Multimedia
(Lab 02)

Spring, 2020

Yong Ju Jung (정용주)

가천대학교 소프트웨어학과
Gachon University

Summary

- In previous lab, you have installed Visual C++ and OpneCV library in your computer.

- In this lab, continue to the basic practice.
  - You will learn about simple pixel transforms in images.

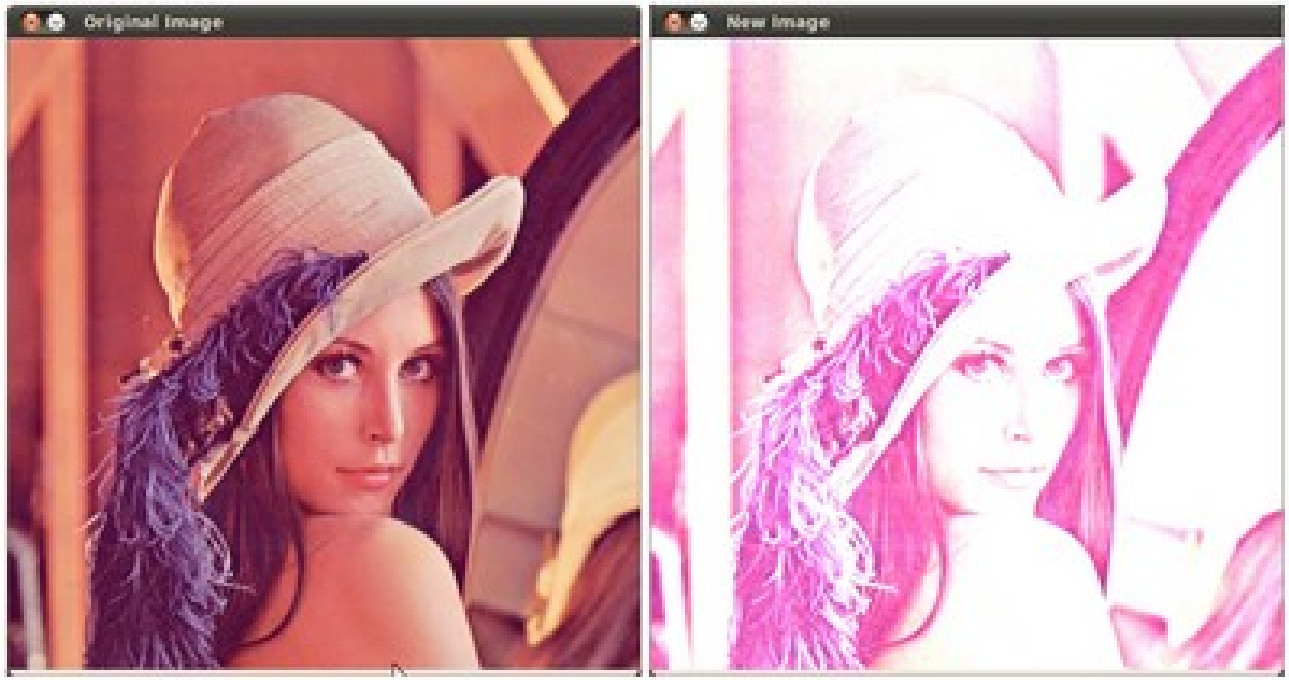
[Lab02-1]

- A general image processing operator is a function that takes one or more input images and produces an output image.

- Simple pixel transform: We will learn how to change our image appearance.
  - Load an image (using **cv::imread**)
  - Change its contrast and brightness of an image
    - $g(x) = \alpha * f(x) + \beta$ //알파를 조정하면 contrast를 조정함. 베타는 밝기?
    - The parameters $\alpha>0$ and β are often called the *gain* and *bias* parameters; sometimes these parameters are said to control *contrast* and *brightness* respectively.
  - Display the result image in an OpenCV window (using **cv::imshow**)

Example results

- using α=2.2 and β=50



* Source from OpenCV tutorial

Some useful function

- **Saturation Arithmetic**
  - OpenCV deals a lot with image pixels that are often encoded in a compact, **8- or 16-bit per channel**, form and thus have a limited value range.
  - Furthermore, certain operations on images, like color space conversions, brightness/contrast adjustments, sharpening, complex interpolation can **produce values out of the available range**.
  - If you just store the lowest 8 (16) bits of the result, this results in visual artifacts and may affect a further image analysis.
  - To solve this problem, the so-called *saturation* arithmetics is used.
  - For example, to store *r*, the result of an operation, to an 8-bit image, you find the nearest value within the 0..255 range:
    - **I**(*x*,*y*)=min(max(round(*r*),0),255)
-

```
I.at<uchar>(y, x) = saturate_cast<uchar>(r);
```

//r값을 0~255 사이로 잘라서 uchar타입으로

- To access each element of an image matrix, use
  - .at<data type>(y, x)

    //순서가 행렬이라서 y,x 순서로 거꾸로 옴. (x.y)에 접근하려면 이렇게 써줘야함
  - For example, in case of gray image
    - Mat gray_img(Size(1920, 1080), CV_8UC1);
    - int value = gray_img.at<uchar>(100, 200);
  - In case of color image
    - Mat color_img(Size(1920, 1080), CV8UC3); //8비트 채널에 3개의 컬러?
    - int blue = color_img.at<Vec3b>(y, x)[0]; //vec3b라는 3바이트짜리 벡터에서 0,1,2의 3차원 인덱스로 각 색깔 접근
    - int green = color_img.at<Vec3b>(y, x)[1];
    - int red = color_img.at<Vec3b>(y, x)[2];

Useful Tip for Debug

- **Image Watch: viewing in-memory images in the Visual Studio debugger**
  - You will learn how to visualize OpenCV matrices and images within Visual Studio 2012 (or better).

To bring up Image Watch, select View –> Other Windows –> Image Watch.

[Lab02-2]

- Linear blending of two images
  - Load two images (using **cv::imread**)
  - Do linear blending of the two images
    - $dst(x,y) = \alpha * src_1(x,y) + \beta * src_2(x,y)$
    - where $\beta = 1 - \alpha$.
    - cv::addWeighted(src1, α, src2, β, γ, dst)
  - Display the result image in an OpenCV window (using **cv::imshow**)

    - By varying α from 0→1 this operator can be used to perform a temporal cross-dissolve between two images