HW 04 – REPORT

소속 : 정보컴퓨터공학부

학번 : 201824523

이름 : 안혜준

1. 서론

RANSAC 알고리즘을 구현해보고, RANSAC을 이용해 파노라마 이미지를 생성한다. 다른 descriptors와 구별되는 특징을 가지는 이미지의 descriptors가 있을 때, descriptors 정보를 바탕으로 다른 이미지와 매칭을 하여 여러 이미지에서 동일 한 부분을 찾아낼 수 있다. 이미지들에서 동일한 부분이 일치되도록 이미지를 비틀어(warping) 두 이미지를 이어 붙임으로써 파노라마 이미지를 만들 수 있다.

RANSAC은 RANdom SAmple Comsesus의 약자로, 여러 개에서 일부분의 샘플을 골라 하나의 가정을 만들고, 가정이 다른 데이터에도 잘 맞아 떨어지면 채택한다. 이미지 매칭을 위해 RANSAC은 일치한다고 판단되는 descriptors들에 대해 적용할 것이다. 틀린 매칭들은 전부 제 각각으로 표현되지만, 옳은 매칭들은 같이 움직인다.

파노라마 이미지는 한 이미지에서 다른 이미지 공간으로 projection으로 만들 수 있다. 이동, 회전, 크기 뿐만 아니라 비틀림까지 나타내기 위해서 homography를 사용한다. Homography는 매칭되는 점 4개를 통해 구할 수 있고, 가장 잘 맞는 homography를 구하기 위해 RANSAC 기법을 사용할 것이다.

실습은 두 파트로 나눠 진행된다. Descriptors의 방향과 크기를 바탕으로 RANSAC을 통해 이미지 매칭을 수행한다. Homography를 통한 projection으로 RANSAC을 수행해 파노라마 이미지를 만든다.

매칭을 위한 이미지

여러 이미지를 매칭시켜 만든 파노라마 이미지

Scene 이미지에서 각 책들의 사진을 바탕으로 이미지 매칭을 수행하고 RANSAC을 적용했을 때 달라지는 점을 알아본다. 같은 대상을 찍은 여러 이미지를 공통되는 특징점을 찾아 하나로 합치는 파노라마를 수행해본다.

2. 본론

1. SIFT Keypoint Matching

1. Image descriptors Matching

각 이미지의 keypoints와 descriptors들은 이미지와 함께 개별 파일로 제공된다. Keypoints는 (row, column, scale, orientation)의 정보를 가지고 있고, descriptors는 각각 128차원의 SIFT descriptors이다.

두 이미지에서 가져온 이미지를 바탕으로 가능한 모든 descriptors 쌍에 대해 매칭이 되는지 확인한다. 매칭 확인은 두 descriptor를 내적한 후 arccos을 취하여 얼마나 두 descriptor가 근접한지 계산한다. 0에 가까울수록 일치성이 높다.

하지만 그 descriptor가 이미지에 비슷한 것이 여러 개 있을 수 있으므로 두 번째 best match도 계산하여 두 번째 best match를 나누어 값을 계산하게 된다. Descriptor가 유일하다면 두번째 best는 값이 커져 최종 값을 더 작게 만들 것이고, 비슷한 descriptor가 존재한다면 최종 값에 0에 가까운 값을 나누게 되어서 최종 값을 증가시켜 매칭을 무시할 수 있게 해준다. 최종 값이 threshold 이하이면 매칭으로 판단한다.

앞서의 scene 이미지에 book과 box 이미지를 매칭 시킨다. RANSAC을 거치지 않아 잘못된 매칭이 존재하는 것을 확인할 수 있다.

Scene-book 매칭

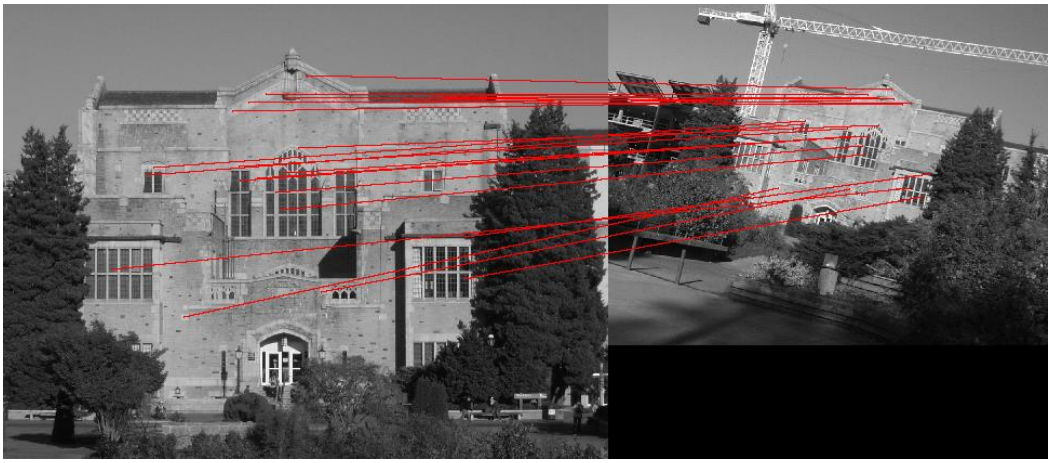
Scene-box 매칭

2. Image descriptors RANSAC Matching

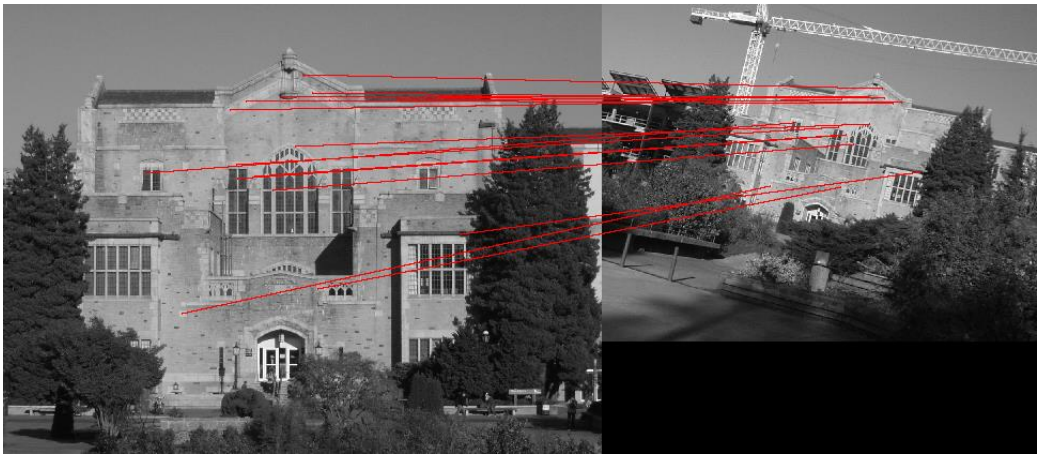
RANSAC 매칭을 추가한다. 여기서는 homography를 사용하지 않고 간단하게 방향과 크기를 통하여 match를 판별한다.

총 10번의 random sample을 수행한다. 하나의 matched pair를 선택하여 다른 matched pair와 방향과 스케일이 얼마나 다른지 계산한다. 이때, 다른 matched pairs가 인자로 받은 orient_agreement와 scale_agreement 범위 내에 존재하면 inlier로 판단한다.

이렇게 계산한 inlier의 수가 가장 큰 sample의 경우를 채택하고, 채택한 sample에서의 inliers들을 최종 match된 pair로 선정한다.



RANSAC을 수행하지 않은 매칭



RANSAC을 수행한 매칭

RANSAC을 수행하지 않은 경우 왼쪽 이미지의 왼쪽 아래 창문이 이상한 부분과 매칭되었지만 RANSAC을 수행하자 잘못된 매칭이 사라졌다.

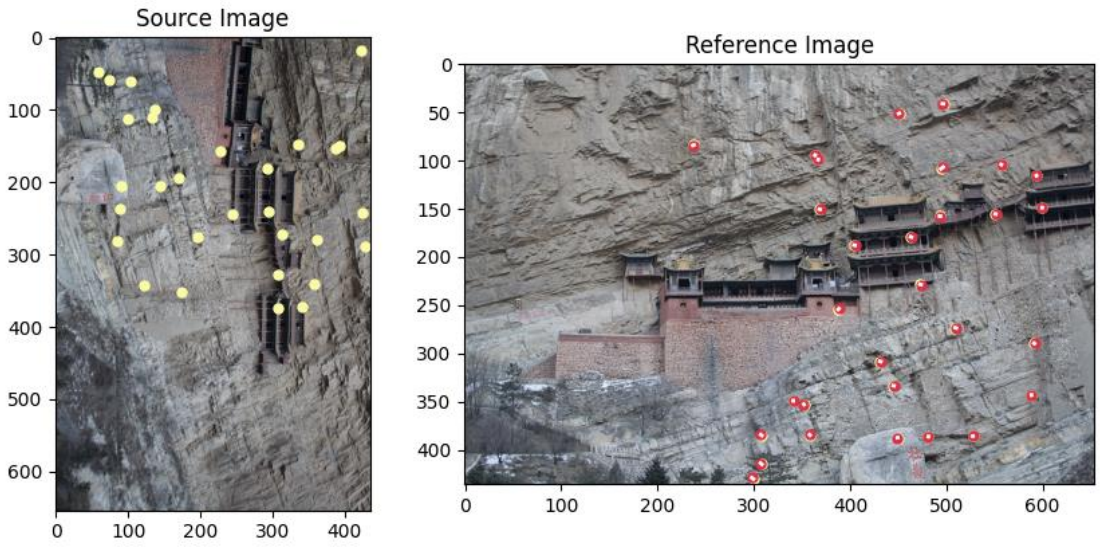
이때의 파라미터 값은 ratio_threshold=0.6, orient_agreement=5, scale_agreement=0.1으로 의미 있는 제거를 위해서 scale과 orient 차이 용납 범위를 최대한 낮춰야 했다. 더 낮추면 옳은 매칭조차 사라지기 때문에 이 이미지에서 실험적으로 적절한 값을 골랐다.

2. Panorama stitching

1. Keypoint Projections

이제 homography를 이용할 차례이다. 이미지 projection은 homography를 사용하여 이루어질 수 있다. projection의 테스트를 위해 Hanging 이미지의 제공된 테스트용 데이터(각각의 xy좌표, homography 행렬)를 사용한다.

Projection 시킬 xy좌표들을 homogeneous 좌표계로 만든다. 모든 좌표들에 대해 homography matrix를 곱한 뒤 w 좌표가 1이 되도록 정규화 시킨다. 이를 제공된 매칭 좌표 위에 뿌려보면 근사하게 일치함을 알 수 있다.



왼쪽 Source Image의 노란 점을 Homography를 적용하여 우측에 뿌렸을 때, 빨간 점과 겹치는 것을 알 수 있다.

2. RANSAC Homography

마지막으로 Homography를 RANSAC을 통해 직접 구해본다. Homography는 3x3 행렬로 homogeneous 좌표계의 2차원 좌표에 곱했을 때, projection된 homogeneous 좌표계 상의 좌표를 구할 수 있다. Homography는 h22가 1로 정규화되어야하므로 결국 자유도 8이라고 할 수 있다. 따라서 매칭되는 점 4개를 사용하여 x, y에 따라 2개씩 총 8개의 등식을 얻을 수 있고, 이를 통해 homography를 계산할 수 있다.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\underset{2n \times 9}{\mathbf{A}} \qquad \underset{9}{\mathbf{h}} \qquad \underset{2n}{\mathbf{0}}$$

Ah가 0에 근접하도록하는 h를 계산해야한다. 여기서 h는 A^T A의 고유 값 중 최소인 고유 값의 상응하는 고유벡터로 구할 수 있다.

```
eig_val, eig_vec = np.linalg.eig(A.T.dot(A))
min_index = eig_val.argmin()
homography = np.array(eig_vec[:, min_index]).reshape((3, 3))
```

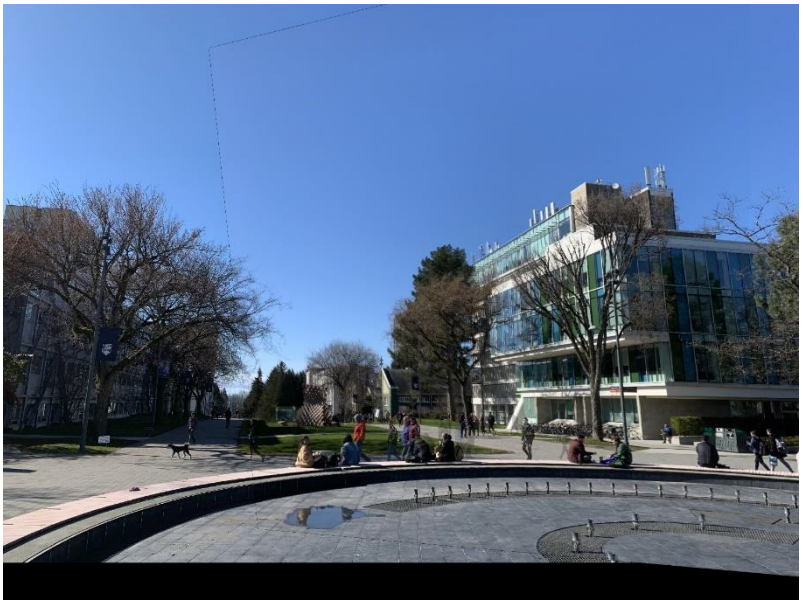
A는 랜덤하게 샘플된 4개의 매칭 pair로 만들었다. Homography를 만든 후 테스트하며 만든 KeypointProjection 함수를 사용하여 homography를 적용한 projection 좌표를 구한다.

Projection 좌표들과 매칭된 좌표들의 유클리드 거리를 계산하고 parameter로 입력받은 tol 이내인 경우를 inlier로 판단한다. Inlier의 개수를 세고, 전체 과정을 여러 번 반복하여 가장 inlier가 많은 경우의 homography를 채택한다.

도출된 homography를 사용하여 이미지들을 하나로 모아 파노라마 이미지를 만들 수 있다. 여기서는 겹치는 부위는 각각 0.5의 투명도를 가지게 하여 합쳤다.

이제 파노라마 결과물 중 3개의 예시를 보이겠다.

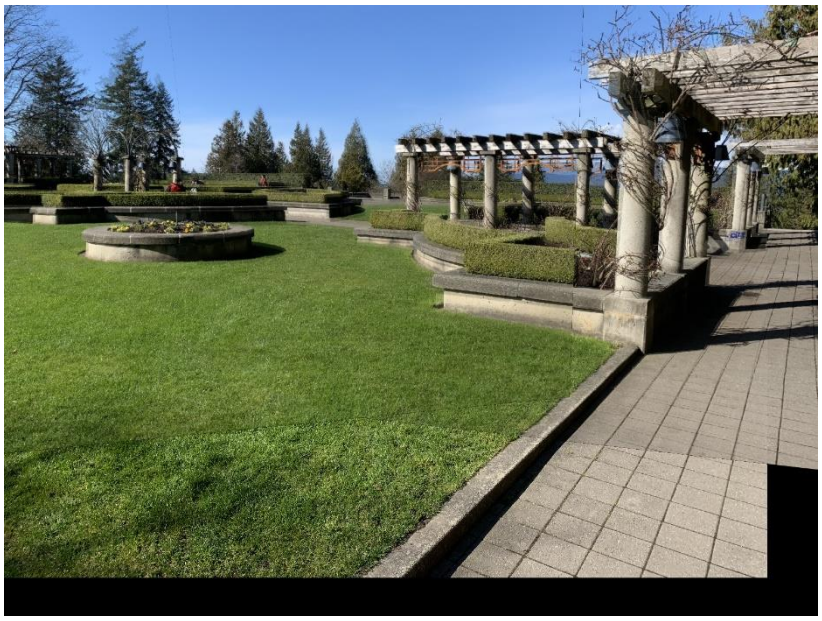
Foundation의 2개 이미지



Foundation panorama 이미지



Garden의 3개 이미지



Garden panorama 이미지

Irving 3개 이미지

Irving panorama 이미지

Foundation의 경우, tol=5, ratio_thres=0.6

Garden의 경우, tol=3, ratio_thres=0.7

Irving의 경우, tol=3, ratio_thres=0.7을 사용했다.

Garden을 foundation과 동일한 파라미터로 하자 끊어짐이 심하였다. Tol을 낮춰보아도 끊김이 심하다가 오히려 ratio threshold를 약간 증가시키자 매칭이 더 잘 되었다. Foundation의 경우 ratio threshold를 증가시켜도 좋은 결과를 얻지 못했다. 각 이미지에 따라 최적의 값이 전부 다름을 알 수 있다.

3. 결론 및 토의

SIFT descriptor를 이용하여 RANSAC으로 이미지 매칭을 수행하였다. 이미지 프로젝션을 위해 homography를 RANSAC을 사용하여 직접 계산해보았고, homography로 이미지들을 하나의 이미지 위에 프로젝션 시켜 파노라마 이미지를 만들었다.

매칭과 RANSAC에 각각 threshold 파라미터가 필요했다. 파라미터들은 최적의 값이 입력 이미지들에 따라 달랐고, 실습에서는 실험적으로 여러 번의 수행으로 가장 좋은 결과물의 파라미터를 도출해냈다. 파라미터 계산까지 자동화를 하기 위해 입력 이미지를 분석하여 최적의 파라미터를 자동으로 찾아내는 수식이 필요할 것이다.

이 실습에서는 homography를 통해 projection 후, 겹치는 부분을 합치는 방식을 사용했다. 이미지마다 빛이 다르기 때문에 겹친 부분의 선이 뚜렷하게 나타났다. 이를 해결하기 위해서 더 나은 blending 방법을 도입할 수 있다. 예를 들어, Laplacian pyramid를 활용하여 해결할 수 있다.

결과 이미지가 다른 이미지를 비틀어 올린 이미지여서 직사각형 캔버스에 빈공간이 존재하여 모양이 지저분하다. 결과 이미지를 분석하여 빈공간이 없는 최대의 직사각형 크기를 계산하여 출력할 수 있을 것이다.