The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering

Spring 2024

INTRODUCTION TO COMPUTER VISION

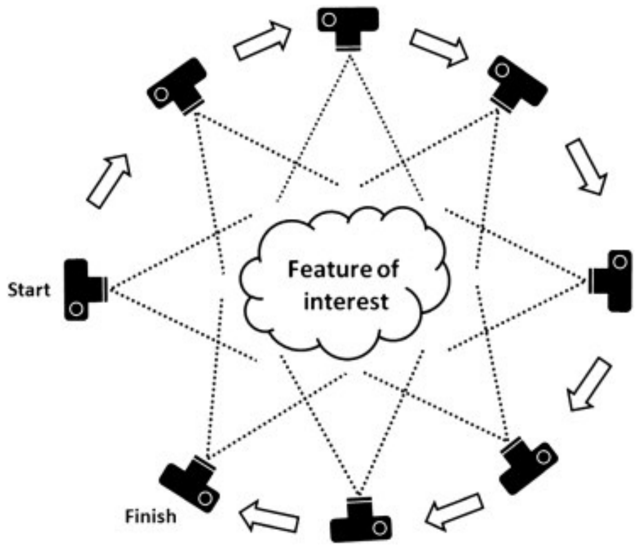
Atlas Wang
Associate Professor, The University of Texas at Austin

Visual Informatics Group@UT Austin
https://vita-group.github.io/


Hello
Vision Pro

“Structure from Motion”

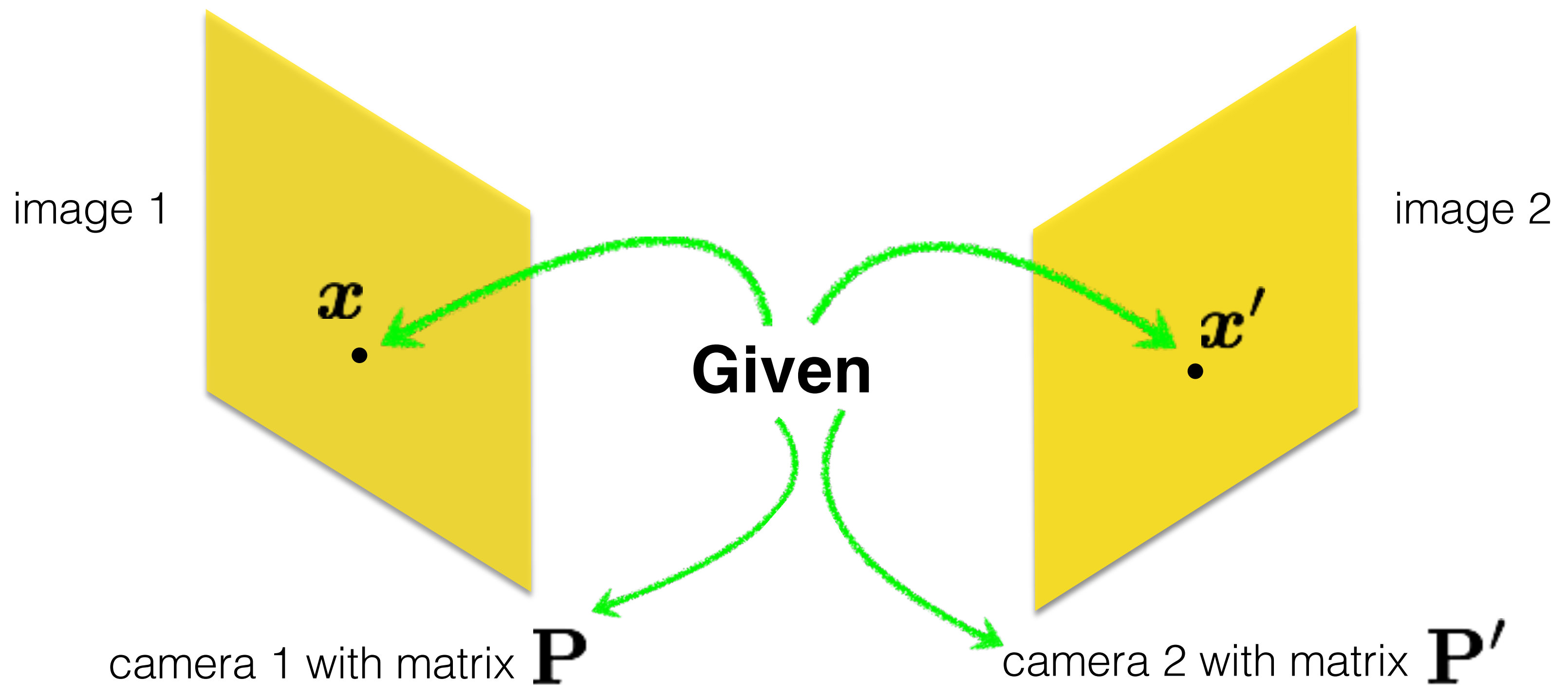
- Humans perceive the 3D structure in their environment by <u>moving around it</u>
  - When the observer moves, objects around them move different amounts depending on their distance from the observer.
  - Even you stand still, most people have two eyes!



- Finding structure from motion presents a similar problem in stereo vision.
  - Estimating three-dimensional structures from two-dimensional image sequences that may be coupled with local motion signals
  - Correspondence between images and reconstruction of 3D object needs to be found

	Structure (scene geometry)	Motion (camera geometry)	**Measurements**
Camera Calibration (a.k.a. pose estimation)	**known**	**estimate**	**3D to 2D correspondences**
Triangulation	**estimate**	**known**	**2D to 2D correspondences**
Reconstruction (including epipolar)	**estimate**	**estimate**	**2D to 2D correspondences**

Triangulation (Two-view geometry)

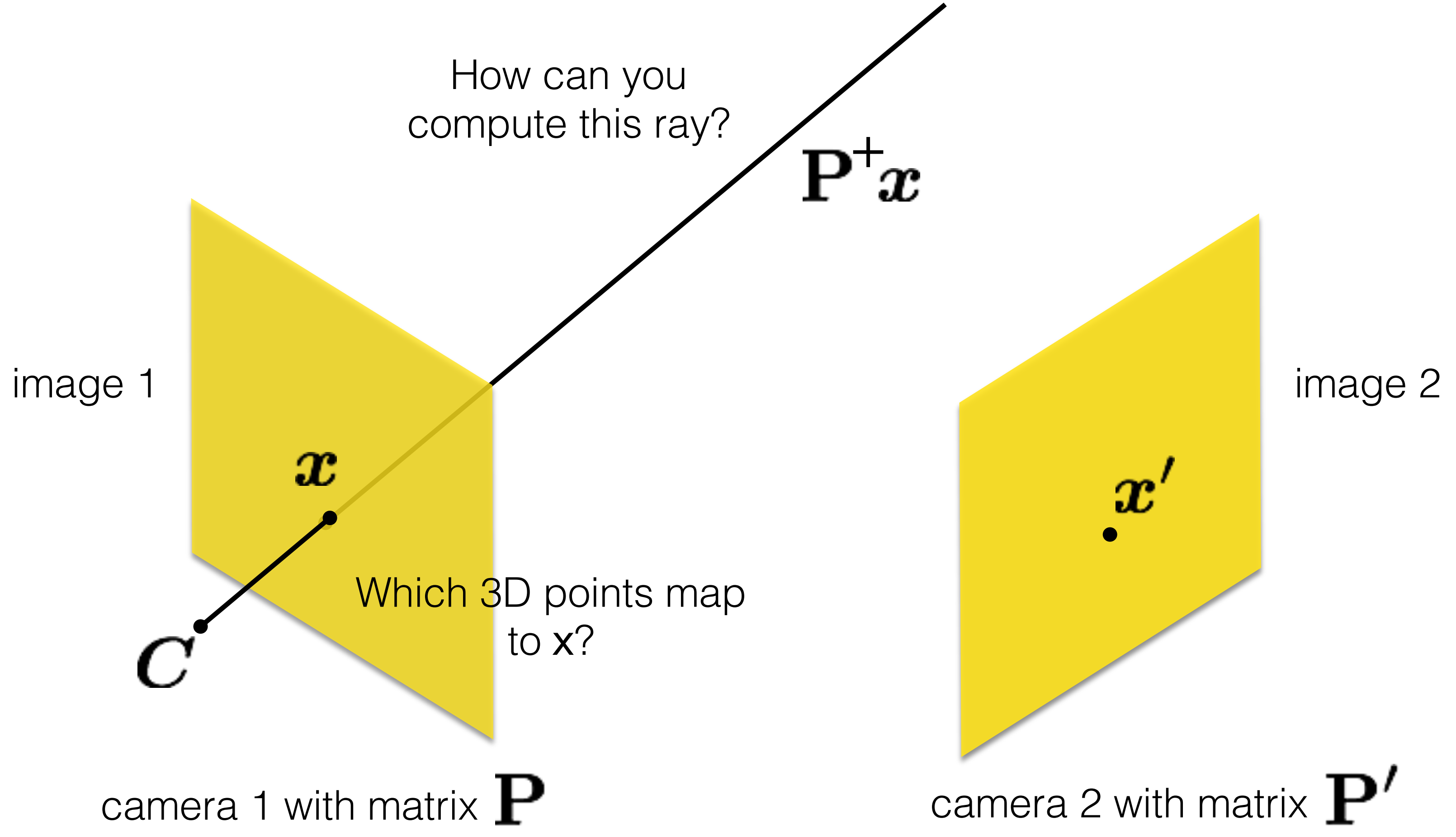


Triangulation

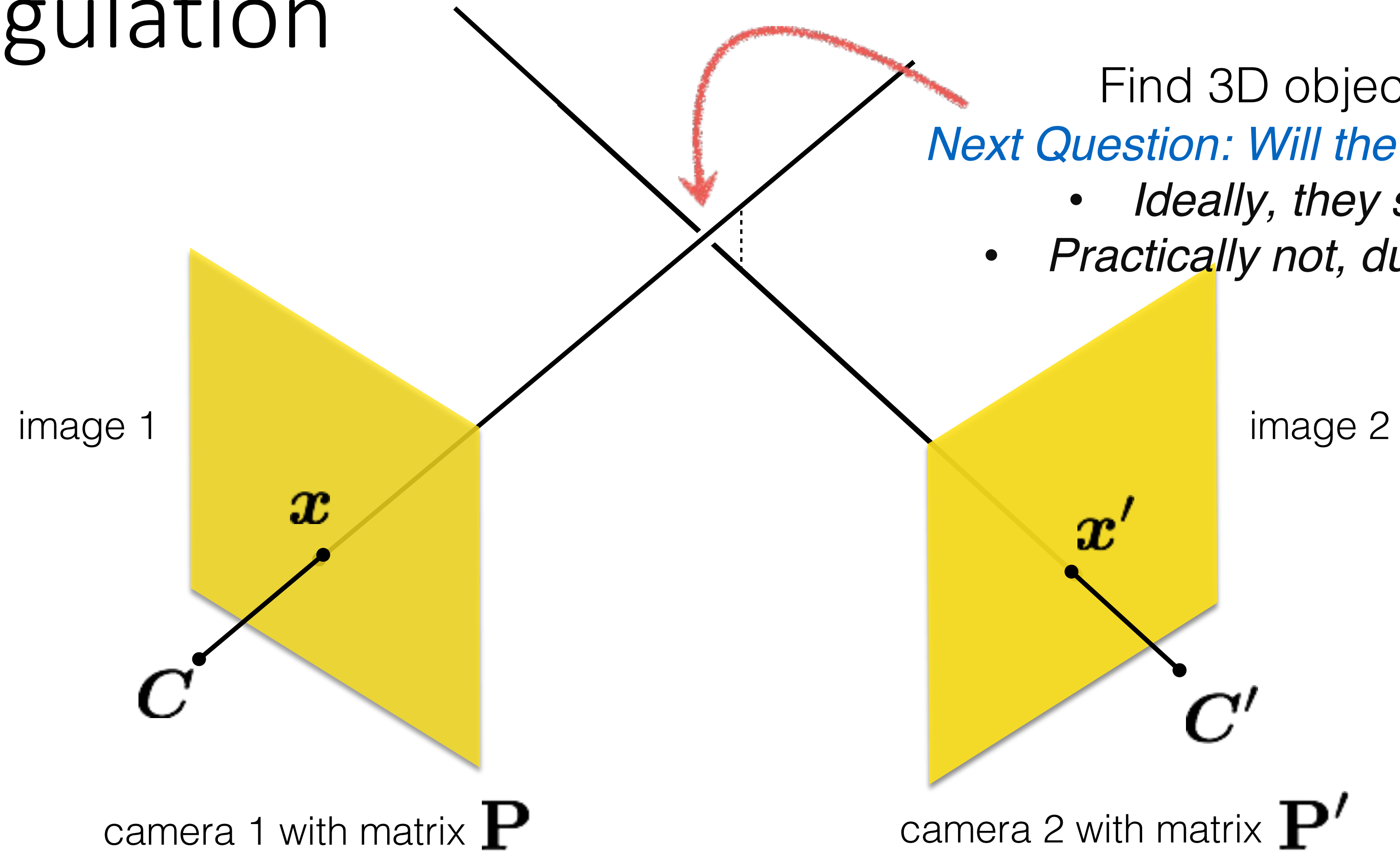
Create two points on the ray:
1) find the camera center; and
2) apply the pseudo-inverse of P on x.

Then connect the two points.

This procedure is called backprojection



Triangulation



Find 3D object point

Next Question: Will the lines intersect?

- *Ideally, they should…*
- *Practically not, due to noise…*

Triangulation

Given a set of (noisy) matched points

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

and camera matrices

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

Estimate the 3D point

$$\mathbf{X}$$

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

known known

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

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

(homogeneous coordinate)

This is a similarity relation because it involves homogeneous coordinates

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

(homogeneous coordinate with a "scale")

Same ray direction but differs by a scale factor

Question: why not directly using homogenous coordinate here?

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

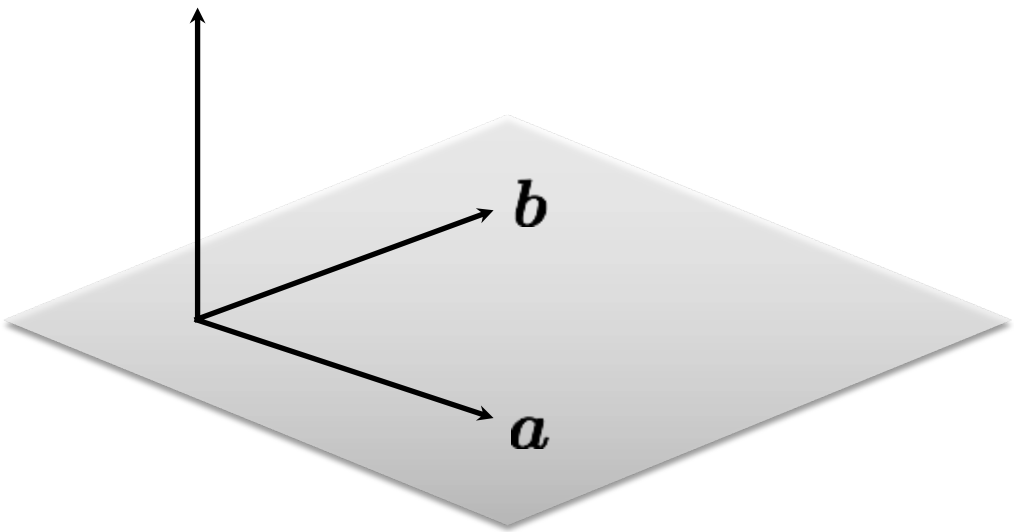
How do we solve for unknowns in a similarity relation?
(e.g., how to remove the unknown scale?)

Linear algebra reminder: cross product

Vector (cross) product

takes two vectors and returns a vector <u>perpendicular</u> to both

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



$$\boldsymbol{a} \times \boldsymbol{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix}$$

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

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

remember this!!!

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

Linear algebra reminder: cross product

Cross product

$$\boldsymbol{a} \times \boldsymbol{b} = \left[\begin{array}{c} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{array} \right]$$

Can also be written as a matrix multiplication

$$\boldsymbol{a} \times \boldsymbol{b} = [\boldsymbol{a}]_\times \boldsymbol{b} = \left[\begin{array}{ccc} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{array} \right] \left[\begin{array}{c} b_1 \\ b_2 \\ b_3 \end{array} \right]$$

Skew symmetric

Back to triangulation

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

Same direction but differs by a scale factor

How can we rewrite this using vector products?

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

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)

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

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

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

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

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

Using the fact that the cross product should be zero

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

$$\left[\begin{array}{c} y\boldsymbol{p}_3^\top \boldsymbol{X} - \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_1^\top \boldsymbol{X} - x\boldsymbol{p}_3^\top \boldsymbol{X} \\ x\boldsymbol{p}_2^\top \boldsymbol{X} - y\boldsymbol{p}_1^\top \boldsymbol{X} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right]$$

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 [redacted] equations

Using the fact that the cross product should be zero

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

$$\left[\begin{array}{c} y\boldsymbol{p}_3^\top \boldsymbol{X} - \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_1^\top \boldsymbol{X} - x\boldsymbol{p}_3^\top \boldsymbol{X} \\ x\boldsymbol{p}_2^\top \boldsymbol{X} - y\boldsymbol{p}_1^\top \boldsymbol{X} \end{array}\right] = \left[\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right]$$

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
(That shows the inherent ambiguity … every point on the ray is a solution!)

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

Remove third row, and rearrange as system on unknowns

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

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

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

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?

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

How do we solve homogeneous linear system?

Concatenate the 2D points from both images

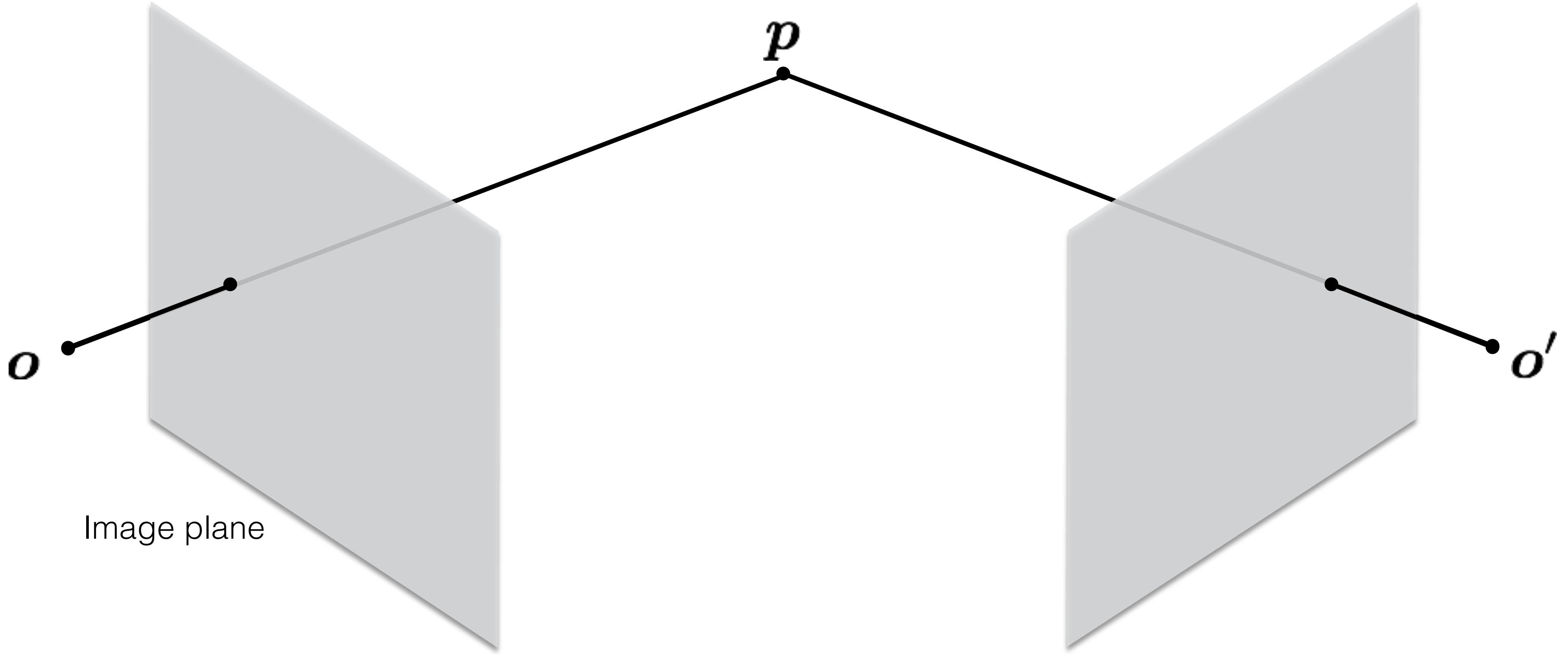
$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

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

How do we solve homogeneous linear system?

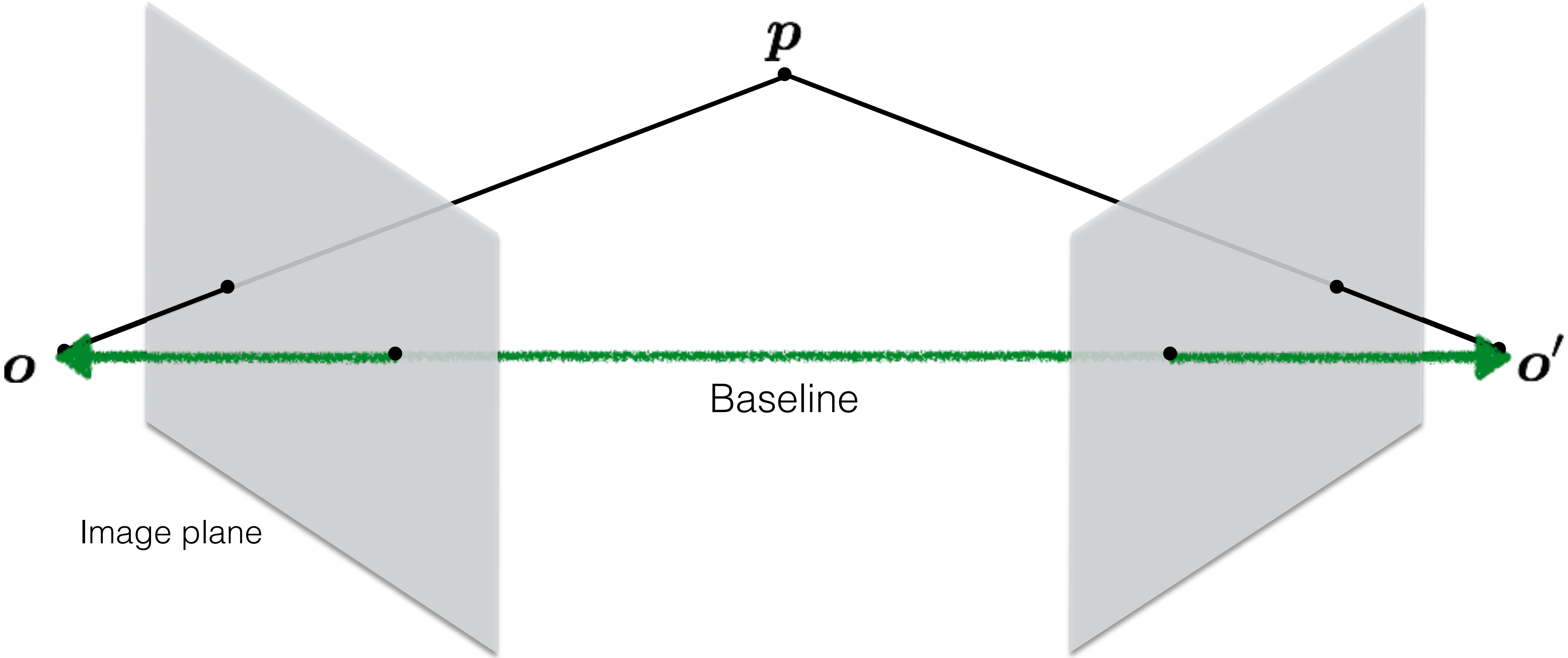
S V D !

Epipolar geometry



Assuming pinhole cameras, given one 2D point on the left image,
where is its counterpart on the right image, that is projected from the same 3D point?

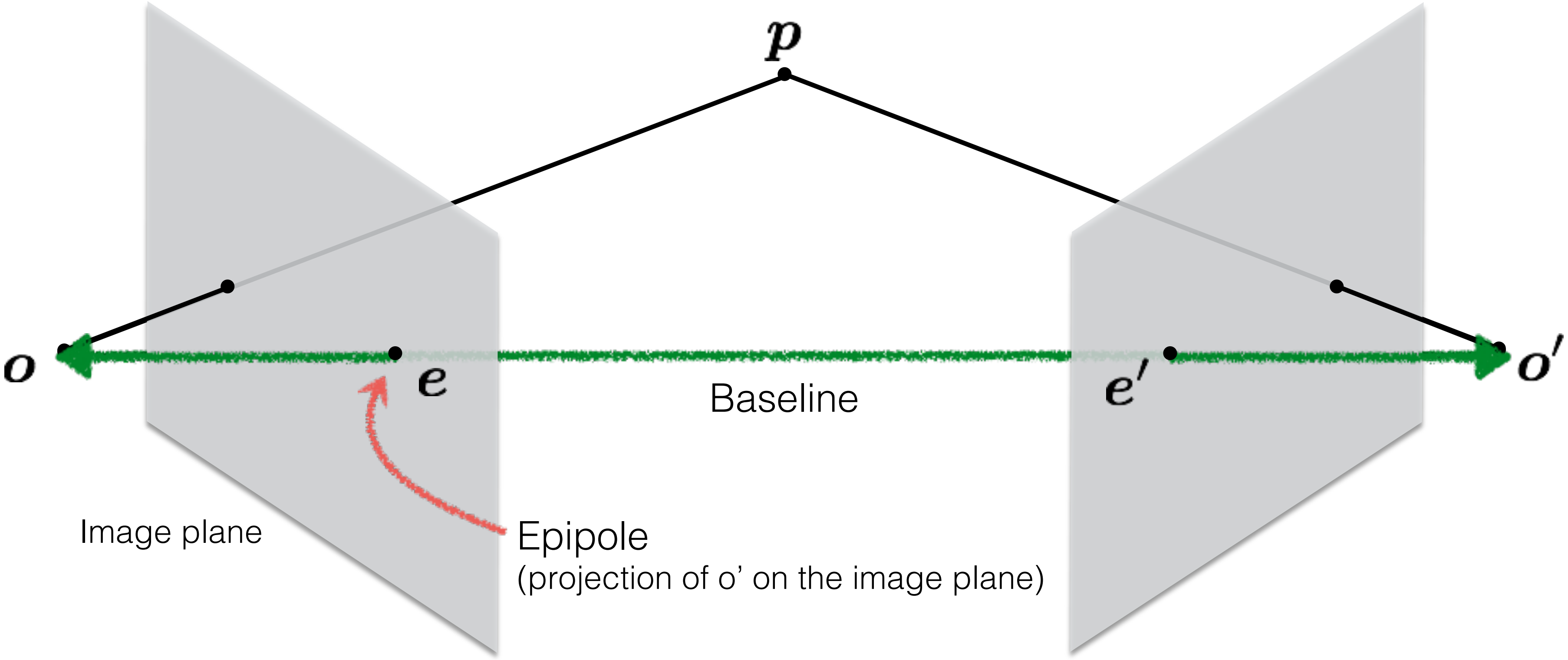
Epipolar geometry



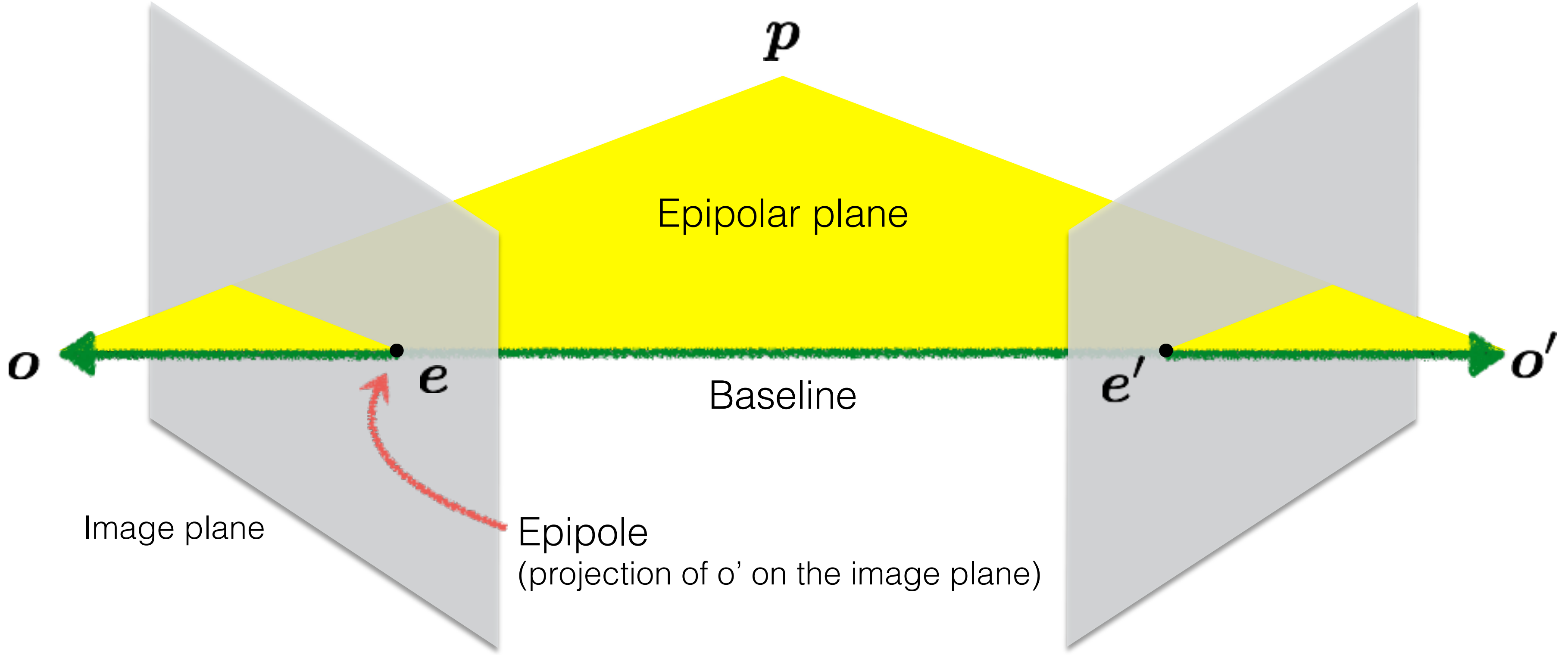
Just like we don't know the actual 3D point, here we don't know either camera center!

Let us just pretend to know them for now… and later they'll become parameters we have to **estimate**

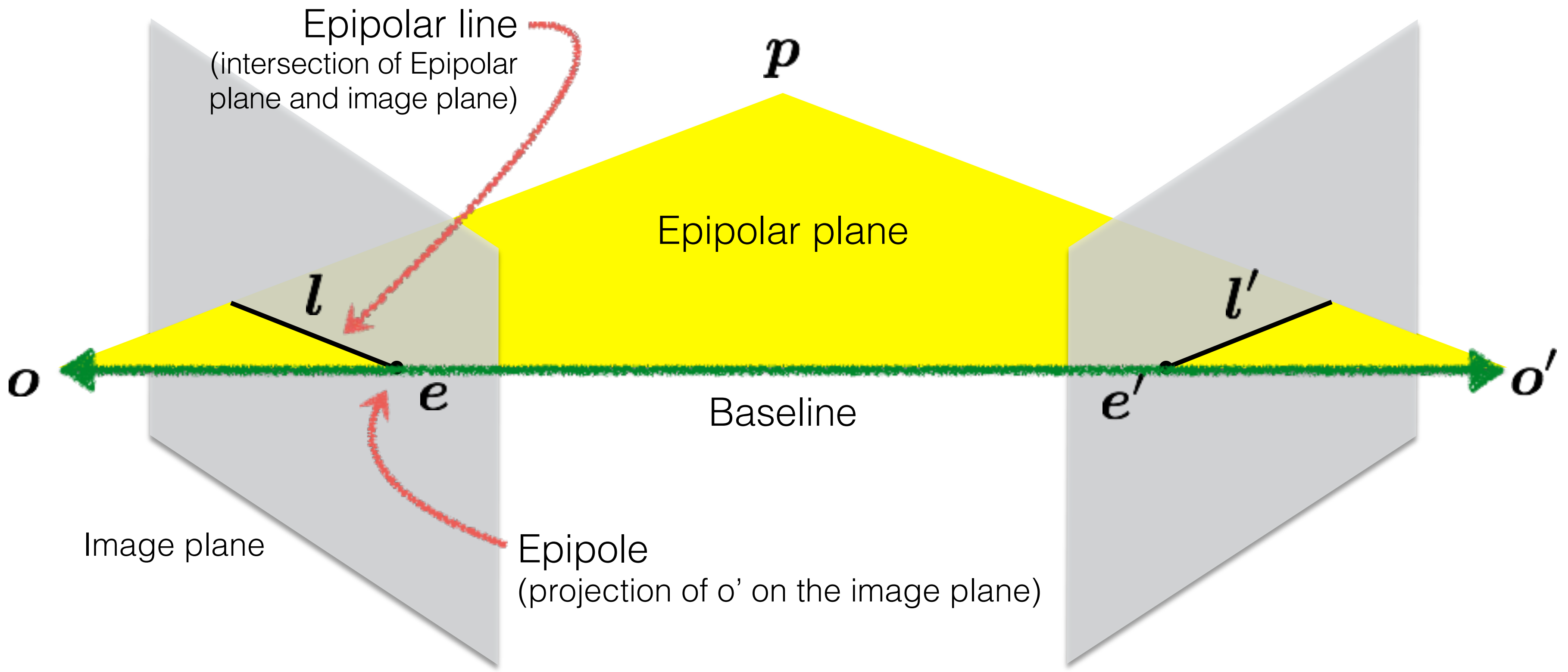
Epipolar geometry



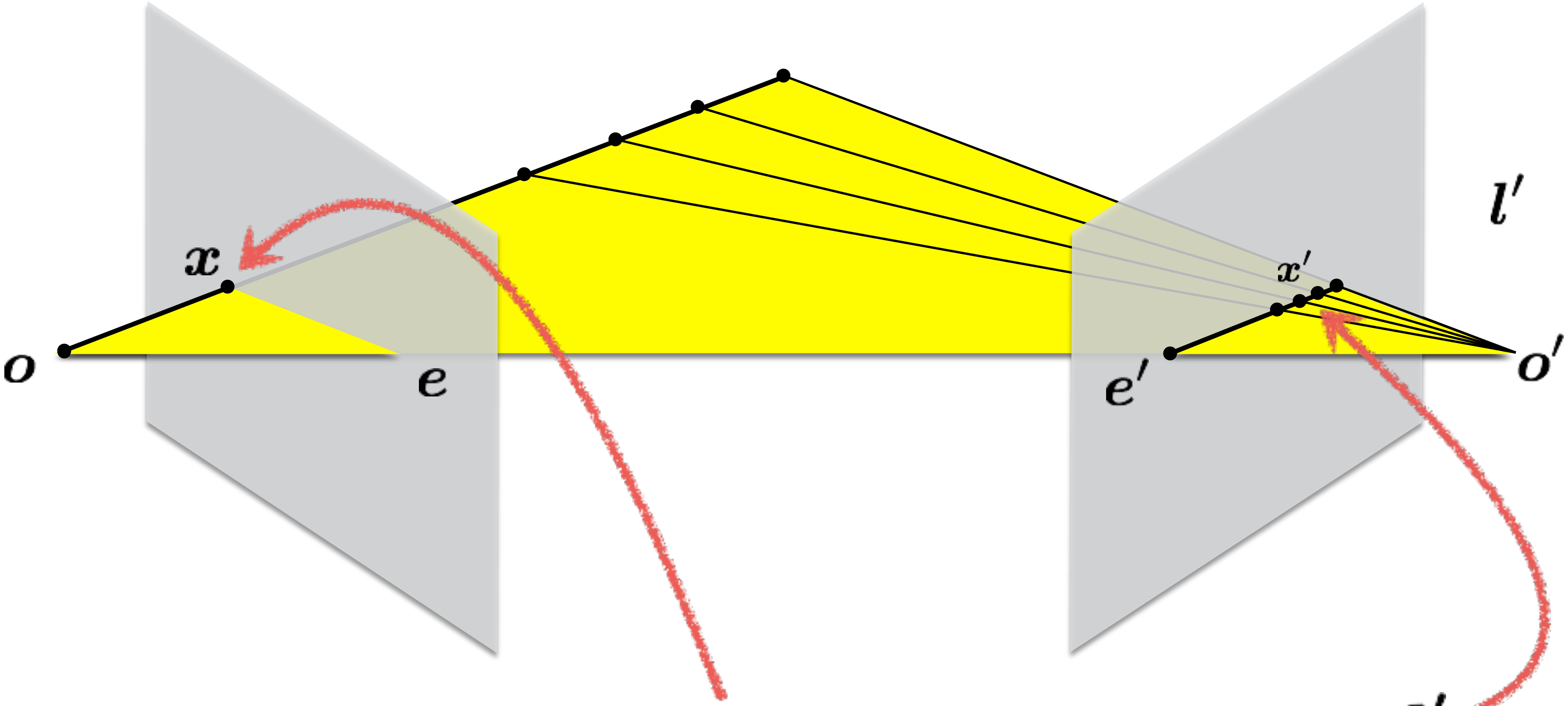
Epipolar geometry



Epipolar geometry



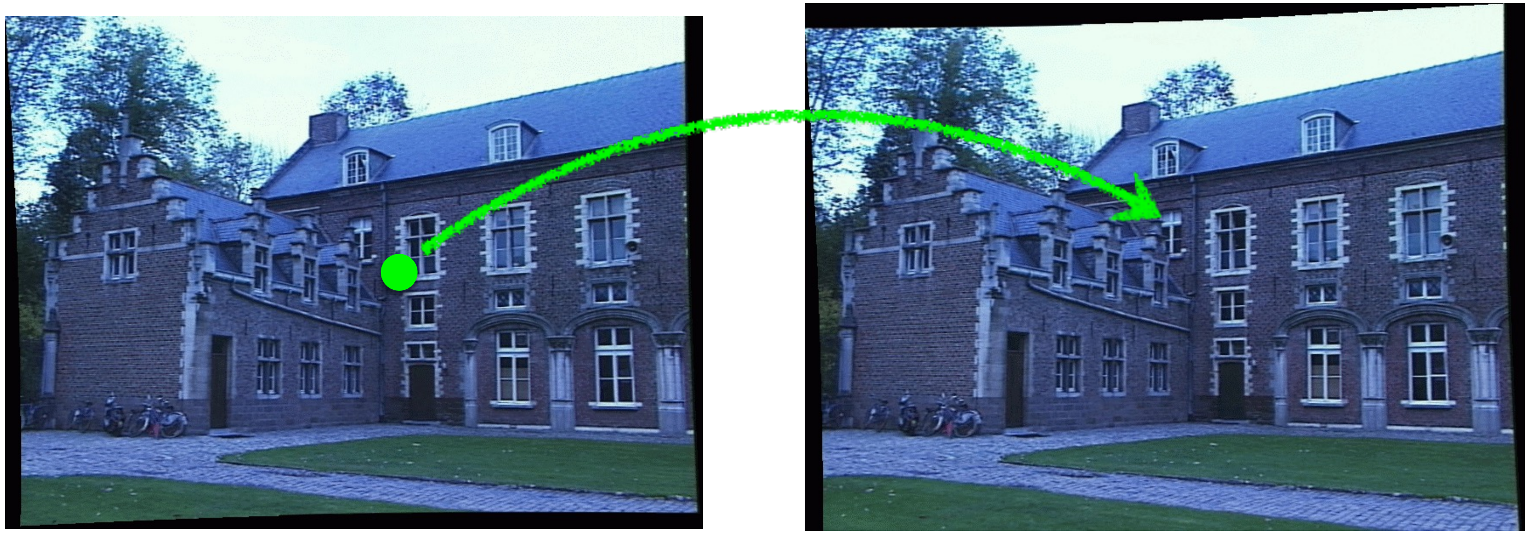
Epipolar constraint



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



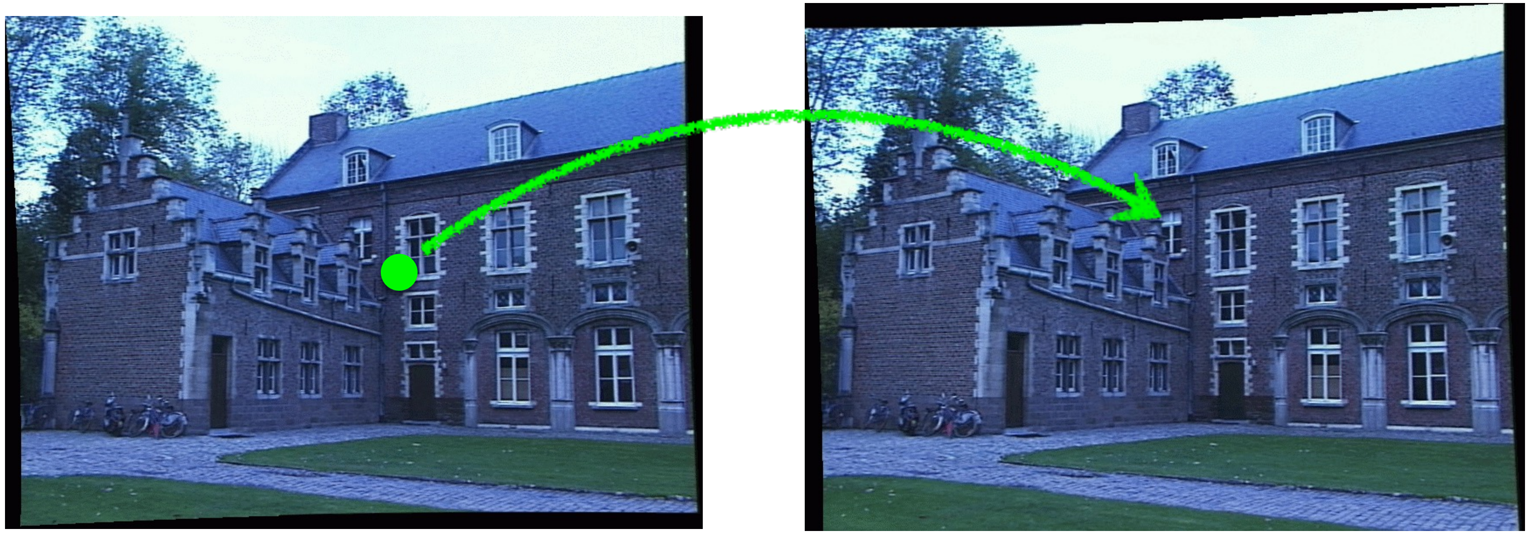
Left image

Right image

How would you do it?

The epipolar constraint is an important concept for stereo vision

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



Left image

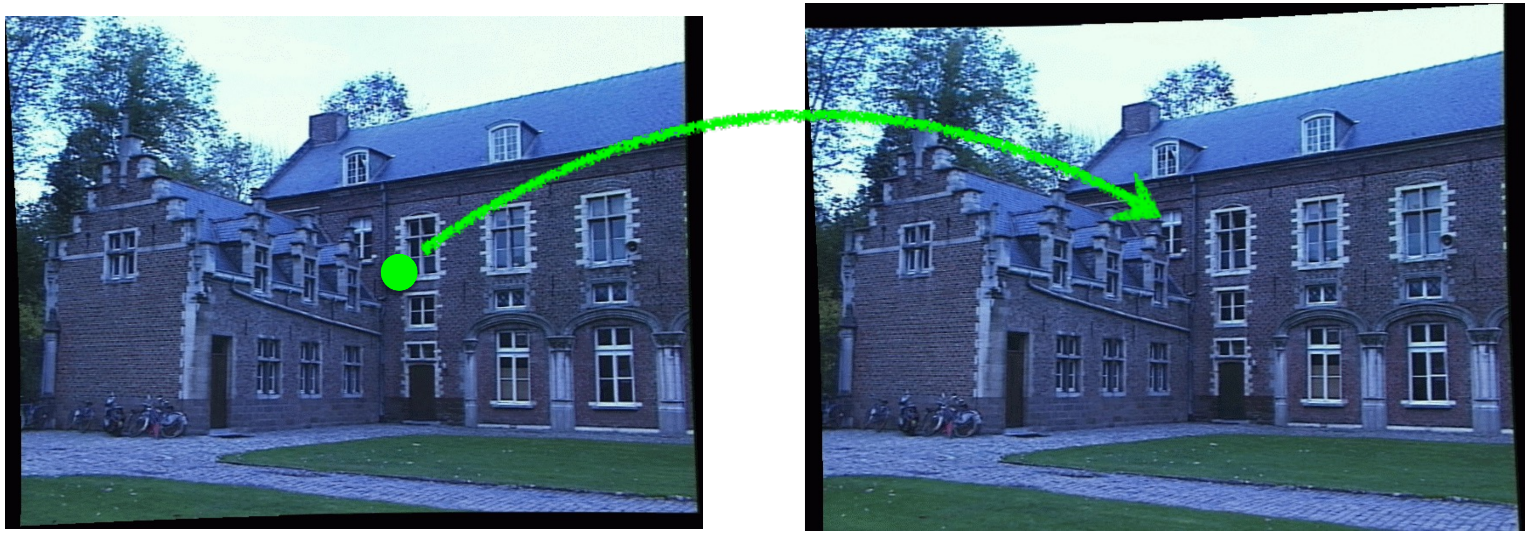
Right image

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

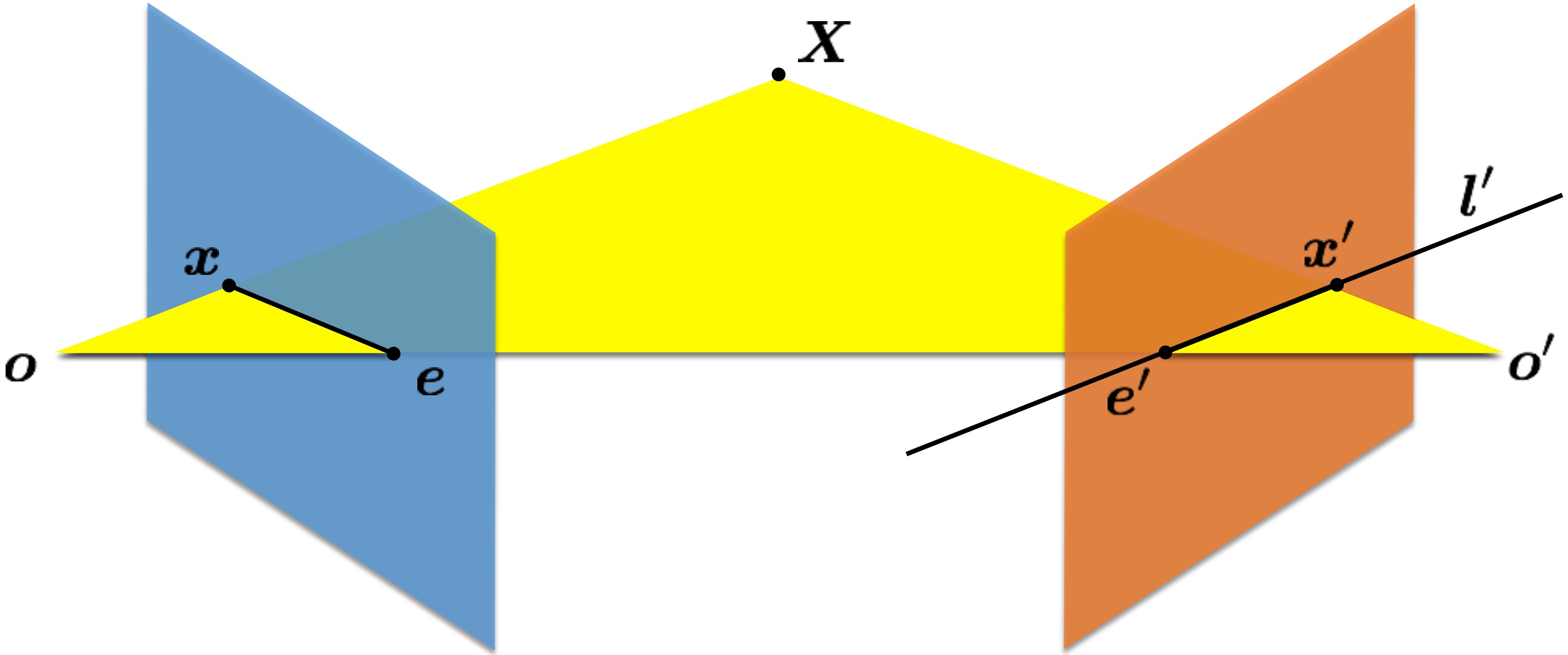
Want to avoid search over entire image

Epipolar constraint reduces search to a single line

How do you compute the epipolar line?

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

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



The Essential Matrix is a 3 x 3 matrix that encodes **epipolar geometry**

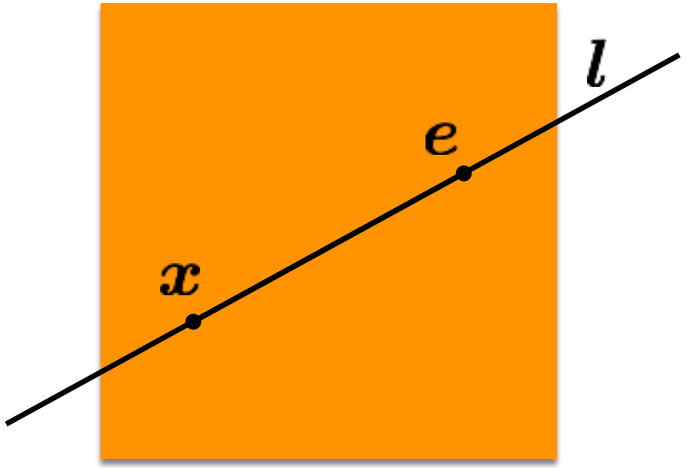
Representing the …

Epipolar Line

$$ax + by + c = 0$$

in vector form

$$\boldsymbol{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

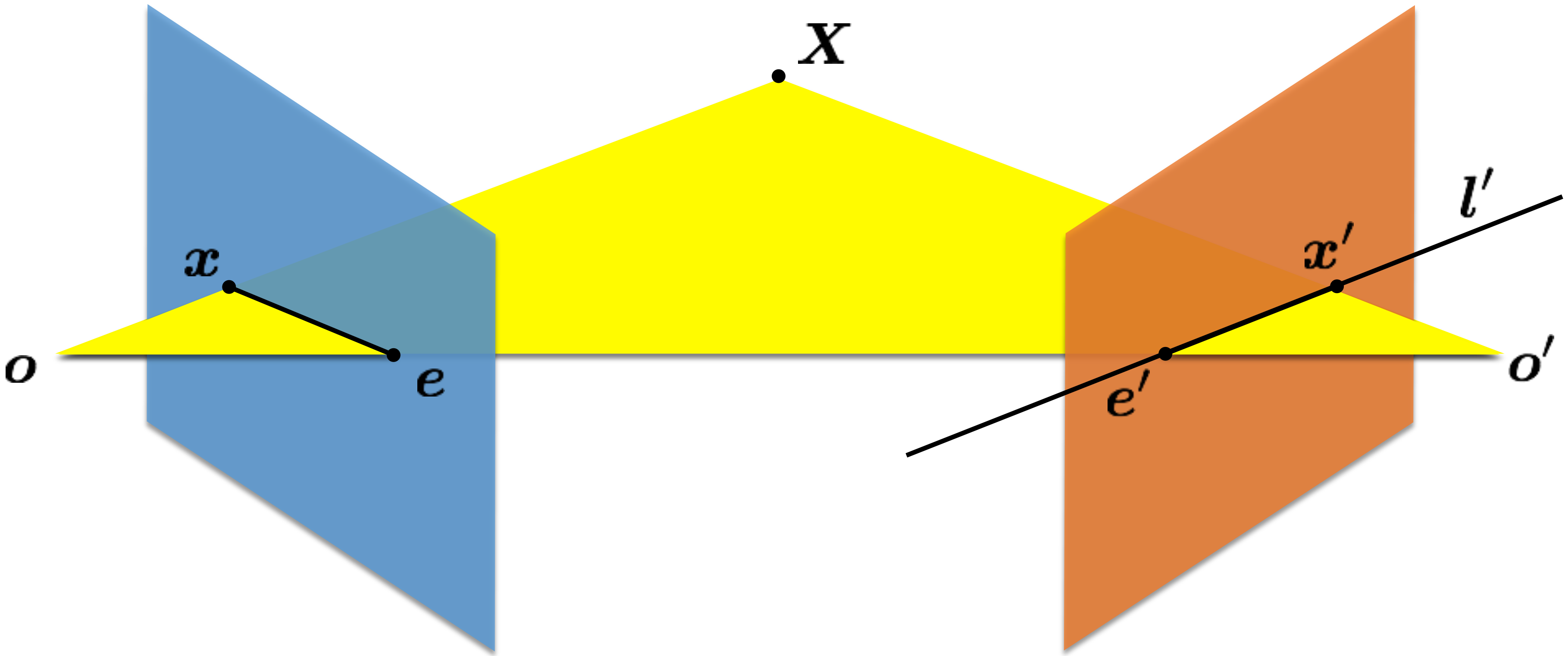


If the point $\boldsymbol{x}$ is on the epipolar line $\boldsymbol{l}$ then

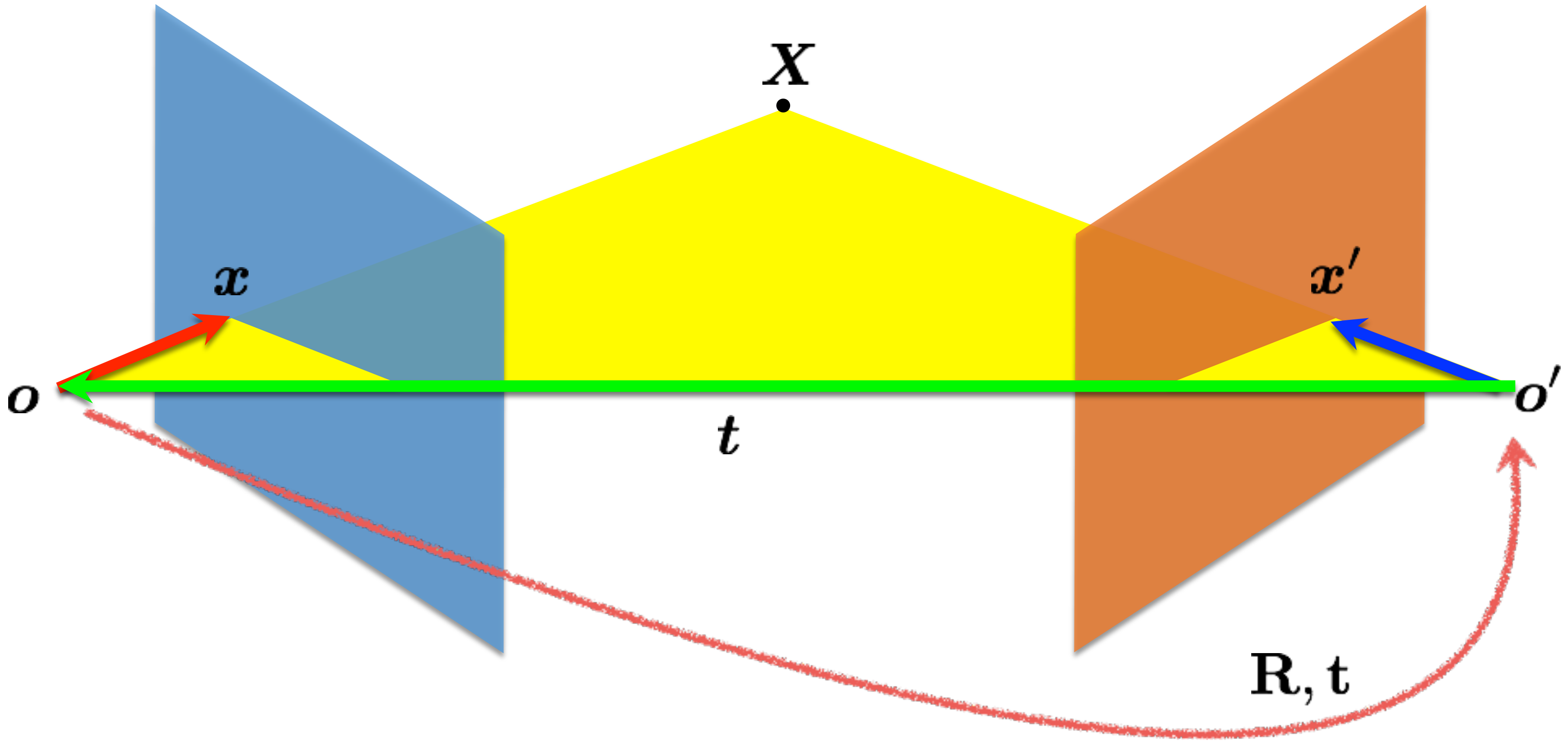
$$\boldsymbol{x}^\top \boldsymbol{l} = \ ?$$

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

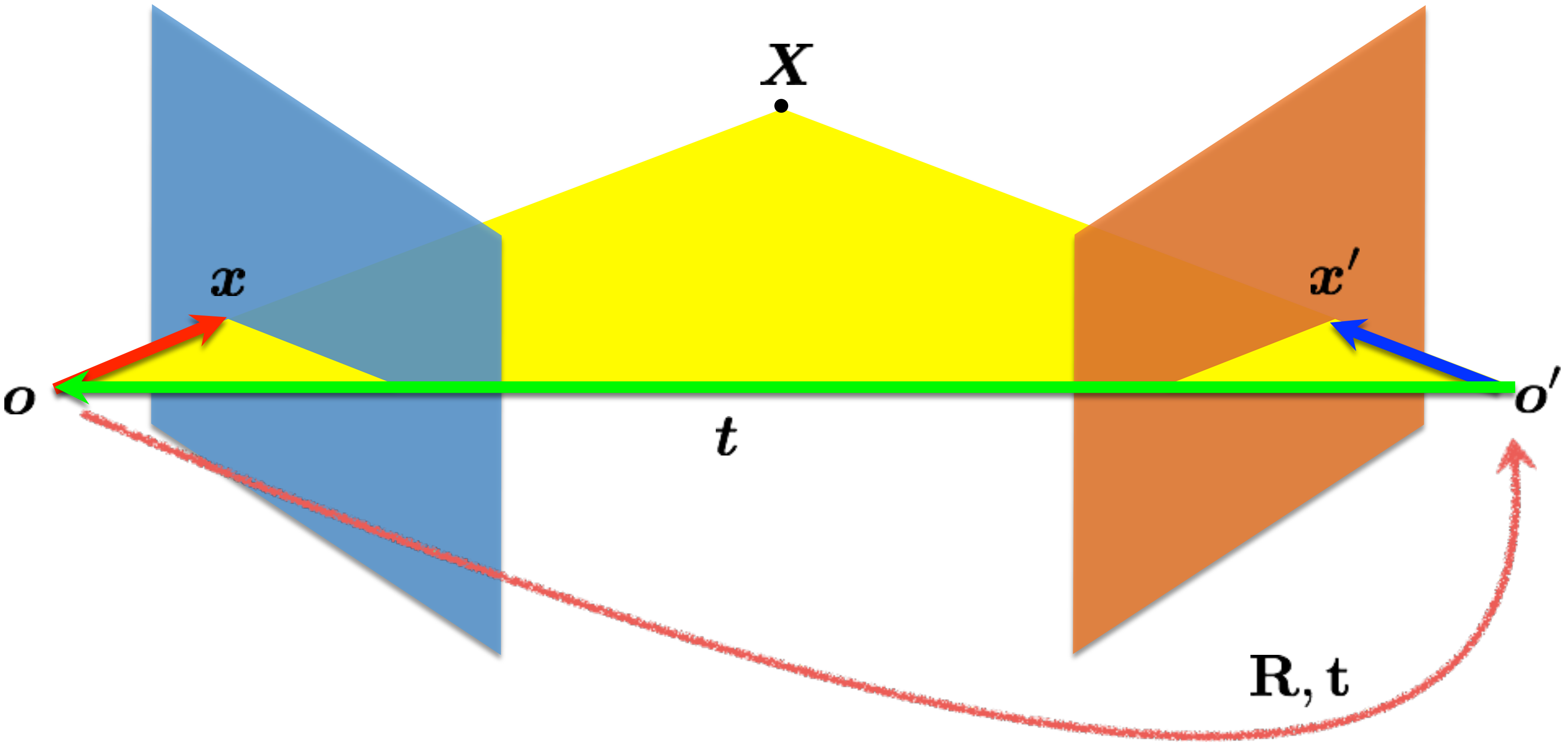
$$\boldsymbol{x}'^{\top}\mathbf{E}\boldsymbol{x} = \ ?$$



Where does the essential matrix come from?

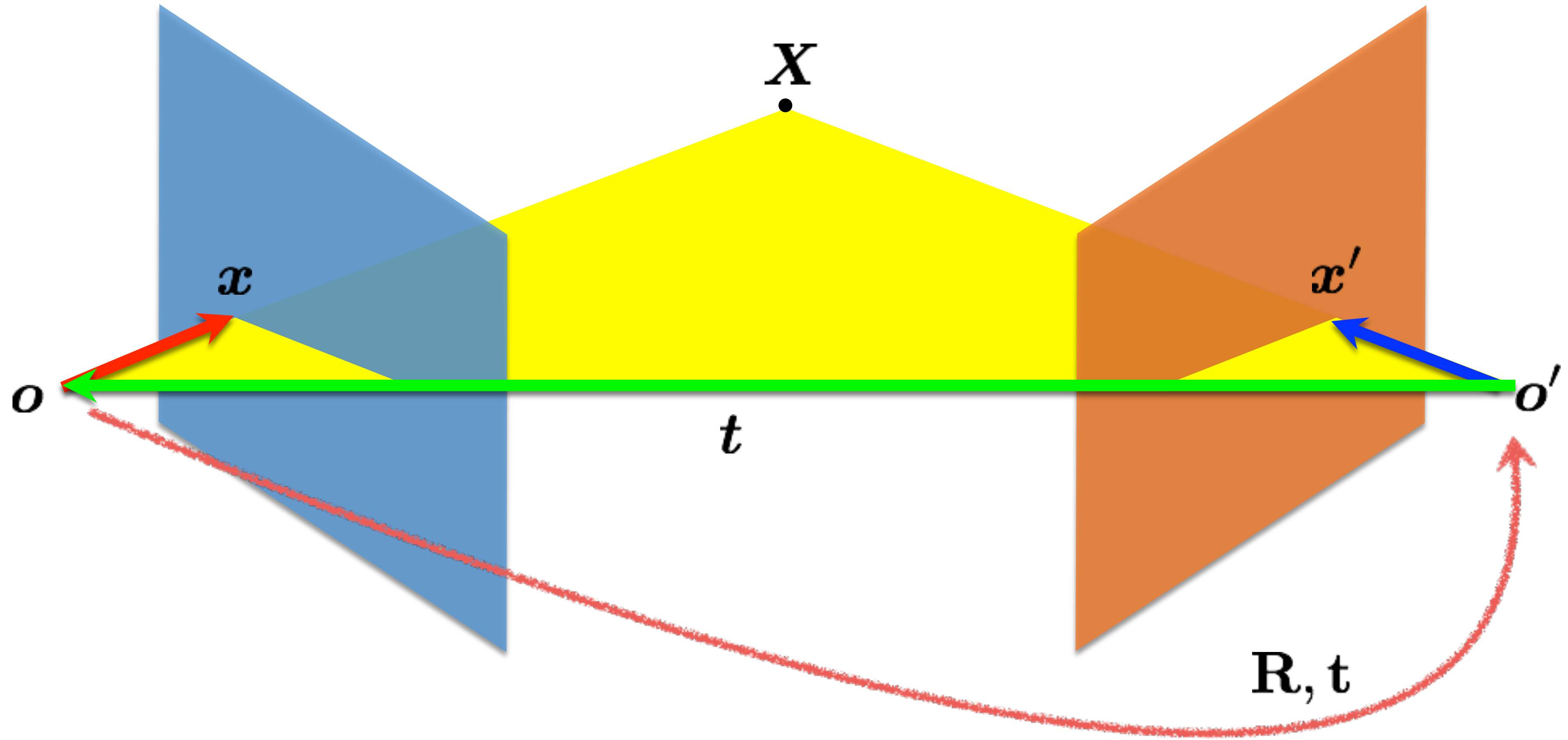


$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$



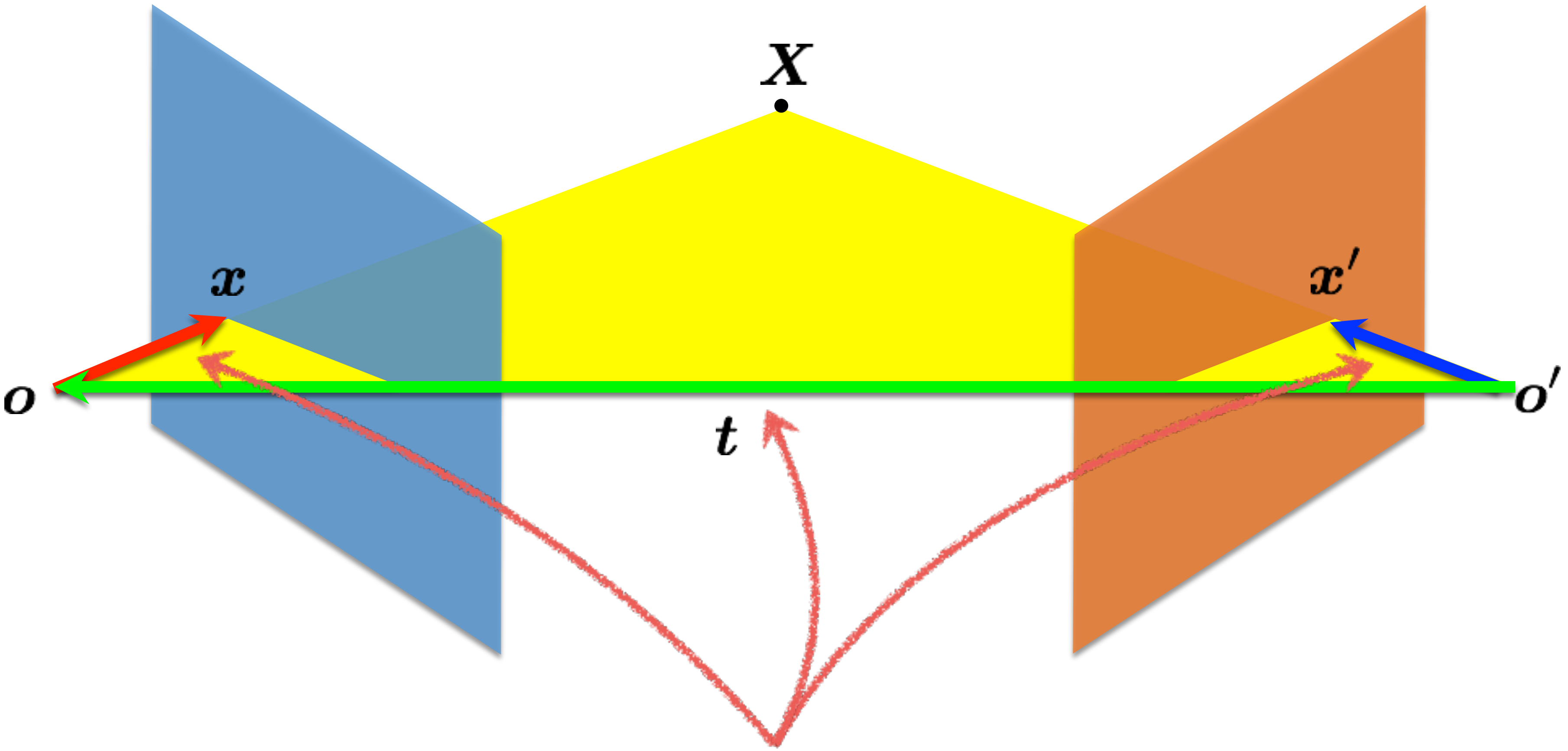
$$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$$

Does this look familiar?



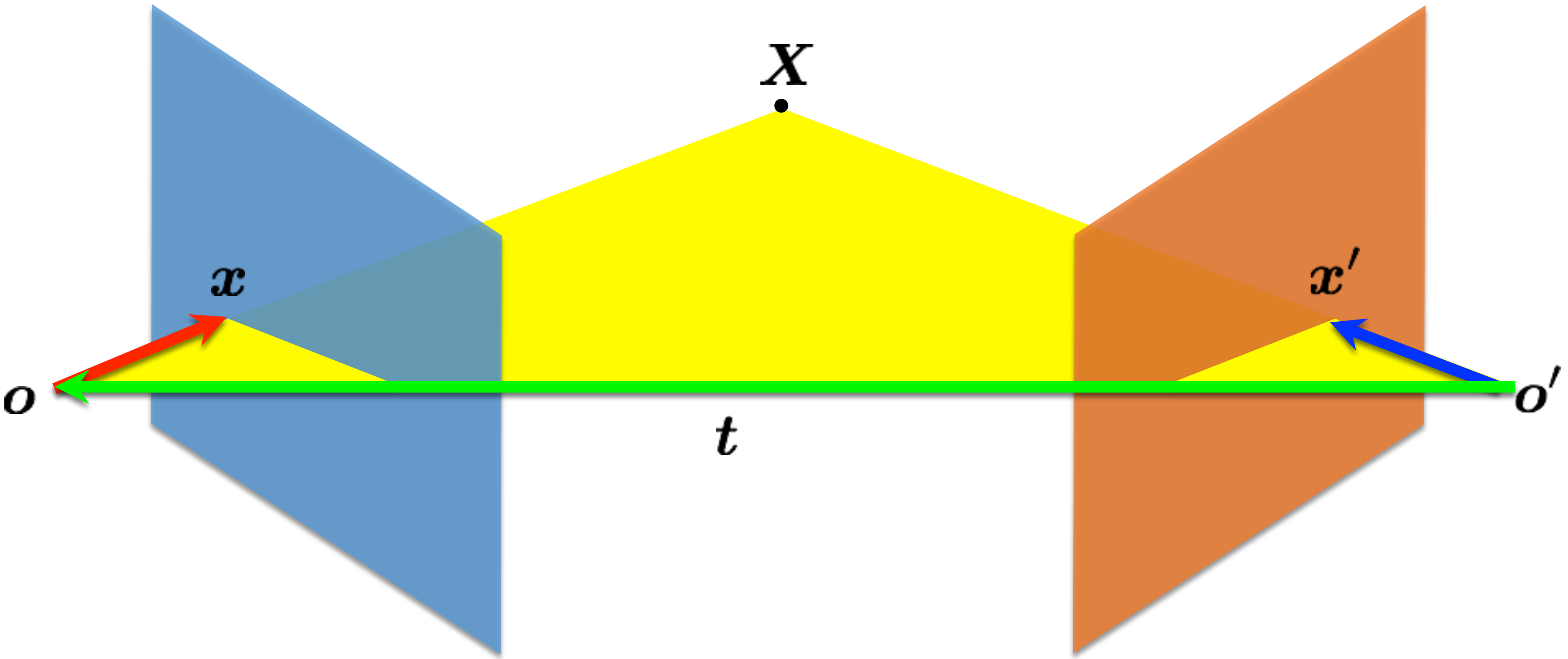
$$x' = \mathbf{R}(x - t)$$

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



These three vectors are coplanar

$$\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{x}'$$

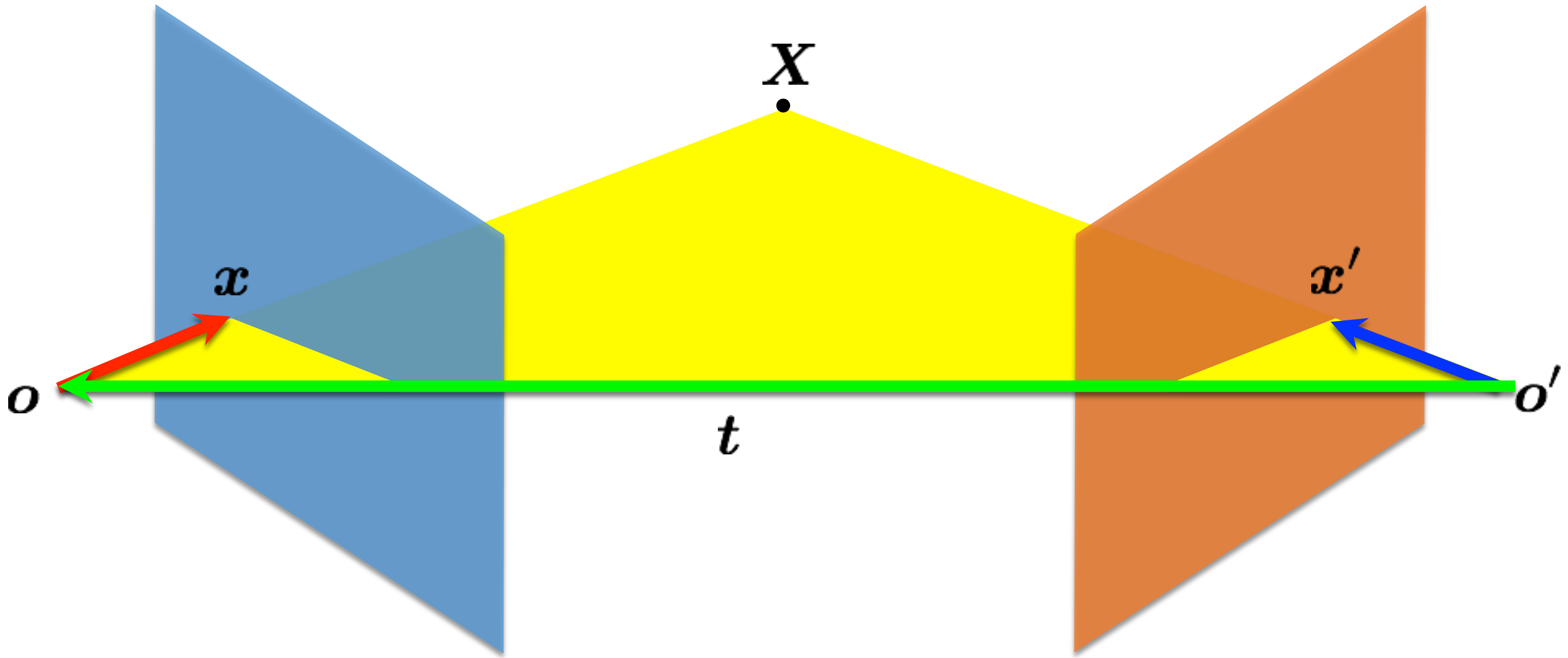


If these three vectors are coplanar $\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{x}'$ then

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

dot product of orthogonal vectors

cross-product: vector orthogonal to plane



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

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

dot product of orthogonal vectors

cross-product: vector orthogonal to plane

Putting it Together

rigid motion

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

coplanarity

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

The outer product (w/ vector) could be re-written as inner product (w/ matrix)!

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

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

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

Rotation matrix is **orthonormal** (transpose = inverse!)

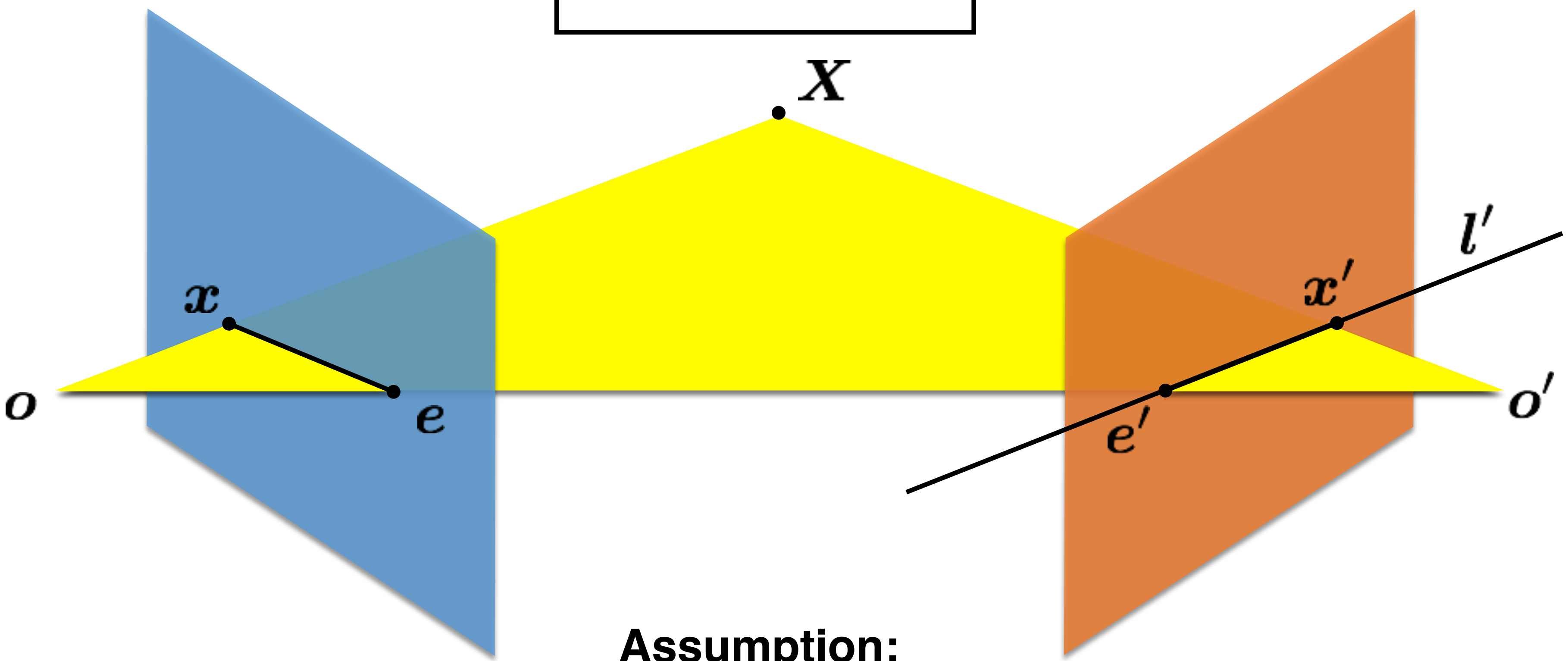
Associativity (Sanity check: dimension?)

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

Essential Matrix
[Longuet-Higgins 1981]

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

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



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

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

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

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

$$\underbrace{\hat{\boldsymbol{x}}}_{\text{camera point}} = \mathbf{K}^{-1}\underbrace{\boldsymbol{x}}_{\text{image point}}$$

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

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

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

$$\underset{\text{camera point}}{\hat{\boldsymbol{x}}} = \mathbf{K}^{-1}\underset{\text{image point}}{\boldsymbol{x}}$$

Writing out the epipolar constraint in terms of image coordinates

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

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

Same equation works in image coordinates!

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

it maps pixels to epipolar lines

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

Now recall: why Tsai's algorithm wants to decompose ***P*** *:-)*

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

How would you solve for F?

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

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?*

Assume you have *M* matched *image* points (via Harris, SIFT...)

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

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?*

S V D

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

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

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

How many equation do you get from one correspondence?

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

ONE correspondence gives you ONE equation

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

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

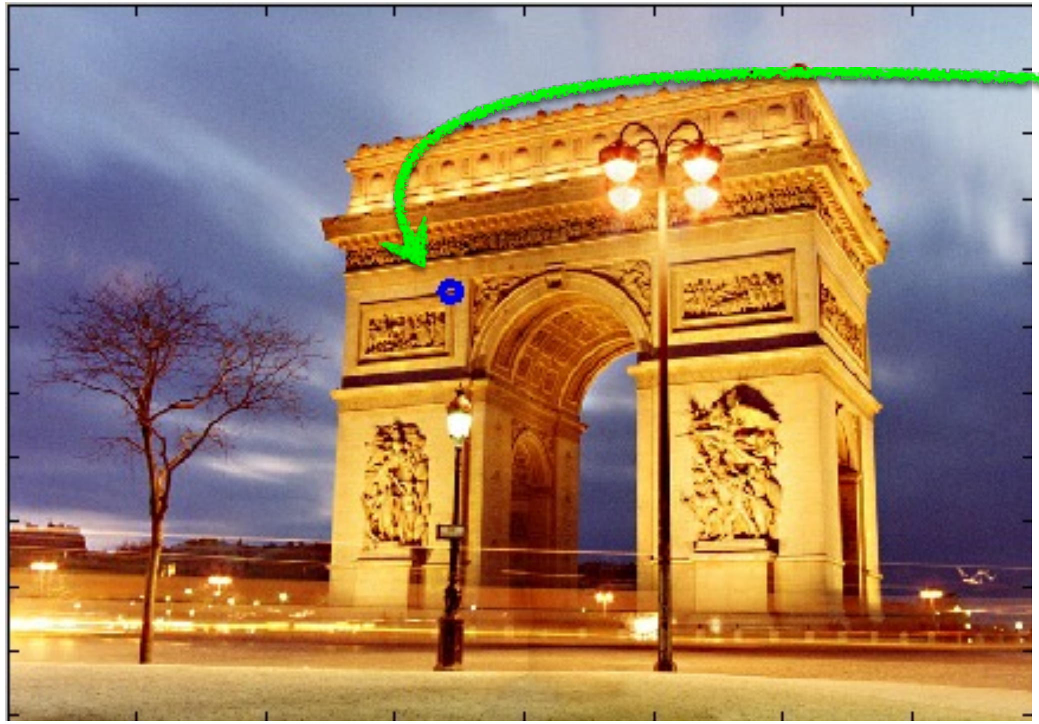
Set up a homogeneous linear system with 9 unknowns

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

S V D !

Example: epipolar lines

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



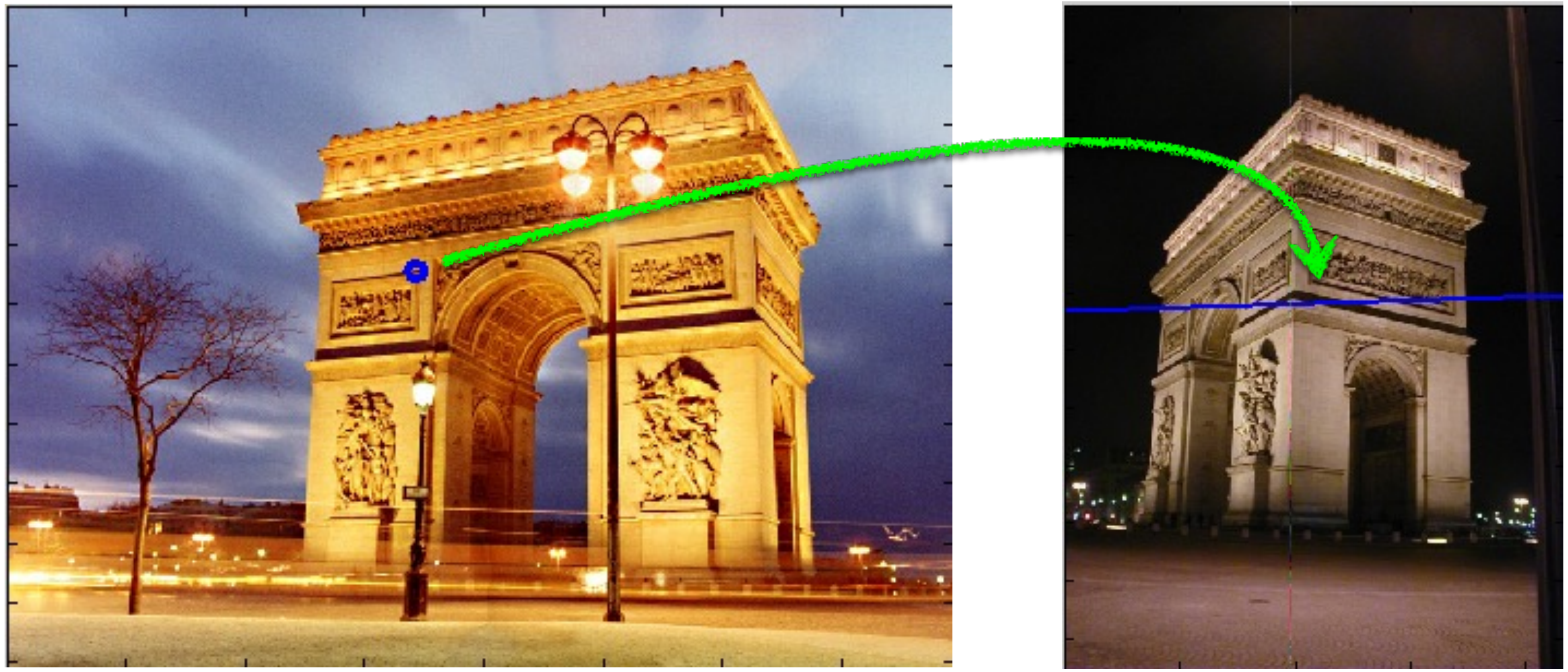
$$\boldsymbol{x} = \begin{bmatrix} 343.53 \\ 221.70 \\ 1.0 \end{bmatrix}$$

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

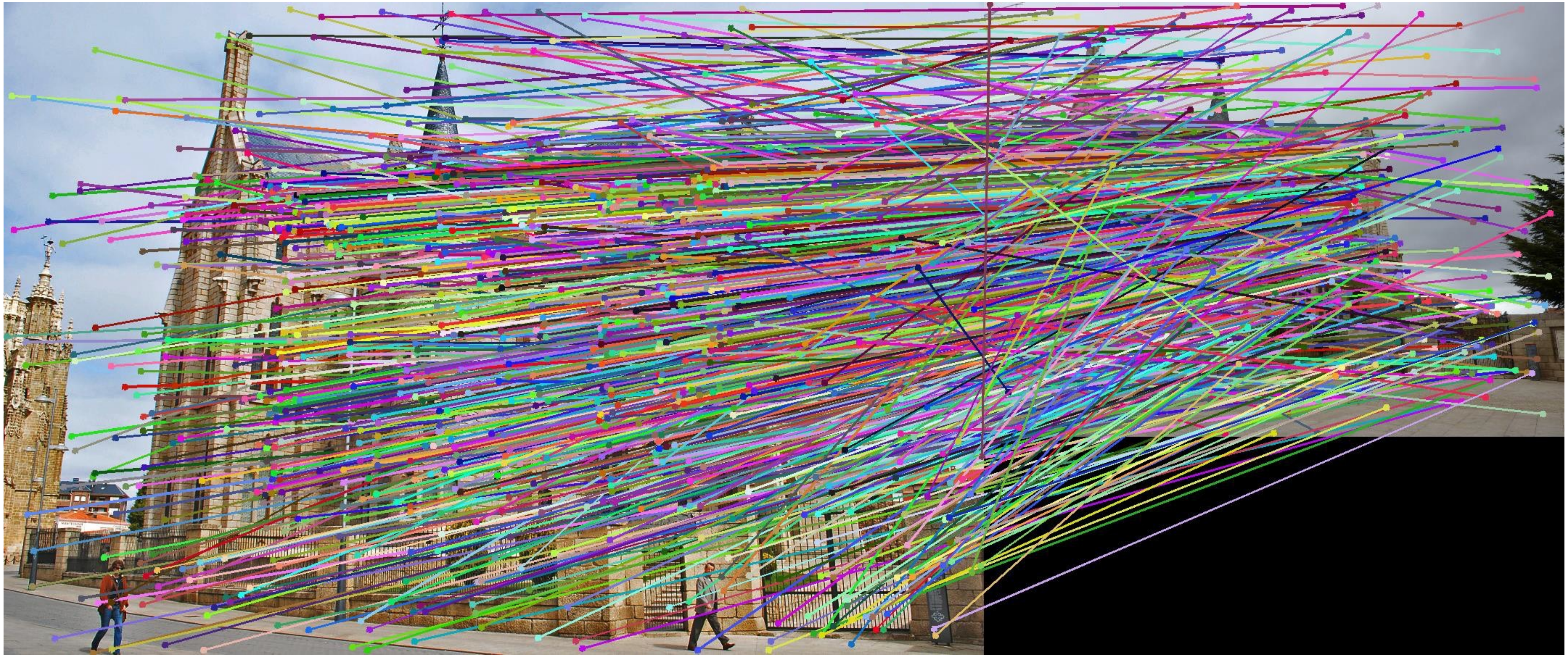
$$= \begin{bmatrix} 0.0295 \\ 0.9996 \\ -265.1531 \end{bmatrix}$$

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

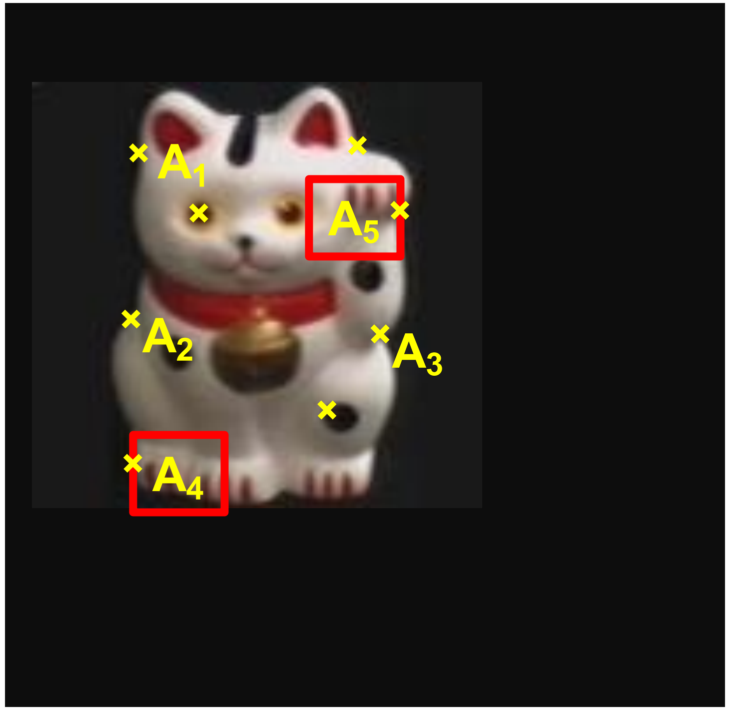
$$= \begin{bmatrix} 0.0295 \\ 0.9996 \\ -265.1531 \end{bmatrix}$$



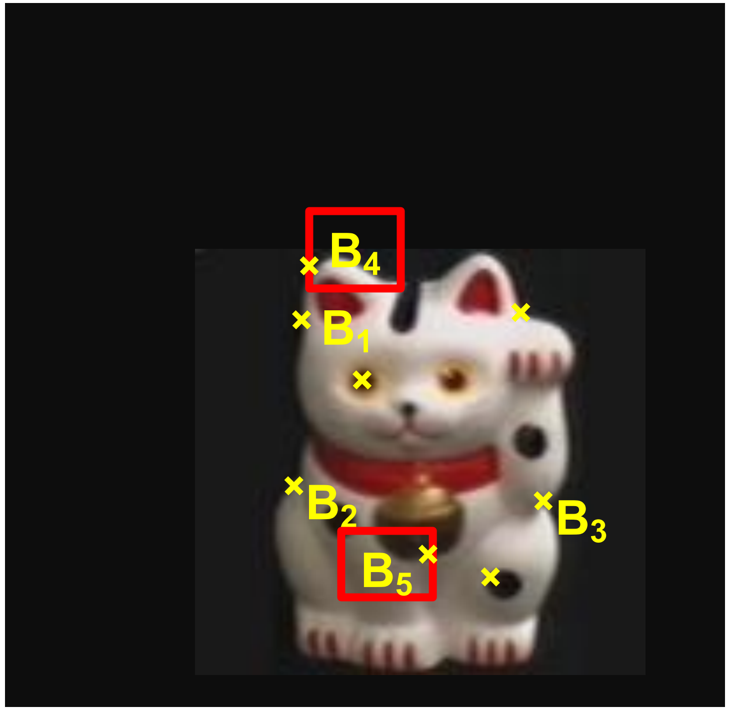
8-point is sufficient in theory to estimate E/F… *but least square often not robust enough*



Example: solving for translation?



(t_x, t_y)



Problem: outliers A_4-B_4 and A_5-B_5 which *incorrectly* correspond

RANSAC solution (RANdom SAmple Consensus) :
Fischler & Bolles in '81.

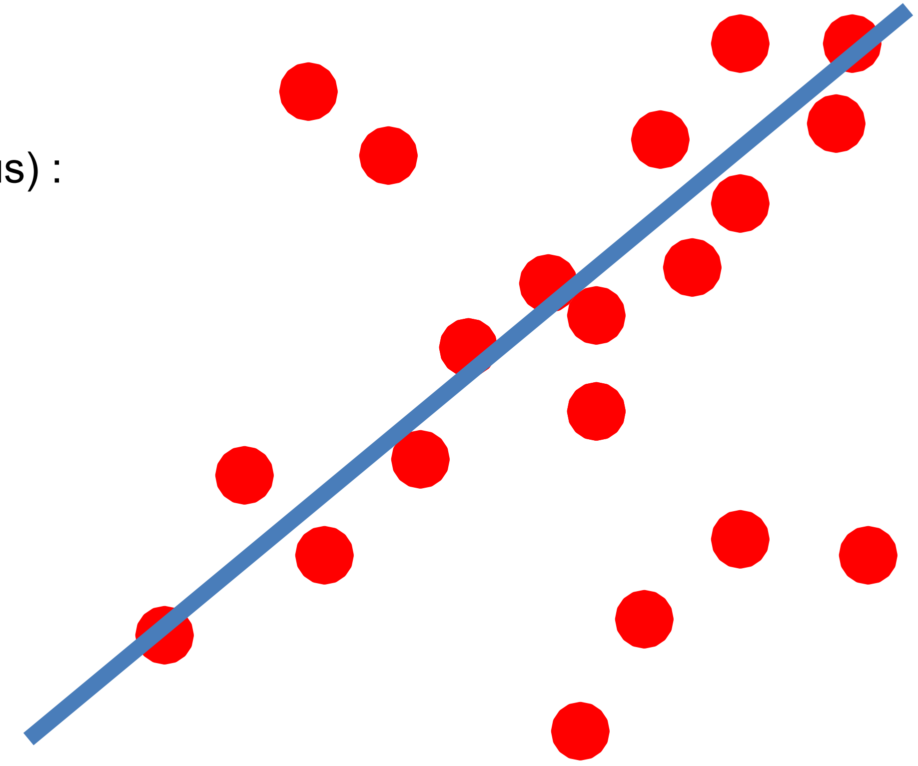
1. Sample a set of matching points (1 pair)
2. Solve for transformation parameters
3. Score parameters with number of inliers
4. Repeat steps 1-3 N times

$$\begin{bmatrix} x_i^B \\ y_i^B \end{bmatrix} = \begin{bmatrix} x_i^A \\ y_i^A \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.

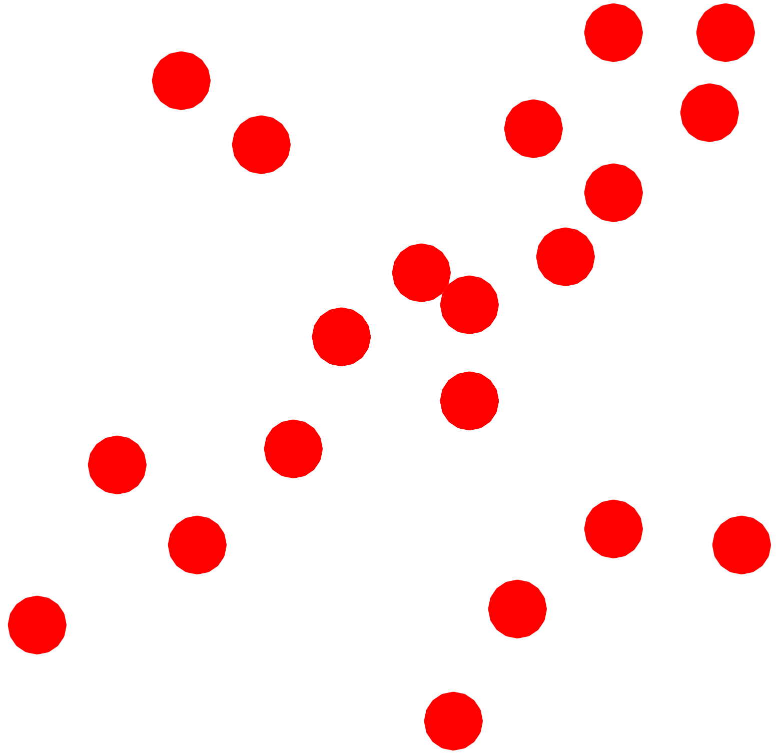


This data is noisy, but we expect a good fit to a known model.

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



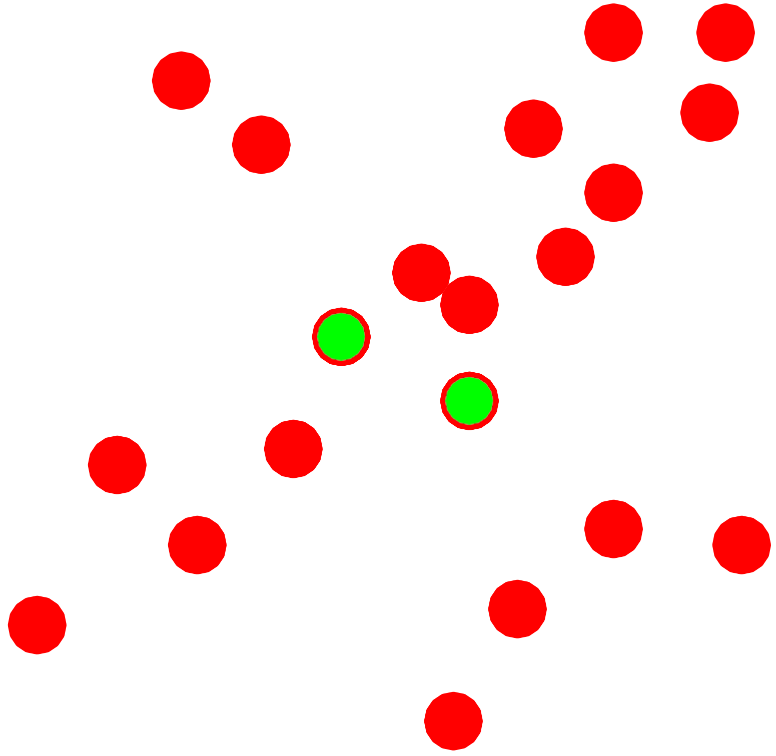
Algorithm:

1. **Sample** (randomly) the number of points *s* 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

RANSAC

Line fitting example



Algorithm:

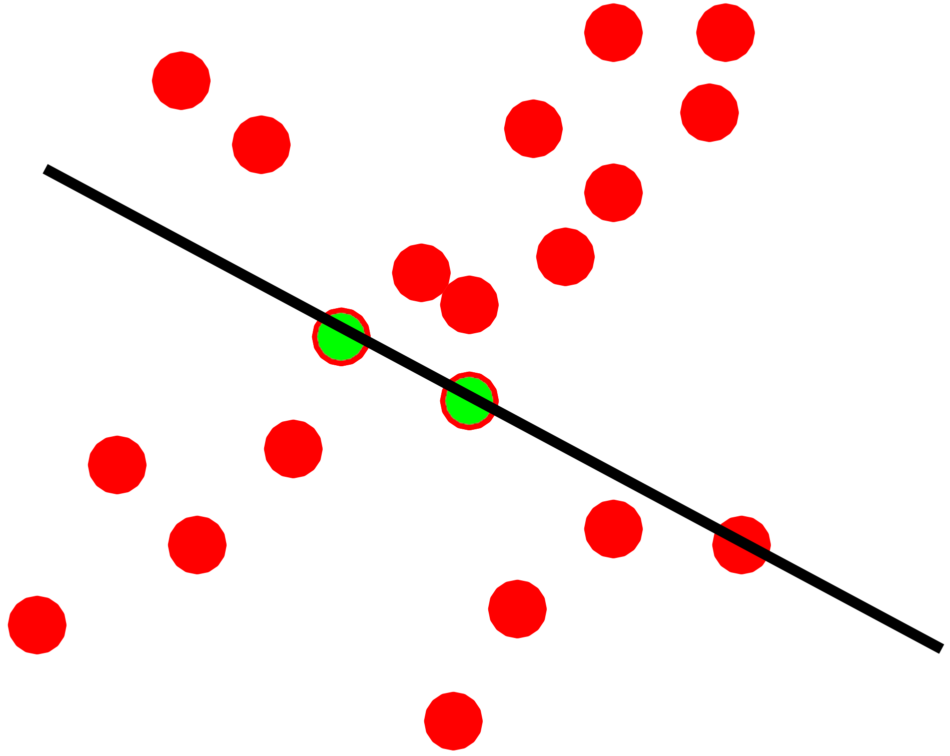
1. **Sample** (randomly) the number of points required to fit the model (s=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

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

Illustration by Savarese

RANSAC

Line fitting example



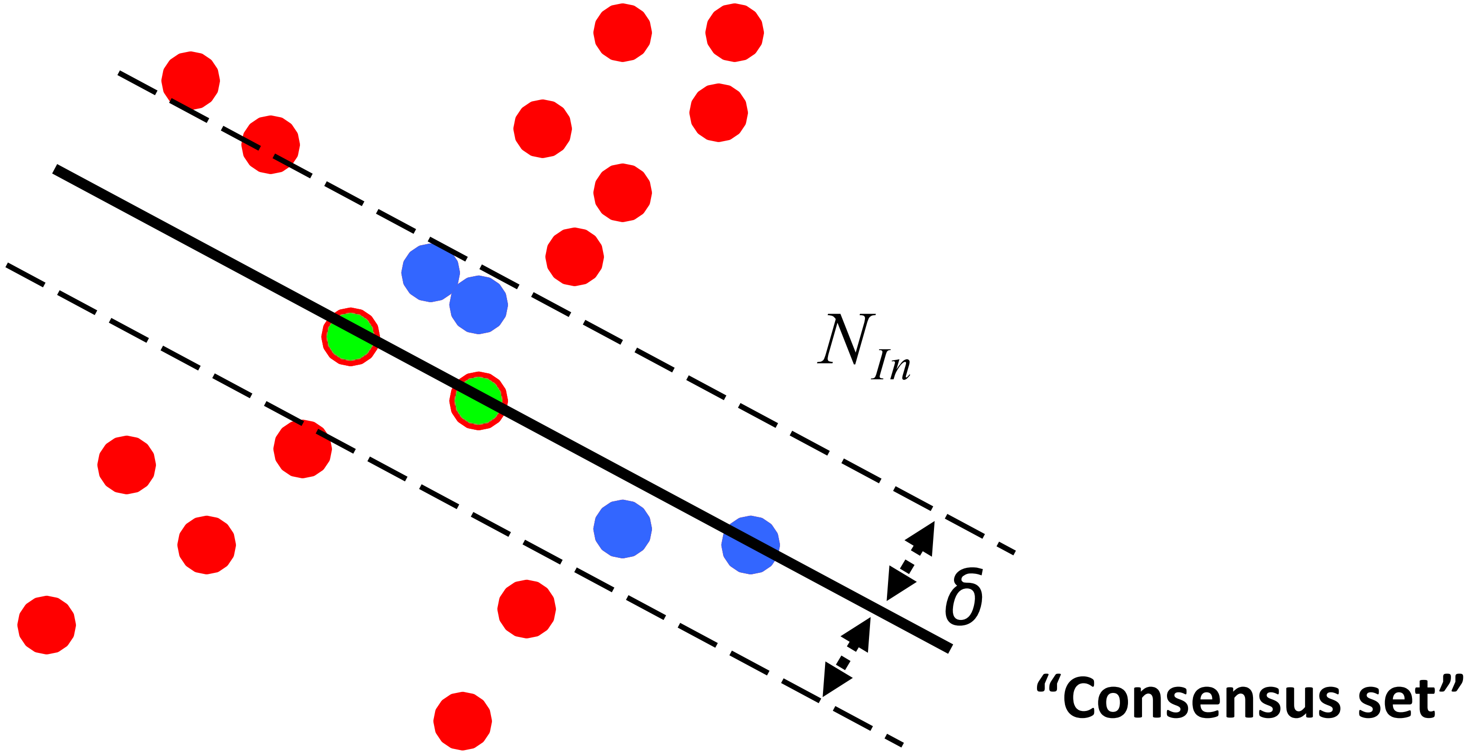
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (*s*=2)
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

RANSAC

Line fitting example

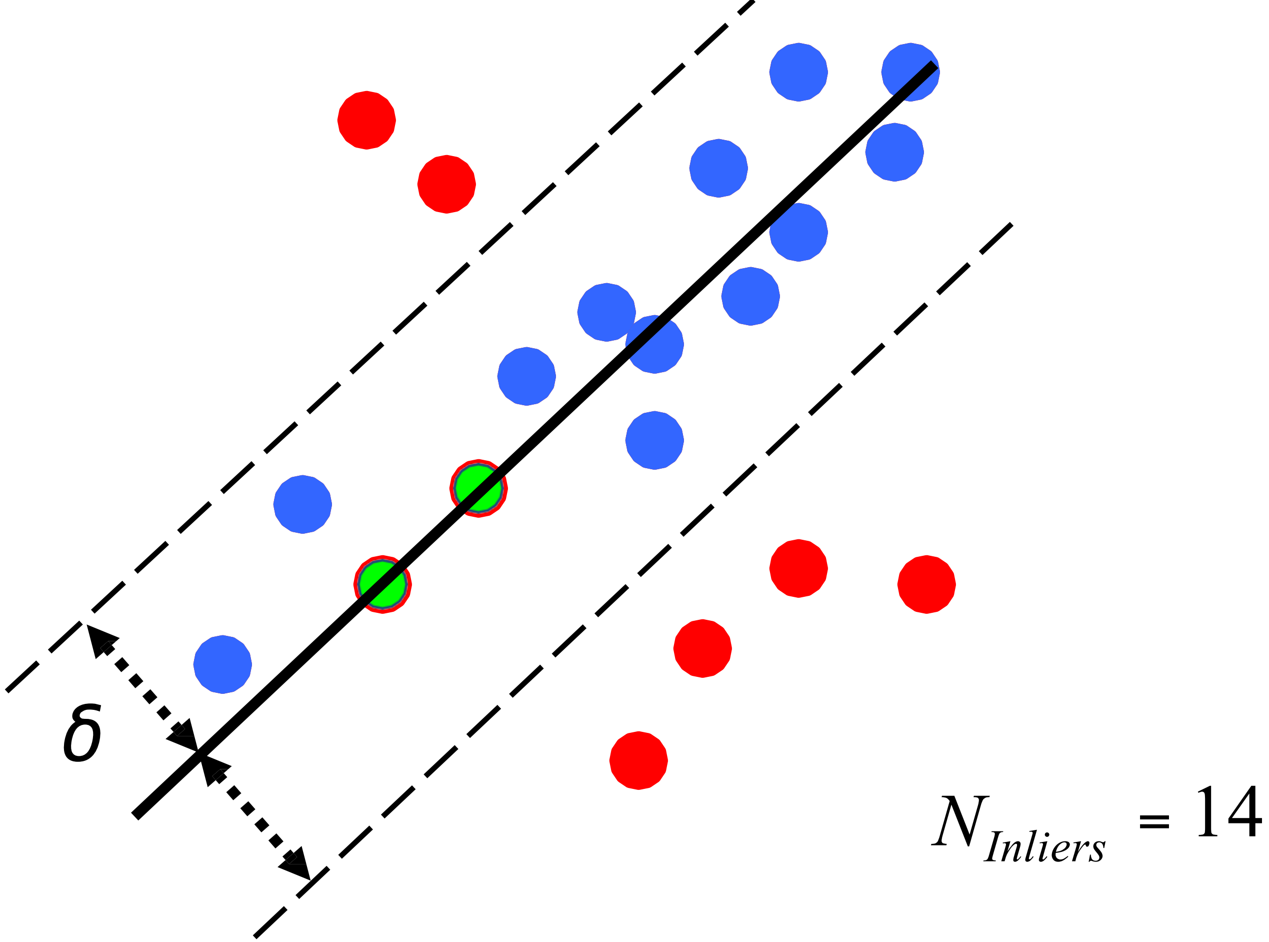


Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (s=2)
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

RANSAC

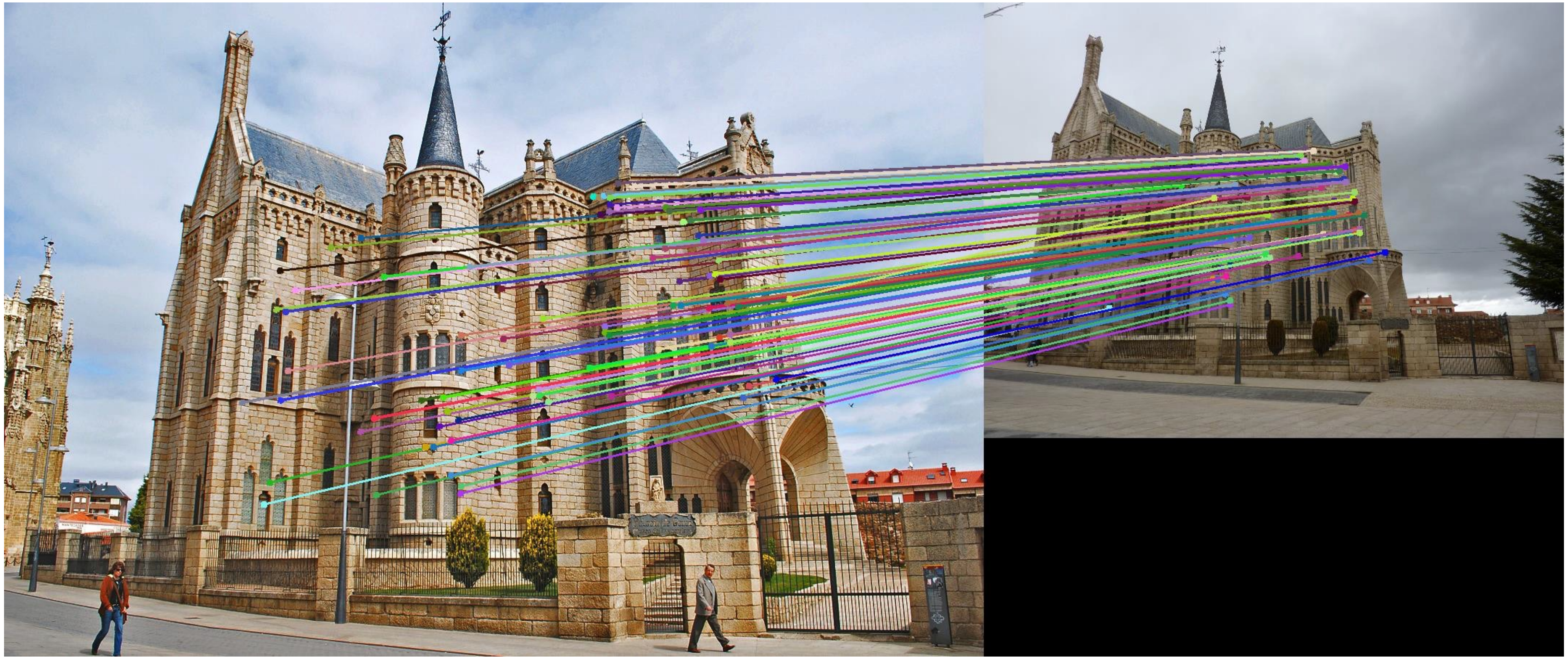


Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (*s*=2)
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

Keep only the matches at are “inliers” with respect to the “best” fundamental matrix (RANSAC)



RANSAC Summary

Good

- Robust to outliers, simple & assumption-free idea
- Applicable for large number of objective function parameters
- Optimization parameters are relatively easier to choose

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not good for getting multiple fits

Most common applications

- Estimating fundamental matrix (relating two views)
- Computing a homography (e.g., image stitching)

Recap: epipolar geometry & camera calibration

- If we know the calibration matrices of the two cameras, we can estimate the essential matrix: $E = K^T F K'$

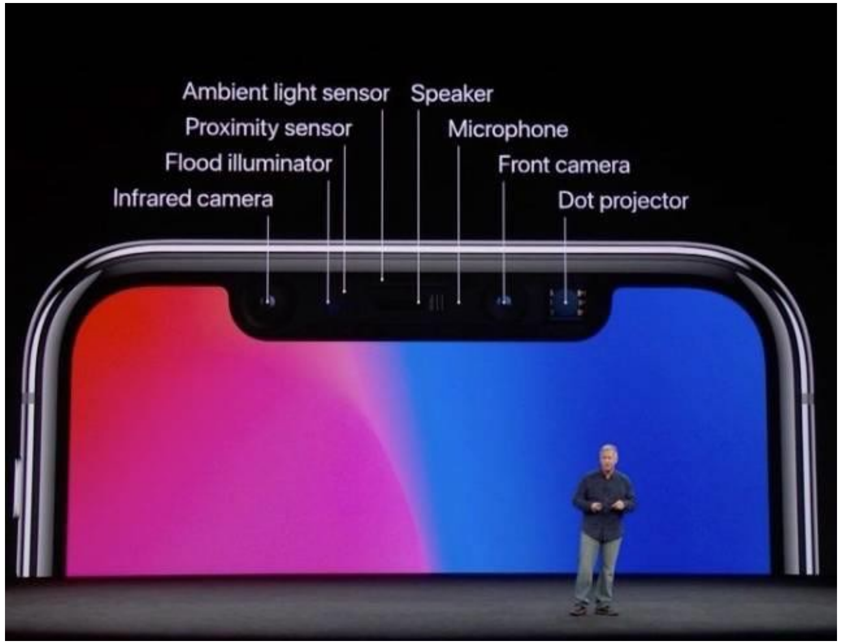
- The essential matrix gives us the relative rotation and translation between the cameras, or their extrinsic parameters.

- Fundamental matrix lets us compute relationship up to scale for cameras with unknown intrinsic calibrations.

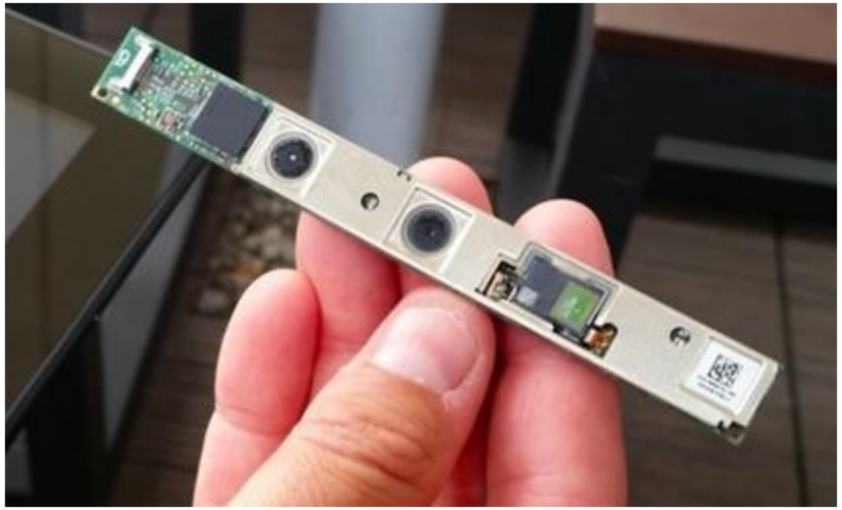
- Estimating the fundamental matrix is a kind of "weak calibration"

Depth and Camera

Microsoft Kinect v1



iPhone X



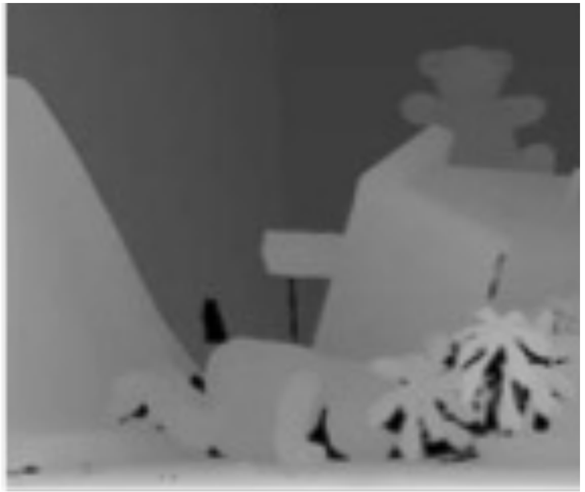
Intel laptop depth camera

What's different between these two images?

Objects that are close move more or less?

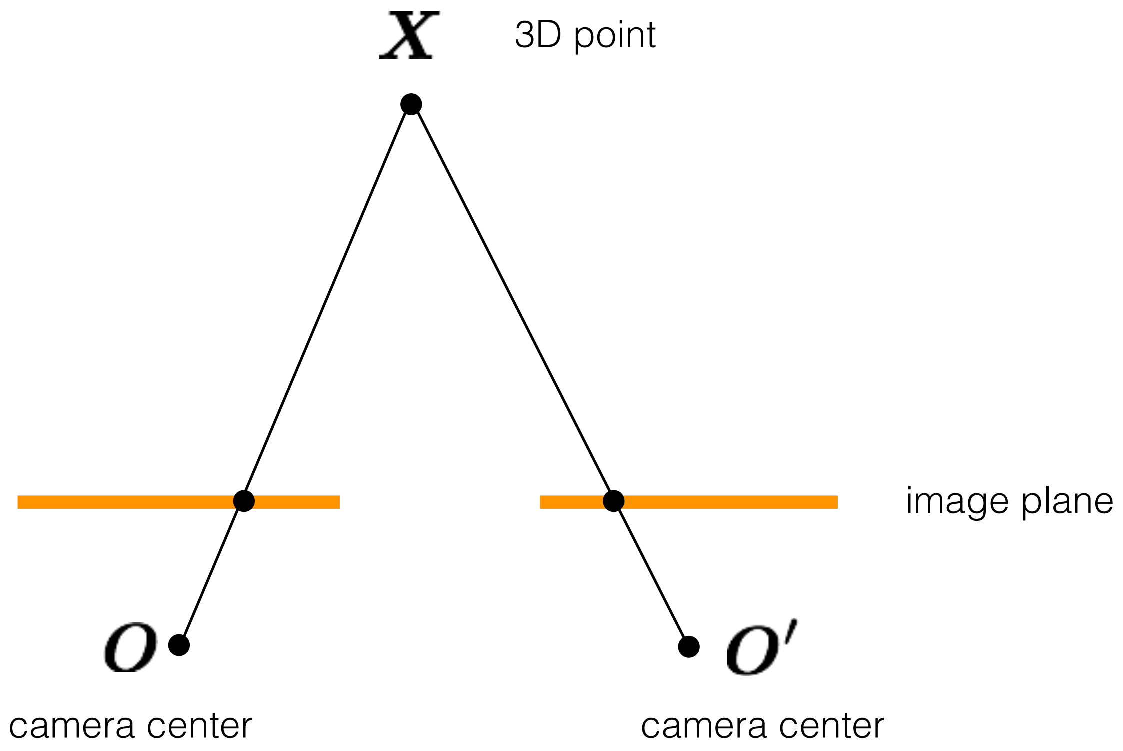
The amount of horizontal movement is inversely proportional to …

The amount of horizontal movement is
inversely proportional to …

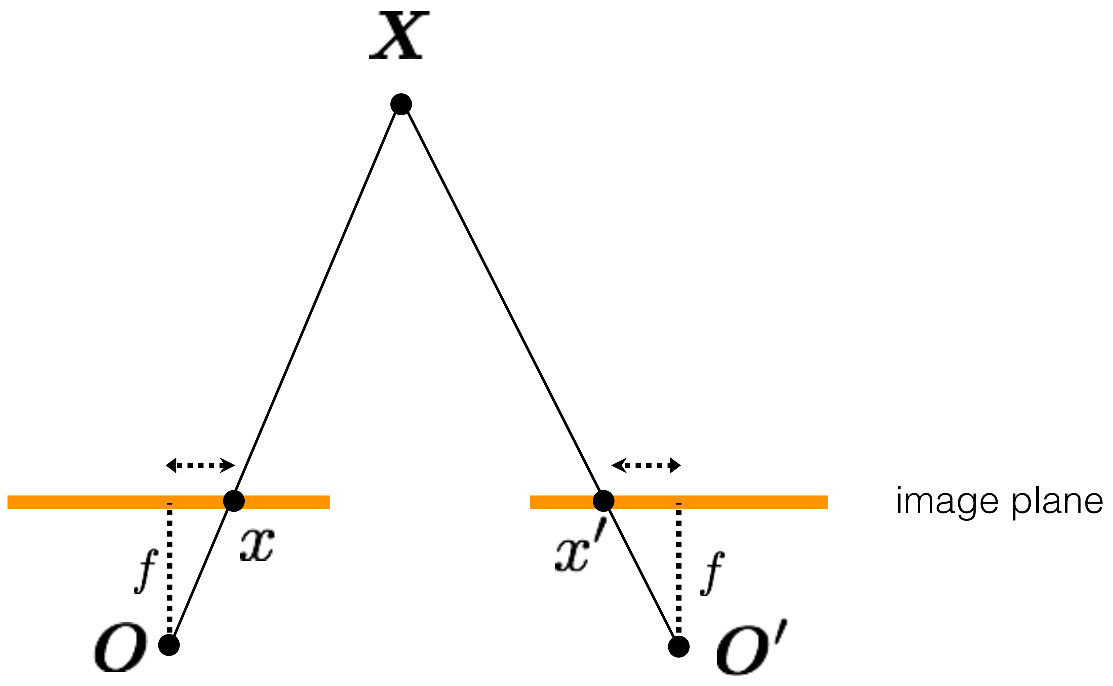
  

… the distance from the camera.

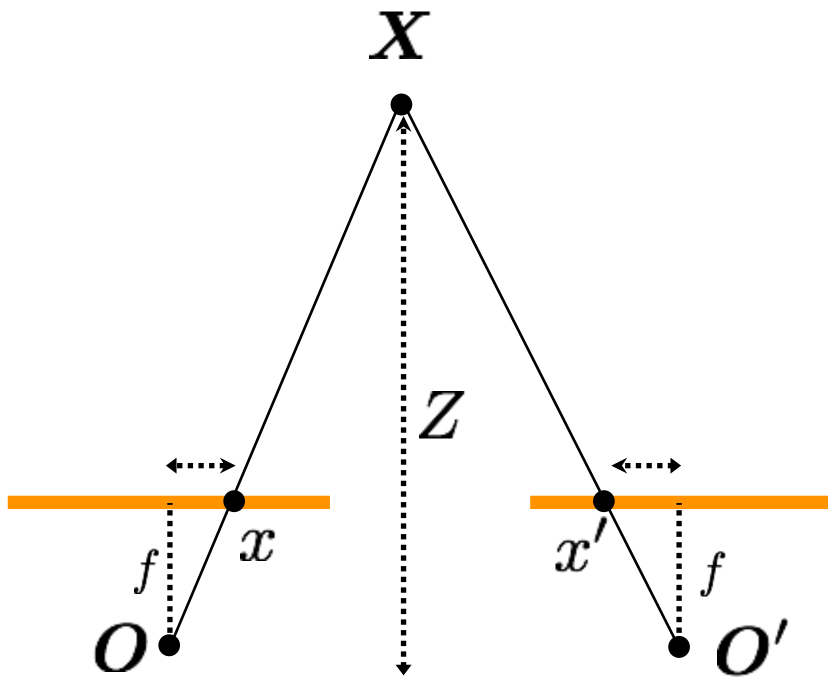
More formally…

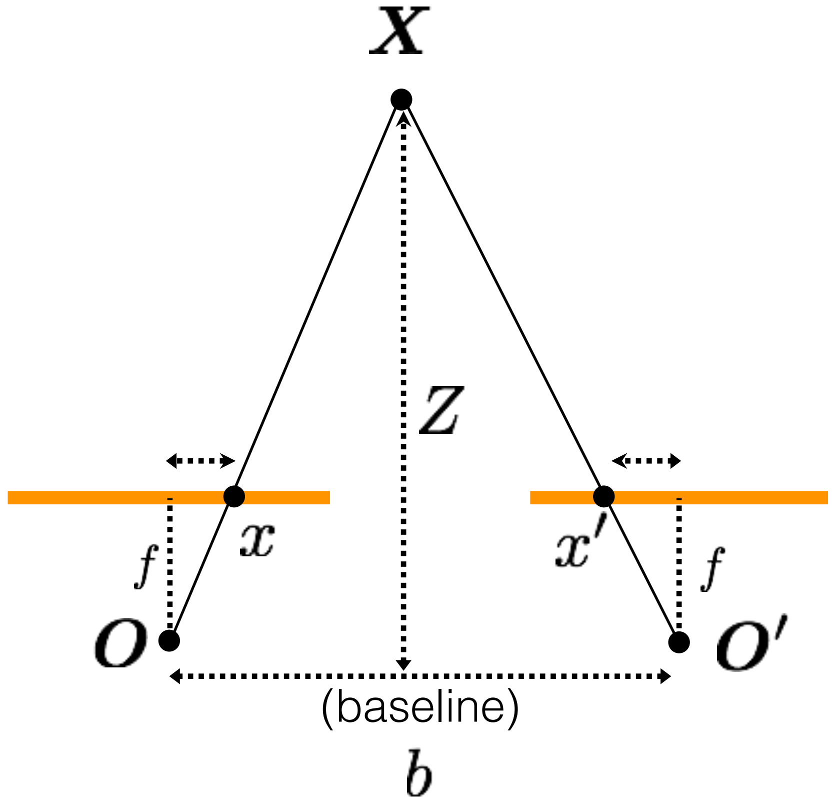


X
3D point
image plane
O
O'
camera center
camera center



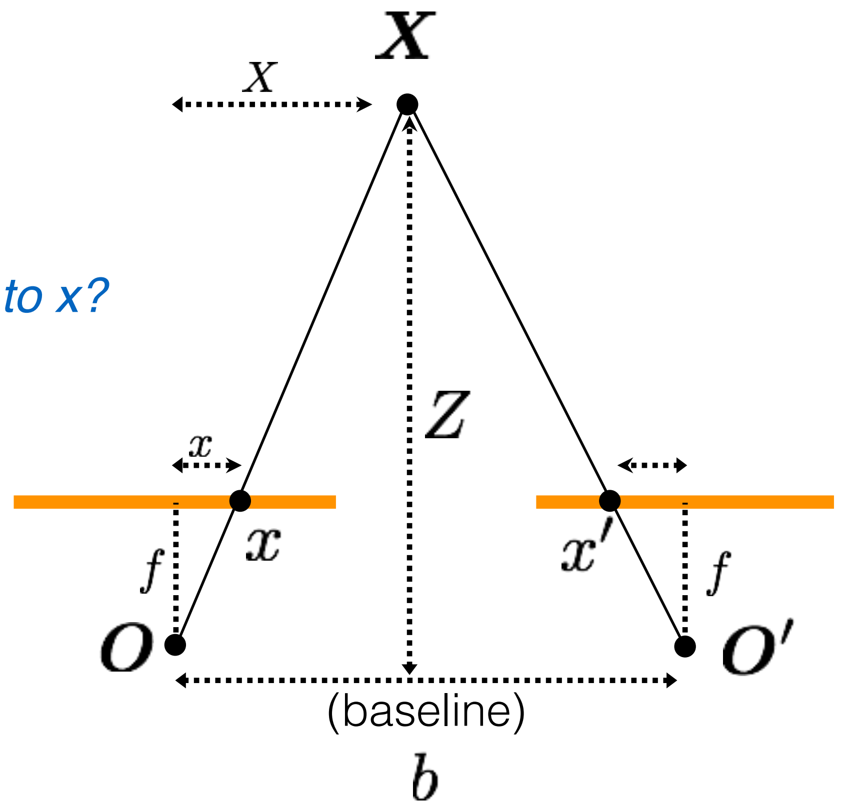
X
x
x'
f
f
O
O'
image plane


X
Z
x
x'
f
f
O
O'

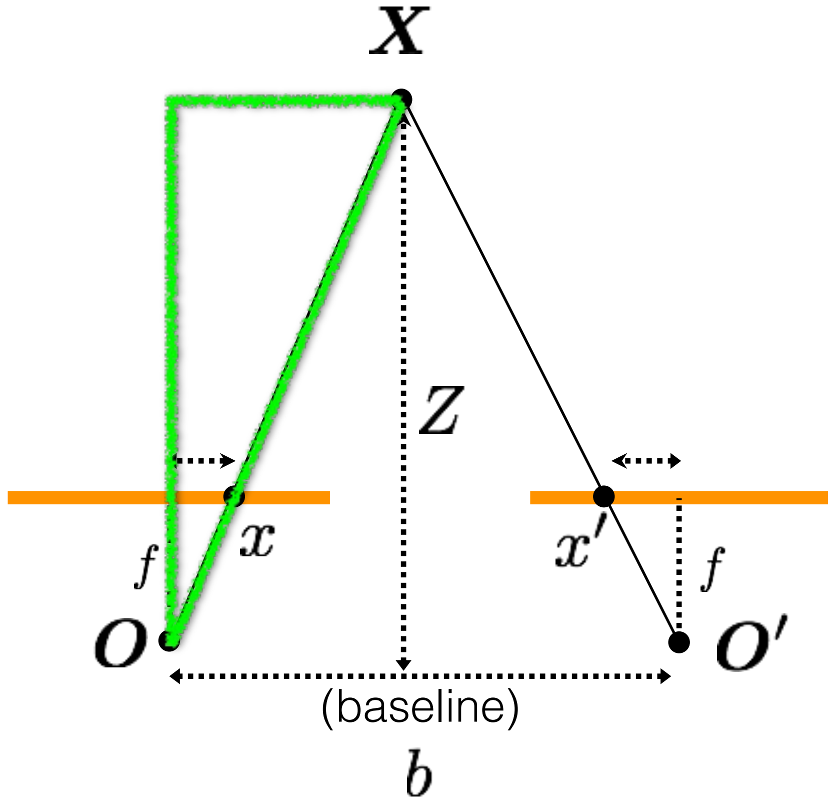


X
Z
x
x'
f
f
O
O'
(baseline)
b

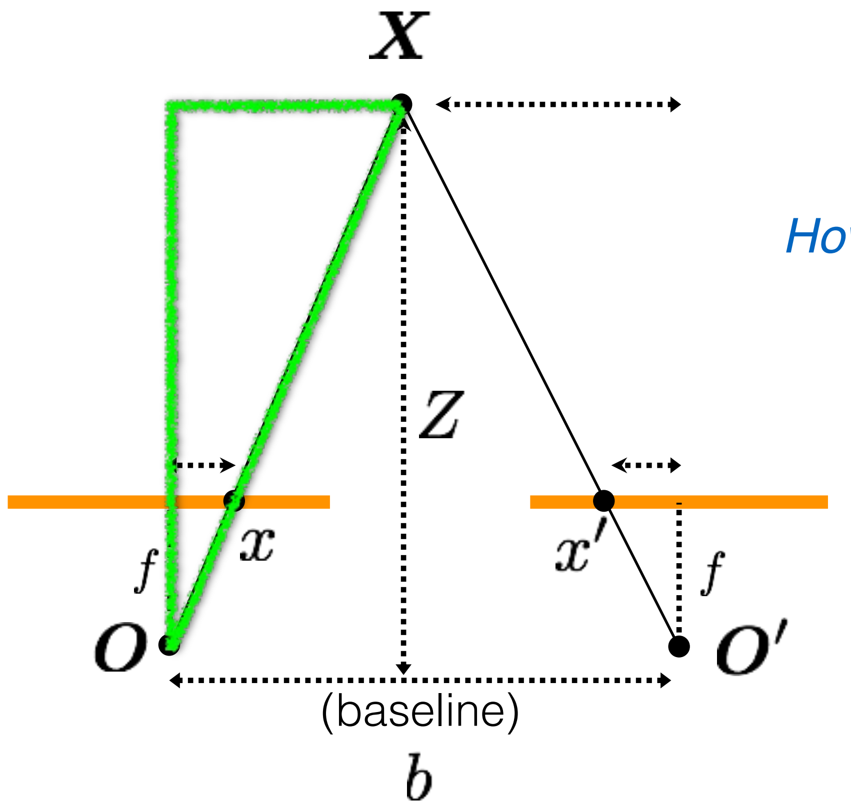
How is X related to x?



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

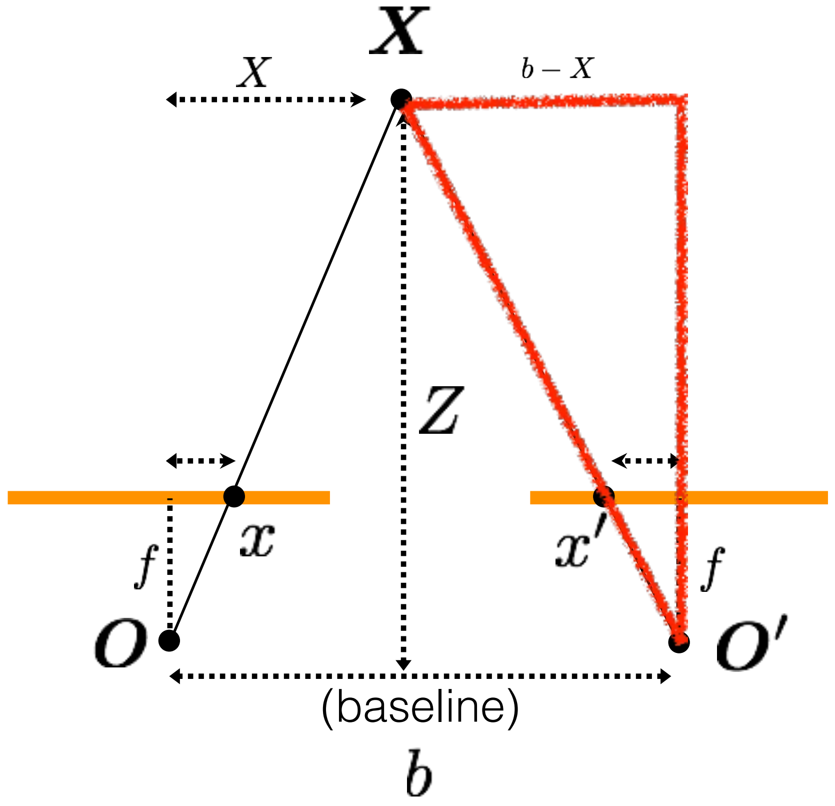


$$\frac{X}{Z} = \frac{x}{f}$$



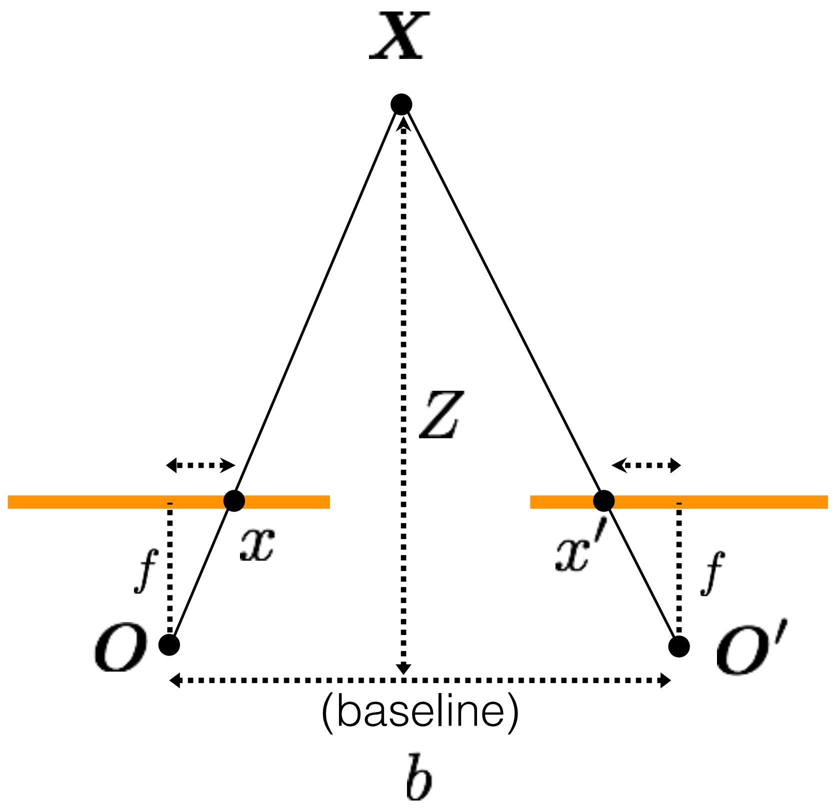
How is X related to x'?

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



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

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



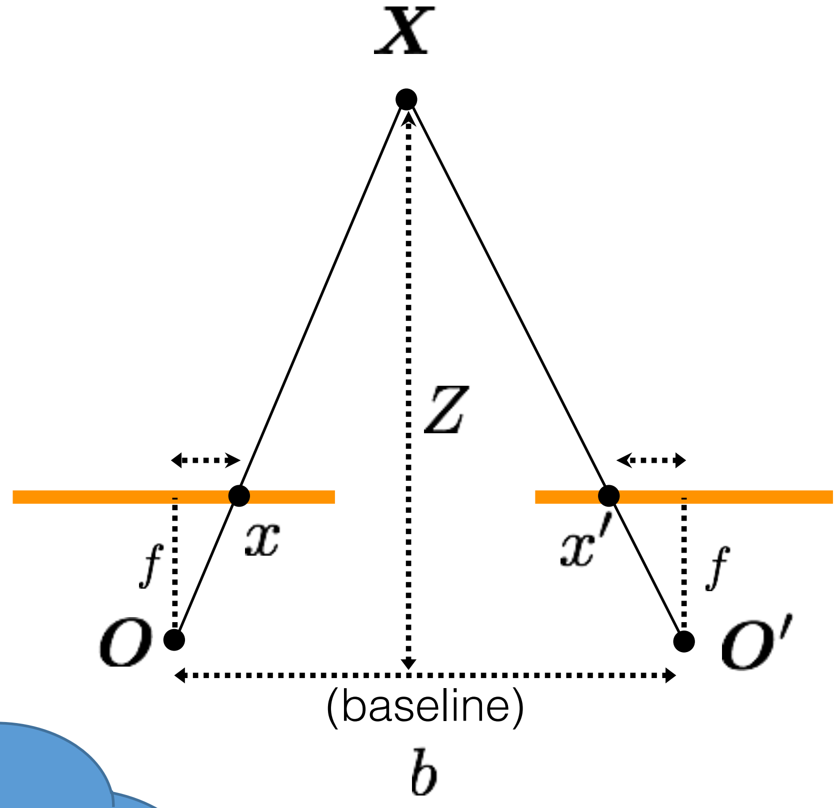
$$\frac{b - X}{Z} = \frac{x'}{f}$$

Disparity

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

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

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



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

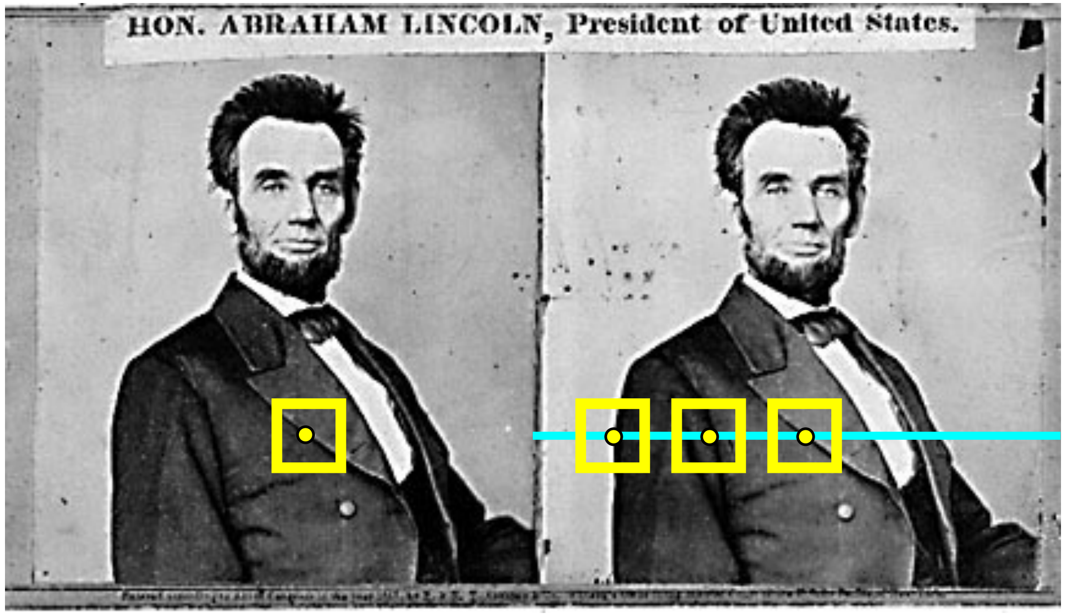
So, if I know *x* and *x'*, I can compute depth!!

Disparity

$$d = x - x'$$
$$= \frac{bf}{Z}$$

inversely proportional
to depth

Depth Estimation via Stereo Matching



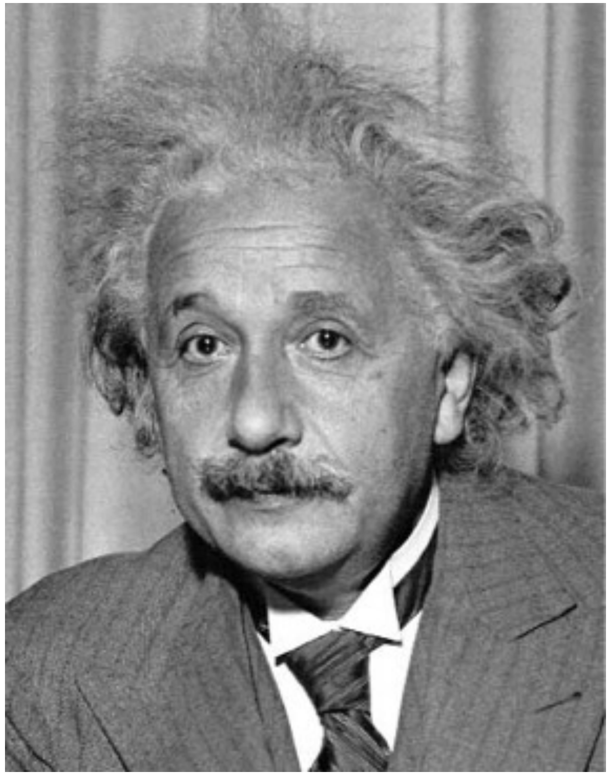
1. Rectify images
   (make epipolar lines horizontal)
2. For each $\boxed{\text{pixel}}$
   a. Find epipolar line
   b. Scan line for best match
   c. Compute depth from disparity

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

How would you do this?
Template Matching

Find this template

How do we detect the template  in the following image?



What will the output look like?

filter  template mean

output

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

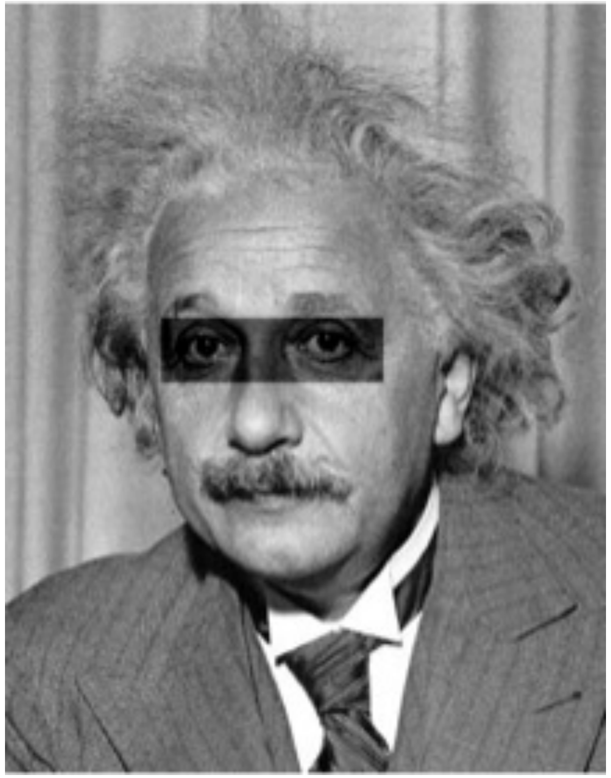
image

local patch mean

Normalized cross-correlation (NCC).

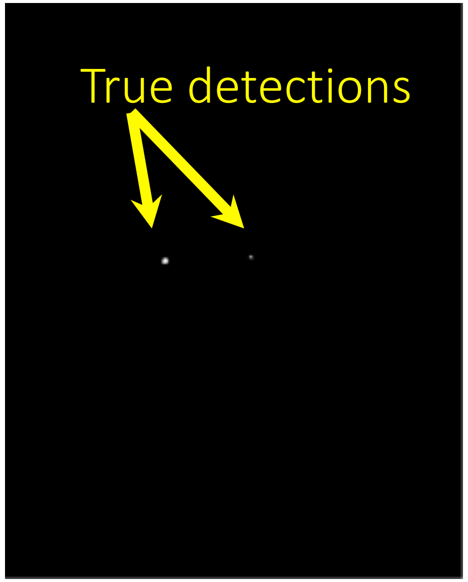
Find this template

How do we detect the template  in the following image?

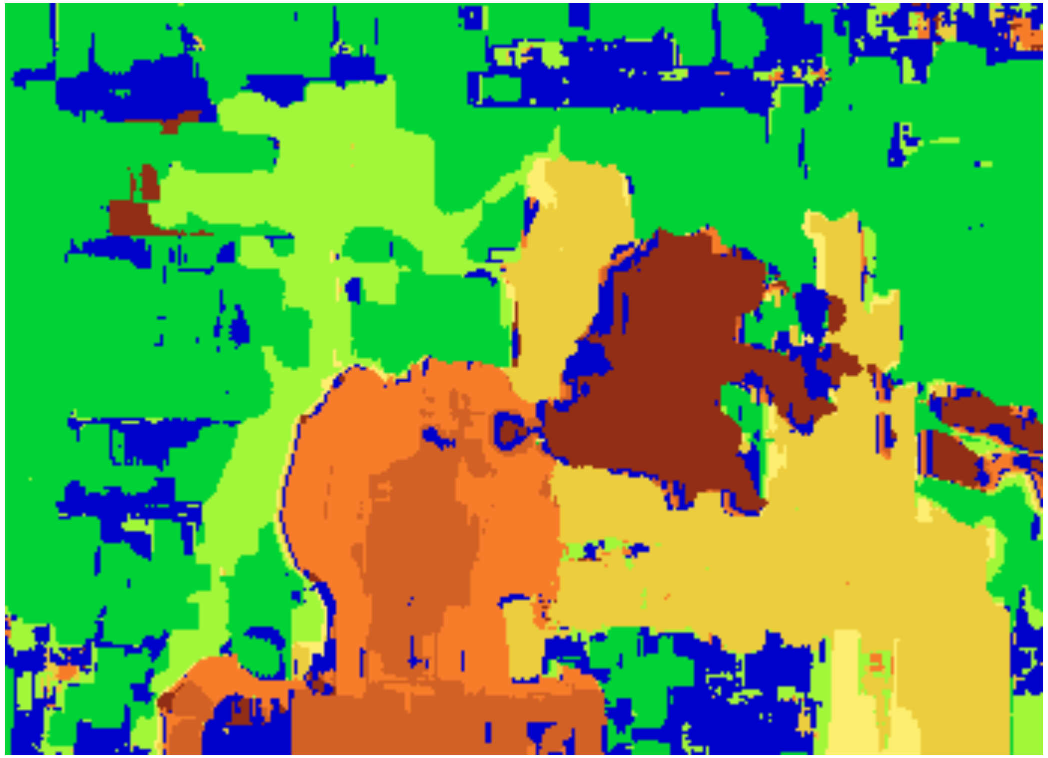


1-output

thresholding



Normalized cross-correlation (NCC).



How can we improve depth estimation?

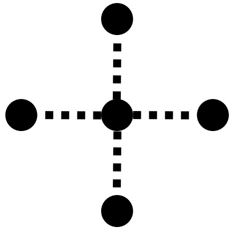
Too many discontinuities.
We expect disparity values to change slowly.

Let's make an assumption:
depth should change smoothly

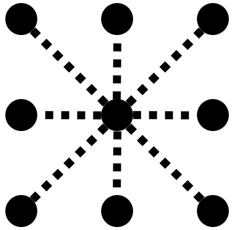
$$E_s(d) = \sum_{(p,q)\in\mathcal{E}} V(d_p, d_q)$$

smoothness term

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



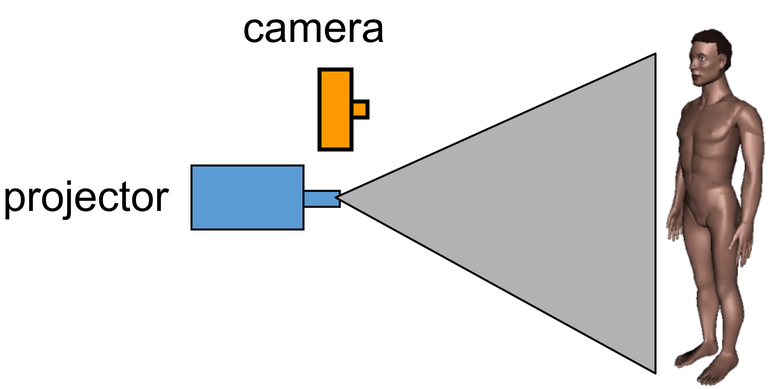
4-connected neighborhood



8-connected neighborhood

Active stereo with structured light

- Project "structured" light patterns onto the object
  - Simplifies the correspondence problem
  - Allows us to use only one camera

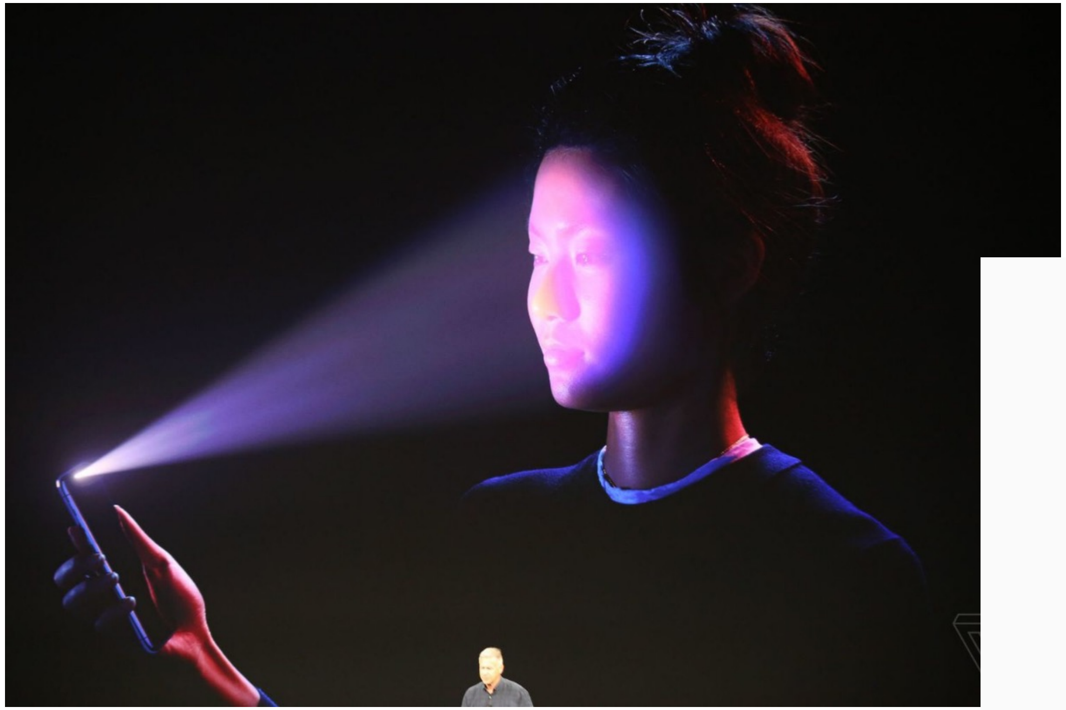


L. Zhang, B. Curless, and S. M. Seitz. [Rapid Shape Acquisition Using Color Structured Light and Multi-pass Dynamic Programming.]() *3DPVT* 2002

Kinect: Structured infrared light

http://bbzippo.wordpress.com/2010/11/28/kinect-in-infrared/

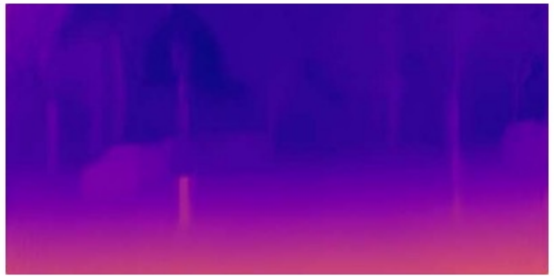
iPhone X



iPhone 12 has lidar

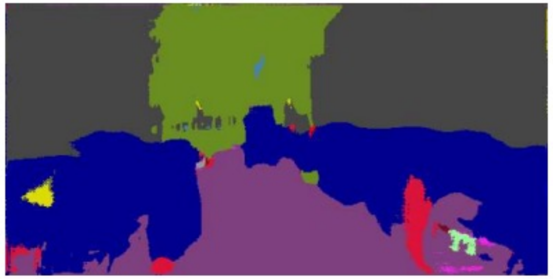
“Semantic” Depth Estimation

(a) Input Image



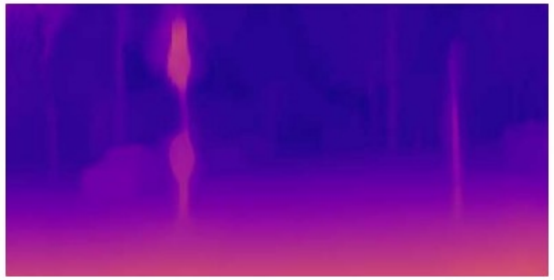
(b) Baseline Disparity Map

(a) Input Image

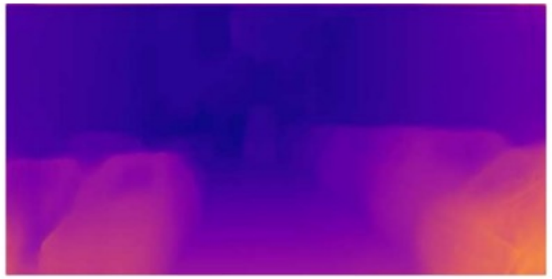


(b) Baseline Semantic Map

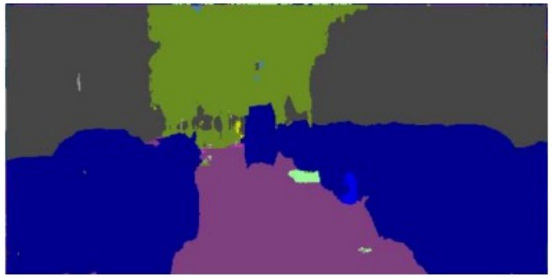
(c) SceneNet Semantic Map



(d) SceneNet Disparity Map



(c) SceneNet Disparity Map



(d) SceneNet Semantic Map

“Towards Scene Understanding: Unsupervised Monocular Depth Estimation with Semantic-aware Representation”, CVPR'19

The University of Texas at Austin
Electrical and Computer Engineering
Cockrell School of Engineering