數位影像處理 DIP Homework Chapter 5_1

1. Please use **Inverse Filter and Wiener Filter** to correct the image corrupted severe turbulence of the assigned image, 'Fig5.25.jpg', and print out the source code and the corrected image? (40)

原圖 Fig5.25.tif

(這邊使用本書官網上的圖來做)

degradation function H 使用課本 p.357 頁 EXAMPLE 5.9 的公式

$$H(u,v) = e^{-k\left[(u - M/2)^2 + (v - N/2)^2\right]^{5/6}}$$

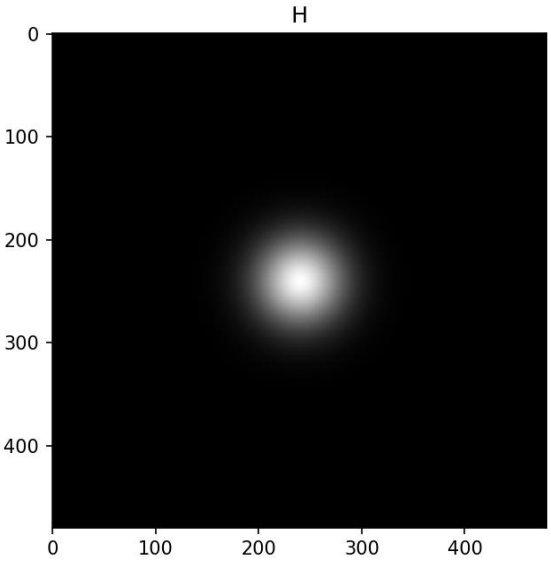
我認為這此公式有錯誤之處，公式裡次方項應該改成 **-M/2** 才對，而在實作上也要特別注意，次方項 u-M/2 及 v-N/2 皆是代表 u, v 做 shifting 的意思(移到影像中心處)，因此在建 **degradation function H** 有以下兩個方式:

```
M, N, c  = img.shape
k = 0.0025

## degradation function H(u, v)
x, y = np.mgrid[0:M, 0:N]
H = np.exp(-k * (((x-M/2)**2 + (y-N/2)**2)**(5/6)) )
```

```
M, N, c  = img.shape
k = 0.0025

## degradation function H(u, v)
x, y = np.mgrid[-M//2:M//2, -N//2:N//2]
H = np.exp(-k * (((x)**2 + (y)**2)**(5/6)) )
```

以上兩種方式皆會產生相同的下圖結果

```
plt.imshow(np.log(abs(H) + 1), cmap='gray'), plt.title("H")
plt.show()
```

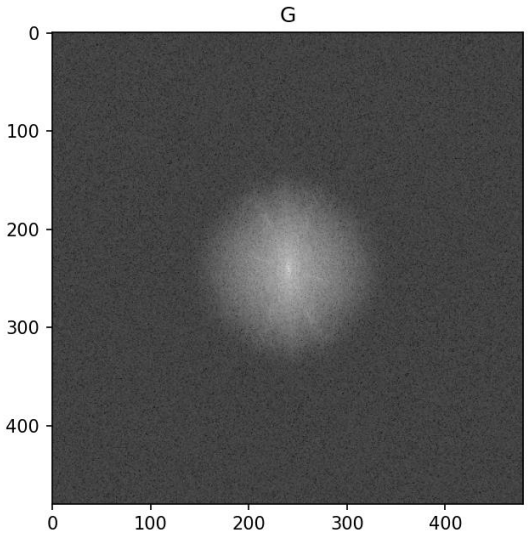


1. Inverse Filter

以課本 P.356 提到的 direct inverse filtering 公式(如下圖) 還原原圖

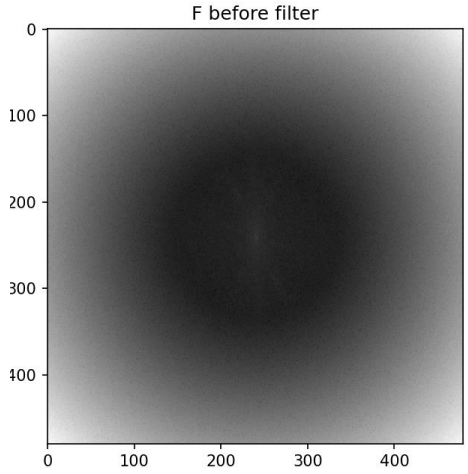
$$\hat{F}(u,v) = \frac{G(u,v)}{H(u,v)} \qquad (5\text{-}78)$$

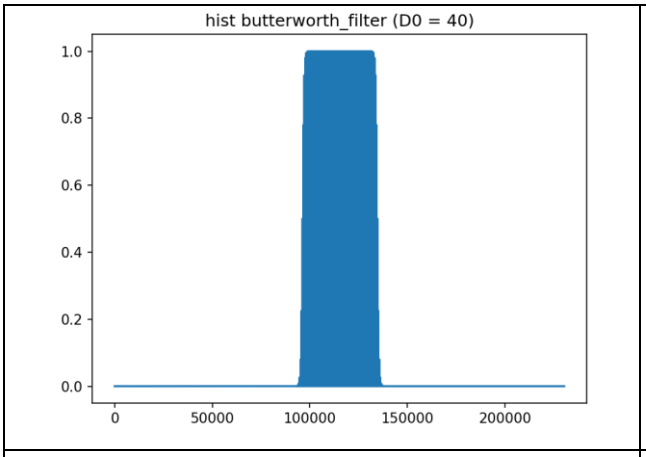
上式中的 G(u, v)為原圖 Fig5.25.tif 做 fourier transform 之後的結果(如下圖)

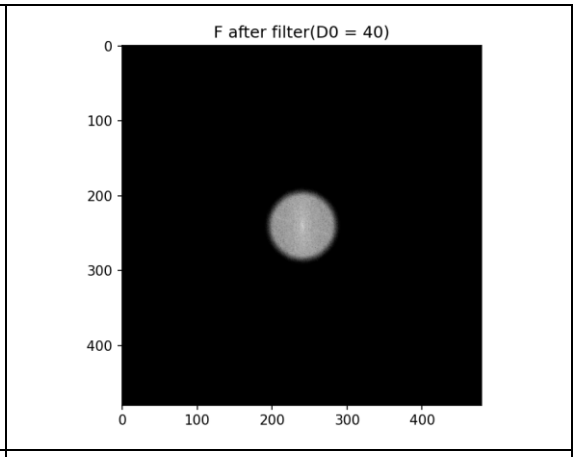


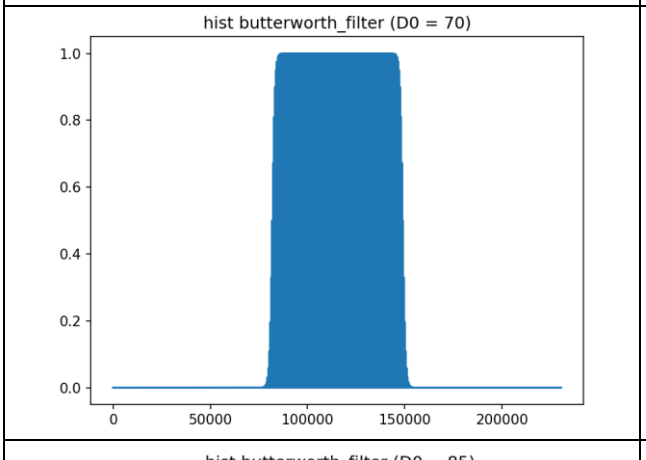
接著以 butterworth lowpass filter 對 F 做濾波以得到更好的結果，這裡同課本 EXAMPLE 5.9 實作三個 radius (or cutoff frequency)值 ：**40, 70, 85**

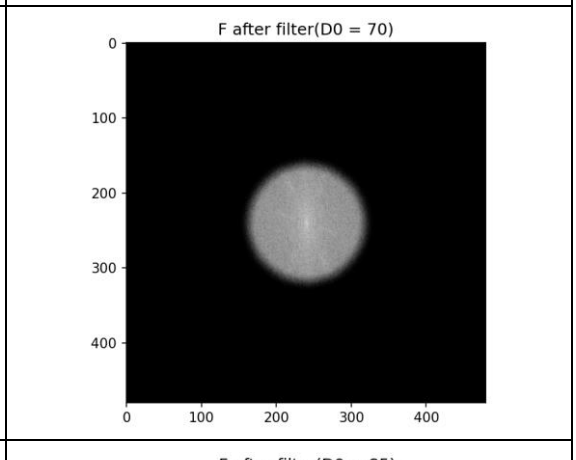
如下結果可以看到跟課本結論不同，order 固定為 25 時，**radius 越大，可得到越清楚的還原圖。**

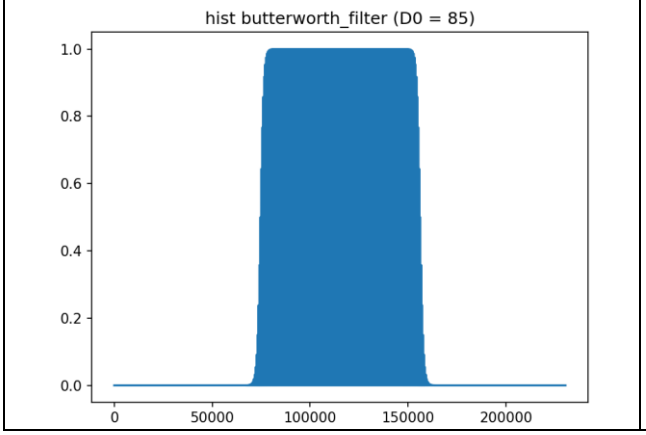

F before filter
0
100
200
300
400
0
100
200
300
400

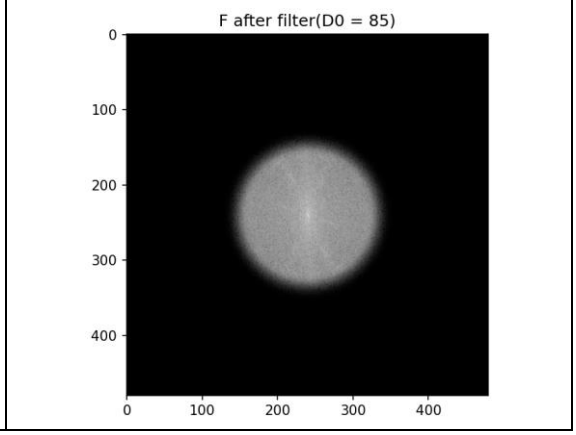

hist butterworth_filter (D0 = 40)
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000


F after filter(D0 = 40)
0
100
200
300
400
0
100
200
300
400


hist butterworth_filter (D0 = 70)
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000

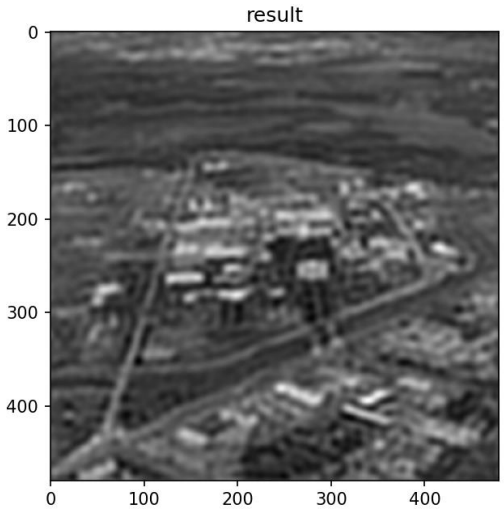

F after filter(D0 = 70)
0
100
200
300
400
0
100
200
300
400


hist butterworth_filter (D0 = 85)
1.0
0.8
0.6
0.4
0.2
0.0
0
50000
100000
150000
200000

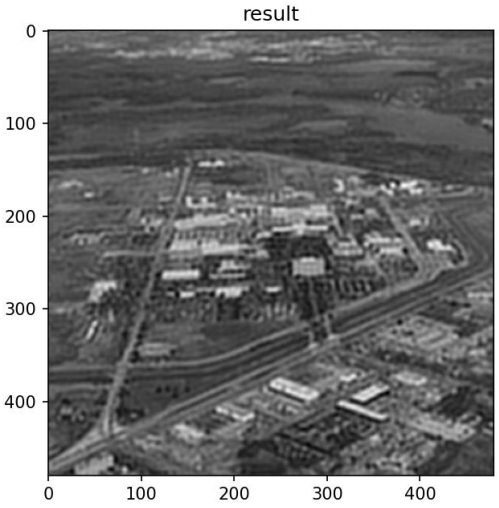

F after filter(D0 = 85)
0
100
200
300
400
0
100
200
300
400

最後對經 Butterworth lowpass filter 後的 F 做 Inverse Fourier Transform 後得到下圖結果(選用 D0(cutoff frequency) = 85 的結果最好)

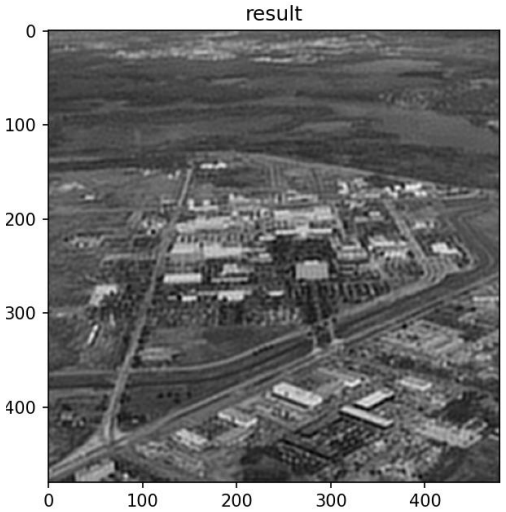
butterworth_filter (D0 = 40, n = 25)



butterworth_filter (D0 = 70, n = 25)



butterworth_filter (D0 = 85, n = 25)



Butterworth 程式碼

```
def butterworth(img, D0, n, mode="lowpass"):
    # D0 is cut-off frequency, n is order, mode=lowpass/highpass
    M, N = img.shape
    x, y = np.mgrid[-M//2:M//2, -N//2:N//2]
    bw_filter = 1/(1+((x**2+y**2)/D0**2)**(n))   # 課本 p.282公式
    plt.plot(bw_filter.ravel()), plt.title("hist butterworth_filter (D0 = 85)")
    plt.show()
    if mode == "lowpass":
        return bw_filter
    else:
        return (1 - bw_filter)
```

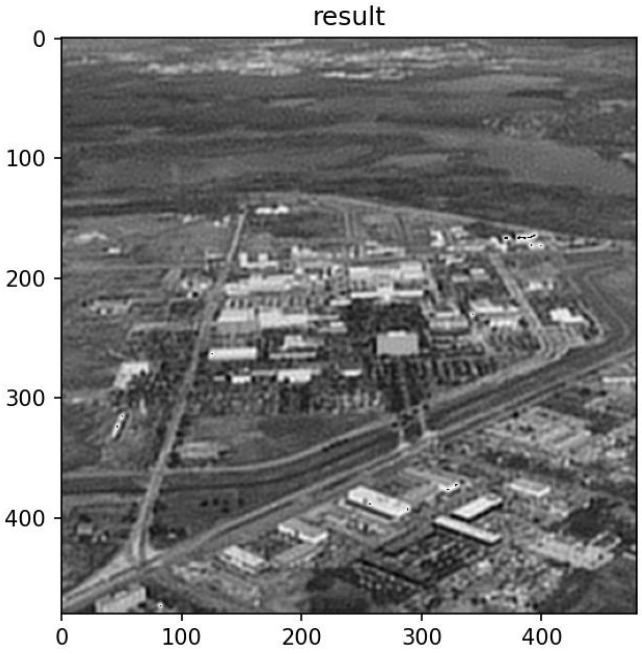
2. Wiener Filter

使用課本 p.360 的公式建 Wiener Filter，K 選用 0.0001 可得到最佳的結果

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right]G(u,v) \qquad (5\text{-}85)$$

```
## wiener filter
K = 0.0001
W = (1/H)*(abs(H)**2/(abs(H)**2+K))
F = W*G
```

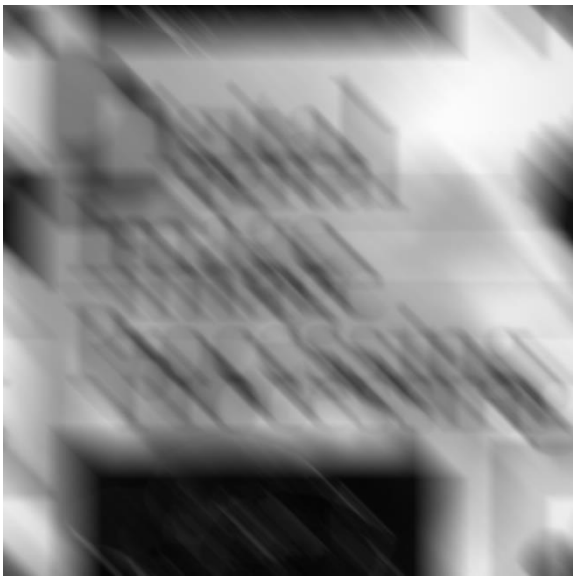
將 F 做 ifft2 得到與上題 inverse filter butterworth_filter (D0 = 85, n = 25)相似的結果，如下圖(甚至有更清楚)。



2. Please use **Inverse Filter and Wiener Filter** to correct the image corrupted by motion blur of the assigned image, 'book-cover-blurred.tif', and print out the source code and the corrected image? (40)

原圖 book-cover-blurred.tif

可觀察到與課本 p.362 EXAMPLE 5.11 範例不同，這裡原圖似乎沒有雜訊只有 motion blur 的結果

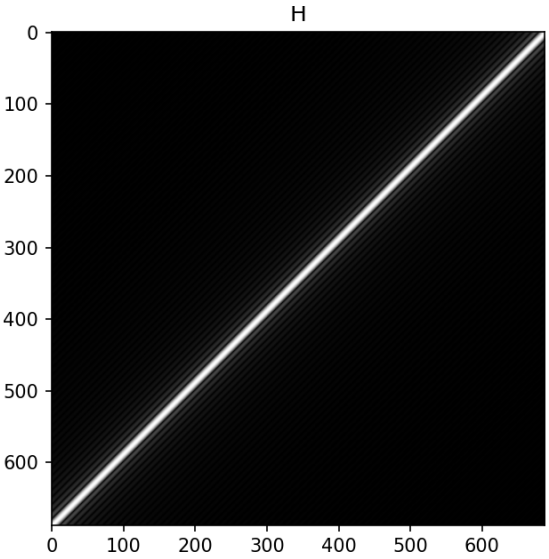


degradation function H 使用課本 p.356 頁 EXAMPLE 5.8 公式

$$H(u,v) = \frac{T}{\pi(ua + vb)}\sin[\pi(ua + vb)]e^{-j\pi(ua + vb)} \qquad (5\text{-}77)$$

需要特別注意在 u = 0 且 v = 0 處前項分母會為 0，因此除下來會式 nan 要另外處理，這裡我的處理方式是將 H 為 NAN 的地方變為 T(如下)

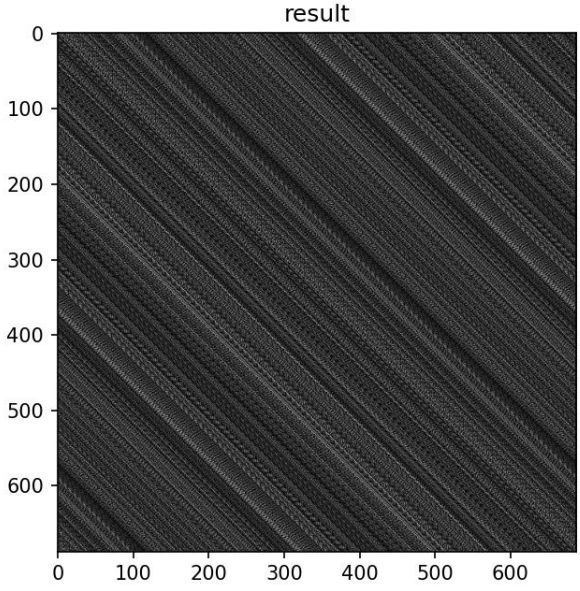
```
M, N, c  = img.shape
a = 0.1
b = 0.1
T = 1

## degradation function H(u, v)
x, y = np.mgrid[-M//2:M//2, -N//2:N//2]
k = np.pi*(x*a+y*b)
H = (T/(k)*np.sin(k))*np.exp(k*-1j)
H[np.isnan(H)] = T
```



1. Inverse Filter

直接做 Inverse Filter，結果如同課本 P.362 FIGURE 5.29(b)，結果不甚理想

```
#for each channel (R,G,B)
for i in range(c):
    g = img[:,:,i]

    ## Fourier Transform of img
    G = np.fft.fftshift(np.fft.fft2(g))

    ## F = G/H
    F = G/H

    ## inverse Fourier Transform of F
    ifft_img = np.fft.ifft2(np.fft.ifftshift(F))
    ifft_img = np.abs(ifft_img)


    ifft_img_n = (ifft_img - ifft_img.min())/(ifft_img.max() - ifft_img.min())*255
    result[:,:,i] = ifft_img_n

plt.imshow(result, cmap='gray'), plt.title("result")
plt.show()
```



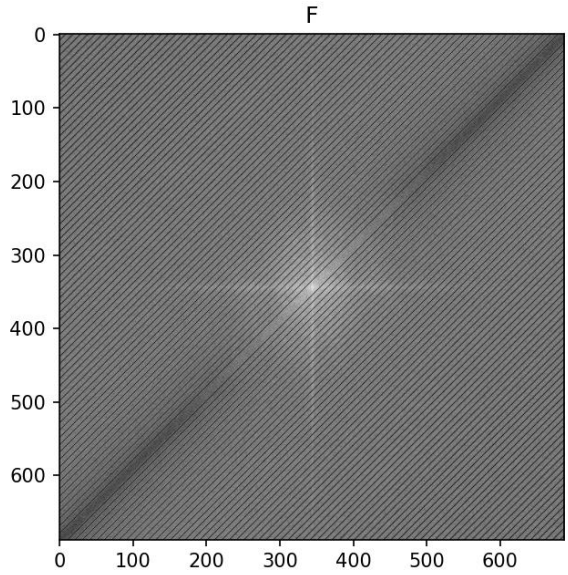
1. Wiener Filter

使用課本 p.360 的公式建 Wiener Filter，K 選用 0.0001 可得到最佳的結果

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2 + K}\right]G(u,v) \qquad (5\text{-}85)$$

寫一個 for 迴圈找最佳的 K 值

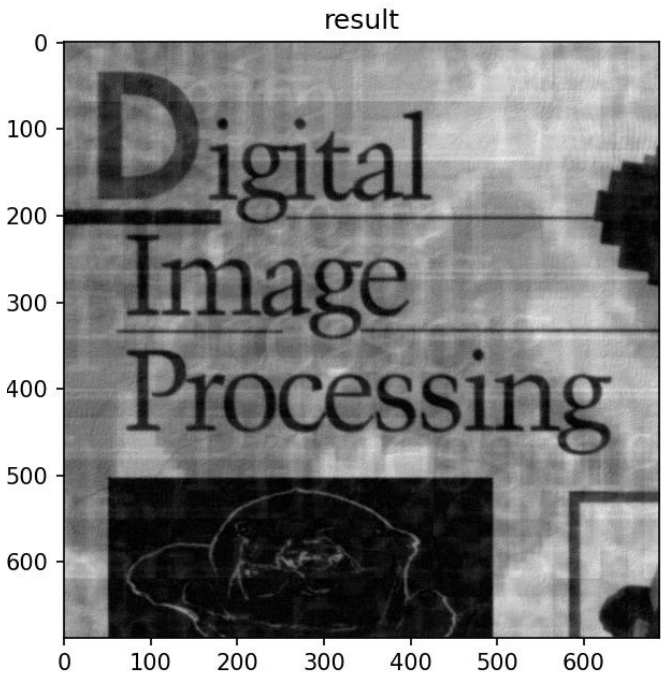
```
## wiener filter
# for p in np.arange(1, 20, 1):
#     print(p)
p = 4  # p=4 is the best
K = 1/10**p
W = (1/H)*(abs(H)**2/(abs(H)**2+K))
F = W*G
plt.imshow(np.log(abs(F) + 1), cmap='gray'), plt.title("F")
plt.show()
```



完整程式碼及結果

(最後 iff_F 有做 normalization 才合併，沒做會有一塊黑黑的)

```
#for each channel (R,G,B)
for i in range(c):
    g = img[:,:,i]

    ## Fourier Transform of img
    G = np.fft.fftshift(np.fft.fft2(g))

    ## wiener filter
    # for p in np.arange(1, 20, 1):
    #     print(p)
    p = 4   # p=4 is the best
    K = 1/10**p
    W = (1/H)*(abs(H)**2/(abs(H)**2+K))
    F = W*G

    ## Inverse Fourier Transform
    ifft_F = np.fft.ifft2(np.fft.ifftshift(F))
    ifft_F = np.abs(ifft_F)

    ifft_F_n = (ifft_F - ifft_F.min())/(ifft_F.max() - ifft_F.min())*255
    result[:,:,i] = ifft_F_n

plt.imshow(result, cmap='gray'), plt.title("result")
plt.show()
```



結果可看到相對 Inverse filter，Wiener Filter 有較好的還原結果。

3. Please comment and compare your two design filters? (20)

兩題做下來的結果，第一題是類似 Gaussian blur 的雜訊，在 image restoration 上，使用 Inverse Filter 和 Wiener Filter，結果並沒有差很多(但選用對的 radius(ideal lowpass filter)或 cutoff frequency(butterworth lowpass filter 相對重要)

第二題是 motion blur，與課本 p.362 EXAMPLE 5.11 範例的圖不同，此題提供的圖用肉眼觀察似乎不帶有雜訊，因此理應用 inverse filter 能夠還原(下方公式少了 N)，但嘗試過套 butterworth filter 仍無法還原原圖，這部分或許可再探討原因。

$$G(u,v) = H(u,v)F(u,v) + N(u,v) \qquad (5\text{-}2)$$

而此題用 MINIMUM MEAN SQUARE ERROR (Wiener Filter)能得到較好的結果(但過程須先找到適合的 K 值)。