컴퓨터비전 HW2 Report

2017203033 이재하

과제 구현 완료 여부

1번 : 작성

2번 1) 작성

2번 2) 성공

2번 3) 성공

3번 1) 성공

3번 2) (1) 성공

3번 2) (2) 실패 (0% 완료)

1. RGB 영상 포맷 확인

<소스코드>

main.cpp

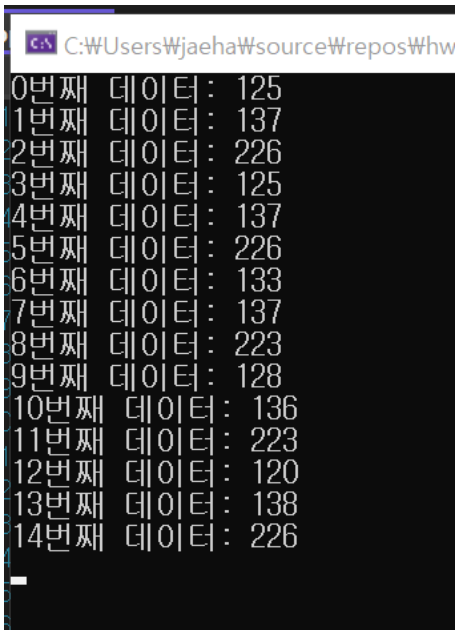
hw2 (전역 범위)

```
#include <math.h>
#include <opencv/cv.h>
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
int main()
{
    Mat img_in;
    img_in = imread("Lena.png");
    imshow("source img", img_in);
    uchar* img_data = img_in.data;

    for (int i = 0; i < 15; i++) {
        uchar pixel = img_data[i];
        printf("%d번째 데이터: %d₩n", i, pixel);
    }
    waitKey(0);
    return 0;
}
```

<읽은 파일>-Lena.png

그림판을 활용하여 (0,0)부터 (4,0)까지 RGB값을 알아냈다

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)
빨강(R): 226	빨강(R): 226	빨강(R): 223	빨강(R): 223	빨강(R): 226
녹색(G): 137	녹색(G): 137	녹색(G): 137	녹색(G): 136	녹색(G): 138
파랑(U): 125	파랑(U): 125	파랑(U): 133	파랑(U): 128	파랑(U): 120



1. 데이터가 일렬로 구성되어 있고 RGB값이 연속적으로 저장되어있다.

2. 각 픽셀의 RGB 데이터는 파랑(Blue), 녹색(Green), 빨강(Red) 순서로 저장되어 있는 것을 알 수 있었다.

2. YUV(YCbCr) color space

디지털 비디오 스트림 전송 포멧이다

Y는 영상에서 어둡고 밝은지 정도를 나타내는 성분인 휘도(Luminance)를 의미하고,

U, V는 색상 정보를 가지는 색차를 의미한다.

YUV에는 서브샘플링 방법에 따라 YUV444, YUV422, YUV411, YUV420 등으로 세분화할 수 있다.

<RGB-to-YUV 변환>

$$Y = (0.257 * R) + (0.504 * G) + (0.098 * B) + 16$$
$$Cr = V = (0.439 * R) - (0.368 * G) - (0.071 * B) + 128$$
$$Cb = U = -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128$$

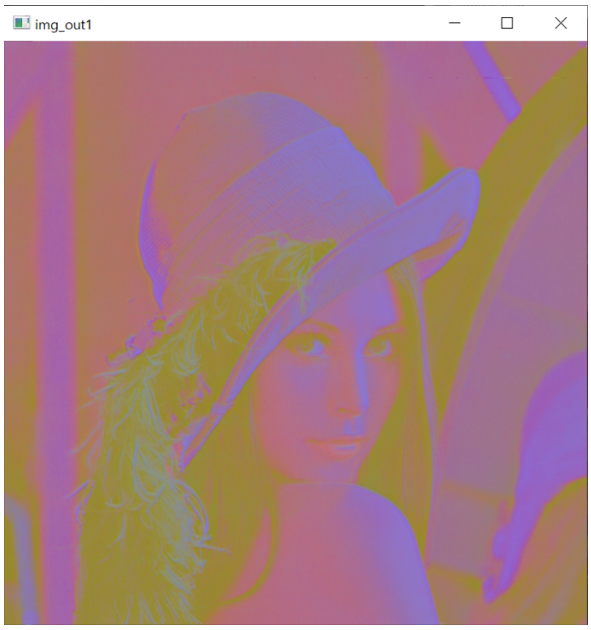
main.cpp

hw2 (전역 범위)

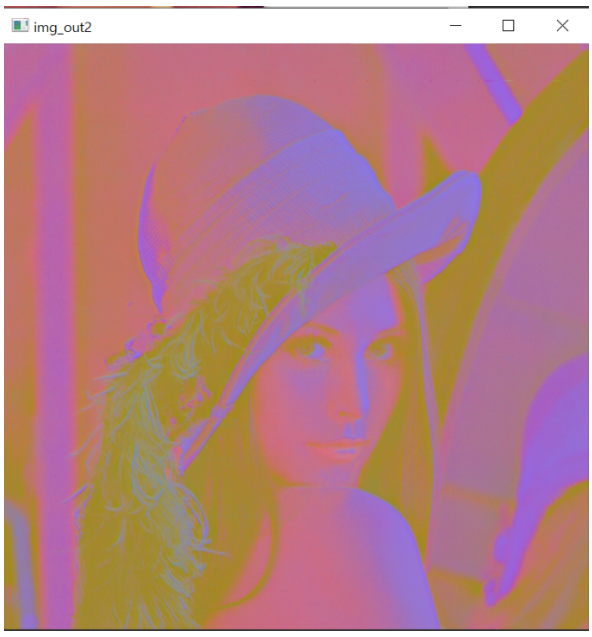
```cpp
#include <math.h>
#include <opencv/cv.h>
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;
void RGBtoYUV(Mat& img_in, Mat& img_out) {
    for (int row = 0; row < img_in.rows; row++)
    {
        for (int col = 0; col < img_in.cols; col++)
        {
            uchar b = img_in.at<Vec3b>(row, col)[0];
            uchar g = img_in.at<Vec3b>(row, col)[1];
            uchar r = img_in.at<Vec3b>(row, col)[2];

            uchar y = (0.257 * r) + (0.504 * g) + (0.098 * b) + 16;
            uchar u = -(0.148 * r) - (0.291 * g) + (0.439 * b) + 128;
            uchar v = (0.439 * r) - (0.368 * g) - (0.071 * b) + 128;

            img_out.at<Vec3b>(row, col)[0] = y;
            img_out.at<Vec3b>(row, col)[1] = u;
            img_out.at<Vec3b>(row, col)[2] = v;
        }
    }
}
```

```cpp
int main()
{
    Mat img_in = imread("Lena.png");
    imshow("source img", img_in);
    Mat img_out1 = Mat(img_in.cols, img_in.rows, img_in.type());
    RGBtoYUV(img_in, img_out1);
    imshow("img_out1", img_out1);

    Mat img_out2;
    cvtColor(img_in, img_out2, COLOR_BGR2YUV);
    imshow("img_out2", img_out2);
    for (int i = 0; i < 30; i++) {
        printf("%d번째 데이터: %d, %d₩n", i, img_out1.data[i], img_out2.data[i]);
    }
    waitKey(0);
    return 0;
}
```

<RGBtoYUV 함수 쓴 결과>

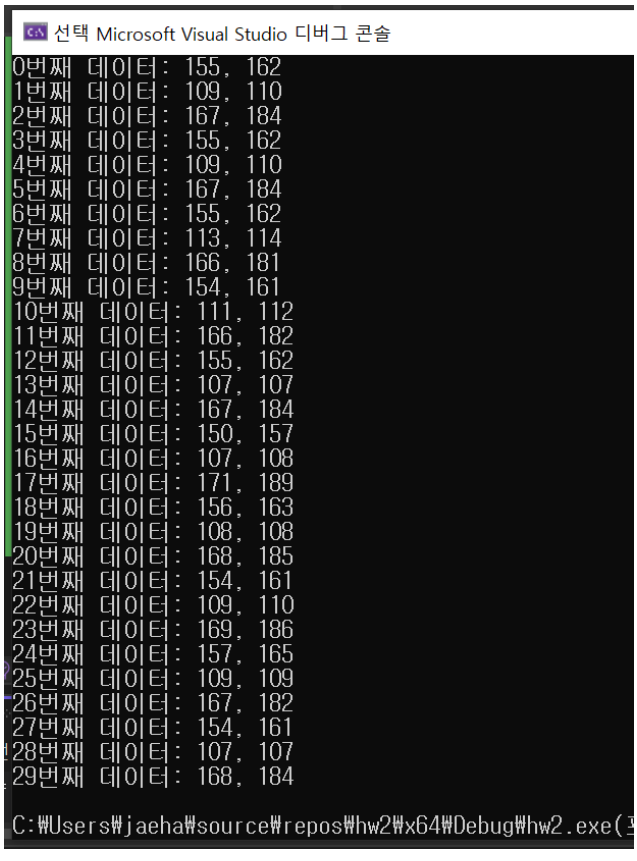


<cvtColor 쓴 결과>



거의 비슷하게 나온 것을 알 수 있다

<데이터 값 비교>



완벽하게 비슷하지 않고 오차가 있지만, 거의 비슷하게 나온 것을 알 수 있었다

3. Color Slicing

 (사용한 사진)

1) 얼굴영역에 대한 색정보 확인

얼굴의 색상 정보를 그림판을 이용하여 확인해보니

대략 120~200(b) / 150~255(r) / 170~255(g) 정도인 것으로 파악되었다.

빨강(R): 238	빨강(R): 226	빨강(R): 235
녹색(G): 191	녹색(G): 185	녹색(G): 190
파랑(U): 171	파랑(U): 163	파랑(U): 169

위의 사진은 대표적으로 확인해 본 3번의 RGB값이다

2) 영상의 pixel 값을 접근하여 얼굴색 여부 판단

(1) 결과 영상 생성

<소스코드>

main.cpp

hw2 (전역 범위)

```
#include <math.h>
#include <opencv/cv.h>
#include <opencv2/opencv.hpp>
using namespace cv;
using namespace std;

int main()
{
    Mat src = imread("myimage.jfif");
    for (int row = 0; row < src.rows; row++)
    {
        for (int col = 0; col < src.cols; col++)
        {
            uchar b = src.at<Vec3b>(row, col)[0];
            uchar g = src.at<Vec3b>(row, col)[1];
            uchar r = src.at<Vec3b>(row, col)[2];
            //얼굴색이 아닐때
            if (120 > b || b > 200 || 150 > g || g > 255 || 170 > r || r > 255) {
                src.at<Vec3b>(row, col)[0] = 0;
                src.at<Vec3b>(row, col)[1] = 0;
                src.at<Vec3b>(row, col)[2] = 0;
            }
            //얼굴색일때
            else {
                src.at<Vec3b>(row, col)[0] = 255;
                src.at<Vec3b>(row, col)[1] = 255;
                src.at<Vec3b>(row, col)[2] = 255;
            }
        }
    }
    imshow("src", src);
    waitKey(0);
    return 0;
}
```

<결과-생성된 영상>

imshow로 확인했더니, black and white로 얼굴 영역을 볼 수 있었다

(2) 얼굴 map을 저장할 Mat를 생성(실패)

- Map 을 사용하지 않고 그대로 영상의 픽셀값만 바꿔서 출력(0%)

(방법이 다르므로 0%로 표기하였음)