Camera Calibration

Pramook Khungurn

December 11, 2019

The material in this note is lifted from relevant sections of Szeliski's *Computer Vision* book. A note on notation is that if $\mathbf{v}$ is a vector, then $\bar{\mathbf{v}}$ is its homogeneous coordinate. Therefore, if $\mathbf{v} \in \mathbb{R}^d$, then $\bar{\mathbf{v}} \in \mathbb{R}^{d+1}$.

1 Camera Intrinsics

- Pixels are indexed by **pixel coordinates** $\mathbf{x}_s = (x_s, y_s)$. We let $\bar{\mathbf{x}}_s$ denote the homogeneous coordinate of (x_s, y_s). That is,
$$\bar{\mathbf{x}}_s \sim \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$
where $\sim$ denotes "equals modulo a scaling factor."

- Most coordinate systems starts at the top left of the image plane. We let $\mathbf{c}_s$ denote 3D coordinate of this top left point, which we call the **origin**.

- $\mathbf{O}_c$ denote the camera's center of projection.

- We let $\mathbf{p}_c$ denote a *camera-centered* point, which is a point written in a coordinate system where $\mathbf{O}_c$ is the origin. Let $\mathbf{p}$ be the projection of $\mathbf{p}_c$ on to the image plane.

- To map $\bar{\mathbf{x}}_s$ to a 3D point on the image place, we first scale it by the pixel spacing (s_x, s_y), and then rotate it with a 3D rotation $\mathbf{R}_s$ and then add the resulting vector to the origin $\mathbf{c}_s$:
$$\mathbf{p} = \begin{bmatrix} \mathbf{R}_s & \big| & \mathbf{c}_s \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \mathbf{M}_s \bar{\mathbf{x}}_s.$$
Here, $\mathbf{M}_s$ is called the **sensor homography**.

- The matrix $\mathbf{M}_s$ has eight parameters:
  - 3 parameters to describe the rotation $\mathbf{R}_s$.
  - 3 parameters to describe the center $\mathbf{c}_s$.
  - 2 parameters to describe the scale (s_x, s_y).

- In practice, unless we have accurate information about sensor spacing or sensor orientation, there are only 7 degrees of freedom.

  The reason is that the distance from the sensor to the origin cannot be separated from sensor spacing, based on external image measurement alone.

- The relationship between the pixel center $\mathbf{p}$ and the camera-centered point $\mathbf{p}_c$ is given by an unknown scaling factor α, where $\mathbf{p} = \alpha \mathbf{p}_c$. Hence,

$$\bar{\mathbf{x}}_s = \mathbf{M}_s^{-1}\mathbf{p} = \alpha \mathbf{M}_s^{-1}\mathbf{p}_c$$

- The **calibration matrix K** is defined as

$$\mathbf{K} = \alpha \mathbf{M}_c^{-1}.$$

- Literature treats $\mathbf{K}$ as being an upper triangular matrix with having only 5 degrees of freedom instead of the full 7 or 8. This is because we cannot retrieve all the degrees of freedom using external measurements alone.

2 Camera Pose Estimation

- The input a number of known 3D locations $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \dots, \mathbf{p}_n$ and its corresponding screen coordinates $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$.
- We would like to find a 3×4 matrix $\mathbf{C}$ such that $\bar{\mathbf{x}}_i = \mathbf{C}\bar{\mathbf{p}}_i$. This is called the **camera matrix**.
- After we have $\mathbf{C}$, we can retrieve the calibration matrix $\mathbf{K}$ by the following equation:

$$\mathbf{C} = \mathbf{K} \left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \end{array} \right]$$

  where $\mathbf{K}$ is upper triangular and $\mathbf{R}$ is a 3×3 matrix.

- $\mathbf{K}$ in the above equation can be found by QR factorization. That is, we know that

$$\mathbf{C}^T = QR = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} * & * & * \\ & * & * \\ & & * \end{bmatrix}$$

  Note that, $P^T = QEER$ where

$$E = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

  Hence, we can write $P = R^T E E Q^T$. Note that $R^T E$ is upper triangular because

$$\begin{aligned} R^T E &= \left(\begin{bmatrix} * & * & * \\ & * & * \\ & & * \end{bmatrix} \right)^T \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} * & & \\ * & * & \\ * & * & * \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} * & * & * \\ & * & * \\ & & * \end{bmatrix}. \end{aligned}$$

  Therefore, we can set $\mathbf{K} = R^T E$ and $[\mathbf{R}|\mathbf{t}] = EQ^T$.

2.1 A Simple Analytic Algorithm

- Consider a point $\mathbf{p}_i = (X_i, Y_i, Z_i)$ and its screen coordinate $\mathbf{x}_i = (x_i, y_i)$. We know that

$$x_i = \frac{c_{00}X_i + c_{01}Y_i + c_{02}Z_i + c_{03}}{c_{20}X_i + c_{21}Y_i + c_{22}Z_i + c_{23}}$$
$$y_i = \frac{c_{10}X_i + c_{11}Y_i + c_{12}Z_i + c_{13}}{c_{20}X_i + c_{21}Y_i + c_{22}Z_i + c_{23}}$$

- The above equations give us the following system of linear equations:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & & & & & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ & & & & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & & & & & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ & & & & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ & & & & & & & \vdots & & & & \\ X_n & Y_n & Z_n & 1 & & & & & -x_nX_n & -x_2Y_n & -x_nZ_n & -x_n \\ & & & & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_2Y_n & -y_nZ_n & -y_n \end{bmatrix} \begin{bmatrix} c_{00} \\ c_{01} \\ \vdots \\ c_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

In other words, if we let

$$\mathsf{c} = \begin{bmatrix} c_{00} \\ c_{01} \\ \vdots \\ c_{23} \end{bmatrix}$$

and we let A denote the big matrix on the right, then we want to solve the equation

$$\mathsf{A}\mathsf{c} = \mathbf{0}.$$

- Note that if we find $\mathsf{c} \neq \mathbf{0}$ that is a solution to the above system, then any matrix of the form $k\mathsf{c}$ should also be a solution to the system. This implies that A should be rank deficient.

  Because c has dimention 12 and each 3D-2D correspondence gives us 2 equations, we need at least 6 points in order to be able to solve the equation. However, we need more than 6 points to ensure that A is actually rank deficient.

- To actually find c, we may set one component of c, say c_{23}, to 1, and get a smaller system of linear equations to solve.

- Alternatively, since $\mathbf{A}$ is rank deficient, its smallest singular value is 0 (or should be close to 0). We can take $\mathbf{c}$ to be the singular vector corresponding to this singular value.

- This method is called the **direct linear transform** (DLT).

2.2 Iterative Algorithms

- Instead of solving for the entries of $\mathbf{C}$, we try to minimize the reprojection error of 2D points as a function of unknown pose parameters in $(\mathbf{R}, \mathbf{t})$ and $\mathbf{K}$.

- We first write the projection equation as:

$$\mathbf{x}_i = \mathbf{f}(\mathbf{p}_i; \mathbf{R}, \mathbf{t}, \mathbf{K})$$