

LAB 2

In this lab is requested to calibrate a camera and then to fix a distorted test image.

The camera calibration requires to take some pictures of a specified pattern (in our case a chessboard), but due to some issues (my personal printer is not working very well) I have had to use the provided dataset.

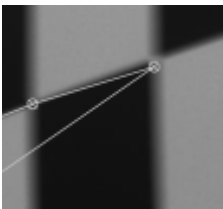
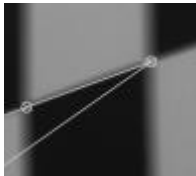
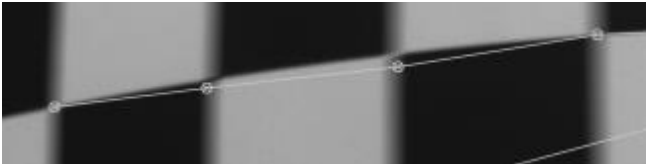



Images loading and points detection

After the program loads the images, it starts to compute the inner chessboard corners into 3D points and 2D points for each acquired image.

The 3D points are all the same for each picture: the 3D world origin frame is set in the inner upper left corner of the chessboard; the X axis is going along the chessboard columns; the Y axis is going along the chessboard rows and the Z axis is set as outcoming from the chessboard (obviously all the axis are orthogonal among each other), this means that the Z coordinate is 0 for all the points. This process is done by the function `points3Dinitialization`, created by me.

The 2D points are computed by using the `findChessboardCorners` OpenCV function. The function provides a flag parameter to set some operations during the process of finding the corners: I have decided to use the default values plus the `CALIB_CB_FAST_CHECK` to speed up the program.

Once I have completed this part, I have created a function `showCorners` that allows me to use `drawChessboardCorners` to check empirically the quality of the corner detection. In fact, I have noticed some bad results on corner detection; so I have decide to use `cornerSubPix` to improve the results before proceeding with the camera calibration part. After some empirical tests, I have decided to set the `winSize` parameter in `cornerSubPix` with (22,22); this value is sufficient to have good results.

IMG	BEFORE <code>cornerSubPix</code>	AFTER <code>cornerSubPix</code>
16		
27		
45		

Camera calibration

Once the program has computed the 3D and 2D points, it proceeds to compute the intrinsic and distortion parameters of the camera and the extrinsic parameters of each acquired image by using `calibrateCamera`.

With the new parameters (intrinsic, extrinsic and distortion), the program reprojects all the 3D points in the new 2D points and it computes the Euclidean distance among the old 2D points and the new ones: this allows to have a numerical error and it's calculated the mean error for each image to show which image has a the best and worst performance in the calibration. It is also computed and shown the RMS error (process mostly done by the `calibrateCamera` function) because it's more accurate than the mean error. This allows to notice that, with the provided dataset, the mean and RMS errors provide the same image as best performing one, instead the worst performing image is different.

Test image undistortion and correction

With the computed camera matrix and the distortion parameters, the program uses `initUndistortRectifyMap` to define a map function that allows to remap the distorted test image into an undistorted one. This is done by using `remap` function and the result is shown in a single window, next to the test distorted image. The two images are resized to properly fit them into the screen.



test image (distorted)

result image (undistorted)