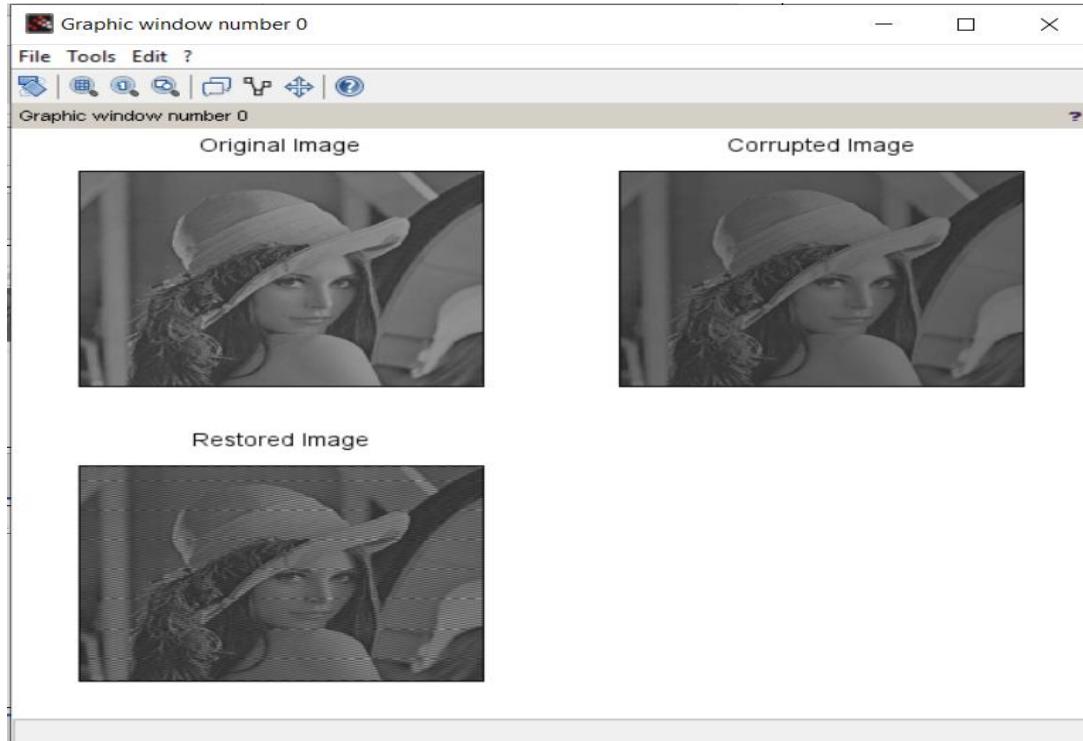
### [D]: Program for high frequency emphasis filtering, high boost, and homomorphic filtering.

#### [i]: High Frequency Emphasis Filtering

**Code:**

```
clear all;
clc;
I = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\Lena_dark.png');
GM = im2double(I);
t = 512;
p = 1/t;1;
for k = 1:t
l(k,:) = p();
end
product = GM.*5;
log_GM = log(GM + 1);
log_product = log(product + 1); 5
fft_GM = (fftsift(fft2(log_GM)));
fft_product = (fftsift(fft2(log_product)));
diff_ill = (fft_product - fft_GM);
restored = (fft_product - diff_ill);
restored_im = ifft(restored);
restored_image = exp(restored_im) - 1;
abs_restored_image = abs(restored_image);
subplot(2,2,1);
imshow(uint8(I));
```

```
title("Original Image");
subplot(2,2,2);
imshow(log_product);
title("Corrupted Image");
subplot(2,2,3);
imshow(abs_restored_image);
title("Restored Image");
```

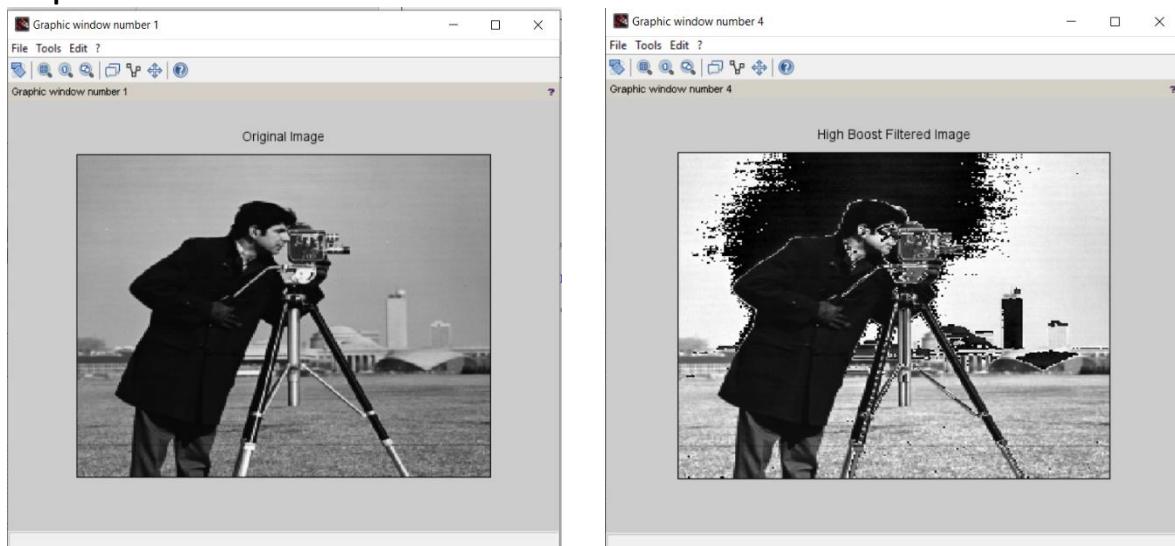
**Output:****[ii]: High Boost****Code:**

```
clc;
close all;
clear all;
a = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\cameraman.png');
a = double(a);
[row,col] = size(a);
D0 = 1 ;
for u = 1:row
for v = 1:col
D = ((u-(row/2))^2 + (v - (col/2))^2)^0.5;
if D < D0
H(u,v) = 0;
H = H(u,v);
else
H(u,v) = 1;
H = H(u,v);
end
end
end
A = 1.5;
H1 = (A-1) + H;
```

```

ftImg = fft2(a);
shftImg = fftshift(ftImg);
x = shftImg*H;
x1 = shftImg*H1;
X = abs(fft(x));
X1 = abs(fft(x1));
figure(1);
imshow(uint8(a));
title('Original Image');
//figure(2);
//mesh(H);
//title('Frequency Response of high pass filter');
figure(3);
imshow(uint8(X));
title('High pass filtered Image');
figure(4);
imshow(uint8(X1));
title('High Boost Filtered Image');

```

**Output:**

### [iii]: Homomorphic Filtering

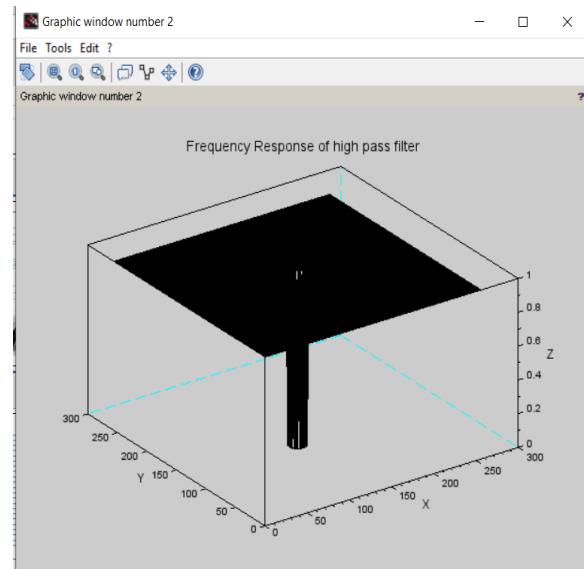
**Code:**

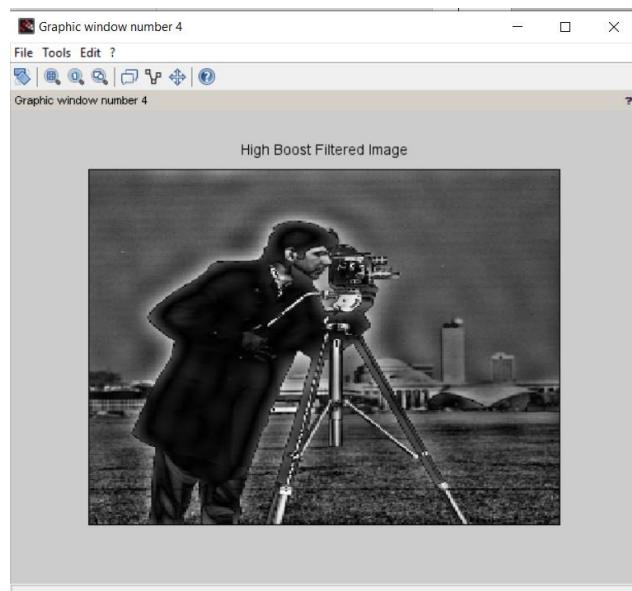
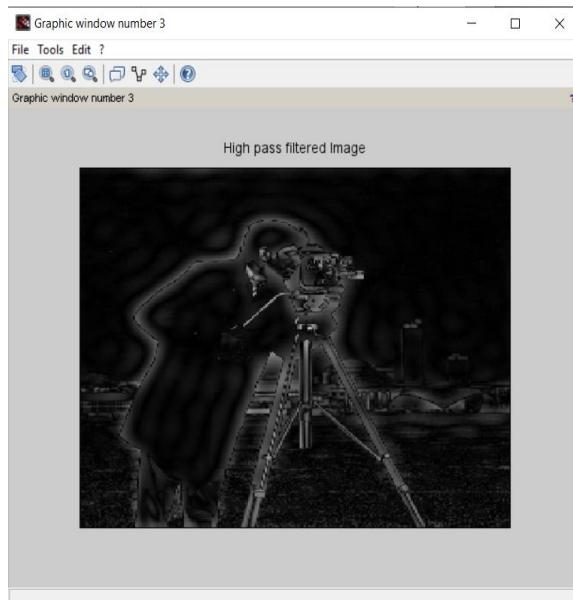
```

clc;
clear all;
a=imread("C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\cameraman.png");
a = double(a);
[row col] = size(a);
D0 = input('Enter cut-off frequency : ');
for u = 1:row
    for v = 1:col
        D = ((u-(row/2))^2 + (v-(col/2))^2)^0.5;
        if D<D0 //High pass filter
            H(u,v) = 0;
        else
            H(u,v) = 1;
        end
    end
end
A=1.5; //high boost factor
H1 = (A-1)+H; //high boost filter
ftImg = fft2(a); shftImg = fftshift(ftImg);
x = shftImg.*H; //multiply by high pass filter
x1 = shftImg.*H1; //multiply by high boost filter
X = abs(ifft(x));
X1 = abs(ifft(x1));
figure(1);
imshow(uint8(a));
title('Original Image');
figure(2);
mesh(H);
title('Frequency Response of high pass filter');
figure(3);
imshow(uint8(X));
title('High pass filtered Image');
figure(4);
imshow(uint8(X1));
title('High Boost Filtered Image');

```

**Output:**





**Practical 4**

**Aim:** Image Denoising

**Writeup:**

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

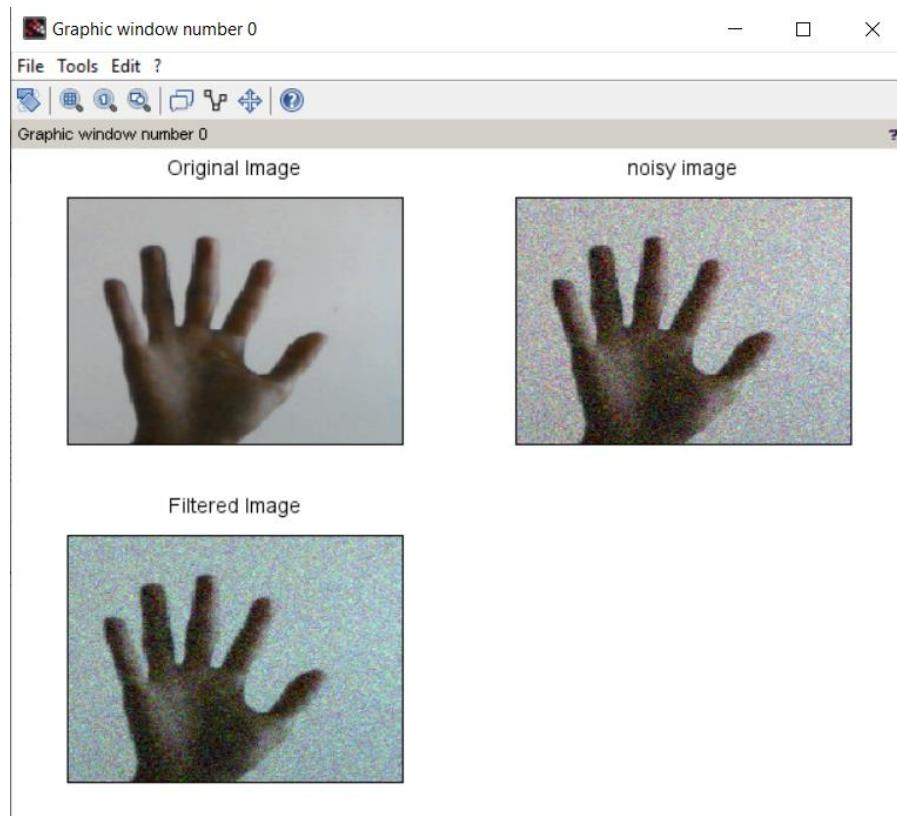
---

**[A]:**Program to denoise using spatial mean filtering.

**Code:**

```
clc;
clear all;
a=uigetfile('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\hand.jpg');
a=imread(a);
b1=double(a);
c=imnoise(a,'gaussian');
d=double(c);
b=d;
m=(1/9)*(ones(3,3));
[r1,c1]=size(a);
for i=2:r1-1
    for j=2:c1-1
        a1=d(i-1,j-1)+d(i-1,j)+d(i-1,j+1)+d(i,j-1)+d(i,j)+d(i,j+1)+d(i+1,j-1)+d(i+1,j)
        +d(i+1,j+1);
        b(i,j)=a1*(1/9);
    end
end
subplot(2,2,1);
imshow(uint8(a));
title('Original Image')
subplot(2,2,2);
imshow(uint8(c));
title('noisy image')
subplot(2,2,3);
imshow(uint8(b));
title('Filtered Image');
```

**Output:**

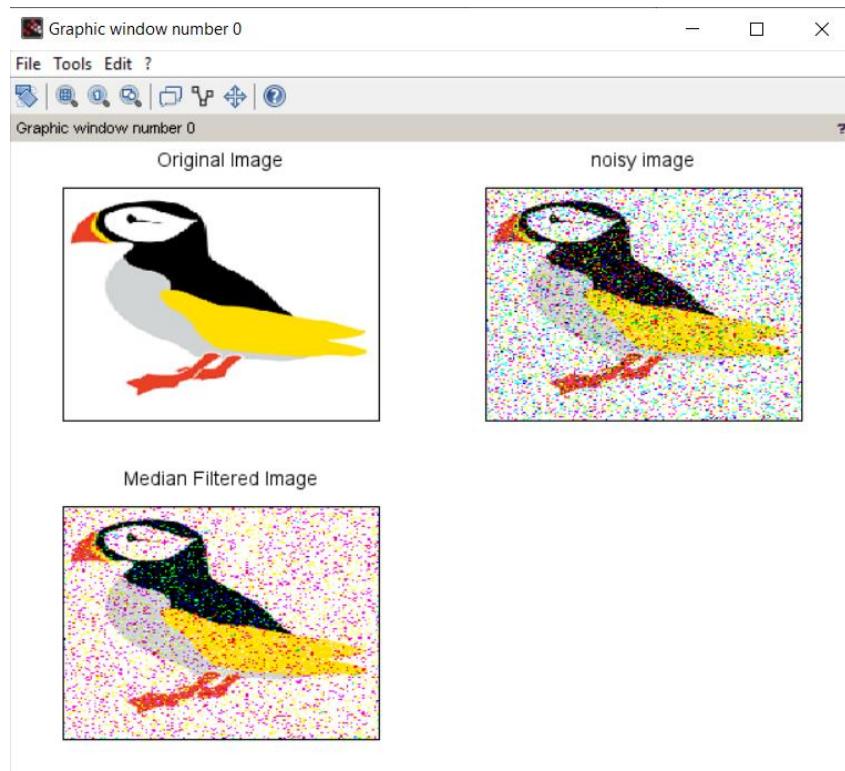


**[B]:** Program to denoise using Median filtering.

**Code:**

```
clc;
clear all;
a=uigetfile('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\puffin.png');
a=imread(a);
b1=double(mlb_double(a));
c=imnoise(a,"salt & pepper",0.2);
d=double(mlb_double(c));
b=d;
m=(1/9)*ones(3,3);
[r1,c1]=size(mlb_double(a));
for i=2:r1-1
for j=2:c1-1
a1=[d(i-1,j-1),d(i-1,j),d(i-1,j+1),d(i,j-1),d(i,j),d(i,j+1),d(i+1,j-1),d(i+1,j),d(i+1,j+1)];
a2=gsort(a1,"g","i");
med=a2(5);
b(i,j)=med;
end;
end;
b=uint8(b);
subplot(2,2,1);
imshow(uint8(a));
title('Original Image')
subplot(2,2,2);
imshow(uint8(c));
title('noisy image')
subplot(2,2,3);
imshow(b);
title("Median Filtered Image");
```

**Output:**



**[C]: Program for Image deblurring using inverse, Weiner filters**

**Code:**

```

clc;
clear all;
Ioriginal=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\hand.jpg');

subplot(2,2,1);
imshow(Ioriginal)
title('Original Image')

PSF=fspecial('motion',21,11);
Idouble=im2double(Ioriginal);
imf=imfilter(Idouble,PSF,'circular','full');

subplot(2,2,2);
imshow(imf)
title('Blurred Image')

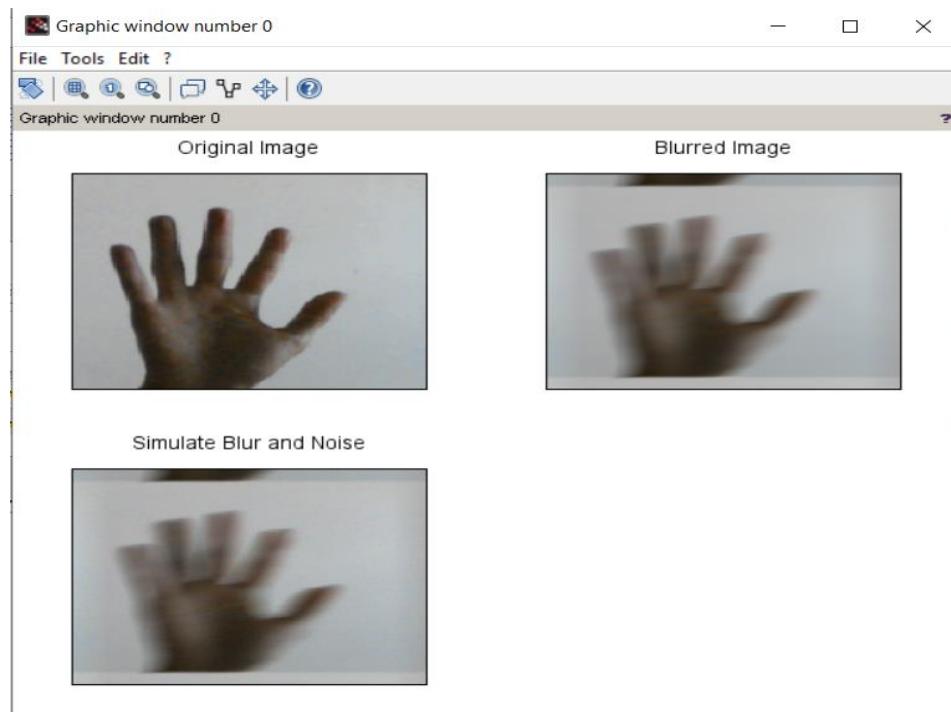
noise_mean=0;
noise_var=0.0001;
blurred_noisy=imnoise(imf,'gaussian',noise_mean,noise_var);

subplot(2,2,3);
imshow(blurred_noisy)
title('Simulate Blur and Noise')

estimated_nar=noise_var;
wnr3=imdeconvwiener(blurred_noisy,PSF,estimated_nsr);
subplot(2,2,4);
imshow(wnr3)
title('Restoration of Blurred, Noisy Image Using Estimated NSR');

```

**Output:**



### Practical 5

**Aim:** Colour Image Processing

**Writeup:**

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

**[A]:** Program to read a color image and segment into RGB planes , histogram of color image

**Code:**

```
clc;
clear all;
original=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\peppers.png');
im_red=original(:,:,1);
im_green=original(:,:,1);
im_blue=original(:,:,3);
subplot(3,3,1),imshow(original),title('Original');
subplot(3,3,2),imshow(im_red),title('Red');
r=size(im_red,1);c=size(im_red,2);h=zeros(1,256);
for i=1:r
for j=1:c
if (im_red(i,j)==0)
im_red(i,j)=1;
end
k=im_red(i,j);
h(k)=h(k)+1;
end
end
subplot(3,3,3),plot(h),title('Histogram of Red Image');
subplot(3,3,4),imshow(im_green),title('Green');
subplot(3,3,5),imshow(im_blue),title('Blue');
a=size(im_blue,1);b=size(im_blue,2);x=zeros(1,256);
for i=1:a
for j=1:b
if (im_blue(i,j)==0)
im_blue(i,j)=1;
end
y=im_blue(i,j);
x(y)=x(y)+1;
end
end
subplot(3,3,6),plot(x),title('Histogram of the Blue Image');
```

**Output:**



**[B]:** Program for converting from one colour model to another model

**Code:**

```

clc;
clear all;
a = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\Lilium.JPG');
figure(1),imshow(a);
title('Original Image');

k=rgb2ntsc(a);
figure(2),
imshow(k);
title('RGB TO NTSC');

l=rgb2ycbcr(a);
figure(3),imshow(l);
title('RGB TO YCbCr');

m=imcomplement(a);
figure(4),imshow(m);
title('RGB TO CMY');

imr=a(:,:,1);
img=a(:,:,2);
imb=a(:,:,3);
figure(5),
imshow(imr);

figure(6),
imshow(img);

figure(7),
imshow(imb);
I=(imr+img+imb)/3;
[m,n]=size(imr);
for c=1:m
    for d=1:n
        min1=min(imr(c,d),img(c,d));
        min2=min(min1,imb(c,d));
        S(c,d) = 1-(3/(imr(c,d)+img(c,d)+imb(c,d)))*min2;
    end
end

for c=1:m
    for d=1:n
        temp=(0.5*(imr(c,d)-img(c,d))+(imr(c,d)-
        imb(c,d))/sqrt(double(imr(c,d)*imr(c,d)+(imr(c,d)*imb(c,d))*(img(c,d)-imb(c,d)))));
        H(c,d)=acos(double(temp));
    end
end

for c=1:m
    for d=1:n
        finali(c,d,1)=I(c,d);
        finali(c,d,2)=S(c,d);
        finali(c,d,3)=H(c,d);
    end
end
figure(8),
imshow(finali);
title('Final image');

```

**Output:**

**[C]: Program to apply false colouring(pseudo) on a Gray scale image**

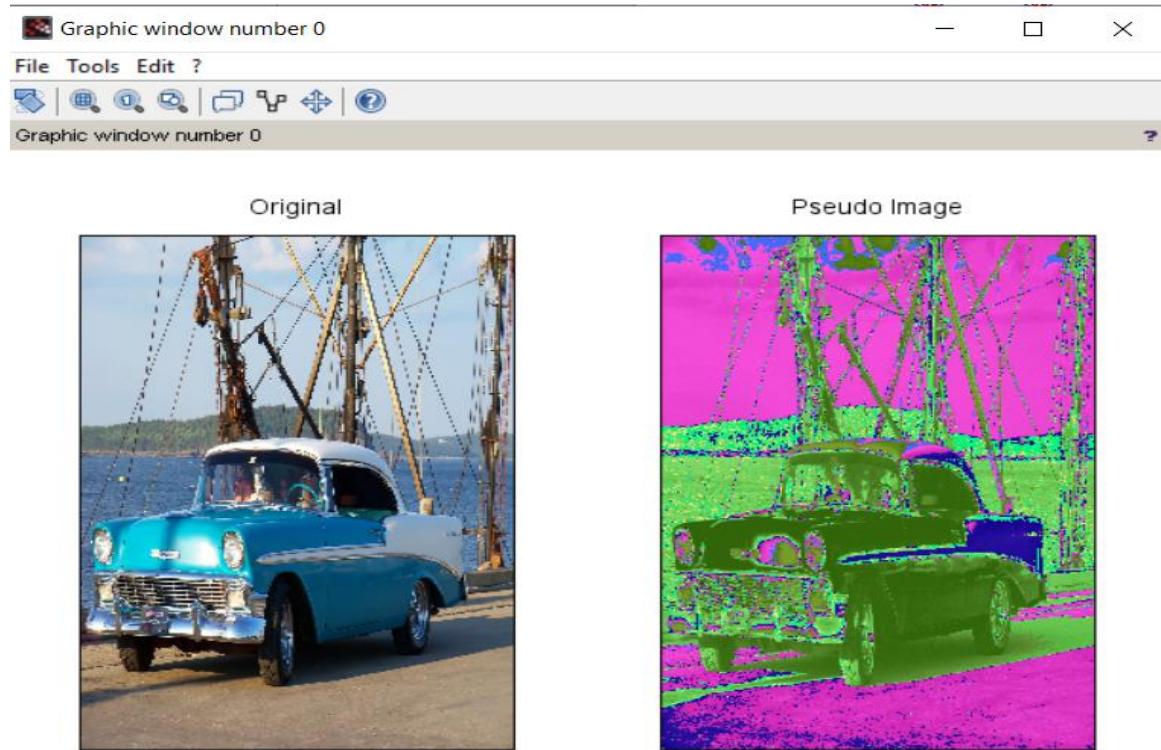
**Code:**

```

clc;
close;
a=imread('C:\Users\OneDrive\Documents\Rahul Kewat\IPCV\images\vintage_car_cars_classic_car.jpg');
[l,m,n]=size(a);
for i=1:l
    for j=1:m
        for k=1:n
            if a(i,j)>=0 & a(i,j) < 50
                b(i,j,1)=a(i,j,1)+50;
                b(i,j,2)=a(i,j,1)+100;
                b(i,j,3)=a(i,j,1)+10;
            end
            if a(i,j)>=50 & a(i,j) < 100
                b(i,j,1)=a(i,j,1)+35;
                b(i,j,2)=a(i,j,1)+128;
                b(i,j,3)=a(i,j,1)+10;
            end
            if a(i,j)>=100 & a(i,j) < 150
                b(i,j,1)=a(i,j,1)+152;
                b(i,j,2)=a(i,j,1)+130;
                b(i,j,3)=a(i,j,1)+15;
            end
            if a(i,j)>=150 & a(i,j) < 200
                b(i,j,1)=a(i,j,1)+50;
                b(i,j,2)=a(i,j,1)+140;
                b(i,j,3)=a(i,j,1)+25;
            end
            if a(i,j)>=200 & a(i,j) < 256
                b(i,j,1)=a(i,j,1)+120;
                b(i,j,2)=a(i,j,1)+160;
                b(i,j,3)=a(i,j,1)+45;
            end
        end
    end
end
subplot (1,2,1),imshow(a),title('Original');
subplot (1,2,2),imshow(b),title('Pseudo Image');

```

**Output:**



**Practical 6**

**Aim:** Fourier Related Transformation

**Writeup:**

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

**[A]: Program to compute Discrete Cosine Transformation.****Code:**

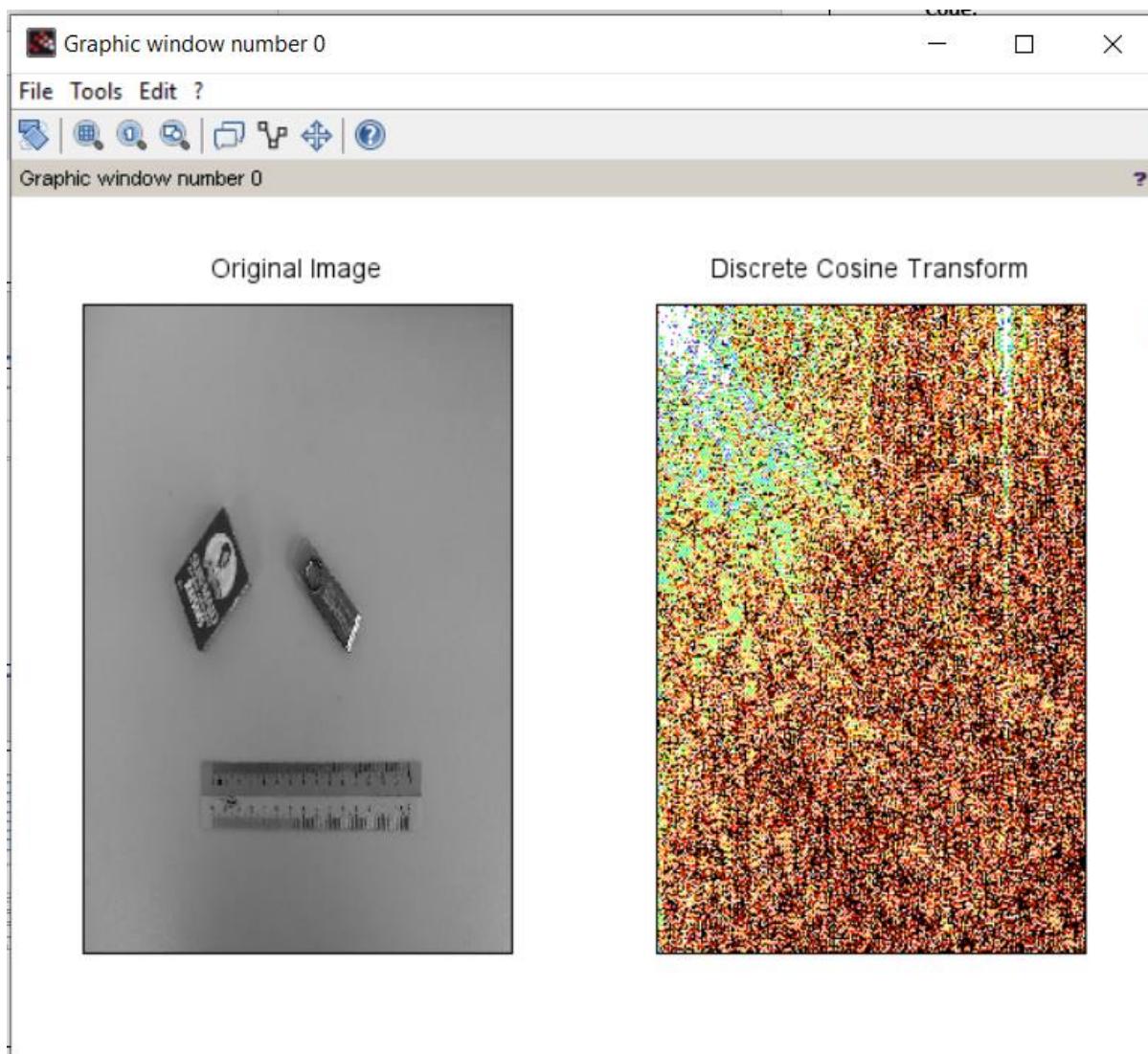
```
clc;
clear all;

img=imread('C:\Users\OneDrive\Documents\Rahul Kewat\IPCV\images\measure_gray.jpg');

dct_img = imdct(img);

subplot(1,2,1);
imshow(img);
title('Original Image');

subplot(1,2,2);
imshow(dct_img,rainbowcolormap(32));
title('Discrete Cosine Transform');
```

**Output:**

### Practical 7

**Aim:** Morphological Image Processing

**Writeup:**

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

**[A]: Program to apply erosion, dilation, opening, closing Opening.**

**Code:**

```
clear;
clc;
img=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\cameraman.png');
subplot(2,3,1),imshow(img),title('originalimage');
se = imcreatestruct('rect',3,3);

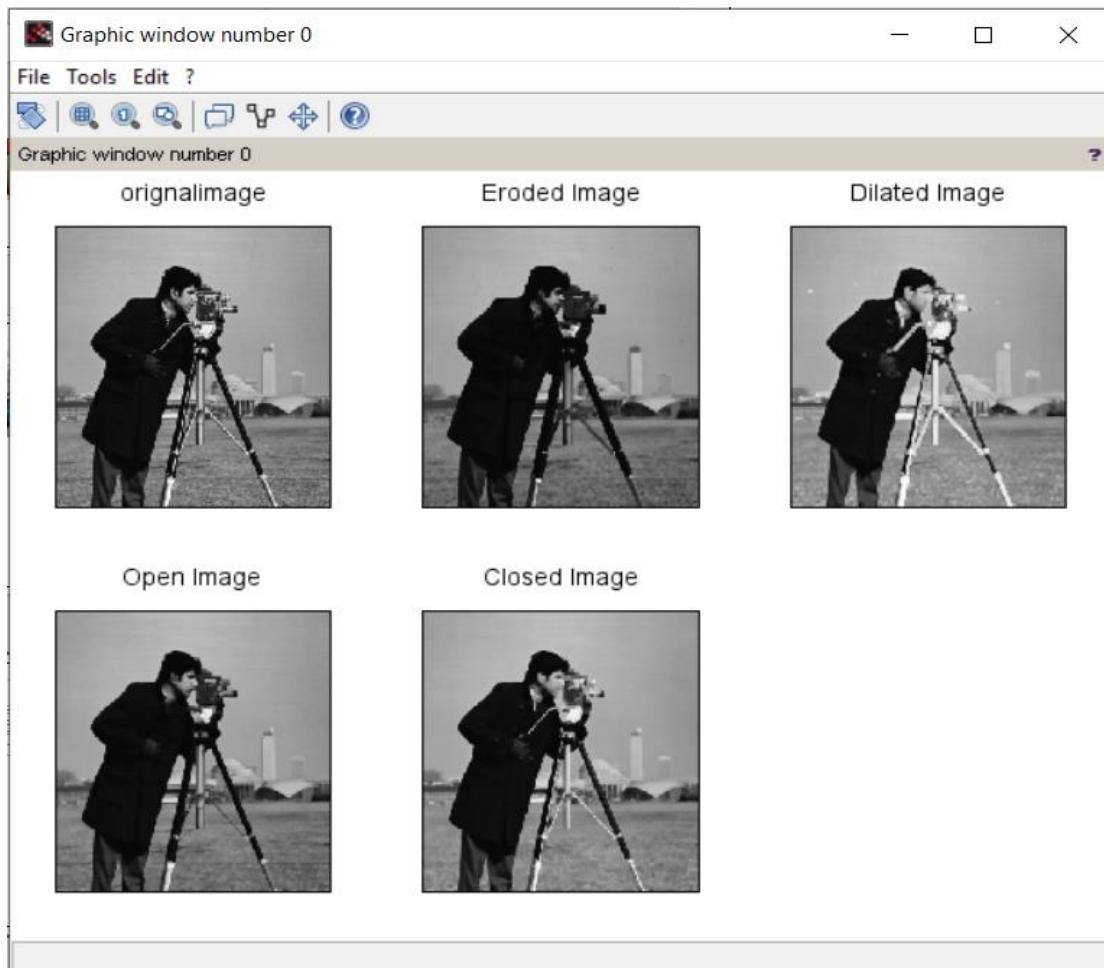
img_Erode = imerode(img,se);
subplot(2,3,2),imshow(img_Erode),title('Eroded Image');

img_Dilate = imdilate(img,se);
subplot(2,3,3),imshow(img_Dilate),title('Dilated Image');

open = imopen(img,se);
subplot(2,3,4),imshow(open),title('Open Image');

closed = imclose(img,se);
subplot(2,3,5),imshow(closed),title('Closed Image');
```

**Output:**



**[B]:** Program for detecting boundary of an image.

**Code:**

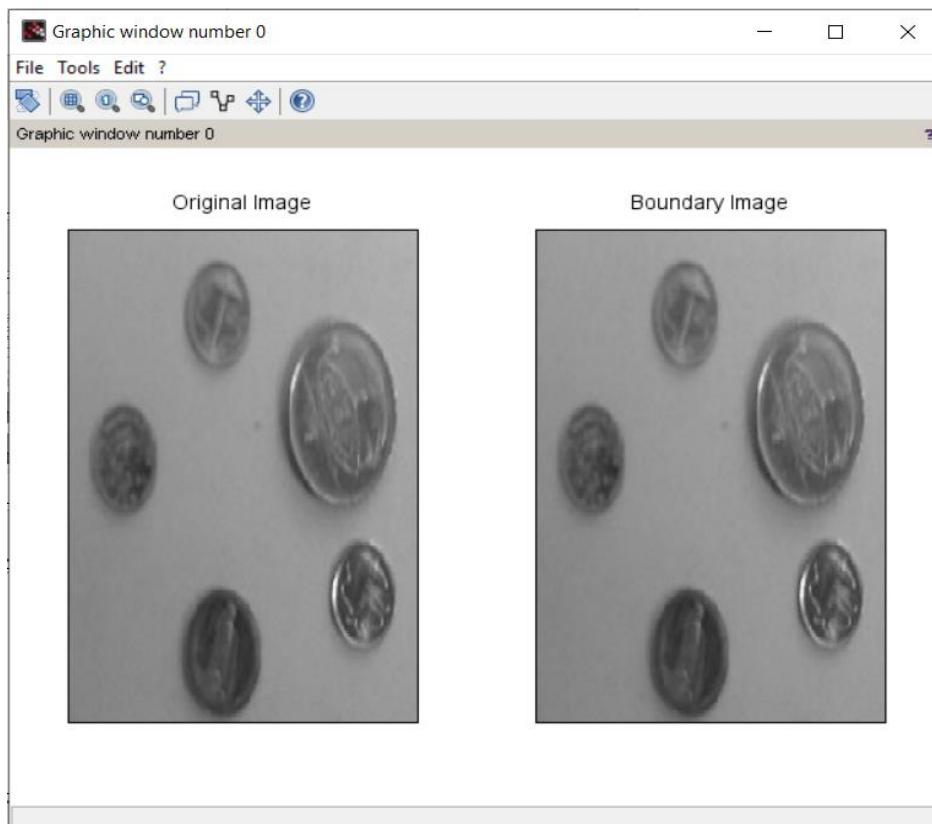
```

clc;
clear;
a = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\coins_gray.jpg');
img = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\coins_gray.jpg');
subplot(1,2,1),imshow(a),title("Original Image");
subplot(1,2,2),imshow(img),title("Boundary Image");
BW = im2bw(img);
imshow(BW);
dim = size(BW)
col = round(dim(2)/2)-90;
row = min(find(BW(:,col)));
boundary = bwtraceboundary(BW,[row, col],'N');
imshow(img);
plot(boundary(:,2),boundary(:,1),'g','LineWidth',3);
BW_filled = imfill(BW,'holes');
boundaries = bwboundaries(BW_filled);

for k=1:10
    b = boundaries{k};
    plot(b(:,2),b(:,1),'g','LineWidth',3);
end

```

**Output:**



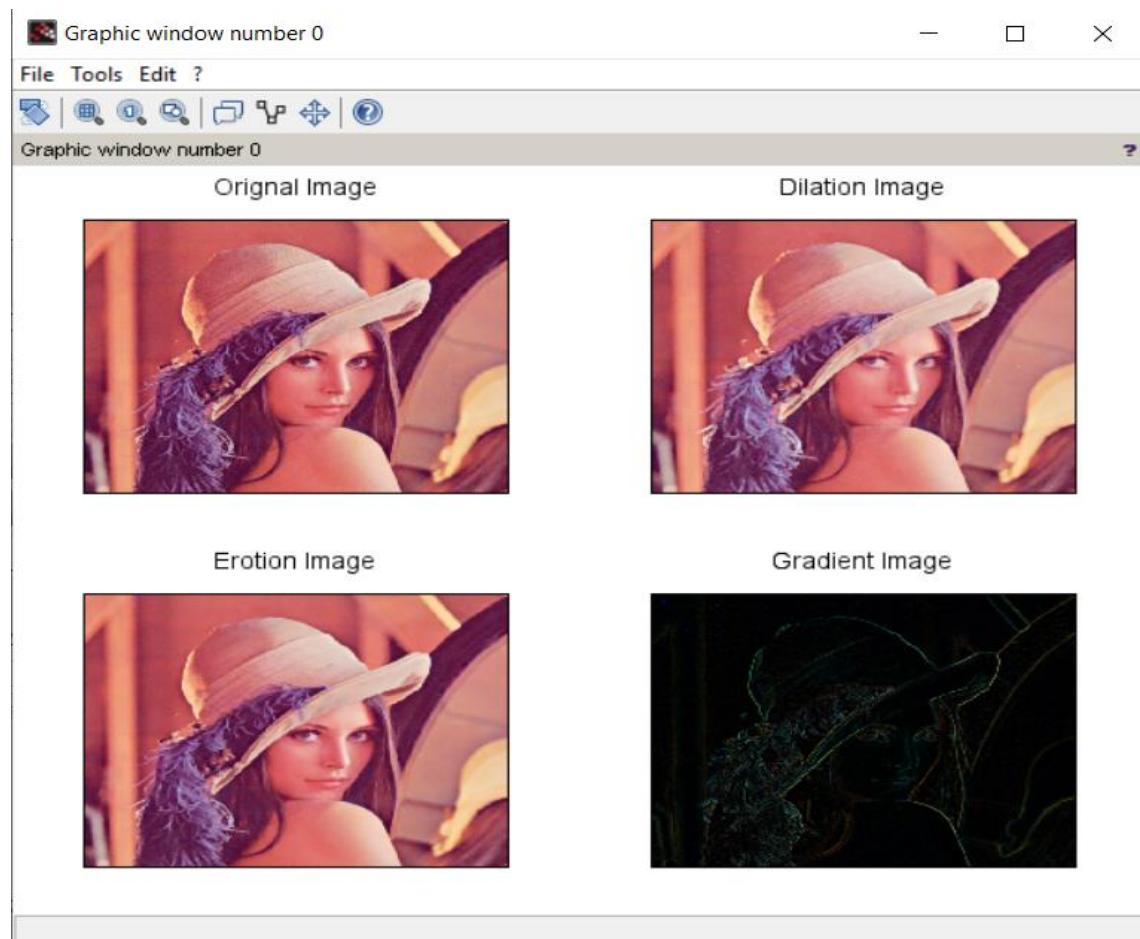
[C]: Program to apply morphological gradient on an image.

**Code:**

```
clc;
clear;
img=imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\lena.png');
se = imcretese('rect',3,3);
img_dilate = imdilate(img,se);
img_eroode = imerode(img_dilate,se);
g=img_dilate-img_eroode;

subplot(2,2,1),imshow(img),title('Orignal Image');
subplot(2,2,2),imshow(img_dilate),title('Dilation Image');
subplot(2,2,3),imshow(img_eroode),title('Erode Image');
subplot(2,2,4),imshow(g),title('Gradient Image');
```

**Output:**



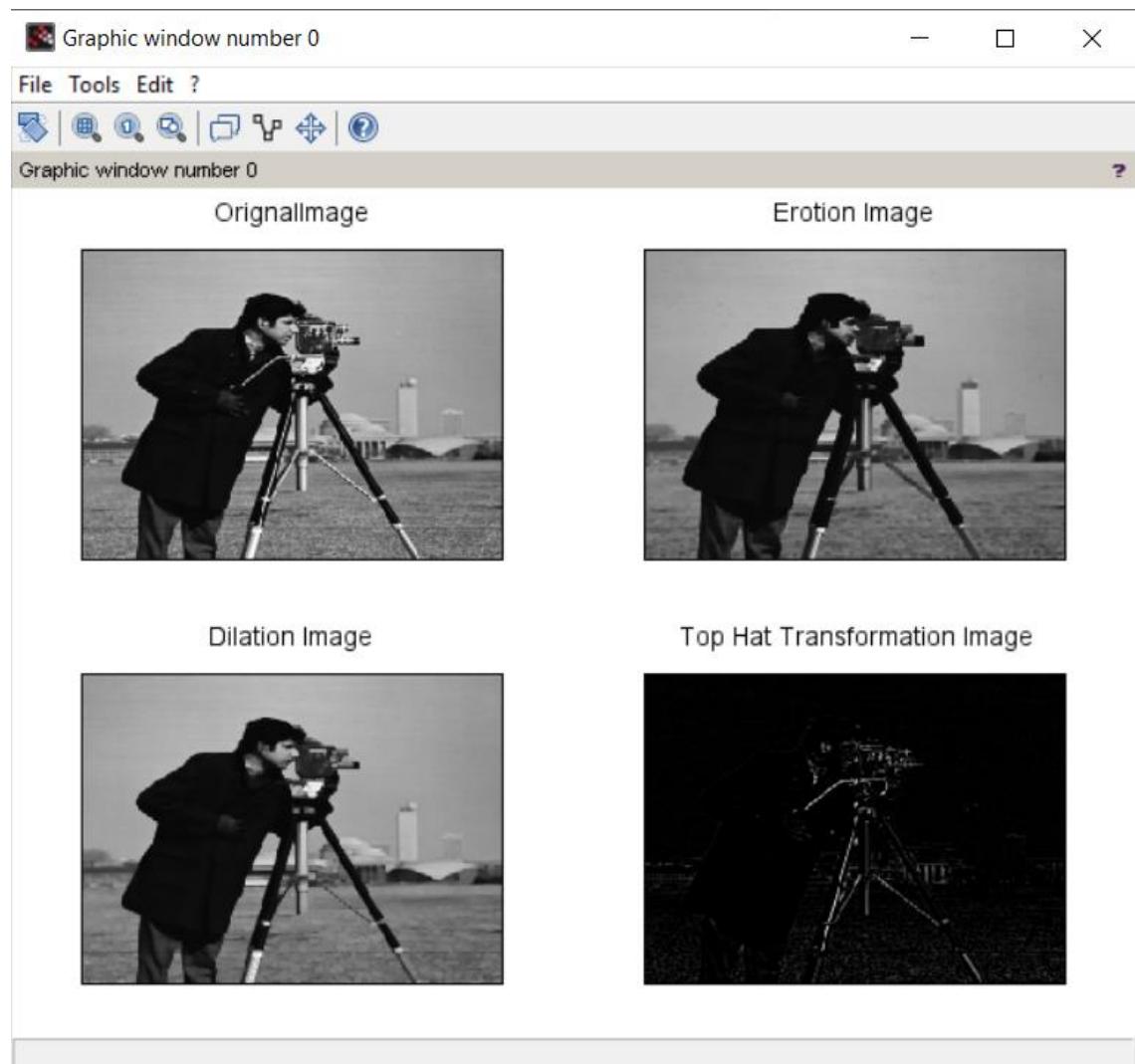
[D]: Program to apply Top-Hat/Bottom-hat Transformations.

**Code:**

```
clc;
clear;
img=imread('C:\Users\OneDrive\Documents\Rahul Kewat\IPCV\images\cameraman.png');
se1=imrect('rect',3,3);
im1=imerode(img,se1);
im2=imdilate(im1,se1);
h=img-im2;

subplot(2,2,1),imshow(img),title('OrignalImage');
subplot(2,2,2),imshow(im1),title('Eroton Image');
subplot(2,2,3),imshow(im2),title('Dilation Image');
subplot(2,2,4),imshow(h),title('Top Hat Transformation Image');
```

**Output:**



**Practical 8**

**Aim:** Image Segmentation

**Writeup:**

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

**[A]: Program for Edge detection using Sobel, Prewitt, Marr-Hildreth and Canny.**

**Code:**

```
clc;
clear all;
img = imread('C:\Users\Del\OneDrive\Documents\Rahul Kewat\IPCV\images\checkbox.png');
img = rgb2gray(img);

ed_Sobel = edge(img,'sobel');
ed_Prewitt = edge(img,'prewitt');
ed_Canny = edge(img,'canny');

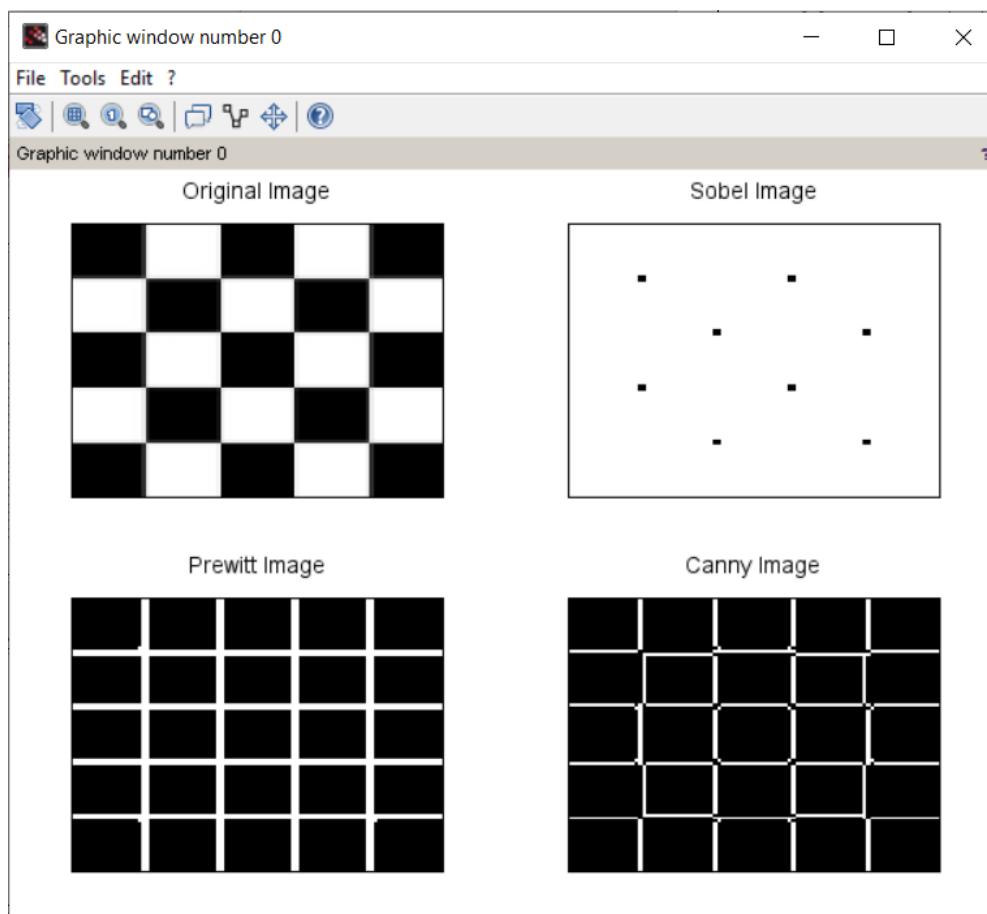
subplot(2,2,1);
imshow(img);
title('Original Image');

subplot(2,2,2);
imshow(ed_Sobel);
title('Sobel Image');

subplot(2,2,3);
imshow(ed_Prewitt);
title('Prewitt Image');

subplot(2,2,4);
imshow(ed_Canny);
title('Canny Image');
```

**Output:**



# PRESENTATION

## Image Negation and gray level slicing

Rahul Kewat

Msc-IT Part-1

Roll No-22006

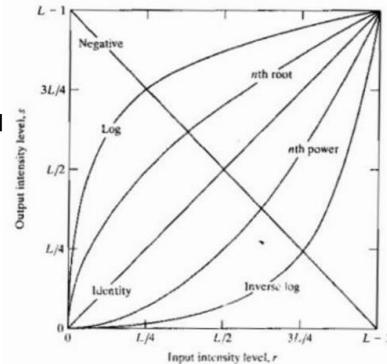
## Image Negation

- Intensity transformations are among the simplest of all image processing techniques.
- In this case  $g$  depends only on the value of  $f$  at  $(x, y)$  and  $T$  becomes an intensity or gray level transformation function of the form  
$$S=T(r)$$
  - r-Denotes the gray level of  $f(x, y)$
  - s-Denotes the gray level of  $g(x, y)$  at any point  $(x, y)$
- Because enhancement at any point in an image deepens only on the gray level at that point, technique in this category is referred to as point processing.

## Image Negation

### ➤ Basic Intensity Level Transformation

- 1) linear (negative and identity transformations),
- 2) logarithmic (log and inverse-log transformations), and
- 3) power-law ( $n$ th power and  $n$ th root transformations).



## Image Negation

### ➤ Identity Function

- Output intensities are identical to input intensities
- This function doesn't have an effect on an image, it was included in the graph only for completeness
- Its expression:

$$s = r$$

## Image Negation

➤ **Image Negation:**

The negative of an image with gray level in the range [0, L-1] is obtained by using the negative transformation.

The expression of the transformation is

$$s = L-1-r$$

- Reverting the intensity levels of an image in this manner produces the equivalent of a photographic negative.
- This type of processing is practically suited for enhancing white or gray details embedded in dark regions of an image especially when the black areas are dominant in size.

## Image Negation

➤ **Image Negatives (Negative Transformation)**

Image (r)



Image (s) after applying T (negative)

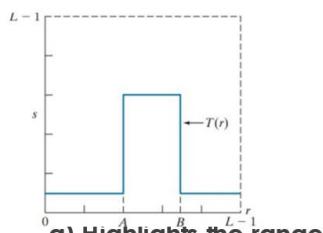
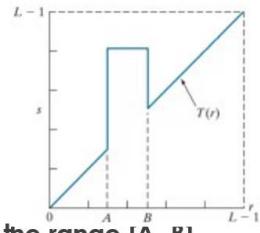


## Gray Level Slicing

- This technique is used to highlight a specific range of gray levels in a given image.
- Similar to thresholding
- Other levels can be suppressed or maintained
- Useful for highlighting features in an image
- There are two approach
- One approach is to display a high value for all gray levels in the range of interest and a low value for all other gray levels.
- The second approach, based on the transformation brightens the desired range of gray levels but preserves gray levels unchanged.

## Gray Level Slicing

Approach 1 Approach 2

a) Highlights the range  $[A, B]$ b) Highlights the range  $[A, B]$  and preserves background details

## Gray Level Slicing

