HW 04 – REPORT

소속 : 정보컴퓨터공학부

학번 : 201924548

이름 : 이풍헌

1. 서론

실습 목표 및 이론적 배경 기술 (1~2페이지)

이번 실습에서는 파노라마 이미지를 생성하는 것을 목표로 한다. 과정은 다음과 같다.

1. 사진에서 feature descriptor를 추출한다.

   Feature descriptor는 사진이 변하더라도 변하지 않고 각 포인트에 대해 unique하다.

   feature들간의 distance는 ||f1-f2||로 구할 수 있지만 비슷한 점에 의해 outlier가 측정될 수 있으므로 ratio distance ||f1-f2|| / ||f1-f2'||로 계산한다. 모호한 매치에 대해 큰 값을 반환한다.

2. Feature descriptor를 이미지에 매칭시켜 transform matrix를 얻어 alignment를 해야 한다. 이번 실습에서는 homography에 대한 transform matrix를 얻어야 하므로 D.O.F=8, 적어도 4개의 매치를 필요로 한다. Ah=0에 대한 lsq solution은 [v, λ] = eig(A^T A)에 대해 가장 작은 λ에 대한 고유 벡터이다.

3. Ransac은 랜덤으로 뽑은 매치에 대해 inlier들을 구해 가장 많은 inlier를 가지는 model을 추적하는 방식이다.

   A. Ransac의 순서는 다음과 같다.

   B. 최소 샘플 사이즈 만큼의 샘플을 고른다.

   C. 고른 샘플들에 대한 모델을 만든다.

   D. 모델에 근사한 inlier의 숫자를 센다.

   E. N번 반복한다.

   F. 가장 많은 inlier를 가지는 model을 고른다.

   Ransac은 심플하고 실전에서 잘 동작한다는 장점을 가지고 있다.

4. Panorama는 homography matrix를 통해 여러 장의 사진을 하나의 사진으로 만드는 과정이다.

   A. 두 장의 사진에 대해 ransac을 이용해 homography를 계산한다.

   B. 두번째 사진을 transform해 첫번째 사진위에 overlap한다.

이를 통해 우리는 paranoma사진을 얻을 수 있다.

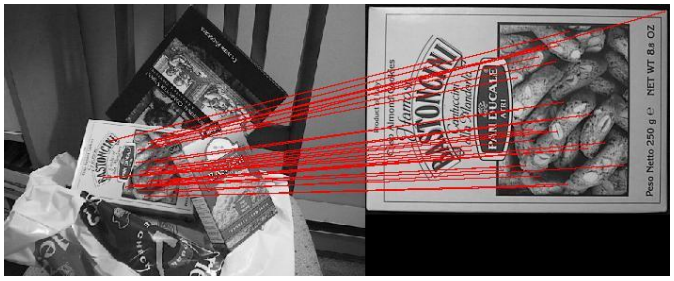
2. 본론

실습 내용 및 결과 기술 (2페이지 이상)

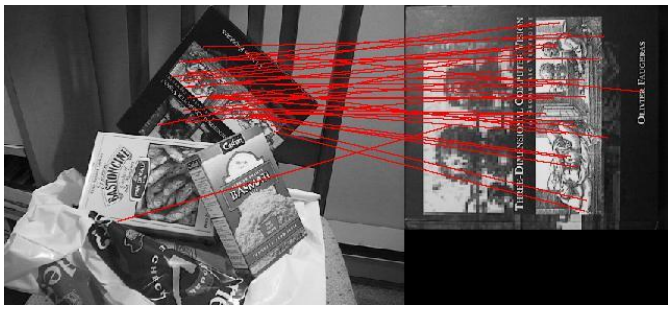
첫 번째 함수는 FindBestMatches로 1번 사진에 대한 desciptor1, 2번 사진에 대한 desciptor2, 임계선 threshold를 인수로 받아 ratio distance가 threshold를 넘는 match들을 반환한다.

```
## START
## the following is just a placeholder to show you the output format
y1 = descriptors1.shape[0]#1번 그림 descriptor 개수
y2 = descriptors2.shape[0]#2번 그림 descriptor 개수

temp = np.zeros(y2)#1번그림과 2번그림의 유사도 저장할 공간

matched_pairs = []#유사한 descriptor저장할 공간

for i in range(y1):
    for j in range(y2):
        temp[j] = math.acos(np.dot(descriptors1[i],descriptors2[j]))#1번 그림 descriptor마다 2번 그림의 모든 descriptor의 유사도 검사해 temp해 저장

    compare = sorted(range(len(temp)), key = lambda k : temp[k])#temp값을 기준으로 index정렬

    if(temp[compare[0]] / temp[compare[1]] < threshold):#가장 유사한 값을 두번째로 유사한 값으로 나눈 ratio distance사용
        matched_pairs.append([i,compare[0]])            #threshold이상이면 matched_pairs에 저장
## END
```

Descriptor1에 대해 descriptor2의 모든 원소와 distance를 구해 temp에 저장하고 sort를 통해 제일 작은 두 값의 distance를 통해 ratio distance를 구한다. 이 ratio distance가 threshold이하라면 match라고 판단해 matched_pairs에 추가한다.



Scene-box match이미지

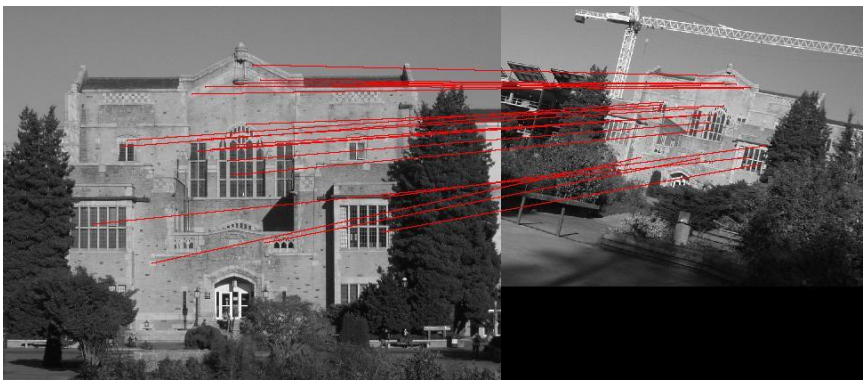


Scene-book match이미지

두번째 함수 RANSACFilter는 1에서 구한 match와 이미지의 keypoints(row,col,scale,orientation), 수용선 *_agreement를 받아 ransac을 수행한다.

```
## START
largest_set = [] #가장 많은 inlier이 들은 집합

#print(keypoints1, keypoints2)
#print(orient_agreement, scale_agreement)
orient_agreement = orient_agreement / 180 * math.pi
for i in range(10) : # 10번 반복
    rand = random.randrange(0, len(matched_pairs))
    choice = matched_pairs[rand] # matched_pairs 중 하나를 랜덤으로 선택

    orientation_i = (keypoints1[choice[0]][3] - keypoints2[choice[1]][3]) % (2 * math.pi) # 선택한 매치에 대해 orientation 계산
    scale_i = keypoints2[choice[1]][2] / keypoints1[choice[0]][2] # 선택한 매치에 대해 scale ratio 계산

    temp = []

    for j in range(len(matched_pairs)): # 다른 모든 match에 대해 계산
        if j != rand:
            # rand가 아닌 match에 대해 orientation 계산
            orientation_j = (keypoints1[matched_pairs[j][0]][3] - keypoints2[matched_pairs[j][1]][3]) % (2 * math.pi)

            # rand가 아닌 match에 대해 scale ratio 계산
            scale_j = keypoints2[matched_pairs[j][1]][2] / keypoints1[matched_pairs[j][0]][2]

            # orientation차이가 agreement보다 작은지 확인
            if (abs(orientation_i - orientation_j) < orient_agreement) or (abs(orientation_i - orientation_j) > (2 * math.pi) - orient_agreement):
                # scale비율이 +-agreement 이내인지 확인
                if( scale_i - scale_i * scale_agreement < scale_j < scale_i + scale_i * scale_agreement):
                    temp.append(j)# 둘 다 agreement하면 index를 temp에 저장

    if len(temp) > len(largest_set): # 가장 많은 match를 가진 temp를 largest_set에 저장
        largest_set = temp

for i in range(len(largest_set)):#index로 저장한 배열을 다시 descriptor로 바꿈
    largest_set[i] = (matched_pairs[largest_set[i]][0], matched_pairs[largest_set[i]][1])
    |
## END
```

랜덤으로 선택한 match i 에 대해 orientation_i와 scale_i를 구한다. 그리고 나머지 match j에 대해 orientation_j와 scale_j를 구해 orientation_i, scale_i와 비교해 agreement안에 들어가는지 비교한다. agreement안에 들어가면 inlier로 판단해 temp에 추가하고 가장 많은 inlier를 가진 largest_set을 반환한다.

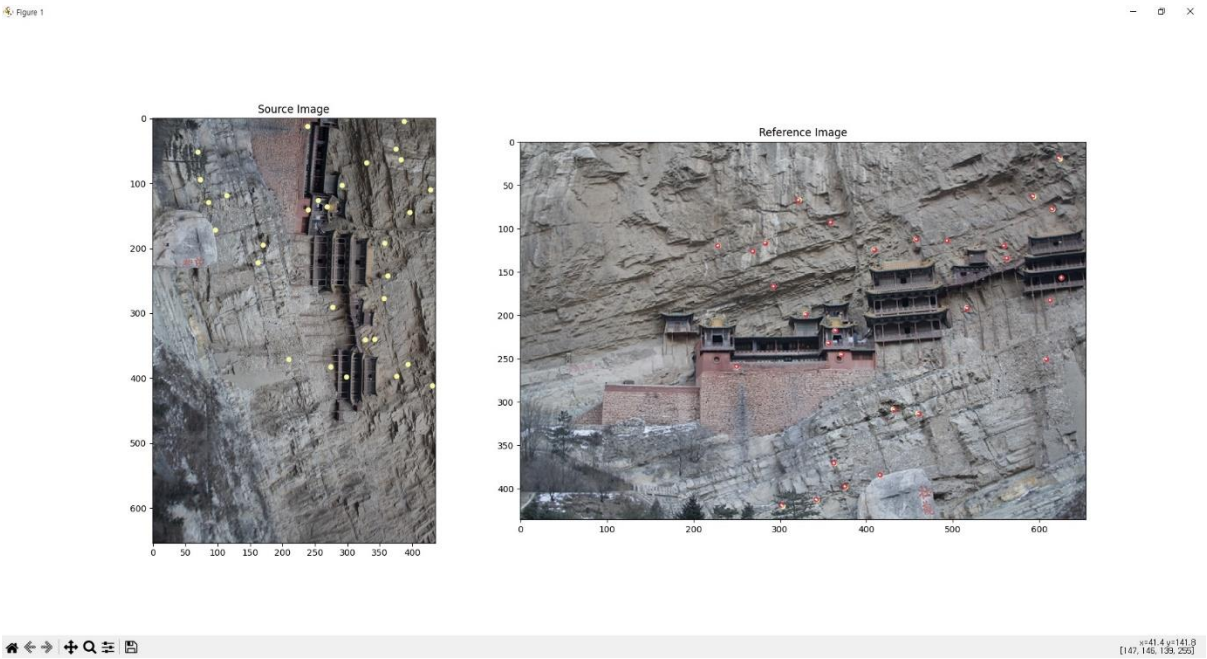


library-library2

세번째 함수는 source image를 homography matrix를 통해 reference image로 projection시키는 함수이다.

```
# START

hc_xys = np.pad(xy_points,pad_width=((0,0),(0,1)), mode= 'constant', constant_values= 1)
xys_p = h @ hc_xys.T #homography매트릭스와 연산
z_cor = np.where(xys_p[-1:] == 0 , 1e-10, xys_p[-1:]) #z좌표가 0이라면 매우 작은 수로 대치
hc_xys_p = xys_p/z_cor #z좌표로 나누어 평면으로 옮김
xys_p = hc_xys_p[:-1, :] #z좌표 같으므로 제거
xy_points_out = xys_p.T #transpose해 위치 표시


# END
```

동차 좌표계를 위해 xy_points에 1을 패딩으로 넣고 homography matrix h와 연산한다. 다시 z좌표에 대해 x, y좌표를 나누어 주면 projection된 좌표를 얻을 수 있다.

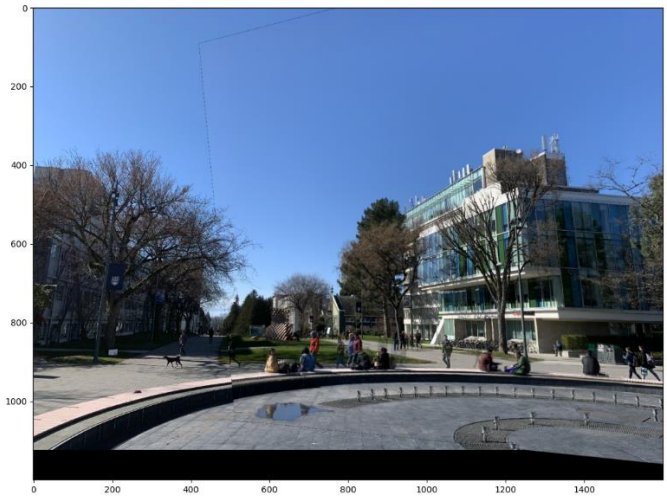


Hanging1-Hanging2

4번째 함수는 RANSAC Homography이다. match되는 xy좌표를 받아 ransac을 수행해 homography matrix를 반환하는 함수이다.

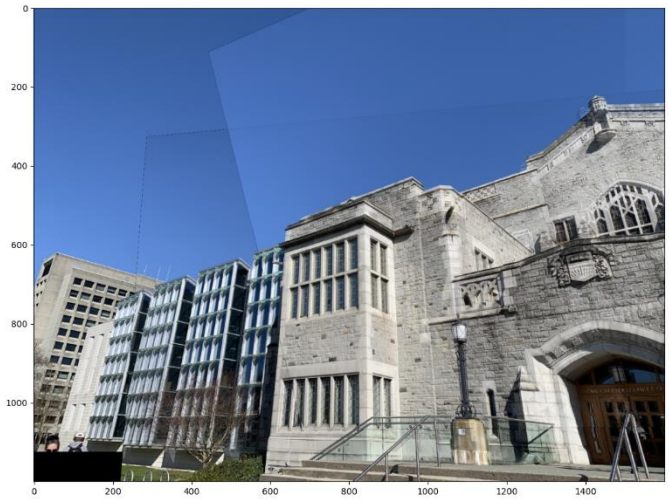
```
max_inliers = 0
h = None
N = xy_src.shape[0]
for _ in range(num_iter):

    sample = np.random.choice(N,4,replace = False)
    src_sample = xy_src[sample]
    ref_sample = xy_ref[sample]


    A=[]
    for i in range(4):
        x,y = src_sample[i]
        x_p, y_p = ref_sample[i]
        A.append([x, y, 1, 0, 0, 0, -x_p*x, -x_p*y, -x_p])
        A.append([0, 0, 0, x, y, 1, -y_p*x, -y_p*y, -y_p])
    A = np.array(A)

    #At=0
    eig_val, eig_vec = np.linalg.eig(A.T.dot(A))#A의 고유값,고유벡터
    min_index = eig_val.argmin() #최소 고유값 인덱스
    #최소값 가지는 고유벡터를 3*3모양으로 다시만들음
    homography = np.array(eig_vec[:, min_index]).reshape((3, 3))

    #src를 구한 homography를 통해 projection
    xy_out = KeypointProjection(xy_src, homography)

    # 유클리드 거리 계산
    dists = np.sqrt(np.sum((xy_out - xy_ref)**2, axis=1))

    # inlier개수최대로 가지는 h구하기
    inliers_num = np.count_nonzero(dists <= tol) #tol보다 작은 distance가지는 것만 inlier
    if inliers_num > max_inliers:
        h = homography
        max_inliers = inliers_num

# END
```

random으로 뽑은 4개의 match에 대해 A matrix를 구성한다. Lsq를 위해 A의 매트릭스의 가장 작은 고유 값을 가지는 고유벡터를 3*3으로 만들어 h매트릭스를 구한다. h매트릭스를 통해 src이미지를 ref이미지에 projection하고 유클리드 거리가 tol보다 작은 inlier를 가장 많이 가지는 h matrix를 반환한다.



fountain4-fountain0

garden0-garden3-garden4



irving_out3-irving_out6- irving_out5

3. 결론

토의 및 결론 (1페이지)

SIFT descriptor를 ransac을 이용하여 Projection에 필요한 homography를 직접 얻어 여러 이미지를 align시켜 paranoma를 만드는 실습을 하였다.

Ransac이 실전에서 잘 적용되는 것을 실습할 수 있었다. 하지만 많은 parameter를 가지기에 이미지마다 최적의 paranoma이미지를 얻기 위해 다양한 threshold를 조절해야 하는 필요를 느꼈다.