```
import os
import cv2 as cv2
import numpy as np
from matplotlib import pyplot as plt
def rgb2hsv(img):            # 轉換成 hsv, input 參數 = RGB 圖像, output = H,S,V
    r,g,b = cv2.split(img)          # 分離 RGB
    r, g, b = r/255.0, g/255.0, b/255.0
    # HSV 分別承接經過公式轉換後的 H,S,V 值      # 高(直行)img.shape[0]      # 寬(橫列)img.shape[1]    # \ 多行語句
    H, S, V = np.zeros((img.shape[0], img.shape[1]), np.float32), \
        np.zeros((img.shape[0], img.shape[1]), np.float32),   np.zeros((img.shape[0], img.shape[1]), np.float32)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            max_ = max((b[i, j], g[i, j], r[i, j]))
            min_ = min((b[i, j], g[i, j], r[i, j]))
            # H
            if max_ == min_:                # 當 max = min 時，相位角為'0 度'
                H[i, j] = 0
            elif max_ == r[i, j]:          # 當 r 為 max 時，相位角為'60 度'
                if g[i, j] >= b[i, j]:          # 當 g >= b 時，+0
                    H[i, j] = (60 * ((g[i, j]) - b[i, j]) / (max_ - min_))
                else:                                # 當 g < b 時，+360
                    H[i, j] = (60 * ((g[i, j]) - b[i, j]) / (max_ - min_))+360
            elif max_ == g[i, j]:           # 當 g 為 max 時，相位角為'60 度' + 120
                H[i, j] = 60 * ((b[i, j]) - r[i, j]) / (max_ - min_) + 120
            elif max_ == b[i, j]:           # 當 b 為 max 時，相位角為'60 度' + 240
                H[i, j] = 60 * ((r[i, j]) - g[i, j]) / (max_ - min_)+ 240
            # hsv 中的值是 0-360, 0-1, 0-1，但是在 openCV 中 hsv 的值是 0-180, 0-255, 0-255
            H[i,j] =int( H[i,j] / 2)
            # S
            if max_ == 0:
                S[i, j] = 0
```

```
                else:
                    S[i, j] =int( (max_ - min_)/max_*255)
                # V
                V[i, j] =int( max_*255)
                # print(H[i, j],S[i, j],V[i, j])
        #  合併輸出
        # HSV = cv2.merge([H,S,V])
        # HSV=np.array(HSV,dtype='uint8')
        # return HSV
        #  不合併輸出
        return H, S, V

def merge_hsv(h, s, v):    #  合併 HSV, input 參數  = H,S,V, output =    HSV 圖像
        hsv = cv2.merge([h, s, v])          #  合併 HSV
        hsv=np.array(hsv,dtype='uint8')
        return hsv

def hsv_2_bi(h, s, v):          # HSV  圖像轉換為黑白影像, input = h, s, v, output =  黑白影像
        bi_img_hav = np.zeros((img.shape[0], img.shape[1]), np.uint8)
        for i in range(img.shape[0]):
            for j in range(img.shape[1]):
                if (116 > h[i, j]) and 80 < h[i, j] \
                    and 255 > s[i, j] and 48 < s[i, j] \
                    and 255 > v[i, j] and 50 < v[i, j]:
                    bi_img_hav[i, j] = 255
                else:
                    bi_img_hav[i, j] = 0
        bi_img_hav = np.array(bi_img_hav,dtype='uint8')
        return bi_img_hav

def rgb_2_bi(img):          # RGB  圖像轉換為黑白影像, input =  圖像, output =  黑白影像
        b,g,r = cv2.split(img)          #  分離 RGB
        bi_img_rgb = np.zeros((img.shape[0], img.shape[1]), np.uint8)
        for i in range(img.shape[0]):
```

```
            for j in range(img.shape[1]):
                max_ = max((b[i, j], g[i, j], r[i, j]))
                min_ = min((b[i, j], g[i, j], r[i, j]))
                if r[i, j] > 95 and b[i, j] > 20 and (max_ - min_) > 15 and r[i, j] > g[i, j] and r[i,
j] > b[i, j]:
                    bi_img_rgb[i, j] = 255
                else:
                    bi_img_rgb[i, j] = 0

        # rgb = cv2.merge([r, g, b])          # bgr 改用 rgb 存
        # rgb=np.array(rgb,dtype='uint8')
        bi_img_rgb = np.array(bi_img_rgb,dtype='uint8')
        return bi_img_rgb

def read_img(path):         # 讀檔, input = 影像路徑, output = 圖像,檔名,副檔名
        img_path = path
        img_filepath = os.path.splitext(img_path)[0]       # 拆分路徑 & 副檔名，0 為路徑
        img_fileextension = os.path.splitext(img_path)[1]   # 1 為副檔名
        img_filename = os.path.basename(img_filepath)        # 取出檔名不含副檔名
        img = cv2.imread(img_filename + img_fileextension,-1)   # 讀檔
        return img, img_filename, img_fileextension

img, file_name, file_extension = read_img('E:/Program_File/PYTHON/數位影像處理作業
/HW_2/hand6.jpg')



h, s, v = rgb2hsv(img)
hsv = merge_hsv(h, s, v)
bi_hsv = hsv_2_bi(h, s, v)   # 黑白 HSV
bi_rgb = rgb_2_bi(img)          # 黑白 RGB

kernel = cv2.getStructuringElement(cv2.MORPH_RECT,(3, 3)) # 4 連通，
# kernel = np.ones((3,3),np.uint8)
bi_hsv_close = cv2.morphologyEx(bi_hsv, cv2.MORPH_CLOSE, kernel, iterations=1)   #
```

```
閉合，先擴張 後侵蝕，去除圖像中的小黑點
bi_hsv_close_open = cv2.morphologyEx(bi_hsv_close, cv2.MORPH_OPEN, kernel, iterations=1)  # 斷開，先侵蝕 後擴張，去除圖像中的小亮點
bi_rgb_close = cv2.morphologyEx(bi_rgb, cv2.MORPH_CLOSE, kernel, iterations=1)  # 閉合，先擴張 後侵蝕，去除圖像中的小黑點
bi_rgb_close_open = cv2.morphologyEx(bi_rgb_close, cv2.MORPH_OPEN, kernel, iterations=1)  # 斷開，先侵蝕 後擴張，去除圖像中的小亮點

# cv2.imshow('img_RGB', img)
# cv2.imshow('img_HSV', hsv)
cv2.imshow('img_bi_hsv', bi_hsv)
cv2.imshow('img_bi_rgb', bi_rgb)
cv2.imshow('img_bi_hsv_open_close', bi_hsv_close_open)
cv2.imshow('img_bi_rgb_open_close', bi_rgb_close_open)
# cv2.imwrite(file_name + '_hsv' + file_extension,hsv)#  寫檔
# cv2.imwrite(file_name + '_bi_hsv' + file_extension,bi_hsv)#  寫檔
# cv2.imwrite(file_name + '_bi_rgb' + file_extension,bi_rgb)#  寫檔
# cv2.imwrite(file_name + '_bi_hsv_close_open' + file_extension,bi_hsv_close_open)# 寫檔
# cv2.imwrite(file_name + '_bi_rgb_close_open' + file_extension,bi_rgb_close_open)# 寫檔
cv2.waitKey()
```

結果圖

1.

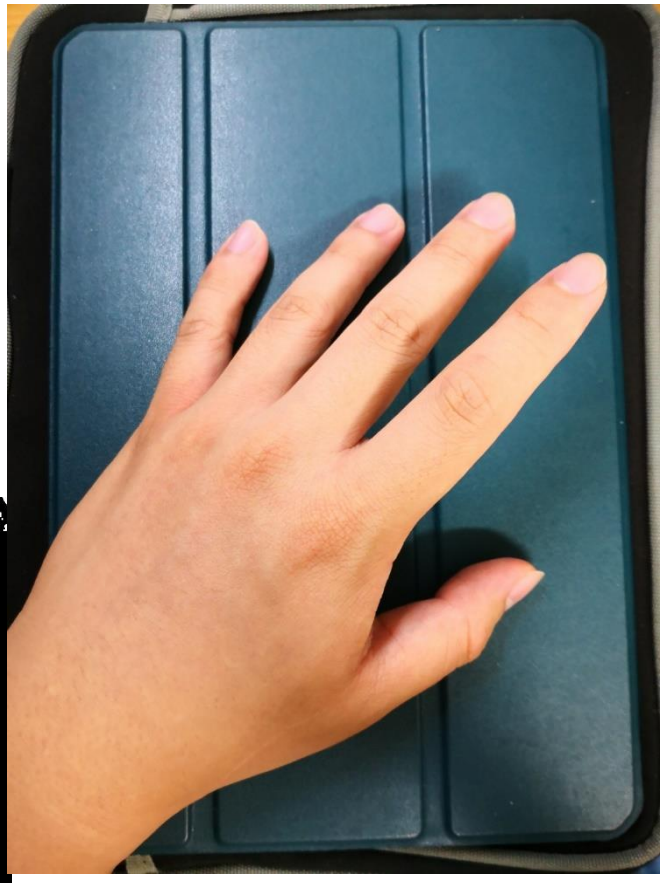
上：原圖

右：HSV

下：RGB

2.

上：原圖



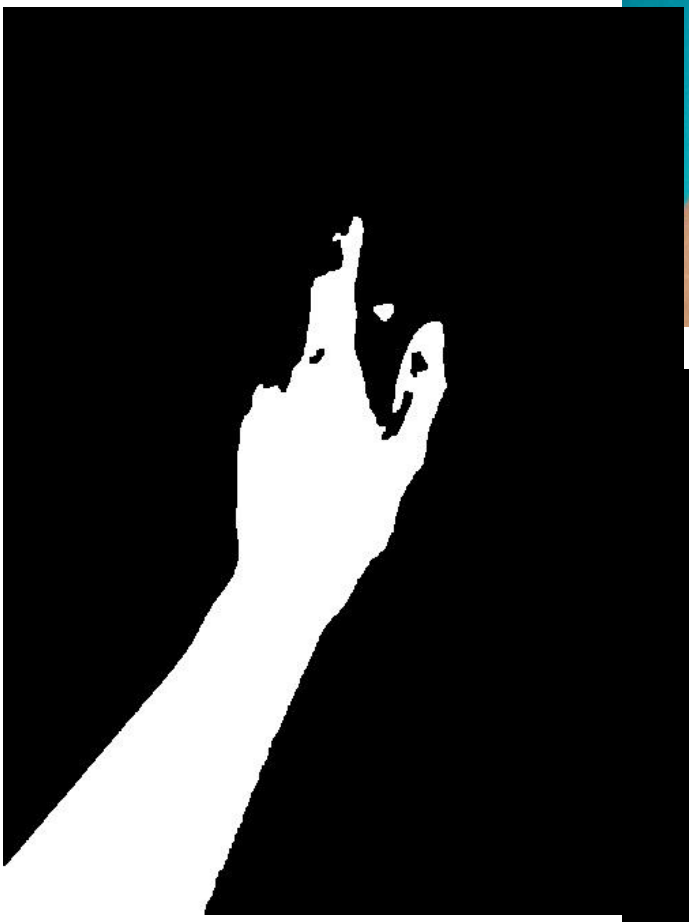
左：HSV

下：RGB

3.

上：原圖

左：HSV



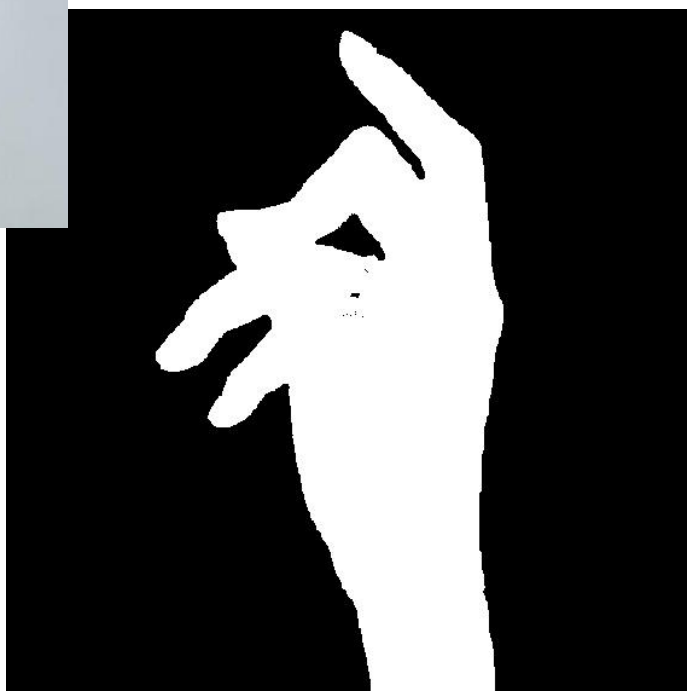
下：RGB

心得：

在課堂中學習到的說法是說，當要取出膚色時，由於 RGB 會容易將附近其餘雜色取出，因此較難框出手部的範圍，因此要學習 HSV 並將 RGB 圖像轉換為 HSV 並取出符合膚色範圍的數值，以此框出手部範圍，但是我自己在時做後發現不同背景的情況下會有截然不同的表現。

在 HSV 時，當手部與背景顏色較接近時(橘黃，木色)手部難與背景分隔； 在 RGB 時，當背景為紅色調時，會很難將手部取出，例如結果圖 1，除了取出手部以外，也會將背景的紅色花紋一起取出

但是我嘗試淺色的背景色彩時，發現用 HSV 反而不如使用 RGB 取的結果，上方為 HSV 取出的結果，下方為 RGB 取出的結果，左方為原圖



在複雜背景時使用 HSV 以及 RGB 效果都並不好，都會有一定範圍的雜訊，令手部形狀不完整

Open 斷開，先侵蝕 後擴張，去除圖像中的小亮點
Close 閉合，先擴張 後侵蝕，去除圖像中的小黑點

在這組圖像中主要的雜訊在手部中的黑色雜訊，因此在此圖像中要用 close 效果顯著，使用 open 會導致手部中的雜訊近一步放大

右 HSV

左下 HSV 做 open

右下 HSV 做 close