White Balance

Computational Photography: Project 2

1 - Objective

The goal of this project is to understand the purpose of white balance and some of the different methods of doing so. To this end, you will implement and use several white balancing techniques on images of your choice.

2 - Deadline

It should be submitted on T-Square by 11:55PM on Monday.

3 - Process

3.1 Download the base source

Download and unzip the folder with the base code for this project.

3.2 Project description

In this project you will adjust white balance in Processing using three methods. For the first method we will provide working Processing code, and for the other two methods you will implement the math by filling in the necessary code in this same Processing file.

You will take 2 photographs for this project. You will select 1 white object (a sheet of paper, plastic, etc…) and one colorful object (a cola can, book cover, whiteboard markers, etc…) and place them in the same scene. You will photograph them together once under one lighting condition (indoor, outdoor, yellow light, white light, moonlight, candlelight, etc…) and again under a different lighting condition. Record the conditions you choose. You will then use the 3 different methods you implement to perform white balancing on each photo and save the results.

As we talked about in class, white balance adjustment changes the "tint" of the colors in the image by multiplying each of the red, green, and blue channels by different constant correction factors - sR for the red channel, sG for the green channel, and sB for the blue channel. In mathematical and programming terms, this means that for the ith pixel, where (R_i, G_i, B_i) is the original pixel color and (R_i', G_i', B_i') is the color-corrected pixel color, we compute

$$R_i' = s_R R_i \qquad G_i' = s_G G_i \qquad B_i' = s_B B_i \quad (1)$$

The idea is to determine these correction factors such that a color that is gray in reality but tinted in the image is adjusted to a shade of gray without changing its brightness. One could come up with any number of ways to set these correction factors, and for this assignment we'll try three of them.

1. **Spot Method (solution given in source code)**

   For this method, we take an image location that we know should be a shade of gray, and determine correction factors for the entire image that make this location gray. Say that R0;G0;B0 is the color at the chosen image location. We can make the "target" gray value c, i.e. the shade of gray we'd like to adjust this chosen color to, be the average intensity of this chosen color:

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

   We can now solve for the scale factors in Eq. 1. The corrected color we'd like is the gray level c for each channel, thus substituting in Equation 1:

   $$c = s_R R_i \qquad c = s_G G_i \qquad c = s_B B_i$$

Then divide by the chosen color to get the scaling factors:

$$s_R = \frac{c}{R} \qquad s_G = \frac{c}{G} \qquad s_B = \frac{c}{B}$$

Next, as described above, we just apply this correction to every pixel in the image, using the same correction factors, using Equation 1.

2. Max Channel Method

The max channel method is very similar, but instead of obtaining the original color from a single spot in the image, we instead obtain it by individually computing the maximum value of each channel over all pixels in the image.

For each i^{th} pixel, compute the maximum value in the original image as:

$$R_{max} = \max_i R_i \qquad G_{max} = \max_i G_i \qquad B_{max} = \max_i B_i$$

Then compute the target gray level similarly as before:

$$c = \frac{1}{3}(R_{max} + G_{max} + B_{max})$$

and the scale factors similarly to before:

$$s_R = \frac{c}{R_{max}} \qquad s_G = \frac{c}{G_{max}} \qquad s_B = \frac{c}{B_{max}}$$

Again, then apply this correction to every pixel in the image using Equation 1.

3. "Gray World" Method

The "gray world" method is again very similar. This time we obtain the original color as the average of all pixels in the image. The idea is that on average, the color of the scene should be gray.

This time, compute the average color of the original image:

$$R_{mean} = \frac{1}{N}\sum_i R_i \qquad G_{mean} = \frac{1}{N}\sum_i G_i \qquad B_{mean} = \frac{1}{N}\sum_i B_i$$

where N is the number of pixels in the image. Then, compute the target grey level similarly to before as:

$$c = \frac{1}{3}(R_{mean} + G_{mean} + B_{mean})$$

and the scale factors as:

$$s_R = \frac{c}{R_{mean}} \qquad s_G = \frac{c}{G_{mean}} \qquad s_B = \frac{c}{B_{mean}}$$

Again, then apply this correction to every pixel in the image using Equation 1.

Discussion: You will submit a write-up with your code. In this write-up you will create a 3x2 table of pictures where the 3 rows correspond to the 3 different white balance techniques and the 2 columns are the 2 example pictures we gave you in the 'data' folder, white balanced with the corresponding method. Note that in both Microsoft Word and Google Docs, one way to create this table is to simply drag images onto the page, two per text row and resize them to fit next to one another. Fit this table onto 1 page. Create another page with another table, this time using the 2 photographs you took. On a 3rd page, briefly discuss the following:

1. Which technique worked best and why?
2. How do you judge the quality of the result?
3. Is there any variation among images in which technique works best?

3.3 Source code

This Processing sketch works so that the user presses “1” to use the Spot Method and clicks the mouse to select the reference location, presses “2” to correct using the Max Method, and presses “3” to correct using the Gray World Method. At any time, pressing “s” saves the corrected image in the same directory as the sketch.

You should modify the source code as explained in the description and comment your code (include your name in the header). The source code is written in Processing. Visit “Processing.org/reference/” for more information on built in functions and structure. Please not that **you are not allowed to use built in processing functions** to accomplish the tasks listed in the project description. We want you to access and directly manipulate the pixels in question to perform each task. When in doubt, ask.

3.4 Authorship Rules

The code that you turn in entirely your own. You are allowed to talk to other members of the class and to the Professor and the TA about general implementation of the assignment. It is, for example, perfectly fine to discuss how one might organize the data for a matrix stack. It is also fine to seek the help of others for general Processing/Java programming questions. You may not, however, use code that anyone other than yourself has written. Code that is explicitly not allowed includes code taken from the Web, from books, from previous assignments or from any source other than yourself. The only exception to this rule is that you should use the GT Graphics Library routines and the test code that we provide. You may NOT use other library routines for matrices and stacks. You should not show your code to other students. Feel free to seek the help of the Professor and the TA's for suggestions about debugging your code.

3.5 Submission

Your two unedited pictures should be located in the data folder where the default photos also reside. Your write-up in PDF format should be in the same folder as your code. In order to run the source code, it must be in a folder named after the main file. When submitting any assignment, leave code in this folder, compress it and submit via T-square.