影像處理作業-HW4

學生：林正浩 學號：r11631029 日期：11/05

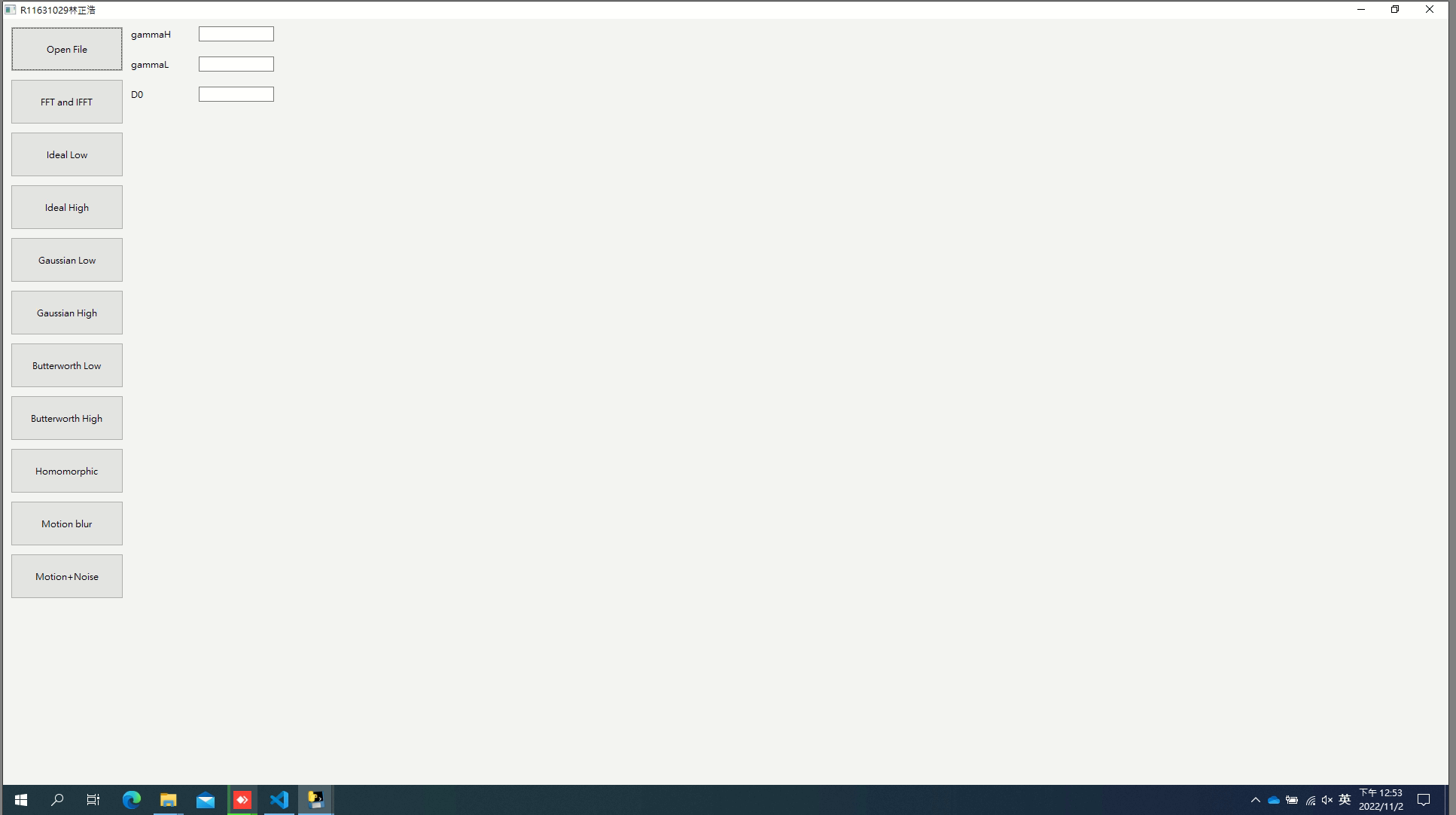
1. 介面介紹：

Part1主要就是「FFT and IFFT」的按鈕。

Part2就是按鈕「ideal Low」~「butterworth High」根據輸入欄位D0的數值去做變化。＊D0一定要有數值。

Part3為按鈕「homomorphic」根據輸入「gammaH, gammaL, D0」去進行運算。

Part4為最下面兩的按鈕，差別主要為是否有加入隨機雜訊。



1. Part1 傅立葉轉換及傅立葉反轉換

傅立葉轉換code:

Step: 轉換🡪平移🡪計算spectrum and phase angle

def FT(Gray\_img):

f = np.fft.fft2(Gray\_img) # fourier transform

f\_shift = np.fft.fftshift(f) # shift origin point

magnitude\_spectrum = 20 \* np.log(1 + np.abs(f\_shift)) # show specturm photo

phase\_angle = np.angle(f\_shift) # show phase angle fig

return f\_shift, magnitude\_spectrum, phase\_angle # 平移後的fourier transform, 頻譜圖, 相角圖

傅立葉反轉換code:

def FIT(sample\_after\_FT):

f\_ishift = np.fft.ifftshift(sample\_after\_FT) # shift again for one shift when has run FT.

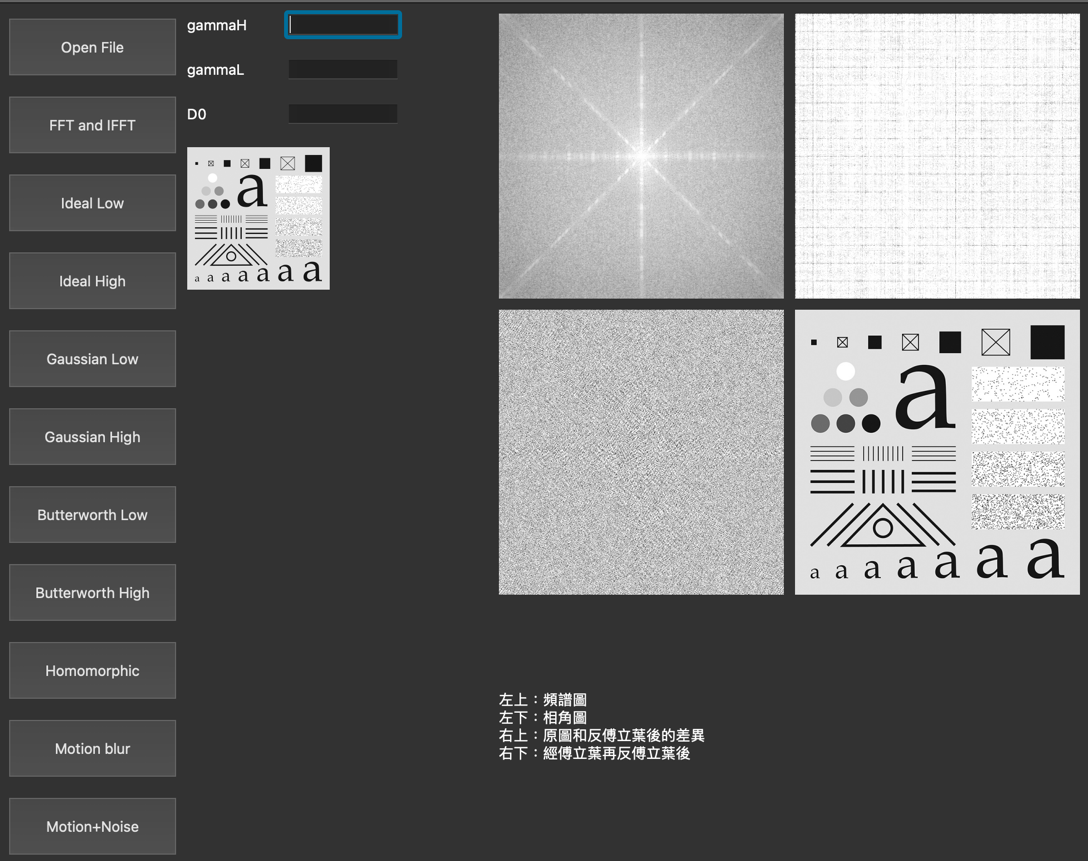
img\_back = np.fft.ifft2(f\_ishift) # fourier inverse transform

img\_back = np.abs(img\_back) # show image of FIT

return img\_back # fourier 逆轉換後的照片

運行結果：

發現使用numpy本身的傅立葉轉換跟反傅立葉後與原圖雖然肉眼看不出差異，但仍是有差異存在。

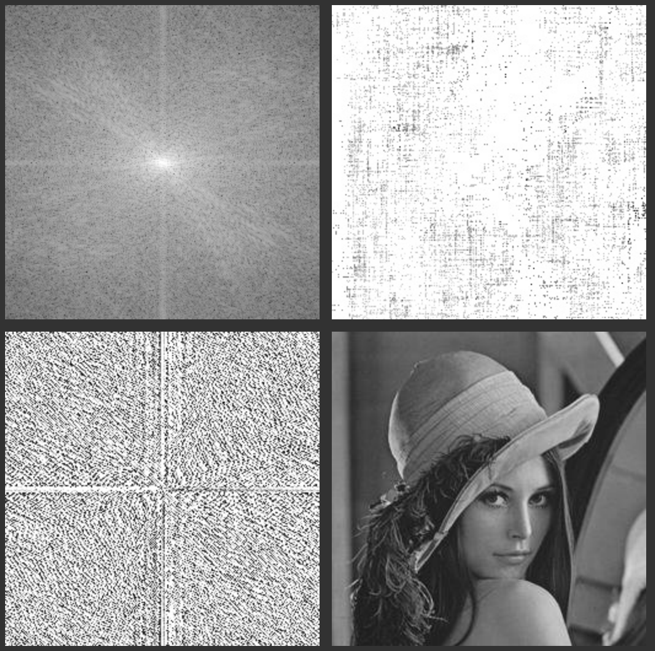
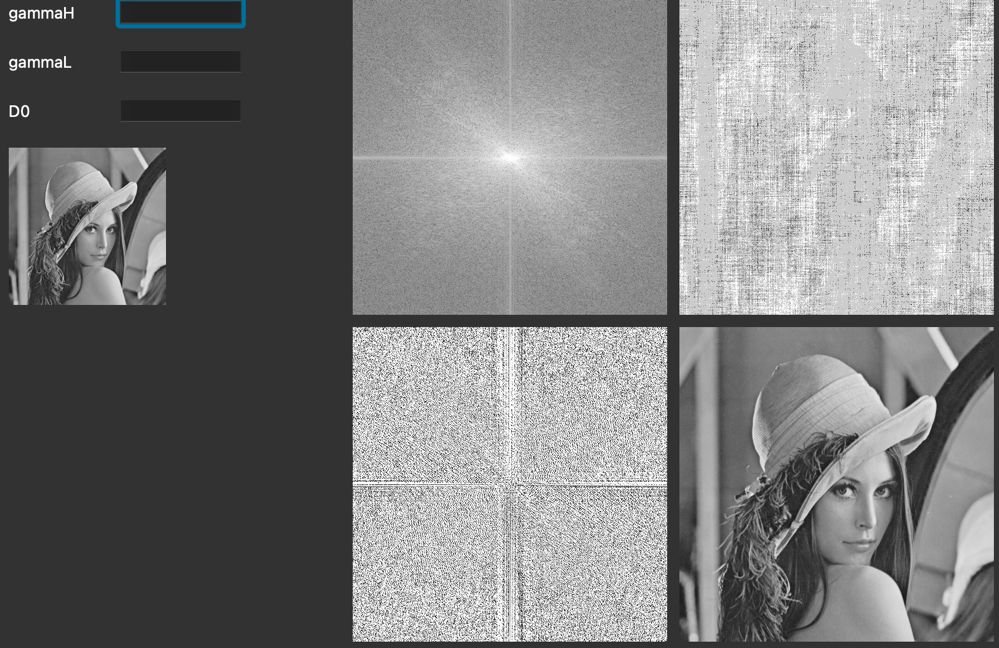


使用兩張不同size的照片進行：

256\*256需要：0.00776600837707519 s

512\*512需要：0.03227710723876953 s

時間差大約4~5倍。

1. Part2 highpass and lowpass mask
2. Ideal filter:

# ideal\_pass(gray\_img, radius, Low or High)

def ideal\_Pass(img\_gray, radius, Type): # ideal Lowpass, 設定一個頻率域的threshhold, 去進行保留或去除

# set parameter

h, w = img\_gray.shape

center\_X = h/2

center\_Y = w/2

R = radius

# Fourier Transform

fft\_Shift, magnitude\_spectrum, phase\_angle = FT(img\_gray)

# Run ideal pass --> classify Low or High

if (Type == 'Low'):

for i in range(h):

for j in range(w):

if ((i - center\_X)\*\*2 + (j - center\_Y)\*\*2 > R\*\*2):

fft\_Shift[i, j] = 0

else:

for i in range(h):

for j in range(w):

if ((i - center\_X)\*\*2 + (j - center\_Y)\*\*2 < R\*\*2):

fft\_Shift[i, j] = 0

# Fourier inverse transform

img = FIT(fft\_Shift)

# spectrum

specturm = 20 \* np.log(1 + np.abs(fft\_Shift))

return img, specturm

1. Gaussian filter:

def gaussian\_pass(img\_gray, D0, Type):

# set parameter

h, w = img\_gray.shape

center\_X = h/2

center\_Y = w/2

# Fourier Transform

fft\_Shift, magnitude\_spectrum, phase\_angle = FT(img\_gray)

# Run gaussian pass --> classify Low or High

if (Type == 'Low'):

val, direct = (0, 1)

else:

val, direct = (1, -1)

for i in range(h):

for j in range(w):

fft\_Shift[i, j] = fft\_Shift[i, j] \* (val + direct \* (np.exp(-((i - center\_X)\*\*2 + (j - center\_Y)\*\*2) / (2 \* (D0\*\*2)))))

# Fourier inverse transform

img = FIT(fft\_Shift)

# spectrum

specturm = 20 \* np.log(1 + np.abs(fft\_Shift))

return img, specturm

1. Butterworth filter:

def butterworth\_pass(img\_gray, D0, n, Type):

# set parameter

h, w = img\_gray.shape

center\_X = h/2

center\_Y = w/2

# Fourier Transform

fft\_Shift, magnitude\_spectrum, phase\_angle = FT(img\_gray)

# Run gaussian pass --> classify Low or High

if (Type == 'Low'):

val, direct = (0, 1)

else:

val, direct = (1, -1)

for i in range(h):

for j in range(w):

fft\_Shift[i, j] = fft\_Shift[i, j] \* (val + direct \* (1 / (1 + (((i - center\_X)\*\*2 + (j - center\_Y)\*\*2) / D0)\*\*n)))

# Fourier inverse transform

img = FIT(fft\_Shift)

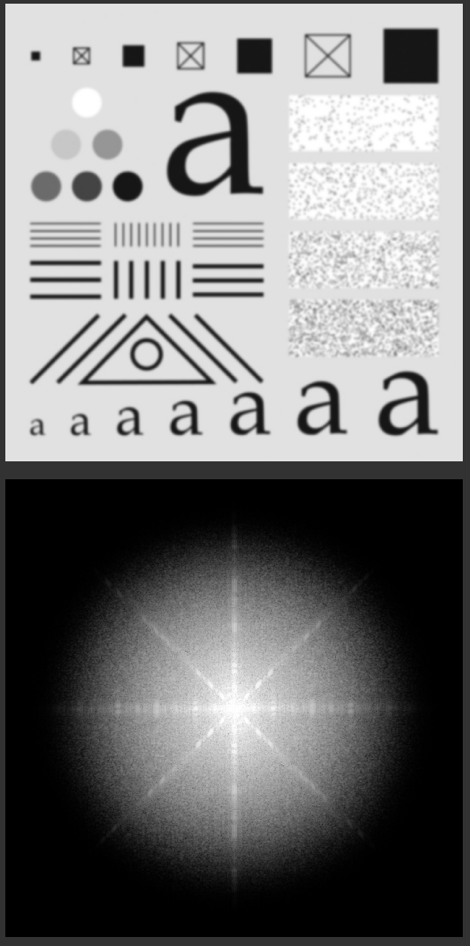
# spectrum

specturm = 20 \* np.log(1 + np.abs(fft\_Shift))

return img, specturm

根據課本及上課講義上的formular去調整每一個filter的算法。因為圖片眾多，故選擇用高斯作為報告例子。

D0為頻譜的半徑長度，發現當D0越小時，高斯低通濾波越強烈。由左至右是D0=10,30,60。

1. Part3 Homomorphic filter

def homomorphic(img\_gray, gammaH, gammaL, D0, c):

img\_gray = np.log(img\_gray) # get ln() for fig. 4.58

h, w = img\_gray.shape

center\_X = h/2

center\_Y = w/2

# Fourier Transform

fft\_Shift, magnitude\_spectrum, phase\_angle = FT(img\_gray)

# run homomorphic --> reference: https://www.twblogs.net/a/5bca7d8d2b71777351979e58

for i in range(h):

for j in range(w):

fft\_Shift[i, j] = fft\_Shift[i, j] \* ((gammaH - gammaL) \* (1 - np.exp(-c \* ((i - center\_X)\*\*2 + (j - center\_Y)\*\*2) / D0\*\*2)) + gammaL) # formular: 4-147

# Fourier inverse transform

img = FIT(fft\_Shift)

img = np.exp(img) # get exp() for fig. 4.58

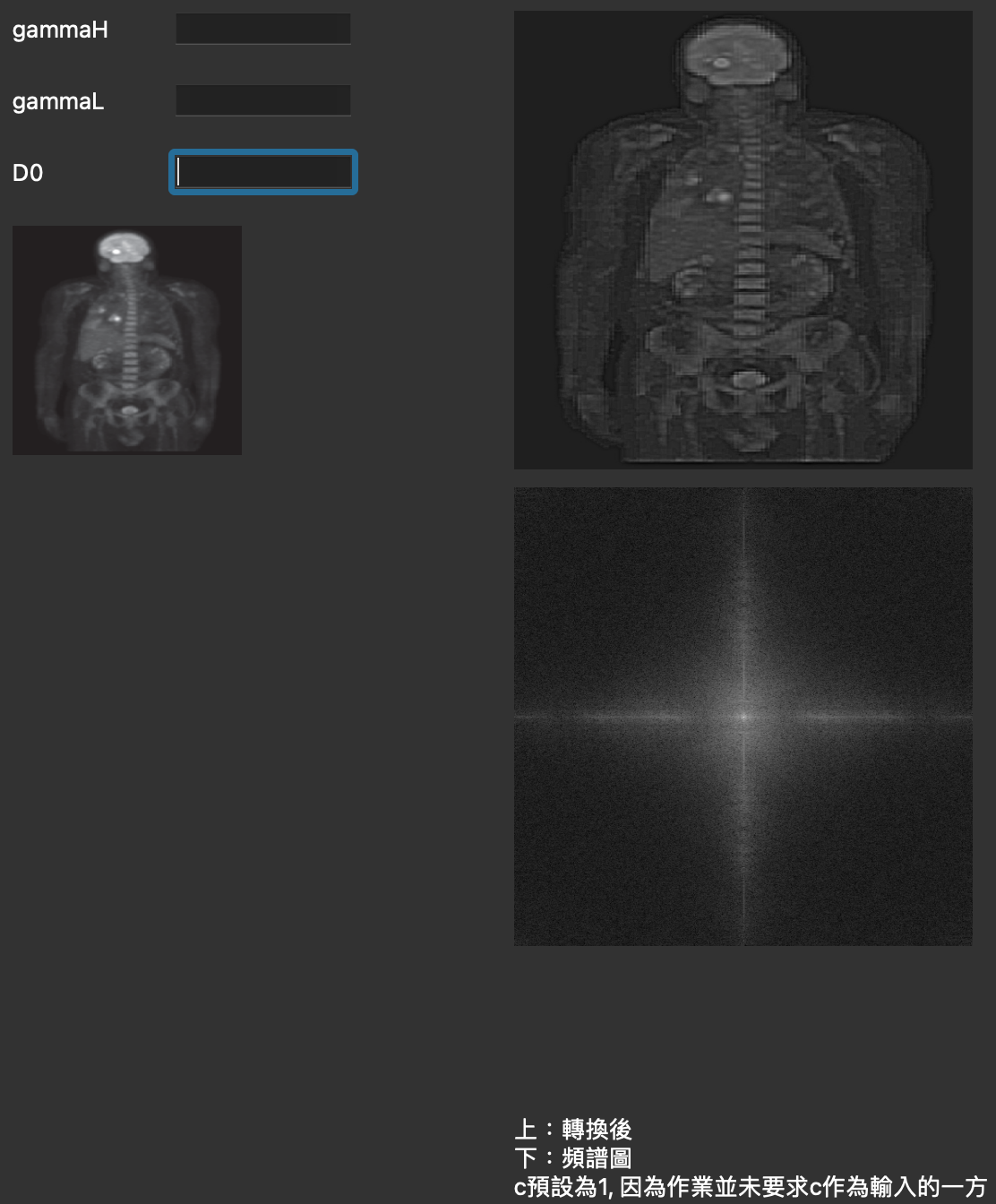
# spectrum

specturm = 20 \* np.log(1 + np.abs(fft\_Shift))

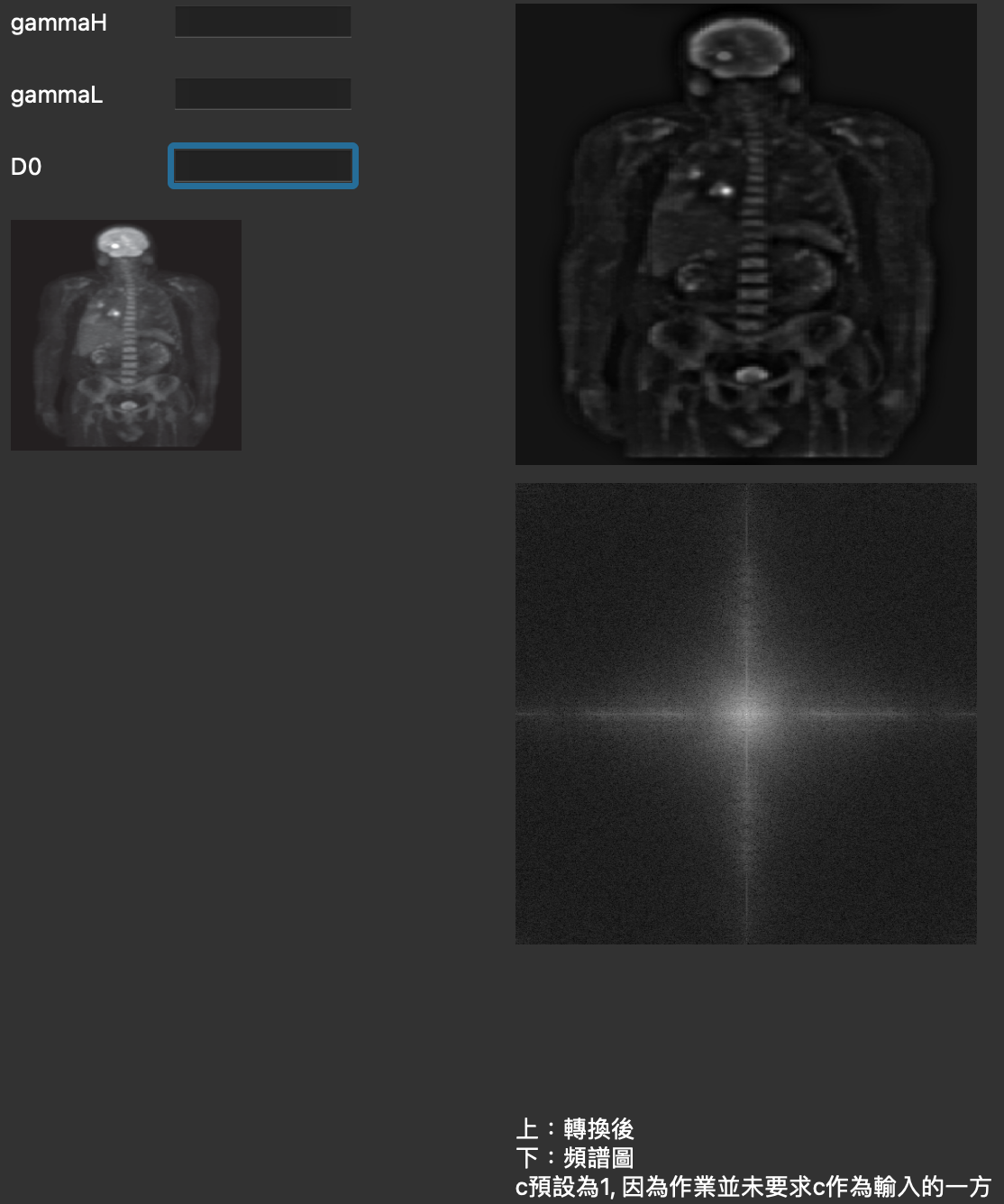
return img, specturm

根據課本內容， gammaL<1，gammaH>=1，能提升對比，達到會銳化效果。

GammaH = 2, gammaL = 0.2, D0 = 60



GammaH = 2, gammaL = 0.2, D0 = 10



1. Part4 修復移動模糊及隨機雜訊

根據課本公式將算法寫入for迴圈。

# motion blur

def motion\_blured(img\_gray, addNoise, T, a, b, K):

# Motion blur parameters

# addNoise: 1--> yes, 0 --> no

# T: duration of exposure

# a: vertical motion

# b: horizontal motion

# K: the wiener parameter, small is better

# FT

fft\_Shift, magnitude\_spectrum, phase\_angle = FT(img\_gray)

#-----Creat array H(u, v) "start"-----#

# H(u, v), motion blur function in frequency domain

# Create matrix H (motion blur function H(u, v))

M,N = fft\_Shift.shape

H = np.zeros((M + 1, N + 1), dtype = np.complex128) # +1 to avoid zero division

# Fill matrix H

for u in range(1, M + 1):

for v in range(1, N + 1):

w = np.pi \* (u \* a + v \* b)

H[u, v] = (T / w) \* np.sin(w) \* np.exp(-1j \* w) # formular 5-77

# index slicing

H = H[1 : , 1 : ]

#-----Creat array H(u, v) "end"-----#

# add noise

if addNoise == 1:

# G(u, v), blurred image in frequency domain

G = H \* fft\_Shift

# Fourier inverse transform

img = FIT(G)

# creat noise matrix

mean = 0

variance = 20 # 題目設定

noise = np.random.normal(mean, variance, img.size)

noise = noise.reshape(img.shape[0], img.shape[1]).astype('uint8')

img = img \* noise

G, magnitude\_spectrum, phase\_angle = FT(img)

else:

# G(u, v), blurred image in frequency domain

G = H \* fft\_Shift

# Fourier inverse transform

img = FIT(G)

# do inverse filter

# according to formular 5-78

F\_inverse = G / H

img\_after\_inverse = FIT(F\_inverse)

# do wiener filter

# according to formular 5-85

# K = 0.0001

F\_wiener = (1 / H) \* (np.abs(H)\*\*2 / (np.abs(H)\*\*2 + K)) \* G

img\_after\_wiener = FIT(F\_wiener)

# compare with inverse and wiener

img\_compare = img\_after\_inverse - img\_after\_wiener

return img, img\_after\_inverse, img\_after\_wiener

由下方結果可見，產生的效果與課本有所出入，相同的點是有產生殘影跟模糊。對於只有移動模糊的照片而言，inverse效果比wiener好很多，但我想這是應該的，因為我們本來就知道真正的Ｈ（u, v），當wiener的Ｋ=0時其實就是inverse，也就是說K🡪0效果會越好。

討論一下有移動跟雜訊模糊的照片，發現wiener效果好很多。上網搜尋一番也都表明，wiener對於噪點的處理比較好。



