COL780 : Computer Vision
Assignment 3

Gaurav Jain : 2019CS10349

1 Method

I have implemented the method for finding the intrinsic parameters of a smartphone camera based on the method developed by Zhang et al. To calibrate the camera, I shot a video of the chessboard pattern by placing it on a table. I have taken frames from the video at a gap of almost 20 frames for calibration.
In the second part, we need to calculate extrinsic parameters of camera as well before inserting 3D objects in the frames. After calculating the camera matrix P, it is easy to project the corners of a cuboid and join them to form an image of a cuboid placed on the chessboard.

1.1 Camera Calibration: Finding Intrinsic Parameters

First, we find the corners of the chessboard using OpenCV's `findChessboardCorners()` function and find vanishing points in two perpendicular directions for all frames by calculating the point of intersection of two lines.
As discussed in the class, for perpendicular directions d_1 and d_2:

$$d_2 K^{-1} K^{-T} d_1 = 0$$

Let $\Omega = K^{-1}K^{-T} = \begin{pmatrix} \Omega_{11} & \Omega_{12} & \Omega_{13} \\ \Omega_{21} & \Omega_{22} & \Omega_{23} \\ \Omega_{31} & \Omega_{32} & \Omega_{33} \end{pmatrix}$
So,

$$d_2 \Omega d_1 = 0$$

We know that Ω is a symmetric matrix and we assume skew to be zero and thus, $\Omega_{12} = \Omega_{21} = 0$, $\Omega_{13} = \Omega_{31}$ and $\Omega_{23} = \Omega_{32}$.
We can write

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

Let $d_1 = (x_1, y_1, 1)$ and $d_2 = (x_2, y_2, 1)$. For frame F, using $d_2 \Omega d_1 = 0$, we get $\hat{x_F}\hat{\Omega} = 0$ where

$$\hat{x_F} = \begin{pmatrix} x_1 y_1 & x_1 + y_1 & x_2 y_2 & x_2 + y_2 & 1 \end{pmatrix} \text{ and}$$

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

Thus, stacking all frames we get $\hat{X}\hat{\Omega} = 0$. The solution is given by

$$\bar{\Omega} = \text{last column of } V \text{ where } \hat{X} = U\Sigma V^T \text{ is the SVD of } \hat{X}$$

According to Zhang et al, we can directly find intrinsic parameters K using $\bar{\Omega}$ given by,

$$u_y = (\bar{\Omega}_{12}\bar{\Omega}_{13} - \bar{\Omega}_{11}\bar{\Omega}_{23})/(\bar{\Omega}_{11}\bar{\Omega}_{22} - \bar{\Omega}_{12}^2)$$

$$\lambda = \bar{\Omega}_{33} - (\bar{\Omega}_{13}^2 + u_y(\bar{\Omega}_{12}\bar{\Omega}_{13} - \bar{\Omega}_{11}\bar{\Omega}_{23}))/\bar{\Omega}11$$

$$f_x = \sqrt{(\lambda/\bar{\Omega}_{11})}$$

$$f_y = \sqrt{(\lambda\bar{\Omega}_{11}/(\bar{\Omega}_{11}\bar{\Omega}_{22} - \bar{\Omega}_{12}^2))}$$

$$u_x = -\bar{\Omega}_{13}/\bar{\Omega}_{11}$$

1.2 Camera Calibration: Finding Extrinsic Parameters

We need to compute Homography H between the image points $(x, y, 1)$ and object points $(X, Y, 1)$. We do so using Direct Linear Transformation (DLT). Since $P = K[R|t]$ and $x = PX$ where R is rotation matrix and t is translation matrix, we get $H = K[r_1|r_2|t]$ as $Z = 0$ in world coordinate system. The final equations using K and $H = [h_1|h_2|h_3]$ are:

$$s = 0.5 * (||K^{-1}h_1|| + ||K^{-1}h_2||)$$
$$r_1 = \frac{1}{s}K^{-1}h_1$$
$$r_2 = \frac{1}{s}K^{-1}h_2$$
$$t = \frac{1}{s}K^{-1}h_3$$
$$r_3 = r_1 \times r_2$$

R is approximated by the closest orthogonal matrix to R given by UV^T where $R = U\Sigma V^T$ is the SVD of R.

1.3 Rendering Cuboid

After finding intrinsic and extrinsic parameters of the camera, we can write the camera matrix P. Thus, we can now project all the eight corners of a cuboid and join them to get an image of the cuboid being placed on chessboard.

2 How to run the code?

Run the following command:

```
python assign3.py input_dir_path output_dir_path
```

3 Generated Outputs

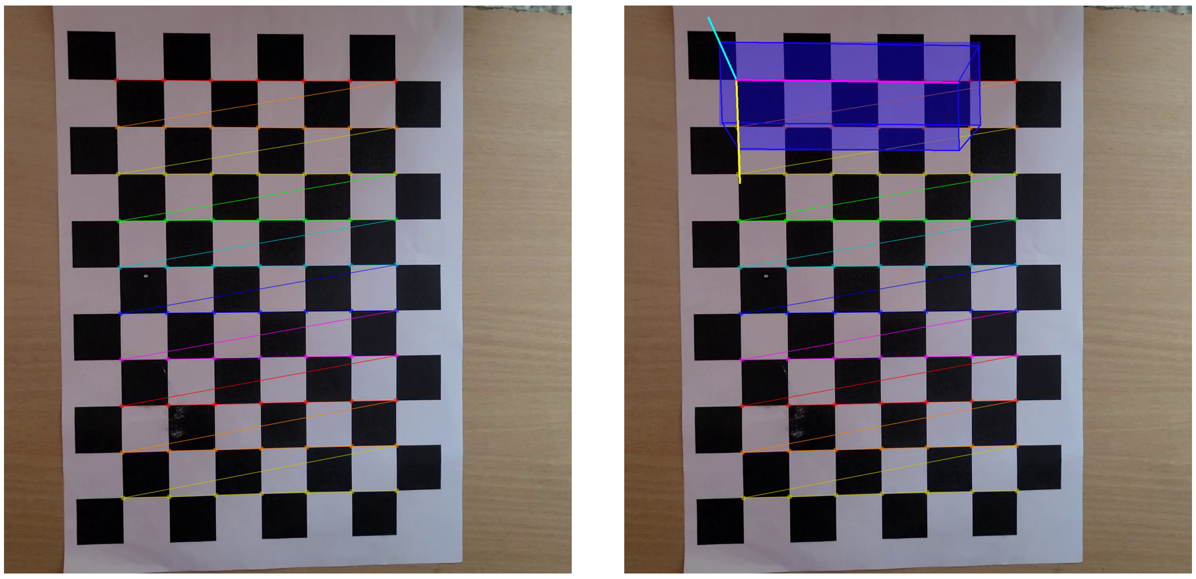
(a) Marked Corners on Chessboard (b) Rendered cuboid on Chessboard

Figure 1: Sample output frame with marked corners on chessboard and rendered with coordinate system

4 Link

This **link** contains the generated outputs of video frames in the original quality.

References

[1] Zhengyou Zhang, ”A Flexible New Technique for Camera Calibration”. 1998. Paper

[2] OpenCV Camera Calibration and 3D Reconstruction Tutorial. Link

[3] OpenCV Homography Tutorial. Link