Institute of Distance & Open Learning
M.Sc.I.T

UNIVERSITY OF MUMBAI

Certificate

This is to certify that **Rajpurohit Harsh Hargopalsingh** Seat no **600257** has successfully completed all the practical of paper titled **"Image Processing"** for M.Sc. (Information Technology) Part 1 Sem 2 in the year 2022-2023

Signature
Faculty In-Charge

Head of the Department

Examiner

INDEX

Sr.no	Title	Date	Teachers Sign
1.	Basics		
2.	Image enhancement		
3.	Filtering in Frequency Domain		
4.	Color Image Processing		
5.	Fourier Related Transforms		
6.	Image compression		
7.	Image compression		
8.	Morphological Image Processing		
9.	Image Segmentation		

Practical No : 1

Basics

A. Program to calculate number of samples required for image.

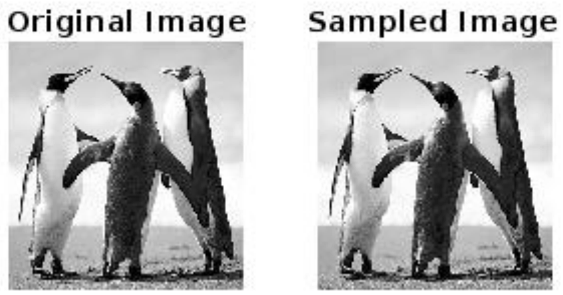
```
clc;
close;
//dimension of the image in inches
m=4;
n=6;
N=400;  //number of dots per inch in each direct
N2=2*N; //number of dots per inch in both horiz
Fs=m*N2*n*N2;
fprintf('Number of samples required to preserve the information in the image=
%d',Fs)
```

Output:

```
Number of samples required to preserve the information in the image= 15360000
```

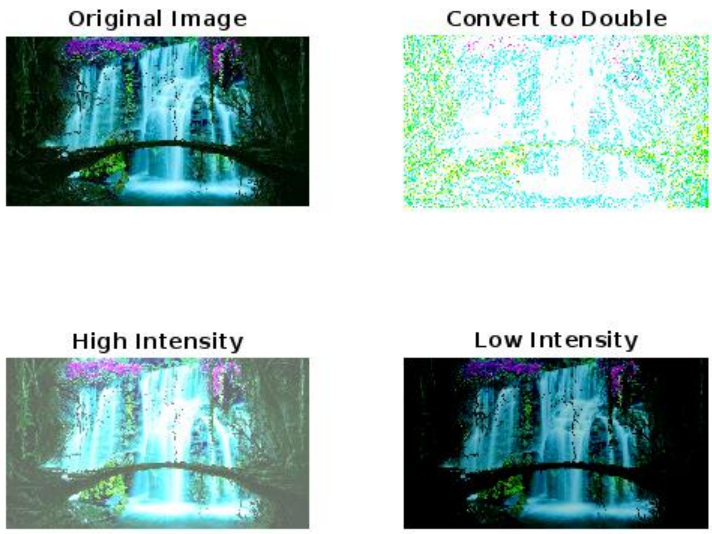
B. Program to study the effects of reducing the spatial resolution of a digital image

```
clc;
clear all;
n=input('Enter the input samples');
img= rgb2gray(imread("IP_img_1.jpg"));
a=size(img);
w=a(2);
h=a(1);
im=zeros(100);
 for i=1:n:h
 for j=1:n:w
 for k=0:n-1
 for l=0:n-1
    im(i+k,j+1)=img(i,j);
 end
 end
 end
 end
 subplot(1,2,1);
 imshow(uint8(img));title('Original Image');
 subplot(1,2,2);
 imshow(uint8(im));title('Sampled Image');
```



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

```
a=imread('Scenary.jpg');
b=double(a);
b1=b+100;
b2=b-50;
subplot(221);
imshow(a);
title('Original Image');
subplot(222);
imshow(b);
title('Convert to Double');
subplot(223);
imshow(uint8(b1));
title('High Intensity');
subplot(224);
imshow(uint8(b2));
title('Low Intensity');
```



D. Program to perform image averaging (image addition) for noise reduction.

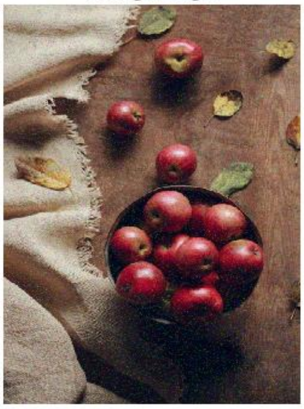
```
close;
clear;
clc;
im=imread('Apple_Image.jpeg');
imshow(im);
title('Orignal image')
figure
imd=im2double(im);
noi=imnoise(imd,'salt & pepper');
imshow(noi);
title('Noisy Image')
figure
s=struct;
for i=1:5
   s(i).noiseimage=imnoise(imd,'salt & pepper');
end
sum=0;
for i=1:5
   sum=sum+s(i).noiseimage;
end
avg=sum/5;
imshow(avg);
title('5 avg image')
figure
s=struct;
for i=1:25
   s(i).noiseimage=imnoise(imd,'salt & pepper');
end
sum=0;
for i=1:25
   sum=sum+s(i).noiseimage;
end
avg=sum/25;
imshow(avg);
title('25 avg image')
figure
s=struct;
for i=1:50
   s(i).noiseimage=imnoise(imd,'salt & pepper');
```

```
end
sum=0;
for i=1:50
   sum=sum+s(i).noiseimage;
end
avg=sum/50;
imshow(avg);
title('50 avg image')
```

Orignal image

Noisy Image

5 avg image



25 avg image

50 avg image

E. Program to compare images using subtraction for enhancing the difference between images.

```
clc;
clear;
close;
warning off;
x=imread('Apple_Image.jpeg');
y=imread('Scenary.jpg');
g=size(x);
y=imresize(y,[g(1),g(2)]);
```

```
figure;
imshow(x);
title('First image');
figure;
imshow(y);
title('Second image');
figure;
imshow(x-y);
title('Difference of two images');
```

Practical No : 2
Image enhancement

A. Basic Intensity Transformation functions

1. Program to perform Image negation

```
clc;
clear all;
a=imread('IP_img_2.jpg');
subplot(1,2,1);
imshow(a);
title('Original img');
[m,n]=size(a);
for i=1:m
for j=1:n
   c(i,j)=255-a(i,j);
end
end
subplot(1,2,2);
imshow(c);
title('Negation img');
```

Output:

Original img

Negation img

2. Program to perform threshold on an image

```
clc;
clear all;
a=imread('IP_img_2.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');
```

3. Log transformation

```
clc;
clear all;
a=imread('IP_img_2.jpg');
b=double(a)
subplot(1,2,1);
imshow(a);
title('Original img');
t=10;
[m,n]=size(b);
for i=1:m
for j=1:n
    c(i,j)=t*log(1+b(i,j));
end
end
subplot(1,2,2);
imshow(uint8(c));
title('Threshold img');
```

Output:

4. Power-law transformations

```
clc;
clear all;
a=imread('IP_img_2.jpg');
b=double(a)
```

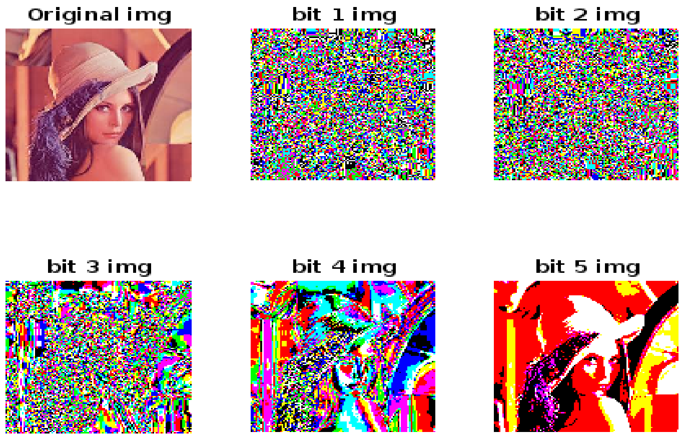
```
subplot(1,2,1);
imshow(a);
title('Original img');
k=1;
gamma=1;
[m,n]=size(b);
for i=1:m
for j=1:n
   c(i,j)=k*(b(i,j)^gamma);
end
end
subplot(1,2,2);
imshow(uint8(c));
title('Power law img');
```

Output:

5. Piecewise linear transformations

```
clc;
clear all;
a=imread('IP_img_2.jpg');
b=double(a)
subplot(2,3,1);
imshow(a);
title('Original img');
f1=bitget(b,1),
subplot(2,3,2);
imshow(f1);
title('bit 1 img');
f2=bitget(b,2),
subplot(2,3,3);
imshow(f2);
title('bit 2 img');
f3=bitget(b,4),
subplot(2,3,4);
imshow(f3);
title('bit 3 img');
f4=bitget(b,6),
```

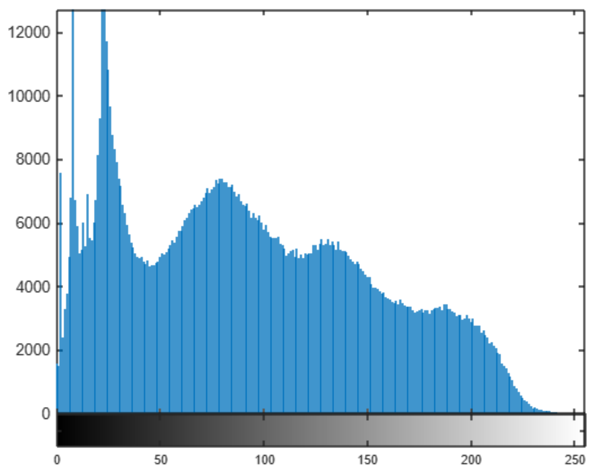
```
subplot(2,3,5);
imshow(f4);
title('bit 4 img');
f5=bitget(b,8),
subplot(2,3,6);
imshow(f5);
title('bit 5 img');
```



B. Intensity transformation and Spatial Filtering

1. Program to plot the histogram of an image and categorise

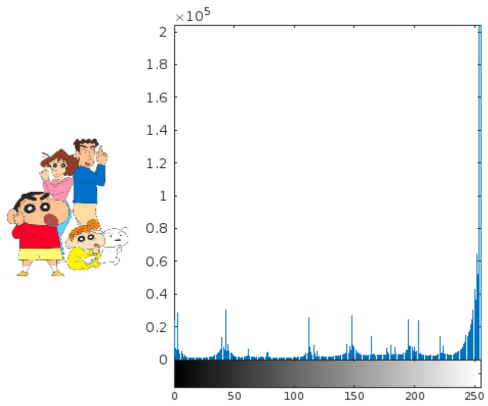
```
clear;
clc;
I=imread('Apple_Image.jpeg');
imshow(I)
figure;
imhist(I);
```

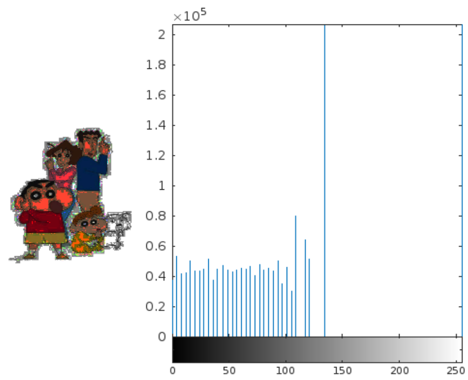


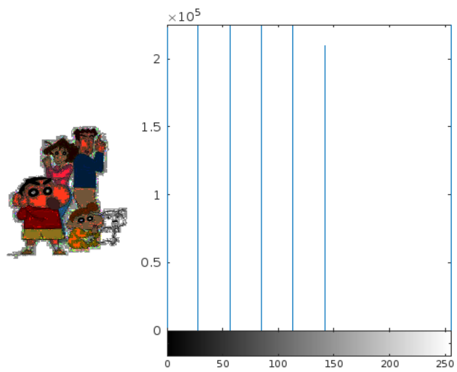
2. Program to apply histogram equalization

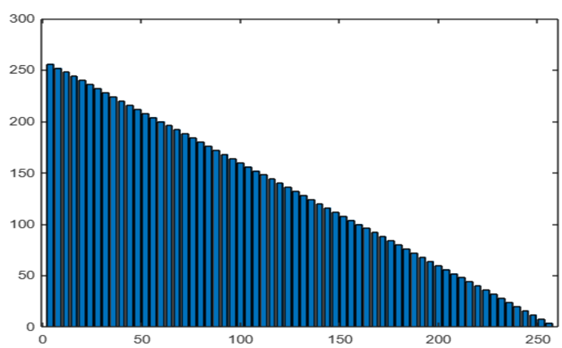
```
I = imread('Cartoon2.jpg');
figure
subplot(1,3,1)
imshow(I)
```

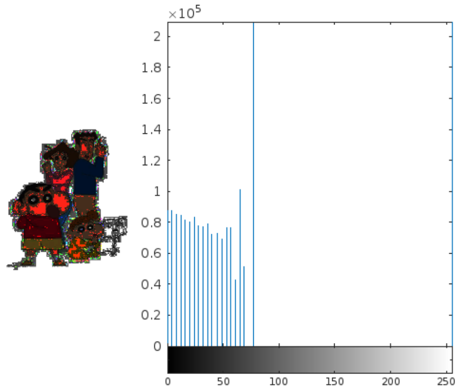
```
subplot(1,3,2:3)
imhist(I)
J = histeq(I);
figure
subplot(1,3,1)
imshow(J)
subplot(1,3,2:3)
imhist(J)
nbins = 10;
K = histeq(I,nbins);
figure
subplot(1,3,1)
imshow(K)
subplot(1,3,2:3)
imhist(K)
target = 256:-4:4;
figure
bar(4:4:256,target)
L = histeq(I,target);
figure
subplot(1,3,1)
imshow(L)
subplot(1,3,2:3)
imhist(L)
```





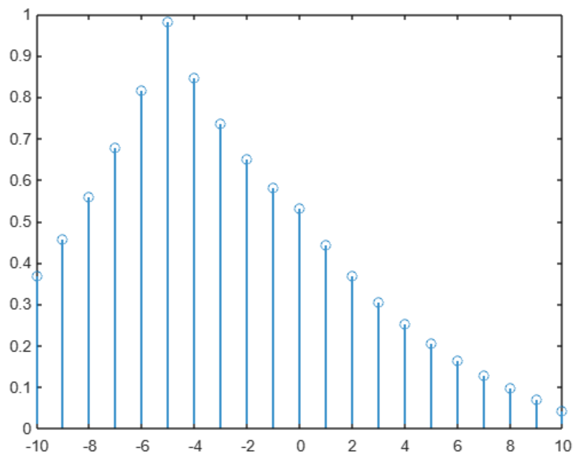






C. Write a program to perform convolution and correlation

```
clear;
clc;
n = 0:15;
x = 0.84.^n;
y = circshift(x,5);
[c,lags] = xcorr(x,y);
stem(lags,c)
n = 0:15;
x = 0.84.^n;
[c,lags] = xcorr(x);
stem(lags,c)
n = 0:15;
x = 0.84.^n;
y = circshift(x,5);
[c,lags] = xcorr(x,y,10,'normalized');
stem(lags,c)
```

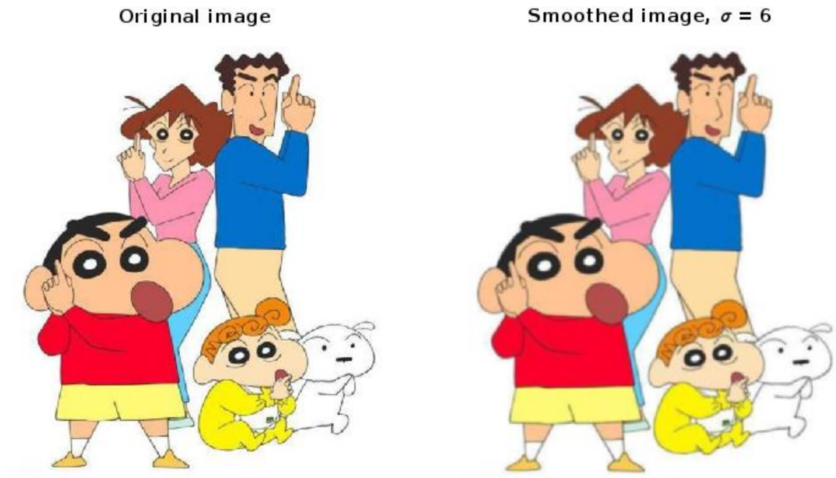


D. Write a program to apply smoothing and sharpening filters on grayscale and color images

a) Low Pass

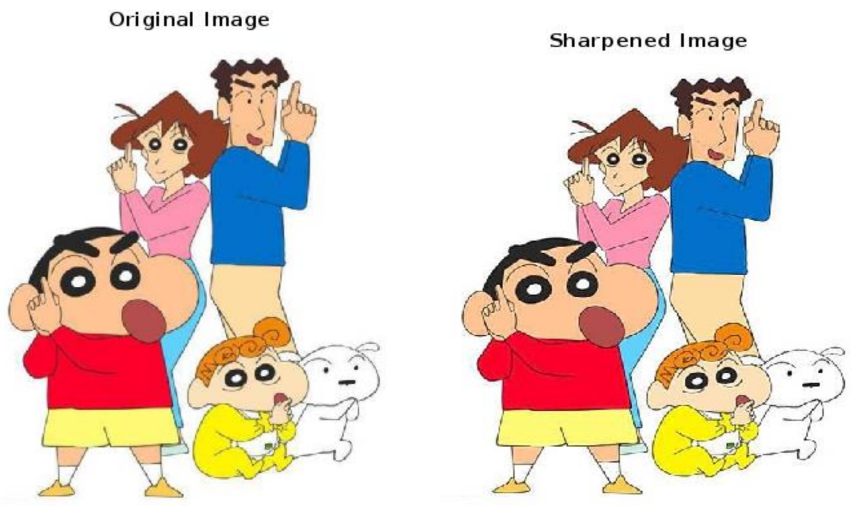
```
clear;
clc;
I = imread('Cartoon2.jpg');
```

```
Iblur1 = imgaussfilt(I,2);
Iblur2 = imgaussfilt(I,4);
Iblur3 = imgaussfilt(I,8);
figure
imshow(I)
title('Original image')
figure
imshow(Iblur1)
title('Smoothed image, \sigma = 6')
```



b) High Pass

```
clear;
clc;
a = imread('Cartoon2.jpg');
imshow(a), title('Original Image');
b = imsharpen(a,'Radius',2,'Amount',1);
figure, imshow(b)
title('Sharpened Image');
```



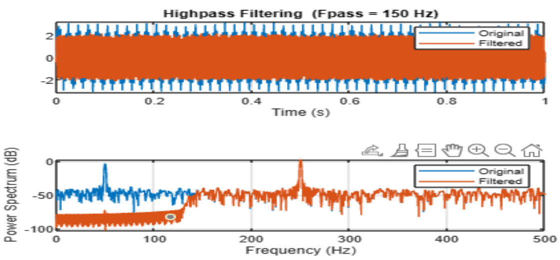
Practical 3
Filtering in Frequency Domain

A. Program to apply Discrete Fourier Transform on an image

```
close;
clear;
clc;
I=imread('Scenary 2.jpeg');
I1=rgb2gray(I);
I2=fft2(I1);
subplot(311);
imshow(I);
subplot(312)
imshow(I1);
subplot(313);
imshow(I2);
I3=ifft2(I2);
I4=uint8(I3);
figure,
subplot(2,1,1);
imshow(I3);
subplot(2,1,2);
imshow(I4);
```

B. Program to apply Low pass and High pass filters in frequency domain

```
clear;
clc;
fs = 1e3;
t = 0:1/fs:1;
x = [1 2]*sin(2*pi*[50 250]'.*t) + randn(size(t))/10;
highpass(x,150,fs)
```



C. Program to apply Laplacian filter in frequency domain

```
k=imread("Scenary .jpeg");
k1=rgb2gray(k);
k1=double(k1);
Laplacian=[0 1 0; 1 -4 1; 0 1 0];
k2=conv2(k1, Laplacian, 'same');
imtool(k1, []);
imtool(abs(k2,[]));
```

D. Program for high frequency emphasis filtering, high boost and homomorphic filtering.

```
alpha = 0.5;
beta = 1.5;
Hemphasis = alpha + beta*H;
plot(1:30,H(1,1:30),'r',1:30,Hemphasis(1,1:30),'b','LineWidth',2);
grid on;
legend('High-pass Filter','High-frequency Emphasis Filter','Location','best');
If = fft2(I, M, N);
Iout = real(ifft2(Hemphasis.*If));
Iout = Iout(1:size(I,1),1:size(I,2));
Ihmf_2 = exp(Iout) - 1;
imshowpair(I, Ihmf_2, 'montage')
imshowpair(Ihmf, Ihmf_2, 'montage')
imshow(histeq(mat2gray(Ihmf)))
imshowpair(histeq(mat2gray(Ihmf)), histeq(mat2gray(Ihmf_2)), 'montage')
```

Practical 4
Image Denoising

A. Program to denoise using spatial mean, median and adaptive mean filtering

```
clc;
clear ;
close all;
a=imread('Apple_Image.jpg');
figure(1),imshow(a),title('origanal image');
b=imnoise(a,'salt & pepper',.02);
figure(2),imshow(b),title('noisy image');
Smax=9;
for i=1:254
  for j=1:254
    n=b(i:i+2,j:j+2);
    Zmin=min(n(:));
    Zmax=max(n(:));
    Zmed=median(n(:));
    sx=3;
    sy=3;
        A1=Zmed-Zmin;
        A2=Zmed-Zmax;
       if (A1>0) && (A2<0)
         B1 = Zxy-Zmin;
         B2 = Zxy-Zmax;
         if (B1>0) && (B2<0)
              b(i:i+2,j:j+2) = n(i,j);
            break;
         else
             b(i:i+2,j:j+2) = Zmed;
            break;
         end
       else
         sx=sx+2;
         sy=sy+2;
         if (sx > Smax) && (sy > Smax)
            b(i:i+2,j:j+2) = n(i,j);
         end
       end
  end
```

```
end
figure(3),imshow(a),title('denoised image');
```

B. Program for Image deblurring using inverse, Weiner filters

```
Ioriginal = imread('Apple_Image.jpg');
imshow(Ioriginal);
title('Original Image');
PSF = fspecial('motion',21,11);
Idouble = im2double(Ioriginal);
blurred = imfilter(Idouble,PSF,'conv','circular');
imshow(blurred);
title('Blurred Image');
wnr1 = deconvwnr(blurred,PSF);
imshow(wnr1);
title('Restored Blurred Image')
noise_mean = 0;
noise_var = 0.0001;
blurred_noisy = imnoise(blurred,'gaussian',noise_mean,noise_var);
imshow(blurred_noisy)
title('Blurred and Noisy Image');
```

Practical 5
Color Image Processing

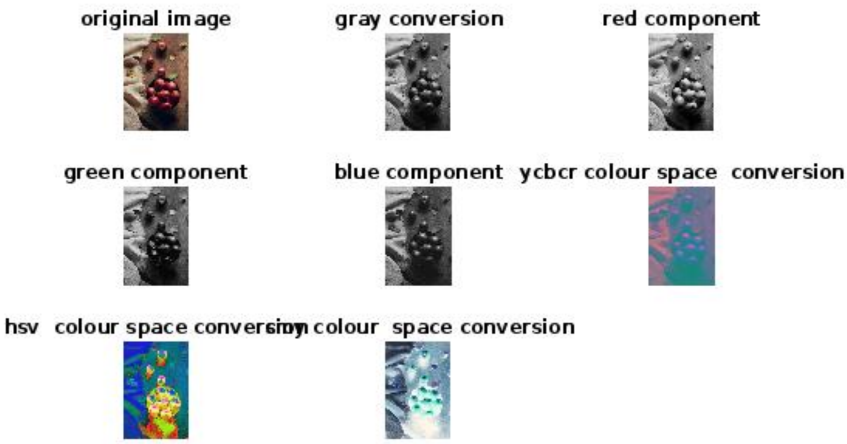
A. Program to read a color image and segment into RGB planes , histogram of color image

```
clc;
clear;
close;
RGB= imread('Apple_Image.jpeg');
imshow(RGB);
figure
ShowColorImage(RGB,'RGB Color Image');
YIQ=rgb2ntsc(RGB);
figure
ShowColorImage(YIQ,'NTSC Image YIQ');
RGB=ntsc2rgb(YIQ);
YCC=rgb2ycbcr(RGB);
figure
ShowColorImage(YCC,'Equivalent HSV image Ycbcr');
RGB=ycbcr2rgb(YCC);
HSV=rgb2hsv(RGB);
figure
ShowColorImage(HSV,'Equivalent HSV image');
RGB=hsv2rgb(HSV);
R=RGB(:,:,1);
G=RGB(:,:,2);
B=RGB(:,:,3);
figure
ShowImage(R,'Red Matrix');
figure
ShowImage(G,'Green Matrix');
figure
ShowImage(B,'Blue Matrix');
```

B. Program for converting from one color model to another model

```
clc;
clear;
close;
warning off;
A=imread('Apple_Image.jpeg');
subplot(4,3,1);
imshow(A);
title('original image');
B=rgb2gray(A);
subplot(4,3,2);
imshow(B);
title('gray conversion');
rgb=im2double(A);
r=rgb(:,:,1);
g=rgb(:,:,2);
b=rgb(:,:,3);
subplot(4,3,3);
imshow(r);
title('red component');
subplot(4,3,4);
imshow(g);
title('green component');
subplot(4,3,5);
imshow(b);
title('blue component');
ycbcr_img=rgb2ycbcr(A);
subplot(4,3,6);
imshow(ycbcr_img);
title('ycbcr colour space  conversion');
hsv_img=rgb2hsv(A);
subplot(4,3,7);
imshow(hsv_img);
title('hsv  colour space conversion');
cmy_img=imcomplement(A);
subplot(4,3,8);
imshow(cmy_img);
title('cmy colour  space conversion');
num=.5*((r-g)+(r-b));
den=sqrt(r-g).^2+(r-g).*(g-b);
```

```
theta=acos(num./den);
H=theta;
Hb greater than g=2*pi-Hb greater than g;
H=H/(2*pi);
num1=min(min(r,g),b);
den1=r+g+b;
den(den==0)=eps;
S=1-3.*num1./den1;
H(S==0)=0;
I=(r+g+b)/3;
hsi=cat(3,H,S,I);
subplot(4,3,9);
imshow(hsi);
title('HSI');
subplot(4,3,10);
imshow(H);
title('hue component');
subplot(4,3,11);
imshow(S);
title('saturation component');
subplot(4,3,12);
imshow(I);
title('brightness component');
```



C. Program to apply false colouring(pseudo) on a gray scale image

Step1:

```
[filename,pathname]=uigetfile('*_*','Select grayscale image');
filewithpath=strcat(pathname,filename);
img=imread(filewithpath);
[r,c]=size(img);
```

```
R=uint8(zeros(r,c));
G=uint8(zeros(r,c));
B=uint8(zeros(r,c));
nos=input('Enter number of Slices:');
color=zeros(nos,3);
interval=zeros(nos,2);
for i=1:nos
   interval(i,:)=input('Enter Intensity Interval');
   color(i,:)=uint8(255*uisetcolor('Select Color for Pseudo coloring'));
end
for s=1:nos
   slice=interval(s,:); LL=slice(1);UL=slice(2);
   rgb=color(s,:); red=rgb(1); green=rgb(2); blue=rgb(3);
   for i=1:r
      for j=1:c
         if img(i,j)>=LL && img(i,j)<=UL
            R(i,j)=red;
            G(i,j)=green;
            B(i,j)=blue;
         end
      end
   end
end
imgc=cat(3,R,G,B);
imshow(imgc);
title('Pseudo color image');
```

Step2:

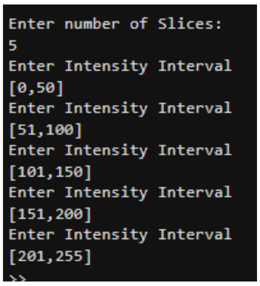
```
[filename,pathname]=uigetfile('*_*','Select grayscale image');
filewithpath=strcat(pathname,filename);
img=imread(filewithpath);
[r,c]=size(img);
R=uint8(zeros(r,c));
G=uint8(zeros(r,c));
B=uint8(zeros(r,c));
coltype=input('Enter choice of color map as string');
colmap=uint8(255*coltype);
if (length(colmap)~=256)
   colmap=imresize(colmap,[256,3]);
end
```

```
for i=1:r
   for j=1:c
      R(i,j)=colmap(img(i,j)+1,1);
      G(i,j)=colmap(img(i,j)+1,2);
      B(i,j)=colmap(img(i,j)+1,3);
   end
end
imgc=cat(3,R,G,B);
imshow(imgc);
title('Pseudo color image');
```

Step3:

```
[filename,pathname]=uigetfile('*_*','Select grayscale image');
filewithpath=strcat(pathname,filename);
img=imread(filewithpath);
[r,c]=size(img);
R=zeros(r,c);
G=zeros(r,c);
B=zeros(r,c);
theta=linspace(0,4*pi,256);
Rv=255*abs(sin(theta)+pi);
Gv=255*abs(sin(theta)-(pi/3));
Bv=255*abs(sin(theta+(pi)/3));
for i=1:r
   for j=1:c
      R(i,j)=Rv(img(i,j)+1);
      G(i,j)=Gv(img(i,j)+1);
      B(i,j)=Bv(img(i,j)+1);
   end
end
imgc=cat(3,uint8 (R), uint8 (G), uint8 (B));
imshow(imgc);
title('Pseudo color image');
```

Step1

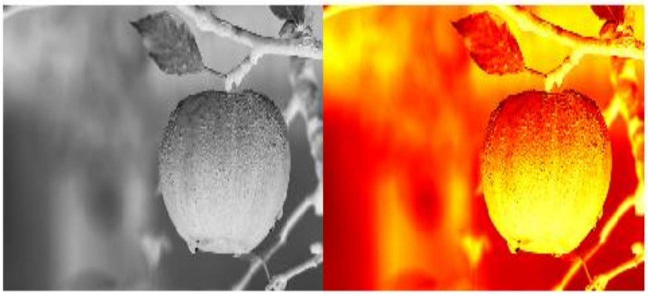


Step2

```
>> Pract5C2
Enter choice of color map as string
hot
>>
```

```
>> montage({img,imgc})
>>
```

Pseudo color image



Step3

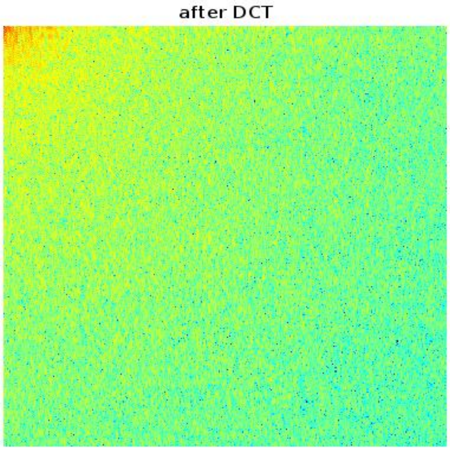
Pseudo color image

Practical 6

Fourier Related Transforms

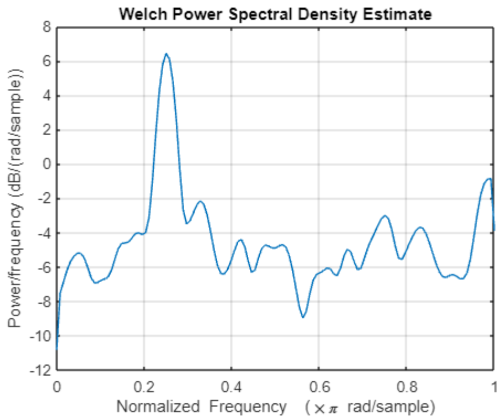
A. Program to compute Discrete Cosine Transforms

```
clc;
I=imread('Scenary 3.jpeg');
I=I(100:460,100:460);
A=double(I);
B=zeros(size(A));
Temp=zeros(size(A));
[M,N]=size(A);
x=1:M;
x=repmat(x',1,N);
y=repmat(1:N,M,1);
for i=1:M
   for j=1:N
      if(i==1)
         AlphaP=sqrt(1/M);
      else
         AlphaP=sqrt(2/M);
      end
      if(j==1)
         AlphaQ=sqrt(1/N);
      else
         AlphaQ=sqrt(2/N);
      end
      cs1=cos((pi*(2*x-1)*(i-1))/(2*M));
      cs2=cos((pi*(2*y-1)*(j-1))/(2*N));
      Temp=A.*cs1.*cs2;
      B(i,j)=AlphaP*AlphaQ*sum(sum(Temp));
   end
end
figure(1);
imshow(I);
title('Original image');
figure(2);
imshow(log(abs(B)),[]);
colormap(jet);
title('after DCT');
```



B. Walsh -Hadamard Transforms

```
rng default
n = 0:319;
x = cos(pi/4*n)+randn(size(n));
pxx = pwelch(x);
pwelch(x)
```



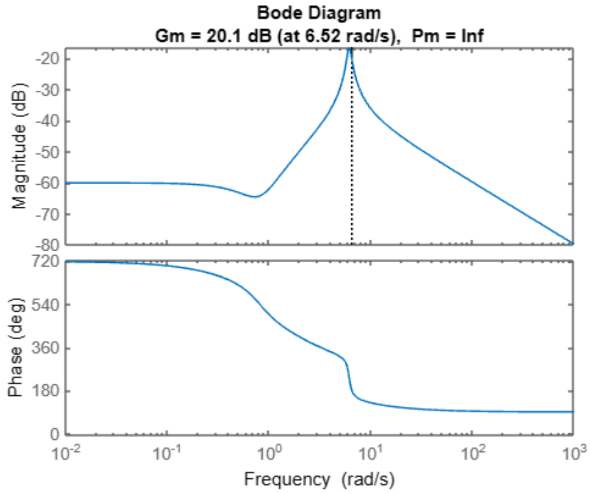
C. Haar Transform Wavelet

Step1

```
image_input=imread('Scenary 5.jpeg');
delta=0.5;
harr_wt(image_input,delta)
if margin>2
   error('harr_wt:TooManyInputs',...
      'requires at most 1 optional inputs');
end
if (margin==1)
   delta=0.01;
end
if (delta>1 || delta<0)
   error('harr_wt: Delta must be a value between 0 & 1');
```

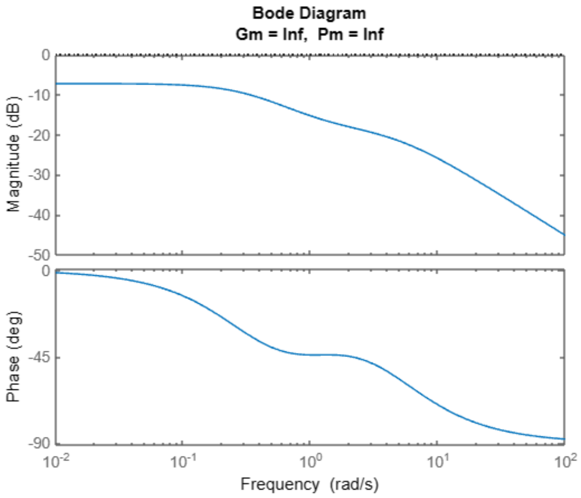
```
end
H1=[0.5 0 0 0 0.5 0 0 0;0.5 0 0 0 -0.5 0 0 0;0 0.5 0 0 0 0.5 0 0;0 0.5 0 0 0 -0.5 0 0; 0 0
0.5 0 0 0 0.5 0; 0 0 0.5 0 0 0 -0.5 0;0 0 0 0.5 0 0 0 0.5;0 0 0 0.5 0 0 0 -0.5];
H2=[0.5 0 0.5 0 0 0 0 0;0.5 0 -0.5 0 0 0 0 0;0 0.5 0 0.5 0 0 0 0;0 0.5 0 -0.5 0 0 0 0; 0 0 0
0 1 0 0 0; 0 0 0 0 0 1 0 0;0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 1];
H3=[0.5 0.5 0 0 0 0 0 0;0.5 -0.5 0 0 0 0 0 0;0 0 1 0 0 0 0 0; 0 0 0 1 0 0 0 0;0 0 0 0 1 0 0
0;0 0 0 0 0 1 0 0;0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 1];
H1=normc(H1);
H2=normc(H2);
H3=normc(H3);
H=H1*H2*H3;
x=double(imread(image_input));
len=length(size(x));
if len~=2
   error('harr_wt: Input image must be a grey image, use "harr_wt_rgb"function to
compress RGB Images');
end
y=zeros(size(x));
[r,c]=size(x);
for i=0:8:r-8
   for j=0:8:c-8
      p=i+1;
      q=j+1;
      y(p:p+7,q:q+7)=(H')*x(p:p+7,q:q+7)*H;
   end
end
figure;
imshow(x/255);
title('Original image');
n1=nnz(y);
z=y;
m=max(max(y));
y=y/m;
y(abs(y)<delta)=0;
y=y*m;
n2=nnz(y);
for i=0:8:r-8
   for j=0:8:c-8
      p=i+1;
      q=j+1;
```

```
        z(p:p+7,q:q+7)=H*y(p:p+7,q:q+7)*H';
    end
end
figure;
imshow(z/255);
imwrite(x/255,'Original.tif');
imwrite(z/255,'Compressed.tif');
title('Compressed image');
```



Step2:

```
if margin>2
    error('harr_wt:TooManyInputs',...
        'requires at most 1 optional inputs');
end
if (margin==1)
    delta=0.01;
end
if (delta>1 || delta<0)
    error('harr_wt: Delta must be a value between 0 & 1');
end
r=rgb(:,:,1);
g=rgb(:,:,2);
b=rgb(:,:,3);
r=hwt(r,delta);
g=hwt(g,delta);
b=hwt(b,delta);
rgb(:,:,1)=r;
rgb(:,:,2)=g;
rgb(:,:,3)=b;
imwrite(rgb/255,'compressed_rgb.jpg');
```

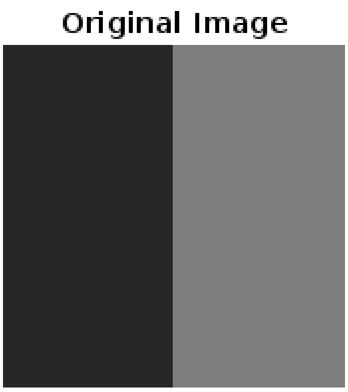


Bode Diagram
Gm = Inf, Pm = Inf
Magnitude (dB)
Phase (deg)
Frequency (rad/s)

Practical 7
Image compression

A. Program to apply compression and decompression algorithm on an image (Arithmetic, Huffman and LZW coding techniques.

```
clear;
clc;
I=[39 39 126 126
   39 39 126 126
   39 39 126 126
   39 39 126 126];
subplot(1,2,1);
imshow(uint8(I));
title('Original Image');
[m,n]=size(I);
Totalcount=m*n;
symbols=unique(I);
counts=histcounts(I(:),symbols);
pro=counts./Totalcount;
dict=huffmandict(symbols,pro);
newvec=reshape(I.',1,[]);
hcode=huffmanenco(newvec,dict);
dhsig1=huffmandeco(hcode,dict);
dhsig=uint8(dhsig1);
back=reshape(dhsig,4,4);
subplot(1,2,2);
imshow(back.');
title('decoded image');
```



Practical 8
Morphological Image Processing

A. Program to apply erosion, dilation, opening, closing

a. Erosion of image

```
clear;
close;
clc;
I=imread('Scenary.jpg');
se1=strel('disk',11);
er=imerode(I,se1);
figure,
imshow(I);
title('Original Image');
figure,
imshow(er);
title('Image after erosion');
```

b. Dilation Of Image

```
A=imread("Apple_Image.jpeg");
se2=strel('ball',5,5);
de=imdilate(A,se2);
figure,
imshow(A);
title('Original Image');
figure,
imshow(de);
title('Image after dilation');
```

Original Image

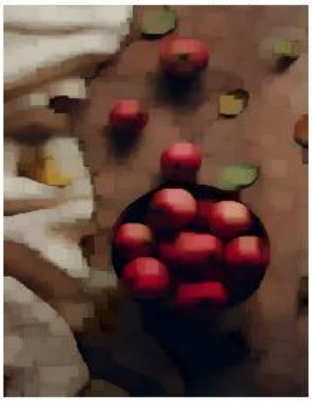
Image after dilation

c. Opening Operation:

```
close;
clear;
clc;
I=imread('Apple_Image.jpeg');
se1=strel('square',20);
io=imopen(I,se1);
figure,
imshow(I);
title('Original Image');
figure,
imshow(io);
title('Image after opening operation');
```

Original Image

Image after opening operation

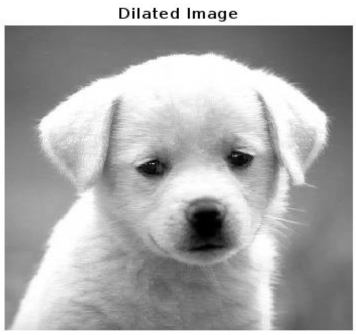


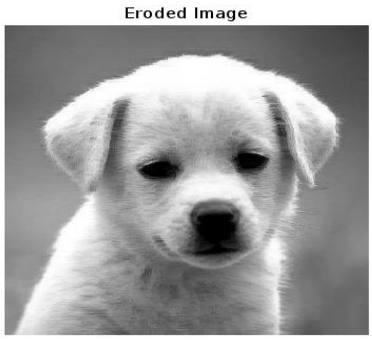
d. Closing Operation:

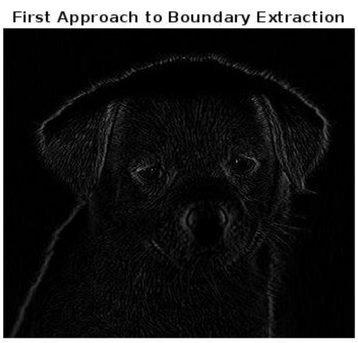
```
A=imread('Scenary.jpg');
se2=strel('disk', 15);
ic=imclose(A,se2);
figure,
imshow(A);
title('Original Image');
figure,
imshow(ic);
title('Image after closing operation');
```

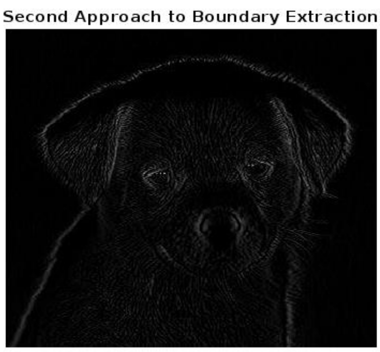
B. Program for detecting boundary of an image

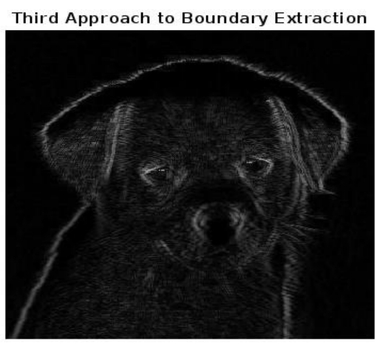
```
clc;
clear;
close;
A=imread('Grayscale.jpg');
B=[0 1 0: 1 1 1: 0 1 0];
A1=imdilate(A,B);
A2=imerode(A,B);
A3=A-A2;
A4=A1-A;
A5=A1-A2;
imshow(A), title('Original Image');
figure, imshow(A1), title('Dilated Image');
figure, imshow(A2), title('Eroded Image');
figure, imshow(A3), title('First Approach to Boundary Extraction');
figure, imshow(A4), title('Second Approach to Boundary Extraction');
figure, imshow(A5), title('Third Approach to Boundary Extraction');
```







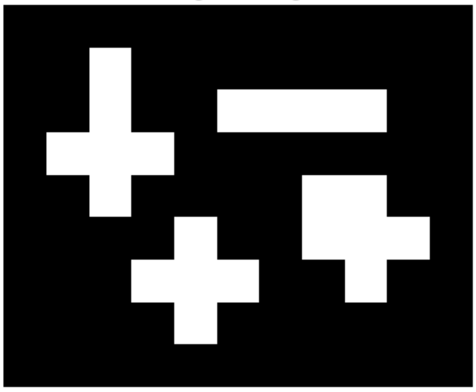




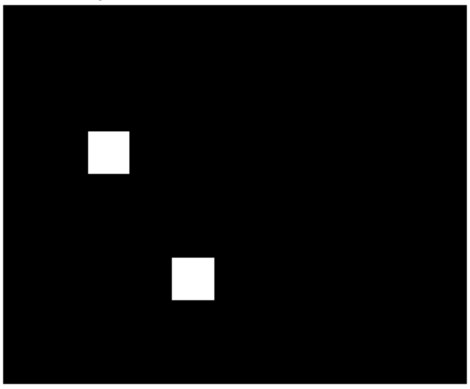
C. Program to apply Hit-or-Miss transform

```
clc;
clear;
close;
warning off
A=[0 0 0 0 0 0 0 0 0 0 0
   0 0 1 0 0 0 0 0 0 0 0
   0 0 1 0 0 1 1 1 1 0 0
   0 1 1 1 0 0 0 0 0 0 0
   0 0 1 0 0 0 0 1 1 0 0
   0 0 0 0 1 0 0 1 1 1 0
   0 0 0 1 1 1 0 0 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];
imshow(A);
title('Original Image');
se1=[0 1 0;1 1 1;0 1 0];
se2=~se1;
bw=bwhitmiss(A,se1,se2);
figure;
imshow(bw);
title('Output of Hit and Miss Transform');
```

Original Image



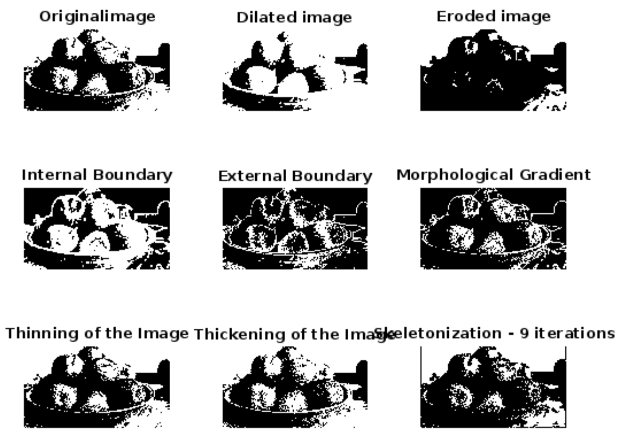
Output of Hit and Miss Transform



D. Program to apply morphological gradient on an image

```
clc;
close;
clear;
myorigimg = imread('Apple_Image.jpg');
myorigimg = imbinarize(rgb2gray(myorigimg));
subplot(3, 3, 1);
imshow(myorigimg);title('Originalimage');
se = strel('disk', 9);
```

```
mydilatedimg = imdilate(myorigimg, se);
subplot(3, 3, 2);
imshow(mydilatedimg);title('Dilated image');
myerodedimg = imerode(myorigimg, se);
subplot(3, 3, 3);
imshow(myerodedimg);title('Eroded image');
internalboundimg = mydilatedimg & ~ myerodedimg;
subplot(3, 3, 4);
imshow(internalboundimg,[]);title('Internal Boundary');
externalboundimg = mydilatedimg & ~myorigimg;
subplot(3, 3, 5);
imshow(externalboundimg,[]);title('External Boundary');
mymorphgradimg = imsubtract(myorigimg,myerodedimg);
subplot(3, 3, 6);
imshow(mymorphgradimg,[]);title('Morphological Gradient');
thinf = bwmorph(myorigimg,'thin');
subplot(3,3,7);
imshow(thinf);title('Thinning of the Image');
thickf = bwmorph(myorigimg,'thicken');
subplot(3,3,8);
imshow(thickf);title('Thickening of the Image');
skelf100 = bwmorph(myorigimg,'skel',9);
subplot(3,3,9);
imshow(skelf100);title('Skeletonization - 9 iterations');
```



E. Program to apply Top-Hat/Bottom-hat Transformations

TopHat

```
original = imread('Scenary 4.jpeg');
figure(1);
imshow(original);
```

```
se = strel('disk',12);
tophat_trans = imtophat(original,se);
figure(2);
imshow(tophat_trans);
```

BottomHat

```
original = imread('Apple_Image.jpeg');
figure(1);
imshow(original);
se = strel('disk',3);
J = imsubtract(imadd(original,imtophat(original,se)),imbothat(original,se));
figure(2);
imshow(J);
```

Practical 9

Image Segmentation

A. Program for Edge detection using

a. Sobel

```
input_image = imread('Apple_Image.jpg');
input_image = uint8(input_image);
figure, imshow(input_image); title('Input Image');
input_image = rgb2gray(input_image);
input_image = double(input_image);
filtered_image = zeros(size(input_image));
Mx = [-1 0 1; -1 0 1; -1 0 1];
My = [-1 -1 -1; 0 0 0; 1 1 1];
for i = 1:size(input_image, 1) - 2
   for j = 1:size(input_image, 2) - 2
      Gx = sum(sum(Mx.*input_image(i:i+2, j:j+2)));
      Gy = sum(sum(My.*input_image(i:i+2, j:j+2)));

      filtered_image(i+1, j+1) = sqrt(Gx.^2 + Gy.^2);
   end
end
filtered_image = uint8(filtered_image);
figure, imshow(filtered_image); title('Filtered Image');
thresholdValue = 100;
output_image = max(filtered_image, thresholdValue);
output_image(output_image == round(thresholdValue)) = 0;
output_image = im2binarize(output_image);
figure, imshow(output_image);
title('Edge Detected Image');
```

Input Image

Filtered Image

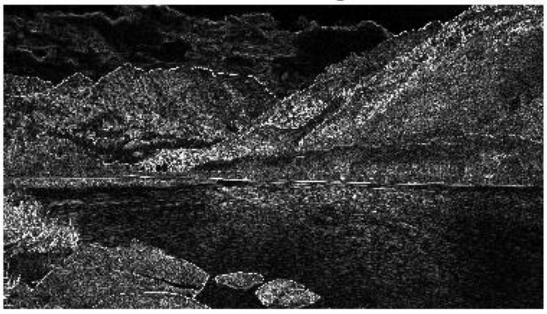
b. Prewitt

```
input_image = imread('Scenary 2.jpeg');
input_image = uint8(input_image);
figure, imshow(input_image); title('Input Image');
```

```
input_image = rgb2gray(input_image);
input_image = double(input_image);
filtered_image = zeros(size(input_image));
Mx = [-1 0 1; -2 0 2; -1 0 1];
My = [-1 -2 -1; 0 0 0; 1 2 1];
for i = 1:size(input_image, 1) - 2
for j = 1:size(input_image, 2) - 2
    Gx = sum(sum(Mx.*input_image(i:i+2, j:j+2)));
Gy = sum(sum(My.*input_image(i:i+2, j:j+2)));
filtered_image(i+1, j+1) = sqrt(Gx.^2 + Gy.^2);
end
end
filtered_image = uint8(filtered_image);
figure, imshow(filtered_image); title('Filtered Image');
thresholdValue = 100;
output_image = max(filtered_image, thresholdValue);
output_image(output_image == round(thresholdValue)) = 0;
output_image = im2binarize(output_image);
figure, imshow(output_image);
title('Edge Detected Image');
```

Input Image

Filtered Image



c. Marr-Hildreth and Canny

```
I = rgb2gray(imread("Scenary 5.jpeg"));
subplot(2, 4, 1),
imshow(I);
title("Gray Scale Image");
L = edge(I, 'Roberts');
subplot(2, 4, 4),
imshow(L);
title("Robert");
M = edge(I, 'log');
subplot(2, 4, 5),
imshow(M);
```

```
title("Log");
M = edge(I, 'zerocross');
subplot(2, 4, 6),
imshow(M);
title("Zerocross");
N = edge(I, 'Canny');
subplot(2, 4, 7),
imshow(N);
title("Canny");
```

Input image for this practical:scenary.jpg