Eplploar Geometry

[Code]

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

    F = None
    ### YOUR CODE BEGINS HERE

    # 전치행렬
    t1 = np.transpose(x1)
    t2 = np.transpose(x2)

    # A matrix shape, 형식 지정
    A = np.zeros((9, 9))

    # build matrix for equations in Page 52

    # A matrix 구성.
    for i in range(x1.shape[0]):
        A[i, :] = np.array([[t1[i][0] * t2[i][0],
                             t1[i][1] * t2[i][0],
                             t2[i][0],
                             t1[i][0] * t2[i][1],
                             t1[i][1] * t2[i][1],
                             t2[i][1],
                             t1[i][0],
                             t1[i][1],
                             1]])

    # SVD
    _, _, V =  np.linalg.svd(A)
    F = V[-1].reshape(3, 3)
    U, S, V = np.linalg.svd(F)
    # constrain F: make rank 2 by zeroing out last singular value (Page 53)
    # to rank 2
    S[2] = 0
    F = np.dot(U, np.dot(np.diag(S), V))

    ### YOUR CODE ENDS HERE

    return F
```

[설명]

입력값에 대한 전치 행렬을 구한 뒤에 linear equation을 통해 Aㄹ = 0꼴로 만들어준다. 이후 SVD를 적용하여 F에 대해 구하여 리턴한다.

[Code]

```
def compute_epipoles(F):
    e1 = None
    e2 = None
    ### YOUR CODE BEGINS HERE
    _, _, V = np.linalg.svd(F)
    e1 = V[-1].reshape((3, 1))
```

```
    _, _, V = np.linalg.svd(np.transpose(F))
    e2 = V[-1].reshape((3, 1))

    # 3 번째 요소를 1 로 normalize 시켜준 것을 리턴하기 위해 3 번째 요소로 나눠준다.
    e1 = e1[:2] / e1[2]
    e2 = e2[:2] / e2[2]
    ### YOUR CODE ENDS HERE

    return e1, e2
```

[설명]

Fe꼴을 변환시켜주기 위해서 SVD를 적용한다. 3번째 요소를 1로 normalize 시켜준 후 리턴하기 위해 3번째 요소로 나눠준다.

[Code]

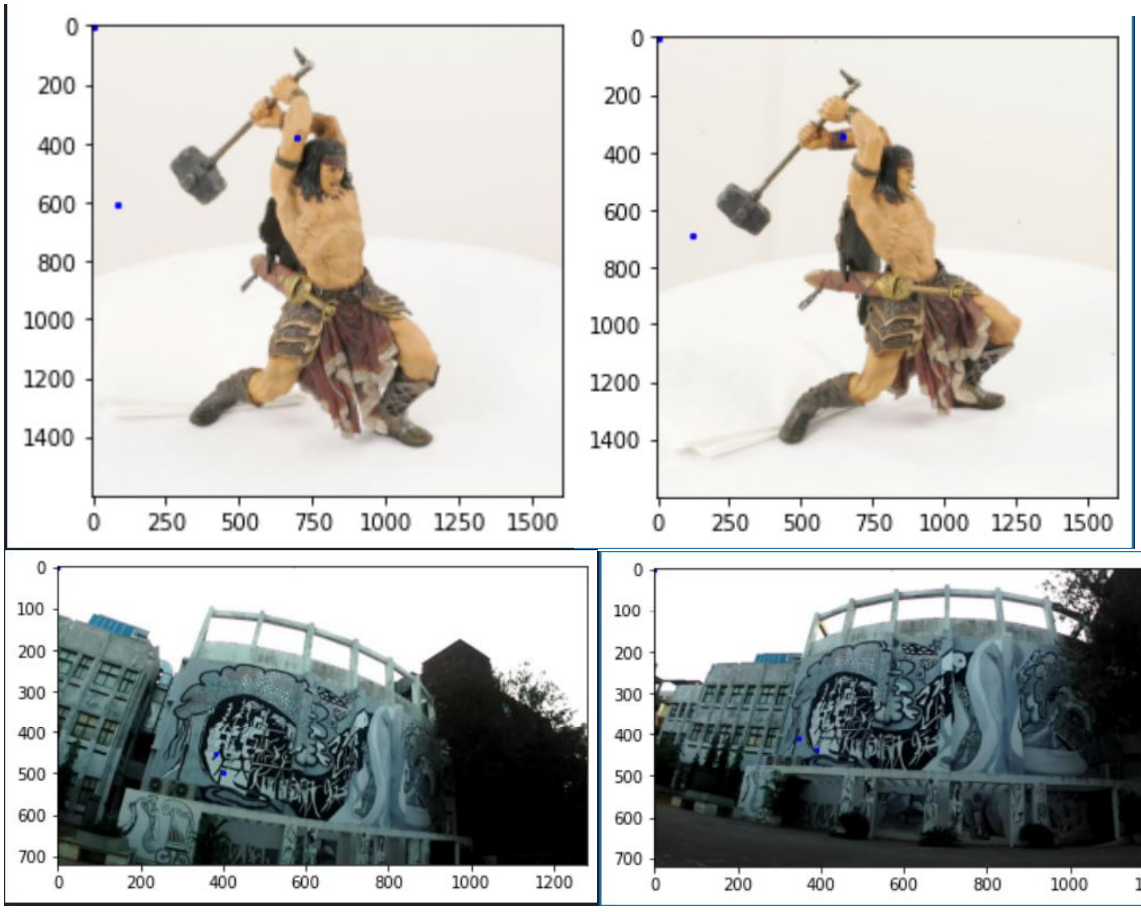
```
def draw_epipolar_lines(img1, img2, cor1, cor2):
    F = compute_norm_fundamental(cor1, cor2)
    e1, e2 = compute_epipoles(F)
    ### YOUR CODE BEGINS HERE
    plt.figure(1)
    plt.imshow(img1
    lim_1, lim_2 = plt.ylim()
    for i in range(0, cor1.shape[0]):
        plt.plot(e1[0], e1[1], c = 'b', linewidth = 0.5)

    plt.scatter(cor2[:, 0], cor2[:, 1], c = 'r', marker = 'o', s = 10)
    plt.ylim(lim_1, lim_2)
    plt.figure(2)
    plt.imshow(img2)
    lLim,rLim = plt.ylim()
    for i in range(0, cor2.shape[0]):
        plt.plot(e2[0], e2[1], c = 'b', linewidth = 0.5)

    plt.scatter(cor1[:,0], cor1[:,1], c = 'r', marker = 'o', s = 10)
    plt.ylim(lim_1, lim_2)
    plt.show()
    ### YOUR CODE ENDS HERE
    return
```

[설명]

앞서 만든 함수를 이용하여 F, e1, e2 를 각각 구한 후에 각 포인터에 맞는 값들을 출력해준다.
matplotlib 를 이용해 Scatter 로각 포인트에 맞는 값들을 출력해준다


0
200
400
600
800
1000
1200
1400
0 250 500 750 1000 1250 1500
0
100
200
300
400
500
600
700
0 200 400 600 800 1000 1200