HW 04 – REPORT

소속 : 정보컴퓨터공학부

학번 : 201925111

이름 : 김건호

1. 서론

1. 실습 목표 – Panorama

1) RANSAC(Random Sample Consensus)을 통해 잘못된 matching을 제거할 수 있다.

2) RANSAC(Random Sample Consensus)을 사용하여 homography matrix를 만들 수 있다.

3) homography matrix를 통해, keypoint를 projection 할 수 있다.

2. 이론적 배경

지난 실습에선, image의 keypoint를 detection하는 방법에 대해 학습했습니다. 이제 detection한 image의 keypoint의 특징을 나타내는 description vector를 SIFT(Scale-Invariant-Feature Transform)을 통해, 구할 수 있습니다. SIFT의 장점은 빠르고, 최대 60°까지 회전에는 관계없는 description vector를 추출할 수 있습니다. 아래는 SIFT의 과정입니다.

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

이후, homography matrix를 계산해야 합니다. 이는 cv2의 findHomography method를 통해 구하거나, 직접 계산할 수 있습니다. homography matrix를 직접 계산하기 위해 먼저 point를 homogenous coordinate(동차좌표계, 2차원 좌표를 z=1을 추가함)로 변환시킵니다.

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

이후, homography matrix 내부 value들을 미지수로 설정합니다. matrix equation을 전개하면, 아래와 같은 식이 나옵니다.

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

이는 아래와 같은 matrix로 표현할 수 있습니다.

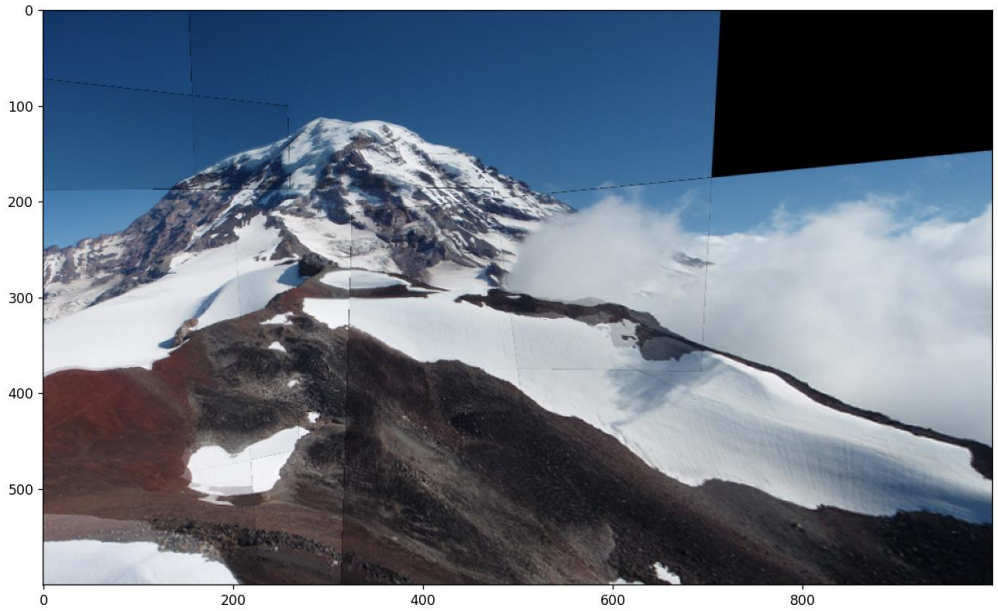
$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix} \begin{bmatrix}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

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

이는 결국, Ax=0의 non-trival least square solution을 구하는 것과 동일합니다. h는 $A^T * A$ 의 eigenvector과 eigenvalue를 통해 구할 수 있습니다. 최소 eigenvalue에 mapping되는 eigenvector가 homography matrix가 됩니다. 이는 numpy library의 linalg(linear algebra).eig를 통해 구할 수 있습니다.

이를 모두 계산하는 것은 많은 시간이 걸리기에, RANSAC(Random Sample Consensus)을 사용하여 homography matrix를 계산합니다. matrix의 미지수가 8개가 있으므로, 최소 4개의 x,y point pair가 필요합니다. n개의 point를 선택한 후, homography matrix를 계산합니다. 계산된 matrix를 사용해서, 제일 inlier의 수가 많은 적합한 homography matrix를 선택합니다.

아래는 RANSAC을 통해 homography matrix를 계산하고 만든 panorama image의 예시입니다.



2. 본론

Part 1: SIFT Keypoint Matching

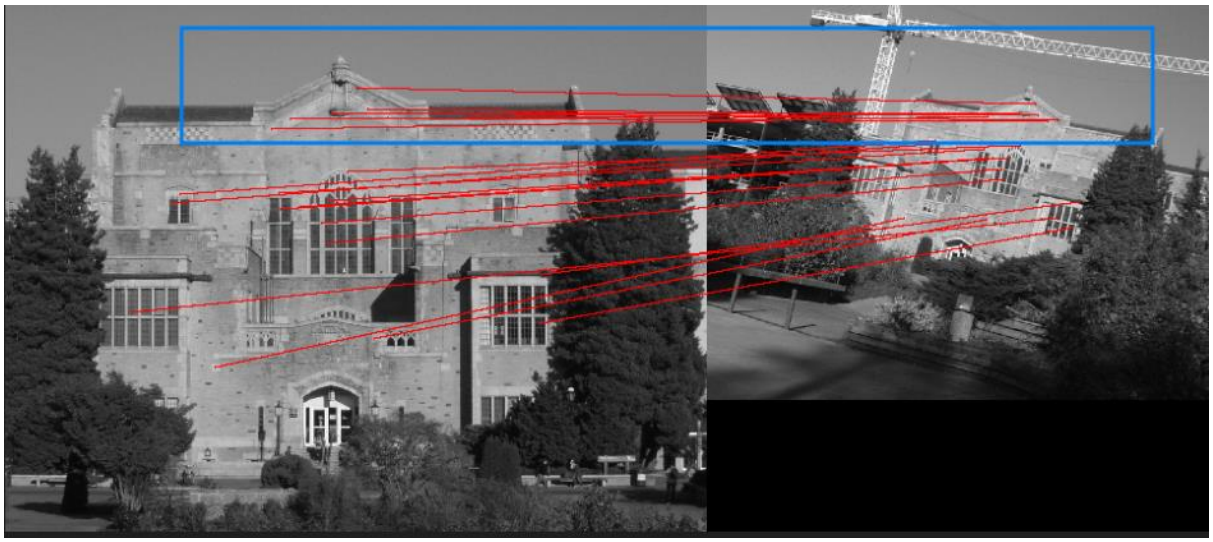
1. FindBestMatches

image들의 keypoint를 descriptor를 매개변수로 갖는 함수입니다. image1의 descriptor1를 image2의 모든 descriptor2들과 비교해서, 제일 matching이 잘 되는 point pair를 return하는 함수입니다. 제일 matching이 잘 되는지는, best matching angle과 second best matching의 비율인 ratio distance를 통해 판단합니다. 만약 ratio distance가 hyperparameter threshold보다 크다면, 각 point pair는 matching이 잘 되지 않는다고 판단하고 discard합니다. 아래는 scene-book을 threshold=0.6을 통해 matching한 결과입니다.

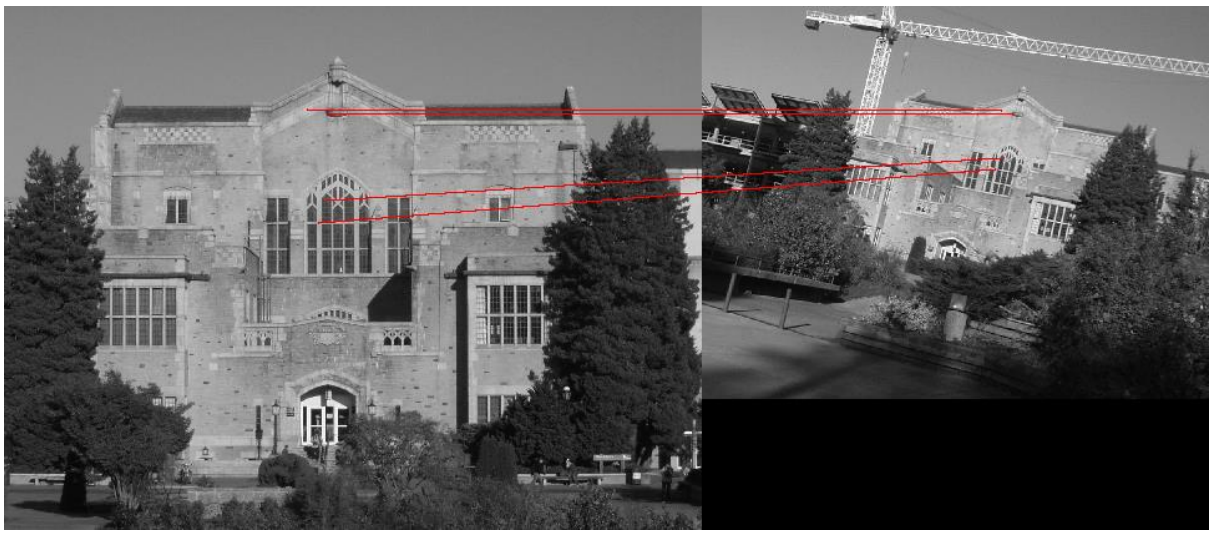
아래는 scene-box를 threshold=0.6으로 matching 한 결과입니다.

2. RANSACFilter

위에서 구현한 FindBestMatches는 각 point들의 descriptor를 모두 비교해, matching되는 point pair를 return합니다. 만약 잘못된 matching point matching이 존재한다면, 이는 잘못된 결과를 낳을 것입니다. 이를 방지하기 위해, RANSAC(Random Sample Consensus)을 사용합니다. matching된 몇 개의 pair를 random하게 추출합니다. 실습에선 10개의 random한 pair를 추출합니다. 이후 random하게 추출한 matching pair의 keypoint에 대해, 모든 matching pair의 keypoint의 orientation과 scale을 비교하며, threshold를 벗어나지 않는다면, 이는 consensus set에 추가됩니다. 이렇게 얻은 consensus set의 크기가 inlier의 수입니다. 이를 n번 반복하여, 제일 inlier의 수가 많은 consensus set을 return합니다. 아래는 ratio_thres=0.6, orient_agreement=30, scale_agreement=0.5로 library와 library2를 matching한 결과입니다. 파란 부분에서, 왼쪽 부분이 오른쪽으로 가는 False matching을 볼 수 있었습니다.



아래는 false matching을 감소시키기 위해, ratio_thres=0.45, orient_agreement=30, scale_agreement=0.4로 hyperparameter를 수정한 결과입니다. outlier의 수가 0인 결과를 확인할 수 있습니다.

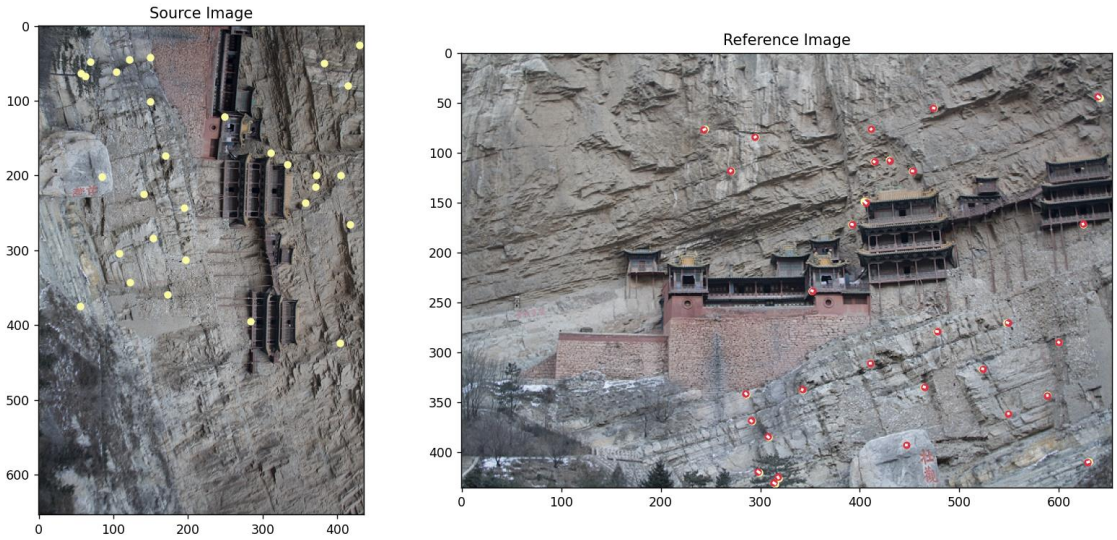


Part 2: Panorama

1. Keypoint Projections

homography matrix (3x3)과 point의 x,y 좌표가 주어진다면 이를 projection하는 함수입니다. 먼저 x,y 좌표를 homogeneous coordinate(동차좌표계)로 변환시킵니다. 이후, homography matrix를 통해 homography 변환을 합니다. 이후 변환이 된 결과를 return하는 함수입니다.

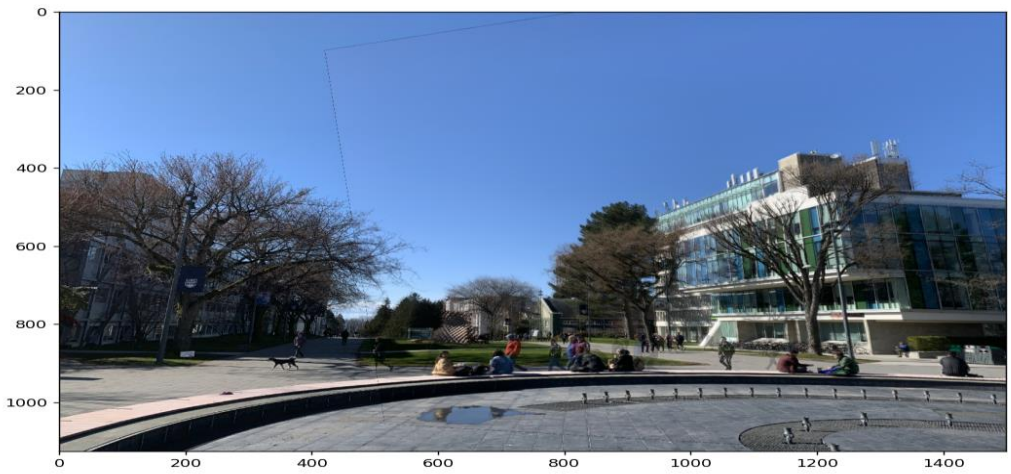
아래는 Hanging1 image와 Hanging2 image를 주언 homography matrix를 사용해서 projection한 결과입니다.



2. RANSAC Homograpy

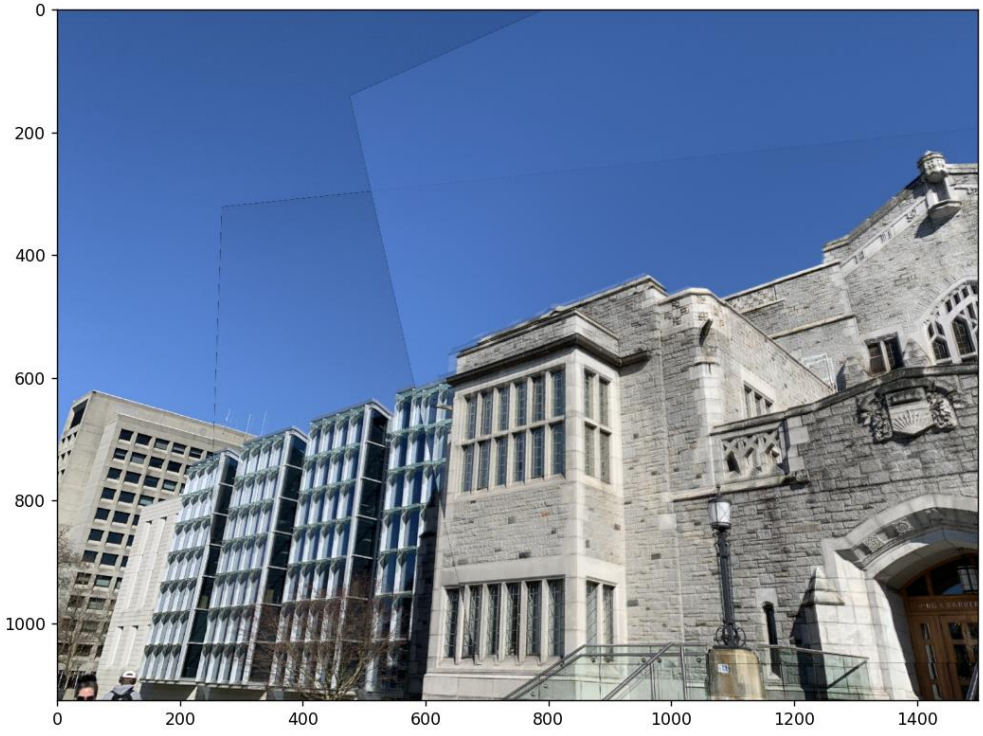
RANSACHomography function은, 몇개의 점을 random하게 고른 후, 이를 통해 homography matrix를 계산합니다. 실습에선 4개의 random한 점을 사용합니다. 계산된 homography matrix를 통해 projection을 진행하고, projection된 점들과 reference image의 각 점들의 거리를 계산합니다. 점들의 거리가 tol을 넘지 않는다면, inlier로 판단합니다. 이를 n번 반복하여, 제일 inlier의 수가 많은 homography matrix를 return합니다.

'fountain4 - fountain0'를 사용해서 panorama image를 만들기 위해, canvas_height와 canvas_width를 각각 1125,1500로 조정했습니다. 아래는 'fountain4 - fountain0'를 사용해서 만든 panorama image입니다.



'garden4 - garden3 – garden0'를 사용해서 panorama image를 만들기 위해, canvas_height와 canvas_width를 각각 1125, 1500로 조정했습니다. 아래는 'garden4 - garden3 – garden0'를 사용해서 만든 panorama image입니다.

'irving_out3 - irving_out6 – irving_out5'를 사용해서 panorama image를 만들기 위해, canvas_height와 canvas_width를 각각 1125, 1500로 조정했습니다. 아래는 'irving_out3 - irving_out6 – irving_out5'를 사용해서 만든 panorama image입니다.


0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400

3. 결론

이번 실습에서, RANSAC을 통해 잘못된 keypoint matching을 pruning을 할 수 있었을 뿐 아니라, RANSAC을 통해, 효율적으로 homography matrix를 만들 수도 있었습니다. 이를 통해 얻은 matrix로 keypoint projection도 성공적으로 수행할 수 있었습니다.

panorama image를 만들기 위해서는 homography matrix를 찾아야 합니다. 실습에선, eigen decompostion을 통해 homography matrix를 계산하였지만, SVD(Singular Value Decomposition, 특잇값 분해)를 통해서도 homography를 계산할 수 있습니다. singular value가 최소가 되는 right singular vector를 사용하면 Ah=0에 가장 근사한 h matrix를 찾을 수 있기 때문입니다. 이는 python의 numpy.linlag.svd를 통해서도 구할 수 있습니다.

또한, fountain panorama image를 만들 때, ratio_thres를 0.4로 설정하면, 하단 부분이 matching이 잘 되지 않는 모습을 볼 수 있습니다. 여러 최적화 알고리즘을 통해, 적절한 ratio_thres를 찾는 것 또한 중요하다고 생각합니다.