Industrial Computer Vision
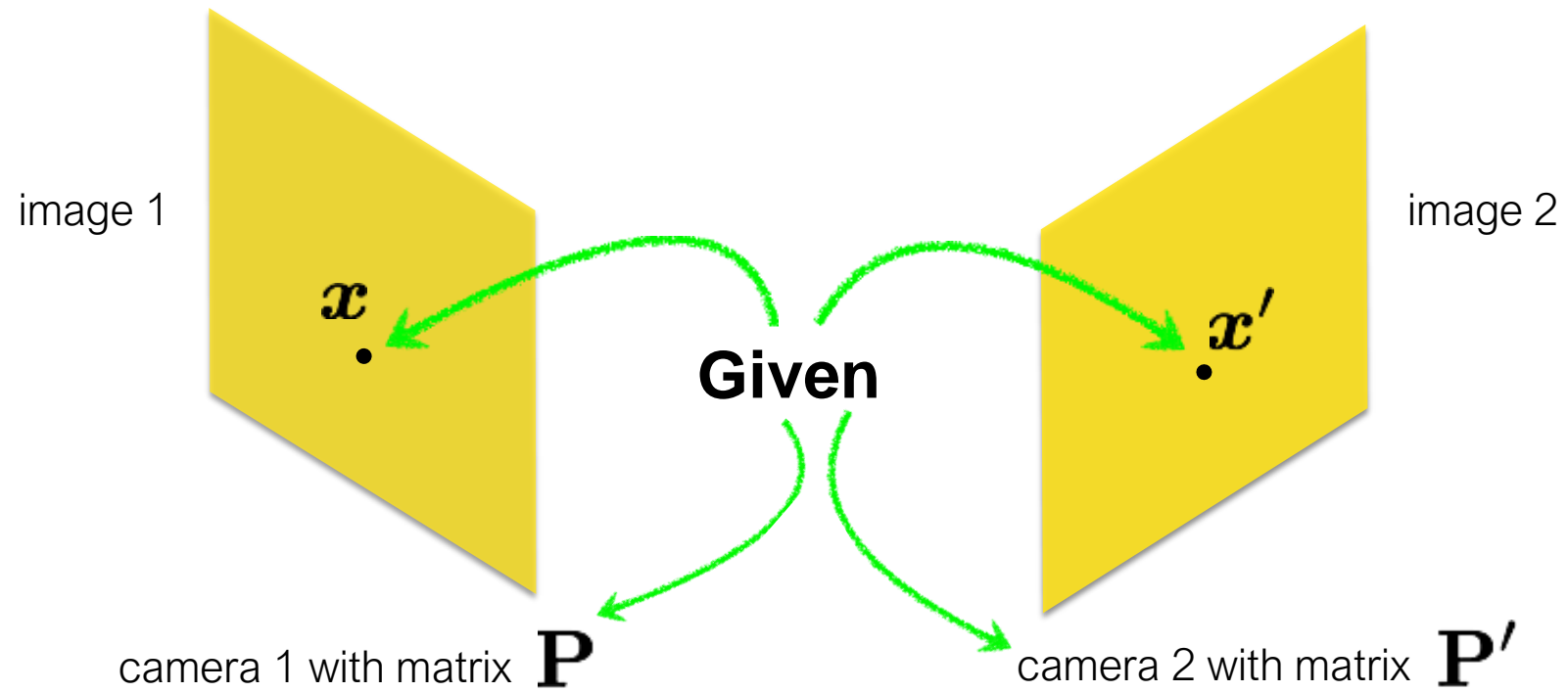## - Epipolar Geometry

12th lecture, 2021.11.24
Lecturer: Youngbae Hwang

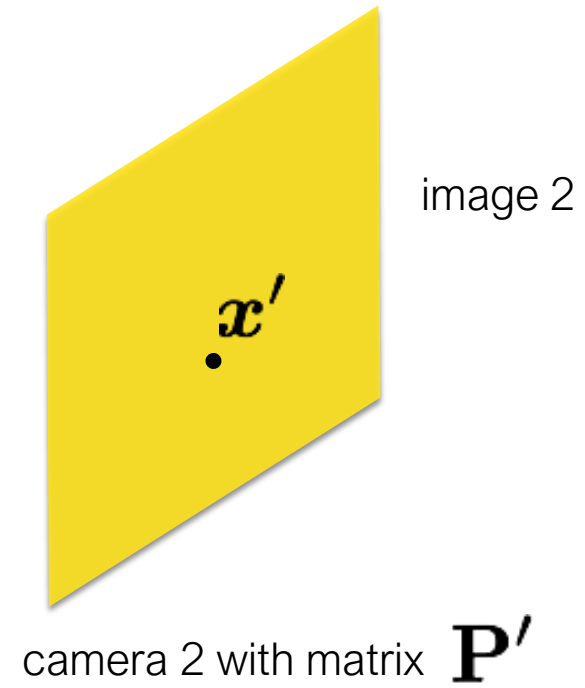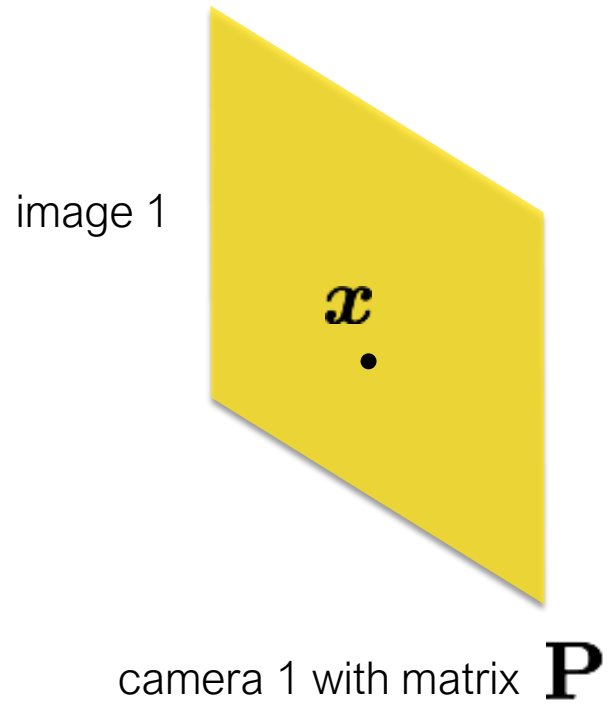# Contents

- Triangulation

- Epipolar Geometry

- Essential/Fundamental Matrix

- Stereo Rectification

image 1

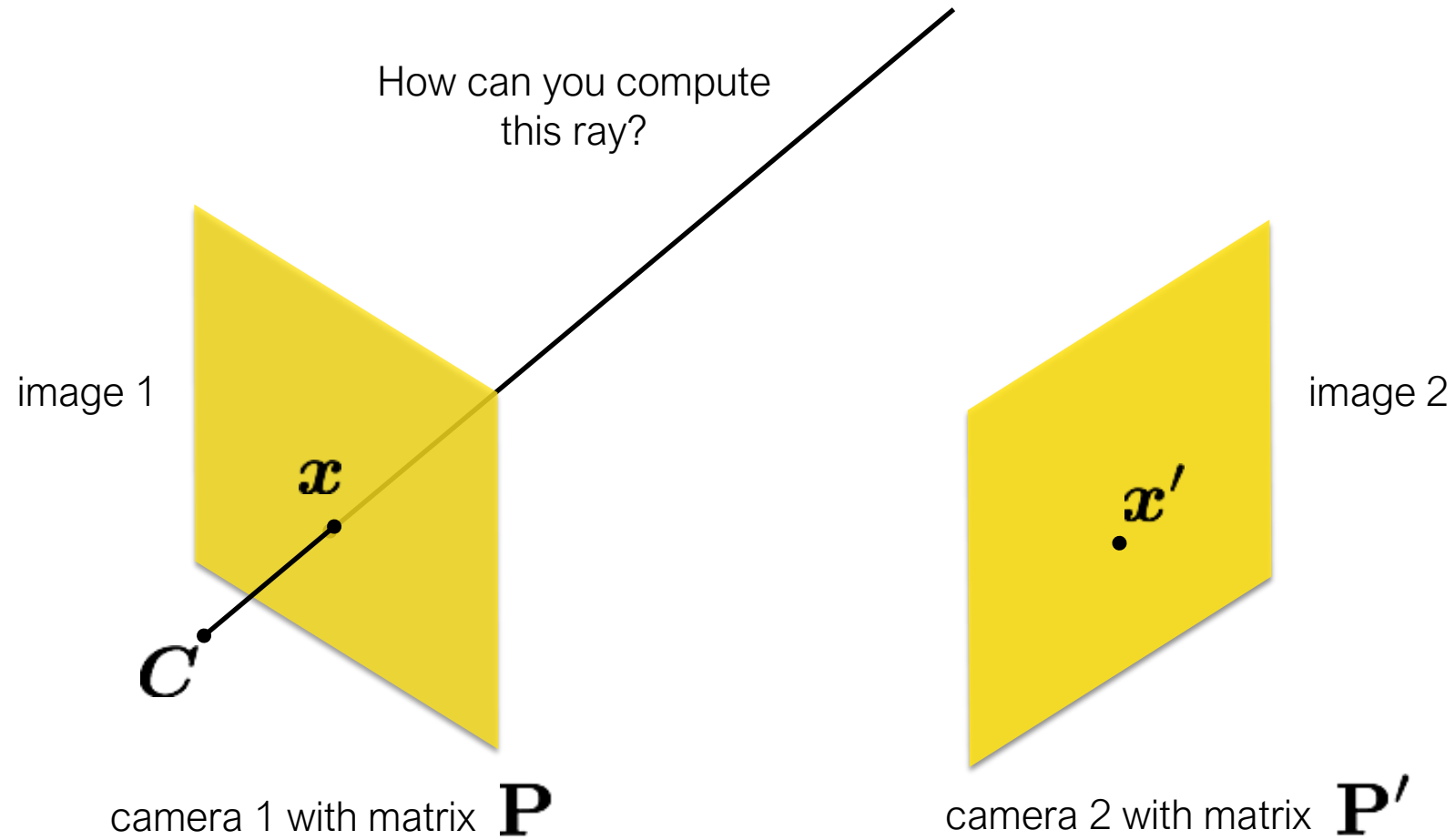x

Given

image 2

x'

camera 1 with matrix $\mathbf{P}$

camera 2 with matrix $\mathbf{P'}$

Triangulation

Which 3D points map to **x**?

image 1

x

camera 1 with matrix $\mathbf{P}$

image 2

x'

camera 2 with matrix $\mathbf{P}'$

How can you compute
this ray?

image 1

image 2

x

x'

C

camera 1 with matrix $\mathbf{P}$

camera 2 with matrix $\mathbf{P}'$
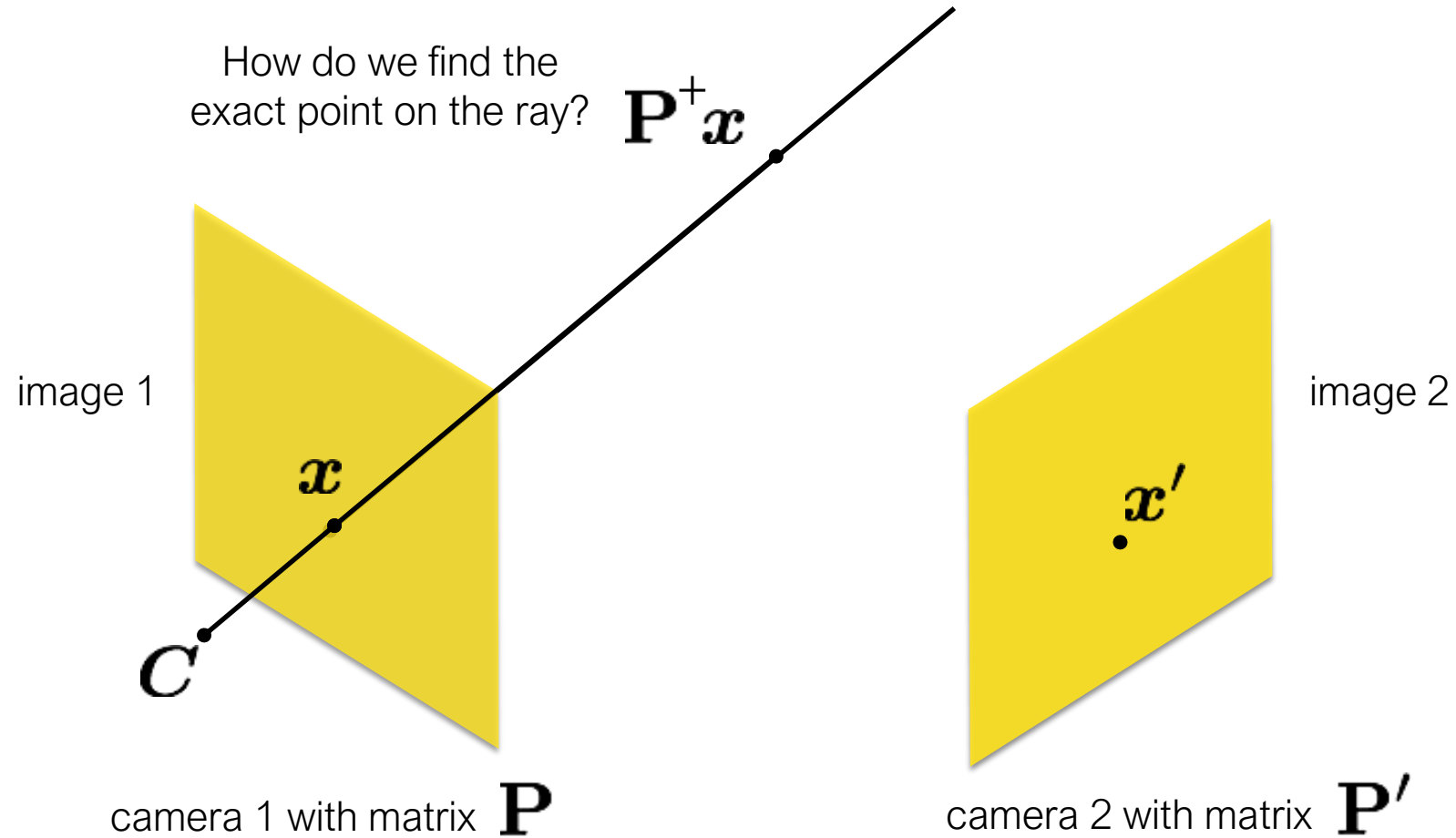
Create two points on the ray:
1) find the camera center; and
2) apply the pseudo-inverse of **P** on **x**.
Then connect the two points.
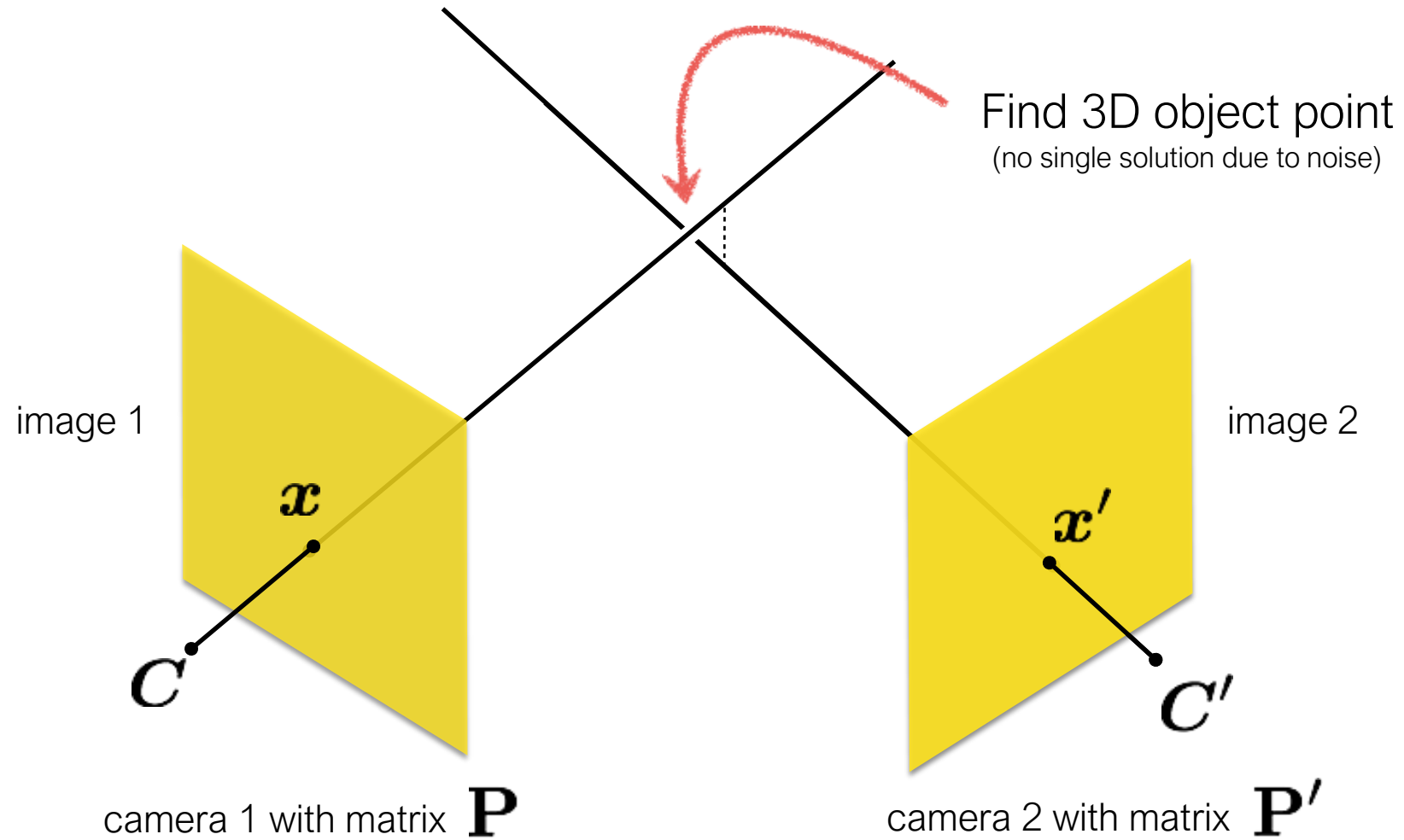**This procedure is called** backprojection

$$\mathbf{P}^{+}\boldsymbol{x}$$

Why does this poin
t map to **x**?

image 1

$$\boldsymbol{x}$$

image 2

$$\boldsymbol{x'}$$

$$\boldsymbol{C}$$

camera 1 with matrix $\mathbf{P}$

camera 2 with matrix $\mathbf{P'}$

How do we find the exact point on the ray? $\mathbf{P}^+\boldsymbol{x}$

image 1

image 2

$\boldsymbol{x}$

$\boldsymbol{x'}$

$\boldsymbol{C}$

camera 1 with matrix $\mathbf{P}$

camera 2 with matrix $\mathbf{P'}$

Find 3D object point
Will the lines intersect?

image 1

image 2

x

x'

C

C'

camera 1 with matrix $\mathbf{P}$

camera 2 with matrix $\mathbf{P}'$

Electronics Engineering, CBNU

Find 3D object point
(no single solution due to noise)

image 1

image 2

x

x'

C

C'

camera 1 with matrix $\mathbf{P}$

camera 2 with matrix $\mathbf{P}'$
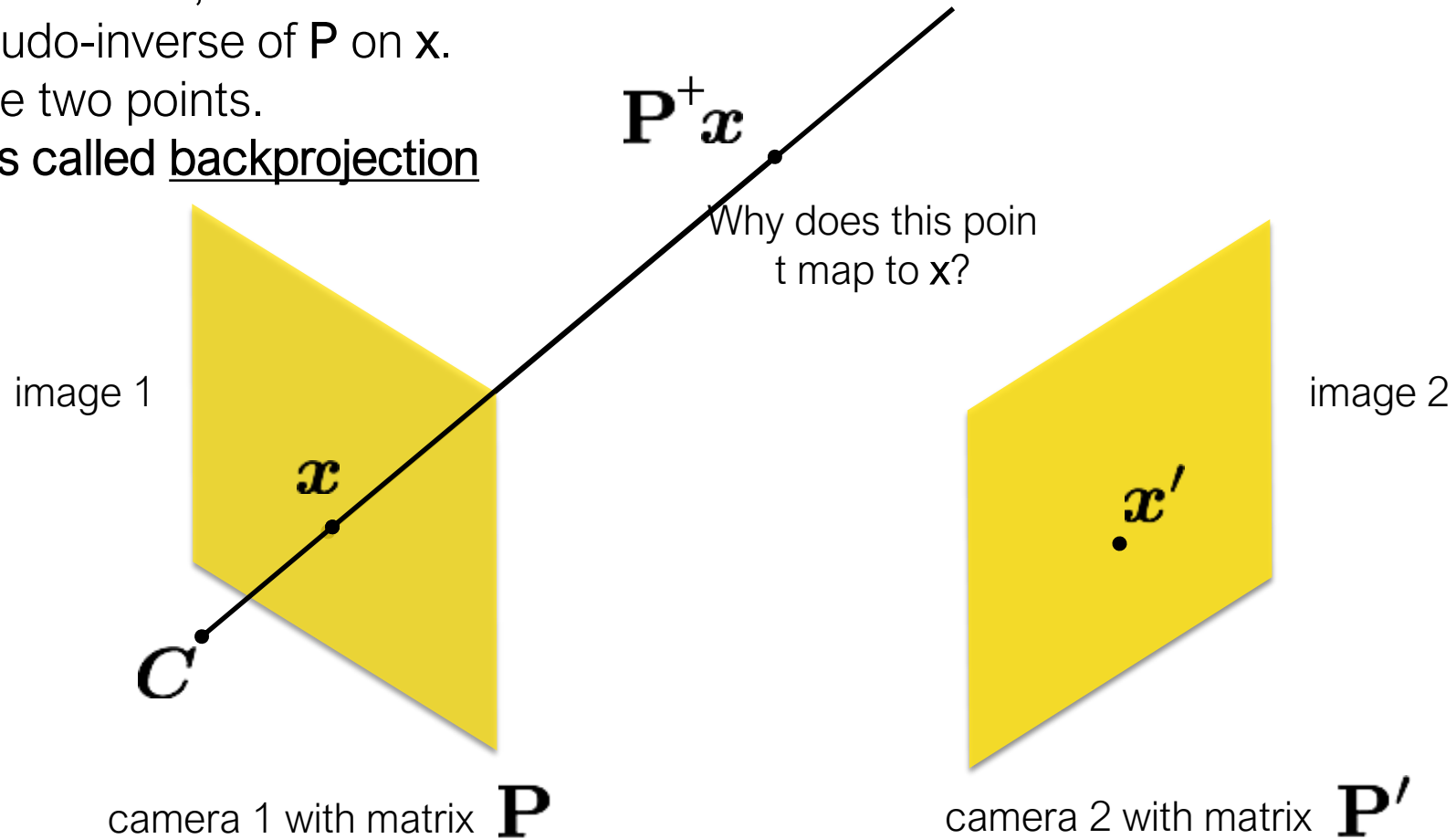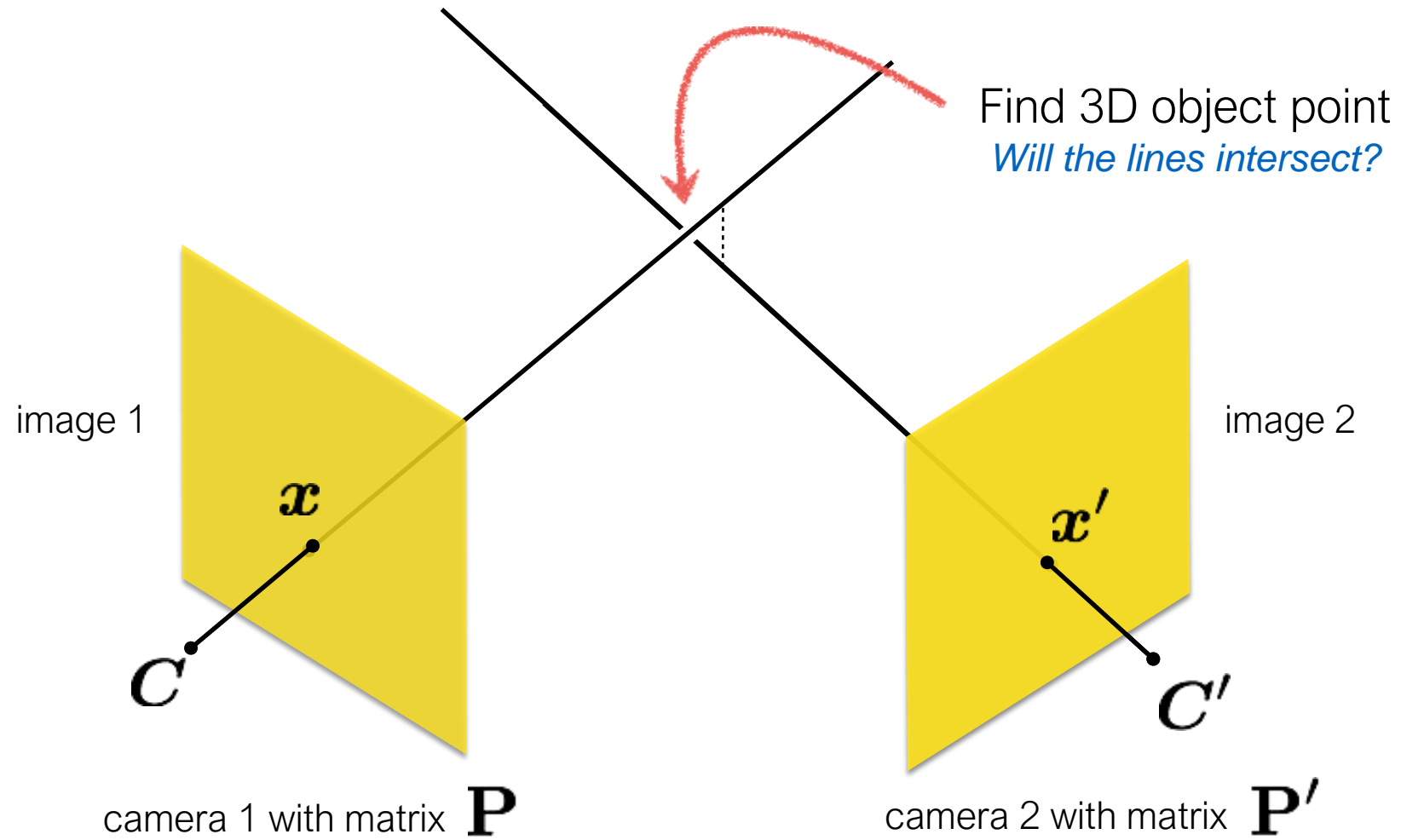
Given a set of (noisy) matched points

$$\{x_i, x_i'\}$$

and camera matrices

$$\mathbf{P}, \mathbf{P}'$$

Estimate the 3D point

$$\mathbf{X}$$

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

known   known

*Can we compute **X** from a single correspondence **x**?*

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

(homogeneous coordinate)

This is a similarity relation because it involves homogeneous coordinates

$$\mathbf{x} = \alpha\mathbf{P}X$$

(homorogeneous coordinate)

Same ray direction but differs by a scale factor

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

How do we solve for unknowns in a similarity relation?

Electronics Engineering, CBNU

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

(homogeneous co ordinate)

Also, this is a similarity relation because it involves homogeneous coordinates

$$\mathbf{x} = \alpha \mathbf{P}X$$

(inhomogeneous coordinate)

Same ray direction but differs by a scale factor

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

How do we solve for unknowns in a similarity relation?

Electronics Engineering, CBNU

$$\mathbf{x} = \alpha \mathbf{P} \mathbf{X}$$

Same direction but differs by a scale factor

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

Cross product of two vectors of same direction is zero
(this equality removes the scale factor)

Electronics Engineering, CBNU

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Do the same after first expanding out the camera matrix and points

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} \text{---} \ \boldsymbol{p}_1^\top \ \text{---} \\ \text{---} \ \boldsymbol{p}_2^\top \ \text{---} \\ \text{---} \ \boldsymbol{p}_3^\top \ \text{---} \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{X} \\ | \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} \boldsymbol{p}_1^\top \boldsymbol{X} \\ \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_3^\top \boldsymbol{X} \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} \boldsymbol{p}_1^\top \boldsymbol{X} \\ \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_3^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} y\boldsymbol{p}_3^\top \boldsymbol{X} - \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_1^\top \boldsymbol{X} - x\boldsymbol{p}_3^\top \boldsymbol{X} \\ x\boldsymbol{p}_2^\top \boldsymbol{X} - y\boldsymbol{p}_1^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Using the fact that the cross product should be zero

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

$$\begin{bmatrix} y\mathbf{p}_3^\top \mathbf{X} - \mathbf{p}_2^\top \mathbf{X} \\ \mathbf{p}_1^\top \mathbf{X} - x\mathbf{p}_3^\top \mathbf{X} \\ x\mathbf{p}_2^\top \mathbf{X} - y\mathbf{p}_1^\top \mathbf{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Third line is a linear combination of the first and second lines.
(x times the first line plus y times the second line)

One 2D to 3D point correspondence give you ▊ equations

Using the fact that the cross product should be zero

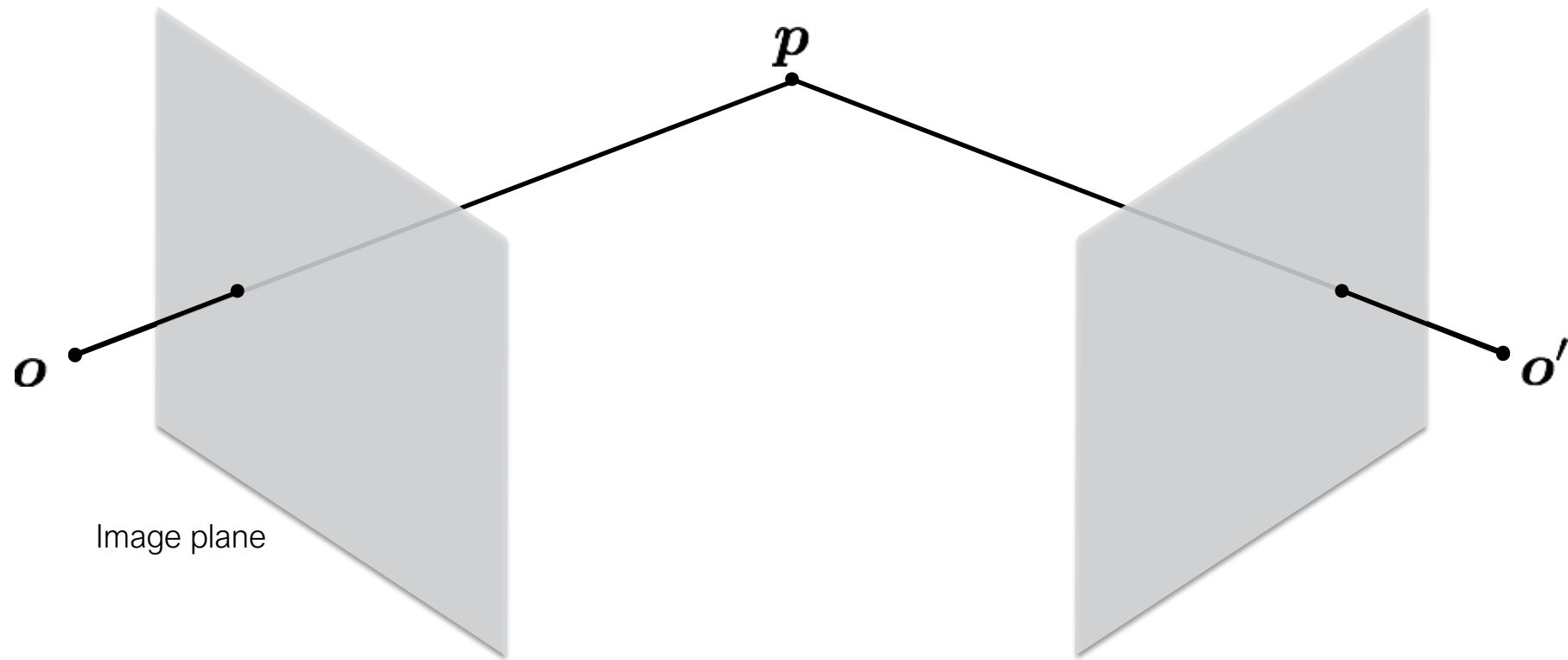$$\mathbf{x} \times \mathbf{PX} = \mathbf{0}$$

$$\begin{bmatrix} y\mathbf{p}_3^\top X - \mathbf{p}_2^\top X \\ \mathbf{p}_1^\top X - x\mathbf{p}_3^\top X \\ x\mathbf{p}_2^\top X - y\mathbf{p}_1^\top X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Third line is a linear combination of the first and second lines.
(x times the first line plus y times the second line)

One 2D to 3D point correspondence give you 2 equations

$$\begin{bmatrix} y\boldsymbol{p}_3^\top X - \boldsymbol{p}_2^\top X \\ \boldsymbol{p}_1^\top X - x\boldsymbol{p}_3^\top X \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Remove third row, and rear range as system on unkno wns

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

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

Now we can make a system of linear equations
(two lines for each 2D point correspondence)

Electronics Engineering, CBNU

Concatenate the 2D points from both images

Two rows from camera one

Two rows from camera two

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p'}_3^\top - \boldsymbol{p'}_2^\top \\ \boldsymbol{p'}_1^\top - x'\boldsymbol{p'}_3^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

sanity check! dimensions?

$$\mathbf{AX} = \mathbf{0}$$

How do we solve homogeneous linear system?

Concatenate the 2D points from both images

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p'}_3^\top - \boldsymbol{p'}_2^\top \\ \boldsymbol{p'}_1^\top - x'\boldsymbol{p'}_3^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{AX} = \mathbf{0}$$

How do we solve homogeneous linear system?

S V D !

Image plane

Electronics Engineering, CBNU

Baseline

Image plane

Epipolar geometry

p

o

e

Baseline

e'

o'

Image plane

Epipole
(projection of o' on the image plane)

Electronics Engineering, CBNU

Epipolar geometry

p

Epipolar plane

o e

Baseline

e' o'

Image plane

Epipole
(projection of o' on the image plane)

Epipolar geometry

Epipolar line
(intersection of Epipolar plan
e and image plane)

p

Epipolar plane

l

l'

o

e

e'

o'

Baseline

Image plane

Epipole
(projection of o' on the image plane)
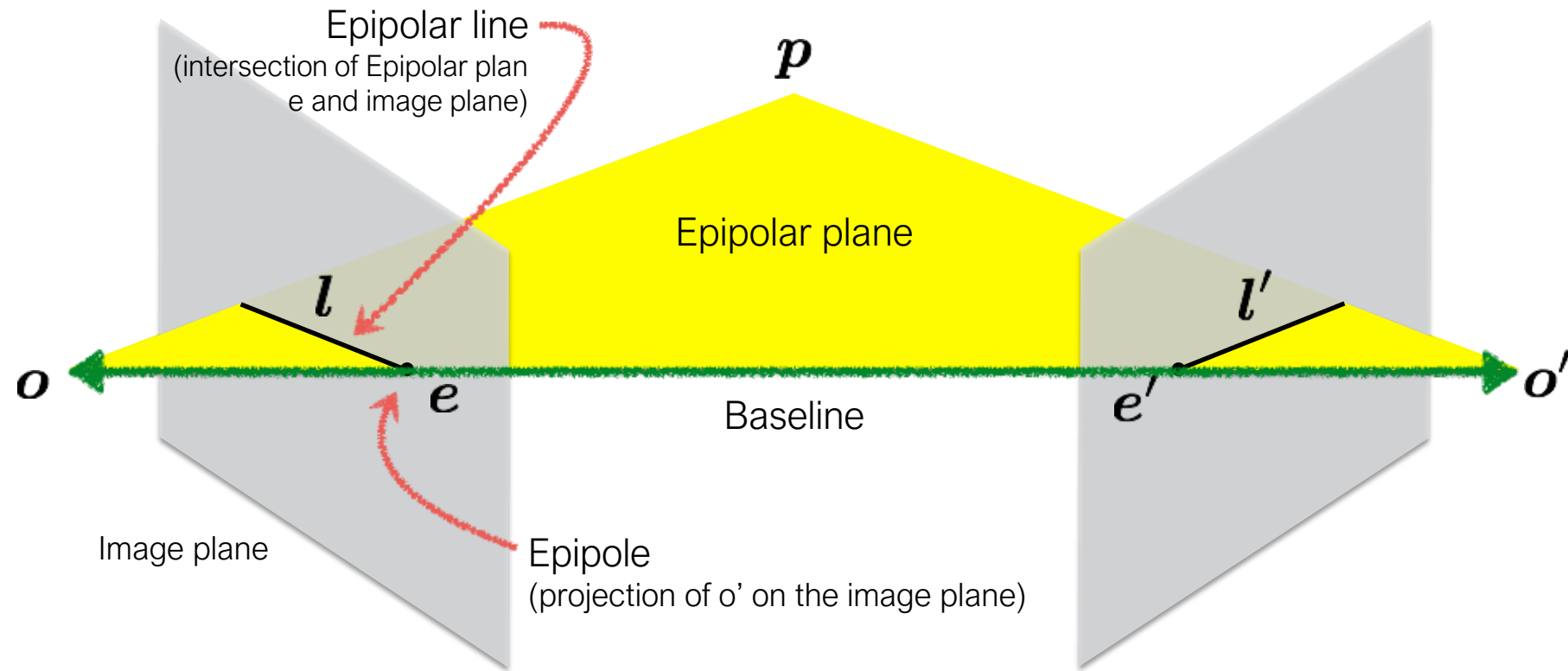
Backproject x to a r
ay in 3D

Epipolar line
(intersection of Epipolar plan
e and image plane)

x

o

e

l'

e'

o'
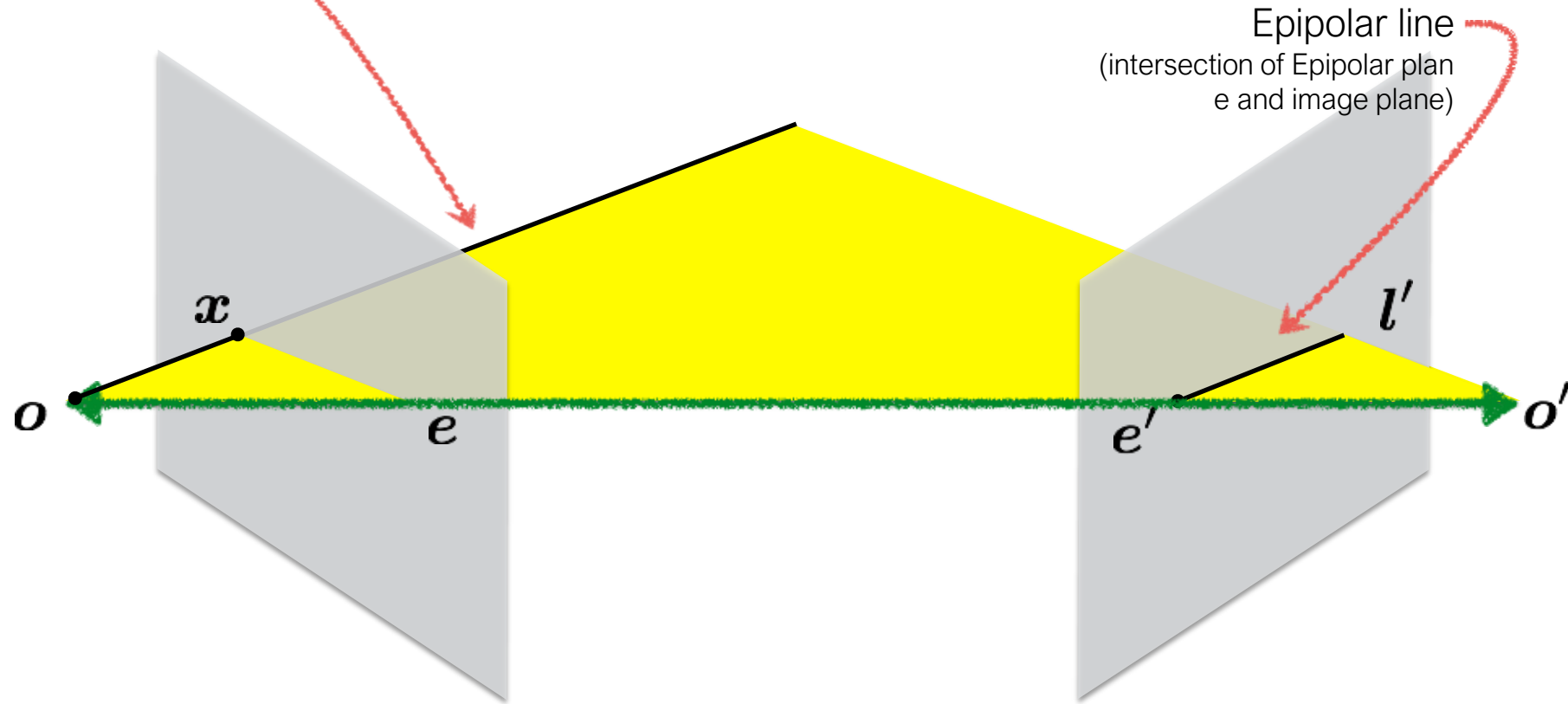
Another way to construct the epipolar plane, this time given x
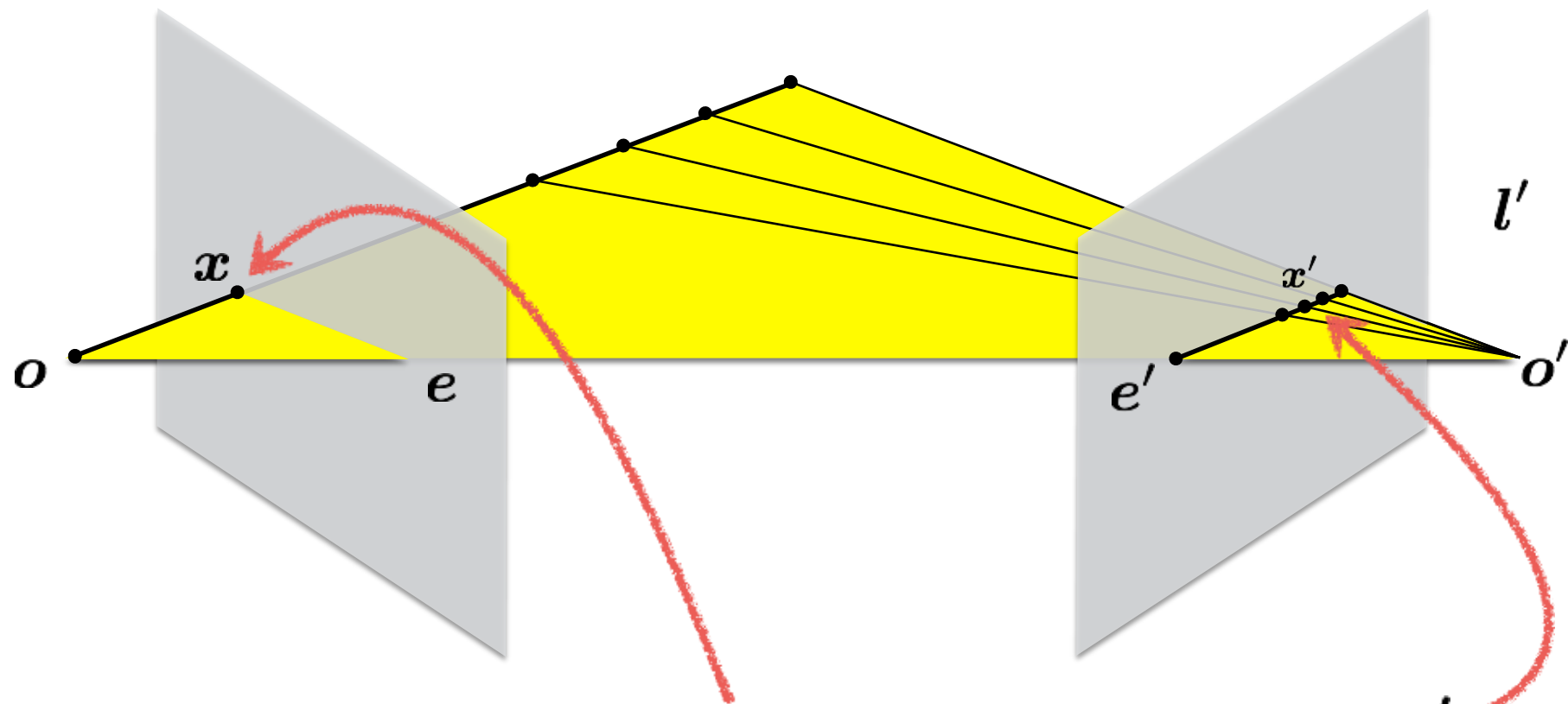
Potential matches for x lie on the epipolar line l'

Converging cameras
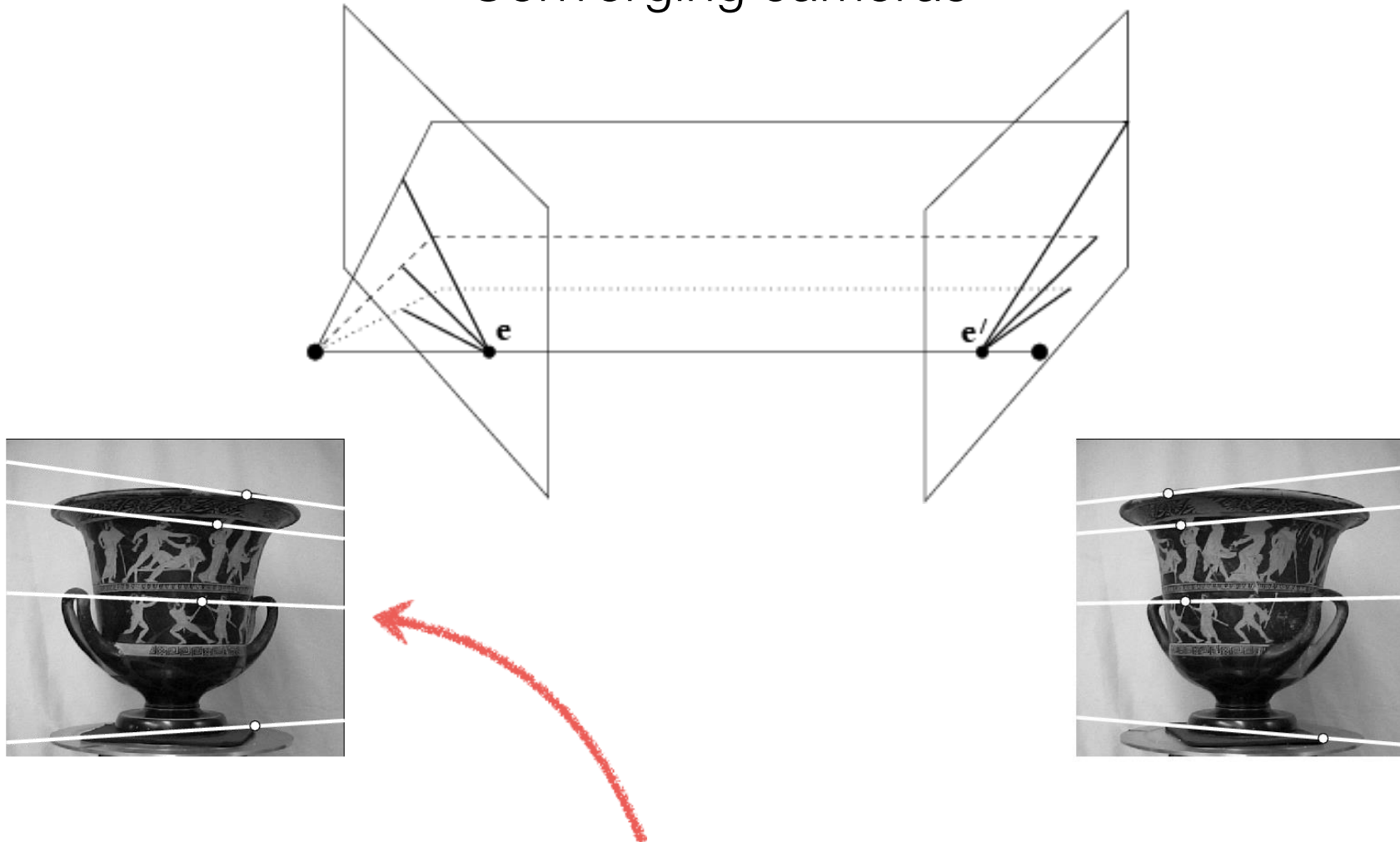
Where is the epipole in this image?

Electronics Engineering, CBNU

Converging cameras
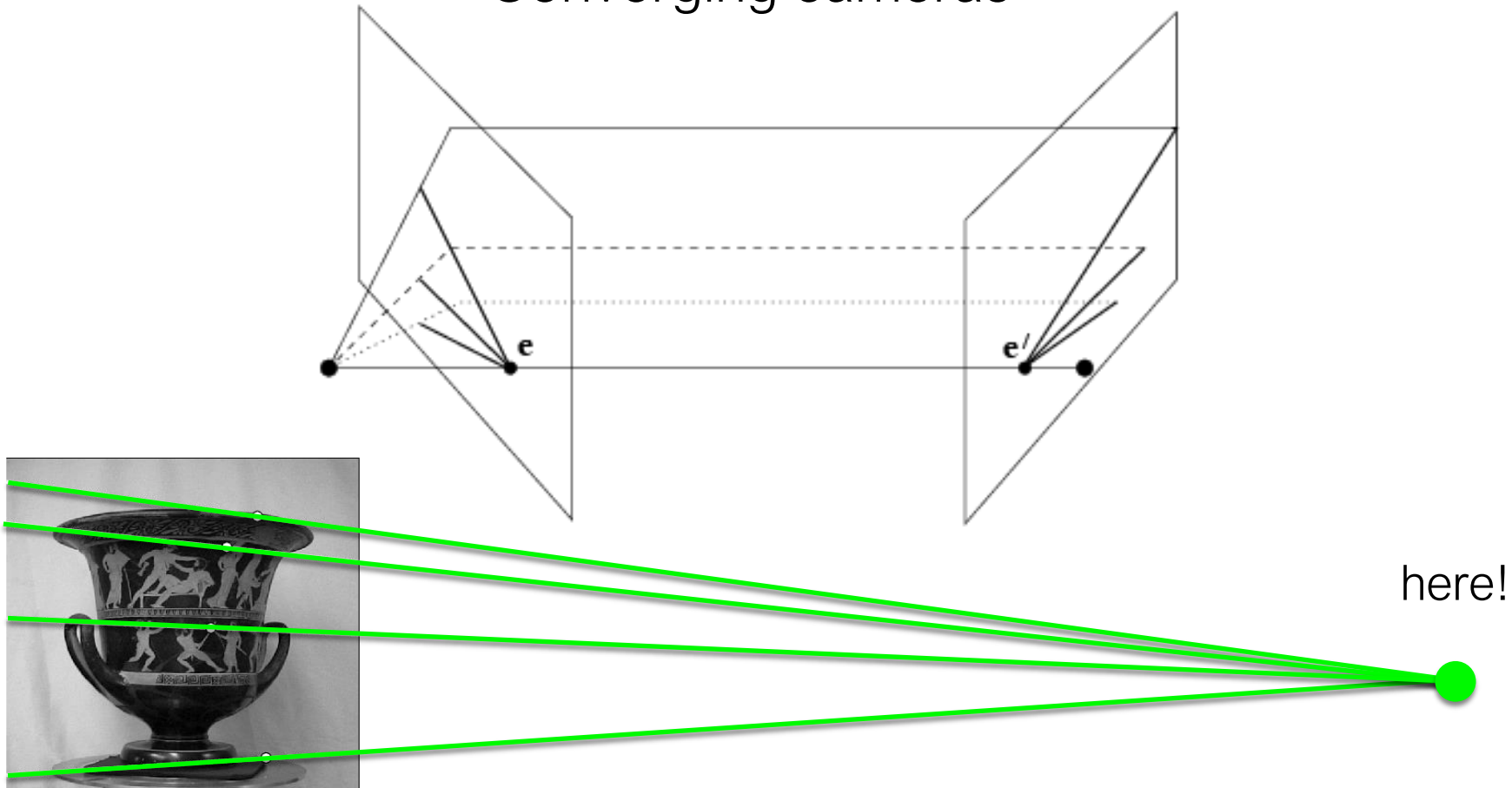
here!

Where is the epipole in this image?   It's not always in the image
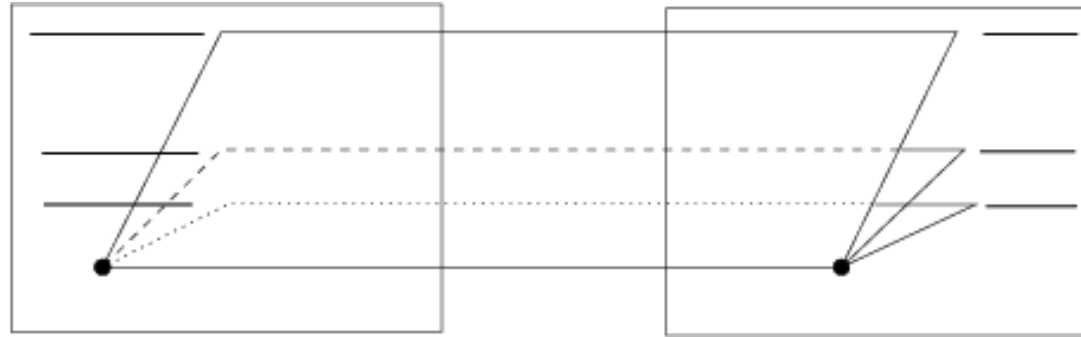
Parallel cameras

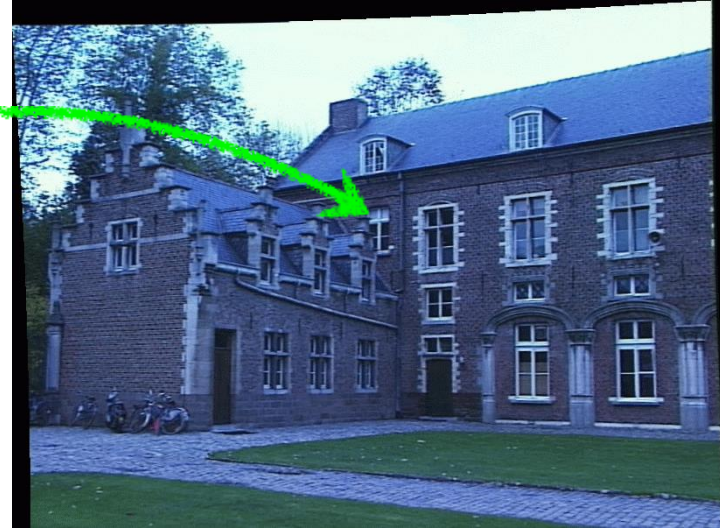

Where is the epipole?

Parallel cameras
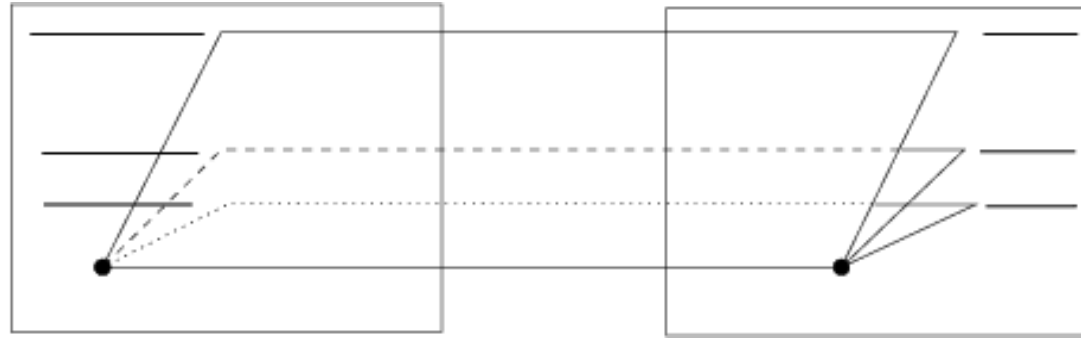
epipole at infinity

The epipolar constraint is an important concept for stereo vision

**Task:** Match point in left image to point in right image
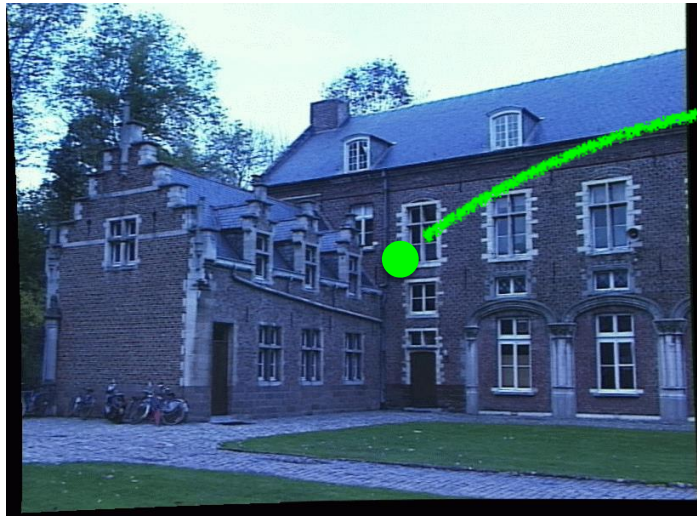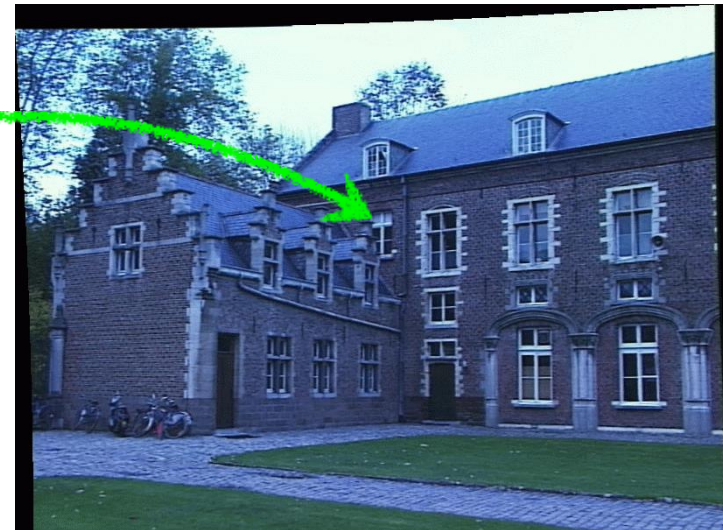
Left image   Right image

How would you do it?

Electronics Engineering, CBNU
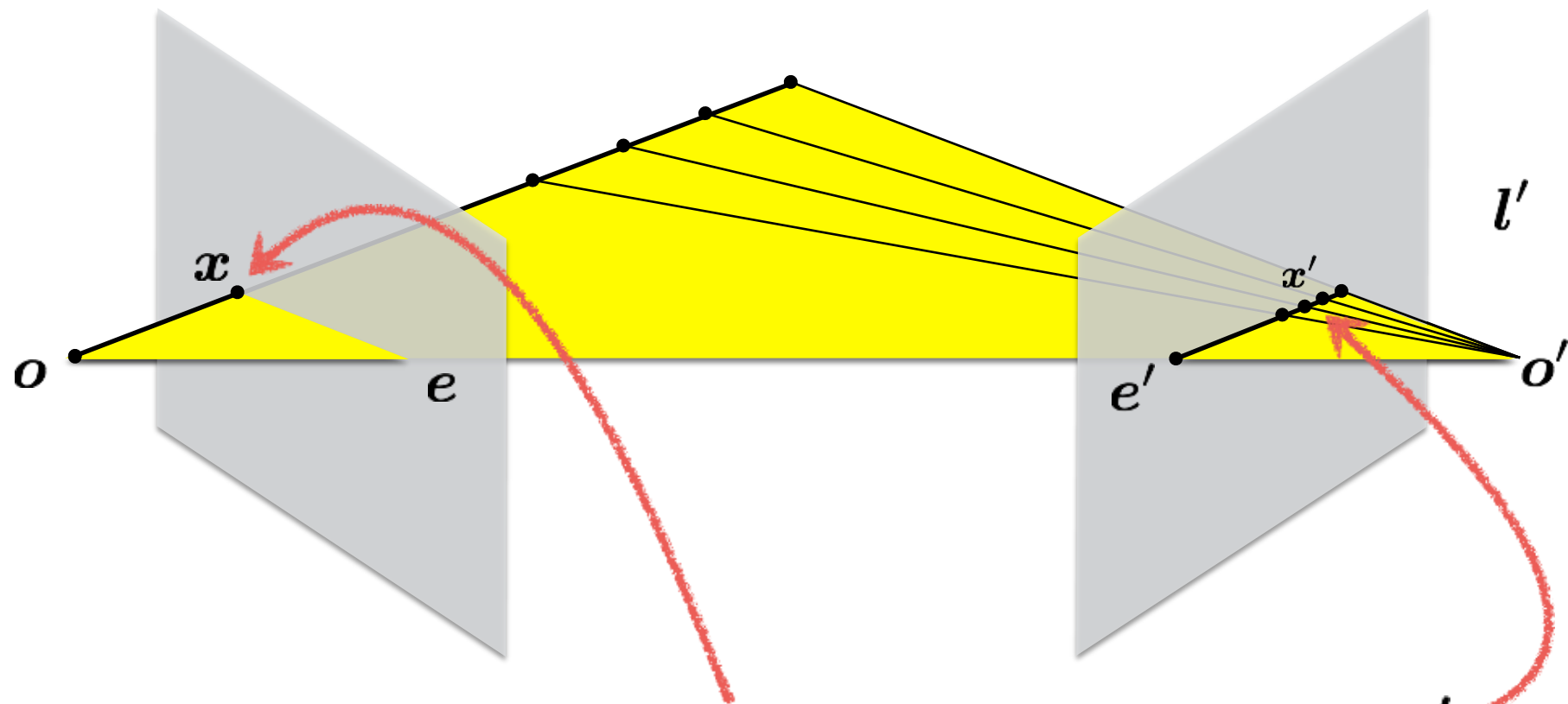
Potential matches for x lie on the epipolar line l'

The epipolar constraint is an important concept for stereo vision

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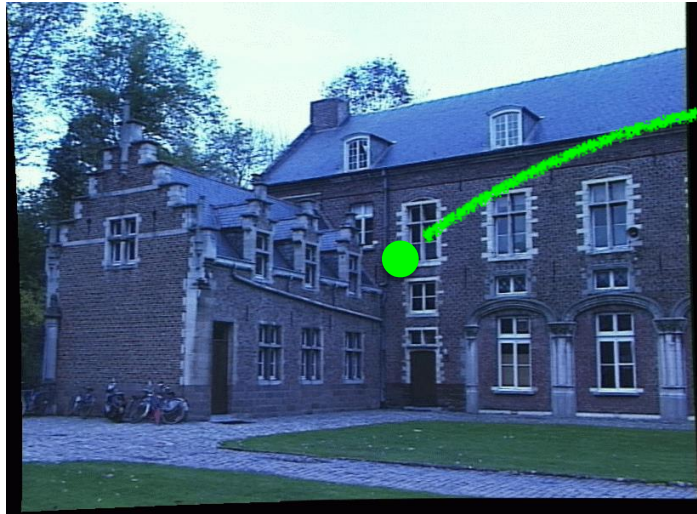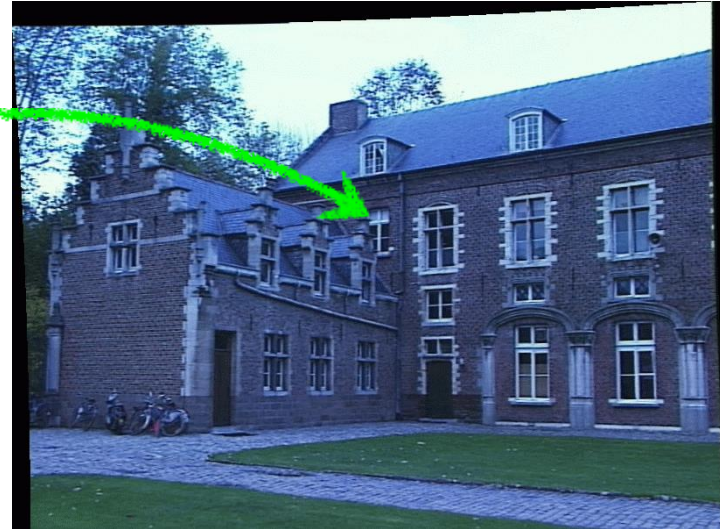
Left image   Right image

Want to avoid search over entire image

Epipolar constraint reduces search to a single line

The epipolar constraint is an important concept for stereo vision

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

Left image   Right image

Want to avoid search over entire image

Epipolar constraint reduces search to a single line

How do you compute the epipolar line?
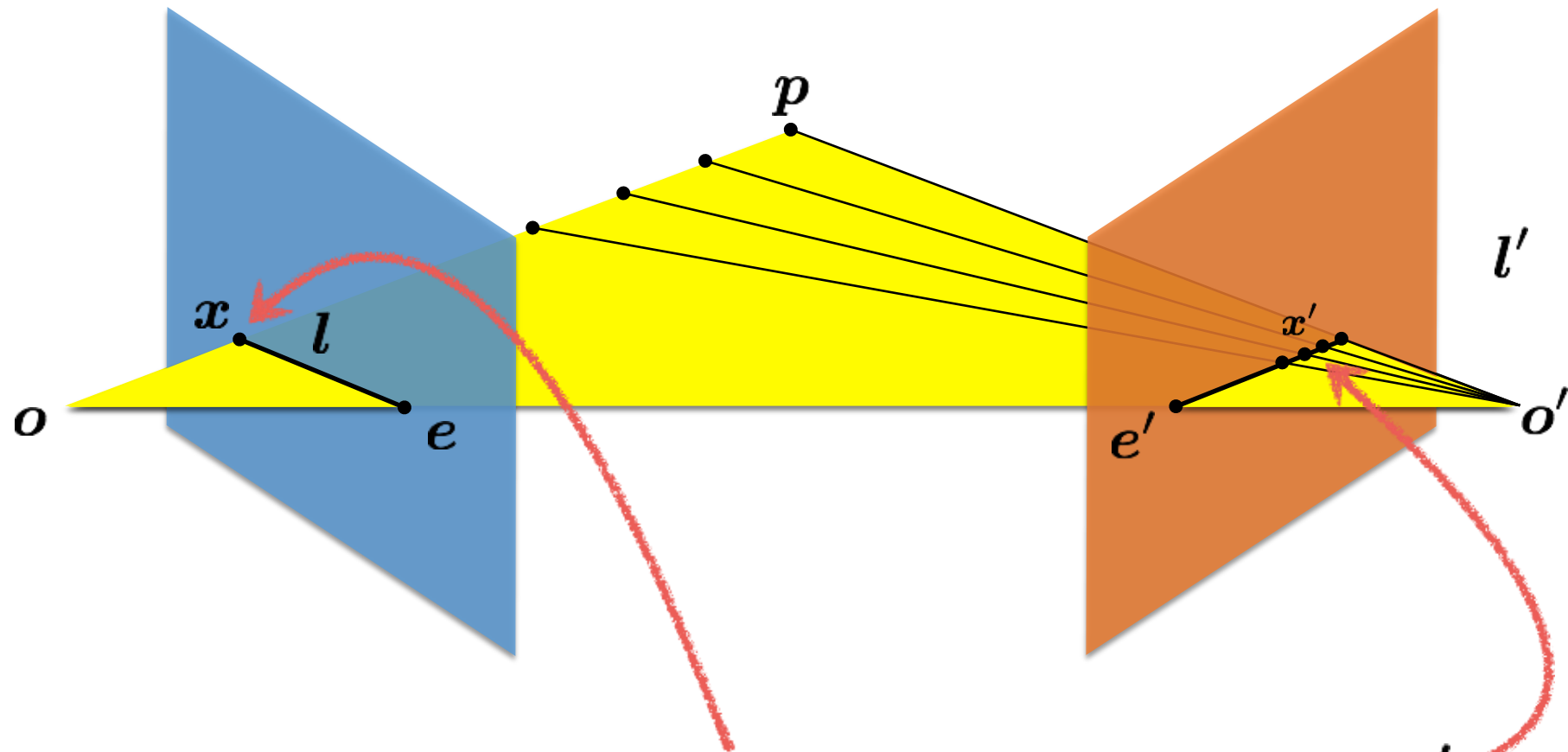
Potential matches for x lie on the epipolar line l'

Given a point in one image,
multiplying by the **essential matrix** will tell us
the **epipolar line** in the second view.

$$\boxed{\mathbf{E}x = l'}$$
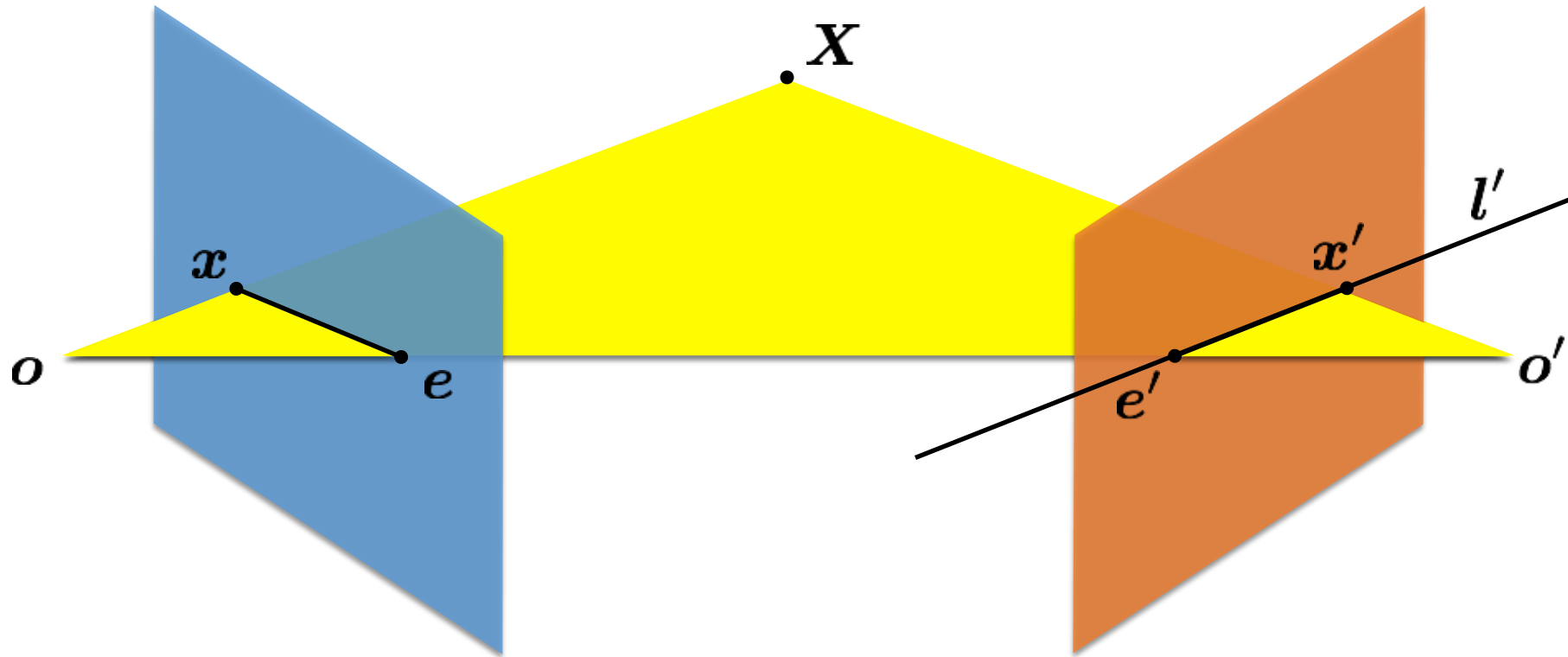
The Essential Matrix is a 3 x 3 matrix that en codes **epipolar geometry**

Given a point in one image,
multiplying by the **essential matrix** will tell us
the **epipolar line** in the second image.

So if $x'^{\top} l' = 0$ and $\mathbf{E} x = l'$ then

$$\boxed{x'^{\top} \mathbf{E} x = 0}$$
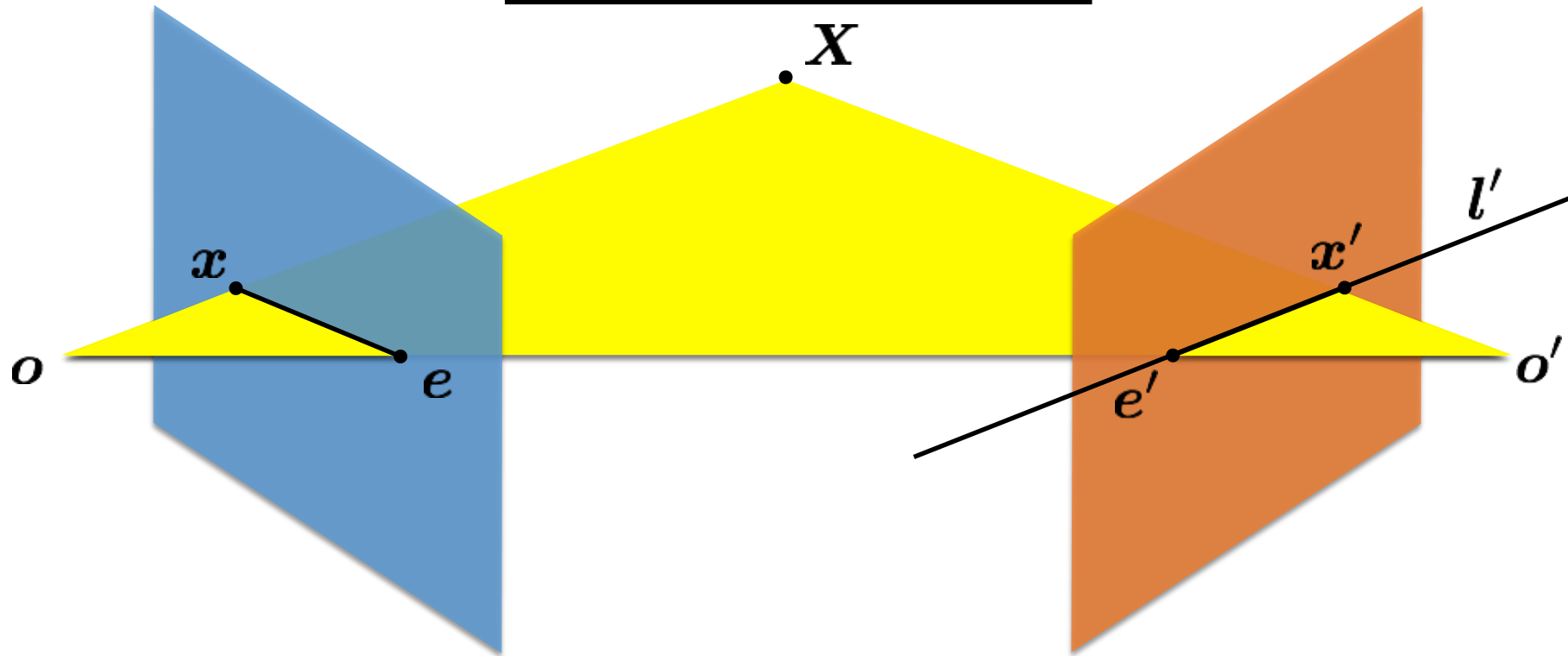
Essential Matrix vs Homography

What's the difference between the essential matrix and a homography?

They are both 3 x 3 matrices but …

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

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

Essential matrix maps a
point to a **line**

Homography maps a
point to a **point**

Electronics Engineering, CBNU

Longuet-Higgins equation $\qquad x'^{\top} \mathbf{E} x = 0$

(2D points expressed in <u>camera</u> coordinate system)

properties of the E matrix

| Longuet-Higgins equation | $x'^{\top} \mathbf{E} x = 0$ |

Epipolar lines

$$x^{\top} l = 0 \qquad\qquad x'^{\top} l' = 0$$

$$l' = \mathbf{E} x \qquad\qquad l = \mathbf{E}^{T} x'$$

(2D points expressed in <u>camera</u> coordinate system)

Longuet-Higgins equation	$x'^\top \mathbf{E} x = 0$	

Epipolar lines	$x^\top l = 0$ $l' = \mathbf{E} x$	$x'^\top l' = 0$ $l = \mathbf{E}^T x'$

Epipoles	$e'^\top \mathbf{E} = 0$	$\mathbf{E} e = 0$

(2D points expressed in camera coordinate system)

Given a point in one image,
multiplying by the **essential matrix** will tell us
the **epipolar line** in the second view.

$$\mathbf{E}\boldsymbol{x} = \boldsymbol{l}'$$
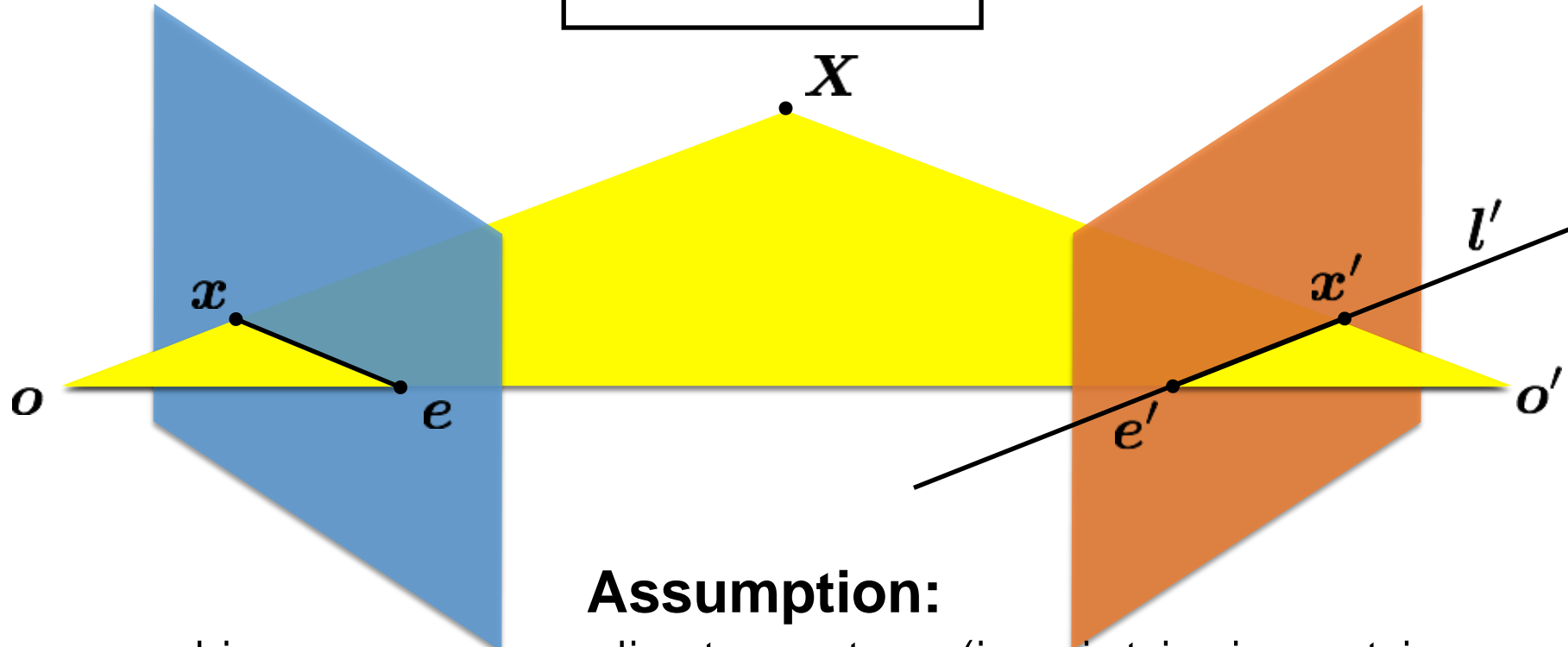
Assumption:
2D points expressed in camera coordinate system (i.e., intrinsic matrices are identities)

How do you generalize to non-identity intrinsic matrices?

The
fundamental matrix
is a
generalization
of the
essential matrix,
where the assumption of
Identity matrices
is removed

$$\hat{x}'^{\top} \mathbf{E} \hat{x} = 0$$

The essential matrix operates on image points expressed in **2D coordinates** expressed in the camera coordinate system

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

camera point   image point

$$\hat{x}'^{\top} \mathbf{E} \hat{x} = 0$$

The essential matrix operates on image points expressed in
2D coordinates expressed in the camera coordinate system

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

camera
point

image
point

Writing out the epipolar constraint in terms of image coordinates

$$\mathbf{K}'^{-\top} \mathbf{E} \mathbf{K}^{-1} x = 0$$
$$x'^{\top} (\mathbf{K}'^{-\top} \mathbf{E} \mathbf{K}^{-1}) x = 0$$
$$x'^{\top} \mathbf{F} x = 0$$

Same equation works in image coordinates!

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

it maps pixels to epipolar lines

Electronics Engineering, CBNU

Longuet-Higgins equation $\qquad x'^{\top}\mathbb{E}x = 0$

Epipolar lines

$$x^{\top}l = 0 \qquad\qquad x'^{\top}l' = 0$$

$$l' = \mathbb{E}x \qquad\qquad l = \mathbb{E}^T x'$$

Epipoles $\qquad e'^{\top}\mathbb{E} = 0 \qquad\qquad \mathbb{E}e = 0$

(points in **image** coordinates)

Assume you have *M* matched *image* points

$$\{\boldsymbol{x}_m, \boldsymbol{x}'_m\} \qquad m = 1, \ldots, M$$

Each correspondence should satisfy

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

*How would you solve for the 3 x 3 **F** matrix?*

Set up a homogeneous linear system with 9 unknowns

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

$$\begin{bmatrix} x_m' & y_m' & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = 0$$

How many equation do you get from one correspondence?

Electronics Engineering, CBNU

$$\begin{bmatrix} x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = 0$$

ONE correspondence gives you ONE equation

$$x_m x'_m f_1 + x_m y'_m f_2 + x_m f_3 +$$
$$y_m x'_m f_4 + y_m y'_m f_5 + y_m f_6 +$$
$$x'_m f_7 + y'_m f_8 + f_9 = 0$$

Electronics Engineering, CBNU

$$
\begin{bmatrix} x'_m & y'_m & 1 \end{bmatrix}
\begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}
\begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = 0
$$

Set up a homogeneous linear system with 9 unknowns

$$
\begin{bmatrix}
x_1 x'_1 & x_1 y'_1 & x_1 & y_1 x'_1 & y_1 y'_1 & y_1 & x'_1 & y'_1 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
x_M x'_M & x_M y'_M & x_M & y_M x'_M & y_M y'_M & y_M & x'_M & y'_M & 1
\end{bmatrix}
\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}
$$

How many equations do you need?

Each point pair (according to epipolar constraint)
contributes only one <u>scalar</u> equation

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

Note: This is different from the Homography estimation where
each point pair contributes 2 equations.

We need at least 8 points

Hence, the 8 point algorithm!

0. (Normalize points)

1. Construct the M x 9 matrix **A**

2. Find the SVD of **A**

3. Entries of **F** are the elements of column of **V**

   corresponding to the least singular value

4. (Enforce rank 2 constraint on F)

5. (Un-normalize F)
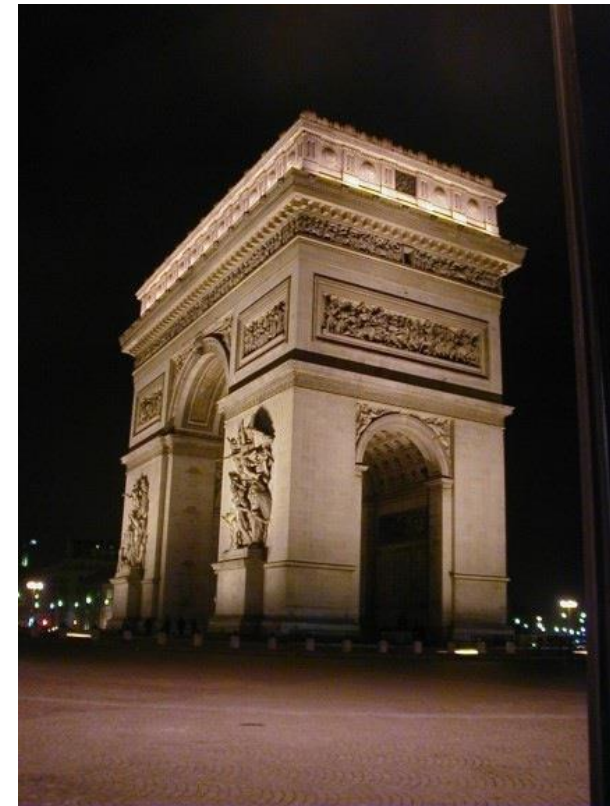
$$\mathbf{F} = \begin{bmatrix} -0.00310695 & -0.0025646 & 2.96584 \\ -0.028094 & -0.00771621 & 56.3813 \\ 13.1905 & -29.2007 & -9999.79 \end{bmatrix}$$
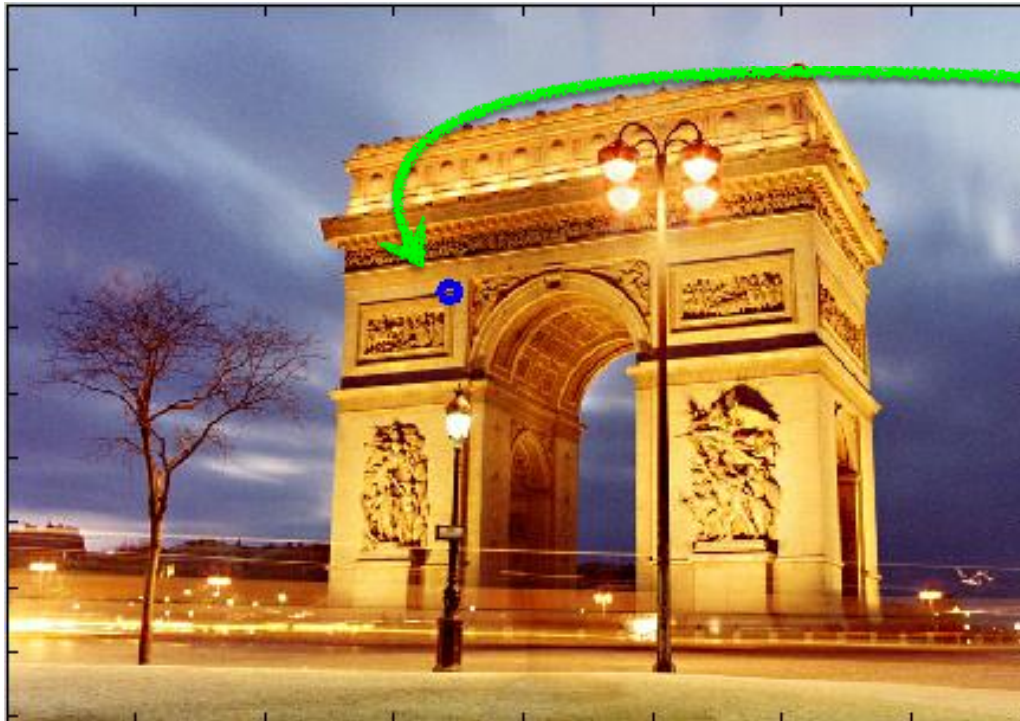
$$\boldsymbol{x} = \begin{bmatrix} 343.53 \\ 221.70 \\ 1.0 \end{bmatrix}$$

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

$$= \begin{bmatrix} 0.0295 \\ 0.9996 \\ -265.1531 \end{bmatrix}$$

Electronics Engineering, CBNU

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

$$= \begin{bmatrix} 0.0295 \\ 0.9996 \\ -265.1531 \end{bmatrix}$$

Electronics Engineering, CBNU
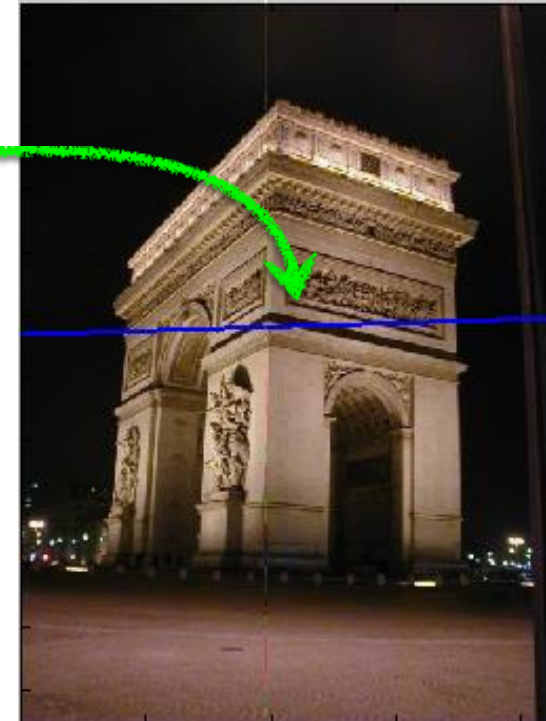
How would you compute it?

Electronics Engineering, CBNU

$$\mathbf{F}e = 0$$

The epipole is in the right null space of **F**

How would you solve for the epipole?

Electronics Engineering, CBNU

$$\mathbf{F}e = 0$$

The epipole is in the right null space of **F**

How would you solve for the epipole?

S V D !

Left image

Right image

Left image

Right image

1. Select point in one image (how?)

Electronics Engineering, CBNU

Left image

Right image

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)

Electronics Engineering, CBNU

Left image

Right image

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
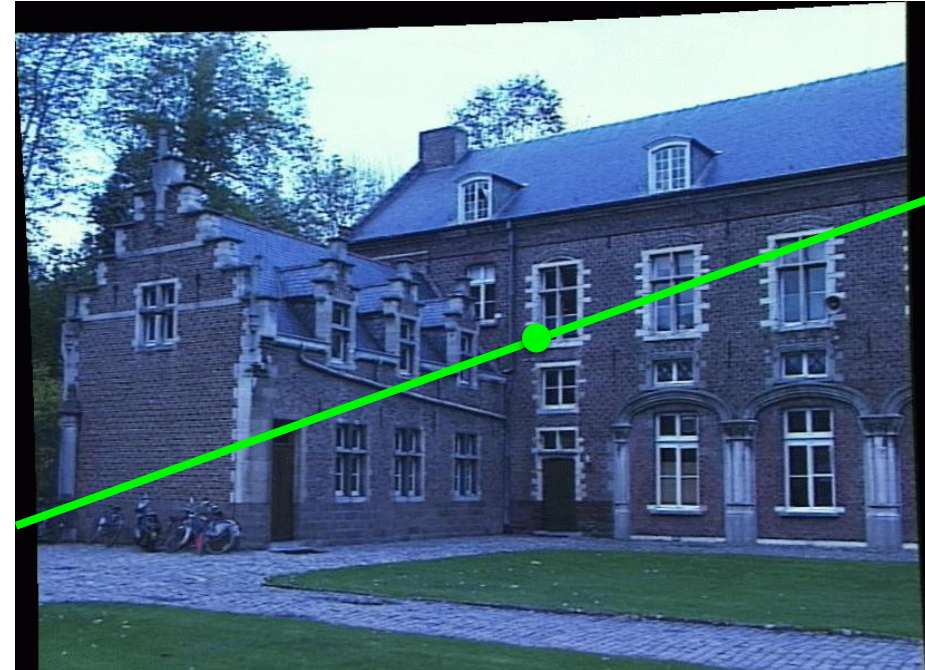3. Find matching point along line (how?)

Left image

Right image

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
3. Find matching point along line (how?)
4. Perform triangulation (how?)

3D point

left image

right image

x

x'

C

C'

left camera with matrix P

right camera with matrix P'

How would you reconstruct 3D points?

Left image

Right image

1. Select point in one image (how?)
2. Form epipolar line for that point in second image (how?)
3. Find matching point along line (how?)
4. Perform triangulation (how?)

What are the disadvantages of this procedure?

Electronics Engineering, CBNU

What's different between these two images?

Electronics Engineering, CBNU

Electronics Engineering, CBNU

Objects that are close move more or less?

Electronics Engineering, CBNU

The amount of horizontal movement is inversely proportional to …
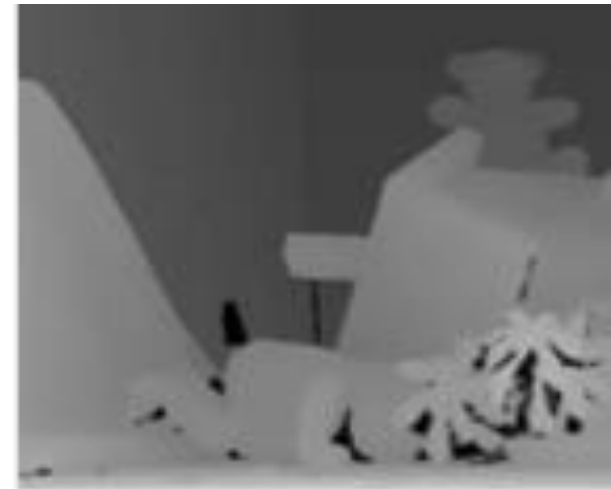
The amount of horizontal movement is inversely proportional to …
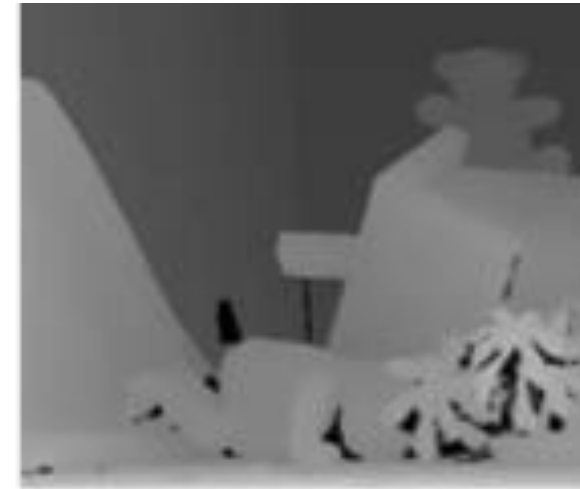
… the distance from the camera.
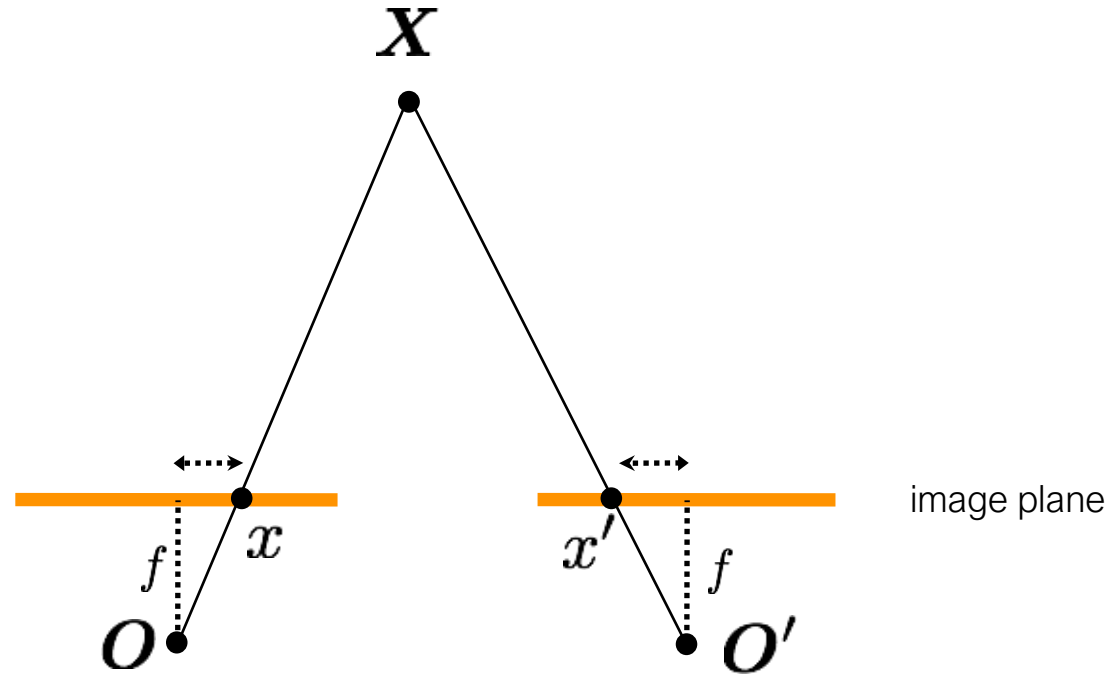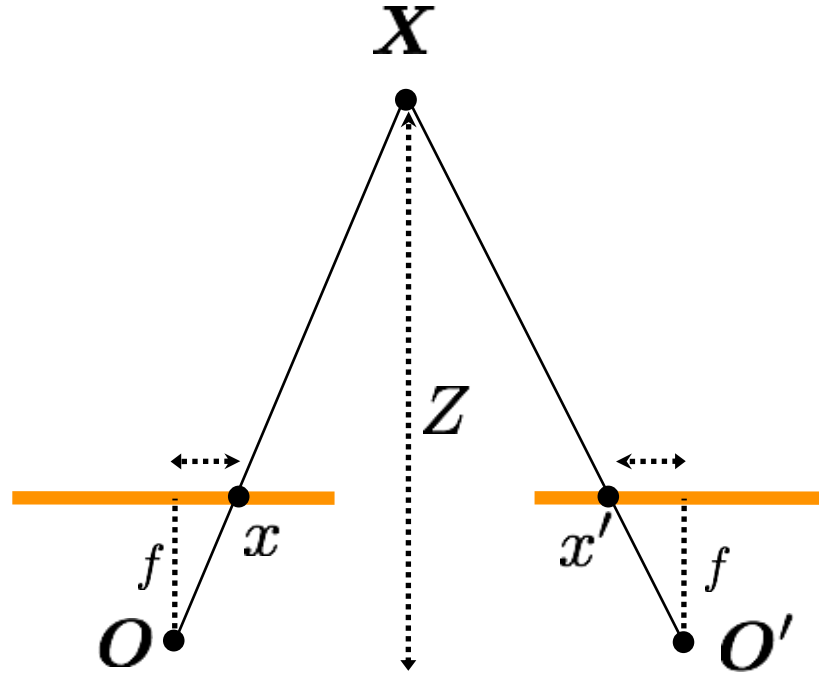
More formally…
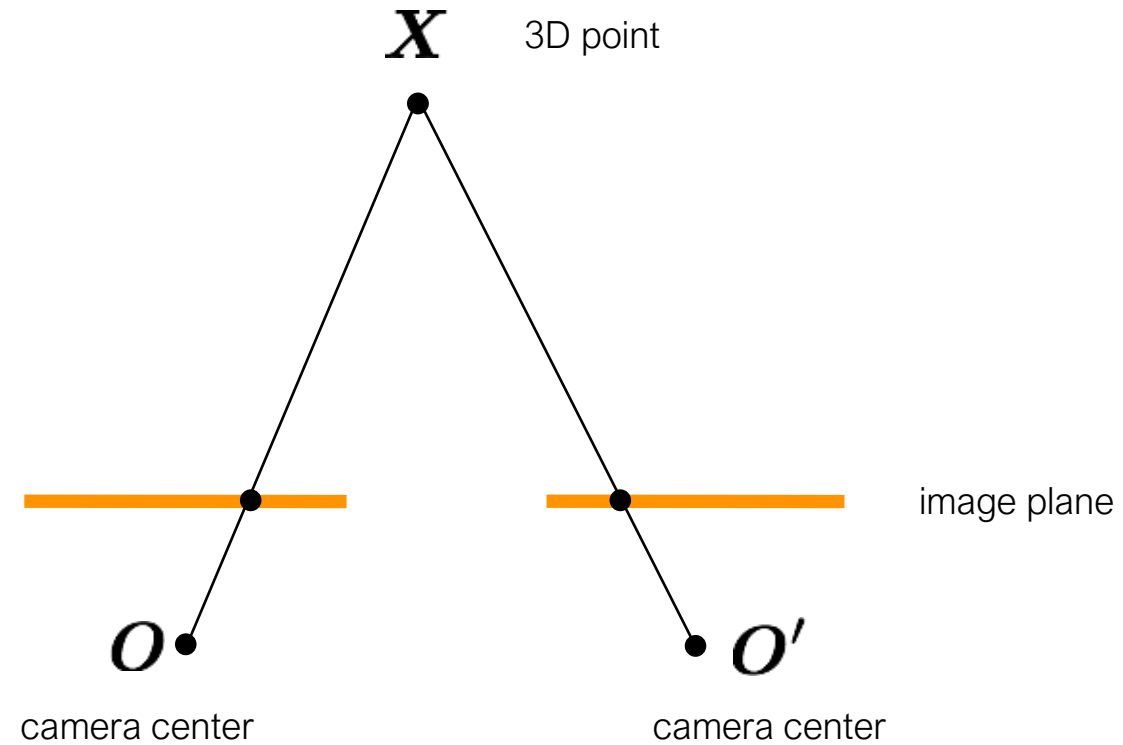
X 3D point

image plane

O

O'

camera center   camera center

X

x

f

O

x'

f

O'

image plane

(baseline)

How is X related to x?

$$\frac{X}{Z} = \frac{x}{f}$$

X

Z

x

x'

f

f

O

O'

(baseline)

b

$$\frac{X}{Z} = \frac{x}{f}$$

How is X related to x'?

$$\frac{X}{Z} = \frac{x}{f}$$

$$\frac{b - X}{Z} = \frac{x'}{f}$$

$$\frac{X}{Z} = \frac{x}{f}$$

$$X$$

$$Z$$

$$f \quad x$$

$$O \qquad O'$$

(baseline)

$$b$$

$$\frac{b - X}{Z} = \frac{x'}{f}$$

$$x' \quad f$$

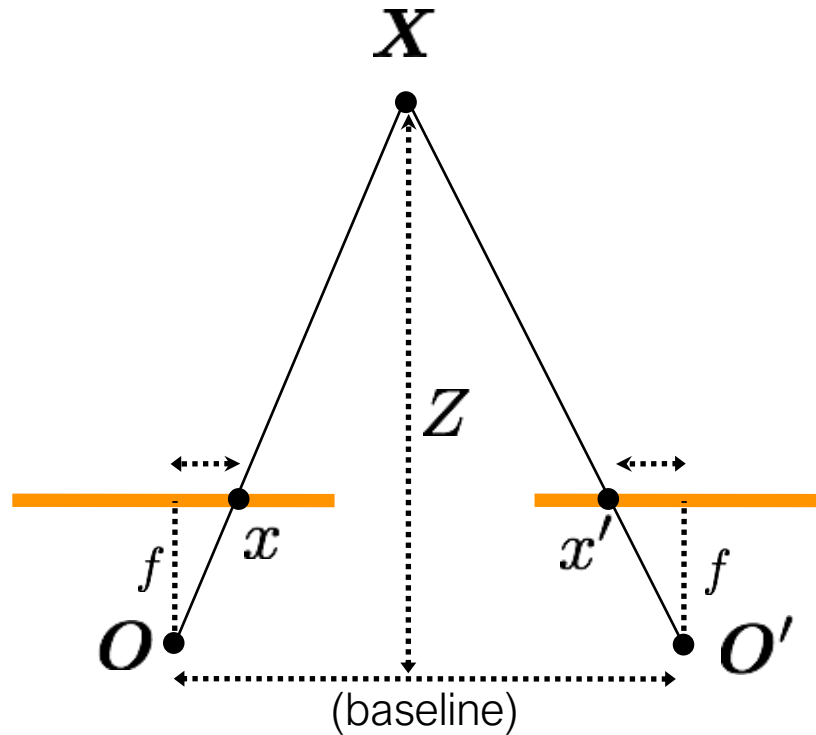## Disparity

$$d = x - x'$$ (wrt to camera origin of image plane)

$$= \frac{bf}{Z}$$
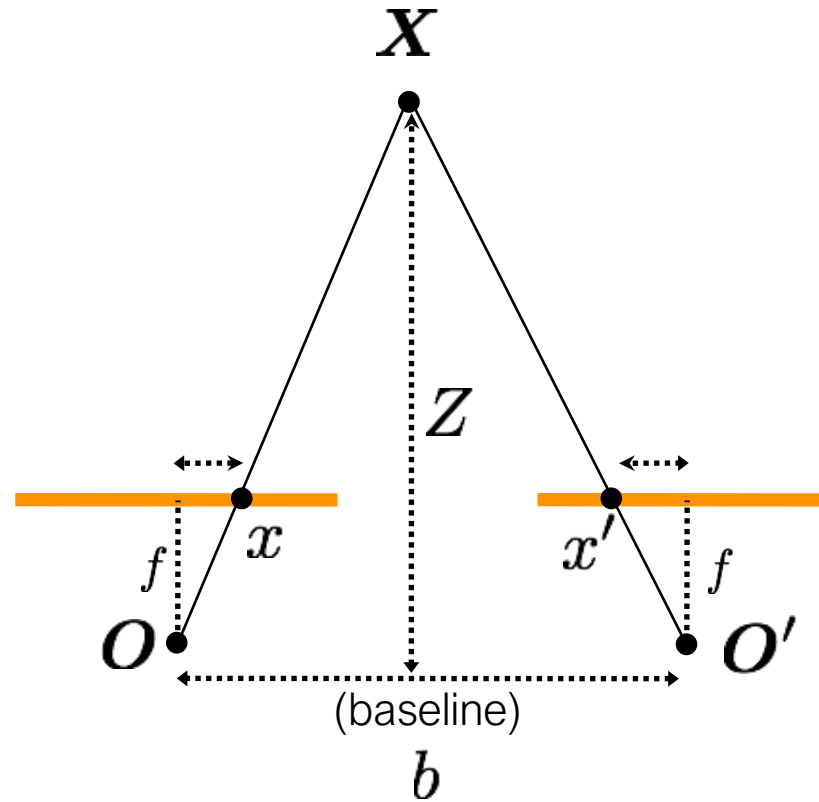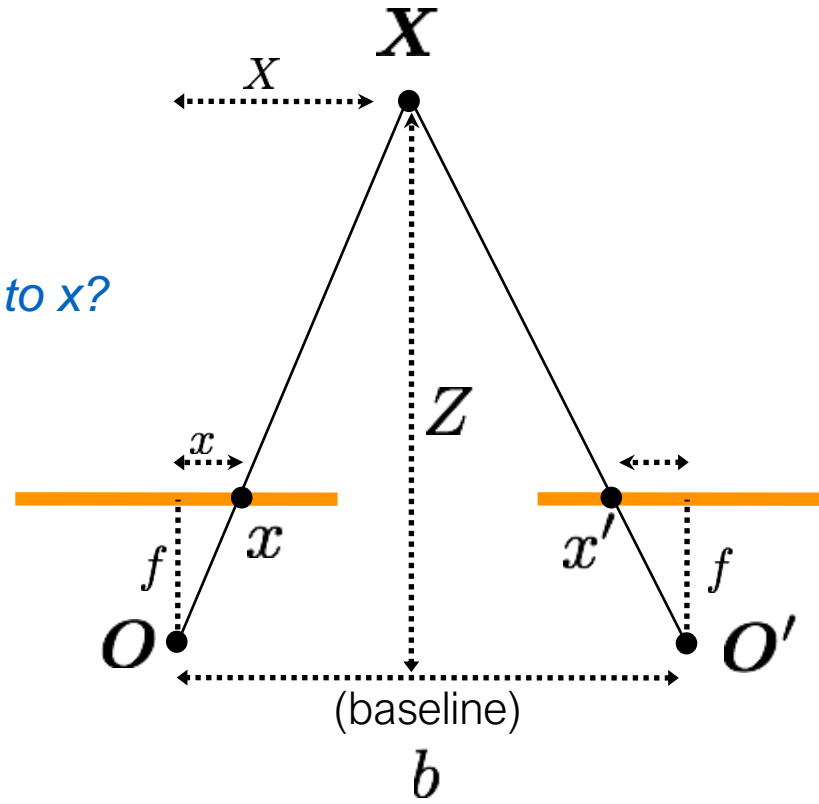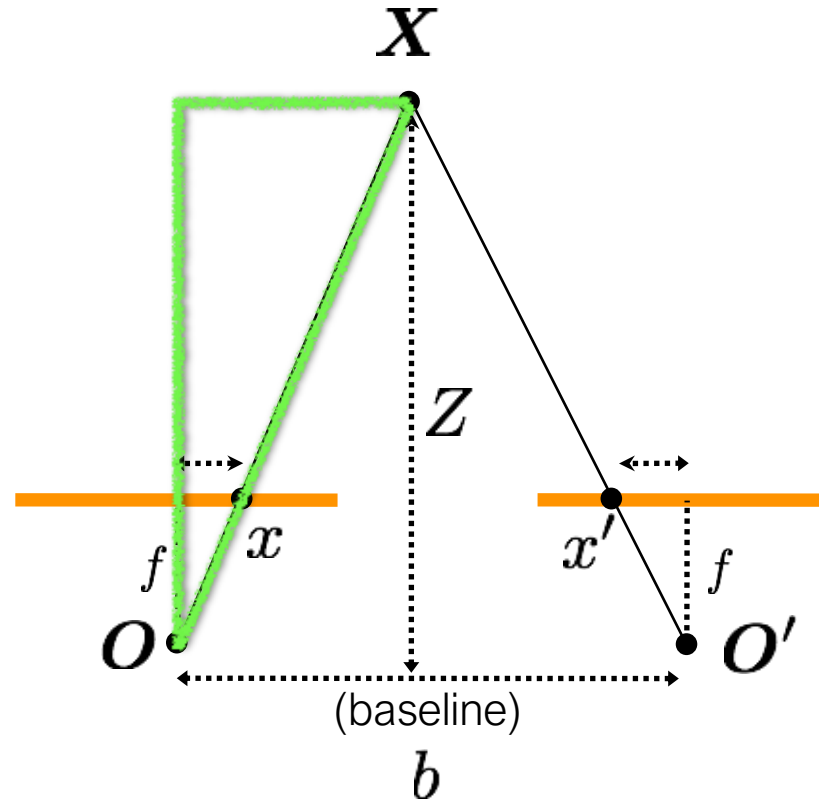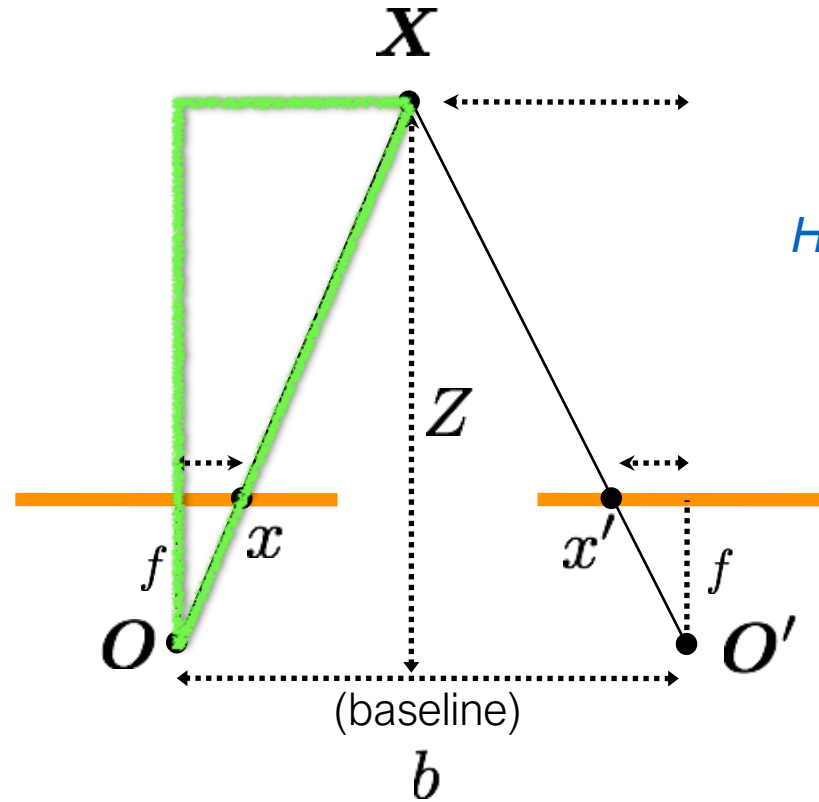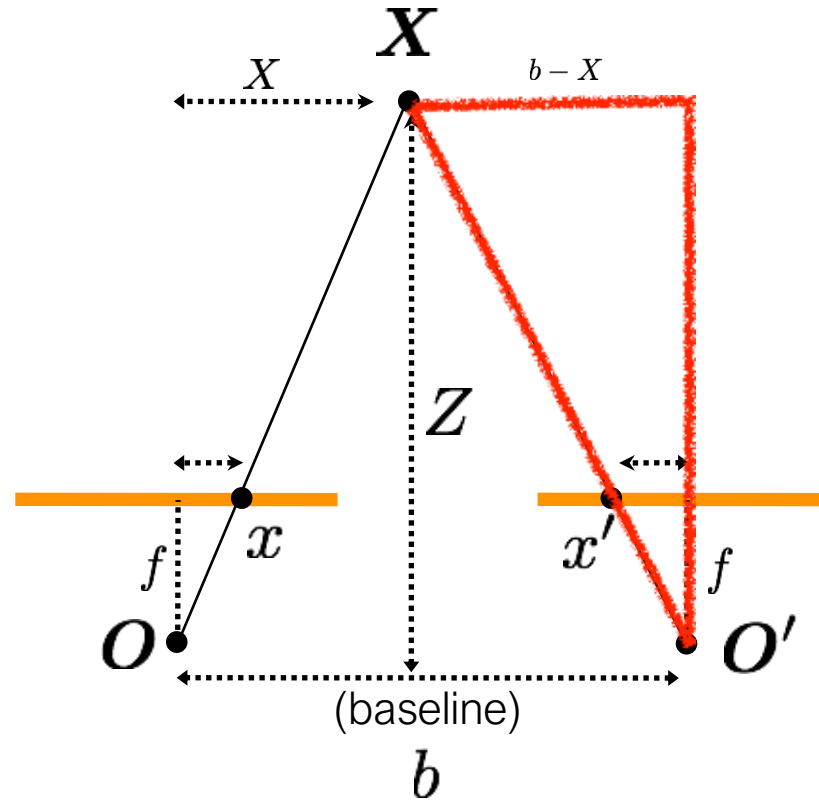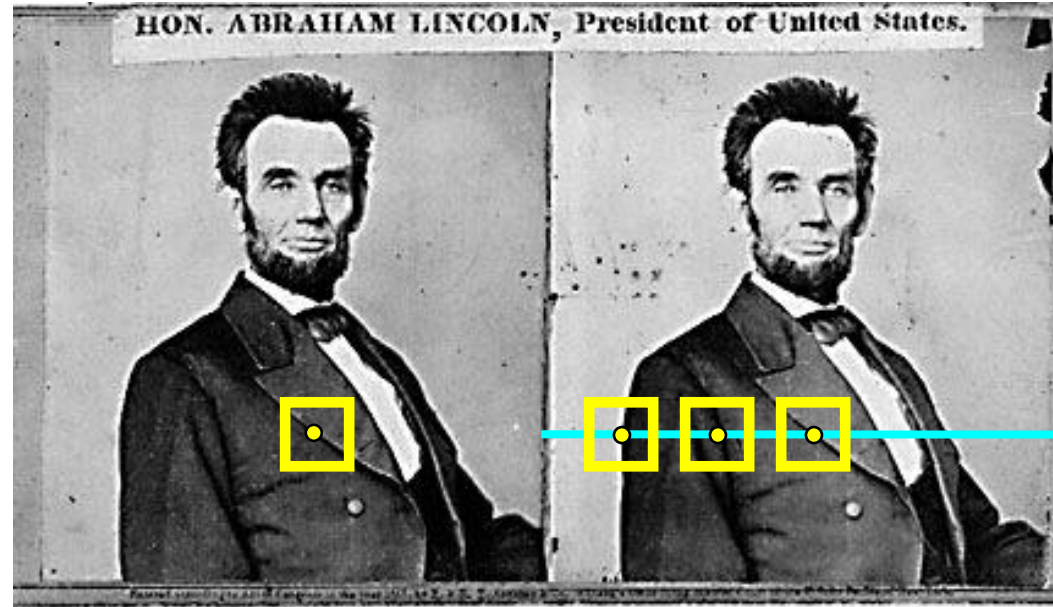
$$\frac{X}{Z} = \frac{x}{f}$$

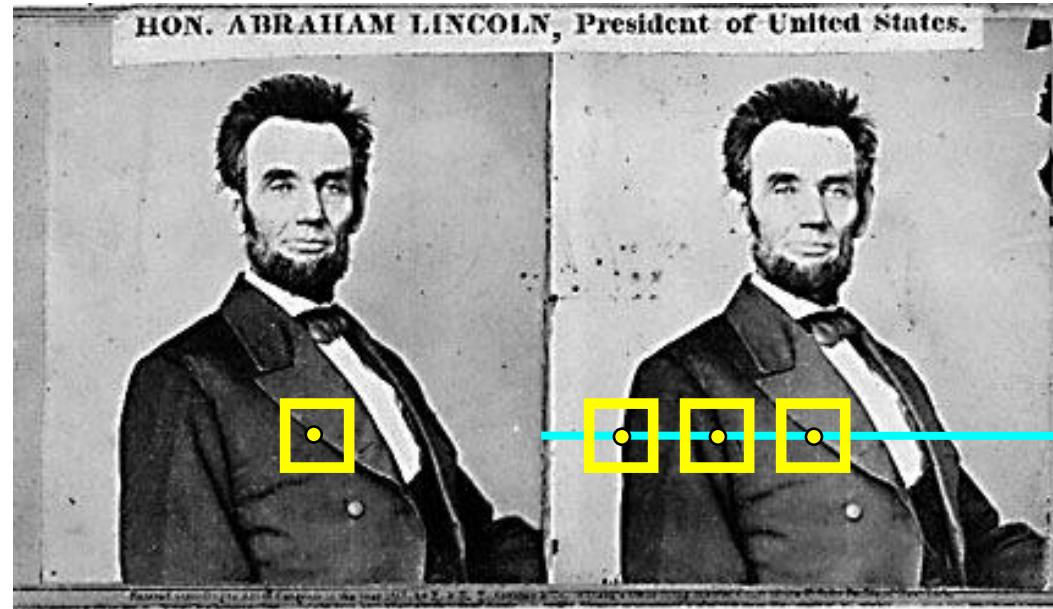$$\frac{b - X}{Z} = \frac{x'}{f}$$

Disparity

$$d = x - x'$$

$$= \frac{bf}{Z}$$

inversely proportional to depth

1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity $Z = \dfrac{bf}{d}$

How can you make the epipolar lines horizontal?

X 3D point

image plane

O O'

camera center   camera center
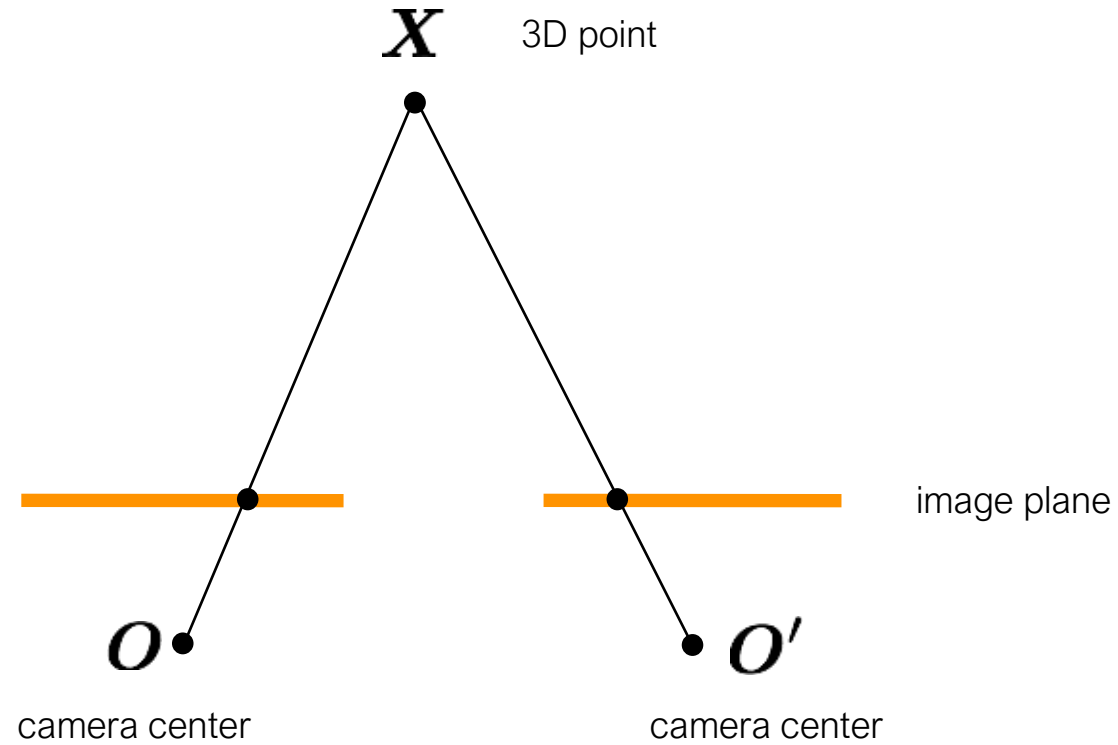
What's special about these two cameras?
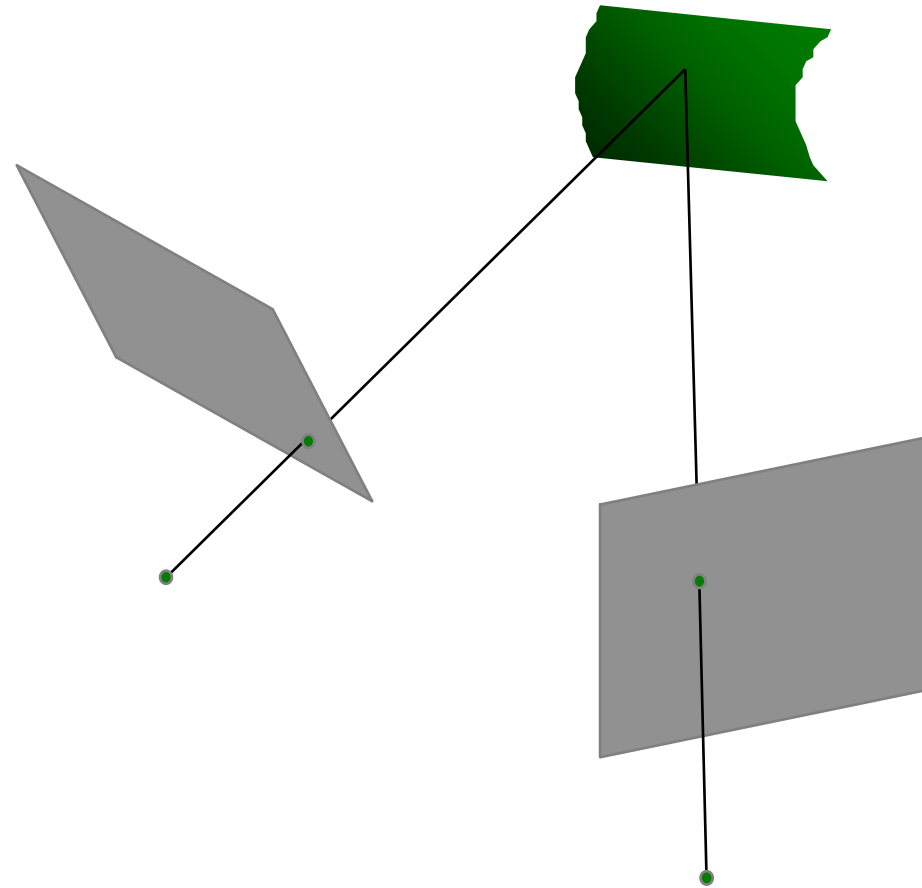
Electronics Engineering, CBNU

What is stereo rectification?

Electronics Engineering, CBNU

What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers
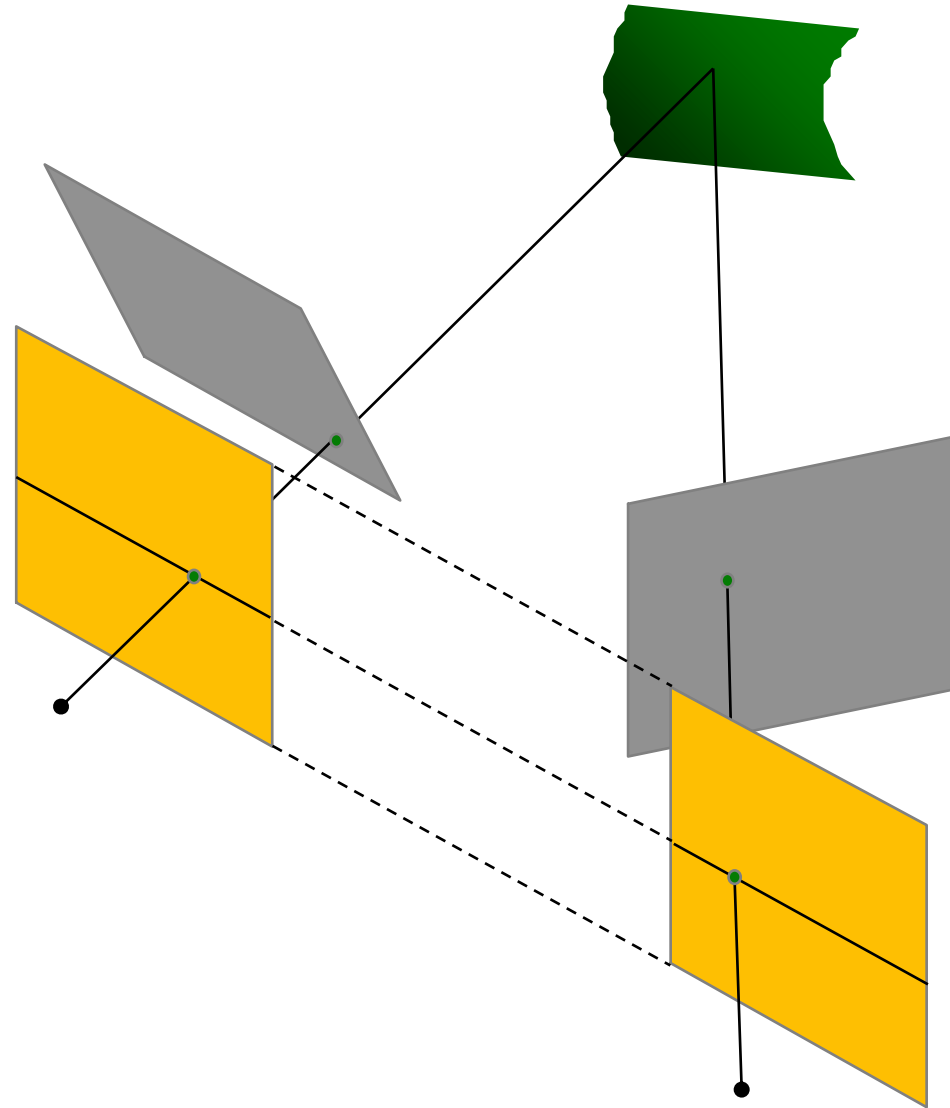
How can you do this?

What is stereo rectification?

Reproject image planes onto a common plane parallel to the line between camera centers

Need two homographies (3x3 transform), one for each input image reprojection

C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. Computer Vision and Pattern Recognition, 1999.

Electronics Engineering, CBNU
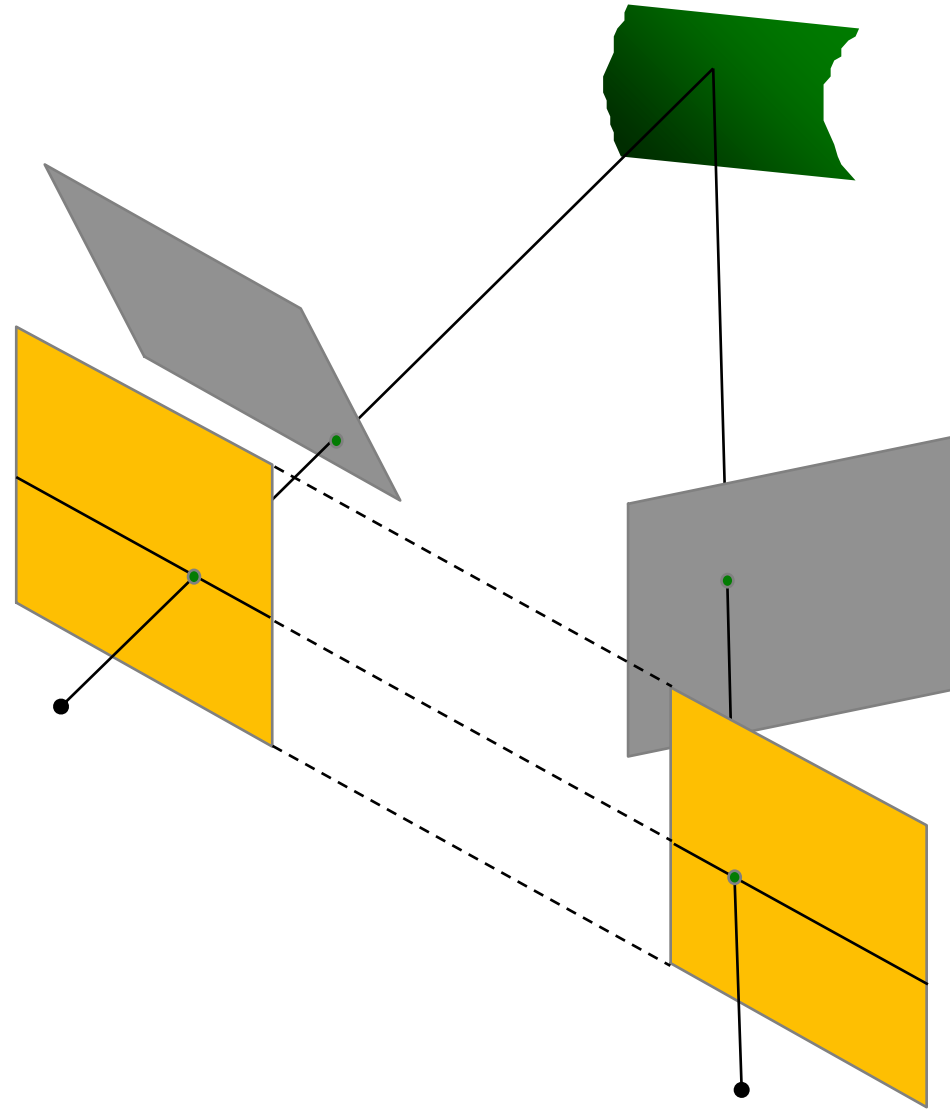
1. **Rotate** the right camera by **R**
   (aligns camera coordinate system orientation only)

2. Rotate (**rectify**) the left camera so that the epipole is at infinity

3. Rotate (**rectify**) the right camera so that the epipole is at infinity

4. Adjust the **scale**

Center of Projection
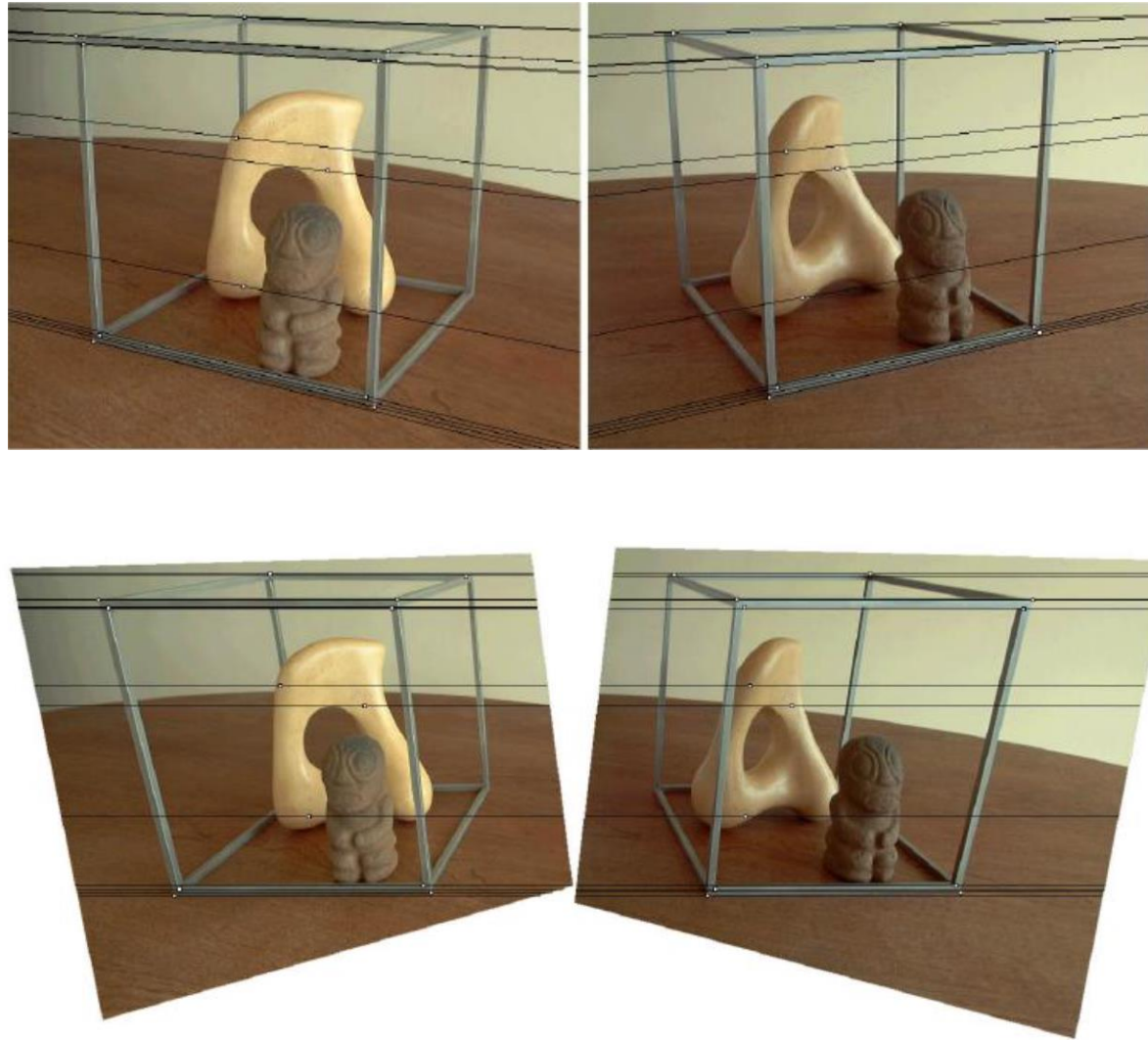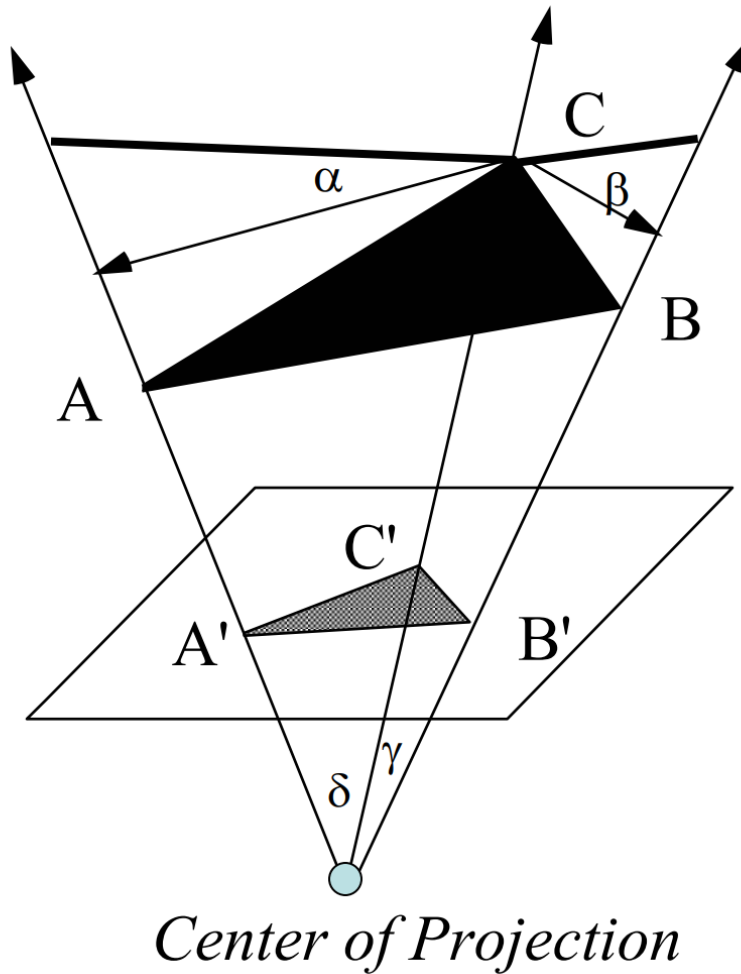
- If distance R_c to C is known, then possible locations of A (and B) can be computed
  - they lie on the intersections of the line of sight through A' and the sphere of radius AC centered at C
  - Once A and B are located, their distance can be computed and compared against the actual distance AB
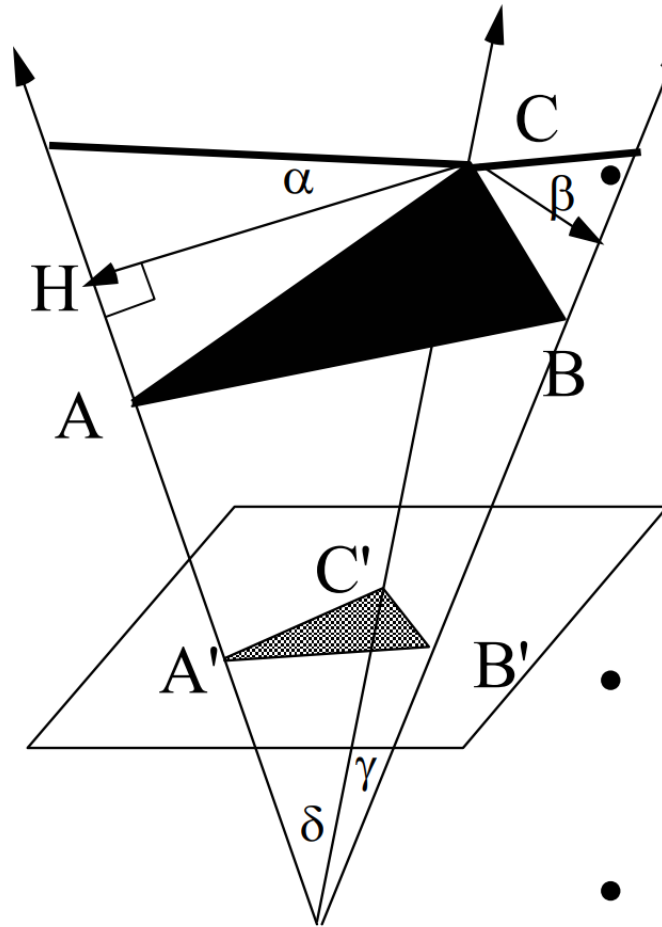
- Not practical to search on R_c since it is unbounded
- Instead, search on one angular pose parameter, α.
  - $R_c = AC \cos \alpha / \sin \delta$
  - $R_a = R_c \cos \delta \pm AC \sin \alpha$
  - $R_b = R_c \cos \gamma \pm [(BC^2 - (RC \sin \gamma)^2]^{1/2}$
- This results in four possible lengths for side AB
- Keep poses with the right AB length

Stereo Block Matching
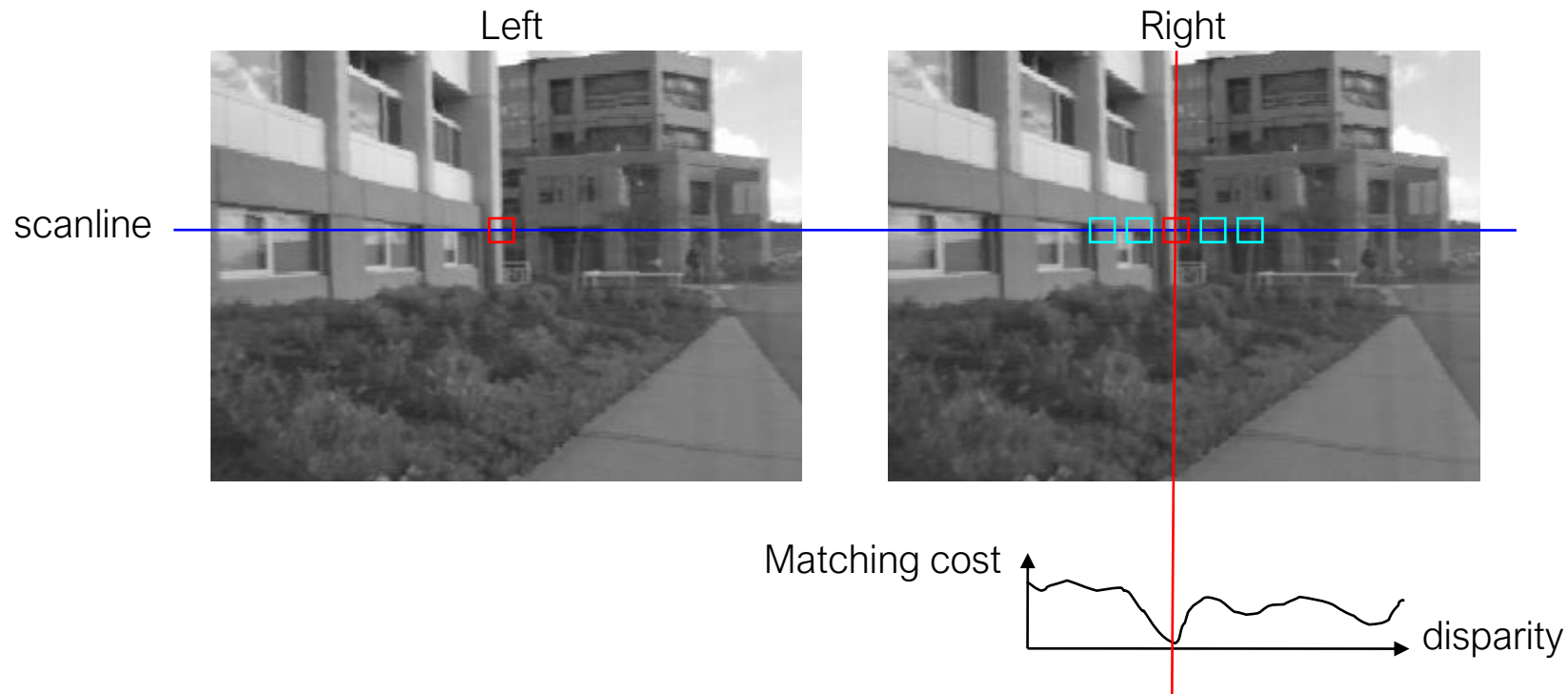
- Slide a window along the epipolar line and compare contents of th at window with the reference window in the left image
- Matching cost: SSD or normalized correlation

SSD
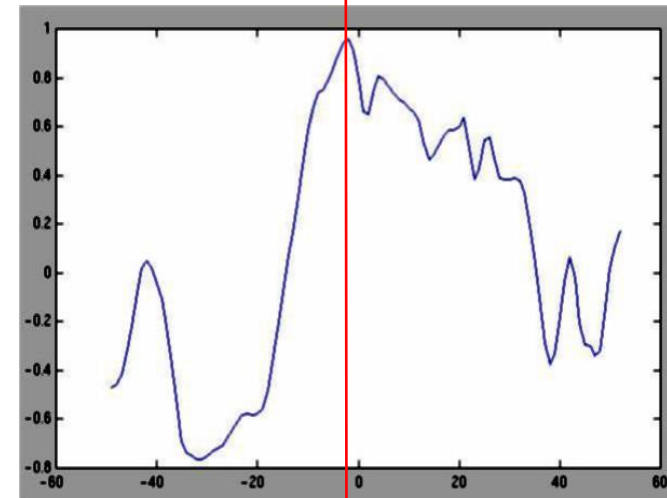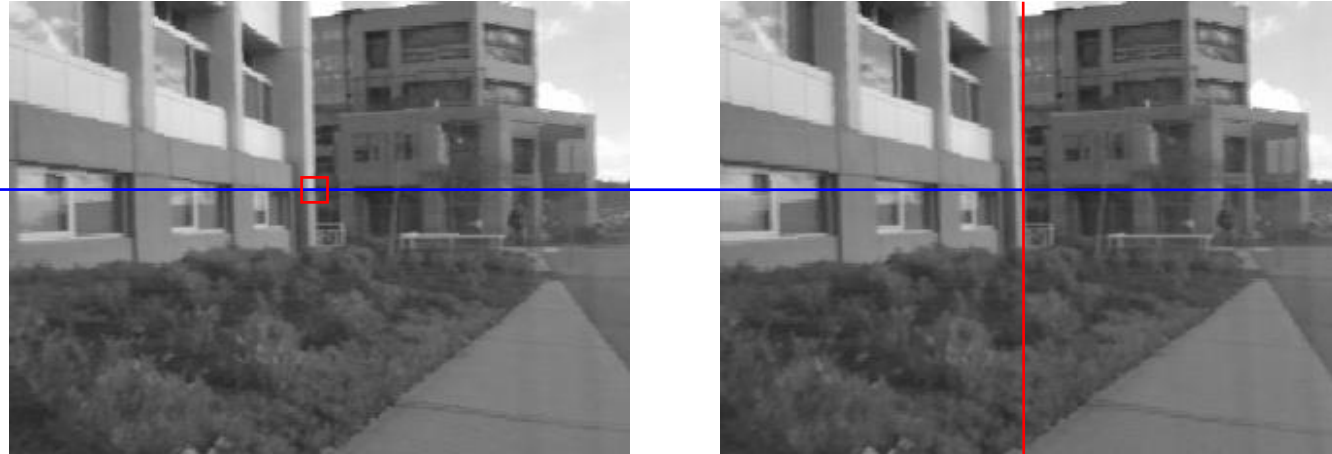
Normalized cross-correlation

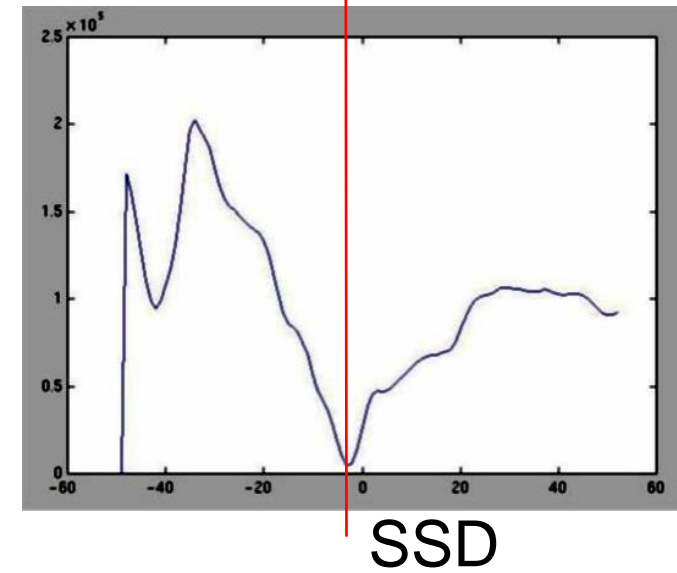| Similarity Measure	Formula
Sum of Absolute Differences (SAD)	$$\sum_{(i,j)\in W} |I_1(i,j) - I_2(x+i, y+j)|$$
Sum of Squared Differences (SSD)	$$\sum_{(i,j)\in W} \left(I_1(i,j) - I_2(x+i, y+j)\right)^2$$
Zero-mean SAD	$$\sum_{(i,j)\in W} |I_1(i,j) - \bar{I}_1(i,j) - I_2(x+i, y+j) + \bar{I}_2(x+i, y+j)|$$
Locally scaled SAD	$$\sum_{(i,j)\in W} |I_1(i,j) - \frac{\bar{I}_1(i,j)}{\bar{I}_2(x+i, y+j)} I_2(x+i, y+j)|$$
Normalized Cross Correlation (NCC)	$$\frac{\sum_{(i,j)\in W} I_1(i,j).I_2(x+i, y+j)}{\sqrt[2]{\sum_{(i,j)\in W} I_1^2(i,j).\sum_{(i,j)\in W} I_2^2(x+i, y+j)}}$$
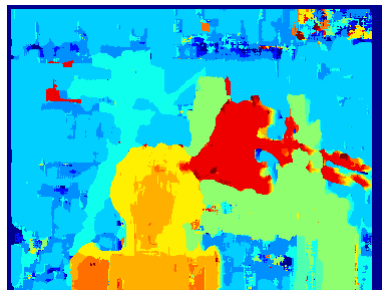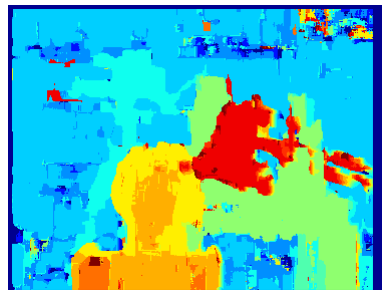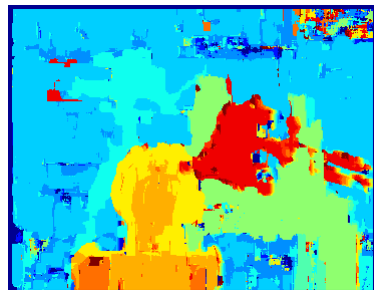
SAD   SSD   NCC   Ground truth

Effect of window size

W = 3

W = 20

Effect of window size

W = 3

W = 20

Smaller window
  + More detail
  - More noise
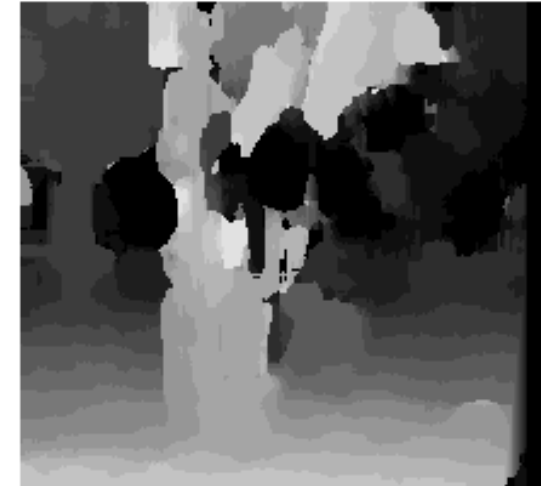
Larger window
  + Smoother disparity maps
  - Less detail
  - Fails near boundaries

When will stereo block matching fail?
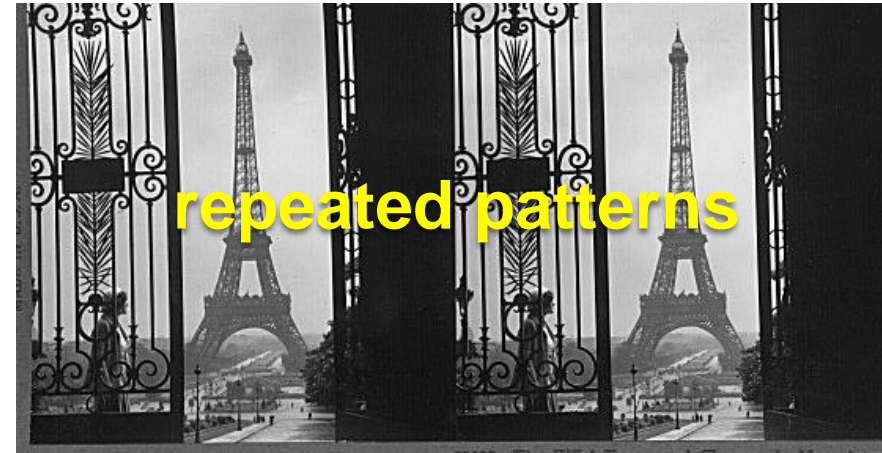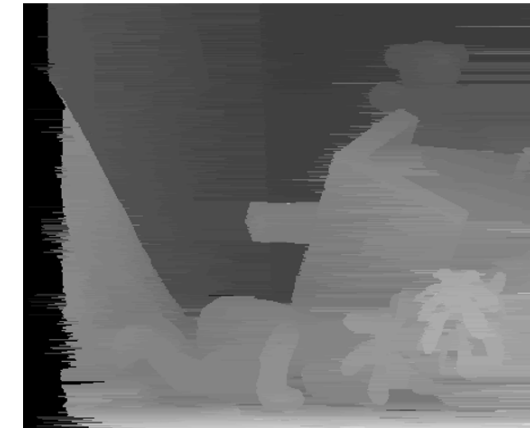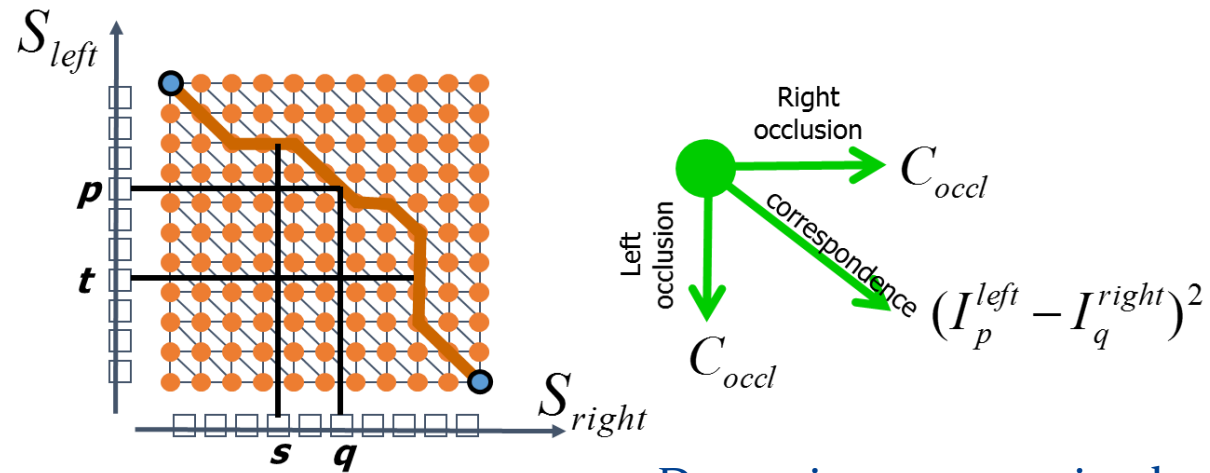
Dynamic programming based Stereo matching

(a) Minimum Cost Path $L_r(p, d)$
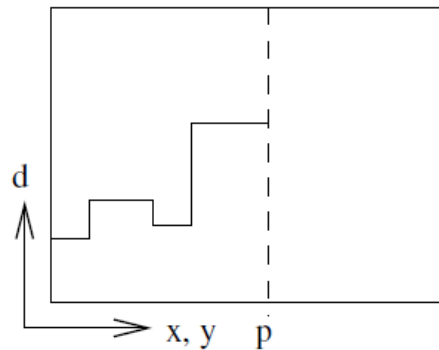
(b) 16 Paths from all Directions r
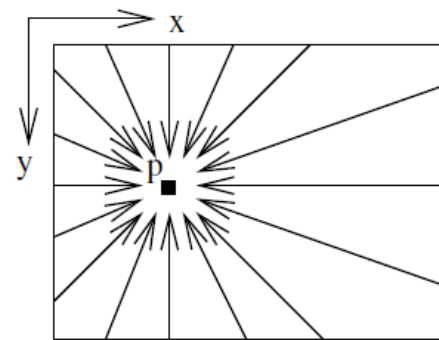
Figure 1. Aggregation of costs.

3D triangulation

```python
import cv2
import numpy as np

P1 = np.eye(3, 4, dtype=np.float32)
P2 = np.eye(3, 4, dtype=np.float32)
P2[0, 3] = -1

N = 5
points3d = np.empty((4, N), np.float32)
points3d[:3, :] = np.random.randn(3, N)
points3d[3, :] = 1

points1 = P1 @ points3d
points1 = points1[:2, :] / points1[2, :]
points1[:2, :] += np.random.randn(2, N) * 1e-2

points2 = P2 @ points3d
points2 = points2[:2, :] / points2[2, :]
points2[:2, :] += np.random.randn(2, N) * 1e-2

points3d_reconstr = cv2.triangulatePoints(P1, P2, points1, points2)
points3d_reconstr /= points3d_reconstr[3, :]

print('Original points')
print(points3d[:3].T)
print('Reconstructed points')
print(points3d_reconstr[:3].T)
```

Electronics Engineering, CBNU

Find relative camera-object pose using PnP algorithm

```python
import cv2
import numpy as np


camera_matrix = np.load('../data/pinhole_calib/camera_mat.npy')
dist_coefs = np.load('../data/pinhole_calib/dist_coefs.npy')
img = cv2.imread('../data/pinhole_calib/img_00.png')


pattern_size = (10, 7)
res, corners = cv2.findChessboardCorners(img, pattern_size)
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 1e-3)
corners = cv2.cornerSubPix(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY),
                           corners, (10, 10), (-1, -1), criteria)


pattern_points = np.zeros((np.prod(pattern_size), 3), np.float32)
pattern_points[:, :2] = np.indices(pattern_size).T.reshape(-1, 2)


ret, rvec, tvec = cv2.solvePnP(pattern_points, corners, camera_matrix, dist_coefs,
                               None, None, False, cv2.SOLVEPNP_ITERATIVE)


img_points, _ = cv2.projectPoints(pattern_points, rvec, tvec, camera_matrix, dist_coefs)

for c in img_points.squeeze():
    cv2.circle(img, tuple(c), 10, (0, 255, 0), 2)

cv2.imshow('points', img)
cv2.waitKey()

cv2.destroyAllWindows()
```

Electronics Engineering, CBNU

Stereo rectification

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams.update({'font.size': 20})
np_load_old = np.load
np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)


data = np.load('../data/stereo/case1/stereo.npy').item()
Kl, Dl, Kr, Dr, R, T, img_size = data['Kl'], data['Dl'], data['Kr'], data['Dr'], \
                                  data['R'], data['T'], data['img_size']


left_img = cv2.imread('../data/stereo/case1/left14.png')
right_img = cv2.imread('../data/stereo/case1/right14.png')

R1, R2, P1, P2, Q, validRoi1, validRoi2 = cv2.stereoRectify(Kl, Dl, Kr, Dr, img_size, R, T)

xmap1, ymap1 = cv2.initUndistortRectifyMap(Kl, Dl, R1, Kl, img_size, cv2.CV_32FC1)
xmap2, ymap2 = cv2.initUndistortRectifyMap(Kr, Dr, R2, Kr, img_size, cv2.CV_32FC1)

left_img_rectified = cv2.remap(left_img, xmap1, ymap1, cv2.INTER_LINEAR)
right_img_rectified = cv2.remap(right_img, xmap2, ymap2, cv2.INTER_LINEAR)
```

```python
plt.figure(0, figsize=(12,10))
plt.subplot(221)
plt.title('left original')
plt.imshow(left_img, cmap='gray')
plt.subplot(222)
plt.title('right original')
plt.imshow(right_img, cmap='gray')
plt.subplot(223)
plt.title('left rectified')
plt.imshow(left_img_rectified, cmap='gray')
plt.subplot(224)
plt.title('right rectified')
plt.imshow(right_img_rectified, cmap='gray')
plt.tight_layout()
plt.show()
```

Electronics Engineering, CBNU

Fundamental matrix computation

```python
import cv2
import numpy as np


np_load_old = np.load
np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)


data = np.load('../data/stereo/case1/stereo.npy').item()
Kl, Kr, Dl, Dr, left_pts, right_pts, E_from_stereo, F_from_stereo = \
data['Kl'], data['Kr'], data['Dl'], data['Dr'], data['left_pts'], data['right_pts'], data['E'], data['F']


left_pts = np.vstack(left_pts)
right_pts = np.vstack(right_pts)


left_pts = cv2.undistortPoints(left_pts, Kl, Dl, P=Kl)
right_pts = cv2.undistortPoints(right_pts, Kr, Dr, P=Kr)


F, mask = cv2.findFundamentalMat(left_pts, right_pts, cv2.FM_LMEDS)


E = Kr.T @ F @ Kl


print('Fundamental matrix:')
print(F)
print('Essential matrix:')
print(E)
```

Electronics Engineering, CBNU

Essential decomposition into rotation and translation

```python
import cv2
import numpy as np


np_load_old = np.load
np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)


data = np.load('../data/stereo/case1/stereo.npy').item()
E = data['E']


R1, R2, T = cv2.decomposeEssentialMat(E)


print('Rotation 1:')
print(R1)
print('Rotation 2:')
print(R2)
print('Translation:')
print(T)
```

Estimating disparity map for stereo images

```python
import cv2
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams.update({'font.size': 20})

left_img = cv2.imread('../data/stereo/left.png')
right_img = cv2.imread('../data/stereo/right.png')

stereo_bm = cv2.StereoBM_create(32)
dispmap_bm = stereo_bm.compute(cv2.cvtColor(left_img, cv2.COLOR_BGR2GRAY),
                               cv2.cvtColor(right_img, cv2.COLOR_BGR2GRAY))

stereo_sgbm = cv2.StereoSGBM_create(0, 32)
dispmap_sgbm = stereo_sgbm.compute(left_img, right_img)

plt.figure(figsize=(12,10))
plt.subplot(221)
plt.title('left')
plt.imshow(left_img[:, :, [2,1,0]])
plt.subplot(222)
plt.title('right')
plt.imshow(right_img[:, :, [2,1,0]])
plt.subplot(223)
plt.title('BM')
plt.imshow(dispmap_bm, cmap='gray')
plt.subplot(224)
plt.title('SGBM')
plt.imshow(dispmap_sgbm, cmap='gray')
plt.show()
```

Electronics Engineering, CBNU