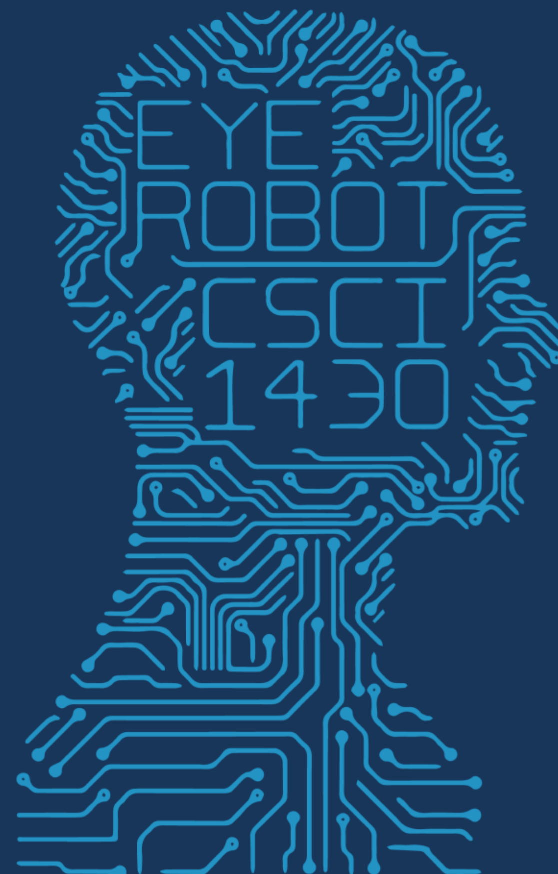
1950

Future Vision

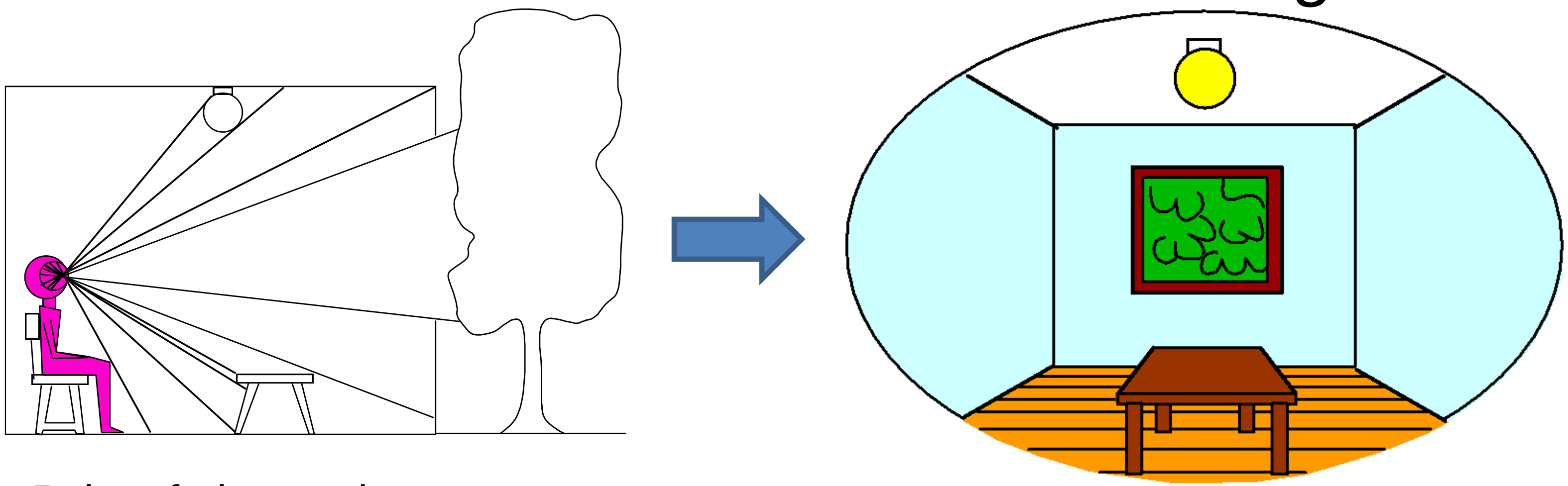


6 April 2020

Computer Vision

Dimensionality Reduction Machine (3D to 2D)

3D world *2D image*



Point of observation

Lengths are lost…

…and so area is lost.

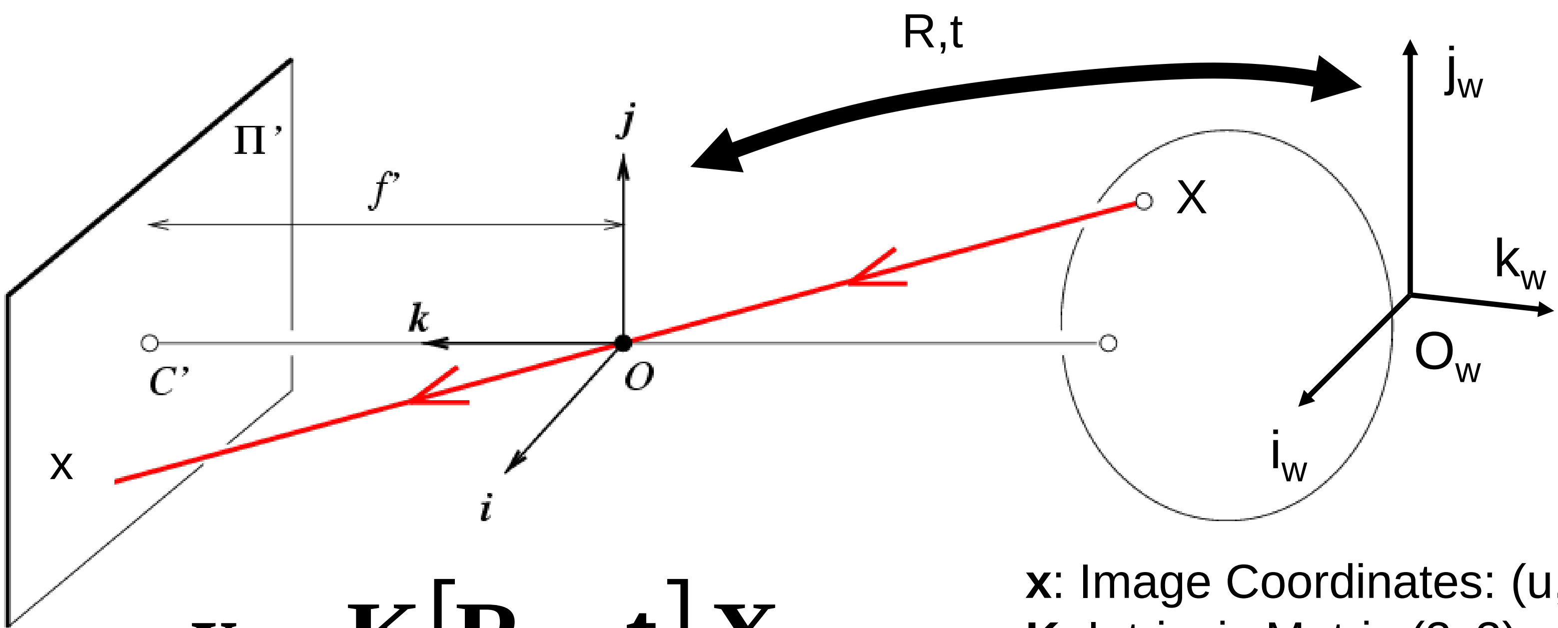
Angle preservation is lost…

…so parallel/perpendicular lines are lost.

How can we recover scene geometry to measure the world?

Slide Credit: Savarese

Camera (projection) matrix



$$\mathbf{x} = \mathbf{K}\underbrace{\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}}_{\text{Extrinsic Matrix}}\mathbf{X}$$

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

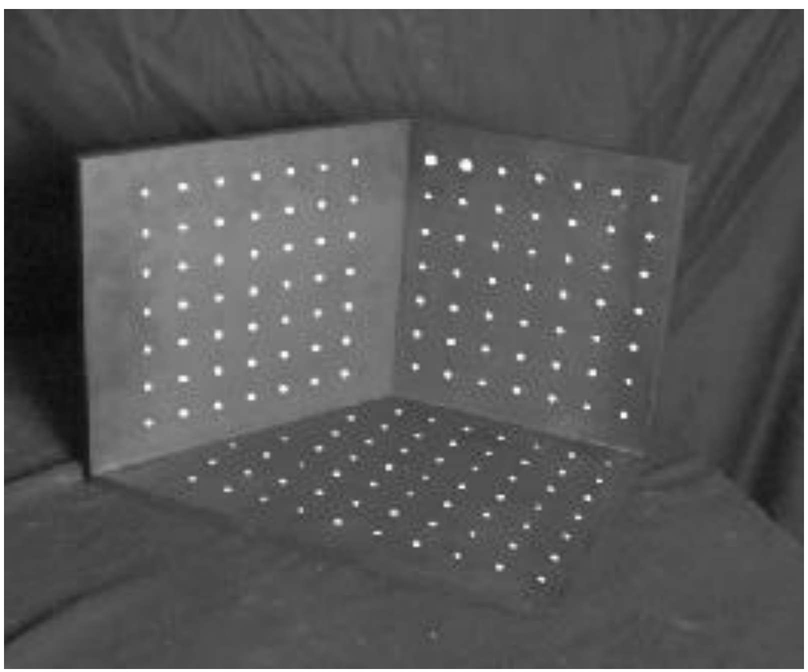
$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

James Hays

Calibrating the Camera

Use an scene with **known** geometry

- Correspond image points to 3d points
- Get least squares solution (or non-linear solution)



Known 2d image coords

Known 3d world locations

$$
\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \overset{\mathbf{M}}{\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
$$

Unknown Camera Parameters

Can we factorize M back to K [R | T]?

- Yes!
- We can directly solve for the individual entries of K [R | T].

$\mathbf{a}_n$ = nth column of A

Extracting camera parameters

$$\frac{M}{\rho} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \frac{\beta}{\sin\theta}\boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \frac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix} = \mathrm{K}\begin{bmatrix} \mathrm{R} & \mathrm{T} \end{bmatrix}$$

$$\underbrace{\qquad}_{A} \qquad \underbrace{\qquad}_{\mathbf{b}}$$

$$\mathrm{K} = \begin{bmatrix} \alpha & -\alpha\cot\theta & u_o \\ 0 & \frac{\beta}{\sin\theta} & v_o \\ 0 & 0 & 1 \end{bmatrix}$$

Box 1

$$\mathrm{A} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ \mathbf{a}_3^{\mathrm{T}} \end{bmatrix} \qquad \mathrm{b} = \begin{bmatrix} \mathrm{b}_1 \\ \mathrm{b}_2 \\ \mathrm{b}_3 \end{bmatrix}$$

Estimated values

Intrinsic

$$\rho = \frac{\pm 1}{|\mathbf{a}_3|} \qquad u_o = \rho^2(\mathbf{a}_1 \cdot \mathbf{a}_3) \qquad \mathrm{v}_o = \rho^2(\mathbf{a}_2 \cdot \mathbf{a}_3)$$

$$\cos\theta = \frac{(\mathbf{a}_1 \times \mathbf{a}_3) \cdot (\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_1 \times \mathbf{a}_3| \cdot |\mathbf{a}_2 \times \mathbf{a}_3|}$$

James Hays

Extracting camera parameters

$$\frac{\mathcal{M}}{\rho} = \begin{pmatrix} \boxed{\begin{matrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T \\ \dfrac{\beta}{\sin\theta}\boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T \\ \boldsymbol{r}_3^T \end{matrix}} & \boxed{\begin{matrix} \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ t_z \end{matrix}} \end{pmatrix} = K\begin{bmatrix} R & T \end{bmatrix}$$

$$\mathbf{A} \qquad\qquad \mathbf{b}$$

$$\boxed{A = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \end{bmatrix} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}}$$

Estimated values

Intrinsic

$$\alpha = \rho^2 |\mathbf{a}_1 \times \mathbf{a}_3| \sin\theta$$

$$\beta = \rho^2 |\mathbf{a}_2 \times \mathbf{a}_3| \sin\theta$$

James Hays

Extracting camera parameters

$$\frac{\mathcal{M}}{\rho} = \begin{pmatrix} \alpha \boldsymbol{r}_1^T - \alpha \cot\theta \boldsymbol{r}_2^T + u_0 \boldsymbol{r}_3^T & \alpha t_x - \alpha \cot\theta t_y + u_0 t_z \\ \dfrac{\beta}{\sin\theta}\boldsymbol{r}_2^T + v_0 \boldsymbol{r}_3^T & \dfrac{\beta}{\sin\theta} t_y + v_0 t_z \\ \boldsymbol{r}_3^T & t_z \end{pmatrix} = K\begin{bmatrix} R & T \end{bmatrix}$$

$$\underbrace{}_{A} \qquad \underbrace{}_{\mathbf{b}}$$

$$A = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Estimated values

Extrinsic

$$\mathbf{r}_1 = \frac{(\mathbf{a}_2 \times \mathbf{a}_3)}{|\mathbf{a}_2 \times \mathbf{a}_3|} \qquad \mathbf{r}_3 = \frac{\pm \mathbf{a}_3}{|\mathbf{a}_3|}$$

$$\mathbf{r}_2 = \mathbf{r}_3 \times \mathbf{r}_1 \qquad T = \rho\, K^{-1}\mathbf{b}$$

James Hays

Can we factorize M back to K [R | T]?

Yes!

We can also use *RQ* factorization (not QR)

- R in RQ is not rotation matrix R; crossed names!

- *R* (right diagonal) is K
- *Q* (orthogonal basis) is R the rotation matrix.
- T, the last column of [R | T], is inv(K) * last column of M.
  - But you need to do a bit of post-processing to make sure that the matrices are valid. See http://ksimek.github.io/2012/08/14/decompose/

James Hays

Recovering the camera center

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & s & u_0\\0 & \beta & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_x\\r_{21} & r_{22} & r_{23} & t_y\\r_{31} & r_{32} & r_{33} & t_z\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

t

This is not the camera center C.
It is –RC, as the point is rotated before t_x, t_y, and t_z are added

So we need
$-R^{-1} K^{-1} m_4$ to get C.

m_4

$$\begin{bmatrix}su\\sv\\s\end{bmatrix} = \begin{bmatrix}* & * & * & *\\* & * & * & *\\* & * & * & *\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

Q

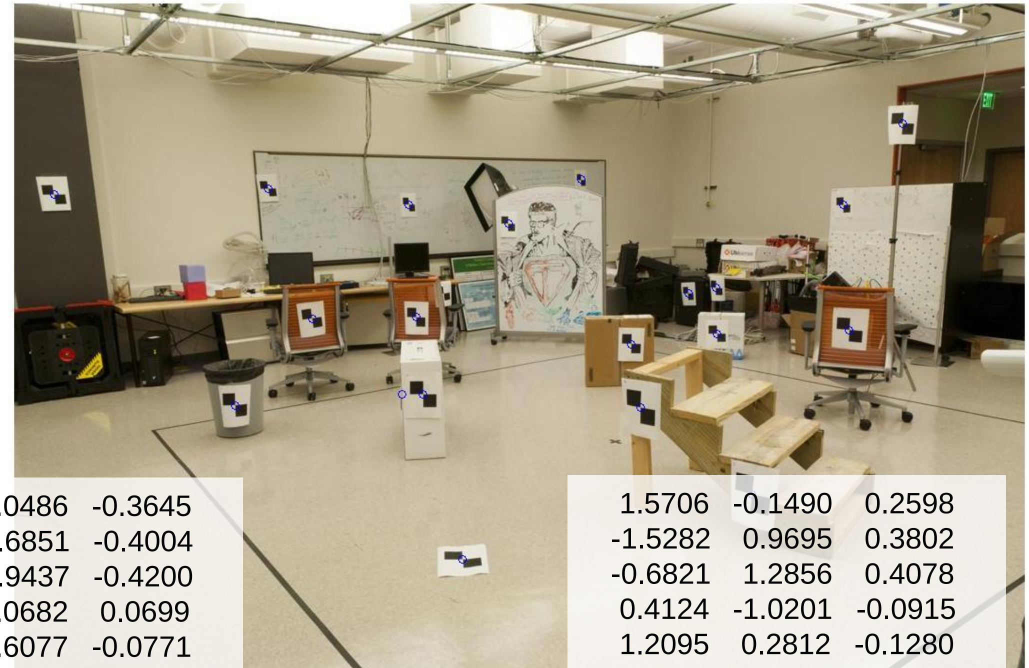
This is **t** × K

So K^{-1} m_4 is **t**

Q is K × R.
So we just need $-Q^{-1}$ m_4

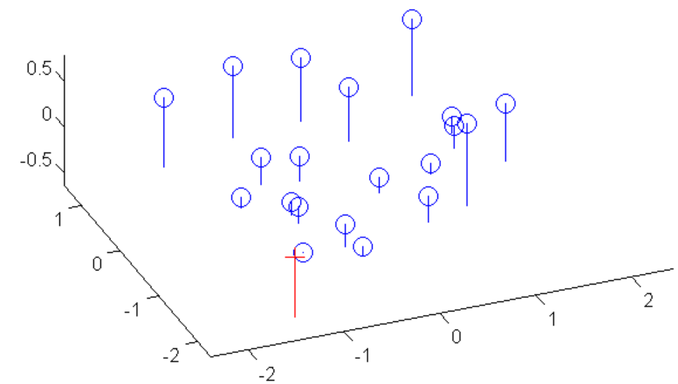
James Hays

Estimate of camera center



1.0486	-0.3645
-1.6851	-0.4004
-0.9437	-0.4200
1.0682	0.0699
0.6077	-0.0771
1.2543	-0.6454
-0.2709	0.8635
-0.4571	-0.3645
-0.7902	0.0307
0.7318	0.6382
-1.0580	0.3312
0.3464	0.3377
0.3137	0.1189
-0.4310	0.0242
-0.4799	0.2920
0.6109	0.0830
-0.4081	0.2920
-0.1109	-0.2992
0.5129	-0.0575
0.1406	-0.4527

1.5706	-0.1490	0.2598
-1.5282	0.9695	0.3802
-0.6821	1.2856	0.4078
0.4124	-1.0201	-0.0915
1.2095	0.2812	-0.1280
0.8819	-0.8481	0.5255
-0.9442	-1.1583	-0.3759
0.0415	1.3445	0.3240
-0.7975	0.3017	-0.0826
-0.4329	-1.4151	-0.2774
-1.1475	-0.0772	-0.2667
-0.5149	-1.1784	-0.1401
0.1993	-0.2854	-0.2114
-0.4320	0.2143	-0.1053
-0.7481	-0.3840	-0.2408
0.8078	-0.1196	-0.2631
-0.7605	-0.5792	-0.1936
0.3237	0.7970	0.2170
1.3089	0.5786	-0.1887
1.2323	1.4421	0.4506



Great! So now I have K and Rt

Well, what is that useful for?

Goal: reconstruct depth.

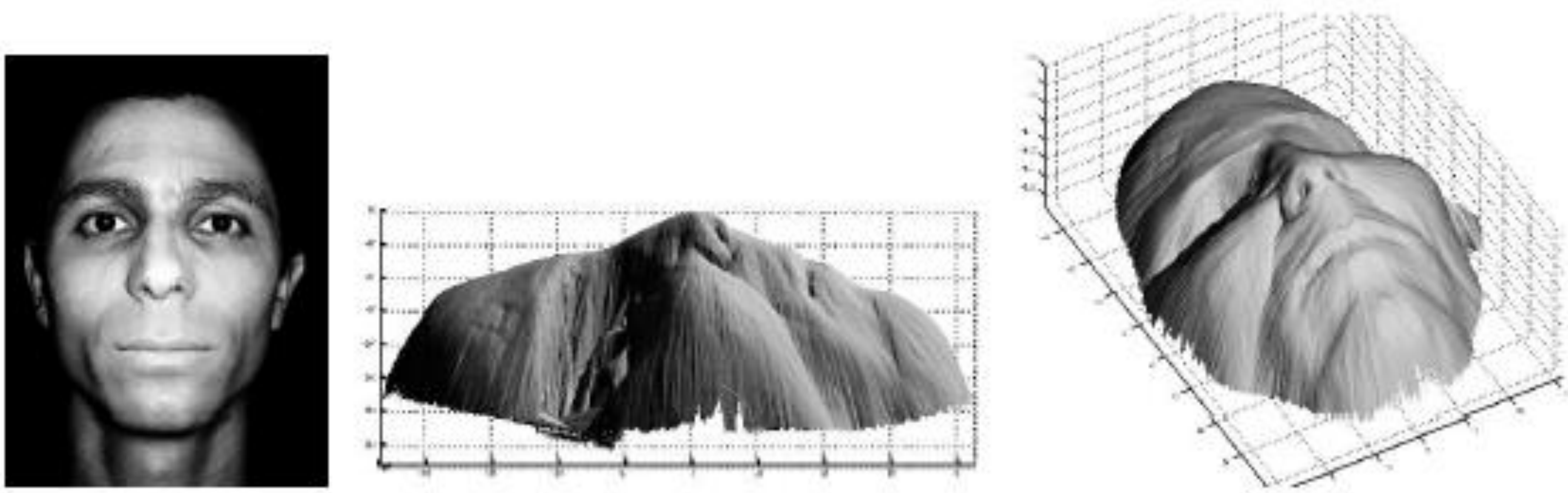
So far: we have 'calibrated' one camera.

Or, potentially two…

Think-Pair-Share

What visual or physiological cues help us to perceive 3D shape and depth?

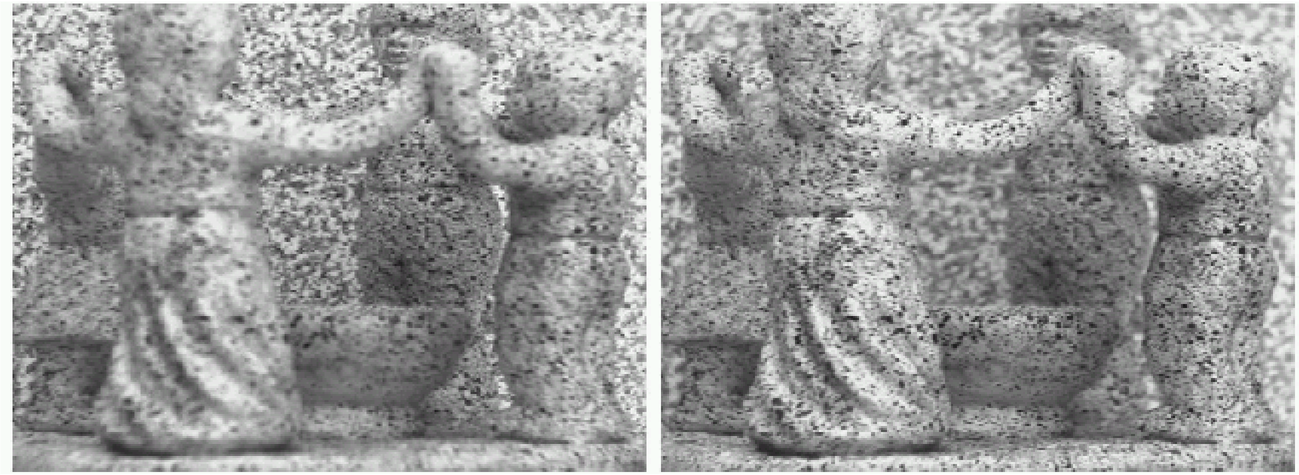
Shading



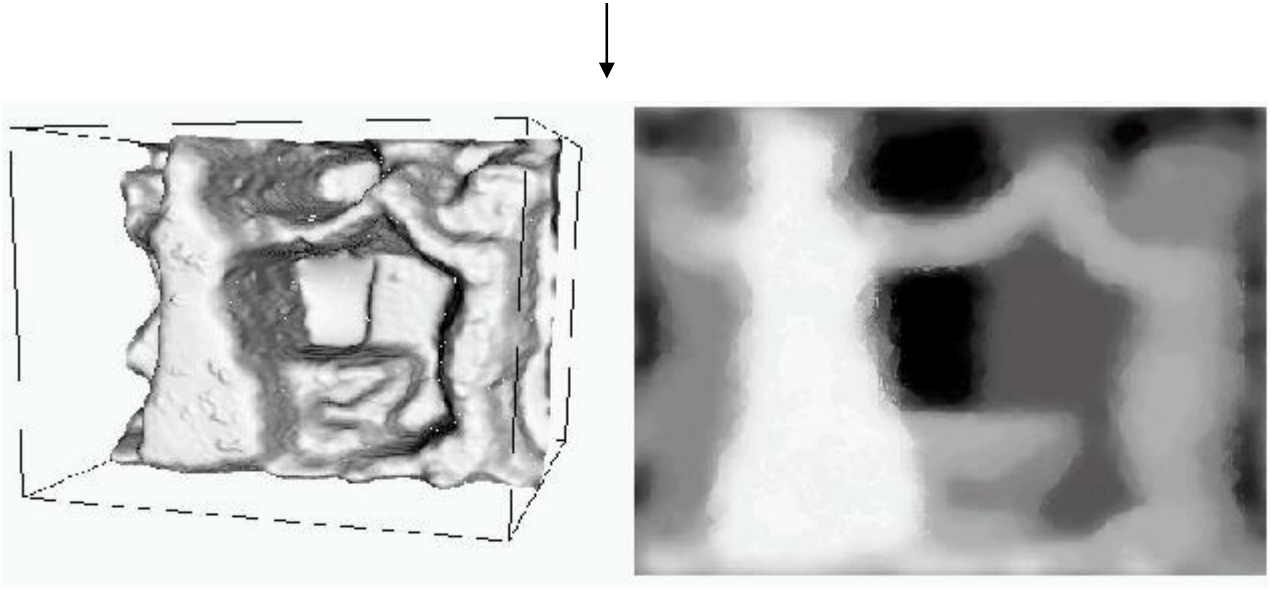
a) b) c)

[Figure from Prados & Faugeras 2006]

Focus/defocus



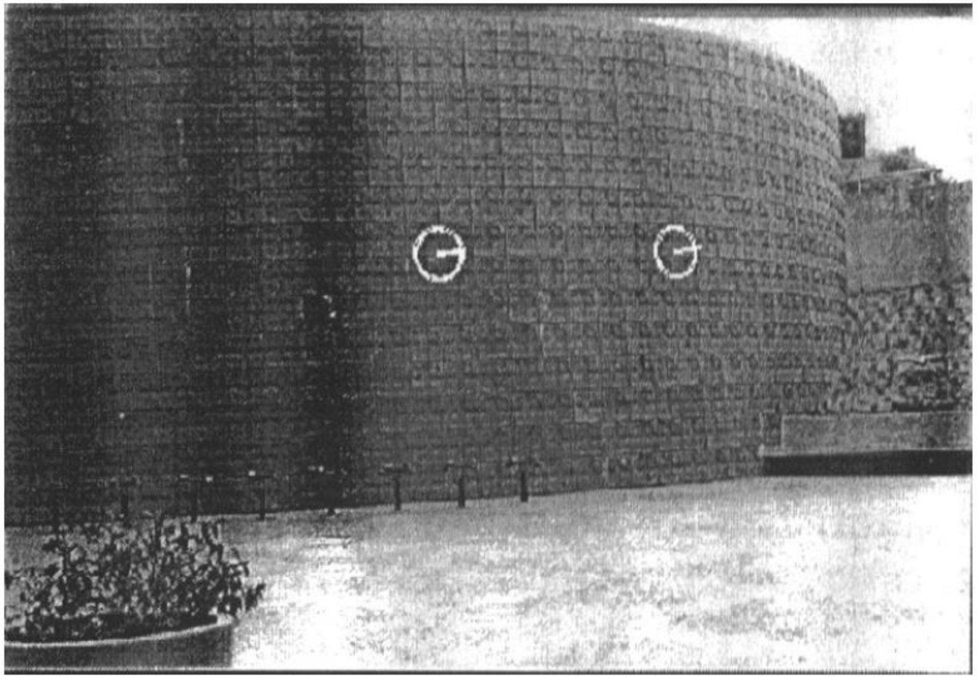
Images from same point of view, different camera parameters

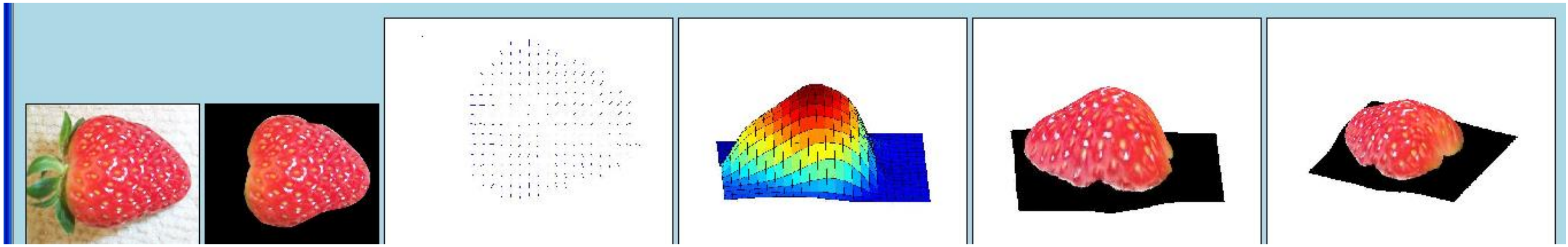


3d shape / depth estimates

[figs from H. Jin and P. Favaro, 2002]

Texture



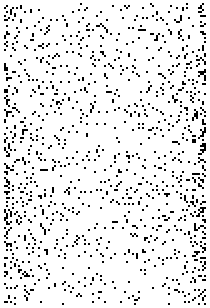


[From A.M. Loh. The recovery of 3-D structure using visual texture patterns. PhD thesis]

Perspective effects

Image credit: S. Seitz

Motion



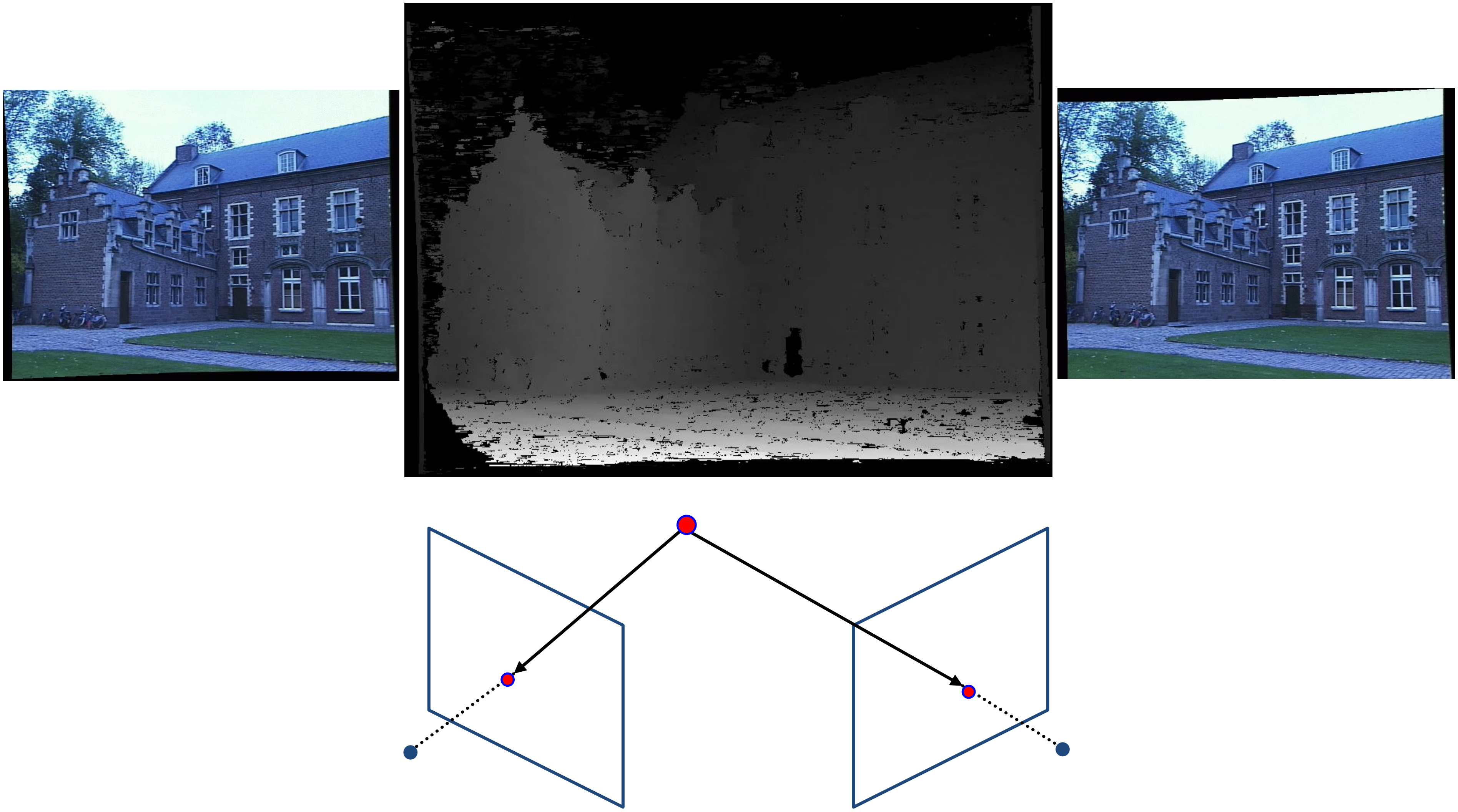
Figures from L. Zhang

http://www.brainconnection.com/teasers/?main=illusion/motion-shape

Occlusion

Rene Magritt'e famous painting *Le Blanc-Seing* (literal translation: "The Blank Signature") roughly translates as "free hand" or "free rein".

Stereo



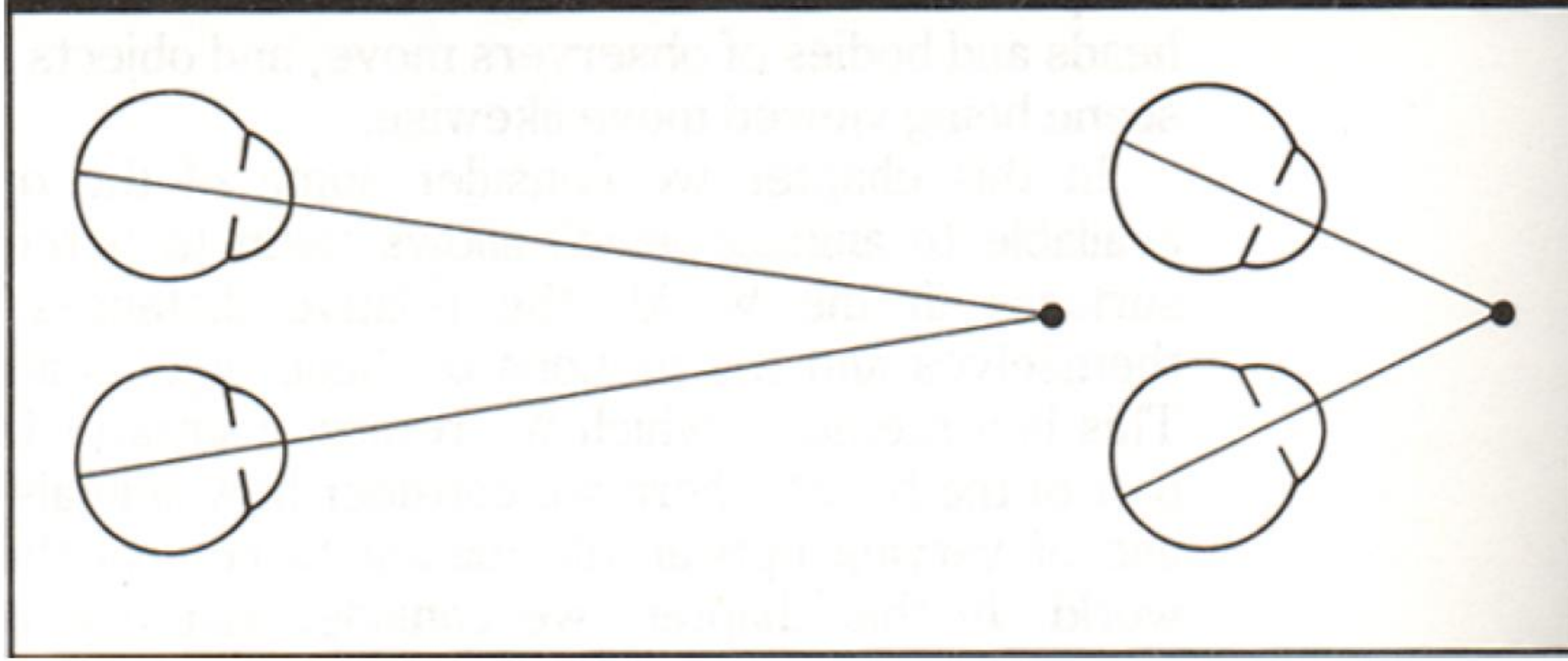
Slides: James Hays and Kristen Grauman

If stereo were critical for depth perception, navigation, recognition, etc., then rabbits would never have evolved.

Human stereopsis

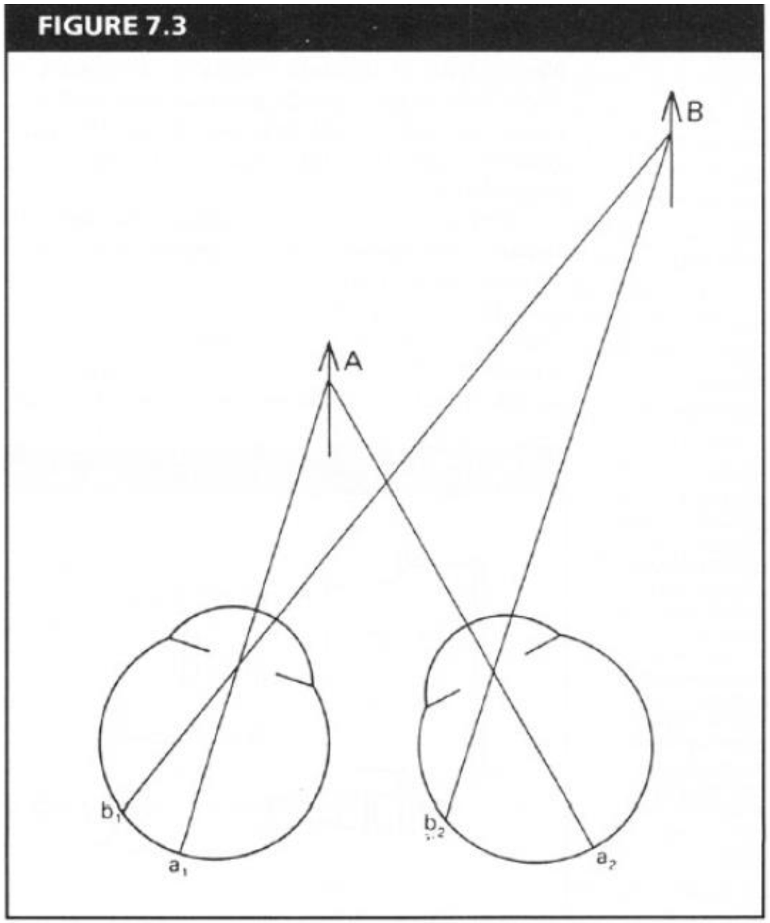
Human eyes **fixate** on point in space – rotate so that corresponding images form in centers of fovea.

FIGURE 7.1



From Bruce and Green, Visual Perception, Physiology, Psychology and Ecology

Human stereopsis: disparity



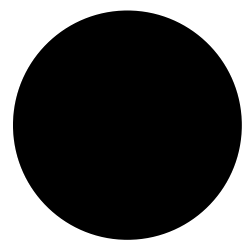
From Bruce and Green, Visual Perception, Physiology, Psychology and Ecology

Disparity occurs when eyes fixate on one object; others appear at different visual angles.

Disparity is distance from b1 to b2 along retina.

Adapted from David Forsyth, UC Berkeley

Yes, you can be stereoblind.



Random dot stereograms

- Julesz 1960:

  Do we identify local brightness patterns before fusion (monocular process) or after (binocular)?

- Think Pair Share – yes / no? how to test?

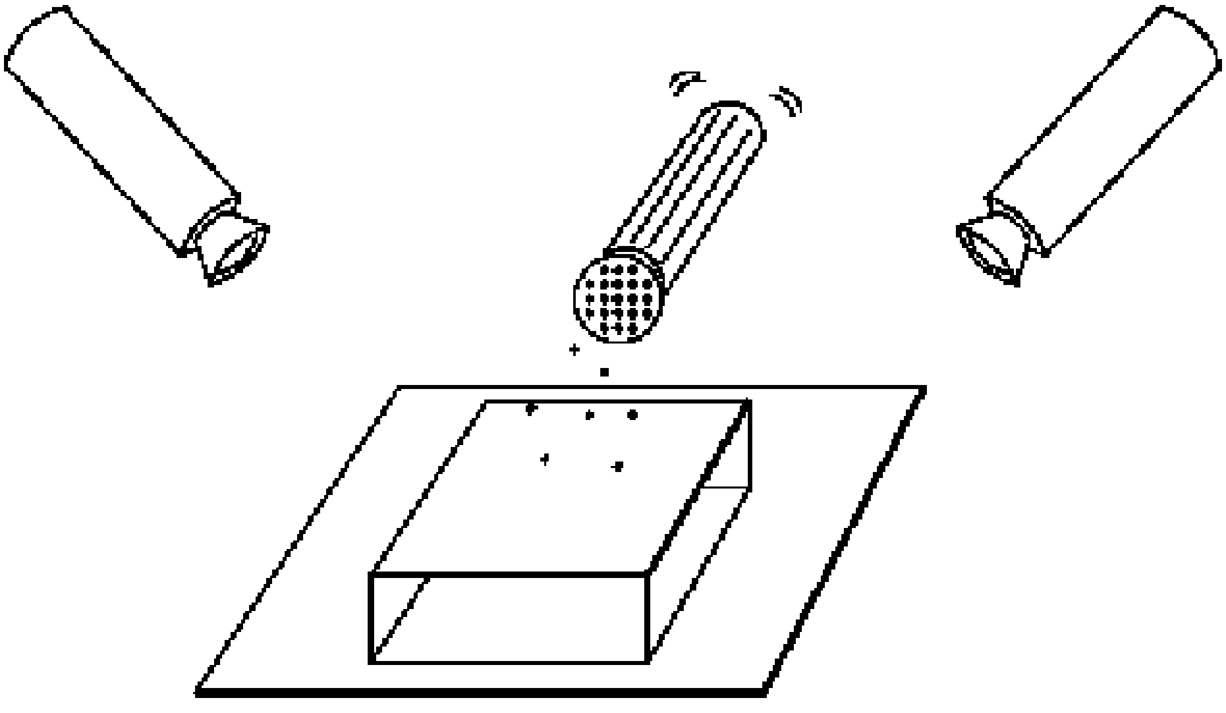
Random dot stereograms

- Julesz 1960:

  Do we identify local brightness patterns before fusion (monocular process) or after (binocular)?

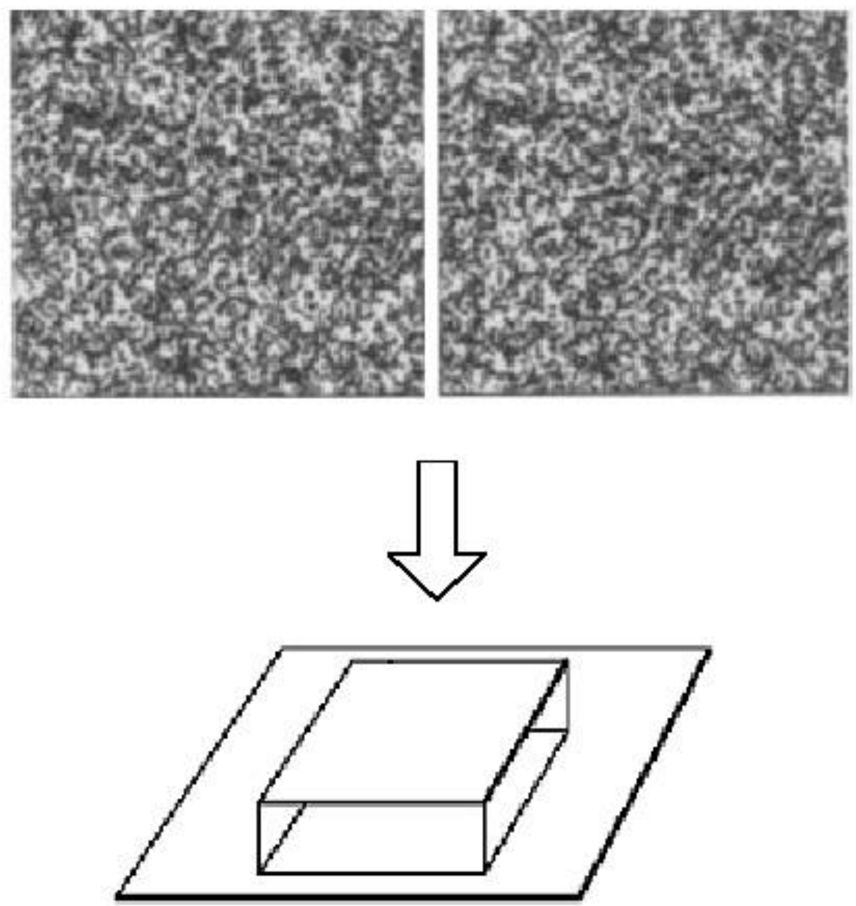
- To test: pair of synthetic images obtained by randomly spraying black dots on white objects

Random dot stereograms

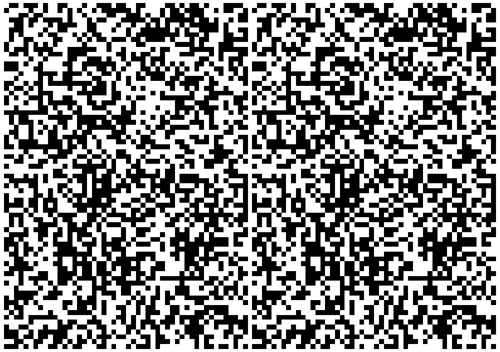


Forsyth & Ponce

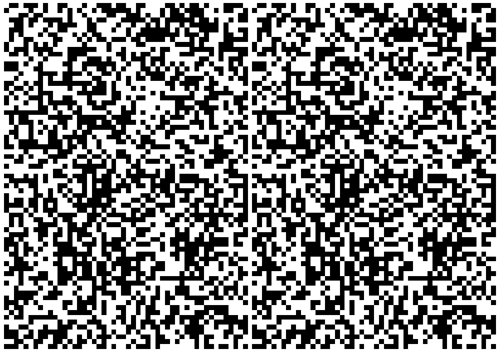
Random dot stereograms



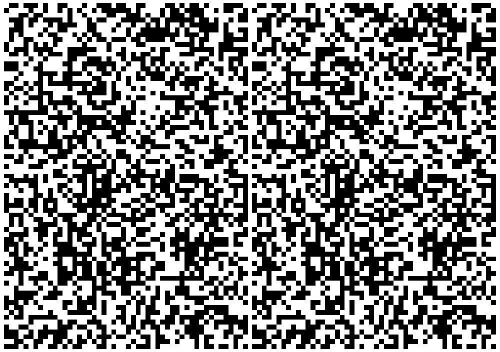
1. Create an image of suitable size. Fill it with random dots. Duplicate the image.



2. Select a region in one image.



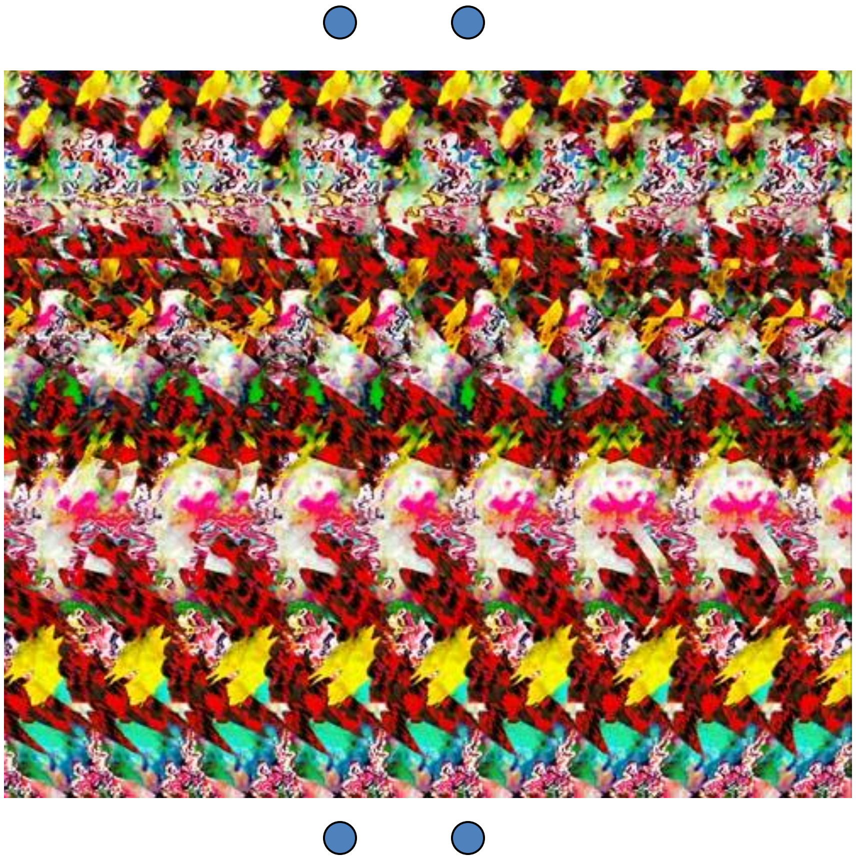
3. Shift this region horizontally by a small amount. The stereogram is complete.



Random dot stereograms

- When viewed monocularly, they appear random; when viewed stereoscopically, see 3d structure.
- Human binocular fusion not directly associated with the physical retinas; must involve the central nervous system (V2, for instance).
- Imaginary "*cyclopean retina*" that combines the left and right image stimuli as a single unit.
- High level scene understanding not required for stereo…but, high level scene understanding is arguably *better* than stereo.

Autostereograms – ‘Magic Eye’



Exploit disparity as depth cue using single image.

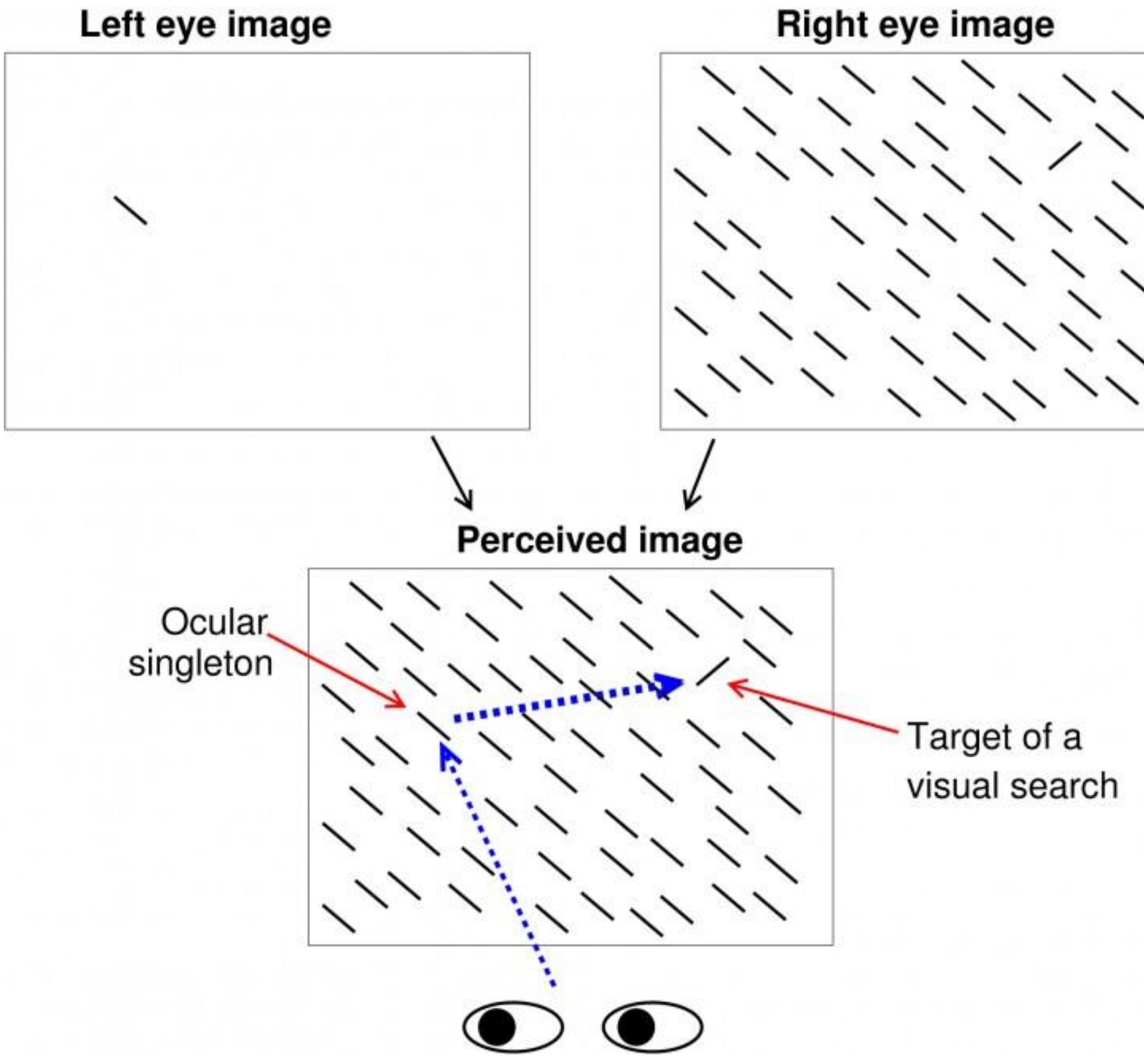
(Single image random dot stereogram, Single image stereogram)

Images from magiceye.com

Autostereograms

Images from magiceye.com

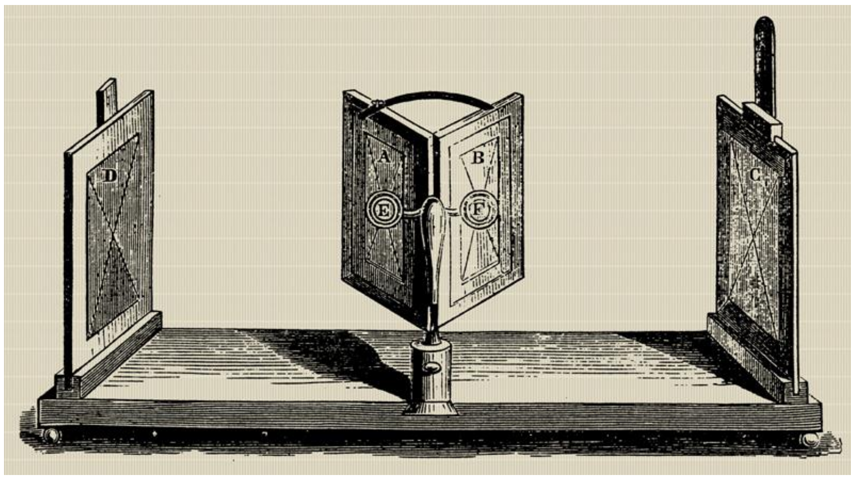
Stereo attention is weird wrt. mind's eye



[Li Zhaoping, Understanding Vision]

Stereo photography and stereo viewers

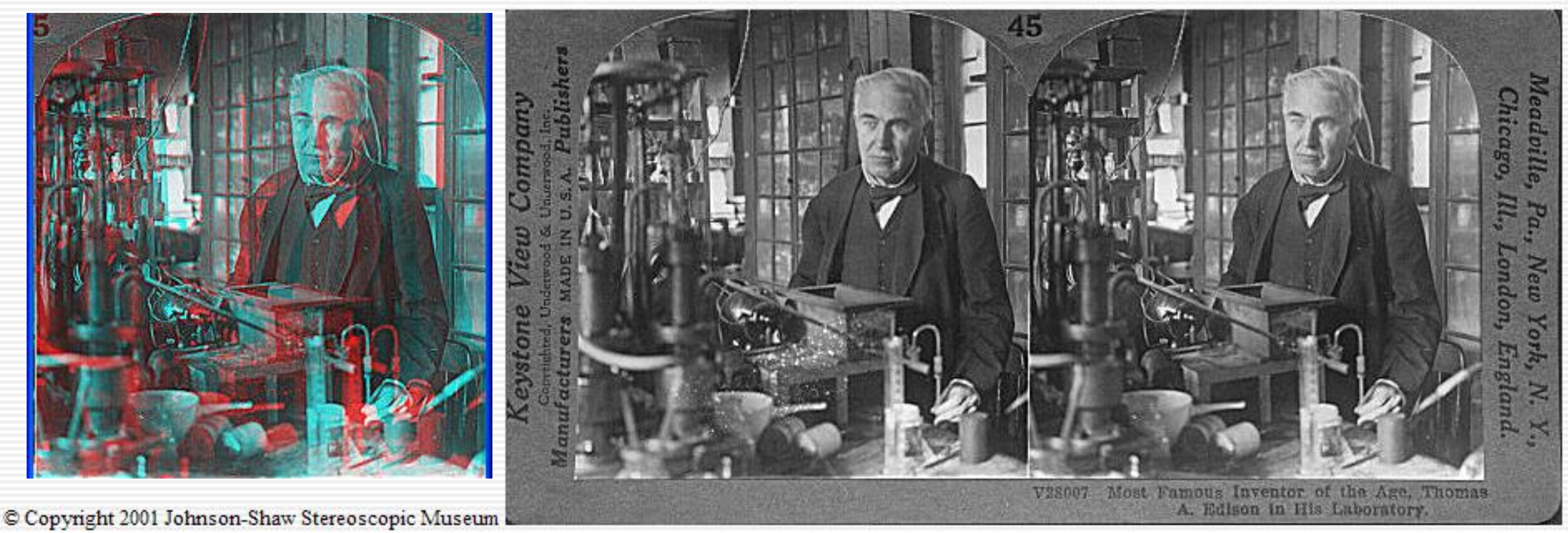
Take two pictures of the same subject from two slightly different viewpoints and display so that each eye sees only one of the images.



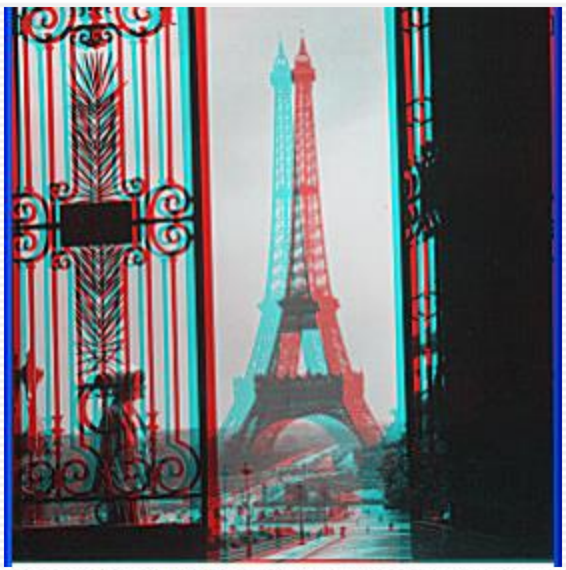
Invented by Sir Charles Wheatstone, 1838

Image from fisher-price.com

Anaglyph stereo



http://www.johnsonshawmuseum.org



http://www.johnsonshawmuseum.org

Wiggle images

http://www.well.com/~jimg/stereo/stereo_list.html

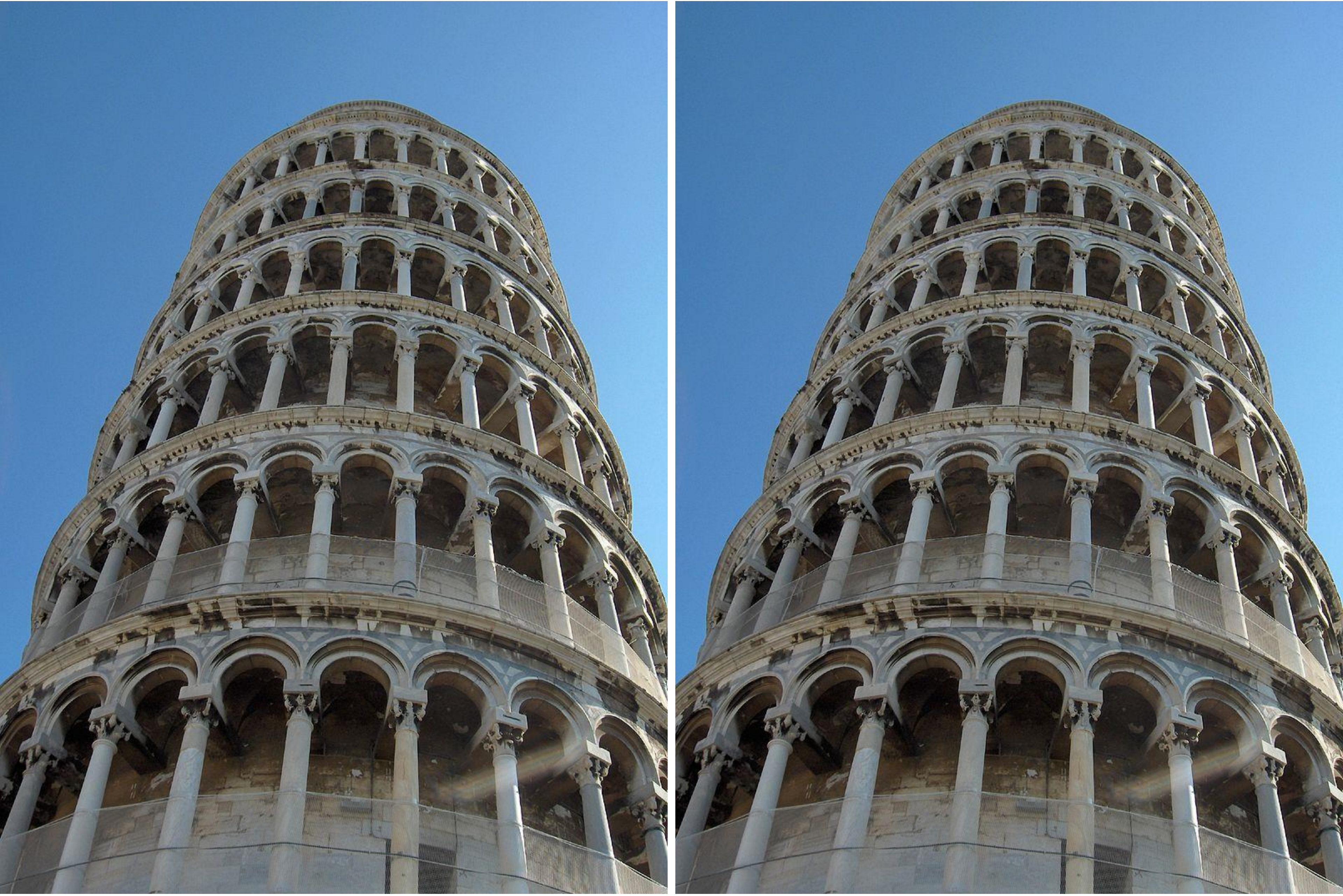



Photo: Georges Jansoon. Illusion: Frederick Kingdom, Ali Yoonessi and Elena

Frederick Kingdom

Stereo vision

Two cameras, simultaneous views

Single moving camera and static scene

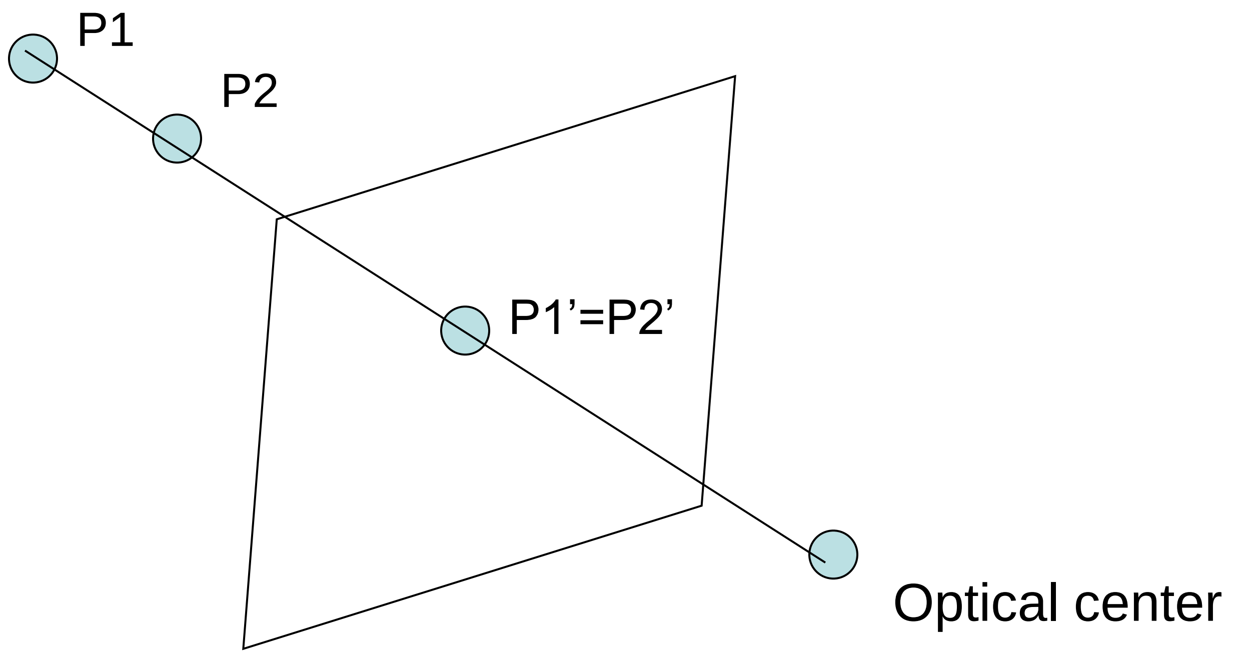
Why multiple views?

Structure and depth can be ambiguous from single views...

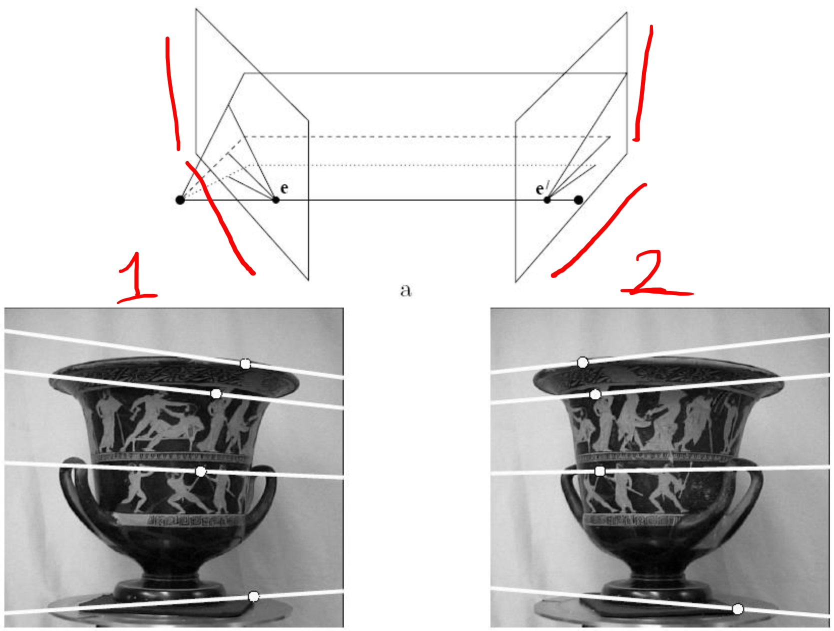
Images from Lana Lazebnik

Why multiple views?

Points at different depths along a line project to a single point



Multiple views



Hartley and Zisserman

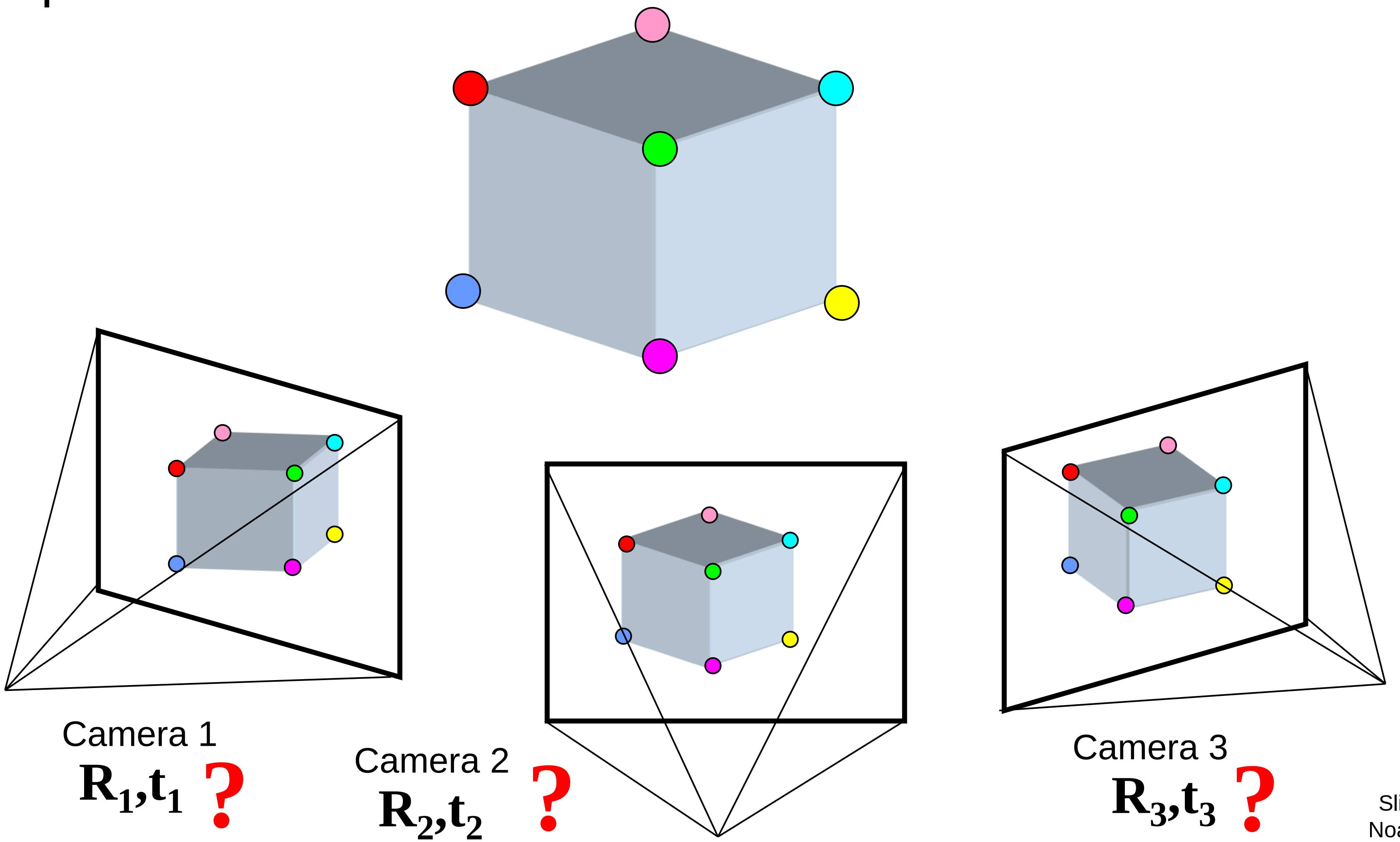
SLAM

Simultaneous
location localization
and
mapping

- Stereo vision
- Structure from motion
- Optical flow

Multi-view geometry problems

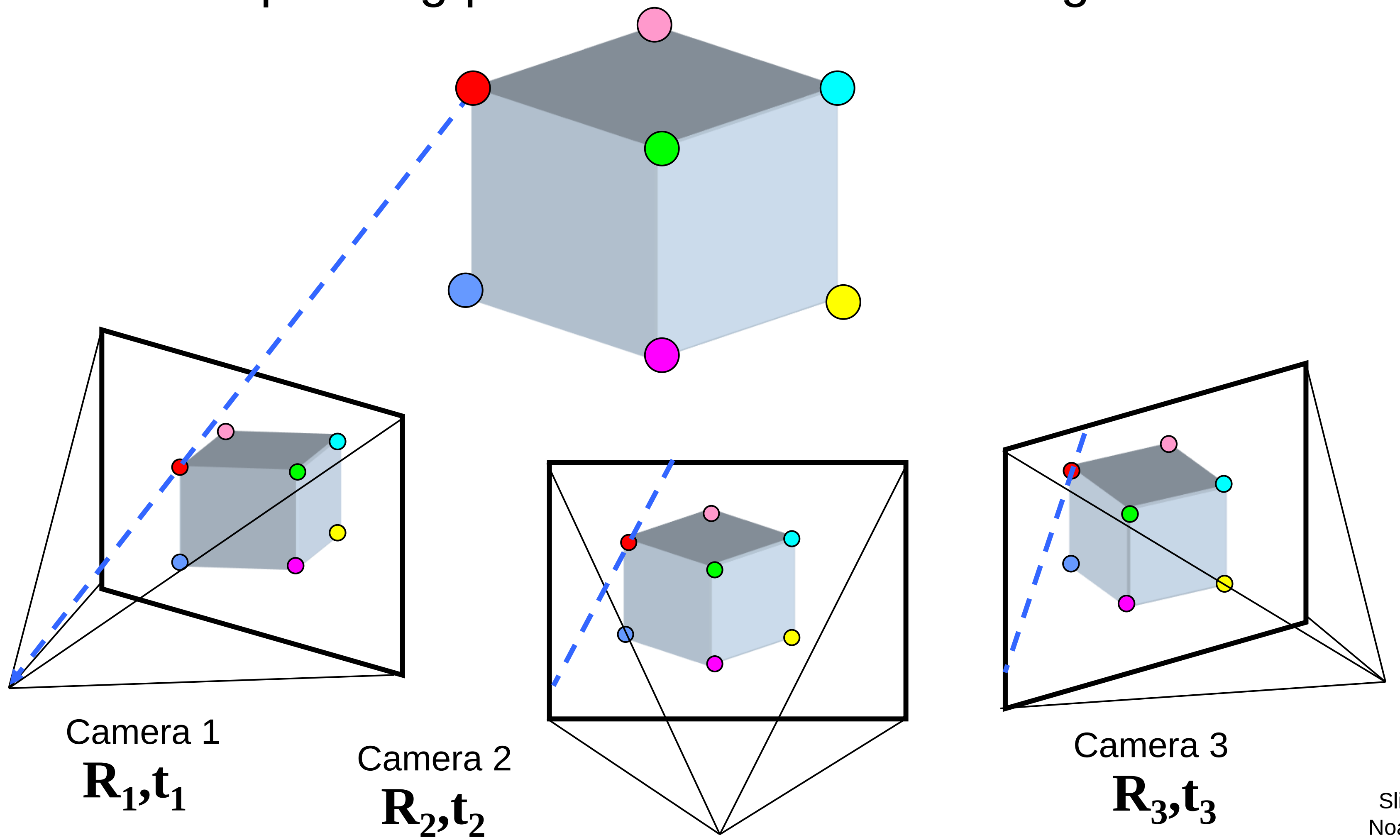
- **Camera 'Motion':** Given a set of corresponding 2D/3D points in two or more images, compute the camera parameters.



Slide credit: Noah Snavely

Multi-view geometry problems

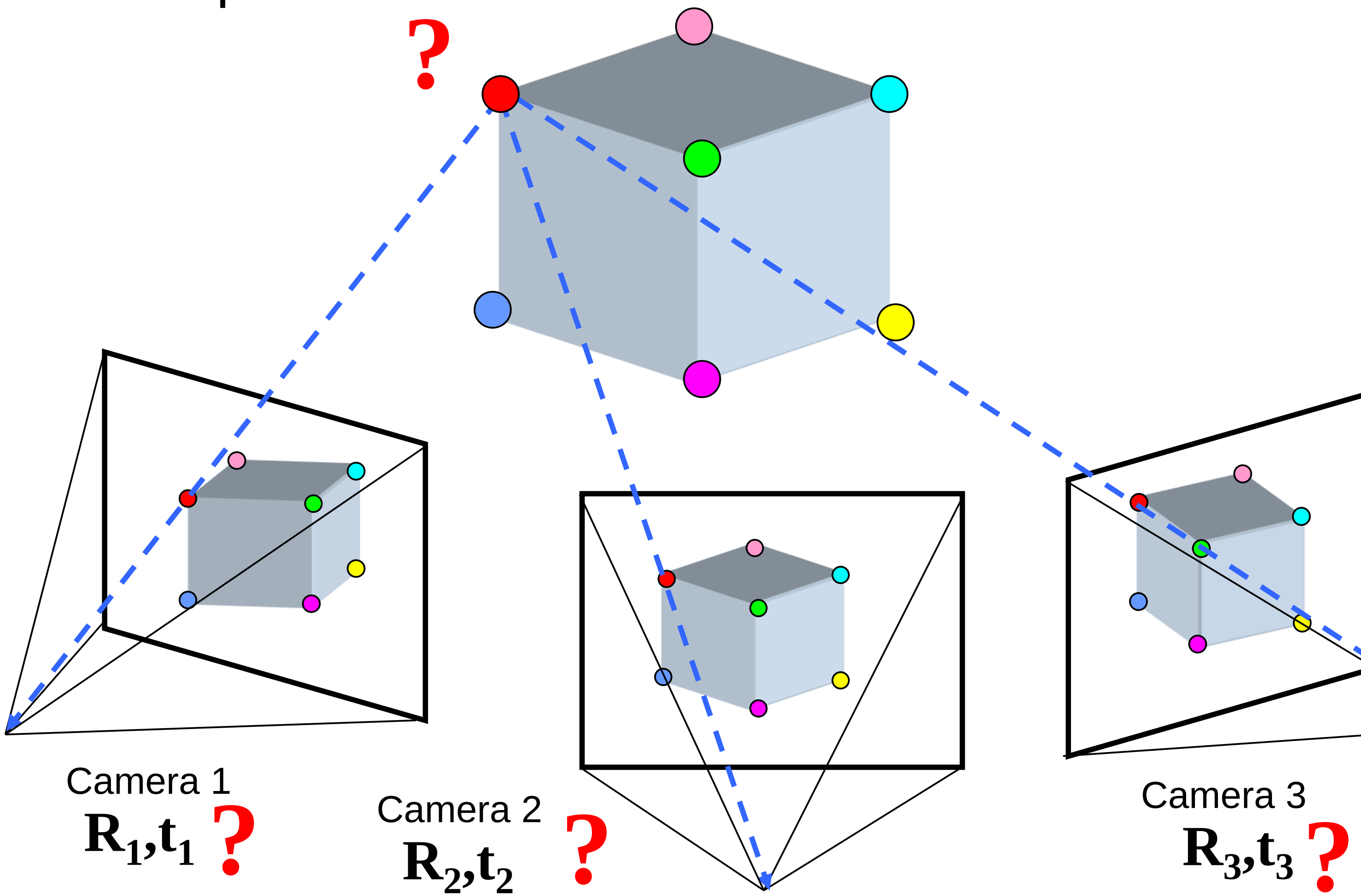
- **Stereo correspondence:** Given known camera parameters and a point in one of the images, where could its corresponding points be in the other images?



Slide credit: Noah Snavely

Multi-view geometry problems

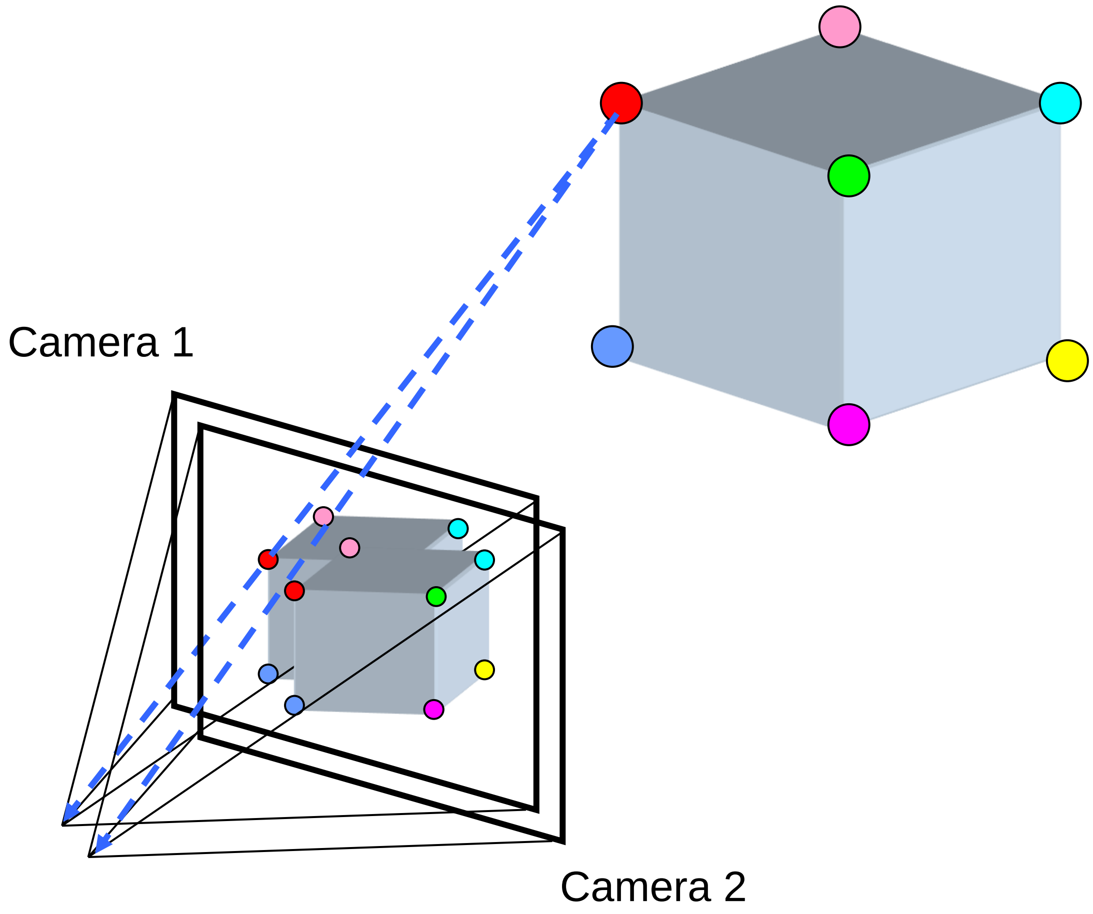
- **Structure from Motion:** Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point



Slide credit: Noah Snavely

Multi-view geometry problems

- **Optical flow:** Given two images, find the location of a world point in a second close-by image with no camera info.



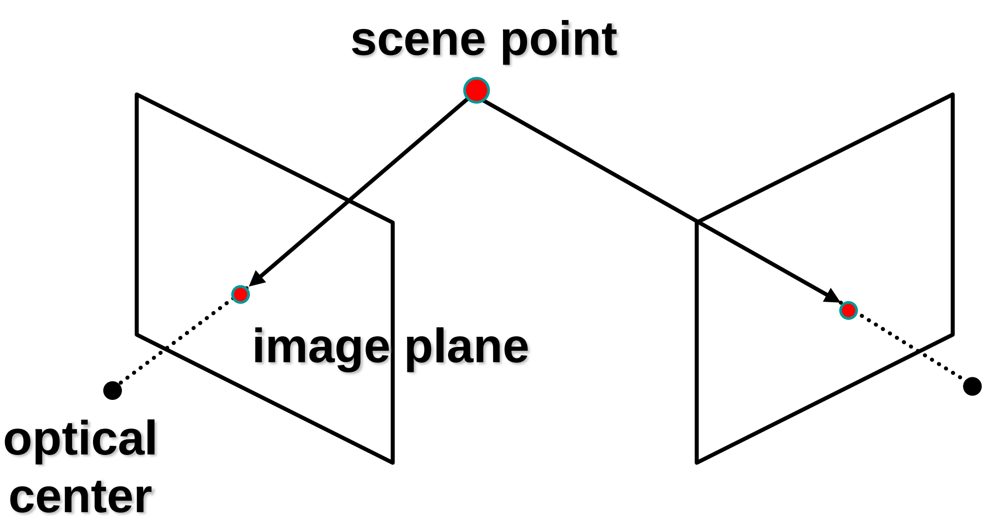
Multiple views - Dogception

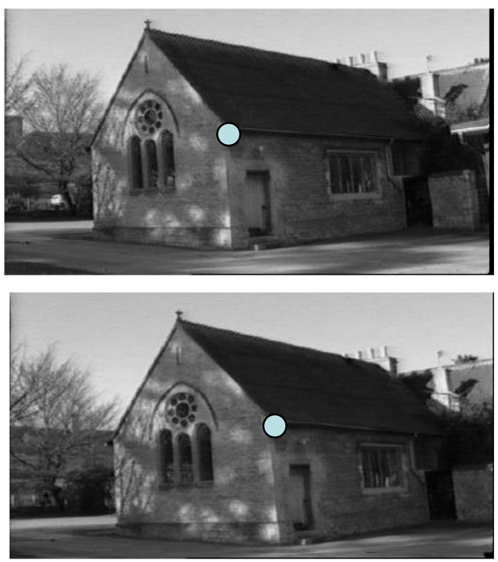
Estimating depth with stereo

- **Stereo**: shape from "motion" between two views
- We'll need to consider:
  - Info on camera pose ("calibration")
  - Image point correspondences

intrinsic K

relative extrinsics

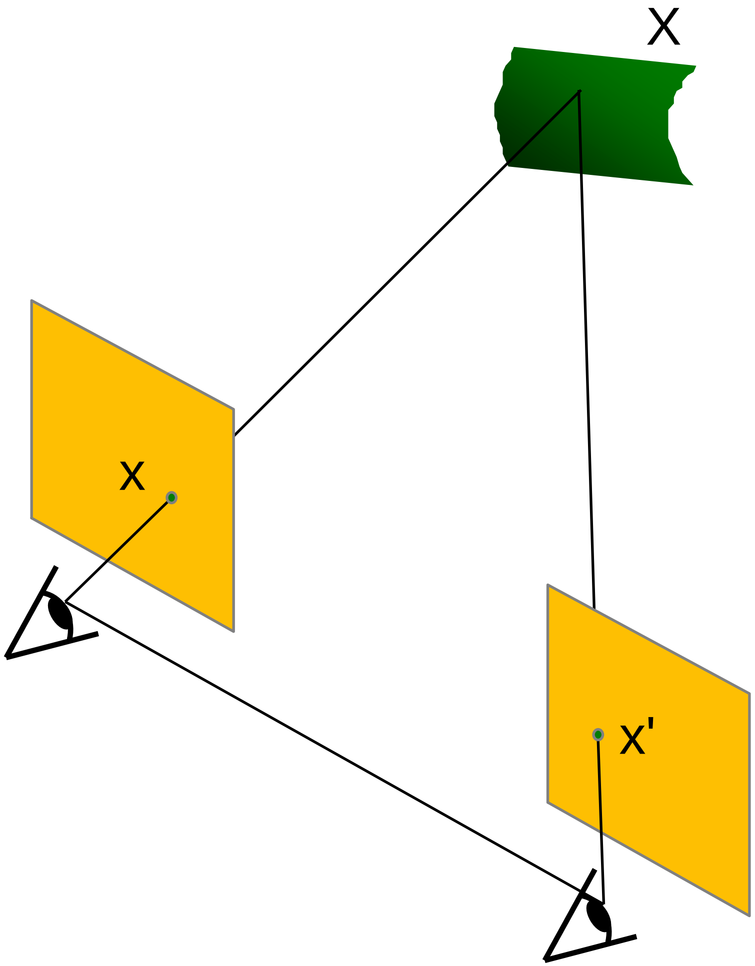


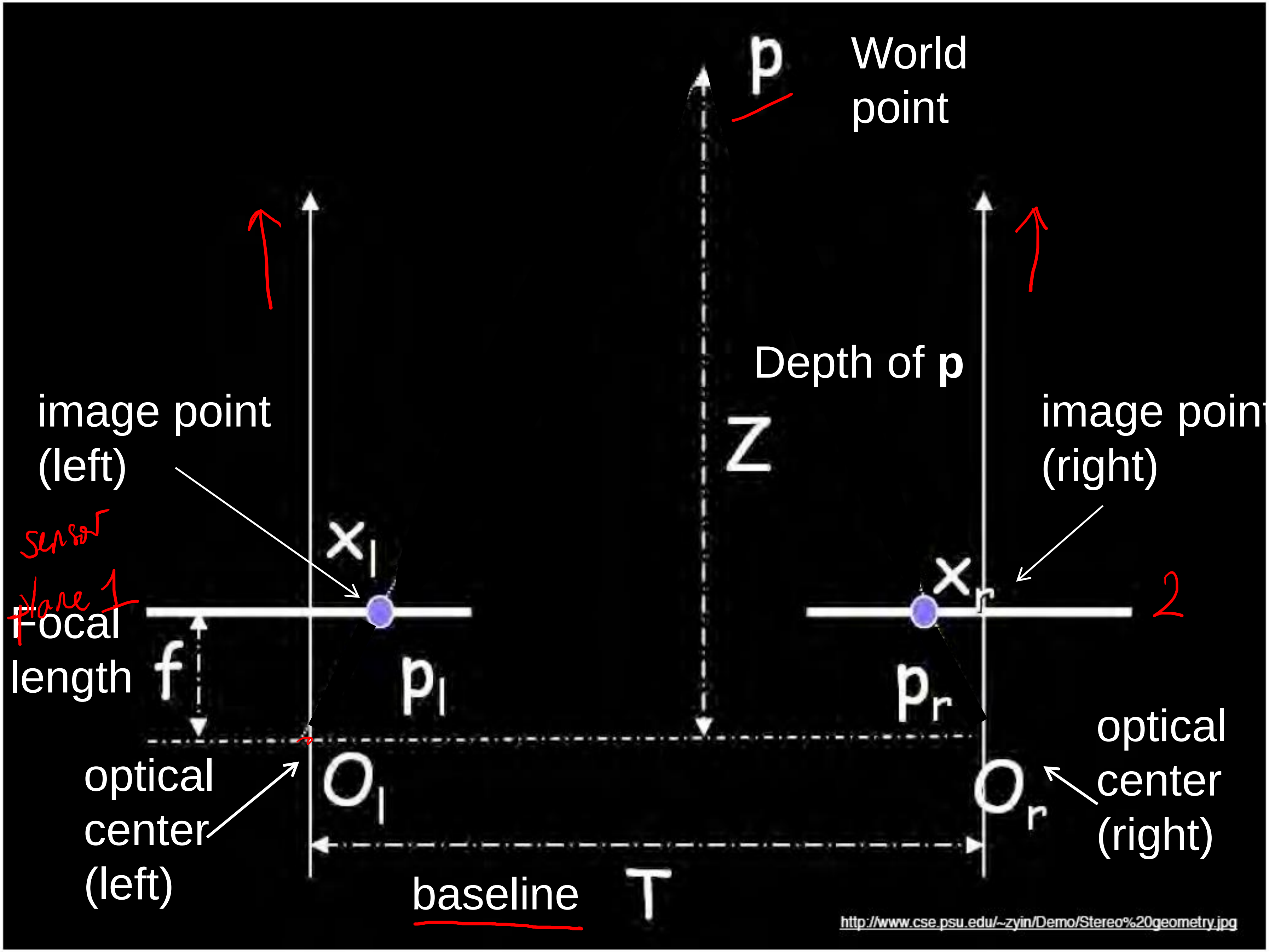


Geometry for a simple stereo system

- Assume:
  - parallel optical axes,
  - known camera parameters (i.e., calibrated cameras):

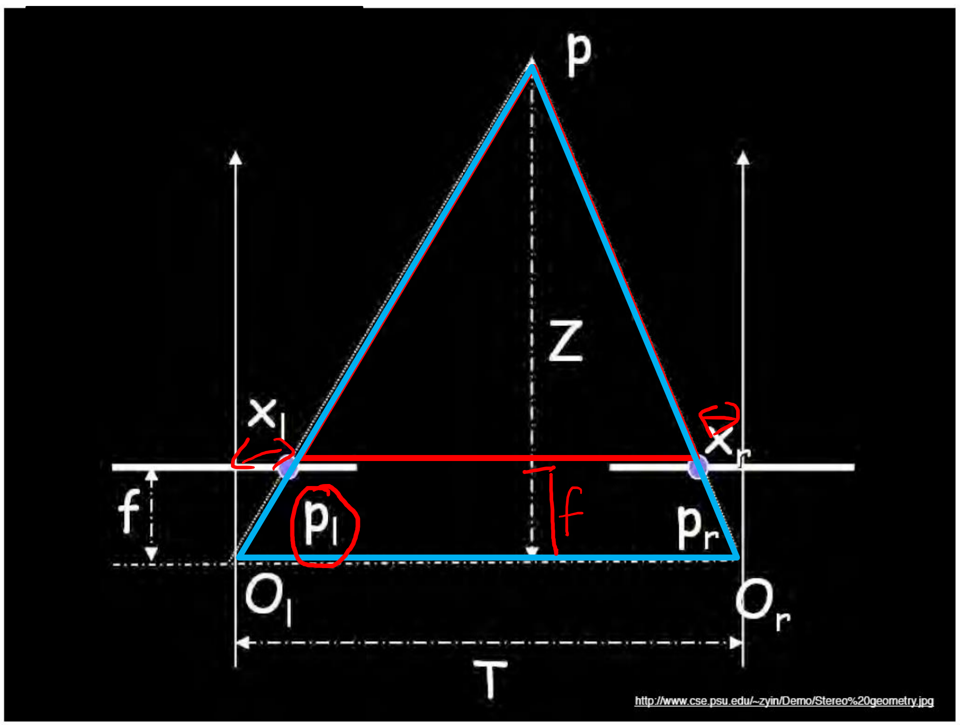
- Goal: recover depth of X by finding image coordinate x' that corresponds to x




p
World point
Depth of **p**
Z
image point (left)
image point (right)
x_l
x_r
sensor plane 1
2
Focal length
f
p_l
p_r
optical center (left)
optical center (right)
O_l
O_r
baseline
T
http://www.cse.psu.edu/~zyin/Demo/Stereo%20geometry.jpg

Geometry for a simple stereo system

- Assume parallel optical axes, known camera parameters (i.e., calibrated cameras). **What is expression for Z?**



Similar triangles (p_l, P, p_r) and (O_l, P, O_r):

$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

$$Z = f\frac{T}{x_r - x_l}$$

disparity

Depth from disparity

image I(x,y)

Disparity map D(x,y)

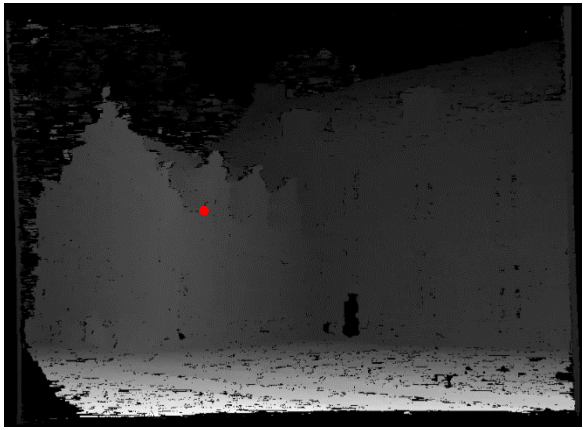


image I´(x´,y´)

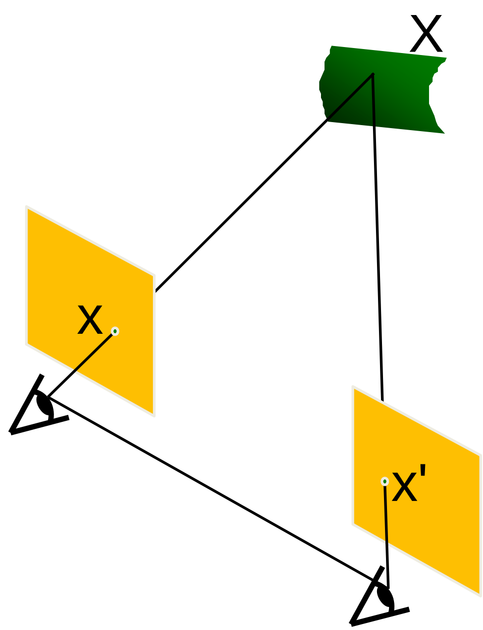
$$(x´,y´)=(x+\underline{D(x,y)}, y)$$

If we could find the **corresponding points** in two images, we could **estimate relative depth**…

James Hays

Depth from disparity

- Goal: recover depth by finding image coordinate x’ that corresponds to x
- Sub-Problems
  1. Calibration: How do we recover the relation of the cameras (if not already known)?
  2. Correspondence: How do we search for the matching point x’?



What do we need to know?

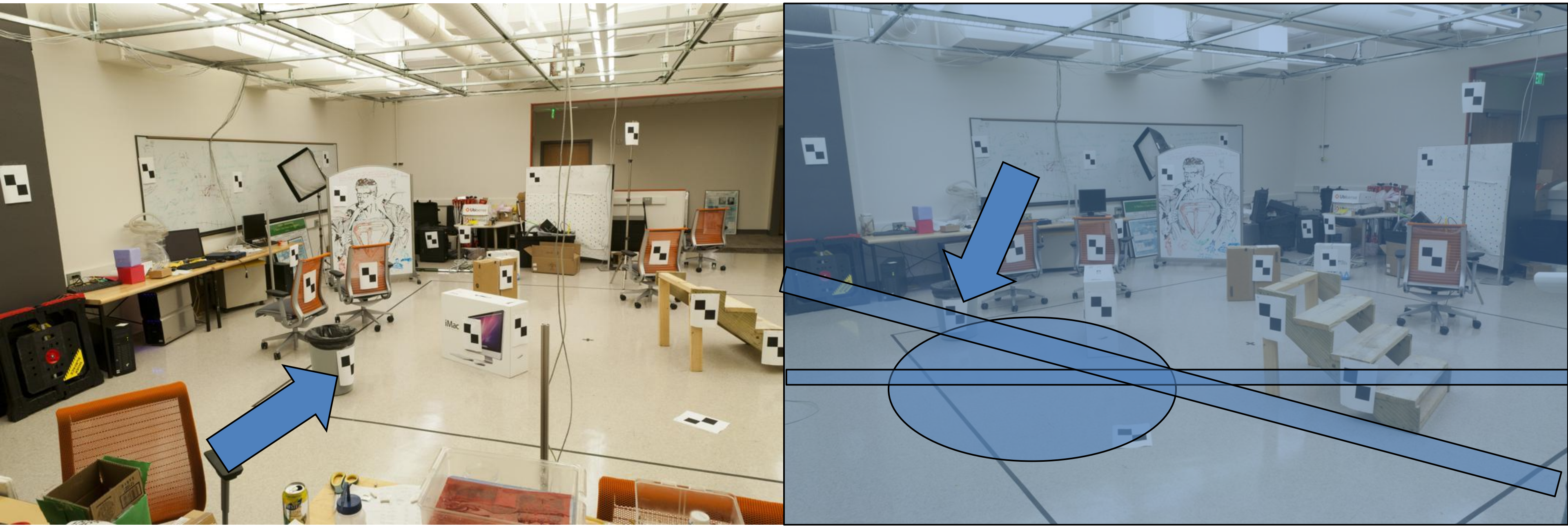
1. Calibration for the two cameras.
    1. Intrinsic matrices for both cameras (e.g., f)
    2. Baseline distance T in parallel camera case
    3. R, t in non-parallel case

2. Correspondence for every pixel.

    Like project 2, but project 2 is “sparse”.

    We need “dense” correspondence!

Correspondence for every pixel.
Where do we need to search?

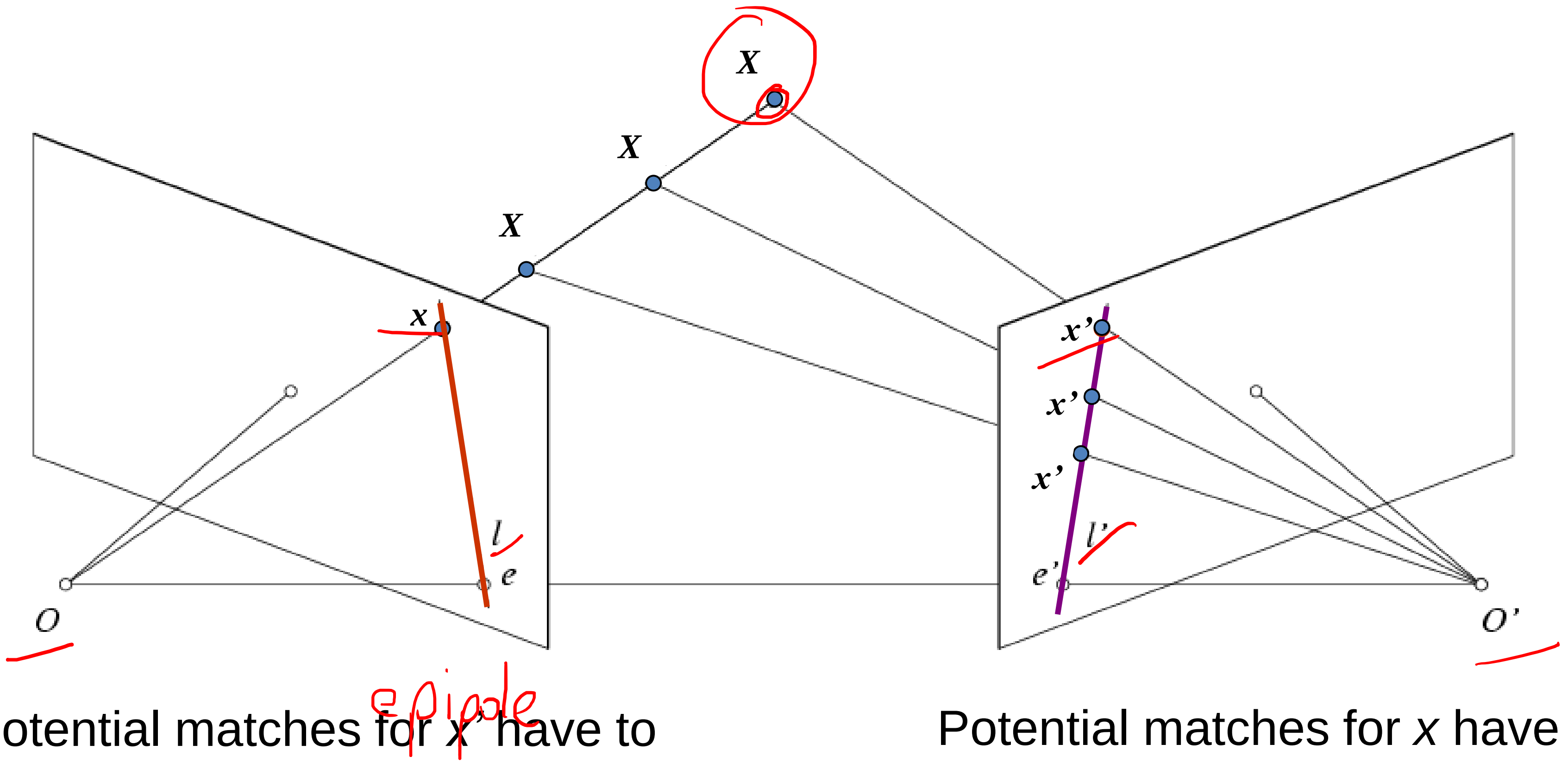


Wouldn’t it be nice to know where matches can live?

Epipolar geometry

Constrains 2D search to 1D

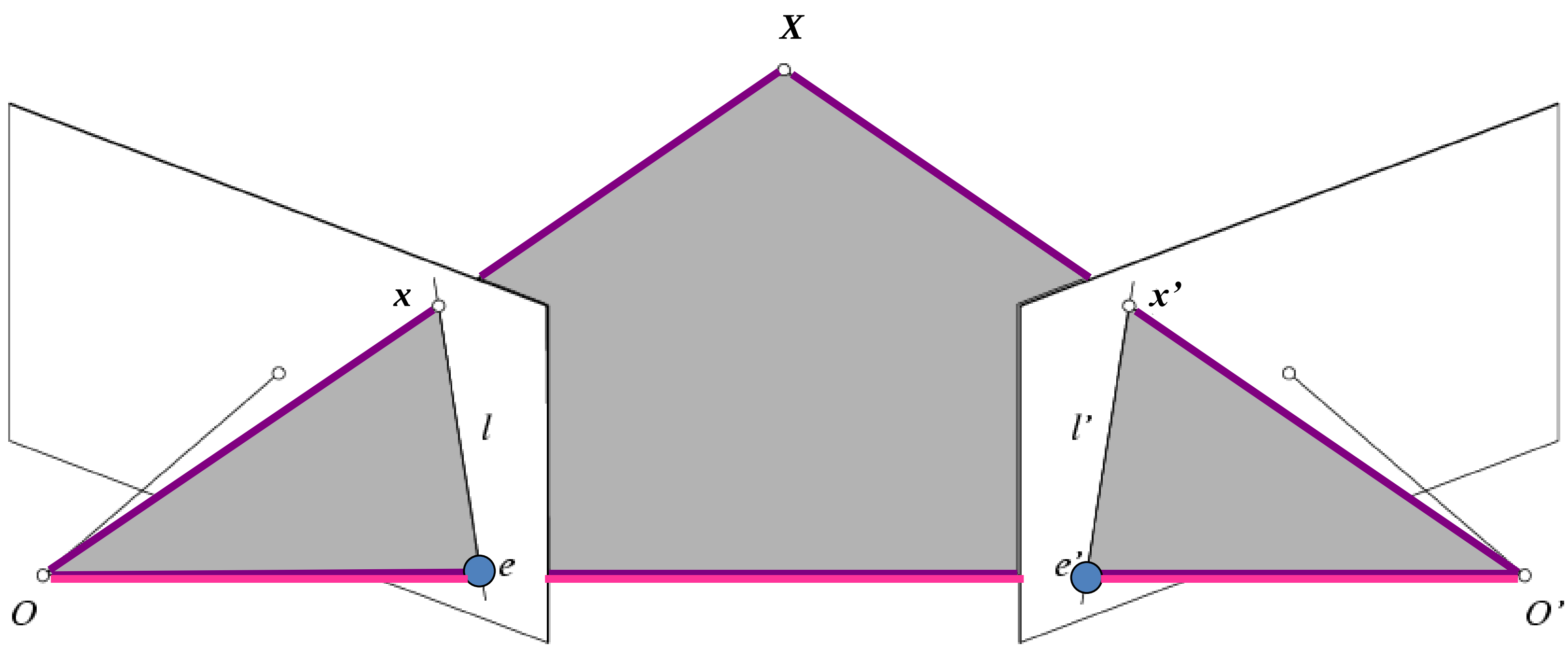
Key idea: Epipolar constraint



Potential matches for *x'* have to lie on the corresponding line *l*.

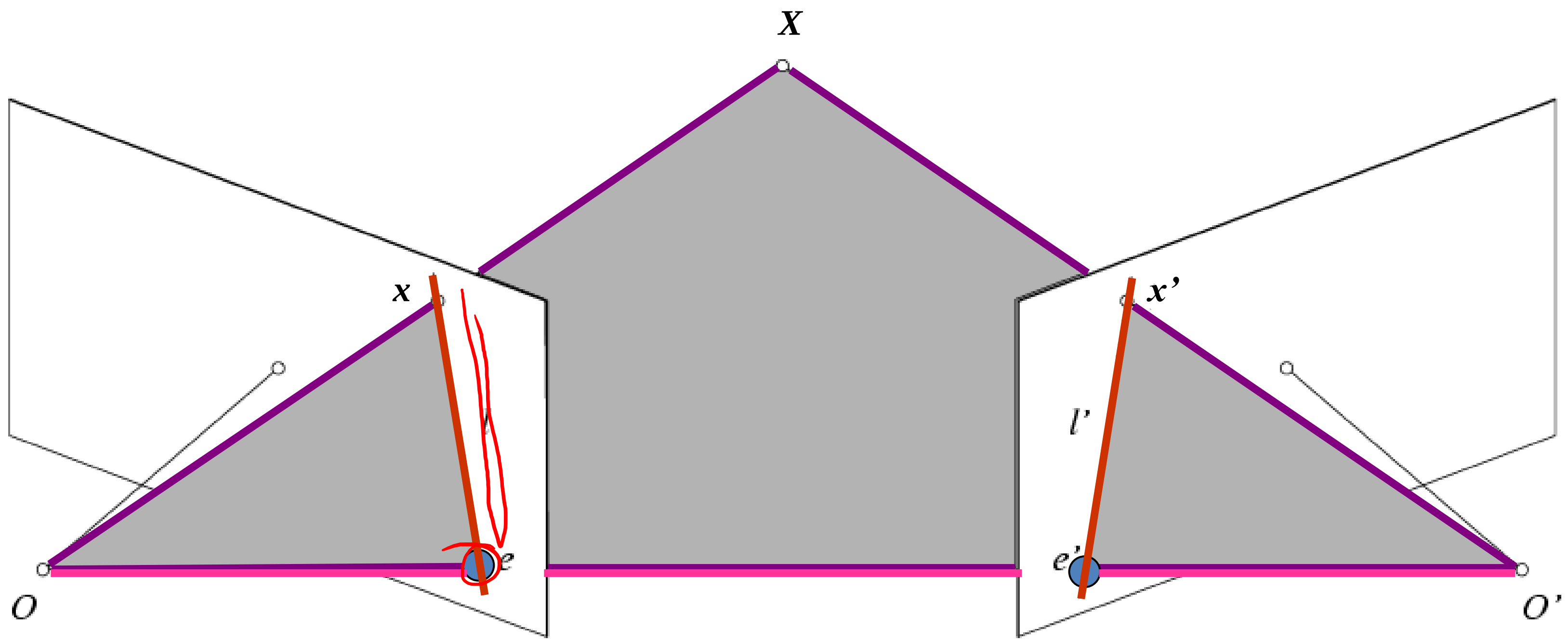
Potential matches for *x* have to lie on the corresponding line *l'*.

Epipolar geometry: notation



- **Baseline** – line connecting the two camera centers
- **Epipoles**
= intersections of baseline with image planes
= projections of the other camera center
- **Epipolar Plane** – plane containing baseline (1D family)

Epipolar geometry: notation

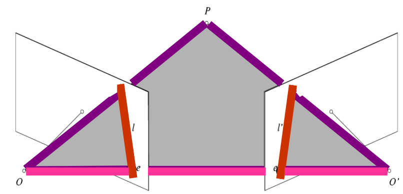


- **Baseline** – line connecting the two camera centers
- **Epipoles**

= intersections of baseline with image planes
= projections of the other camera center

- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)

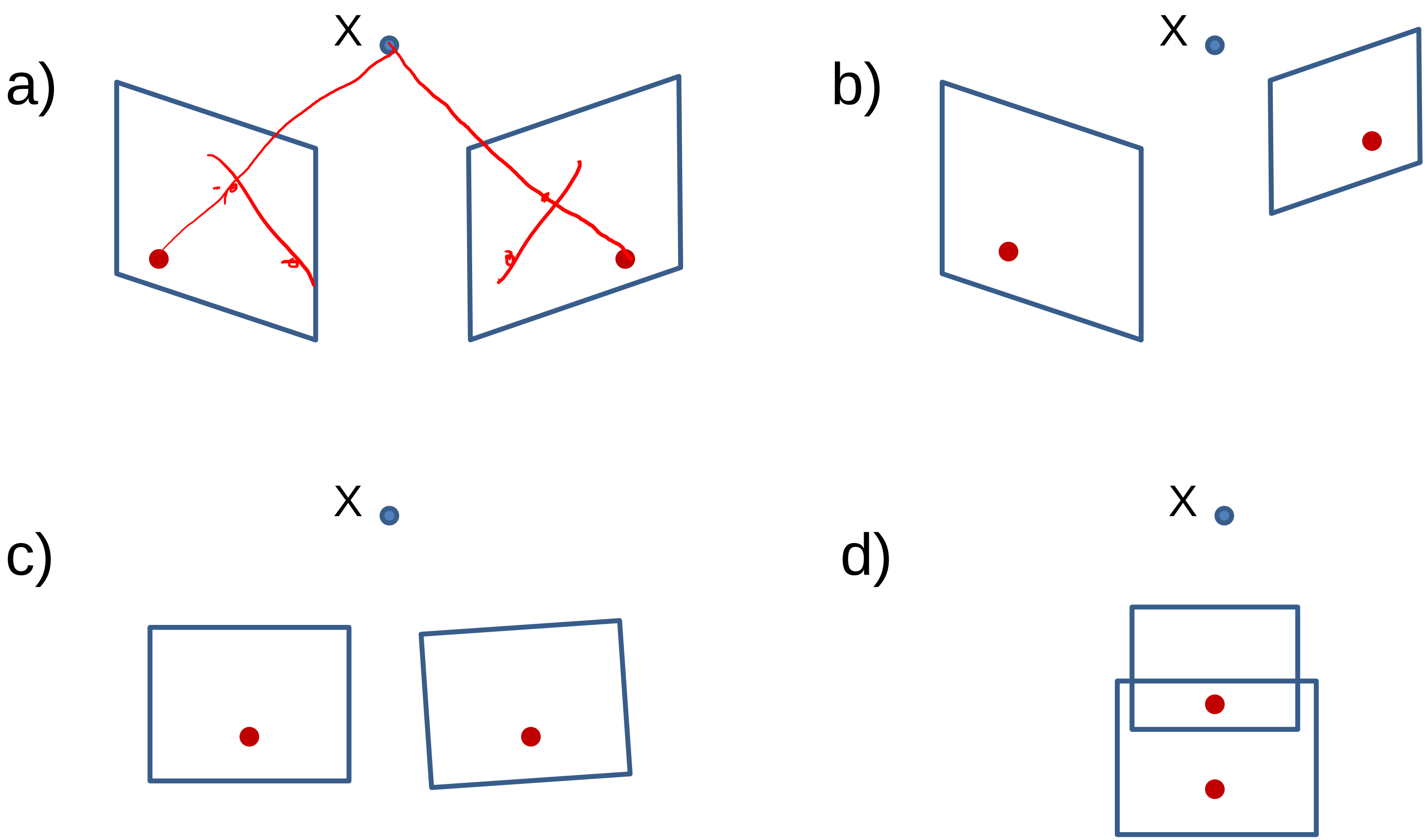
= camera center



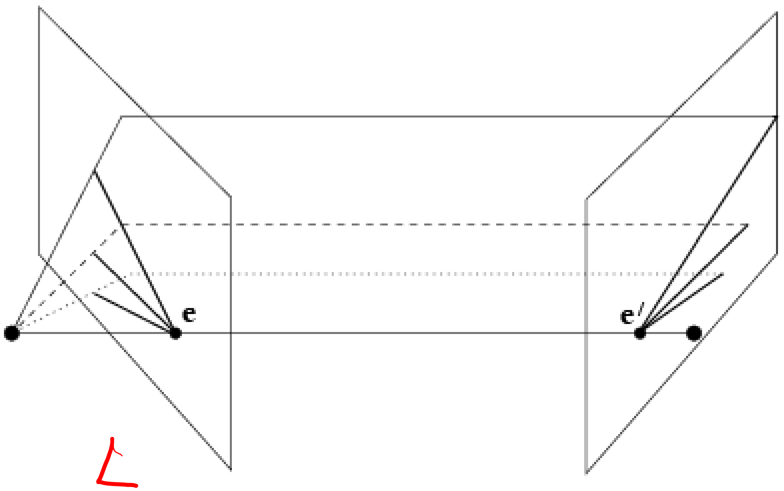
Think Pair Share

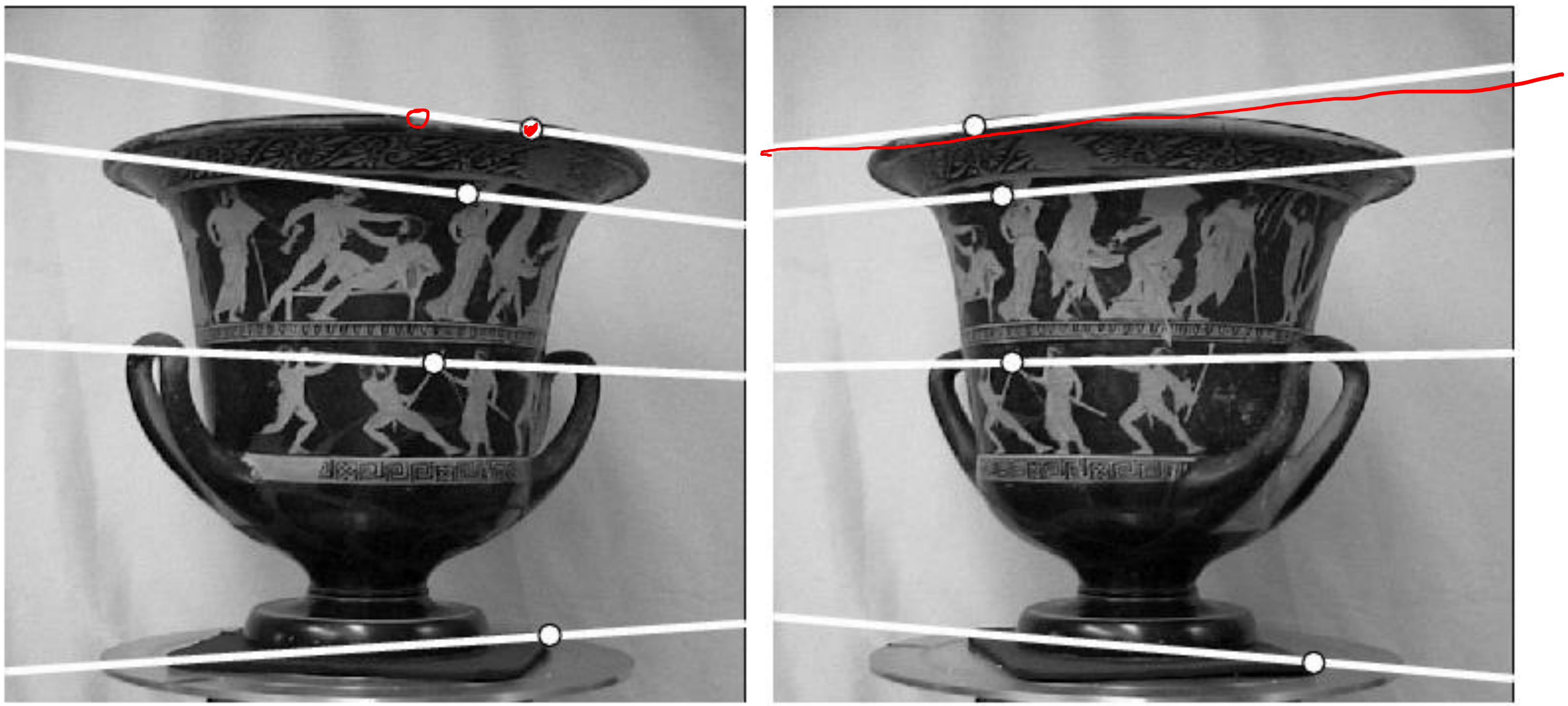
Where are the epipoles?

What do the epipolar lines look like?

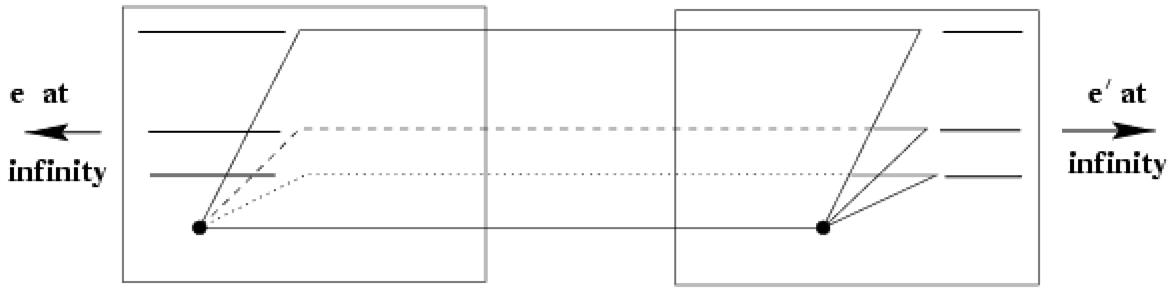


Example: Converging cameras

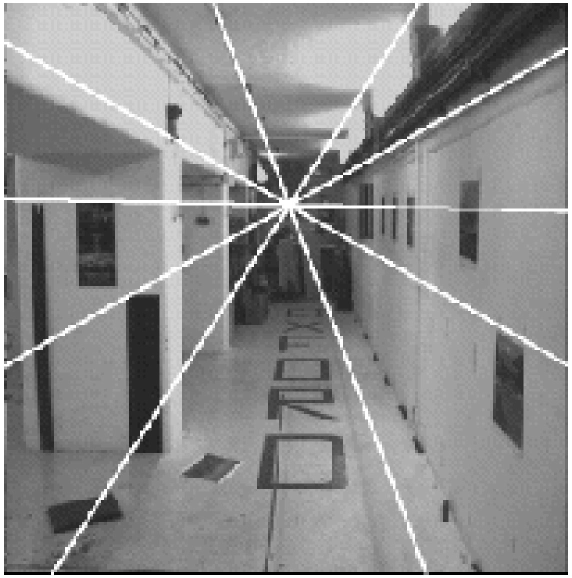


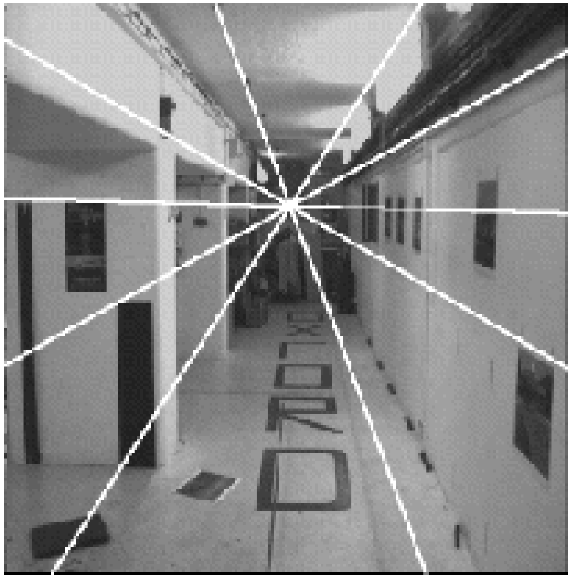


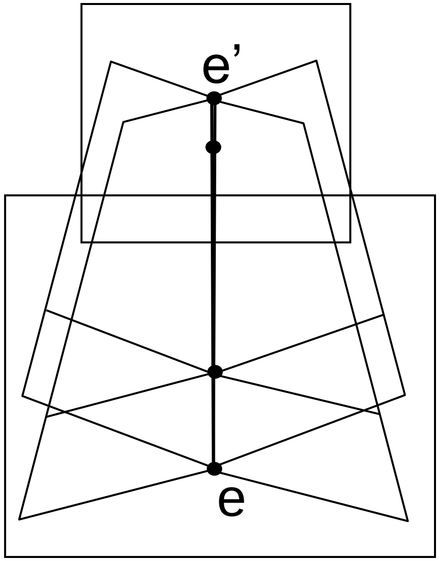
Example: Motion parallel to image plane



Example: Forward motion

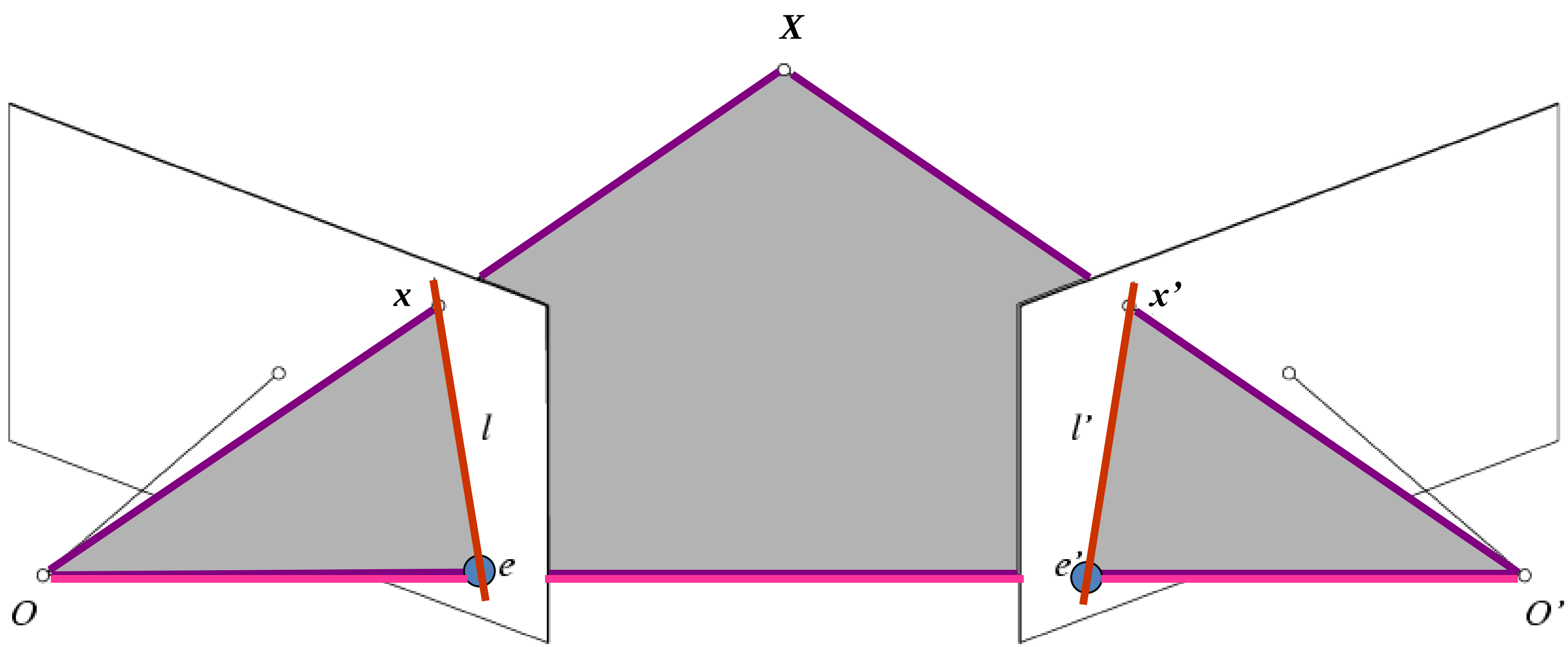






Epipole has same coordinates in both images.
Points move along lines radiating from e: "Focus of expansion"

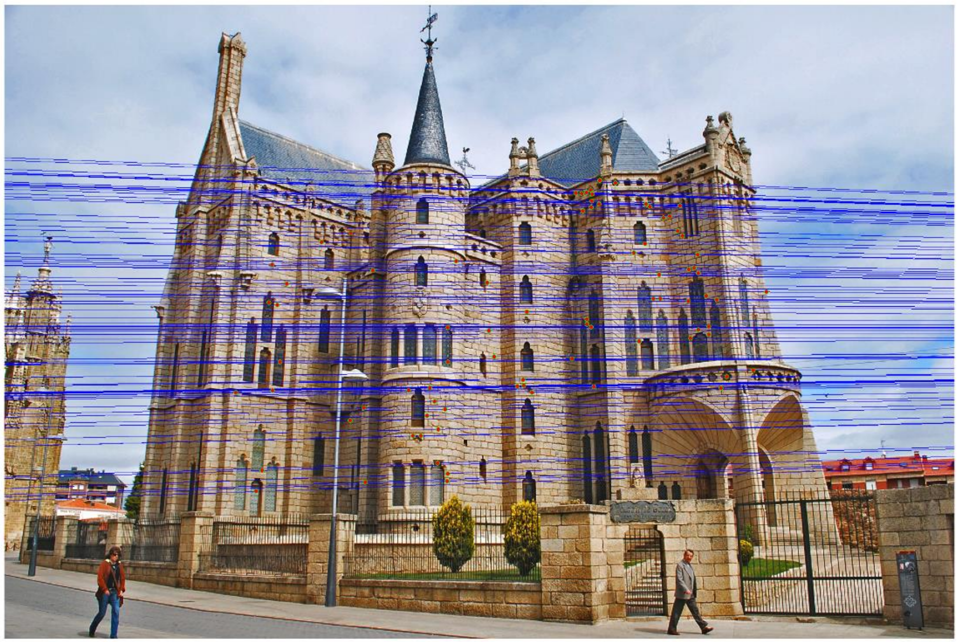
What is this useful for?

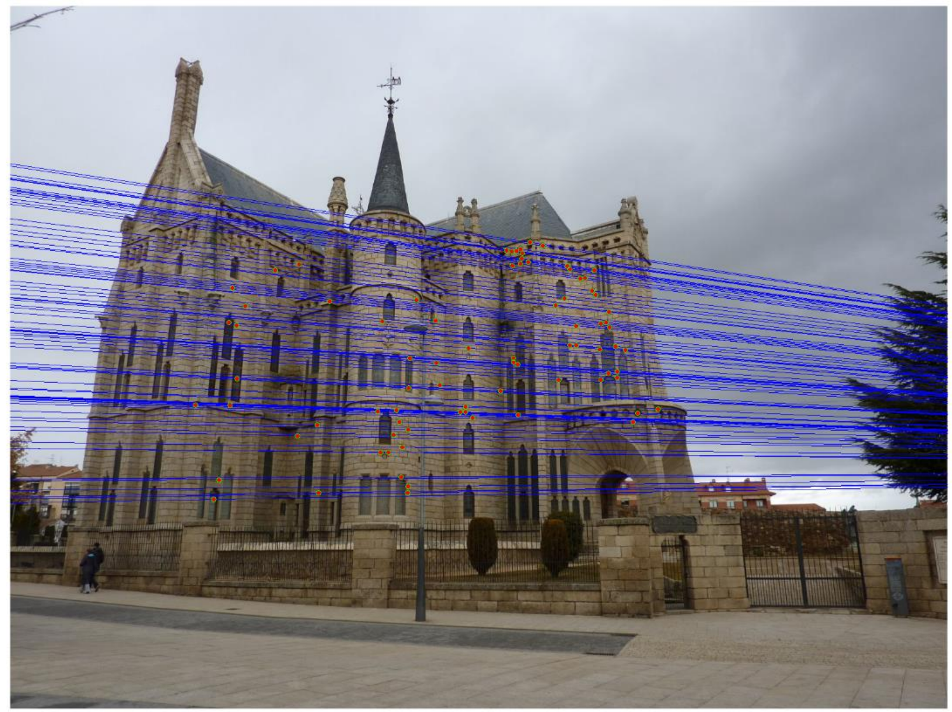


Reduce search space for stereo disparity estimation.

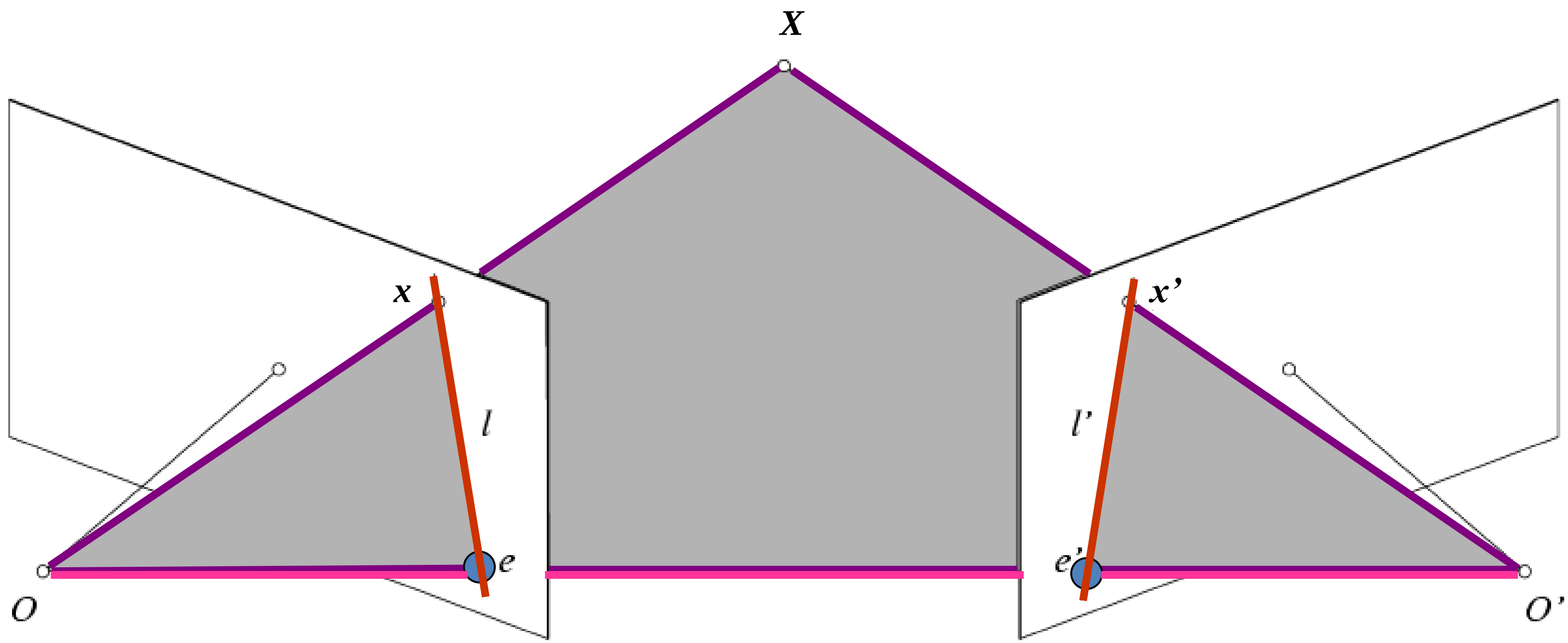
- Help find *x'*: If I know *x*, and have calibrated cameras (known intrinsics K,K' and extrinsic relationship), I can restrict *x'* to be along *l'*.

Epipolar lines



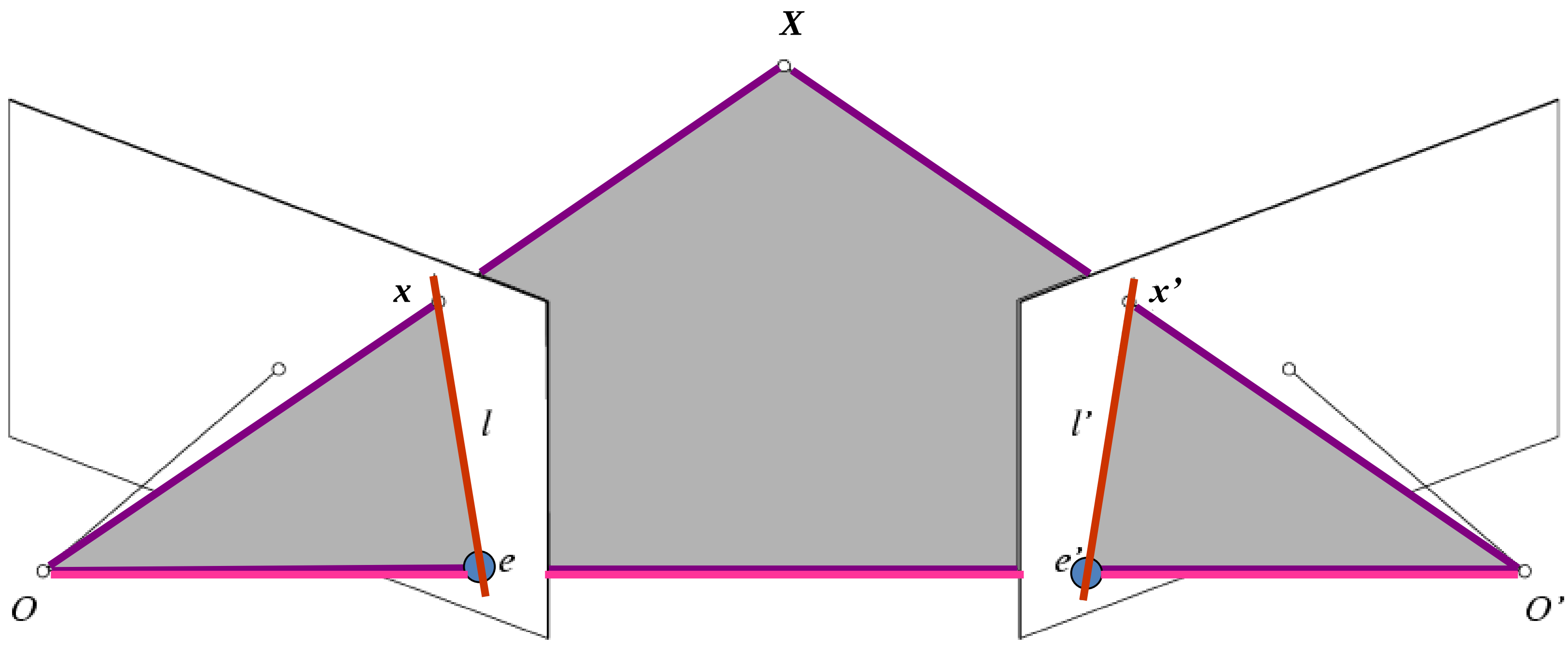


What is this useful for?



If we have enough *x, x’* correspondences, we can estimate relative position and orientation between the cameras and the 3D position of corresponding image points -> estimate *E*.

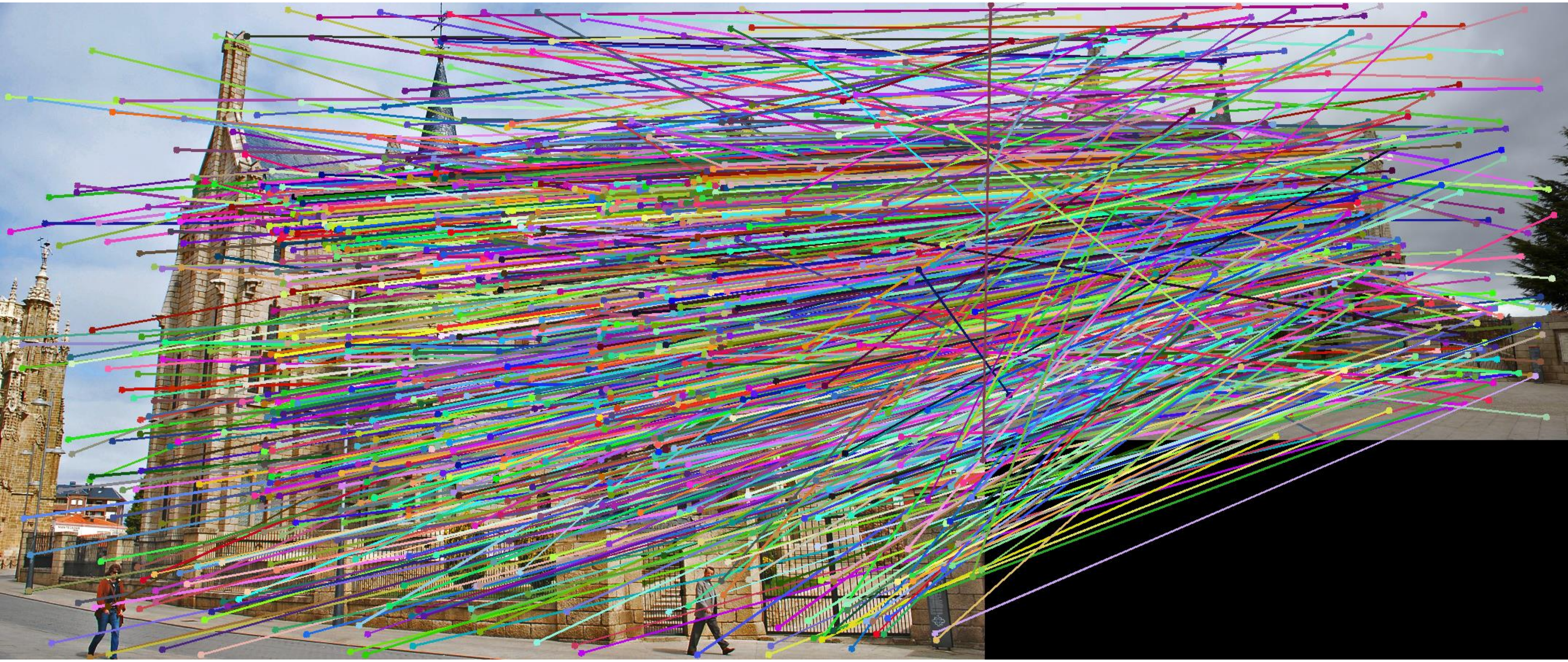
What is this useful for?



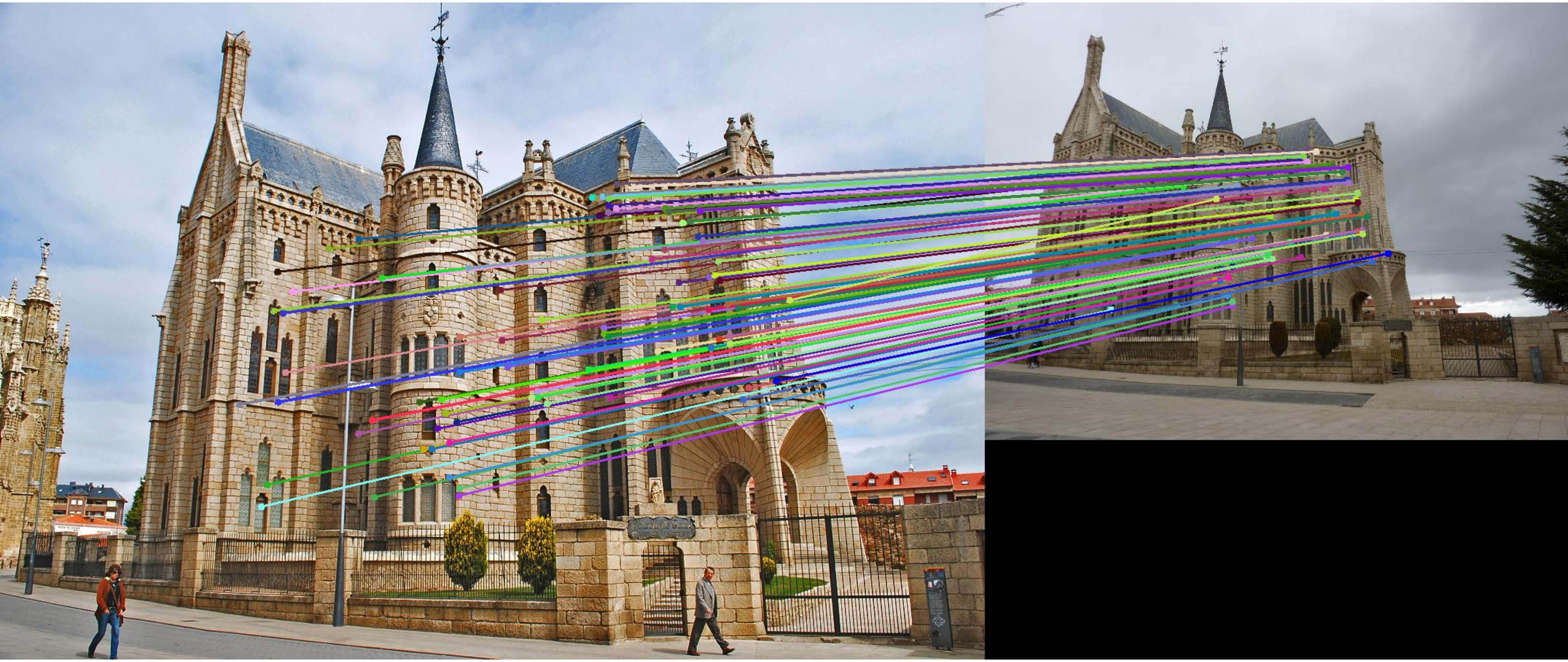
Camera model ‘sanity check’:

- See if candidate *x, x’* correspondences fit estimated projection models of cameras 1 and 2.

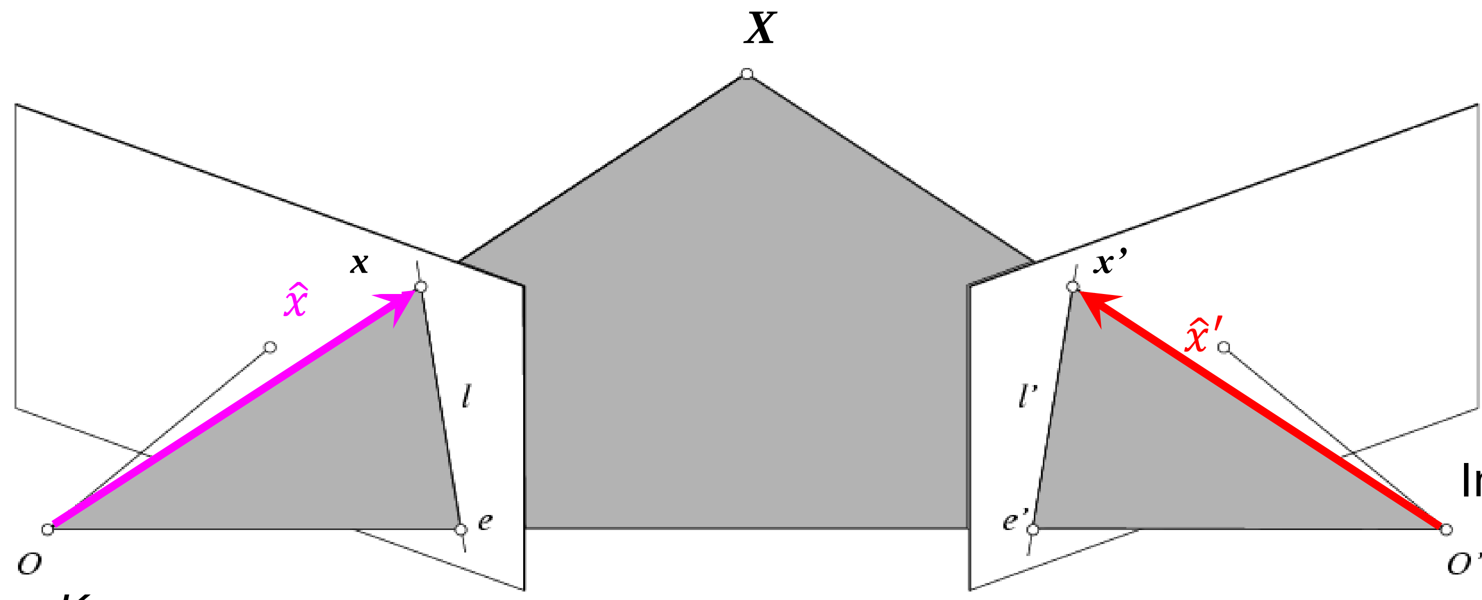
VLFeat’s 800 most confident matches among 10,000+ local features.



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



Epipolar constraint: Calibrated case



Intrinsics K

Intrinsics K'

$$\hat{x} = K^{-1} x = X$$

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

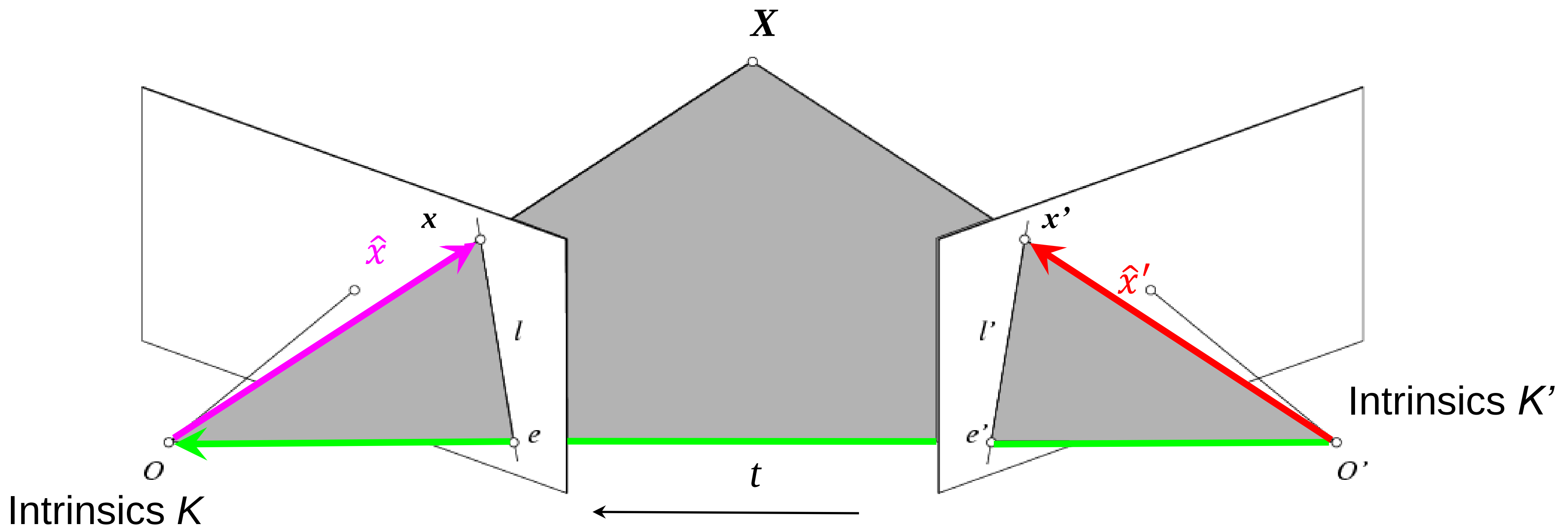
Homogeneous 2d point
(3D ray towards X)

2D pixel coordinate
(homogeneous)

3D scene point

3D scene point in 2nd
camera's 3D coordinates

Epipolar constraint: Calibrated case



$$\hat{x} = K^{-1} x = X$$

Homogeneous 2d point (3D ray towards X)

2D pixel coordinate (homogeneous)

3D scene point

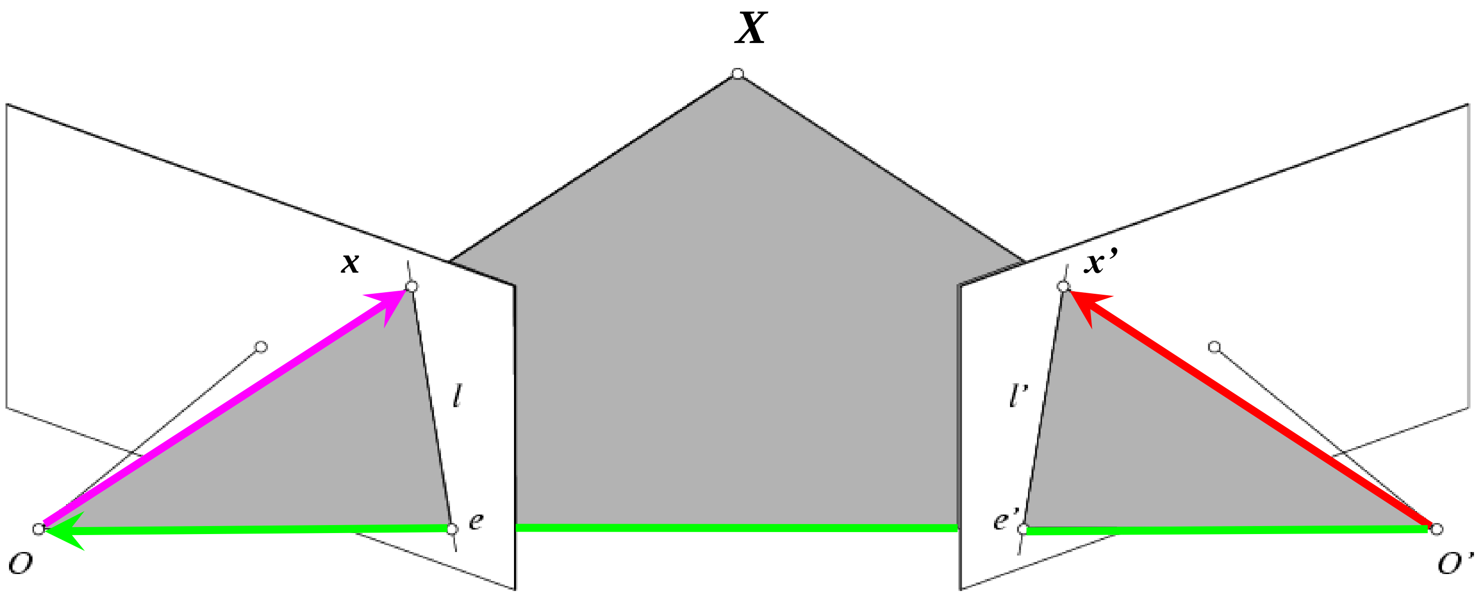
$$\hat{x}' = K'^{-1} x' = X'$$

3D scene point in 2nd camera's 3D coordinates

$$\hat{x} \cdot [t \times (R\hat{x}')] = 0$$

(because $\hat{x}, R\hat{x}'$, and t are co-planar)

Essential matrix



$$\hat{x} \cdot [t \times (R\hat{x}')] = 0 \quad \Rightarrow \quad \hat{x}^T E \hat{x}' = 0 \quad \text{with} \quad E = [t]_\times R$$

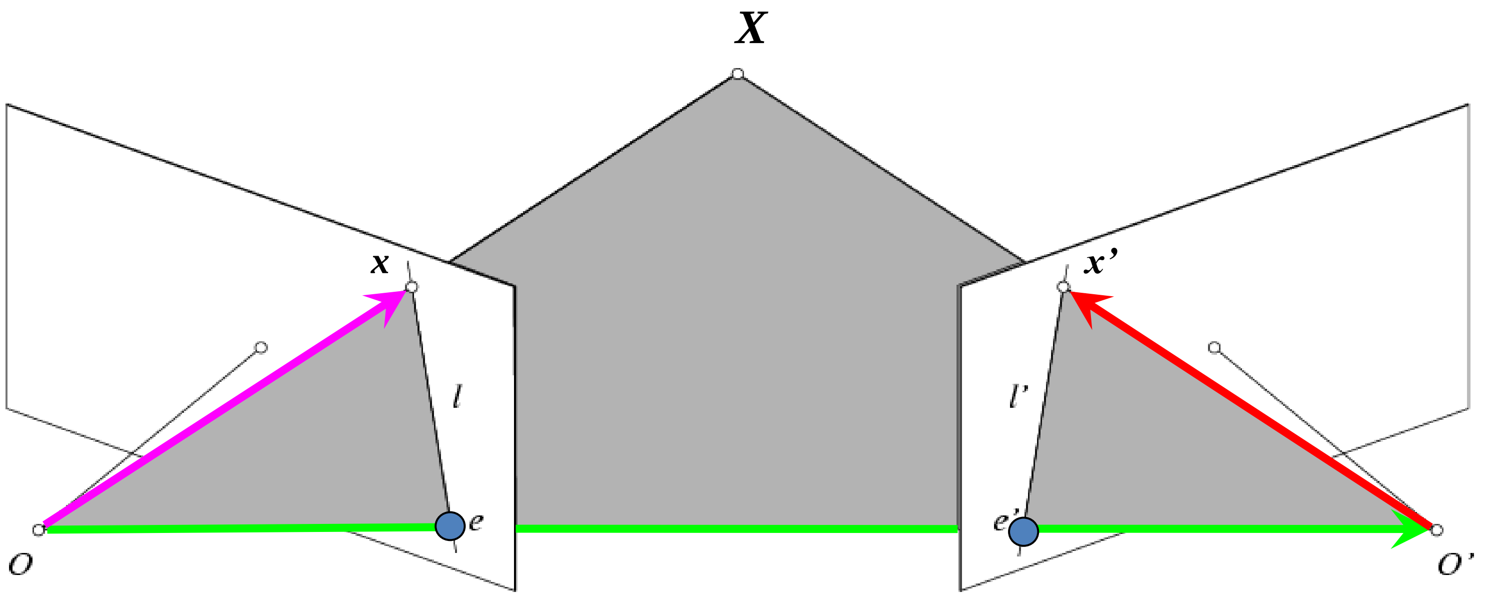
Essential Matrix
(Longuet-Higgins, 1981)

E is a 3x3 matrix which relates corresponding pairs of normalized homogeneous image points across pairs of images – for intrinsic *K* calibrated cameras.

Estimates relative position/orientation.

Note: $[t]_\times$ is matrix representation of cross product

Epipolar constraint: Uncalibrated case



If we don't know intrinsics *K* and *K'*, then we can write the epipolar constraint in terms of *unknown* normalized coordinates:

$$\hat{x}^T E \hat{x}' = 0 \qquad\qquad x = K\hat{x}, \quad x' = K'\hat{x}'$$

The Fundamental Matrix

Without knowing K and K', we can define a similar relation using *unknown* normalized coordinates

$$\hat{x}^T E \hat{x}' = 0$$

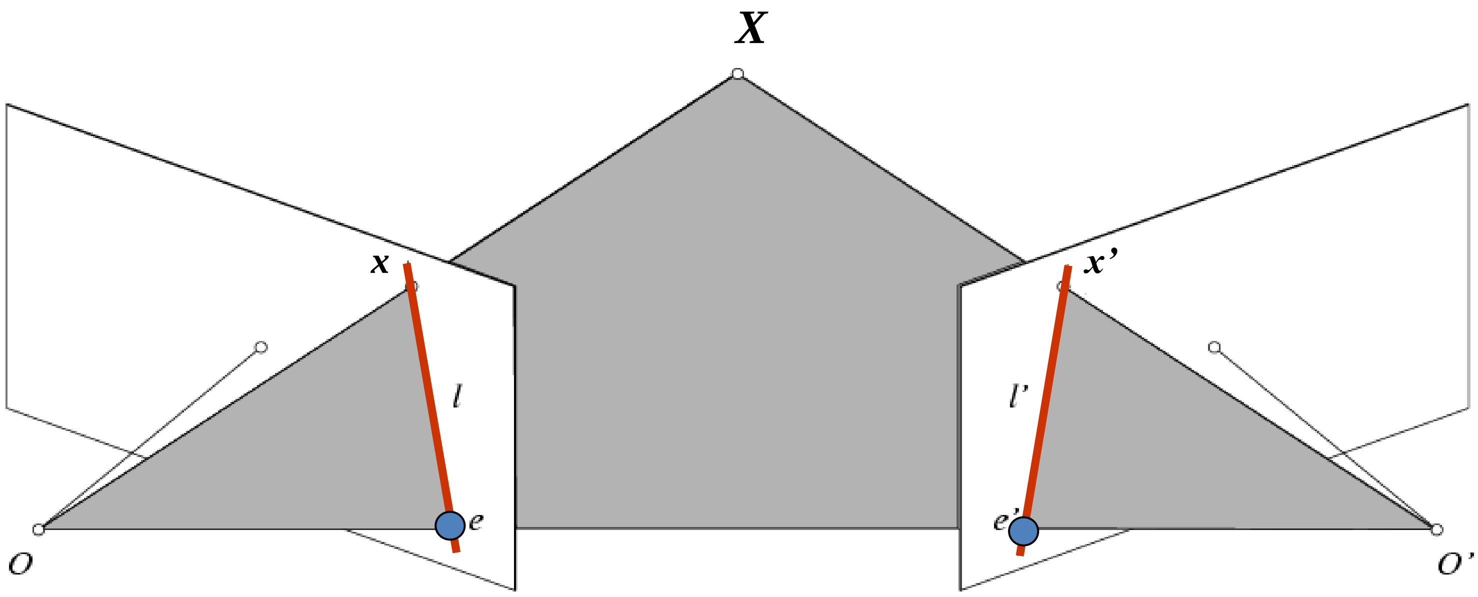
$$\hat{x} = K^{-1} x$$

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

$$x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

Fundamental Matrix
(Faugeras and Luong, 1992)

Properties of the Fundamental matrix



$$x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

- $F x' = 0$ is the epipolar line l associated with x'
- $F^T x = 0$ is the epipolar line l' associated with x

- F is singular (rank two): det(F)=0
- $F e' = 0$ and $F^T e = 0$ (nullspaces of F = e'; nullspace of F^T = e')
- F has seven degrees of freedom: 9 entries but defined up to scale, det(F)=0

F in more detail

- F is a 3x3 matrix
- Rank 2 -> projection; one column is a linear combination of the other two.
- Determined up to scale.
- 7 degrees of freedom

$$\begin{bmatrix} a & b & \alpha a + \ \beta b \\ c & d & \alpha c + \beta d \\ e & f & \alpha e + \beta f \end{bmatrix}$$ where *a* is scalar; e.g., can normalize out.

Given x projected from X into image 1, F constrains the projection of x' into image 2 to an epipolar line.

Estimating the Fundamental Matrix

- 8-point algorithm
  - Least squares solution using SVD on equations from 8 pairs of correspondences
  - Enforce det(F)=0 constraint using SVD on F

Note: estimation of F (or E) is degenerate for a planar scene.

8-point algorithm

1. Solve a system of homogeneous linear equations
   a. Write down the system of equations

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

$$uu'f_{11} + uv'f_{12} + uf_{13} + vu'f_{21} + vv'f_{22} + vf_{23} + u'f_{31} + v'f_{32} + f_{33} = 0$$

$$\mathrm{A}\boldsymbol{f} = \begin{bmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_v' & u_nv_n' & u_n & v_nu_n' & v_nv_n' & v_n & u_n' & v_n' & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ \vdots \\ f_{33} \end{bmatrix} = \mathbf{0}$$

8-point algorithm

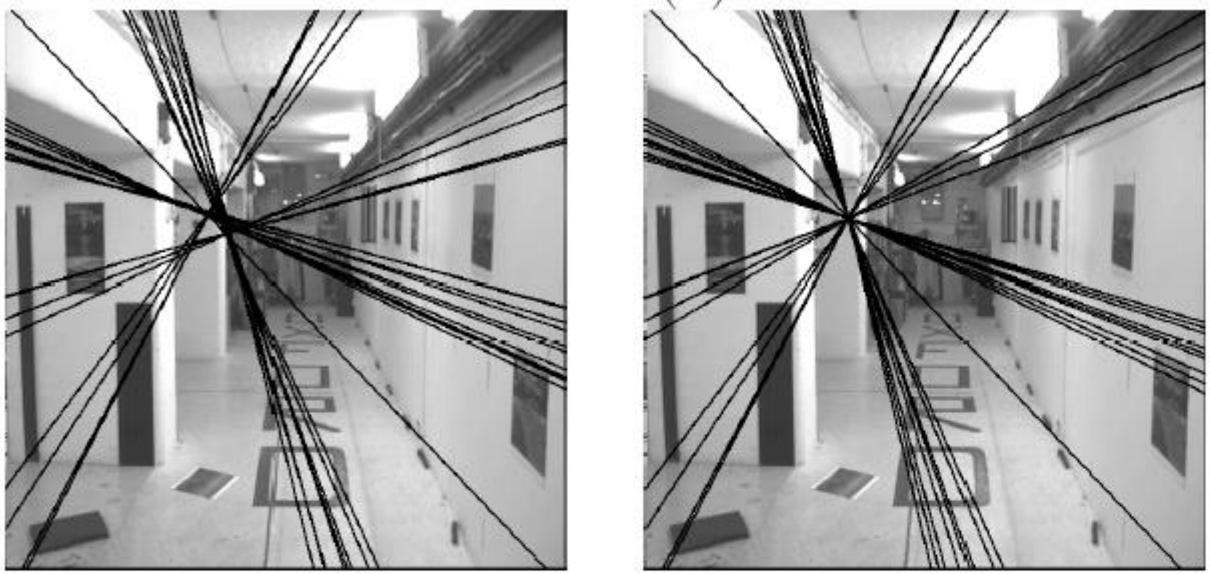
1. Solve a system of homogeneous linear equations
    a. Write down the system of equations
    b. Solve **f** from A**f**=**0** using SVD

Matlab:
```
[U, S, V] = svd(A);
f = V(:, end);
F = reshape(f, [3 3])';
```

```
Python Numpy:
U, S, Vh = np.linalg.svd(A)
# V = Vh.T -> note = different from MATLAB
F = Vh[-1,:]
F = np.reshape(F, (3,3))
```

Need to enforce singularity constraint

Fundamental matrix has rank 2 : $\det(\mathtt{F}) = 0$.



Left : Uncorrected $\mathtt{F}$ – epipolar lines are not coincident.

Right : Epipolar lines from corrected $\mathtt{F}$.

8-point algorithm

1. Solve a system of homogeneous linear equations
   a. Write down the system of equations
   b. Solve **f** from A**f**=**0** using SVD

Matlab:

```
[U, S, V] = svd(A);
f = V(:, end);
F = reshape(f, [3 3])';
```

Python Numpy:

```
U, S, Vh = np.linalg.svd(A)
F = Vh[-1,:]
F = np.reshape(F, (3,3))
```

2. Resolve det(F) = 0 constraint using SVD

Matlab:

```
[U, S, V] = svd(F);
S(3,3) = 0;
F = U*S*V';
```

Python Numpy:

```
U, S, Vh = np.linalg.svd(F)
S[-1] = 0
F = U @ np.diagflat(S) @ Vh
```

@ operator = matrix multiplication

Problem with eight-point algorithm

$$\begin{bmatrix} u'u & u'v & u' & v'u & v'v & v' & u & v \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \end{bmatrix} = -1$$

Problem with eight-point algorithm

250906.36	183269.57	921.81	200931.10	146766.13	738.21	272.19	198.81
2692.28	131633.03	176.27	6196.73	302975.59	405.71	15.27	746.79
416374.23	871684.30	935.47	408110.89	854384.92	916.90	445.10	931.81
191183.60	171759.40	410.27	416435.62	374125.90	893.65	465.99	418.65
48988.86	30401.76	57.89	298604.57	185309.58	352.87	846.22	525.15
164786.04	546559.67	813.17	1998.37	6628.15	9.86	202.65	672.14
116407.01	2727.75	138.89	169941.27	3982.21	202.77	838.12	19.64
135384.58	75411.13	198.72	411350.03	229127.78	603.79	681.28	379.48

$$\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \end{bmatrix} = -1$$

Poor numerical conditioning

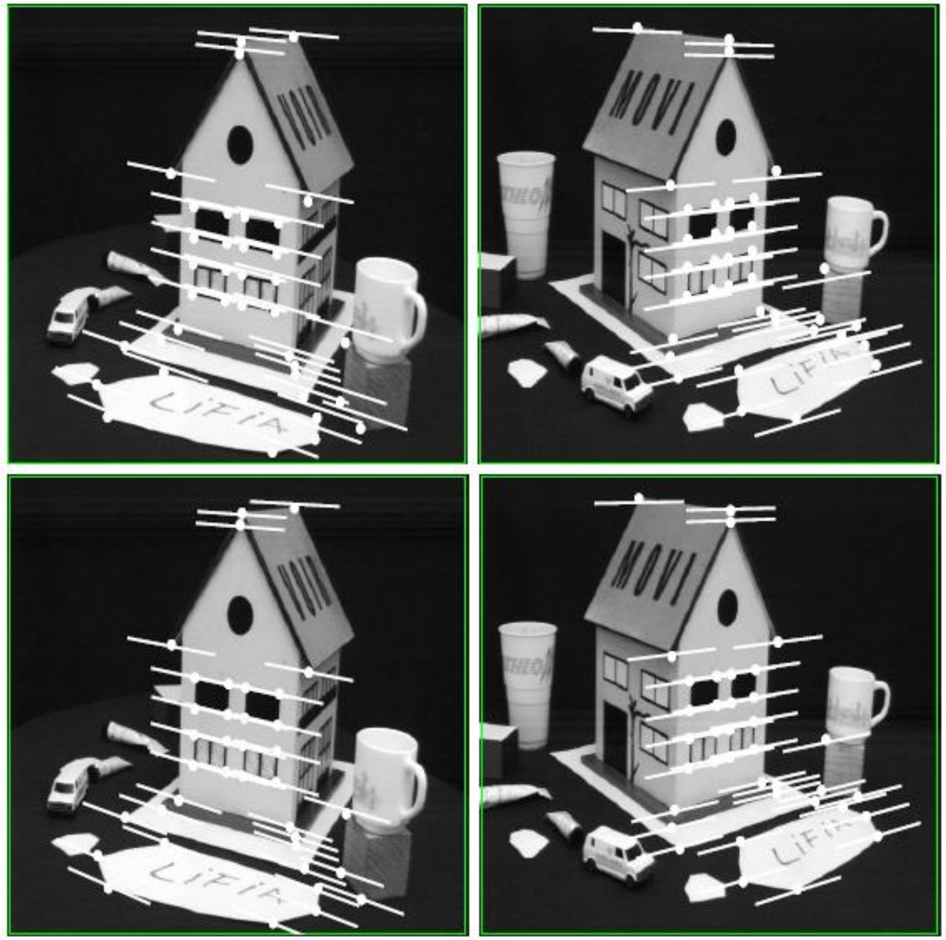
Can be fixed by rescaling the data

The normalized eight-point algorithm

(Hartley, 1995)

- Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels
- Use the eight-point algorithm to compute $\boldsymbol{F}$ from the normalized points
- Enforce the rank-2 constraint (for example, take SVD of $\boldsymbol{F}$ and throw out the smallest singular value)
- Transform fundamental matrix back to original units: if $\boldsymbol{T}$ and $\boldsymbol{T'}$ are the normalizing transformations in the two images, than the fundamental matrix in original coordinates is $\boldsymbol{T'}^T \boldsymbol{F} \boldsymbol{T}$

Comparison of estimation algorithms



	8-point	Normalized 8-point	Nonlinear least squares
Av. Dist. 1	2.33 pixels	0.92 pixel	0.86 pixel
Av. Dist. 2	2.18 pixels	0.85 pixel	0.80 pixel

From epipolar geometry to 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"

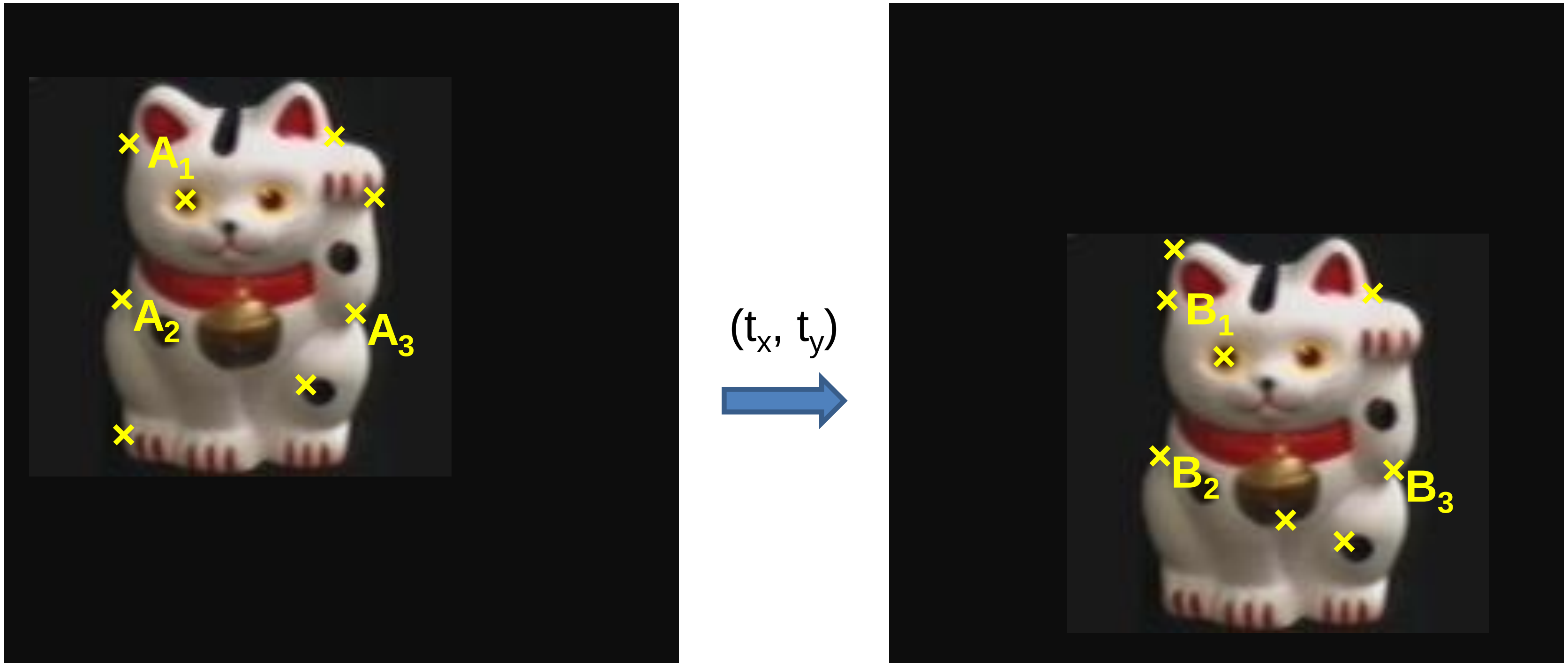
Let’s recap…

- Fundamental matrix song

- http://danielwedge.com/fmatrix/

Among all my matches, how do I know which ones are good?

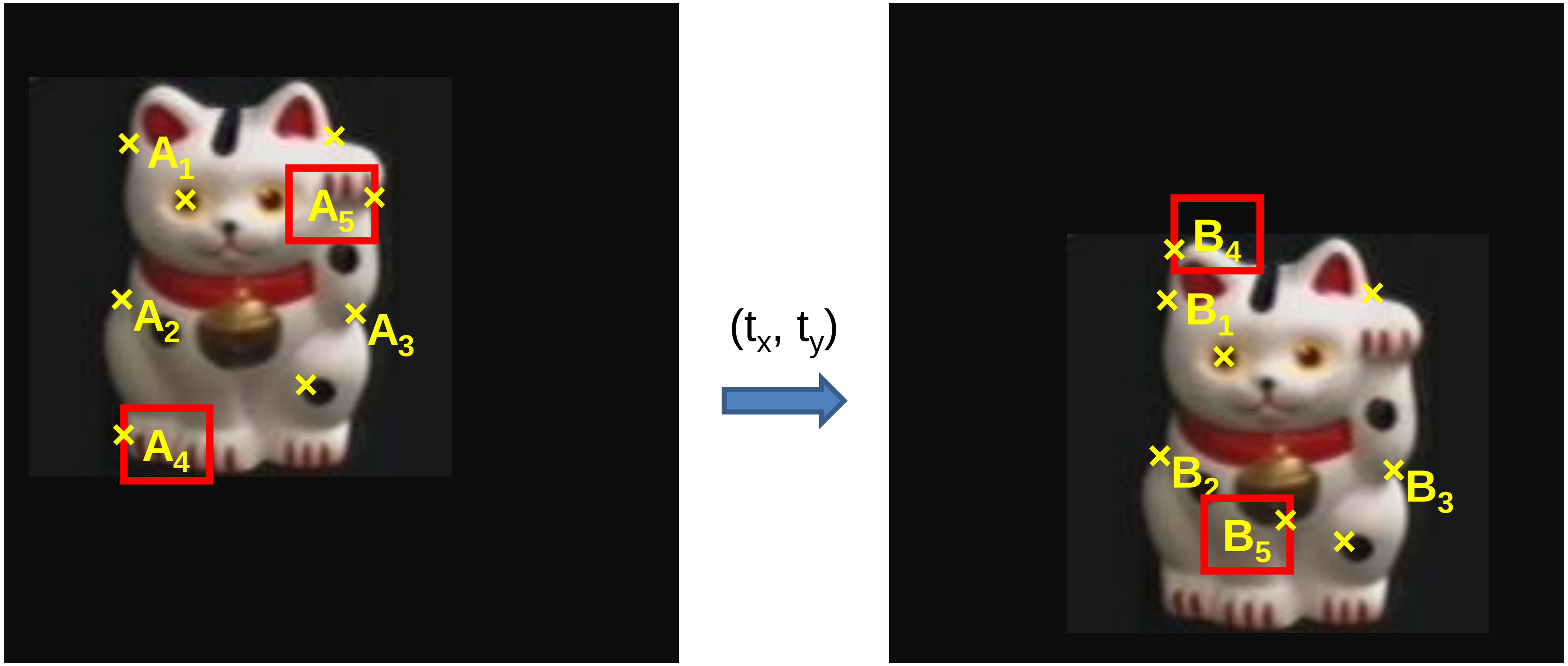
Example: solving for translation



Given matched points in {A} and {B}, estimate the translation of the object

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

Example: solving for translation

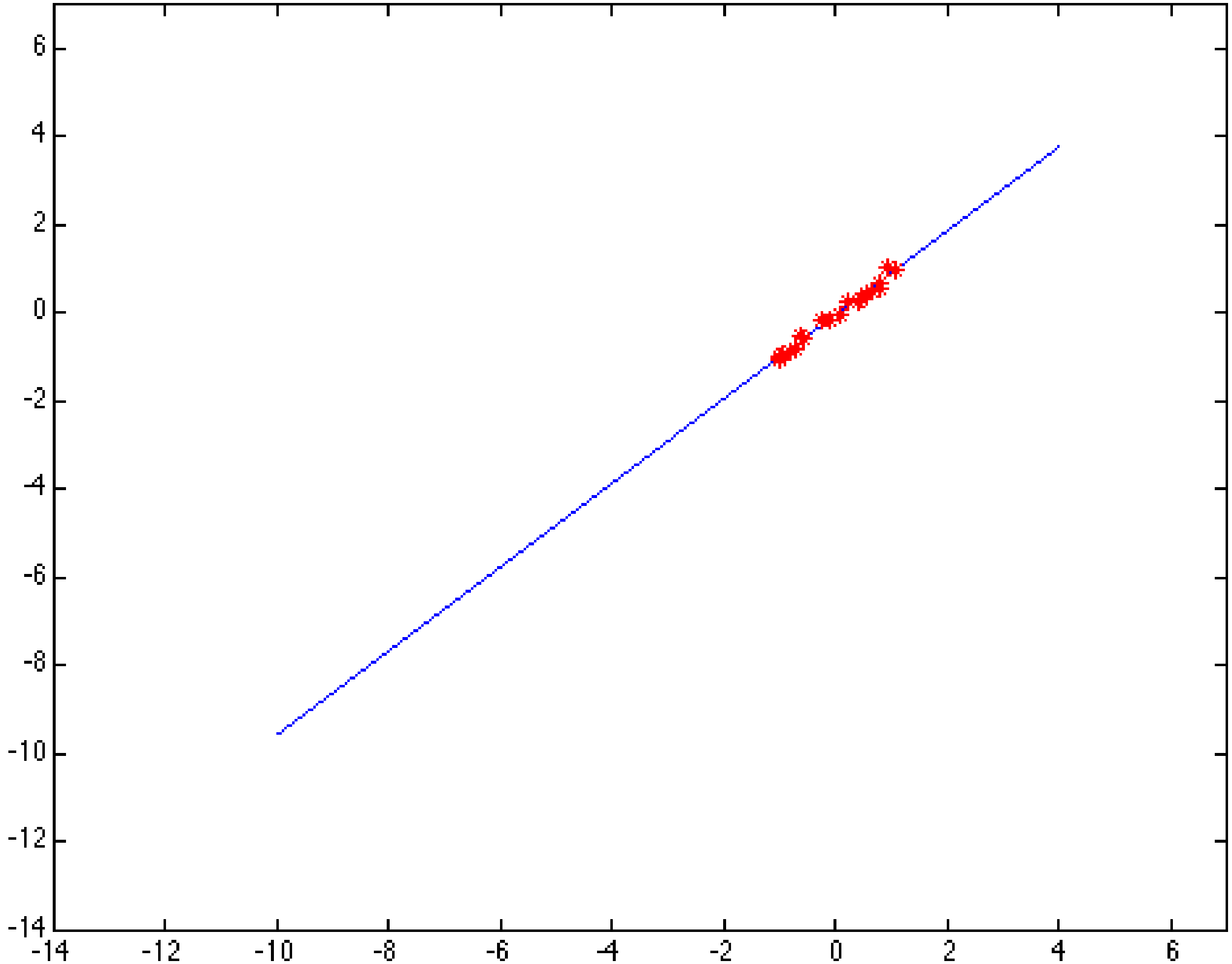


Problem: outliers A_4-B_4 and A_5-B_5 which *incorrectly* correspond

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

Least squares: Robustness to noise

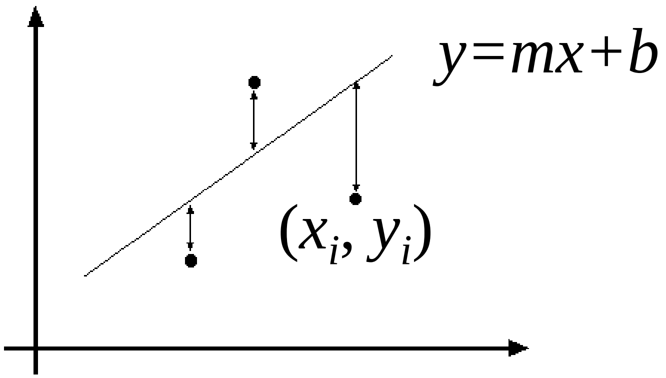
- Least squares fit to the red points:



Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m\,x_i + b$
- Find (m, b) to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$



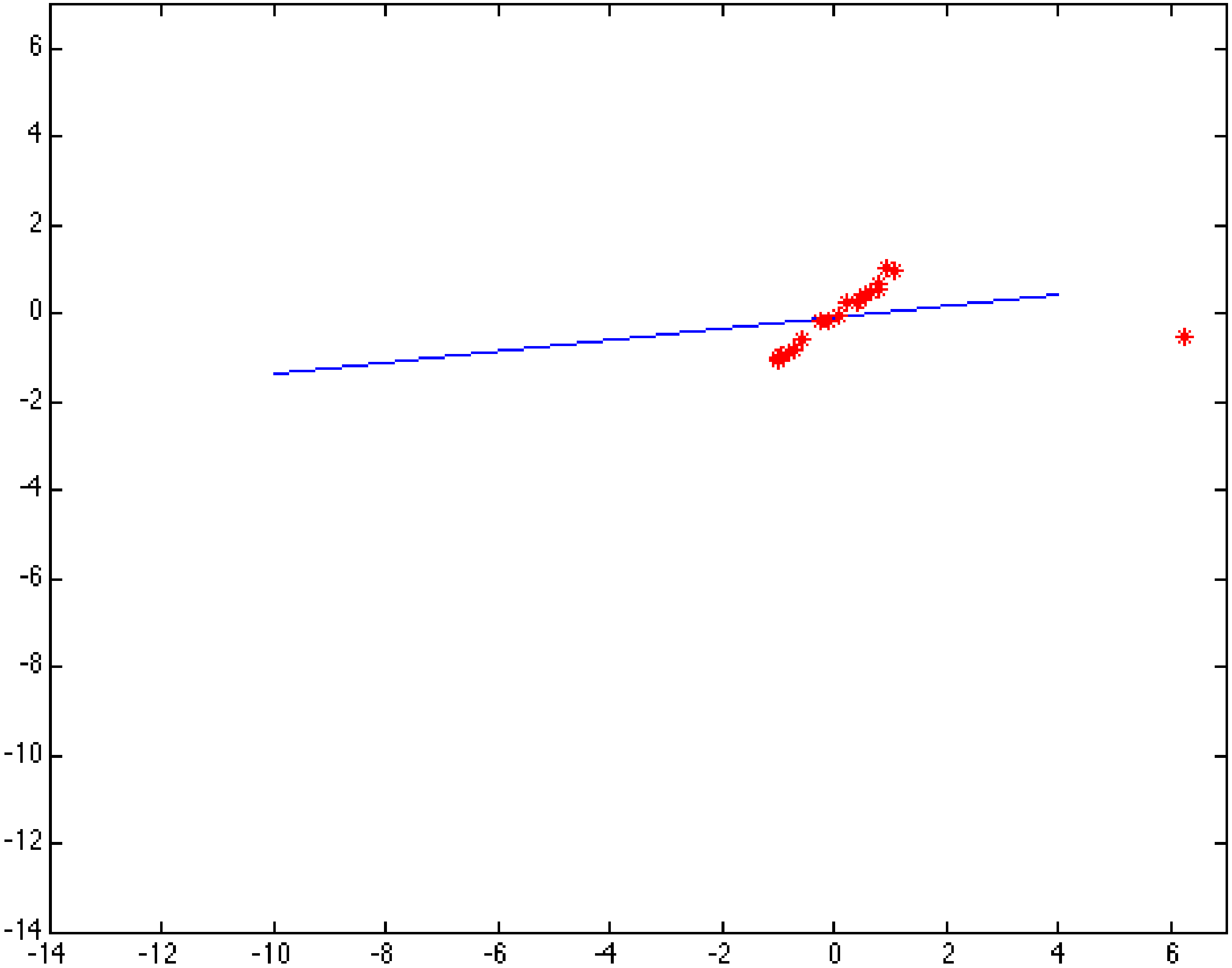
Matlab: `p = A \ y;`
Python: `p = np.linalg.lstsq(A,y)[0]`

(Closed form solution)

Modified from S. Lazebnik

Least squares: Robustness to noise

- Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

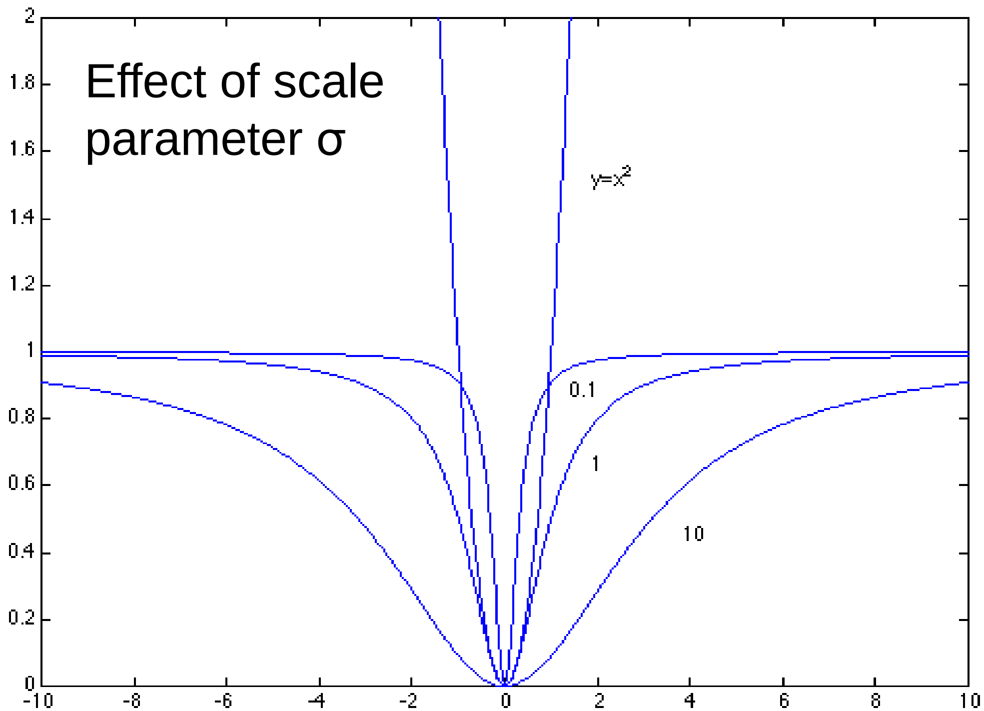
Robust least squares (to deal with outliers)

General approach:

minimize
$$\sum_i \rho\left(u_i\left(x_i, \theta\right); \sigma\right) \qquad u^2 = \sum_{i=1}^{n} (y_i - m x_i - b)^2$$

$u_i(x_i, \theta)$ – residual of i[th] point w.r.t. model parameters θ

ρ – robust function with scale parameter σ



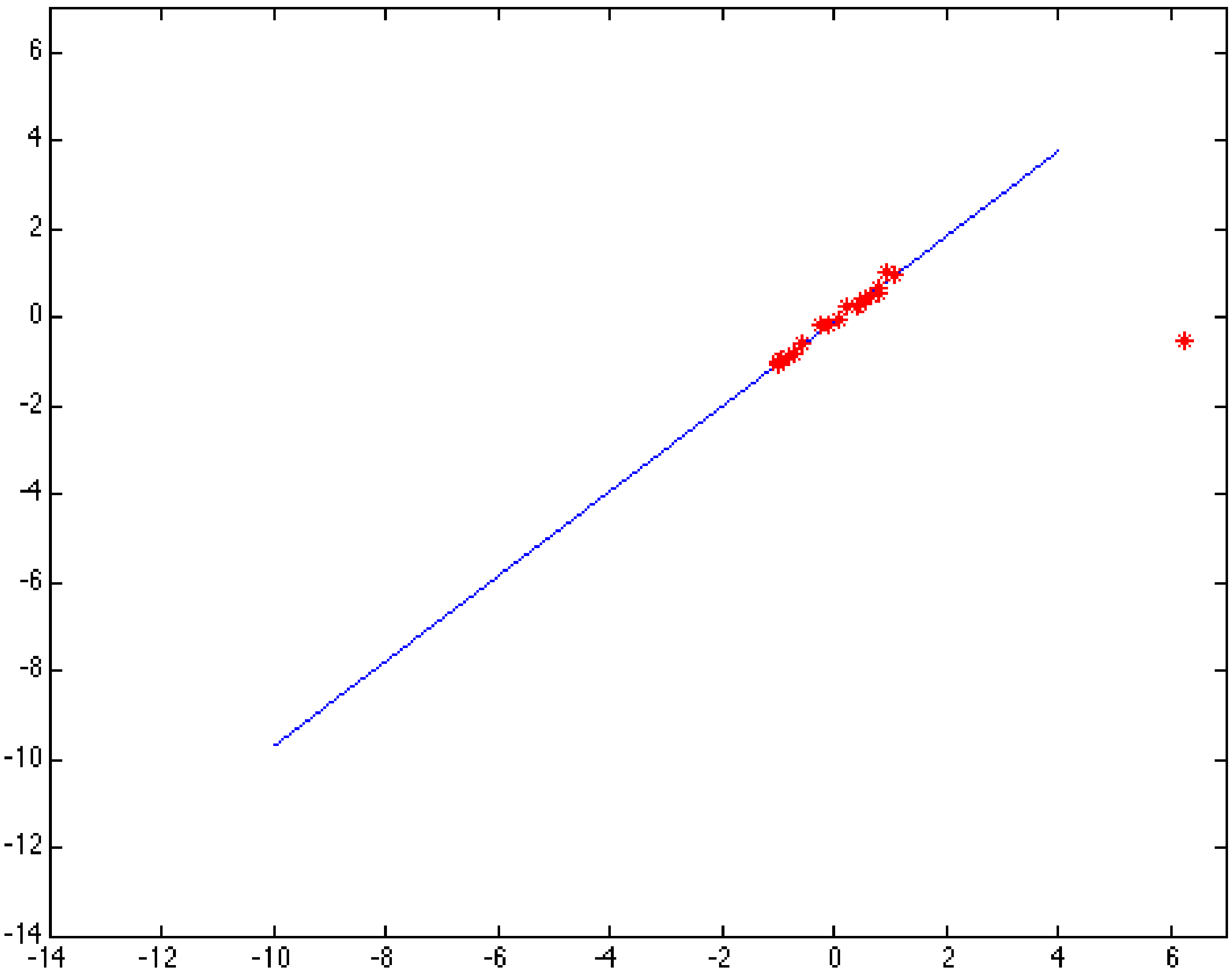
The robust function ρ

- Favors a configuration with small residuals
- Constant penalty for large residuals

$$\boxed{\rho(u; \sigma) = \frac{u^2}{\sigma^2 + u^2}}$$

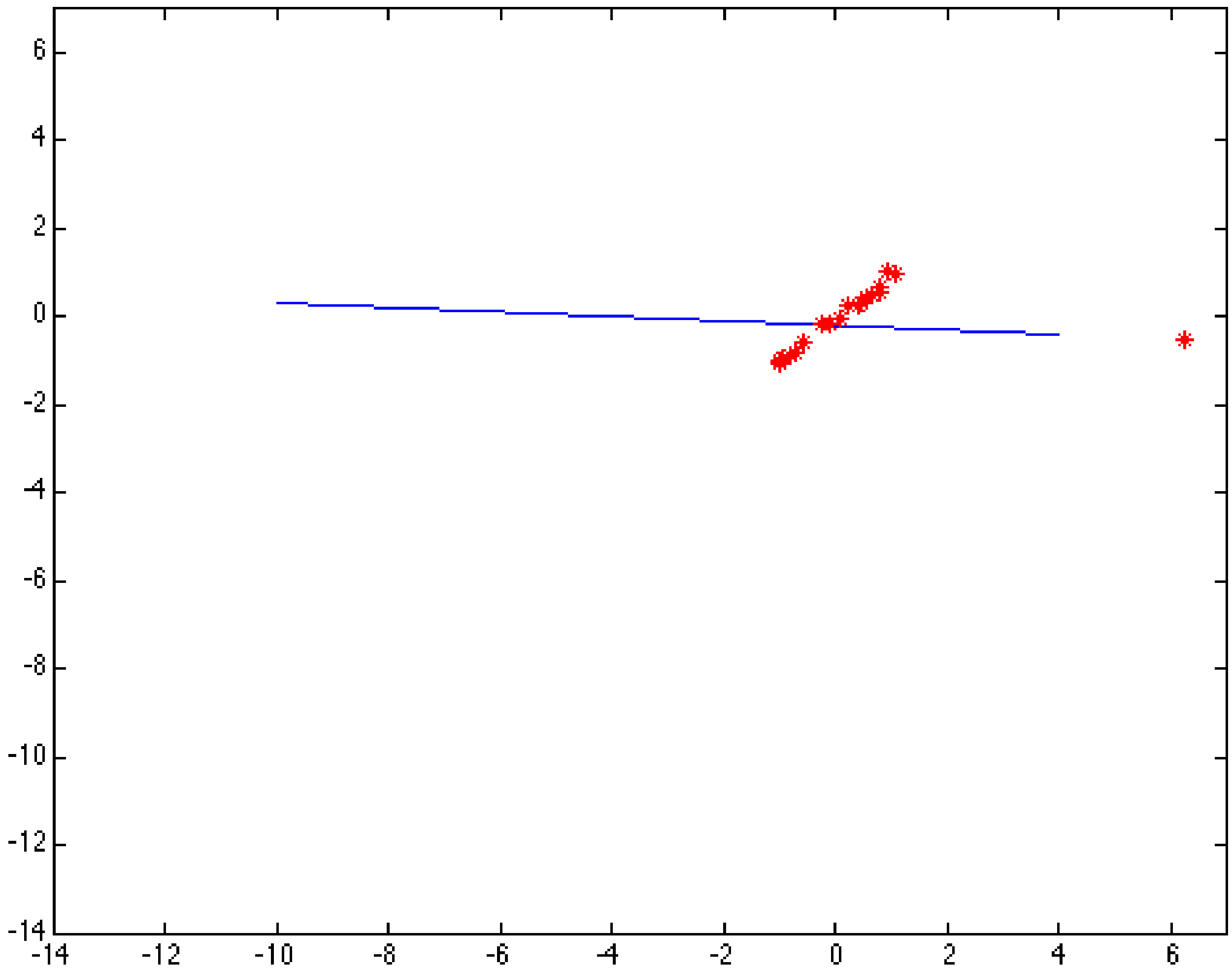
Slide from S. Savarese

Choosing the scale: Just right



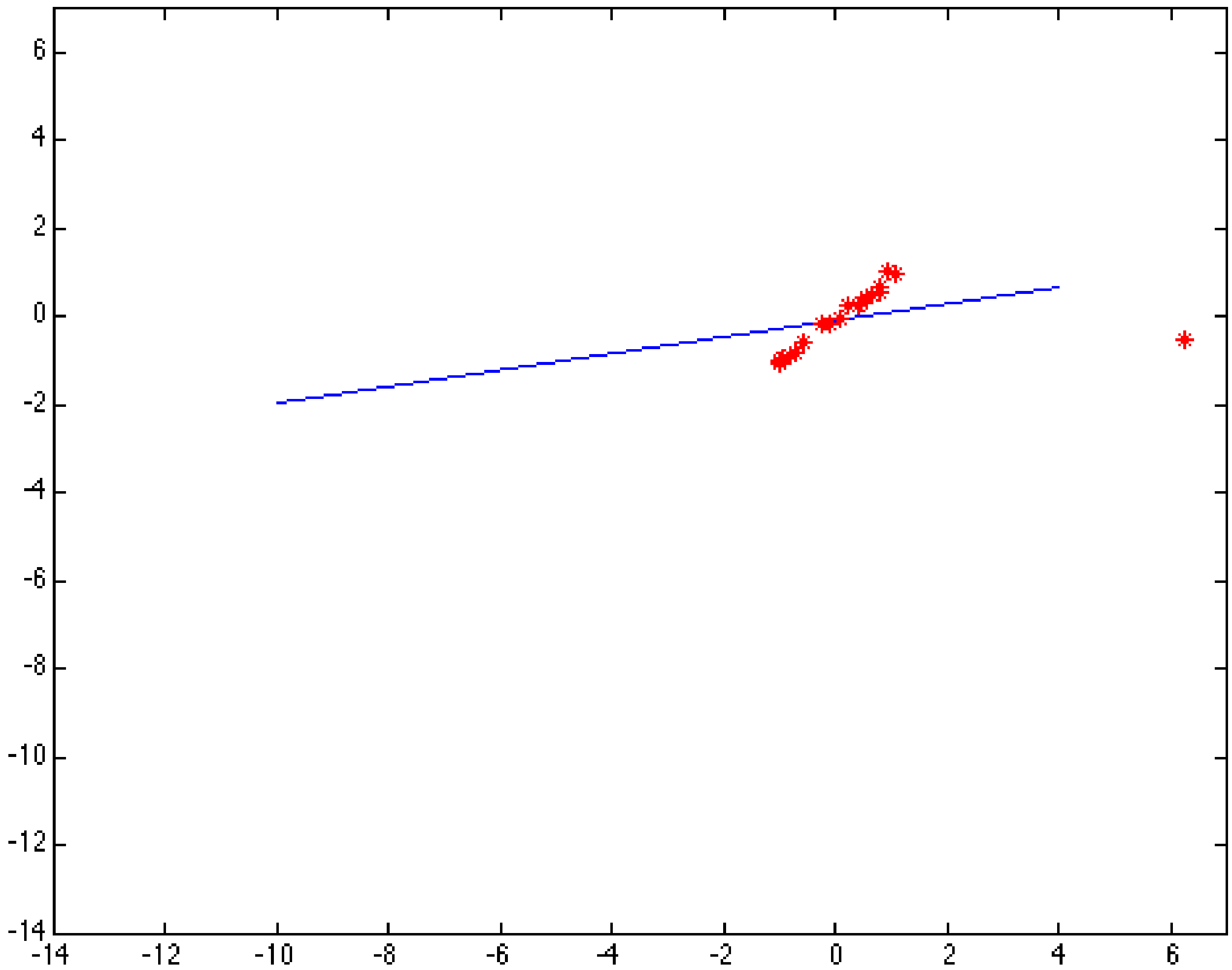
The effect of the outlier is minimized

Choosing the scale: Too small



The error value is almost the same for every point and the fit is very poor

Choosing the scale: Too large



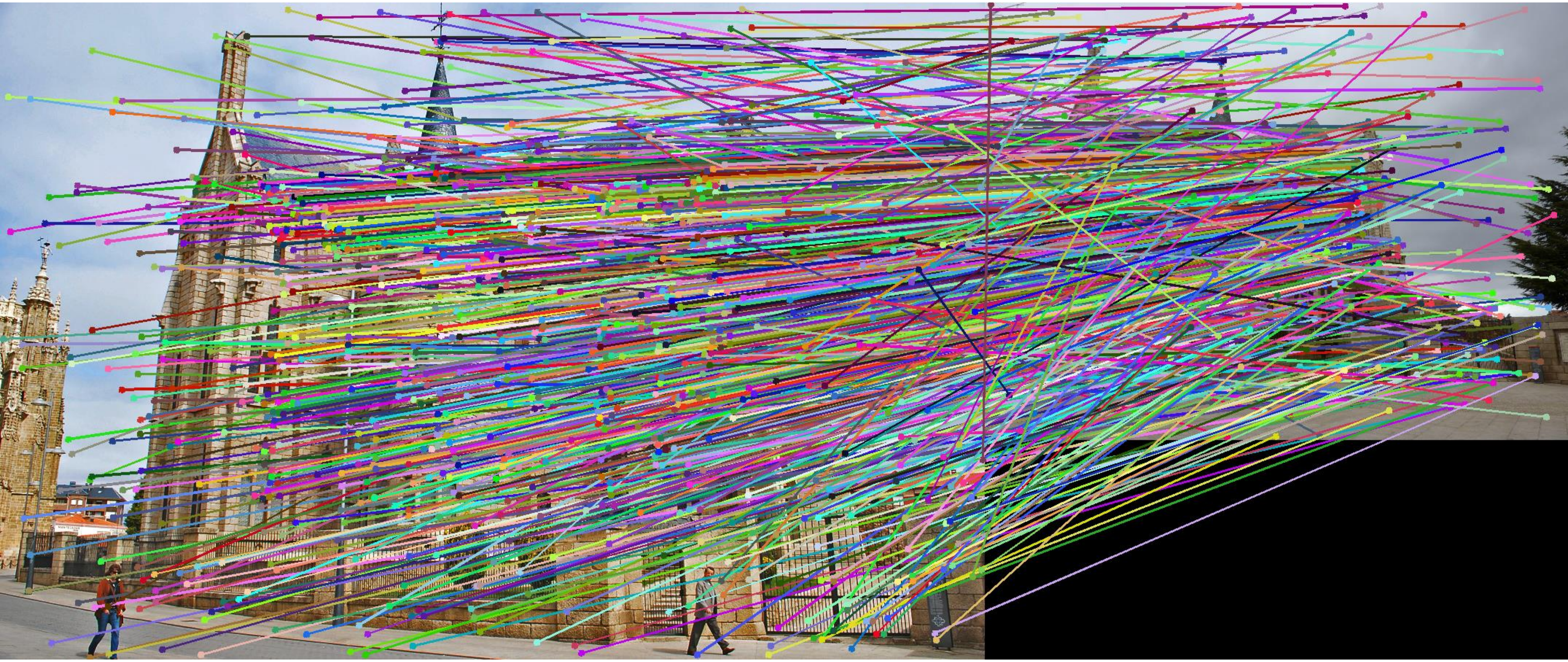
Behaves much the same as least squares

Robust estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Scale of robust function should be chosen adaptively based on median residual
- Least squares solution can be used for initialization

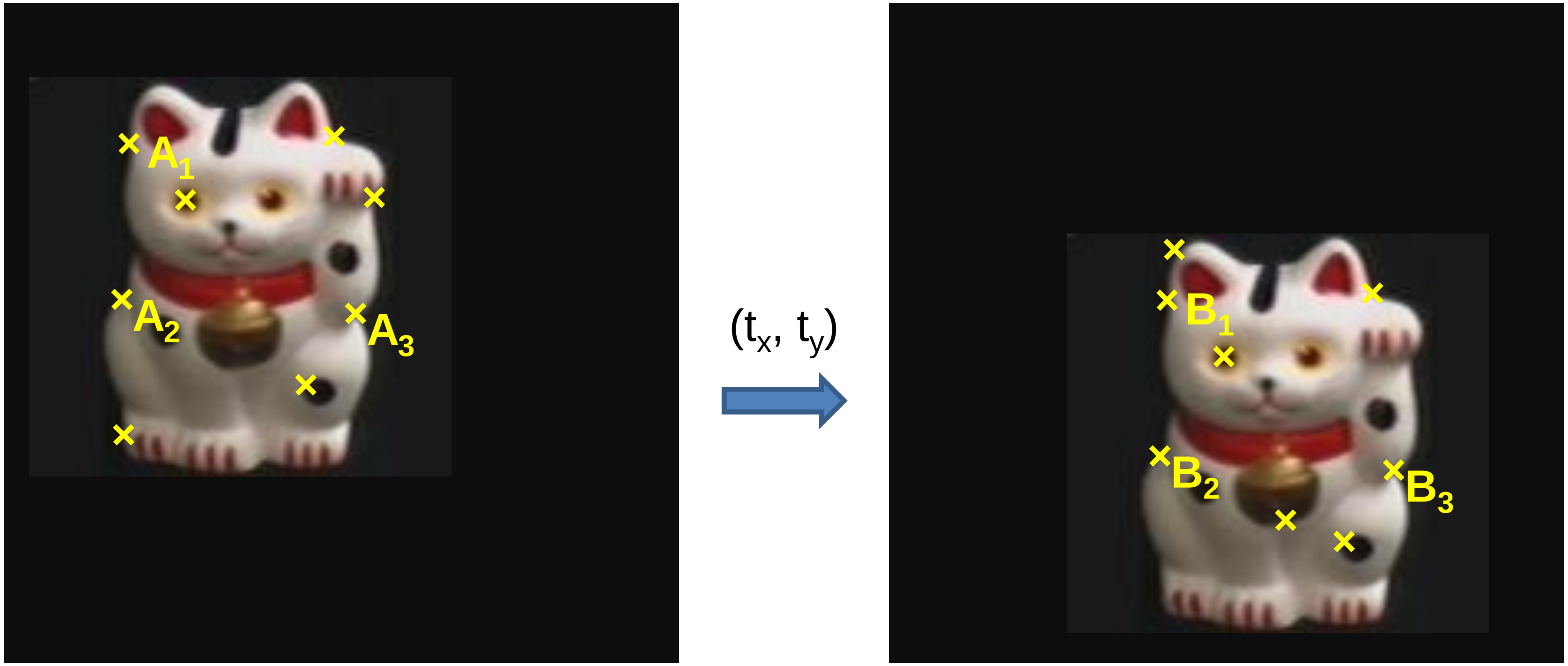
What if I have *many* outliers?

Episcopal Gaudi image pair



VLFeat's 800 most confident matches among 10,000+ local features.

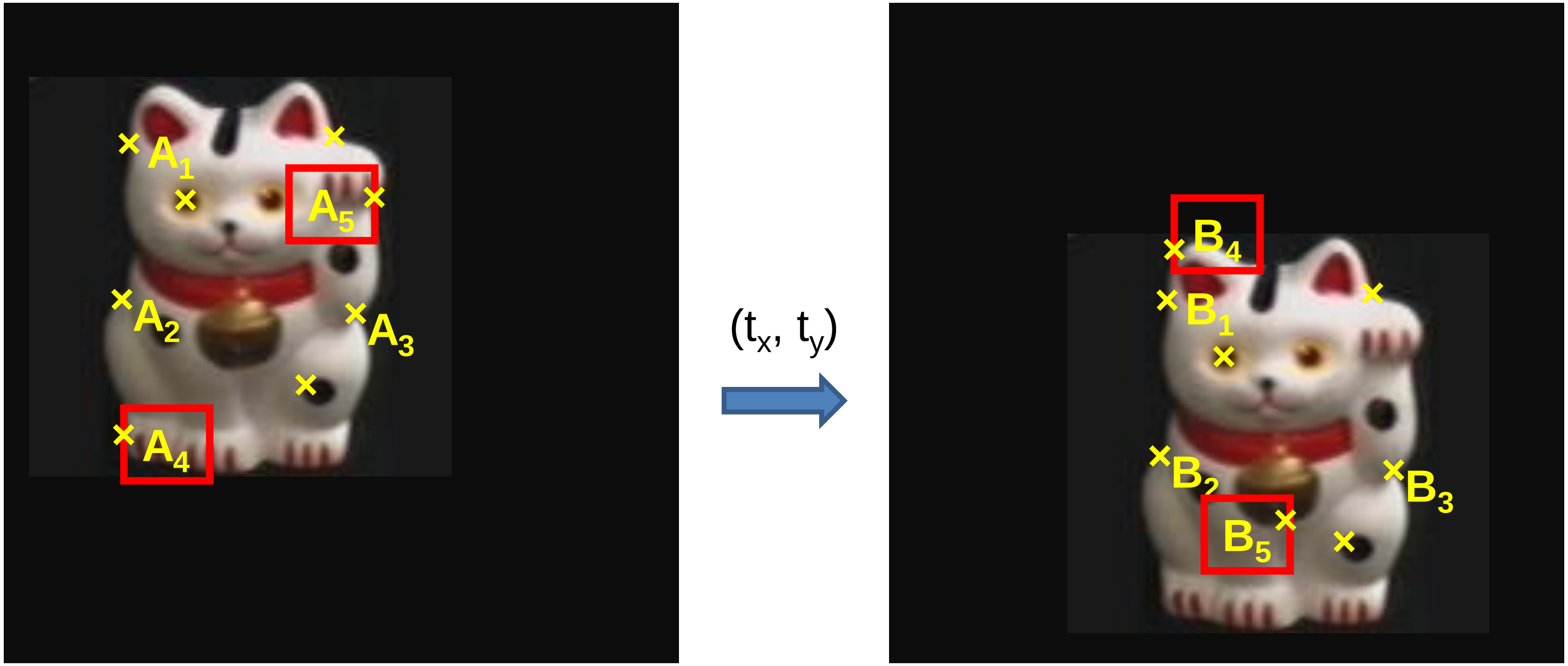
Example: solving for translation



Given matched points in {A} and {B}, estimate the translation of the object

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

Example: solving for translation



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

RANSAC solution

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

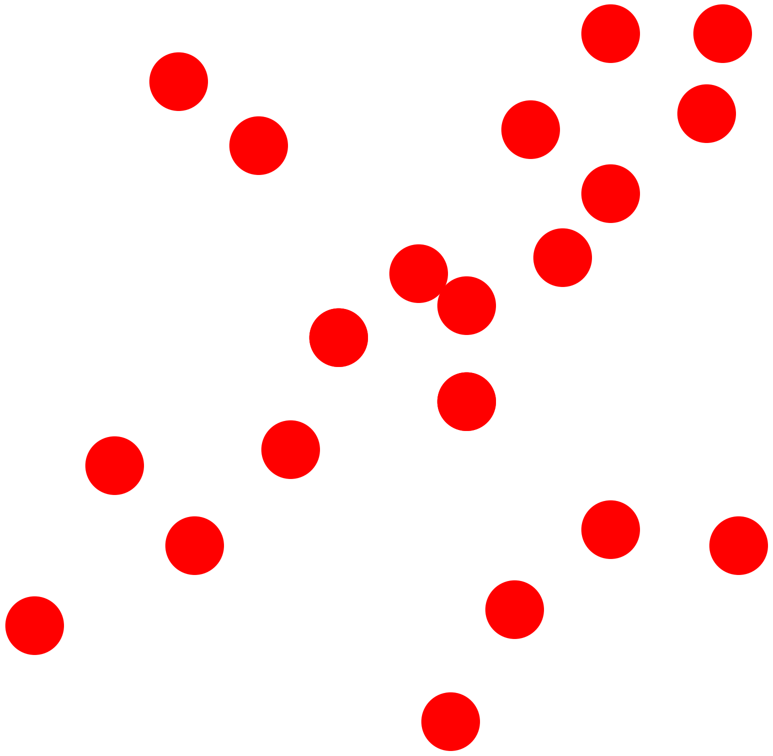
(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.

RANSAC

(RANdom SAmple Consensus) :

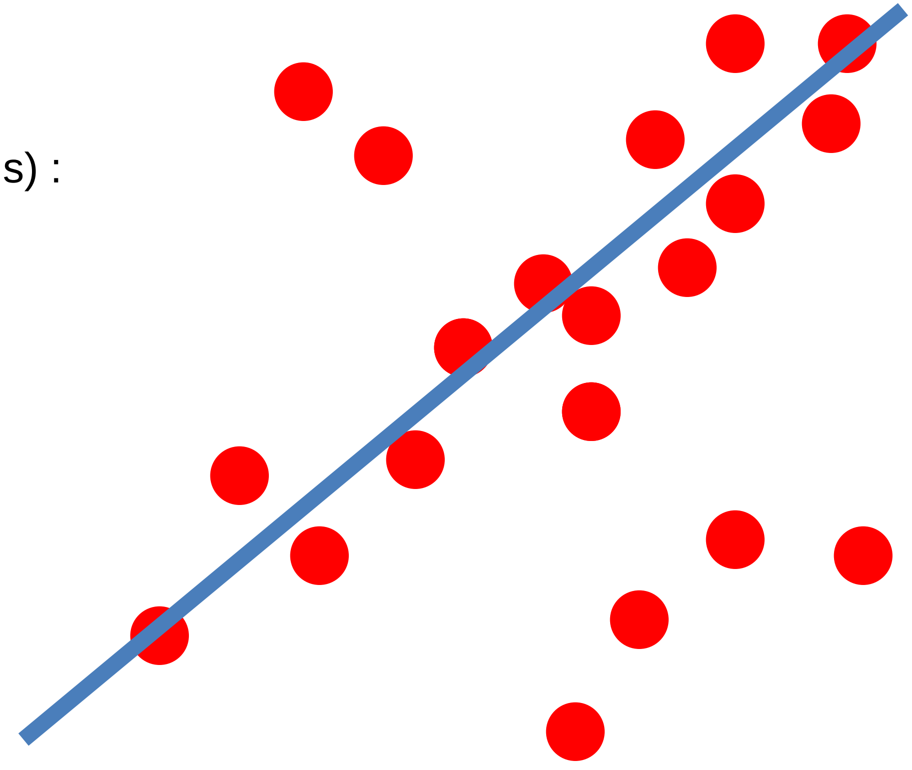
Fischler & Bolles in ‘81.



RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.

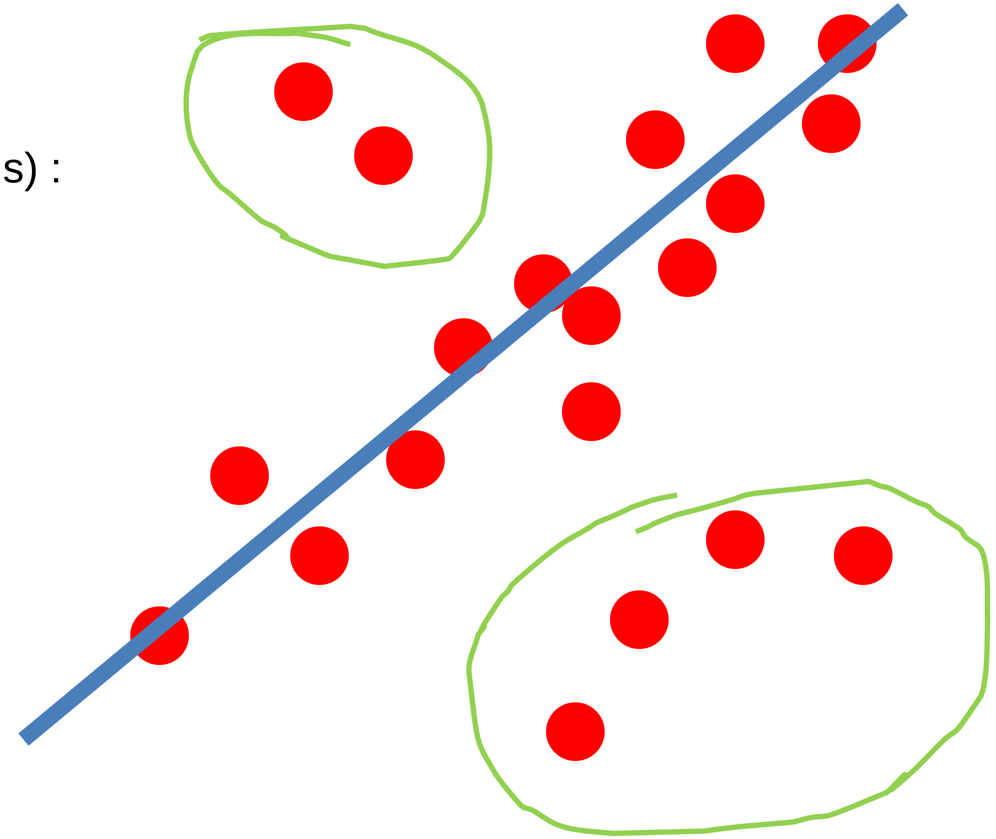


This data is noisy, but we expect a good fit to a known model.

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



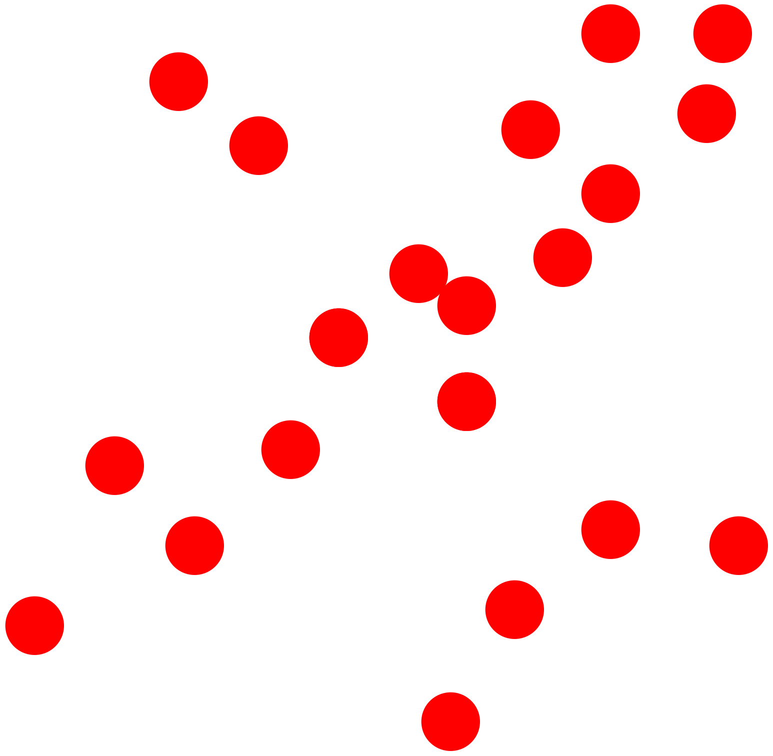
This data is noisy, but we expect a good fit to a known model.

Here, we expect to see a line, but least-squares fitting will produce the wrong result due to strong outlier presence.

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



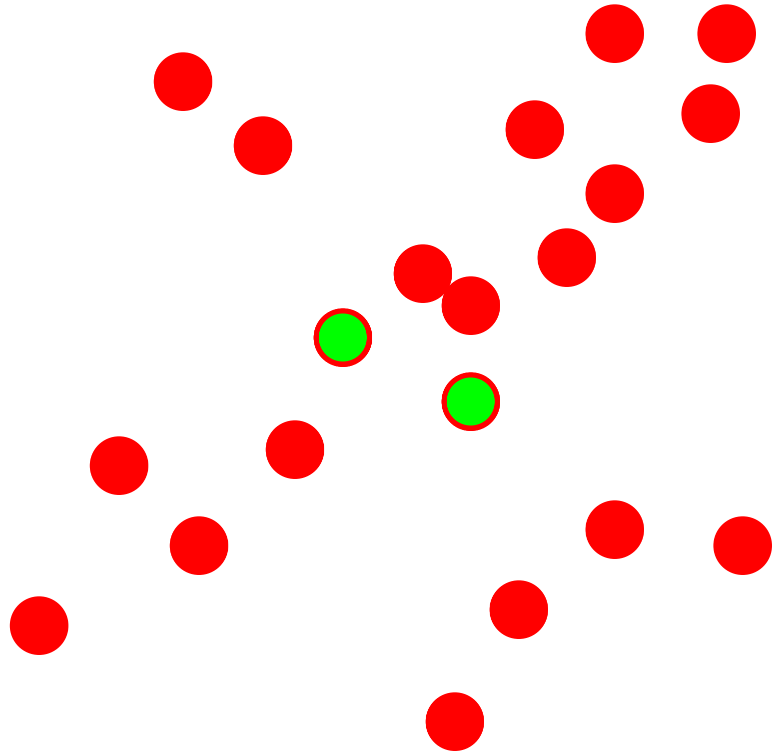
Algorithm:

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:

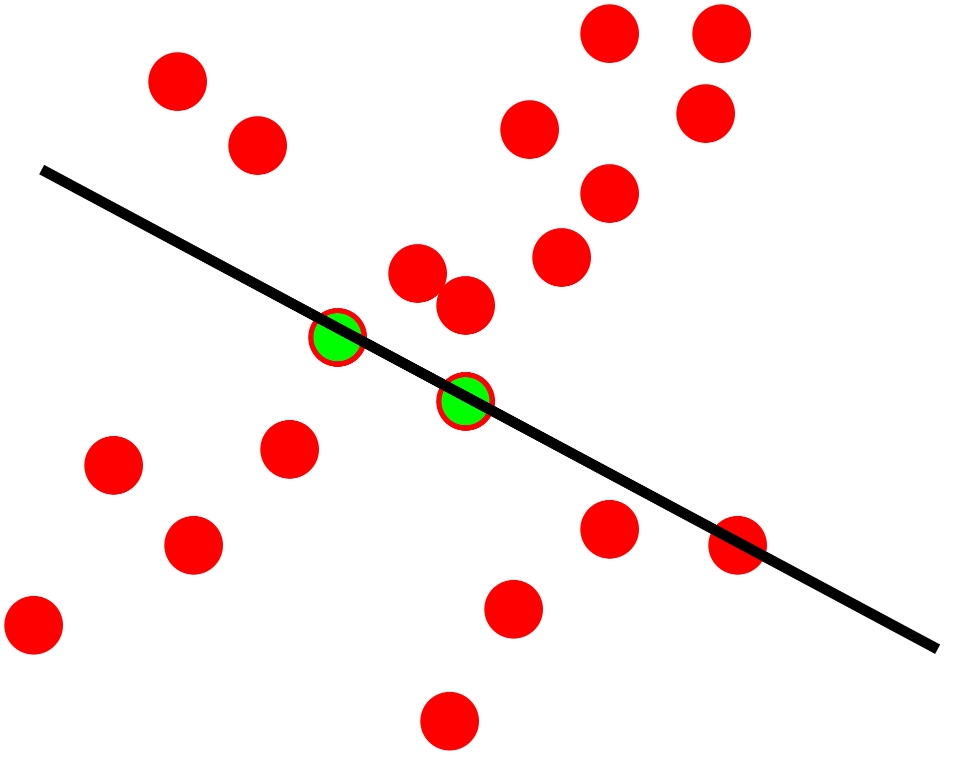
1. **Sample** (randomly) the number of points required to fit the model (s=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

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

Illustration by Savarese

RANSAC

Line fitting example

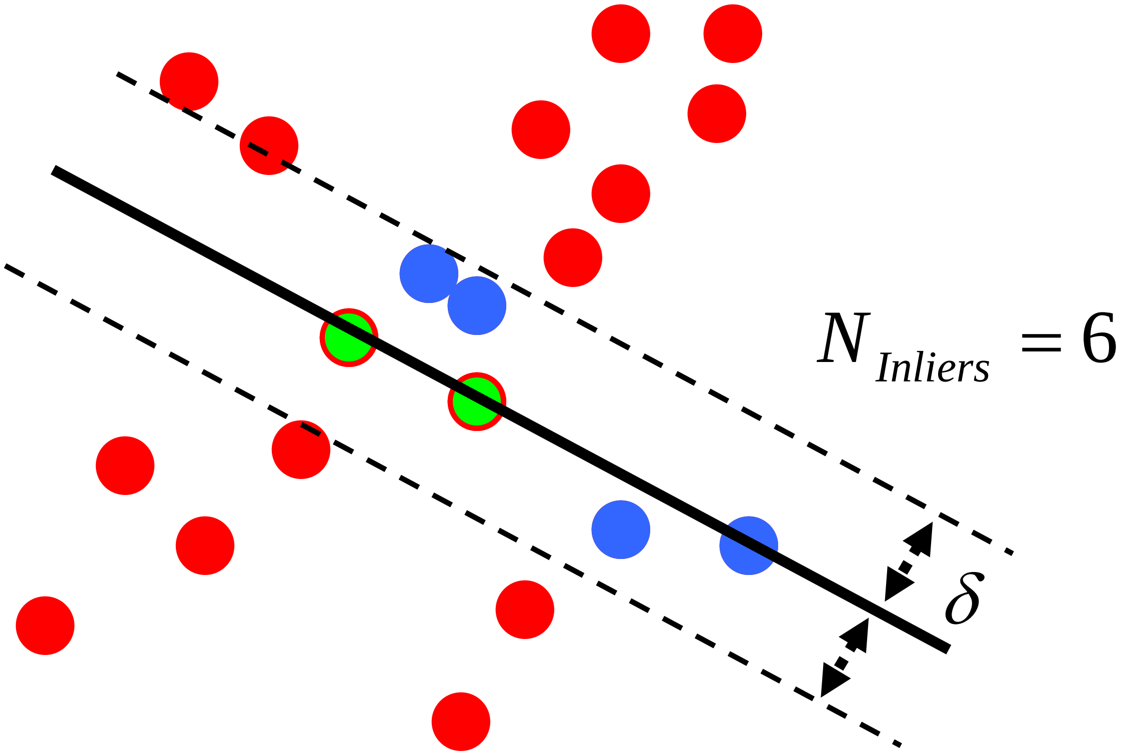


Algorithm:

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

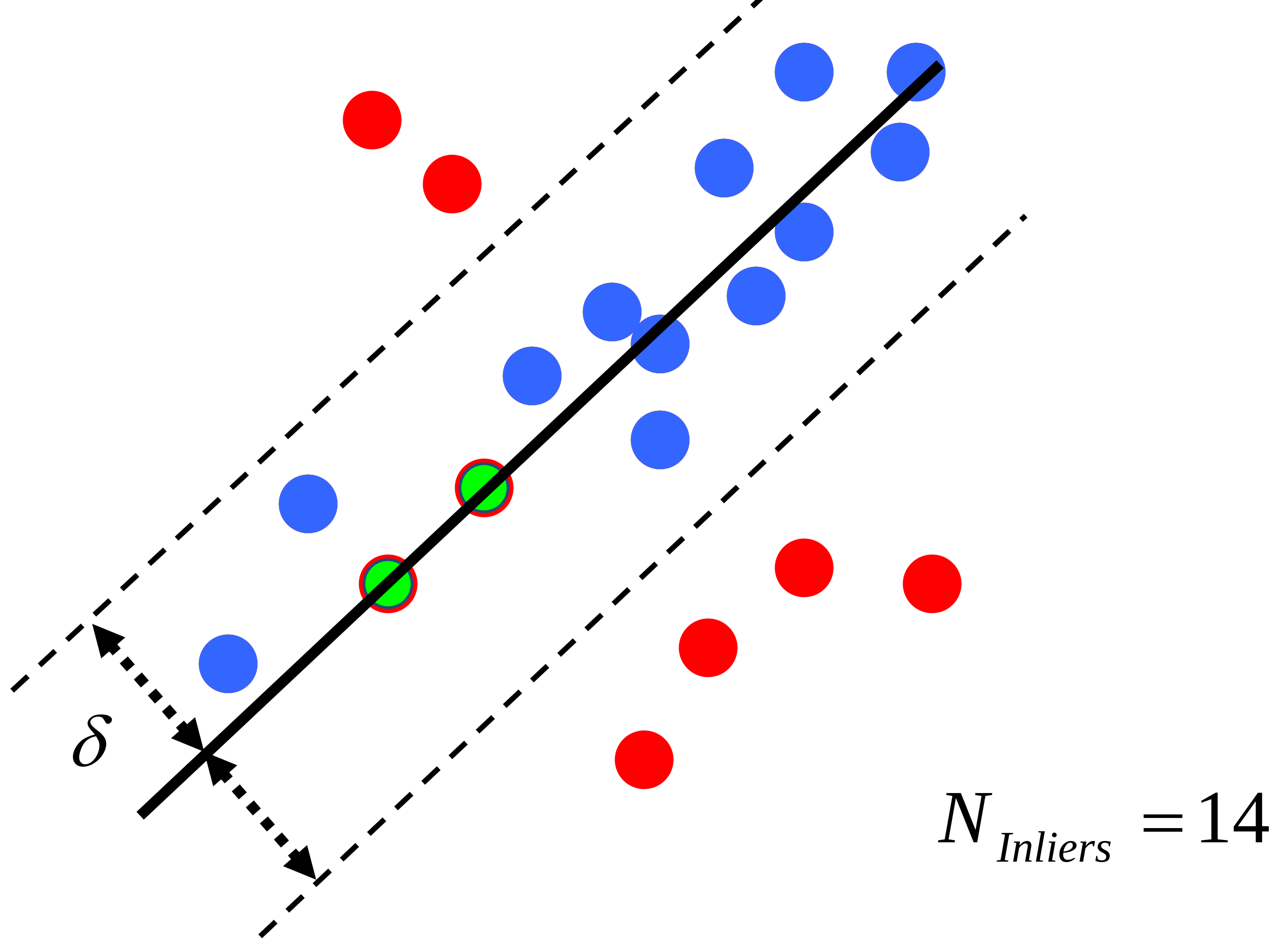


Algorithm:

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

How to choose parameters?

- Number of algorithm iterations *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g., *p*=0.99) (outlier ratio: *e*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

How many iterations do I need?

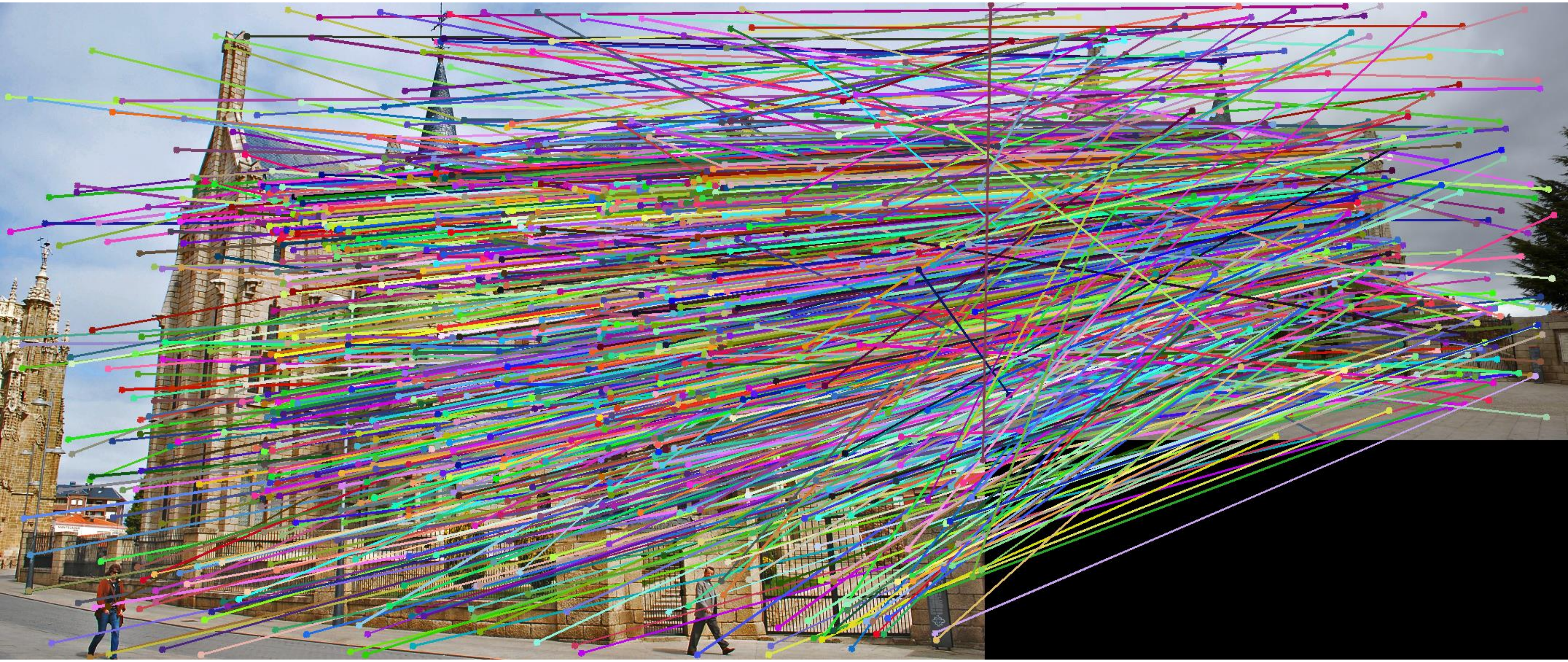
$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

	Proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

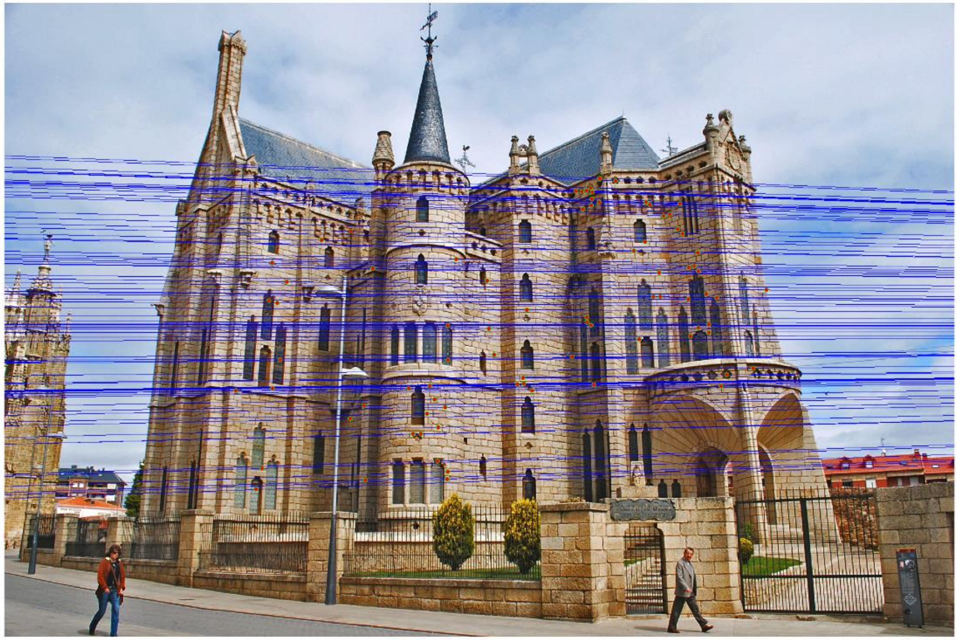
For $p = 0.99$

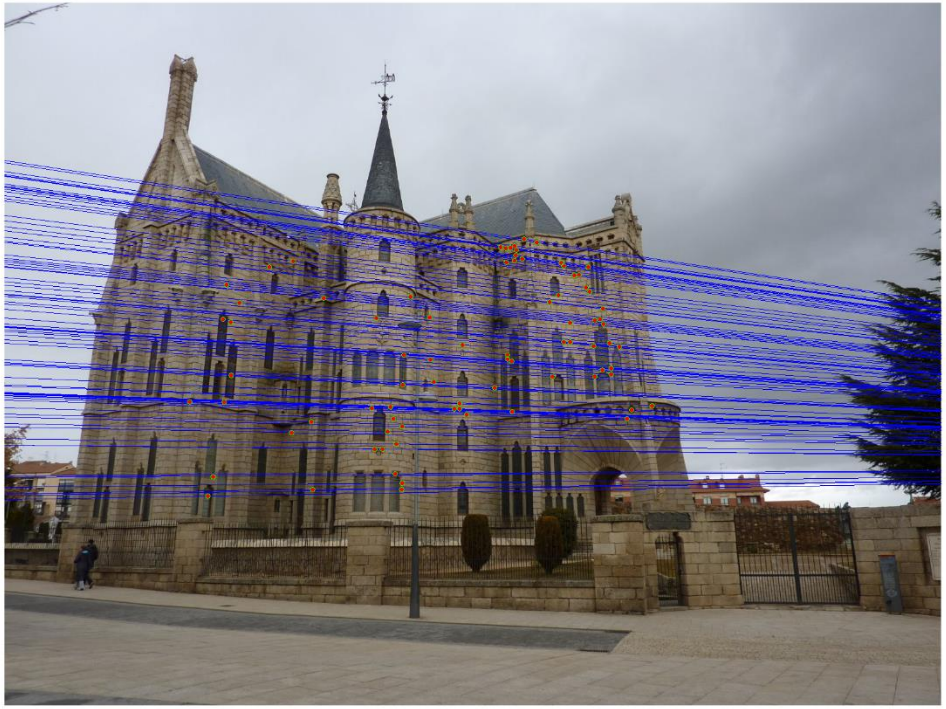
modified from M. Pollefeys

VLFeat's SIFT produces 800 most confident matches among 10,000+ local features.

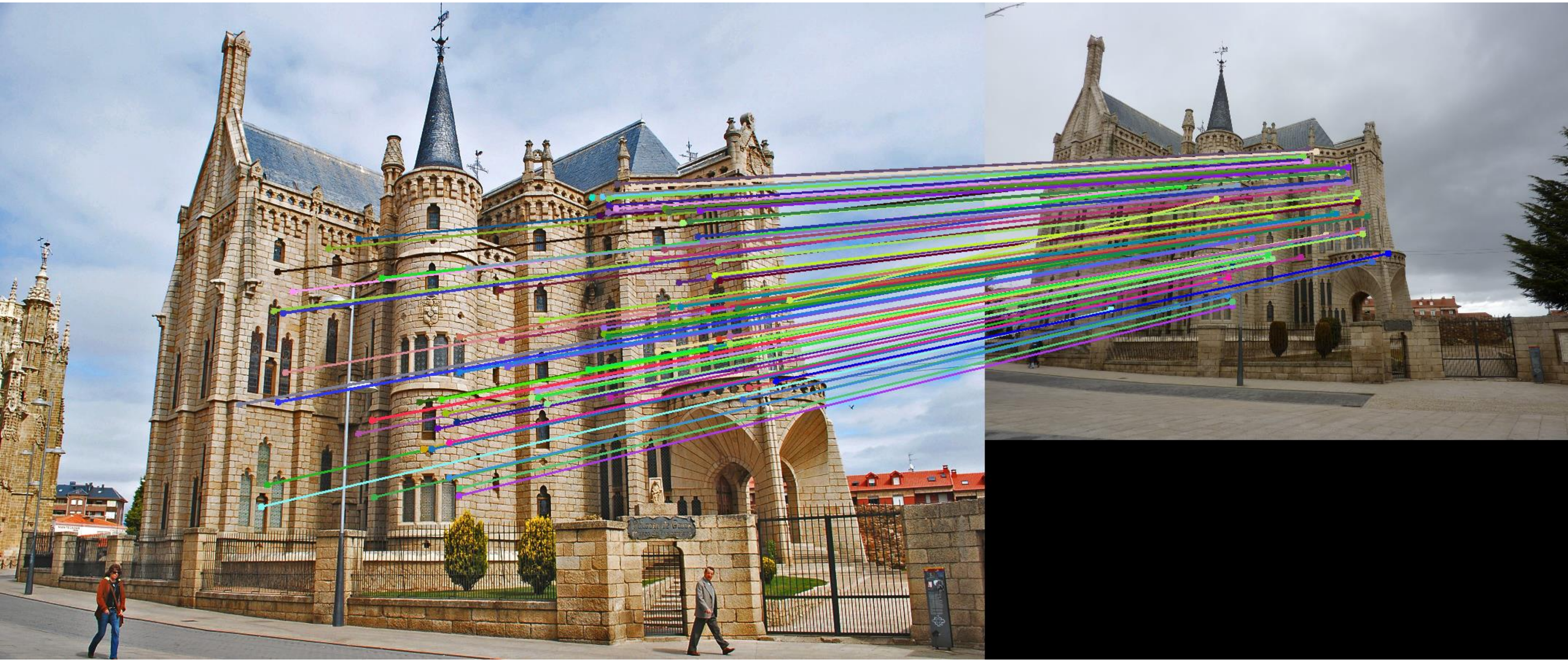


Epipolar lines





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



RANSAC conclusions

Good

- Robust to outliers
- Applicable for large number of objective function parameters (than Hough transform)
- Optimization parameters are easier to choose (than Hough transform)

Bad

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

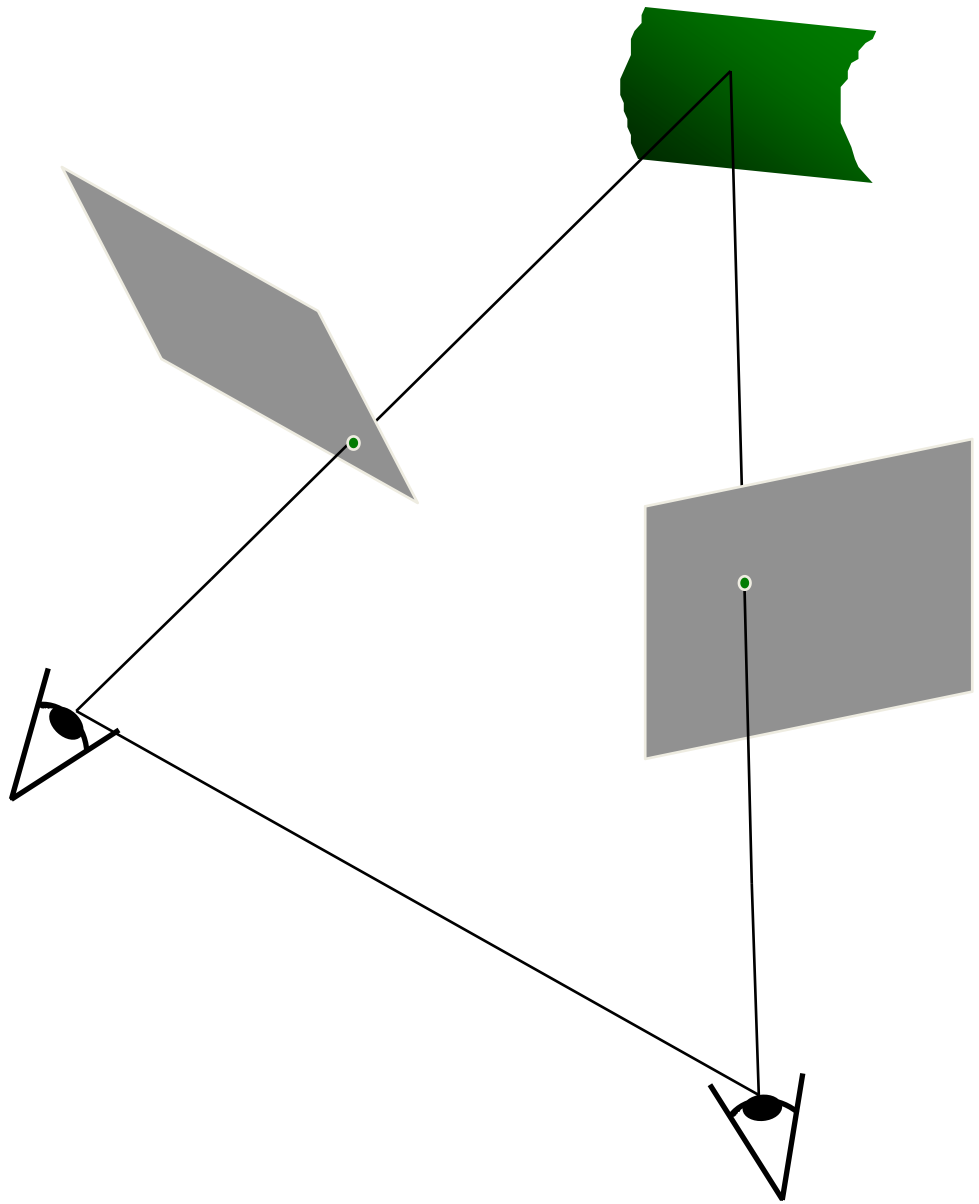
Common applications

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

Found F – now what?

SCANLINE ALIGNMENT VIA RECTIFICATION

Stereo image rectification

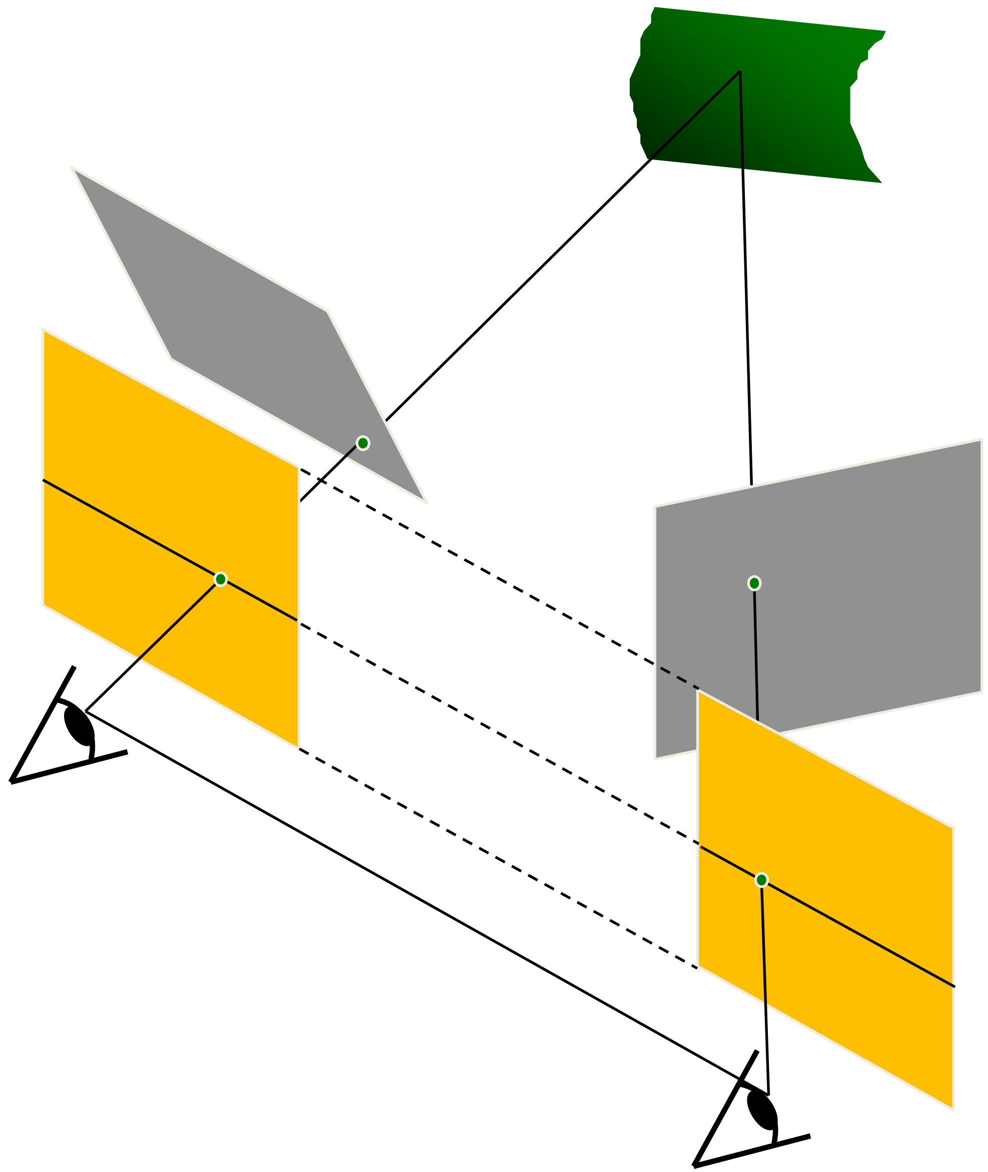


Stereo image rectification

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

- Pixel motion is horizontal after this transformation

- Two homographies (3x3 transform), one for each input image reprojection



➢ C. Loop and Z. Zhang. [Computing Rectifying Homographies for Stereo Vision]. IEEE Conf. Computer Vision and Pattern Recognition, 1999.

Rectification example