Department of Computer Science and Software Engineering
Comp 6771 Image Processing
Assignment 3

Mandana Samiei, ID: 40059116 *1/11/2017*

Theoretical Questions

1- 3*3 Spatial Mask averages the four closest neighbors excluding the point itself:
The four closest neighbors are the ones that are connected to the centered point in an orthogonal line:

f(x-1,y+1)	f(x,y+1)	f(x+1,y+1)
f(x-1,y)	f(x,y)	f(x+1,y)
f(x-1,y-1)	f(x,y-1)	f(x+1,y+1)

$g(x,y) = f(x,y) \times h(x,y) = \frac{1}{4}\{f(x-1,y) + f(x,y+1) + f(x+1,y) + f(x,y-1)\}$

2-D Fourier Transform Formula:

$$F(u,v) = \sum_{x=0}^{M}\sum_{y=0}^{N} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

Fourier transform of the equivalent filter (frequency domain):

$G(u,v) = \frac{1}{4}\left\{e^{\frac{-j2\pi u}{M}} + e^{\frac{j2\pi v}{N}} + e^{\frac{j2\pi u}{M}} + e^{\frac{-j2\pi v}{N}}\right\} \times F(u,v) = H(u,v) \times F(u,v)$

So: $H(u,v) = \frac{1}{4}\left\{e^{\frac{-j2\pi u}{M}} + e^{\frac{j2\pi v}{N}} + e^{\frac{j2\pi u}{M}} + e^{\frac{-j2\pi v}{N}}\right\} = \frac{1}{4}\left\{2\cos\left(\frac{2\pi u}{M}\right) + 2\cos\left(\frac{2\pi v}{N}\right)\right\} = \frac{1}{2}\left\{\cos\left(\frac{2\pi u}{M}\right) + \cos\left(\frac{2\pi v}{N}\right)\right\}$

2- First Derivatives in 2D, the corresponding mask in the spatial domain is like the below image:

0	**0**	**0**
-1	**2**	**0**
0	**-1**	**0**

$$g(x,y) = f(x,y) - f(x-1,y) + f(x,y) - f(x,y-1)$$

$$G(u,v) = \left\{e^{j2\pi(0)} - e^{-j2\pi\left(\frac{ux}{M}\right)} + e^{j2\pi(0)} - e^{-j2\pi\left(\frac{vy}{N}\right)}\right\} \times F(u,v) =$$

$$\underbrace{\left\{1 - e^{-j2\pi\left(\frac{u}{M}\right)} + 1 - e^{-j2\pi\left(\frac{v}{N}\right)}\right\}}_{H(u,v)} \times F(u,v)$$

$$H(u,v) = \left(1 - e^{-j2\pi\left(\frac{u}{M}\right)}\right) + \left(1 - e^{-j2\pi\left(\frac{v}{N}\right)}\right)$$

For the first term we multiply it by $e^{j\pi(u/M)}$ and for the second term by $e^{j\pi(v/N)}$:

$$H(u,v) = 2j\sin(\pi u/M)\, e^{-j\pi u/M} + 2j\sin(\pi v/N) e^{-j\pi v/N}$$

Derivative filter is a high pass filter since it specifies intensity changes between two adjacent pixels.
To justify that, we calculate the value of filter in different places:
First we need to change the function to the familiar centered form:

$$H(u,v) = 2j\sin\left(\pi(u - \frac{M}{2})/M\right) e^{-j\pi u/M} + 2j\sin\left(\pi(v - \frac{N}{2})/N\right) e^{-j\pi v/N}$$

We will check the value of the filter:

- $u = 0\ and\ v = 0: H(u,v) = -4j$
- $u = \frac{M}{2}\ and\ v = \frac{N}{2} : H(u,v) = 0$
- $u = M\ and\ v = N: H(u,v) \approx 4j$

$H(u,v)$ is zero in the center of the filter that means the DC term is eliminated, also when the value of u and v increases, the value of filter decreases, which are the characteristics of a high pass filter.

3-

a) Yes, it does matter which one we apply first, because high pass filters generally destroy the contrast. We can also prove this relation by using their formula:
Consider that we first apply histogram equalization:
$g(x,y) = T\{f(x,y)\}$ and then we apply high pass filter:
$A(x,y) = g(x,y) * h_{hp}(x,y) = T\{f(x,y) * h_{hp}(x,y)\}$

The we try to apply high pass filter first:
$A(x,y) = f(x,y) * h_{hp}(x,y)$
$g(x,y) = T\{A(x,y)\} = T\{f(x,y) * h_{hp}(x,y)\}$

Generally, Histogram Equalization is not a linear transform because it uses the number of pixels with a particular intensity that is calculated by probability density, so:
$T\{f(x,y) * h_{hp}(x,y)\} \neq T\{f(x,y) * h_{hp}(x,y)\}$
Thus, the order does matter.

b) Since high pass filter reduce the contrast of an image, if we do histogram equalizing and then apply high pass filter, all the gain in contrast improvement will be lost in filtering process. So, it is more practical that we apply high pass filter first to emphasize edges and then use histogram equalization to repair the contrast. Also, I checked both of these methods in the MATLAB and I found the result of second one more suitable, because you can easily detect the edges and structure of objects in the picture.

First approach:

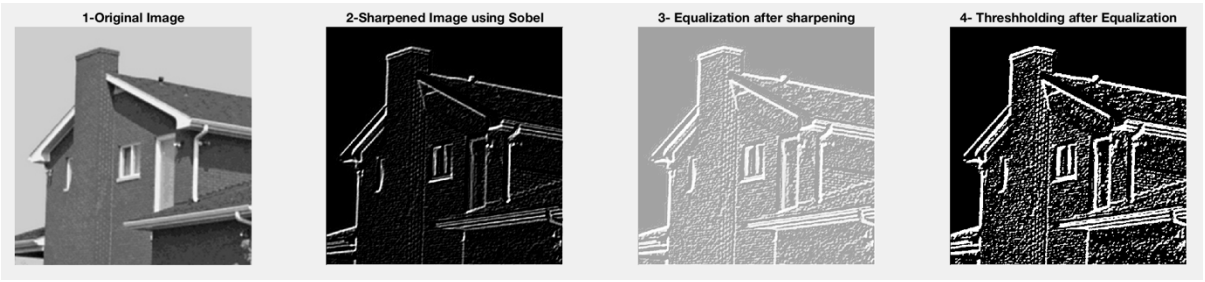
```
I = imread('house.tif');
J = I(:,:,1);%% Extracting Channel 1
figure;
subplot(1,3,1);
imshow(J);title('Original Image');
%% First Histogram Equalizing
equalized_img = histeq(J);
subplot(1,3,2);
imshow(equalized_img);title('Equalized Image');
%% Then Sharpening
BW=edge(equalized_img, 'sobel');%sharpening
subplot(1,3,3);
imshow(BW); title('Sharpened Image using Sobel');
```

Second approach:

```
I = imread('house.tif');
J = I(:,:,1);%% Extracting Channel 1
TH=0.8;
figure;
subplot(1,4,1);
imshow(J);title('1-Original Image');

%%First Sharpening
gx=[-1,-2,-1 ; 0,0,0 ; 1,2,1];
gy =[-1,0,1 ; -2,0,2 ; -1,0,1];
Gx = conv2(im2double(J), gx,'same');
Gy = conv2(im2double(J), gy,'same');
G = Gx+Gy;
subplot(1,4,2);
imshow(G);title('2-Sharpened Image using Sobel');

%%Then Histogram equalizing and thresholding
equalized_img = histeq(G);
subplot(1,4,3);
imshow(equalized_img); title('3- Equalization after sharpening');
subplot(1,4,4);
imshow(equalized_img>TH); title('4- Thresholding after
Equalization');
```



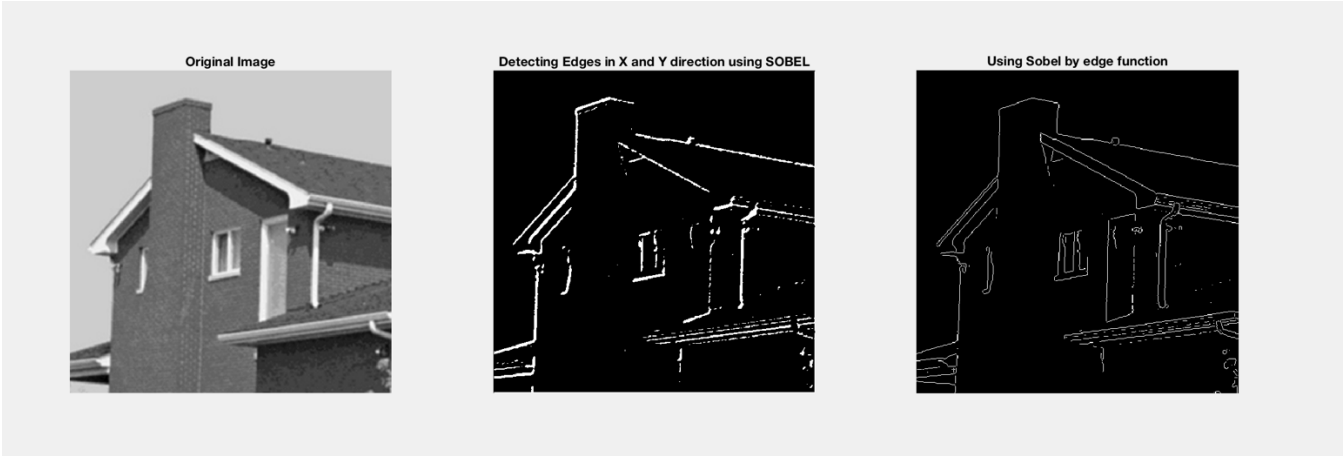
Programing Questions

4-

a) Sobel Filter

```
%%----------Question 4-a --------------
%%--------- Mandana Samiei-------------
%%--------- Student Id: 40059116 ------

%%Edge Detecting Using Canny Filter
I = imread('house.tif');
J = I(:,:,1);%% Extracting Channel 1
TH=120; %% Threshholding
gx=[-1,-2,-1 ; 0,0,0 ; 1,2,1];
gy =[-1,0,1 ; -2,0,2 ; -1,0,1];
img_double = double(J); %% Converting uint8 image to double
Gx = conv2(img_double, gx,'same');
Gy = conv2(img_double, gy,'same');
G = Gx+Gy;
%Using Edge Function
BW = edge(img_double, 'sobel');
%% Plotting result
figure;
subplot(1,3,1);
imshow(J);
title('Original Image');
subplot(1,3,2);
imshow(G>TH);
title('Detecting Edges in X and Y direction using SOBEL');
subplot(1,3,3);
imshow(BW);
title('Using Sobel by edge function');
```



The Sobel operator uses two 3×3 kernels which are convolved with the original image to calculate the derivatives – one for horizontal changes, and one for vertical.

b) Canny Filter

```
%%Edge Detecting Using Canny Filter
I = imread('house.tif');
J = I(:,:,1);%% Extracting Channel 1
img_double = double(J); %% Converting uint8 image to double
BW = edge(img_double,'canny');
%% Plotting result
figure;
subplot(1,2,1);
imshow(J);
title('Original Image');
subplot(1,2,2);
imshow(BW);
title('After Applying Canny Filter');
```

The Canny method result differs from Sobel edge detector in that it uses two different thresholds (to detect strong and weak edges), and includes the weak edges in the output only if they are connected to strong edges. This method is therefore less likely than the others to be affected by noise, and more likely to detect true weak edges.