Project 3

ID： 311512040　　name：林胤宏

(a) Source codes:

```
import cv2
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

```
img = cv2.imread("./Kid2 degraded.tiff", 0)
plt.imshow(img,cmap='gray')
plt.show()
print(img.shape)
```

```
crop_img = img[0:100,0:100]   #裁切左上角 100*100 來檢視 noise
plt.imshow(crop_img,cmap='gray')
plt.show()
print(crop_img.shape)
```

```
hist = cv2.calcHist([crop_img], [0], None, [256], [0,256])
#cv2.calcHist(影像, 通道, 遮罩, 區間數量, 數值範圍)
print(hist.shape)

plt.title("Histogram")
plt.xlabel("Brightness")
plt.ylabel("pixel numbers")
plt.plot(hist)
plt.show()

print(hist[0][0])
print(hist[255][0])
```

```
def AlphaTrimmedMean(data, filter_size):
    temp = []
    indexer = filter_size // 2    #2
    data_final = []
    data_final = np.zeros((len(data),len(data[0])))
    for i in range(len(data)):

        for j in range(len(data[0])):

            for z in range(filter_size):
                if i + z - indexer < 0 or i + z - indexer > len(data) - 1:
                    for c in range(filter_size):
                        temp.append(0)
                else:
```

```
                if j + z - indexer < 0 or j + indexer > len(data[0]) - 1:
                    temp.append(0)
                else:
                    for k in range(filter_size):
                        temp.append(data[i + z - indexer][j + k - indexer])

        temp.sort()
        data_final[i][j] = sum(temp[8:17])/9
        temp = []
    return data_final
```

```
delnoise_img=AlphaTrimmedMean(img,5)
plt.imshow(delnoise_img,cmap='gray')
plt.show()
```

```
delnoise_img = cv2.copyMakeBorder(delnoise_img, 0, 800, 0, 800, cv2.BORDER_CONSTANT)
#padding
kid_DFT = np.fft.fft2(delnoise_img)
kid_DFT = np.fft.fftshift(kid_DFT)
M, N = kid_DFT.shape
print(kid_DFT.shape)
```

```
## BLPF AND GLPF
BLPF = np.zeros((M, N), dtype=np.float32)
GLPF = np.zeros((M, N), dtype=np.float32)
D01 = 200 # Butter worth LPF cutoff frequency (fixed)
D02 = [100, 150, 200, 250] # Gassian LPF cutoff frequency (changed)
B = 0.414
n = 5 # order
for D0 in D02:
    print(D0)
    for u in range(M):
        for v in range(N):
            D = np.sqrt((u-M/2)**2 + (v-N/2)**2)
            BLPF[u,v] = 1 / (1 + B*((D/D01)**(2*n))) # Butter worth LPF
            GLPF[u,v] = np.exp(-D**2/(2*D0*D0)) # Gassian LPF
    Gshift = BLPF* (kid_DFT / GLPF)
    kid_invshift = np.fft.ifftshift(Gshift)
    kid_invDFT = np.abs(np.fft.ifft2(kid_invshift))
    img = Image.fromarray(kid_invDFT[0:800, 0:800])
    img = img.convert("L")
    img.save('output/'+str(D0)+'RESULT.jpg', dpi=(200, 200))
```

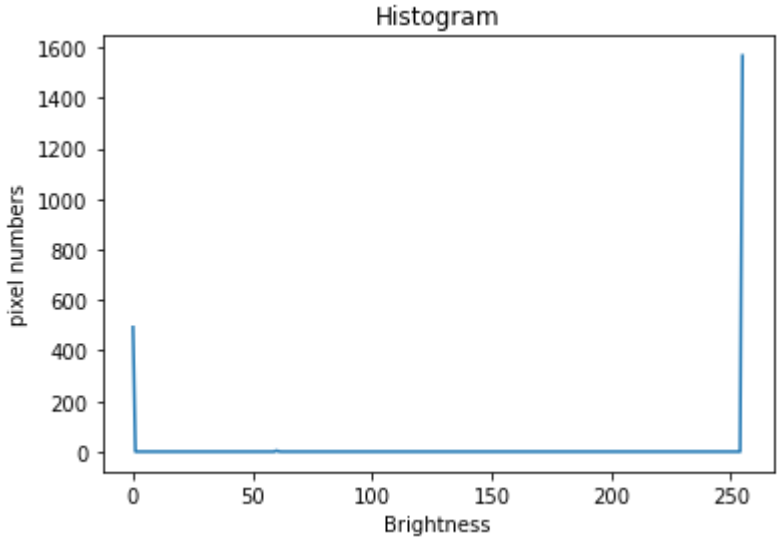
(b) Results of noise model and model parameters:

裁切左上角 100x100 的區域:

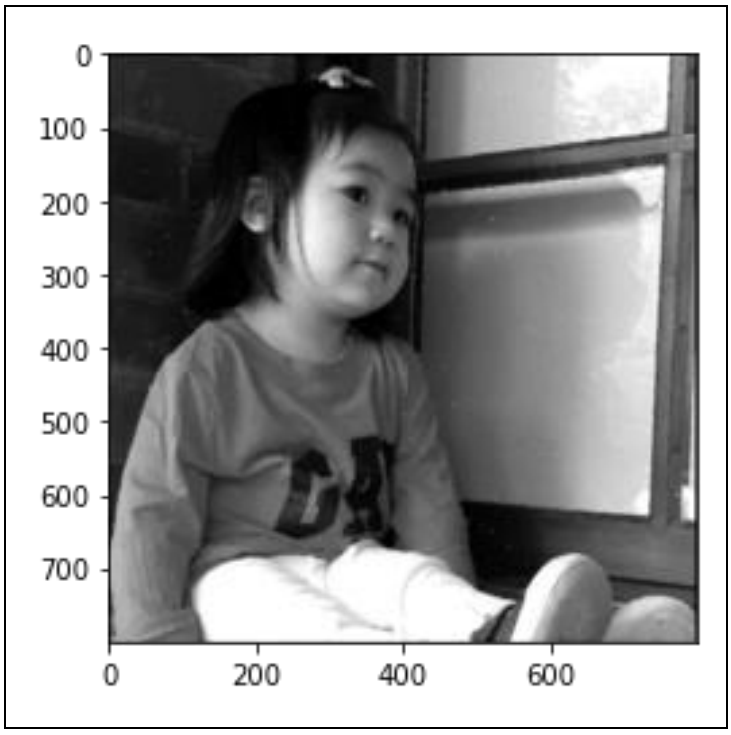
Pa 為黑點佔該區域的 probability

Pb 為白點佔該區域的 probability

```
Pa=  0.0493
Pb=  0.1568
```



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



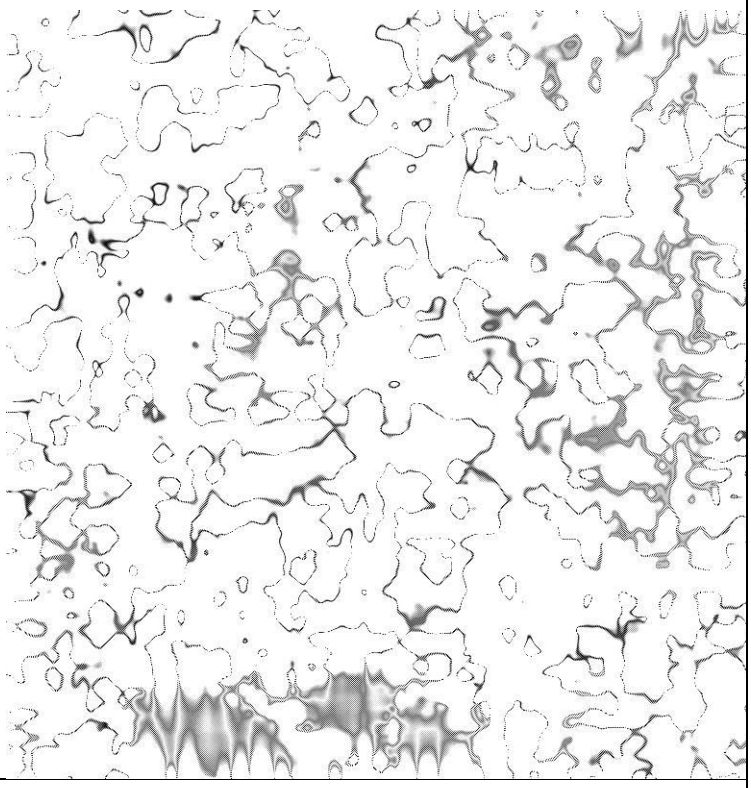
(d) Output image、parameters:

固定 Butter worth LPF (n = 5, and D0 = 200)

使用 Gaussian LPF 做 inverse filter，分別使用 D0 = 100, 150, 200, 250 來做，從上圖的結果可以得知當 cutoff frequency 越小的時候，所能得到的頻率範圍越小，所以只能得出一張白色的圖案，將 D0 條大之後，所能得到的頻率範圍變大，所得到的圖片也會越清晰。

D0 = 100

D0 = 150



D0 = 200

D0 = 250