Image Processing Homework #3

Implementations of Color Image Enhancement on RGB, HSI, L*a*b Color Spaces

Advisor : Jin-Jang Leou (柳金章)
Student : Chih-Chia Lo (羅治嘉)
Registration Number : 610410002

Date Due : 2021/12/30 23:59
Date Handed In : 2021/12/22 17:00

Outline

I. Technical Description

A. Conversion from RGB to HSI

Given an image in RGB color format, the HSI components can be obtained by:

$$H = \begin{cases} \theta, & \text{if } B \geq G, \\ 360 - \theta, & \text{if } B < G, \end{cases} \tag{1}$$

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

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

$$I = \frac{1}{3}(R+G+B), \tag{4}$$

where H denotes hue, S denotes saturation, and I denotes intensity. The RGB values have been normalized to the range [0, 1].

The following figure is the source code of RGB to HSI handcrafted conversion:

```
H = acosd(((1/2)*((R-G)+(R-B)))./((((R-G).^2+((R-B).*(G-B))).^0.5)+eps));
if B > G
    H = 360 - acosd((((R-G)+(R-B))./2)./(sqrt((R-G).^2+(R-B).*(G-B))));
end
H = H/360;  %Normalized
S= 1 - (3./(R+G+B+0.000001)).*min(img,[],3);
I = (R+G+B)./3;
```

Fig. 1: Converting RGB to HSI.

B. Conversion from HSI to RGB

Given HSI values in the interval [0,1], we begin by multiplying H by 360° so that the hue values will be returned to its original range of [0°, 360°]. RG sector, GB sector, and BR sector are as follows.

RG sector ($0° \leq H < 120°$):

$$B = I(1-S), \tag{5}$$

$$R = I[1 + \frac{ScosH}{cos(60° - H)}], \tag{6}$$

$$G = 3I - (R+B). \tag{7}$$

GB sector ($120° \leq H < 240°$):

$$H = H - 120°, \tag{8}$$

$$R = I(1-S), \tag{9}$$

$$G = I[1 + \frac{ScosH}{cos(60° - H)}], \tag{10}$$

$$B = 3I - (R + G). \tag{11}$$

BR sector (240° ≤ H < 360°):

$$H = H - 240°, \tag{12}$$

$$G = I(1 - S), \tag{13}$$

$$B = I[1 + \frac{ScosH}{cos(60° - H)}], \tag{14}$$

$$R = 3I - (B + G). \tag{15}$$

The following figure is the source code of HSI to RGB handcrafted conversion:

```
H1=HSI(:,:,1);
S1=HSI(:,:,2);
I1=HSI(:,:,3);

R1=zeros(size(H1));
G1=zeros(size(H1));
B1=zeros(size(H1));
img_HSI=zeros([size(H1),3]);
%Multiply Hue by 360 to represent in the range [0 360]
H1=H1*360;
%RG Sector(0<=H<120)
%When H is in the above sector, the RGB components equations are
B1(H1<120)=I1(H1<120).*(1-S1(H1<120));
R1(H1<120)=I1(H1<120).*(1+((S1(H1<120).*cosd(H1(H1<120)))./cosd(60-H1(H1<120))));
G1(H1<120)=3.*I1(H1<120)-(R1(H1<120)+B1(H1<120));
%GB Sector(120<=H<240)
%When H is in the above sector, the RGB components equations are
%Subtract 120 from Hue
H2=H1-120;
R1(H1>=120&H1<240)=I1(H1>=120&H1<240).*(1-S1(H1>=120&H1<240));
G1(H1>=120&H1<240)=I1(H1>=120&H1<240).*(1+((S1(H1>=120&H1<240).*cosd(H2(H1>=120&H1<240)))./cosd(60-H2(H1>=120&H1<240))));
B1(H1>=120&H1<240)=3.*I1(H1>=120&H1<240)-(R1(H1>=120&H1<240)+G1(H1>=120&H1<240));
%BR Sector(240<=H<=360)
%When H is in the above sector, the RGB components equations are
%Subtract 240 from Hue
H2=H1-240;
G1(H1>=240&H1<=360)=I1(H1>=240&H1<=360).*(1-S1(H1>=240&H1<=360));
B1(H1>=240&H1<=360)=I1(H1>=240&H1<=360).*(1+((S1(H1>=240&H1<=360).*cosd(H2(H1>=240&H1<=360)))./cosd(60-H2(H1>=240&H1<=360))));
R1(H1>=240&H1<=360)=3.*I1(H1>=240&H1<=360)-(G1(H1>=240&H1<=360)+B1(H1>=240&H1<=360));
%Form RGB Image
img_HSI(:,:,1)=R1;
img_HSI(:,:,2)=G1;
img_HSI(:,:,3)=B1;
%Represent the image in the range [0 255]
```

Fig. 2: Converting HSI to RGB.

*C. Conversion from RGB to L*a*b**

The L*a*b* color components are:

$$L* = 116 \cdot h(\frac{Y}{Y_W}) - 16, \tag{16}$$

$$a* = 500[h(\frac{X}{X_W}) - h(\frac{Y}{Y_W})], \tag{17}$$

$$b* = 500[h(\frac{Y}{Y_W}) - h(\frac{Z}{Z_W})], \tag{18}$$

where

$$h(q) = \begin{cases} \sqrt[3]{q}, & q > 0.008856, \\ 7.787q + \frac{16}{116}, & q \le 0.008856, \end{cases} \tag{19}$$

and X_W, Y_W, and Z_W are reference white D65 with RGB working space sRGB. The matrix is shown as follow:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{20}$$

The L*a*b* color space is colorimetric (i.e., colors perceived as matching are encoded identically), perceptually uniform (i.e., color differences among various hues are perceived uniformly, and device independent. The L*a*b* system is an excellent decoupler of intensity (L^*) and color ($a^* = R\text{-}G$, $b^* = G\text{-}B$). [1] The following figure is the source code of RGB to L*a*b* handcrafted conversion:

```matlab
function [L,a,b] = my_RGB2Lab(R,G,B)
    if nargin == 1
        B = double(R(:,:,3));
        G = double(R(:,:,2));
        R = double(R(:,:,1));
    end
    if max(max(R)) > 1.0 || max(max(G)) > 1.0 || max(max(B)) > 1.0
        R = double(R) / 255;
        G = double(G) / 255;
        B = double(B) / 255;
    end
    % Set a threshold
    T = 0.008856;
    [M, N] = size(R);
    s = M * N;
    RGB = [reshape(R,1,s); reshape(G,1,s); reshape(B,1,s)];
    % RGB to XYZ
    MAT = [0.412453 0.357580 0.180423;
           0.212671 0.715160 0.072169;
           0.019334 0.119193 0.950227];
    XYZ = MAT * RGB;
    X = XYZ(1,:) / 0.950456;
    Y = XYZ(2,:);
    Z = XYZ(3,:) / 1.088754;
    XT = X > T;
    YT = Y > T;
    ZT = Z > T;
    Y3 = Y.^(1/3);
    fX = XT .* X.^(1/3) + (~XT) .* (7.787 .* X + 16/116);
    fY = YT .* Y3 + (~YT) .* (7.787 .* Y + 16/116);
    fZ = ZT .* Z.^(1/3) + (~ZT) .* (7.787 .* Z + 16/116);
    L = reshape(YT .* (116 * Y3 - 16.0) + (~YT) .* (903.3 * Y), M, N);
    a = reshape(500 * (fX - fY), M, N);
    b = reshape(200 * (fY - fZ), M, N);
    if nargout < 2
        L = cat(3,L,a,b);
    end
end
```

Fig. 3: Converting RGB to L*a*b*.

D. Conversion from L*a*b* to RGB

The XYZ components can be obtained by using inverse function h as follows:

$$Y = h(\frac{L^* + 16}{116})^{-1} \tag{20}$$

$$X = h(Y + \frac{a^*}{500})^{-1} \tag{21}$$

$$Z = h(Y - \frac{b^*}{200})^{-1} \tag{22}$$

and the RGB components can be obtained by using inverse matrix of reference white D65 with RGB working space sRGB:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{23}$$

The following figure is the source code of L*a*b* to RGB handcrafted conversion:

```
function [R, G, B] = my_Lab2RGB(L, a, b)
    if nargin == 1
        b = L(:,:,3);
        a = L(:,:,2);
        L = L(:,:,1);
    end
    % Thresholds
    T1 = 0.008856;
    T2 = 0.206893;
    [M, N] = size(L);
    s = M * N;
    L = reshape(L, 1, s);
    a = reshape(a, 1, s);
    b = reshape(b, 1, s);
    % Compute Y
    fY = ((L + 16) / 116) .^ 3;
    YT = fY > T1;
    fY = (~YT) .* (L / 903.3) + YT .* fY;
    Y = fY;
    % Alter fY slightly for further calculations
    fY = YT .* (fY .^ (1/3)) + (~YT) .* (7.787 .* fY + 16/116);
    % Compute X
    fX = a / 500 + fY;
    XT = fX > T2;
    X = (XT .* (fX .^ 3) + (~XT) .* ((fX - 16/116) / 7.787));
    % Compute Z
    fZ = fY - b / 200;
    ZT = fZ > T2;
    Z = (ZT .* (fZ .^ 3) + (~ZT) .* ((fZ - 16/116) / 7.787));
    % Normalize for D65 white point
    X = X * 0.950456;
    Z = Z * 1.088754;
    % XYZ to RGB
    MAT = [ 3.240479 -1.537150 -0.498535;
           -0.969256  1.875992  0.041556;
            0.055648 -0.204043  1.057311];
    RGB = max(min(MAT * [X; Y; Z], 1), 0);
    R = reshape(RGB(1,:), M, N);
    G = reshape(RGB(2,:), M, N);
    B = reshape(RGB(3,:), M, N);
    if nargout < 2
        R = uint8(round(cat(3,R,G,B) * 255));
    end
end
```

Fig. 4: Converting L*a*b* to RGB.

E. Color Image Enhancement

There are plenty of approaches to enhance color image. In this study, histogram equalization is applied. This approach usually increases the global contrast of many images, especially when the image is represented by a narrow range of intensity values. Through this adjustment, the intensities can be better distributed on the histogram utilizing the full range of intensities evenly. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the highly populated intensity values which is used to degrade image contrast. [2]

The first step of Histogram Equalization is to count the numbers of each pixel value and all pixels, so that we can obtain the probability of occurrence of gray level in an image can be approximated by:

$$p_x(i) = p(x = i) = \frac{n_i}{n}, \quad 0 \le i < L \tag{24}$$

where n_i denotes the count of each pixel value, n denotes the count of all pixels, and L denotes the total pixels in an image. After that, calculate the cumulative density function (cdf) of each pixel:

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j), \tag{25}$$

where $p_x(j)$ is from equation (24). The following figure is handcrafted function of histogram equalization:

```
function new_img = my_histogram(part_img)
    [row,col] = size(part_img);
    all_pixel = row * col;
    %Probaility Density Function
    pixel_count = zeros(1,256);
    for i=1:row
        for j=1:col
            pixel_count(part_img(i,j) + 1) = pixel_count(part_img(i,j) + 1) + 1;
        end
    end
    part_img_pdf = pixel_count / all_pixel;

    %Cumulative Distribution Function 累加
    part_img_cdf = zeros(1,256);
    part_img_cdf(1) = part_img_pdf(1);
    for i=2:256
        part_img_cdf(i) = part_img_cdf(i-1) + part_img_pdf(i);
    end

    %Make new image
    new_img = zeros(row,col);
    for i=1:row
        for j=1:col
            new_img(i,j) = part_img_cdf(part_img(i,j) + 1);
        end
    end
end
```

Fig. 5: Histogram Equalization.

F. The entire processes of the project

The following flow charts show the entire processes for each color spaces:

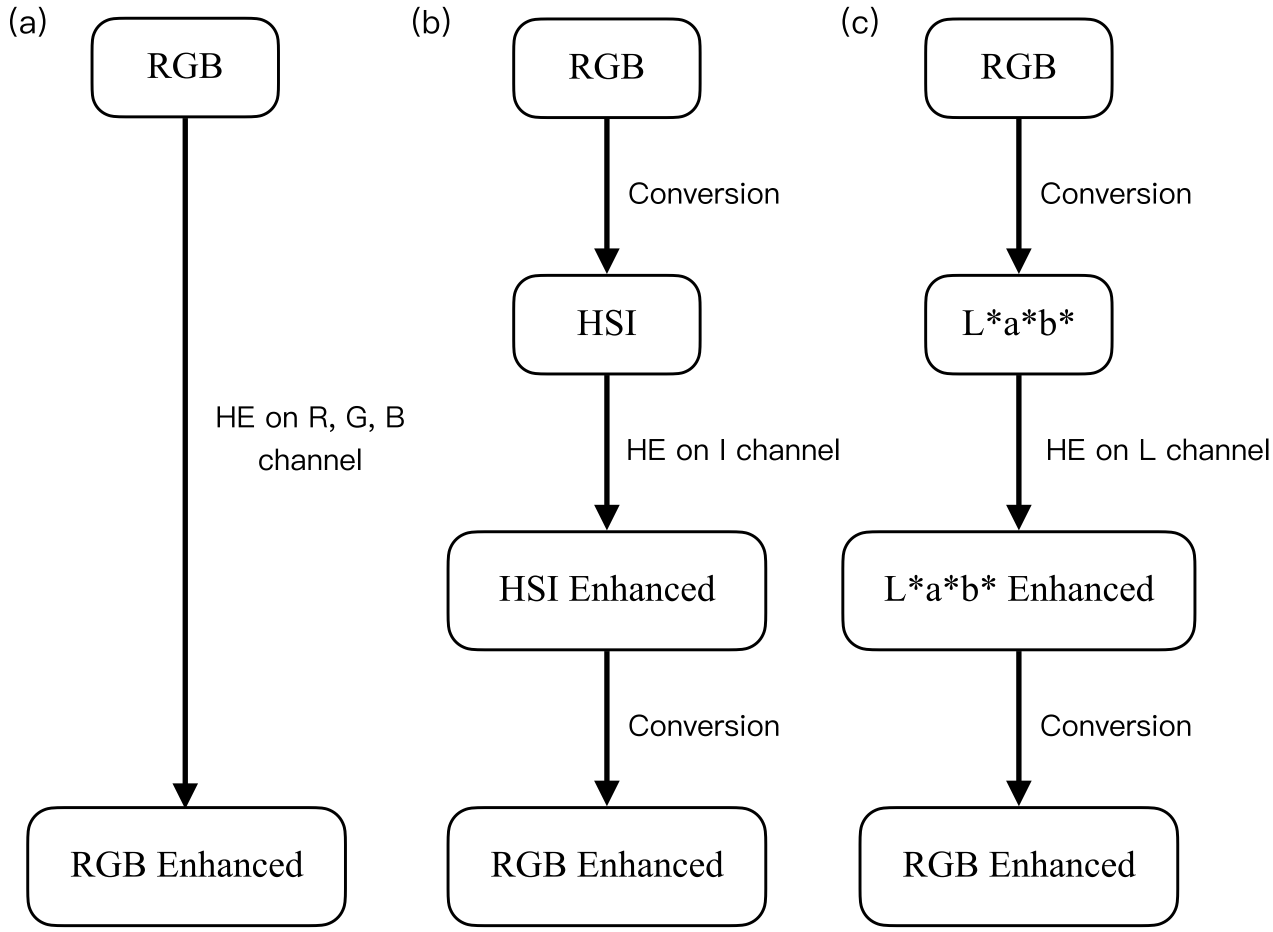

Fig. 6: Entire processes for each color spaces (a) RGB (b) HSI (c) L*a*b*.

where HE denotes histogram equalization and RGB Enhanced denotes the result for each color space enhancement. The rest of the code is assigning input image to variable and building subplots for RGB enhanced image.

II. Experimental Results

A. Datasets and Environments

This project contains a dataset of HW3_test_image from ecourse2, which there are four color images given in this dataset. The programming language of this project is Matlab_R2021b.

B. RGB, HSI, L*a*b* color image enhancement

The following figure is RGB, HSI, L*a*b* color image enhancement on image kitchen, house, church, and aloe in each row in order:

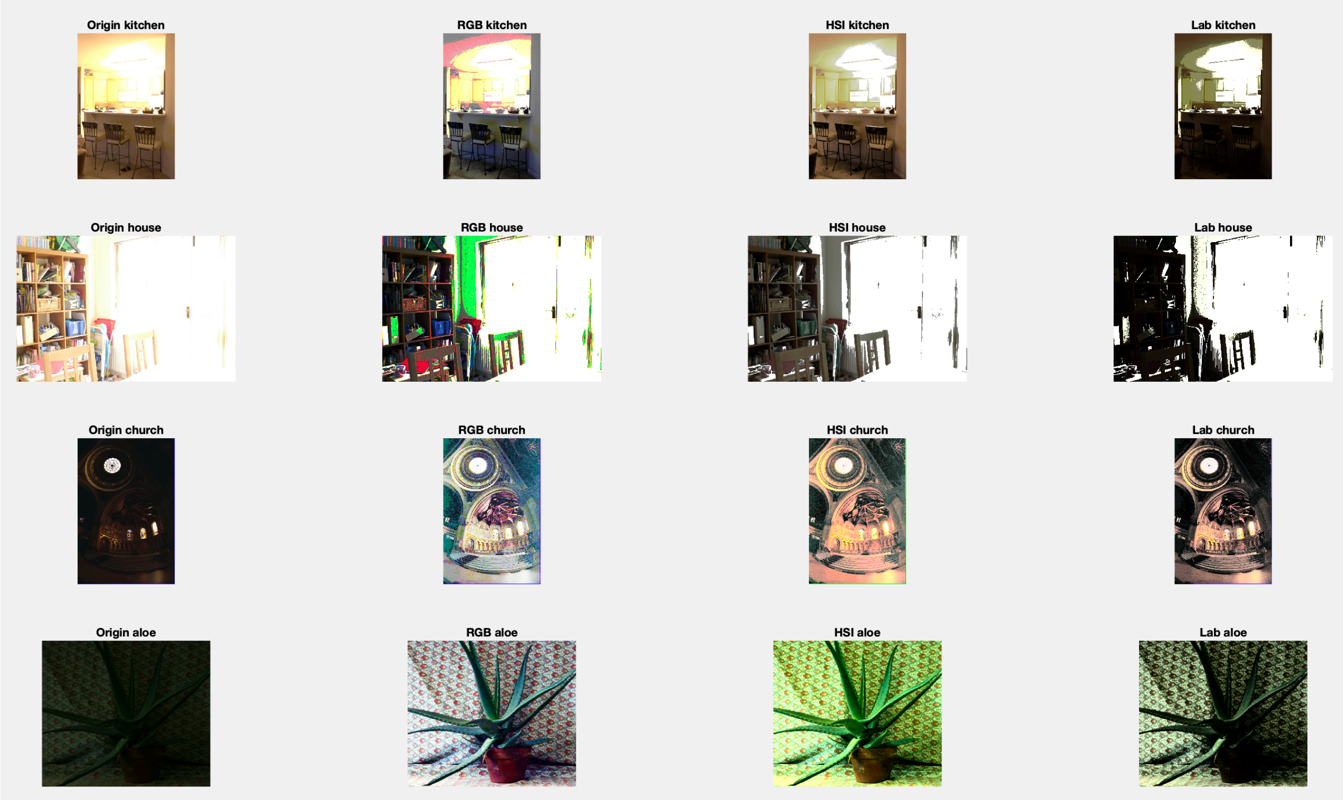

Fig. 7: Result for color space enhancement.

where the first column displays original image, the second column displays RGB enhancement result, the third column displays HSI enhancement result, and the last column displays L*a*b* result.

III. Discussions

A. Histogram Equalization on color image

There are two methods to enhance image, using power-law transformation (gamma correction) to adjust brightness and histogram equalization to adjust contrast. In this study, only histogram equalization is applied. It is clear that the performances of some enhanced images are good, while others are not suitable using histogram equalization. For example, church using histogram equalization on HSI and L*a*b* are far better than the original church being too dark to see, and aloe using histogram equalization on RGB image has the best performance among these color space results. However, it is awful doing histogram equalization on image house since pixels are immensely condense in one and nearby values, causing image house on RGB, HSI, and L*a*b* color space enhancement are completely ruined. In my opinion, gamma correction might be suitable for image house instead of histogram equalization.

IV. References and Appendix

[1] R. C. Gonzalez and R. E. Woods, "Digital Image Processing, 4th Ed., Pearson, New York, NY, 2018." in Chapter 6 Color Image Enhancement L*a*b* color model.
[2] R. C. Gonzalez and R. E. Woods, "Digital Image Processing, 4th Ed., Pearson, New York, NY, 2018." in Chapter 3 Image Enhancement Histogram Equalization.