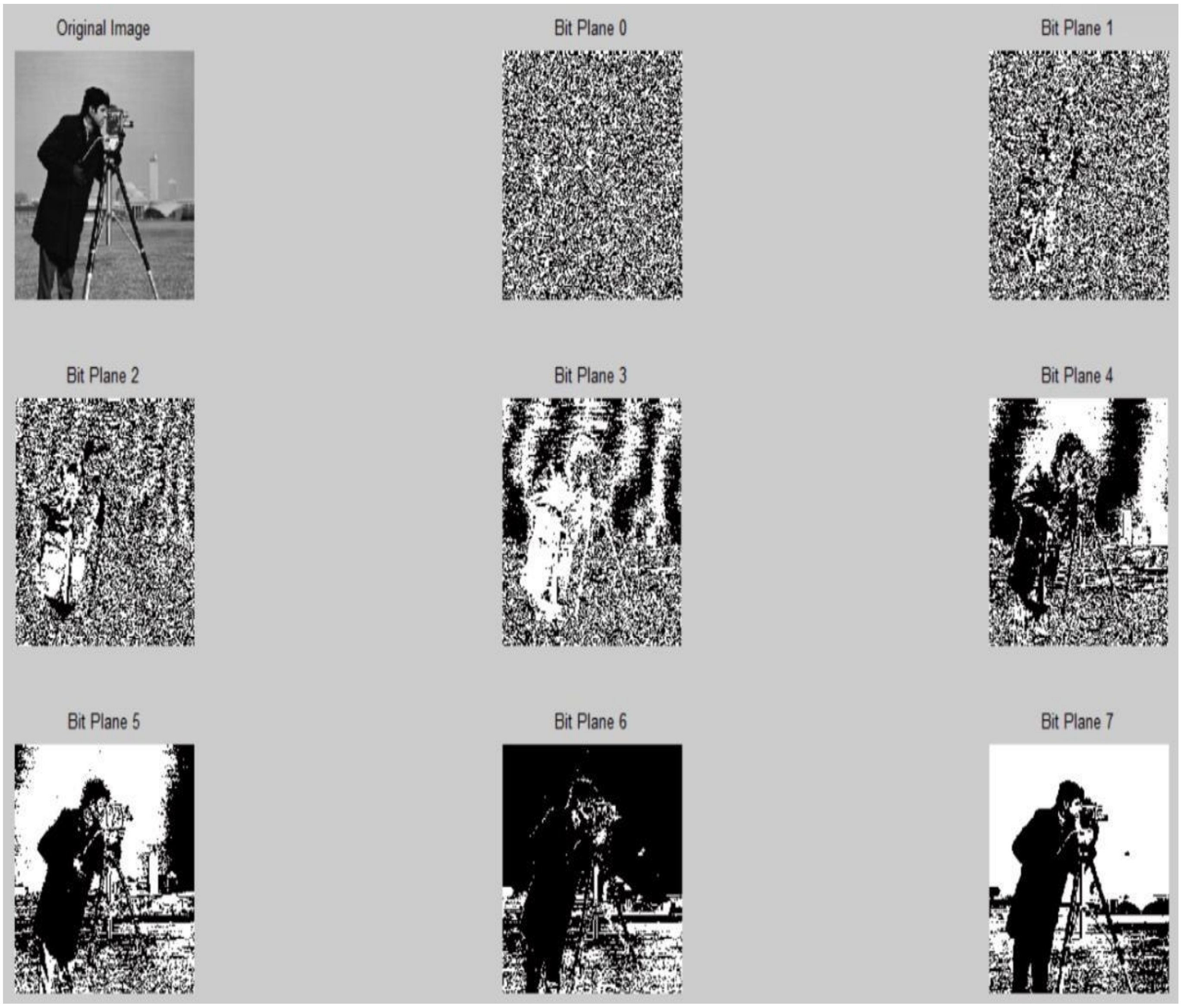
Exp4: Implement Bit Plane Slicing to read cameraman image and see the effect of each bit on image.

```
clc;
clear all;
close all;

A=imread('cameraman.tif');
[row,col]=size(A);
subplot(3,3,1), imshow(A), title('Original Image');
C=zeros(row,col,8);
for k=1:8
    for i=1:row
        for j=1:col
            C(i,j,k)=bitget(A(i,j),k);         %Bit slicing
        end
    end
    subplot(3,3,k+1), imshow(C(:,:,k)), title(['Bit Plane ',num2str(k-1)]);
end
```

OUTPUT:

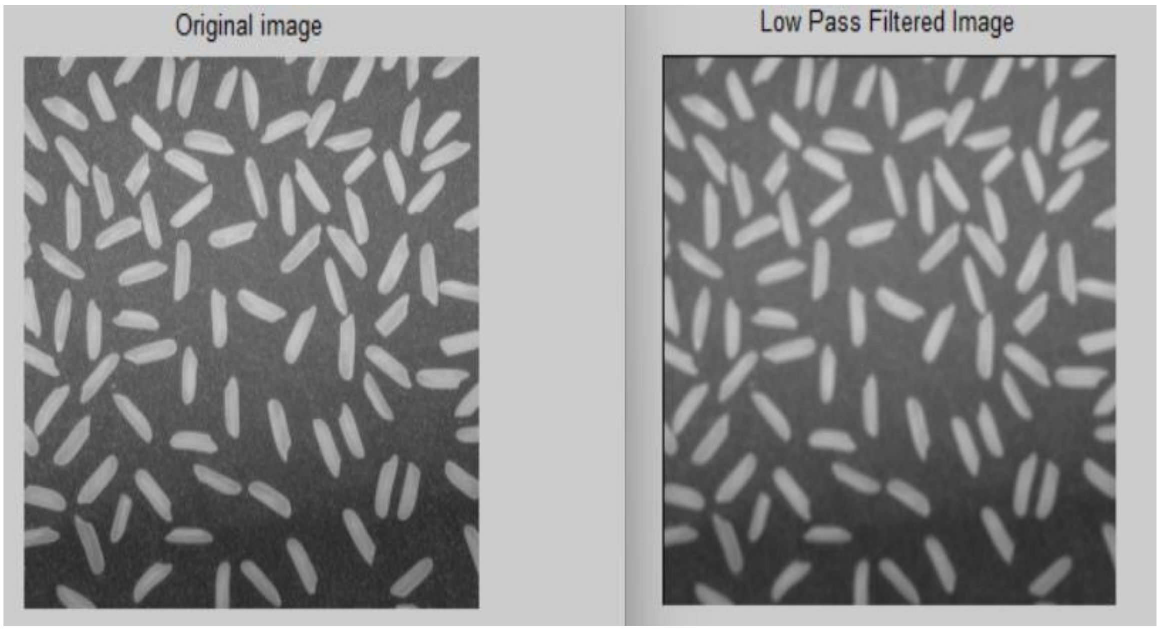


Exp5: Implement various Smoothing spatial filter.

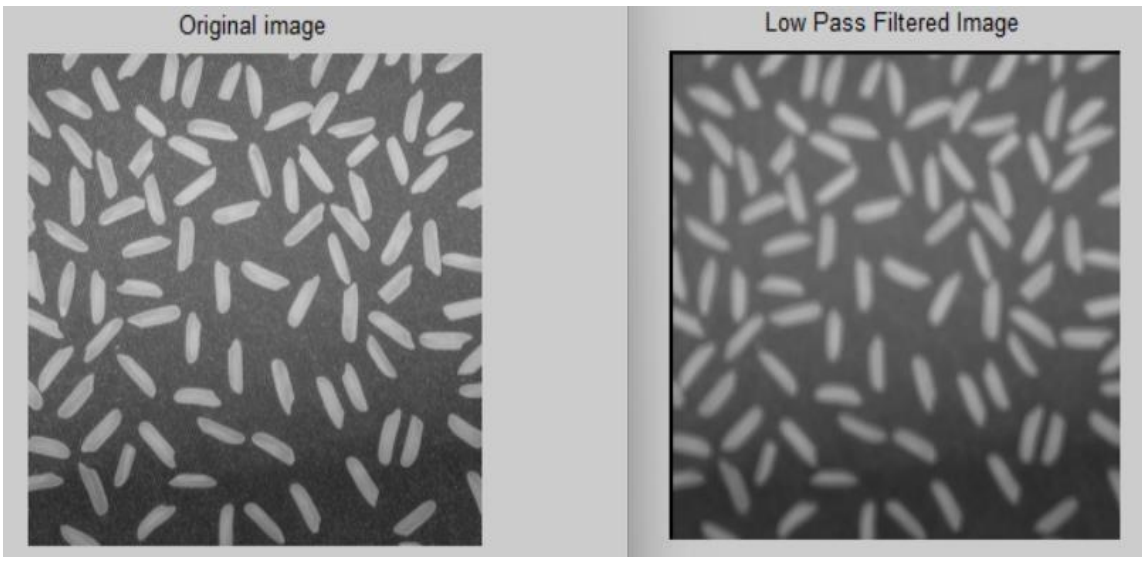
```
% Program for low pass filter %

clc;
close all;
clear all;
 % getting an image
a = imread('rice.png');
a = double(a);
imshow(uint8(a));
title('Original image');

% for filtering accepting the mask size from user
m = input('Enter the mask size(3 or 5) : ');
d = m*m;
mask = (1/d)*ones(m)

% according to the given input selecting the specified
% mask function
if m == 3
    result = mask3(a,mask);
else if m == 5
         result = mask5(a,mask);
     end
end

% displaying the resultant filtered image
figure
imshow(uint8(result));
title('Low Pass Filtered Image');
% function for low pass filtering using 3x3 mask


function out = mask3(a,mask)

[r c] = size(a);
p = zeros(r-1,c-1);
for s = 2:r-1
    for t = 2:c-1
        p(s,t) = (a(s-1,t-1)*mask(1)+a(s-1,t)*mask(2)+a(s-
1,t+1)*mask(3)+...
                  a(s,t-1)*mask(4)+a(s,t)*mask(5)+a(s,t+1)*mask(6)+...
                  a(s+1,t-1)*mask(7)+a(s+1,t)*mask(8)+a(s+1,t+1)*mask(9));
    end
end

out = p;


% function for low pass filtering using 5x5 mask

function out = mask5(a,mask)

[r c] = size(a);
p = zeros(r-2,c-2);
for s = 3:r-2
    for t = 3:c-2
        p(s,t) = (a(s-2,t-2)*mask(1) + a(s-2,t-1)*mask(2) + a(s-
2,t)*mask(3)+ a(s-2,t+1)*mask(4) + a(s-2,t+2)*mask(5)+...
a(s-1,t-2)*mask(6) + a(s-1,t-1)*mask(7) + a(s-1,t)*mask(8)+ a(s-
1,t+1)*mask(9) + a(s-1,t+2)*mask(10)+...
                  a(s, t-2)*mask(11) + a(s, t-1)*mask(12) +
a(s,t)*mask(13)+ a(s,t+1)*mask(14)  + a(s,t+2)*mask(15)+...
a(s+1,t-2)*mask(16)  + a(s+1,t-1)*mask(17)
+ a(s+1,t)*mask(18)  + a(s+1,t+1)*mask(19)+ a(s+1,t+1)*mask(20)+...
                  a(s+2,t-2)*mask(21)  + a(s+2,t-
1)*mask(22)+a(s+2,t)*mask(23)+ a(s+2,t+1)*mask(24) +a(s+2,t+1)*mask(25));
    end
end

out = p;
```

Output:

Enter the mask size(3 or 5) : 3

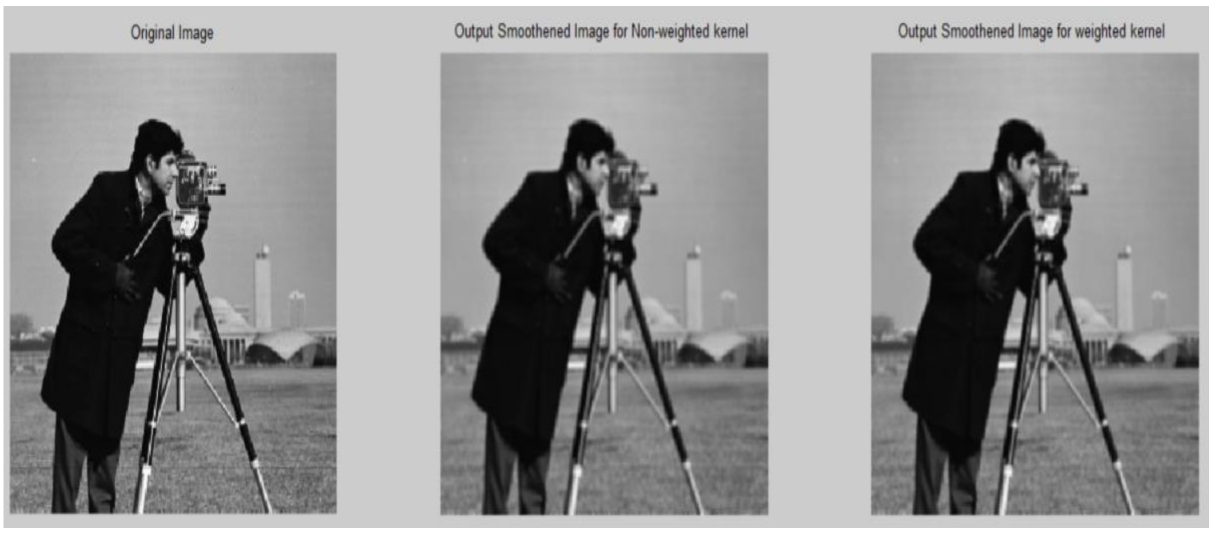


Enter the mask size(3 or 5) : 5



Exp5: Implement various Smoothing spatial filter. (With weighted filter and Non-weighted filter)

```
%%MATLAB CODE FOR SMOOTHING SPATIAL FILTER %%
clc;
clear;
close all;
F=double(imread('cameraman.tif'));
H=F;
I=F;
[rows cols]=size(F);
A=[1,1,1;1,1,1;1,1,1]; %Non-Weighted Kernel
B=[1,2,1;2,4,2;1,2,1]; %Weighted Kernel
G1=zeros(3,3);
G2=zeros(3,3);
for i = 2:rows-1
for j = 2:cols-1
for k=-1:1
for l=-1:1
G1(k+2,l+2)=F(i+k,j+l).*A(k+2,l+2);
G2(k+2,l+2)=F(i+k,j+l).*B(k+2,l+2);
end
end
sumG1=sum(sum(G1));
H(i,j)=sumG1./sum(sum(A));
sumG2=sum(sum(G2));
I(i,j)=sumG2./sum(sum(B));
end
end
H = uint8(round(H - 1));
I = uint8(round(I - 1));
figure()
subplot(131);imshow(uint8(F));
title('Original Image');
subplot(132);
imshow(H);
title('Output Smoothened Image for Non-weighted kernel ');
subplot(133);
imshow(I);
title('Output Smoothened Image for weighted kernel ');
```

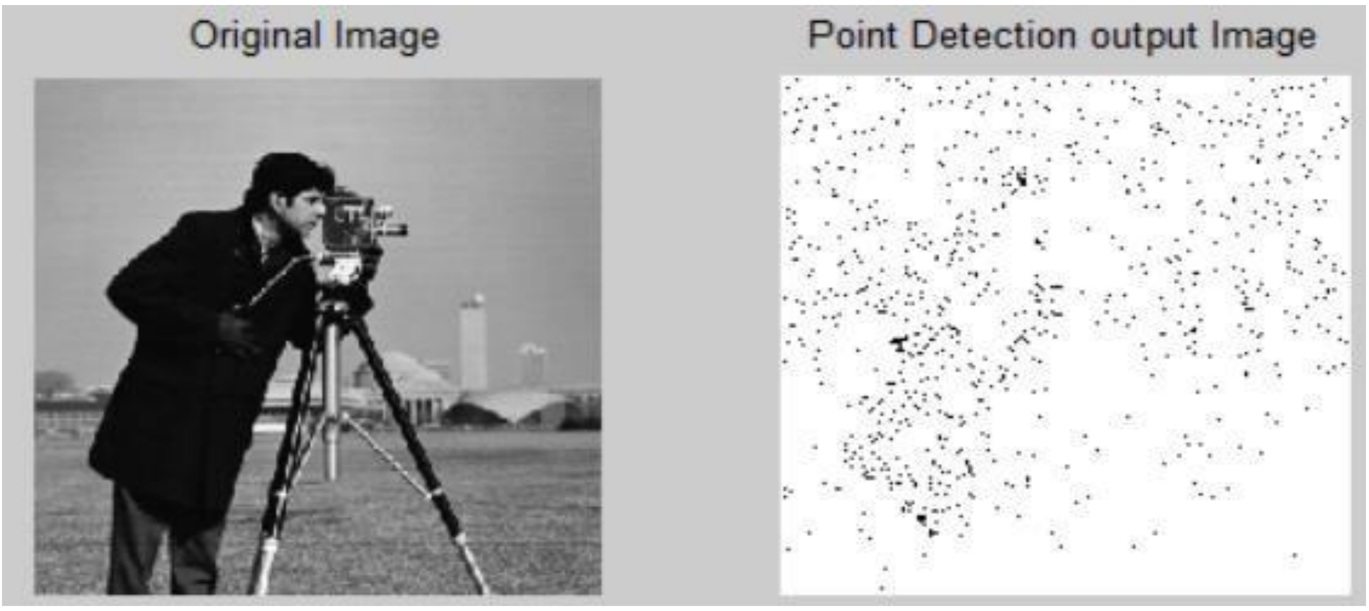

Original Image
Output Smoothened Image for Non-weighted kernel
Output Smoothened Image for weighted kernel

Exp6: Perform the Point detection, Line Detection and Edge Detection Operations on image.

1 Point Detection

```
%%MATLAB CODE FOR POINT DETECTION %%
clc;
close all;
clear all;
a=imread('cameraman.tif');
[m n]=size(a);
I=double(a);
In=ones(m,n);
%In=a;
mask1=[-1 -1 -1;-1 8 -1;-1 -1 -1];
for i=2:m-1
for j=2:n-1
neighbour_matrix1=mask1.*I(i-1:i+1, j-1:j+1);
avg_value1=sum(neighbour_matrix1(:));
%using max function for detection of final Points
In(i, j)=abs(max(avg_value1));
end
end
figure;
subplot(121);
imshow(a);
title('Original Image');
subplot(122);
imshow(In);
title('Point Detection output Image');

%Point detection uisng inbuilt function
a=imread('cameraman.tif');
figure,subplot(121);imshow(a);title('Original Image');
w=[-1,-1,-1;-1,8,-1;-1,-1,-1];
g=abs(imfilter(double(a),w));
subplot(122);imshow(g);title('Point detection output using inbuilt
function');
```

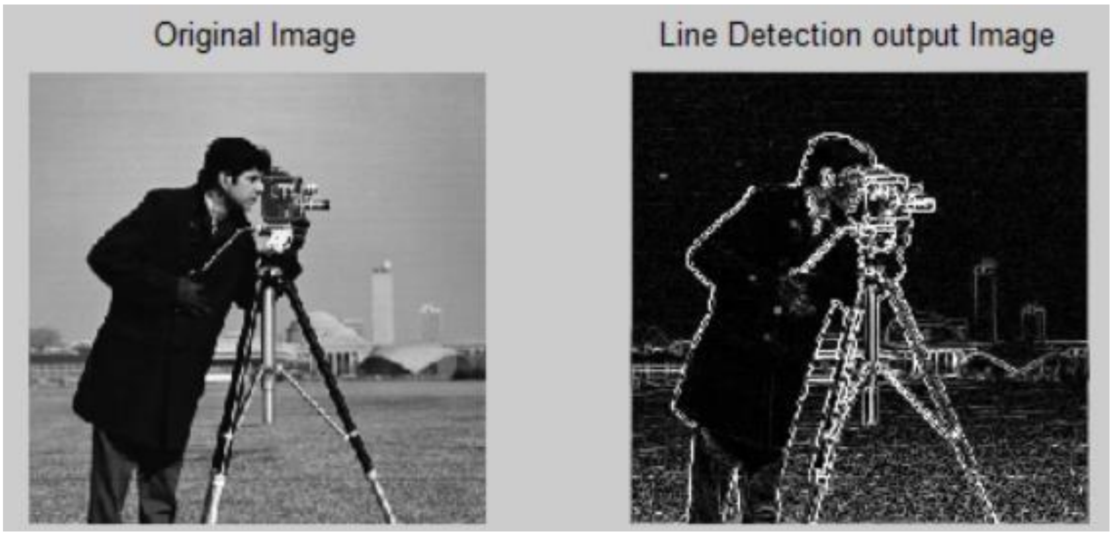
Output:

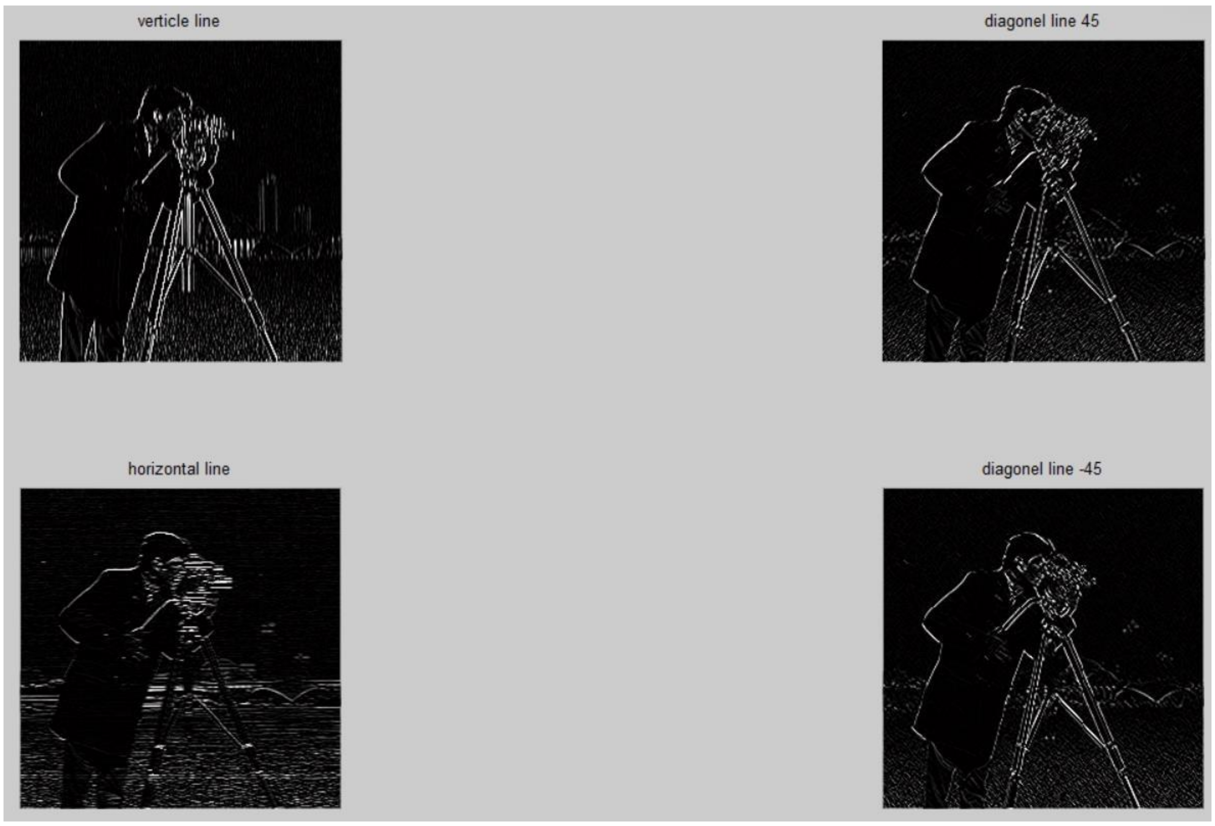


2 Line Detection

```
clc;
close all;
clear all;
a=imread('cameraman.tif');
I=double(a);
In=I;
I1=In;
I2=I;I3=I;I4=I;
mask1=[-1 2 -1;-1 2 -1;-1 2 -1];
mask2=[-1 -1 2;-1 2 -1;2 -1 -1];
mask3=[-1 -1 -1;2 2 2;-1 -1 -1];
mask4=[2 -1  -1;-1 2 -1;-1 -1 2];
for i=2:size(I, 1)-1
for j=2:size(I, 2)-1
neighbour_matrix1=mask1.*In(i-1:i+1, j-1:j+1);
avg_value1=sum(neighbour_matrix1(:));
neighbour_matrix2=mask2.*In(i-1:i+1, j-1:j+1);
avg_value2=sum(neighbour_matrix2(:));
neighbour_matrix3=mask3.*In(i-1:i+1, j-1:j+1);
avg_value3=sum(neighbour_matrix3(:));
neighbour_matrix4=mask4.*In(i-1:i+1, j-1:j+1);
avg_value4=sum(neighbour_matrix4(:));
%using max function for detection of final lines
I(i, j)=max([avg_value1, avg_value2, avg_value3, avg_value4]);
I1(i,j)=max(avg_value1);
I2(i,j)=max(avg_value2);
I3(i,j)=max(avg_value3);
I4(i,j)=max(avg_value4);
end
end
figure;
subplot(121);
imshow(a);
title('Original Image');
subplot(122);
imshow(uint8(I));
title('Line Detection output Image');
figure,subplot(221);imshow(uint8(I1));title('verticle line');
subplot(222);imshow(uint8(I2));title('diagonel line 45');
subplot(223);imshow(uint8(I3));title('horizontal line');
subplot(224);imshow(uint8(I4));title('diagonel line -45');
```

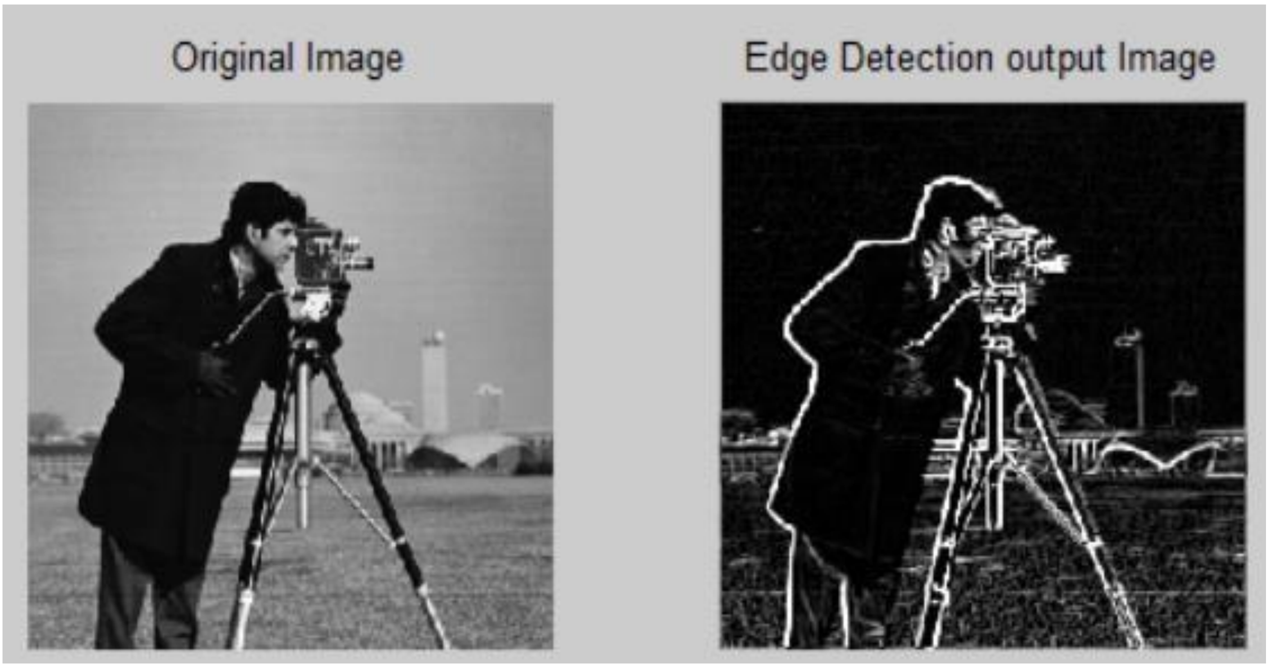
Output




verticle line
diagonel line 45
horizontal line
diagonel line -45

3 Edge Detection

```
clc;
close all;
clear all;
a=imread('cameraman.tif');
I=double(a);
In=I;
mask1=[1, 0, -1;1, 0, -1;1, 0, -1];
mask2=[1, 1, 1;0, 0, 0;-1, -1, -1];
mask3=[0, -1, -1;1, 0, -1;1, 1, 0];
mask4=[1, 1, 0;1, 0, -1;0, -1, -1];
for i=2:size(I, 1)-1
for j=2:size(I, 2)-1
neighbour_matrix1=mask1.*In(i-1:i+1, j-1:j+1);

avg_value1=sum(neighbour_matrix1(:));
neighbour_matrix2=mask2.*In(i-1:i+1, j-1:j+1);
avg_value2=sum(neighbour_matrix2(:));
neighbour_matrix3=mask3.*In(i-1:i+1, j-1:j+1);
avg_value3=sum(neighbour_matrix3(:));
neighbour_matrix4=mask4.*In(i-1:i+1, j-1:j+1);
avg_value4=sum(neighbour_matrix4(:));
%using max function for detection of final edges
I(i, j)=max([avg_value1, avg_value2, avg_value3, avg_value4]);
end
end
figure;
subplot(121);
imshow(a);
title('Original Image');
subplot(122);
imshow(uint8(I));
title('Edge Detection output Image');
```

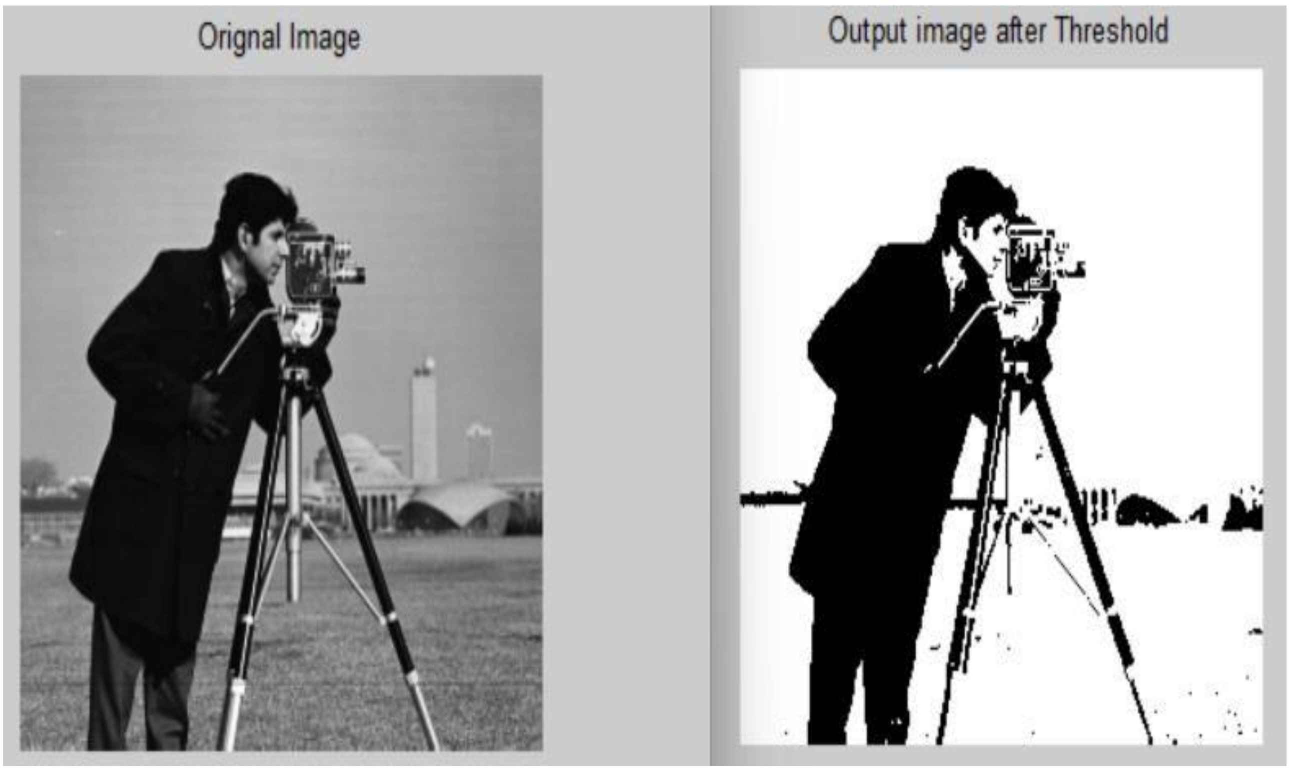


Output

Exp 7 Perform the Thresholding Operation on Image.

```
clc;
clear all;
close all;
a=imread('cameraman.tif');
figure(1);
imshow(a);
title('Orignal Image');
[row col]=size(a);
h=zeros(1,256);
for i=1:row
for j=1:col
t=a(i,j);
h(t)=h(t)+1;
end
end
[x,y]=ginput(1);
%%takes value of each pixel
%% Incrementing each counter
%%selction of threshold value from image

for i=1:col
for j=1:col
if a(i,j)<x;
a(i,j)=0;
else
a(i,j)=255;
end
end
end
figure(2);
imshow(uint8(a));
title('Output image after Threshold');
```

Output

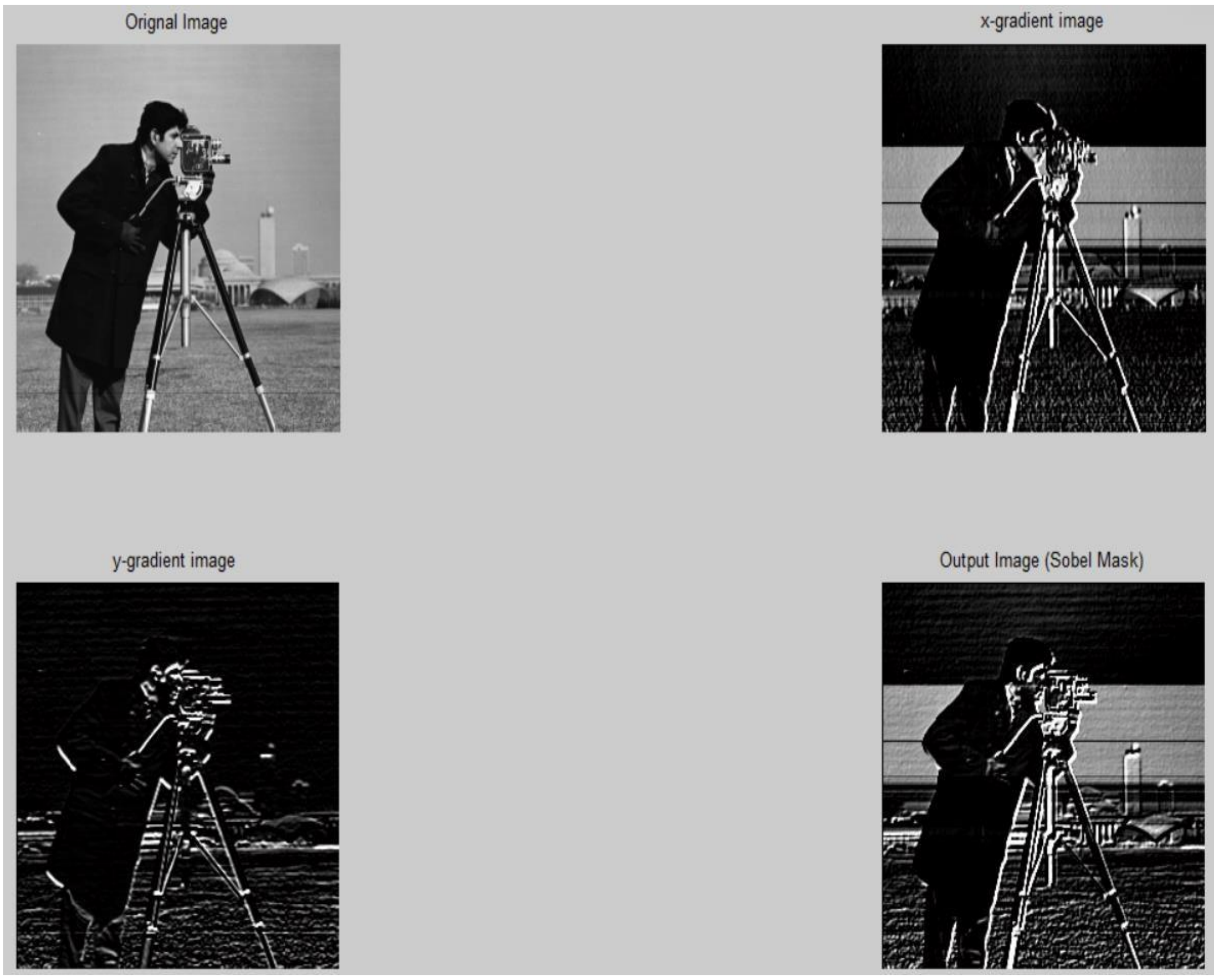


Exp8: Implement and study the effect of Different Mask (Sobel, Prewitt and Roberts)

Sobel Mask

```
%%%MATLAB CODE FOR SOBEL MASK (OPERATOR)%%
clc;
clear all;
close all;
aa=imread('cameraman.tif');
subplot(2,2,1);
imshow(aa);
title('Orignal Image');
a=double(aa);
[row col]=size(a);
w1=[-1 -2 -1; 0 0 0; 1 2 1 ];
w2=[-1 0 1; -2 0 2; -1 0 1];
for x=2:1:row-1;
for y=2:1:col-1;
a1(x,y)=w1(1)*a(x-1,y-1)+w1(2)*a(x-1,y)+w1(3)*a(x-1,y+1)+w1(4)*a(x,y-
1)+w1(5)*a(x,y)+w1(6)*a(x,y+1)+w1(7)*a(x+1,x-
1)+w1(8)*a(x+1,y)+w1(9)*a(x+1,y+1);

a2(x,y)=w2(1)*a(x-1,y-1)+w2(2)*a(x-1,y)+w2(3)*a(x-1,y+1)+w2(4)*a(x,y-
1)+w2(5)*a(x,y)+w2(6)*a(x,y+1)+w2(7)*a(x+1,y-
1)+w2(8)*a(x+1,y)+w2(9)*a(x+1,y+1);
end
end
a3=a1+a2;
subplot(2,2,2);
%%final gradient Value
imshow(uint8(a1));%%the x-gradient image
title('x-gradient image');
subplot(2,2,3);
imshow(uint8(a2)) %%the y-gradient image
title('y-gradient image ');
subplot(2,2,4);
imshow(uint8(a3)) %%final image%%
title('Output Image (Sobel Mask)');
```

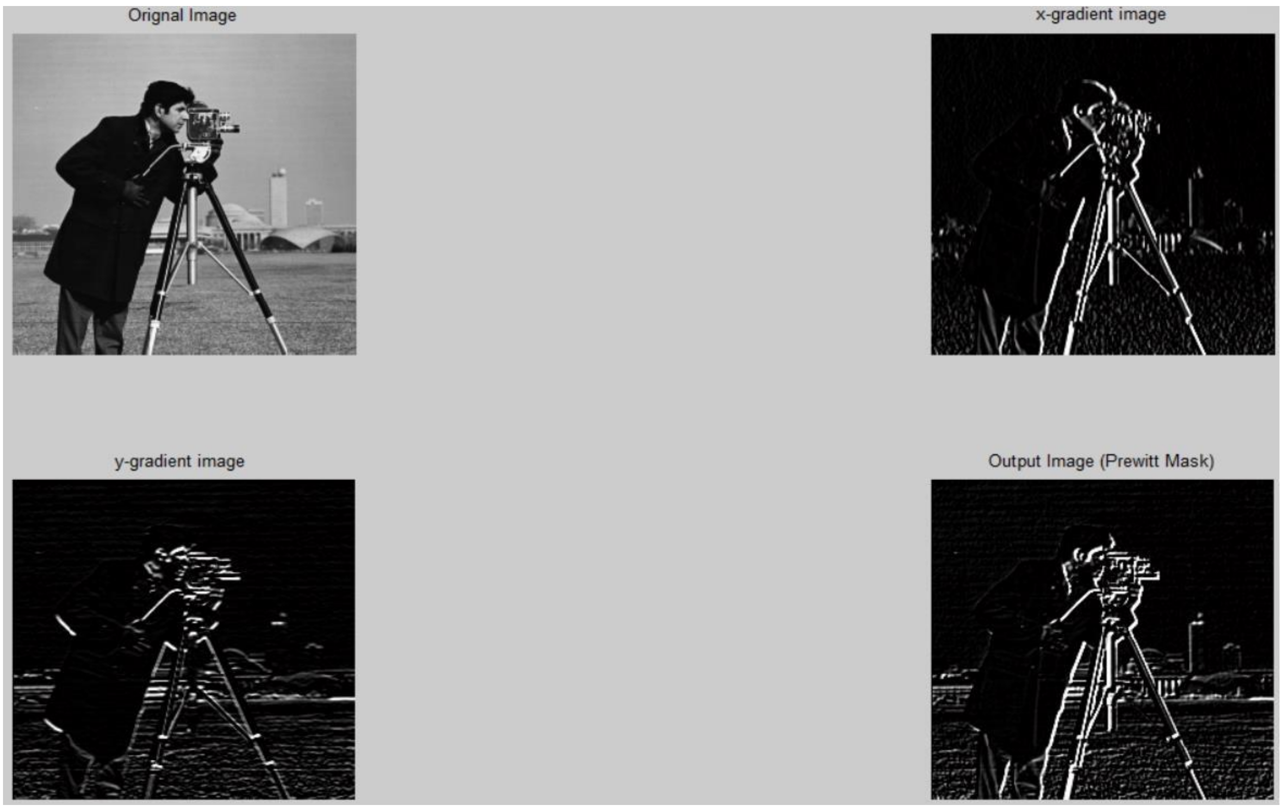
Output


Orignal Image
x-gradient image
y-gradient image
Output Image (Sobel Mask)

Prewitt Mask

```
%%%MATLAB CODE FOR PREWITT MASK (OPERATOR)%%
clc;
clear all;
close all;
aa=imread('cameraman.tif');
subplot(2,2,1);
imshow(aa);
title('Orignal Image');
a=double(aa);
[row col]=size(a);
w2=[-1 0 1; -1 0 1; -1 0 1];
w1=[-1 -1 -1; 0 0 0; 1 1 1];
for x=2:1:row-1;

for y=2:1:col-1;
a1(x,y)=w1(1)*a(x-1,y-1)+w1(2)*a(x-1,y)+w1(3)*a(x-1,y+1)+w1(4)*a(x,y-
1)+w1(5)*a(x,y)+w1(6)*a(x,y+1)+w1(7)*a(x+1,y-
1)+w1(8)*a(x+1,y)+w1(9)*a(x+1,y+1);
a2(x,y)=w2(1)*a(x-1,y-1)+w2(2)*a(x-1,y)+w2(3)*a(x-1,y+1)+w2(4)*a(x,y-
1)+w2(5)*a(x,y)+w2(6)*a(x,y+1)+w2(7)*a(x+1,y-
1)+w2(8)*a(x+1,y)+w2(9)*a(x+1,y+1);
end
end
a3=a1+a2;
subplot(2,2,2);
%%final gradient Value
imshow(uint8(a1));%%the x-gradient image
title('x-gradient image');
subplot(2,2,3);
imshow(uint8(a2)) %%the y-gradient image
title('y-gradient image ');
subplot(2,2,4);
imshow(uint8(a3)) %%final image%%
title('Output Image (Prewitt Mask)');
```

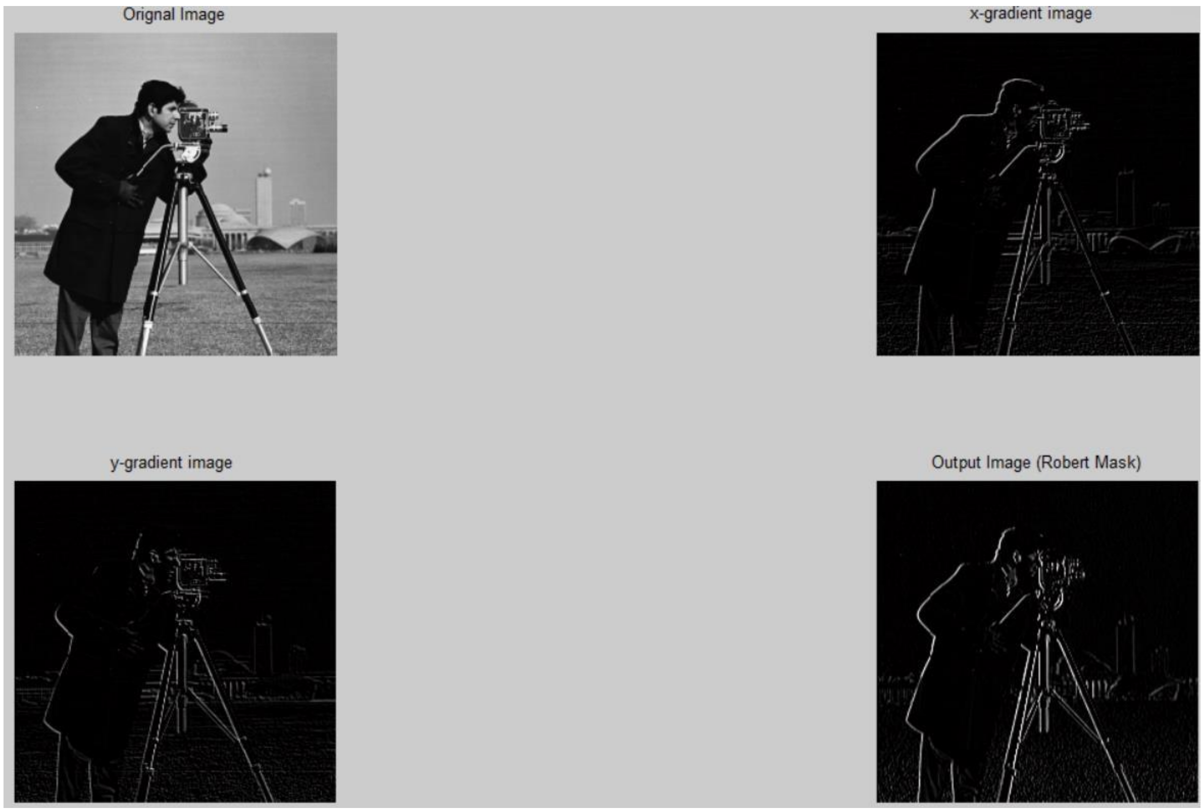
Output



Robert Mask

```
clc;
clear all;
close all;
aa=imread('cameraman.tif');
subplot(2,2,1);
imshow(aa);
title('Orignal Image');
a=double(aa);
[row col]=size(a);


w1=[1 0;0 -1];
w2=[0 1; -1 0];
for x=2:row-1;
for y=2:1:col-1;
a1(x,y)=w1(1)*a(x,y)+w1(2)*a(x,y+1)+w1(3)*a(x+1,y)+w1(4)*a(x+1,y+1);
a2(x,y)=w2(1)*a(x,y)+w2(2)*a(x,y+1)+w2(3)*a(x+1,y)+w2(4)*a(x+1,y+1);
end
end
a3=a1+a2;
subplot(2,2,2);
%%final gradient value
imshow(uint8(a1));%%the x-gradient image
title('x-gradient image ');
subplot(2,2,3);
imshow(uint8(a2)) %%the y-gradient image
title('y-gradient image ');
subplot(2,2,4);
imshow(uint8(a3)) %%final image
title('Output Image (Robert Mask)');
```

Output

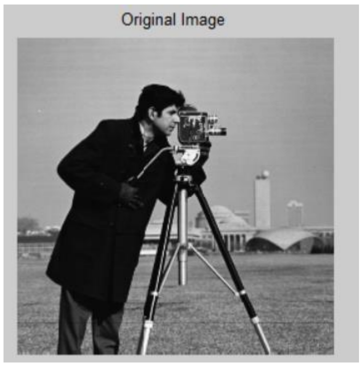


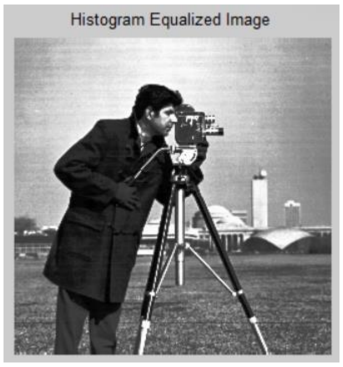
Exp 9 Title: Read an image, plot its histogram then do histogram equalization.

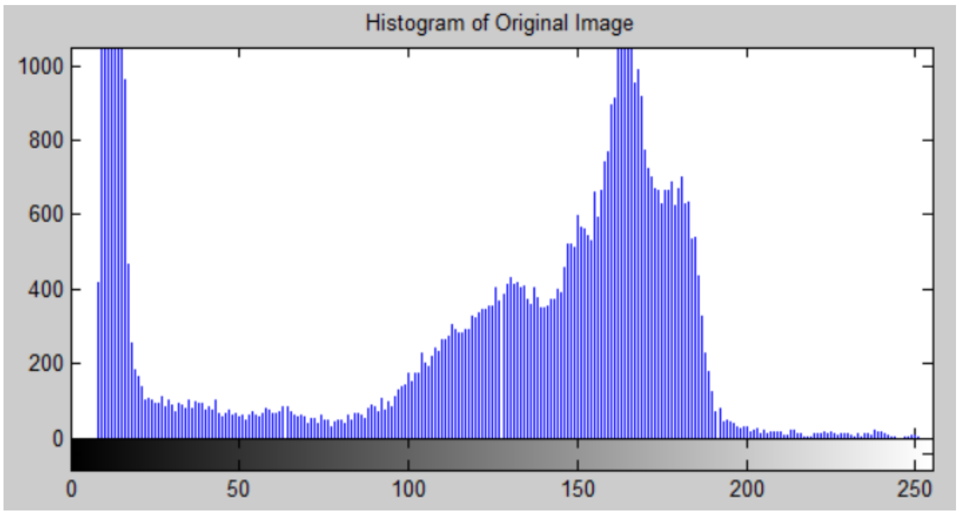
Code:

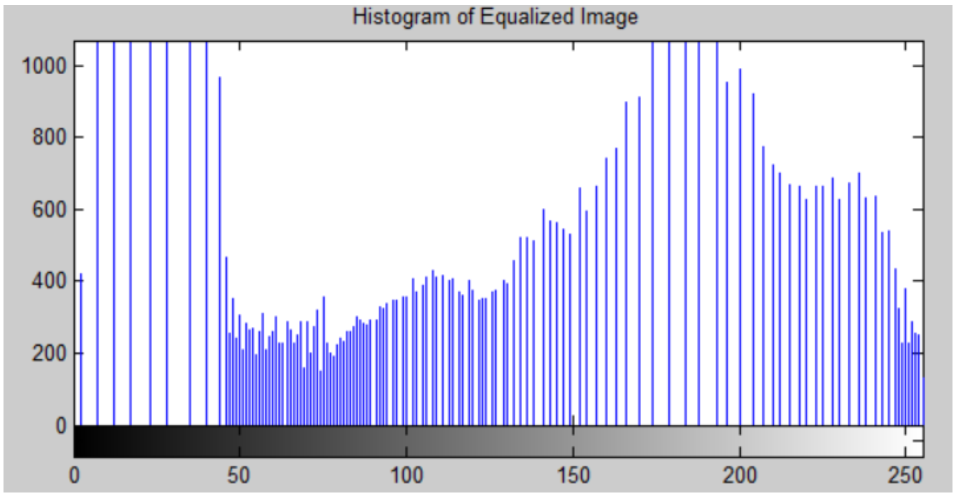
```
close all;
clear all;
x=imread('cameraman.tif');
[r c]=size(x);
m=max(max(x));
h=zeros(1,m);
z=0;
for i=1:r
for  j=1:c
if x(i,j)==0
z=z+1;
else
h(1,x(i,j))=h(1,x(i,j))+1;
end
end
end
out=[z,h];
% Equalization of histogram
h_in=out;
[l_r l_c]=size(h_in);
maxquanta=2.^(ceil(log2(l_c)))-1;
tp=r*c;
pdf=h_in/tp;
cdf=zeros(l_r,l_c);
cdf(1)=pdf(1);
for i=2:l_c
cdf(i)=cdf(i-1)+pdf(i);
end
eqtable=round(cdf*maxquanta);
out2=zeros(r,c);
temp=0;
for i=1:r
for j=1:c
out2(i,j)=eqtable(1,x(i,j)+1);
end
end
out2=uint8(out2);
figure;
subplot(221)
imshow(x);
title('Original Image');
subplot(222)
imshow(out2);
title('Histogram Equalized Image');
subplot(223)
imhist(x);
title('Histogram of Original Image');
subplot(224)
imhist(out2);
title('Histogram of Equalized Image');
```

OUTPUT:


Original Image


Histogram Equalized Image


Histogram of Original Image
1000
800
600
400
200
0
0
50
100
150
200
250

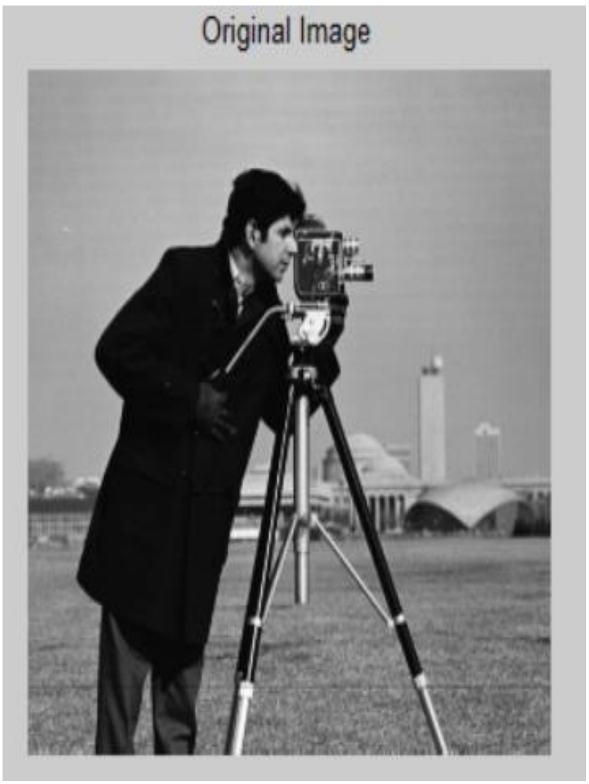

Histogram of Equalized Image
1000
800
600
400
200
0
0
50
100
150
200
250

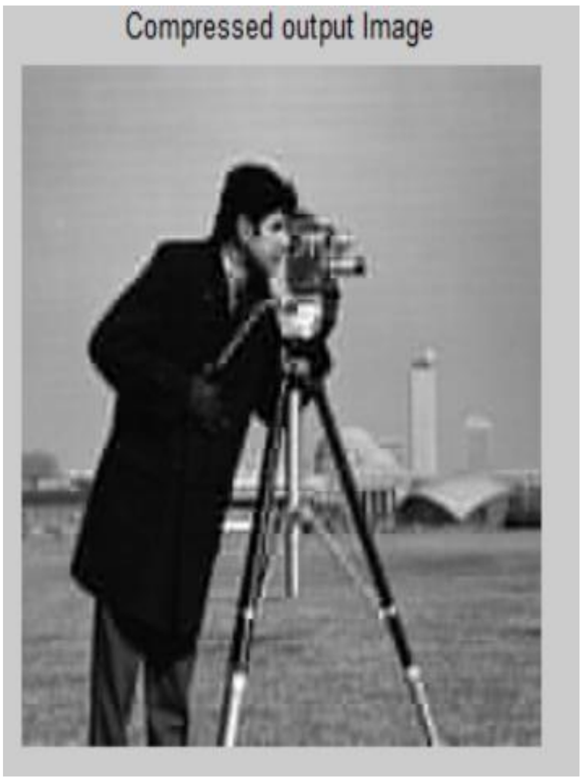
Exp10:Title: Implement Image compression using DCT Transform.

Code:

```
clc;
close all;
clear all;
I=imread('cameraman.tif');
I = im2double(I);
T = dctmtx(8);
disp(T);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask=[1    1    1    1    0    0    0    0
      1    1    1    0    0    0    0    0
      1    1    0    0    0    0    0    0
      1    0    0    0    0    0    0    0
      0    0    0    0    0    0    0    0
      0    0    0    0    0    0    0    0
      0    0    0    0    0    0    0    0
      0    0    0    0    0    0    0    0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
figure;
imshow(I);
title('Original Image');

figure;
imshow(I2);
title('Compressed output Image');
```

Output:

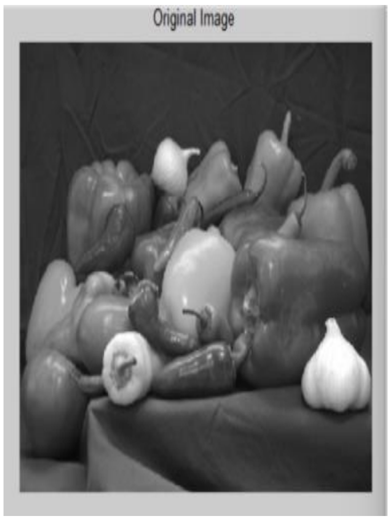


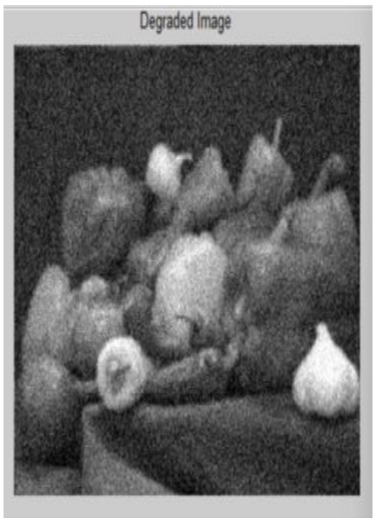


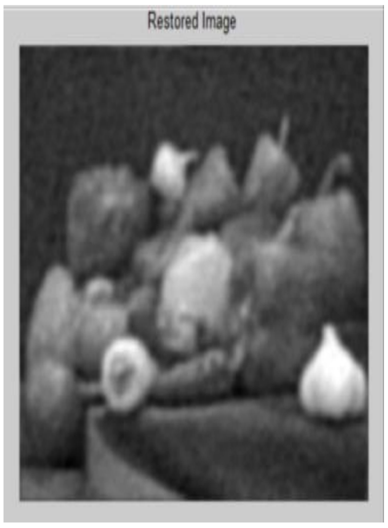
Exp11 **Title: Implement wiener filter over image and comment on it.**

```
clc;
close all;
clear all;
x = imread('peppers.png');
x=double(rgb2gray(x));
sigma = 50;
gamma = 1;
alpha = 1;
% It indicates Wiener filter
[M N]=size(x);
h = ones(5,5)/25;
Freqa = fft2(x);
Freqh = fft2(h,M,N);
y = real(ifft2(Freqh.*Freqa))+25*randn(M,N);
Freqy = fft2(y);
Powy = abs(Freqy).^2/(M*N);
sFreqh = Freqh.*(abs(Freqh)>0)+1/gamma*(abs(Freqh)==0);
iFreqh = 1./sFreqh;
iFreqh = iFreqh.*(abs(Freqh)*gamma>1)...
+gamma*abs(sFreqh).*iFreqh.*(abs(sFreqh)*gamma<=1);
Powy = Powy.*(Powy>sigma^2)+sigma^2*(Powy<=sigma^2);
Freqg = iFreqh.*(Powy-sigma^2)./(Powy-(1-alpha)*sigma^2);
ResFreqa = Freqg.*Freqy;
Resa = real(ifft2(ResFreqa));
imshow(uint8(x)),title('Original Image')
figure,imshow(uint8(y)),title('Degraded Image')
figure,imshow(uint8(Resa)),title('Restored Image')
```

Output:



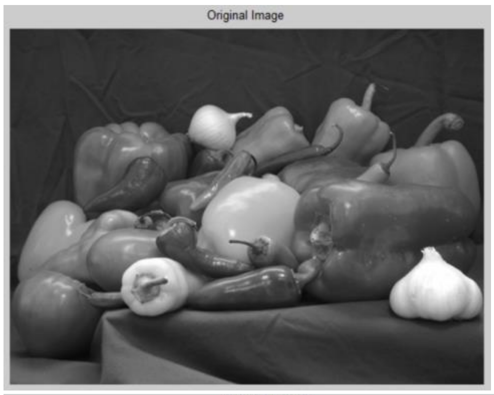


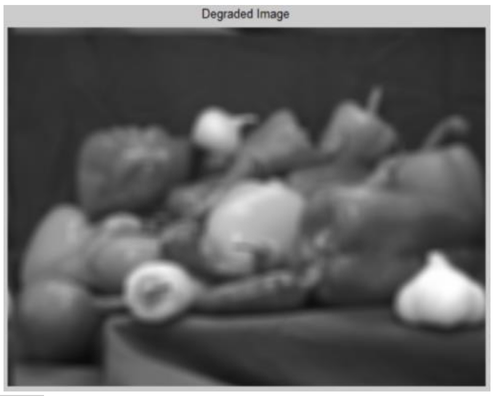


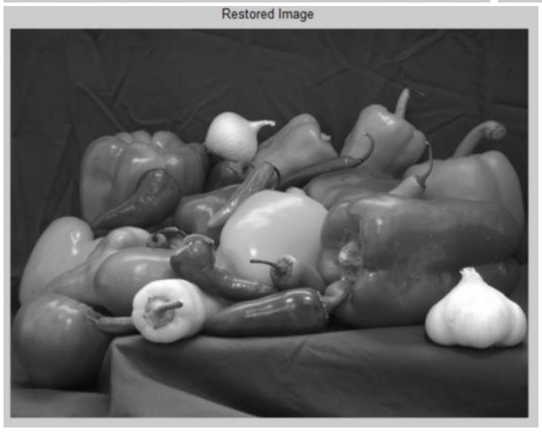
Title: Implement inverse filter over image and comment on it

```
clc;
close all;
clear all;
x =imread('peppers.png');
x=double(rgb2gray(x));
[M N]=size(x);
h = ones(11,11)/121;
sigma = sqrt(4*10^(-7));
freqx = fft2(x);% Fourier transform of input image
freqh = fft2(h,M,N);%Fourier transform of degradation
y = real(ifft2(freqh.*freqx));
freqy = fft2(y);
powfreqx = freqx.^2/(M*N);
alpha = 0.5;%Indicates inverse filter
freqg = ((freqh.')').*abs(powfreqx) ...
./(abs(freqh.^2).*abs(powfreqx)+alpha*sigma^2);
Resfreqx = freqg.*freqy;
Resa = real(ifft2(Resfreqx));
imshow(uint8(x)),title('Original Image')
figure,imshow(uint8(y)),title('Degraded Image')
figure,imshow(uint8(Resa)),title('Restored Image')
```

OUTPUT:


Original Image


Degraded Image


Restored Image