COMPUTER VISION AND PATTERN RECOGNITION

Project Number 1

Alessandro Viol
Federica Azzalini

December 2021

Contents

1 Introduction

The goal of our work was to perform camera calibration on a set of given images of a checkerboard, as well as a set of pictures taken by us to calibrate our own device. The model is cameracentric, therefore the camera was fixed while the images of the checkerboard were taken as it was moving to different poses.

The calibration procedure consists in determining the geometric parameters of the image formation process, which is a crucial step in many computer vision applications. The model of the camera is defined by its intrinsic and extrinsic parameters. The first ones describe the internal characteristics of the camera, the second ones describe the position of the checkerboards with respect to the camera.

After calibrating, we performed compensation for the radial distortion that the camera lens applies to the image. To show the improvement, we computed the reprojection error before and after the compensation. Lastly, we superimposed a cylinder to the calibration object in the image.

2 Description of the approach

To calibrate the camera we applied Zhang method, which is based on homographies. The homographies are estimated from the correspondences between the points of the checkerboard and the points on the image. The checkerboard points' reference frame origins in the upper left corner of the calibration object. We build an over-determined linear equation system on the same unknown, where every couple of equations was generated from the correspondence between a point (u,v) on the image and the same point $m = [x,y,1]^T$ on the checkerboard. The unknowns h_1, h_2, h_3 are the columns of the matrix H that represents the homography

$$\begin{cases} m^T h_1 - u m^T h_3 = 0 \\ m^T h_2 - v m^T h_3 = 0 \end{cases}$$

The system was solved by single value decomposition.

At this point we could estimate the intrinsic parameters matrix from the relation $H = [p_1,\ p_2,\ p_4]$ between the homography matrix H and the perspective projection matrix $P = K[R|t]$, where K is the intrinsic parameters matrix, R is the rotation matrix and t is the translation vector. Exploiting the orthogonality of matrix R we can express

matrix K as a function of H, obtaining the two constraints

$$\begin{cases} h_1^T B h_2 = 0 \\ h_1^T B h_1 = h_2^T B h_2 \end{cases}$$

where $B = (KK^T)^{-1}$. By taking n planes and the corresponding estimated homographies we got $2n$ constraints on the same B, obtaining an over-determined linear system of equations we then solved using single value decomposition. We computed K through Cholesky factorization of B. In case B is negative definite instead of positive definite we apply Cholesky factorization to $-B$.

Having K we could extract the six intrinsic parameters of the camera. The first ones are the coordinates of the center of projection projected on the sensor, u_0 and v_0. We get them respectively from K as $K_{1,3}$ and $K_{2,3}$. The third one is the skew angle of the sensor $\delta = \arctan(\frac{K_{1,1}}{K_{1,2}})$. The last ones are the reciprocal of the height and width of the pixels multiplied by the focal length $\alpha_u = K_{1,1} = \frac{f}{s_u}$, $\alpha_v = \frac{K2,2}{sin(\delta)} = \frac{f}{s_v}$.

$$K = \begin{bmatrix} \alpha_u & \alpha_u \cot\theta & u_0 \\ 0 & \frac{\alpha_v}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} fk_u & fk_u \cot\theta & u_0 \\ 0 & \frac{fk_v}{\sin\theta} & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Once we found the intrisic parameters we could estimate the extrinsics ones. To compute the extrinsic parameters we applied the following formulas

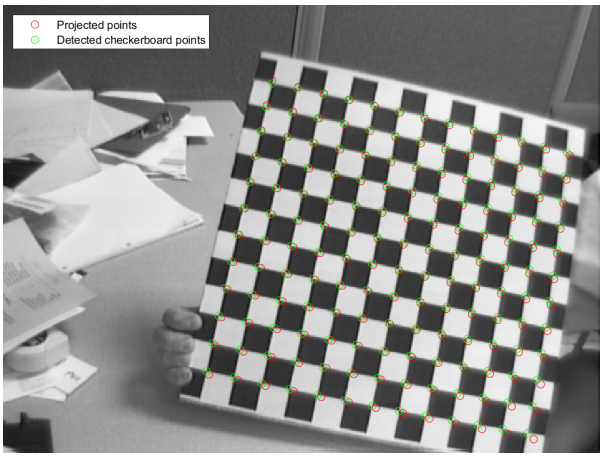
$$r_1 = \lambda K^{-1} h_1, \quad r_2 = \lambda K^{-1} h_2, \quad t = \lambda K^{-1} h_3, \quad r_3 = r_1 \times r_2$$

where $\lambda = \frac{1}{||K^{-1}h_1||} = \frac{1}{||K^{-1}h_2||}$. We then found the closest orthogonal matrix that approximates the matrix R by using single value decomposition.

We then evaluated the quality of our work so far by computing the total reprojection error for all the N images and then summing the N results

$$\sum_{j=1}^{n} \left(\left(\frac{P_1^T m_j}{P_3^T m_j} - u_j \right)^2 + \left(\frac{P_2^T m_j}{P_3^T m_j} - v_j \right)^2 \right)$$

where n is the number of points in the image, m_j are the coordinates of the points of the checkerboard and (u_j, v_j) are the coordinates of the corresponding points in the image. The following image is an example of the resulting projection. For further examples, we also added the projected points in the examples in the section Verification and results.



Now that we've calibrated the camera, we could further improve the reprojection error by introducing radial distortion compensation. To do so, we need to estimate the parameters k_1, k_2 which describe the radial distortion.

$$\begin{cases} \hat{u} - u = (u - u_0) {r_d}^2 k_1 + (u - u_0) r_d^4 k_2 \\ \hat{v} - v = (v - v_0) {r_d}^2 k_1 + (v - v_0) r_d^4 k_2 \end{cases}$$

where $(\hat{u}, \hat{v})$ are the distorted point pixel coordinates, (u, v) are the undistorted pixel coordinates and $r_d^2 = \left(\frac{u-u_0}{\alpha_u}\right)^2 + \left(\frac{v-v_0}{\alpha_v}\right)^2$. From every point of every image, we can get one pair of equations on the same k_1, k_2, thus obtaining an overdetermined non-homogeneous linear system of equations. We solved the system $A \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = b$, by applying the formula $\begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = (A^T A)^{-1} A^T b$.

By normalizing the coordinates with the following expressions

$$x = \frac{u - u_0}{\alpha_u}, \quad y = \frac{v - v_0}{\alpha_v}$$

and rearranging the radial distortion equations we get the non-linear system

$$\begin{cases} \hat{x} = x(1 + k_1(x^2 + y^2) + k_2(x^2 + y^2)^2) \\ \hat{y} = y(1 + k_1(x^2 + y^2) + k_2(x^2 + y^2)^2) \end{cases}$$

By inverting the equations of the system, we could obtain the undistorted pixel coordinates from the distorted points coordinates. Unfortunately, the solution to this problem doesn't come in closed form. So, we now considered the two equations as a vector value function and applied the iterative Newton method to approximate the root of the

function, which are the coordinates of the undistorted point.

$$x = \hat{x} - J(x)^{-1} f(x)$$

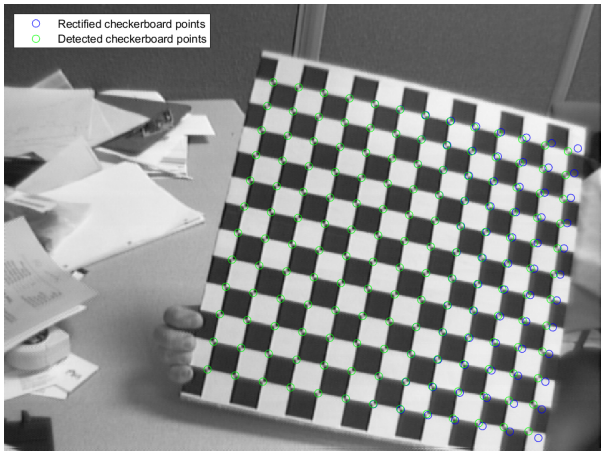
where J is the Jacobian matrix $[J_{ij}] = \frac{\delta f_i(x)}{x_i}$ and

$$f(x,y) = \begin{cases} x(1+k_1(x^2+y^2)+k_2(x^2+y^2)^2) - \hat{x} \\ y(1+k_1(x^2+y^2)+k_2(x^2+y^2)^2) - \hat{y} \end{cases}$$

At this point all we had to do is de-normalize the coordinates as follows

$$u = x\alpha_u + u_0, \quad v = y\alpha_v + v_0$$

The following image shows circled in blue the rectified points. For further examples, we also added the rectified points in the examples in the section Verification and results.



We then calculated the reprojection error without compensation and we compared it with the previous value. After that, we iterated the entire procedure to improve the results.

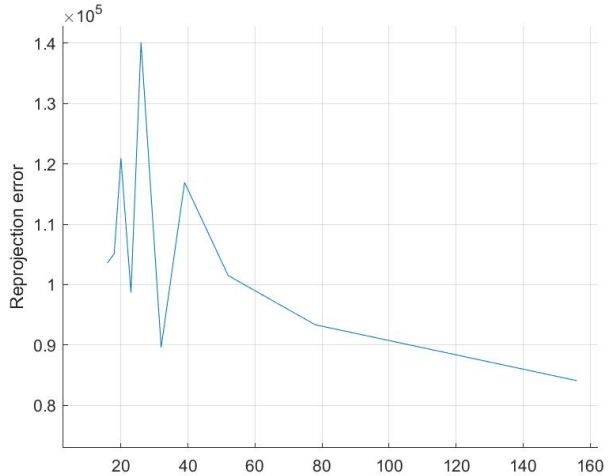
The last procedure we performed is superimposing a cylinder to the calibration object. By verifying that the cylinder is superimposed exactly in the right position, we further prove that the matrix P and the parameters k_1 and k_2 previously found are correct. To do that we considered the points of a cylinder with respect to the checkerboard reference frame, we multiplied them with the orthogonal matrix R and we summed the result with t. Then we added radial distortion to the projected points and plotted the result.

3 Implementation choices

The entire code was written in matlab. We were provided a set of 20 images of a checkerboard and to this set we added 7 checkerboard images taken from our smartphone.

To get the pixel coordinates of the checkerboard's corners we used the method 'detectCheckerboardPoints' of the matlab 'Computer Vision toolbox' libraries. From the given image set we eliminated image 9 because the method wasn't able to detect all points of the checkerboard, setting their coordinates values to 'Not a number'. Moreover, for some images the method 'detectCheckerboardPoints' detected more rows or columns than they actually are. When this happened we thought it was enough to eliminate the ones in excess to obtain a correct calibration, but in these cases the method also misplaces the origin of the checkerboard. To solve this issue we also eliminated all the images with this problem. In the given dataset the only one was image 2.

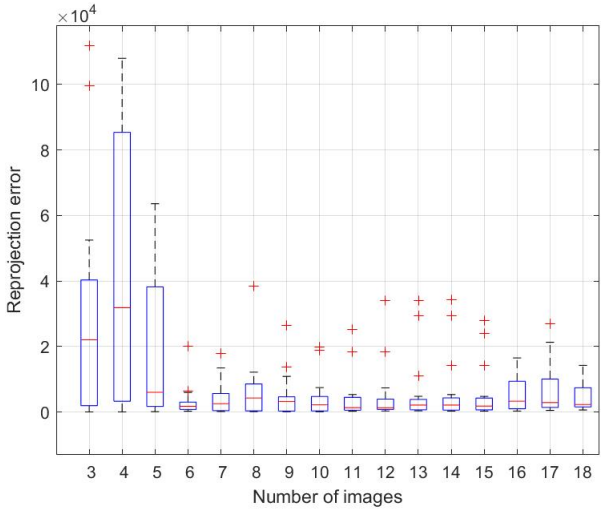
To estimate the homographies we used every detected corner of the checkerboard's squares, since the graph of the reprojection error suggested that the error decreases with the increasing number of points. Furthermore, for a small number of points in the homography estimation the result is numerically inconsistent. Some of these homographies, if used to estimate K, generated B matrices that are neither positive nor negative definite. That prevents us to use Cholesky factorization to obtain the intrinsic parameters matrix K. The following image shows the reprojection error as a function of the number of points considered in the homography estimation.



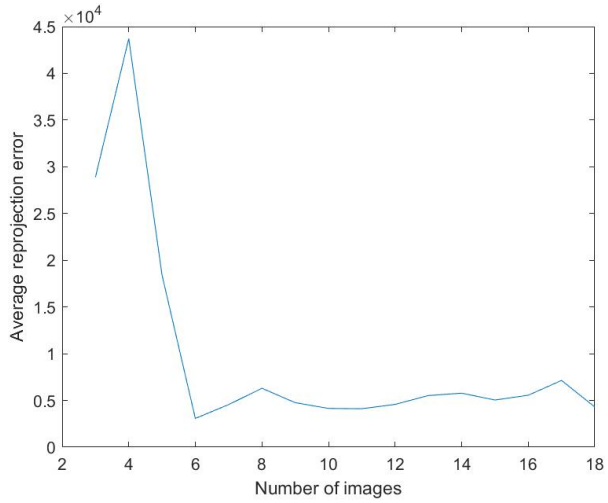
Reprojection error obtained estimating the homographies with an increasing number of points

To determine K, we decided to test the optimal number of images to consider. We

performed a procedure similar to leave-one-out cross validation. We iterated the test for as many times as the images we have, each time leaving out one image. For every cycle we calculated the reprojection error on the left out image obtained estimating K for every number of images between 3, which is the minimum necessary, and all images, excluded the one we left out. From the graph shown below is clear that the option that minimizes the error is taking 6 images, yet we think this introduces the problem of outliers. We noticed that if we obtain the K that produces the outlier, the compensation of the radial distortion does not work properly, probably because it strongly depends on K. If we want a more stable result we can take all 18 images, even if it doesn't minimize the error.

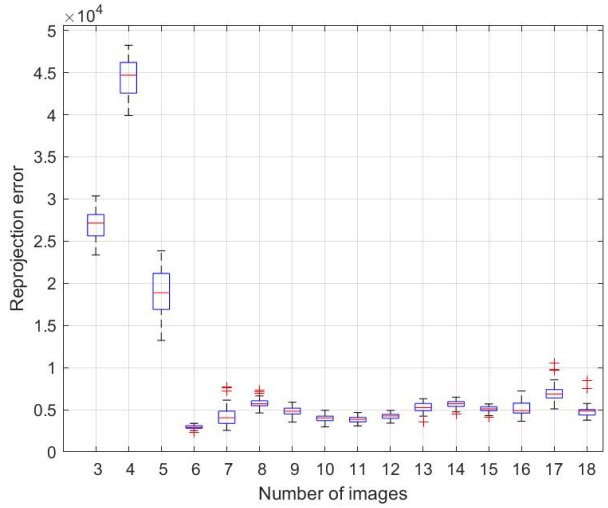


Distribution of the reprojection errors obtained with leave-one-out cross validation



Average of the errors for each group of images

To make sure that the order of the images in the test doesn't count, we choose to iterate the whole procedure 50 times, randomly shuffling the indexes of the images before every cycle. The plot below shows the distribution of the 50 averages obtained. This shows that the order of the images doesn't influence the results, since the results obtained each iteration are quite consistent with respect to each other, as the distribution of the averages of the 50 iterations is not too spread apart.

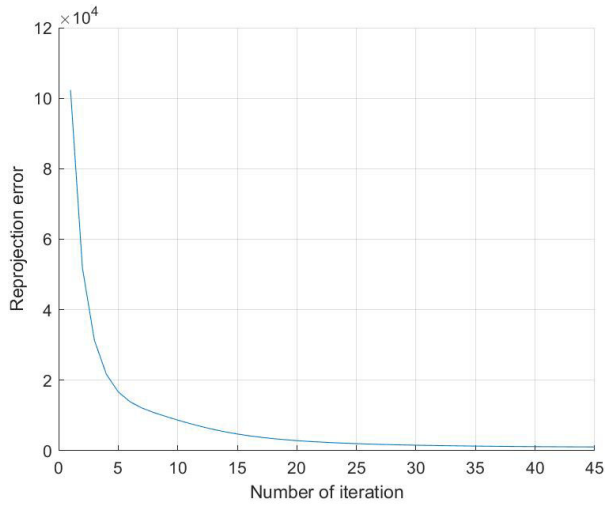


Iteration for 50 times of figure 2 procedure,
by randomly selecting the order of the images

To calculate the undistorted coordinates, as mentioned above, we iterated Newton's method, this way we can retrieve the undistorted pixel coordinates of the checkerboard

corners. We chose to keep iterating the method until the difference between the found points and the distorted points detected by the 'detectCheckerboardPoints' measures if $\frac{1}{156}$ pixel in both directions, or, in the worst cases, we stopped it at 100 iterations. To be able to compare the points in the stopping criterion, we needed to apply radial distortion to the found points with the current k_1 and k_2 parameters.

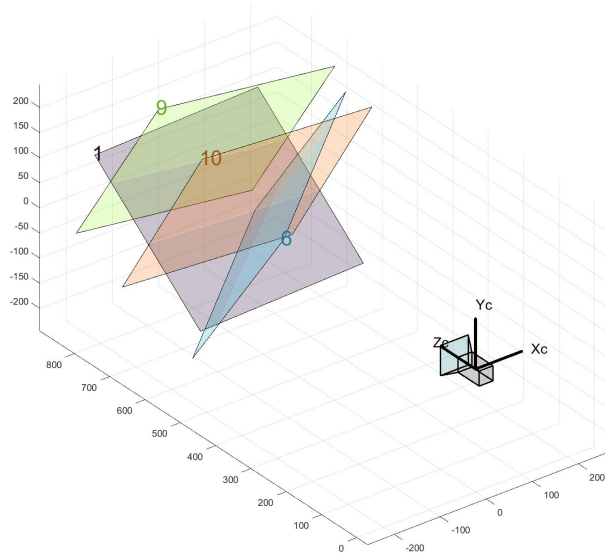
Regarding the iterative compensation of the radial distortion, we decided to stop at 45 loops, because the reprojection error reached an asymptote, so for the next iterations the error decreased negligibly.



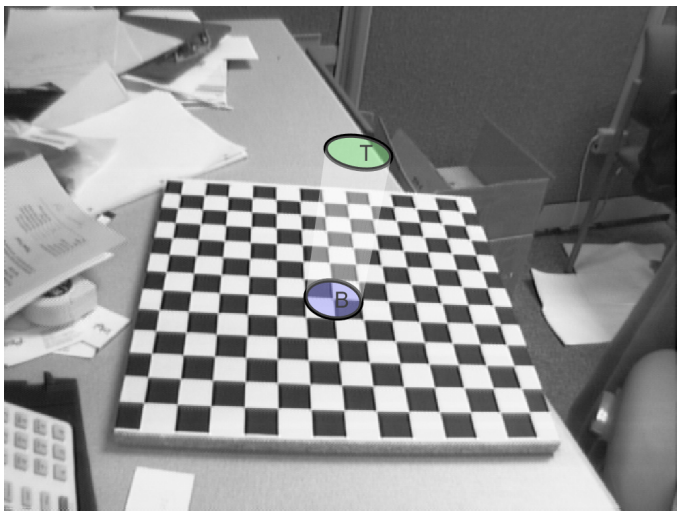
Reprojection error at each step of the compensation iteration

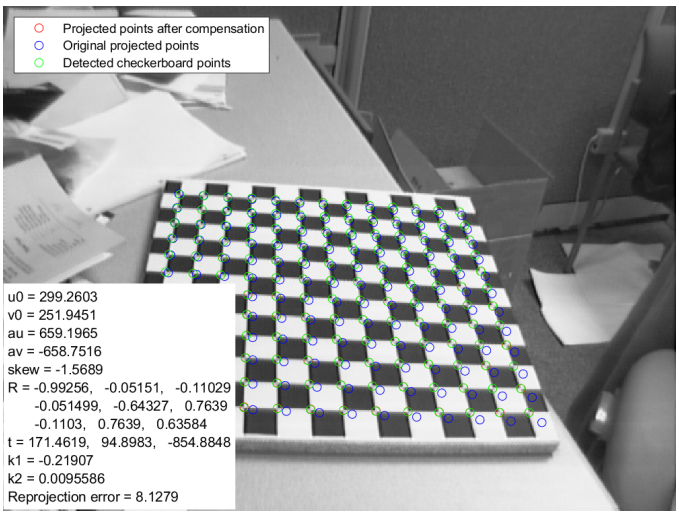
4 Verification and results

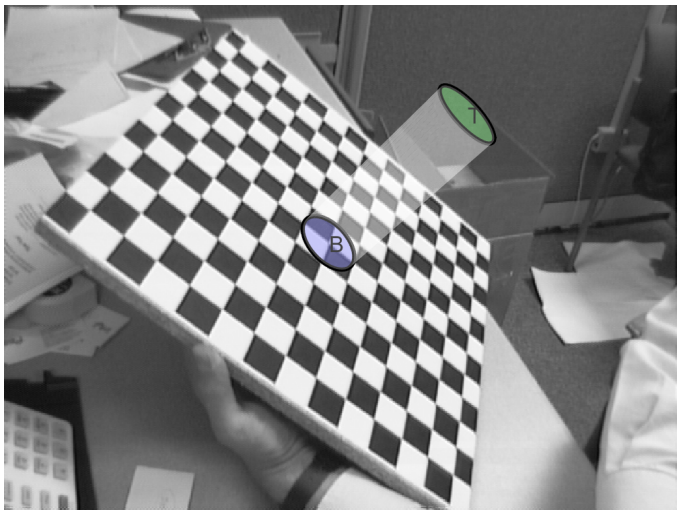
To visualize the extrinsic parameters of the camera, we built a 3D visual representation showing the relative position and rotation of the checkerboards with respect to the camera.

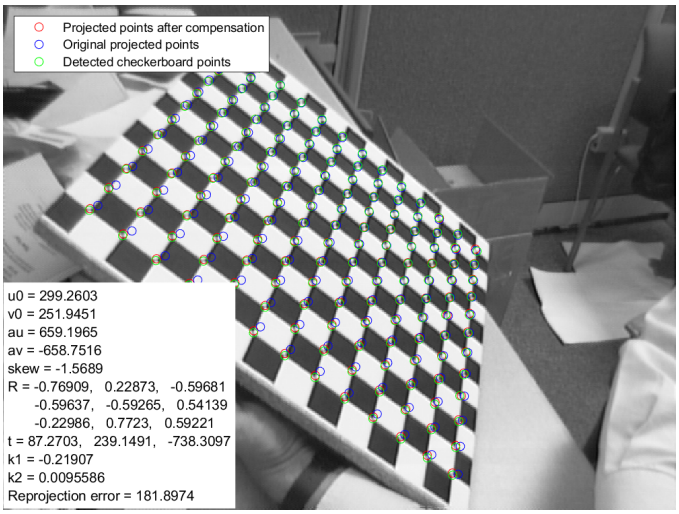


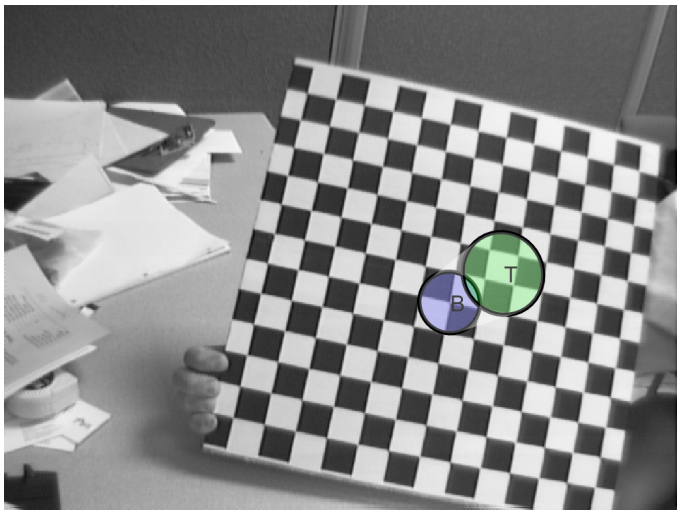
Furthermore, to verify that the calibration was correct, we superimposed a cylinder to a point in the checkerboard. We can see that the camera is correctly calibrated because the cylinder appears as expected and on the correct spot we chose, in the center of the image. We now show some examples. For an image we show two plots, where we display the cylinder in the first one and the detected, projected and rectified points in the second one.

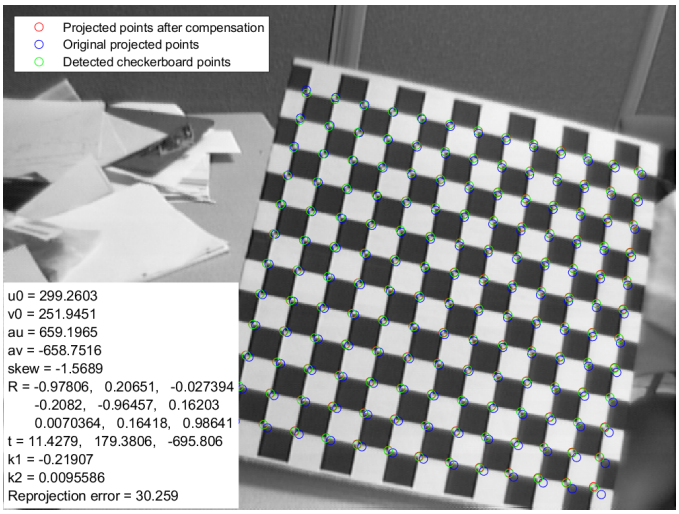


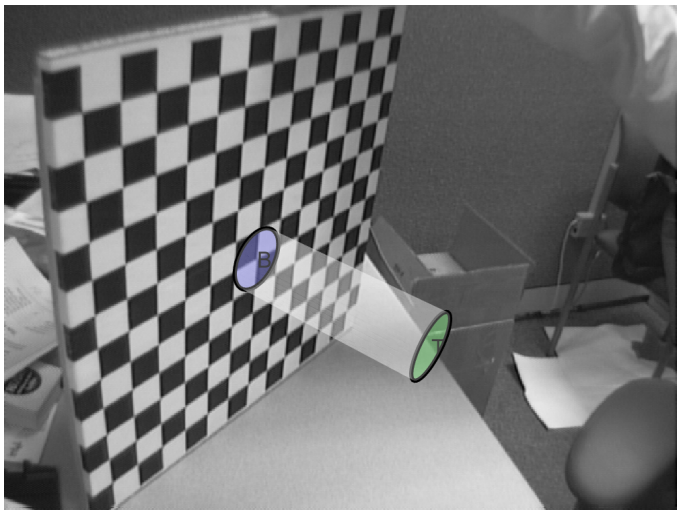


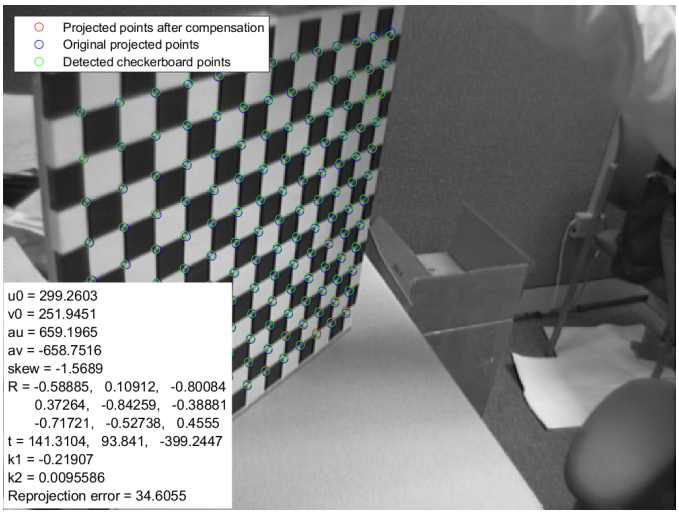

T
B


Projected points after compensation
Original projected points
Detected checkerboard points
u0 = 299.2603
v0 = 251.9451
au = 659.1965
av = -658.7516
skew = -1.5689
R = -0.76909, 0.22873, -0.59681
-0.59637, -0.59265, 0.54139
-0.22986, 0.7723, 0.59221
t = 87.2703, 239.1491, -738.3097
k1 = -0.21907
k2 = 0.0095586
Reprojection error = 181.8974

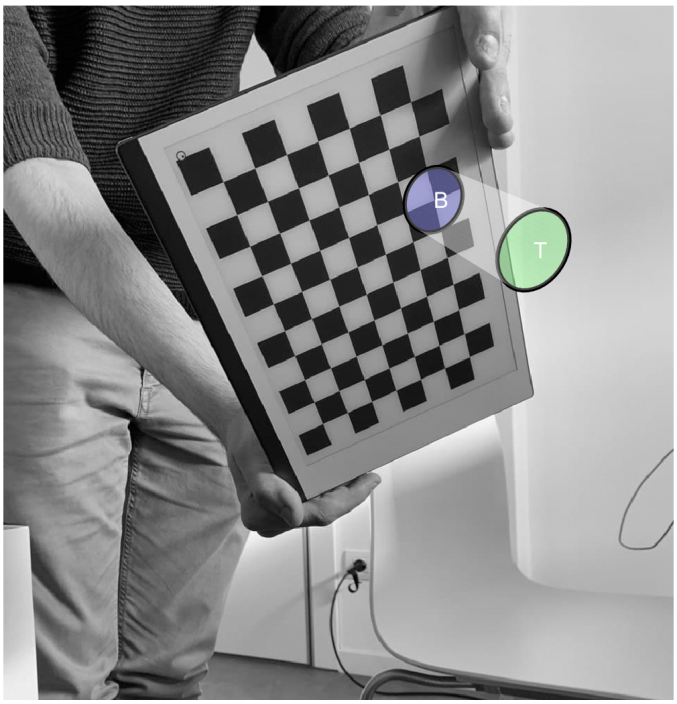

T
B

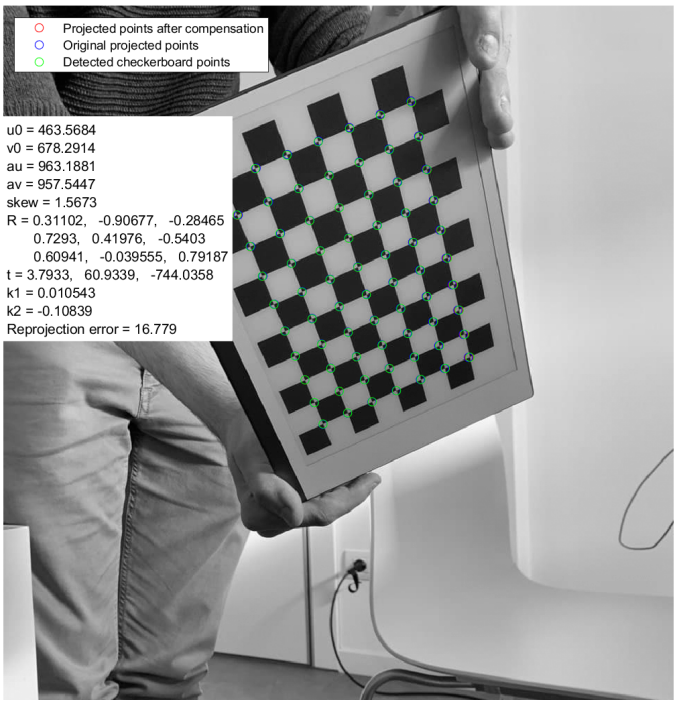

Projected points after compensation
Original projected points
Detected checkerboard points
u0 = 299.2603
v0 = 251.9451
au = 659.1965
av = -658.7516
skew = -1.5689
R = -0.97806, 0.20651, -0.027394
-0.2082, -0.96457, 0.16203
0.0070364, 0.16418, 0.98641
t = 11.4279, 179.3806, -695.806
k1 = -0.21907
k2 = 0.0095586
Reprojection error = 30.259





The last examples show the same results on our own images for our device calibration.





References

[1] Burger Wilhelm: *Zhang's Camera Calibration Algorithm: In-Depth Tutorialand Implementation*, Technical Report HGB16-05, University of Applied Sciences Upper Austria, School of Informatics, Communications and Media, Dept. of DigitalMedia, Hagenberg, Austria, May 2016. `https://www.researchgate.net/publication/303233579_Zhang\OT1\textquoterights_Camera_Calibration_Algorithm_In-Depth_Tutorial_and_Implementation`