Convert to HSV Space

```
def RGBTOHSV(image):
    r, g, b = cv2.split(image)
    r = np.true_divide(r, 255, dtype=np.float)
    g = np.true_divide(g, 255, dtype=np.float)
    b = np.true_divide(b, 255, dtype=np.float)
    HSV = np.zeros(image.shape, dtype=np.uint8)
    # print(image.shape)
    # print(type(r))

    for i in range(r.shape[0]):
        for j in range(r.shape[1]):
            channelMax = max(r[i][j], g[i][j], b[i][j])
            channelMin = min(r[i][j], g[i][j], b[i][j])
            # print(r[i][j], g[i][j], b[i][j])
            delta = channelMax - channelMin
            if(delta < 0.00001):
                HSV[i][j][0] = 0
                HSV[i][j][1] = 0

            else:

                #s channel
                if(channelMax > 0):
                    HSV[i][j][1] = delta / channelMax * 255
                else:
                    HSV[i][j][1] = 0

                #H channel
                if(r[i][j] >= channelMax):
                    # print('delta', delta)
                    # print((g[i][j] - b[i][j]) / delta * 60)
                    HSV[i][j][0] = (g[i][j] - b[i][j]) / delta *
60
                elif(g[i][j] >= channelMax):
                    HSV[i][j][0] = (2 + (b[i][j] - r[i][j]) /
delta) * 60
                else:
```

```
                    HSV[i][j][0] = (4 + (r[i][j] - g[i][j]) /
delta) * 60

                if(HSV[i][j][0] < 0):
                    HSV[i][j][0] += 179

            #v channel
            HSV[i][j][2] = channelMax * 255

    return HSV
```

把圖像分開RGB channel 找出最大最小值根據RGB to HSV的數學定義把計算得到的值放到hsv的每個channel中

Dilate

```
def dilate(image, kernel):
    widthHalf = kernel.shape[0]//2
    heighHalf = kernel.shape[1]//2
    newImg = np.ones((image.shape[0],img.shape[1]),np.uint8)
    for row in range(image.shape[0]):
        for column in range(image.shape[1]):
            startRow = row - widthHalf
            startColumn = column - heighHalf
            Max = 1
            for kernelRow in range(kernel.shape[0]):
                for kernelColumn in range(kernel.shape[1]):
```

```
                    try:
                        if(kernel[kernelRow][kernelColumn] == 1
and image[startRow+kernelRow][startColumn+kernelColumn] > Max):
                            Max = image[startRow+kernelRow]
[startColumn+kernelColumn]
                    except:
                        pass
            newImg[row][column] = Max
    return newImg
```

根據kernel的大小設定要尋找的範圍，並找出最大值給予當前的位置。

Erode

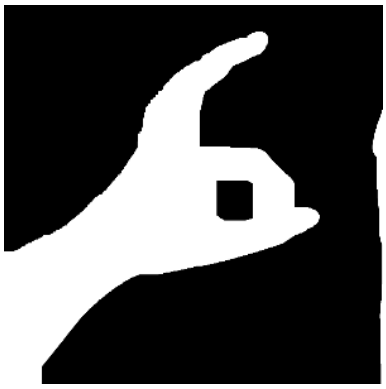
```
def erode(image, kernel):
    widthHalf = kernel.shape[0]//2
    heighHalf = kernel.shape[1]//2
    newImg = np.ones((image.shape[0],image.shape[1]),np.uint8)
    for row in range(image.shape[0]):
        for column in range(image.shape[1]):
            startRow = row - widthHalf
            startColumn = column - heighHalf
            Min = 255
            for kernelRow in range(kernel.shape[0]):
                for kernelColumn in range(kernel.shape[1]):
                    try:
                        if(kernel[kernelRow][kernelColumn] == 1
and image[startRow+kernelRow][startColumn+kernelColumn] < Min):
                            Min = image[startRow+kernelRow]
[startColumn+kernelColumn]
                    except:
                        pass
            newImg[row][column] = Min
    return newImg
```

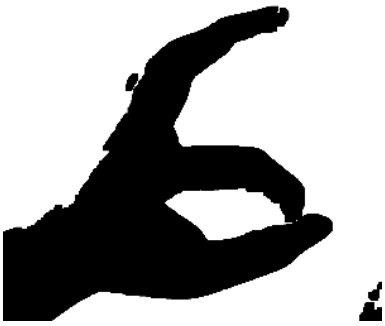
根據kernel的大小設定要尋找的範圍，並找出最小值給予當前的位置。

RGB原圖



HSV 結果





RGB結果



程式碼

```
image1 = cv2.imread('1.jpg')
image2 = cv2.imread('2.jpg')
image3 = cv2.imread('3.jpg')
image1 = cv2.resize(image1, (960, 540))
# image2 = cv2.resize(image2, (960, 540))
# image3 = cv2.resize(image3, (960, 540))

cv2.imshow('rgb1', cv2.resize(image1, (300, 300)))
cv2.imshow('rgb2', cv2.resize(image2, (300, 300)))
cv2.imshow('rgb3', cv2.resize(image3, (300, 300)))

# image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2RGB)
# image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2RGB)
# image3 = cv2.cvtColor(image3, cv2.COLOR_BGR2RGB)

HSVimage1 = cv2.cvtColor(image1, cv2.COLOR_RGB2HSV)
HSVimage2 = cv2.cvtColor(image2, cv2.COLOR_RGB2HSV)
HSVimage3 = cv2.cvtColor(image3, cv2.COLOR_RGB2HSV)

cv2.imshow('hsv1', cv2.resize(HSVimage1, (300, 300)))
cv2.imshow('hsv', cv2.resize(RGBTOHSV(image1), (300, 300)))
cv2.waitKey(0)
cv2.imshow('hsv2', cv2.resize(HSVimage2, (300, 300)))
cv2.imshow('hsv3', cv2.resize(HSVimage3, (300, 300)))


rgbLow = np.array([80, 60 ,70])
rgbHigh = np.array([250,230,200])
kernel = np.ones((50, 50),  np.int8)

HSVMask1 = cv2.inRange(HSVimage1, rgbLow, rgbHigh)
HSVMask1 = cv2.morphologyEx(HSVMask1, cv2.MORPH_CLOSE, kernel)

HSVMask2 = cv2.inRange(HSVimage2, rgbLow, rgbHigh)
HSVMask2 = cv2.morphologyEx(HSVMask2, cv2.MORPH_CLOSE, kernel)

HSVMask3 = cv2.inRange(HSVimage3, rgbLow, rgbHigh)
```

```
HSVMask3 = cv2.morphologyEx(HSVMask3, cv2.MORPH_CLOSE, kernel)


cv2.imshow('HSVMask1', cv2.resize(HSVMask1, (300, 300)))
cv2.imshow('HSVMask2', cv2.resize(HSVMask2, (300, 300)))
cv2.imshow('HSVMask3', cv2.resize(HSVMask3, (300, 300)))

RGBMask1 = cv2.inRange(image1, rgbLow, rgbHigh)
RGBMask1 = cv2.morphologyEx(RGBMask1, cv2.MORPH_CLOSE, kernel)

RGBMask2 = cv2.inRange(image2, rgbLow, rgbHigh)
RGBMask2 = cv2.morphologyEx(RGBMask2, cv2.MORPH_CLOSE, kernel)

RGBMask3 = cv2.inRange(image3, rgbLow, rgbHigh)
RGBMask3 = cv2.morphologyEx(RGBMask3, cv2.MORPH_CLOSE, kernel)

cv2.imshow('RGBMask1', cv2.resize(RGBMask1, (300, 300)))
cv2.imshow('RGBMask2', cv2.resize(RGBMask2, (300, 300)))
cv2.imshow('RGBMask3', cv2.resize(RGBMask3, (300, 300)))

cv2.waitKey(0)
```

先把圖片resize, 把圖片轉成HSV設定顏色範圍把範圍內的顏色設1否則0， 再用closeing把空洞填好。

心得

光源及亮度對取得皮膚的成效有很大的關係，同種背景顏色是否單一也很重要。發現open的RGB TO HSV的顏色與自己轉換的不一樣是因為opencv有設定一個變數去調整顏色。