HW1: AutoCalib (Automatic Camera Calibration) RBE549

Karter Krueger
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA 01609
Email: kkrueger2@wpi.edu

I. AutoCalibration

For this homework, we implemented a Zhang's method [0] of automatically calibrating a camera based on a set of images to find both the intrinsic and the radial distortion properties of the lens.

This method is composed of several steps to accurately determine the calibration parameters.

A. Step 1: Find the Corners

To calibrate the camera of this Google Pixel XL, we use the provided 13 images of a 6x9 checkerboard with squares that measure 21.5mm in size. Each image is from a different perspective and different position to provide a variety of viewpoints to optimize over. Placing the pattern in different areas of the view plane also provides for a more accurate estimation of the k distortion values as the radial model can better fit from examples near the edges where the distortion is more severe. We use the OpenCV function `cv2.findChessboardCorners(img, GRID_SIZE)` to detect the corners of each cell, placed in an array with consistent ordering. We refer to these points as m in homogeneous form with the last column as ones. If visualized, the detected corners look like the below image in Fig. 1.

In addition to detecting the real corners on the image, we must also generate a set of "model points" based on the known 21.5mm spacing of the squares. We refer to this matrix as M in homogeneous form with the last column as ones.

B. Step 2: Estimate Homographies

For each image i, we estimate a homography matrix H_i that describes a relationship between the set image points m_i and model points M. The homography is estimated based on the eigenvector corresponding to the minimum eigenvalue, extracted using the SVD of the matrix $(L.T * L)$ where L is composed of linear equations that relate the points in M and (u, v) coordinates of points in m_i, using the standard method to solve a homography.

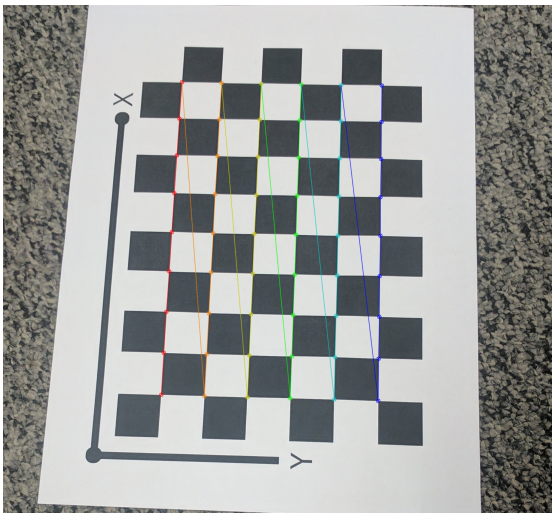

Fig. 1. A view of the Chessboard corners rendered on an image using OpenCV

C. Step 3: Solve For The Camera Intrinsic Matrix

Based on the homographies in step 2, we generated values using the method outlined in Zhang's paper to generate V. We then solve for the linear system of equations $Vb = 0$ using SVD on $V.T * V$ where we again get b from the eigenvector corresponding to the smallest eigenvalue. A closed form solution highlighted in Zhang's paper is then used to solve for the intrinsic matrix K composed of alpha, beta, u0, v0, gamma, and the scale.

D. Step 4: Estimate Rotation and Translation For Each Image

Now that we have an estimate of the camera intrinsic matrix, we can loop back over the homographies to determine the rotation and translation values that describe the relation between the chessboard and camera in each images.

E. Step 5: Optimize Parameters

In the final step, we use `scipy.optimize.minimize` to minimize the reprojection error of the model points onto the the camera plane. We intitalize the optimizer with an initial

estimate $x0$ based on the earlier results of the closedform solution. We define the error function as the sum of errors across all images, with each image's error determined by a sum of squared differences formula applied to the differences between the image corner points m_i and reprojected model points.

Model points are first rotated to the camera perspective using the R and T values from step 4. Then we calculated radius r values to be used later in distortion correction. Next we multiply the rotated model points by the intrinsic matrix K. Lastly we apply the distortion values k_1 and k_2 to the model points to make them more similar to the real image points that we will soon compare with.

Finally, we take calculate the error for the image by summing the squared values of $m_i - M_distorted$.

F. Results:

Finally, we have determined both the intrinsic matrix K and the radial distortion parameters k_1 and k_2.

The final K matrix, postoptimization, is provided below:

$$\begin{bmatrix} 2051.49 & 0.0 & 763.97 \\ 0.0 & 2033.98 & 134.82 \\ 0 & 0 & 1.0 \end{bmatrix}$$

The final results for distortion were
$k1 = 0.0709$ and
$k2 = -0.386$.

The resulting images below have been rectified to correct the camera distortion and the reprojected corners (with distortion) have been rendered as green dots on the images. You will see in the below figures that the reprojected points are very close to the square intersections and that the corners of the images have been slightly warped in line with the k values.

The final reprojection error as a sum across all images was 468.71. Across 13 images, this comes out to an average sum of squared distance error of 36 per image. Given that there are 54 points in each image on the board, this is a norm error of 0.6mm per corner.

My error sums per image are also listed below:

$$\begin{bmatrix} 21.56 \\ 31.02 \\ 45.72 \\ 64.20 \\ 16.26 \\ 24.79 \\ 47.34 \\ 18.29 \\ 24.75 \\ 23.92 \\ 42.81 \\ 61.60 \\ 46.39 \end{bmatrix}$$

G. References

Zhang, Z. (2000). A flexible new technique for camera calibration. IEEE Transactions on pattern analysis and machine intelligence, 22(11), 1330-1334.

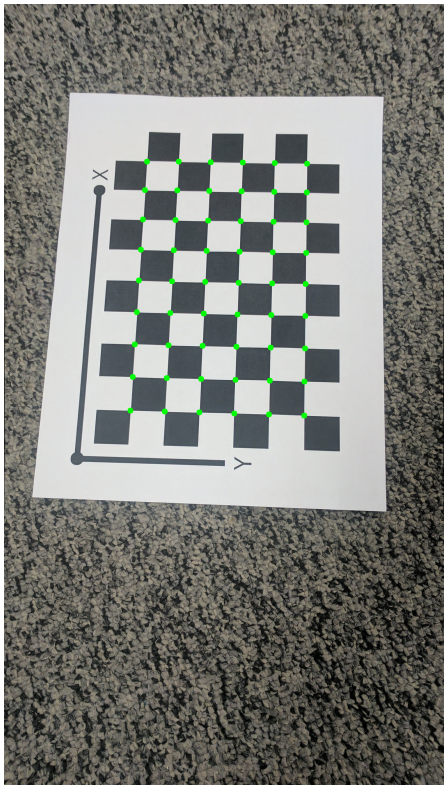

Fig. 2. Rectified IMG 20170209 042606 with reprojected model points

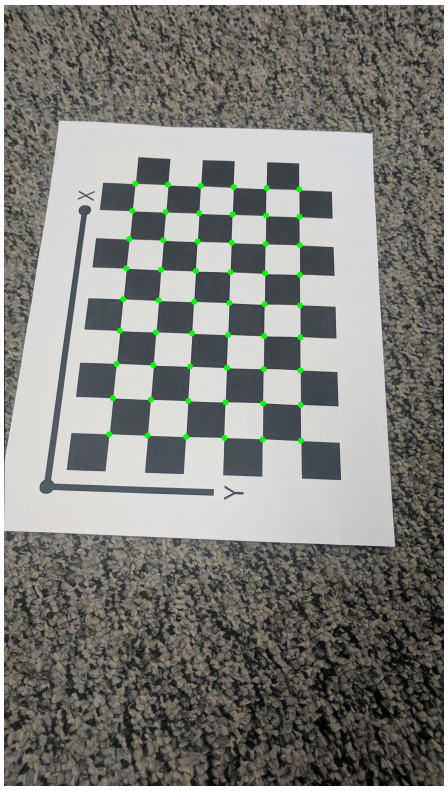

Fig. 3. Rectified Img 20170209 042608 with reprojected model points

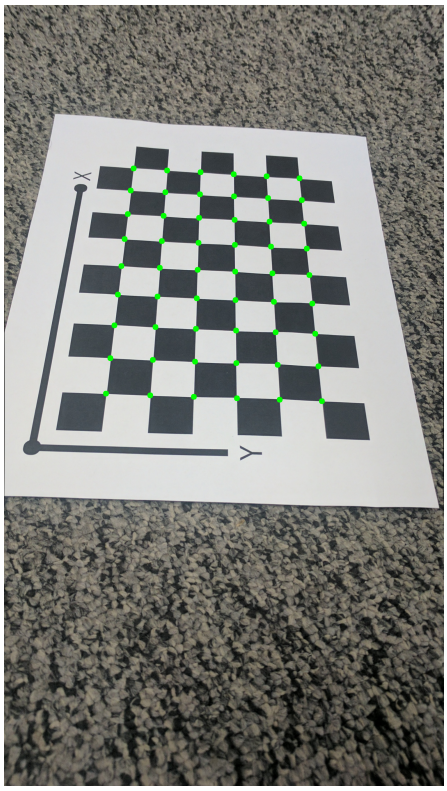

Fig. 5. Rectified Img 20170209 042612 with reprojected model points

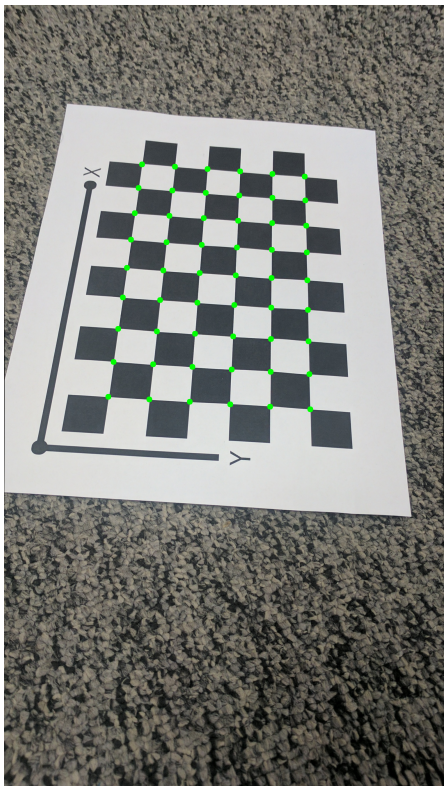

Fig. 4. Rectified Img 20170209 042610 with reprojected model points

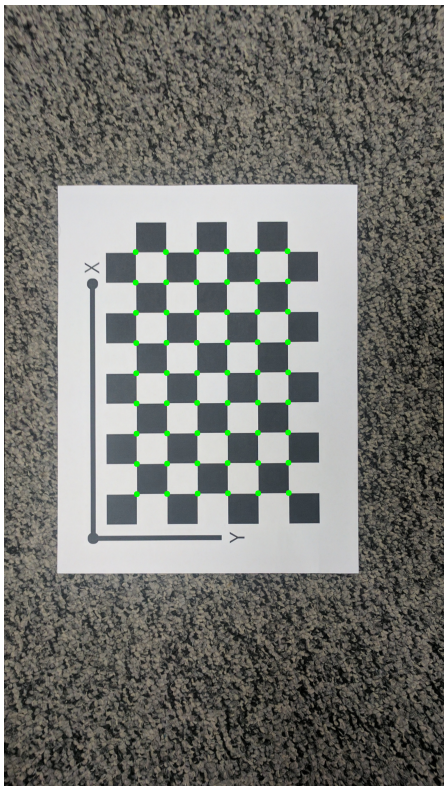

Fig. 6. Rectified Img 20170209 042614 with reprojected model points

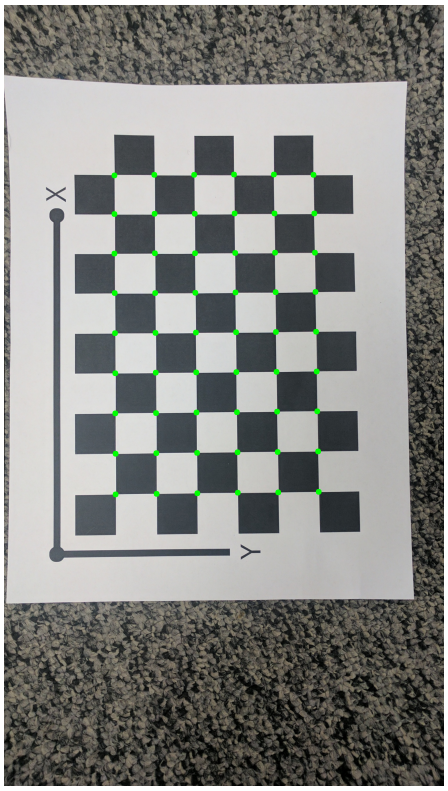

Fig. 7. Rectified Img 20170209 042616 with reprojected model points

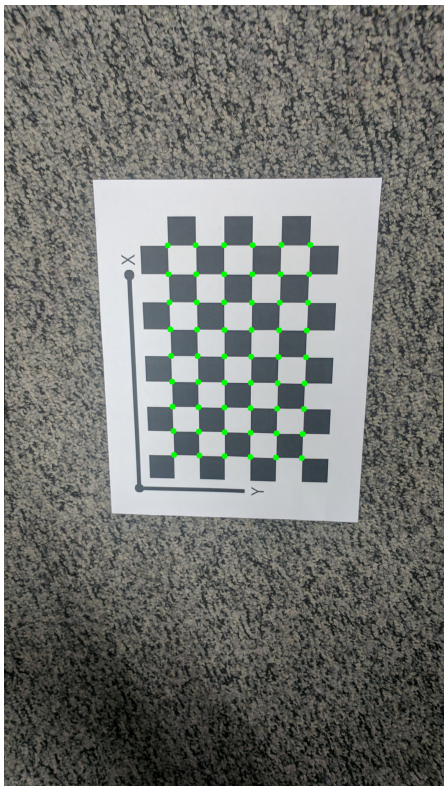

Fig. 9. Rectified Img 20170209 042621 with reprojected model points

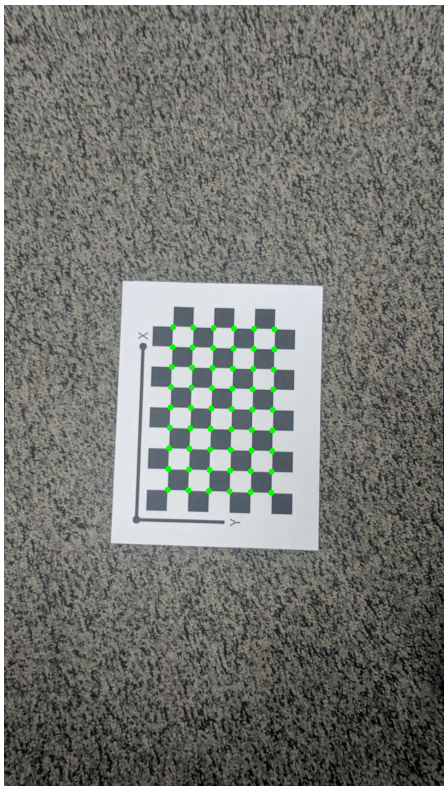

Fig. 8. Rectified Img 20170209 042619 with reprojected model points

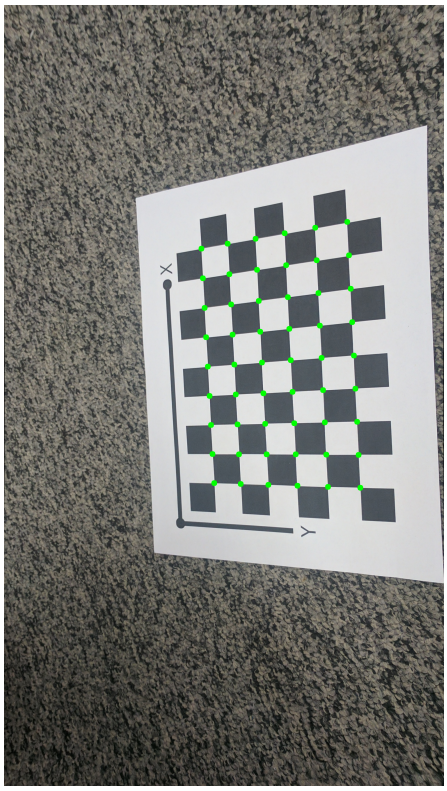

Fig. 10. Rectified Img 20170209 042624 with reprojected model points

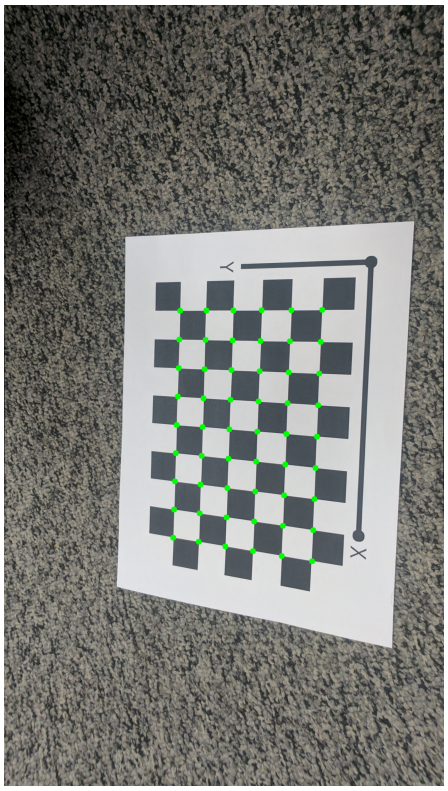

Fig. 11. Rectified Img 20170209 042627 with reprojected model points

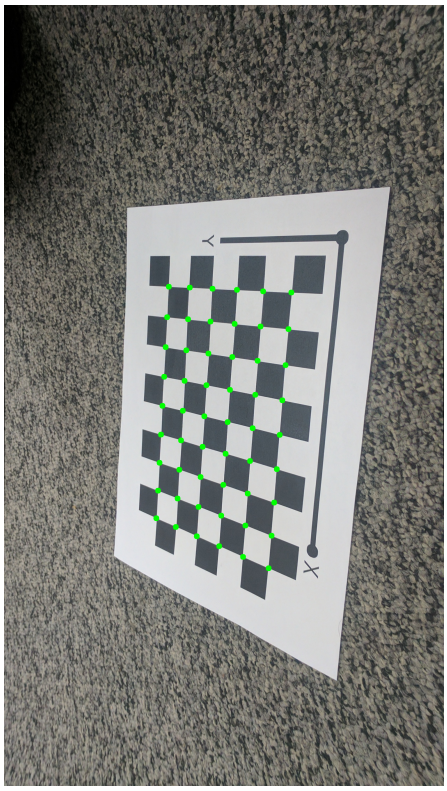

Fig. 13. Rectified Img 20170209 042630 with reprojected model points

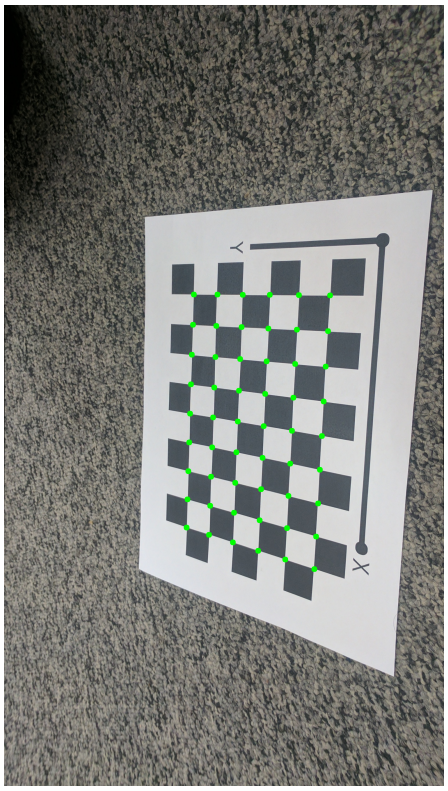

Fig. 12. Rectified Img 20170209 042629 with reprojected model points

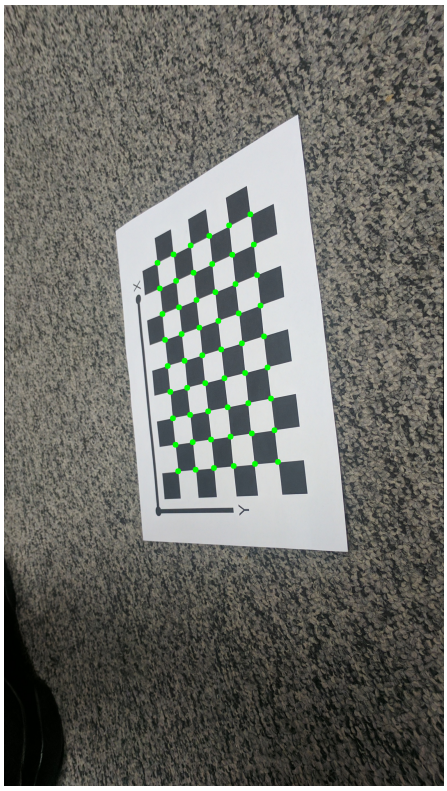

Fig. 14. Rectified Img 20170209 042634 with reprojected model points