Computer Vision LAB 2 Report

Mattia Giacomello
I.D. 1210988

1 Goal

The goal is create a script that extrapolate the chessboard pattern in the images and computes the camera matrix and the distortion coefficients of the camera used. After the computation of these parameters they are used to remap an image, removing the distortions.

2 Procedure

The image set used is self taken and is used a chessboard pattern with 7 corners per row, 6 corners per column and squares with edge size of 3 cm. After loading the images, the program search for the chessboard inner corners in each image through the `findChessboardCorners()` OPENCV method, that gives a vector of 2D points in the image plane. To calibrate the camera is needed also a 3D reference of the chessboard in the real world, that is created fixing Z=0 and placing the corners on the XY plane with the real measures. From these 2 set of points, thanks to the `calibrateCamera()` method, we can obtain the camera matrix and the distortion coefficients. To reduce the reprojection error is possible to use the `cornerSubPix()` method that increase the accuracy of the corners position for each image through a 11x11 pixels search window. This value is chosen after some trial and gives a good result lowering the reprojection error. If the window is larger the error still lowers but it is a very small change against a large time required. The other parameters are the recommended ones since no relevant changes are observed modifying them.
The obtained results are reported in the table (1). Observing the root mean square error is possible to notice that this refinement is useful since there is a relevant change without and with, it is halved in this case. The image with the worst RMS reported in figure (1) is blured due to the focus and this makes difficult find the corners, but we can observe that thanks the refinement there is a big improvement. The camera matrices are similar, with a small change in the third column, the offset. What really changes is the distortion coefficients, where the third order coefficient of the radial distortion and the second radial coefficient are halved and the first tangent coefficient even changes sign.
From the computed instrinsic parameters it si possible to undistort and rectify the images throught the `initUndistortRectifyMap()` and `remap()` methods. In particular, the first one gives us 2 maps, one for x and one for y, that specify the mapping between the distorted and undistroted image's pixels, while the second one take these matrices and applies the effective mapping, giving us the final image. Since the image has a low distortion the effect of this remap is not so obvious at first sight. Looking at the edges of the picture is visible that there is a change since the pixels are stretched, thanks to the radial distortion correction. Moreover the straight lines in the original image are a bit bent while in the remapped image the desk's edge and the chessboard's lines result more straight. The toy cars also have more natural proportions after the processing while in the original image they seam a bit crocked. In conclusion, the `cornerSubPix()` function is useful to achieve an higher accuracy after a first, fast, search to improve the precision of the parameters. Regarding the remapping, even if the distortion coefficients are low, after the undistortion procedure the image has a more natural look.

	w/o `cornerSubPix()`	w `cornerSubPix()`
Camera Matrix $\begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 2223 & 0 & 1499 \\ 0 & 2222 & 970.9 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 2224 & 0 & 1496 \\ 0 & 2222 & 966.1 \\ 0 & 0 & 1 \end{bmatrix}$
Distortion Coefficients $\begin{bmatrix} k_1 \\ k_2 \\ k_3 \\ p_1 \\ p_2 \end{bmatrix}$	$\begin{bmatrix} -153.7 \times 10^{-3} \\ 210.2 \times 10^{-3} \\ -169.1 \times 10^{-3} \\ 215.2 \times 10^{-6} \\ 707.3 \times 10^{-6} \end{bmatrix}$	$\begin{bmatrix} -144.9 \times 10^{-3} \\ 155.8 \times 10^{-3} \\ -74.33 \times 10^{-3} \\ -256.5 \times 10^{-6} \\ 362.1 \times 10^{-6} \end{bmatrix}$
Euclidean Mean Error EMR	621.4×10^{-3}	347.3×10^{-3}
Root Mean Squared Error RMS	831.9×10^{-3}	444.4×10^{-3}
Best Image	EMR=266.6×10^{-3} RMS=305.6×10^{-3}	EMR=204.6×10^{-3} RMS=231.3×10^{-3}
Worst Image	EMR=1.081 RMS=1.456	EMR=715.9×10^{-3} RMS=901.3×10^{-3}

Table 1: Results of calibration procedure with and without the `cornerSubPix()` method.

Figure 1: The worst case in the self taken set

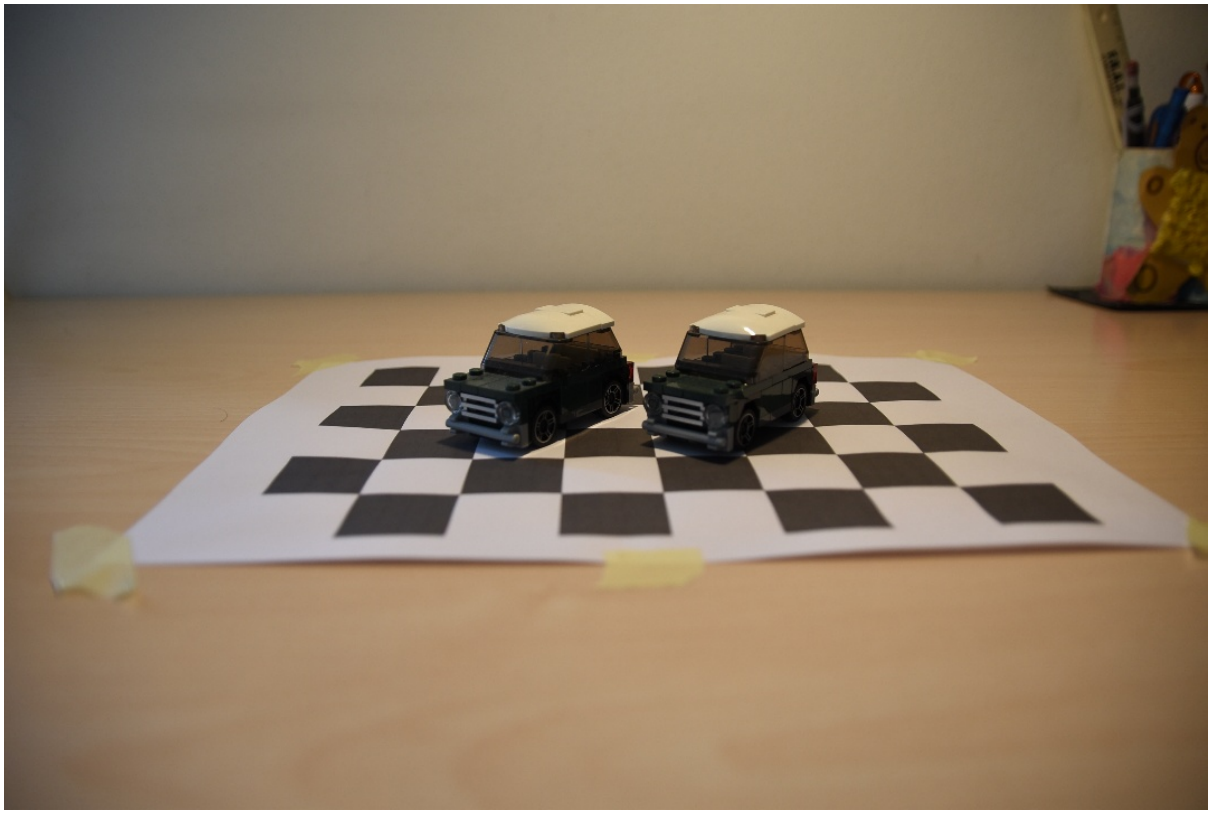

Figure 2: The original distorted image

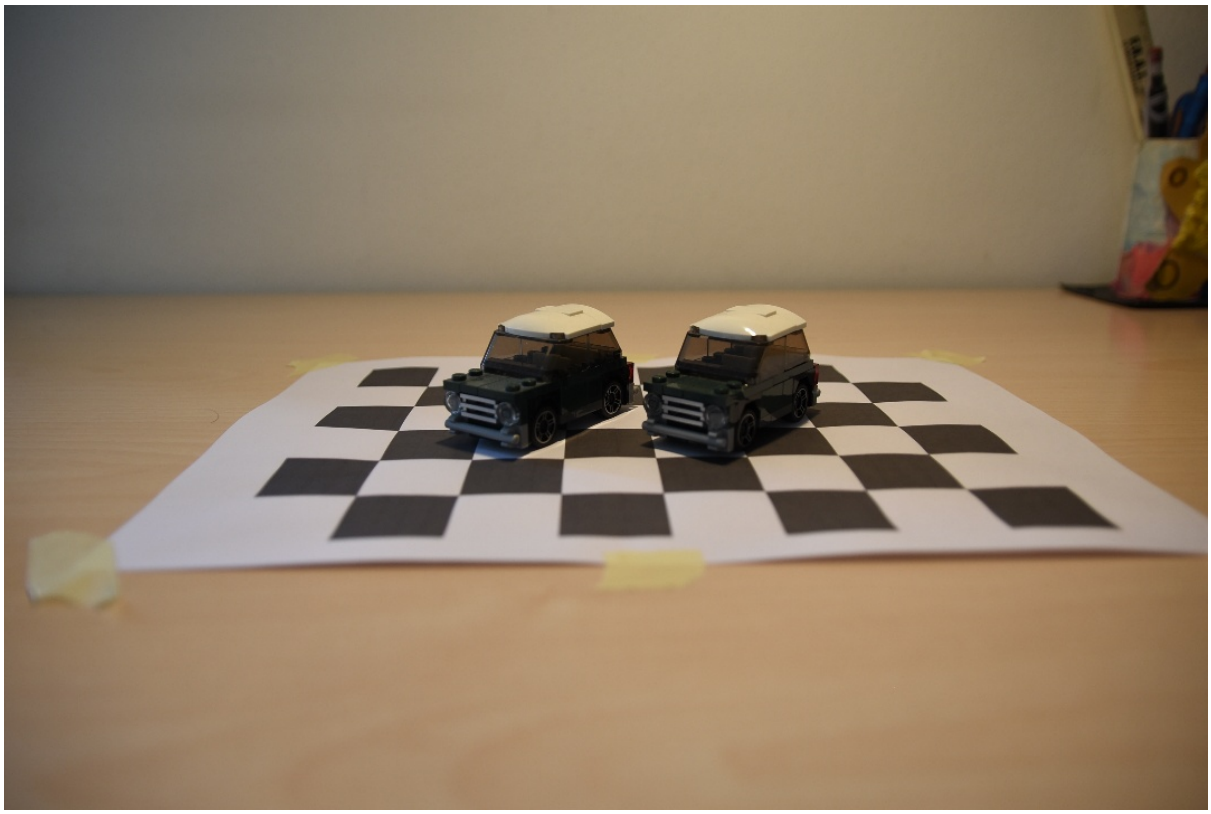

Figure 3: The undistorted image