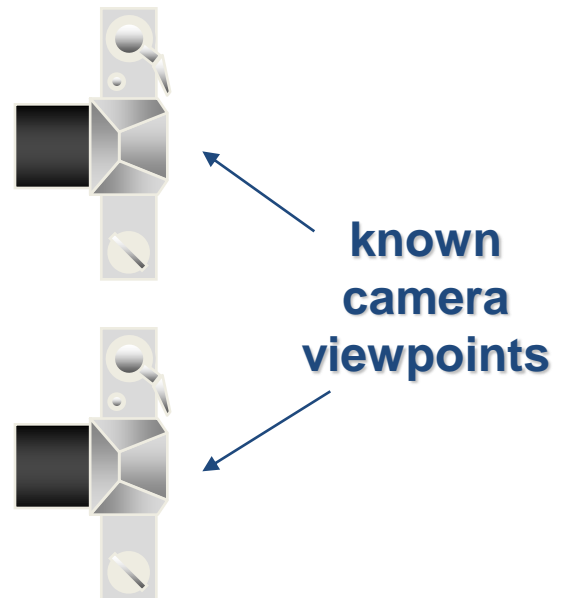


---

# Stereo Reconstruction

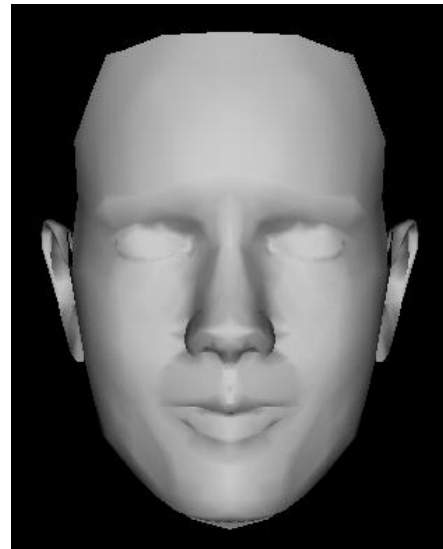
---

Shape (3D) from two (or more) images



# Example

images



shape



surface  
reflectance

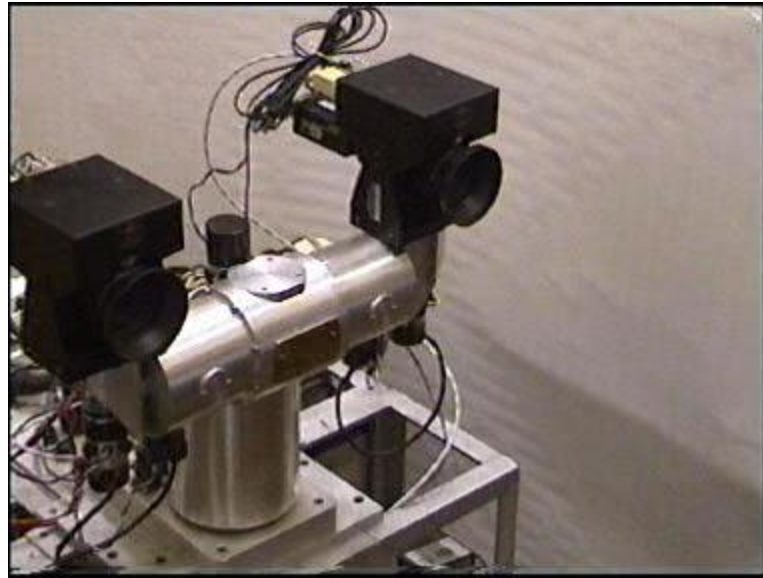
# Scenarios

The two images can arise from

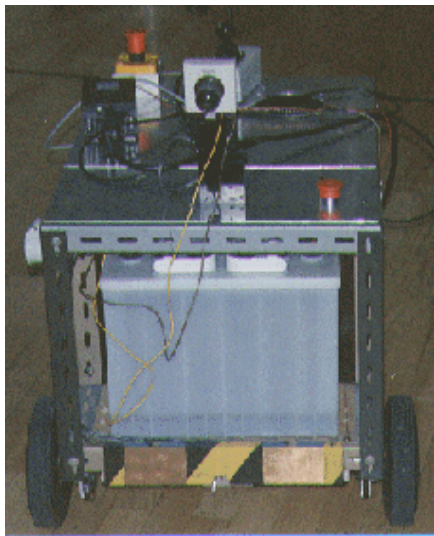
- A stereo rig consisting of two cameras
  - the two images are acquired **simultaneously**
- or
- A single moving camera (static scene)
  - the two images are acquired **sequentially**

The two scenarios are geometrically equivalent

Stereo head



Camera on a mobile vehicle



# The objective

Given two images of a scene acquired by known cameras compute the 3D position of the scene (structure recovery)



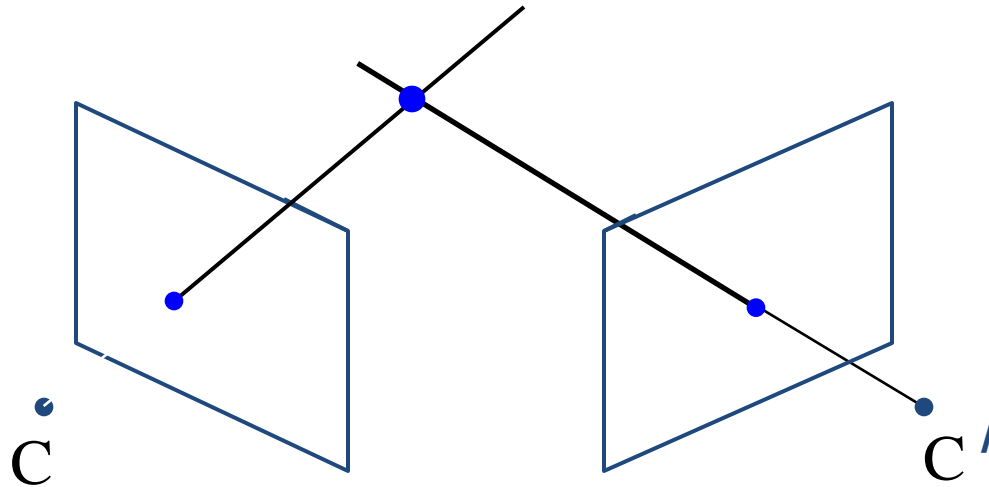
Basic principle: triangulate from corresponding image points

- Determine 3D point at intersection of two back-projected rays

Corresponding points are images of the same scene point



Triangulation



The back-projected points generate rays which intersect at the 3D scene point

# An algorithm for stereo reconstruction

1. For each point in the first image determine the corresponding point in the second image

(this is a search problem)

2. For each pair of matched points determine the 3D point by triangulation

(this is an estimation problem)



# The correspondence problem

Given a point  $x$  in one image find the corresponding point in the other image



This appears to be a 2D search problem, but it is reduced to a 1D search by the **epipolar constraint**



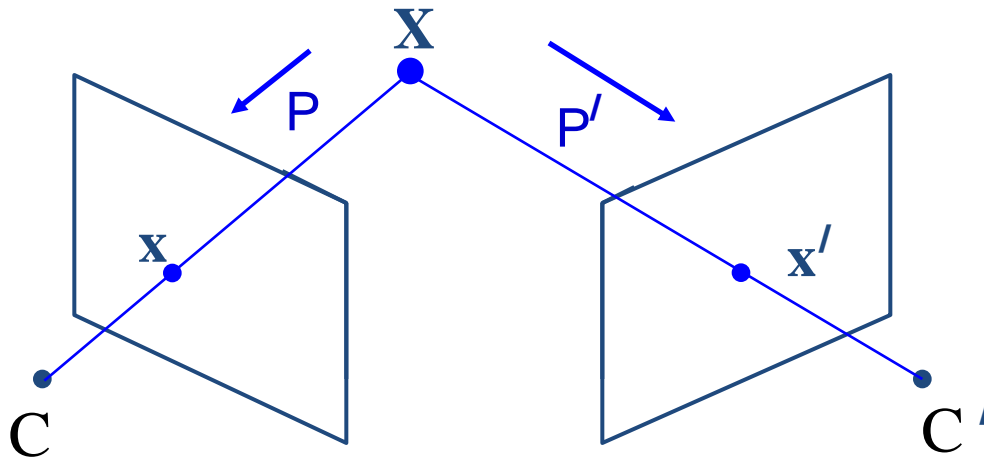
# Outline

1. Epipolar geometry
  - the geometry of two cameras
  - reduces the correspondence problem to a line search
2. Stereo correspondence algorithms
3. Triangulation

# Notation

The two cameras are  $P$  and  $P'$ , and a 3D point  $\mathbf{X}$  is imaged as

$$\mathbf{x} = P\mathbf{X} \quad \mathbf{x}' = P'\mathbf{X}$$



$P$  :  $3 \times 4$  matrix

$\mathbf{X}$  : 4-vector

$\mathbf{x}$  : 3-vector

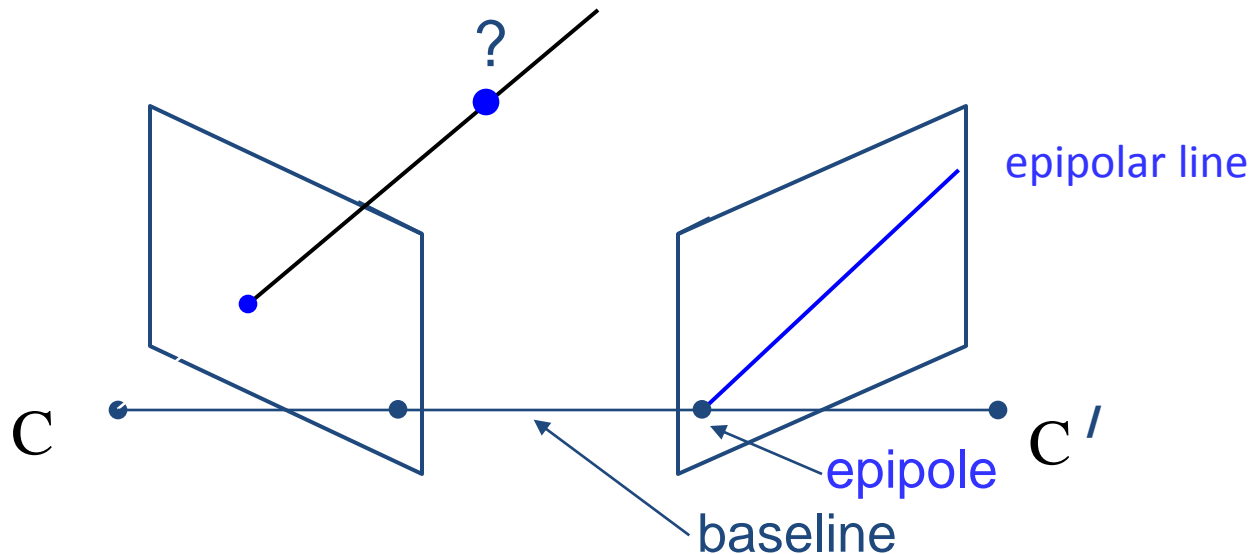
## Warning

for equations involving homogeneous quantities '=' means 'equal up to scale'

# Epipolar geometry

# Epipolar geometry

Given an image point in one view, where is the corresponding point in the other view?



- A point in one view “generates” an **epipolar line** in the other view
- The corresponding point lies on this line

# Epipolar line

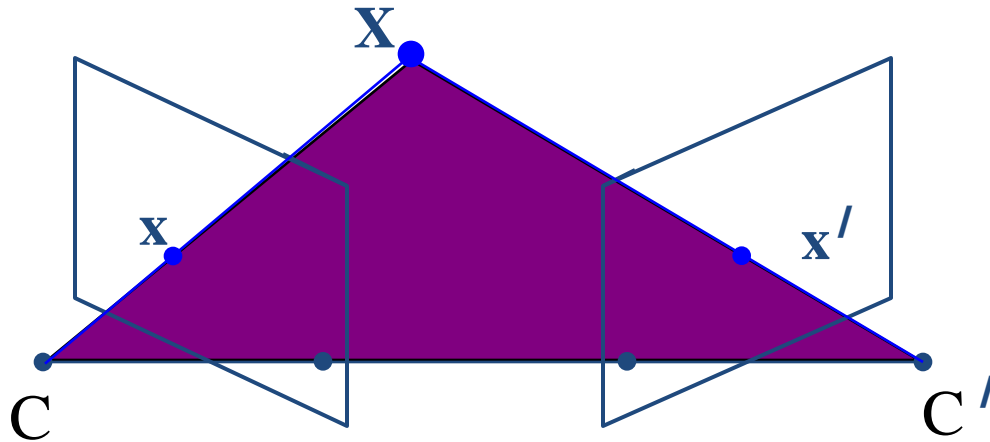


## Epipolar constraint

- Reduces correspondence problem to 1D search along an epipolar line

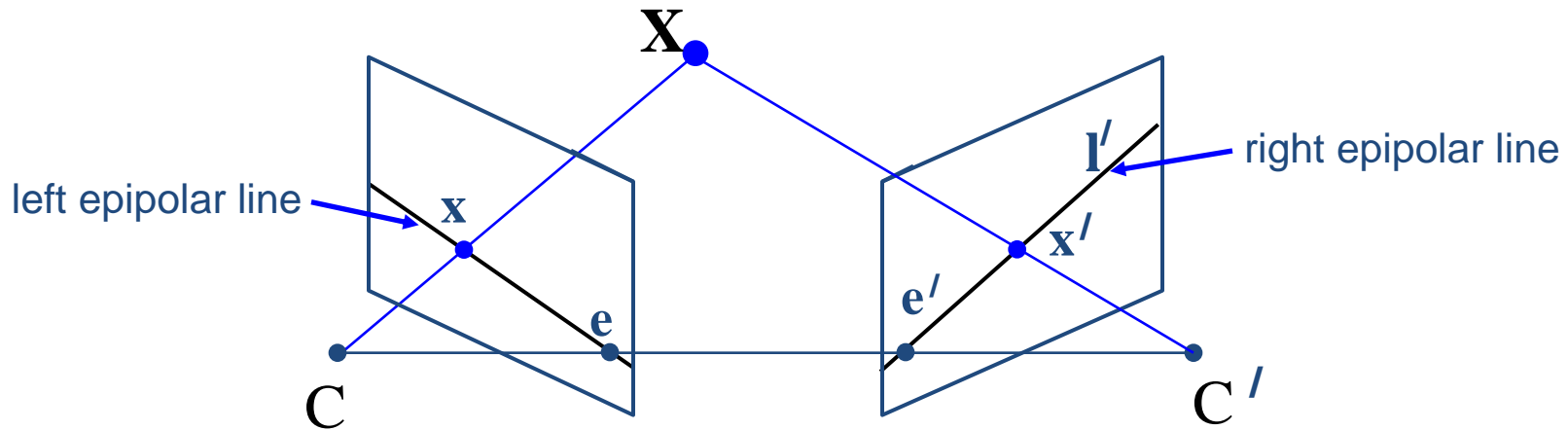
# Epipolar geometry continued

Epipolar geometry is a consequence of the coplanarity of the camera centres and scene point



The camera centres, corresponding points and scene point lie in a single plane, known as the **epipolar plane**

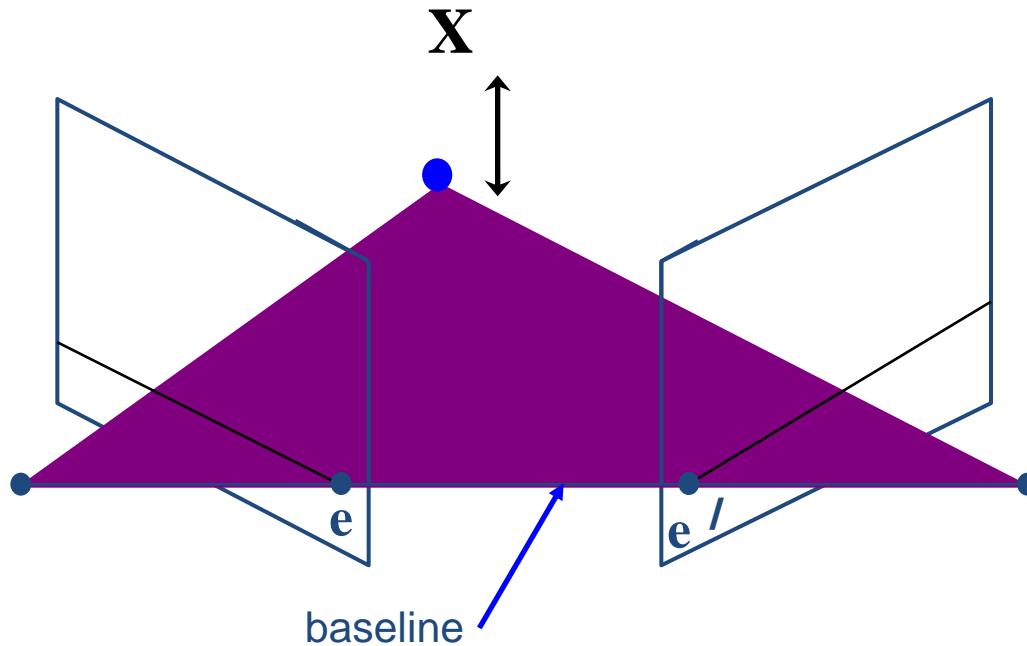
# Nomenclature



- The **epipolar line**  $l'$  is the image of the ray through  $x$
- The **epipole**  $e$  is the point of intersection of the line joining the camera centres with the image plane
  - this line is the **baseline** for a stereo rig, and
  - the translation vector for a moving camera
- The epipole is the image of the centre of the other camera:  $e = PC'$ ,  $e' = P'C$



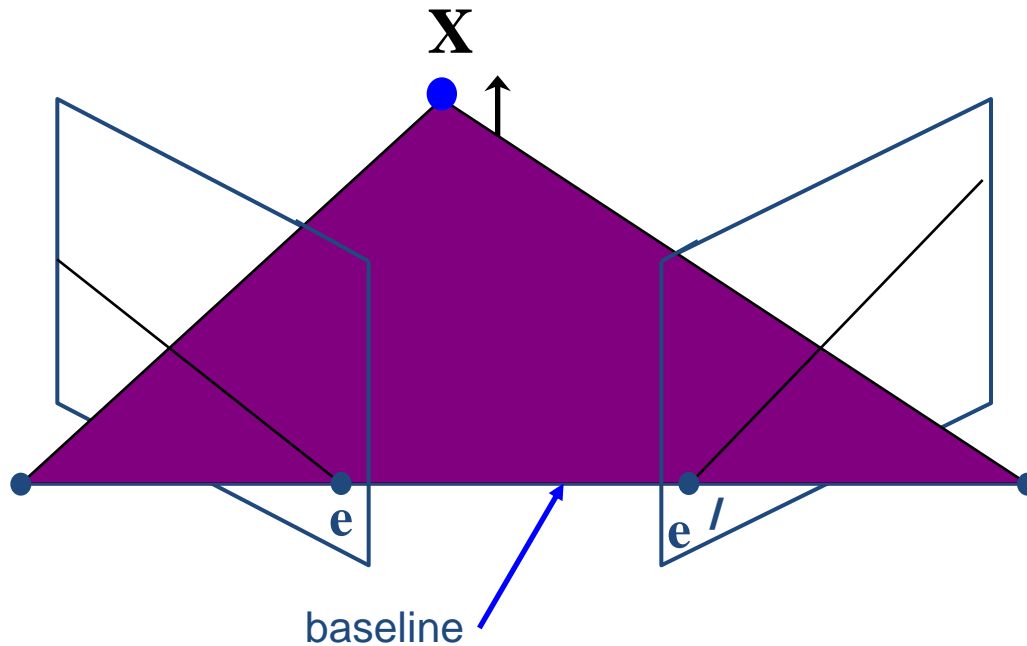
# The epipolar pencil



As the position of the 3D point  $\mathbf{X}$  varies, the epipolar planes “rotate” about the baseline. This family of planes is known as an **epipolar pencil**. All epipolar lines intersect at the epipole.

(a pencil is a one parameter family)

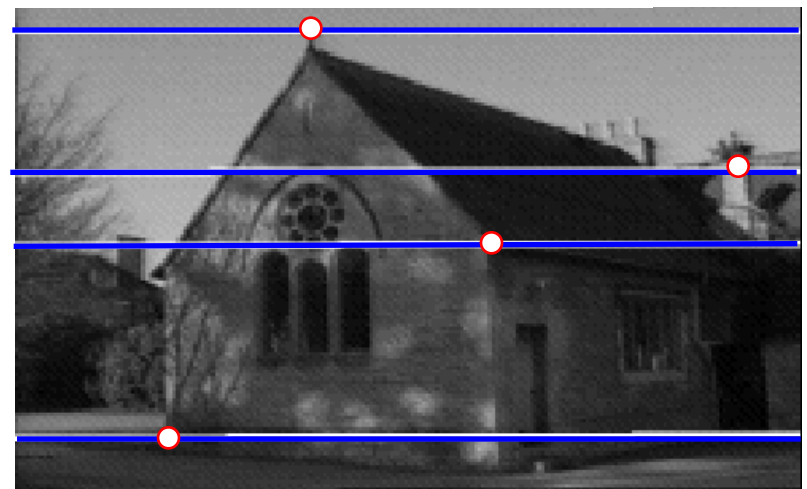
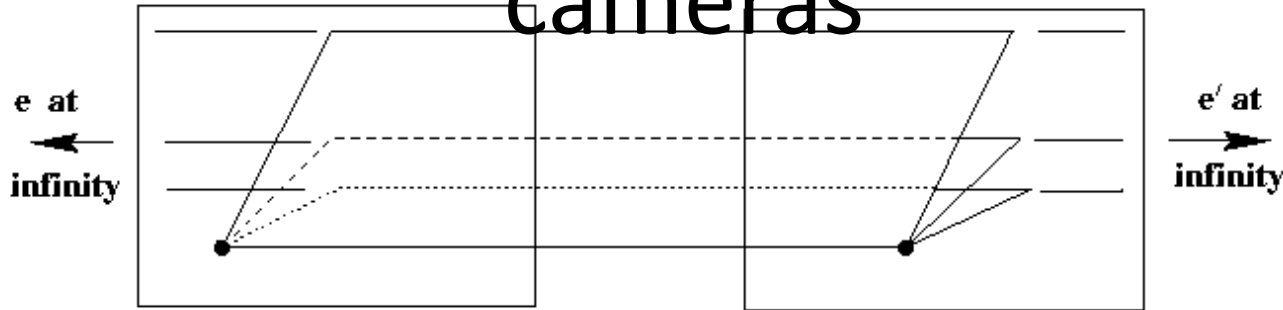
# The epipolar pencil



As the position of the 3D point  $\mathbf{X}$  varies, the epipolar planes “rotate” about the baseline. This family of planes is known as an **epipolar pencil**. All epipolar lines intersect at the epipole.

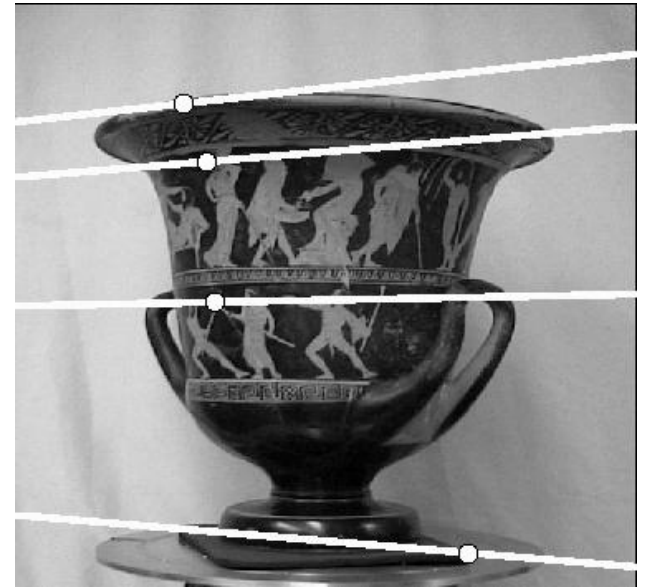
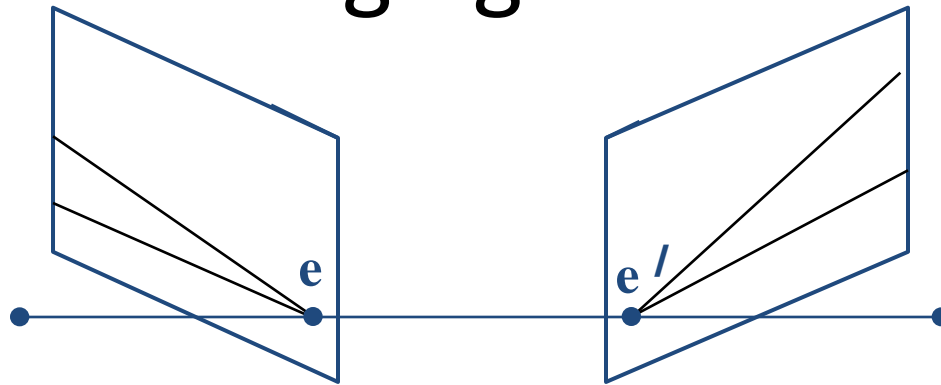
(a pencil is a one parameter family)

# Epipolar geometry example I: parallel cameras



Epipolar geometry depends **only** on the relative pose (position and orientation) and internal parameters of the two cameras, i.e. the position of the camera centres and image planes. It does **not** depend on the scene structure (3D points external to the camera).

# Epipolar geometry example II: converging cameras



Note, epipolar lines are in general **not** parallel

# Homogeneous notation for lines

Recall that a point  $(x, y)$  in 2D is represented by the homogeneous 3-vector  $\mathbf{x} = (x_1, x_2, x_3)^\top$ , where  $x = x_1/x_3, y = x_2/x_3$

A **line** in 2D is represented by the homogeneous 3-vector

$$\mathbf{l} = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix}$$

which is the line  $l_1x + l_2y + l_3 = 0$ .

Example represent the line  $y = 1$  as a homogeneous vector.

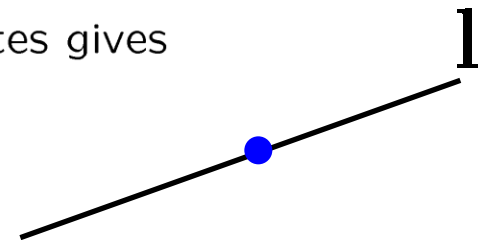
Write the line as  $-y + 1 = 0$  then  $l_1 = 0, l_2 = -1, l_3 = 1$ , and  $\mathbf{l} = (0, -1, 1)^\top$ .

Note that  $\mu(l_1x + l_2y + l_3) = 0$  represents the same line (only the ratio of the homogeneous line coordinates is significant).

Writing both the point and line in homogeneous coordinates gives

$$l_1x_1 + l_2x_2 + l_3x_3 = 0$$

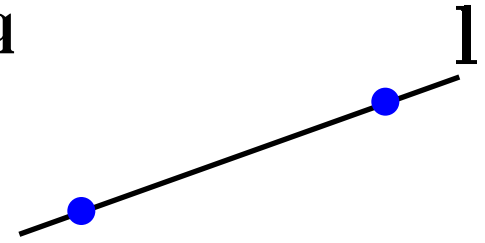
• **point on line**  $\mathbf{l} \cdot \mathbf{x} = 0$  or  $\mathbf{l}^\top \mathbf{x} = 0$  or  $\mathbf{x}^\top \mathbf{l} = 0$



- The line  $\mathbf{l}$  through the two points  $\mathbf{p}$  and  $\mathbf{q}$  is  $\mathbf{l} = \mathbf{p} \times \mathbf{q}$

Proof

$$\mathbf{l} \cdot \mathbf{p} = (\mathbf{p} \times \mathbf{q}) \cdot \mathbf{p} = 0 \quad \mathbf{l} \cdot \mathbf{q} = (\mathbf{p} \times \mathbf{q}) \cdot \mathbf{q} = 0$$



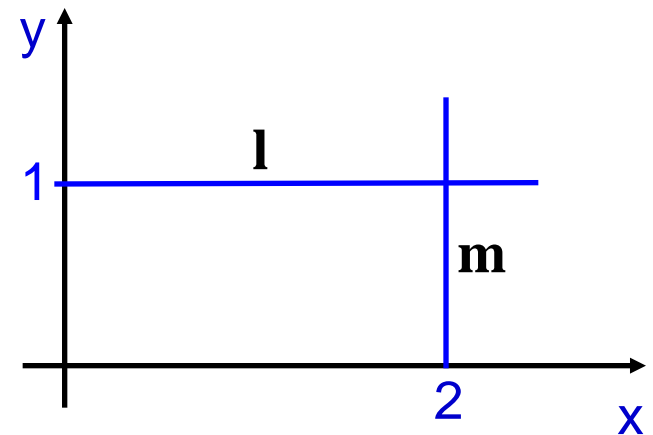
- The intersection of two lines  $\mathbf{l}$  and  $\mathbf{m}$  is the point  $\mathbf{x} = \mathbf{l} \times \mathbf{m}$

Example: compute the point of intersection of the two lines  $\mathbf{l}$  and  $\mathbf{m}$  in the figure below

$$\mathbf{l} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} \quad \mathbf{m} = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix}$$

$$\mathbf{x} = \mathbf{l} \times \mathbf{m} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & -1 & 1 \\ -1 & 0 & 2 \end{vmatrix} = \begin{pmatrix} -2 \\ -1 \\ -1 \end{pmatrix}$$

which is the point (2,1)



# Matrix representation of the vector cross product

The vector product  $\mathbf{v} \times \mathbf{x}$  can be represented as a matrix multiplication

$$\mathbf{v} \times \mathbf{x} = \begin{pmatrix} v_2 x_3 - v_3 x_2 \\ v_3 x_1 - v_1 x_3 \\ v_1 x_2 - v_2 x_1 \end{pmatrix} = [\mathbf{v}]_{\times} \mathbf{x}$$

where

$$[\mathbf{v}]_{\times} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

- $[\mathbf{v}]_{\times}$  is a  $3 \times 3$  skew-symmetric matrix of rank 2.
- $\mathbf{v}$  is the null-vector of  $[\mathbf{v}]_{\times}$ , since  $\mathbf{v} \times \mathbf{v} = [\mathbf{v}]_{\times} \mathbf{v} = \mathbf{0}$ .



Example: compute the cross product of **l** and **m**

$$\mathbf{l} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} \quad \mathbf{m} = \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} \quad [\mathbf{v}]_{\times} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

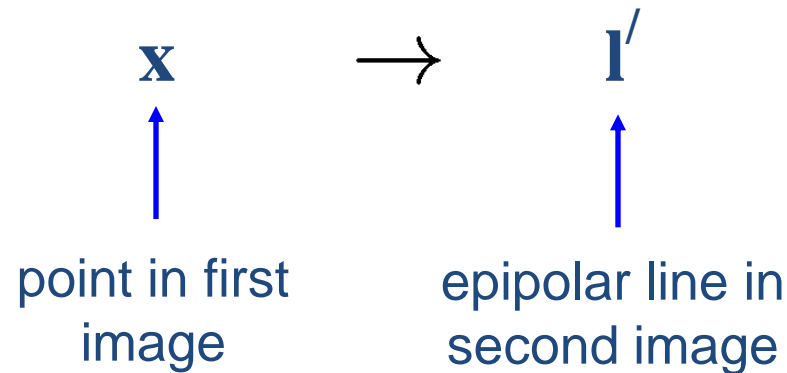
$$\mathbf{x} = \mathbf{l} \times \mathbf{m} = [\mathbf{l}]_{\times} \mathbf{m} = \begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{pmatrix} -1 \\ 0 \\ 2 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \\ -1 \end{pmatrix}$$

Note

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

# Algebraic representation of epipolar geometry

We know that the epipolar geometry defines a mapping



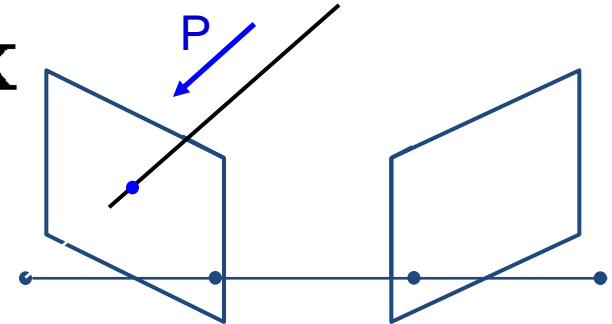
- the map only depends on the cameras  $P, P'$  (not on structure)
- it will be shown that the map is **linear** and can be written as  $\mathbf{l}' = F\mathbf{x}$ , where  $F$  is a  $3 \times 3$  matrix called the **fundamental matrix**

# Derivation of the algebraic expression

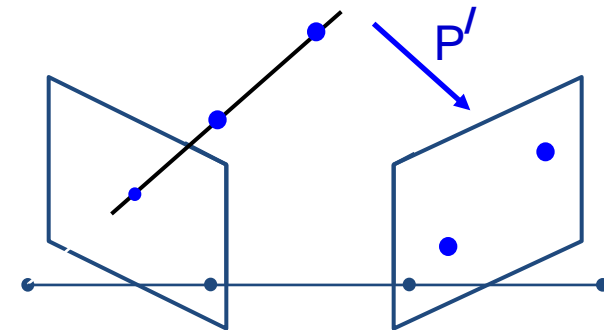
## Outline

$$\mathbf{l}' = \mathbf{F}\mathbf{x}$$

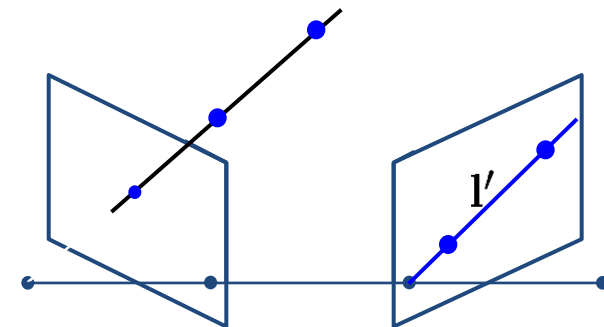
**Step 1:** for a point  $\mathbf{x}$  in the first image  
back project a ray with camera  $\mathbf{P}$



**Step 2:** choose two points on the ray and  
project into the second image with camera  $\mathbf{P}'$



**Step 3:** compute the line through the two  
image points using the relation  $\mathbf{l}' = \mathbf{p} \times \mathbf{q}$

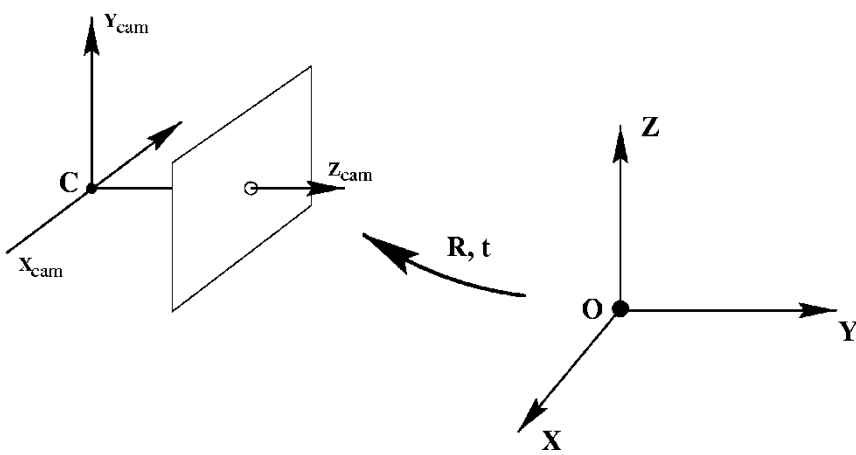


- choose camera matrices

$$P = K [R | t]$$

internal calibration      rotation      translation

from world to camera coordinate frame

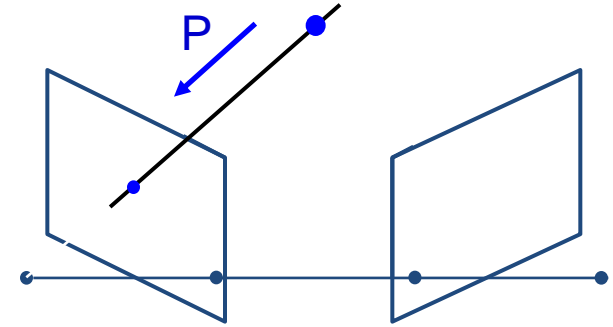


- first camera  $P = K [I | 0]$

world coordinate frame aligned with first camera

- second camera  $P' = K' [R | t]$

Step 1: for a point  $\mathbf{x}$  in the first image  
back project a ray with camera  $\mathbf{P} = \mathbf{K} [\mathbf{I} \mid \mathbf{0}]$



A point  $\mathbf{x}$  back projects to a ray

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = z\mathbf{K}^{-1} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = z\mathbf{K}^{-1}\mathbf{x}$$

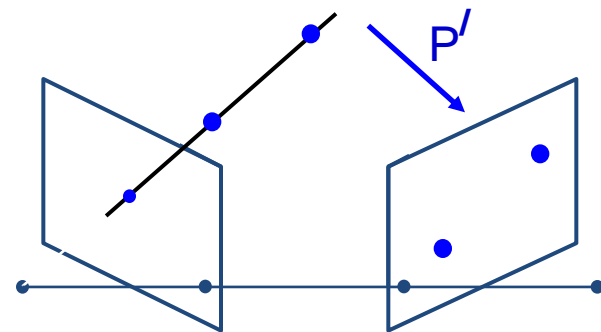
where  $\mathbf{Z}$  is the point's depth, since

$$\mathbf{X}(z) = \begin{pmatrix} z\mathbf{K}^{-1}\mathbf{x} \\ 1 \end{pmatrix}$$

satisfies

$$\mathbf{P}\mathbf{X}(z) = \mathbf{K}[\mathbf{I} \mid \mathbf{0}]\mathbf{X}(z) = \mathbf{x}$$

Step 2: choose two points on the ray and project into the second image with camera  $P'$



Consider two points on the ray  $\mathbf{X}(z) = \begin{pmatrix} zK^{-1}\mathbf{x} \\ 1 \end{pmatrix}$

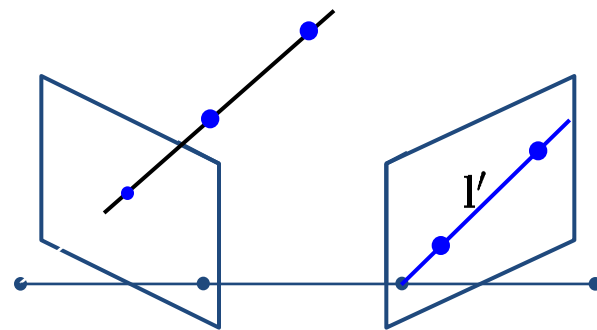
- $\mathbf{Z} = 0$  is the camera centre  $\begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$
- $\mathbf{Z} = \infty$  is the point at infinity  $\begin{pmatrix} K^{-1}\mathbf{x} \\ 0 \end{pmatrix}$

Project these two points into the second view

$$P' \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = K'[\mathbf{R} \mid \mathbf{t}] \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} = K'\mathbf{t}$$

$$P' \begin{pmatrix} K^{-1}\mathbf{x} \\ 0 \end{pmatrix} = K'[\mathbf{R} \mid \mathbf{t}] \begin{pmatrix} K^{-1}\mathbf{x} \\ 0 \end{pmatrix} = K'\mathbf{R}K^{-1}\mathbf{x}$$

Step 3: compute the line through the two image points using the relation  $\mathbf{l}' = \mathbf{p} \times \mathbf{q}$



Compute the line through the points  $\mathbf{l}' = (\mathbf{K}'\mathbf{t}) \times (\mathbf{K}'\mathbf{R}\mathbf{K}^{-1}\mathbf{x})$

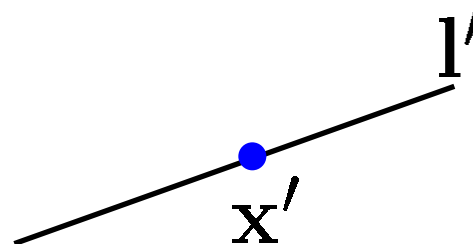
Using the identity  $(\mathbf{M}\mathbf{a}) \times (\mathbf{M}\mathbf{b}) = \mathbf{M}^{-\top}(\mathbf{a} \times \mathbf{b})$  where  $\mathbf{M}^{-\top} = (\mathbf{M}^{-1})^{\top} = (\mathbf{M}^{\top})^{-1}$

$$\mathbf{l}' = \mathbf{K}'^{-\top} \left( \mathbf{t} \times (\mathbf{R}\mathbf{K}^{-1}\mathbf{x}) \right) = \underbrace{\mathbf{K}'^{-\top}[\mathbf{t}]_{\times}\mathbf{R}\mathbf{K}^{-1}}_{\mathbf{F}}\mathbf{x} \quad \text{F is the fundamental matrix}$$

$$\mathbf{l}' = \mathbf{F}\mathbf{x} \quad \mathbf{F} = \mathbf{K}'^{-\top}[\mathbf{t}]_{\times}\mathbf{R}\mathbf{K}^{-1}$$

Points  $\mathbf{x}$  and  $\mathbf{x}'$  correspond ( $\mathbf{x} \leftrightarrow \mathbf{x}'$ ) then  $\mathbf{x}'^{\top}\mathbf{l}' = 0$

$$\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$$





Example I: compute the fundamental matrix for a parallel camera stereo rig

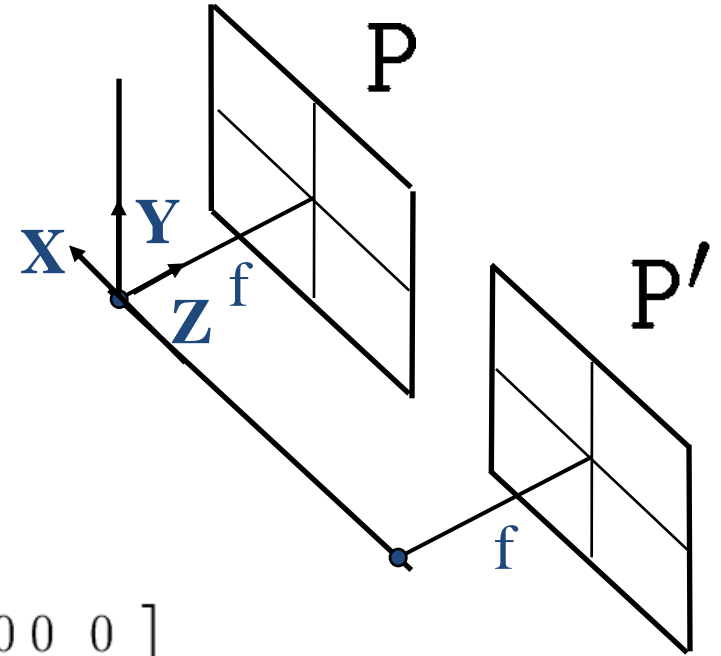
$$\mathbf{P} = \mathbf{K}[\mathbf{I} \mid \mathbf{0}] \quad \mathbf{P}' = \mathbf{K}'[\mathbf{R} \mid \mathbf{t}]$$

$$\mathbf{K} = \mathbf{K}' = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R} = \mathbf{I} \quad \mathbf{t} = \begin{pmatrix} t_x \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{F} = \mathbf{K}'^{-\top} [\mathbf{t}]_{\times} \mathbf{R} \mathbf{K}^{-1}$$

$$= \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -t_x \\ 0 & t_x & 0 \end{bmatrix} \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\mathbf{x}'^{\top} \mathbf{F} \mathbf{x} = (x' \ y' \ 1) \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$



- reduces to  $y = y'$ , i.e. raster correspondence (horizontal scan-lines)

F is a rank 2 matrix

The epipole  $e$  is the null-space vector (kernel) of  $F$  (exercise), i.e.  $Fe = 0$

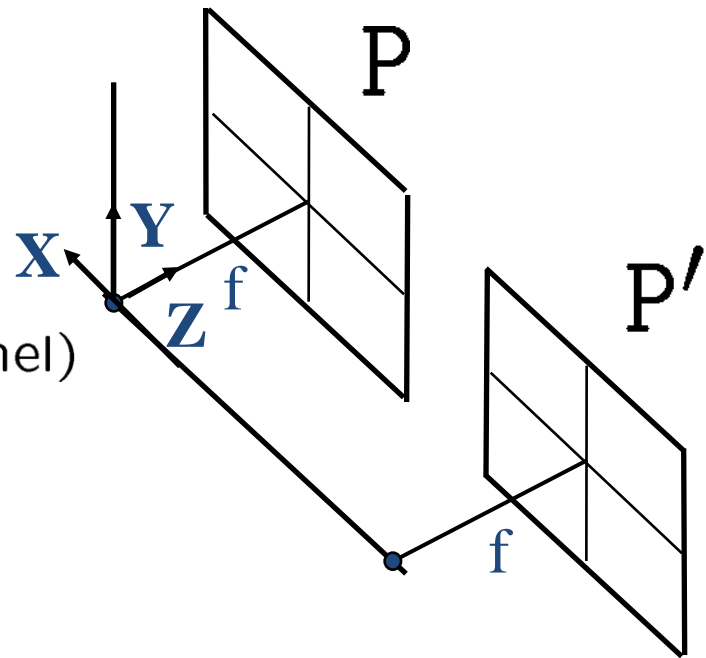
In this case

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

so that

$$e = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

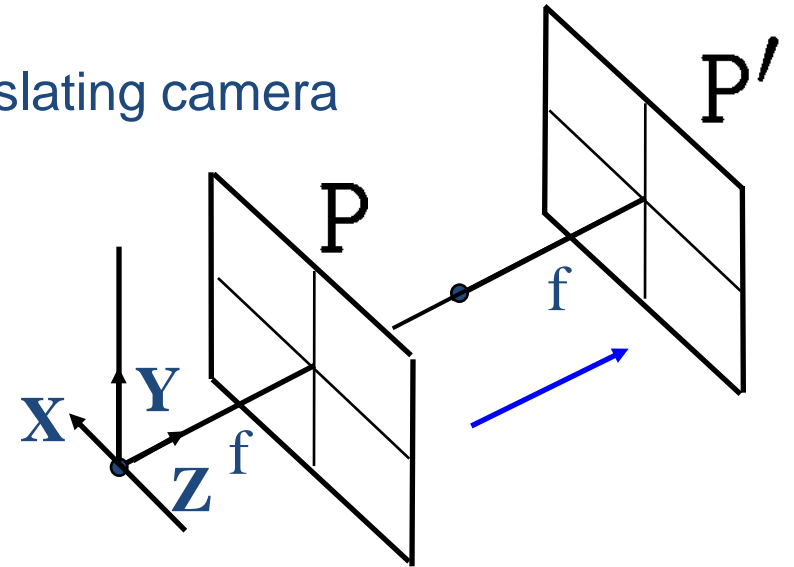
Geometric interpretation ?



Example II: compute  $F$  for a forward translating camera

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

$$K = K' = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R = I \quad \mathbf{t} = \begin{pmatrix} 0 \\ 0 \\ t_z \end{pmatrix}$$

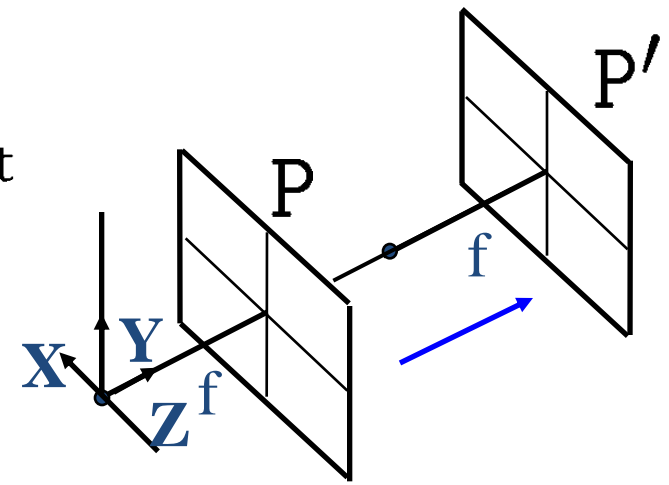


$$\begin{aligned} F &= K'^{-\top} [\mathbf{t}]_{\times} R K^{-1} \\ &= \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -t_z & 0 \\ t_z & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{aligned}$$

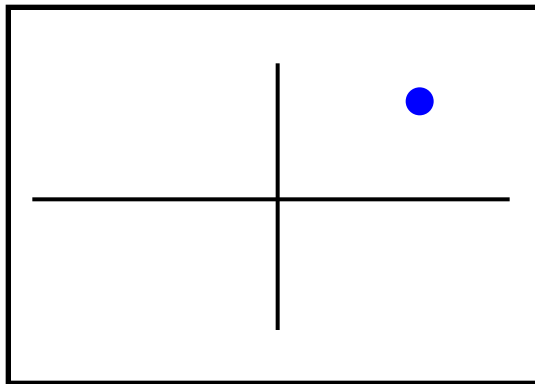
From  $\mathbf{l}' = \mathbf{F}\mathbf{x}$  the epipolar line for the point  $\mathbf{x} = (x, y, 1)^\top$  is

$$\mathbf{l}' = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} -y \\ x \\ 0 \end{pmatrix}$$

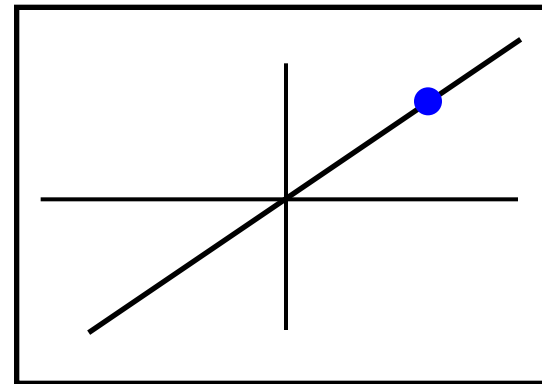
The points  $(x, y, 1)^\top$  and  $(0, 0, 1)^\top$  lie on this line



first image



second image







# Summary: Properties of the Fundamental matrix

- $F$  is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence:**  
if  $\mathbf{x}$  and  $\mathbf{x}'$  are corresponding image points, then  $\mathbf{x}'^T F \mathbf{x} = 0$ .
- **Epipolar lines:**
  - ◇  $\mathbf{l}' = F \mathbf{x}$  is the epipolar line corresponding to  $\mathbf{x}$ .
  - ◇  $\mathbf{l} = F^T \mathbf{x}'$  is the epipolar line corresponding to  $\mathbf{x}'$ .
- **Epipoles:**
  - ◇  $F \mathbf{e} = \mathbf{0}$ .
  - ◇  $F^T \mathbf{e}' = \mathbf{0}$ .
- **Computation from camera matrices  $P, P'$ :**  
 $P = K[I \mid \mathbf{0}]$ ,  $P' = K'[R \mid \mathbf{t}]$ ,  $F = K'^{-T}[\mathbf{t}]_{\times} R K^{-1}$



# Stereo correspondence algorithms

# Problem statement

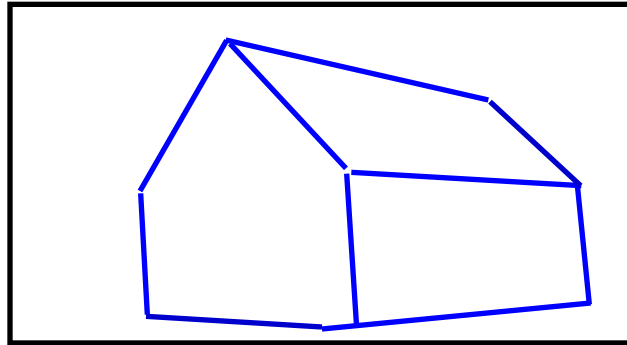
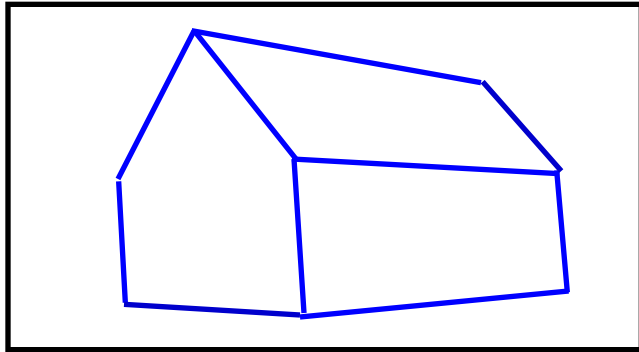
Given: two images and their associated cameras  
compute  
corresponding image points.

Algorithms may be classified into two types:

1. Dense: compute a correspondence at every pixel
2. Sparse: compute correspondences only for features

The methods may be top down or bottom up

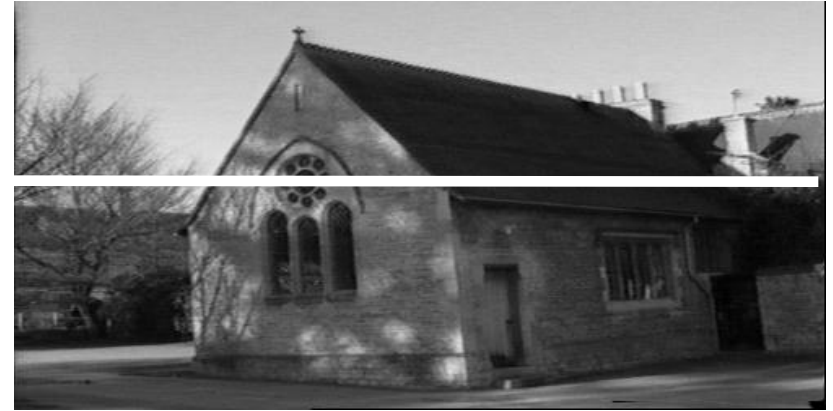
# Top down matching



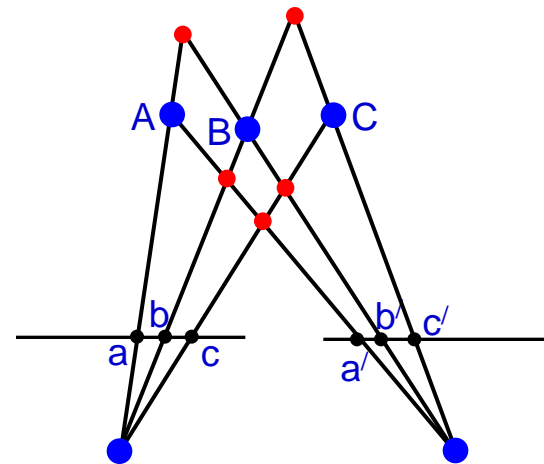
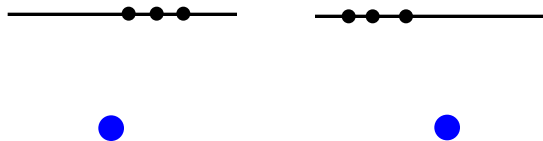
1. Group model (house, windows, etc) independently in each image
2. Match points (vertices) between images

# Bottom up matching

- epipolar geometry reduces the correspondence search from 2D to a 1D search on corresponding epipolar lines



- 1D correspondence problem



# Correspondence algorithms

Algorithms may be top down or bottom up – random dot stereograms are an existence proof that bottom up algorithms are possible

From here on only consider bottom up algorithms

→ Algorithms may be classified into two types: ←

1. Dense: compute a correspondence at every pixel
2. Sparse: compute correspondences only for features

# Example image pair – parallel cameras



# First image



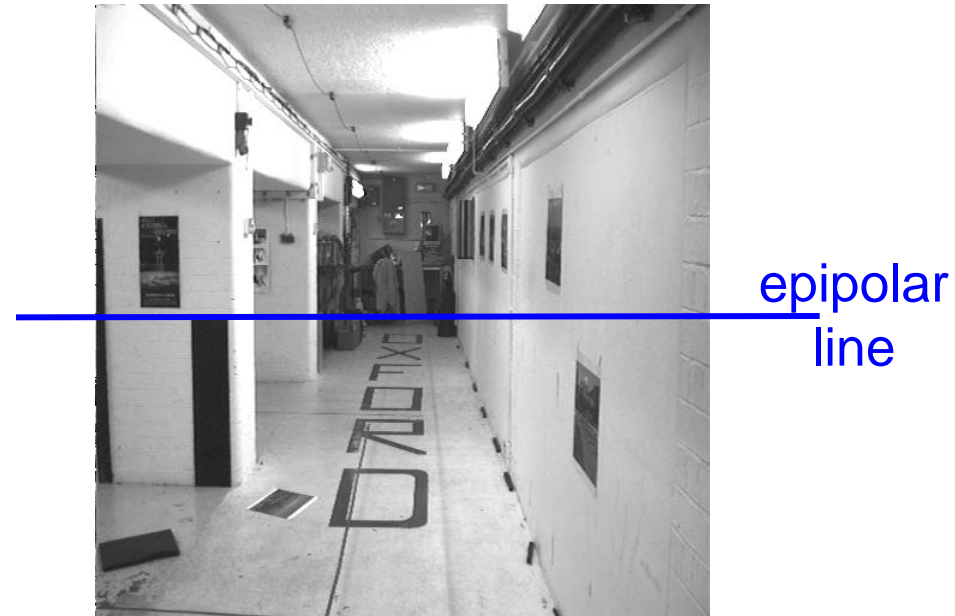
# Second image





# Dense correspondence algorithm

Parallel camera example – epipolar lines are corresponding rasters

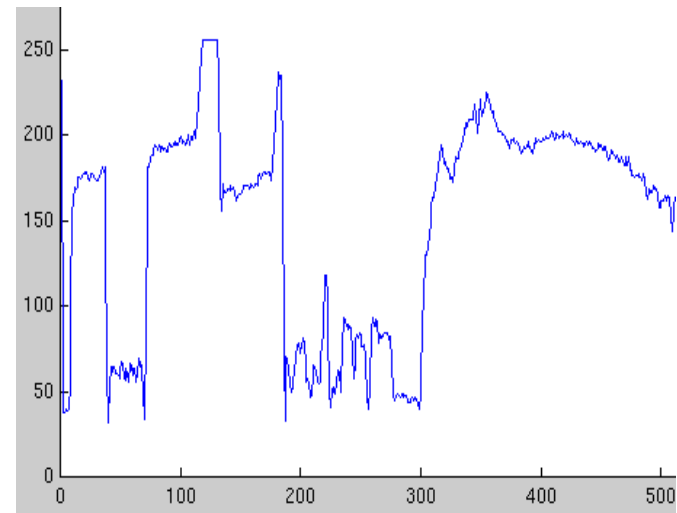
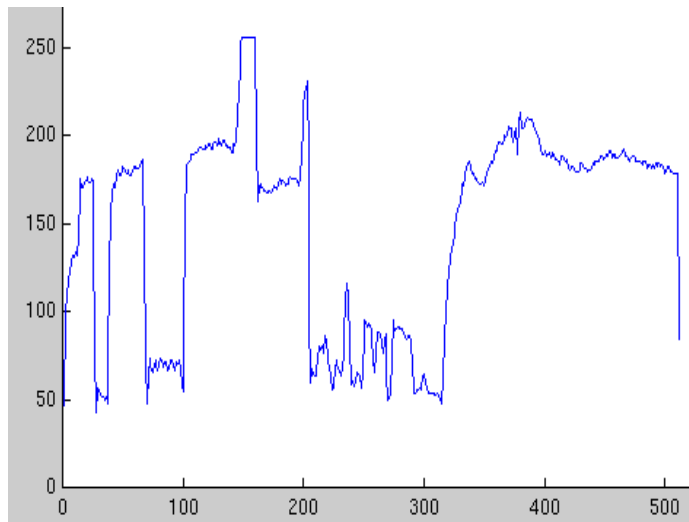
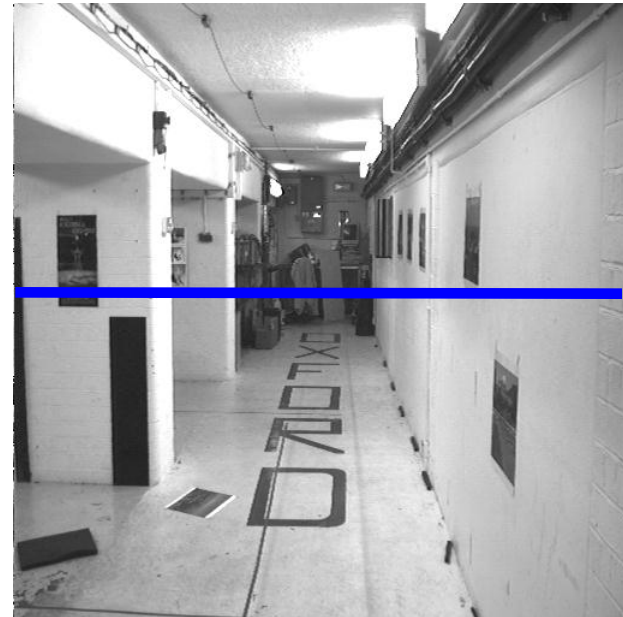


Search problem (geometric constraint): for each point in the left image, the corresponding point in the right image lies on the epipolar line (1D ambiguity)

Disambiguating assumption (photometric constraint): the intensity neighbourhood of corresponding points are similar across images

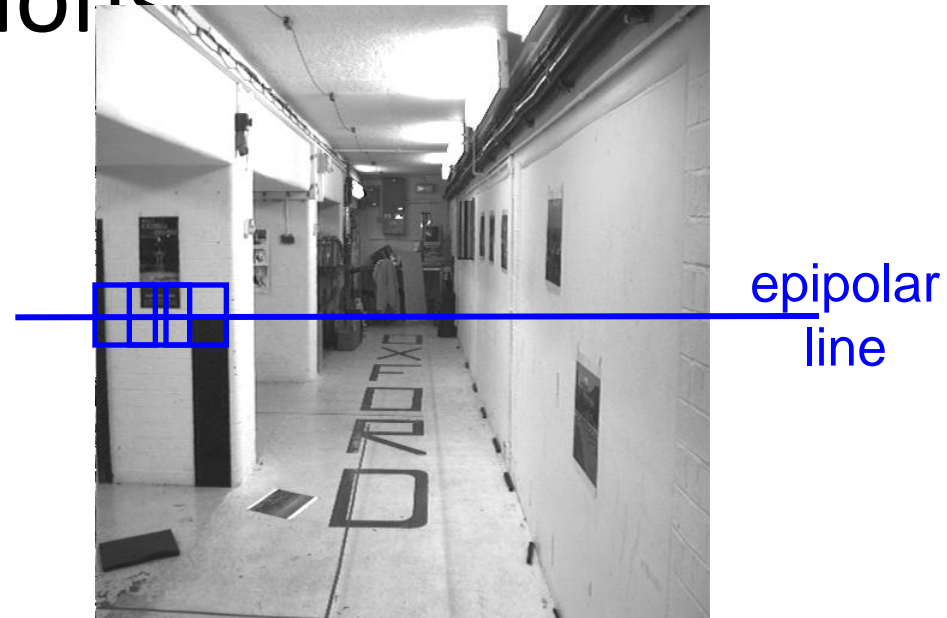
Measure similarity of neighbourhood intensity by cross-correlation

# Intensity profiles



- Clear correspondence between intensities, but also noise and ambiguity

# Cross-correlation of neighbourhood regions



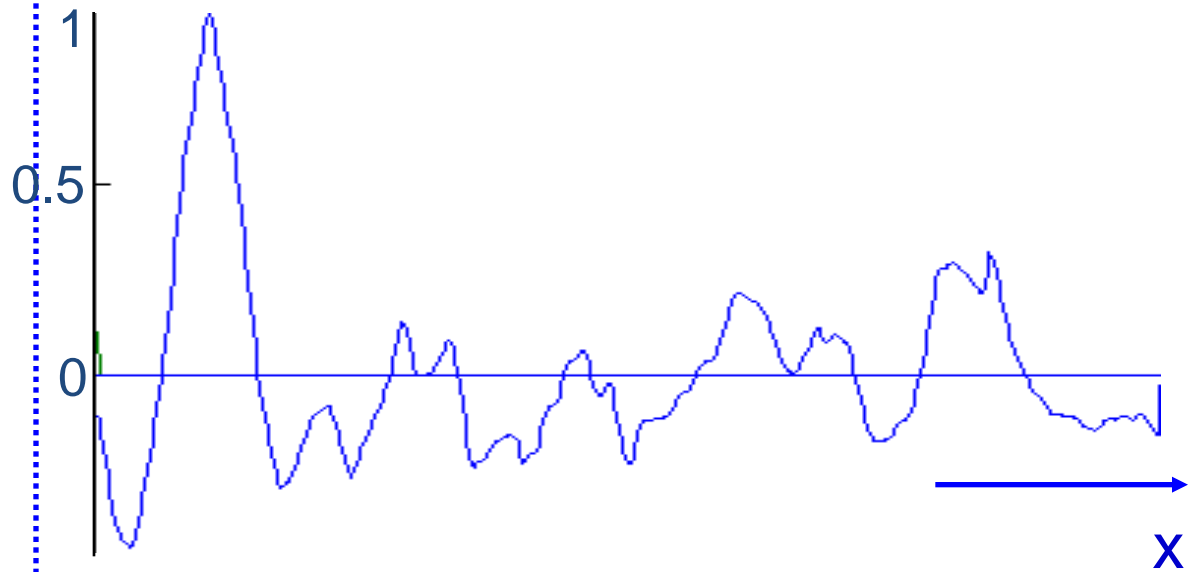
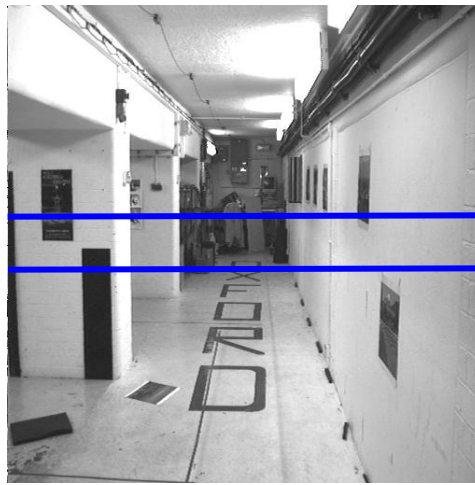
regions A, B, write as vectors  $\mathbf{a}$ ,  $\mathbf{b}$

translate so that mean is zero

$$\mathbf{a} \rightarrow \mathbf{a} - \langle \mathbf{a} \rangle, \quad \mathbf{b} \rightarrow \mathbf{b} - \langle \mathbf{b} \rangle$$

$$\text{cross correlation} = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| |\mathbf{b}|}$$

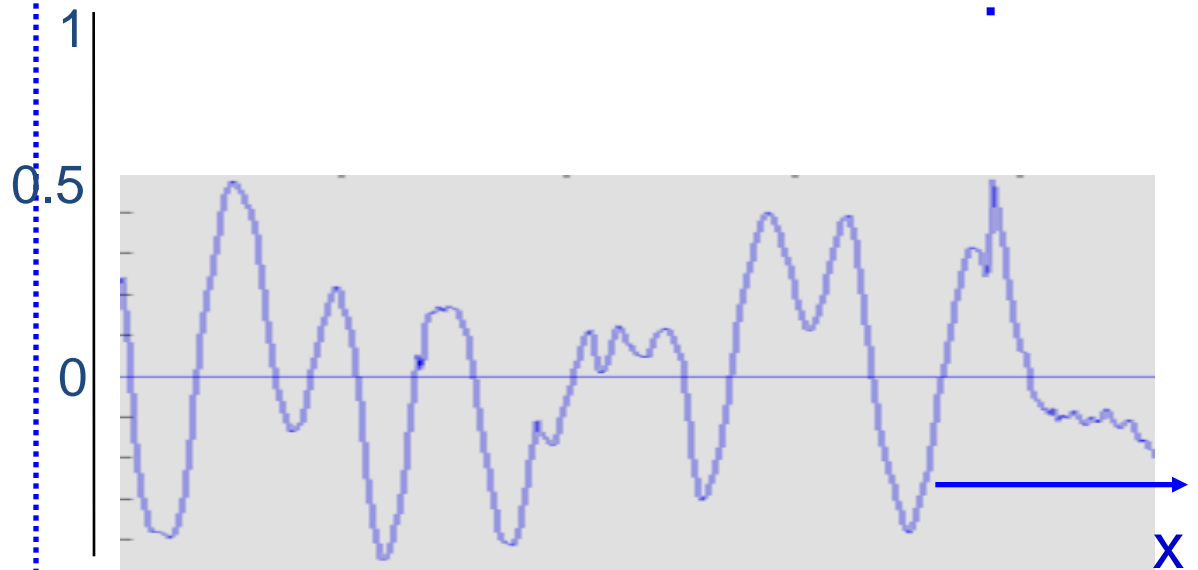
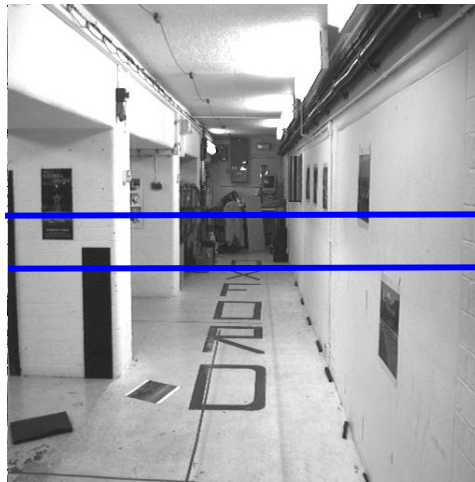
Invariant to  $I \rightarrow \alpha I + \beta$   
(exercise)



left image band

right image band

cross  
correlation



target region

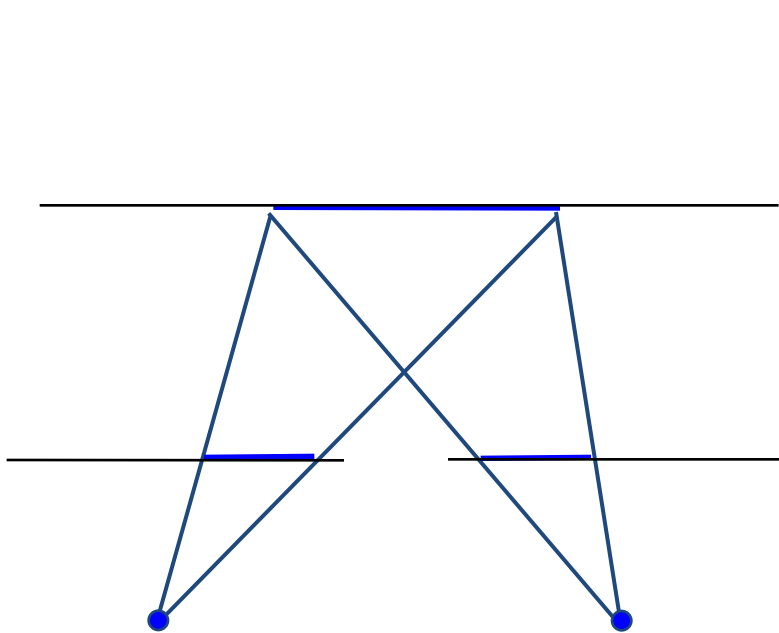
left image band

right image band

cross  
correlation

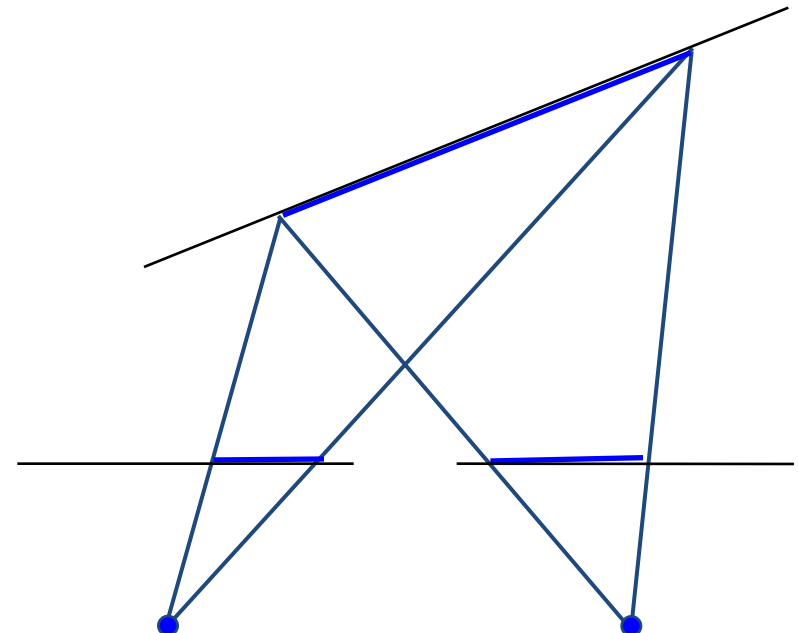
# Why is cross-correlation such a poor measure in the second case?

1. The neighbourhood region does not have a “distinctive” spatial intensity distribution
2. Foreshortening effects



fronto-parallel surface

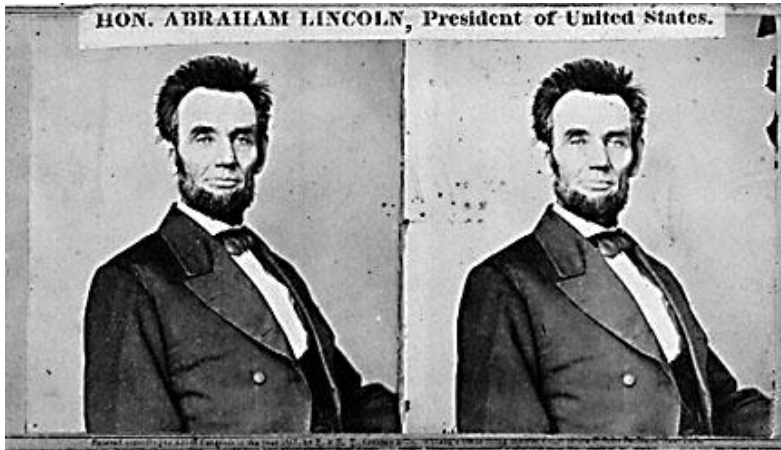
imaged length the same



slanting surface

imaged lengths differ

# Limitations of similarity constraint



Textureless surfaces



Occlusions, repetition



Non-Lambertian surfaces, specularities

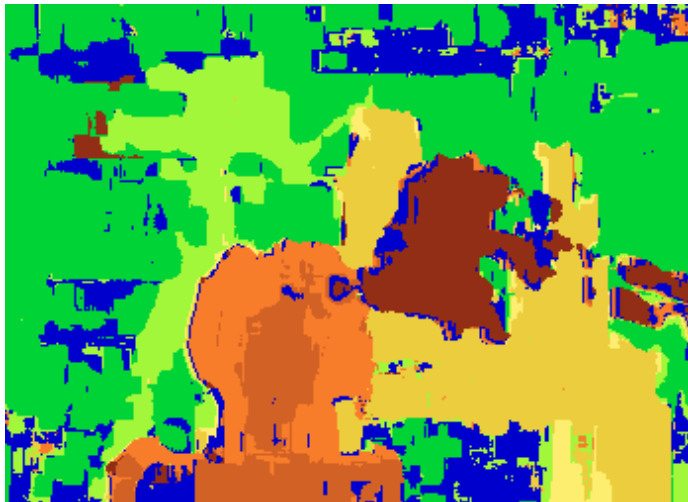


# Results with window search

Data



Window-based matching



Ground truth





# Sketch of a dense correspondence

For each pixel in the left image

- compute the neighbourhood cross correlation along the corresponding epipolar line in the right image
- the corresponding pixel is the one with the highest cross correlation

## Parameters

- size (scale) of neighbourhood
- search disparity

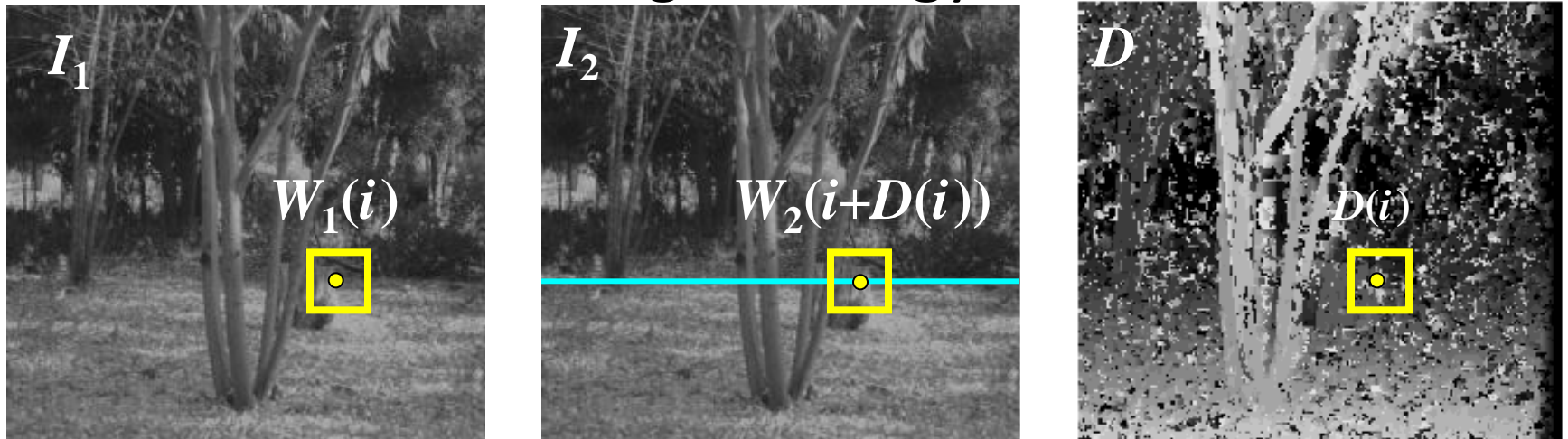
## Other constraints

- uniqueness
- ordering
- smoothness of disparity field

## Applicability

- textured scene, largely fronto-parallel

# Stereo matching as energy minimization



MAP estimate of disparity image  $D$ :  $P(D | I_1, I_2) \propto P(I_1, I_2 | D)P(D)$

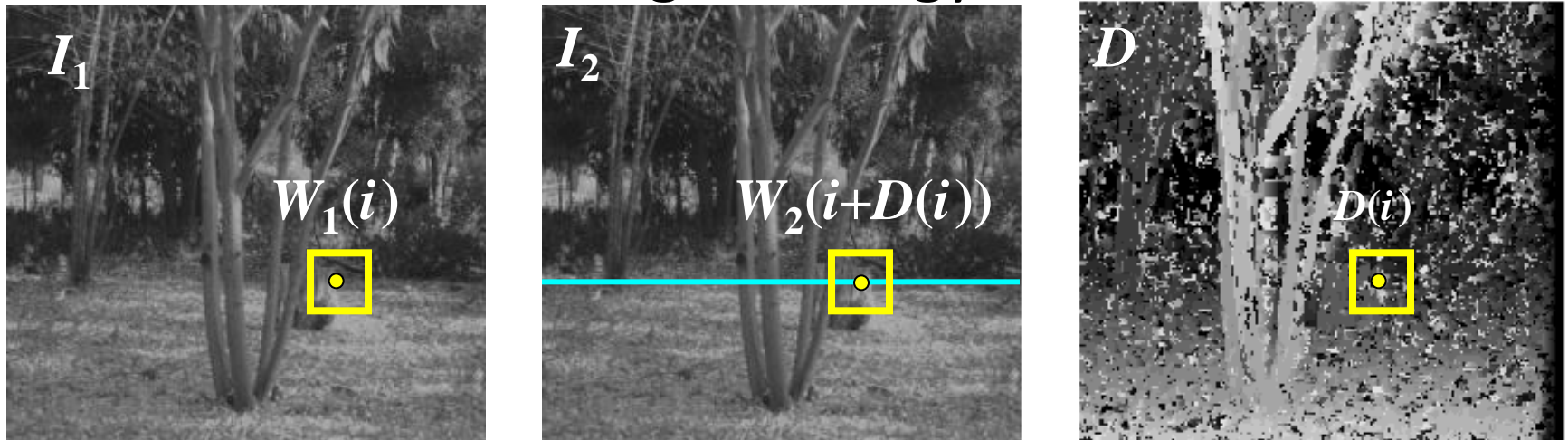
$$-\log P(D | I_1, I_2) \propto -\log P(I_1, I_2 | D) - \log P(D)$$

$$E = \alpha E_{\text{data}}(I_1, I_2, D) + \beta E_{\text{smooth}}(D)$$

$$E_{\text{data}} = \sum_i (W_1(i) - W_2(i + D(i)))^2$$

$$E_{\text{smooth}} = \sum_{\text{neighbors } i, j} \rho(D(i) - D(j))$$

# Stereo matching as energy minimization



$$E = \alpha E_{\text{data}}(I_1, I_2, D) + \beta E_{\text{smooth}}(D)$$

$$E_{\text{data}} = \sum_i (W_1(i) - W_2(i + D(i)))^2$$

$$E_{\text{smooth}} = \sum_{\text{neighbors } i, j} \rho(D(i) - D(j))$$

- Energy functions of this form can be minimized using *graph cuts*

Y. Boykov, O. Veksler, and R. Zabih, [Fast Approximate Energy Minimization via Graph Cuts](#), PAMI 2001

# Graph cuts solution



Graph cuts



Ground truth

Y. Boykov, O. Veksler, and R. Zabih, [Fast Approximate Energy Minimization via Graph Cuts](#), PAMI 2001

For the latest and greatest: <http://www.middlebury.edu/stereo/>

## Example dense correspondence algorithm



left image



right image



## 3D reconstruction



right image



depth map  
intensity = depth

# Texture mapped 3D triangulation



# Pentagon example

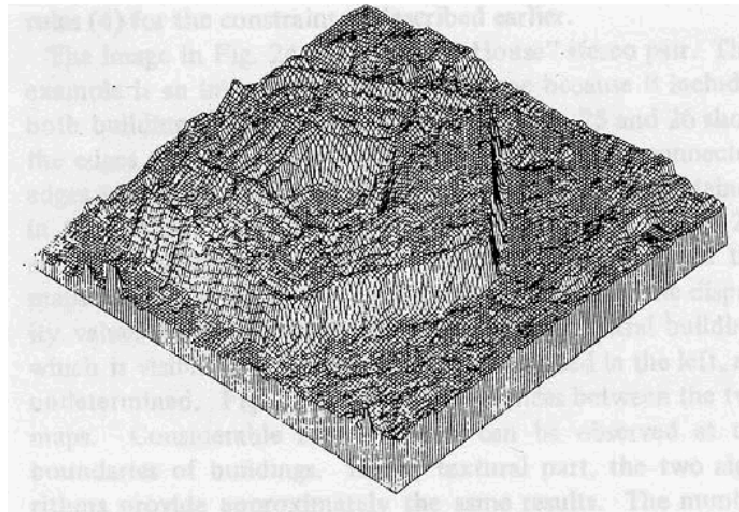
left image



right image



range map

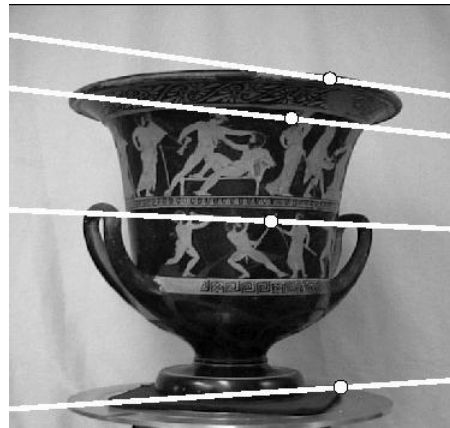
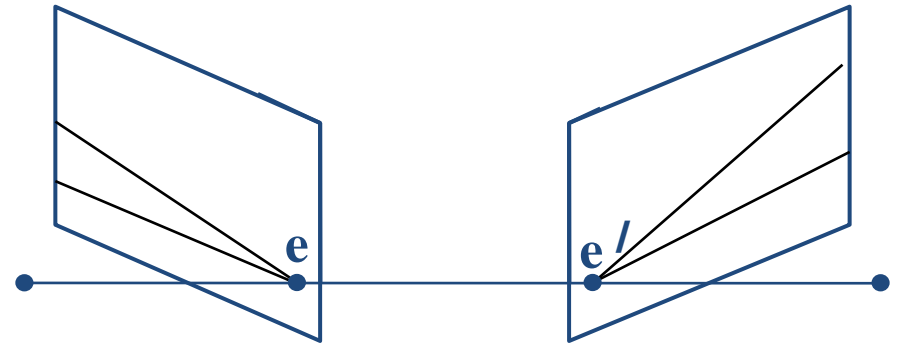




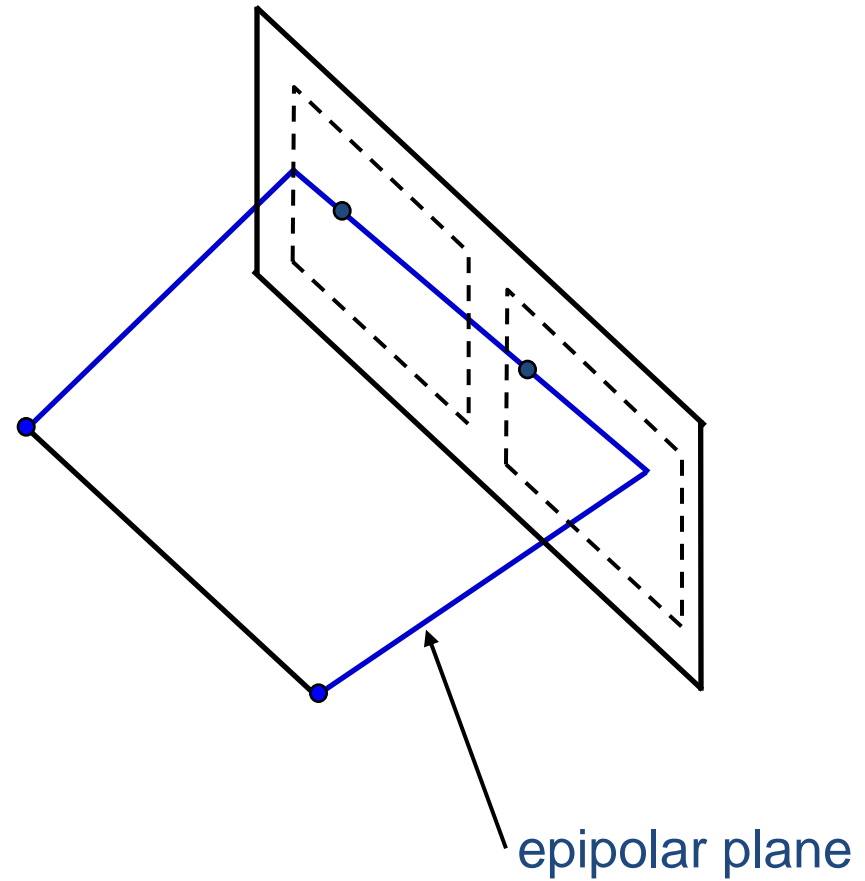
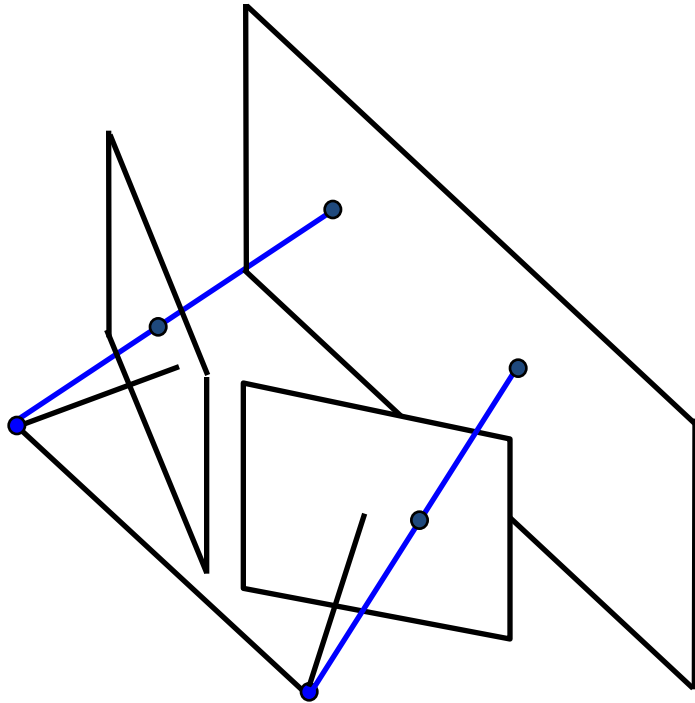
# Rectification

## For converging cameras

- epipolar lines are not parallel



## Project images onto plane parallel to baseline



# Rectification continued

Convert converging cameras to parallel camera geometry by an image mapping

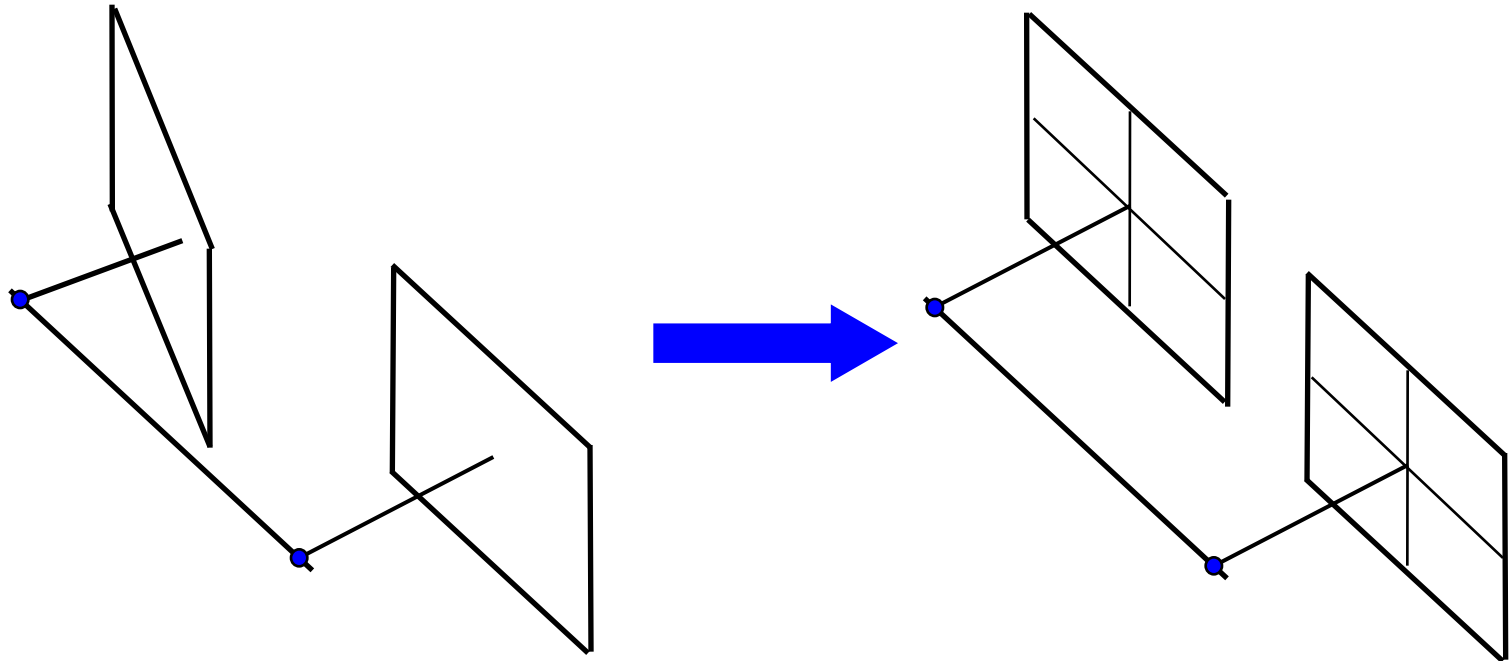


Image mapping is a 2D homography (projective transformation)

$$H = KRK^{-1} \quad (\text{exercise})$$

# Rectification continued

Convert converging cameras to parallel camera geometry by an image mapping

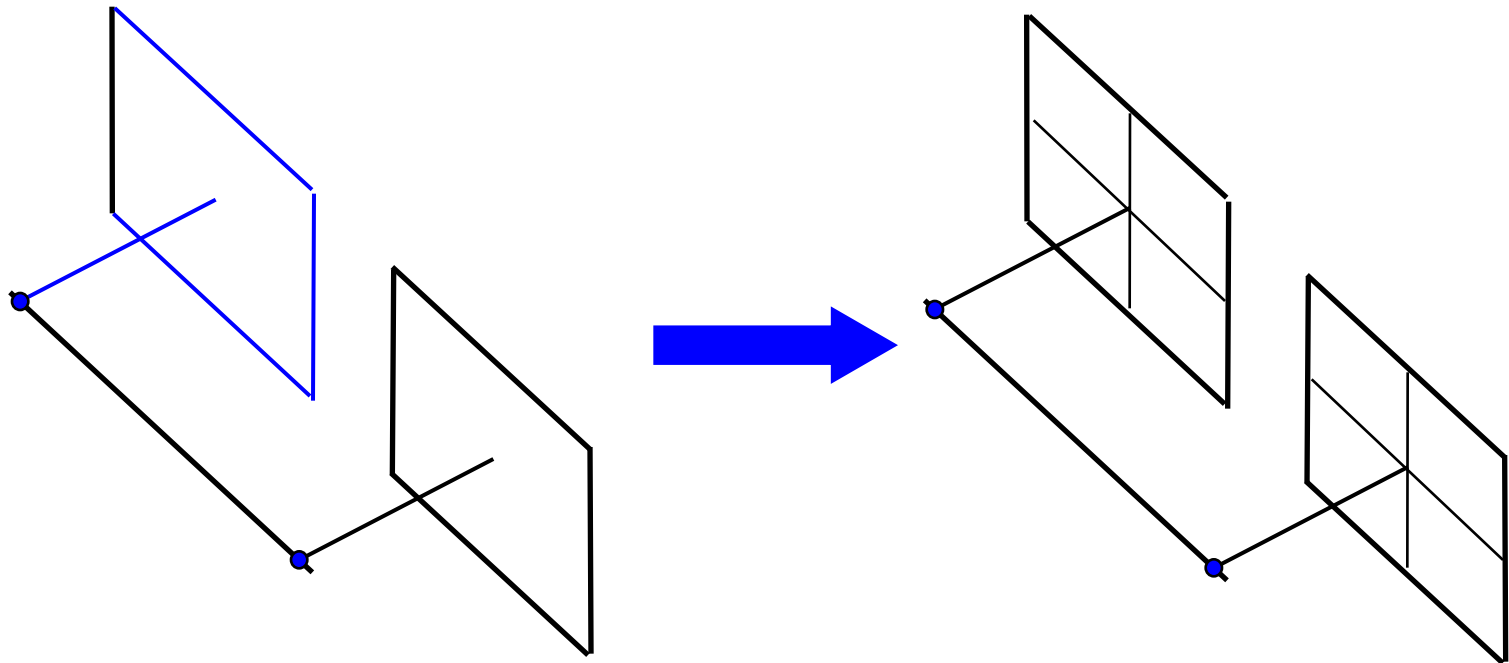
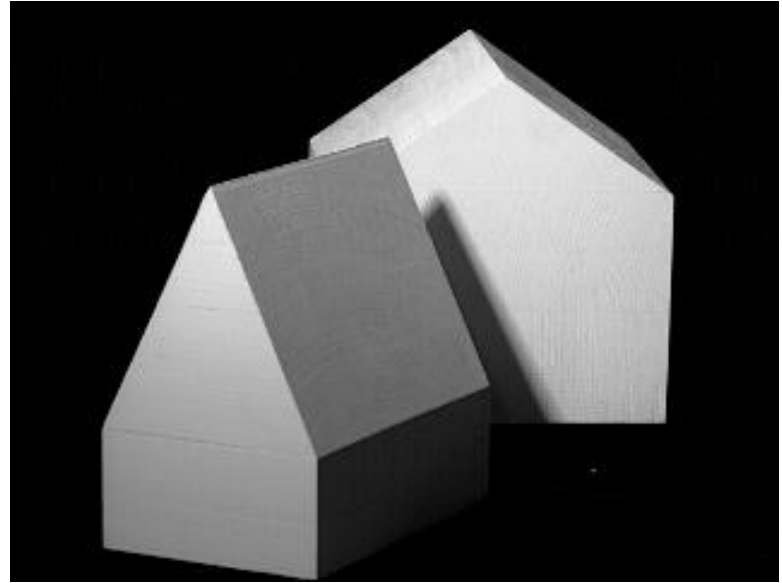
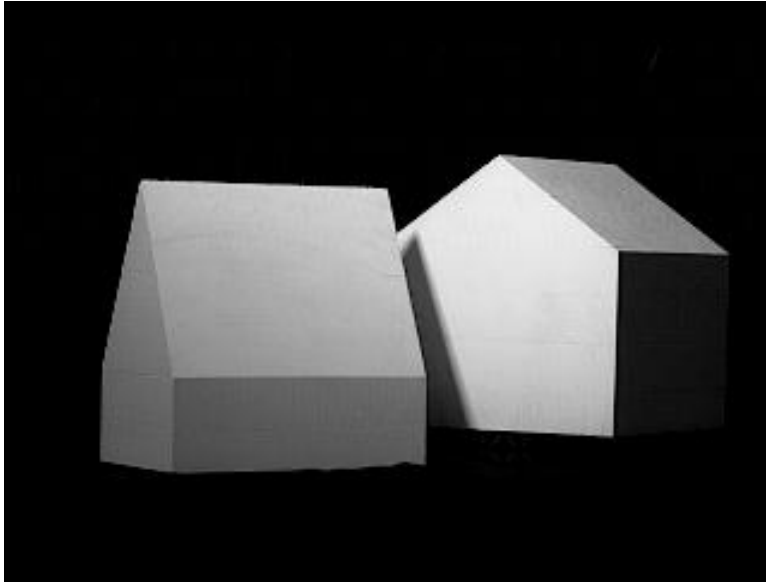


Image mapping is a 2D homography (projective transformation)

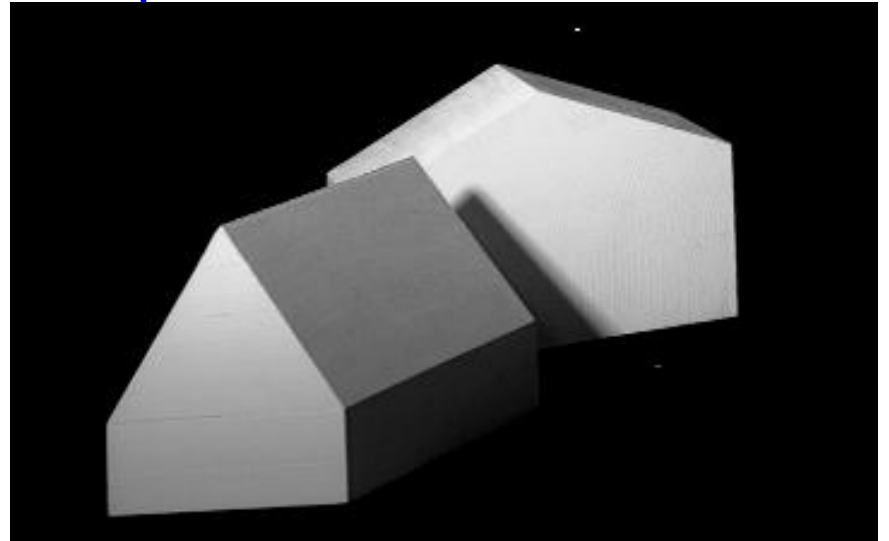
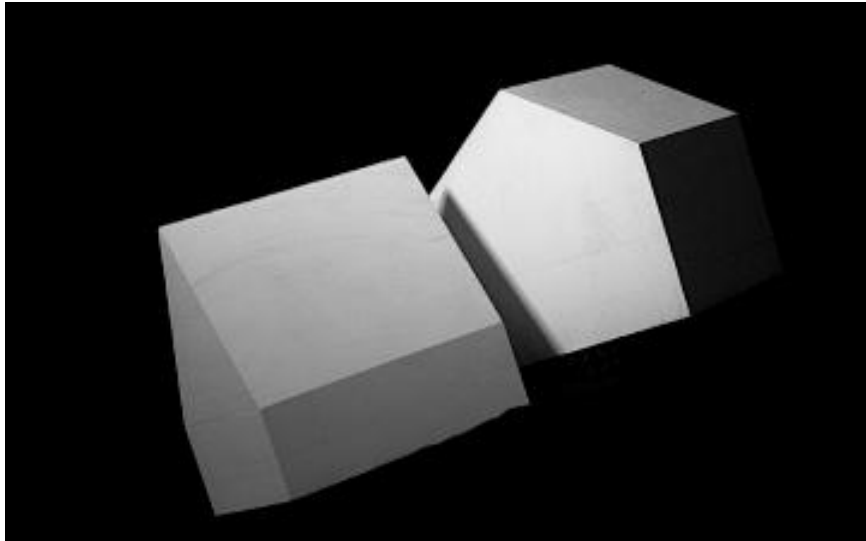
$$H = KRK^{-1} \quad (\text{exercise})$$

## Example

original stereo pair



rectified stereo pair



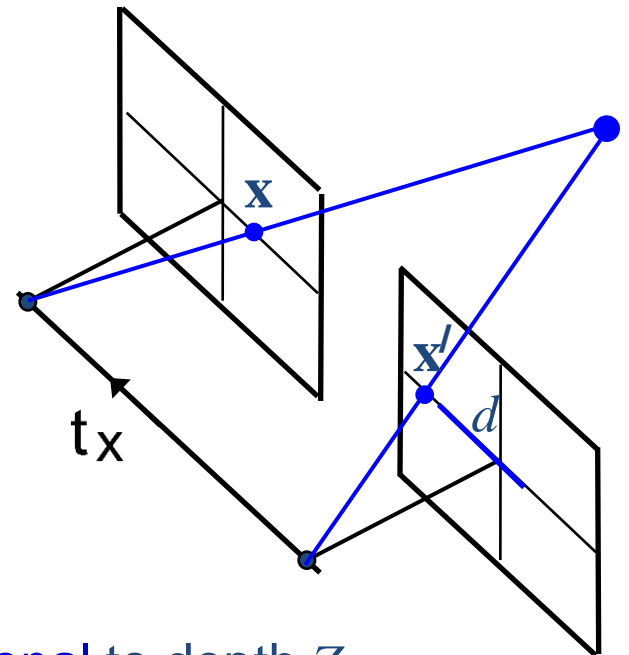
## Example: depth and disparity for a parallel camera stereo rig

$$K = K' = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad R = I \quad \mathbf{t} = \begin{pmatrix} t_x \\ 0 \\ 0 \end{pmatrix}$$

Then,  $y' = y$ , and the disparity  $d = x' - x = \frac{ft_x}{Z}$

### Derivation

$$\frac{x}{f} = \frac{X}{Z} \quad \frac{x'}{f} = \frac{X + t_x}{Z}$$
$$\frac{x'}{f} = \frac{x}{f} + \frac{t_x}{Z}$$



### Note

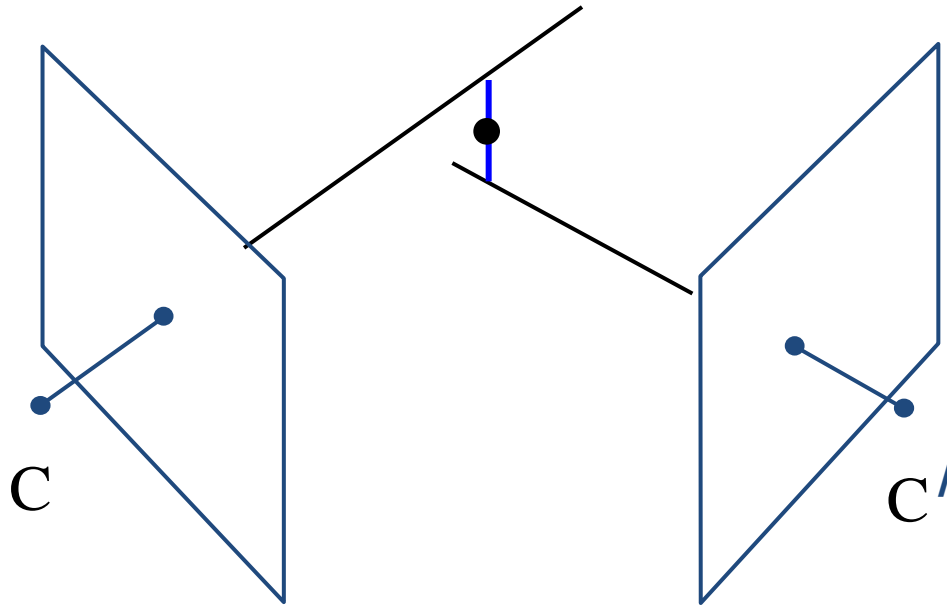
- image movement (disparity) is inversely proportional to depth  $Z$

as  $z \rightarrow \infty$ ,  $d \rightarrow 0$

- depth is inversely proportional to disparity

# Triangulation

# 1. Vector solution



Compute the mid-point of the shortest line between the two rays



## 2. Linear triangulation (algebraic solution)

Use the equations  $\mathbf{x} = \mathbf{P}\mathbf{X}$  and  $\mathbf{x}' = \mathbf{P}'\mathbf{X}$  to solve for  $\mathbf{X}$

For the first camera:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} \mathbf{p}^{1\top} \\ \mathbf{p}^{2\top} \\ \mathbf{p}^{3\top} \end{bmatrix}$$

where  $\mathbf{p}^{i\top}$  are the rows of  $\mathbf{P}$

- eliminate unknown scale in  $\lambda\mathbf{x} = \mathbf{P}\mathbf{X}$  by forming a cross product  $\mathbf{x} \times (\mathbf{P}\mathbf{X}) = \mathbf{0}$

$$x(\mathbf{p}^{3\top}\mathbf{X}) - (\mathbf{p}^{1\top}\mathbf{X}) = 0$$

$$y(\mathbf{p}^{3\top}\mathbf{X}) - (\mathbf{p}^{2\top}\mathbf{X}) = 0$$

$$x(\mathbf{p}^{2\top}\mathbf{X}) - y(\mathbf{p}^{1\top}\mathbf{X}) = 0$$

- rearrange as (first two equations only)

$$\begin{bmatrix} x\mathbf{p}^{3\top} - \mathbf{p}^{1\top} \\ y\mathbf{p}^{3\top} - \mathbf{p}^{2\top} \end{bmatrix} \mathbf{X} = \mathbf{0}$$

Similarly for the second camera:

$$\begin{bmatrix} x' \mathbf{p}'^{3\top} - \mathbf{p}'^{1\top} \\ y' \mathbf{p}'^{3\top} - \mathbf{p}'^{2\top} \end{bmatrix} \mathbf{X} = \mathbf{0}$$

Collecting together gives

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

where  $\mathbf{A}$  is the  $4 \times 4$  matrix

$$\mathbf{A} = \begin{bmatrix} x \mathbf{p}^{3\top} - \mathbf{p}^{1\top} \\ y \mathbf{p}^{3\top} - \mathbf{p}^{2\top} \\ x' \mathbf{p}'^{3\top} - \mathbf{p}'^{1\top} \\ y' \mathbf{p}'^{3\top} - \mathbf{p}'^{2\top} \end{bmatrix}$$

from which  $\mathbf{X}$  can be solved up to scale.

**Problem:** does not minimize anything meaningful

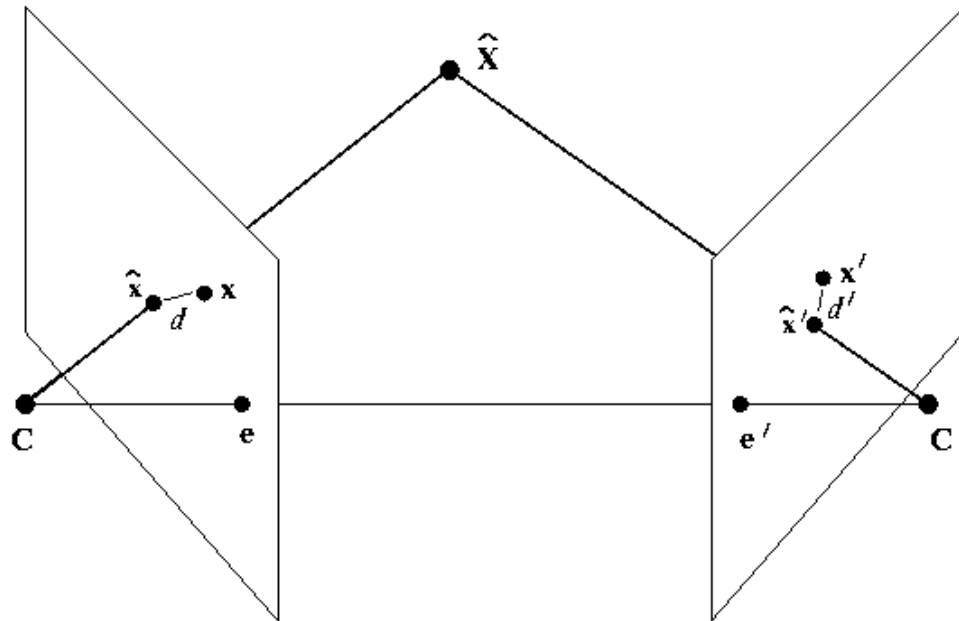
**Advantage:** extends to more than two views

### 3. Minimizing a geometric/statistical

The idea is to estimate a 3D point  $\hat{\mathbf{X}}$  which exactly satisfies the supplied camera geometry, so it projects as

$$\hat{\mathbf{x}} = \mathbf{P}\hat{\mathbf{X}} \quad \hat{\mathbf{x}}' = \mathbf{P}'\hat{\mathbf{X}}$$

and the aim is to estimate  $\hat{\mathbf{X}}$  from the image measurements  $\mathbf{x}$  and  $\mathbf{x}'$ .



$$\min_{\hat{\mathbf{X}}} \mathcal{C}(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x}}')^2$$

where  $d(*, *)$  is the Euclidean distance between the points.

- It can be shown that if the measurement noise is Gaussian mean zero,  $\sim N(0, \sigma^2)$ , then minimizing geometric error is the **Maximum Likelihood Estimate** of  $X$
- The minimization appears to be over three parameters (the position  $X$ ), but the problem can be reduced to a minimization over one parameter

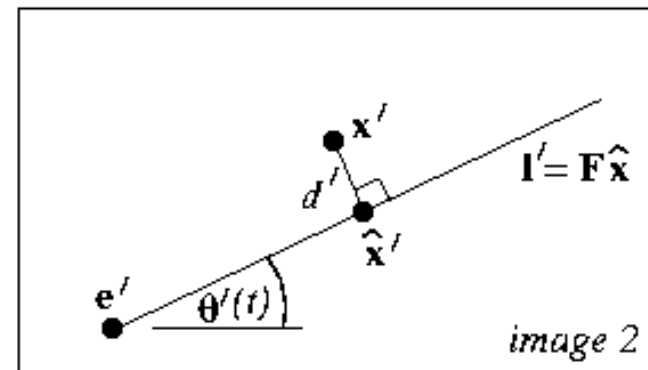
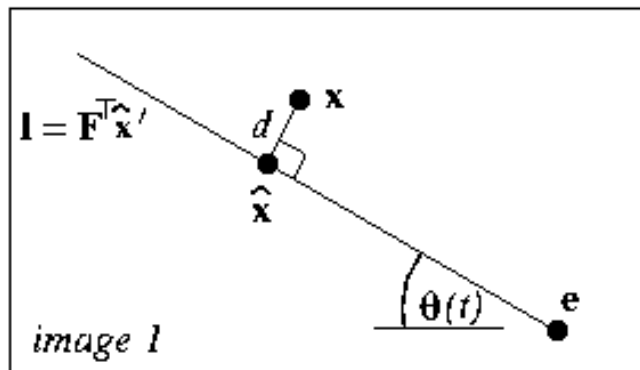
# Different formulation of the problem

The minimization problem may be formulated differently:

- Minimize

$$d(\mathbf{x}, \mathbf{l})^2 + d(\mathbf{x}', \mathbf{l}')^2$$

- $\mathbf{l}$  and  $\mathbf{l}'$  range over all choices of corresponding epipolar lines.
- $\hat{\mathbf{x}}$  is the closest point on the line  $\mathbf{l}$  to  $\mathbf{x}$ .
- Same for  $\hat{\mathbf{x}}'$ .



## Minimization method

- Parametrize the pencil of epipolar lines in the first image by  $t$ , such that the epipolar line is  $\mathbf{l}(t)$
- Using  $\mathbf{F}$  compute the corresponding epipolar line in the second image  $\mathbf{l}'(t)$
- Express the distance function  $d(\mathbf{x}, \mathbf{l})^2 + d(\mathbf{x}', \mathbf{l}')^2$  explicitly as a function of  $t$
- Find the value of  $t$  that minimizes the distance function
- Solution is a 6<sup>th</sup> degree polynomial in  $t$

