Computer Vision
Homework Assignment 1 : Calibration

Autumn 2018

Nicolas Marchal

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision
and Geometry Lab

1.1 Data Normalization

I first run the script `exercise1.m` to select the points that I will use for calibration (see figure 1). These will give me the classic coordinate. In the beginning of my `runDLT.m` I add an extra coordinate (=1) to get homogeneous coordinate.

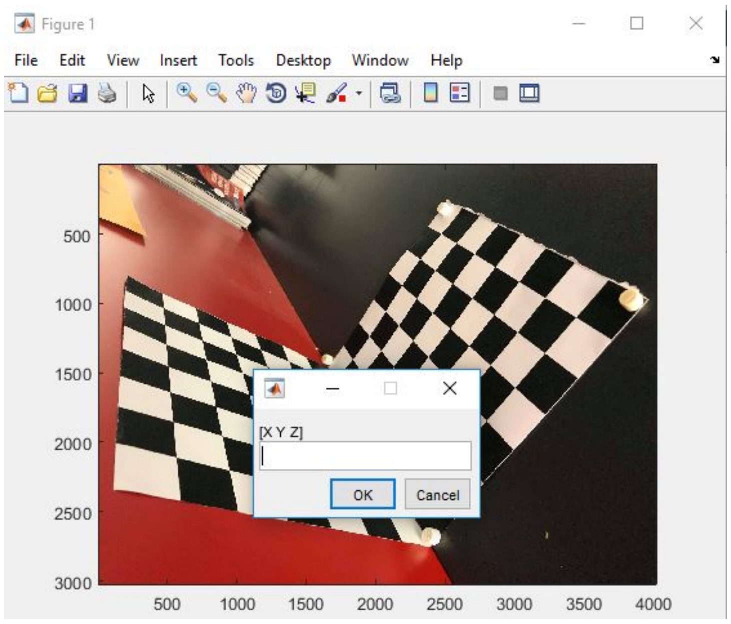


Figure 1: exercise1.m

I then compute the transformation matrices T and U such that the new vectors $\hat{\boldsymbol{x}}_i = T\boldsymbol{x}_i$ and $\hat{\boldsymbol{X}}_i = U\boldsymbol{X}_i$ have their centroid at the origin and that their average distance to the origin is $\sqrt{2}$ and $\sqrt{3}$ respectively. This is done in the function `normalization` and works as follows:

Algorithm 1: normalization

```
1 Compute the centroids of the points ;
2 Subtract the centroids to the points such that the new centroids is at the origin ;
3 Scale the vectors such that their norm (without the added homogeneous
    coordinate) is $\sqrt{2}$ for image points and $\sqrt{3}$ for object points ;
```

This gives in matrix form :

$$T = \begin{bmatrix} \lambda & 0 & -\lambda\bar{x} \\ 0 & \lambda & -\lambda\bar{y} \\ 0 & 0 & 1 \end{bmatrix} \quad (1) \qquad U = \begin{bmatrix} \lambda & 0 & 0 & -\lambda\bar{x} \\ 0 & \lambda & 0 & -\lambda\bar{y} \\ 0 & 0 & \lambda & -\lambda\bar{z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

1.2 Direct Linear Transform (DLT)

• We now perform the DLT algorithm in the function `dlt.m`. First we have to compute the normalized camera matrix $\hat{P}$ with the following equation :

$$\underbrace{\begin{bmatrix} w_i\hat{\boldsymbol{X}}_i^\top & \boldsymbol{0}^\top & -x_i\hat{\boldsymbol{X}}_i^\top \\ \boldsymbol{0} & -w_i\hat{\boldsymbol{X}}_i^\top & y_i\hat{\boldsymbol{X}}_i^\top \end{bmatrix}}_{A_{12x12}} \begin{pmatrix} \hat{\boldsymbol{P}}^1 \\ \hat{\boldsymbol{P}}^2 \\ \hat{\boldsymbol{P}}^3 \end{pmatrix} = \boldsymbol{0} \tag{3}$$

we can easily find the solution for $\boldsymbol{P}$ as is the right null vector of A. This can be found using the singular value decomposition that is available in matlab (`svd(A)`). We take the last column (eigenvector corresponding to the smallest eigenvalue) of the matrix V. This method is guaranteed to work because we have normalised our vector earlier. Once this is done, we put the vector back into a matrix form and we get the normalized camera matrix $\hat{P}$.

• We can now denormalize $\hat{P}$ to get the camera matrix P as follows :

$$P = T^{-1}\hat{P}U \tag{4}$$

• I then completed the function `decompose(P)` to get the intrinsic camera matrix K, the rotation matrix R and the camera center $\boldsymbol{C}$.

$$P = [M| - M\boldsymbol{C}] = K[R| - R\boldsymbol{C}] \tag{5}$$

To find K and R we use the fact that since $M = KR$ then $M^{-1} = R^{-1}K^{-1}$. We can do a QR decomposition of M^{-1} such that $M^{-1} = Q_{qr}R_{qr}$ where R is upper triangular (like the K matrix we are looking for).

$$M^{-1} = R^{-1}K^{-1} = Q_{qr}R_{qr} \quad \Rightarrow \quad \begin{cases} K = {R_{qr}}^{-1} \cdot \lambda_k \\ R = {Q_{qr}}^{-1} \cdot \frac{1}{\lambda_k} \end{cases} \tag{6}$$

The multiplier λ_k is necessary to assure that $K(3,3) = 1$.

Finally, the the camera center $\boldsymbol{C}$ can be found because it is the point for which $P \cdot \boldsymbol{C} = 0$. It is therefore the right null-vector of P and it is found using `svd(P)`.

• I added a visualisation function `visualizePoints(P,xy)` that allows to visualize where my matric predicts that every corner will B. The pseudocode is given below and the results is shown in figure 2.

```
Algorithm 2: visualizePoints
1 for all corners on the grid do
2     calculate their projection with $\hat{\boldsymbol{x}}_i = P\boldsymbol{X}_i$ (scale such that homogeneous
      cordinate = 1) ;
3     Show the point on the image
4 circle the 6 points used for calibration.
```

The results are summarised below in figure 2 and table 1 :

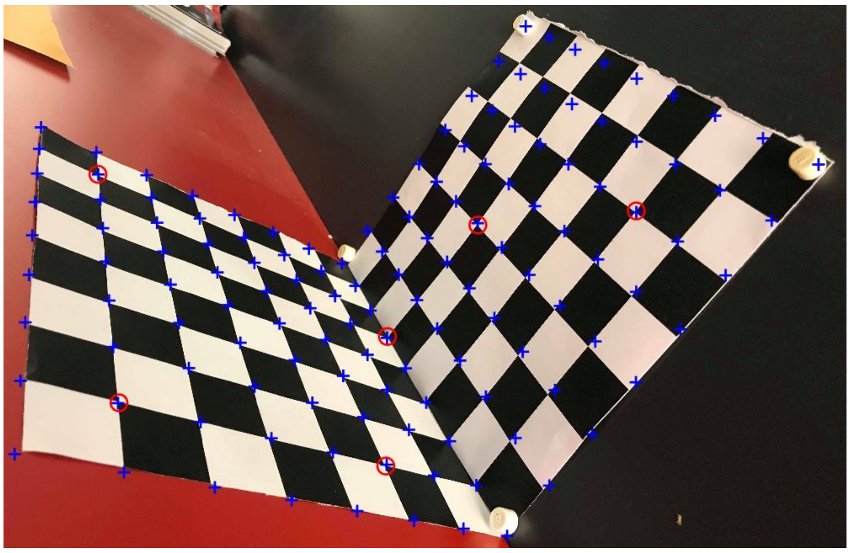


Figure 2: DLT results

K	R	**C**	**t**
$\begin{pmatrix} 3318 & -109 & 2216 \\ 0 & 3264 & 1072 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1.2331 & 1.1 & -0.8532 \\ -0.7832 & -1.4904 & -0.7897 \\ -1.1509 & -0.1644 & 1.4515 \end{pmatrix}$	$\begin{pmatrix} 0.2490 \\ 0.1276 \\ -0.1480 \\ 1.0000 \end{pmatrix}$	$\begin{pmatrix} 0.0405 \\ 0.2683 \\ 0.5224 \end{pmatrix}$

P	Error
$\begin{pmatrix} -6556 & 3448 & 472 & 1263 \\ -3789 & -5040 & -1022 & 1435 \\ -1.2 & -0.2 & -1.5 & 0.5 \end{pmatrix}$	5.915

Table 1: DLT Algorithm

• **Note** : We must use normalized vectors to guarantee that we will to find a good matrix P using `svd` (as explained previously). In my case I still got good results (with similar error) using unnormalized vectors but depending on the points it could be problematic.

1.3 Gold Standard Algorithm

• We normalise the inputs and run DLT algorithm as explained previously to get $\tilde{P}$.

• Using $\tilde{P}$, the results are shown in figure 3a (circled points were manually entered).

• We perform an optimization step to tune the estimated matrix $\tilde{P}$ to minimize the error. We define the cost function as $\sum_i d(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)^2$ in the function `fminGoldStandard`. We then minimise this cost function using the built-in Matlab function `fminsearch`.

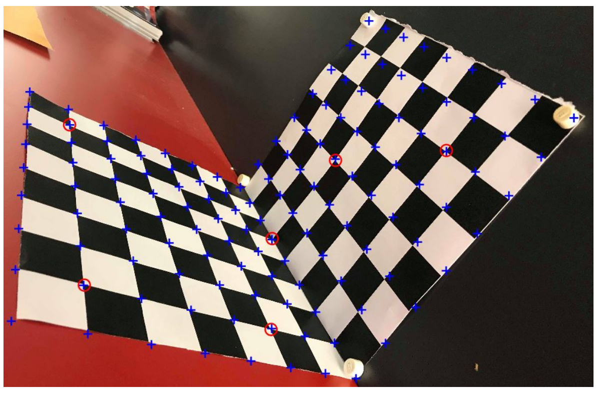
• We then Denormalized the camera matrix as in task 1.2.

• The results of this algorithm is given below. Table 2 shows the resulting parameters. Figure 3b shows the small improvement that we achieved.

K	R	**C**	**t**
$\begin{pmatrix} 3352 & -107 & 2206 \\ 0 & 3303 & 1057 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1.2286 & 1.0878 & -0.8383 \\ -0.7775 & -1.4781 & -0.7785 \\ -1.1320 & -0.1654 & 1.4445 \end{pmatrix}$	$\begin{pmatrix} 0.2504 \\ 0.1288 \\ -0.1504 \\ 1.0000 \end{pmatrix}$	$\begin{pmatrix} 0.0414 \\ 0.2680 \\ 0.5220 \end{pmatrix}$

$\tilde{P}$	$P_{optimized}$	Error
$\begin{pmatrix} -6556 & 3448 & 472 & 1263 \\ -3789 & -5040 & -1022 & 1435 \\ -1.2 & -0.2 & 1.5 & 0.5 \end{pmatrix}$	$\begin{pmatrix} -6532 & 3439 & 459 & 1262 \\ -3765 & -5056 & -1044 & 1437 \\ -1.1 & -0.2 & 1.4 & 0.5 \end{pmatrix}$	5.1083

Table 2: Gold Standard with 6 points


(a) DLT

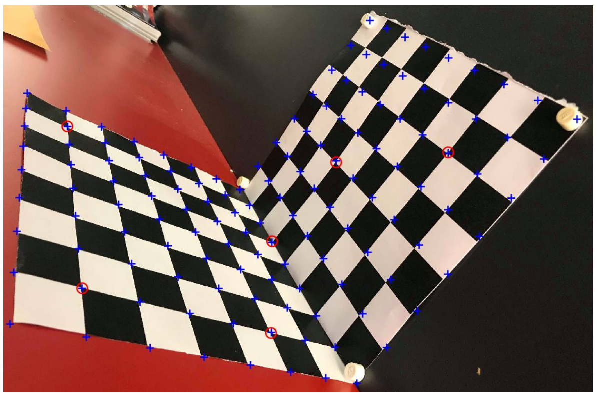

(b) Gold Standard 6 points

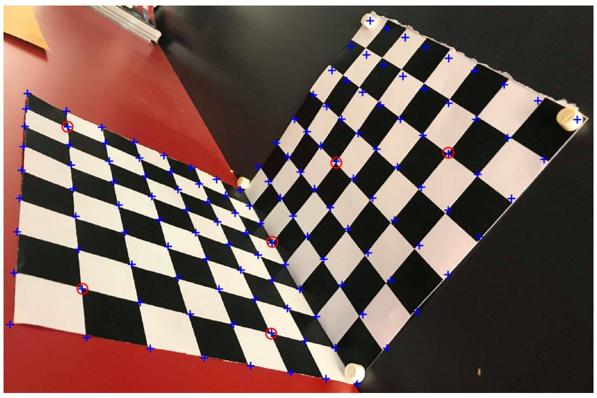
Figure 3: Result comparison

• Although the Gold standard algorithm has improved our results, we can further improve it by taking more and "better" points (points distributed uniformly on the pattern). With a new set of 7 points we have really increased the results and have obtained a very satisfying calibration. The results are shown in table 3 and figure 4b.

K	R	**C**	**t**
$\begin{pmatrix} 3357 & 15.8 & 2017 \\ 0 & 3386 & 1446 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1.3383 & 1.0713 & -0.7148 \\ -0.6528 & -1.4529 & -0.9553 \\ -1.1102 & -0.4371 & 1.4234 \end{pmatrix}$	$\begin{pmatrix} 0.2385 \\ 0.1374 \\ -0.1524 \\ 1.0000 \end{pmatrix}$	$\begin{pmatrix} 0.0630 \\ 0.2098 \\ 0.5419 \end{pmatrix}$

P	$P_{optimized}$	Error
$\begin{pmatrix} -6556 & 3448 & 472 & 1263 \\ -3789 & -5040 & -1022 & 1435 \\ -1.2 & -0.2 & 1.5 & 0.5 \end{pmatrix}$	$\begin{pmatrix} -6742 & 2692 & 456 & 1307 \\ -3815 & -5552 & -1177 & 1494 \\ -1.1 & -0.4 & 1.4 & 0.5 \end{pmatrix}$	1.8507

Table 3: Gold Standard with 7 better points


(a) Gold Standard 6 points

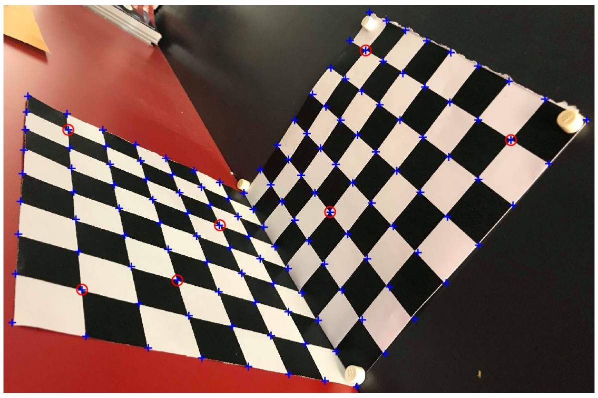

(b) Gold Standard 7 better points

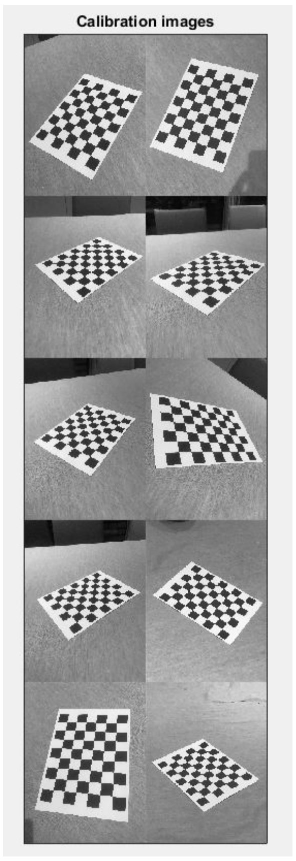
Figure 4: Result comparison with 7 points

1.4. Bouget Calibration Toolbox

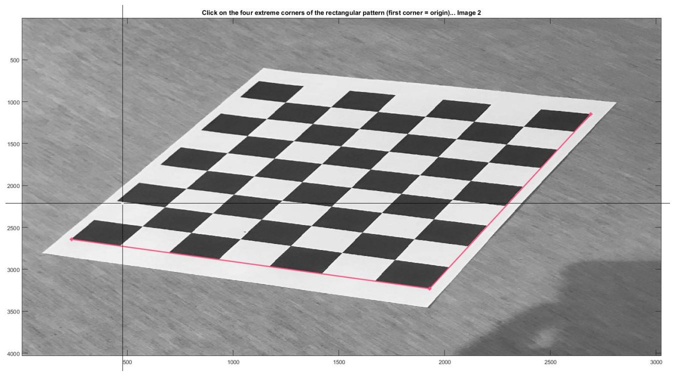
There exists many programs that use a similar approach as we have done before to do camera calibration. However, these toolboxes have been very well designed and allow for example to automatically detect the corners in our pattern as seen in figure 5b and 5c.

Once we have identified all the corners, we can run the calibration. As you can see on 6 the results are okay but they are not better than with the gold Standard algorithm.

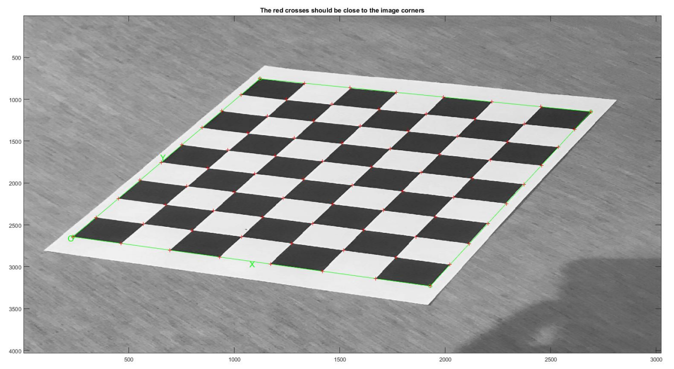
From the toolbox, we can extract the camera intrinsic parameters as well as the error (see table 4). The parameters are in the same range as previously. They can vary quite



(a) Images used for calibration


Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 2

(b) Pattern Borders



(c) Corner detection

Figure 5: Calibration Steps

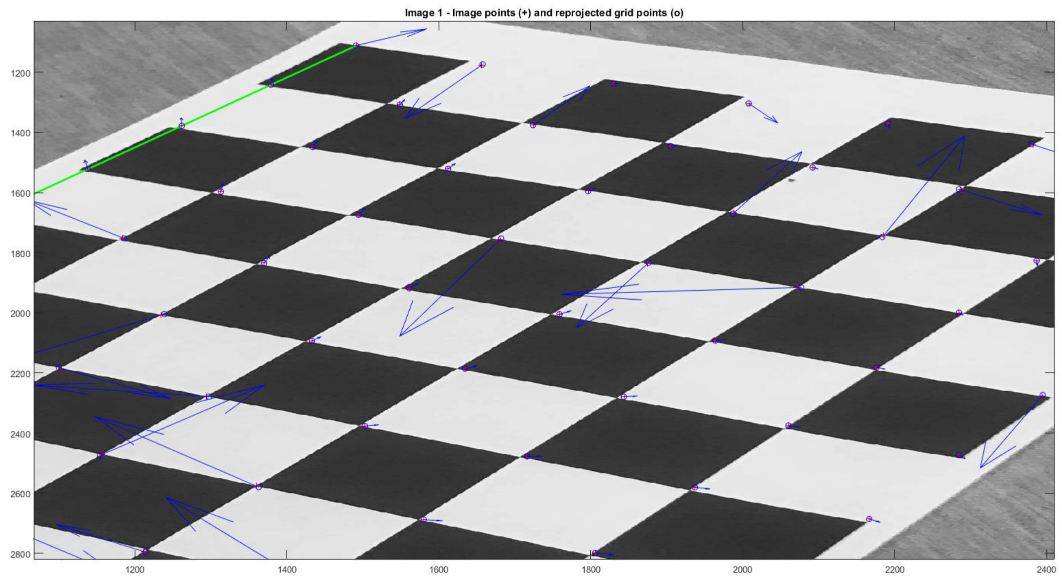
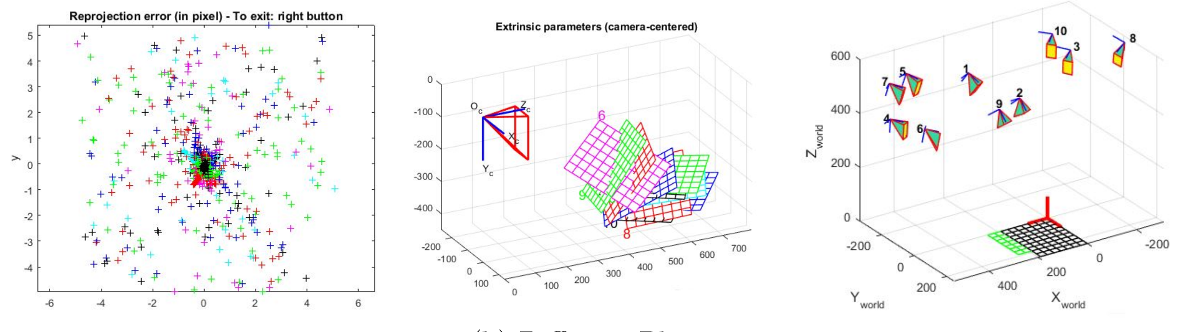


Figure 6: Calibration results

a lot when doing the calibration multiple times because I cannot be precise when I select the edges of the pattern. The error is similar to the one of gold standard.

K	Error
$\begin{pmatrix} 4062 & 0 & 1512 \\ 0 & 4062 & 2015 \\ 0 & 0 & 10 \end{pmatrix}$	1.6

Table 4: Toolbox Results



(a) Error (b) Different Planes (c) Different Views

Figure 7: Error and Calibration analysis

Note on the error : The error calculated for DLT and Gold standard is the average (on the 6 corners manually entered) of the distance between the projected 2D point and their real 2D coordinates. The error in the toolbox takes into account all corners. We must therefore be careful when comparing the errors between different method. The visual result gives a good indication of the performance of each method.

We can compare the results of the three methods in tables 5 and 6 :

K_{DLT}	$K_{goldStandard}$ (7 pts)	$K_{toolbox}$
$\begin{pmatrix} 3318 & -109 & 2216 \\ 0 & 3264 & 1072 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 3357 & 15.8 & 2017 \\ 0 & 3386 & 1446 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 4062 & 0 & 1512 \\ 0 & 4062 & 2015 \\ 0 & 0 & 10 \end{pmatrix}$

Table 5: Intrinsic Parameters

$error_{DLT}$	$error_{goldStandard}$ (6 pts)	$error_{goldStandard}$ (7 pts)	$Error_{toolbox}$
5.9	5.1	1.85	1.6

Table 6: Errors