**cs512 Assignment 4: Report**

**Yash Patel**

**Department of Computer Science**

**Illinois Institute of Technology**

**November 26, 2020**

**Abstract**

This is a report of programming assignment 3 which covers an implementation of camera calibration under the assumption of noisy data. Here, I implemented non-coplanar calibration. Robust estimation through RANSAC is used to eliminate outliers.

1. **Problem statement**

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

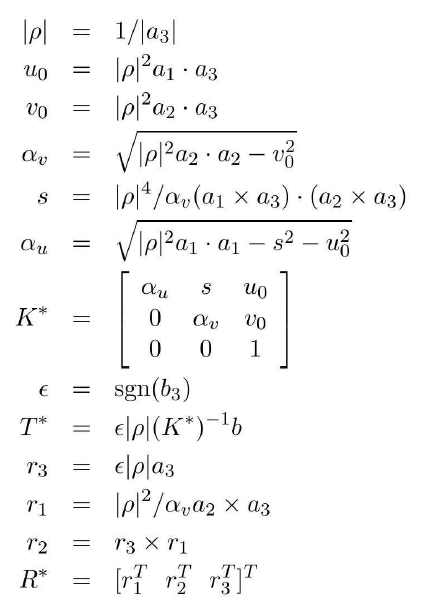
Write a second program to compute camera parameters as described below that uses the point file produced by the first program.

* A point correspondence file is a text file containing in each row a pair of corresponding points (3D-2D) as real numbers separated by space. The first three numbers in each row give x, y, z coordinates of a 3D world point whereas the last 2 numbers give the x, y coordinates of the corresponding 2D image point.
* The program should display the intrinsic and extrinsic parameters of the camera as determined by calibration process. The program should compute and display the mean square error between the known and computed position of the image points. The computed position should be obtained by using the estimated camera parameters to project the 3D points onto the image plane.

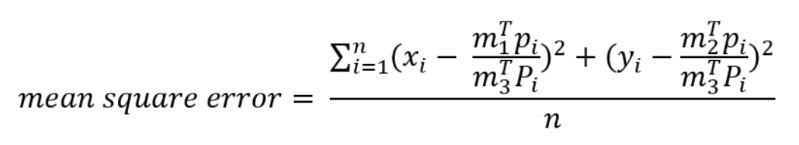
Implement RANSAC algorithm for robust estimation. Your implementation of the RANSAC algorithm should include automatic estimation of the number of draws and of the probability that a data point is an inlier. The final values of these estimates should be displayed by the program. In your estimation of 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’.

1. **Proposed solution**

* **Required packages:**
  1. OpenCV
  2. SciPy
  3. NumPy
* I used non-coplanar camera calibration.
* **Feature extraction:**
  1. Used OpenCV’s findchessboardcorners function to extract corner points in image and then store 3D-2D point pairs in a file.
* **Find camera parameters:**
  1. Read correspondence 3D-2D points from a file.
  2. Find matrix A.
  3. Find matrix M using SVD.
  4. Find camera parameters using below equations.

****

* **Find mean square error:**
  1. Find mean square error using below equation.

****

* **RANSAC algorithm for robust estimation:**
  1. Read correspondence 3D-2D points from a file.
  2. Read ‘RANSAC.config’ file for using parameters for RANSAC algorithm.
  3. Algorithm:
     + Find threshold using median distance.
     + Draw N points randomly(uniform) from correspondence points.
     + Find matrix A and M using random points.
     + Find norm-2 distance on all correspondence points.
     + Find inliers using threshold.
     + If number of inliers are greater than before that recompute models on inlier points and update the best model.
     + Update T, W, K after every iteration.
  4. Compute camera parameters using best model as before.
  5. Compute mean square error using best model as before.

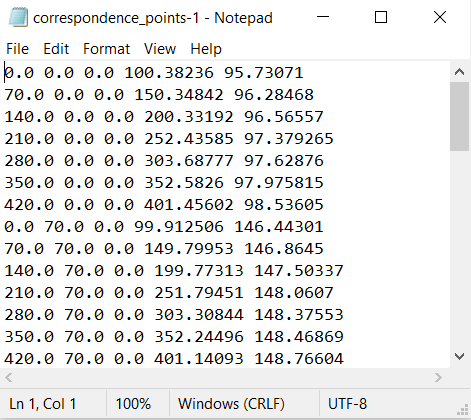
1. **Implementation details**

* **Design issues:**
  + In implementation of RANSAC algorithm one issue I found in recompute a model step. I was confused that how to recompute a model and with which points (random or inliers) should I take to recompute a model.
* **Problems faced and solutions:**
  + Matrix manipulation was one problem that I faced. But NumPy’s documentation was helpful.
  + Compute camera parameters with NumPy array was little tricky. It took little time to understand the use of cross and dot product.
* **Instructions for using program:**
  + Put all files from src and data directories together.
  + Install all required python packages.
  + Execute Homework-4.ipynb file in Jupyter notebook.

1. **Results and Discussion**
   * As I said before in this assignment, I use non-coplanar camera calibration.
   * **Feature extraction:**
     1. Detected corner points as feature points showed on image as below.



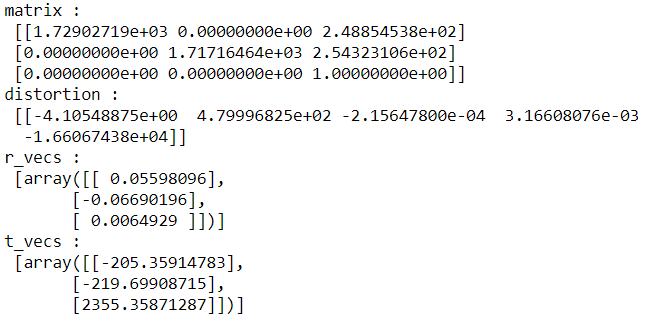
* + 1. extracted corresponding feature points (3D-2D) are saved in file as below.



* + **Find camera parameters:**
    1. Used corresponding 3D-2D points dimensions. Here first dimension is number of views, second is number of points.



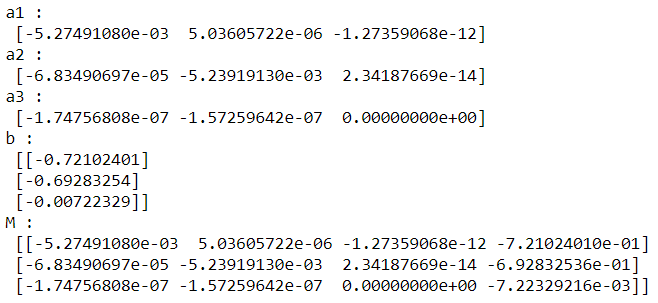
* + 1. Output of OpenCV calibratecamera function.



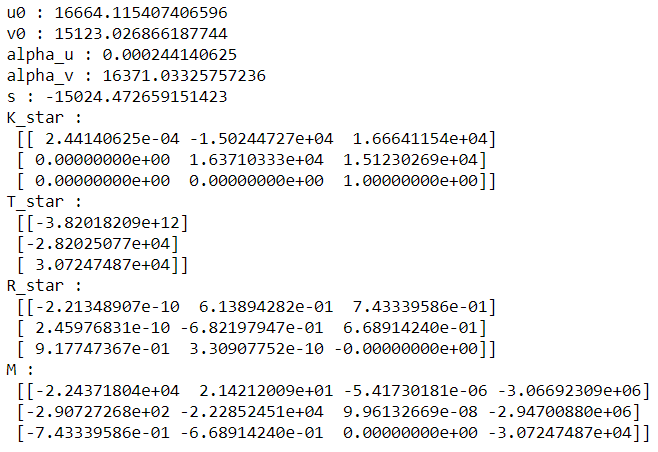
* + 1. Shape of matrix A.



* + 1. Matrix M prime and a1, a2, a3, b



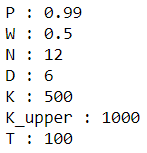
* + 1. Camera parameters and final matrix M:



* + 1. Mean square error of all corresponding points using found camera calibration.

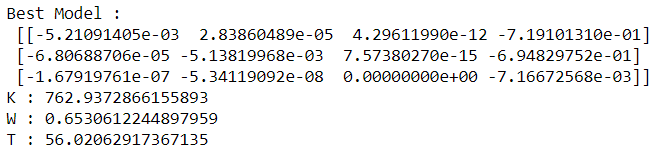


* + **RANSAC algorithm for robust estimation:**
    1. Used parameters:

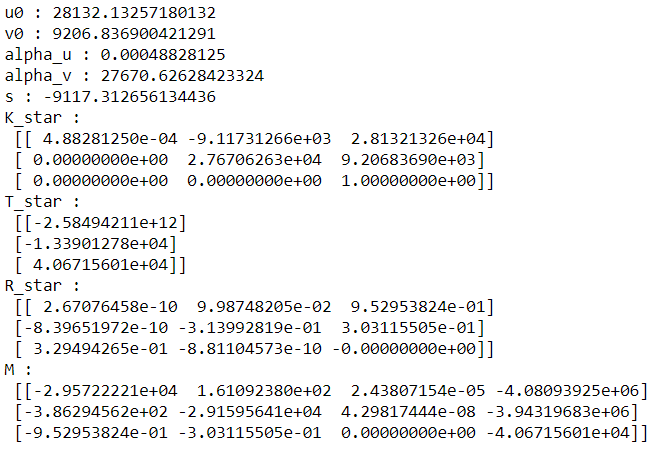
****

Here, I take N=12 means in every iteration algo draws 12 random points. And out of these 12 points at least 6 points should be inliers for good model. Reason behind this is, for estimate camera calibration at least 6 oints are required. So, initially is draw 2 times (12) points. So, my prediction was half the points should be inliers. Number of trails are K=500 and upper bound for k is 1000.

* + 1. Best model (M prime) generated by RANSAC and parameters in last step:

****

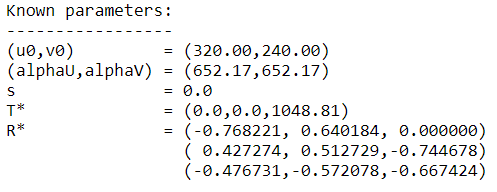
* + 1. Camera parameters and final matrix M:

****

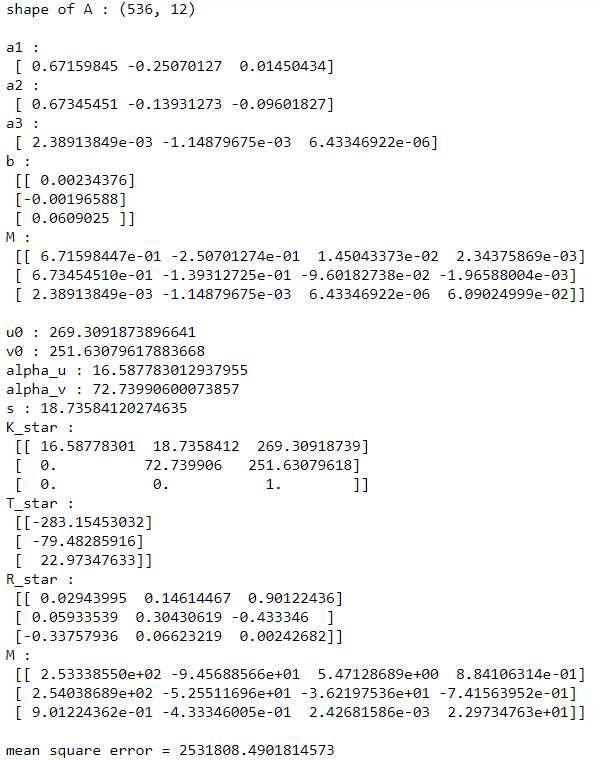
* + 1. Mean square error of RANSAC generated best model:

****

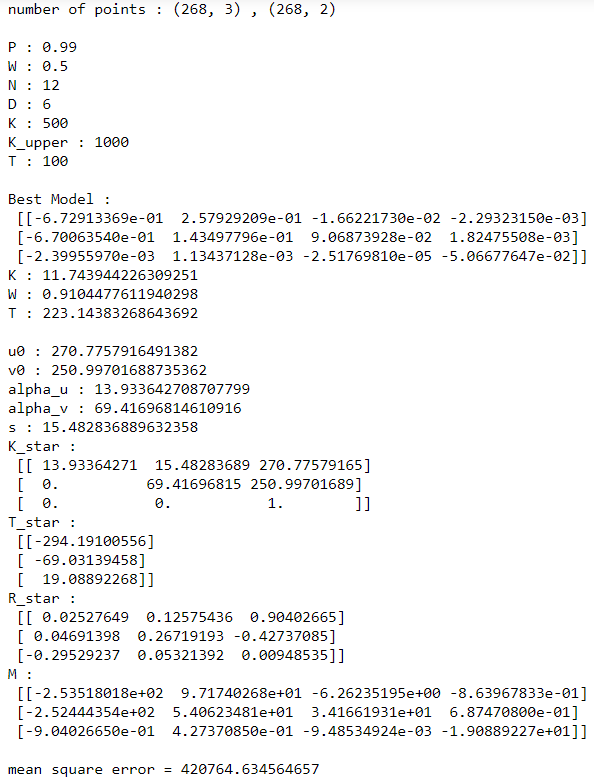
* + **Test of non-coplanar camera calibration on provided data:**
    1. Non-coplanar calibration data:
       1. Actual output:



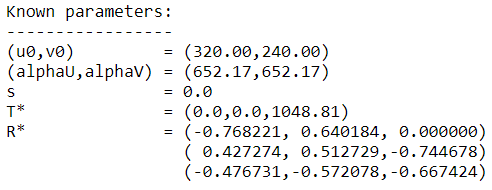
* + - 1. Camera calibration output:



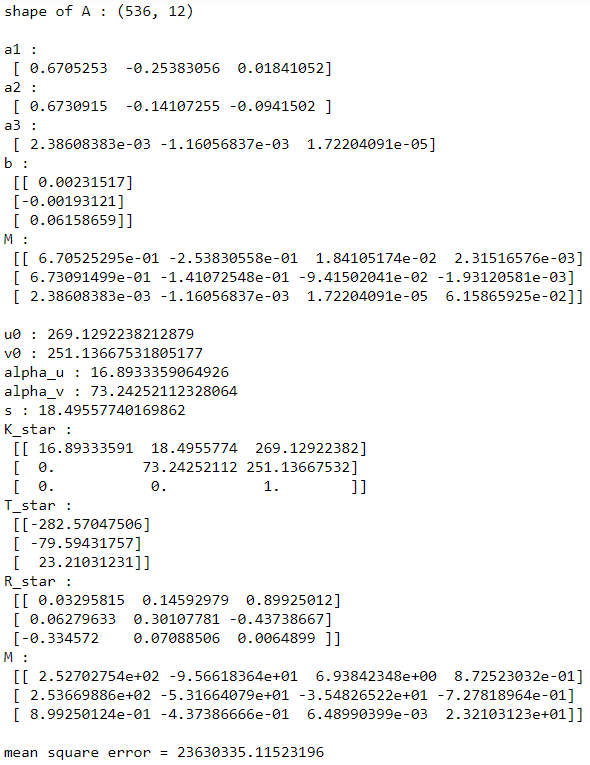
* + - 1. RANSAC robust estimation output:



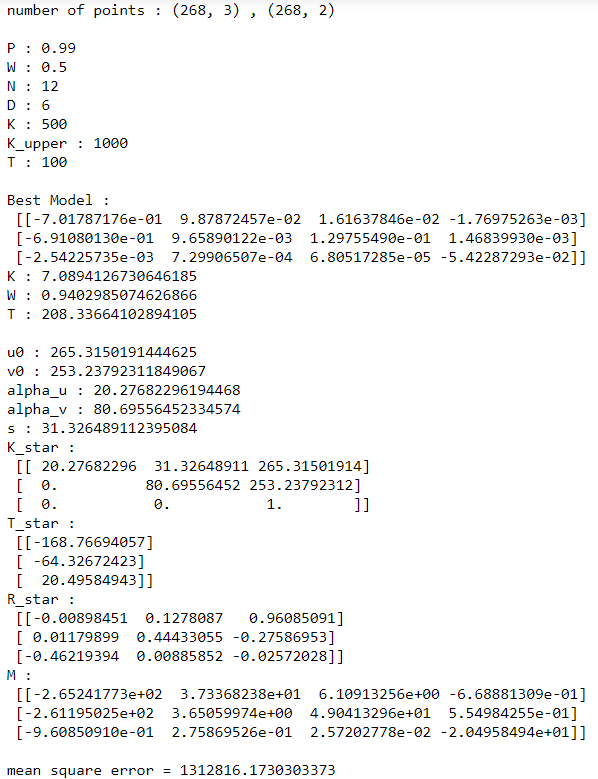
* + 1. Noise version 1:
       1. Actual output:



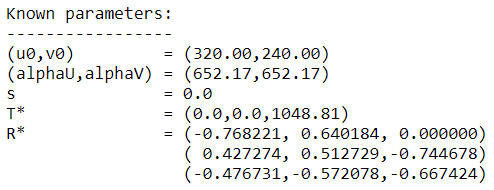
* + - 1. Camera calibration output:



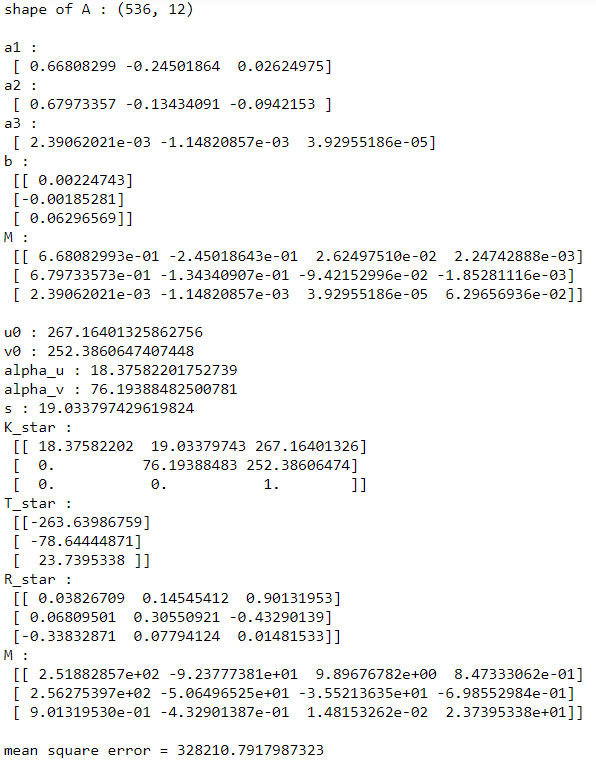
* + - 1. RANSAC robust estimation output:



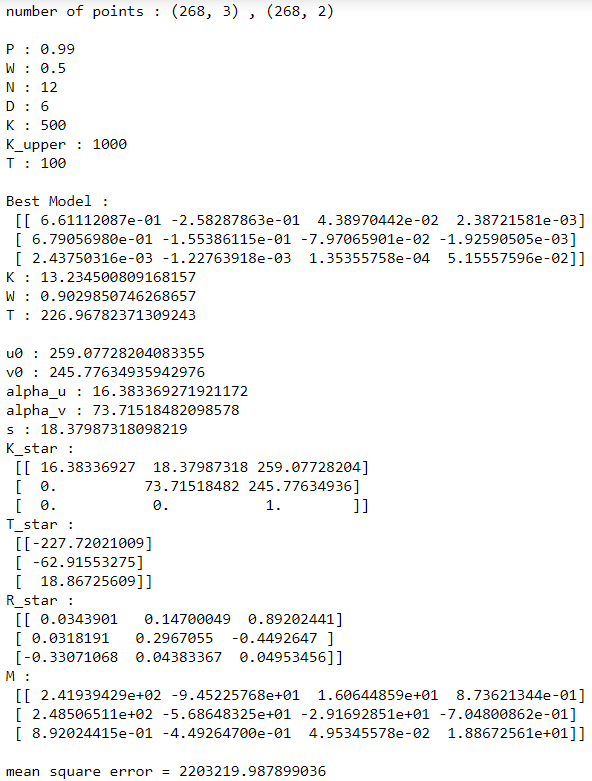
* + 1. Noise version 2:
       1. Actual output:



* + - 1. Camera calibration output:



* + - 1. RANSAC robust estimation output:



When testing the program on noisy data I noticed that, RANSAC didn’t handle well one of the provided cases. Error on non-coplanar data was around 420764, error on noise version 1 was around 1312816 and error on noise version 2 was around 2203219. So, noise version 1 was noisy than non-coplanar data and noise version 2 has more noise than noise version 1. It means noise version 2 has highest noise in provided cases.

Here I got high value of mean square error for noise version 2. Because there are many reasons for that. RANSAC algorithm is random sample consensus and robust to outliers. In this algo every data point gets a vote for being in a model. This would make generated model more robust to outliers. For better model more inliers data points must be agreed. RANSAC is bad when more parameters to tune (here 12 parameters) and cannot be used when there are more outliers than inliers. So, it doesn’t work well when inliers/outliers ratio is higher than 50%. This must be a reason why RANSAC didn’t handle the noise version 2 case.