**고급 소프트웨어 실습 1주차 과제**

**컴퓨터공학과 20181593 계인혜**

1. 모든 컬러 값들을 grayscale로 변환해주는 공식에 대입하였다. 이는 특정 컬러 값이 많이 나타나는 이미지에서 불필요하게 동일한 연산을 반복하는 결과를 불러온다. 이러한 문제를 해결할 수 있는 효율 적인 컬러 변환 방법에 대하여 기술한다.

픽셀 수가 매우 큰 사진을 grayscale로 변환한다고 생각해보자. 하나의 픽셀을 grayscale로 변환하는 데에 float 곱셈이 3번, 덧셈이 2번이다. 실습 방식대로 코드를 실행시키면 픽셀 수 \* 5만큼의 연산이 필요하다.

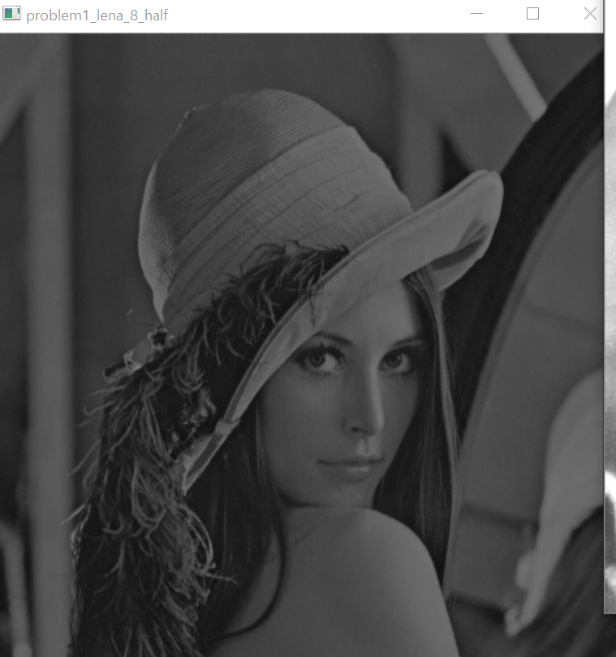
만약 픽셀 수가 10억 개라면 대략 50억번이 연산이 필요한 것이다. 그러나, rgb가 표현할 수 있는 픽셀의 경우의 수는 256^3 = 16,777,216으로 이 경우 동일한 연산을 여러 번 반복하게 될 것이다. 이를 막기 위해서 미리 rgb 값의 모든 경우를 계산하여 저장해두어 변환 시에 rgb 값을 더하고 shift연산을 이용하여 곱셈 처리를 해서 미리 저장해둔 배열에서 grayscale값을 불러오면 된다.

이 방법은 픽셀 수가 rgb가 표현할 수 있는 모든 경우의 수보다 큰 경우 반복 연산을 줄여줄 수 있지만, 픽셀 수가 상대적으로 작은 경우에는 적용할 수 없을 것이다.

1. Open source 라이브러리인 OpenCV에 컬러 변환을 수행하는 함수들이 지원된다. 이러한 함수를 사용하여 위에서 실습한 컬러 변환을 해보고, 어떤 경우에 OpenCV와 같은 툴의 사용이 제한될 수 있는지에 관하여 기술한다. 제공되는 OpenCVcolorConversion 프로젝트를 이용한다.
2. OpenCV 컬러 변환 함수를 사용하여 실습과 같은 컬러 변환하기

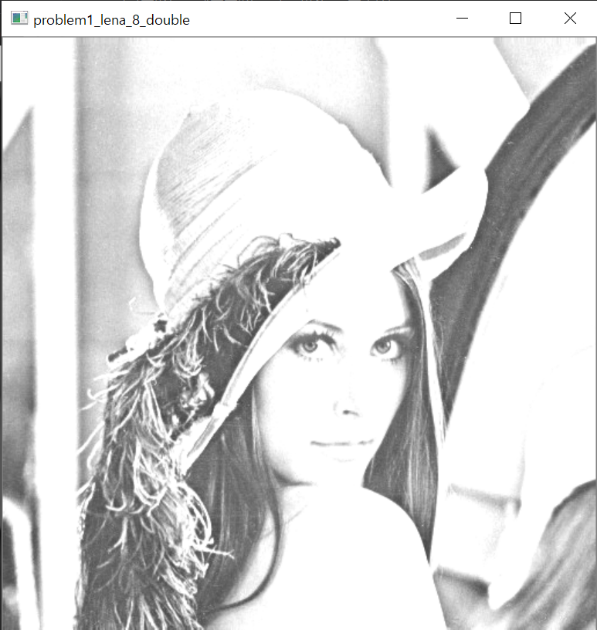
문제 1.1

addWeighted(img1, alpha, 0, beta, 0, img1\_Mat\_half);



문제 1.2

addWeighted(img1, alpha, 0, beta, 0, img1\_Mat\_double);

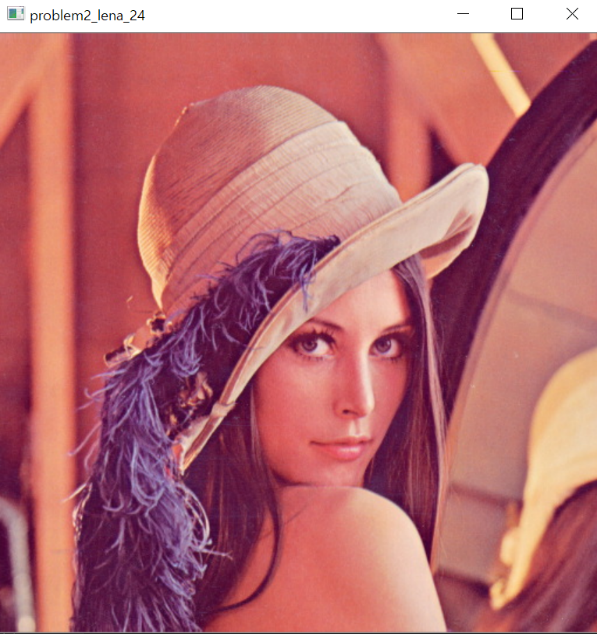


문제 2

cvtColor(img2, img2\_Mat\_gray, COLOR\_RGB2GRAY);

imshow("problem2\_lena\_24\_gray", img2\_Mat\_gray);

imwrite("problem2\_lena\_24\_gray.bmp", img2\_Mat\_gray);



문제 3

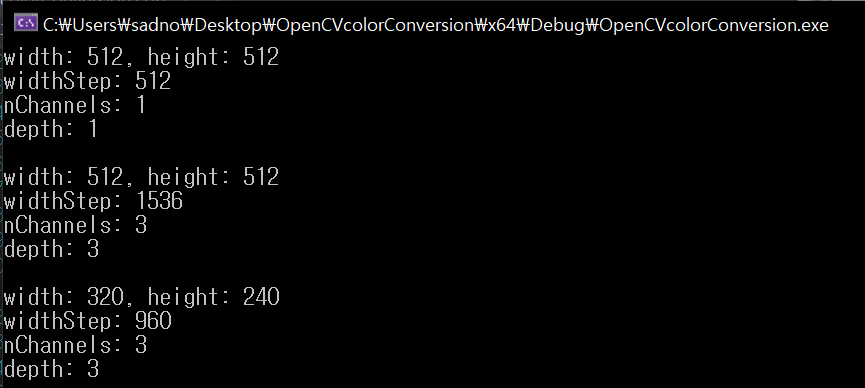
cvtColor(img3, img3\_Mat\_gray, COLOR\_RGB2GRAY);

imshow("problem3\_image\_16\_gray", img3\_Mat\_gray);

imwrite("problem3\_image\_16\_gray.bmp", img3\_Mat\_gray);



콘솔 창 출력



1. OpenCV 툴의 사용이 제한되는 경우

openCV를 사용하여 프로그램을 만들 경우 보통 DLL(Dynamic Link Library) 방식을 사용한다. DLL 방식을 사용하는 이유는 한 코드를 여러 프로그램이 동시에 사용하여 메모리 절약에 유리하고, 사용되는 디스크 공간을 줄일 수 있기 때문이다. 또한 모듈식 프로그램을 효율적으로 개발할 수 있으며, 배포와 설치가 용이하다.

그러나, 이 방식을 사용하면 프로그램을 배포할 때 openCV에 있는 DLL도 같이 보내야한다. DLL 파일의 크기가 큰 편이기 때문에 불편한 점이 생길 수 있다.

또한, 코드의 호환성 문제도 존재한다. 하위 버전에서 개발된 코드를 상위 버전에서 사용하기 위해 수정이 필요하기 때문이다.