CSCI435/CSCI935
Computer Vision: Algorithms and Systems

Spring 2018

Assignment One (15%)

Due Date: 11:59pm 19 August 2018

Objectives

- Getting familiar with OpenCV 3.4.1
- Reading, processing and displaying images using OpenCV 3.4.1
- Understanding the principles of single sensor digital cameras

Task One (5%)

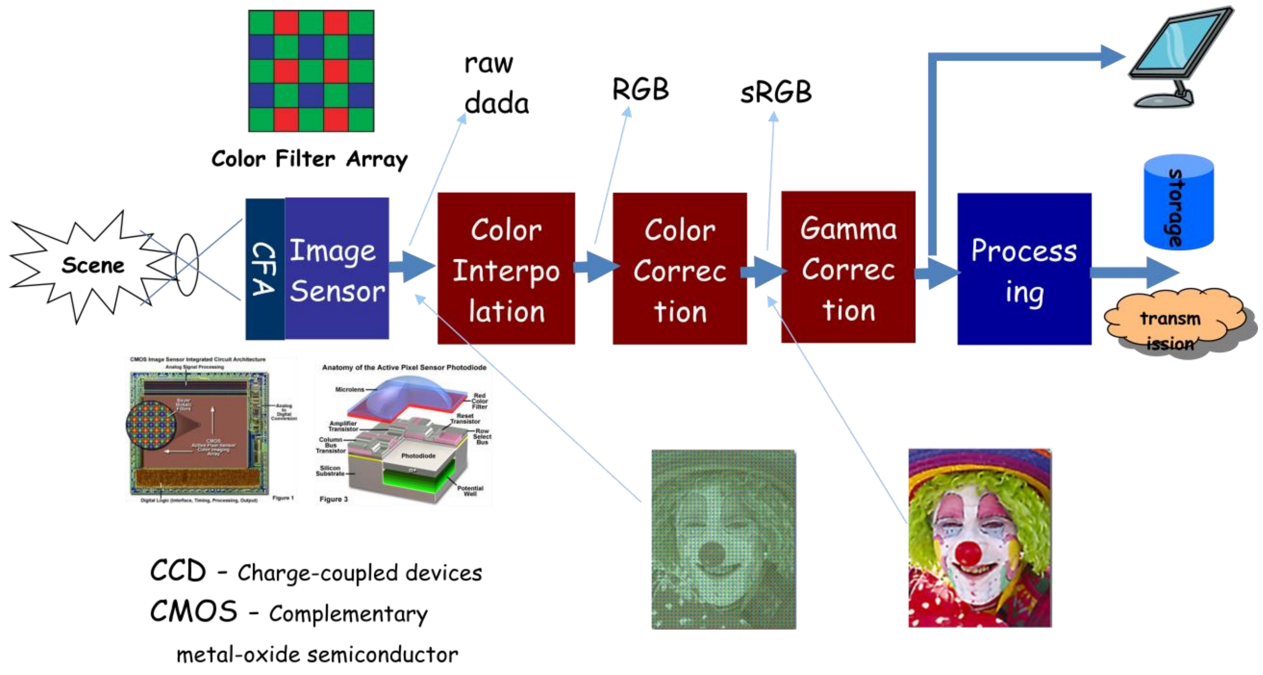
Many color spaces are available to represent color image pixel values. Some are perceptually uniform and others are not. This task is to read an image and display the original image in color and its components in different color spaces, such as CIE-XYZ, CIE-Lab, YCrCb and HSB, in gray-scale in **a single window** as illustrated below.

Original Image	**C1 (e.g. X, L, Y or B)**
C2 (e.g. Y, a, Cr or H)	**C3 (e.g. Z, b, Cb, or S)**

Note that you may need to scale the color component values properly so as to display the components in gray-scale images whose pixel values are within 0 to 255.

Task Two (10%)

Images captured by semiconductor sensors have to be processed before they are passed to later stages in computer vision systems. In this assignment, you are required to **develop a C/C++ program that implements the color image processing chain converting raw image data captured by a CMOS image sensor into true color RGB images.** The chain consists of three components: **color interpolation, color correction and gamma correction** as shown below**.**



You are provided with two bmp files `test1.bmp`, `test2.bmp and test3.bmp`, which contain the raw images (Bayer Pattern Color Filter Array data) directly captured by a CMOS image sensor. The Bayer pattern used in the CMOS sensor is shown below:

🟩	🟥	🟩	🟥	…
🟦	🟩	🟦	🟩	…
🟩	🟥	🟩	🟥	
🟦	🟩	🟦	🟩	
				…

Color Interpolation - You are required to implement the **bilinear color interpolation algorithm** to produce an RGB image with the ***same width and height*** as the input raw image. Therefore, you need to take into account boundary conditions, as pixels around the image boundary may not have neighboring pixels on one or two sides. *Try to find the most efficient and simple solution to interpolate boundary pixels*.

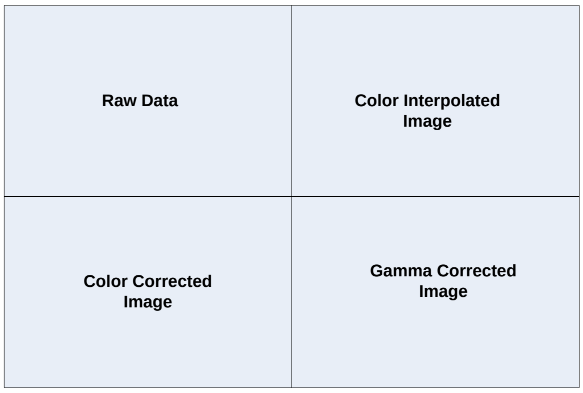
Color Correction - Color Correction is a matrix operation. You should use the following matrix that was optimized for this CMOS image sensor:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} 1.18 & -0.05 & -0.13 \\ -0.24 & 1.29 & -0.05 \\ -0.18 & -0.44 & 1.62 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Matrix multiplication can result in output values <0 or >255. Thus, you need to check the values and clip them to 0 or 255 if needed.

Gamma Correction - Gamma correction should be implemented using a look-up table. The table entries must be calculated using $\gamma = 0.5$ and implement the transfer function to achieve Gamma correction. You need to find the expression how to calculate and fill the table.

The program should display the raw image (data), color interpolated image, color corrected image and the gamma corrected image **in a single window** as illustrated below.



Requirements on coding

1. The program should be named as "**colorChain**" and shall take an image file as the input image with/without an option `-XYZ, -Lab,-YCrCb or -HSB` for different tasks, e.g.
   ```
   colorChain -XYZ|-Lab|-YCrCb|-HSB imagefile    # task one
   colorChain bmpfile                            # task two
   ```
2. No other third-party libraries should be used in the program except OpenCV 3.4.1. The code has to be in C/C++.
3. The code should be modularized with detail comments AND all source code should be placed in a single file "`colorChain.cpp`" or "`colorChain.c`".

Marking Scheme

1. Zero marks may be graded if your code cannot be compiled.
2. Program structure, comments and usability (3%)
   Specific comments on how you scale the color components to gray-scales must be included in the beginning of your source C/C++ file.
3. Display of the raw image and color components (5%)
4. Generation and display of the color interpolated image (3%)
5. Generation and display of the color corrected image (2%)
6. Generation and display of the gamma corrected image (2%)

Submission

Zip the **SOURCE** file to ***your_login_name*.zip**. The zip file has to be submitted in Moodle

IMPORTANT:

a) ***DO NOT** include and submit any object files and images in the zip file. Your submission may not be accepted if you do so.*

b) Submission through email ***WILL NOT*** be accepted