RBE 549 Homework 1: AutoCalib

Amrit Krishna Dayanand
MS Robotics Engineering
WPI
Email: adayanand@wpi.edu

***Abstract*—The aim of this project is to automatically calibrate a camera by using it to take multiple images of a checkerboard of known dimensions in the environment. This approach applies least-squares minimization to estimate an initial camera intrinsic matrix and distortion coefficients (up to the 2nd order) and then use a non-linear geometric optimization to refine the estimates. A reprojection of the checkerboard's world coordinates is superimposed onto the rectified image to evaluate the performance of the calibration method.**

I. Introduction

A camera can be easily modeled using the pinhole model. This linear model can be used to project objects in the 3D world onto a discrete, 2D image plane in pixels. The transformation involves the camera extrinsics (rotation and translation of the world frame with respect to the camera) and the camera's intrinsics (geometry of the camera like focal length, principal point and skewness). The aim of this homework is to use an object with known positions in the world to calibrate the intrinsic matrix of a camera. Furthermore, we take into consideration radial distortion and estimate its value.

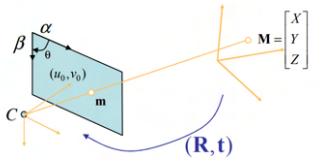

Fig. 1. Pinhole model of a camera is used to project a 3D point in space onto a discrete, 2D image plane

The paper by Zhengyou Zhang (Microsoft Research) published a robust and efficient way of automatic camera calibration. The key steps are estimating the camera intrinsic matrix, the extrinsics, and approximate distortion, followed by a non-linear geometric error minimization to refine all parameters.

II. Basic Equations

The camera calibration process assumes a pinhole camera model, as shown in 1. Let $X = [X, Y, Z]^T$ denote the 3D coordinate of a point in the world coordinate frame, and let $x = [u, v]^T$ denote a 2D image coordinate representing the projection of X on the image plane. Let $\tilde{X} = [X, Y, Z, 1]^T$ and $\tilde{x} = [u, v, 1]^T$ represent the homogeneous coordinates of X and x, respectively.

Upto a scale, s, the projection of the 3D point to a 2D image is given by

$$s\tilde{\boldsymbol{x}} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\tilde{\boldsymbol{X}} \tag{1}$$

where s is a scale factor, $(\boldsymbol{R}, \boldsymbol{t})$ are the 3x3 rotation matrix and 3x1 translation vector of the world coordinate frame with respect to the camera coordinate system (extrinsic parameters), and K is the camera intrinsic matrix describing the camera's inner geometry.

The camera intrinsic matrix describes the focal length in the x direction, f_x, and focal length in the y direction, f_y, the skewness of the image plane axes, γ and the principal point coordinates, (u_0, v_0). The intrinsic matrix, K is defined as follows:

$$K = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

In most cases, the γ parameter will be 0, implying that the image axes, and consequently each pixel, is rectangular.

The abbreviation K^{-T} implies $(K^{-1})^T$ or $(K^T)^{-1}$.

In this model, a key assumption is made that points in the 3D world lie on the same plane. Therefore, $\tilde{\boldsymbol{X}} = [X, Y, 0, 1]$. This simplifies Equation 1 from a 3x4 matrix to a 3x3 matrix given by

$$s\tilde{\boldsymbol{x}} = \boldsymbol{K}[\boldsymbol{r_1}|\boldsymbol{r_2}|\boldsymbol{t}]\tilde{\boldsymbol{X}}$$

where $\tilde{\boldsymbol{X}} = [X, Y, 1]$. Since 3D points lie on a plane, we can relate $\tilde{\boldsymbol{X}}$ and $\tilde{\boldsymbol{x}}$ by a homography such that

$$s\tilde{\boldsymbol{x}} = \boldsymbol{H}\tilde{\boldsymbol{X}} \tag{2}$$

with

$$\boldsymbol{H} = \lambda \boldsymbol{K}[\boldsymbol{r_1}|\boldsymbol{r_2}|\boldsymbol{t}] = [\boldsymbol{h_1}|\boldsymbol{h_2}|\boldsymbol{h_3}]$$

III. Estimating Camera Intrinsic Matrix

To solve camera calibration, a closed-form solution is computed to generate an initial guess of the camera intrinsic matrix.

We start by finding corners in the images, considering only the inner grid of the checkerboard (of size 6 x 9). We can get these corners using cv2.findChessboardCorners from the OpenCV library. This yields a set of points that correspond to a 6x9 world-grid with an arbitrary origin at (0,0) in world-coordinates.

From this we can compute the homography between the world-grid and image corners for each of the N images. I implemented this using my own Direct Linear Transform function.

Next we define $B = A^{-T}A^{-1}$. B is a symmetric matrix parameterized by a 6-d vector $\boldsymbol{b} = \begin{pmatrix} B11 & B12 & B22 & B13 & B23 & B33 \end{pmatrix}$. Knowing that

Next, from the orthonormal property of the rotation matrix we generate two constraints. This is used to create a system of linear equations such that $\boldsymbol{Vb} = 0$. Each image contributes to two rows of matrix $\boldsymbol{V}$ as follows:

$$\begin{bmatrix} \boldsymbol{v}_{12}^T \\ (\boldsymbol{v}_{11} - \boldsymbol{v}_{22})^T \end{bmatrix} \boldsymbol{b} = 0$$

where

$$\boldsymbol{v}_{ij} = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix}$$

Using singular value decomposition, we solve $\boldsymbol{Vb} = 0$ and recover the $\boldsymbol{b}$ vector. The values of K are unpacked from this as follows

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

IV. Estmiating Camera Extrinsics Matrix

Once we know the camera intrinsics matrix K, we compute R and t as follows

$$\begin{aligned} \boldsymbol{r_1} &= \lambda \boldsymbol{K^{-1}} \boldsymbol{h_1} \\ \boldsymbol{r_2} &= \lambda \boldsymbol{K^{-1}} \boldsymbol{h_2} \\ \boldsymbol{t} &= \lambda \boldsymbol{K^{-1}} \boldsymbol{h_3} \end{aligned}$$

with $\lambda = 1/||\boldsymbol{K^{-1}}\boldsymbol{h_1}||$. We are not concerned with $\boldsymbol{r_3}$ as we assume that the 3D model's Z component is 0.

V. Estimating Approximate Distortion k

Radial distortion can be modeled by the following

$$\begin{aligned} u_i &= u + (u - u_0)[k_1 r + k_2 r^2] \\ v_i &= v + (v - v_0)[k_1 r + k_2 r^2] \end{aligned}$$

where (u_i, v_i) is the observed image coordinate, (u, v) is the model predicted image coordinate, (k_1, k_2) are the first and second order distortion coefficients and $r = x^2 + y^2$, where (x, y) are the 3D point in the camera coordinate frame.

As an initial estimate we can assume minimal lens distortion, $k = [k_1, k_2]^T = [0, 0]^T$.

VI. Non-linear Geometric Error Minimization

We refine our estimates of the intrinsic matrix, K, and distortion coefficient vector, k vector by minimizing the following

$$argmin_{f_x,\gamma,f_y,u_0,v_0,k_1,k_2} \sum_{i=1}^{N}\sum_{j=1}^{M} ||x_{ij} - \hat{x}_{ij}(K, R_i, t_i, X_j, k)|| \tag{3}$$

where the reprojection error is the 2-norm of the difference between the observed image point and the model's estimate, which includes radial distortion. I chose to also minimize for skewness, γ, though this is not necessary and can be constrained if it is known that the camera has a rectangular sensor.

VII. Results

After optimization, the images were rectified using cv2.undistort. The corners were reprojected using the optimized camera intrinsic matrix, K, and distortion coefficients, k. These images are shown below. The mean reprojection error before and after optimization changed by 2%.

TABLE I
Mean Reprojection Error before and after Optimization

Error (unoptimized)	Error (optimized)
37.7049	36.9613

The final camera intrinsic matrix, K_f, and lens distortion vector, k_f, were:

$$K_f = \begin{bmatrix} 2049.227 & -3.409 & 7.605 \\ 0 & 2036.762 & 1363.150 \\ 0 & 0 & 1 \end{bmatrix}$$

$$k_f = \begin{bmatrix} 0.00914 \\ -0.06394 \end{bmatrix}$$

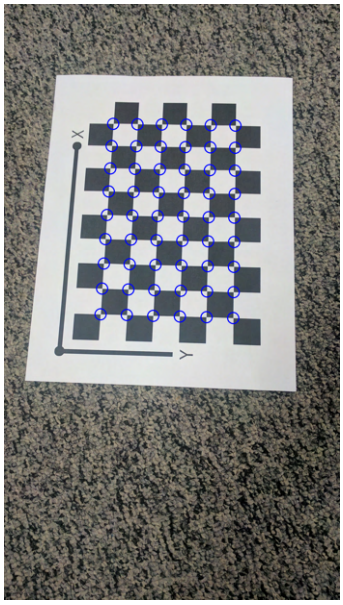

Fig. 2. Rectified image with reprojected points 1

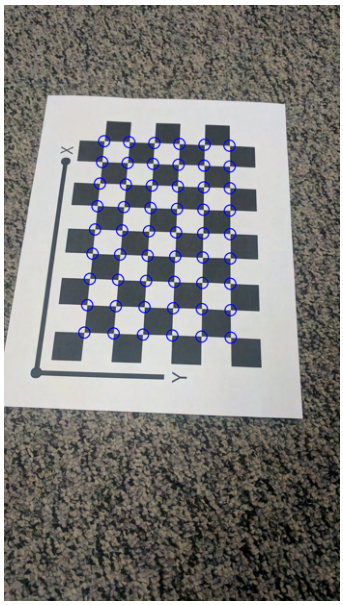

Fig. 3. Rectified image with reprojected points 2

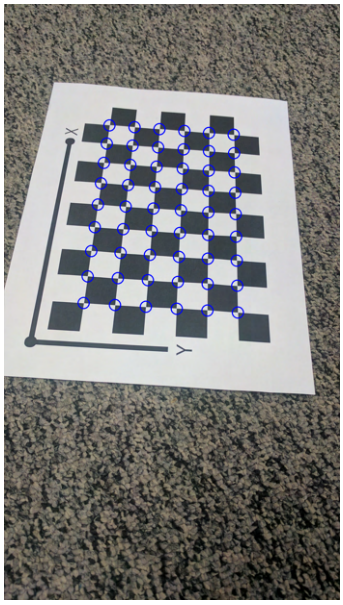

Fig. 4. Rectified image with reprojected points 3

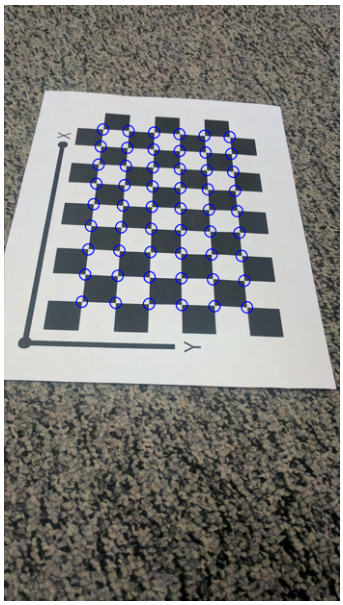

Fig. 5. Rectified image with reprojected points 4

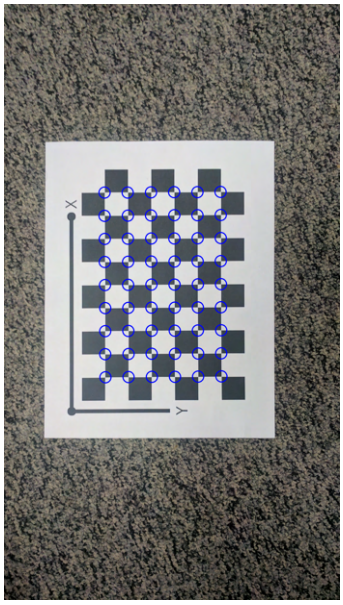

Fig. 6. Rectified image with reprojected points 5

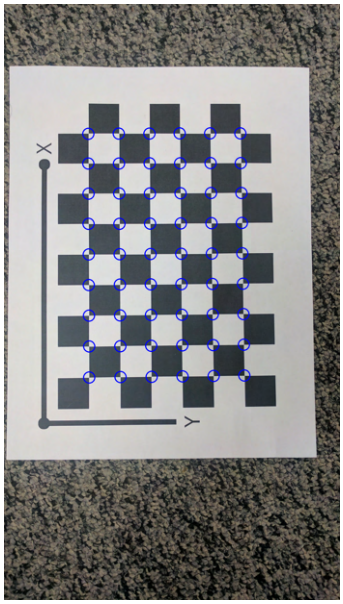

Fig. 7. Rectified image with reprojected points 6

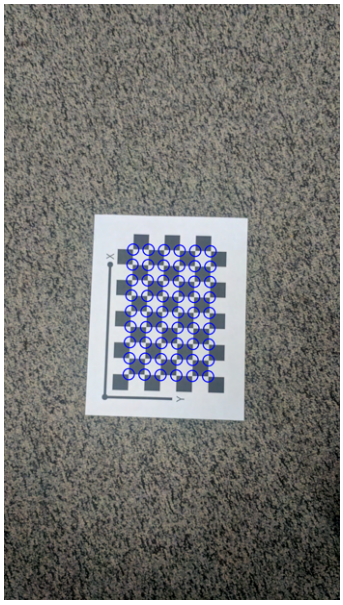

Fig. 8. Rectified image with reprojected points 7

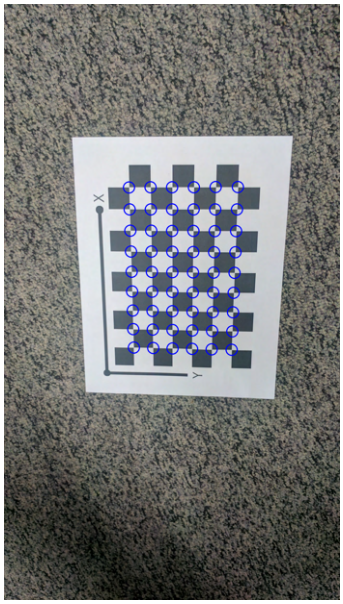

Fig. 9. Rectified image with reprojected points 8

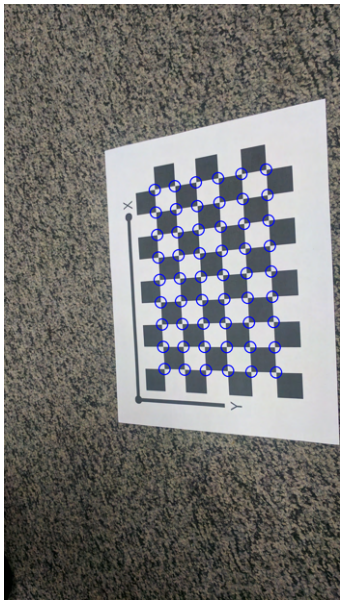

Fig. 10. Rectified image with reprojected points 9

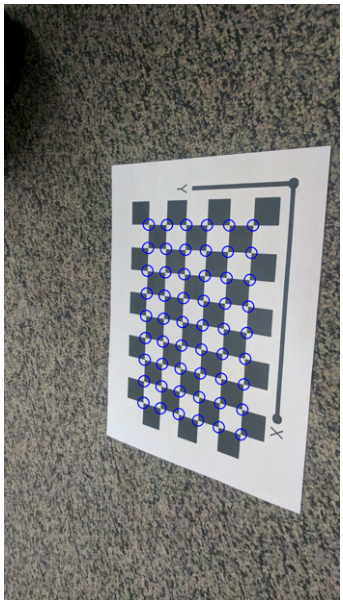

Fig. 12. Rectified image with reprojected points 11

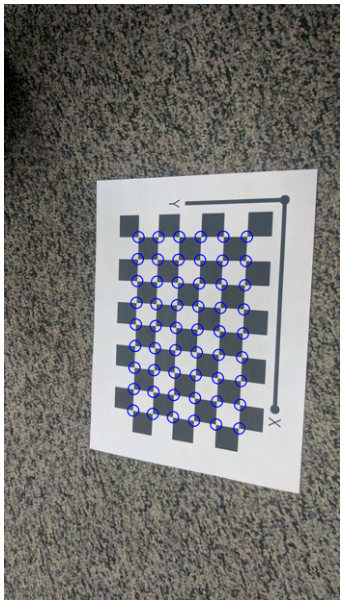

Fig. 11. Rectified image with reprojected points 10

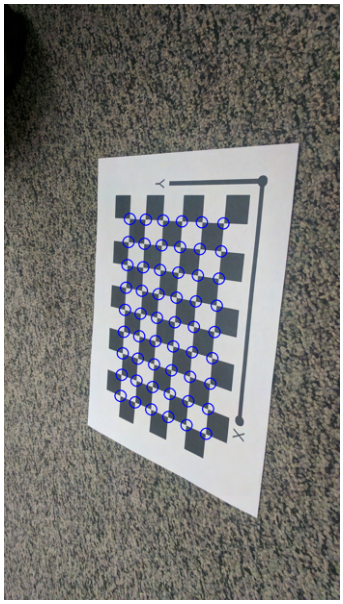

Fig. 13. Rectified image with reprojected points 12

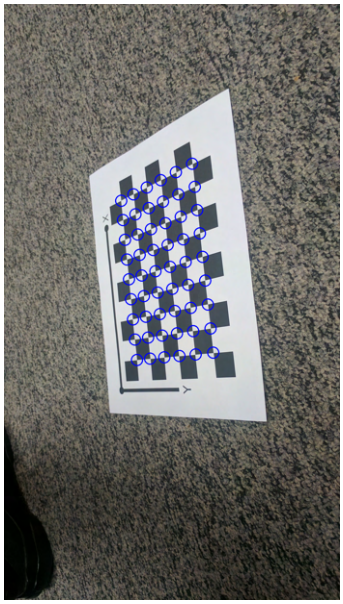

Fig. 14. Rectified image with reprojected points 13