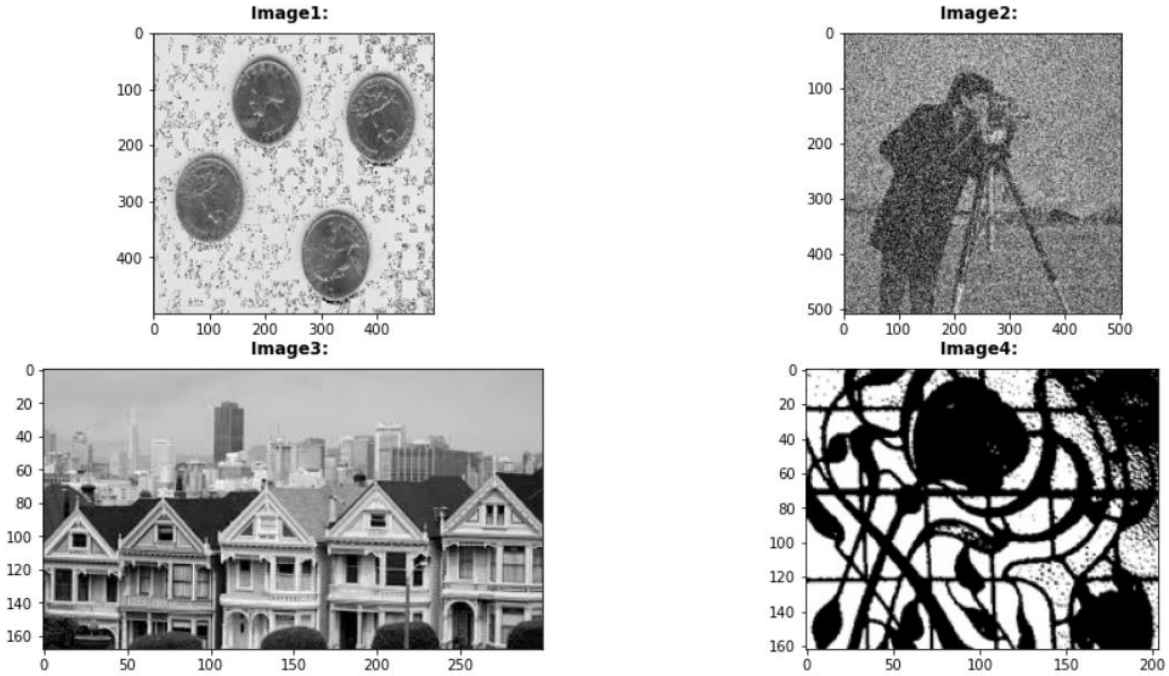
پردازش تصویر(دکتر ساجدی)– تمرین دوم

فرشاد برجعلی زاده – ۶۱۰۳۹۹۰۱۵

در تمرین قصـــد داریم بر روی عکس های داده شـــده چندین فیلتر را اعمال کنیم در قســـمت اول Filtering in Spatial Domain و در قسمت دوم Filtering in Frequency Domain را بررسی می کنیم.

در این مجموعه داده ۶ عکس وجود دارد



در بالا ۴ عکس اول دیتاست داده شده را مشاهده میکنیم.

مطابق همه قسمت های برنامه نویسی ابتدا کتابخانه های مورد نظر را وارد میکنیم.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

سپس چهار عکس ابتدایی را خوانده

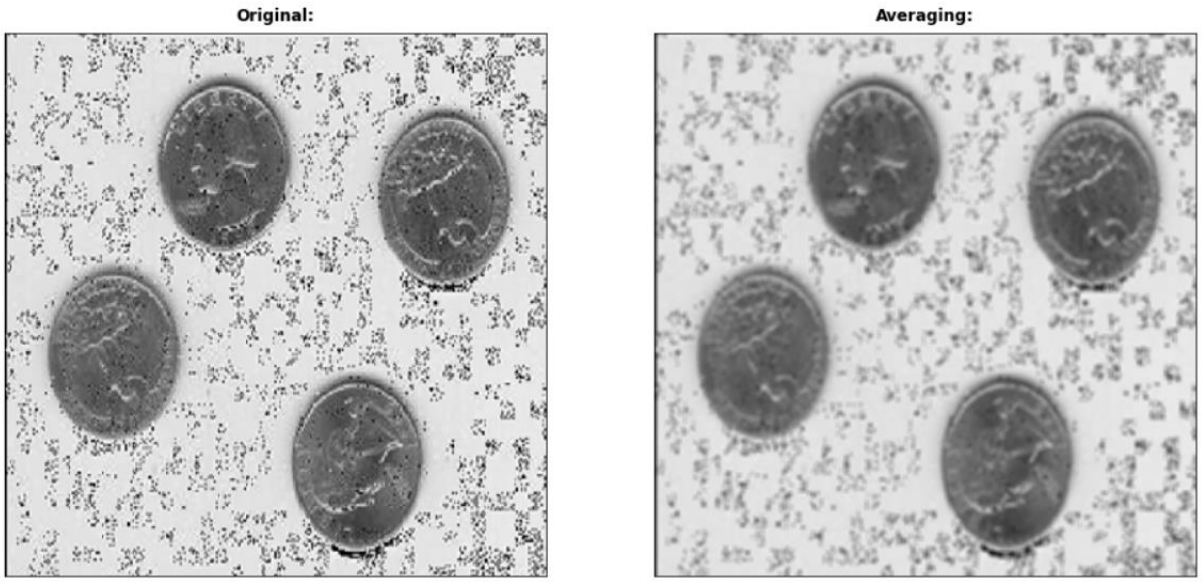
```
#Read original images
img1 = cv2.imread('images/image1.jpg')
img2 = cv2.imread('images/image2.jpg')
img3 = cv2.imread('images/image3.jpg')
img4 = cv2.imread('images/image4.jpg')
```

برای اعمال Average filtering با اندازه کرنل ۵*۵ ابتدا ماتریسی با درایه های ۱ تعریف میکنیم و از آنجایی که مجموع درایه های آن ۲۵ می شود تمام درایه ها را بر ۲۵ تقسیم کرده و سپس با تابع filter2D **عکس اول را با ماتریس میانگین ساخته شده ترکیب می کنیم.**

```
kernel = np.ones((5,5), np.float32)/25

average = cv2.filter2D(img1,-1,kernel)
```

نتیجه حاصل را در تصویر پایین مشاهده می کنیم.



در قسمت بعد Median filter را با اندازه کرنل ۵*۵ بر روی عکس دوم اعمال می کنیم برای این کار می توان از تابع از پیش تعریف medianBlur **استفاده کرد، نتیجه را در عکس پایین مشاهده می کنیم.**

Original:

Median:

برای پیاده سازی **Laplacian filter** با اندازه کرنل ۵*۵ هم میتوان از تابع **cv2.Laplacian** استفاده کرد.

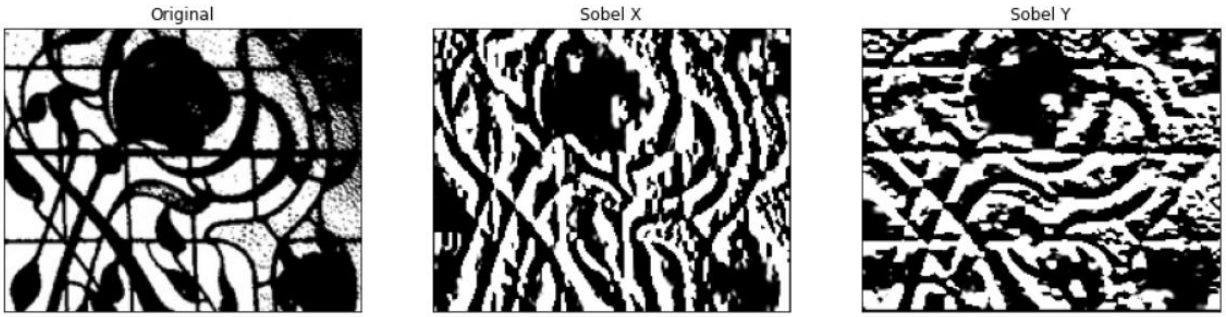
```
laplacian = cv2.Laplacian(img3, ddepth=0 , ksize=5)
```

Original:

laplacian:

در قسمت آخر هم برای پیاده سازی **Sobel(X and Y) Filter** میتوان از قطعه کد مقابل استفاده کرد.

```
sobelx = cv2.Sobel(img4, 0, 1, 0, ksize=7)
sobely = cv2.Sobel(img4, 0, 0, 1, ksize=7)
```

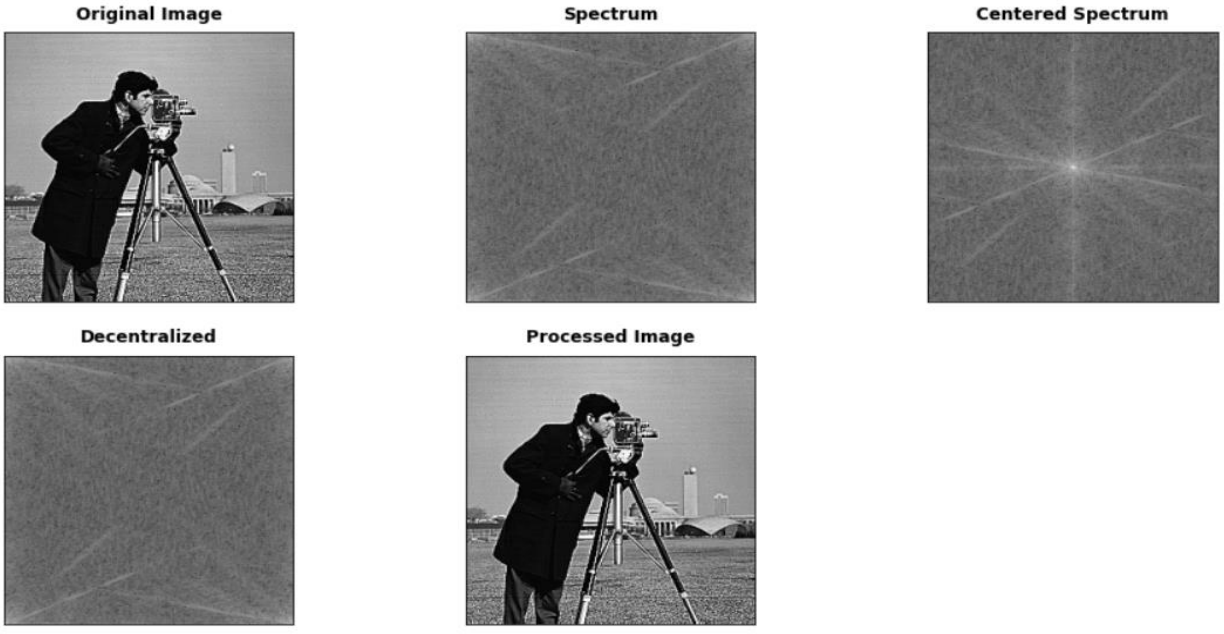


نتایج حاصل از اعمال Sobel(X and Y) Filter بر روی عکس چهارم

در قسمت Filtering in Frequency Domain ابتدا عکس پنجم را خوانده و سپس با استفاده از توابع موجود در کتابخانه numpy الگوهای Fast Fourier Transformation را بر روی تصویر اعمال می کنیم.

```
img5 = cv2.imread("images/image5.jpg", 0)

img_c2 = np.fft.fft2(img5)
img_c3 = np.fft.fftshift(img_c2)
img_c4 = np.fft.ifftshift(img_c3)
img_c5 = np.fft.ifft2(img_c4)
```



برای قسمت Ideal Low Pass Filter از همان شیوه فلیتر کردن به روش الگوریتم میانگین عمل می کنیم

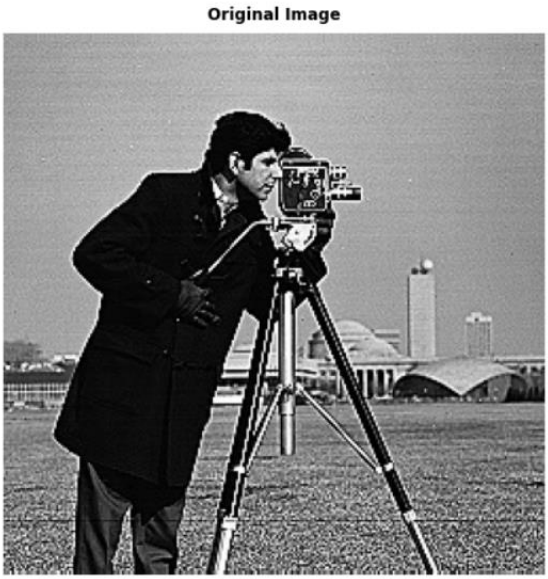
```
#Read image
img5 = cv2.imread('images/image5.jpg')

#prepare the 5x5 shaped filter
kernel = np.array([[1, 1, 1, 1, 1],
                   [1, 1, 1, 1, 1],
                   [1, 1, 1, 1, 1],
                   [1, 1, 1, 1, 1],
                   [1, 1, 1, 1, 1]])

kernel = kernel/sum(kernel)

#filter the source image
img_rst = cv2.filter2D(img,-1,kernel)
```

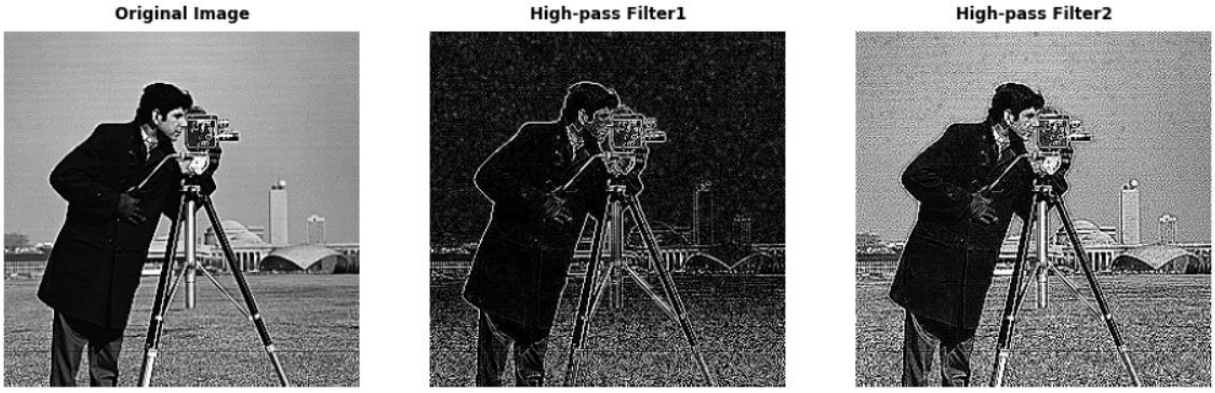
و نتیجه آن بدین صورت می باشد.



برای قسمت Ideal High Pass Filter از دو کرنل متفاوت استفاده کردیم که آن کرنل ها بدین صورت است

```
#edge detection filter
kernel = np.array([[0.0, -1.0, 0.0],
                   [-1.0, 4.0, -1.0],
                   [0.0, -1.0, 0.0]])

kernel2 = np.array([[0.0, -1.0, 0.0],
                   [-1.0, 5.0, -1.0],
                   [0.0, -1.0, 0.0]])
```



نتایج حاصل از اعمال high-pass filter

در قسمت آخر از این تمرین هم می خواهیم فیلتر Gaussian Smoothing را بر روی تصویر۶ با اندازه کرنل های متفاوت اعمال کنیم، برای این کار از تابع cv2.blur **میتوان استفاده کرد.**