Robot Sensing:
Multiple View Geometry

METR 4202: **Robotics** & Automation

Dr Surya Singh -- Lecture # 9

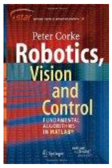
September 21, 2016

metr4202@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~metr4202/ [http://**metr4202.com**]

Schedule of Events

Week	Date	Lecture (W: 12:05-1:50, 50-N202)
1	27-Jul	Introduction
2	3-Aug	Representing Position & Orientation & State (Frames, Transformation Matrices & Affine Transformations)
3	10-Aug	Robot Kinematics Review (& *Ekka Day*)
4	17-Aug	Robot Inverse Kinematics & Kinetics
5	24-Aug	Robot Dynamics (Jacobeans)
6	31-Aug	Robot Sensing: Perception & Linear Observers
7	7-Sep	Robot Sensing: Single View Geometry & Lines
8	14-Sep	Robot Sensing: Feature Detection
9	**21-Sep**	**Robot Sensing: Multiple View Geometry**
	28-Sep	*Study break*
10	5-Oct	Motion Planning
11	12-Oct	Probabilistic Robotics: Localization & SLAM
12	19-Oct	Probabilistic Robotics: Planning & Control
13	26-Oct	State-Space Automation (Shaping the Dynamic Response/LQR) + Course Review

Follow Along Reading:



Robotics, Vision & Control
by Peter Corke

Also online:SpringerLink

UQ Library eBook:
364220144X

Today

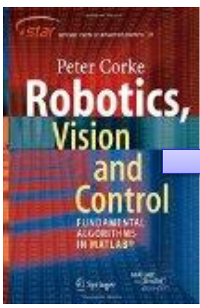
➔ Sensing and Vision ←

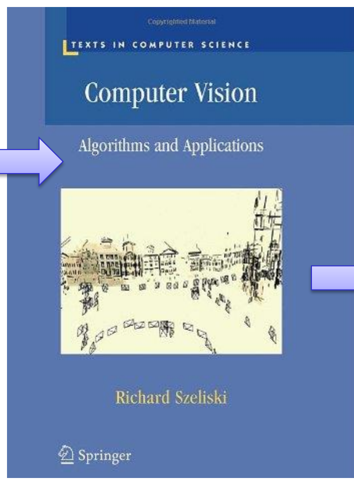
- Multiple View Geometry
  - P. 47
  - Hartley & Zisserman:
    Chapter 6: Camera Models
    Chapter 7: Camera Matrix

- Planning
  - pp. 91-103
    (Yup! That's all Peter Corke has to say on that – which explains why there is no planning at ACRV ☺).

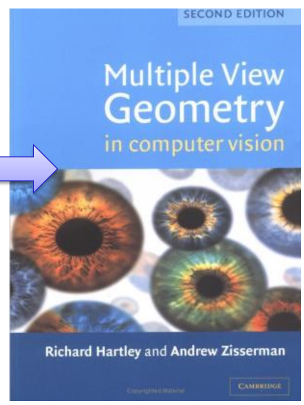
Next Time

Reference Material





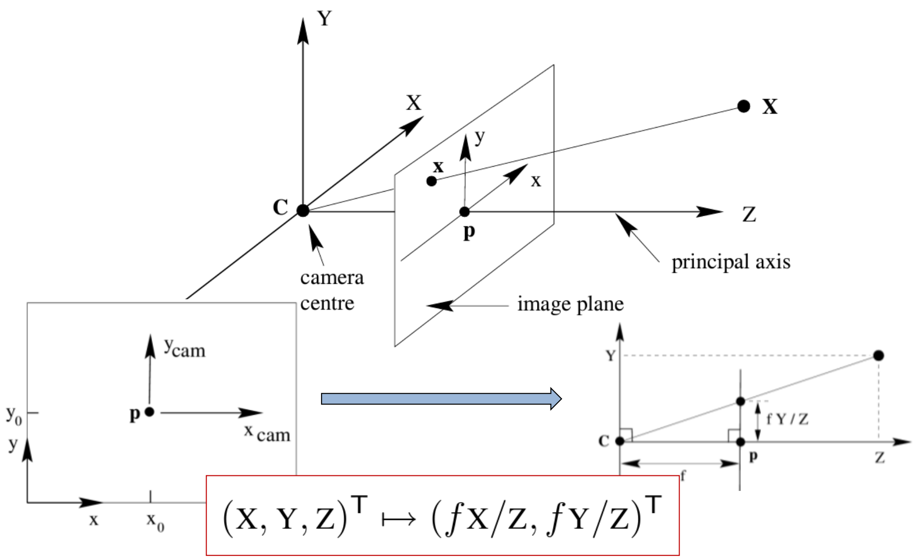
**UQ Library/
SpringerLink**



UQ Library
(ePDF)

Multiple View Geometry

Image Formation – Single View Geometry [I]



$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T$$

Hartly & Zisserman, Ch. 6

Image Formation – Single View Geometry [II]

➔ Camera Projection Matrix

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

world camera Intrinsics

- x = Image point
- $\mathbf{X}$ = World point
- K = Camera Calibration Matrix

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

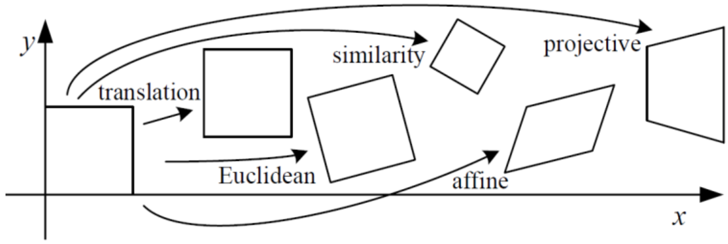
$$\mathbf{x} = K[I \mid \mathbf{0}]\mathbf{X}_{\text{cam}}.$$

➔Perspective Camera as:

$$P = K[R \mid \mathbf{t}]$$

where: P is **3**×4 and of **rank 3**

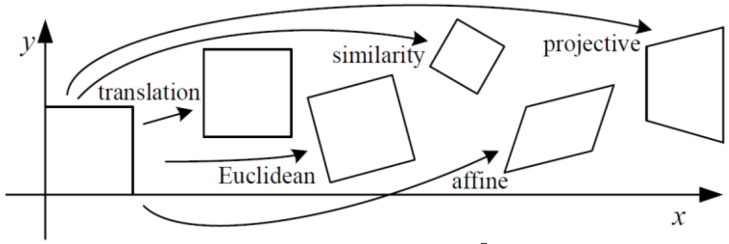
Transformations ☆



- ***x'*: New Image & $\underline{x}$: Old Image**
- Euclidean: (Distances preserved)
  $$x' = \begin{bmatrix} R & t \end{bmatrix} \underline{x}$$
- Similarity (Scaled Rotation): (Angles preserved)
  $$x' = \begin{bmatrix} sR & t \end{bmatrix} \underline{x}$$

Fig. 2.4 from Szeliski, *Computer Vision: Algorithms and Applications*

Transformations [2]



- Affine :
  (|| lines remain ||)
- Projective:
  (straight lines preserved)
  H: Homogenous 3x3 Matrix

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \underline{x}$$

$$x' = \mathbf{H}\underline{x}$$

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}$$

$$y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$$

Fig. 2.4 from Szeliski, Computer Vision: Algorithms and Applications

2-D Transformations

➔ x' = point in the **new** (or 2nd) image

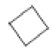
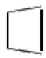
➔ x = point in the old image

- Translation x' = x + t
- Rotation x' = R x + t
- Similarity x' = sR x + t
- Affine x' = A x
- Projective x' = A x

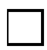
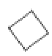
here, x is an inhomogeneous pt (2-vector)

x' is a homogeneous point

2-D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times 3}$	2	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times 3}$	3	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times 3}$	4	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times 3}$	6	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times 3}$	8	straight lines	

3D Transformations

Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{3\times 4}$	3	orientation $+\cdots$	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{3\times 4}$	6	lengths $+\cdots$	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{3\times 4}$	7	angles $+\cdots$	
affine	$\begin{bmatrix} A \end{bmatrix}_{3\times 4}$	12	parallelism $+\cdots$	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{4\times 4}$	15	straight lines	

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Projection Models

- Orthographic
- Weak Perspective
- Affine
- Perspective
- Projective

$$\mathbf{\Pi} = \begin{bmatrix} i_x & i_y & i_z & t_x \\ j_x & j_y & j_z & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{\Pi} = f\begin{bmatrix} i_x & i_y & i_z & t_x \\ j_x & j_y & j_z & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{\Pi} = [\mathbf{R} \quad \mathbf{t}]$$

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$

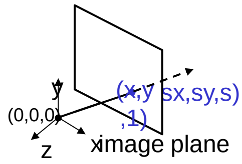
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Properties of Projection

- Preserves
  - Lines and conics
  - Incidence
  - Invariants (cross-ratio)
- Does not preserve
  - Lengths
  - Angles
  - Parallelism

Planar Projective Transformations

- Perspective projection of a plane
  - lots of names for this:
    - homography, colineation, planar projective map
  - Easily modeled using homogeneous coordinates



$$\begin{bmatrix} sx' \\ sy' \\ s \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

p' **H** **p**

To apply a homography **H**

- compute **p'** = **Hp**
- **p''** = **p'**/s normalize by dividing by third component

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Image Formation – Two-View Geometry [Stereopsis] ➔ Fundamental Matrix

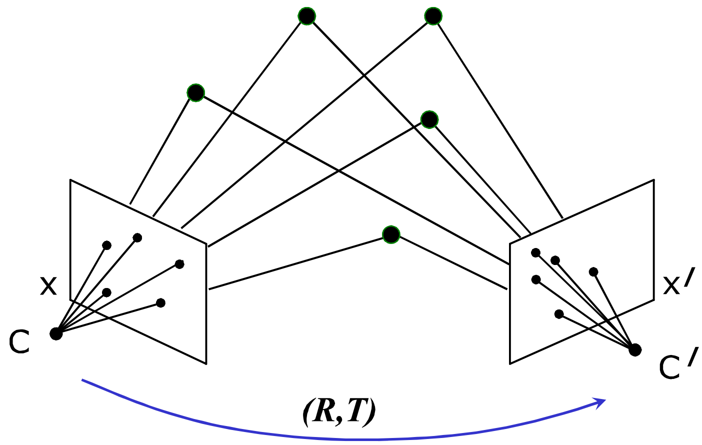
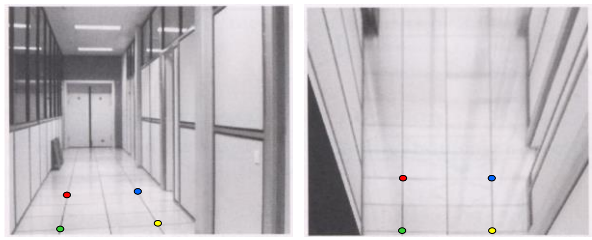


Image Rectification



To unwarp (rectify) an image

- solve for **H** given **p''** and **p**
- solve equations of the form: s**p''** = **Hp**
  - linear in unknowns: s and coefficients of **H**
  - need at least 4 points

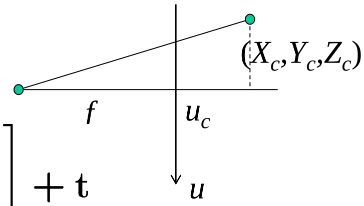
Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D Projective Geometry

- These concepts generalize naturally to 3D
  - Homogeneous coordinates
    - Projective 3D points have four coords: P = (X,Y,Z,W)
  - Duality
    - A plane L is also represented by a 4-vector
    - Points and planes are dual in 3D: L P=0
  - Projective transformations
    - Represented by 4x4 matrices T: P' = TP, L' = L T-1
  - Lines are a special case…

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D → 2D Perspective Projection (Image Formation Equations)



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathbf{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

3D → 2D Perspective Projection

- Matrix Projection (camera matrix):

$$\mathbf{p} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi P}$$

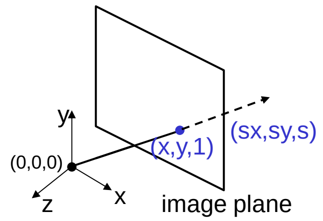
It's useful to decompose Π into **T** → **R** → project → **A**

$$\mathbf{\Pi} = \underbrace{\begin{bmatrix} s_x & 0 & -t_x \\ 0 & s_y & -t_y \\ 0 & 0 & 1/f \end{bmatrix}}_{\text{intrinsics}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{orientation}} \underbrace{\begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix}}_{\text{position}}$$

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

The Projective Plane

- Why do we need homogeneous coordinates?
  - Represent points at infinity, homographies, perspective projection, multi-view relationships
- What is the geometric intuition?
  - A point in the image is a ray in projective space

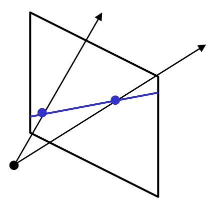


- Each *point* (x,y) on the plane is represented by a *ray* (sx,sy,s)
  - all points on the ray are equivalent: $(x, y, 1) \cong (sx, sy, s)$

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Projective Lines

- What is a line in projective space?



- A line is a *plane* of rays through origin
  - all rays (x,y,z) satisfying: ax + by + cz = 0

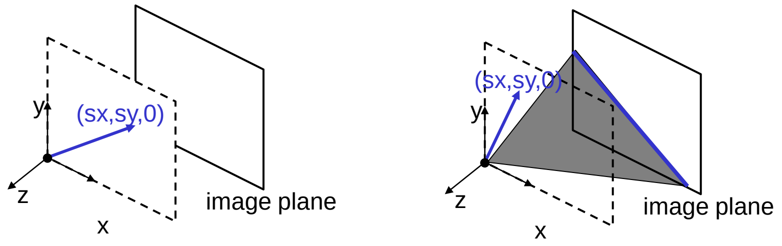
$$\text{in vector notation:} \quad 0 = \underbrace{\begin{bmatrix} a & b & c \end{bmatrix}}_{\mathbf{l}^T} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\mathbf{p}}$$

- A line is represented as a homogeneous 3-vector **l**

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Ideal points and lines

- Ideal point ("point at infinity")
  - $p \cong (x, y, 0)$ – parallel to image plane
  - It has infinite image coordinates

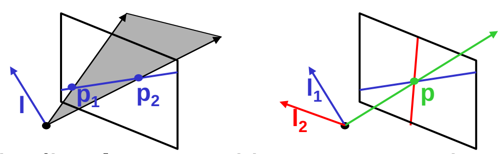


Line at infinity

- $l_\infty \cong (0, 0, 1)$ – parallel to image plane
- Contains all ideal points

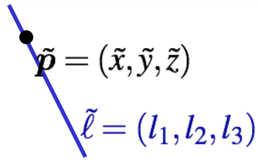
Point and Line Duality

- A line l is a homogeneous 3-vector (a ray)
- It is $\perp$ to every point (ray) p on the line: $l^T p=0$



- What is the line **l** spanned by rays $\mathbf{p_1}$ and $\mathbf{p_2}$?
  - **l** is $\perp$ to $\mathbf{p_1}$ and $\mathbf{p_2}$ $\Rightarrow$ $\mathbf{l} = \mathbf{p_1} \times \mathbf{p_2}$ (**l** is the plane normal)
- What is the intersection of two lines $\mathbf{l_1}$ and $\mathbf{l_2}$?
  - **p** is $\perp$ to $\mathbf{l_1}$ and $\mathbf{l_2}$ $\Rightarrow$ $\mathbf{p} = \mathbf{l_1} \times \mathbf{l_2}$
- Points and lines are *dual* in projective space
  - every property of points also applies to lines

Point and Line Duality [II]



Homogeneous ⇔ Cartesian

- Point:

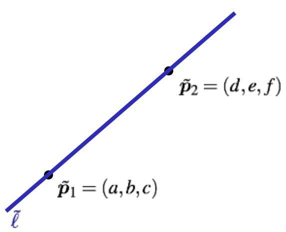
  $\tilde{\boldsymbol{P}} = (\tilde{x}, \tilde{y}, \tilde{z})$ | $\boldsymbol{P} = (x, y)$ $x = \frac{\tilde{x}}{\tilde{z}}, y = \frac{\tilde{y}}{\tilde{z}}$

- Line:
  - Is such that $\tilde{l}^T \tilde{p} = 0$
  - Point Eq of a line is: $y = mx + b$

Image/Notation from: Corke, Ch. 11

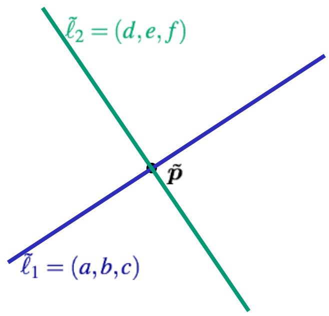
Point and Line Duality [III]

- 2 Points Make a Line



$$\tilde{\ell} = \tilde{\boldsymbol{p}}_1 \times \tilde{\boldsymbol{p}}_2$$

- 2 Lines Make Point!

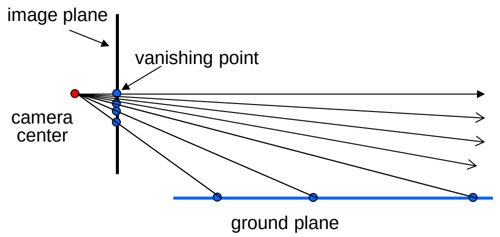


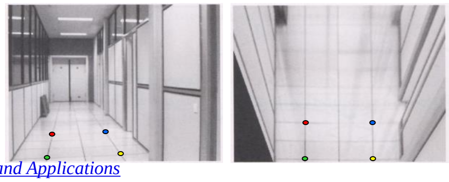
$$\tilde{\boldsymbol{p}} = \tilde{\ell}_1 \times \tilde{\ell}_2$$

Image/Notation from: Corke, Ch. 11

Vanishing Points

- Vanishing point
  - projection of a point at infinity
  - whiteboard capture, architecture,…

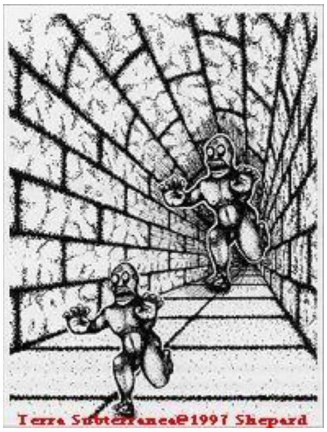


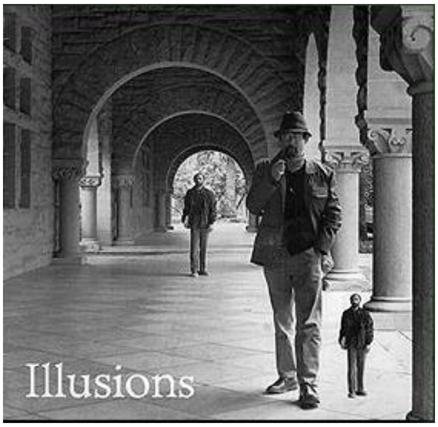


Slide from Szeliski, *Computer Vision: Algorithms and Applications*

"Fundamental" Multi-View Geometry

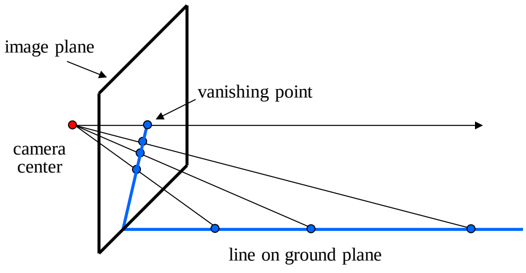
Fun With Vanishing Points



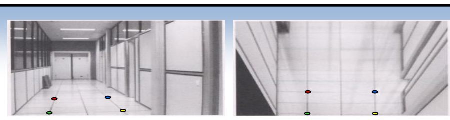


Slide from Szeliski, *Computer Vision: Algorithms and Applications*

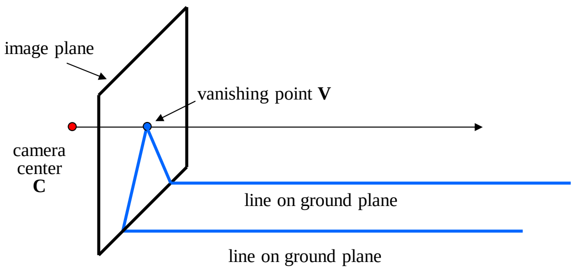
Vanishing Points (2D)



Vanishing Points

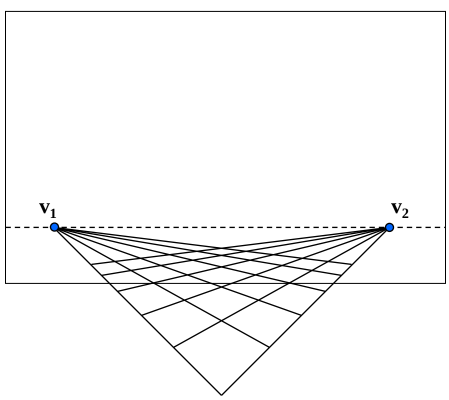


- Properties
  - Any two parallel lines have the same vanishing point
  - The ray from C through v point is parallel to the lines
  - An image may have more than one vanishing point



Vanishing Lines

- Multiple Vanishing Points
  - Any set of parallel lines on the plane define a vanishing point
  - The union of all of these vanishing points is the horizon line

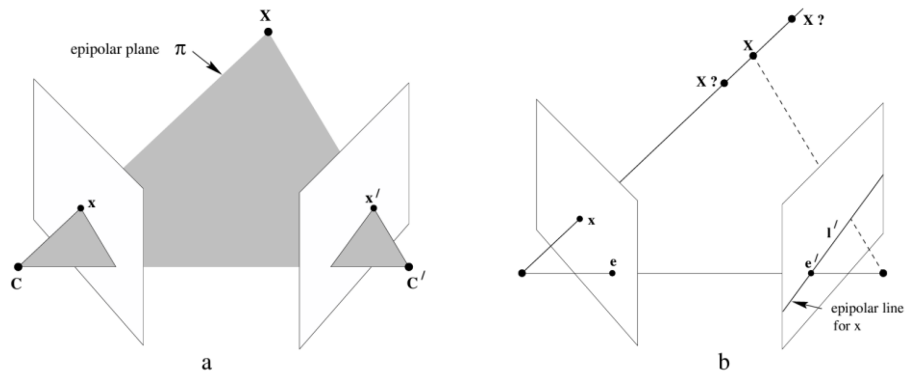


Stereo: epipolar geometry

- for two images (or images with collinear camera centers), can find epipolar lines
- epipolar lines are the projection of the pencil of planes passing through the centers
- Rectification: warping the input images (perspective transformation) so that epipolar lines are horizontal

Slide from [Szeliski](), *[Computer Vision: Algorithms and Applications]()*

Two-View Geometry: Epipolar Plane



- **Epipole:** The *point* of intersection of the line joining the camera centres (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera centre of the other view.
- **Epipolar plane** is a plane containing the baseline. There is a one-parameter family (a pencil) of epipolar planes
- **Epipolar line** is the intersection of an epipolar plane with the image plane. All epipolar lines intersect at the epipole. An epipolar plane intersects the left and right image planes in epipolar lines, and defines the correspondence between the lines.

Two-frame methods

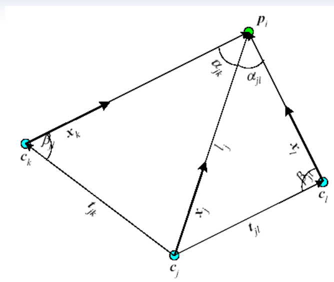
- Two main variants:
- Calibrated: “Essential matrix” E
  use ray directions (xi, xi’)
- Uncalibrated: “Fundamental matrix” F

- [Hartley & Zisserman 2000]

From Szeliski, *Computer Vision: Algorithms and Applications*

Essential matrix

- Co-planarity constraint:
  - $x' \approx R\,x + t$
  - $[t]_\times\, x' \approx [t]_\times\, R\,x$
  - $x'\,[t]_\times\, x' \approx x'\,[t]_\times\, R\,x$
  - $x'\,E\,x = 0$ with $E = [t]_\times R$

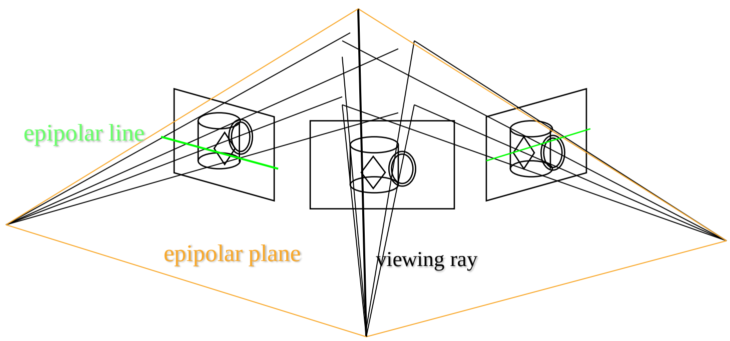


- Solve for **E** using least squares (SVD)
- t is the least singular vector of E
- R obtained from the other two s.v.s

From Szeliski, *Computer Vision: Algorithms and Applications*

Stereo: Epipolar geometry

- Match features along epipolar lines



Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Fundamental Matrix

- The fundamental matrix is the algebraic representation of epipolar geometry.

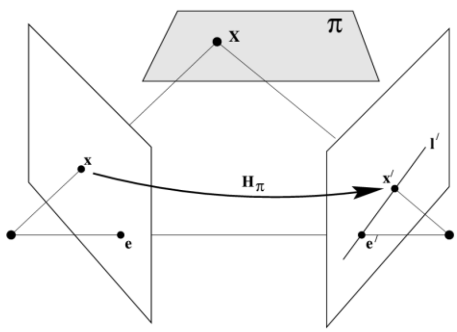


Fig. 9.5. *A point* $\mathbf{x}$ *in one image is transferred via the plane* π *to a matching point* $\mathbf{x}'$ *in the second image. The epipolar line through* $\mathbf{x}'$ *is obtained by joining* $\mathbf{x}'$ *to the epipole* $\mathbf{e}'$. *In symbols one may write* $\mathbf{x}' = \mathtt{H}_\pi \mathbf{x}$ *and* $\mathbf{l}' = [\mathbf{e}']_\times \mathbf{x}' = [\mathbf{e}']_\times \mathtt{H}_\pi \mathbf{x} = \mathtt{F}\mathbf{x}$ *where* $\mathtt{F} = [\mathbf{e}']_\times \mathtt{H}_\pi$ *is the fundamental matrix.*

Fundamental matrix

- Camera calibrations are unknown
-     x' F x = 0 with F = [e]× H = K'[t]× R K-1
- Solve for F using least squares (SVD)
  - re-scale (xi, xi') so that |xi|≈1/2 [Hartley]
- e (epipole) is still the least singular vector of F
- H obtained from the other two s.v.s
- "plane + parallax" (projective) reconstruction
- use self-calibration to determine K [Pollefeys]

From Szeliski, *Computer Vision: Algorithms and Applications*

Fundamental Matrix Example

- Suppose the camera matrices are those of a calibrated stereo rig with the world origin at the first camera

$$\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}] \qquad \mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}].$$

- Then:

$$\mathtt{P}^{+} = \begin{bmatrix} \mathtt{K}^{-1} \\ \mathbf{0}^{\mathsf{T}} \end{bmatrix} \qquad \mathbf{C} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$$

- Epipoles are at:

$$\mathbf{e} = \mathtt{P}\begin{pmatrix} -\mathtt{R}^{\mathsf{T}}\mathbf{t} \\ 1 \end{pmatrix} = \mathtt{K}\mathtt{R}^{\mathsf{T}}\mathbf{t} \qquad \mathbf{e}' = \mathtt{P}'\begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = \mathtt{K}'\mathbf{t}.$$

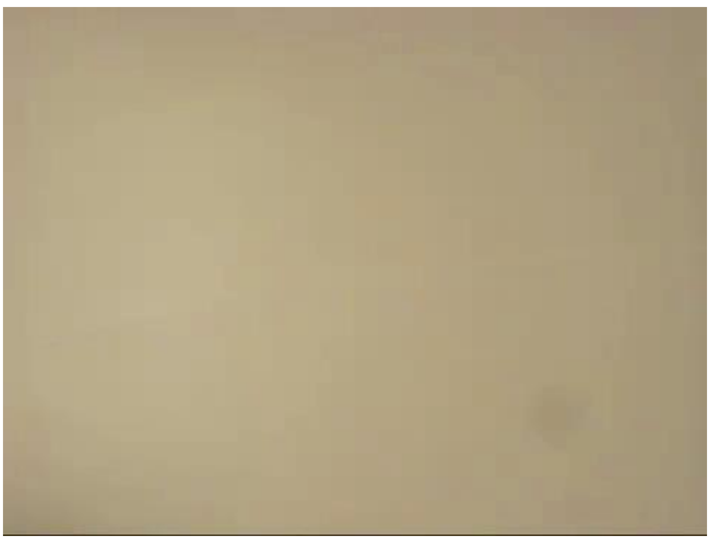
∴

$$\mathtt{F} = [\mathbf{e}']_{\times}\mathtt{K}'\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}[\mathbf{t}]_{\times}\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}[\mathtt{R}^{\mathsf{T}}\mathbf{t}]_{\times}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}\mathtt{K}^{\mathsf{T}}[\mathbf{e}]_{\times}$$

Summary of fundamental matrix properties

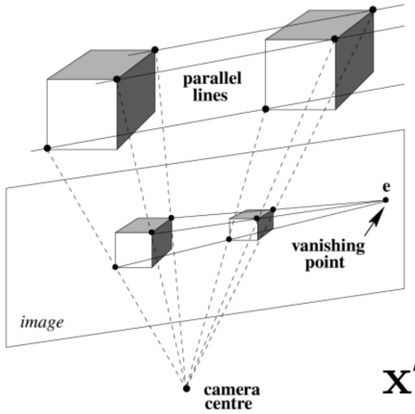
- F is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence**: If $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image points, then $\mathbf{x}'^{\mathsf{T}}\mathtt{F}\mathbf{x} = 0$.
- **Epipolar lines**:
  - $\mathbf{l}' = \mathtt{F}\mathbf{x}$ is the epipolar line corresponding to $\mathbf{x}$.
  - $\mathbf{l} = \mathtt{F}^{\mathsf{T}}\mathbf{x}'$ is the epipolar line corresponding to $\mathbf{x}'$.
- **Epipoles**:
  - $\mathtt{F}\mathbf{e} = \mathbf{0}$.
  - $\mathtt{F}^{\mathsf{T}}\mathbf{e}' = \mathbf{0}$.
- **Computation from camera matrices** $\mathtt{P}, \mathtt{P}'$:
  - General cameras,
    $\mathtt{F} = [\mathbf{e}']_\times \mathtt{P}'\mathtt{P}^+$, where $\mathtt{P}^+$ is the pseudo-inverse of $\mathtt{P}$, and $\mathbf{e}' = \mathtt{P}'\mathbf{C}$, with $\mathtt{P}\mathbf{C} = \mathbf{0}$.
  - Canonical cameras, $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = [\mathtt{M} \mid \mathbf{m}]$,
    $\mathtt{F} = [\mathbf{e}']_\times \mathtt{M} = \mathtt{M}^{-\mathsf{T}}[\mathbf{e}]_\times$, where $\mathbf{e}' = \mathbf{m}$ and $\mathbf{e} = \mathtt{M}^{-1}\mathbf{m}$.
  - Cameras not at infinity $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$,
    $\mathtt{F} = \mathtt{K}'^{-\mathsf{T}}[\mathbf{t}]_\times \mathtt{R}\mathtt{K}^{-1} = [\mathtt{K}'\mathbf{t}]_\times \mathtt{K}'\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}\mathtt{K}^{\mathsf{T}}[\mathtt{K}\mathtt{R}^{\mathsf{T}}\mathbf{t}]_\times$.

Cool Robotics Share



D. Wedge, *The Fundamental Matrix Song*

Fundamental Matrix & Motion

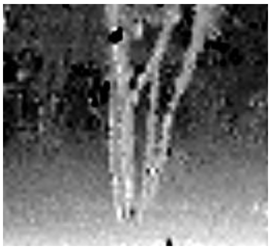


$$\mathbf{x}'^{\mathsf{T}}\mathbf{F}\mathbf{x} = 0.$$

- Under a pure translational camera motion, 3D points appear to slide along parallel rails. The images of these parallel lines intersect in a vanishing point corresponding to the translation direction. The epipole **e** is the vanishing point.

Finding correspondences

- Apply feature matching criterion (e.g., correlation or Lucas-Kanade) at all pixels simultaneously
- Search only over epipolar lines (many fewer candidate positions)



Slide from Szeliski, *Computer Vision: Algorithms and Applications*

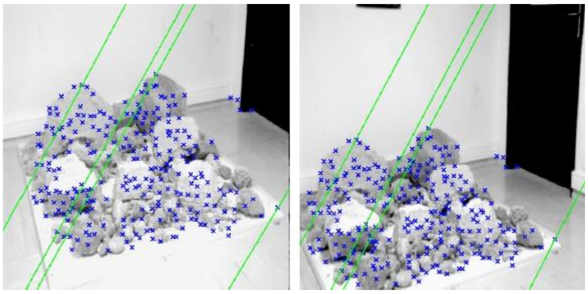
Matching criteria

- Raw pixel values (correlation)
- Band-pass filtered images [Jones & Malik 92]
- "Corner" like features [Zhang, …]
- Edges [many people…]
- Gradients [Seitz 89; Scharstein 94]
- Rank statistics [Zabih & Woodfill 94]

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

Feature-based stereo

- Match "corner" (interest) points



- Interpolate complete solution

Slide from Szeliski, *Computer Vision: Algorithms and Applications*

SFM: Structure from Motion
(& Cool Robotics Share (this week))

Structure [from] Motion

- Given a set of feature tracks,
  estimate the 3D structure and 3D (camera) motion.
- Assumption: orthographic projection
- Tracks: (u_{fp}, v_{fp}), f: frame, p: point
- Subtract out **mean** 2D position…

  $\mathbf{i}_f$: rotation, $\mathbf{s}_p$: position

$$u_{fp} = i_f^T s_p, v_{fp} = j_f^T s_p$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Structure from motion

- How many points do we need to match?
- 2 frames:
  - (R,t): 5 dof + 3n point locations ≤
  - 4n point measurements ⇒
  - n ≥ 5

$$\hat{u}_{ij} = f(\boxed{\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i})$$
$$\hat{v}_{ij} = g(\boxed{\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i})$$

- k frames:
  - 6(k–1)-1 + 3n ≤ 2kn
- always want to use many more

From Szeliski, *Computer Vision: Algorithms and Applications*

Measurement equations

- Measurement equations

$$u_{fp} = \boldsymbol{i}_f^T \boldsymbol{s}_p \qquad \boldsymbol{i}_f\text{: rotation, } \boldsymbol{s}_p\text{: position}$$
$$v_{fp} = \boldsymbol{j}_f^T \boldsymbol{s}_p$$

- Stack them up…

$$\boldsymbol{W} = \boldsymbol{R}\, \boldsymbol{S}$$
$$\boldsymbol{R} = (\boldsymbol{i}_1,\ldots,\boldsymbol{i}_F, \boldsymbol{j}_1,\ldots,\boldsymbol{j}_F)^T$$
$$\boldsymbol{S} = (\boldsymbol{s}_1,\ldots,\boldsymbol{s}_P)$$

From Szeliski, *Computer Vision: Algorithms and Applications*

Factorization

$$W = R_{2F\times3}\, S_{3\times P}$$

SVD

$$W = U\Lambda V \quad \Lambda \textbf{ must be rank 3}$$

$$W' = (U\Lambda^{1/2})(\Lambda^{1/2} V) \quad = U' V'$$

Make R orthogonal

$$R = QU', \;\; S = Q^{-1}V'$$

$$i_f^T Q^T Q i_f = 1 \ldots$$

From Szeliski, [Computer Vision: Algorithms and Applications]

Results

- Look at paper figures…

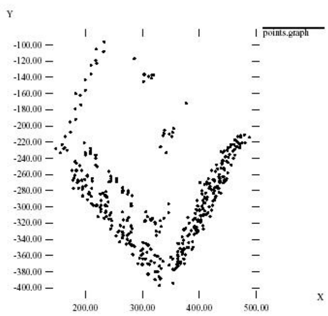


Figure 4.5: A view of the computed shape from approximately above the building (compare with figure 4.6).

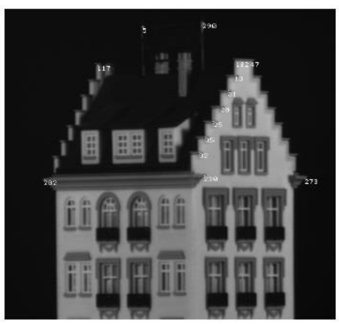


Figure 4.7: For a quantitative evaluation, distances between the features shown in the picture were measured on the actual model, and compared with the computed results. The comparison is shown in figure 4.8.

From Szeliski, [Computer Vision: Algorithms and Applications]

Bundle Adjustment

- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

$$
\begin{aligned}
\hat{u}_{ij} &= f(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i}) \\
\hat{v}_{ij} &= g(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i})
\end{aligned}
$$

From Szeliski, *Computer Vision: Algorithms and Applications*

More Cool Robotics Share!