HOMEWORK 4
CAMERA ISP & JPEG DEVELOPMENT

CSED551 – COMPUTATIONAL PHOTOGRAPHY
FALL 2024
SUNGHYUN CHO (S.CHO@POSTECH.AC.KR)

CAMERA ISP[1] & JPEG DEVELOPMENT[2]

Most digital cameras provide captured images as JPEG files. However, DSLR and mirrorless cameras and some high-end smartphone cameras also provide unprocessed data, to which none of image processing operations such as white balance, and demosaicing are applied. Such unprocessed data are called RAW images and provided in camera vendors' own formats, e.g., CRW (Canon RAW format), ARW (Sony Alpha Raw), and Adobe DNG (Adobe Digital Negative Format).

Compared to JPEG images, RAW images provide several benefits:

- They allow users to adjust image processing parameters in a more delicate way.

    - E.g., you can change the white balance, tone and color manually using software tools such as Adobe Photoshop and Adobe Lightroom.

- JPEG compression is a lossy compression algorithm, which means that some information is lost after compression. In contrast, RAW images provide uncompressed original data.

- RAW files usually use 12 or 14 bits to store light intensities, while JPEG files use 8 bits. Thus, RAW files provide richer information. This allows users to obtain JPEG images with different exposure settings from a RAW image even after the image was captured.

Due to these benefits, many professional photographers often capture images using RAW images, and then later develop JPEG images from them to produce high-quality results.

In this assignment, you will implement your own version of a very basic image signal processing pipeline for the development of a JPEG image from a RAW image. Specifically, you need to implement:

- White balance

---

[1] The term "image signal processor" is also often used to refer to the image processing pipeline to develop a JPEG image from a RAW image.

[2] The term "development" comes from the analogy with the process of developing film into a photograph. The word is translated to a Korean word "인화". (사진을 인화하다.)

- CFA interpolation

- Gamma correction

These three are the most essential operations to develop a JPEG image from a RAW image. More details about white balance and gamma correction will be provided later. For white balance, you need to implement an automatic white balance (AWB) algorithm based on the gray world assumption, which is described in the following.
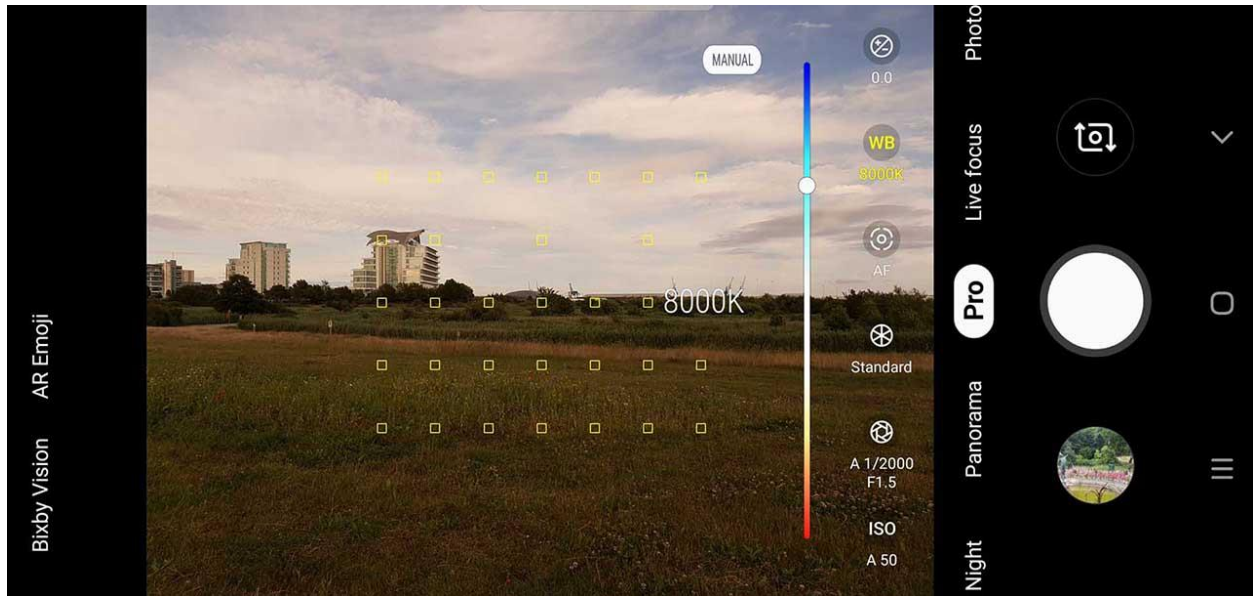
WHITE BALANCE

Suppose that you are in a room with a yellowish lamp and some white objects. The lamp radiates yellow light rays, and some of them are reflected on the surface of the objects and go to your eyes, so you can perceive the objects. The reflected light on the surface is yellowish because it is a combination of the yellow light radiated from the lamp and the color of the surface of the objects. Nevertheless, we can still recognize the color of the white objects as white because our brain automatically cancels out the color of the light source. This is called color constancy. The color constancy will be covered in more detail later in one of the classes.

Then, suppose that you take a picture of the white objects in the room. Your camera will simply capture the incoming light, so the objects in the picture will be yellowish, making the picture look unnatural. To avoid this, cameras also try to cancel out the color of the illuminant and adjust the colors of objects. This processing is called white balance. If the camera automatically figures out the parameters for white balance, it is called automatic white balance. White balance can be easily implemented using a simple transform, which can be expressed as follows:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} s_G/s_R & & \\ & 1 & \\ & & s_G/s_B \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{1}$$

In this equation, (R,G,B) are the color components at each pixel of an input image, and (R',G',B') are the color components after white balance. (S_R, S_G, S_B) is the color of the illuminant. Most cameras provide users a way to manually change the white balance parameters (S_R, S_G, S_B). For example, the figure below shows a screenshot of white balance setting of the camera app of Samsung Galaxy S10.

Manual white balance can be done either choosing a camera preset based on lighting or using an object in photograph whose color is white. Specifically, most cameras provide presets to users as shown below. On the other hand, when there is a white object, we can figure out the color of the illuminant from the captured color of the white object. Then, we can cancel out the color of the illuminant from the captured image using the equation (1).

WB SETTINGS	COLOR TEMPERATURE	LIGHT SOURCES
	10000 - 15000 K	Clear Blue Sky
	6500 - 8000 K	Cloudy Sky / Shade
	6000 - 7000 K	Noon Sunlight
	5500 - 6500 K	Average Daylight
	5000 - 5500 K	Electronic Flash
	4000 - 5000 K	Fluorescent Light
	3000 - 4000 K	Early AM / Late PM
	2500 - 3000 K	Domestic Lightning
	1000 - 2000 K	Candle Flame

Automatic white balance automatically finds the white balance parameters. One of the most widely used algorithm for auto white balance is the gray world assumption algorithm, which is based on the gray world assumption that assumes that the average color of the world is gray. Specifically, the algorithm performs automatic white balance using the following equation:
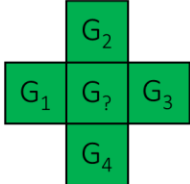
$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} G_{avg}/R_{avg} & & \\ & 1 & \\ & & G_{avg}/B_{avg} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where R_{avg}, G_{avg} and B_{avg} are the average intensity of the red, green and blue color channels, respectively.

CFA INTERPOLATION

CFA interpolation is one of the most important step for the visual quality of images. For more effective CFA interpolation, a countless number of approaches have been proposed including recent deep learning-based approaches. In this homework, you can implement a simple algorithm for CFA interpolation such as bilinear interpolation. Bilinear interpolation can be implemented by simply computing the average of 4 neighbors as shown below.

Bilinear interpolation: Simply average your 4 neighbors.

$$G_? = \frac{G_1 + G_2 + G_3 + G_4}{4}$$

Neighborhood changes for different channels:
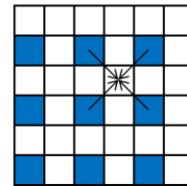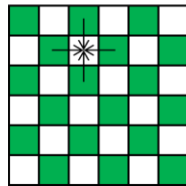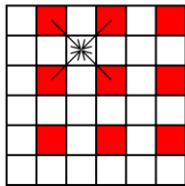
GAMMA CORRECTION

Cathode ray tube (CRT) displays, which were widely used in the past, had a non-linear response curve, i.e., the images displayed on them had non-linearly distorted brightness. Specifically, images are displayed darker on CRT displays as shown below.

Original image   Displayed on CRT

The response curve of a CRT display is often approximated as:

$$I' = I^{\gamma}$$

where $\gamma = 2.2$ is a constant. Thus, to resolve this issue, gamma correction was introduced, which transforms pixel values by

$$I' = I^{1/\gamma}$$

Gamma correction is also called gamma encoding when the exponent ($1/\gamma$ in the equation above) is smaller than 1.

Gamma correction was originally applied to digital images to correctly show the images on CRT displays. Even though CRT displays are no longer used these days and LCD displays do not have this issue, all digital images used in computers must be still gamma encoded to be properly displayed. The figure below describes the effect of gamma correction.

| sensor: linear curve | Gamma encoding | Response curve of CRT display | net effect: linear curve |
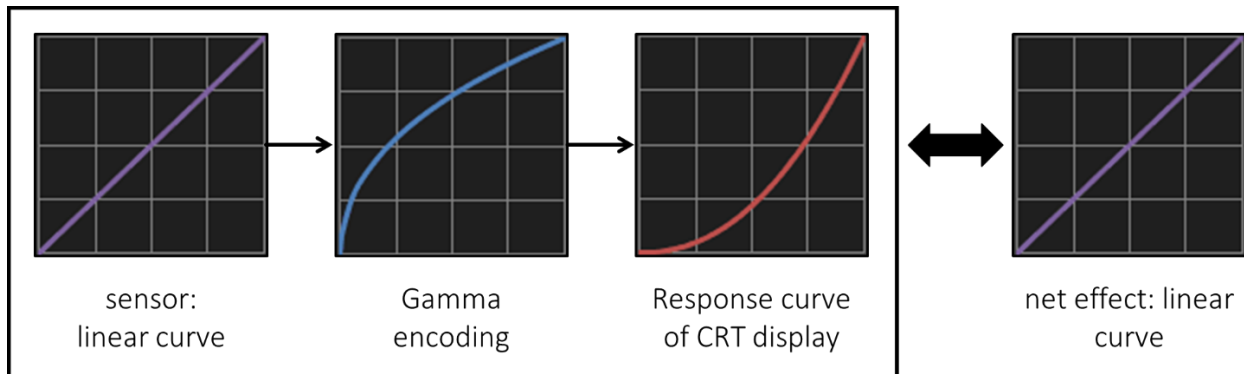
Another benefit of using gamma correction is that it better reflects the human perception. The human perception is non-linear to the brightness, i.e., human eyes are more sensitive to differences between dark tones than bright tones. Thus, to better reflect this, we can non-linearly assign bits to the brightness, i.e., assign more bits to dark tones using gamma encoding.
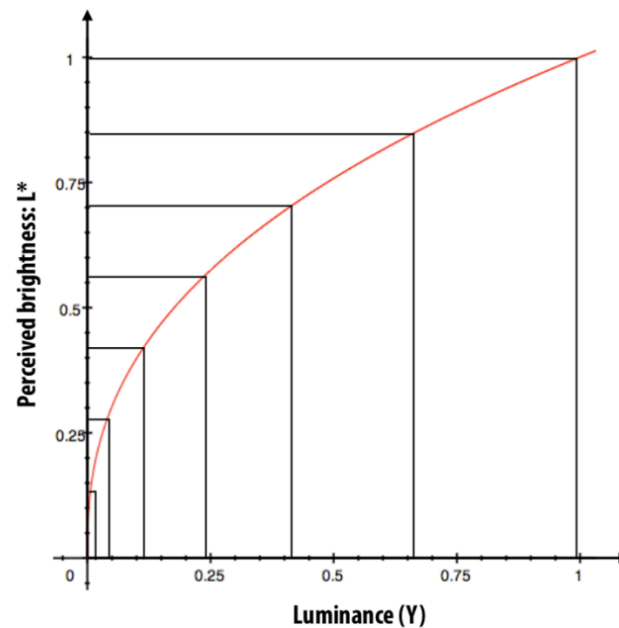
DECODING CAMERA RAW FILES

Camera vendors use their own proprietary formats to store RAW images, and you cannot directly read such RAW images using typical image libraries such as OpenCV. Instead, there is a command-line tool called dcraw[3]. Using this tool, you can convert a camera RAW file into a .tiff file, which you can read using OpenCV or in Matlab. After having installed dcraw, you can run the following command:

```
dcraw -4 -d -v -T [RAW_filename]
```

While dcraw also provides functions for producing JPEG images from RAW images, in this homework, you use the tool only for extracting raw data.

REQUIREMENTS

You need to implement a command-line tool for JPEG development from a RAW image. Specifically, your program takes a .tiff file as input and produces a .jpg file as described below.

```
# ./tiff2jpg.py image.tiff
# ls
image.tiff    image.jpg
```

---

[3] https://www.dechifro.org/dcraw/

Your program must perform the three operations listed below:

- Automatic white balance (gray world assumption)

- CFA interpolation

  - You can implement any simple algorithm, e.g., bilinear interpolation

- Gamma correction

You also need to implement additional operations of your own that make images similar to the images produced by commercial JPEG development tools such as native camera ISPs, Adobe Photoshop, or Adobe Lightroom, or make images visually more pleasing. The points will be given according to the final image quality.

Two RAW images are provided. You need to produce JPEG images from them and discuss their results in your report. You also need to take RAW images using your own camera and include their results too.

You need to write a report that includes:

- Detailed discussion on your implementation

- Detailed discussion on the results of each step

- Detailed discussion on the final results of your program compared to other methods for JPEG development (e.g., dcraw, native camera ISPs, Adobe Photoshop or Lightroom,)

You must upload a single zip file that contains the following to the LMS:

- code/ - a directory containing all your code for this assignment

- images/ - a directory containing your input images and their results

- report.pdf – your report as a PDF file

You can use any programming language for your implementation. I recommend you to use one of the followings:

- C++ and OpenCV - OpenCV is a powerful computer vision library that provides many useful features.

- Matlab - You will need to install Image Processing Toolbox

- Python - There are several useful libraries, e.g., a python binding of OpenCV, scipy, numpy, …

Due: Dec. 8th, 23:59

Penalty for late submission

- 1 day: 70%

- 2 days: 30%

- 3 days: 0%

RUBRIC

- 20 pts: White balance

- 20 pts: CFA interpolation

- 20 pts: Gamma correction

- 10 pts: Additional operations

- 30 pts: Report

Your homework will be scored based on your report, and I am not going to compile or run your code. Thus, your report must include all necessary details of your implementation and results.