Image Sharpening

電機所　609415159　江昀融

Date due:2020/11/16

Date handed in:2020/11/16

Technical description:

使用 opencv 用於圖片的基本讀寫與顯示
numpy 用於進行陣列操作
math 用於影像處理的公式計算
matplotlib 用於圖表的顯示
scipy.fftpack 用於傅立葉轉換

```
import cv2 as cv
import numpy as np
import math
from matplotlib import pyplot as plt
from scipy.fftpack import fft,ifft
```

cv.imread 讀取輸入圖片
並且讀取圖片長寬並顯示

```
#choose read which image
img = cv.imread('blurry_moon.tif', cv.IMREAD_GRAYSCALE)
#img = cv.imread('skeleton_orig.bmp', cv.IMREAD_GRAYSCALE)

#print height and width of input image
height, width = img.shape
print(type(img))
print(img.shape)
print("height = ", height)
print("width = ", width)
```

將原圖做二維傅立葉轉換
並存入 fft2

```
#Fourier transform
fft2 = np.fft.fft2(img)
lap_fft2 = np.fft.fft2(img)
plt.subplot(231)
plt.imshow(np.abs(fft2),'gray'),plt.title('fft2')
```

```
#Centralization
shift2center = np.fft.fftshift(fft2)
shift2center[int((height/2)-1) : int((height/2)+1), int((width/2)-1) : int((width/2)+1)] = 0
plt.subplot(232)
plt.imshow(np.abs(shift2center),'gray'),plt.title('shift2center')
```

將得到的頻譜進行中心化，即將低頻訊號集中至中心

```
#laplacian sharpening
for i in range(height):
    for j in range(width):
        lap_fft2[i][j] = -4*(math.pi**2)*abs((i-height/2)**2 + (j-width/2)**2)*shift2center[i][j]
plt.subplot(233)
plt.imshow(np.abs(lap_fft2),'gray'),plt.title('lap_fft2')
```

$$\nabla^2 f(x,y) \Leftrightarrow -4\pi^2\left|(u-M/2)^2+(v-N/2)^2\right|F(u,v).$$

將中心化後的頻譜做拉普拉斯轉換，並以上方的公式進行運算，也在計算中改變濾波器中心

```
#Inverse Centralization
center2shift = np.fft.ifftshift(lap_fft2)
plt.subplot(234)
plt.imshow(np.abs(center2shift),'gray'),plt.title('center2shift')
```

將做完拉普拉斯的頻譜進行逆中心化

```
#Inverse Fourier transform
ifft2 = np.fft.ifft2(center2shift)
plt.subplot(235)
plt.imshow(np.abs(ifft2),'gray'),plt.title('ifft2')
```

將逆中心化的頻譜進行傅立葉逆轉
並且針對三種不同的影像處理技術進行分別的動作:

1.sharpen images by Laplacian operator

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y) \\ f(x,y)+\nabla^2 f(x,y) \end{cases}$$

以右方的公式將原圖與
特徵圖相加

```
#normalization & image enhancement
lap_img = np.abs(ifft2)/np.max(np.abs(ifft2))
result_img = lap_img + (img/255)
```

2.by unsharp masking

$$f_{hp}(x,y)=f(x,y)-f_{lp}(x,y)$$

以右方的公式將原圖與
特徵圖相減

```
#normalization & unsharp masking
lap_img = np.abs(ifft2)/np.max(np.abs(ifft2))
result_img = (img/255) - lap_img
```

3.by high-boost filtering

$$f_{hb}(x,y)=Af(x,y)-f_{lp}(x,y)$$

以右方公式將原圖乘以
A 倍並減去特徵圖

```
A = 2
#normalization & High-Boost
lap_img = np.abs(ifft2)/np.max(np.abs(ifft2))
result_img = A*(img/255) - lap_img
```

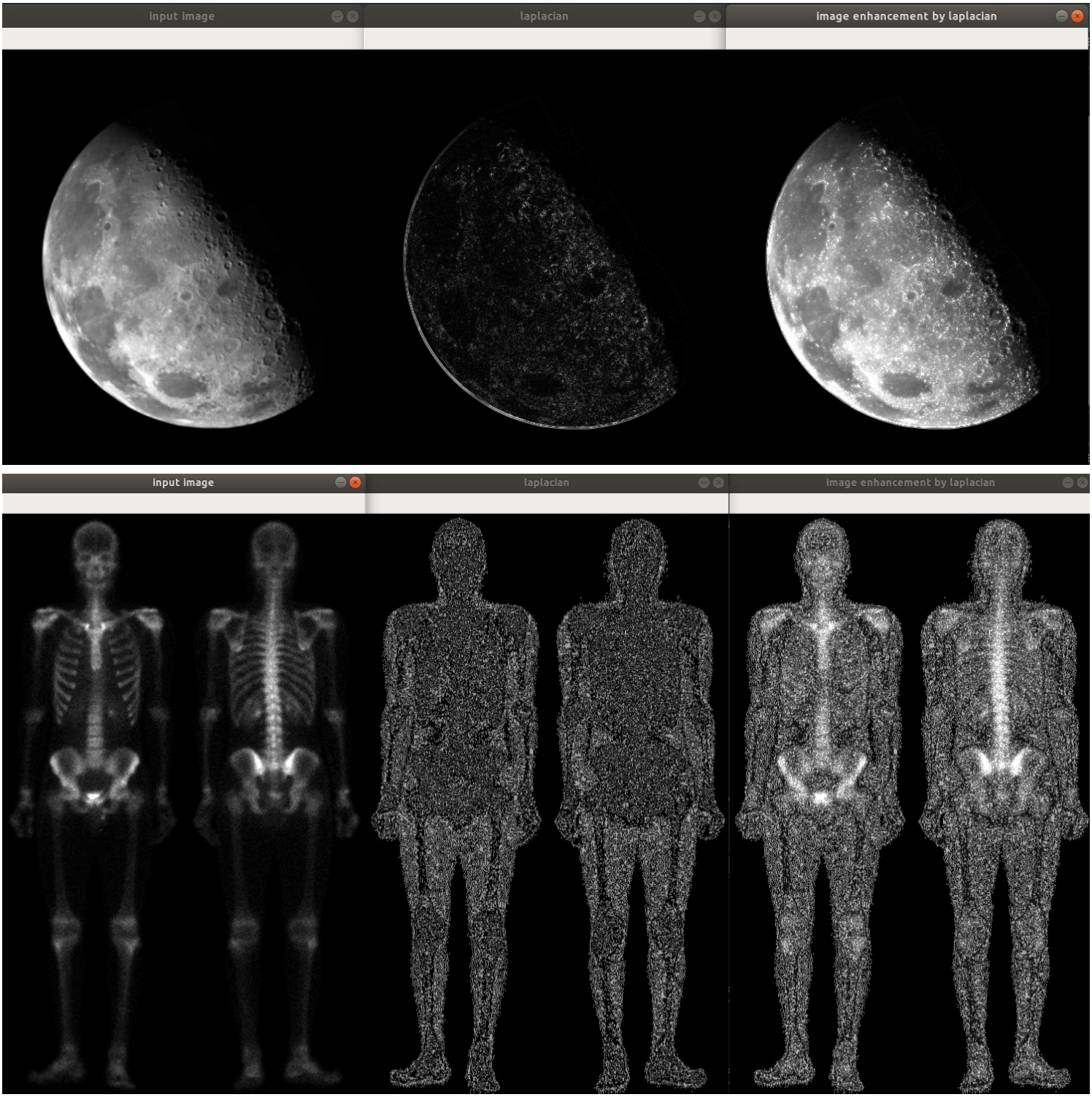
將三種方法的結果以同樣的程序做結尾:

最後整理圖表並顯示

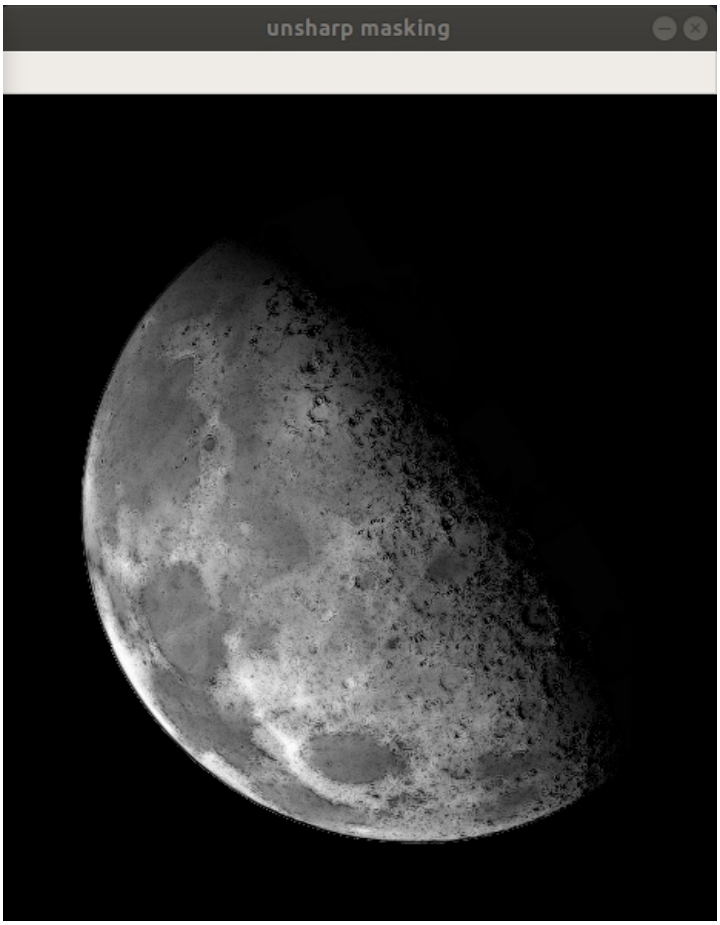
```
#show image
plt.subplots_adjust(left=0.125,
                    bottom=0.1,
                    right=0.9,
                    top=0.9,
                    wspace=0.2,
                    hspace=0.35)
cv.imshow('input image', img)
cv.imshow('laplacian', lap_img)
cv.imshow('image enhancement by laplacian', result_img)

#destroy Windows
cv.waitKey(0)
cv.destroyAllWindows()
```

Experimental results:

1.Laplacian operator



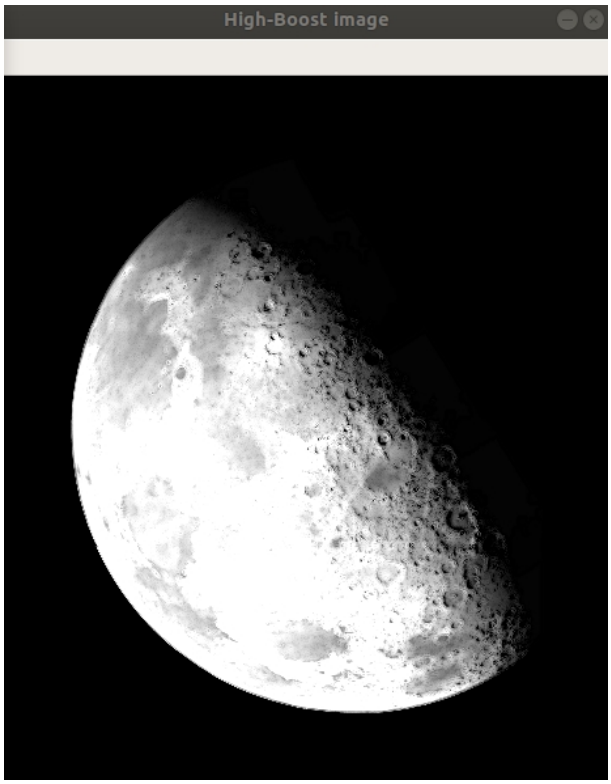
2.unsharp masking



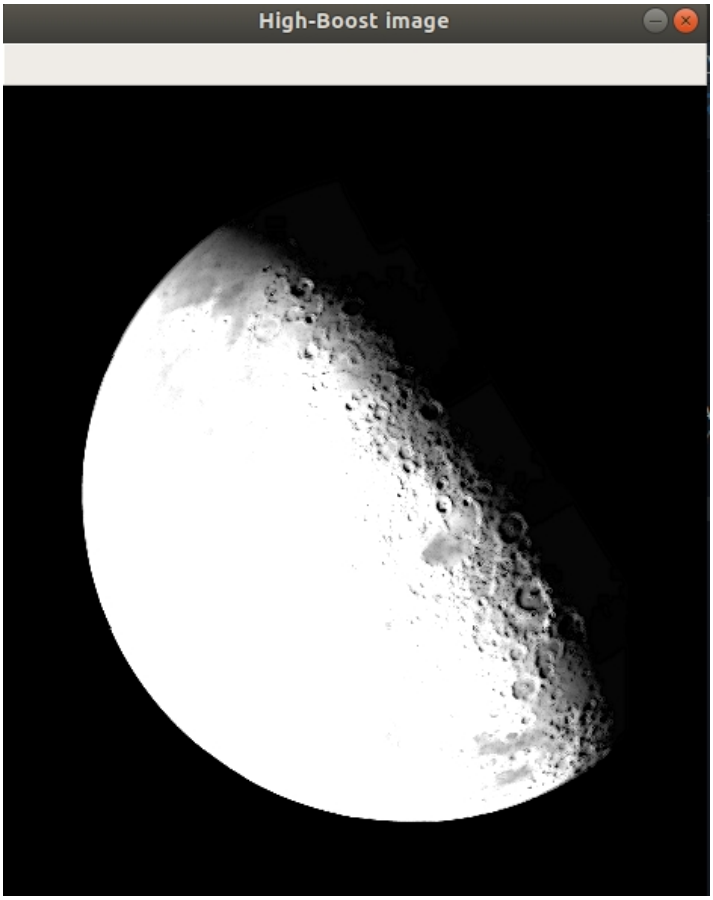
high-boost filtering

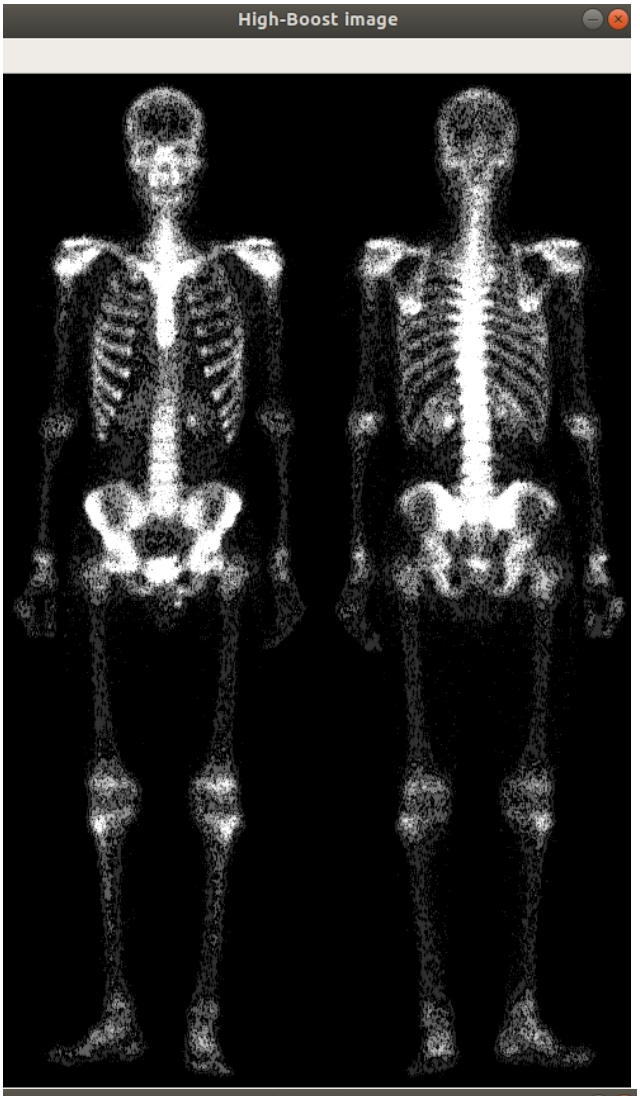
A = 1，即 unsharp masking 之結果

A = 2

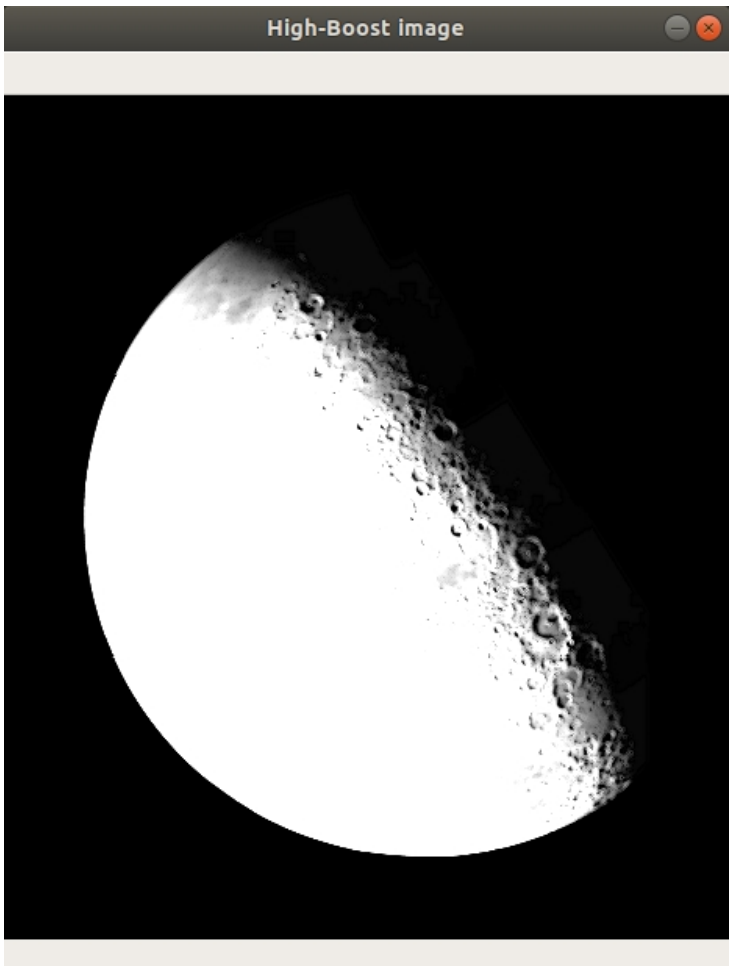


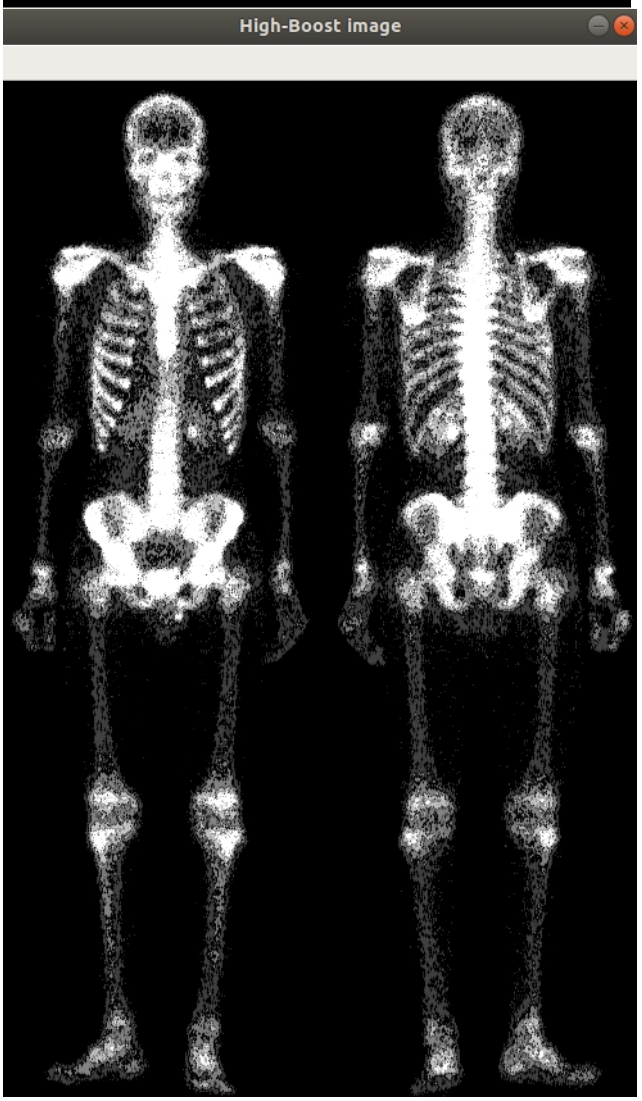
A = 3


High-Boost image


High-Boost image

A = 4


High-Boost image


High-Boost image

Discussions:

在上一次的作業是在圖像的空間座標做轉換，比起頻域相對直觀且好理解，所以上一次的作業做的比較輕鬆愜意。但這次的頻域轉換則進入了之前所沒接觸過得領域，光是經過傅立葉轉換到頻域的過程就很複雜難理解，而轉換後的頻域圖更是沒辦法明顯的看出與原圖的關聯，還得經過中心化，並且在做頻域資料的轉換過程也要注意中心的轉移，操作上比空間上複雜許多。經過老師的上課內容與網路上的查詢資料，花了很多時間實驗各種傅立葉轉換與中心轉移與遮罩等等的影響，最後終於做出與老師所展現的結果相似的圖形。
在這次的作業中做了三種轉換過程，並且三個轉換方程式都各自不一樣，為了搞懂轉換方程式也是花了很多時間。在拉普拉斯的基礎上有三種不同的操作，在研究完頻域的轉換後接下來就比較輕鬆了。三個方法分別為

1. Laplacian operator
使用拉普拉斯方程式，將已經過傅立葉轉換＆中心化的訊號進行轉換，最後再使用逆轉換還原訊號，將得到的特徵圖像加在原圖上得到增強影像
但同樣的有會增強雜訊的問題，因為雜訊經轉換後也是屬於高頻的部份，會跟著被放大

2. unsharp masking
這種的銳化遮罩是將原圖減去特徵圖，可以將雜訊抑制不被放大，但效果也比較一般

3. high-boost filtering
高增益濾波器是蕤化遮罩的加強版，它會將原圖以倍數相乘，使高頻部份更加的高頻，最後在減去特徵圖，對於抑制雜訊的功能有更好的改良

針對頻域的轉換有很多種，這次是針對於拉普拉斯做轉換，來取得圖像的特徵。雖然在嘗試的過程中，我覺的對於有些圖做完的特徵不太理想，或許是不適合用此方法，不然就是我程式打錯了xd
對於頻域還處於懵懵懂懂，以後可以在更詳細的研讀它。

References and Appendix:

https://jennaweng0621.pixnet.net/blog/post/404319692-%5Bpython-%2B-opencv%5D-%E5%82%85%E7%AB%8B%E8%91%89%E8%BD%89%E6%8F%9B-%28fourier-transform%29
https://www.itread01.com/content/1546988055.html
https://www.itread01.com/content/1548455257.html
https://www.itread01.com/content/1547200686.html