UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF ENGINEERING, SCHOOL OF COMPUTING

SKUDAI, 81310 JOHOR BAHRU, JOHOR DARUL TAKZIM

SEMESTER I SESSION 2020/2021

SCSV3213 - 01

FUNDAMENTAL OF IMAGE PROCESSING

LAB TUTORIAL

LECTURER:

DR. MD SAH HJ SALAM

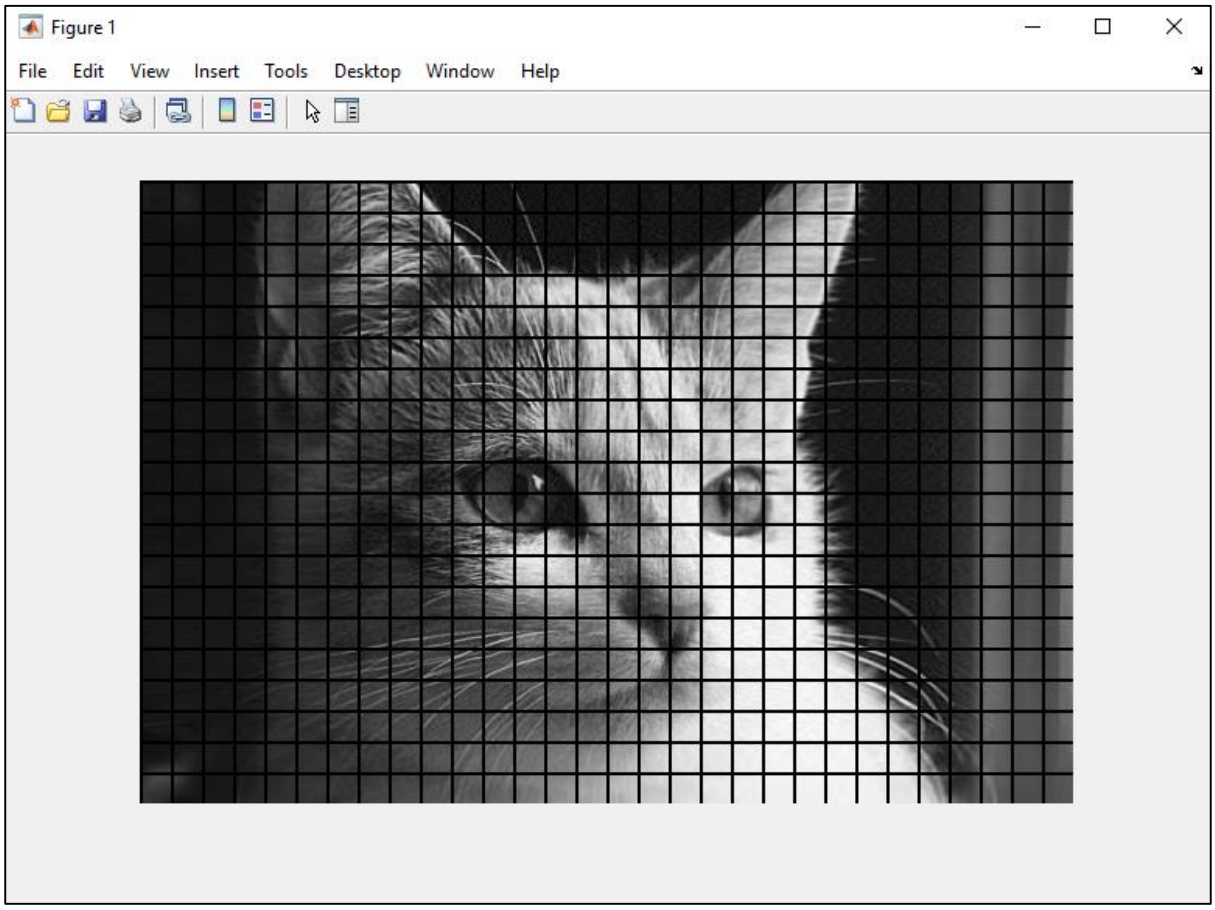
PREPARED BY:

Nuramyra Natasha Binti Ismalludin (B19EC0035)

LAB 1 – Test Your Understanding

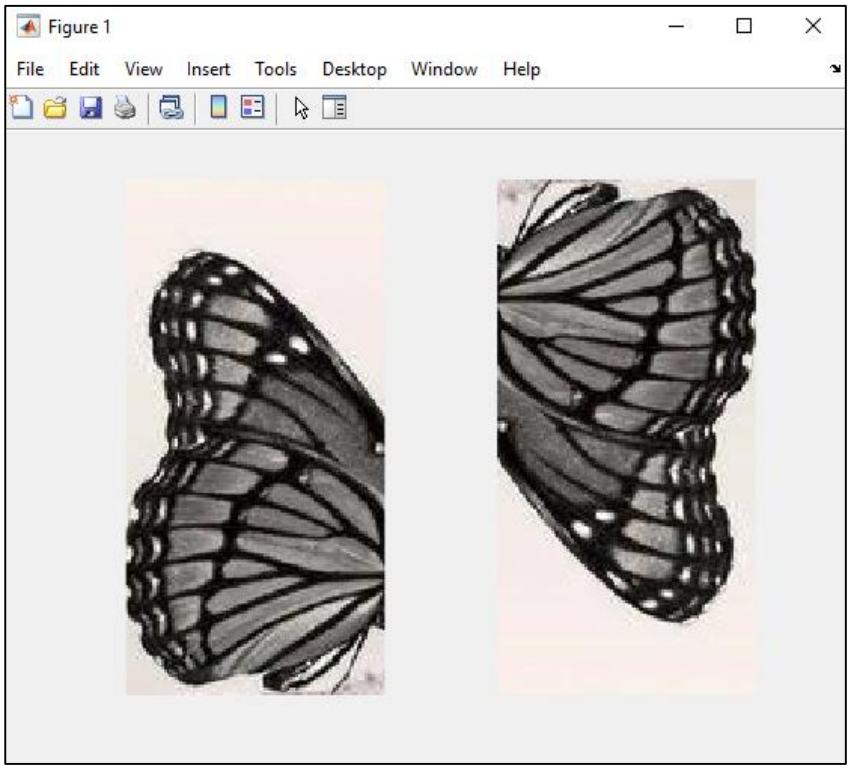
Tutorial 1	Grayscale Image Behind cage
Tutorial 2	Colour Image Rotation

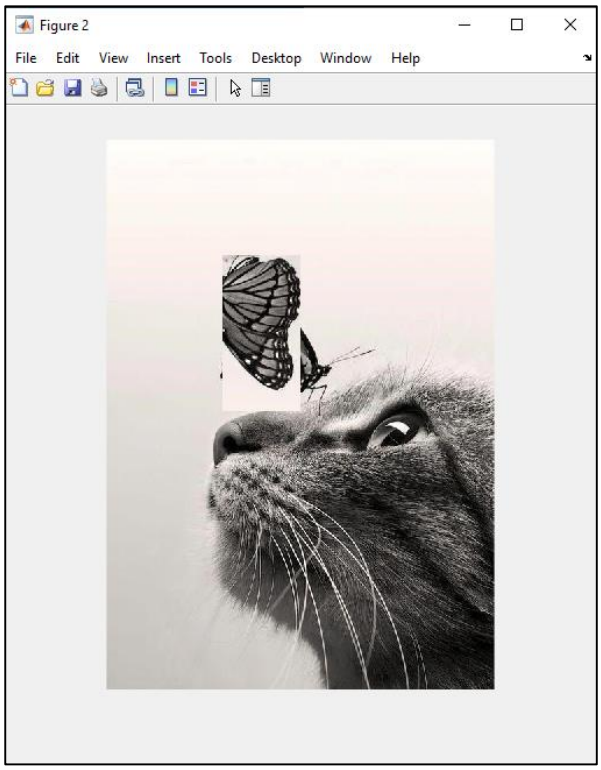
TUTORIAL 1 : Behind Cage




```
I = imread('meow.png');
[x,y] = size(I);
A=I;
for n=1:20:y
    A(:,n:n+1)=0;

end

for n=1:20:x
    A(n:n+1,:)=0;

end
imshow(A)
```

TUTORIAL 2: Rotate Image





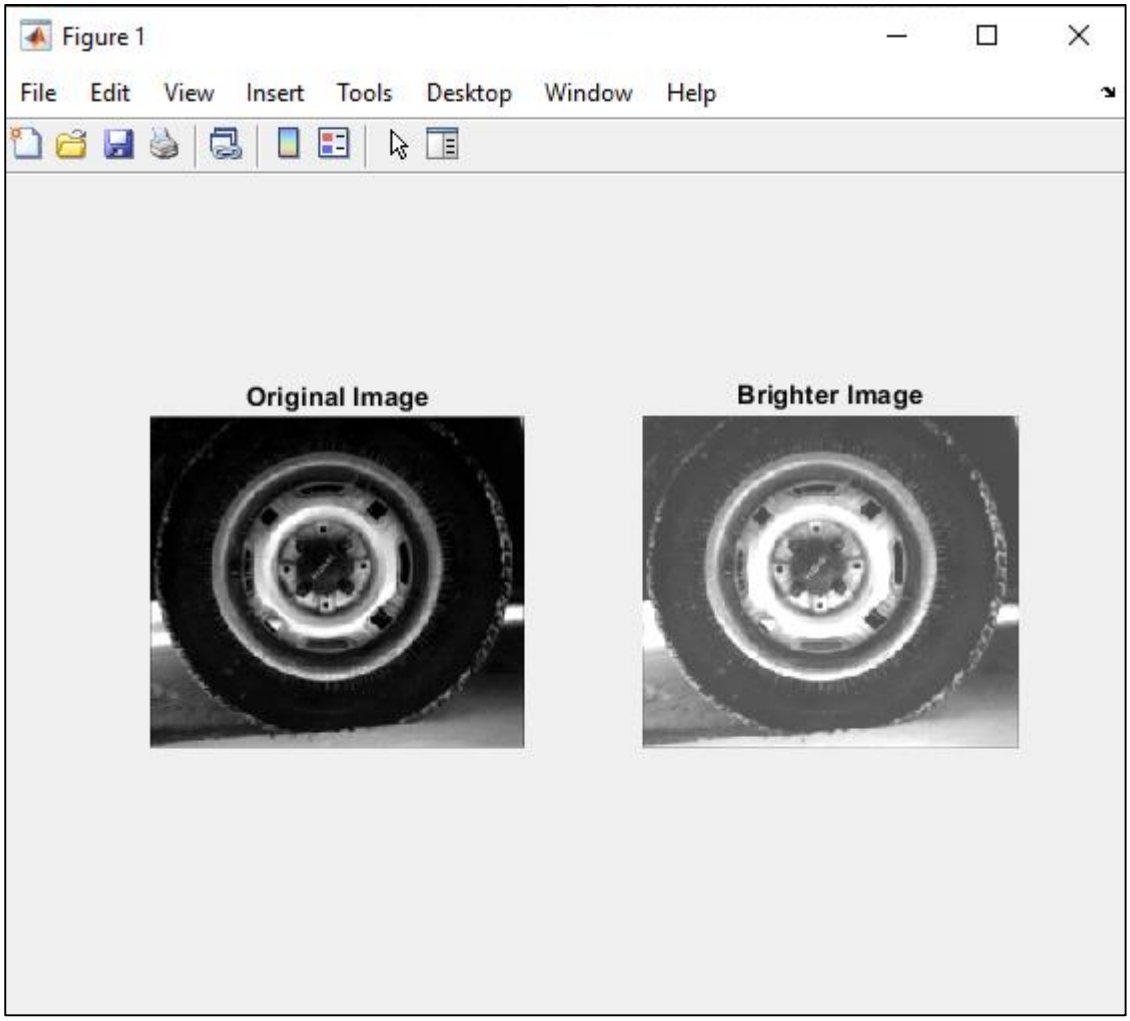

```
Z=imread('catt.jpg');
subplot(1,2,1), imshow(Z);
C= Z(150:350,150:250);
C=Z(150:350,150:250,:);
imshow(C);
C=imrotate(C,180);
subplot(1,2,2), imshow(C);
Z(150:350,150:250,:)=C;
figure, imshow(Z);
```

LAB 2 – TUTORIAL ON ARITHMETIC AND LOGIC OPERATION

Tutorial 1	Addition
Tutorial 2	Subtraction
Tutorial 3	Multiply
Tutorial 4	Multiply and Division
Tutorial 5	Logic operation
Test your understanding	

TUTORIAL 1

a. Brighten an image using imadd()

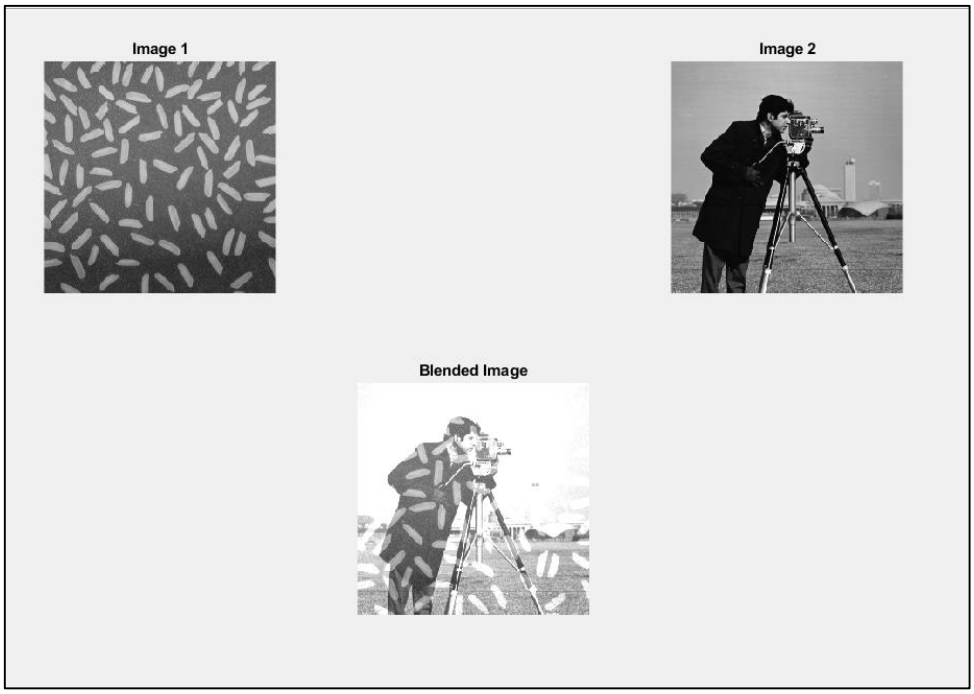



```
I = imread('tire.tif');
I2 = imadd(I,75);
figure
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I2), title('Brighter Image');
```

Explanation:

imadd(); function as to add the number of 75 into the image to increase the intensity of the image.

b. Blend two images



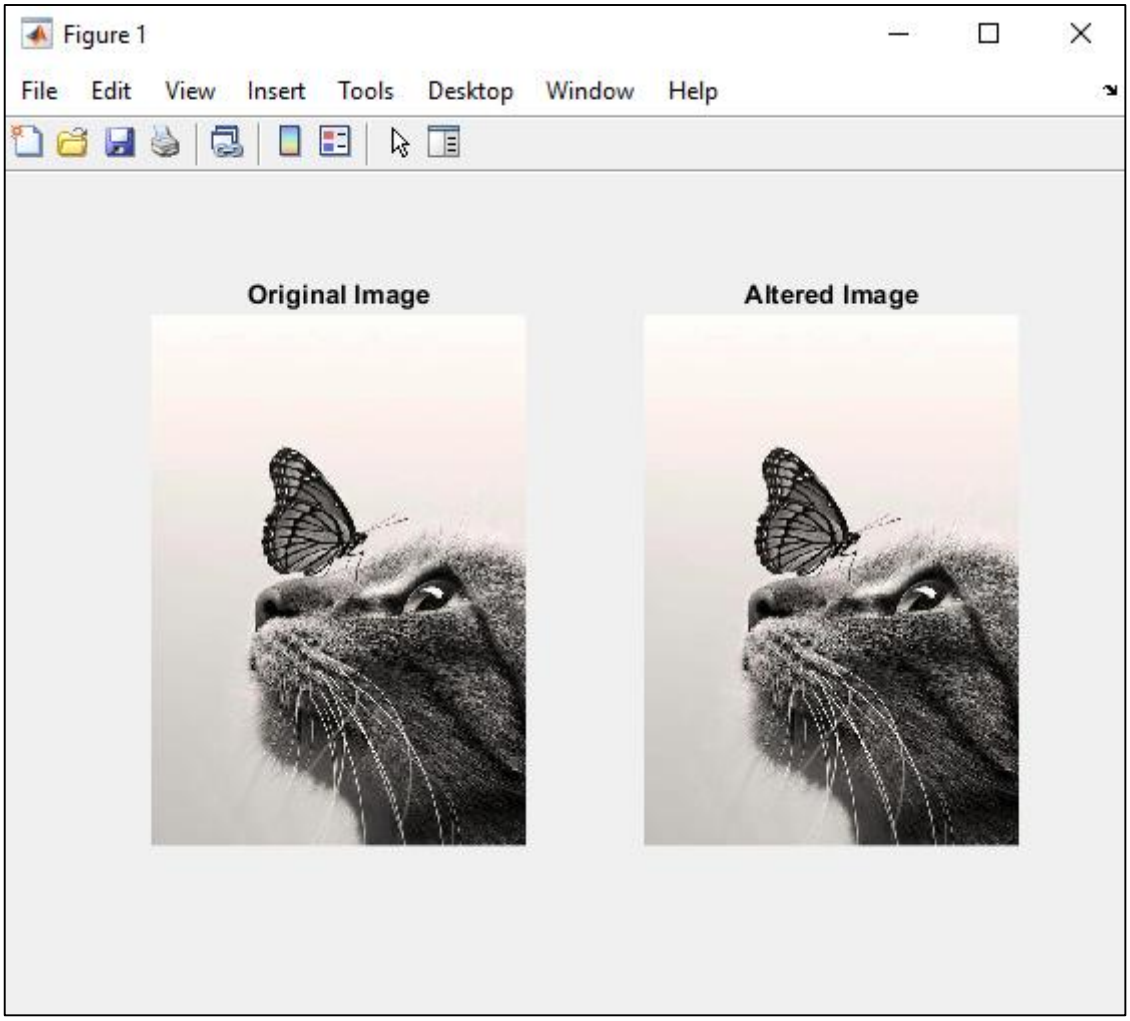

```
Ia = imread('rice.png');
Ib = imread('cameraman.tif');
Ic = imadd(Ia, Ib);
figure; subplot(2,2,1); imshow(Ia), title('Image 1');
subplot(2,2,2),imshow(Ib), title('Image 2');
subplot(2,2,3:4),imshow(Ic), title('Blended Image');
```

Explanation:

Ic = imadd(Ia, Ib); will resulted to combination of the two images.

TUTORIAL 2 – SUBTRACTION

1. Load two images and display them

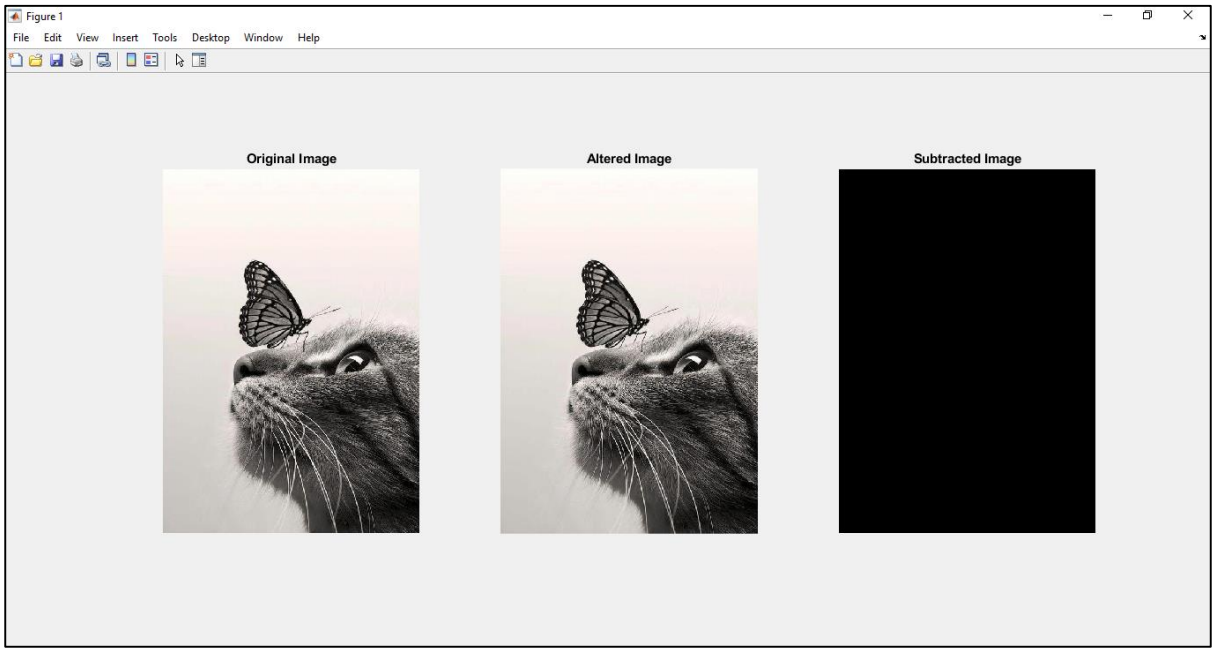



```
I = imread('cat.jpg');
J = imread('cat2.jpg');
figure(1);
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(J), title('Altered Image');
```

Explanation:

imread(); read the image from specified filename and use imshow(); display the image in the figure.

2. Subtract both images and display the result

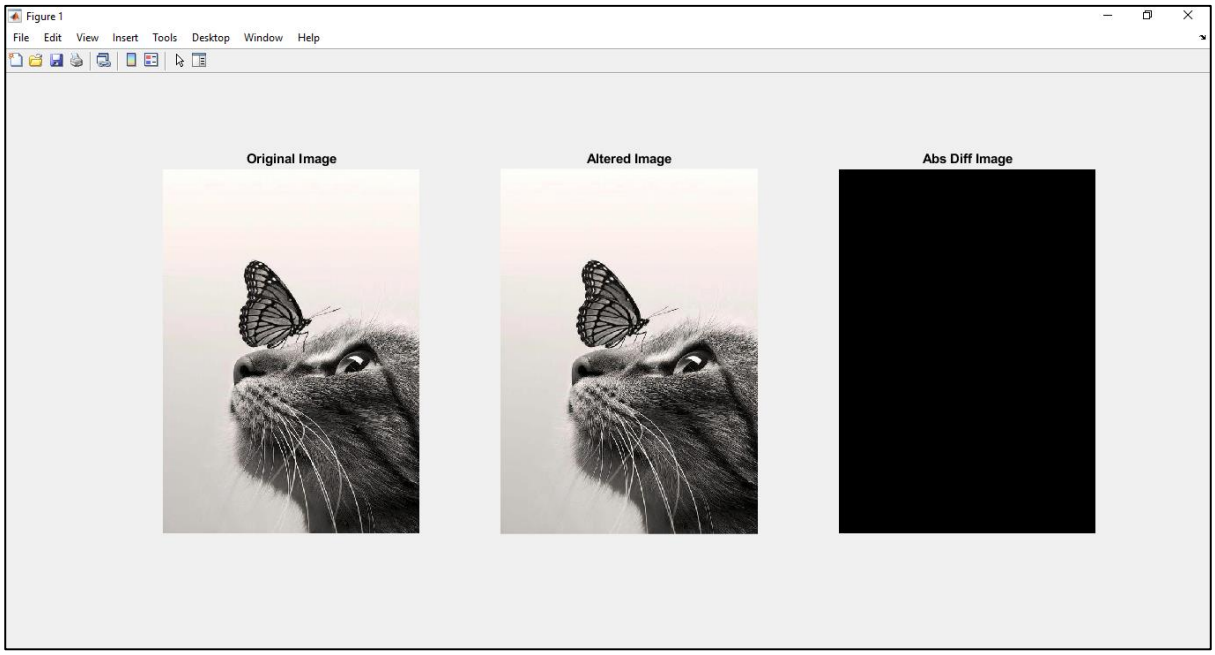



```
I = imread('cat.jpg');
J = imread('cat2.jpg');
figure(1);
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(J), title('Altered Image');

diffim = imsubtract(I,J);
subplot(1,3,3), imshow(diffim), title('Subtracted Image');
```

Explanation:

imsubtract(); subtract the image of J from I or subtract the constant from image and return the difference of image.

3. Absolute diff function




```
I = imread('cat.jpg');
J = imread('cat2.jpg');
diffim2 = imabsdiff(I,J);
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(J), title('Altered Image');
subplot(1,3,3), imshow(diffim2), title('Abs Diff Image');
```

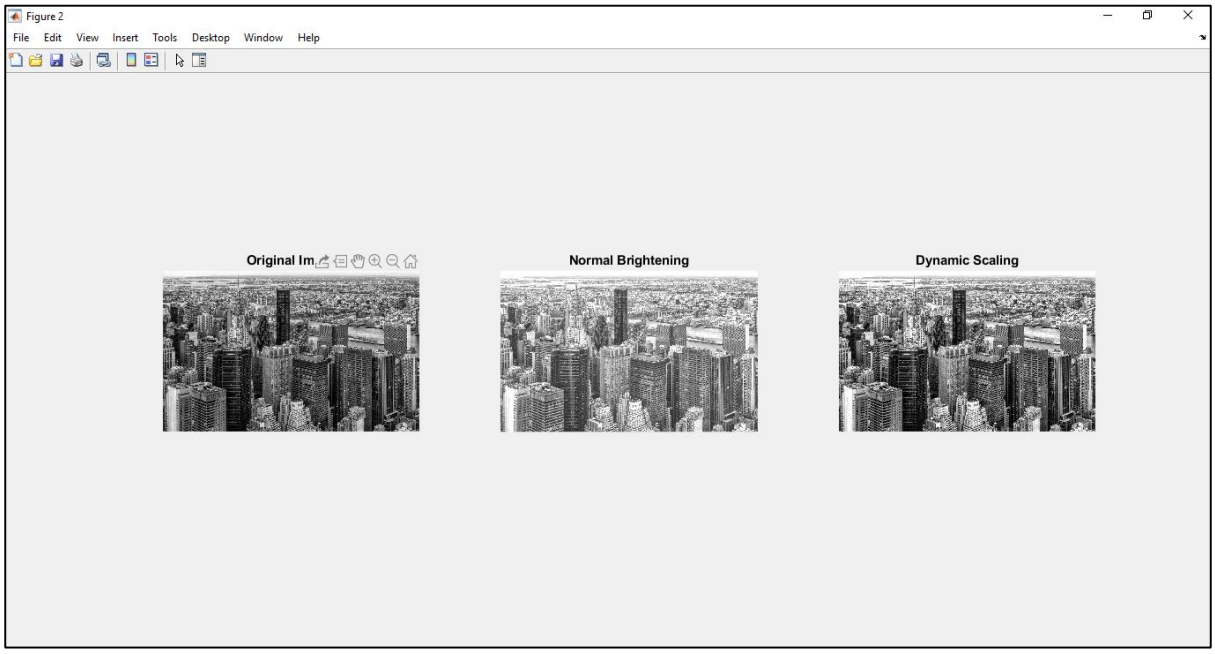
Explanation:

Imabsdiff(); calculate and display the absolute difference between the image.

Absolute value prevent negative values from being rounded to zero like subtract.

TUTORIAL 3 – MULTIPLY

a. Comparing between normal brightening and dynamic brigtening using multiply operation

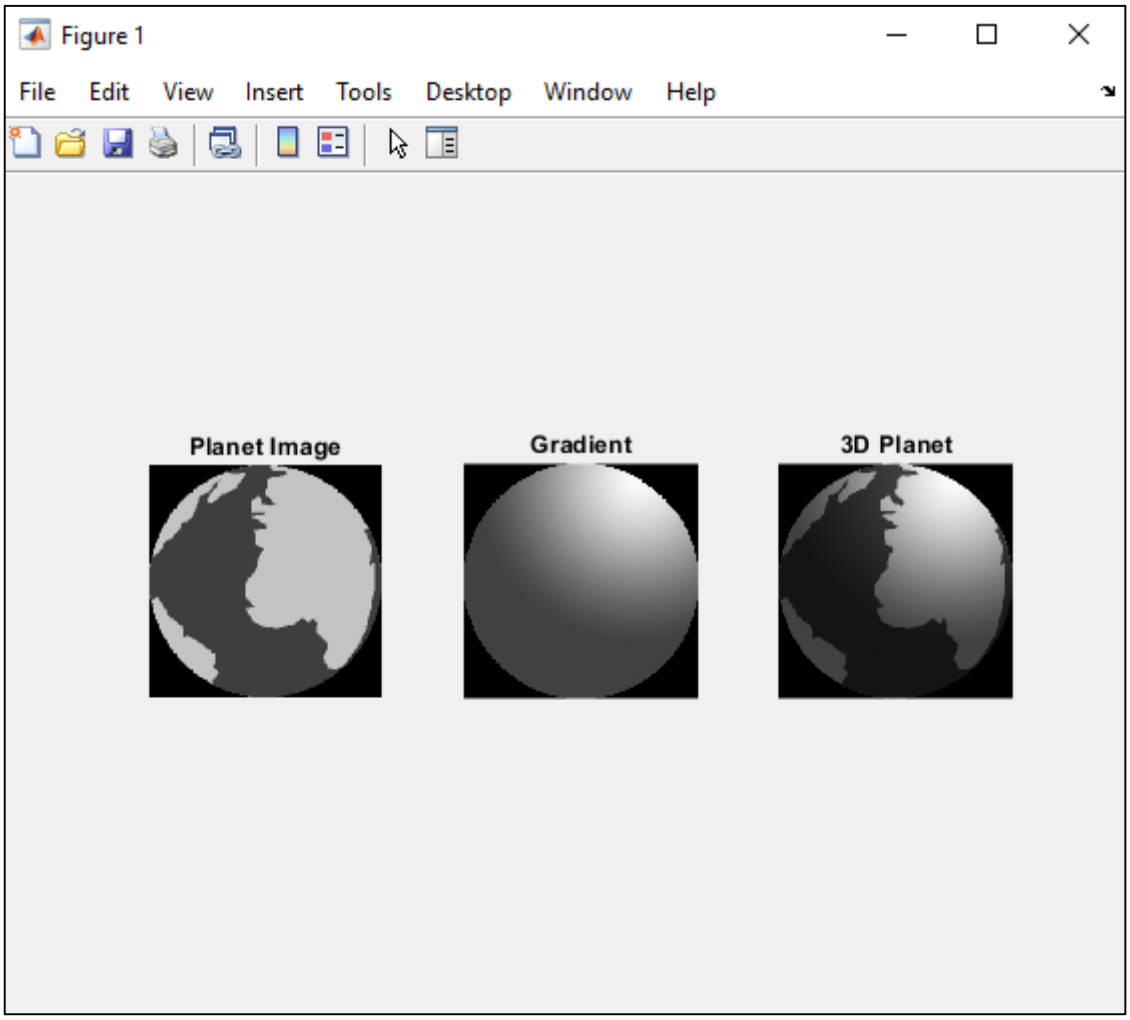



```
I = imread('ny.jpg');
I2 = imadd(I, 50);
I3 = immultiply(I, 1.2);
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I2), title('Normal Brightening');
subplot(1,3,3), imshow(I3), title('Dynamic Scaling');
```

Explanation:

imadd(); adding two image or add constant value to image
immultiply(); multiply by a constant factor of 1.2 into the each value of image to increase the sharp and brightness.

b. Blend two images using multiply operation



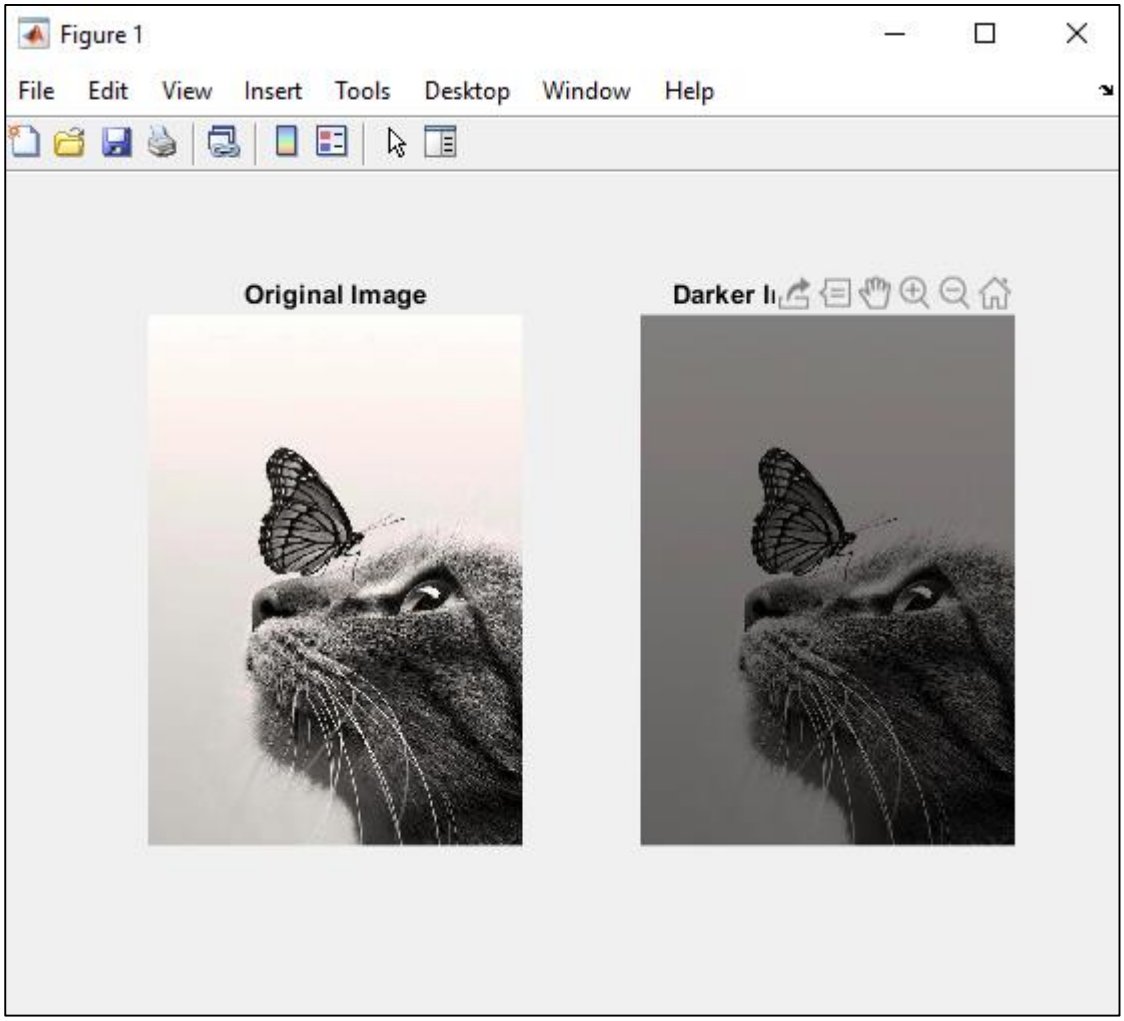

```
I = im2double(imread('earth1.tif'));
J = im2double(imread('earth2.tif'));
K = immultiply(I,J);
figure
subplot(1,3,1), imshow(I), title('Planet Image');
subplot(1,3,2), imshow(J), title('Gradient');
subplot(1,3,3), imshow(K,[]), title('3D Planet');
```

Explanation:

Immultiply(); can blend two image like additional operation. From images above it shown how the image blend together to produce the images.

TUTORIAL 4 – MULTIPLY and DIVISION

a. Use division operation to dynamically darken the image

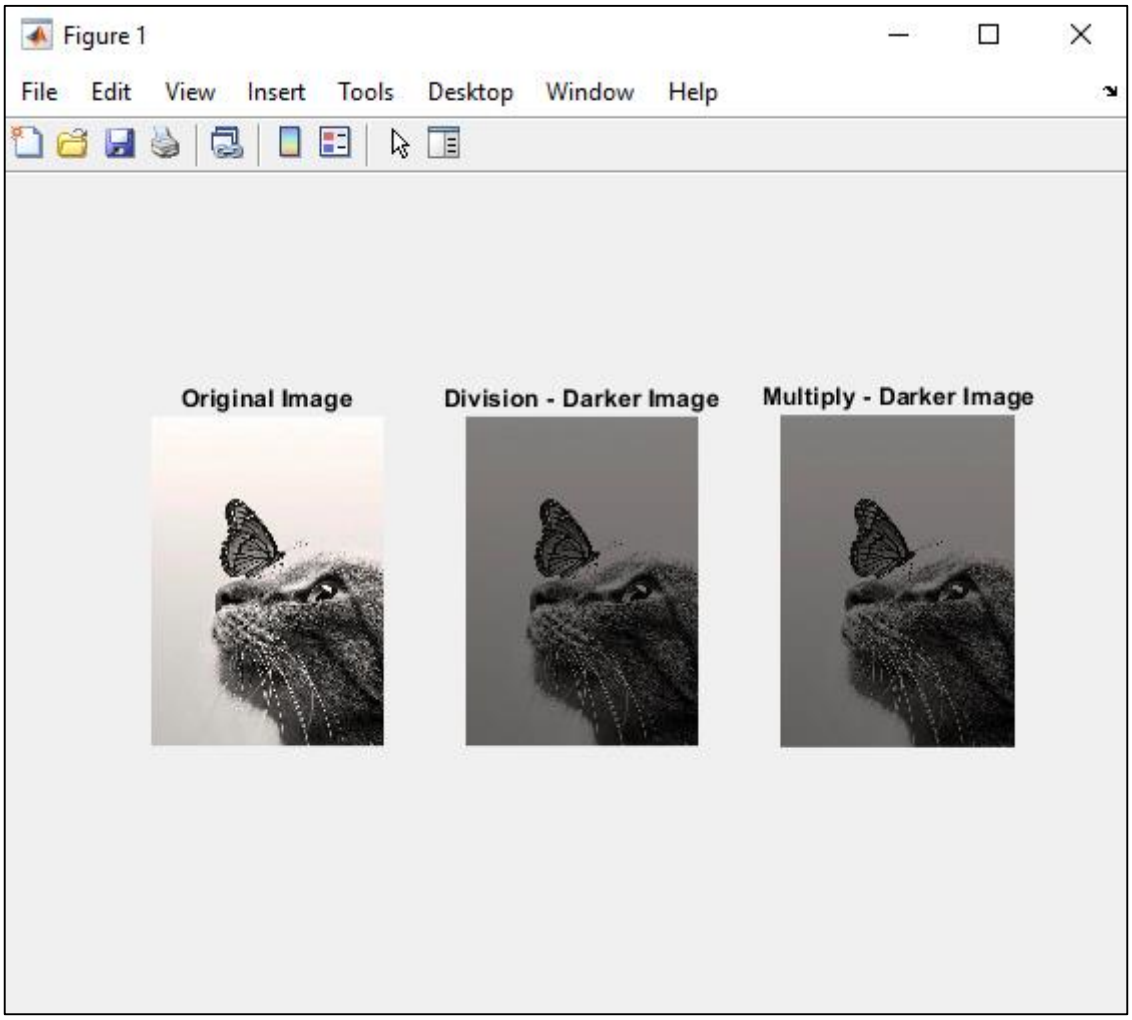



```
I = imread('cat.jpg');
I2 = imdivide(I,2);
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I2), title('Division - Darker Image');
```

Explanation:

imdivide(); divide each value of image by constant value factor of 2

b. Get the same result using multiply



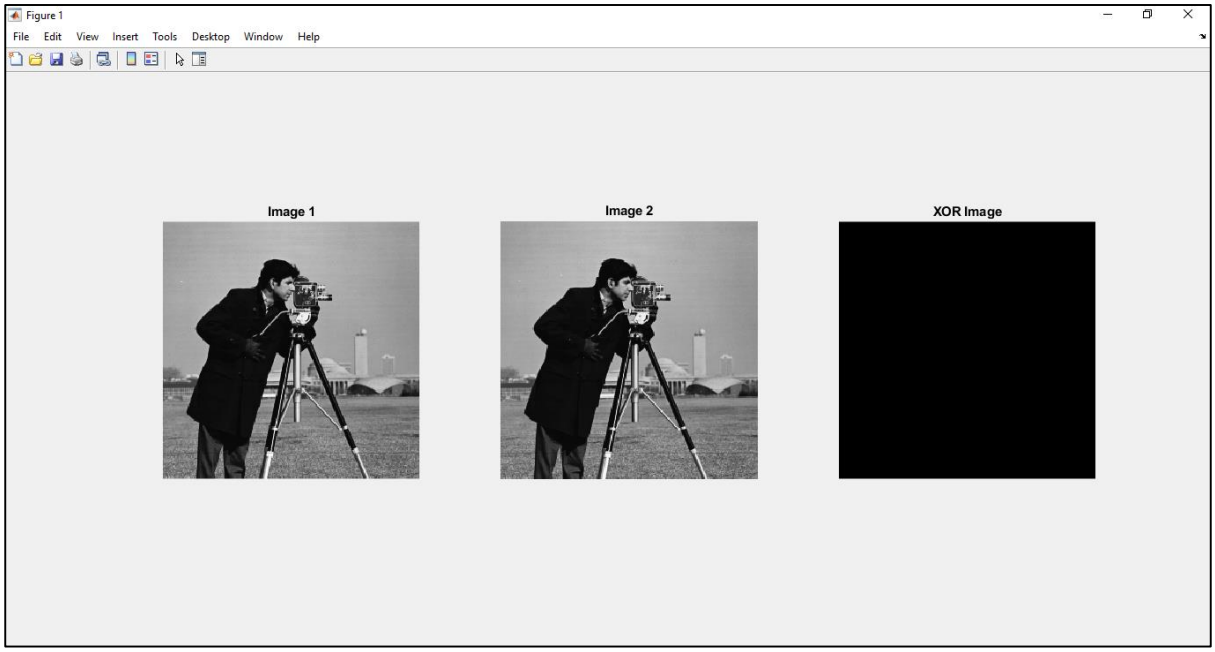

```
I = imread('cat.jpg');
I2 = imdivide(I,2);
I3 = immultiply(I,0.5);
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I2), title('Division - Darker Image');
subplot(1,3,3), imshow(I3), title('Multiply - Darker Image');
```

Explanation:

Use imdivide(); divide each value of image by constant value factor of 2 will darken the image while using immultiply(); multiply by a constant factor of 1.2 into the each value of image will give the same result as divide to produce the same pixel value

TUTORIAL 5 – LOGIC OPERATION

a. Use bitxor operation to find the different of two images

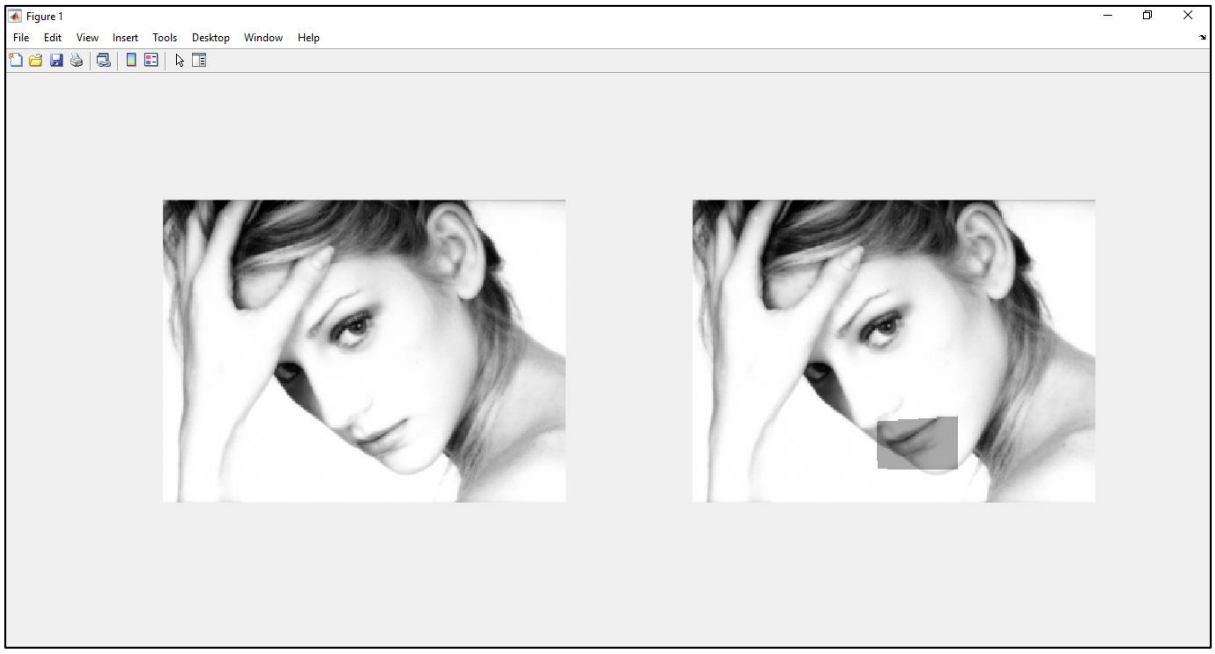



```
I = imread('cameraman.tif');
I2 = imread('cameraman2.tif');
I_xor = bitxor(I,I2);
figure; subplot(1,3,1), imshow(I), title('Image 1');
subplot(1,3,2), imshow(I2), title('Image 2');
subplot(1,3,3), imshow(I_xor,[]), title('XOR Image');
```

Explanation:

I_xor = bitxor(I,I2); returns the bitwise XOR operation of the images to find the different between the images used.

b. Make lindsay eyes dark

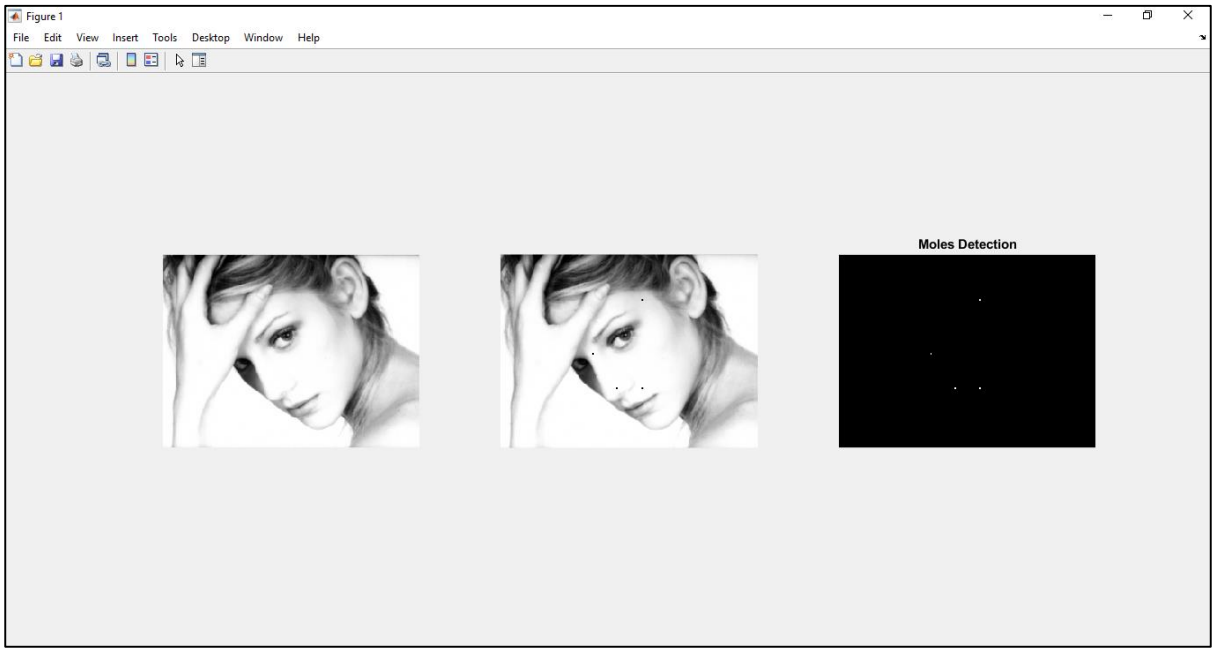



```
I = imread('lindsay.tif');
I_adj = imdivide(I,1.5);
subplot(1,2,1); imshow(I);
subplot(1,2,2); imshow(I_adj);
bw = im2uint8(roipoly(I)); % Generate a mask by creating a region of
interest polygon.
% Use logic operators to show the darker image only within the region of
% interest while displaying the original image elsewhere.
bw_cmp = bitcmp(bw); %mask complement
roi = bitor(I_adj,bw_cmp); %roi image
not_roi = bitor(I,bw); %non_roi image
new_img = bitand(roi,not_roi); %generate new image
imshow(new_img) %display new image
```

Explanation:

imdivide();divide each value of image by constant value factor of 1.5
bw = roipoly(); create binary mask image by interative polygon tool to select part of the image to be displyed.

TEST YOUR UNDERSTAND



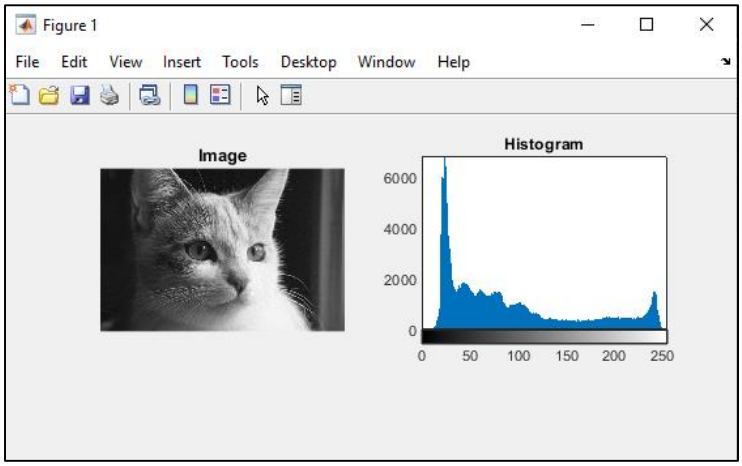

```
I = imread('lindsay.tif');
I2=I;
I2(166:167,176:177)=0;
I2(166:167,145:146)=0;
I2(56:57,176:177)=0;
I2(123:124,115:116)=0;
I3 = bitxor(I,I2);
figure;
subplot(1,3,1);imshow(I);
subplot(1,3,2);imshow(I2);
subplot(1,3,3),imshow(I3), title('Moles Detection');
```

LAB 4 – TUTORIAL ON HISTOGRAM

Tutorial 1	Displaying historgram	
	a.	Displaying an image and its histogram using 256 bins
	b.	Using different bins
	c.	Get the value of the hist to C and normalize the value
	d.	Displaying using bar char
Tutorial 2	Hist Equalization	
	a.	Image histogram equalization
	b.	Image histogram equalization 2
Tutorial 3	Hist Modification	
	a.	Addition/ Sliding in histogram
	b.	Histogram sliding
	c.	Histogram streching
	d.	Histogram shrinking
	e.	Histogram shrinking with gamma value

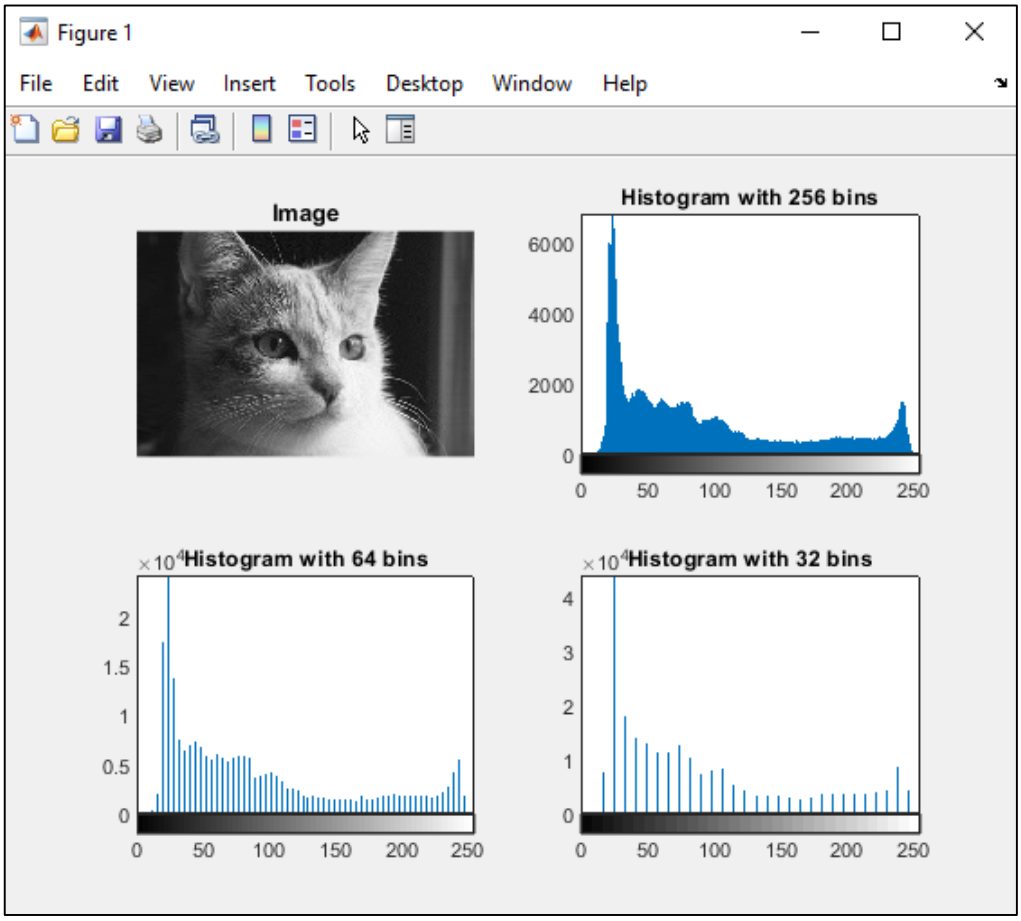
TUTORIAL 1 – DISPLAYING HISTOGRAM

a. Displaying an image and its histogram using 256 bins




```
I = imread('meow.png');
figure, subplot(2,2,1), imshow(I), title('Image')
subplot(2,2,2), imhist(I,256),axis tight, title('Histogram')
```

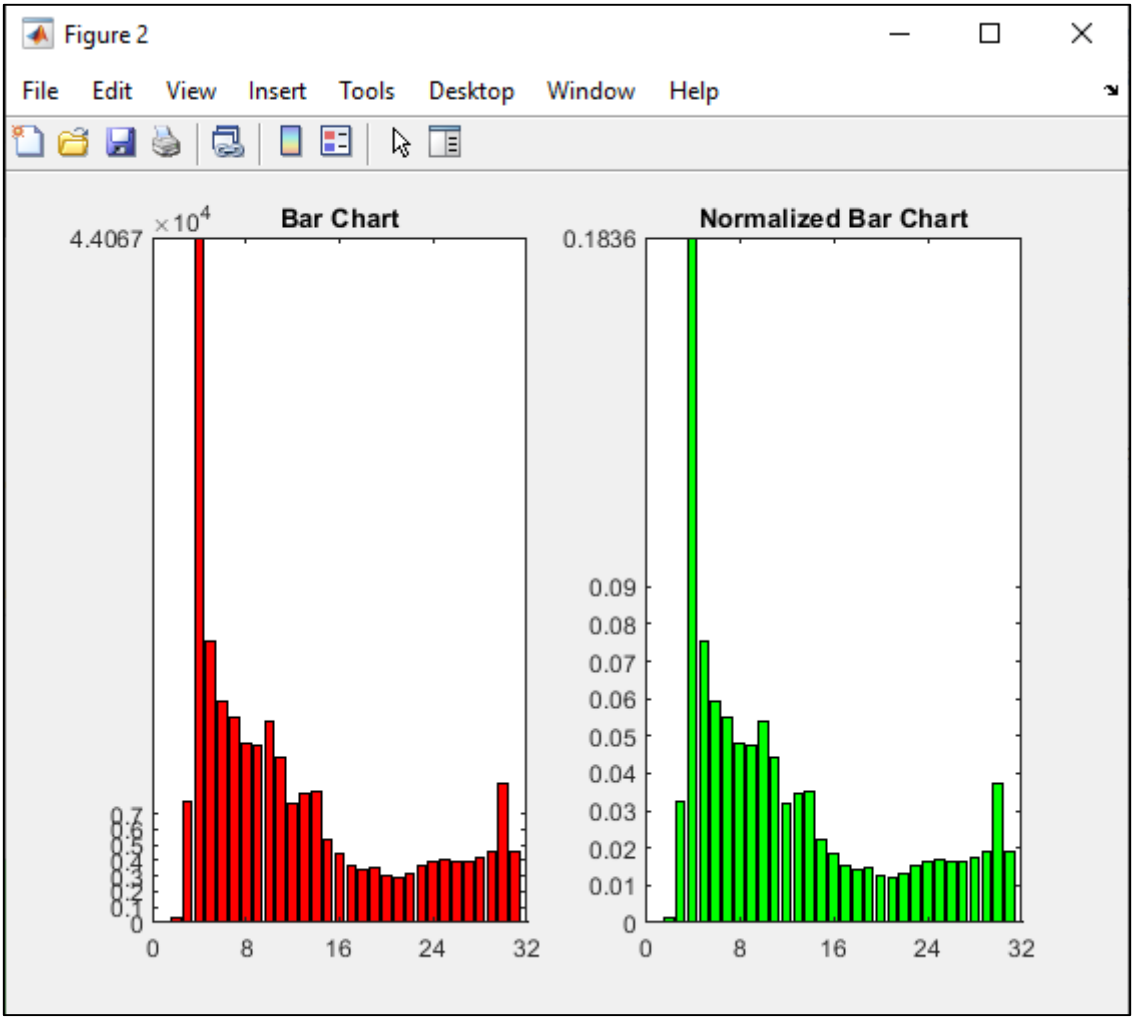
b. Using different bins



```
I = imread('meow.png');
figure, subplot(2,2,1), imshow(I), title('Image')
subplot(2,2,2), imhist(I,256),axis tight, title('Histogram with 256
bins')
subplot(2,2,3), imhist(I,64), axis tight, title('Histogram with 64 bins')
subplot(2,2,4), imhist(I,32), axis tight, title('Histogram with 32 bins')
```

The image has intensities range of 0-255. Histogram with 256 bins, can count how many pixels of each gray level of image. Number of bins used to calculate the histogram.

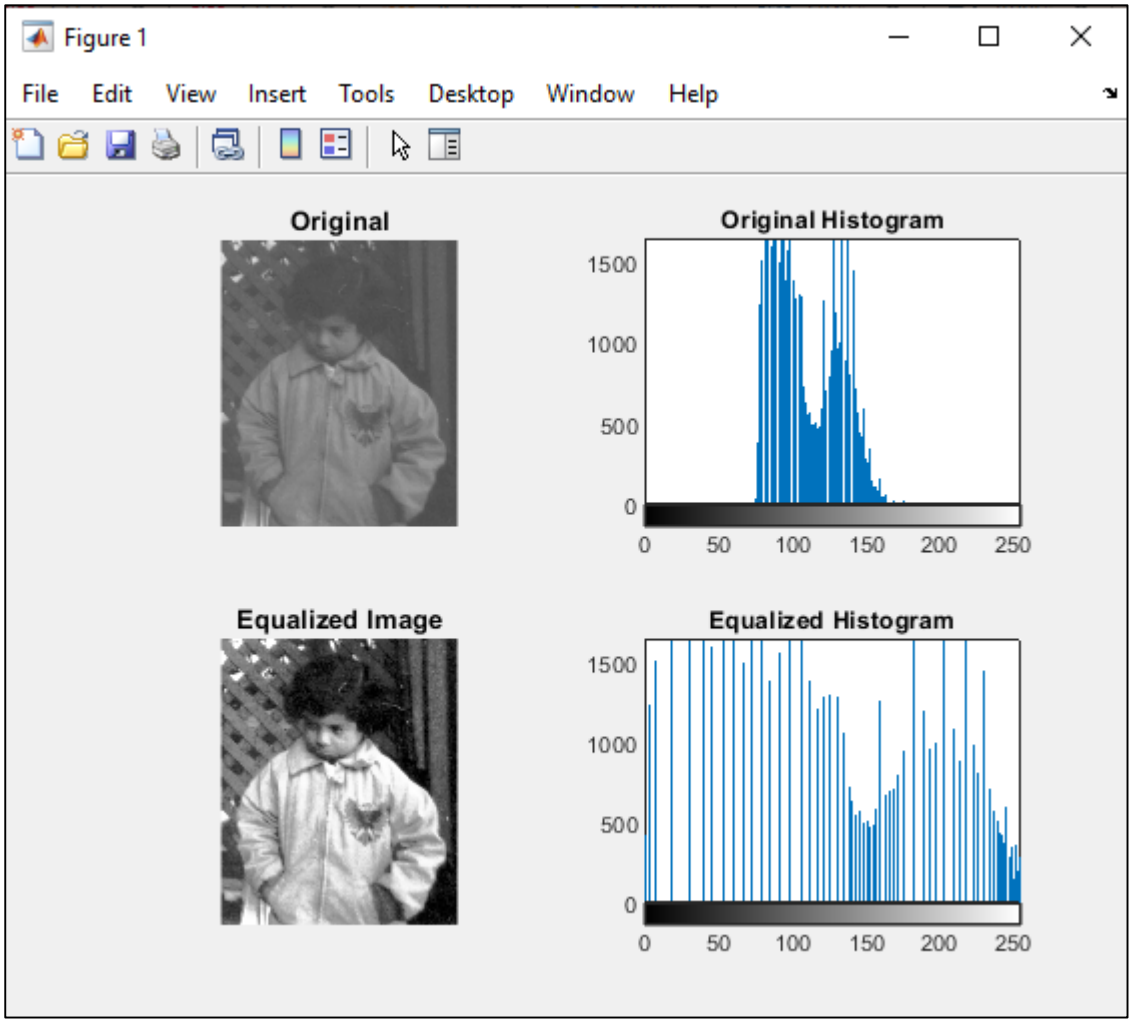
c. Get the value of the hist to C and normalize the value




```
figure, subplot(1,2,1), bar_1 = bar(c);
set(gca, 'XLim', [0 32], 'YLim', [0 max(c)]);
set(gca, 'XTick', [0:8:32], 'YTick', ...
[linspace(0,7000,8) max(c)]);
set(bar_1, 'FaceColor', 'r'), title('Bar Chart')
subplot(1,2,2), bar_2 = bar(c_norm);
set(gca, 'XTick', [0:8:32], 'YTick', ...
[linspace(0,0.09,10) max(c_norm)])
xlim([0 32]), ylim([0 max(c_norm)])
title('Normalized Bar Chart')
set(bar_2, 'FaceColor', 'g')
```

TUTORIAL 2 – HIST EQUALIZATION

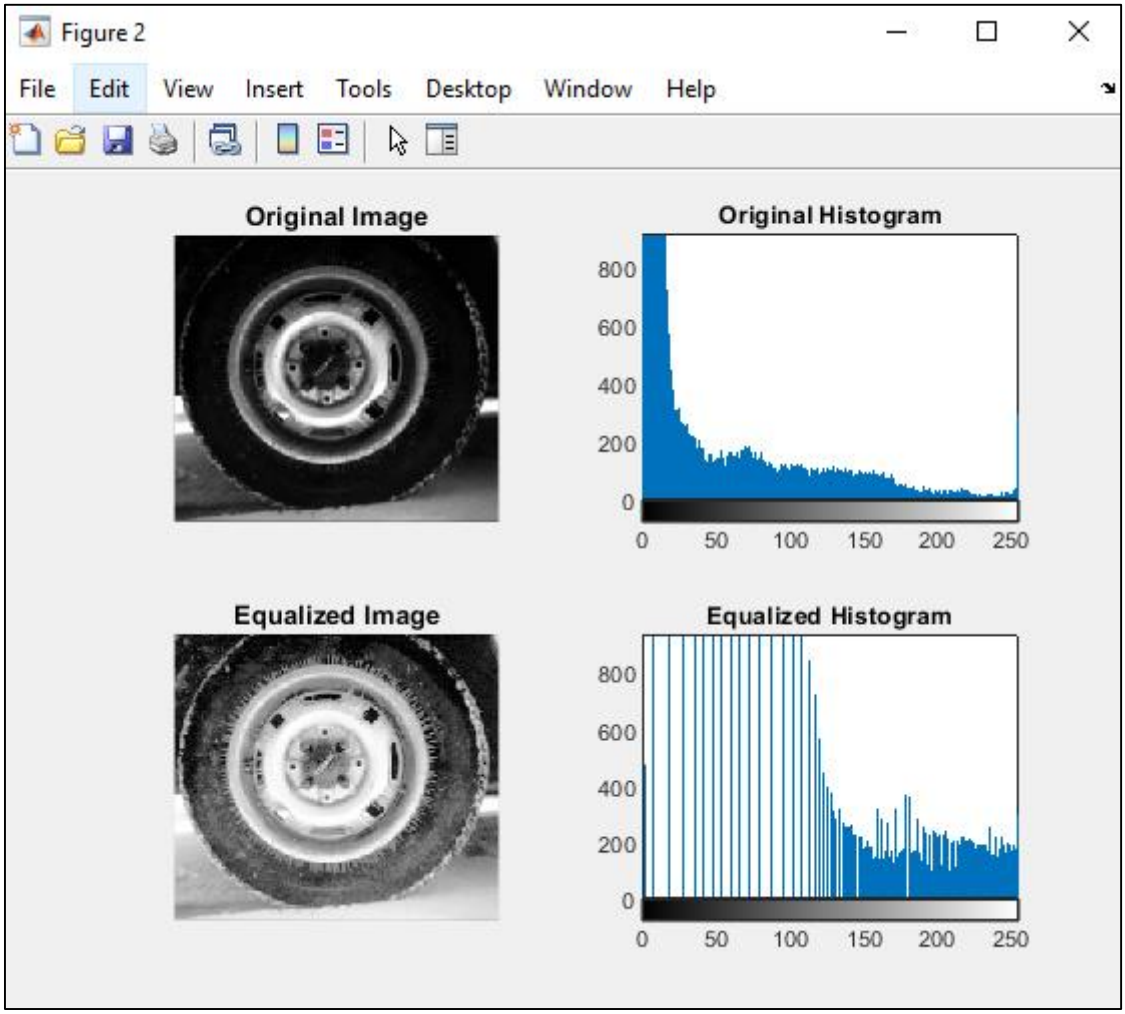
a. Image Histogram Equalization




```
I = imread('pout.tif');
figure, subplot(2,2,1),imshow(I), title('Original')
subplot(2,2,2), imhist(I), title('Original Histogram')
l_eq = histeq(I,256);
subplot(2,2,3),imshow(l_eq),title('Equalized Image')
subplot(2,2,4),imhist(l_eq),title('Equalized Histogram')
```

Histogram equalization is a technique for processing images in order to change the contrast of the image by changing the intensity distribution of the histogram. Mainly focus is technique used to improve contrast in image.

b. Image Histogram Equalization 2

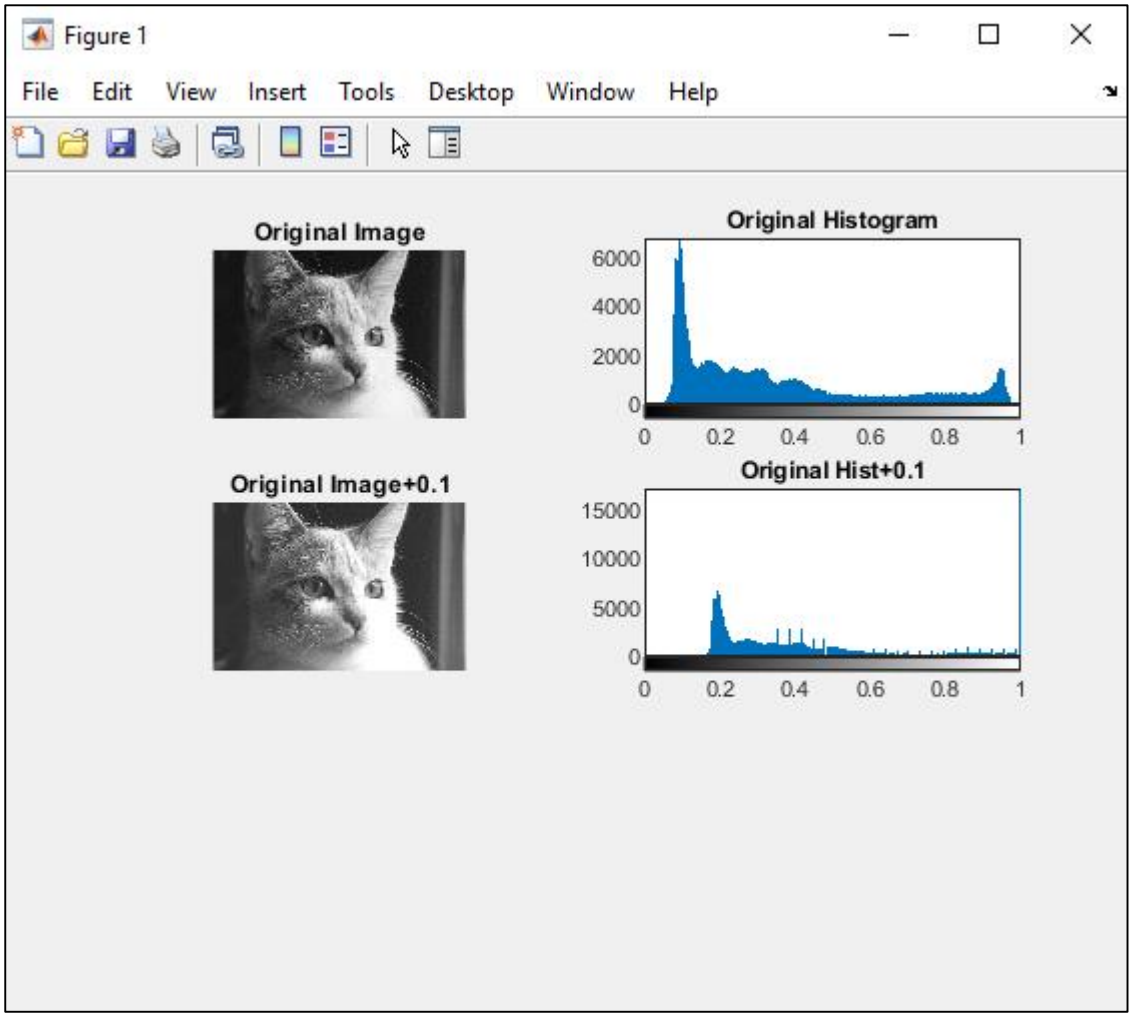



```
I=imread('tire.tif');
l_eq=histeq(I,256);
figure, subplot(2,2,1),imshow(I),title('Original Image')
subplot(2,2,2), imhist(I), title('Original Histogram')
subplot(2,2,3),imshow(l_eq),title('Equalized Image')
subplot(2,2,4),imhist(l_eq),title('Equalized Histogram')
```

Pout are more suitable than tire. The image has intensities range of 0-255 and value of grayscale image to use intensity 256 not suitable for tire because it is the high value for the original image. Imhist(I,256) is default.

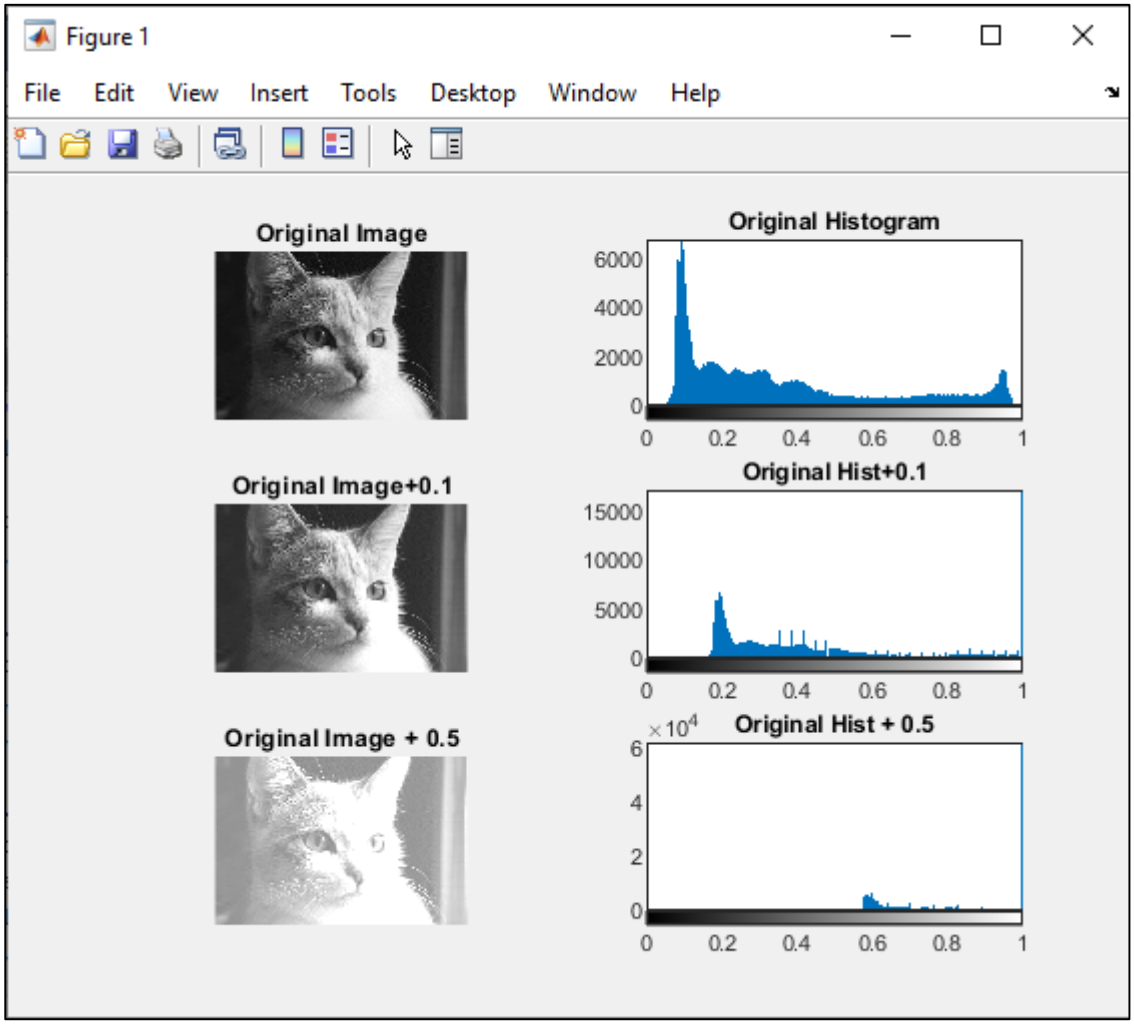
TUTORIAL 3 – HIST MODIFICATION

a. Addition/ Sliding in Histogram




```
J = imread('meow.png');
I = im2double(J);
clear J
figure, subplot(3,2,1), imshow(I), title('Original Image')
subplot(3,2,2), imhist(I), axis tight, title('Original Histogram');
const = 0.1;
I2 = I + const;
subplot(3,2,3), imshow(I2), title('Original Image + 0.1')
subplot(3,2,4), imhist(I2), axis tight, title('Original Hist + 0.1')
I2 = I + const;
subplot(3,2,3), imshow(I2), title('Original Image + 0.1')
subplot(3,2,4), imhist(I2), axis tight, title('Original Hist + 0.1')
```

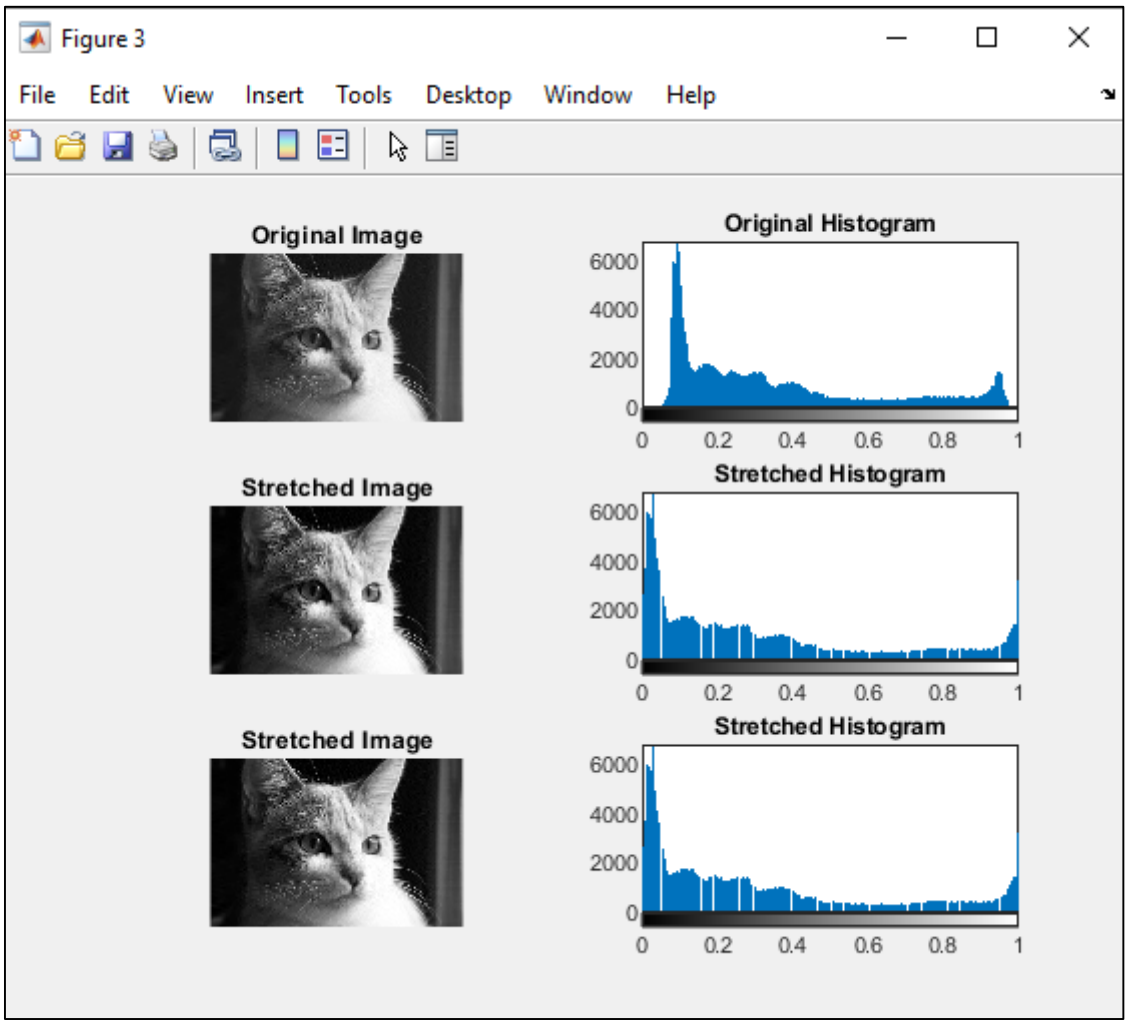
b. Histogram Sliding




```
const = 0.5;
I3 = I + const;
bad_values = find(I3 > 1);
I3(bad_values) = 1;
subplot(3,2,5), imshow(I3), title('Original Image + 0.5')
subplot(3,2,6), imhist(I3), axis tight, title('Original Hist + 0.5')
```

Increase the brightness of the images. Histogram sliding can shift to left and right to manipulating brightness.

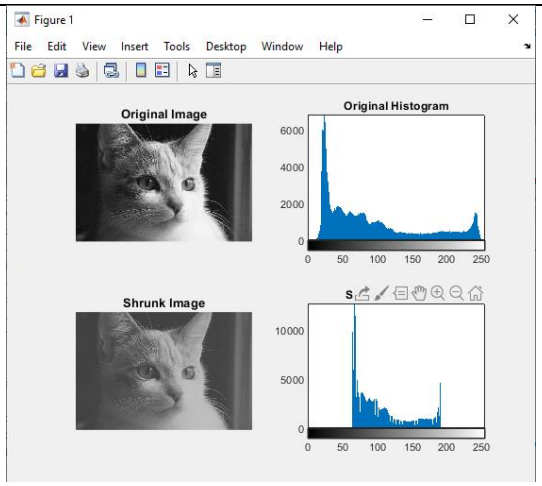
c. Histogram Streching

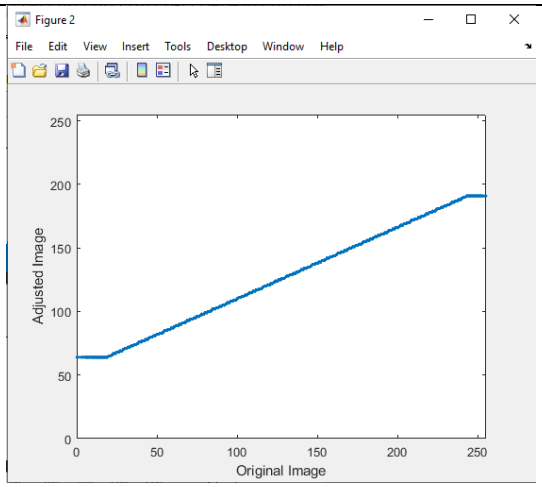



```
img_limits = stretchlim(I);
I_stretch = imadjust(I,img_limits,[]);
figure; subplot(3,2,1), imshow(I), title('Original Image')
subplot(3,2,2), imhist(I), axis tight, title('Original Histogram')
subplot(3,2,3), imshow(I_stretch), title('Stretched Image')
subplot(3,2,4), imhist(I_stretch), axis tight, title('Stretched
Histogram')

I_stretch2 = imadjust(I);
subplot(3,2,5), imshow(I_stretch2), title('Stretched Image')
subplot(3,2,6), imhist(I_stretch2), axis tight, title('Stretched
Histogram')

I_stretch_diff = imabsdiff(I_stretch, I_stretch2);

figure, imshow(I_stretch_diff,[])
min(I_stretch_diff(:))
max(I_stretch_diff(:))
```

Increase the contrast of the image. The maximum value is 225 and the minimum value is 0.

d. Histogram shrinking

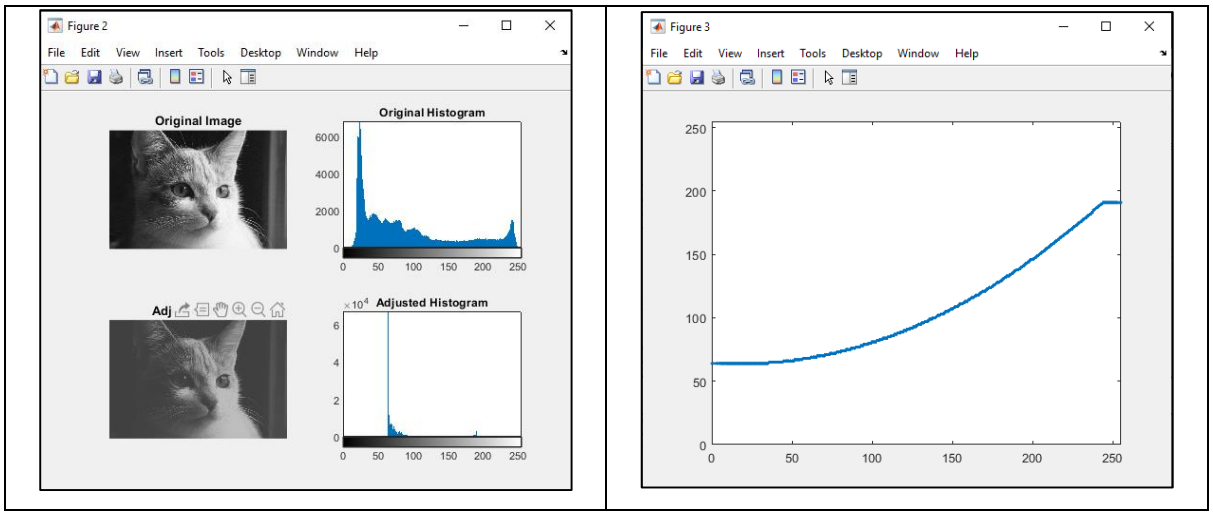





```
I = imread('meow.png');
I_shrink = imadjust(I,stretchlim(I),[0.25 0.75]);
figure;
subplot(2,2,1), imshow(I), title('Original Image')
subplot(2,2,2), imhist(I), axis tight, title('Original Histogram')
subplot(2,2,3), imshow(I_shrink), title('Shrunk Image')
subplot(2,2,4), imhist(I_shrink), axis tight,title('Shrunk Histogram')

% Display the transformation function for the adjustment performed in the
% previous step.

X = reshape(I,1,prod(size(I)));
Y = reshape(I_shrink,1,prod(size(I_shrink)));
figure, plot(X,Y,'.')
xlim([0 255]); ylim([0 255]);
xlabel('Original Image');
ylabel('Adjusted Image');
```

Histogram shrinking will decrease the contrast of the image by compressing the gray levels of the image.

e. Histogram shrinking with gamma value

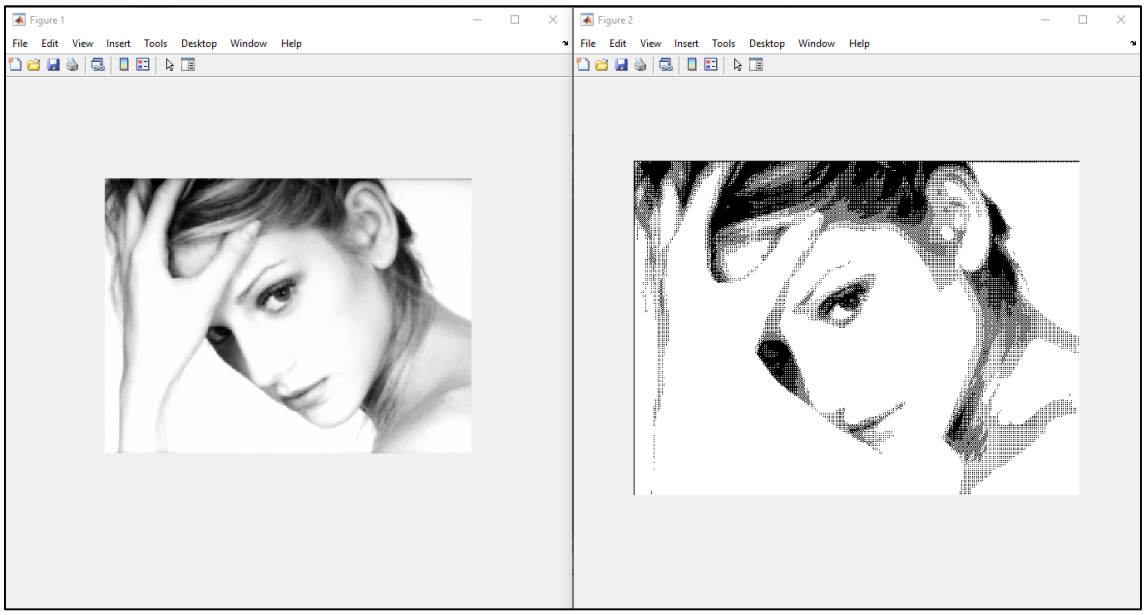



```
I_shrink = imadjust(I,stretchlim(I),[0.25 0.75],2);
X = reshape(I,1,prod(size(I)));
Y = reshape(I_shrink,1,prod(size(I_shrink)));
figure
subplot(2,2,1), imshow(I), title('Original Image')
subplot(2,2,2), imhist(I), axis tight, title('Original Histogram')
subplot(2,2,3), imshow(I_shrink), title('Adjusted Image')
subplot(2,2,4), imhist(I_shrink), axis tight, title('Adjusted Histogram')
figure, plot(X,Y,'.'), xlim([0 255]), ylim([0 255])
```

LAB 4B – TUTORIAL ON HAFTONING

Tutorial 1	Halftoning (patterning)
Tutorial 2	Dithering
Tutorial 3	Test Your Understanding

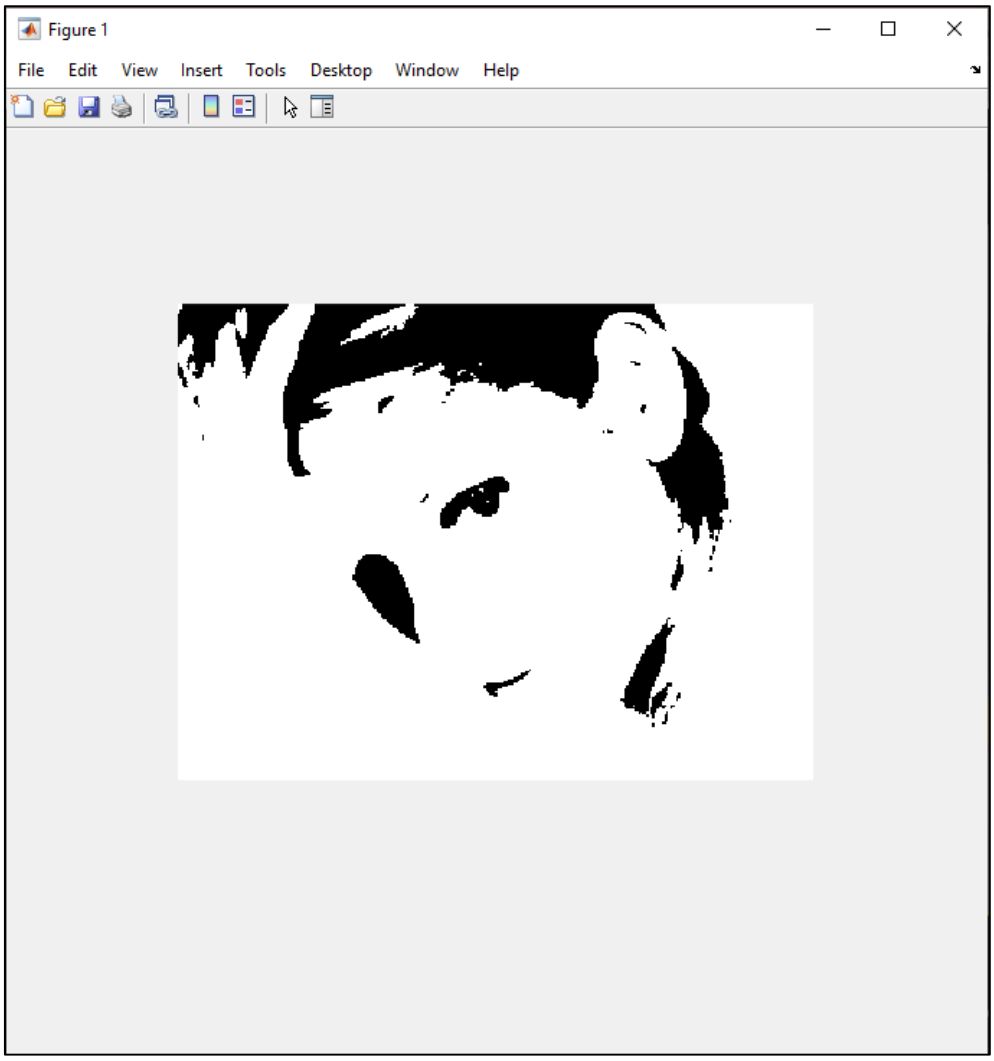
TUTORIAL 1 – HALFTONING




```
imgIn=imread('lindsay.tif');
figure('Position', [0 0 10 10]);
title('source');
imshow(imgIn);
figure('Position', [0 400 10 10]);
title('half toned');
imgOut = half_toning(imgIn);
imshow(imgOut);

function [ imgOut ] = half_toning( imgIn )
imgOut = zeros(size(imgIn)*2);
for i = 1 : size(imgIn, 1)
for j = 1: size(imgIn, 2)
if imgIn(i, j) > 50
imgOut(i*2, j*2 + 1) = 1;
end
if imgIn(i, j) > 101
imgOut(i*2 + 1, j*2) = 1;
end
if imgIn(i, j) > 152
imgOut(i*2 + 1, j*2 + 1) = 1;
end
if imgIn(i, j) > 203
imgOut(i*2, j*2) = 1;
end
end
end
end
```

Half_toning function is done by a threshold imposed on the gray level value of the image. The threshold will be replaced the pixels on where threshold place and it continue to all pixels in the image and replace with the corresponding font pattern.

TUTORIAL 2 – DILTERING



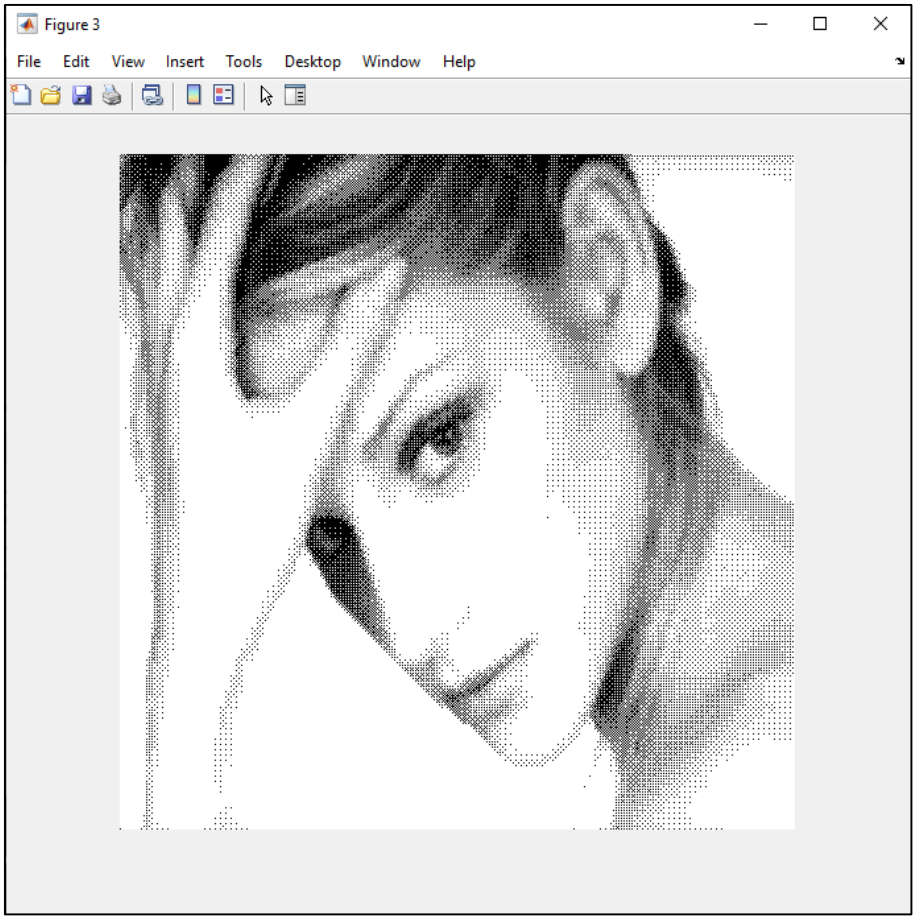

```
function [ imgOut ] = dithering( imgIn,~ )
imgOut = zeros(size(imgIn));
for i = 1 : size(imgIn, 1)
for j = 1: size(imgIn, 2)
if imgIn(i, j) > 50
imgOut(i*2, j*2 + 1) = 1;
end
if imgIn(i, j) > 101
imgOut(i*2 + 1, j*2) = 1;
end
if imgIn(i, j) > 152
imgOut(i*2 + 1, j*2 + 1) = 1;
end
if imgIn(i, j) > 203
imgOut(i*2, j*2) = 1;
end
end
end
end
```

```
clc; clear all
x=imread('lindsay.tif');
```

```
T=round(255*(B+0.5)/M);
[m, n]=size(x);
```

```
y=im2bw(x);
figure(1);
imshow(y)
x=imresize(x,[512 512]);
figure(2)
imshow(x)
B2=[1 2; 3 0];
num = 4;
M=num*num;
B4=[4*B2+1 4*B2+2; 4*B2+3 4*B2+0];
B8=[4*B4+1 4*B4+2; 4*B4+3 4*B4+0];
B16=[4*B8+1 4*B8+2; 4*B8+3 4*B8+0];
B=[0 1; 2 3];
if num==2 B=B2;
elseif num==4
    B=B4;
elseif num==8
    B=B8;
elseif num==16
    B=B16;
else
    fprint('wrong value for bayer
matrix');
end
```

```
y=zeros(512,512);
c=0;
h=(512/num)-1;
for f=0:1:h
for e=0:1:h
c=c+1;
for u=0:num-1
for q=0:num-1
r=(num*e)+1+u;
d=(num*f)+1+q;
if x(r,d)< T(u+1,q+1)
y(r,d)=0;
else
y(r,d)=255;
end
end
end
end
end
R=y;
G=y;
B=y;
%out=cat(R,G,B);
figure(3)
imshow(y)
```



Explain the difference between the 3 images.

First images was converted to the grayscale image to binary image BW by replacing all the pixels in the images. Diltering work by thresholding the image against a dither matrix where matrix is laid over the entire image and each pixel value is compared with corresponding threshold from the matrix. The pixel become white if its value exceed the threshold or black otherwise. Second image was resize the image to bigger using the dither matrix to show the image clearly. Last but not least, third image will resize the images with the scale stated. The value of rows and colums defined by vector effect the image pixels.

TUTORIAL 3 – TEST YOUR UNDERSTANDING

Question 1

a. What is the output image size if an image A of size 200x200 pixels is halftoning using 2x2 binary font using patterning method?
   Answer: 40,000

b. Using the same question above, what is the size if using dithering method?
   Answer: 160,000

Question 2

a. Given pixels' value of an image sample below. What is the new image portion below when applying patterning of 2x2 used in algorithm 1.0?
   Answer:

0	1	0	0	0	1
0	0	0	0	1	0
1	1	0	1	1	1
1	1	1	0	1	1
0	1	0	1	0	1
0	0	1	1	1	1

Question 3

a. Given pixels' value of an image sample below. What is the new image when applying dithering matric of size 2x2 D1 as below?
   Answer:

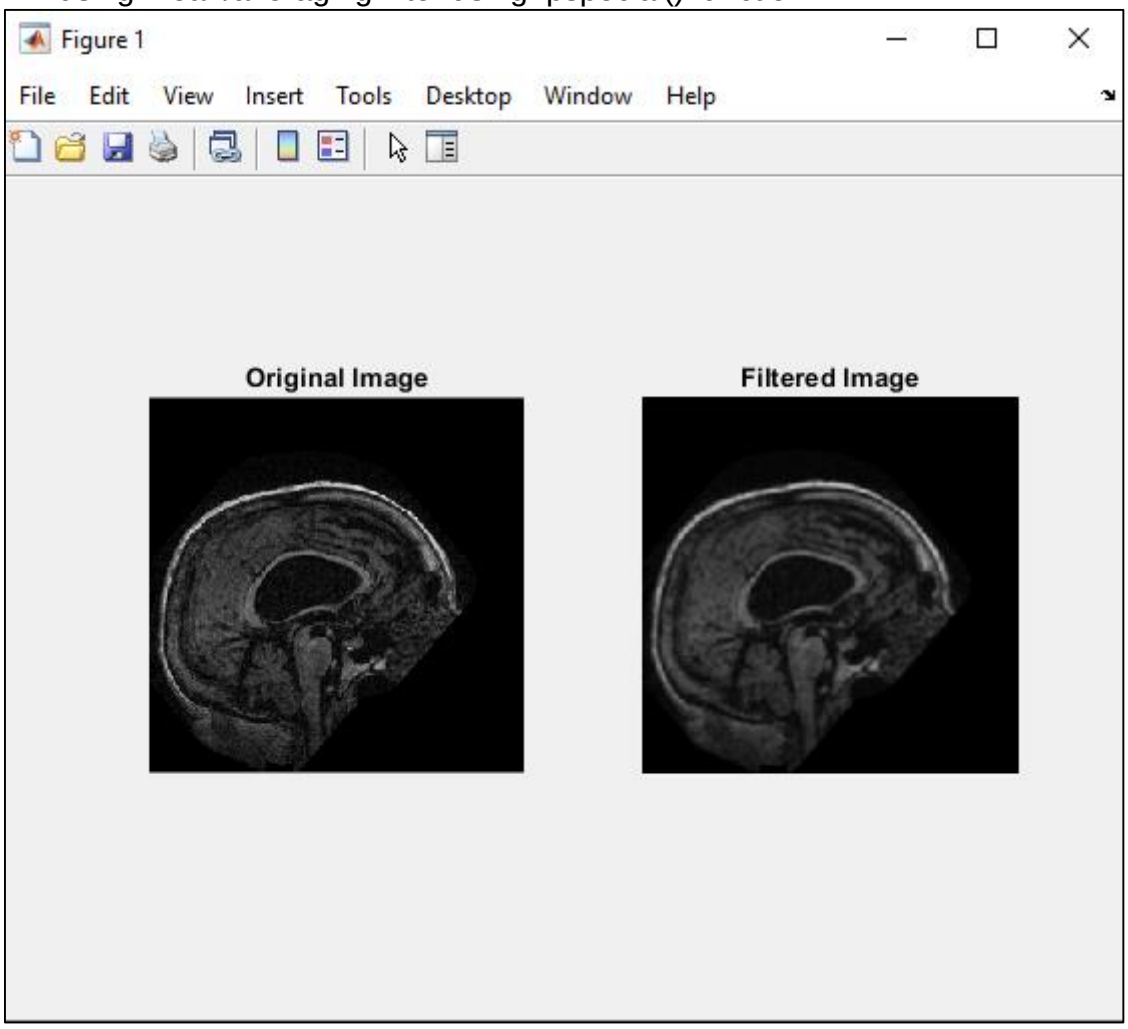
1	0	0	0
1	1	1	0
1	1	1	0
0	1	0	1

LAB 5 – TUTORIAL ON FILTERING (NEIGHBOURHOOD PROCESSING)

Tutorial 1	Smoothing use fpspecial() function and non-uniform filter
Tutorial 2	Create and apply Gaussian filter
Tutorial 3	Create and use laplacian filter
Tutorial 4	Use laplacian – composite mask
Tutorial 5	Use Laplacian filters on blur image and adjust to sharp image
Tutorial 6	Use Laplacian filters on blur image, subtract and add to sharp image
Tutorial 8	Median filter for noise restoration

TUTORIAL 1

1. using mean/averaging filter using fpspecial() function

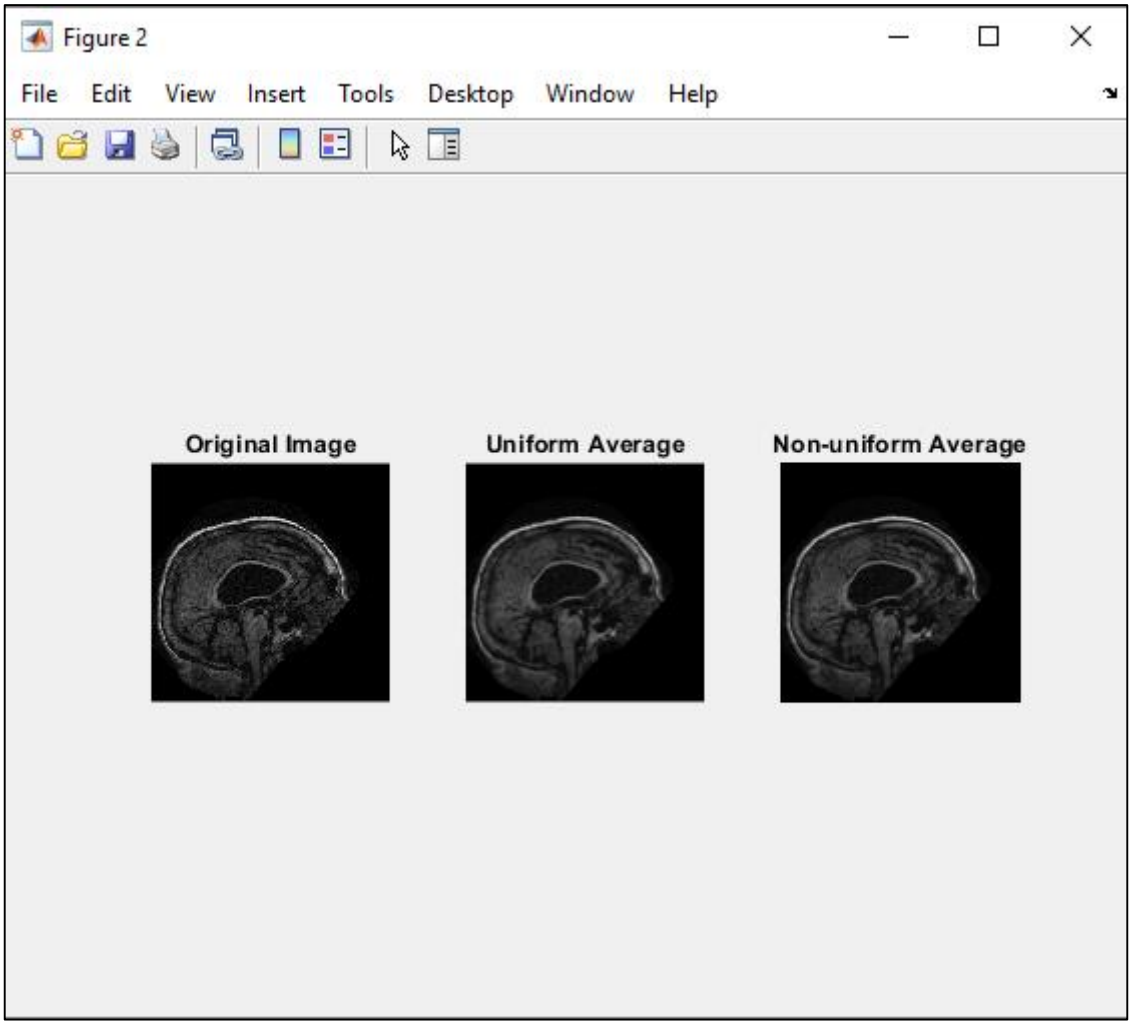



```
I = imread('OAS1_0021_MR1_mpr-4_anon_sag_66.gif');
figure,
subplot(1,2,1), imshow(I), title('Original Image');
fn = fspecial('average');
I_new = imfilter(I,fn);
subplot(1,2,2), imshow(I_new), title('Filtered Image');
```

Explanation:

fn = fspecial('average'); will return average filter to the image. Moreover, imfilter is used as a appropriate form to return the fspecial filtering type.

2. Create and use non-uniform filter on the same image



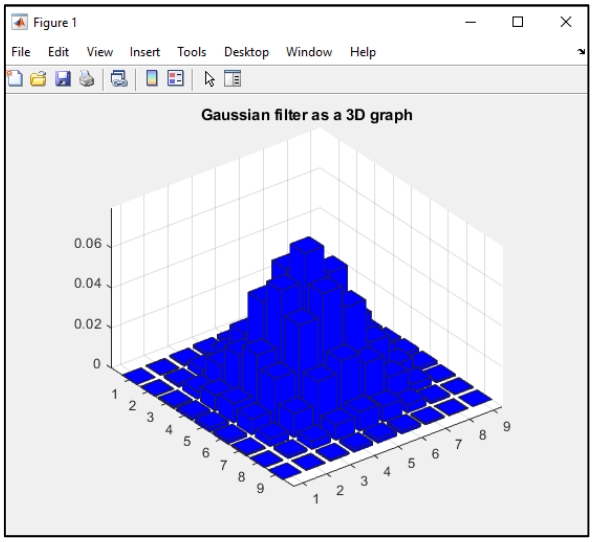

```
I = imread('OAS1_0021_MR1_mpr-4_anon_sag_66.gif');
figure,
subplot(1,3,1), imshow(I), title('Original Image');
fn = fspecial('average');
I_new = imfilter(I,fn);
%subplot(1,2,2), imshow(I_new), title('Filtered Image');

fn2 = [1 2 1; 2 4 2; 1 2 1];
fn2 = fn2 * (1/16);
I_new2 = imfilter(I,fn2);
subplot(1,3,2), imshow(I_new), title('Uniform Average');
subplot(1,3,3), imshow(I_new2), title('Non-uniform Average');
```

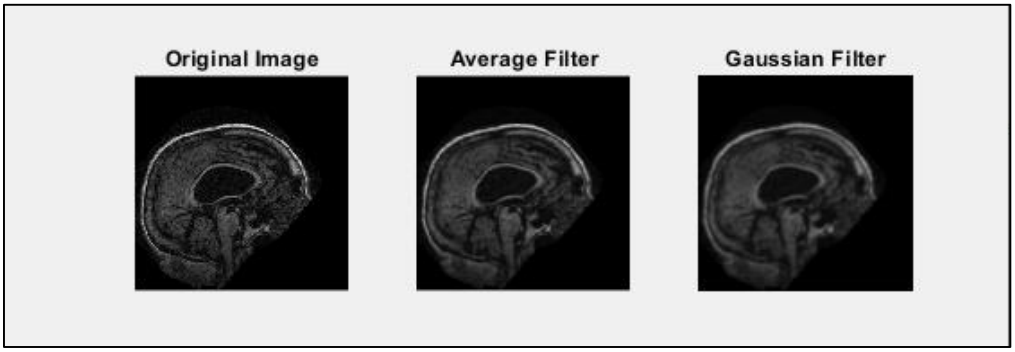
Explanation:

Uniform average is generate using mean/average filter while non-uniform average is being apply with coefficient value to the image.

TUTORIAL 2

1. Create and apply Gaussian filter





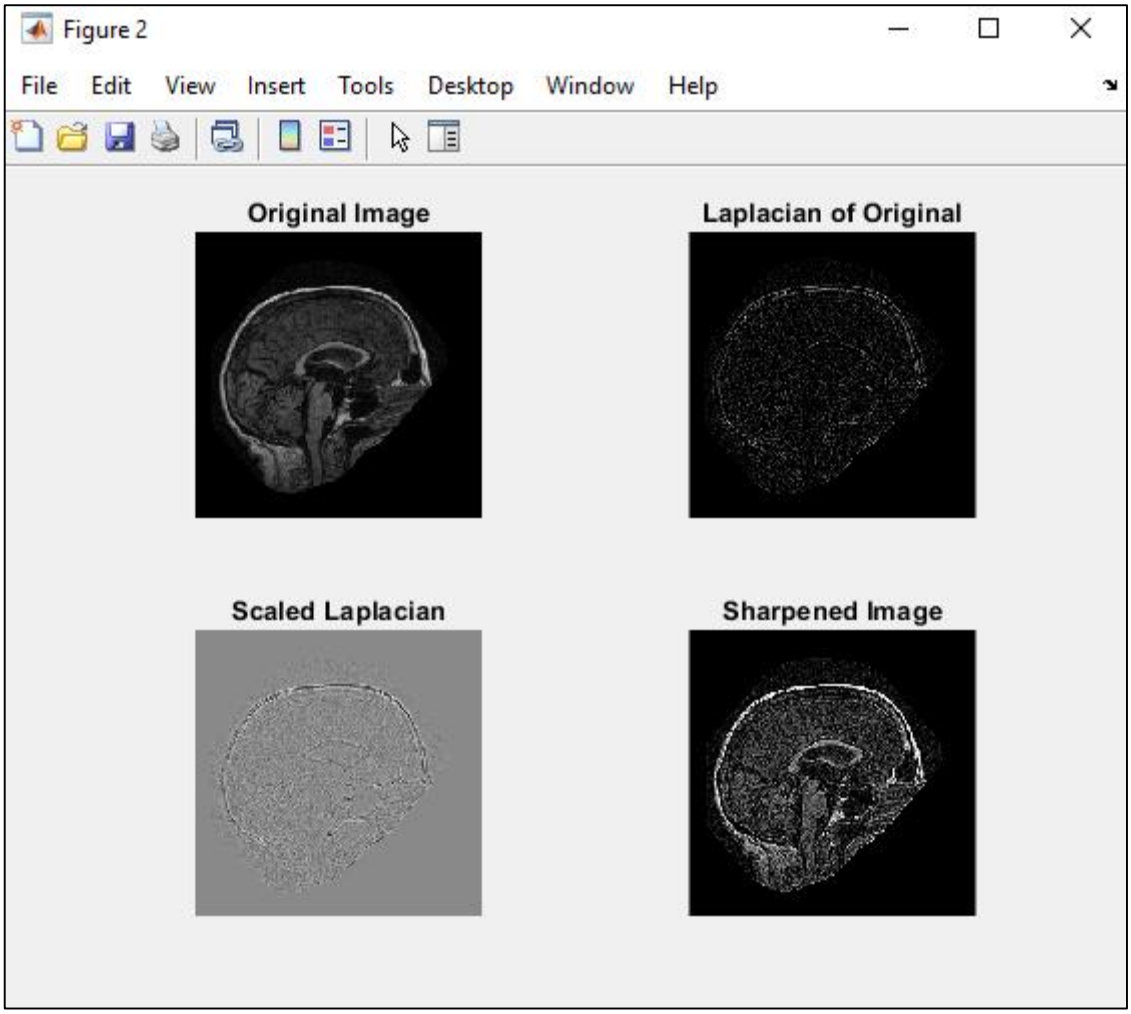

```
I = imread('OAS1_0021_MR1_mpr-4_anon_sag_66.gif');
fn_gau = fspecial('gaussian',9,1.5);
figure, bar3(fn_gau,'b'), title('Gaussian filter as a 3D graph');
I_new = imfilter(I,fn);
I_new3 = imfilter(I,fn_gau);
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I_new), title('Average Filter');
subplot(1,3,3), imshow(I_new3), title('Gaussian Filter');
```

Explanation:

Gaussian filtering is use as smoothing images. Gaussian also can remove noise and detail in the image. The Gaussian 3D graph indicate the minimization of image noise in the filter.

TUTORIAL 3

1. Create and use laplacian filter

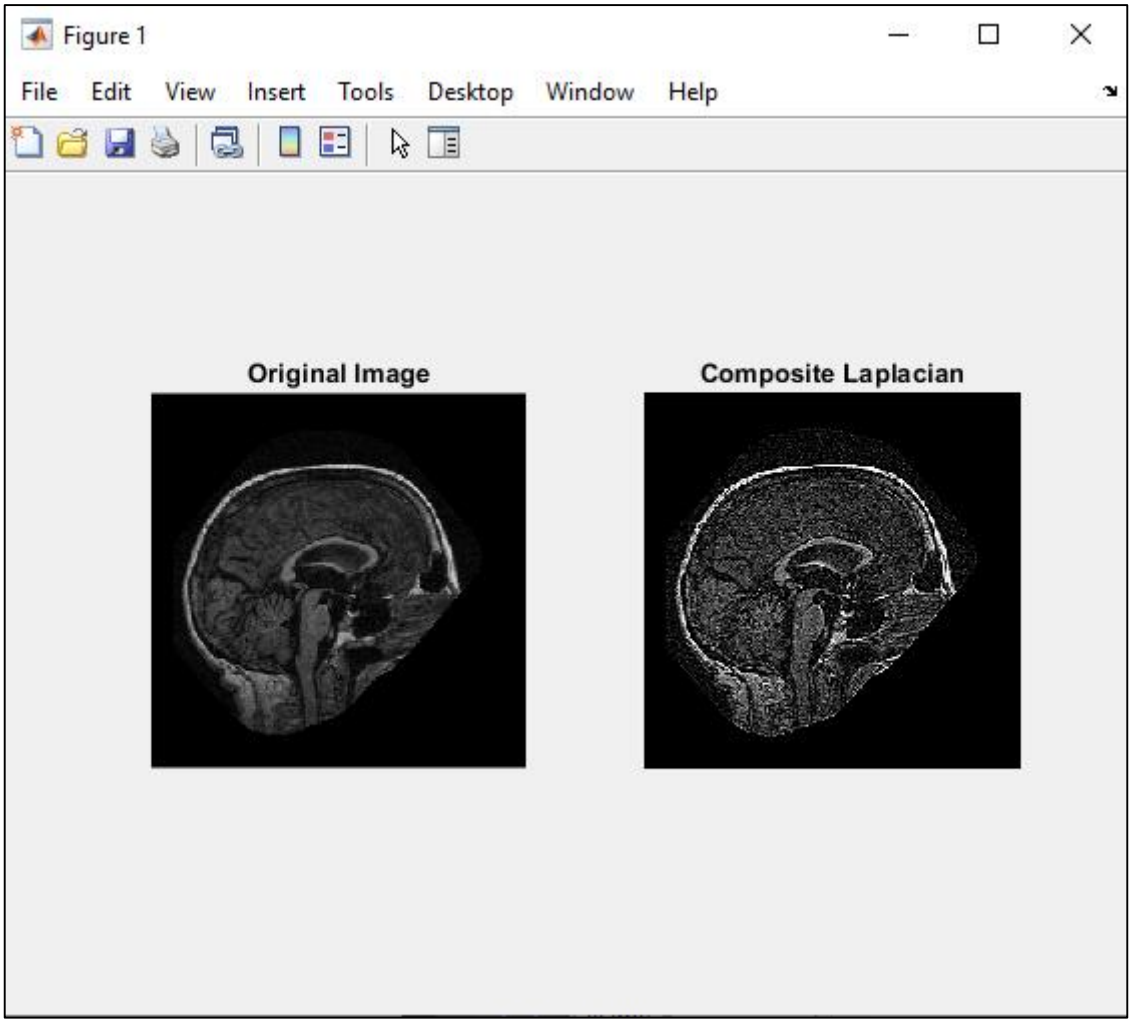



```
I = imread('OAS1_0001_MR1_mpr-4_anon_sag_66.gif');
Id = im2double(I);
figure, subplot(2,2,1), imshow(Id), title('Original Image');
f = fspecial('laplacian',0);
I_filt = imfilter(Id,f);
subplot(2,2,2), imshow(I_filt), title('Laplacian of Original');
subplot(2,2,3), imshow(I_filt,[]), title('Scaled Laplacian');
I_sharp = imsubtract(Id,I_filt);
subplot(2,2,4), imshow(I_sharp), title('Sharpened Image');
```

Explanation:

Laplacian filters are used on pictures to reduce their sensitivity to noise. It focuses on sharpening pictures. As shown in the picture above, the laplacian image shown after applying 0 to the image, 0 or alpha controls the Laplacian shape. Scaled laplacian, using filter result from laplacian to image. And lastly for sharpened image, imsubtract of image is used and produce image as follows.

TUTORIAL 4



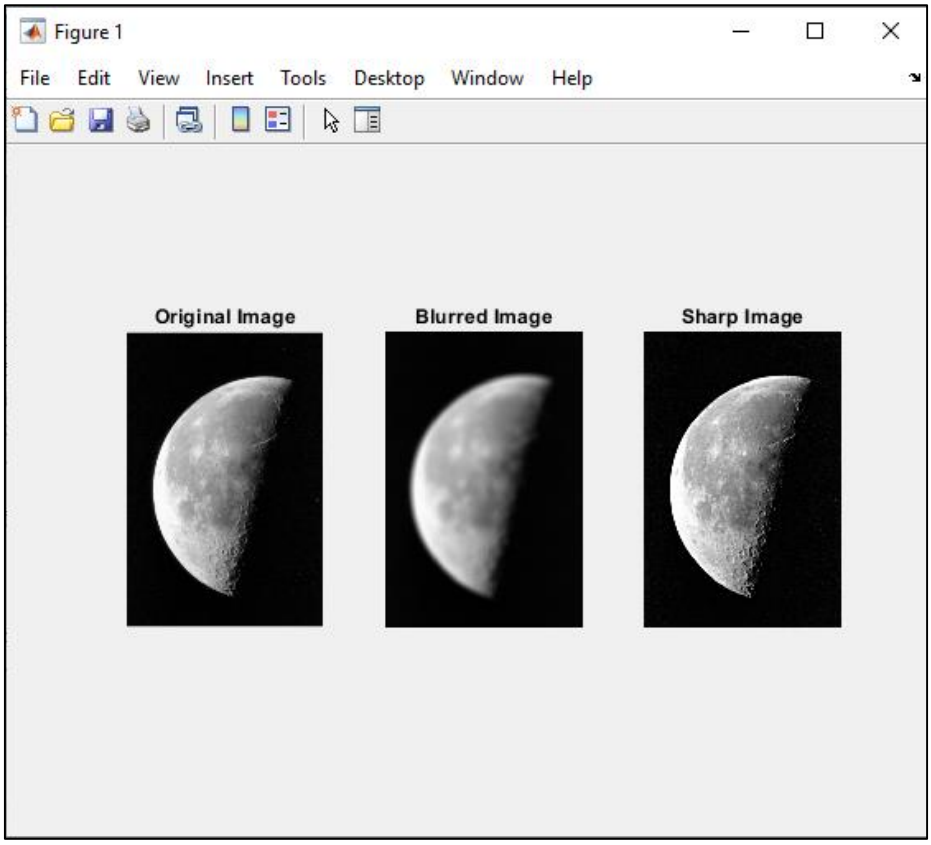

```
I = imread('OAS1_0001_MR1_mpr-4_anon_sag_66.gif');
Id = im2double(I);
f2 = [0 -1 0; -1 5 -1; 0 -1 0]
I_sharp2 = imfilter(Id,f2);
figure, subplot(1,2,1), imshow(Id), title('Original Image');
subplot(1,2,2), imshow(I_sharp2), title('Composite Laplacian');
```

Explanation:

This method is another way of using laplacian using a composite mask.

TUTORIAL 5

1. Another way in using lapacian filter on blur image



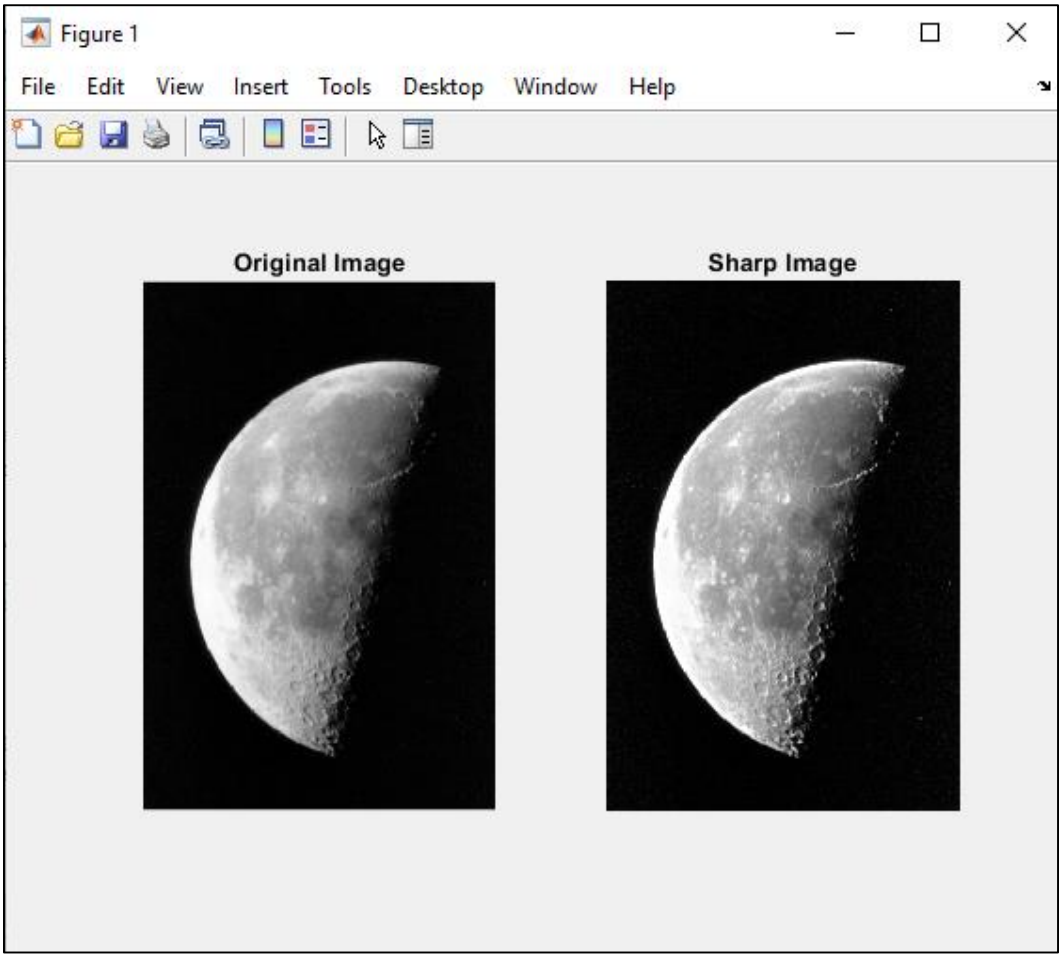

```
I = imread('moon.tif');
f_blur = fspecial('average',13);
I_blur = imfilter(I,f_blur);
figure, subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I_blur), title('Blurred Image');

I_blur_adj = imadjust(I_blur,stretchlim(I_blur),[0 0.4]);

I_sharp = imsubtract(I,I_blur_adj);

I_sharp_adj = imadjust(I_sharp);
subplot(1,3,3), imshow(I_sharp_adj), title('Sharp Image');
```

Explanation:

This process are use to apply lapacian filter on blur image. Average filter are being use with the hsize of 13. After that filter being applied on image then use imadjust for intensity values in grayscale image. Imadjust shrink the histogram of the blur image after that subtract the blurred image from the original image. Last step, stretch the sharpened image histogram to the full dynamic grayscale.

TUTORIAL 6

1. Another way in using lapacian filter on blur image



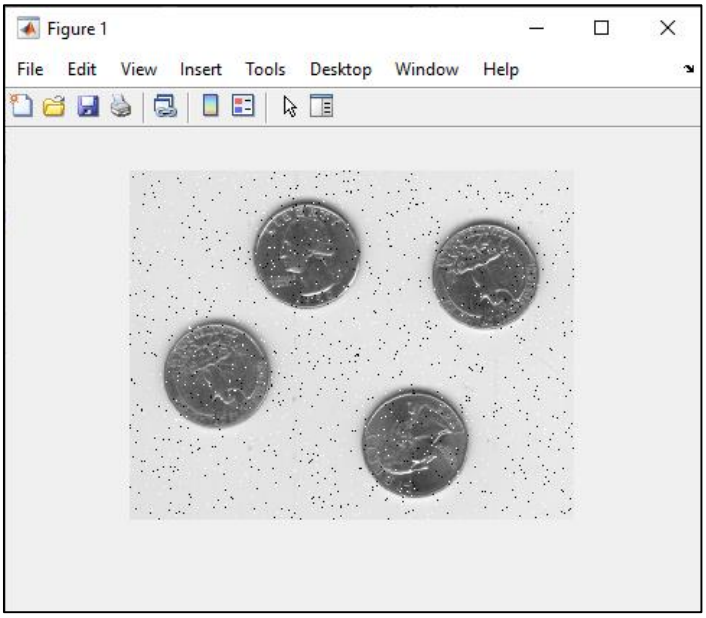

```
I = imread('moon.tif');
f_blur = fspecial('average',13);
I_blur = imfilter(I,f_blur);
figure
subplot(1,2,1), imshow(I), title('Original Image');

I_sharpening = imsubtract(I,I_blur);

% Add sharpening image to original image to produce final result.
I_sharp2 = imadd(I,I_sharpening);
subplot(1,2,2), imshow(I_sharp2), title('Sharp Image');
```

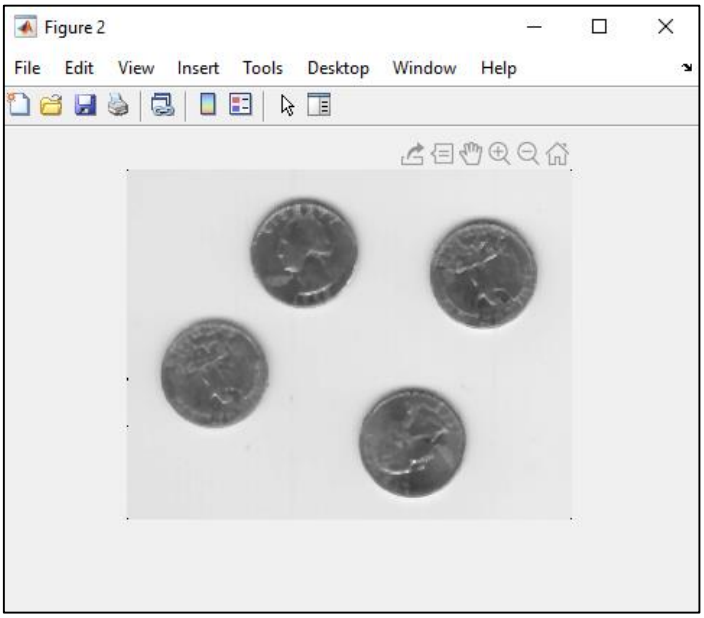
Explanation:

There are other methods to apply lapacity filters on blurred images by subtracting blurred images from the original image to produce a sharpening image. After that, add the sharpening picture to the original image to produce the result.

TUTORIAL 8

1. Sample median filter usage for noise restoration






```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
K = medfilt2(J);
imshow(J), figure, imshow(K)
```

Explanation:

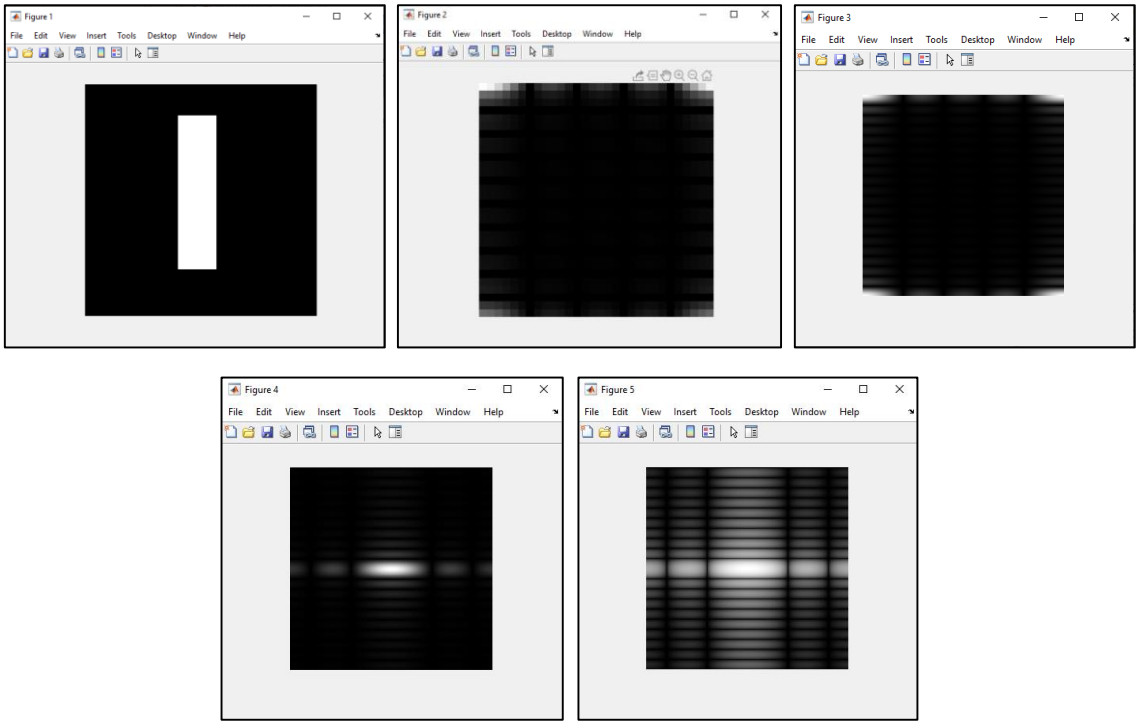
Add salt and pepper noise with neighborhood size. Then use a median filter to filter out the noise from the image.

LAB 6 – TUTORIAL FREQUENCY DOMAIN

Tutorial 1	How to Display a Fourier Spectrum using MATLAB
Tutorial 2	Low Pass Filter in Freq Domain
Tutorial 3	High Filter in Freq Domain
Tutorial 4	Notch Filter in Freq Domain

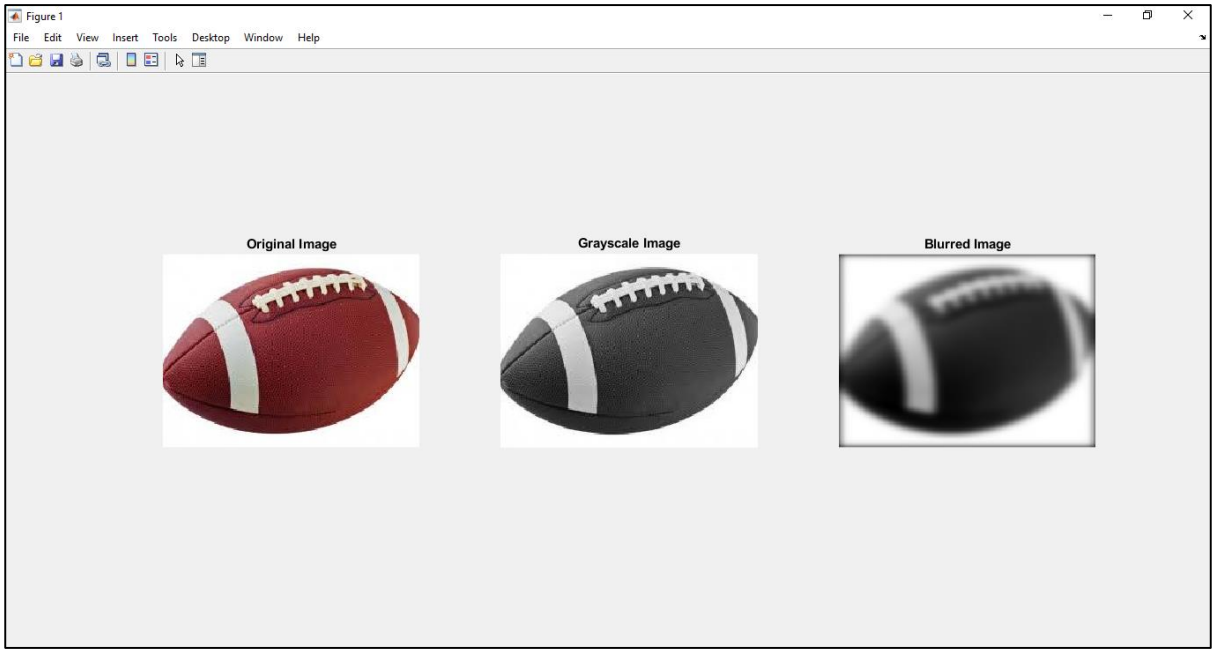
TUTORIAL 1: How to Display a Fourier Spectrum using MATLAB

a. Draw an image




```
f=zeros(30,30); %Create a black 30x30 image
f(5:24,13:17)=1; %With a white rectangle in it.
imshow(f,'InitialMagnification', 'fit'); % show the image

F=fft2(f); %Calculate the DFT.
%There are real and imaginary parts to F. Use the abs function to compute the %
magnitude of the combined components.
F2=abs(F);
figure, imshow(F2,[], 'InitialMagnification','fit') % show the image

%To create a finer sampling of the Fourier transform, you can add zero %
%padding to f when computing its DFT. Also note that we use a power of 2,
%2^256 . This is because the FFT -Fast Fourier Transform - is fastest when the
%image size has many factors.
F=fft2(f, 256, 256); F2=abs(F);
figure, imshow(F2, [])

%The zero-frequency coefficient is displayed in the upper left hand
corner.
%To display it in the center, you can use the function fftshift.
F2=fftshift(F);
F2=abs(F2);
figure,imshow(F2,[])

%In Fourier transforms, high peaks are so high they hide details.
%Reduce contrast with the log function.
F2=log(1+F2);
figure,imshow(F2,[])
```

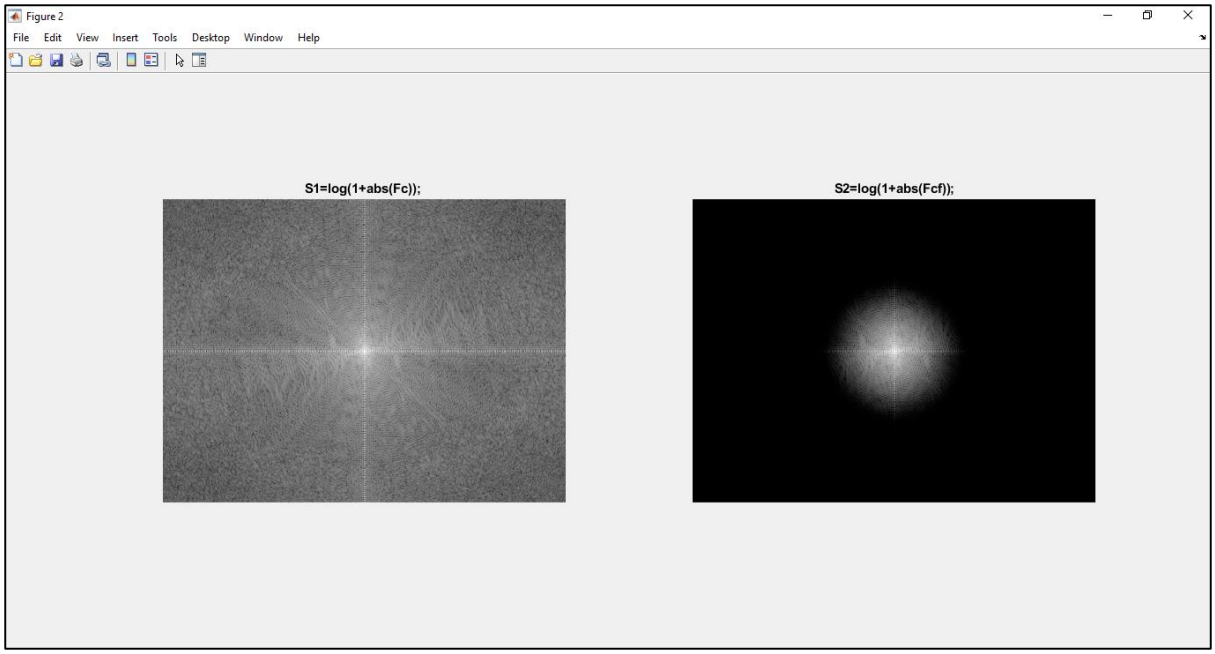
TUTORIAL 2: Low Pass Filter in Freq Domain




```
footBall=imread('football.jpg');
figure,
subplot(1,3,1), imshow(footBall), title('Original Image');
footBall=rgb2gray(footBall);
subplot(1,3,2), imshow(footBall), title('Grayscale Image');
PQ = paddedsize(size(footBall));

D0 = 0.05*PQ(1);
H = lpfilter('gaussian', PQ(1), PQ(2), D0);

F=fft2(double(footBall),size(H,1),size(H,2));
LPFS_football = H.*F;
LPF_football=real(ifft2(LPFS_football));
LPF_football=LPF_football(1:size(footBall,1), 1:size(footBall,2));

subplot(1,3,3), imshow(LPF_football, []), title('Blurred Image')
```

Explanation:

Gaussian with low pass filter used to make the image blur. Firstly, we convert the original image to grayscale using rgb2gray. Then determine the paddedsize for fourier transform based on the image used. Create a Gaussian lowpass filter 5% the width of the Fourier transform to image. Crop the image to undo padding and display the blurred image.

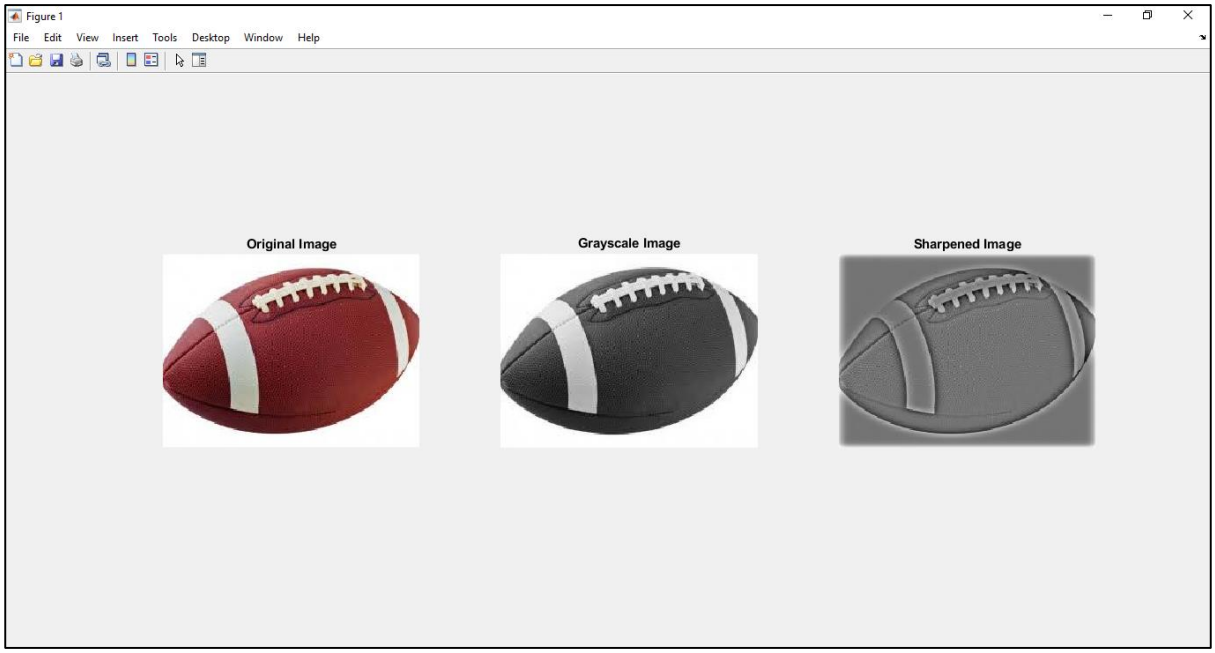



```
Fc=fftshift(F);
Fcf=fftshift(LPFS_football);
% use abs to compute the magnitude and use log to brighten display
S1=log(1+abs(Fc));
S2=log(1+abs(Fcf));
figure, imshow(S1,[])
figure, imshow(S2,[])
```

Explanation:

This represented display the Fourier spectrum. Fftshift(F) move the origin of the transform to the center of the frequency rectangle. After that use abs to compute the magnitude and use log to brighten the image displayed.

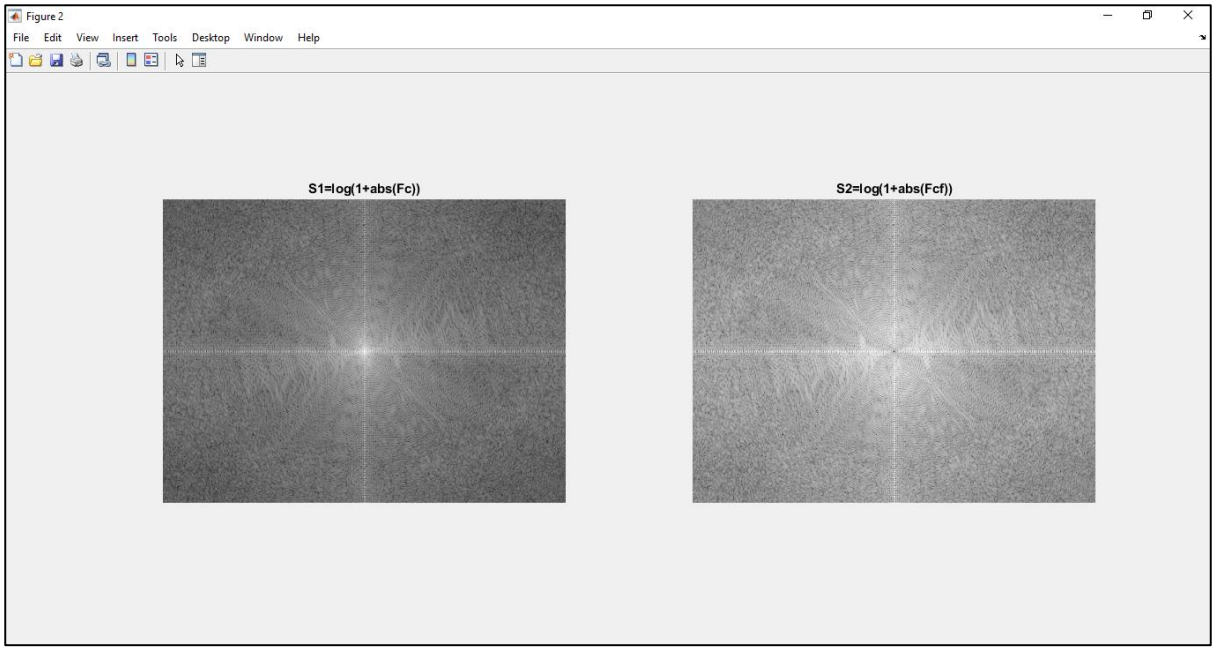
TUTORIAL 3: High Filter in Freq Domain




```
footBall=imread('football.jpg');
footBall=rgb2gray(footBall);
subplot(1,3,2), imshow(footBall), title('Grayscale Image');
PQ = paddedsize(size(footBall));

D0 = 0.05*PQ(1);
H = hpfilter('gaussian', PQ(1), PQ(2), D0);

F=fft2(double(footBall),size(H,1),size(H,2));
HPFS_football = H.*F;
HPF_football=real(ifft2(HPFS_football));
HPF_football=HPF_football(1:size(footBall,1), 1:size(footBall,2));

subplot(1,3,3), imshow(HPF_football, []), title('Sharpened Image')
```

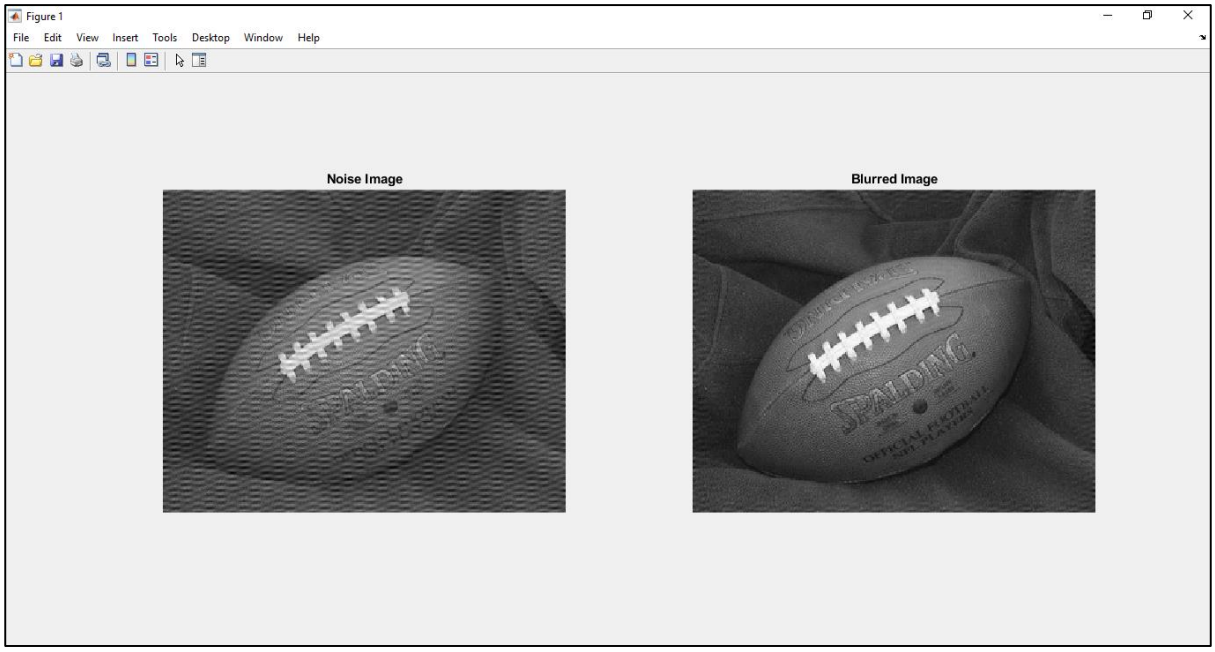
Explanation:

This operation is high pass filter and this script use gaussian highpass filter to sharpen the image. First, we need convert the image to grayscale image. Then determine a good padding to use the Fourier transform. As mentioned before, gaussian highpass filter is used with 5% the width of the Fourier transform. Then calculate the discrete Fourier transform of the image and apply the highpass filter to the fourier spectrum of the image. After all the operation we convert the result of image to the spatial domain and crop the image to undo padding. Sharpen image will be produce and displayed.




```
Fc=fftshift(F);
Fcf=fftshift(HPFS_football);
% use abs to compute the magnitude and use log to brighten display
S1=log(1+abs(Fc));
S2=log(1+abs(Fcf));
figure,
subplot(1,2,1), imshow(S1,[]), title('S1=log(1+abs(Fc))');
subplot(1,2,2), imshow(S2,[]), title('S2=log(1+abs(Fcf))');
```

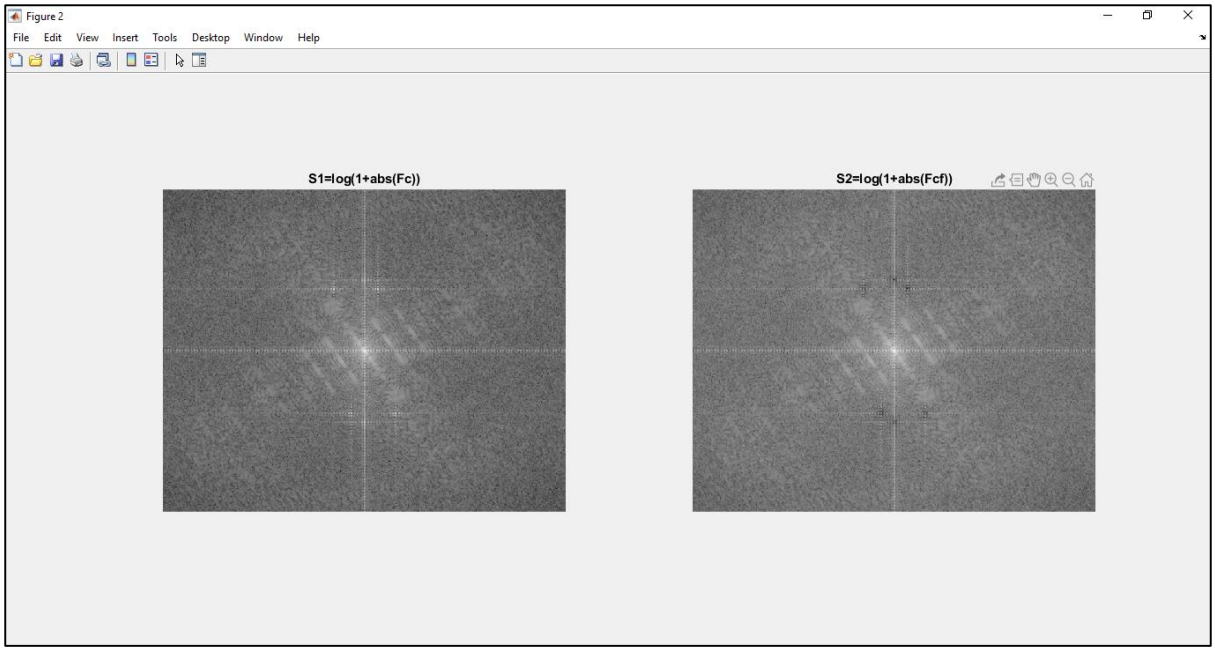
TUTORIAL 4: Notch Filter in Freq Domain




```
footBall=imread('noiseball.png');
subplot(1,2,1), imshow(footBall), title('Noise Image');

PQ = paddedsize(size(footBall));

H1 = notch('btw', PQ(1), PQ(2), 10, 50, 100);
H2 = notch('btw', PQ(1), PQ(2), 10, 1, 400);
H3 = notch('btw', PQ(1), PQ(2), 10, 620, 100);
H4 = notch('btw', PQ(1), PQ(2), 10, 22, 414);
H5 = notch('btw', PQ(1), PQ(2), 10, 592, 414);
H6 = notch('btw', PQ(1), PQ(2), 10, 1, 114);

F=fft2(double(footBall),PQ(1),PQ(2));
FS_football = F.*H1.*H2.*H3.*H4.*H5.*H6;
F_football=real(ifft2(FS_football));
F_football=F_football(1:size(footBall,1), 1:size(footBall,2));

subplot(1,2,2), imshow(F_football,[]), title('Blurred Image');
```

Explanation:

Notch filter being apply to make the image clear and brighter. Frequency domain can run a notch filter by determine the good padding for Fourier transform. Create a notch filters corresponding to extra peaks in the Fourier Transform. After that calculate the discrete Fourier transform of the image. Apply the notch filters to the Fourier spectrum of the image. Before crop the image to undo the padding, convert the result to the special domain. Then, display the blurred image.




```
% Display the Fourier Spectrum
% Move the origin of the transform to the center of the frequency
rectangle.
Fc=fftshift(F);
Fcf=fftshift(FS_football);

% use abs to compute the magnitude and use log to brighten display
S1=log(1+abs(Fc));
S2=log(1+abs(Fcf));
figure,
subplot(1,2,1), imshow(S1,[]), title('S1=log(1+abs(Fc))');
subplot(1,2,2), imshow(S2,[]), title('S2=log(1+abs(Fcf))');
```