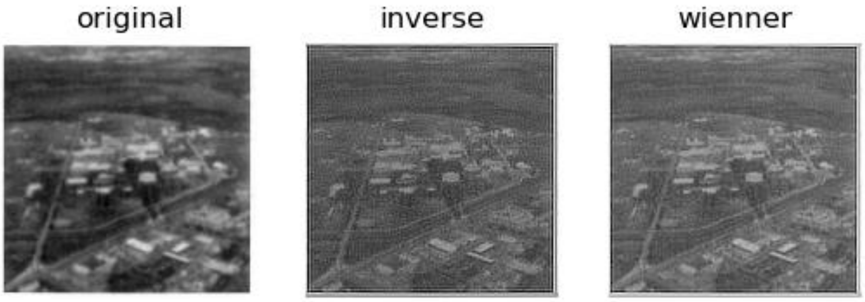
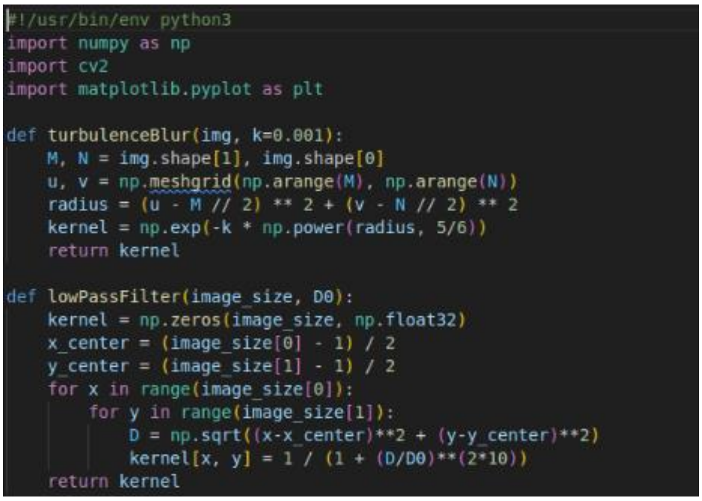
數位影像處理 DIP Homework Chapter 5_1

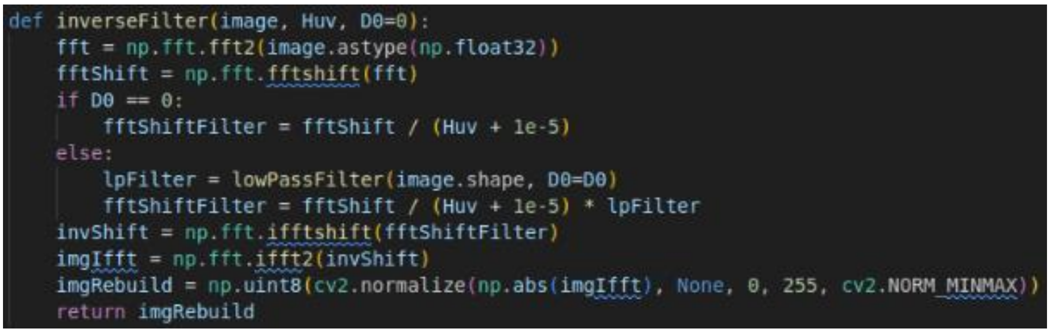
顏志憲 312512049

1. **use Inverse Filter and Wiener Filter to correct the image corrupted severe turbulence of the assigned image**

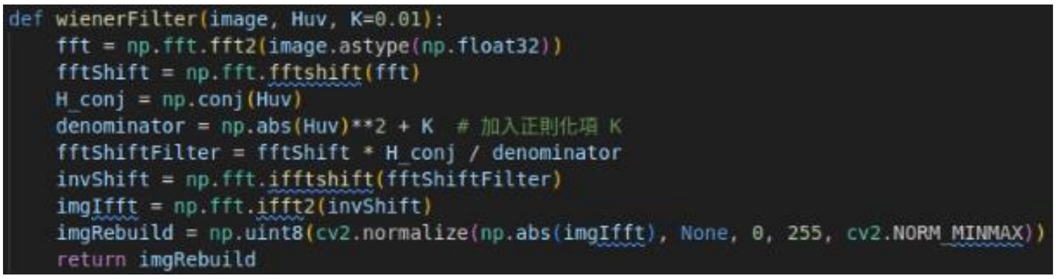




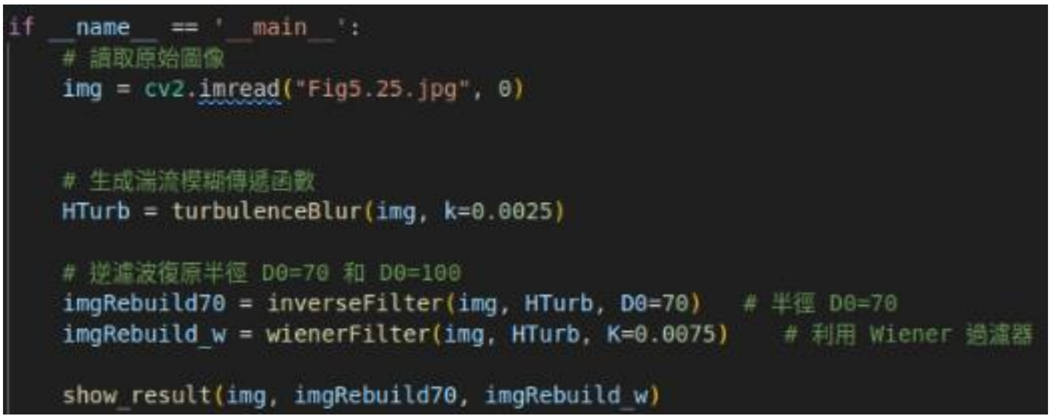
```
#!/usr/bin/env python3
import numpy as np
import cv2
import matplotlib.pyplot as plt

def turbulenceBlur(img, k=0.001):
    M, N = img.shape[1], img.shape[0]
    u, v = np.meshgrid(np.arange(M), np.arange(N))
    radius = (u - M // 2) ** 2 + (v - N // 2) ** 2
    kernel = np.exp(-k * np.power(radius, 5/6))
    return kernel

def lowPassFilter(image_size, D0):
    kernel = np.zeros(image_size, np.float32)
    x_center = (image_size[0] - 1) / 2
    y_center = (image_size[1] - 1) / 2
    for x in range(image_size[0]):
        for y in range(image_size[1]):
            D = np.sqrt((x-x_center)**2 + (y-y_center)**2)
            kernel[x, y] = 1 / (1 + (D/D0)**(2*10))
    return kernel
```



```
def inverseFilter(image, Huv, D0=0):
    fft = np.fft.fft2(image.astype(np.float32))
    fftShift = np.fft.fftshift(fft)
    if D0 == 0:
        fftShiftFilter = fftShift / (Huv + 1e-5)
    else:
        lpFilter = lowPassFilter(image.shape, D0=D0)
        fftShiftFilter = fftShift / (Huv + 1e-5) * lpFilter
    invShift = np.fft.ifftshift(fftShiftFilter)
    imgIfft = np.fft.ifft2(invShift)
    imgRebuild = np.uint8(cv2.normalize(np.abs(imgIfft), None, 0, 255, cv2.NORM_MINMAX))
    return imgRebuild
```

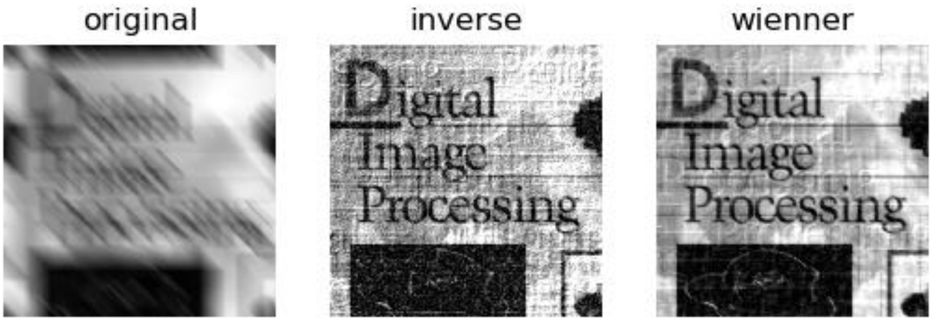


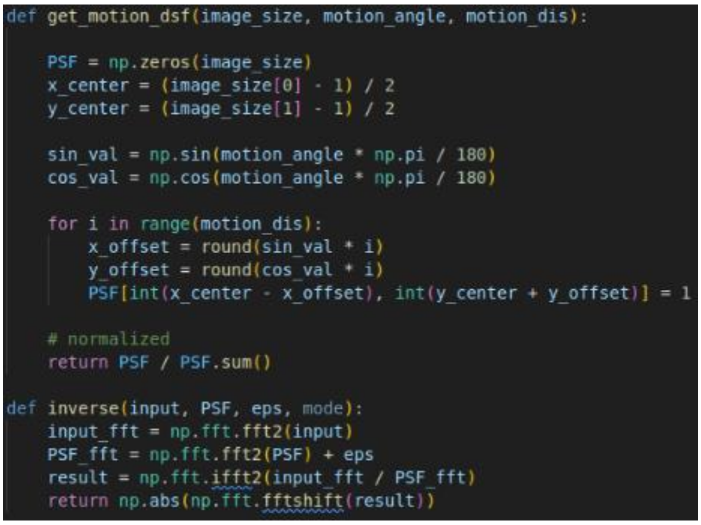
```
def wienerFilter(image, Huv, K=0.01):
    fft = np.fft.fft2(image.astype(np.float32))
    fftShift = np.fft.fftshift(fft)
    H_conj = np.conj(Huv)
    denominator = np.abs(Huv)**2 + K  # 加入正則化項 K
    fftShiftFilter = fftShift * H_conj / denominator
    invShift = np.fft.ifftshift(fftShiftFilter)
    imgIfft = np.fft.ifft2(invShift)
    imgRebuild = np.uint8(cv2.normalize(np.abs(imgIfft), None, 0, 255, cv2.NORM_MINMAX))
    return imgRebuild
```



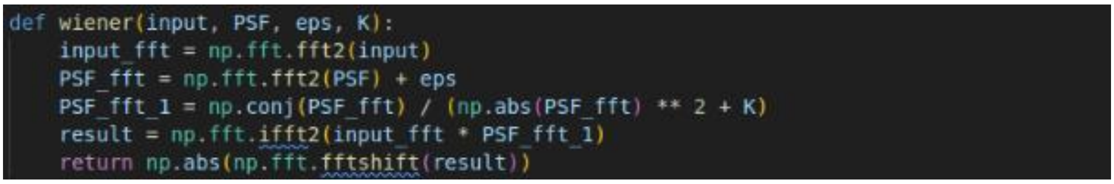
```
if __name__ == '__main__':
    # 讀取原始圖像
    img = cv2.imread("Fig5.25.jpg", 0)


    # 生成湍流模糊傳遞函數
    HTurb = turbulenceBlur(img, k=0.0025)

    # 逆濾波復原半徑 D0=70 和 D0=100
    imgRebuild70 = inverseFilter(img, HTurb, D0=70)   # 半徑 D0=70
    imgRebuild_w = wienerFilter(img, HTurb, K=0.0075)    # 利用 Wiener 過濾器

    show_result(img, imgRebuild70, imgRebuild_w)
```

2. use Inverse Filter and Wiener Filter to correct the image corrupted by motion blur of the assigned image

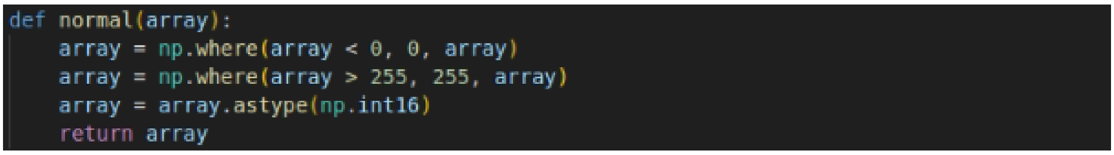




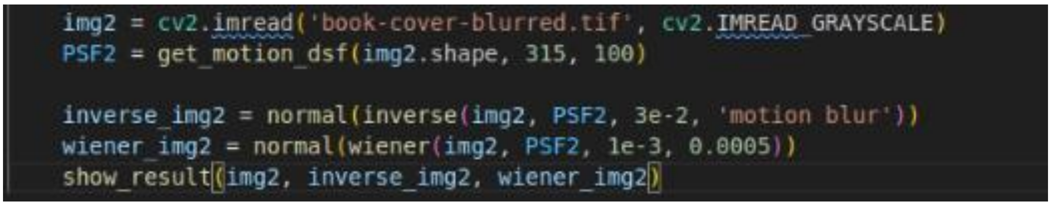
```
def get_motion_dsf(image_size, motion_angle, motion_dis):

    PSF = np.zeros(image_size)
    x_center = (image_size[0] - 1) / 2
    y_center = (image_size[1] - 1) / 2

    sin_val = np.sin(motion_angle * np.pi / 180)
    cos_val = np.cos(motion_angle * np.pi / 180)

    for i in range(motion_dis):
        x_offset = round(sin_val * i)
        y_offset = round(cos_val * i)
        PSF[int(x_center - x_offset), int(y_center + y_offset)] = 1

    # normalized
    return PSF / PSF.sum()

def inverse(input, PSF, eps, mode):
    input_fft = np.fft.fft2(input)
    PSF_fft = np.fft.fft2(PSF) + eps
    result = np.fft.ifft2(input_fft / PSF_fft)
    return np.abs(np.fft.fftshift(result))
```



```
def wiener(input, PSF, eps, K):
    input_fft = np.fft.fft2(input)
    PSF_fft = np.fft.fft2(PSF) + eps
    PSF_fft_1 = np.conj(PSF_fft) / (np.abs(PSF_fft) ** 2 + K)
    result = np.fft.ifft2(input_fft * PSF_fft_1)
    return np.abs(np.fft.fftshift(result))
```



```
def normal(array):
    array = np.where(array < 0, 0, array)
    array = np.where(array > 255, 255, array)
    array = array.astype(np.int16)
    return array
```



```
    img2 = cv2.imread('book-cover-blurred.tif', cv2.IMREAD_GRAYSCALE)
    PSF2 = get_motion_dsf(img2.shape, 315, 100)

    inverse_img2 = normal(inverse(img2, PSF2, 3e-2, 'motion blur'))
    wiener_img2 = normal(wiener(img2, PSF2, 1e-3, 0.0005))
    show_result(img2, inverse_img2, wiener_img2)
```

3. Please comment and compare your two design filters?

由結果也可以發現，理想上雖然退化函數，能夠用 inversfilter 復原，但由於其受雜訊影響較大，當退化函數為零或極小值時，會放大雜訊影響，而使用低通濾波能夠有效限制原點附近的頻域，能稍微改善雜訊帶來的影響，而 wiener filter 使影像與原影像間的均方誤差最小，因此有更好的效果且較不易受雜訊影響。