AutoCalib

Shounak Naik
Robotics Engineering Department,
Worcester Polytechnic Institute,
Worcester, MA, USA.
ssnaik@wpi.edu

I. Introduction

This report explains the working and the results of a Camera Calibration technique as described by Zhang. This is done by using 13 different perspectives of a Checkerboard. These images are captured with a focus locked setting.

II. Camera Calibration

A. Estimate K

I have used the *cv2 ChessBoardCorners* on each image to find out the corners in each of the 13 images. An example image can be seen in Figure 1
There are 9 rows and 6 columns in each image. We know that the physical distance between corners is 21.5mm. Thus I have found out the world point coordinates for each corner. Additionally, I also know the image coordinates of each point from the above *cv2* function. Thus, I have found out the homography between the world coordinates and the image coordinates. Using these estimated homography matrices, I have then found out the camera intrinsic matrix K by following Section 3.1 in Reference 1. I have also explicitly computed the γ instead of choosing it as 0.

B. Estimate R and t

Using the estimated K and the homography matrices, I have computed the R and the t matrix for each image with reference to Section 3.1 in the paper. The K matrix will be the same for all images as it is the camera's intrinsic parameters. Since the calculated R matrix doesn't necessarily fulfill the properties of a rotation matrix, I have followed the procedure given in the Appendix C.

C. Distortion Coefficients

I have taken both the distortion coefficients, $k1$ and $k2$ as 0.

D. Non Linear Geometric Optimization

We now have estimates of both the extrensic and the intrinsic parameters. But these are just estimates and we want to find out the optimized parameters. These parameters are optimized on the projection error such that distance between the projected point and the ground truth should be as minimum as possible. I have used the *scipy optimize* package to achieve the same. To use this , I have written a function that outputs the projection error as a 1 dimensional vector.

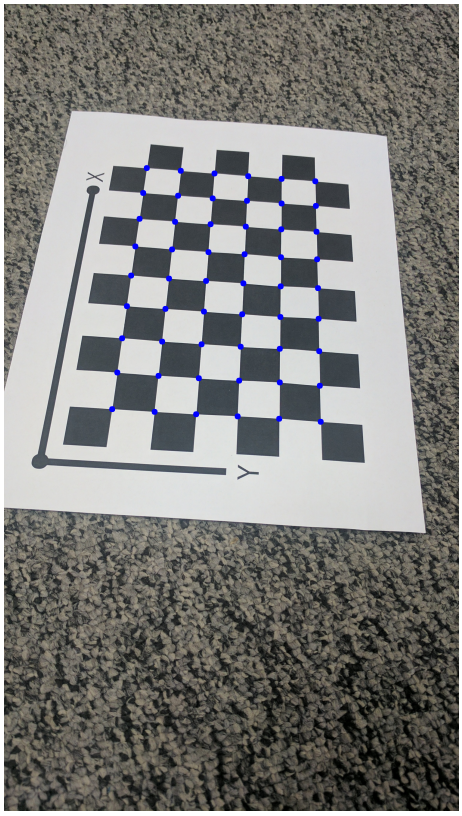

Fig. 1. Chess Board Corners

E. Results

Initial Estimates:

$$K = \begin{bmatrix} 2052 & -0.369 & 763 \\ 0 & 2036 & 1352 \\ 0 & 0 & 1 \end{bmatrix}$$

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

After optimization:

$$K = \begin{bmatrix} 2046 & -0.83 & 763 \\ 0 & 2031 & 1351 \\ 0 & 0 & 1 \end{bmatrix}$$

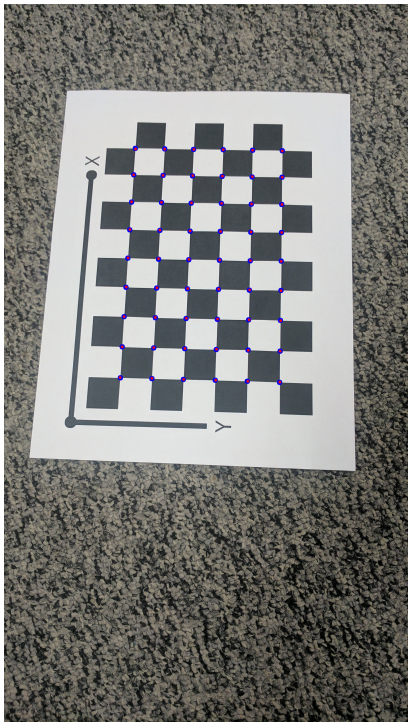

Fig. 2.

$$k_s = \begin{bmatrix} 0.095 & -0.565 \end{bmatrix}$$

The initial error is 1.96 per pixel and after optimization it changed to **1.843**.

III. Conclusion

This assignment helped me understand the camera calibration tasks taught in the class to a much deeper level. The reprojected points can be seen in the following images. The blue circles are the corners detected by *cv2* initially and the red dots are the projected points using the parameters that we calculated earlier. The reason that these points are very close to each other is because the images are not much distorted in the first place.

References

[1] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr98-71.pdf
[2] https://github.com/SrinidhiSreenath/AutoCalib-Caibration-of-Camera/blob/master/Wrapper.py
[3] https://cmsc733.github.io/assets/2019/hw1/results/pdf/patilameya$_h w1.pdf$

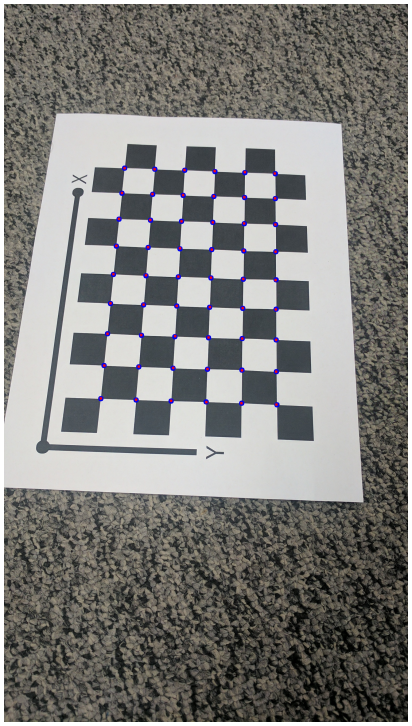

Fig. 3.

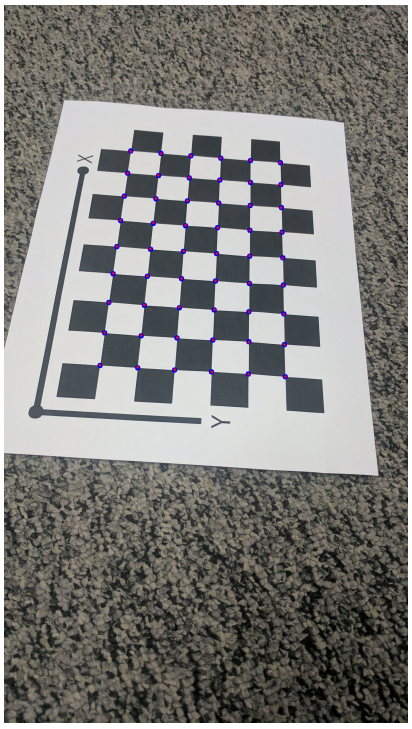

Fig. 4.

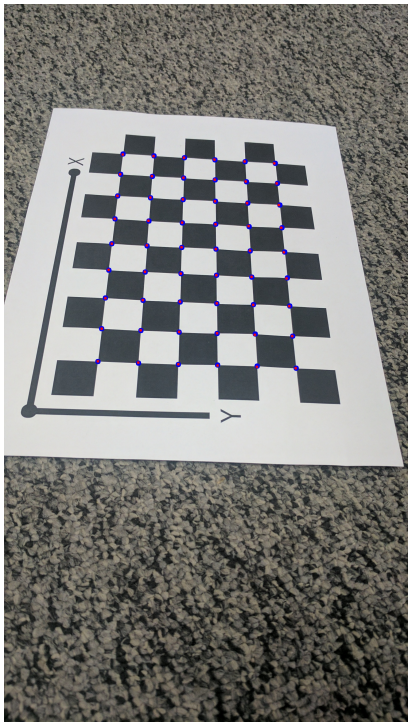

Fig. 5.

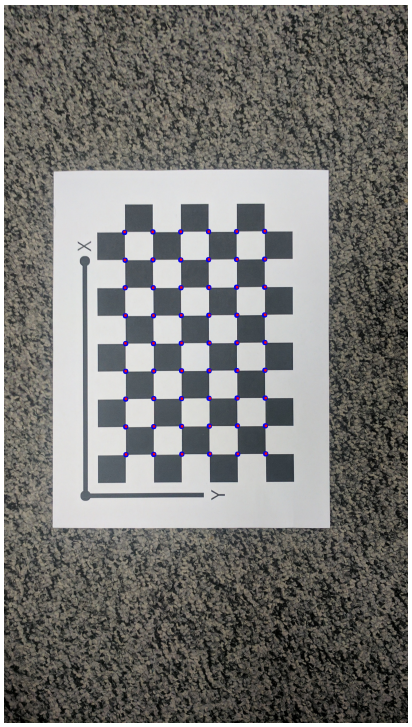

Fig. 6.

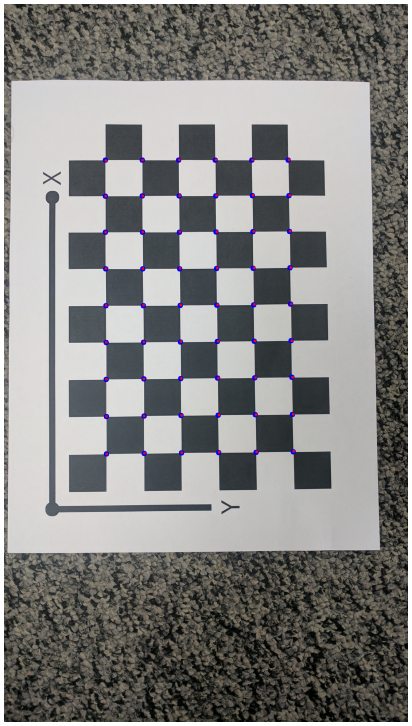

Fig. 7.

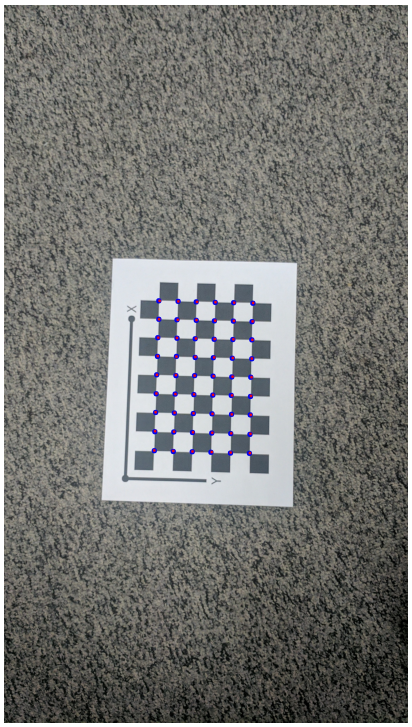

Fig. 8.

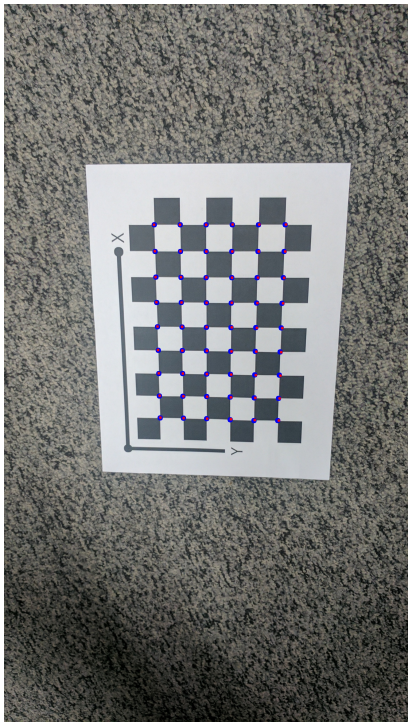

Fig. 9.

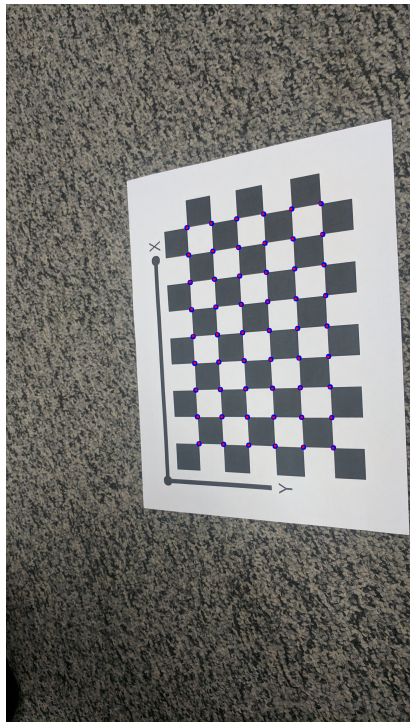

Fig. 10.

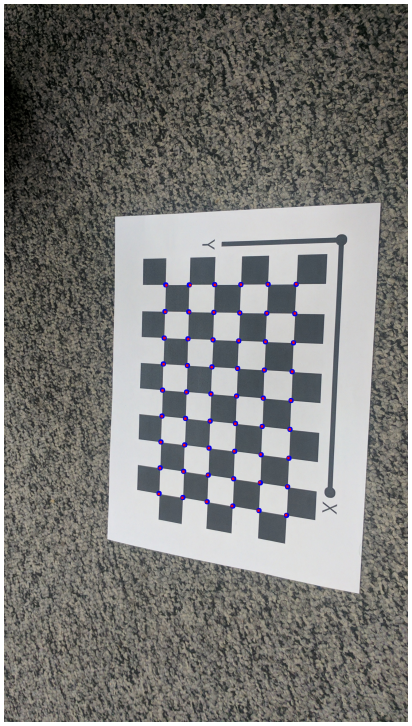

Fig. 11.

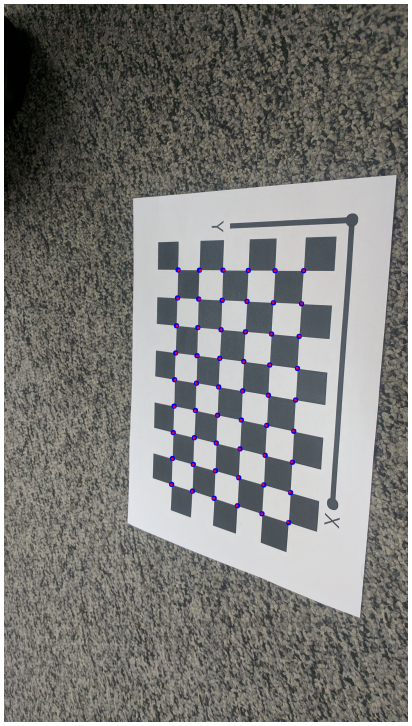

Fig. 12.

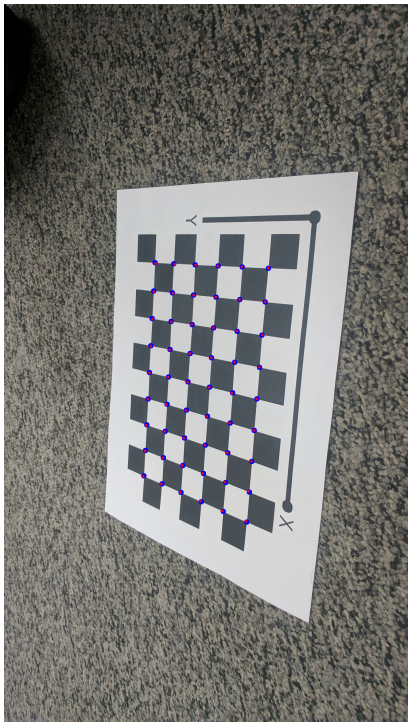

Fig. 13.

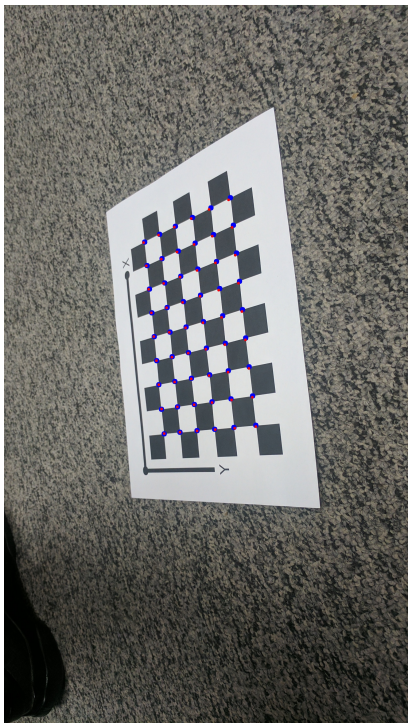

Fig. 14.