SCHOOL OF COMPUTING

FACULTY OF ENGINEERING

SECV3213-02 ASAS PEMPROSESAN IMEJ (FUNDAMENTAL OF IMAGE PROCESSING)

SECTION 02

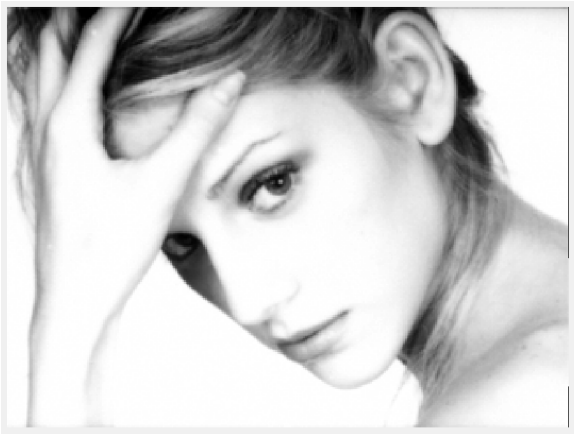
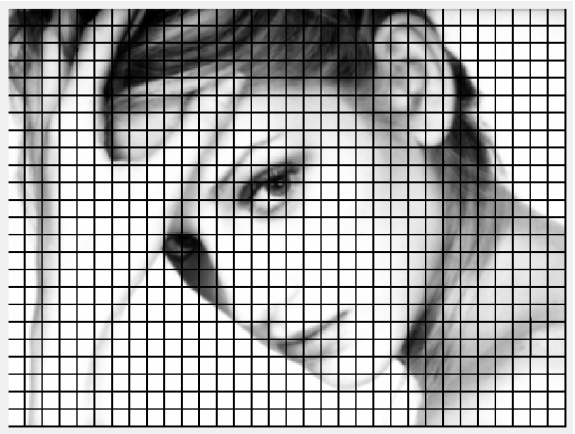
LAB AND TUTORIAL COMPILATION

Lecturer	**Dr. Md Sah bin Hj Salam**
Name	**Ng Jing Er**
Matric Number	**A19EC0115**

TABLE OF CONTENTS

LAB 1 : Understanding in Manipulation of Matrix

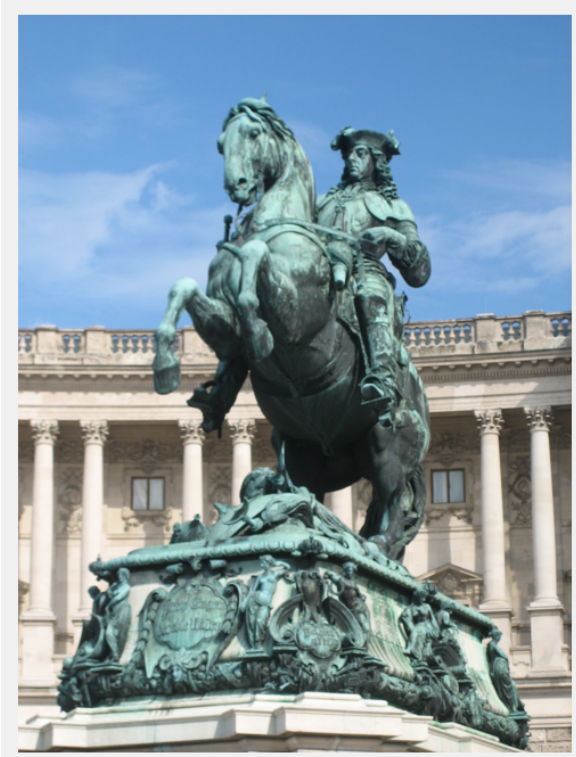
A. Tutorial 1 Gray Scale Image Behind Cage

Original Image	Output Behind Cage
	

Script

img = imread('lindsay.tif');
img(10:10:end,:,:) = 0;
img(:,10:10:end,:) = 0;
imshow(img);

B. Tutorial 2 Color Image Rotation

Original Image	Output After Cropping and Rotation
	

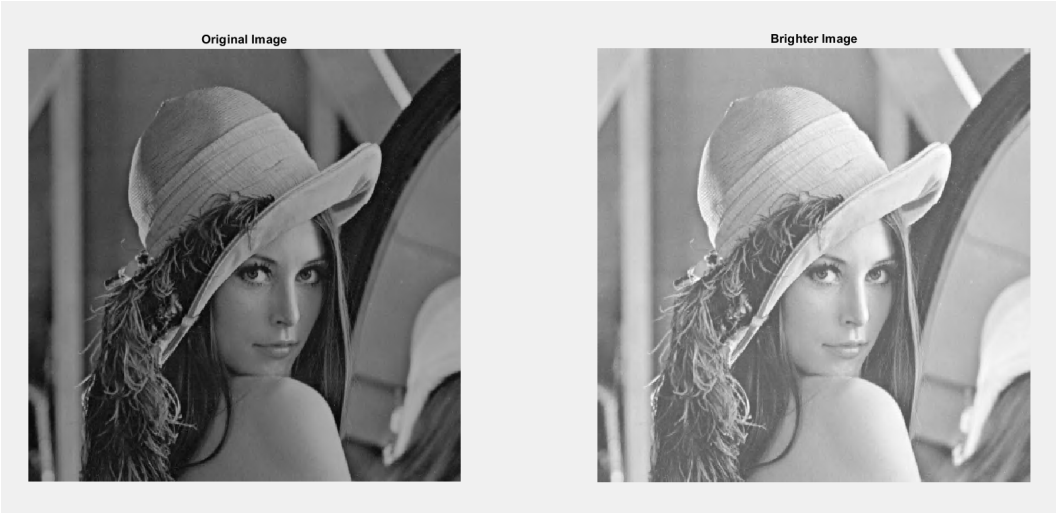
Script

```
I = imread('statue.png');

x=290;
y=110;

I2 = imcrop(I,[x y 30 30]);
RI2= imrotate( I2 , 180 );
I(y:y+size(RI2,1)-1, x:x+size(RI2,2)-1, :) = RI2;

imshow(I);
whos I
```

LAB 2 : Understanding in Arithmetic And Logic Operation

A. Tutorial 1 Addition

i. Brighten an image using imadd()

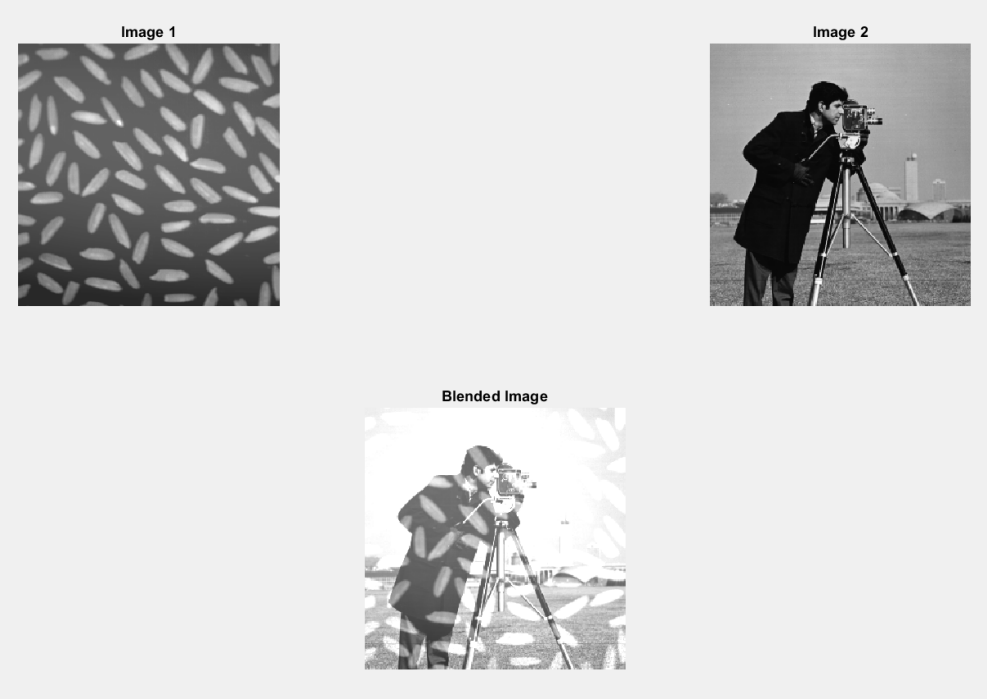
Example Output


Script

I = imread('lenna.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: The imadd() function has been used to add the number of 75 into the image to increase the intensity of the image, therefore the output of the image at the right becomes brighter.

ii. Blend 2 images

Example Output


Script

Ia = imread('rice.tiff');
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: In this tutorial, imadd() is used to combine two images. Ic = imadd(Ia, Ib) results in the image of Ib added into Ia and forms a combination of these images as shown in output of Blended Image.

B. Tutorial 2 Subtraction

i. Load two images and display them

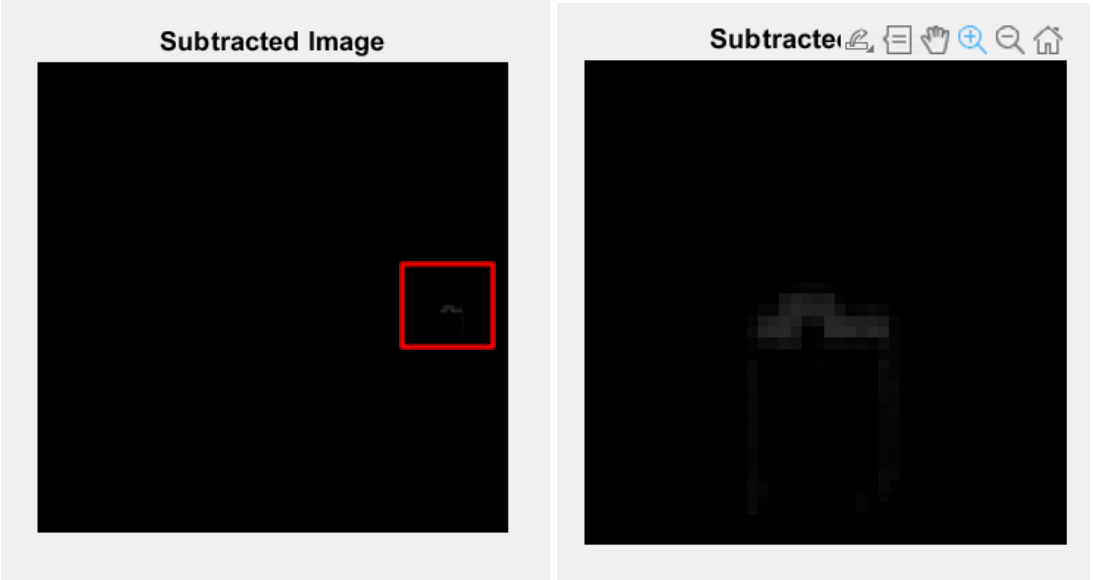
Example Output


Script

I = imread('Cameraman.tif');
J = imread('cameraman2.tif');
figure(1);
subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(J), title('Altered Image');

Explanation: imread() has been used to read the image from specified filename ‘Cameraman.tif’ and ‘cameraman2.tif’ then used the imshow() function to display the image in the figure.

ii. Subtract both image and display result

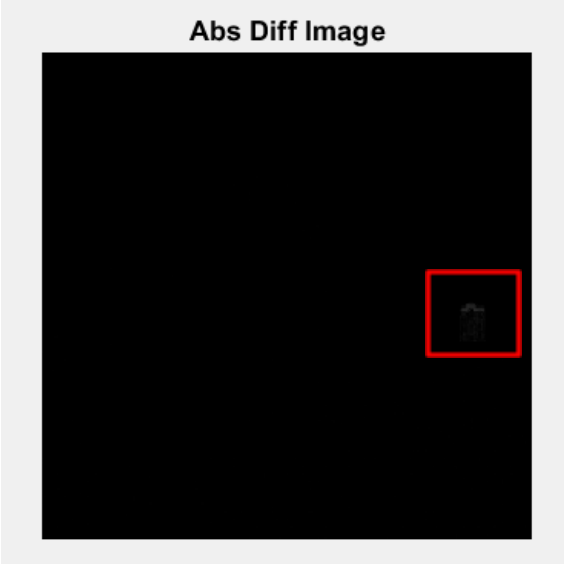
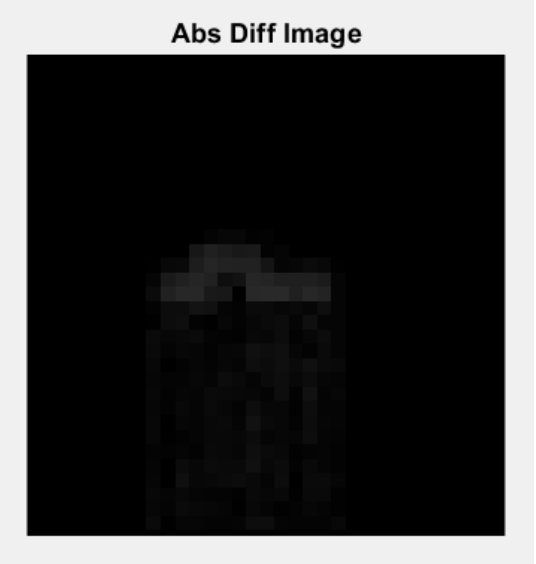
Example Output


Script

diffim = imsubtract(I,J);
figure(2);
subplot(2,2,1), imshow(diffim), title('Subtracted Image');

Explanation: imsubtract() function is used to subtract the image of J from I or subtract the constant from image and return the difference of image. Small region of pixels with faint white is spotted.

iii. Absolute diff function and display result

Example Output
 

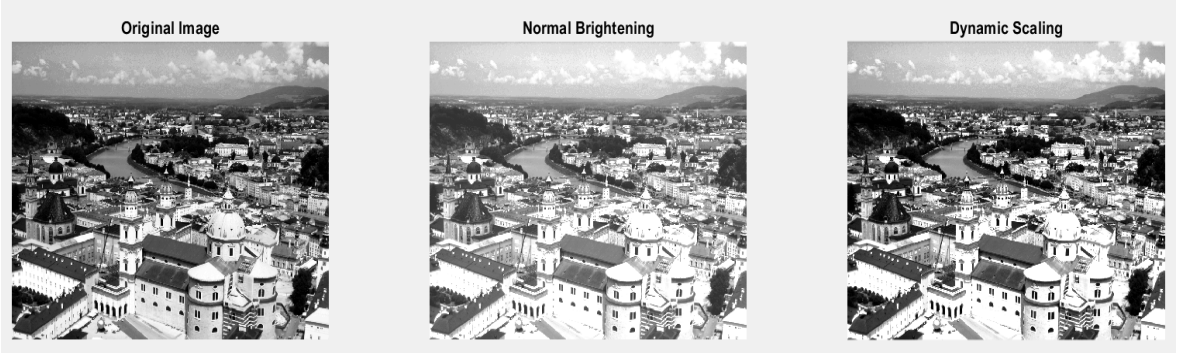
Script

diffim2 = imabsdiff(I,J);
figure(3);
subplot(2,2,2), imshow(diffim2), title('Abs Diff Image');

Explanation: Imabsdiff() is used to calculate and display the absolute difference between the images. Absolute value prevents negative values from being rounded to zero like subtract. Small region of pixels with faint white is spotted.

C. Tutorial 3 Multiply

i. Comparing between normal brightening and dynamic brightening using multiply operation

Example Output


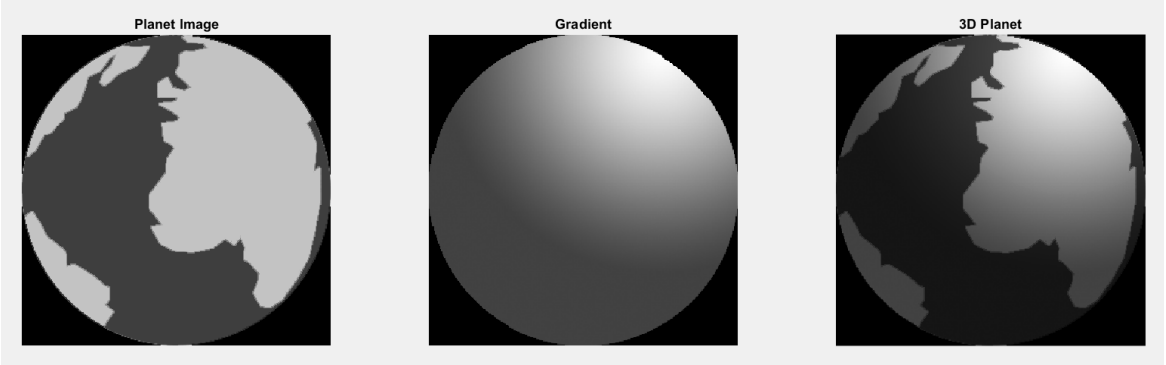
Script

I = imread('salzburg.png');
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() function is used to add two images or add constant value of 50 to image while immultiply() function is used to multiply a constant factor of 1.2 into each value of the image to increase the sharpness and brightness.

ii. Blend two images using multiply operation.

Example Output



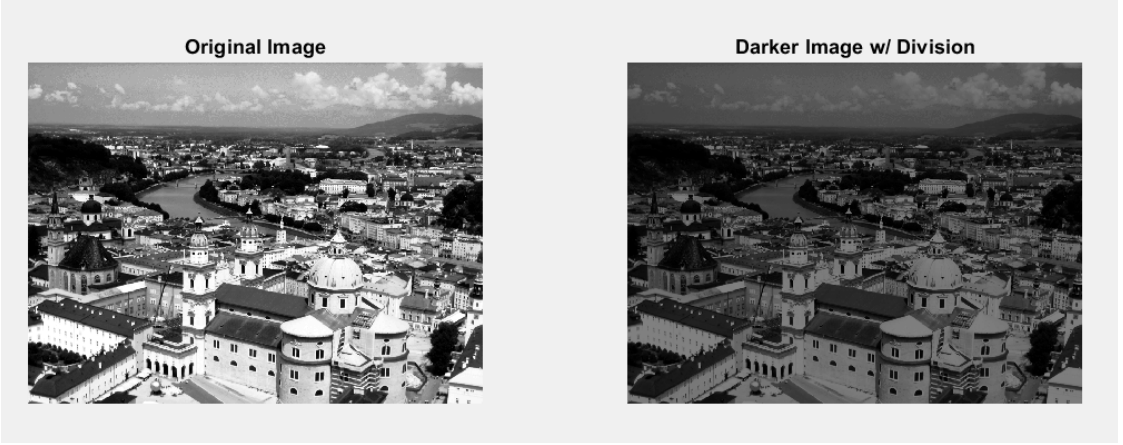
Script

```
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:
im2double() converts the image I to double precision.then, Immultiply() function is used to blend two images like additional operation. As shown in the output above, the images blend together thus producing the blended image in the third output.

D. Tutorial 4 Multiply and Division

i. Use division operation to dynamically darken the image

Example Output


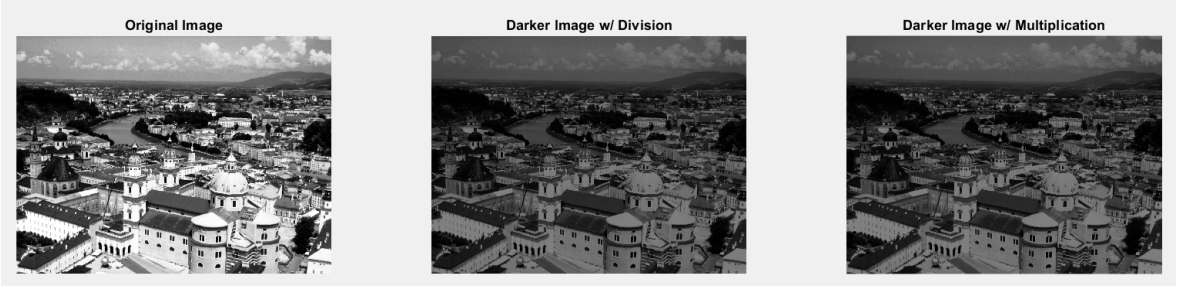
Script

I = imread('salzburg.png');
I2 = imdivide(I,2);
 	
figure
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I2), title('Darker Image w/ Division');

Explanation: imdivide() function is used to divide each value of image by a constant value factor of 2. The image becomes darker compared with the original colour as the value of the image is reduced.

ii. Get the same result using multiply

Example Output



Script

```
I = imread('salzburg.png');
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('Darker Image w/ Division');
subplot(1,3,3), imshow(I3), title('Darker Image w/ Multiplication');
```

Explanation:
imdivide() function will divide each value of the image by a constant value factor of 2. Another approach to get the same result of a darkened image is using immultiply() where immultiply() function is used to multiply the image by a constant factor of 0.5 into the each value of image.

E. Tutorial 5 Logic Operation

i. Use bitxor op to find the different of two images

Example Output

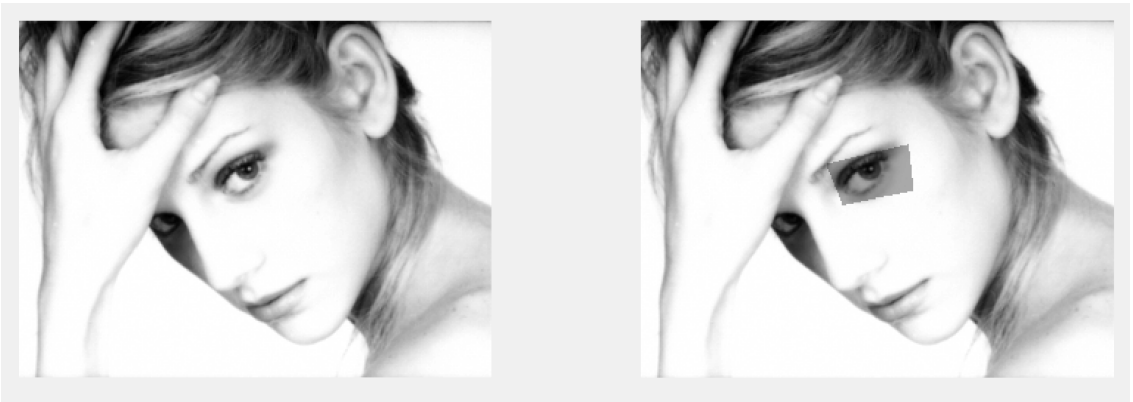

Script

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) will return the bitwise XOR operation of the images to find the difference between the images used. As shown in the output, the difference shown where the building appeared in Image 1 which is not in the Image 2.

ii. Make lindsay eyes dark

Example Output



Script

```
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() function has been used to divide each value of image by constant value factor of 1.5. roipoly() function is used to create an interactive polygon tool associated with the image displayed in the application. This function allows the user to select a polygonal region of interest within the image and make it darker.

F. Test Understanding

Example Output

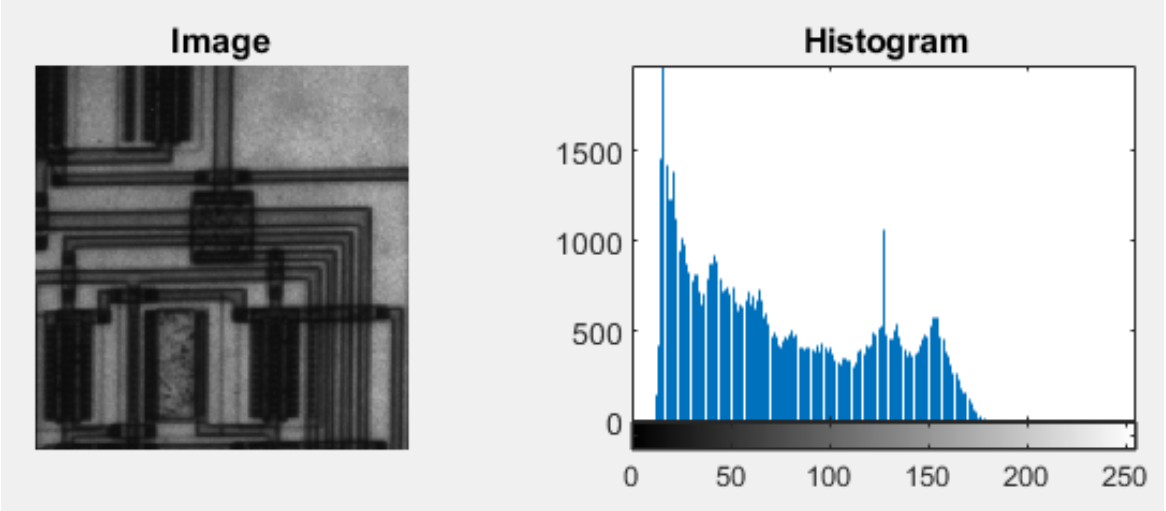

Script

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 3 : Understanding in Histogram

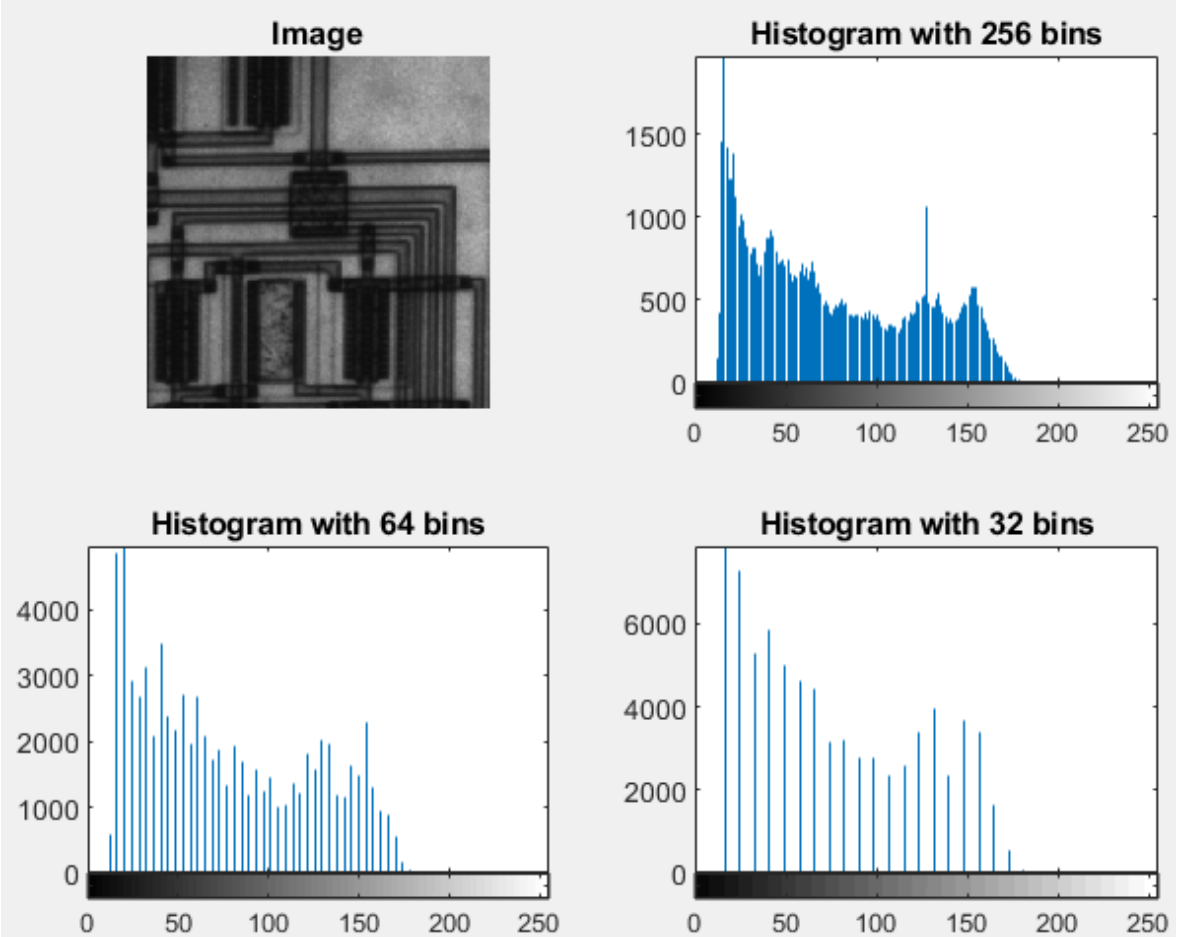
A. Tutorial 6 Displaying Histogram

i. Displaying an image and its histogram using 256 bins

Example Output


Script
I = imread('circuit.tif'); figure, subplot(2,2,1), imshow(I), title('Image') subplot(2,2,2), imhist(I,256),axis tight, title('Histogram')

ii. Using different bins

Example Output


Script
I = imread('circuit.tif'); 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')
Explanation: Based on the image selected, the image's intensity has a range of 0-255. The imhist function is used to create a histogram plot by defining 256,64, 32 equally spaced bins, each representing a range of data values. The number of pixels within each range is calculated.

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

Example Output

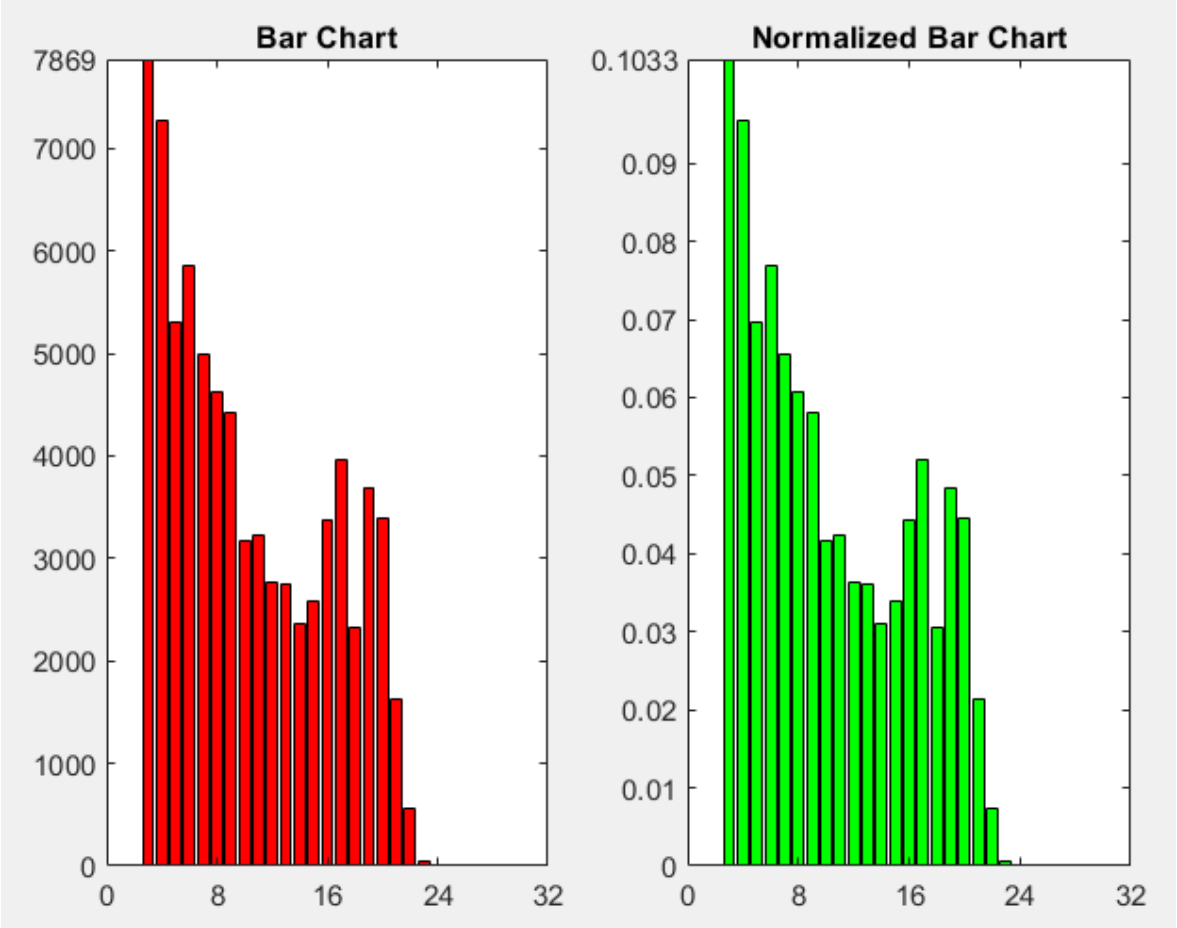
```
c =
 0      15     7869   7271   5305   5852   4983   4630   4413   3173
 3321   2765   2756   2357   2575   3380   3971   2323   3691   3385
 1621   553    48     3      0      0      0      0      0      0   0   0

c_norm =
 0         0.0002   0.1033   0.0955   0.0697   0.0768   0.0654   0.0608   0.0579   0.0417
 0.0423    0.0363   0.0362   0.0309   0.0338   0.0444   0.0521   0.0305   0.0485   0.0444
 0.0213    0.0073   0.0006   0.0000   0        0        0        0        0        0   0   0
```

Script

```
c = imhist(I,32);
c_norm = c / numel(I);
```

iv. Displaying using bar chart

Example Output

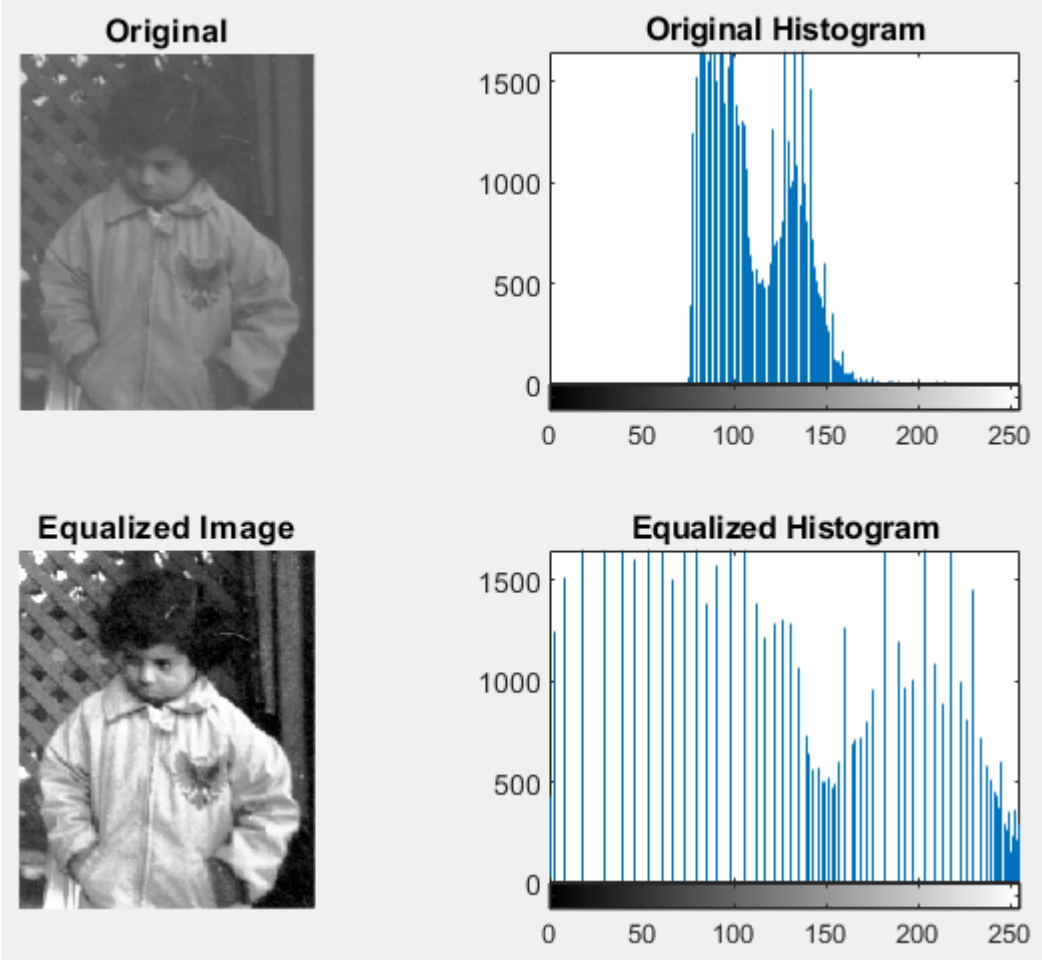


Script

```
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')
```

B. Tutorial 7 Hist Equalization

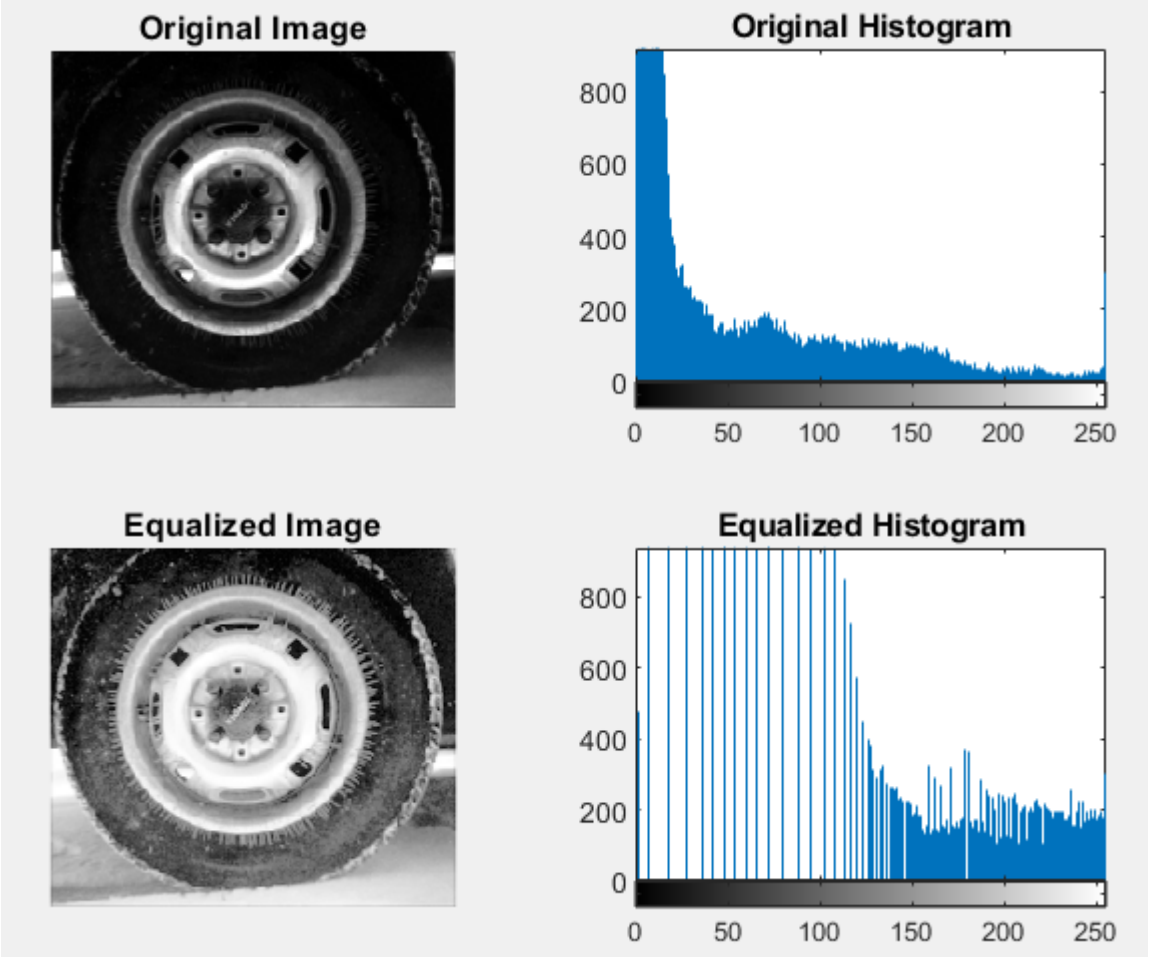
i. Image histogram equalization 1

Example Output


Script
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')
Explanation: histeq() function will enhance the contrast of images by transforming the values in an intensity image where it will change the contrast of the image by changing the intensity distribution of the histogram.

ii. Image histogram equalization 2

Example Output



Script

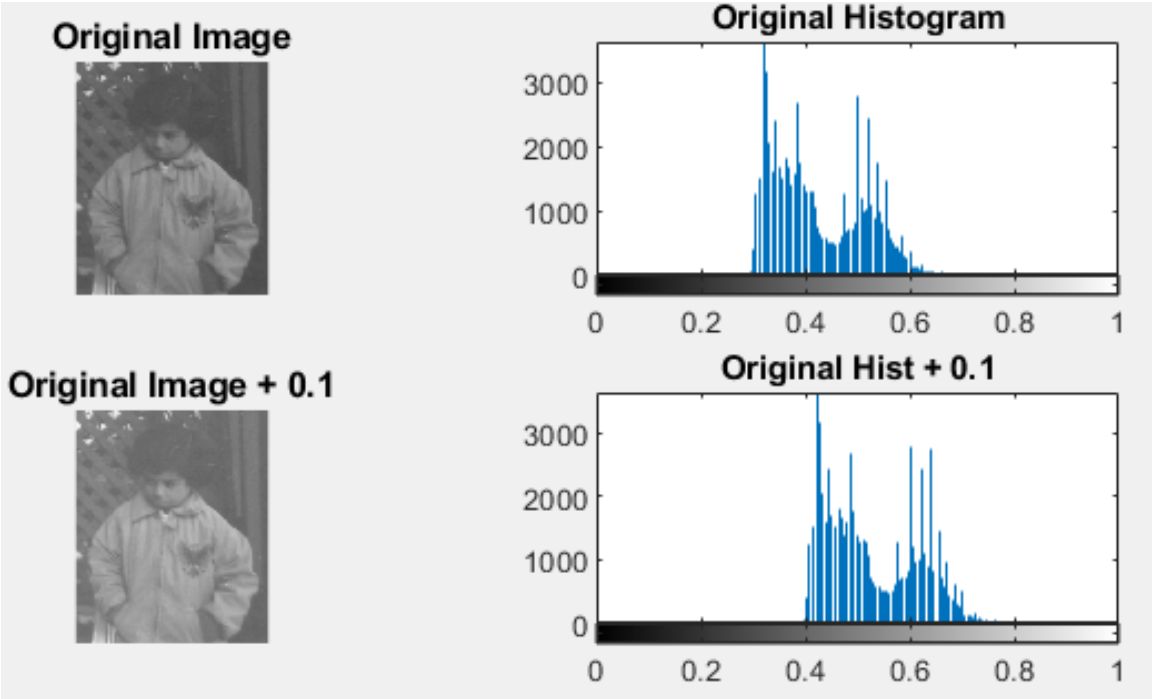
```
I=imread('tire.tif');
l_eq=histeq(I,256);
figure, subplot(2,2,1),imshow(I),title('Original Image')
subplot(2,2,2), imhist(I), title('Original Histogram')
subplot(2,2,3),imshow(l_eq),title('Equalized Image')
subplot(2,2,4),imhist(l_eq),title('Equalized Histogram')
```

Explanation:
As shown in the output above, the equalized image of ‘tire.tif’ did not show a good image and it is not suitable to apply histeq compared with ‘pout.tif’. This is because the image of the tire has an intensity range of 0-255 and the value of grayscale image to use intensity 256 is not suitable as it has higher value for the original image where imhist(I,256) is default.

C. Tutorial 8 Hist Modification

i. Addition / sliding in histogram

Example Output



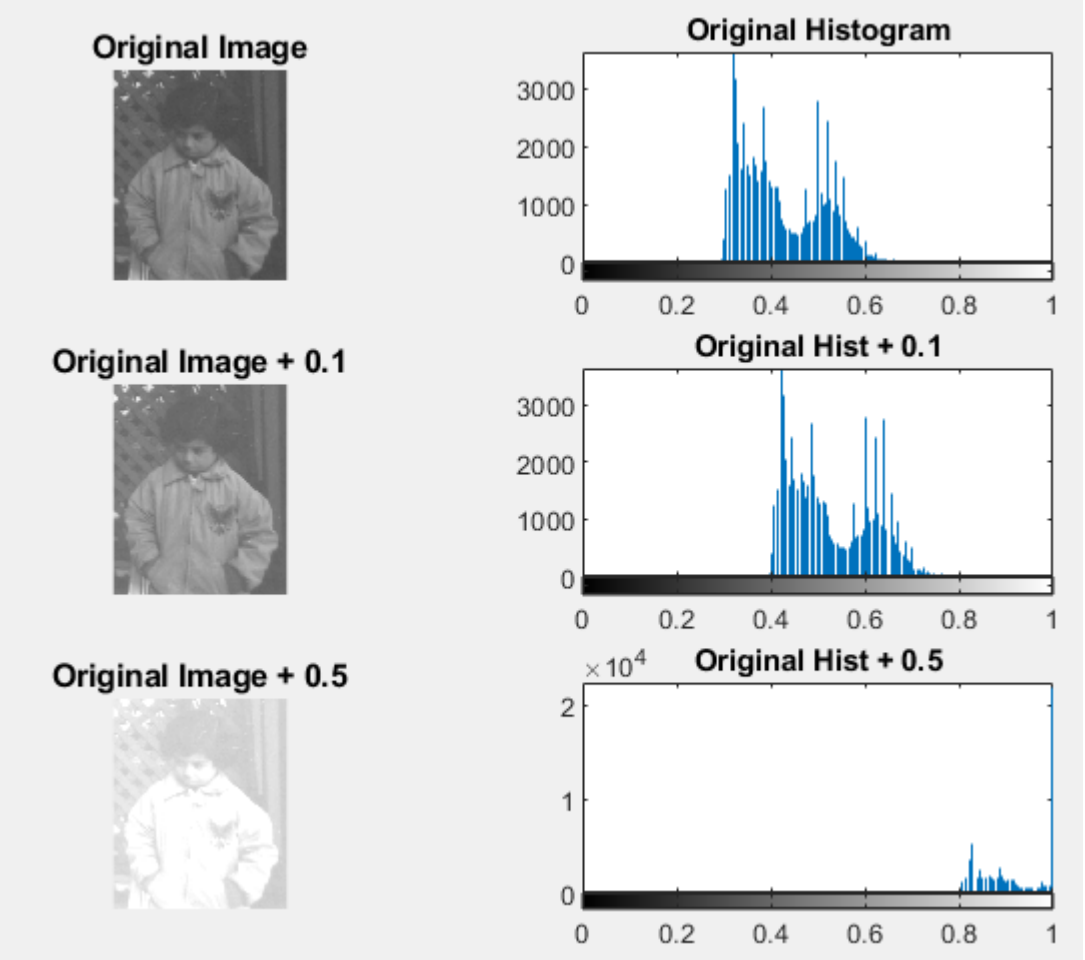
Script

```
J = imread('pout.tif');
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')
```

D. Tutorial 9 Hist Modification

i. Histogram sliding

Example Output



Script

```
J = imread('pout.tif');
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')
```

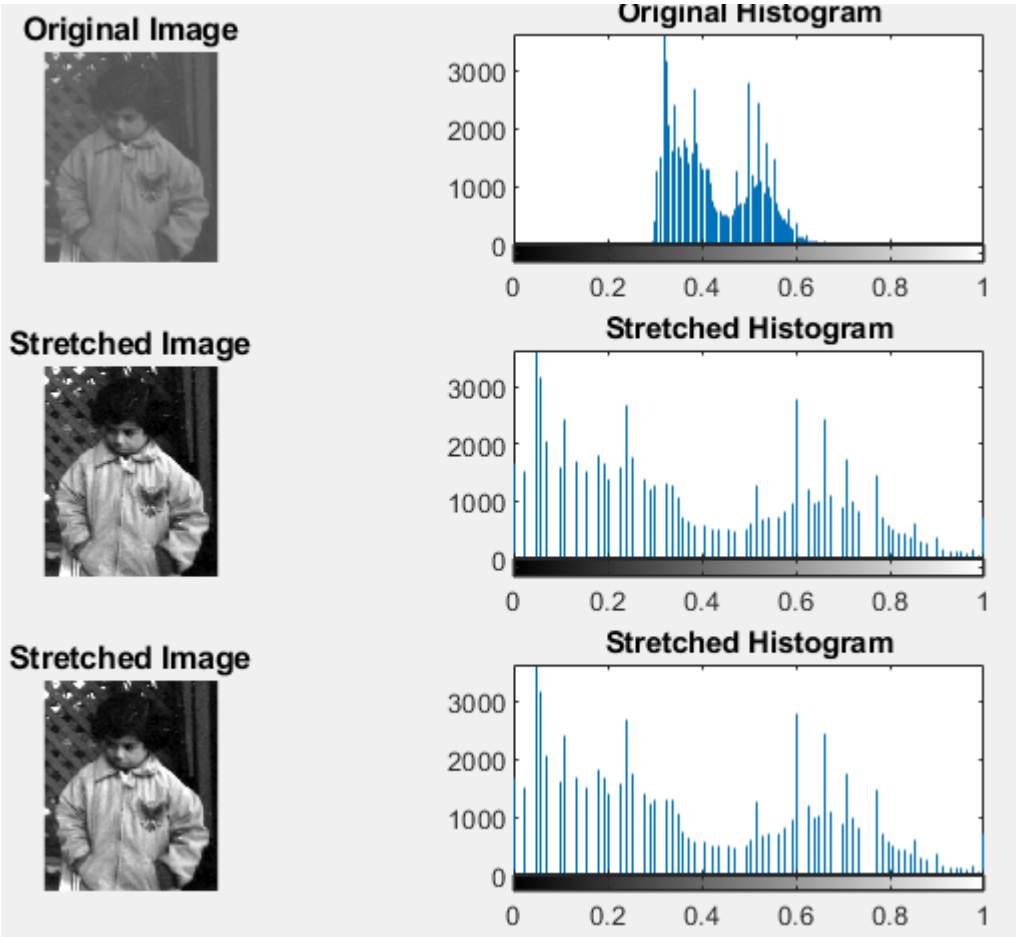
```
const = 0.5;
I3 = I + const;
bad_values = find(I3 > 1);
I3(bad_values) = 1;
subplot(3,2,5), imshow(I3), title('Original Image + 0.5')
subplot(3,2,6), imhist(I3), axis tight, title('Original Hist + 0.5')
```

Explanation:
In this tutorial, histogram sliding is used to manipulate the brightness of an image by adding a constant value 0.1 and 0.5. The complete histogram is simply shifted towards rightwards which shows an increase of the brightness of the image.

E. Tutorial 10 Hist Modification

i. Histogram stretching

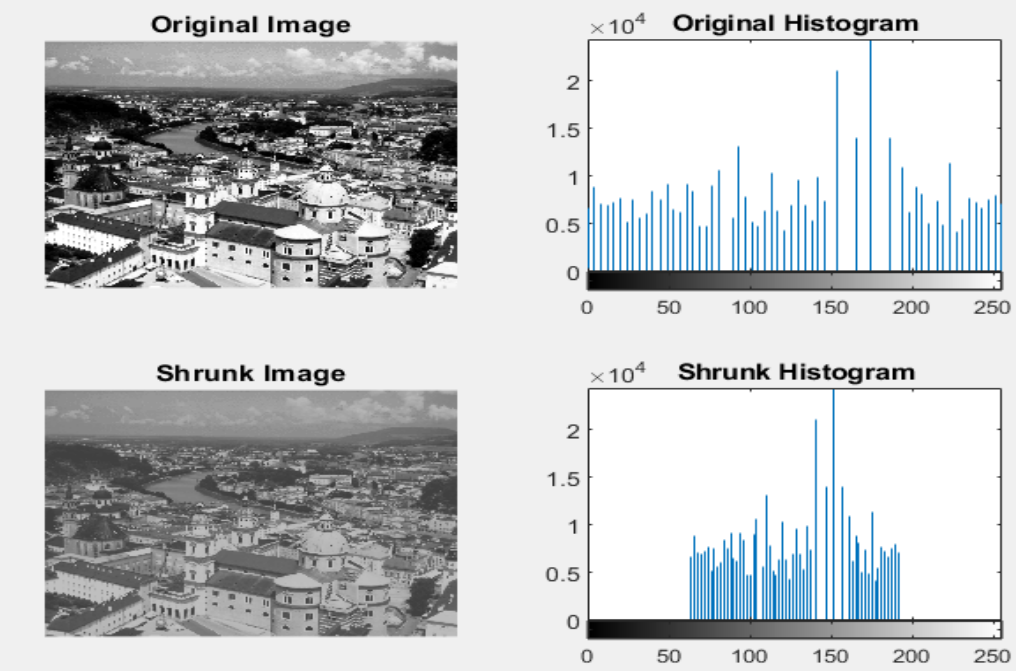
Example Output

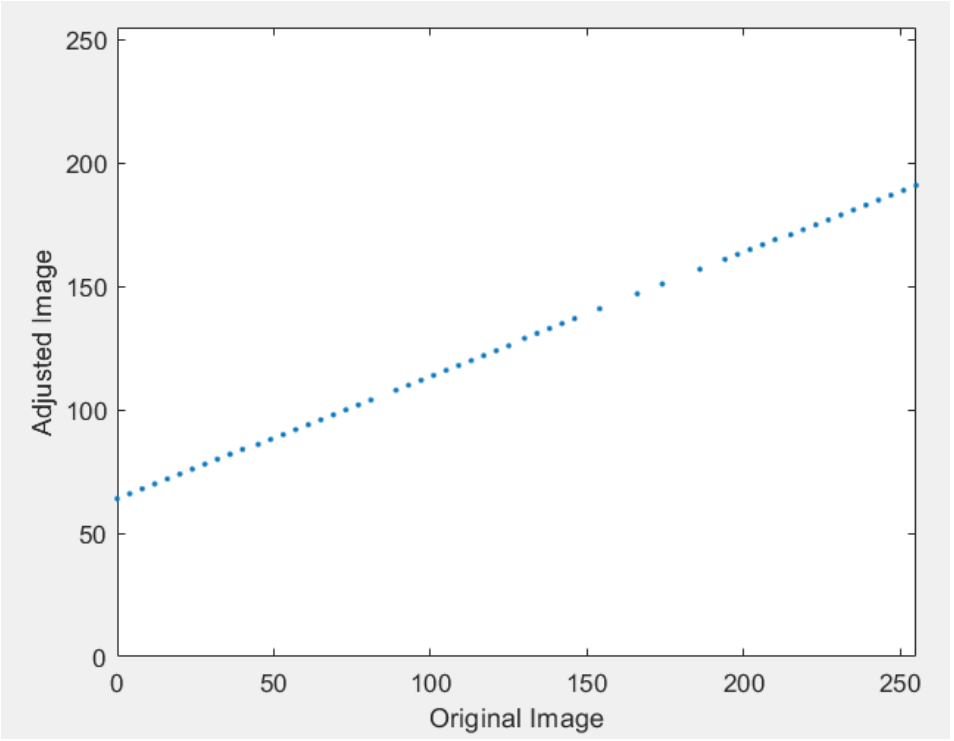


Script
img_limits = stretchlim(I); I_stretch = imadjust(I,img_limits,[]); figure; subplot(3,2,1), imshow(I), title('Original Image') subplot(3,2,2), imhist(I), axis tight, title('Original Histogram') subplot(3,2,3), imshow(I_stretch), title('Stretched Image') subplot(3,2,4), imhist(I_stretch), axis tight, 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(:))
Explanation: stretchlim is used to compute the lower and upper limits that can be used for contrast stretching grayscale image I. And the limits are returned in img_limits. I_stretch = imadjust(img_limits) is used to map the intensity values in grayscale image 'pout.tif' to new values in I_stretch. This operation increases the contrast of the output image.

F. Tutorial 11 Hist Modification

i. Histogram shrinking

Example Output




Script

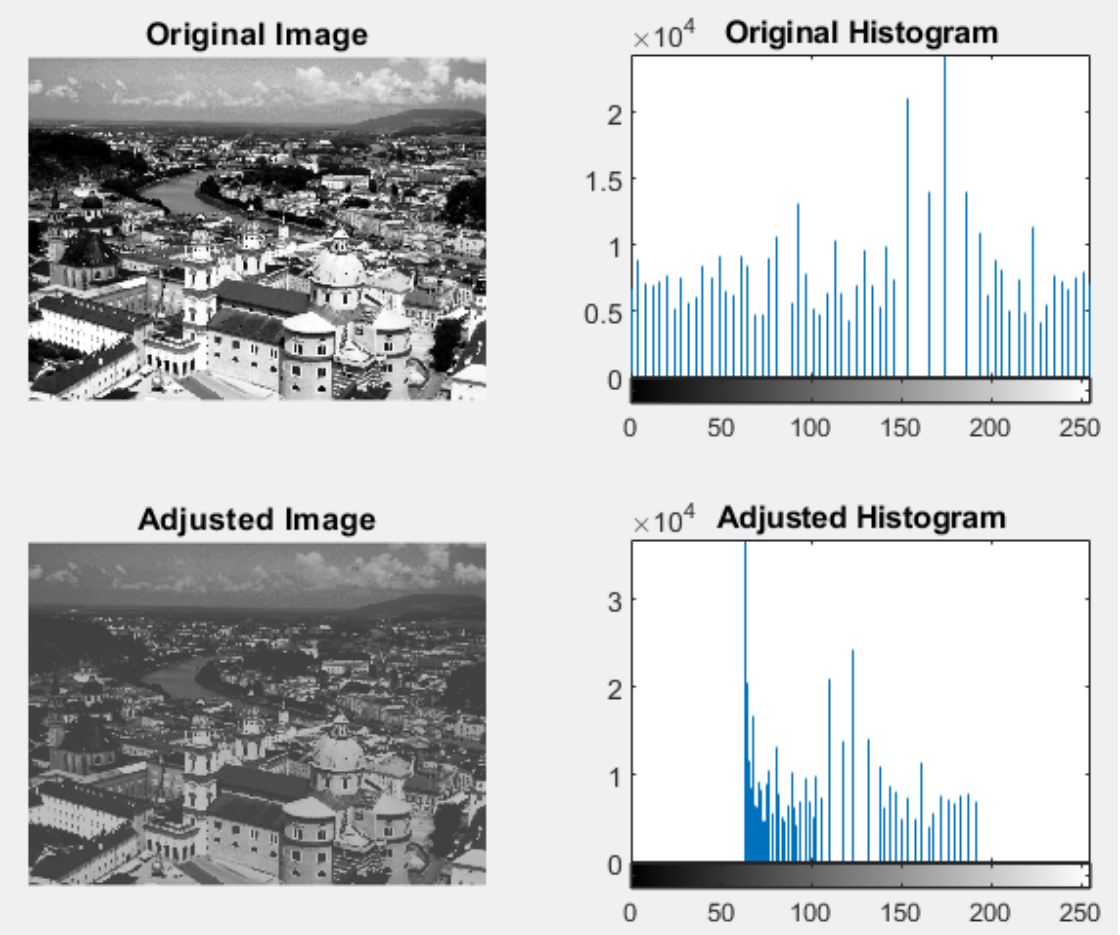
```
I = imread('salzburg.png');
I_shrink = imadjust(I,stretchlim(I),[0.25 0.75]);
figure;
subplot(2,2,1), imshow(I), title('Original Image')
subplot(2,2,2), imhist(I), axis tight, title('Original Histogram')
subplot(2,2,3), imshow(I_shrink), title('Shrunk Image')
subplot(2,2,4), imhist(I_shrink), axis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

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');
```

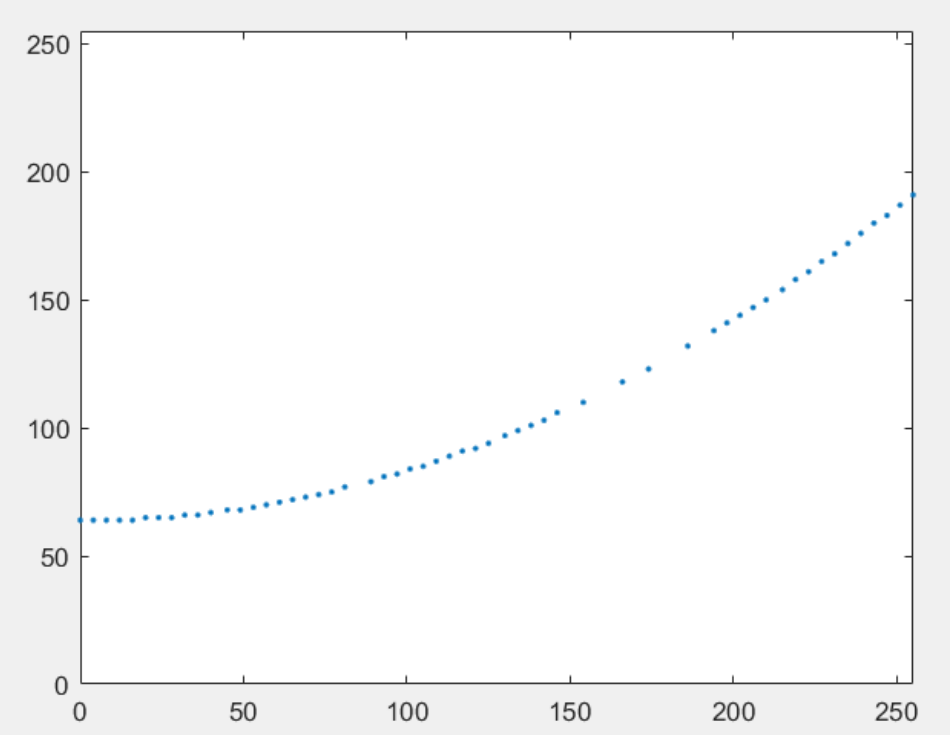
Explanation:
This tutorial is a continuation form the previous tutorial and histogram shrinking has been used to decrease the contrast of the image by compressing the gray levels of the image.

G. Tutorial 12 Hist Modification

i. Histogram shrinking with gamma value

Example Output





Script

```
I = imread('salzburg.png');

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 4 : Understanding in Filtering

A. Tutorial 1 Smoothing filter

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

Example Output


Script

I = imread('cameraman.tif');
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');
 	
fn2 = [1 2 1; 2 4 2; 1 2 1]
fn2 = fn2 * (1/16)
I_new2 = imfilter(I,fn2);
figure, subplot(1,2,1), imshow(I_new), title('Uniform Average');
subplot(1,2,2), imshow(I_new2), title('Non-uniform Average');

Explanation: fspecial() is used to creates a two-dimensional filter h of the specified type. It will return an average filter to the image. While imfilter() is used as an appropriate form to return the fspecial filtering type.

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

Example Output

Script

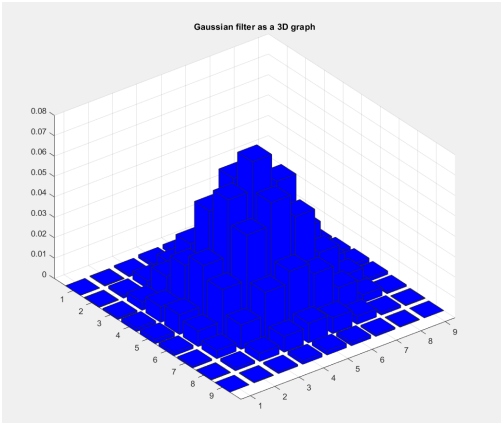
```
I = imread('cameraman.tif');
figure, subplot(1,2,1), imshow(I), title('Original Image');
fn = fspecial('average')
I_new = imfilter(I,fn);
subplot(1,2,2), imshow(I_new), title('Filtered Image');

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

Explanation:
An uniform filter is generated using a mean/average filter while non-uniform average is being applied with coefficient value to the image.

B. Tutorial 2 Smoothing filter 2

i. create and apply Gaussian filter

Example Output



Script

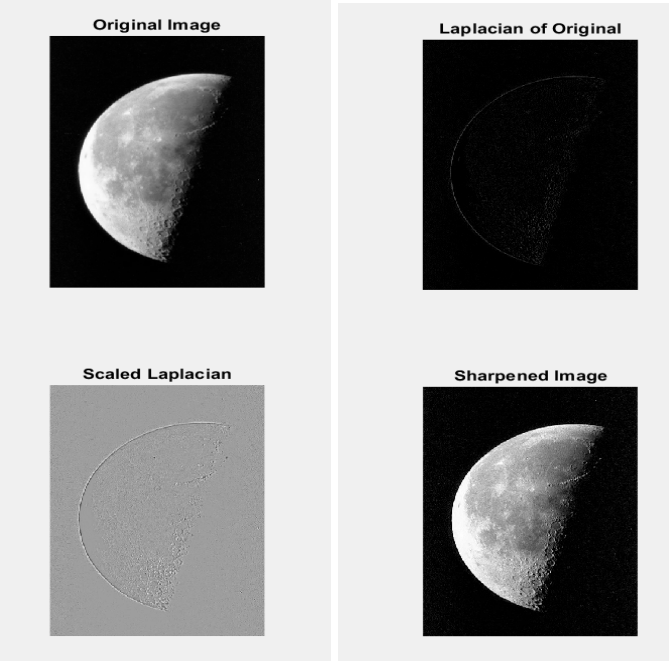
```
I = imread('cameraman.tif');
% create and show in bar graph
fn = fspecial('average');
fn_gau = fspecial('gaussian',9,1.5);
figure, bar3(fn_gau,'b'), title('Gaussian filter as a 3D graph');
% apply on the cameran image
I_new = imfilter(I,fn);
I_new3 = imfilter(I,fn_gau);
figure
subplot(1,3,1), imshow(I), 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 used for smoothing images. A Gaussian filter is a filter whose impulse response is a Gaussian function. Gaussian can be used to remove noise and detail in the image while the Gaussian 3D graph indicate the minimization of image noise in the filter

C. Tutorial 3 Sharpening filter 1

i. Create and use laplacian filter

Example Output



Script

```
% load the image moon
I = imread('moon.tif');
Id = im2double(I);
figure, subplot(2,2,1), imshow(Id), title('Original Image');
% create laplacian filter
f = fspecial('laplacian',0);
I_filt = imfilter(Id,f);
subplot(2,2,2), imshow(I_filt), title('Laplacian of Original');
% Display a scaled version of the Laplacian image for display purposes.
subplot(2,2,3), imshow(I_filt,[]), title('Scaled Laplacian');
% Subtract the filtered image from the original image to create the
% sharpened image.
```

```
I_sharp = imsubtract(Id,I_filt);
subplot(2,2,4), imshow(I_sharp), title('Sharpened Image');
```

Explanation:
A Laplacian filter is an edge detector used to compute the second derivatives of an image. In this tutorial, Laplacian filters are used on pictures to reduce their sensitivity to noise. It focuses on sharpening pictures.Scaled laplacian is a method that uses filter results from laplacian to image. For Sharpened image, an imsubtract of image is used to produce the output as above.

D. Tutorial 4 Sharpening filter 2

i. Another way in using laplacian filter – composite mask

Example Output

Script

```
% This script assume the variable from previous tutorial is active
I = imread('moon.tif');
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 using laplacian using a composite mask by defining the value for f2.

E. Tutorial 5 Sharpening filter 3

i. Another way in using laplacian filter on blur image

Example Output

Script

```
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');
% shrink the histogram of the blur image
I_blur_adj = imadjust(I_blur,stretchlim(I_blur),[0 0.4]);
% Now subtract the blurred image from the original image
I_sharp = imsubtract(I,I_blur_adj);
% Stretch the sharpened image histogram to the full dynamic grayscale
% range and display the final result.
I_sharp_adj = imadjust(I_sharp);
subplot(1,3,3), imshow(I_sharp_adj), title('Sharp Image');
```

Explanation:
In this tutorial, Sharpening filter method is different by applying laplacian filters on blur images. Average filter has been used by initialising the value of hsize with 13. The filter being applied on the image then uses imadjust for intensity values in grayscale image. Imadjust has been used to shrink the histogram of the blur image which subtracted the blurred image from the original image. Then, the sharpened image histogram is stretched to the full dynamic grayscale.

F. Tutorial 6 and Tutorial 7

i. Another way in using laplacian filter on blur image

Example Output

Script

```
% Subtract the blurred image from the original image to generate a sharpening 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:
In this tutorial, the method is applying lapacity filters on blurred images by subtracting blurred images from the original image to produce a sharpened image. The sharp image is added to the original image to produce the output as shown above.

G. Tutorial 8

i. Sample median filter usage for noise restoration

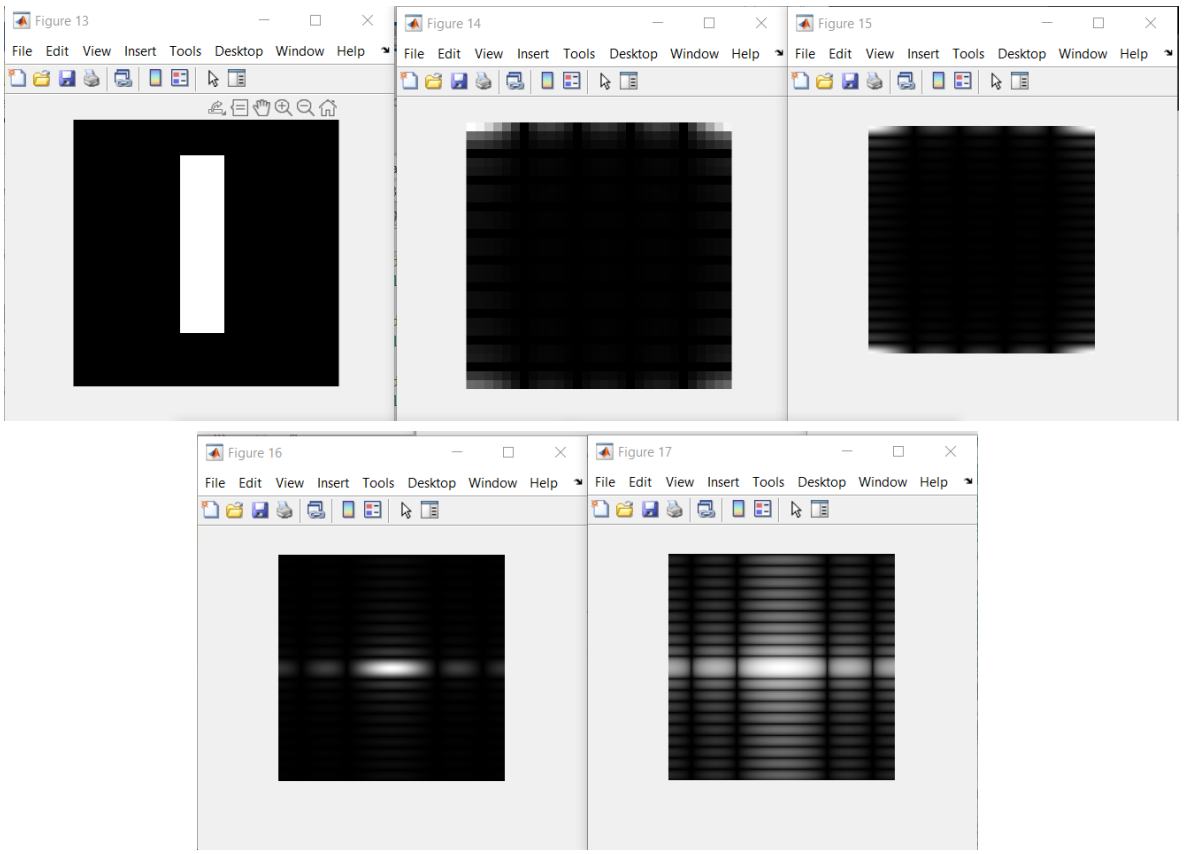
Example Output
 

Script
I = imread('eight.tif'); J = imnoise(I,'salt & pepper',0.02); K = medfilt2(J); imshow(J), figure, imshow(K)
Explanation: The salt and pepper noise is added with neighborhood size as shown is output at left. And the median filter is used to filter out the noise from the image.

LAB 5 : Understanding in Frequency Domain Filter

A. Tutorial 1 How to Display a Fourier Spectrum using MATLAB

Example Output



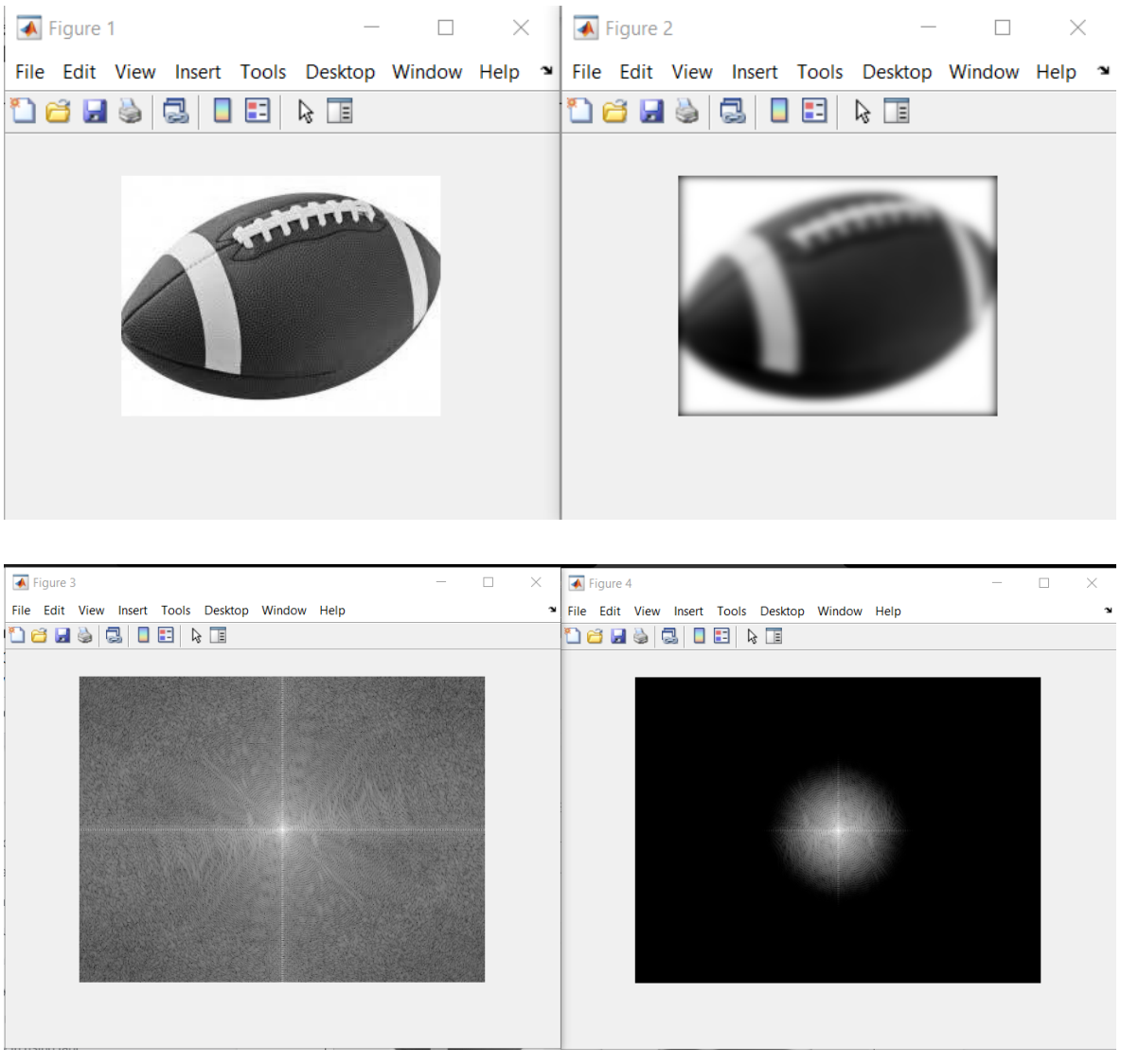
Script

```
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,[])
```

B. Tutorial 2 Low Pass Filter in Freq Domain

Example Output



Script

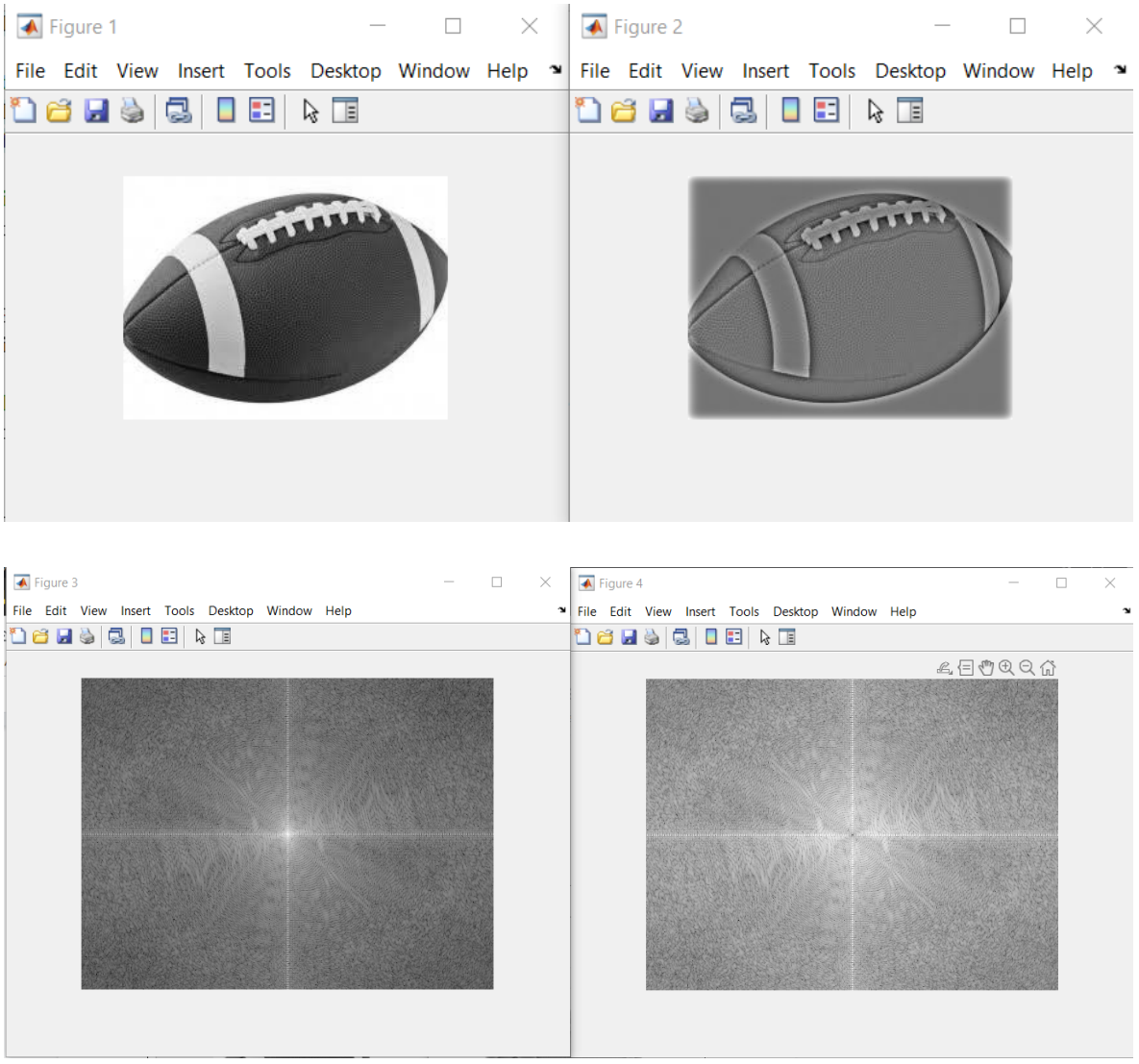
```
footBall=imread('football.jpg');
%Convert to grayscale
footBall=rgb2gray(footBall);
imshow(footBall)

%Determine good padding for Fourier transform
PQ = paddedsize(size(footBall));

%Create a Gaussian Lowpass filter 5% the width of the Fourier transform
D0 = 0.05*PQ(1);
H = lpfilter('gaussian', PQ(1), PQ(2), D0);
% Calculate the discrete Fourier transform of the image
F=fft2(double(footBall),size(H,1),size(H,2));
% Apply the highpass filter to the Fourier spectrum of the image
LPFS_football = H.*F;

% convert the result to the spacial domain.
LPF_football=real(ifft2(LPFS_football));
% Crop the image to undo padding
LPF_football=LPF_football(1:size(footBall,1), 1:size(footBall,2));
%Display the blurred image
figure, imshow(LPF_football, [])

% Display the Fourier Spectrum
% Move the origin of the transform to the center of the frequency rectangle.
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:
In this tutorial, Gaussian with a low pass filter is also used to make the image blur. The scripts are as follow :

1. The original image is converted to grayscale using rgb2gray.
2. The paddedsize is determined for fourier transform based on the example of the image used.
3. A Gaussian lowpass filter is created using lpfilter() function, where it is 5% the width of the Fourier transform to image.The image is cropped to undo padding and display the blurred image.
4. The Fourier spectrum is displayed. fftshift() is used to move the origin of the transform to the center of the frequency rectangle. The abs() is then used to compute the magnitude and the log is used to brighten the image displayed.

C. Tutorial 3 High Filter in Freq Domain

Example Output



Script

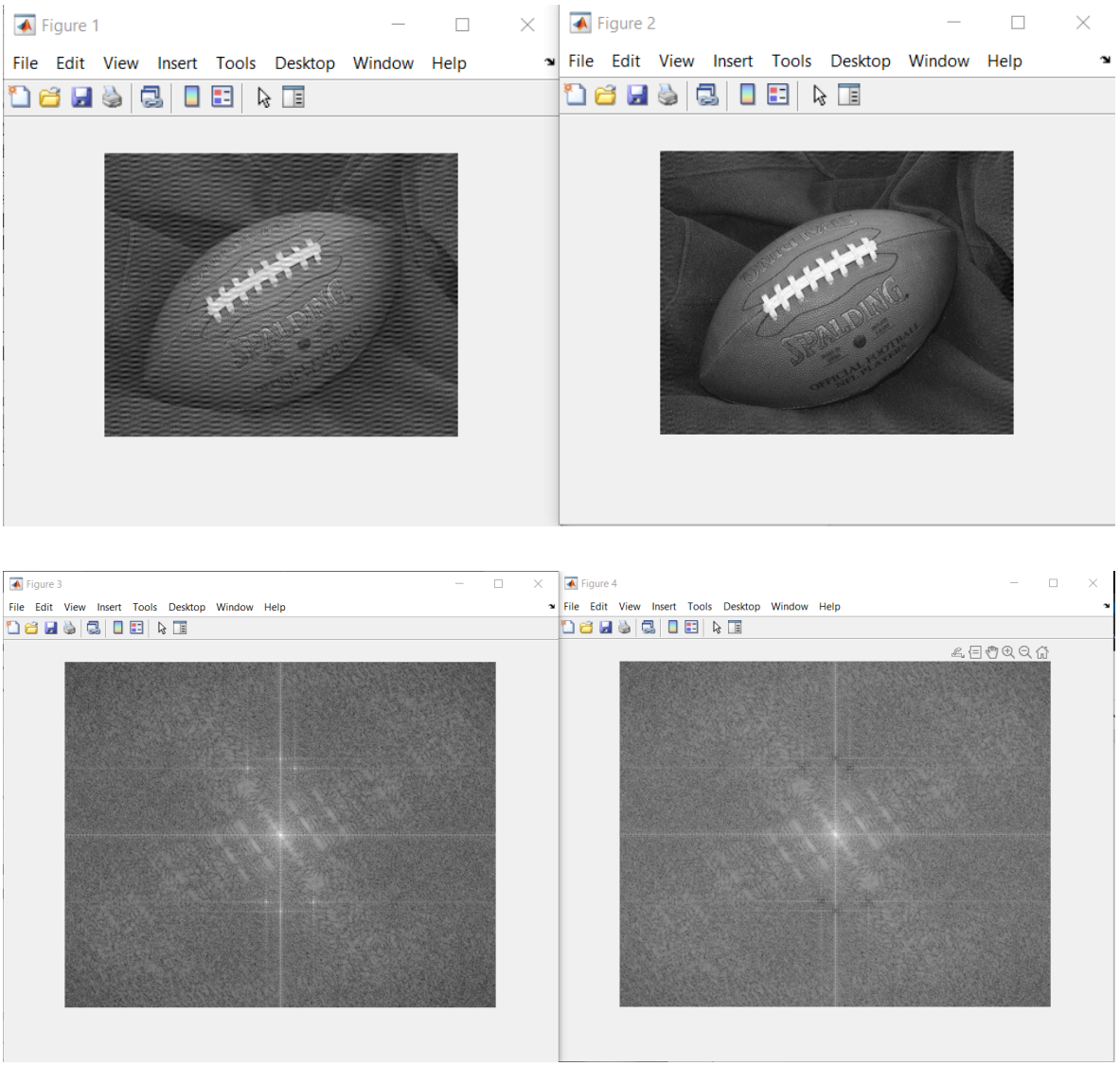
```
footBall=imread('football.jpg');

%Convert to grayscale
footBall=rgb2gray(footBall);
imshow(footBall)

%Determine good padding for Fourier transform
PQ = paddedsize(size(footBall));

%Create a Gaussian Highpass filter 5% the width of the Fourier transform
```

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

% Calculate the discrete Fourier transform of the image
F=fft2(double(footBall),size(H,1),size(H,2));

% Apply the highpass filter to the Fourier spectrum of the image
HPFS_football = H.*F;

% convert the result to the spacial domain.
HPF_football=real(ifft2(HPFS_football));

% Crop the image to undo padding
HPF_football=HPF_football(1:size(footBall,1), 1:size(footBall,2));

%Display the "Sharpened" image
figure, imshow(HPF_football, [])

% Display the Fourier Spectrum
% Move the origin of the transform to the center of the frequency rectangle.
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, imshow(S1,[])
figure, imshow(S2,[])
```

Explanation:
This tutorial involves a high pass filter. Based on the code above, the Gaussian high pass filter is used to sharpen the image.The steps are similar with low pass where hpfilter() is used instead of lpfilter(). By using a high pass filter, the output of the image displayed will be sharpened instead of the low pass filter which is used to smoothen the image.

D. Tutorial 4 Notch Filter in Freq Domain

Example Output



Script

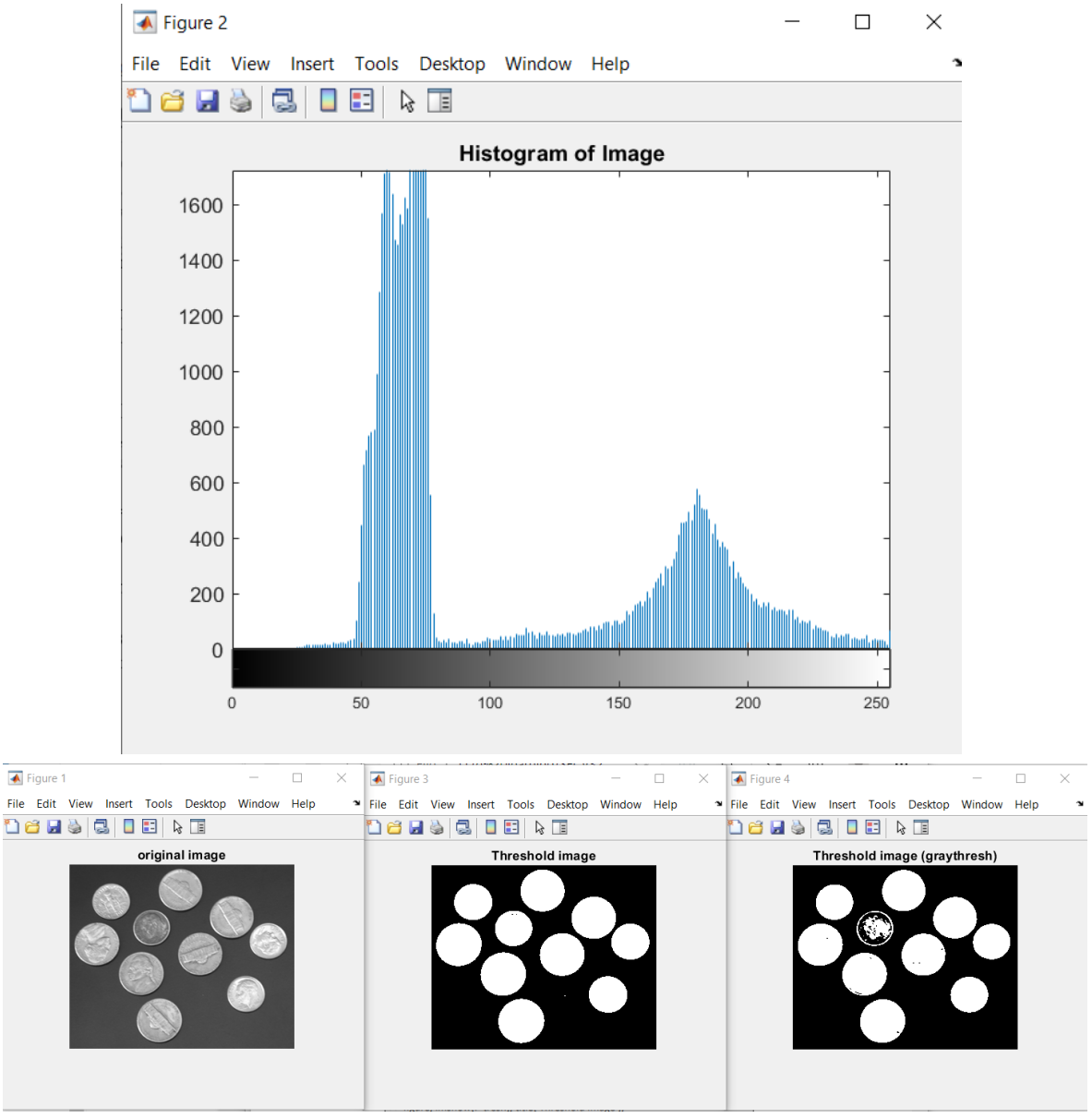
```
footBall=imread('noiseball.png');
imshow(footBall)

%Determine good padding for Fourier transform
PQ = paddedsize(size(footBall));

%Create Notch filters corresponding to extra peaks in the Fourier transform
```

```
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);

% Calculate the discrete Fourier transform of the image
F=fft2(double(footBall),PQ(1),PQ(2));

% Apply the notch filters to the Fourier spectrum of the image
FS_football = F.*H1.*H2.*H3.*H4.*H5.*H6;

% convert the result to the spatial domain.
F_football=real(ifft2(FS_football));

% Crop the image to undo padding
F_football=F_football(1:size(footBall,1), 1:size(footBall,2));

%Display the blurred image
figure, imshow(F_football,[])

% 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, imshow(S1,[])
figure, imshow(S2,[])
```

Explanation:
In this tutorial, Notch filter is being used to make the image output more clearer and brighter. The frequency domain will run a notch filter and determine the good padding for Fourier transform. A notch filter is created corresponding to extra peaks in the Fourier Transform. The discrete Fourier is then calculated to transform the image and the notch filter is applied to the Fourier spectrum of the image.

LAB 6 : Understanding in Segmentation

A. Tutorial 1 Tutorial Heuristic Thresholding

Example Output


Script

I = imread('coins.png');
figure, imshow(I), title('original image');
figure, imhist(I), title('Histogram of Image');
T=85; I_tresh=im2bw(I,(T/255));

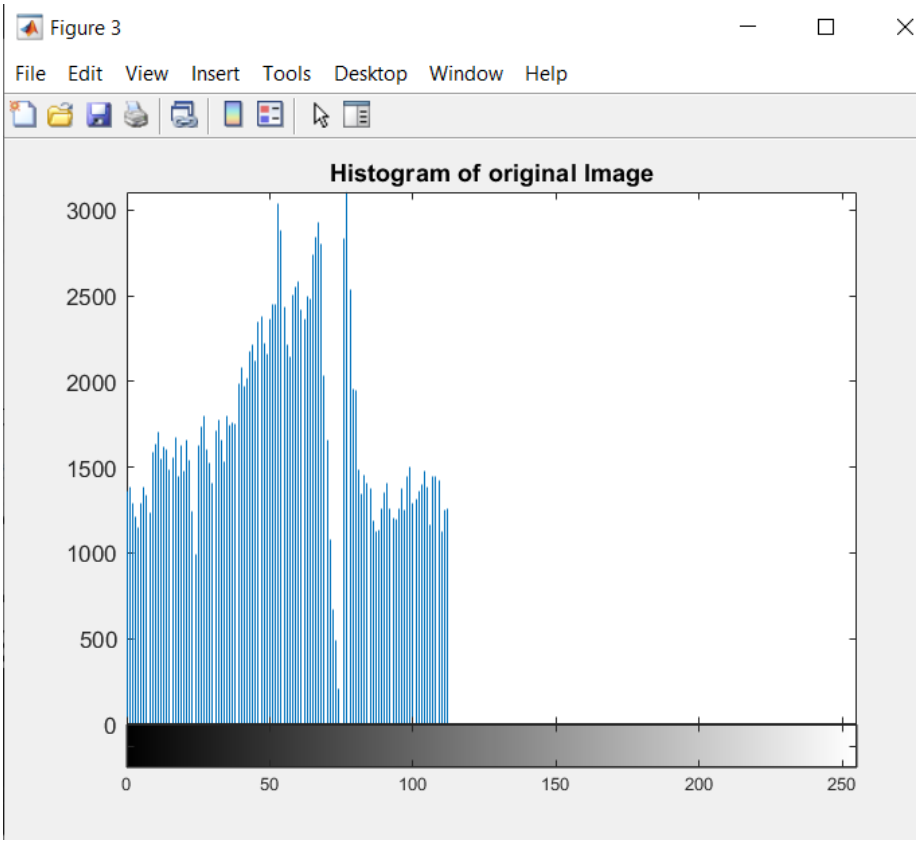
```
figure, imshow(I_tresh), title('Threshold image');

T2=graythresh(I);
I_tresh2=im2bw(I,T2);
figure, imshow(I_tresh2), title('Threshold image')
```

Explanation

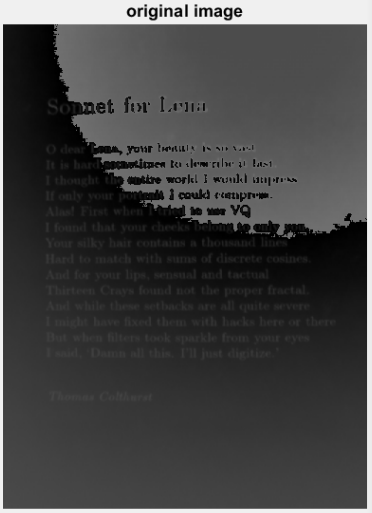
In this tutorial, the first method of image thresholding is using imhist() function to inspect the image's histogram. The value of T here is set as 85 which is applied to the image by using im2bw() function 4. The second method is using optimal image thresholding by using graythresh() function instead of imhist. In this case, the result for the first method is acceptable instead of the second method.

B. Tutorial 2 Tutorial Adaptive Thresholding

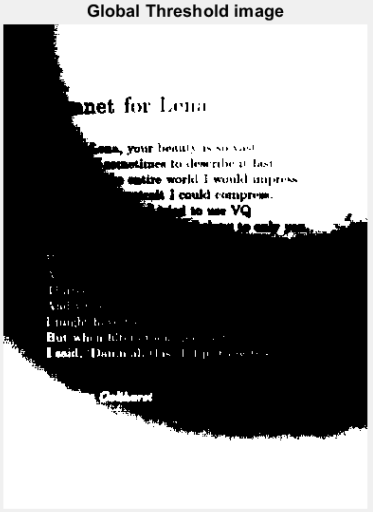
Example Output



original image

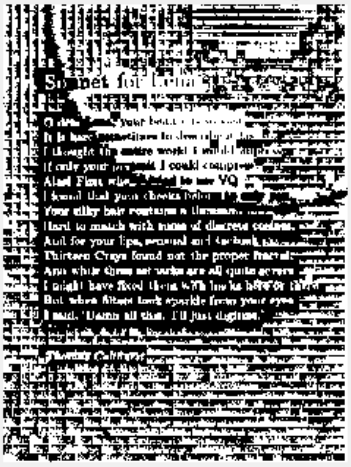


Global Threshold image



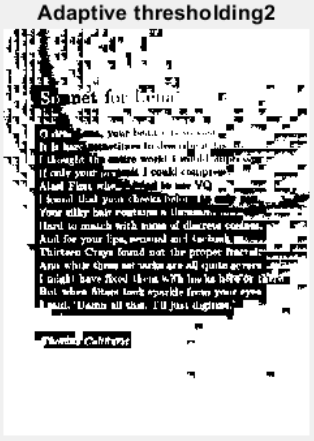
original image

Adaptive thresholding



original image

Adaptive thresholding2



Script

```
I=imread('gradient_with_text.tif');
figure, imshow(I), title('original image');
 I_gthresh=im2bw(I,graythresh(I));
figure, imshow(I_gthresh), title('Global Threshold image');
figure, imhist(I), title('Histogram of original Image');

I_thresh=blkproc(I,[10,10],@adapt_thresh);
figure,subplot(1,2,1),imshow(I),title('original image');
subplot(1,2,2),imshow(I_thresh),title('Adaptive thresholding');


std_without_text = std2(I(1:10, 1:10));
std_with_text = std2(I(100:110, 100:110));

I_thresh=blkproc(I,[10,10],@adapt_thresh2);
figure,subplot(1,2,1),imshow(I),title('original image');
subplot(1,2,2),imshow(I_thresh),title('Adaptive thresholding2');


function y = adapt_thresh(x)
 y=im2bw(x,graythresh(x));
end

function y = adapt_thresh2(x)
if std2(x)<1
y=ones(size(x,1),size(x,2));
else y=im2bw(x,graythresh(x));
end
end
```

Explanation

In this tutorial, initially the image thresholding issue is using Global thresholding, and follows with an adaptive thresholding. The function of adapt_thresh(x) is created for thresholding. And this adapt_thresh(x) function is called to get an adaptive threshold via a 10x10 pixel block by using the blkproc() function. The output image seems to be messy and unclear with the content.
While the second adaptive thresholding is an Improving adaptive thresholding from previous adaptive thresholding. The previous function is changed by using std value in the area of text and non-text. And the output shows that the noise is reduced compared with the first adaptive thresholding.