COMPUTER VISION COURSE (2022/23)
Master in Electrical and Computer Engineering
Laboratory Assignment 3

Geometric Camera Calibration

1 Learning Goals and Description

In this assignment you will calibrate your own camera. The geometric camera calibration computes the parameters of a camera and estimates the distortion coefficients of the lens.

Two algorithms will be implemented:

1. the simple Direct Linear Transform algorithm
2. the Gold Standard algorithm.

As a preparatory step you must create a calibration pattern or use a previously acquires image of a 3D calibration object. An example of a typical calibration object is shown in Figure 1. When acquiring your calibration image make sure that the calibration object fills almost the whole image. The closer the checkerboard patterns are to the image borders, the more precise the calibration and especially the radial distortion can be estimated.

To run the calibration you need the image coordinates and the 3D coordinates of the calibration object corners. To get the 3D coordinates you can simply assign the origin of the coordinate system somewhere on the calibration object (as shown in Figure 1). The calibration pattern shown in figure 1 is composed of 5cm square patches. When taking multiple images, make sure to keep the coordinate systems consistent. To calibrate the camera with an image similar to the one shown in figure 1, a SINGLE image is enough, and you will need to select at least SIX points. Remember that these SIX points must not lie on the same plane. In case you decide to adquire multiple images of a single plane, your set of SIX points must be extracted from at least TWO images.

Your submission for this assignment should be a zip file, `<ClassIDNamesID.zip>`, composed of your report, your Matlab implementations (including any helper functions), and, if the case, your implementations and results for extra credits (optional).

Your final upload should have the files arranged in this layout:

`<ClassIDNamesID>.zip`

- `<NamesID>`
  - `<NamesId.pdf> - PDF Assignment report`
  - `Matlab`
    - `CamCalScript.m`
    - `decomposeQR.m`
    - `decomposeEXP.m`
    - `dlt.m`

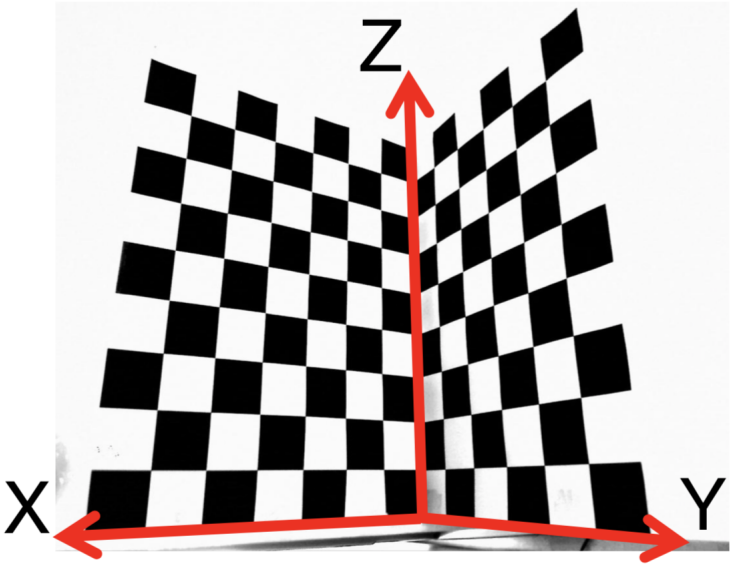

Figure 1: Typical Calibration Pattern.

    - `fminGold.m`
    - `fminGoldRadial.m`
    - `getpoints.m`
    - `normalization.m`
    - `runDLT.m`
    - `runGold.m`
    - `runGoldRadial.m`
    - `Other functions needed`
  - `Extra - Optional`
    - `Other functions needed`

Please make sure that any file paths that you use are relative and not absolute. Not `imread('/name/Documents/hw4/data/xyz.jpg')` but `imread('../data/xyz.jpg')`.

Every script and function you write in this section should be included in the `matlab/` folder. Please include resulting images in your report.

2 Assignment Pipeline

This assignment has several parts: Data Normalization, Direct Linear Transform, Camera Parameters Decomposition, Gold Standard Algorithm and Gold Standard Algorithm with radial distortion estimation.

Data Normalization Data normalization is an essential part of the `DLT` algorithm. First, for all points, scale the homogeneous vectors such that the last entry becomes 1, i.e., $p = (x, y, 1)^T$ and $P = (X, Y, Z, 1)^T$.

Then, transform the input points so that the centroid of the points is at the origin and that the average Euclidean distance from the origin is $\sqrt{2}$ for the image points, and $\sqrt{3}$ for the object points.

NOTE: The distance to the origin should not contain the last value of the homogeneous coordinate, so for instance, the distance to the origin for image points is $\sqrt{x^2 + y^2}$.

Find transformations matrices $\mathbf{T}$ and $\mathbf{U}$ such that the normalized image points are $\hat{p}_i = \mathbf{T} \cdot p_i$ and the normalized object points are $\hat{P}_i = \mathbf{U} \cdot P_i$. Make sure the last entries remain as 1.

$$\mathbf{T} = \begin{bmatrix} s_{2D} & 0 & c_x \\ 0 & s_{2D} & c_y \\ 0 & 0 & 1 \end{bmatrix}^{-1} \qquad \mathbf{U} = \begin{bmatrix} s_{3D} & 0 & 0 & C_x \\ 0 & s_{3D} & 0 & C_y \\ 0 & 0 & s_{3D} & C_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \tag{1}$$

Direct Linear Transform

Implement the `DLT` algorithm to estimate the camera matrix from the given set of point correspondences. The goal of the `DLT` is to minimize the algebraic error of the linear system of our camera model.

- Compute the normalized camera matrix $\hat{\mathbf{M}}$ with the `DLT` algorithm:
  Each correspondence $\hat{P}_i \leftrightarrow \hat{p}_i$ is inserted to the equation

$$\begin{bmatrix} \omega_i \hat{P}_i^T & 0^T & -x_i \hat{P}_i^T \\ 0^T & \omega_i \hat{P}_i^T & -y_i \hat{P}_i^T \end{bmatrix} \begin{bmatrix} \hat{M}^1 \\ \hat{M}^2 \\ \hat{M}^3 \end{bmatrix} = 0 \tag{2}$$

  where $\hat{M}^i$ are the row vectors of $\hat{\mathbf{M}}$.

  Stacking the equations for all n points correspondences results in a $2n \times 12$ matrix `A` where $\mathtt{A} \cdot \mathbf{M} = 0$. The solution for $\mathbf{M}$ is the right null-vector of `A` that can be computed with the singular value (`SVD`) decomposition of `A`.
- Compute the denormalized camera matrix $\mathbf{M} = \mathbf{T}^{-1}\hat{\mathbf{M}}\mathbf{U}$ using the matrices from last stage.

Camera Matriz Decomposition

Once $\mathbf{M}$ is known, you still got to recover the intrinsic and extrinsic parameters! This is a decomposition problem, **NOT** an estimation problem.

In this assignment you are going to use TWO different strategies: The QR-Factorization and the EXPlicit decomposition;

QR-Factorization:

- Factorize the camera matrix into the intrinsic camera matrix $\mathbf{K}$, a rotation matrix $\mathbf{R}$ and the camera center C.

$$\mathbf{M} = [\mathbf{S}| - \mathbf{S}C] = \mathbf{K}\,[\mathbf{R}| - \mathbf{R}C] \tag{3}$$

  where $\mathbf{S} = \mathbf{KR}$ is the product of the intrinsic matrix and the camera orientation rotation matrix and C is the camera center. The matrix $\mathbf{S}$ can be decomposed with a `QR-decomposition`.

  NOTE : The `QR-decomposition` of a matrix $\mathbf{A} = \mathbf{QR}$, where the matrix $\mathbf{Q}$ is orthogonal and the matrix $\mathbf{R}$ is upper-triangular.

To obtain $\mathbf{K}$ and $\mathbf{R}$, run `QR-decomposition` on the inverse of the left 3×3 part of $\mathbf{M}$, i.e, matrix $\mathbf{S}^{-1}$. Invert both result matrices to get $\mathbf{K}$ and $\mathbf{R}$.

The camera center C is the point for which $\mathbf{M}C = 0$, which means that it corresponds to the the right null-vector that can be obtained from the `SVD` of $\mathbf{M}$.

EXPlicit Decomposition:

- Let us explicitly represent the camera matrix $\mathbf{M}$ as

$$\mathbf{M} = \begin{bmatrix} \alpha r_1^T - \alpha cot\theta r_2^T + u_0 r_3^T & \alpha t_x - \alpha cot\theta t_y + u_0 t_z \\ \frac{\beta}{sin\theta} r_2^T + v_0 r_3^T & \frac{\beta}{sin\theta} t_y + v_0 t_z \\ r_3^T & t_z \end{bmatrix} \tag{4}$$

where $\alpha = k_x f$, $\beta = k_y f$, θ is the angle between the two image axes ($\cong 90°$), (u_0, v_0) the coordinates of the Principal Point and r_i^T and t_i denote, respectively, the three rows of the rotation matrix $\mathbf{R}$ and the coordinates of the vector $\mathbf{t}$ in the frame attached to the camera.

Note that in this setting, $\mathbf{M} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$ and the matrix $\mathbf{M}$ determines the coordinate vector $\mathbf{C}$ of the camera's optical center in the world coordinate system

$$\mathbf{M}\begin{bmatrix}\mathbf{C} \\ 1\end{bmatrix} = 0. \tag{5}$$

In particular, if $\mathbf{M} = \begin{bmatrix}\mathbf{A} & \mathbf{b}\end{bmatrix}$ then $\mathbf{C} = -\mathbf{A^{-1}b}$.

In this setting, $\mathbf{M}$ is only defined up to scale. If we write as before ($\rho\mathbf{M} = \rho\begin{bmatrix}\mathbf{A} & \mathbf{b}\end{bmatrix}$), we have

$$\rho\begin{bmatrix} a_1^T \\ a_2^T \\ a_3^T \end{bmatrix} = \begin{bmatrix} \alpha r_1^T - \alpha cot\theta r_2^T + u_0 r_3^T \\ \frac{\beta}{sin\theta} r_2^T + v_0 r_3^T \\ r_3^T \end{bmatrix}. \tag{6}$$

Considering the properties of the rotation matrix, i.e., rows have unit length and are perpendicular to each other, yields

$$\begin{cases} \rho = \frac{\varepsilon}{|a_3|}, \\ r_3 = \rho a_3, \\ u_0 = \rho^2 (a_1 \cdot a_3), \\ v_0 = \rho^2 (a_2 \cdot a_3), \end{cases} \tag{7}$$

where $\varepsilon = \mp 1$.

In addition, we have

$$\begin{cases} \rho^2 (a_1 \times a_3) = -\alpha r_2 - \alpha cot\theta r_1 \\ \rho^2 (a_2 \times a_3) = \frac{\beta}{sin\theta} r_1, \end{cases} \tag{8}$$

and

$$\begin{cases} \rho^2 |a_1 \times a_3| = \frac{|\alpha|}{sin\theta} \\ \rho^2 |a_2 \times a_3| = \frac{|\beta|}{sin\theta}, \end{cases} \tag{9}$$

since θ is always in the neighborhood of $\frac{\pi}{2}$ with a positive sine, and it follows that

$$\begin{cases} cos\theta = -\varepsilon_u \varepsilon_v \frac{(a_1 \times a_3) \cdot (a_2 \times a_3)}{|a_1 \times a_3||a_2 \times a_3|}, \\ \alpha = \varepsilon_u \rho^2 \left| a_1 \times a_3 \right| sin\theta, \\ \beta = \varepsilon_v \rho^2 \left| a_2 \times a_3 \right| sin\theta, \end{cases} \tag{10}$$

where $\varepsilon_u = \frac{\alpha}{|\alpha|}$ and $\varepsilon_v = \frac{\beta}{|\beta|}$.
We can now compute r_1 and r_2 from equation 9 as

$$\begin{cases} r_1 = \frac{\rho^2 sin\theta}{\beta} \left(a_2 \times a_3 \right) = \frac{1}{|a_2 \times a_3|} \left(a_2 \times a_3 \right), \\ r_2 = r_3 \times r_1. \end{cases} \tag{11}$$

Note that there are **four** possible choices for the matrix $\mathbf{R}$ depending on the values of ε and ε_v.
Finally, the translation parameters are recovered by writing

$$\rho \begin{bmatrix} \alpha t_x - \alpha cot\theta t_y + u_0 t_z \\ \frac{\beta}{sin\theta} t_y + v_0 t_z \\ t_z \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{12}$$

In pratical situations, the sign of t_z is often known in advance (this corresponds to knowing whether the origin of the world coordinate system is in front or behind the camera), which allows the choice of a unique solution for the calibration parameters.

Performance Evaluation:

- Using the computed camera matrix $\mathbf{M}$, obtain the reprojected points of all checkerboard corners in the calibration object.

$$\begin{cases} x = \frac{m_1 \cdot P}{m_3 \cdot P} \\ y = \frac{m_2 \cdot P}{m_3 \cdot P}. \end{cases} \tag{13}$$

  Draw the reprojected points, as shown in figure 2, and compute the sum of squared reprojection errors (see below) to evaluate the performance of the calibration procedure.
- What happens if you use the unnormalized points?
- How different are the camera's parameters obtained with both decomposition strategies ?
- In your opinion what are the Pros/Cons of both strategies ?

Gold Standard algorithm

The Gold Standard algorithm minimizes the geometric error $\sum_i d(p_i, \tilde{p}_i)^2$ where p_i is the measured (clicked) point and $\tilde{p}_i$ is the projected object point $\tilde{p}_i = \mathbf{M} P_i$. Here $d(p_i, \tilde{p}_i)$ is the Euclidean distance between p_i and $\tilde{p}_i$.

- Normalize the input points and run the `DLT` algorithm to get an initial camera matrix $\tilde{\mathbf{M}}$ for the optimization.

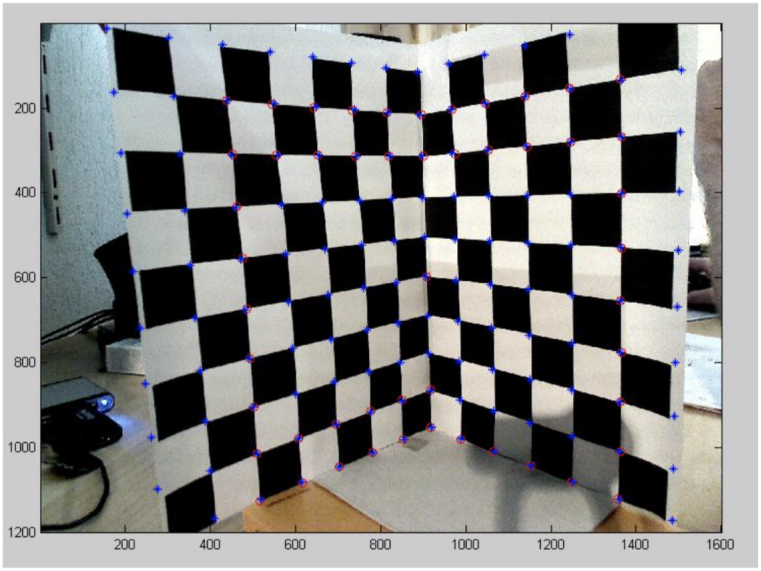

Figure 2: Reprojection of the 3D points.

- Visualize the hand-clicked points and the reprojection of the points on the calibration object obtained by using the computed camera matrix.
- Compute optimal $\tilde{\mathbf{M}}$ by minimizing the sum of squared reprojection errors

$$\arg\min_{\tilde{\mathbf{M}}} \sum_{i=1}^{n} dist(\hat{p}_i, \tilde{\mathbf{M}}\hat{P}_i)^2 \tag{14}$$

  NOTE: To compute optimal $\tilde{\mathbf{M}}$ use `Matlab`'s function `fminsearch(...)`. Explore how to use this function.
- Denormalize and decompose the camera matrix using the strategies described before.
- Visualize the reprojected points of all checkerboard corners on the calibration object with the computed camera matrix.

Gold Standard algorithm with radial distortion estimation

During the optimization of $\tilde{\mathbf{M}}$ also the coefficients of the radial distortion introduced by the lens can be estimated. The standard model for radial distortion is

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = L(\check{r}) \begin{bmatrix} \check{x} \\ \check{y} \end{bmatrix} \tag{15}$$

where $(\check{x}, \check{y})$ is the ideal linear projection of a 3D point P

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [\mathbf{R}| - \mathbf{R}C]\, P \tag{16}$$

$$\check{x} = \frac{x}{z} \qquad \check{y} = \frac{y}{z} \tag{17}$$

and (x_d, y_d) is the position after radial distortion. To compute the actual position in the image, the position $(x_d, y_d, 1)$ has to be multiplied with the intrinsic matrix $\mathbf{K}$. $\check{r}$ is the radial distance $\sqrt{\check{x}^2 + \check{y}^2}$ from the center for radial distortion and $L(\check{r})$ is the distortion factor, which is a function of the radius $\check{r}$ only.

The standard approximation of $L(r)$ may be given by its `Taylor` expansion $L(r) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \ldots$ where $r > 0$ and $L(0) = 1$. The center of the radial distortion can be assumed to be the same as the principal point (p_x, p_y).

Add the radial distortion model to the computation of the projection of $\tilde{P}_i$ (when computing the reprojection-error $dist(\hat{x}, \tilde{\mathbf{M}}\hat{P}_i)^2$) in the Gold Standard minimization and estimate the first two radial distortion coefficients κ_1 and κ_2.

NOTE: To compute the optimal parameters for the camera use again the `Matlab`'s function `fminsearch(...)`.

3 Extra (Optional) credits

Image undistortion

If the radial distortion coefficients are known, it is also possible to warp the whole image so that all the radial distortion vanishes. To accomplish this, two strategies can be used: direct or inverse image warp.

Using a direct warp strategy we simply map the distorted image pixels $(x_d^{'}, y_d^{'})$ into the corresponding undistorted ones $(x^{'}, y^{'})$ using

$$\begin{aligned} \check{x}^{'} &= p_x + L(r)(x_d^{'} - p_x) \\ \check{y}^{'} &= p_y + L(r)(y_d^{'} - p_y) \end{aligned} \tag{18}$$

where $r^2 = (x_d)^2 + (y_d)^2$.

For the inverse warp strategy, an undistorted image is built by sampling the corresponding pixel in the distorted image $(x_d^{'}, y_d^{'})$ using

$$\begin{aligned} x_d^{'} &= p_x + L(\check{r})(\check{x}^{'} - p_x) \\ y_d^{'} &= p_y + L(\check{r})(\check{y}^{'} - p_y) \end{aligned} \tag{19}$$

where $\check{r}^2 = (\check{x})^2 + (\check{y})^2$ and $(\check{x}^{'}, \check{y}^{'})$ represent the pixel coodinates of the undistorted point.

Note that in our formulation we modelled $L(\check{r})$ as distortion factor and not $L(r)$, which means that our formultion is suitable for the inverse warp strategy.

Due to the discrete nature of digital images, sampling data points with sub-pixel precision can be done using a bilinear interpolation or a nearest-neighbour approach.

Show the undistorted image of the calibration pattern.

4 Experiments

Now you're ready to go!

Using the *CamCalScript* that we provide, you can load the calibration pattern image stored in folder `/images`. The code framework provides a `getpoints` function that you can use to acquire a set of corresponding points. Use this set of points to calibrate your camera. For each clicked image point you have to enter the 3D-coordinate of the correspondent point in the calibration object.

Implement the necessary code in the `Matlab` functions supplied in order to accomplish the following three calibration tasks:

1. DLT Algorithm : `[K, R, t, error] = runDLT(xy, XYZ, Decomp);`
2. Gold Algorithm : `[K, R, t, error] = runGold(xy, XYZ, Decomp);`
3. Gold Algorithm with Radial Distortion : `[K, R, t, dist, error] = runGoldRadial(xy, XYZ, Decomp);`

For each one of the calibration tasks, evaluate the performance of both strategies for decomposing the camera matrix into its intrinsic and extrinsic parameters.

For extra-credits, do:

1. Implement a function that removes radial distortion from images and show both images : distorted (original) image and undistorted (corrected) image.
2. Replace the boring task of image clicking used before by and automatic corner points detector. For that purpose use the Harris corners detector implemented in assignment 2 : `HarrisCorner.m`.

To better validate your calibration algorithm, build your own calibration pattern and acquire a similar image with your own smartphone camera. Run the *CamCalScript* and evaluate the parameters obtained and the 3D points reprojection error.

Write a short report explaining the main steps of your implementation and discussing the results of the methods. The report should contain images showing the clicked $2D$ points before calibration and the reprojected $3D$ points using the calibration parameters. In your report, you should describe how well your code worked on different images, what effect do the parameters have and any improvements you made to your code to make it work better.

5 Hints

Useful `Matlab` functions:

- `ginput`
- `cross`
- `maketform`
- `imtransform`

Be careful when using `maketform`. Here, transformation matrices are applied to image coordinates from the right side. This means that you may need to transpose your transformation matrix.

- Work with normalized homogeneous coordinates always. Camera calibration $\mathbf{K}$ should respect this convention.
- When using the QR-Factorization check that the obtained orientation $\mathbf{R}$ correspond to the expected world value, otherwise $\mathbf{K}$ will have negative values in its diagonal.
- When reprojecting points use average of reprojection error for comparison and remember to normalize reprojected coordinates to $w = 1$.
- Remember to use the same camera with the same settings for all tasks!