# Lab 06, Stereo Vision

João Santos, *MRSI, 76912*

***Index Terms*—OpenCV, Computer Vision, MRSI, UA, DETI, LaTeX.**

## I. Introduction

THIS report is intended to be used alongside the Python3 code developed for this Lab.

The Lab #06 is a introduction class to stereo vision, OpenCV and Python 3.

This report was written using LaTeX .

## II. Exercises

Lets analyse the resolution of the proposed exercises.

### A. Ex. 6.1: Chessboard calibration

This exercise is based on the examples of the previous class. The minor difference is that, is this one, it was required to obtain the calibration parameter for both a left and a right camera simultaneously. In order to do so, some changes had to be made to the provided code in order to store the image points of both cameras.

This information will be used in the next exercise.

### B. Ex. 6.2: Stereo calibration

The goal of this exercise was to use *cv2.stereoCalibrate()* to obtain the intrinsic parameters of both camera, simultaneously.

To be possible to the calibration algorithm to converge, we had to indicate to the refereed method to use the same focal length, which comes from the fact stated earlier.

The output for the intrinsic matrices were

$$right = \begin{bmatrix} 536.826 & 0 & 334.914 \\ 0 & 535.937 & 242.392 \\ 0 & 0 & 1 \end{bmatrix}$$

and

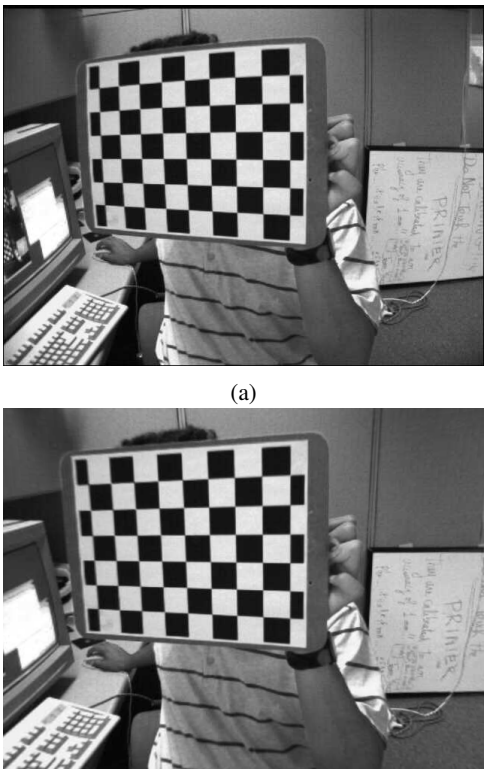
$$left = \begin{bmatrix} 536.826 & 0 & 335.265 \\ 0 & 535.937 & 241.698 \\ 0 & 0 & 1 \end{bmatrix}$$

which are, as expected, very similar given that both camera are of the same specification.

This method also outputs the distortions coefficients for both cameras alongside with rotation, translation, essential and fundamental matrices from between both cameras.

### C. Ex. 6.3: Lens distortion

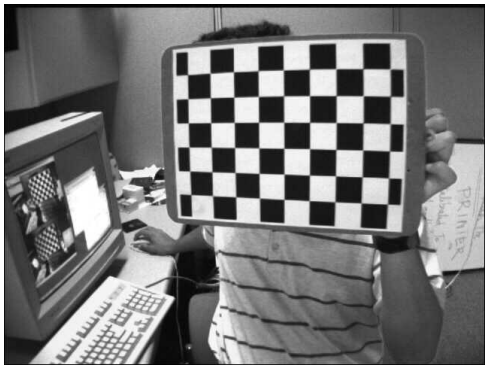
Using the parameters obtained and stored on the previous exercise, we are now able to rectify the lens distortion from both cameras. Figs. 1 and 2 show the output of the operation, in both cameras.



(a)

(b)

Fig. 1: Lens distortion correction for the right camera. (a) Original image and (b) rectified image.



(a)

(b)

Fig. 2: Lens distortion correction for the left camera. (a) Original image and (b) rectified image.

### *D. Ex. 6.4: Epipolar lines*

Building upon the previous exercise, we should now complete it in order to now compute and draw the epipolar lines.

Notice that the method *cv2.computeCorrespondEpilines()* requires, as an argument, the index of the image that contains the points.

Figs. 3 and 4 show to examples of epipolar lines.

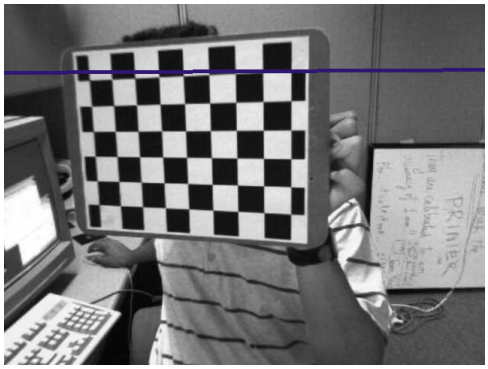


Fig. 3: Epipolar line selected on the left image and drawn on the right one.

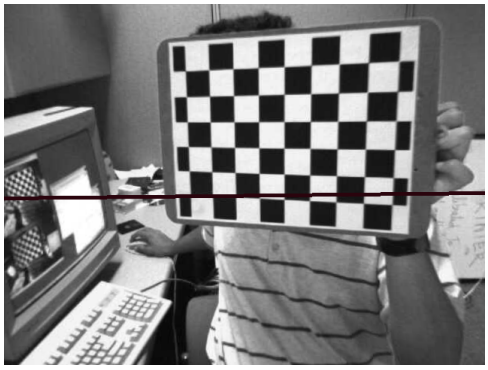


Fig. 4: Epipolar line selected on the right image and drawn on the left one.

### *E. Ex. 6.5: Image rectification*

In the final exercise of this class, the goal was to rectify the images using rectification maps so that, when drawing epipolar lines, they would be not only on the same row within the image but also on both images too.

Figs. 5 and 6 show the outcomes. This particular example shows that the calibration procedure had some mismatch because the selected point in one image was (marginally) outside the epipolar line on the other one.

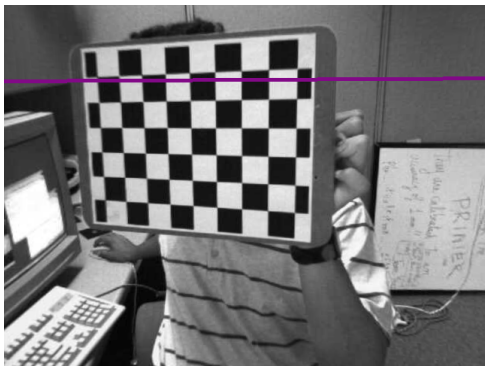


Fig. 5: Epipolar line selected on the left image and drawn on the right one.

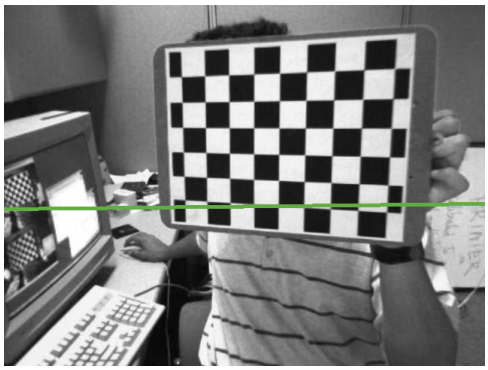


Fig. 6: Epipolar line selected on the right image and drawn on the left one.