

# Color Normalization of Retinal Images

Seyed Hesamoddin Hosseini

خلاصه

تصویربرداری رنگ در شرایط نورپردازی مختلف باعث بدمست آمدن تصاویر متفاوت از صحنه‌ی یکسان می‌شود. همین امر موجب می‌شود که سیستم‌های پیتای ماشین در برخی کاربردها، با وجود این مساله دچار مشکل شوند. یکی از راه حل‌های غلبه بر این مشکل این است که رنگ را در تصاویر نرمالیزه کنیم.

برای این منظور، ابتدا تصویر را از فضای رنگ RGB به فضای رنگ  $L\alpha\beta$  می‌بریم که در آن مولفه روشنایی از رنگ جدا شود. سپس با در نظر گرفتن مولفه‌های رنگ یک تصویر به عنوان مرجع، سعی می‌کنیم مولفه‌های رنگ دیگر تصاویر را به مولفه‌های رنگ تصویر مرجع تبدیل کنیم.

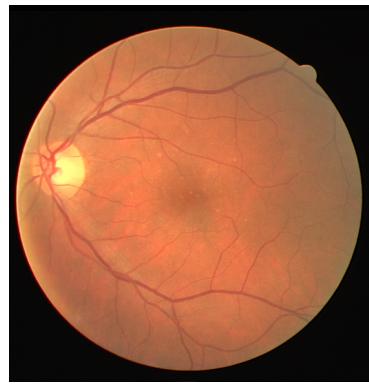
برای پیاده سازی این پروژه از Matlab استفاده می‌نماییم.

شرح تکنیکاً:

با توجه به توضیحات داده شده ابتدا تصویر را از فضای رنگ RGB به فضای رنگ  $L\alpha\beta$  می‌بریم که در آن مولفه روشنایی از رنگ جدا شود. سپس با در نظر گرفتن مولفه‌های رنگ یک تصویر به عنوان مرجع، سعی می‌کنیم مولفه‌های رنگ دیگر تصاویر را به مولفه‌های رنگ تصویر مرجع تبدیل کنیم.

مرحله اول:

ابتدا تصویر مرجع را از فضای رنگ RGB به فضای رنگ  $L\alpha\beta$  می‌بریم.



شکل 1 - تصویر هدف

برای این منظور از روابط زیر استفاده می‌کنیم. ابتدا تصویر را به فضای رنگ XYZ برد و سپس به فضای رنگ LSM می‌بریم و در نهایت به فضای رنگ  $L\alpha\beta$  تبدیل می‌کنیم.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (1)$$

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3897 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} L \\ \alpha \\ \beta \end{bmatrix} = \text{diag} \left( \frac{1}{\sqrt{3}} \frac{1}{\sqrt{6}} \frac{1}{\sqrt{2}} \right) \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \log L \\ \log M \\ \log S \end{bmatrix} \quad (3)$$

مرحله دوم:

تصویر ورودی را نیز از فضای رنگ RGB به فضای رنگ  $L\alpha\beta$  می بيريم.  
برای اين کار از روابط بالا استفاده می نمایيم. يعني ابتدا تصویر را به فضای رنگ XYZ برده و سپس به فضای رنگ LSM می بيريم و در نهايit به فضای رنگ  $L\alpha\beta$  تبديل می کنيم.

مرحله سوم:

حال لایه  $\alpha$  را بدون تغيير کنار گذاشته و لایه های  $\alpha$  و  $\beta$  را به کمک فرمول زير، بر اساس تصویر هدف و تصویر ورودی، محاسبه می نمایيم:

$$L_o = \frac{\sigma_{t,l}}{\sigma_{s,l}} (L_s - \mu_s) + \mu_t \quad (4)$$

برای محاسبه فرمول بالا، باید ميانگين و واريانس پيكسل های تصویر را در لایه های  $\alpha$  و  $\beta$  محاسبه نمایيم.  
منظور از  $L_o$  يك لایه از تصویر ورودی و منظور از  $L_s$  يك لایه از تصویر خروجي می باشد. از آنجا که هدف ما تغيير دادن مولفه های رنگ می باشد، پس لایه های  $\alpha$  و  $\beta$  را تغيير می دهيم.  
قطعه کد آن به صورت زير می باشد:

```
Lab_target_img_a = Lab_target_img(:,:,2);
meu_Lab_target_img_a = mean(Lab_target_img_a(:));

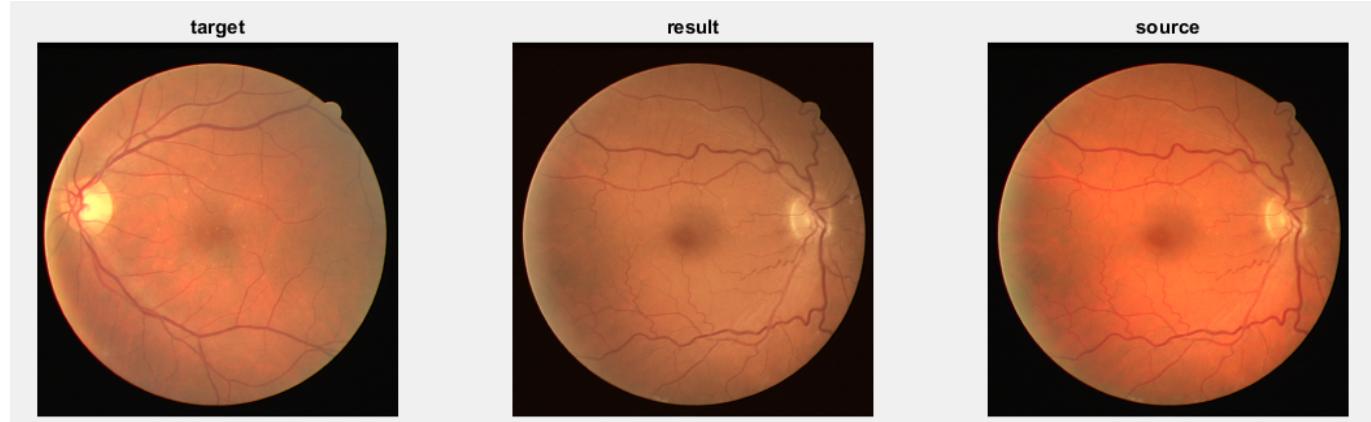
Lab_source_img_a = Lab_source_img(:,:,2);
meu_Lab_source_img_a = mean(Lab_source_img_a(:));

sigma_a = var(Lab_target_img_a(:))/var(Lab_source_img_a(:));

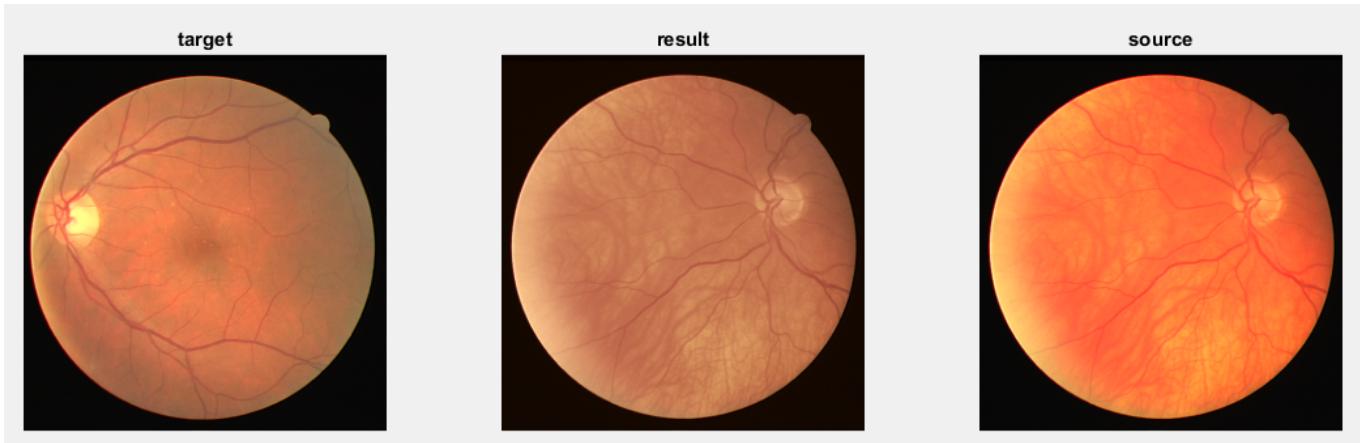
for row = 1 : img_row_size
    for col = 1 : img_col_size
        Lab_result_img(row,col,2) = (sigma_a * (Lab_source_img_a(row,col) -
meu_Lab_source_img_a)) + meu_Lab_target_img_a;
    end
end
```

مرحله چهارم:

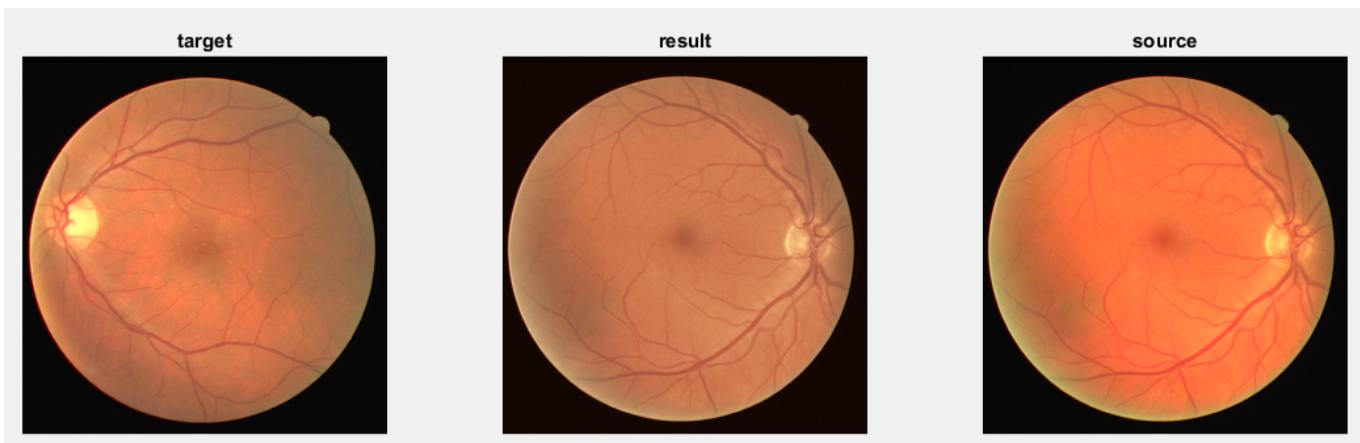
حال تصویر نتيجه را از فضای رنگ  $L\alpha\beta$  به فضای رنگ RGB بر ميگردانيم. برای اين کار از روابط (1) تا (3) به صورت معکوس استفاده می نمایيم. برای معکوس کردن ماتریس های ضرب از تابع inv استفاده می کنيم.  
خروجی الگوریتم برای انواع ورودی، به شکل زير می باشد:



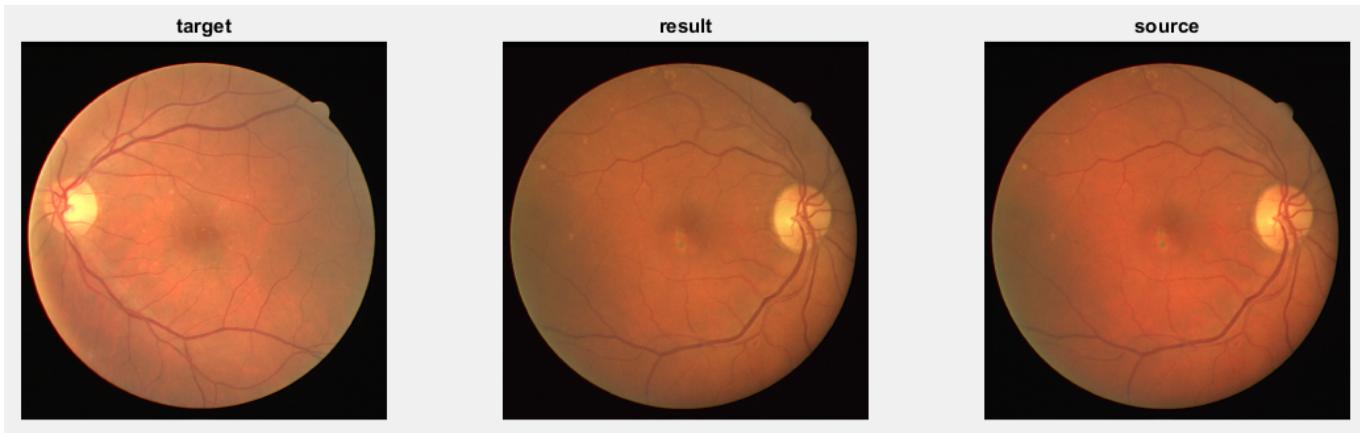
شکل 2 - نرماليزه نمودن تصویر شبکیه چشم برای مشاهده بهتر رگ ها



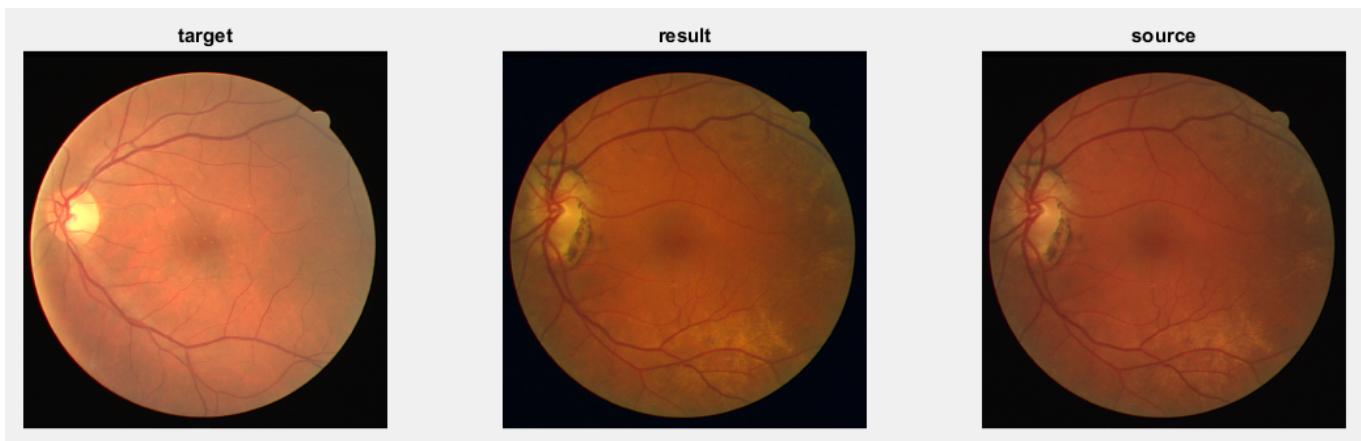
شکل 3 - نرمالایزه نمودن تصویر شبکیه چشم برای مشاهده بهتر رگ ها



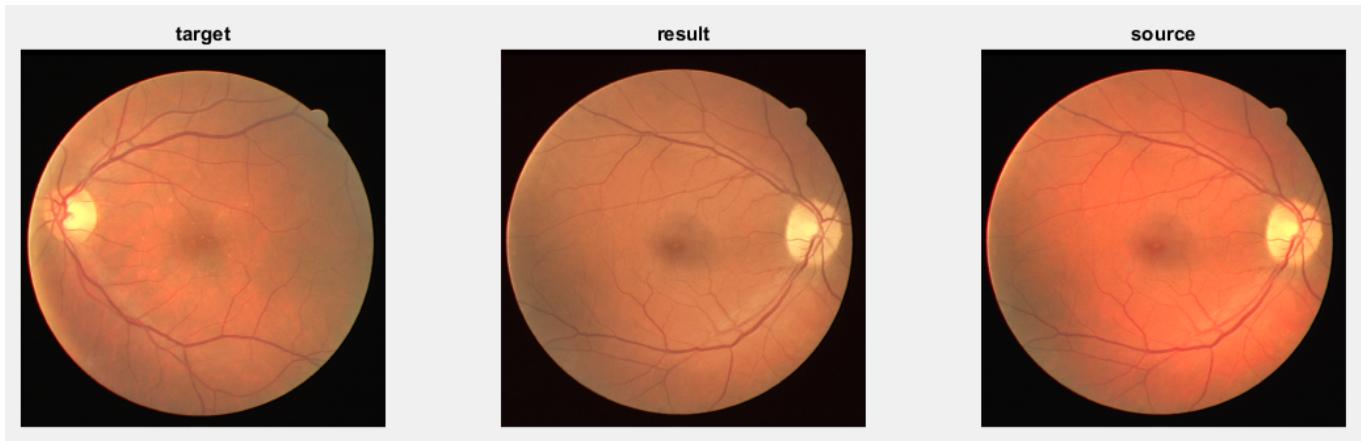
شکل 4 - نرمالایزه نمودن تصویر شبکیه چشم برای مشاهده بهتر رگ ها



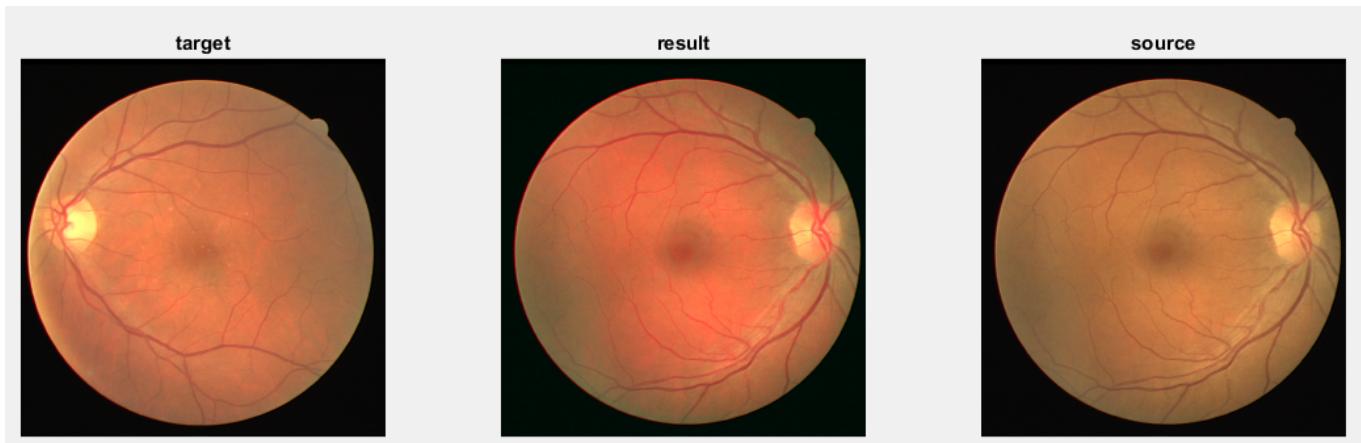
شکل 5 - نرمالایزه نمودن تصویر شبکیه چشم برای مشاهده بهتر رگ ها



شکل 6 - نرمالیزه نمودن تصویر شبکیه چشم برای مشاهده بهتر رگ ها



شکل 7 - نرمالیزه نمودن تصویر شبکیه چشم برای مشاهده بهتر رگ ها



شکل 8 - نرمالیزه نمودن تصویر شبکیه چشم برای مشاهده بهتر رگ ها

**نتیجه گیری:**

در فضای رنگ RGB مفهوم رنگ و مفهوم روشنایی، ماهیت مستقلی از یکدیگر ندارند. و از آنجا که هدف ما تغییر رنگ و ثابت نگه داشتن روشنایی بود، لازم شد تا فضای رنگ انتخاب کنیم که در آن ماهیت روشنایی و رنگ از یکدیگر مستقل باشند. برای این منظور، ابتدا تصویر را از فضای RGB به فضای رنگ  $L\alpha\beta$  می‌بریم که در آن مولفه روشنایی از رنگ جدا شود. سپس با در نظر گرفتن مولفه‌های رنگ یک تصویر به عنوان مرجع، سعی می‌کنیم مولفه‌های رنگ دیگر تصاویر را به مولفه‌های رنگ تصویر مرجع تبدیل کنیم.

مشاهده می‌شود که در اکثر نتایج بدست آمده، این الگوریتم به خوبی عمل کرده است و تشخیص رگ های چشم، آسان تر شده است.

سورس کد پروژه با نرم افزار متلب:

$L\alpha\beta$  به RGB پیاده سازی تابع تبدیل فضای رنگ

```

function [output] = myRGB2Lab(RGB_img)

[img_row_size, img_col_size] = size(RGB_img(:,:,1));

R = RGB_img(:,:,1);
G = RGB_img(:,:,2);
B = RGB_img(:,:,3);

X = zeros(img_row_size, img_col_size);
Y = zeros(img_row_size, img_col_size);
Z = zeros(img_row_size, img_col_size);

M1 = [0.4124, 0.3576, 0.1805;
       0.2126, 0.7152, 0.0722;
       0.0193, 0.1192, 0.9505];

for row = 1:img_row_size
    for col = 1:img_col_size

        result = M1 * [ R(row, col); G(row, col); B(row, col) ];
        X(row, col) = result(1);
        Y(row, col) = result(2);
        Z(row, col) = result(3);

    end
end

LL = zeros(img_row_size, img_col_size);
M = zeros(img_row_size, img_col_size);
S = zeros(img_row_size, img_col_size);

M1 = [0.3897, 0.6890, -0.0787;
       -0.2298, 1.1834, 0.0464;
       0.0000, 0.0000, 1.0000];

for row = 1:img_row_size
    for col = 1:img_col_size

        result = M1 * [X(row, col); Y(row, col); Z(row, col)];
        LL(row, col) = result(1);
        M(row, col) = result(2);
        S(row, col) = result(3);

    end
end

M1 = [1, 1, 1;
       1, 1, -2;
       1, -1, 0];

L = zeros(img_row_size, img_col_size);
a = zeros(img_row_size, img_col_size);
b = zeros(img_row_size, img_col_size);

for row = 1:img_row_size
    for col = 1:img_col_size

```

```

        result = diag([1/sqrt(3),1/sqrt(6),1/sqrt(2)]) * M1 * [LL(row, col) ; M(row,
col); S(row, col)];

        L(row, col) = result(1);
        a(row, col) = result(2);
        b(row, col) = result(3);

    end
end

output = cat(3,L,a,b);

end

```

پیاده سازی تابع تبدیل فضای رنگ  $\text{LaB}^{\star}$  به RGB

```

function [output] = myLab2RGB(Lab_img)

[img_row_size, img_col_size] = size(Lab_img(:,:,1));

L = Lab_img(:,:,1);
a = Lab_img(:,:,2);
b = Lab_img(:,:,3);

LL = zeros(img_row_size, img_col_size);
M = zeros(img_row_size, img_col_size);
S = zeros(img_row_size, img_col_size);

M1 = diag([1/sqrt(3),1/sqrt(6),1/sqrt(2)]) * [1, 1, 1;
                                                1, 1, -2;
                                                1, -1, 0];

for row = 1:img_row_size
    for col = 1:img_col_size

        result = inv(M1) * [L(row, col); a(row, col); b(row, col)];
        LL(row, col) = result(1);
        M(row, col) = result(2);
        S(row, col) = result(3);

    end
end

X = zeros(img_row_size, img_col_size);
Y = zeros(img_row_size, img_col_size);
Z = zeros(img_row_size, img_col_size);

M1 = [0.3897, 0.6890, -0.0787;
      -0.2298, 1.1834, 0.0464;
      0.0000, 0.0000, 1.0000];

for row = 1:img_row_size
    for col = 1:img_col_size

        result = inv(M1) * [LL(row, col); M(row, col); S(row, col)];
        X(row, col) = result(1);
        Y(row, col) = result(2);
        Z(row, col) = result(3);

    end
end

```

```

M1 = [0.4124, 0.3576, 0.1805;
      0.2126, 0.7152, 0.0722;
      0.0193, 0.1192, 0.9505];

R = zeros(img_row_size, img_col_size);
G = zeros(img_row_size, img_col_size);
B = zeros(img_row_size, img_col_size);

for row = 1:img_row_size
    for col = 1:img_col_size

        result = inv(M1) * [X(row, col); Y(row, col); Z(row, col)];
        R(row, col) = result(1);
        G(row, col) = result(2);
        B(row, col) = result(3);

    end
end

output = cat(3,R,G,B);

end

```

پیاده سازی پروژه:

```

close all;
clear;

rgb_target_img = imread('D:/Im281-1.tif');
[img_row_size, img_col_size] = size(rgb_target_img(:,:,1));
rgb_target_img = im2double(rgb_target_img);
Lab_target_img = myRGB2Lab(rgb_target_img);

rgb_source_img = imread('D:/Im288-1.tif');
rgb_source_img = im2double(rgb_source_img);
Lab_source_img = myRGB2Lab(rgb_source_img);

Lab_result_img = zeros(img_row_size, img_col_size, 3);

Lab_target_img_a = Lab_target_img(:,:,2);
Lab_target_img_b = Lab_target_img(:,:,3);

meu_Lab_target_img_a = mean(Lab_target_img_a(:));
meu_Lab_target_img_b = mean(Lab_target_img_b(:));

Lab_source_img_a = Lab_source_img(:,:,2);
Lab_source_img_b = Lab_source_img(:,:,3);

meu_Lab_source_img_a = mean(Lab_source_img_a(:));
meu_Lab_source_img_b = mean(Lab_source_img_b(:));

sigma_a = var(Lab_target_img_a(:))/var(Lab_source_img_a(:));
sigma_b = var(Lab_target_img_b(:))/var(Lab_source_img_b(:));

for row = 1 : img_row_size
    for col = 1 : img_col_size
        Lab_result_img(row,col,2) = (sigma_a * (Lab_source_img_a(row,col) -
meu_Lab_source_img_a)) + meu_Lab_target_img_a;
    end
end

for row = 1 : img_row_size

```

```
for col = 1 : img_col_size
    Lab_result_img(row,col,3) = (sigma_b * (Lab_source_img_b(row,col) -
meu_Lab_source_img_b)) + meu_Lab_target_img_b;
end
end

Lab_result_img(:,:,1) = Lab_source_img(:,:,1);

rgb_result_img = myLab2RGB(Lab_result_img);

figure;
subplot(1,3,1);
imshow(rgb_target_img, []);
title('target');
subplot(1,3,2);
imshow(rgb_result_img, []);
title('result');
subplot(1,3,3);
imshow(rgb_source_img, []);
title('source');
```