INHA UNIVERSITY

ICE4027 디지털영상처리설계

실습 1주차

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.
2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.
3. 나는 보고서의 내용을 조작하지 않겠습니다.
4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.
5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2023년 03 월 15 일

학부 정보통신공학
학년 3
성명 김동한
학번 12191727

1. 개요

Homework

실습 및 과제

- 주어진 영상(img1.jpg)에 빨강, 파랑, 초록 색의 점을 각각 설정한 개수만큼 무작위로 생성하는 프로그램을 작성할 것
- 앞서 생성한 영상에서 빨강, 파랑, 초록 색의 점을 각각 카운트하는 프로그램을 작성하고 카운트 결과가 실제와 일치하는지 검증할 것
- 주어진 영상을 이용해(img2.jpg) 다음과 같은 두 영상을 생성하는 프로그램을 작성하고(픽셀 값 접근을 이용) 히스토그램 일치 여부를 확인 및 그러한 결과가 나온 이유를 분석할 것

img2.jpg　　위로 갈수록 점점 어두움　　아래로 갈수록 점점 어두움

- 설정한 개수만큼 무작위로 점을 찍는 함수를 강의노트

```
void SpreadSalts(Mat img, int num) {
        // num: 점을 찍을 개수
        for (int n = 0; n < num; n++) {
                int x = rand() % img.cols; // img.cols는 이미지의 폭 정보를 저장
                int y = rand() % img.rows; // img.rows는 이미지의 높이 정보를 저장
                /*
                나머지는 나누는 수를 넘을 수 없으므로 무작위 위치가
                이미지의 크기를 벗어나지 않도록 제한하는 역할을 하여줌
                */

                if (img.channels() == 1) {
                        // img.channels()는 이미지의 채널 수를 반환
                        img.at<uchar>(y, x) = 255; // 단일 채널 접근
                }
                else {
                        img.at<Vec3b>(y, x)[0] = 255; // Blue 채널 접근
                        img.at<Vec3b>(y, x)[1] = 255; // Green 채널 접근
                        img.at<Vec3b>(y, x)[2] = 255; // Red 채널 접근
                }
        }
}
```

를 참고하고, Blue Green Red channel은 각각 [0] [1] [2] 가 차례로 255 0 0 / 0 255 0/ 0 0 255 인 경우이다. 이를 참고해서 image 속의 점의 개수를 count해 내가 뿌린 점의 개수와 일치하는 지 확인한다.

- 두번째 그라데이션은 먼저 히스토그램 분석을 먼저 해야한다.

```
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;

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

        // 히스토그램 plot
        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()); // 정규화

        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); // 값과 값을 잇는 선을 그리는 방식으로 plot
        }

        return histImage;
}
```

gray_scale로 image를 입력받은뒤 히스토그램을 출력한다. 그 뒤에, rows 가 높이정보를 저장한다는 것을 이용해서 위로 올라갈수록 어두워지는 혹은, 아래로 내려갈수록 어두워지는 이미지를 구현한다.

Homework

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

제출 파일

- 소스코드 파일 (반드시 .cpp, 또는 .h 파일만 첨부)
- 구현 방법 및 결과를 기술한 보고서 (워드 작성 후 PDF로 변환해 제출)


```
Mat imgA = imread("landing.jpg", 1);
Mat imgB = imread("vin.png", 1);
resize(imgB, imgB, Size(imgA.cols, imgA.rows));
// 두 이미지 사이즈를 통일

Mat dst1, dst2, dst3, dst4;
add(imgA, imgB, dst1);
// dst1 = imgA + imgB;도 동일함
add(imgA, Scalar(100, 100, 100), dst2);
// dst2 = imgA + Scalar(100, 100, 100);도 동일함
subtract(imgA, imgB, dst3);
subtract(imgA, Scalar(100, 100, 100), dst4);
```

```
void cv::resize ( InputArray  src,
                  OutputArray dst,
                  Size        dsize,
```

- 영상에 대한 산술 연산을 활용해서 img3 과 비네팅 img4를 합성해서 중앙기준으로 멀어질수록 어두워지도록 배경을 설정한뒤, 영상의 이진화를 이용해서 img5의 로고를 추출해 배경사진에 더해주는 것으로 구현한다.

2. 상세 설계 내용

#1

Gethistogrm

```
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;
}
```

//Blue 점 찍는 함수

```
void SpreadSalts_B(Mat img, int num) {

	for (int n = 0; n < num; n++) {
		int x = rand() % img.cols; //x에 이미지 폭 정보 저장
		int y = rand() % img.rows; //y에 이미지 높이 정보 저장
		/*
		나머지는 나누는 수를 넘을 수 없으므로 무작위 위치가
		이미지의 크기를 벗어나지 않도록 제한하는 역할을 함
		*/
		if (img.channels() == 1) {
			img.at<uchar>(y, x) = 255;
		}
		else {
			img.at<Vec3b>(y, x)[0] = 255; //Blue 채널 접근
			img.at<Vec3b>(y, x)[1] = 0;
			img.at<Vec3b>(y, x)[2] = 0;
		}
	}
}
```

//Red 점 찍는 함수

```
void SpreadSalts_R(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)[0] = 0;
			img.at<Vec3b>(y, x)[1] = 0;
			img.at<Vec3b>(y, x)[2] = 255; //Red 채널 접근
		}
	}
}
```

//Green 점 찍는 함수

```
void SpreadSalts_G(Mat img, int num) {

	for (int n = 0; n < num; n++) {
		int x = rand() % img.cols;
		int y = rand() % img.rows;
		/*
		나머지는 나누는 수를 넘을 수 없으므로 무작위 위치가
		이미지의 크기를 벗어나지 않도록 제한하는 역할을 함
		*/

		if (img.channels() == 1) {
			img.at<uchar>(y, x) = 255;
		}
		else {
			img.at<Vec3b>(y, x)[0] = 0;
			img.at<Vec3b>(y, x)[1] = 255; //Green 채널 접근
			img.at<Vec3b>(y, x)[2] = 0;
		}
	}
}
```

//파랑, 초록, 빨강 점의 개수를 세는 count함수

```
void count(Mat img) {

	int B = 0; int G = 0; int R = 0;
	for (int x = 0; x < img.cols; x++) {
		for (int y = 0; y < img.rows; y++) {
			if (img.at<Vec3b>(y, x)[0] == 255 && img.at<Vec3b>(y, x)[1] == 0 &&
img.at<Vec3b>(y, x)[2] == 0) {
				B++;
			}//255,0,0 은 blue
			else if (img.at<Vec3b>(y, x)[0] == 0 && img.at<Vec3b>(y, x)[1] == 255 &&
img.at<Vec3b>(y, x)[2] == 0) {
				G++;
			} //0,255,0 은 green
			else if (img.at<Vec3b>(y, x)[0] == 0 && img.at<Vec3b>(y, x)[1] == 0 &&
img.at<Vec3b>(y, x)[2] == 255) {
				R++;
```

```
                        } //0,0,255 은 red
                }
        }
        cout << "파랑점의 갯수: " << B << "초록점의 갯수: " << G << "빨강점의 갯수" << R;
}
```

#2

```
void Grad(Mat img) {

        Mat grad1 = img.clone();
        Mat grad2 = img.clone();

        //위로 갈수록 점점 어두움
        for (int i = 0; i < grad1.rows; i++)
        {
                for (int j = 0;j < grad1.cols;j++) {
                        int a = grad1.at<uchar>(i, j);
                        int darkness = (grad1.rows-i)*255/grad1.rows;

                        if (a-darkness > 0)
                        {
                                grad1.at<uchar>(i, j) = a - darkness;
                        }
                        else
                        {
                                grad1.at<uchar>(i, j) = 0;
                        }
                }
        }
        imshow("#2_1", grad1);
        Mat grad1_his = Gethistogram(grad1);
        imshow("#2_1_his", grad1_his);
        waitKey(0);

        //아래로 갈수록 점점 어두움
        for (int i = 0; i < grad2.rows; i++)
        {
                for (int j = 0;j < grad2.cols;j++) {
                        int a = grad2.at<uchar>(i, j);
                        int darkness = i * 255/grad2.rows;


                        if (a - darkness > 0)
                        {
                                grad2.at<uchar>(i, j) = a - darkness;
                        }
                        else
                        {
                                grad2.at<uchar>(i, j) = 0;
                        }
                }
        }
        imshow("#2_2", grad2);
        Mat grad2_his=Gethistogram(grad2);
        imshow("#2_2_his", grad2_his);
        waitKey(0);
}
```

흑백색으로 단일 채널 접근을 한다. int darkness = (grad1.rows-i)*255/grad1.rows; 구문을 통해서 높이가 높아질수록 비례해서 밝기가 어두워지게 한다. 반대로 아래로 갈수록 어두워지게 하기 위해서 int darkness = i * 255/grad2.rows; 구문을 사용한다. 현재 선택 된 i,j 좌표의 화소 밝기값과 새로 설정한 darkness의 차이가 0보다 크면 그 값을 0보다 작으면 0으로 설정해서 이를 imshow를 통해서 출력한다.

강의 노트의 히스토그램 분석함수를 통해서 위나 아래로 점점 밝기가 어두워지는 그림은 같은 히스토그램을 가진다는 것을 알 수 있었다.

#3

```
Mat imgA = imread("C:₩₩images₩₩img3.jpg", 1); //우주선 그림
        Mat imgB = imread("C:₩₩images₩₩img4.jpg", 1); //명암 그림
        Mat logo = imread("C:₩₩images₩₩img5.jpg", 1); // spacex 로고

        resize(imgB, imgB, Size(imgA.cols, imgA.rows));
        Mat dst1; //dst1: 우주선과 명암그림 합성
        subtract(imgA, imgB, dst1);

        Mat roi; //관심영역
        roi = dst1(Rect(Point(350, 300), Point(950, 500)));


        Mat gray_img;
        cvtColor(logo, gray_img, CV_BGR2GRAY);
        resize(gray_img, gray_img, Size(600, 200));

        Mat blackmask;
        Mat whitemask;
        Mat blackmaskc3;
        Mat whitemaskc3;


        threshold(gray_img, blackmask, 220, 255, THRESH_BINARY);
        bitwise_not(blackmask, whitemask); //whitemask는 글씨가 흰색 배경이 검은색인 sapceX 로고


        cvtColor(whitemask, whitemaskc3, COLOR_GRAY2BGR);

        Mat roiblack;
        Mat roiwhite;

        roi.copyTo(roiblack, blackmask);
        roiwhite = whitemaskc3 + roiblack;

        resize(logo, logo, Size(roiwhite.cols, roiwhite.rows));

        logo.copyTo(roi, whitemask);



        imshow("#3", dst1);

        waitKey(0);

        destroyWindow("#3"); // 이미지 출력창 종료

        return 0;
```

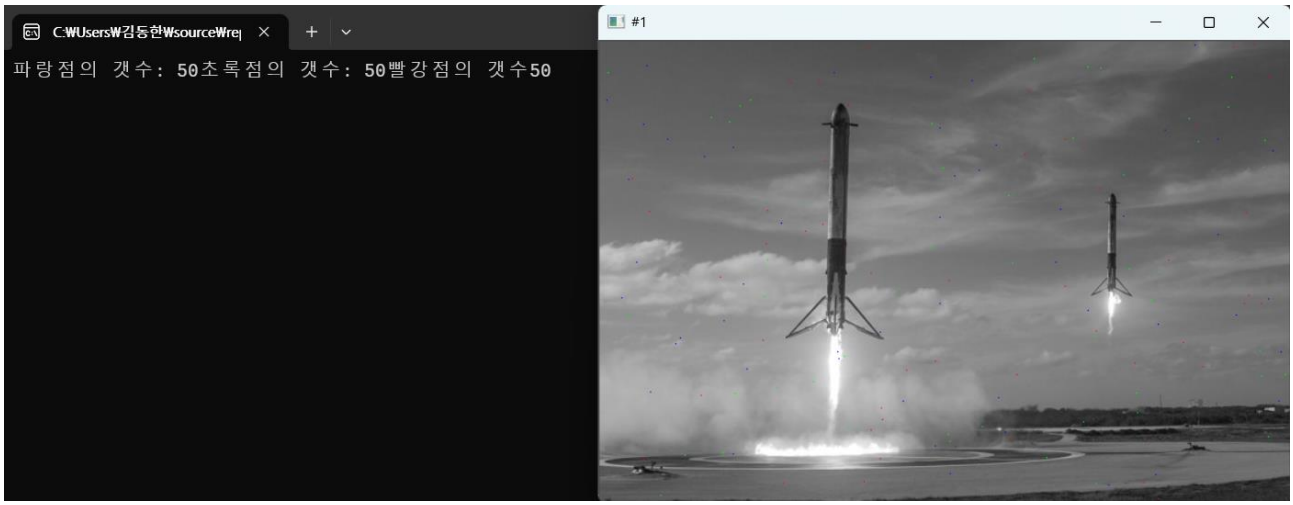
subtract(imgA, imgB, dst1);를 통해서 dst1 에 우주선그림 – 명암그림을 저장한다. 이를 배경화면으로 설정한다. 배경사진에 작은 사진을 하나 추가해야하는 것이기 때문에, 관심영역 roi를 설정해준다.먼저 space x 로고를 BGR2GRAY로 회색으로 바꾼다. x에 회색이 살짝 있기 때문에 이진화 과정에서 자동으로 하는 것이 아니라, 220 을 기준으로 이진화한다. bitwise_not 로 글씨를 흰색 배경을 검은색으로 바꾼뒤에,이 whitemask를 컬러 사진끼리만 연산이 되기 때문에,COLOR_GRAY2BGR로 다시 색이있는 이미지로 바꿔준다.마지막으로 logo.copyTo(roi,whitemask)로 관심영역 roi에 whitemaks에 원본 사진 logo를 합성함으로써 흰색 배경이 삭제된 로고를 사진에 합성할 수 있다.

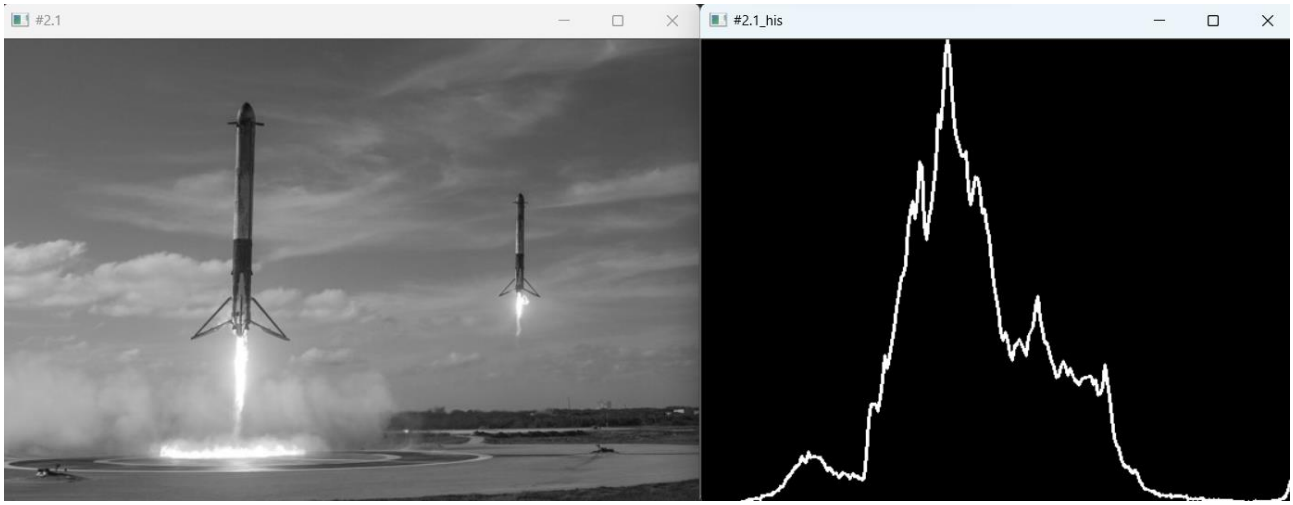
아래는 main함수이다.

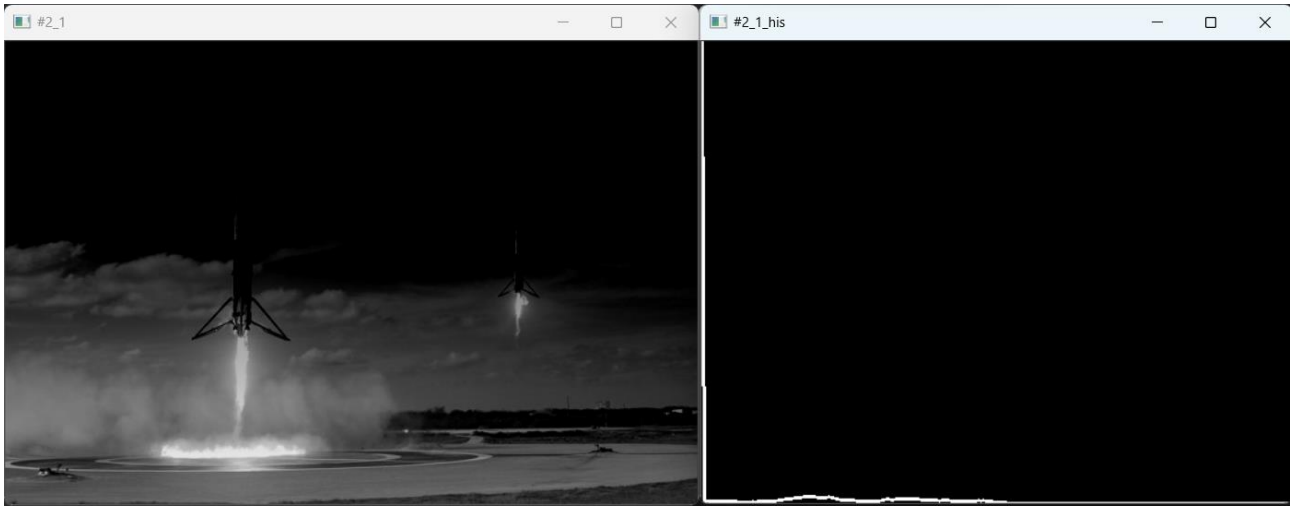
```
int main() {
	Mat src_img1 = imread("C:₩₩images₩₩img1.jpg", 1);
	Mat src_img2 = imread("C:₩₩images₩₩img1.jpg", 0);


	SpreadSalts_B(src_img1, 50);
	SpreadSalts_G(src_img1, 50);
	SpreadSalts_R(src_img1, 50);

	count(src_img1);
	imshow("#1", src_img1);
	waitKey(0);

	Mat image_his = Gethistogram(src_img2);
	imshow("#2.1", src_img2);
	waitKey(0);

	imshow("#2.1_his", image_his);
	waitKey(0);
	/*
	2번 과제
	*/
	Grad(src_img2);
	/*
	3번 과제
	*/
	Mat imgA = imread("C:₩₩images₩₩img3.jpg", 1); //우주선 그림
	Mat imgB = imread("C:₩₩images₩₩img4.jpg", 1); //명암 그림
	Mat logo = imread("C:₩₩images₩₩img5.jpg", 1); // spacex 로고

	resize(imgB, imgB, Size(imgA.cols, imgA.rows));
	Mat dst1; //dst1: 우주선과 명암그림 합성
	subtract(imgA, imgB, dst1);

	Mat roi; //관심영역
	roi = dst1(Rect(Point(350, 300), Point(950, 500)));


	Mat gray_img;
	cvtColor(logo, gray_img, CV_BGR2GRAY);
	resize(gray_img, gray_img, Size(600, 200));

	Mat blackmask;
	Mat whitemask;
	Mat blackmaskc3;
```

```
        Mat whitemaskc3;


        threshold(gray_img, blackmask, 220, 255, THRESH_BINARY);
        bitwise_not(blackmask, whitemask); //whitemask는 글씨가 흰색 배경이 검은색인 sapceX 로고


        cvtColor(whitemask, whitemaskc3, COLOR_GRAY2BGR);

        Mat roiblack;
        Mat roiwhite;

        roi.copyTo(roiblack, blackmask);
        roiwhite = whitemaskc3 + roiblack;

        resize(logo, logo, Size(roiwhite.cols, roiwhite.rows));

        logo.copyTo(roi, whitemask);



        imshow("#3", dst1);

        waitKey(0);

        destroyWindow("#3"); // 이미지 출력창 종료

        return 0;
}
```

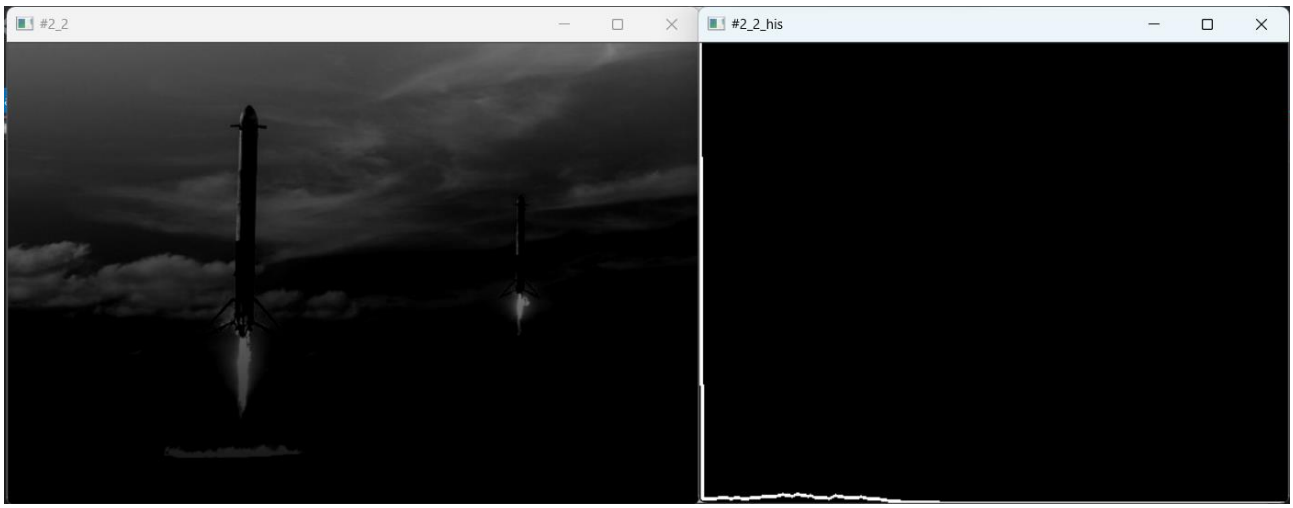
3. 실행 화면







위로갈수록 점점 어두워지는 이미지



아래로 갈수록 점점 어두워지는 이미지

3번 완성본

whitemask

blackmask

roiblack

roiwhite

4. 결론

이번 실습과제에선 픽셀로의 접근을 통해서 사진속에 점을 찍는다거나, 빛의 밝기를 조절하는 것을 구현했다. 또한, 히스토그램을 통해서 밝기에 대한 정보를 확률분포로 볼수 있다. 영상의 논리연산이나 이진화를 통해서 흔히 말하는 누끼를 따는 즉 배경을 지우는 것이 가능했다. 과제 3번에서 배경사진에 특정 관심영역을 설정해둔뒤, 수정한 로고를 copy하는 과정은 처음 해보는것이라 생소하고 어려웠다.