**Name: Hamad Nasir**

**Roll No: 120312**

**Section: BSCS-6C**

**Speech and Image Processing Lab 4**

**Tasks:**

**Part (a)**

Take a 300x300 pixel photo jpeg image and read it in MATLAB using imread() function and display it using imshow() function now convert it to double using double() and again display it using imshow() (note: for double image the color range is from 0 to 1 and not from 0 to 255). Convert this 300x300x3 RGB image array into 300x300 grayscale array and display it

**Matlab Code:**

%Task - 1

img1 = imread('img.jpg');

figure,imshow(img1);

title('Original Image')

%disp(img1)

img2 = im2double(img1);

figure,imshow(img2);

title('Double Image')

%disp(img2)

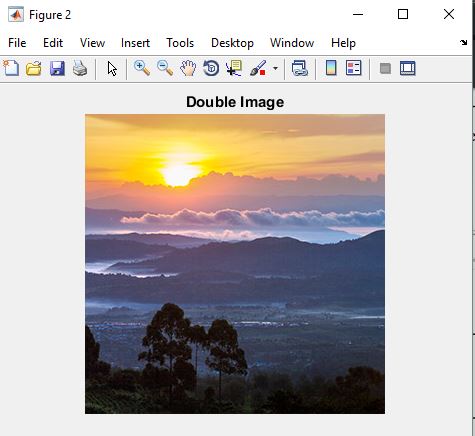
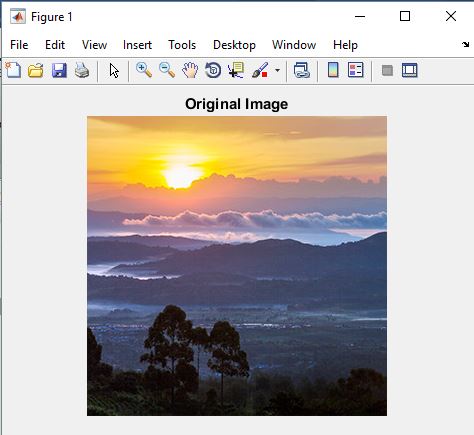
img3 = rgb2gray(img2);

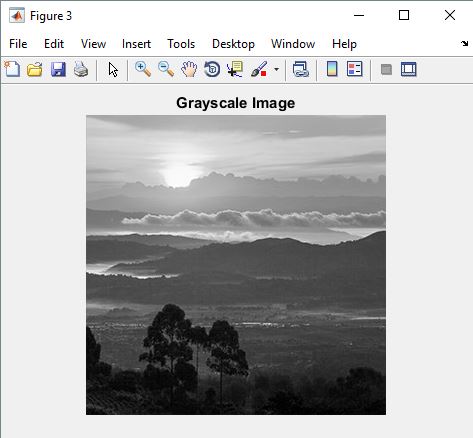
figure,imshow(img3);

title('Grayscale Image')

%disp(img3)

**Output Screenshot:**

****

****

**Part (b)**

Now take 2-dimensional Fourier transform of the grayscale image using fft2() function. The output is a complex numbered array. Take the magnitude of this frequency spectrum and display it on screen as an intensity image. (hint: in MATLAB it is possible to give intensity level range e.g. imshow(img, [0 10] while displaying a grayscale image). Now take the inverse fourier transform using ifft2() function, and show that the original image and fft2 followed by ifft2 yield the same result.

**Matlab Code:**

%Task - 2

img4 = fft2(img3);

figure,imshow(abs(fftshift(img4)),[0 10])

title('Fourier Transform Image')

%disp(img4)

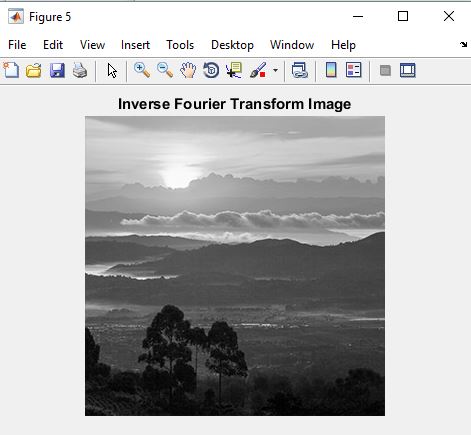
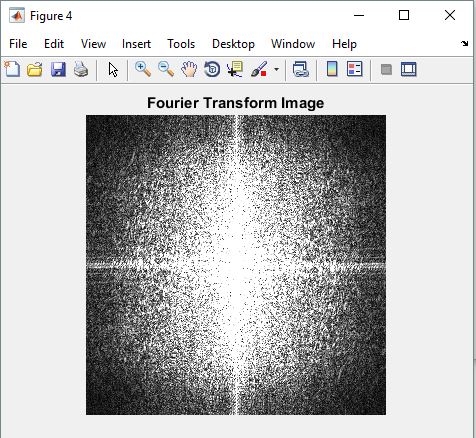
img5 = ifft2(img4);

figure,imshow(img5);

title('Inverse Fourier Transform Image')

%disp(img5)

**Output Screenshot:**

****

**Part (c)**

On the fourier transform array, remove high frequencies by setting them to zero. The high frequency values are the ones near the center of image away from corners. Now take ifft2 and compare the result with the original image. It should be a blurred version.

**Matlab Code:**

%Task - 3

a=img3;

%getting height and width of image

[m n]=size(a);

%taking fourier transform of grayscale image

f\_transform=fft2(a);

%moving the zero-frequency component to the center of the array.

f\_shift=fftshift(f\_transform);

%dividing height by 2

p=m/2;

%dividing width by 2

q=n/2;

%setting cutoff freuency

%lesser the value greater the blur

d0=29;

%creating the filter mask

for i=1:m

for j=1:n

distance=sqrt((i-p)^2+(j-q)^2);

%low\_filter is the created mask

low\_filter(i,j)=exp(-(distance)^2/(2\*(d0^2)));

end

end

%applying filter for removing high freuencies

filter\_apply = f\_shift .\* low\_filter;

image\_orignal=ifftshift(filter\_apply);

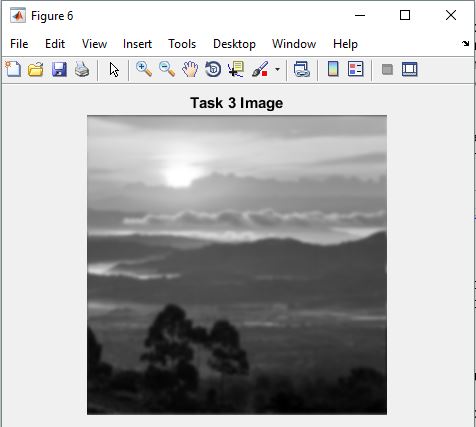
image\_filter\_apply=abs(ifft2(image\_orignal));

%displaying the image

figure,imshow(image\_filter\_apply,[])

title('Task 3 Image')

**Output Screenshot:**

****

**Part (d)**

On the fourier transform array, remove low frequencies by setting them to zero. The low frequency values are the ones near the corners of image. Now take ifft2 and compare the result with the original image. It should contain only edges and corners.

**Matlab Code:**

%Task - 4

%taking inverse of filter for removing low freqencies

low\_filter = imcomplement(low\_filter);

%applying filter

filter\_apply = f\_shift .\* low\_filter;

image\_orignal=ifftshift(filter\_apply);

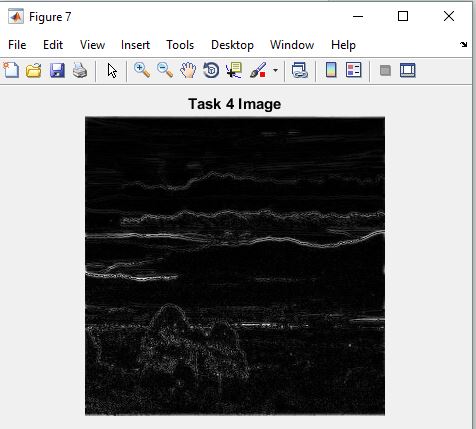
image\_filter\_apply=abs(ifft2(image\_orignal));

%displaying the image

figure,imshow(image\_filter\_apply,[])

title('Task 4 Image')

**Output Screenshot:**

****