CS512 Assignment 5: Report

Kajol Tanesh Shah
Department of Computer Science
Illinois Institute of Technology
April 29, 2022

Abstract

In this assignment, we have performed camera calibration, extracted feature points from image using openCV functions, computed camera parameters and implemented Robust estimation using RANSAC. To do these operations, we have used OpenCV libraries in python and other libraries like Numpy, Matplotlib, Math, Pandas, etc. were also used.

1. Problem Statement

In this assignment, our aim was to extract feature points from a given calibration target and show them on the image. Then, using the world and image points that we get from the image, compute the camera parameters i.e. intrinsic and extrinsic camera parameters. For this, we can either use a planar or non-planar calibration approach. After computing parameters, calculate the mean square error between known and computed position of image points. Then, implement RANSAC algorithm for robust estimation

2. Proposed solution

Using OpenCV, NumPy, Pandas, Math and other libraries, we have implemented camera calibration and robust estimation in Python.

Part 1: Extract feature points from calibration target
1)For this, we first read the image and converted the image to grayscale
2)Then, we extracted the features in the image using findChessboardCorners() and drew the corners on the image using drawChessboardCorners()
3)After this, we wrote the computed world and image points in separate files

Part 2: Compute camera parameters using non-planar approach
1)In this step, we loaded the image and world points from the first step and constructed a projection matrix front the points
2)After this, we computed the intrinsic and extrinsic parameters of the camera using the formulas for finding parameters using non-planar approach
Parameters found included rho, u0, v0, alphaU, alphaV, K*, T*, r1, r2, r3, etc
3)After that, we calculated the mean square error between known and computed position of image points

Part 3: RANSAC for Robust estimation

1)In this step, we created a .config file and passed parameters such as probability, no. of draws, etc in the file and then loaded this file in our program

2)Here, we randomly selected some corresponding points from our world and image points and then tried to find the number of inliers. We repeated this process for ‘n’ number of draws and computed the best matrix which had the maximum number of inliers. Based on this, we then calculated the camera parameters.

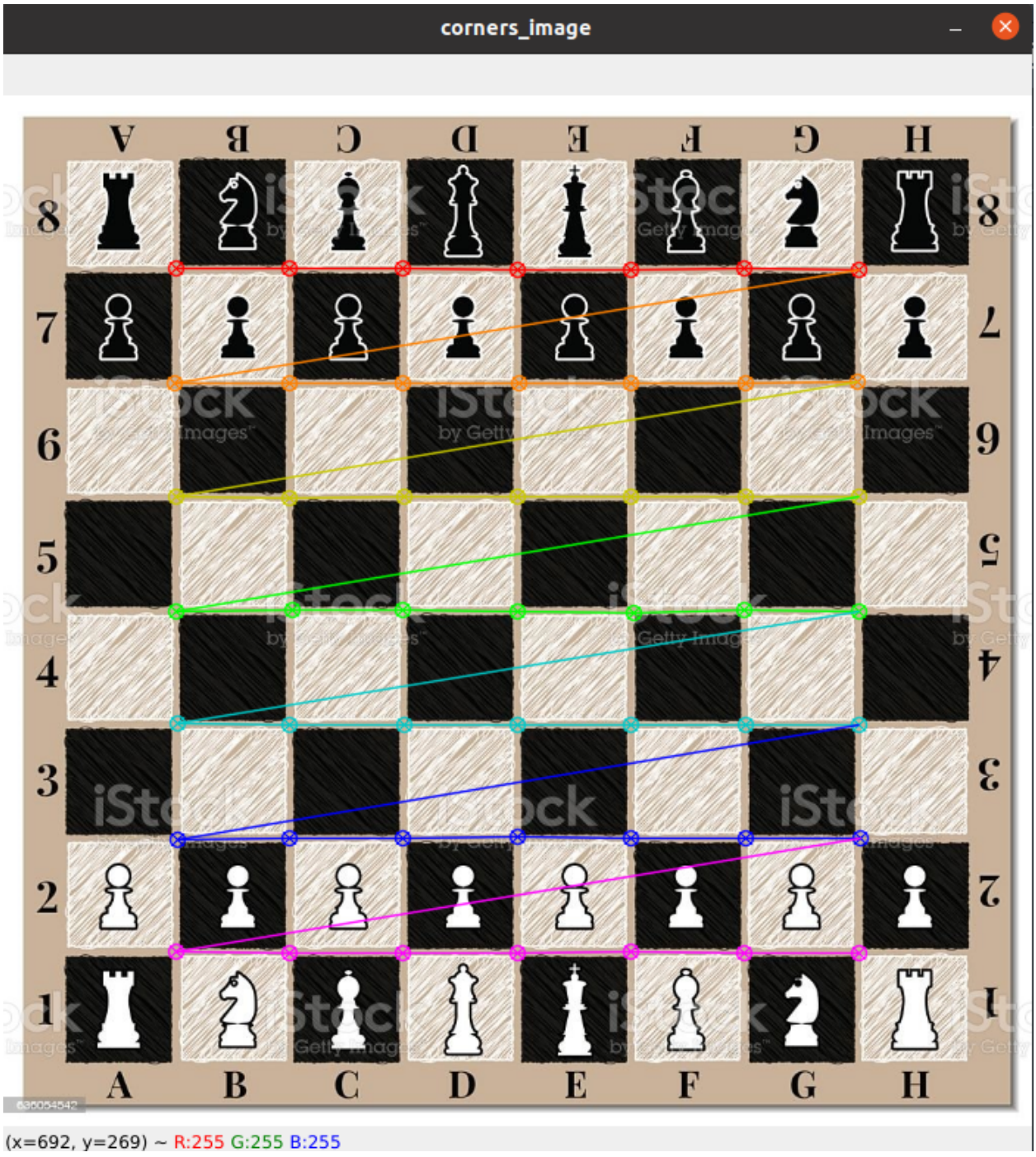
3. Implementation details

Some of the problems and design issues that were faced are as mentioned below:

- In the first step, I was not getting different coloured feature points and all corners were getting displayed in the same colour. The issue was because of passing gray image to drawcorners function instead of original image
- In part 2, the image points and world points which I got from the first part were not giving proper results for camera parameters. So, I used the data files provided on the link provided on shared drive to test my program and it worked fine
- In the 3rd part, I was not able to successfully implement the RANSAC as some parts of my code were broken and halfway through the program, I was getting errors
- The issue for my implementation of RANSAC lies with the error distance calculating function which I was not able to fix

4. Results

Part 1: In this image, we can see the detected corners using the findChessboardCorners() method



Part 2: Below are the results for the camera parameters computed for one of the data files provided for world and image points

```
In [18]: runfile('/home/kajol/Desktop/AS5_Q2.py', wdir='/home/kajol/Desktop')
jjj (268, 3)
(12,)
(3, 4)
Camera parameters are:-
Rho:  -419526.1371019501
Intrinsic parameters:
u0:  320.00017039455435
v0:  239.9999709523749
Alpha v:  652.1740748292576
s:  -3.39837251940966e-05
Alpha u:  652.1740688650467
K*:  [[ 6.52174069e+02 -3.39837252e-05  3.20000170e+02]
 [ 0.00000000e+00  6.52174075e+02  2.39999971e+02]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00]]
Epsilon:  -1
Extrinsic parameters:
T*: [ 2.57726950e-04 -3.26846051e-05 -1.04880905e+03]
r3:  [0.47673145 0.57207743 0.66742381]
r1:  [-7.68221190e-01  6.40184508e-01  1.46359878e-07]
r2:  [-0.4272743  -0.51272918  0.74467809]
R*:  [[-7.68221190e-01  6.40184508e-01  1.46359878e-07]
 [-4.27274298e-01 -5.12729182e-01  7.44678091e-01]
 [ 4.76731452e-01  5.72077427e-01  6.67423808e-01]]
Mean Square error:  1.6605412420597685e-09
```

Part 3: RANSAC implementation was not successful

References

https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html

https://learnopencv.com/tag/findchessboardcorners/

https://www.youtube.com/watch?v=EkYXjmiolBg