影像處理HW2

內容：針對任意背景的手部影像三張(來源自己拍)，框選出手部的區域

函數： Convert to HSV Space

- 依據課程講義所學

HSV color space

$$
\begin{aligned}
&H \in [0 \,..\, 360];\ S, V, R, G, B \in [0, 1] \\
&MAX = \max(R, G, B);\ MIN = \min(R, G, B) \\
&H = \begin{cases}
\text{undefined}, & \text{if } MAX = MIN \\
60 \times \frac{G-B}{MAX-MIN} + 0, & \text{if } MAX = R \text{ and } G \geq B \\
60 \times \frac{G-B}{MAX-MIN} + 360, & \text{if } MAX = R \text{ and } G < B \\
60 \times \frac{B-R}{MAX-MIN} + 120, & \text{if } MAX = G \\
60 \times \frac{R-G}{MAX-MIN} + 240, & \text{if } MAX = B
\end{cases} \\
&S = \begin{cases}
0, & \text{if } MAX = 0 \\
1 - \frac{MIN}{MAX}, & \text{otherwise}
\end{cases} \\
&V = MAX
\end{aligned}
$$

  - 先將影像 b, g, r 值，依序取出，再除255
  - 最後依據公式轉換後，須分別做正規化
    - h[0...180]、s[0...255]、v[0...255]

```
img = cv2.imread("finger11.jpg", -1)

weight = img.shape[0]
height = img.shape[1]

hsv = np.zeros_like(img)

for x in range(weight):
    for y in range(height):
        b, g, r = img[x, y]

        # b, g, r [0,1]
        b = b / 255
        g = g / 255
        r = r / 255

        MAX = max(r,g,b)
```

```
        MIN = min(r,g,b)

        # H[0..360]
        if MAX == MIN:  h = 0
        elif MAX == r and g >= b:   h = 60 * (g - b) / (MAX - MIN)
        elif MAX == r and g < b:    h = 60 * (g - b) / (MAX - MIN) + 360
        elif MAX == g:  h = 60 * (b - r) / (MAX - MIN) + 120
        elif MAX == b:  h = 60 * (r - g) / (MAX - MIN) + 240

        # s[0,1]
        if MAX == 0:    s = 0
        else:   s = 1 - MIN / MAX

        # v[,,1]
        v = MAX

        # 正規化
        h = h / 360 * 180
        s = s * 255
        v = v * 255
        hsv[x,y] = np.array([h,s,v])
```

函數： Morphological Operation：Open、Close

- Opening: Erosion then Dilation
  - $A \circ B = A \ominus B \oplus B$

```
def Morphology_open(img):
    img = Morphology_erode(img)
    img = Morphology_dilate(img)
    return img
```

- Closing : Dilation then Erosion
  - $A \circ B = (A \oplus B) \ominus B$

```
def Morphology_close(img):
    img = Morphology_dilate(img)
    img = Morphology_erode(img)
    return img
```

- dilate
  - Take the maximum under the kernel
  - (op: pixel wise AND, then OR)
  - 程式碼使用 8-connected kernel，結果較顯著

```
def Morphology_dilate(img):
    image = img.copy()
    height = img.shape[0]
    weight = img.shape[1]
    # 8-connected kernel
    kernel = np.array(((255, 255, 255),(255, 255, 255),(255, 255, 255)),
dtype=np.int)

    for x in range(1,height - 1):
        for y in range(1,weight - 1):
            edge = img[x - 1:x + 2,y - 1:y + 2]
            edge = np.bitwise_and(edge,kernel)
            image[x - 1,y - 1] = np.max(edge)
    return image
```

- erode
- Take the minimum under the kernel
    - (op: pixel wise AND, then AND)
    - 程式碼使用 8-connected kernel，結果較顯著

```
def Morphology_erode(img):
    image = img.copy()
    height = img.shape[0]
    weight = img.shape[1]
    # 8-connected kernel
    kernel = np.array(((255, 255, 255),(255, 255, 255),(255, 255, 255)),
dtype=np.int)

    for x in range(1,height - 1):
        for y in range(1,weight - 1):
            edge = img[x - 1:x + 2,y - 1:y + 2]
            edge = np.bitwise_and(edge,kernel)
            image[x - 1,y - 1] = np.min(edge)
    return image
```

HSV : 將不同背景手的照片的手部區域正確標記出

- 將hsv圖片取出膚色範圍
    - 運算時 H[0...360] (色調角度)、S[0...100] (飽和度 0% - 100%)、V[0...100] (強度 0% - 100%)
    - H 取值[0,40]、S 取值[30,100]、 V 取值 [30,100]
        - 範圍內 => pixel = 255，範圍外 => pixel = 0

```
weight = hsv.shape[1]
height = hsv.shape[0]

hsv_catch = hsv.copy( )


for x in range(height):
    for y in range(weight):
        H = hsv[x,y,0] * 2
        S = hsv[x,y,1] / 255 * 100
        V = hsv[x,y,2] / 255 * 100
```

```
        if not ( H >= 0 and H <= 40 and S >= 30 and S <= 100 and V >= 30 and V
<= 100 ):
            hsv_catch[x,y,0] = hsv_catch[x,y,1] = hsv_catch[x,y,2] = 0
        else:
            hsv_catch[x,y,0] = hsv_catch[x,y,1] = hsv_catch[x,y,2] = 255
```

RGB: 將不同背景手的照片的手部區域正確標記出

- 將rbg圖片取出膚色範圍
  - b 取值[0,255]、g 取值[0,210]、 r 取值 [150,230]
    - 範圍內 => pixel = 255，範圍外 => pixel = 0

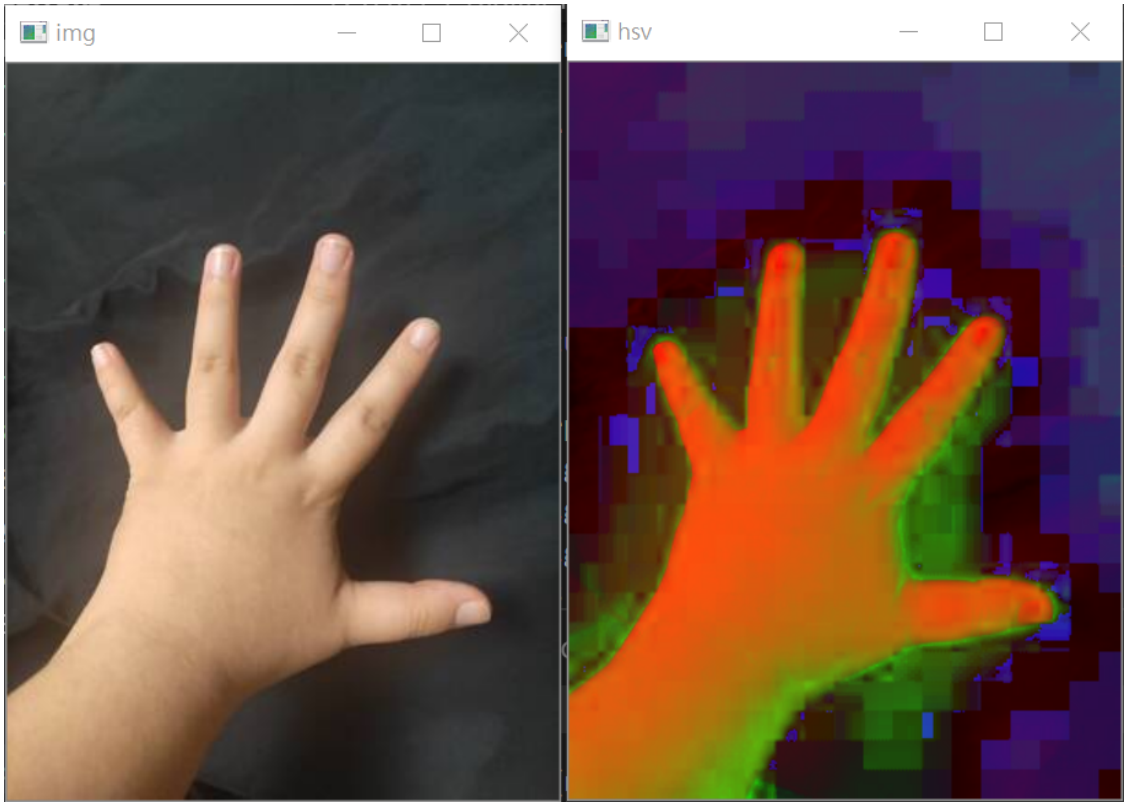
```
weight = img.shape[1]
height = img.shape[0]

for x in range(height):
    for y in range(weight):
        b, g, r = img[x, y]
        if not ( b >= 0 and b <= 255 and g >= 0 and g <= 210 and r >= 150 and r
<= 230 ):
            img[x,y,0] = img[x,y,1] = img[x,y,2] = 0
        else:
            img[x,y,0] = img[x,y,1] = img[x,y,2] = 255
```

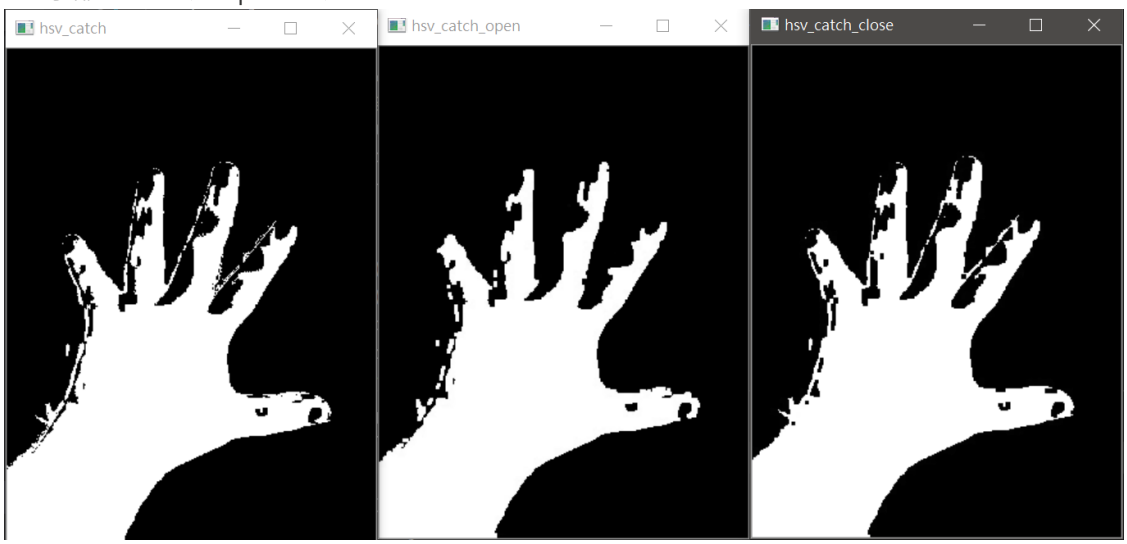
結果圖

image1

- 原圖 vs hsv



- hsv手部區域 vs 經 open vs 經 close



- rgb手部區域 vs 經 open vs 經 close

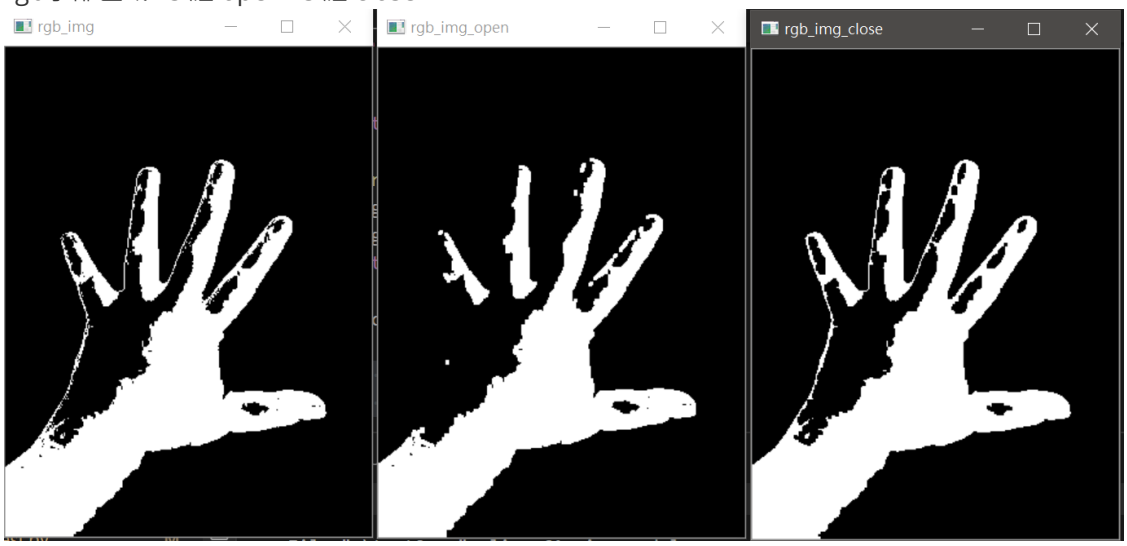
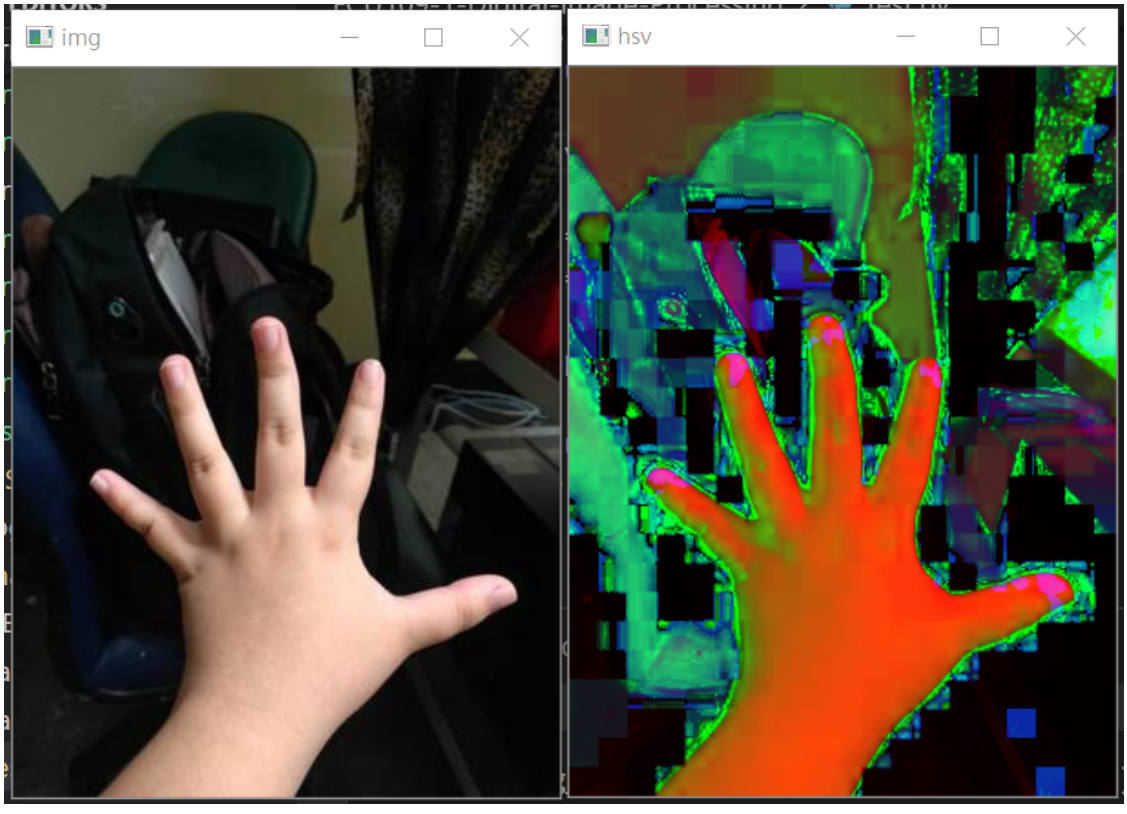
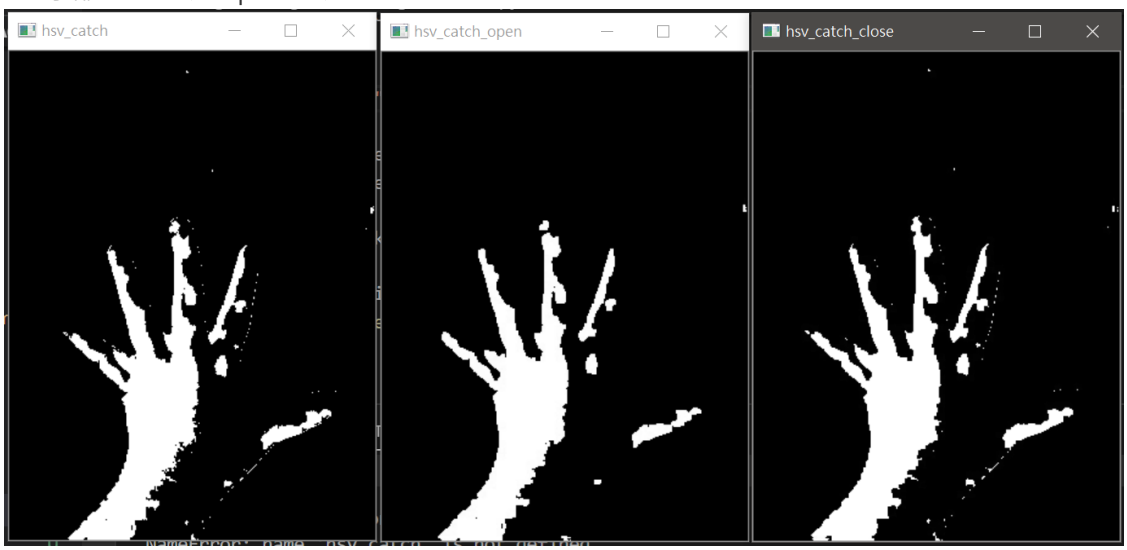


image2

- 原圖 vs hsv



- hsv手部區域 vs 經 open vs 經 close



- rgb手部區域 vs 經 open vs 經 close

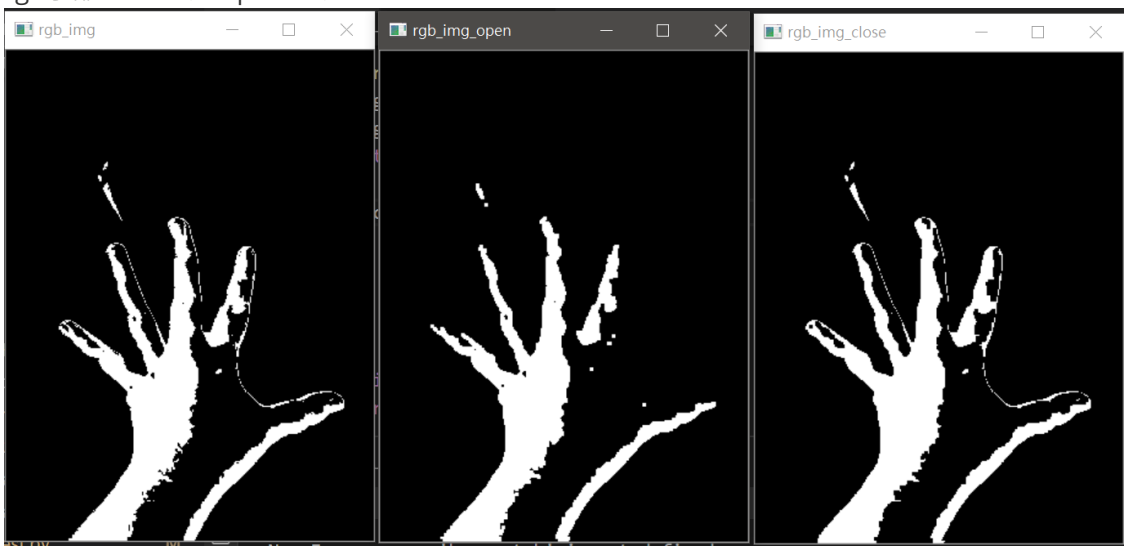
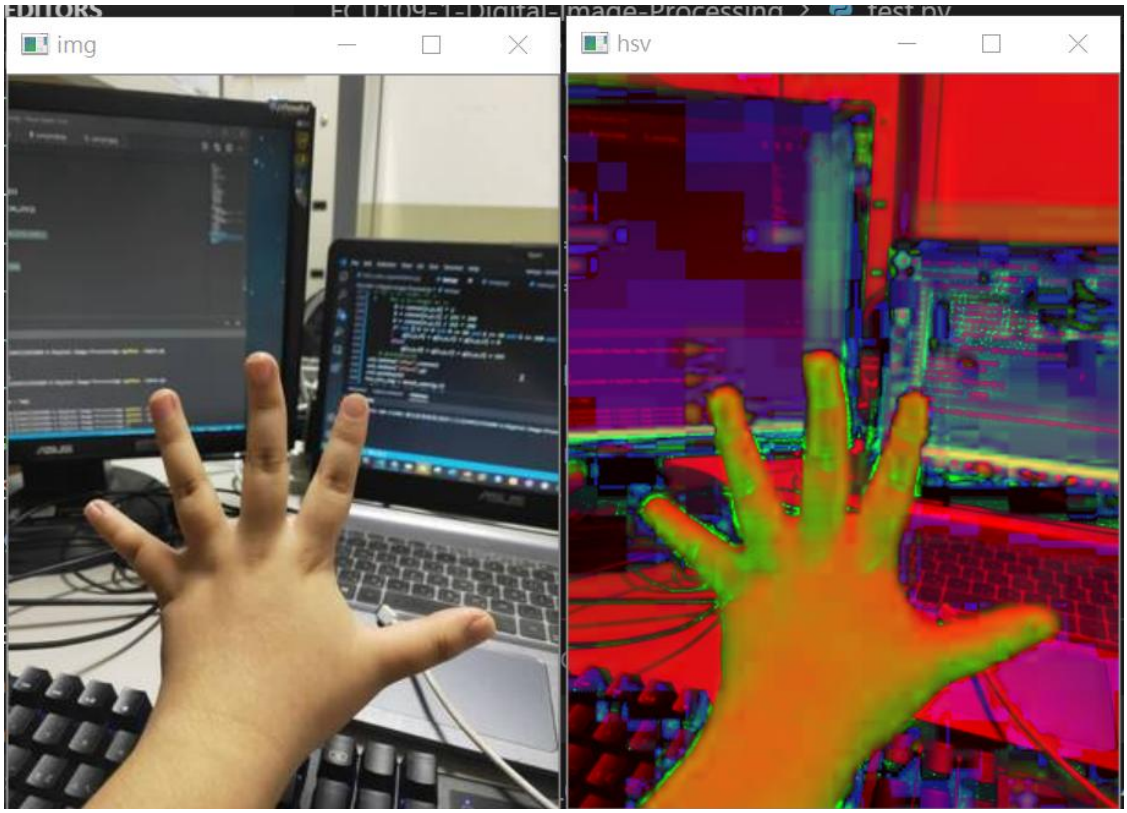
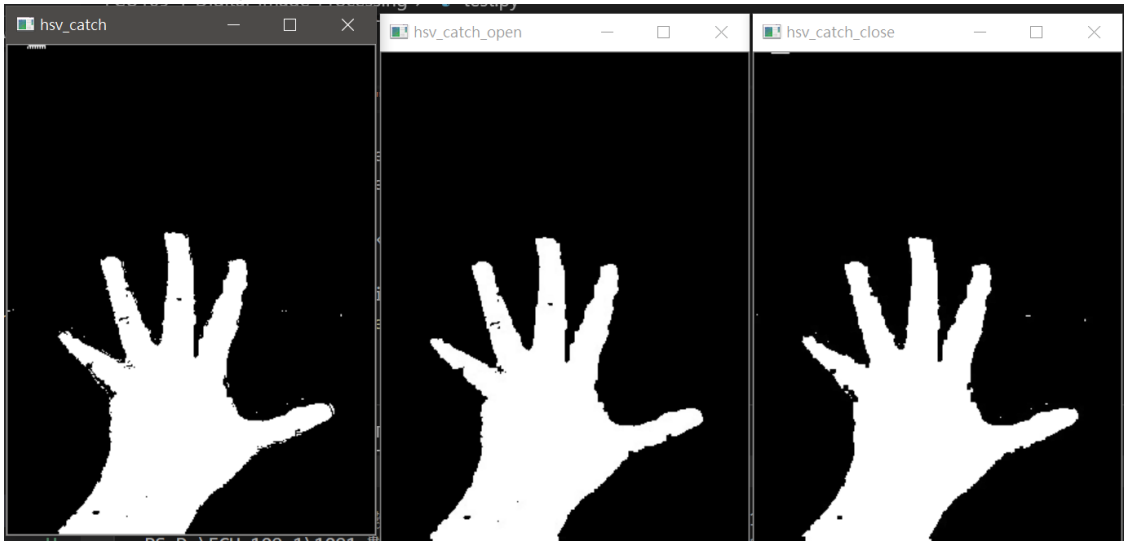


image3

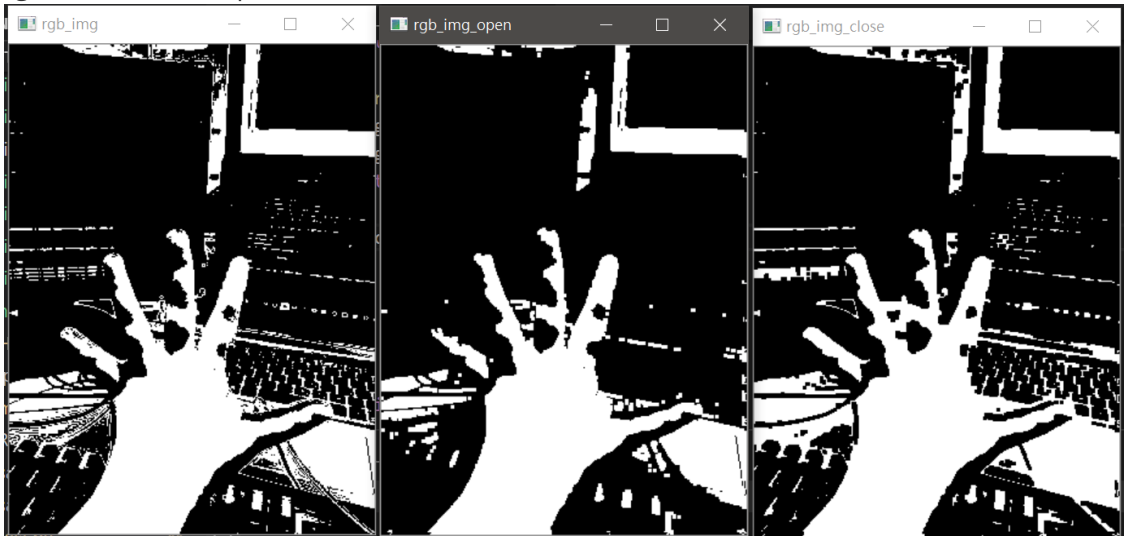
- 原圖 vs hsv



- hsv手部區域 vs 經 open vs 經 close



- rgb手部區域 vs 經 open vs 經 close



心得

在這次的功課中，非常的有一點難QQ，主要是卡在期中考週，又不想要用套件去完成作業，hsv的效果比起rgb來說hsv的範圍如果在正確的範圍內搜尋，會是看起來非常完美的圈到手的範圍。但rgb的框起來比較好一點，hsv要去找範圍有點麻煩ㄟ，真佩服那些做套件的人。

完整code

- HSV

```
import numpy as np
import cv2

def Morphology_dilate(img):
    image = img.copy()
    height = img.shape[0]
    weight = img.shape[1]
    # 8-connected kernel
    kernel = np.array(((255, 255, 255),(255, 255, 255),(255, 255, 255)),
dtype=np.int)

    for x in range(1,height - 1):
        for y in range(1,weight - 1):
            edge = img[x - 1:x + 2,y - 1:y + 2]
            edge = np.bitwise_and(edge,kernel)
            image[x - 1,y - 1] = np.max(edge)
    return image

def Morphology_erode(img):
    image = img.copy()
    height = img.shape[0]
    weight = img.shape[1]
    # 8-connected kernel
    kernel = np.array(((255, 255, 255),(255, 255, 255),(255, 255, 255)),
dtype=np.int)

    for x in range(1,height - 1):
        for y in range(1,weight - 1):
            edge = img[x - 1:x + 2,y - 1:y + 2]
            edge = np.bitwise_and(edge,kernel)
            image[x - 1,y - 1] = np.min(edge)
    return image

def Morphology_open(img):
    img = Morphology_erode(img)
    img = Morphology_dilate(img)
    return img

def Morphology_close(img):
    img = Morphology_dilate(img)
    img = Morphology_erode(img)
    return img

img = cv2.imread("finger81.jpg", -1)

weight = img.shape[0]
height = img.shape[1]
```

```
hsv = np.zeros_like(img)

for x in range(weight):
    for y in range(height):
        b, g, r = img[x, y]

# b, g, r [0,1]
        b = b / 255
        g = g / 255
        r = r / 255

        MAX = max(r,g,b)
        MIN = min(r,g,b)
        # H[0..360]
        if MAX == MIN:  h = 0
        elif MAX == r and g >= b:   h = 60 * (g - b) / (MAX - MIN)
        elif MAX == r and g < b:    h = 60 * (g - b) / (MAX - MIN) + 360
        elif MAX == g:  h = 60 * (b - r) / (MAX - MIN) + 120
        elif MAX == b:  h = 60 * (r - g) / (MAX - MIN) + 240

        # s[0,1]
        if MAX == 0:    s = 0
        else:   s = 1 - MIN / MAX

        # v[,,1]
        v = MAX

        h = h / 360 * 180
        s = s * 255
        v = v * 255
        hsv[x,y] = np.array([h,s,v])

weight = hsv.shape[1]
height = hsv.shape[0]

hsv_catch = hsv.copy( )


for x in range(height):
    for y in range(weight):
        H = hsv[x,y,0] * 2
        S = hsv[x,y,1] / 255 * 100
        V = hsv[x,y,2] / 255 * 100
        if not ( H >= 0 and H <= 40 and S >= 30 and S <= 100 and V >= 30 and V
<= 100 ):
            hsv_catch[x,y,0] = hsv_catch[x,y,1] = hsv_catch[x,y,2] = 0
        else:
            hsv_catch[x,y,0] = hsv_catch[x,y,1] = hsv_catch[x,y,2] = 255

cv2.imshow("img",img)
cv2.imshow("hsv",hsv)
cv2.imshow("hsv_catch",hsv_catch)
cv2.waitKey(0)

gray = cv2.imwrite("hsv_catch.jpg",hsv_catch)
hsv_catch = cv2.imread("hsv_catch.jpg",0)

hsv_catch_close = Morphology_close(hsv_catch)
```

```
hsv_catch_open = Morphology_open(hsv_catch)
cv2.imshow('hsv_catch_close',hsv_catch_close)
cv2.imshow('hsv_catch_open',hsv_catch_open)

cv2.waitKey(0)
```

- RGB

```
import numpy as np
import cv2
def Morphology_dilate(img):
    image = img.copy()
    height = img.shape[0]
    weight = img.shape[1]
    # 8-connected kernel
    kernel = np.array(((255, 255, 255),(255, 255, 255),(255, 255, 255)),
dtype=np.int)

    for x in range(1,height - 1):
        for y in range(1,weight - 1):
            edge = img[x - 1:x + 2,y - 1:y + 2]
            edge = np.bitwise_and(edge,kernel)
            image[x - 1,y - 1] = np.max(edge)
    return image

def Morphology_erode(img):
    image = img.copy()
    height = img.shape[0]
    weight = img.shape[1]
    # 8-connected kernel
    kernel = np.array(((255, 255, 255),(255, 255, 255),(255, 255, 255)),
dtype=np.int)

    for x in range(1,height - 1):
        for y in range(1,weight - 1):
            edge = img[x - 1:x + 2,y - 1:y + 2]
            edge = np.bitwise_and(edge,kernel)
            image[x - 1,y - 1] = np.min(edge)
    return image

def Morphology_open(img):
    img = Morphology_erode(img)
    img = Morphology_dilate(img)
    return img

def Morphology_close(img):
    img = Morphology_dilate(img)
    img = Morphology_erode(img)
    return img

img = cv2.imread("finger81.jpg", -1)

weight = img.shape[1]
height = img.shape[0]
```

```
for x in range(height):
    for y in range(weight):
        b, g, r = img[x, y]
        if not ( b >= 0 and b <= 255 and g >= 0 and g <= 212 and r >= 130 and r
<= 230 ):
            img[x,y,0] = img[x,y,1] = img[x,y,2] = 0
        else:
            img[x,y,0] = img[x,y,1] = img[x,y,2] = 255



cv2.imshow("rgb_img",img)
cv2.waitKey(0)

rgb_img_close = Morphology_close(img)
rgb_img_open = Morphology_open(img)
cv2.imshow('rgb_img_close',rgb_img_close)
cv2.imshow('rgb_img_open',rgb_img_open)

cv2.waitKey(0)
```