CIS 5800

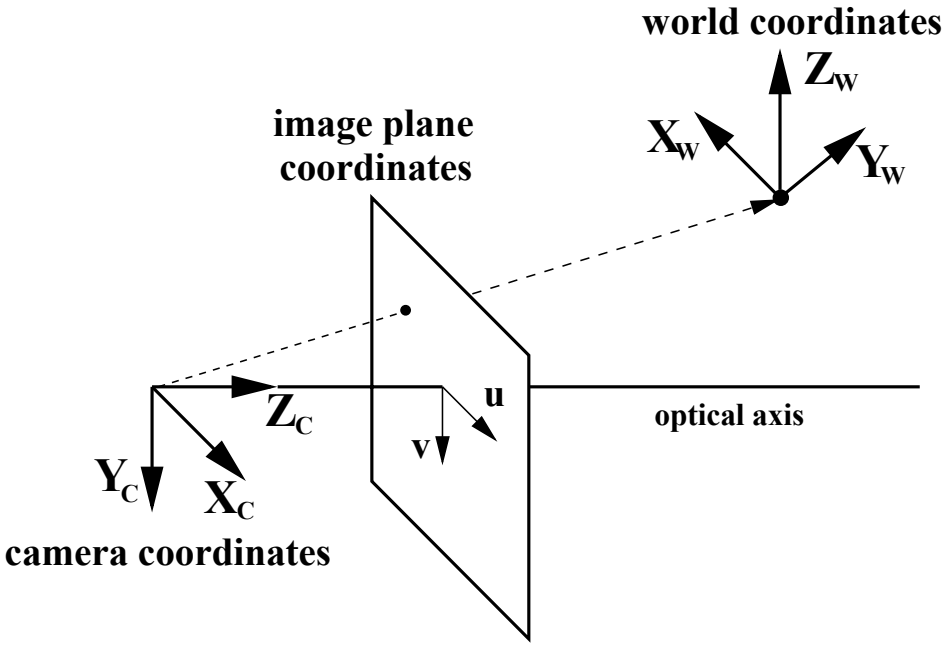
Machine Perception

Instructor: Lingjie Liu

Lec 10: Feb 26, 2025

Calibrating Camera Under More General Intrinsics

Recap: Perspective Projection Matrix



camera 3D coords to pixels Convert world to camera coordinates

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} [R_{3\times 3} \quad \boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

Generalizing intrinsics (1/2)

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & & u_0 \\ & f & v_0 \\ & & 1 \end{bmatrix} [R_{3\times 3} \quad \boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

1. If pixels are not square?

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & & u_0 \\ & s_y & v_0 \\ & & 1 \end{bmatrix} [R_{3\times 3} \quad \boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

What Does (Intrinsics) “Calibration” Do?

Estimates the *intrinsic parameters*

- f focal length (or s_x, s_y to allow non-square pixels)
- (u_o, v_o) image center

Method: High-Level Overview

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & & u_0 \\ & s_y & v_0 \\ & & 1 \end{bmatrix} [R_{3\times3} \quad \boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

Camera projection matrix $M_{3\times4}$

(Note: we've also called this P before)

Method: High-Level Overview

$M = P = projection\ matrix$

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Estimation of the full projection matrix is linear

but M has to be decomposed into $K, R, \boldsymbol{t}$ afterwards

("QR decomposition" of first 3 columns of M to get K, R. Then solve for **t**)

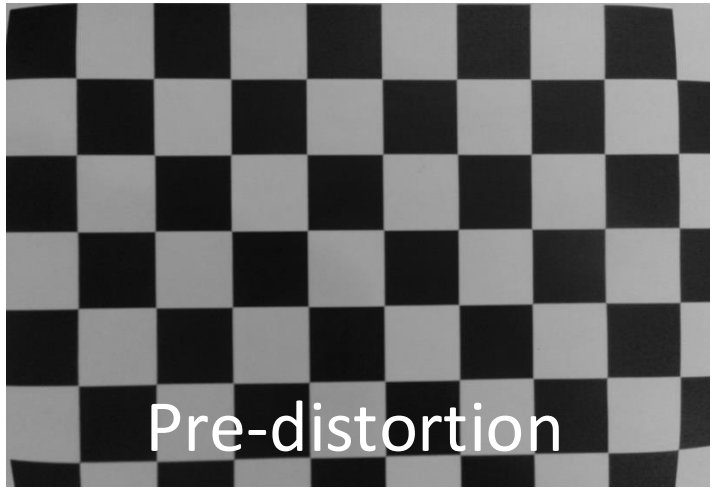
Generalizing intrinsics (2/2)

1. If pixels are not square?

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & & u_0 \\ & s_y & v_0 \\ & & 1 \end{bmatrix} [R_{3\times 3} \quad \boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = K[R|\boldsymbol{t}]\boldsymbol{X}_w$$

2. If “radial distortions”, then the intrinsics can no longer be represented as a linear operator any more

Parametrizing radial distortion in large field-of-view cameras



Then, correct for radial distortion:

$$u_{\text{pre-distortion}}(r) = u_{\text{post-distortion}}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots)$$
$$v_{\text{pre-distortion}}(r) = v_{\text{post-distortion}}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots)$$

where r is the distance from a (usually unknown) image center location (u_0, v_0).

Can choose the degree of the radial distortion to calibrate for. More => more accurate, but requires more images to fit well.

What Does (Intrinsics) “Calibration” Do?

Estimates the *intrinsic parameters*

- f focal length (or s_x, s_y to allow non-square pixels)
- (u_o, v_o) image center
- $k_1, k_2, \ldots$ radial distortion parameters

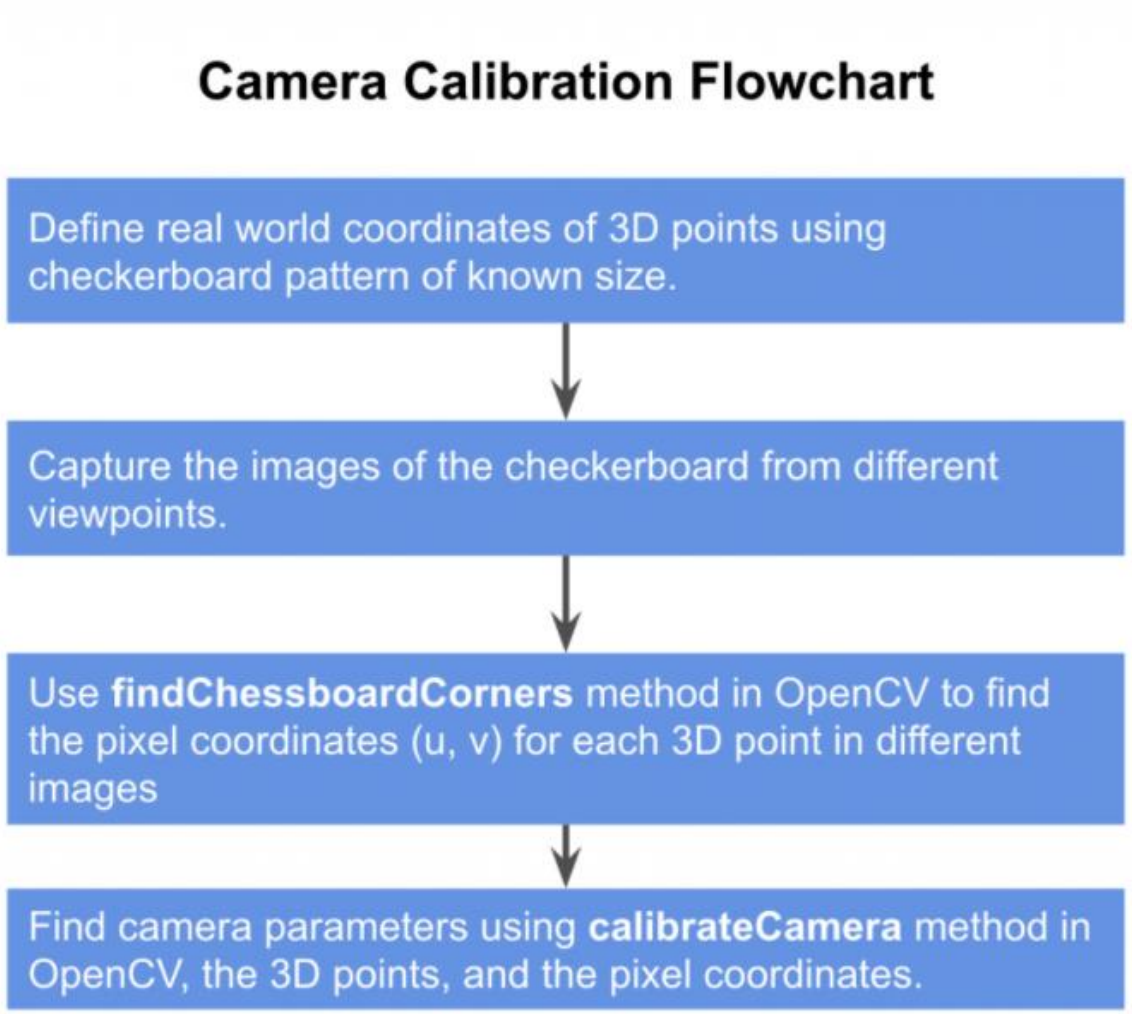
Won’t see how to do this in mathematical detail.

Colab worksheet (Camera Calibration with OpenCV)!

- https://colab.research.google.com/drive/1Vf6pBFo84OKR0E3j3t3HJjEN71o6k0sk?authuser=0

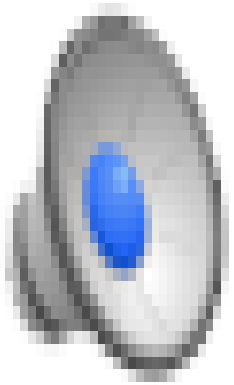
- (Optional) Also see OpenCV Tutorial on Calibration.

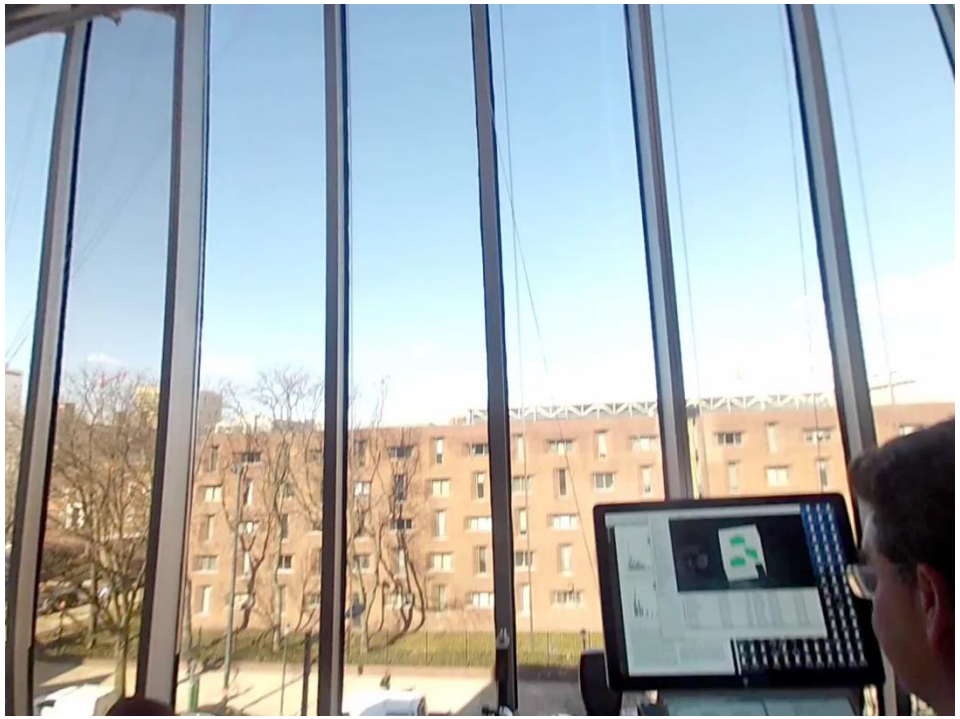
https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html



https://learnopencv.com/camera-calibration-using-opencv/

Output: Undistorted images and video!





And also …

Credit: Kostas Daniilidis