

**과목명 | 디지털영상처리**

**실습 | 2**

**담당교수 | 정진우**

**학과 | 컴퓨터공학과**

**학년 | 3**

**학번 | 2010112406**

**이름 | 박도령**

**제출일 | 2016. 3. 31**

**실습에 사용한 이미지)**

**Lenna\_gray.PNG**

****

**Grayscale Image이다.**

**실습 1)**

**(1) Contrast & Brightness adjustment by LUT(Look Up Table)[Input : 8 bit gray-scale image]**

♋ **동작 순서 및 원리**

0. 이미지를 Matrix 객체 형태로 로드한다.

1. 밝기와 명암을 입력 받는다. 밝기의 범위는 (-255 ~ 255)이고 명암의 범위는 (-100 ~ 100)이다. 명암의 경우 명암 비율(ratio = 1 + contrast/100)을 계산한다.

2. Lookup Table Matrix(히스토그램)를 만든다. Matrix를 256의 공간을 가진 1차원 배열로 만든다. 인덱스는 해당 밝기를 뜻하고 인덱스 안의 원소는 해당 인덱스(밝기)에 매칭되어 변환되는 새로운 밝기 값이다.

3. Lookup Table의 내부를 Set한다. 입력한 밝기는 현재의 밝기에 더해주고 내부를 Set할 때, Clamping을 코드로 구현해 놓는다. LookUp Table의 내부 코드는 다음과 같다.

Mat lkuptab{ 1, 256, CV\_8U };//Lookup Table 생성

for (int i = 0; i < 256; i++) {//Lookup Table Set

if ((float)(i + bright) \* ratio > 255) //Clamping(255보다 큰 경우)

lkuptab.at<uchar>(i) = 255;

else if ((float)(i + bright) \* ratio < 0) //Clamping(0보다 작은 경우)

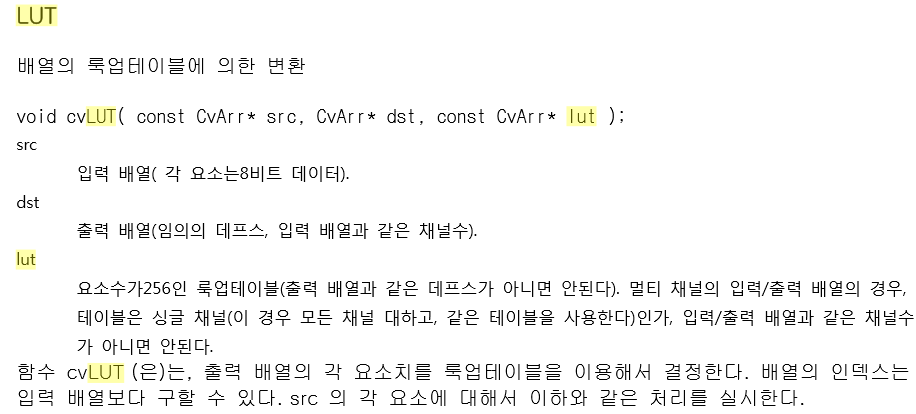
lkuptab.at<uchar>(i) = 0;

else //정상 경우 i + bright \* ratio = 현재 이미지에 입력한 밝기와 명암 적용

lkuptab.at<uchar>(i) = (float)(i + bright) \* ratio;

}

4. Opencv에서 원래 이미지에 LUT를 적용하여 결과 이미지를 생성할 수 있게 해주는 LUT 함수를 사용한다. 함수의 내용은 다음과 같다.



5. 결과 이미지를 윈도우에 출력한다.

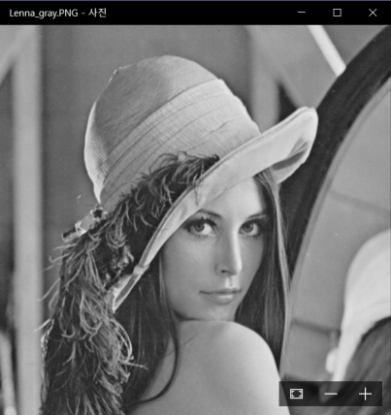
♋ **실행 결과 (ReportCode3-1.cpp)**

그림 1. Original Image

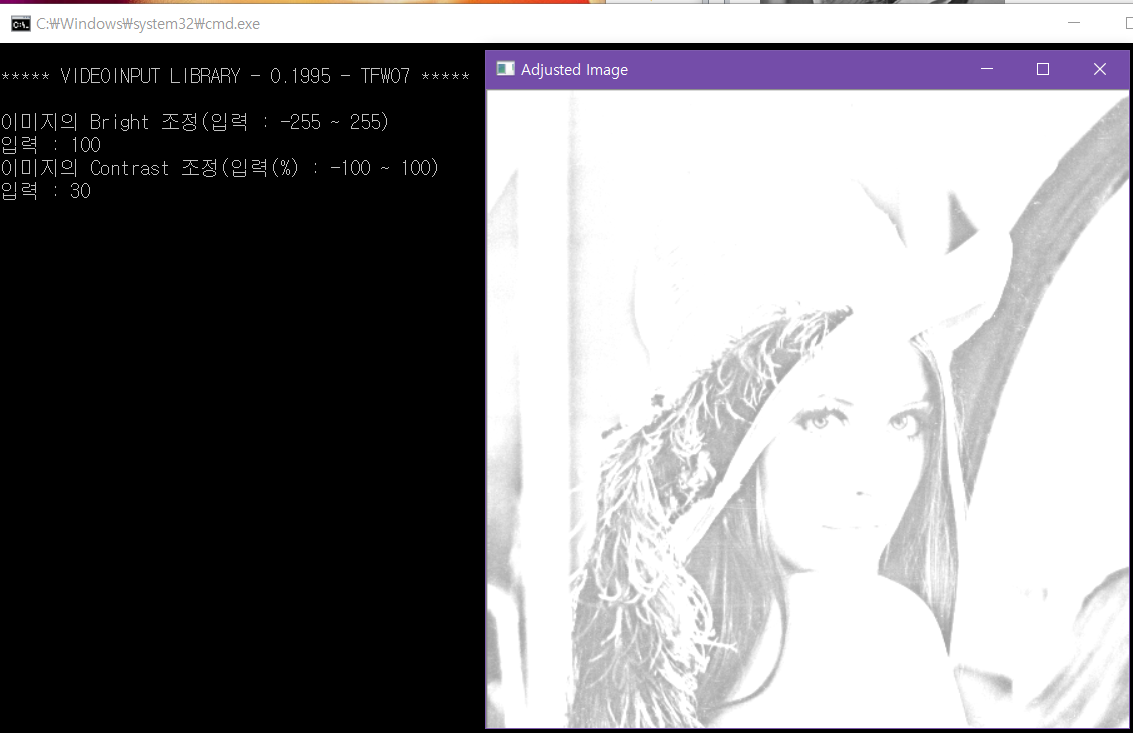


그림 2. Image Adjusted by Bright : 100, Contrast : 30

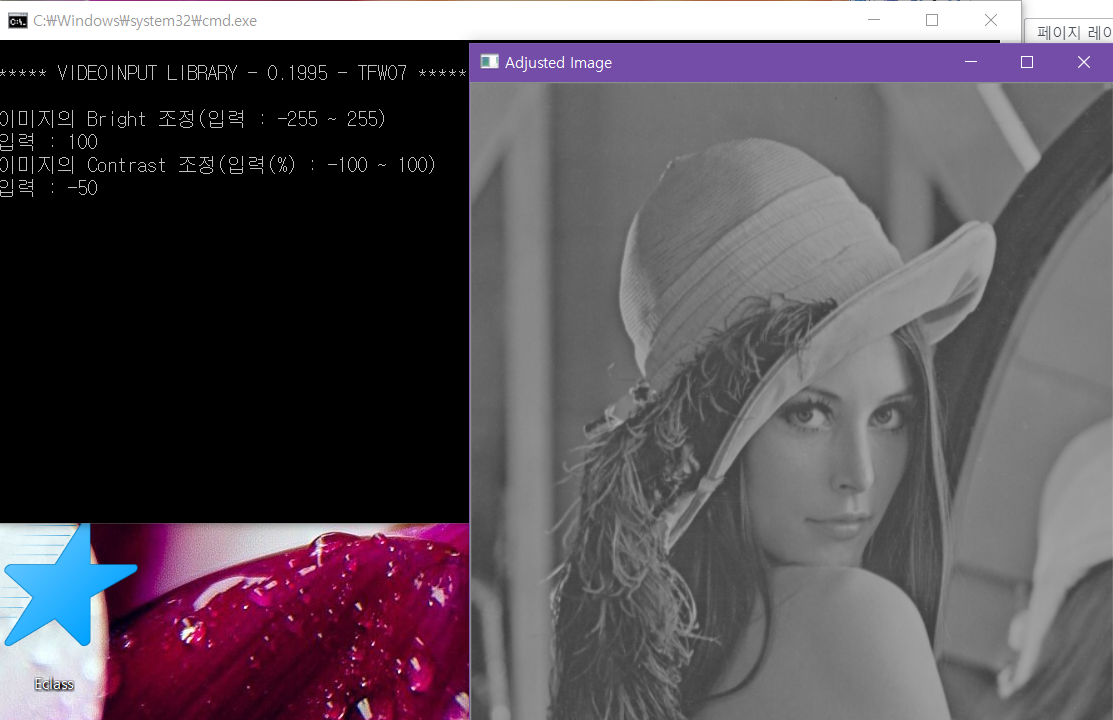


그림 3. Image Adjusted by Bright : 100, Constrast : -50

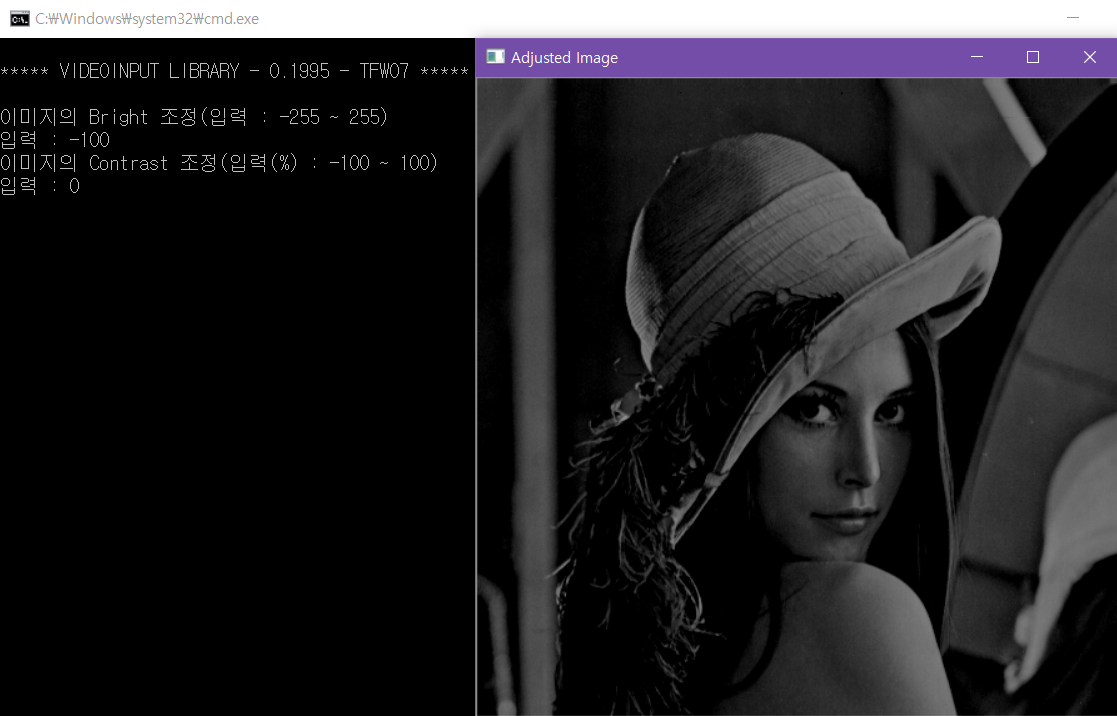


그림 4. Image Adjusted by Bright : -100, Contrast : 0

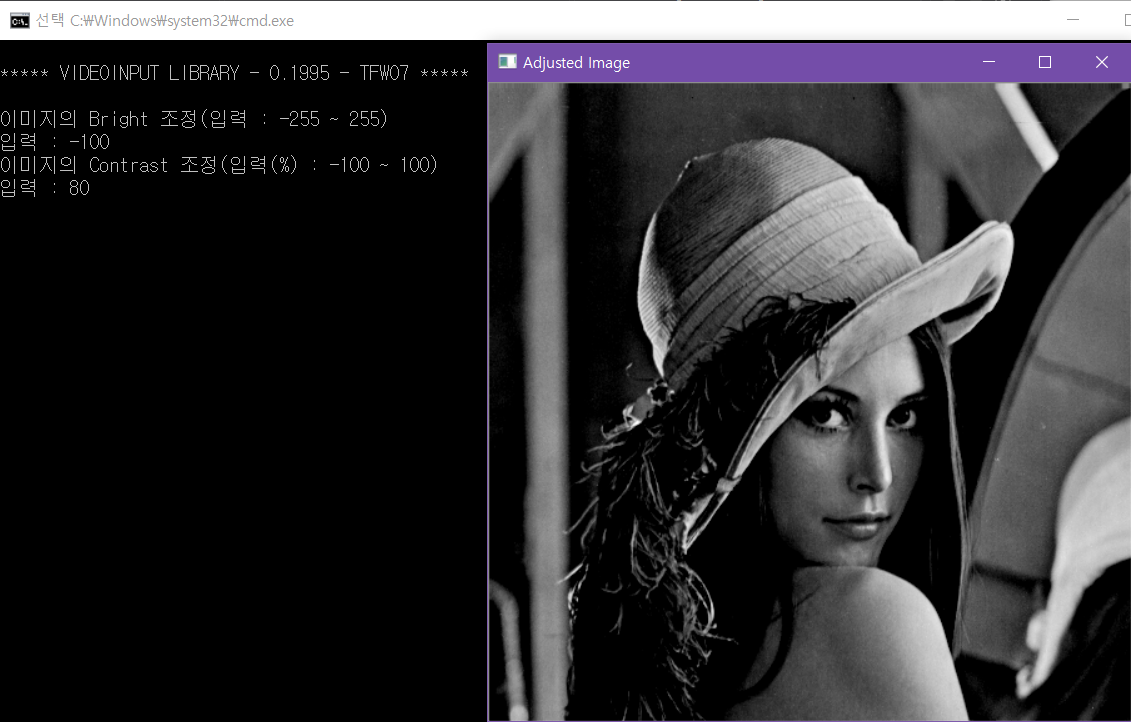


그림 5. Image Adjusted by Bright : -100, Contrast : 80

**실습 2)**

**(2) Contrast adjustment (Multiplcation)[Input : 8 bit gray-scale image]**

♋ **동작 순서 및 원리**

0. 이미지를 Matrix객체 형태로 로드한다.

1. 이미지의 명암 값을 입력 받는다. (-100 ~ 100) 명암 비율을 계산한다. (ratio = (1 + input / 100))

2. 이미지의 행과 열을 추출하여 이중 For문을 사용한다. (방향 : 가로 아래)

3. 이중 For문 안에서 한 픽셀에 대한 Red Channel, Green Channel, Blue Channel 값을 추출한다.

auto chBlue = (float)(image1.at<cv::Vec3b>(j, i)[0]); //이미지의 Blue Channel 추출

auto chGreen = (float)(image1.at<cv::Vec3b>(j, i)[1]); //이미지의 Green Channel 추출

auto chRed = (float)(image1.at<cv::Vec3b>(j, i)[2]); //이미지의 Red Channel 추출

4. 추출한 세 Channel의값에 명암 비율을 곱한다. 그 뒤 Clamping 처리를 한다.

auto ResultBlue = chBlue \* ratio; //Blue Channel 값에 입력한 비율만큼 곱한다.

auto ResultGreen = chGreen \* ratio; //Green Channel 값에 입력한 비율만큼 곱한다.

auto ResultRed = chRed \* ratio; //Red Channel 값에 입력한 비율만큼 곱한다.

//Clamping (모두 0으로)

if (ResultBlue < 0.0)

ResultBlue = 0.0f;

if (ResultGreen < 0.0)

ResultGreen = 0.0f;

if (ResultRed < 0.0)

ResultRed = 0.0f;

//Clamping (모두 255로)

if (ResultBlue > 255.0)

ResultBlue = 255.0f;

if (ResultGreen > 255.0)

ResultGreen = 255.0f;

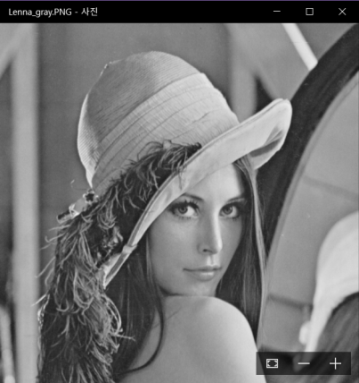
if (ResultRed > 255.0)

ResultRed = 255.0f;

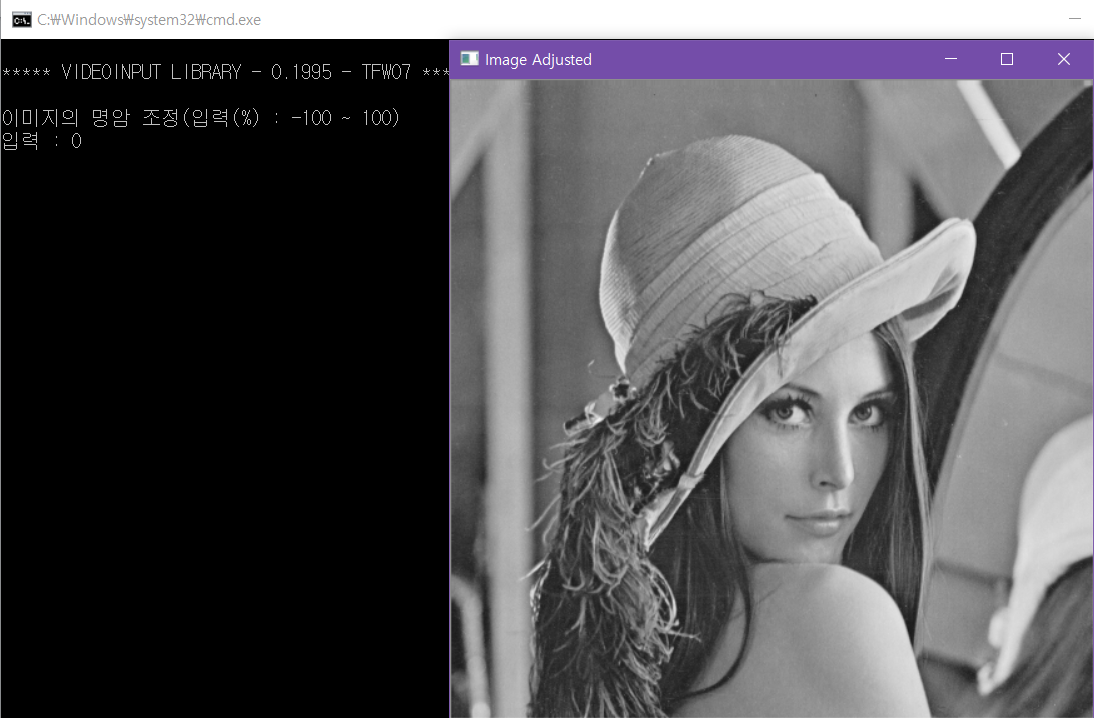
5. 비율을 곱한 세 Channel 값을 결과 이미지의 세 Channel에 각각 대입한다.

6. 윈도우 창과 함께 결과 이미지를 출력한다.

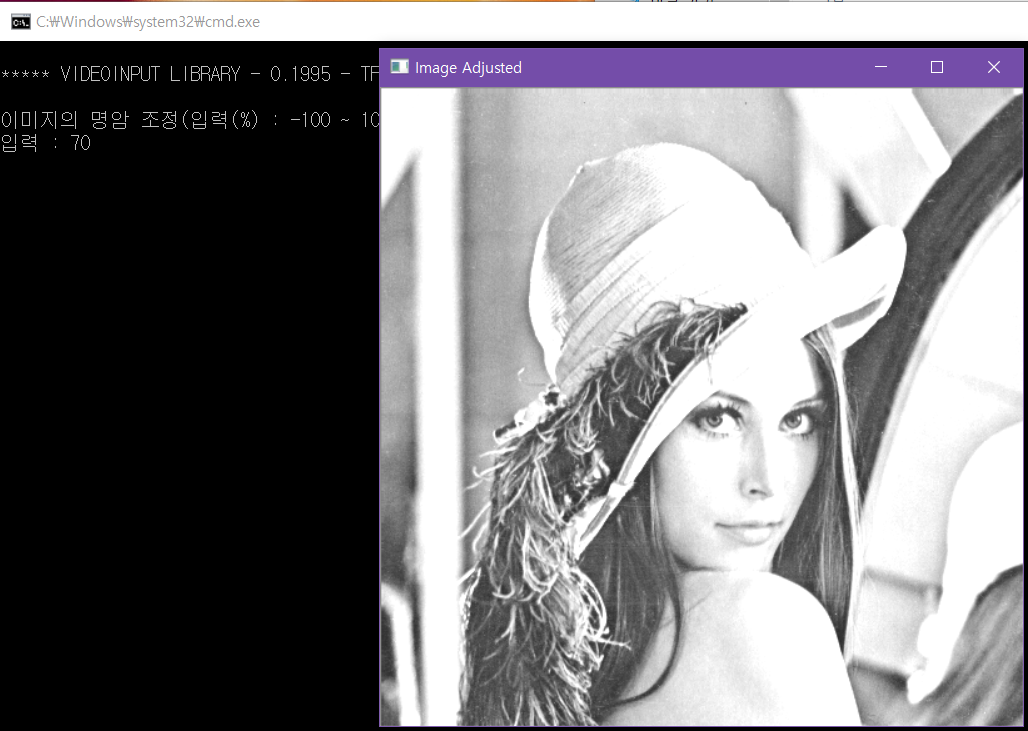
♋ **실행 결과 (ReportCode3-2.cpp)**



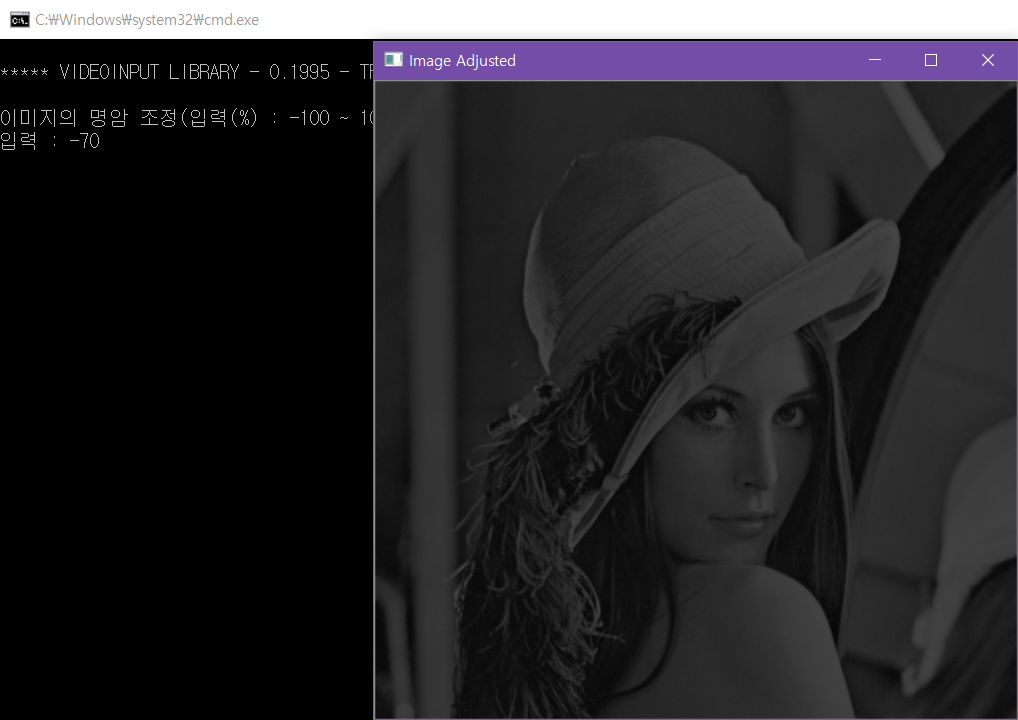
**그림 1. Original Image**



**그림 2. Image Adjusted Contrast 0**



**그림 3. Image Adjusted by Contrast 70**



**그림 4. Image Adjusted by Contrast -70**