RBE 549 HomeWork 1: AutoCalib

Venkata Sai Krishna Bodda
MS Robotics Engineering
Worcester Polytechnic Institute
Email: vbodda@wpi.edu

***Abstract*—The main aim of this Homework is to calculate Camera Calibration Parameters. The camera Calibration parameters include, Intrinsic Parameters, Extrinsic Parameters and distortion parameters. Intrinsic Parameters include fx(focal length * pixel density in x direction), fy(focal length*pixel desnity in y direction), principle point co-ordinates(cx, cy), and distortion parameters(k1, k2). Zhengyou Zhang Proposed an efficient and well known camera calibration technique which will be implemented in this homework.**

Index Terms - Camera Calibration, Camera Model, Intrinsic Camera Parameters

I. Introduction

The main purpose of Camera Calibration is to create a Camera Model which can convert world points in 3d space in a defined co-ordinate system to image plane frame in image pixel co-ordinate system. we can obtain camera model by calculating Intrinsic parameters matrix with extrinsic parameters matrix(Transformation Matrix). A representation of intrinsic matrix is given below in Fig.1.

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Fig. 1. Intrinsic Matrix

Zhang's method relies on calibration using a 2d chess patterned sheet. The Global Centre lies on the sheet thus we are say that Z = 0 for all the corners. This makes using r3 in rotational matrix redundant as it's multiplied by 0. Omitting r3 reduces the camera model equation which is represented in Fig.2.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$
$$= \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} .$$

Fig. 2. Conversion Method

II. Initial Parameter Estimation

We employ the optimization process to acquire the intrinsic properties of the camera. However, for this purpose, it is essential to initially estimate the parameters accurately. To achieve this, we utilize 13 images captured with a Google Pixel XL smartphone, with the focal length fixed. The physical dimension of each side of the checkerboard square is 21.5mm. Although the checkerboard has a size of (7 x 10), we disregard the corner columns and rows, thus effectively working with a (6 x 9) checkerboard size.

A. Solving for approximate K or camera intrinsic matrix

The process of getting Intrinsic parameters starts with finding chess corner co-ordinates in both image pixel co-ordinates and global co-ordinates and use them to get camera model. For finding chess corners in image plane in pixel co-ordinates, we use cv2.findChessboardCorners function to detect find corners in every image. There are a total of 54 corners in each image. cv2.drawChessboardCorners function is used visualize these corners in each image. Fig.3 shows an example of corners traced on an image.

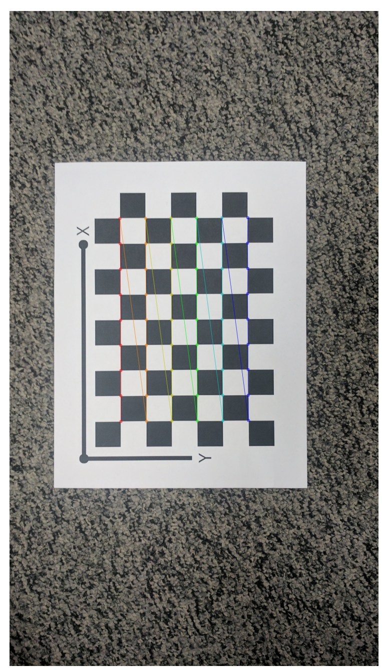

Fig. 3. Chess Board Corners

Each side of a square within the pattern measures 21.5mm. Consequently, to determine the global corners, we multiply the row and column values of each corner by 21.5.

The homography matrix between the sheet plane and the image plane can be computed for each image using these local corner coordinates and global corner coordinates. The **"Homography"** function within the code accomplishes this task.

After getting Homography Matrix, the main goal is to decompose this into A and [r1, r2, t]. The closed form equation represented in Fig. 4 allows us to efficiently decompose the matrix.

$$\mathbf{B} = \mathbf{A}^{-T}\mathbf{A}^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix}$$
$$= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix}$$

Fig. 4. Closed Form Equation

Note that B is symmetric, defined by a 6D vector

$$\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$$

Fig. 5. 6d vector representation of B

Let the i^{th} column vector of H be $h^i = [h^{i1}, h^{i2}, h^{i3}]^T$. Then, we have

$$h_i^T B h_j = v_{ij}^T b \tag{1}$$

where v_{ij} can be represented as

$$\mathbf{v}_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2},$$
$$h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T$$

Fig. 6. V_{ij} representation

Therefore, the two fundamental constraints (3) and (4), from a given homography, can be rewritten as 2 homogeneous equations in b:

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

Fig. 7. $v * b$ representation

If n images of the model plane are observed, by stacking n such equations as (8) we have $Vb = 0$. We can Solve this equation to get the b matrix values from which camera intrinsic parameters can be solved by using Closed Form Equation Matrix(fig.4)

B. Estimate approximate R and t or camera extrinsics

Once the Camera's Intrinsic parameters are calculated, Extrinsic parameters can be derived using the formula represented in figure 8.

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

Fig. 8. Extrinsic Matrix Calculation

where $\lambda = 1/\|\mathbf{A^{-1}h_1}\|$

C. Approximate Distortion k

Since we've assumed minimal distortion in the camera, we can set the distortion coefficients as k = $[0, 0]^T$ for a reliable initial estimation. But For getting predicted points, we need to take distortion in effect while calculating the predicted points. This can be done by following equation in figure 9.

$$\breve{u} = u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$
$$\breve{v} = v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2]$$

Fig. 9. Distortion

III. Non-linear Geometric Error Minimization

Once the initial estimates of K,R,t,k are calculated, the optimization problem can be represented as equation in Fig.10

$$\sum_{i=1}^{N}\sum_{j=1}^{M} ||\boldsymbol{x}_{i,j} - \hat{\boldsymbol{x}}_{i,j}\left(\boldsymbol{K}, \boldsymbol{R}_i, \boldsymbol{t}_i, \boldsymbol{X}_j, \boldsymbol{k}\right)||$$

Fig. 10. Optimization Problem

The loss function in the Optimization problem is geometric error function (L2 norm of difference of co-ordinates).

scipy.optimise.least_squares function is used to minimise the distance between predicted corner from global co-ordinates and original corner. The intrinsic Parameters K and distortion co-ordinates after optimization are given below along with re projection error.

```
---Optimizing---
Parameters After Optimization: \[[2.58034932e+03 0.00000000e+00 7.63975031e+02]
 [0.00000000e+00 2.55904359e+03 1.34842123e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]
Radial Distortion After Optimization: [0.00825965138633731, -0.08578054326725851]
Optimised co-ordinates loss is: 0.7608570938400088
```

Fig. 11. Optimization Problem

here,
$f_x = 2580$
$f_y = 2559$
$c_x = 764$

$c_y = 1348$
$k_1 = 0.00825$ and $k_2 = -0.0857$

The predicted corners from global co-ordinates are plotted on each image after distorting the image using cv2.undistort function. THe results of the predicted corners are shown in below figures.

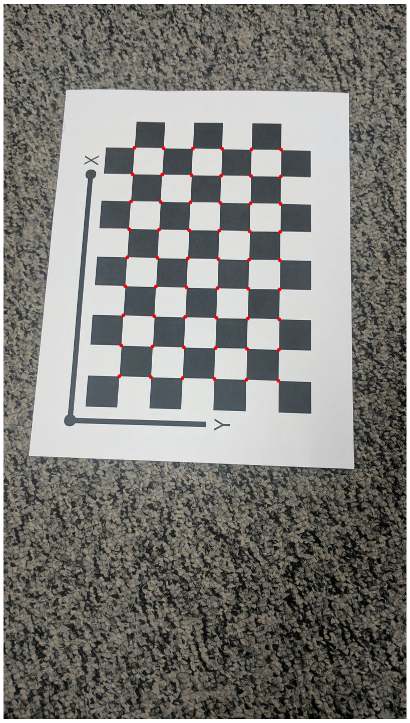

Fig. 12. Reprojection of corners on rectified image

IV. References

1) Z. Zhang, "A flexible new technique for camera calibration," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 11, pp. 1330-1334, Nov. 2000, doi: 10.1109/34.888718.

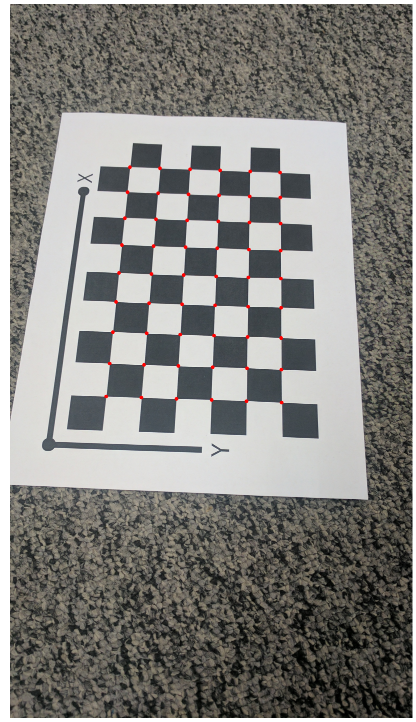

Fig. 13. Reprojection of corners on rectified image

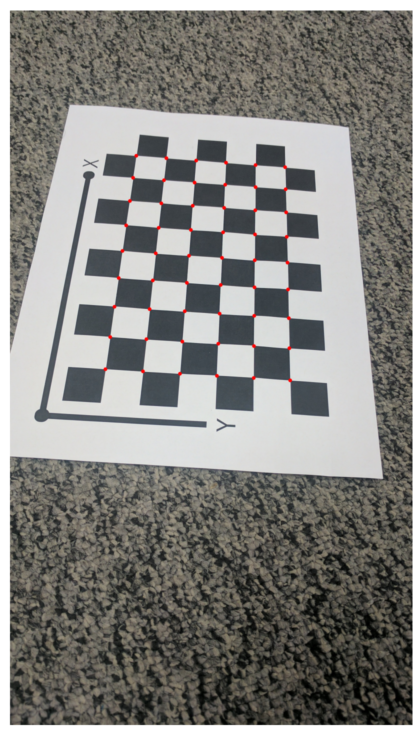

Fig. 14. Reprojection of corners on rectified image

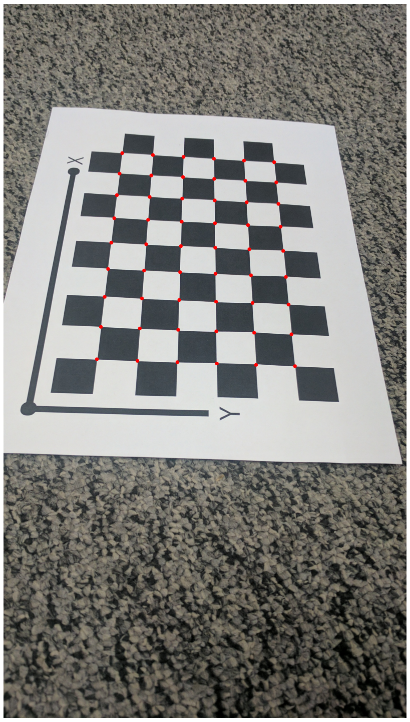

Fig. 15. Reprojection of corners on rectified image

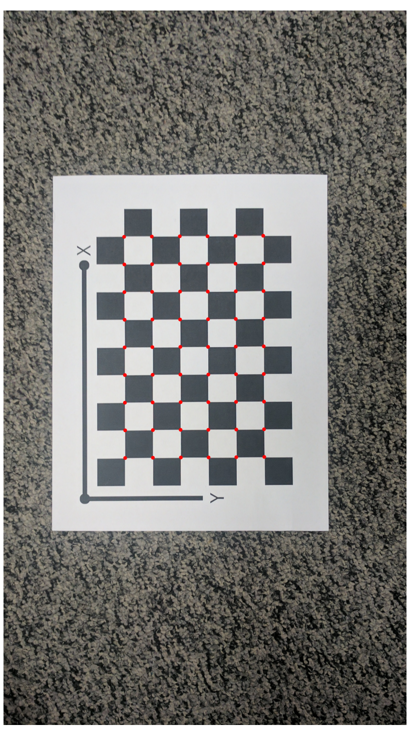

Fig. 16. Reprojection of corners on rectified image

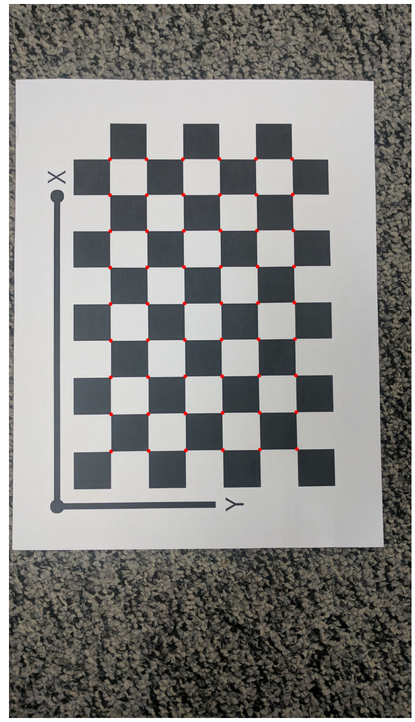

Fig. 17. Reprojection of corners on rectified image

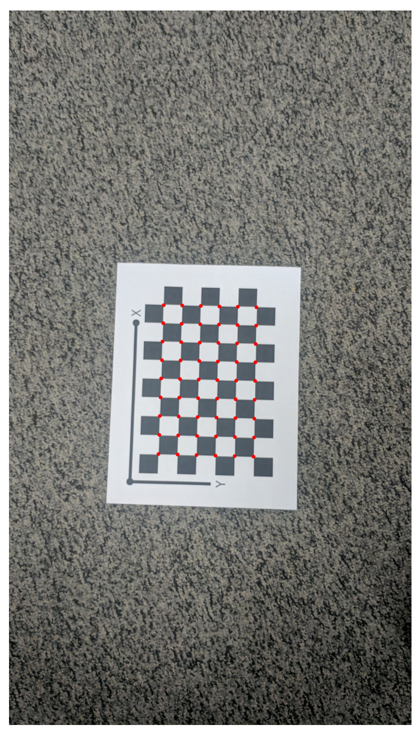

Fig. 18. Reprojection of corners on rectified image

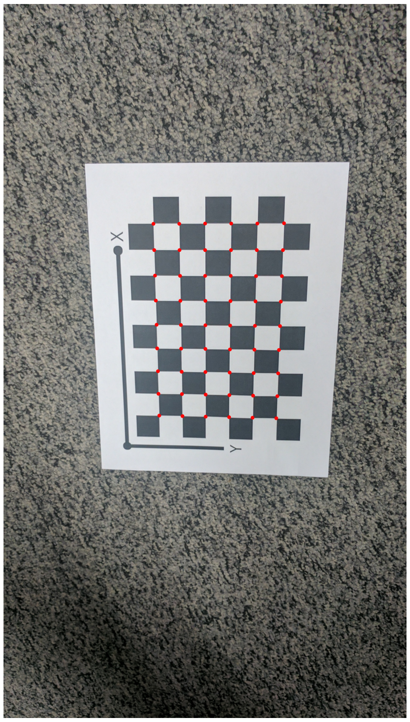

Fig. 19. Reprojection of corners on rectified image

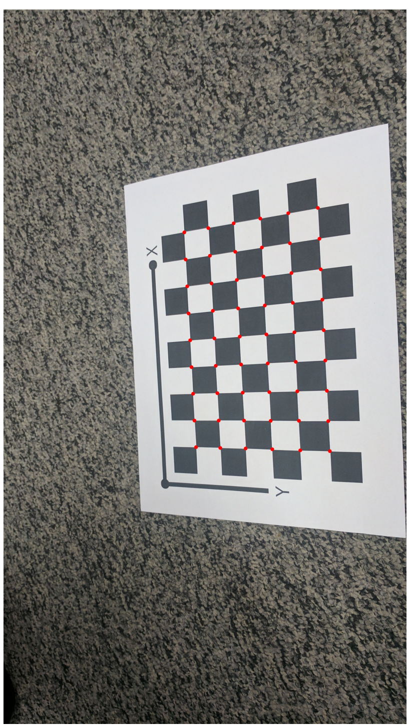

Fig. 20. Reprojection of corners on rectified image

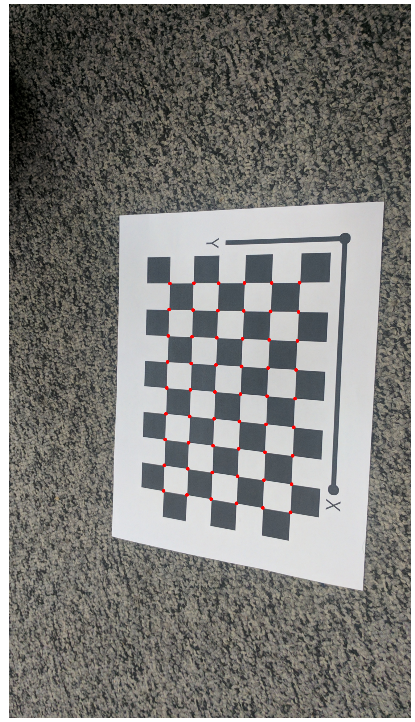

Fig. 21. Reprojection of corners on rectified image

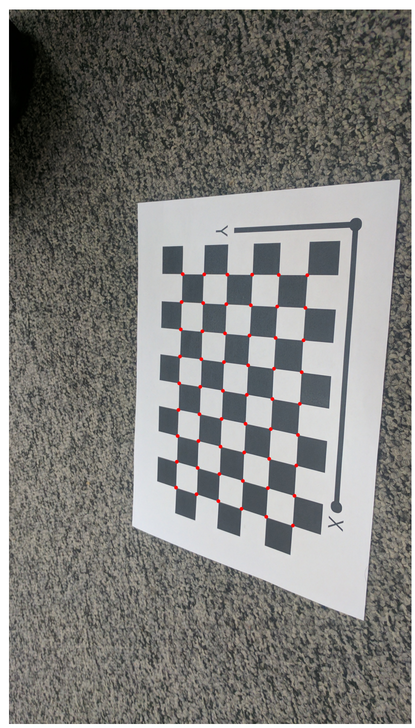

Fig. 22. Reprojection of corners on rectified image

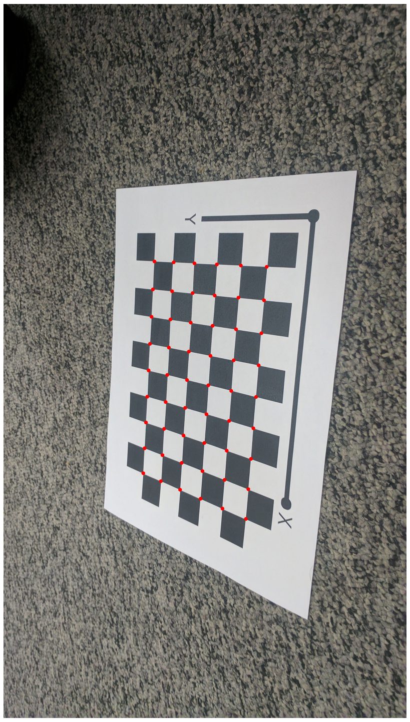

Fig. 23. Reprojection of corners on rectified image

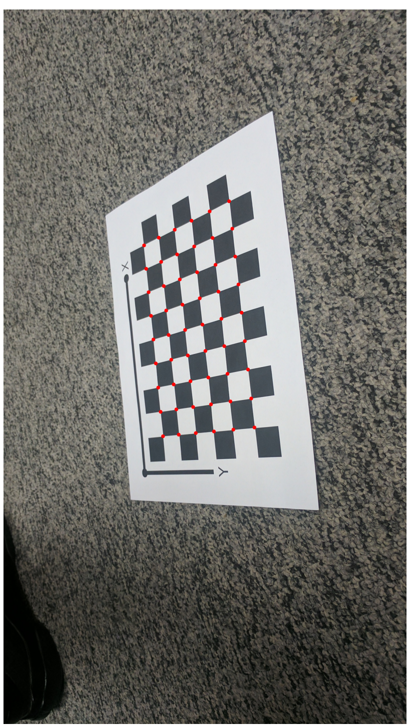

Fig. 24. Reprojection of corners on rectified image