Principles of Robot Autonomy I

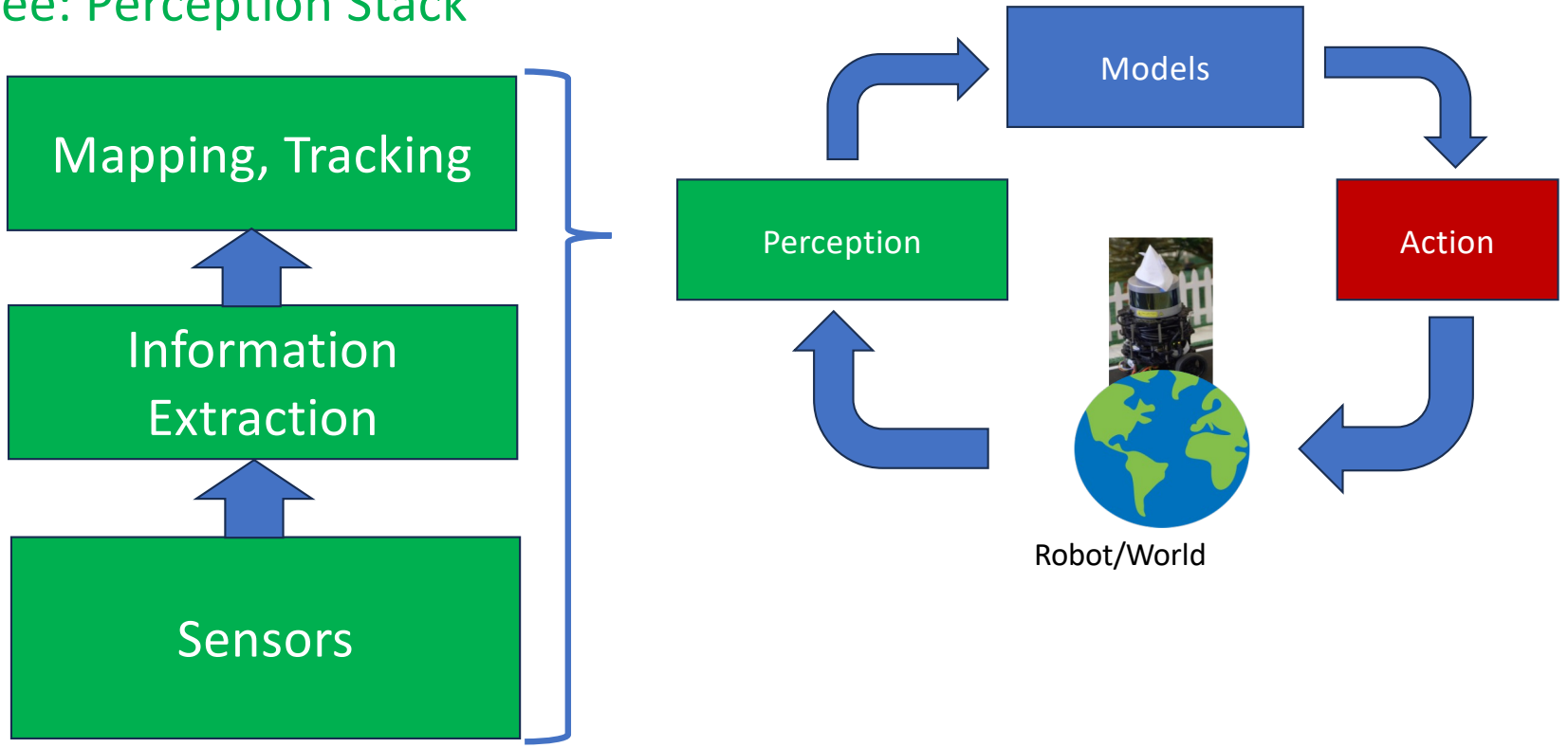
Camera models, camera calibration, PnP

Logistics

- Homework 2: due **Friday, Oct 17, at 5pm**.
- Homework 3: out Thurs, Oct 16, due Tues, Oct 28
- Midterm window: Wed, Oct 29, 5pm – Fri, Oct 31
  - Take home, 48 hour window
  - Check out exam on gradescope
  - You will have personal 5 hour time slot
  - Open notes, book, HW solutions
  - No internet, no GenAI, no working with others
- Lecture 8:
  - Point cloud alignment, ICP
  - Pinhole camera model, camera calibration

Robot Perception

See: Perception Stack

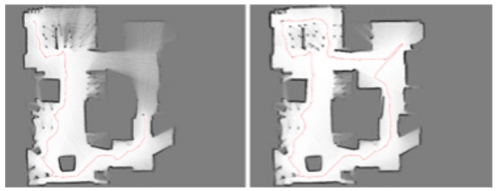


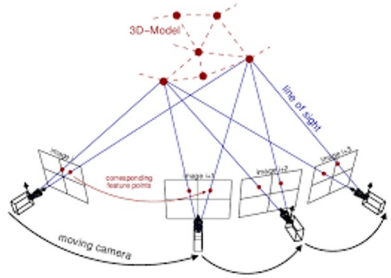
Perception Stack: Computer vision, filtering, SLAM

Use sensor data to update the models

Localization, Mapping, Tracking:

- EKF/Monte Carlo localization
- Occupancy grid mapping
- Pose graph optimization
- Tracking (EKF and Particle Filter)
- **AA273: Filtering (Schwager)**
- **AA275: Navigation (Gao)**



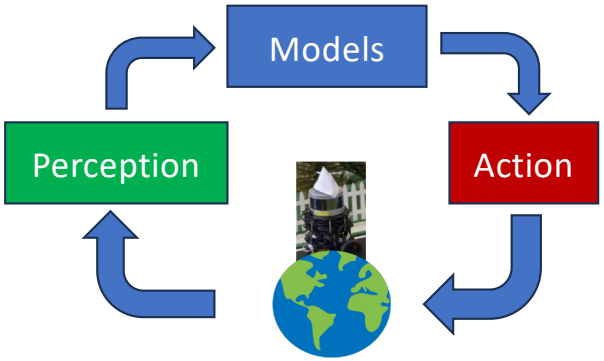


Information extraction

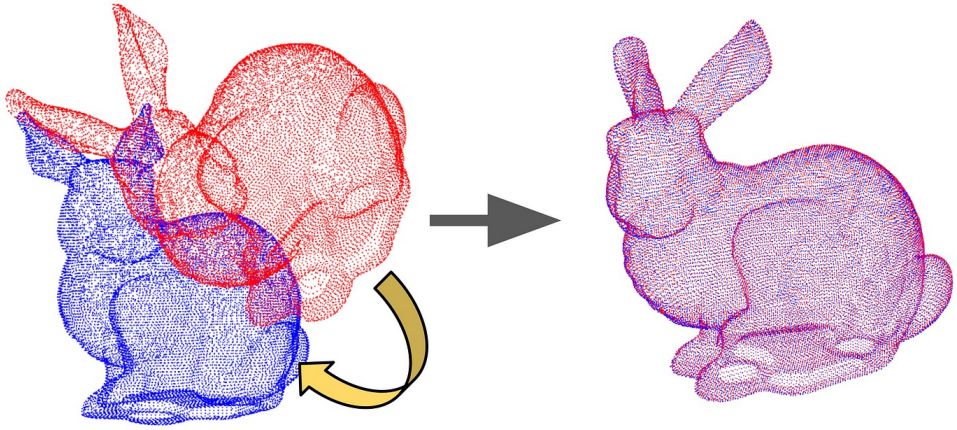
- Computer vision: features, correspondences, Structure from Motion (SfM), depth
- Lidar scan matching, ICP
- **CS231A: Comp Vision**

Sensors:

- RGB Cameras, RGB-D/stereo cameras, Lidar
- IMU, GPS, wheel encoders



Recall from last time: Iterative Closest Point (ICP) for point cloud alignment



- Obtain the relative pose: The pose transform required to move P_A to align with P_B

$$(\tau_{BA}, R_{BA})$$

Translation vector

Rotation matrix

Basics and notation:

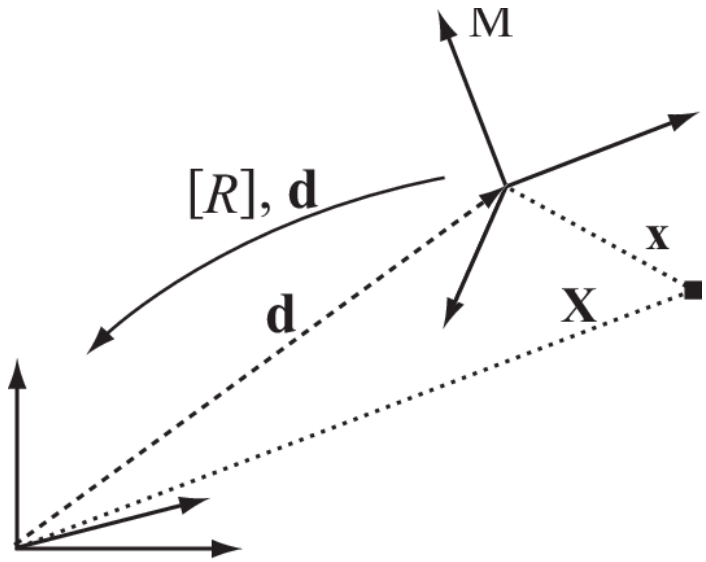
- Rigid body transforms, SE(2), SE(3)

$$(\tau_{BA}, R_{BA}) \in \mathrm{SE}(2) \quad \text{or} \quad \mathrm{SE}(3)$$

$$P_B = R_{AB} P_A + \tau_{BA}$$

rotate

translate



Alignment with Known Correspondences

- Find (τ_{BA}, R_{BA})

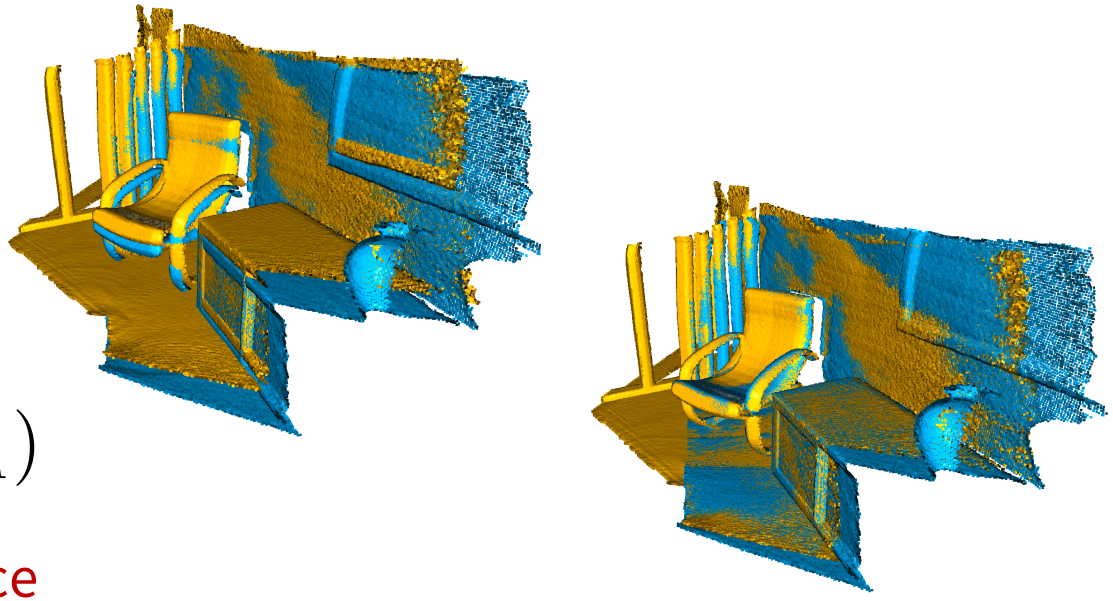
$$P_B = R_{AB} P_A + \tau_{BA}$$

$$\min_{(\tau_{BA}, R_{BA})} \|P_B - (R_{BA} P_A + \tau_{BA})\|_{\mathcal{F}}^2$$
$$\text{s.t.} \quad R_{BA}^T R_{BA} = I$$

- Known solution based on SVD, sometimes called Wahba's problem

ICP: iterate with closest point heuristic

- For each point in P_A, p_i^A match to closest point p_j^B from P_B

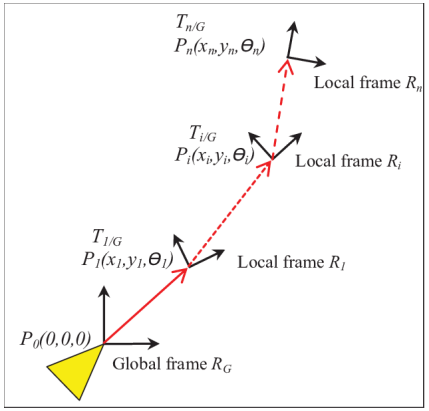


- Solve for (τ_{BA}, R_{BA})
- Repeat until convergence

Images: Open3D

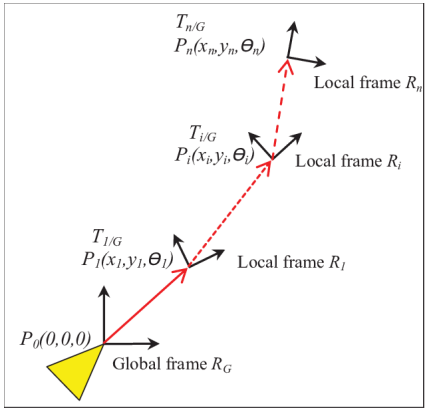
Why? Simultaneous Localization and Mapping (SLAM)

- Lidar odometry: Gives relative poses for robot between successive Lidar scans.
- Pose graph: These Graph where nodes are unknown global poses, and edges are relative pose measurements (e.g., from ICP)
- SLAM front end: constructing the pose graph from raw sensor measurements
- SLAM back end: pose graph optimization (PGO) to find self-consistent set of global robot poses and corresponding global map



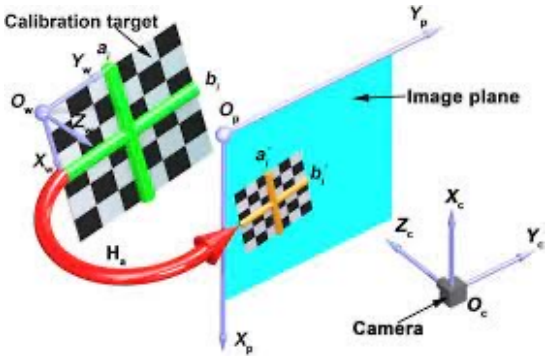
Computer Vision

- We can also build a pose graph from an RGB camera!
- Or from a stereo camera
- Or from an RGB-D camera
- First we need a mathematical model of the camera

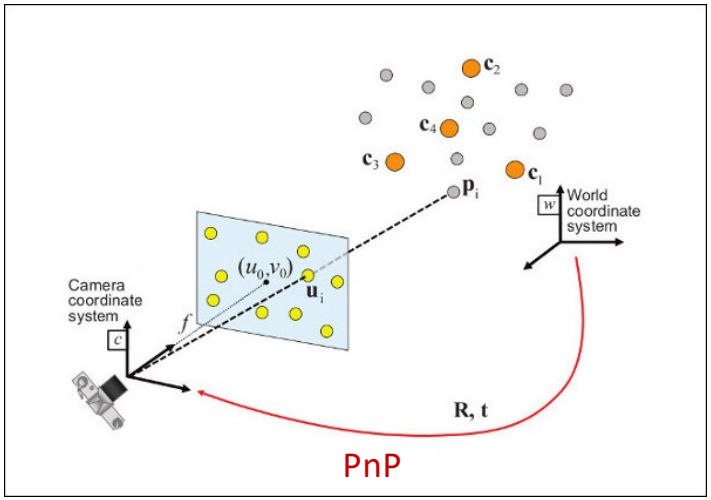


Classic Vision Problems: Geometric

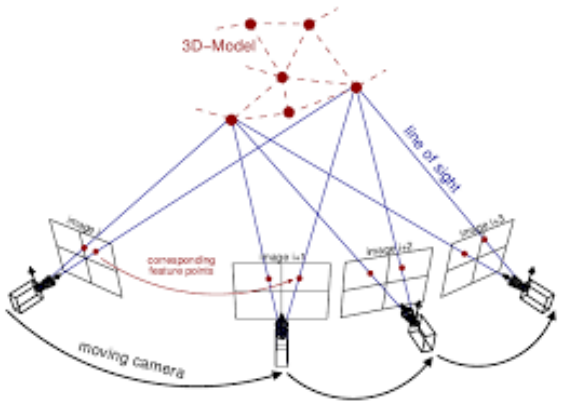
- Camera calibration: Find calibration matrix (intrinsic parameters) and camera pose (extrinsic parameters)
- N Point Perspective (PnP): Match 2D points in the image with 3D points in a pre-existing 3D model
- Structure from Motion (SfM, multi-view geometry): Find 3D locations of points in scene based on pairs or collections of images from different perspectives. Gives a point cloud and camera poses.



Camera calibration



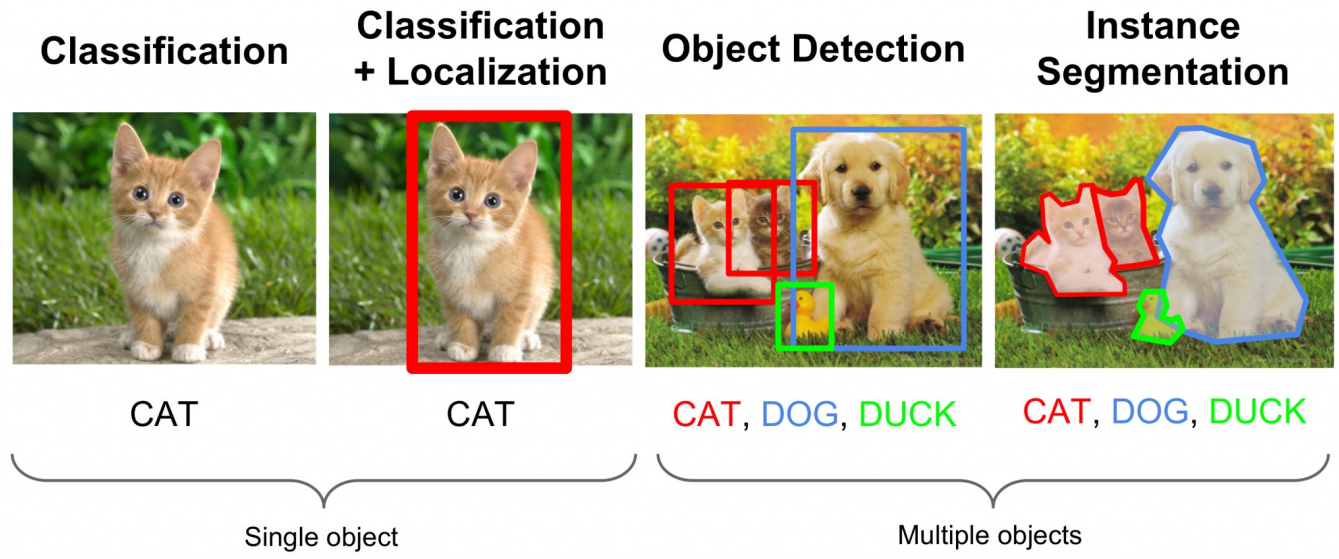
PnP



SfM

Classic Vision Problems: Semantic

- Classification: What's in the image?
- Detection: Draw bounding boxes around objects with labels
- Segmentation: Identify pixels (masks) belonging to object classes



Camera models and camera calibration

- Aim
  - Understand the pinhole camera model and projective geometry
  - Learn about camera calibration and the PnP problem

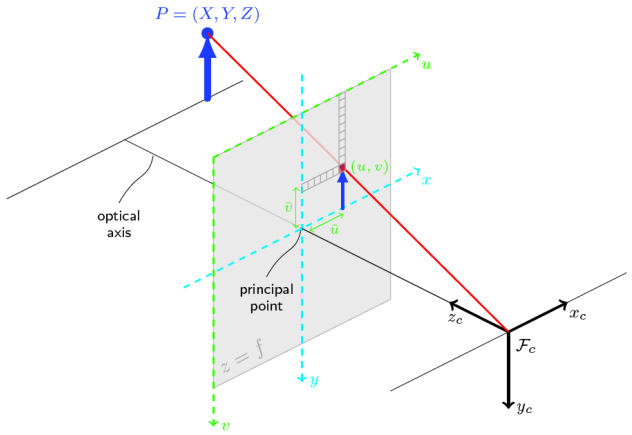
- Readings
  - SNS: 4.2.3
  - D. A. Forsyth and J. Ponce [FP]. Computer Vision: A Modern Approach (2nd Edition). Prentice Hall, 2011. Chapter 1.
  - R. Hartley and A. Zisserman [HZ]. Multiple View Geometry in Computer Vision. Academic Press, 2002. Chapter 6.1.
  - Z. Zhang. A Flexible New Technique for Camera Calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000.

Pinhole camera model

- Produces a matrix of RGB color vectors

$$I = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \ddots & & \vdots \\ c_{M1} & \cdots & & c_{MN} \end{bmatrix}$$

$$c_{uv} = (r_{uv}, g_{uv}, b_{uv})$$



Calibration matrix:

$$K = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

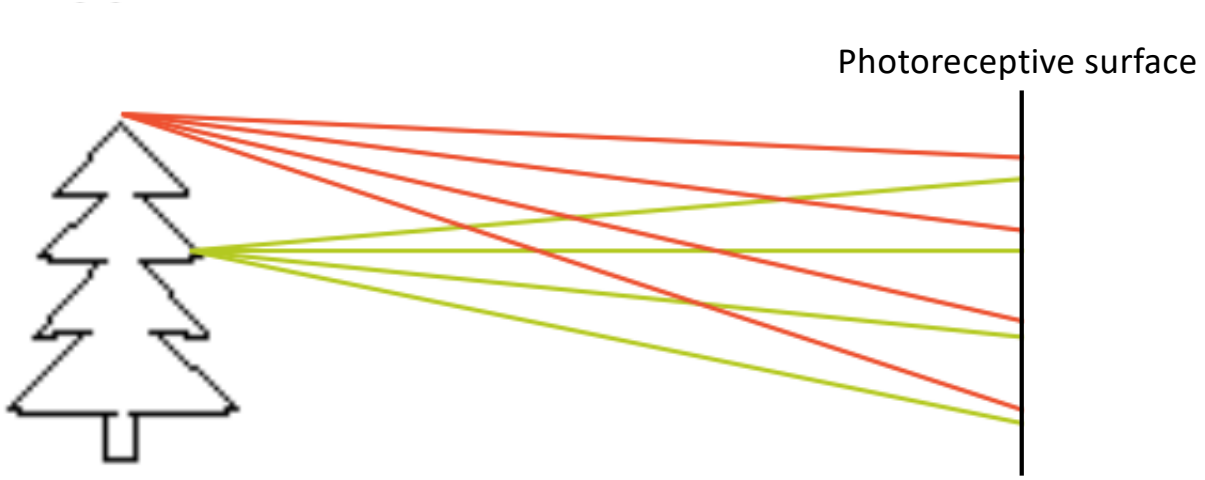
Pinhole camera model:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$$

- Projection mapping from the "thing" in the world at location (p_x, p_y, p_z) to the pixel coordinates (u, v) that have color c_{uv} in the image.

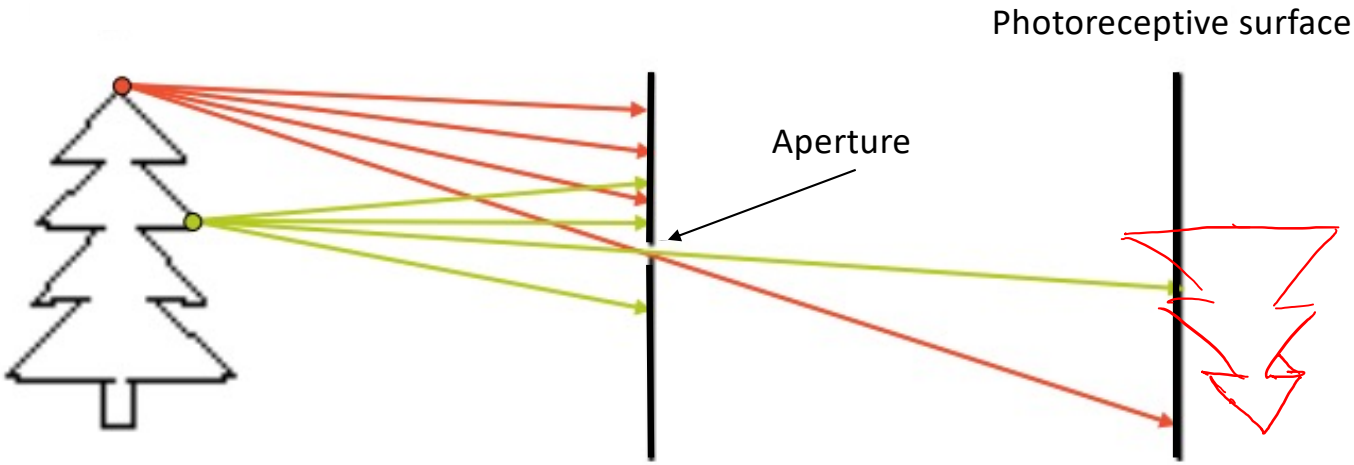
How to capture an image of the world?

- Light is reflected by the object and scattered in all directions
- If we simply add a photoreceptive surface, the captured image will be extremely blurred



Pinhole camera

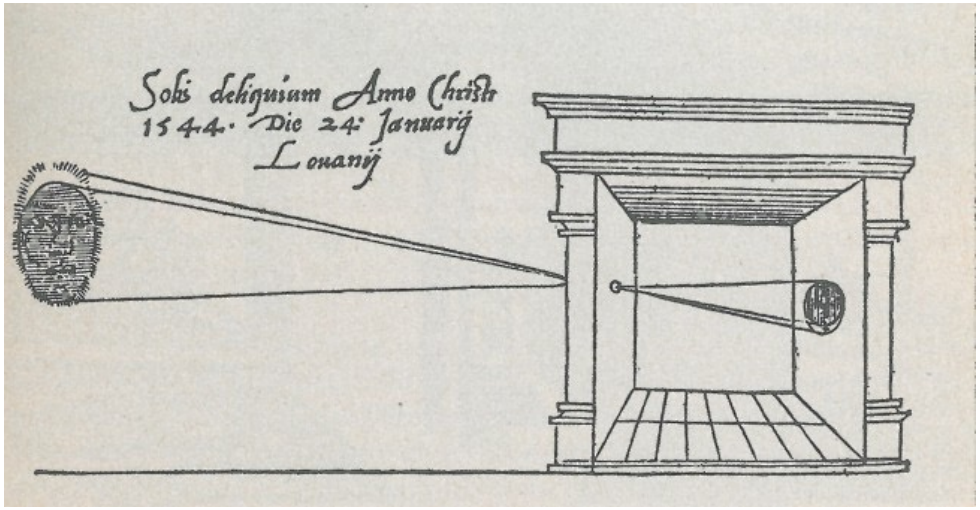
- Idea: add a barrier to block off most of the rays



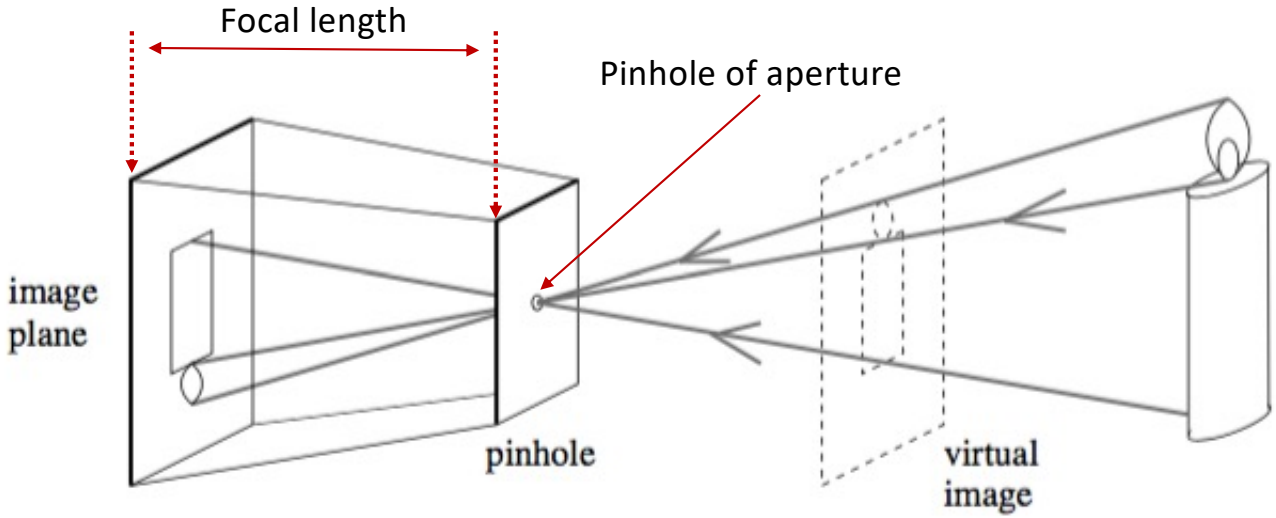
- Pinhole camera: a camera *without a lens* but with a tiny aperture, a *pinhole*

A long history

- Very old idea (several thousands of years BC)
- First clear description from Leonardo Da Vinci (1502)
- Oldest known published drawing of a camera obscura by Gemma Frisius (1544)



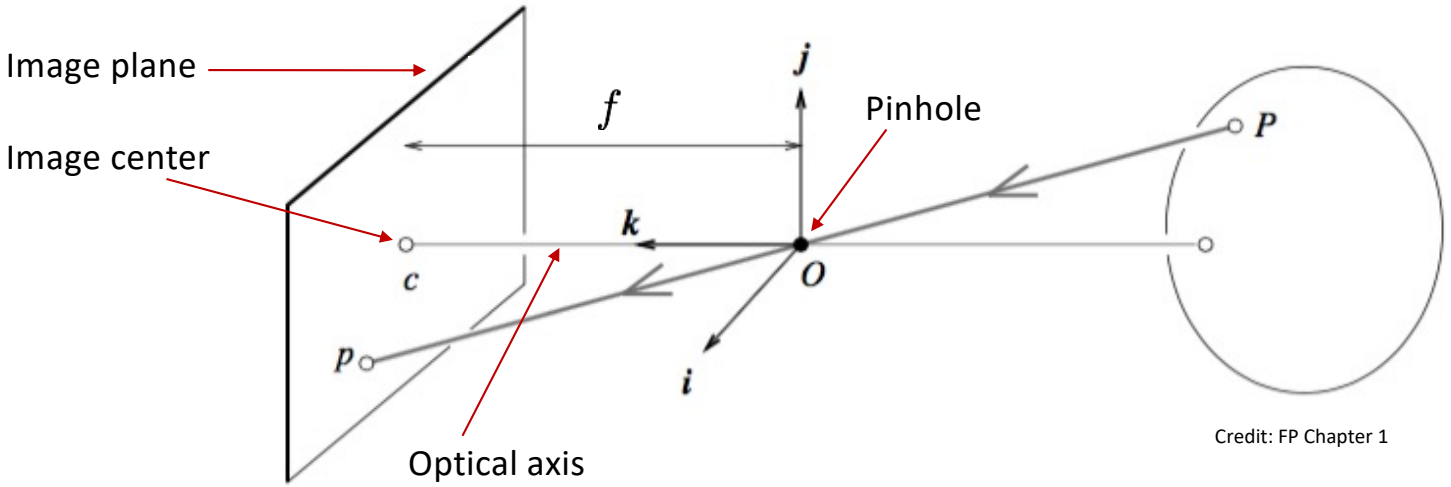
Pinhole camera



Credit: FP Chapter 1

- Perspective projection creates inverted images
- Sometimes it is convenient to consider a *virtual image* associated with a plane lying in front of the pinhole
- Virtual image not inverted but otherwise equivalent to the actual one

Pinhole perspective



Credit: FP Chapter 1

$P = (X, Y, Z)$

Perspective

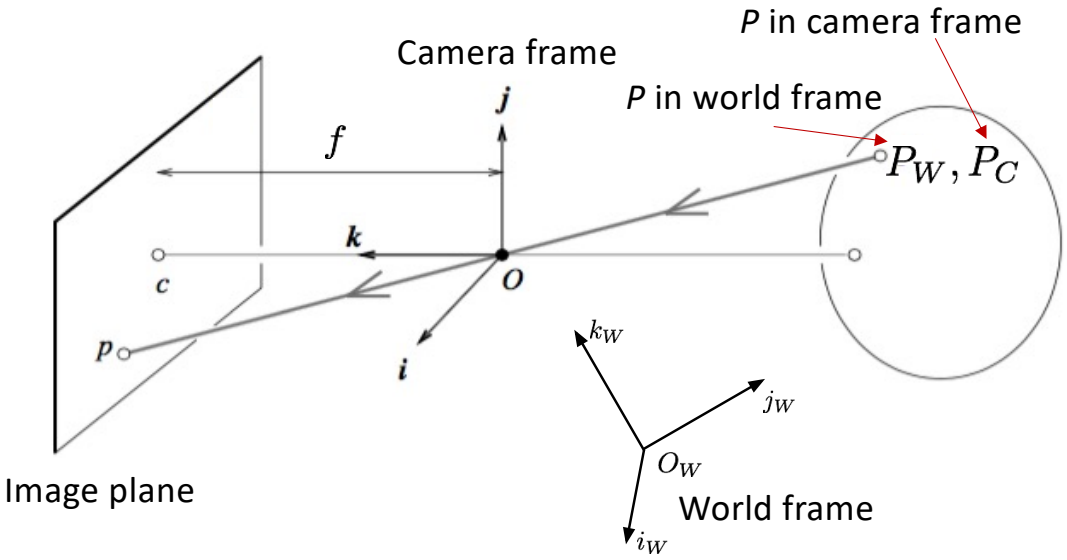
$p = (x, y, z)$

- Since *P*, *O*, and *p* are collinear: $\overline{Op} = \lambda \overline{OP}$ for some $\lambda \in R$
- Also, *z=f*, hence

$$\begin{cases} x = \lambda X \\ y = \lambda Y \\ z = \lambda Z \end{cases} \Leftrightarrow \lambda = \frac{x}{X} = \frac{y}{Y} = \frac{z}{Z} \Rightarrow \begin{cases} x = f\frac{X}{Z} \\ y = f\frac{Y}{Z} \end{cases}$$

Perspective projection

- Goal: find how world points map in the camera image
- Assumption: pinhole camera model (*all results also hold under thin lens model, assuming camera is focused at* ∞)



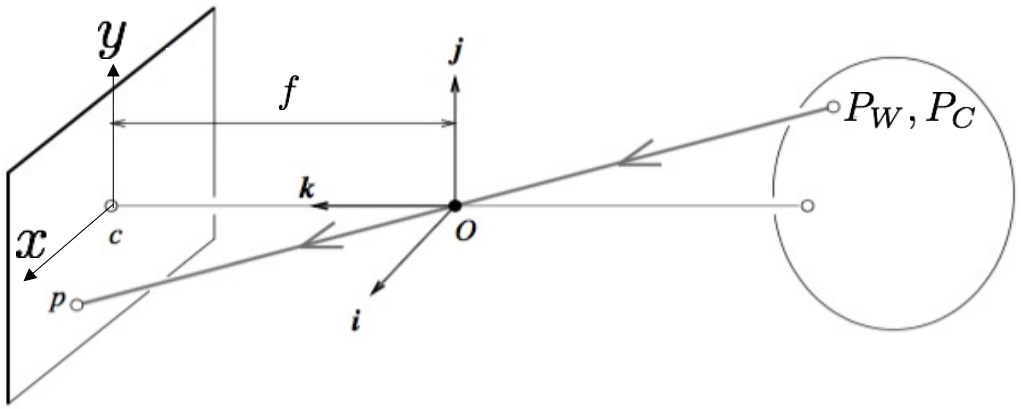
Roadmap:

1. Map P_c into *p* (image plane)
2. Map *p* into (u,v) (pixel coordinates)
3. Transform P_w into P_c

Step 1

- Task: Map $P_c = (X_C, Y_C, Z_C)$ into $p = (x, y)$ (image plane)
- From before

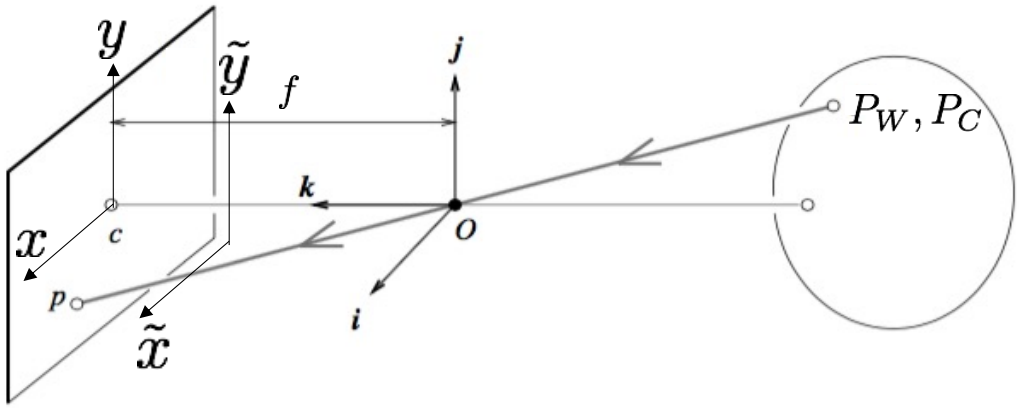
$$\begin{cases} x = f\frac{X_C}{Z_C} \\ y = f\frac{Y_C}{Z_C} \end{cases}$$

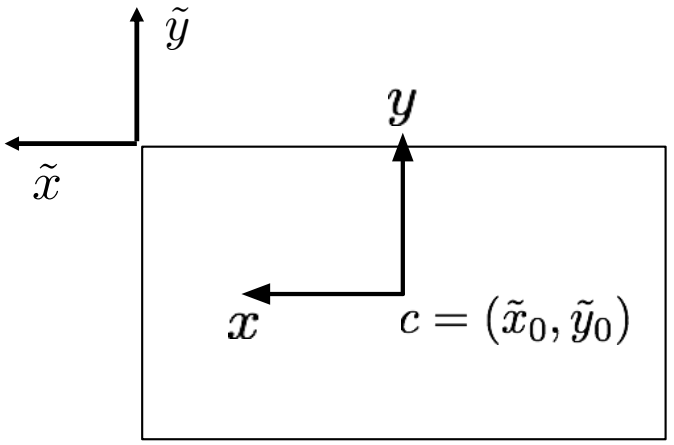


Step 2.a

- Actual origin of the camera coordinate system is usually at a corner (e.g., top left, bottom left)

$$\tilde{x} = f\,\frac{X_C}{Z_C} + \tilde{x}_0, \qquad \tilde{y} = f\,\frac{Y_C}{Z_C} + \tilde{y}_0,$$





Step 2.b

- Task: convert from image coordinates $(\tilde{x}, \tilde{y})$ to pixel coordinates (u, v)
- Let k_x and k_y be the number of pixels per unit distance in image coordinates in the *x* and *y* directions, respectively

$$u = k_x \tilde{x} = \overbrace{k_x f}^{\alpha} \frac{X_C}{Z_C} + \overbrace{k_x \tilde{x}_0}^{u_0}$$

$$v = k_y \tilde{y} = \underbrace{k_y f}_{\beta} \frac{Y_C}{Z_C} + \underbrace{k_y \tilde{y}_0}_{v_0}$$

$$\Rightarrow \quad \boxed{\begin{aligned} u &= \alpha \frac{X_C}{Z_C} + u_0 \\ v &= \beta \frac{Y_C}{Z_C} + v_0 \end{aligned}}$$

Nonlinear transformation

Homogeneous coordinates

- Goal: represent the transformation as a linear mapping
- Key idea: introduce homogeneous coordinates

Inhomogenous -> homogeneous

$$\begin{bmatrix} x \\ y \end{bmatrix} \Longrightarrow \lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Longrightarrow \lambda \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Homogenous -> inhomogeneous

$$\begin{bmatrix} x\lambda \\ y\lambda \\ \lambda \end{bmatrix} \Longrightarrow \begin{bmatrix} x \\ y \end{bmatrix} \qquad \begin{bmatrix} x\lambda \\ y\lambda \\ z\lambda \\ \lambda \end{bmatrix} \Longrightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Perspective projection in homogeneous coordinates

- Projection can be equivalently written in homogeneous coordinates

$$\begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha X_c + u_0 Z_c \\ \beta Y_c + v_0 Z_c \\ Z_c \end{pmatrix}$$

K

Camera matrix/
Matrix of intrinsic parameters

P_c in homogeneous coordinates

Homogeneous pixel coordinates

- In homogeneous coordinates, the mapping is linear:

$$p^h = [K \quad 0_{3\times 1}] P_C^h$$

Point *p* in homogeneous pixel coordinates

Point P_c in homogeneous camera coordinates

Skewness

- In some (rare) cases

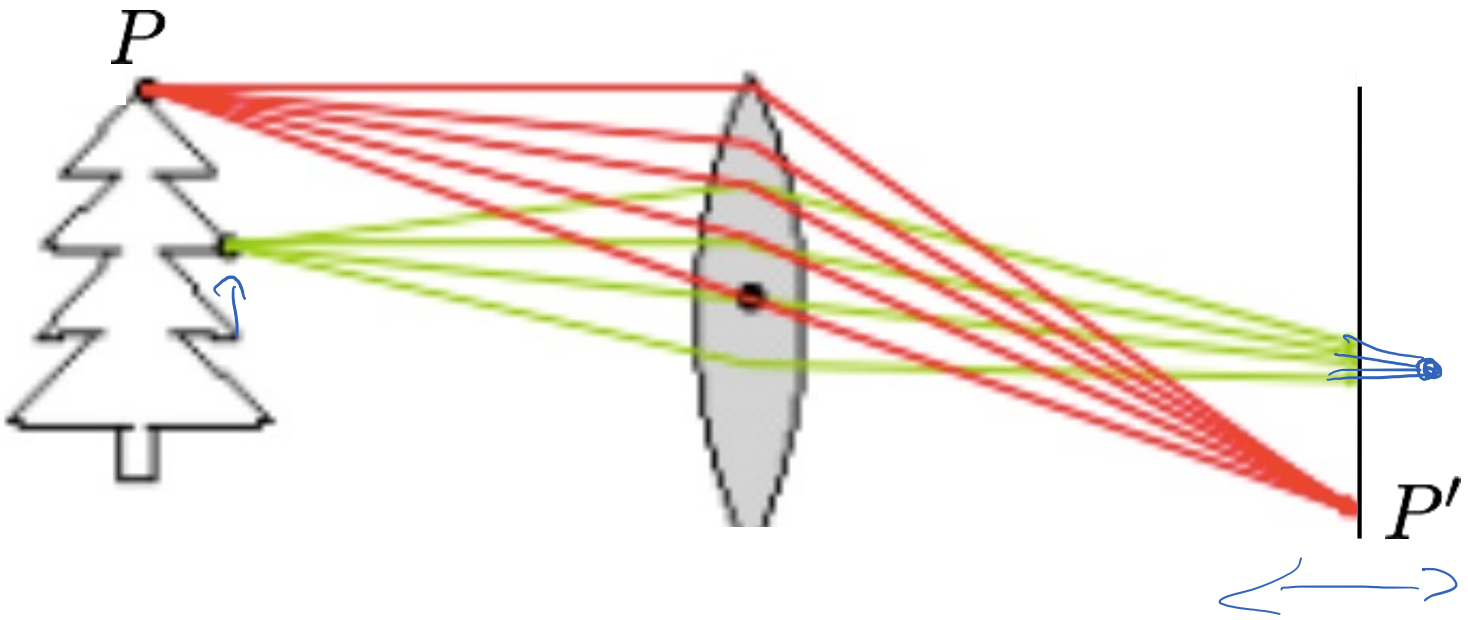
Skew parameter

$$K = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- When is $\gamma \neq 0$?
  - x- and y-axis of the camera are not perpendicular (unlikely)
  - For example, as a result of taking an image of an image
- Five parameters in total!

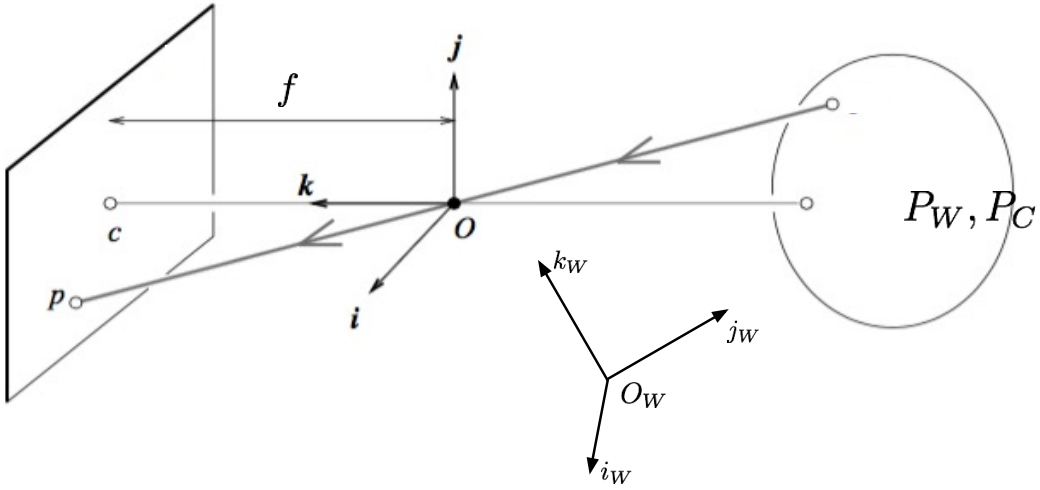
Lenses

- Lens: an optical element that focuses light by means of refraction

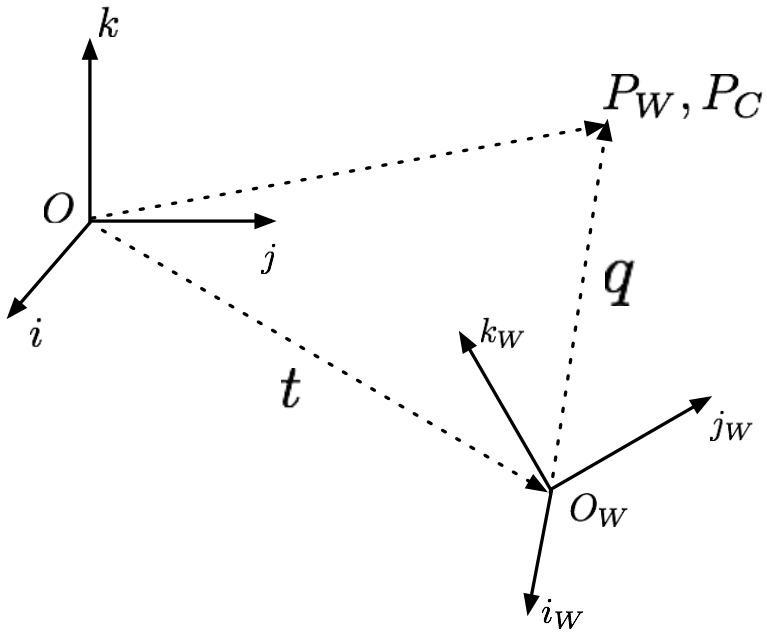


Step 3

- Last step is to include in our mapping an additional transformation to account for the difference between the world frame and the 3D camera reference frame



Rigid body transformations



$$P_C = t + q$$

$$q = R\,P_W$$

where *R* is the rotation matrix relating camera and world frames

$$R = \begin{bmatrix} i_W \cdot i & j_W \cdot i & k_W \cdot i \\ i_W \cdot j & j_W \cdot j & k_W \cdot j \\ i_W \cdot k & j_W \cdot k & k_W \cdot k \end{bmatrix}$$

$$\Rightarrow P_C = t + R\,P_W$$

Rigid transformations in homogeneous coordinates

$$\begin{pmatrix} P_C \\ 1 \end{pmatrix} = \begin{bmatrix} R & t \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{pmatrix} P_W \\ 1 \end{pmatrix}$$

Point P_c in homogeneous coordinates

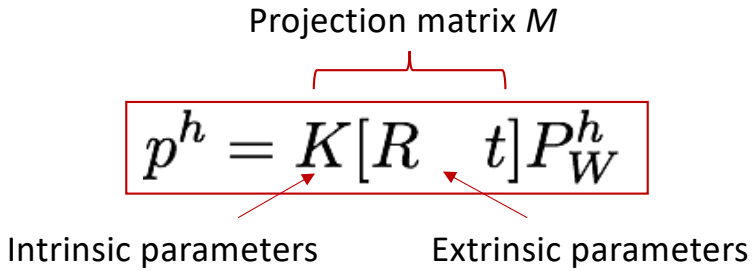
Point P_w in homogeneous coordinates

Perspective projection equation

- Collecting all results

$$p^h = [K \quad 0_{3\times 1}]P_C^h = K[I_{3\times 3} \quad 0_{3\times 1}] \begin{bmatrix} R & t \\ 0_{1\times 3} & 1 \end{bmatrix} P_W^h$$

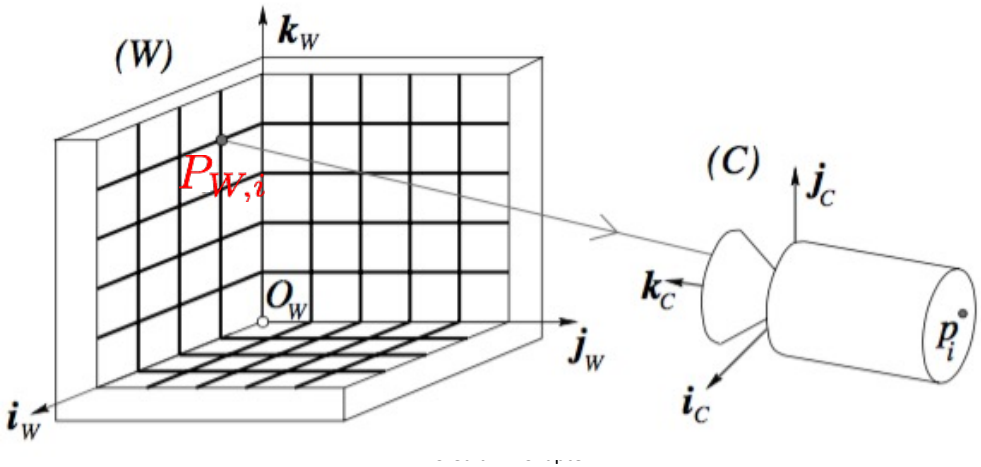
- Hence



- Degrees of freedom: 4 for *K* (or 5 if we also include skewness), 3 for *R*, and 3 for *t*. Total is 10 (or 11 if we include skewness)

Camera calibration: direct linear transformation method

- Goal: find the intrinsic and extrinsic parameters of the camera

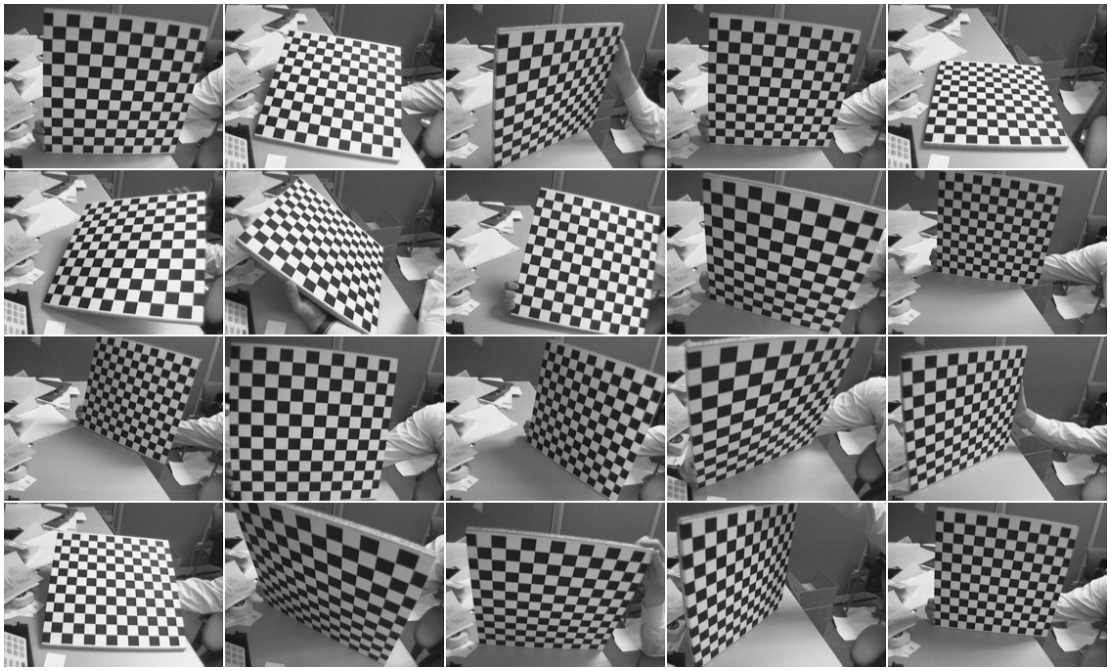


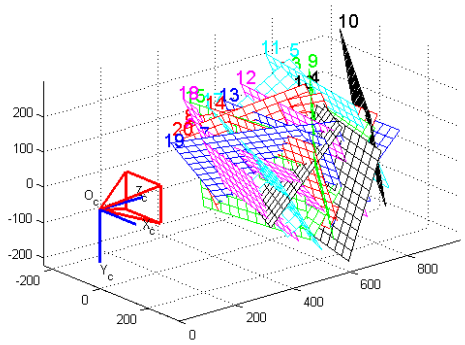
Strategy: given known correspondences $p_i \leftrightarrow P_{W,i}$, compute unknown parameters *K*, *R*, *t* by applying perspective projection

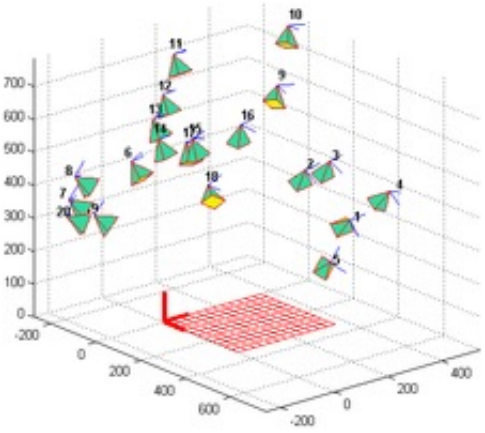
$P_{W,1}, P_{W,2}, \ldots, P_{W,n}$ with known positions in world frame

$p_1, p_2, \ldots, p_n$ with known positions in image frame

Examples for Calibration Images







Source: Wikipedia

Step 1

- First consider combined parameters

1×4 vector

$$p_i^h = M\, P_{W,i}^h,\ \ i = 1, \ldots, n, \qquad \text{where} \quad M = K[R \quad t] = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}$$

- This gives rise to $2n$ component-wise equations, for $i = 1, \ldots, n$

$$u_i = \frac{m_1 \cdot P_{W,i}^h}{m_3 \cdot P_{W,i}^h} \qquad \text{or} \qquad u_i \left(m_3 \cdot P_{W,i}^h\right) - m_1 \cdot P_{W,i}^h = 0$$

$$v_i = \frac{m_2 \cdot P_{W,i}^h}{m_3 \cdot P_{W,i}^h} \qquad\qquad v_i \left(m_3 \cdot P_{W,i}^h\right) - m_2 \cdot P_{W,i}^h = 0$$

Calibration problem

- Stacking all equations together

$$\tilde{P}m = 0, \qquad \text{where } m = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix}$$

$2n$ x 12 matrix of known coefficients

12 x 1 vector of unknown coefficients

12 x 1

- $\tilde{P}$ contains in block form the known coefficients stemming from the given correspondences
- To estimate 11 coefficients, we need at least 6 correspondences

Solution

- To find non-zero solution

$$\begin{aligned} \min_{m \in R^{12}} \quad & \|\tilde{P}m\|^2 \\ \text{subject to} \quad & \|m\|^2 = 1 \end{aligned}$$

- Solution: select eigenvector of $\tilde{P}^T\tilde{P}$ with the smallest eigenvalue
- Readily computed via SVD (singular value decomposition)

Step 2

- Next, we need to extract the camera parameters, i.e., we want to factorize *M* as

$$M = [KR \quad Kt]$$

- This can be done efficiently (indeed, explicitly) by using QR factorization, whereby the submatrix $\mathrm{M}_{1:3,1:3}$ is decomposed into the product of an upper triangular matrix *K* and a rotation matrix *R*

- Calibration will be investigated in Problem 1 in HW3