PSET 1 Part 2 + Project Proposal

Krishnan Srinivasan

CS231A

04/19/2024

Overview

- Lecture Review
- PSET 1
- Project Proposal Tips

Goal of calibration

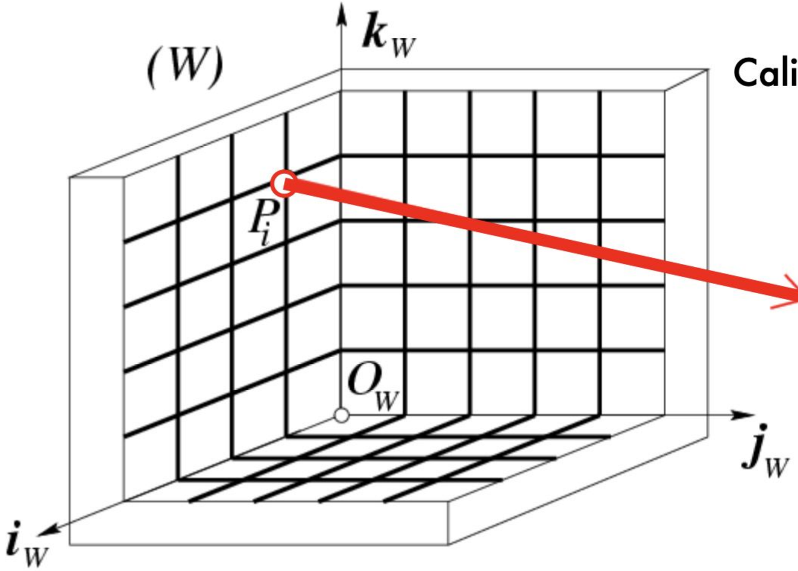
$$P' = M\, P_w = \underbrace{K}_{\text{Internal parameters}} \underbrace{\begin{bmatrix} R & T \end{bmatrix}}_{\text{External parameters}} P_w$$

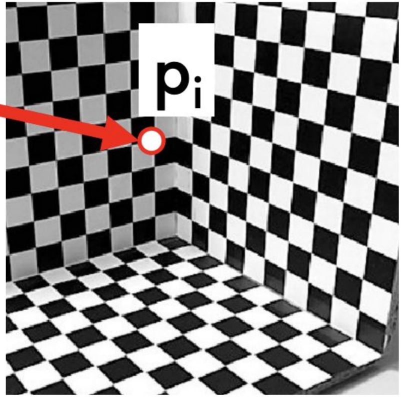
Estimate intrinsic and extrinsic parameters from 1 or multiple images

Change notation:
$P = P_w$

$p = P'$

Calibration Problem

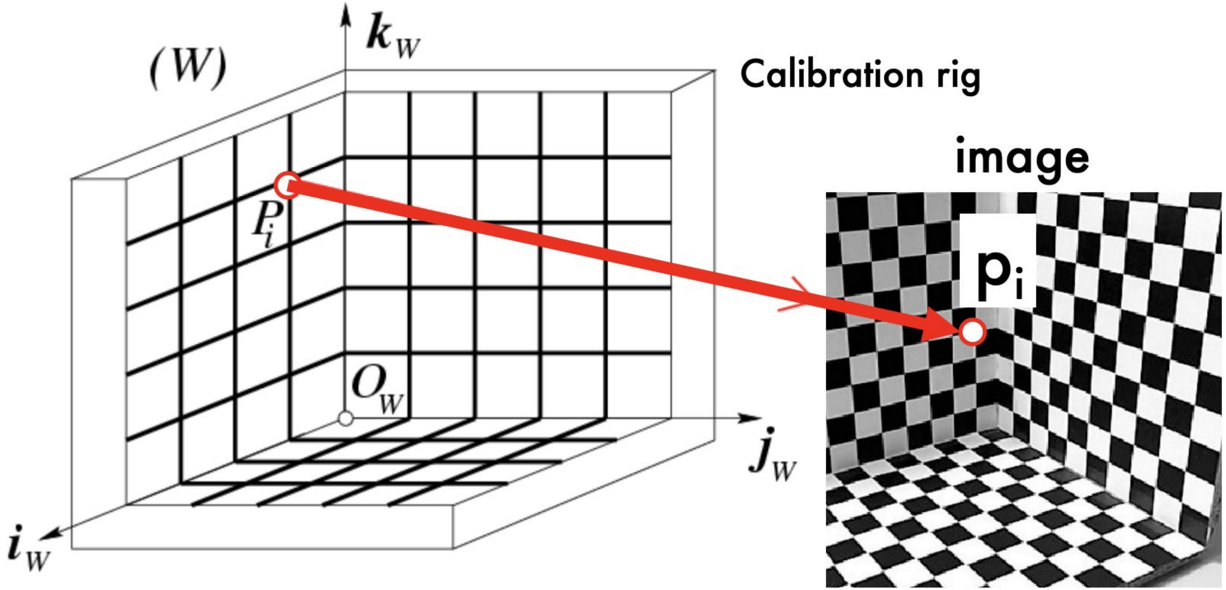




- $P_1 \ldots P_n$ with known positions in $[O_w, i_w, j_w, k_w]$
- $p_1, \ldots p_n$ known positions in the image

Goal: compute intrinsic and extrinsic parameters

Calibration Problem



How many correspondences do we need?

- M has 11 unknowns • We need 11 equations • 6 correspondences would do it

Calibration Problem

[Eq. 1]

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \dfrac{\mathbf{m}_1 P_i}{\mathbf{m}_3 P_i} \\ \dfrac{\mathbf{m}_2 P_i}{\mathbf{m}_3 P_i} \end{bmatrix}$$

$$u_i = \frac{\mathbf{m}_1 P_i}{\mathbf{m}_3 P_i} \rightarrow u_i(\mathbf{m}_3 P_i) = \mathbf{m}_1 P_i \rightarrow u_i(\mathbf{m}_3 P_i) - \mathbf{m}_1 P_i = 0$$

$$v_i = \frac{\mathbf{m}_2 P_i}{\mathbf{m}_3 P_i} \rightarrow v_i(\mathbf{m}_3 P_i) = \mathbf{m}_2 P_i \rightarrow v_i(\mathbf{m}_3 P_i) - \mathbf{m}_2 P_i = 0$$

[Eqs. 2]

Calibration Problem

$$\begin{cases} -u_1(\mathbf{m}_3\ P_1) + \mathbf{m}_1\ P_1 = 0 \\ -v_1(\mathbf{m}_3\ P_1) + \mathbf{m}_2\ P_1 = 0 \\ \vdots \\ -u_n(\mathbf{m}_3\ P_n) + \mathbf{m}_1\ P_n = 0 \\ -v_n(\mathbf{m}_3\ P_n) + \mathbf{m}_2\ P_n = 0 \end{cases}$$

[Eqs. 3]

known

unknown

$$\longrightarrow \quad \boxed{\mathbf{P}\,\mathbf{m} = 0}$$

[Eq. 4]

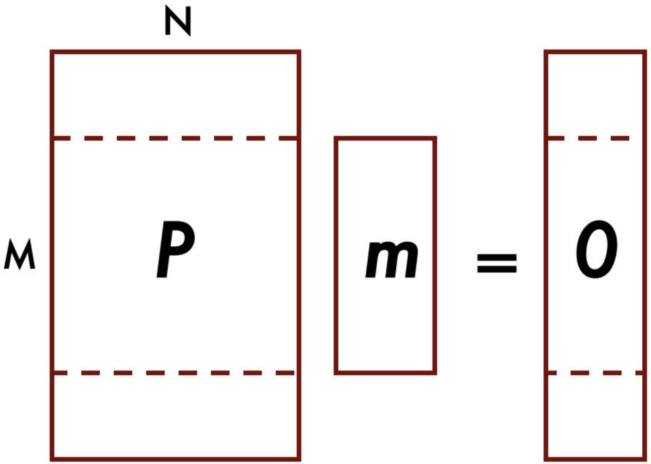
Homogenous linear system

$$\mathbf{P} \stackrel{\text{def}}{=} \begin{pmatrix} \mathrm{P}_1^T & \mathbf{0}^T & -u_1 \overset{1\text{x}4}{\mathrm{P}_1^T} \\ \mathbf{0}^T & \mathrm{P}_1^T & -v_1 \mathrm{P}_1^T \\ & \vdots & \\ \mathrm{P}_n^T & \mathbf{0}^T & -u_n \mathrm{P}_n^T \\ \mathbf{0}^T & \mathrm{P}_n^T & -v_n \mathrm{P}_n^T \end{pmatrix}_{2n \text{ x } 12}$$

$$\boldsymbol{m} \stackrel{\text{def}}{=} \begin{pmatrix} \overset{4\text{x}1}{\mathbf{m}_1^{\mathrm{T}}} \\ \mathbf{m}_2^{\mathrm{T}} \\ \mathbf{m}_3^{\mathrm{T}} \end{pmatrix}_{12\text{x}1}$$

Homogeneous M x N Linear Systems

M=number of equations = 2n
N=number of unknown = 11



Rectangular system (M>N)

- 0 is always a solution
- To find non-zero solution

Minimize $|\mathbf{P}\ \mathbf{m}|^2$

under the constraint $|\mathbf{m}|^2 = 1$

Calibration Problem

$$\boxed{\mathbf{P}}\,\mathbf{m} = 0$$

SVD decomposition of P

$$\boxed{U_{2n\times 12}\, D_{12\times 12} V^{T}_{12\times 12}}$$

Last column of V gives $\boldsymbol{m}$

Why? See pag 592 of HZ

$$\boldsymbol{m} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{m}_1^T \\ \mathbf{m}_2^T \\ \mathbf{m}_3^T \end{pmatrix}$$

$$\boldsymbol{m} \longrightarrow M$$

PSET 1 3(b) needs this!

Extracting camera parameters

See [FP], Sec. 1.3.1

$$\frac{M}{\rho} = \left(\underbrace{\begin{matrix}\alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T \\ \dfrac{\beta}{\sin\theta}\boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T \\ \boldsymbol{r}_3^T\end{matrix}}_{A} \quad \underbrace{\begin{matrix}\alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ t_z\end{matrix}}_{\mathbf{b}}\right)$$

Box 1

$$A = \begin{bmatrix}\mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T\end{bmatrix} \quad b = \begin{bmatrix}b_1 \\ b_2 \\ b_3\end{bmatrix}$$

Estimated values

Intrinsic

$$\rho = \frac{\pm 1}{|\mathbf{a}_3|} \quad u_o = \rho^2(\mathbf{a}_1 \cdot \mathbf{a}_3)$$

$$v_o = \rho^2(\mathbf{a}_2 \cdot \mathbf{a}_3)$$

$$\cos\theta = \frac{(\mathbf{a}_1 \times \mathbf{a}_3)\cdot(\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_1 \times \mathbf{a}_3| \cdot |\mathbf{a}_2 \times \mathbf{a}_3|}$$

Extracting camera parameters

See [FP], Sec. 1.3.1

$$\frac{M}{\rho} = \begin{pmatrix} \underbrace{\begin{matrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T \\ \dfrac{\beta}{\sin\theta}\boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T \\ \boldsymbol{r}_3^T \end{matrix}}_{A} & \underbrace{\begin{matrix} \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ t_z \end{matrix}}_{\mathbf{b}} \end{pmatrix}$$

Box 1

$$\mathrm{A} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ \mathbf{a}_3^{\mathrm{T}} \end{bmatrix} \quad \mathrm{b} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \mathrm{b}_3 \end{bmatrix}$$

Estimated values

Intrinsic

$$\alpha = \rho^2 |\mathbf{a}_1 \times \mathbf{a}_3| \sin\theta$$

$$\beta = \rho^2 |\mathbf{a}_2 \times \mathbf{a}_3| \sin\theta$$

Extracting camera parameters

See [FP], Sec. 1.3.1

$$\frac{M}{\rho} = \left(\begin{array}{c|c} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta}\boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{array}\right) = \mathrm{K}\begin{bmatrix} \mathrm{R} & \mathrm{T} \end{bmatrix}$$

$$\underbrace{\qquad}_{A} \qquad \underbrace{\qquad}_{\mathbf{b}}$$

Box 1

$$\mathrm{A} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ \mathbf{a}_3^{\mathrm{T}} \end{bmatrix} \qquad \mathrm{b} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \mathrm{b}_3 \end{bmatrix}$$

Estimated values

Extrinsic

$$\mathbf{r}_1 = \frac{(\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_2 \times \mathbf{a}_3|} \qquad \mathbf{r}_3 = \frac{\pm \mathbf{a}_3}{|\mathbf{a}_3|}$$

$$\mathbf{r}_2 = \mathbf{r}_3 \times \mathbf{r}_1 \qquad \mathrm{T} = \rho\, \mathrm{K}^{-1}\mathbf{b}$$

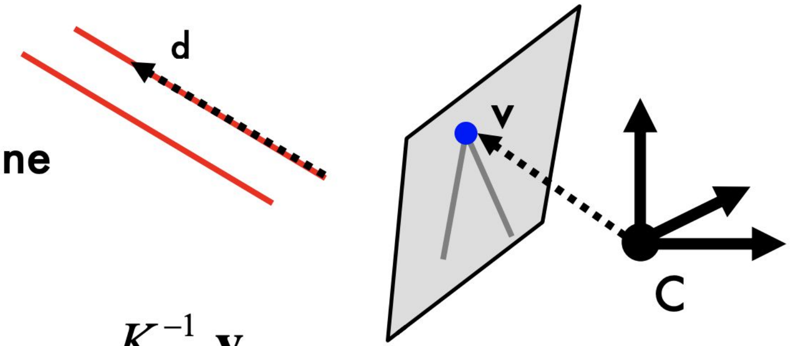
Lines in 3D

- Lines have 4 degrees of freedom - hard to represent in 3D-space

- Can be defined as intersection of 2 planes

$$\begin{aligned}\mathbf{d} &= \text{direction of the line} \\ &= [a, b, c]^T\end{aligned}$$

Vanishing points and directions



d = direction of the line
= [a, b, c]^T

$$\mathbf{v} = K\,\mathbf{d}$$

[Eq. 24]

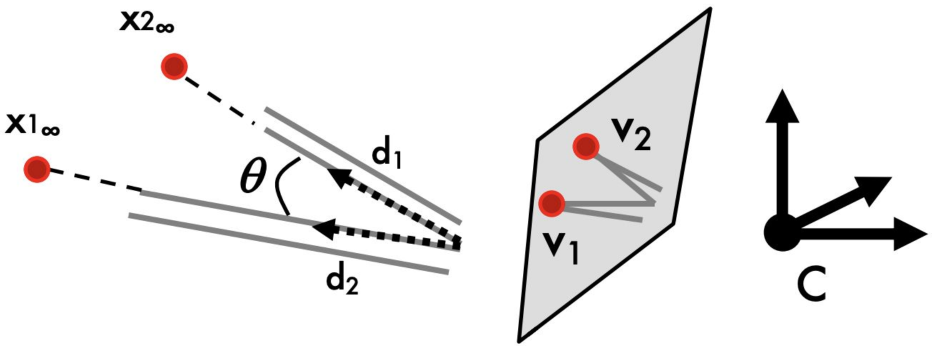
$$\mathbf{d} = \frac{K^{-1}\,\mathbf{v}}{\left\| K^{-1}\,\mathbf{v} \right\|}$$

[Eq. 25]

Proof:

$$X_\infty = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \xrightarrow{\mathrm{M}} \mathbf{v} = \mathbf{M}\,\mathbf{X}_\infty = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} = \mathbf{K}\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Angle between 2 vanishing points



$$\cos\theta = \frac{v_1^T \omega\, v_2}{\sqrt{v_1^T \omega\, v_1}\sqrt{v_2^T \omega\, v_2}}$$

[Eq. 28]

$$\omega = (K\, K^T)^{-1}$$

[Eq. 30]

If $\theta = 90 \quad \rightarrow \quad \boxed{v_1^T \omega\, v_2 = 0}$ [Eq. 29]

Scalar equation

Properties of ω

$$\boxed{\omega = (K\,K^T)^{-1}} \quad \text{[Eq. 30]}$$

$$M = K\begin{bmatrix} R & T \end{bmatrix}$$

1. $\omega = \begin{bmatrix} \omega_1 & \omega_2 & \omega_4 \\ \omega_2 & \omega_3 & \omega_5 \\ \omega_4 & \omega_5 & \omega_6 \end{bmatrix}$ symmetric and known up scale

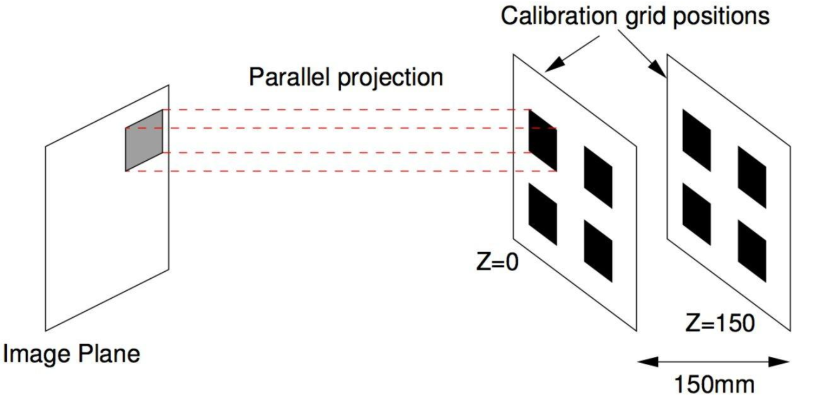
2. $\omega_2 = 0$ zero-skew

3. $\omega_2 = 0$, $\omega_1 = \omega_3$ square pixel

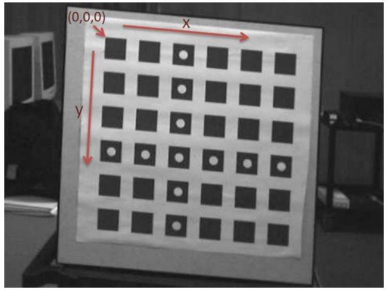
Problem Outline

- Q1: Projective Geometry
- Q2: Affine Camera Calibration
- Q3: Single View Geometry

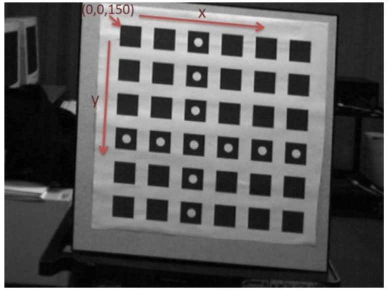
P2: Setup



(a) Image formation in an affine camera. Points are projected via parallel rays onto the image plane



(b) Image of calibration grid at Z=0



(c) Image of calibration grid at Z=150

P2: Affine Camera Model

(a) Given correspondences for the calibrating grid, solve for the camera parameters using Eq. 2. Note that each measurement $(x_i, y_i) \leftrightarrow (X_i, Y_i, Z_i)$ yields two linear equations for the 8 unknown camera parameters. Given N corner measurements, we have $2N$ equations and 8 unknowns. Using the given corner correspondences as inputs, complete the method `compute_camera_matrix()`. You will construct a linear system of equations and solve for the camera parameters to minimize the least-squares error. After doing so, you will return the 3×4 affine camera matrix composed of these computed camera parameters. **Explain your approach and include the camera matrix that you compute in the written report. [15 points for code + 5 for write-up]**

P2: Affine Camera Model

$$\begin{bmatrix} x \\ y \\ \boxed{1} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ \boxed{0 \quad 0 \quad 0 \quad 1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- **Linear**
- **8 Unknowns**

P2: Affine Camera Model

$$x = \begin{bmatrix} x_1 & ... & x_n \\ y_1 & ... & y_n \\ 1 & ... & 1 \end{bmatrix}, P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & 1 \end{bmatrix}, X = \begin{bmatrix} X_1 & ... & X_n \\ Y_1 & ... & Y_n \\ Z_1 & ... & Z_n \\ 1_1 & ... & 1_n \end{bmatrix}$$

$$x = PX$$

Solution 1

PM = p

P: (24, 4)

M: (4, 2)

p: (24, 2)

Recall from Lecture 3

Perspective Camera Model

$$p_i = \begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} \dfrac{\mathbf{m}_1\ \mathrm{P_i}}{\mathbf{m}_3\ \mathrm{P_i}} \\ \dfrac{\mathbf{m}_2\ \mathrm{P_i}}{\mathbf{m}_3\ \mathrm{P_i}} \end{bmatrix} = M\ P_i$$

in pixels

[Eq. 1]

$$\mathrm{M} = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \mathbf{m}_3 \end{bmatrix}$$

$$\mathrm{u_i} = \frac{\mathbf{m}_1\ \mathrm{P_i}}{\mathbf{m}_3\ \mathrm{P_i}} \rightarrow \mathrm{u_i}(\mathbf{m}_3\ \mathrm{P_i}) = \mathbf{m}_1\ \mathrm{P_i} \rightarrow \mathrm{u}_i(\mathbf{m}_3\ \mathrm{P}_i) - \mathbf{m}_1\ \mathrm{P}_i = 0$$

$$\mathrm{v_i} = \frac{\mathbf{m}_2\ \mathrm{P_i}}{\mathbf{m}_3\ \mathrm{P_i}} \rightarrow \mathrm{v_i}(\mathbf{m}_3\ \mathrm{P_i}) = \mathbf{m}_2\ \mathrm{P_i} \rightarrow \mathrm{v}_i(\mathbf{m}_3\ \mathrm{P}_i) - \mathbf{m}_2\ \mathrm{P}_i = 0$$

[Eqs. 2]

$$\begin{cases} u_1(\mathbf{m}_3\ P_1) - \mathbf{m}_1\ P_1 = 0 \\ v_1(\mathbf{m}_3\ P_1) - \mathbf{m}_2\ P_1 = 0 \\ \vdots \\ u_i(\mathbf{m}_3\ P_i) - \mathbf{m}_1\ P_i = 0 \\ v_i(\mathbf{m}_3\ P_i) - \mathbf{m}_2\ P_i = 0 \\ \vdots \\ u_n(\mathbf{m}_3\ P_n) - \mathbf{m}_1\ P_n = 0 \\ v_n(\mathbf{m}_3\ P_n) - \mathbf{m}_2\ P_n = 0 \end{cases}$$

[Eqs. 3]

Recall from Lecture 3

$$\begin{cases} -u_1(\mathbf{m}_3 P_1) + \mathbf{m}_1 P_1 = 0 \\ -v_1(\mathbf{m}_3 P_1) + \mathbf{m}_2 P_1 = 0 \\ \vdots \\ -u_n(\mathbf{m}_3 P_n) + \mathbf{m}_1 P_n = 0 \\ -v_n(\mathbf{m}_3 P_n) + \mathbf{m}_2 P_n = 0 \end{cases}$$

[Eqs. 3]

$$\longrightarrow \boxed{\underbrace{\mathbf{P}}_{\text{known}}\ \underbrace{\mathbf{m}}_{\text{unknown}} = 0}$$

[Eq. 4]

Homogenous linear system

$$\mathbf{P} \stackrel{\text{def}}{=} \begin{pmatrix} \mathrm{P}_1^T & \mathbf{0}^T & -u_1\mathrm{P}_1^T \\ \mathbf{0}^T & \mathrm{P}_1^T & -v_1\mathrm{P}_1^T \\ & \vdots & \\ \mathrm{P}_n^T & \mathbf{0}^T & -u_n\mathrm{P}_n^T \\ \mathbf{0}^T & \mathrm{P}_n^T & -v_n\mathrm{P}_n^T \end{pmatrix}_{2n \times 12}$$

1x4

$$\boldsymbol{m} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{m}_1^{\mathrm{T}} \\ \mathbf{m}_2^{\mathrm{T}} \\ \mathbf{m}_3^{\mathrm{T}} \end{pmatrix}_{12\mathrm{x}1}$$

4x1

Solution 2

PM = p

P: (48, 8)

M: (8,)

p: (48,)

P2: Affine Camera Model

numpy.linalg.lstsq

`linalg.lstsq(a, b, rcond='warn')` [source]

Return the least-squares solution to a linear matrix equation.

Computes the vector *x* that approximately solves the equation `a @ x = b`. The equation may be under-, well-, or over-determined (i.e., the number of linearly independent rows of *a* can be less than, equal to, or greater than its number of linearly independent columns). If *a* is square and of full rank, then *x* (but for round-off error) is the "exact" solution of the equation. Else, *x* minimizes the Euclidean 2-norm $||b - ax||$. If there are multiple minimizing solutions, the one with the smallest 2-norm $||x||$ is returned.

Solve for M using np.linalg.lstsq or np.linalg.pinv
https://numpy.org/doc/stable/reference/generated/numpy.linalg.lstsq.html
https://numpy.org/doc/stable/reference/generated/numpy.linalg.pinv.html

Once you got the values for M, be careful about how to reconstruct the final camera matrix!

P2: Affine Camera Model

(b) After finding the calibrated camera matrix, you will compute the RMS error between the given N image corner coordinates and N corresponding calculated corner locations in `rms_error()`. Recall that

$$\text{RMS}_{\text{total}} = \sqrt{\sum((x - x')^2 + (y - y')^2)/N}$$

Compute the RMS error for the camera matrix that you found in part (a). **Include the RMS error that you compute in the written report. [10 points for code]**

Reproject the scene points onto the image plane and calculate the error.
N is the number of points.

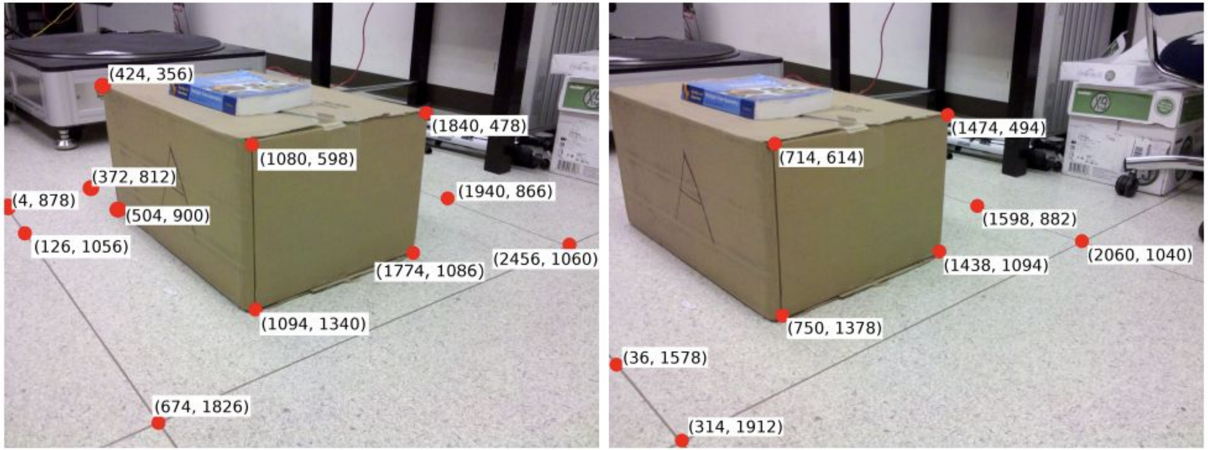
(c) Could you calibrate the matrix with only one checkerboard image? **Explain briefly in one or two sentences. [5 points]**

Problem Outline

- Q1: Projective Geometry
- Q2: Affine Camera Calibration
- Q3: Single View Metrology

P3: Single View Metrology

In this question, we will estimate camera parameters from a single view and leverage the projective nature of cameras to find both the camera center and focal length from vanishing points present in the scene above.



(a) Image 1 (1.jpg) with marked pixels (b) Image 2 (2.jpg) with marked pixels

Figure 2: Marked pixels in images taken from different viewpoints.

P3: Single View Metrology

(a) In Figure 2, we have identified a set of pixels to compute vanishing points in each image. Please complete `compute_vanishing_point()`, which takes in these two pairs of points on parallel lines to find the vanishing point. You can assume that the camera has zero skew and square pixels, with no distortion. **[5 points]**

- Points in L_1: (x_1, y_1), (x_2, y_2) → slope: $m_1 = (y_2 - y_1)/(x_2 - x_1)$
- Points in L_2: (x_3, y_3), (x_4, y_4) → slope: $m_2 = (y_4 - y_3)/(x_4 - x_3)$
- Intersection of L_1 and L_2: Vanishing Point

```
v1 = compute_vanishing_point(np.array([[1080, 598],[1840, 478],[1094,1340],[1774,1086]]))
v2 = compute_vanishing_point(np.array([[674,1826],[4, 878],[2456,1060],[1940,866]]))
v3 = compute_vanishing_point(np.array([[1094,1340],[1080,598],[1774,1086],[1840,478]]))

v1b = compute_vanishing_point(np.array([[314,1912],[2060,1040],[750,1378],[1438,1094]]))
v2b = compute_vanishing_point(np.array([[314,1912],[36,1578],[2060,1040],[1598,882]]))
v3b = compute_vanishing_point(np.array([[750,1378],[714,614],[1438,1094],[1474,494]]))
```

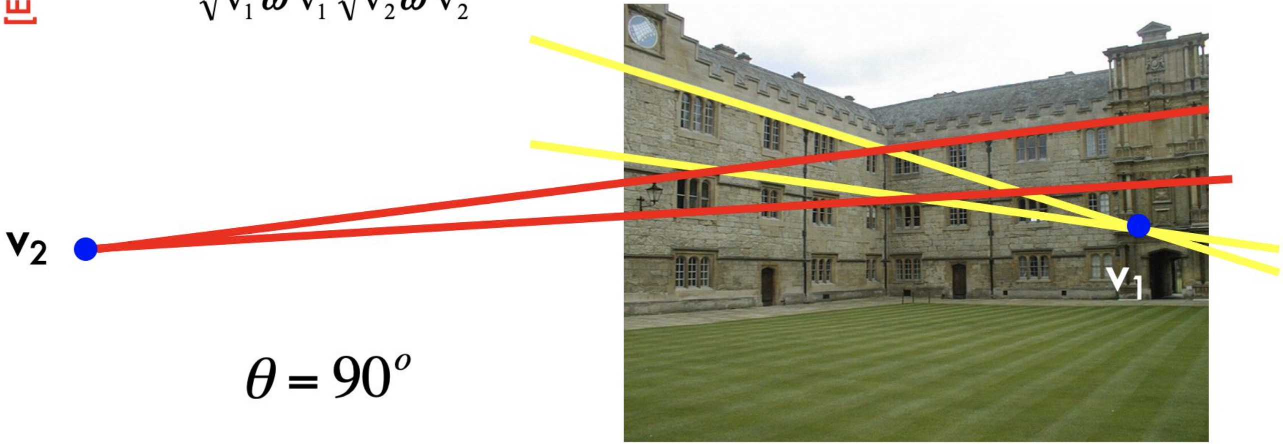
P3: Single View Metrology

(b) Using three vanishing points, we can compute the intrinsic camera matrix used to take the image. Do so in `compute_K_from_vanishing_points()`. [**10 points**]

Single view calibration - example

[Eq. 28]

$$\cos\theta = \frac{v_1^T \boldsymbol{\omega}\, v_2}{\sqrt{v_1^T \boldsymbol{\omega}\, v_1}\sqrt{v_2^T \boldsymbol{\omega}\, v_2}}$$



$$\theta = 90^o$$

[Eq. 29]

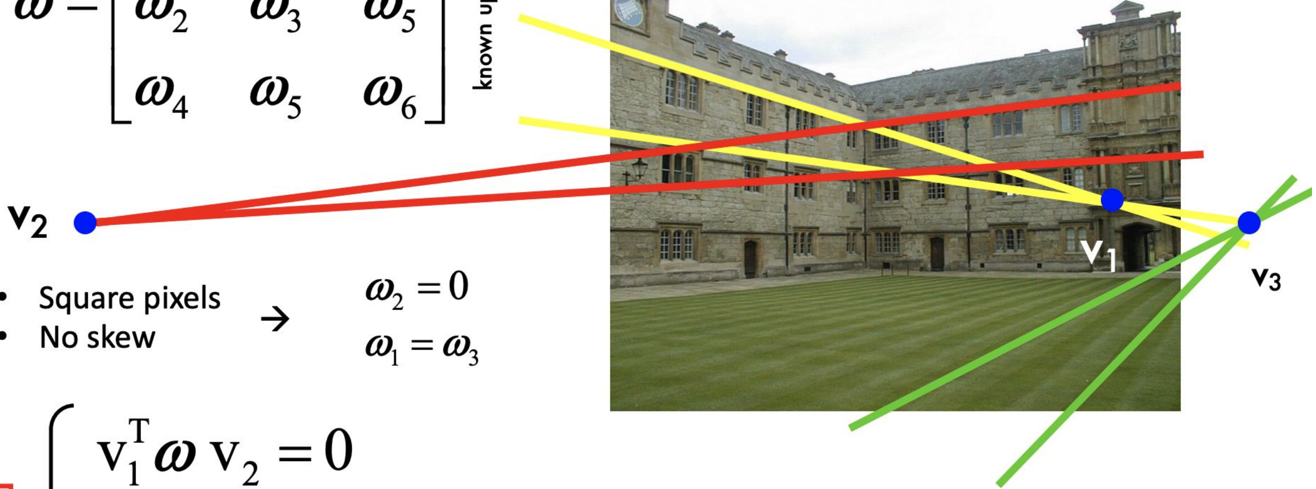
$$\begin{cases} v_1^T \boldsymbol{\omega}\, v_2 = 0 \\ \omega = (K\,K^T)^{-1} \end{cases}$$

Do we have enough constraints to estimate K?
K has 5 degrees of freedom and Eq.29 is a scalar equation ☹

Single view calibration - example

$$\omega = \begin{bmatrix} \omega_1 & \omega_2 & \omega_4 \\ \omega_2 & \omega_3 & \omega_5 \\ \omega_4 & \omega_5 & \omega_6 \end{bmatrix}$$

known up to scale



- Square pixels
- No skew

→ $\omega_2 = 0$

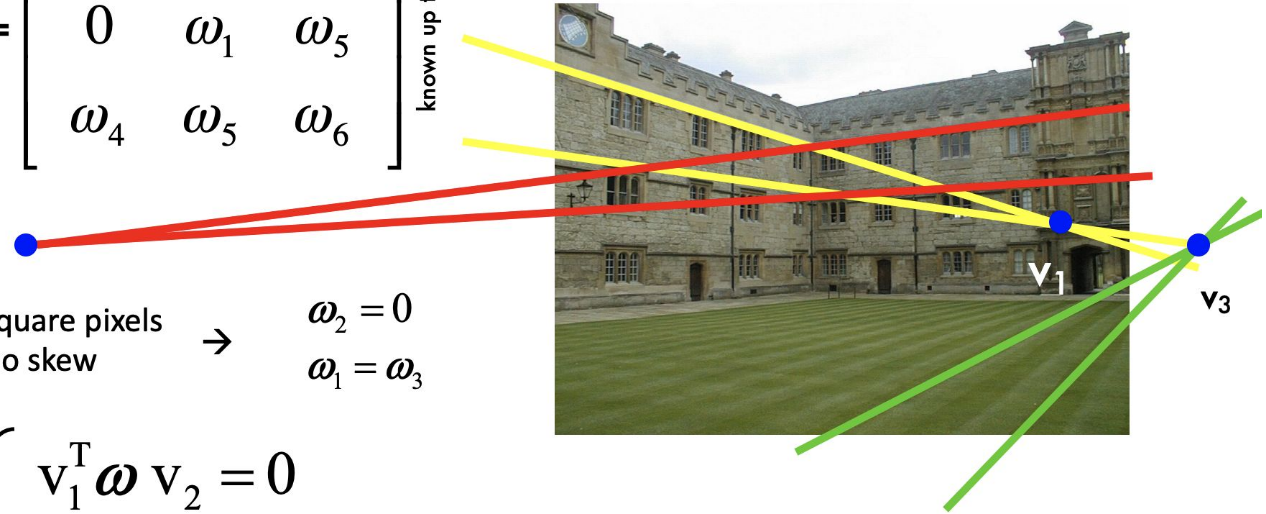
$\omega_1 = \omega_3$

[Eqs. 31]

$$\begin{cases} v_1^T \omega \, v_2 = 0 \\ v_1^T \omega \, v_3 = 0 \\ v_2^T \omega \, v_3 = 0 \end{cases}$$

Single view calibration - example

$$\omega = \begin{bmatrix} \omega_1 & 0 & \omega_4 \\ 0 & \omega_1 & \omega_5 \\ \omega_4 & \omega_5 & \omega_6 \end{bmatrix}$$ known up to scale



- Square pixels
- No skew

→ $\omega_2 = 0$, $\omega_1 = \omega_3$

[Eqs. 31]

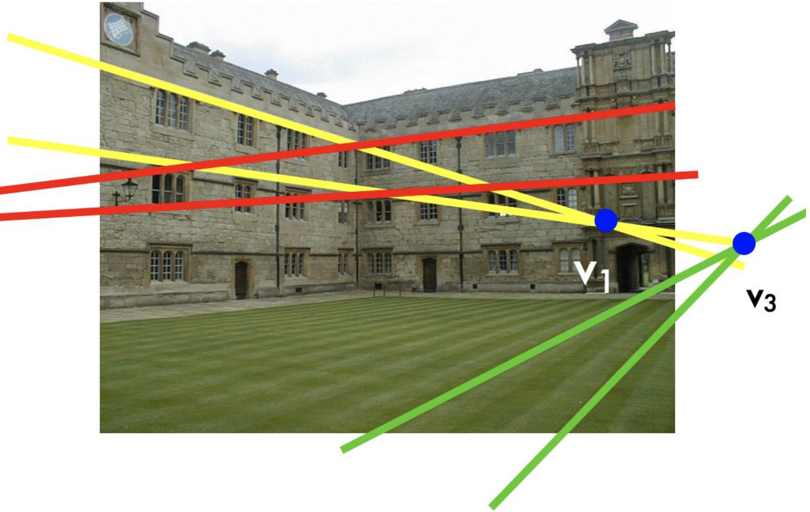
$$\begin{cases} v_1^T \boldsymbol{\omega}\, v_2 = 0 \\ v_1^T \boldsymbol{\omega}\, v_3 = 0 \\ v_2^T \boldsymbol{\omega}\, v_3 = 0 \end{cases}$$

→ Compute ω !

Single view calibration - example

$$\omega = \begin{bmatrix} \omega_1 & 0 & \omega_4 \\ 0 & \omega_1 & \omega_5 \\ \omega_4 & \omega_5 & \omega_6 \end{bmatrix}$$

v_2



- Square pixels
- No skew

$\rightarrow$ $\omega_2 = 0$, $\omega_1 = \omega_3$

[Eqs. 31]

$$\begin{cases} v_1^T \boldsymbol{\omega}\, v_2 = 0 \\ v_1^T \boldsymbol{\omega}\, v_3 = 0 \\ v_2^T \boldsymbol{\omega}\, v_3 = 0 \end{cases}$$

Once ω is calculated, we get K:

$$\omega = (K\ K^T)^{-1} \longrightarrow \mathrm{K}$$

(Cholesky factorization; HZ pag 582)

P3: Single View Metrology

[Eqs. 31]

$$\begin{cases} v_1^T \omega \, v_2 = 0 \\ v_1^T \omega \, v_3 = 0 \\ v_2^T \omega \, v_3 = 0 \end{cases}$$

Aw = 0
A is a 3 x 4 matrix
w is a 4 x 1 vector
w is a null vector of A

- **Solve for w with SVD**
  - **A = UDV^T**
  - **Last column of V is the solution for w**
- **Use Cholesky Factorization to get K**

P3: Single View Metrology

(c) Is it possible to compute the camera intrinsic matrix for any set of vanishing points? Similarly, is three vanishing points the minimum required to compute the intrinsic camera matrix? **Justify your answer. [5 points]**

(d) The method used to obtain vanishing points is approximate and prone to noise. **Discuss what possible sources of noise could be, and select a pair of points from the image that are a good example of this source of error. [5 points]**

P3: Compute Angle Between Planes

(e) This process gives the camera internal matrix under the specified constraints. For the remainder of the computations, use the following internal camera matrix:

$$K = \begin{bmatrix} 2448 & 0 & 1253 \\ 0 & 2438 & 986 \\ 0 & 0 & 1 \end{bmatrix}$$

Use the vanishing lines we provide in the starter code to verify numerically that the ground plane is orthogonal to the plane front face of the box in the image. Fill out the method `compute_angle_between_planes()` and **include a brief description of your solution and your computed angle in your report. [8 points for code + 2 points for write-up]**

P3: Compute Angle Between Planes

- Vanishing lines L_1 and L_2
- $L_1 = v_1 \times v_2$; $L_2 = v_3 \times v_4$
  - v_1 *and* v_2 = vanishing points corresponding to one plane
  - v_3 *and* v_4 for the other plane

$$\boxed{cos\theta = \frac{\mathbf{l}_1^T \omega^* \mathbf{l}_2}{\sqrt{\mathbf{l}_1^T \omega^* \mathbf{l}_1}\sqrt{\mathbf{l}_2^T \omega^* \mathbf{l}_2}}}$$

, where $\omega^* = \omega^{-1} = KK^T$

See course notes for more information

P4: Compute Rotation Matrix

(f) Assume the camera rotates but no translation takes place. Assume the internal camera parameters remain unchanged. An Image 2 of the same scene is taken. Use vanishing points to estimate the rotation matrix between when the camera took Image 1 and Image 2. Fill out the method `compute_rotation_matrix_between_cameras()` and **submit a brief description of your approach and your results in the written report. [8 points for code + 2 points for write-up]**

P4: Compute Rotation Matrix

- Find corresponding vanishing points from both images (v_1, v_2, v_3) and (v_1', v_2', v_3')
- Calculate directions of vanishing points:
  - $v = K\,d \rightarrow$ $$\boxed{\mathbf{d} = \frac{K^{-1}\,\mathbf{v}}{\left\|K^{-1}\,\mathbf{v}\right\|}}$$
- $d_i' = R\,d_i$, where
  - d_i' = direction of the i^{th} vanishing point in second image
  - d_i = direction of the i^{th} vanishing point in first image

Project Proposal

The project proposal should be up to 2 pages and should contain the following:

- What is the problem that you will be investigating? Why is it interesting?
- What reading will you examine to provide context and background? Additionally, has is some prior work related to this problem? Please provide at least 2 specific citations.
- What method or algorithm are you proposing? If there are existing implementations, will you use them and how? How do you plan to improve or modify such implementations?
- What data will you use, if any? If you are collecting new datasets, how do you plan to collect them?
- How will you evaluate your results? Qualitatively, what kind of results do you expect (e.g. plots or figures)? Quantitatively, what kind of analysis will you use to evaluate and/or compare your results (e.g. what performance metrics or statistical tests)?
- By what dates will you complete certain parts of your project? List specific goals for the midterm progress report.

We highly recommend submitting a project proposal and talking to the course staff about your proposed project throughout the quarter. Generally, we find that students who do this end up with very strong, and even publishable, final projects.

If your proposed project is joint with another class' project (with the consent of the other class' instructor), make this clear in the proposal.

Project Proposal

- Maximum of 2 pages
- Submit the proposal as a PDF document through Gradescope
- Include the following sections:
    - Title and authors
    - Introduction
    - Prior Work
    - Dataset
    - Technical Approach
    - Evaluation Metrics
    - Milestones (dates and sub-goals)
    - References

Project Proposal

- Due 11:59 PM April 25
- Potential Topics
  - Review Last Week's CA Section
  - Come to office hours and discuss your ideas
- If you haven't found a team, check out the Team Forming Thread on Ed
- We have published the Project Reports from last year
  - https://web.stanford.edu/class/cs231a/projects2022
- Good Project Proposals from last year
  - You can find them in the latest Canvas announcement

Thanks!

Questions