COL-780
ASSIGNMENT-3 REPORT

BY

KURISETI RAVI SRI TEJA

2019CS10369

Contents

1 Camera Calibration

- I initially shot a video of the chessboard pattern by placing it on a table.
- I then extracted some random frames from the video and then used them for calibration.

1.1 Finding Intrinsic Parameters

- Using Open-CV method **cv2.findChessboardCorners()**, I found the corners.
- Using the four extreme corners in each frame, I found the vanishing points in two perpendicular directions for each frame by solving the equations for lines.
- Using the vanishing points in perpendicular directions, I framed the equation $\mathbf{x_{d_2}^T K^{-1} K^{-T} x_{d_1} = 0}$ (discussed in class) where $\mathbf{x_{d_1}}, \mathbf{x_{d_2}}$ are vanishing points computed in previous step.
- Assuming that $\boldsymbol{\Omega} = \mathbf{K^{-1} K^{-T}}$,we have that $\mathbf{x_{d_2}^T} \boldsymbol{\Omega} \mathbf{x_{d_1}} = \mathbf{0}$.
- Let $\mathbf{x_{d_1}} = (\mathbf{x_1}, \mathbf{y_1}, \mathbf{1})$ **and** $\mathbf{x_{d_2}} = (\mathbf{x_2}, \mathbf{y_2}, \mathbf{1})$ and

$$\Omega = \begin{pmatrix} \Omega_{11} & \Omega_{12} & \Omega_{13} \\ \Omega_{21} & \Omega_{22} & \Omega_{23} \\ \Omega_{31} & \Omega_{32} & \Omega_{33} \end{pmatrix}$$

- Now we have that Ω is symmetric and assuming that skew is zero we get that $\Omega_{12} = \Omega_{21} = 0$, $\Omega_{13} = \Omega_{31}$ and $\Omega_{23} = \Omega_{32}$.
- Let $\Omega_{11} = a, \Omega_{13} = g, \Omega_{22} = b, \Omega_{23} = f, \Omega_{33} = c$

- We get that

$$\Omega = \begin{pmatrix} a & 0 & g \\ 0 & b & f \\ g & f & c \end{pmatrix}$$

- For frame-F with vanishing points $\mathbf{x_{d_1}}, \mathbf{x_{d_2}}$, using the fact that $\mathbf{x_{d_2}^T K^{-1} K^{-T} x_{d_1}} = \mathbf{0}$ we get that $\mathbf{\hat{x}_F \hat{\Omega}} = 0$ where

$$\mathbf{\hat{x}_F} = \begin{pmatrix} x_1 y_1 & x_1 + y_1 & x_2 y_2 & x_2 + y_2 & 1 \end{pmatrix} \textbf{ and } \mathbf{\hat{\Omega}} = \begin{pmatrix} a \\ g \\ b \\ f \\ c \end{pmatrix}$$

- Now on stacking up $\mathbf{\hat{x}_F}$ for all frames, we get that $\mathbf{\hat{X}\hat{\Omega}} = 0$ and solution for the equation is given by
$\mathbf{\bar{\Omega}} =$ last column of V where $\mathbf{\hat{X}} = \mathbf{U \Sigma V^T}$ is the SVD of $\mathbf{\hat{X}}$.

- Now I assumed that $\mathbf{\bar{\Omega}} = \lambda K^{-1} K^{-T}$(Here λ is a parameter for scale) and then directly solved for the entries in K i.e., f_x, f_y, u_x, u_y and also λ using the solution described in [1] which is given by

  1. $\mathbf{u_y} = (\mathbf{\bar{\Omega}_{12}\bar{\Omega}_{13}} - \mathbf{\bar{\Omega}_{11}\bar{\Omega}_{23}})/(\mathbf{\bar{\Omega}_{11}\bar{\Omega}_{22}} - \mathbf{\bar{\Omega}_{12}^2})$
  2. $\lambda = \mathbf{\bar{\Omega}_{33}} - (\mathbf{\bar{\Omega}_{13}^2} + \mathbf{u_y}(\mathbf{\bar{\Omega}_{12}\bar{\Omega}_{13}} - \mathbf{\bar{\Omega}_{11}\bar{\Omega}_{23}}))/\mathbf{\bar{\Omega}_{11}}$
  3. $\mathbf{f_x} = \sqrt{(\lambda/\mathbf{\bar{\Omega}_{11}})}$
  4. $\mathbf{f_y} = \sqrt{(\lambda\mathbf{\bar{\Omega}_{11}}/(\mathbf{\bar{\Omega}_{11}\bar{\Omega}_{22}} - \mathbf{\bar{\Omega}_{12}^2}))}$
  5. $\mathbf{u_x} = -\mathbf{\bar{\Omega}_{13}}/\mathbf{\bar{\Omega}_{11}}$

1.2 Finding Extrinsic Parameters

- For finding extrinsic parameters of each frame, I initially used the DLT to compute a homography **H** between the image points (x,y,1) and object points (X,Y,1)(Assuming that the object plane is the XY-plane for 3D-World Coordinate System and hence Z=0).

- We know that camera-matrix P=sK[R|t]=sK[r_1|r_2|r_3|t] where s is the scale, R=[r_1|r_2|r_3] is the rotation matrix and t is the translation matrix between image points (x,y,1) and 3D-world points (X,Y,Z,1).

- Hence we get that H=sK[r_1|r_2|t] and using K,H=[h_1|h_2|h_3] we get that

    1. $s = ||K^{-1}h_1|| = ||K^{-1}h_2||$
    2. $r_1 = \frac{1}{s}K^{-1}h_1$
    3. $r_2 = \frac{1}{s}K^{-1}h_2$
    4. $t = \frac{1}{s}K^{-1}h_3$
    5. $r_3 = r_1 \times r_2$

- The obtained R=[r_1|r_2|r_3] need not be a rotation matrix and hence it is approximated by the closest orthogonal matrix to R which is given by $\mathbf{\hat{R}} = \mathbf{UV^T}$ where $\mathbf{R} = \mathbf{U\Sigma V^T}$ is the SVD of matrix **R**.

- Hence the final camera parameters are $\mathbf{K}, \mathbf{\hat{R}}, \mathbf{t}$.

2 Placing the Object

- I then used the above-computed parameters $\mathbf{K}, \hat{\mathbf{R}}, \mathbf{t}$ to get the image coordinates of any 3D-object point (X,Y,Z,1) by using the formula $\hat{\mathbf{x}} = \mathbf{K}[\hat{\mathbf{R}}|\mathbf{t}]\hat{\mathbf{X}}$ where $\hat{\mathbf{x}} = (\mathbf{x}, \mathbf{y}, \mathbf{1})$ and $\hat{\mathbf{X}} = (\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{1})$ for every frame.

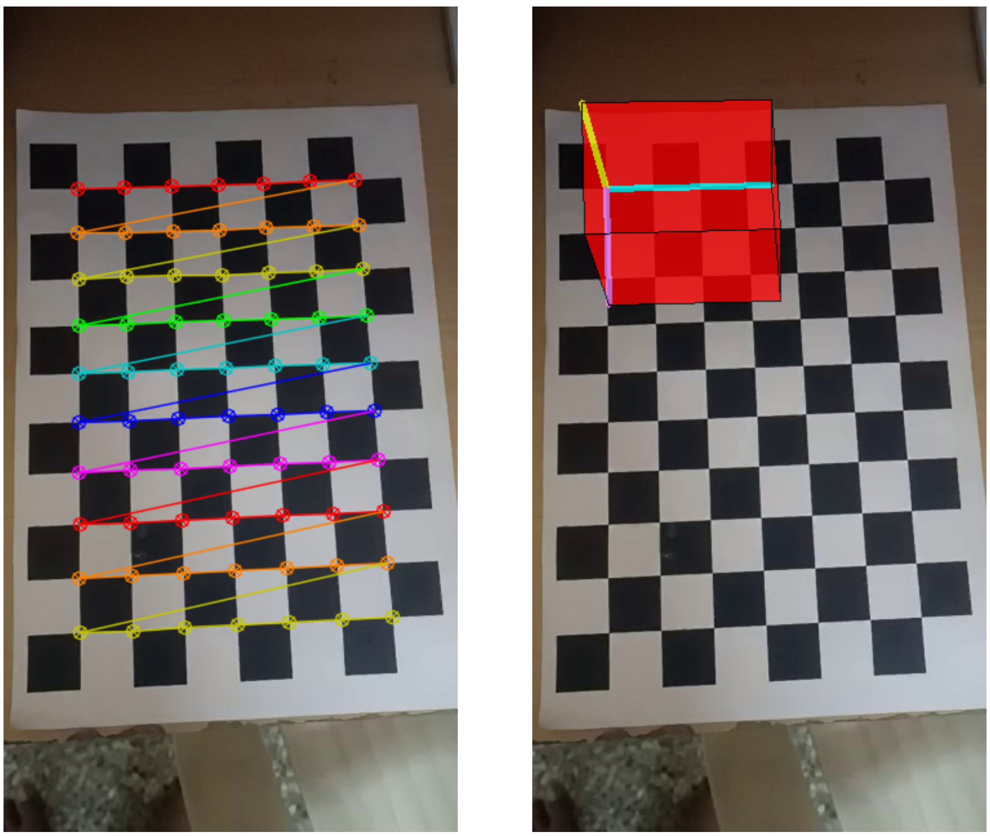
- I took the object to be a cube of size 3x3x3 and then projected all its 8 corners into the image-plane and joined the corners and faces to get an image of cube being placed on the chess-board.

3 Instructions to run the code

Use the command **python3 main.py arg1 arg2** to run the code where arg1 is the path to initial frames, arg2 is path to the output directory where the output images which contain a cube drawn on them are saved.

4 Results

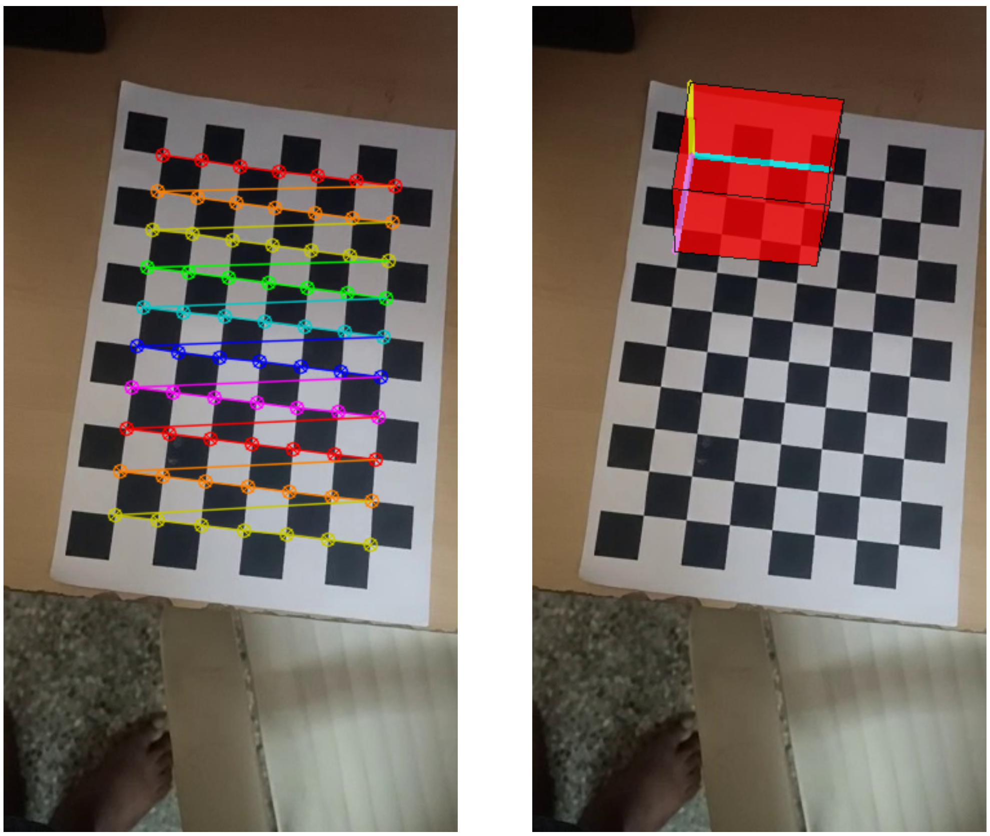
The X,Y,Z-axes are marked in different colours to show the location of origin.



(a) Marked Chess-board Corners (b) Cube placed on chess-board

Figure 1: Sample output of a frame with corners marked on left and cube placed at top-left corner

The X,Y,Z-axes are marked in different colours to show the location of origin.



(a) Marked Chess-board Corners (b) Cube placed on chess-board

Figure 2: Sample output of a frame with corners marked on left and cube placed at top-left corner

5 References

[1] **Zhang's Paper**

[2] **OpenCV Tutorials**