Color Image Enhancement

電機所　609415159　江昀融

Date due:2020/12/18

Date handed in:2020/12/18

Technical description:

使用 opencv 用於圖片的基本讀寫與顯示
numpy 用於進行陣列操作
math 用於影像處理的公式計算
matplotlib 用於圖表的顯示

```
import cv2 as cv
import numpy as np
import math
from matplotlib import pyplot as plt
```

cv.imread 讀取輸入圖片
並且讀取圖片長寬並顯示

```
#choose read which image
img = cv.imread('aloe.jpg', cv.IMREAD_COLOR)
#img = cv.imread('church.jpg', cv.IMREAD_COLOR)
#img = cv.imread('kitchen.jpg', cv.IMREAD_COLOR)
#img = cv.imread('house.jpg', cv.IMREAD_COLOR)

#print height and width of input image
height, width, channel = img.shape
print(type(img.shape))
print(img.shape)
print("height = ", height)
print("width = ", width)
print("channel = ", channel)
print("*************")
```

選擇 laplacian kernel 進行影像增強
並創建 laplacian kernel

```
#creat new image for output & laplacian kernel
lap_s = np.zeros( (height-2, width-2, channel), np.float )
l_kernel = np.array([[ 0.0,-1.0, 0.0],
                     [-1.0, 5.0,-1.0],
                     [ 0.0,-1.0, 0.0]])
```

並且針對三種不同的影像處理技術進行分別的動作:

1.RGB

對原圖使用 laplacian kernel 進行 convolution 計算

```
for i in range(height-2):
    for j in range(width-2):
        for k in range(channel):
            lap_s[i, j, k] = np.sum(l_kernel * img[i:i+3, j:j+3, k])
            if(lap_s[i, j, k] < 0):lap_s[i, j, k] = 0
            if(lap_s[i, j, k] >= 255):lap_s[i, j, k] = 255
```

整理圖表並顯示

```
cv.imshow('source', img)
cv.imshow('laplacian_sharpening(float)', lap_s)
```

2.HSI

先對原圖做正規化

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

```
#RGB transform to HSI space
#H
for i in range(height):
    for j in range(width):
        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]))
        theta = math.acos(0.5*(normal_img[i,j,2] - normal_img[i,j,1]
                        + normal_img[i,j,2] - normal_img[i,j,0])
                    /sqrt
                    )
        if(sqrt == 0):
            hsi_img[i,j,0] = 0
        elif(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
        hsi_img[i,j,0] = hsi_img[i,j,0]/(2*np.pi)

#S
        buffer[:] = normal_img[i, j, :]
        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 - (buffer.min() * 3 / (normal_img[i, j, 0] + normal_img[i, j, 1] + normal_img[i, j, 2]) )

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

接著進行 RGB 到 HSI 的轉換

使用右側公式分別對 H,S,I 進行個別的計算

並將結果解正規化

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

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

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

$$I = \frac{1}{3}(R+G+B),$$

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

對 HSI 空間使用 laplacian kernel 進行 convolution 計算

```
for i in range(height-2):
    for j in range(width-2):
        for k in range(channel):
            laphsi_img[i+1, j+1, k] = np.sum(l_kernel * hsi_img[i:i+3, j:j+3, k])
            if(laphsi_img[i+1, j+1, k] < 0):laphsi_img[i+1, j+1, k] = 0
            if(laphsi_img[i+1, j+1, k] >= 255):laphsi_img[i+1, j+1, k] = 255
```

並將 HSI 空間正規化

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

```
#HSI transform to RGB space
for i in range(height):
    for j in range(width):
        if((normal_img[i,j,0]*180/math.pi)>=0 and (normal_img[i,j,0]*180/math.pi)<120):
            #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]*math.cos(normal_img[i,j,0])/math.cos(60*math.pi/180 - normal_img[i,j,0])))
            #G
            rgb_img[i,j,1] = 3 * normal_img[i,j,2] - (rgb_img[i,j,2] + rgb_img[i,j,0])

        elif((hsi_img[i,j,0]*180/math.pi)>=120 and (hsi_img[i,j,0]*180/math.pi)<240):
            #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])/math.cos(180*math.pi/180 - normal_img[i,j,0])))
            #B
            rgb_img[i,j,0] = 3 * normal_img[i,j,2] - (rgb_img[i,j,2] + rgb_img[i,j,1])

        elif((hsi_img[i,j,0]*180/math.pi)>=240 and (hsi_img[i,j,0]*180/math.pi)<360):
            #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])/math.cos(300*math.pi/180 - normal_img[i,j,0])))
            #R
            rgb_img[i,j,2] = 3 * normal_img[i,j,2] - (rgb_img[i,j,1] + rgb_img[i,j,0])
```

RG sector ($0° \le H < 120°$):

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

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

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

BR sector ($240° \le H \le 360°$):

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

Then the RGB components are:

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

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

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

GB sector ($120° \le H < 240°$):

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

Then the RGB components are:

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

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

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

接著進行 HSI 到 RGB 的轉換
使用右側公式分別對 H,S,I 進行個別的計算
針對三種區間個別處理
並將結果解正規化

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

整理圖表並顯示

```
#show image
cv.imshow('source', img)
cv.imshow('normalized', normal_img)
cv.imshow('HSI', hsi_img.astype(np.uint8))
cv.imshow('LAP_HSI', laphsi_img.astype(np.uint8))
cv.imshow('HSV', hsv_img)
cv.imshow('RGB', rgb_img.astype(np.uint8))
cv.imshow('RGB2', rgb_img)
```

3.L*a*b

先建立轉換空間會用到的各種矩陣

```
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]])

l_kernel = np.array([[ 0.0,-1.0, 0.0],
                     [-1.0, 5.0,-1.0],
                     [ 0.0,-1.0, 0.0]])

xyz_n = np.array([0.9515, 1.0000, 1.0886])
buffer = np.zeros( 3, np.float )
f_xyz = np.zeros( 3, np.float )
```

先對原圖做正規化

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

進行 RGB 到 XYZ 的轉換
使用右側的公式

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

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

接著進行 XYZ 到 L*a*b 的轉換
使用右側的公式

$$L^* = \begin{cases} 116 \times \left(\frac{Y}{Yn}\right)^{\frac{1}{3}} - 16, & \frac{Y}{Yn} > 0.008856 \\ 903.3 \times \frac{Y}{Yn}, & \text{otherwise} \end{cases}$$

$$a^* = 500 \times \left(f\left(\frac{X}{Xn}\right) - f\left(\frac{Y}{Yn}\right) \right)$$

$$b^* = 200 \times \left(f\left(\frac{Y}{Yn}\right) - f\left(\frac{Z}{Zn}\right) \right)$$

where

$$Xn = 0.9515$$

$$Yn = 1.0000$$

$$Zn = 1.0886$$

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

```
#XYZ transform to L*A*B space
for i in range(height):
    for j in range(width):
        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])

liblab_img = cv.cvtColor(img, cv.COLOR_BGR2LAB)
```

對 L*a*b 空間使用 laplacian kernel 進行 convolution 計算

```
for i in range(height-2):
    for j in range(width-2):
        for k in range(channel):
            laplab_img[i+1, j+1, k] = np.sum(l_kernel * lab_img[i:i+3, j:j+3, k])
            if(laplab_img[i+1, j+1, k] < 0):laplab_img[i+1, j+1, k] = 0
            if(laplab_img[i+1, j+1, k] >= 255):laplab_img[i+1, j+1, k] = 255
```

接著進行 L*a*b 到 XYZ 的轉換
使用右側的公式

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

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

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

$$\text{if} \quad f_y > 0.008856 \quad \text{then} \quad Y = Yn \times f_y^3$$

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

$$\text{if} \quad f_x > 0.008856 \quad \text{then} \quad X = Xn \times f_x^3$$

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

$$\text{if} \quad f_z > 0.008856 \quad \text{then} \quad Z = Zn \times f_z^3$$

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

```
#L*A*B transform to XYZ space
for i in range(height):
    for j in range(width):
        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]
```

接著進行 XYZ 到 RGB 的轉換
使用右側的公式

```python
#XYZ transform to RGB space
for i in range(height):
    for j in range(width):
        rgb_img[i, j, :] = np.dot(xyz2rgb, xyz_img[i,j,:])
```

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

將結果解正規化

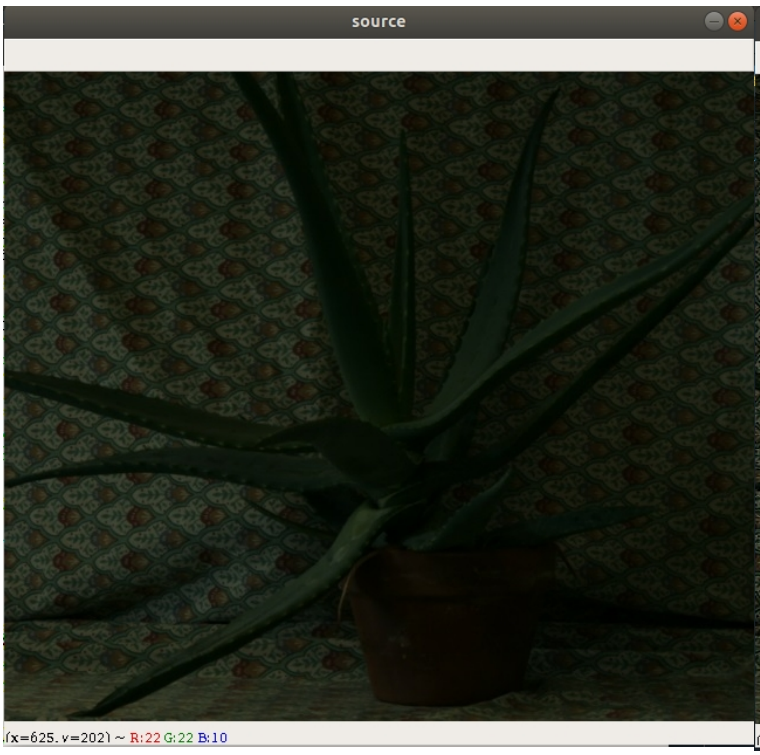
```python
#non-ormalization
rgb_img[:,:,:] = rgb_img[:,:,:] * 255
```

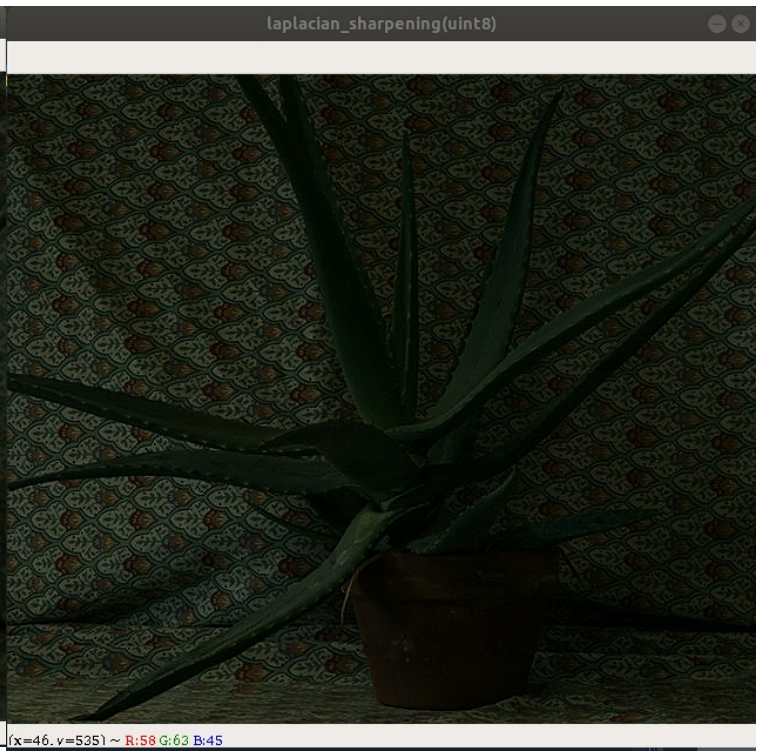
整理圖表並顯示

```python
#show image
cv.imshow('source', img)
cv.imshow('normalized', normal_img)
cv.imshow('CIE_XYZ', xyz_img)
cv.imshow('LAB', lab_img.astype(np.uint8))
cv.imshow('LAP_LAB', laplab_img.astype(np.uint8))
cv.imshow('lib_LAB', liblab_img)
cv.imshow('RGB', rgb_img.astype(np.uint8))
cv.imshow('RGB2', rgb_img)
```

Experimental results:

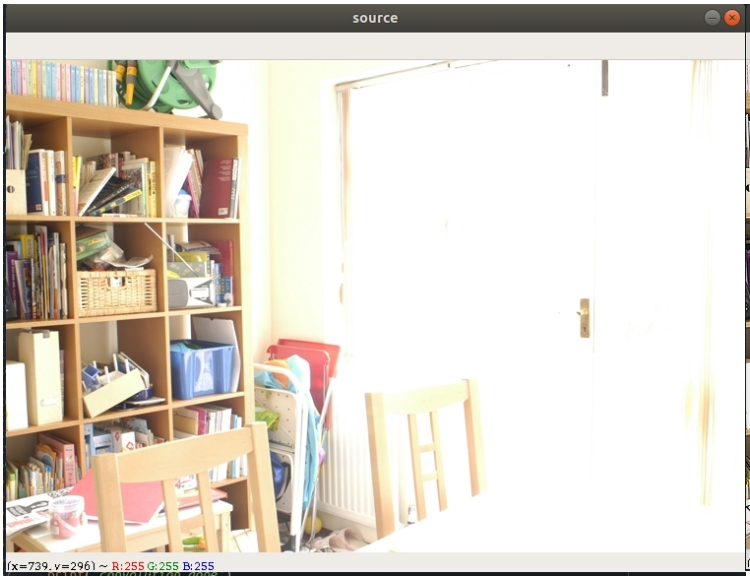
1.enhance images in the RGB color spaces.

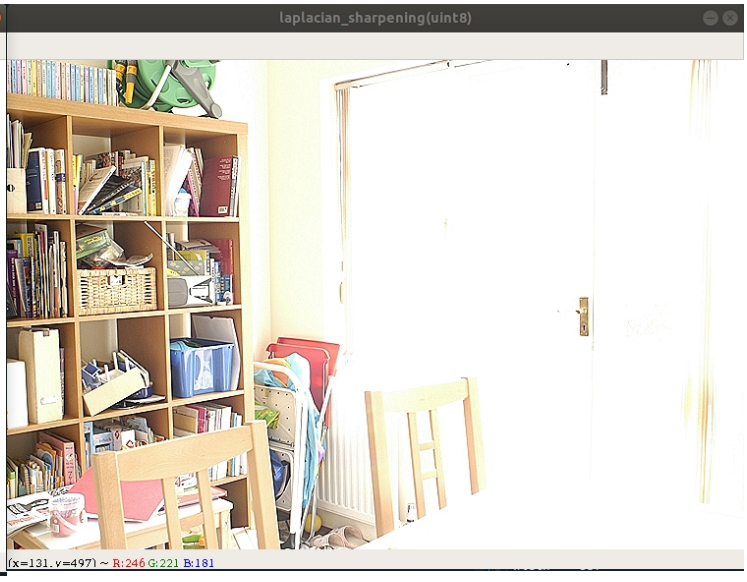


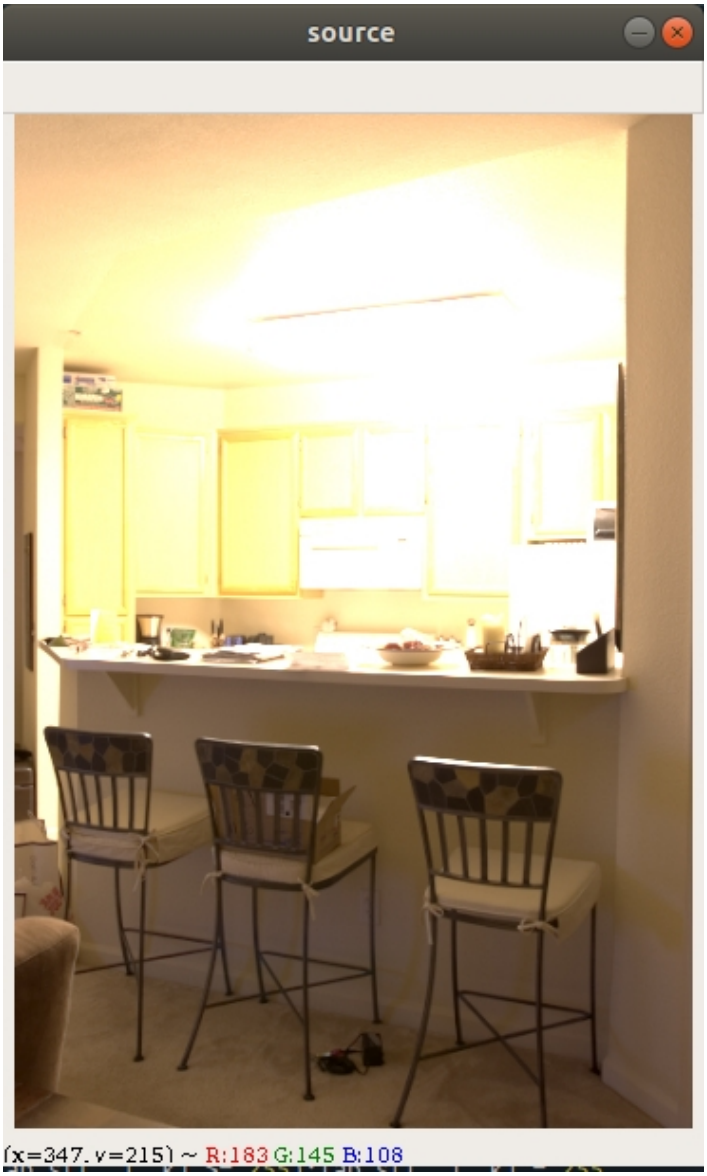


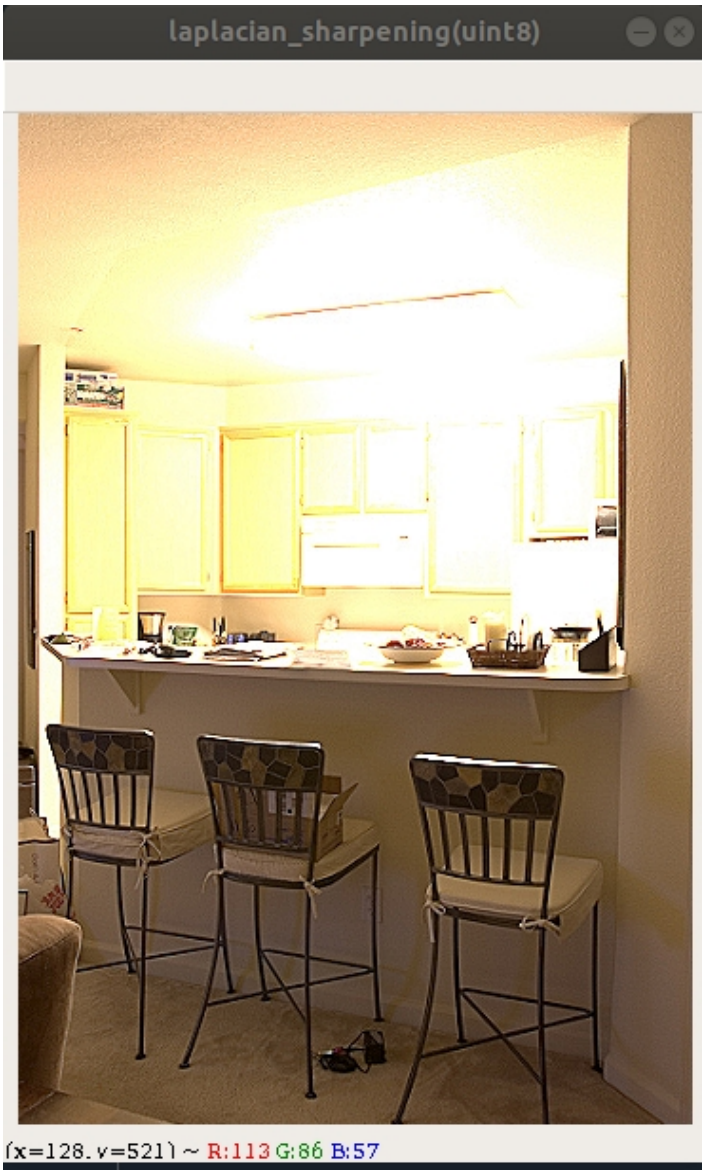

source
(x=494, y=63) ~ R:15 G:19 B:20


laplacian_sharpening(uint8)
(x=496, y=437) ~ R:20 G:21 B:19

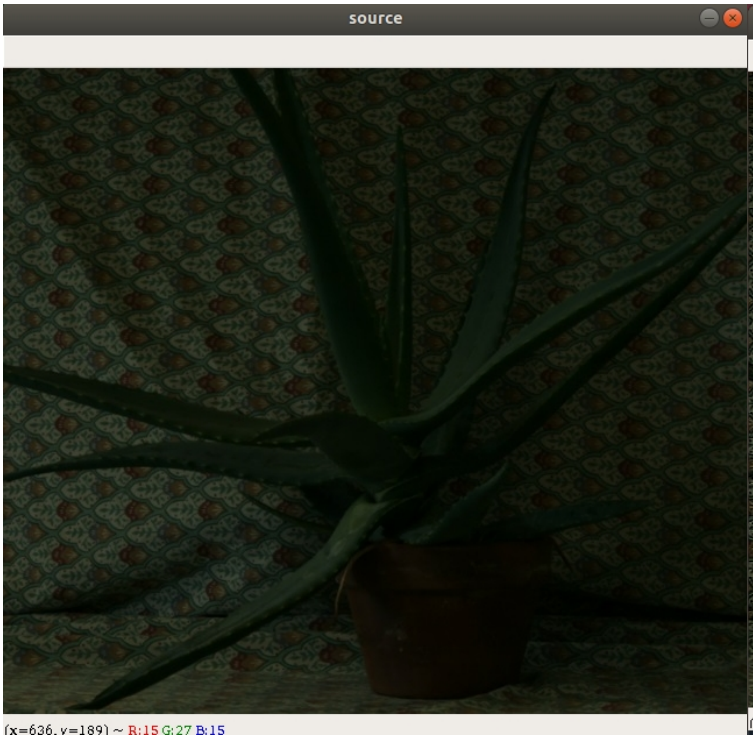

source
(x=739, y=296) ~ R:255 G:255 B:255

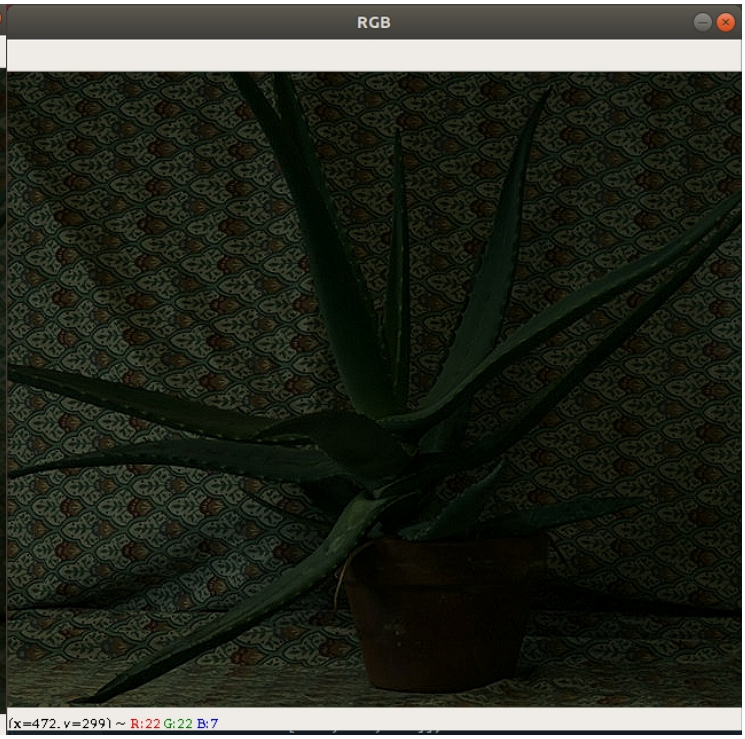

laplacian_sharpening(uint8)
(x=131, y=497) ~ R:246 G:221 B:181





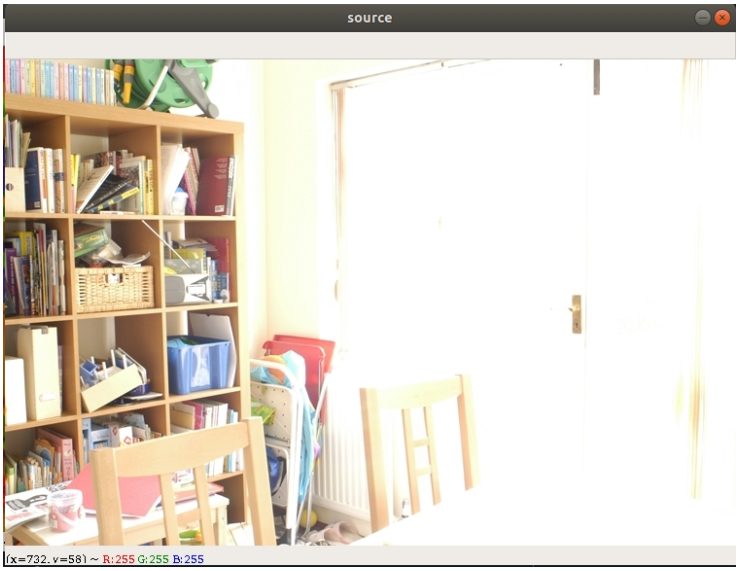
2.enhance images in the HSI color spaces.

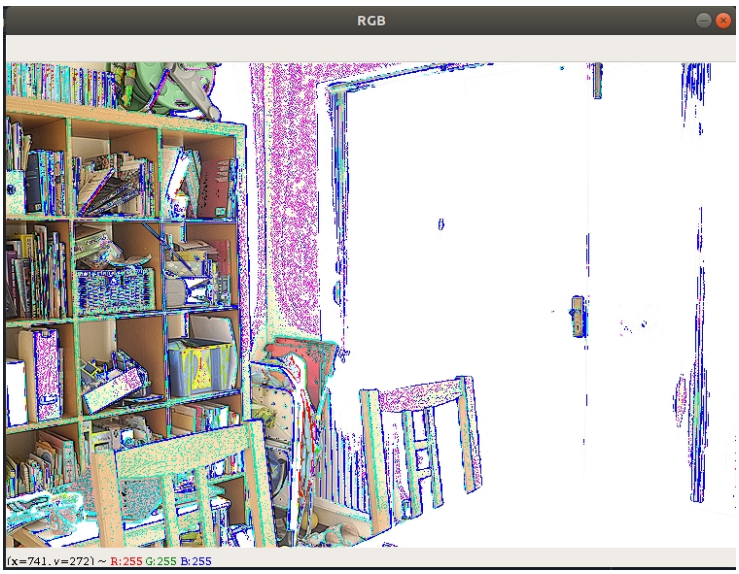


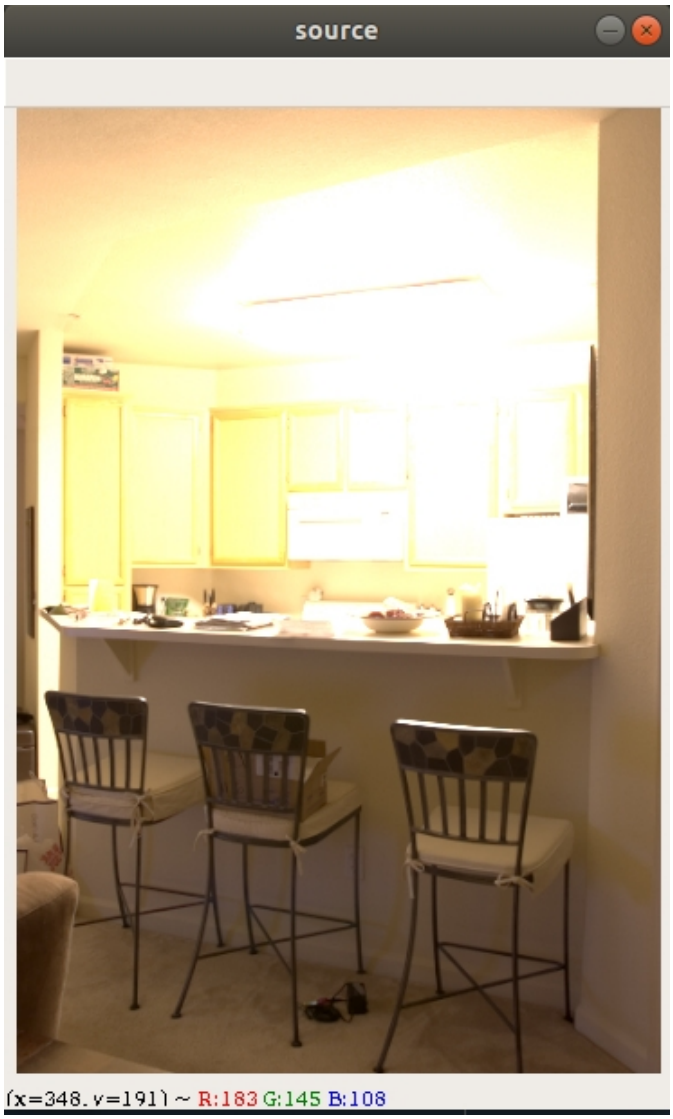


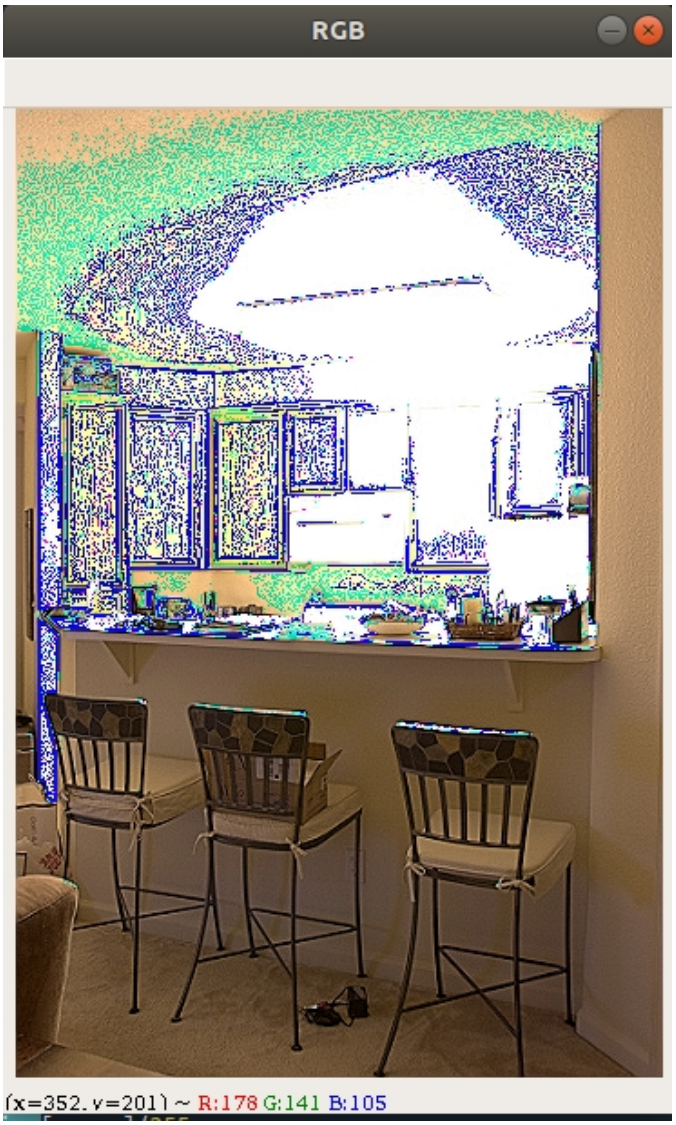

source
(x=500, y=461) ~ R:25 G:19 B:19


RGB
(x=496, y=113) ~ R:18 G:20 B:19

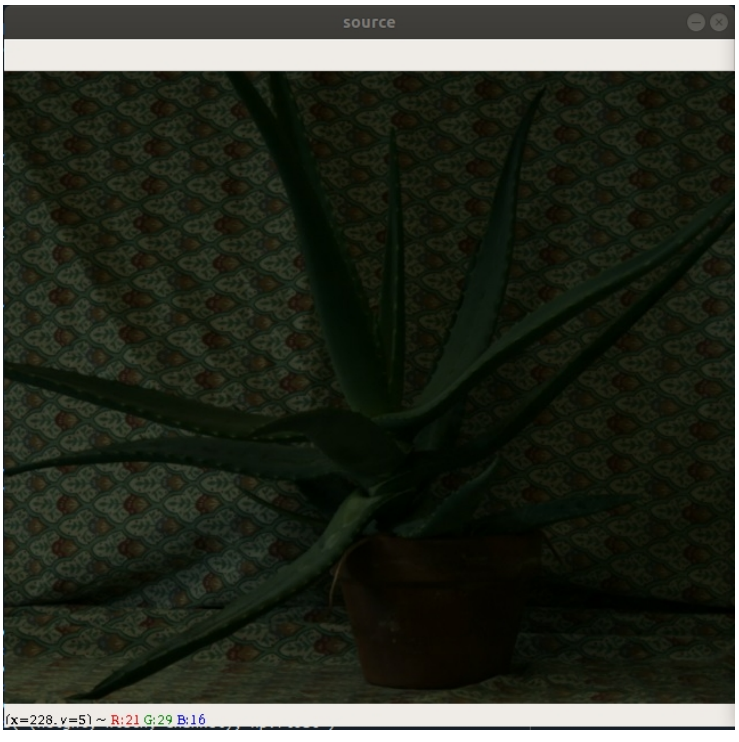

source
(x=732, y=58) ~ R:255 G:255 B:255

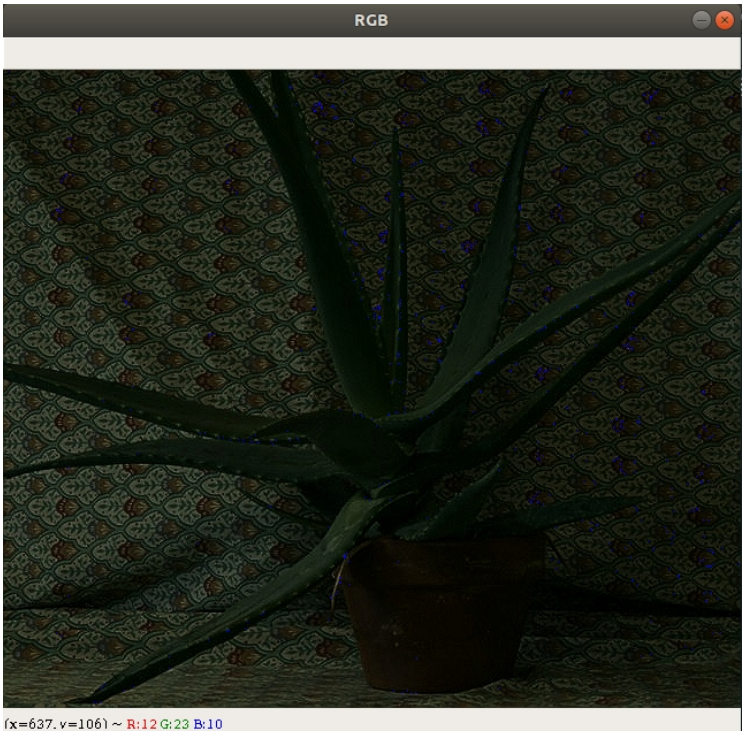

RGB
(x=741, y=272) ~ R:255 G:255 B:255





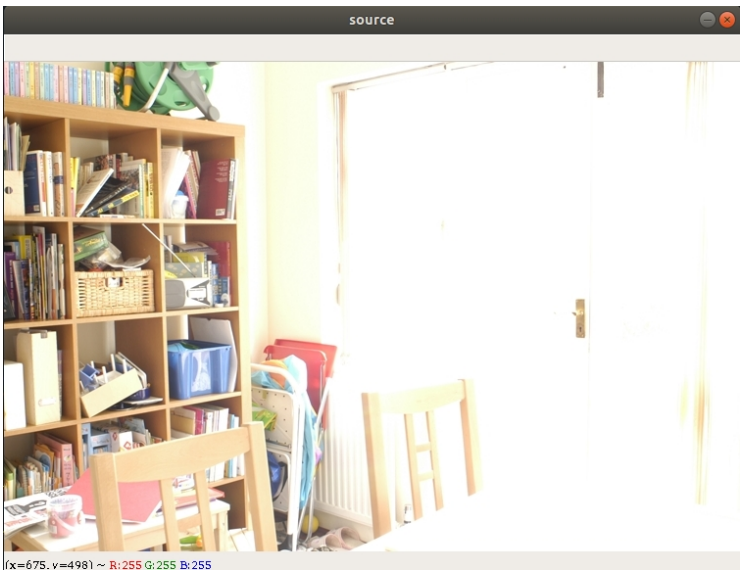
3.enhance images in the L*a*b* color spaces.

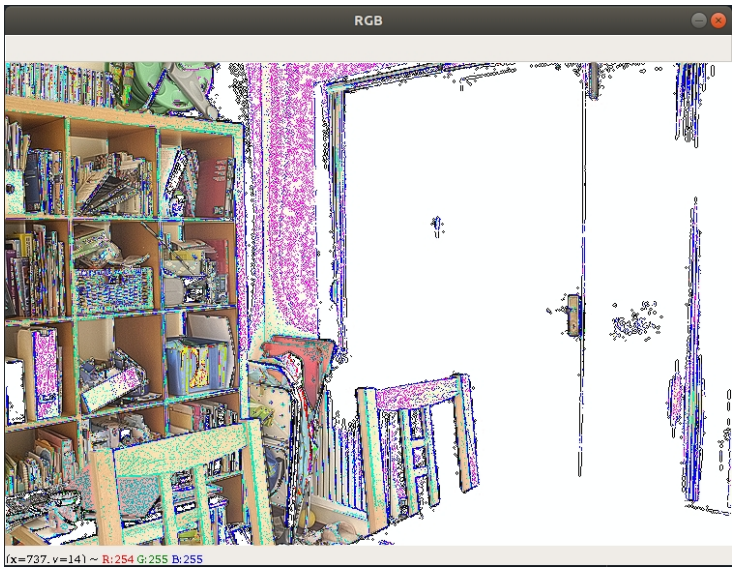


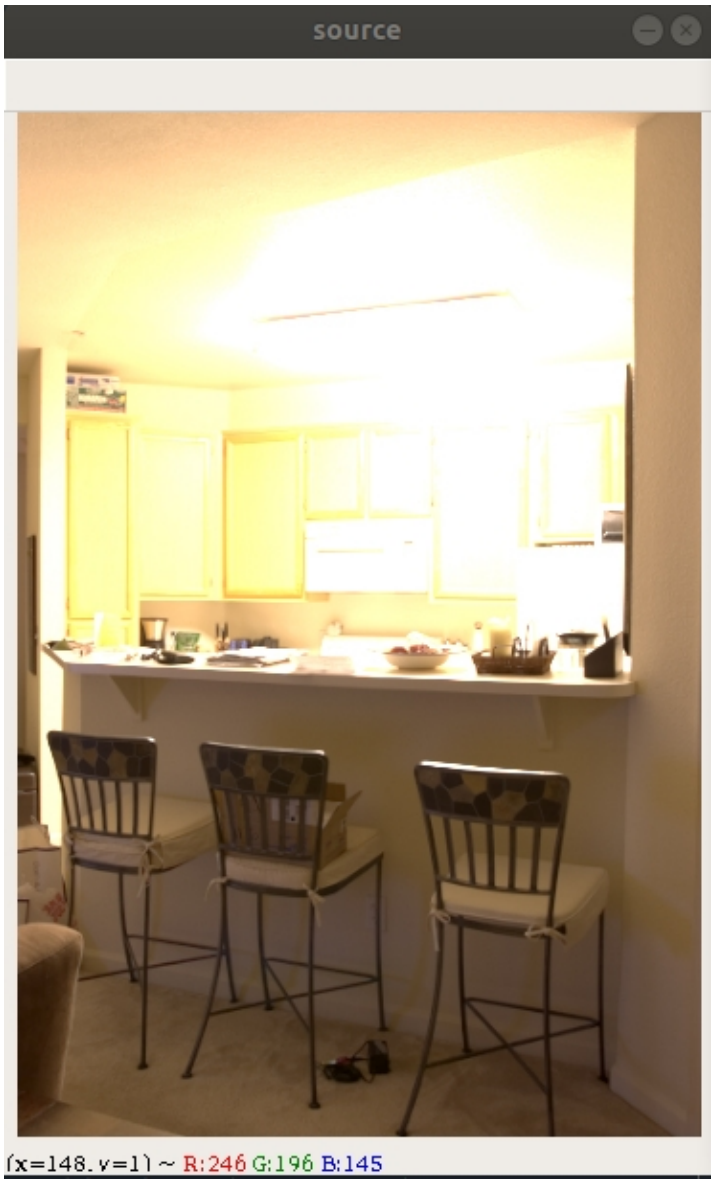


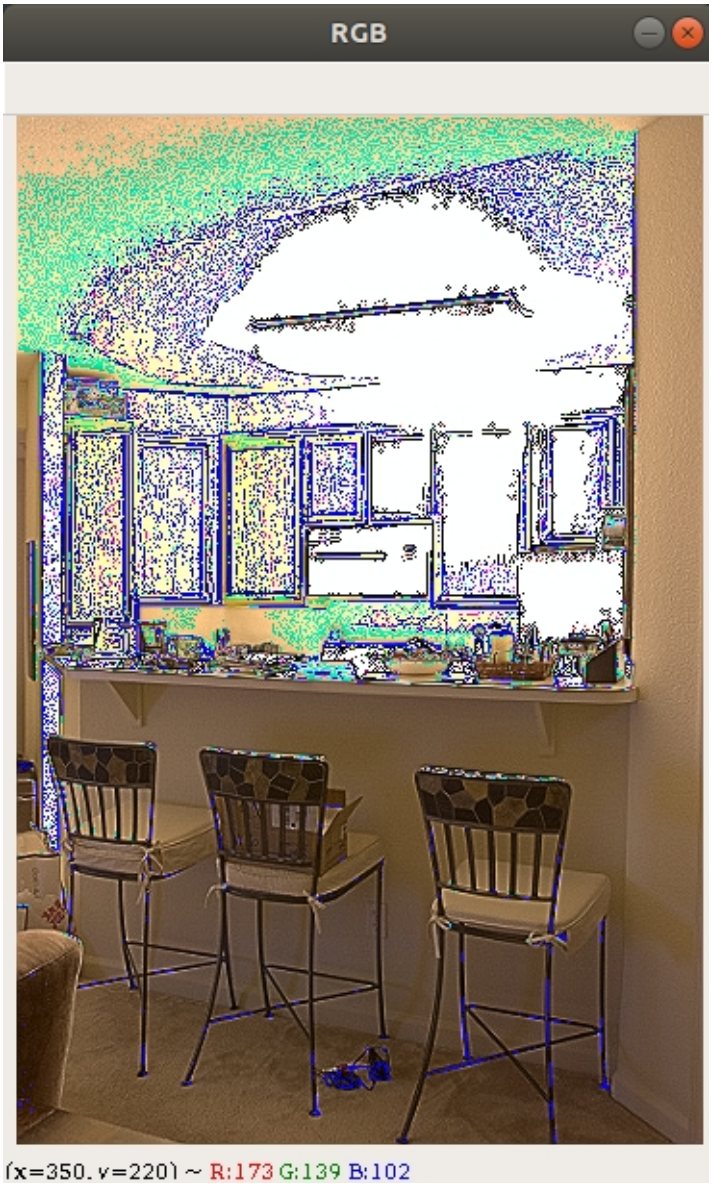

source
(x=69, y=71) ~ R:14 G:18 B:17


RGB
(x=509, y=221) ~ R:20 G:22 B:218


source
(x=675, y=498) ~ R:255 G:255 B:255


RGB
(x=737, y=14) ~ R:254 G:255 B:255





Discussions:

這次的作業要將最基本的色彩三通道，以紅藍綠三原色組合而成的彩色圖，轉換至其他種類的空間。在老師的上課中，老師介紹了很多種的色彩間轉換，也講解了他們各自的表示意義與特色，所以本次作業就讓我們嘗試在其他影像空間進行影像增強，實驗並討論其結果。

首先在最原本的 RGB 空間內進行影像增強，我再這次的作業所使用的影像增強技術為座標空間的拉普拉斯轉換，即是以拉普拉斯 kernel，再將原圖以拉普拉斯的算式簡換成與 kernel 的卷積計算，此種方法可以增強圖像中的邊緣，即是相鄰像素相差較大的地方。在像素相差較大的地方，人類在觀看時會覺的此部份較為強烈，所以可以達到影像增強的目的。

而此作業分成了三個部份，分別要在三種不同的色彩間做轉換，並進行影像增強，RGB 領域不用說，是大家做熟悉且基本的色彩空間，做完影像增強後效果很不錯，分別針對三通道做卷積，整體可以有效的增強圖像的特徵部份。

接下來將 RGB 轉至 HSI 色彩空間，HSI 分別代表了色相，飽和度，與明亮度。我也在之前有通過 HSV 色彩空間，這兩者極為相似，只相差了強度與明亮度的部份。HSI 將顏色部份以角度區分，直接概括所有顏色，飽和度表示了此顏色的深淺，而明亮度決定了物體所反射光的明亮度，極值為黑與白。在此空間進行影像增強確實可以再三種通道做到增強特徵的效果，但由於 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。

最後一部份為 L*a*b 空間，要轉至此空間必須要將 RGB 轉至 CIE XYZ 色彩空間， XYZ 色彩空間是將 RGB 轉換至更符合人體感色的機制，兩者之間有線性關係。接下來再將 XYZ 轉至 L*a*b 空間。L*a*b 空間是種顏色對立空間，以 L 表示明度，再以 a 跟 b 代表顏色的對立，a 表示從紅色到綠色的範圍，b 表示從黃色到藍色的距離。在此空間進行影像增強，雖然將 L*a*b 空間以圖像顯示就直接變成無法理解了，空間三通道所代表的意思雖然懂但將圖片顯示出來就很詭異，且經過影像增強後結果雖然不差但變化比預期的還特別。最後將增強後的影像轉回 RGB，經過多重的空間轉換，影像增強是針對 L*a*b 空間，所以在 RGB 會造成誤差是可以預期的，所以在結果的顯示上，會有神奇的功效。

最後總結我覺的有些影像處理技術是有明確且有效的，但在不同的色彩空間下所進行的影像處理技術，在物理意義下會變得有差異，像是在 RGB 增加顏色對比度，但在 HSI 空間下就會增強飽和度與明暗度的對比，所以在轉回 RGB 的色彩空間下，會變得與單純在 RGB 色彩空間進行影像處理產生不同的效果，當然 L*a*b 同理。所以必須針對自己所著重的重點進行影像處理，才可以達到理想的結果。

References and Appendix:

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

https://www.itread01.com/content/1544974982.html?fbclid=IwAR3HQ6_hPNuoc7oFsXR7nr2jsS5AxprErnAs6alN0Ra3rTPLrOVpp1txSls

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/