디지털 영상 처리 설계 과제 보고서 (2주차)

정보통신공학과

12171797

신원철

1. Introduction

A. 픽셀 값에 접근하여 이미지를 편집한다.

B. 히스토그램을 사용하여 이미지를 분석한다.

C. 연산자를 사용하여 이미지를 합성한다.

2. Problem

A. 주어진 사진에 랜덤하게 픽셀에 R, G, B의 픽셀을 삽입하고, 결과를 확인하는 프로그프로 만들 것.

```
void SpreadSalts(Mat img, int num) {
    for (int n = 0; n < num; n++) {
        int x = rand() % img.cols;
        int y = rand() % img.rows;
        int x_1 = rand() % img.cols;
        int y_1 = rand() % img.rows;
        int x_2 = rand() % img.cols;
        int y_2 = rand() % img.rows;

        if (img.channels() == 1) {
            img.at<uchar>(y, x) = 255;
        }
        else {
            img.at<Vec3b>(y, x)[0] = 0; //b
            img.at<Vec3b>(y, x)[1] = 255; //g
            img.at<Vec3b>(y, x)[2] = 0; //r

            img.at<Vec3b>(y_1, x_1)[0] = 255; //b
            img.at<Vec3b>(y_1, x_1)[1] = 0; //g
            img.at<Vec3b>(y_1, x_1)[2] = 0; //r

            img.at<Vec3b>(y_2, x_2)[0] = 0; //b
            img.at<Vec3b>(y_2, x_2)[1] = 0; //g
            img.at<Vec3b>(y_2, x_2)[2] = 255; //r
        }
    }
}
```

위와 같이 숫자를 입력으로 받는 SpreadSalt()함수를 정의하였다. Rand()함수를 사용하여 랜덤한 정수를 만들고 이를 이미지의 가로, 세로 길이로 나눈 나머지를 사용하여 이미지 안에 존재하는 임의의 픽셀에 접근하였다. 흑백 이미지의 경우와 칼라 이미지의 경우를 나누어 함수를 만들었고, 칼라 이미지의 경우 R, G, B각각 입력 받은 숫자만큼 점을 찍도록 하였다.

```
void CountSalt(Mat img) {
    for (int i = 0; i < img.cols; i++) {
        for (int j = 0; j < img.rows; j++) {
            if (img.at<Vec3b>(j, i)[0] == 255 && img.at<Vec3b>(j, i)[1] == 0 && img.at<Vec3b>(j, i)[2] == 0) b++;
            if (img.at<Vec3b>(j, i)[0] == 0 && img.at<Vec3b>(j, i)[1] == 255 && img.at<Vec3b>(j, i)[2] == 0) g++;
            if (img.at<Vec3b>(j, i)[0] == 0 && img.at<Vec3b>(j, i)[1] == 0 && img.at<Vec3b>(j, i)[2] == 255) r++;
        }
    }
    cout << "R: " << r << " G: " << g << " B: " << b << endl;
}
```

또한 각각의 픽셀 값에 접근하여 픽셀의 색이 R, G, B인 경우 전역 변수로 선언된 r, g, b의 숫자를 하나씩 증가하여 흑백 사진에서 점의 개수를 count하였다.

B. 아래로 갈수록 이미지가 어두워지게 하는 함수와 위로 갈수록 이미지가 어두워지게 함수를 구현해야 한다.

```
void dark_and_darker(Mat img) {
    for (int j = 0; j < img.rows; j++) {
        for (int k = 0; k < img.cols; k++) {
            if(img.at<uchar>(j, k) - (j * 255 /img.rows) < 0)
                img.at<uchar>(j, k) = 0;
            else
                img.at<uchar>(j, k) -= (j * 255 / img.rows);

        }
    }
}
```

위와 같이 함수를 선언하여 이미지가 아래로 갈수록 어두워지는 함수를 만들었다 이미지의 row값에 따라 0~255범위에서 점점 증가하도록 하였고, 전체 이미지에서 해당 값을 빼 점점 픽셀의 값이 작아지게 함으로써 위 함수를 구현하였다.

```
void dark_and_darker(Mat img) {
    for (int j = 0; j < img.rows; j++) {
        for (int k = 0; k < img.cols; k++) {
            if (img.at<uchar>(j, k) - (255 - j * 255 / img.rows) < 0)
                img.at<uchar>(j, k) = 0;
            else
                img.at<uchar>(j, k) -= (255 - j * 255 / img.rows);
        }
    }
}
```

함수를 위와 같이 수정하게 되면 위로 갈수록 어두워지는 이미지를 생성할 수 있다.

이 두 이미지에 대해 히스토그램을 사용하여 분석하였다.

C. 크기가 다른 세 이미지를 합성해야 한다.

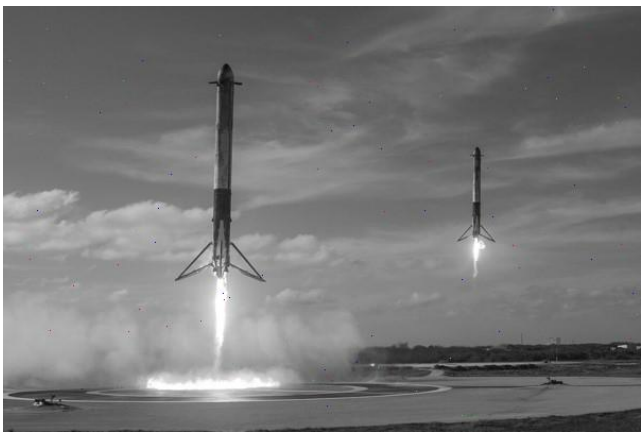
```
resize(src_img4, src_img4, Size(src_img3.cols, src_img3.rows));
dst1 = src_img3 - src_img4;
int x = dst1.cols/4, y = dst1.rows/2;
for (int i = x; i < x+src_img5.cols; i++) {
    for (int j = y; j < y + src_img5.rows; j++) {
        if (src_img5.at<Vec3b>(j - y, i - x)[0] != 255 && src_img5.at<Vec3b>(j - y, i - x)[1] != 255 && src_img5.at<Vec3b>(j - y, i - x)[2] != 255) {
            for (int k = 0; k < 3; k++) {
                dst1.at<Vec3b>(j, i)[k] = src_img5.at<Vec3b>(j - y, i - x)[k];
            }
        }
    }
}
```

위와 같이 img4를 img3와 같은 크기로 resize한 후에 뺄셈 연산을 해주었다. 그 다

음 img5에 해당하는 크기만큼 픽셀에 접근하여 img5의 흰색에 해당하는 부분은 그대로 남기고 색이 있는 부분은 img5의 픽셀 색으로 대체하여 합성을 진행하였다.

3. Result

A.

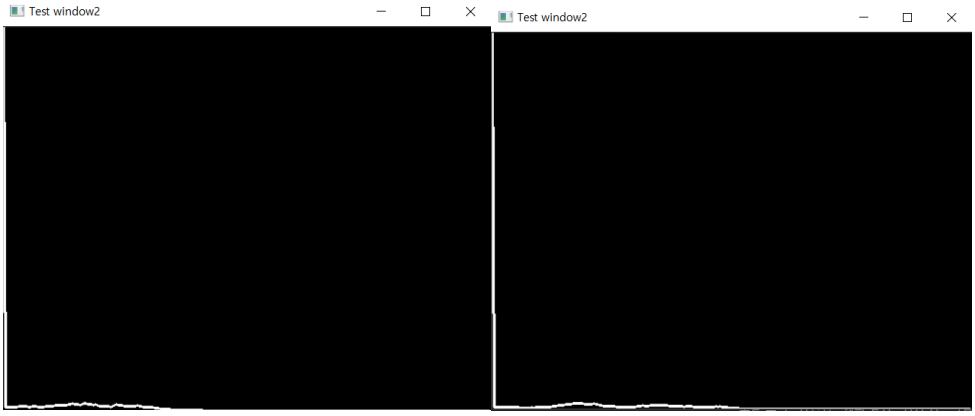


위와 같이 랜덤하게 r, g, b 색의 픽셀이 추가된 것을 확인할 수 있다.

R, g, b의 개수도 정확하게 출력되는 것을 확인할 수 있다.

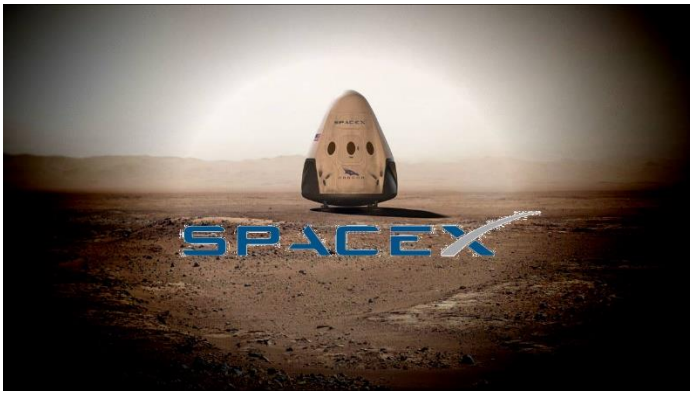
B. 위 함수를 실행한 결과는 다음과 같다.

정상적으로 올라감에 따라 어두워지는 영상과 내려감에 따라 어두워지는 영상이 생성된 것을 확인할 수 있다. 위 히스토그램은 다음과 같다.



두 히스토그램은 상당히 유사하지만 살짝 다른 것을 확인할 수 있다. 기본적으로 두 사진 모두 어두운 부분이 많기 때문에 0쪽의 값이 상당히 높은 것을 확인할 수 있다. 그러나 위 사진이 생성하기 위한 뺄셈 영상을 수행하는 과정에서 픽셀 값이 0보다 작은 경우는 0으로 설정하였기 때문에 두 히스토그램이 완전히 같은 값을 가지지는 않는 사실을 확인할 수 있다.

C. 위 코드를 수행한 결과는 다음과 같다.



위 결과에서 spaceX글자 주변 부분에 하얀색 픽셀이 많은 것을 확인할 수 있는데, 이는 해당 픽셀 값이 완벽하게 흰색을 나타내지 않아서 발생하는 문제임을 유추할 수 있었다.

4. Conclusion

A, B는 결과가 좋았지만, C번의 경우 결과가 완벽하게 출력되지 않았다.