Homework 4: 3D Reconstruction

Yasser Corzo

November 2023

Q1.1

From class, we know that a set of corresponding points from two cameras must obey this rule:

$$p^T F p^{'} = 0$$

Since both p and $p^{'}$ lie in the origin of their respective planes,

$$p^T F p^{'} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0$$

$$F_{33} = 0$$

Hence, $F_{33} = 0$

Q1.2

Since there's no rotation and the rotation matrix is written as $\begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$, let $\theta = 0$. Hence

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Since the second camera differs from the first by a pure translation that is parallel to the x-axis, $[t]_x = [a, 0, 0]$

$$E = [t]_x R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -a \\ 0 & a & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -a \\ 0 & a & 0 \end{bmatrix}$$

let $p = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ and $p' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$

Epipolar line in camera 2:

$$l^{'} = Ep = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -a \\ 0 & a & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -a \\ ay \end{bmatrix}$$

Similarly for epipolar line in camera 1:

$$l = E^T x' = \begin{bmatrix} 0 \\ a \\ -ay \end{bmatrix}$$

As we see here, the epipolar lines l' and l are horizontal, thus parallel to the x-axis.

Q1.3

Suppose p_i is point 1 in camera 1 at time i and p_j is point 2 in camera 2 at time j. Assuming the camera intrinsics (K) are known, :

$$p_i \equiv M_1 P$$

$$p_i = K[R_i|t_i]P = K[R_i P + t_i]$$

(assuming P is a homogenous 3D point)

$$p_i K^{-1} = R_i * P + t_i$$

$$p_i K^{-1} - t_i = R_i * P$$

$$R_i^{-1}[K^{-1} p_i - t_i] = P$$

Substitute P:

$$p_j \equiv M_2 P = K[R_j|t_j]P = K[R_j|t_j]R_i^{-1}[K^{-1} p_i - t_i]$$

$$= R_i^{-1}(p_i[R_j|t_j] - t_i[R_j|t_j])$$

$$= R_i^{-1}([R_j * p_i + t_j] - [R_j * t_i + t_j])$$

$$= (R_i^{-1} * R_j * p_i + t_j) - (R_i^{-1} * R_j * t_i + t_j)$$

therefore $R_{rel} = R_i^{-1} * R_j$ and $t_{rel} = R_{rel} * t_i$

$$E = [t_{rel}]_x R_{rel}$$

$$F = K^{-T} E K^{-1} = K^{-T}[t_{rel}]_x R_{rel} K^{-1}$$

Q1.4

From Q1.3, we have that $R_{rel} = R_i * R_i^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I$ and $t_{rel} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$

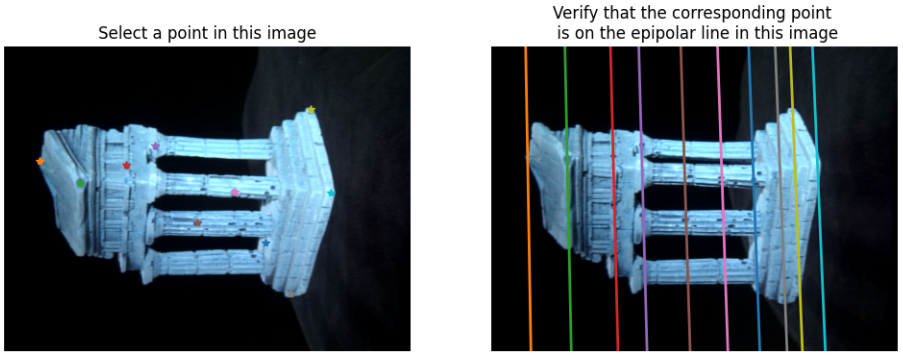
$$\begin{aligned} E = [t_{rel}]_x R_{rel} &= \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= [t_{rel}]_x \end{aligned}$$

$$F = K^{-T} E K^{-1} = K^{-T} \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} K^{-1} \tag{1}$$

$$F^T = (K^{-T} E K^{-1})^T = K^{-T} E^T K^{-1} = K^{-T} \begin{bmatrix} 0 & t_z & -t_y \\ -t_z & 0 & t_x \\ t_y & -t_x & 0 \end{bmatrix} K^{-1} = -F \tag{2}$$

By property of skew symmetry, since $F^T = -F$, F is skew symmetric.

Q2.1




```
F = [[-1.31716374e-09 -1.30582858e-07  1.12462845e-03]
     [-5.94677158e-08  3.55807763e-09 -1.64980639e-05]
     [-1.08023632e-03  3.03025741e-05 -4.10639901e-03]]
```

Q3.1

```
E = [[-3.04477699e-03 -3.02949405e-01  1.66026654e+00]
    [-1.37963814e-01  8.28452408e-03 -5.12671289e-02]
    [-1.66531742e+00 -1.26601678e-02 -1.36557606e-03]]
```

Q3.2

$$\mathbf{A_i} = \begin{bmatrix} yC1_3^T - C1_2^T \\ C1_1^T - xC1_3^T \\ y'C2_3^T - C2_2^T \\ C2_1^T - x'C2_3^T \end{bmatrix}$$

where $C1_i^T$ is the ith row of Camera 1, $C2_j^T$ is the jth row of Camera 2.

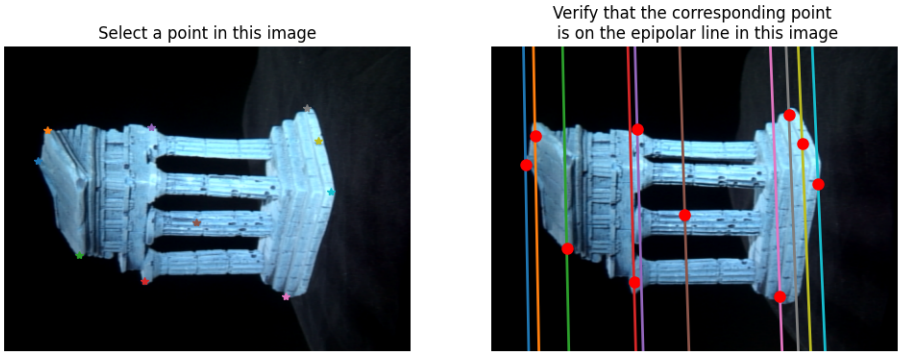
Q3.3

```
Reprojection errors of corresponding M2s from camera2()
89.3935075432261 89.3935075432261 89.39351648101835 89.39351648101835

M2, C2, and P results from findM2.py:

M2: [[ 0.999264    0.03734759  0.00875291 -0.03081067]
 [-0.03833874  0.9648562   0.25996665 -1.        ]
 [ 0.00126383 -0.26011089  0.96557792  0.08290169]]
C2: [[ 1.51966307e+03 -2.18534485e+01  3.05221448e+02 -2.17817004e+01]
 [-5.81890781e+01  1.40806050e+03  6.35055335e+02 -1.50543406e+03]
 [ 1.26382500e-03 -2.60110892e-01  9.65577924e-01  8.29016884e-02]]
P: [[-3.43519876e-01 -3.75810991e-02  3.61191021e+00]
 [ 1.60443732e-02  9.26460930e-02  3.80763319e+00]
 [-3.41604703e-01 -5.15182807e-02  3.59332064e+00]
 [-3.86771137e-01  2.08857173e-01  3.83366122e+00]
 [-2.68207801e-01  1.73649074e-01  3.83292012e+00]
 [ 4.35318879e-04  6.51544274e-02  3.80421435e+00]
 [-3.60076482e-01  1.93518777e-01  3.82862709e+00]
 [-3.57311310e-01 -2.72163919e-01  3.77969007e+00]
 [-1.74340965e-01  2.37026179e-01  3.88358588e+00]
 [-1.56432744e-01  3.51691253e-02  3.79792540e+00]
 [-1.15787915e-01  5.75408787e-02  3.79602614e+00]
 [-2.69412149e-01  1.06204582e-01  3.75756107e+00]
 [-2.90613716e-01  2.06099333e-01  3.82911570e+00]
 [-2.37751620e-01 -8.49060429e-02  3.61178133e+00]
 [-1.28665394e-01  9.23635648e-02  3.79581928e+00]
 [-3.48168293e-01  7.11327953e-02  3.72641964e+00]
 [-2.05018365e-01  1.14440304e-01  3.78548165e+00]
 [-4.18828489e-01  2.02085758e-01  3.75498188e+00]
 [-3.61694219e-01  1.43705749e-01  3.77258608e+00]
 [ 4.39717071e-01  3.52937269e-01  3.92846570e+00]
 [-9.61634805e-02 -1.91745195e-01  3.80620410e+00]
 [-3.73142165e-01  6.31194013e-02  3.68659026e+00]
 [-4.47094454e-03  2.87606993e-01  3.98502609e+00]
 [-3.56199133e-01 -6.27726885e-02  3.56423926e+00]
 [-3.50967998e-01 -5.10049049e-02  3.55811188e+00]
 [-1.21827165e-01  8.94643300e-02  3.77833849e+00]
 [-2.64292275e-01 -2.14768931e-01  3.72943455e+00]
 [ 3.76299032e-01  3.82386493e-01  3.99386547e+00]
 [-3.39678049e-01 -4.50699461e-03  3.66834070e+00]
 [-2.86726714e-01  2.67626545e-01  3.88446137e+00]
 [-3.62808105e-02  1.09914455e-01  3.80065332e+00]
 [ 2.48452875e-01  2.38021863e-01  3.98624318e+00]
 [ 1.43519730e-01  6.53206723e-02  3.81552810e+00]
```

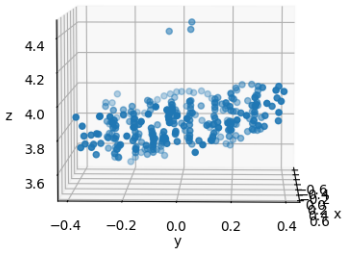
```
[-1.81866395e-01  2.39495678e-01  3.88236207e+00]
[ 4.51990986e-01 -2.26649649e-01  3.68303702e+00]
[-1.13499369e-01 -7.85723203e-02  3.64750767e+00]
[-9.62064934e-02  8.23038300e-02  3.79083722e+00]
[-8.66311100e-02 -3.79373942e-02  3.64784995e+00]
[ 4.41242299e-01 -3.61775725e-01  3.83642561e+00]
[-3.48442687e-01 -2.70506988e-01  3.79119552e+00]
[-2.50642690e-01 -2.30516073e-01  3.74704167e+00]
[-2.64175774e-01  5.38111990e-02  3.71043232e+00]
[-3.64077755e-01  2.12602476e-01  3.85595324e+00]
[-3.68778294e-01 -3.96275383e-02  3.58249974e+00]
[ 1.88030257e-01  9.04869466e-02  3.82315046e+00]
[-1.90675224e-01  1.57499737e-01  3.80690145e+00]
[-3.82911273e-01 -2.85752737e-01  3.76289275e+00]
[-3.69387514e-01 -2.79337427e-01  3.77619512e+00]
[-1.63741742e-01  3.01801905e-02  3.79651540e+00]
[-1.98555540e-01  1.57837988e-01  3.81508307e+00]
[ 2.55068381e-01 -2.86023986e-02  3.67055133e+00]
[-2.54535354e-01 -1.38569445e-01  3.65364060e+00]
[ 2.86646921e-01 -2.63402532e-02  3.69274448e+00]
[-5.96509761e-01 -1.56261273e-01  3.67389994e+00]
[-1.13902101e-01 -1.05513386e-01  3.67002313e+00]
[ 2.80825683e-01  3.27359718e-01  4.02518929e+00]
[ 4.76840337e-01 -2.28729127e-01  3.67711608e+00]
[-1.65606714e-01  7.88365843e-02  3.74406533e+00]
[-5.71488388e-01 -2.21453375e-01  3.63744979e+00]
[-1.06068611e-01 -7.84071990e-02  3.63984390e+00]
[ 1.64300647e-02  7.24939703e-02  3.79752276e+00]
[ 2.50227730e-02 -6.44583580e-02  3.66013351e+00]
[ 4.31686819e-02  1.02737801e-01  3.81192676e+00]
[ 3.57126720e-02  1.02709767e-01  3.81084597e+00]
[ 1.92686812e-01  1.05546286e-01  3.82437400e+00]
[ 1.96543829e-01  8.28125330e-02  3.81485707e+00]
[-1.19122283e-01 -1.93075947e-01  3.78346274e+00]
[ 1.94181642e-01  1.13372764e-01  3.83423663e+00]
[ 1.56074841e-01 -4.31201104e-02  3.68128638e+00]
[ 2.76981789e-01 -3.11596397e-02  3.69085335e+00]
[ 2.79050826e-02  1.26087111e-01  3.83837028e+00]
[-3.29015888e-01  1.96748732e-01  3.84260941e+00]
[-2.82446804e-01 -2.27811735e-01  3.74292375e+00]
[-2.54307271e-01 -1.87475588e-02  3.63503924e+00]
[-1.04842054e-01 -1.94153459e-01  3.80458888e+00]
[-3.44906632e-01 -2.33258375e-01  3.75156892e+00]
[ 2.57375908e-02 -9.86193687e-02  3.68195697e+00]
[ 3.46545123e-01 -3.19559644e-01  3.87349260e+00]
[-8.36864670e-02 -1.91837640e-01  3.80803646e+00]
```

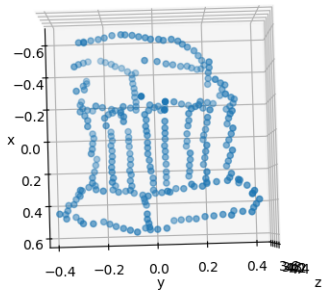
```
[ 3.15249644e-01  3.49640393e-01  4.00820481e+00]
[-3.40176325e-01 -1.45948043e-01  3.65815892e+00]
[ 2.33500546e-01  8.05866410e-02  3.82930823e+00]
[-4.43303540e-01 -2.99452179e-01  3.77996255e+00]
[-1.52700492e-01 -2.21645609e-01  3.76332897e+00]
[ 4.31506561e-01 -3.64401614e-01  3.86422051e+00]
[-1.08990067e-01 -4.98291986e-02  3.64317435e+00]
[-4.70853229e-01 -2.80683098e-01  3.76079427e+00]
[-2.50496782e-01 -6.89359334e-03  3.66310327e+00]
[ 1.76535661e-01  1.15813547e-01  3.83174511e+00]
[-3.63453299e-01 -6.02979998e-02  3.55529907e+00]
[-3.11463202e-01  1.98418257e-01  3.82637627e+00]
[ 2.94516786e-01 -5.74909746e-02  3.67259071e+00]
[ 3.43513037e-02 -5.01124061e-02  3.66325965e+00]
[-2.60113540e-01 -3.01412934e-01  3.83683810e+00]
[ 3.21409659e-02 -6.19347987e-02  3.65273302e+00]
[-1.86127720e-01 -1.51587544e-01  3.67910765e+00]
[-1.69901948e-01  7.12838058e-02  3.73402725e+00]
[-2.47385510e-01 -1.40608890e-02  3.65405121e+00]
[-4.51872446e-01 -2.99322964e-01  3.77853313e+00]
[-4.78161466e-01 -2.80603077e-01  3.75971558e+00]
[ 1.88096743e-01 -2.62447521e-02  3.67886607e+00]
[-8.46647693e-03  7.75930862e-02  3.80343987e+00]
[-2.84926220e-01 -2.77110314e-01  3.77963283e+00]
[ 1.70133622e-01  1.05462456e-01  3.82110758e+00]
[-1.58351129e-01 -2.90445995e-01  3.82490723e+00]
[-1.59171559e-01 -9.27803872e-03  3.65821452e+00]
[ 2.82358638e-01 -5.74623319e-02  3.67089656e+00]
[ 4.78952739e-01 -2.60276220e-01  3.71551495e+00]
[ 1.75473029e-02 -5.00787428e-02  3.66090430e+00]
[-1.63449009e-01 -2.16133810e-01  3.75325501e+00]]
```

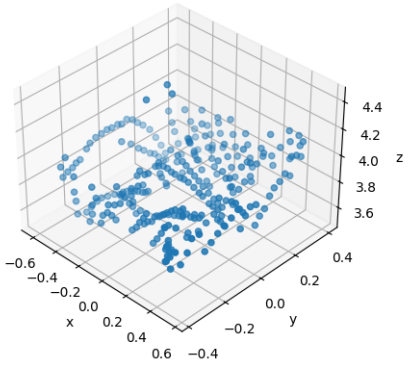
Q4.1

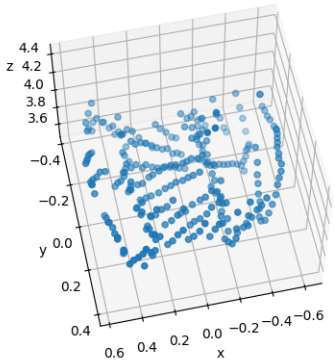

Select a point in this image
Verify that the corresponding point
is on the epipolar line in this image

Q4.2


4.4
4.2
z 4.0
3.8
3.6
−0.4 −0.2 0.0 0.2 0.4
y
x


−0.6
−0.4
−0.2
x 0.0
0.2
0.4
0.6
−0.4 −0.2 0.0 0.2 0.4
y
z


4.4
4.2
4.0
z
3.8
3.6
−0.6 −0.4 −0.2 0.0 0.2 0.4 0.6
x
0.4 0.2 0.0 −0.2 −0.4
y


4.4
z 4.2
4.0
3.8
3.6
−0.4
−0.2
y 0.0
0.2
0.4
0.6 0.4 0.2 0.0 −0.2 −0.4 −0.6
x

Q5.1

Resulting F determined from noisy correspondences without ransac, F calculated with ransac (F ransac) and the number of inliers detected with RANSAC.

```
    F = [[ 5.32073592e-07 -8.48584037e-07  3.27677346e-05]
         [ 1.27977927e-06 -1.46055928e-06  2.96569125e-05]
         [-3.98530247e-04  5.69253593e-04 -1.89251357e-02]]
F_ransac = [[-2.63825183e-07  1.34303194e-06 -2.30161965e-04]
            [-1.05271305e-06 -1.35594587e-06  5.56990379e-04]
            [ 3.02336409e-04 -2.58200346e-05 -6.87297977e-02]]
number of inliers detected = 106
```

The metric I used to determine whether a certain point was an inlier was by using the geometric distance from a point to a line. In this case, p' is our point and the epipolar line (produced by $F * p$) is l'. The geometric distance is defined as follows:

$$d(p', l') = \frac{p'^T l'}{\sqrt{(l')_1^2 + (l')_2^2}} \tag{3}$$

If the result of equation (3) was less than the tolerance, then p' is considered an outlier. Other optimizations I made was changing the default tolerance of the function to 0.67.

Q5.2

```
'''
Q5.2:Extra Credit  Rodrigues formula.
    Input:  r, a 3x1 vector
    Output: R, a rotation matrix
'''
def rodrigues(r):
    theta = np.linalg.norm(r)
    I = np.eye(3)
    if theta == 0:
        return I
    r = r / theta
    kx = r[0, 0]
    ky = r[1, 0]
    kz = r[2, 0]
    K = np.array(([0, -kz, ky],
                  [kz, 0, -kx],
                  [-ky, kx, 0]))
    return I*np.cos(theta) + (1 - np.cos(theta))*(r @ r.T) + K*np.sin(theta)

'''
Q5.2:Extra Credit  Inverse Rodrigues formula.
    Input:  R, a rotation matrix
    Output: r, a 3x1 vector
'''
def invRodrigues(R):
    '''
    theta = np.arccos((np.trace(R) - 1) / 2)
    w = (1 / (2 * np.sin(theta))) * np.array(([R[2, 1] - R[1, 2]],
                                              [R[0, 2] - R[2, 0]],
                                              [R[1, 0] - R[0, 2]]))
    r = theta * w
    '''
    # source: https://courses.cs.duke.edu/cps274/fall13/notes/rodrigues.pdf
    A = (R - R.T) / 2
    rho = np.array([A[2, 1], A[0, 2], A[1, 0]]).reshape(-1, 1)
    s = np.linalg.norm(rho)
    c = (np.trace(R) - 1) / 2
    if (s == 0) and (c == 1):
        r = np.zeros((3, 1))

    elif (s == 0) and (c == -1):
        # let v be a non-zero column of R + I
        v_temp = R + np.eye(len(R))
        v = None
```

```
    for j in range(v_temp.shape[1]):
        # check whether column j is non-zero
        col = v_temp[:, j]
        if np.all(col != 0):
            v = col
            break
    u = v / np.linalg.norm(v)
    r = u * np.pi
    # modeled after function S_1/2
    if (np.linalg.norm(r) == np.pi) and ((r[0] == 0 and r[1] == 0 and r[2] < 0)
        or (r[0] == 0 and r[1] < 0) or (r[0] < 0)):
        r = -r
    r = r.reshape((3, 1))

theta = np.arccos((np.trace(R) - 1) / 2)
if (np.sin(theta) != 0):
    u = rho / s
    theta = np.arctan2(s, c)
    r = u * theta
return r
```

Q5.3

```
'''
Q5.3: Extra Credit Rodrigues residual.
    Input:  K1, the intrinsics of camera 1
            M1, the extrinsics of camera 1
            p1, the 2D coordinates of points in image 1
            K2, the intrinsics of camera 2
            p2, the 2D coordinates of points in image 2
            x, the flattened concatenationg of P, r2, and t2.
    Output: residuals, 4N x 1 vector, the difference between original and estimated projecti
'''
def rodriguesResidual(K1, M1, p1, K2, p2, x):
    # retrieve number of 2D coordinates
    N = p1.shape[0]

    # break up x into 3D coords P, r2, and t2
    t2 = x[-3:]
    r2 = x[-6:-3]
    P = x[:3 * N].reshape(N, 3)

    # make 3D coordinate homogenous (N x 4)
    P_homogenous = np.hstack((P, np.ones(N).reshape(-1, 1)))

    # after getting r2, get R2 (with Rodriguez function)
    R2 = rodrigues(r2)

    # after getting R2, get M2 = [R2 | t2]
    M2 = np.hstack((R2, t2))

    # get C1 (K1 @ M1) and C2 (K2 @ M2)
    C1 = K1 @ M1
    C2 = K2 @ M2

    # calculate p1_hat and p2_hat (multiply Ci @ 3D coords)
    p1_hat_homogenous = (C1 @ P_homogenous.T).T
    p2_hat_homogenous = (C2 @ P_homogenous.T).T

    # convert homogenous coordinates to heterogenous
    p1_hat = p1_hat_homogenous[:, :-1] / p1_hat_homogenous[-1]
    p2_hat = p2_hat_homogenous[:, :-1] / p2_hat_homogenous[-1]

    residuals = np.concatenate([(p1-p1_hat).reshape([-1]), (p2-p2_hat).reshape([-1])])
    return residuals

'''
```

```
Q5.3 Extra Credit  Bundle adjustment.
    Input:  K1, the intrinsics of camera 1
            M1, the extrinsics of camera 1
            p1, the 2D coordinates of points in image 1
            K2,  the intrinsics of camera 2
            M2_init, the initial extrinsics of camera 1
            p2, the 2D coordinates of points in image 2
            P_init, the initial 3D coordinates of points
    Output: M2, the optimized extrinsics of camera 1
            P2, the optimized 3D coordinates of points
'''
def bundleAdjustment(K1, M1, p1, K2, M2_init, p2, P_init):
    # Replace pass by your implementation
    pass
```