Practical – 1

Aim: 2D Linear Convolution, Circular Convolution between two 2D matrices.

(A) 2D Linear Convolution

Code:

```
clc;
x =[1 ,2 ,3;4 ,5 ,6;7 ,8 ,9];
h = [1 ,1;1 ,1;1 ,1];
y = conv2(x,h);
disp(y,'Linear 2D convolution y=');
```

Output:

```
Linear 2D convolution y=
   1.    3.    5.    3.
   5.    12.   16.   9.
   12.   27.   33.   18.
   11.   24.   28.   15.
   7.    15.   17.   9.
```

(B) Circular Convolution

Code:

```
clc ;
x = input("Enter the values of x(n)");
h = input("Enter the values of h(n)");
X = fft2(x);
H = fft2(h);
Y = X.*H;
y = ifft(Y);
disp (y, 'Circular Convolution Result y = ');
```

Output:

```
Enter the values of x(n)[1,2;3,4]

Enter the values of h(n)[5,6;7,8]


 Circular Convolution Result y =

   70.   68.
   62.   60.
```

Practical – 2

Aim: Circular Convolution expressed as linear convolution plus alias.

Code:

```
clc ;
x = [1 ,2;3 ,4];
h = [5 ,6;7 ,8];
y = conv2 (x,h);
y1 = [y(:,1)+y(:,$),y(:,2)];
y2 = [y1(1,:)+y1($,:);y1(2,:)];
disp(y,'Linear Convolution Result y =');
disp(y2,'Circular Convolution Expressed as Linear Convolution plus alias = ');
```

Output:

```
Linear Convolution Result y =

   5.    16.   12.
   22.   60.   40.
   21.   52.   32.

 Circular Convolution Expressed as Linear Convolution plus alias =

   70.   68.
   62.   60.
```

Practical – 3

Aim: Linear Cross correlation of a 2D matrix, Circular correlation between two signals and Linear auto correlation of a 2D matrix.

(A) Linear Cross Correlation of a 2D matrix

Code:

```
clc ;
x = [3 ,1;2 ,4];
h1 = [1 ,5;2 ,3];
h2 = h1 (:,$ : -1:1);
h = h2($: -1:1 ,:);
y = conv2 (x,h);
disp(y, "Linear Cross Correlation Result y = ");
```

Output:

```
Linear Cross Correlation Result y =
   9.    9.    2.
   21.   24.   9.
   10.   22.   4.
```

(B) Circular Correlation between two signals

Code:

```
clc ;
x = [1 ,5;2 ,4];
h = [3 ,2;4 ,1];
h = h(:,$ : -1:1) ;
h = h($: -1:1 ,:);
X = fft2 (x);
H = fft2 (h);
Y = X.*H;
y = ifft (Y);
disp(y, "Circular Correlation Result y = ");
```

Output:

```
Circular Correlation Result y =
   37.   23.
   35.   25.
```

(C) Linear auto Correlation of a 2D matrix

Code:

```
clc ;
x1 = [1 ,1;1 ,1];
x2 = x1 (:,$ : -1:1);
x2 = x2($: -1:1 ,:);
x = conv2 (x1 ,x2);
disp(x, "Linear auto Correlation Result x = ");
```

Output:

```
Linear auto Correlation Result x =
   1.   2.   1.
   2.   4.   2.
   1.   2.   1.
```

Practical – 4

Aim: DFT of 4x4 gray scale image.

Code:

```
clc ;
f = [1 ,1 ,1 ,1;1 ,1 ,1 ,1;1 ,1 ,1 ,1;1 ,1 ,1 ,1];
t = fft2(f);
disp(t, "2D DFT of given 2D image = ");
```

Output:

```
2D DFT of given 2D image =

  16.   0.   0.   0.
  0.    0.   0.   0.
  0.    0.   0.   0.
  0.    0.   0.   0.
```

Practical – 5

Aim: Compute discrete cosine transform, Program to perform KL transform for the given 2D matrix

(A) Discrete Cosine transform of an image

Code:

```
//OS: Windows 7
//Scilab Version: Scilab 5.4.1
//one dimensional cosine transform
clc;
clear all;
//f=[1 2 4 7];        //Input: A row matrix
//Input ex. f=[1 2 4 7]
N=4;//finding length ofinput sequence
F=zeros(1,N);//cosine transform of input
//C=zeros(N,N);
for k=1:N
  for n=1:N
    if (k-1)==0
      C(k,n)=inv(sqrt(N));      //cosine transform matrix
    else
      C(k,n)=sqrt(2)*inv(sqrt(N))*cos(%pi*(2*(n-1)+1)*(k-1)/(2*N));
    end
    disp(C(k,n));
  end
end
```

Output:

```
   0.5
   0.5
   0.5
   0.5

   0.6532815
   0.2705981
  -0.2705981
  -0.6532815

   0.5
  -0.5
  -0.5
   0.5

   0.2705981
  -0.6532815
   0.6532815
  -0.2705981
```

(B) KL transform for the given 2D matrix

Code:

```
clear ;
clc ;
X = [4 ,3 ,5 ,6;4 ,2 ,7 ,7;5 ,5 ,6 ,7];
//X=[4 -2; -1 3];
[m , n ]= size (X) ;
A = [0];
E = [0];
for i =1: n
    A= A + X (: , i ) ;
    E= E + X (: , i ) * X (: , i)';
end
```

```
mx = A / n ; //mean matrix
E = E / n;
C = E - mx * mx '; // covariance matrix C = E[ xx '] −mx∗mx'
[V , D ] = spec ( C ) ; // eigenvalues and eigenvectors
d = diag ( D ) ; // diagonal elements od eigenvalues
disp(d)
[d , i ] = gsort ( d ) ; // sorting the elements of D in descending order
   for j = 1: length (d )
      T (: , j )= V (: , i ( j ) ) ;
   end
T = T'
disp (d , ' Eigen Values are U = ' )
disp (T , ' The eigenvector matrix T = ' )
disp (T , ' The KL tranform basis is = ' )
 //KL transform
for i = 1: n
   Y (: , i )= T * X (: , i ) ;
end
disp (Y , 'KL transformation of the input matrix Y = ' )
 // Reconstruction
for i = 1: n
   x (: , i )= T'* Y (: , i ) ;
end
disp (x , ' Reconstruct matrix of the given sample matrix X = ' )
```

Output:

```
0.0264211
 0.2147417
6.1963372

 Eigen Values are U =

  6.1963372
  0.2147417
  0.0264211

 The eigen vector matrix T =

  0.4384533   0.8471005   0.3002988
  0.4460381  -0.4951684   0.7455591
 -0.780262    0.1929481   0.5949473

 The KL tranform basis is =

  0.4384533   0.8471005   0.3002988
  0.4460381  -0.4951684   0.7455591
 -0.780262    0.1929481   0.5949473

 KL transformation of the input matrix Y =

  6.6437095   4.5110551   9.9237632   10.662515
  3.5312743   4.0755729   3.2373664   4.4289635
  0.6254808   1.0198466   1.0190104   0.8336957

 Reconstruct matrix of the given sample matrix X =

  4.   3.   5.   6.
  4.   2.   7.   7.
  5.   5.   6.   7.
```

Practical – 6

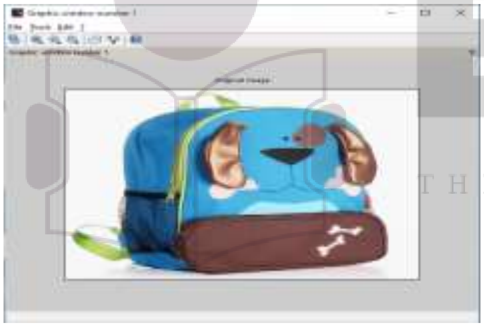
Aim: Brightness enhancement of an image, Contrast Manipulation, image negative.

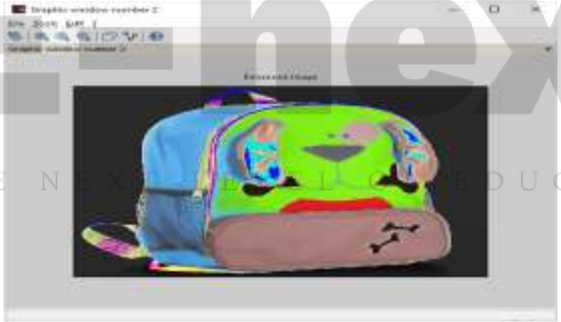
(A) Brightness enhancement of an image

Code:

```
clc;
a = imread ("C:\Users\sushil\Downloads\children_bag.jpg");
b = double (a) +50;
b = uint8 (b);
figure (1)
imshow (uint8(a));
title ( ' Original Image ' )
figure (2)
imshow (uint8(b));
title ( ' Enhanced Image ' )
```

Output:





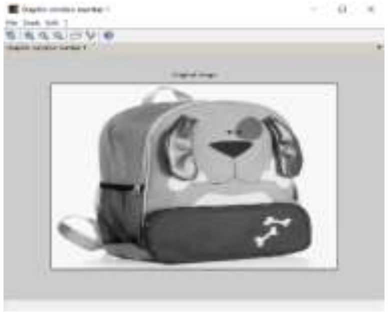
(B) Contrast Manipulation

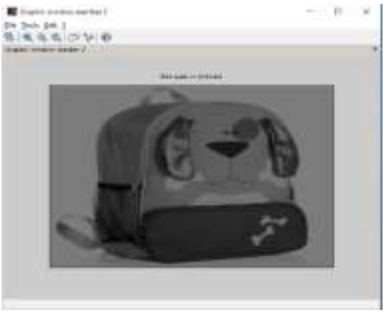
Code:

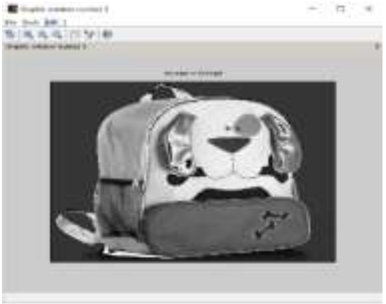
```
clc ;
close ;
a = imread ("C:\Users\sushil\Downloads\children_bag.jpg");
a = rgb2gray (a);
b = double (a) *0.5;
b = uint8 (b);
c = double (b) *2.5;
c = uint8 (c);
figure (1)
imshow(uint8(a));
```

```
title ( 'Original Image' );
figure (2)
imshow(b);
title ( 'Decrease in Contrast' );
figure (3)
imshow(c);
title ( 'Increase in Contrast' );
```

Output:





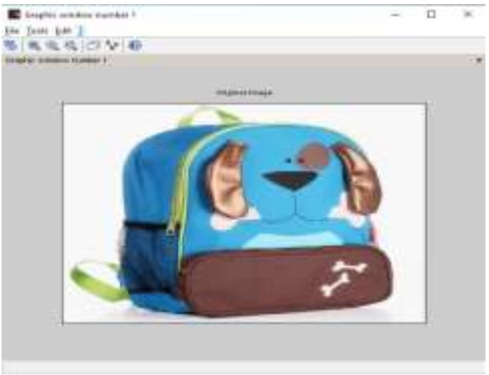


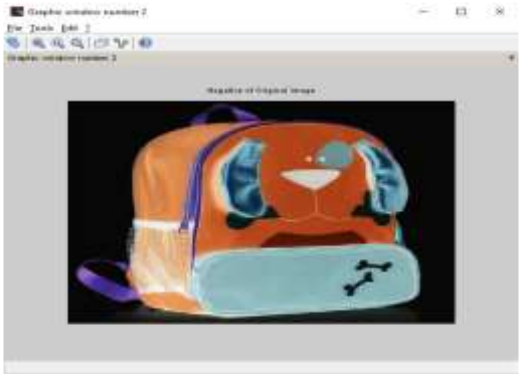
(C) Image Negative

Code:

```
clc ;
close ;
a = imread ("C:\Users\sushil\Downloads\children_bag.jpg");
k = 255 - double (a);
k = uint8 (k);
figure(1);
imshow (uint8(a));
title ( 'Original Image' );
figure(2);
imshow (k);
title ( 'Negative of Original Image' );
```

Output:





Practical – 7

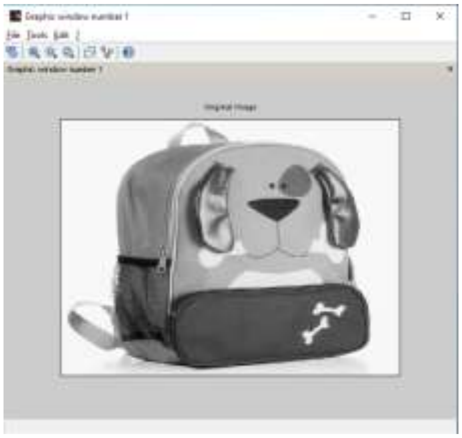
Aim: Perform threshold operation, perform gray level slicing without background.

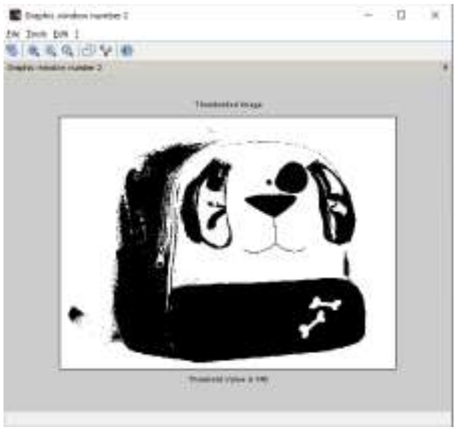
(A) Perform threshold operation

Code:

```
clc ;
a = imread ("C:\Users\sushil\Downloads\children_bag.jpg");
a = rgb2gray (a);
[m n] = size (a);
t = input("Enter the threshold parameter ");
for i = 1:m
    for j = 1:n
        if(a(i,j)<t)
            b(i,j)=0;
        else
            b(i,j) =255;
        end
    end
end
figure (1)
imshow(uint8(a));
title ( ' Original Image ' )
figure (2)
imshow(uint8(b));
title ( ' Thresholded Image ' )
xlabel ( sprintf ( 'Threshold Value is %g ' ,t))
```

Output:



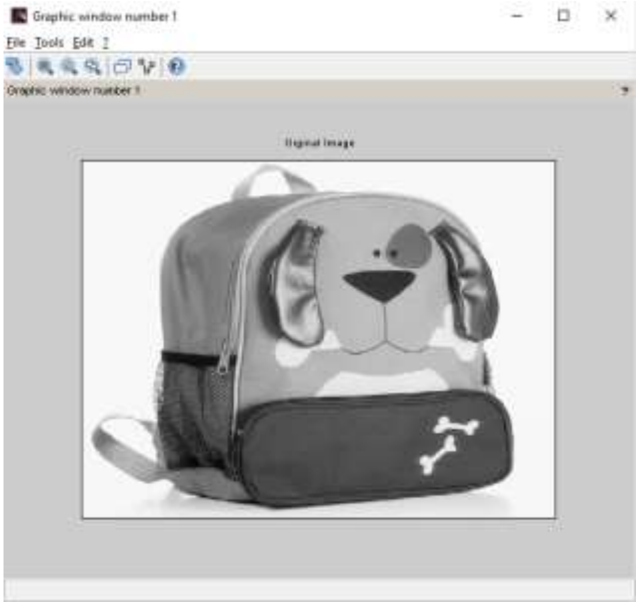


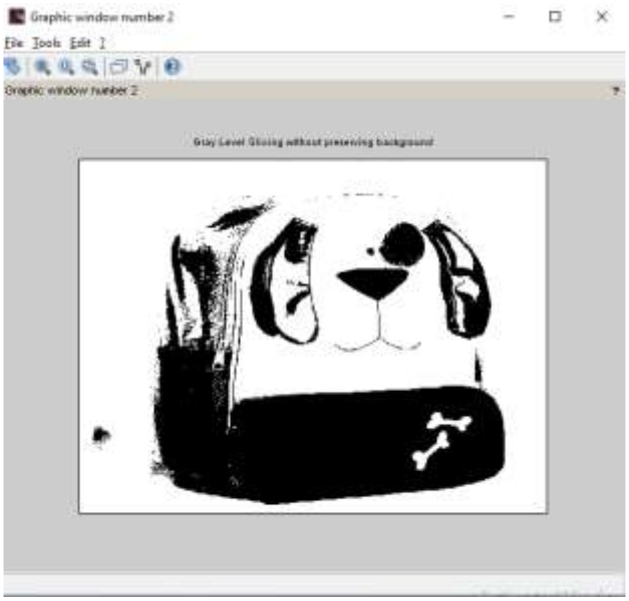
(B) Perform gray level slicing without background

Code:

```
clc ;
x = imread ("C:\Users\sushil\Downloads\children_bag.jpg");
x = rgb2gray(x);
y = double(x);
 [m n] = size(y);
L = max(x);
a = round(L/2);
b = L;
for i =1: m
    for j =1: n
        if(y(i,j) >=a & y(i,j) <=b)
            z(i,j) = L;
        else
            z(i,j)=0;
        end
    end
end
z = uint8 (z);
figure (1)
imshow(x)
title ( ' Orginal Image ' )
figure (2)
imshow(z);
title ( 'Gray Level Slicing without preserving background ' )
```

Output:





Practical – 8

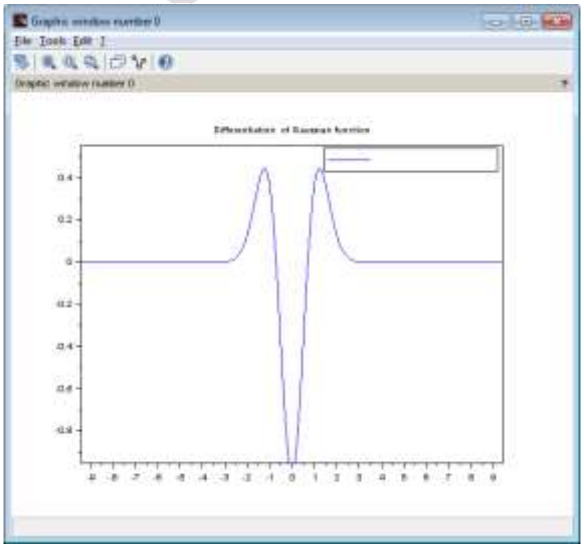
Aim: Image Segmentation.

Differentiation of Gaussian function

Code:

```
sigma=input(' Enter  the   value of sigma : ')
i= -10:.1:10;
j= -10:.1:10;
r=sqrt(i.*i+j.*j);
y=(1/( sigma ^2))*(((r.*r)/sigma ^2) -1).*exp(-r.*r/2*sigma ^2);
plot(i,y)
legend(sprintf(' The   sigma   value   is   %g ',sigma))
xtitle(' Differentiation  of Gaussian function ')
```

Output:

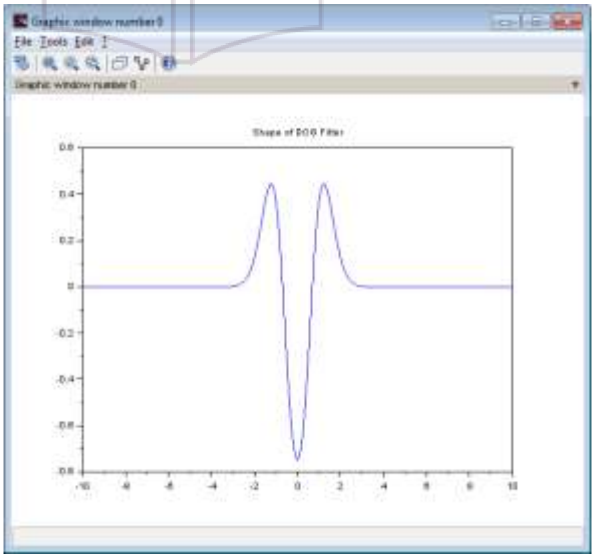


Shape of DOG Filter

Code:

```
sigma1 =input(' Enter the value of sigma1 : ')
sigma2 =input(' Enter the   value of sigma2 : ')
i= -10:.1:10;
j= -10:.1:10;
r=sqrt(i.*i+j.*j);
y1 = (1/( sigma1 ^2))*(((r.*r)/sigma1 ^2) -1).*exp(-r.*r/2* sigma1 ^2);
y2 = (1/( sigma2 ^2))*(((r.*r)/sigma2 ^2) -1).*exp(-r.*r/2* sigma2 ^2);
y = y1 -y2;
plot(i,y)
xtitle(' Shape of DOG Filter  ')
```

Output:



Edge Detection

Code:

```
img = imread("D:\\Picture1.png");

img=rgb2gray(img);

c=edge(img,'sobel',0.5)

d=edge(img,'prewitt')

e=edge(img,'canny')

f=edge(img,'log')

figure(1)

imshow(img)

title('Original Image')

figure(2)

imshow(c)

title('Sobel')

figure(3)

imshow(d)

title('Prewitt')

figure(4)

imshow(e)

title('Canny')

figure(5)

imshow(f)

title('LOG')
```

Practical – 9

Aim: Image Compression

Arithmetic coding

Code:

```
clc;
clear all;
n=input("Enter the no. of symbols : ");
for i = 1:n
    printf("\nEnter the probability(<=1) of symbol %d: ",i);//Input: Taking the probability of occurence
p(i)=input("");
end
printf("\nThe cdf of symbol 1: %.3f ",p(1));
c(1)=p(1);
for i = 2:n
    c(i)=p(i)+c(i-1);
    printf("\nThe cdf of symbol %d: ",i);
    printf("%.3f",c(i));
end
s=input("Enter the no. of symbols in sequence");/
printf("Enter the sequence ");
for j = 1:s
b(j)=input("");//Inserting the sequence
end
//Setting the lower and upper limit for 1st stage
if b(1) == 1 then
l(1)=0;
u(1)=c(b(1));
```

```
else
l(1)=c(b(1)-1);
u(1)=c(b(1));
end
//Calculating lower and upper limits for 2nd stage and ahead
for k = 2:s
if b(k) == 1 then
l(k)=l(k-1);
u(k)=l(k-1)+((u(k-1)-l(k-1))*c(b(k)));
else
l(k)=l(k-1)+((u(k-1)-l(k-1))*c(b(k)-1));
u(k)=l(k-1)+((u(k-1)-l(k-1))*c(b(k)));
end
end
tag=(l(s)+u(s))/2;//Generating tag
printf("The tag of the sequence is= %.10f",tag);//Output: The tag of the sequence
//Output for ex tag=0.1375781250
```

Output:

Note: for inputs refer the solved example of DIP book page 457

```
Enter the no. of symbols : 4


Enter the probability(<=1) of symbol 1:
--> 0.4


Enter the probability(<=1) of symbol 2:
--> 0.2


Enter the probability(<=1) of symbol 3:
```

```
--> 0.1


Enter the probability(<=1) of symbol 4:
--> 0.3


The cdf of symbol 1: 0.400
The cdf of symbol 2: 0.600
The cdf of symbol 3: 0.700
The cdf of symbol 4: 1.000
Enter the no. of symbols in sequence3

Enter the sequence
--> 4

--> 1

--> 4

The tag of the sequence is= 0.8020000000
```

Run length Coding

Code:

```
clc;

clear;

close;

in=input('Enter squares matrix::::');

[m,n]=size(in);

y=0;

tx(1)=0;

o=1

for j=1:m
```

```
    for k=1:n
        x=in(j,k);
        if x==y
         tx(o)=tx(o)+1;
        else
          o=o+1;
          tx(o)=1;
        end
        y=x;
    end
end
disp('code sucsess');
disp(tx);
```

Output:

```
Enter squares matrix::::[2 2 2;1 1 1;3 3 1]

 code sucsess

   3.
   3.
   2.
   1.
```

Practical – 10

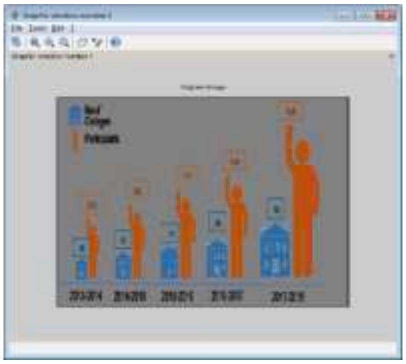
Aim: Binary Image Processing and Colour Image processing.

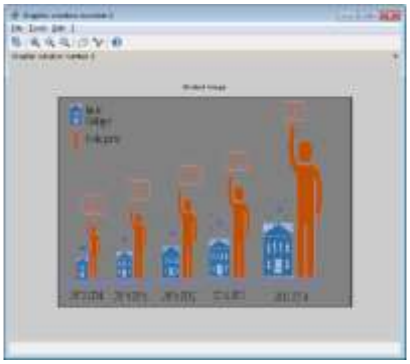
(A) Binary Image Processing-Dilation and Erosion

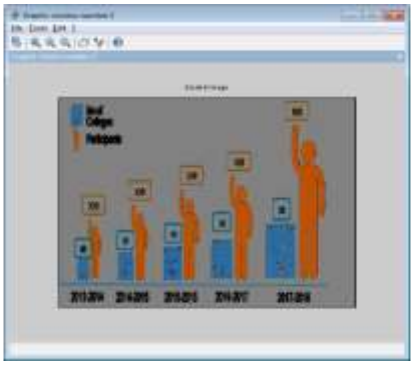
Code:

```
a=imread('D:\\Picture1.png');
//se=CreateStructureElement('square',3);
se=imcreatese('cross',3,3);
a1=imdilate(a,se);
a2=imerode(a,se);
figure(1)
imshow(a);
title('Original Image');
figure(2)
imshow(a1);
title('Dilated Image');
figure(3)
imshow(a2);
title('Eroded Image');
```

Output:





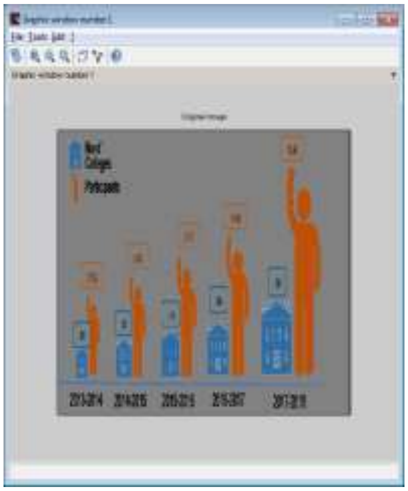


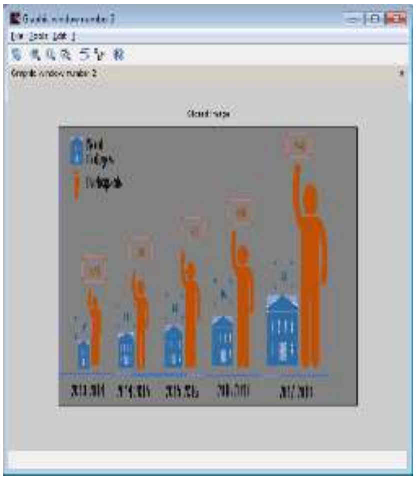
Binary Image Processing-Opening and closing

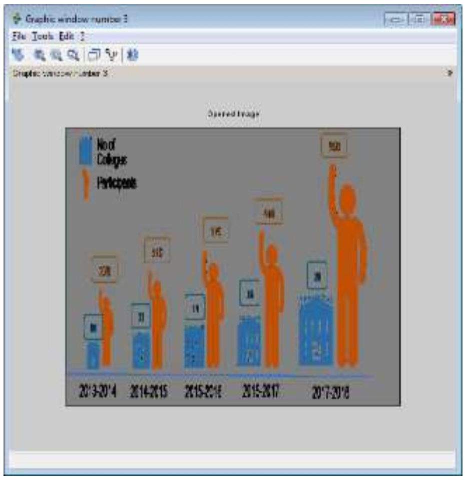
Code:

```
a=imread('D:\\Picture1.png');
//se=CreateStructureElement('square',3);
se=imcreatese('rect',3,3);

//Code for Closeing image
a1=imdilate(a,se);
a2=imerode(a1,se);

figure(1)
imshow(a);
title('Original Image');
figure(2)
imshow(a2);
title('Closed Image');

//Code for Opening image
a1=imerode(a,se);
a2=imdilate(a1,se);
figure(3)
imshow(a2);
title('Opened Image');
```

Output:







(B) Colour Image processing

Code:

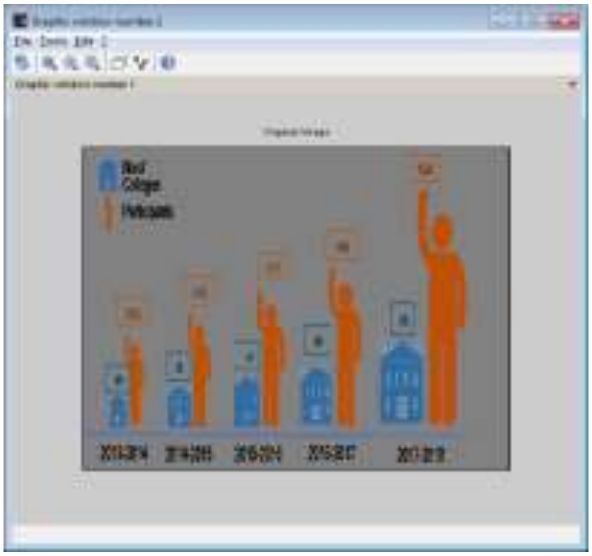
```
img=imread('D:\\Picture1.png');
histB=calcHist(img,0,[],1,32,[0 256]);
scf();
bar(histB(:),'blue');

histG=calcHist(img,1,[],1,32,[0 256]);
scf();
bar(histG(:),'blue');

histR=calcHist(img,2,[],1,32,[0 256]);
scf();
bar(histB(:),'red');
```

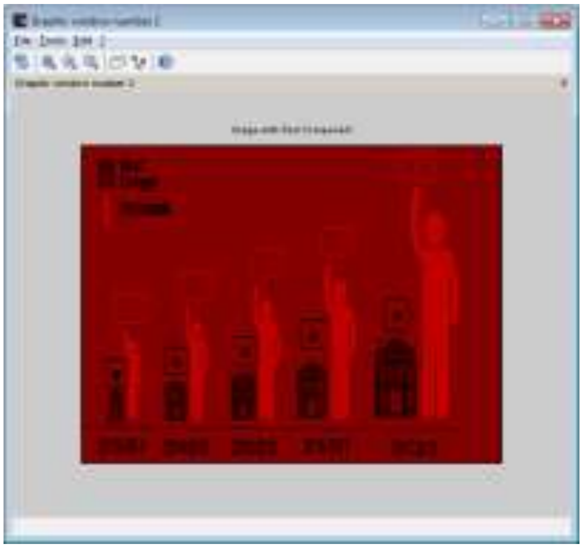
Output:

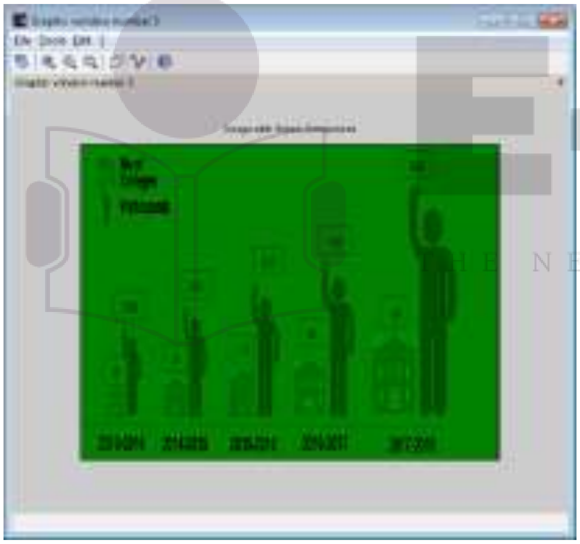
Image with RED Green Blue Component

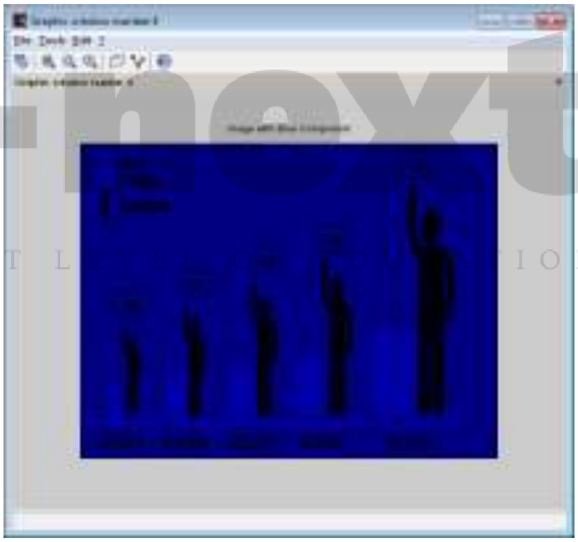
```
RGB=imread('D:\\Picture1.png');
R=RGB;
G=RGB;
B=RGB;
R(:,:,2)=0;
R(:,:,3)=0;
G(:,:,1)=0;
G(:,:,3)=0;
B(:,:,1)=0;
B(:,:,2)=0;
figure(1)
imshow(RGB)
title('Original Image')
figure(2)
imshow(R)
title('Image with Red Component')
figure(3)
imshow(G)
title('Image with Green Component')
figure(4)
imshow(B)
title('Image with Blue Component')
```

Output:









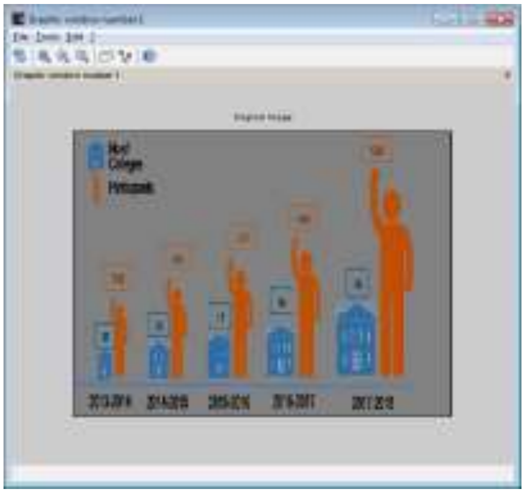
'Histogram equalised Image'

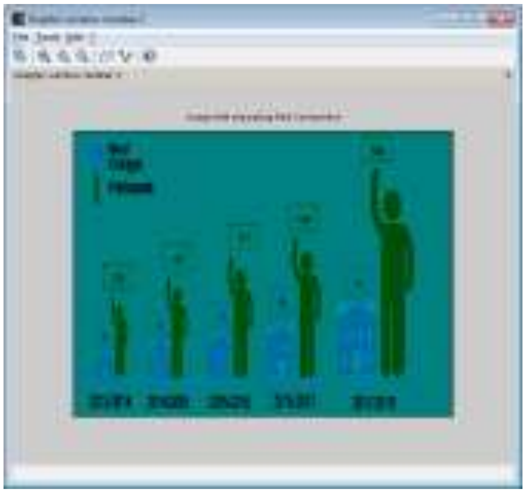
Code:

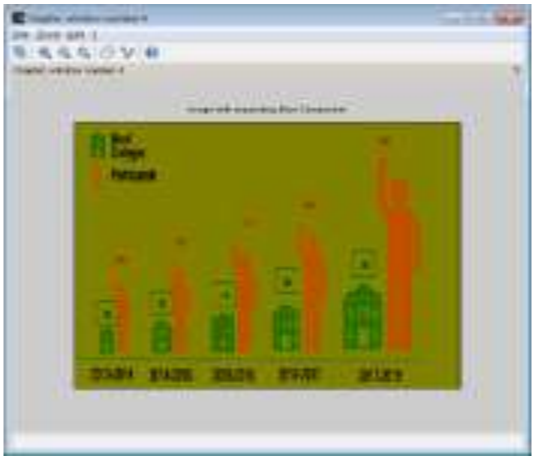
```
a=imread('D:\\Picture1.png');
b=rgb2ntsc(a);
b(:,:,1)=imhistequal(b(:,:,1));
c=ntsc2rgb(b);
figure(1)
imshow(a)
title('Original Image')
```

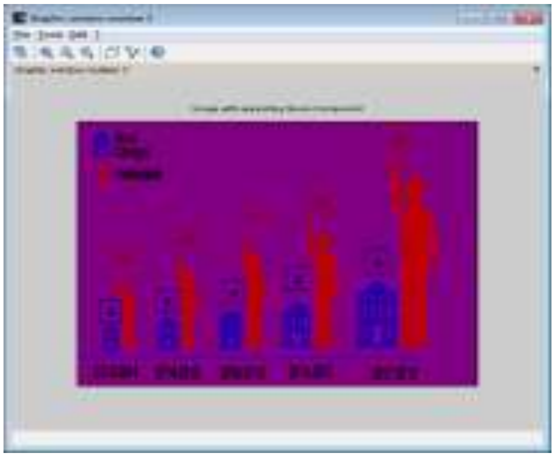
```
figure(2)
imshow(c)
title('Histogram equalised Image')


RGB=imread('D:\\Picture1.png');
a1=RGB;
a2=RGB;
a3=RGB;
a1(:,:,1)=0;
a2(:,:,2)=0;
a3(:,:,3)=0;
figure(1)
imshow(RGB)
title('Original Image')
figure(2)
imshow(a1)
title('Image with separating Red Component')
figure(3)
imshow(a2)
title('Image with separating Green Component')
figure(4)
imshow(a3)
title('Image with separating Blue Component')
```

Output:







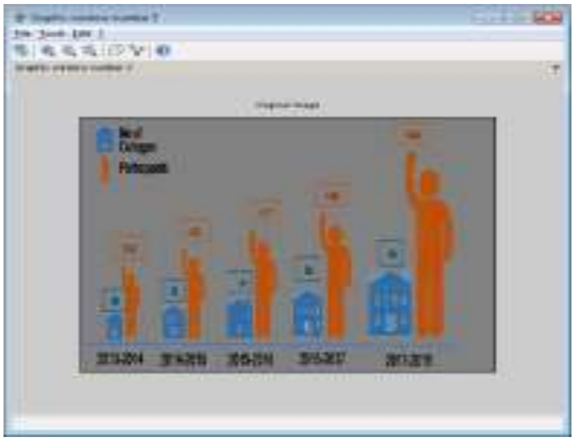


Histogram of gray image

Code:

```
a=imread('D:\\Picture1.png');
[count1, cells]=imhist(a(:,:));
figure(1);plot(count1)
b=imhistequal(a(:,:))
figure(2)
imshow(b)
figure(3)
imshow(a)
title('Original Image')
[count, cells]=imhist(b(:,:));
figure(4);plot(count)
```

Output:


File Zoom Edit ?