3nd Week Lab Assignment

12211728 강민영

▪ 주어진 영상(img1.jpg)에 빨강, 파랑, 초록 색의 점을 각각 설정한 개수만큼 무작위로 생성하는 프로그램을 작성할 것

```cpp
#include <iostream>
#include "opencv2/core/core.hpp" // Mat class와 각종 data structure 및 산술 루틴을 포함하는 헤더
#include "opencv2/highgui/highgui.hpp" // GUI와 관련된 요소를 포함하는 헤더(imshow 등)
#include "opencv2/imgproc/imgproc.hpp" // 각종 이미지 처리 함수를 포함하는 헤더
using namespace cv;
using namespace std;


void BlueSpreadsalts(Mat img, int num)//여기에서 img는 Blue를 뿌리는 이미지의 배경이고 num은 그 픽셀의 수를 나타낸다.
{
    for (int n = 0; n < num; n++)
    {
        int x = rand() % img.cols;//이미지의 폭 정보 저장
        int y = rand() % img.rows;//이미지의 높이 정보 저장

        if (img.channels() == 1) //이미지의 채널 수 반환
        {
            img.at<uchar>(y, x) = 255;//단일 채널 접근

        }
        else
        {
            img.at<Vec3b>(y, x)[0] = 255;//Blue의 픽셀의 가도를 max를 설정해야잘보인다.
            img.at<Vec3b>(y, x)[2] = 0;
            img.at<Vec3b>(y, x)[1] = 0;
        }
    }
}
```

```cpp
void GreenSpreadSalts(Mat img, int num)
{
    for (int n = 0; n < num; n++) {
        int x = rand() % img.cols;
        int y = rand() % img.rows;

        if (img.channels() == 1) {
            img.at<uchar>(y, x) = 255;

        }
        else
        {
            img.at<Vec3b>(y, x)[1] = 255;
            img.at<Vec3b>(y, x)[0] = 0;
            img.at<Vec3b>(y, x)[2] = 0;
        }
    }
}
```

```
void RedSpreadSalts(Mat img, int num)
{
    for (int n = 0; n < num; n++) {
        int x = rand() % img.cols;
        int y = rand() % img.rows;

        if (img.channels() == 1) {

            img.at<uchar>(y, x) = 255;

        }
        else
        {
            img.at<Vec3b>(y, x)[2] = 255;
            img.at<Vec3b>(y, x)[0] = 0;
            img.at<Vec3b>(y, x)[1] = 0;
        }
    }
}
```

```
int main() {
    namedWindow("window", 1);
    Mat src_img = imread("img1.jpg", -1); // 이미지 읽기
    // int flags = IMREAD_COLOR 또는 int flags = 1 -> 컬러 영상으로 읽음
    // int flags = IMREAD_GRAYSCALE 또는 int flags = 0 -> 흑백 영상으로 읽음
    // int flags = IMREAD_UNCHANGED 또는 int flags = -1 -> 원본 영상의 형식대로 읽음
    BlueSpreadSalts(src_img, 300);
    GreenSpreadSalts(src_img, 400);
    RedSpreadSalts(src_img, 600);
    imshow("Test window", src_img); // 이미지 출력
    waitKey(0); // 키 입력 대기(0: 키가 입력될 때 까지 프로그램 멈춤)
    destroyWindow("Test window"); // 이미지 출력창 종료
    imwrite("langding_gray.png", src_img); // 이미지 쓰기


    return 0;
}
```

빨강, 파랑, 초록의 점을 흩뿌리는 코드를 작성하고 이를 통해 main함수를 작성 후 실행하면 아래와 같은 결과를 얻을 수 있다. 빨강, 파랑, 초록 점이 수에 맞게 흩뿌려진 이미지를 얻을 수 있다.

▪ 앞서 생성한 영상에서 빨강, 파랑, 초록 색의 점을 각각 카운트하는 프로그램을 작성하고 카운트 결과가 실제와 일치하는지 검증할 것

```
int countingBlue(Mat img)
{
    int count = 0;
    for (int y = 0; y < img.rows; y++) {
        for (int x = 0; x < img.cols; x++) {
            if (img.at<Vec3b>(y, x)[0] == 255 && img.at<Vec3b>(y, x)[1] == 0 && img.at<Vec3b>(y, x)[2] == 0)
                count++;
        }
    }
    return count;
}
```

```
int countingGreen(Mat img)
{
    int count = 0;
    for (int y = 0; y < img.rows; y++)
    {
        for (int x = 0; x < img.cols; x++)
        {
            if (img.at<Vec3b>(y, x)[1] == 255 && img.at<Vec3b>(y, x)[0] && img.at<Vec3b>(y, x)[2] == 0)
                count++;
        }
    }
    return count;
}
```

```
int countingRed(Mat img)
{
    int count = 0;
    for (int y = 0; y < img.rows; y++)
    {
        for (int x = 0; x < img.cols; x++)
        {
            if (img.at<Vec3b>(y, x)[2] == 255 && img.at<Vec3b>(y, x)[0] && img.at<Vec3b>(y, x)[1] == 0)
                count++;
        }
    }
    return count;
}
```

아까 흩뿌리는데 썼던 spread 함수와 함께 counting 함수로 흩뿌린 점들의 개수를 각각 색에 따라 센다. 셀 때는 파랑, 빨강, 초록을 다르게 표현하여 함수를 만든다. 먼저 spread함수를 만들 때 채널의 값을 구하고자 하는 값으로 255로 하고 나머지는 모두 0으로 설정하였기 때문에 counting함수도 이에 맞추어서 함수를 만든다. 그리고 main함수에서 cout을 통해 그 값을 확인한다.

```
int main() {
    namedWindow("window", 1);
    Mat src_img = imread("img1.jpg", -1); // 이미지 읽기
    // int flags = IMREAD_COLOR 또는 int flags = 1 -> 컬러 영상으로 읽음
    // int flags = IMREAD_GRAYSCALE 또는 int flags = 0 -> 흑백 영상으로 읽음
    // int flags = IMREAD_UNCHANGED 또는 int flags = -1 -> 원본 영상의 형식대로 읽음
    BlueSpreadSalts(src_img, 300);
    GreenSpreadSalts(src_img, 400);
    RedSpreadSalts(src_img, 600);
    imshow("Test window", src_img); // 이미지 출력
    waitKey(0); // 키 입력 대기(0: 키가 입력될 때 까지 프로그램 멈춤)
    destroyWindow("Test window"); // 이미지 출력창 종료
    imwrite("langding_gray.png", src_img); // 이미지 쓰기

    cout << "Blue : " << countingBlue(src_img) << "\n"
        << "Green : " << countingGreen(src_img) << "\n"
        << "Red : " << countingRed(src_img) << "\n";

    return 0;
}
```

```
Blue : 299
Green : 397
Red : 600
```

spread함수에서 파랑, 초록, 빨강 순으로 각 300, 400, 600 개를 뿌렸기 때문에 결과가 거의 비슷하게 나오는 것을 알 수 있다.

▪ 주어진 영상을 이용해(img2.jpg) 다음과 같은 두 영상을 생성하는 프로그램을 작성하고(픽셀 값 접근을 이용) 히스토그램 일치 여부를 확인 및 그러 한 결과가 나온 이유를 분석할 것

```
void bright(Mat& img)
{
    float k = 0.0;
    for (int i = 0; i < img.rows; i++)
    {
        if (k > 255)
            k = 255.0;

        for (int j = 0; j < img.cols; j++)
        {
            img.at<uchar>(i, j) = k;
        }
        k += 0.6;
    }
}
```

영상에서 위로 갈수록 점점 어두움과, 아래로 갈수록 점점 어두움을 구현하기 위해 먼저 bright함수를 만드는데 행에 따라 밝기 변화를 주기 위해서 각 행에 밝기 값을 k로하고 k를 점점 증가시키는 함수를 만든다. 히스토그램을 만들기 위해 밝기 값은 <uchar>을 이용하였고, 두가지 영상의 밝기를 구현하기 위해서는 반대로 어두워지는 경우를 bitwise_not함수를 이용하면 된다.

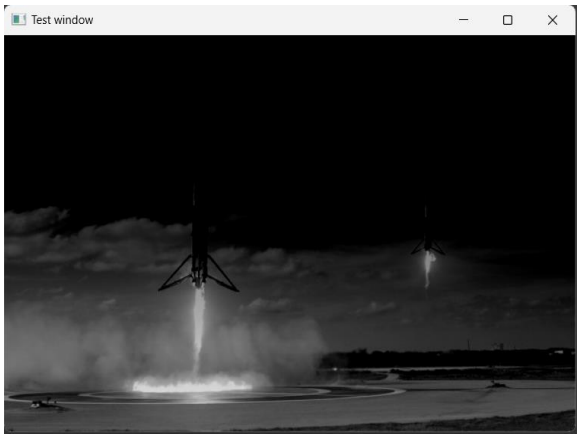
```
int main() {
    namedWindow("window", 1);
    Mat src_img = imread("img2.jpg", 0); // 이미지 읽기
    Mat bef = imread("img2.jpg", 0);
    bright(src_img);
    Mat gr;
    bitwise_not(src_img, gr);
    Mat result;
    //result = bef - gr;
    result = bef - src_img;

    //Mat histo = GetHistogram(result);

    imshow("Test window", result); // 이미지 출력
    waitKey(0); // 키 입력 대기(0: 키가 입력될 때 까지 프로그램 멈춤)
    destroyWindow("Test window"); // 이미지 출력창 종료

}
```

main함수를 이와 같이 구현하고 먼저 bright함수를 통해 src_img를 보면 왼쪽과 같이 위로 갈수록 어두워지는 이미지가 나오고 이와 반대로하는 bitwise_not(src_img, gr)을 넣은 gr은 오른쪽처럼 위로 갈수록 밝아지는 이미지가 나온다는 것을 알 수 있다.



이를 이제 img2.jpg에 적용하면 왼쪽사진이 img2에서 src_img를 뺀 결과이다. 이는 아래로 갈수록 점점 어두워지는 것을 확인할 수 있고, 오른쪽 사진은 img2에서 gr을 뺀 결과이고 이는 위로 갈 수록 점점 어두워지는 것을 알 수 있다.

```cpp
Mat GetHistogram(Mat& src)
{
    Mat histogram;
    const int* channel_numbers = { 0 };
    float channel_range[] = { 0.0, 255.0 };
    const float* channel_ranges = channel_range;
    int number_bins = 255;

    calcHist(&src, 1, channel_numbers, Mat(), histogram, 1, &number_bins, &channel_ranges);

    int hist_w = 512;
    int hist_h = 400;
    int bin_w = cvRound((double)hist_w / number_bins);

    Mat histImage(hist_h, hist_w, CV_8UC1, Scalar(0, 0, 0));
    normalize(histogram, histogram, 0, histImage.rows, NORM_MINMAX, -1, Mat());

    for (int i = 1; i < number_bins; i++)
    {
        line(histImage, Point(bin_w * (i - 1), hist_h - cvRound(histogram.at<float>(i - 1))),
            Point(bin_w * (i), hist_h - cvRound(histogram.at<float>(i))),
            Scalar(255, 0, 0), 2, 8, 0);
    }
    return histImage;
}
```

이는 이제 두 사진을 히스토그램 형태로 보여주기 위한 함수 코드이다.

img2.jpg에 적용하면 왼쪽사진이 img2에서 src_img를 뺀 히스토그램이고, 오른쪽 사진은 img2에서 gr을 뺀 히스토그램이다. 둘은 비슷한 양상을 띄는 것을 알 수 있고 둘 다 왼쪽에서 값이 나타나는 것을 확인할 수 있다. 값이 나타난 이유는 밝은 불꽃부분 때문인 것으로 보인다. 둘이 일치하는 이유도 반짝거리는 불꽃 부분이 같기 때문인 것으로 보인다.

▪ 주어진 영상(img3.jpg, img4.jpg, img5.jpg)을 이용해 다음의 영상을 완성할 것

```
int main() {
    namedWindow("window", 1);
    Mat src_img = imread("img3.jpg", -1); // 이미지 읽기
    Mat s1 = imread("img4.jpg", -1);
    resize(s1, s1, Size(src_img.cols, src_img.rows));// s1의 크기를 src_img 크기에 맞게 조정
    Mat s2 = imread("img5.jpg", IMREAD_GRAYSCALE); // img5 사진 흑백으로 선언
    Mat s3 = imread("img5.jpg", -1); // img5 사진 원본으로 선언
    resize(s2, s2, Size(400, 100)); // 가로 400 세로 100으로 크기 조정
    resize(s3, s3, Size(400, 100)); // 가로 400 세로 100으로 크기 조정
    Mat result;
    subtract(src_img, s1, result);// 원하는 배경을 얻기 위해 외곽을 subtract함수로 검정색으로 만듦
    Mat roi;
    roi = result(Rect(Point(400, 400), Point(800, 500))); //mask를 하기 위해 값 설정
    Mat s;
    s = s3 - roi; // s에는 s3에서 roi를 뺀 값이 들어감 -> s를 not에 통과하면 기존 배경화면에 글씨도 하얗게 변함
    Mat s4;
    bitwise_not(s, s4); // s4를 s의 not 연산으로 처리하여 배경화면에 있는 글씨 배경이 사라지게 함
    bitwise_and(s4, s3, roi); // and문을 이용하여 결합


    imshow("Test window", result); // 이미지 출력
    waitKey(0); // 키 입력 대기(0: 키가 입력될 때 까지 프로그램 멈춤)
    destroyWindow("Test window"); // 이미지 출력창 종료



    return 0;
}
```

문제의 조건을 만족하기 위한 코드는 위와 같다.

아래는 이에 만족하는 결과 사진이다.