# Pattern Recognition Project Report

## [Face recognition program development]

**서강대학교**

**모바일 컴퓨팅 & 데이터 공학 연구실**

**120150241 김상근**

1. **프로젝트 목표**

얼굴을 인식하는 프로그램을 개발하고, 주어진 얼굴 이미지에 대하여 성능을 측정하여라. 주어진 얼굴 이미지는 트레이닝 이미지로 fa image, fa image에서 감정변화가 있는 fb, 좌측면 이미지인 ql 그리고 우측면 이미지인 qr image가 주어진다. 또, 프로젝트 진행을 위하여 눈의 위치 정보인 ground\_truths 도 주어진다.

첫번째로 수행해야될 목표는 fb, qr, ql 총 300장의 이미지를 fa에서 검색하여 세 개의 Score matrices와 CMC Curve를 그리는 것이다. 두번째 목표는 첫번째로 수행한 이미지 매칭 알고리즘의 성능을 향상시킨 뒤, 향상된CMC Curve를 그리는 것이다.

1. **개발 범위 및 내용**
   1. 개발 범위

fb, ql, qr 이미지를 query point로 하여 fa 이미지에서 해당 이미지들을 검색하여 Score matrices와 CMC curve를 구하는 프로그램을 개발한다. 그리고 성능 향상을 위하여 공부하고, 수정하여 향상된 CMC Curve를 구한다.

* 1. 개발 내용
     1. 개발 환경

Programming Language : python 2.7.6 version

Opencv : opencv 2.4.11 version

Python library : numpy (숫자 관련 라이브러리)

Matplotlib (그래프 관련 라이브러리)

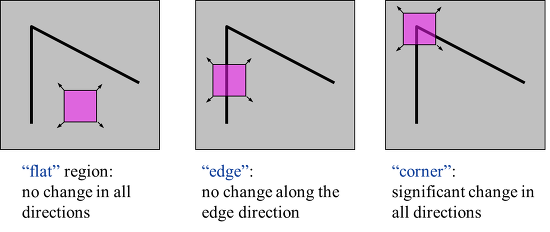
* + 1. Feature 검출 알고리즘과 이미지 매칭 알고리즘

이미지 매칭 프로그램의 순서는 다음과 같다. 먼저 모든 이미지에서 특징점들을 뽑아내고, fb, ql, qr의 Query 이미지의 특징점과 fa Gallery 이미지의 특징점을 매치하여 가장 매치된 양이 많은 이미지를 유사한 이미지로 정한다. 이 매치된 양을 기준으로 rank-k를 구하고, CMC Curve를 구할 수 있다.

특징점 검출 알고리즘으로 Harris Corner, Haar Feature, SIFT(Scale-Invariant Feature Transform)과 SURF(Speedup Robust Features)등이 존재한다. 특징점을 검출 한 후, 특징점끼리 매칭하는 알고리즘으로는 RANSAC 등이 있다.

1. **특징점 검출 및 매칭 알고리즘 정리**
   1. Harris Corner

영상에서 특징점을 찾는 가장 대표적인 방법으로 1988년에 C. Harris and M. Stephens, “A Combined corner and edge detector”에서 발표된 Harris Corner 방법이다. 영상에서 코너를 찾는 기본적인 아이디어는 아래 그림과 같이 영상에서 윈도우를 조금씩 Shift했을 때, 코너의 경우에는 모든 방향으로 영상 변화가 커야 한다는 점을 이용한다.

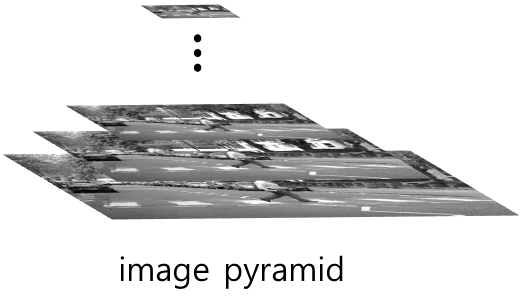


위 과정을 구현하기 위해서 영상의 각 픽셀 위치에 대해 윈도우를 수직, 수평, 좌대각선, 우대각선 이렇게 4개 방향으로 1 픽셀씩 이동시켰을 때의 영상변화량(SSD) E를 계산한 후, E의 최소값을 해당 픽셀의 영상변화량 값으로 설정한다. 설정된 min(E)값이 지역적으로 극대과 되는 지점을 코너점으로 찾는다. 이 방법을 수학적 지식을 통하여 수정 보완한 방법이 Harris Corner 알고리즘이다.

* 1. SIFT(Scale-Invariant Feature Transform)

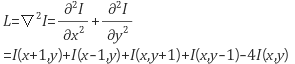
널리 알려진 특징점 추출 알고리즘이다. Harris corner의 영상 스케일 변화에 민감한 문제를 해결하기 위하여 DOG(Difference of Gaussian)을 기반으로 이미지 내에서 뿐만 아니라 Scale 축으로도 코너성이 극대인 점을 찾는다.

어떤 이미지 I가 있을 때, I의 크기를 단계적으로 축소시켜서 축소된 이미지들을 생성할 수 있다. 이렇게 생성된 이미지들을 이미지 피라미드라 부르고, 각 스케일마다 코너점들을 찾는다.

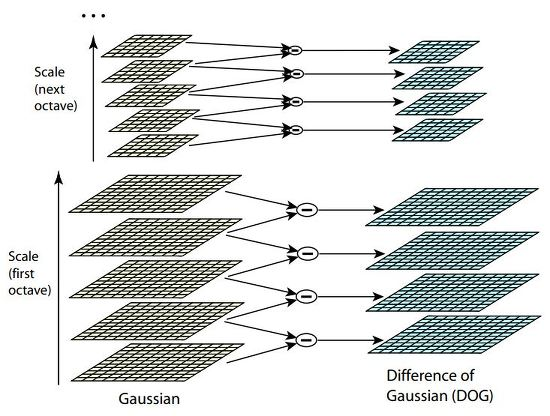


대부분의 경우, 인접한 여러 이미지 스케일에 걸쳐서 동일한 지점이 코너점으로 검출 된다. 이 때, 스케일 축을 따라서도 코너성이 극대인 점을 찾으면 Scale-invariant한 Feature이다.

SIFT 방법에서는 각 Scale에서 코너성을 측정하기 위하여 Harris corner 방법과는 달리 Laplacian 함수값을 사용하고, 각 Scale 축의 이미지 내에서 뿐만 아니라 Scale축으로도 극대인 점들을 특징점으로 선택한다.



Laplacian 함수값 *L*은 이미지의 밝기 변화에 대한 2차 미분값으로서, 이미지의 밝기 변화가 일정한 곳에서는 0에 가까운 값을, 영역의 경계와 같이 밝기 변화가 급격한 곳에서는 높은 값을 갖는다. 실제 SIFT에서는 속도 문제로 인하여 이 함수 값을 직접 계산하지는 않고, DOG를 이용하여 각 Scale별 함수값을 근사적으로 계산한다. DOG는 입력영상에 Gaussian 필터를 적용하여 블러링시킨 이미지들에서 인접 이미지들간의 차를 의미하고, Laplacian 함수값을 이용한 것과 거의 동일한 결과를 보여준다.



이렇게 얻어진 DOG 피라미드에서 극대 또는 극소점을 찾으면 특징점이된다.

* 1. SURF(Speedup Robust Features)

SUFT는 말그대로 빠르게 특징점을 효과적으로 찾는 방법이다. 이 방법은 SIFT못지않게 성능을 유지하면서 계산속도를 향상시키는 방법이다. Scale Space상에서 Laplacian이 극대인 점들을 특징점으로 검출하는데, 모든 Scale에 대해서 특징점을 찾는 SIFT와는 달리 단 3개의 Scale로만 구성된 이미지 피라미드에서 특징점을 찾는다는 점에서 차이점이 있다.

그 외에도 속도향상을 위하여 크게 Integral Image를 이용(4개의 point만 알면 그 영역의 면적을 간단하게 계산할 수 있다.), 간편화한 Detector와 Descriptor를 활용하여 차원수를 줄이고, Contrast를 이용하여 간단한 매칭을 하는 방법을 제안하였다.

* 1. RANSAC(RANdom Sample Consensus) 특징점 매칭

위에서 뽑은 특징점들을 이용하여 가장 많은 특징점이 매칭되는 이미지가 해당 이미지와 가장 유사하다고 할 수 있다. 이 때 특징점 매칭알고리즘으로 RANSAC 알고리즘이 있다.

먼저 트레이닝 이미지를 Template, 매칭 시킬 이미지를 Target 이미지라고 한다. 이 때, Template과 Target이미지에서 랜덤하게 k pair를 찾는다. 이후 Template이미지의 모든 특징점을 아핀 변환을 이용하여 Target이미지에 매칭된 k 특징점의 각도로 변환시킨다. 변환 시킨 후 Template의 특징점과 Target의 특징점의 차이를 이용하여 두 이미지가 유사한지 알 수 있다.

* 1. Brute-Force Matcher

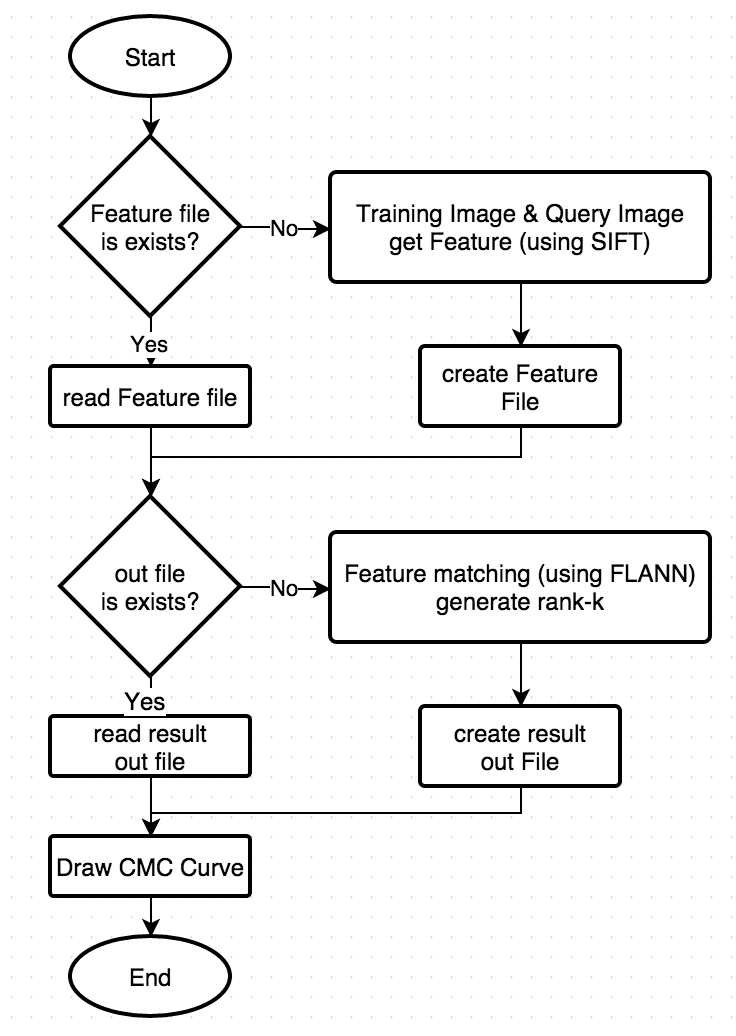
Opencv에는 Brute-force 방법으로 Feature 매칭해주는 라이브러리가 존재한다. Brute-force라는 이름에 맞게 트레이닝 이미지와 질의 이미지의 특징점이 주어졌을 때, 각각의 특징점 개수를 n,m 개라고 했을 경우 O(n\*m) 방법으로 모든 특징점들을 매칭해보는 방법이다. cv2.BFMatcher() 로 실행할 수 있다.

* 1. FLANN based Matcher

Opencv에는 FLANN(Fast Library for Approximate Nearest Neighbors) 라는 고차원 데이터 혹은 큰 사이즈의 데이터에서 빠르게 NN을 찾는 알고리즘 라이브러리를 제공한다. 이는 Brute-Force 방법보다 빠르고, 같은 결과를 보여준다. cv2.FlannBasedMatcher() 로 실행할 수 있고, 이번 프로젝트에서는 이 라이브러리를 이용하여 KDTree indexing 을 이용하였다.

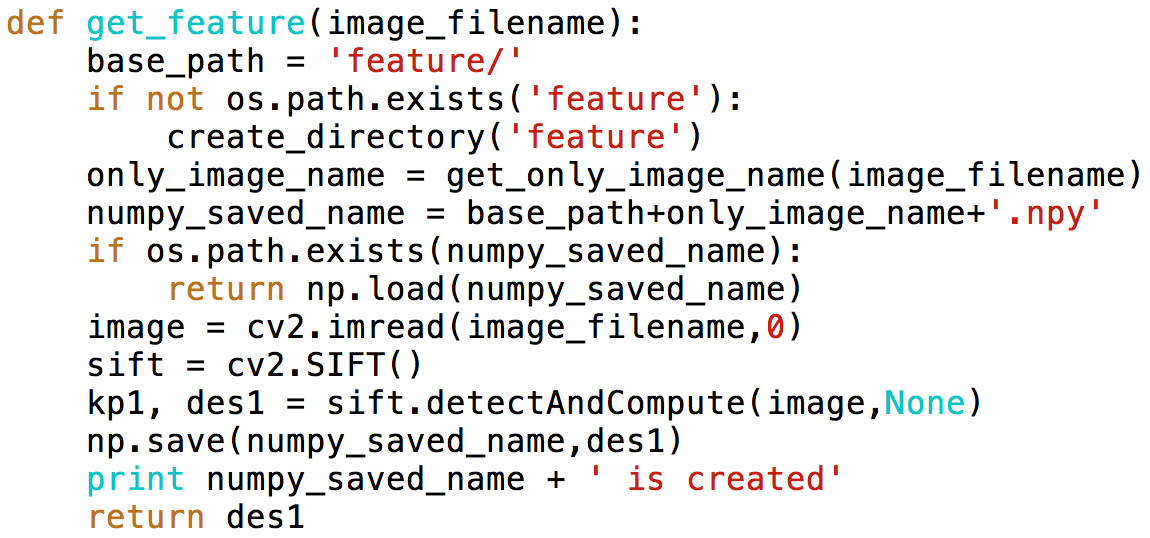
1. 개발 과정
2. 프로그램 순서도

이미지 매칭 프로그램의 전체적인 순서는 다음과 같다.



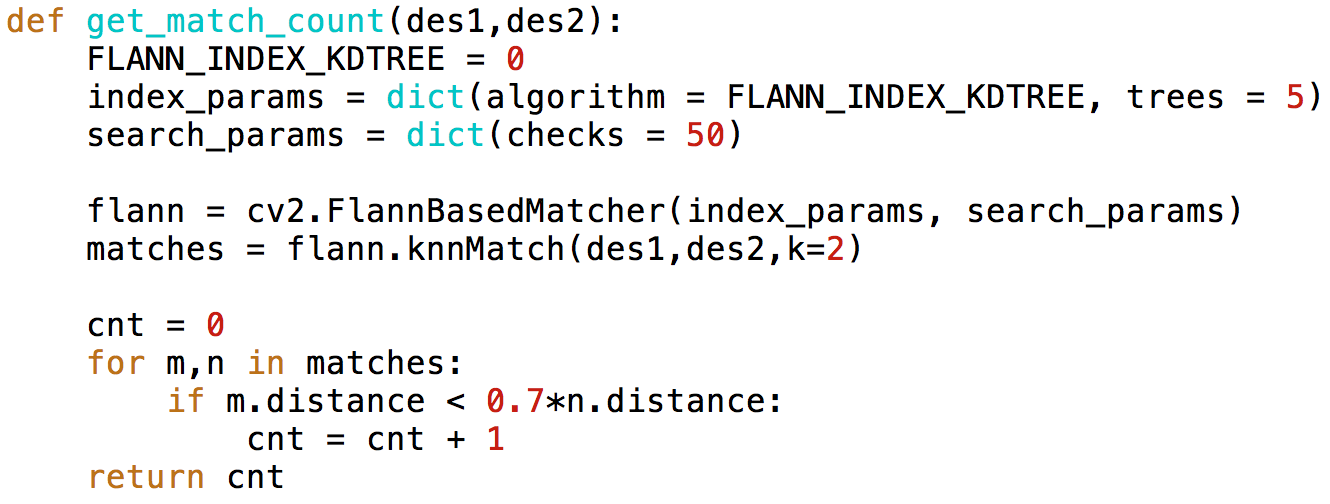
총 400개의 이미지 Feature를 뽑아내고, 각각의 Query 이미지마다 fa 이미지와 비교를 해야되므로 O(Feature\_time+100\*100\*matching\_time)이 걸리게 된다. 따라서 이 프로그램을 실행했을 때, 한번만 계산하기 위하여 Feature 뽑은 결과와 Matching 결과의 rank-k 값을 파일에 저장하도록 하였다.

1. Get\_feature



get\_feature 함수는 SIFT 알고리즘을 이용하여 이미지의 특징점을 찾아서 반환해주는 함수이다. 만약 예전에 특징점을 찾았다면(프로그램을 돌려서 특징점을 저장해 두었다면) numpy library를 이용하여 특징점을 불러온다.

1. Get\_match\_count

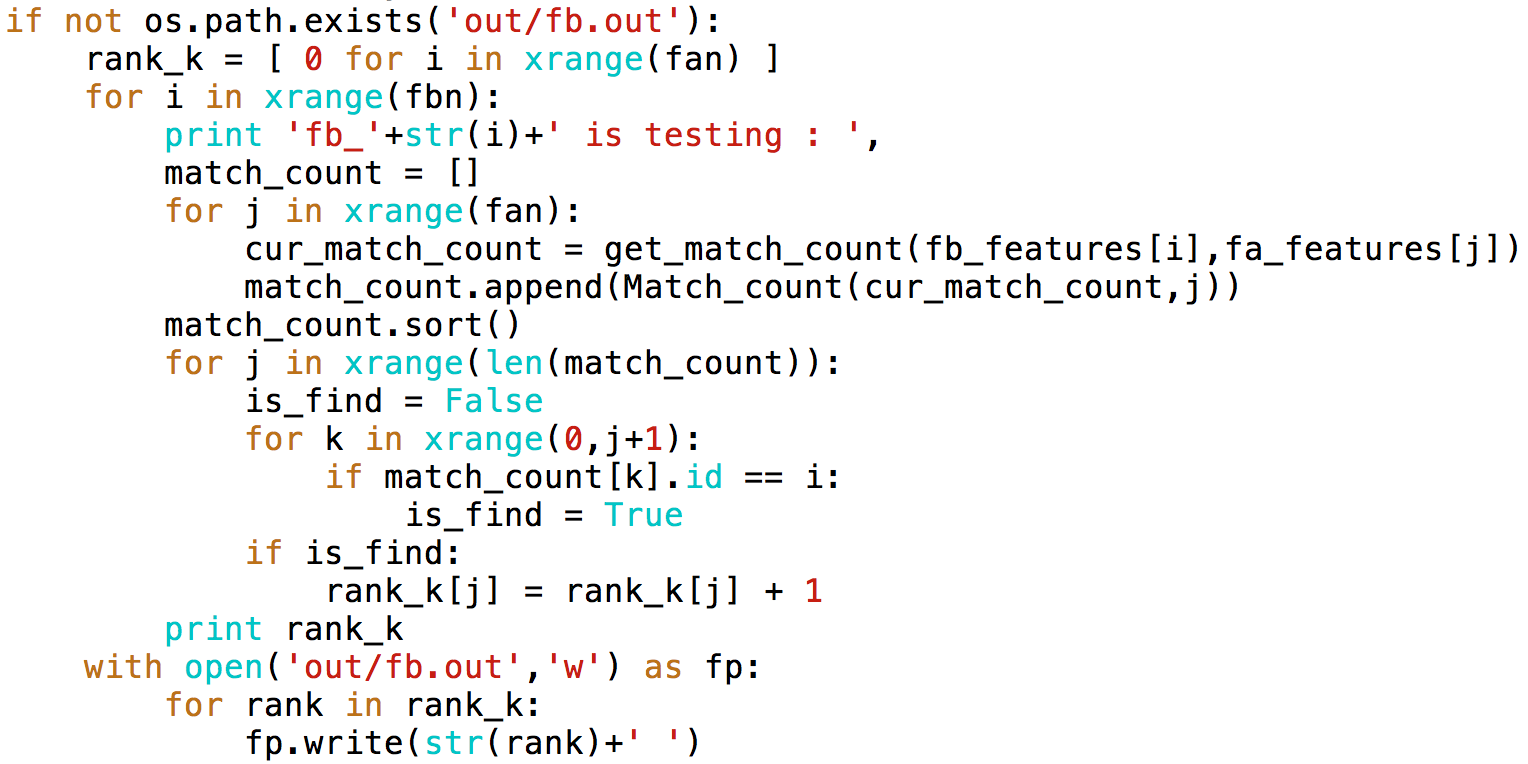


get\_match\_count 함수는 argument로 특징점 리스트 두 개를 받아서 특징점끼리 매칭한 결과를 반환해준다. Brute-force 방법인 BFMatcher() 대신 FlannBasedMatcher()를 사용하였다. 아래쪽의 성공적인 매칭 기준은 다음 식을 이용하여 계산하였다.

Matching points whose ratio

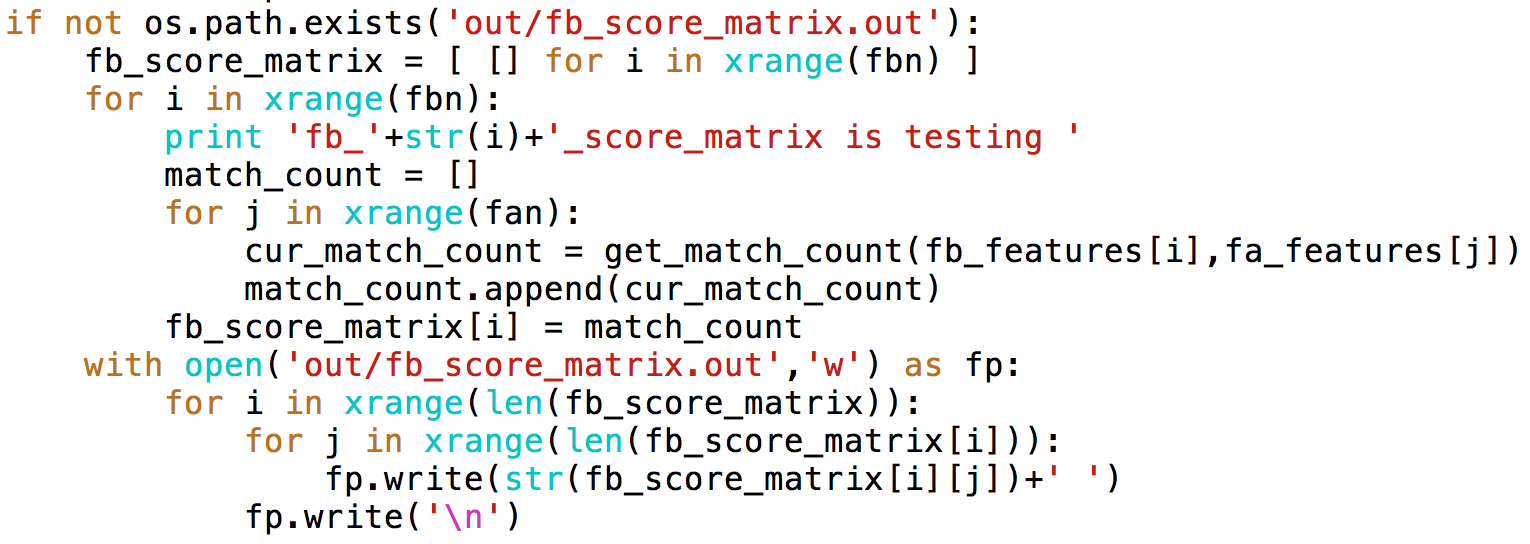
= (best\_match\_cost / second\_best\_match\_cost) < 0.7

1. Rank-k



각각의 Query 이미지 별로 Training 이미지와 매칭 카운트를 세어 매칭 카운트가 많을순서로 정렬하였다. 정렬한 k만큼의 이미지에서 실제 이미지와 매칭되는 query 이미지의 개수를 rank-k로 저장하였다. 위 코드는 fb에 대해서 이미지 매칭하는 부분이고, 이와 유사하게 qr과 ql에 대해서도 수행한다.

1. Score\_matrix

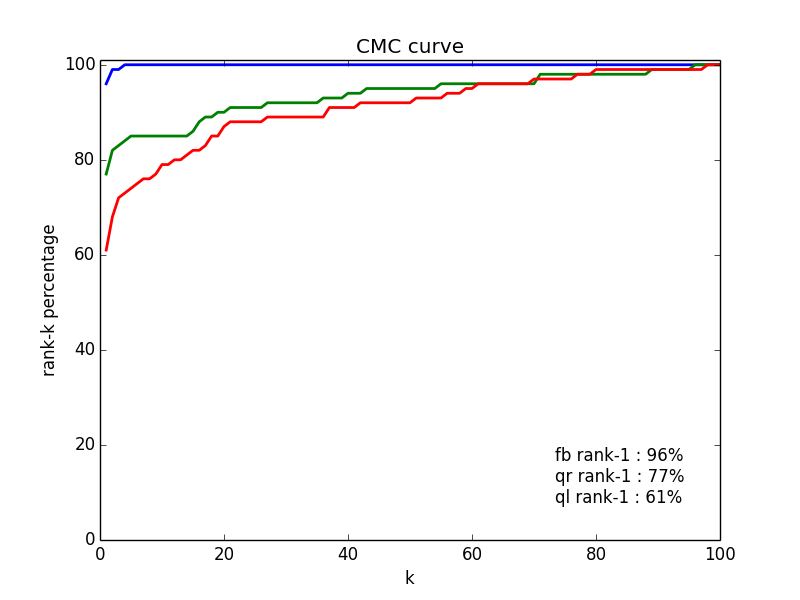


Score matrix는 Query이미지와 Training 이미지 사이의 매칭 카운트를 2차원 테이블 형태로 저장한다. 가장 큰 매칭 카운트가 두 이미지 간의 유사도가 가장 큰 것이다.

1. **실험 결과 및 개선점**
2. Score matrix

Score matrix는 100\*100의 2차원 테이블로써 이 문서에 담기에 크기 때문에 첨부된 파일 score\_matrix.xlsx 파일에 있다.

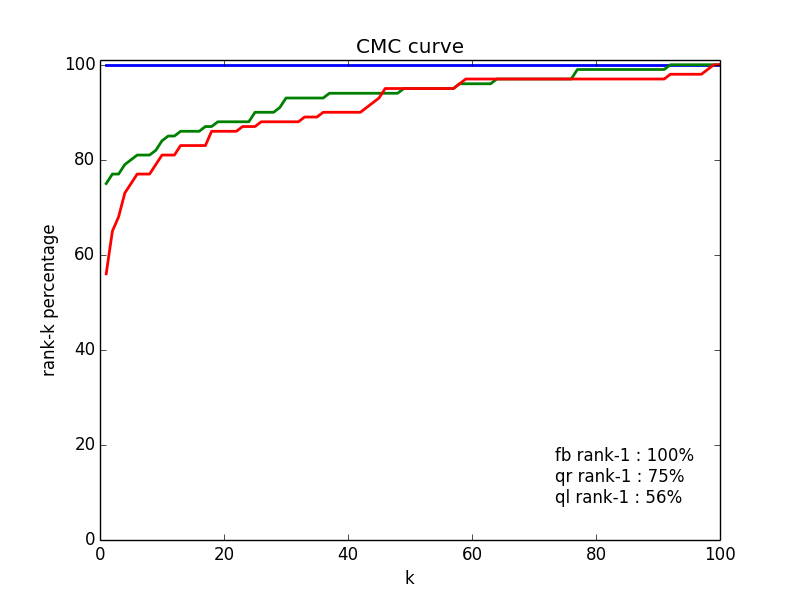
1. CMC Curve



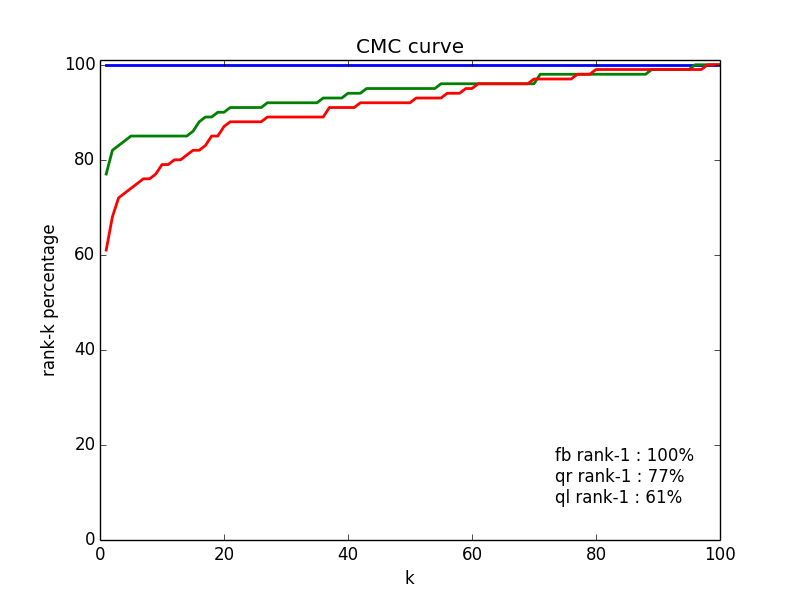
위 CMC curve는 증가되는 k에 따른 rank-k 값을 보여준다. 곡선이 완만 할수록 성공적인 이미지 매칭인데, rank-1이 fb는 96%, qr은 77%, ql은 61%로써 완만하다고 볼 수 있고, 세 query의 rank-1의 평균은 78%로 높다.

1. 개선점

이미지 매칭의 개선점으로는 더욱 효율적인 feature의 획득과 feature 간의 matching 정확도로 볼 수 있다. 하지만 feature 를 뽑아내는 알고리즘 중 SURF는 SIFT의 느린 속도를 빠르게 개선했을 뿐 더욱 정확한 feature라고 할 수 없다. 그것을 확인하기 위하여 SURF 알고리즘을 이용하여 rank-k에 대한CMC Curve를 그려보았다.



SURF 알고리즘을 이용해서 특징점을 뽑고, FLANN 을 이용하여 매칭했더니 rank-1로 fb는 100%, qr은 75%, ql은 56%의 높은 결과를 얻을 수 있었다. 하지만 평균을 내보면 77%로 SIFT의 결과보다 낮음을 알 수 있었다. 따라서 qr과 ql의 성능이 좋은 SIFT와 fb의 성능이 100%가 나오는 SURF의 결과를 종합하여 fb의 query를 처리할 때에는 SURF를 이용하고, 나머지 query에 대해서는 SIFT를 이용하여 아래와 같은 결과를 얻었고, 평균 rank-1은 **79.33%**의 성능을 얻었다.



1. 결론

ql에 대한 성능은 높지 않지만, 최종적인 이미지 매칭 확률은 평균 79.33%를 얻었으므로 전체적으로 높은 성능을 얻었다.

이미지에 대한 프로그램은 만들어보지 않았었는데, 이번 프로젝트를 통하여 이미지 프로세싱 알고리즘에 대하여 공부하게 되어서 재미있었고 직접 구현을 통하여 이미지가 매칭되는 것을 보면서 영상처리에 더욱 흥미를 갖게 되었다.

1. 참고문헌

1. C. Harris and M. Stephens, "A combined corner and edge detector", Alvey Vision Conference, 1988

2. Lowe, D.G., "Distinctive image features from scale-invariant keypoints", IJCV 2004.

3. Bay, H., Tuytelaars, T., and Van Gool, L., "Surf: Speeded up robust features," in ECCV 2006

4. https://en.wikipedia.org/wiki/RANSAC

5. http://docs.opencv.org/