3nd Week Lab Assignment

12201928 이상혁

Task 1

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

```
void SpreadSalts(Mat img, int R, int G, int B) {
    for (int n = 0; n < R; n++) {
        int x = rand() % img.cols;
        int y = rand() % img.rows;
        if (img.channels() == 1) {
            img.at<uchar>(y, x) = 255;
        }
        else {
            //Red Point 출력
            img.at<Vec3b>(y, x)[0] = 0;
            img.at<Vec3b>(y, x)[1] = 0;
            img.at<Vec3b>(y, x)[2] = 255;
        }
    }
    for (int n = 0; n < G; n++) {
        int x = rand() % img.cols;
        int y = rand() % img.rows;
        if (img.channels() == 1) {
            img.at<uchar>(y, x) = 255;
        }
        else {
            //Green Point 출력
            img.at<Vec3b>(y, x)[0] = 0;
            img.at<Vec3b>(y, x)[1] = 255;
            img.at<Vec3b>(y, x)[2] = 0;
        }
    }
    for (int n = 0; n < B; n++) {
        int x = rand() % img.cols;
        int y = rand() % img.rows;
        if (img.channels() == 1) {
            img.at<uchar>(y, x) = 255;
        }
        else {
            //Blue Point 출력
            img.at<Vec3b>(y, x)[0] = 255;
            img.at<Vec3b>(y, x)[1] = 0;
            img.at<Vec3b>(y, x)[2] = 0;
        }
    }
}
```

-void SpeardSalts

4개의 매개변수를 받는 SpreadSalt 함수를 만들었다. R,G,B 각각의 점 개수를 매개변수로 받았으며 rand에 픽셀 값을 % 계산하여 사진 내에 랜덤으로 찍힐 수 있게 구현하였다.
[0] 이 BLUE, [1] 이 Green, [2] 가 Red를 담당하므로 각각의 배열의 값에 255을 대입하고 다른 값들에는 0을 대입하여 온전한 색이 나올 수 있도록 하였다.

-main

코드를 실행할때마다 랜덤으로 출력될 수 있도록 srand를 사용하였다.

Imread를 사용해 img1을 불러왔으며 RGB 각각 30,50,70개 점이 랜덤으로 찍히도록 함수에 매개변수로 전달해주었다.

점이 찍힌 다음 imshow로 이미지 출력을 진행하였다. 키입력대기, 이미지 출력종료까지 실행한다.

Figure 1 void SpeardSalts 함수 Code 사진

```
int main() {
    srand(time(NULL));
    Mat src_img = imread("img1.jpg", 1);
    if (src_img.empty()) {
        cerr << "image load failed!" << endl;
        return -1;
    }
    SpreadSalts(src_img, 30,50,70); // 소금효과 이미지에 추가, 앞부터 RGB 순서
    imshow("Task 1", src_img); // 이미지 출력
    //Count(src_img);
    waitKey(0);
    destroyWindow("Task 1");
}
```

Figure 2 main 함수 Code 사진

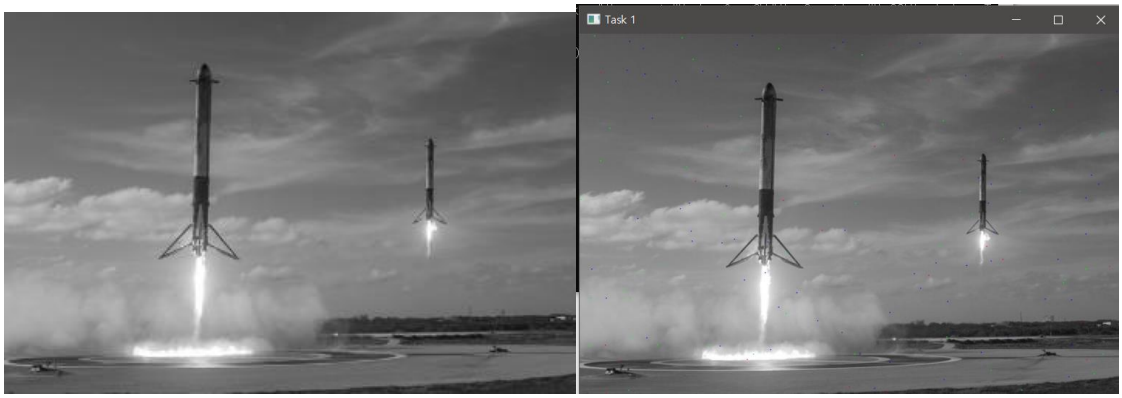

Figure 3 좌 – 원본사진, 우- 점을 찍은 후 사진

Task 2

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

```
void Count(Mat img) {
    int R_P = 0;
    int B_P = 0;
    int G_P = 0;
    for (int x = 0; x < img.cols; x++) {
        for (int y = 0; y < img.rows; y++) {
            if (img.at<Vec3b>(y, x)[0] == 0 && img.at<Vec3b>(y, x)[1] == 0 && img.at<Vec3b>(y, x)[2] == 255) R_P++;
            if (img.at<Vec3b>(y, x)[0] == 0 && img.at<Vec3b>(y, x)[1] == 255 && img.at<Vec3b>(y, x)[2] == 0) G_P++;
            if (img.at<Vec3b>(y, x)[0] == 255 && img.at<Vec3b>(y, x)[1] == 0 && img.at<Vec3b>(y, x)[2] == 0) B_P++;
        }
    }
    cout << "Red : " << R_P << "WnGreen : " << G_P << "WnBlue : " << B_P << endl;
}
```

Figure 4 점의 개수를 새는 Count 함수 CODE

1. for 루프를 이용하여 이미지의 모든 열과 행을 반복한다.
2. 각 픽셀에 대해 해당하는 RGB 색상 값이 특정 값인지 확인한다.

현재 픽셀의 색상이 (0, 0, 255)인 경우, R_P 변수를 증가,

현재 픽셀의 색상이 (0, 255, 0)인 경우, G_P 변수를 증가

현재 픽셀의 색상이 (255, 0, 0)인 경우, B_P 변수를 증가

그 후 저장된 값들을 출력하여 확인할 수 있도록 하였다. main에서는 이 함수를 사용하는 코드만 작성하였다.

다음 페이지의 사진에서 확인할 수 있듯이 위 Task1에서 보낸 값인 30,50,70이 동일하게 찍힘을 확인할 수 있었다.

```
    SpreadSalts(src_img, 30,50,70); // 소금효과 이미지에 추가
    imshow("Task 1", src_img); // 이미지 출력
    Count(src_img);
    waitKey(0);
    destroyWindow("Task 1");
}
```

```
tbb480_64d.dll
[ INFO:0@0.276
x64₩vc17₩bin₩op
[ INFO:0@0.278
480_64d.dll =>
[ INFO:0@0.279
x64₩vc17₩bin₩op
[ INFO:0@0.281
nmp480_64d.dll
Red : 30
Green : 50
Blue : 70
```

Figure 5 좌 – Count 함수를 실행한 main CODE,
우 – Figure 3 의 사진에서 점을 COUNT 한 후 출력된 R,G,B 의 개수

Task 3

▪ 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 Gradation1(Mat img) {
    for (int y = 0; y < img.rows; y++) {
        int k = y * 255 / img.rows;
        for (int x = 0; x < img.cols; x++) {
            Vec3b& pixel = img.at<Vec3b>(y, x);
            pixel -= Vec3b(k, k, k);
        }
    }
    imshow("12201928 이상혁 1", img);
}
void Gradation2(Mat img) {
    for (int y = (img.rows-1); y >=0 ; y--) {
        int k = (img.rows - 1-y) * 255 / img.rows;
        for (int x = 0; x < img.cols; x++) {
            Vec3b& pixel = img.at<Vec3b>(y, x);
            pixel -= Vec3b(k, k, k);
        }
    }
    imshow("12201928 이상혁 2", img);
}
```

Figure 6 Task3 에서 두 영상을 생성하는 함수 CODE

Gradation 1

이미지의 각 행에 대해 반복하면서, 그 행의 위치에 그라데이션을 적용하게 된다.
y값을 기준으로 0에서 img.rows까지의 범위에서 0부터 255까지의 값을 생성하고, 이를 k에 저장한다. 그리고 이미지의 각 픽셀에 대해 현재 픽셀의 각 채널 값에서 k만큼 빼준다.

이렇게 진행을 하게 되면 이미지의 아래쪽으로 갈수록 어두워지는 그라데이션 효과가 적용된다.

Gradation 2

이미지의 각 행에 대해 반대 즉 역순으로 반복하면서, 그 행의 위치에 그라데이션을 적용하게 된다. y값을 기준으로 img.rows-1에서 0까지의 범위에서까지의 범위에서 0부터 255까지의 값을 생성하고, 이를 k에 저장한다. 그리고 이미지의 각 픽셀에 대해 현재 픽셀의 각 채널 값에서 k만큼 빼준다.

이렇게 진행을 하게 되면 이미지의 위으로 갈수록 어두워지는 그라데이션 효과가 적용된다.

이후 동일하게 imshow을 사용하여 이미지를 표시한다.

```
int main() {
    Mat img2 = imread("img2.jpg", 1);
    Mat img3 = imread("img3.jpg", 1);
    Mat img4 = imread("img4.jpg", 1);
    Mat img5 = imread("img5.jpg", 1);
    if (img2.empty() || img3.empty() || img4.empty() || img5.empty()) {
        cerr << "image load failed!" << endl;
        return -1;
    }
    Gradation1(img2);
    Gradation2(img2);
    //histogram(img1, img2);
    //MakeVideo(img3, img4, img5);
    waitKey(0);
    destroyAllWindows();
}
```

Figure 7 Task3 의 Main CODE

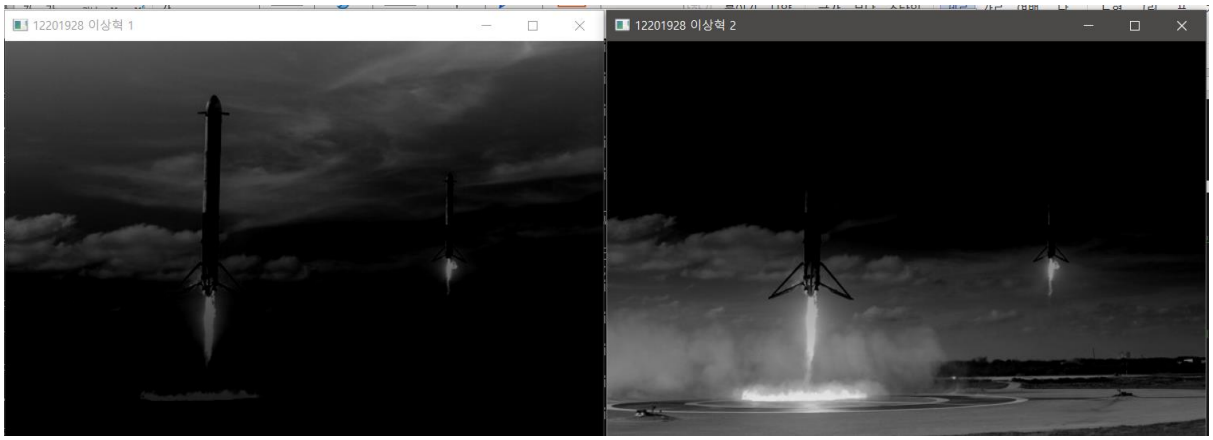

Figure 8 코드를 실행시킨 결과 왼쪽이 Gradation1 오른쪽이 Gradation2 이다.

```cpp
Mat GetColorHistogram(Mat src) {
    vector<Mat> bgr_planes;
    split(src, bgr_planes);

    int histSize = 256;
    float range[] = { 0, 256 };
    const float* histRange = { range };
    bool uniform = true, accumulate = false;
    Mat b_hist, g_hist, r_hist;

    calcHist(&bgr_planes[0], 1, 0, Mat(), b_hist, 1, &histSize, &histRange, uniform, accumulate);
    calcHist(&bgr_planes[1], 1, 0, Mat(), g_hist, 1, &histSize, &histRange, uniform, accumulate);
    calcHist(&bgr_planes[2], 1, 0, Mat(), r_hist, 1, &histSize, &histRange, uniform, accumulate);

    int hist_w = 512; // 히스토그램 이미지의 너비
    int hist_h = 400; // 히스토그램 이미지의 높이
    int bin_w = cvRound((double)hist_w / histSize);

    Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

    normalize(b_hist, b_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
    normalize(g_hist, g_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
    normalize(r_hist, r_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());

    for (int i = 1; i < histSize; i++) {
        line(histImage, Point(bin_w * (i - 1), hist_h - cvRound(b_hist.at<float>(i - 1))),
            Point(bin_w * (i), hist_h - cvRound(b_hist.at<float>(i))),
            Scalar(255, 0, 0), 2, 8, 0);
        line(histImage, Point(bin_w * (i - 1), hist_h - cvRound(g_hist.at<float>(i - 1))),
            Point(bin_w * (i), hist_h - cvRound(g_hist.at<float>(i))),
            Scalar(0, 255, 0), 2, 8, 0);
        line(histImage, Point(bin_w * (i - 1), hist_h - cvRound(r_hist.at<float>(i - 1))),
            Point(bin_w * (i), hist_h - cvRound(r_hist.at<float>(i))),
            Scalar(0, 0, 255), 2, 8, 0);
    }

    return histImage;
}

void histogram(Mat img1,Mat img2) {
    Mat histImage1 = GetColorHistogram(img1);
    Mat histImage2 = GetColorHistogram(img2);
    imshow("12201928 이상혁 3", histImage1);
    imshow("12201928 이상혁 4", histImage2);
}
```

Figure 9 히스토그램을 진행할 코드

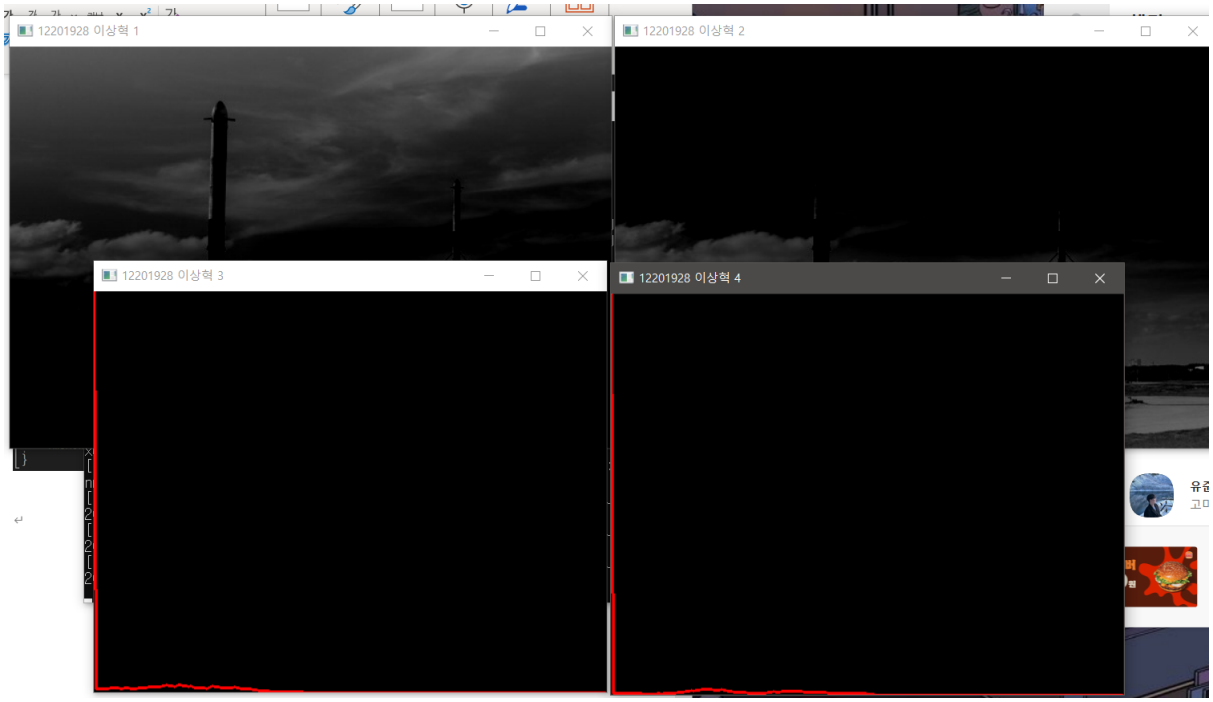

Figure 10 히스토그램 사진

거의 유사한 모습의 histogram을 확인할 수 있다. 화소의 강도(Gray 값 또는 하나의 채널에 대한 값)를 갖는 화소의 개수를 각각 X축과 Y축으로 표시한다. 오른쪽의 히스토그램이 조금 더 화소의 강도가 높음을 확인할 수 있다. 이미지에도 보면 알 수 있듯이 오른쪽 이미지가 조금더 밝기가 높기 때문으로 분석할 수 있다. 그 이외의 값들은 비슷한 채널의 color로 이루어져 있으며 비슷한 화소의 개수 및 밝기로 인해 유사하다고 판단할 수 있었다.

Task 4

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

```
void MakeVideo(Mat img3, Mat img4,Mat img5) {
    resize(img4, img4, Size(img3.cols, img3.rows));
    Mat result1,result2,img5_gray,img5_mask;
    subtract(img3, img4, result1);
    resize(img5, img5, Size(img5.cols*0.9, img5.rows*0.9));
    cvtColor(img5, img5_gray, COLOR_BGR2GRAY);
    threshold(img5_gray, img5_mask, 180, 255, THRESH_BINARY_INV);
    Rect logoRoi = Rect(320, 340, img5.cols, img5.rows);
    Mat result1_roi(result1, logoRoi);
    img5.copyTo(result1_roi, img5_mask);
    imshow("12201928 이상혁 5", result1);
}
```

Figure 11 주어진 영상을 만들기 위한 함수

MakeVideo 함수는 세 개의 이미지(img3, img4, img5)를 사용한다.

1. Resize 를 사용해서 img4 이미지를 img3 이미지와 동일한 크기로 조정하게 된다, 이때 size 로 im3 과 동일한 크기를 가지도록 조정한다.
2. Subtract 를 사용하여 두 이미지의 histogram 을 빼주게 된다. 이렇게 한다면 아래의 사진과 같이 img4 의 밝은 부분이 img3 에서 어둡게 나타날 수 있게된다!
3. 이후 spaceX 로고가 담긴 img5 의 size 를 resize 해준다.
4. Cvtcolor 을 사용하여 grayscale 영사으로 변환해준다.
5. 이후 threshold 함수를 사용하여 img5_gray 를 임계값 180 을 기준으로 이진화한 이후 mask 에 저장하게 됩니다. 이때 임계값 조절에 따라서 mask 의 생성 모습 차이를 확인할 수 있었다.
6. Rect 클래스를 사용하여 img5 를 적용할 Roi 를 만들어주었다
   여기서는 (320, 340) 위치에서 img5 의 크기로 해주었다.
7. img5 를 result1 의 ROI 에 적용하여 결과를 result1_roi 에 저장한다. 이 때, img5_mask 를 마스킹하여 적용한다.
8. 이후 copyto 를 사용하여 로고 마스크와 만들어준 roi 를 합쳐준다.

Figure 12 MakeVideo 를 실행한 결과

고찰 : openCV에 대한 여러가지 기능을 다루어보면서 배우고 이에 주어진 과제를 실행해보았다. 픽셀 값 접근에 대해서 이해할 수 있었으며 matrix operation 인 subtract 연산을 해볼 수 있을 수 있었다. 마지막 과제에서 어려움이 있었다. 처음에는 그냥 합쳤는데 배경이 제거 되지 않은 상태였기에 덮어버리는 모습이였다. 이를 해결하기위해 배경제거 방법을 찾게 되었고, 이진화와 마스크를 이용한 배경제거로 글자만 뽑을 수 있었다. 그 이후 roi 에 대해서 알 수 있었으며 적절한 값의 할당이 중요함을 확인하게 되었다.