An Implementation of Camera Calibration Algorithms

Meredith Drennan

Department of Electrical and Computer Engineering

Clemson University

Abstract

Camera calibration is an important preprocessing step in computer vision applications. This paper seeks to provide an introduction to camera calibration procedures. It also discusses an implementation of automating point correspondences in known planar objects. Finally, results of a new implementation of Zhang's calibration procedure are compared to other open source implementation results.

1. Introduction

Computer vision is inherently plagued by the loss of dimensionality incurred when capturing a 3D photo in a 2D image. Camera calibration is an important step towards recovering three-dimensional information from a planar image. Over the past twenty years several algorithms have proposed solutions to the problem of calibration. Among the most popular are Roger Tsai's algorithm [5], Direct Linear Transformation (DLT), and Zhang's algorithm[6]. The focus of this paper is on Zhang's algorithm because it is the basis behind popular open source implementations of camera calibration i.e. Intel's Open CV and Matlab's calibration toolkit [1].

2. Methods

Calibration relates known points in the world to points in an image, in order to do so one must first acquire a series of known world points. The most common method is to use known planar objects at different orientations with respect to the camera to develop an independent series of data points. The calibration object chosen in this implementation is a 6x6 checkerboard with the corner points as the known world points. Most corner detector algorithms for camera calibration use edge detection to find the structure of the checkerboard, fit lines to the data points and compute the intersection of the lines in order to find the corners. This technique can be very accurate, providing in some cases accuracy of better than one tenth of a pixel but requires complicated line fitting algorithms. The implementation used here is based on feature detection by pinpointing windows with high variances in the X and Y directions. The points corresponding to the 36 highest variances (6x6 checkerboard implies 36 corners) are then chosen as the corner points. A simple image masking technique is used to ensure that no corner is detected twice. This is a much simpler implementation but experimental results show that accuracy is lost (see results section).

The corners may be transformed into world points by assuming an origin at the top left corner of the checkerboard and then imposing the constant distance of each square between neighboring corners.

Once a series of world points have been developed the homography matrix must be computed. This matrix becomes essentially a 3x3 matrix relating world points to image points. The homography (**H**) can then be processed into intrinsic parameter (**A**), rotation, and translation matrices. We may assume that $Z = 0$ without loss of generality because a planar object is used to perform the calibration [6].

The steps to compute the homography and intrinsic parameter matrices are as follows:

$$sm = \mathbf{H}M \qquad (1)$$

Where m = $[u, v, 1]^T$ in the image plane coordinates and M = $[x, y, 1]^T$ in the model plane coordinates. From equation 1, the homography may be determined to within a scale factor (s).

Computing the homography matrix takes the following form (from [2]):

$$\begin{array}{l} [x_1, y_1, 1, 0, 0, 0, -u_1x_1, -u_1y_1, -u_1, \\ 0, 0, 0, x_1, y_1, 1, -v_1x_1, -v_1y_1, -v_1 \\ \vdots \qquad \vdots \qquad \vdots \qquad (2) \\ x_n, y_n, 1, 0, 0, 0, -u_nx_n, -u_ny_n, -u_n, \\ 0, 0, 0, x_n, y_n, 1, -v_nx_n, -v_ny_n, -v_n]\mathbf{h'} = 0 \end{array}$$

Where **h'** is a 9x1 column vector to be reshaped into the 3x3 homography **H**. By stacking this equation for n points in an image, the over-determined system can be solved using the eigen vector associated with the second smallest eigen value found via the SVD.

Note that equation 2 introduced a matrix whose elements have various units, some of pixels, some meters, and some pixels*meters. It is important to normalize this matrix in order to achieve more stable results in the output.

The following normalization matrix is used [4]

$$N = \begin{matrix} 2/w & 0 & -1 \\ 0 & 2/h & -1 \\ 0 & 0 & 1 \end{matrix} \qquad (3)$$

where *w* and *h* are the image width and height respectively.

Once **H** has been calculated the value of matrix **B** is estimated

$$\mathbf{B} = \begin{matrix} B11 & B12 & B13 \\ B21 & B22 & B23 \\ B31 & B32 & B33 \end{matrix} \qquad (4)$$

B is a symmetric 3x3 matrix

Let b = [B11 B12 B22 B13 B23 B33] (5)

and

$$Vij = [h_ih_{ji}, h_{i1}h_{j2}+h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1}+h_{i1}h_{j3}, h_{i3}h_{j2}+h_{i2}h_{j3}, h_{i3}h_{j3}] \qquad (6)$$

$$\mathbf{G} = \begin{matrix} v12 \\ (v11 - v22) \end{matrix} \qquad (7)$$

then

$$\mathbf{G}b = 0 \qquad (8)$$

By using the homography elements h_{ij} to form the rows of **G** we can begin to solve for the elements of **B**. Note that by using several (3 or greater) different views of the planar object we will have an over-determined system and the same method as mentioned previously using the SVD can be utilized to solve this system of equations.

Once the elements of **B** are known, they can be related to the intrinsic matrix elements via the following equations

$$v_0 = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22}-B_{12}^2) \quad (9)$$

$$\lambda = B_{33}-[B_{13}^2+v_0(B_{12}B_{13}-B_{11}B_{23})]/B_{11} \quad (10)$$

$$\alpha = \mathrm{sqrt}(\lambda/B_{11}) \quad (11)$$

$$\beta = \mathrm{sqrt}(\lambda/B_{11}) \quad (12)$$

$$\gamma = -B_{12}\,\alpha^2\,\beta/\,\lambda \quad (13)$$

$$u_0 = \gamma v_0/\,\beta - B_{13}\alpha^2/\,\lambda \qquad (14)$$

γ represents the skew of the pixels and is almost always 0, therefore this parameter is set equal to zero by assuming $B_{12} = 0$ [6].

$$\mathbf{A'} = \begin{matrix} \alpha & \gamma & u0 \\ 0 & \beta & v0 \\ 0 & 0 & 1 \end{matrix} \qquad (15)$$

The intrinsic matrix, A can then be found by denormalizing A'

$$A = N^{-1}A' \qquad (16)$$

For more information on the preceding algorithms see [3], [4], [6]. Further calculations can be done to find the rotation and translation matrices corresponding to each image and the first and second order distortion coefficients as described in [6].

3. Results

Code has been written in C++ in order to test the accuracy of the algorithms for feature detection and Zhang's method discussed previously. As a baseline, the same camera (Dynex DX-WEB1C webcam) was also calibrated using two common open source implementations of Zhang's calibration; Matlab's toolkit available on the web [2], and Intel's Open CV implementation. The number of input images to all three calibration tools has been kept constant at three. Using Blepo's demo implementation of the Open CV code requires a nine square by six checkerboard input while both Matlab and the author's version received the same 6x6 checkerboard images, therefore roughly the same data was fed to all three versions.

A. Corner Detection

Using a standard feature detection algorithm which searches for areas of high variances in X and Y directions, accuracy can be obtained at an average of within 2.2 pixels with a standard deviation of 1.4. While this is clearly not as accurate as calibration tools which use line fitting techniques, the prime advantage of this method of corner detection is simplicity.

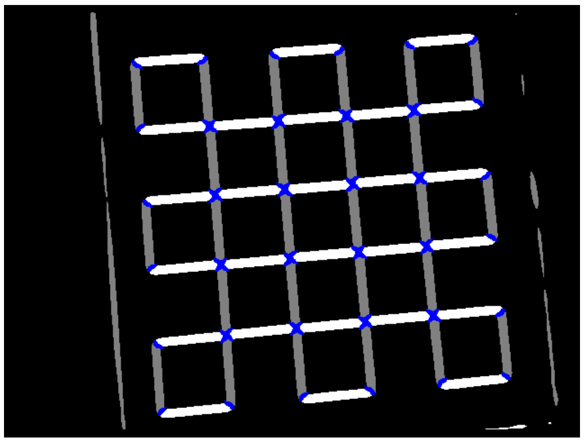

Figure 1 Checkerboard Variances and Corners

Figure 1 shows an output of the feature detection on a standard checkerboard. Lines in gray mark areas of high variance in the X direction, and lines in white indicate areas of high variance in the Y direction. The points in blue mark the possible corners.

A standard sort algorithm finds the areas of highest variance. Two techniques are used to ensure that a corner is not detected twice, first the image is fed to a Canny edge detector in order to reduce the number of possible corners. Then an image mask ensures that no other point within 15 pixels is marked as a corner.

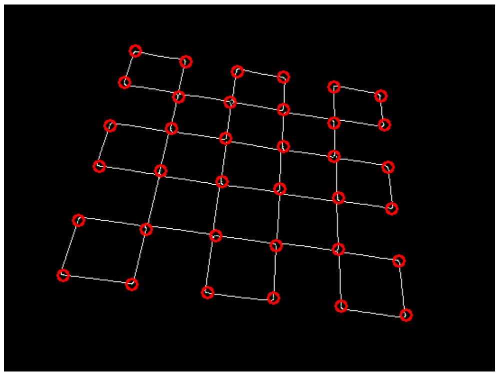

Figure 2 Corners Found

B. Calibration Algorithm

Once the corners have been detected, the calibration algorithm performs a series of steps as indicated in the methods section. The output of the calibration is an intrinsic parameter matrix (**A**) whose elements are described below and compared to Matlab and Open CV results.

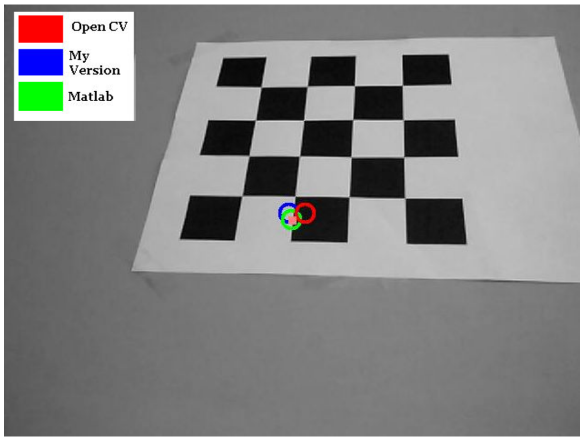

Figure 1 Principle Point

Figure 1 shows the observed principle point according to each implementation. All shown estimations are reasonable as the expected principle point would be at the center. For reference purposes a pink square is drawn at the center of the image.

The matrix **A** also contains values of α and β, where α/ β is the ratio of pixel width to height and is equal to 1 for most standard cameras. For the given camera this ratio averages at .9998 in Open CV, and .9919 in Matlab, whereas the mean ratio of the author's implementation equals 1.53. While this parameter provides a large drawback to the use of this software as a calibration tool, it is important to note that both Matlab and Open CV perform optimization after the initial calculations based on Zhang's method which use apriori knowledge of typical camera values such as α/ β ~1, [1] therefore future work can provide much better results.

Implementation	**α/β**
Open CV	0.9998
Matlab	0.9919
Author's Version	1.536

Figure 2 Aspect Ratio Comparison

One advantage to the use of the software developed by the author and described throughout this paper over other open source calibration tools is the capability of displaying not only the intrinsic matrix A, but also rotation, translation, and homography matrices for each image as well as focal length. This may not be important to a casual user, but it has the potential to help guide those who are developing their own calibration software.

Disclaimer: the focal length, rotation, and translation matrices are not calculated using Zhang's method but are the result of an incomplete implementation of Tsai's algorithm the author developed prior to implementing Zhang's algorithm. See [5] for more detail.

4. Conclusions

This paper describes a novel implementation of Zhengyou Zhang's calibration algorithm with automatic feature detection to find the corners of the calibration object. The accuracy of which is reasonable although not as impressive as the currently available open source software. Future work can be done to improve this by implementing optimization algorithms such as gradient descent or Levenberg-Marquardt. One advantage to the author's calibration implementation as described throughout this paper is its ability to display intermediate matrices for guidance to those users who are developing their own calibration software.

5. References

[1] Bouguet, Jean-Yves, (2008) Camera Calibration Toolbox for Matlab http://www.vision.caltech.edu/bouguetj/calib_doc/

[2] Boyle, Roger, Vaclav Hlavac, and Milan Sonka (1999) Image Processing, Analysis, and Machine Vision Second Edition. PWS Publishing.

[3] Hanning, Tobais and Rene Schone (2007) Additional Constraints for Zhang's Closed Form Solution of the Camera Calibration Problem. University of Passau Technical Report.

http://www.fim.uni-
passau.de/fileadmin/files/forschung/mip-berichte/MIP-
0709.pdf

[4] Teixeira,Lucas Marcello Gattass, and Manuel
Fernandez. Zhang's Calibration: Step by Step
http://www.tecgraf.puc-
rio.br/~mgattass/calibration/zhang_latex/zhang.pdf

[5] Tsai, Roger Y (1987) A Versatile Camera Calibration
Technique for High Accuracy 3D Machine Vision
Metrology Using Off the Shelf Cameras and Lenses
IEEE Journal of Robotics and Automation Vol. RA-3 No 4
pp.323-346
http://www.vision.caltech.edu/bouguetj/calib_doc/papers/T
sai.pdf

[6] Zhang,Zhengyou. (1999) A Flexible New Technique for
Camera Calibration. Microsoft Research Technical Report