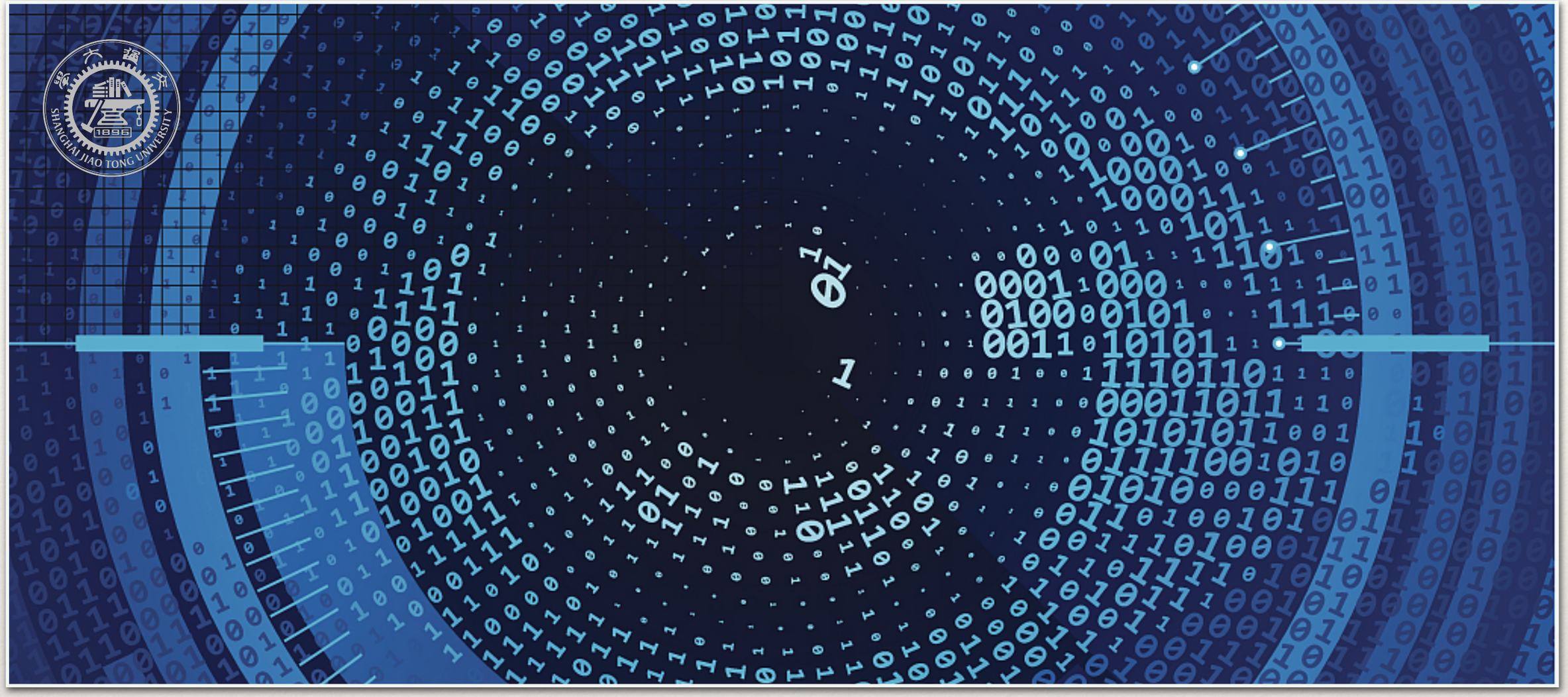


Shanghai Jiao Tong University

Computer Vision

Instructor: Xu Zhao

Class No.: C032703 F032528

Spring 2020

Xu Zhao @ Shanghai Jiao Tong university

Lecture 6: Structure from motion

Contents

- **Structure from motion: the problem**
- **Fundamental matrix computation**
- **Motion perspective**

Geometry of two views

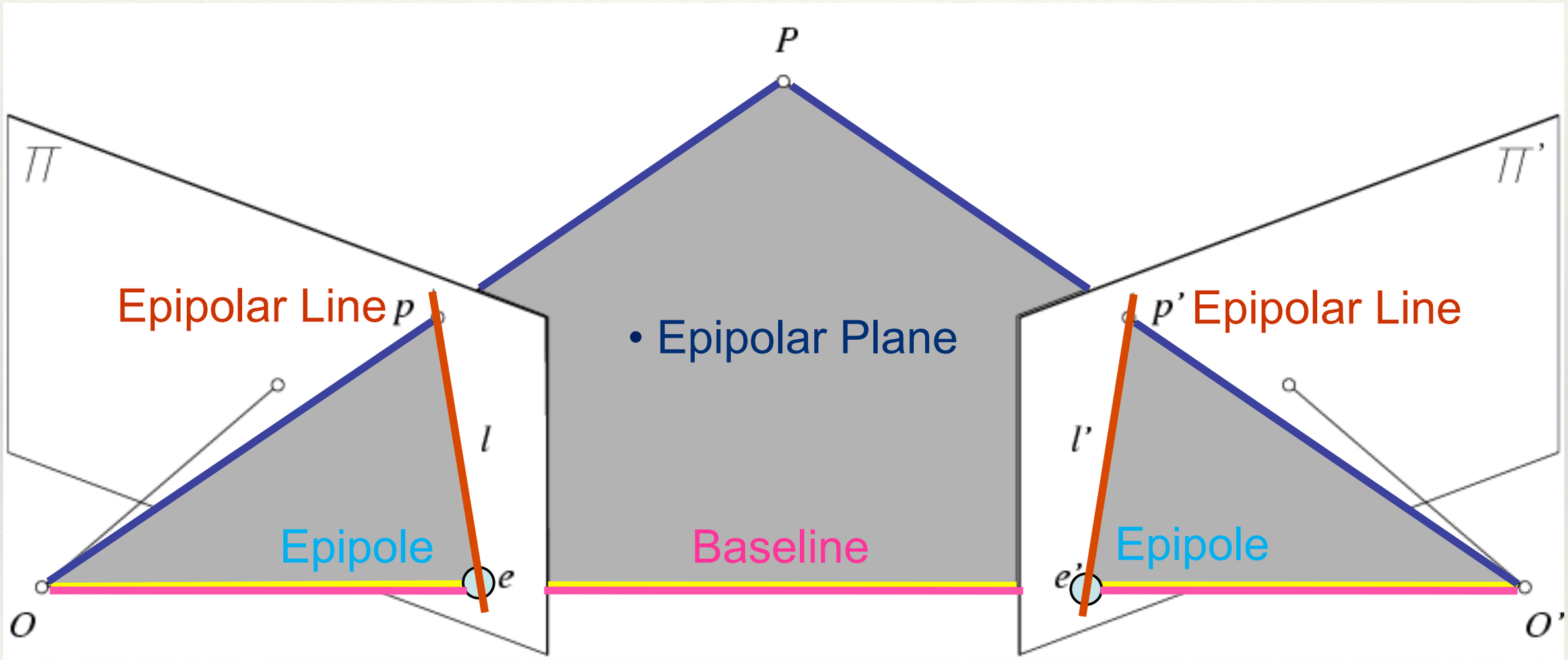
- Two cameras associated with camera matrix: P, P'
  - $\mathbf{x} = P\mathbf{X}, \mathbf{x}' = P'\mathbf{X}$ ($\mathbf{X}$: 3-D space point, $\mathbf{x}, \mathbf{x}'$: image points)
- Questions:

I. **Correspondence geometry**: Given an image point $\mathbf{x}$ in the first view, how does this constrain the position of the corresponding point $\mathbf{x}'$ in the second view?

II. **Camera geometry (motion)**: Given a set of corresponding image points $\mathbf{x} \leftrightarrow \mathbf{x}'$, what are the cameras P, P'

III. **Scene geometry (structure)**: Given a set of corresponding image points $\mathbf{x} \leftrightarrow \mathbf{x}'$, and cameras P, P', what is the position of $\mathbf{X}$ in 3-D space?

For the first question: Epipolar geometry



The fundamental matrix

- Without knowing K and K’, we can define a similar relation using *unknown* normalized coordinates

$$\hat{x}^T E \hat{x}' = 0$$

$$\hat{x} = K^{-1} x$$

$$\hat{x}' = K'^{-1} x'$$

$$\Rightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

Fundamental Matrix
(Faugeras and Luong, 1992)

Structure from motion

- Given a set of corresponding image points $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$,
  - what are the cameras P, P' ($\mathbf{x_i} = P\mathbf{X_i}, \mathbf{x_i'} = P'\mathbf{X_i}$) and 3-D point $\mathbf{X}_i$?
  - **uncalibrated camera:** K, R, t are unknown
- General outline:
  I. Compute the fundamental matrix from point correspondences
  II. Compute the camera matrices from the fundamental matrices
  III. For each point correspondence $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$, compute the points in 3-D space

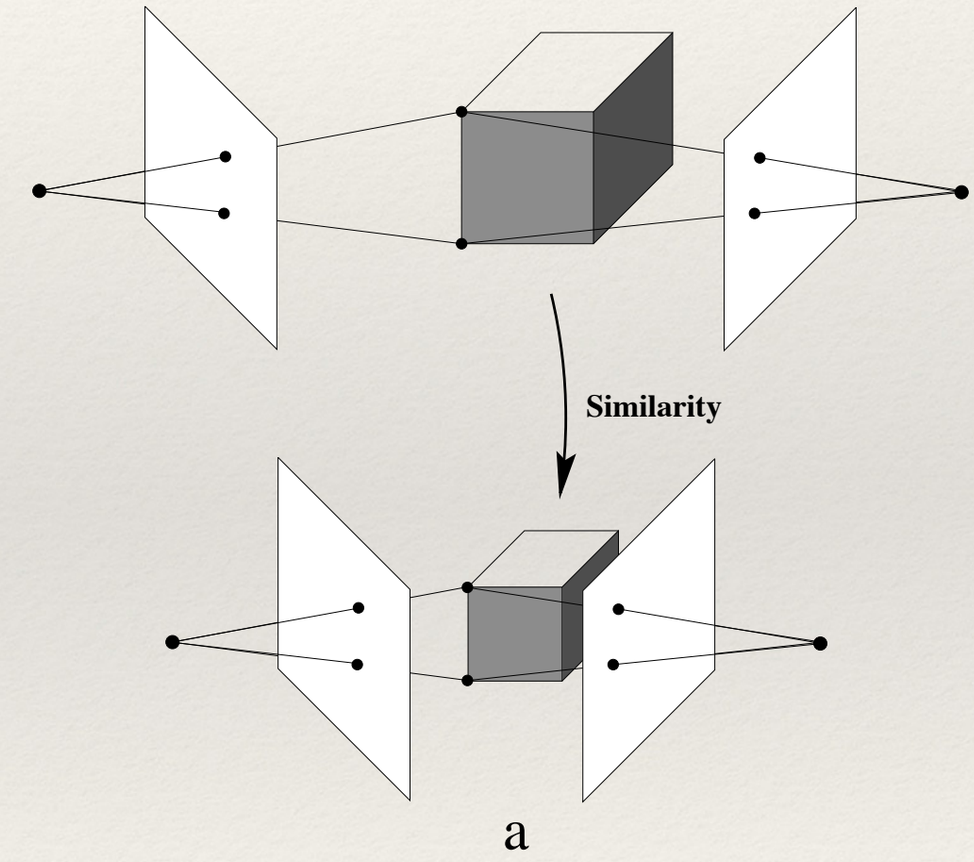
Reconstruction ambiguity

- Based on the images alone without knowledge of the 3D scene, it is impossible to estimate the absolute position and orientation of a scene.

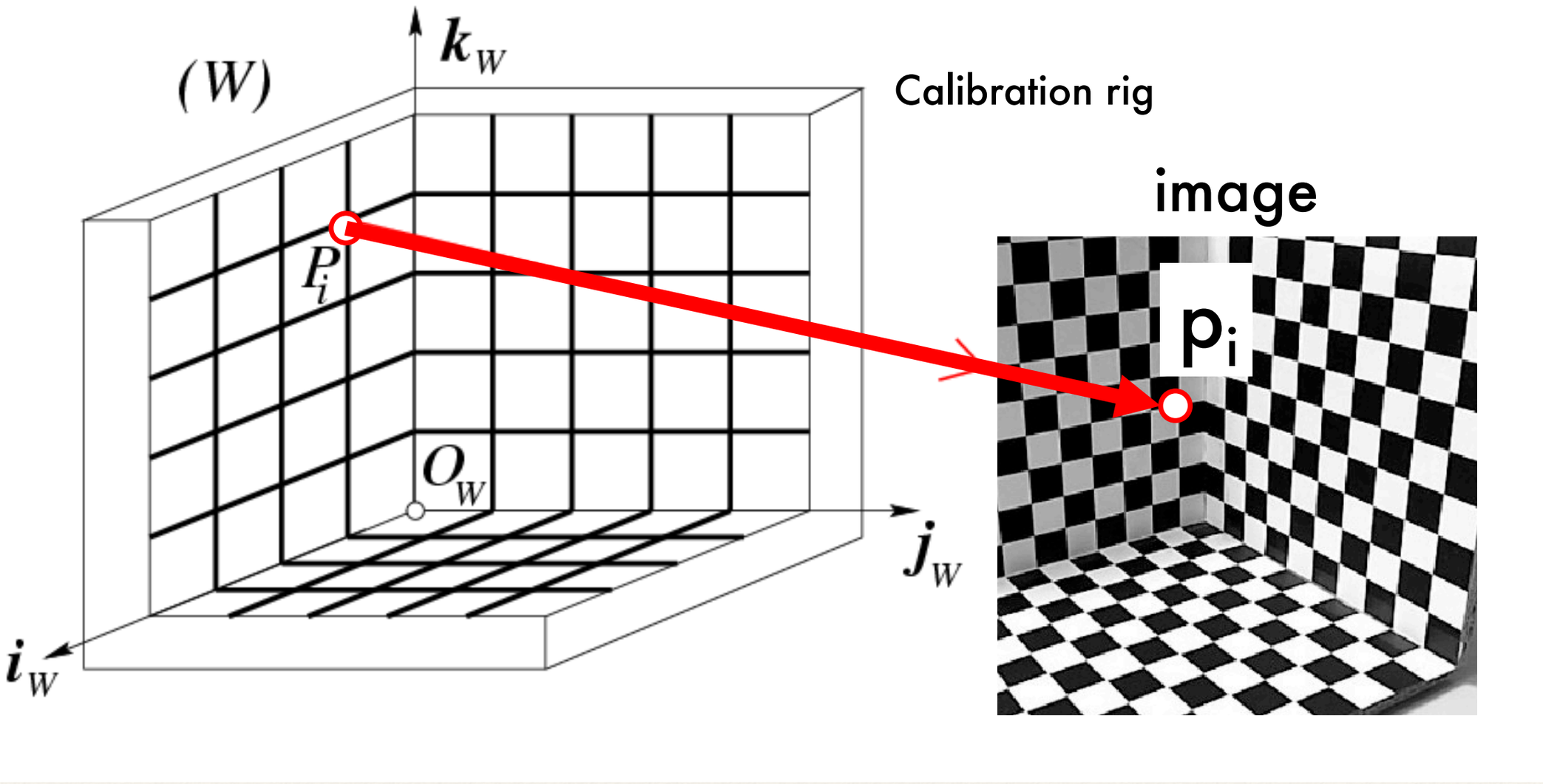
Real car?
Toy car?

Similarity (metric) ambiguity

- $\mathbf{x_i} = P\mathbf{X_i}, \mathbf{x_i'} = P'\mathbf{X_i}$
- $H_s = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & \lambda \end{bmatrix}$ - similarity transformation
- $\mathbf{X}_i \rightarrow H_s\mathbf{X}_{i'}, P \rightarrow PH_s^{-1}, P' \rightarrow P'H_s^{-1}$
  - $P\mathbf{X}_i = (PH_s^{-1})(H_sX_i)$: No change of the image points!
- The ambiguity exists even for (intrinsically) calibrated cameras
- For calibrated cameras, the similarity ambiguity is the only ambiguity



Resolving the similarity ambiguity

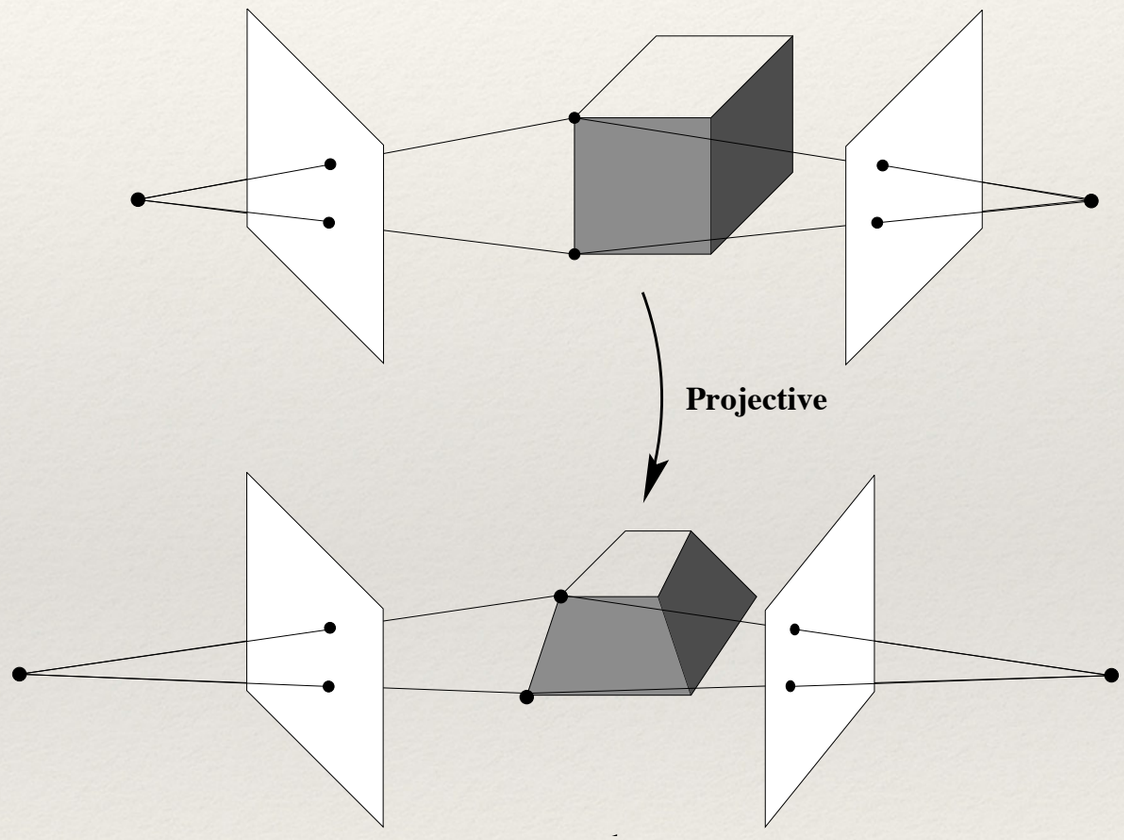


Projective ambiguity

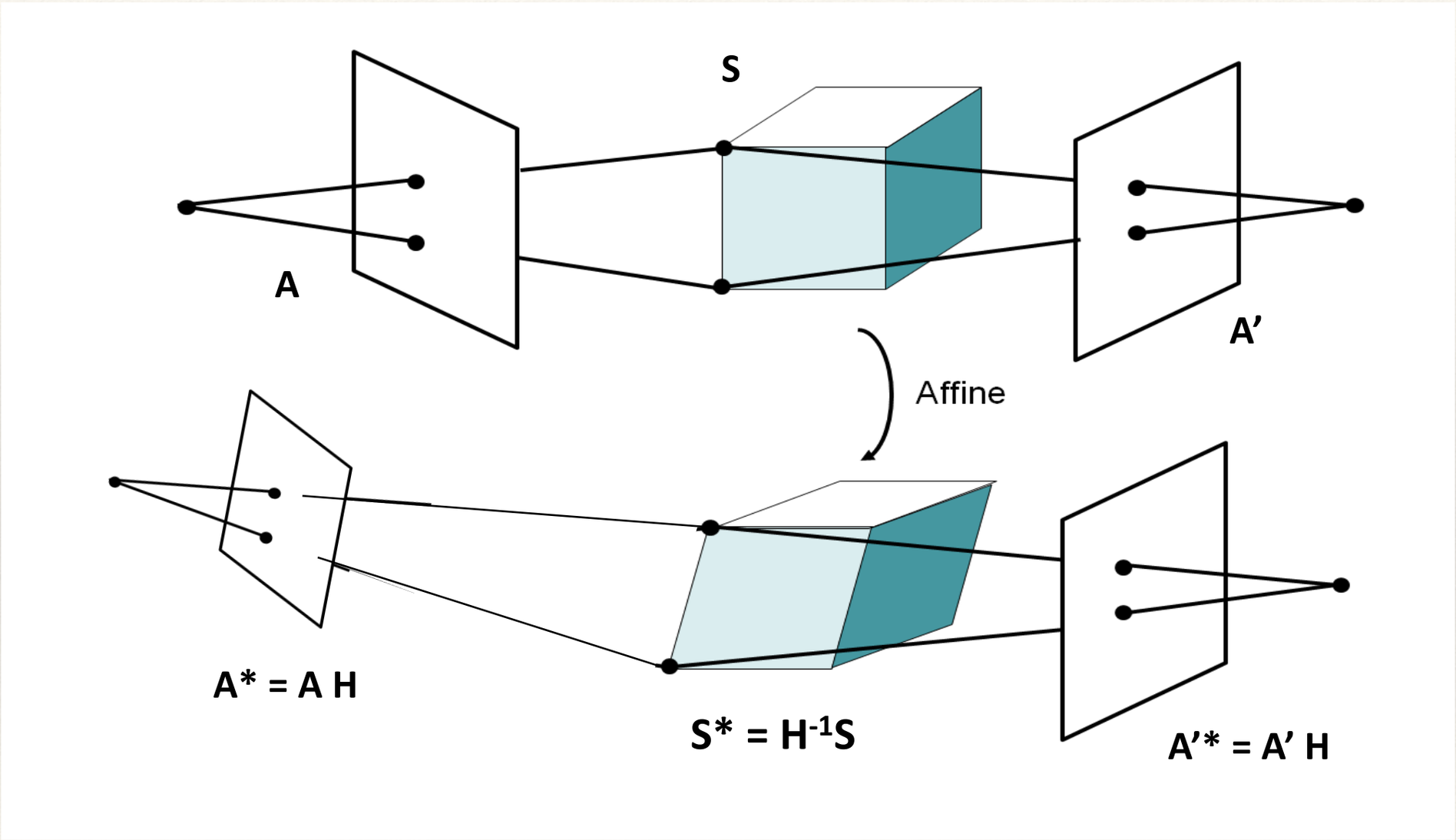
- If nothing is known of the calibration of either camera, nor the placement of one camera with respect to the other, then the ambiguity of reconstruction is expressed by an arbitrary projective transformation

- H is any 4 × 4 invertible matrix, representing a projective transformation of IP^3 , then replacing points $\mathbf{X}_i$ by $H\mathbf{X}_i$ and matrices P and P' by PH^{-1} and $P'H^{-1}$ does not change the image points.

- Thus reconstruction from uncalibrated cameras is possible *up to a* ***projective transformation***



Affine ambiguity



Computation of the fundamental matrix

- 8-point algorithm
- Least squares solution using SVD on equations from 8 pairs of correspondences
- Enforce det(F)=0 constraint using SVD on F

8-point algorithm

1. Solve a system of homogeneous linear equations

   (1) Write down the system of equations

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

$$uu'f_{11} + uv'f_{12} + uf_{13} + vu'f_{21} + vv'f_{22} + vf_{23} + u'f_{31} + v'f_{32} + f_{33} = 0$$

$$\mathrm{A}\boldsymbol{f} = \begin{bmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & u_nv_n' & u_n & v_nu_n' & v_nv_n' & v_n & u_n' & v_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ \vdots \\ f_{33} \end{bmatrix} = \mathbf{0}$$

8-point algorithm

1. Solve a system of homogeneous linear equations

   (1) Write down the system of equations

   (2) Solve **f** from A**f**=**0** using SVD

Matlab:

```
[U, S, V] = svd(A);
f = V(:, end);
F = reshape(f, [3 3])’;
```

Python Numpy:

```
U, S, Vh = np.linalg.svd(A)
# V = Vh.T -> note = different from MATLAB
F = Vh[-1,:]
F = np.reshape(F, (3,3))
```

Need to enforce singularity constraint

Non-singular fundamental matrix

Corrected fundamental matrix

8-point algorithm

1. Solve a system of homogeneous linear equations

   (1) Write down the system of equations

   (2) Solve **f** from A**f**=**0** using SVD

2. Resolve det(F) = 0 constraint using SVD

Matlab:

```
[U, S, V] = svd(F);
S(3,3) = 0;
F = U*S*V';
```

Python Numpy:

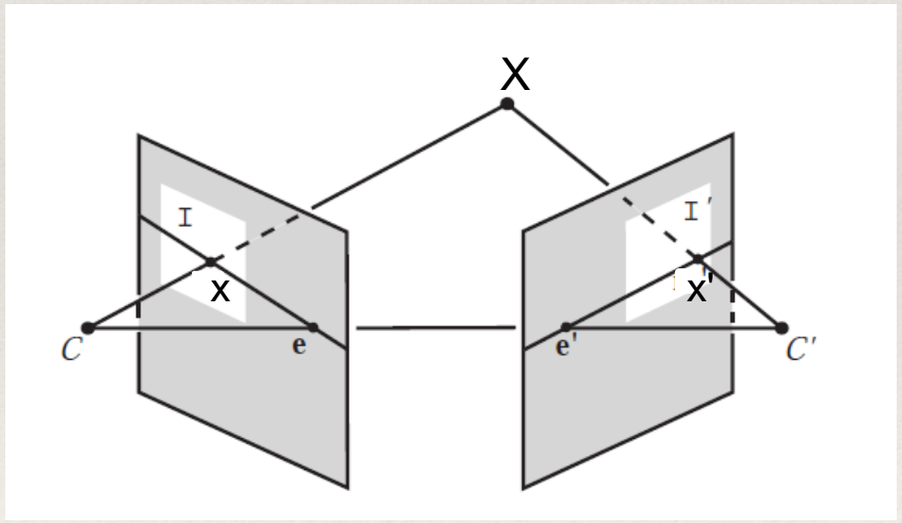
```
U, S, Vh = np.linalg.svd(F)
S[-1] = 0
F = U @ np.diagflat(S) @ Vh
```

The normalized eight-point algorithm

- Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels
- Use the eight-point algorithm to compute ***F*** from the normalized points
- Enforce the rank-2 constraint (for example, take SVD of ***F*** and throw out the smallest singular value)
- Transform fundamental matrix back to original units: if ***T*** and ***T'*** are the normalizing transformations in the two images, than the fundamental matrix in original coordinates is $\boldsymbol{T'^T F T}$

"Gold standard" algorithm

- Use 8-point algorithm to get initial value of F
- Use F to solve for P and P′ (discussed later)
- Jointly solve for 3d points **X** and **F** that minimize the squared re-projection error



See Algorithm 11.2 and Algorithm 11.3 in Hartley-Zisserman (pages 284-285) for details

Structure from motion

- Given a set of corresponding image points $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$,
  - what are the cameras P, P' ($\mathbf{x_i} = P\mathbf{X_i}, \mathbf{x_i'} = P'\mathbf{X_i}$) and 3-D point $\mathbf{X}_i$?
  - **uncalibrated camera:** K, R, t are unknown
- General outline:
  I. Compute the fundamental matrix from point correspondences
  II. Compute the camera matrices from the fundamental matrices
  III. For each point correspondence $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$, compute the points in 3-D space

Retrieving the camera matrices

- We can get projection matrices P and P’ up to a projective ambiguity

K’*rotation

$$\mathbf{P} = [\mathbf{I} \mid \mathbf{0}] \qquad \mathbf{P}' = [[\mathbf{e}']_\times \mathbf{F} \mid \mathbf{e}'] \qquad \mathbf{e}'^T \mathbf{F} = 0$$

K’*translation epipole

If we know the intrinsic matrices (K and K’), we can resolve the ambiguity

See Hartley-Zisserman (pages 255-256) for details

Structure from motion

- Given a set of corresponding image points $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$,
  - what are the cameras P, P' ($\mathbf{x_i} = P\mathbf{X_i}, \mathbf{x_i'} = P'\mathbf{X_i}$) and 3-D point $\mathbf{X}_i$?
  - **uncalibrated camera:** K, R, t are unknown
- General outline:
  I. Compute the fundamental matrix from point correspondences
  II. Compute the camera matrices from the fundamental matrices
  III. For each point correspondence $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$, compute the points in 3-D space

Triangulation

- There are errors in the measured points x and x’
- The back-projected rays don’t intersect at X
  - there will *not* be a point X which exactly satisfies x = PX, x′= P′X
  - the image points do *not* satisfy the epipolar constraint $\mathbf{x}'^{T}F\mathbf{x} = 0$.
- P and P′ are with great accuracy comparing with image point measurement

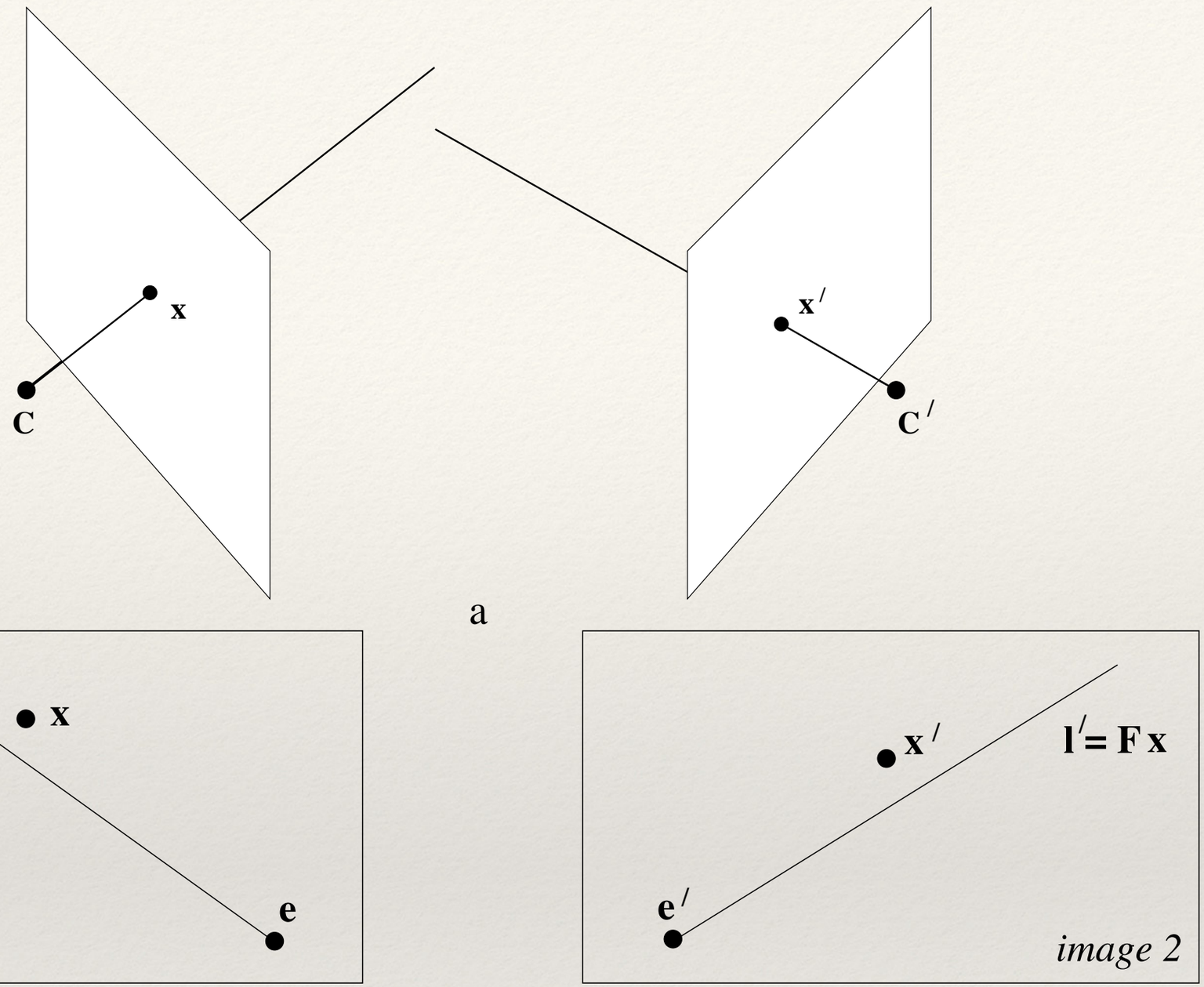


Fig. 12.1. *(a) Rays back-projected from imperfectly measured points* $\mathbf{x}, \mathbf{x}'$ *are skew in 3-space in general. (b) The epipolar geometry for* $\mathbf{x}, \mathbf{x}'$*. The measured points do not satisfy the epipolar constraint. The epipolar line* $\mathbf{l}' = \mathtt{F}\mathbf{x}$ *is the image of the ray through* $\mathbf{x}$*, and* $\mathbf{l} = \mathtt{F}^\mathsf{T}\mathbf{x}'$ *is the image of the ray through* $\mathbf{x}'$*. Since the rays do not intersect,* $\mathbf{x}'$ *does not lie on* $\mathbf{l}'$*, and* $\mathbf{x}$ *does not lie on* $\mathbf{l}$*.*

Linear method

- x = PX, x′ = P′X
- x × (PX) = 0 $\longrightarrow$

$$
\begin{aligned}
x(\mathbf{p}^{3\mathsf{T}}\mathbf{X}) - (\mathbf{p}^{1\mathsf{T}}\mathbf{X}) &= 0 \\
y(\mathbf{p}^{3\mathsf{T}}\mathbf{X}) - (\mathbf{p}^{2\mathsf{T}}\mathbf{X}) &= 0 \\
x(\mathbf{p}^{2\mathsf{T}}\mathbf{X}) - y(\mathbf{p}^{1\mathsf{T}}\mathbf{X}) &= 0
\end{aligned}
$$

- AX=0

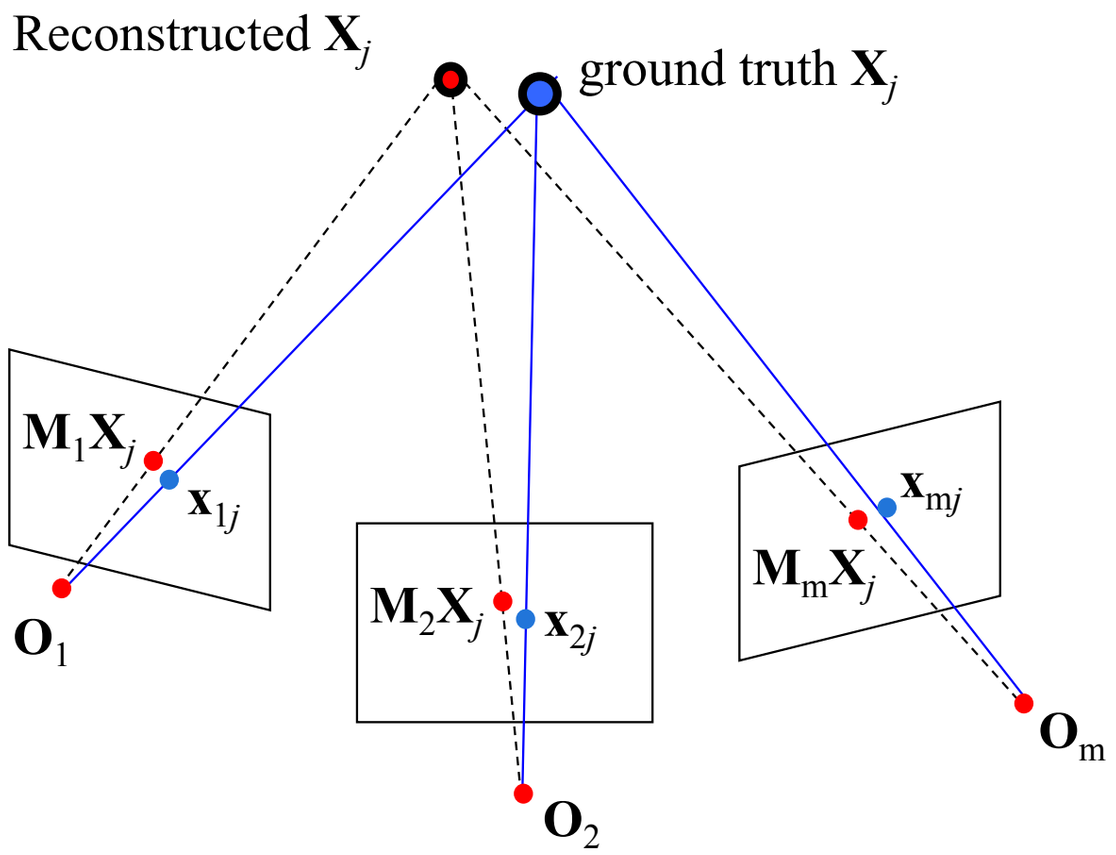
$$
\mathtt{A} = \begin{bmatrix} x\mathbf{p}^{3\mathsf{T}} - \mathbf{p}^{1\mathsf{T}} \\ y\mathbf{p}^{3\mathsf{T}} - \mathbf{p}^{2\mathsf{T}} \\ x'\mathbf{p}'^{3\mathsf{T}} - \mathbf{p}'^{1\mathsf{T}} \\ y'\mathbf{p}'^{3\mathsf{T}} - \mathbf{p}'^{2\mathsf{T}} \end{bmatrix}
$$

Find the solution using DLT via SVD

Bundle adjustment

- Non-linear method for refining structure and motion
- Minimizes re-projection error

$$E(M, \mathbf{X}) = \sum_{i=1}^{m} \sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, M_i \mathbf{X}_j\right)^2$$



Bundle adjustment

$$E(M, \mathbf{X}) = \sum_{i=1}^{m} \sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, M_i \mathbf{X}_j\right)^2$$

measurements **parameters**

D is the nonlinear mapping

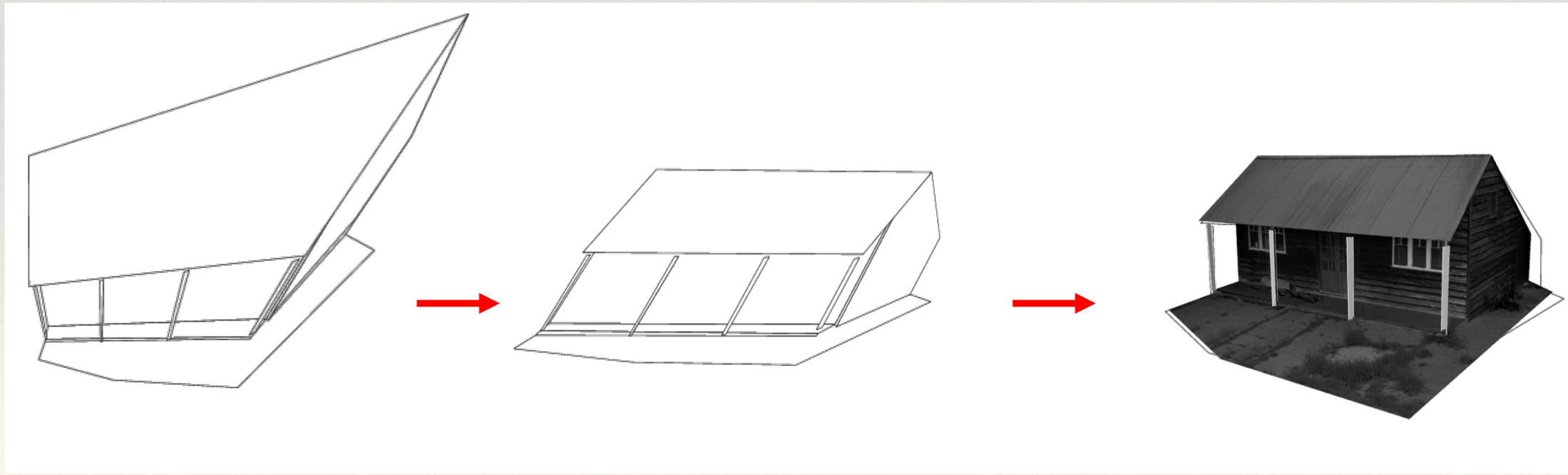
- Newton Method
- Levenberg-Marquardt Algorithm
  - Iterative, starts from initial solution
  - May be slow if initial solution far from real solution
  - Estimated solution may be function of the initial solution
  - Newton requires the computation of J, H
  - Levenberg-Marquardt doesn't require the computation of H

From epipolar geometry to camera calibration

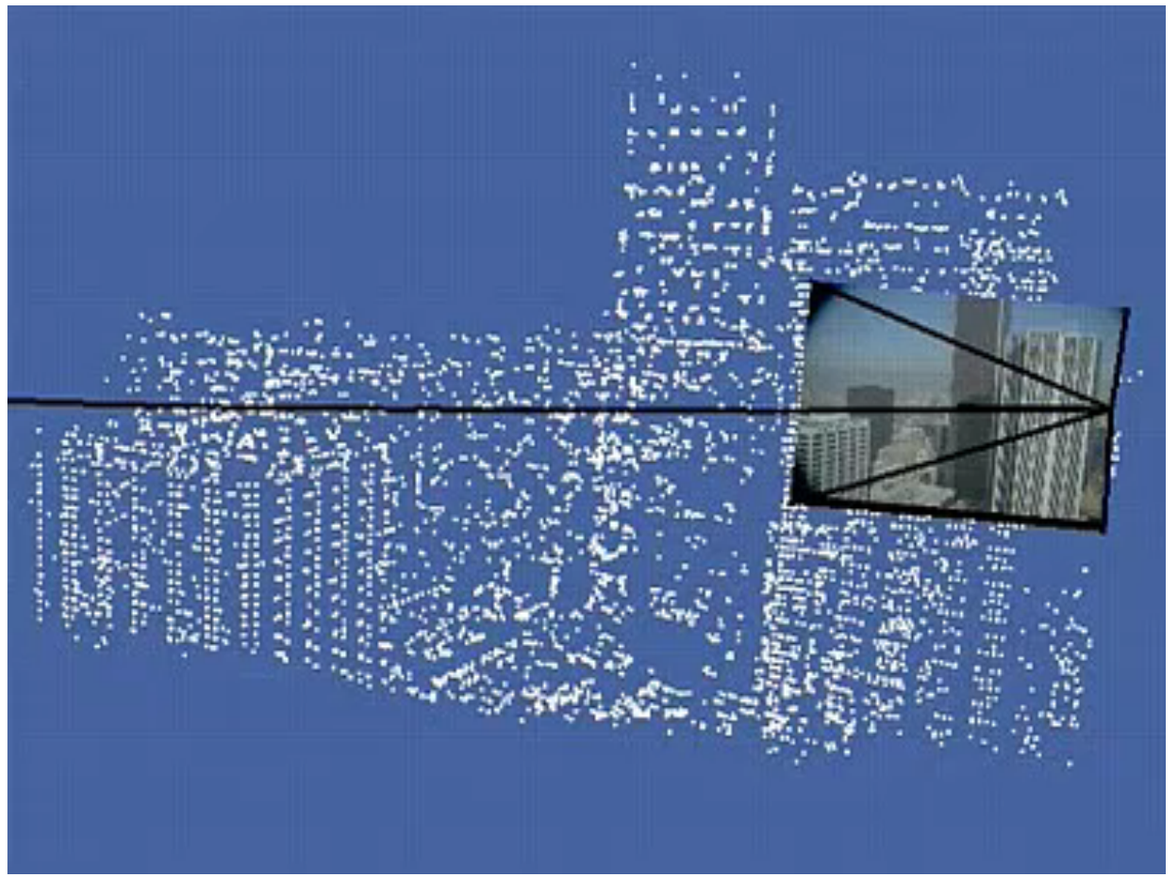
- If we know the calibration matrices of the two cameras, we can estimate the essential matrix: $E = K^T F K'$
- The essential matrix gives us the relative rotation and translation between the cameras, or their extrinsic parameters.
- Fundamental matrix lets us compute relationship up to scale for cameras with unknown intrinsic calibrations.
- Estimating the fundamental matrix is a kind of “weak calibration”

Self calibration

- Self-calibration is the problem of recovering the metric reconstruction from the perspective (or affine) reconstruction
- We can self-calibrate the camera by making some assumptions about the cameras



Application



Courtesy of Oxford **Visual Geometry Group**

Lucas & Kanade, 81
Chen & Medioni, 92
Debevec et al., 96
Levoy & Hanrahan, 96
Fitzgibbon & Zisserman, 98
Triggs et al., 99
Pollefeys et al., 99
Kutulakos & Seitz, 99

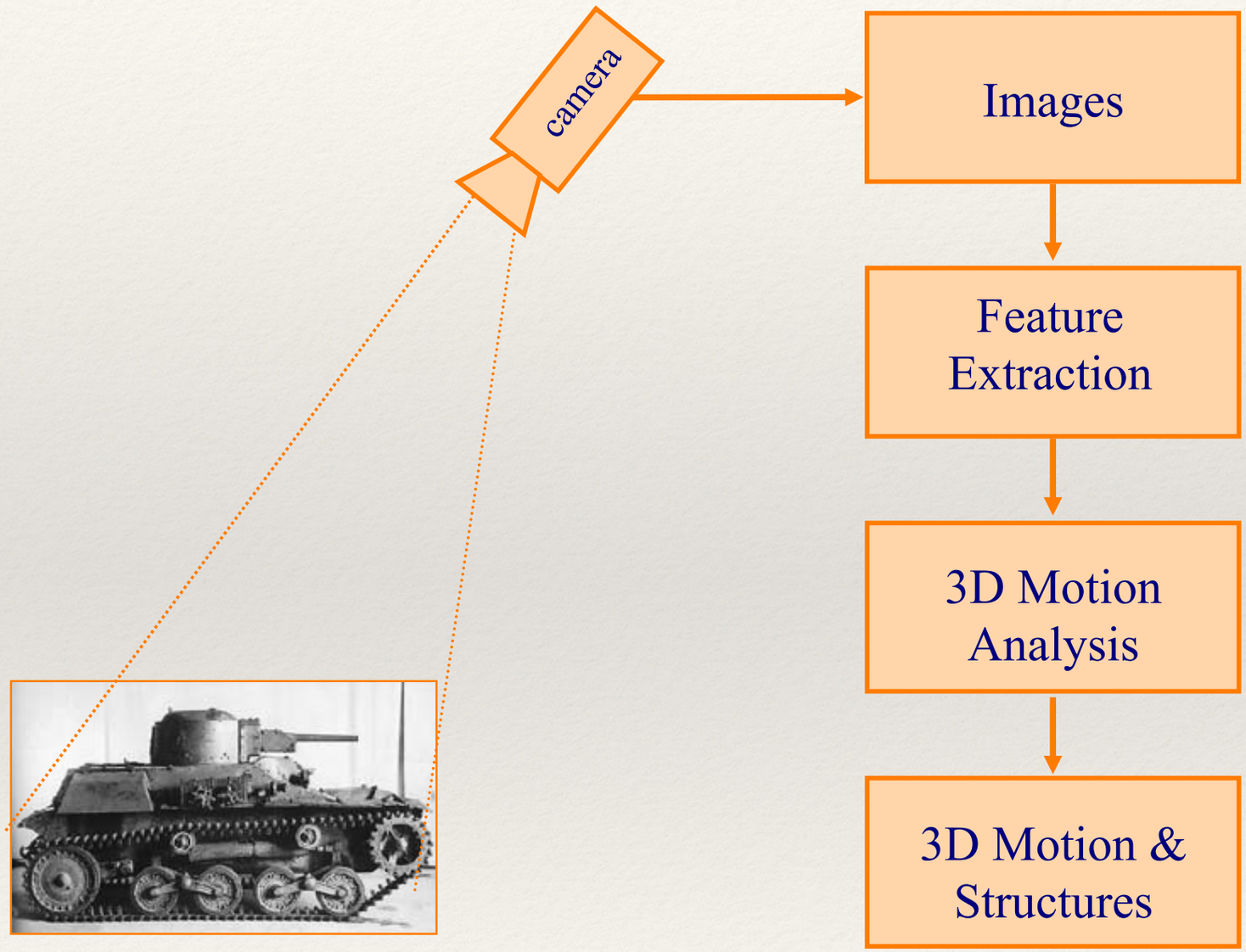
Levoy et al., 00
Hartley & Zisserman, 00
Dellaert et al., 00
Rusinkiewic et al., 02
Nistér, 04
Brown & Lowe, 04
Schindler et al, 04
Lourakis & Argyros, 04
Colombo et al. 05

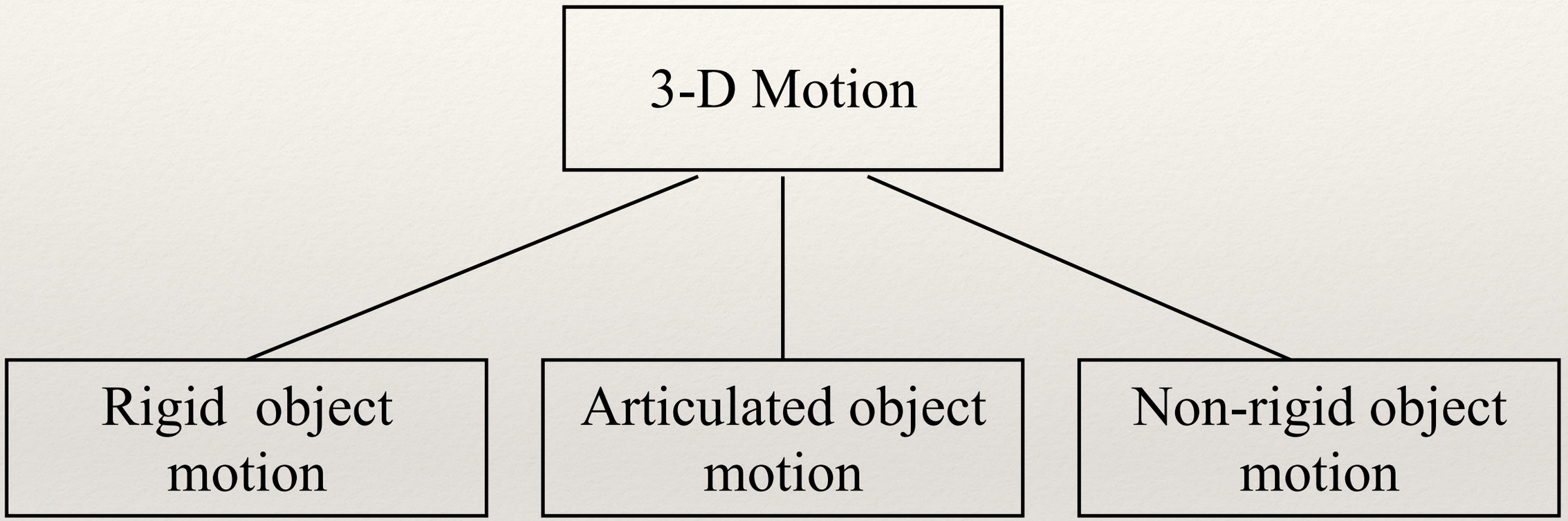
Golparvar-Fard, et al. JAEI 10
Pandey et al. IFAC , 2010
Pandey et al. ICRA 2011
Microsoft's PhotoSynth
Snavely et al., 06-08
Schindler et al., 08
Agarwal et al., 09
Frahm et al., 10

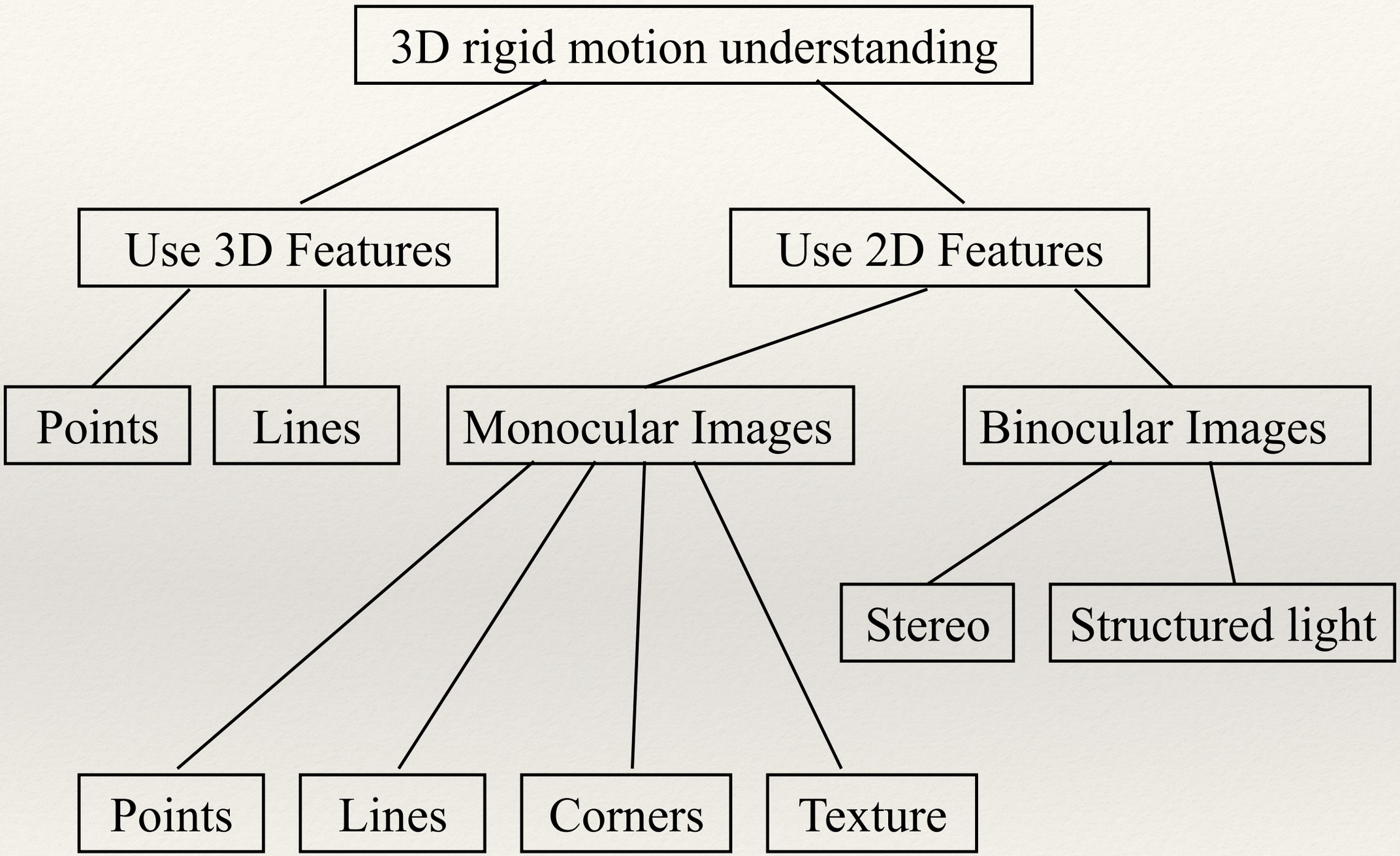
Application

Noah Snavely, Steven M. Seitz, Richard Szeliski, "Photo tourism: Exploring photo collections in 3D," ACM Transactions on Graphics (SIGGRAPH Proceedings),2006,

Motion and structure: motion perspective




3-D Motion
Rigid object motion
Articulated object motion
Non-rigid object motion


3D rigid motion understanding
Use 3D Features
Use 2D Features
Points
Lines
Monocular Images
Binocular Images
Stereo
Structured light
Points
Lines
Corners
Texture

Motion expression

- Motion of a rigid object in 3D is usually expressed as a rotation around system origin followed by a translation.

a 3D point of an object at time t_1:

$$\vec{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

the point after motion at t_2:

$$\vec{p}' = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

R be the rotation, a 3*3 orthogonal matrix

$$\vec{T} = \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}$$ be translation vector.

Then, from time t_1 to t_2 :

$$\vec{p}' = R\vec{p} + \vec{T}$$

Motion model-rotation matrix

- When specifying the motion by 9 components of the rotation matrix, we have 6 constraints

$$
\begin{aligned}
r_{11}^2 + r_{12}^2 + r_{13}^2 &= 1 \\
r_{21}^2 + r_{22}^2 + r_{23}^2 &= 1 \\
r_{31}^2 + r_{32}^2 + r_{33}^2 &= 1 \\
r_{22}r_{33} - r_{23}r_{32} &= r_{11} \\
r_{23}r_{31} - r_{21}r_{33} &= r_{12} \\
r_{21}r_{32} - r_{22}r_{31} &= r_{13}.
\end{aligned}
$$

12 free parameters- 6 constraints = 6 free parameters

Note: There are only 3 independent parameters in R

- Some properties of **R**

$$\left\|\vec{r}^{(i)}\right\|^2 = 1, \quad \vec{r}^{(i)} \bullet \vec{r}^{(k)} = 0, \quad i = 1,2,3, \quad i \neq k$$

$$R^{-1} = R^T \quad or \quad RR^T = R^T R = 1$$

$$\left\|R\vec{U}\right\| = \left\|\vec{U}\right\|$$

$$R\vec{U} \bullet R\vec{V} = \vec{U} \bullet \vec{V}$$

$$R\vec{U} \times R\vec{V} = R(\vec{U} \times \vec{V})$$

Motion model - Euler angles

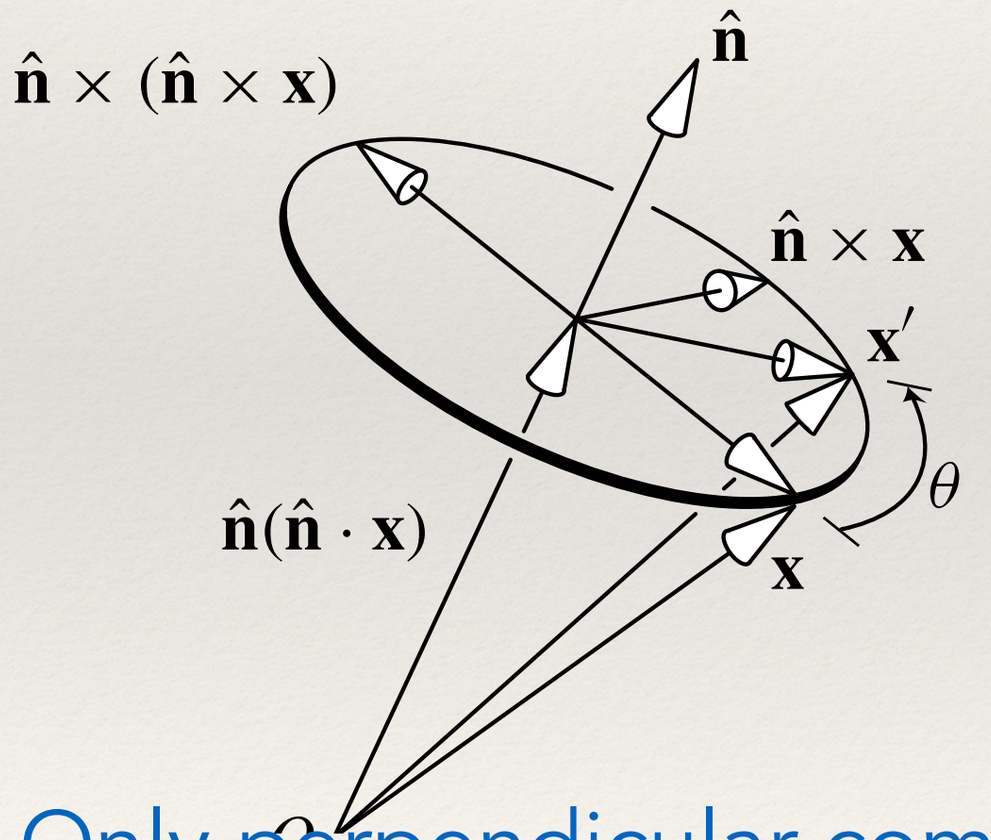
- A rotation can be represented by three successive rotations around x, y and z axis by angles respectively

$$\boldsymbol{R} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\psi & 0 & -\sin\psi \\ 0 & 1 & 0 \\ \sin\psi & 0 & \cos\psi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

six free parameters: 3 angles + 3 translations

Motion model - Axis/angle (exponential twist)

- A rotation can be represented by a rotation axis $\hat{n}$ and an angle θ: $\hat{n}$ is unit vector



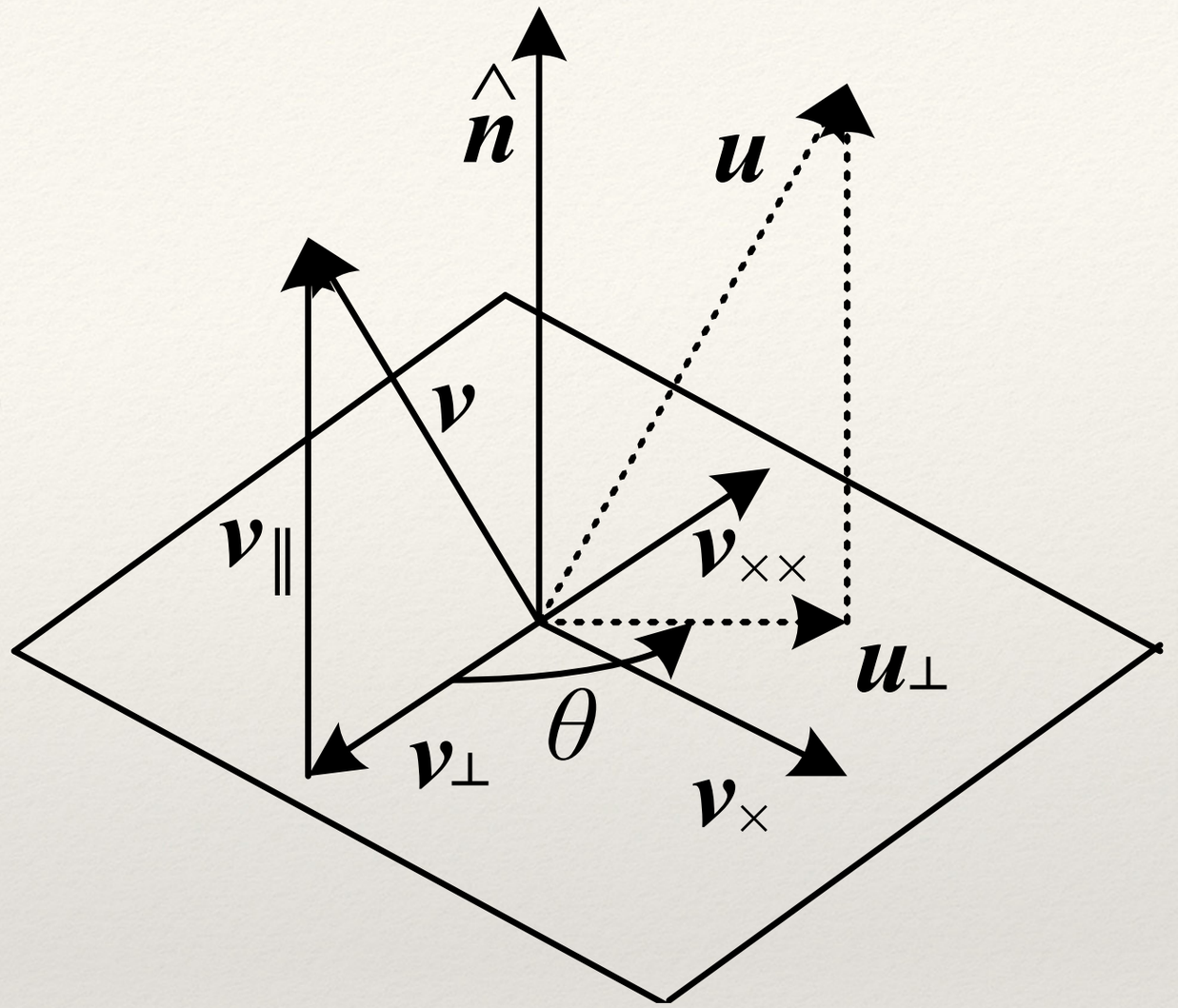
$$\hat{n} = \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix}$$

Only perpendicular component is affected by the rotation

$$\boldsymbol{v}_{\parallel} = \hat{\boldsymbol{n}}(\hat{\boldsymbol{n}} \cdot \boldsymbol{v}) = (\hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^T)\boldsymbol{v}$$

$$\boldsymbol{v}_{\perp} = \boldsymbol{v} - \boldsymbol{v}_{\parallel} = (\boldsymbol{I} - \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^T)\boldsymbol{v}.$$

$$\boldsymbol{v}_{\times} = \hat{\boldsymbol{n}} \times \boldsymbol{v} = [\hat{\boldsymbol{n}}]_{\times}\boldsymbol{v}$$



$$[\hat{\boldsymbol{n}}]_{\times} = \begin{bmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{bmatrix}$$

$$\boldsymbol{v}_{\times\times} = \hat{\boldsymbol{n}} \times \boldsymbol{v}_{\times} = [\hat{\boldsymbol{n}}]_{\times}^2\boldsymbol{v} = -\boldsymbol{v}_{\perp}$$

$$\boldsymbol{v}_{\parallel} = \boldsymbol{v} - \boldsymbol{v}_{\perp} = \boldsymbol{v} + \boldsymbol{v}_{\times\times} = (\boldsymbol{I} + [\hat{\boldsymbol{n}}]_{\times}^2)\boldsymbol{v}$$

$$\boldsymbol{u}_\perp = \cos\theta \boldsymbol{v}_\perp + \sin\theta \boldsymbol{v}_\times = (\sin\theta[\hat{\boldsymbol{n}}]_\times - \cos\theta[\hat{\boldsymbol{n}}]_\times^2)\boldsymbol{v}.$$

$$\boldsymbol{u} = \boldsymbol{u}_\perp + \boldsymbol{v}_\parallel = (\boldsymbol{I} + \sin\theta[\hat{\boldsymbol{n}}]_\times + (1-\cos\theta)[\hat{\boldsymbol{n}}]_\times^2)\boldsymbol{v}$$

Rodriguez's formula

$$\boldsymbol{R}(\hat{\boldsymbol{n}}, \theta) = \boldsymbol{I} + \underline{\sin\theta[\hat{\boldsymbol{n}}]_\times} + \underline{(1-\cos\theta)[\hat{\boldsymbol{n}}]_\times^2}$$

skew-symmetric part

symmetric part

For the case where the rotation angle is small:

$$\boldsymbol{\omega} = \theta\hat{\boldsymbol{n}} = (\omega_x, \omega_y, \omega_z)$$

$$\boldsymbol{R}(\boldsymbol{\omega}) \approx \boldsymbol{I} + \sin\theta[\hat{\boldsymbol{n}}]_\times \approx \boldsymbol{I} + [\theta\hat{\boldsymbol{n}}]_\times = \begin{bmatrix} 1 & -\omega_z & \omega_y \\ \omega_z & 1 & -\omega_x \\ -\omega_y & \omega_x & 1 \end{bmatrix}$$

exponential twist: $\boldsymbol{R}(\hat{\boldsymbol{n}}, \theta) = \lim_{k\to\infty}(\boldsymbol{I} + \frac{1}{k}[\theta\hat{\boldsymbol{n}}]_\times)^k = \exp[\boldsymbol{\omega}]_\times$

$$\begin{aligned} \exp[\boldsymbol{\omega}]_\times &= \boldsymbol{I} + \theta[\hat{\boldsymbol{n}}]_\times + \frac{\theta^2}{2}[\hat{\boldsymbol{n}}]_\times^2 + \frac{\theta^3}{3!}[\hat{\boldsymbol{n}}]_\times^3 + \cdots \\ &= \boldsymbol{I} + (\theta - \frac{\theta^3}{3!} + \cdots)[\hat{\boldsymbol{n}}]_\times + (\frac{\theta^2}{2} - \frac{\theta^3}{4!} + \cdots)[\hat{\boldsymbol{n}}]_\times^2 \\ &= \boldsymbol{I} + \sin\theta[\hat{\boldsymbol{n}}]_\times + (1 - \cos\theta)[\hat{\boldsymbol{n}}]_\times^2, \end{aligned}$$

- Considering translation, for axis/angle motion model, the free parameters are:

$$n_1, n_2, n_3, \theta, t1, t2, t3$$

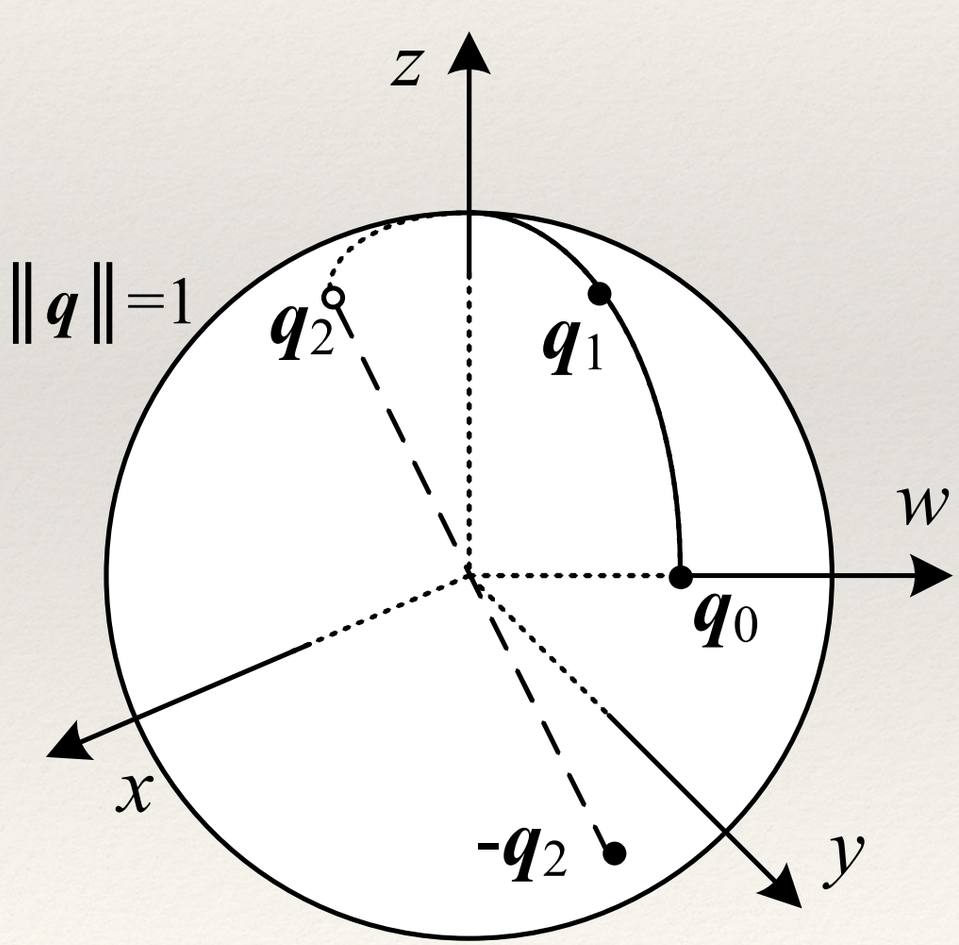
- But with the constraint (unit vector):

$$n_1^2 + n_2^2 + n_3^2 = 1$$

the number of free parameters will be 6

Motion models - quaternions

- Closely related with axis/angle representation
- A unit length 4 vector: q = (x,y,z,w)
- Advantage: continuous representation



From Rodriguez's formula:

$$\boldsymbol{R}(\hat{\boldsymbol{n}}, \theta) = \boldsymbol{I} + \sin\theta[\hat{\boldsymbol{n}}]_\times + (1 - \cos\theta)[\hat{\boldsymbol{n}}]_\times^2$$

and the formula:

$$\boldsymbol{q} = (\boldsymbol{v}, w) = (\sin\frac{\theta}{2}\hat{\boldsymbol{n}}, \cos\frac{\theta}{2})$$

we have:

$$\begin{aligned} \boldsymbol{R}(\hat{\boldsymbol{n}}, \theta) &= \boldsymbol{I} + \sin\theta[\hat{\boldsymbol{n}}]_\times + (1 - \cos\theta)[\hat{\boldsymbol{n}}]_\times^2 \\ &= \boldsymbol{I} + 2w[\boldsymbol{v}]_\times + 2[\boldsymbol{v}]_\times^2. \end{aligned}$$

The matrix form:

$$\boldsymbol{R}(\boldsymbol{q}) = \begin{bmatrix} 1-2(y^2+z^2) & 2(xy-zw) & 2(xz+yw) \\ 2(xy+zw) & 1-2(x^2+z^2) & 2(yz-xw) \\ 2(xz-yw) & 2(yz+xw) & 1-2(x^2+y^2) \end{bmatrix}$$

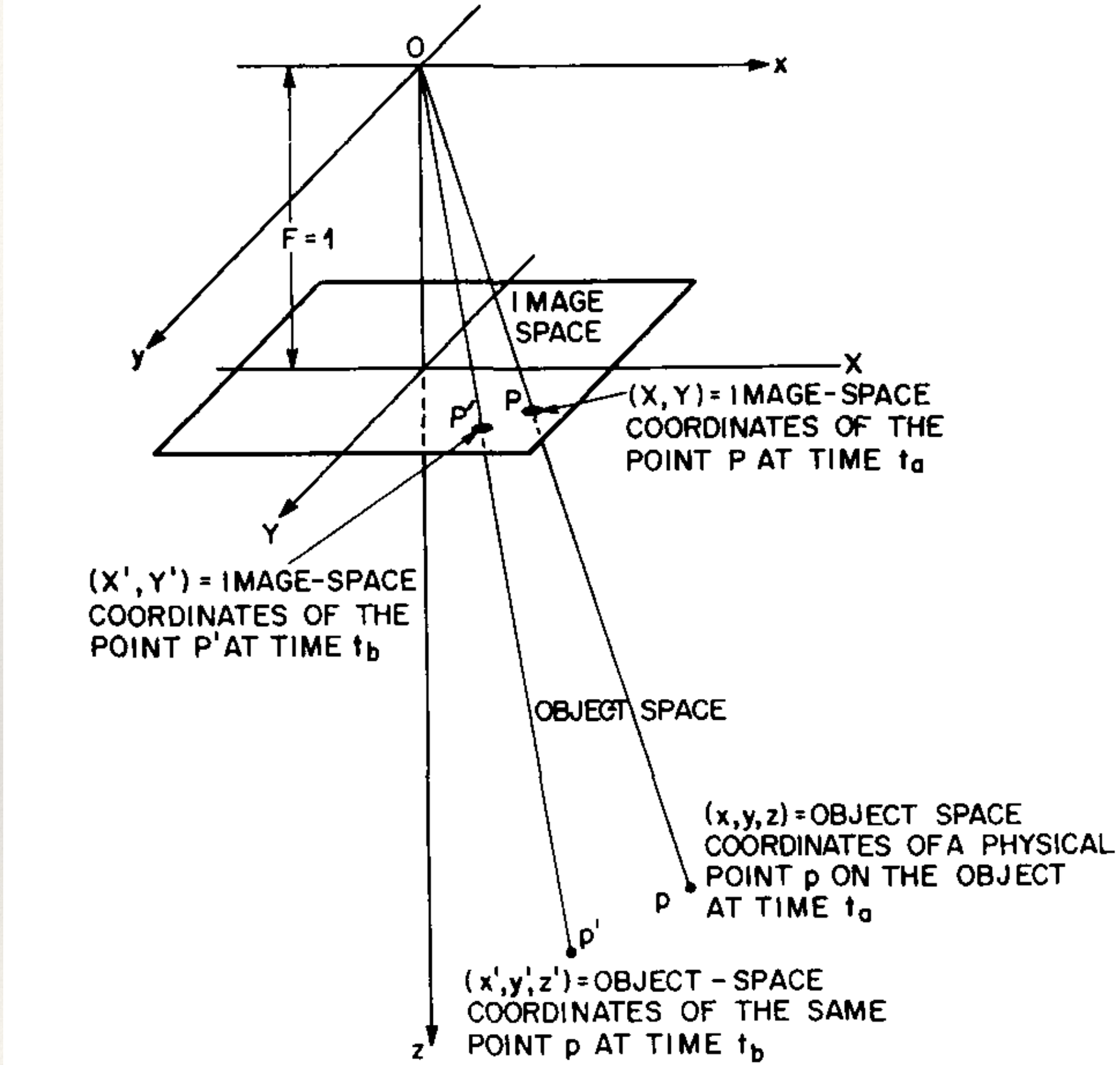
$$\boldsymbol{q}_2 = \boldsymbol{q}_0\boldsymbol{q}_1 = (\boldsymbol{v}_0 \times \boldsymbol{v}_1 + w_0\boldsymbol{v}_1 + w_1\boldsymbol{v}_0,\ w_0w_1 - \boldsymbol{v}_0 \cdot \boldsymbol{v}_1)$$

$$\boldsymbol{R}(\boldsymbol{q}_2) = \boldsymbol{R}(\boldsymbol{q}_0)\boldsymbol{R}(\boldsymbol{q}_1)$$

So it is easier to interpolate between rotations and to chain rigid transformations.

Motion from Feature Correspondence (FC)

	Point	Line
3D-to-3D	Find motion parameters R and T from the feature correspondences	
2D-to-3D		
2D-to-2D		



Huang and Netravali, 1994

Motion from 3D-to-3D FCs

Applications

- Motion estimation of rigid body
- Motion estimation using stereo or other range finding devices
- positioning a known 3D object using stereo or other range-finding devices.

Point features

(1) Method 1:

3 or more points on a rigid object (at least 3 points not collinear) can be seen over two-time instances t_1 and t_2

$$\vec{p}_1' = R\vec{p}_1 + \vec{T} \quad (1)$$

$$\vec{p}_2' = R\vec{p}_2 + \vec{T} \quad (2)$$

$$\vec{p}_3' = R\vec{p}_3 + \vec{T} \quad (3)$$

Subtract (1) from (2), (3) respectively:

$$\vec{p}_2' - \vec{p}_1' = R(\vec{p}_2 - \vec{p}_1) \quad (4)$$

$$\vec{p}_3' - \vec{p}_1' = R(\vec{p}_3 - \vec{p}_1) \quad (5)$$

(4)×(5)

$$(\vec{p}_2' - \vec{p}_1') \times (\vec{p}_3' - \vec{p}_1') = R[(\vec{p}_2 - \vec{p}_1) \times (\vec{p}_3 - \vec{p}_1)] \quad (6)$$

From (4),(5) and (6), **R** can be solved,

(never forget rotation matrix constraints).

After finding **R**

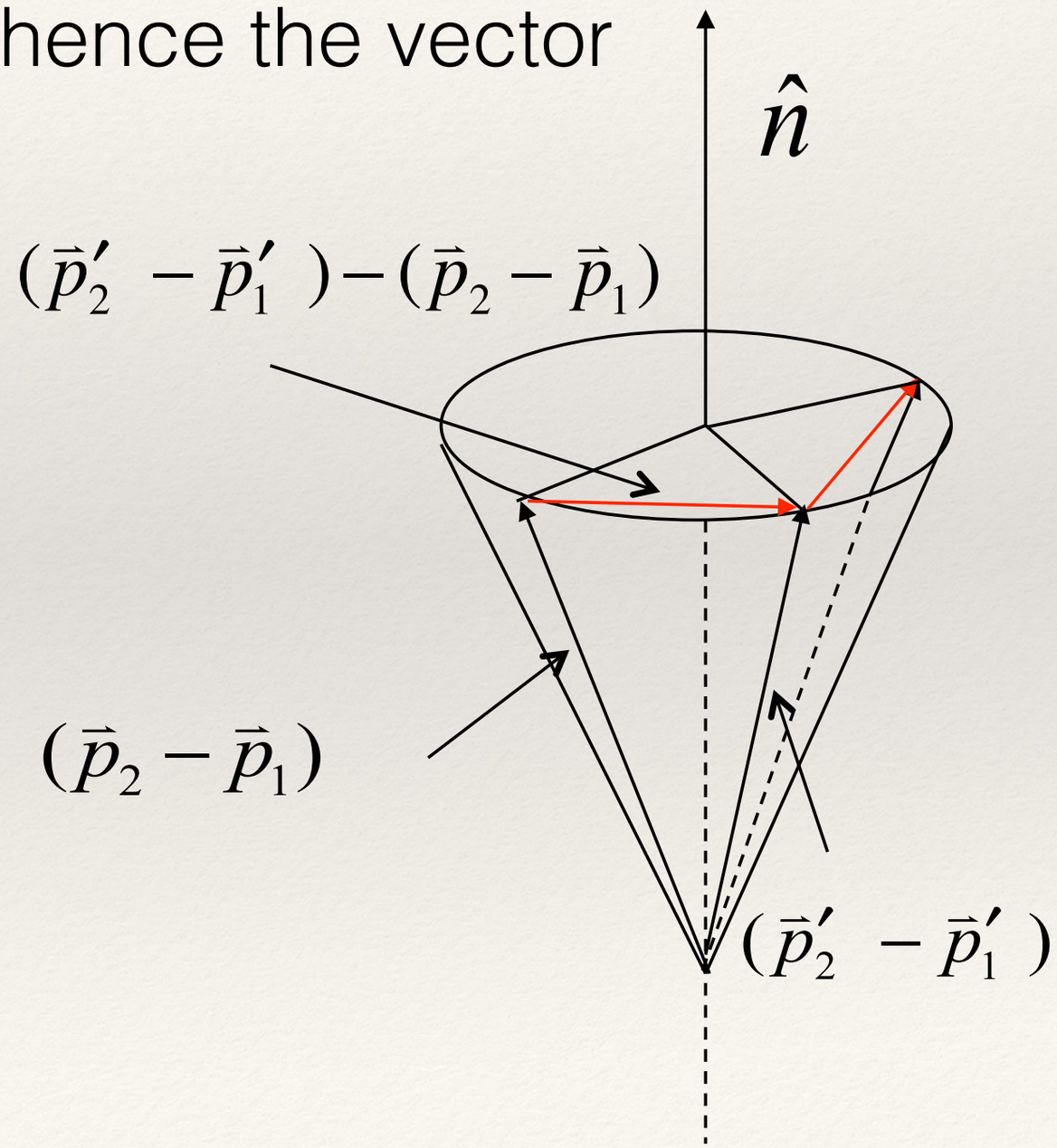
$$\vec{T} = \frac{1}{3}\sum_{i=1}^{3}(\vec{p}_i' - R\vec{p}_i)$$

(2) Method 2:

From eq (4), vector $\vec{p}_2 - \vec{p}_1$ to $\vec{p}_2' - \vec{p}_1'$ subjects to a pure rotation, hence the vector

$(\vec{p}_2' - \vec{p}_1') - (\vec{p}_2 - \vec{p}_1)$

is perpendicular to rotation axis $\hat{n}$ .The same is true for $\vec{p}_3 - \vec{p}_1$



Thus,

$$\hat{n} = \frac{[(\vec{p}_2' - \vec{p}_1') - (\vec{p}_2 - \vec{p}_1)] \times [(\vec{p}_3' - \vec{p}_1') - (\vec{p}_3 - \vec{p}_1)]}{\left\| [(\vec{p}_2' - \vec{p}_1') - (\vec{p}_2 - \vec{p}_1)] \times [(\vec{p}_3' - \vec{p}_1') - (\vec{p}_3 - \vec{p}_1)] \right\|}$$

And rotation angle can be determined by the angle of two planes

$$\hat{n} \times (\vec{p}_2 - \vec{p}_1) \qquad \hat{n} \times (\vec{p}_2' - \vec{p}_1')$$

So,

$$\cos\theta = \frac{[\hat{n} \times (\vec{p}_2 - \vec{p}_1)] \bullet [\hat{n} \times (\vec{p}_2' - \vec{p}_1')]}{\left\| [\hat{n} \times (\vec{p}_2 - \vec{p}_1)] \right\| \bullet \left\| [\hat{n} \times (\vec{p}_2' - \vec{p}_1')] \right\|}$$

At least 3 not collinear 3D points on rigid object is needed to determine 3D motion!

(3) Method 3:

In practice, since the point correspondences are obtained from measurements, they are subject to error and therefore one prefers to work with more than three point correspondences. It turns to a least squares problem:

$$\underset{\text{w.r.t. } \boldsymbol{R}, \boldsymbol{t}}{\text{minimize}} \left\{ \sum_{i=1}^{N} ||\boldsymbol{p}'_i - (\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t})||^2 \right\}$$

subject to the constraint that R is a rotation matrix. Here ||.|| represents the Euclidean norm.

Line features

- Problem: given corresponding $(l_i, l_i'), i=1, \cdots, N$, find R and T.

- Assumption: these lines are considered infinitely long and that no point correspondences on these lines are assumed to be known.

- Solution: It can be proved that two nonparallel "sensed" lines are necessary and sufficient to determine R and T uniquely.

(1) **Method 1**: From the nonparallel sensed lines, we can generate three noncollinear point correspondences and then use the algorithms mentioned above.

(2) **Method 2**: if v_i and v_i^’ are unit vectors along the direction of lines l_i and l_i^’ (i=1,2) respectively, and if

$$v_3 = v_1 \times v_2$$
$$v_3' = v_1' \times v_2'$$

then

$$v_i' = \boldsymbol{R} v_i, i = 1, \cdots, 3.$$

So R can be determined. How to find T?

Motion from 2D-to-3D FCs

Applications

- Single camera calibration, i.e., determination of position and orientation of a camera knowing some features of 3D objects as imaged on the camera.

- Passive navigation of a vehicle using a single camera and based on the knowledge of 3D landmarks.

Point features

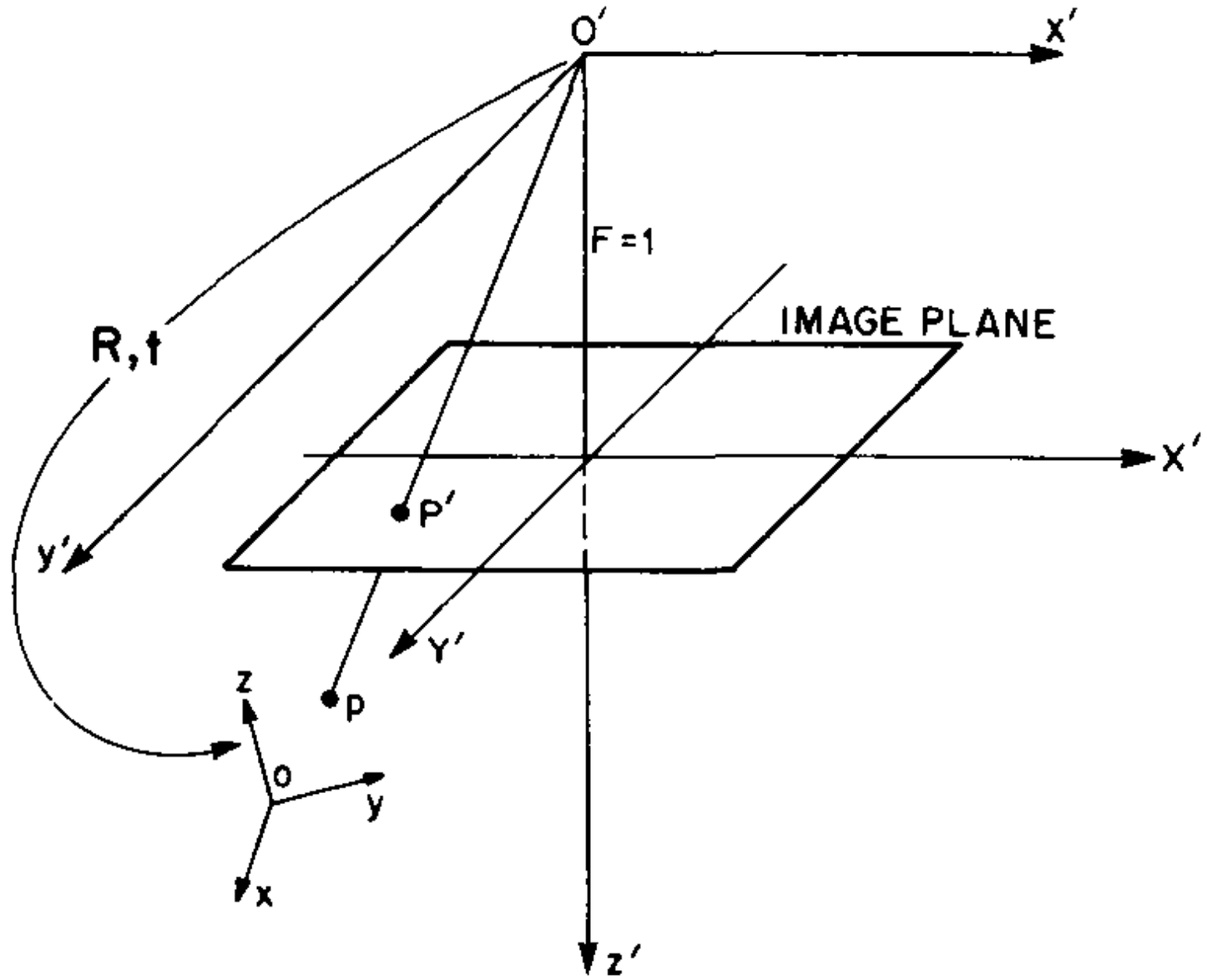


Fig. 2. Imaging geometry for 2D-to-3D correspondence problem. A point p in coordinate system $oxyz$ is imaged at location P' on the image plane which is specified in coordinate system $ox'y'z'$.

Huang and Netravali, 1994

$$X_i' = \frac{x_i'}{z_i'} \qquad Y_i' = \frac{y_i'}{z_i'}. \quad (1)$$

to determine R and T, the motion from the coordinate system $o'x'y'z'$ to $oxyz$ knowing the N point correspondences $(p_i, P_i')_{i=1,\cdots,N}$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \boldsymbol{R} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \boldsymbol{t}. \quad (2)$$

From (1) and (2), we have:

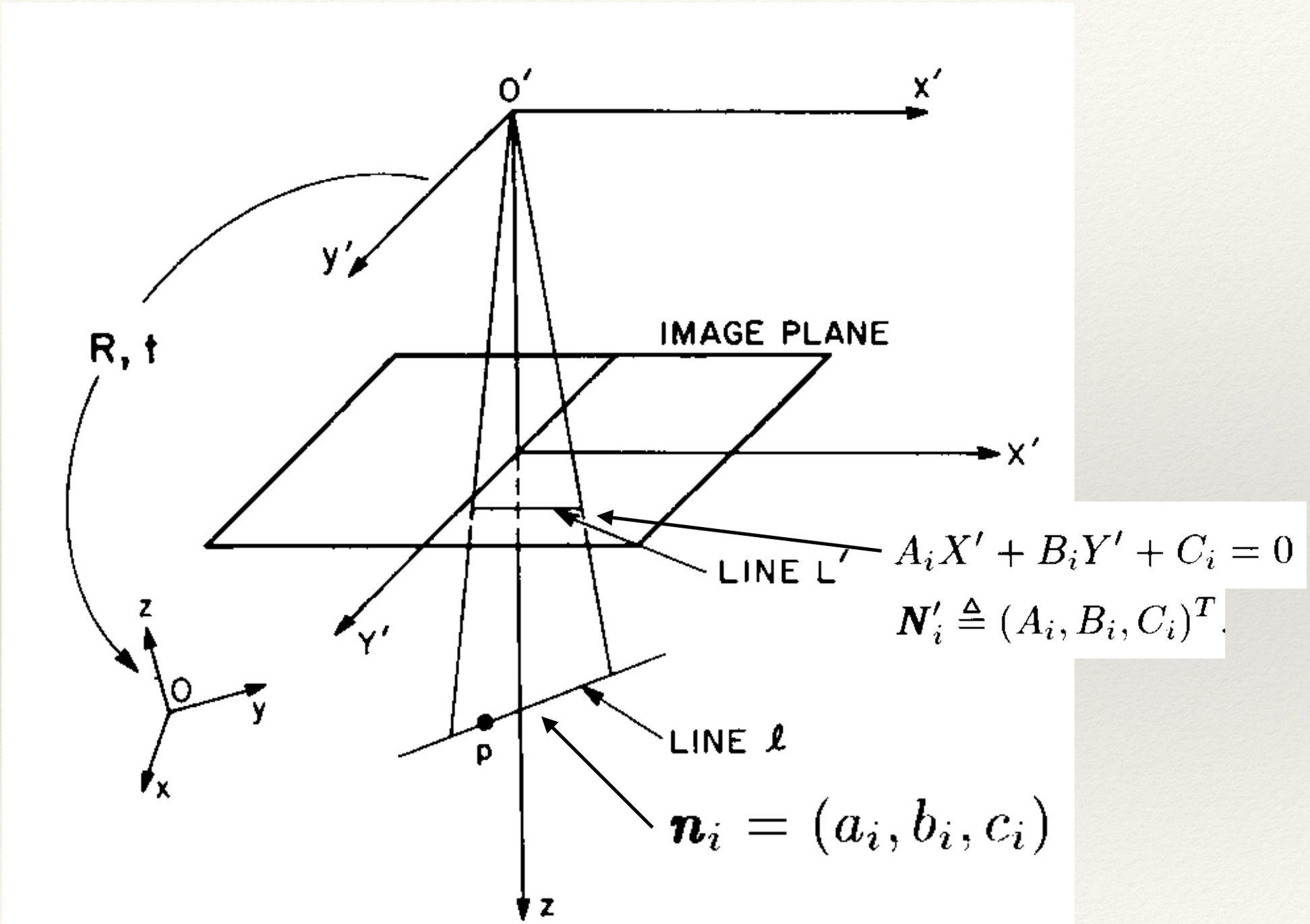
$$X_i' = \frac{r_{11}x_i + r_{12}y_i + r_{13}z_i + t_1}{r_{31}x_i + r_{32}y_i + r_{33}z_i + t_3}, \quad i = 1, \cdots, N$$

$$Y_i = \frac{r_{21}x_i + r_{12}y_i + r_{13}z_i + t_2}{r_{31}x_i + r_{32}y_i + r_{33}z_i + t_3}, \quad i = 1, \ldots, N.$$

Six unknowns. 3 PCs will generate 6 nonlinear equations. So need to iteratively solve a nonlinear least square problem:

$$\underset{\text{w.r.t. } \boldsymbol{R},\boldsymbol{t}}{\text{minimize}} \left\{ \sum_{i=1}^{N} \left(X_i' - \frac{r_{11}x_i + r_{12}y_i + r_{13}z_i + t_1}{r_{31}x_i + r_{32}y_i + r_{33}z_i + t_3} \right)^2 \right.$$
$$\left. + \left(Y_{i'} - \frac{r_{21}x_i + r_{22}y_i + r_{23}z_i + t_2}{r_{31}x_i + r_{32}y_i + r_{33}z_i + t_3} \right)^2 \right\}$$

Line features



Given N correspondences $(l_i, L_{i'})_{i=1,\cdots,N}$ determine R and T.

Liu, Huang and Faugeras Algorithms:

the plane containing o' and L' has an equation:

$$A_i x' + B_i y' + C_i z' = 0$$

l_i lies in this plane, so

$$\boldsymbol{N}_i'^T \boldsymbol{R} \boldsymbol{n}_i = 0.$$

With 3 LCs R can be find by solving the nonlinear transcendental equations iteratively.

Motion from 2D-to-2D FCs

Applications

- Finding relative attitudes of two cameras which are both observing the same 3D features.

- Estimating motion and structure of objects moving relative to a camera.

- Passive navigation, i.e., finding the relative attitude of a vehicle at two different time instants.

- Efficient coding and noise reduction of image sequences by estimating motion of objects.

Point features

Assumption

- Single stationary camera.
- Central projection model.
- Rigid moving object.
- Focus length f = 1, thus

$$X = \frac{x}{z}; \quad Y = \frac{y}{z};$$

- 3D point: $\vec{p} = (x, y, z)^T$
- 2D image: $\vec{P} = (X, Y, 1)^T = \frac{1}{z}\vec{p}$

Let 3D motion from $\vec{p}$ of time t_1 to $\vec{p}'$ of t_2

$$\vec{p}' = \mathrm{R}\ \vec{p} + \vec{\mathrm{T}} \qquad (1)$$

Where

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}; \qquad \vec{T} = \begin{pmatrix} T_x & T_y & T_z \end{pmatrix}^T$$

From equ (1)

$$z'\,\vec{P}' = z\,\mathrm{R}\,\vec{P} + \vec{\mathrm{T}} \qquad (2)$$

Apply $\vec{T}\times$ to both sides of equ (2)

$$z'\,\vec{T}\times\vec{P}' = z\,\vec{T}\times\mathrm{R}\,\vec{P} \qquad (3)$$

Apply $\vec{P}'\bullet$ to both sides of equ (3)

$$\vec{P}'\bullet\vec{T}\times\mathrm{R}\,\vec{\mathrm{P}}=0 \qquad (4)$$

Let we define

$$\mathrm{E}=\vec{\mathrm{T}}\times\mathrm{R}=[\,\vec{\mathrm{T}}\times\mathrm{R}^{(1)}\mid\vec{\mathrm{T}}\times\mathrm{R}^{(2)}\mid\vec{\mathrm{T}}\times\mathrm{R}^{(3)}\,] \qquad (5)$$

The eq(4) can be rewritten as

$$\vec{P}'\bullet\ \mathrm{E}\,\vec{\mathrm{P}}=0 \qquad (6)$$

Note: eq.(6) is a homogeneous scalar equation.

$\mathrm{E}=\vec{\mathrm{T}}\times\mathrm{R}$ is a 3×3 matrix containing only motion parameters,

8 or more PCs can uniquely determine E, subject to:

$$\|E\|^2=2$$

After matrix E is found, translation $\vec{T}$ can be solved:

Or $$E \bullet \vec{T} = \vec{T} \times R \bullet \vec{T} = 0$$

$$E^T \vec{T} = 0 \qquad (7)$$

$\vec{T}$ can be determined from eq(7) subject to

$$\|T\|^2 = 1$$

Once $\vec{T}$ is obtained, rotation ***R*** can be obtained by least-square method:

$$\vec{T} \times R - E = 0 \qquad (8)$$

Or let $E = [\ E^{(1)} \mid E^{(2)} \mid E^{(3)}\ ]$

$$R = [\ R^{(1)} \mid R^{(2)} \mid R^{(3)}\ ]$$

$$R = [E^{(1)} \times \vec{T} + E^{(2)} \times E^{(3)} \mid E^{(2)} \times \vec{T} + E^{(3)} \times E^{(1)} \mid E^{(3)} \times \vec{T} + E^{(1)} \times E^{(2)}]$$

Note, 180° reflection of motion is still a solution of equ (7) (homogeneous equation). In this case, object is moving behind the camera. To check for a real solution, we apply $\vec{P}'\times$ to both sides of equ (2).

$$z\,\vec{P}' \times R\,\vec{P} + \vec{P}' \times \vec{T} = 0$$

Therefore if z > 0, it must hold that

$$(\vec{P}' \times \mathrm{R}\ \vec{\mathrm{P}}) \bullet (\vec{\mathrm{T}} \times \vec{\mathrm{P}}') > 0$$

Thus if,

$$\sum_i (\vec{P}' \times \mathrm{R}\ \vec{\mathrm{P}}) \bullet (\vec{\mathrm{T}} \times \vec{\mathrm{P}}') < 0$$

let

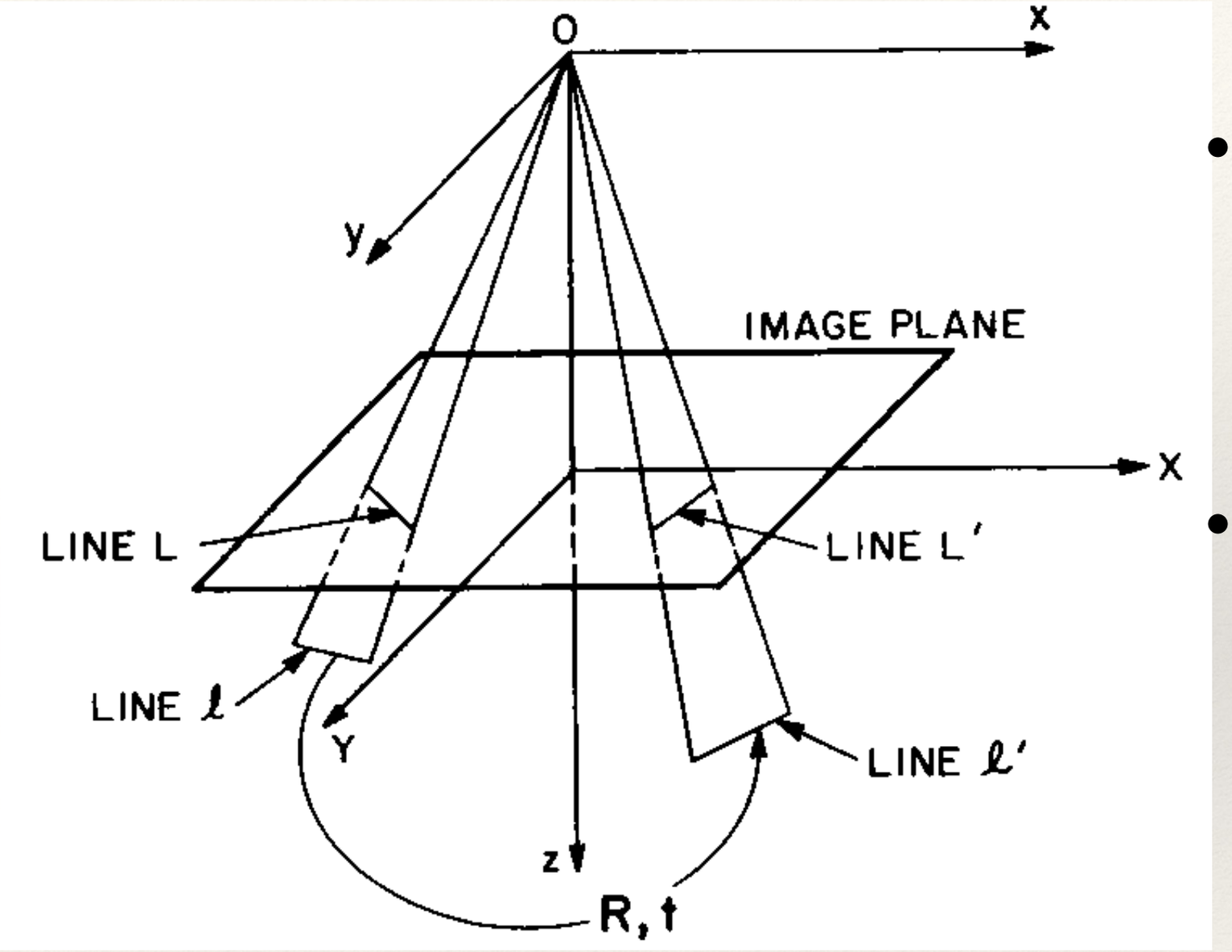
$$\vec{T} \leftarrow -\,\vec{T}$$

Remark:

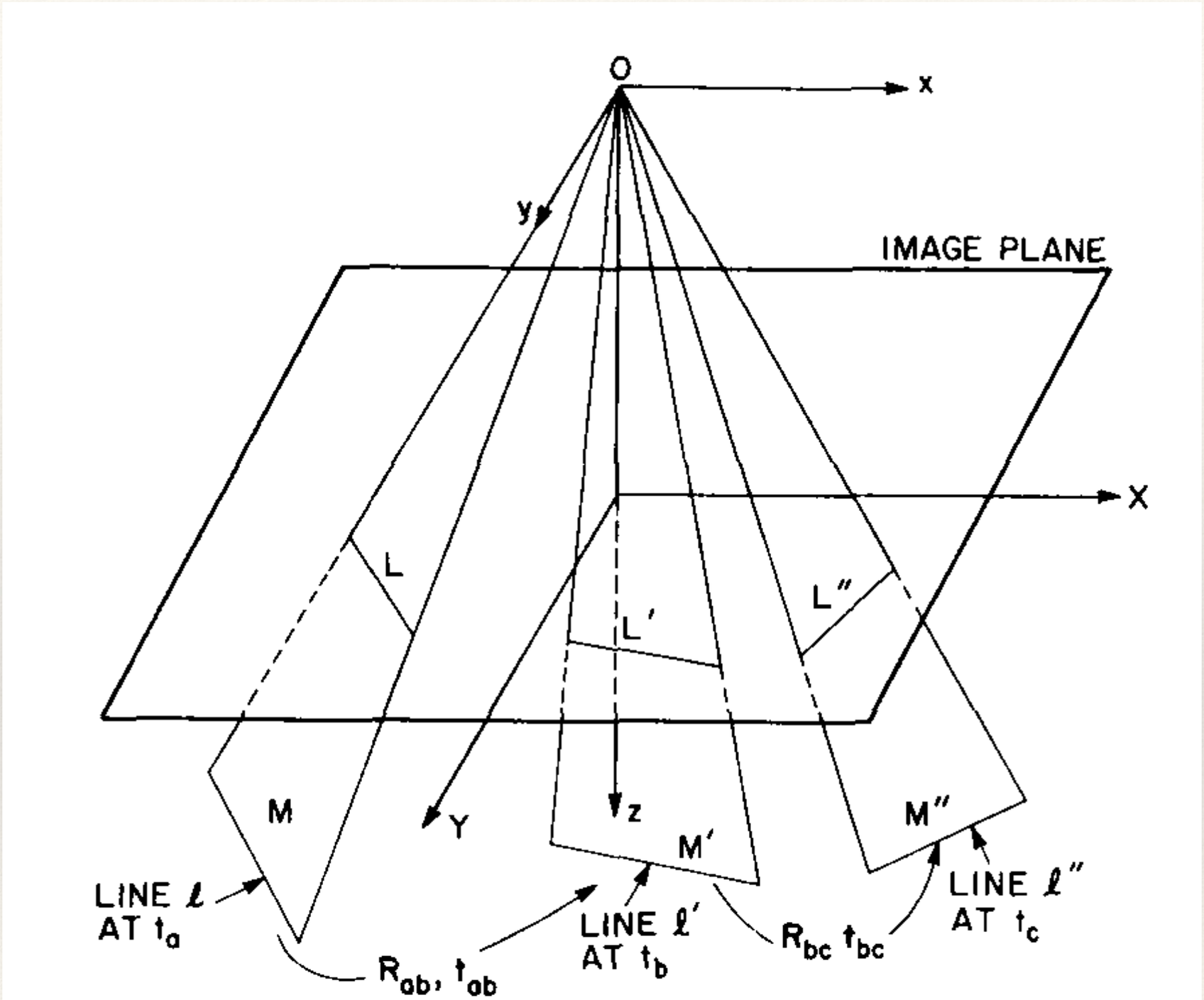
- LINEAR ALGORITHM: at least 8 non-degenerated PC's over two image frames are needed to solve for a 3D motion using a linear method!

- NONLINEAR ALGORITHM: at least 5 PC's over two image frames are needed to solve for a 3D motion, with the 3D points not lie on a quadratic surface.

Line features



- The number of solutions is infinite no matter how large N is.
- One way to make solutions unique (or finite in number) is to take three views.


O
x
y
IMAGE PLANE
X
L
L'
L''
M
Y
Z
M'
M''
LINE ℓ
AT t_a
R_{ab}, t_{ab}
LINE ℓ'
AT t_b
R_{bc} t_{bc}
LINE ℓ''
AT t_c

$$
\begin{aligned}
L &: AX + BY + C &&= 0 \\
L' &: A'X + B'Y + C' &&= 0 \\
L'' &: A''X + B''Y + C'' &&= 0
\end{aligned}
$$

$$\boldsymbol{N}^T (= (A, B, C))$$

$$\boldsymbol{N'}^T (= (A', B', C'))$$

$$\boldsymbol{N''}^T (= (A'', B'', C''))$$

$$(\boldsymbol{R}_{ab}\boldsymbol{N}) \cdot \boldsymbol{N'} \times (\boldsymbol{R}_{bc}^{-1}\boldsymbol{N''}) = 0.$$

6 LCs over 3 views for six rotation unknowns

$$\boldsymbol{t}_{ab} \cdot (\boldsymbol{R}_{ab}\boldsymbol{N}) = \frac{||N' \times \boldsymbol{R}_{ab}\boldsymbol{N}||}{||N' \times \boldsymbol{R}_{bc}^{-1}\boldsymbol{N''}||} \boldsymbol{R}_{bc}^{-1}\boldsymbol{t}_{bc} \cdot (\boldsymbol{R}_{bc}^{-1}\boldsymbol{N''})$$

Summary

Linear solutions

	Point	Line
3D-to-3D	3PCs (Noncollinear)	2LCs(Nonparallel)
2D-to-3D	6PCs (Coplaner)	8LCs
2D-to-2D	8PCs (2 views)	13LCs(3 views)