Project 3

ID：311512064 name：鄧書桓

(a) Source codes:

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt
from openpyxl import load_workbook
from PIL import Image
from random import randint
import math
import random
img_path = "./Kid2 degraded.tiff"
# 0 is open this img as gray img
img_ori = cv2.imread(img_path, 0)


def filter_process(img, alpha):
    vector1 = []
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            vector1.append(img[i, j])
    vector1.sort()
    sum = 0
    for i in range(math.floor(alpha/2.), len(vector1) - math.floor(alpha/2.)):
        sum += vector1[i]
    sum /= len(vector1) - 2 * math.floor(alpha/2.)
    return int(sum)


if __name__ == "__main__":
    kernel = 5
    alpha = 16

    # padding
    side_leng = math.floor(kernel/2.)
    height, width = img_ori.shape
    padded_img = np.zeros((height+2*side_leng, width+2*side_leng))
```

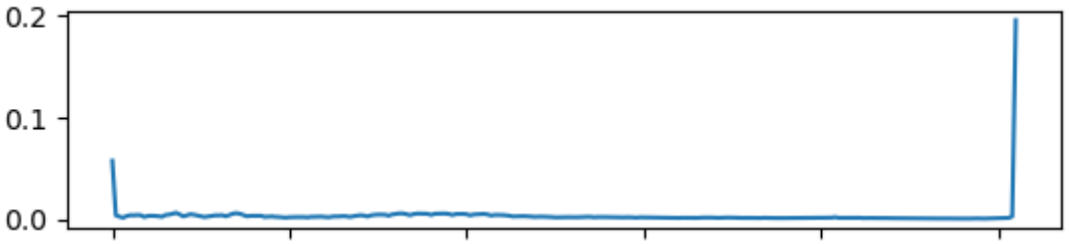
```
    denoise_img = np.zeros((height, width))
    padded_img[2:height+2, 2:width+2] = img_ori
    side_leng = math.floor(kernel/2.)
    # denoising
    for i in range(side_leng, height+side_leng):
        for j in range(side_leng, width+side_leng):
            tmp_img = padded_img[i - side_leng:i +
                                 side_leng + 1, j - side_leng:j + side_leng + 1]
            denoise_img[i-2, j-2] = filter_process(tmp_img, alpha)

    # save denoise_img
    PILimage = Image.fromarray(denoise_img.astype(np.uint8))
    PILimage.save("img/(c)denoise_img.png", dpi=(200, 200))

    # noise model and model parameters
    # show the result and calculate the noise model parameter
    original_hist = cv2.calcHist([img_ori.astype(np.uint8)], [
                                 0], None, [256], [0, 256])/(800**2)
    denoise_hist = cv2.calcHist([denoise_img.astype(np.uint8)], [
                                0], None, [256], [0, 256])/(800**2)
    probdiff_hist = original_hist - denoise_hist

    plt.subplot(311), plt.plot(original_hist)
    plt.subplot(312), plt.plot(denoise_hist)
    plt.subplot(313), plt.plot(probdiff_hist)
    plt.show()
    Pa = probdiff_hist[0]
    Pb = probdiff_hist[255]
    print("Pa = ", Pa)
    print("Pb = ", Pb)
```

```
# prepare for frequency application
img_padding = cv2.copyMakeBorder(
    denoise_img, 0, 800, 0, 800, cv2.BORDER_CONSTANT)
g = np.fft.fft2(img_padding)
G = np.fft.fftshift(g)

M = img_padding.shape[0]
N = img_padding.shape[1]

# lowpass filter
# cutoff frequency
D0_butter = 250
D0_guassian = [100, 150, 200, 250]
# order
n = 10
B = 0.414
BLPF = np.zeros((M, N), dtype=np.float32)
GLPF = np.zeros((M, N), dtype=np.float32)
for i in range(len(D0_guassian)):
    for u in range(M):
        for v in range(N):
            D = np.sqrt((u-M/2)**2 + (v-N/2)**2)
            # Butterworth lowpass filter
            BLPF[u, v] = 1 / (1 + B*(D/D0_butter)**(2*n))
            # Gusian lowpass filter
            GLPF[u, v] = np.exp(-(D**2) / (2 * (D0_guassian[i]**2)))
    F_shift = G*BLPF/GLPF
    F = np.fft.ifftshift(F_shift)
    f = np.abs(np.fft.ifft2(F))
    f = f[0:800, 0:800]
    z = 221+i
    plt.subplot(z)
```

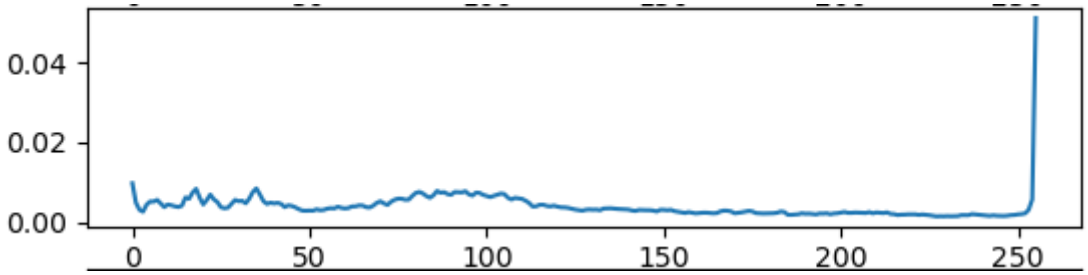
```
        plt.imshow(f, cmap='gray')
        # save reconstructed img
        PILimage = Image.fromarray(f.astype(np.uint8))
        PILimage.save("img/(d)reconstructed_D0_" +
                      str(D0_guassian[i])+".png", dpi=(200, 200))
    plt.show()
```

(b) Results of noise model and model parameters:

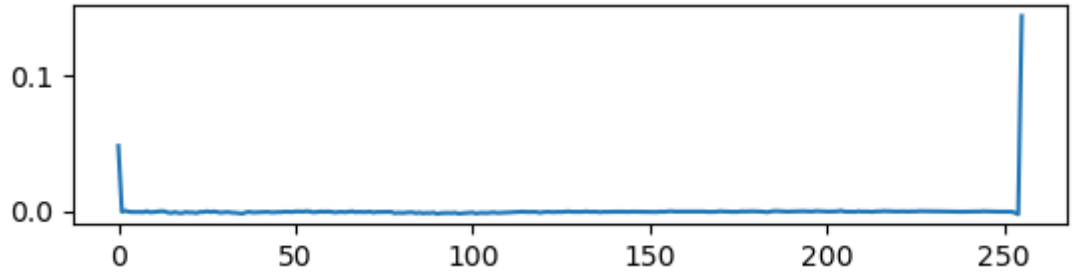
Original picture:



Denoise model:



Noise model:



使用原圖的 gray-level 與 de-noise 後的 gray-level 相減而得出 noise-model

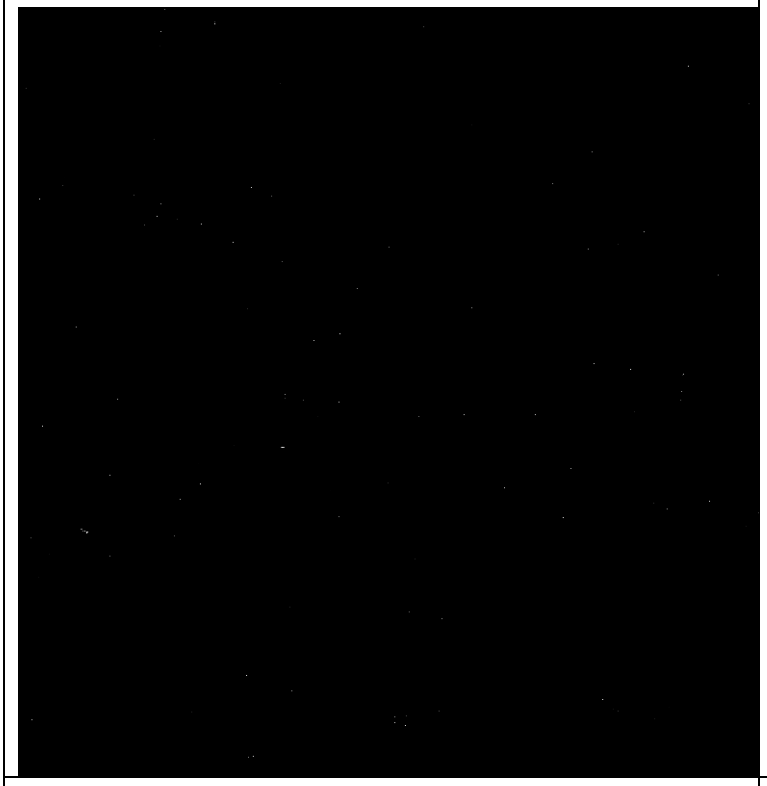
Parament:

```
Pa =  [0.04806094]
Pb =  [0.14421094]
```

(c) De-noised image by alpha-trimmed mean filter:

(d) Output image、parameters:

D0 = 100



D0 = 150

D0 = 200	D0 = 250
	

Butterworth 中的 D0 為限制 inverse filter 使用的範圍，若太小會無法還原，因此這邊使用 D0 = 250。另外，n 越大越接近理想 LPF，反之則越接近 GLPF， 實際測試下來 n 越小越模糊，因此這邊使用 n = 10。

由上面 4 張圖可知，當 Guassian 的 D0 為 100 時無法顯示出來，我認為是因為 Butterworth 設的範圍小於或剛好等於 Guassian deconvolution 的範圍，因此什麼都看不到。另外，當 Guassian 的 D0 越大時，照片中的黑色線條(deconvolution 而來)會跟著減少，可能是 deconvolution 的範圍減少，因使類似銳化的區域也變小。