Homework 1: AutoCalib
CMSC733

Using 1 Late Day

Arjun Gupta
University of Maryland
College Park, Maryland 20740
Email: arjung15@terpmail.umd.edu

I. Introduction

The aim of this project is to estimate the extrinsic parameters, intrinsic parameters and the distortion coefficients of a camera for which a set of 13 images of checkerboard are provided. For the purpose of this project I implement the technique suggested by Zhang in [1]. In the sections below I discuss the our implementation and the obtained results.

II. Implementation

A. Detecting the Corners/Points of Representation

In order to find an estimate of the *K* matrix, I first need to find the corners of the provided images. I use the in built function of opencv $cv2.findChessboardCorners()$ to detect all the corners. For the refinement of the detection I implemented $cv2.cornerSubPix()$. An image with the detected corners is shown below in Figure 1.

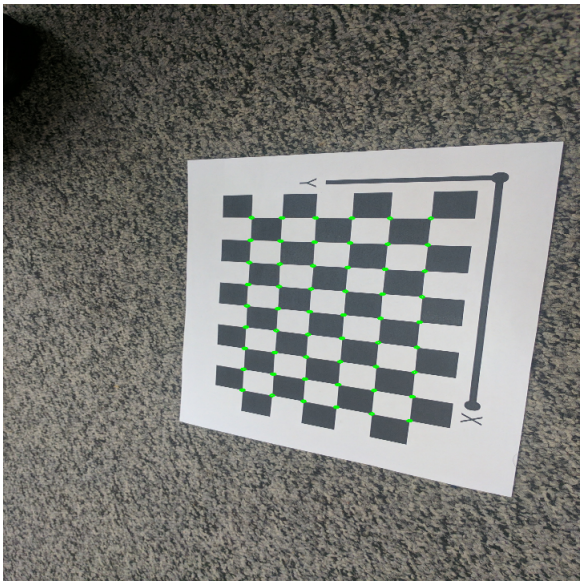

Fig. 1: A Checkerboard image with detected corners in green color

B. Finding Homography

Once the corners are detected we utilize the information we know about the squares of the checkerboard to associate the detected corners with the corresponding world point coordinates. For example we associate the coordinates $(21.5, 21.5)$ (where 21.5 is the size of a square in mm) to the top left corner. Similarly we do this for all the remaining 53 corners. Since we need only four points to obtain homography we use the outermost corners for that. We calculate homography using $cv2.findHomography()$.

C. Estimating extrinsics R and t or the camera

After estimation of the initial intrinsic parameters (apart from distortion coefficient), I follow the instructions and the method detailed in [1] to estimate the R and t. While exploring the literature I found that some of the implementations used $\lambda = \left(\frac{||(A^{-1}h_1 + A^{-1}h_2)||}{2}\right)^{-1}$, thus I calculated my reprojection error using this λ and the $\lambda = \frac{1}{||A^{-1}h_1||}$ as suggested in [1]. Using the two different formulations we get the reprojection error values as **2.67** and **2.06** respectively. Hence, based on the reprojection error value I stick with the formula suggested by Zhang in [1]. Furthermore, since the obtained R matrix do not satisfy all the properties of a rotation matrix, I attempted to implement the method suggested in Appendix C but the results (error value) seem to degrade. I believe it is due to the bug in the implementation. It would be great if any resources or solution can be provided for this.

D. Distortion Coefficients

As given in the homework page we take the initial distortion coefficients as $(0, 0)$. After optimization the obtained distortion coefficients had a very small value suggesting no or very less distortion in the images. Visually the results made sense as the distortion in the original images seemed to be almost none.

E. Non-linear Geometric Error Minimization

For optimizing the non linear equation I use functionality of the scipy library. I implemented $scipy.optimize.least\_square()$ [3] [2] to solve this problem. For minimization purpose I used $Levenberg–Marquardt$ to minimize the error function. I only optimize the non zero parameters of the initially estimated intrinsic matrix and not all the nine values. The L2 norm is used to measure the difference between the estimated and the actual corner points.

F. Results

- Initial Estimate of K:

$$\mathrm{K} = \begin{bmatrix} 2.034405e+03 & -4.130692e-01 & 7.739411e+02 \\ 0.000000 & 2.019987e+03 & 1.361549e+03 \\ 0.000000 & 0.000000 & 1.000000e+00 \end{bmatrix}$$

$$\mathrm{k}_s = [0 \quad 0]$$

- After optimization value of K:

$$\mathrm{K} = \begin{bmatrix} 2.030606e+03 & -2.945681e-01 & 7.739156e+02 \\ 0.000000 & 2.024856e+03 & 1.391645e+03 \\ 0.000000 & 0.000000 & 1.000000e+00 \end{bmatrix}$$

$$\mathrm{k}_s = [0.00575877 \quad -0.01440005]$$

The reprojection error is calculated as the Root mean square error value between the original corners and projected corners (obtained using the new K). I obtain the reprojection error after image rectification: **2.0629** per pixel. The images show from figure 2 to 14 shows the output of the pipeline with reprojected corners and the caption denotes the name of the original raw file.

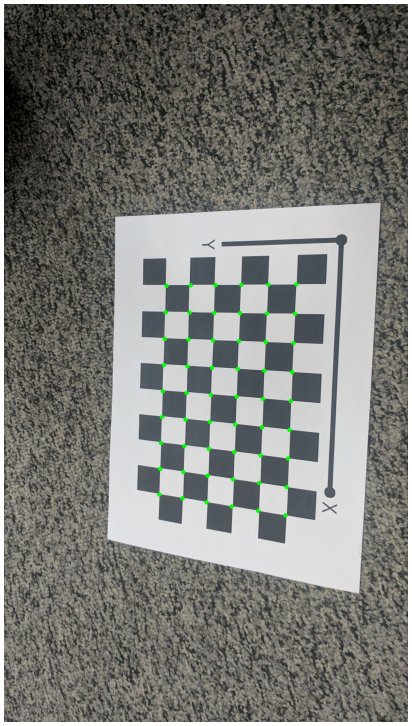

Fig. 2: Output of IMG_20170209_042627.jpg

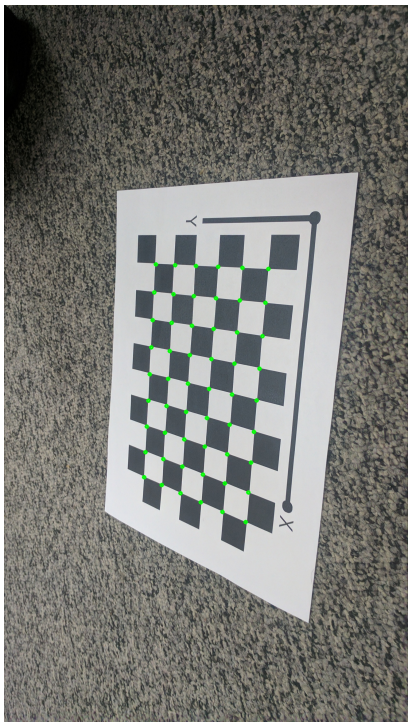

Fig. 3: Output of IMG_20170209_042630.jpg

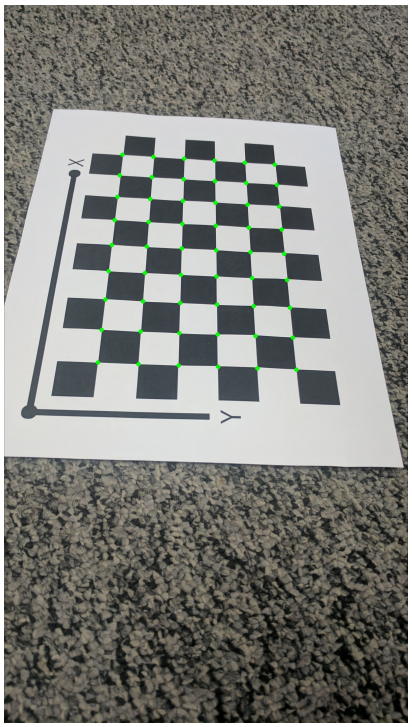

Fig. 4: Output of IMG_20170209_042612.jpg

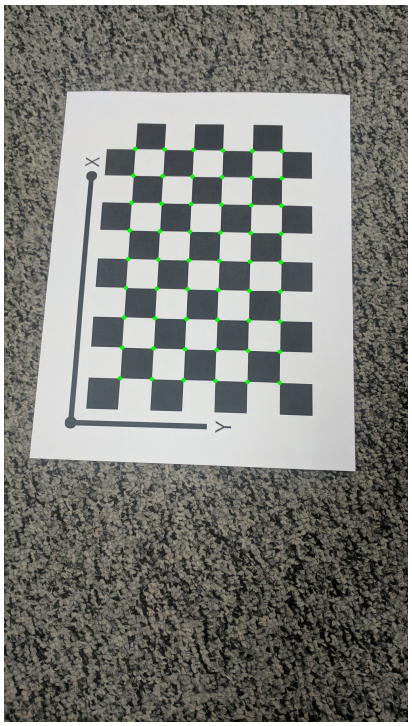

Fig. 5: Output of IMG_20170209_042606.jpg

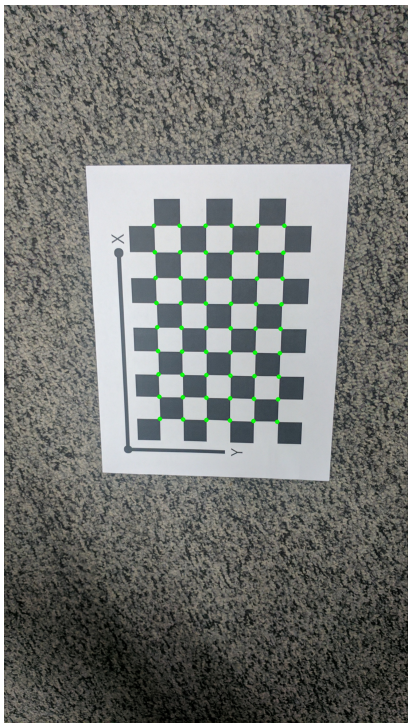

Fig. 6: Output of IMG_20170209_042621.jpg

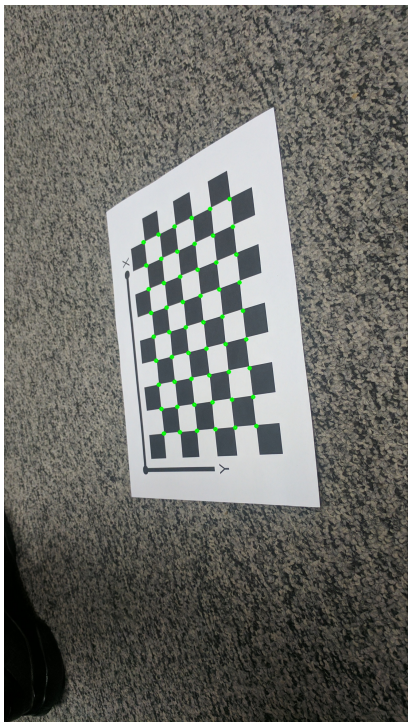

Fig. 7: Output of IMG_20170209_042634.jpg

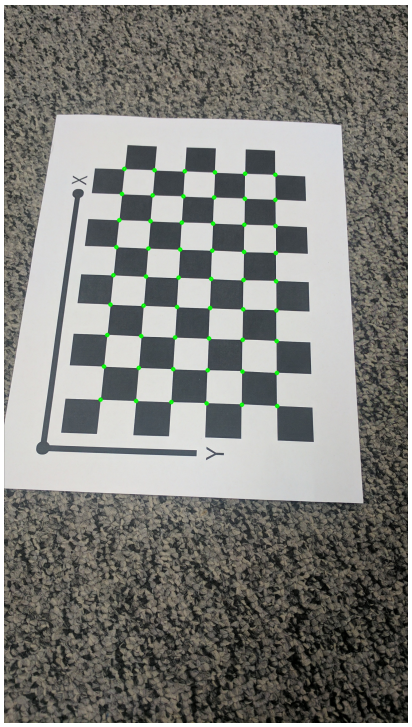

Fig. 8: Output of IMG_20170209_042608.jpg

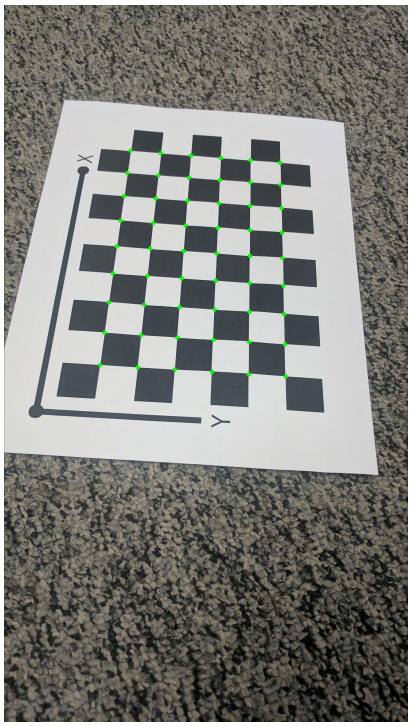

Fig. 9: Output of IMG_20170209_042610.jpg

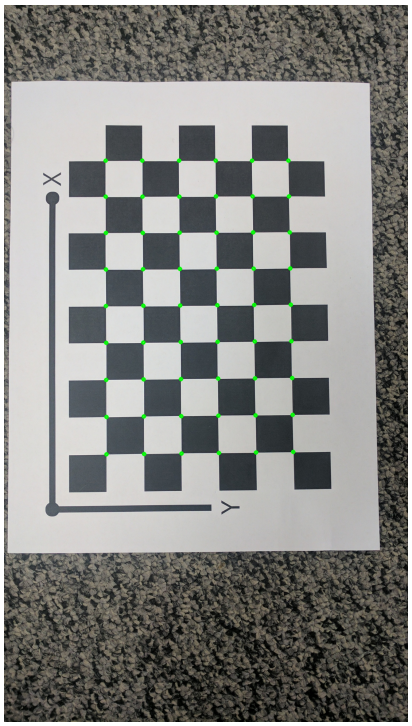

Fig. 11: Output of IMG_20170209_042616.jpg

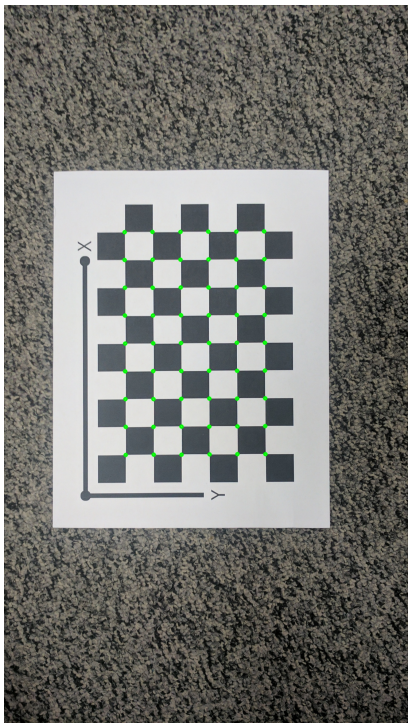

Fig. 10: Output of IMG_20170209_042614.jpg

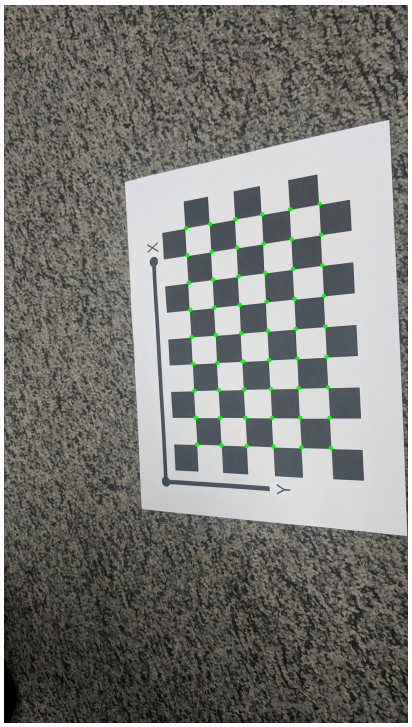

Fig. 12: Output of IMG_20170209_042624.jpg

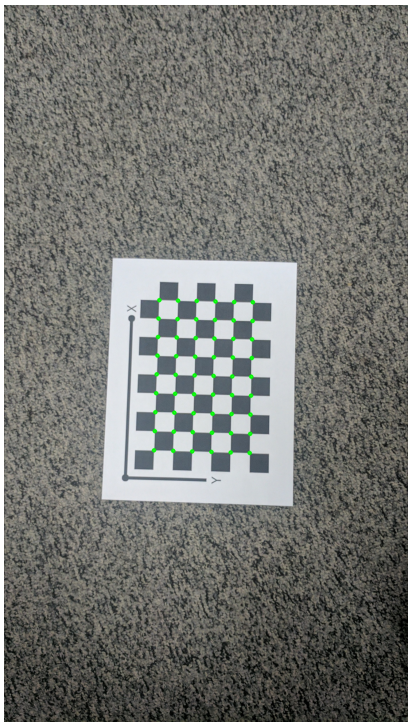

Fig. 13: Output of IMG_20170209_042619.jpg

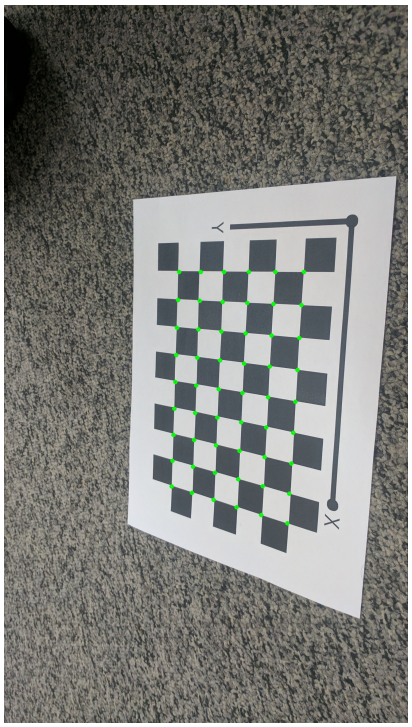

Fig. 14: Output of IMG_20170209_042629.jpg

References

[1] Zhengyou Zhang. “A flexible new technique for camera calibration”. In: *IEEE Transactions on pattern analysis and machine intelligence* 22.11 (2000), pp. 1330–1334.

[2] *Reference Implementation*. https://kushalvyas.github.io/calib.html. Accessed: 2020-04-08.

[3] *Scipy Documentation*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.least_squares.html. Accessed: 2020-04-08.