Course: CS 512 Assignment-5

Name: Arpit Hasmukhbhai Patel

CWID: A20424085

Illinois Institute of Technology

November 14, 2018

❖ Problem Statement

In this assignment, I need to implement a camera calibration algorithm under the assumption of noisy data. In this assignment I implemented non-coplanar calibration. RANSAC should be used to eliminate outliers.

1. Write a program to extract feature points from the calibration algorithm and show them on the image.
2. Write program that do the calibration process from a text file with 3D points and its correspondent 2D points. The program must display MSE (Mean square error), intrinsic and extrinsic parameters.
3. Implement RANSAC algorithm for robust estimation. Your implementation of the RANSAC algorithm should include automatic estimation of the number of draws and of the probability that a data point is an inlier. The final value of these estimates should be displayed by the program. In your estimation of these values, assume a desired probability of 0.99 that at least one of the draws is free from outliers. Set a maximum number of draws that can be performed. When testing the program on noisy data you will note that RANSAC is not handling well one of the provided cases. Explain the reason for RANSAC not being able to handle this case properly. Parameters used in the RANSAC algorithm should be read from a text file named "RANSAC.config".
4. In addition to test your program with testing files, make sure to test it with real images of a calibration target that you generate.

❖ Proposed Solution

- I used OpenCV functions which professor mentioned in the assignment called cvFindChessboardCorners, cvFindCornerSubPix, cvDrawChessBoardCorners, to extract feature points from the calibration target.
- Non-planer calibration parameters

$$\begin{aligned}
|\rho| &= 1/|a_3| \\
u_0 &= |\rho|^2 a_1 \cdot a_3 \\
v_0 &= |\rho|^2 a_2 \cdot a_3 \\
\alpha_v &= \sqrt{|\rho|^2 a_2 \cdot a_2 - v_0^2} \\
s &= |\rho|^4/\alpha_v (a_1 \times a_3) \cdot (a_2 \times a_3) \\
\alpha_u &= \sqrt{|\rho|^2 a_1 \cdot a_1 - s^2 - u_0^2} \\
K^* &= \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \\
\epsilon &= \operatorname{sgn}(b_3) \\
T^* &= \epsilon|\rho|(K^*)^{-1} b \\
r_3 &= \epsilon|\rho| a_3 \\
r_1 &= |\rho|^2/\alpha_v a_2 \times a_3 \\
r_2 &= r_3 \times r_1 \\
R^* &= [r_1^T \;\; r_2^T \;\; r_3^T]^T
\end{aligned}$$

- Mean Square error

$$mean\ square\ error = \frac{\sum_{i=1}^{n}(x_i - \frac{m_1^T p_i}{m_3^T P_i})^2 + (y_i - \frac{m_2^T p_i}{m_3^T P_i})^2}{n}$$

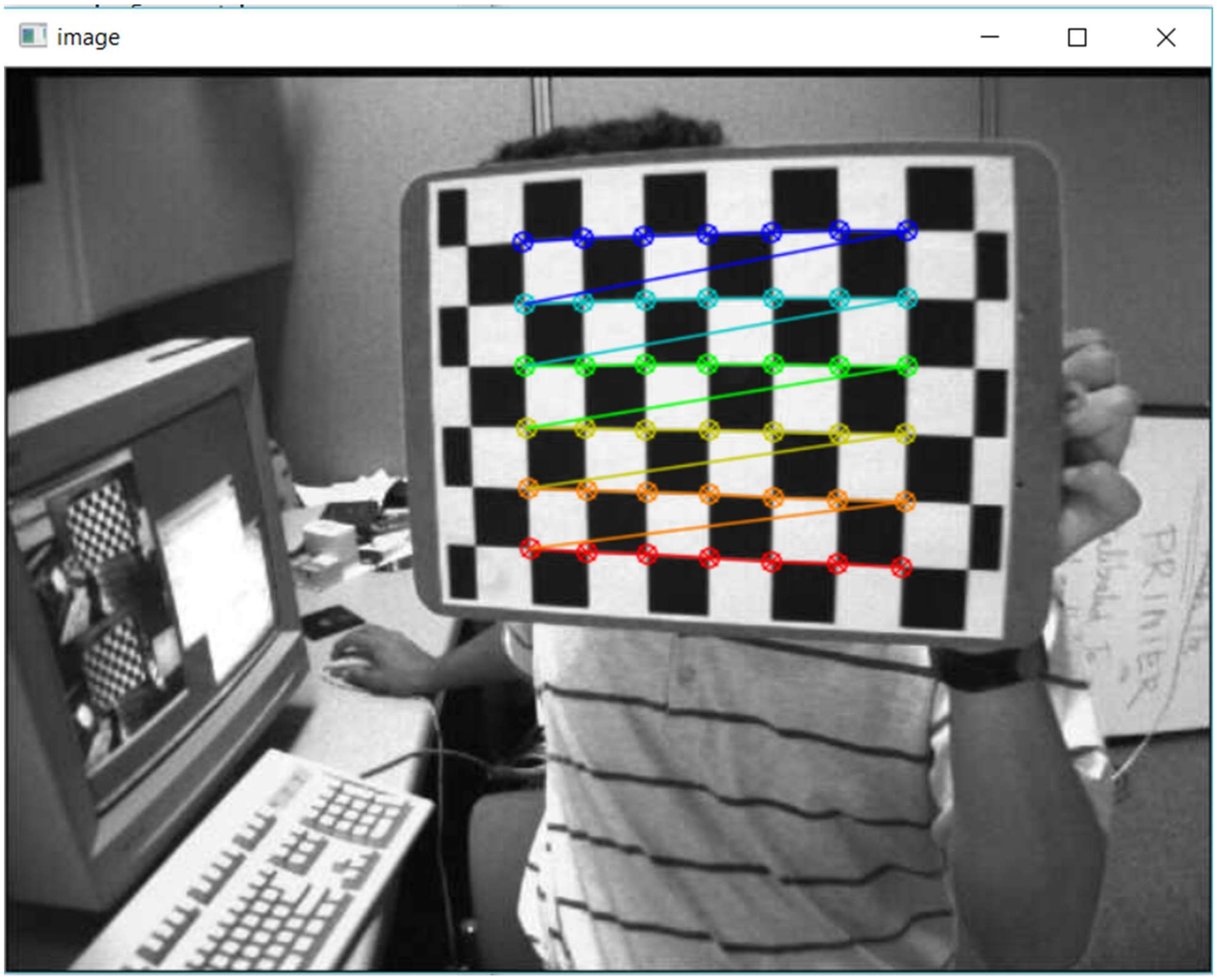
- RANSAC algorithm steps: we must do the next k times
  1. Compute the matrix M with n random points.
  2. Compute the estimated image points using the matrix M.
  3. Compute the distance between the 2D estimated points and the real ones.
  4. Compute the t as 1.5*median of the values in the step 3.
  5. Find all inliers in the data with distances smaller than t.
  6. Recompute matrix M with all the inliers.
  7. Compute MSE for all the points using the matrix M in step 6.

❖ Implement Details

- First, I took the image and then extract the feature points using the method which professor given. I used cv2.findChessboardCorners, cv2.cornerSubPix, drawChessboardCorners method.
- I Used zip () function to iterate two list at same time. This function is very useful.
- In the for loop, I printed world point and image points into file called correspondencePoints.txt.
- When process numpy array as matrix reshape () function may cause more bucket in following data processing.
- Np.concatenate make it easier to combine np array.
- I used RANSAC algorithm for robust estimation as recommended in the assignment.
- I printed Mean Square Error and Intrinsic and Extrinsic parameters.

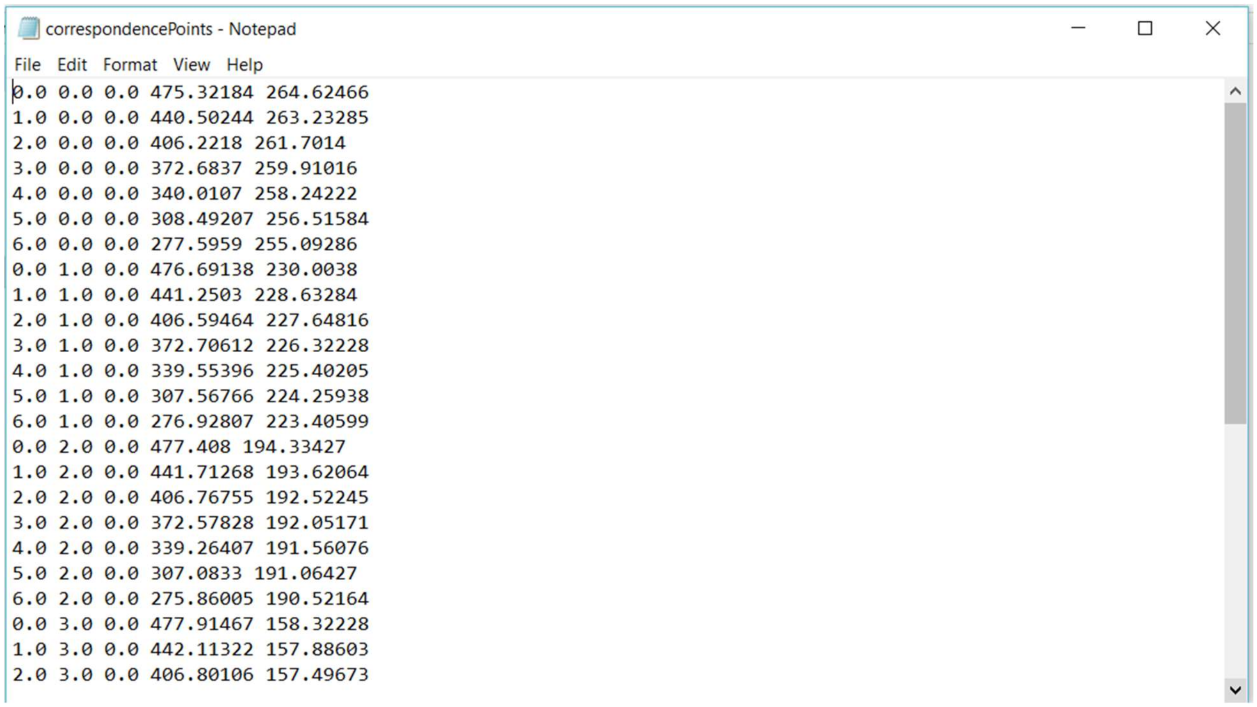
❖ Outputs

First, I took a image of a chess board and extract the feature points from that image.



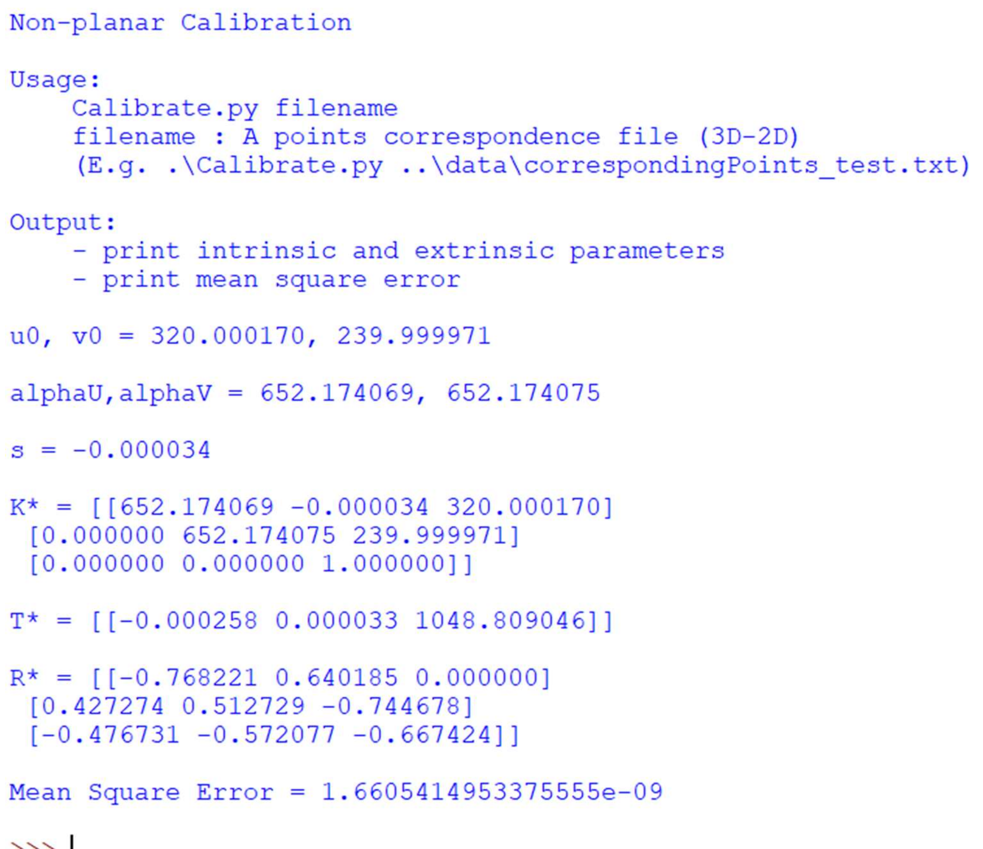
Output:

1. Correspondence point(3D-2D) text file.



```
0.0 0.0 0.0 475.32184 264.62466
1.0 0.0 0.0 440.50244 263.23285
2.0 0.0 0.0 406.2218 261.7014
3.0 0.0 0.0 372.6837 259.91016
4.0 0.0 0.0 340.0107 258.24222
5.0 0.0 0.0 308.49207 256.51584
6.0 0.0 0.0 277.5959 255.09286
0.0 1.0 0.0 476.69138 230.0038
1.0 1.0 0.0 441.2503 228.63284
2.0 1.0 0.0 406.59464 227.64816
3.0 1.0 0.0 372.70612 226.32228
4.0 1.0 0.0 339.55396 225.40205
5.0 1.0 0.0 307.56766 224.25938
6.0 1.0 0.0 276.92807 223.40599
0.0 2.0 0.0 477.408 194.33427
1.0 2.0 0.0 441.71268 193.62064
2.0 2.0 0.0 406.76755 192.52245
3.0 2.0 0.0 372.57828 192.05171
4.0 2.0 0.0 339.26407 191.56076
5.0 2.0 0.0 307.0833 191.06427
6.0 2.0 0.0 275.86005 190.52164
0.0 3.0 0.0 477.91467 158.32228
1.0 3.0 0.0 442.11322 157.88603
2.0 3.0 0.0 406.80106 157.49673
```

2. Input a file with correspondence points(3D-2D)



```
Non-planar Calibration

Usage:
    Calibrate.py filename
    filename : A points correspondence file (3D-2D)
    (E.g. .\Calibrate.py ..\data\correspondingPoints_test.txt)

Output:
    - print intrinsic and extrinsic parameters
    - print mean square error

u0, v0 = 320.000170, 239.999971

alphaU,alphaV = 652.174069, 652.174075

s = -0.000034

K* = [[652.174069 -0.000034 320.000170]
 [0.000000 652.174075 239.999971]
 [0.000000 0.000000 1.000000]]

T* = [[-0.000258 0.000033 1048.809046]]

R* = [[-0.768221 0.640185 0.000000]
 [0.427274 0.512729 -0.744678]
 [-0.476731 -0.572077 -0.667424]]

Mean Square Error = 1.6605414953375555e-09

>>> 
```

3. RANSAC Config

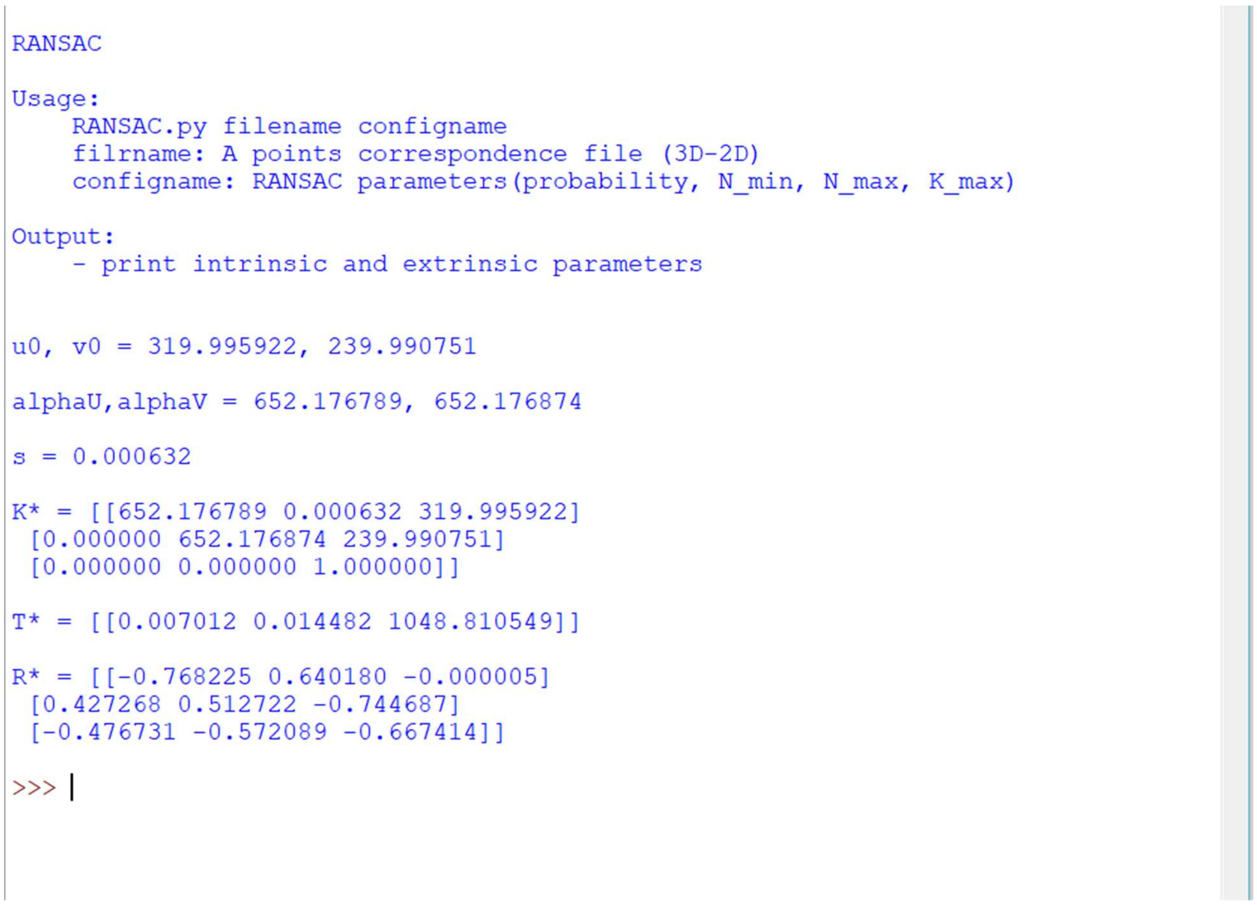
0.999 - probability - Probability that at least one of the draws is free from outliers.

1000- K_max - Maximum number of draws that can be performed.

6 – N_min - Minimum number of points needed to fit model.

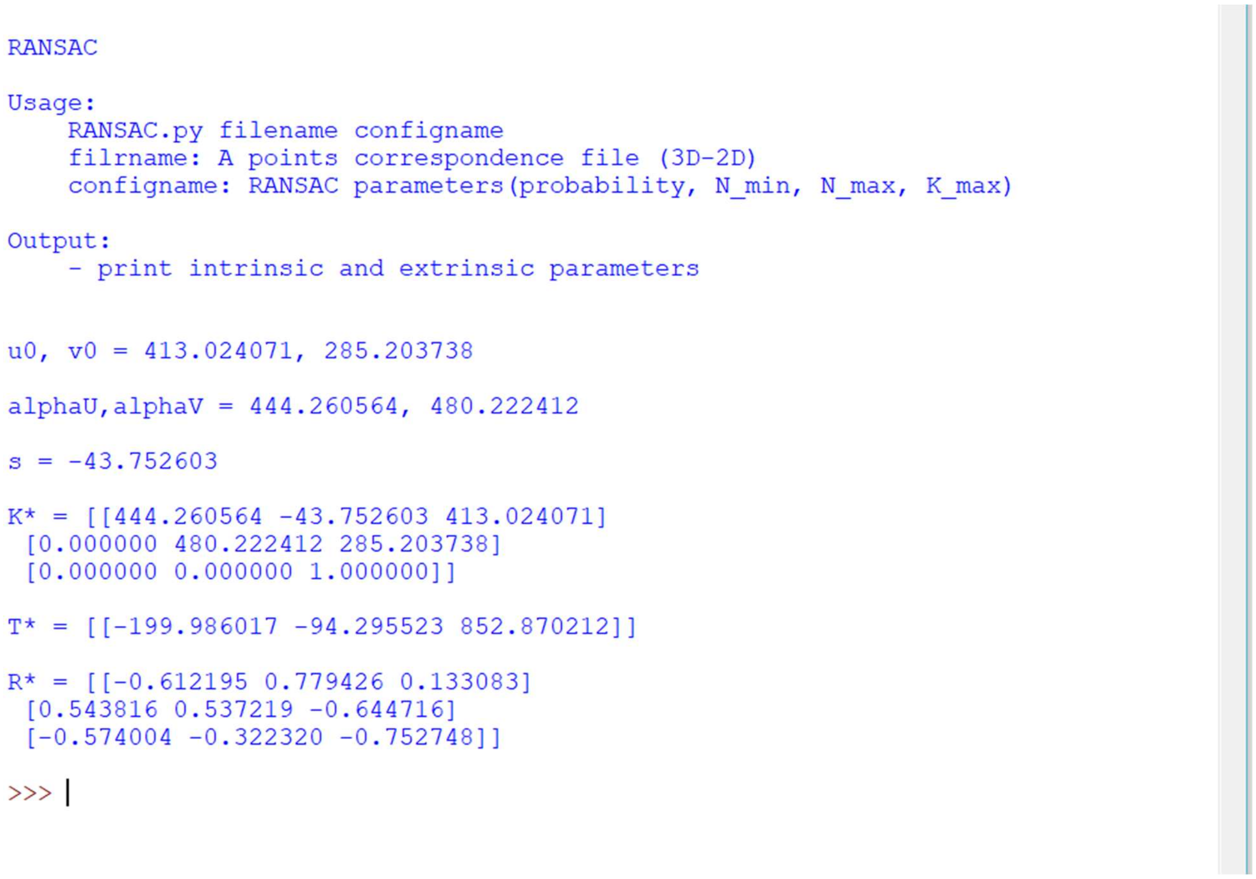
10 – N_max - Maximum number of points needed to fit model.

Output for Noise 1.



```
RANSAC

Usage:
    RANSAC.py filename configname
    filrname: A points correspondence file (3D-2D)
    configname: RANSAC parameters(probability, N_min, N_max, K_max)

Output:
    - print intrinsic and extrinsic parameters


u0, v0 = 319.995922, 239.990751

alphaU,alphaV = 652.176789, 652.176874

s = 0.000632

K* = [[652.176789 0.000632 319.995922]
 [0.000000 652.176874 239.990751]
 [0.000000 0.000000 1.000000]]

T* = [[0.007012 0.014482 1048.810549]]

R* = [[-0.768225 0.640180 -0.000005]
 [0.427268 0.512722 -0.744687]
 [-0.476731 -0.572089 -0.667414]]

>>> |
```

Output for Noise 0



```
RANSAC

Usage:
    RANSAC.py filename configname
    filrname: A points correspondence file (3D-2D)
    configname: RANSAC parameters(probability, N_min, N_max, K_max)

Output:
    - print intrinsic and extrinsic parameters


u0, v0 = 413.024071, 285.203738

alphaU,alphaV = 444.260564, 480.222412

s = -43.752603

K* = [[444.260564 -43.752603 413.024071]
 [0.000000 480.222412 285.203738]
 [0.000000 0.000000 1.000000]]

T* = [[-199.986017 -94.295523 852.870212]]

R* = [[-0.612195 0.779426 0.133083]
 [0.543816 0.537219 -0.644716]
 [-0.574004 -0.322320 -0.752748]]

>>> |
```

RANSAC is not working properly in Noise 0 but it is working good in Noise 1.

❖ Reference

https://docs.opencv.org/3.0-beta/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

https://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html

https://docs.opencv.org/3.0-beta/doc/tutorials/features2d/trackingmotion/corner_subpixeles/corner_subpixeles.html

https://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html