1091402 顏心妤

作業 1

程式功能：

執行後開啟兩個含有滑動條與圖片的視窗，一個能旋轉整張圖片，另一個旋轉中心內切圓區域，並透過滑動條控制旋轉角度（逆時針旋轉 0 度至 359 度）

開發環境：

Windows10、Spyder (Python 3.8)、OpenCV 4.7.0

程式碼說明：

```
4    img = cv2.imread('yzu.bmp')
5    cv2.imshow('Rotate image', img)
6    cv2.imshow('Rotate CR image', img)
7
8    (h, w, d) = img.shape
9    center = (w//2, h//2)
10   radius = min(w//2, h//2)
```

cv2.imread() 讀取圖片
cv2.imshow() 顯示圖片
(h, w, d) = img.shape 獲取圖片長寬高
center = (w//2, h//2) 計算中心座標
radius = min(w//2, h//2) 得到最小半徑

cv2.createTrackbar() 分別在兩視窗建立滑動條

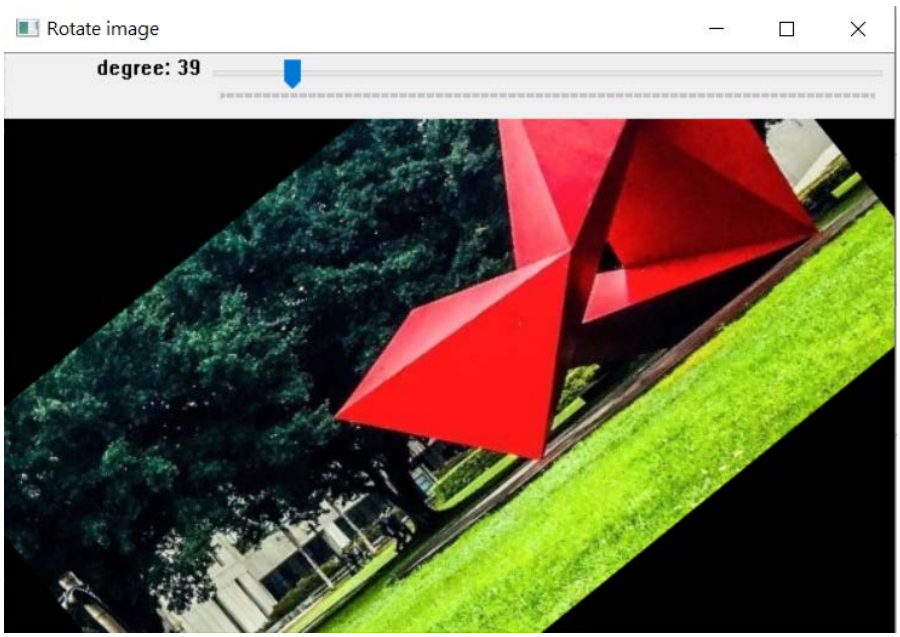
```
31   cv2.createTrackbar('degree', 'Rotate image', 0, 359, rotate)
32   cv2.createTrackbar('degree', 'Rotate CR image', 0, 359, rotateCR)
```

分別寫一個 function，讓兩張圖能隨著滑動條呈現旋轉後的圖像。
(a) def rotate(angle):
(b) def rotateCR(angle):

cv2.waitKey(0) 等待按鍵輸入
cv2.destroyAllWindows() 關閉視窗

(a)

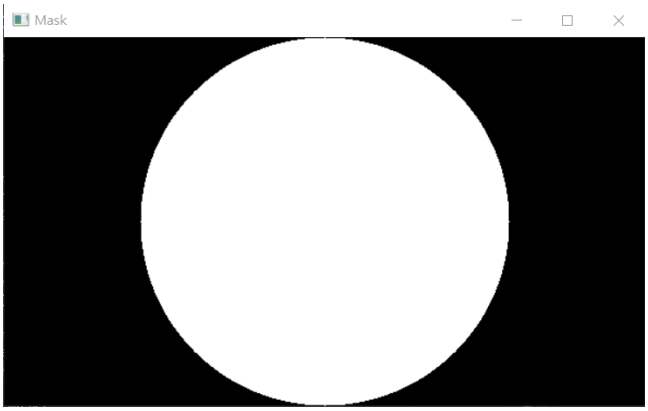
cv2.getRotationMatrix2D() 創建二維旋轉矩陣
cv2.warpAffine() 將指定圖像根據旋轉矩陣進行旋轉
cv2.imshow() 顯示圖片



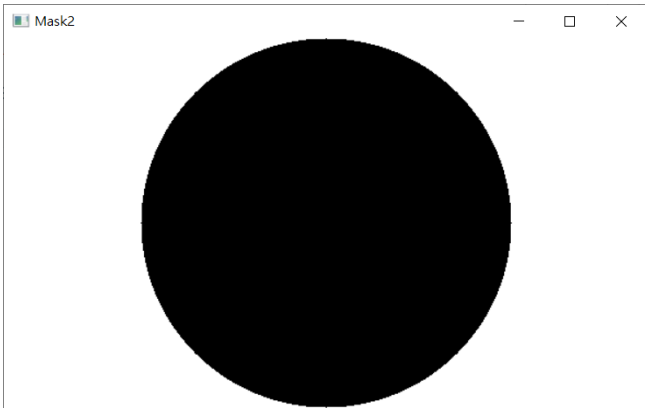
(b)

np.zeros_like() 先建立與圖形一樣大小的遮罩

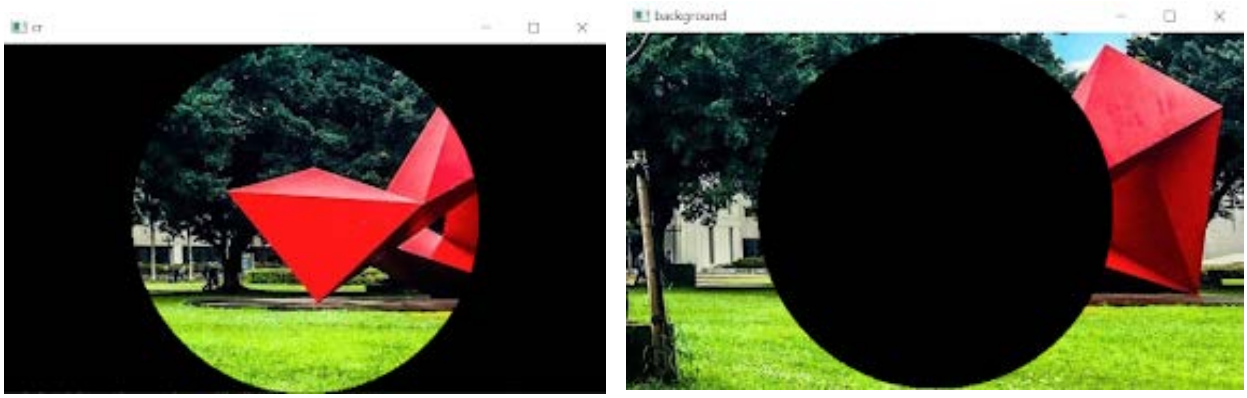
cv2.circle() 然後在遮罩上畫一個圓形



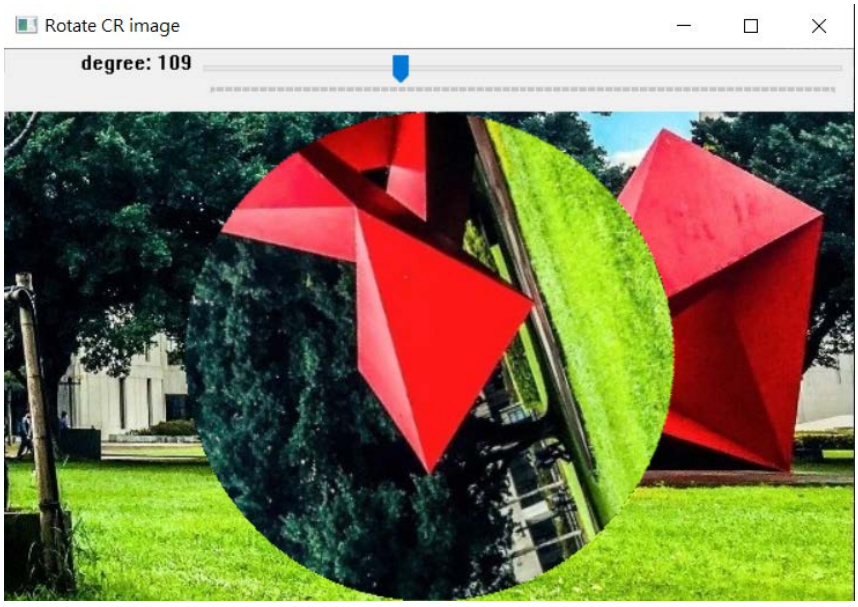
cv2.bitwise_not() 建立一個與原本反過來的遮罩，並建立白色背景



將原圖分別在兩遮罩上進行 cv2.bitwise_and()以得到所需部分的圖(cr、background)



同(a)，對圖像進行矩陣旋轉
最後用 cv2.add()將兩圖(cr、background)合併
cv2.imshow() 顯示圖片



Demo 影片網址：https://youtu.be/CrVKOu2hDRc

code：

```
import cv2
import numpy as np
```

```
img = cv2.imread('yzu.bmp')
cv2.imshow('Rotate image', img)
cv2.imshow('Rotate CR image', img)

(h, w, d) = img.shape
center = (w//2, h//2)
radius = min(w//2, h//2)

def rotate(angle):
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(img, M, (w, h))
    cv2.imshow('Rotate image', rotated) #update img

def rotateCR(angle):
    mask = np.zeros_like(img)
    cv2.circle(mask, center, radius, (255, 255, 255), -1)
    #cv2.imshow('Mask',mask)

    mask2=cv2.bitwise_not(mask)
    cv2.circle(mask2, center, radius, (0, 0, 0), -1)
    #cv2.imshow('Mask2',mask2)

    cr=cv2.bitwise_and(img,mask)
    #cv2.imshow('cr',cr)
    background=cv2.bitwise_and(img,mask2)
    #cv2.imshow('background',background)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv2.warpAffine(cr, M, (w, h))
    result=cv2.add(background,rotated)
    cv2.imshow('Rotate CR image',result)

cv2.createTrackbar('degree', 'Rotate image', 0, 359, rotate)
cv2.createTrackbar('degree', 'Rotate CR image', 0, 359, rotateCR)



cv2.waitKey(0)
cv2.destroyAllWindows()
```

作業 2

程式功能：

以灰階模式讀取一張圖像 imread(path, IMREAD_GRAYSCALE)
(a)利用 Sobel Operators 偵測並輸出邊緣成分圖
(b)設計一個類似素描線條的自畫像圖案。

開發環境：

Windows10、Spyder (Python 3.8)、OpenCV 4.7.0

程式碼說

cv2.imread() 讀取圖片

(原圖)

(a)
cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) 將圖片轉成灰階

用 cv2.Sobel(影像,輸出圖像的數據類型,計算方向 x 軸,計算方向 y 軸) 分別對 x、y 軸進行邊緣檢測

cv2.convertScaleAbs() 將計算完的 x、y 軸的值轉回原本的 unit8 格式

cv2.addWeighted() 將計算完的 x、y 合併

(b)

cv2.bitwise_not() 將黑白反轉

Code：

```
import cv2

img = cv2.imread('fig.jpg')

IMREAD_GRAYSCALE=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) #轉灰階
cv2.imshow('gray',IMREAD_GRAYSCALE)

x = cv2.Sobel(IMREAD_GRAYSCALE, cv2.CV_64F, 1, 0)
y = cv2.Sobel(IMREAD_GRAYSCALE, cv2.CV_64F, 0, 1)

absX = cv2.convertScaleAbs(x)
absY = cv2.convertScaleAbs(y)

dst = cv2.addWeighted(absX, 0.5, absX,0.5,0)
cv2.imshow('dst',dst)

sketch=cv2.bitwise_not(dst)
cv2.imshow('sketch',sketch)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

作業 3

題目敘述

離散傅立葉轉換 DFT 練習

撰寫傅利葉轉換程式(Forward Fourier Transform and Inverse Fourier Transform)將一張圖
像轉換至頻域後，將頻譜大小與相位角度各以灰階 256 色圖像方式呈現出，再呈現還
原後圖像。

開發環境

Windows10、Spyder (Python 3.8)、OpenCV 4.7.0

說明

1、讀取圖像並轉為灰階

2、用 getOptimalDFTSize 得到最適合進行傅里葉變換的圖像大小，然後用 copyMakeBorder 進行 padding，並轉成 float32 來計算

2、fp(x,y) × (−1)^(x+y) to center the Fourier transform

```
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
            padded_img[i][j] *= (-1)**(i+j)
```

3、dft()計算傅立葉轉換，然後用 split()將虛數與實數分開

magnitude()計算幅值，並取 log 乘上 20 得到振幅
normalize() 標準化，讓圖像以灰階 256 色圖像方式呈現，並轉成 uint8 型態(避免負數)

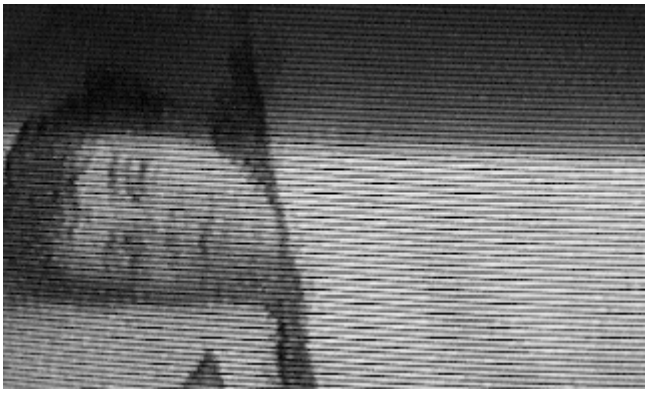
```
S = 20 * np.log(cv2.magnitude(planes[0], planes[1]))
# normalize the output image to [0, 255]
S = cv2.normalize(S, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
```

phase() 計算相位
並同上進行標準化。

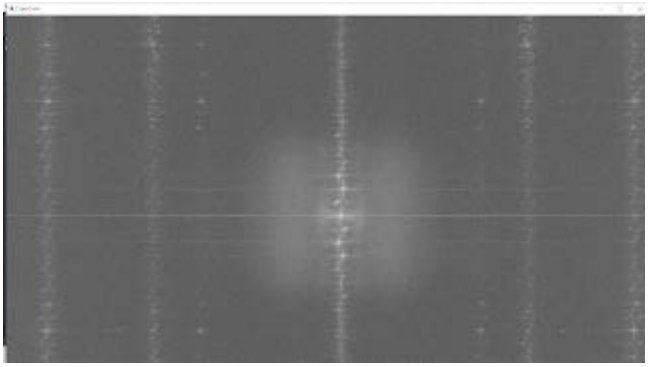
```
phase = cv2.phase(planes[0], planes[1], angleInDegrees=True)
phase = cv2.normalize(phase, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
```

idft() 計算還原後圖像
magnitude()計算幅值
並同上進行標準化。

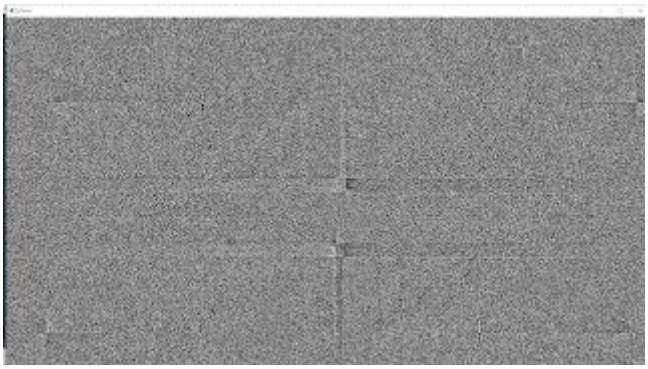
```
idft = cv2.idft(DFT)
idft = cv2.magnitude(idft[:, :, 0], idft[:, :, 1])
idft = cv2.normalize(idft, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
```



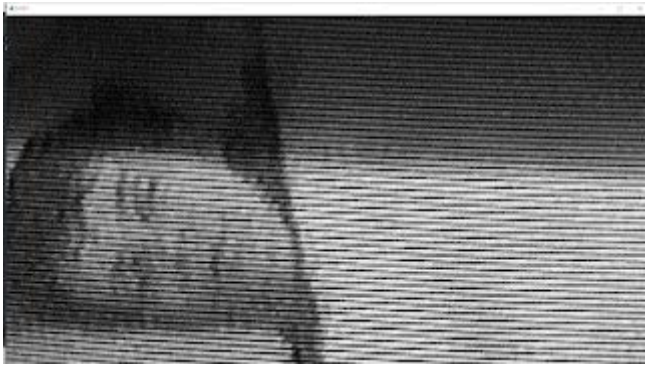
Original



Spectrum



Phase



IDFT

Code：

```
import cv2
import numpy as np

img = cv2.imread('image3.png', cv2.IMREAD_GRAYSCALE)

P = cv2.getOptimalDFTSize(img.shape[0])
Q = cv2.getOptimalDFTSize(img.shape[1])

# padding
padded_img = cv2.copyMakeBorder(img, 0, P - img.shape[0], 0, Q - img.shape[1],
cv2.BORDER_CONSTANT, value=0)
padded_img = padded_img.astype(np.float32)

# center image
for i in range(img.shape[0]):
        for j in range(img.shape[1]):
                padded_img[i][j] *= (-1)**(i+j)

# compute DFT
DFT = cv2.dft(padded_img, flags=cv2.DFT_COMPLEX_OUTPUT)
planes = cv2.split(DFT)

# compute spectrum
S = 20 * np.log(cv2.magnitude(planes[0], planes[1]))
# normalize the output image to [0, 255]
S = cv2.normalize(S, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)

# compute phase spectrum
phase = cv2.phase(planes[0], planes[1], angleInDegrees=True)
phase = cv2.normalize(phase, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)

# compute inverse DFT
idft = cv2.idft(DFT)
idft = cv2.magnitude(idft[:, :, 0], idft[:, :, 1])
idft = cv2.normalize(idft, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)

cv2.imshow('Spectrum', S)
cv2.imshow('phase', phase)
```

```
cv2.imshow('IDFT', idft)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

作業 4

題目敘述：

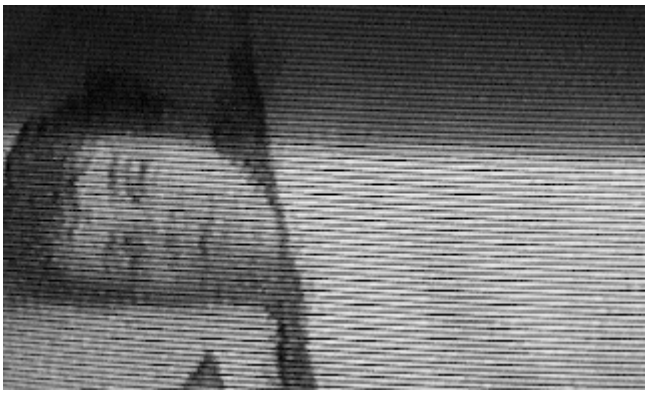
影像還原練習

附件中的 image4 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(將圖中雜訊去除)。

開發環境：

Windows10、Spyder (Python 3.8)、OpenCV 4.7.0

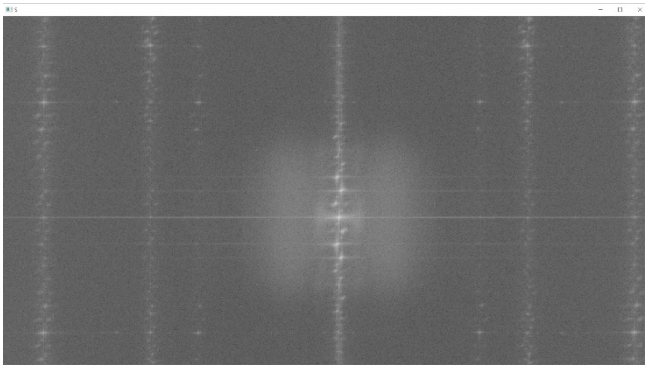
說明：



(原圖)

前半部分(至 27 行)為作業 3 的內容

延用作業 3 得到的頻譜圖，可以看到六個明顯的白點，即雜訊(峰值較高)



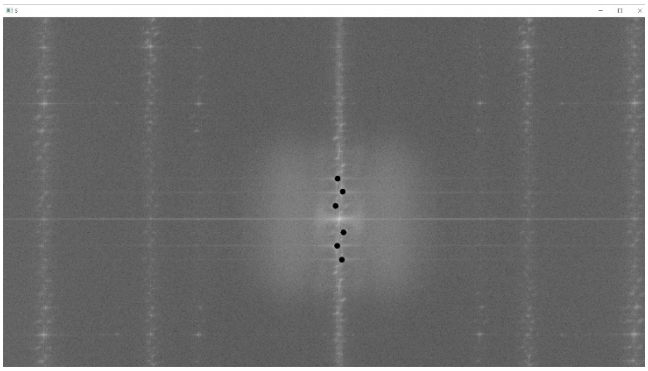
(頻譜圖)

利用 notch filter 去除雜訊

用滑鼠點選雜訊部分，並用黑點遮蓋，結束後按下 esc 離開繪圖。

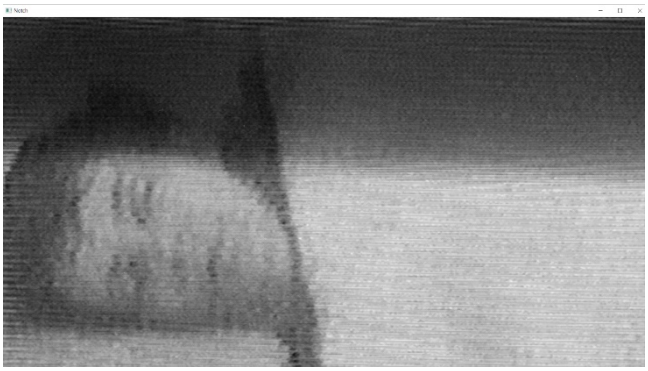
```
#draw circle
def show_xy(event,x,y,flags,param):
    if event==1:    #mouse click
        cv2.circle(S, (x,y), 7, (0,0,0), -1)    #img,coordinate,radius,color,Solid
        cv2.imshow('S',S)
cv2.setMouseCallback('S', show_xy)

while(1):
    key=cv2.waitKey(0)
    if key==27:
        break
cv2.destroyAllWindows()
```



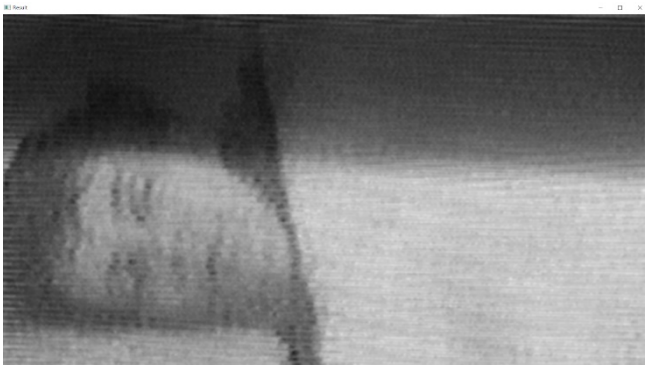
將修改後的頻譜圖乘上原本傅立葉轉換後的值
然後進行 Inverse Fourier Transform，得到過濾後圖像
經過 magnitude()計算幅值、normalize() 標準化，讓圖像以灰階 256 色圖像方式呈現，並轉成 uint8 型態(避免負數)
呈現出圖像。

```
S = np.expand_dims(S, axis=2)   #讓S跟fshift形狀一樣(原本一個二維一個三維)
fshift=DFT*S
idft=cv2.idft(fshift, flags=cv2.DFT_COMPLEX_OUTPUT)
planes2 = cv2.split(idft)
idft = cv2.magnitude(planes2[0], planes2[1])
idft = cv2.normalize(idft, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
```



(經 notch filter 後的圖)

最後再用高斯模糊處理其他雜訊



(高斯模糊後的圖)

Code：

```
import cv2
import numpy as np

img = cv2.imread('image4.png', cv2.IMREAD_GRAYSCALE)

P = cv2.getOptimalDFTSize(img.shape[0])
Q = cv2.getOptimalDFTSize(img.shape[1])

# padding
padded_img = cv2.copyMakeBorder(img, 0, P - img.shape[0], 0, Q - img.shape[1],
cv2.BORDER_CONSTANT, value=0)
padded_img = padded_img.astype(np.float32)

# center image
for i in range(img.shape[0]):
        for j in range(img.shape[1]):
                padded_img[i][j] *= (-1)**(i+j)

# compute DFT
DFT = cv2.dft(padded_img, flags=cv2.DFT_COMPLEX_OUTPUT)
planes = cv2.split(DFT)
# compute spectrum
S = 20 * np.log(cv2.magnitude(planes[0], planes[1]))
```

```
# normalize the output image to [0, 255]
S = cv2.normalize(S, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)

cv2.imshow('S',S)
#以上為作業 3 的部分

#draw circle
def show_xy(event,x,y,flags,param):
    if event==1:        #mouse click
        cv2.circle(S, (x,y), 7, (0,0,0), -1)        #img,coordinate,radius,color,Solid
        cv2.imshow('S',S)
cv2.setMouseCallback('S', show_xy)

while(1):
    key=cv2.waitKey(0)
    if key==27:
        break
cv2.destroyAllWindows()

S = np.expand_dims(S, axis=2)      #讓 S 跟 fshift 形狀一樣(原本一個二維一個三維)
fshift=DFT*S
idft=cv2.idft(fshift, flags=cv2.DFT_COMPLEX_OUTPUT)
planes2 = cv2.split(idft)
idft = cv2.magnitude(planes2[0], planes2[1])
idft = cv2.normalize(idft, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8)
Result=cv2.GaussianBlur(idft,(11,11),0)

#cv2.imshow('Notch',idft)
cv2.imshow('Result',Result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

作業 5

題目敘述：

利用你所學的顏色(Color)的知識與技術，撰寫一個程式來偵測一張輸入照片中的皮膚區域並將其標示出。(請用附件中的三張照片做測試)

Write a program that detect skin color from an image.

開發環境：

Windows10、Spyder (Python 3.8)、OpenCV 4.7.0

說明：

1、

```
img1 = cv2.imread('img1.jpg')
resized_img1=cv2.resize(img1,(int(img1.shape[1]/10), int(img1.shape[0]/10)))
cv2.imshow('original1',resized_img1)
```

讀取圖片 、用 resize()調整圖片大小、顯示原圖

2、hsv_img1 = cv2.cvtColor(resized_img1, cv2.COLOR_BGR2HSV)
將圖片色彩(由 BGR)轉成 HSV

3、

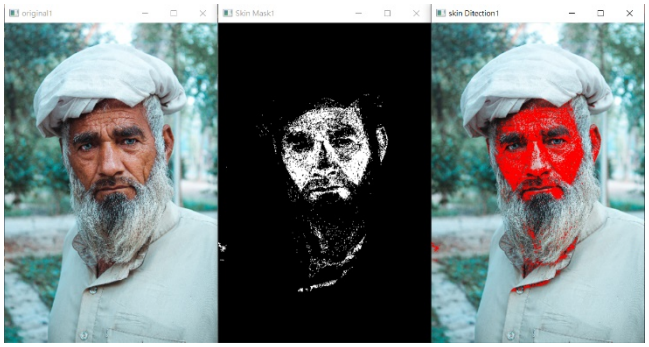
```
lower_skin = np.array([5, 60, 40], dtype=np.uint8)
upper_skin = np.array([40, 180, 255], dtype=np.uint8)
```

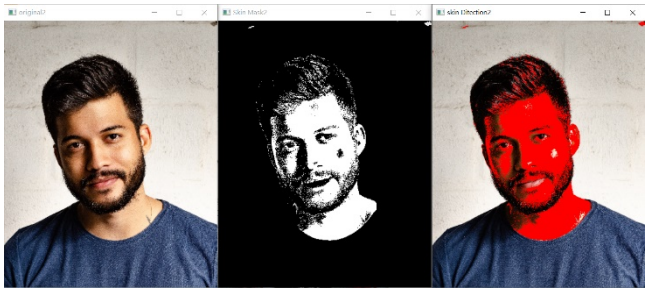
設定膚色範圍(5<H<40, 60<S<180, 40<V<255)

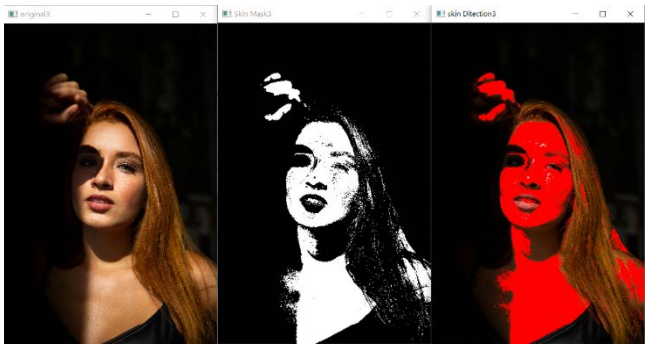
4、skin_mask1 = cv2.inRange(hsv_img1, lower_skin, upper_skin)
cv2.inRange 獲取皮膚區域，建立 mask

5、resized_img1[skin_mask1>0]=(0,0,255)
將膚色區域於原圖標示成紅色

6、最後顯示 mask 以及於原圖上標示膚色區域







Code：

```
import cv2
import numpy as np

img1 = cv2.imread('img1.jpg')
resized_img1=cv2.resize(img1,(int(img1.shape[1]/10), int(img1.shape[0]/10)))
cv2.imshow('original1',resized_img1)
img2 = cv2.imread('img2.jpg')
resized_img2=cv2.resize(img2,(int(img2.shape[1]//6), int(img2.shape[0]//6)))
cv2.imshow('original2',resized_img2)
img3 = cv2.imread('img3.jpg')
resized_img3=cv2.resize(img3,(int(img3.shape[1]//10), int(img3.shape[0]//10)))
cv2.imshow('original3',resized_img3)

hsv_img1 = cv2.cvtColor(resized_img1, cv2.COLOR_BGR2HSV)
hsv_img2 = cv2.cvtColor(resized_img2, cv2.COLOR_BGR2HSV)
hsv_img3 = cv2.cvtColor(resized_img3, cv2.COLOR_BGR2HSV)

lower_skin = np.array([5, 60, 40], dtype=np.uint8)
upper_skin = np.array([40, 180, 255], dtype=np.uint8)

skin_mask1 = cv2.inRange(hsv_img1, lower_skin, upper_skin)
```

```
skin_mask2 = cv2.inRange(hsv_img2, lower_skin, upper_skin)
skin_mask3 = cv2.inRange(hsv_img3, lower_skin, upper_skin)

resized_img1[skin_mask1>0]=(0,0,255)
resized_img2[skin_mask2>0]=(0,0,255)
resized_img3[skin_mask3>0]=(0,0,255)

cv2.imshow('skin Ditection1', resized_img1)
cv2.imshow('skin Ditection2', resized_img2)
cv2.imshow('skin Ditection3', resized_img3)

cv2.imshow('Skin Mask1', skin_mask1)
cv2.imshow('Skin Mask2', skin_mask2)
cv2.imshow('Skin Mask3', skin_mask3)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

作業 6

題目敘述

Run-Length Based Image Compression 練習

附件中為三張利用將晶片高度以色彩視覺化後的圖片。
請設計一個基於 Run-Length 的壓縮法方，對圖檔作無失真壓縮後儲存成新檔案。
部落格上應敘述你的壓縮方法，提供壓縮檔之格式，並計算三張圖的平均壓縮率(compression ratio)。

開發環境

Windows10、Spyder (Python 3.8)、OpenCV 4.7.0

說明

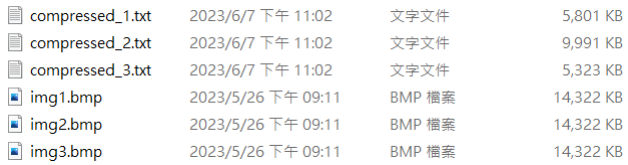
1. 讀入圖檔，並輸出檔案大小
2. cv2.split(image)將圖片分成 b、g、r 三個通道
3. 分別將 flatten()將 b、g、r 存成一維陣列

4. rle_encode 函式處理基於 Run-Length 的壓縮法方法，即用連續出現的次數取代連續重複的數據，例如：[1, 1, 1, 2, 2, 3, 3, 3, 3]->[3, 1, 2, 2, 4, 3]。

```python
def run_length_encode(data):
    encoded_data = []
    count = 1
    for i in range(1, len(data)):
        if data[i] == data[i-1]:
            count += 1 #計算重複的次數
        else:
            encoded_data.append(count)
            encoded_data.append(data[i-1])
            count = 1
    encoded_data.append(count)
    encoded_data.append(data[-1])
    return encoded_data
```

5. 生成.txt 檔案，寫入圖像的長寬尺寸(image.shape[])以及壓縮數據，並輸出壓縮檔之檔案大小
6. 計算平均壓縮率

結果



壓縮檔及原圖

```
img1
original: 14665254 bytes
compressed: 5940087 bytes
compress ratio: 40.50%

img2
original: 14665254 bytes
compressed: 10230759 bytes
compress ratio: 69.76%

img3
original: 14665254 bytes
compressed: 5450436 bytes
compress ratio: 37.17%

平均壓縮率: 49.14%
```

各檔案之原始大小、壓縮後大小、壓縮率
以及平均壓縮率

Code：

```
import cv2
import os

#Run-Length Encoding
def run_length_encode(data):
    encoded_data = []
    count = 1
    for i in range(1, len(data)):
        if data[i] == data[i-1]:
            count += 1 #計算重複的次數
        else:
            encoded_data.append(count)
            encoded_data.append(data[i-1])
            count = 1
    encoded_data.append(count)
    encoded_data.append(data[-1])
    return encoded_data

def get_file_size(filename):
    return os.path.getsize(filename)

files = ['img1.bmp', 'img2.bmp', 'img3.bmp']
orig_total=0
compressed_total=0
for i, file in enumerate(files):
    image = cv2.imread(file)
    orig_size = get_file_size(file)
    print(f"img{i+1}\noriginal: {orig_size} bytes")
    orig_total+=orig_size

    b, g, r = cv2.split(image)

    b_flat = b.flatten()
    g_flat = g.flatten()
    r_flat = r.flatten()

    b_encoded = run_length_encode(b_flat)
    g_encoded = run_length_encode(g_flat)
```

```
        r_encoded = run_length_encode(r_flat)

        compressed_file = f'compressed_{i+1}.txt'
        with open(compressed_file, 'w') as f:
            f.write(f"{image.shape[0]}\n")
            f.write(f"{image.shape[1]}\n")
            f.write(','.join(map(str, b_encoded)) + '\n')
            f.write(','.join(map(str, g_encoded)) + '\n')
            f.write(','.join(map(str, r_encoded)) + '\n')
            compressed_size = get_file_size(compressed_file)
        print(f"compressed: {compressed_size} bytes")
        compressed_total+=compressed_size

        print(f"compress ratio: {(compressed_size/orig_size)*100:.2f}%\n")
print(f"平均壓縮率: {(compressed_total/orig_total)*100:.2f}%")
```