Maxime Plumejeau
ID : 92454759
2024/05/01

Exercise : camera calibration

Camera calibration is a fundamental process in computer vision that aims to accurately determine the intrinsic and extrinsic parameters of a camera system. By calibrating a camera, we can correct distortions introduced by the camera lens and accurately relate the 3D world to the 2D image captured by the camera.

In the first part, the goal of this exercise is presented, then, the experimental method is explained. The obtained results are presented in a third part. After that, a discussion is conducted about the results, and a conclusion ends this report.

I - Goal

The goal of this exercise is to calibrate the camera of my smartphone (Redmi note 9 pro) using Zhang's method.

Calibrating a camera means determining its intrinsic parameters, which include the focal length f and the coordinate of the optical center (u_0, v_0). They are parameters defining the optical properties of the camera. From a given point in the camera 3D coordinate system, the intrinsic parameters allow to obtain the corresponding point in the digital image sensor 2D coordinate system. They are put into the intrinsic matrix K.

However, for computer vision applications, it is more convenient to determine the corresponding point in the digital image sensor 2D coordinate system from a world 3D coordinate system, distinct from the camera one, and consider stationary. For this purpose, extrinsic parameters allow to obtain the corresponding point in the camera 3D coordinate system from the point's position in the world 3D coordinate system. These parameters are composed of rotation parameters and translation parameters, and they are put into the [R|t] matrix.

The projection matrix P is obtained by the multiplication of the matrixes K and [R|t] and is used to obtain the corresponding point in the digital image sensor 2D coordinate system from a world 3D coordinate system.

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}] = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{21} & R_{31} & t_X \\ R_{12} & R_{22} & R_{32} & t_Y \\ R_{13} & R_{23} & R_{33} & t_Z \end{bmatrix}$$

Figure 1 : the projection matrix P

The intrinsic matrix K depends on the camera properties, hence, it is the same for all the pictures taken with a given camera. However, the intrinsic matrix [R|t] depends on the

position and angle of the camera when the picture was taken, hence, it can be different for different pictures.

For this exercise, I have taken several pictures of a chessboard image with my smartphone's camera. Then, the objectives were to determine the intrinsic parameters of the camera using these images, and to determine the extrinsic parameters for each image in order to draw the xyz axes into them.

II - Method

A useful and effective way to calibrate a camera is Zhang's method. It gives the intrinsic and extrinsic parameters from pictures taken at different positions and angles of the same chessboard. This method is implemented in the OpenCV library, hence, I have developed a Python script using OpenCV to complete the camera calibration.

With several chessboard pictures as input, this script computes the intrinsic and extrinsic parameters of the camera, and draws the xyz axes on the input pictures, assuming the world coordinate is defined by a corner of the chessboard. Let's have a rough analysis of the script.

```python
import numpy as np
import cv2 as cv
import glob
import os

nb_success = 0

# number of corners
width = 9
height = 7

# termination criteria
criteria = (cv.TERM_CRITERIA_EPS + cv.TERM_CRITERIA_MAX_ITER, 30, 0.001)

# prepare object points, like (0,0,0), (1,0,0), (2,0,0) ....,(6,5,0)
objp = np.zeros((height*width,3), np.float32)
objp[:,:2] = np.mgrid[0:width,0:height].T.reshape(-1,2)

# Arrays to store object points and image points from all the images.
objpoints = [] # 3d point in real world space
imgpoints = [] # 2d points in image plane.
```

Figure 2 : first part of the script - setting up some variables

This first part of the script sets up some variables used later, such as the number of corners of the chessboard and arrays to store 3D/2D points for the calibration. Moreover, it sets up the criteria used for finding the chessboard corners in the second part.

```python
images = glob.glob('images/set_3/*.jpg')

for fname in images:

    img = cv.imread(fname)
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    # Find the chess board corners
    ret, corners = cv.findChessboardCorners(gray, (width,height), None)

    # If found, add object points, image points (after refining them)
    if ret == True:

        objpoints.append(objp)
        corners2 = cv.cornerSubPix(gray,corners, (11,11), (-1,-1), criteria)
        imgpoints.append(corners2)

        # Draw and display the corners
        cv.drawChessboardCorners(img, (width,height), corners2, ret)
        resized_img = cv.resize(img, (928, 522))
        cv.imshow('img', resized_img)
        cv.waitKey(500)

        # Save the calibrated image
        filename = os.path.join('images_calibrated', os.path.basename(fname))
        cv.imwrite(filename, img)

        nb_success += 1
```

Figure 3 : second part of the script - finding chessboard corners

The second part of the script is a *for* loop on each input image. The purpose is to find each corner of the given image using the function *cv.cornerSubPix* and to store these 2D points into the array *imgpoints*.

```python
# Compute the calibration parameters
ret, mtx, dist, rvecs, tvecs = cv.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None, None)

rmats = []
for rvec in rvecs:
    R, _ = cv.Rodrigues(rvec)
    rmats.append(R)
```

Figure 4 : third part of the script - computing the calibration parameters

The third part of the script uses the function *cv.calibrateCamera* to compute the calibration parameters of the camera from the chessboard corners found previously into the input pictures. After that, the function *cv.Rodriguez* is used to convert the rotation vector *rvecs* into the rotation matrix *rmats*. Hence, the intrinsic parameters of the camera are stored in the matrix *mtx*, while the extrinsic parameters of each image are stored in the array of matrix *rmats* for the rotation parameters and in the array of vector *tvecs* for the translation parameters.

```python
# Draw axis in each image
for i in range(nb_success):

    img = cv.imread(images[i])
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    # Project 3D axis points onto the image
    axis_points, _ = cv.projectPoints(np.float32([[3,0,0], [0,3,0], [0,0,-3]]), rvecs[i], tvecs[i], mtx, dist)
    origin = tuple(imgpoints[i][0].ravel())  # Origin is the first corner of the chessboard

    # Draw and display the corners
    img = cv.drawFrameAxes(img, mtx, dist, rvecs[i], tvecs[i], 2)
    resized_img = cv.resize(img, (928, 522))
    cv.imshow('img', resized_img)
    cv.waitKey(500)

    # Save the image with axis
    filename = os.path.join('images_axis', os.path.basename(images[i]))
    cv.imwrite(filename, img)
```

Figure 5 : fourth part of the script - drawing the xyz axes on each picture

The last part of the script uses the results of the calibration to draw the xyz axes on each input picture, assuming the world coordinate system is defined by the chessboard used for the calibration.

III - Results

I have taken 12 pictures of a chessboard image from different positions and angles with my smartphone camera in order to calibrate it. Here are the input pictures I gave to the Python script.

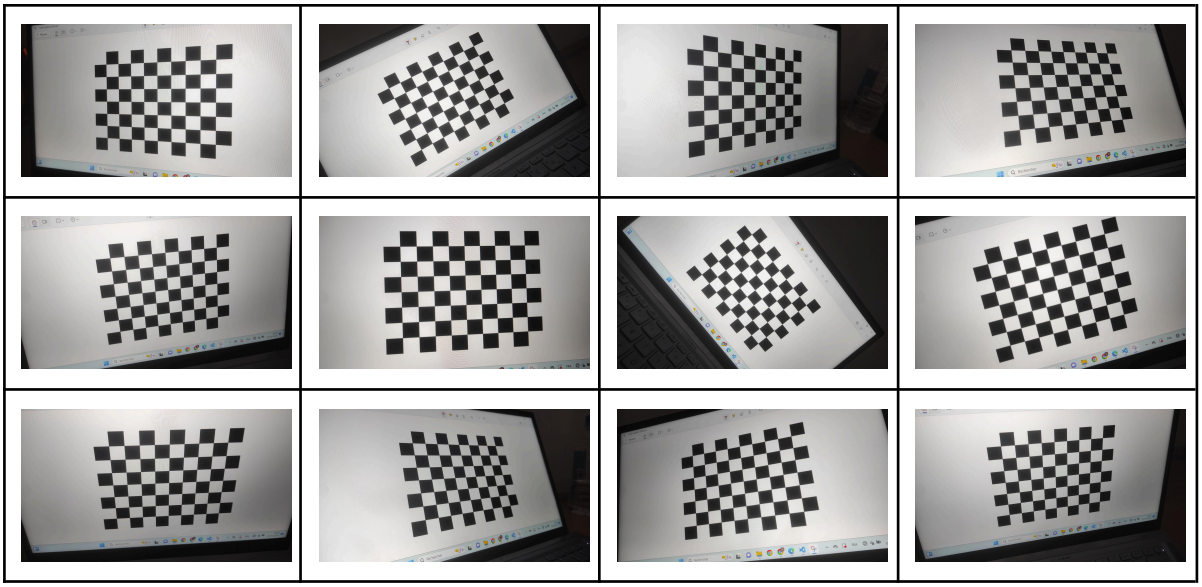

Figure 6 : the 12 chessboard input pictures used for the calibration

The script gives me several outputs. Firstly, it gives back the chessboard images with the corners detection shown. Indeed, each corner is marked with a colored dot if it has been found by the script.

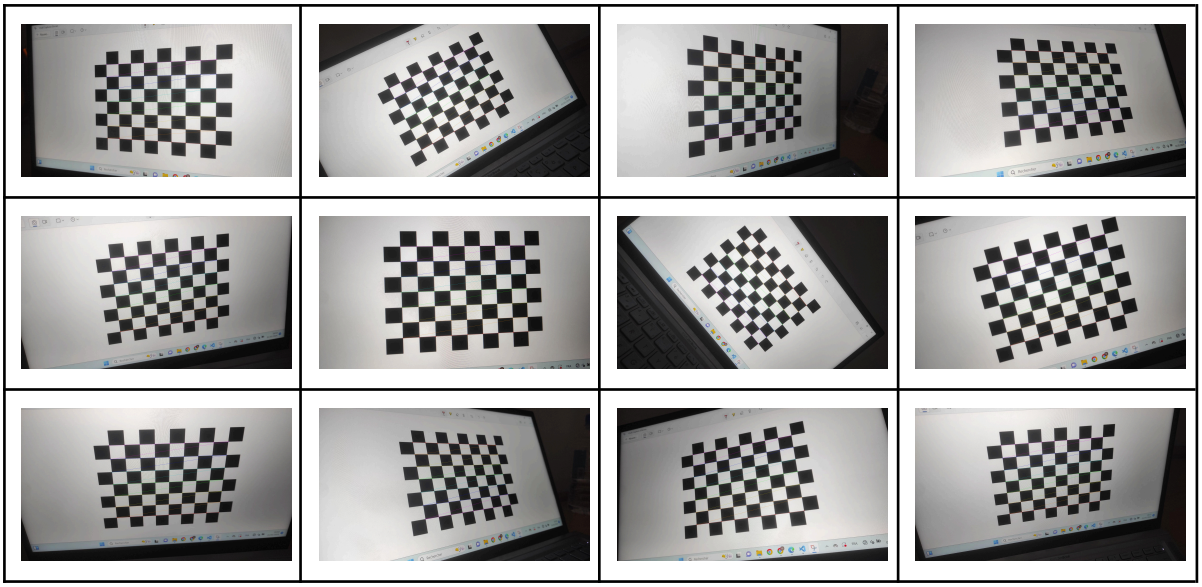

Figure 7 : the 12 chessboard pictures with corners detected

Here are the output corners detection images of the script. We can hardly see the effective corners detection due to the high resolution of the initial pictures, however, it is possible to see it with a bigger image or a zoom (as shown with the figure 8). Hence, we can conclude that the chessboard corners have been well detected for each image.

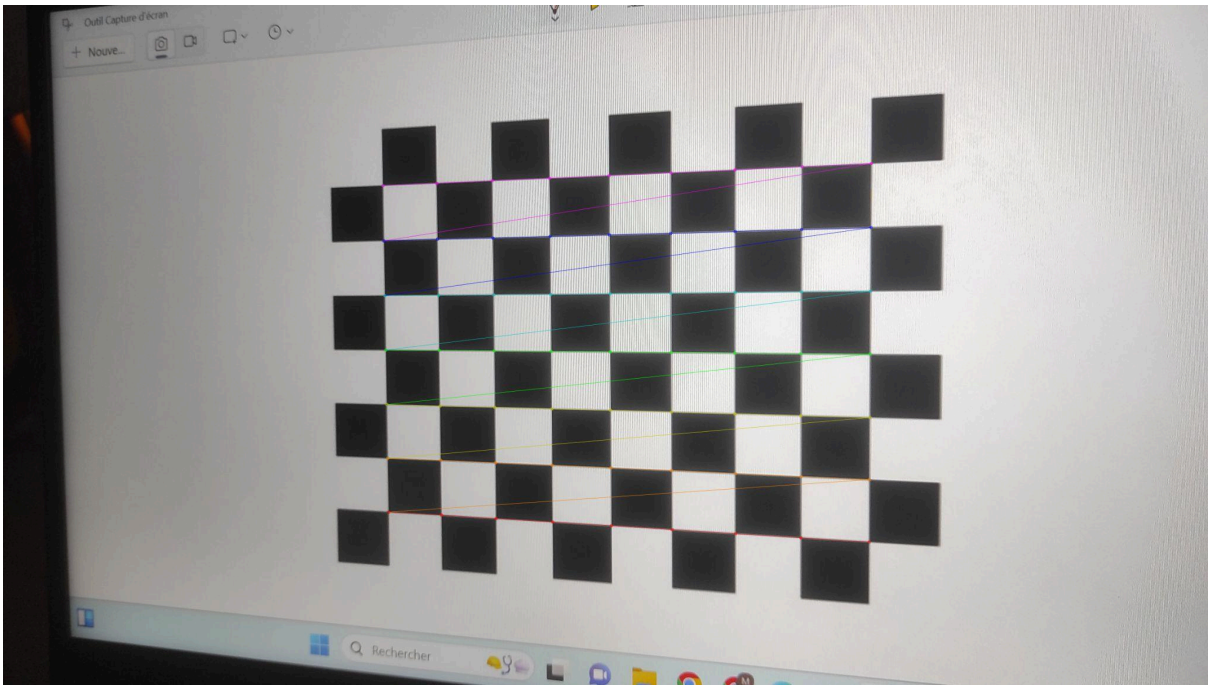

Figure 8 : a chessboard picture with corners detected in bigger size

Another output of the script is the calibration parameters. It gives the intrinsic matrix K associated to the camera and the extrinsic matrixes [R|t] associated with each input picture (on the figure 9, for each image, the matrix R is given above the vector t).

```
intrinsic parameters :
[[3.78832693e+03 0.00000000e+00 2.05826430e+03]
 [0.00000000e+00 3.54100386e+03 1.11470974e+03]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

extrinsic parameters :

image  0
[[-0.9424779  -0.0120747   0.33405031]
 [ 0.00617915 -0.99980594 -0.01870572]
 [ 0.33421135 -0.01556558  0.94236961]]
[[ 4.98067717]
 [ 3.95241184]
 [14.44190099]]

image  1
[[-0.87214401 -0.4619781  -0.16106229]
 [ 0.4696397  -0.88279026 -0.0109503 ]
 [-0.13712542 -0.08519149  0.98688349]]
[[ 5.69324679]
 [ 1.40548435]
 [19.21946768]]
```

Figure 9 : calibration parameters

Hence, it is possible to extract the focal length f and the coordinate of the optical center from the matrix. I obtain for my smartphone's camera :

$f = 3.66 * 10^3$ pixels (average of both f computed)

$u_0 = 2.06 * 10^3$ pixels

$v_0 = 1.11 * 10^3$ pixels

Finally, a last output given by the script is the initial pictures modified with the xyz axes drawn on them. It allows to verify the correctness of the camera calibration, indeed, the axes are correctly drawn on each image, proving that both intrinsic parameters and extrinsic parameters have been correctly determined in the previous steps.

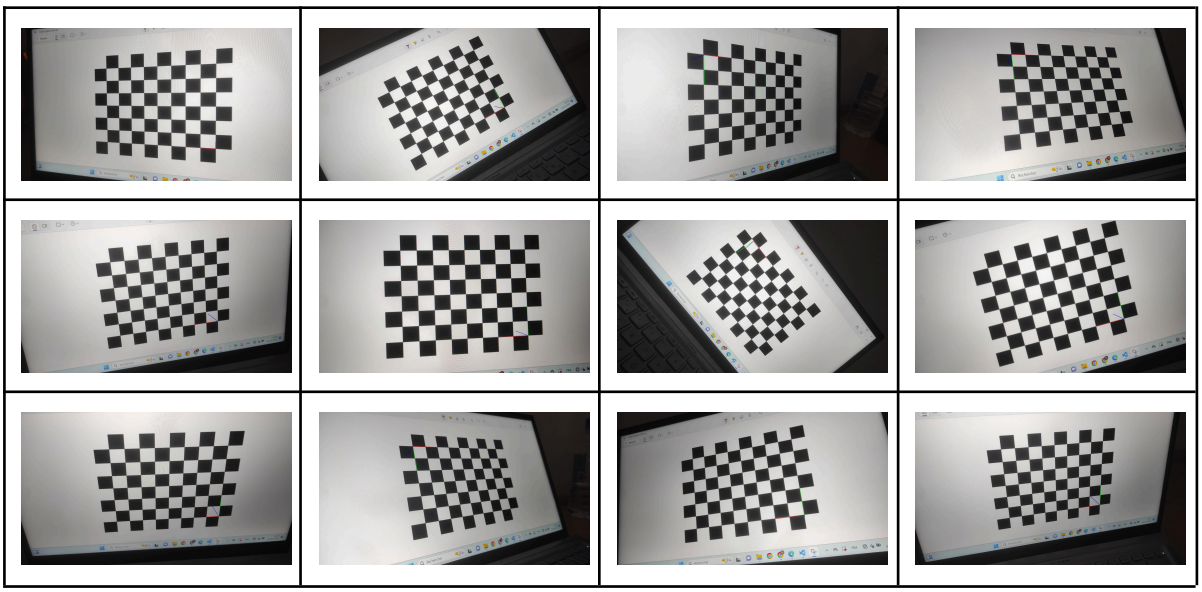

Figure 10 : the 12 chessboard pictures with the xyz axes drawn

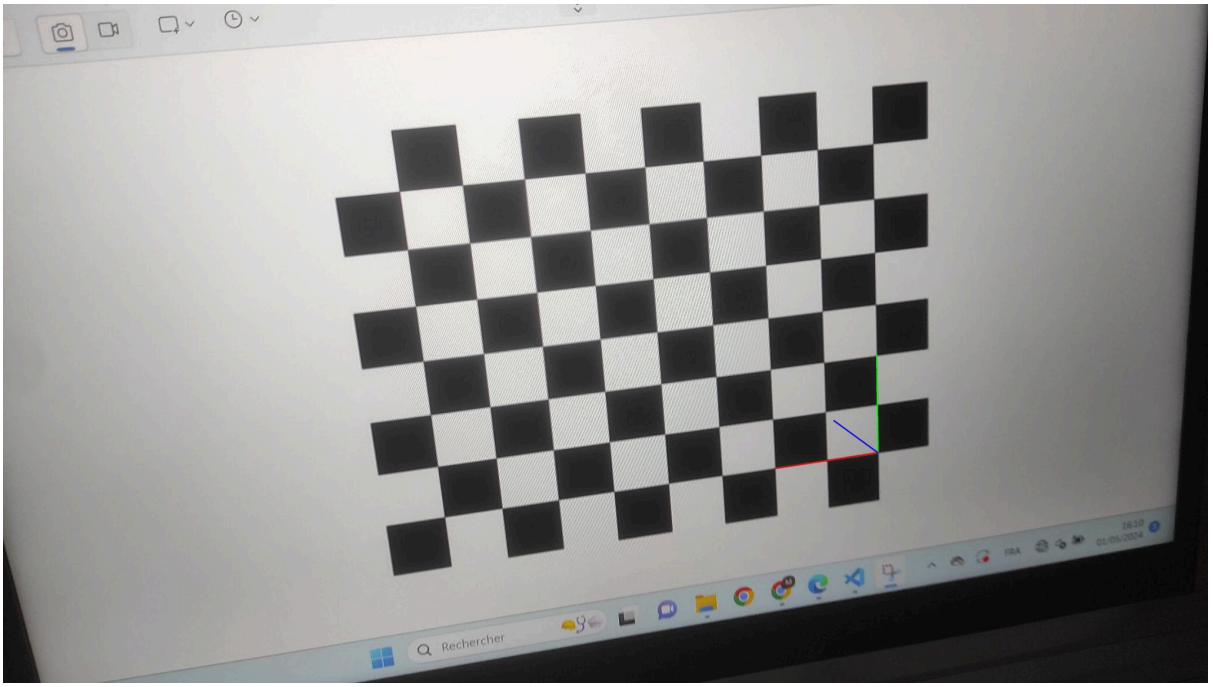

Figure 11 : a chessboard picture with the xyz axes drawn in bigger size

IV - Discussion

After having performed the camera calibration, it is possible to compare the computed focal length f with the theoretical focal length f_{th} available in the spec sheet of my smartphone. I found :

f_{th} = 4.7 mm

It is possible to convert this value in pixels with the following formula :

$f_{th,pix} = f_{th,mm} / pp$

with pp the pixel pitch, the distance between 2 pixels on the camera sensor, in mm

I found pp = 1.6 µm for my smartphone's camera, thus, I obtain :

f_{th} = 4.7 mm = 2.94 * 10^3 pixels

It is now possible to compute the relative error of the computed focal length f :

$E = |f_{th} - f|/f_{th} * 100 =$ 24%

The computed relative error is actually pretty huge, but the obtained result is still of the same order of magnitude as the theoretical value. The error may have several factors, such as the quality, the light or the distortion of the input pictures.

V - Conclusion

To conclude this exercise, I have successfully calibrated my smartphone's camera using Zhang's method. From chessboard pictures, I have been able to obtain the intrinsic and extrinsic parameters of the camera and to draw the xyz axes using a Python script and the OpenCV library.