Homework 1: AutoCalib
RBE 459: Computer Vision

Riley Blair

School of Engineering
Worcester Polytechnic Institute
Email: rpblair@wpi.edu

***Abstract*—Before any meaningful data can be captured from cameras, the translation from world frame to the image frame must be calculated. Due to manufacturing errors and impressions in the methods for modeling a camera, the set of properties exist that reflect the translation differ from camera to camera. In this homework, we implement a robust algorithm for calculating the intrinsic camera parameters from a set of test images and report re-projection error after undistorting the images.**

I. Introduction

This process is based off of a 1998 paper from Zhengyou Zhang [1] whereby we calculate the intrinsic parameters K of a specific camera from a set of checkerboard images given to us. A checkerboard is used in this case as it has very prominent fiducials (corners) for which we can base our process off.

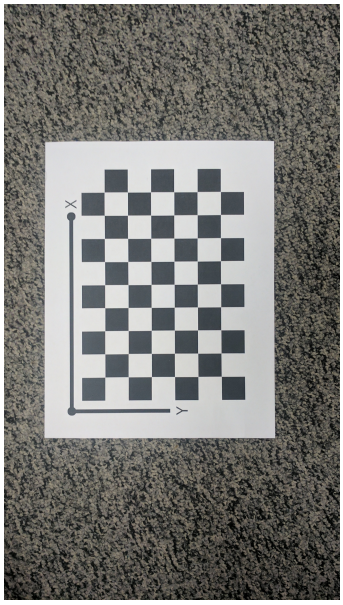

Fig. 1: Image with a Checkerboard pattern

The general road-map for calibrating goes as follows:

1) Estimate homography for each image
2) Use homographies for closed-form calculation of intrinsic matrix K
3) Estimate extrinsic matrices $[R \mid T]$ for each image
4) Use a non-linear optimizer for refining K estimation and distortion estimate k.

The results of the process are validated by calculating re-projection error on corners

A. Nomenclature

In this paper, there are several representations that need to be used for brevity. Most will be defined in their respective sections. But some universal ones are worth noting now.

The representation of a point in a given frame is given by:

$$\begin{aligned} M &= \begin{bmatrix} X \\ Y \end{bmatrix}, \text{ for points in the world plane} \\ m &= \begin{bmatrix} u \\ v \end{bmatrix}, \text{ for points in the image plane} \end{aligned} \tag{1}$$

These points are homogenized for later use and given the follow forms:

$$\tilde{M} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \quad \tilde{m} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2}$$

The intrinsic camera matrix is a 5-DOF matrix with the following form

$$K = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

The extrinsic matrix is represented by a 4x4 homogeneous-transformation matrix

$$T = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

B. Estimating Homography

The full transformation from the world frame to the image frame comes in the form

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

$$= \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{6}$$

For our case, all of the images are on the same depth (all $Z = 0$, making our equations simpler.

This means that each transformation from the word to image frame can be estimated as [1]

$$s\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{M}} \quad \text{with} \quad \mathbf{H} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}. \tag{7}$$

By finding a set of H matrices, we can estimate the intrinsic parameters, which are a constant factor between images.

This step begins by finding all of the corners from our checkerboard pattern and assigning them a label based on their grid position. This is accomplished by running the `cv2.findChessboardCorners()` method. I utilize an cv2 method `cv2.cornerSubPix()` to increase the precision of the found corner.

After finding the corners, we attempt to find each images relative homography via a non-linear optimization. We set up a system of equations to find an initial guess in the form [1]

$$\begin{bmatrix} \tilde{M_0}^T & \mathbf{0}^T & -u_0\tilde{M_0}^T \\ \mathbf{0}^T & \tilde{M_0}^T & -v_0\tilde{M_0}^T \\ \tilde{M_1}^T & \mathbf{0}^T & -u_1\tilde{M_1}^T \\ \mathbf{0}^T & \tilde{M_1}^T & -v_1\tilde{M_1}^T \\ \vdots & \vdots & \vdots \\ \tilde{M_n}^T & \mathbf{0}^T & -u_n\tilde{M_n}^T \\ \mathbf{0}^T & \tilde{M_n}^T & -v_n\tilde{M_n}^T \end{bmatrix} \mathbf{h} = \mathbf{0}. \tag{8}$$

for all n corners in the image. We use Singular-value decomposition to solve for our initial guess h.

We feed this initial guess into a non-linear optimizer `scipy.optimize.least_squares()` that utilizes the Levenberg-Marquardt Algorithm.

We optimize via reducing the Sum of Squared Differences (SSD) between m_i and $h * M_i$

C. Estimating Intrinsics

From the list of homographies of our checkerboard set, we can solve for a closed form solution to the camera intrinsics. This step follows a similar setup to the previous in that we set up a system of equations governed by

$$\begin{bmatrix} v_{12}(H_0)^T \\ (v_{11}(H_0) - v_{22}(H_0))^T \\ v_{12}(H_1)^T \\ (v_{11}(H_1) - v_{22}(H_1))^T \\ \vdots \\ v_{12}(H_n)^T \\ (v_{11}(H_n) - v_{22}(H_n))^T \end{bmatrix} \mathbf{b} = \mathbf{0}. \tag{9}$$

for all n homography estimates. v_{ij} is defined as

$$\mathbf{v}_{ij} = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix}^T. \tag{10}$$

for equation 6 [1]. We use SVD again to solve for b. We use this b matrix to find the 5 properties in our intrinsic matrix.

$$\begin{aligned} v_0 &= \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2} \\ \lambda &= B_{33} - \frac{B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})}{B_{11}} \\ \alpha &= \sqrt{\frac{\lambda}{B_{11}}} \\ \beta &= \sqrt{\frac{\lambda B_{11}}{B_{11}B_{22} - B_{12}^2}} \\ \gamma &= -\frac{B_{12}\alpha^2\beta}{\lambda} \\ u_0 &= \frac{\gamma v_0}{\beta} - \frac{B_{13}\alpha^2}{\lambda} \end{aligned}$$

D. Estimating Extrinsic Matrices

From section C we have generated the B matrix by way of

$$B = K^{-T} * K^{-1} \tag{11}$$

The extrinsic params can them be found by way of rearranging equation 6.

$$\begin{aligned} r_1 &= \lambda * K^{-1} * h_1 \\ r_2 &= \lambda * K^{-1} * h_1 \\ r_3 &= r_1 \times r_2 \\ t &= \lambda * K^{-1} * h_3 \end{aligned}$$

where

$$\lambda = \frac{1}{||K^{-1} * h_2||}$$

and

$$T = [r_1, r_2, r_3, t]$$

for each h in our homography list.

E. Non-linear optimization on System

The closed form solution for K is a good start, but does not account for radial distortion, or generate optimal values for the intrinsic matrix K. For this, we can formulate a non-linear optimization problem to minimize the re-projection error of the system.

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \left\|\mathbf{m}_{ij} - \hat{\mathbf{m}}\left(\mathbf{K}, k, T, M_j\right)\right\|^2, \tag{12}$$

As K dominates k values, we can use $k = [0, 0]$ as an initial guess, and feed the loss function into `scipy.optimize.least_squares()`.

II. Results

During the allotted time, we were unable to generate values for the radial distortion that did not break the system, so we used a $k = [0, 0]$ for our distortion parameter.

After running steps 1-3, we found the following K parameters

$$K = \begin{bmatrix} 2064.345 & 0.215 & 762.554 \\ 0 & 2045.158 & 1337.639 \\ 0 & 0 & 1 \end{bmatrix} \quad (13)$$

The outputs from our pipeline were tested by calculating re-projection error from the world frame to the image frame. The process for doing this goes as follows:

1) Project $\tilde{M}$ by multiplying with the extrinsic matrix $[R \mid T]$
   $M_{proj} = [R \mid T] * \tilde{M}$
2) Normalize against the Z output
   $m_X^{norm} = M_X^{Proj}/M_Z^{Proj}, m_Y^{norm} = M_Y^{Proj}/M_Z^{Proj}$
3) adjust for radial distortion using k
4) Error is the l2 norm $||m - m_{norm}||_2$

Doing this for each corner of each image gives us the total error of the system. Dividing by the number of images gives us the mean re-projection for the given image.

The mean re-projection error without optimization was **37.681**.

We then ran our non-linear optimizer during step 1 for finding better homography estimations after SVD. The following parameters were found after running this optimization step.

$$K_{optim} = \begin{bmatrix} 2063.656 & 0.174 & 730.309 \\ 0 & 2055.838 & 1401.694 \\ 0 & 0 & 1 \end{bmatrix}, \quad (14)$$

The mean re-projection error after optimization was **39.406**. Using these parameters, we undistorted each of the input images to generate the before/after. All of the images can be found in appendix A

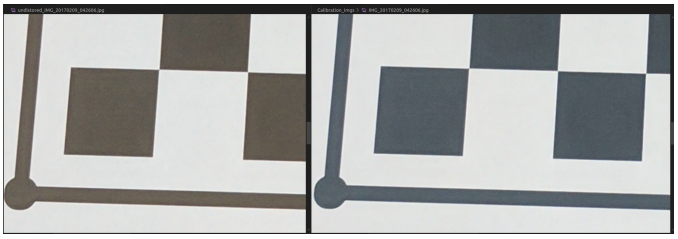

Fig. 2: Close-up comparison of an undistorted (left) and original (right) checkerboard square

A. Improvements

The big failure in this report was being unable to find valid values for k. The major future work includes generating a proper k estimation by using non-linear optimization on the system via the steps in Section E.

One future improvement would be normalizing the data in the homography estimation step. One potential source of inaccuracy is the fact that the m is very large in comparison to M, which propagates inaccuracies when using SVD+optimization.

Additionally, the optimization step after SVD in step 1 seemed to generates values with greater error. The result suggests an error in our formulation, and in future iterations we would seek to fix this.

B. Conclusion

In this homework we implemented step-by-step Zhang's camera calibration algorithm for a Google Pixel XL. We then analyzed the results of re-projection using our initial guess and optimized solution.

References

[1] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

III. Appendix

A. Appendix A: Undistorted Image results

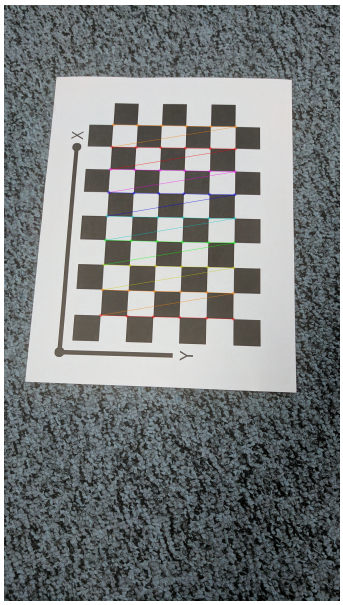

Fig. 3: Image 606 undistorted

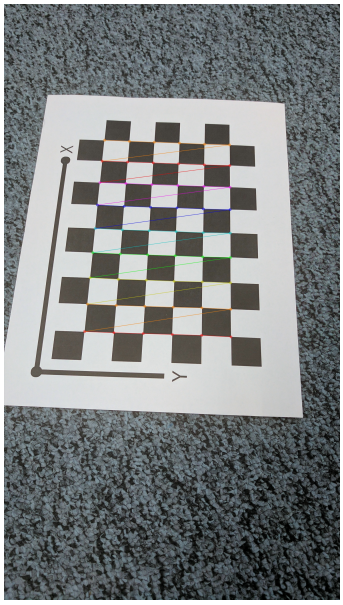

Fig. 4: Image 608 undistorted

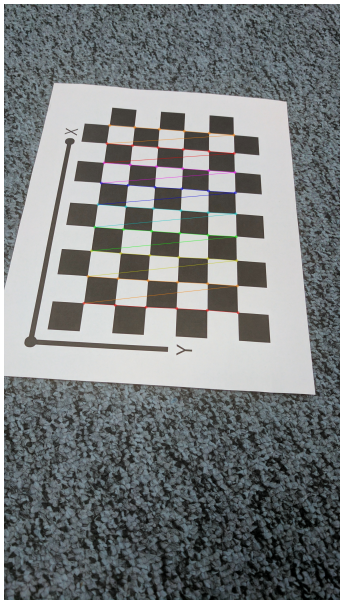

Fig. 5: Image 610 undistorted

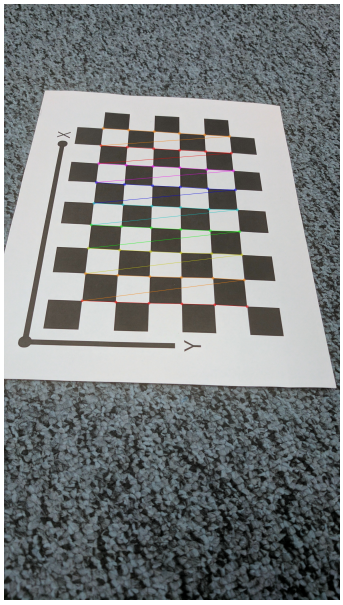

Fig. 6: Image 612 undistorted

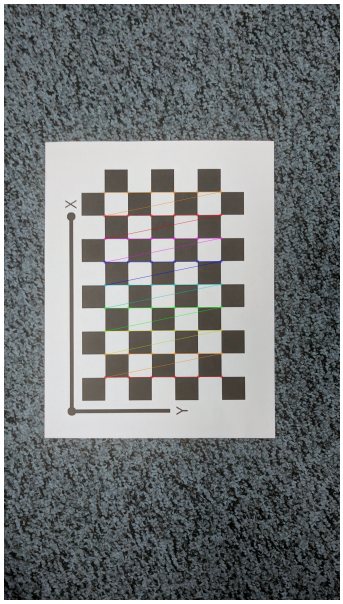

Fig. 7: Image 614 undistorted

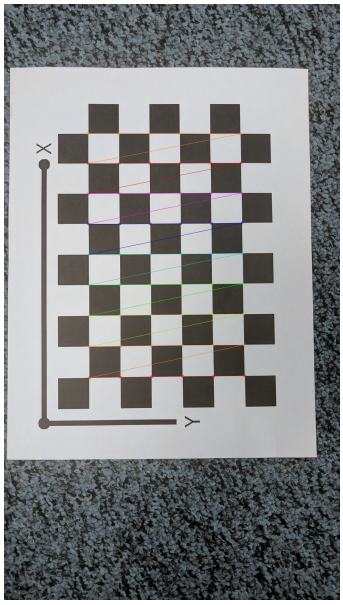

Fig. 8: Image 616 undistorted

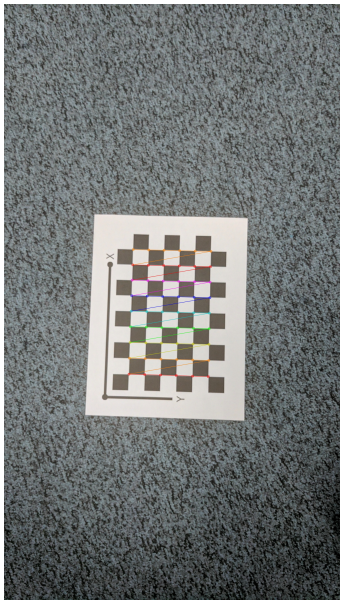

Fig. 9: Image 619 undistorted

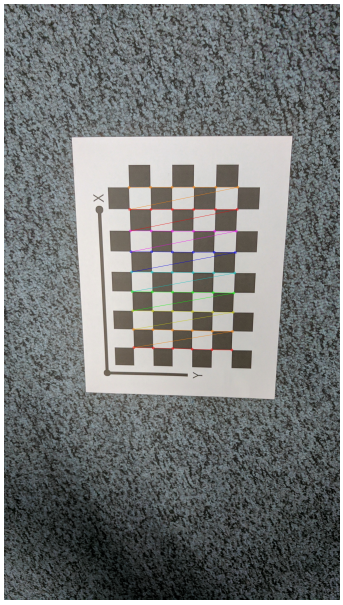

Fig. 10: Image 621 undistorted

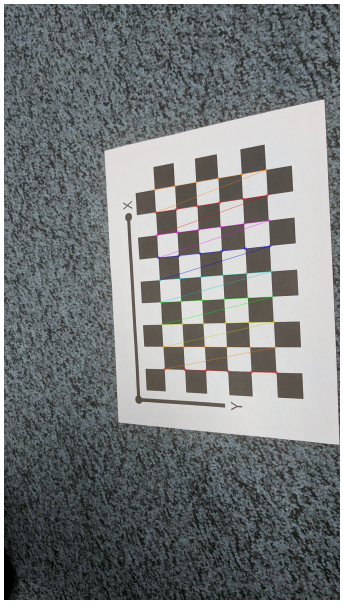

Fig. 11: Image 624 undistorted

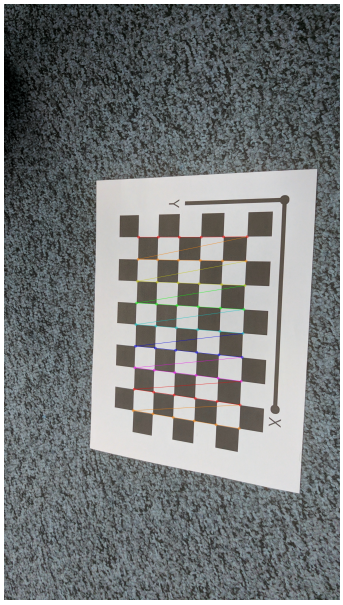

Fig. 12: Image 627 undistorted

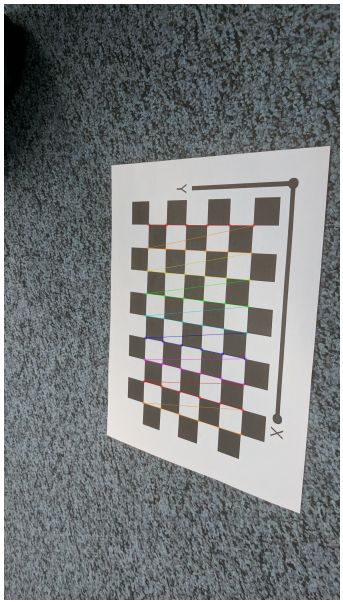

Fig. 13: Image 629 undistorted

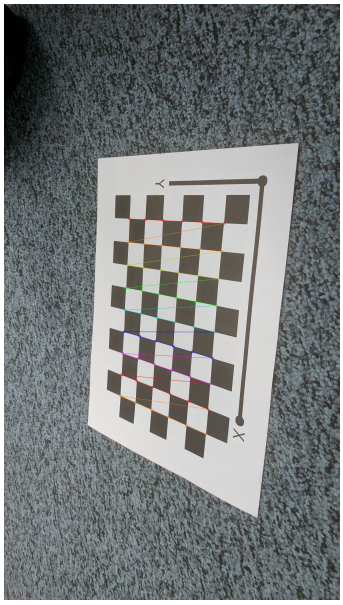

Fig. 14: Image 630 undistorted

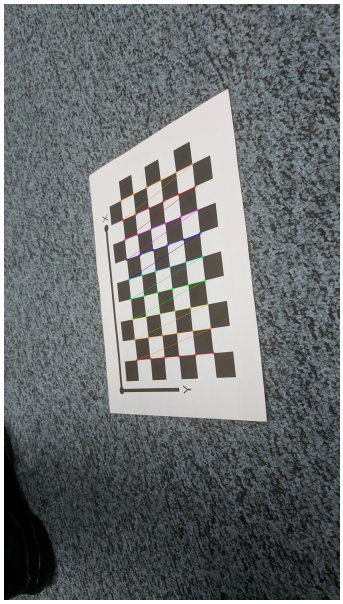

Fig. 15: Image 634 undistorted