Homework 1 - AutoCalib

Krunal M. Bhatt
Masters of Science in Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA - 01609
Email: kmbhatt@wpi.edu

***Abstract*—Homework 1 focuses on the estimation of the parameters of a camera. These parameters include focal length, distortion coefficients, and principle point. Calibrating these parameters is known as Camera Calibration. We will follow and implement an automatic way to calibrate the camera using the approach presented by Zhengyou Zhang [1].**

I. Introduction

We have the camera calibration matrix K as:

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

and radial distortion parameters are denoted by k_1 and k_2 respectively. In this project, we will estimate the following parameters: $f_x, f_y, c_x, c_y, k_1, k_2$. To calibrate the camera we need a calibration target, in our case, we are using a checkerboard image as depicted in Fig. 1.

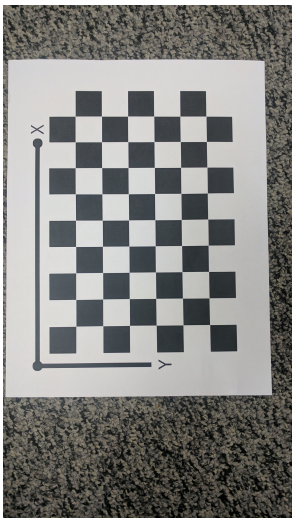


Fig. 1. Calibraiton Target

This checkerboard image is used as our calibration target. In this image, each square is 21.5mm in size. The Y-axis has an odd number of squares while the x-axis has an even number of squares on it. It is a general practice to ignore the outer squares (the row and columns of squares on the edge of the checkerboard). We have been provided with 13 images from a Google Pixel XL phone with focus locked, which we will be using to calibrate.

II. Initial Parameter Estimation

A. Acquiring Image and World Frame:

The first step in this process is to find the corners of the checkerboard image using the $cv2.findChessboardCorners$ function. We take a 9 x 6 pattern size such that all the inner points excluding the edges are covered. We have a total of 54 points. Fig. 2 shows how the image looks like when the function is used.

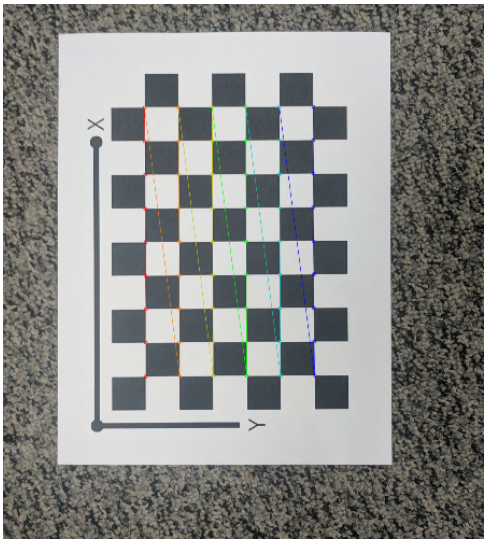


Fig. 2. Find Corners

The next step would be to define the world coordinate frame, in 3D, for all the points that are shown in Fig. 2. The **Z** value for his coordinate system will be 0 as the checkerboard is on the XY plane. Fig. 3 shows how the system looks like, it goes up until the final value (X, Y, Z) = (172, 107.5, 0)

Now, we have the world coordinates, and the correspondence between the world coordinates and the pixel coordinates in 2D is available and is given by:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Here, λ is a scale factor and r_i is the ith column vector and t translation vector.

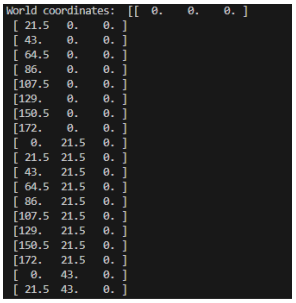


Fig. 3. World coordinates

B. Solving for Approximate K (Camera Intrinsic Matrix):

For the closed-form solution, vector b which is shown below is used. Also, DLT is used to find the homography **H** between the 3D and 2D points.

$$b = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix}$$

Here, matrix B_{ij} is the matrix described in section 3.1 equation 5 in Zhang's paper. We have the following equation describing the solution to find the b matrix.

$$Vb = 0$$

Here, V is a 2n x 6 matrix. If $n \geq 3$, we will have in general a unique solution **b** defined up to scale factor.

$$V = \begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix}$$

We can use the skewless constraint $\gamma = 0$, i.e.;

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \mathbf{b} = 0$$

C. Estimating Approximate R and t (Camera Extrinsic):

Let i^{th} column of H (homography matrix) is h_i. Once K is calculated, we can calculate R and t using the formulas below.

$$\begin{aligned} r_1 &= \lambda A^{-1} h_1, \\ r_2 &= \lambda A^{-1} h_2, \\ r_3 &= r_1 x r_2, \\ t &= \lambda A^{-1} h_3, \end{aligned}$$

here, λ = $1/A^{-1}h_1$. We get the R matrix as:

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix}^T$$

Now, we take the initial value of distortion to be 0. i.e.;

$$K_c = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$$

This is considered to be a good initial point and also that we have assumed the camera has minimal distortion.

$$\sum_{i=1}^{N} \sum_{j=1}^{M} \|x_{ij} - x_{ij}(K, R, \hat{t}_i, Xj, k_c)\|$$

D. Non-Linear Geometric error Minimization

Now that we have the initial estimates for K, R, t, k_c, we optimize the geometric error shown in the equation above. We typically minimize this cost function to achieve the desired results. We do so by using the $scipy.opt.leastsquare$ function.

Formally, the optimization problem is as follows:

$$\text{argmin}_{f_x, f_y, c_x, c_y, k_1, k_2} \sum_{i=1}^{N} \sum_{j=1}^{M} ||x_{i,j} - \hat{x}_{i,j}(K, R_i, t_i, X_j, k)||$$

To reduce the error, the function needs to be minimized for the error while dealing with radial distortion as well. we use the below variables to get the distortion: k_1 and K_2 are radial

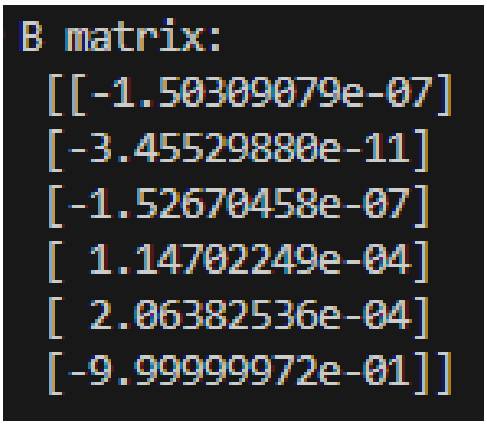
$$\begin{aligned} \breve{u} &= u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \\ \breve{v} &= v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \end{aligned}$$

distortion coefficients. (u0, v0) is the principal point obtained from intrinsic parameters.

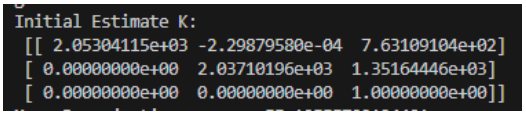
III. Results

Following what Zhang said and implementing it, we get the the following outputs:

1) B matrix initial estimate:



2) Initial Estimate of K matrix:



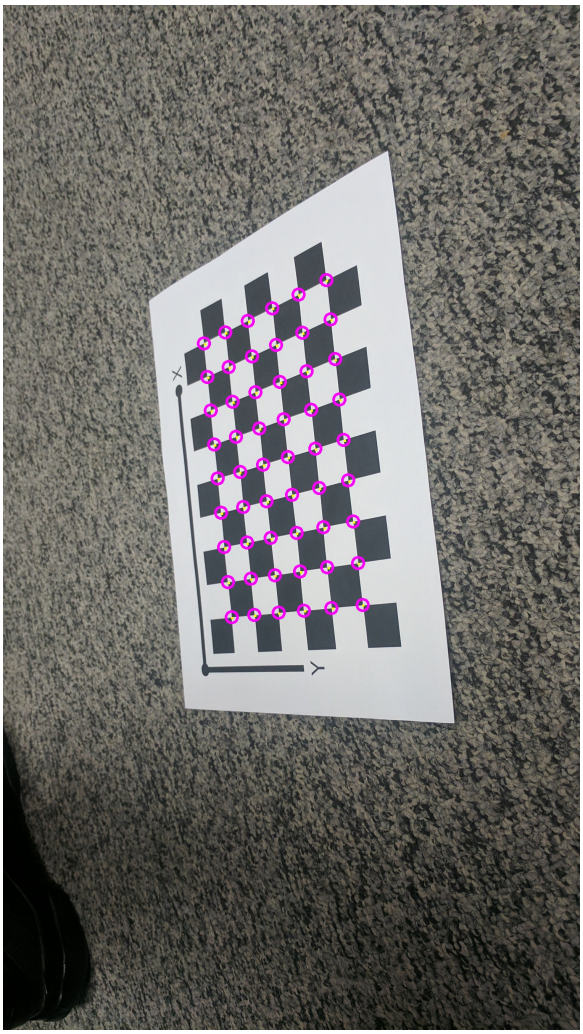
3) Final output:

```
v0:  1351.6444605728395
lambda_:  -0.6335494524094911
alpha:  2053.041145107885 8
beta:  2037.1019621867842
gamma:  -0.00022987958042074827
Mean Reprojection error:  55.10555792124401
Optimized K:  [[ 2.05304115e+03 -2.29879580e-04  7.63109104e+02]
 [ 0.00000000e+00  2.03710196e+03  1.35164446e+03]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]
Calibration matrix optimimzed:

[[ 2.05304115e+03 -2.29879580e-04  7.63109104e+02]
 [ 0.00000000e+00  2.03710196e+03  1.35164446e+03]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]
Distortion coeff optimized:
0.0

Mean Reprojection error optimized:
55.10555792124403
```

4) Reprojected points:
The final output after implementing the algorithm shows the original points (purple) and the re-projected points (yellow).



Furthermore, I faced some issues while optimizing the model. The issue was that sometimes the $opt.least_squares$ function would result like "Value Error", meaning that the value is either NaN or inf. It was due to the random initial value taken for re-projected error. I used the "np.empty" function to initialize the same. Changing it to np.zeros solved the issue of ValueError. The technique was successfully implemented in this assignment.

REFERENCES

[1] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 1330 – 1334, 12 2000.