RBE549: Homework 1: AutoCalib

Edwin Clement
MS RBE
Email: eclement@wpi.edu

***Abstract*—Calibrating a camera for use in Computer Vision is one of the primary steps needed before any further processing is done downstream. The better and closer the calibrated output, the less effort the rest of the pipeline needs to process the data. In this paper, camera calibration was obtained via the method given by Zhang Z. [1].**

I. Introduction

There are two relevant parts to camera calibration. Conversion to world coordinates to image coordinates which is denoted by extrinsic parameters and then image coordinates to pixel positions which are denoted by intrinsic parameters.

The camera extrinsic parameters can be encoded with the following matrix:

$$Rt = \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix}$$

The camera intrinsic parameters are denoted by the following matrix:

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Here, α and β govern the scaling factor, and γ is the degree of skew. The point (u_0, v_0) is the optical center of the system.

II. Dataset

To estimate camera parameters, I initially used the given dataset that contains 13 images taken by a modern phone. To further test out the system, I created another dataset upon which exaggerated lens distortion was applied in GIMP.

III. Initial Parameter Estimation

I used the given `pdf` as ground truth. Using `convert_from_path` method from the package `pdftoimage`, the given pdf was converted to an reference image manipulable by OpenCV.

Using `cv2.findChessboardCorners()`, the world corners are obtained.

A. Getting Approximate Camera Intrinsic Matrix

Using the same `cv2.findChessboardCorners()`, we can get the corners for each image. Given that the images are of the same flat object and the fact that the corners are sorted when returned by the function, we know the exact points in both images that correspond to each other.

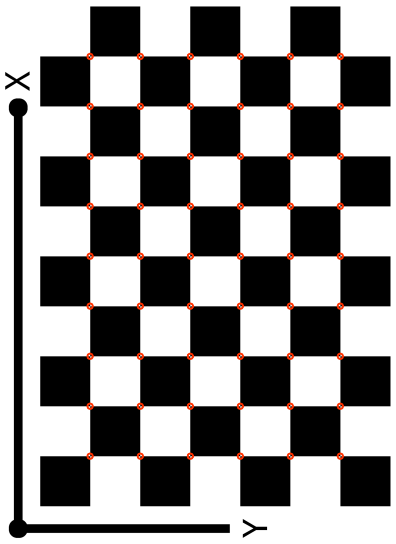


Fig. 1: Ground Truth Checkerboard pattern 9×6

The following are the detected corners in the image

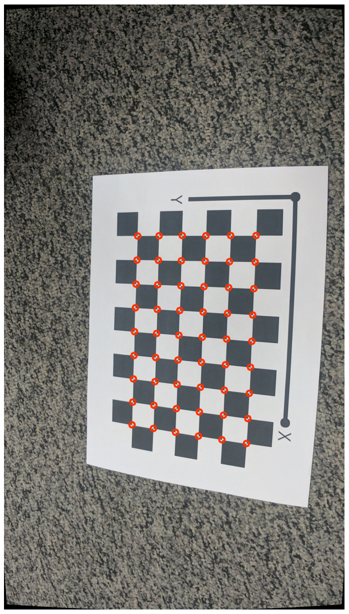


Fig. 2: Corners in the given image

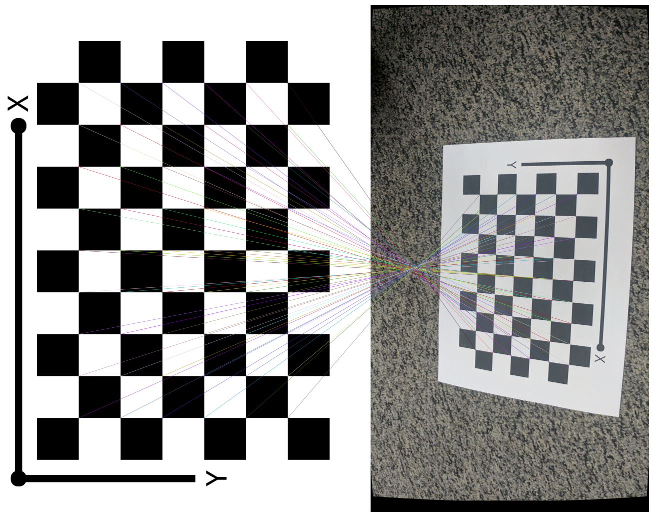


Fig. 3: Matches with the ground truth, regardless of rotation

Then, Singular Vector Decomposition was used to calculate the Homography using all the above point correspondences. RANSAC was not needed here as the points were in order and we could just use all of them in the calculation as is.

The calculated homography can then be represented in the following form.

$$H = A \times R_t$$

where A is the camera distortion and R_t is the rotation/translation between the ground truth to the image frame

To calculate A, we can use the following approach.

Given that $H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix}$

we have a closed form solution for A of the form

$$h_1^T (A^{-T} A^{-1}) h_2 = 0$$

Let $B = A^{-T} A^{-1}$, be a symmetric matrix. Then, this equation can also be written as

$$h_1^T (A^{-T} A^{-1}) h_2 = v_{ij}^T b$$

where

$$b = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix}$$
$$v_{ij} = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix}$$

By using similar math as we used in calculating homography, we can get A matrix. In my particular case, my initial guess was

$$b = \begin{bmatrix} 1.141 \times 10^7 \\ -6.8529 \times 10^8 \\ 5.814 \times 10^7 \\ 1.866 \times 10^5 \\ -0.0006 \\ 0.9999 \end{bmatrix}$$
$$A = \begin{bmatrix} 1188.1776 & 0.0000 & 557.6944 \\ 0.0000 & 561.9869 & 1201.2464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

Note that it is assumed there is no skew error which is why $\gamma = 0$.

B. *Estimate approximate R and t or camera extrinsics*

Using the approximate values of A and h, we calculate the extrinsic parameters of the camera for each image.

$$r_1 = \lambda K^{-1} h_1$$
$$r_2 = \lambda A^{-1} h_2$$
$$r_3 = r_1 \times r_2$$
$$t = \lambda A^{-1} h_3$$

The Resultant vectors can be stacked to form a 3 $times4$ matrix like so

$$R_t = \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix}$$

C. *Approximate Distortion k*

This however doesn't account for the radial distortion inherent in cameras. The initial guess for the distortion parameters were set to 0, i.e.

$$k = \begin{bmatrix} 0 & 0 \end{bmatrix}$$

D. *Non-linear Geometric Error Minimization*

For numerically approximating it, we use an iterative approximating method given in SciPy, `scipy.optimize.least_squares`.

We take the projected corners according to the calculated camera parameters, distort it with the assumed values for distortion and then minimize the error with the corners found by the visual method used by `cv2.findChessboardCorners()`.

We are trying to minimize the following

$$\arg \min_{f_x, f_y, c_x, c_y, k_1, k_2} \sum_{i=1}^{N} \sum_{j=1}^{M} \| \mathbf{x}_{i,j} - \hat{\mathbf{x}}_{i,j}(A, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j, k) \|$$

This gives us the following optimized camera intrinsic matrix and the distortion co-efficients:

$$A' = \begin{bmatrix} 1188.1776 & 0.0000 & 557.6944 \\ 0.0000 & 561.9869 & 1201.2464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$
$$k' = \begin{bmatrix} 0.0002 & 0.0000 \end{bmatrix}$$

TABLE I: Re-Projection errors on given dataset

Experiment	Error Value
Initial A and k = [0, 0]	542.64
Optimized A and k	541.05

IV. Results

Following are the results obtained before and after applying the optimization method

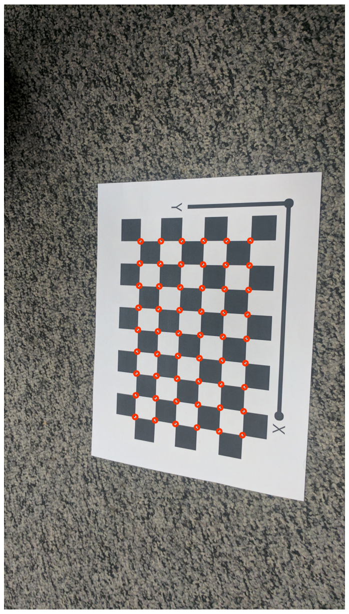


Fig. 4: Reprojected Corners on given dataset

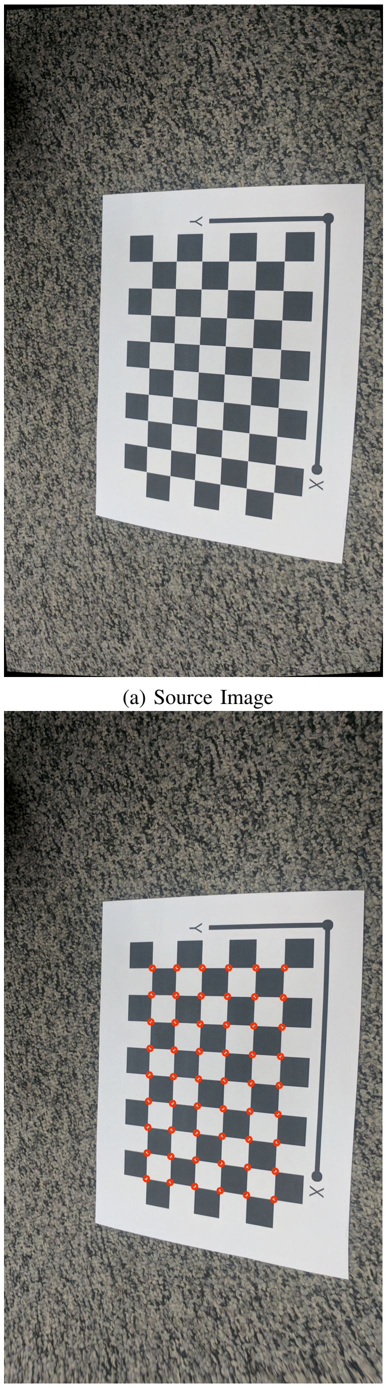
This low improvement is due to the extremely good software of the modern-era smartphones. In order to truly test the limits of the algorithm, I modified the images used with a lens distort filter in GIMP and here are the results for that dataset. With this one, the results are as follows:

TABLE II: Re-Projection errors on new Data: 4% better

Experiment	Error Value
Initial A and k = [0, 0]	888.46
Optimized A and k	852.14

References

[1] Z. Zhang. A flexible new technique for camera calibration. *IEEE transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.



(a) Source Image

(b) Undistorted Image with Projected Corners

Fig. 5: Applying algorithm on manually distorted data