CSCI 3290 Assignment 2 – Color Transfer & Contrast Preserving Decolorization

Total points: 100

Due Date: 11:59pm on Wednesday, November 4th, 2015

I. Backgrounds

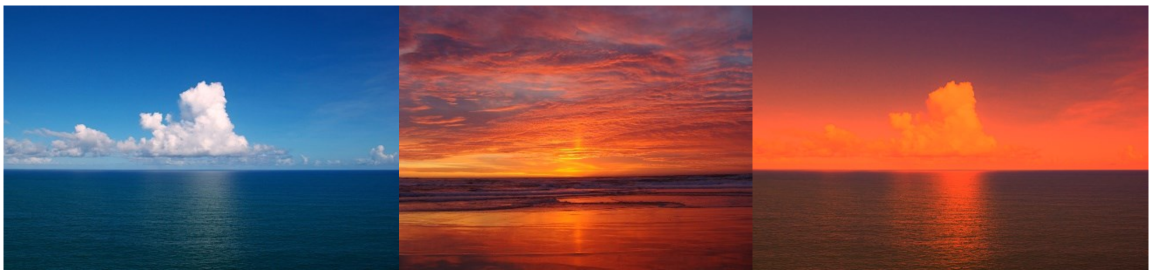
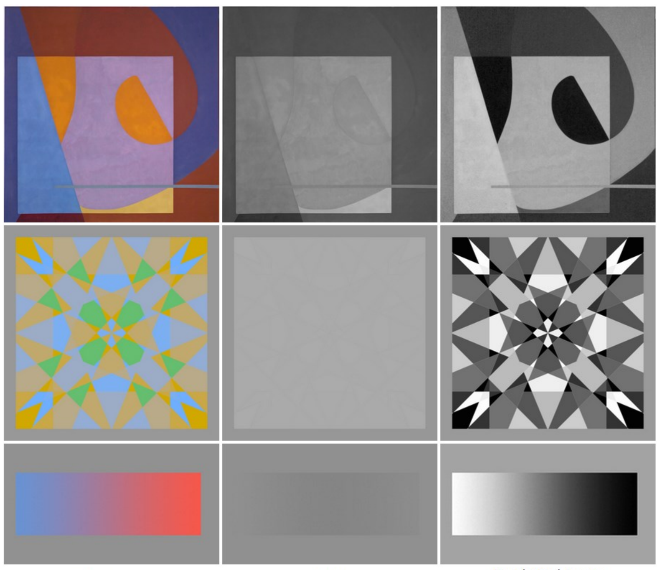


Image A Image B Color Transfer

Color Transfer - Transfer color style from one image to another. For example, we want to change the appearance of Image A (ocean_day) into the style of Image B (ocean_sunset). Above is one effect generated by “Color Transfer between images” [6].

Decolorization - the process to transform a color image to a grayscale one - is a basic tool in digital printing, stylized black-and-white photography, and in many single channel image processing applications.



Color Image CIE Y Decolorized Image (Chrominance contrast is preserved)

In this assignment, you will implement approaches aiming at transferring color and maximally preserving the original color contrast in decolorization. To start your assignment, you will use test images and the skeleton code provided by us. You are also encouraged to use your own pictures after coding is finished.

II. Details

Part 1: Color Transfer between Images

We call the image that we want to change "**source** image" *img_src*, and the desired image "**target** image" *img_tar*. Our algorithm follows that of "Color Transfer between images" [6].

The goal is to transfer source image's color to the target image. The modifications need to be done in the colorspace named "Lab" color space.

Hint: you **CANNOT** use build-in functions to do the colorspace transformation.

Step 1: Convert input img_src & img_tar into Lab- color space

1. From RGB to LMS: for each color (R,G,B) in RGB color space.

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2. From LMS to log-LMS

$$\begin{aligned} \boldsymbol{L} &= \log L \\ \boldsymbol{M} &= \log M \\ \boldsymbol{S} &= \log S \end{aligned}$$

Attention: (L,M,S) values may be very close to 0, then log-LMS will be –inf. Please use x = log(max(eps,x)) instead of x = log(x) for numerical stability (eps is a very small float number)

3. From log-LMS to Lab color space

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{L} \\ \mathbf{M} \\ \mathbf{S} \end{bmatrix}$$

Step 2: Calculate Mean and Standard Deviation for each channel of source and target image in Lab- color space

l_s, a_s, b_s are 3 channels of *img_src*

l_t, a_t, b_t are 3 channels of *img_tar*

$\langle l_s \rangle, \langle a_s \rangle, \langle b_s \rangle$ are mean value of *img_src*'s 3 channels

$\langle l_t \rangle, \langle a_t \rangle, \langle b_t \rangle$ are mean value of *img_tar*'s 3 channels

$\sigma_s^l, \sigma_s^a, \sigma_s^b$ are Standard Deviation value of *img_src*'s 3 channels

$\sigma_t^l, \sigma_t^a, \sigma_t^b$ are Standard Deviation value of *img_tar*'s 3 channels

Step 3: Color transfer

Get the 3 channels of the result image by the following

$$l_{res} = (l_s - \langle l_s \rangle)\frac{\sigma_t^l}{\sigma_s^l} + \langle l_t \rangle$$

$$a_{res} = (a_s - \langle a_s \rangle)\frac{\sigma_t^a}{\sigma_s^a} + \langle a_t \rangle$$

$$b_{res} = (b_s - \langle b_s \rangle)\frac{\sigma_t^b}{\sigma_s^b} + \langle b_t \rangle$$

Step 4: Convert back to RGB color space

Please derive the equations to convert from Lab back to RGB color space.

Part 2: Basic Decolorization Algorithm

Suppose I is the input color image and g is the resulting gray-scale image. In order to achieve "contrast preserving", we need to make the difference between neighboring gray-scale pixels (g_p, g_q) similar to the contrast between original color pixels (I_p, I_q).

Our objective is to optimize a contrast preserving energy function

$$\min_g \sum_{p,q} \left(g_p - g_q - \delta_{p,q}\right)^2 \qquad (1)$$

g_p and g_q are the gray values for two pixels p and q, which are neighbors. The neighboring is defined either in x or y direction, i.e., for image position $p = (\mathrm{i}, \mathrm{j})$ in the i-th row and j-th column, it has two neighbors where $q = (i+1, j)$ or $q = (i, j+1)$. $\delta_{p,q}$ is the color difference between pixels I_p and I_q.

Based on the Euclidian distance in *Lab* color space, color contrast is defined as

$$|\delta_{p,q}| = \sqrt{\left(L_p - L_q\right)^2 + \left(a_p - a_q\right)^2 + \left(b_p - b_q\right)^2}$$

$$\mathrm{sign}\left(\delta_{p,q}\right) = \mathrm{sign}\left(L_p - L_q\right)$$

The sign of $\delta_{p,q}$ is determined by L channel. Solving Equation (1) needs to take its first order derivative with respect to each pixel value g_p, and let the derivative equal to 0. This results in the following linear equations

$$g_p - g_q = \delta_{p,q}, \qquad \text{for all pixels } p = (\mathrm{i},\mathrm{j}) \text{ and their neighbors } q \qquad (2)$$

Since each pixel p has 2 neighbors q (one on the right of p with q = $(i+1,j)$, one immediately under p with $q = (i, j+1)$). Therefore, we have 2 equations for each pixel p accordingly.

Equation (2) can be solved via constructing linear equations as

$$AG = \Delta;$$

Where G is a column vector form of the gray-scale image g as

$$G = \left[g_{1,1}, g_{2,1}, \cdots, g_{m,1}, g_{1,2}, g_{2,2}, \cdots, g_{m,2}, \cdots, g_{1,n}, g_{2,n}, \cdots, g_{m,n}\right]^T$$

Here image g is with resolution $m \times n$. $g_{i,j}$ is the i-th row, j-th column pixel value.

Δ is the vector form for all pixel pair (p,q). $\delta_{p,q}$ as $\Delta = \left[\eta^x_{1,1}, \eta^x_{2,1}, \ldots \eta^x_{m,n}, \eta^y_{1,1}, \eta^y_{2,1}, \ldots \eta^y_{m,n}\right]^T$. $\eta^x_{i,j} = \delta_{p,q}$ where $p = (i,j)$ and $q = (i+1,j)$. $\eta^y_{i,j} = \delta_{p,q}$ where $p = (i,j)$ and $q = (i,j+1)$.

A is a highly sparse matrix with size $2mn \times mn$ with each row representing one neighborhood equation in (2) corresponding to Δ.

So what you need to do is as follows:

Step 1: Convert the input image to Lab space

Do the same steps as those in part 1 of this assignment.

Step 2: Compute $\delta_{p,q}$ for each two neighboring pixels

Step 3: Constructing matrix A and vectors G and Δ

To convert a matrix G into vector format, you can use Matlab function `g = G(:);` to convert a vector to matrix, use Matlab function `reshape()`.

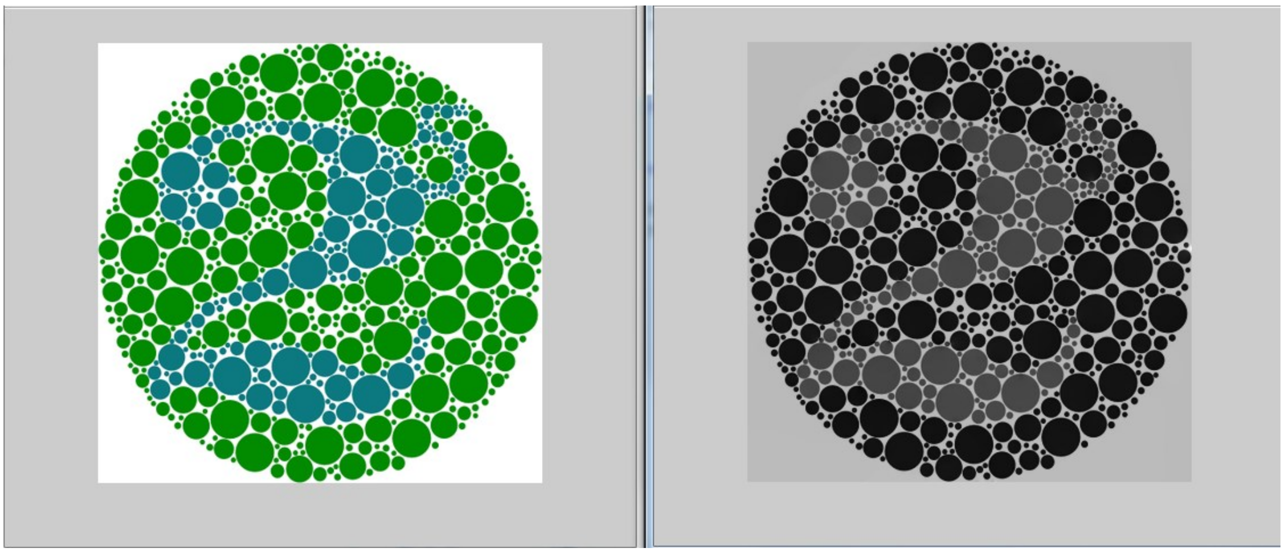
You can use the sparse matrix functions `sparse()` and `spdiags()` in MATLAB to increase the speed for constructing A.

We will discuss it in our tutorial in more details.

Step 4: Solve the objective function to get G

You can use `inv()` or ‘\’ in MATLAB. Please perform running time comparison for these two methods in your report, and search information online to discuss their difference.

We show an example below for you to ensure that you are implementing the algorithm in a right way.



Part 3: Decolorization Evaluation: Color Contrast Preserving Ratio (CCPR)

To quantitatively evaluate the decolorization algorithms, we propose a new metric. It is based on the finding that if the color difference δ is smaller than a threshold τ, it becomes nearly invisible in human vision. The task of contrast-preserving decolorization is therefore to maintain color change that is perceivable by human. We define a color contrast preserving ratio (CCPR) as

$$CCPR = \frac{\#\{(p,q)|(p,q) \in \Omega, |g_p - g_q| \geq \tau\}}{||\Omega||}$$

where Ω is the set containing all neighboring pixel pairs with their original color difference $\delta_{p,q} \geq \tau$. $||\Omega||$ is the number of pixel pairs in Ω.

$\#\{(p,q)|(p,q) \in \Omega, |g_p - g_q| \geq \tau\}$ is the number of pixel pairs in Ω that are still distinctive after decolorization.

For each image, please report the average CCPR score under different τ (ranging from 1 to 15).

Packages we provide:

1. Specification of Assignment 2
2. MATLAB skeleton code
3. Sample images for testing
4. Links for more data and related resource

III. Requirements

Your submission should contain the following:

a) Codes including:

- `\color_transfer\run_color_transfer.m` – To start your algorithm
- `\color_transfer\color_rgb2lab.m` - Convert RGB to Lab
- `\color_transfer\color_lab2rgb.m` - Convert Lab to RGB
- `\color_transfer\color_transfer.m` - Transfer color in Lαβ
-
- `\color2gray\run_cprgb2gray.m` – To start your algorithm
- `\color2gray\cprgb2gray.m` - Part 2: Basic decolorization algorithm
- `\color2gray\CCPR.m` - Part 3: Method evaluation by CCPR
- Other codes you would like to add.

Remarks:

- You can use functions provided by MATLAB, its Toolboxes and libraries supplied by TAs (Unless identified elsewhere). Note you need to describe the algorithms in your **OWN** words in the report.
- If you use others' functions or codes, please clearly claim all of them in the report, and **SUBMIT** them together with your assignment. Otherwise, it might be considered as plagiarism.
- You own work **MUST** be the majority of all your uploaded codes.

b) Reports:

- For this assignment, and all other assignments, you must make a report in **HTML**.
- In the report, you will describe your algorithm and any decisions you made to write your algorithm a particular way. Then you will show and discuss the results of your algorithm for all the test cases.
- Discussion on algorithms' efficiency.
- Discuss any extra credit you did if you want.
- Ideas and algorithms from others **MUST** be clearly claimed. List papers or links as reference if needed.
- Feel free to add any other information you feel is relevant.

c) Hand-in

The folder you hand in must contain the following:

1. **README.docx** - anything about the project that you want to tell the TAs, including brief introduction of the usage of the codes

2. **code/** - directory containing all your code for this assignment

   - code/color_transfer/ - For Part 1
   - code/color2gray/ - For Part 2 & 3

3. **html/** - directory containing all your html report for this assignment (including images). Your web page should only display compressed images (e.g. **jpg** or **png**).

4. **html/index.html** - home page for your results

Compressed the folder into <*your student ID*>**-Asgn2.zip** or <*your student ID*>**-Asgn2.rar**, and upload it to e-Learning system.

IV. Scoring & Extra Bonus

a) Score

- ✧ +10 pts: Color transfer algorithm
- ✧ +25 pts: Basic decolorization algorithm
- ✧ +25 pts: The CCPR metric to evaluate the decolorization
- ✧ +20 pts: Report
- ✧ +20 pts: Extra credits
- ✧ -5*n pts: Losing 5 points every time (after the first) you do not follow the instructions to prepare the uploading file format

b) Extra Credit

Three options for extra credit:

1. Implement a more advanced decolorization algorithm considering **weak color order** in paper "Contrast Preserving Decolorization". You can refer to the PPT to understand the algoirthm. (up to 20 pts)

   The parametric decolorization function is defined as $\mathbf{g} = \mathrm{f}(\mathbf{I})$, which is a mapping from color image **I** to gray image **g**. The mapping function is further expressed as

$$f(r,g,b,w) = \sum_i w_i m_i$$

   where m_i is an element in $\{r,g,b,rg,rb,gb,r^2,g^2,b^2\}$; w_i is the corresponding coefficients. The weak color order is defined in Eq. (5) of the paper. Considering the weak order, the objective function becomes

$$E(w) = -\sum_{(x,y)\in\mathbb{N}} \ln\{\alpha_{x,y} \exp\left\{-\frac{\left|\sum_i w_i l_{i(x,y)} - \delta_{x,y}\right|^2}{2\sigma^2}\right\} + (1-\alpha_{x,y})\exp\{-\frac{\left|\sum_i w_i l_{i(x,y)} + \delta_{x,y}\right|^2}{2\sigma^2}\}\} \quad (3)$$

To solve the Eq. (3) here (the same as Eq. (11) in the paper), the first order derivative can be expressed as Eq. (13) in the paper. Define $G(\delta_{x,y}, \sigma^2) = \exp\left\{-\frac{\left|\sum_i w_i l_{i(x,y)} - \delta_{x,y}\right|^2}{2\sigma^2}\right\}$.

The following algorithm solves w iteratively.

Algorithm 1 Weak-Order Decolorization

```
1: **input:** color image $\mathbf{c} = (r, g, b)$
2: initialize $\omega_i^0$, $k \leftarrow 0$;
3: compute $\delta_{x,y}$ and $l_{i(x,y)}$ for each neighboring pixel
   pair;
4: **repeat**
5:     compute $\beta^k$ given $\omega^k$;
6:     solve for $\omega^{k+1}$;
7:     $k \leftarrow k + 1$;
8: **until** $k > k_{max}$
9: $g = f(\mathbf{c}; \omega^k)$;
10: map $g$ back to the range $[\min(\mathbf{c}), \max(\mathbf{c})]$;
11: **output:** grayscale image $g$
```

Where β and w can be updated by the following equations.

$$\beta_{x,y} := \frac{\alpha_{x,y} G(\delta_{x,y}, \sigma^2)}{\alpha_{x,y} G(\delta_{x,y}, \sigma^2) + (1-\alpha_{x,y}) G(-\delta_{x,y}, \sigma^2)}.$$

$$\sum_{(x,y)\in\mathcal{N}} \sum_i \omega_i^{k+1} l_{i(x,y)} l_{j(x,y)} = (2\beta_{x,y}^k - 1) l_{j(x,y)} \delta_{x,y}.$$

Initial values for w can be found in the paper.

2. Implement the even faster real-time decolorization method described in the PPT. (up to 15 pts)

3. Evaluate a total of three different decolorization algorithms (including what you have implemented in this assignment) under CCPR metric. To find more decolorization algorithm, see here. Explain each algorithm you have tried and analyze their performance. You can try whatever you think is simple and reasonable (or some existing implementation that is theoretically different) for decolorization. Include code you used or implemented in your package. (up to 10 pts)

V. Other Remarks

- **Five** late days total, to be spent wisely
- 20% off from each extra late day

- The three assignments are to be completed **INDIVIDUALLY** on your own.
- You are encouraged to use methods in related academic papers.
- Absolutely **NO sharing or copying** code! **NO sharing or copying** reports! Offender will be given a failure grade and the case will be reported to the faculty.

VI. Reference

[1] http://www.cs.northwestern.edu/~ago820/color2gray/color2gray.pdf

[2] http://www.cse.cuhk.edu.hk/~leojia/papers/decolorization_iccp12.pdf

[3] http://www.cse.cuhk.edu.hk/~leojia/papers/decolorization_ijcv14.pdf

[4] http://www.cse.cuhk.edu.hk/~leojia/projects/color2gray/c2gslides.pptx

[5] http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/

[6] http://www.cs.utah.edu/~shirley/papers/ColorTransfer.pdf