|  |  |  |
| --- | --- | --- |
| Sr. No. | Practicals | Signature |
| 01 | Image sampling and quantization |  |
| 02 | Analysis of spatial and intensity resolution of images |  |
| 03 | Analysis of spatial and intensity resolution of images |  |
| 04 | DFT analysis of images |  |
| 05a | Walsh Transform |  |
| 05b | Hadamard Transform |  |
| 05c | DCT Transform |  |
| 05d | Haar Transform |  |
| 06 | Haar Transform |  |
| 07 | Image Enhancement-Spatial filtering |  |
| 08 | Image Enhancement- Filtering in frequency domain |  |

# Practical 01

## Aim: Image sampling and quantization

**Code:**

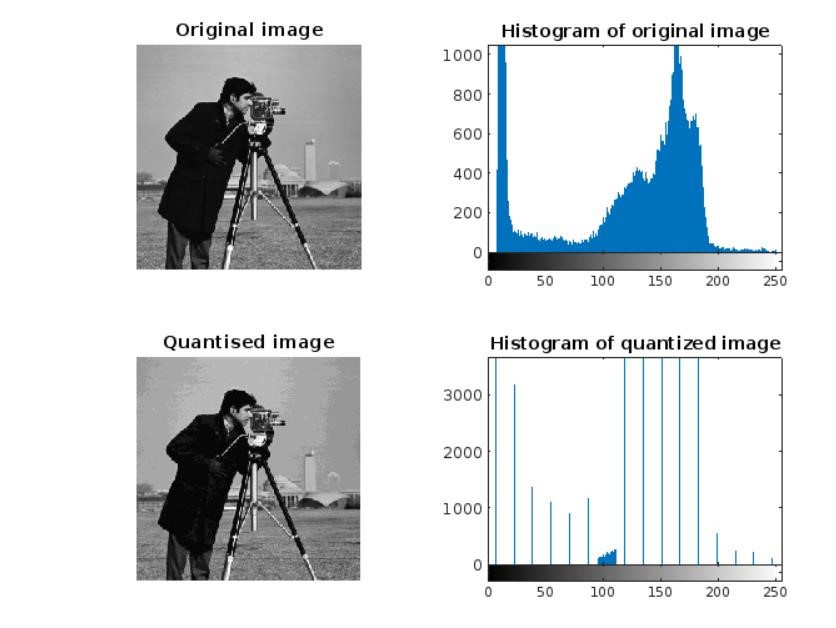
a=imread('cameraman.tif'); subplot(2,2,1) imshow(a); title('Original image'); subplot(2,2,2); imhist(a);

title('Histogram of original image');

[m n]=size(a); for i=1:1:m for j=1:1:n if a(i,j)<16 a(i,j)=7; elseif a(i,j)>=16 && a(i,j)<32 a(i,j)=23; elseif a(i,j)>=32 && a(i,j)<48 a(i,j)=39; elseif a(i,j)>=48 && a(i,j)<64 a(i,j)=55; elseif a(i,j)>=64 && a(i,j)<80 a(i,j)=71; elseif a(i,j)>=80 && a(i,j)<96 a(i,j)=87; elseif a(i,j)>=96 && a(i,j)<96 a(i,j)=103; elseif a(i,j)>=112 && a(i,j)<128 a(i,j)=119; elseif a(i,j)>=128 && a(i,j)<144 a(i,j)=135; elseif a(i,j)>=144 && a(i,j)<160 a(i,j)=151; elseif a(i,j)>=160 && a(i,j)<176 a(i,j)=167; elseif a(i,j)>=176 && a(i,j)<192 a(i,j)=183; elseif a(i,j)>=192 && a(i,j)<208 a(i,j)=199; elseif a(i,j)>=208 && a(i,j)<224 a(i,j)=215; elseif a(i,j)>=224 && a(i,j)<240 a(i,j)=231; elseif a(i,j)>=240 && a(i,j)<256 a(i,j)=247;

end end end subplot(2,2,3) imshow(a); title('Quantised image') subplot(2,2,4) imhist(a);

title('Histogram of quantized image')

**output :**

# Practical 02

## Aim: Analysis of special and intensity of resolution

**Spacial resolution:**

**Code:**

z=imread('cameraman.tif'); z=imresize(z,[1024,1024]);

[r c]=size(z); l=1; for i=1:2:r k=1; for j=1:2:c a(l,k)=z(i,j); k=k+1;

end

l=l+1; end

l=1; for i=1:4:r k=1; for j=1:4:c

b(l,k)=z(i,j); k=k+1; end

l=l+1; end

l=1; for i=1:8:r k=1; for j=1:8:c e(l,k)=z(i,j); k=k+1;

end

l=l+1; end l=1; for i=1:16:r k=1; for j=1:16:c d(l,k)=z(i,j); k=k+1; end

l=l+1; end subplot(2,2,1),imshow(a) subplot(2,2,2),imshow(b) subplot(2,2,3),imshow(e) subplot(2,2,4),imshow(d)

**output :**



**Intensity resolution Code :**

% Reading the image and converting it to a gray-level image.

I=imread('saturn.png');

I=rgb2gray(I);

% A 256 gray-level image: [I256,map256]=gray2ind(I,256);

subplot(2,2,1); imshow(I256,map256);

% A 128 gray-level image: [I128,map128]=gray2ind(I,128);

subplot(2,2,2); imshow(I128,map128);

% A 64 gray-level image: [I64,map64]=gray2ind(I,64)

subplot(2,2,3); imshow(I64,map64);

% A 32 gray-level image: [I32,map32]=gray2ind(I,32);

subplot(2,2,4); imshow(I32,map32);

% A 16 gray-level image:

[I16,map16]=gray2ind(I,16);

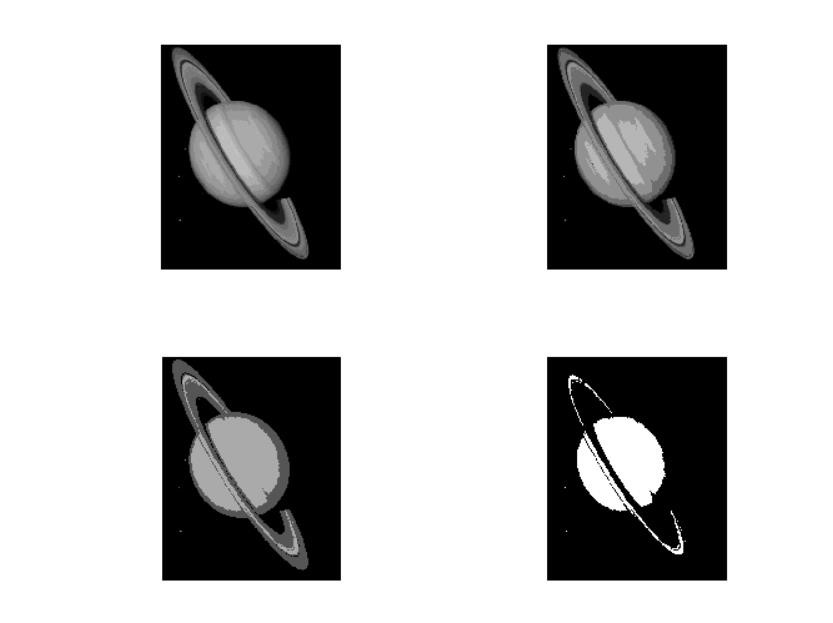
figure, subplot(2,2,1); imshow(I16,map16);

% A 8 gray-level image: [I8,map8]=gray2ind(I,8); subplot(2,2,2); imshow(I8,map8);

% A 4 gray-level image: [I4,map4]=gray2ind(I,4); subplot(2,2,3); imshow(I4,map4);

% A 2 gray-level image: [I2,map2]=gray2ind(I,2); subplot(2,2,4); imshow(I2,map2);

**Output :**



# Practical 03

## Aim: Information transformation of images

1. **photographic negative Code:**

I=imread('cameraman.tif'); imshow(I) J=imcomplement(I);

figure, imshow(J)

**Output:**



1. **Gamma transformation Code:**

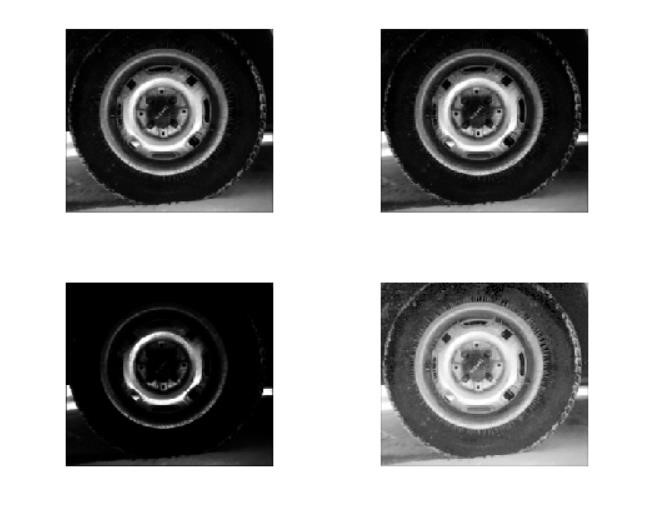
I=imread('tire.tif'); subplot(2,2,1);

imshow(I)

J=imadjust(I,[],[],1);

J2=imadjust(I,[],[],3); J3=imadjust(I,[],[],0.4); subplot(2,2,2); imshow(J); subplot(2,2,3); imshow(J2); subplot(2,2,4); imshow(J3);

**output :**



1. **Logarithmic transformation**

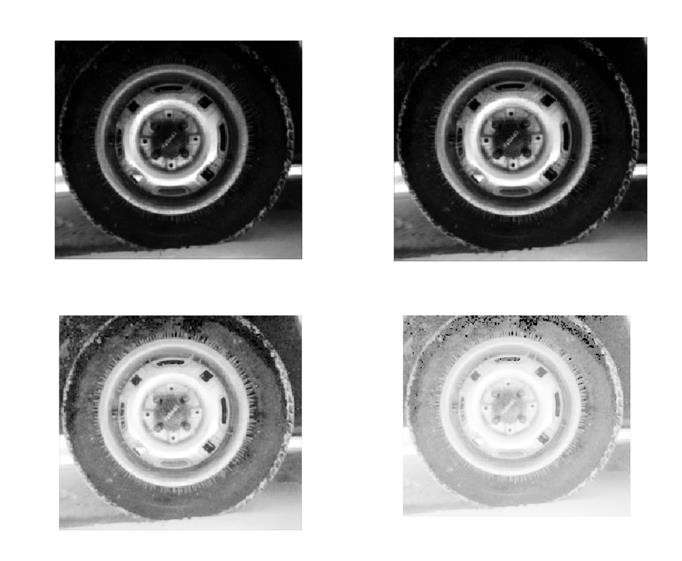
**Code:**

tire = imread('tire.tif'); d = im2double(tire); figure, imshow(d); %log on domain [0,1]

f = d; c = 1/log(1+1); j1 = c\*log(1+f); figure, imshow(j1); %log on domain [0, 255]

f = d\*255; c = 1/log(1+255); j2 = c\*log(1+f); figure, imshow(j2); %log on domain [0, 2^16]

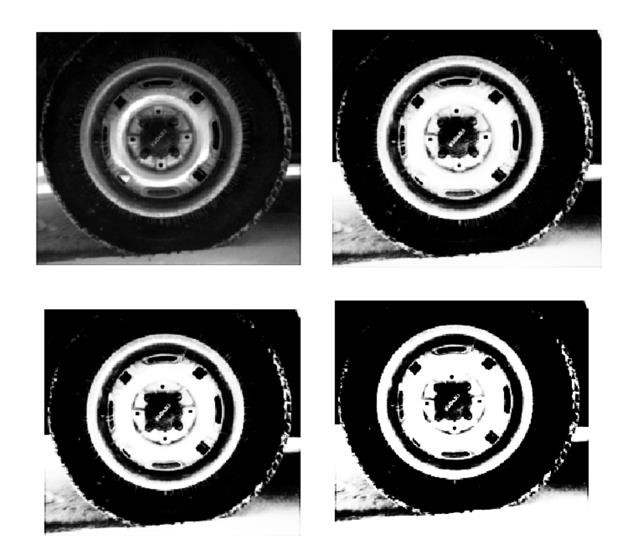
f = d\*2^16; c = 1/log(1+2^16); j3 = c\*log(1+f); figure, imshow(j3);



1. **Contrast stretching with changing E Code:**

I=imread('tire.tif'); I2=im2double(I); m=mean2(I2)

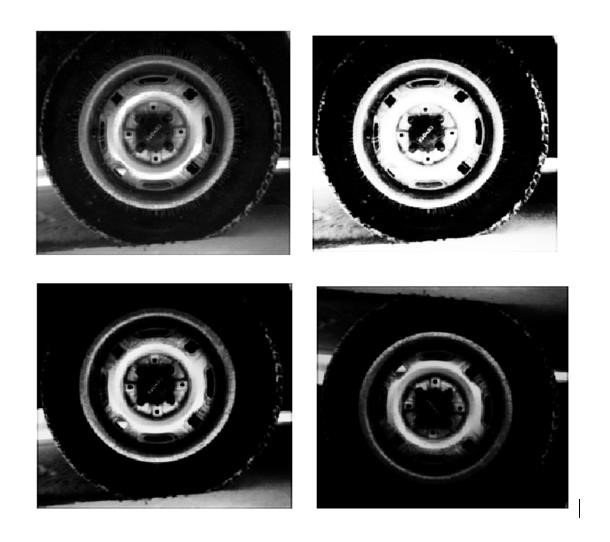
contrast1=1./(1+(m./(I2+eps)).^4); contrast2=1./(1+(m./(I2+eps)).^5); contrast3=1./(1+(m./(I2+eps)).^10); imshow(I2) figure,imshow(contrast1) figure,imshow(contrast2) figure,imshow(contrast3)



1. **Contrast stretching with changing m Code:**

I=imread('tire.tif'); I2=im2double(I); contrast1=1./(1+(0.2./(I2+eps)).^4) contrast2=1./(1+(0.5./(I2+eps)).^4); contrast3=1./(1+(0.7./(I2+eps)).^4);

imshow(I2) figure,imshow(contrast1) figure,imshow(contrast2) figure,imshow(contrast3)



# Practical 04

**Aim: DFT analysis of image Code:**

a=imread('coins.png'); subplot(2,3,1); imshow(a);

title('Original');

b=im2double(a); c=fft2(b); subplot(2,3,2);

imshow(c);

title('FFT'); d=ifft2(c); subplot(2,3,3);

imshow(d); title('IFFT');

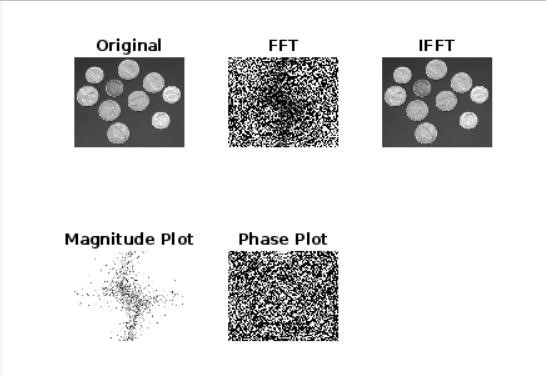
mag=abs(c);

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

title('Magnitude Plot'); ang=angle(c); subplot(2,3,5); imshow(ang);

title('Phase Plot');

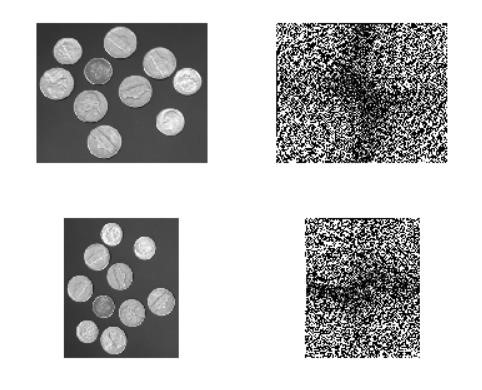
**Output :**



**2. Rotation property Code :**

a=imread('coins.png'); subplot(2,2,1); imshow(a); a1=im2double(a); b=fft2(a1); subplot(2,2,2); imshow(b); c=imrotate(a1,90); subplot(2,2,3); imshow(c); d=fft2(c); subplot(2,2,4); imshow(d);

**Output :**



# Practical 05

## Aim: Walsh transformation

**Code:**

% Getting the name and extension of the image file from the user. a=imread('cameraman.tif');

N=length(a);

% Computing Walsh Transform of the image file. n=log2(N); n=1+fix(n); f=ones(N,N); for x=1:N; for u=1:N p=dec2bin(x-1,n); q=dec2bin(u-1,n); for i=1:n;

f(x,u)=f(x,u)\*((-1)^(p(n+1-i)\*q(i)));

end; end; end;

F=(1/N)\*f\*double(a)\*f;

% Shifting the Fourier spectrum to the center of the frequency square. for i=1:N/2; for j=1:N/2 G(i+N/2,j+N/2)=F(i,j); end; end for i=N/2+1:N; for j=1:N/2 G(i-N/2,j+N/2)=F(i,j);

end; end for i=1:N/2; for j=N/2+1:N G(i+N/2,j-N/2)=F(i,j);

end; end for i=N/2+1:N; for j=N/2+1:N; G(i-N/2,j-N/2)=F(i,j); end; end;

% Computing and scaling the logarithmic Walsh spectrum. H=log(1+abs(G)); for i=1:N

H(i,:)=H(i,:)\*255/abs(max(H(i,:)));

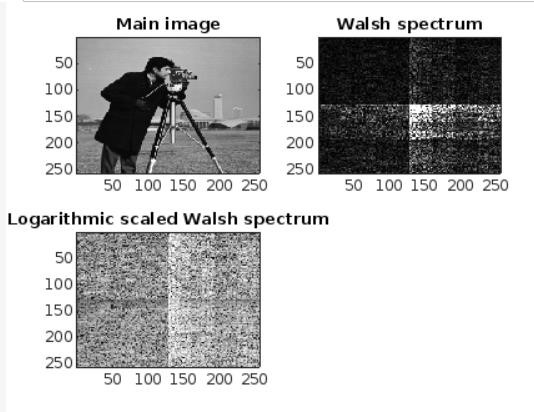
end

% Changing the color map to gray scale (8 bits).

colormap(gray(255));

% Showing the main image and its Walsh spectrum. subplot(2,2,1),image(a),title('Main image'); subplot(2,2,2),image(abs(G)),title('Walsh spectrum'); subplot(2,2,3),image(H),title('Logarithmic scaled Walsh spectrum');

**Output :**



**b. Hadamard transformation code :**

% Getting the name and extension of the image file from the user a=imread('cameraman.tif');

N=length(a);

%Computing Hadamard Transform of the image file n=log2(N); n=1+fix(n); f=ones(N,N); for x=1:N; for u=1:N p=dec2bin(x-1,n); q=dec2bin(u-1,n); for i=1:n; f(x,u)=f(x,u)\*((-1)^(p(n+1-i)\*q(n+1-i)));

end; end; end;

F=(1/N)\*f\*double(a)\*f;

% Shifting the Fourier spectrum to the center of the frequency square.

for i=1:N/2; for j=1:N/2 G(i+N/2,j+N/2)=F(i,j); end; end for i=N/2+1:N; for j=1:N/2 G(i-N/2,j+N/2)=F(i,j);

end; end for i=1:N/2; for j=N/2+1:N G(i+N/2,j-N/2)=F(i,j);

end; end for i=N/2+1:N; for j=N/2+1:N; G(i-N/2,j-N/2)=F(i,j); end; end;

% Computing and scaling the logarithmic Hadamard spectrum. H=log(1+abs(G)); for i=1:N

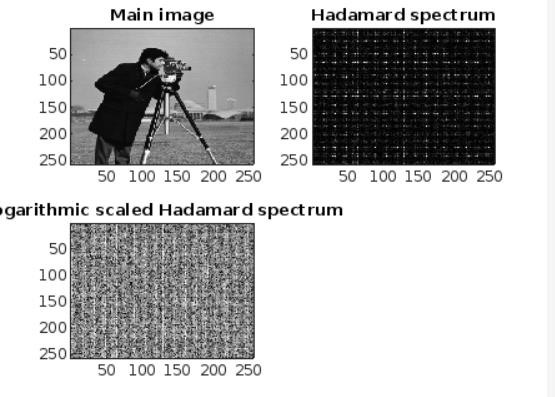
H(i,:)=H(i,:)\*255/abs(max(H(i,:)));

end

% Changing the color map to gray scale (8 bits). colormap(gray(255));

% Showing the main image and its Hadamard spectrum. subplot(2,2,1),image(a),title('Main image'); subplot(2,2,2),image(abs(G)),title('Hadamard spectrum'); subplot(2,2,3),image(H),title('Logarithmic scaled Hadamard spectrum');

**Output :**



**c. descrete consine transformation code :**

a=imread('cameraman.tif');

N=length(a);

F=dct2(double(a));

% Shifting the Fourier spectrum to the center of the frequency square. for i=1:N/2; for j=1:N/2 G(i+N/2,j+N/2)=F(i,j); end; end for i=N/2+1:N; for j=1:N/2 G(i-N/2,j+N/2)=F(i,j);

end; end for i=1:N/2; for j=N/2+1:N G(i+N/2,j-N/2)=F(i,j);

end; end for i=N/2+1:N; for j=N/2+1:N; G(i-N/2,j-N/2)=F(i,j); end; end;

% Computing and scaling the logarithmic Cosine spectrum. H=log(1+abs(G)); for i=1:N

H(i,:)=H(i,:)\*255/abs(max(H(i,:)));

end

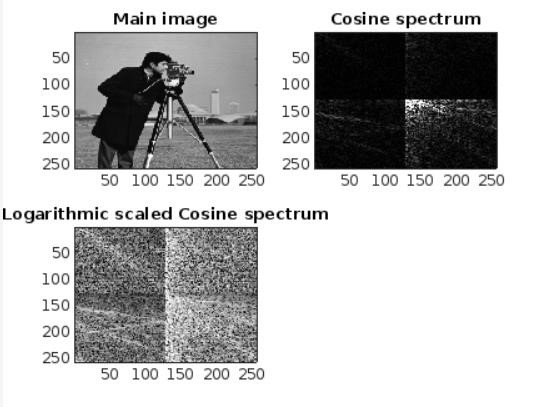
% Changing the color map to gray scale (8 bits).

colormap(gray(255));

% Showing the main image and its Cosine spectrum. subplot(2,2,1),image(a),title('Main image'); subplot(2,2,2),image(abs(G)),title('Cosine spectrum');

subplot(2,2,3),image(H),title('Logarithmic scaled Cosine spectrum');

**output :**



**d. Farr transformation code :**

a=imread('cameraman.tif');

N=length(a); for i=1:N; p=fix(log2(i)); q=i-(2^p); for j=1:N z=(j-1)/N; if(z>=(q-1)/(2^p))&&(z<(q-1/2)/2^p) f(i,j)=(1/(sqrt(N)))\*(2^(p/2)); elseif(z>=(q-1)/(2^p))&&(z<(q/2)/2^p) f(i,j)=(1/(sqrt(N)))\*(-2^(p/2));

else f(i,j)=0; end; end; end;

F=f\*double(a)\*f

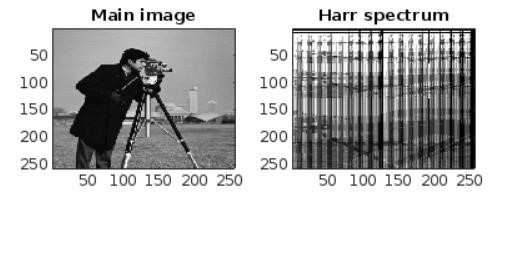
% Changing the color map to gray scale (8 bits).

colormap(gray(255));

% Showing the main image and its Harr spectrum. subplot(2,2,1),image(a),title('Main image');

subplot(2,2,2),image(abs(F)),title('Harr spectrum');

**Output :**



# Practical 06

## Aim: To study the histogram and histogram equalization

**Histogram without inbuilt function**

**Code:**

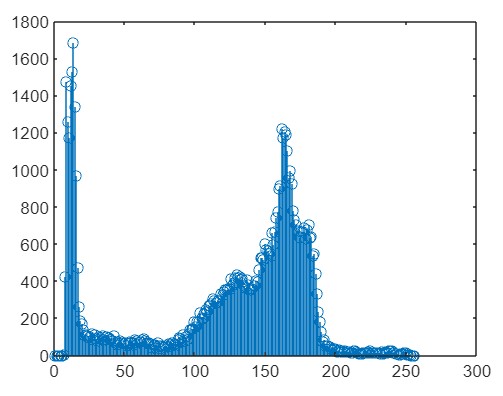
histo=zeros(1,256); I=imread('cameraman.tif'); imshow(I); si=size(I); for i=1:si(1) for j=1:si(2) for g=1:256

if I(i,j)==g histo(g)=histo(g)+1;

end end end end

figure,stem(histo)

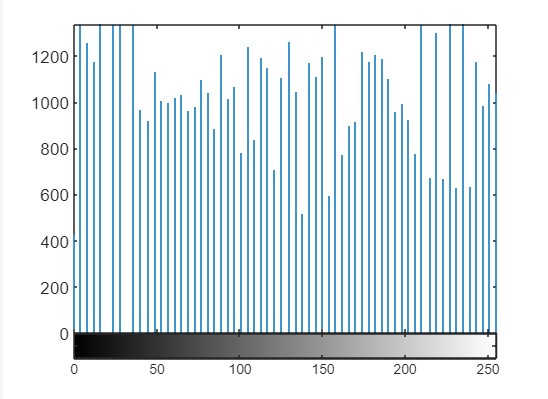
**Output :**



**2. histogram equalization Code :**

I=imread('cameraman.tif'); a=histeq(I); imshow(a); figure,imhist(a)

**Output :**



# Practical 07

## Aim: To perform image enhancement by special filtering

1. **Average Code:**

i=imread('cameraman.tif'); imshow(i); w=fspecial('average',[3 3]); g=imfilter(i,w,'symmetric');

figure,imshow(g,[])

**Output :**



1. **Guassian code:**

i=imread('cameraman.tif'); w=fspecial('gaussian',[3 3],0.5); g=imfilter(i,w,'symmetric'); imshow(g,[])

**Output :**



1. **Laplacian Code:**

i=imread('cameraman.tif'); w=fspecial('laplacian', 0.5); g=imfilter(i,w,'symmetric');

imshow(g,[])

**Output :**



1. **sobel**

**Code :**

i=imread('cameraman.tif'); w=fspecial('sobel'); g=imfilter(i,w,'symmetric');

imshow(g,[])

**Output :**



1. **non linear order static filter Code :**

i=imread('cameraman.tif'); h=ordfilt2(i,1,ones(3,3)); h1=ordfilt2(i,3\*3,ones(3,3)); h2=ordfilt2(i,median(1:3\*3),ones(3,3)); subplot(2,2,1) imshow(i); subplot(2,2,2) imshow(h,[]); subplot(2,2,3) imshow(h1,[]); subplot(2,2,4) imshow(h2,[]);

**Output :**



1. **Median filter Code:**

g=imread('cameraman.tif'); m=medfilt2(g,[3 3]);

imshow(m,[]);

**Output :**



# Practical 08

**Aim: To obtain frequency domain filters from spacial domain**

**a. average**

**code:**

f=imread('cameraman.tif'); h=fspecial('average',[5 5]);

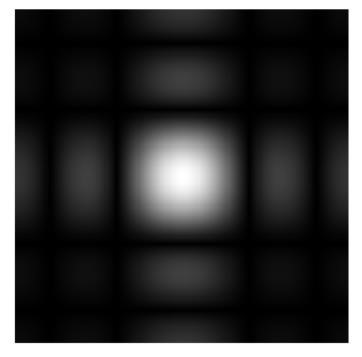
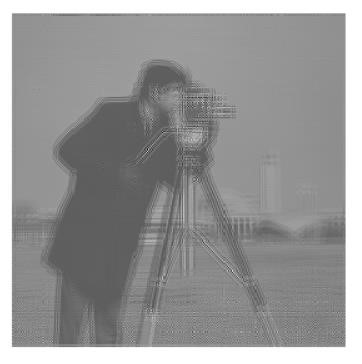
Fs=size(f);

F=fft2(f);

H=freqz2(h,Fs(1),Fs(2));

G=F.\*H; g=ifft2(G); imshow(real(g),[]); figure,imshow(abs(H));

Output :



1. **Guassian Code :**

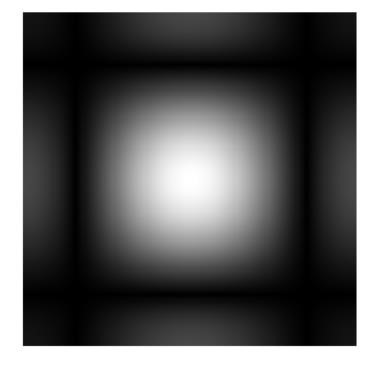
f=imread(‘cameraman.tif’); h=fspecial(‘gaussian’,[3 3],2);

Fs=size(f);

F=fft2(f);

H=freqz2(h,Fs(1),Fs(2)); G=F.\*H; g=ifft2(G); imshow(real(g),[]); figure,imshow(abs(H));

**Output :**



1. **sobel code:**

f=imread('cameraman.tif'); h=fspecial('sobel');

Fs=size(f);

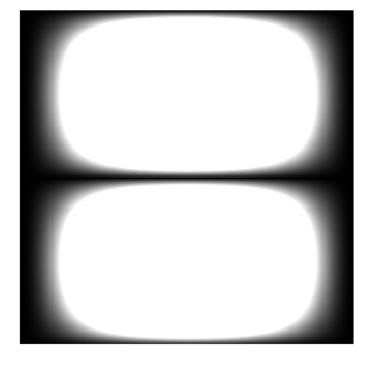
F=fft2(f);

H=freqz2(h,Fs(1),Fs(2));

G=F.\*H; g=ifft2(G); imshow(real(g),[]);

figure,imshow(abs(H));

**Outupt :**



**4b. To generate filters directly in the frequency domain’**

1. **Butterworth LowPass filter**

**code:**

clear; clc;

img=imread('Coins.png'); [X,Y]=size(img); N=input('Order of Filter='); x=ceil(X/2); y=ceil(Y/2); rad=26; for i=1:X for j=1:Y d(i,j)=sqrt((i-x).^2+(j-y).^2);

h(i,j)=1/(1+((d(i,j))/rad).^(2\*N));

end end

fft1=fftshift(fft2(img)); fil=h.\*fft1;

fin=ifft2(fil); fin1=uint8(fin); subplot(2,2,1); imshow(img);

title('Original'); subplot(2,2,2);

imshow(fin1);

title('After LPF'); subplot(2,2,3); surf(h); title('LPF in 3D'); subplot(2,2,4); imshow(h);

title('LPF as Image');

**b. Butterworth high pass:** Code:

clear; clc;

img=imread('Coins.png'); [X,Y]=size(img); N=input('Order of Filter='); x=ceil(X/2); y=ceil(Y/2); rad=26; for i=1:X for j=1:Y

d(i,j)=sqrt((i-x).^2+(j-y).^2);

h(i,j)=1/(1+(rad/d(i,j)).^(2\*N));

end end fft1=fftshift(fft2(img)); fil=h.\*fft1; fin=ifft2(fil); fin1=uint8(fin); subplot(2,2,1); imshow(img);

title('Original'); subplot(2,2,2); imshow(fin1); title('After HPF'); subplot(2,2,3); surf(h); title('HPF in 3D'); subplot(2,2,4); imshow(h); title('HPF as Image');

**C . Guassian low pass:**

**Code:**

clear;

clc;

img=imread('Coins.png'); [X,Y]=size(img); N=input('Order of Filter='); x=ceil(X/2); y=ceil(Y/2); rad=26; for i=1:X for j=1:Y d(i,j)=sqrt((i-x).^2+(j-y).^2);

h(i,j)=exp(-(d(i,j).^2)/(2\*((rad).^2))); end end

fft1=fftshift(fft2(img)); fil=h.\*fft1;

fin=ifft2(fil); fin1=uint8(fin); subplot(2,2,1); imshow(img);

title('Original'); subplot(2,2,2); imshow(fin1); title('After Gaussian LPF'); subplot(2,2,3); surf(h); title('Gaussian LPF in 3D'); subplot(2,2,4); imshow(h); title('Gaussian LPF as Image');

**d. Gussian high pass filter:**

**code:**

clear; clc;

img=imread('Coins.png'); [X,Y]=size(img); N=input('Order of Filter='); x=ceil(X/2); y=ceil(Y/2); rad=26; for i=1:X for j=1:Y d(i,j)=sqrt((i-x).^2+(j-y).^2);

h(i,j)=1-exp(-(d(i,j).^2)/(2\*((rad).^2)));

end end

fft1=fftshift(fft2(img)); fil=h.\*fft1; fin=ifft2(fil); fin1=uint8(fin); subplot(221); imshow(img);

title('Original'); subplot(2,2,2); imshow(fin1); title('After Gaussian HPF'); subplot(2,2,3); surf(h); title('Gaussian HPF in 3D'); subplot(2,2,4); imshow(h); title('Gaussian HPF as Image');

**e. Ideal Low Pass filter: Code:**

clear;

clc;

img=imread('Coins.png'); [X,Y]=size(img); N=input('Order of Filter='); x=ceil(X/2); y=ceil(Y/2); rad=26; for i=1:X for j=1:Y d(i,j)=sqrt((i-x).^2+(j-y).^2);

h(i,j)=double(d(i,j)<=rad); end end

fft1=fftshift(fft2(img)); fil=h.\*fft1;

fin=ifft2(fil); fin1=uint8(fin); subplot(2,2,1); imshow(img);

title('Original'); subplot(2,2,2);

imshow(fin1);

title('After LPF'); subplot(2,2,3); surf(h); title('LPF in 3D'); subplot(2,2,4); imshow(h); title('LPF as Image');

1. **Ideal high pass filter:**

**Code:**

clear;

clc;

img=imread('Coins.png'); [X,Y]=size(img); N=input('Order of Filter='); x=ceil(X/2); y=ceil(Y/2); rad=26; for i=1:X for j=1:Y d(i,j)=sqrt((i-x).^2+(j-y).^2);

h(i,j)=double(d(i,j)>rad); end end

fft1=fftshift(fft2(img)); fil=h.\*fft1;

fin=ifft2(fil); fin1=uint8(fin); subplot(2,2,1); imshow(img);

title('Original'); subplot(2,2,2); imshow(fin1); title('After HPF'); subplot(2,2,3); surf(h); title('HPF in 3D'); subplot(2,2,4); imshow(h); title('HPF as Image');