Computer Vision
Lab 2: camera calibration

Riccardo De Zen. 2019295

4 May 2021

1 Introduction

This report covers the second lab experience, which was centered around camera calibration. Starting from a set of checkerboard pictures taken with the same camera, we need to compute its intrinsic parameters and its distortion coefficients.

2 Implementation

My first issue involved computing the reprojection error. I did understand the formula in class already, but used the `norm` function poorly, forgetting the `NORM_L2` flag. After comparing my results with the ones returned from the camera calibration I confirmed the correctness of my computations.

When detecting the checkerboard corners on the images, I initially did not use the `cornerSubPix` function, leading to a higher error. When I did introduce said function, I first used a window size of 5×5, which barely improved the error no matter the ϵ and iteration limit. The error decreased substantially when I used a window size of 15×15.

I initially used the cmake `file(COPY ...)` directive to copy the dataset into the binaries directory, but later implemented some basic command line arguments to allow using other files with different checkerboard pattern sizes.

A few parameters needed to be tweaked:

- The size of the checkerboard squares in meters. While tecnically required in order to provide the list of 3d points for the checkerboard corners, its value does not influence the results, because the camera's intrinsics are measured in pixels. I left it as 0.11 since that was its real value.
- The `cornerSubPix` termination parameters. I empirically found a window size of 15×15, $\epsilon = 0.001$ and 30 maximum iterations to be a good combination.
- Scaling parameter for image rectification. Essentially, the `getOptimalNewCameraMatrix` function can return a camera matrix that preserves more or less of the original view. Since the region of interest is returned either way, I left the parameter at 1.

3 Results and discussion

With the provided dataset I reached a mean reprojection error of: 0.145015. Without using `cornerSubPix` the error was around 0.75, which was still quite good. As a matter of fact, the improvement in image rectification before and after this change is negligible to the naked eye. The intrinsics appeared to change very little regardless of the parameters:

$$\begin{bmatrix} a_u = 1245.1 & 0 & u_c = 981.34 \\ 0 & a_v = 1245.2 & v_c = 683.19 \\ 0 & 0 & 1 \end{bmatrix}$$

Distortion parameters were as follows:
$k_1 : -0.306929$
$k_2 : 0.135636$
$k_3 : -0.0344555$
$p_1 : -0.0000297316$
$p_2 : -0.000101691$

To define the concept of "best" and "worst" performing images, I just used reprojection error since it is the value minimized by the calibration algorithm. The image that performed best was `0057_color.png` with an error of 0.048815 and the one that performed worst was `0017_color.png` with an errof of 0.33389.

No particular issue arises in the results. The intrinsics share a similar magnitude, as they should according to the OpenCV documentation, and the distortion parameters are not particularly high. The main point that can be discussed is the impact of the `cornerSubPix` window size on the reprojection error. From what I could see, the improvements are due to the fact some images have corners that are slightly out of focus, making it harder to locate them with the base corner detector. The error decreases as the window size increases, up to a size of 15, but above such value the images with the checkerboard further away from the camera start to perform much worse.

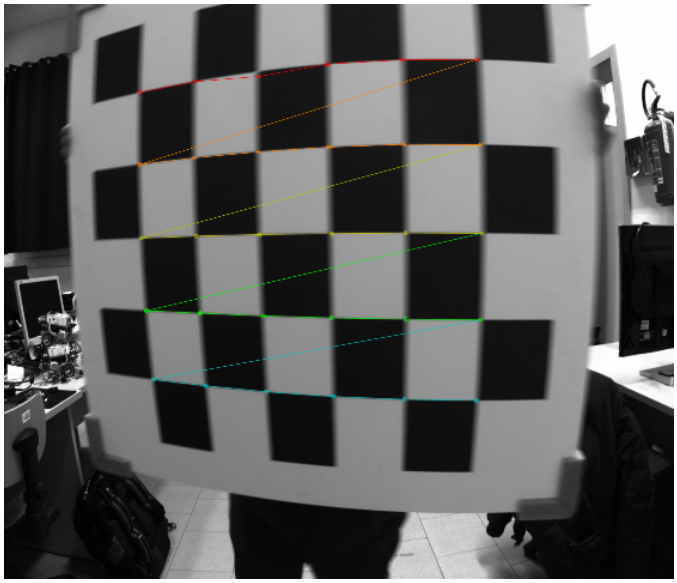
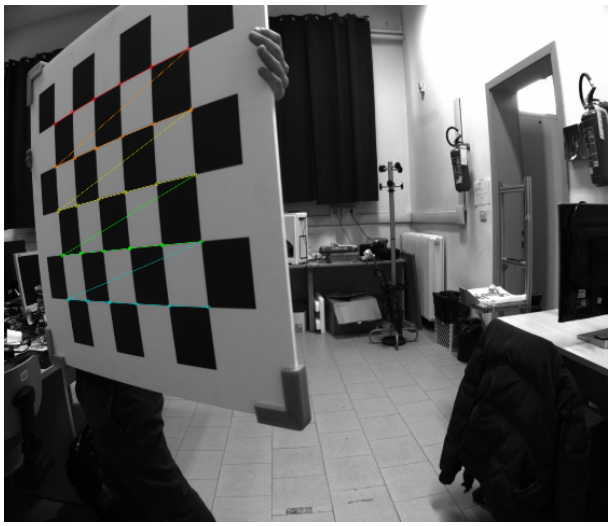
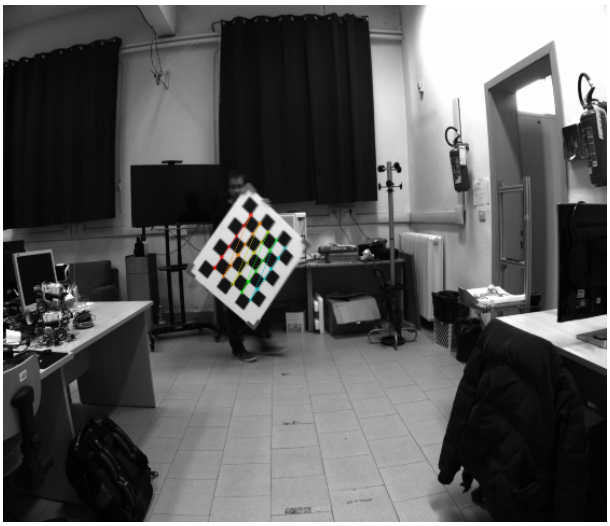


Figure 1: Worst performing image with no subpix precision (`0027_color.png`). Notice the third corner of the first row is quite poorly detected (image error > 3). With a big window size (e.g. 21×21) the corners are almost perfectly detected.



(a) Worst performing image with 15×15 window (image error $= 0.33389$).



(b) Worst performing image with 21×21 window (image error > 5).

Figure 3: Test image before and after rectification.