Computer Vision Assignment 4 (2018/19)

Pedro Santos, *Universidade de Aveiro, 76532* Silvério Pereira, *Universidade de Aveiro, 76505*

***Index Terms*—OpenCv,c++,assigment_3**

I. Introduction

THIS report has a description and explications about the experiences done in the lesson number four and the topic covered is Camera calibration.

II. Exercises

A. Exercise 1

In the first exercise it is necessary to compile and test the file chessboard.cpp. This code is detecting the corners in a chessboard pattern using openCV functions and shows the results of the detection for a series of images. The next thing that it was necessary to do was to calibrate the camera using the function calibrateCamera. To save the results of the camera calibration process it is necessary to create four variables that are translation_vectors, rotaction_vectors, distCoeffs and intrinsic. This variables will be passe in the function calibrateCamera to save there the result. The other two arguments used are object_points, image_points and the image size. The variable object_points has the teorical value of the coordinates of each corner and the image_points has the value of the coordinates of the corners detected by the function FindAndDisplayChessboard. Lastly, the values of the results of the camera calibration are printed. and the values of the intrinsic and distortion matrices are saved in an xml file.

If the pattern was different to get the distance correctly evaluated it would be necessary to put in the vector the coordinates 3D of the new pattern.

B. Exercise 2

In the second exercise was implemented code to project an orthogonal line (normal) in the provided images. To do that for each image the image is read, then it is used the function projectPoints that receives the results of the camera calibration, the coordinates of the points to project and it is also passed the vector where it will be saved the corresponding coordinates of the points in the image. The, it is used the function line from openCV to project the orthogonal line in the image. The arguments of this function are the image, the two points that were calculated using the function project points, the color, the thickness and the type of the line. The last step was to show the final result. The final result of the first 5 images can be seen in the figure **??**, **??**, **??**, **??** and **??**. The other results are not presented in the report because they do not add value and to the report do not become so unnecessarily extensive.

Fig. 1. orthogonal line in image 1

Fig. 2. orthogonal line in image 1

Fig. 3. orthogonal line in image 1

C. Exercise 3

In this exercise the goal is to use the camera from our computer to process the chessboard. To solve this challenge the frames captured from the web cam are analyzed and if detects all the corners of the chessboard it will save the coordinates and it will count one more success. Then, it will wait from a

Fig. 4. orthogonal line in image 1

Fig. 5. orthogonal line in image 1

key to be pressed. After 13 success this part of the program ends and it will calibrate the camera using the information of the images captured using the web cam. Doing like this is much easier because at least in web cams that are not so good or the light is not good it is hard to captured all the corners, so by doing like this it is possible to be concentrated in trying to put the chessboard in the best possible way to be captured and then it will captured without the need to press a key. Then, when we are ready to capture the next frame it is pressed a key. The results of two frames are presented in the 7 and **??**. The number of success necessaries are thirteen to be equal to the number of images, so it is easier to change the code to read from images.

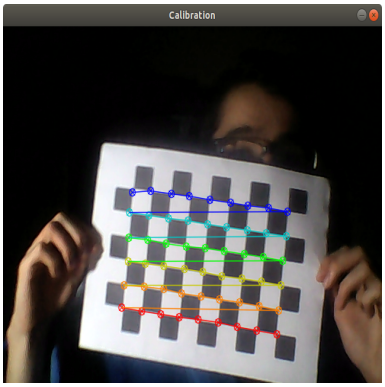

Fig. 6. Image 1 captured by web cam

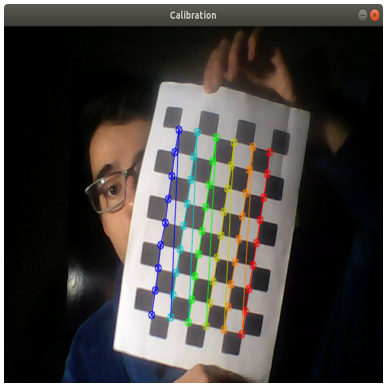

Fig. 7. Image 2 captured by web cam

D. Exercise 4

In the last exercise was necessary to calibrate the camera and save the camera parameter file in a xml file. Then, it was necessary to read the intrinsic and distortion parameters from the file and perform external parameters calibration (using function solvePnP). The first time that the program is run it will create the xml file. In the beggining of the program it will check if there is the xml file, if there isn't a xml, it will do the normal process, if there is it will perform external parameters calibration. So, if there is the XML file the first thing that is done is read from the xml file the two matrix. One of the matrix is the intrinsic_matrix and the other one is distortion_coeffs. After this step for each image it will be calculated the corners using the function FindAndDisplayChessboard like in the others examples. However, now after this it will be used the function solvePnP. This function receives the coordinates of the corners, the coordinates 3d of the pattern, the two matrix read from the xml file and it will be passed the matrix of the rotation_vectors and translation_vectors to save there the correspondents values. Then, it is possible to project the line like in the previous exercise by using the projectPoints and line functions. Some of the used images can be seen in the figures 8, 9 and 10.

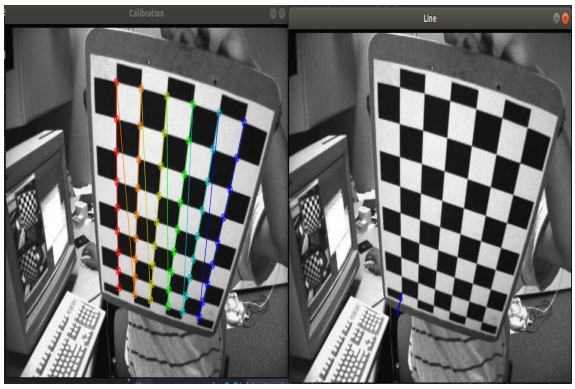

Fig. 8. Image 1 using External calibration

E. Extra

The last thing done for this class was to project the orthogonal line using the web cam and see the result in real

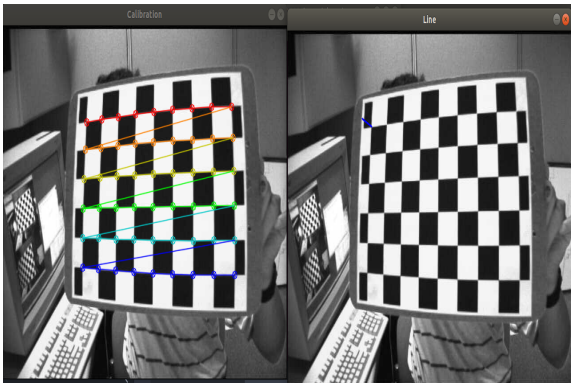

Fig. 9. Image 2 using External calibration

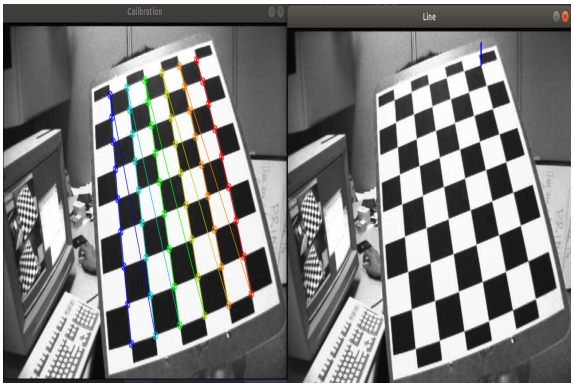

Fig. 10. Image 3 using External calibration

time. To accomplish this challenge was made a infinity cycle where in each cycle it will be captured a frame and it each frame captured will be analyzed and if all the corners are found in the frame the camera will be calibrated, the coordinates of the project points will be calculated and then the line will be placed in the image. This will create a real time projection. The result is much more explicit in video, however in the figure 11 and 12 can be seen two prints of the program running. In one of the windows the corners are displayed and in the other window the orthogonal line in the chessboard.

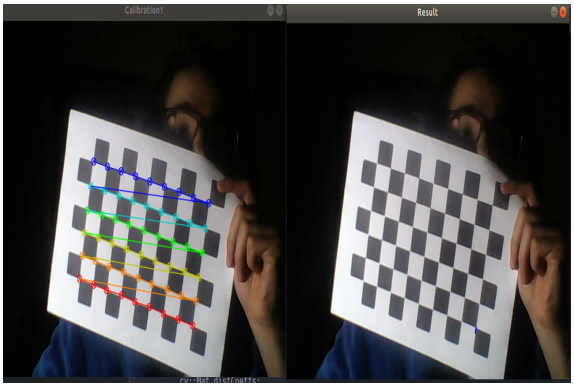

Fig. 11. Image 1 using real time projection

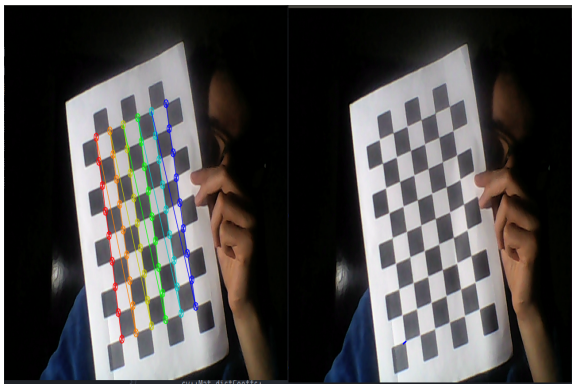

Fig. 12. Image 2 using real time projection