INHA UNIVERSITY

ICE4027 디지털영상처리설계

실습 11주차

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 05월 18일

학부 정보통신공학
학년 3
성명 김동한
학번 12191727

1. 개요

Homework

실습 및 과제

1. coin.png의 동전 개수를 알아내는 프로그램을 구현
2. OpenCV의 1. corner detection과 2. circle detection을 이용해 삼각형, 사각형, 오각형, 육각형의 영상을 순차적으로 읽어와 각각 몇 각형인지 알아내는 프로그램을 구현(도형 4개는 그림판, PPT 등을 이용해 각자 생성할 것)
3. church.jpg에 투시변환(perspective change)과 밝기 변화를 같이 수행한 후 SIFT 특징점을 구했을 때 원본 영상의 SIFT 특징점이 보존되는지 확인해 볼 것(warpPerspective()함수 사용)

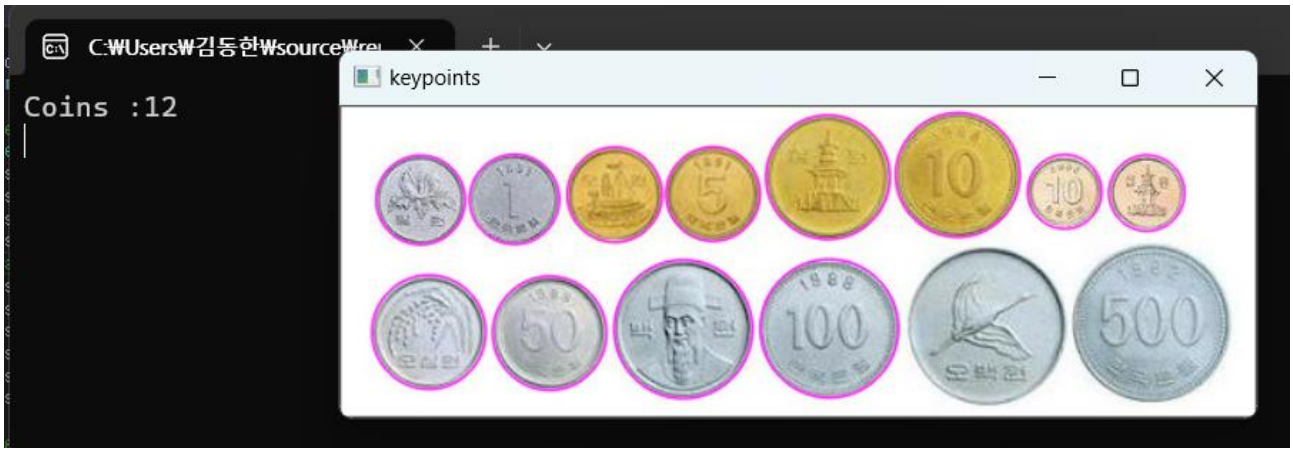
1. 상세한 구현과정과 결과 분석을 반드시 포함할 것
2. 보고서에도 코드와 실험결과 사진을 첨부할 것
3. 반드시 바로 실행가능한 코드(.cpp)를 첨부할 것

2. 상세 설계 내용

#1

제일 먼저 강의노트에 있는 코드대로 parameter를 설정해서 수행해본 결과 minCircularity값이 너무 크게 설정되어있어서 완전한 원의 형태가 아닌 몇 개의 동전은 detect하지 못한 것을 알 수 있었다.

minCircularity값을 따라서 0.5까지 낮춰 수행해본결과 이번에는 동전이 아닌데도 불구하고 원의 형태에 근접해서 벼이삭과 같은 point들이 detect되는 것을 알 수 있었다. 따라서 minArea의 크기를 100으로 증가시켜줬다.



이번에는 500원크기의 동전들이 detect되지 않은 것을 확인할 수 있었다. 따라서 마지막으로 maxArea parameter를 선언한뒤 그 크기를 10000으로 설정하였다.

완벽하게 coin 14개를 detect한 것을 확인 할 수 있었다. 여기서 coin의 개수를 count할 때는, detect한 원에 해당하는 keypoints의 size를 출력함으로써 구현했다.

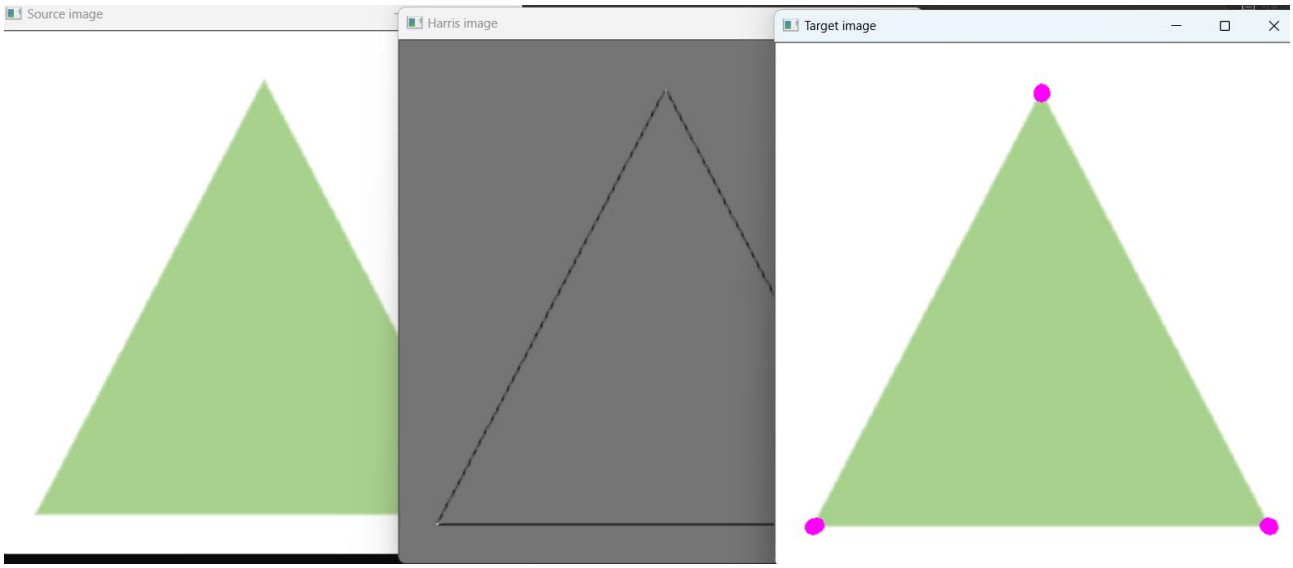
```
int cvBlobDetection(Mat img) {

	// <Set params>
	SimpleBlobDetector::Params params;
	params.minThreshold = 10;
	params.maxThreshold = 300;
	params.filterByArea = true;
	params.minArea = 100;
	params.maxArea = 10000;
	params.filterByCircularity = true;
	params.minCircularity = 0.5;
	params.filterByConvexity = true;
	params.minConvexity = 0.9;
	params.filterByInertia = true;
	params.minInertiaRatio = 0.01;

	// <Set blob detector>
	Ptr<SimpleBlobDetector> detector = SimpleBlobDetector::create(params);

	// <Detect blobs>
	std::vector<KeyPoint> keypoints;
	detector->detect(img, keypoints);

	// <Draw blobs>
	Mat result;
	drawKeypoints(img, keypoints, result,
		Scalar(0, 255, 255), DrawMatchesFlags::DRAW_RICH_KEYPOINTS);

	imshow("keypoints", result);
	waitKey(0);
	destroyWindow("keypoints");
	return keypoints.size();
}
```

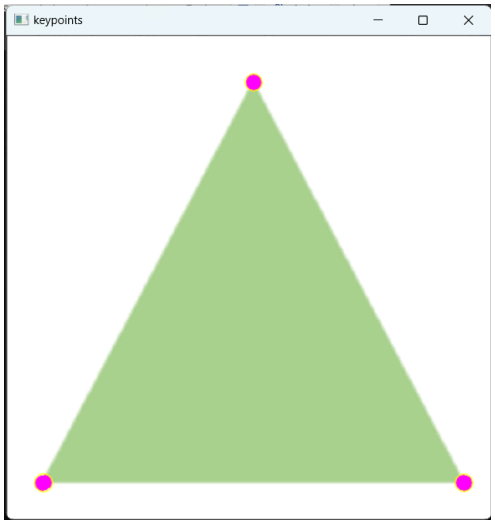
#2

```
Mat cvHarrisCorner(Mat img) {
	if (img.empty()) {
		cout << "Empty image!₩n";
		exit(-1);
	}
	resize(img, img, Size(500, 500), 0, 0, INTER_CUBIC);

	Mat gray;
	cvtColor(img, gray, CV_BGR2GRAY);

	// < Do Harris corner detection >
```

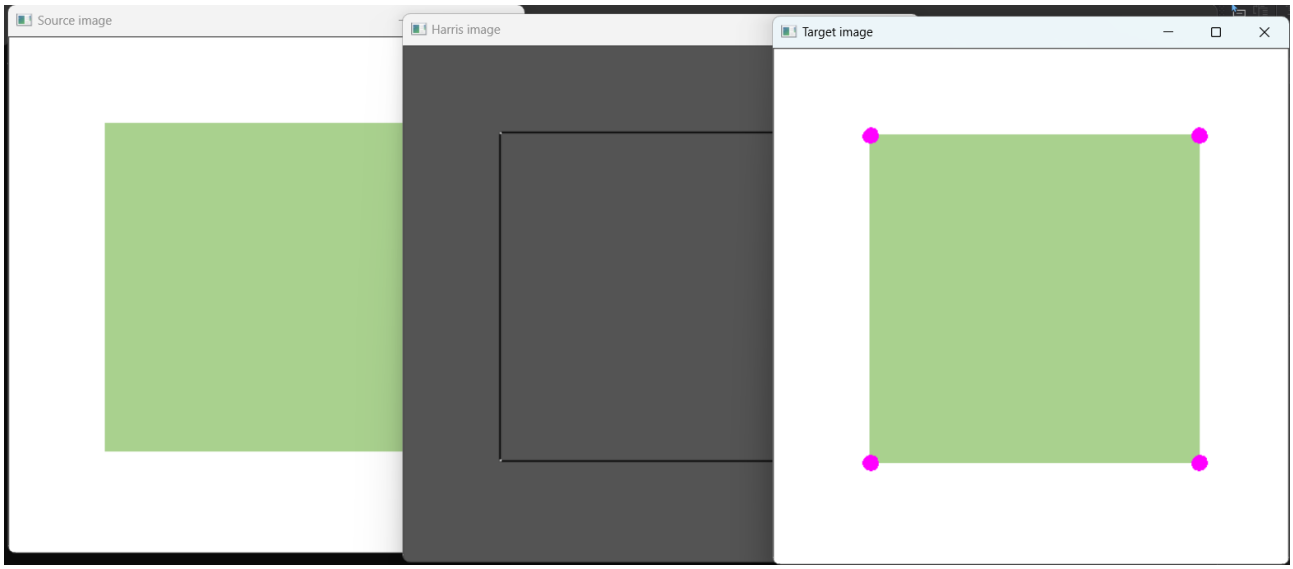
```
	Mat harr;
	cornerHarris(gray, harr, 2, 3, 0.05, BORDER_DEFAULT);
	normalize(harr, harr, 0, 255, NORM_MINMAX, CV_32FC1, Mat());

	// < Get abs for Harris visualization >
	Mat harr_abs;
	convertScaleAbs(harr, harr_abs);

	// < Print corners >
	int thresh = 125;
	Mat result = img.clone();
	for (int y = 0; y < harr.rows; y+=1)
	{
		for (int x = 0;x < harr.cols;x+=1) {
			if ((int)harr.at<float>(y, x) > thresh)
				circle(result, Point(x, y), 7, Scalar(255, 0, 255), -1, 4, 0);
		}
	}
	imshow("Source image", img);
	imshow("Harris image", harr_abs);
	imshow("Target image", result);
	waitKey(0);
	destroyWindow("Source image");
	destroyWindow("Harris image");
	destroyWindow("Target image");
	return result;
}
```

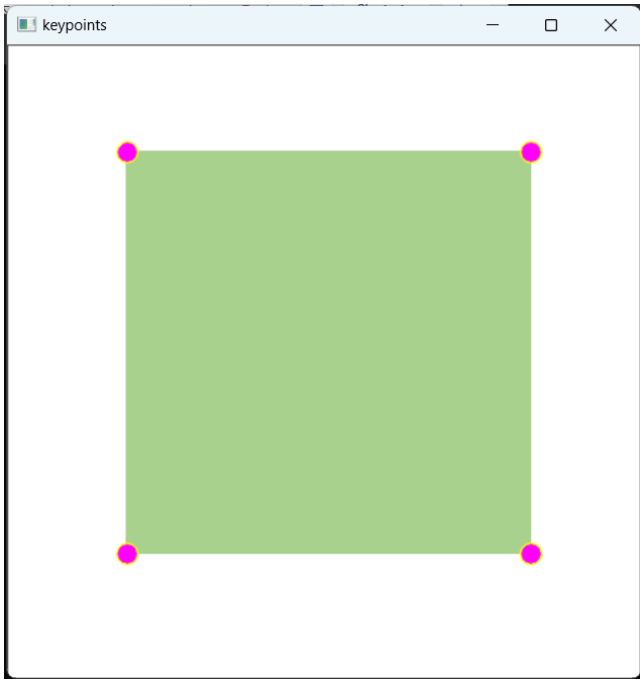
삼각형



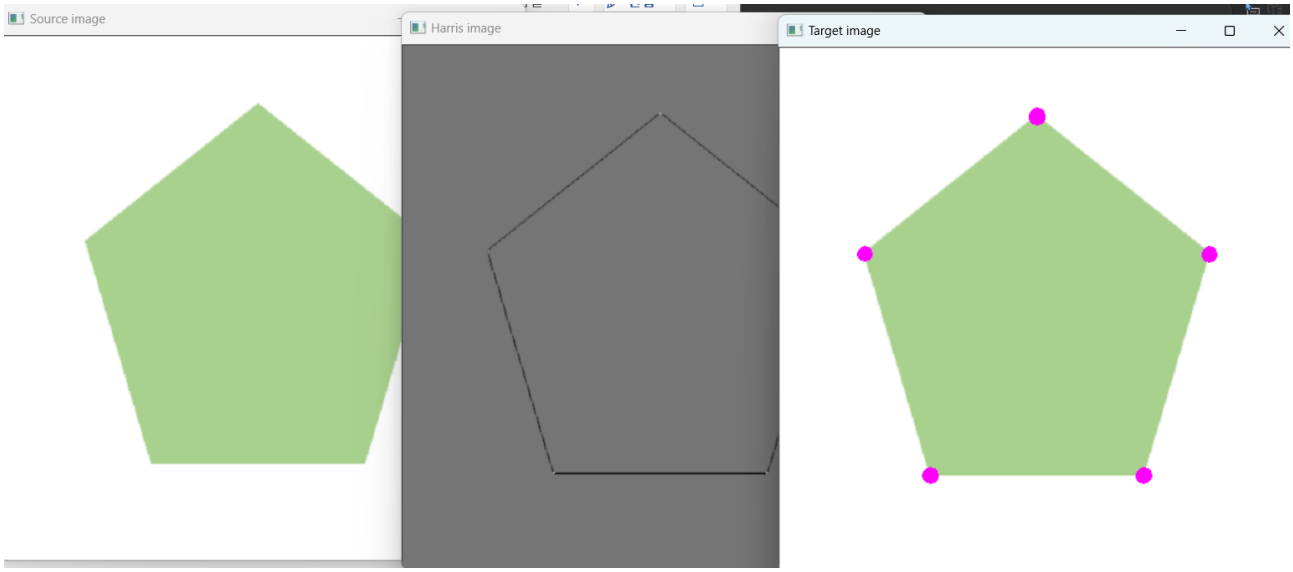

keypoints

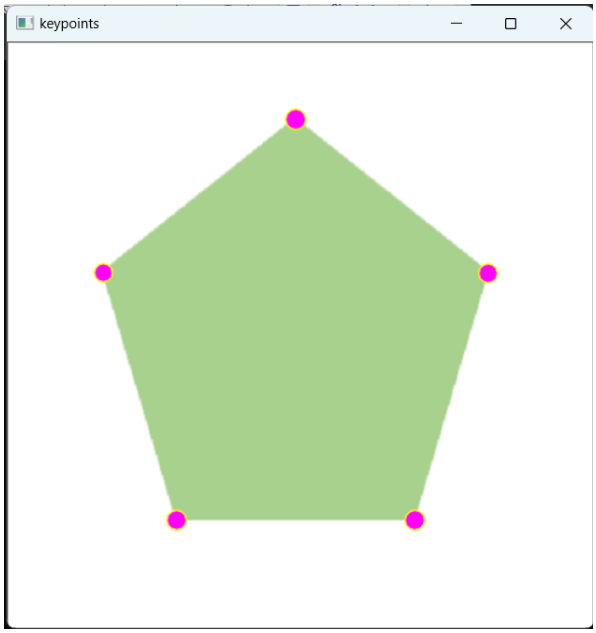
사각형


Source image
Harris image
Target image

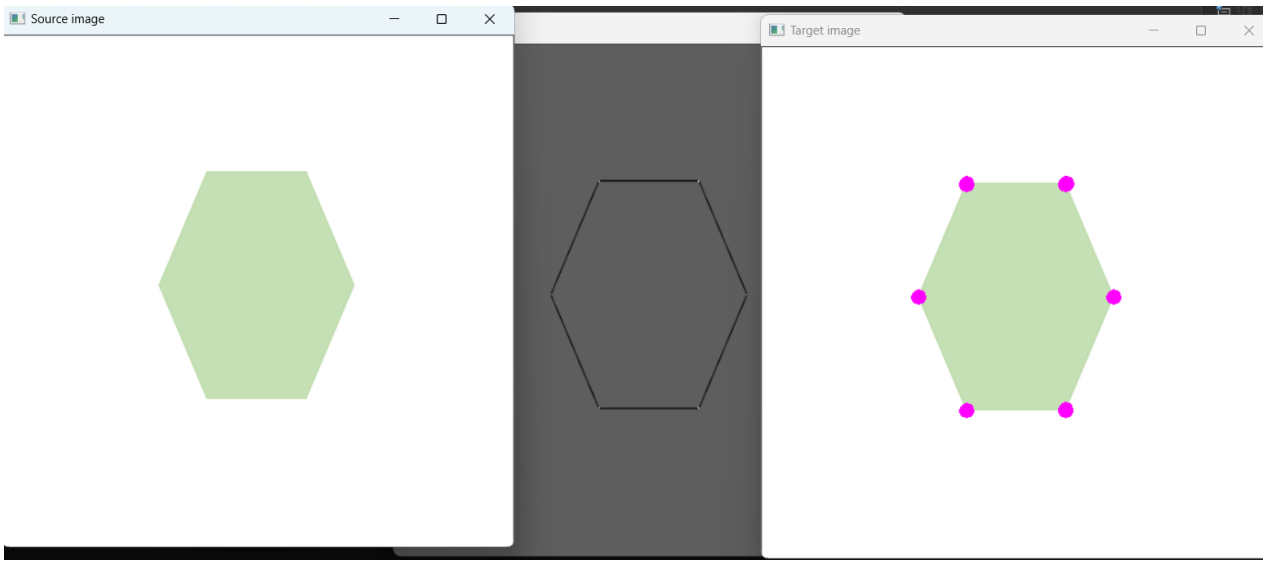

keypoints

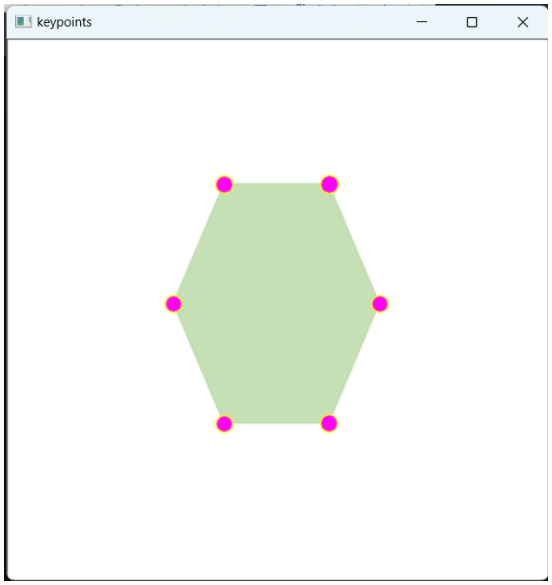
오각형


Source image
Harris image
Target image


keypoints

육각형

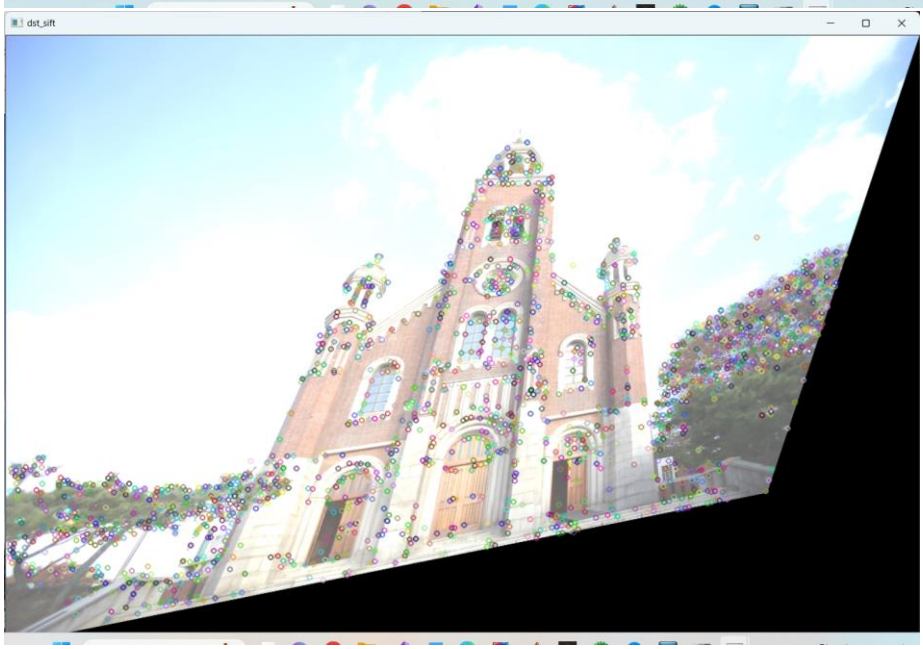

Source image
Target image




```
Coins :14
3각형
4각형
5각형
6각형

C:\Users\김동한\source\repos\ocv_project\x64\Debug\ocv_project.exe(5340 프로
이 창을 닫으려면 아무 키나 누르세요.
```

```
//#2
Mat img_2 = imread("C:▩▩images▩▩3.png");
Mat dst_2 = cvHarrisCorner(img_2);
cout << cvBlobDetection(dst_2) << "각형" << endl;
Mat img_3 = imread("C:▩▩images▩▩4.png");
Mat dst_3 = cvHarrisCorner(img_3);
cout << cvBlobDetection(dst_3) << "각형" << endl;
Mat img_4 = imread("C:▩▩images▩▩5.png");
Mat dst_4 = cvHarrisCorner(img_4);
cout << cvBlobDetection(dst_4) << "각형" << endl;
Mat img_5 = imread("C:▩▩images▩▩6.png");
Mat dst_5 = cvHarrisCorner(img_5);
cout << cvBlobDetection(dst_5) << "각형" << endl;
```

cvHarrisCorner 함수의 반환값으로 corner를 detect한 이미지를 반환한다. 그 뒤, cvBlobDetection 함수의 반환값으로 원의 개수를 반환하게 한다. 이때 cvBlobDetection의 인자로 corner에 원을 그려둔 dst_image를 입력한뒤, 각 다각형의 코너의 원을 인식해 몇각형인지 알아내도록 구현했다. 이때 만들어둔 다각형을 인자로 전달해줄 때, pixel을 resize하는 과정에서 다각형의 선이 일그러져 코너 인식을 못하는 경우도 종종 생겨서 source로 주는 이미지 자체를 500x500 pixel값의 이미지를 전달해줬다.



원본의 영상과 warp및, 밝기변화를 시킨 이미지에 큰 차이는 없었지만, 밝기를 더 밝게 설정하며 구름이 하늘과 비슷한 화소값을 가지게 되어 특징점이 잡히지 않게 되었다.