|  |  |  |  |
| --- | --- | --- | --- |
| **Index** | | | |
| **Sr.No.** | **Date** | **Title** | **Signature** |
| 1.a. |  | 2D Linear Convolution |  |
| 1.b. |  | Circular Convolution between two 2D matrices |
| 2 |  | Circular Convolution expressed as linear convolution plus alias |  |
| 3. a. |  | Linear Cross correlation of a 2D matrix |  |
| 3. b. |  | Circular correlation between two signals |
| 3. c. |  | Linear auto correlation of a 2D matrix |
| 4 |  | DFT of 4x4 gray scale image |  |
| 5. a. |  | Compute discrete cosine transform |  |
| 5. b. |  | Program to perform KL transform for the given 2D  matrix |
| 6. a. |  | Brightness enhancement of an image |  |
| 6. b. |  | Brightness suppression of an image |
| 6. c. |  | Contrast Manipulation |
| 6. d. |  | Image negative |
| 7. a. |  | Perform threshold operation |  |
| 7. b. |  | perform gray level slicing without background |
| 8 |  | **Image Segmentation** |  |
| 8. a. |  | Differentiation of Gaussian function |
| 8. b. |  | Shape Of DOG Filter |
| 9 |  | **Binary Image Processing** |  |
| 9. a. |  | Image Dilation |
| 9. b. |  | Image Erosion |
| 9. c. |  | Image Opening |
| 9. d. |  | Image Closing |
| 10 |  | **Color Image processing** |  |
| 10. a. |  | Histogram of Gray Scale Image |
| 10. b. |  | Histogram of Color Image |
| 10.c. |  | Histogram Equalization |
| 10.d. |  | Histogram equ. Without inbuilt function |
| 10. e |  | Read an RGB image and Extract the three color components red green blue |
| 10. f |  | Read a Color image and separate the color image into red green and blue planes |

# 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**

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

# Circular Convolution between two 2D matrices Code

"Linear 2D convolution y="

clc ;

x = [1,2;3,4];

h = [5,6;7,8];

X = fft2(x); H = fft2(h); Y = X.\*H;

y = ifft(Y);

disp (y, 'Circular Convolution Result y = ');

**Output**

"Circular Convolution Result y = " 70. 68.

62. 60.

# 2 Circular Convolution expressed as linear convolution plus alias

"Linear Convolution Result y ="

"Circular Convolution Expressed as Linear Convolution plus alias = " 70. 68.

62. 60.

**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**

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

# 3. 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 of a 3D matrix y = ");

**Output**

|  |  |  |
| --- | --- | --- |
| 9. | 9. | 2. |
| 21. | 24. | 9. |
| 10. | 22. | 4. |

# 3. b. Circular correlation between two signals

"Linear Cross Correlation of a 3D matrix y = "

**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.

# 3. 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.

# 4 DFT of 4x4 gray scale image

"display"

**Code**

clc;

f=[ 1 1 1 1;1 1 1 1;1 1 1 1;1 1 1 1];

t=fft2(f); disp(t,'display')

**Output**

|  |  |  |  |
| --- | --- | --- | --- |
| 16. | 0. | 0. | 0. |
| 0. | 0. | 0. | 0. |
| 0. | 0. | 0. | 0. |
| 0. | 0. | 0. | 0. |

# 5. a. Compute discrete cosine transform

"Kernel Matrix of 2d DFT="

1. + 0.i 1. + 0.i 1. + 0.i 1. + 0.i

0. - i 0. - i 0. - i 0. - i

-1. + 0.i -1. + 0.i -1. + 0.i -1. + 0.i

0. + i 0. + i 0. + i 0. + i

"2D DFT of given 2D image="

**Code**

clc; N=[1,2,3,4]

x=dct(N);

disp(x,'DCt matrix of order four')

**Output**

"DCt matrix of order four"

5. -2.2304425 0. -0.1585127

# b. Program to perform KL transform for the given 2D matrix

**Code**

clc; f=[1,1,1,1;1,1,1,1;1,1,1,1;1,1,1,1];

N=4

for k=0:N-1 for l=0:N-1

w(k+1,l+1)=int(cos((2\*%pi\*k\*1)/N)-%i\*sin((2\*%pi\*k\*1)/N)) end

end

disp(w,'Kernel Matrix of 2d DFT=')

F=w\*f\*w

disp(F,'2D DFT of given 2D image=')

**Output**

|  |  |  |  |
| --- | --- | --- | --- |
| 0. + 0.i | 0. + 0.i | 0. + 0.i | 0. + 0.i |
| 0. + 0.i | 0. + 0.i | 0. + 0.i | 0. + 0.i |
| 0. + 0.i | 0. + 0.i | 0. + 0.i | 0. + 0.i |
| 0. + 0.i | 0. + 0.i | 0. + 0.i | 0. + 0.i |

# a. Brightness enhancement of an image

**Code**

clc; close;

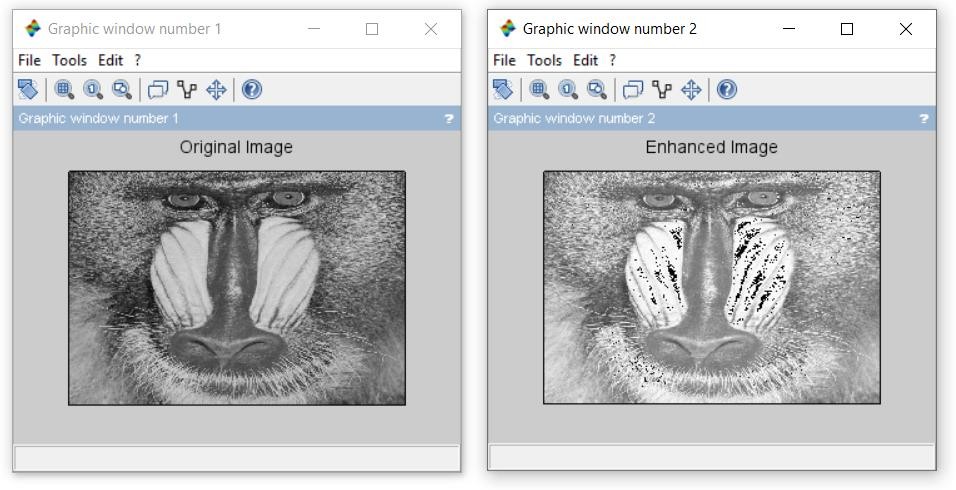
a=imread((getIPCVpath() +"/images/"+'baboon.png')) a=rgb2gray(a);

b=double(a)+50; b=uint8(b); figure(1) imshow(a);

title('Original Image') figure(2)

imshow(b); title('Enhanced Image')

**Output**



# 6. b. Brightness suppression of an image

**Code**

clc; close;

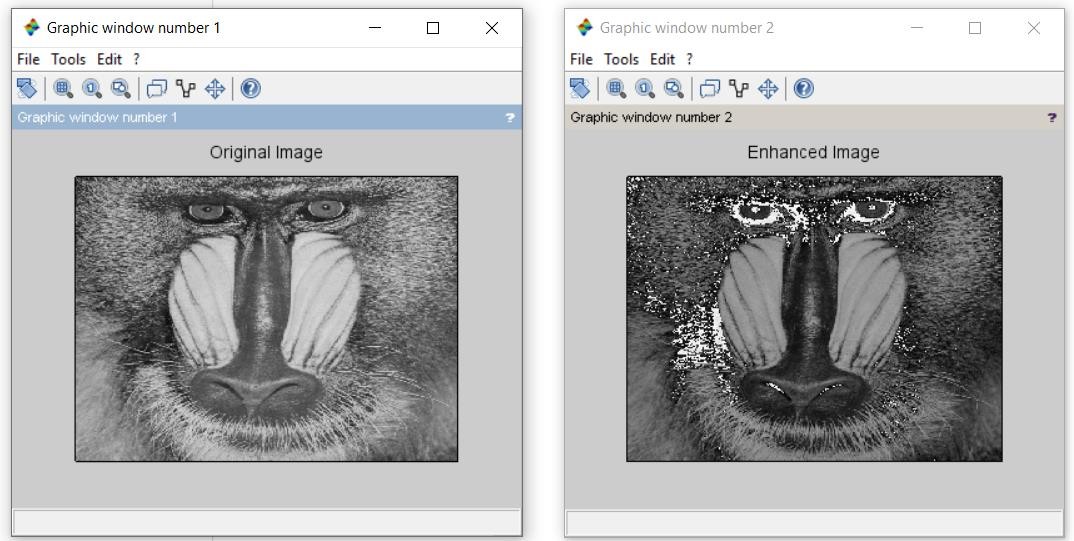
a=imread((getIPCVpath() +"/images/"+'baboon.png')) a=rgb2gray(a);

b=double(a)-50; b=uint8(b); figure(1) imshow(a);

title('Original Image') figure(2)

imshow(b); title('Enhanced Image')

**Output**



# 6. c. Contrast Manipulation

**Code**

clc; close;

a=imread((getIPCVpath() +"/images/"+'baboon.png')) a=rgb2gray(a);

b=double(a)\*0.5; b=uint8(b); c=double(b)\*2; c=uint8(c); figure(1) imshow(a);

title('Original Image') figure(2)

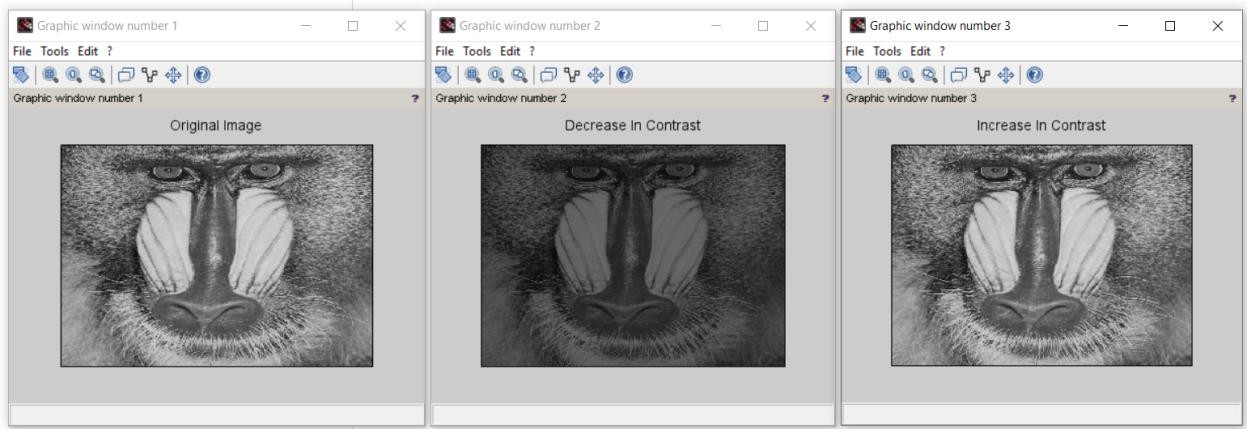
imshow(b);

title('Decrease In Contrast') figure(3)

imshow(c);

title('Increase In Contrast')

**Output**



# d. Image negative

**Code**

clc; close;

a=imread(fullpath(getIPCVpath() +"/images/"+'baboon.png')); a=rgb2gray(a);

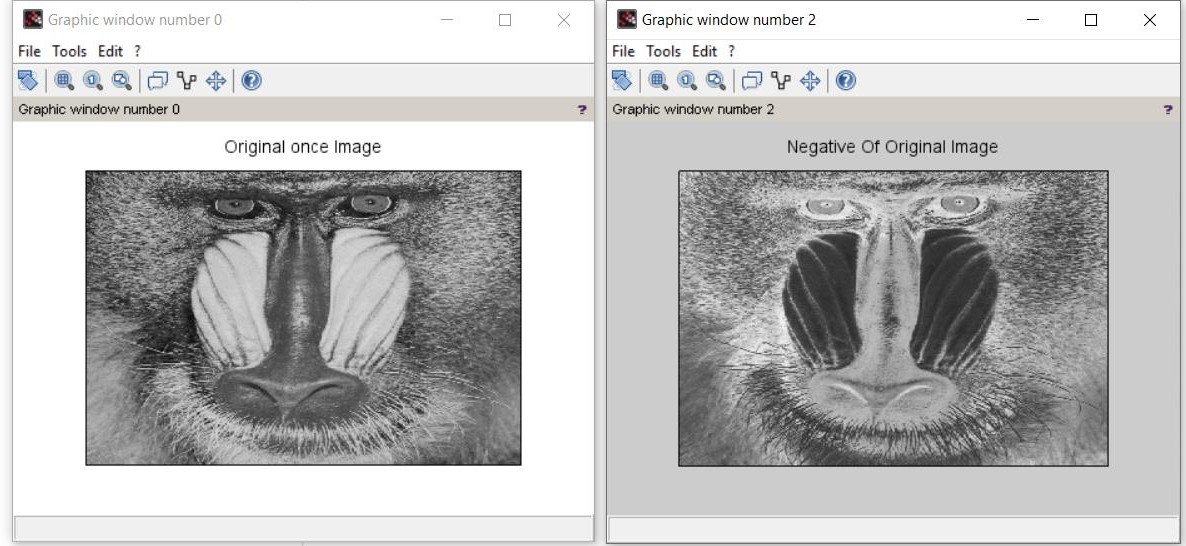
k=255-double(a); k=uint8(k); imshow(a);

title('Original once Image') figure(2)

imshow(k);

title('Negative Of Original Image')

**Output**



**Image Negative of color image Code**

clc; close;

a=imread(fullpath(getIPCVpath() +"/images/"+'baboon.png')); k=255-double(a);

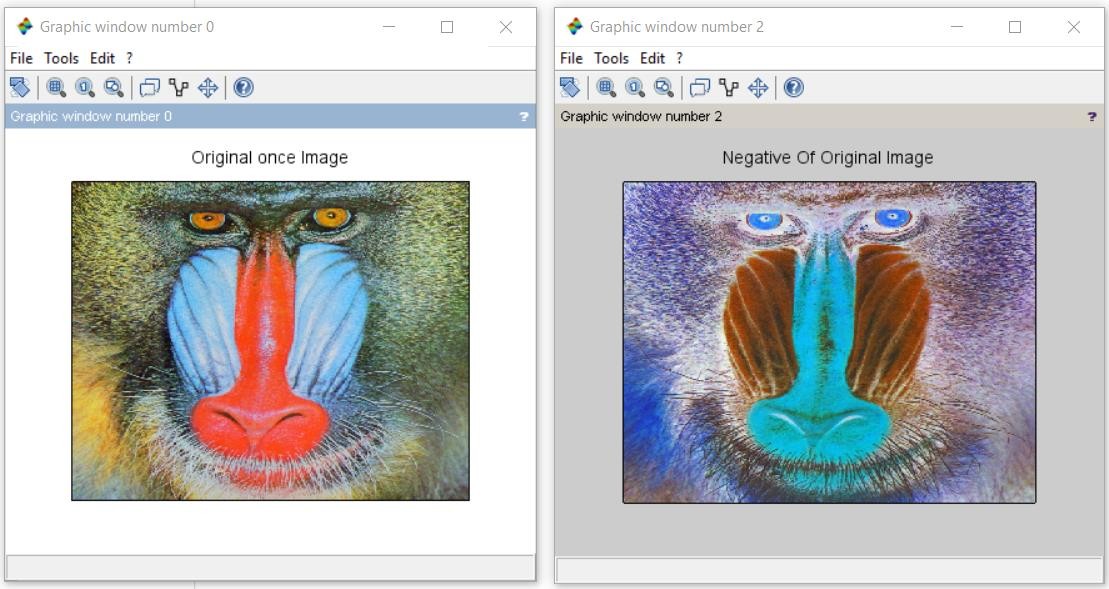
k=uint8(k); imshow(a);

title('Original once Image') figure(2)

imshow(k);

title('Negative Of Original Image')

**Output**



# a. Perform threshold operation

**Code**

clc; close;

a=imread((getIPCVpath() +"/images/"+'baboon.png')) a=rgb2gray(a);

[ m n ] = size (a); for i =1 :m

for j = 1 : n if(a(i,j)< 100)

b(i,j)=0; else

b(i,j)=255;

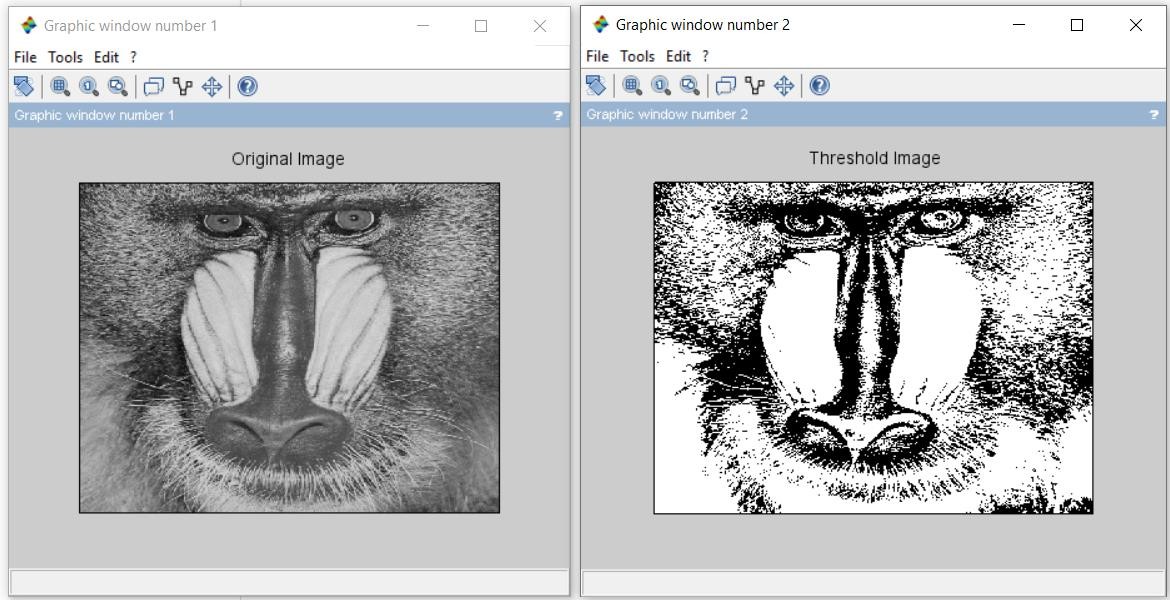
end end

end figure(1) imshow(a);

title('Original Image'); figure(2)

imshow(b); title('Threshold Image');

**Output**



# b. perform gray level slicing without background

**Code**

clc;

x=imread(fullpath(getIPCVpath() +"/images/"+'baboon.png')); x=rgb2gray(x);

y=double(x); [m,n] =size(y);

L=max(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); for i = 1:m for j = 1:n

if(y(i,j)>= a & y(i,j)<=b ) w (i,j)=L;

else w(i,j)=y(i,j); end

end end

z=uint8(z); w=uint8(w); figure(1) imshow(x);

title('Original Image') figure(2)

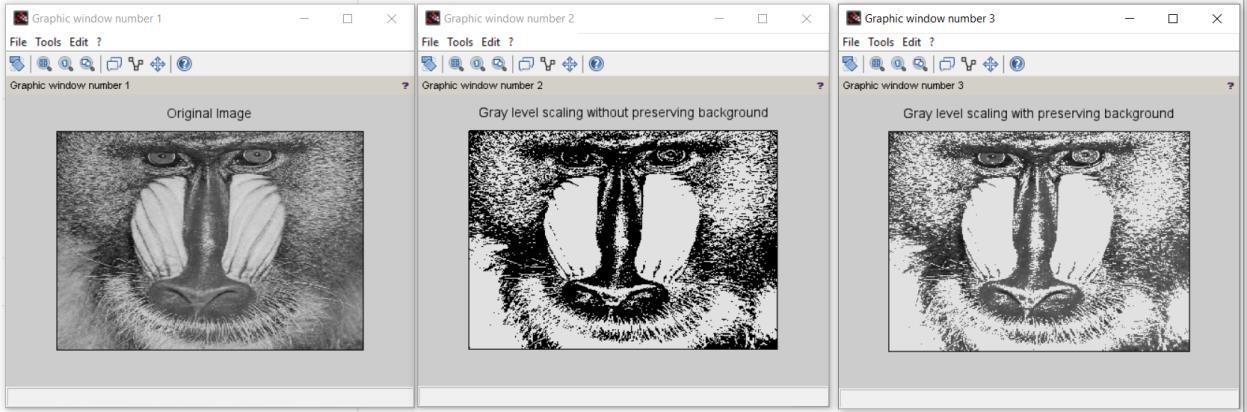
imshow(z);

title(' Gray level scaling without preserving background') figure(3)

imshow(w);

title(' Gray level scaling with preserving background')

**Output**



# Image Segmentation

**8. a. Differentiation of Gaussian function**

**Code**

clc; close;

sigma=input('Enter the value of sigma:'); i=-10:.1:10;

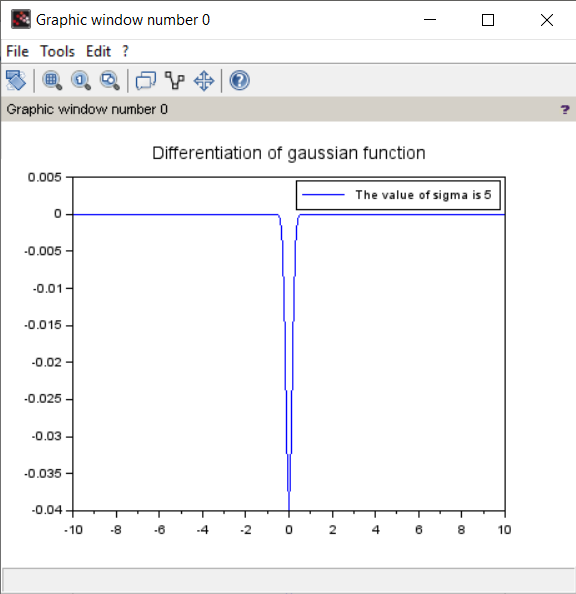
j=-10:.1:10;

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 value of sigma is %g',sigma)) xtitle(' Differentiation of gaussian function ')

**Output**



# b. Shape Of DOG Filter

**Code**

clc; close;

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);

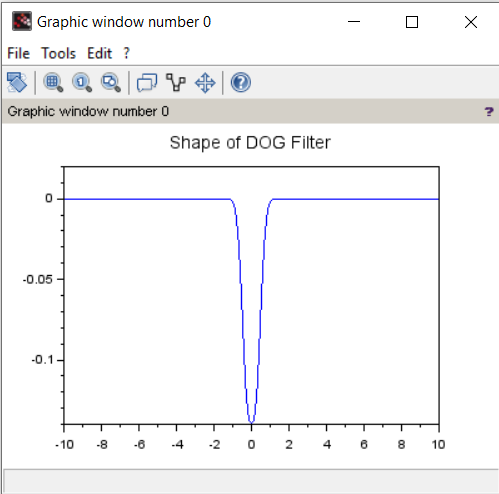
y1 = (1/( sigma1 ^2))\*(((r.\*r)/sigma1 ^2) -1).\*exp(-r.\*r/2\* sigma1 ^2); y2 = (1/( sigma2 ^2))\*(((r.\*r)/sigma2 ^2) -1).\*exp(-r.\*r/2\* sigma2 ^2); y = y1 -y2;

plot(i,y)

xtitle(' Shape of DOG Filter ')

**Output**

**Enter the value of sigma1 : 2 Enter the value of sigma2 : 3**



# Binary Image Processing

**9. a. Image Dilation**

**Code**

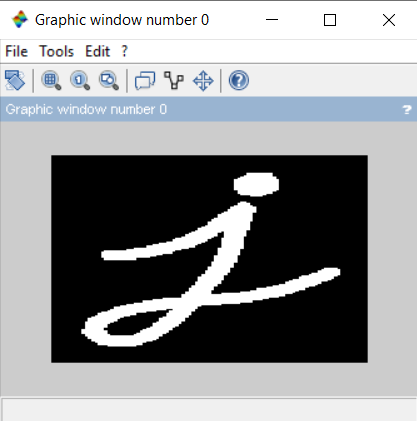
S=imread(fullpath(getIPCVpath() +"/images/"+'morpex.png')) figure(0);

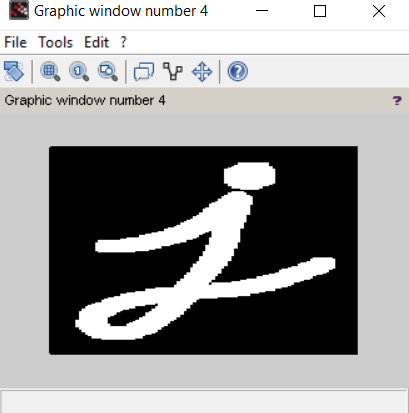
imshow(S);

se = imcreatese('ellipse',9,9); c= imdilate(S,se);

figure(1); imshow(c);

**Output Original Image**



**Dilated Image**

# 9. b. Image Erosion

**Code**

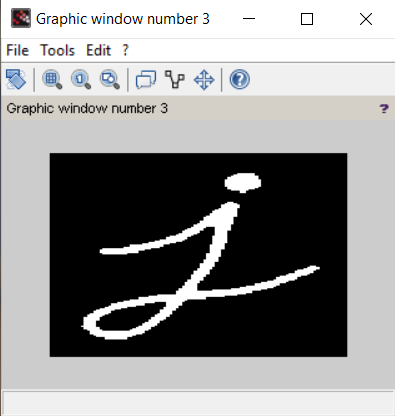
S=imread(fullpath(getIPCVpath() +"/images/"+'morpex.png')) figure(0);

imshow(S);

se= imcreatese('rect',3,3); b = imerode(S,se); figure(1);

imshow(b);

**Output Eroded Image**



# 9. c. Image Opening

**Code**

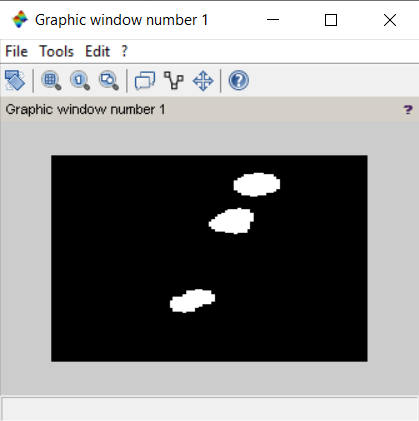
S=imread(fullpath(getIPCVpath() +"/images/"+'morpex.png')) figure(0);

imshow(S);

se = imcreatese('ellipse',9,9); S2 = imopen(S,se); figure(1);

imshow(S2);

**Output**



# d. Image Closing

**Code**

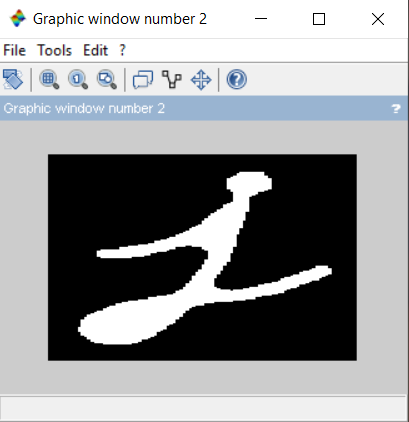
S=imread(fullpath(getIPCVpath() +"/images/"+'morpex.png')) figure(0);

imshow(S);

se= imcreatese('ellipse',11,11); S3 = imclose(S,se);

figure(2); imshow(S3);

**Output**



# Color Image processing

**10. a. Histogram of Gray Scale Image**

**Code**

a=imread(fullpath(getIPCVpath() +"/images/"+'Lena\_dark.png')); a=double(a);

big=max(max(a)); [row col] = size(a); c=row\*col; h=zeros(1,300); z=zeros(1,300); for n= 1:1:row

for m= 1:1:col

if a (n,m)==0 then a(n,m)=1;

end end end

for n=1:1:row for m=1:1:col

t=a(n,m);

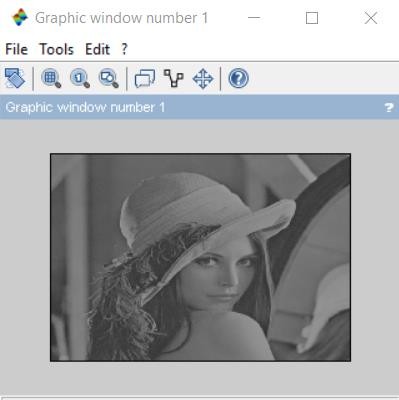
h(t)=h(t)+1;

end end

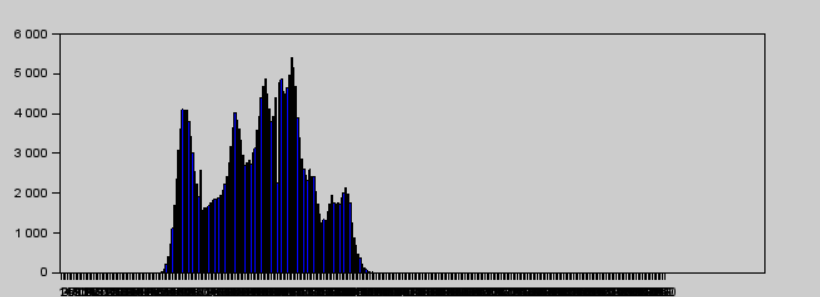
figure(1); imshow(uint8(a)); figure(2);

bar(h);

**Original Grayscale Image**



**Original Grayscale Image Histogram**



# b. Histogram of Color Image

**Code**

a=imread(fullpath(getIPCVpath() +"/images/"+'baboon.png')); a=double(a);

big=max(max(a)); [row col] = size(a); c=row\*col; h=zeros(1,300); z=zeros(1,300); for n= 1:1:row

for m= 1:1:col

if a (n,m)==0 then a(n,m)=1;

end end end

for n=1:1:row for m=1:1:col

t=a(n,m);

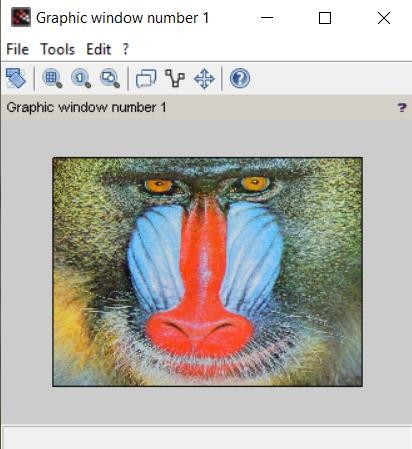
h(t)=h(t)+1;

end end

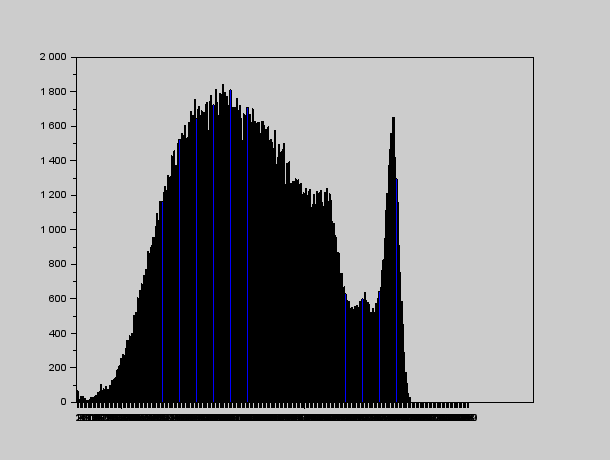
figure(1); imshow(uint8(a)); figure(2);

bar(h);

**Original Color Image**



**Color Image Histogram**



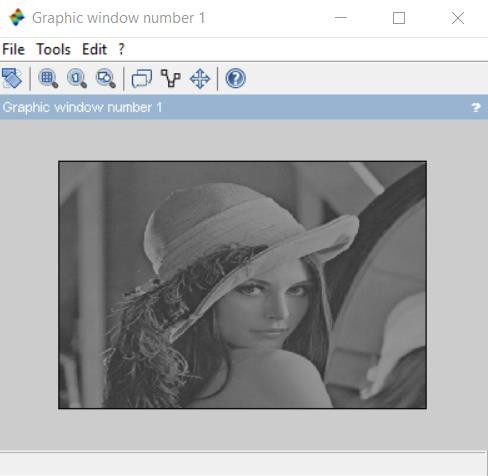
# Histogram Equalization with imhistequal function

**Code**

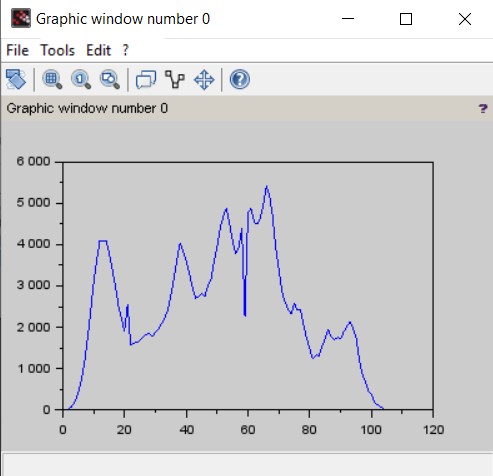
S=imread(fullpath(getIPCVpath() +"/images/"+'Lena\_dark.png')); J = imhistequal(S);

figure(1); imshow(S); figure(2); imshow(J);

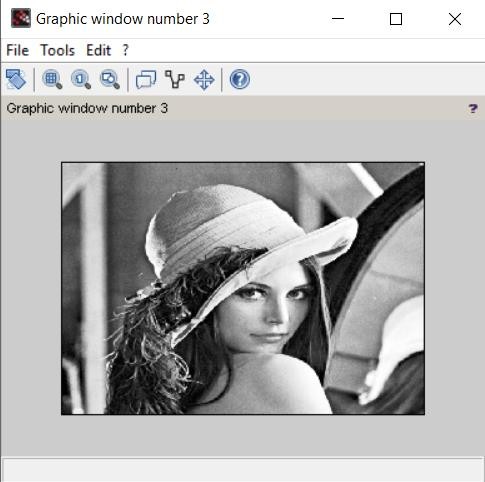
**Original Image (Lena\_dark)**



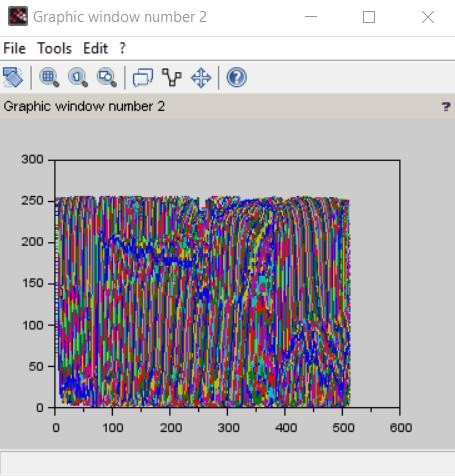
**Original Image Histogram (Lena\_dark)**



**Image Enhancement After Procedure Of Histogram Equalization (Lena\_dark)**



**Histogram After Procedure Of Histogram Equalization (Lena\_dark)**



# Histogram Equalization Without Inbuilt Function

Equalization is a process that attempts to spread out the gray levels in an image so that they are evenly distributed across their range. Histogram Equalization reassigns the brightness values of pixels based on the image histogram. Histogram Equalization is a technique where the histogram of the resultant image is as flat as possible. Histogram Equalization provides more visually pleasing results across a wider range of images.

**Procedure to Perform Histogram Equalization Histogram:**

Equalization is done by performing the following steps:

1. Find the running sum of the histogram values.
2. Normalize the values from Step (1) by dividing by the total number of pixels.
3. Multiply the values from Step (2) by the maximum gray-level value and round.
4. Map the gray level values to the results from Step (3) using a one-to-one correspondence

**Scilab Code:**

a=imread('tire.tif'); *//read image*

figure(0);

imshow(a); *// display image* histogram =histc(a); plot(histogram);

a=double(a); *//converts inttype integers or booleans into decimal encoding*

big=max(max(a)); *// maximum*

[r c] = size(a); *//matrix size*

tot = r\*c *//multiply rows and columns* h=zeros(1,256); *//store histogram values* z=zeros(1,256);

for i=1:1:r for j=1:1:c

if a(i,j)==0 a(i,j)=1;

end end

end

for i=1:1:r for j=1:1:c

t=a(i,j);

h(t)=h(t)+1; end

end

pdf=h/tot; *//probability distribution function*

cdf(1)=pdf(1); *//cumulative distribution function i.e.,running sum*

for i=2:1:big

cdf(i)=pdf(i)+cdf(i-1); end

new=round(cdf\*big); *//rounds the elements of x to the nearest integers.*

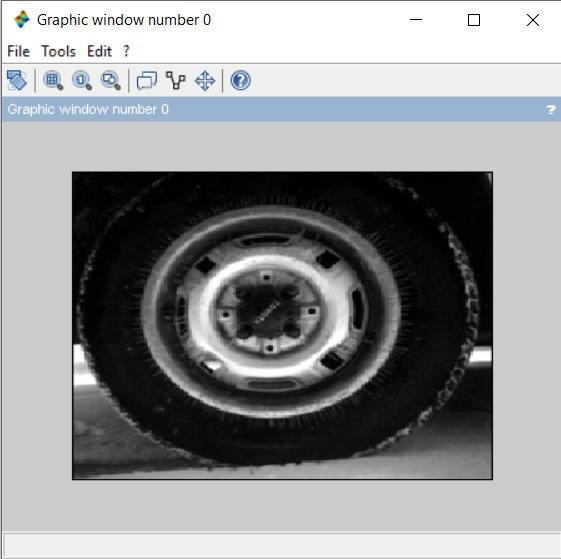
new=new+1; for i=1:1:r

for j=1:1:c temp=a(i,j); b(i,j)=new(temp); t=b(i,j);

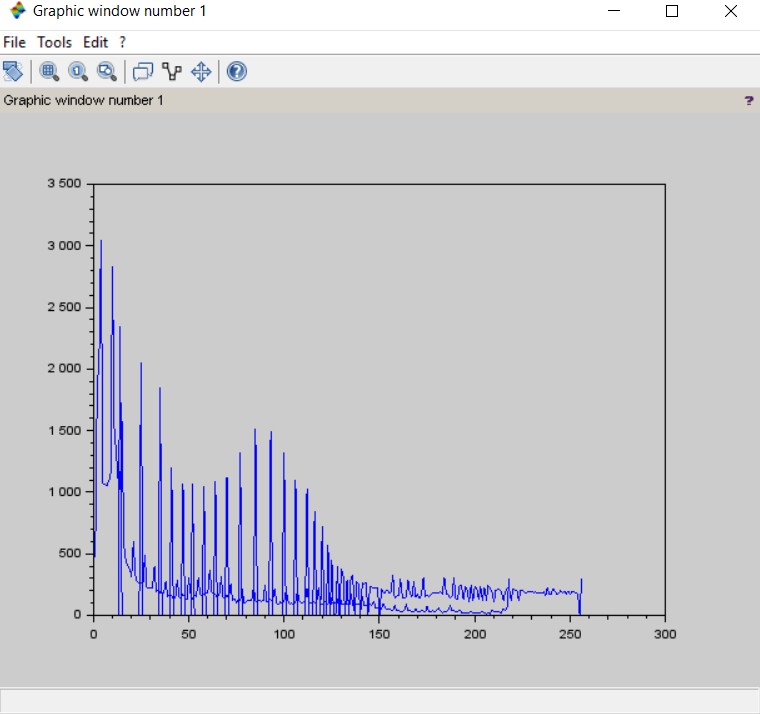
z(t)=z(t)+1; end

end subplot(2,1,2); plot(z); figure(1); imshow(b);

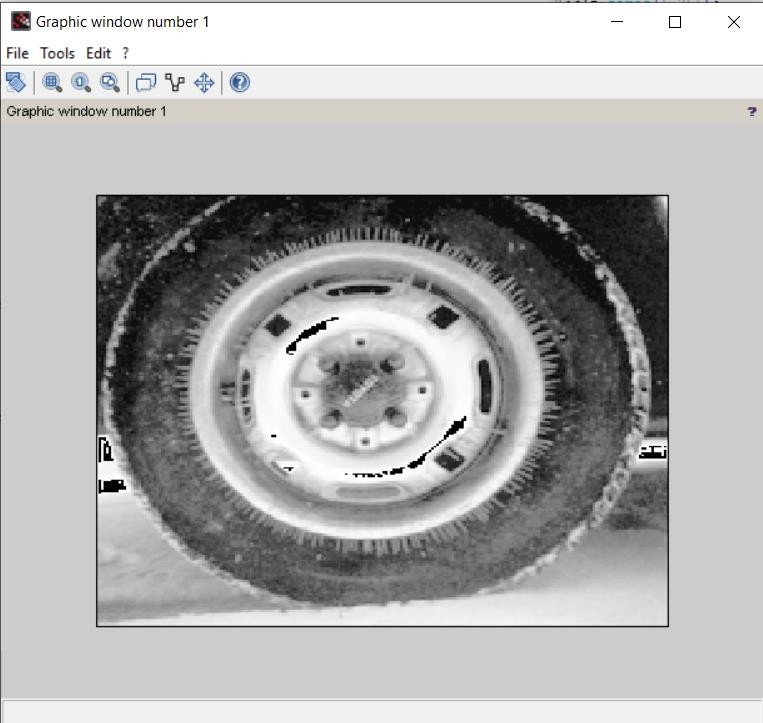
**Original Image (Tire)**



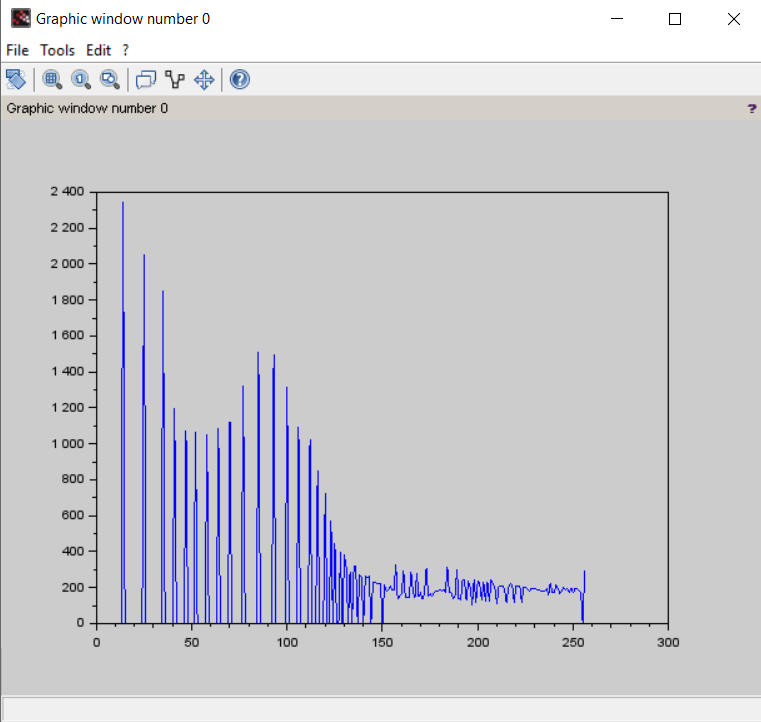
**Original Image Histogram (Tire)**



**Image Enhancement After Procedure Of Histogram Equalization (Tire)**



**Histogram After Procedure Of Histogram Equalization (Tire)**



# 10. e Read an RGB image and Extract the three color components red green blue

**Code**

clc; close;

RGB=imread((getIPCVpath() +"/images/"+'baboon.png')); R=RGB;

G=RGB; B=RGB; R(:,:,2)=0;

R(:,:,3)=0;

G(:,:,1)=0;

G(:,:,3)=0;

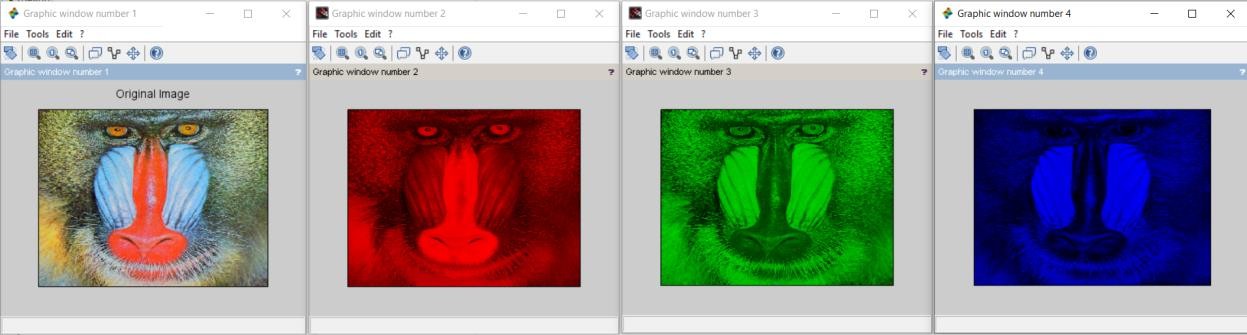
B(:,:,1)=0;

B(:,:,2)=0;

figure(1) imshow(RGB); title('Original Image') figure(2)

imshow(R); figure(3) imshow(G); figure(4) imshow(B);

**Output**



# 10. f Read a Color image and separate the color image into red green and blue planes

**Code**

clc; close;

RGB=imread((getIPCVpath() +"/images/"+'baboon.png')); a1=RGB;

b1=RGB; c1=RGB; a1(:,:,1)=0;

b1(:,:,2)=0;

c1(:,:,3)=0;

figure(1) imshow(RGB); figure(2) imshow(a1); figure(3) imshow(b1); figure(4) imshow(c1);

**Output**

