Hw3_Color Image Enhancement

學號:610415145 姓名:許晉偉

Date due:2022/12/23

Date handed in:2022/12/23

■Technical description

```
import cv2
import numpy as np
img = cv2.imread('aloe.jpg')
#img = cv2.imread('church.jpg')
#img = cv2.imread('kitchen.jpg')
#img = cv2.imread('house.jpg')
```

使用 opencv 函式庫進行基本圖片操作

使用 numpy 函式庫進行矩陣建立

```
H,W,CH = img.shape[:]
lap = np.zeros( (H, W, CH), np.float64)
kernel = np.array([[-1,-1,-1],
                   [-1, 9,-1],
                   [-1,-1,-1]])
```

建立存放結果所需矩陣

建立 kernel 用於 convolution

(1)RGB

```
for i in range(H-2):
    for j in range(W-2):
        for k in range(CH):
            lap[i, j, k] = (np.sum(kernel * img[i:i+3, j:j+3, k])
            if(lap[i, j, k] < 0):lap[i, j, k] = 0
            if(lap[i, j, k] >= 255):lap[i, j, k] = 255

cv2.imshow('input', img)
cv2.imshow('laplacian', lap.astype(np.uint8))
cv2.imwrite('RGB_alon.jpg', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

對 3 個通道進行 convolution 並進行 Relu

接者顯示結果並儲存結果

(2)HSI

```
import cv2
import numpy as np
#img = cv2.imread('aloe.jpg')
#img = cv2.imread('church.jpg')
#img = cv2.imread('kitchen.jpg')
img = cv2.imread('house.jpg')
H,W,C = img.shape[:]
normal_img = np.zeros( (H, W, C), np.float64 )
hsi_img = np.zeros( (H, W, C), np.float64 )
laphsi_img = np.zeros( (H, W, C), np.float64 )
hsv_img = np.zeros( (H, W, C), np.uint8 )
rgb_img = np.zeros( (H, W, C), np.float64 )

kernel= np.array([[-1.0,-1.0,-1.0],
                  [-1.0, 9.0,-1.0],
                  [-1.0,-1.0,-1.0]])
```

創立所需陣列以及卷積核

```
normal_img[:,:,:] = img[:,:,:]/255
```

對原圖做正規化

```
for i in range(H):
    for j in range(W):
        sqrt = math.sqrt((normal_img[i,j,2] - normal_img[i,j,1])**2
                         +(normal_img[i,j,2] - normal_img[i,j,0])
                         *(normal_img[i,j,1] - normal_img[i,j,0]))
        if(sqrt != 0):
            theta = np.arccos(0.5*(normal_img[i,j,2] - normal_img[i,j,1]+ normal_img[i,j,2] - normal_img[i,j,0])/sqrt)
            if(normal_img[i, j, 0] <= normal_img[i, j, 1]):
                hsi_img[i,j,0] = theta
            elif(normal_img[i, j, 0] > normal_img[i, j, 1]):
                hsi_img[i,j,0] = 2*np.pi - theta
        else: hsi_img[i,j,0] = 0
        hsi_img[i,j,0] = hsi_img[i,j,0]/(2*np.pi)
```

利用講義公式求出 H 和角度

$$H = \begin{cases} \theta & \text{if } B \le G \\ 360 - \theta \end{cases}$$

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}} \right\},$$

```
if((normal_img[i, j, 0] + normal_img[i, j, 1] + normal_img[i, j, 2]) == 0):
    hsi_img[i, j, 1] = 0
else:
    hsi_img[i, j, 1] = 1 - (normal_img[i, j, :].min() * 3 / (normal_img[i, j, 0] + normal_img[i, j, 1] + normal_img[i, j,
```

接者求出 S

$$S = 1 - \frac{3}{(R+G+B)}\left[\min(R,G,B)\right],$$

```
#I
hsi_img[:,:,2] = (normal_img[:,:,0] + normal_img[:,:,1] + normal_img[:,:,2])/3

#non-ormalization
hsi_img[:,:,:] = hsi_img[:,:,:]*255
```

最後求出 I 並解正規化

```
for i in range(H-2):
    for j in range(W-2):
        for k in range(C-1):
            laphsi_img[i+1, j+1, k+1] = np.sum(kernel * hsi_img[i:i+3, j:j+3, k+1])
            if(laphsi_img[i+1, j+1, k+1] < 0):laphsi_img[i+1, j+1, k+1] = 0
            if(laphsi_img[i+1, j+1, k+1] >= 255):laphsi_img[i+1, j+1, k+1] = 255
```

在 HIS 空間進行 convolution 並進行 Relu

```
normal_img[:,:,:] = laphsi_img[:,:,:]/255
```

接者一樣進行正規化

```
for i in range(H):
    for j in range(W):
        if(normal_img[i,j,0]*2*math.pi >= 0 and normal_img[i,j,0]*2*math.pi < 2*math.pi/3):
            #B
            rgb_img[i,j,0] = normal_img[i,j,2] * (1 - normal_img[i,j,1])
            #R
            rgb_img[i,j,2] = normal_img[i,j,2] * (1 + (normal_img[i,j,1]*np.cos(normal_img[i,j,0]*2*math.pi)/np.cos(math.pi/3 - r
            #G
            rgb_img[i,j,1] = 3 * normal_img[i,j,2] - (rgb_img[i,j,2] + rgb_img[i,j,0])
```

針對不同角度套用不同的公式

$RG\ sector\ (0^\circ \le H < 120^\circ)$:

$$B = I(1-S)$$

$$R = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$

$$G = 3I - (R+B).$$

```
#HSI transform to RGB space
for i in range(H):
    for j in range(W):
        if(normal_img[i,j,0]*2*math.pi >= 0 and normal_img[i,j,0]*2*math.pi < 2*math.pi/3):
            #B
            rgb_img[i,j,0] = normal_img[i,j,2] * (1 - normal_img[i,j,1])
            #R
            rgb_img[i,j,2] = normal_img[i,j,2] * (1 + (normal_img[i,j,1]*np.cos(normal_img[i,j,0]*2*math.pi)/np.cos(math.pi/3 - r
            #G
            rgb_img[i,j,1] = 3 * normal_img[i,j,2] - (rgb_img[i,j,2] + rgb_img[i,j,0])

        if(normal_img[i,j,0]*2*math.pi >= 2*math.pi/3 and normal_img[i,j,0]*2*math.pi < 4*math.pi/3):
            #R
            rgb_img[i,j,2] = normal_img[i,j,2] * (1 - normal_img[i,j,1])
            #G
            rgb_img[i,j,1] = normal_img[i,j,2] * (1 + (normal_img[i,j,1]*math.cos(normal_img[i,j,0]*2*math.pi-2*math.pi/3)/math.c
            #B
            rgb_img[i,j,0] = 3 * normal_img[i,j,2] - (rgb_img[i,j,2] + rgb_img[i,j,1])

        if(normal_img[i,j,0]*2*math.pi >= 4*math.pi/3 and normal_img[i,j,0]*2*math.pi < 6*math.pi/3):
            #G
            rgb_img[i,j,1] = normal_img[i,j,2] * (1 - normal_img[i,j,1])
            #B
            rgb_img[i,j,0] = normal_img[i,j,2] * (1 + (normal_img[i,j,1]*math.cos(normal_img[i,j,0]*2*math.pi-4*math.pi/3)/math.c
            #R
            rgb_img[i,j,2] = 3 * normal_img[i,j,2] - (rgb_img[i,j,1] + rgb_img[i,j,0])

#non-ormalization
rgb_img[:,:,:] = rgb_img[:,:,:]*255
rgb_img = np.maximum(rgb_img, 0)
rgb_img = np.minimum(rgb_img, 255)
```

針對不同角度套用不同的公式

$GB\ sector\ (120^\circ \le H < 240^\circ)$:

$$H = H - 120^\circ.$$

Then the RGB components are:

$$R = I(1 - S)$$
$$G = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$
$$B = 3I - (R + G).$$

$BR\ sector\ (240^\circ \le H \le 360^\circ)$:

$$H = H - 240^\circ.$$

Then the RGB components are:

$$G = I(1 - S)$$
$$B = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$
$$R = 3I - (G + B).$$

最後解正規化得到結果

```
cv2.imshow('source', img)
cv2.imshow('HSI', hsi_img.astype(np.uint8))
cv2.imshow('LAP_HSI', laphsi_img.astype(np.uint8))
cv2.imshow('LIB_HSV', hsv_img)
cv2.imshow('RGB', rgb_img.astype(np.uint8))
cv2.imwrite('HSI_alon.jpg', rgb_img.astype(np.uint8))

cv2.waitKey(0)
cv2.destroyAllWindows()
```

顯示出結果並儲存結果

(2)Lab

```
rgb2xyz = np.array([[0.412453, 0.357580, 0.180423],
                    [0.212671, 0.715160, 0.072169],
                    [0.019334, 0.119193, 0.950227]])

xyz2rgb = np.array([[ 3.240479,-1.537150,-0.498535],
                    [-0.969256, 1.875992, 0.041556],
                    [ 0.055648,-0.204043, 1.057311]])

kernel= np.array([[-1.0,-1.0,-1.0],
                  [-1.0, 9.0,-1.0],
                  [-1.0,-1.0,-1.0]])
```

依照正向變換反向變換設定矩陣

正向變換

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

反向變換

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

```
#normalization
normal_img[:,:,:] = img[:,:,:]/255

#RGB transform to XYZ space
for i in range(H):
    for j in range(W):
        xyz_img[i, j, :] = np.dot(rgb2xyz, normal_img[i,j,:])
```

正規化後按照正向變化內積得到 XYZ 座標

```
#XYZ transform to L*A*B space
for i in range(H):
    for j in range(W):
        buffer[:] = xyz_img[i, j, :]/xyz_n[:]

        for k in range(3):
            if(buffer[k] > 0.008856):
                f_xyz[k] = math.pow(buffer[k], 1/3)
            else:
                f_xyz[k] = 7.787 * buffer[k] +16/116
        #L*
        if(buffer[1] > 0.008856):
            lab_img[i, j, 0] = 116 * math.pow(buffer[1], 1/3) - 16
        else:
            lab_img[i, j, 0] = 903.3 * buffer[1]

        #A*
        lab_img[i, j, 1] = 500 * (f_xyz[0] - f_xyz[1])

        #B*
        lab_img[i, j, 2] = 200 * (f_xyz[1] - f_xyz[2])
```

接著使用右側的公式

進行 XYZ 到 L*a*b 的轉換

$$L^* = 116 \bullet h\left(\frac{Y}{Y_W}\right) - 16$$

$$a^* = 500\left[h\left(\frac{X}{X_W}\right) - h\left(\frac{Y}{Y_W}\right)\right]$$

$$b^* = 200\left[h\left(\frac{Y}{Y_W}\right) - h\left(\frac{Z}{Z_W}\right)\right]$$

$$where\ X_n = 0.9515$$
$$Y_n = 1.0000$$
$$Z_n = 1.0886$$

$$f(t) = \begin{cases} t^{\frac{1}{3}}, & t > 0.008856 \\ 7.787 \times t + \frac{16}{116}, & otherwise \end{cases}$$

```
for i in range(H-2):
    for j in range(W-2):
        laplab_img[i+1, j+1, 0] = np.sum(kernel * lab_img[i:i+3, j:j+3, 0])
        if(laplab_img[i+1, j+1, 0] < 0):laplab_img[i+1, j+1, 0] = 0
        if(laplab_img[i+1, j+1, 0] >= 255):laplab_img[i+1, j+1, 0] = 255
```

在 L*a*b 空間中 進行 convolution 計算

```
for i in range(H):
    for j in range(W):
        f_xyz[1] = (laplab_img[i, j, 0] + 16) / 116
        f_xyz[0] =  f_xyz[1] + laplab_img[i, j, 1]/500
        f_xyz[2] =  f_xyz[1] - laplab_img[i, j, 2]/200
        #X
        if(f_xyz[0] > 0.008856):
            xyz_img[i, j, 0] = xyz_n[0] * (f_xyz[0]**3)
        else:
            xyz_img[i, j, 0] = ((f_xyz[0]-16) / 116) * 3 * (0.008865**2) * xyz_n[0]
        #Y
        if(f_xyz[1] > 0.008856):
            xyz_img[i, j, 1] = xyz_n[1] * (f_xyz[1]**3)
        else:
            xyz_img[i, j, 1] = ((f_xyz[1]-16) / 116) * 3 * (0.008865**2) * xyz_n[1]
        #Z
        if(f_xyz[2] > 0.008856):
            xyz_img[i, j, 2] = xyz_n[2] * (f_xyz[2]**3)
        else:
            xyz_img[i, j, 2] = ((f_xyz[2]-16) / 116) * 3 * (0.008865**2) * xyz_n[2]
```

接著使用下方的公式進行反向轉換從 L*a*b 轉換回到 XYZ 空間

$$f_y = \frac{L^* + 16}{116}$$

$$f_x = f_y + \frac{a^*}{500}$$

$$f_z = f_y - \frac{b^*}{200}$$

$$if\ f_y > 0.008856\ then\ Y = Y_n \times {f_y}^3$$

$$\text{else Y} = \left(\frac{f_y - 16}{116}\right) \times 3 \times 0.008865^2 \times Y_n$$

$$if\ f_x > 0.008856\ then\ X = X_n \times {f_x}^2$$

$$\text{else X} = \left(\frac{f_x - 16}{116}\right) \times 3 \times 0.008865^2 \times X_n$$

$$if\ f_z > 0.008856\ then\ Z = Z_n \times {f_z}^2$$

$$\text{else Z} = \left(\frac{f_z - 16}{116}\right) \times 3 \times 0.008865^2 \times Z_n$$

```
for i in range(H):
    for j in range(W):
        rgb_img[i, j, :] = np.dot(xyz2rgb, xyz_img[i,j,:])
```

按照前面設定好的反轉換的矩陣進行內積從 XYZ 轉換回 RGB

```
#non-ormalization
rgb_img[:,:,:] = rgb_img[:,:,:] * 255
rgb_img = np.maximum(rgb_img, 0)
rgb_img = np.minimum(rgb_img, 255)


cv2.imshow('source', img)
cv2.imshow('LAB', lab_img.astype(np.uint8))
cv2.imshow('LAP_LAB', laplab_img.astype(np.uint8))
cv2.imshow('lib_LAB', liblab_img)
cv2.imshow('RGB', rgb_img.astype(np.uint8))
cv2.imwrite('LAB_alon.jpg', rgb_img.astype(np.uint8))
```

最後解正規化得到結果並儲存起來

Experimental results

aloe

原圖

RGB

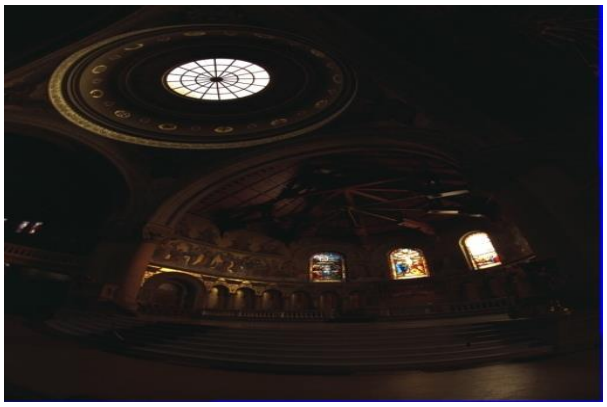
HSI

Lab

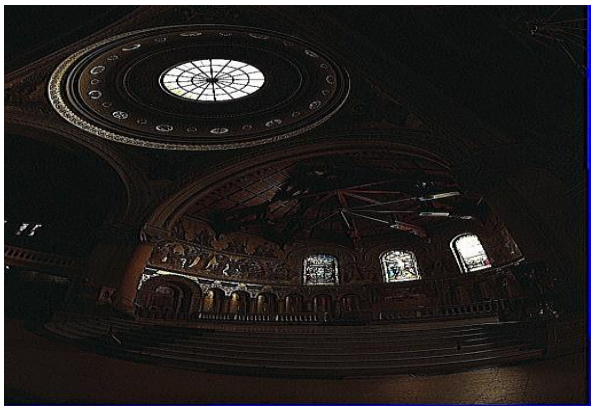
church

原圖

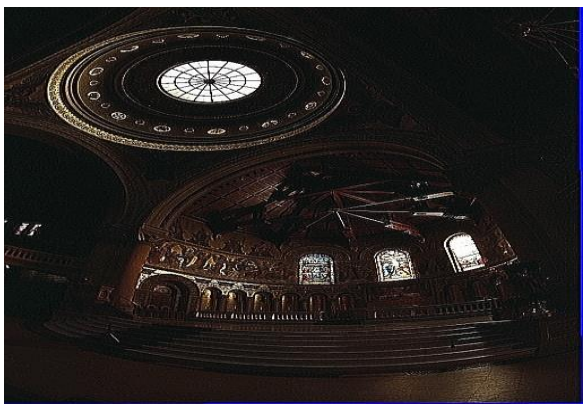
RGB



HSI



Lab



kitchen

原圖

RGB

HSI

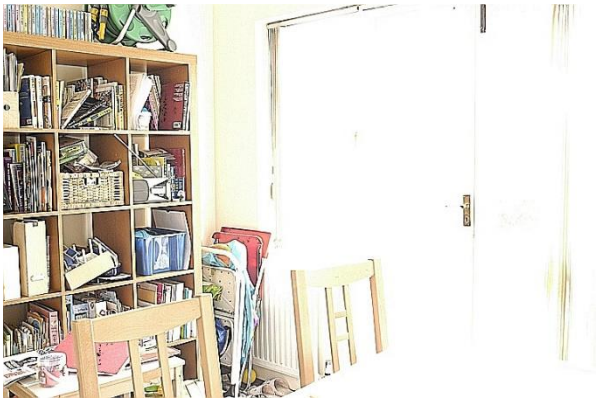
Lab

house

原圖

RGB

HSI



Lab

■Discussions

RGB

在 rgb 空間進行拉普拉斯銳化並沒有很困難結果也跟之前做過得差不多

HSI

色調 H(Hue)：與光波的波長有關，它表示人的感官對不同顏色的感受，如紅色、綠色、藍色等，它也可表示一定範圍的顏色，如暖色、冷色等。

飽和度 S(Saturation)：表示顏色的純度，純光譜色是完全飽和的，加入白光會稀釋飽和度。飽和度越大，顏色看起來就會越鮮艷，反之亦然。簡而言之，飽和度就是顏色鮮明與否的指標。

強度 I(Intensity)：對應成像亮度和圖像灰度，是顏色的明亮程度。

由於 HSI 決定顏色的機制與 RGB 不同，所以在對 HSI 做完影像增強再轉回

RGB 會造成三原色的誤差，使得結果比較不如預期，但在 HSI 空間下的影像增強 確實是有效的,可以透過調整 HSI 色彩空間的參數去避免因光源強弱不同而造成顏色辨識不準確的情況。

Lab

特色：L*a*b*色彩空間是顏色-對立 (color-component)空間，帶有維度 L 表示亮度，a 和 b 表示顏色對立維度，基於了非線性壓縮的 CIE XYZ 色彩空間坐標。

Lab 顏色被設計來接近人類視覺,在結果上我也覺得它比其他種轉換效果好

但是不同於其他顏色空間可以直接和 RGB 轉換 Lab 必須透過 xyz 空間轉換所以步驟更繁瑣

■ References and Appendix

https://www.jianshu.com/p/8ff4872e088f?fbclid=IwAR2qCB2Cw-WS-K5-cle2utrRVtU-Cs0BF0S2LhPdefBR773eKSLPE6Aueas

https://www.796t.com/content/1544974982.html

http://a1239537.pixnet.net/blog/post/342082176-hsi%E8%89%B2%E5%BD%A9%E7%A9%BA%E9%96%93%E6%87%89%E7%94%A8%E7%AD%86%E8%A8%98%281%29

https://cg2010studio.com/2012/10/02/rgb%e8%88%87cielab%e8%89%b2%e5%bd%a9%e7%a9%ba%e9%96%93%e8%bd%89%e6%8f%9b/