COL780:A3: Camera Calibration and Cube Insertion

Sidharth Agarwal 2019CS50661

April 7, 2023

Libraries Used:

matplotlib, numpy, pandas, scipy, openCV and standard python libraries.

Run:

For running the camera camera calibration use **python3 main.py {input_data.csv} {test_data.csv}**. You will be able to see the original points and the predicted points for test_data.csv visualized in **output.png**

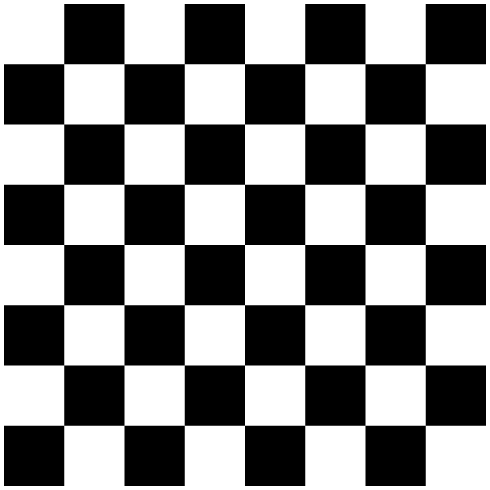
For getting the 2d co-ordinates of clicked points on an image, use **python3 2dpoints.py {img.jpeg}**

For creating a cube on a new checkboard image use, **python3 ar.py {data.csv} {board.jpeg} {length of side of cube}**

Q1

Dataset creation

Now I took my RealMe 3 phone camera to create the dataset. I printed 3 images of the checkerboard given below and placed them orthogonally. I marked around 16 points manually on the 3 sheets and



calculated their 3d coordinates. After that I clicked the setup's picture and later manually marked those points on the image using opencv functions in **2dpoints.py** to get their 2d coordinates. This all helped me to create the dataset **data.csv** which contains points and their corresponding 3d and 2d coordinates. I also marked another new set of 8 points in similar fashion to create **test.csv**, which I will be using for testing projection parameters.

Procedure used

I converted the equations of projective geometry as discussed in class, to matrix multiplication using the concept of direct linear transformation. Then using its SVD decomposition, I calculated the eigenvector corresponding to the smallest eigenvalue, to give me the value of the optimal projection variables. Now using the first three columns of the projection matrix and its RQ decomposition, I got the K matrix which consists of the intrinsic parameters of the camera. Now writing K as the following matrix as discussed in class.

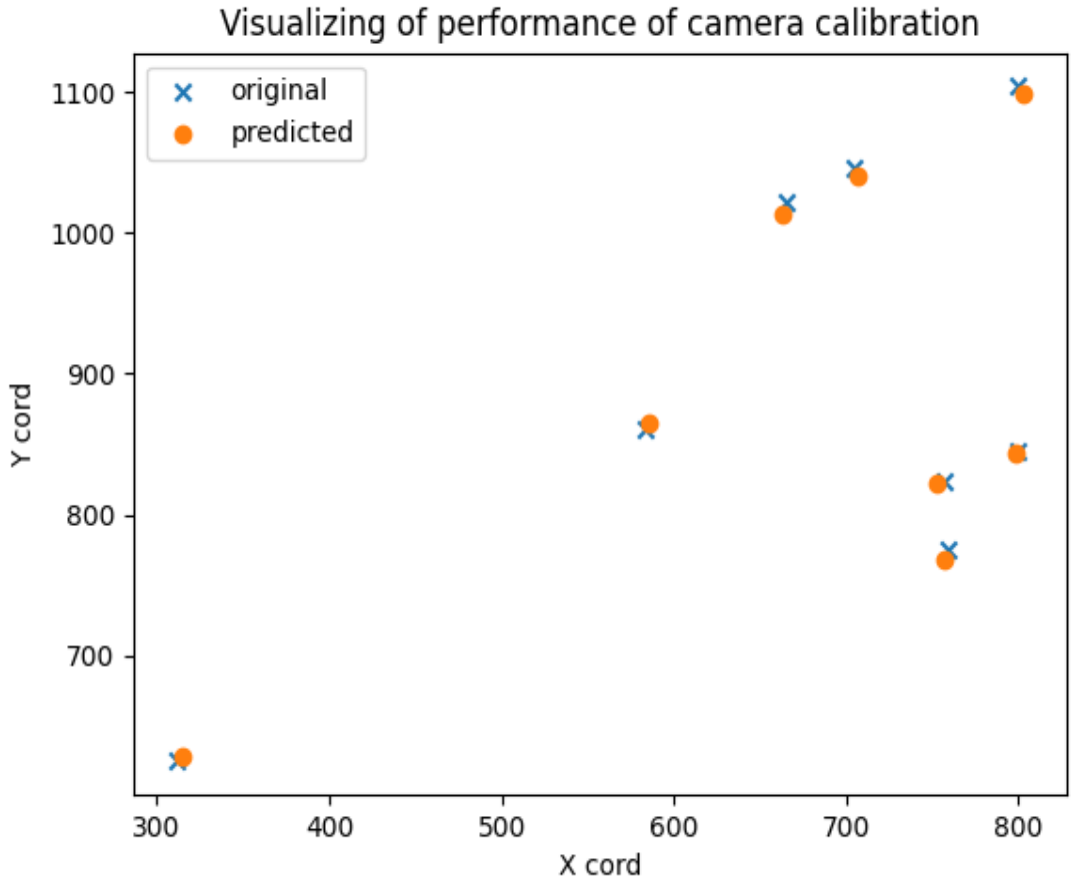
$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

I equated the parameters with obtained K, to get the final variable values in pixels as follows.

f_x	1249.23
f_y	1249.64
s	31.13
c_x	633.03
c_y	787.37

Testing

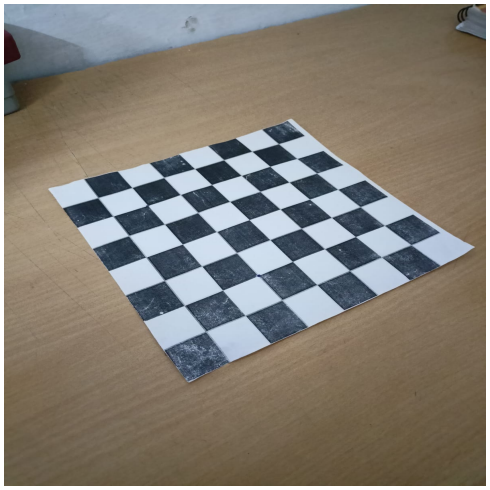
Here i used the **test.csv** dataset. The rmse error between the original(manually) and predicted(using projection matrix) points came out to be **5.87 pixels**. The figure below gives us a visual idea of the accuracy of our approach.



Q2

Dataset creation

Here I first took an image of the board placed on a table in **board.jpeg** and then later created the 3d-2d point coordinates pair like in Q1 using **2dpoints.py** in **board.csv**

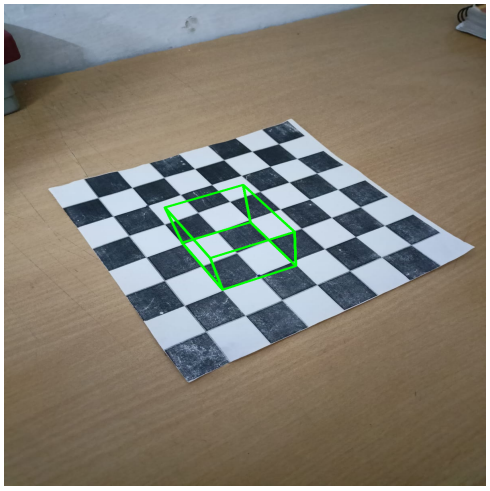


Procedure

Now what is different from the previous case is that we don't have non-zero z coordinates of the points this time. So we will create the equations of DLT using a 3x3 projection matrix P, which is sort of obtained after removing the 3rd column of our original 3x4 projection matrix. Now after solving the DLT equations we will insert a zero column at the 3rd place of our P and then multiply this P with inverse of K, which was obtained from the 1st question. Now if the whole matrix was correct P, then we would have obtained $R|t$. But because the third column was not solved for, we will obtain $R1|R2|0|t$. Now since R3 is supposed to be perpendicular to R1 and R2, we can obtain R3 as vector along R1 x R2. Then we again multiply the $R1|R2|R3|t$ matrix with K, to obtain the complete projection matrix P for this case. Now we can project the points of the required cube to obtain their corresponding 2d plane coordinates.

Testing

After generating the cube of side length as 2, the following image was obtained. Note that the cube's origin is be default set to be placed at (2,2) of the checkerboard.



References

- Lecture on DLT
- concepts of camera projection
- intrinsic parameters
- pose estimation