Computer Vision Exercise 1

Viviane Yang 16-944-530

2nd October 2019

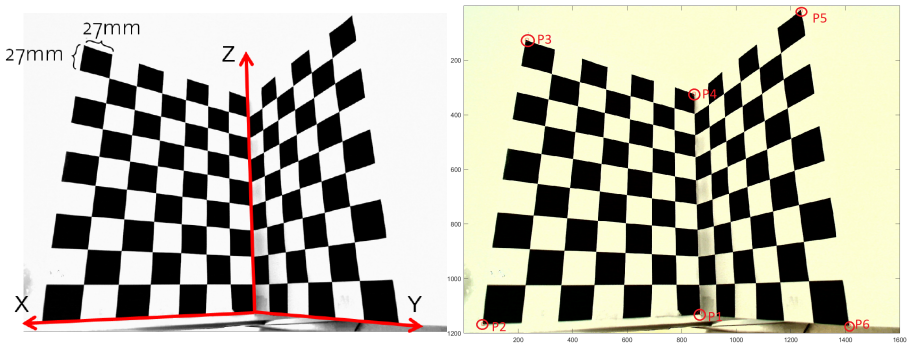
1 Notation

Table 1: Table of Notation

N	$\triangleq$	Number of points used in calibration
$\boldsymbol{x} = [x, y, 1]^T$	$\triangleq$	2D coordinates on the image plane
$\boldsymbol{X} = [X, Y, Z, 1]^T$	$\triangleq$	3D estimated coordinates
$\hat{\boldsymbol{x}} = [\hat{x}, \hat{y}, 1]^T$	$\triangleq$	normalized 2D coordinates
$\hat{\boldsymbol{X}} = [\hat{X}, \hat{Y}, \hat{Z}, 1]^T$	$\triangleq$	normalized 3D coordinates
$\mu_j, j \in \{x, y, X, Y, Z\}$	$\triangleq$	mean
$\sigma_j, j \in \{x, y, X, Y, Z\}$	$\triangleq$	scaling factor

2 Calibration of Points

Using the provided getpoints.m script we can click on the 2D image plane and add 3D coordinates to them, which are estimated by looking at the picture. I assumed that the checkered pattern is has an 90° folded angle in the middle and thus use the axis described in the exercise and in Figure 1a. However for simplicity, I choose to use the length of the square as the unit in the 3D estimated points instead of converting them to meters.



(a) 3D Coordinate Axis (b) Points selected on the Image

Figure 1: Camera Calibration

Thus, the points that were selected are:

$P_1(0,0,0), P_2(7,0,0), P_3(7,0,9), P_4(0,0,9), P_5(0,6,9), P_6(0,6,0)$

3 Data Normalization

To improving the numerical stability of the DLT, we first want to normalize the 2D and 3D data points we measured and estimated on out picture. For that, we need to normalize the points so they have a zero mean and unit variance along each axis.

$$\frac{1}{N}\sum_{i=0}^{N-1}\hat{x_i} = 0 \qquad (1) \qquad\qquad \frac{1}{N}\sum_{i=0}^{N-1}\hat{x_i}^2 = 1 \qquad (2)$$

In other words, we need to scale and shift the coordinates along their axis. We can write linear equations to transform the coordinates as such: $\hat{x_i} = \sigma_x \cdot (x_i - \mu_x)$. Then we solve for μ_x and σ_x:

$$\mu_x = \frac{1}{N}\sum_{i=0}^{N-1} x_i \qquad (3) \qquad\qquad \sigma_x = \sqrt{\frac{N}{\sum_{i=0}^{N-1}(x_i - \mu_x)^2}} \qquad (4)$$

Analogously, we can compute the mean and scale factor for the other coordinates y, X, Y and Z. To find the transformation matrices T for the 2D-coordinates and U for the 3D coordinates, we can plug in the equations such that:

$$\hat{\boldsymbol{x}} = T\boldsymbol{x} = \begin{bmatrix} \sigma_x & 0 & -\sigma_x\mu_x \\ 0 & \sigma_y & -\sigma_y\mu_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{5}$$

$$\hat{\boldsymbol{X}} = U\boldsymbol{X} = \begin{bmatrix} \sigma_X & 0 & 0 & -\sigma_X\mu_X \\ 0 & \sigma_Y & 0 & -\sigma_Y\mu_Y \\ 0 & 0 & \sigma_Z & -\sigma_Z\mu_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{6}$$

For the 6 points we have chosen in the camera calibration part, the numerical values rounded to 3 decimals of the T and U matrices are:

$$T = \begin{bmatrix} 0.002 & 0 & -1.596 \\ 0 & 0.002 & -1.290 \\ 0 & 0 & 1 \end{bmatrix} \quad U = \begin{bmatrix} 0.303 & 0 & 0 & -0.707 \\ 0 & 0.354 & 0 & -0.707 \\ 0 & 0 & 0.222 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4 Direct Linear Transform

We now want to compute the transformation matrix P using our normalized data points. For the normalized points, we first compute $\hat{P}$, the normalized transformation matrix and we get the original transformation matrix with the relation $P = T^{-1}\hat{P}U$. Since $\hat{P}$ is a 3×4-matrix, we have 12DOF and therefore need 6 2D-points to fully determine it.

Using the relationship we can solve for $\hat{\boldsymbol{P}}$ using singular value decomposition. The elements of $\hat{\boldsymbol{P}}$ are just the last column of the V matrix with $A = USV^T$:

$$\boldsymbol{A}\hat{\boldsymbol{P}} = \begin{bmatrix} -\hat{\boldsymbol{X}}_{\boldsymbol{i}}{}^T & \mathbf{0_{1\times 4}} & \hat{x_i}\hat{\boldsymbol{X}}_{\boldsymbol{i}}{}^T \\ \mathbf{0_{1\times 4}} & -\hat{\boldsymbol{X}}_{\boldsymbol{i}}{}^T & \hat{y_i}\hat{\boldsymbol{X}}_{\boldsymbol{i}}{}^T \end{bmatrix} \begin{pmatrix} \hat{\boldsymbol{P}}^{\mathbf{1}} \\ \hat{\boldsymbol{P}}^{\mathbf{2}} \\ \hat{\boldsymbol{P}}^{\mathbf{3}} \end{pmatrix} = \mathbf{0} \tag{7}$$

To get the calibration matrix K and the rotation matrix R, we use a QR decomposition of $M^{-1} = R^{-1}K^{-1}$, with M in $P = [M|-MC]$. The re-projected points are then created by multiplying P with the 3D coordinates, which are the red circles on Figure 2 and the green crosses are the clicked 2D points.

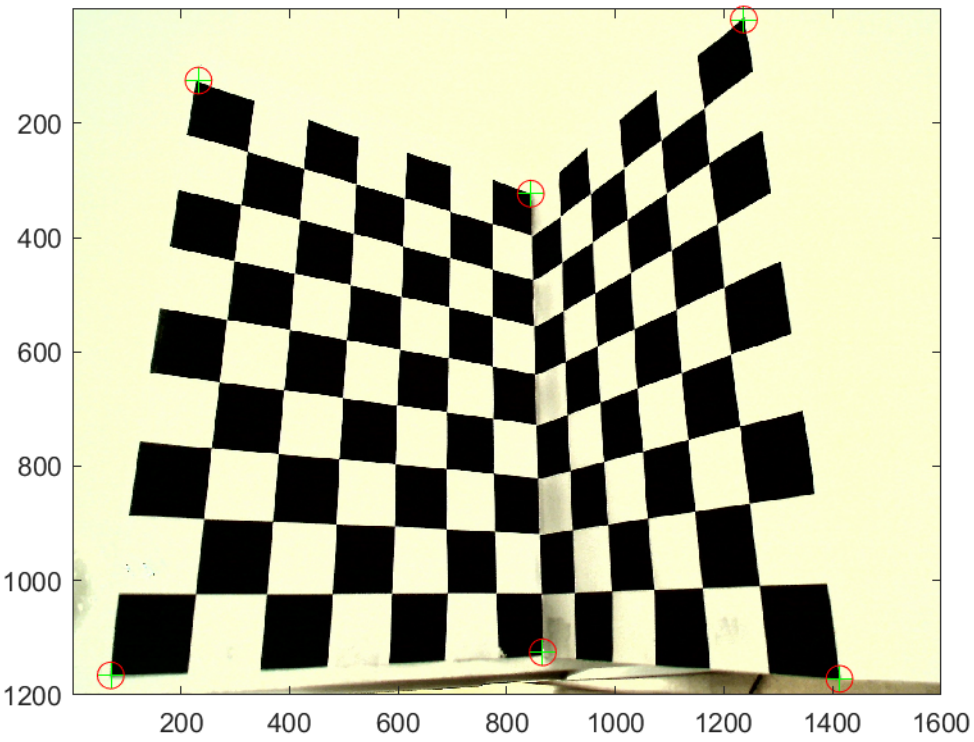


Figure 2: DLT with 6 points

The average reprojection error of the DLT is the sum of squared Euclidean distances between clicked and reprojected points: $error_{DLT} = 0.3058$

What happens if you use the unnormalized points?
We need to normalize the data first so that if we get noisy data, the noise is distributed in all directions proportionally. If the data normalization step is left out, any errors in the transformation matrix along that axis is increased.

5 Gold Standard Algorithm

The Golden Standard Algorithm just runs DLT and tries to minimize the cost function $\sum_{i=0}^{N-1} d(\hat{\boldsymbol{x}}_{\boldsymbol{i}}, \hat{\boldsymbol{P}}_{\boldsymbol{i}}\hat{\boldsymbol{X}}_{\boldsymbol{i}})$ within multiple runs. Therefore, we expect a lower error that the not optimized DLT error.

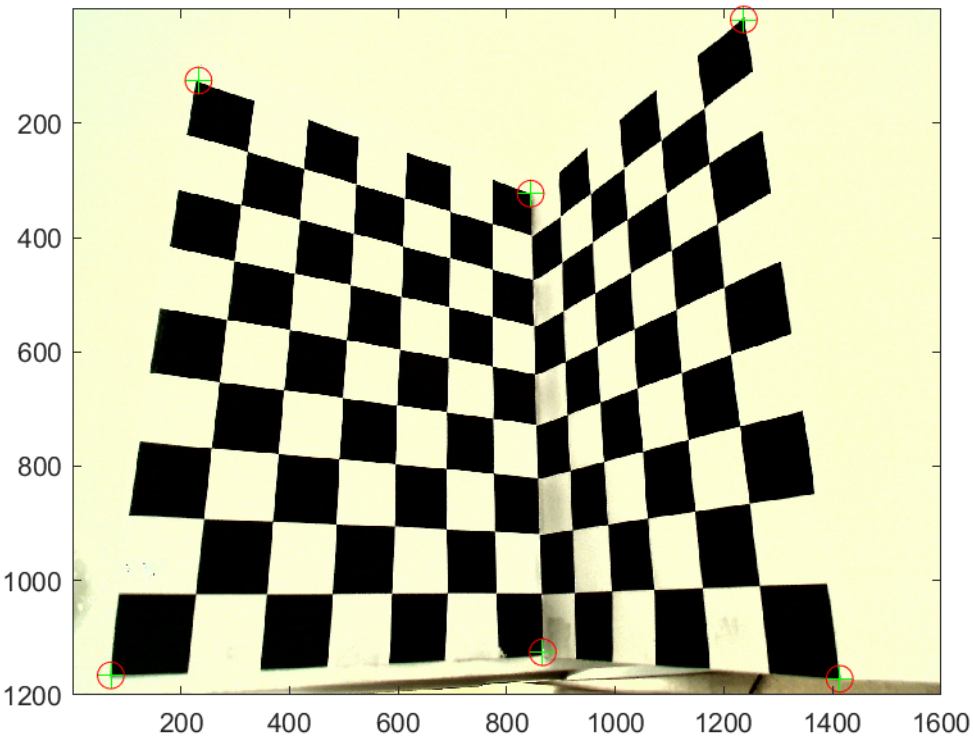


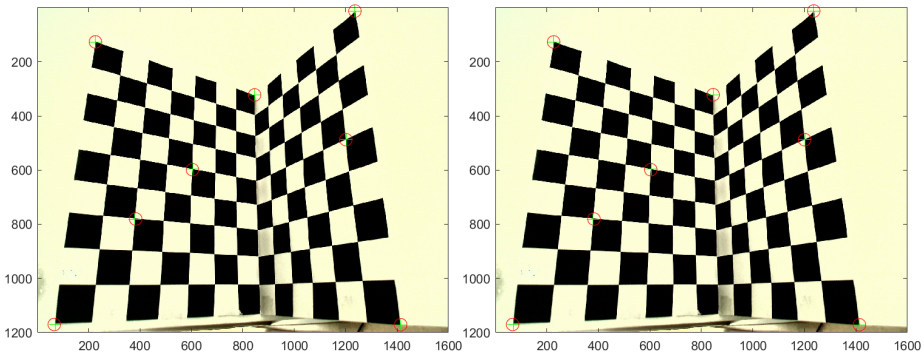
Figure 3: Gold Standard Algorithm with 6 points

The improvement is barely noticeable on the picture, but the computed average error is indeed lower:

$error_{GSA} = 0.2549$

6 Random Points

To test the implemented algorithms I also made another example using more than 6 points. 8 Points were chosen and the reprojected 2D points are visualized in Figure 4.



(a) DLT with 8 points (b) GSA with 8 points

Figure 4: Reprojection of 8 random points

Both DLT and GSA seem to perform well and the reprojected points are very close to the clicked points. By comparing the errors, we can again see some improvements while using the GSA.

$error_{DLT} = 4.0840$ $error_{GSA} = 3.8018$