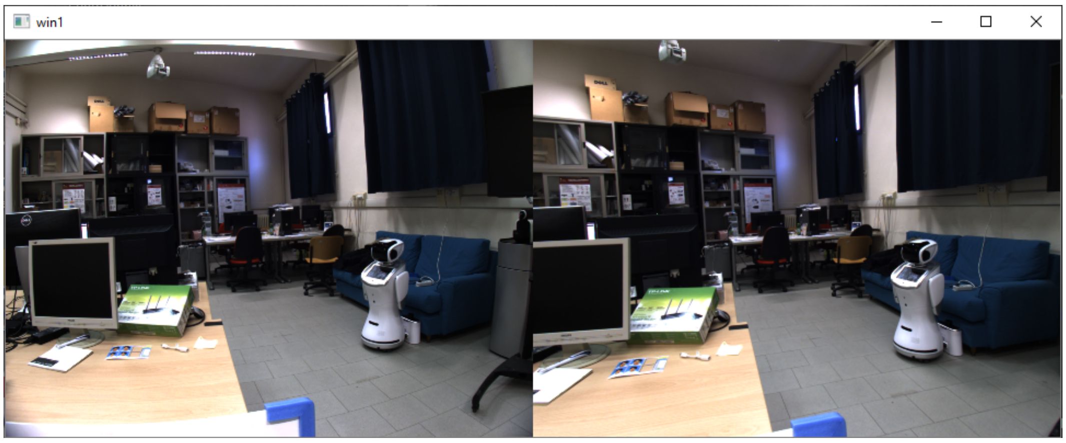
Camera calibration

Mautone Alberto

The code starts with the definition of 2 functions that are *computeReprojectionErrors(...)* and *unidstort(...)*. The first one returns a **double value**, since its aim is that to calculate the **reprojection error** without using the *calibrateCamera(...)* function. So basing on the theory it calculates the sum of every (squared) error thanks to the function *projectPoints(...)* that calculate every 3D point projection on a 2D plane and normalizes them with the norm(...) funcion which calculates an **absolute norm** of the array I give as parameters. Also I choose the norm type **L2** since it's the one required to calculate the error. Since we need the sum of the **square** of every error we just add squares to the total error, giving the line totalErr += err*err; the *undistort(...)* function instead, as suggested by the name, returns a Mat with the **undistorted image** (taking as parameters the image and its intrinsic and distortion coefficients matrixes) and it just makes use of 2 functions already offedered by OpenCV: *initUndistortRectifyMap(...)* which is based on the use of 2 new Mats called map x and map y. The function computes the joint undistortion and rectification transformation and represents the result in the form of maps for remap and that's why it's followed by the *remap(...)* function that finally gives me the **undistorted image.** Follows the *main()* function: first I upload all the pics given thanks to the *glob()* function putting them into an **array** called fn. I define some **variables** that I'll use later, as the vector with all the 3D points in the physical space (that is needed to be filled). How? Mathematically, I just create a vector of 3D point called **obj**: thanks to a for loop I create a list of coordinates (0, 0, 0), (0, 1, 0), (0, 2, 0), ... (1, 4, 0) and so on. Each corresponds to a particular **vertex**. And so, while detecting every **chesserboard corner** with the *findChessboardCorners(...)* function I fill 2 vectors: the one containing the corners detected (2D) and the one containing all the 3D points calculated before. I create some matrixes of the unkown as the one of the **intrinsic parameters** and only place 1 as value of the focal lenght among x and y axis. I'm finally ready to apply the *calibrateCamera(...)* function, since I have all the explicit parameters I need. After that I just calculate every **single error** on the input chesserboard images and thanks to this parameter I'm able to say which one **best performs** the calibration (the one with the lowest error) and which one performs the worst (the one with the highest error) by using some comparings and for loops. Finally I **undistort** the original image thanks to the method defined at the beginning: clearly the new image looks better, and I removed the distortion from the origina pic. The reault is clear in the split view obtained with the *hconcat(...)* function.

I have **printed** all the numerical result with the *cout «...«endl;* keyword: I also choose to print the reprojection error given by the *calibrationCamera(...)* function just to have a **compare value** to the error given by the function *computeReprojectionError(...)* written by me, finding out that they are the **same**.



On the left: original image.
On the right: undistorted image.