In the game of God

Sharif University of Technology

Dr. Fatemizade

Amirreza Hatamipour

97101507

Theoretical exercise:

First we know this from FFT:

We know the FFT of Gaussian is stay Gaussian filter and now show it. we can write in Spherical space and then we have:

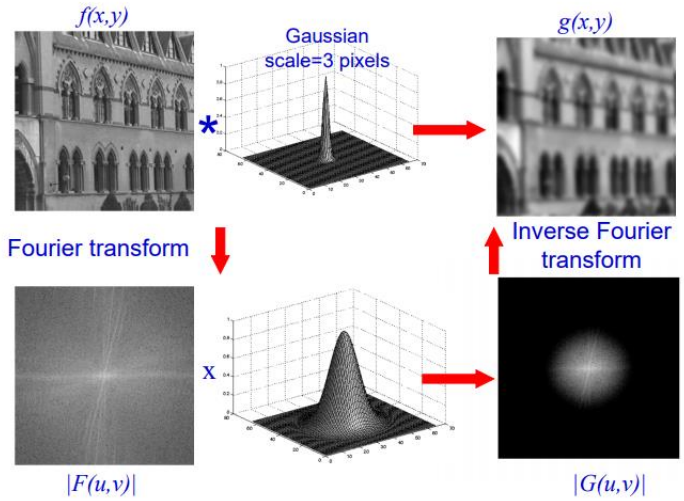


Figure 1 Gaussian filter in spatial domain and frequency domain

$$f(r) = \frac{1}{2\pi\sigma} e^{-\frac{r^2}{2\sigma^2}} \qquad , r^2 = x^2 + y^2$$

And we know the Fourier transform of exponential function is:

$$F\left[e^{-ax^2}\right](k) = \sqrt{\frac{\pi}{a}}\, e^{-\frac{\pi^2 k^2}{a}}$$

And finally we have:

$$F[u,v] = F[\rho] = e^{-2\pi^2\rho^2\sigma^2} \qquad , \rho^2 = u^2 + v^2$$

As a result, we can see the FFT of the Gaussian low pass filter It remains low pass.

Simulation exercises:

Question 1:

In each section we calculate SNR for four part of image. We reach to this table:

	Top-left(salt)	*Top-right(without)*	*button-left(both)*	*button-right(gauss)*
Before	*11.99*	*48.92*	*4.13*	*3.25*
*After 3*3 median filter*	*23.84*	*33.39*	*9.26*	*10.02*
*After 3*3 gauss filter(σ=2)*	*20.19*	*29.78*	*12.59*	*13.71*
*After 5*5 box filter*	*19.81*	*29.99*	*12.05*	*13.08*

Table 1:SNR of 4 part of image before and after apply different filtering

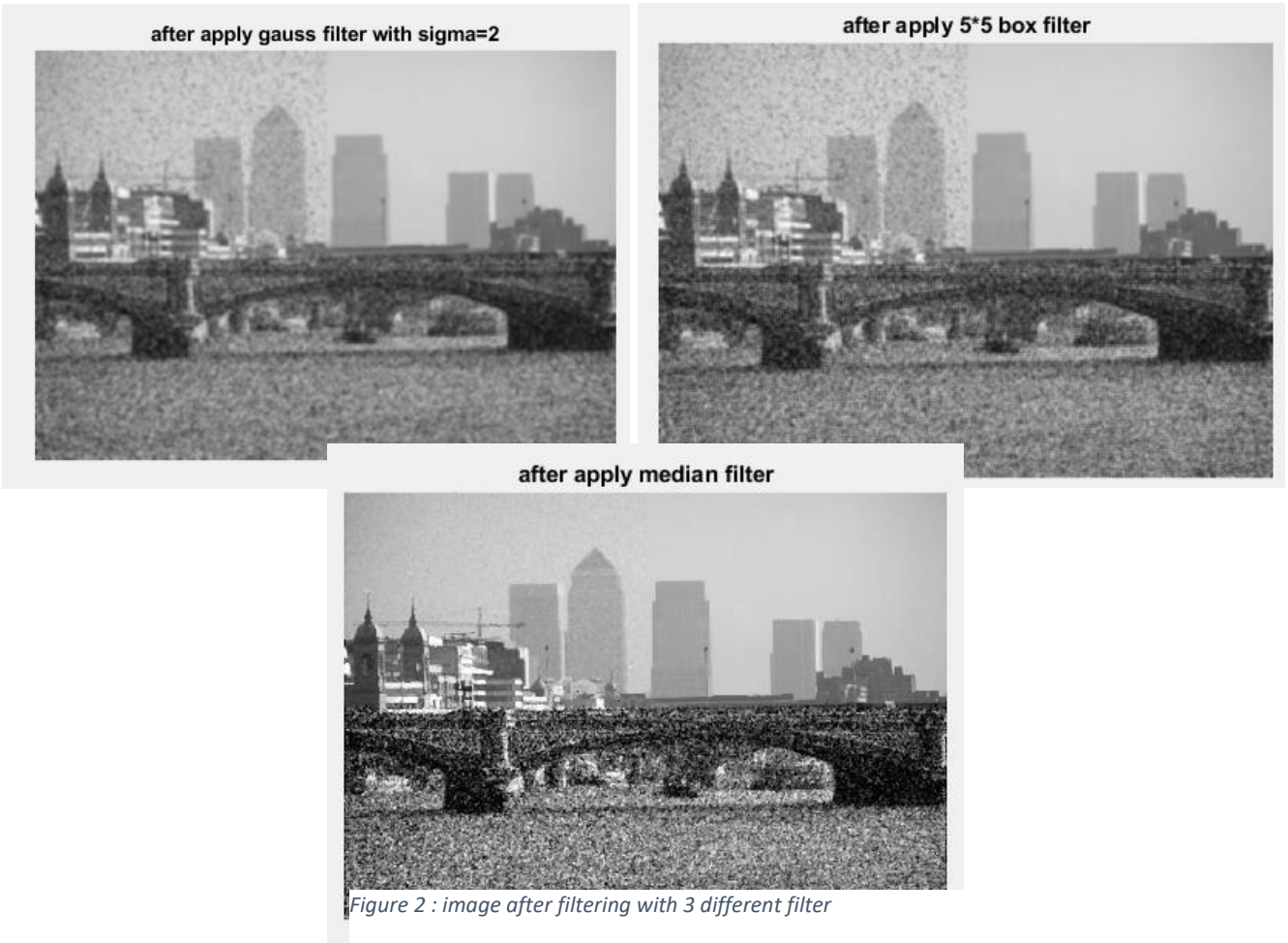


Figure 2 : image after filtering with 3 different filter

Base on the above table, we see for salt & pepper noise, the median filter is better than others because has maximum SNR among else and after that gauss filter is better. for Gaussian noise as we see on the table, the box filter and the gauss filter are better than the median filter. And for the section we have both noises, the gauss filter has maximum SNR among others.

From the results of filtering, we can see the median filter is good for salt & pepper noise and approximately can remove this noise from the picture.in addition to, the Gaussian filter is good for Gaussian noise.

Question 2:

A) *First load image to matlab. Then calculate Fourier transform of it and finally shift to center. we have:*

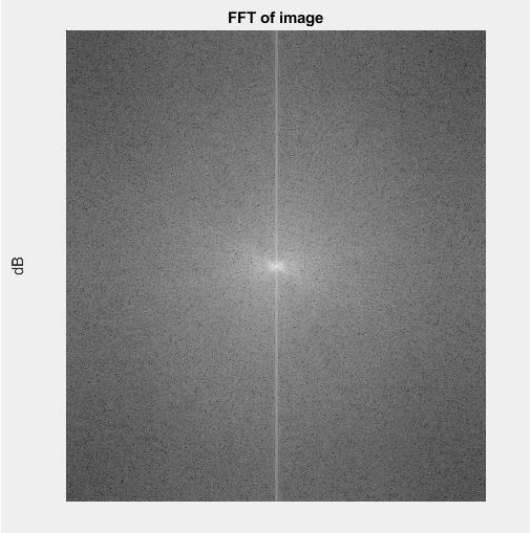


Figure 3: FFT of image

B) from slides we know the average of intensity of image is equal to f [0,0].

$$F(u,v) = \Sigma\Sigma f(m,n) e^{-j2\pi\left(u*\left(\frac{m}{M}\right)+v*\left(\frac{n}{N}\right)\right)}$$

$$if\ u = v = 0 \quad F(0,0) = \Sigma\Sigma f(m,n)$$

Then we have: (notice: if we want to use this method, we have to multiply the power of image and FFT for that Parseval identity to reach)

```
meanIntensity_fft =

    36.3036
```

And to image in spatial domain we have this value:

```
meanIntensity_image =

    36.3036
```

As we can see both calculation have a same result: 36.3036

Question 3:

A)

First load image to python.

consider first kernel is ([1,-1]) :

if apply this filter to image we have:

As we see this filter can use for approximation of first order derivative because in each pixel, we calculate the difference in intensity with the next pixel (in X axis)

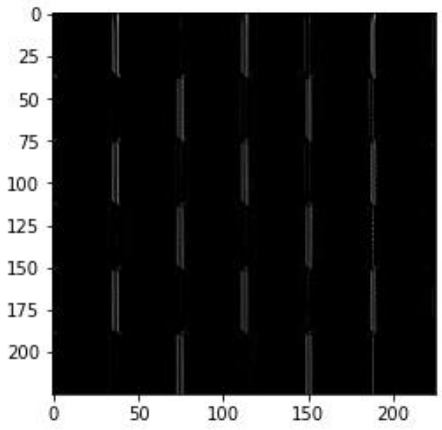


Figure 4: filtered image with([1,-1])

The next kernel is ([1,0]):

As we expect when apply this kernel to image we don't see any difference. because the other elements were zero when apply to each pixel.

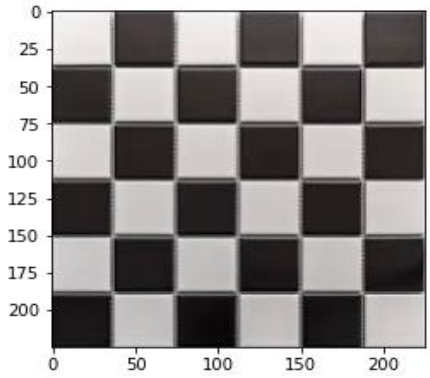


Figure 5: filtered image with ([1,0])

The next kernel is ([1,0, -1]):

This filter is for approximation of first order derivative with this difference we use the difference the next and previous pixel of each pixel. From approximation of Tylor series, we have:

$$f_x{}'(x,y) \cong \frac{f(x-1,y) - f(x+1,y)}{2}$$

Then we have this image:

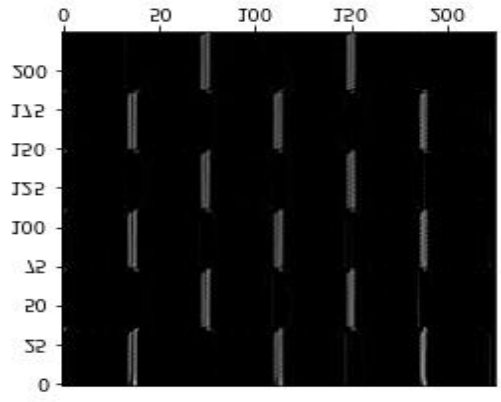


Figure 6: filtered image with filter ([1,0,-1)]

And the next filter is [[1], [0], [-1]]:

This filter has the same function as the previous filter with this difference that apply in other direction and give us an approximation of first-order horizontal derivative. It is obtained from the following formula:

$$f_y'(x,y) \cong \frac{f(x,y-1) - f(x,y+1)}{2}$$

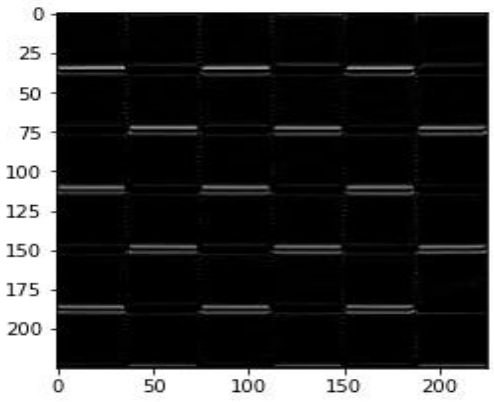


Figure 7: filtered image with filter ([[1],[0],[-1]])

and the last filter is [[-1, -1, -1], [-1,8, -1], [-1, -1, -1]]:

as we can see, we can use this filter to approximation of first-order derivative in both horizontal and vertical side of image. This filter named Laplacian filter.

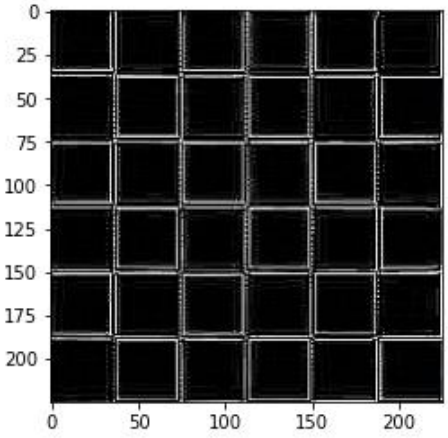


Figure 8: : filtered image with filter[[-1 , -1,-1],[-1,8,-1],[-1,-1,-1]]

B)

First we apply canny, sobbed and LOG to image. We have this result:

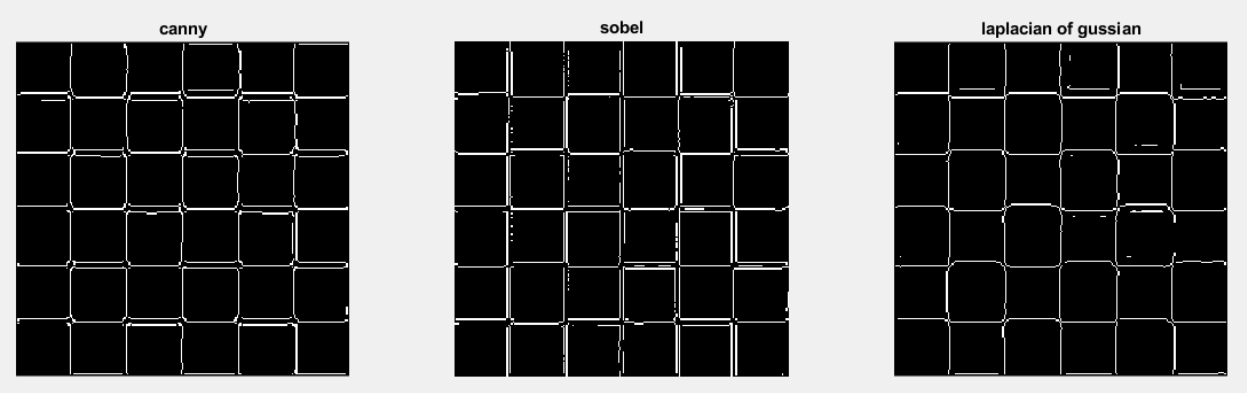


Figure 9: result of filtering

And now describe each filter algorithms:

Canny:

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images.

The Canny edge detection algorithm is composed of 5 steps:

1. *Noise reduction:*

    - *by applying Gaussian blur to smooth image.*

2. *Gradient calculation:*

    - *by calculating the gradient of the image using edge detection operators.*
    - *Edges correspond to a change of pixels' intensity. To detect it, the easiest way is to apply filters that highlight this intensity change in both directions: horizontal (x) and vertical (y)*

- *We can use these filters:*

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}.$$

And finally calculate G and $\boldsymbol{\Theta}$*:*

$$|G| = \sqrt{I_x^2 + I_y^2},$$

$$\theta(x, y) = arctan\left(\frac{I_y}{I_x}\right)$$

3. *Non-maximum suppression;*

- *the algorithm goes through all the points on the gradient intensity matrix and finds the pixels with the maximum value in the edge directions.*

4. *Double threshold;*

- *The double threshold step aims at identifying 3 kinds of pixels: strong, weak, and non-relevant*

5. *Edge Tracking by Hysteresis.*

- *Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one.*

Sobel:

The Sobel operator performs a 2-D spatial gradient measurement on an image and so emphasizes regions of high spatial frequency that correspond to edges. Typically,

it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image.

We have two kernel:

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

And then calculate magnitude in this way:

$$|G| = \sqrt{{G_x}^2 + {G_y}^2}$$

Laplacian of Gaussian:

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian smoothing filter in order to reduce its sensitivity to noise, and hence the two variants will be described together here. The operator normally takes a single gray-level image as input and produces another gray-level image as output.

Two commonly used discrete approximations to the Laplacian filter:

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

And finally we have this:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Question 4)

In this section we want rotate picture with using Fourier transform.

Figure 10 : rotated image

From the properties of the Fourier transform we know if we want to rotate the image 180 degrees in the spatial domain, we must calculate the conjugate of the Fourier transform of the original image and apply it to the image. (we know conjugate of the Fourier transform is same to shift the phase of FFT 180 degree)

Question 5:

As we see in slides of this course, we know important information have in phase of Fourier transform. So, as we expect each image that existed phase in it similar to the original image. For example, when we add phase of hand_xray to magnitude of brain_xray, the result similar to hand_xray (figure 7.d)

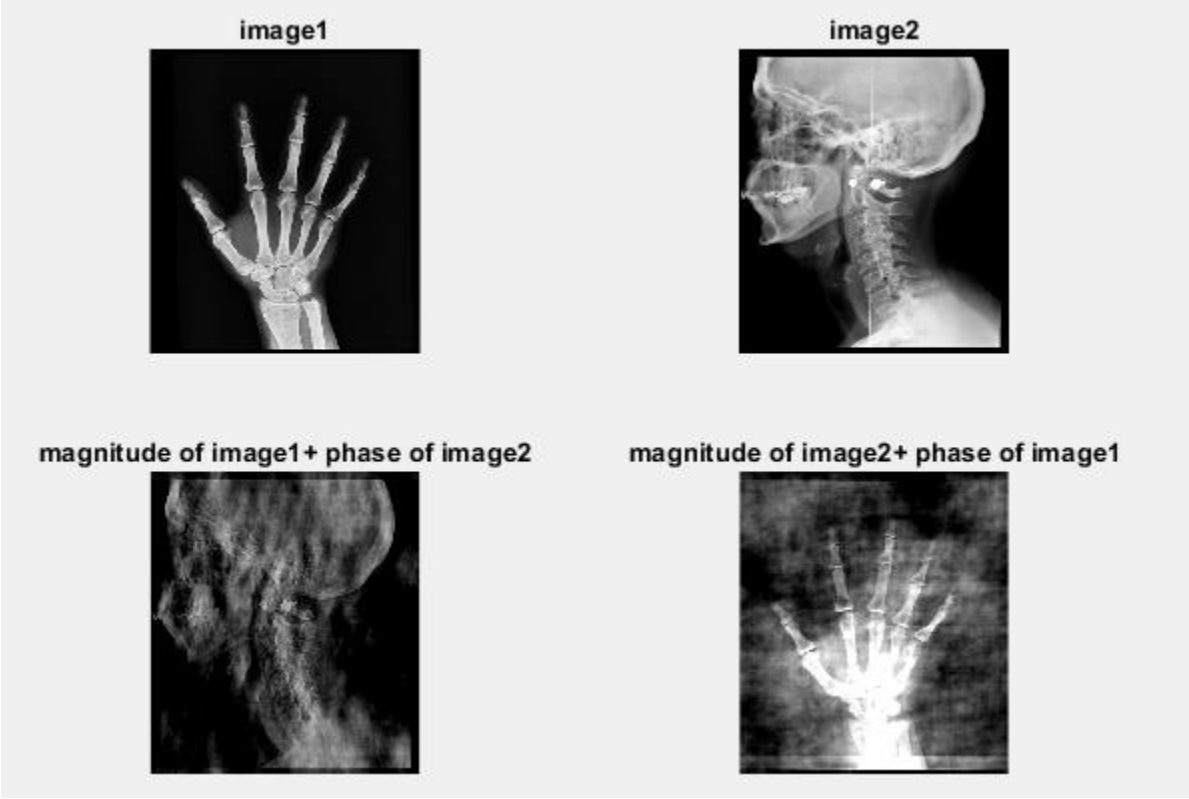
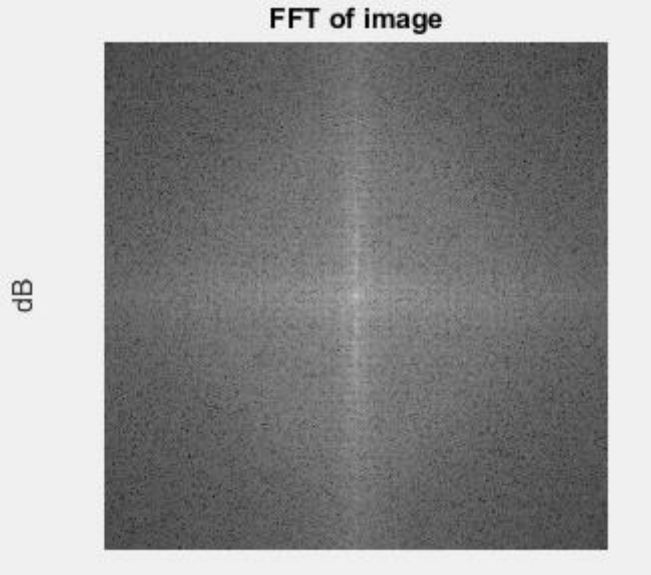


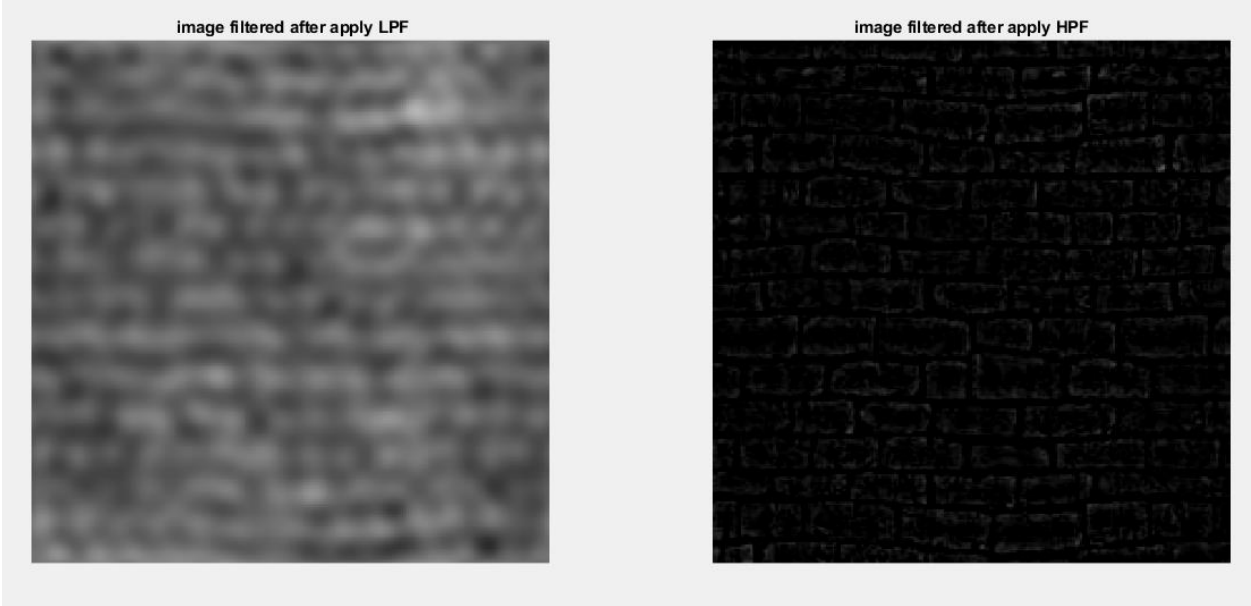
Figure 11:a) up left : original image of hand-xray up right b)original image of brain-xray c) down left: result of add phase of brain and magnitude of hand d) down left : result of add phase of hand and magnitude of brain

Question 6:

A) *If we apply Fourier transform to image and then take fftshiht from it, we reach this result:*



B) If we apply low pass filter to image we have:



D)

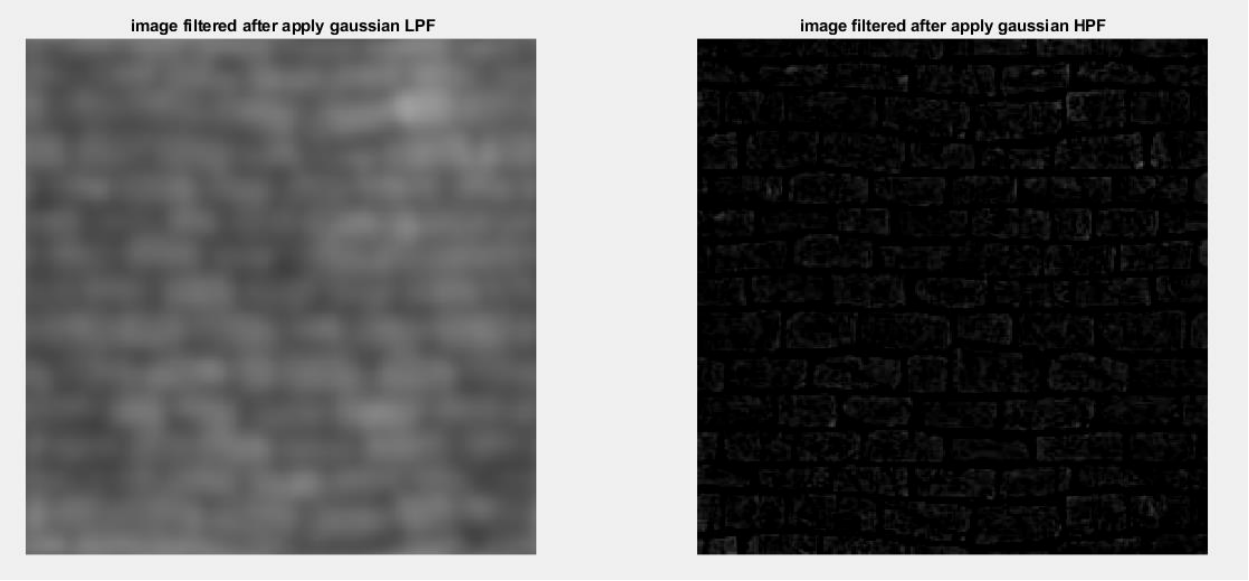


Figure 12: gaussian filter

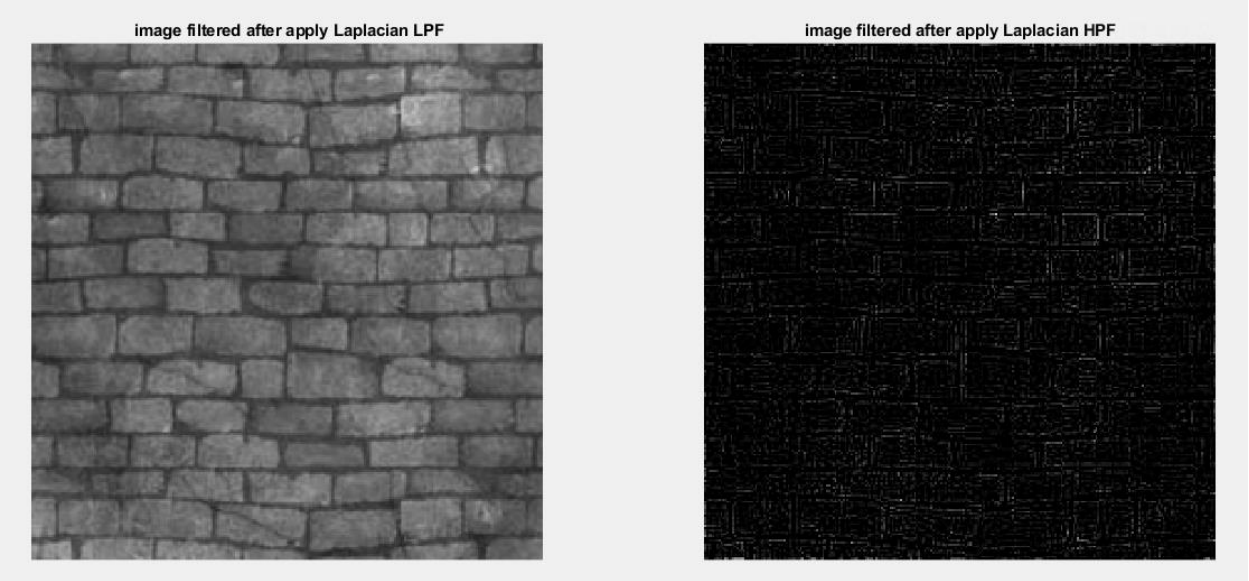


Figure 13 Laplacian filter

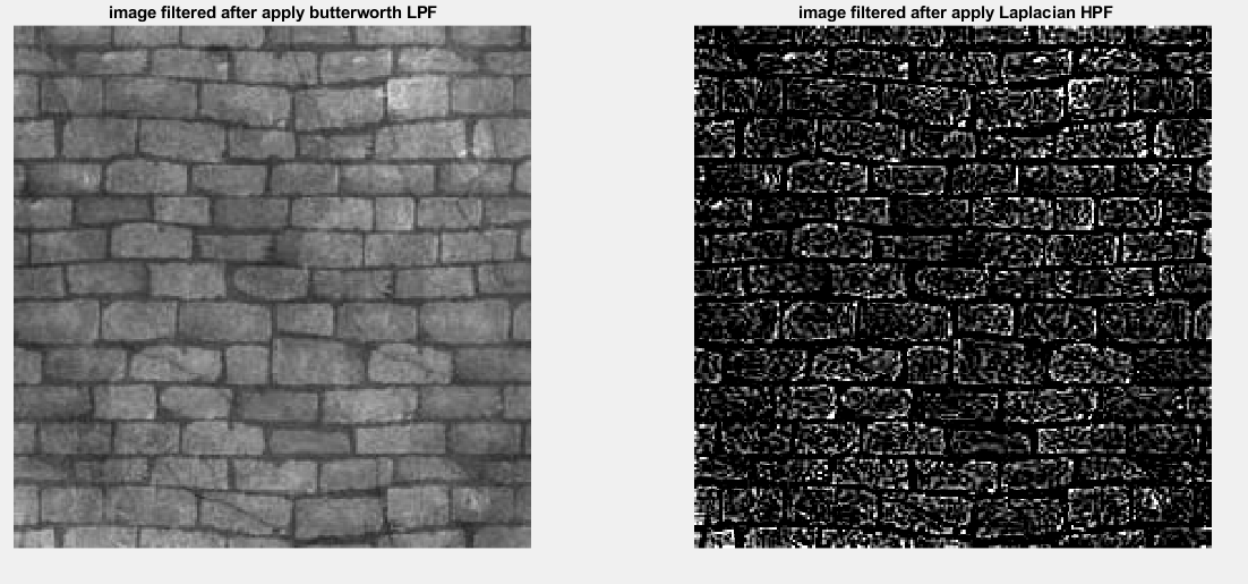


Figure 14: butterworth filter

Based on our application, we can use each of above filters.

E)

As we see in this course, ideal filter has sharp cut off and remove all frequency but other filters have a smooth cut off frequency and Gradually zero the frequency.

another reason is inverse FFT of the ideal filter is sinc. so, it can cause ringing in the image.

Refrence:

1. *https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123*
2. *https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm*
3. *https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm*