Chapter 7 – Camera Calibration

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Camera Calibration

Camera calibration is needed for recovering 3D metric from images (3D reconstruction, object localization etc.) but most importantly, it’s needed for 3D (real world) – 2D (camera image) correlation.

Calibration means actually finding the internal and external parameters of the camera.

- 3D Points in the real-world coordinate : $\boldsymbol{X_w} = (X_w, Y_w, Z_w)^T$
- 3D Points in the camera coordinate : $\boldsymbol{X_c} = (X_c, Y_c, Z_c)^T$
- 2D Points in image coordinate : $\boldsymbol{x} = (x, y)^T$

1.2 Projection Matrix

In order to pass from a 3D image into a 3D camera image and then to a normal 2D image we use the Projection Matrix P, which is a relation between X_w and $\boldsymbol{x}$, such that $\boldsymbol{x} = \boldsymbol{P} \cdot \boldsymbol{X_w}$

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

There is a number “1” in the X_w vector since we are working with homogeneous coordinates (meaning that we add another dimension, correlated to the other ones). In this way we introduce the so called “**Scale ambiguity**”, which is represented mathematically in the homogeneous coordinates.

The s (which is the scaling factor), is a value $\neq 0$ which gets multiplied by the point vector:

$$\begin{array}{ccccc} & & [2] & & [4] \\ 2 & \cdot & [2] & => & [4] \\ & & [1] & & [2] \end{array}$$

These quantities are referring to the same exact pixel, but at different scale (s = 2).

1.3 Intrinsic and extrinsic parameters

The Projection Matrix P can be decomposed in 2, intrinsic and extrinsic parameters.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

The one on the left is the A matrix, **intrinsic** matrix. The one on the right is the R matrix, also known as the Rotation Matrix along with the T vector (translation vector), which together with the X_w vector, form the extrinsic matrix.

αx and αy is the focal length of the camera, while x_0 and y_0 is the camera center (where the camera is located).

As for the **extrinsic** parameters, we need this in order to pass from a 3D point in the world coordinate to a 3D point in the camera image, which can be explained as follows:

$$\mathbf{X}_c = [\mathbf{R}|\mathbf{t}]\,\mathbf{X}_w,$$
$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

In the end, we can rewrite passing from a 3D representation of the image in the world coordinate, to a 2D representation through this simple formula:

$$\mathbf{x} = \mathbf{P}\mathbf{X}_w = \mathbf{A}\,[\mathbf{R}|\mathbf{t}]\,\mathbf{X}_w,$$
$$\rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

=>

$$\mathbf{x} = \mathbf{A}\,[\mathbf{R}|\mathbf{t}]\,\mathbf{X}_w = \mathbf{A}\mathbf{X}_c,$$
$$\rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

1.4 Steps for passing from a 3D world point to a 2D image point

We start off by saying that this is only possible if we know the parameters of the camera, meaning that the camera must be calibrated.

<u>STEP 1 – 3D World point to 3D camera point</u>

As we already said, we need a rotation matrix + translation vector applied on top of the 3D world point. This gives us a vector of 3D points in the camera coordinate multiplied by a scaling factor.

<u>STEP 2 – 3D Camera point to 2D camera point</u>

In order to pass from a 3D camera coordinate to a 2D camera coordinate, we need to retrieve C_{X_u} and C_{y_u}

Which are denoted as:

$$^{C}X_u = \frac{f}{^{W}Z_w}\,^{W}X_w \qquad ^{C}Y_u = \frac{f}{^{W}Z_w}\,^{W}Y_w,$$

This means that W_{X_w}, W_{Y_w} and W_{Z_w} are the X, Y and Z 3D points in the world coordinate while f is the focal length. With this simple formula, we have the 2D camera points always multiplied by a scaling factor ------------------------------------>

$$s\begin{bmatrix} ^{C}X_u \\ ^{C}Y_u \\ 1 \end{bmatrix}$$

<u>STEP 3 – Lens distortion</u>

We always need to consider some lens distortion (like fisheye) when passing from a 3D representation to a 2D representation, which is specified by δx and δy (distortion along that x and y axis) summed to their respective C_{X_u} and C_{Y_u}. If $\delta x, \delta y = 0$ it means that there is, of course, no lens distortion.

<u>STEP 4 – 2D Camera point to 2D image point</u>

So, in the final step here, we want to pass from a 2D camera point in the metric system to a 2D image point in the pixel system.

$$s\begin{bmatrix} {}^{I}X_d \\ {}^{I}Y_d \\ 1 \end{bmatrix} = \begin{bmatrix} -k_u & 0 & u_0 \\ 0 & -k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{C}X_d \\ {}^{C}Y_d \\ 1 \end{bmatrix},$$

$${}^{I}X_d = -k_u {}^{C}X_d + u_0 \qquad {}^{I}Y_d = -k_v {}^{C}Y_d + v_0,$$

Very briefly, we multiple the C_{X_d} vector by a matrix of $-k_u$ and $-k_v$, which are denoted as the parameters of transformation from metric measures to pixel, while u_0 and v_0 define the final projection of the focal point into the 2D plane (like some sort of shift).

1.5 Estimating Projective Matrix P

So, with this being said, the general idea of camera calibration is to estimate the Projective Matrix P and from that the component A, R, and t.

From 1.2, we recover the general P matrix and set $p_{34} = 1$, due to the scale ambiguity factor, removing essentially one DoF and generalizing the i-th image point as follows:

$$x_i = \frac{X_i p_{11} + Y_i p_{12} + Z_i p_{13} + p_{14}}{X_i p_{31} + Y_i p_{32} + Z_i p_{33} + 1}$$
$$y_i = \frac{X_i p_{21} + Y_i p_{22} + Z_i p_{23} + p_{24}}{X_i p_{31} + Y_i p_{32} + Z_i p_{33} + 1}$$

This fraction (both x_i and y_i) is coming from multiplying (for e.g.) the first row · column vector etc. and so if I want to recover x_i and y_i, I divide by the multiplication between the 3rd row · column vector (note the 1 next to Z_{iP33} due to the fact that we set $p_{34} = 1$).

We can solve this optimization problem w.r.t to **P**, in 2 ways:

1. Linear Method -> **Ax = b**
2. Linear Method -> **Ax = 0**

1.5.1 Resolution of Ax = b

In the resolution n.1, Hall et al. proposed a way to retrieve x_i and y_i, as follows:

$$X_i p_{11} + Y_i p_{12} + Z_i p_{13} + p_{14} - x_i X_i p_{31} - x_i Y_i p_{32} - x_i Z_i p_{33} = x_i \quad (27)$$
$$X_i p_{21} + Y_i p_{22} + Z_i p_{23} + p_{24} - y_i X_i p_{31} - y_i Y_i p_{32} - y_i Z_i p_{33} = y_i \quad (28)$$

Where (27) and (28) are 2D points while x_i and y_i is ONE ASSOCIATED PIXEL to that 2D point. We notice that in the formulas, there is x_i and y_i, meaning that we have to find these values before actually retrieving them. Also, the P vector is unknown. With this being said, we pass from (27) and (28) to an equation like **Ap = b**, where **A** ∈ $\mathbb{R}^{2N\ X\ 11}$ (Point matrix), **p** ∈ $\mathbb{R}^{11}$ (vector of 11 elements, projection matrix parameters vector), **b** ∈ $\mathbb{R}^{2N}$ (vector of x_1, y_1 … x_N, y_N which are the 2D points vector).

Very simply, the solution of this equation **Ap = b** is given by minimizing an energy function as follows:

$$E_1 = \|Ap - b\|^2$$

Where $\hat{p} = argmin\ E_1 = argmin\ \|Ap - b\|^2 = argmin\ (Ap - b)^T \cdot (Ap - b)$

If we differentiate by E_1 w.r.t. p, we have something like this:

$$\frac{\partial E_1}{\partial \mathbf{p}} = 0$$
$$\rightarrow \mathbf{A}^\mathsf{T}(\mathbf{A}\hat{\mathbf{p}} - \mathbf{b}) = 0$$
$$\rightarrow \mathbf{A}^\mathsf{T}\mathbf{A}\hat{\mathbf{p}} = \mathbf{A}^\mathsf{T}\mathbf{b}$$
$$\rightarrow \hat{\mathbf{p}} = (\mathbf{A}^\mathsf{T}\mathbf{A})^{-1}\mathbf{A}^\mathsf{T}\mathbf{b}$$

Which is actually a pretty easy equation, considering the matrix multiplications and the fact that if we divide by $A^T A$, trying to isolate $\hat{p}$, we multiply by the inverse in the other part of the equation. Of course, $\hat{p}$ can only be estimated if $A^T A$ is actually invertible, meaning that this resolution Ax = b relies on whether this matrix is invertible or not.

If not, we pass on the Ax = 0 resolution.

1.5.2 Resolution of Ax = 0

So, remembering the equation in the 1.5.1 paragraph, we simply put these (27) and (28) equal to 0, retrieving more or less the same set of equation of 1.5.1, but we are now trying to resolve **Ap = 0.**

Again, through the use of an energy function such as $E_2 = \|Ap\|^2$ and the constraint that $\|p\|^2 - 1 = 0$ (which is needed in order to not let p become a zero-vector).

We introduce, after this new energy function, also the Lagrange multiplier (λ) which is a value > 0 which gives us the certainty that p is not equal to 0. The equation becomes the following:

$$\begin{aligned} E_2(\mathbf{p}, \lambda) &= \|\mathbf{Ap}\|^2 - \lambda(\|\mathbf{p}\|^2 - 1) \\ &= (\mathbf{Ap})^\mathsf{T}(\mathbf{Ap}) - \lambda(\mathbf{p}^\mathsf{T}\mathbf{p} - 1). \end{aligned}$$

where in the end we are trying to find the argmin w.r.t. P of this quantity in the last row. Through various matrix multiplications, through differentiating E_2 by p and λ, we can derive that minimizing $\|Ap\|^2$ is equal to minimize the value of λ.

In the end, due to an intermediate step, we can conclude that $\hat{p}$ is the eigenvector of $A^T A$ and λ is the corresponding eigenvalue, which should be minimized as much as possible (ideally 0).

So, now that we have an estimate of the projective matrix P, we can decompose it into A, R and T.

1.6 Decomposition into A, R and T

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

We set this matrix P as follows: P = A[R|t].

We have two ways, one is finding the camera center C, thus resolving Pc = 0 with Singular Value Decomposition (SVD) of P where c is the Eigenvector corresponding to the smallest Eigenvalue, while for finding the intrinsic matrix A and rotation R we use the RQ Decomposition.