1. 서론

이번 과제에서는 두 이미지 사이의 epipolar line을 만들어 내는 법을 배운다.

epipolar line을 찾아내기 위해서는 아래와 같은 과정이 수행되어야 한다.

1. 두 이미지에서 feature를 찾는다.
2. 찾은 feature들을 서로 매칭한다.
3. 매칭한 feature쌍들을 이용해서 Fundamental Matrix를 구한다.
    a. Fundamental Matrix를 구할 때에는 Rank 축소가 필요하다.
4. Fundamental Matrix와 feature point를 이용해서 epipolar line을 그린다.

2. 본론

1. Fundamental Matrix Estimation

`.npy` 형식으로 제공된 이미지의 포인터들의 좌표를 이용하여 Fundamental Matrix, `F`을 찾는다.

첫 번째 이미지의 지점 `(u, v)`에 대응하는 두 번째의 이미지의 좌표를 `(u', v')`라고 할 경우 아래의 식을 통해서 F를 획득할 수 있다. 왼 쪽의 Matrix를 A라고 했을 때, `A.transpose * A`의 eigenvalue 중에서 가장 작은 값에 대응하는 eigenvector를 찾는 것이다.

과정을 수행하는 도중 마지막 entry가 1이 되게끔 일반화하는 작업을 거치고, rank3을 rank2로 낮추기 위해 SVD를 수행하여 S Matrix의 마지막 항을 제거한다.

Feature의 수가 필요한 8개보다 더 많기 때문에 가능한 8개의 포인터를 모두 고려하여 Fundamental Matrix를 구한 후, 오차를 계산하여 가장 작은 오차를 갖는 Fundamental Matrix를 구한다.

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

작성된 코드는 아래와 같다. 위의 내용을 코드로 구현한 것이다.

```
def compute_fundamental(x1, x2):
    n = x1.shape[1]
    if x2.shape[1] != n:
        exit(1)

    ### YOUR CODE BEGINS HERE

    from itertools import combinations

    numbers = list(range(n))
    subsets = list(combinations(numbers, 8)) # 가능한 모든 8개의 point를 선택
    mse = None
    optimal_F = None

    for subset_indexes in subsets:
        A_array = []
        for i in subset_indexes:  # 추출한 8개의 포인터쌍에 대해서...
            # A Matrix를 구한다.
            A_array.append([x1[0][i] * x2[0][i], x1[1][i] * x2[0][i], x2[0][i],
                            x1[0][i] * x2[1][i], x1[1][i] * x2[1][i], x2[1][i],
                            x1[0][i], x1[1][i], 1])
        A_np_array = np.array(A_array)  # 배열을 np.array로 변경
        ATA = np.dot(A_np_array.T, A_np_array)  # A^T*A를 구하기

        eigenvalues, eigenvectors = np.linalg.eig(ATA)  # eigenvector/value 구하기
        smallest_eigenvalue_index = np.argmin(eigenvalues)  # 가장 작은 값의 index 구함
        smallest_eigenvector = eigenvectors[:, smallest_eigenvalue_index]  # 가장 작은 값의 eigenvector 획득
        F_temp = np.reshape(smallest_eigenvector, (3, 3))  # eigenvector를 통해 F획득

        U, S, V = np.linalg.svd(F_temp)  # svd 실행
        S[-1] = 0  # rank를 한 단계 낮춤
        S = np.diag(S)
        F_rank_2 = np.dot(np.dot(U, S), V)  # 획득한 F'
        F_rank_2 = F_rank_2/F_rank_2[-1][-1]


        # 모든 포인터 쌍에 대해서 오차 계산
        sum_of_squared_error = 0.0
        for i in range(n):
            sum_of_squared_error += np.sum(np.square(x2[:, i].T - np.dot(F_rank_2, x1[:, i].T)))
        # 오차가 최소일 경우 optimal_F를 업데이트
        if mse is None or mse > sum_of_squared_error:
            mse = sum_of_squared_error
            optimal_F = F_rank_2

    return optimal_F
```

2. Compute Epipole

epipole point를 구한다. epipole point는 모든 epipole line을 지나는 point로 상대 이미지를 촬영한 카메라의 위치를 나타낸다. 대응되는 feature와 epipole point를 연결하여 epipole line을 구할 수 있다.

epipole point는 Fundamental Matrix로 투사시켰을 때, 0이 되는 점이다. SVD의 V Matrix를 통해서 구할 수 있다.

```
def compute_epipoles(F):
    e1 = None
```

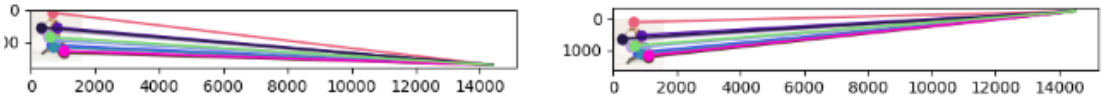
```
    e2 = None
    ### YOUR CODE BEGINS HERE
    # e1 계산
    U, S, V = np.linalg.svd(F)
    e1 = V[-1]
    e1 = e1/e1[2]

    # e2 계산
    U, S, V = np.linalg.svd(F.T)
    e2 = U[-1]
    e2 = e2/e2[2]
    ### YOUR CODE ENDS HERE

    return e1, e2
```

3. Epipolar lines

위에서 얻은 epipolar point와 각 요소들의 점을 연결하여 epipolar line을 그린다.

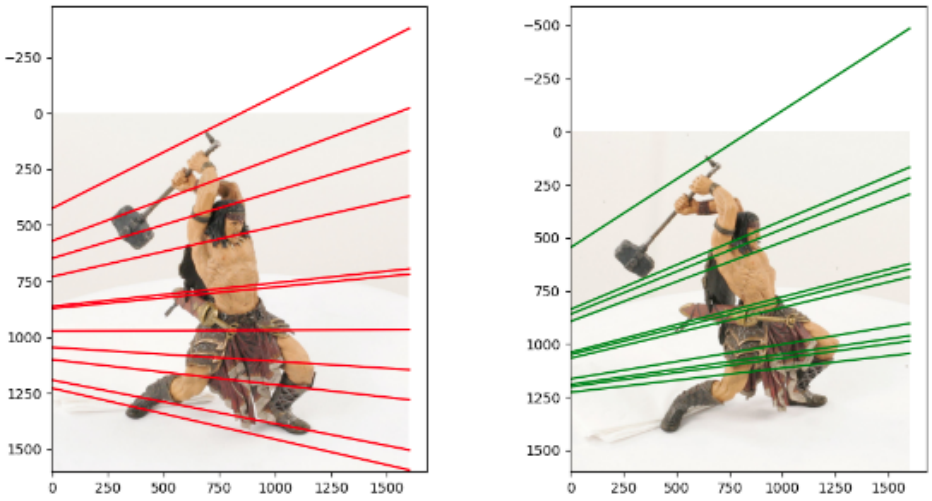
과제에서 요구한대로 각 요소의 점을 찍고 각 요소들을 구별할 수 있로록 색을 달리해야 한다.

```
def draw_epipolar_lines(img1, img2, cor1, cor2):
    F = compute_norm_fundamental(cor1, cor2)
    e1, e2 = compute_epipoles(F)

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

    #랜덤한 사진 색깔 분배
    colors = []
    for i in range(cor1.shape[1]):
        colors.append(np.random.rand(3))

    # img1에 대해서
    ax1.imshow(img1)
    index = 0
    for i in range(cor1.shape[1]):
        # 점 찍기
        ax1.plot(cor1[0, i], cor1[1, i], 'o', color=colors[index])
        # line 그리기
        x = np.array([cor1[0, i], e1[0]])
        y = np.array([cor1[1, i], e1[1]])
        ax1.plot(x, y, color=colors[index])
        index += 1

    # img2에 대해서
    ax2.imshow(img2)
    index = 0
    for i in range(cor2.shape[1]):
        # 점 찍기
        ax2.plot(cor2[0, i], cor2[1, i], 'o', color=colors[index])
        # line 그리기
        x = np.array([cor2[0, i], e2[0]])
        y = np.array([cor2[1, i], e2[1]])
        ax2.plot(x, y, color=colors[index])
        index += 1

    plt.show()
```

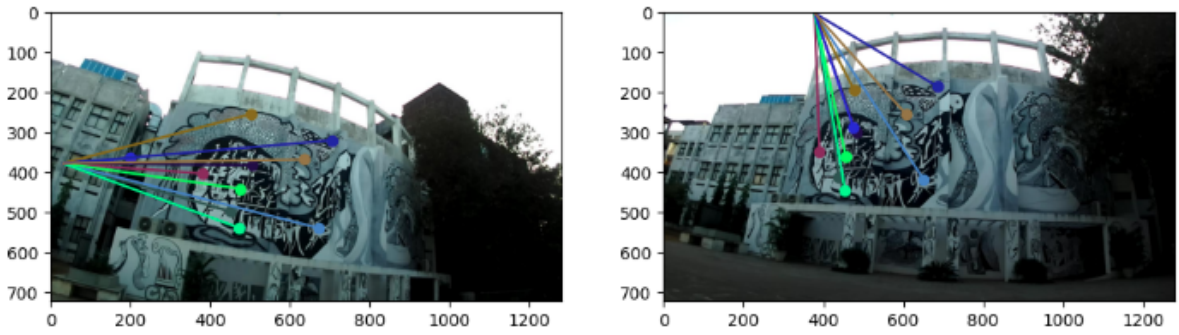


epipolar point의 계산이 잘못된 것으로 예측된다.

그러나 실수로 squared error가 가장 큰 fundamental Matrix를 이용했을 때에는 아래와 같이 그나마 과제 pdf와 유사한 결과를 얻을 수 있었다. 아래의 그림이 그것이다.(좌측에서 우측으로 퍼져나가는 방사형)



아래는 graffiti에 적용한 결과물이다.



3. 결론

epipolar line을 찾아내기 위해서는 아래와 같은 과정이 수행되어야 한다.

1. 두 이미지에서 feature를 찾는다.
2. 찾은 feature들을 서로 매칭한다.
3. 매칭한 feature쌍들을 이용해서 Fundamental Matrix를 구한다.

a. Fundamental Matrix를 구할 때에는 Rank 축소가 필요하다.

4. Fundamental Matrix와 feature point를 이용해서 epipolar line을 그린다.

방식을 통해서 8개의 점을 임의로 골라낸 후 least square를 적용했을 때, 적절한 Fundamental Matrix를 구할 수 없었다.

약간 의문스러운 점은 그저 cor1매트릭스를 SVD하여 얻은 V Matrix를 Fundamental Matrix로 사용했을 때, 더 정답과 유사한 결과를 얻을 수 있었다는 점이다.