INHA UNIVERSITY

ICE4027 디지털영상처리설계

실습 13주차

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 06월 01일

학부 정보통신공학
학년 3
성명 김동한
학번 12191727

1. 개요

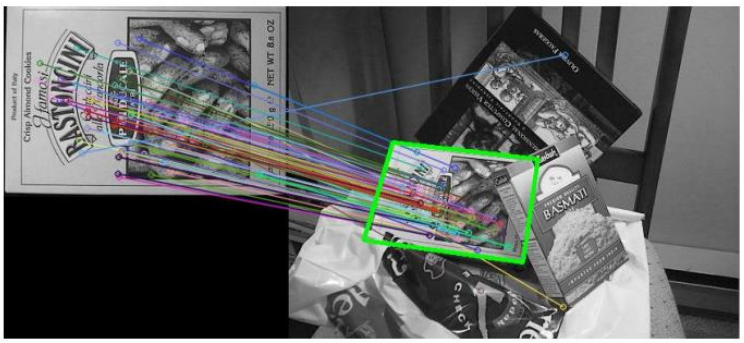
Homework

- 과제 1
  - 직접 촬영한 영상 세 장으로 panorama stitching을 수행해볼 것
  - 금일 실습 두 가지 방법을 각각 적용하고 분석할 것
  - Tip. 주변 대상이 적어도 5m 이상 떨어져 있고 특징점이 많이 추출될 수 있는 장면에서 수행할 것

영상 3개

Homework

- 과제 2
  - Book1.jpg, Book2.jpg, Book3.jpg가 주어졌을 때 Scene.jpg에서 이것들을 찾아 아래의 그림처럼 윤곽을 찾아 그려주는 프로그램을 구현할 것
  - SIFT 특징점 추출, brute force 매칭, findHomograpy()를 사용해 구현할 것
  - 상세한 코드 설명과 주석을 첨부할 것



2. 상세 설계 내용

#1

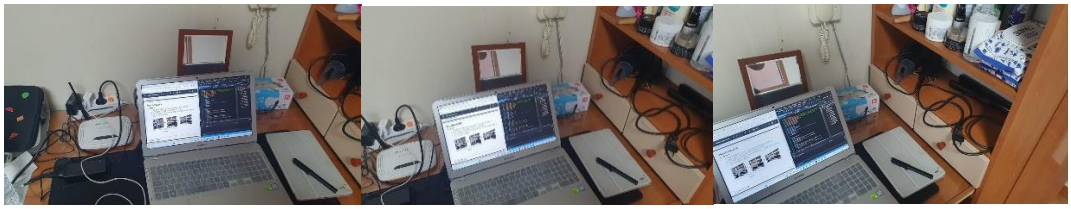


Figure 1 source images

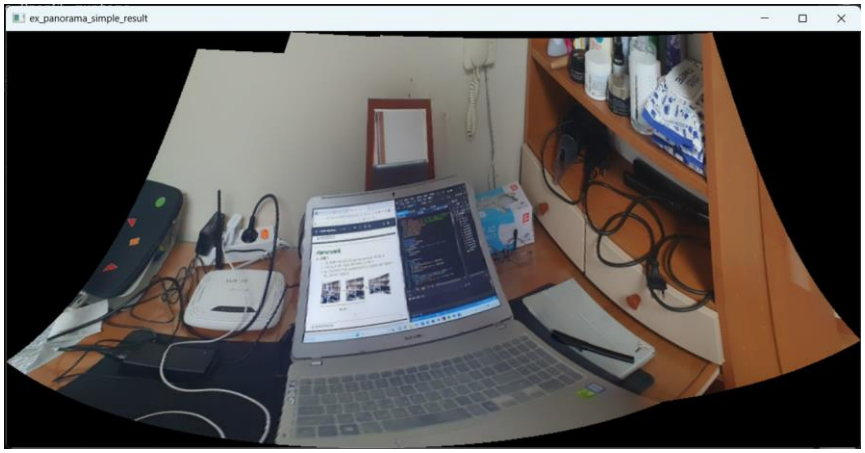


Figure 2 Stitcher

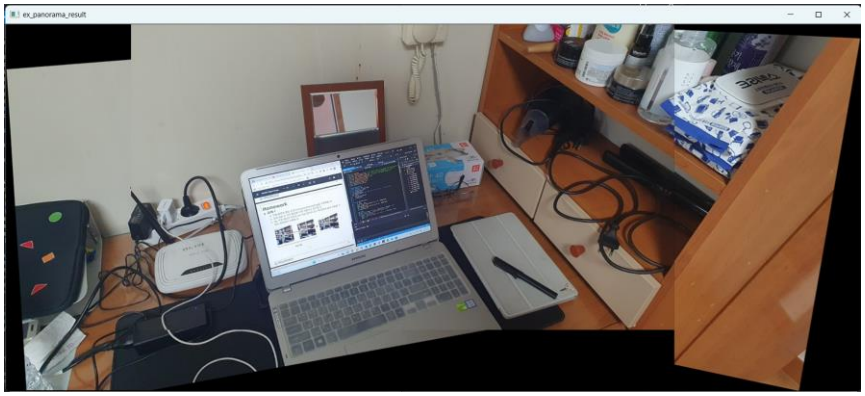


Figure 3 SURF 특징점 검출 통한 stitching

세 이미지가 둘을 비교했을 때, 물체의 특징점을 맞춘 방법이 더 자연스럽게 stitching이 이루어진 것을 확인할 수 있다.

#2

makePanorama 함수를 수정해서 sift를 사용해 특징점을 추출하고, homography matrix로 변환하는 것처럼 책 모서리를 통해서 line함수로 scene 이미지에 책의 윤곽선을 그리는 것으로 구현했다.

```
//<특징점(KEY POINT) 추출> SIFT
Ptr<SiftFeatureDetector> Detector = SIFT::create(300);
vector<KeyPoint> kpts_obj, kpts_scene;
Detector->detect(img_gray_l, kpts_obj);
Detector->detect(img_gray_r, kpts_scene);
```

key point 추출시 SiftFeatuerDetector를 사용했다. (기존에 SURFdetect였음)

```
//기술자 추출 SIFT
Ptr<SiftDescriptorExtractor> Extractor = SIFT::create(100, 4, 3, false, true);
```

기술자 추출하는 부분도 SIFT로 바꿨다.

```
// corner point 저장
vector<Point2f> obj_corners(4);
obj_corners[0] = Point(0, 0);
obj_corners[1] = Point(img_l.cols, 0);
obj_corners[2] = Point(img_l.cols, img_l.rows);
obj_corners[3] = Point(0, img_l.rows);
```

또한, scene에서 찾고자 하는 object의 corner point를 저장하였다.

```
//<homograpy 행렬을 이용해 시점 역변환>
Mat img_result;
warpPerspective(img_r, img_result, mat_homo,
    Size(img_l.cols + 2, img_l.rows + 1.2), INTER_CUBIC);
//영상이 잘리는 것을 방지하기 위해 여유공간을 부여
```

```
// < homography 행렬을 통하여 이미지 warping>
perspectiveTransform(obj_corners, scene_corners, mat_homo);
```

homoraphy행렬을 이용해서 이미지 warping을 수행하는 코드로 수정했다.

```
// < scene에서 mapping된 object의 코너 사이의 선을 그린다 >
line(img_matches_good, scene_corners[0] + Point2f(img_object.cols, 0),
    scene_corners[1] + Point2f(img_object.cols, 0), Scalar(255,0 , 0), 3);
line(img_matches_good, scene_corners[1] + Point2f(img_object.cols, 0),
    scene_corners[2] + Point2f(img_object.cols, 0), Scalar(255, 0, 0), 3);
line(img_matches_good, scene_corners[2] + Point2f(img_object.cols, 0),
    scene_corners[3] + Point2f(img_object.cols, 0), Scalar(255, 0, 0), 3);
line(img_matches_good, scene_corners[3] + Point2f(img_object.cols, 0),
    scene_corners[0] + Point2f(img_object.cols, 0), Scalar(255, 0, 0), 3);
```

```
contour(src_img1, scene_img, 2, 80); // 임계값 조정
contour(src_img2, scene_img, 3, 80);
contour(src_img3, scene_img, 3, 80);
waitKey(0);
```

매칭된 결과를 임계값을 조정하여 scene에서 Book1 Book2 Book3를 각각 찾는 코드를 구현했다.

```
//<기술자를 이용한 특징점 매칭>
BFMatcher matcher(NORM_L2);
vector<DMatch> matches;
matcher.match(img_des_obj, img_des_scene, matches);
```

BFMatcher 객체를 생성해서 match함수를 사용해 기술자를 매칭하고, Brute Force Matcher를 사용해 img_des_obj와 img_des_scene의 기술자를 매칭했다.

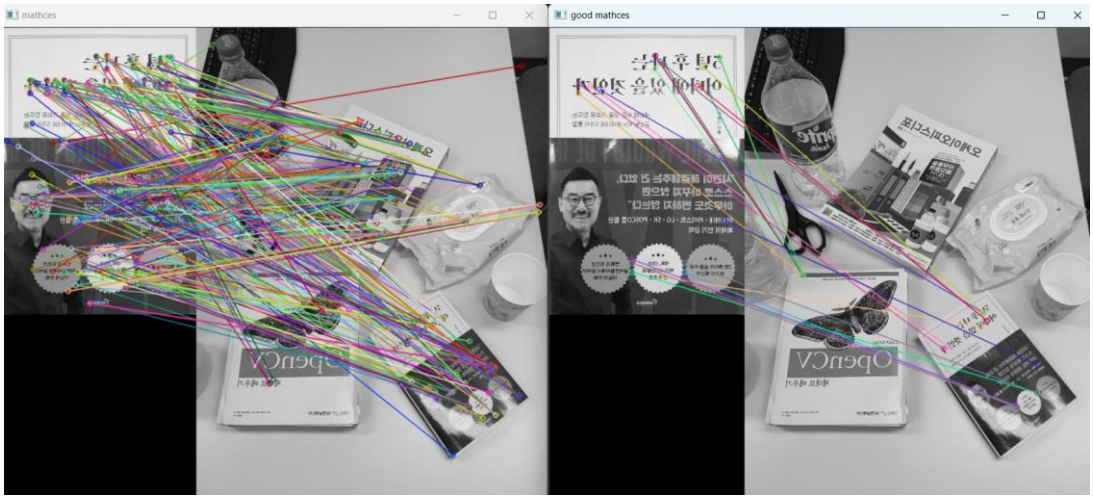


그림 1 L:매칭 결과 R:정제된 매칭결과

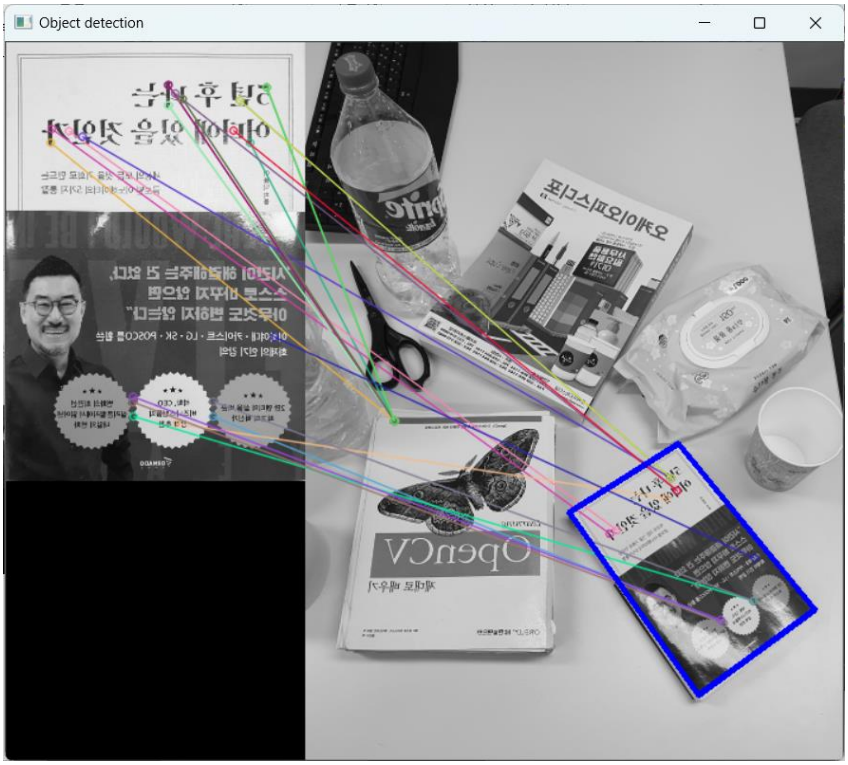


그림 2 object detection

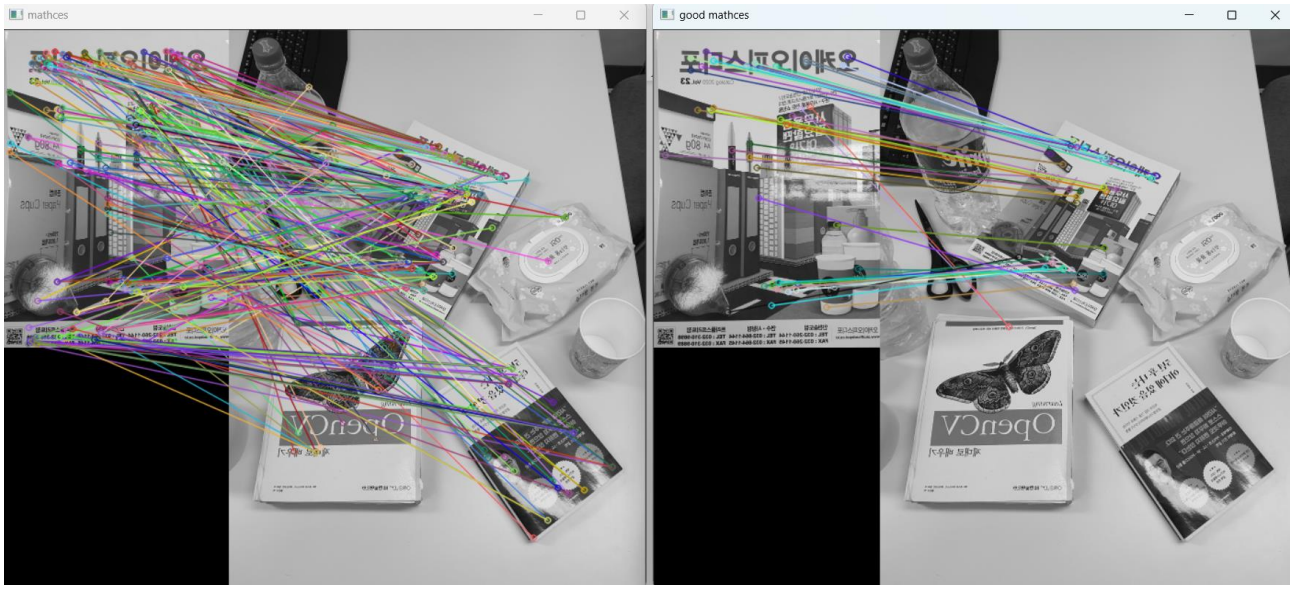


그림 3 L:매칭 결과 R:정제된 매칭결과

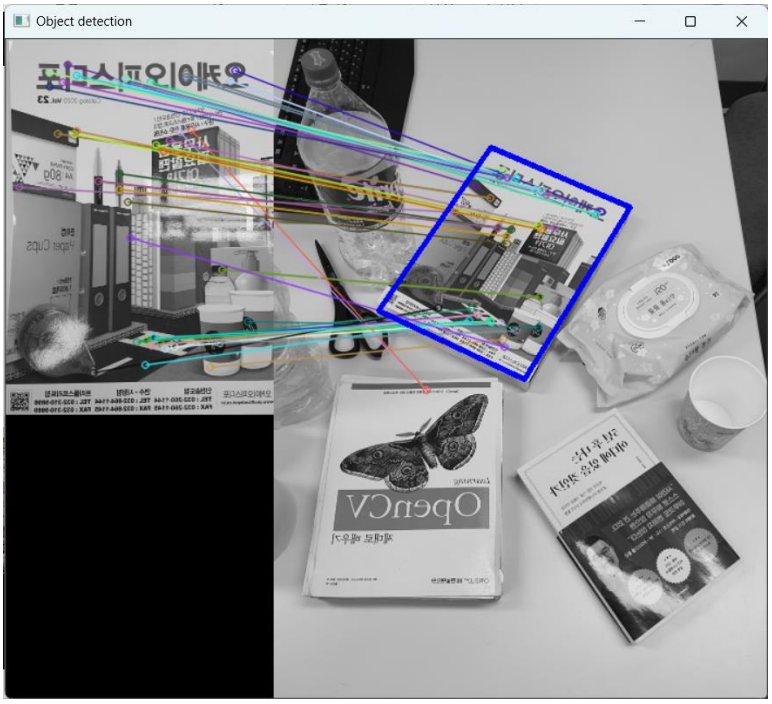


그림 4 object detection

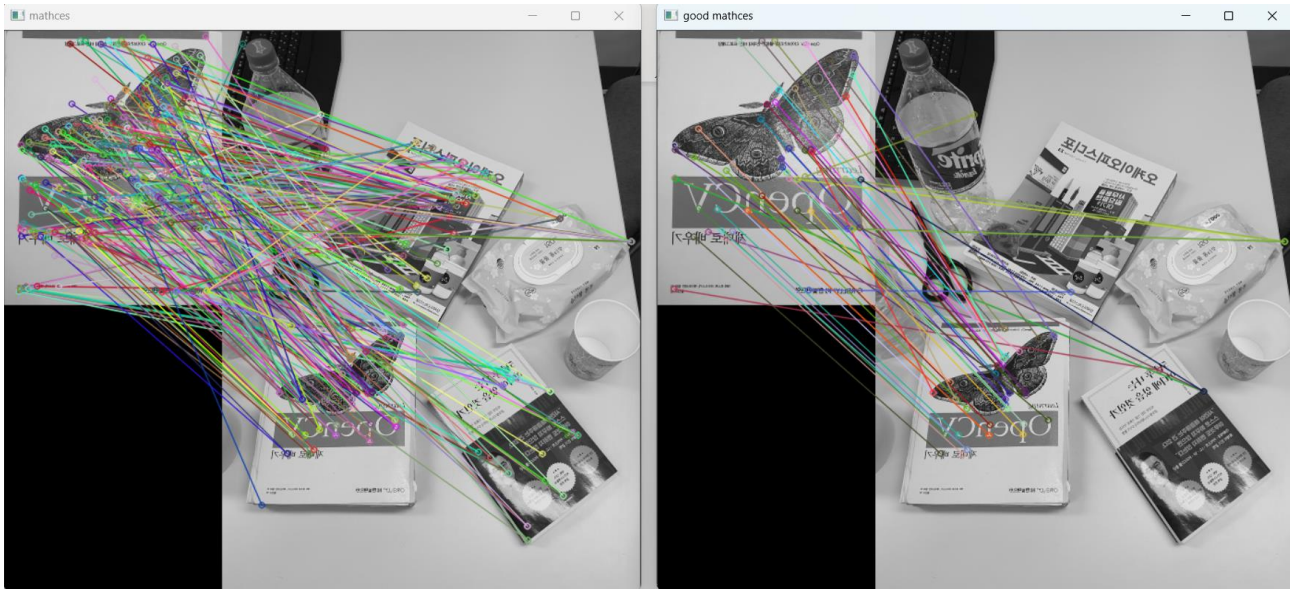


그림 5 L:매칭 결과 R:정제된 매칭 결과

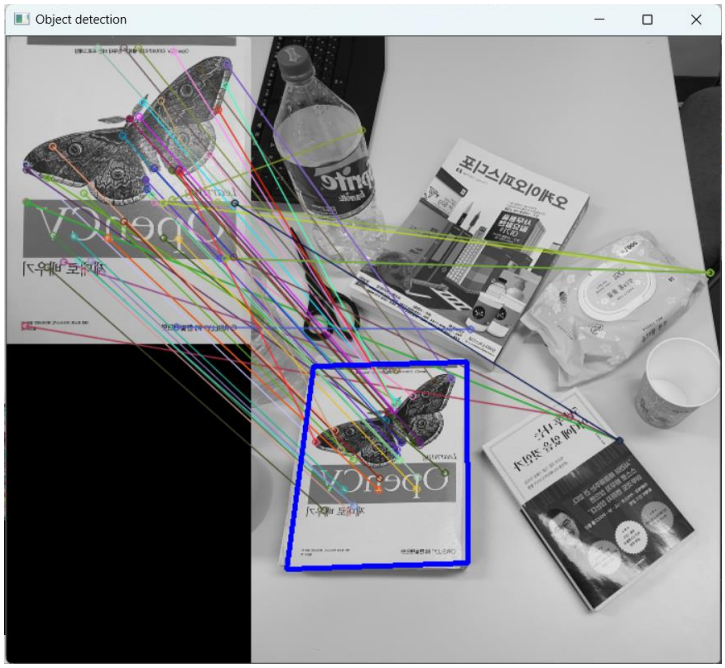


그림 6 object detection

이과 같이 scene jpg에서 Book을 잘 찾음을 확인할 수 있었다.