Camera Calibration
Project 02 - Report

Usama Munir Sheikh
ECSE 6650: Computer Vision with Dr. Qiang Ji, RPI
Troy, New York
sheiku@rpi.edu

***Abstract*—Goal of the Project: Given a set of 2D and 3D points, the intrinsic and extrinsic parameters of two cameras have been linearly estimated using the least squares method. The use of RANSAC as a robust method for camera calibration has also been investigated and implemented and its results compared to the results of the simple linear least squares method.**

Keywords*—*Camera Calibration; Extrinsic Parameters, Intrinsic Parameters, Linear Least Squares, Projection Matrix, RANSAC.

I. Introduction

Cameras are the most important sensors in the field of computer vision and therefore, it is important to understand how a camera can be accurately calibrated.

The goal of this project is to implement camera calibration. The purpose of camera calibration is to estimate the extrinsic and intrinsic parameters of a camera including the skew angle and the lens distortion coefficient(s).

Camera calibration must be performed accurately and robustly, since the estimated parameters are of great use in the field of vision, particularly in 3D reconstruction and recognition. Other applications include 'Dense Reconstruction', 'Visual Inspection', 'Object Localization' and 'Camera Localization.'[4]

There are multiple ways to calibrate a camera. The conventional method uses a calibration pattern that consists of 2D and 3D data to compute camera parameters. Generally, only one view of the image is needed[1] for this method and it is the method we employ here.

Another method is camera self-calibration that only needs the image data for camera calibration, but from multiple views.[1]

For this project, only the intrinsic and extrinsic camera parameters have been estimated, with a linear method, using pairs of corresponding 2D and 3D points.

A. *Intrinsic and Extrinsic Camera Parameters*

The intrinsic camera parameters are encapsulated in the matrix, W, given below:

$$W = \begin{pmatrix} S_x f & cot(\alpha) & c_0 \\ 0 & \dfrac{S_y f}{sin(\alpha)} & r_0 \\ 0 & 0 & 1 \end{pmatrix} \quad (1)$$

where α is the skew angle, (c_0, r_0) is the principle point and $S_x f$ and $S_y f$ are the x and y sampling frequencies times the focal-length(f) respectively.

If we ignore the skew parameters, the W matrix becomes:

$$W = \begin{pmatrix} S_x f & 0 & c_0 \\ 0 & S_y f & r_0 \\ 0 & 0 & 1 \end{pmatrix} \quad (2)$$

The extrinsic parameters of the camera are given by the M matrix which is made up of the rotaion matrix and the translation vector:

$$M = [R\ T] = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} = \begin{pmatrix} r_1 & t_x \\ r_2 & t_y \\ r_3 & t_z \end{pmatrix} \quad (3)$$

where the rows of the rotation matrix, R, have been clumped together into row vectors, r_1, r_2 and r_3 .

B. *Full Perspective Projection and the Projection Matrix*

The projection matrix, P, is the relation between 3D points in the world/object frame and the 2D points in the row-column frame.

Let $(c, r)^t$ be the image coordinates in the row-column frame and $(x, y, z)^t$ be the corresponding 3D coordinates in the world frame, then the relationship between the two in homogeneous coordinates is called the perspective equation. In general, it can be written as:

$$\lambda \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = P \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (4)$$

$$\lambda \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \quad (5)$$

We can also write the P matrix as the following, where we have represented the first 3 entries of each row collectively as a row vector. This will prove useful when we solve for the P matrix using our linear method for camera calibration:

$$P = \begin{pmatrix} p_1 & p_{14} \\ p_2 & p_{24} \\ p_3 & p_{34} \end{pmatrix} \tag{6}$$

For the Full Perspective Projection Model, the matrix, P in terms of the intrinsic and the extrinsic camera parameters can be written as a function of W and M:

$$P = WM = \begin{pmatrix} S_x f r_1 + c_0 r_3 & s_x f t_x + c_0 t_z \\ S_y f r_2 + r_0 r_3 & s_y f t_y + r_0 t_z \\ r_3 & t_z \end{pmatrix} \tag{7}$$

II. Problem Statement and Methodology

To find the camera parameters we must (i) determine the Projection Matrix P and then (ii) derive Camera Parameters from P.

A. Determining P: Theory of the Linear Method

Given image points $m_i^{2\times1} = (c_i, r_i)^t$ and the corresponding 3D points $M_i^{3\times1} = (x_i, y_i, z_i)^t$ where $i = 1, ..., N$, we want to compute P.

Then for each pair of 2D-3D points, we have:

$$M_i^t p_2 + p_{14} - c_i M_i^t p_3 - c_i p_{34} = 0 \tag{8}$$

$$M_i^t p_2 + p_{24} - r_i M_i^t p_3 - r_i p_{34} = 0 \tag{9}$$

For N points, we can setup a system of linear equations:

$$AV = 0 \tag{10}$$

where A is a $2N \times 12$ matrix depending only on the 3-D and 2-D coordinates of the calibration points, and V is a 12×1 vector: $(p_1^t \; p_{14} \; p_2^t \; p_{24} \; p_3^t \; p_{34})^t$.

$$A = \begin{pmatrix} M_1^t & 1 & \hat{0} & 0 & -c_1 M_1^t & -c1 \\ \hat{0} & 0 & M_1^t & 1 & -r_1 M_1^t & -r1 \\ & & \cdot & & & \\ & & \cdot & & & \\ & & \cdot & & & \\ M_N^t & 1 & \hat{0} & 0 & -c_N M_N^t & -cN \\ \hat{0} & 0 & M_N^t & 1 & -r_N M_N^t & -rN \end{pmatrix} \tag{11}$$

where $\hat{0}^{1\times3} = [0\ 0\ 0]$

For the linear method to work, we need $N >= 6$ points and the N points cannot be coplanar.

In general, the rank of matrix A is 11 (for 12 unknowns), which means the solution is up-to a scale factor. But due to the effect of noise, mismatched points and location errors, matrix A may be full rank, which may make the solution unique, specifically, it'll correspond to the eigen-vector of the smallest eigen-value.[1]

"The rank of A may also change for certain special configurations of the input 3D points, for example collinear points, coplanar points, etc. If the 3D points are coplanar, then the rank(A) < 11 (in fact, it equals 8), which means there is an infinite number of solutions".[1]

To estimate the projection matrix, P, we must solve for V. In this project, I have used Linear Method 2, described in [1] to solve for P.

To start, let $A = [B\ b]$ where,

$$B = \begin{pmatrix} M_1^t & 1 & \hat{0} & 0 & -c_1 M_1^t \\ \hat{0} & 0 & M_1^t & 1 & -r_1 M_1^t \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ M_N^t & 1 & \hat{0} & 0 & -c_N M_N^t \\ \hat{0} & 0 & M_N^t & 1 & -r_N M_N^t \end{pmatrix} \tag{12}$$

$$b = (-c_1 \; -r_1 \; ... \; -c_N \; -r_N)^t \tag{13}$$

Also,

$$V = p_{34} \begin{pmatrix} Y \\ 1 \end{pmatrix} \tag{14}$$

where,

$$Y = (p_1^t \; p_{14} \; p_2^t \; p_{24} \; p_3^t)^t / p_{34}. \tag{15}$$

Then, $AV = p_{34}(BY + b)$. Since p_{34} is a constant, minimizing $||AV||^2$ is the same as minimizing $||BY + b||^2$, whose solution is:

$$Y = -(B^t B)^{-1} B^t b \tag{16}$$

The rank of matrix B must be 11.

However, the solution of Y is up to a scale factor p_{34}. To recover the scale factor, we can use the fact that $||p3|| = 1$.[1] This is because of the special property of the rotation matrix. Thus, the scale factor p_{34} can be recovered as

$$p34 = \frac{1}{\sqrt{Y^2(9) + Y^2(10) + Y^2(11))}} \tag{17}$$

We can now use Y to get the final projection matrix (vector), V:

$$V = p_{34} \begin{pmatrix} Y \\ 1 \end{pmatrix} \tag{18}$$

B. Deriving Camera Parameters from P

Now that we have determined P, the camera parameters can be computed by comparing equation (6) to equation (7).

$$r_3 = p_3 \tag{19}$$

$$t_z = p_{34} \tag{20}$$

$$c_0 = p_1^t p_3 \tag{21}$$

$$r_0 = p_2^t p_3 \tag{22}$$

$$S_x f = \sqrt{p_1^t p_1 - c_0^2} \tag{23}$$

$$S_y f = \sqrt{p_2^t p_2 - r_0^2} \tag{24}$$

$$t_x = \frac{p_{14} - c_0 t_z}{S_x f} \tag{25}$$

$$t_y = \frac{p_{24} - r_0 t_z}{S_y f} \tag{26}$$

$$r_1 = \frac{p_1 - c_0 r_3}{S_x f} \tag{27}$$

$$r_2 = \frac{p_2 - r_0 r_3}{S_y f} \tag{28}$$

C. Computing the Projection Matrix Using RANSAC

Alas, the linear LSQ method to compute P (discussed earlier) is sensitive to image errors and outliers. In the real world conditions, the corresponding 2D image data may be acquired by some feature extraction methods which may give rise to outliers. And using only the LSQ camera calibration algorithm, by itself, will corrupt the results due to the presence of outliers that result from mismatching and occlusion.[2]

There are many methods in computer vision literature to improve the robustness and accuracy of camera calibration methods. In this project, the RANSAC (Random Sample Consensus) method is used to robustify the model fitting by selecting a minimum sample set required for the model among the remaining samples. Thus, projection "models containing outliers are rejected since they do not generate sufficient consensus".[2] The assumptions of RANSAC are (i) The model can be estimated from K data items and (ii) There are $N(N > K)$ data items in total.[2]

The first step in RANSAC is to determine the minimum number of subsets, of the data, needed to produce the correct camera calibration. "Assuming that the whole set of data may contain up to a fraction ϵ of outliers, then the probability that all K data points in a subset are good is $(1-\epsilon)^K$, and the probability that all s different subsets will contain at least one or more outliers is $(1-(1-\epsilon)^K)^s$".[2] So the probability that at least one random subset has no outliers is given by

$$prob = (1 - (1-\epsilon)^K)^s \tag{29}$$

Thus the number of iterations (subsets) needed is computed as,

$$s = \frac{ln(1 - prob)}{ln(1 - (1-\epsilon)^K)} \tag{30}$$

The RANSAC algorithm works in the following steps: These steps are copied from [2]

Step 1) Randomly pick a subset of $K(K >= 6)$ pairs of points from $N(N > K)$ pairs, compute the projection matrix P using these points.
Step 2) Using the P matrix, compute the projection error (Euclidean Distance) for all of the remaining points. If it is within a threshold distance, increment a counter of the number of the inliers.
Step 3) Repeat Steps one and two, s number of times and select the subset of points with the largest number of inliers, which agree with the hypothesized P.
Step 4) Using all of the inliers to recompute the best P.

III. Experiments and Results

In this section a discussion will be carried out on the collection of data points and the results obtained from running both algorithms will be presented.

A. Experimental Data Used and How it was Produced

For this project, we were given two sets of 3D data points $(x, y, z)^t$. Each set had 72 points in total. One set corresponded to good 3D points and the other bad. The good set of 3D data points had no mismatches with the corresponding 2D data provided, where as the bad set had about 18% mismatched points.

Also, there were two sets of 2D data called 'left' and 'right' respectively, each for a different camera. The two 2D images with the 2D data sets are shown in figure 1 and 2. The 2D points were plotted, over the two images provided, in MATLAB.

The two images used are those of a checkerboard. Checkerboard patterns (Tsai Grids) are routinely used for camera calibration in computer vision applications. This is because the dimensions of a checkerboard and the sizes of the squares are known. This provides us with the 3D information needed to calibrate a camera.

Also, the corresponding 2D points on a checkerboard can be easily recognized using image processing techniques like the Canny Edge Detection. Straight line fitting can be applied to detected linked edges and intersecting the lines gives image corners. Then we can match the image corners to the 3D target checkerboard corners by counting if the whole target is visible in the image.[3]

It is also plausible that the 3D data provided to us is made up by the professor using the 2D data and some made-up (or manufacturer specified) intrinsic and extrinsic camera parameters. Also note, that the 2D image may have been rectified of distortion since we are neglecting that in the project. Another possibility is the use of stereo 3D reconstruction techniques using two cameras to get the 3D points.

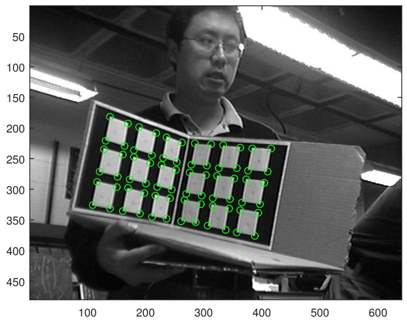

Fig.1. Right Image

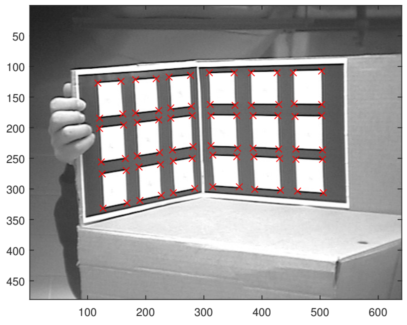

Fig.2. Left Image

B. Estimated Camera Parameters

1) Camera 1 (Right Camera) - Linear Only: The estimated camera parameters for the 'right' camera, using the linear least squares method only, are given by the following intrinsic and extrinsic matrices:

$$W_{linear}^{good} = \begin{pmatrix} 836.74 & 0 & 317.41 \\ 0 & 803.66 & 325.94 \\ 0 & 0 & 1 \end{pmatrix} \tag{31}$$

$$M_{linear}^{good} = \begin{pmatrix} -0.6179 & 0.7636 & 0.1874 & -117.7 \\ -0.1418 & 0.1276 & -0.9816 & -63.8 \\ -0.7735 & -0.6331 & 0.0295 & 1356.9 \end{pmatrix} \tag{32}$$

The mean square re-projection error using the resulting P matrix is 1.2602 pixels.

$$W_{linear}^{bad} = \begin{pmatrix} 12.18 & 0 & 251.51 \\ 0 & 26.02 & 305.69 \\ 0 & 0 & 1 \end{pmatrix} \tag{33}$$

$$M_{linear}^{bad} = \begin{pmatrix} -0.7500 & 0.5475 & 0.3711 & -20.7.0 \\ 0.2517 & 0.2837 & -0.9253 & -58.0 \\ -0.6606 & -0.6484 & -0.3784 & 208.9 \end{pmatrix} \tag{34}$$

The mean square re-projection error using the resulting P matrix is 584.6371 pixels.

2) Camera 1 (Right Camera) - RANSAC: Using RANSAC with $s = 5000$ subsets and the threshold $= 5$ pixels, we get the following results for the 'right' camera:

$$W_{RNSC}^{good} = \begin{pmatrix} 836.74 & 0 & 317.41 \\ 0 & 803.66 & 325.94 \\ 0 & 0 & 1 \end{pmatrix} \tag{35}$$

$$M_{RNSC}^{good} = \begin{pmatrix} -0.6179 & 0.7636 & 0.1874 & -117.67 \\ -0.1418 & 0.1276 & -0.9816 & -63.77 \\ 0.7735 & 0.6331 & -0.0295 & 1356.9 \end{pmatrix} \tag{36}$$

The mean square re-projection error using the resulting P matrix is 1.2602 pixels.

$$W_{RNSC}^{bad} = \begin{pmatrix} 869.38 & 0 & 321.63 \\ 0 & 834.96 & 336.83 \\ 0 & 0 & 1 \end{pmatrix} \tag{37}$$

$$M_{RNSC}^{bad} = \begin{pmatrix} -0.6153 & 0.7662 & 0.1854 & -123.94 \\ -0.1340 & 0.1326 & -0.9821 & 45.93 \\ -0.7770 & -0.6291 & 0.0211 & 1406.4 \end{pmatrix} \tag{38}$$

The mean square re-projection error using the resulting P matrix is 100.9705 pixels.

3) Camera 2 (Left Camera) - Linear Only: For the second or the 'left' camera the camera parameters are given by the following intrinsic and extrinsic camera matrices:

$$W_{linear}^{good} = \begin{pmatrix} 1637.9 & 0 & 389.95 \\ 0 & 1599.9 & 250.15 \\ 0 & 0 & 1 \end{pmatrix} \tag{39}$$

$$M_{linear}^{good} = \begin{pmatrix} -0.5722 & 0.8194 & -0.0328 & -104.8 \\ 0.1109 & 0.0347 & -0.9932 & 76.3 \\ -0.8127 & -0.5720 & -0.1107 & 1884.7 \end{pmatrix} \tag{40}$$

The mean square re-projection error using the resulting P matrix is 1.6636 pixels.

$$W_{linear}^{bad} = \begin{pmatrix} 15.20 & 0 & 294.08 \\ 0 & 39.13 & 239.33 \\ 0 & 0 & 1 \end{pmatrix} \tag{41}$$

$$M_{linear}^{bad} = \begin{pmatrix} -0.6294 & 0.7538 & -0.1886 & 15.9 \\ 0.3242 & 0.2183 & -0.9205 & 58.2 \\ -0.6586 & -0.6463 & -0.3853 & 209.2 \end{pmatrix} \tag{42}$$

The mean square re-projection error using the resulting P matrix is 518.2823 pixels.

4) Camera 2 (Left Camera) - RANSAC: Using RANSAC with $s = 5000$ subsets and the threshold $= 5$ pixels, we get the following for the left camera:

$$W_{RNSC}^{good} = \begin{pmatrix} 1637.9 & 0 & 389.94 \\ 0 & 1599.9 & 250.14 \\ 0 & 0 & 1 \end{pmatrix} \tag{43}$$

$$M_{RNSC}^{good} = \begin{pmatrix} -0.5722 & 0.8194 & -0.0328 & -104.79 \\ 0.1109 & 0.0347 & -0.9932 & 76.34 \\ -0.8127 & -0.5720 & -0.1107 & 1884.7e \end{pmatrix} \tag{44}$$

The mean square re-projection error using the resulting P matrix is 1.6636 pixels.

$$W^{bad}_{RNSC} = \begin{pmatrix} 1652.9 & 0 & 436.08 \\ 0 & 1617.7 & 251.20 \\ 0 & 0 & 1 \end{pmatrix} \tag{45}$$

$$M^{bad}_{RNSC} = \begin{pmatrix} -0.5462 & 0.8371 & -0.0290 & -158.34 \\ 0.1114 & 0.0346 & -0.9932 & 75.15 \\ -0.8304 & -0.5457 & -0.1122 & 1902.6 \end{pmatrix} \tag{46}$$

The mean square re-projection error using the resulting P matrix is 136.8549 pixels.

IV. Result Discussion

A. Comparison: Linear LSQ VS RANSAC

By looking at the results we can draw some obvious conclusions between each method.

When it comes to the good set of 3D points we see that the results for both, Linear Least Squares and RANSAC are exactly the same. Also, the mean square re-projection errors are also the same and really low: 1.2 pixels. This is true for both cameras. Therefore, we can conclude that RANSAC is at least as good as Linear Least Squares when there are no mismatched points or noise in the 3D data. And that they both work really well with exact data.

However, the results differ significantly when it comes to the bad set of 3D points, that are 18% mismatched. In real world situations, there is bound to be error in measurements and an 18% error is not particularly that high when it comes to matching the exact 3D points to corresponding 2D points. In this case, we see that RANSAC clearly outperforms the Linear Least Squares method by a great margin.

Consider the results for Camera 1, for the bad points, given in equations $33, 34, 37$ and 38. We see the values for S_xf and S_yf are 12.18 and 26.02 respectively where as the actual values are 836.74 and 803.66. The error for these values is 98.54% and 96.76% respectively. This is really high. Where as, the results for S_xf and S_yf using RANSAC are 869.38 and 834.96 respectively. This gives corresponding errors of 3.75% and 31.3% only. Similar results are seen for all the other parameters, both intrinsic and extrinsic. This is true for camera 2 as well.

The mean square re-projections errors using Linear LSQ and RANSAC, for the bad set of points, differ a lot. While the re-projection error for Linear LSQ only method is $500 pixels$, it is only around 100 pixels for RANSAC. Do note, however, that the accuracy of RANSAC may very well be increased by increasing the number of subsets, s, in equation 30 or controlling the threshold distance.

We can therefore conclude that RANSAC is a great method to reject noise and disturbance within the data and therefore increase robustness of the camera calibration process.

B. Methods to Validate Calibration Results

We can verify and validate our results for camera calibration in multiple ways:

1. We can choose multiple random and arbitrary 3D points that were not part of our data and project them using the P matrix we got from our calibration results. We can then compare the 2D coordinates obtained as a result, with the actual corresponding 2D coordinates of those points. If the results are within an acceptable error margin, we know that our calibration results are accurate. If, however, there is significant error, we can try a different data set or try and find problems in our algorithm.

2. We can also create an arbitrary set of data using an assumed (known) projection, P, and perform camera calibration using our algorithm. If the results match to the known matrix P, we know that our algorithm is working fine.

3. To see if our results are not completely off, we can use a reliable calibration method, either through an on-line application or through a computer vision toolbox or a previously established method to calibrate the camera and then use our algorithm to see if the results are at least similar. This method is similar to method 2, above, it was useful to see if our RANSAC algorithm was working correctly by seeing it's result for the good set of 3D points. Since we already had an accurate set of results for the good set using Linear LSQ, this helped out in debugging code for RANSAC until it gave similar results.

V. Conclusion and Summary

To summarize, I have implemented two, linear, camera calibration methods to find the intrinsic and extrinsic camera parameters. Given sets of 2D and 3D points, I have applied the Linear Least Squares Method and RANSAC to calibrate two cameras.

From the results, I found out that using linear LSQ only to calibrate a camera will not work in a real world situation because data collection methods always have some inherent noise and error that affects the results of this method quite adversely. However, using RANSAC, in a similar situation, coupled with the linear LSQ method, gives much more robust results.

I have learnt about the robustness of the RANSAC algorithm and how it rejects noise/errors/disturbances in the data. Also, I have learnt to validate my results, which helps to debug code and ascertain the correctness of my implementation of the algorithm. I have solidified my concepts on linear camera calibration. Finally, I have learnt how to use LATEX.

I encountered a lot of frustrating difficulties during the course of the project. It was hard to debug my code for the RANSAC algorithm since it worked fine for the good set of data but gave really bad results for the bad data. I had to debug my code, line-by-line until I figured out the small mistake I was making. Even then, the problem wasn't clear until I ran the algorithm for $100,000$ subsets and noticed the time taken was much longer than expected. This gave me an idea to test the loop that found least squares solution in every iteration because this part of the algorithm is the only step that takes longer than the others. There, I found out that the intermediate B and b matrices weren't resetting after every iteration of the big loop. After correcting this error, the algorithm worked fine.

This project was quite tough and implementing the RANSAC algorithm was a great learning experience. I would have liked to include lens distortion in the project as well, perhaps as an extra-credit assignment. For future work, I will still look into the estimation of the lens-distortion coefficient(s), even though the project did not require it and incorporate it into the algorithm.

References

[1] Ji, Qiang. ”ECSE 6650 Lecture Notes.” N.p., n.d. Web. 17 Mar. 2016. <https://www.ecse.rpi.edu/Homepages/qji/CV/ecse6650_lecture_notes.html>

[2] Ji, Qiang. ”ECSE 6650 Project Description.” N.p., n.d. Email. 03 Mar. 2016. <calibration_ransac.doc>

[3] ”Tsai Camera Calibration.” Tsai Camera Calibration. N.p., n.d. Web. 20 Mar. 2016. <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/DIAS1/ >

[4] Salvi, Joaquim, Xavier Armangu, and Joan Batlle. ”A comparative review of camera calibrating methods with accuracy evaluation.” Pattern recognition 35.7 (2002): 1617-1635.