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.

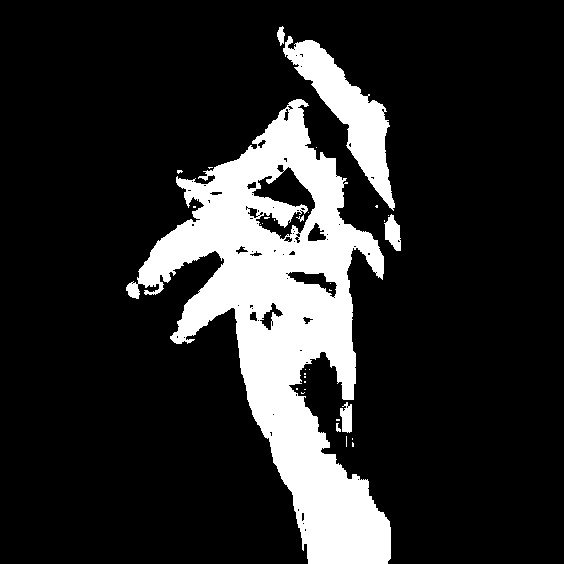
上 : 原圖

左 : HSV 

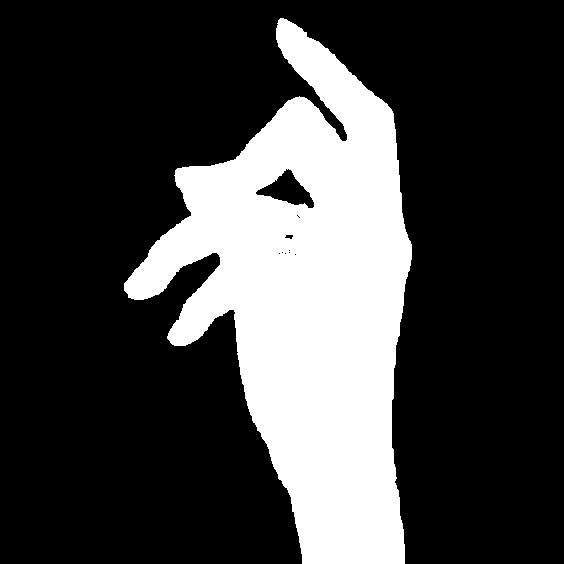
下 : RGB

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