RBE/CS 549 - HW1: AutoCalib

Using 1 Late Day

Blake Bruell
Worcester Polytechnic Institute
Worcester, Massachusetts 01609
babruell@wpi.edu

***Abstract*—In this paper the robust technique for automatic camera calibration proposed in "A Flexible New Technique for Camera Calibration" by Zhang is explained step by step and tested on a dataset. The technique uses a set of images of a planar pattern at different (at least 2) orientations. Homographies for each image are approximated with a closed form solution, and then minimized. These homographies are used to calculate camera intrinsics and extrinsics using a closed form solution, and then the intrinsics alongside distortion parameters are optimized using the Levenberg-Marquardt algorithm across all images simultaneously.**

I. Introduction

A. Perpective Projection Model

A camera captures a set of 3D world points (or model points) via a 2D sensor yielding 2D coordinates (or image points), and thus a projection between these two coordinates is required. A 2d image point is denoted $\mathbf{x} = (x, y)^\top$, and its homogeneous representation is $\bar{\mathbf{x}} = (x, y, 1)^\top$. A 3D world point is denoted by $\mathbf{X} = (X, Y, Z)^\top$, and its homogeneous representation is $\bar{\mathbf{X}} = (X, Y, Z, 1)^\top$. This transformation is modeled via a pinhole camera, which gives the following [1]:

$$s\bar{\mathbf{x}} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\bar{\mathbf{X}} \tag{1}$$

where s as an arbitrary scale factor, $\mathbf{K}$ is the camera intrinsics matrix, and $\mathbf{R}$ and $\mathbf{t}$ are rotation and translation transforms which relate the world coordinates to the camera coordiantes, referred to as the camera extrinsics.

For our use case, without loss of generality, we can assume that the model plane in our image lies with $Z = 0$ in the world coordinates, and so we can simplify the right hand side of Equation 1:

$$s\bar{\mathbf{x}} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\0\\1\end{bmatrix} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\1\end{bmatrix}$$

where $\mathbf{r}_i$ denotes the i^{th} column of $\mathbf{R}$. From here on out $\mathbf{X}$ refers to the simplified 3D world point with $Z = 0$. We define the homography between the model plane and the image as follows:

$$\mathbf{H} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix} \tag{2}$$

An homography can be estimated directly from an image of a known planar pattern, using the model points of the planar pattern and the image points detected from the image. From here the goal of camera calibration is extract the camera intrinsics $\mathbf{K}$ from the homographies for a given set of images.

B. Homography Estimation

The homography estimation from a set of image points and corresponding is performed by minimizing the following equation

$$\sum_j ||\mathbf{x}_j - \hat{\mathbf{x}}_j||^2$$

where

$$\hat{\mathbf{x}}_j = \frac{1}{\bar{\mathbf{h}}_3^\top \mathbf{X}_j}\begin{bmatrix}\bar{\mathbf{h}}_1^\top \mathbf{X}_j\\ \bar{\mathbf{h}}_2^\top \mathbf{X}_j\end{bmatrix}$$

with $\bar{\mathbf{h}}_i$, the i^{th} row of $\mathbf{H}$ [1]. This is done using the Levenberg-Marquardt algorithm, which is performed in code using `optimize.least_squares` from `scipy`, with the parameter `method='lm'`.

An initial guess is obtained by solving

$$\begin{bmatrix}\bar{\mathbf{X}}^\top & \mathbf{0}^\top & -u\bar{\mathbf{X}}^\top\\ \mathbf{0}^\top & \bar{\mathbf{X}}^\top & -u\bar{\mathbf{X}}^\top\end{bmatrix}\mathbf{a} = 0$$

with $\mathbf{a} = \left(\bar{\mathbf{h}}_1^\top, \bar{\mathbf{h}}_2^\top, \bar{\mathbf{h}}_3^\top\right)^\top$ using singular value decomposition.

The above method yields poor results as the inputs aren't normalized, and so a x and y values are normalized using a normalization matrix on the homogeneous coordinates.

II. Camera Calibration Method

This section details the method for recovering camera intrinsic and distortion parameters from a set of N model points and a set of M images each with N images points.

A. Camera Intrinsics

The camera intrinsics matrix $\mathbf{K}$ consists of 5 free variables: the coordinates of the principle point, denoted (u_0, v_0), the scale factors on the u and v axes, denoted α and β respectively, and the skew factor, denoted γ [1].

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

Applying the knowledge that $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormal, we get

$$\mathbf{h}_1^\top \mathbf{K}^{-\top}\mathbf{K}^{-1}\mathbf{h}_2 = 0 \tag{4}$$

$$\mathbf{h}_1^\top \mathbf{K}^{-\top}\mathbf{K}^{-1}\mathbf{h}_1 = \mathbf{h}_2^\top \mathbf{K}^{-\top}\mathbf{K}^{-1}\mathbf{h}_2 \tag{5}$$

where $\mathbf{h}_i$ denotes the i^{th} column of $\mathbf{H}$ [1].

We can then get a closed form solution for $\mathbf{K}$. Let

$$\mathbf{B} = \mathbf{K}^{-\top}\mathbf{K}^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{6}$$

$\mathbf{B}$ is a symmetric matrix, and thus is defined by the 6D vector

$$\mathbf{b} = \begin{bmatrix} B_{11} & B_{12} & B_{22} & B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{7}$$

which gives

$$\mathbf{h}_i^\top \mathbf{B} \mathbf{h} j = v_{ij}^\top \mathbf{b} \tag{8}$$

with

$$\begin{aligned} \mathbf{v}_{ij} = [&h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, \\ &h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}] \end{aligned}$$

[1]. Finally, this can be rewritten in in the form

$$\begin{bmatrix} \mathbf{v}_{12}^\top \\ (\mathbf{v}_{11} - \mathbf{v}_{22}^\top \end{bmatrix} \mathbf{b} = \mathbf{0} \tag{9}$$

Stacking this for M images of the model plane we get

$$\mathbf{V}\mathbf{b} = \mathbf{0} \tag{10}$$

where $\mathbf{V}$ is a $2M \times 6$ matrix [1]. We can solve for $\mathbf{b}$ using singular value decomposition, finding the right singular vector of $\mathbf{V}$ associated with the smallest singular value. Once $\mathbf{b}$ is estimated, $\mathbf{K}$ can be recovered trivially [1].

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

B. Camera Extrinsics

Once the camera intrinsics are estimated, the extrinsics can easily be recovered. From (2), we have

$$\begin{aligned} \mathbf{r}_1 &= \lambda \mathbf{K}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda \mathbf{K}^{-1}\mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda \mathbf{K}^{-1}\mathbf{h}_3 \end{aligned}$$

with $\lambda = 1/||\mathbf{K}^{-1}\mathbf{h}_1|| = 1/||\mathbf{K}^{-1}\mathbf{h}_2||$.

Due to noise, $\mathbf{R}$ will not satisfy the properties of a real rotation matrix, and so a real rotation matrix is derived using singular value decomposition. SVD yields $\mathbf{U}$, $\mathbf{S}$, and $\mathbf{V}$, and $\mathbf{R}_{real} = \mathbf{U}\mathbf{V}$.

C. Radial Distortion

The process up to this point has assumed that the camera has no distortion, but this is not accurate. Most cameras exhibit large distortion, particularly radial distortion. For our purposes, only the first two terms of distortion are considered.

Let $\mathbf{x} = (x, y)$ be the ideal (without distortion) normalized image coordinates, and $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y})$ be the distorted image coordinates. The ideal points are the projection of the model points according to the pinhole model.

$$\begin{aligned} r^2 &= x^2 + y^2 \\ D &= k_1 r^2 + k_2 r^4 \\ \tilde{\mathbf{x}} &= \mathbf{x}(1 + D) \end{aligned} \tag{12}$$

where k_1 and k_2 are the coefficients of the radial distortion. This follows as the center of the radial distortion is coincident with the principal point (u_0, v_0).

These distortion equations are applied to the model points while in normalized coordinate space, and then the distorted model points are mapped into the pixel space. This simplifies the calculation of the distortion.

This gives the final form for our projection from model points $\mathbf{X}_j$ to pixel coordinates $\mathbf{u}_j$. Let

$$\mathbf{R} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

Then,

$$\mathbf{x}_j = \frac{1}{\bar{\mathbf{R}}_3^\top \mathbf{X}_j} \begin{bmatrix} \bar{\mathbf{R}}_1^\top \mathbf{X}_j \\ \bar{\mathbf{R}}_2^\top \mathbf{X}_j \end{bmatrix} \tag{13}$$

we apply the distortion to get $\tilde{\mathbf{x}}_j$, and finally in pixel coordinates

$$\mathbf{u}_j = \begin{bmatrix} \bar{\mathbf{K}}_1 \tilde{\mathbf{x}}_j \\ \bar{\mathbf{K}}_2 \tilde{\mathbf{x}}_j \end{bmatrix} \tag{14}$$

where $\bar{\mathbf{K}}_i$ is the i^{th} row of $\mathbf{K}$ [2].

While closed form methods exist to approximate k_1 and k_2, $k_1 = k_2 = 0$ is assumed, as it is sufficiently accurate to permit the optimization to converge.

D. Projection Error

In order to optimize the defined parameters, an error function must be defined. Recall that $\mathbf{K}$ is the camera intrinsic matrix, k_1 and k_2 distortion parameters, $\mathbf{R}_i$ and $\mathbf{t}_i$ are the rotation and translations extrinsics for the i^{th} image, and $\mathbf{X}_j$ is the j^{th} model point. [2].

$$Error = \sum_{i=1}^{M} \sum_{j=1}^{N} ||\mathbf{x}_{ij} - P(\mathbf{K}, k_1, k_2, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)||^2 \tag{15}$$

where P is the projection of a point according to Equations 13 and 14.

E. Parameterizing the extrinsic rotation matrices

Each rotation matrix $\mathbf{R}_i$ consists of 9 elements, but a rotation matrix only has 3 degrees of freedom. To address this issue, the Rodrigues representation of a rotation matrix is used, which is denoted as ρ, and is a 3D vector.

F. Optimization

In order to improve our estimation of the camera intrinsics we would like to optimize our total projection error. As the problem is non-linear, iterative methods must be used, in our case the Levenberg-Marquardt algorithm. To apply this optimization we must combine parameters for all images into one vector $\mathcal{P}$.

$$\mathbf{a} = (\alpha, \beta, \gamma, u_0, v_0, k_1, k_2)$$
$$\mathbf{w}_i = (\rho_{i,x}, \rho_{i,y}, \rho_{i,z}, t_{i,x}, t_{i,y}, t_{i,z})$$
$$\mathcal{P} = (\mathbf{a}^\top | \mathbf{w}_1^\top | \dots | \mathbf{w}_M^\top)^\top$$

$\mathbf{a}$ contains all the camera parameters, and $\mathbf{w}_i$ the transformation data associated with the i^{th} image. $\mathcal{P}$ is a $(7 + 6M)$ vector, and it is then optimized with respect to the error previously defined. This is performed using `optimize.least_squares` from `scipy`, with the parameter `method='lm'`.

III. Results

The results of the outlined method were applied to a test set of 13 images. This section explains how the technique was applied to the images, as well as presents the resulting camera intrinsic and errors.

A. Input Data

The input data for the method described is a set of model points, $(N \times 2)$, and a set of image points, $(M \times N \times 2)$. To get this data, a grid size of $(9, 6)$ was chosen, and model points were generated using the a grid tile size of 1 (this choice is arbitrary). Finally, `cv2.findChessboardCorners` was used to find the $(9, 6)$ grid of image points for each image, as shown in Figure 1. The output for all images is shown in Appendix A.

Image	Mean Geometric Error	
	Unoptimized	**Optimized**
0	0.81	0.537
1	1.916	0.53
2	2.678	0.789
3	1.341	0.782
4	5.272	0.792
5	1.878	0.541
6	1.163	0.423
7	1.959	0.473
8	2.109	0.704
9	2.366	0.926
10	1.164	0.649
11	1.217	0.805
12	1.705	0.587
Mean	**1.968**	**0.657**

TABLE I: Reprojection Error

B. Camera Intrinsics

The calibration matrix for the dataset, before and after optimization, are as follows:

$$\mathbf{K} = \begin{bmatrix} 2052.789 & -0.37 & 763.06 \\ 0 & 2036.635 & 1352.614 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{K_{opt}} = \begin{bmatrix} 2045.677 & -1.594 & 759.524 \\ 0 & 2037.36 & 1345.386 \\ 0 & 0 & 1 \end{bmatrix}$$

and the distortion parameters:

$$\mathbf{k} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$
$$\mathbf{k_{opt}} = \begin{bmatrix} 0.172 & -0.737 \end{bmatrix}$$

. The error for each image as well as overall is shown in Table I. The final rectified image with re-projected corners plotted is shown in Figure 2. The output for all images is shown in Appendix B.

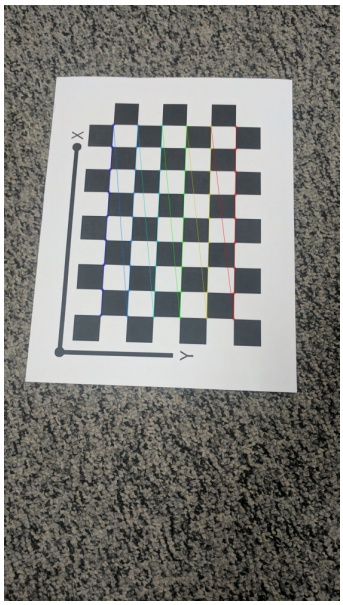

Fig. 1: **Checkerboard Corners**

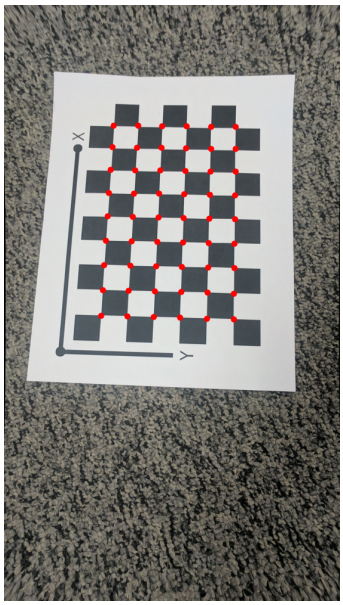

Fig. 2: **Rectified Image with Re-projected Corners**

References

[1] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000. DOI: 10.1109/34.888718.

[2] W. Burger, "Zhang's camera calibration algorithm: In-depth tutorial and implementation," University of Applied Sciences Upper Austria, School of Informatics, Communications and Media, Dept. of Digital Media, Hagenberg, Austria, Tech. Rep. HGB16-05, May 2016. [Online]. Available: https://www.researchgate.net/publication/303233579_Zhang%E2%80%99%20s_Camera_Calibration_Algorithm_InDepth_Tutorial_and_Implementation.

Appendix A
Corner Detection

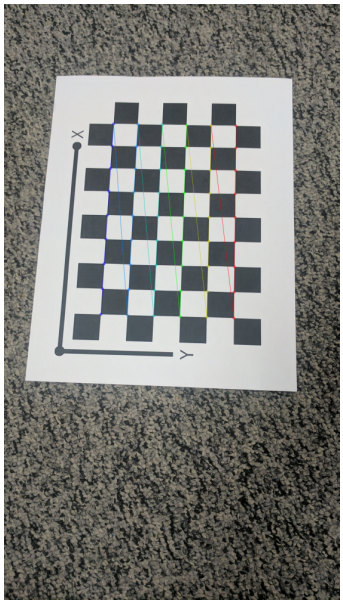

Fig. 3: **Image 0: Checkerboard Corners**

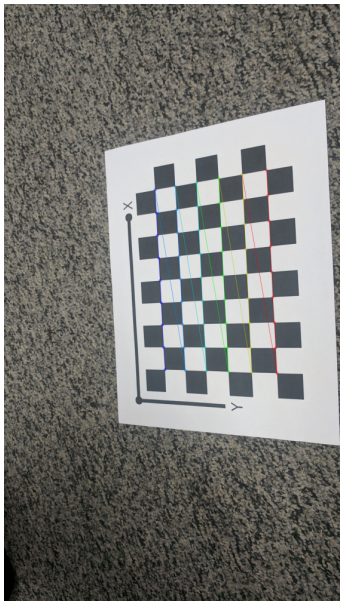

Fig. 4: **Image 1: Checkerboard Corners**

Fig. 5: **Image 2: Checkerboard Corners**

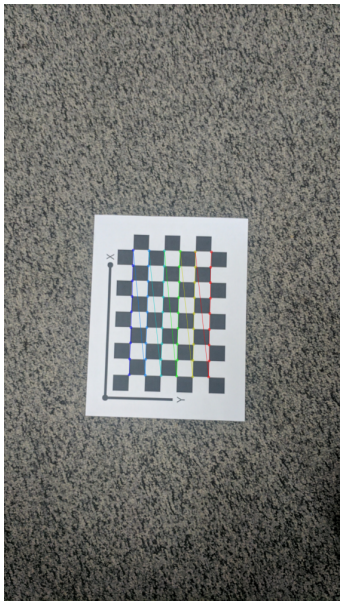

Fig. 6: **Image 3: Checkerboard Corners**

Fig. 7: **Image 4: Checkerboard Corners**

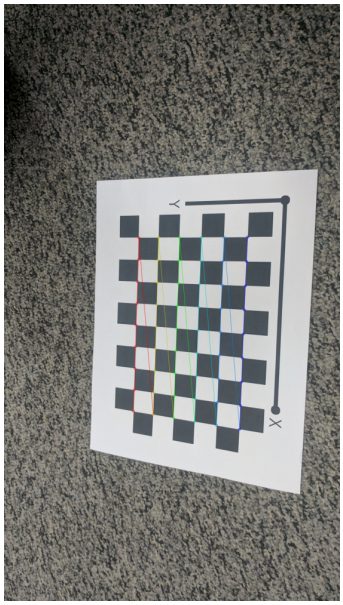

Fig. 8: **Image 5: Checkerboard Corners**

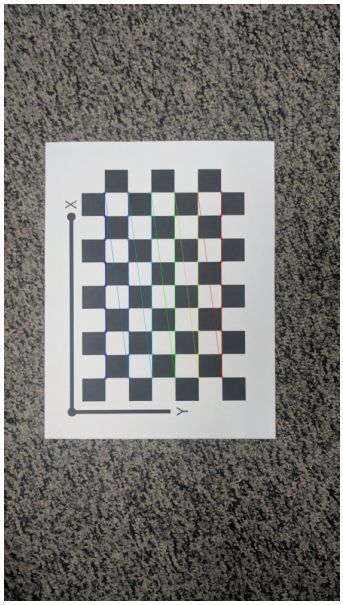

Fig. 9: **Image 6: Checkerboard Corners**

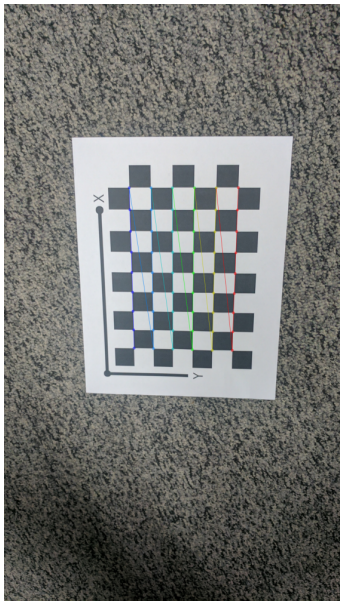

Fig. 10: **Image 7: Checkerboard Corners**

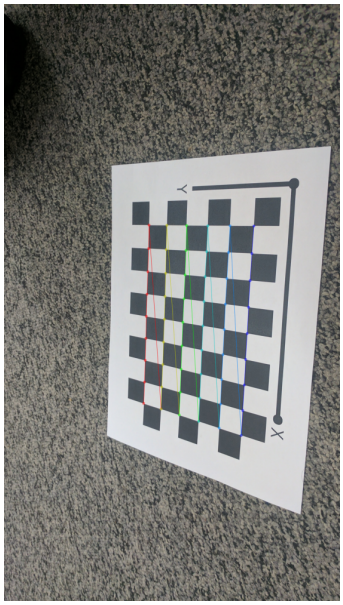

Fig. 11: **Image 8: Checkerboard Corners**

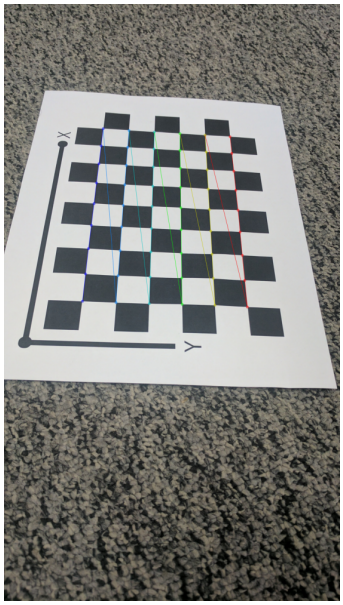

Fig. 12: **Image 9: Checkerboard Corners**

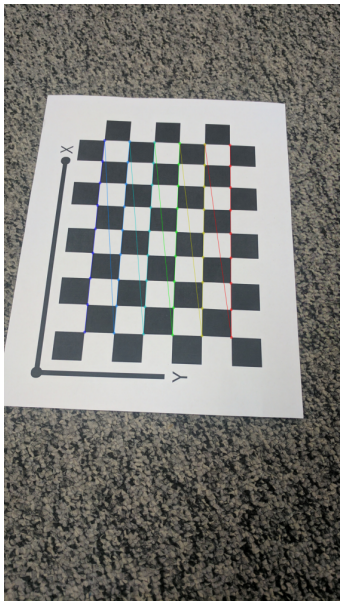

Fig. 13: **Image 10: Checkerboard Corners**

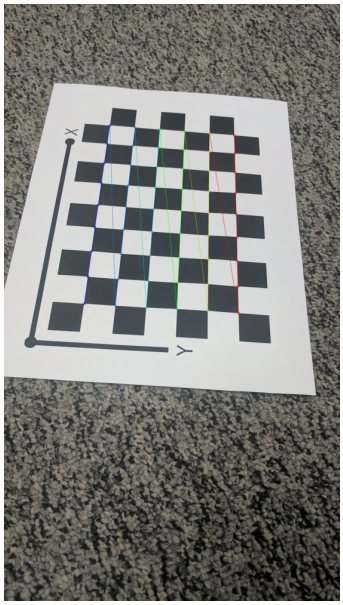

Fig. 14: **Image 11: Checkerboard Corners**

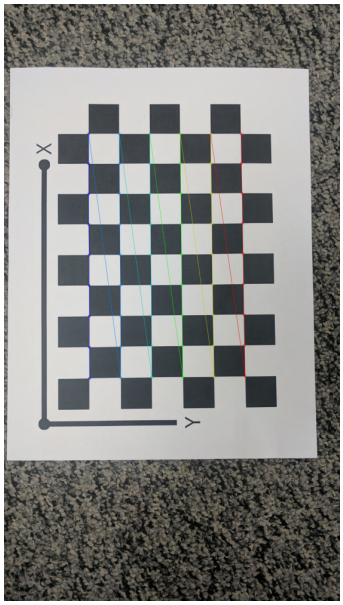

Fig. 15: **Image 12: Checkerboard Corners**

APPENDIX B
RE-PROJECTION

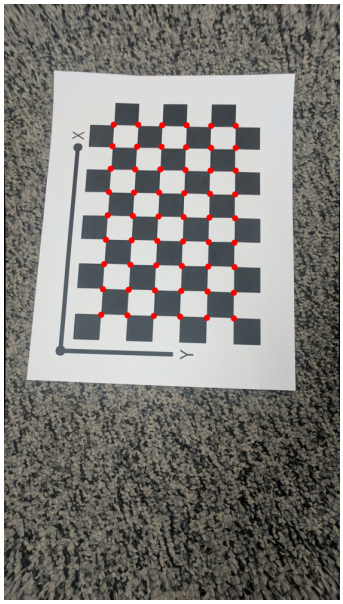

Fig. 16: **Image 0: Rectified and with Reprojected Corners**

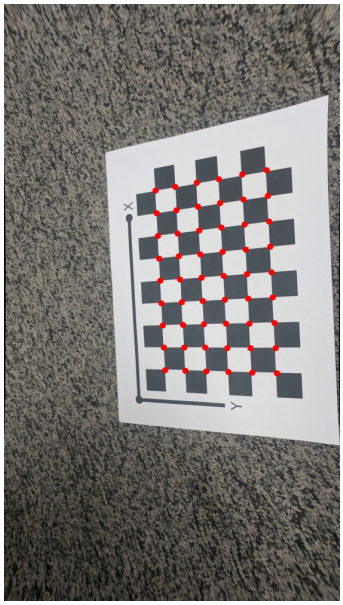

Fig. 17: **Image 1: Rectified and with Reprojected Corners**

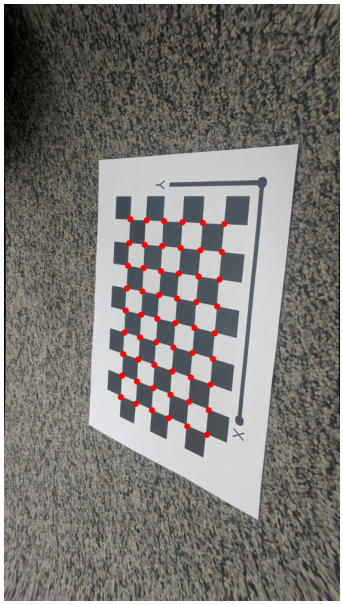

Fig. 18: **Image 2: Rectified and with Reprojected Corners**

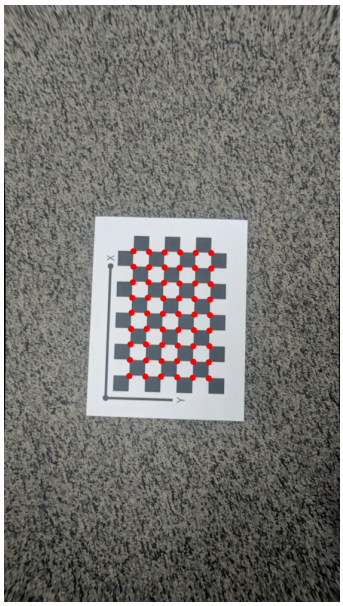

Fig. 19: **Image 3: Rectified and with Reprojected Corners**

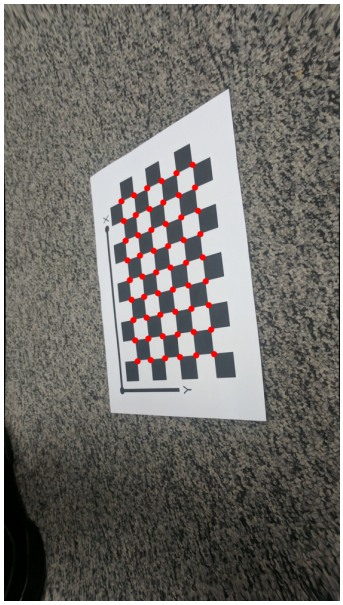

Fig. 20: **Image 4: Rectified and with Reprojected Corners**

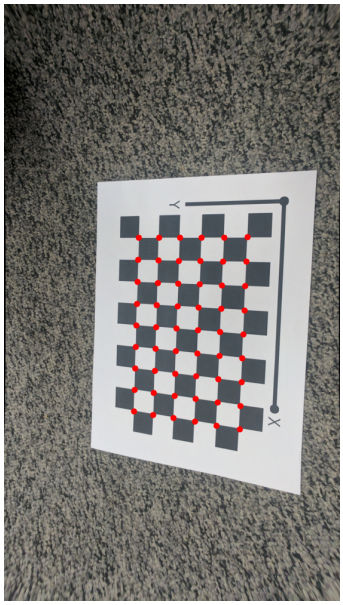

Fig. 21: **Image 5: Rectified and with Reprojected Corners**

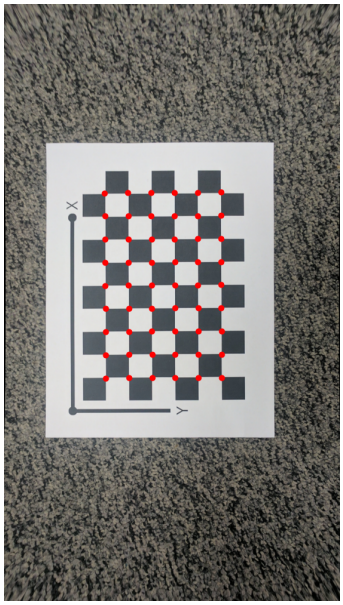

Fig. 22: **Image 6: Rectified and with Reprojected Corners**

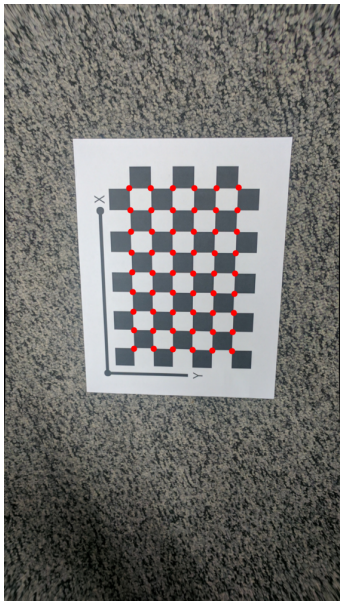

Fig. 23: **Image 7: Rectified and with Reprojected Corners**

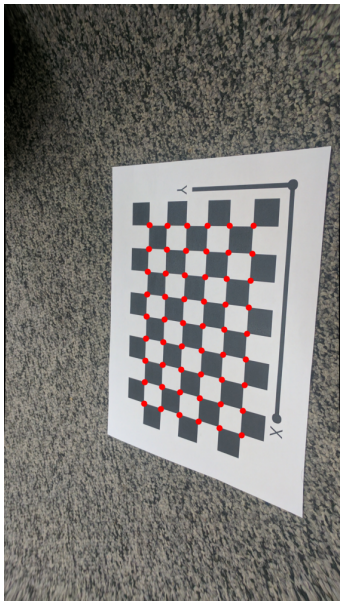

Fig. 24: **Image 8: Rectified and with Reprojected Corners**

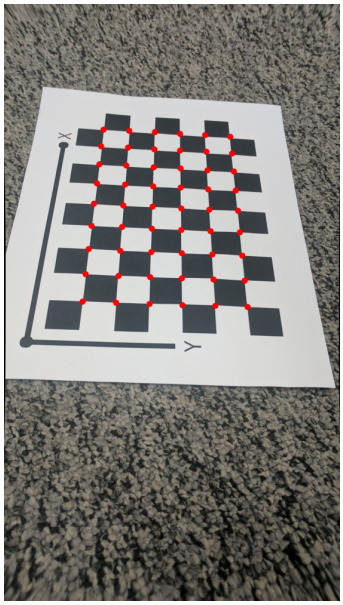

Fig. 25: **Image 9: Rectified and with Reprojected Corners**

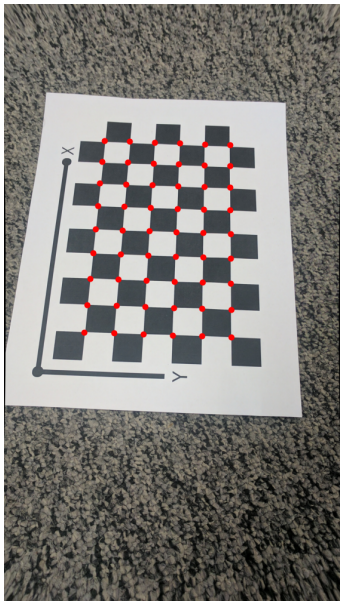

Fig. 26: **Image 10: Rectified and with Reprojected Corners**

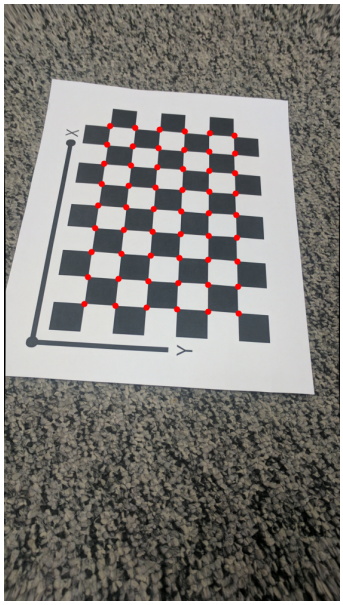

Fig. 27: **Image 11: Rectified and with Reprojected Corners**

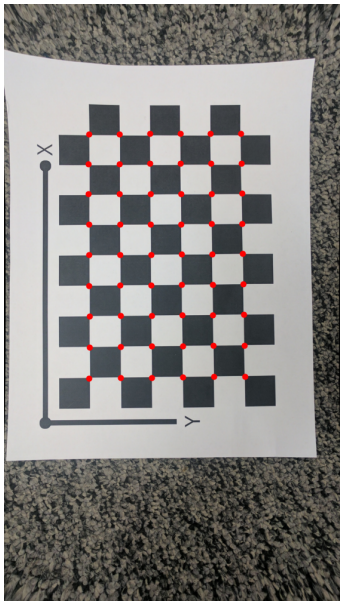

Fig. 28: **Image 12: Rectified and with Reprojected Corners**