**Open Software Project**

**Assignment 6 Technical Report**

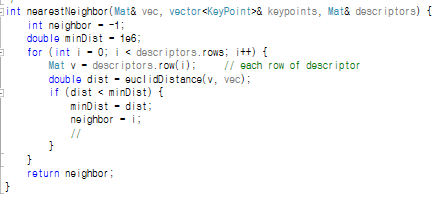
**경영학부 1517034 정성원**

**1. SIFT**

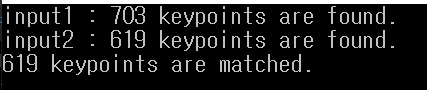
**1) Case 1**

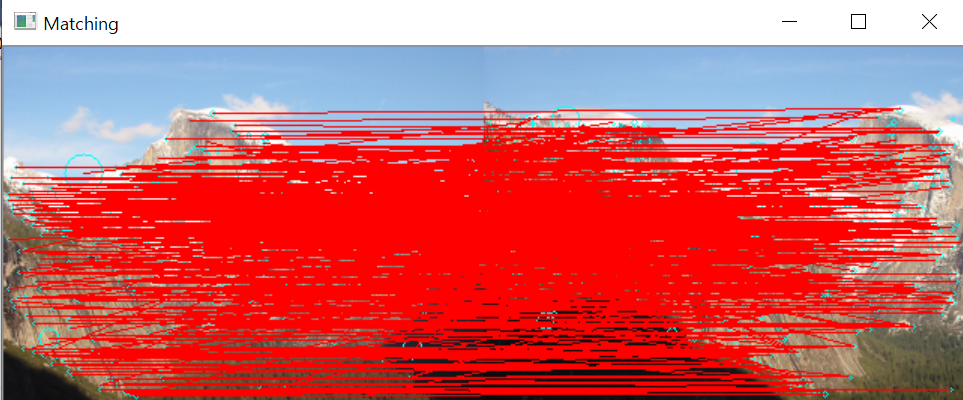
**A. Code**

descriptors의 i 번째 row는 i 번째 keypoints의 descriptor이다. 주어진 vec과 가장 거리가 가까운 descriptor의 row index를 반환



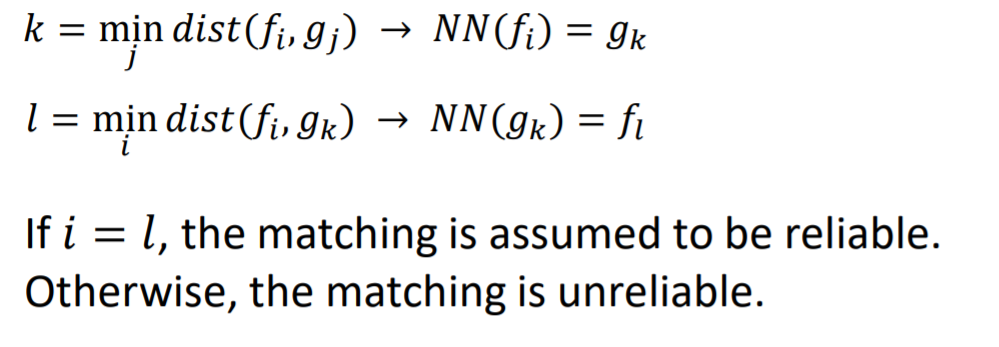
**B. Result**

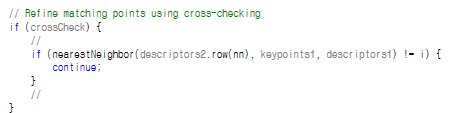




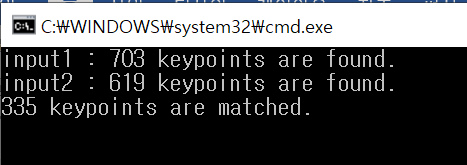
**2) Case 2 – using Cross-Checking**

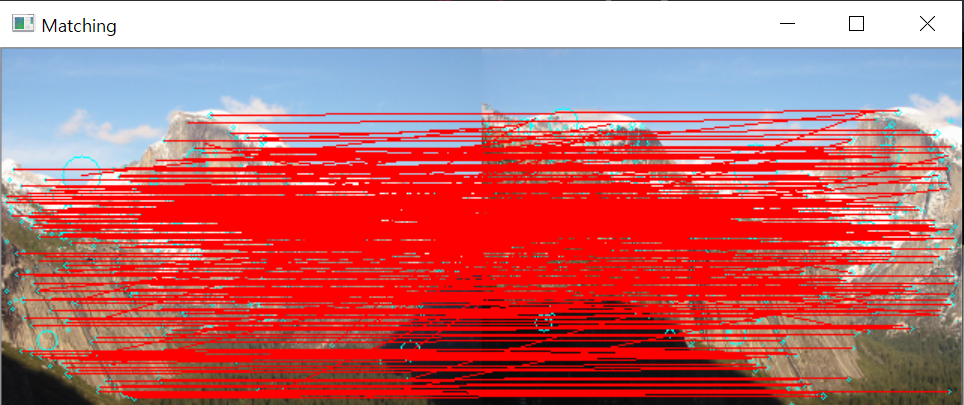
**A. Code**





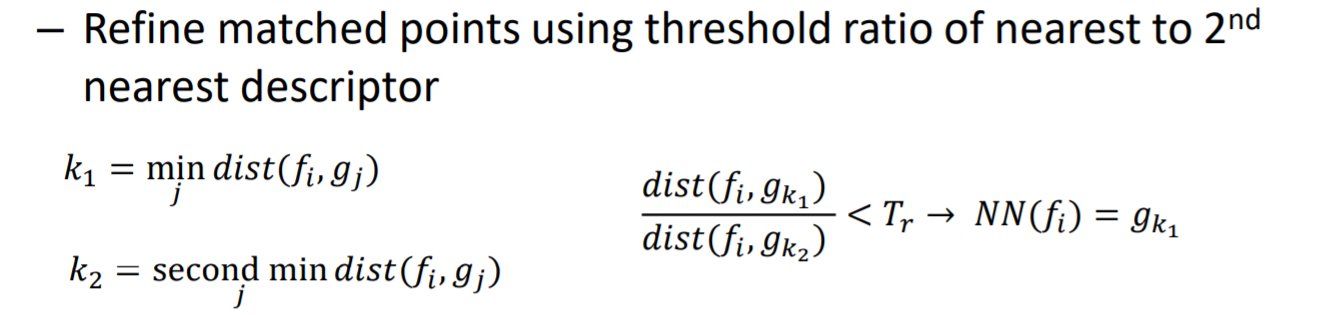
**B. Result**

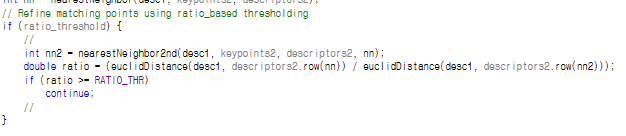




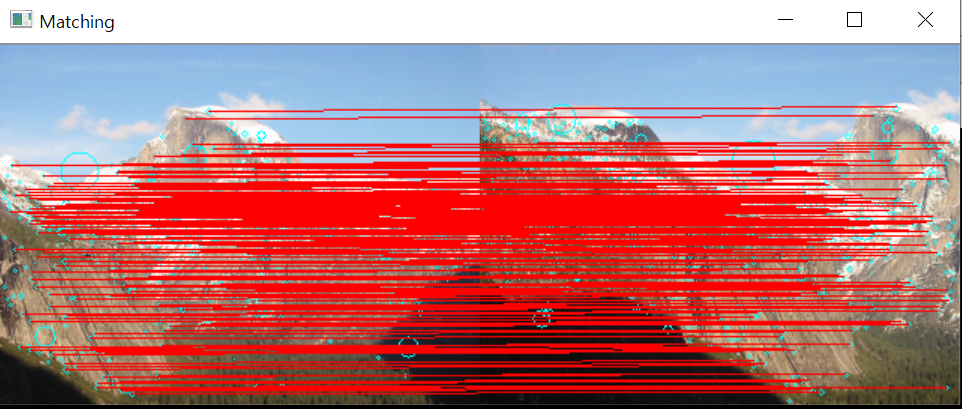
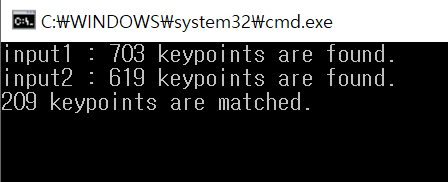
**3) Case 3 – using Cross-Checking and Ratio-based Thresholding**

**A. Code**





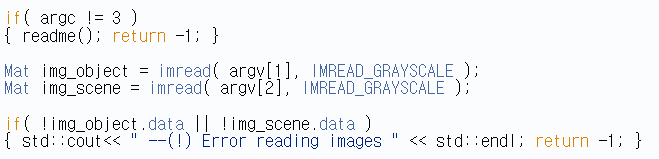
**B. Result**



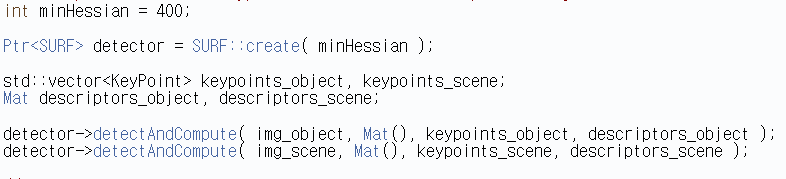
**2. Features2d + Homography to find a known object**

**A. Code Analysis**

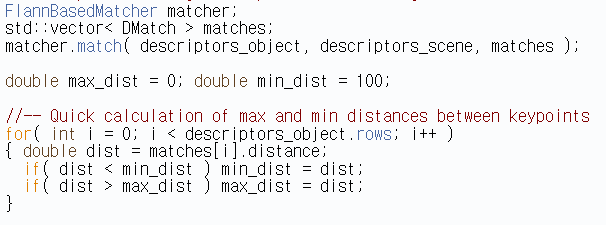
-<https://docs.opencv.org/3.4.1/d7/dff/tutorial_feature_homography.html>의 코드를 실행하고 분석한다. 해당코드는 cv::findHomography를 이용해 matched keypoints 간 transform을 찾고 cv:: perspective Transform을 이용해 point들을 mapping한다.

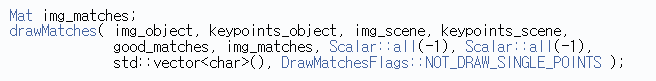


먼저 이용할 이미지들을 불러와 각각 img\_object, img\_scene이라는 Mat에 저장한다.

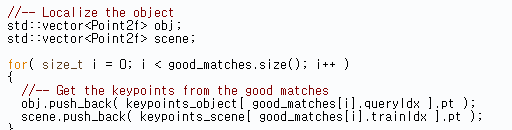


다음엔 SURF를 이용해 img\_object와 img\_scene의 특징점(keypoint)을 검출한다. 입력 이미지의 적분 영상을 만든 후 Hessian Detector를 이용해 특징점을 검출한다. 검출된 특징점들은 각각 keypoints\_object와 keypoints\_scene에 keypoint의 descriptors는 각각 descriptor\_object, descriptor\_scene에 저장된다.

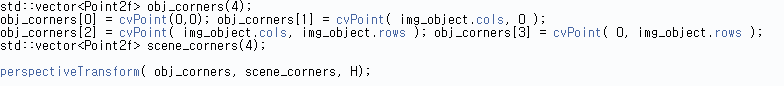




img\_object와 img\_scene에서 검출한 특징을 FLANN Matcher를 이용해 매치한다. 매치된 특징점들 중 good match(whose distance is less than 3\*min\_dist)만 선택하기 위해 특징점 간 거리의 최대값과 최소값을 구한다. 이후 good match인 것들만 선택한다.

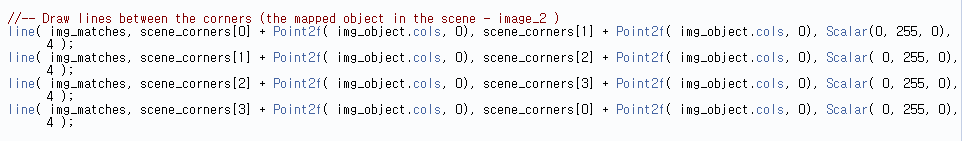


선택된 Good match의 keypoints 위치를 obj과 scene에 각각 저장한다. obj과 scene에 저장된 keypoint와 RANSAC을 이용해 변환관계(Homography)를 찾는다. 찾은 Homography는 Mat H에 저장한다.

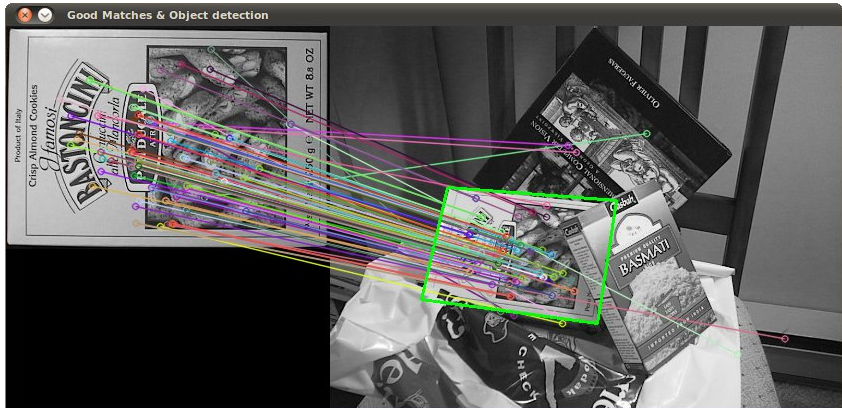


detect될 이미지(img\_object)에서 corner를 찾은 후 코너의 위치와 H를 이용해 img\_scene에서 대응되는 코너를 찾는다.

이후 시각화를 위해 corners를 잇는 선을 그린다.



코드 실행 결과는 아래와 같다.



(코드 실행에 실패하여 해당 링크에 있는 이미지를 첨부했습니다.)