Multimedia
(Lab 04)

Spring, 2020

Yong Ju Jung (정용주)

미래창조과학부 SW중심대학

가천대학교 소프트웨어학과

Gachon University Department of Software

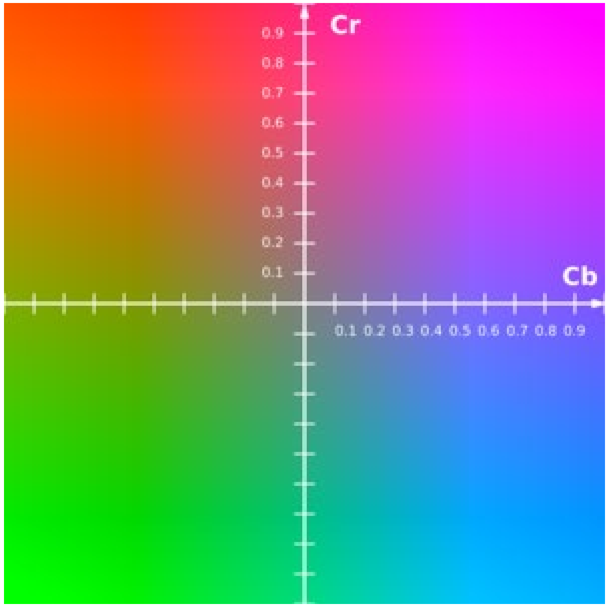
Summary

- In this lab, you will learn about
  - Simple color image transform and processing

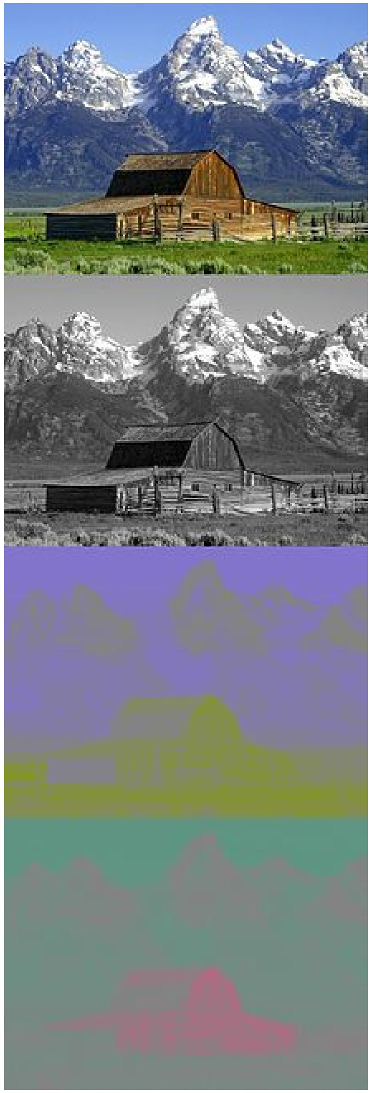
[Lab04-1]

- Color transform
  - Load a color Lena image (using **cv::imread**)
  - Do color transform from RGB to YCbCr, as shown in the next slide.
  - Display original RGB image & each channels of YCbCr as grayscale image

- You can refer to the following OpenCV library:
  - cvtColor(src, dst, CV_BGR2YCrCb);
  - Mat dst_Y = zeros(src.size(), CV_8UC1);



The CbCr plane at constant luma Y'=0.5



A color image and its Y, CB and CR components.

* Source from Wikipedia

5/4 1:50분 정도에 설명.

[Lab04-2]

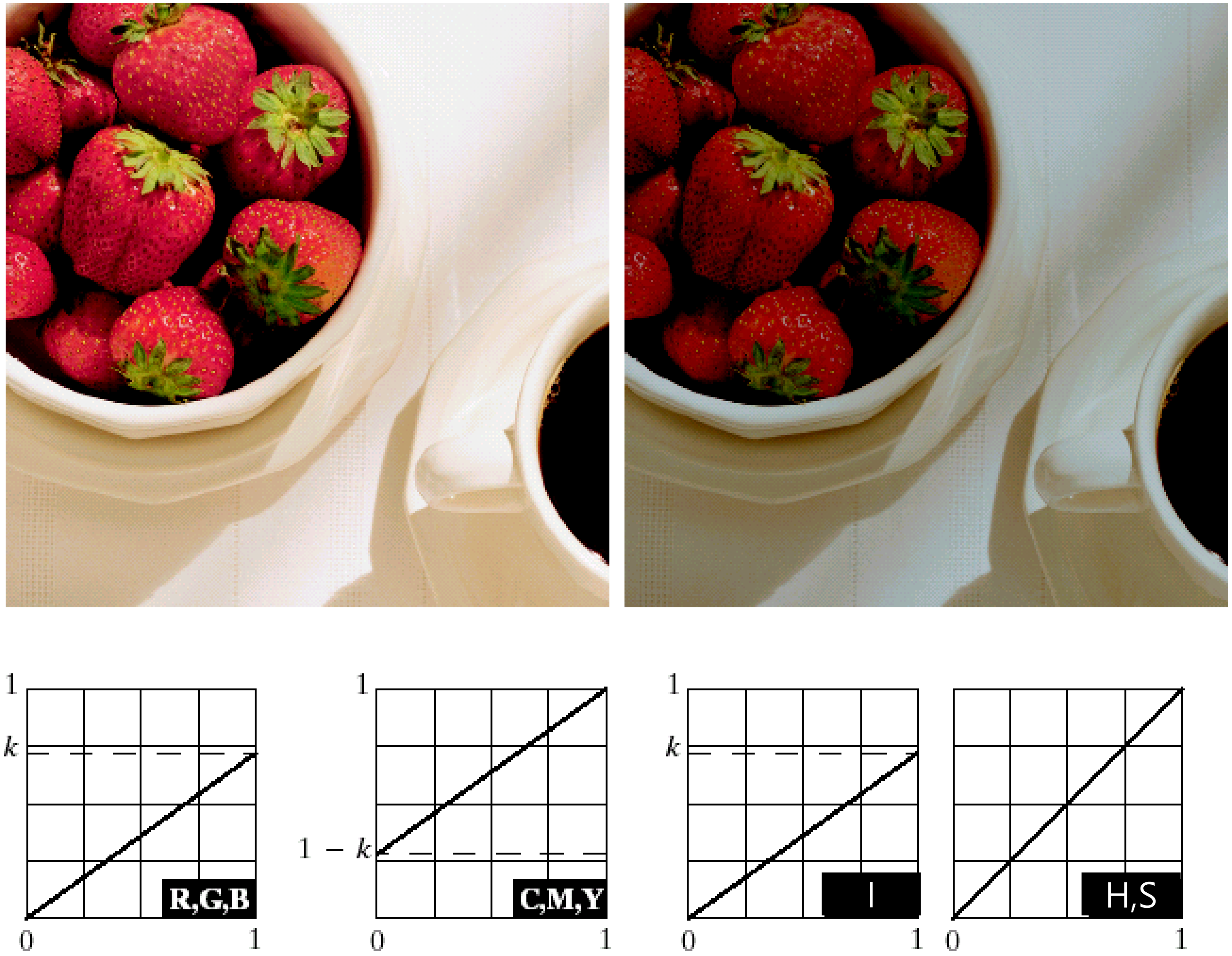
- Color transform
  - Load an image (using **cv::imread**)
  - Do color transform & modify intensity in various domains, as shown in the next slide (Fig. 6.31).
    - RGB
    - YCbCr
    - CMY
    - HSV (similar to HSI)

  - Display original & result images (using **cv::imshow**)

  - You can use the following OpenCV library:
    - cvtColor(src, dst, COLOR_BGR2YCrCb); //RGB컬러를 yCrcb로 변환.
  - However, you have to write your own code for RGB to CMY conversion.

색깔은 알아서 변환해줘야함.

Color Transformation



a b
c d e

FIGURE 6.31 Adjusting the intensity of an image using color transformations. (a) Original image. (b) Result of decreasing its intensity by 30% (i.e., letting $k = 0.7$). (c)–(e) The required RGB, CMY, and HSI transformation functions. (Original image courtesy of MedData Interactive.)

Various Representation of **Lightness**

- I (intensity in HSI)
  - (R+G+B)/3
- V (value in HSV)
  - Max(R, G, B)
- Y (luminance in YCbCr)
  - 0.30R + 0.59G + 0.11B

Notes on cvtColor

- RGB <-> HSV (CV_BGR2HSV, CV_HSV2BGR)
  - cvtColor(src, dst, COLOR_BGR2YCrCb);

BGR을 HSV로 변환.
imshow()할때는 BGR이여야 하므로 다시 BGR로 변환해줘야함.

http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html?highlight=cvtcolor

```
//RGB Color Space에서 HSI Color Space로 변환
void rgb2hsi(Mat& RGB_image, Mat& HSI_image){
    vector<Mat> RGB_image_components, HSI_image_components;
    for (int i = 0; i < 3; i++){
        HSI_image_components.push_back(Mat(RGB_image.size(), CV_8UC1));
    }
    split(RGB_image, RGB_image_components);

    for (int i = 0; i < RGB_image.rows; i++){
        for (int j = 0; j < RGB_image.cols; j++){
            float r = RGB_image_components[2].at<uchar>(i, j);
            float g = RGB_image_components[1].at<uchar>(i, j);
            float b = RGB_image_components[0].at<uchar>(i, j);
            float hue, saturation, intensity, min_val;

            intensity = (r+g+b)/(3.0);
            min_val = min(r, min(g, b));
            if (intensity > 0.0)
                saturation = 1 - (min_val / intensity);
            if (saturation < 0.00001){
                saturation = 0;
            }
            else if (saturation > 0.99999){
                saturation = 1;
            }
            if (saturation > 0){
                hue = (0.5 * ((r - g) + (r - b))) / sqrt(((r - g) * (r - g)) + ((r - b) * (g - b)));
                hue = acos(hue);
                if (b > g){
                    hue = ((360 * PI) / 180.0) - hue;
                }
            }
            else{
                hue = 0;
            }

            HSI_image_components[2].at<uchar>(i, j) = intensity;
            HSI_image_components[1].at<uchar>(i, j) = saturation * 100;
            HSI_image_components[0].at<uchar>(i, j) = (hue * 180) / PI;
        }
    }
    merge(HSI_image_components, HSI_image);
}
```