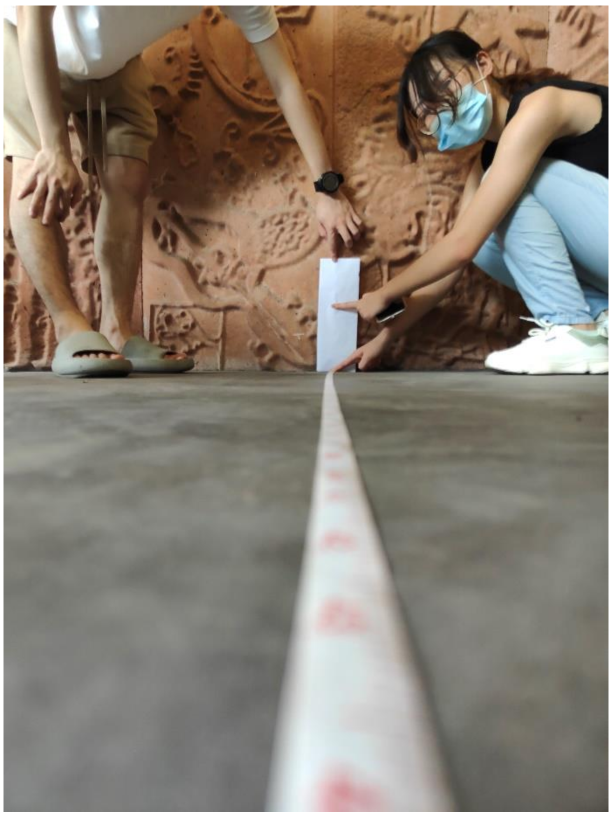
EE6222 Assignment-2

Group Members:

No.	Name	Matric No.
1	Gu Xiuze	G2101686B
2	He Yixuan	G2101553D
3	Liu Jianwei	G2101588E
4	Peng Mingjian	G2101567A
5	Tong Hua	G2101161L
6	Wu Tianwei	G2101446F

1. Find the focal length f of your hand phone (in pixels). You may use real person or printed figure and include one figure of the settings in your report. Make sure you turn the camera's "zooming/auto-focusing" off.



To determine the height of real figure H easily, we placed a piece of A4 size paper against the wall and shot this photo using a hand phone with zooming / auto-focusing off. The photo is shot 1.5m away from the wall. So, the object distance is $L = 1.5m$ and the height of the object is $H = 0.297m$. With the help of IrfanView, we calculated the height of this A4 paper in the photo is $h = 574\,pixels$.

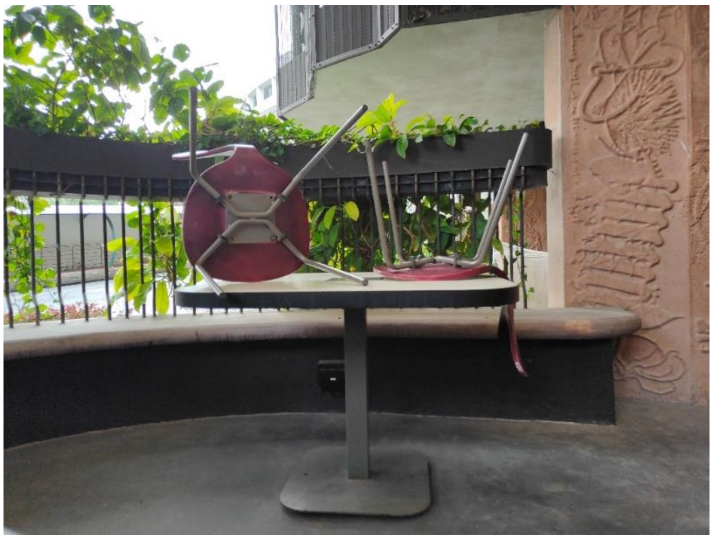
Therefore, the focal length of the hand phone can be computed as

$$f = \frac{L}{H}h = \frac{1.5}{0.297} \times 574 = 2899\,pixels$$

2. Take two snaps of an outdoor scene, with 5 to 10 degrees angle difference. You need to keep the angle as ground truth.

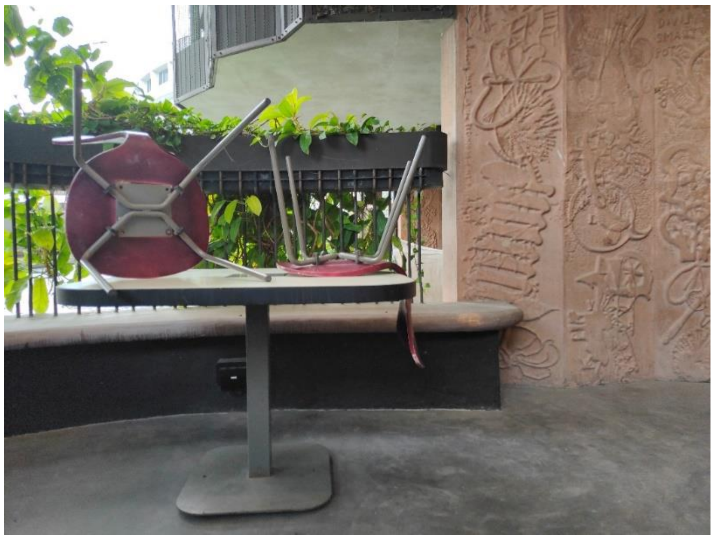
We chose to take two snaps with 10 degrees angle difference.

This is the first taken snap.

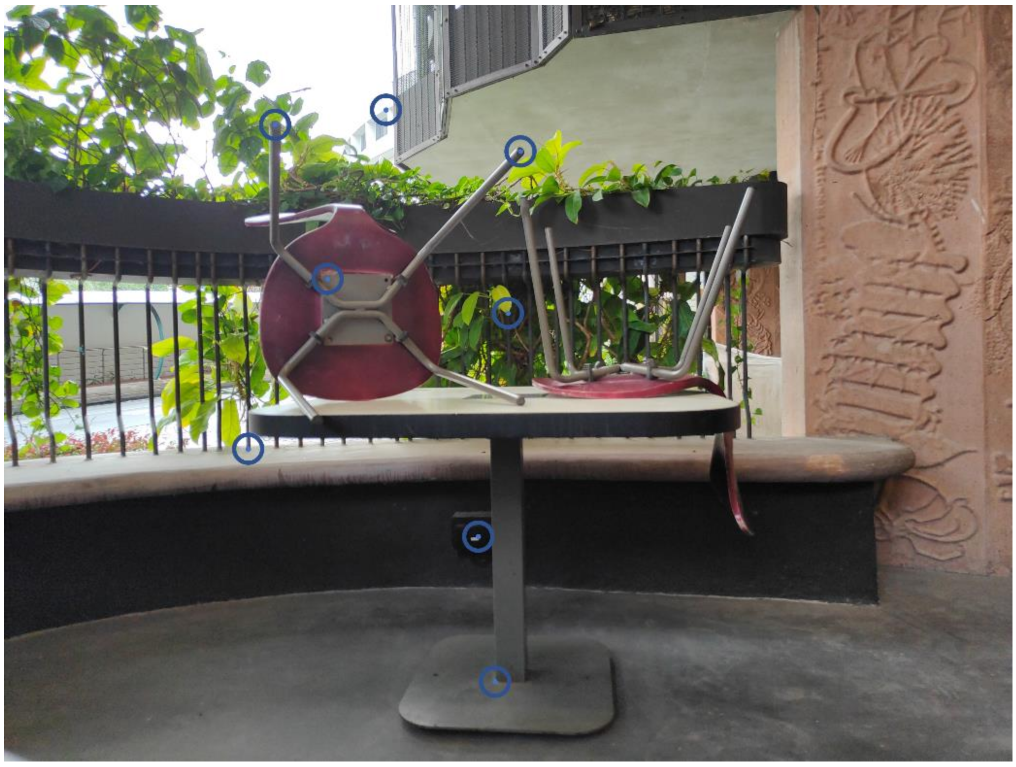


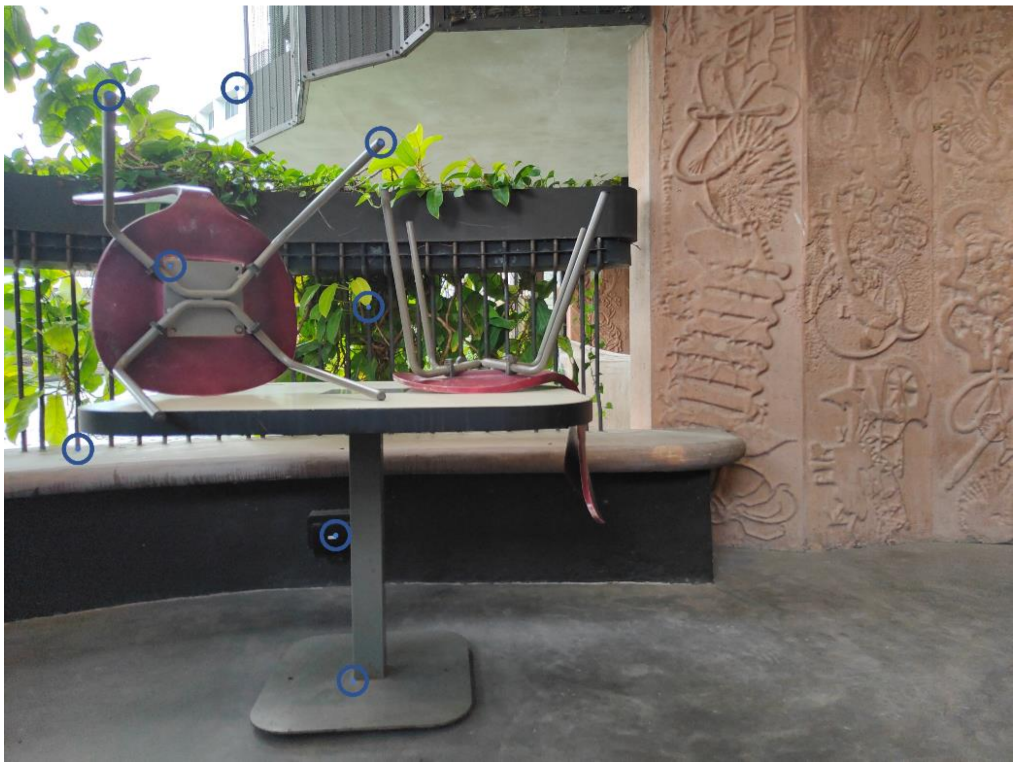
Original

Then, we rotated the hand phone 10° counterclockwise and take the second snap.


After rotation

3. Hand pick 8 points or more from one image and find the matching points on the other image. These points should not be co-planar. You need to turn these points into N-vector and submit them into the equation for calculation.


Original


After rotation

From each snap, we picked 8 points. The points set is

$$\begin{bmatrix} 1097 & 2092 & 1313 & 2044 & 992 & 1926 & 1997 & 1553 \\ 469 & 576 & 1111 & 1255 & 1805 & 2160 & 2746 & 423 \\ 2899 & 2899 & 2899 & 2899 & 2899 & 2899 & 2899 & 2899 \end{bmatrix}$$

in the first snap and the matching points set is

$$\begin{bmatrix} 427 & 1523 & 675 & 1476 & 300 & 1347 & 1420 & 945 \\ 346 & 536 & 1056 & 1227 & 1803 & 2151 & 2738 & 335 \\ 2899 & 2899 & 2899 & 2899 & 2899 & 2899 & 2899 & 2899 \end{bmatrix}$$

in the second snap.

Turning these points into N-vectors, we get two N-vector sets:

$$\begin{bmatrix} 0.3499 & 0.5777 & 0.3895 & 0.5432 & 0.2790 & 0.4702 & 0.4473 & 0.4684 \\ 0.1496 & 0.1591 & 0.3296 & 0.3335 & 0.5076 & 0.5273 & 0.6151 & 0.1276 \\ 0.9248 & 0.8006 & 0.8600 & 0.7705 & 0.8152 & 0.7077 & 0.6493 & 0.8743 \end{bmatrix}$$

and

$$\begin{bmatrix} 0.1447 & 0.4590 & 0.2137 & 0.4245 & 0.0875 & 0.3496 & 0.3355 & 0.3081 \\ 0.1173 & 0.1615 & 0.3344 & 0.3529 & 0.5261 & 0.5583 & 0.6468 & 0.1092 \\ 0.9825 & 0.8736 & 0.9179 & 0.8338 & 0.8459 & 0.7524 & 0.6849 & 0.9451 \end{bmatrix}$$

4. Calculate the rotation angle from the matched points using the quaternion approach (pp 14 in [4]), or the SVD (in [3]).

Quaternion approach:

Here, we chose the positive weights as

$$W = [9 \quad 6 \quad 6 \quad 8 \quad 9 \quad 12 \quad 11 \quad 17]$$

Then, the correlation matrix can be calculated as

$$K = \begin{bmatrix} 10.6869 & 11.7672 & 29.1549 \\ 7.7382 & 12.4640 & 21.5052 \\ 17.7148 & 20.4969 & 53.7840 \end{bmatrix}$$

Using correlation matrix K, the four-dimensional symmetric matrix is calculated as

$$\widehat{K} = \begin{bmatrix} 76.9349 & -1.0083 & 11.4401 & -4.0291 \\ -1.0083 & -55.5611 & 19.5054 & 46.8698 \\ 11.4401 & 19.5054 & -52.0070 & 42.0021 \\ -4.0291 & 46.8698 & 42.0021 & 30.6331 \end{bmatrix}$$

The eigenvalue of $\widehat{K}$ are

$$\sigma_1 = -76.2719, \sigma_2 = -70.6003, \sigma_3 = 68.9262, \sigma_4 = 77.9460$$

Corresponding eigenvector of each eigenvalue are

$$\xi_1 = \begin{bmatrix} 0.0076 \\ 0.9259 \\ -0.1428 \\ -0.3496 \end{bmatrix}, \xi_2 = \begin{bmatrix} 0.0818 \\ 0.0024 \\ -0.9198 \\ 0.3838 \end{bmatrix}, \xi_3 = \begin{bmatrix} -0.0295 \\ 0.3777 \\ 0.3550 \\ 0.8547 \end{bmatrix}, \xi_4 = \begin{bmatrix} -0.9962 \\ -0.0040 \\ -0.0871 \\ 0.0035 \end{bmatrix}$$

So, the largest eigenvalue is 77.9460 and its four-dimensional unit eigenvector is

$$\hat{q} = \begin{bmatrix} -0.9962 \\ -0.0040 \\ -0.0871 \\ 0.0035 \end{bmatrix}$$

Therefore, the rotation R is

$$\begin{bmatrix} 0.9848 & 0.0077 & 0.1736 \\ -0.0064 & 0.9999 & -0.0085 \\ -0.1737 & 0.0073 & 0.9848 \end{bmatrix}$$

Finally, the rotation angle is computed as

$$[-0.3695, 10.0008, 0.4241]$$

The calculated rotation angle 10.0008° is quite close to the actual rotation angle 10°. The error rate is 0.008%.

Matlab Code:

```
clc;
clear;

X = [1097, 469; 2092, 576; 1313, 1111; 2044, 1255; 992, 1805; 1926,
2160; 1997, 2746; 1553, 423];
P = [427, 346; 1523, 536; 675, 1056; 1476, 1227; 300, 1803; 1347,
2151; 1420, 2738; 945, 335];

% Quaternion
[angle_Q, K, K_hat, v, s, q_hat, R_Q] =
rotation_angle_quaternion(X', P', 2899);
angle_Q = angle_Q ./ pi .* 180;
```

```
% X,P ---- points set
% f ---- focal length(pixel)
function [angle, K, K_hat, v, s, q_hat, R]  =
rotation_angle_quaternion(X, P, f)
    [~, cols] = size(X);
    NuX = zeros(3, cols);
    NuP = zeros(3, cols);

    % Calculate N-vector
    for i = 1 : cols
        NuX(:, i) = [X(1, i); X(2, i); f] ./ sqrt(X(1, i)^2 + X(2,
i)^2 + f^2);
        NuP(:, i) = [P(1, i); P(2, i); f] ./ sqrt(P(1, i)^2 + P(2,
i)^2 + f^2);
    end

    % Calculate K and K_hat
    W = [9 6 6 8 9 12 11 17];
    K = zeros(3, 3);
    for i = 1 : cols
        K = K + W(i) * NuX(:, i) * NuP(:, i)';
    end
    K_hat = [K(1, 1) + K(2, 2) + K(3, 3), K(3, 2) - K(2, 3), K(1,
3) - K(3, 1), K(2, 1) - K(1, 2);
        K(3, 2) - K(2, 3), K(1, 1) - K(2, 2) - K(3, 3), K(1, 2) +
K(2, 1), K(3, 1) + K(1, 3);
        K(1, 3) - K(3, 1), K(1, 2) + K(2, 1), - K(1, 1) + K(2, 2) -
K(3, 3), K(2, 3) + K(3, 2);
       K(2, 1) - K(1, 2), K(3, 1) + K(1, 3), K(2, 3) + K(3, 2), -
K(1, 1) - K(2, 2) + K(3, 3)];
    [v, s] = eig(K_hat); % v -- eigenvector matrix; s -- eigenvalue
matrix
    s_max = s(1, 1);
    index = 1;
    for i = 2 : 4
        if s(i, i) > s_max
            s_max = s(i, i);
            index= i;
        end
    end
    q = v(:, index);
    q0 = q(1, 1) / sqrt(q(1, 1)^2 + q(2, 1)^2 + q(3, 1)^2 + q(4,
1)^2);
    q1 = q(2, 1) / sqrt(q(1, 1)^2 + q(2, 1)^2 + q(3, 1)^2 + q(4,
1)^2);
    q2 = q(3, 1) / sqrt(q(1, 1)^2 + q(2, 1)^2 + q(3, 1)^2 + q(4,
1)^2);
    q3 = q(4, 1) / sqrt(q(1, 1)^2 + q(2, 1)^2 + q(3, 1)^2 + q(4,
1)^2);
    sum = q(1, 1)^2 + q(2, 1)^2 + q(3, 1)^2 + q(4, 1)^2;
    disp(sum);
    q_hat = [q0; q1; q2; q3];

    R = [q0^2 + q1^2 - q2^2 - q3^2, 2 * (q1 * q2 - q0 * q3), 2 *
(q1 * q3 + q0 * q2);
       2 * (q2 * q1 + q0 * q3), q0^2 - q1^2 + q2^2 - q3^2, 2 * (q2
* q3 - q0 * q1);
       2 * (q3 * q1 - q0 * q2), 2 * (q3 * q2 + q0 * q1), q0^2 -
q1^2 - q2^2 + q3^2];
    angle = rotm2eul(R);
end
```

SVD:

Using the two N-vector sets, the mass center of the two set can be calculated as

$$\mu_x = \begin{bmatrix} 0.4407 \\ 0.3437 \\ 0.8003 \end{bmatrix}, \mu_y = \begin{bmatrix} 0.2903 \\ 0.3508 \\ 0.8545 \end{bmatrix}$$

Then, the matrix W can be computed as

$$W = \sum_{i=1}^{N_p} x_i' p_i'^T = \begin{bmatrix} 0.0893 & -0.0306 & -0.0146 \\ -0.0171 & 0.2799 & -0.1148 \\ -0.0407 & -0.1093 & 0.0620 \end{bmatrix}$$

Applying SVD to W, we get U, V and the singular value of W:

$$U = \begin{bmatrix} -0.0744 & -0.8914 & 0.4470 \\ 0.9229 & 0.1083 & 0.3695 \\ -0.3778 & 0.4400 & 0.8146 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.0215 & -0.9589 & 0.2831 \\ 0.9201 & 0.0918 & 0.3808 \\ -0.3911 & 0.2687 & 0.8803 \end{bmatrix}$$

$$\sigma_1 = 0.3281, \sigma_2 = 0.1037, \sigma_3 = 0.0017$$

It's obvious that $\sigma_1 \geq \sigma_2 \geq \sigma_3$. So, the rotation R is computed as

$$R = UV^T = \begin{bmatrix} 0.9829 & 0.0199 & 0.1831 \\ -0.0190 & 0.9998 & -0.0065 \\ -0.1831 & 0.0029 & 0.9831 \end{bmatrix}$$

Finally, the rotation angle is

$$[-1.1096, 10.5530, 0.1702]$$

The calculated rotation angle $10.5530°$ is also close to the actual rotation angle $10°$. The error rate is 5.53%.

Matlab Code:

```
clc;
clear;

X = [1097, 469; 2092, 576; 1313, 1111; 2044, 1255; 992, 1805; 1926,
2160; 1997, 2746; 1553, 423];
P = [427, 346; 1523, 536; 675, 1056; 1476, 1227; 300, 1803; 1347,
2151; 1420, 2738; 945, 335];

% SVD
[angle_SVD, Ex, Ep, W, U, S, V, R_SVD] = rotation_angle_SVD(X', P',
2899);
angle_SVD = angle_SVD ./ pi .* 180;
```

```
% X,P ---- points set
% f ---- focal length(pixel)
function [angle, Ex, Ep, W, U, S, V, R] = rotation_angle_SVD(X, P,
f)
    [~, cols] = size(X);
    NuX = zeros(3, cols);
    NuP = zeros(3, cols);

    % Calculate N-vector
    for i = 1 : cols
        NuX(:, i) = [X(1, i); X(2, i); f] ./ sqrt(X(1, i)^2 + X(2,
i)^2 + f^2);
        NuP(:, i) = [P(1, i); P(2, i); f] ./ sqrt(P(1, i)^2 + P(2,
i)^2 + f^2);
    end

    % Find the mass center
    Ex = zeros(3, 1);
    Ep = zeros(3, 1);
    for i = 1 : cols
        Ex = Ex + NuX(:, i);
        Ep = Ep + NuP(:, i);
    end
    Ex = Ex / cols;
    Ep = Ep / cols;

    % Calculate W
    X_1 = NuX - Ex;
    P_1 = NuP - Ep;
    W = X_1 * P_1';

    % Using SVD to find rotation angle
    [U, S, V] = svd(W);
    disp(S); % delta_1 > = delta_2 >= delta_3
    R = U * V';
    angle = rotm2eul(R);
end
```

Error Analysis:

1. The selected points may have three coplanar points in space.

2. When taking two snaps, the rotation angle itself is not accurate enough. Because we rotated the hand phone manually with reference to the actual 10°, the rotation angle is deviated. As we can see from the results that the x-axis and z-axis also have a small amount of rotation.

3. When picking points, the corresponding points of the two images may deviate when reading pixel values in IrfanView.

Comparison of two method:

Comparing the two results of quaternion approach and SVD, we can see that the two results are close to each other, but in this experiment quaternion approach has better performance than SVD.