Kinjal Kachi

Course: cs 512

Assignment 4

Anumber: A20449343

Problem Statement:

## Write a program to extract feature points from the calibration target and show them on the image using OpenCV functions.

## Write a program that does the calibration process from a text file with 3D points and its correspondent 2D points. The program should display the intrinsic and extrinsic parameters as well as the mean square error between the known and computed position of the image points.

## Implement the RANSAC algorithm for robust estimation.

Proposed Solution:

**Extract Feature Points from Calibration target**

Referred to Professor lecture materials, instructions given in Q1 of the coding assignment and most of the code is adopted from the OpenCV link: <https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html>

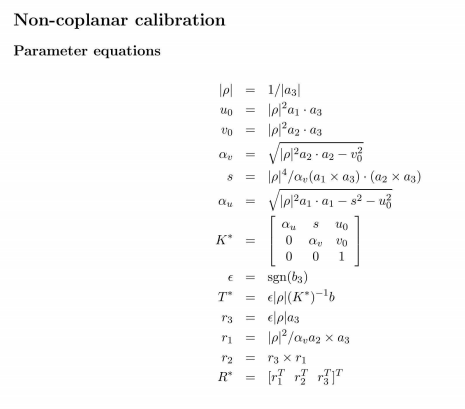
**Calibration Process from a text with 3D points and it’s corresponding 2D points.**

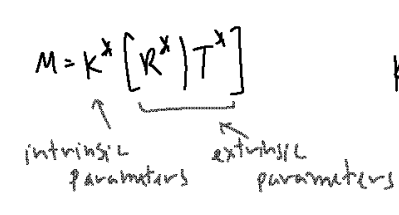
Camera sectioning is the process of estimating the parameters of a pinhole camera model approximating the camera that produced a given photograph or video.

Here we have calculated the following parameters:

Projection Matrix(M):

Projection matrix is a. matrix which describes the mapping of a pinhole camera from 3D points in the world to 2D points in an image. We have **used Singular Value Decomposition** for this purpose. I have calculated the parameters as follows:





Referring to the lecture notes given by professor, K\_star is the intrinsic parameter in the projection matrix and R\_star and T\_star are the extrinsic parameters.

The mean Squared error between known and computed image points, for this convert 3D world co-ordinates to 2DH format by homogenizing it and the subtract it from this 2D image points.

RANSAC algorithm implementation:

Random sample consensus (RANSAC) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. Therefore, it also can be interpreted as an outlier detection method.

RANSAC algorithm implemented is as follows:

Iterate the loop “k” times:

Use a random seed, in the code I used random. Seed(57) and calculate the projection matrix to compute the estimated image points.

Calculate distance between 2D estimated points and the actual points.

Calculate the threshold = 1.5 median

Find all inliers in the data with distances smaller than t and recompute matrix

M with the inliers.

Calculate MSE for all points and name it as Mean\_Squared\_error\_RANSAC.

**Implementation Details:**

To solve Q1 I referred to the OpenCv webpage:

<https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html>

Where in the directions are straight forward:

1. Decide a termination criteria.
2. prepare object points, like (0,0,0), (1,0,0), (2,0,0) ...., (6,5,0)
3. Convert the image to grayscale using function cv2.cvtColor()
4. Finds the positions of internal corners of the chessboard using cv2.findChessboardCorners(grayscale\_image, (7,6), None), input here is the gray scale image created in the previous step.
5. If corners are found then the code will proceed further, corner subpix refines the corner locations.
6. Process of corner position refinement stops either after criteria.maxCount iterations or when the corner position moves by less than criteria.epsilon on some iteration.

Calculation of the Projection Matrix:

I have discussed the parameters that I calculated in the “Proposed Solution” section, to calculate these parameters I have use NumPy library , the np.array() and np.matrix() library to use Numpy arrays and matrices for data storage and calculation.

Np.cross() to yield cross product of two vectors.

np.linalg.norm() function is used to calculate one of the eight different matrix norms or one of the vector norms.

numpy.sign(array [, out]) function is used to indicate the sign of a number element-wise.

For integer inputs, if array value is greater than 0 it returns 1, if array value is less than 0 it returns -1, and if array value 0 it returns 0.

RANSAC:

Randomly select s points (or point pairs) to form a sample

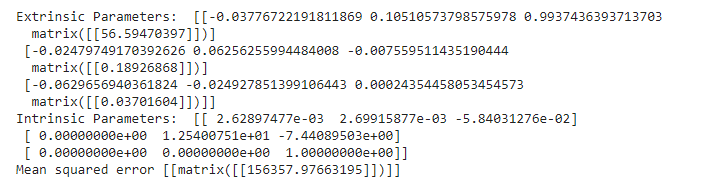
1. Instantiate a model
2. Get consensus set 𝐶𝐶𝑖𝑖
3. If | 𝐶𝐶𝑖𝑖 | > 𝑇𝑇, terminate and return model
4. Repeat for N trials, return model with max | 𝐶𝐶𝑖𝑖 |

As parameter I used n = 9, d = 7, and k = 4. However, this could be wrong in part because of the parameters.

Implementation of Mean Squared Error is same a explained in the proposed solution.

Results:

Calibration Results:



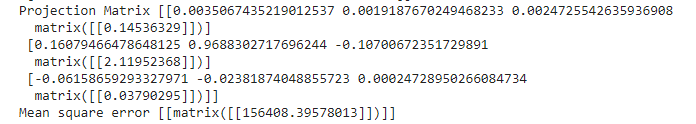
Results from RANSAC:

Projection Matrix from Ransac [[0.0035111720251734665 0.001901051103133921 0.002577898872628641 matrix([[0.14713506]])]]

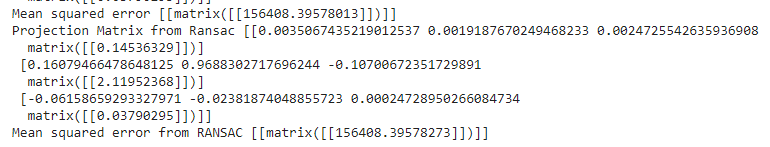
MSE from RANSAC [[matrix([[156357.97662966]])]]

Results for Noisy Data 1:

Calibration Results:



Results from RANSAC



There is no considerable change in results, most of the implementation was done using Professor’s lecture notes.

References:

Professor’s PDF:

<http://www.cs.iit.edu/~agam/cs512/share/calibeq-summary.pdf>

<http://www.cs.iit.edu/~agam/cs512/share/caliba.pdf>

<http://www.cs.iit.edu/~agam/cs512/share/calibb.pdf>

Ransac PDF by professor referred too.

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html>