Boran Yildirim
21401947

EEE391 MATLAB ASSIGNMENT 2

In this assignment, it is required to apply HPL, LPF and a custom designed filter to the image shown below. The image is downloaded from the given website along with the necessary .m files.

Then it is explained that for a NxM image matrix with N indicating the horizontal size and M the vertical size, the Fourier transform (Matlab indices *m* x *n*, vertical and horizontal dimensions respectively)

$w_x = -\pi\ ;\ for\ n = 1\ ,\ w_x = \pi - \frac{2\pi}{N}\ for\ n = N$solving the linear equations will give

$$w_x = -\pi + \frac{2\pi(n-1)}{N}\ ,\ for\ 1 \le n\ \le N$$

Therefore the corresponding 2D frequency in Fourier form for arbitrary horizontal dimension k value is $\frac{(k-1)}{N}$. Similarly, as shown in the below derivation, the corresponding 2D frequency in Fourier form for arbitrary vertical dimension l value is $\frac{(l-1)}{N}$.

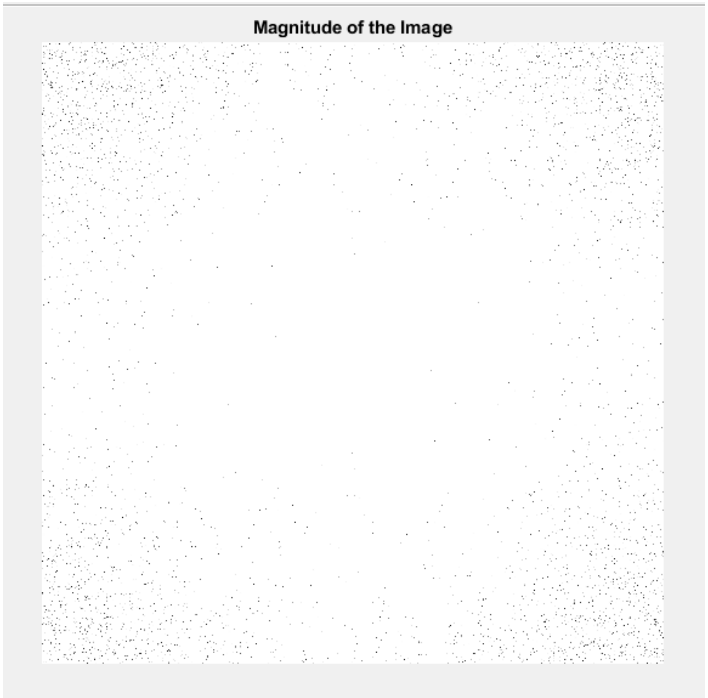
$$w_y = -\pi + \frac{2\pi(m-1)}{M} for\ 1 \le m\ \le M,$$

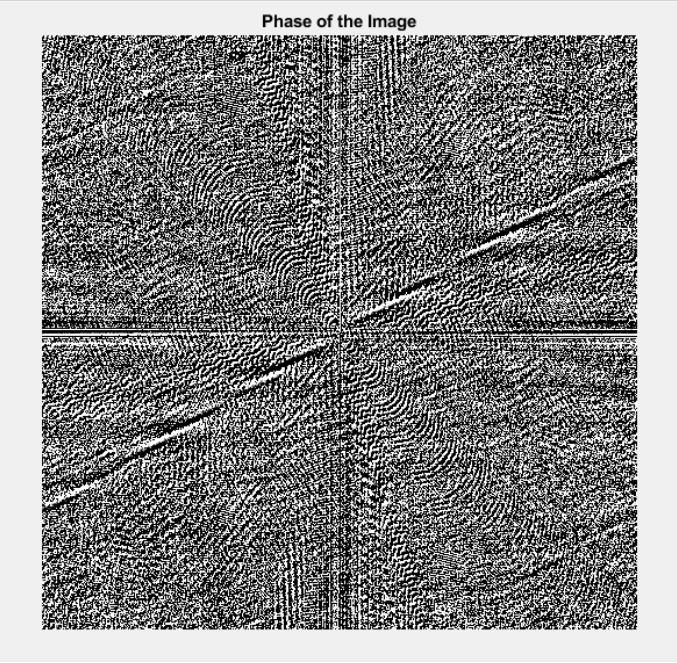
Thus, for [m,n] pair, the corresponding 2D frequency is

$$\left(w_x, w_y\right) = \left[-\pi + \frac{2\pi(n-1)}{N}, -\pi + \frac{2\pi(m-1)}{M}\right]$$

Two dimensional Fourier Transform can be calculated based on one dimensional FFT formula.

Now that image's frequency is distracted, we can plot the magnitude and phase of the original image in the figures below.

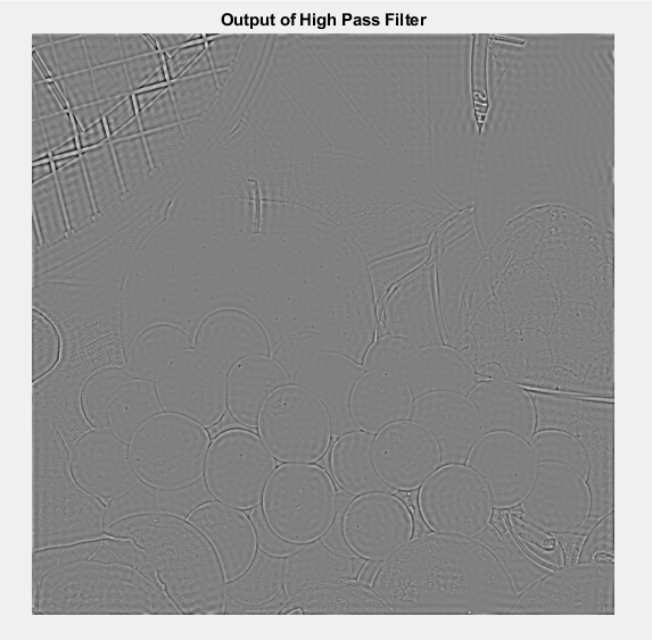




Boran Yildirim
21401947

Part 1: High Pass & Low Pass Filters

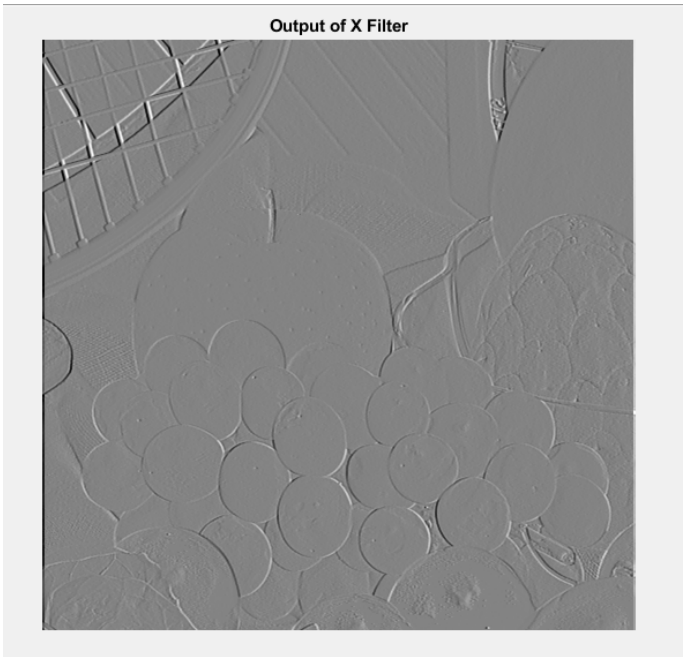
Compared to the original image this image seems blurry. Low pass filter is known as blur mask so it makes sense. Changing the bandwidth will change the effect of the filter. Increasing it will lessen the blurring and the opposite will do the opposite effect.



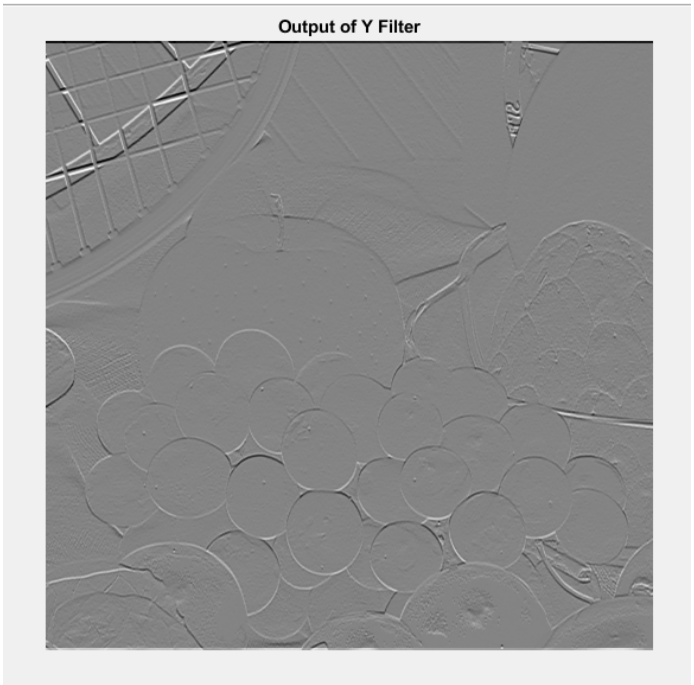
Compared to the original image there is only the edges in this image. High pass filter is known to get the sharp edges. Changing the bandwidth will change the effect of the filter. Increasing it will lessen the sharpening and the opposite will do the opposite effect.

Boran Yildirim
21401947

Part 2: Custom Filters



This filter is a horizontal filter which filters the horizontal parts detecting the horizontal edges in the image.



This filter is a vertical filter which filters the horizontal parts detecting the horizontal edges in the image. Using the given 2D convolution formula we can get the following:

Boran Yildirim
21401947

$$h_{x,y}[-1,-1] = h_x[0,-1]h_y[-1,0] = 1,\ h_{x,y}[-1,1] = h_x[0,1]h_y[-1,0] = -1$$
$$h_{x,y}[1,-1] = h_x[0,-1]h_y[1,0] = -1,\ h_{x,y}[1,1] = h_x[0,1]h_y[1,0] = 1$$

$$h_{x,y}[m,n] = \delta(m+1,n+1) - \delta(m+1,n-1) - \delta(m-1,n+1) + \delta(m-1,n-1)$$

This combined filter acts as both horizontal filter and vertical filter which can be seen from the image that all the edges can be seen.

Boran Yildirim
21401947

The result of this consecutive filter is same with the combined filter showing the convolution associativity.

Boran Yildirim
21401947

Part 3: Sobel Operator

The Sobel Operator is one of the edge detection algorithms, it creates an image emphasizing edges. The edges are trivial compared to the part 2 counterpart.

Boran Yildirim
21401947

APPENDIX

main.m

```
clc;
clear;
close all;

image_rgb=imread('fruits.png');
image=im2double(rgb2gray(image_rgb));

figure();imshow(image,[]);title('Original Image');

FT_image= FourierTransform(image);

[M,N] = size(image);
wy=-pi:2*pi/M:pi-2*pi/M;
wx=-pi:2*pi/N:pi-2*pi/N;
figure;imshow(abs(FT_image));title('Magnitude of the Image');
figure;imshow(angle(FT_image));title('Phase of the Image');
% High Pass & Low-Pass Filters
LPHP(FT_image,wx,wy);
% Custom Filters
hx=[-1,0,1];
hy=[-1;0;1];
hxy=[1,0,-1;0,0,0;-1,0,1];
customFilters(image,hx,hy,hxy);
% Sobel Operator
h_x = [-1,0,1;-2,0,2;-1,0,1];
h_y = [-1,-2,-1;0,0,0;1,2,1];
sobelOperator(image,h_x,h_y)
```

LPHP.m

```
function [] = LPHP(FT_image,wx,wy)
for m=1:length(wy)
    for n=1:length(wx)
        LP(m,n)=(abs(wx(m))<pi/4)*(abs(wx(n))<pi/4);
    end
end
HP=ones(size(LP))-LP;
% figure;imshow(LP)
Filtered_Image = InverseFourierTransform(LP.*FT_image);
figure;imshow(real(Filtered_Image),[]);title('Output of Low Pass Filter')
figure;imshow(real(InverseFourierTransform(HP.*FT_image)),[]);title('Output of
High Pass Filter')
% figure;imshow((wx<pi/4)'*(wy<pi/4));
end
```

customFilters.m

```
function [] = customFilters(img,filter_x,filter_y,filter_xy)
% Calculations
img_x_filtered = convolution2D(img,filter_x);
img_y_filtered = convolution2D(img,filter_y);
img_xy_filtered = convolution2D(img,filter_xy);
img_x_y_filtered = convolution2D(img_x_filtered,filter_y);
% Plots
```

```
figure;imshow(img_x_filtered,[]);title('Output of X Filter');
figure;imshow(img_y_filtered,[]);title('Output of Y Filter');
figure;imshow(img_xy_filtered,[]);title('Output of XY Filter');
figure;imshow(img_x_y_filtered,[]);title('Output of X then Y Filter');
end
```

convolution2D.m

```
function output = convolution2D(input,filter)

sf=size(filter);
output=zeros(sf-1+size(input));

for i=1:sf(1)
    for j=1:sf(2)
        uplimx=i+size(input,1)-1;
        uplimy=j+size(input,2)-1;
        output(i:uplimx,j:uplimy)=output(i:uplimx,j:uplimy)+filter(i,j)*input;
    end
end
end
```

sobelOperator.m

```
function [] = sobelOperator(img,filter_x,filter_y)
G_x = convolution2D(img,filter_x);
G_y = convolution2D(img,filter_y);
G = sqrt(G_x.^2 + G_y.^2);
figure;imshow(G,[]);title('Output of G Filter');
end
```