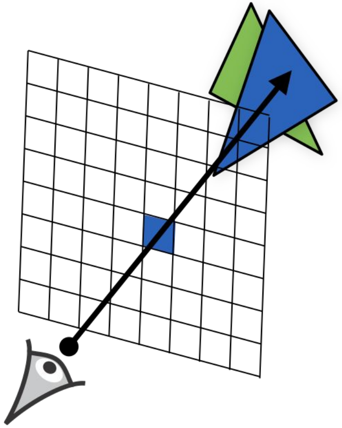
Camera Models

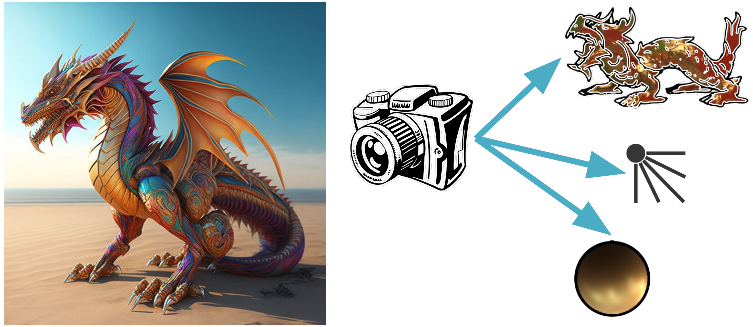
Dr Fangcheng Zhong

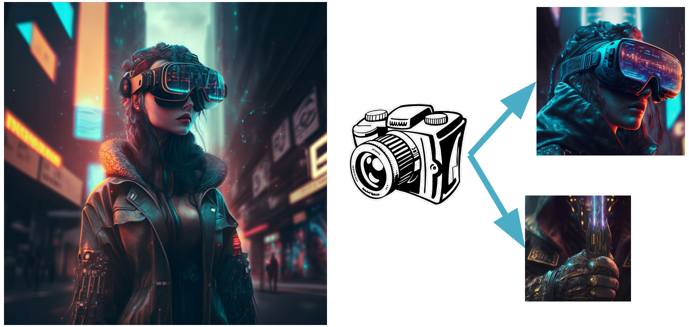
UNIVERSITY OF CAMBRIDGE

Camera Models

- Describe the mathematical relationship between the coordinates of a point in **3D space** and the coordinates of its **projection** onto the image plane







Camera (eye) is the bridge between the real and virtual world

Outline

- Pinhole model
  - MVP matrices
  - intrinsic & extrinsic matrices
  - camera calibration
- Non-pinhole model
  - thin-lens equation
  - lens distortion
  - nonlinear calibration

Pinhole Model

In computer graphics, the MVP matrices describe such a relationship

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \mathbf{P}\,\mathbf{V}\,\mathbf{M} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

screen coordinates object local coordinates

Pinhole Model

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \mathbf{P}\,\mathbf{V} \begin{bmatrix} x_w \\ y_w \\ z_w \\ w_w \end{bmatrix}$$

screen coordinates world coordinates

Pinhole Model

3X3 rotation

translation

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/f \\ 0 & 0 & 1/f & 0 \end{bmatrix} \left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array} \right] \begin{bmatrix} x_w \\ y_w \\ z_w \\ w_w \end{bmatrix}$$

projection matrix

view matrix

focal length

Pinhole Model

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/f \\ 0 & 0 & 1/f & 0 \end{bmatrix} \left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array} \right] \begin{bmatrix} x_w \\ y_w \\ z_w \\ w_w \end{bmatrix}$$

projection matrix

view matrix

Referred to as **extrinsic matrix** in computer vision

Pinhole Model

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/f \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

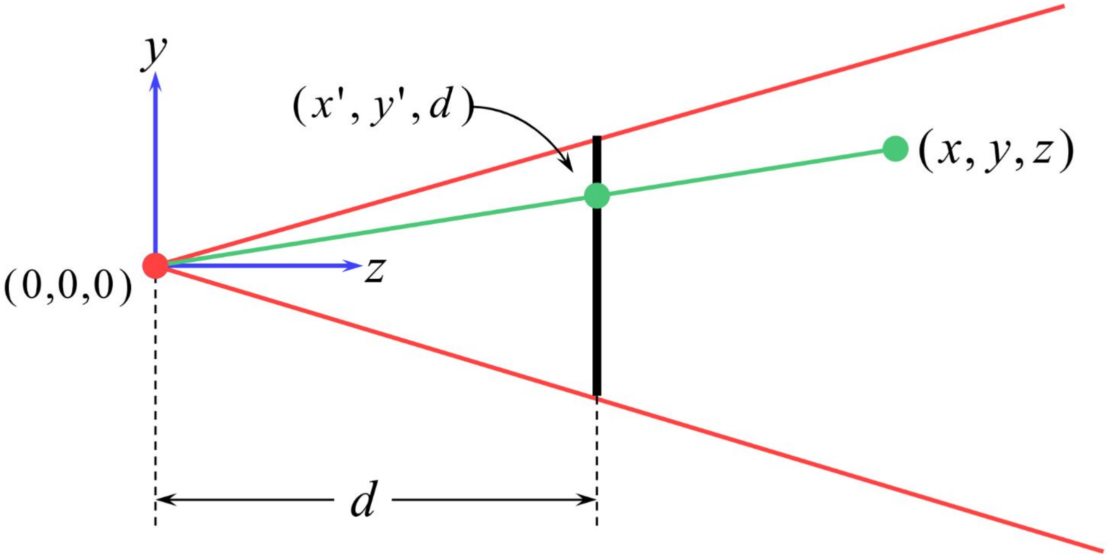
screen coordinates   projection matrix   camera coordinates

focal length

UNIVERSITY OF CAMBRIDGE

Pinhole Model

Recall Introduction to Graphics



$$x' = x\frac{d}{z}$$

$$y' = y\frac{d}{z}$$

Pinhole Model

equivalent relation

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/f \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ 1/f \\ z_c/f \end{bmatrix} \sim \begin{bmatrix} \frac{f}{z_c} x_c \\ \frac{f}{z_c} y_c \\ \frac{1}{z_c} \\ 1 \end{bmatrix}$$

In rasterisation, screen coordinates $[\frac{x_s}{w_s}, \frac{y_s}{w_s}]$ are always clipped between [-1, 1]

Focal length determines the field of view of the virtual camera. How?

UNIVERSITY OF CAMBRIDGE

Pinhole Model

The intrinsic matrix does not preserve the depth information

$$\begin{bmatrix} x_s \\ y_s \\ z_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/f \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ 1/f \\ z_c/f \end{bmatrix} \sim \begin{bmatrix} \frac{f}{z_c} x_c \\ \frac{f}{z_c} y_c \\ \frac{1}{z_c} \\ 1 \end{bmatrix}$$

In computer vision, the projection matrix is replaced by an **intrinsic matrix** which maps the camera coordinates to image/pixel coordinates

Pinhole Model

Convert the projection matrix into an intrinsic matrix

$$\begin{bmatrix} x_s \\ y_s \\ w_s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} \sim \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

Remove depth from screen coordinates

Pinhole Model

Convert the projection matrix into an intrinsic matrix

$$\Rightarrow \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix} \Rightarrow \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

focal length in pixel unit
(2D scaling)

shifting the pixel origin
(2D translation)

Pinhole Model

focal length principal point offset

$$\begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \begin{bmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

image coordinates intrinsic matrix camera coordinates

Pinhole Model

axis skew

$$\begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \begin{bmatrix} f_x & s & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ w_c \end{bmatrix}$$

image coordinates

intrinsic matrix

camera coordinates

Pinhole Model

$$\begin{bmatrix} x_i \\ y_i \\ w_i \end{bmatrix} = \begin{bmatrix} f_x & s & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left[\begin{array}{c|c} \mathbf{R}_{3\times 3} & \mathbf{t}_{3\times 1} \end{array} \right] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \underbrace{\mathbf{K}\,[\mathbf{R}|\mathbf{t}]}_{\mathbf{C}} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

intrinsic matrix
(5 free parameters)

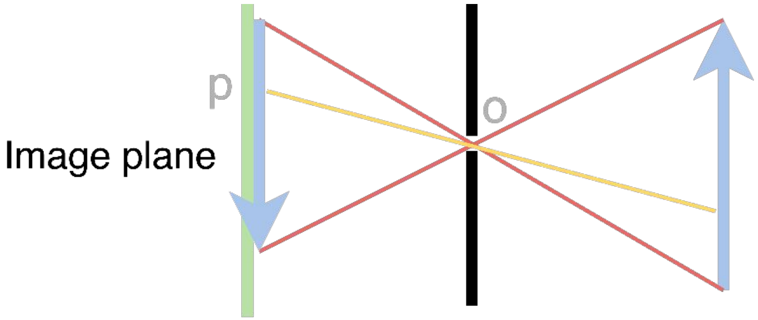
extrinsic matrix
(3+3 free parameters)

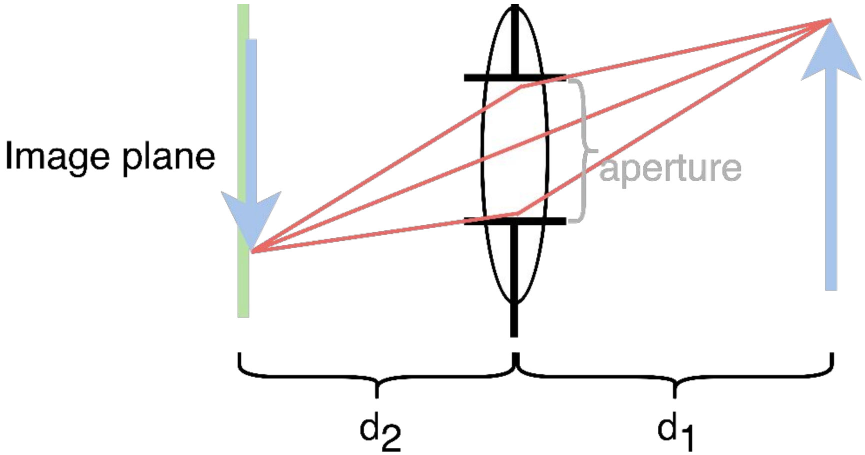
camera matrix
(3x4 shape)

Q: Why is it okay to fix the homogeneous division to 1? How come the extrinsic matrix does not need a scaling factor?

Thin-lens Model

Real cameras are not pinhole

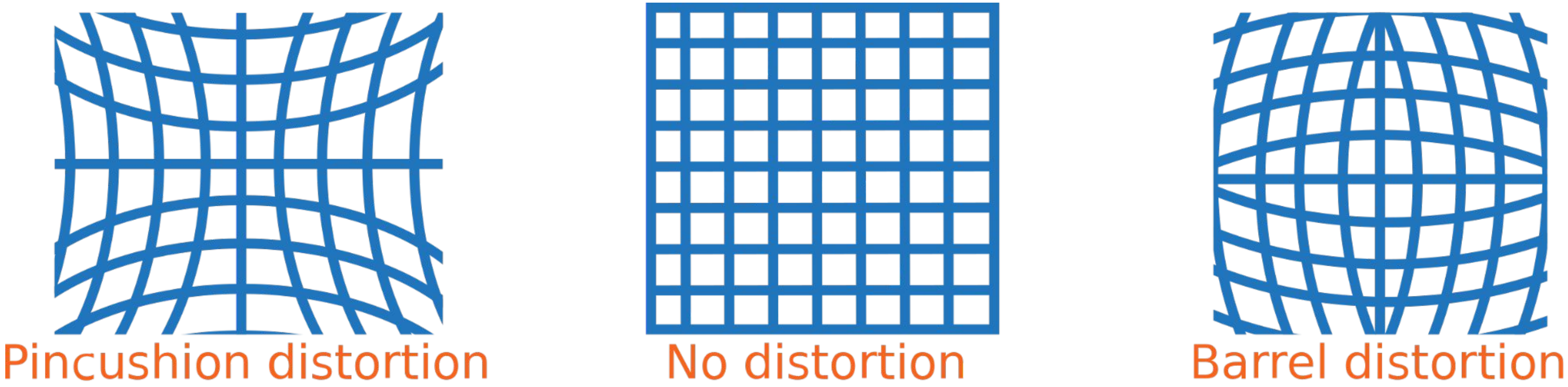




Thin-lens equation: $\frac{1}{d_1} + \frac{1}{d_2} = \frac{1}{f}$

Where is the camera origin of a thin-lens model?

Radial Distortion



Light rays bend at a different angle near the edges of the lens than those at the optical center

Radial Distortion

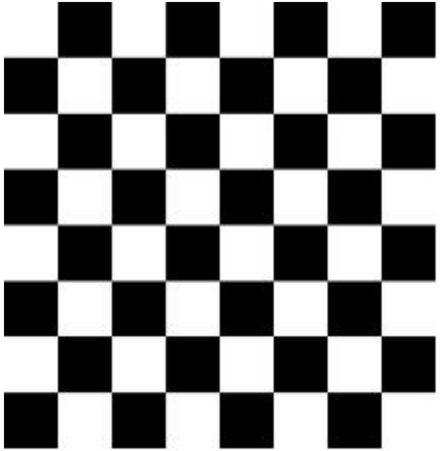
$$x_{\text{distorted}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{\text{distorted}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

x, y — undistorted pixel locations in normalized image coordinates (dimensionless), calculated from pixel coordinates by translating to the optical center and dividing by the focal length in pixels

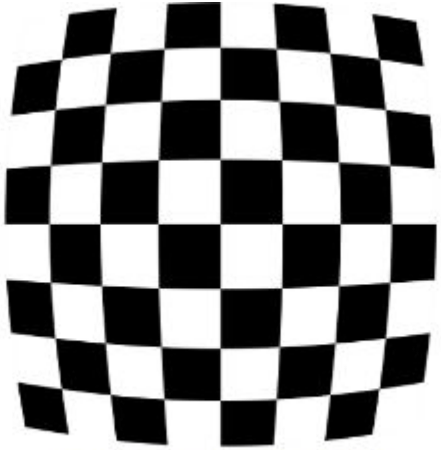
k1, k2, k3 — radial distortion coefficients of the lens

r^2 = x^2 + y^2

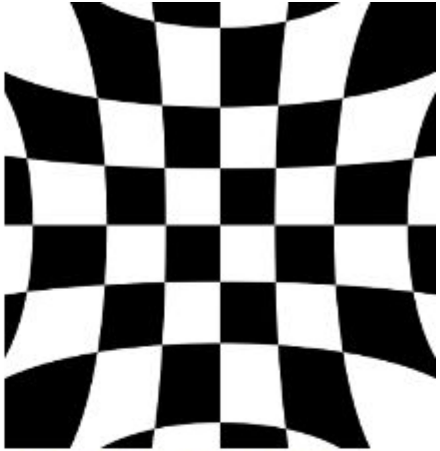
Radial Distortion



No distortion

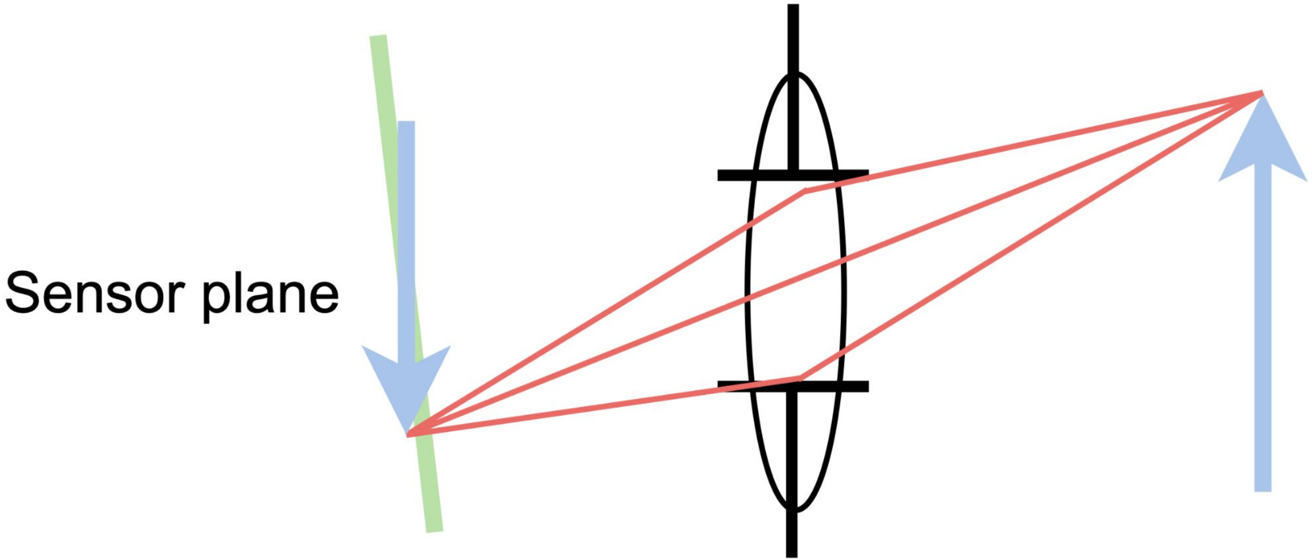


Barrel distortion



Pincushion distortion

Tangential Distortion



Occurs when the lens and the image plane are not parallel

Tangential Distortion

$$x_{\text{distorted}} = x + 2p_1xy + p_2(r^2 + 2x^2)$$
$$y_{\text{distorted}} = y + p_1(r^2 + 2y^2) + 2p_2xy$$

x, y — undistorted pixel locations in normalized image coordinates (dimensionless), calculated from pixel coordinates by translating to the optical center and dividing by the focal length in pixels

p1, p2 — tangential distortion coefficients

r^2 = x^2 + y^2

Camera Resectioning

- The process of estimating the camera parameters (e.g. extrinsic, intrinsic, distortion) given a camera model, i.e. geometric camera calibration

Extrinsic Calibration

- Equivalent to camera pose estimation,
  i.e. camera pose and extrinsics can be mutually converted

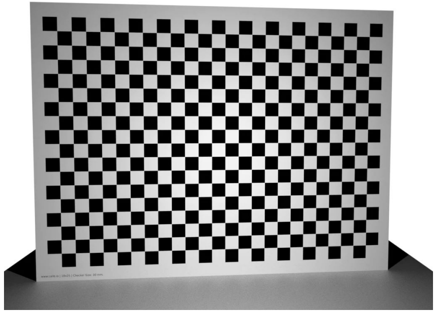
$$\mathbf{RQ} = \mathbf{I} \quad \Rightarrow \quad \mathbf{Q} = \mathbf{R}^T$$

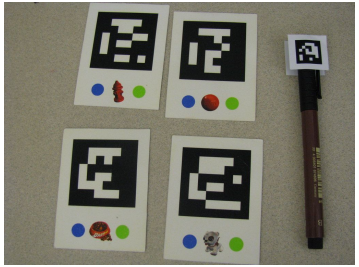
$$[\mathbf{R}|\mathbf{t}]\,\mathbf{c} = \mathbf{Rc} + \mathbf{t} = \mathbf{0} \quad \Rightarrow \quad \mathbf{c} = -\mathbf{R}^T\mathbf{t}$$

Q, **c** — camera pose (orientation **Q** + center **c**)
R, **t** — camera extrinsics (rotation **R** + translation **t**)

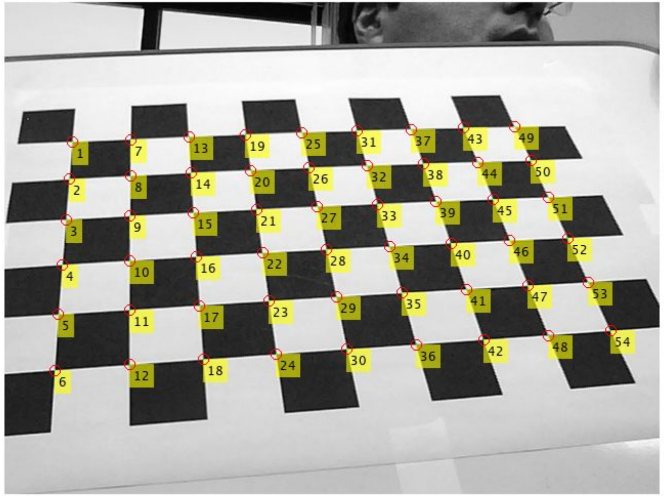
Extrinsic Calibration

- The Perspective-n-Point (PnP) problem: estimating the pose of a calibrated camera, i.e. known intrinsic and distortion, given a set of n 3D points in the world and their corresponding 2D projections in the image
- Correspondence established with known calibration patterns

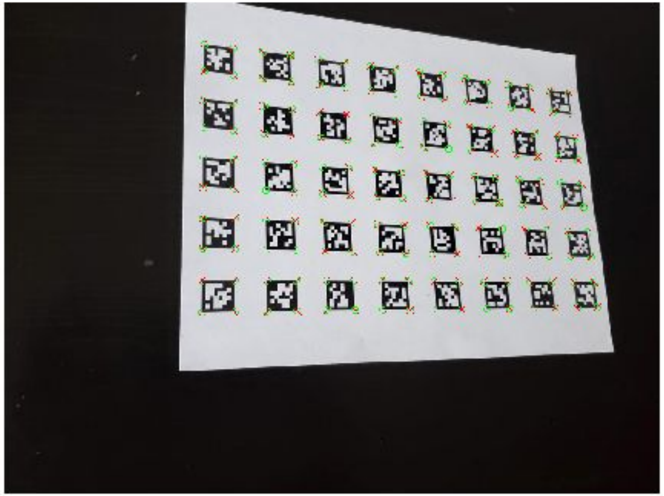




Calibration Patterns



Checkerboard



AprilTags

- similar to QR codes
- encode less data
- faster for real-time applications

Intrinsic Calibration

Calibrating both the camera intrinsics and extrinsics

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim w \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\mathbf{K}\left[\mathbf{R}|\mathbf{t}\right]}_{\mathbf{C}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

image coordinates

camera matrix

world coordinates

Similar idea: solve for C given a set of n 3D points (x_i, y_i, z_i) in the world and their corresponding 2D projections (u_i, v_i) in the image

Intrinsic Calibration

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim w \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

$$u_i = \frac{c_{11}\, x_i + c_{12}\, y_i + c_{13}\, z_i + c_{14}}{c_{31}\, x_i + c_{32}\, y_i + c_{33}\, z_i + c_{34}}$$

$$v_i = \frac{c_{21}\, x_i + c_{22}\, y_i + c_{23}\, z_i + c_{24}}{c_{31}\, x_i + c_{32}\, y_i + c_{33}\, z_i + c_{34}}$$

Intrinsic Calibration

$$\underbrace{\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 & -u_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -v_1x_1 & -v_1y_1 & -v_1z_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & z_n & n & 0 & 0 & 0 & 0 & -u_nx_n & -u_ny_n & -u_nz_n & -u_n \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & n & -v_nx_n & -v_ny_n & -v_nz_n & -v_n \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \\ c_{21} \\ c_{22} \\ c_{23} \\ c_{24} \\ c_{31} \\ c_{32} \\ c_{33} \\ c_{34} \end{bmatrix}}_{\mathbf{c}} = \mathbf{0}$$

- The solution vector **c** holds for an arbitrary scale
- Direct linear transformation (DLT)
  - — find **c** that minimises ||**Ac**|| subject to a unit vector constraint ||**c**||=1
  - — solution **c** = eigenvector of $\mathbf{A}^T\mathbf{A}$ with the smallest eigenvalue

Direct Linear Transformation

Let $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$,

$$
\begin{aligned}
& \arg\min_{\mathbf{c}} ||\mathbf{A}\mathbf{c}|| \quad \text{s.t.} \quad ||\mathbf{c}|| = 1 \\
\iff \quad & \arg\min_{\mathbf{c}} ||\mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{c}|| \quad \text{s.t.} \quad ||\mathbf{c}|| = 1 \\
\iff \quad & \arg\min_{\mathbf{c}} ||\mathbf{D}\mathbf{V}^T\mathbf{c}|| \quad \text{s.t.} \quad ||\mathbf{V}^T\mathbf{c}|| = 1 \\
\iff \quad & \arg\min_{\mathbf{m}} ||\mathbf{D}\mathbf{m}|| \quad \text{s.t.} \quad ||\mathbf{m}|| = 1,\ \mathbf{m} = \mathbf{V}^T\mathbf{c}
\end{aligned}
$$

$||\mathbf{D}\mathbf{m}||$ is minimum when $||\mathbf{m}|| = (0, ..., 0, 1) \ \Rightarrow \ \mathbf{c} = \mathbf{V}\mathbf{m}$, the last column of $\mathbf{V}$
i.e. the eigenvector of $\mathbf{A}^T\mathbf{A}$ with the smallest eigenvalue

Direct Linear Transformation

- ✅
  - Simple to formulate and compute
  - Minimise the algebraic error
- ❌
  - Not directly outputting the camera parameters (can be extracted by an RQ decomposition)
  - Not modelling distortions
  - Not minimising the geometric error

Nonlinear Calibration

- Minimising the geometric error
- Simultaneously estimate all camera parameters (extrinsic, intrinsic, and distortion) using nonlinear least-squares minimisation (e.g. Levenberg–Marquardt algorithm)

$$\arg\min_{\beta} \sum_{i} \| (\mathbf{C}_{\beta}(\mathbf{p}_i) - \mathbf{x}_i) \|^2$$

- Use the DLT solution as the initial estimate of the intrinsics and extrinsics and zero as the initial estimate of the distortion coefficients

Nonlinear Calibration

- In most modern XR devices, the intrinsic and distortion parameters can be provided by the manufacturer (reduced to a PnP problem)

Levenberg–Marquardt (LM) Algorithm

- A trust-region approach to solve the nonlinear least squares problem

$$f(\beta) = \sum_{i=1}^{m} r_i^2(\beta)$$

$$\arg\min_{\beta} \; f(\beta)$$

Levenberg–Marquardt (LM) Algorithm

Gradient descent: $$\beta_{n+1} = \beta_n - \lambda \nabla f(\beta_n)$$

Newton's method: $$\beta_{n+1} = \beta_n - \underbrace{\mathbf{H} f(\beta_n)}_{\text{Hessian}}{}^{-1} \nabla f(\beta_n)$$

LM algorithm: $$\beta_{n+1} = \beta_n - \left(\mathbf{H} f(\beta_n) + \lambda\, \mathbf{I}\right)^{-1} \nabla f(\beta_n)$$

$$\mathbf{H} f(\beta) \approx \left(\mathbf{J}\mathbf{r}(\beta)\right)^T \underbrace{\mathbf{J}\mathbf{r}(\beta)}_{\text{Jacobian}}$$

$$\mathbf{r}(\beta) = \underbrace{\left(r_1(\beta), r_2(\beta), \ldots, r_m(\beta)\right)^T}_{\text{residual vector}}$$

Both Gauss-Newton and LM use this approximation for the nonlinear least square problem

UNIVERSITY OF CAMBRIDGE