3D Vision 2

Week 8

Single-view Geometry: Camera Calibration

Single-view Geometry: Resectioning and Camera Pose

Announcements

- Assignment 2 due Friday (11:59pm Friday 26 April)
  - This includes a one week extension that has already been applied
  - **Zero** marks if either report or code submitted late (unless extension)
    - Submit early; you can always resubmit an updated version later
    - Depending on your internet connection and load on the TurnItIn servers, uploading can sometimes be slow, so please factor this into your submission schedule
  - Submit your report (PDF) and code (ZIP file) **separately under the correct tab** in the submission box
  - Follow the instructions under Submission Requirements

Announcements

- Public Holiday on Thursday 25 April:
  - Thursday lab rescheduled to **13:00-15:00 Tuesday Rm 109 CSIT Building**
- Assignment 2 due 11:59pm Friday 17 May (end of Week 11)

Weekly Study Plan: Overview

Wk	Starting	Lecture	Lab	Assessment
1	19 Feb	Introduction	✗	
2	26 Feb	Low-level Vision 1	1	
3	4 Mar	Low-level Vision 2	1	
		Mid-level Vision 1		
4	11 Mar	Mid-level Vision 2	1	CLab1 report due Friday
		High-level Vision 1		
5	18 Mar	High-level Vision 2	2	
6	25 Mar	High-level Vision 3[1]	2	
	1 Apr	Teaching break	✗	
	8 Apr	Teaching break	✗	
7	15 Apr	3D Vision 1	2	CLab2 report due Friday
8	22 Apr	3D Vision 2	3	
9	29 Apr	3D Vision 3	3	
10	6 May	3D Vision 4	3	
		Mid-level Vision 3		
11	13 May	High-level Vision 4	✗	CLab3 report due Friday
12	20 May	Course Review	✗	

Weekly Study Plan: Part B

Wk	Starting	Lecture	By
7	15 Apr	3D vision: introduction, camera model, single-view geometry	Dylan
8	22 Apr	3D vision: camera calibration, two-view geometry (homography)	Dylan
9	29 Apr	3D vision: two-view geometry (epipolar geometry, triangulation, stereo)	Dylan
10	6 May	3D vision: multiple-view geometry	Weijian
		Mid-level vision: optical flow, shape-from-X	Dylan
11	13 May	High-level vision: self-supervised learning, detection, segmentation	Dylan
12	20 May	Course review	Dylan

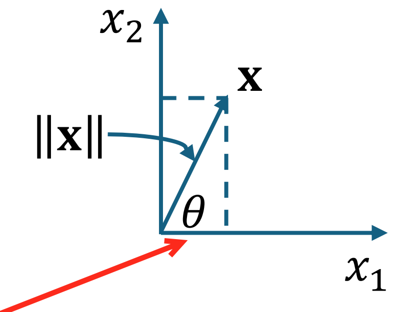
Outline

1. Single-view Geometry: Camera Calibration
2. Single-view Geometry: Resectioning and Absolute Camera Pose
3. Two-view Geometry: Homography Estimation

Vector Operations (Review)

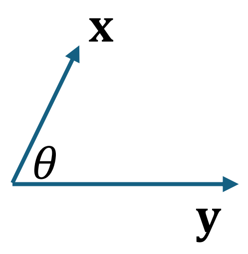
Vectors

- $\mathbf{x} = (x_1, x_2, x_3, \dots, x_n)$
- Magnitude: $\|\mathbf{x}\| = (x_1^2 + x_2^2 + \ x_3^2 + \cdots + x_n^2)^{\frac{1}{2}}$ L-2 norm
- Unit vector (magnitude is one): $\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$
- Orientation (for a 2D vector): $\theta = \tan^{-1}\frac{x_2}{x_1}$
- Homogeneous vectors (in P^n): $\tilde{\mathbf{x}}$
  - One fewer degrees-of-freedom than the number of dimensions
    - Known up to scale: only the ratios between coordinates are significant $x_1 : x_2 : x_3 : \dots$
  - $\tilde{\mathbf{x}} = k\tilde{\mathbf{x}}$



Inner (Dot) Product

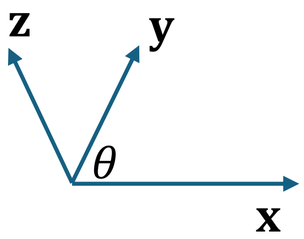
- $\mathbf{x} = (x_1, x_2, x_3, \ldots, x_n)$
- $\mathbf{y} = (y_1, y_2, y_3, \ldots, y_n)$
- $\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 + x_3 y_3 + \cdots + x_n y_n$ ➔ scalar
- $\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos\theta$
- If $\mathbf{x} \perp \mathbf{y}$, $\mathbf{x} \cdot \mathbf{y} = 0$



Vector (Cross) Product

https://en.wikipedia.org/wiki/Cross_product
https://www.mathsisfun.com/algebra/vectors-cross-product.html

- $\mathbf{x} = (x_1, x_2, x_3, \dots, x_n)$
- $\mathbf{y} = (y_1, y_2, y_3, \dots, y_n)$
- $\mathbf{z} = \mathbf{x} \times \mathbf{y}$ ➔ vector
- Magnitude: $\|\mathbf{z}\| = \|\mathbf{x}\|\,\|\mathbf{y}\| \sin\theta$ $\quad \|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\|\,\|\mathbf{b}\|\,|\sin\theta|$
- Orientation:
  - $\mathbf{z} \perp \mathbf{x} \Rightarrow \mathbf{x} \cdot \mathbf{z} = \mathbf{x} \cdot (\mathbf{x} \times \mathbf{y}) = 0$
  - $\mathbf{z} \perp \mathbf{y} \Rightarrow \mathbf{y} \cdot \mathbf{z} = \mathbf{y} \cdot (\mathbf{x} \times \mathbf{y}) = 0$
- If $\mathbf{x} \parallel \mathbf{y}, \mathbf{z} = \mathbf{0}$



- zero in length when vectors **a** and **b** point in the same, or opposite, direction
- reaches maximum length when vectors **a** and **b** are at right angles

Matrix notation [edit]

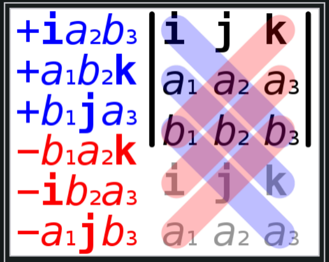
The cross product can also be expressed as the formal determinant:[note 1][1]

$$\mathbf{a} \times \mathbf{b} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{vmatrix}$$

This determinant can be computed using Sarrus's rule or cofactor expansion. Using Sarrus's rule, it expands to

$$\begin{aligned} \mathbf{a} \times \mathbf{b} &= (a_2b_3\mathbf{i} + a_3b_1\mathbf{j} + a_1b_2\mathbf{k}) - (a_3b_2\mathbf{i} + a_1b_3\mathbf{j} + a_2b_1\mathbf{k}) \\ &= (a_2b_3 - a_3b_2)\mathbf{i} + (a_3b_1 - a_1b_3)\mathbf{j} + (a_1b_2 - a_2b_1)\mathbf{k}. \end{aligned}$$

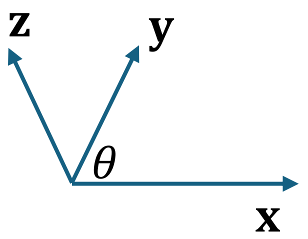
which gives the components of the resulting vector directly.



Use of Sarrus's rule to find the cross product of **a** and **b**

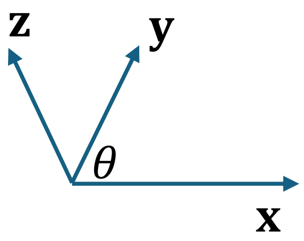
Vector (Cross) Product: Computation (3D)

- $\mathbf{x} = (x_1, x_2, x_3)$
- $\mathbf{y} = (y_1, y_2, y_3)$
- Unit basis vectors: $\hat{\mathbf{i}} = (1,0,0), \hat{\mathbf{j}} = (0,1,0), \hat{\mathbf{k}} = (0,0,1)$
  - $\hat{\mathbf{i}} = \hat{\mathbf{j}} \times \hat{\mathbf{k}}, \hat{\mathbf{j}} = \hat{\mathbf{k}} \times \hat{\mathbf{i}}, \hat{\mathbf{k}} = \hat{\mathbf{i}} \times \hat{\mathbf{j}}$
- $\boxed{\mathbf{z} = (x_2 y_3 - x_3 y_2)\hat{\mathbf{i}} + (\mathrm{x}_3 \mathrm{y}_1 - \mathrm{x}_1 \mathrm{y}_3)\hat{\mathbf{j}} + (x_1 y_2 - x_2 y_1)\hat{\mathbf{k}}}$



Vector (Cross) Product: Alternative Notation

- $\mathbf{x} = (x_1, x_2, x_3)$
- $\mathbf{y} = (y_1, y_2, y_3)$
- $\boxed{\mathbf{x} \times \mathbf{y} = [\mathbf{x}]_\times \mathbf{y} = -[\mathbf{y}]_\times \mathbf{x}}$
- Cross product matrix:
  - $[\mathbf{x}]_\times = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$
  - Anti-symmetric: $A = -A^\top$



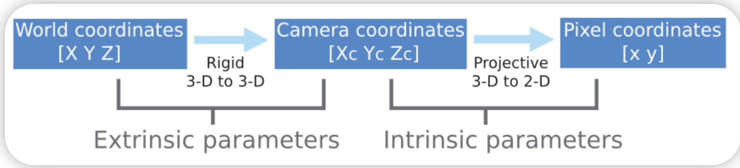
$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

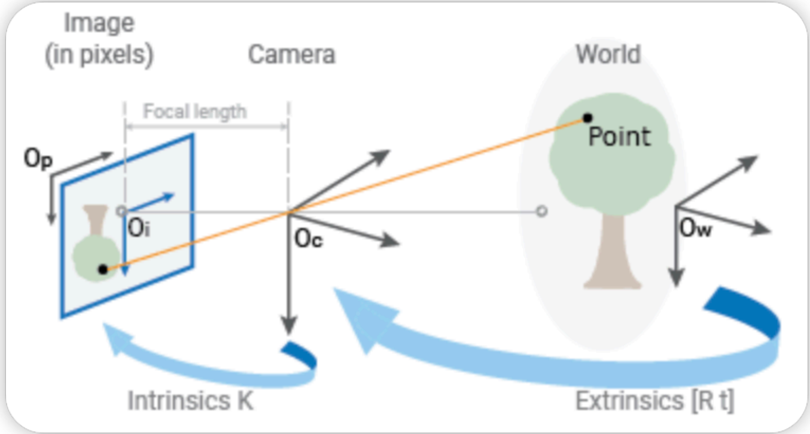
$$\mathbf{a} \times \mathbf{b} = [\mathbf{b}]_\times^{\mathrm{T}} \mathbf{a} = \begin{bmatrix} 0 & b_3 & -b_2 \\ -b_3 & 0 & b_1 \\ b_2 & -b_1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix},$$

Camera Calibration

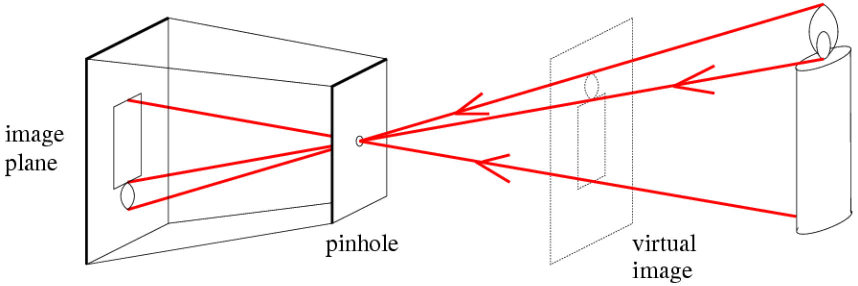
Recovering the Projection Matrix

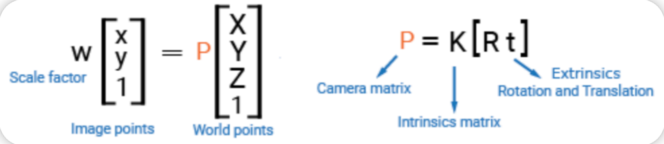
Objectives





- To **calibrate** a perspective camera:
  - To estimate the camera matrix $P = K[R|t]$
  - To estimate the camera calibration (intrinsics) matrix K
  - To estimate the camera extrinsic parameters $R, t/C$
- To understand the **Direct Linear Transformation** (DLT) algorithm





$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \underbrace{\begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}}_{P_{\{3\times4\}}} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Image Projection

- Project world point $\boldsymbol{X} = (x, y, z)$ to image point $\boldsymbol{x} = (u, v)$
- Assuming a pinhole camera model

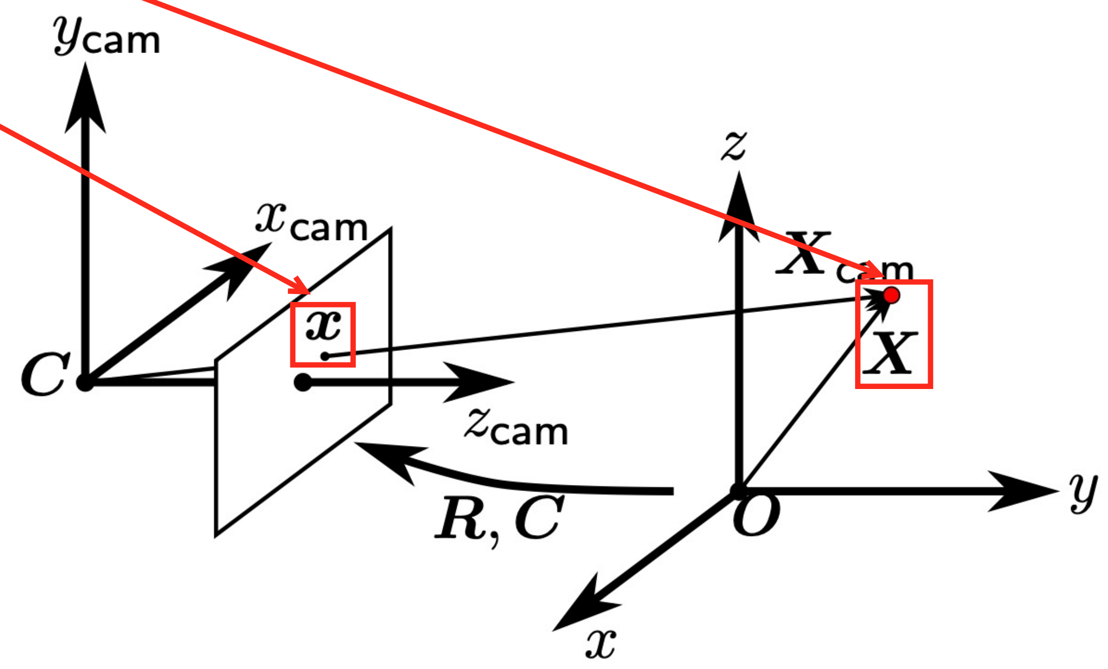


Image Projection

Image Point

World Point

$$\boldsymbol{x} = \boldsymbol{P}\boldsymbol{X}$$

$$= \boldsymbol{K}\boldsymbol{R}[\boldsymbol{I}| - \boldsymbol{C}]\boldsymbol{X} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\boldsymbol{X}$$

$$= \underbrace{\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsics}} \underbrace{\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}}_{\text{Extrinsics}} \boldsymbol{X}$$

- How many parameters?
  - 11 total:
    - 5 intrinsic $(\alpha, \beta, \gamma, u_0, v_0)$
    - 6 extrinsic $(\boldsymbol{R}, \boldsymbol{t})$
- How to compute?
  - Camera calibration!

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

C: Camera centre (vector) C_world

- Location of the camera in the world coordinate system

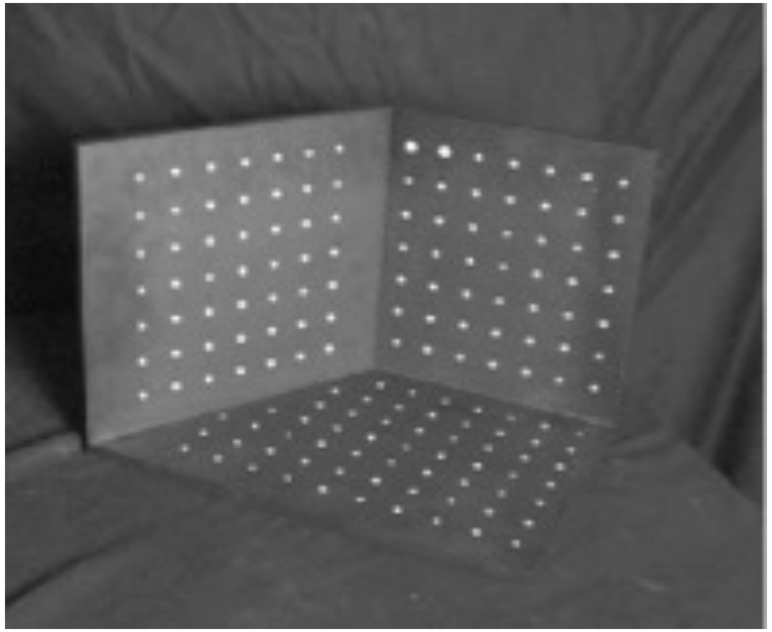
3 rotation angels in R +
3 C_world coord (X_c, Y_c, Z_c) in C

Two Approaches to Calibration

1. Calibrate to a meaningful fixed world coordinate system
   - Solve for P directly
   - Good for a fixed camera, specific applications
     - E.g., tracking vehicles on a highway, a mobile robot on the ground plane
   - But gives no insight into internal calibration parameters
   - If camera–world relationship changes, calibration must start from scratch
2. Compute internal and external parameters separately:
   - $P = K[R|t]$
   - Internal parameters turn camera into a metric device
   - Can now be used for computing 3D rays in Euclidean space
   - Necessary for SFM

Camera Calibration

- Determine the camera parameters from known 3D points or a calibration object(s)
  1. Internal or intrinsic parameters (i.e., focal length, principal point, aspect ratio)
  2. External or extrinsic (pose) parameters (i.e., position and orientation of the camera)
- *3D points cannot all be on the same plane*

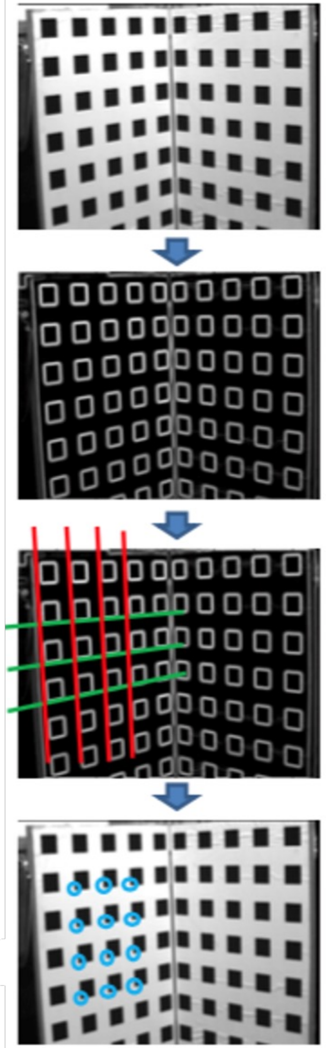


Basic Procedure

1. Prepare a calibration target/object
   - E.g., 2 orthogonal planes with a checkerboard pattern
   - Important: **3D point coordinates are known**
2. Position camera in front of target
   - Capture image of the calibration target
3. Find corners of the target in the image
   - Obtain **2D–3D correspondences**
4. Derive constraints on camera matrix
   - Estimate the intrinsic and extrinsic parameters

Basic Procedure: (3) Find Corners in Image

1. Option 1:
   1. Detect edges with Canny detector
   2. Fit straight lines to detected linked edges (Hough)
   3. Intersect lines to obtain 2D image coordinates
   4. Match image corners & 3D target checkerboard corners
      - E.g., count corners along the line and find corresponding edge in target
   5. Result: 2D–3D correspondences
2. Option 2:
   - Apply a corner detector directly (Harris, Susan, FAST)



Camera Projection Matrix

- Fold intrinsic calibration matrix **K** and extrinsic pose parameters (**R**,**t**) together into a camera matrix **P**
- **P** = **K** [**R** | **t**]

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & \boxed{1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Put 1 in the lower right corner to remove a degree of freedom (DoF)
  - 11 DoFs

Inhomogeneous Projection Equation

- Directly estimate 11 unknowns in the M matrix using known 3D points (X_i, Y_i, Z_i) and measured feature positions (u_i, v_i)
  - **Nonlinear** equations

X/Z $$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

Y/Z $$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

Linear Regression

1. Bring denominator over to the LHS
2. Solve set of (over-determined) linear equations for m_{ij}
   - How? Least squares (pseudo-inverse)

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) \quad = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$
$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) \quad = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

Direct Linear Transformation (DLT) Algorithm for Camera Calibration

- Cross-product trick:
  - When an equation is only **known up to scale**, take the cross product of the LHS with both sides of the equation – no loss of information
  - same direction hence paralllel
    $\mathbf{x}_i = k\mathbf{P}\mathbf{X}_i \Rightarrow \mathbf{x}_i \times \mathbf{P}\mathbf{X}_i = \mathbf{0}$ [Why?] ($\mathbf{x}_i$: image coords; $\mathbf{X}_i$: world coords)

image world

$$[\mathbf{x}_i]_\times \mathbf{P}\mathbf{X}_i = \begin{pmatrix} y_i\mathbf{p}_3^\top\mathbf{X}_i - w_i\mathbf{p}_2^\top\mathbf{X}_i \\ w_i\mathbf{p}_1^\top\mathbf{X}_i - x_i\mathbf{p}_3^\top\mathbf{X}_i \\ x_i\mathbf{p}_2^\top\mathbf{X}_i - y_i\mathbf{p}_1^\top\mathbf{X}_i \end{pmatrix} = \begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \\ -y_i\mathbf{X}_i^\top & x_i\mathbf{X}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = \mathbf{A}_i'\mathbf{p} = \mathbf{0}$$

12 * 12 12 * 1

image

where $\mathbf{x}_i = (x_i, y_i, w_i)^\top$ and $\mathbf{P} = \begin{bmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \end{bmatrix}$; $\mathbf{p}_i \in \mathbb{R}^{4\times1}$

Direct Linear Transformation (DLT) Algorithm for Camera Calibration

- Only 2 out of 3 equations are linearly independent, so pick two

$$\begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix} = \mathbf{A}_i\mathbf{p} = \mathbf{0}$$

- Camera matrix has 11 DoF:
  - 12 parameters defined up to scale
  - Linear solution requires at least 6 points (in fact, 5½)

Direct Linear Transformation (DLT) Algorithm for Camera Calibration

- Using 6 points to solve for $\mathbf{P}$: (6 * 2) * 12

A_i is from (x_i and X_i) pair of data
A_i ∊ R^(2*12)

$$\mathbf{A}\mathbf{p} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_6 \end{bmatrix} \mathbf{p} = \mathbf{0}$$

12 * 1

  - $\mathbf{A} \in \mathbb{R}^{12\times12}$ but $\text{rank}(\mathbf{A}) = 11$
  - So use the first 11 rows: $\mathbb{R}^{11\times12}$

discard the last row from A_6

- How to solve?
  - The trivial solution $\mathbf{p} = \mathbf{0}$ is not interesting
  - Compute the 1D null-space (e.g., via SVD)
  - Fix norm of $\mathbf{p}$ afterwards (e.g., set $||\mathbf{p}|| = 1$)

Direct Linear Transformation (DLT) Algorithm for Camera Calibration

- Using n points to solve for $\mathbf{P}$: the over-determined case

$$\mathbf{Ap} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \mathbf{p} = \mathbf{0}$$

- How to solve?
  - No exact non-trivial solution due to inexact measurements (e.g., noise)
    - $\mathbf{Ap} = \mathbf{0}$ is not possible, so minimise $||\mathbf{Ap}||$ subject to $||\mathbf{p}|| = 1$
  1. Take the singular value decomposition (SVD) of $\mathbf{A}$

A∈R^(11*12) • $\mathbf{A} = \mathbf{U\Sigma V}^\top$ U $(11 * 11)$ Σ $(11 * 12)$ V.T $(11 * 12)$

  2. Take the rightmost column of $\mathbf{V}$ [why?]
     - The right-singular vector of $\mathbf{A}$, corresponding to the smallest singular value (arranged in decreasing singular value order)

Direct Linear Transformation (DLT) Algorithm for Camera Calibration

- Why does the rightmost column of $\mathbf{V}$ minimise $||\mathbf{Ap}||$ s.t. $||\mathbf{p}|| = 1$?
- $\mathbf{Ap} = \mathbf{U}\Sigma\mathbf{V}^\top\mathbf{p} = \sum_{i=1}^{\mathrm{rank}(\mathbf{A})} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \mathbf{p} = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^\top \mathbf{p} + \cdots + \sigma_{11} \mathbf{u}_{11} \mathbf{v}_{11}^\top \mathbf{p}$

  where $\Sigma = \begin{pmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & 0 \\ 0 & 0 & \sigma_{11} \end{pmatrix}$ in decreasing singular value order,

  with left-singular vectors $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_{11}]$ and right-singular vectors $\mathbf{V} = [\mathbf{v}_1 \cdots \mathbf{v}_{11}]$
- Set $\mathbf{p} = \mathbf{v}_{11}$ (final column of $\mathbf{V}$) :
- $\mathbf{A}\mathbf{v}_{11} = \underbrace{\sigma_1}_{\text{Largest singular value}} \mathbf{u}_1 \underbrace{\cancel{\mathbf{v}_1^\top \mathbf{v}_{11}}}_{0\text{ (orthogonal)}} + \cdots + \underbrace{\sigma_{11}}_{\text{Smallest singular value}} \mathbf{u}_{11} \underbrace{\cancel{\mathbf{v}_{11}^\top \mathbf{v}_{11}}}_{1\text{ (unit, parallel)}} = \underbrace{\sigma_{11} \mathbf{u}_{11}}_{\text{Very small}}$

eigenvectors are orthogonal with each other and normalised to 1

Direct Linear Transformation (DLT) Algorithm for Camera Calibration

- Objective:
  - Given $n \geq 6$ 2D–3D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$, determine the 3×4 projection matrix $\mathbf{P}$ such that $\mathbf{x}_i \approx \mathbf{P}\mathbf{X}_i$

- Algorithm:

  2 * 12

  1. For each correspondence $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$ compute $\mathbf{A}_\mathrm{i}$, taking only the first two rows $\begin{bmatrix} \mathbf{0}^\top & -w_i\mathbf{X}_i^\top & y_i\mathbf{X}_i^\top \\ w_i\mathbf{X}_i^\top & \mathbf{0}^\top & -x_i\mathbf{X}_i^\top \end{bmatrix}$
  2. Assemble the n 2×12 $\mathbf{A}_\mathrm{i}$ matrices into a single $2n$×12 matrix $\mathbf{A}$
  3. Compute the SVD of $\mathbf{A}$: $\mathbf{U\Sigma V}^\top$
  4. Take the last column of $\mathbf{V}$ as the solution for $\mathbf{p}$ 12 * 1
  5. Rearrange $\mathbf{p}$ to obtain $\mathbf{P}$ 12 * 1 → 3 * 4

$$\mathbf{Ap} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_n \end{bmatrix} \mathbf{p} = \mathbf{0}$$

Importance of Normalisation

A_i (2*12)
$$\begin{bmatrix} 0 & 0 & 0 & 0 & -w_iX_1 & -w_iX_2 & -w_iX_3 & -w_i & y_iX_1 & y_iX_2 & y_iX_3 & y_i \\ w_iX_1 & w_iX_2 & w_iX_3 & w_i & 0 & 0 & 0 & 0 & -x_iX_1 & -x_iX_2 & -x_iX_3 & -x_i \end{bmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{pmatrix}$$

$$\sim 10^2 \quad \sim 10^2 \quad \sim 10^2 \quad 1 \quad \sim 10^2 \quad \sim 10^2 \quad \sim 10^2 \quad 1 \quad \sim 10^4 \quad \sim 10^4 \quad \sim 10^4 \quad \sim 10^2$$

- Orders of magnitude difference!

Normalised Direct Linear Transformation (DLT) Algorithm for Camera Calibration

- Objective:
  - Given $n \geq 6$ 2D–3D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$, determine the 3×4 projection matrix $\mathbf{P}$ such that $\mathbf{x}_i \approx \mathbf{P}\mathbf{X}_i$
- Algorithm:
  1. Normalise 2D and 3D points: $\tilde{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i,\ \tilde{\mathbf{X}}_i = \mathbf{S}\mathbf{X}_i$
  2. Apply the DLT algorithm to $\left\{\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{X}}_i\right\}$
  3. Denormalise the recovered solution $\tilde{\mathbf{P}}$ using $\mathbf{P} = \mathbf{T}^{-1}\tilde{\mathbf{P}}\mathbf{S}$
- Example normalisation matrices:

$$\mathbf{T} = \begin{bmatrix} w+h & 0 & w/2 \\ 0 & w+h & h/2 \\ 0 & 0 & 1 \end{bmatrix}^{-1} ; \quad \mathbf{S} = \begin{bmatrix} \mathbf{V}\mathrm{diag}(\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1})\mathbf{V}^{-1} & -\mathbf{V}\mathrm{diag}(\lambda_1^{-1}, \lambda_2^{-1}, \lambda_3^{-1})\mathbf{V}^{-1}\boldsymbol{\mu}_{\mathbf{X}_i} \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{V}\mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)\mathbf{V}^{-1} = \mathrm{eig}\left(\textstyle\sum_i \left(\mathbf{X}_{i,\mathrm{inhom}} - \boldsymbol{\mu}_{\mathbf{X}_i}\right)\left(\mathbf{X}_{i,\mathrm{inhom}} - \boldsymbol{\mu}_{\mathbf{X}_i}\right)^\top\right)$$

Camera Calibration

Recovering the Camera Intrinsics

Camera Matrix Decomposition: Computing the Camera Centre **C**

3*4 3*3 3*4 p_i (3*1)

- $\mathbf{P} = \mathbf{K}[\mathbf{R} \mid -\mathbf{R}\mathbf{C}] = [\mathbf{p}_1 \quad \mathbf{p}_2 \quad \mathbf{p}_3 \quad \mathbf{p}_4]$
  - Careful! These $\mathbf{p}_\mathrm{i}$ are now column vectors of $\mathbf{P}$

1. It is the right null-space vector of $\mathbf{P}$ (Hartley & Zisserman p. 163):
   - Take last column of $\mathbf{V}$ where $\mathbf{P} = \mathbf{U}\Sigma\mathbf{V}^\top$ is the SVD of $\mathbf{P}$
   - Why? Observe that $\mathbf{PC} = \mathbf{KR}\begin{bmatrix}1 & & & -X_C \\ & 1 & & -Y_C \\ & & 1 & -Z_C\end{bmatrix}\begin{bmatrix}X_C \\ Y_C \\ Z_C \\ 1\end{bmatrix} = \mathbf{0}$ C, the Cam Centre in world coord., is projected to origin point in the image

2. Or, algebraic derivation (Hartley & Zisserman p. 163):
   - $$\boxed{C = \begin{bmatrix}X \\ Y \\ Z \\ T\end{bmatrix} = \begin{bmatrix}\det([\mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4]) \\ -\det([\mathbf{p}_1, \mathbf{p}_3, \mathbf{p}_4]) \\ \det([\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_4]) \\ -\det([\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3])\end{bmatrix}}$$

Camera Matrix Decomposition: Computing the Intrinsics **K** and Rotation **R**

- $\overset{3*4}{\mathbf{P}} = \overset{3*3}{\mathbf{K}}[\overset{3*3}{\mathbf{R}} \mid -\overset{3*1}{\mathbf{RC}}] = [\overset{3*3}{\mathbf{M}} \mid -\overset{3*1}{\mathbf{MC}}]$

1. RQ decomposition of $\mathbf{M}$:
   - $(\overset{K}{\mathbf{R}_\Delta}, \overset{R}{\mathbf{Q}}) = \mathrm{RQ}(\mathbf{M})$
   - $\mathbf{K} = \mathbf{R}_\Delta$: upper triangular matrix (Δ is just to distinguish it from rotation)
   - $\mathbf{R} = \mathbf{Q}$: orthonormal matrix
2. Algebraic derivation:
   - See next slide

Camera Matrix Decomposition: Computing the Intrinsics **K** and Rotation **R**

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$
$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$
$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- $\mathbf{P} = \mathbf{K}[\mathbf{R} \mid -\mathbf{RC}] = [\mathbf{M} \mid -\mathbf{MC}]$

2. Algebraic derivation:
   - Givens rotations: $\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}$ $\mathbf{R}_y = \begin{bmatrix} c' & 0 & s' \\ 0 & 1 & 0 \\ -s' & 0 & c' \end{bmatrix}$ $\mathbf{R}_z = \begin{bmatrix} c'' & -s'' & 0 \\ s'' & c'' & 0 \\ 0 & 0 & 1 \end{bmatrix}$
     - $c = -\frac{m_{33}}{\sqrt{m_{32}^2 + m_{33}^2}}$ $s = \frac{m_{32}}{\sqrt{m_{32}^2 + m_{33}^2}}$

1. Multiply $\mathbf{M}$ by $\mathbf{R}_x$: the resulting term at (3,2) will be 0 because of the values selected for c and s
2. Multiply resulting matrix by $\mathbf{R}_y$ such that resulting term at (3,1) is zero (select c' and s' accordingly)
3. Multiply resulting matrix by $\mathbf{R}_z$ such that resulting term at (2,1) is zero (select c'' and s'' accordingly)

Camera Matrix Decomposition: Computing the Intrinsics **K** and Rotation **R**

- $\mathbf{P} = \mathbf{K}[\mathbf{R} \mid -\mathbf{RC}] = [\mathbf{M} \mid -\mathbf{MC}]$

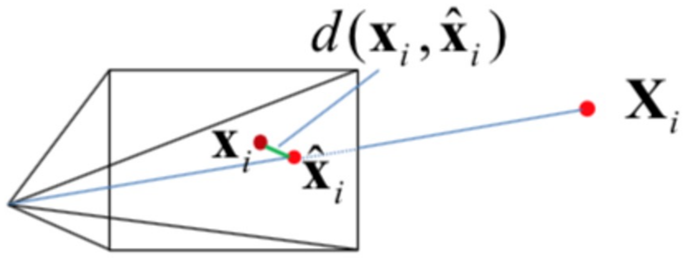
2. Algebraic derivation:
   - Givens rotations: $\mathbf{R}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{bmatrix}$ $\mathbf{R}_y = \begin{bmatrix} c' & 0 & s' \\ 0 & 1 & 0 \\ -s' & 0 & c' \end{bmatrix}$ $\mathbf{R}_z = \begin{bmatrix} c'' & -s'' & 0 \\ s'' & c'' & 0 \\ 0 & 0 & 1 \end{bmatrix}$
     - $c = -\frac{m_{33}}{\sqrt{m_{32}^2 + m_{33}^2}}$ $s = \frac{m_{32}}{\sqrt{m_{32}^2 + m_{33}^2}}$
   - Then,
     - $\mathbf{K} = \mathbf{M}\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z$
     - $\mathbf{M} = \mathbf{K}\mathbf{R}_z^\top\mathbf{R}_y^\top\mathbf{R}_x^\top \Rightarrow \mathbf{R} = \mathbf{R}_z^\top\mathbf{R}_y^\top\mathbf{R}_x^\top$

Camera Calibration

Geometric Solvers for Recovering the Projection Matrix

Camera Calibration with Geometric Solvers

- So far, minimisation of an **algebraic error** criterion
  - Advantage: **linear solutions**
  - Disadvantage: no explicit geometric meaning
- Refinement:
  - Nonlinear minimisation of a **geometric error** loss function



Camera Calibration with Geometric Solvers

- Minimise objective function over $\mathbf{P}$:

pred

$$\min_{\mathbf{P}} \sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2$$

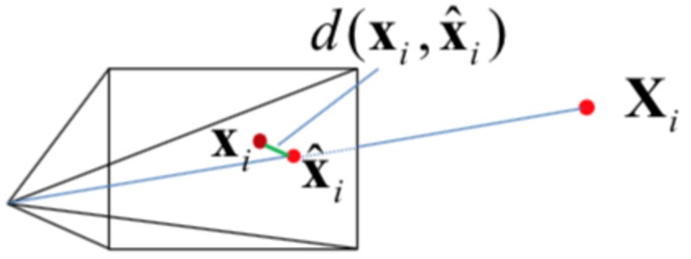
pred

$$= \min_{\mathbf{P}} \sum_i d(\mathbf{x}_i, \mathbf{P}\mathbf{X}_i)^2$$

$$= \min_{\mathbf{P}} \sum_i \left(\frac{x_i}{w_i} - \frac{\mathbf{p}_1^\top \mathbf{X}_i}{\mathbf{p}_3^\top \mathbf{X}_i}\right)^2 + \left(\frac{y_i}{w_i} - \frac{\mathbf{p}_2^\top \mathbf{X}_i}{\mathbf{p}_3^\top \mathbf{X}_i}\right)^2$$

x_img (homo)

for $\boldsymbol{x}_i = (x_i, y_i, w_i)^\top$ and $\mathbf{P} = \begin{bmatrix} \mathbf{p}_1^\top \\ \mathbf{p}_2^\top \\ \mathbf{p}_3^\top \end{bmatrix}$ p_i (4*1)



- Nonlinear optimisation, so going to be slow

“Gold Standard” Algorithm for Camera Calibration

1. Compute an *initial* solution using the normalised DLT algorithm
2. Refine the normalised solution using iterative minimisation of a geometric error
   - Use a nonlinear solver, like lsqnonlin in Matlab
3. Denormalise solution

Summary: DLT Camera Calibration

- Advantages:
  - Very simple to formulate and solve
  - Can recover $\mathbf{K}, \mathbf{R}, \mathbf{C}$ from $\mathbf{P}$ using RQ decomposition [Golub & VanLoan 96]
- Disadvantages:
  - Does not compute internal parameters explicitly
  - Sometimes involves more unknowns than true degrees of freedom
  - Need a separate camera matrix for each new view

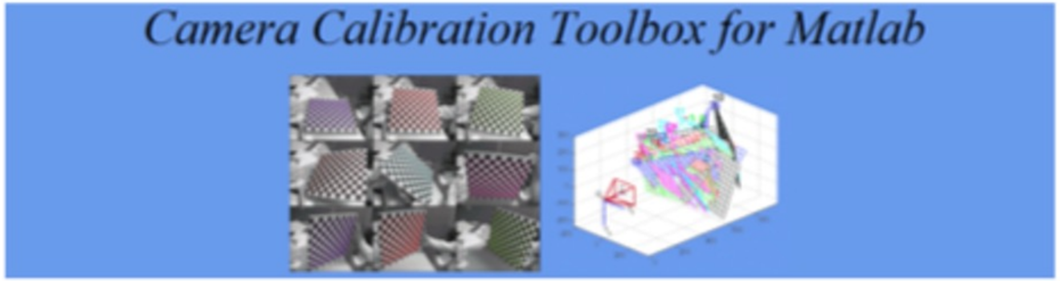
P

Practical / Popular Camera Calibration Algorithms

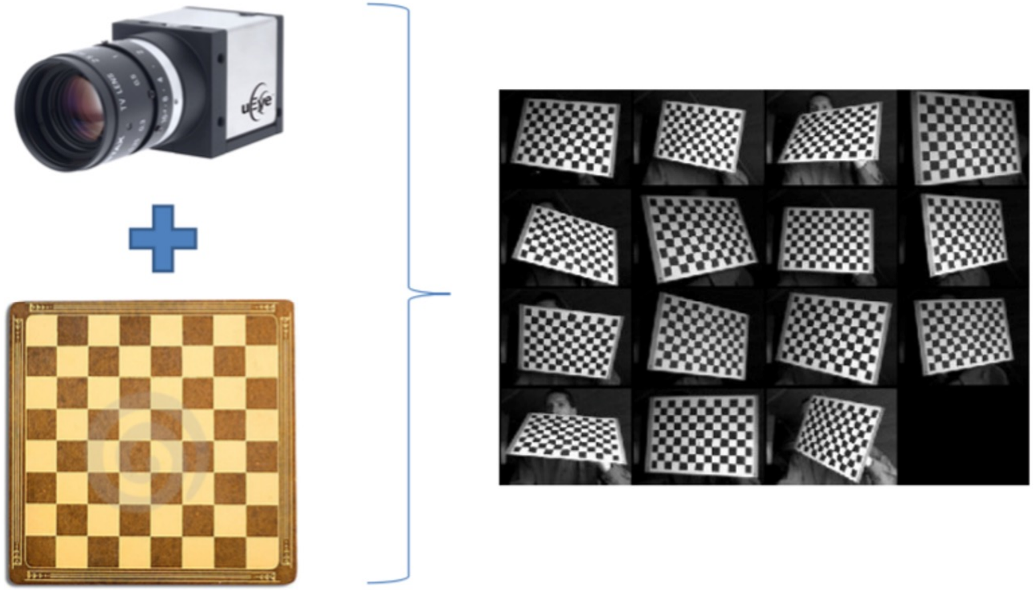
Practical Camera Calibration

1. Load images into a calibration toolbox
2. Calibrate

- Example toolboxes:
  - C++: OpenCV
  - Matlab: Calibration toolbox by Jean-Yves Bouguet
    http://www.vision.caltech.edu/bouguetj/calib_doc/



Multi-plane Calibration



Multi-plane Calibration: Zhang's Method

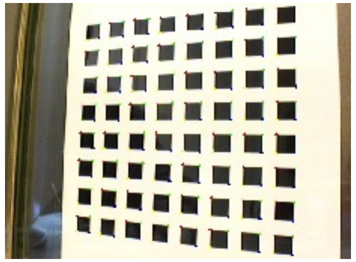
- Use several images of a planar target held at *unknown* poses [Zhang PAMI 99]

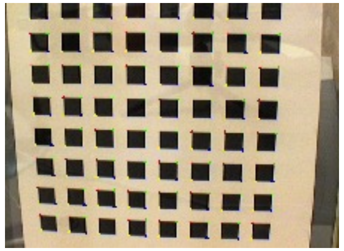
1. Compute plane homographies:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim \mathbf{K}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}] \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim \mathbf{HX}$$

2. Solve for $\mathbf{K}^{-\top}\mathbf{K}^{-1}$ from $\mathbf{H}_k$'s
   - 1 plane if only f unknown
   - 2 planes if (f, u_c, v_c) unknown
   - 3+ planes for full $\mathbf{K}$

- Code available on OpenCV





Camera Resectioning

Absolute Camera Pose

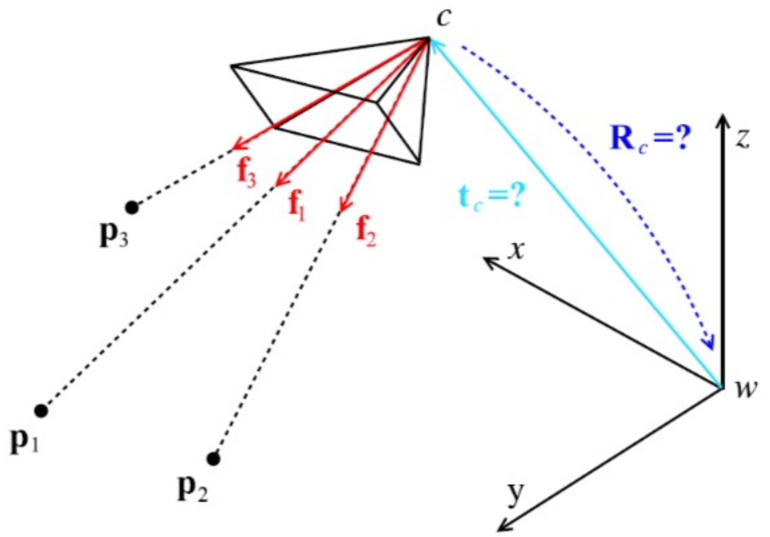
Overview

- Objective:
  - Given 2D–3D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{X}_i\}$ & the camera intrinsics $\mathbf{K}$, estimate the position $\mathbf{C}$ and orientation $\mathbf{R}$ of the camera
- Synonyms:
  - Camera resectioning
  - Absolute camera pose estimation
  - Perspective-n-point problem

Motivation

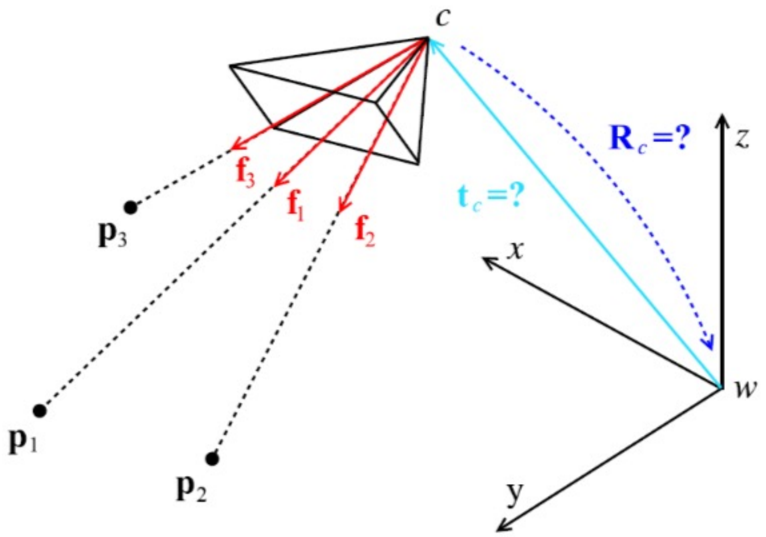
- If we have an existing reconstruction and a calibrated camera, and we just want to know where our camera is right now
  - Offline: calibration, e.g., with the DLT algorithm
  - Online: absolute pose, relative pose, triangulation
    - Unless there is mechanical change, $\mathbf{K}$ will remain constant [why might it change?]
- But didn't we just work out how to recover $\mathbf{R}$ and $\mathbf{C}$?
  - Recovering from the camera matrix involves estimating 11 DoFs
  - This is many more degrees-of-freedom than we need

Perspective-n-Point Problem



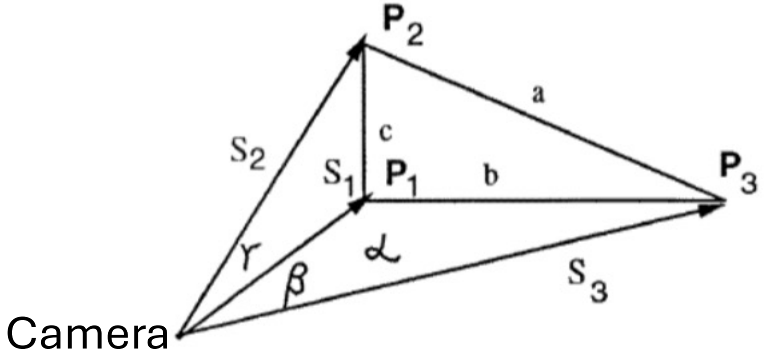
Perspective-n-Point Problem

- Degrees-of-freedom:
  - 6: 3 (translation) + 3 (rotation)
- Minimal solution:
  - 3 point correspondences
  - Perspective-3-point (P3P) problem



Perspective-n-Point Problem

- One solution
  - Haralick et al., IJCV 1994
- Variable elimination leads to a 4th order polynomial
  - 4 solutions
  - Use a fourth point correspondence to disambiguate



$$s_2^2 + s_3^2 - 2s_2s_3\cos\alpha = a^2$$
$$s_1^2 + s_3^2 - 2s_1s_3\cos\beta = b^2$$
$$s_1^2 + s_2^2 - 2s_1s_2\cos\gamma = c^2$$

Cosine rule

Next Lecture

- Two-view geometry:
  - Homographies
  - Homography estimation
  - Epipolar geometry