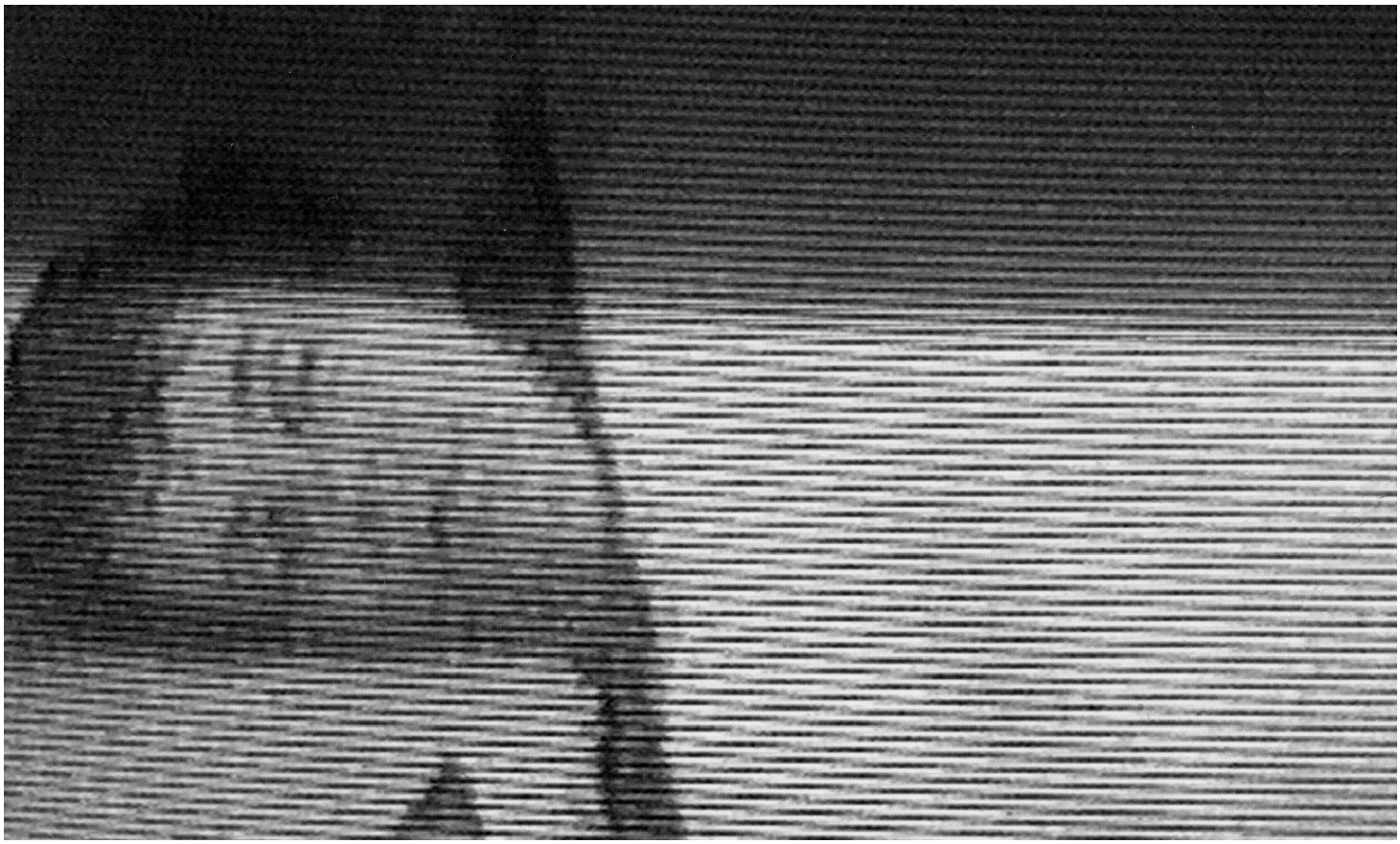
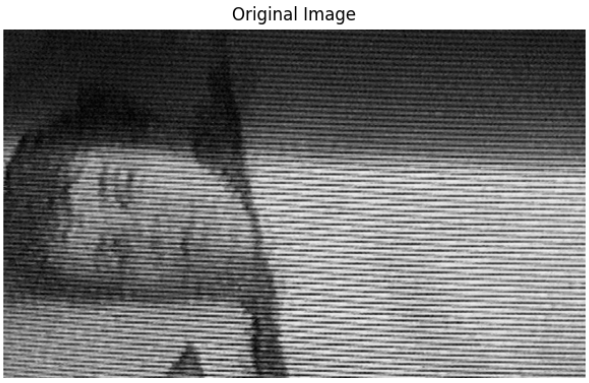
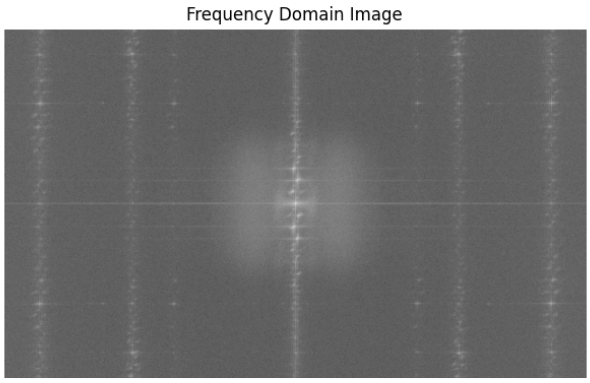
功課要求

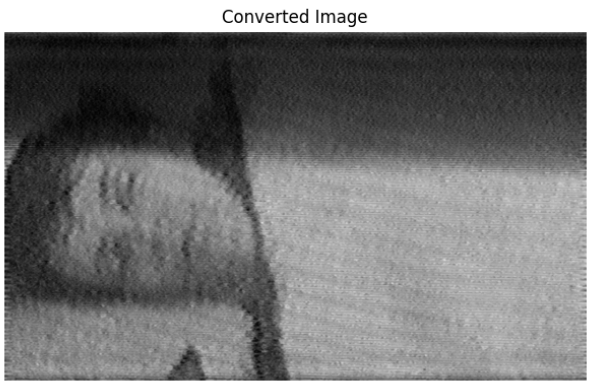
圖片似乎受到某種頻域雜訊干擾，撰寫一個程式嘗試復原此圖像。



成果







開發環境

OS	Editor	Language	OpenCV
Windows 10	Visual Studio Code	Python 3.9.16	OpenCV 4.5.4

實作

> 本次程式碼

使用的 libraries 如下：

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

1/ 使用 DFT 取得頻域圖像與頻譜圖

這裡和作業3的步驟都一樣。

2/ 用滑鼠點選 notch points

notch point 通常用來消除圖像上的特定頻率，這裡我們透過觀察，可以發現頻譜圖上有六個週期性出現的亮點，所以來進行手動選取。

建立一個 function 將點擊處的值設為 0:

```
def add_notch_point(event, x, y, flags, img):
    # if button is clicked, mark the point
    if event == cv2.EVENT_LBUTTONDOWN:
        print("added nothch point at: ", x, y)
        # draw a circle
        cv2.circle(img, (x, y), 20, 0, -1)
```

使用 `cv2.setMouseCallback()` 建立一個可供點擊的視窗：

```
# mouse click to find notch points
notch_points_img = np.ones(magnitude.shape, dtype=np.uint8)
cv2.namedWindow('Frequency Domain Image')
cv2.setMouseCallback('Frequency Domain Image', add_notch_point, notch_points_img)
cv2.imshow('Frequency Domain Image', magnitude)
cv2.waitKey(0)
```

在這裡就能知道當時有平移 DFT 圖像的好處了，可以很快的發現哪些亮點是週期性的重複出現。

3/ 平移得到的 notch points 圖像

剛才我們都是看著平移過的頻譜圖進行點擊，但原先的 DFT 結果並非這樣，所以我們要反向的把 notch points 圖像給平移回來。

```
# swap notch points to match dft_A
tmp = np.copy(notch_points_img[0:cy, 0:cx])
notch_points_img[0:cy, 0:cx] = notch_points_img[cy:dft_A.shape[0], cx:dft_A.shape[1]]
notch_points_img[cy:dft_A.shape[0], cx:dft_A.shape[1]] = tmp
tmp = np.copy(notch_points_img[0:cy, cx:dft_A.shape[1]])
notch_points_img[0:cy, cx:dft_A.shape[1]] = notch_points_img[cy:dft_A.shape[0], 0:cx]
notch_points_img[cy:dft_A.shape[0], 0:cx] = tmp
```

4/ 套用 notch filter

將原本的 DFT 圖像的兩個通道和平移過的 notch filter 相乘，消除滑鼠點過的黑點部分，週期性雜訊就會被去除掉。

```
# apply notch filter
planes[0] = planes[0] * notch_points_img
planes[1] = planes[1] * notch_points_img
dftB = cv2.merge(planes)
```

5/ 利用反向 DFT 還原圖片

最後使用 `cv2.idft()` 還原圖像，由於前面已經把刪除好雜訊的 dftB 做出來了，所以這邊只要還原就能得到去除雜訊後的圖像

```
# inverse dft_B
cv2.idft(dftB, dftB)
cv2.split(dftB, planes)
# get magnitude
inverse_img = cv2.magnitude(planes[0], planes[1])
# normalize to 0~255
cv2.normalize(inverse_img, inverse_img, 0, 255, cv2.NORM_MINMAX)
# convert to 8 bit unsigned integer
inverse_img = inverse_img.astype(np.uint8)
```

總結

將影像轉換至頻域後，透過觀察可以消除一些週期性出現的亮點，從而消除週期性雜訊。

參考資料

- Periodic Noise Removing Filter