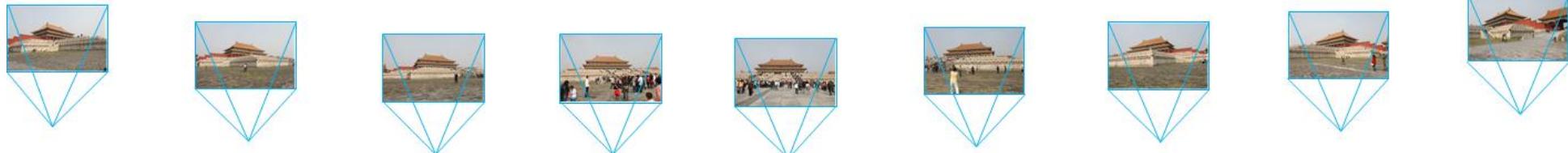
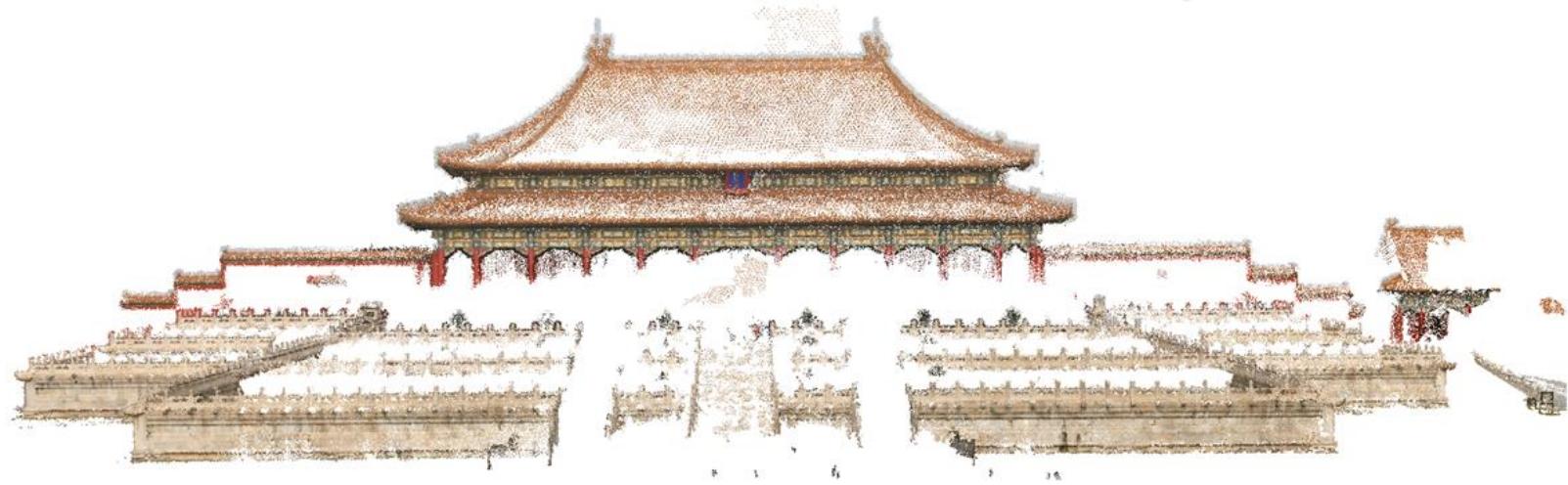


10. Camera Calibration & 2-View Geometry



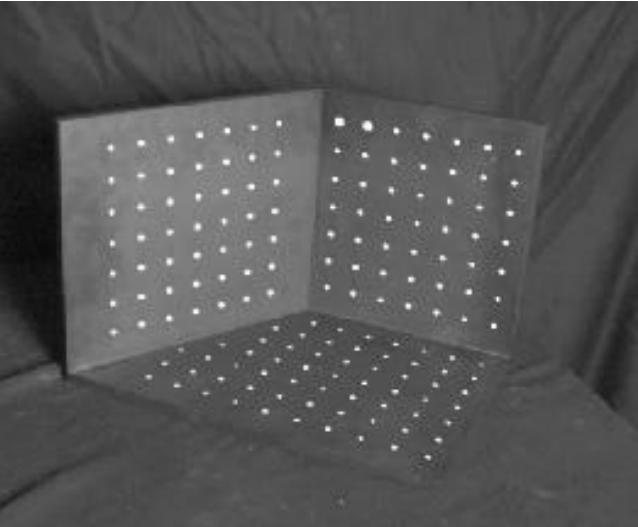
Outline



- Camera Calibration
- Two-view Geometry
- Triangulation

Camera Calibration

- Place a known object in the scene
 - identify correspondence between image and scene
 - compute mapping from scene to image



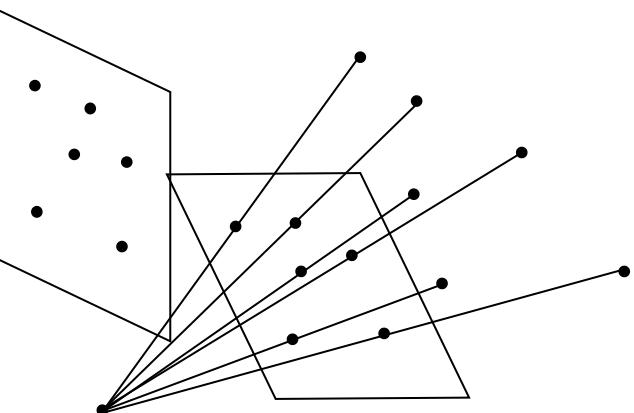
Issues

- must know geometry very accurately
- must know 3D->2D correspondence

Resectioning

Given some 3D points X_i with known coordinates and their image projections x_i , how do we compute the camera matrix P ?

$$X_i \leftrightarrow x_i \rightarrow P ?$$



warning: in practice, it is difficult to obtain 3D points of known positions



Basic Equations

The equations we have:

$$\mathbf{x}_i = \lambda P \mathbf{X}_i$$

Get rid of the λ by cross product

$$\mathbf{x}_i \times P \mathbf{X}_i = 0$$

Change cross product to its matrix form:

$$[\mathbf{x}_i]_x P \mathbf{X}_i = 0$$

Definition: $\vec{a} \times \vec{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$

$$[a]_x = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

Rewrite the formula,
we get 3 equations (2 of them are independent) about P from each pair of $\mathbf{X}_i \leftrightarrow \mathbf{x}_i$

$$\begin{bmatrix} \mathbf{0}^\top & -w_i \mathbf{X}_i^\top & y_i \mathbf{X}_i^\top \\ w_i \mathbf{X}_i^\top & \mathbf{0}^\top & -x_i \mathbf{X}_i^\top \\ -y_i \mathbf{X}_i^\top & x_i \mathbf{X}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{pmatrix} = \mathbf{0}$$

Eventually, we obtain a linear equation:

$$A \mathbf{p} = \mathbf{0}$$



Direct Linear Transform (DLT)

$$Ap = 0$$

Minimal solution

P has 11 dof, 2 independent eq./points

$\Rightarrow 5\frac{1}{2}$ correspondences needed (say 6)

Over-determined solution

$n \geq 6$ points

minimize $\|Ap\|$ subject to constraint

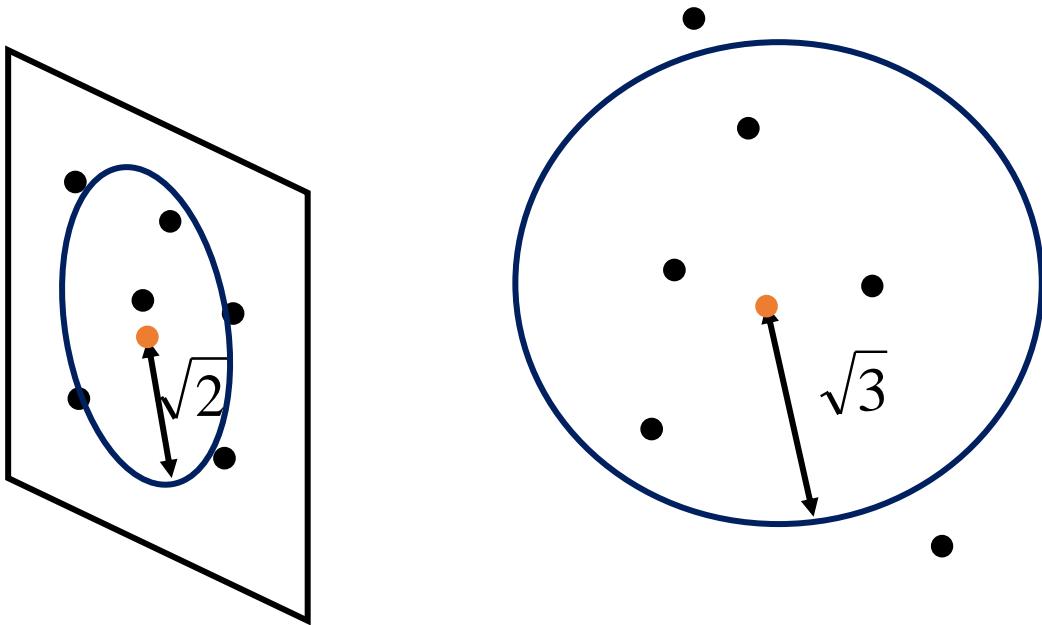
$$\|p\| = 1$$

$$\|m^3\| = 1$$

$$P = \begin{matrix} & \\ & \\ m^3 & \end{matrix}$$

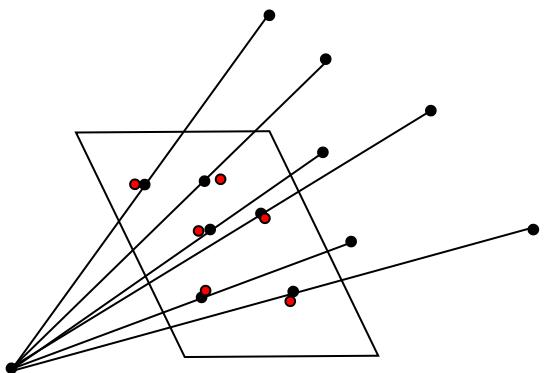
Data Normalization

Simple,
translate and scale the 2D and 3D points
make them to center at origin and with radius of $\sqrt{2}$ or $\sqrt{3}$



Geometric Error

- DLT minimizes the algebraic distance
- It is better to minimize the Euclidean distance between the projected positions and 2D points (geometric distance)
 - Assume 3D points are precisely known.
 - Noises only present in image measurements!



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

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



Geometric Error

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

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

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

$$\sum_i \left| u_i - \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}} \right|^2 + \left| v_i - \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}} \right|^2$$

$$= \sum_i d^2(x_i, \hat{x}_i)$$



Geometric Error vs Direct Linear Transform

The DLT method can be derived from a different way:

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

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

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

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

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

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



Geometric Error vs Direct Linear Transform

Notice the DLT method minimizes a different error

$$\begin{aligned} & \sum_i |u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) - (m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03})|^2 \\ & + \sum_i |v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) - (m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13})|^2 \\ & = \sum_i \hat{w}_i^2 d^2(x_i, \hat{x}_i) \end{aligned}$$

$$\hat{w}_i = ||m^3|| \text{depth}(X, P)$$

$$P = \begin{array}{c} \text{light green rectangle} \\ \text{orange rectangle labeled } m^3 \end{array}$$

The DLT minimizes a weighted geometric error,
where faraway points have larger weights



Gold Standard Algorithm

Objective

Given $n \geq 6$ 3D to 2D point correspondences $\{\mathbf{X}_i \leftrightarrow \mathbf{x}_i\}$, determine the Maximum Likelihood Estimation of \mathbf{P} (assuming 3D points are precisely known)

Algorithm

(i) Linear solution:

(a) Normalization:

$$\tilde{\mathbf{X}}_i = \mathbf{U}\mathbf{X}_i \quad \tilde{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i$$

(b) DLT:

$$\mathbf{A}\tilde{\mathbf{p}} = 0$$

(ii) Minimization of geometric error: using the linear estimate as a starting point minimize the geometric error:

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

(iii) Denormalization:

$$\mathbf{P} = \mathbf{T}^{-1}\tilde{\mathbf{P}}\mathbf{U}$$



Restricted Camera Estimation

Sometimes, we have partial knowledge of the camera matrix

- skew s is zero
- pixels are square $\alpha = 1$, i.e., $f_x = f_y$
- principal point is known, $c_x = c_y = 0$
- complete camera matrix \mathbf{K} is known (e.g. focal length known from EXIF)

$$\mathbf{K} = \begin{bmatrix} -f & s & c_x \\ 0 & -\alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$



Restricted Camera Estimation

Initialization

- Use general DLT
- Clamp values to desired values, e.g., $s = 0$, $f_x = f_y$

Note: can sometimes cause big jump in error

Alternative initialization

- Use general DLT
- Impose soft constraints
- gradually increase weights

$$\sum_i d(\mathbf{x}_i, \mathbf{P}\mathbf{X}_i)^2 + ws^2 + w(f_x - f_y)^2$$

Minimize the geometric error to refine

Questions?





PnP: Perspective-n-Points

- PnP: determine camera extrinsics when intrinsics are known
- How many pairs of $X_i \leftrightarrow x_i$ do we need?
 - Each pair provides 2 constraints
 - There are in total 6 dof (3 for R and 3 for t)
- The PnP algorithm ($n \geq 3$)
 - From n pairs of $X_i \leftrightarrow x_i$, first solve 3D positions of x_i in camera frame
 - Then solve R, t from 3D \leftrightarrow 3D correspondences
 - Between camera frame and world frame
 - Solution easy to derive (try it yourself)



P3P Algorithm

- Assume $\|c - X_i\| = d_i, \|c - X_j\| = d_j, \|X_i - X_j\| = d_{ij}$

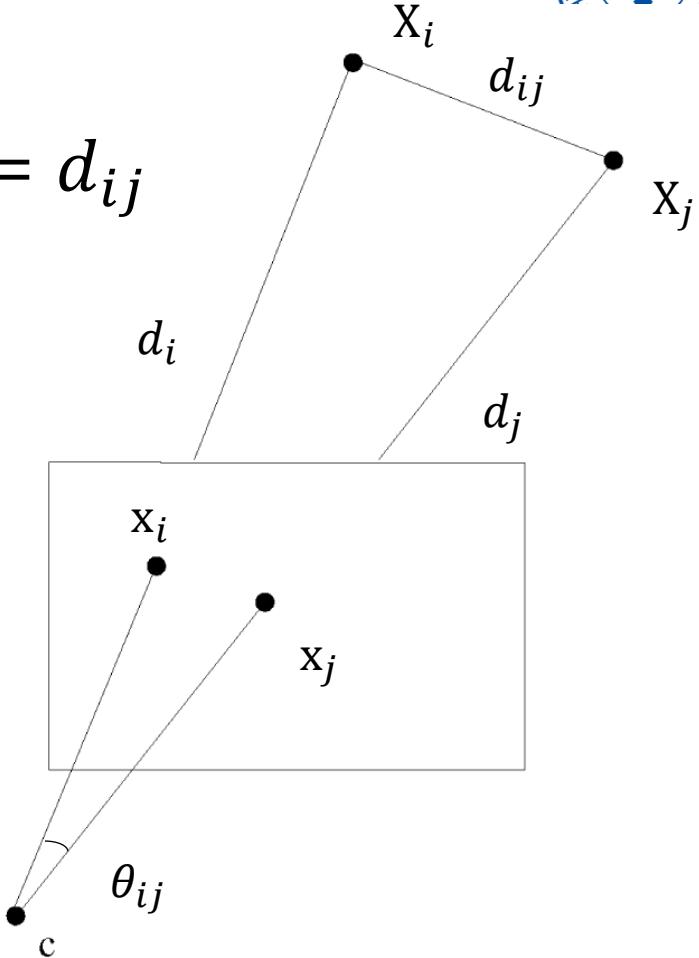
$$d_{ij}^2 = d_i^2 + d_j^2 - 2d_i d_j \cos \theta_{ij}$$

- The angle θ_{ij} is known from the intrinsics
- Denote $f_{ij}(d_i, d_j) = d_i^2 + d_j^2 - 2d_i d_j \cos \theta_{ij} - d_{ij}^2$
- For 3 points, we can solve d_1, d_2, d_3 from

$$\begin{cases} f_{12}(d_1, d_2) = 0 \\ f_{23}(d_2, d_3) = 0 \\ f_{13}(d_1, d_3) = 0 \end{cases}$$

- After eliminating d_2, d_3 , obtain a 4-th order equation of $d_1^2 \triangleq x$

$$g(x) = a_5 x^4 + a_4 x^3 + a_3 x^2 + a_2 x + a_1 = 0$$
- 4 solutions exist (more points needed for uniqueness)





Linear PnP Algorithm

- For $n \geq 5$, pick any point X_i , and another two points
 - Obtain a 4-th order polynomial of d_i ($d_i^2 \triangleq x$)

$$g(x) = a_5^i x^4 + a_4^i x^3 + a_3^i x^2 + a_2^i x + a_1^i = 0$$

- There are $(n - 1)(n - 2)/2$ such polynomials in total
- Stacking all such equations

$$\begin{pmatrix} a_5^i & a_4^i & a_3^i & a_2^i & a_1^i \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} x^4 \\ x^3 \\ x^2 \\ x \\ 1 \end{pmatrix} = 0$$

- Solving d_i linearly, and repeat for all X_i



EPnP Algorithm

- The linear algorithm in previous slice is $O(n^5)$
- EPnP is $O(n)$
 - Also first computes 3D points in the camera frame

EPnP: An Accurate $O(n)$ Solution to the PnP Problem

Vincent Lepetit · Francesc Moreno-Noguer · Pascal Fua

IJCV 2008

Questions?



Calibration from Vanishing Points

- Scenes contain parallel lines, which intersect at vanishing points

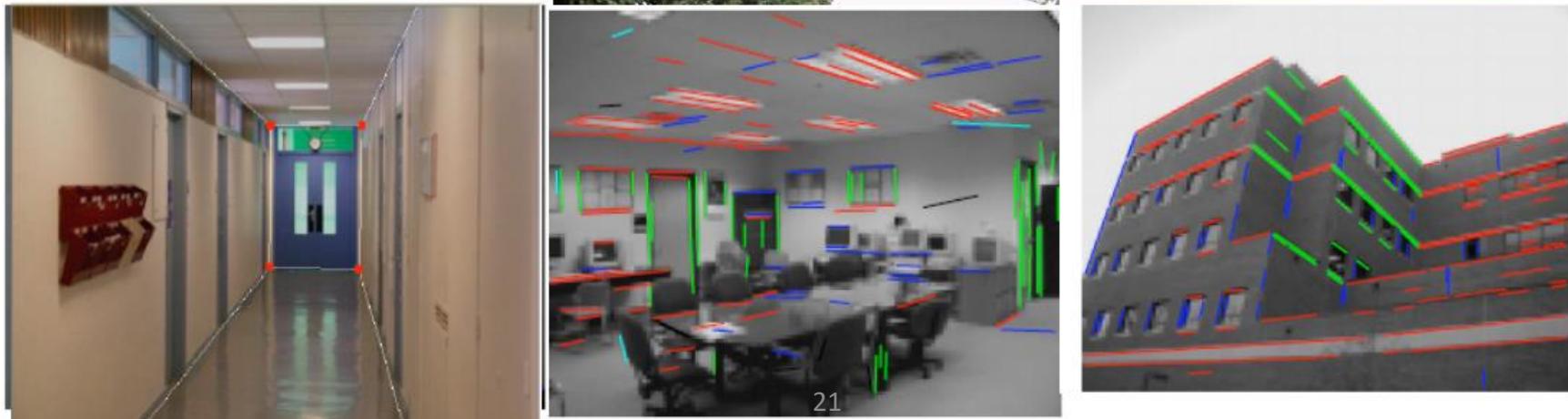
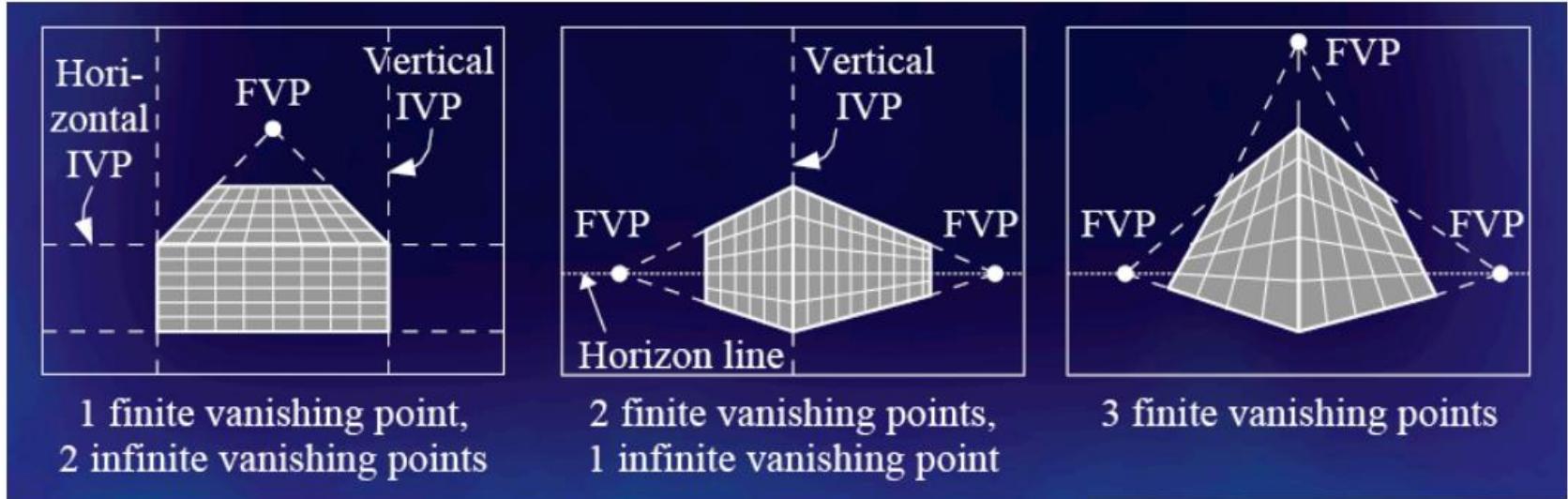




Image of Absolute Conic (IAC)

- A 3D direction e , e.g. $e = [0,0,1]^T$
- Its vanishing point is projected at

$$\begin{aligned} v &= K[R \ t] \begin{bmatrix} e \\ 0 \end{bmatrix} = KRe \\ \Rightarrow \quad e &= R^T K^{-1} v \end{aligned}$$

- For two perpendicular 3D directions e_i, e_j

$$\begin{aligned} 0 &= e_i^T e_j = (R^T K^{-1} v_i)^T (R^T K^{-1} v_j) \\ &= v_i^T K^{-T} R R^T K^{-1} v_j \\ &= v_i^T \underline{K^{-T} K^{-1}} v_j \end{aligned}$$

→ image of the absolute conic
(IAC), denoted by ω

Calibration from Vanishing Points

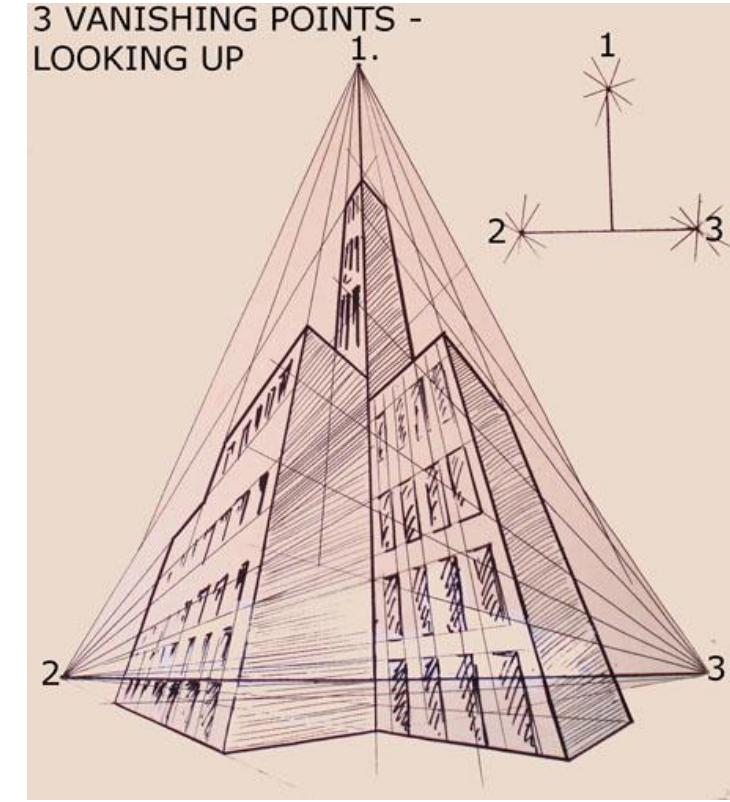
- From 3 known vanishing points v_i, v_j, v_k
 - Assume their directions are mutual orthogonal
- Obtain three equations:

$$\begin{aligned}v_i^T \omega v_j &= 0 \\v_j^T \omega v_k &= 0 \\v_i^T \omega v_k &= 0\end{aligned}$$

- Assume simple intrinsics:

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow K^{-1} = \begin{bmatrix} 1/f & 0 & -u_0/f \\ 0 & 1/f & -v_0/f \\ 0 & 0 & 1 \end{bmatrix}$$

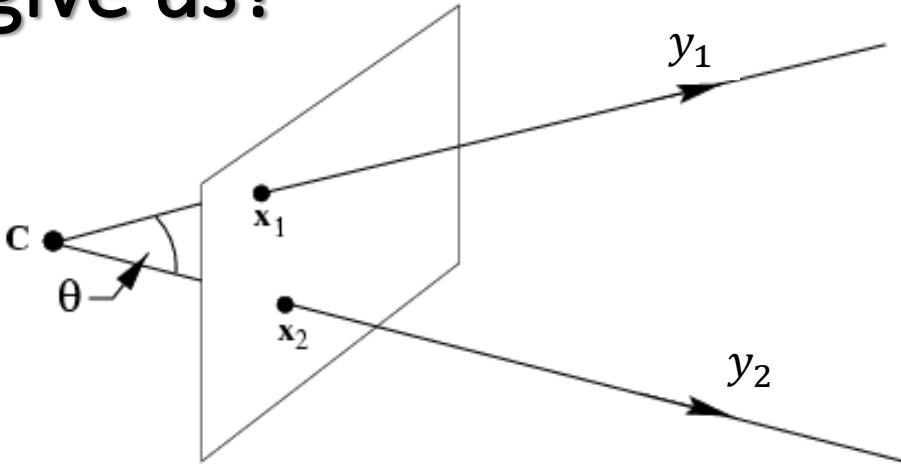
- Solve for f, u_0, v_0 from the three equations



what does the intrinsics give us?

$$x = K[I|0] \begin{bmatrix} y \\ 0 \end{bmatrix}$$

$$\Rightarrow y = K^{-1}x$$



We can recover the 3D ray defined by the camera center and a pixel, if the camera is calibrated.

We can further measure the angle between two such rays.

$$\cos \theta = \frac{y_1^T y_2}{\sqrt{(y_1^T y_1)(y_2^T y_2)}} = \frac{x_1^T (K^{-T} K^{-1}) x_2}{\sqrt{(x_1^T (K^{-T} K^{-1}) x_1)(x_2^T (K^{-T} K^{-1}) x_2)}}$$

image of the absolute conic (IAC)

Questions?





The Circular Points on a 2D Plane

The circular points I, J are two special points at infinity

$$I = \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} \quad J = \begin{pmatrix} 1 \\ -i \\ 0 \end{pmatrix} \quad i = \sqrt{-1}$$

“circular points”: any circle must pass through these two points

Intersect a circle with the line at infinity

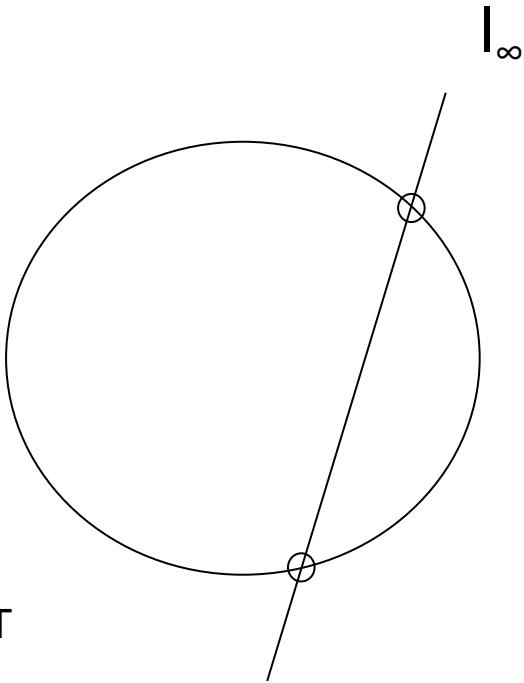
$$\begin{aligned} x_1^2 + x_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 &= 0 \\ x_3 &= 0 \end{aligned}$$

It leads to an equation with **two solutions**

$$x_1^2 + x_2^2 = 0 \quad \rightarrow \quad I = (1, i, 0)^T \quad J = (1, -i, 0)^T$$

Fit a general conic needs five points

Fit a general circle only needs three points (because we implicitly used these two circle points)





The Absolute Conic in 3D Space

- Consider the 3D space consists of many 2D planes (layer by layer).
- Each plane has two circular points.
- All these points form a conic, called the absolute conic Ω_∞ !
 - All points satisfying a conic equation.

$$X_4 = 0;$$

$$(X_1, X_2, X_3) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} = 0 \quad \Omega_\infty = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Any sphere must pass through Ω_∞ !
- Ω_∞ lies on the plane at infinity π_∞ !

Image of Absolute Conic (IAC)

- Why Ω_∞ matters?
- It allows measurement of angles (even after projective transformations)!

$$\cos \theta = \frac{(y_1^T \Omega_\infty y_2)}{\sqrt{(y_1^T \Omega_\infty y_1)(y_2^T \Omega_\infty y_2)}}$$

$(y_1, 0)$ is the vanishing point of a 3D direction

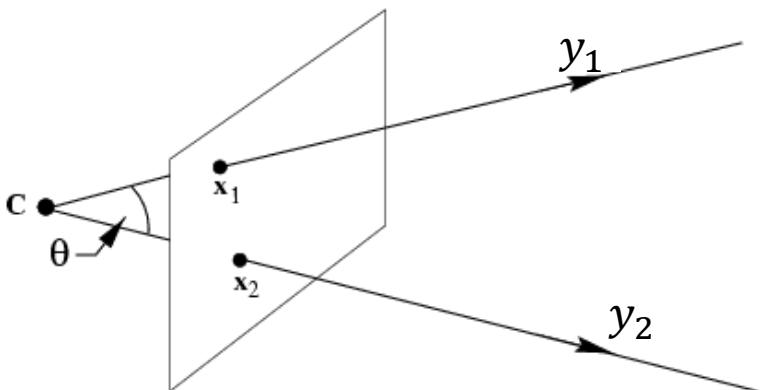
- The angle between two camera rays:

$$\cos \theta = \frac{y_1^T y_2}{\sqrt{(y_1^T y_1)(y_2^T y_2)}} = \frac{x_1^T (K^{-T} K^{-1}) x_2}{\sqrt{(x_1^T (K^{-T} K^{-1}) x_1)(x_2^T (K^{-T} K^{-1}) x_2)}}$$

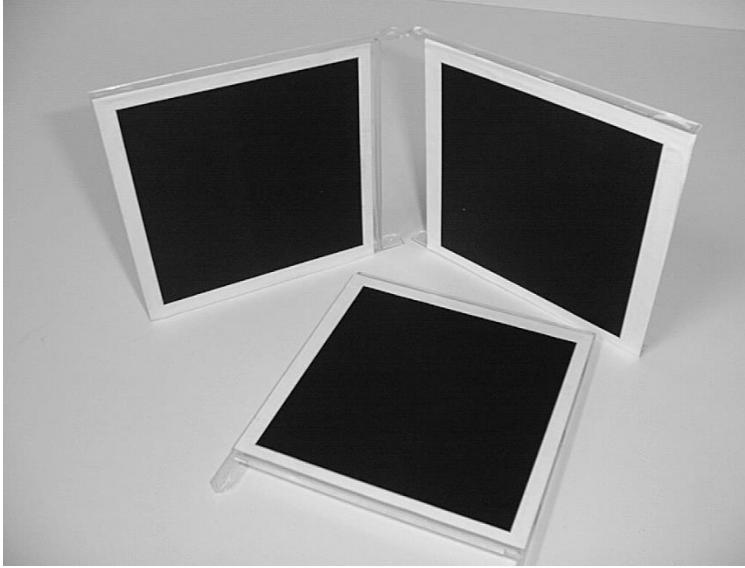
Consider the mapping from π_∞ to the image plane.

$K^{-T} K^{-1}$ is the image of Ω_∞ after mapping!

It allows us to measure angles between rays.



Zhang's Calibration Method



- (i) compute H for each square; mapping the checkboard to the image plane
(assuming corners are $(0,0), (1,0), (0,1), (1,1)$ in the checkboard plane)
- (ii) compute the imaged circular points $H(1, \pm i, 0)^T$
(3D translation of the checkboard does not move its circular points)
- (iii) fit a conic to 6 circular points
- (iv) compute K from ω through Cholesky factorization

Questions?



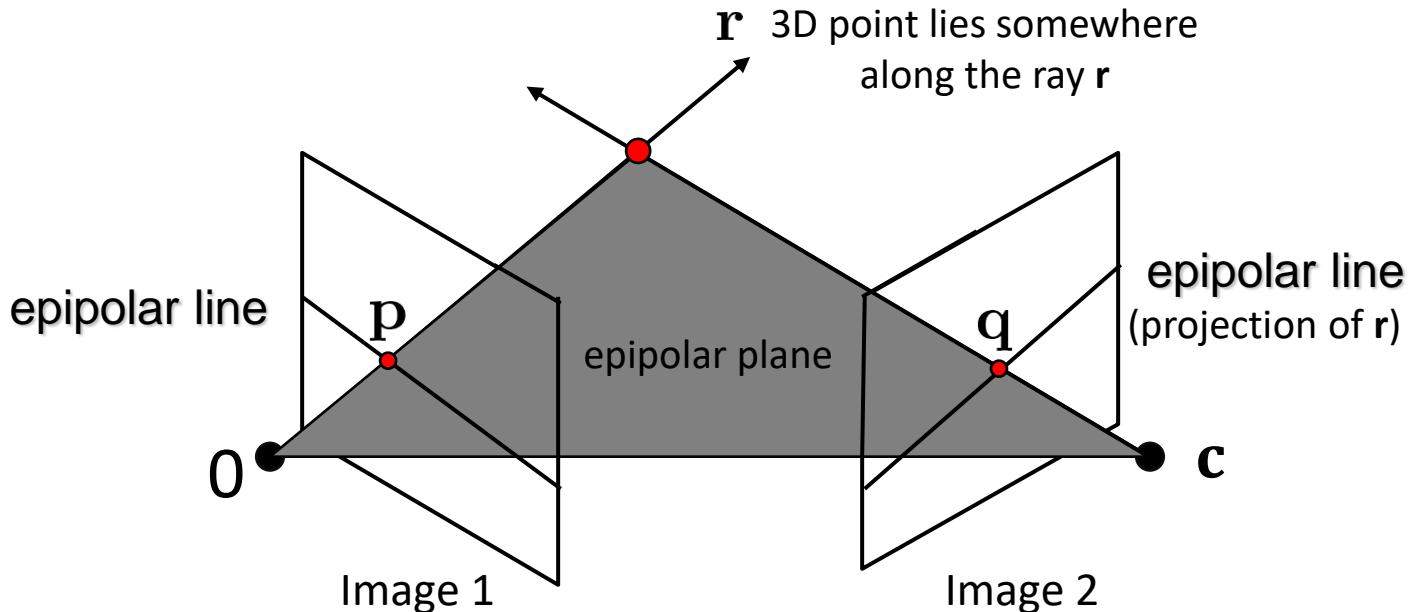
Outline



- Camera Calibration
- Two-view Geometry
- Triangulation

Two-view Geometry

- What if two cameras see the same point?
 - The corresponding point in the second view must lie on a line, the epipolar line
 - The two camera centers and one 3D point defines the epipolar plane
- What can we say about these quantities?



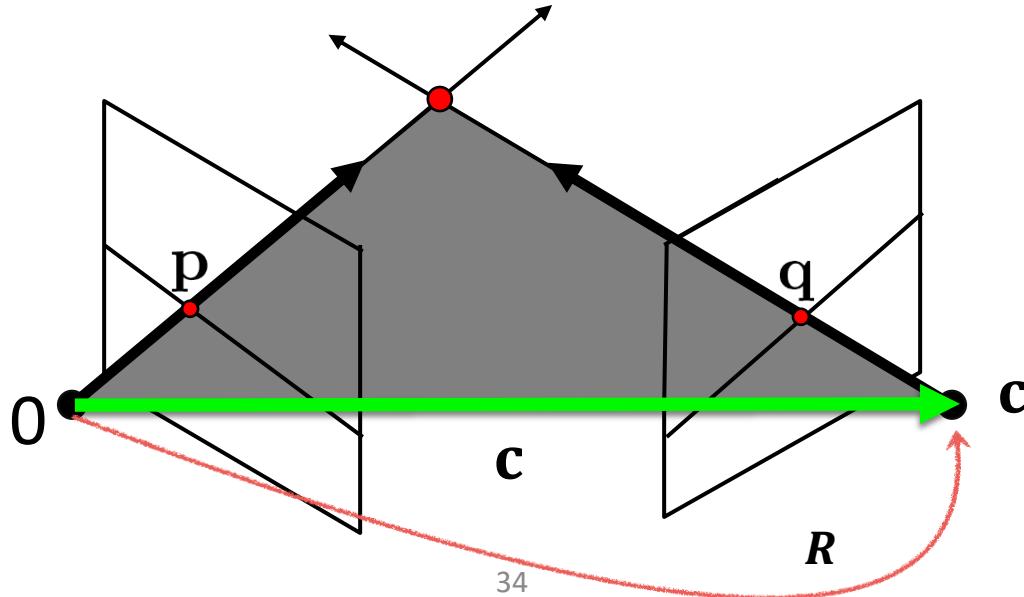


Sophie GERMAIN (portrait de), à l'âge de 11 ans.

Algebra is but written
geometry;
geometry is but drawn
algebra.
-- Sophie Germain

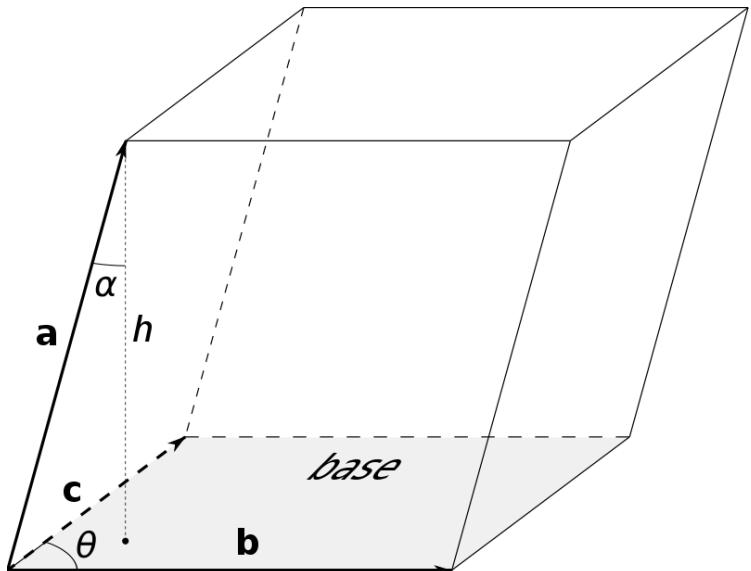
Essential Matrix – calibrated case

- Assume calibrated camera
 - So that we know 3D directions in the camera coordinate system i.e. \mathbf{p} , \mathbf{q} are known directions (\mathbf{p} is in camera frame 1, \mathbf{q} is in camera frame 2)
 - Suppose \mathbf{R} , \mathbf{c} are the rotations and translations between the two cameras i.e. $\mathbf{R}^T \mathbf{q}$ is the direction of \mathbf{q} in the camera frame 1
 - Constraint: \mathbf{p} , \mathbf{c} , $\mathbf{R}^T \mathbf{q}$ are coplanar (and all in camera frame 1)



Vector Mixed Product

- Vector mixed product: $a \cdot (b \times c)$
- Geometric meaning: the volume of a parallelepiped defined by the three vectors a, b , and c
- Three vectors a, b, c are coplanar iff $a \cdot (b \times c) = 0$



from wikipedia
35

Essential Matrix – calibrated case

- Assume calibrated camera

- Constraint: $\mathbf{p}, \mathbf{c}, \mathbf{R}^T \mathbf{q}$ are coplanar (and all in camera frame 1)

$$\mathbf{p} \cdot (\mathbf{c} \times \mathbf{R}^T \mathbf{q}) = 0$$



$$\mathbf{p}^T [\mathbf{c}]_\times \mathbf{R}^T \mathbf{q} = 0$$



$$\mathbf{q}^T \mathbf{R} [\mathbf{c}]_\times \mathbf{p} = \mathbf{q}^T \mathbf{E} \mathbf{p} = 0$$

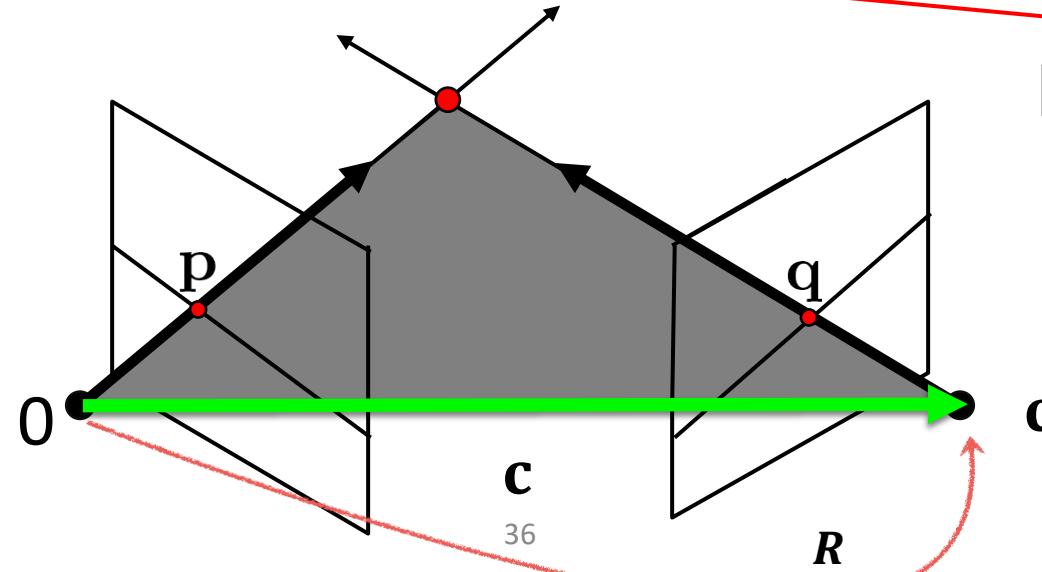
$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

$$\begin{aligned} \mathbf{E} &= \mathbf{R} [\mathbf{c}]_\times = \mathbf{R} [-\mathbf{R}^T \mathbf{t}]_\times \\ &= \mathbf{R} [\mathbf{R}^T \mathbf{t}]_\times = [\mathbf{t}]_\times \mathbf{R} \end{aligned}$$

For any vector \mathbf{t} and non-singular matrix \mathbf{M} one has

$$[\mathbf{t}]_\times \mathbf{M} = \mathbf{M}^* [\mathbf{M}^{-1} \mathbf{t}]_\times = \mathbf{M}^{-T} [\mathbf{M}^{-1} \mathbf{t}]_\times \text{ (up to scale).}$$

Essential Matrix
[Longuet-Higgins 1981]





Fundamental Matrix – uncalibrated case

- How do we generalize the Essential matrix to uncalibrated cameras?
- The way to compute direction from pixel coordinates (see page 24):

$$y = K^{-1}x$$

- We can substitute $p = K_1^{-1}\hat{p}$ and $q = K_2^{-1}\hat{q}$ into $q^T E p = 0$
 - Where \hat{p}, \hat{q} are pixel coordinates
 - Therefore,

$$\hat{q}^T K_2^{-T} E K_1^{-1} \hat{p} = 0$$

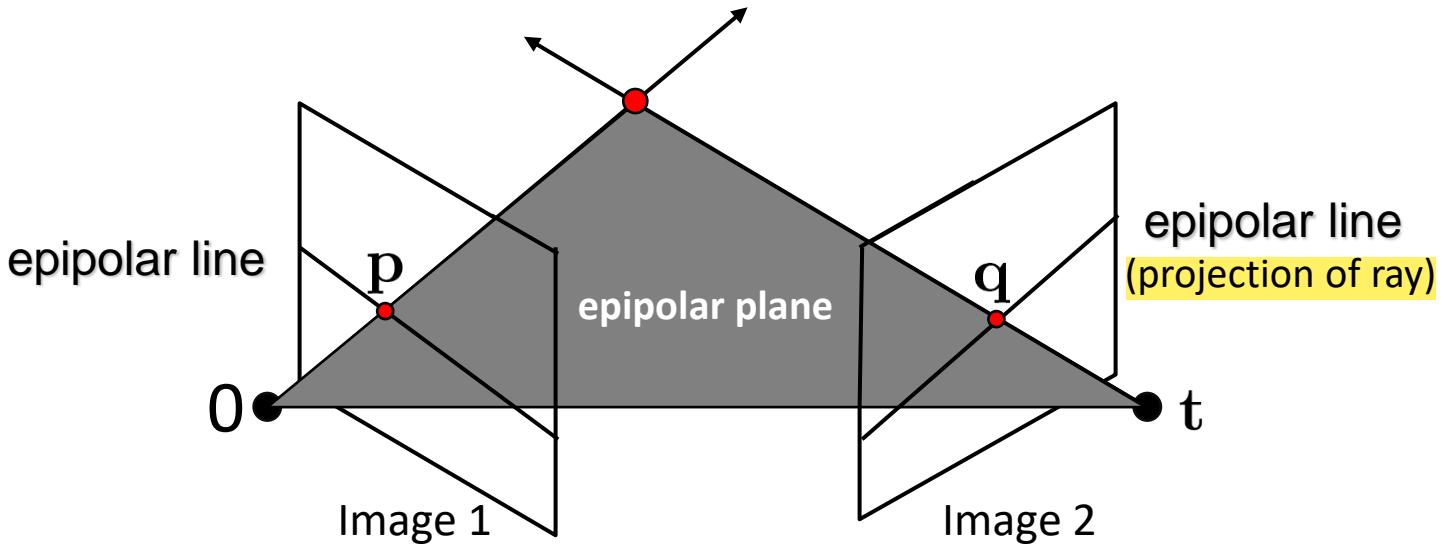


$$\hat{q}^T (K_2^{-T} E K_1^{-1}) \hat{p} = \hat{q}^T \cancel{F} \hat{p} = 0$$

Fundamental Matrix
[Oliver Faugeras 1992]

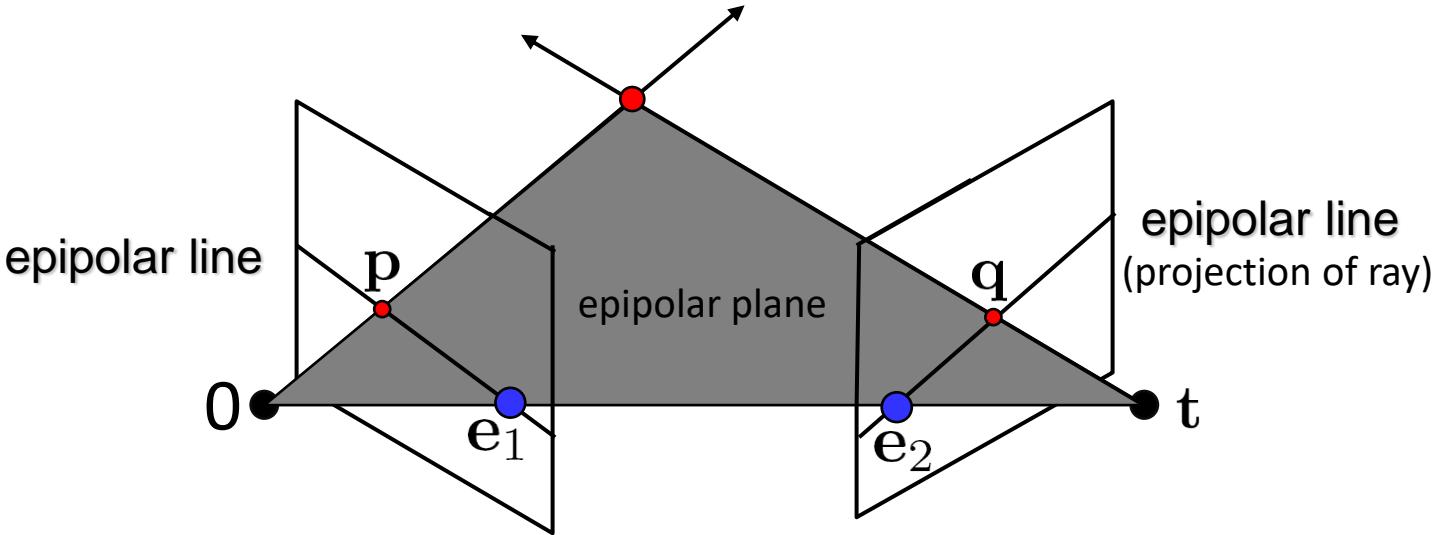
In the following, we abuse the symbols to use p, q instead of \hat{p}, \hat{q} to denote pixel coordinates

Fundamental Matrix – summary



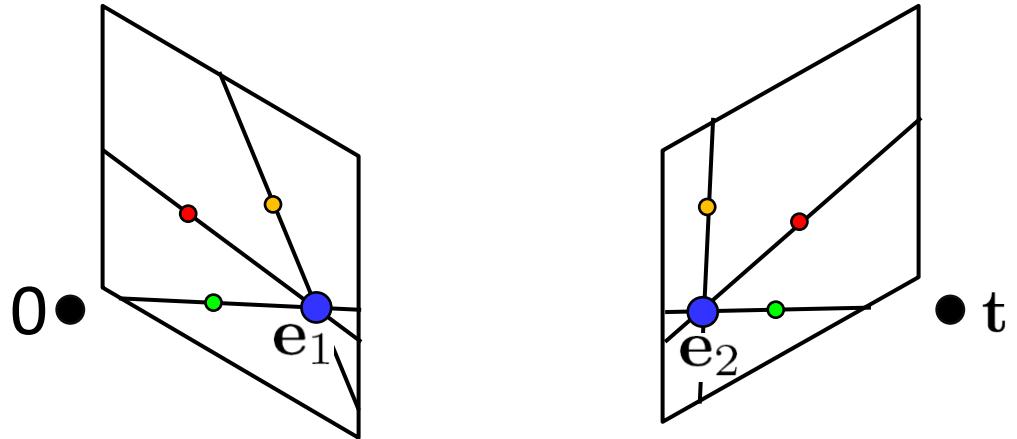
- This *epipolar geometry* of two views is described by a Special 3×3 matrix \mathbf{F} , called the *fundamental matrix*
- \mathbf{F} maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point p is: $\mathbf{F}p$
- *Epipolar constraint* on corresponding points: $\mathbf{q}^T \mathbf{F}p = 0$

Fundamental Matrix – summary



- Two Special points: \mathbf{e}_1 and \mathbf{e}_2 (the *epipoles*): projection of one camera into the other
- All of the epipolar lines in an image pass through the epipole
- Epipoles can be computed from \mathbf{F} as well: $\mathbf{e}_2^T \mathbf{F} = 0$ and $\mathbf{F}\mathbf{e}_1 = 0$
 - For any pixel \mathbf{p} , $\mathbf{F}\mathbf{p}$ is its epipolar line, which must pass through \mathbf{e}_2
 - Therefore, $\mathbf{e}_2^T \mathbf{F}\mathbf{p} = 0$ for any \mathbf{p} $\rightarrow \mathbf{e}_2^T \mathbf{F} = 0$
 - So, \mathbf{F} is rank 2

Fundamental Matrix – summary



- Two Special points: \mathbf{e}_1 and \mathbf{e}_2 (the *epipoles*): projection of one camera into the other
- All of the epipolar lines in an image pass through the epipole
- Epipoles can be computed from \mathbf{F} as well: $\mathbf{e}_2^T \mathbf{F} = 0$ and $\mathbf{F} \mathbf{e}_1 = 0$
 - For any pixel \mathbf{p} , $\mathbf{F}\mathbf{p}$ is its epipolar line, which must pass through \mathbf{e}_2
 - Therefore, $\mathbf{e}_2^T \mathbf{F}\mathbf{p} = 0$ for any \mathbf{p} $\rightarrow \mathbf{e}_2^T \mathbf{F} = 0$
 - So, \mathbf{F} is rank 2

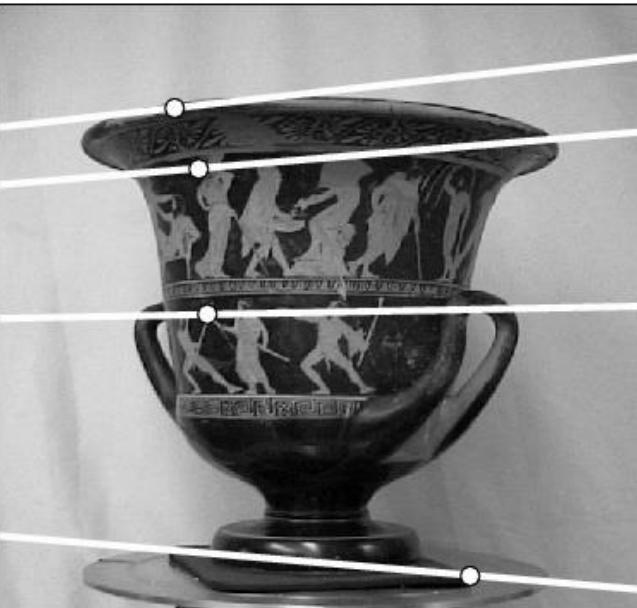
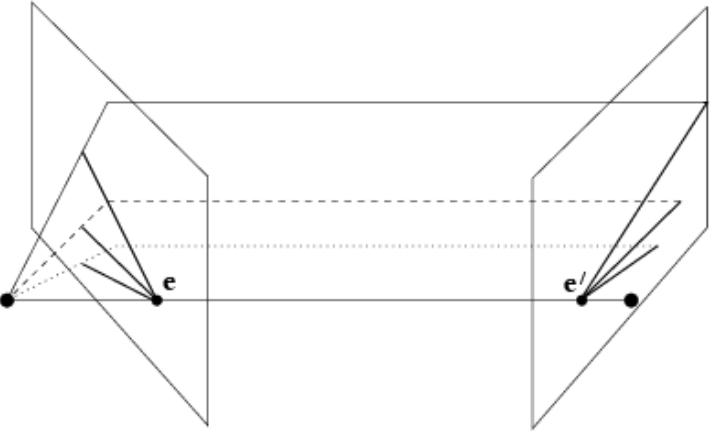
The Fundamental Matrix Song



Questions?

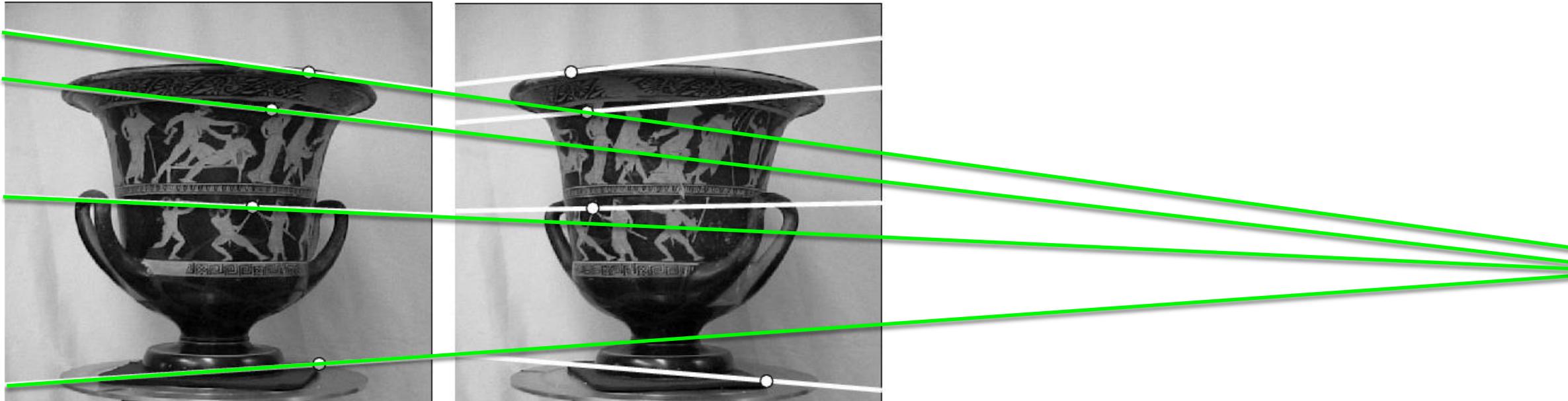
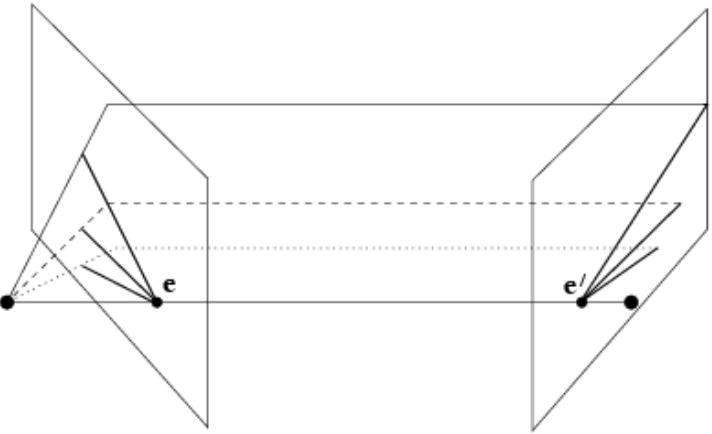


Example: converging cameras

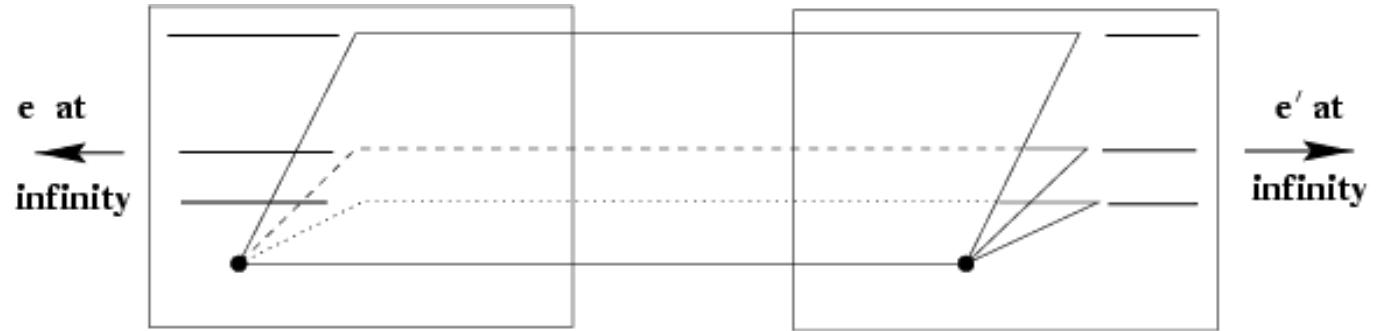


Where is the epipole in this image?

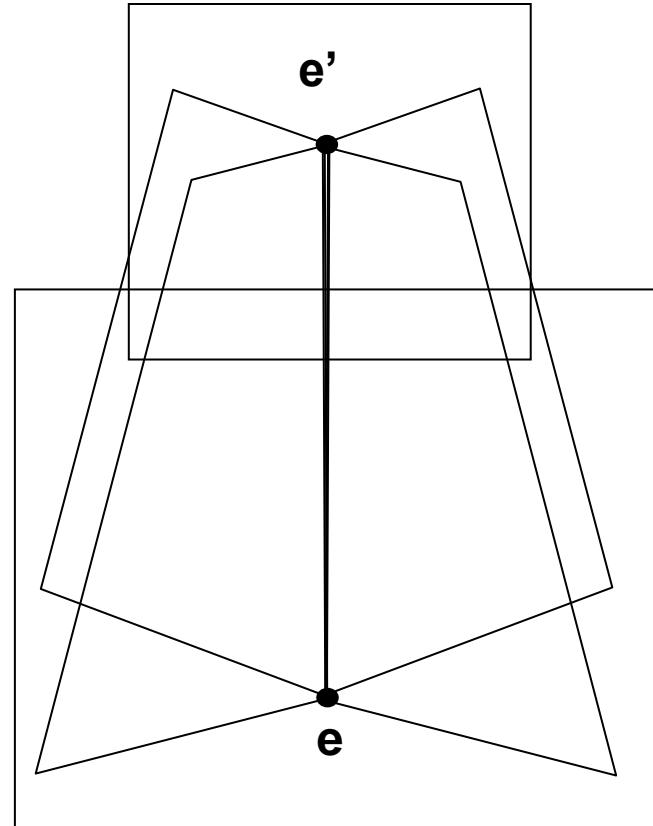
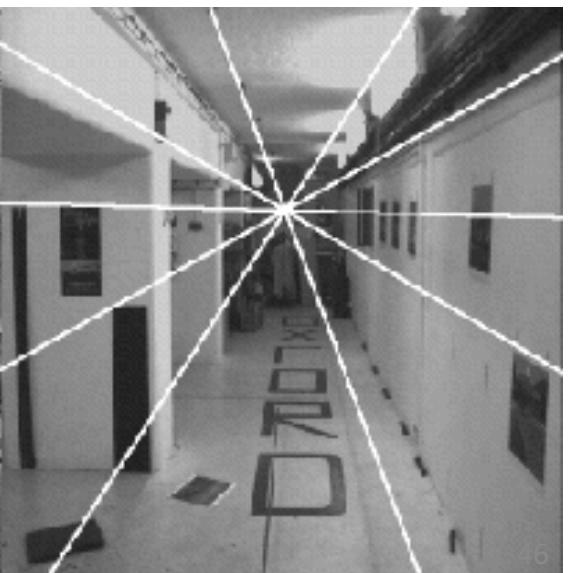
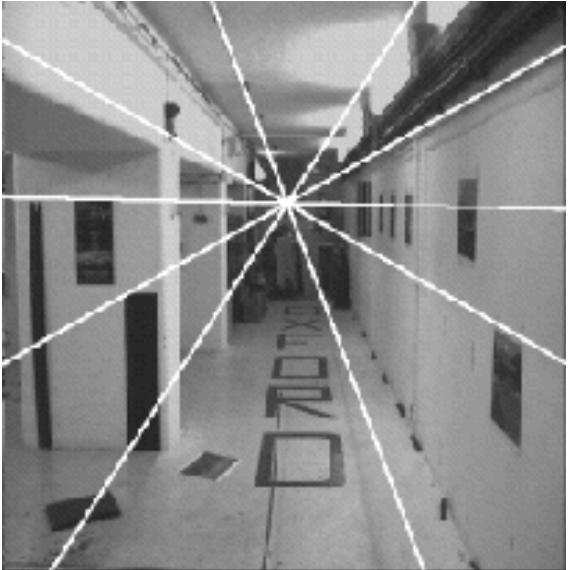
Example: converging cameras



Example: motion parallel with image plane

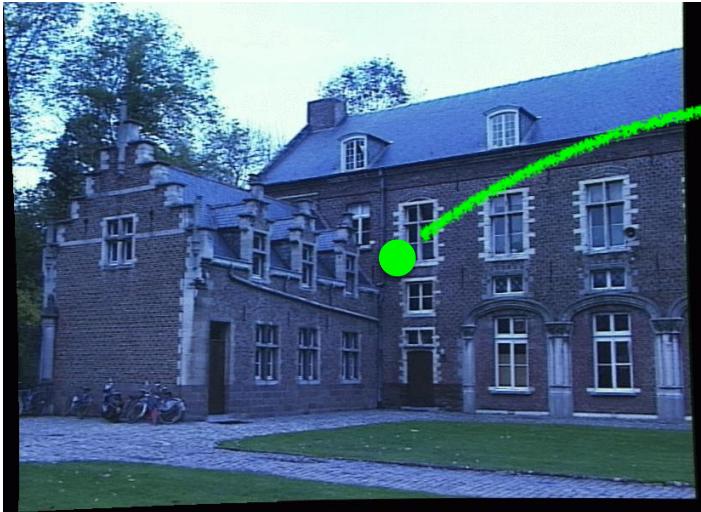


Example: forward motion



the epipolar constraint for stereo vision

Task: Match point in left image to point in right image



Left image



Right image

Epipolar constraint reduces search to a **single** line

Questions?





Estimating the Fundamental Matrix

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$$

$$x' x f_{11} + x' y f_{12} + x' f_{13} + y' x f_{21} + y' y f_{22} + y' f_{23} + x f_{31} + y f_{32} + f_{33} = 0$$

separate known from unknown

$$\begin{bmatrix} x' x, x' y, x', y' x, y' y, y', x, y, 1 \end{bmatrix} \begin{bmatrix} f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33} \end{bmatrix}^T = 0$$

(data) (unknowns)

linear equation

$$\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathbf{f} = 0$$

$$\mathbf{A} \mathbf{f} = 0$$



The Singularity Constraint

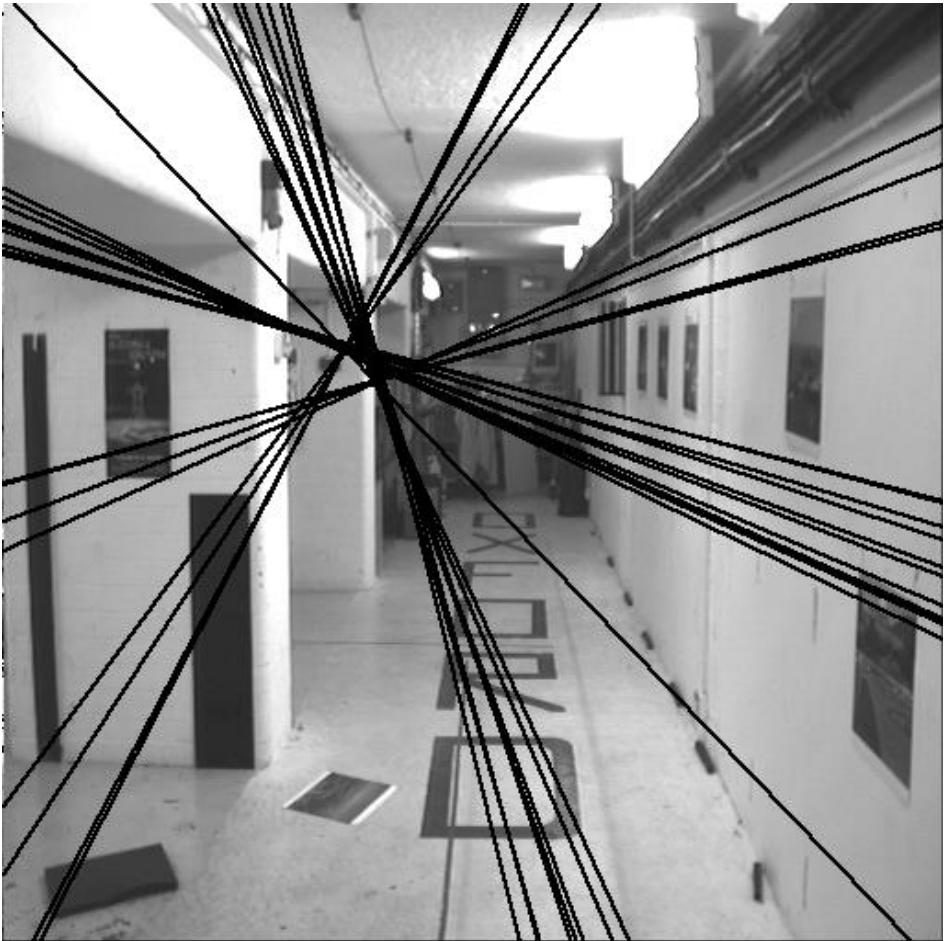
$$e'^T F = 0 \quad Fe = 0 \quad \det(F) = 0 \quad \text{rank } F = 2$$

SVD from linearly computed **F matrix (rank 3)**

$$F = U \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} V^T = U_1 \sigma_1 V_1^T + U_2 \sigma_2 V_2^T + U_3 \sigma_3 V_3^T$$

Compute closest rank-2 approximation $\min \|F - F'\|_F$

$$F' = U \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & 0 \end{bmatrix} V^T = U_1 \sigma_1 V_1^T + U_2 \sigma_2 V_2^T$$



when F is non-singular, epipolar lines won't intersect at the same point

the NOT normalized 8-point algorithm

$$\begin{bmatrix} x_1x_1' & y_1x_1' & x_1' & x_1y_1' & y_1y_1' & y_1' & x_1 & y_1 & 1 \\ x_2x_2' & y_2x_2' & x_2' & x_2y_2' & y_2y_2' & y_2' & x_2 & y_2 & 1 \\ \vdots & \vdots \\ x_nx_n' & y_nx_n' & x_n' & x_ny_n' & y_ny_n' & y_n' & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

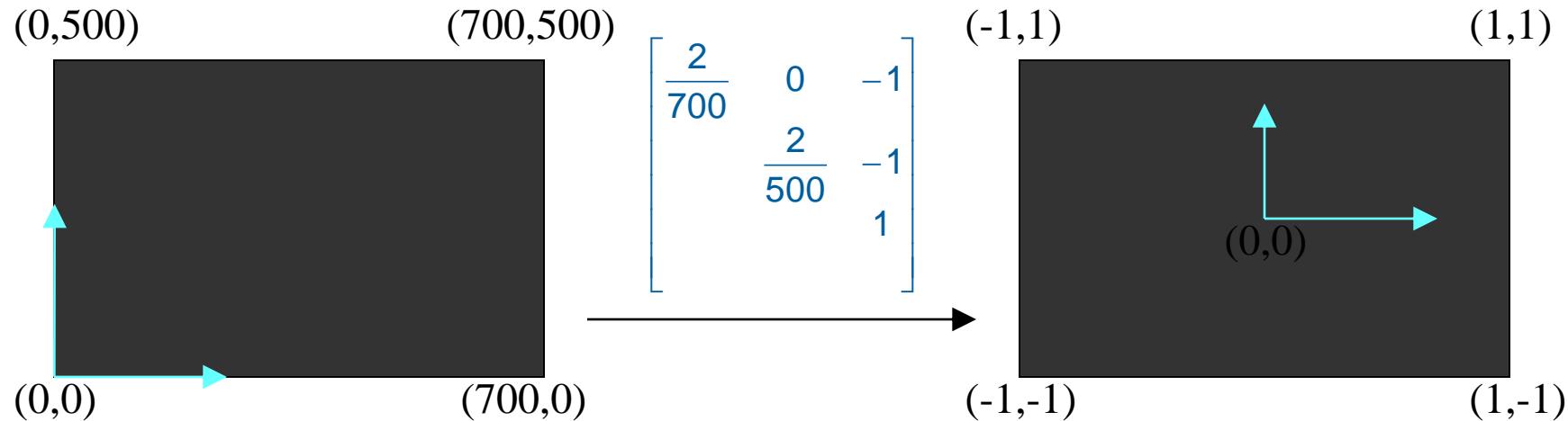
$\sim 10000 \quad \sim 10000 \quad \sim 100 \quad \sim 10000 \quad \sim 10000 \quad \sim 100 \quad \sim 100 \quad \sim 100 \quad 1$



**Orders of magnitude difference
Between column of data matrix
→ least-squares yields poor results**

Normalized 8-point Algorithm

Transform image to $\sim [-1,1] \times [-1,1]$



Least squares yields good results (Hartley, PAMI' 97)

In Defence of the 8-point Algorithm

Richard I. Hartley,



7-point Algorithm – the minimum case

$$\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots \\ x'_7 x_7 & x'_7 y_7 & x'_7 & y'_7 x_7 & y'_7 y_7 & y'_7 & x_7 & y_7 & 1 \end{bmatrix} f = 0$$

7 equations, 9 unknowns

$$A = U_{7 \times 7} \text{diag}(\sigma_1, \dots, \sigma_7, 0, 0) V_{9 \times 9}^T$$

$$\Rightarrow A[V_8 V_9] = 0_{9 \times 2} \quad \Rightarrow A(V_8 + \lambda V_9) = 0_{9 \times 2}$$

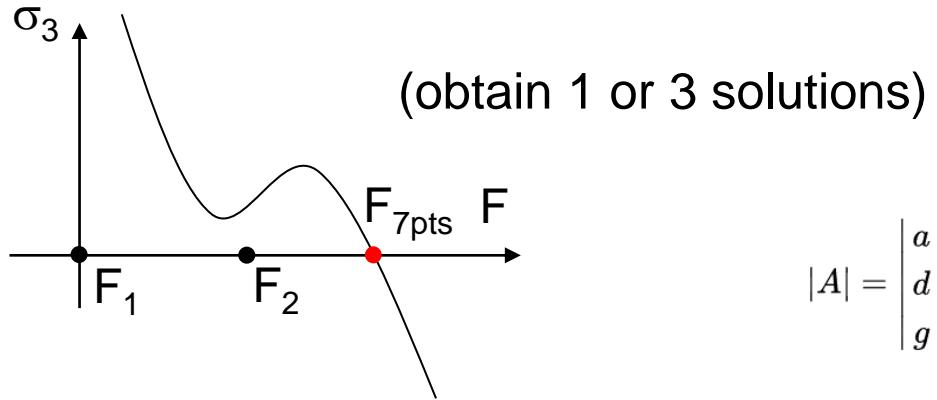
$$x_i^T (F_1 + \lambda F_2) x_i = 0, \forall i = 1 \dots 7$$

one parameter family of solutions

but $F_1 + \lambda F_2$ not automatically rank 2

so we can solve λ by letting the rank equal to 2

7-point Algorithm – the minimum case



$$\begin{aligned} |A| &= \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c \begin{vmatrix} d & e \\ g & h \end{vmatrix} \\ &= aei + bfg + cdh - ceg - bdi - afh \end{aligned}$$

$$\det(F_1 + \lambda F_2) = a_3 \lambda^3 + a_2 \lambda^2 + a_1 \lambda + a_0 = 0 \quad (\text{cubic equation})$$

$$\det(F_1 + \lambda F_2) = \det F_2 \det(F_2^{-1}F_1 + \lambda I) = 0$$

Compute possible λ as eigenvalues of $F_2^{-1}F_1$ (only real solutions are potential solutions)

Three solutions when the points and camera center are on a ‘critical surface’.



Error Functions

The 8-point and 7-point algorithm minimizes an algebraic error

We can define a symmetric geometric error as:

$$\sum_i d(x'_i, Fx_i)^2 + d(x_i, F^T x'_i)^2$$

Minimizing the distance between the corresponding point and epipolar line

The best objective function is the re-projection error (= Maximum Likelihood Estimation for Gaussian noise)

$$\sum_i d(x_i, \hat{x}_i)^2 + d(x'_i, \hat{x}'_i)^2 \text{ subject to } \hat{x}'^T F \hat{x} = 0$$



Recommendations:

1. Do not use unnormalized algorithms
2. Quick and easy to implement: 8-point normalized
3. Better: enforce rank-2 constraint during minimization
4. Best: Maximum Likelihood Estimation by minimizing re-projection error



Degenerate cases:

- Degenerate cases (only a homography can be estimated)
 - Planar scene
 - Pure rotation

Questions?





Computation of the Essential matrix

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$$

Compute \mathbf{F} first, then simply take: $\mathbf{E} = \mathbf{K}_1^T \mathbf{F} \mathbf{K}_2$

$$\mathbf{y}'^T \mathbf{E} \mathbf{y} = 0 \quad y_i = \mathbf{K}^{-1} \mathbf{x}_i$$

Or, take 8-point algorithm to solve \mathbf{E} , then enforce:

$$E = U \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \sigma_3 \end{bmatrix} V^T \rightarrow E = U \begin{bmatrix} \frac{\sigma_1 + \sigma_2}{2} & & \\ & \frac{\sigma_1 + \sigma_2}{2} & \\ & & 0 \end{bmatrix} V^T$$



Computation of the Essential matrix

- E has less degrees of freedom than F
- In principle, 5 pair of corresponding points are sufficient to decide E .
 - The 5-point algorithm by David Nister

An Efficient Solution to the Five-Point Relative Pose Problem

David Nistér
Sarnoff Corporation
CN5300, Princeton, NJ 08530
dnister@sarnoff.com

PAMI 2004



Getting Camera Matrices from E

For a given $\mathbf{E} = \mathbf{U} \text{diag}(1,1,0) \mathbf{V}^T$ (by SVD decomposition),
and the first camera matrix $\mathbf{P} = [\mathbf{I} | 0]$,
there are 4 choices for the second camera matrix \mathbf{P}' , namely

$$\mathbf{P}' = \begin{bmatrix} \mathbf{U} \mathbf{W} \mathbf{V}^T & | & \mathbf{u}_3 \end{bmatrix} \quad \text{or} \quad \mathbf{P}' = \begin{bmatrix} \mathbf{U} \mathbf{W} \mathbf{V}^T & | & -\mathbf{u}_3 \end{bmatrix}$$

$$\text{or} \quad \mathbf{P}' = \begin{bmatrix} \mathbf{U} \mathbf{W}^T \mathbf{V}^T & | & \mathbf{u}_3 \end{bmatrix} \quad \text{or} \quad \mathbf{P}' = \begin{bmatrix} \mathbf{U} \mathbf{W}^T \mathbf{V}^T & | & -\mathbf{u}_3 \end{bmatrix}$$

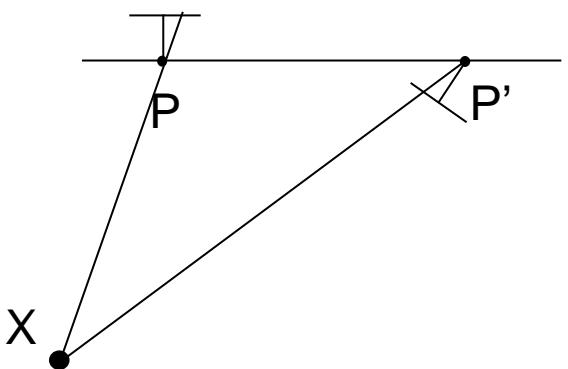
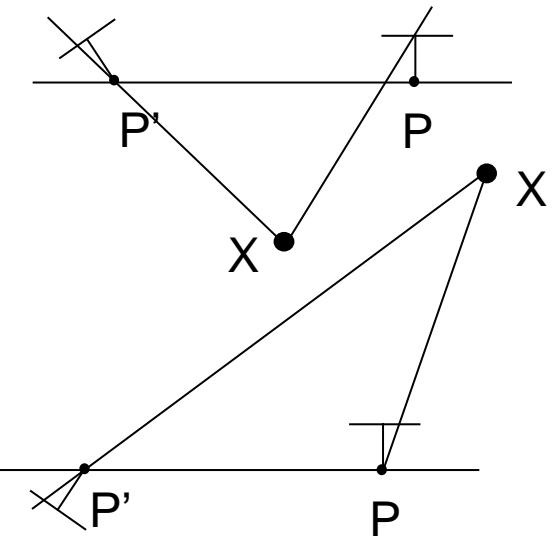
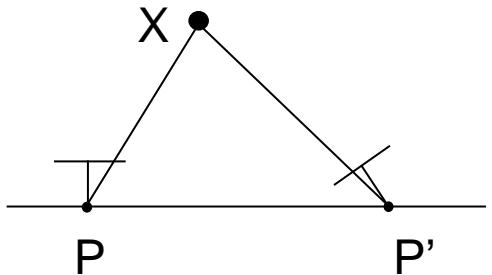
$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

\mathbf{u}_3 is the last column of \mathbf{U}

For the proof, please refer to
section 9.6 of the 'multiview
geometry' book

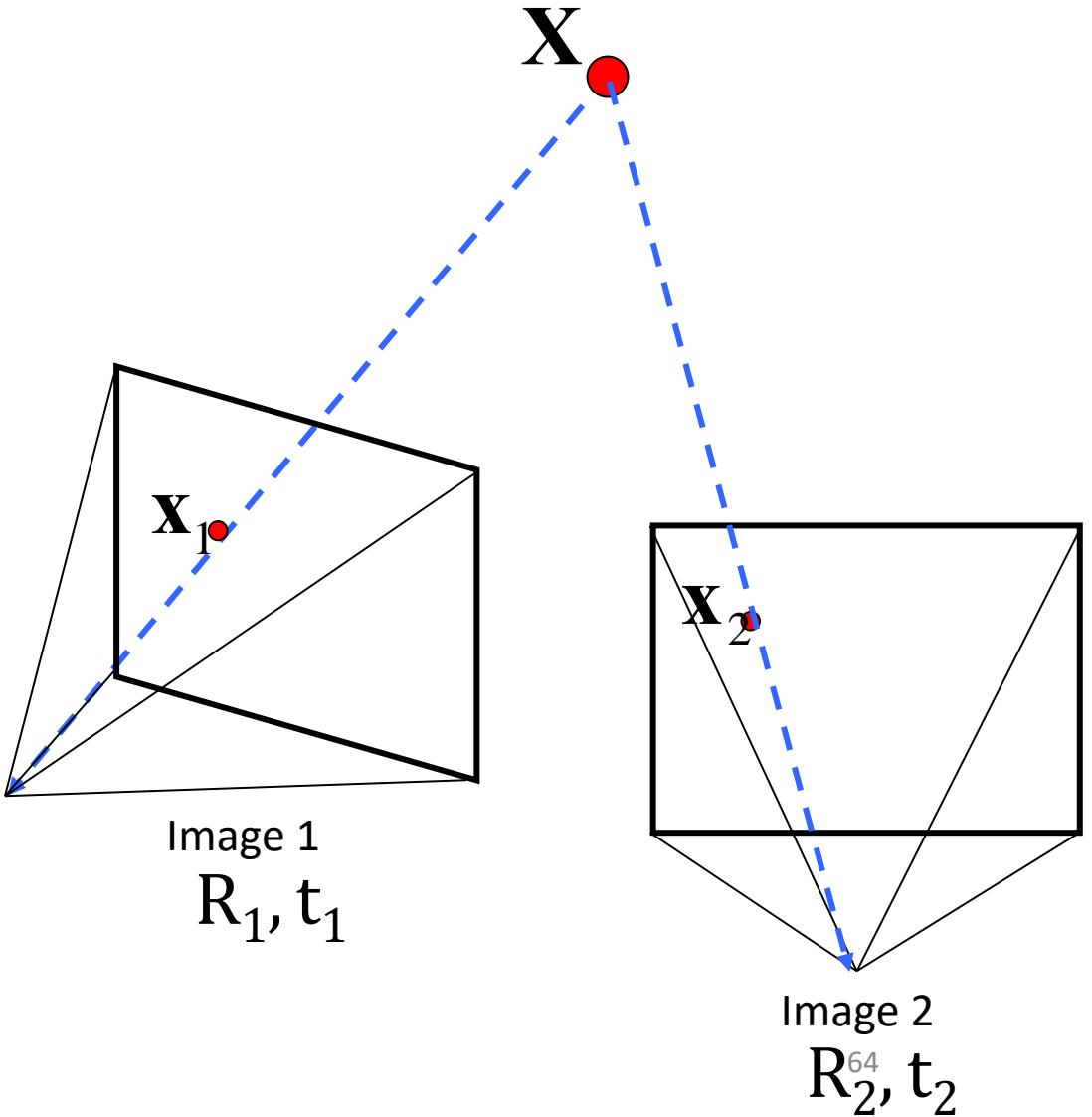
Selecting from the Four Solutions

- Among these four configurations, only one is valid where the reconstructed points are in front of both cameras





Triangulation (to be studied soon)

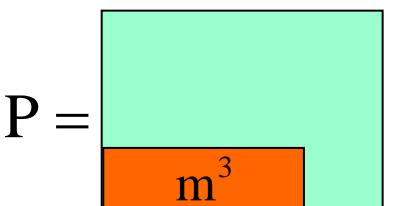




In front of the camera?

- A point X
- Direction from camera center to point $X - C$
- The direction of principal axis m^3
- Compute the angle between $(X - C)$ and m^3
- Just need to test $(X - C) \cdot m^3 > 0$

$$C = \begin{pmatrix} -M^{-1}p_4 \\ 1 \end{pmatrix}$$



Pick the Solution

- With maximal number of points in front of both cameras.

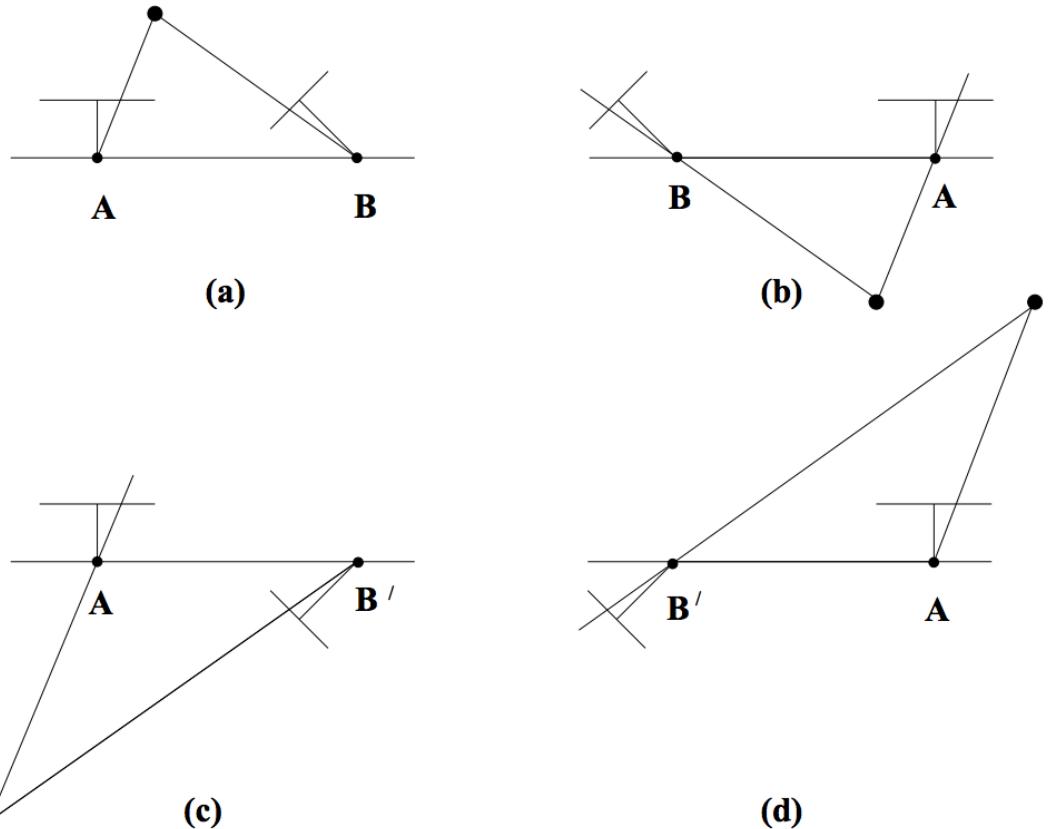


Fig. 9.12. **The four possible solutions for calibrated reconstruction from E.** Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.

Questions?



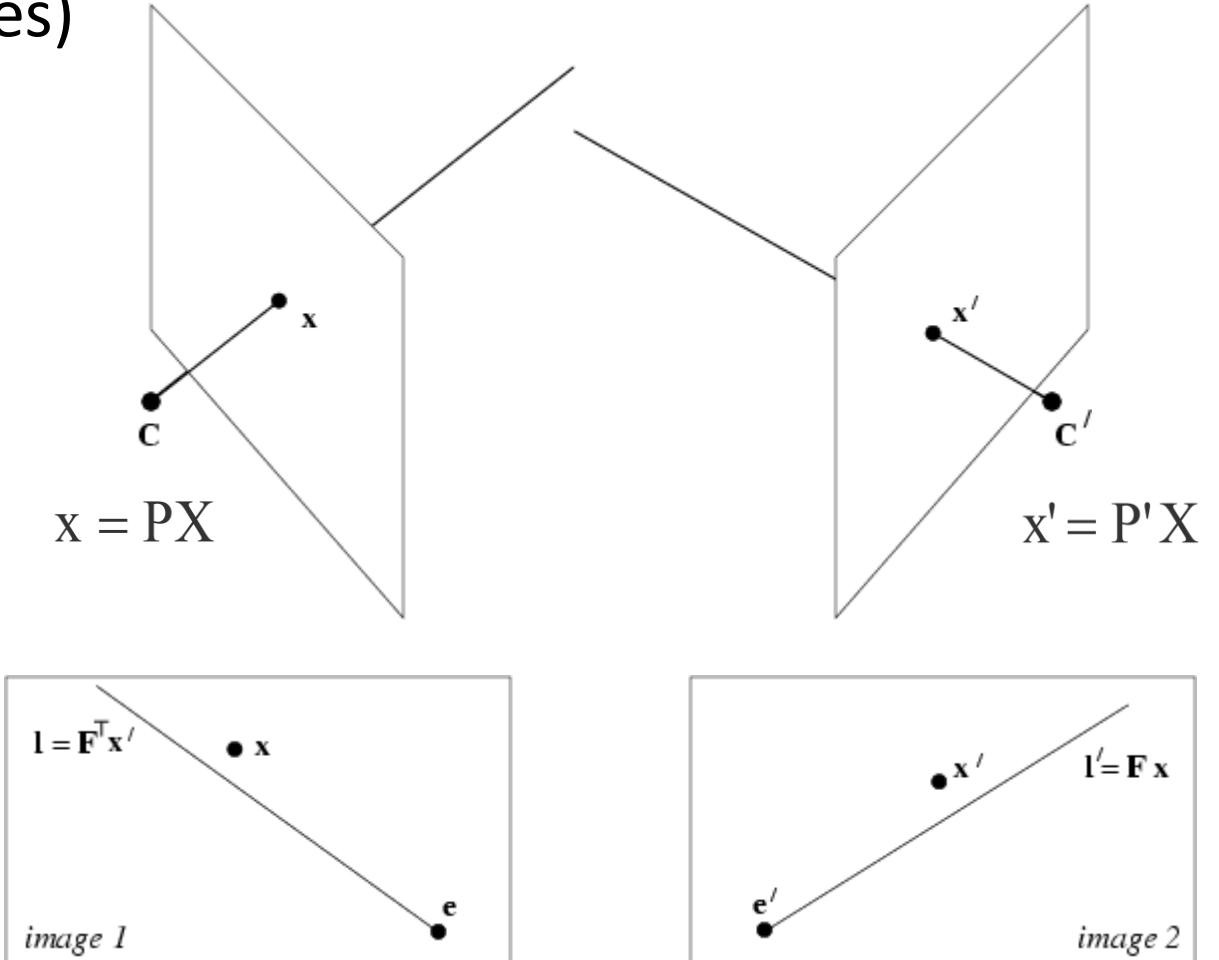
Outline



- Camera Calibration
- Two-view Geometry
- Triangulation

Point Reconstruction

- Estimate 3D point X from known cameras P, P' (and feature correspondences)





Direct Linear Transform

$$x = PX$$

$$\Rightarrow x \times PX = 0$$

$$x' = P'X$$

$$\Rightarrow x' \times P'X = 0$$

$$AX = 0$$

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix}$$

$$P = \begin{bmatrix} p^{1T} \\ p^{2T} \\ p^{3T} \end{bmatrix}$$

Homogeneous coordinate, add constraint

$$\|X\| = 1$$

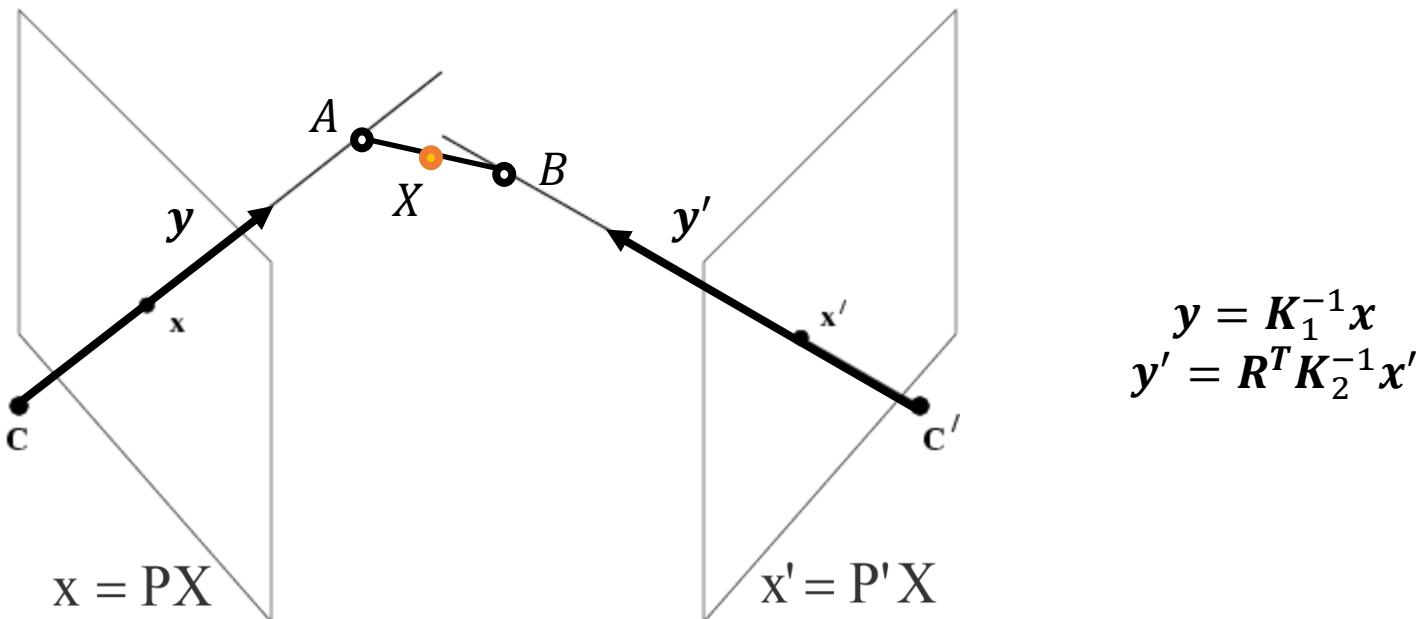
Convert to inhomogeneous coordinate

$$(X, Y, Z, 1)$$

This method minimizes an algebraic error

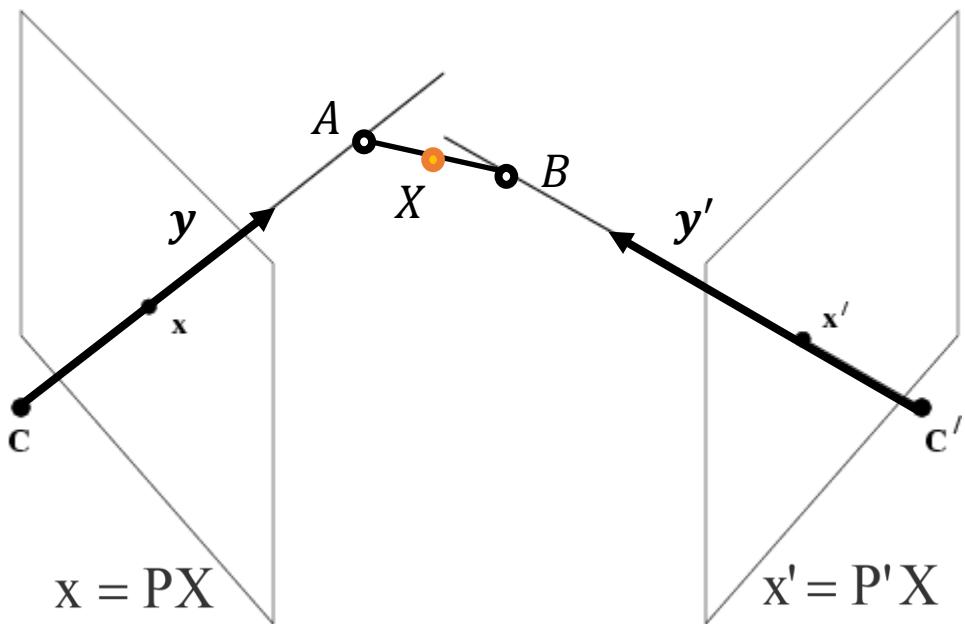
Mid-point Algorithm

- Find the middle point of the mutual perpendicular line segment \mathbf{AB}
- $$\mathbf{A} = \mathbf{c} + d_1 \mathbf{y} \quad \mathbf{B} = \mathbf{c}' + d_2 \mathbf{y}'$$
- $\mathbf{c}, \mathbf{c}', \mathbf{y}, \mathbf{y}'$ are all in the same coordinate system (e.g. camera frame 1)



Mid-point Algorithm

- Choose the first camera's coordinate as a reference
 - $\mathbf{c} = 0, \mathbf{P} = \mathbf{K}_1[\mathbf{I} | 0]$
- Put the second camera in that coordinate system
 - Assume known relative rotation \mathbf{R} , translation \mathbf{t}
 - $\mathbf{P}' = \mathbf{K}_2[\mathbf{R}|\mathbf{t}], \mathbf{c}' = -\mathbf{R}^T\mathbf{t}$
 - $\mathbf{y}' = \mathbf{R}^T\mathbf{K}_2^{-1}\mathbf{x}'$



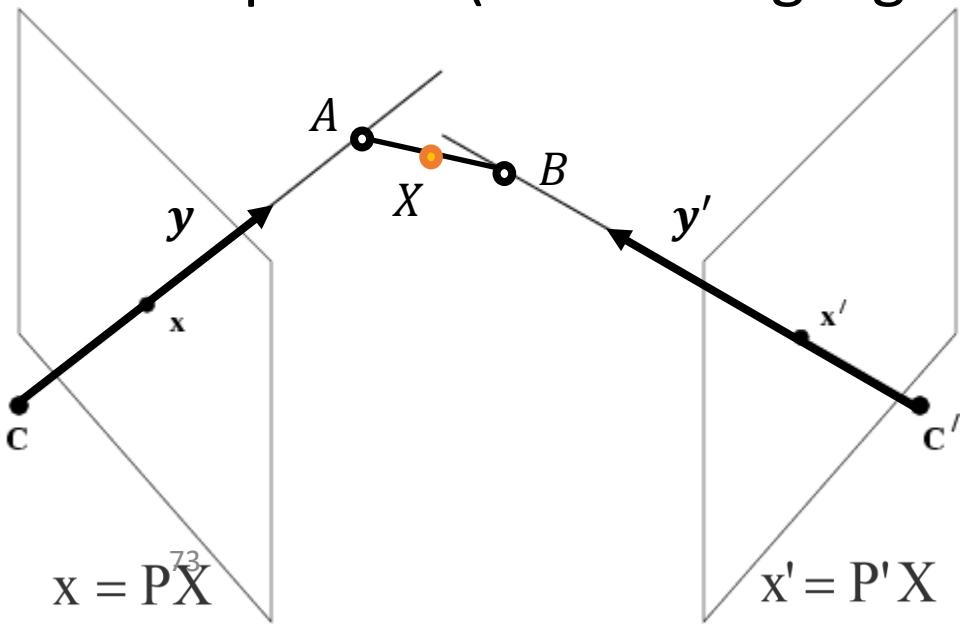
Mid-point Algorithm

- Since AB is the mutual perpendicular line segment
 - $AB \perp y, AB \perp y'$
- This means:

$$(A - B) \times (y \times y') = \mathbf{0}$$

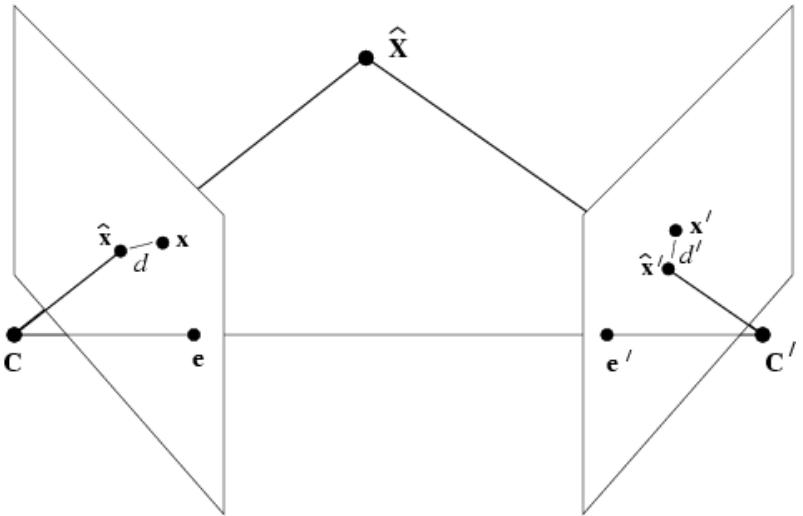
- This generates three equations of d_1, d_2
- Solve d_1, d_2 from the above linear equation (minimizing a geometric error)

$$\begin{aligned} A &= c + d_1 y \\ B &= c' + d_2 y' \end{aligned}$$



Reprojection Error

$d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x}}')^2$ subject to $\hat{\mathbf{x}}'^T \mathbf{E} \hat{\mathbf{x}} = 0$ (or $\hat{\mathbf{x}}'^T \mathbf{F} \hat{\mathbf{x}} = 0$)
 or equivalently subject to $\hat{\mathbf{x}} = \mathbf{P}\hat{\mathbf{X}}$ and $\hat{\mathbf{x}}' = \mathbf{P}'\hat{\mathbf{X}}$



compute using the Levenberg-Marquardt algorithm

This triangulation works for uncalibrated cameras

- The algebraic error and mid-point algorithm needs \mathbf{K}, \mathbf{K}' (pre-calibrated cameras)

Questions?





Algorithms Studied Today

- Camera calibration (resection / PnP)
 - 3D \leftrightarrow 2D correspondences, compute the camera pose
- Epipolar geometry (relative motion estimation)
 - 2D \leftrightarrow 2D correspondences, compute relative camera motion (up to a scale)
- Triangulation
 - 2D \leftrightarrow 2D correspondences (and known camera poses), compute 3D point