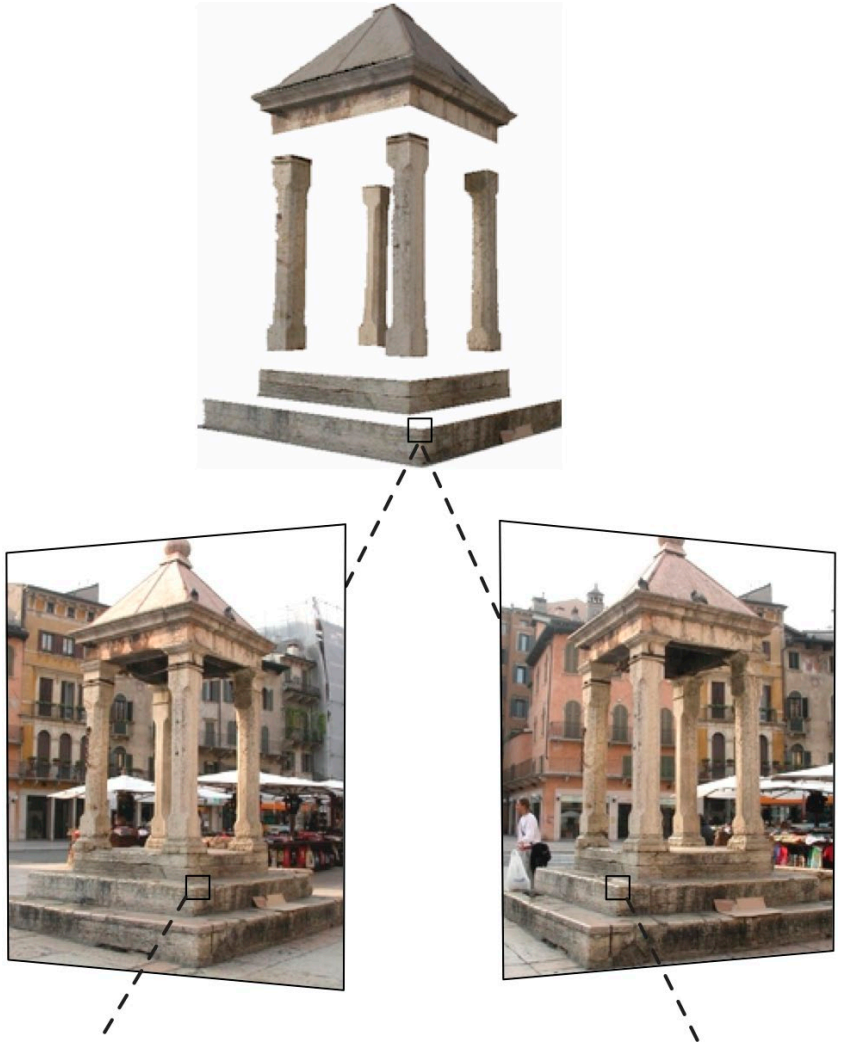
Two-view geometry



https://cmu-ci-lab.github.io/16705

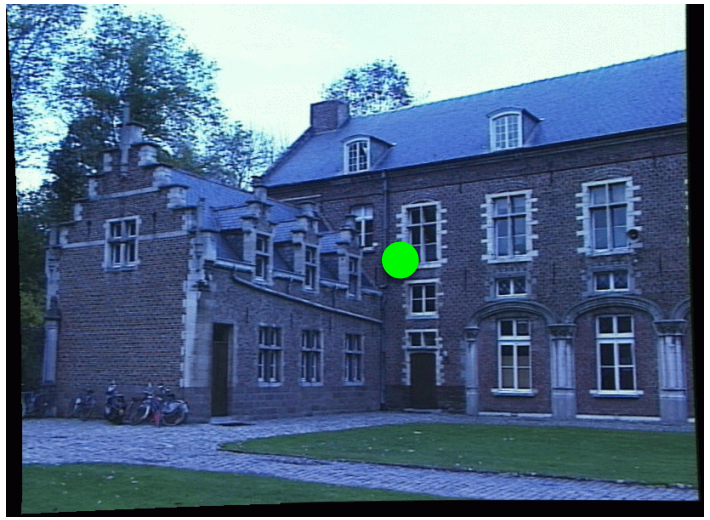
16-705 Fundamentals of Computer Vision
Spring 2026, Lecture 4

Course announcements

- Anyone not on Slack or Canvas?

Triangulation

Stereo pair of images



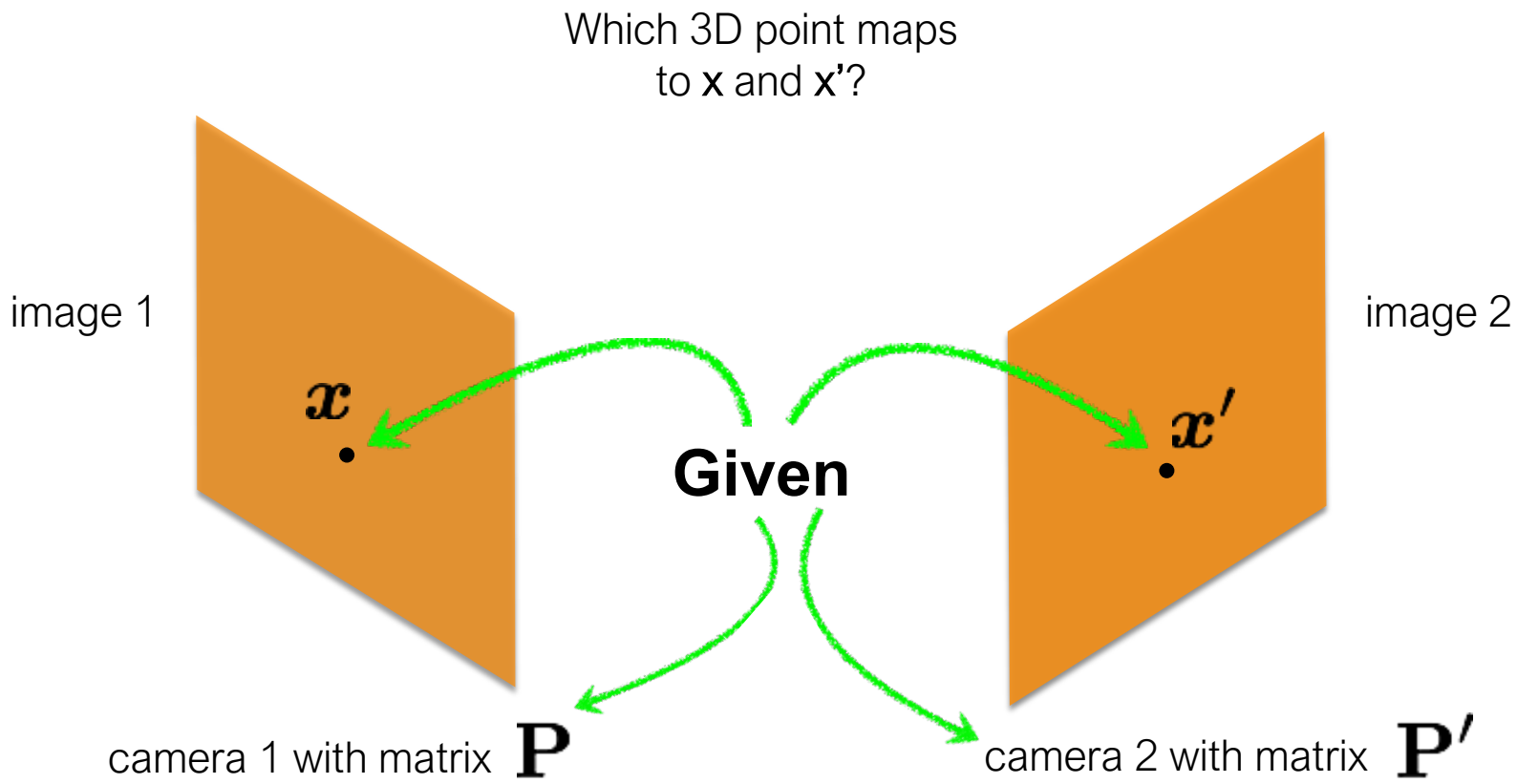
Left image

Right image

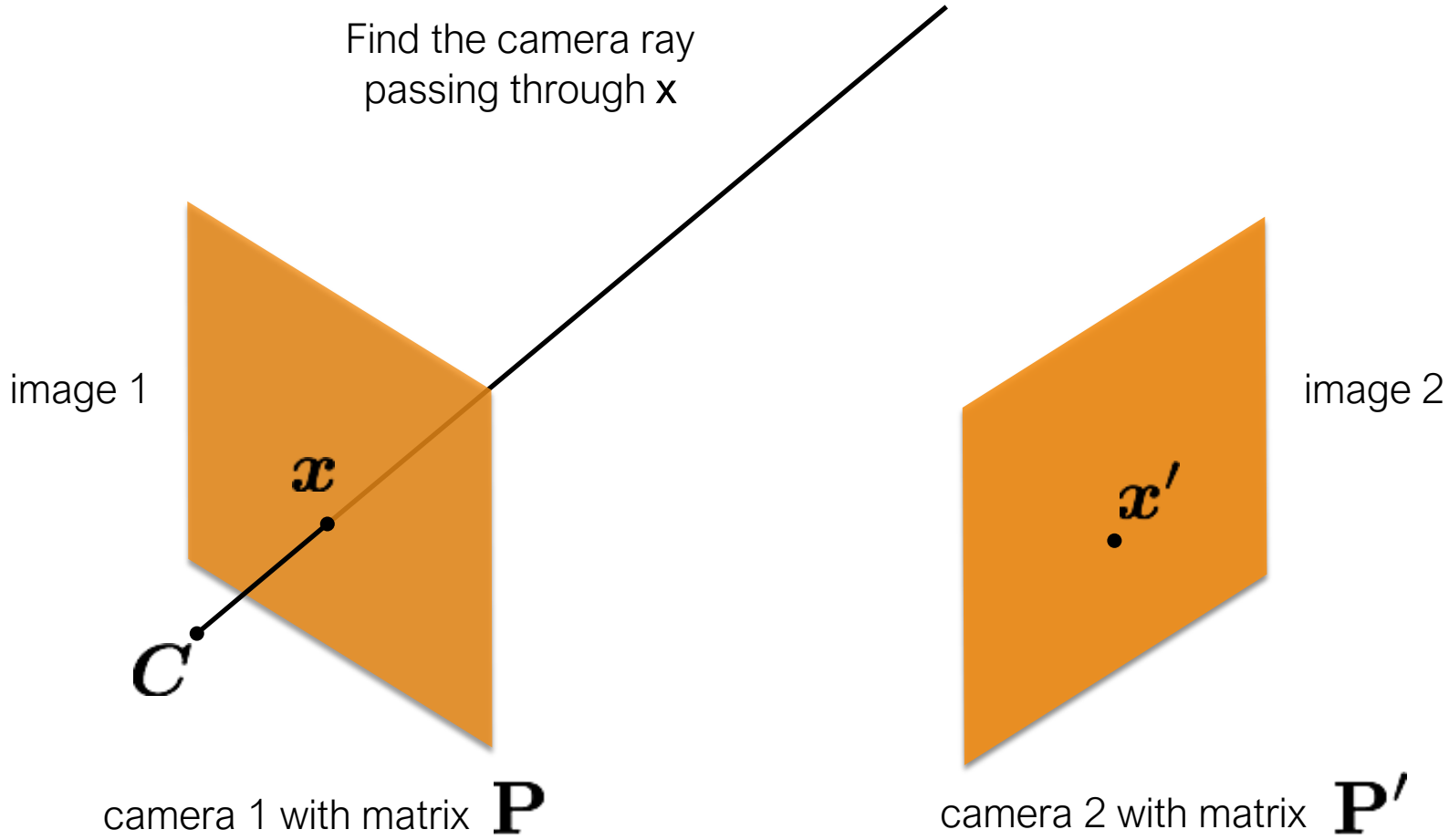
Matched point in images from *calibrated* camera pair

We want to reconstruct the corresponding 3D point

Triangulation



Triangulation

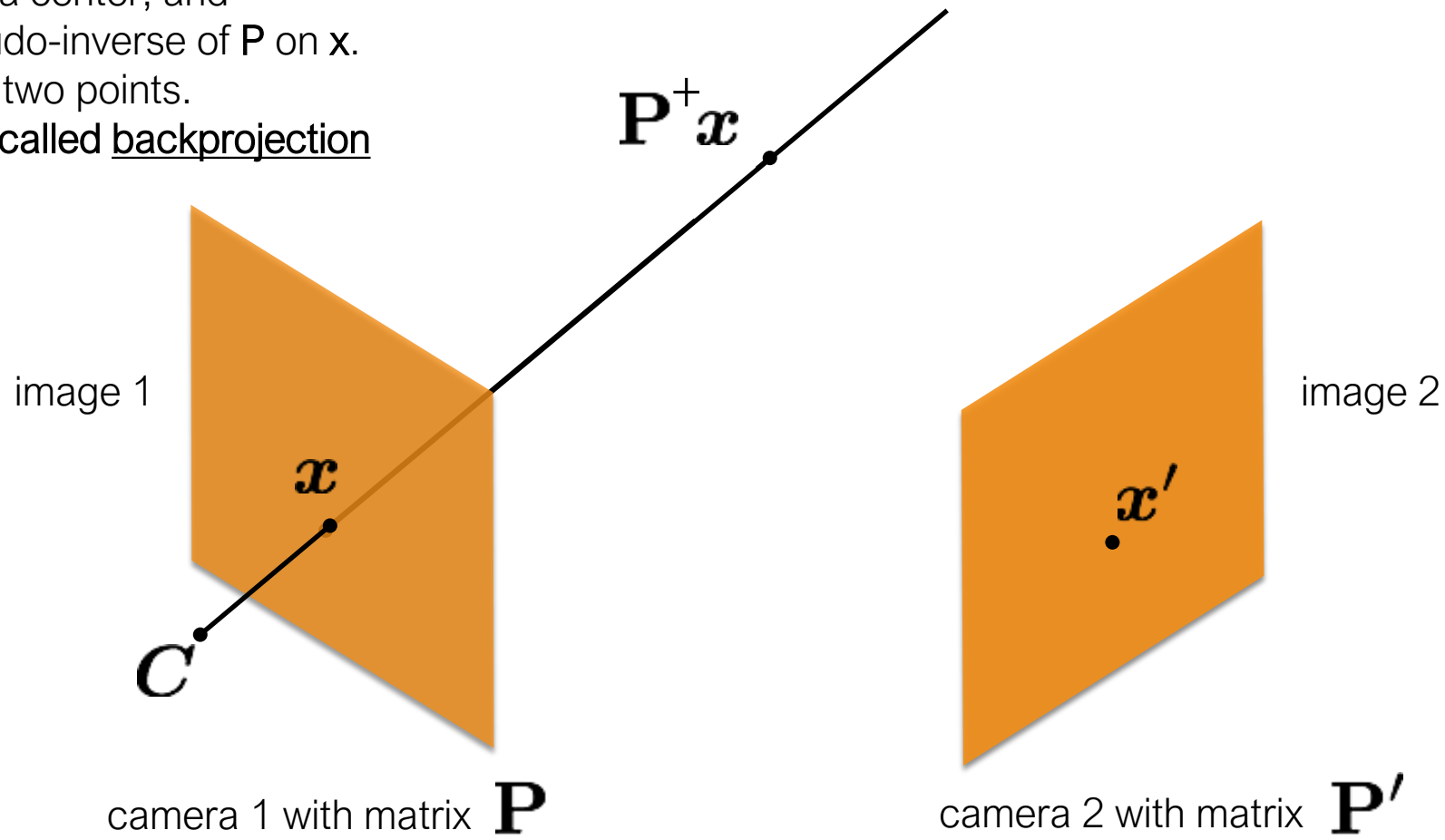


Triangulation

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

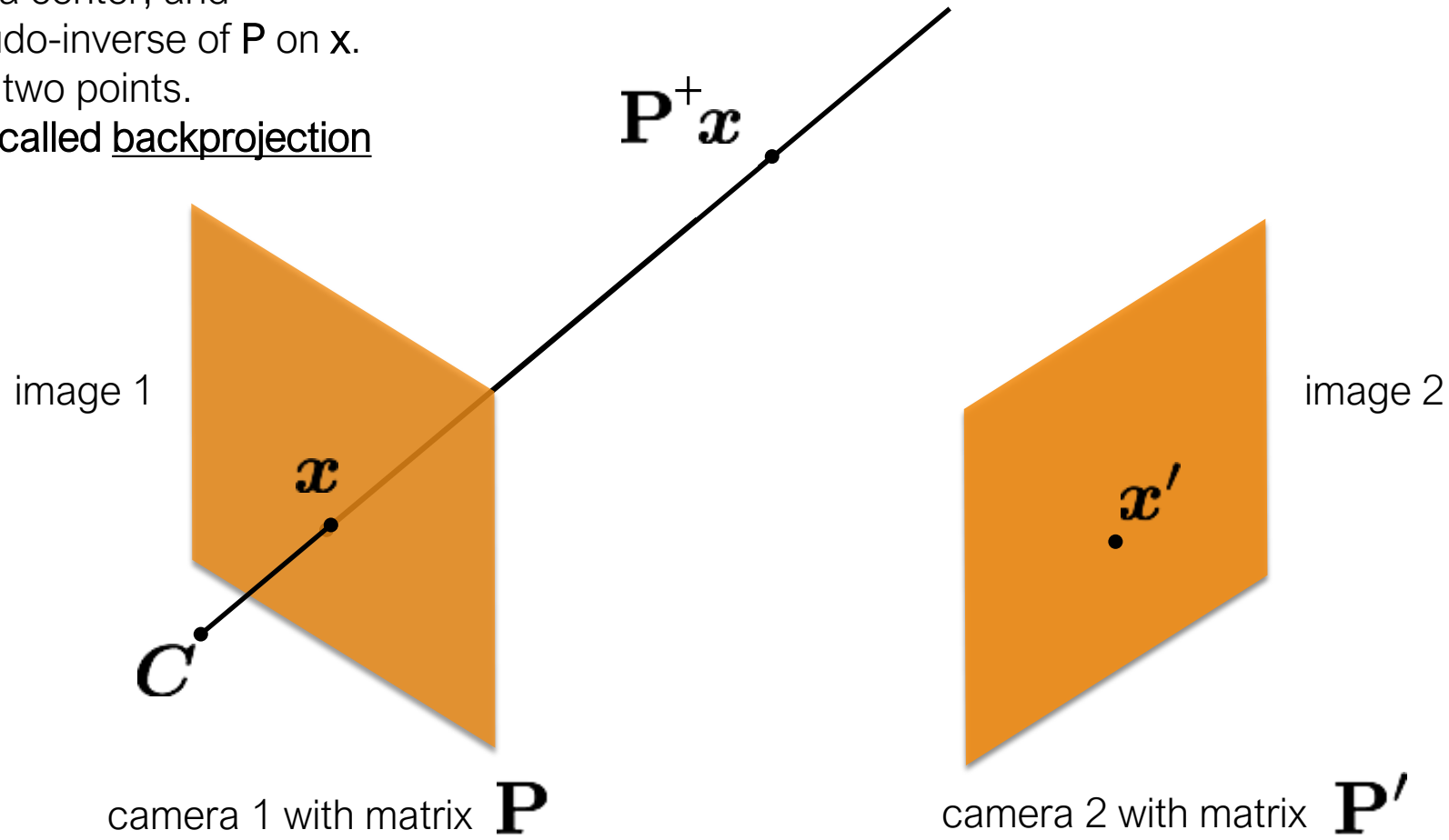


Triangulation

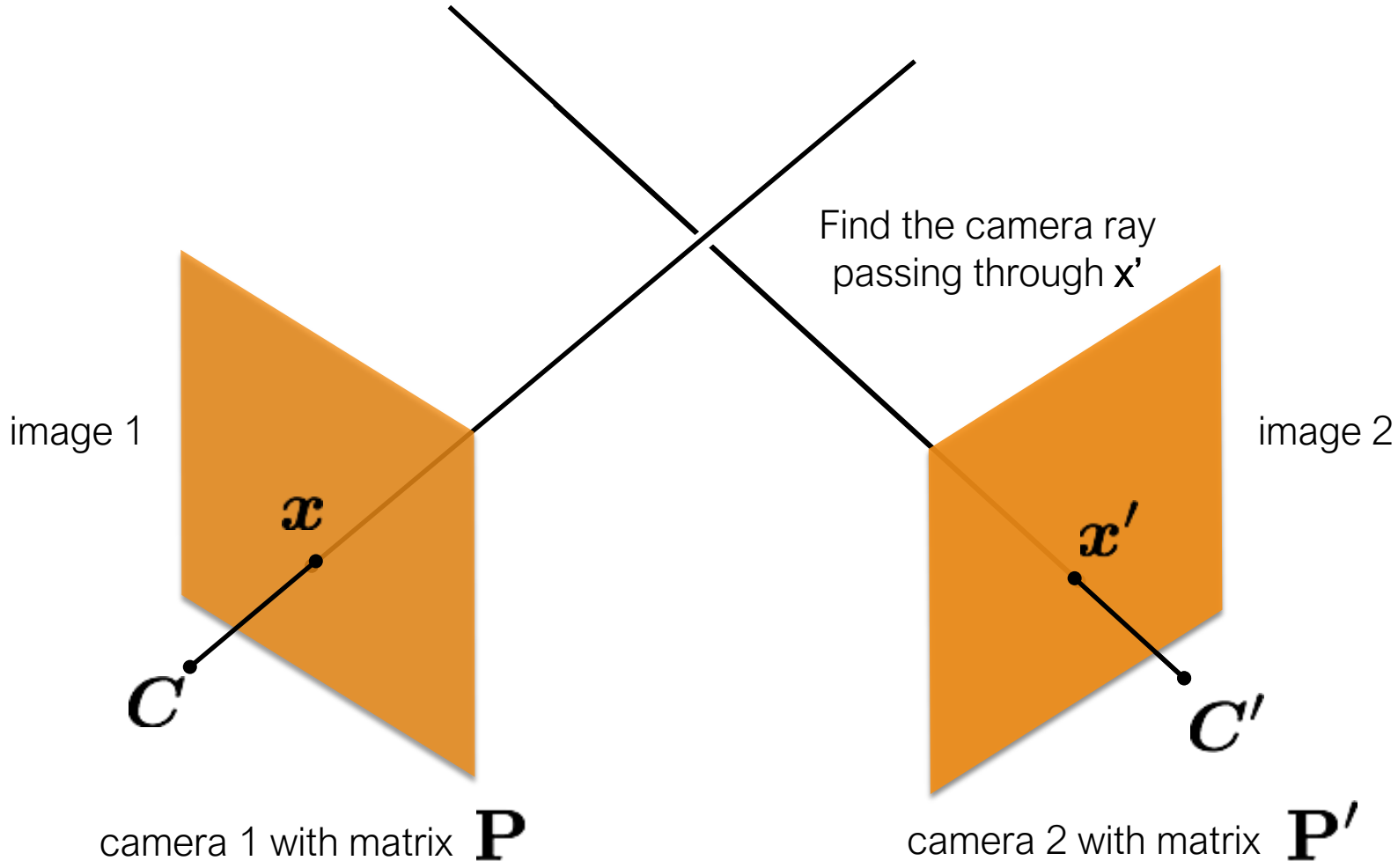
Create two points on the ray:
1) find the camera center; and
2) apply the pseudo-inverse of P on x.

Then connect the two points.

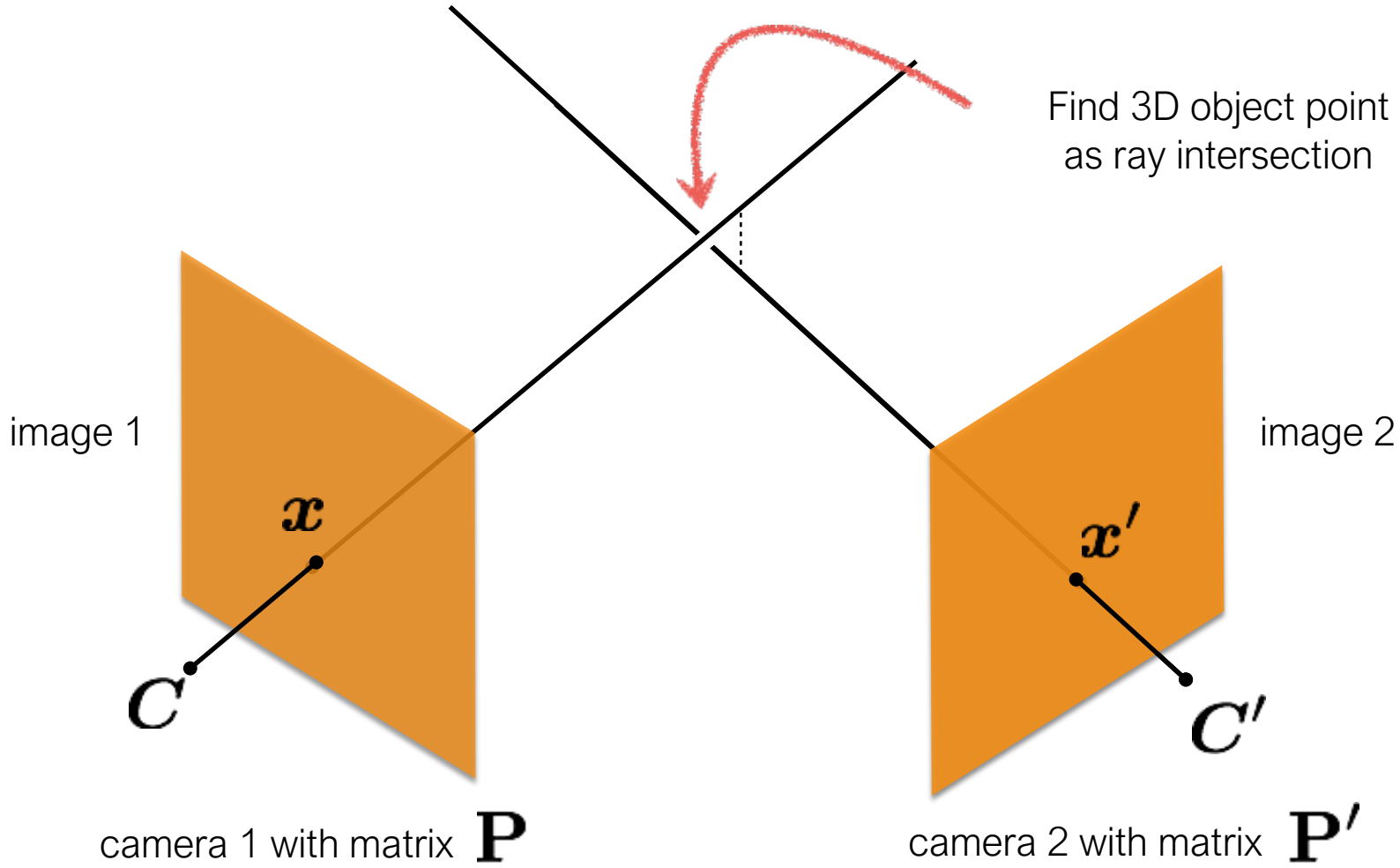
This procedure is called backprojection



Triangulation



Triangulation



Triangulation

Given a pair of (noisy) matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}_i'\}$$

and camera matrices

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

Estimate the 3D point

$$\mathbf{X}$$

Write out the mapping from 3D to 2D homogeneous coordinates

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

(homogeneous coordinate)

The scale factor is due to the unknown scalar when we convert from the measured heterogeneous to homogeneous coordinates

Write out the mapping from 3D to 2D homogeneous coordinates

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

(homogeneous coordinate)

The scale factor is due to the unknown scalar when we convert from the measured heterogeneous to homogeneous coordinates

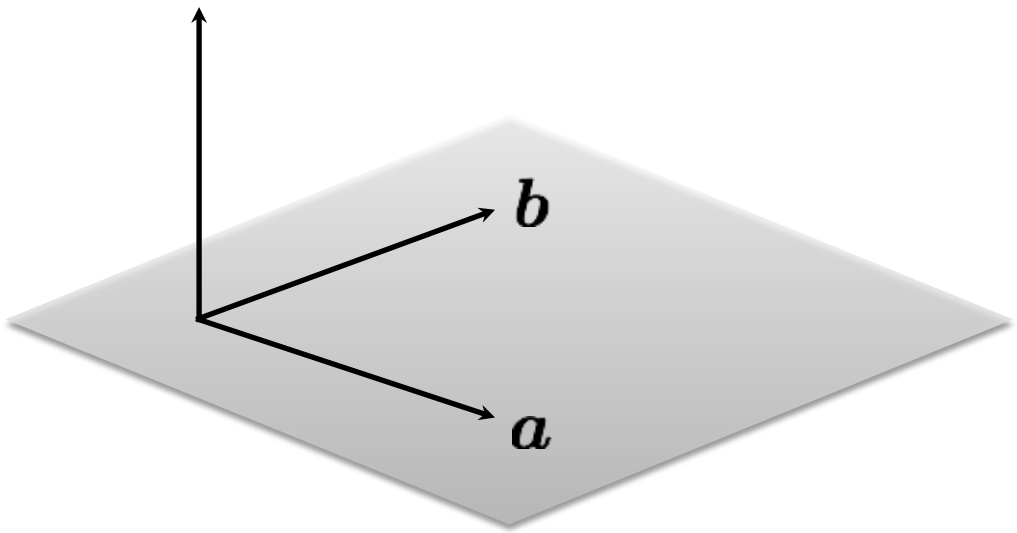
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

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

Linear algebra reminder: cross product

Vector (cross) product
takes two vectors and returns a vector perpendicular to both

$$\boldsymbol{c} = \boldsymbol{a} \times \boldsymbol{b}$$



$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

cross product of two vectors in the same direction is zero vector

$$\boldsymbol{a} \times \boldsymbol{a} = 0$$

$$\boldsymbol{c} \cdot \boldsymbol{a} = 0 \qquad \boldsymbol{c} \cdot \boldsymbol{b} = 0$$

Back to triangulation

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

Same direction but differs by a scale factor

Back to triangulation

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

Same direction but differs by a scale factor

Back to triangulation

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

Same direction but differs by a scale factor

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

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

Back to triangulation

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

Same direction but differs by a scale factor

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

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

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

Expand out the camera matrix

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 point gives us 2 constraints on the unknown 3D point

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

Remove third row

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

Remove third row

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

Rearrange equations as system on unknown 3D point

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

Remove third row

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

Rearrange equations as system on unknown 3D point

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

Homogeneous linear system on unknown 3D point

Concatenate the 2D points from both images

Two rows from camera one

Two rows from camera two

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

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

Concatenate the 2D points from both images

Two rows from camera one

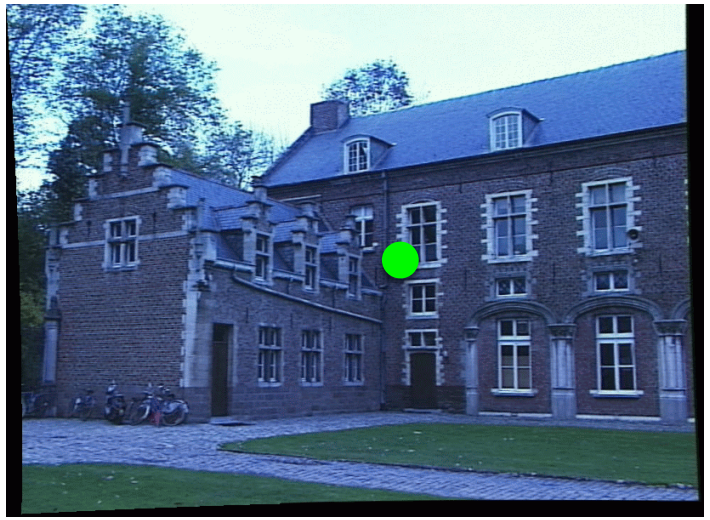
Two rows from camera two

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

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

Solve homogeneous linear system using **SVD**

Remember what we need as input



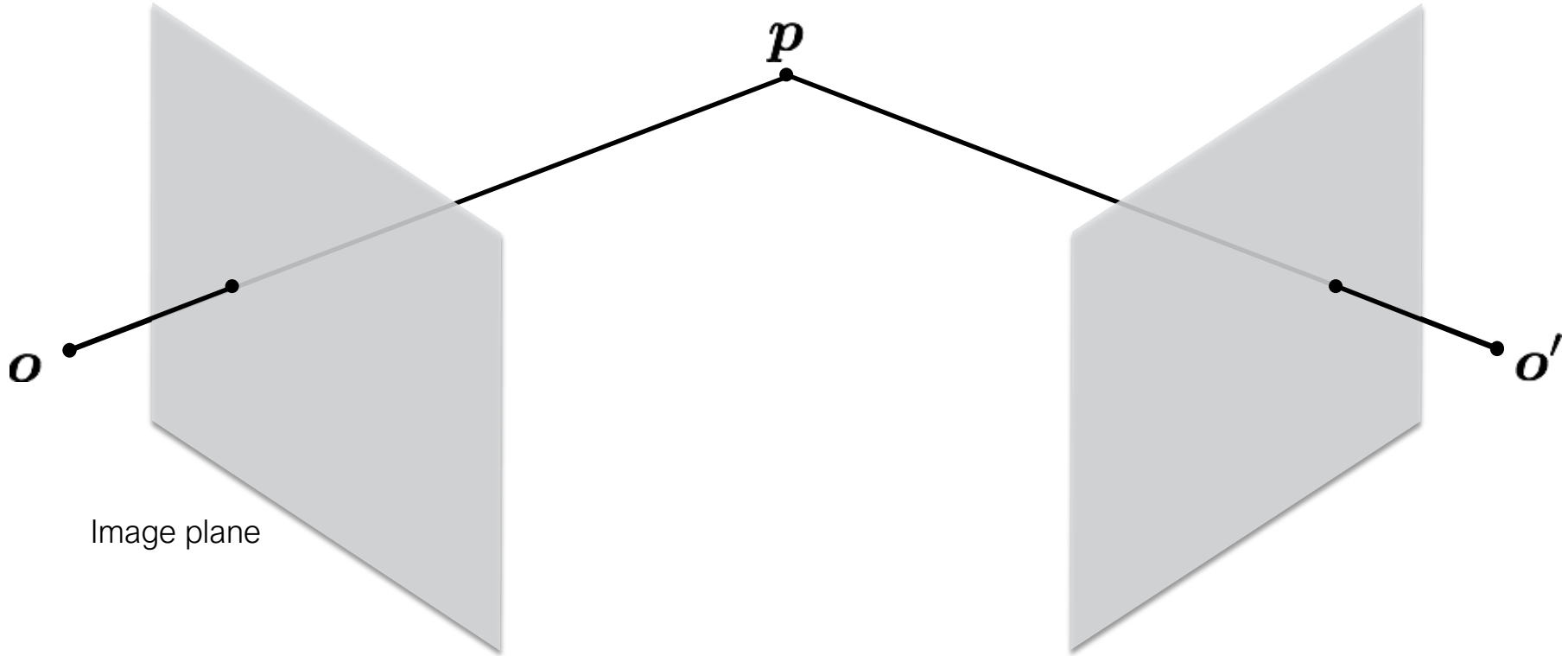
Left image

Right image

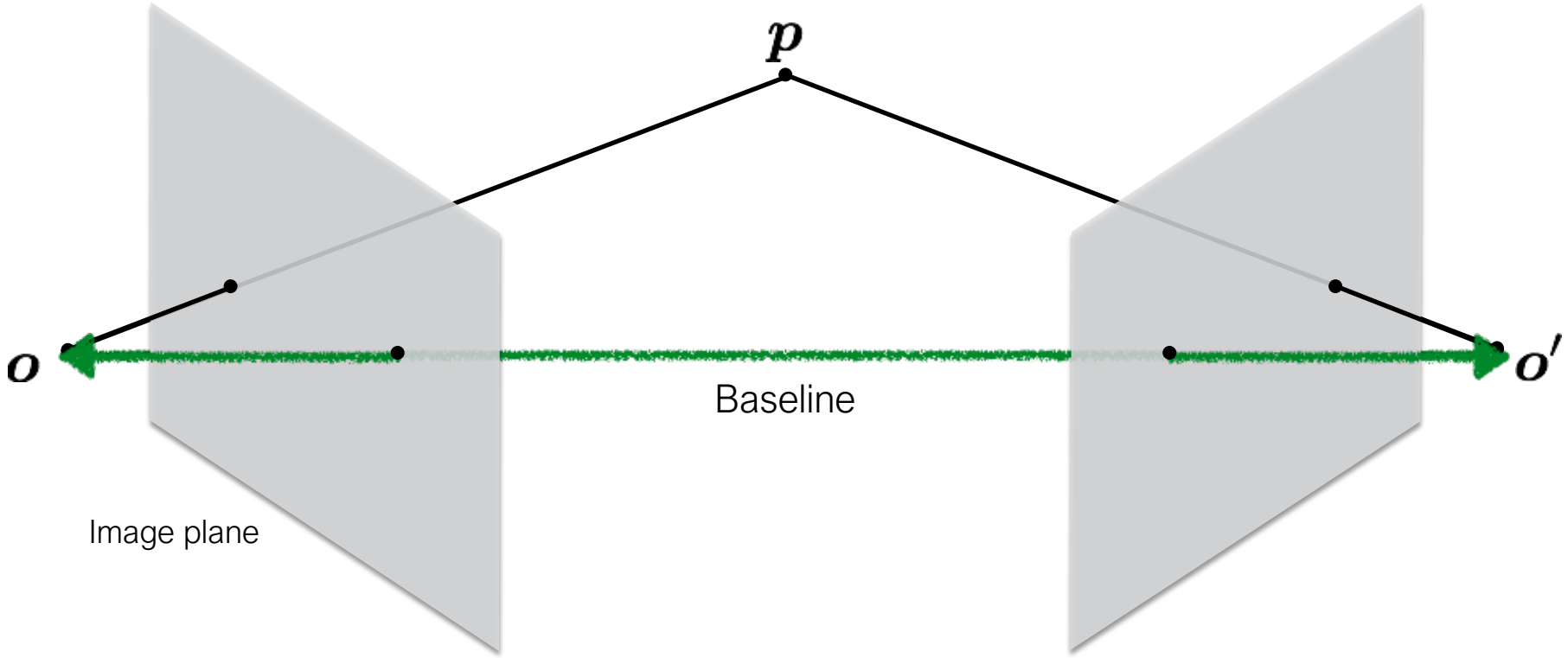
Matching a point in the left image requires doing an expensive 2D search in the right image.

Epipolar geometry

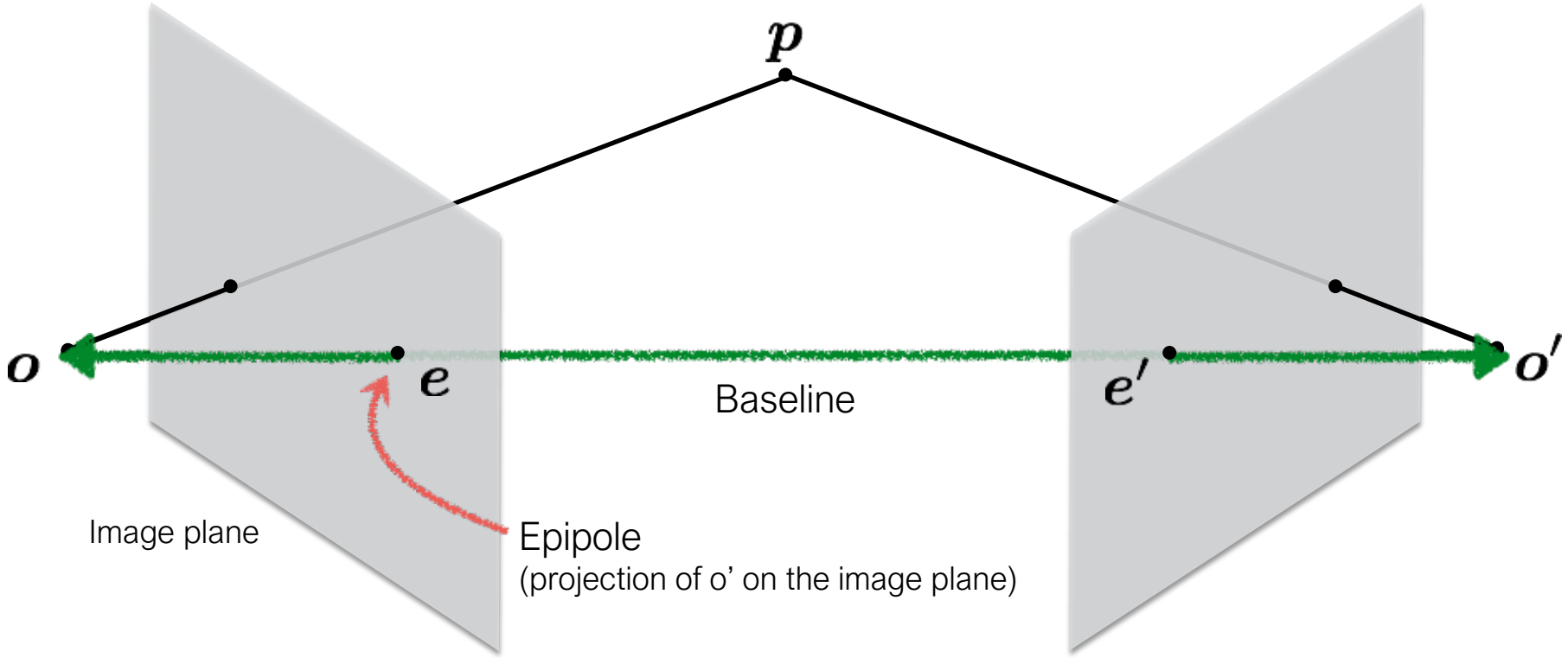
Epipolar geometry



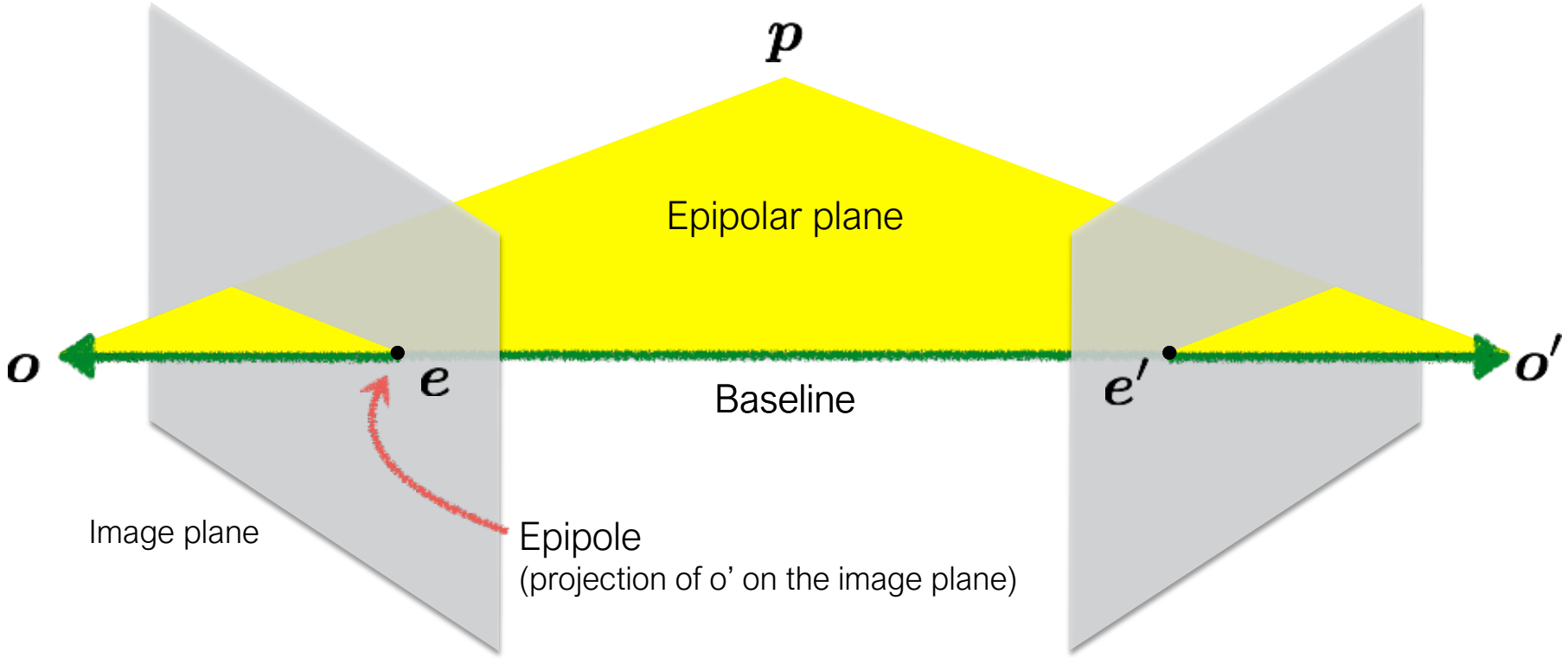
Epipolar geometry



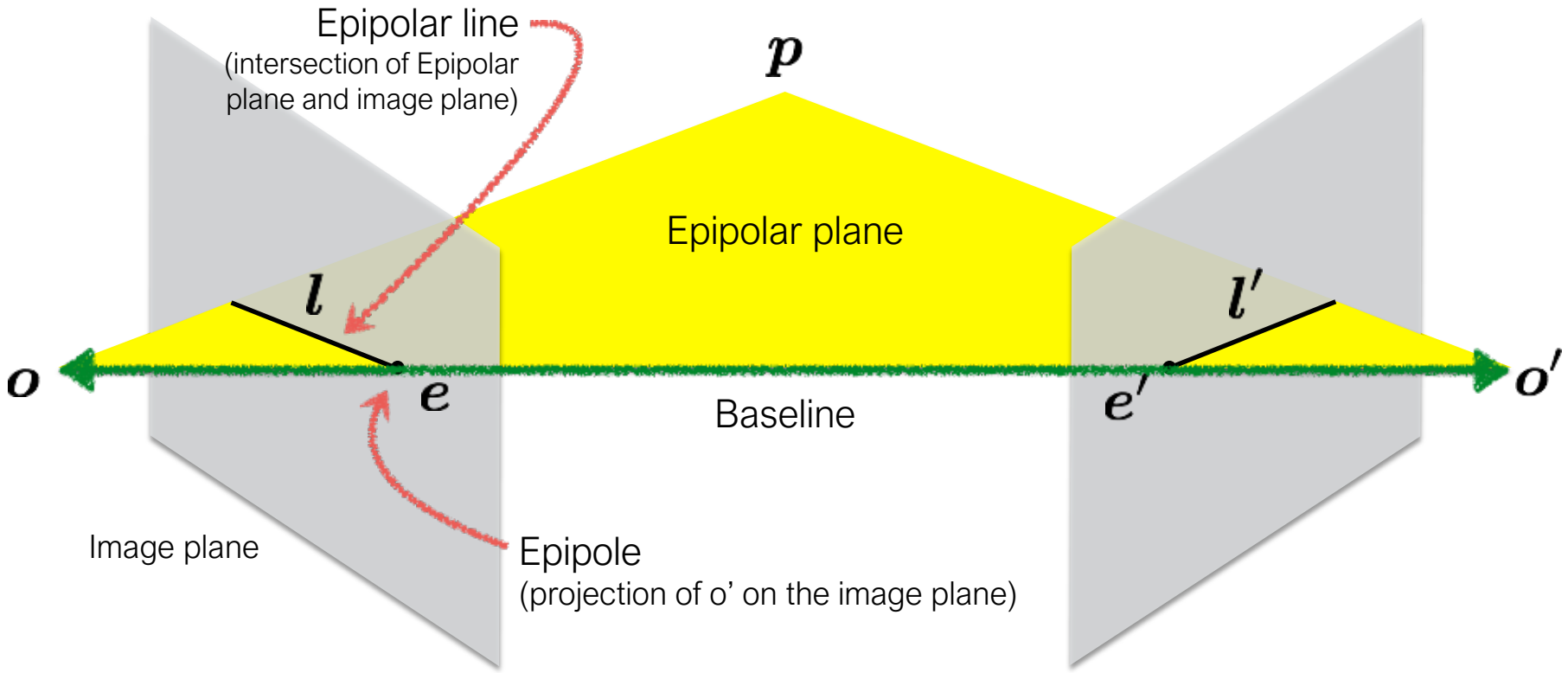
Epipolar geometry



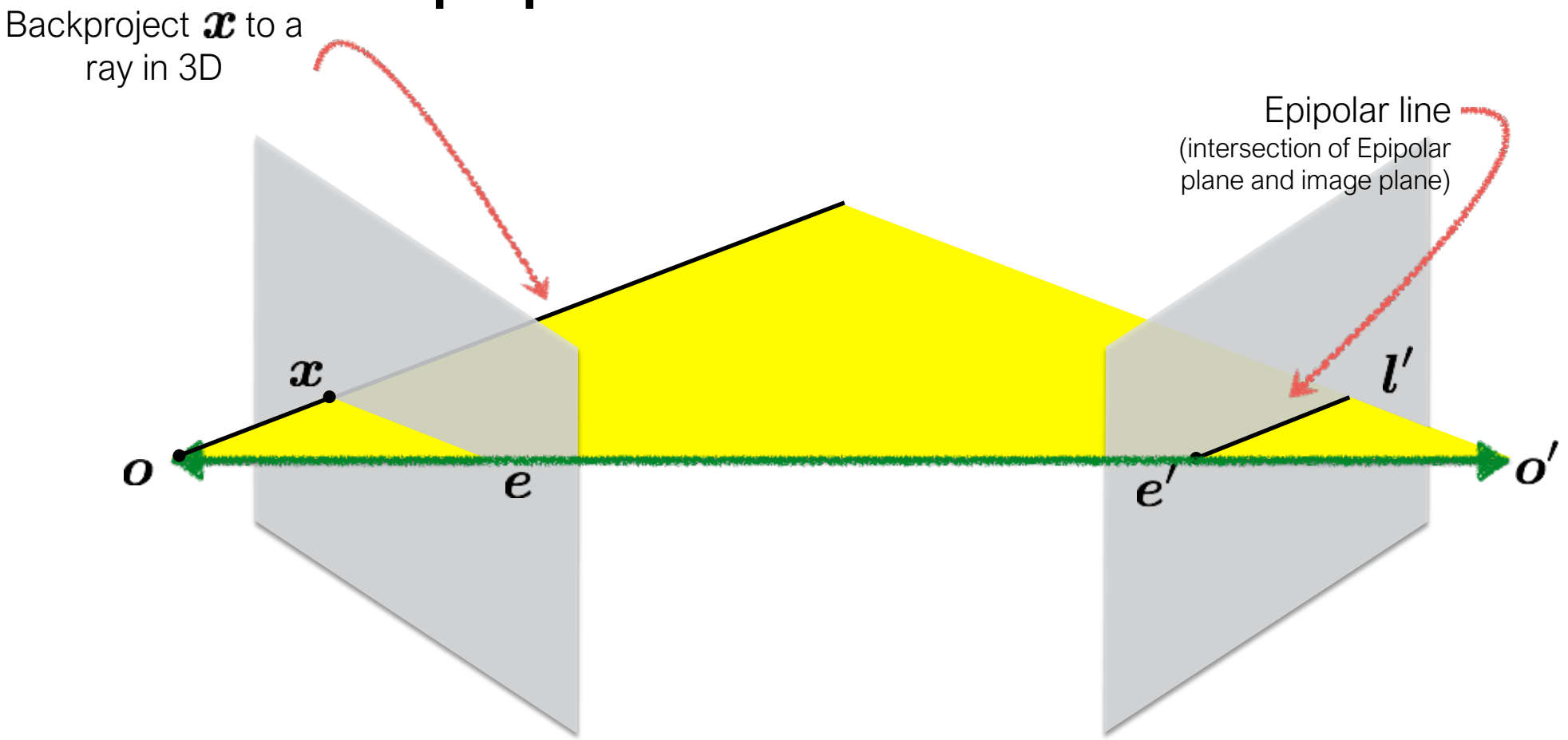
Epipolar geometry



Epipolar geometry

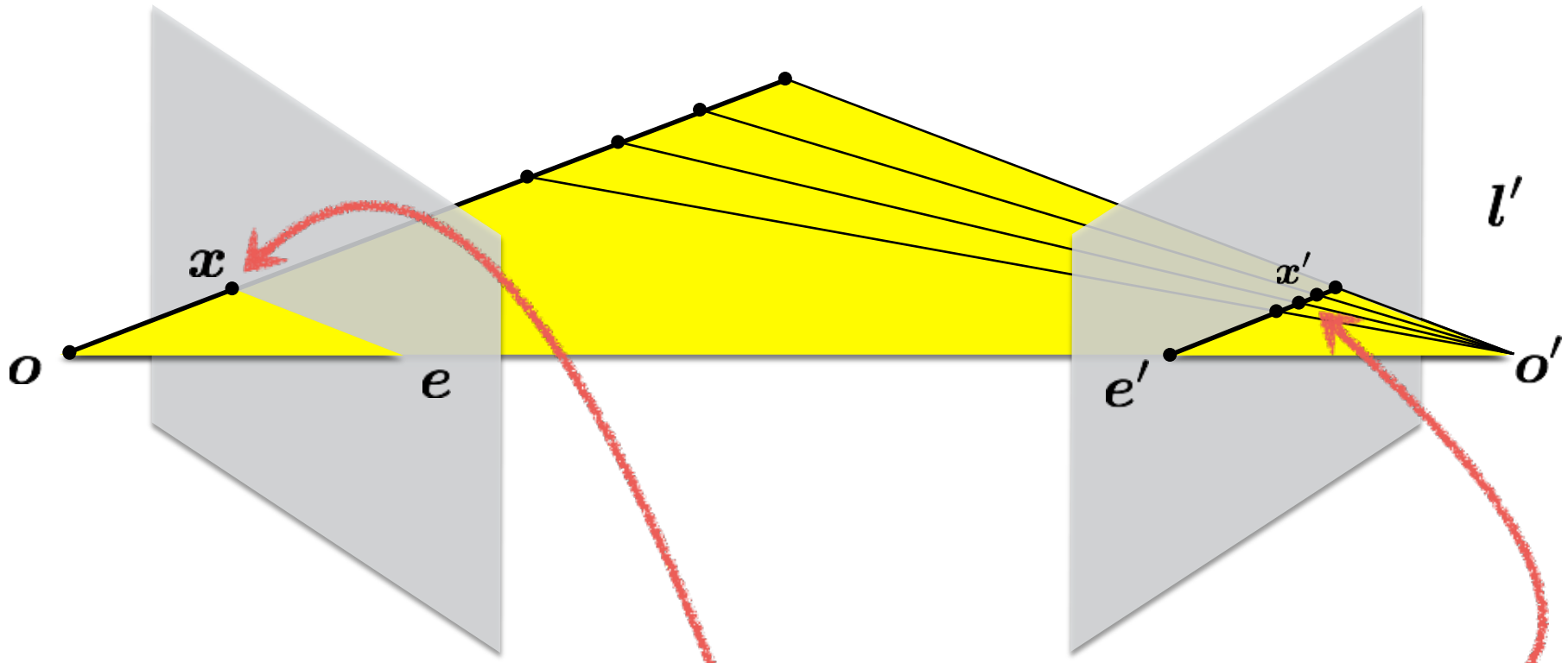


Epipolar constraint



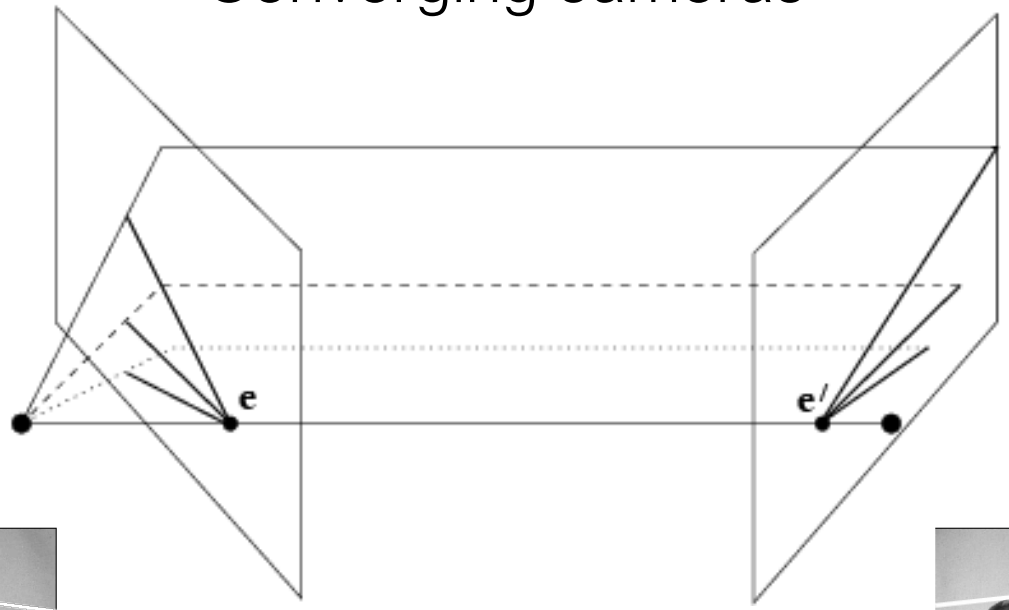
Another way to construct the epipolar plane, this time given x

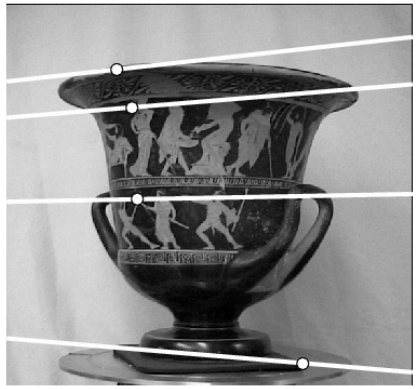
Epipolar constraint



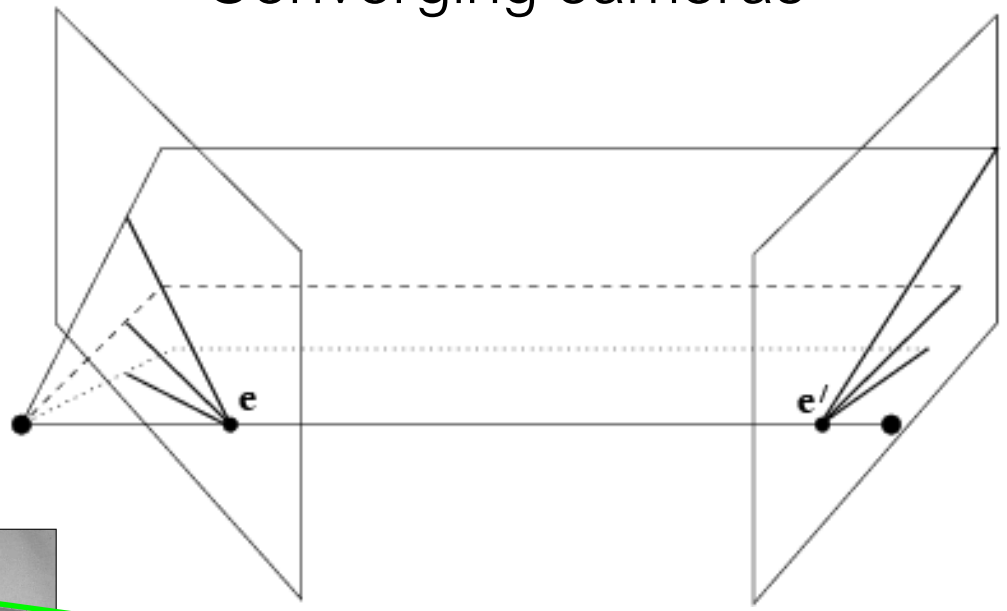
Potential matches for x lie on the epipolar line l'

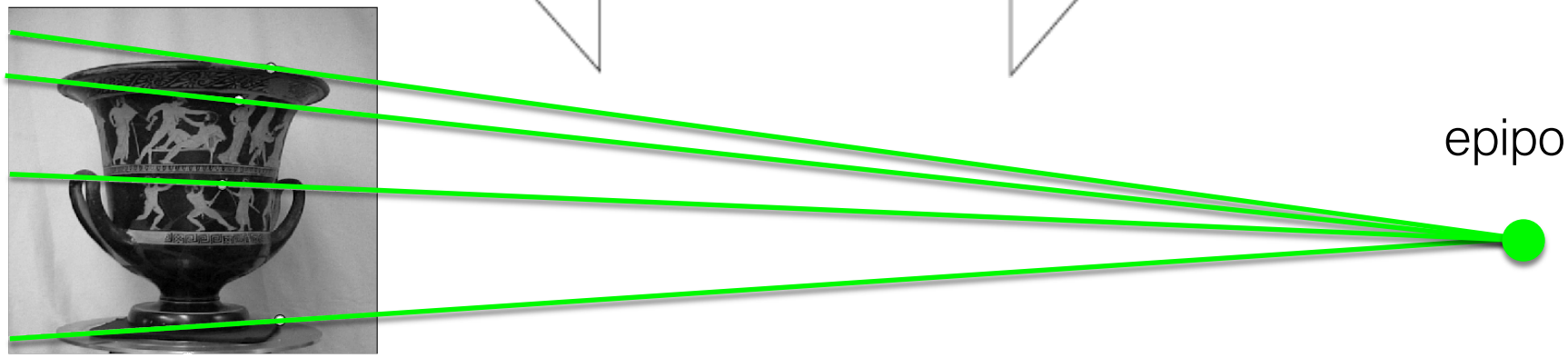
Converging cameras





Converging cameras



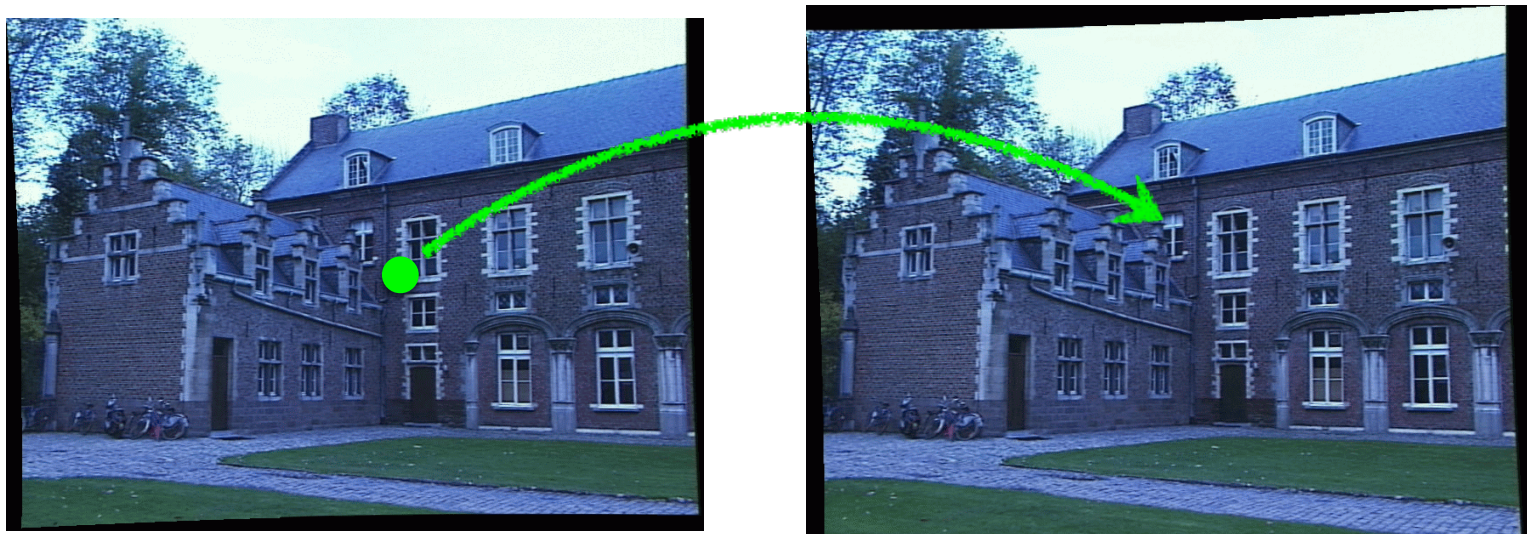


epipole

It’s not always in the image

Why is the epipolar constraint useful

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

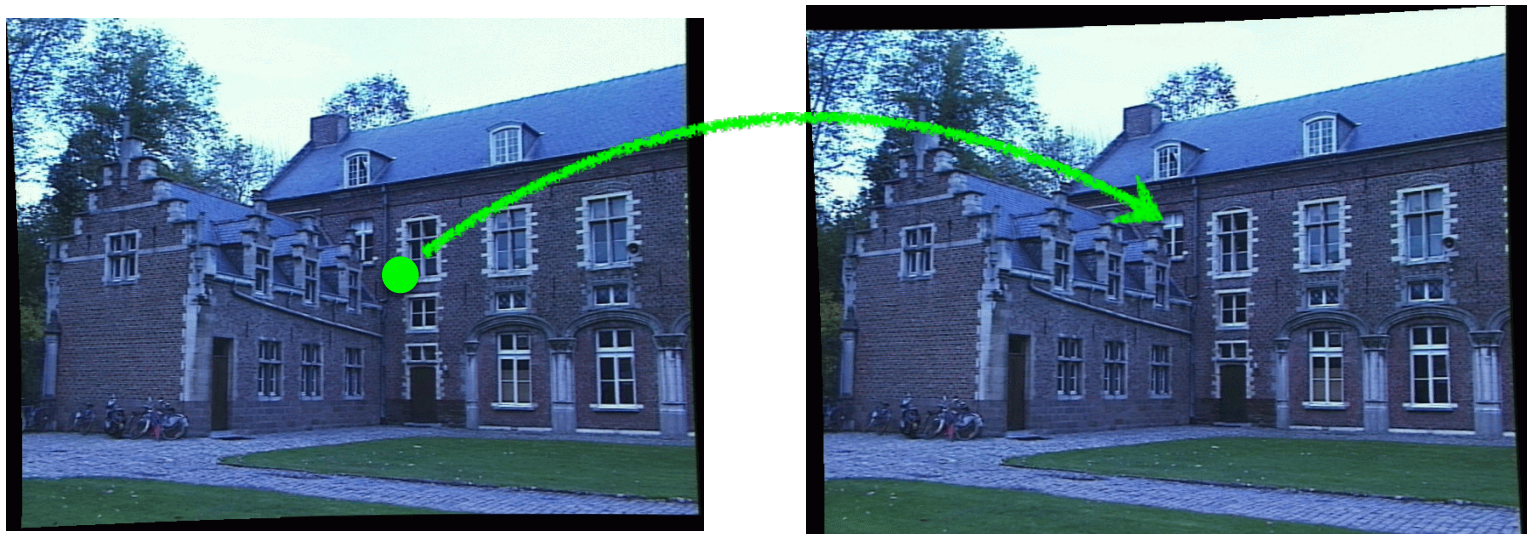


Left image

Right image

Why is the epipolar constraint useful

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



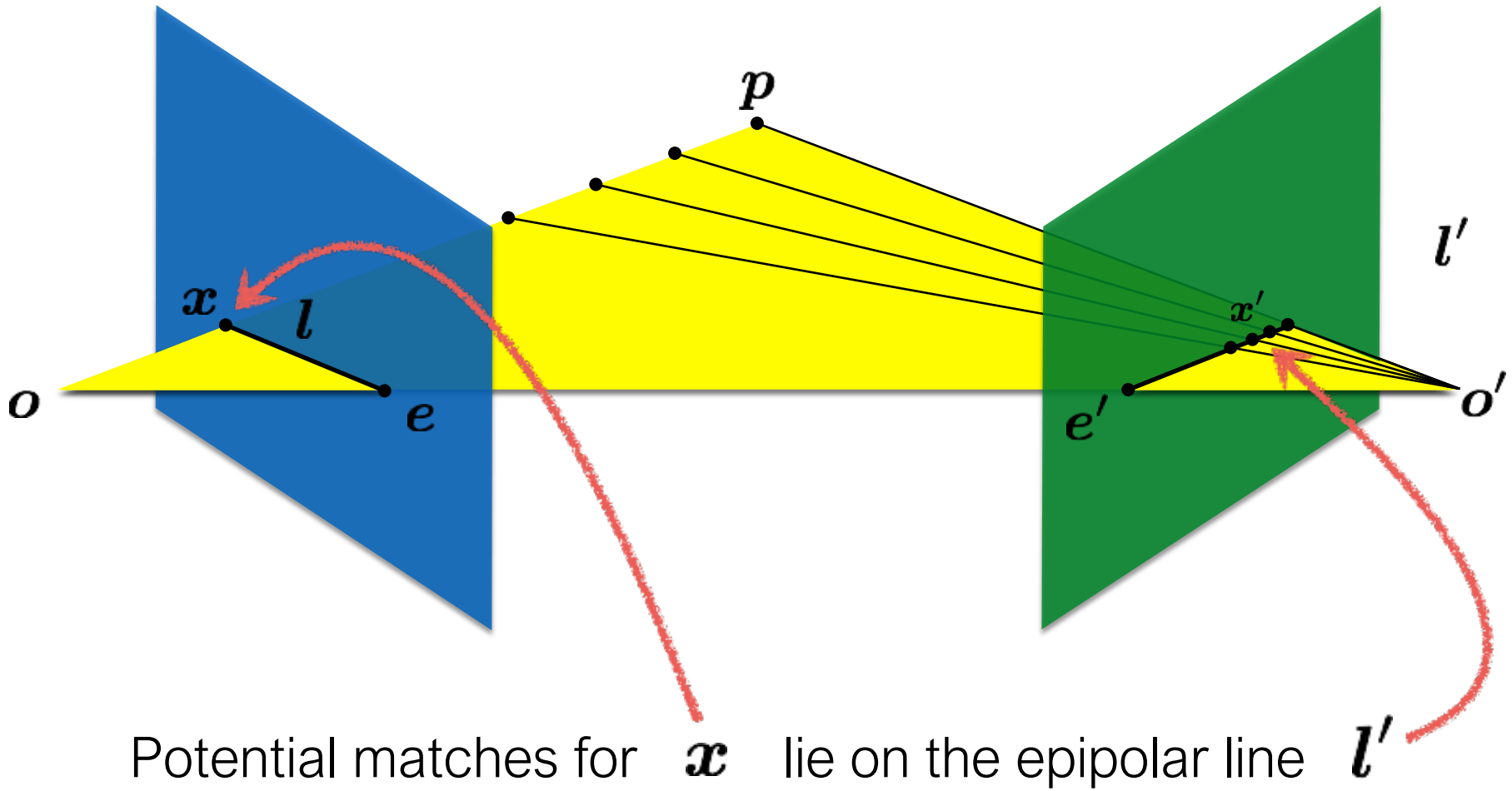
Left image

Right image

Epipolar constraint reduces search for matched point from 2D to a single line, if you know the epipolar line

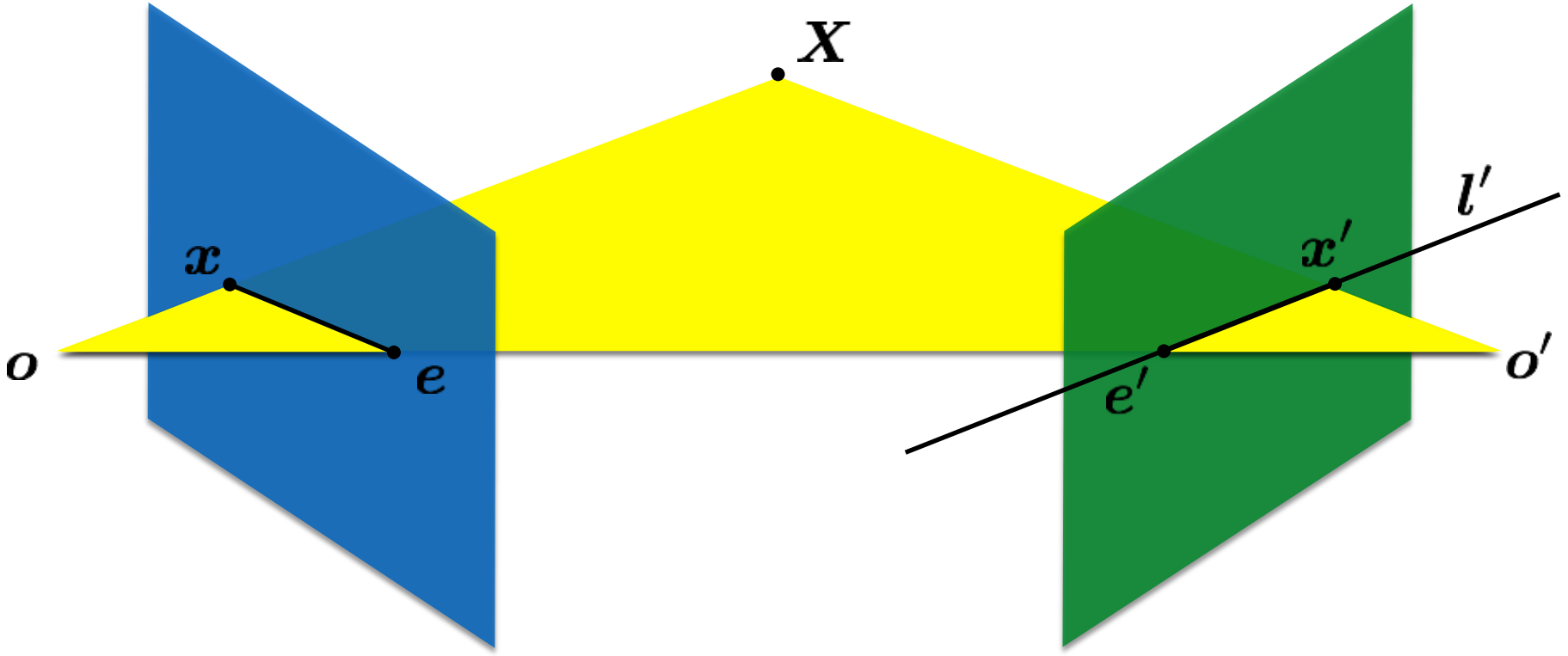
The essential matrix

Recall:Epipolar constraint



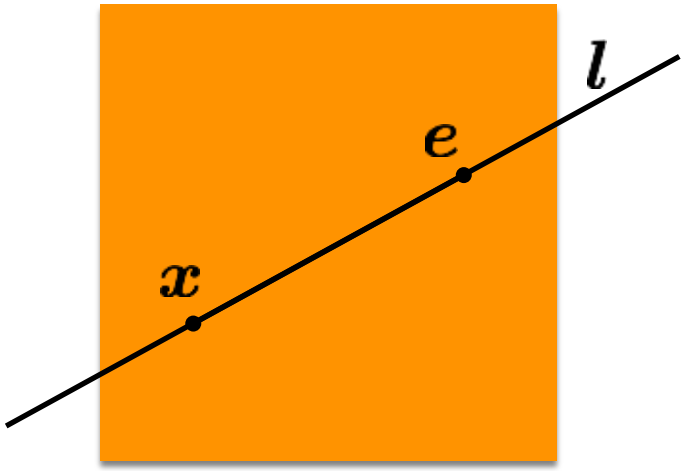
Given a point in one image,
multiplying by the **essential matrix** will tell us
the **epipolar line** in the second image.

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



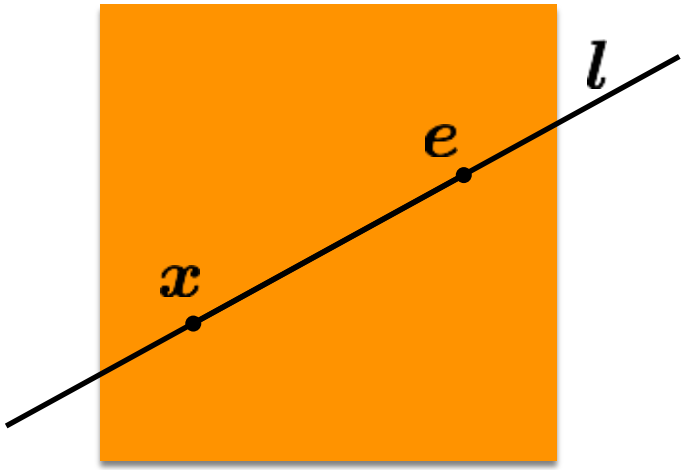
Epipolar Line

$$ax + by + c = 0$$



Epipolar Line

$$ax + by + c = 0$$ in vector form $$\boldsymbol{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

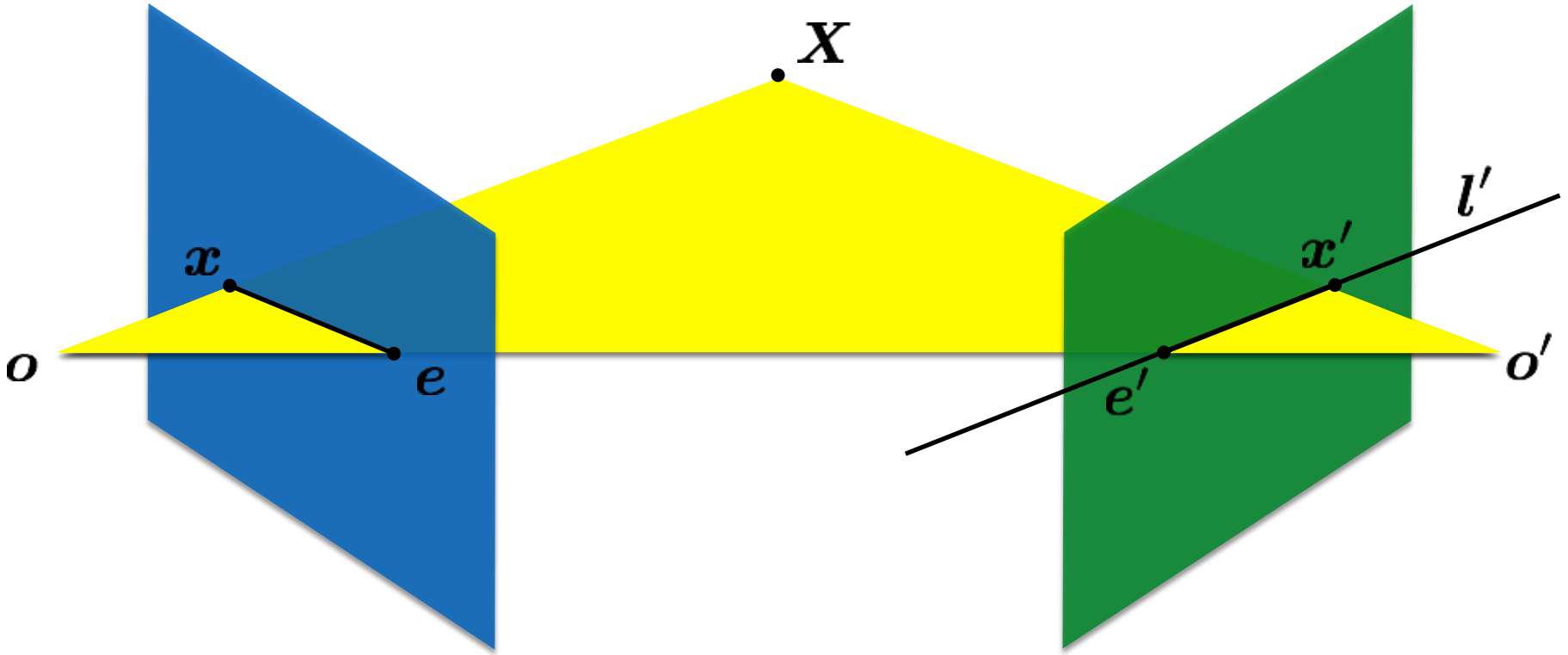


If the point $\boldsymbol{x}$ in homogeneous coordinates is on the epipolar line $\boldsymbol{l}$ then

$$\boldsymbol{x}^\top \boldsymbol{l} = 0$$

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

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



Essential Matrix vs Homography

They are both 3 x 3 matrices but …

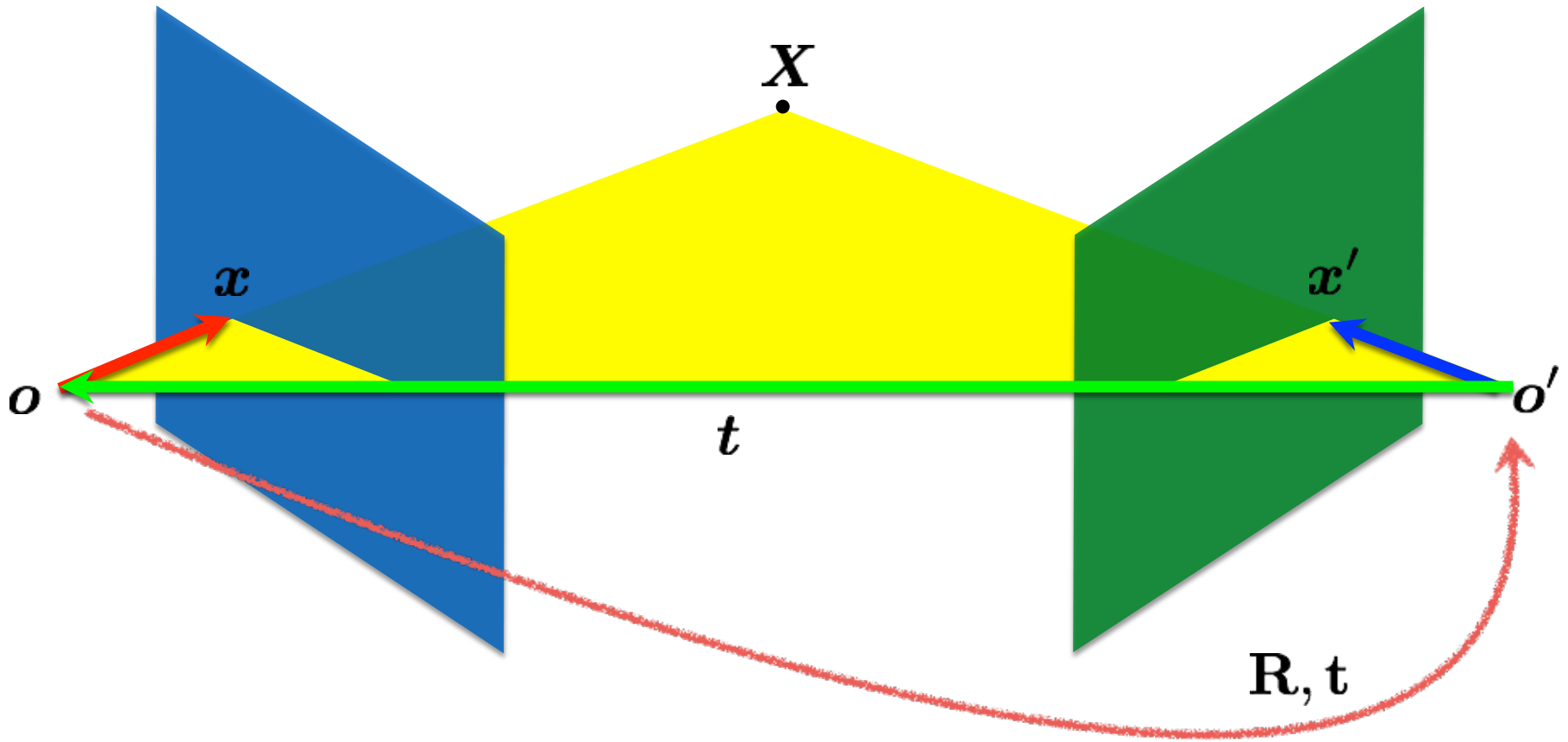
$$\boldsymbol{l}' = \mathbf{E}\boldsymbol{x}$$

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

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

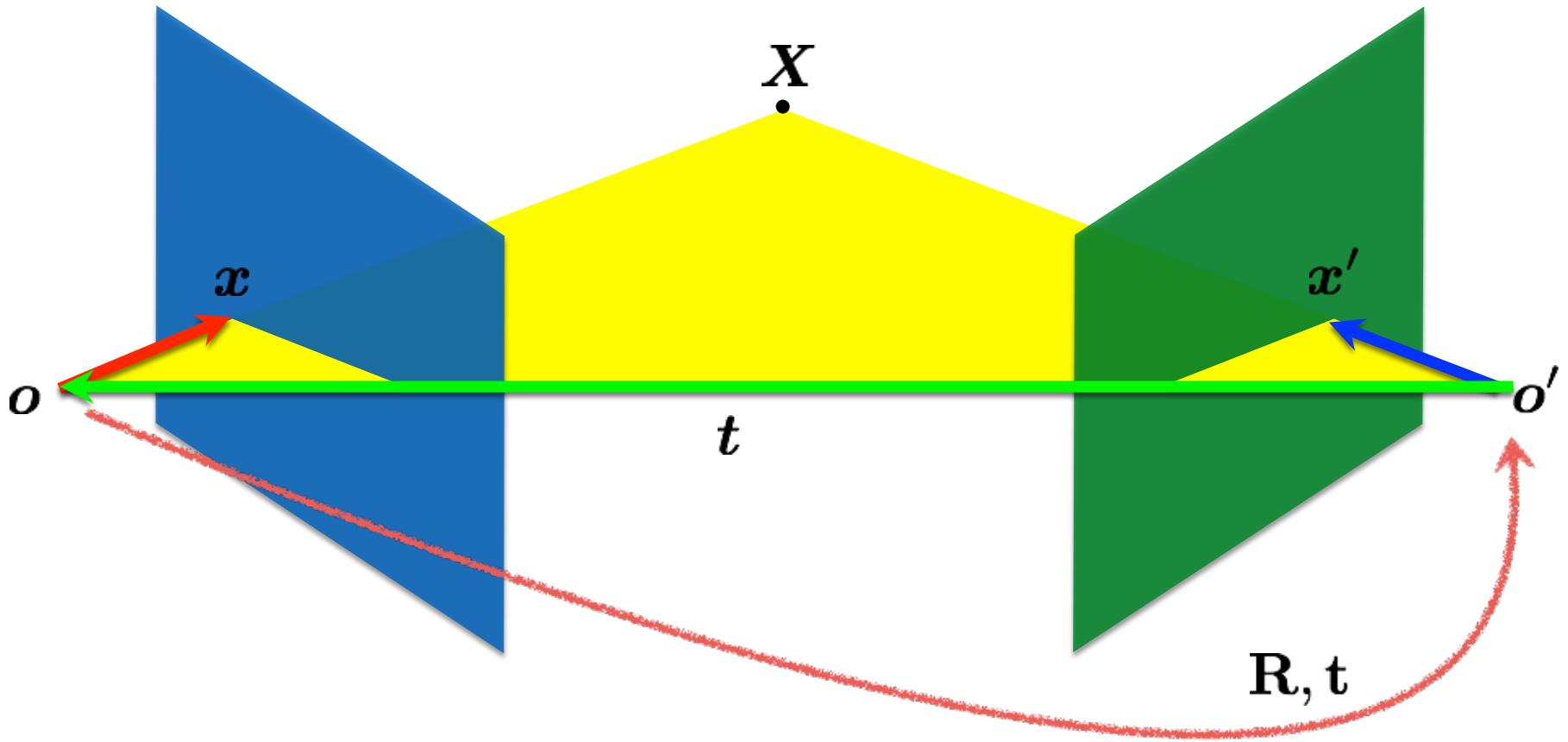
Homography maps a **point** to a **point**

Where does the essential matrix come from?



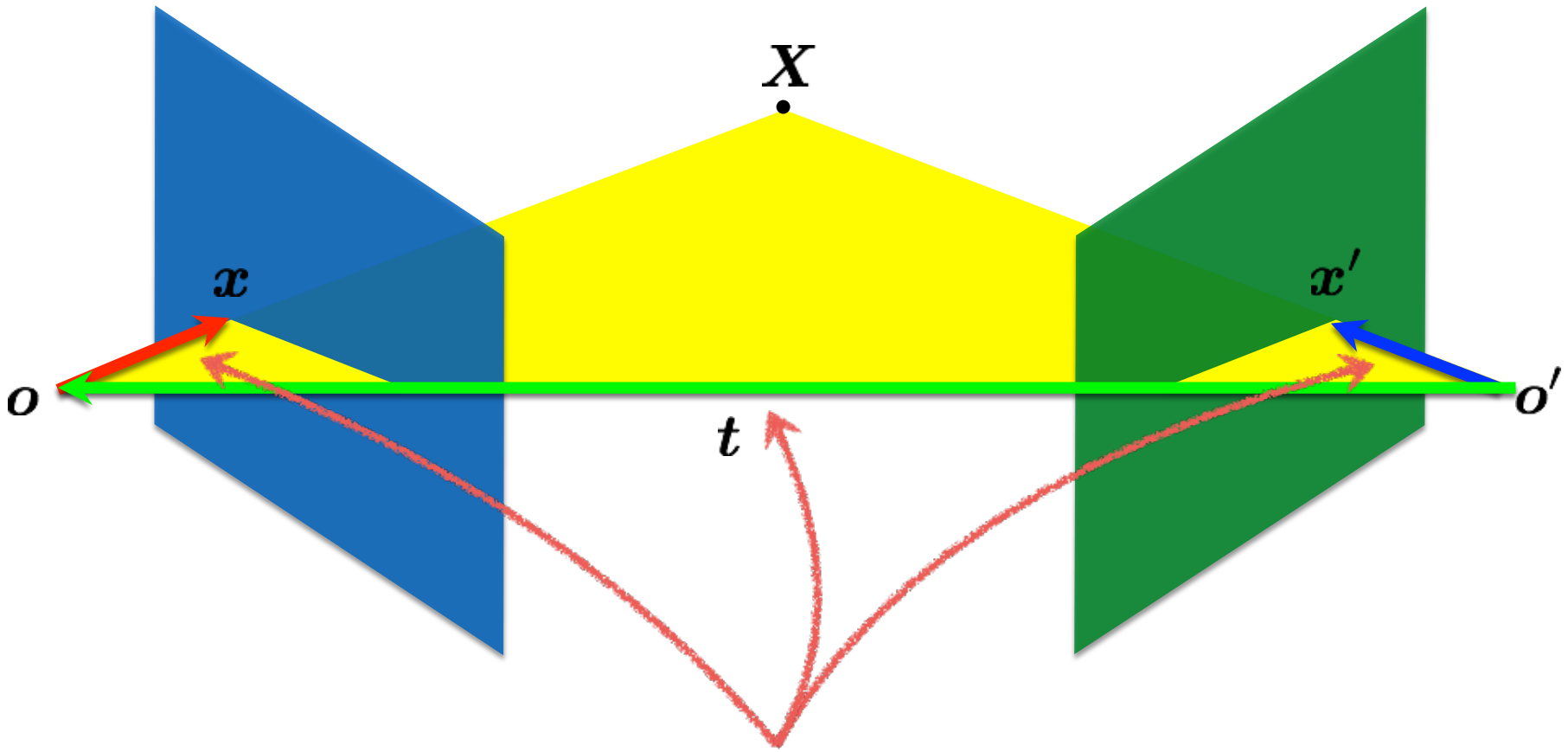
Assumption:

2D points expressed in camera coordinate system (i.e., intrinsic matrices are identities)



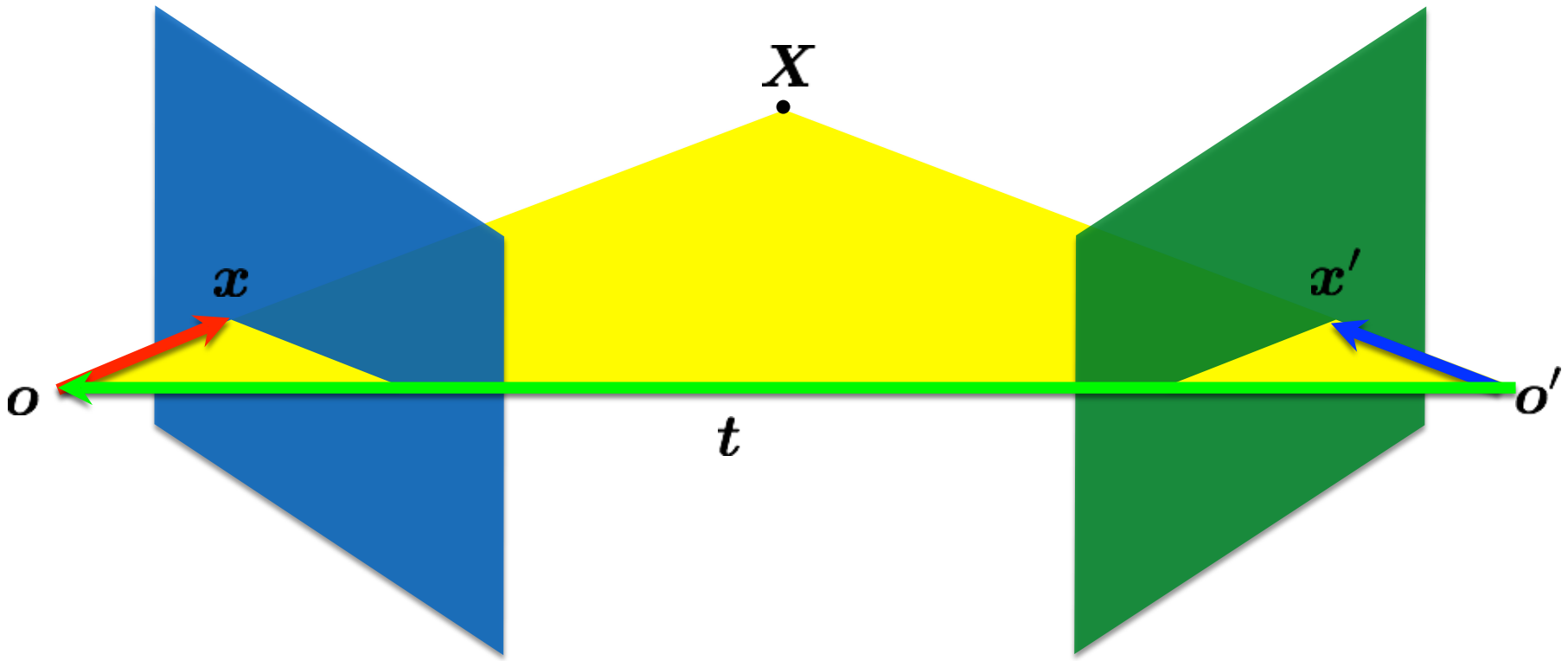
$$x' = \mathbf{R}(x - t)$$

Camera-camera transform just like **world-camera** transform



These three vectors are coplanar

$$x, t, x'$$

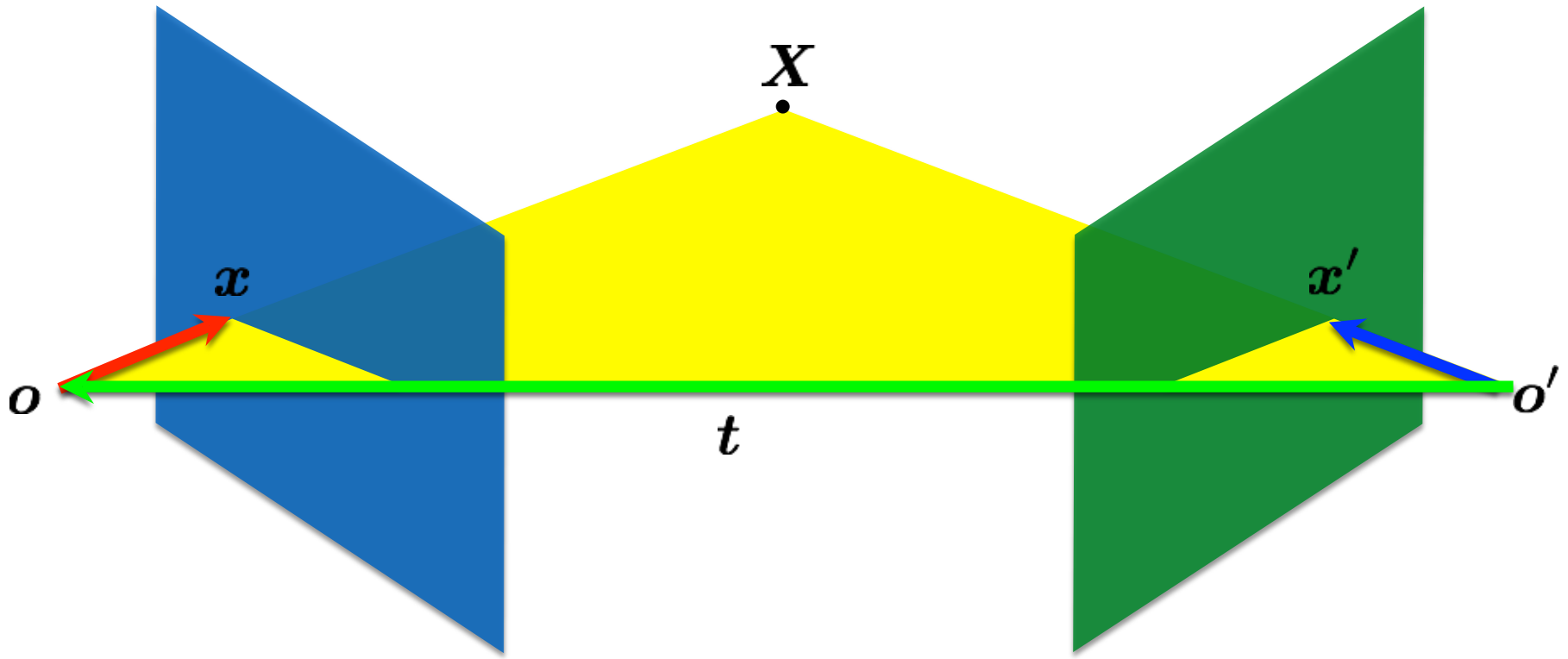


If these three vectors are coplanar $\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{x}'$ then

$$\boldsymbol{x}^\top (\boldsymbol{t} \times \boldsymbol{x}) = 0$$

dot product of orthogonal vectors

cross-product: vector orthogonal to plane



If these three vectors are coplanar $\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{x}'$ then

$$(\boldsymbol{x} - \boldsymbol{t})^\top (\boldsymbol{t} \times \boldsymbol{x}) = 0$$

putting it together

camera-camera transform

$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$

coplanarity

$$(\boldsymbol{x} - \boldsymbol{t})^\top (\boldsymbol{t} \times \boldsymbol{x}) = 0$$

putting it together

camera-camera transform

$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$

coplanarity

$$(\boldsymbol{x} - \boldsymbol{t})^\top (\boldsymbol{t} \times \boldsymbol{x}) = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})(\boldsymbol{t} \times \boldsymbol{x}) = 0$$

Linear algebra reminder: cross product

Cross product

$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

Can also be written as a matrix multiplication

$$\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_\times \boldsymbol{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Skew symmetric

putting it together

rigid motion

$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$

coplanarity

$$(\boldsymbol{x} - \boldsymbol{t})^\top (\boldsymbol{t} \times \boldsymbol{x}) = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})(\boldsymbol{t} \times \boldsymbol{x}) = 0$$

use skew-symmetric matrix to represent cross product

$$(\boldsymbol{x}'^\top \mathbf{R})([\mathbf{t}_\times]\boldsymbol{x}) = 0$$

putting it together

rigid motion

$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$

coplanarity

$$(\boldsymbol{x} - \boldsymbol{t})^\top (\boldsymbol{t} \times \boldsymbol{x}) = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})(\boldsymbol{t} \times \boldsymbol{x}) = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})([\mathbf{t}_\times]\boldsymbol{x}) = 0$$

$$\boldsymbol{x}'^\top (\mathbf{R}[\mathbf{t}_\times])\boldsymbol{x} = 0$$

putting it together

rigid motion

$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$

coplanarity

$$(\boldsymbol{x} - \boldsymbol{t})^\top (\boldsymbol{t} \times \boldsymbol{x}) = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})(\boldsymbol{t} \times \boldsymbol{x}) = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})([\mathbf{t}_\times]\boldsymbol{x}) = 0$$

$$\boldsymbol{x}'^\top (\mathbf{R}[\mathbf{t}_\times])\boldsymbol{x} = 0$$

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

Essential Matrix
[Longuet-Higgins 1981]

properties of the E matrix

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

(2D points expressed in camera coordinate system)

properties of the E matrix

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

Epipolar lines

$\boldsymbol{x}^{\top}\boldsymbol{l} = 0$ $\quad$ $\boldsymbol{x}'^{\top}\boldsymbol{l}' = 0$

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

(2D points expressed in camera coordinate system)

properties of the E matrix

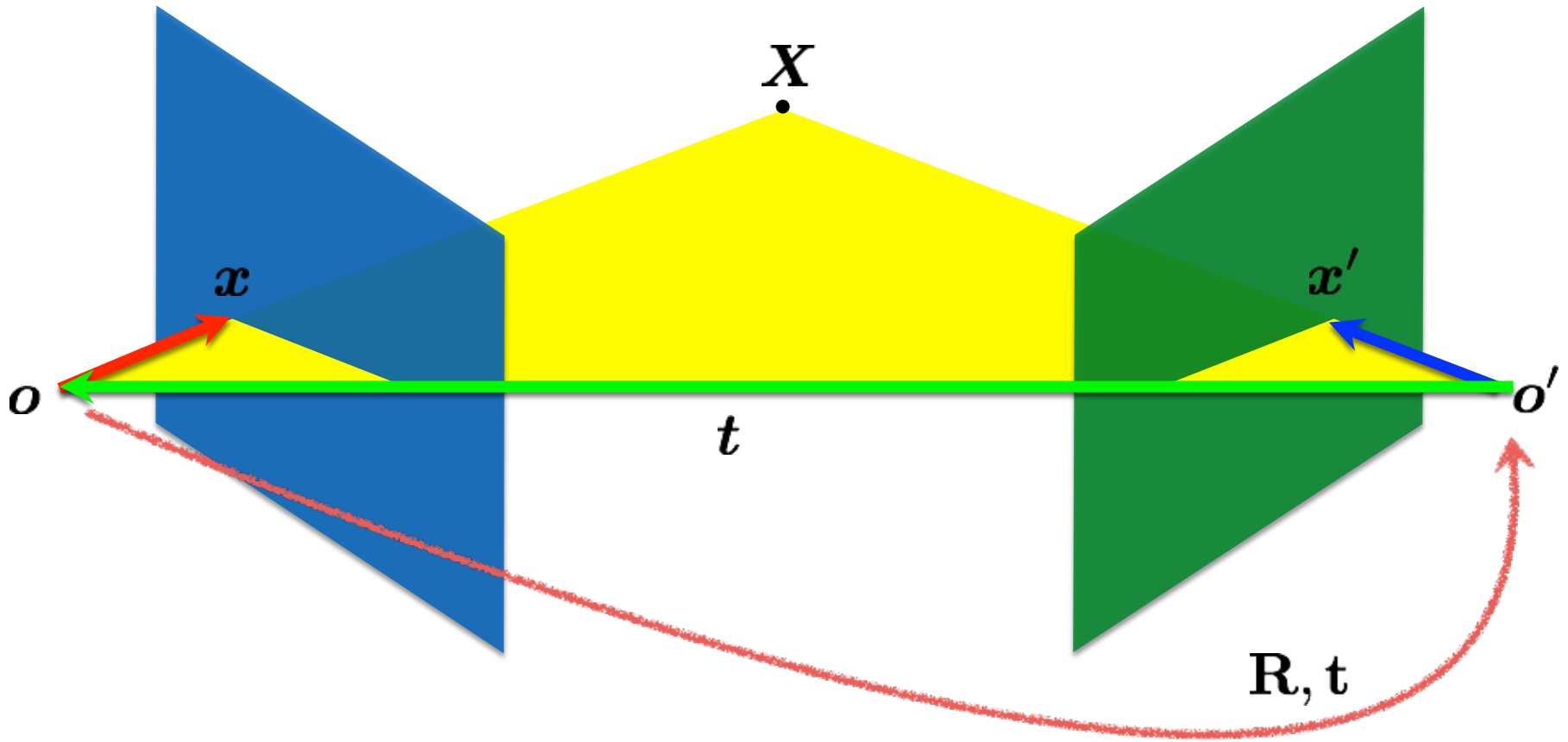
Longuet-Higgins equation $\boldsymbol{x'}^{\top}\mathbf{E}\boldsymbol{x} = 0$

Epipolar lines $\boldsymbol{x}^{\top}\boldsymbol{l} = 0$ $\quad \boldsymbol{x'}^{\top}\boldsymbol{l'} = 0$

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

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

(2D points expressed in camera coordinate system)



Assumption:

2D points expressed in camera coordinate system (i.e., intrinsic matrices are identities)

The fundamental matrix

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

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

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

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

camera point

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

image point

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

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

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

camera point — image point

Writing out the epipolar constraint in terms of image coordinates

$$\mathbf{K}'^{-\top}\mathbf{E}\mathbf{K}^{-1}\boldsymbol{x} = 0$$

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

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

Same equation works in image coordinates!

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

properties of the F/E matrix

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

Epipolar lines

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

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

Epipoles $\boldsymbol{e'}^{\top}\mathbf{F} = \mathbf{0}$ $\qquad$ $\mathbf{F}\boldsymbol{e} = \mathbf{0}$

(2D points expressed in image coordinate system)

Breaking down the fundamental matrix

$$\mathbf{F} = \mathbf{K}'^{-\top}\mathbf{E}\mathbf{K}^{-1}$$

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

Depends on both intrinsic and extrinsic parameters

The 8-point algorithm

Assume you have ***M*** matched *image* points

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

Each correspondence should satisfy

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

Solve for the 3 x 3 fundamental matrix **F**

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

Write out the fundamental matrix property

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

Write out the fundamental matrix property

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

Replace coordinate vectors and the fundamental matrix

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

Write out the fundamental matrix property

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

Replace coordinate vectors and the fundamental matrix

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

Expand the matrix-vector multiplications

Stack equations from all point pairs to form a homogeneous linear system

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx'_M & x_My'_M & x_M & y_Mx'_M & y_My'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

Stack equations from all point pairs to form a homogeneous linear system

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx'_M & x_My'_M & x_M & y_Mx'_M & y_My'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

Solve homogeneous linear system using SVD

Stack equations from all point pairs to form a homogeneous linear system

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx'_M & x_My'_M & x_M & y_Mx'_M & y_My'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

We have 8 degrees of freedom (fundamental matrix is defined up to scale)

Stack equations from all point pairs to form a homogeneous linear system

Each point pair provides one constraint

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx'_M & x_My'_M & x_M & y_Mx'_M & y_My'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

We have 8 degrees of freedom (fundamental matrix is defined up to scale)

Stack equations from all point pairs to form a homogeneous linear system

Each point pair provides one constraint

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx'_M & x_My'_M & x_M & y_Mx'_M & y_My'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

We have 8 degrees of freedom (fundamental matrix is defined up to scale)

We need at least 8 points, hence the **8-point algorithm!**

Eight-Point Algorithm

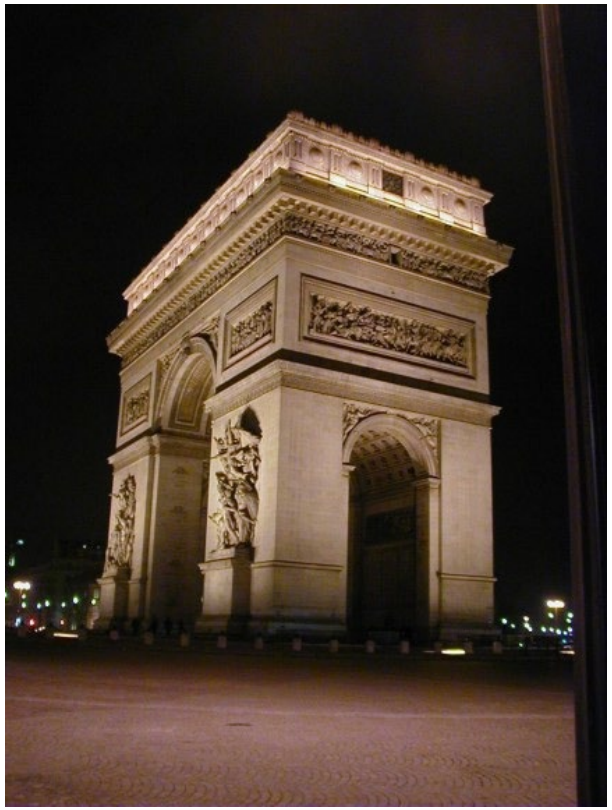
0. (Normalize points)
1. Construct the M x 9 matrix **A**
2. Find the SVD of **A**
3. Entries of **F** are the elements of column of **V** corresponding to the least singular value
4. (Enforce rank 2 constraint on F)
5. (Un-normalize F)

Eight-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)

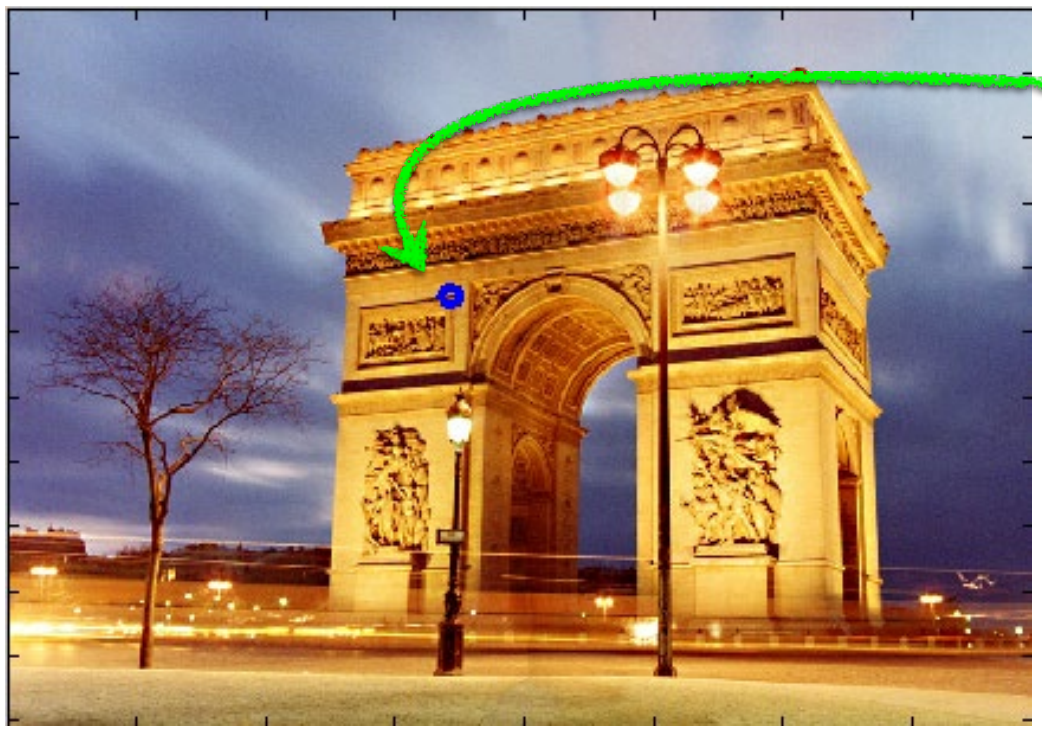
Use SVD

Example



epipolar lines

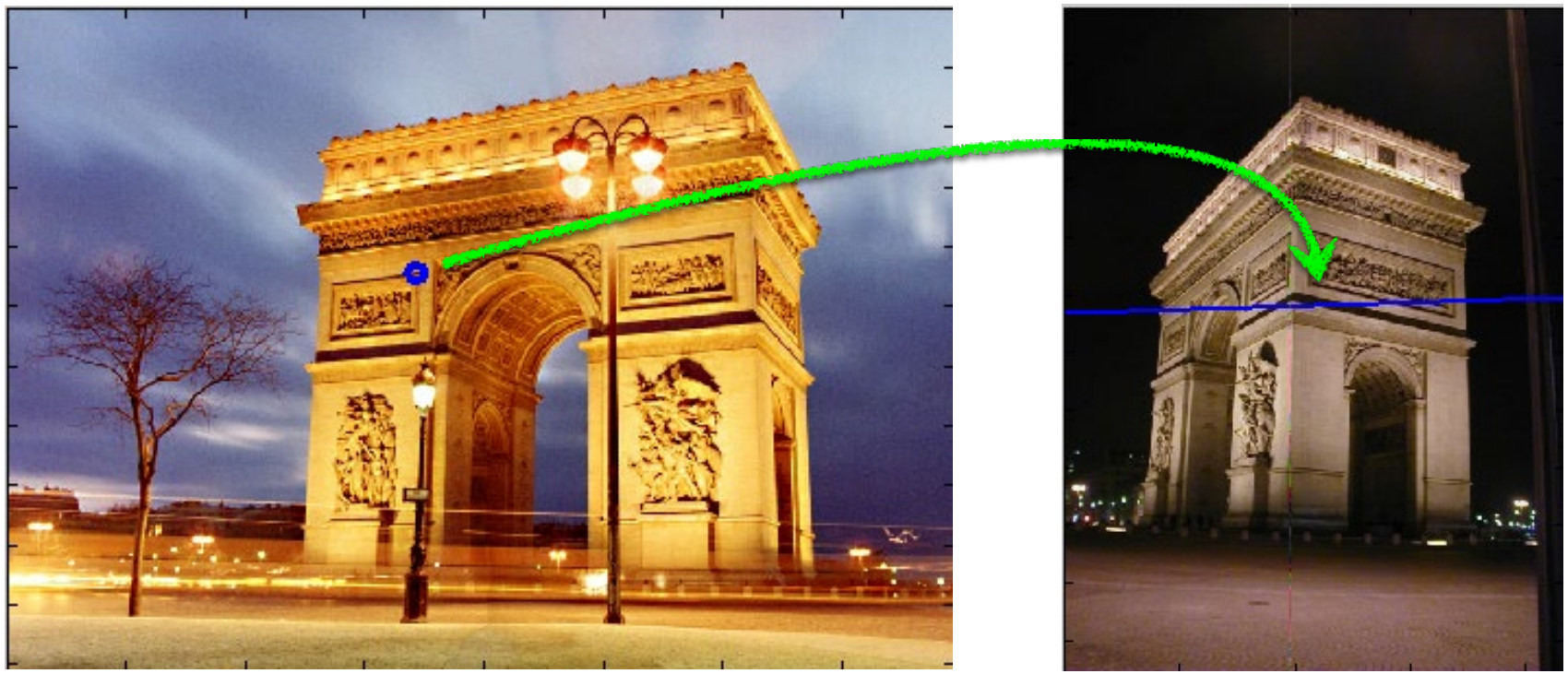
$$\mathbf{F} = \begin{bmatrix} -0.00310695 & -0.0025646 & 2.96584 \\ -0.028094 & -0.00771621 & 56.3813 \\ 13.1905 & -29.2007 & -9999.79 \end{bmatrix}$$



$$\boldsymbol{x} = \begin{bmatrix} 343.53 \\ 221.70 \\ 1.0 \end{bmatrix}$$

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

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



Epipolar lines intersect at the epipole

Epipolar lines intersect at the epipole

The epipole satisfies the equation

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

Epipolar lines intersect at the epipole

The epipole satisfies the equation

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

We can compute the epipole using SVD on **F**

Eight-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)

Normalization

Problem with 8-point algorithm

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Problem with 8-point algorithm

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

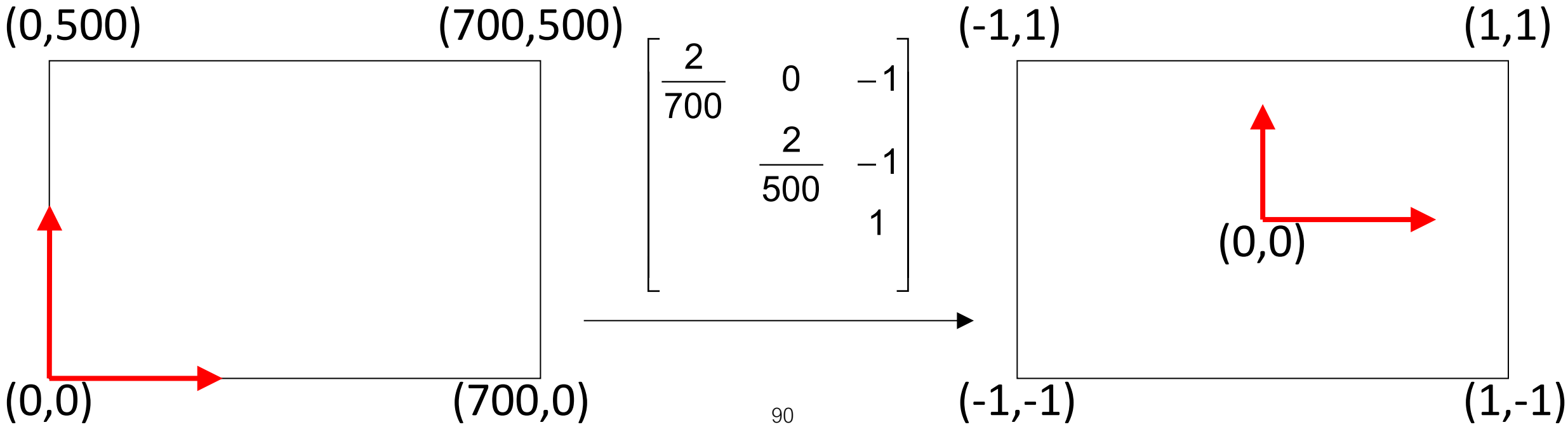
~10000 ~10000 ~100 ~10000 ~10000 ~100 ~100 ~100 1

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

Normalized 8-point algorithm

Normalized least squares yields good results

Transform image to ~[-1,1]x[-1,1]



Normalized 8-point algorithm

1. Transform input by $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{T}\mathbf{x}_\mathbf{i}$, $\hat{\mathbf{x}}'_\mathbf{i} = \mathbf{T}\mathbf{x}'_\mathbf{i}$
2. Call 8-point on $\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i}$ to obtain $\hat{\mathbf{F}}$
3. Obtain final fundamental matrix as $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

Normalized 8-point algorithm

1. Transform input by $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{T}\mathbf{x}_\mathbf{i}$, $\hat{\mathbf{x}}'_\mathbf{i} = \mathbf{T}\mathbf{x}'_\mathbf{i}$
2. Call 8-point on $\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i}$ to obtain $\hat{\mathbf{F}}$
3. Obtain final fundamental matrix as $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

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

Normalized 8-point algorithm

1. Transform input by $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{T}\mathbf{x}_\mathbf{i}$, $\hat{\mathbf{x}}'_\mathbf{i} = \mathbf{T}\mathbf{x}'_\mathbf{i}$
2. Call 8-point on $\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i}$ to obtain $\hat{\mathbf{F}}$
3. Obtain final fundamental matrix as $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0 \qquad \hat{\mathbf{x}}'^{\mathrm{T}}\mathbf{T}'^{-\mathrm{T}}\mathbf{F}\mathbf{T}^{-1}\hat{\mathbf{x}} = 0$$

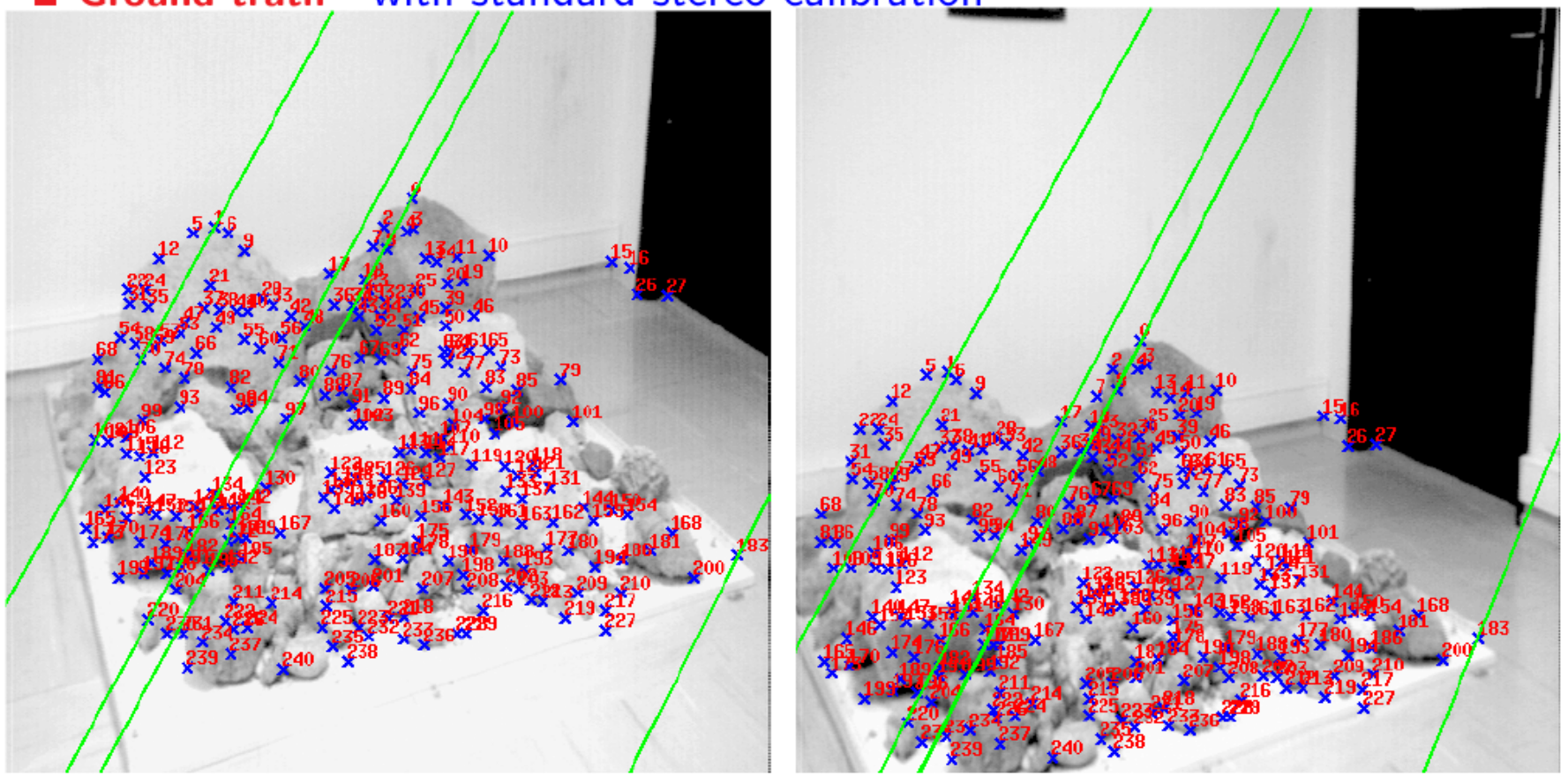
Normalized 8-point algorithm

1. Transform input by $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{T}\mathbf{x}_\mathbf{i}$, $\hat{\mathbf{x}}'_\mathbf{i} = \mathbf{T}\mathbf{x}'_\mathbf{i}$
2. Call 8-point on $\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i}$ to obtain $\hat{\mathbf{F}}$
3. Obtain final fundamental matrix as $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0 \qquad \hat{\mathbf{x}}'^{\mathrm{T}}\underbrace{\mathbf{T}'^{-\mathrm{T}}\mathbf{F}\mathbf{T}^{-1}}_{\hat{\mathbf{F}}}\hat{\mathbf{x}} = 0$$

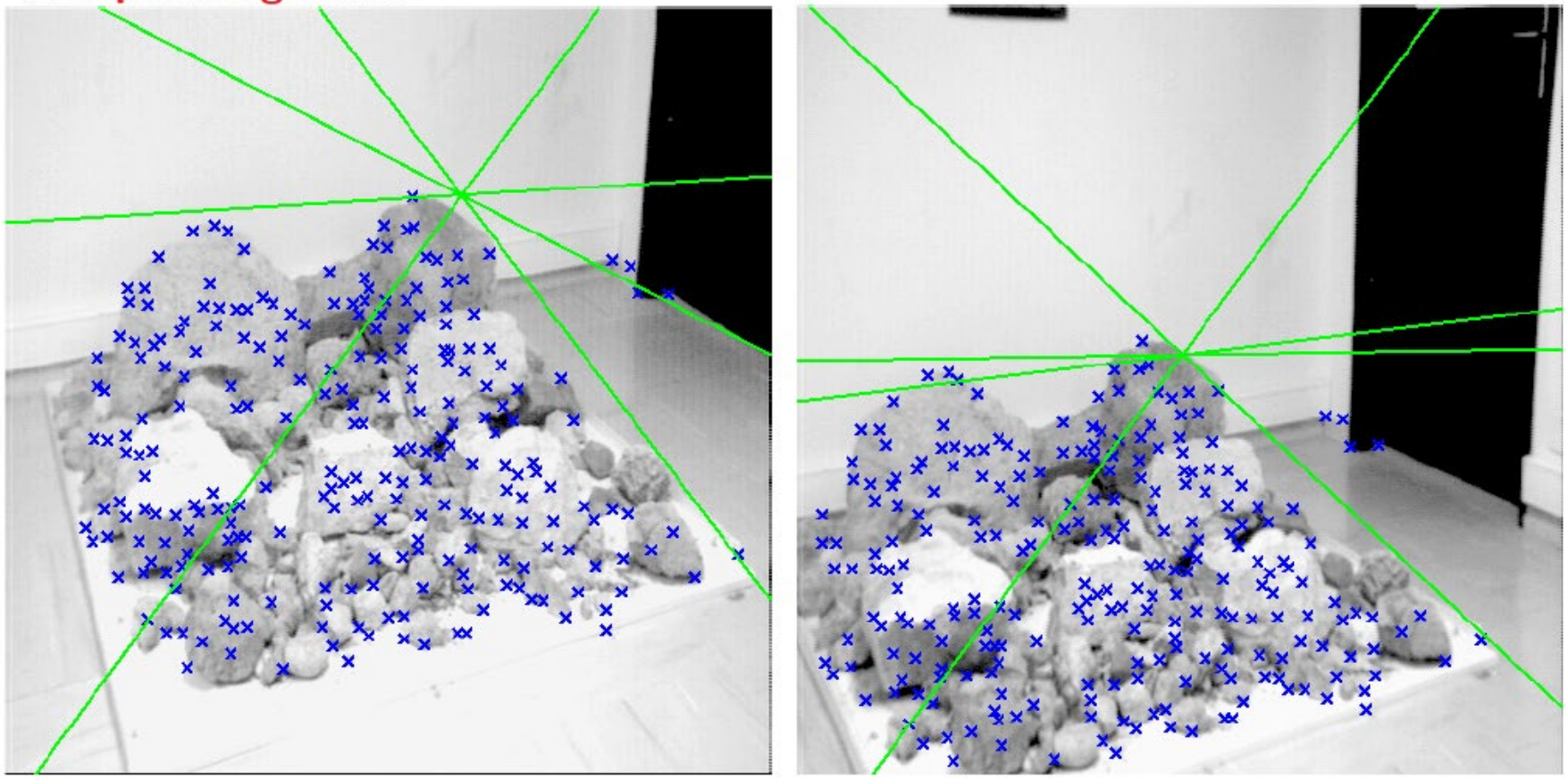
An example: groundtruth

■ **Ground truth** with standard stereo calibration



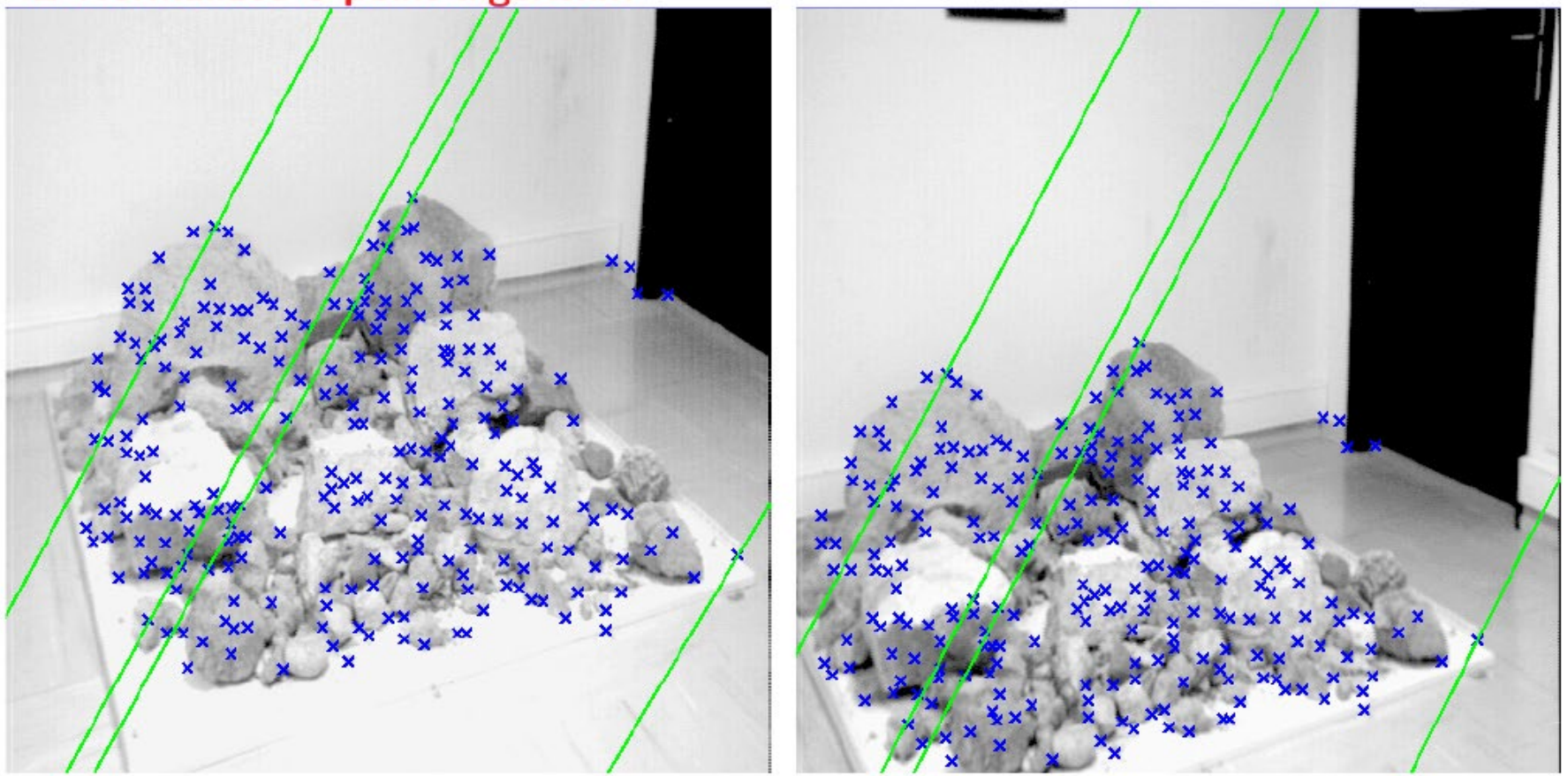
An example: no normalization

- 8-point algorithm



An example: normalization

- Normalized 8-point algorithm



Normalized 8-point algorithm

1. Transform input by $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{T}\mathbf{x}_\mathbf{i}$, $\hat{\mathbf{x}}'_\mathbf{i} = \mathbf{T}\mathbf{x}'_\mathbf{i}$
2. Call 8-point on $\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}'_\mathbf{i}$ to obtain $\hat{\mathbf{F}}$
3. Obtain final fundamental matrix as $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

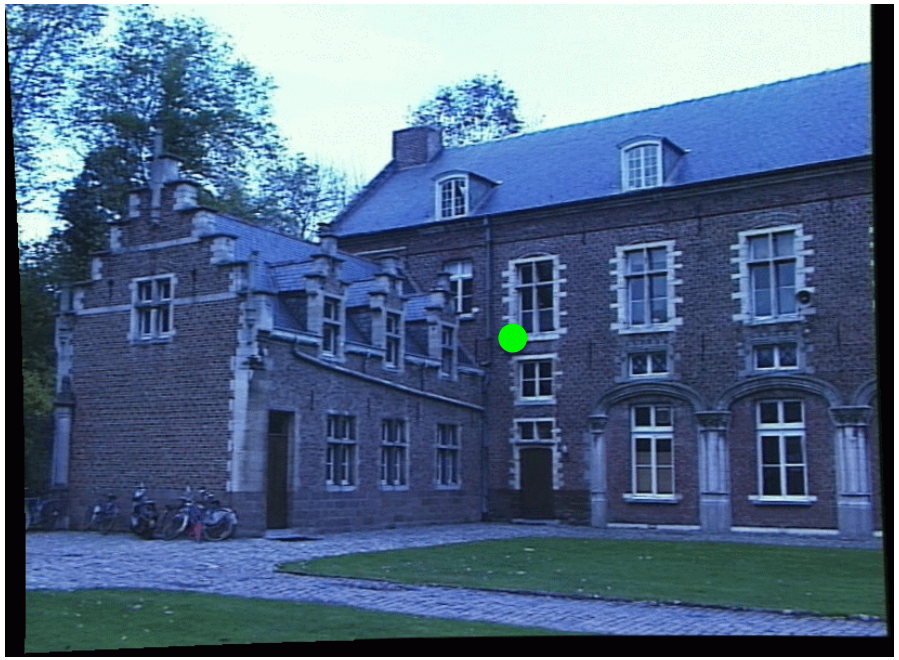
Normalization procedure can benefit other geometry-based vision routines (e.g., DLT for homography estimation, camera calibration)

Recap of triangulation for 3D reconstruction

Left image

Right image

Recap of triangulation for 3D reconstruction

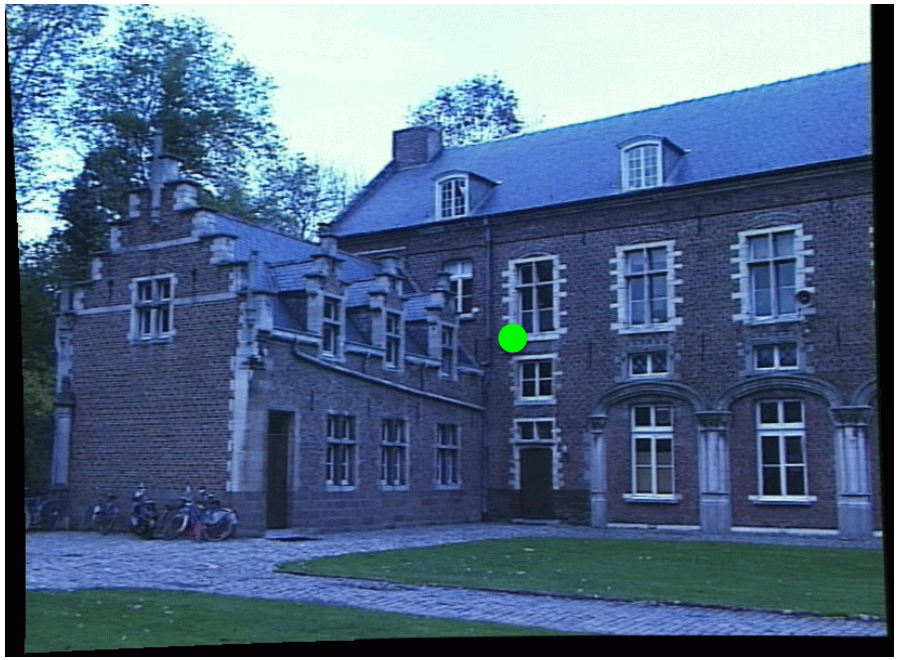


Left image

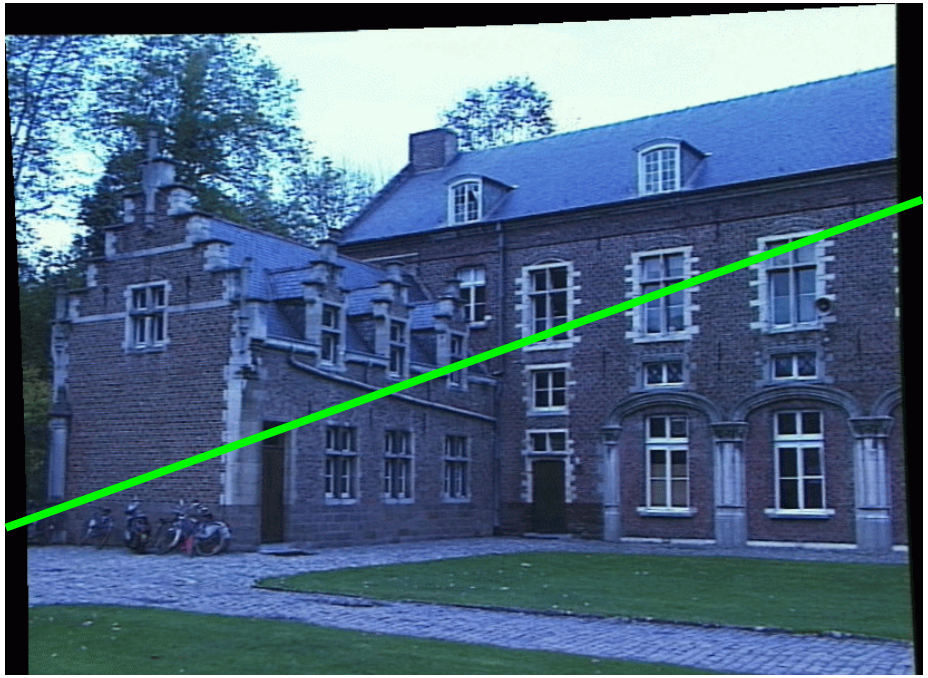
Right image

1. Select point in one image (e.g., using Harris corner detector)

Recap of triangulation for 3D reconstruction



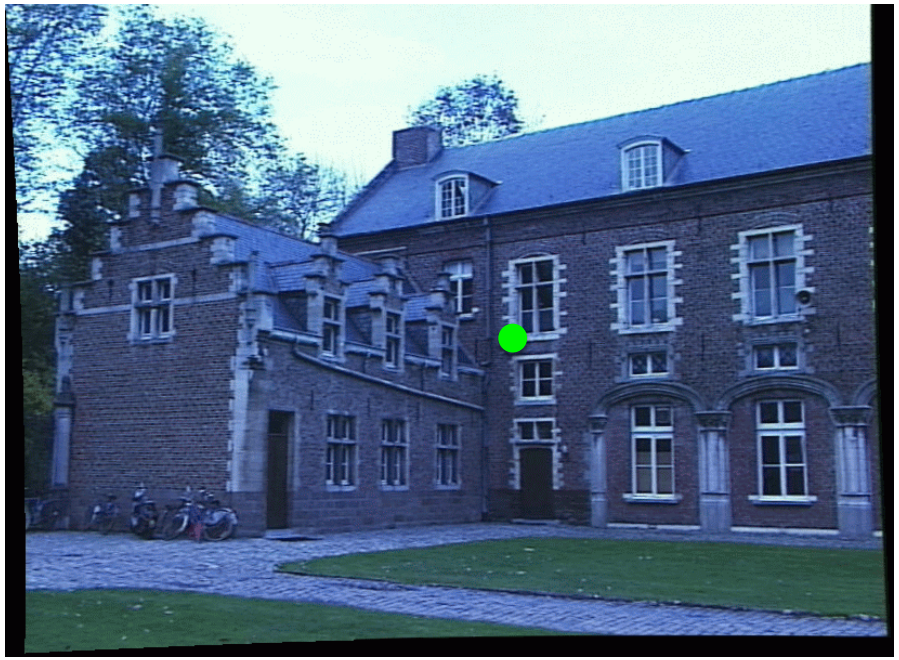
Left image



Right image

1. Select point in one image (e.g., using Harris corner detector)
2. Form epipolar line for that point in second image using fundamental matrix

Recap of triangulation for 3D reconstruction

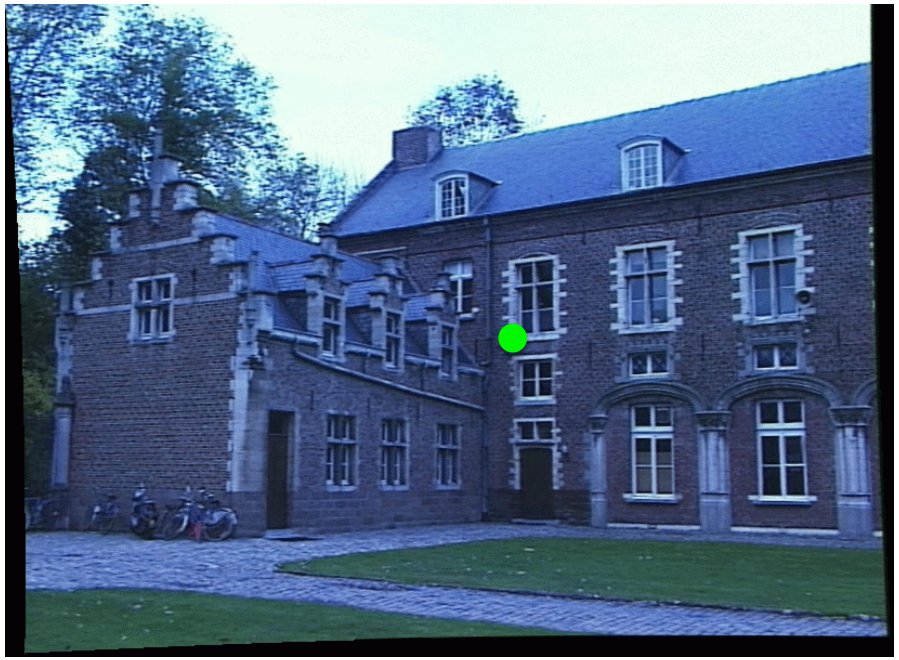


Left image

Right image

1. Select point in one image (e.g., using Harris corner detector)
2. Form epipolar line for that point in second image using fundamental matrix
3. Find matching point along line (e.g., using feature SIFT descriptor at all points along line)

Recap of triangulation for 3D reconstruction



Left image

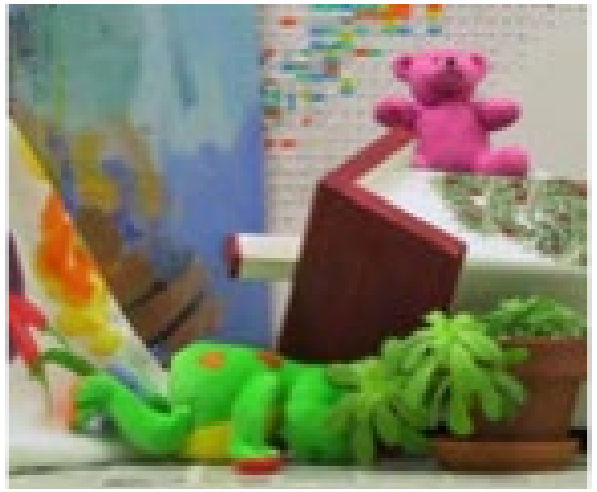
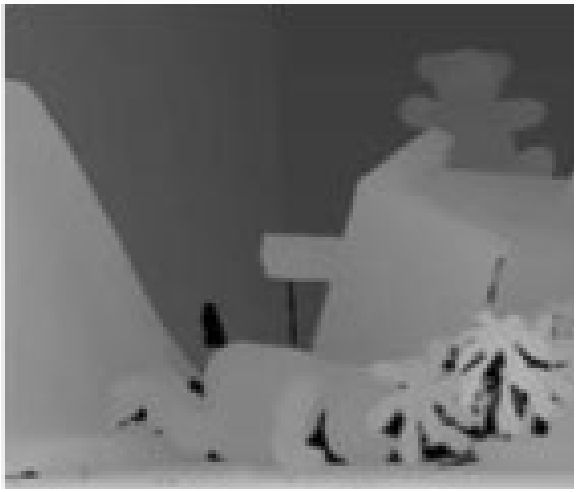
Right image

1. Select point in one image (e.g., using Harris corner detector)
2. Form epipolar line for that point in second image using fundamental matrix
3. Find matching point along line (e.g., using feature SIFT descriptor at all points along line)
4. Perform triangulation

What is disparity

Objects that are closer move more between the views

The amount of horizontal movement is inversely related to …

… the distance from the camera.

Real-time stereo sensing

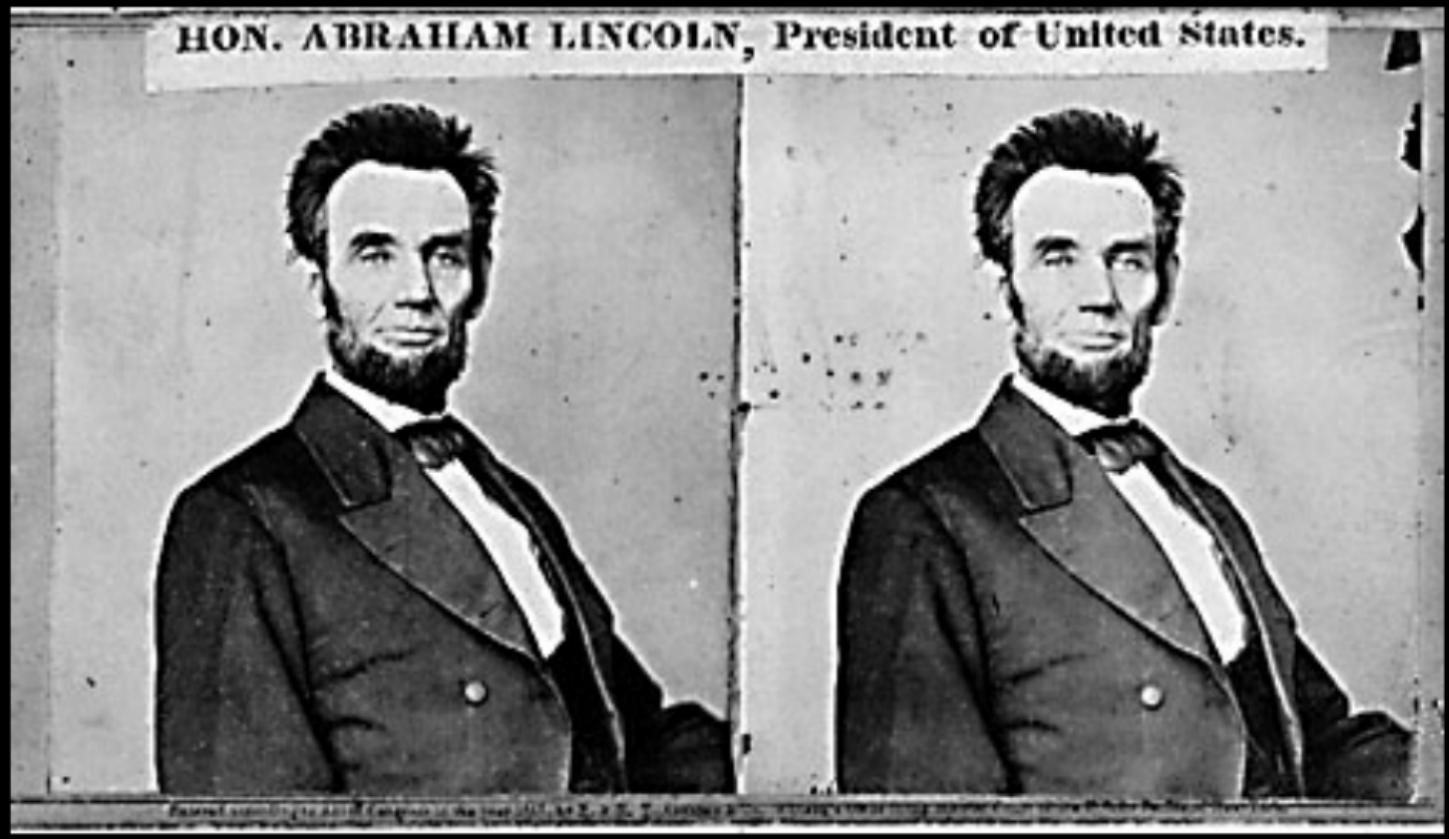
Nomad robot searches for meteorites in Antartica

http://www.frc.ri.cmu.edu/projects/meteorobot/index.html

Subaru
Eyesight system

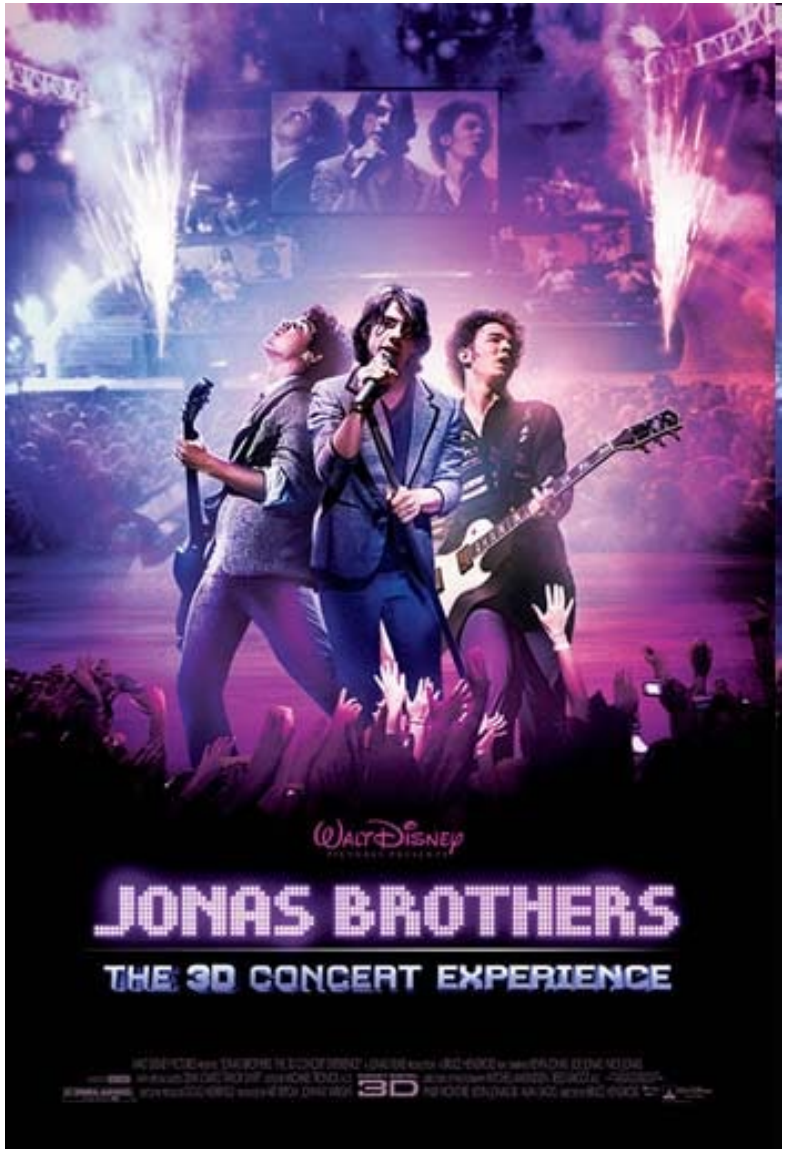
Pre-collision
braking

Stereoscopes: A 19th Century Pastime

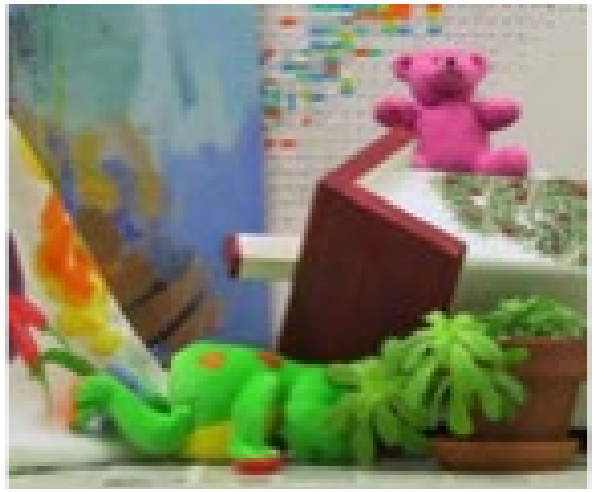
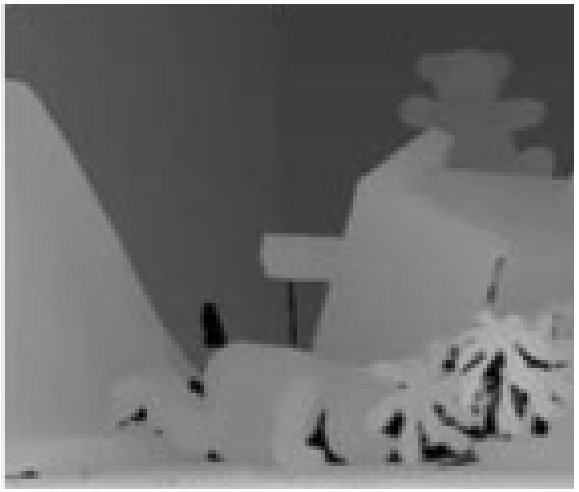

HON. ABRAHAM LINCOLN, President of United States.

Mark Twain at Pool Table", no date, UCR Museum of Photography

This is how 3D movies work

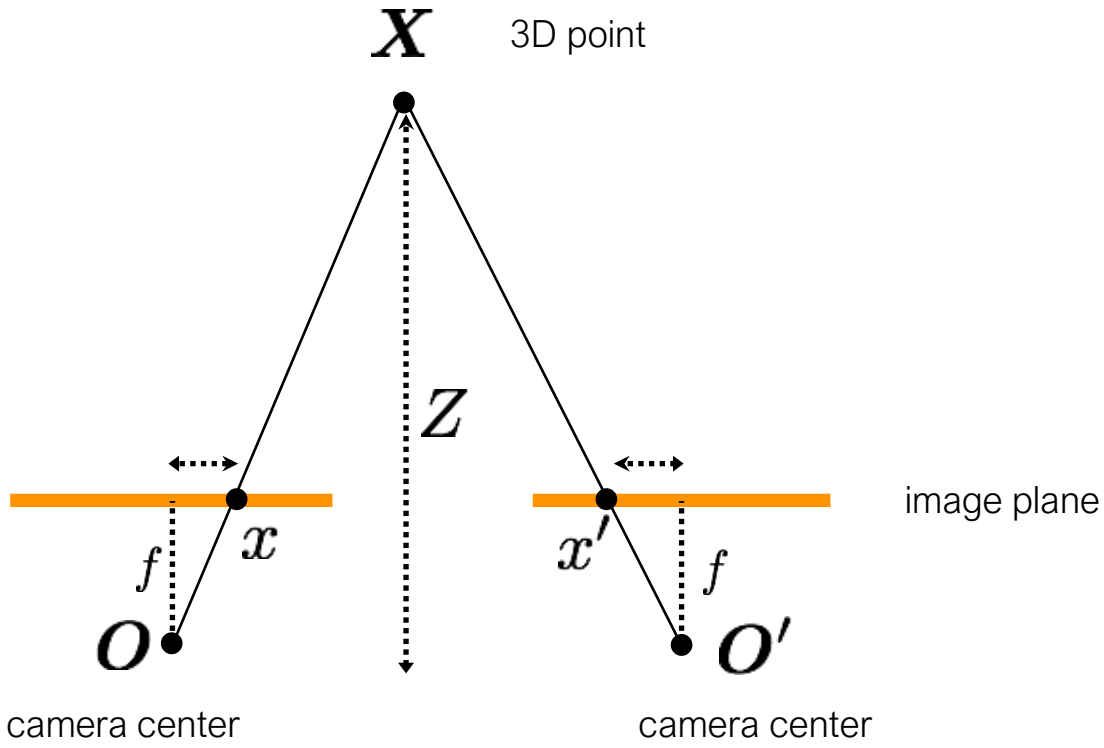


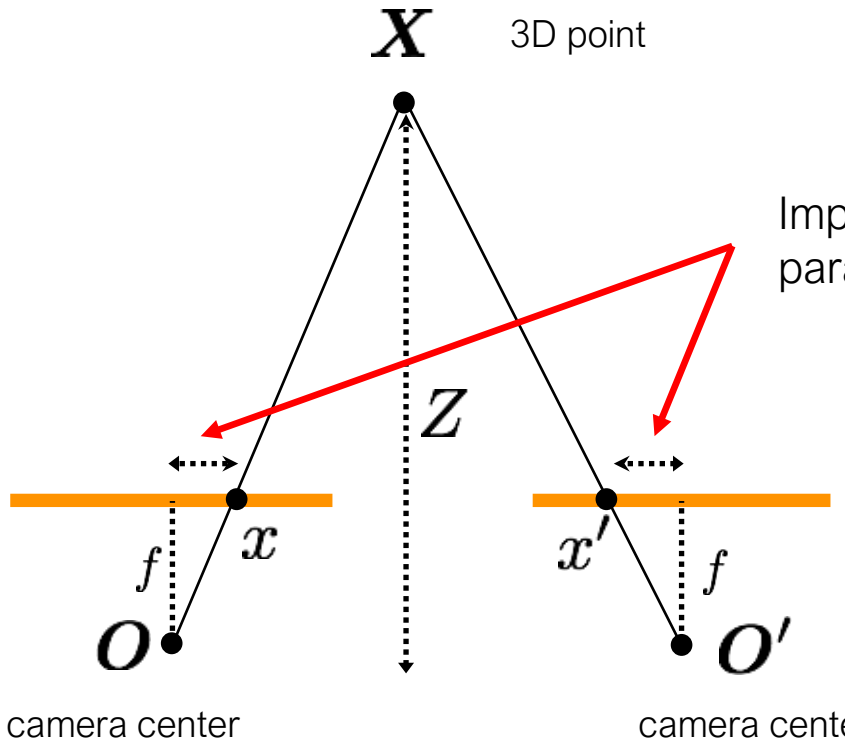
The amount of horizontal movement is inversely related to …

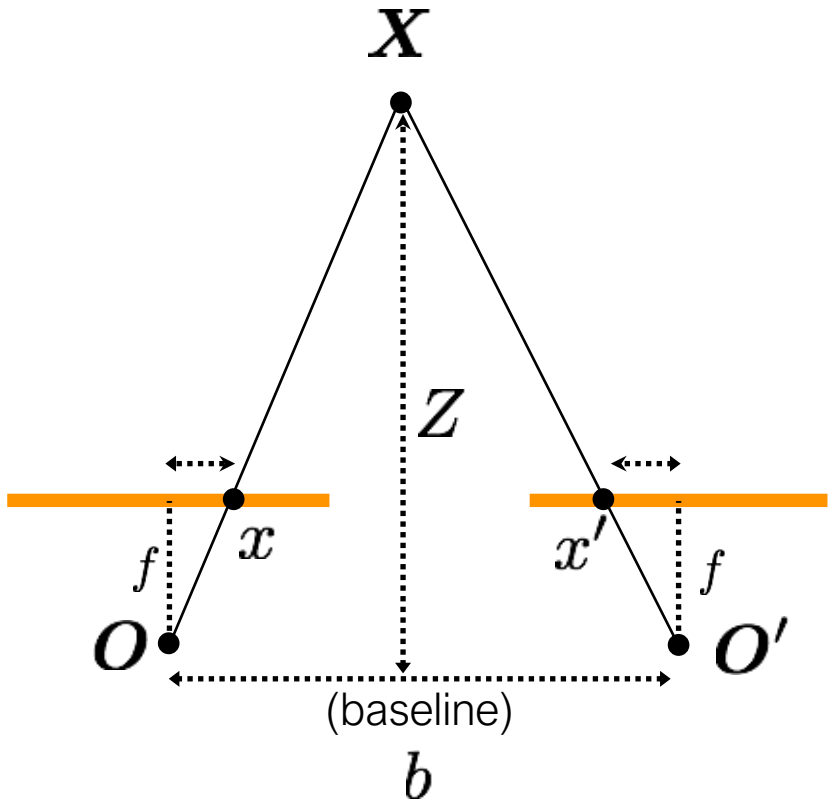
… the distance from the camera.

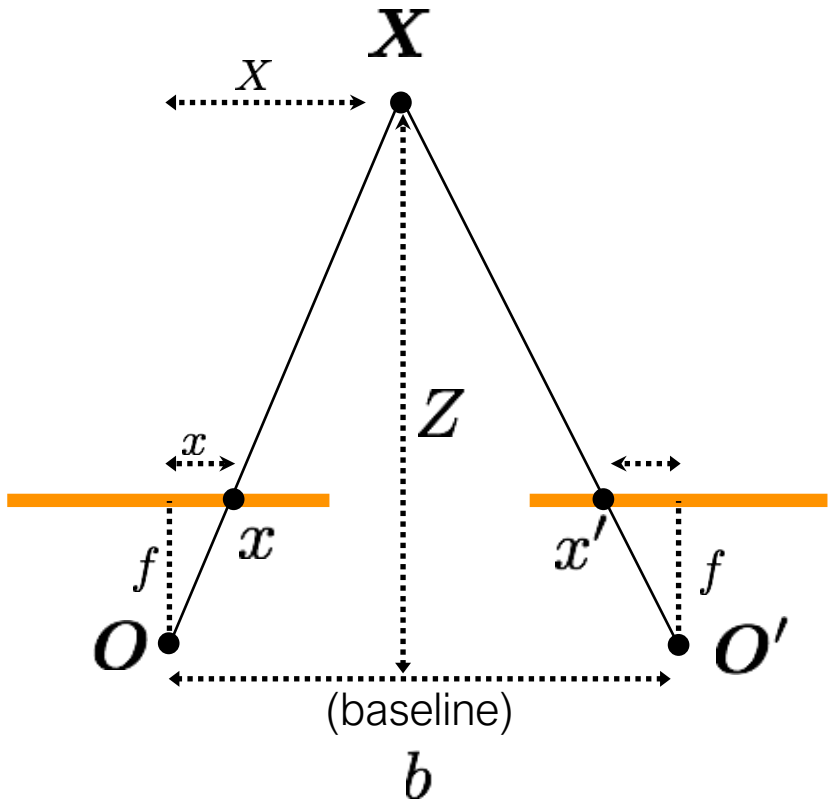
Depth from disparity


X
3D point
Z
image plane
f
x
x′
f
O
O′
camera center
camera center

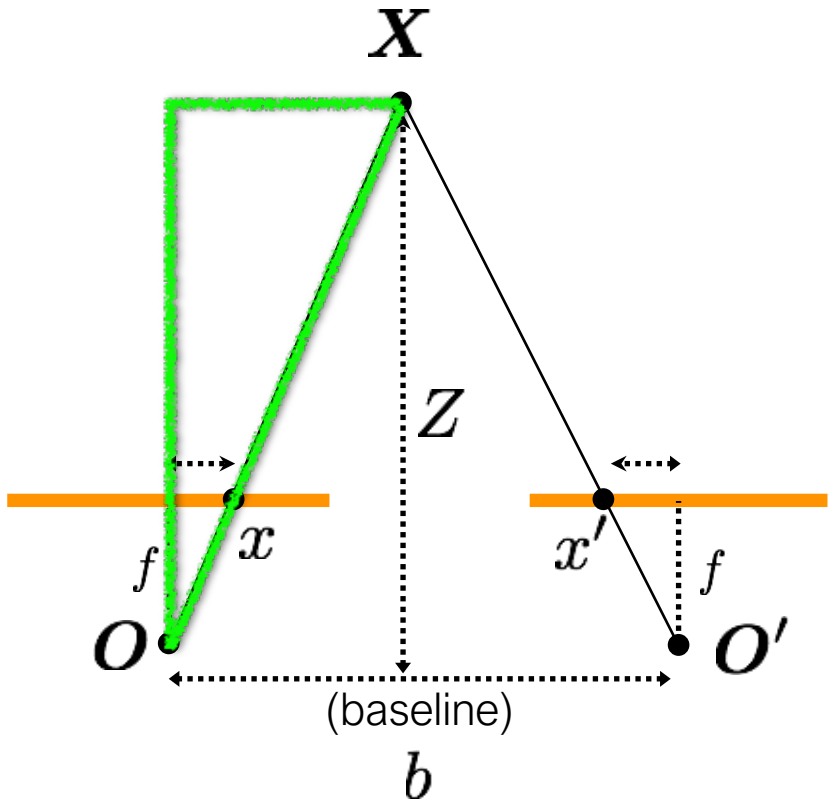


Important: coordinates **x** and **x'** are parameterized with respect to image center.

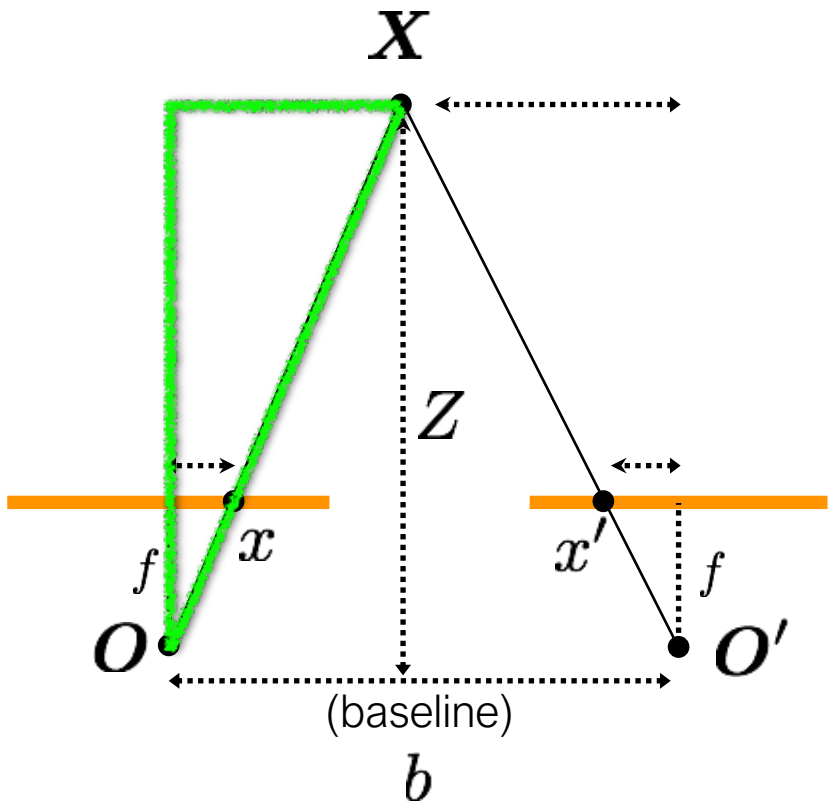

X
Z
x
x′
f
f
O
O′
(baseline)
b


X
X
Z
x
x
f
x′
f
O
O′
(baseline)
b

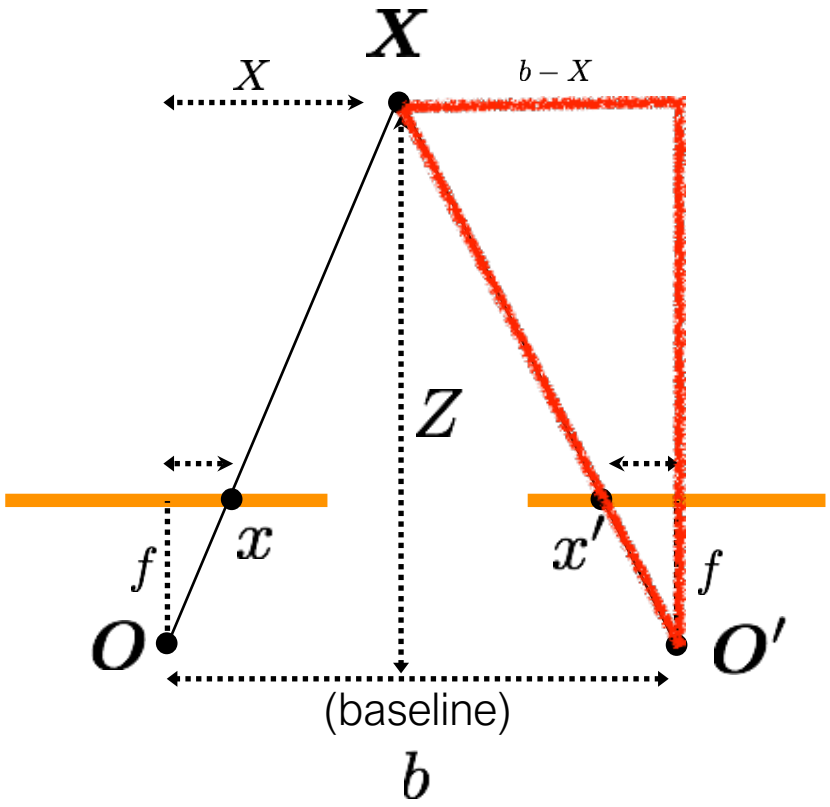
$$\frac{X}{Z} = \frac{x}{f}$$



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

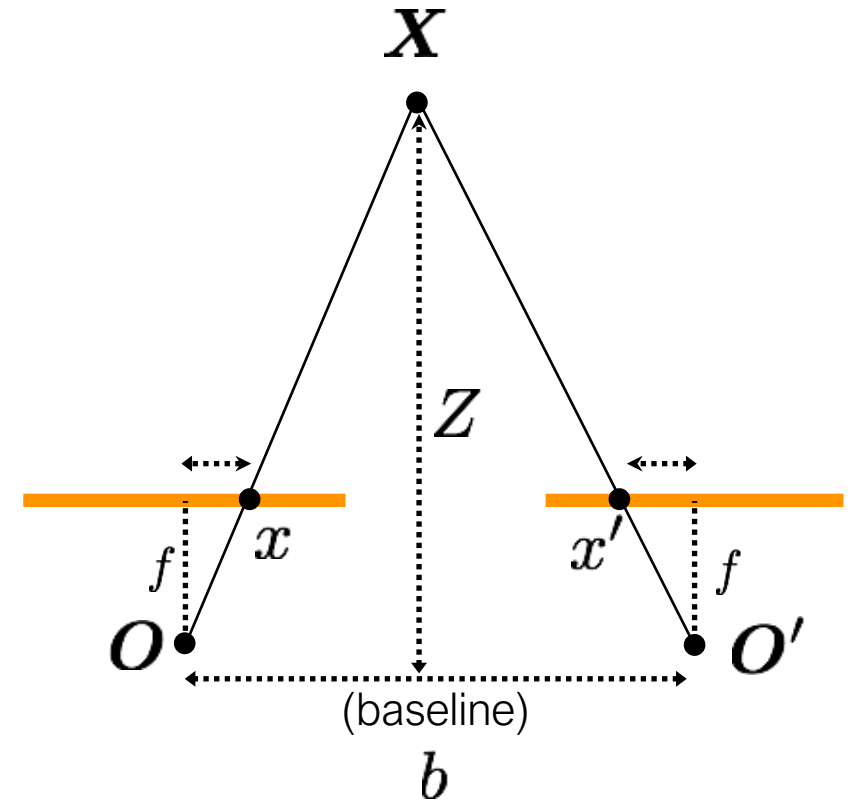


$$\frac{X}{Z} = \frac{x}{f}$$



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

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



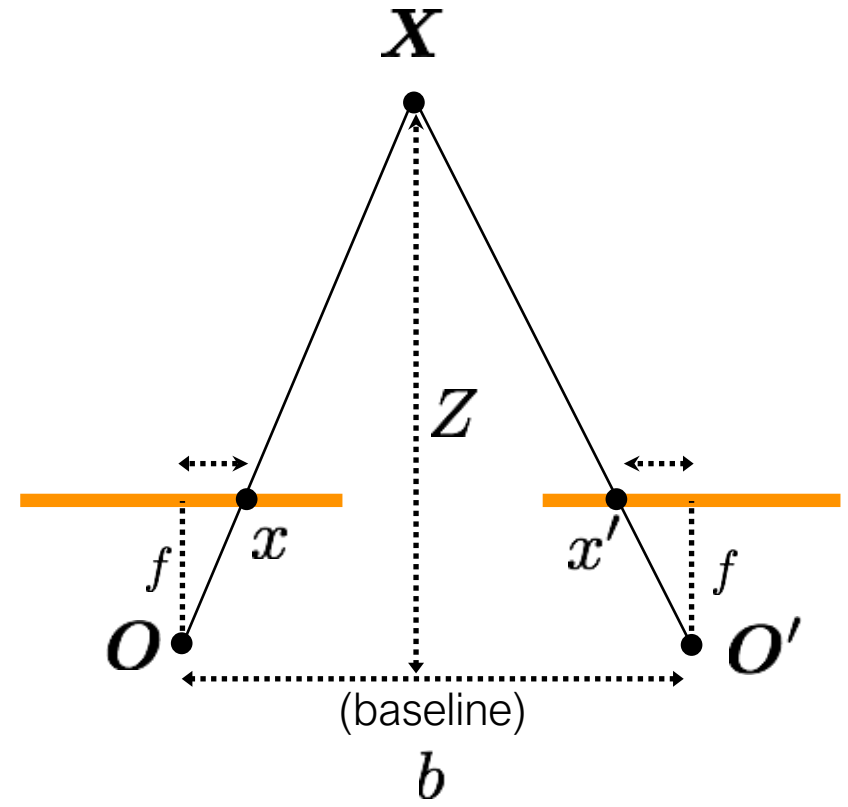
$$\frac{b - X}{Z} = \frac{-x'}{f}$$

Disparity

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

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

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



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

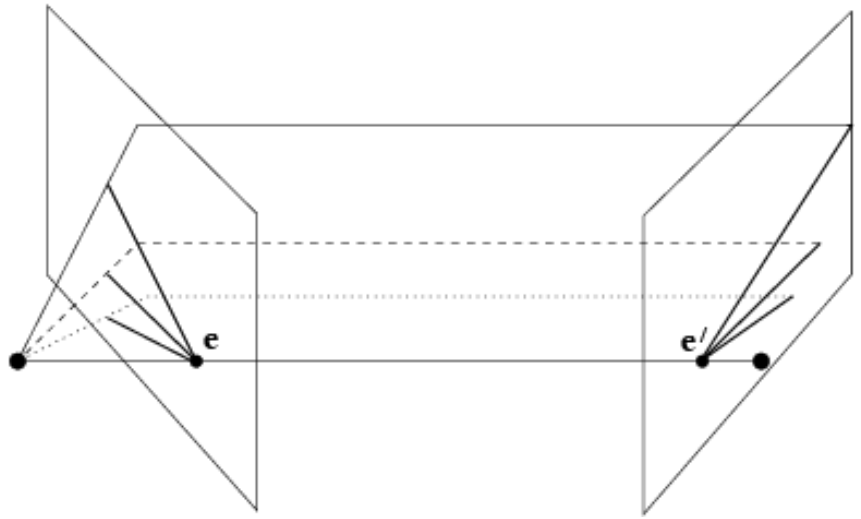
Disparity

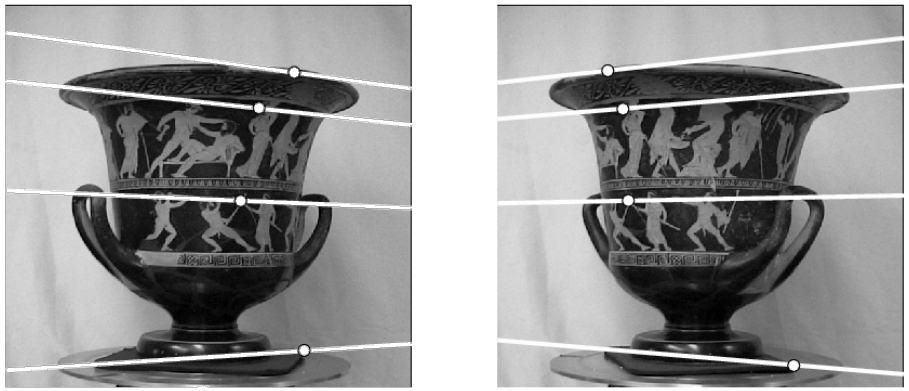
$$d = x - x'$$

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

inversely proportional to depth

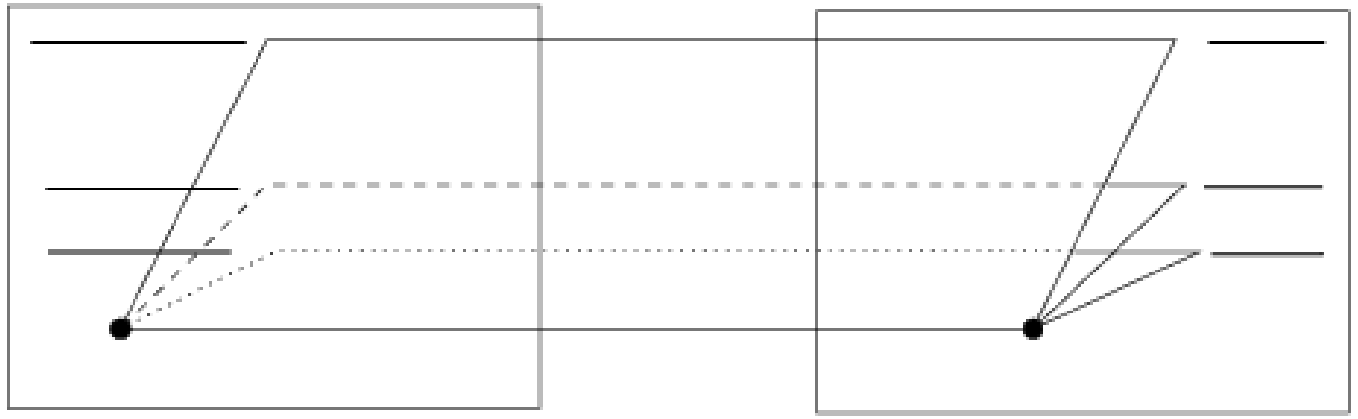
When can we use disparity?

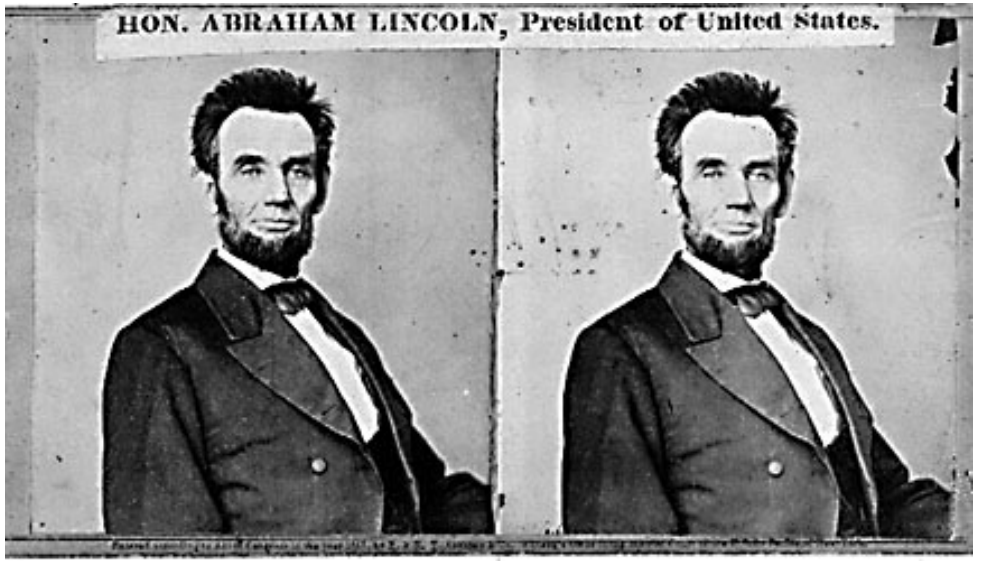




We cannot use disparity on images captured by non-parallel cameras.

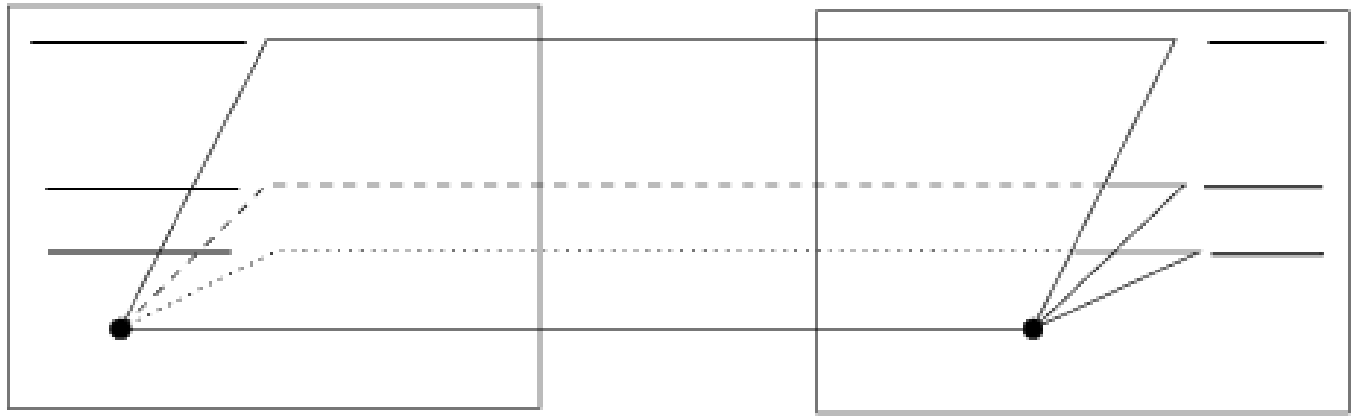
When can we use disparity?

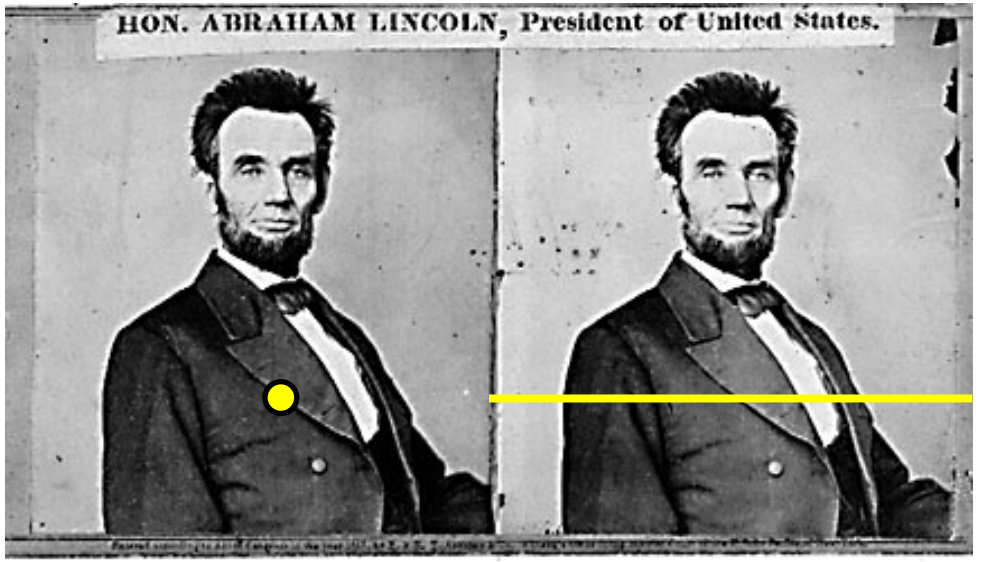




We can use disparity on images captured only by parallel cameras.

When can we use disparity?

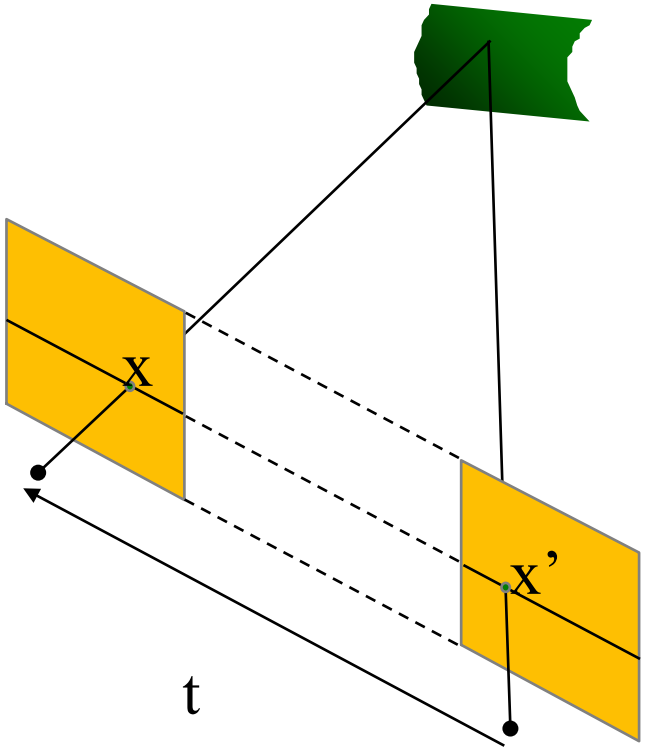




The epipolar line for a point in one image is the corresponding row in the other image.

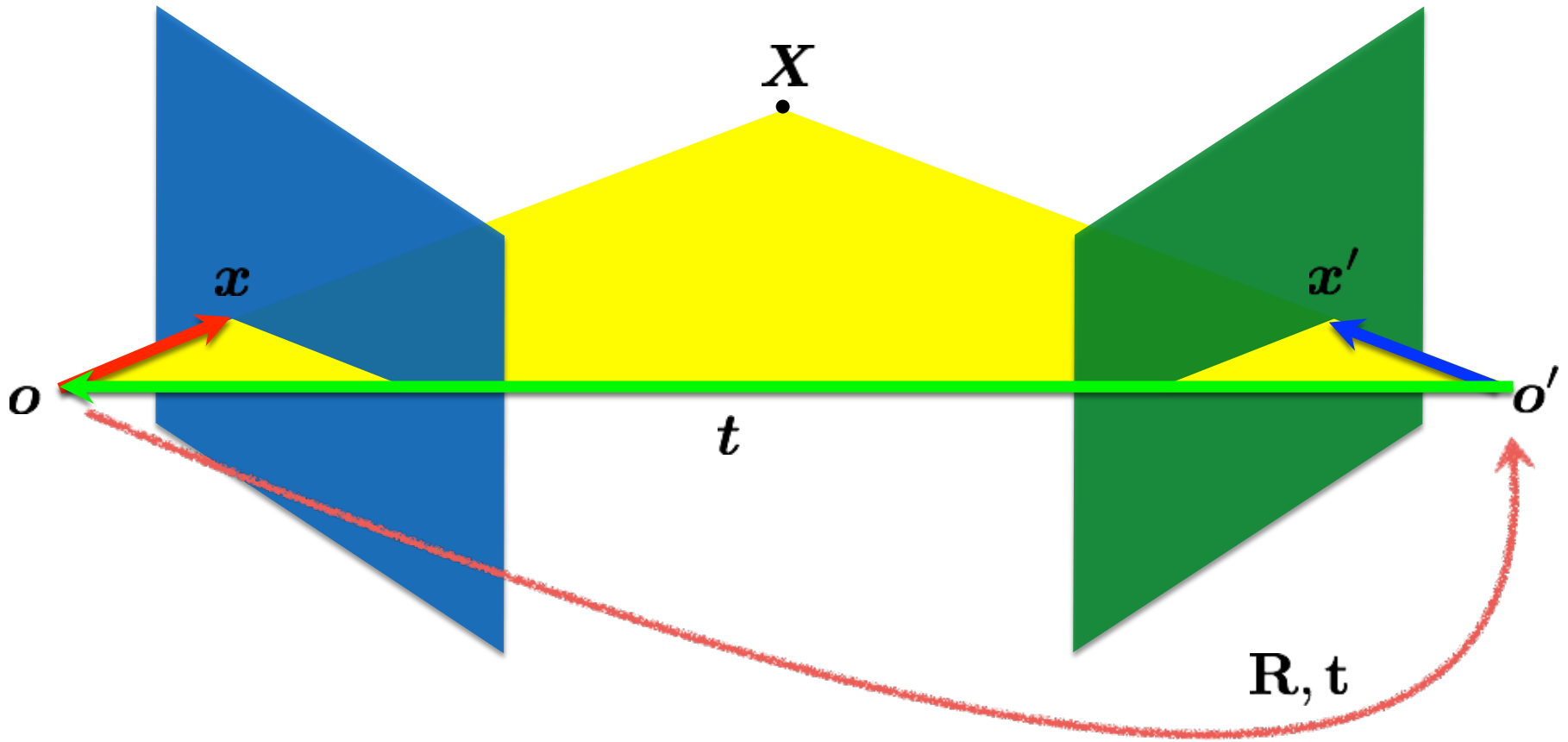
We can use disparity on images captured only by parallel cameras.

When are two cameras parallel?



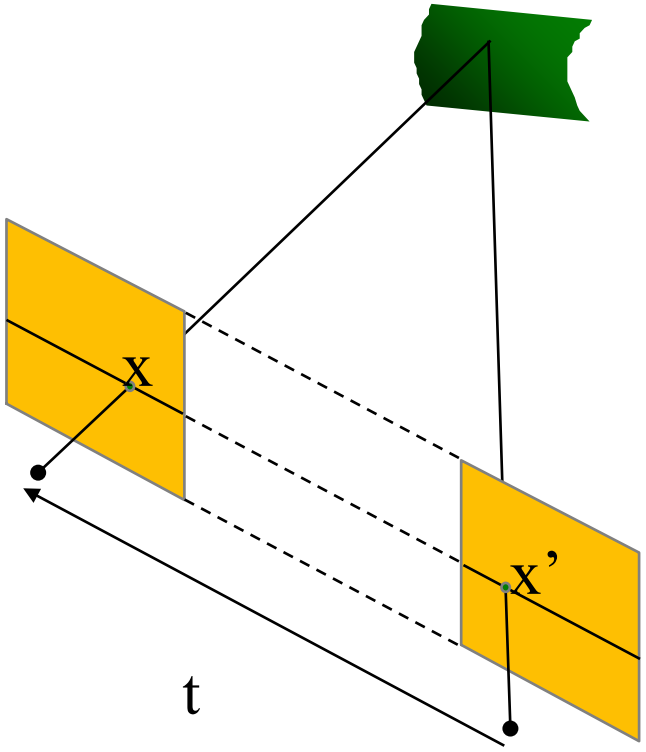
When this relationship holds:

$$R = I \qquad t = (T, 0, 0)$$



$$x' = \mathbf{R}(x - t)$$

When are two cameras parallel?



When this relationship holds:

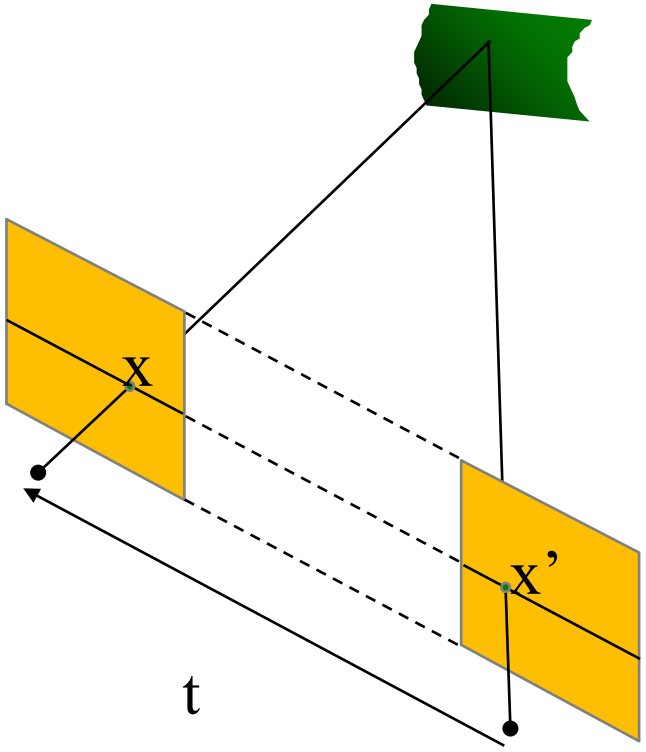
$$R = I \qquad t = (T, 0, 0)$$

Let's try this out…

$$E = t \times R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

This always has to hold: $x^T E x' = 0$

When are two cameras parallel?



When this relationship holds:

$$R = I \qquad t = (T, 0, 0)$$

Let's try this out…

$$E = t \times R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix}$$

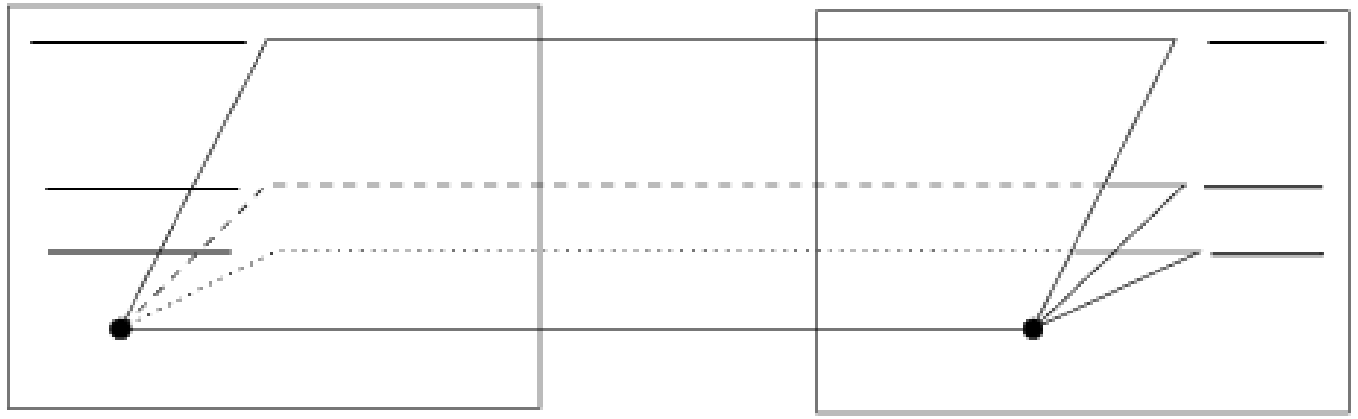
This always has to hold: $x^T E x' = 0$

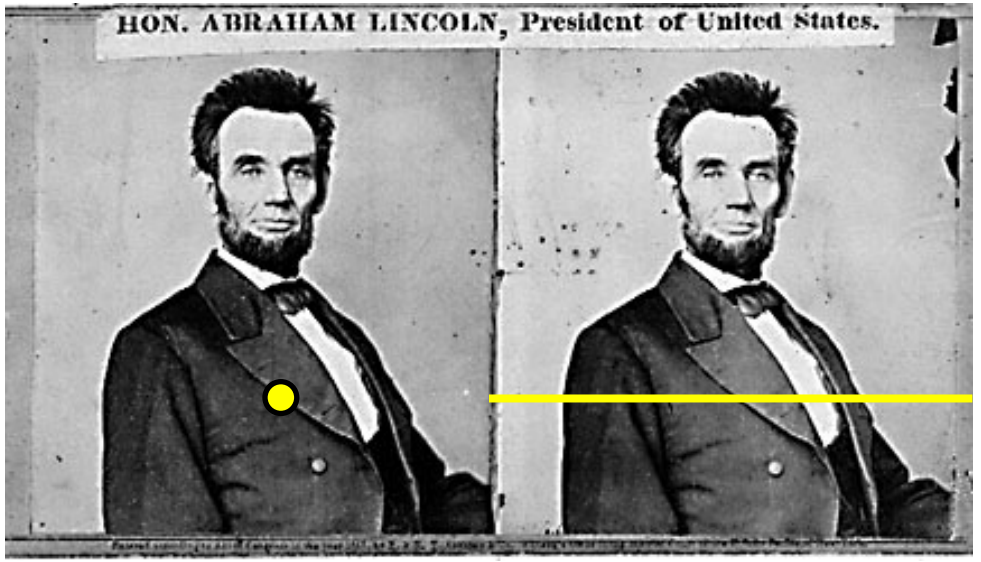
$$\begin{pmatrix} u & v & 1 \end{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -T \\ 0 & T & 0 \end{bmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0$$

$$Tv = Tv'$$

y coordinate is always the same!

When can we use disparity?



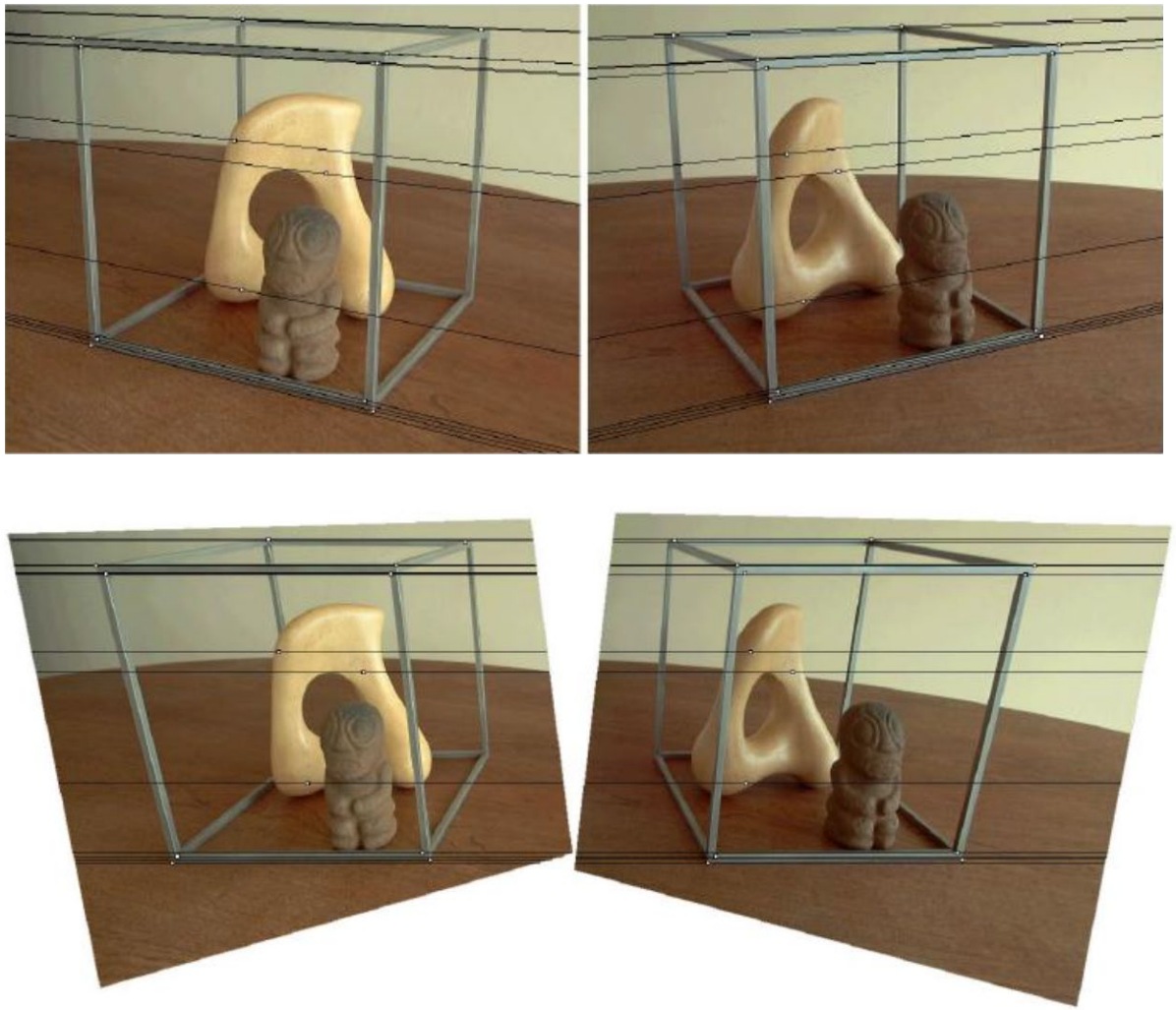


Depth from disparity will only work well for cameras separated by sufficient baseline.

The epipolar line for a point in one image is the corresponding row in the other image.

We can use disparity on images captured only by parallel cameras.

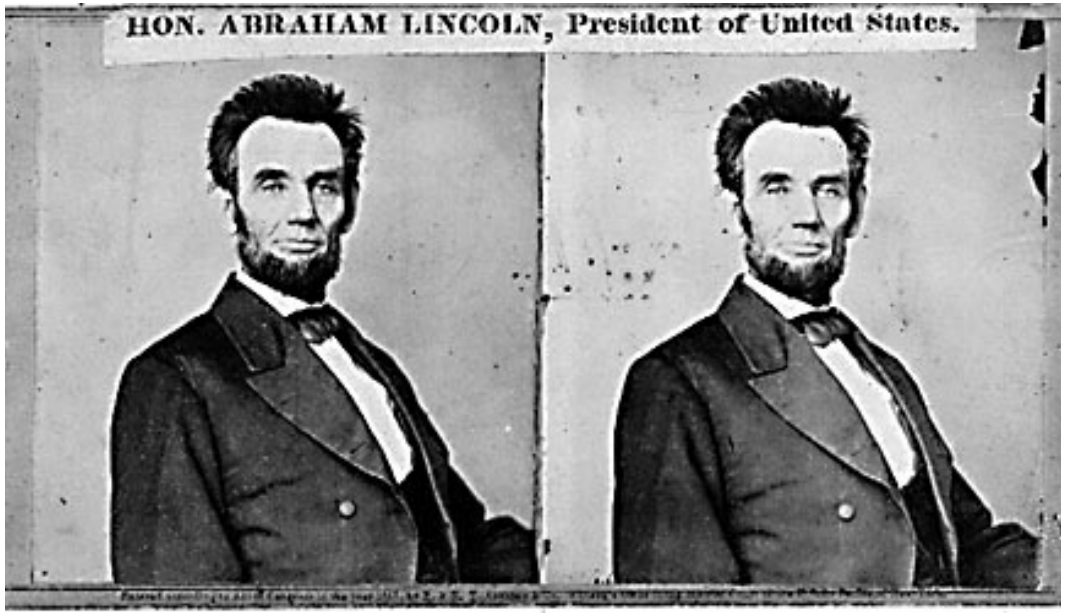
Stereo rectification



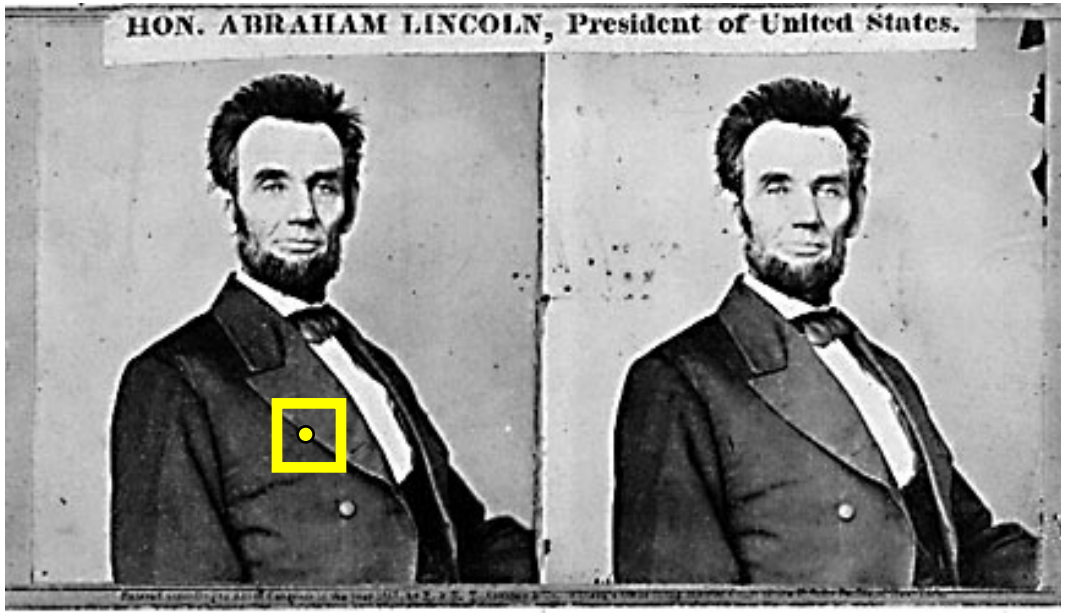
Warp images captured by non-parallel cameras so that they satisfy conditions for depth from disparity.

Stereo matching

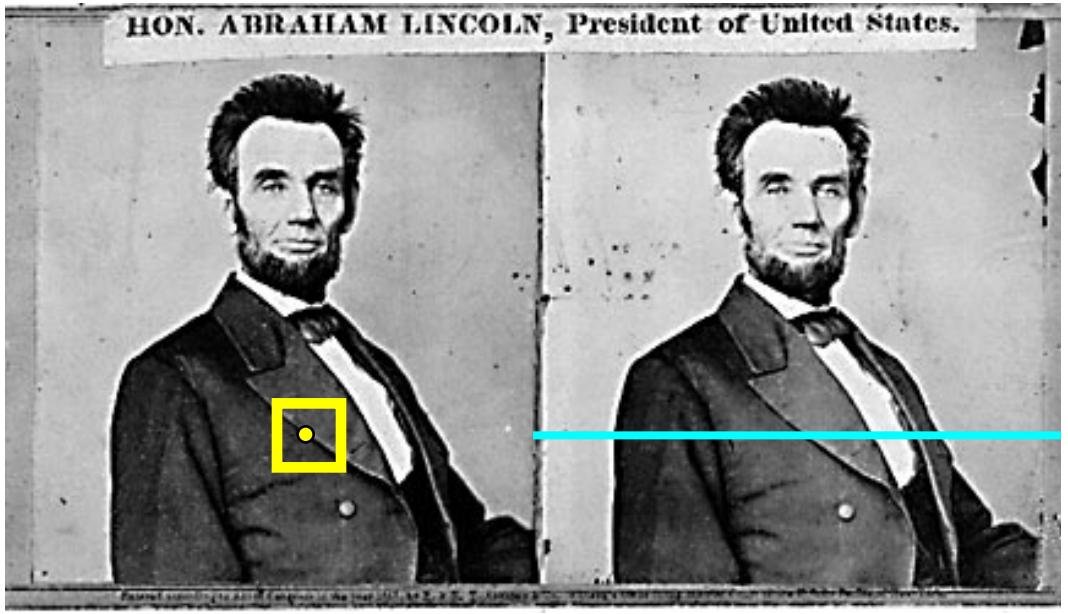
Depth Estimation via Stereo Matching



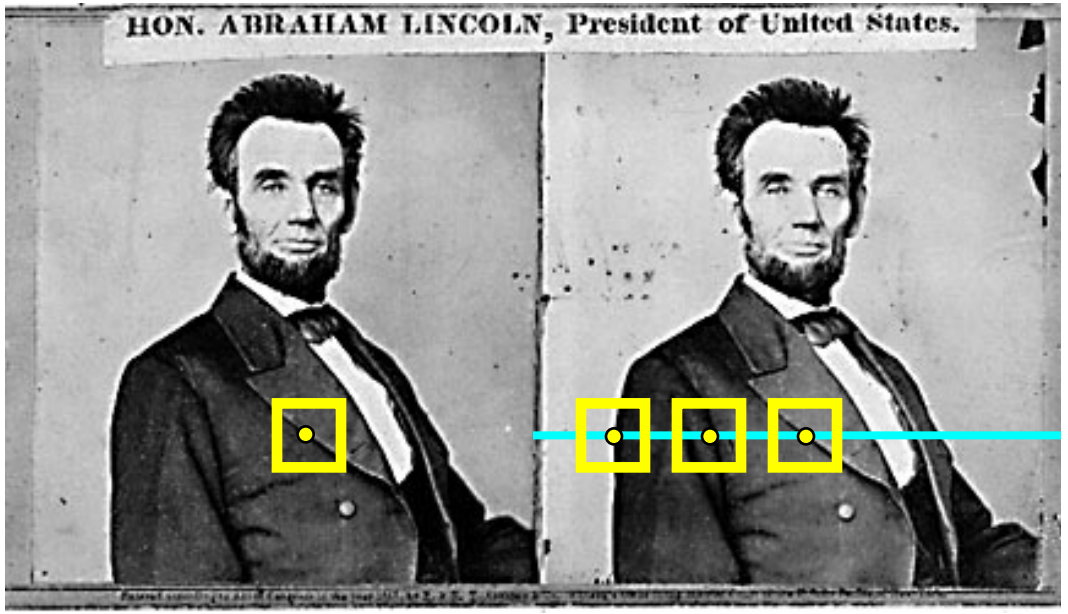
1. Rectify images
   (make epipolar lines horizontal)



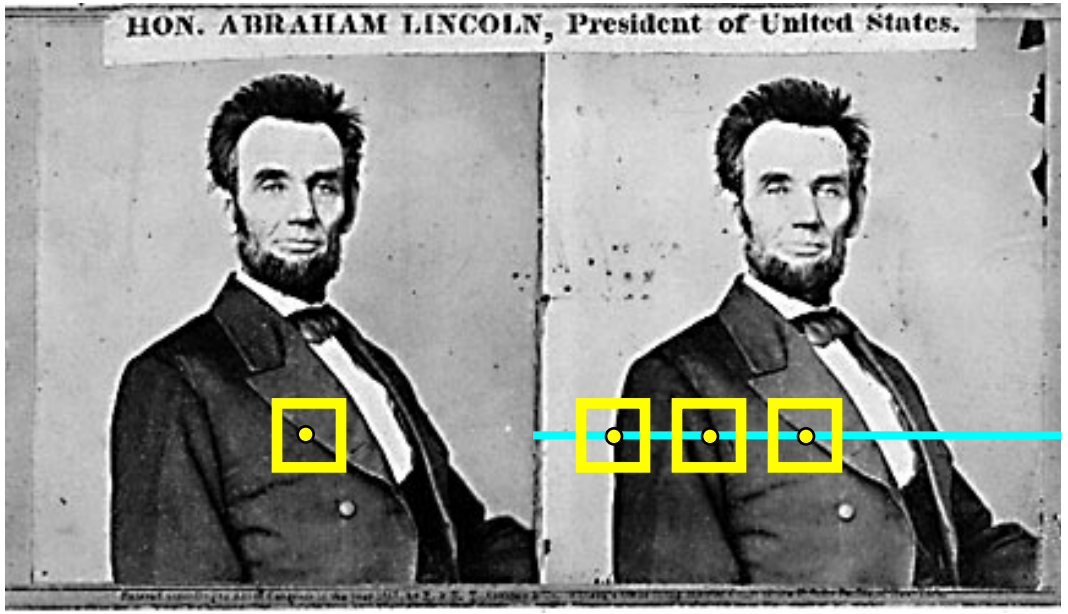
1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity



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



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

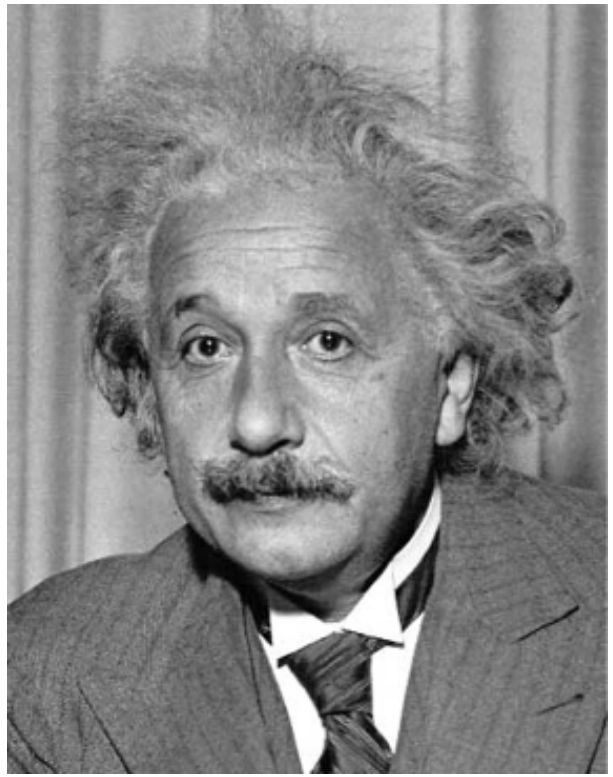


1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity

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

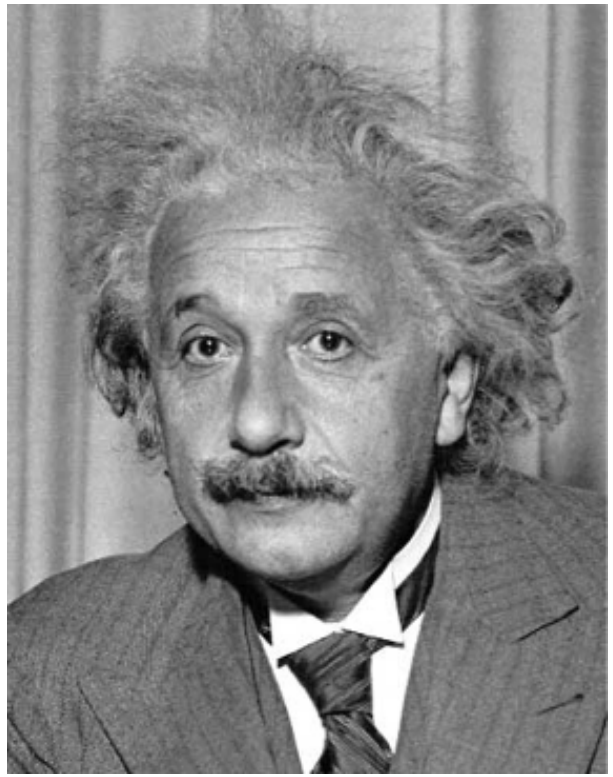
Find this template

How do we detect the template 👁 in he following image?



Find this template

How do we detect the template in he following image?

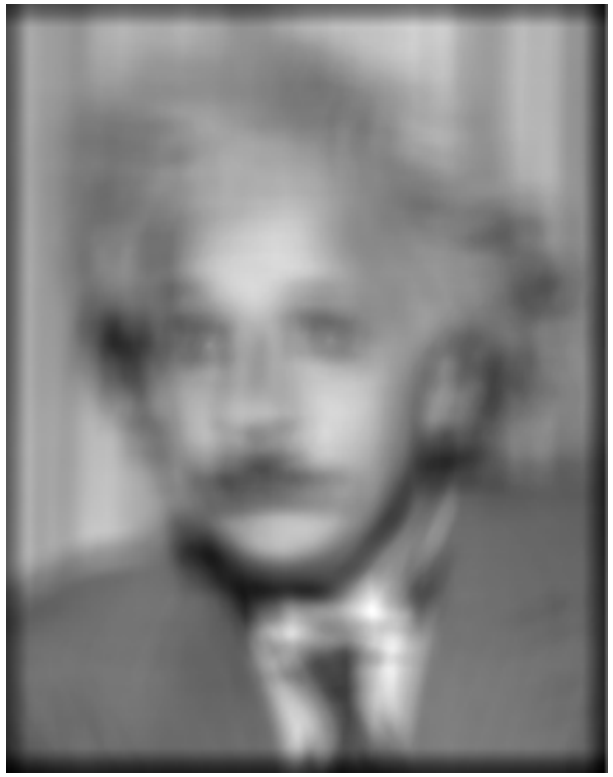


filter

output

$$h[m,n] = \sum_{k,j} g[k,l] f[m+k, n+l]$$

image

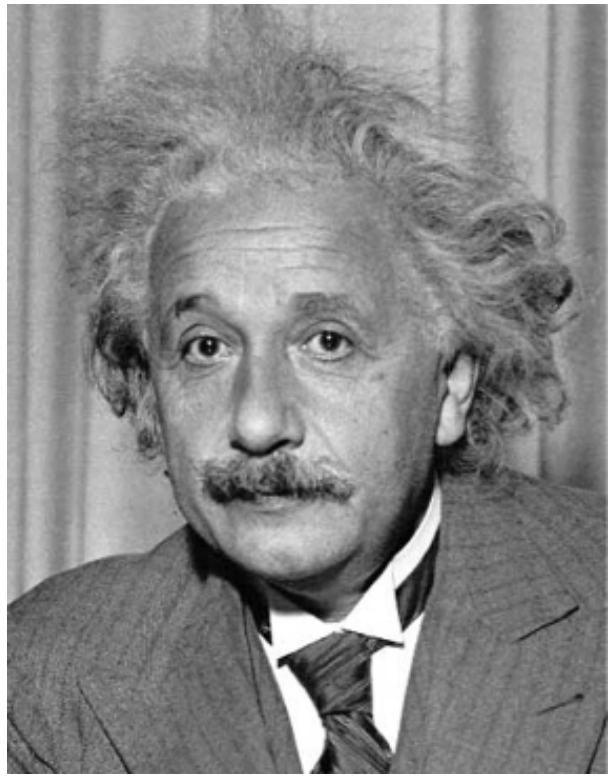


Output increases for higher local intensities.

Solution 1: Filter the image using the template as filter kernel.

Find this template

How do we detect the template in he following image?



output

filter

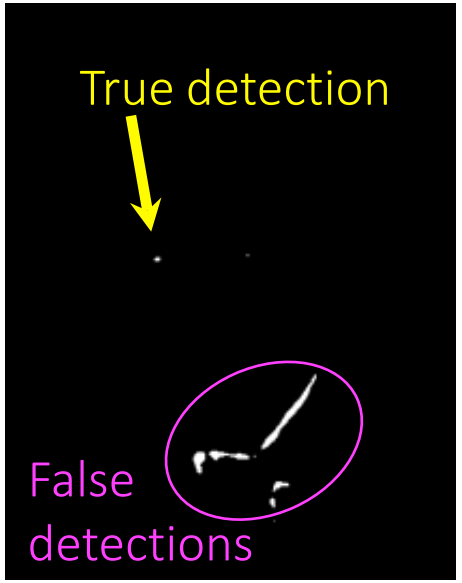
template mean

output

$$h[m,n] = \sum_{k,l}(g[k,l] - \bar{g})f[m+k, n+l]$$

thresholding

image

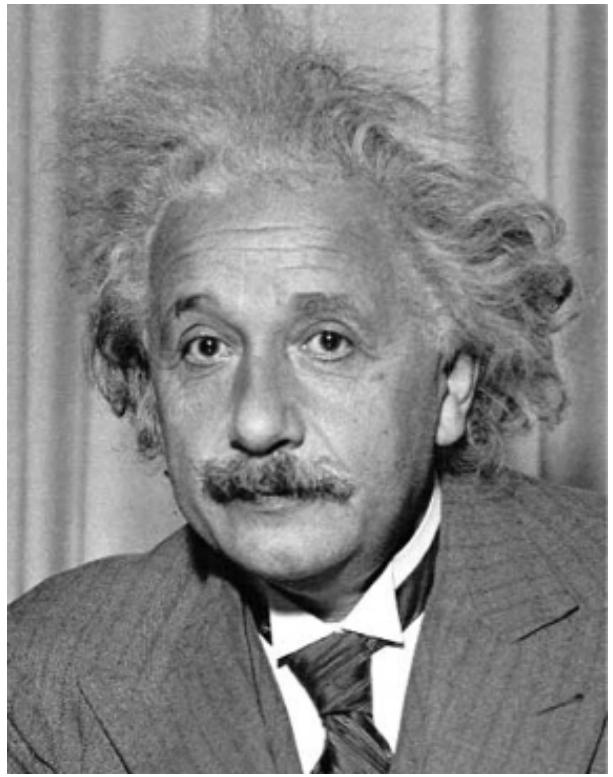


Solution 2: Filter the image using a *zero-mean* template.

Not robust to high contrast

Find this template

How do we detect the template in he following image?



filter

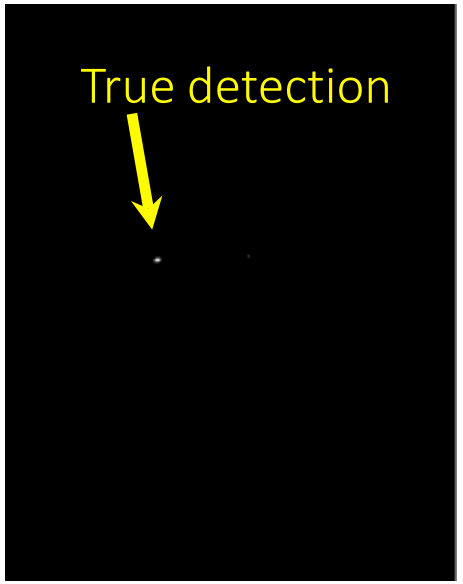
output

$$h[m,n] = \sum_{k,l}(g[k,l] - f[m+k, n+l])^2$$

image

1-output

thresholding

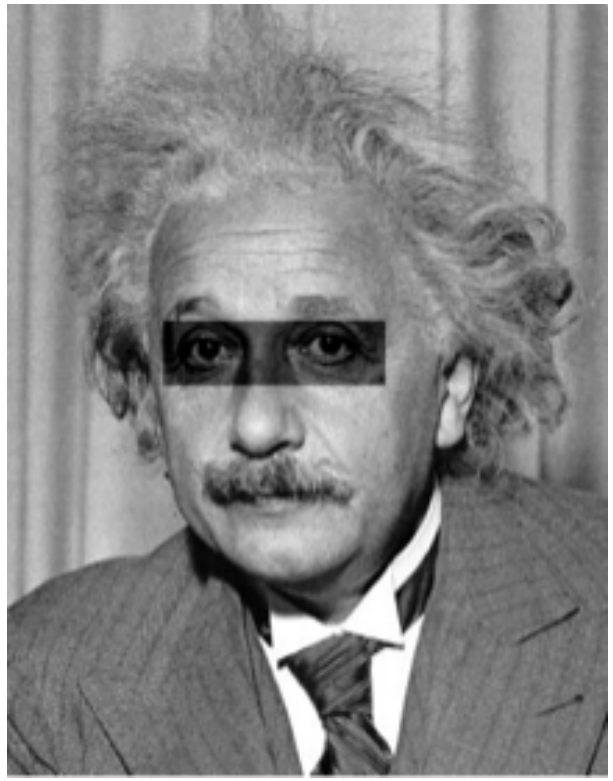


Solution 1: Use sum of squared differences (SSD).

Works well here.

Find this template

How do we detect the template 👁 in he following image?



1-output

filter 👁

output

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k, n+l])^2$$

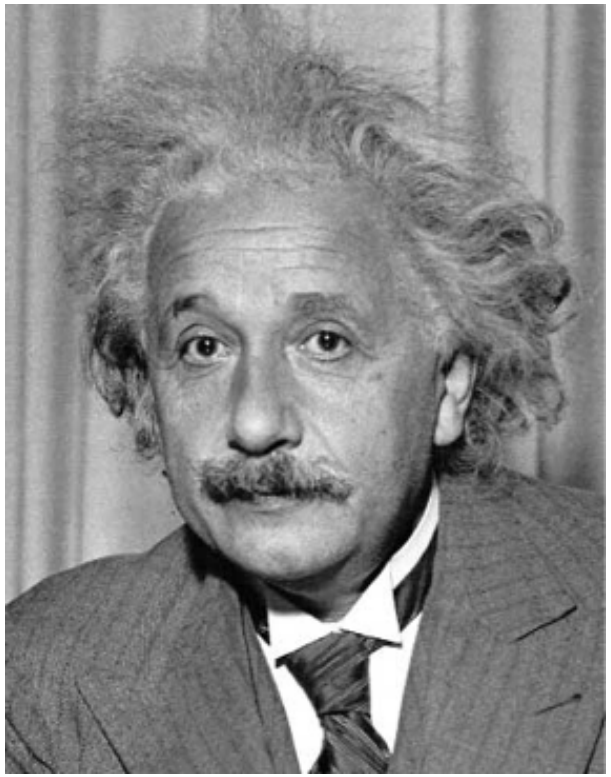
image

Not robust to local intensity changes

Solution 3: Use sum of squared differences (SSD).

Find this template

How do we detect the template in he following image?



filter

template mean

output

$$h[m,n] = \frac{\sum_{k,l}(g[k,l] - \bar{g})(f[m+k,n+l] - \bar{f}_{m,n})}{\sqrt{(\sum_{k,l}(g[k,l] - \bar{g})^2 \sum_{k,l}(f[m+k,n+l] - \bar{f}_{m,n})^2)}}$$

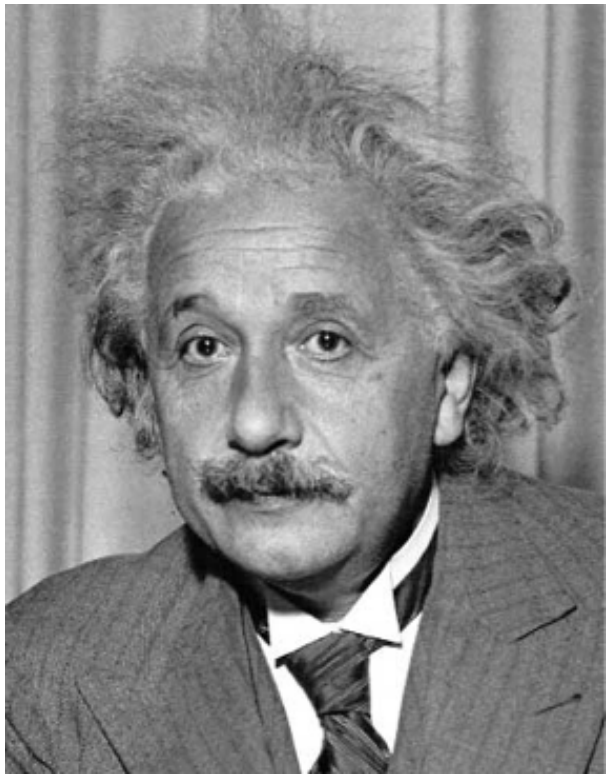
image

local patch mean

Solution 4: Normalized cross-correlation (NCC).

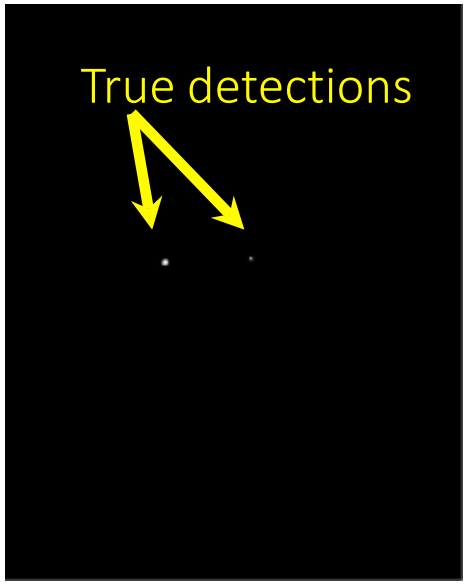
Find this template

How do we detect the template  in he following image?



1-output

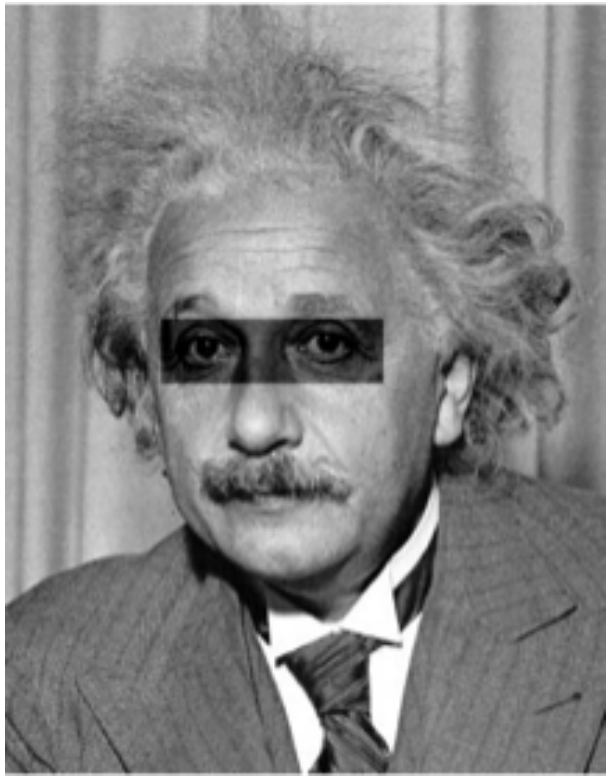
thresholding



Solution 4: Normalized cross-correlation (NCC).

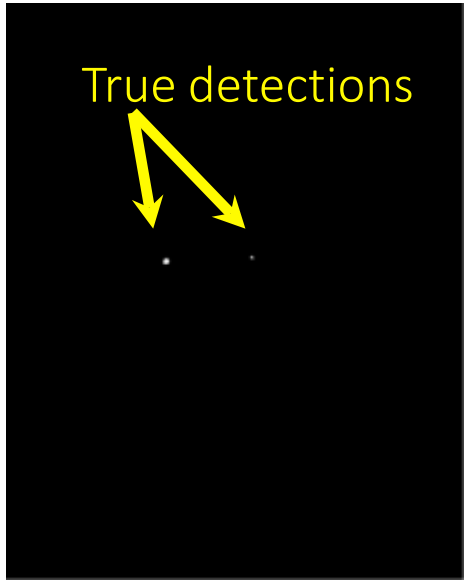
Find this template

How do we detect the template  in he following image?



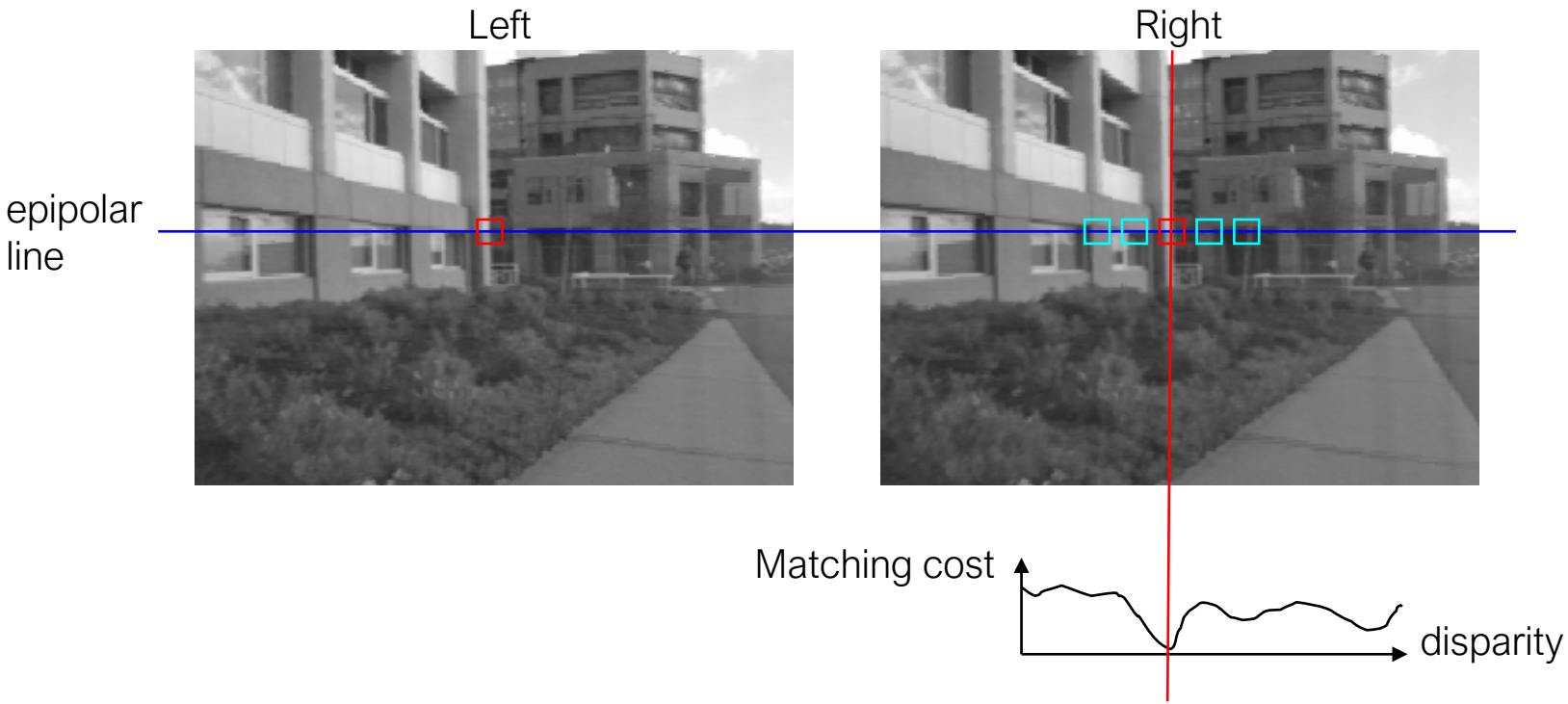
1-output

thresholding

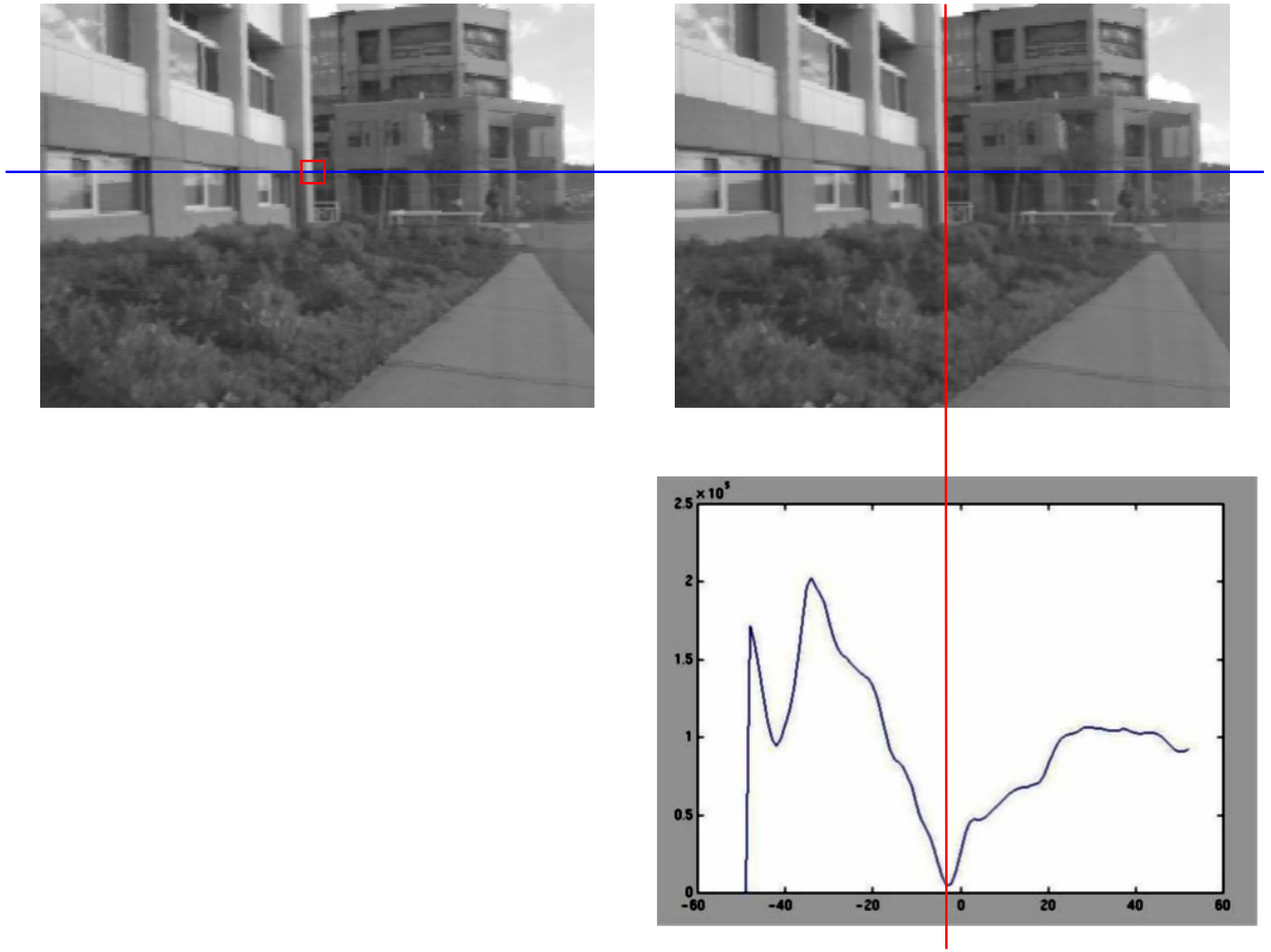


Solution 4: Normalized cross-correlation (NCC).

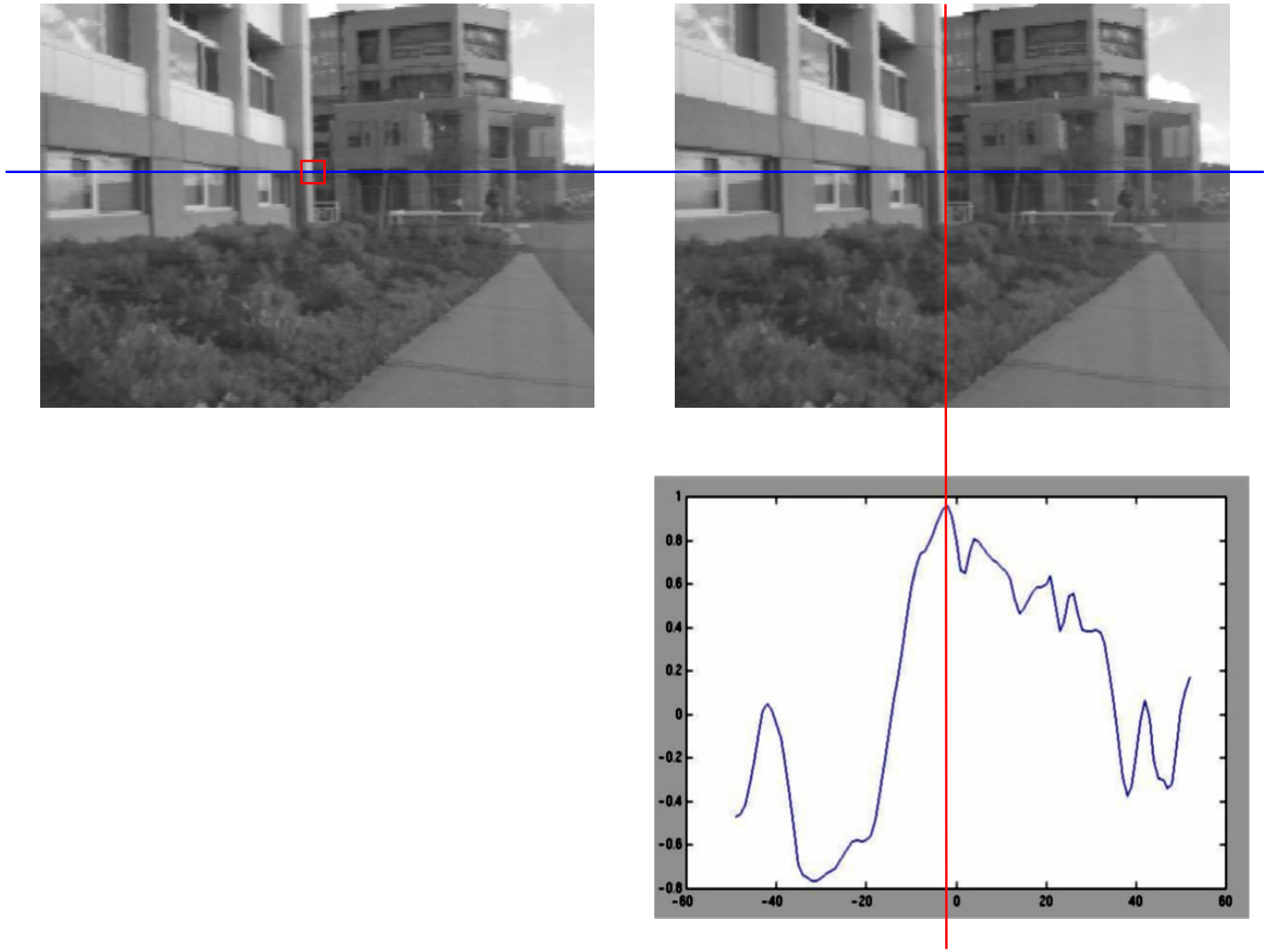
Stereo Block Matching



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



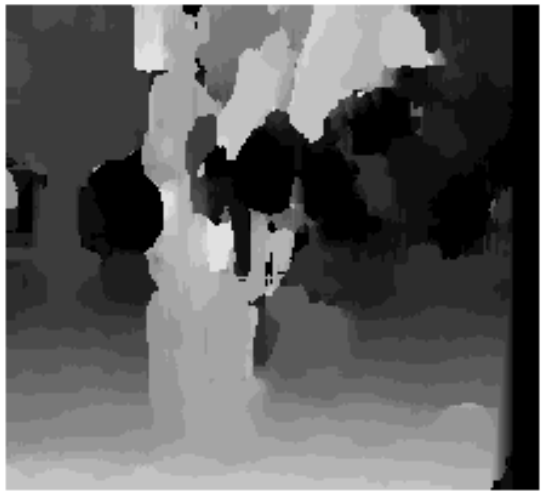
SSD



Normalized cross-correlation

Effect of window size

W = 3



W = 20

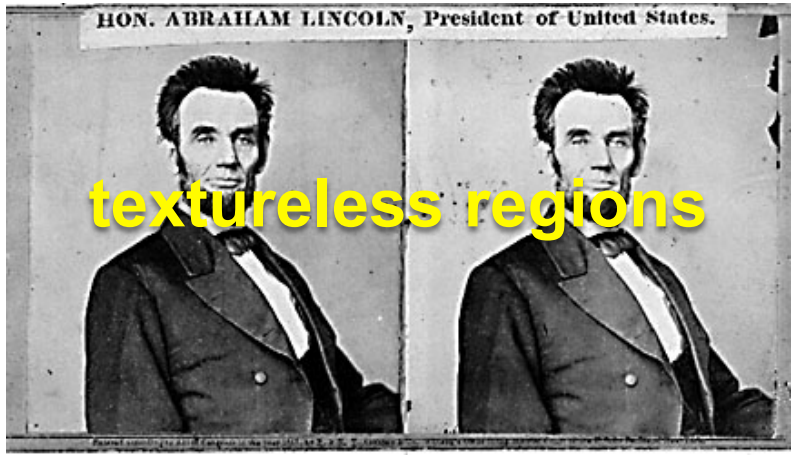
Smaller window

+ More detail
- More noise

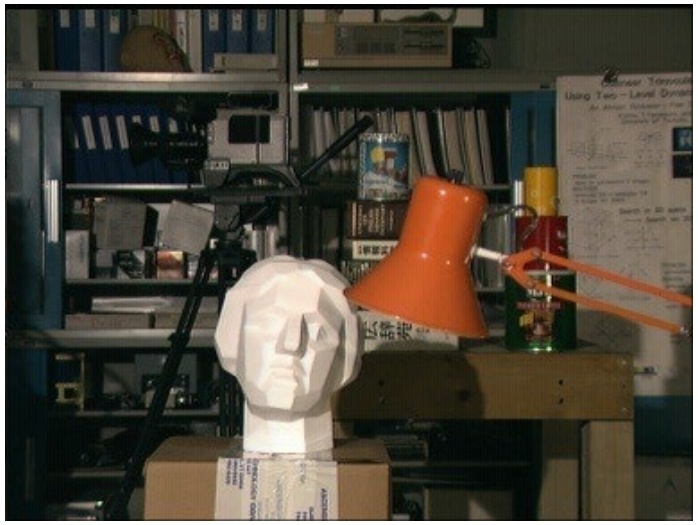
Larger window

+ Smoother disparity maps
- Less detail
- Fails near boundaries

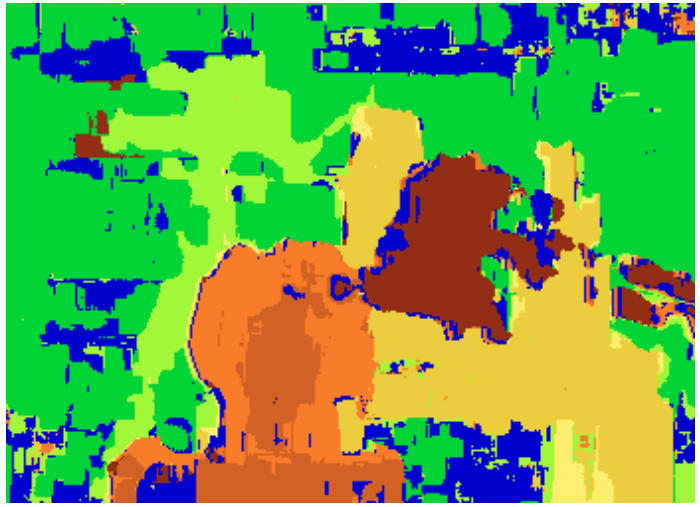
Failure cases for stereo block matching



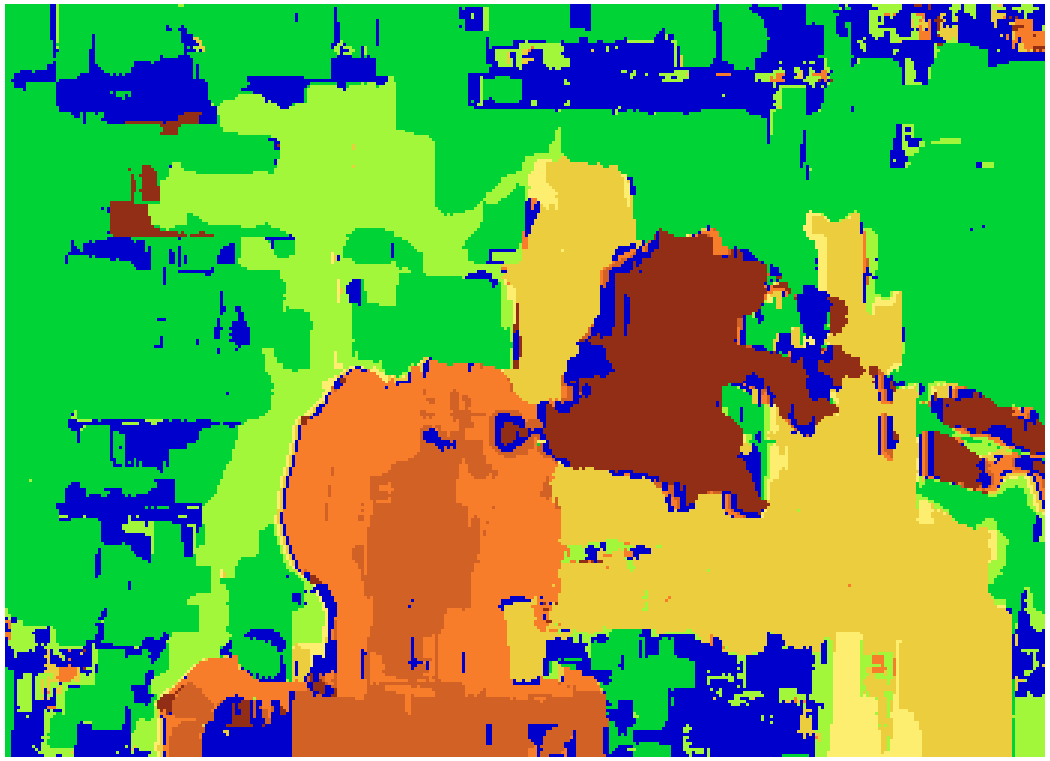
Improving stereo matching



Block matching



Ground truth

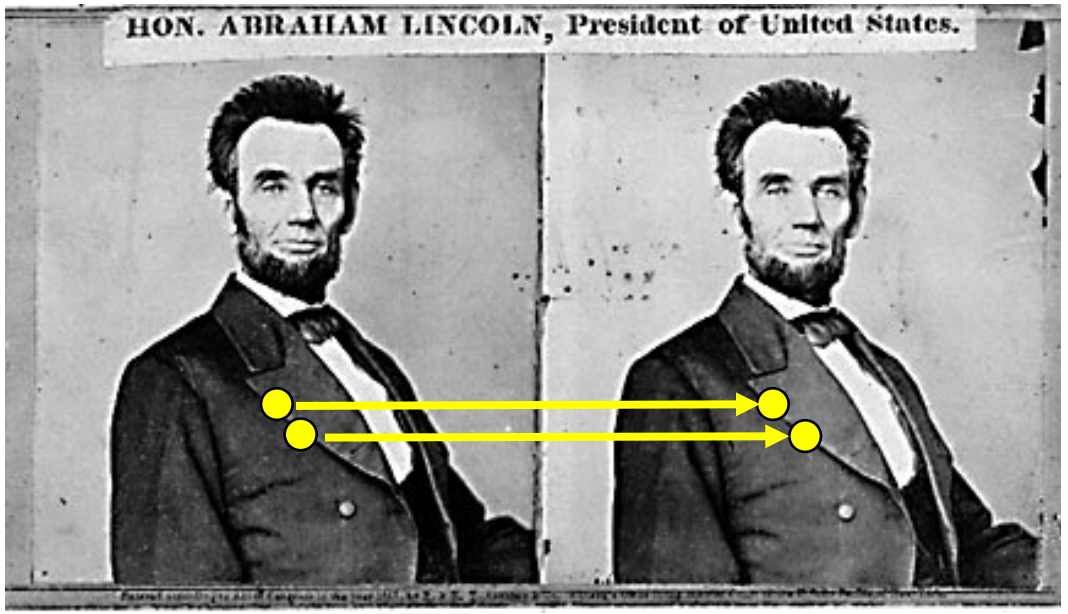


Too many discontinuities.
We expect disparity values to change slowly.

Let’s make an assumption:
depth should change smoothly

Stereo matching as …

Energy Minimization



What defines a good stereo correspondence?

1. **Match quality**
   - Want each pixel to find a good match in the other image
2. **Smoothness**
   - If two pixels are adjacent, they should (usually) move about the same amount

Energy minimization

energy function
(for one pixel)

$$E(d) = \underbrace{E_d(d)}_{\text{data term}} + \lambda \underbrace{E_s(d)}_{\text{smoothness term}}$$

Want each pixel to find a good match in the other image

(block matching result)

Adjacent pixels should (usually) move about the same amount

(smoothness function)

$$E(d) = E_d(d) + \lambda E_s(d)$$

$$E_d(d) = \sum_{(x,y)\in I} C(x, y, d(x, y))$$

data term

SSD distance between windows
centered at I(x, y) and J(x+ d(x,y), y)

$$E(d) = E_d(d) + \lambda E_s(d)$$

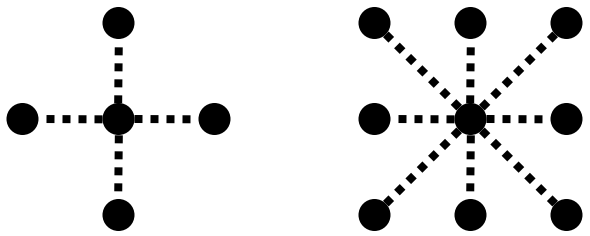
$$E_d(d) = \sum_{(x,y) \in I} C(x, y, d(x, y))$$

SSD distance between windows centered at I(x, y) and J(x+ d(x,y), y)

$$E_s(d) = \sum_{(p,q) \in \mathcal{E}} V(d_p, d_q)$$

smoothness term

$\mathcal{E}$: set of neighboring pixels



4-connected neighborhood

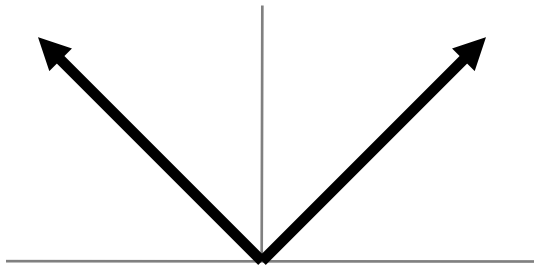
8-connected neighborhood

$$E_s(d) = \sum_{(p,q)\in\mathcal{E}} V(d_p, d_q)$$

smoothness term

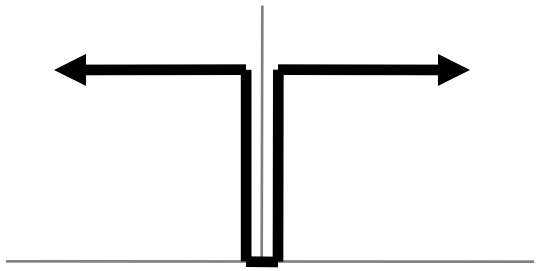
$$V(d_p, d_q) = |d_p - d_q|$$

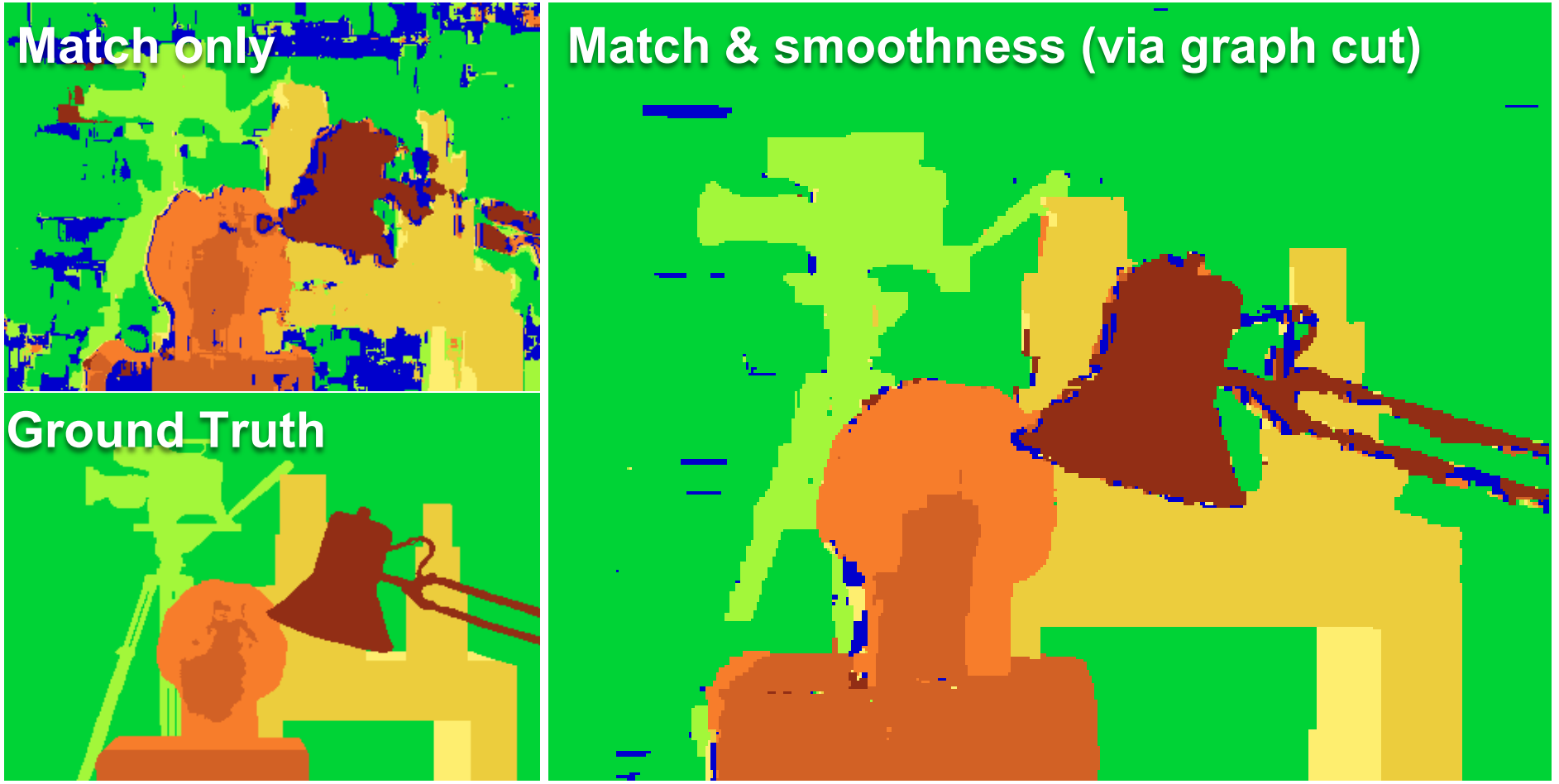
L_1 distance



$$V(d_p, d_q) = \begin{cases} 0 & \text{if } d_p = d_q \\ 1 & \text{if } d_p \neq d_q \end{cases}$$

"Potts model"





Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

Solving linear systems

Heterogeneous linear systems

$$A \cdot x = b$$

Heterogeneous linear systems

$$A \cdot x = b$$

heterogeneous: right hand side **b** is non-zero

Heterogeneous linear systems

$$A \cdot x = b$$

dimensions: $M \times N$ $N \times 1$ $M \times 1$

Heterogeneous linear systems

$$A \cdot x = b$$

dimensions: $M \times N$ $N \times 1$ $M \times 1$

Three cases.

- $M < N$: the system has infinite solutions **(underdetermined system)**
- $M = N$: the system has a unique solution $A^{-1}b$ if A is invertible, otherwise it has infinite solutions
- $M > N$: the system *typically* has no solutions **(overdetermined system)**

Heterogeneous linear systems

$$A \cdot x = b$$

dimensions: $M \times N$ $N \times 1$ $M \times 1$

Three cases.

- $M < N$: the system has infinite solutions **(underdetermined system)**
- $M = N$: the system has a unique solution $A^{-1}b$ if A is invertible, otherwise it has infinite solutions
- $M > N$: the system *typically* has no solutions **(overdetermined system)**

We are typically interested in the $M \gg N$ case

Least-squares solution to heterogeneous linear systems

Replace the linear system:

$$A \cdot x = b$$

$$M \times N \quad N \times 1 \qquad M \times 1$$

with the optimization problem:

$$\min_x \|A \cdot x - b\|^2$$

Least-squares solution to heterogeneous linear systems

Replace the linear system:

$$A \cdot x = b$$

$$M \times N \quad N \times 1 \qquad M \times 1$$

with the optimization problem:

$$\min_x \|A \cdot x - b\|^2$$

- $M = N$: same solution as original system
- $M > N$: find x that comes *closest* to being a solution in the least-squares sense

Solving the linear least squares optimization

Take the loss function:

$$E_{\mathrm{LLS}} = \|A \cdot x - b\|^2$$

Solving the linear least squares optimization

Take the loss function:

$$E_{\mathrm{LLS}} = \|A \cdot x - b\|^2$$

Expand the loss function:

$$E_{\mathrm{LLS}} = x^{\mathsf{T}}(A^{\mathsf{T}}A)x - 2x^{\mathsf{T}}(A^{\mathsf{T}}b) + \|b\|^2$$

Solving the linear least squares optimization

Take the loss function:

$$E_{\mathrm{LLS}} = \|A \cdot x - b\|^2$$

Expand the loss function:

$$E_{\mathrm{LLS}} = x^{\mathsf{T}}(A^{\mathsf{T}}A)x - 2x^{\mathsf{T}}(A^{\mathsf{T}}b) + \|b\|^2$$

Compute the derivative of the loss function:

$$\frac{\mathrm{d}E_{\mathrm{LLS}}}{\mathrm{d}x} = 2(A^{\mathsf{T}}A)x - 2(A^{\mathsf{T}}b)$$

Solving the linear least squares optimization

Take the loss function:

$$E_{\mathrm{LLS}} = \|A \cdot x - b\|^2$$

Expand the loss function:

$$E_{\mathrm{LLS}} = x^{\mathsf{T}}(A^{\mathsf{T}}A)x - 2x^{\mathsf{T}}(A^{\mathsf{T}}b) + \|b\|^2$$

Compute the derivative of the loss function:

$$\frac{\mathrm{d}E_{\mathrm{LLS}}}{\mathrm{d}x} = 2(A^{\mathsf{T}}A)x - 2(A^{\mathsf{T}}b)$$

Compute the derivative of the loss function:

$$\frac{\mathrm{d}E_{\mathrm{LLS}}}{\mathrm{d}x} = 0 \Rightarrow 2(A^{\mathsf{T}}A)x - 2(A^{\mathsf{T}}b) = 0 \Rightarrow (A^{\mathsf{T}}A)x = A^{\mathsf{T}}b$$

Solving the linear least squares optimization

Take the loss function:

$$E_{\mathrm{LLS}} = \|A \cdot x - b\|^2$$

Expand the loss function:

$$E_{\mathrm{LLS}} = x^{\mathsf{T}}(A^{\mathsf{T}}A)x - 2x^{\mathsf{T}}(A^{\mathsf{T}}b) + \|b\|^2$$

Compute the derivative of the loss function:

$$\frac{\mathrm{d}E_{\mathrm{LLS}}}{\mathrm{d}x} = 2(A^{\mathsf{T}}A)x - 2(A^{\mathsf{T}}b)$$

Compute the derivative of the loss function:

$$\frac{\mathrm{d}E_{\mathrm{LLS}}}{\mathrm{d}x} = 0 \Rightarrow 2(A^{\mathsf{T}}A)x - 2(A^{\mathsf{T}}b) = 0 \Rightarrow (A^{\mathsf{T}}A)x = A^{\mathsf{T}}b$$

Solve the resulting heterogeneous linear system:

$$x = (A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}b$$

Solving the linear least squares optimization

Take the loss function:

$$E_{\mathrm{LLS}} = \|A \cdot x - b\|^2$$

Expand the loss function:

$$E_{\mathrm{LLS}} = x^{\mathsf{T}}(A^{\mathsf{T}}A)x - 2x^{\mathsf{T}}(A^{\mathsf{T}}b) + \|b\|^2$$

Compute the derivative of the loss function:

$$\frac{\mathrm{d}E_{\mathrm{LLS}}}{\mathrm{d}x} = 2(A^{\mathsf{T}}A)x - 2(A^{\mathsf{T}}b)$$

Compute the derivative of the loss function:

$$\frac{\mathrm{d}E_{\mathrm{LLS}}}{\mathrm{d}x} = 0 \Rightarrow 2(A^{\mathsf{T}}A)x - 2(A^{\mathsf{T}}b) = 0 \Rightarrow (A^{\mathsf{T}}A)x = A^{\mathsf{T}}b$$

Solve the resulting heterogeneous linear system:

$$x = \boxed{(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}}b$$

Pseudo-inverse of $M \times N$ matrix A:

$$(A^{\mathsf{T}}A)^{-1}A^{\mathsf{T}}$$

- $M = N$: equal to A^{-1}, same solution as original system
- $M > N$: $A^{\mathsf{T}}A$ square $N \times N$ matrix, typically invertible when $M \gg N$.

Heterogeneous linear systems

$$A \cdot x = b$$

dimensions: $M \times N$ $N \times 1$ $M \times 1$

Homogeneous linear systems

$$A \cdot x = 0$$

dimensions: $M \times N$ $N \times 1$ $M \times 1$

Homogeneous linear systems

$$A \cdot x = \boxed{0}$$

homogeneous: right hand side is zero

dimensions: $M \times N$ $N \times 1$ $M \times 1$

Homogeneous linear systems

$$A \cdot x = 0$$

dimensions: $M \times N$ $N \times 1$ $M \times 1$

Three cases.

- $M < N$: the system has at least one solution, the trivial solution $x = 0$
- $M = N$: the system has at least one solution, the trivial solution $x = 0$
- $M > N$: the system has at least one solution, the trivial solution $x = 0$

Homogeneous linear systems

$$A \cdot x = 0$$

dimensions: $M \times N$ $N \times 1$ $M \times 1$

Three cases.

- $M < N$: the system has at least one solution, the trivial solution $x = 0$
- $M = N$: the system has at least one solution, the trivial solution $x = 0$
- $M > N$: the system has at least one solution, the trivial solution $x = 0$

We want to avoid the trivial solution

Least-squares solution to homogeneous linear systems

Replace the linear system:

$$A \cdot x = 0$$

$$M \times N \quad N \times 1 \qquad M \times 1$$

with the optimization problem:

$$\min_x \|A \cdot x\|^2$$

Least-squares solution to homogeneous linear systems

Replace the linear system:

$$A \cdot x = 0$$

$$M \times N \quad N \times 1 \qquad M \times 1$$

with the optimization problem:

$$\min_x \|A \cdot x\|^2$$

and add constraint: $\|x\|^2 = 1$ to avoid trivial solution

Solving the constrained linear least squares optimization

$$\min_{x:\ \|x\|^2=1} \|A \cdot x\|^2$$

Solving the constrained linear least squares optimization

$$\min_{x:\ \|x\|^2=1} \|A \cdot x\|^2$$

Compute the *singular value decomposition* of A:

left singular vectors (orthonormal) — singular values (diagonal) — right singular vectors (orthonormal)

$$A = U \cdot \Sigma \cdot V^\top$$

$M \times N$ $\quad$ $M \times M$ $\quad$ $M \times N$ $\quad$ $N \times N$

Solving the constrained linear least squares optimization

$$\min_{x:\ \|x\|^2=1} \|A \cdot x\|^2$$

Compute the *singular value decomposition* (SVD) of A:

left singular vectors (orthonormal) — singular values (diagonal) — right singular vectors (orthonormal)

$$A = U \cdot \Sigma \cdot V^\top$$

$M \times N \qquad M \times M \quad M \times N \quad N \times N$

Solution is equal to the column of V corresponding to the smallest singular value in Σ.

- Solution has unit norm because V is orthonormal

Summary: least-squares solutions to linear systems

Heterogeneous linear system $A \cdot x = b$

- compute pseudo inverse $(A^\top A)^{-1} A^\top$
- compute solution as $x = (A^\top A)^{-1} A^\top b$

Homogeneous linear system $A \cdot x = 0$

- compute SVD $A = U \cdot \Sigma \cdot V^\top$
- compute solution as x: column of V corresponding to the smallest singular value in Σ

Summary: least-squares solutions to linear systems

Heterogeneous linear system $A \cdot x = b$

- compute pseudo inverse $(A^\top A)^{-1} A^\top$
- compute solution as $x = (A^\top A)^{-1} A^\top b$

In Python, use:
`numpy.linalg.solve`
(generally a bad idea to compute matrix inverses)

Homogeneous linear system $A \cdot x = 0$

- compute SVD $A = U \cdot \Sigma \cdot V^\top$
- compute solution as x: column of V corresponding to the smallest singular value in Σ

In Python, use:
`numpy.linalg.svd`

One more use of SVD: enforcing rank constraints

Problem statement: Given

- an $M \times N$ matrix F,
- an integer $k < \min(M, N)$,

find the matrix F' of rank k that is closest to F in the least-squares sense

$$\min_{\substack{F' \\ \text{rank}(F')=k}} \|F - F'\|^2$$

One more use of SVD: enforcing rank constraints

Problem statement: Given

- an $M \times N$ matrix F,
- an integer $k < \min(M, N)$,

find the matrix F' of rank k that is closest to F in the least-squares sense

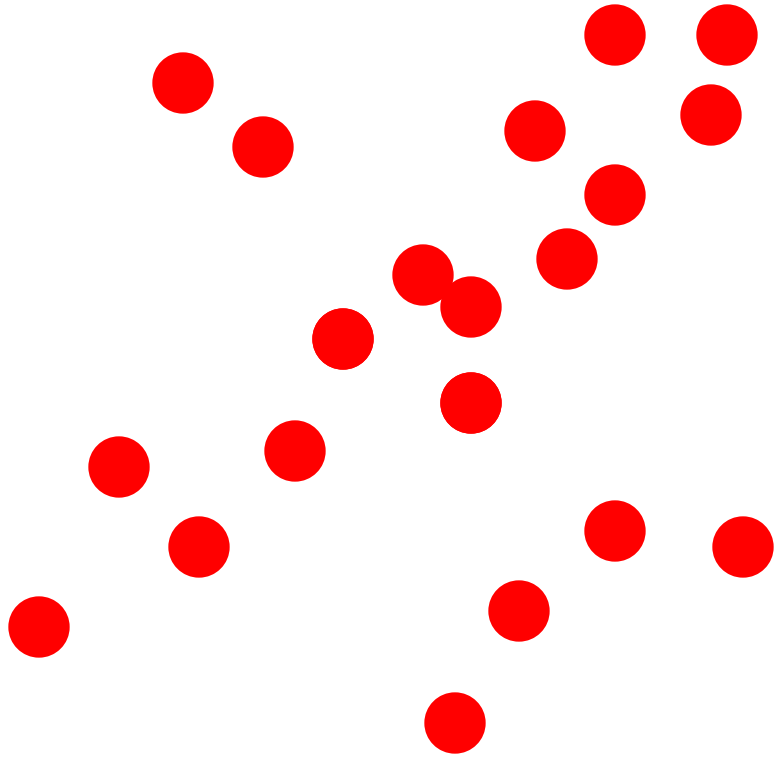
$$\min_{\substack{F' \\ \text{rank}(F')=k}} \|F - F'\|^2$$

Solution:

- compute the SVD $F = U \cdot \Sigma \cdot V^{\top}$
- form a diagonal matrix Σ' by replacing all but the k largest singular values in Σ with 0.
- Compute the solution as $F' = U \cdot \Sigma' \cdot V^{\top}$

Random sample consensus (RANSAC)

Fitting lines
(with outliers)

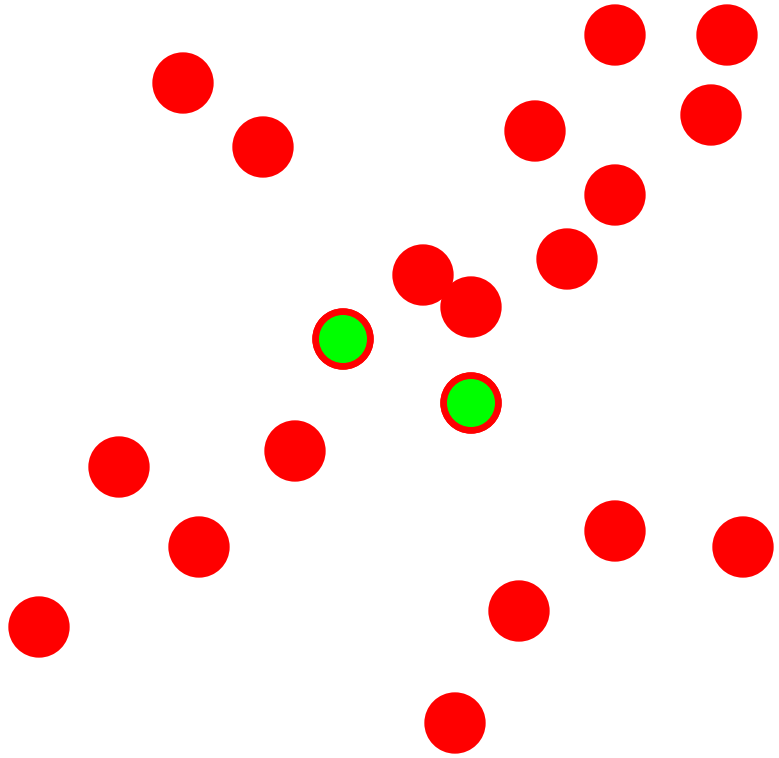


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

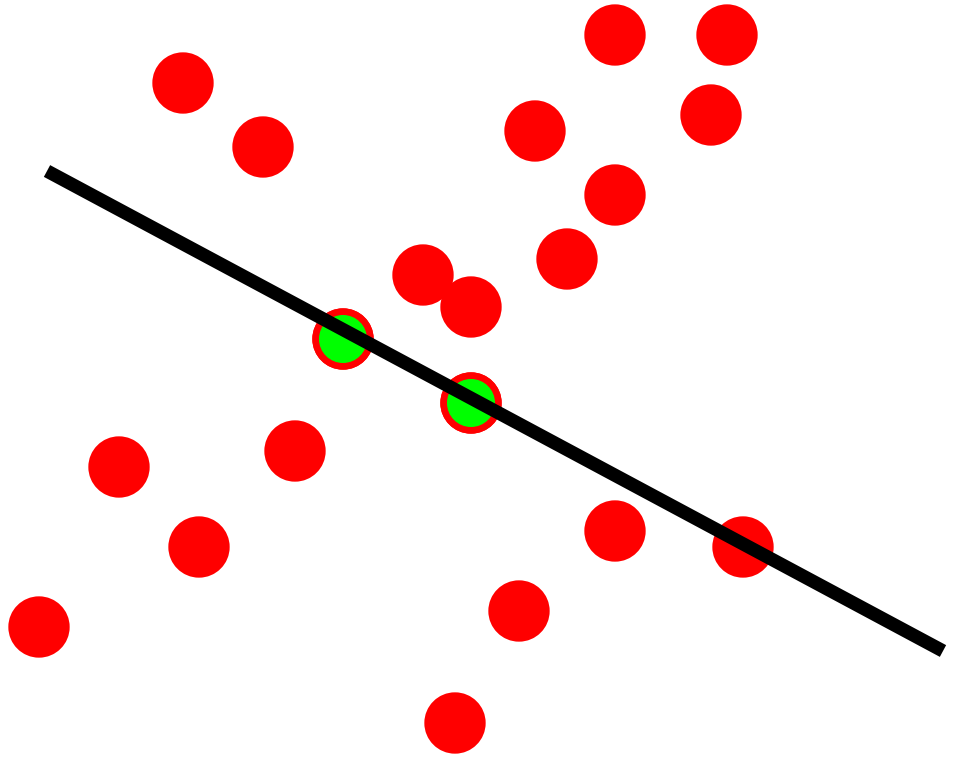


Algorithm:

1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

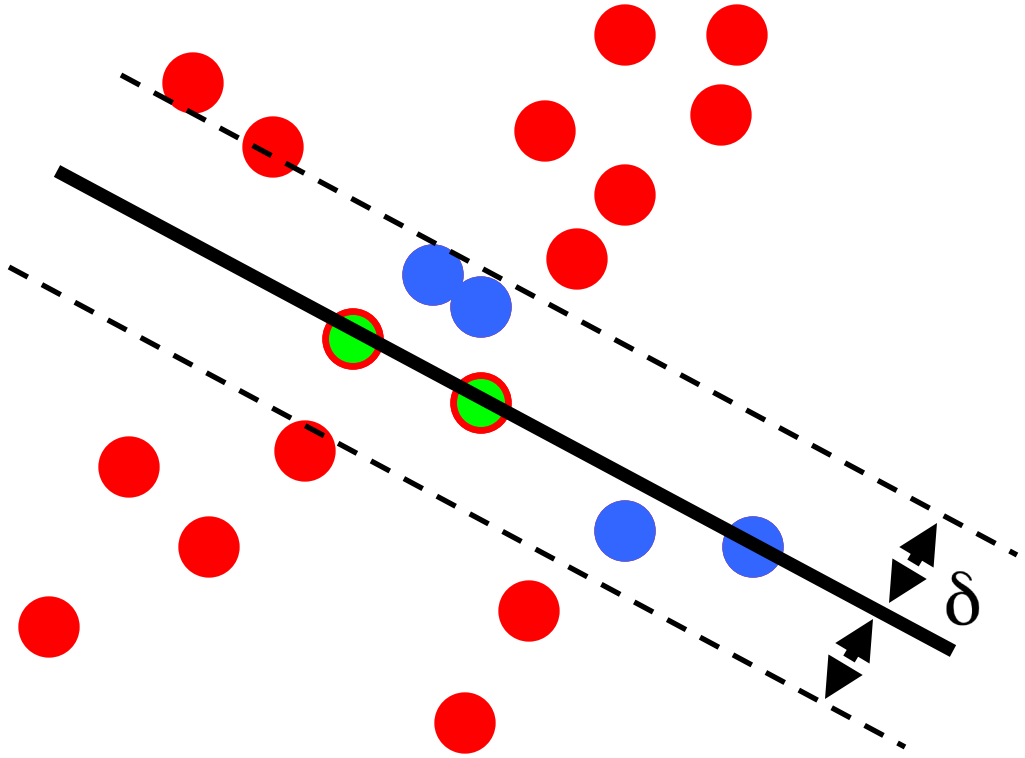


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



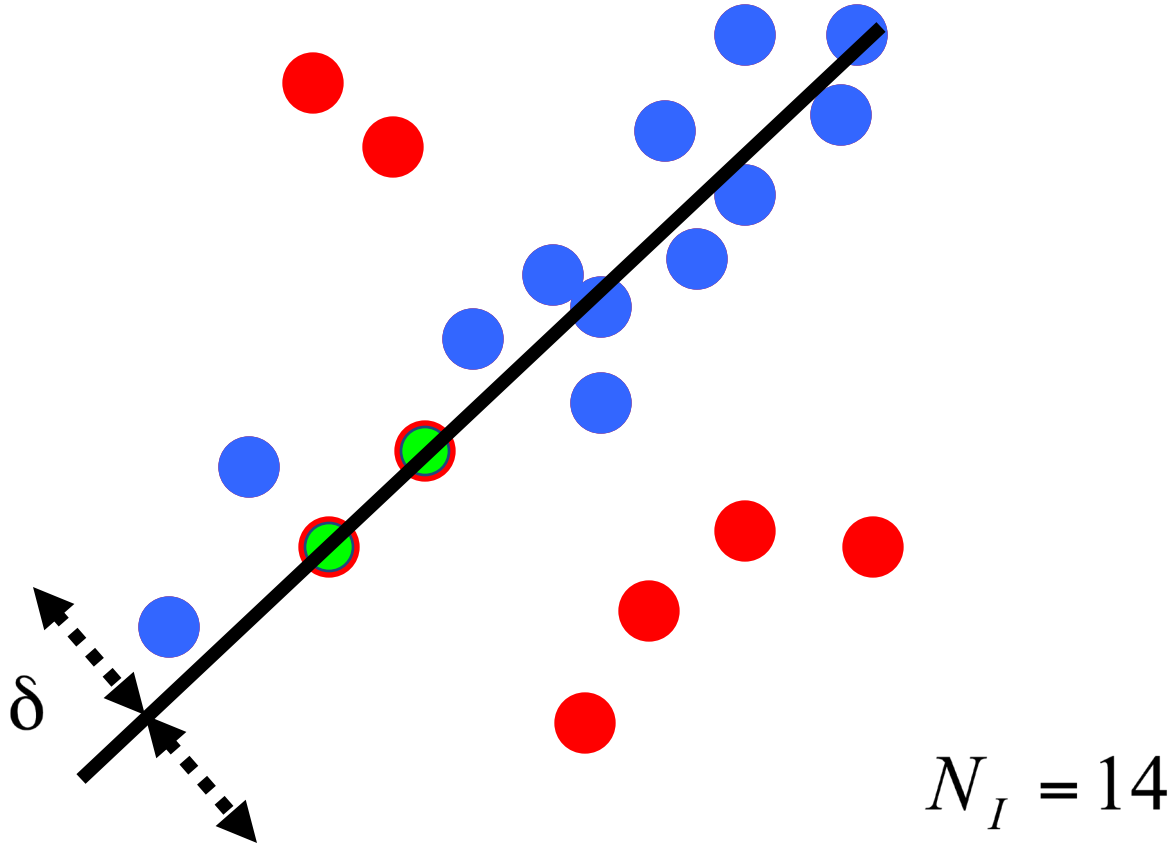
$$N_I = 6$$

Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



Algorithm:

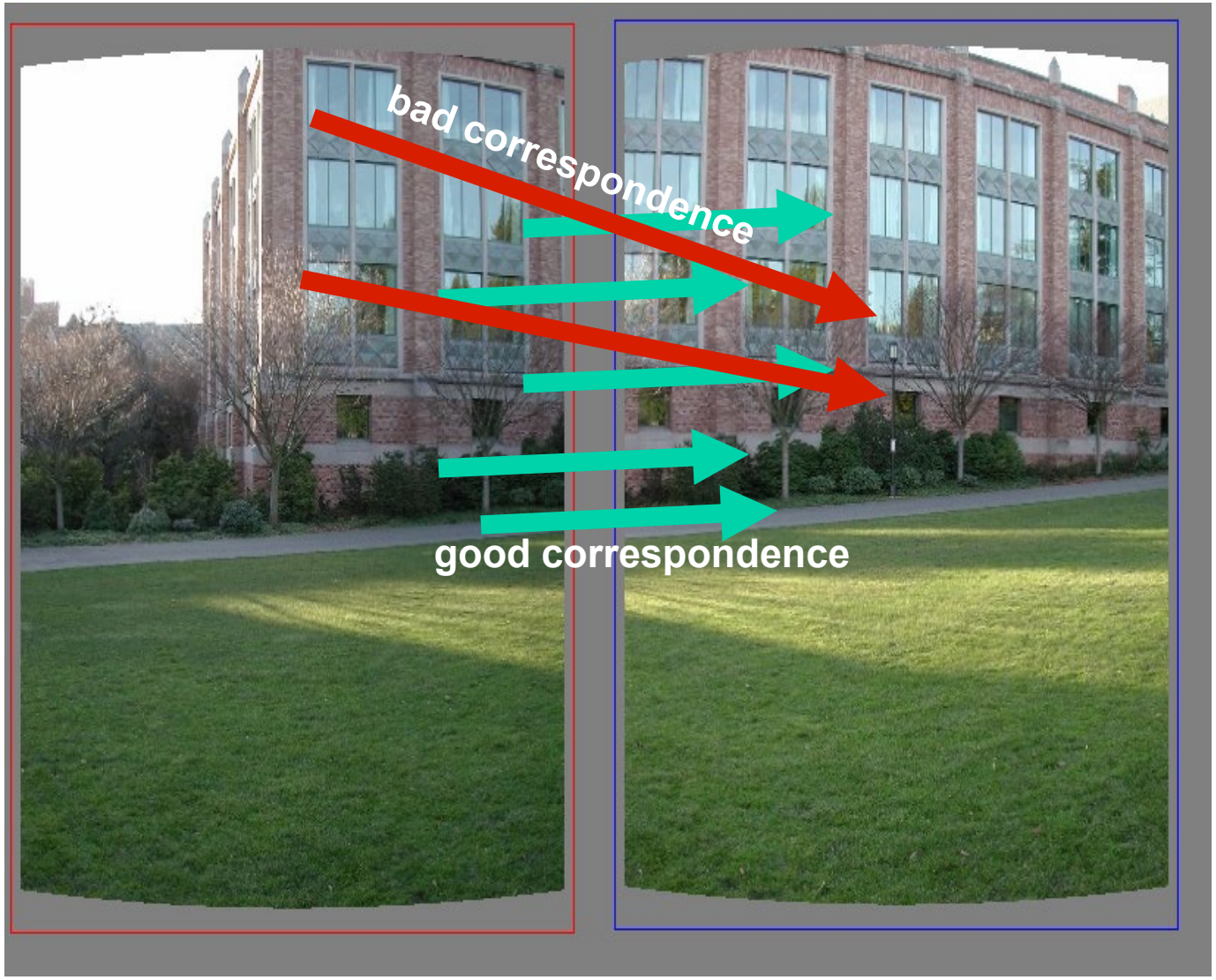
1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

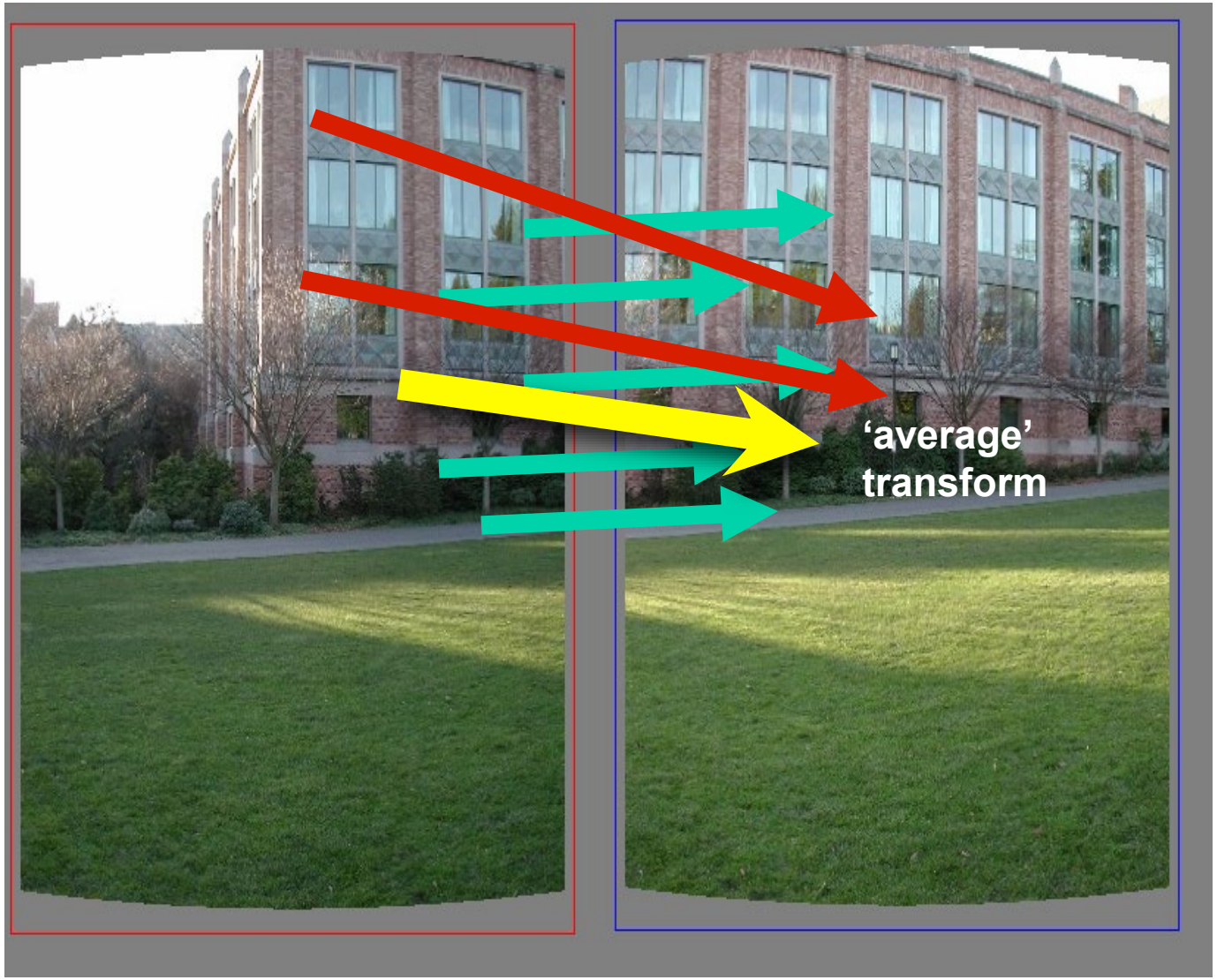
Given two images…

find matching features (e.g., SIFT) and a translation transform

Matched points will usually contain bad correspondences

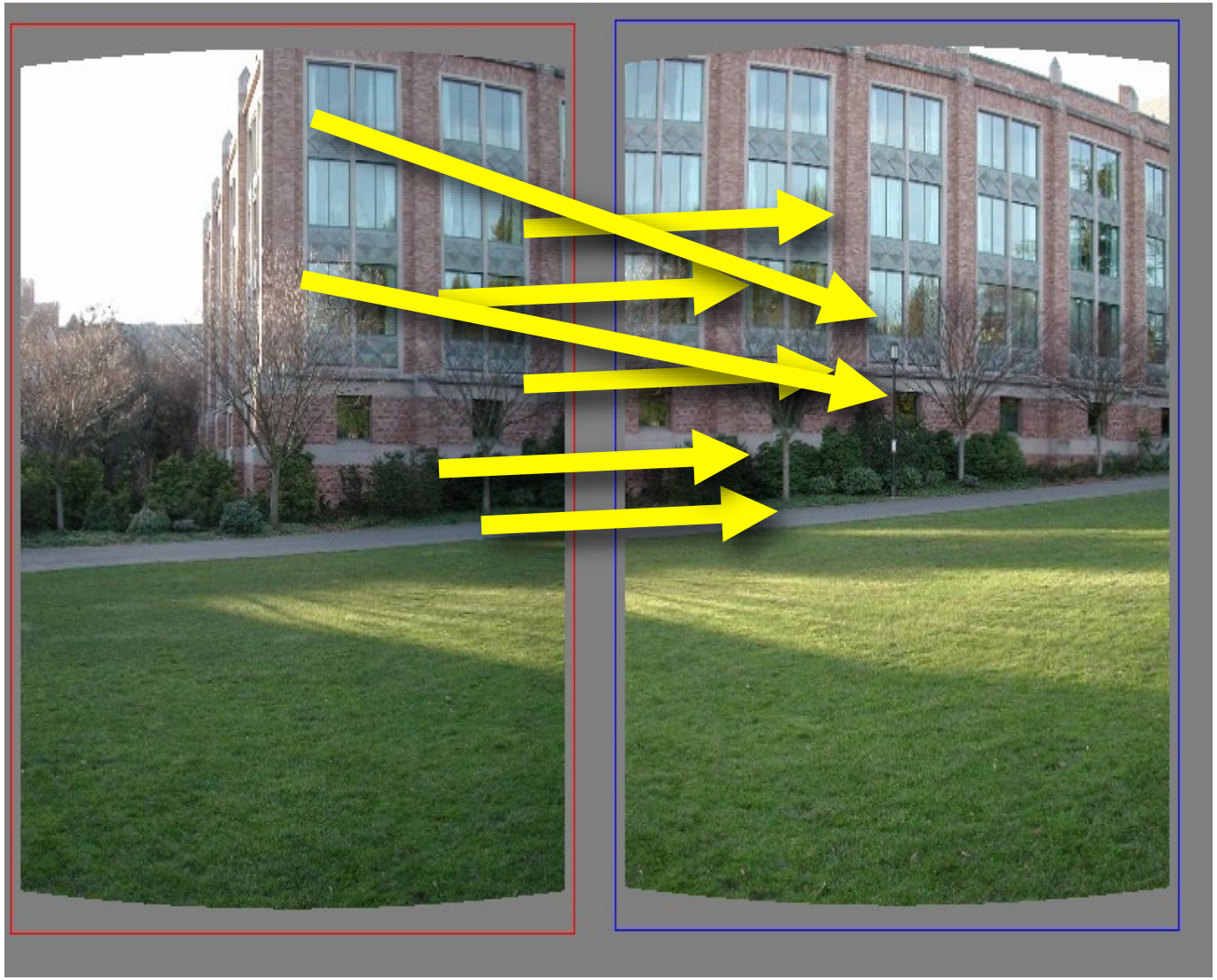


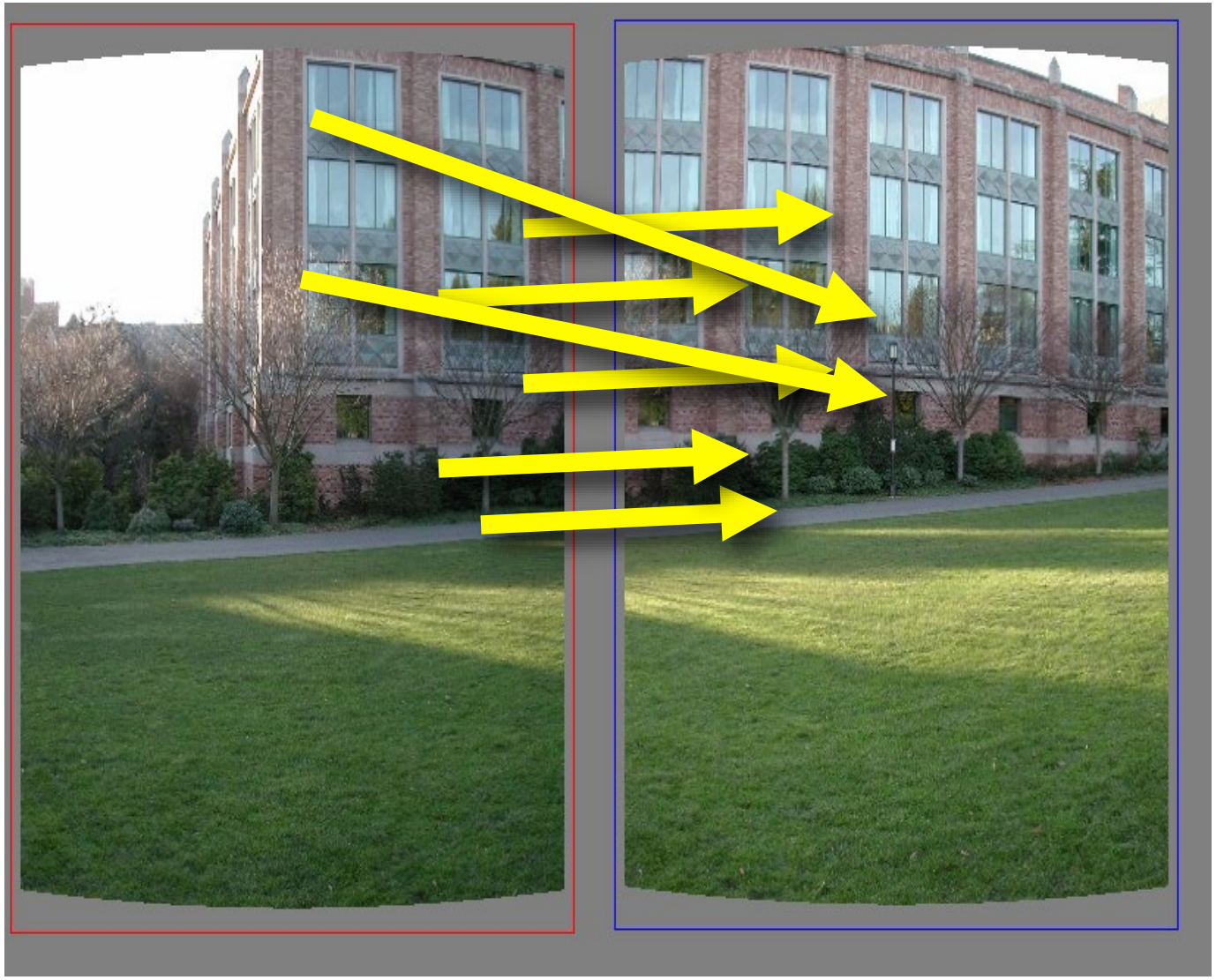
linear least squares will find the ‘average’ transform



solution is corrupted by bad correspondences

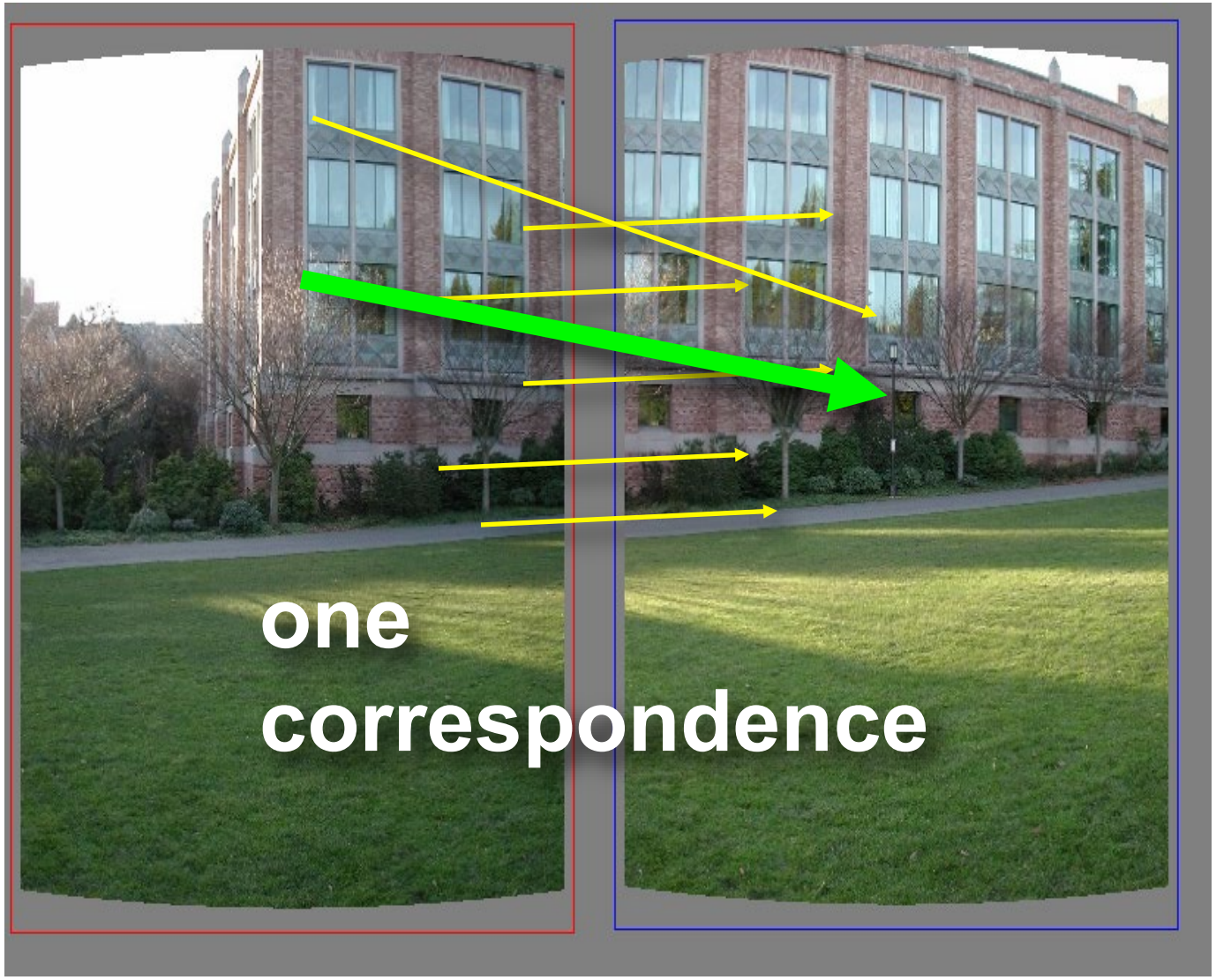
Use RANSAC



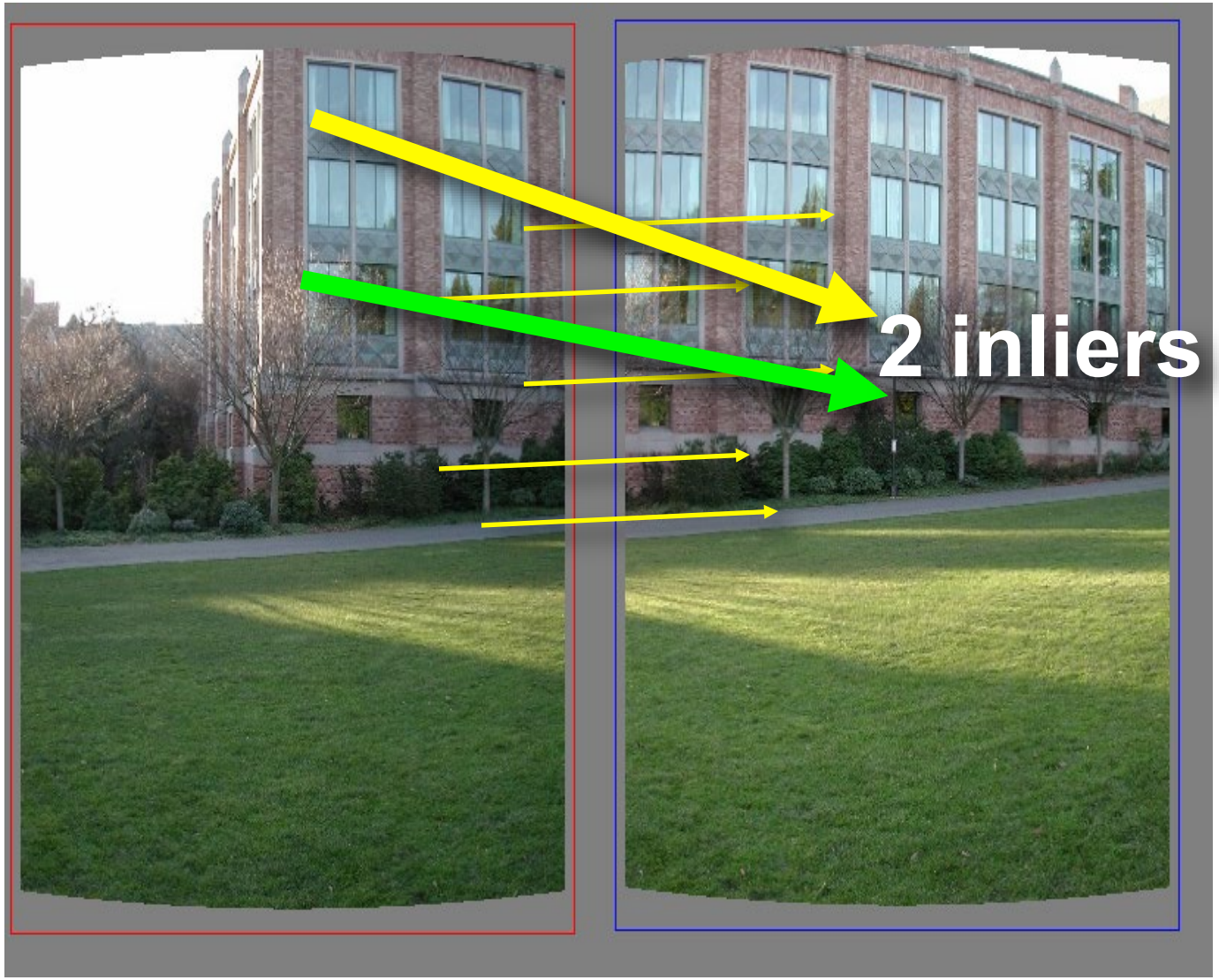


Need only **one correspondence**, to find translation model

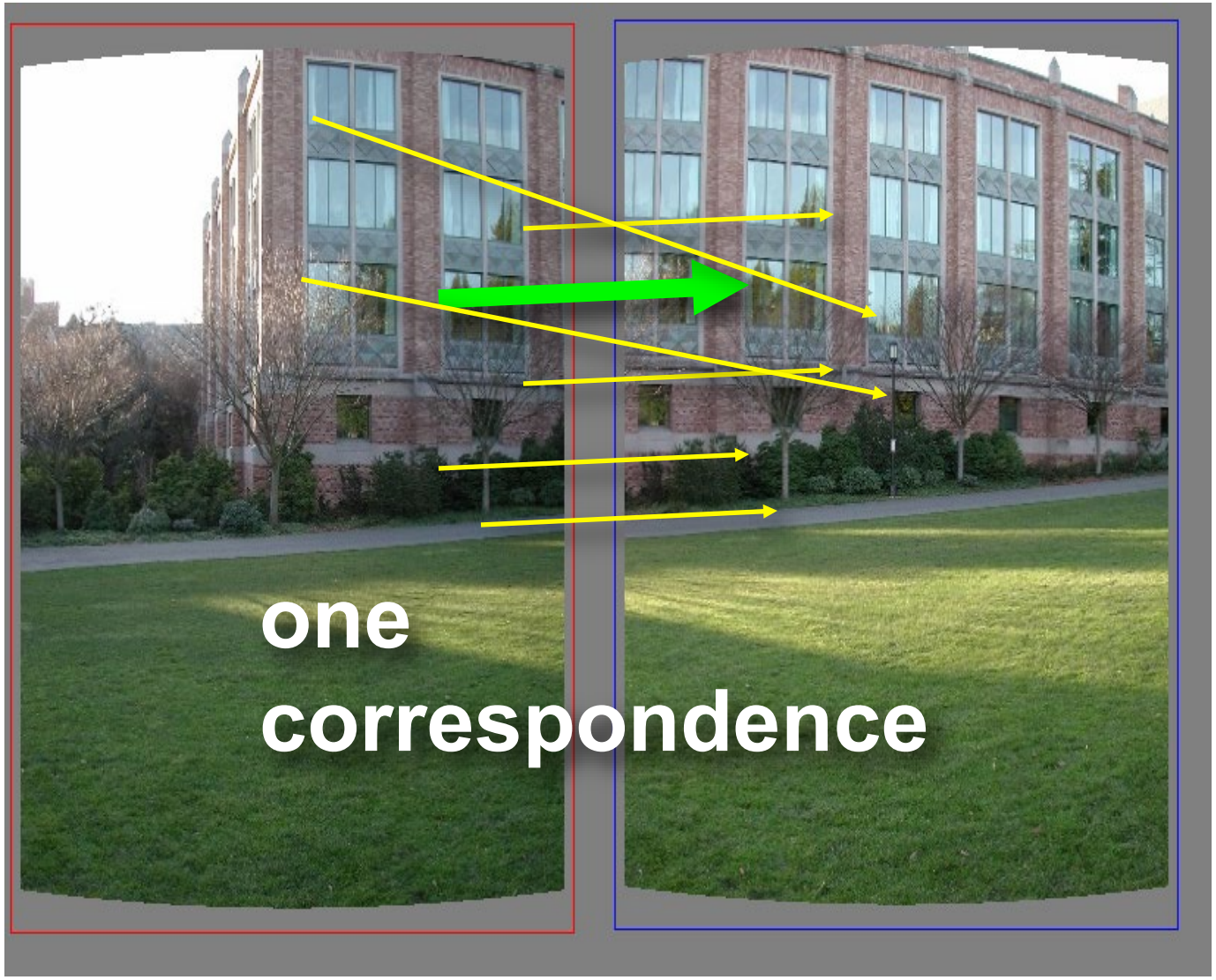
Pick one correspondence, count inliers



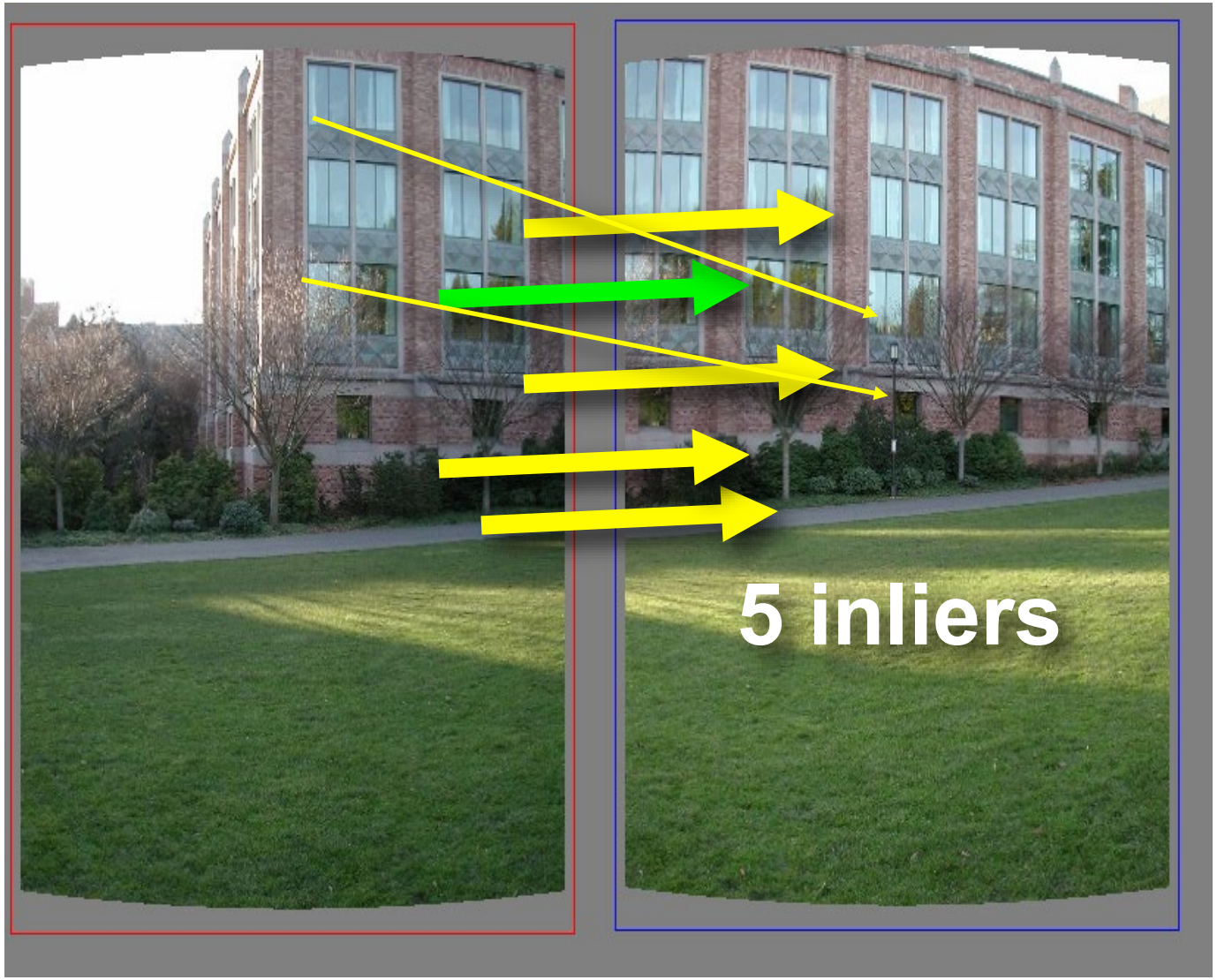
Pick one correspondence, count inliers



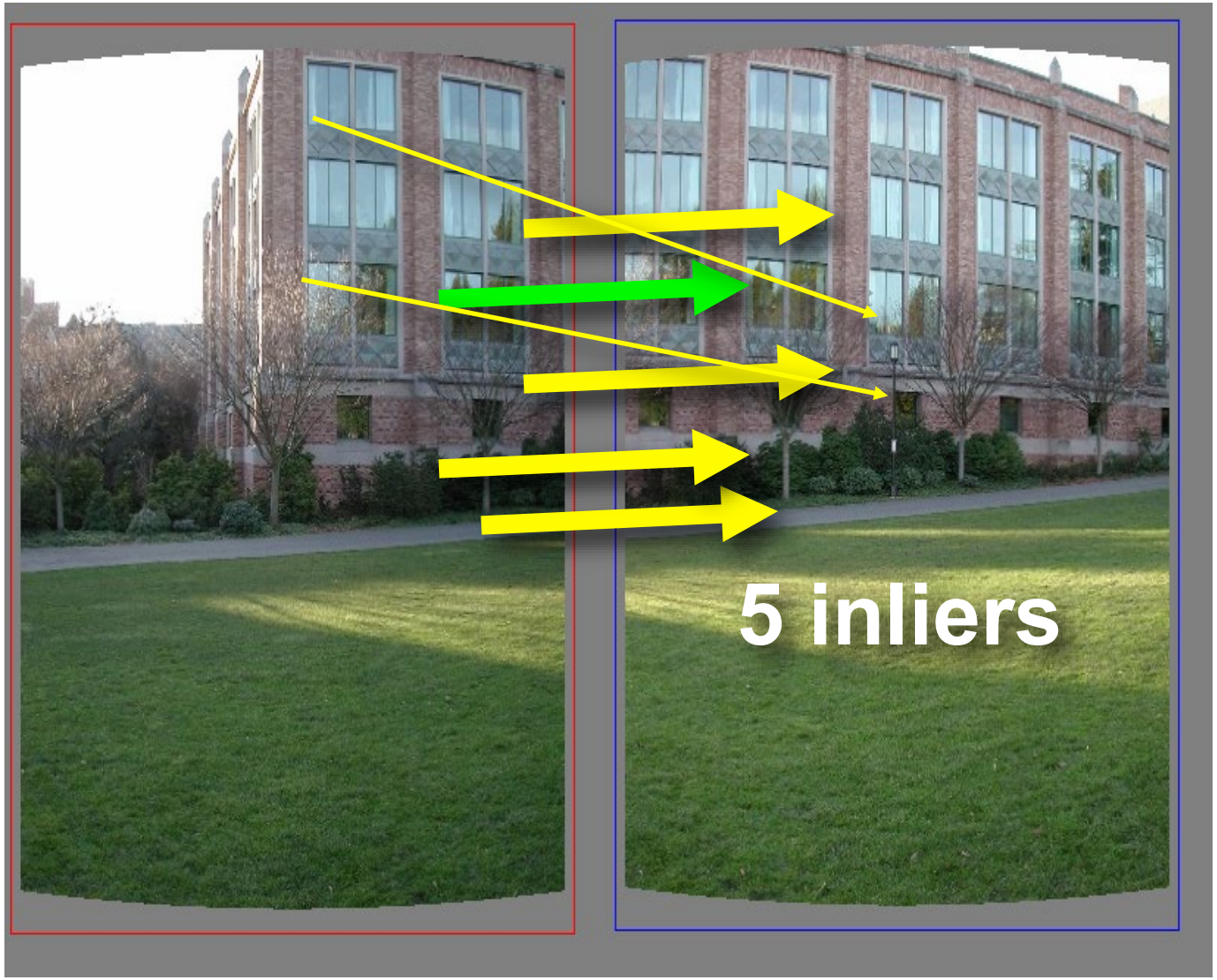
Pick one correspondence, count inliers



Pick one correspondence, count inliers



Pick one correspondence, count inliers



Pick the model with the highest number of inliers!

Estimation using RANSAC

- RANSAC loop
  1. Get required number of point correspondences (randomly)
  2. Fit your model P
  3. Count inliers
  4. Keep P if largest number of inliers
- Recompute P using all inliers

References

Basic reading:

- Szeliski textbook, Section 8.1 (not 8.1.1-8.1.3), Chapter 11, Section 12.2.
- Hartley and Zisserman, Chapters 9, 11, 12.