Stereo Vision (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 six and the topic covered is Stereo Vision

II. Exercises

A. Chessboard Calibration

To get the distance correctly evaluated with another pattern with different size values we could initialize the 3D coordinates to be in multiples of the distance between corners instead of it being percentage of the number of board corners as it is in the example

B. Stereo Calibration

In this exercise we calibrate the stereo pair using the function `cvStereoCalibrate`, for successful calibration the vector it needs:

- The imagined 3D points where the calibration points (points in the same 3D place in both stereo images, from which we will calibrate)
- The calibration points of the first image
- The calibration points of the second image
- The number of calibration points
- The image size

and outputs:

- intrinsic and extrinsic parameters for the first and second images
- The output rotation matrix between the second and first camera coordinate systems, R
- The output translation matrix between the second and first camera coordinate systems, T
- The output essential matrix, E
- The output fundamental matrix, F

There are also some flags we can specify, of which most notably the `CV_CALIB_FIX_INTRINSIC` was used, which fixes the intrinsic and extrinsic parameters of the cameras associated with both images and calculates and calculates parameters R, T, E, F accordingly

The calculated parameters are then saved to a xml file `stereoParams.xml` for later use

C. Image Rectification

For this exercise the parameters computed in II-B were loaded and passed to the `undistort` function, it uses the intrinsic and distortion matrices to compute the undistorted image from the original, the results for a pair of stereo images can be seen in 3 and 4 and were obtained from 1 and 2 respectively

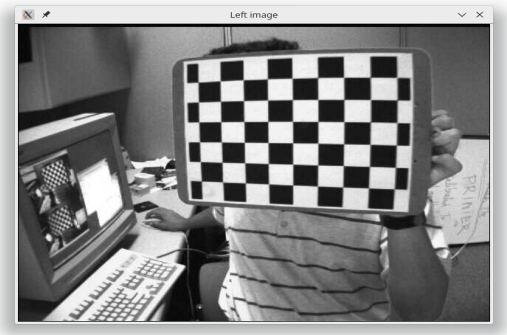

Fig. 1. Left image

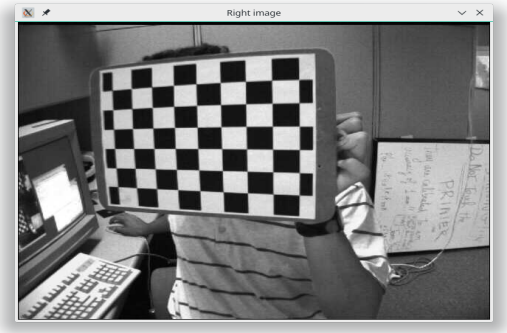

Fig. 2. Right image

Fig. 3. Left image undistorted

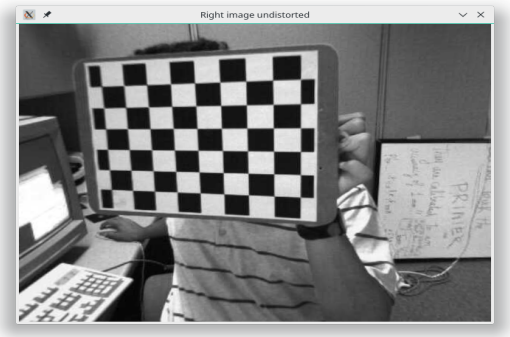

Fig. 4. Right image undistorted

D. Image Rectification

To rectify both images, `cvStereoRectify` is used to first convert both images to the same image plane, outputing the rotation and projection matrices. `cvInitUndistortRectifyMap` is then used to compute the transformation needed to rectify both images, these transformations are then applied with the `Remap` function, the results can be seen in 5 and 6

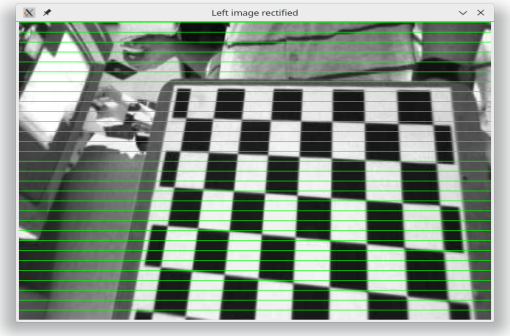

Fig. 5. Left image rectified

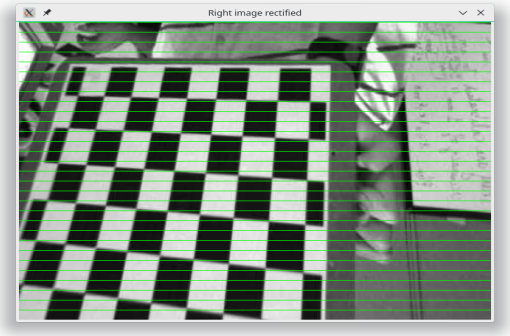

Fig. 6. Right image rectified