CMSC733: Homework 1 - AutoCalib

Khoi Viet Pham
Email: khoi@terpmail.umd.edu
Use 2 late days

I. Initial Parameter Estimation

This section presents my approach to estimate the camera intrinsic parameters, the extrinsic parameters for each provided image, and the distortion coefficients.

A. Approximate camera intrinsic K

For each input image, I use the function *cv2.findChessboardCorners* to find all chessboard corners in the image. Example of detected chessboard corners is shown in figure 1.

The world coordinates for each corner are numbered from 1 to the number of corners on each row (6 in our homework), and from 1 to the number of corners on each column (9 in our homework). For example, this means that the top-left corner has world coordinate $(1, 1)$, and the bottom-right corner has world coordinate $(6, 9)$. From these world coordinates and their image pixel coordinates, we can estimate the homography between the model plane and each input image. With the estimate homography matrices, I follow section 3.1 in [1] to find the camera intrinsic parameters. Instead of assuming the γ value (skewness of the 2 image axes) as 0, I explicitly compute it based on the formulas in the paper.

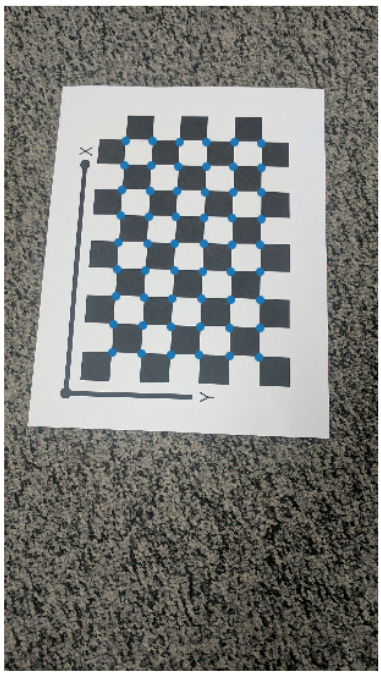

Fig. 1. Example of detected chessboard corners.

B. Estimate camera extrinsics R and t for each image

From the estimated homography matrix and the computed camera intrinsic, I follow section 3.1 in [1] to find the camera extrinsic parameters for each input image. Because the estimated rotation matrix does not satisfy the properties of a rotation matrix as described in the paper, I also follow their approach to estimate the best rotation matrix in appendix C.

C. Approximate distortion coefficients k_c

I follow the instruction to use $k_c = [0, 0]$ as the initial estimate.

II. Non-linear Geometric Error Minimization

In this section, I use *scipy.optimize* to refine the estimated parameters in the previous section.

Because my code keeps failing to find a good solution of the distortion coefficients k_c, I decide to write my own stochastic gradient descent for it. After having found a good distortion solution, I use them as the new initial values for k_c. However, my code that uses $scipy.optimize$ still fails to optimize for a better camera intrinsic matrix (the matrix doesn't change after using$scipy.optimize$). Therefore, I suspect that my implementation is wrong and I look forward to seeing the solution for this part.

III. Output

Following is the output of my code. I output 2 projection errors, one is the model that doesn't estimate the best rotation matrix according to appendix C and one that does estimate it. I also include matrix K as required below.

```
Finding corners in all images...
Done!
Estimating camera intrinsic parameters..
 Done!
K =  [2063.8982, 2.2726, 764.5863]
[0.0000, 2042.7786, 1333.7452]
[0.0000, 0.0000, 1.0000]
Finding extrinsic parameters for all images...
Done!
Finding distortion coefficients k1 and k2...
Done!
k1 = 0.003755, k2 = -0.019014
Performing non-linear geometric error
minimization...
Done!
K (after non-linear minimization) =
```

```
[2063.8982, 2.2726, 764.5863]
[0.0000, 2042.7786, 1333.7452]
[0.0000, 0.0000, 1.0000]
Re-projection error (without using estimated
rotation matrix: 0.759741
Re-projection error (using estimated
rotation matrix: 3.105491
Saving rectified images to Rectified_Imgs/
```

Rectified images are displayed below. Unfortunately, they don't differ much from the inputs.

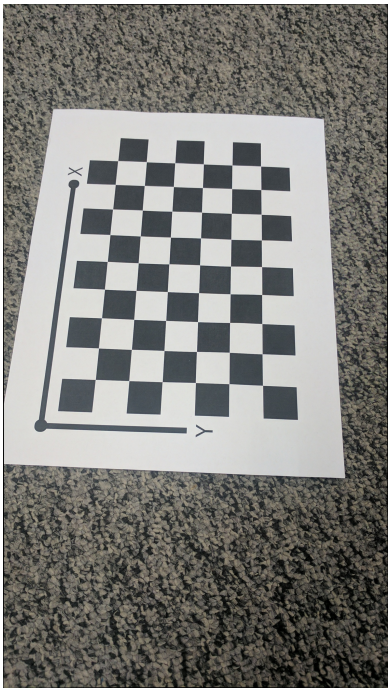

Fig. 3. Output of IMG_20170209_042608.jpg.

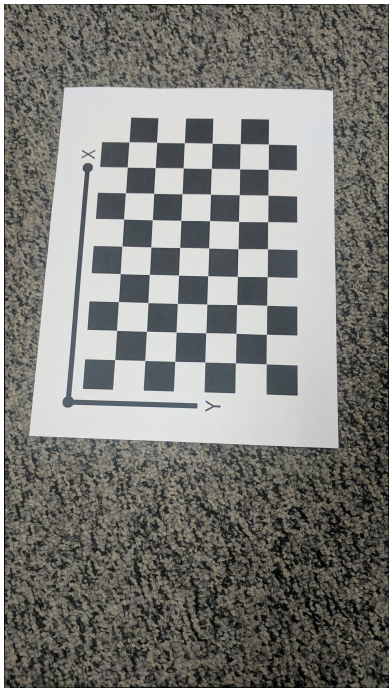

Fig. 2. Output of IMG_20170209_042606.jpg.

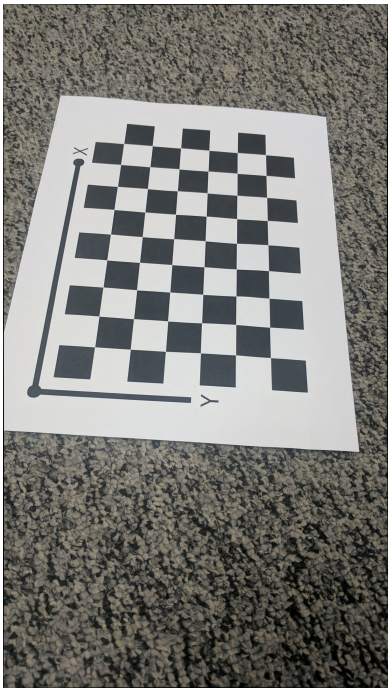

Fig. 4. Output of IMG_20170209_042610.jpg.

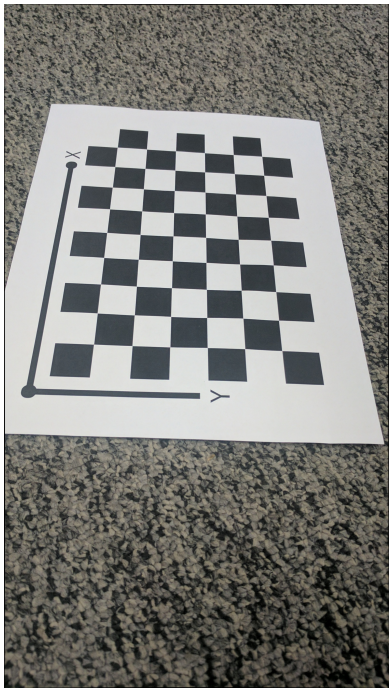

Fig. 5. Output of IMG_20170209_042612.jpg.

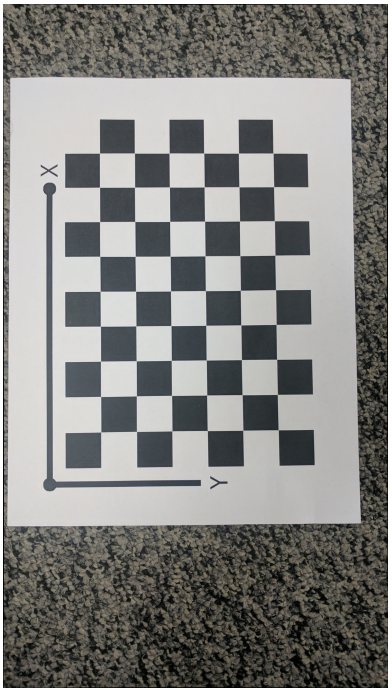

Fig. 7. Output of IMG_20170209_042616.jpg.

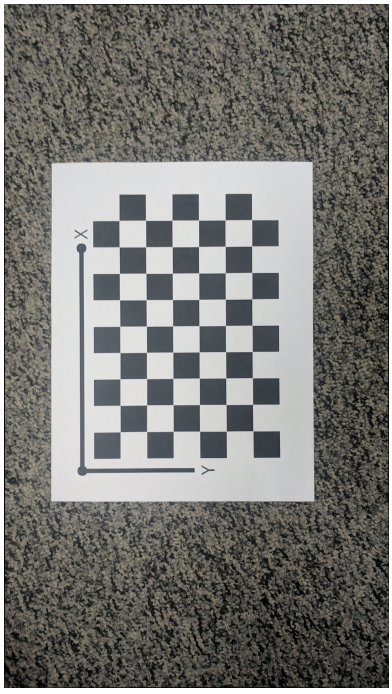

Fig. 6. Output of IMG_20170209_042614.jpg.

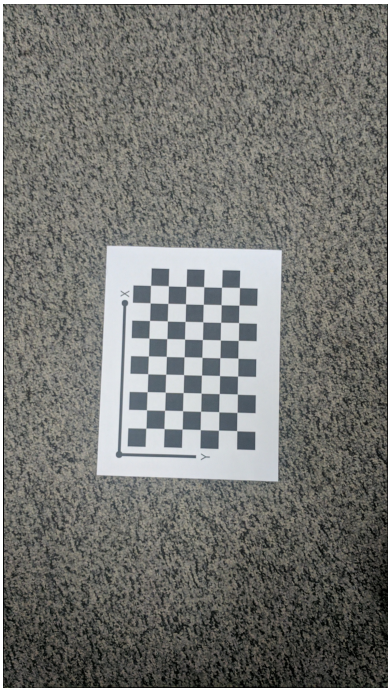

Fig. 8. Output of IMG_20170209_042619.jpg.

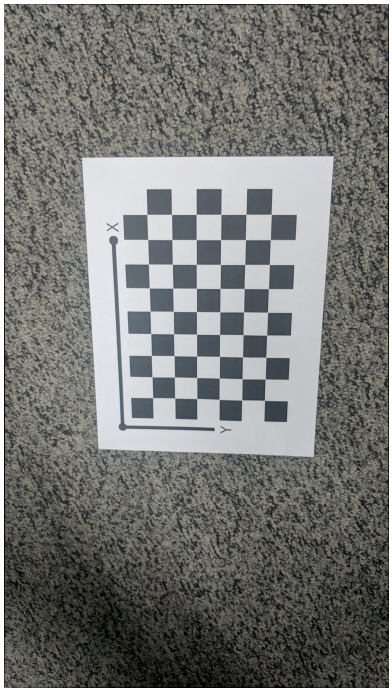

Fig. 9. Output of IMG_20170209_042621.jpg.

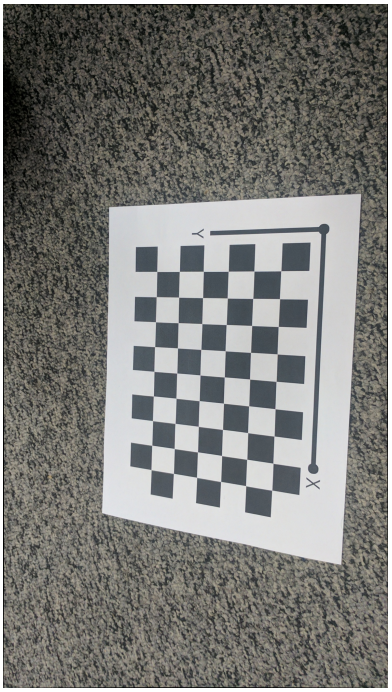

Fig. 11. Output of IMG_20170209_042627.jpg.

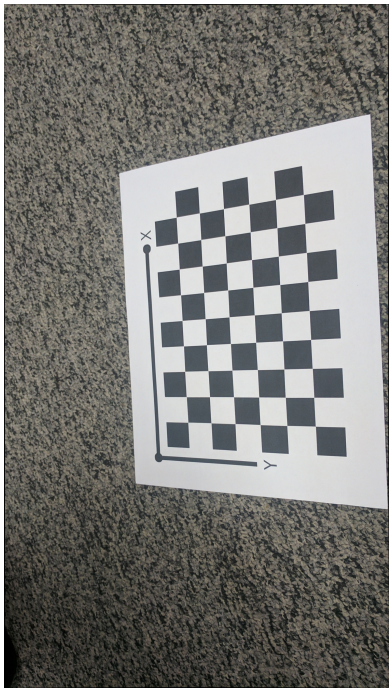

Fig. 10. Output of IMG_20170209_042624.jpg.

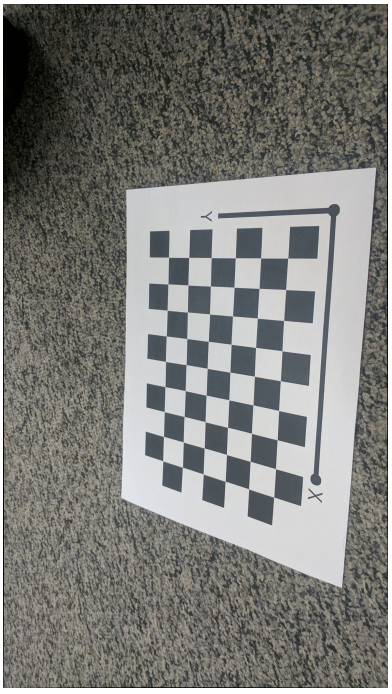

Fig. 12. Output of IMG_20170209_042629.jpg.

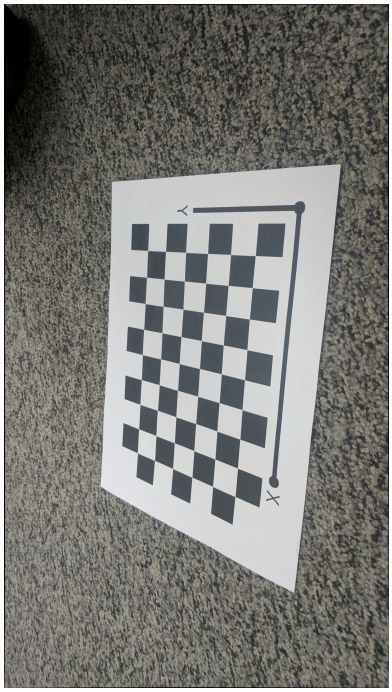

Fig. 13. Output of IMG_20170209_042630.jpg.

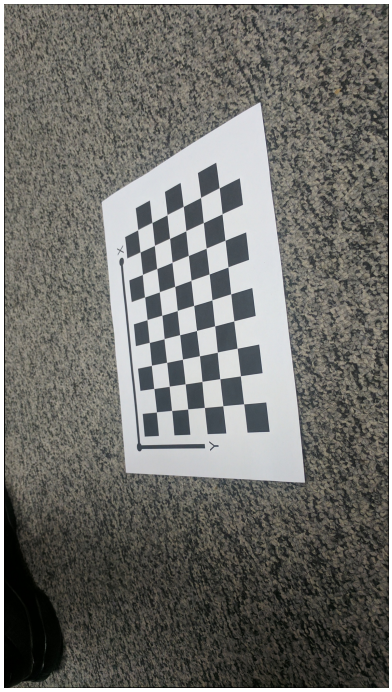

Fig. 14. Output of IMG_20170209_042634.jpg.