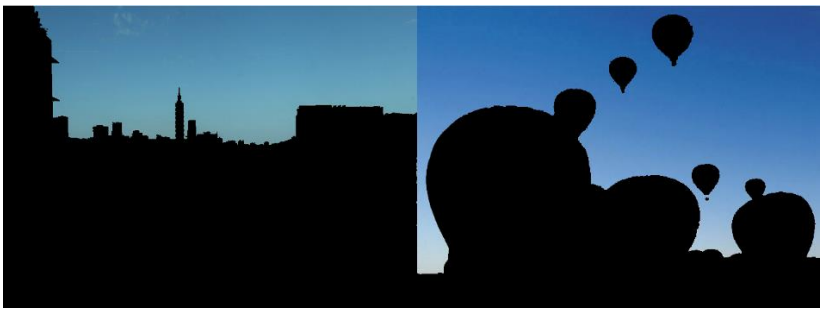
彩色影像處理

作業 2：影像色彩分割(題目 1)

M10915045 施信宏

先對作業附上的兩張圖做處理，將原圖與 mask 做運算，將 mask 為白的部分保留，即可得到藍天的區塊。



BGR

HSV

Y_Cr_Cb

```
get_sky = np.zeros(img_sky.shape,dtype = np.uint8) #宣告全為0的array ,將藍天的值放入
for channel in range(3):
    sky_pixel_row , sky_pixel_col = np.where(img_sky_mask[:,:,channel] > 50) #對 BGR channel 分別執行 得到哪些值大於50 (表不為黑
    get_sky[sky_pixel_row , sky_pixel_col] = img_sky[sky_pixel_row , sky_pixel_col] #將藍天pixel的值 對入get_sky對應的位置
```

取得藍天部分程式碼

取得各個色彩空間的藍天部分後，即可算空間中藍天部分的標準差及平均。

```
# --------BGR---------
B = get_sky[:,:,0]
G = get_sky[:,:,1]
R = get_sky[:,:,2]
print('B,G,R mean {} , {} , {} ' .format(np.mean( B[B> 50] ) ,np.mean( G[G > 50] ) ,np.mean( R[R > 50] )))
print('B,G,R std {} , {} , {} ' .format(np.std( B[B> 50] ) ,np.std( G[G > 50] ) ,np.std( R[R > 50] )))
# --------HSV---------
H = sky_HSV[:,:,0]
S = sky_HSV[:,:,1]
V = sky_HSV[:,:,2]
print('H,S,V mean {} , {} , {} ' .format(np.mean( H[H > 50] ) ,np.mean( S[S > 50] ) ,np.mean( V[V > 50] )))
print('H,S,V std {} , {} , {} ' .format(np.std( H[H > 50] ) ,np.std( S[S > 50] ) ,np.std( V[V > 50] )))
# --------YCrCb---------
Y  = sky_YCC[:,:,0]
Cr = sky_YCC[:,:,1]
Cb = sky_YCC[:,:,2]
print('Y,Cr,Cb mean {} , {} , {} ' .format(np.mean( Y[Y > 50] ) ,np.mean( Cr[Cr > 50] ) ,np.mean( Cb[Cb > 50] )))
print('Y,Cr,Cb std {} , {} , {} ' .format(np.std( Y[Y > 50] ) ,np.std( Cr[Cr > 50] ) ,np.std( Cb[Cb > 50] )))
```

先取得各個通道的陣列，再使用 numpy 中的函式得到 mean 及 std。值如下:

```
B,G,R mean 180.44783273001678 , 132.79401315715742 , 89.37470697781816
B,G,R std 15.36466362588507 , 25.9564454067158 , 26.03894079781026
H,S,V mean 103.21266594676999 , 149.8211188837703 , 180.4580087938525
H,S,V std 3.596399829609684 , 35.109047660892315 , 15.36574122711604
Y,Cr,Cb mean 121.43319249720756 , 114.28103780864197 , 142.3455552340535
Y,Cr,Cb std 25.56408069618912 , 16.173274108828142 , 17.82095439615937
```

有了各通道的平均值及標準差後，便可對其他不同的圖片做藍天的提取。
但圖片也並不是藍天的圖皆能有效的辨識出，與原影像的通道值有很大關聯，若是明亮藍或暗色藍，效果便會顯差，故後續找了兩張藍天相似於原圖的圖片做提取藍天的步驟。

Img1:

Img2:

```python
def get_sky_region(img , mean , std):
    binary_img = np.zeros(img.shape , dtype = np.uint8) #二值化的圖
    res_img = np.zeros(img.shape , dtype = np.uint8) #藍天的圖
    colorSpace_upper = mean + 2 * std # mean加上兩個std
    colorSpace_lower = mean - 2 * std # mean減掉兩個std
    rule_pass = np.zeros(img.shape , dtype = bool) #最後用此標籤判定是否為藍天 ， 三個channel都需要介於upper lower分別對應的值
    for channel in range(3):
        sky_row , sky_col = np.where((img[:,:,channel] >= colorSpace_lower[channel]) & (img[:,:,channel] <= colorSpace_upper[channel]))
        # 找到pixel值有介於upper lower之間的位置 ， 分別對每個channel運算
        rule_pass[sky_row , sky_col ,channel] = True
    index = np.sum(rule_pass , axis = 2)
    poix, poiy = np.where(index == 3) # 要三個channel都符合在範圍內才算是藍天
    res_img[poix, poiy] = img[poix, poiy] # 得到該圖的值
    binary_img[poix, poiy] = 255
    return res_img , binary_img
```

對新圖片取得藍天部分的 function。

當中 binary_img 為二值化影像，result_img 為擷取藍天的結果；，三個通道都必須介於傳入的色彩空間的平均加上正負兩個標準差內，才會認定是藍天。最後將有通過門檻的位置保留，其他將設為 0，如此便可得到只保留藍天部分及二值化後的圖。

```python
img1_BGR ,img1_BGR_mask = get_sky_region(img1, BGR_mean , BGR_std)
img1_HSV ,img1_HSV_mask = get_sky_region(cv2.cvtColor(img1, cv2.COLOR_BGR2HSV) , HSV_mean , HSV_std)
img1_YCC ,img1_YCC_mask = get_sky_region(cv2.cvtColor(img1, cv2.COLOR_BGR2YCR_CB) , YCrCb_mean , YCrCb_std)
#img2
img2_BGR ,img2_BGR_mask = get_sky_region(img2, BGR_mean , BGR_std)
img2_HSV ,img2_HSV_mask = get_sky_region(cv2.cvtColor(img2, cv2.COLOR_BGR2HSV) , HSV_mean , HSV_std)
img2_YCC ,img2_YCC_mask = get_sky_region(cv2.cvtColor(img2, cv2.COLOR_BGR2YCR_CB) , YCrCb_mean , YCrCb_std)
```

分別對 img1，img2 的不同色彩空間做運算

Img1:

BGR:

HSV:

Y_Cr_Cb:

Img2:

BGR:

HSV:

Y_Cr_Cb:

整體的感覺應該是 HSV 的處理最好，在三空間內表現最佳，許多小雜訊也能有效過濾掉。也難怪許多人做影像色彩轉換，取得背景或前景時，都先將原圖投到 HSV，再去做 threshold 等等處理。

附上程式碼為.py 檔，若需要看程式請以記事本開啟，貼到 word 會使版面很亂，造成閱讀不便，謝謝。