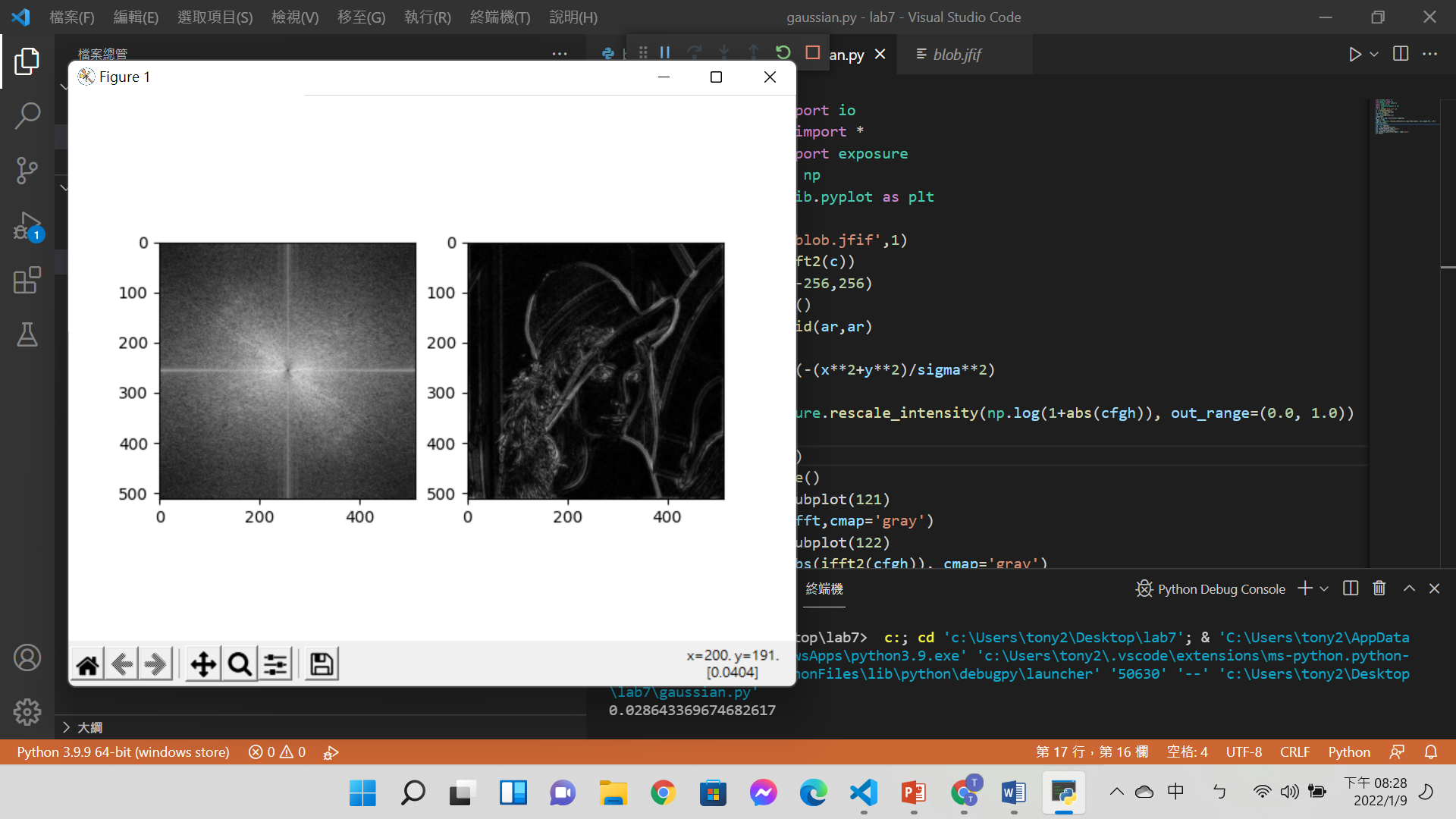
使用FFT製作Gaussian濾波器

花費0.028643369674682617秒



Code:

from skimage import io

from numpy.fft import \*

from skimage import exposure

import numpy as np

import matplotlib.pyplot as plt

import time

c = io.imread('blob.jfif',1)

cf = fftshift(fft2(c))

ar = np.arange(-256,256)

start=time.time()

x,y = np.meshgrid(ar,ar)

sigma=30.0

gh = 1.0-np.exp(-(x\*\*2+y\*\*2)/sigma\*\*2)

cfgh = cf\*gh

cfghfft = exposure.rescale\_intensity(np.log(1+abs(cfgh)), out\_range=(0.0, 1.0))

end=time.time()

print(end-start)

fig = plt.figure()

ax1 = fig.add\_subplot(121)

ax1.imshow(cfghfft,cmap='gray')

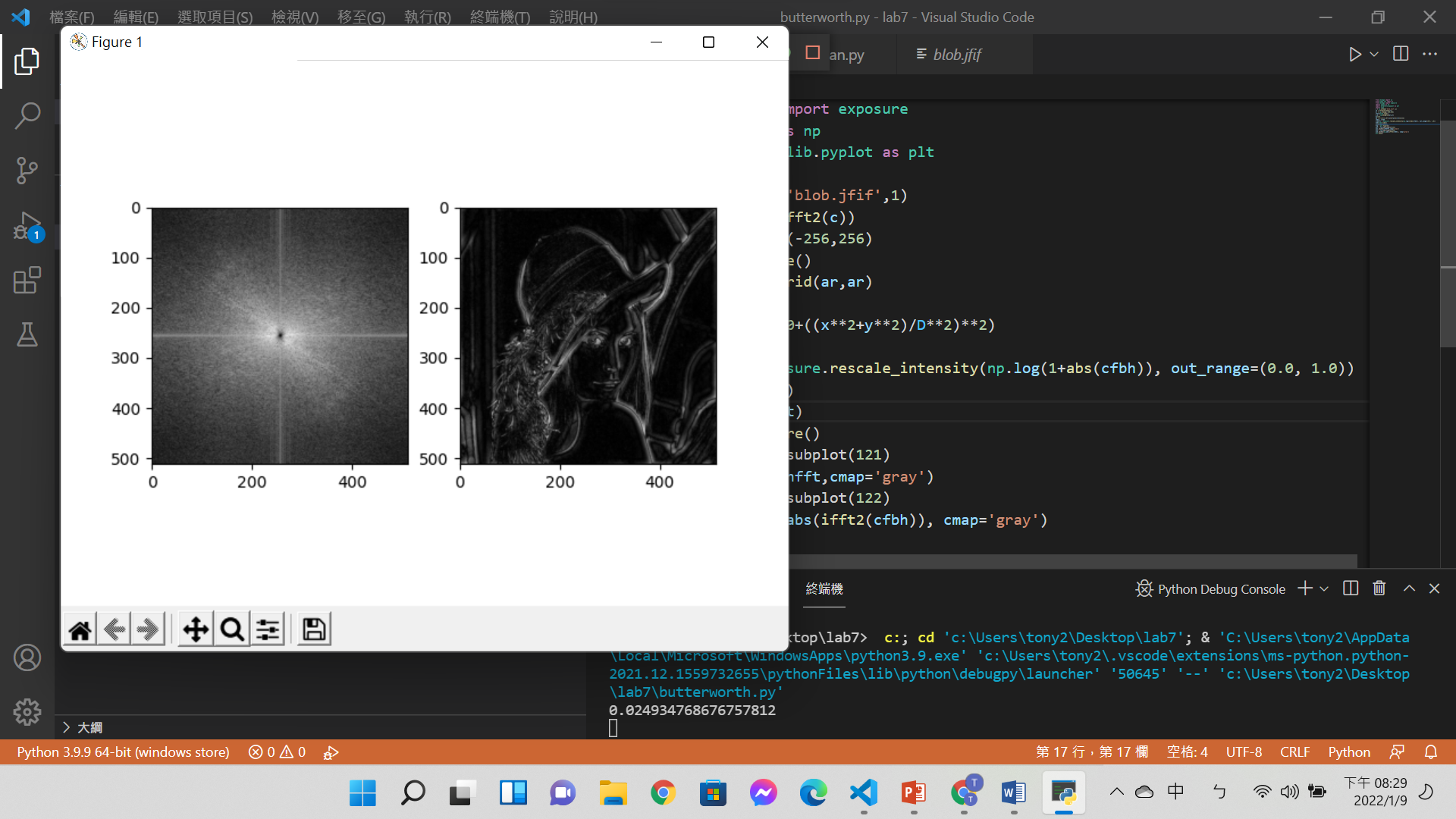
ax2 = fig.add\_subplot(122)

ax2.imshow(np.abs(ifft2(cfgh)), cmap='gray')

plt.show()

使用FFT製作**Butterworth**濾波器

花費0.024934768676757812秒



Code:

from skimage import io

from numpy.fft import \*

from skimage import exposure

import numpy as np

import matplotlib.pyplot as plt

import time

c = io.imread('blob.jfif',1)

cf = fftshift(fft2(c))

ar = np.arange(-256,256)

start=time.time()

x,y = np.meshgrid(ar,ar)

D=15.0

bh = 1-1.0/(1.0+((x\*\*2+y\*\*2)/D\*\*2)\*\*2)

cfbh = cf\*bh

cfbhfft = exposure.rescale\_intensity(np.log(1+abs(cfbh)), out\_range=(0.0, 1.0))

end=time.time()

print(end-start)

fig = plt.figure()

ax1 = fig.add\_subplot(121)

ax1.imshow(cfbhfft,cmap='gray')

ax2 = fig.add\_subplot(122)

ax2.imshow(np.abs(ifft2(cfbh)), cmap='gray')

plt.show()

比較上述兩種做法的執行結果差異及執行速度

Butterworth濾波器約比高斯濾波器快0.0035秒，且結果相近，可發現在這個狀況下Butterworth濾波器是比較有效率的。

心得:

雖然傅立葉轉換的數學方式還是有點不懂，但是運用其他人做好的函式還是能夠順利的完成這兩個濾波器，也從這次實驗發現了兩者濾波器執行速度的差異之大。