# Camera Calibration: Linear Method vs. RANSAC

Hayleigh Sanders (RIN#661195735)

## 1. Introduction

Camera calibration is the estimation of the intrinsic and extrinsic parameters of a camera model. It is important that these parameters are computed accurately because computer vision algorithms depend on accurate camera parameters to translate between 2D and 3D data. With real world data, errors in calibration points may be prevalent and therefore calibration methods must employ some methods to be robust against them.

Given 72 pairs of coordinates corresponding to the corner points of a calibration image, with 18% of the 3D points containing mismatches, we want to compare the performance of the linear method and RANSAC-augmented methods in producing a P matrix which can accurately project 2D data.

## 2. Theories for the algorithm

### 2.1 Full perspective projection model

In this project, camera calibration will be performed under full perspective projection. A point in 3D space in the object frame, $X = (x,y,z)^t$ is related to its corresponding 2D coordinates in the image frame $U = (u,v)^t$ and 2D coordinates in the row-column frame $p = (c,r)^t$ by the full perspective projection model:

$$\lambda\begin{pmatrix} c \\ r \end{pmatrix} = \begin{bmatrix} p11 & p12 & p13 & p14 \\ p21 & p22 & p23 & p24 \\ p31 & p32 & p33 & p34 \end{bmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad (1)$$

Here, $\lambda$ is a scalar and the projection matrix *P = [p11, p12...p34]* can be expressed as:

$$P = WM \qquad (2)$$

Where W is the intrinsic matrix, given by:

$$W = \begin{pmatrix} f s_x & 0 & c_0 \\ 0 & f s_y & r_0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (3)$$

Where $f$ is the focal length, $(c_0,r_0)$ are the coordinates of the principal point, $s_x$ and $s_y$ are the scale factors (pixels/mm) due to spatial quantization.

M is the extrinsic matrix, given by:

$$M = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \qquad (4)$$

Where $R = [r_{11}, r_{12} ... r_{33}]$ is the rotation matrix and $T = [t_x, t_y, t_z]$ is the translation matrix. From equations (2), (3) and (4) it becomes clear that P can be expressed as a function of the intrinsic and extrinsic camera parameters. Thus the full perspective projection model relating 3D object frame coordinates $X = (x,y,z)^t$ and 2D row-column points in the image frame $p = (c,r)^t$ can be expressed as:

$$\lambda \begin{pmatrix} c \\ r \\ 1 \end{pmatrix} = \begin{pmatrix} s_x f \boldsymbol{r}_1 + c_0 \boldsymbol{r}_3 & s_x f t_x + c_0 t_z \\ s_y f \boldsymbol{r}_2 + r_0 \boldsymbol{r}_3 & s_y f t_y + r_0 t_z \\ \boldsymbol{r}_3 & t_z \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad (5)$$

## 2.2 Compute camera parameters from P

From equation (5), it is clear that under full perspective projection,

$$P = \begin{pmatrix} s_x f \boldsymbol{r}_1 + c_0 \boldsymbol{r}_3 & s_x f t_x + c_0 t_z \\ s_y f \boldsymbol{r}_2 + r_0 \boldsymbol{r}_3 & s_y f t_y + r_0 t_z \\ \boldsymbol{r}_3 & t_z \end{pmatrix} \qquad (6)$$

From the projection matrix, each camera parameter can be calculated with the following:

$$r_3 = p_3 \qquad (7)$$

$$t_z = p_{34} \qquad (8)$$

$$r_0 = p_2{}^t p_3 \qquad (9)$$

$$c_0 = p_1{}^t p_3 \qquad (10)$$

$$s_x f = \sqrt{p_1{}^t p_1 - c_0{}^2} \qquad (11)$$

$$s_y f = \sqrt{p_2{}^t p_2 - r_0{}^2} \qquad (12)$$

$$t_x = \frac{p_{14} - c_0 t_z}{s_x f} \qquad (13)$$

$$t_y = \frac{p_{24} - r_0 t_z}{s_y f} \quad (14)$$

$$r_1 = \frac{p_1 - c_0 r_3}{s_x f} \quad (15)$$

$$r_2 = \frac{p_2 - r_0 r_3}{s_y f} \quad (16)$$

### 2.3 Compute projected 2D coordinates from the camera parameters and 3D coordinates

If the intrinsic and extrinsic camera parameters are known, the 3D object frame coordinates *(x,y,z)* can be used to generate the projected 2D image coordinates *(c,r)* using the collinearity equations:

$$c = s_x f \frac{r_{11}x + r_{12}y + r_{13}z + t_x}{r_{31}x + r_{32}y + r_{33}z + t_z} + c_0 \quad (17)$$

$$r = s_y f \frac{r_{21}x + r_{22}y + r_{23}z + t_y}{r_{31}x + r_{32}y + r_{33}z + t_z} + r_0 \quad (18)$$

### 2.4 Compute the projection matrix using the linear method

From the perspective projection equations described above, it is apparent that the projection matrix can be computed given a set of corresponding 2D image points *(c,r)* and 3D object frame points *(x,y,z)*. The intrinsic and extrinsic camera parameters can then be determined from the projection matrix using equations (7) through (16). The projection matrix can be computed from these 2D-3D point correspondences with the linear method.

Suppose a set of 2D image points $m_i = (c_i, r_i)$ correspond to a set of 3D object points $M_i = (x_i, y_i, z_i)$ where $i = 1,2...N$. Each pair of points are defined by the relation:

$$M_i^t p_1 + p_{14} - c_i M_i^t p_3 - c_i p_{34} = 0 \quad (19)$$

$$M_i^t p_2 + p_{24} - r_i M_i^t p_3 - r_i p_{34} = 0 \quad (20)$$

Where $p_1$,$p_2$ and $p_3$ are the row vectors of the projection matrix. Therefore 12 unknowns are defined, so a minimum of N=6 pairs of points are needed for a solution. Suppose *A* is a 12x12 matrix depending on only the calibration point coordinates and *V* is a 12x1 vector containing the values of P, $V = (p_1^t\ p_{14}\ p_2^t\ p_{24}\ p_3^t\ p_{34})^t$. Then the matrix formed by N pairs of calibration coordinates can be set up as a system of linear equations:

$$AV = 0 \quad (21)$$

Therefore, solving for *V* will yield the 12 values of the projection matrix. *V* is equal to the null vector of *A* corresponding to the zero eigenvalue. This is achieved by performing singular value decomposition on *A*, which produces $UD(S^t)$ where *D* is a diagonal matrix containing the singular values of *A* in descending order, and *U* and *S* are orthonormal matrices. The last column of *S* is equivalent to the null vector of *A*, and as a result is equal to *V* up to a scale factor.

The scale factor can be found from the fact that $||p_3||^2=1$. Therefore the scale factor $\boldsymbol{\alpha}$ can be calculated from:

$$\alpha = \sqrt{\frac{1}{V^2(9) + V^2(10) + V^2(11)}} \quad (22)$$

Knowing the scale factor, the P matrix can then be calculated as:

$$P = \alpha \begin{bmatrix} V_1 & V_2 & V_3 & V_4 \\ V_5 & V_6 & V_7 & V_8 \\ V_9 & V_{10} & V_{11} & V_{12} \end{bmatrix} \quad (23)$$

## 2.4 RANSAC Method for P matrix refinement

The linear least squares method performs poorly in the presence of mismatched data. RANSAC (random sample consensus) is an iterative method of outlier data rejection which can be used to improve the accuracy of the linear method given mismatched calibration points. The algorithm steps are the following:

1. Compute a P matrix from a subset of K calibration point pairs from a total of N pairs, where $K \geqq 6$.
2. Use this P matrix to calculate the 2D coordinates of the remaining points. Calculate the projection error between these points and the actual 2D coordinates. 3D coordinates which are accurate within a threshold are added to an inlier pool. The others are considered outliers.
3. Steps 1 and 2 are repeated *s* times to select the subset of points corresponding to the most accurate P matrix.
4. Recompute the best P matrix from this subset.

The iterations *s* can be found with the following equation given a probability p and percentage of mismatched data $\boldsymbol{\varepsilon}$.

$$s = \frac{\ln(1-p)}{\ln(1-(1-\varepsilon)^K)} \quad (24)$$

## 3. Experimental Results

The 72 calibration data points correspond to the corners on a calibration pattern, seen in Figure 1.

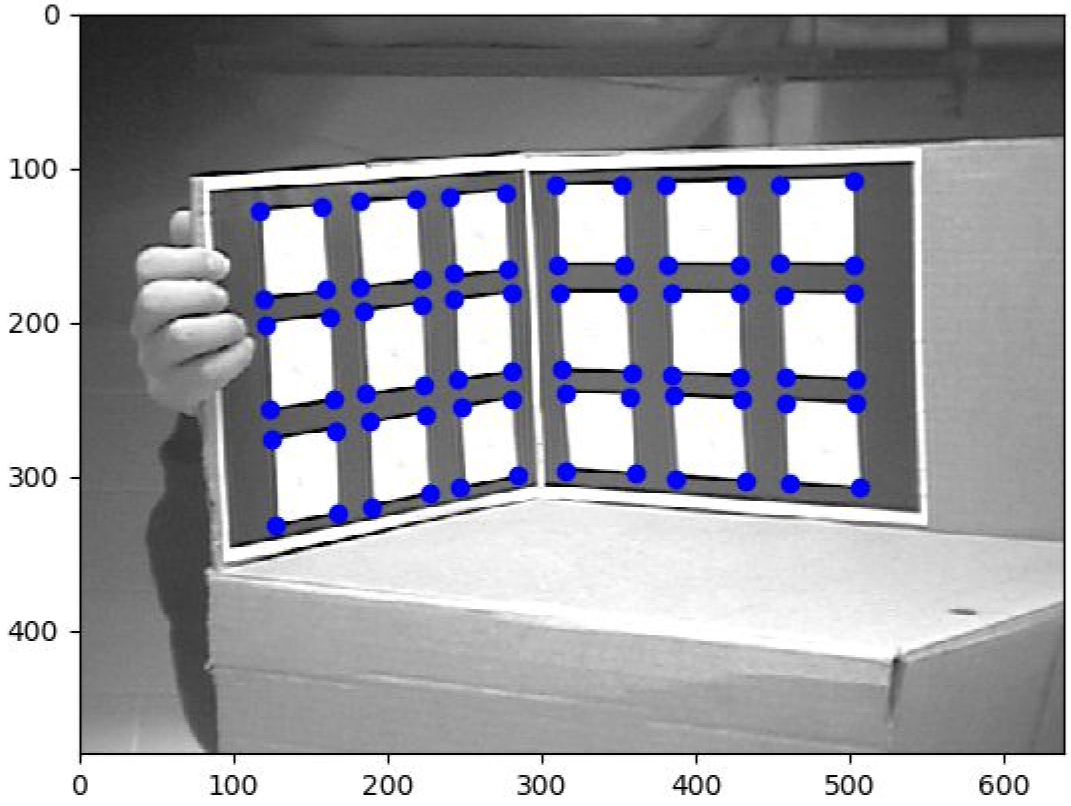


*Figure 1: Correctly plotted 2D calibration data*

### 3.2 Linear method only

The linear method alone was used to produce a projection matrix from the 3D data set containing mismatches from seven randomly selected points from this data set. The points were used to form an A matrix using equations (19), (20) and (21) and its P matrix was solved by performing singular value decomposition on the A matrix to yield the V vector.

The following P matrix was obtained:

$$P = \begin{bmatrix} -976.3 & 807.69 & -34.04 & 41896.93 \\ -71.65 & -132.57 & -1186.01 & 445805.87 \\ -0.78 & -.62 & -.0006 & 1415.93 \end{bmatrix} \quad (25)$$

The following intrinsic and extrinsic matrices were obtained from P:

$$W = \begin{bmatrix} 1239.8 & 0 & 263.79 \\ 0 & 1187.41 & 139.25 \\ 0 & 0 & 1 \end{bmatrix} \tag{26}$$

$$M = \begin{bmatrix} -.62 & .78 & -.027 & 36.62 \\ .113 & .026 & -.99 & 209.39 \\ -.78 & -.62 & -.00064 & 1415.93 \end{bmatrix} \tag{27}$$

The camera parameters were calculated with equations (7) to (16). These parameters were then used to produce the 2D projected points for the remaining points using equations (17) and (18). These are plotted in Figure 2:

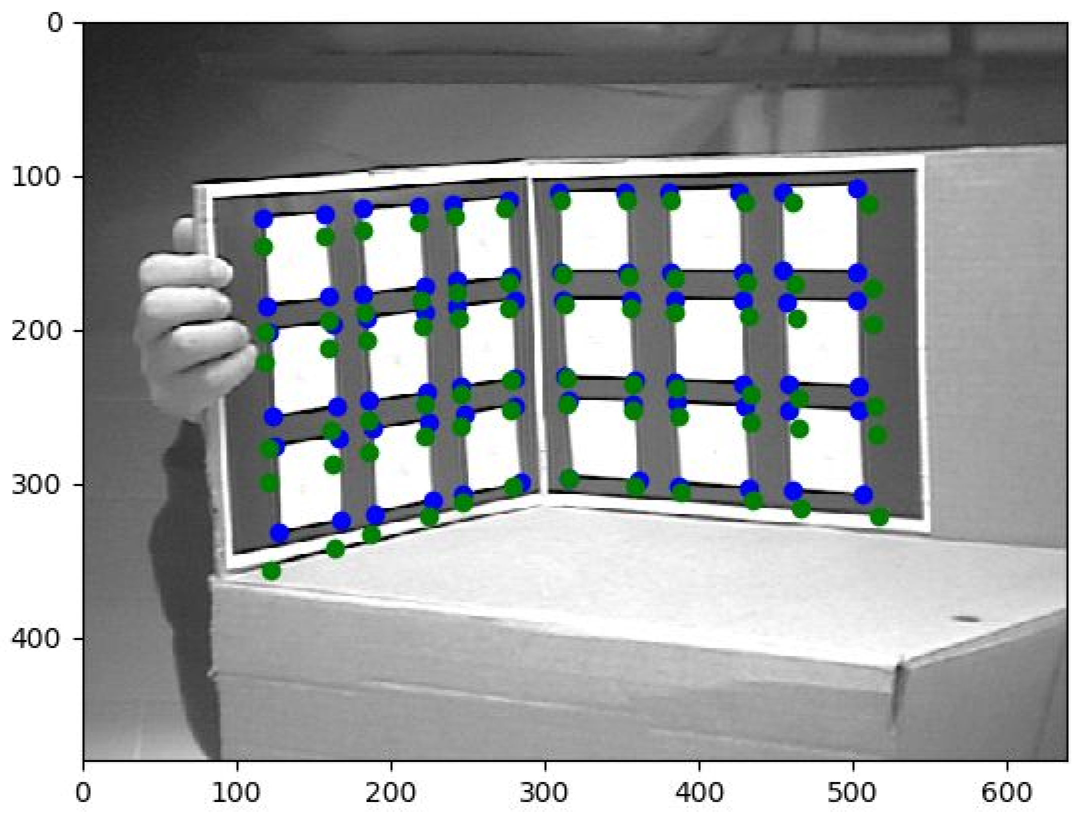


*Figure 2: 2D points generated by the linear method (green) compared to the correct locations (blue)*

The 2D points exhibit error characteristic of poor projection parameters. The distance between the projected 2D points and their correct locations was calculated for each point and averaged to represent the projection error produced by the projection matrix. Here, the average projection error was 9.92.

### 3.2 RANSAC Method

An initial projection matrix was computed using the linear method described above with seven points and used to generate the remaining 2D image points. The distance between the generated 2D points and their actual locations using the correctly matched 3D data were measured. Those points within a threshold of 5 pixels (i.e. within 5 pixels of the correct location) were considered inliers and added to a pool of inlier points. This process was repeated two times. The number of iterations was determined by equation (24) with p=.9 and $\varepsilon$=.18, corresponding to the 18% mismatch rate. The pool of inliers accumulated after two iterations was then used to generate a new P matrix. This projection matrix was calculated to be the following:

$$P=\begin{bmatrix} -1438.55 & 1410.15 & -156.89 & 666807.95 \\ -4.64 & -18.07 & -1916.2 & 701626.75 \\ -.81 & -.53 & -.2 & 2238.49 \end{bmatrix} \quad (28)$$

The following intrinsic and extrinsic matrices were obtained from P:

$$M=\begin{bmatrix} -.54 & .83 & -.03 & -170.42 \\ .19 & .119 & -.97 & -116.26 \\ -.81 & -.53 & -.2 & 2238.49 \end{bmatrix} \quad (29)$$

The camera parameters were calculated with equations (7) to (16). These parameters were then used to produce the 2D projected points for the remaining points using equations (17) and (18). These are plotted in Figure 3:

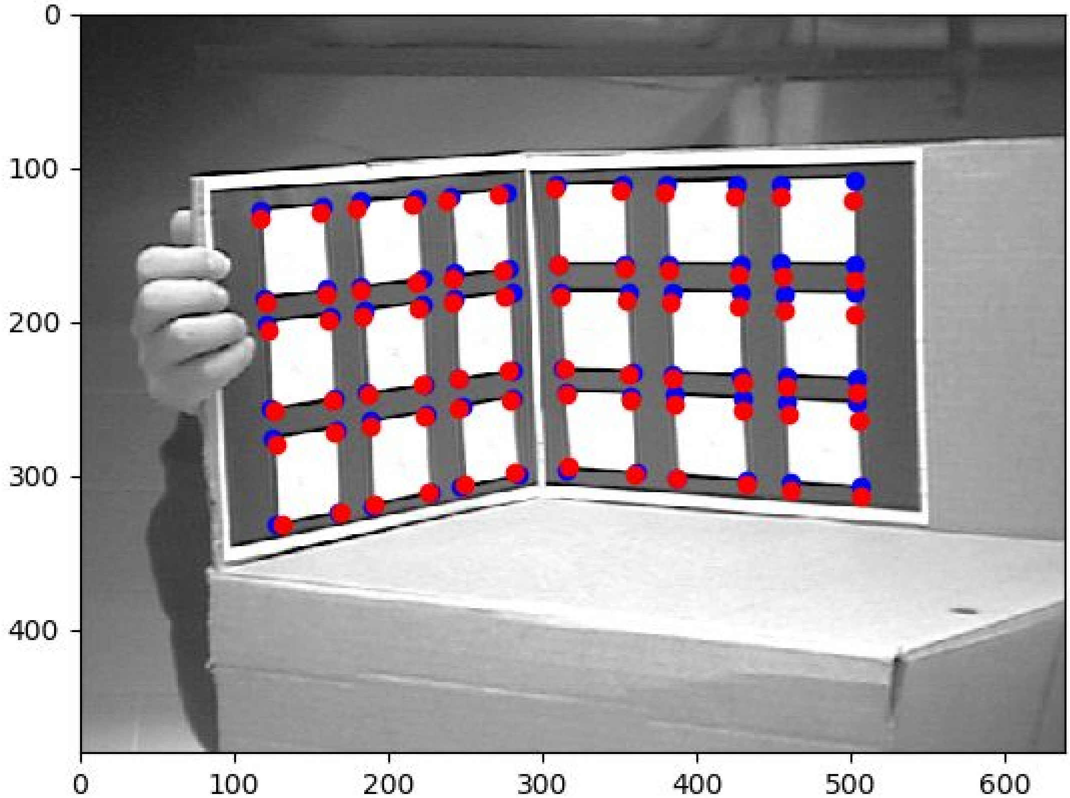


*Figure 3: 2D points generated by the RANSAC method (red) compared to the correct locations (blue)*

The distance between the projected 2D points and their correct locations was calculated for each point and averaged to represent the projection error produced by the projection matrix. Here, the average projection error was 4.34.

## 4. Conclusion and Summary

The RANSAC method demonstrated an improvement over the linear method in terms of projection error, producing 2D projected points much closer to their correct locations than the linear method. This is due to the fact that the algorithm favors correctly matched points which fall inside a projection error threshold.

In conclusion, I was able to manipulate data directly to see how the A matrix is formed from 2D/3D pairs and solved with singular value decomposition to produce projection parameters, and how these parameters are used to translate between 2D and 3D coordinates. While the RANSAC method almost always produced better results than the linear method, total accuracy heavily depended on the quality of random points selected. That is, sample sets containing more mismatched points would heavily skew the projected data. Some challenges I faced were in the particular ways Python multiplied vectors and matrices, leading to some inaccurate results that I solved by performing scalar operations for each matrix element for the parameter calculations. This caused my code to have many redundancies to ensure each element was manipulated properly. I will probably use Matlab in the future.