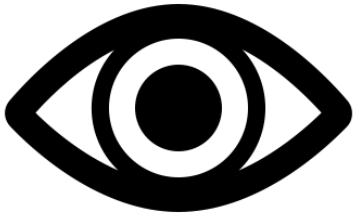


Advanced Image Processing

Introduction to Depth estimation and Multiple View Geometry

SYDE 671

Great! So now I have K and Rt

Well, what is that useful for?

Goal: reconstruct depth.

So far: we have ‘calibrated’ one camera.

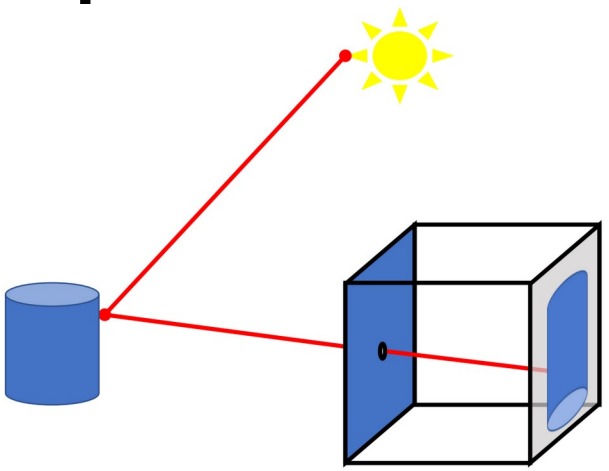
Or, potentially two…

Depth Cues

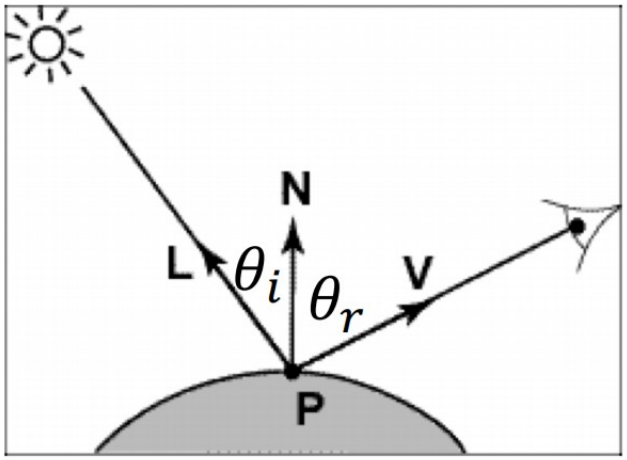
Shape from Shading and Photometric Stereo

Image formation process

- Rays from the light source “reflect” off a surface and reach camera.
- Reflection: Surface absorbs light energy and radiates it back.

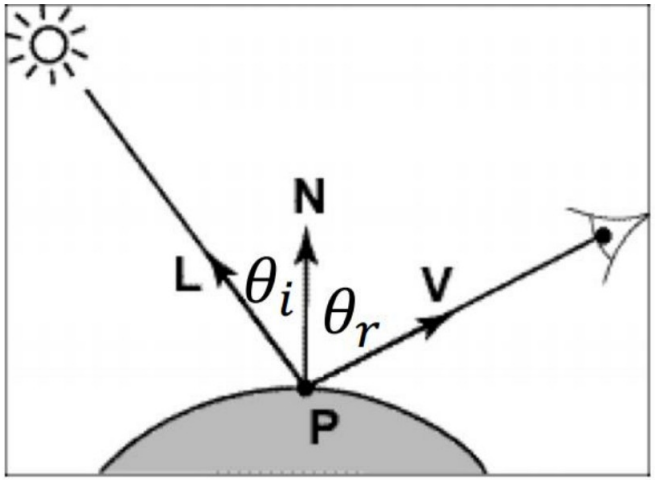


- Light of radiance L_i comes from light source at an incoming direction
- It sends out a ray of radiance L_r in the outgoing direction
- How does L_i relate to L_r ?



N is surface normal
L is direction of light, making with surface normal
V is viewing direction, making with surface normal

Image formation process



$$L_r = \rho(\theta_i, \theta_r) L_i \cos\theta_i$$

ρ : Bi-directional reflectance function (BRDF)

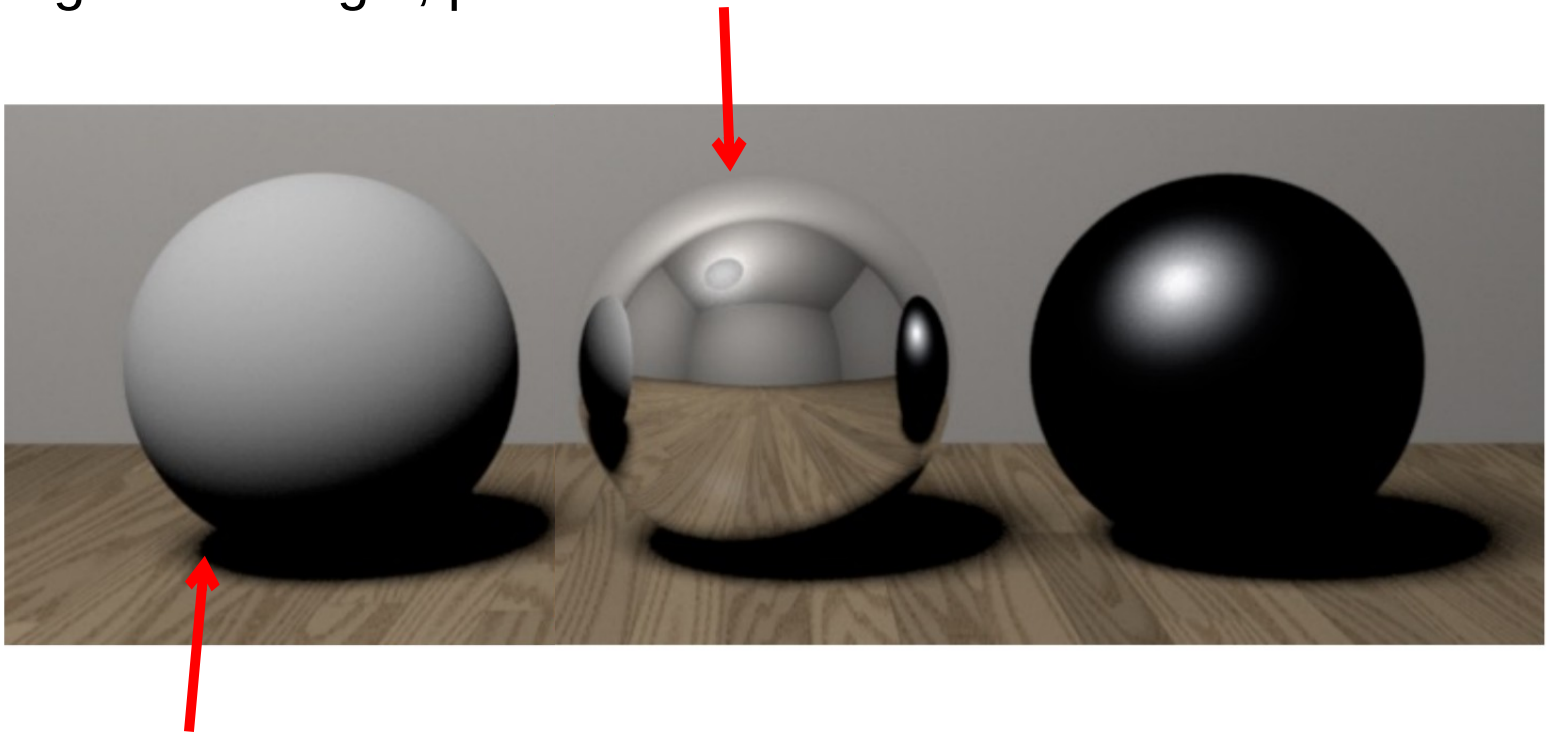
Special case 1: Perfect mirror (also called specular surface)

- $\rho(\theta_i, \theta_r)$ = 0 unless $\theta_i = \theta_r$

- Special case 2: Matte surface (also called lambertian surface)
- $\rho(\theta_i, \theta_r) = \rho_0$ = (constant)

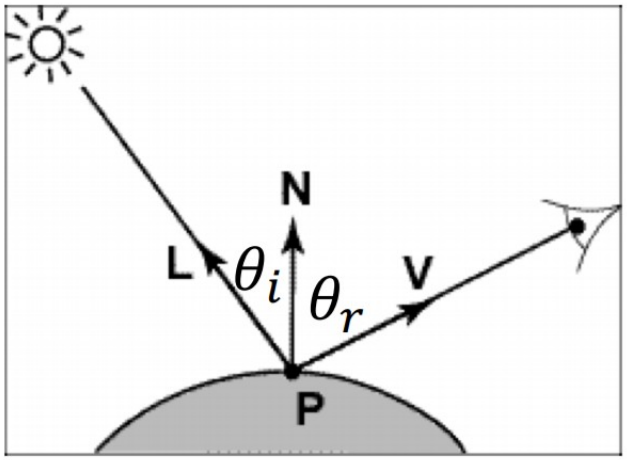
Image formation process

Perfect mirror (also called specular surface): Reflects light in a single, particular direction



Matte surface (also called lambertian surface): Reflected light is independent of viewing direction

Lambertian surface

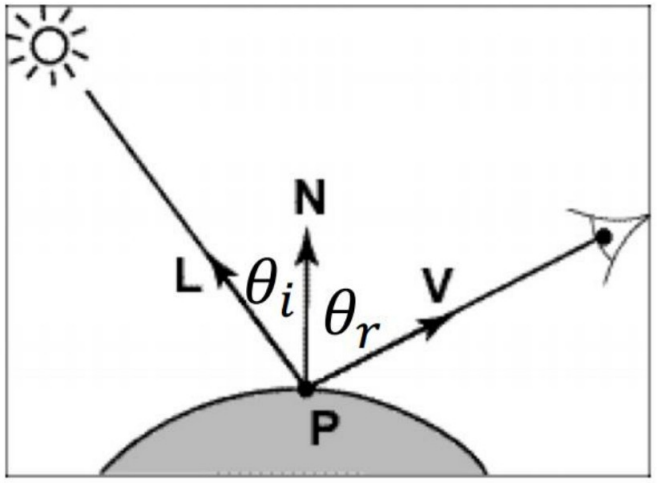


$$L_r = \rho(\theta_i, \theta_r) L_i \cos\theta_i$$

Assume the light is directional: all rays from light source are parallel (equivalent to a light source infinitely far away).

All surface points get light from the same direction **L** and of the same light intensity

Lambertian surface



$$L_r = \rho(\theta_i, \theta_r) L_i \cos\theta_i$$

$$\rightarrow L_r = \rho L_i \cos\theta_i$$

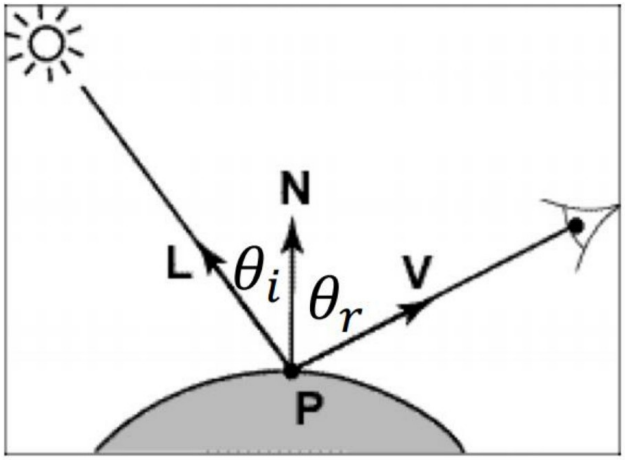
$$\rightarrow L_r = \rho L_i \mathbf{L} \cdot \mathbf{N}$$

For lambertian surfaces:

ρ is called albedo:

- Think of this as paint
- High albedo: white colored surface
- Low albedo: black surface
- Varies from point to point but ρ is a constant at a particular point (irrespective of the light angle)

Lambertian surface



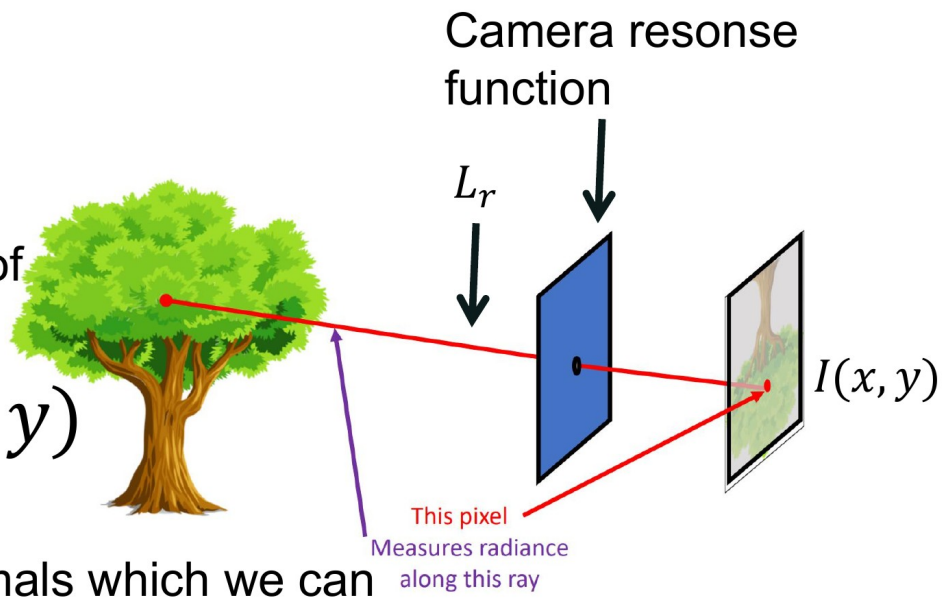
$$L_r = \rho(\theta_i, \theta_r) L_i \cos\theta_i$$

$$\rightarrow L_r = \rho L_i \cos\theta_i$$

$$\rightarrow L_r = \rho L_i \boldsymbol{L} \cdot \boldsymbol{N}$$

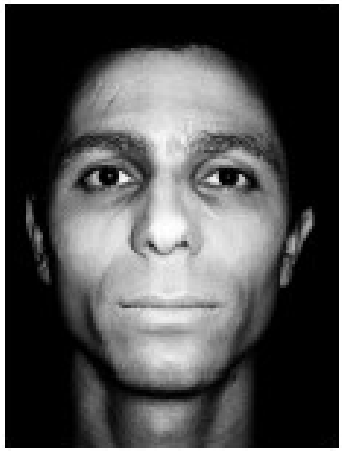
Assumption: camera response function is known: linear relationship between L_r and Image intensity, then at each location (x,y) of the image plane:

$$I(x,y) = \rho(x,y) L_i \boldsymbol{L} \cdot \boldsymbol{N}(x,y)$$

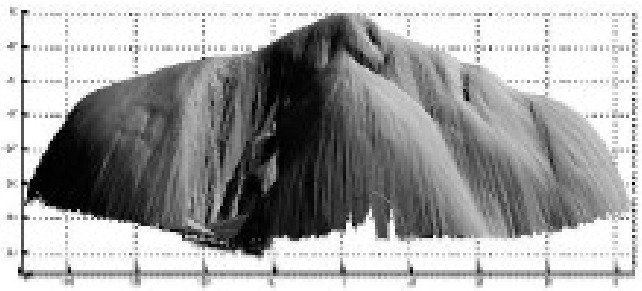


Equation is a constraint on albedo and normals which we can use to recover them.

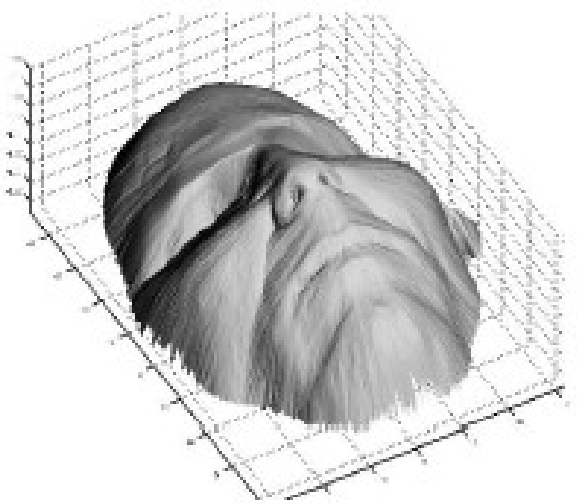
method 1: Shape from Shading SfS



a)



b)



c)

Assumptions: Known properties of the light source, the surface reflectance and the camera response.

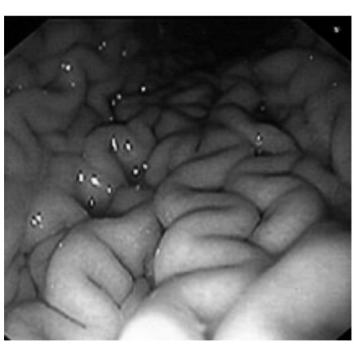
Then it is possible to reconstruct a 3D map of the surface from one image.

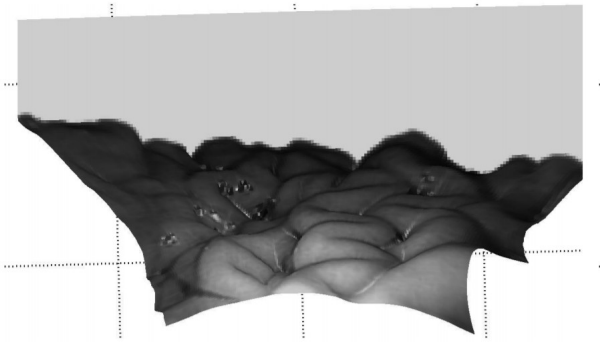
Very Hard to get it to work well in practice!

[Figure from Prados & Faugeras 2006]

Photometric Stereo

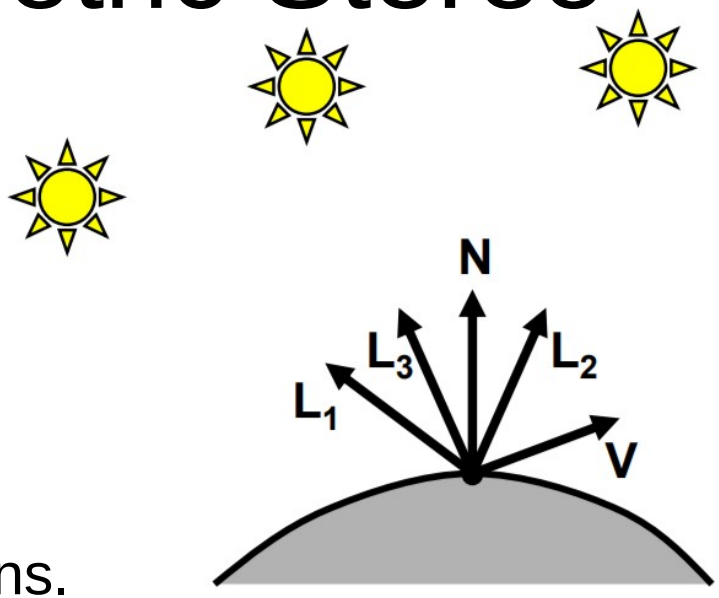
- Shape from shading is a special case of photometric stereo where the number of images is one.





- Photometric stereo: takes multiple of images from a fixed position, while varying the light source location.

Photometric Stereo



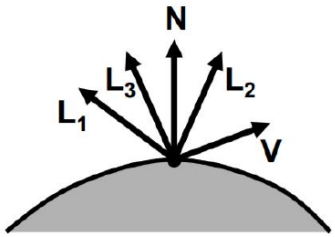
Assumptions:

1) Known light source directions.
2) Light rays are parallel (e.g. light source placed at infinity).
3) No ambient illumination (1 light source illuminates a point P).
4) Camera response function is known
5) Light intensity Li =1 : same light source used for the multiple images, if not then need to account for it which may complicate the math a bit but feasible.

Photometric Stereo

The for each pixel in each image:

$I = \rho \boldsymbol{L} \cdot \boldsymbol{N} = \rho \boldsymbol{N}^{\boldsymbol{T}} \cdot \boldsymbol{L}$



Define G = $\rho \boldsymbol{N}$

where G is a 3 dimensional vector, norm of G = ρ and the direction of G = direction of the 3D surface normal.

Our constraint becomes:

$$I = G^T L = L^T G$$

take many images from same point of view but different light source directions L and stack them together. Then for each pixel we have a set of equations coming from different images:

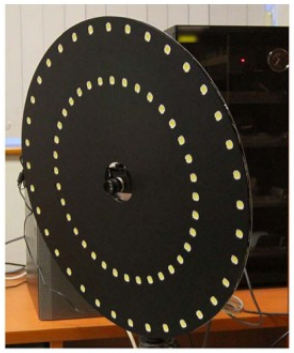
$$I_1 = L_1^T G$$
$$I_2 = L_2^T G$$
$$\vdots$$
$$I_k = L_k^T G$$

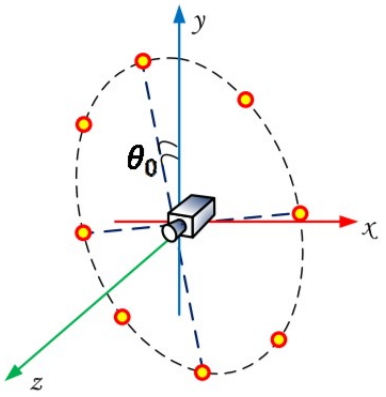
Photometric Stereo

We're assuming that lighting directions L_k is known either from:

1. calibration using a chrome sphere

2. physically measuring the light source angle to the target area (done in applications where multiple source and camera are rigidly mounted on a machine)





Photometric Stereo

G is a 3 dimensional vector and thus requires a minimum of 3 constraints ie. 3 images to fully constrain it.
Get better results by using more lights.

minimal case:

$$\underbrace{\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix}}_{\mathbf{I}} = \underbrace{\begin{bmatrix} \mathbf{L_1}^T \\ \mathbf{L_2}^T \\ \mathbf{L_3}^T \end{bmatrix}}_{\mathbf{L}} \underbrace{\rho_d \mathbf{N}}_{\mathbf{G}}$$

$3 \times 1 \quad 3 \times 3 \quad 3 \times 1$

$$\mathbf{G} = \mathbf{L}^{-1}\mathbf{I}$$

Over constrained case:

$$\begin{bmatrix} I_1 \\ \vdots \\ I_n \end{bmatrix} = \begin{bmatrix} \mathbf{L_1} \\ \vdots \\ \mathbf{L_n} \end{bmatrix} \rho_d \mathbf{N}$$

Least square solution:

$$\begin{aligned} \mathbf{I} &= \mathbf{LG} \\ \mathbf{L^T I} &= \mathbf{L^T L G} \\ \mathbf{G} &= (\mathbf{L^T L})^{-1}(\mathbf{L^T I}) \end{aligned}$$

$$\rho_d = \|\mathbf{G}\|$$

$$\mathbf{N} = \frac{1}{\rho_d}\mathbf{G}$$

The albedo should be between 0 and 1. Use this as a sanity check

Photometric Stereo

Repeat the above for every pixel in the images to form an albedo map and a normal map.

Shape from Normals:

The surface is (x, y, f(x,y)) and the normal is

$$N(x,y) = \frac{1}{\sqrt{1+\frac{\partial f}{\partial x}^2+\frac{\partial f}{\partial y}^2}}\left\{\frac{\partial f}{\partial x},\frac{\partial f}{\partial y},1\right\}^T$$

To recover the depth map we need to find f(x,y) from the unit normals. Assume that the measured value of the unit normal at some point (x, y) is (a(x, y), b(x, y), c(x, y)), then:

$$\frac{\partial f}{\partial x} = \frac{a(x,y)}{c(x,y)} \text{ and } \frac{\partial f}{\partial y} = \frac{b(x,y)}{c(x,y)}$$

This gives us a new sanity check:

$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}$$

Photometric Stereo

- The surface can be reconstructed up to some constant depth error c.
- The partial derivative gives the change in surface height with a small step in either the x or the y direction.
- Therefore we get the surface by summing these changes in height along some path.

$$f(x,y) = \oint_C \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \cdot d\boldsymbol{l} + c$$

up to some constant depth error c

One possible integration path:

$$f(u,v) = \int_0^v \frac{\partial f}{\partial y}(0,y)\,dy + \int_0^u \frac{\partial f}{\partial x}(x,v)\,dx + c$$

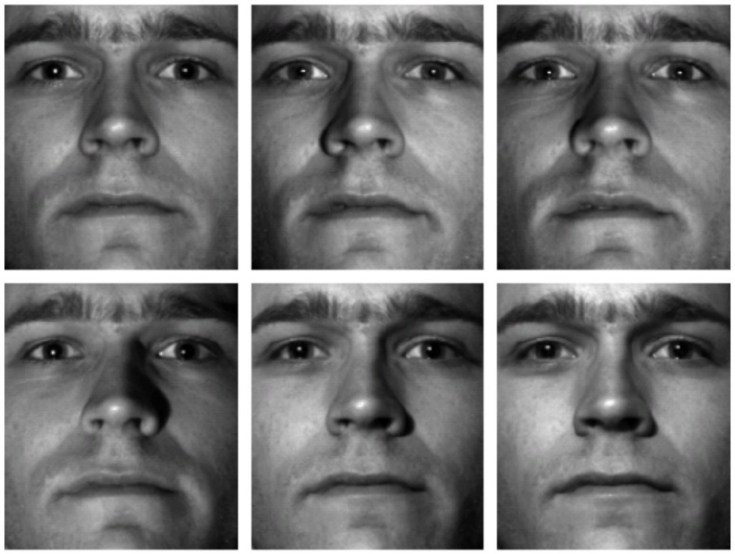
Photometric Stereo algorithm

Final algorithm is very simple!

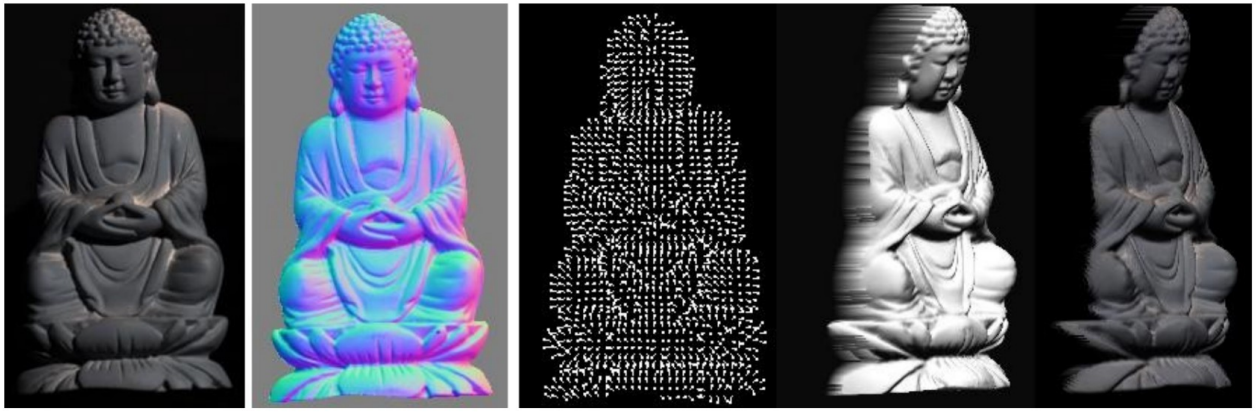
Obtain many images in a fixed view under different illuminants
Determine the matrix $\mathcal{V}$ from source and camera information
Inferring albedo and normal:
For each point in the image array that is not shadowed
  Stack image values into a vector $\boldsymbol{i}$
  Solve $\mathcal{V}\boldsymbol{g} = \boldsymbol{i}$ to obtain $\boldsymbol{g}$ for this point
  Albedo at this point is $|\boldsymbol{g}|$
  Normal at this point is $\frac{\boldsymbol{g}}{|\boldsymbol{g}|}$
  p at this point is $\frac{N_1}{N_3}$
  q at this point is $\frac{N_2}{N_3}$
end
Check: is $(\frac{\partial p}{\partial y} - \frac{\partial q}{\partial x})^2$ small everywhere?

Integration:
Top left corner of height map is zero
For each pixel in the left column of height map
  height value = previous height value + corresponding q value
end
For each row
  For each element of the row except for leftmost
    height value = previous height value + corresponding p value
  end
end

Photometric Stereo results



from Athos Georghiades

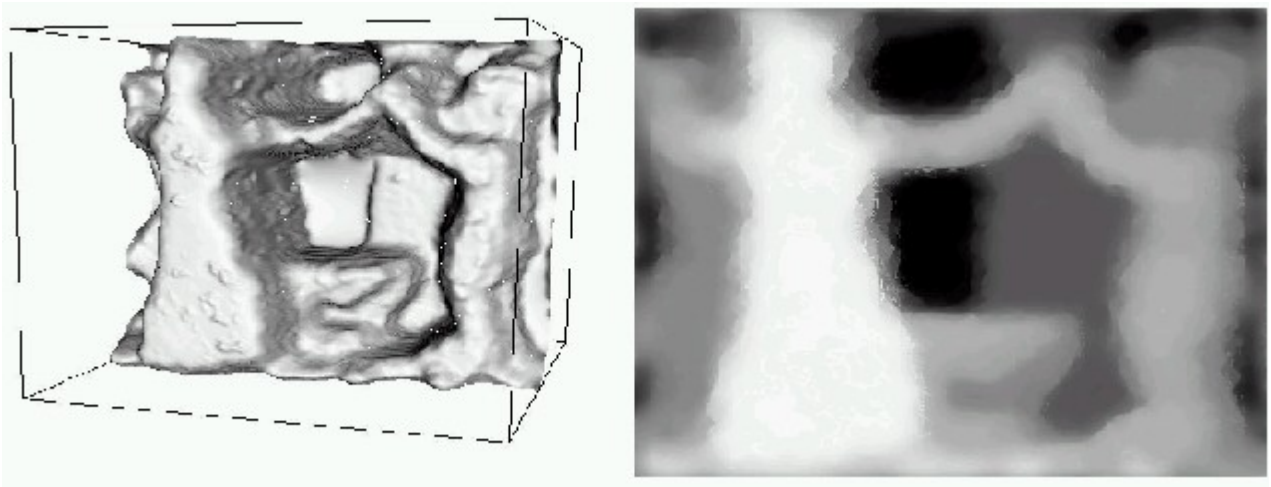


Input (1 of 12) | Normals (RGB colormap) | Normals (vectors) | Shaded 3D rendering | Textured 3D rendering

Other depth cues

Focus/defocus

Images from same point of view, different camera parameters

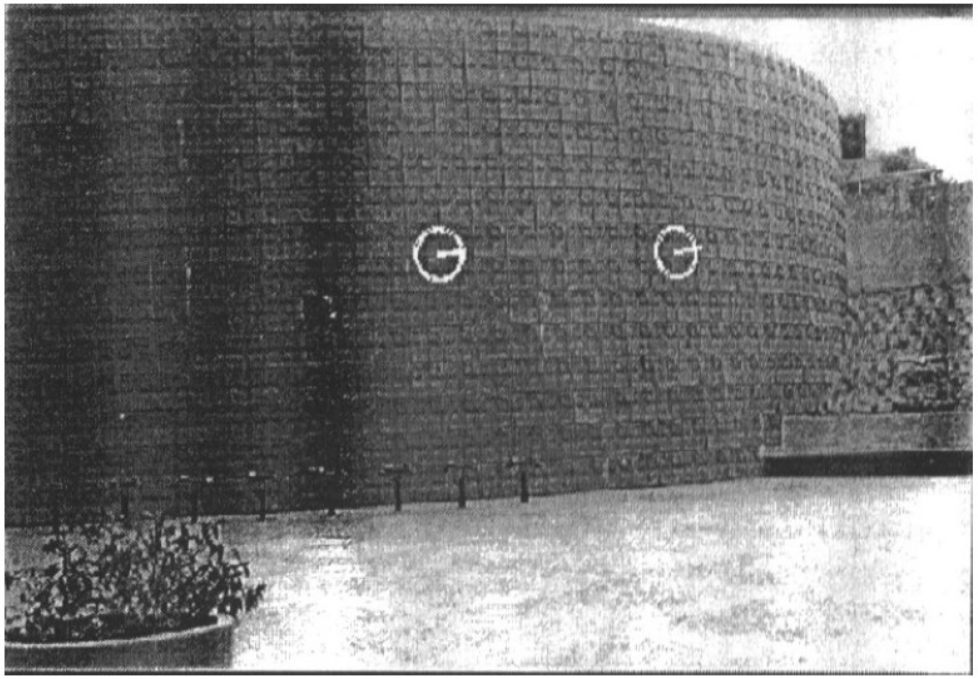


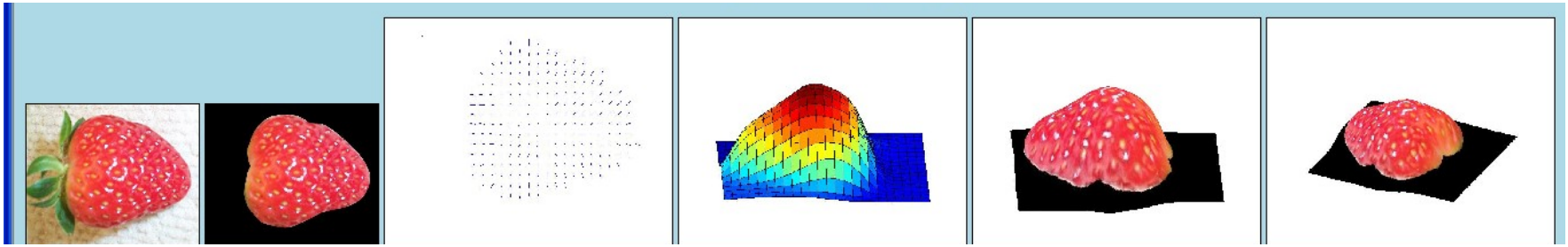
3d shape / depth estimates

A geometric approach to shape from defocus

[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

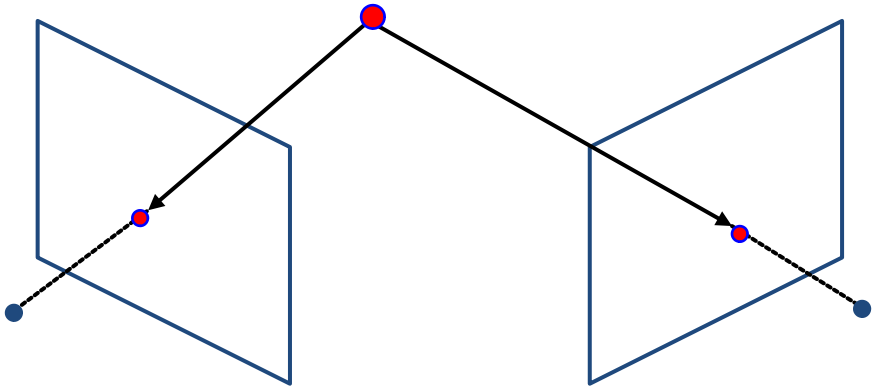
Figures from L. Zhang

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

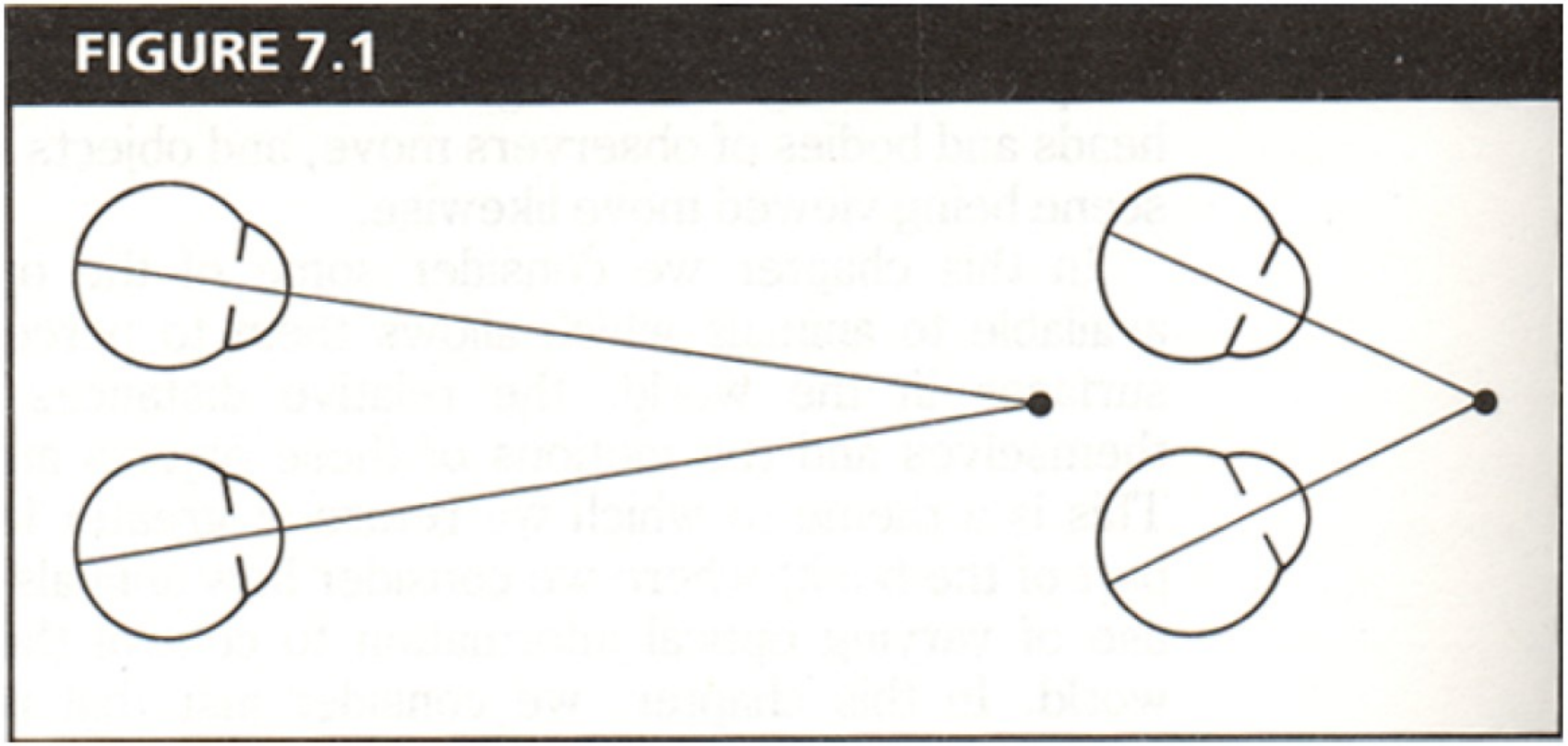


Slides: James Hays and Kristen

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

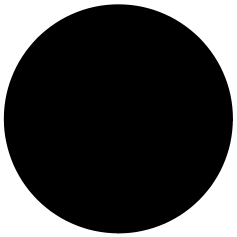
Human stereopsis

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



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

Yes, you can be stereoblind.



Put your finger tip in line with the dot between your eyes and the screen.

Then focus on the dot -> you should see two fingers.
Then focus on your finger -> you should see two dots.

If in the last stage you only see one finger and one dot, then you might be stereoblind…..

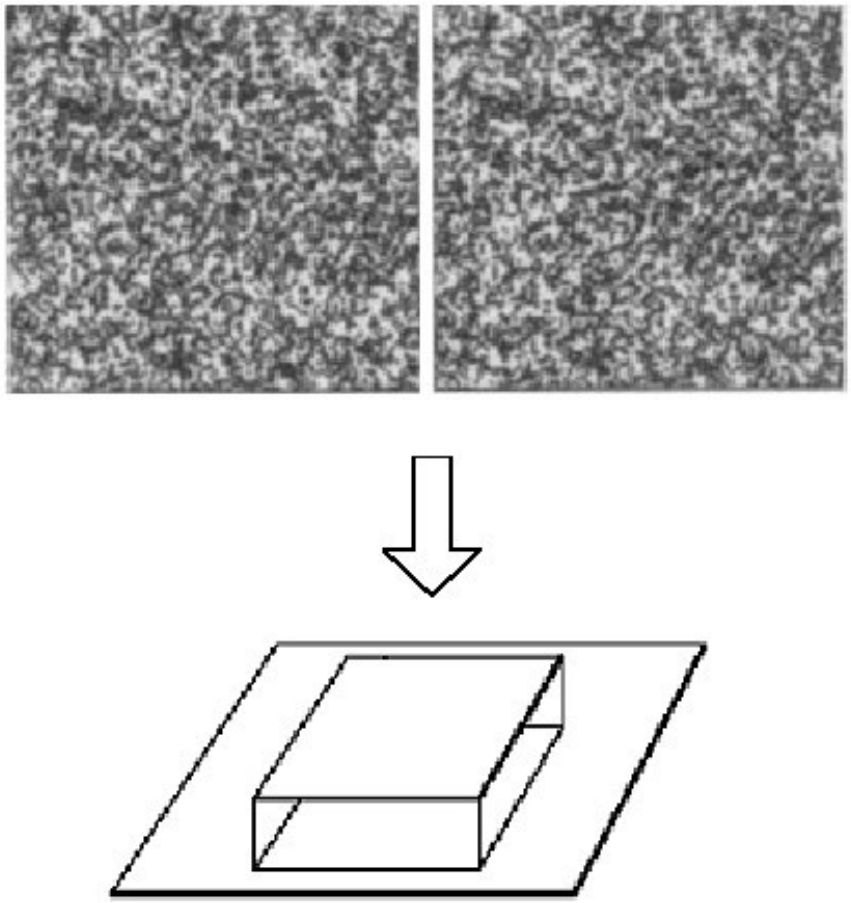
Random Dot Stereograms

- 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

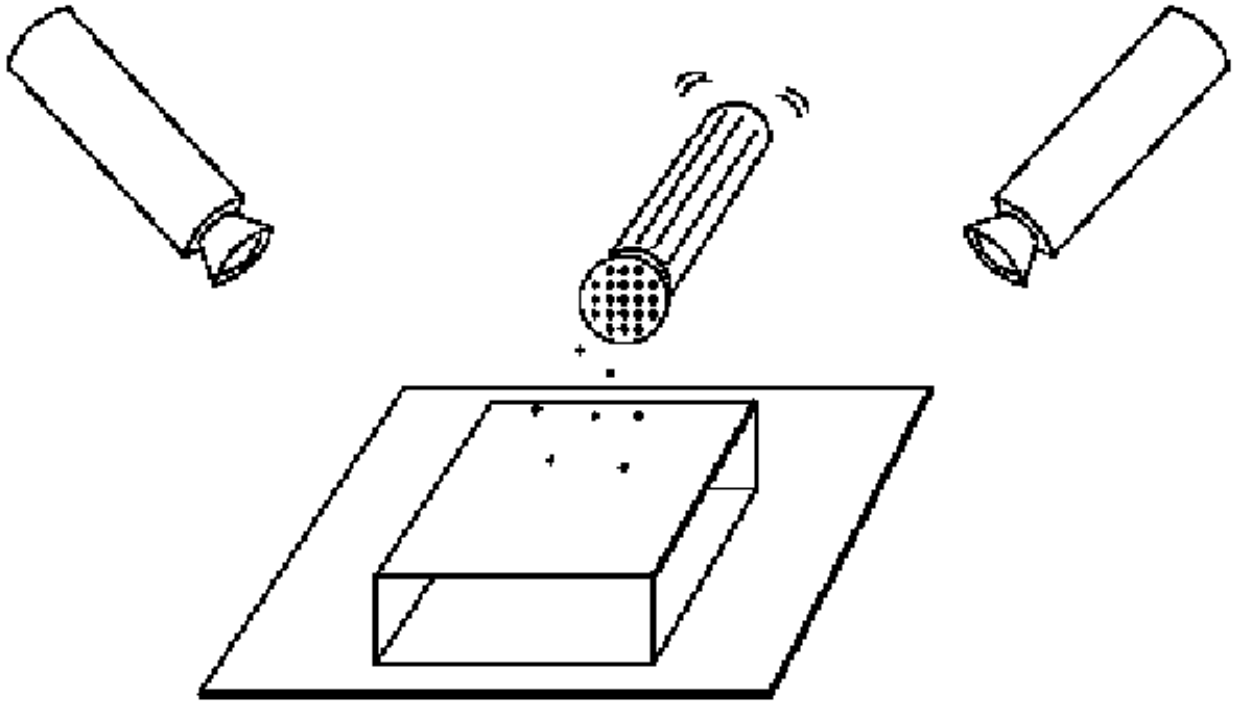


1. Create an image of suitable size. Fill it with random dots. Duplicate the image.

2. Select a region in one image.

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

Random dot stereograms



Forsyth & Ponce

Random dot stereograms

- When viewed monocularly, they appear random; when viewed stereoscopically, see 3d structure.

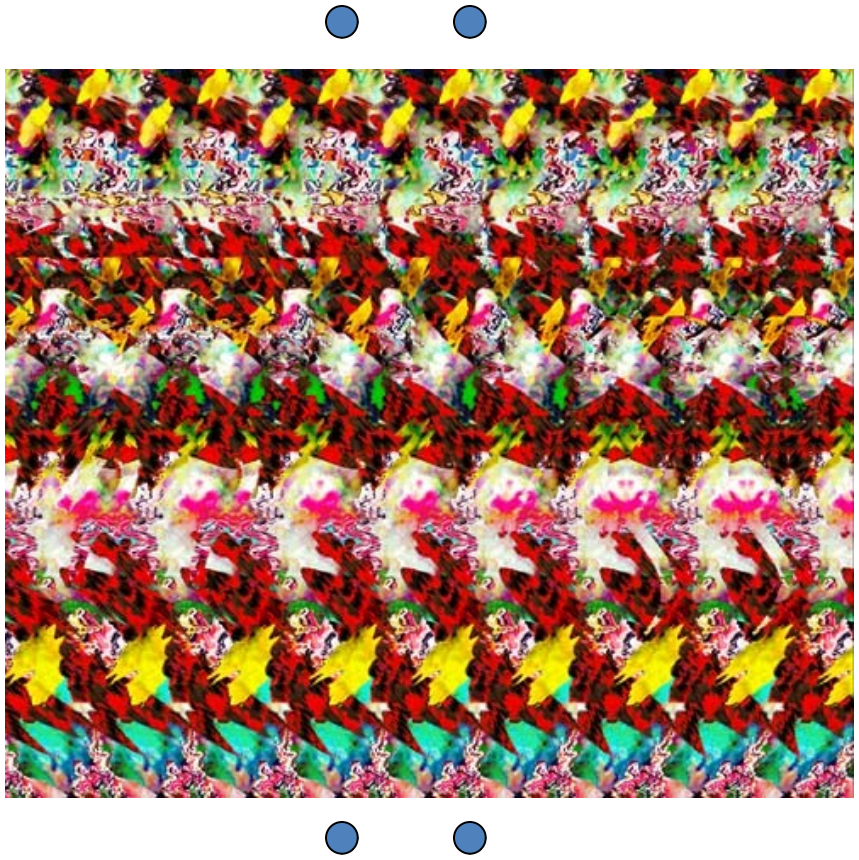
- Human binocular fusion (unified imaginary view of the world) is not directly associated with the physical retinas; must involve the central nervous system (V2, for instance).

- The Imaginary "*cyclopean retina*" *image* 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’

must watch: https://youtu.be/v8O8Em_RPNg



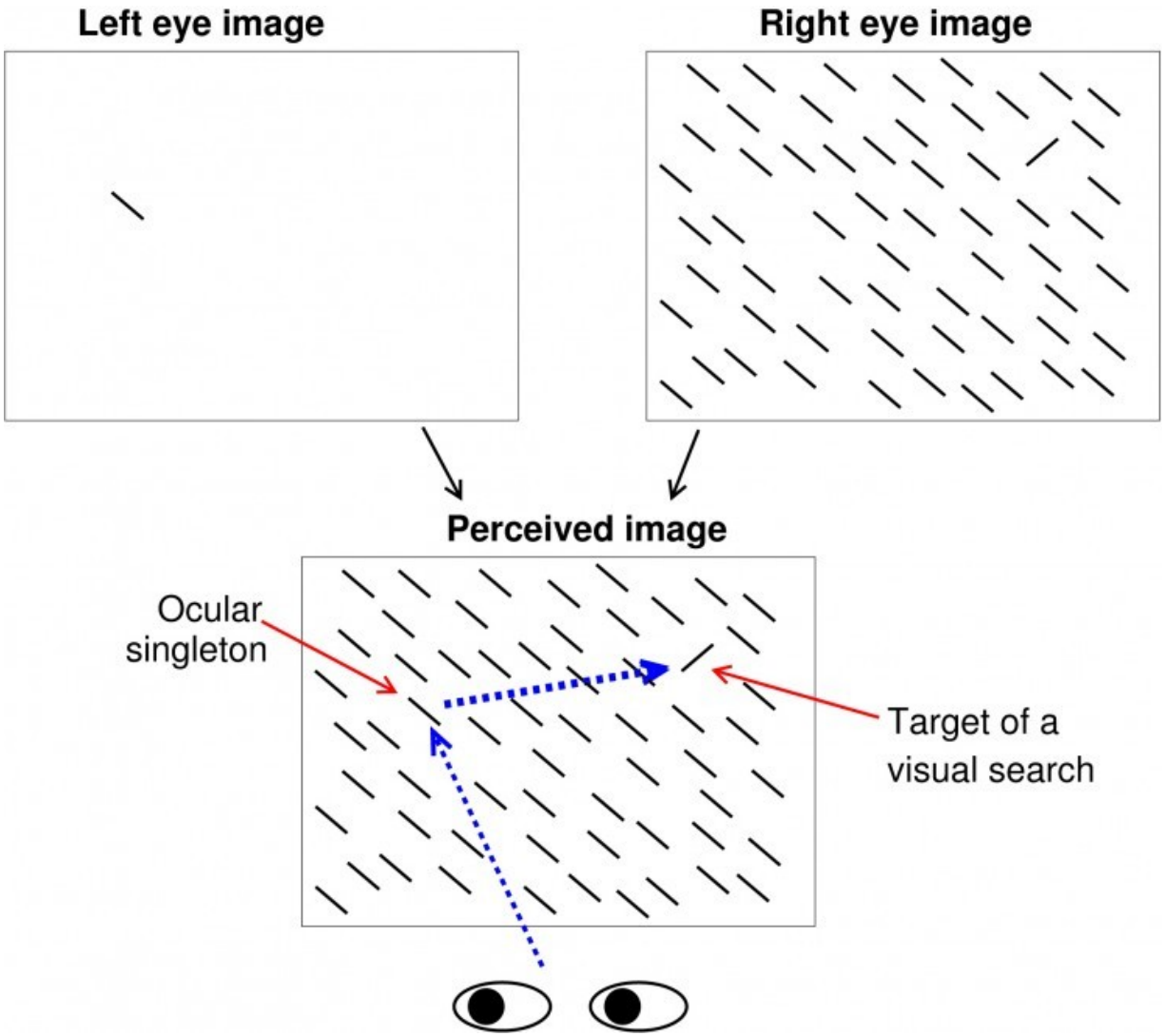
Exploit disparity as depth cue using single image.

(Single image stereogram)

more examples : https://www.youtube.com/watch?v=8LlbUivV-S8

Images from magiceye.com

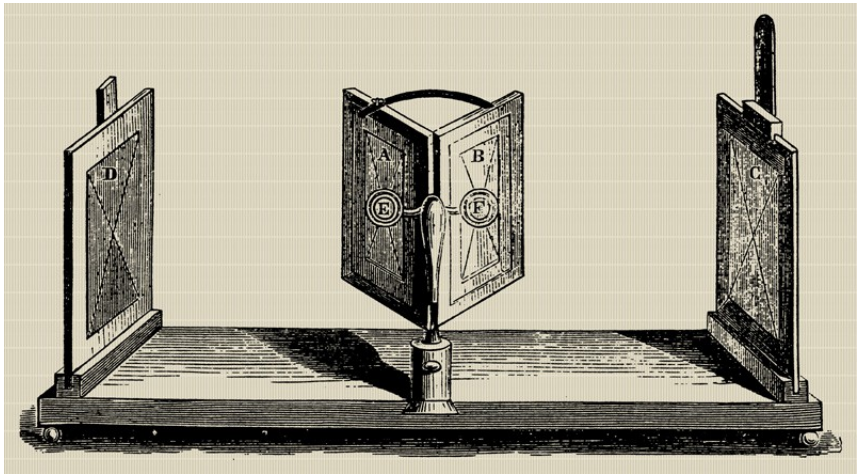
Stereo attention is weird wrt. mind's eye



[Li Zhaoping, Understanding Vision]

Stereo photography and stereo viewers

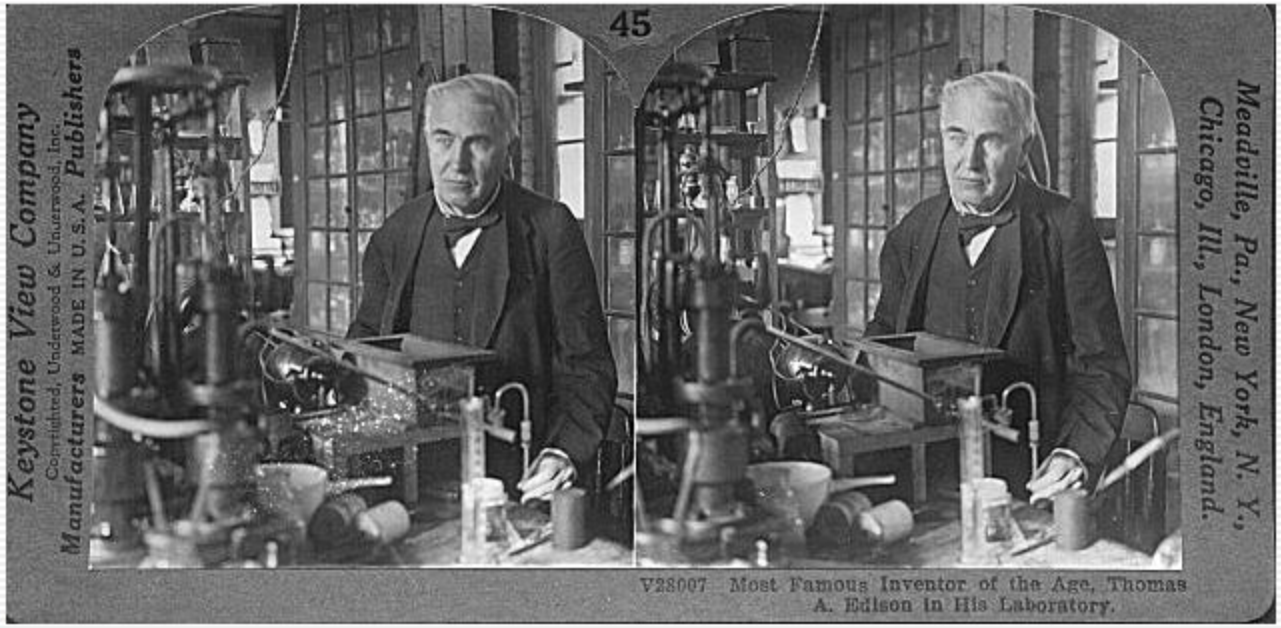
Take two pictures of the same subject from two slightly different viewpoints and display so that each eye sees only one of the images.



Invented by Sir Charles Wheatstone, 1838

Image from fisher-price.com

Anaglyph stereo

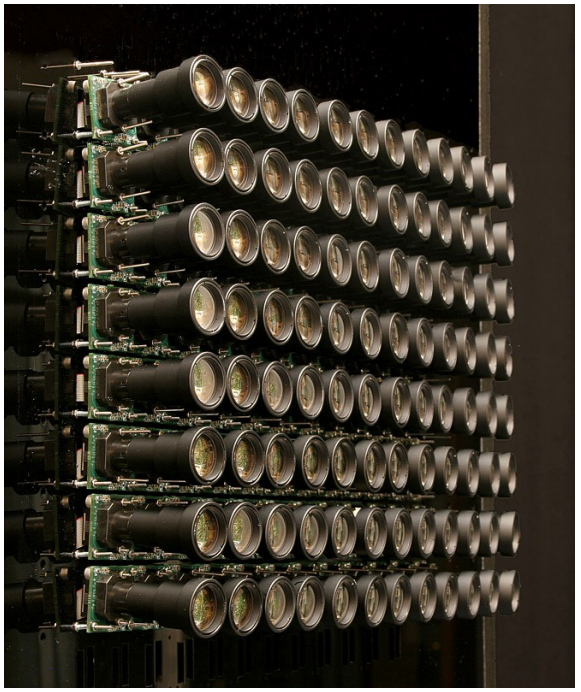


http://www.johnsonshawmuseum.org

Wiggle images

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

Plenoptic/Light Field Cameras



Multiple View Geometry

Why multiple views?

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

Images from Lana Lazebnik

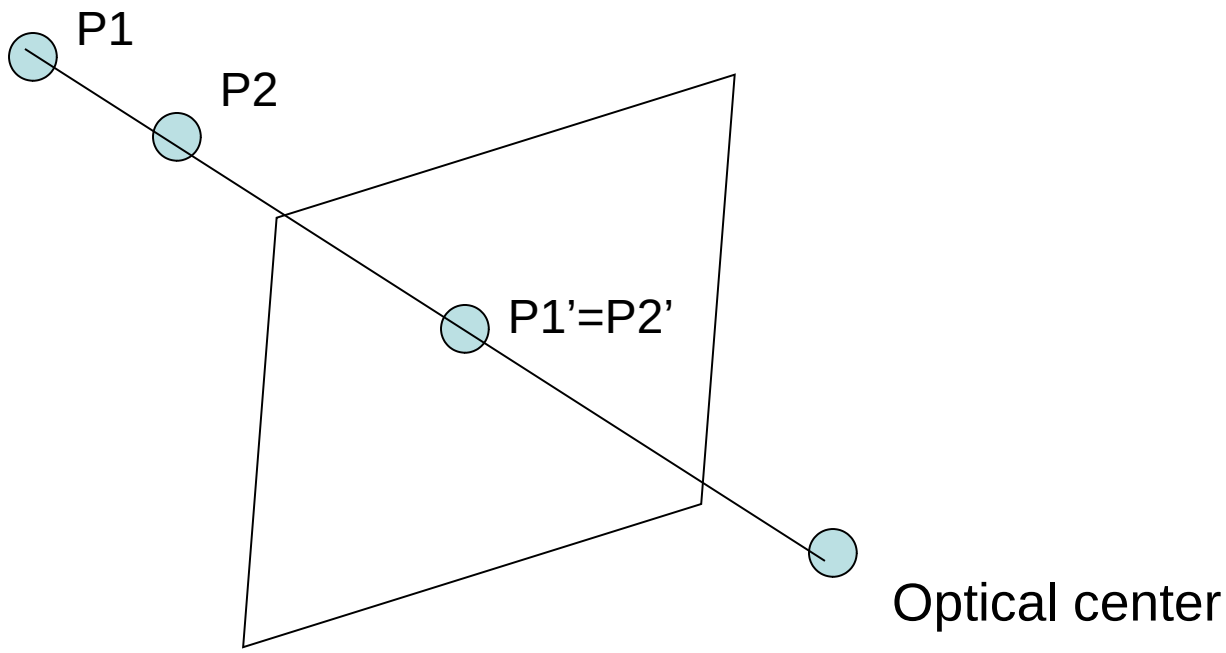
Stereo vision

Two cameras, simultaneous views

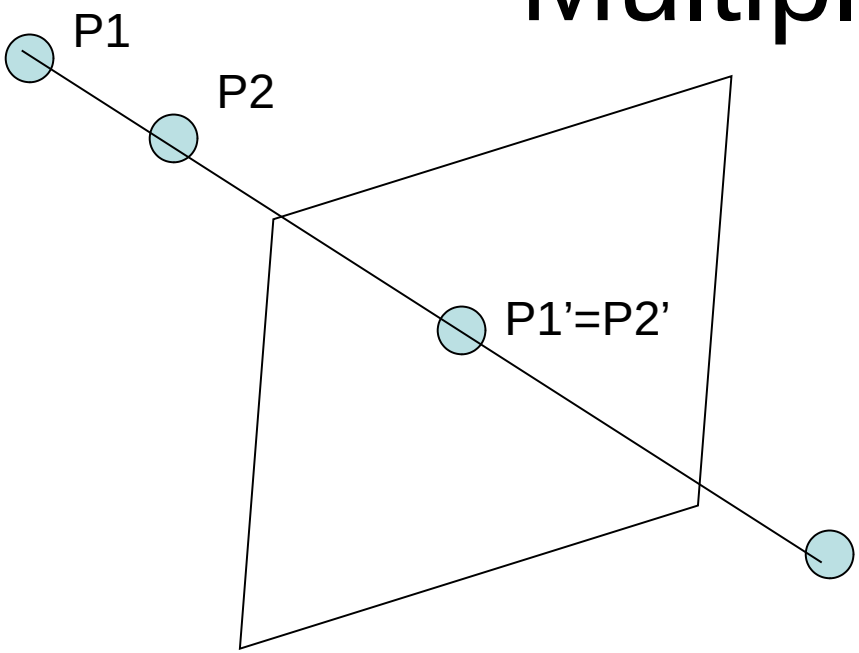
Single moving camera and static scene

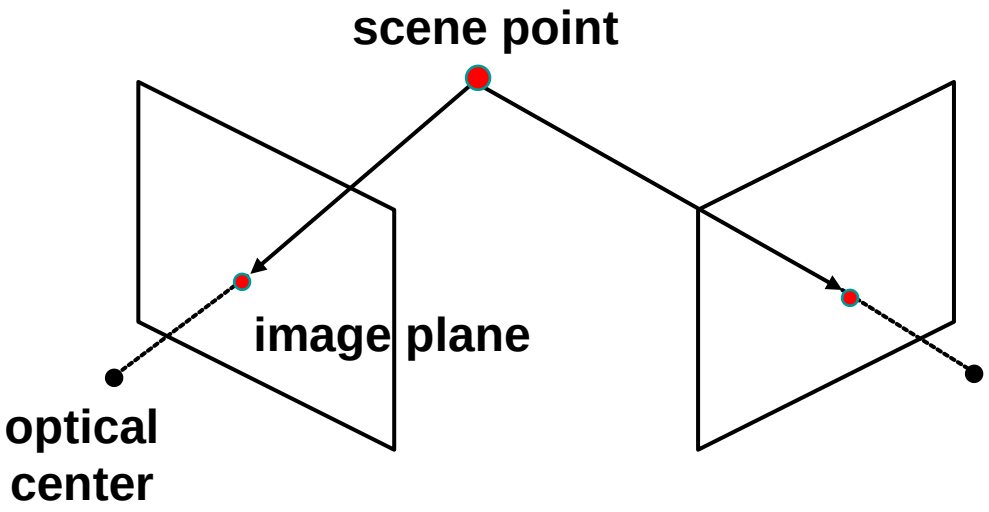
Why multiple views?

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



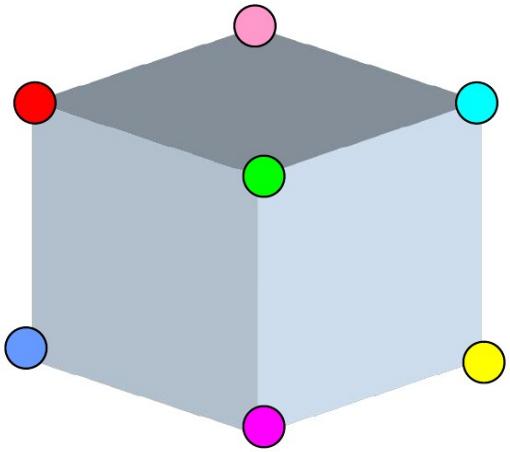
Multiple views

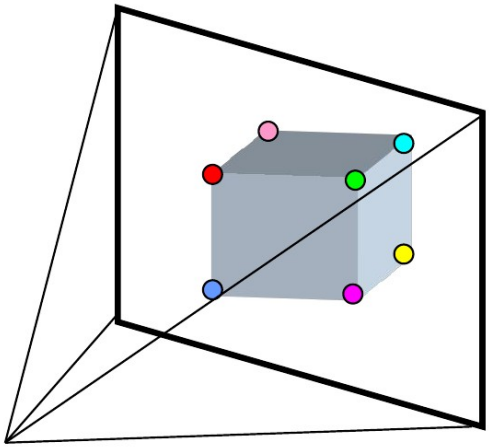




Multi-view geometry problems

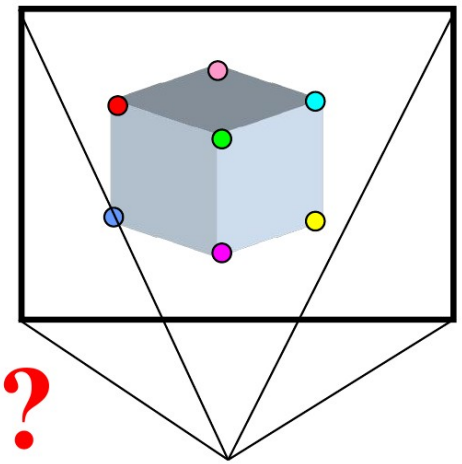
- **Camera 'Motion':** Given a set of corresponding 2D/3D points in two or more images, compute the camera parameters.





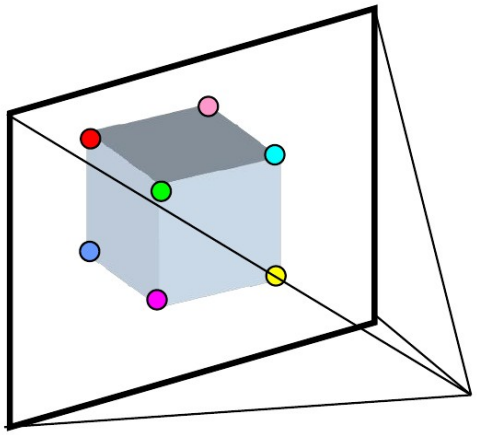
Camera 1

$\mathbf{R}_1,\mathbf{t}_1$?



Camera 2

$\mathbf{R}_2,\mathbf{t}_2$?



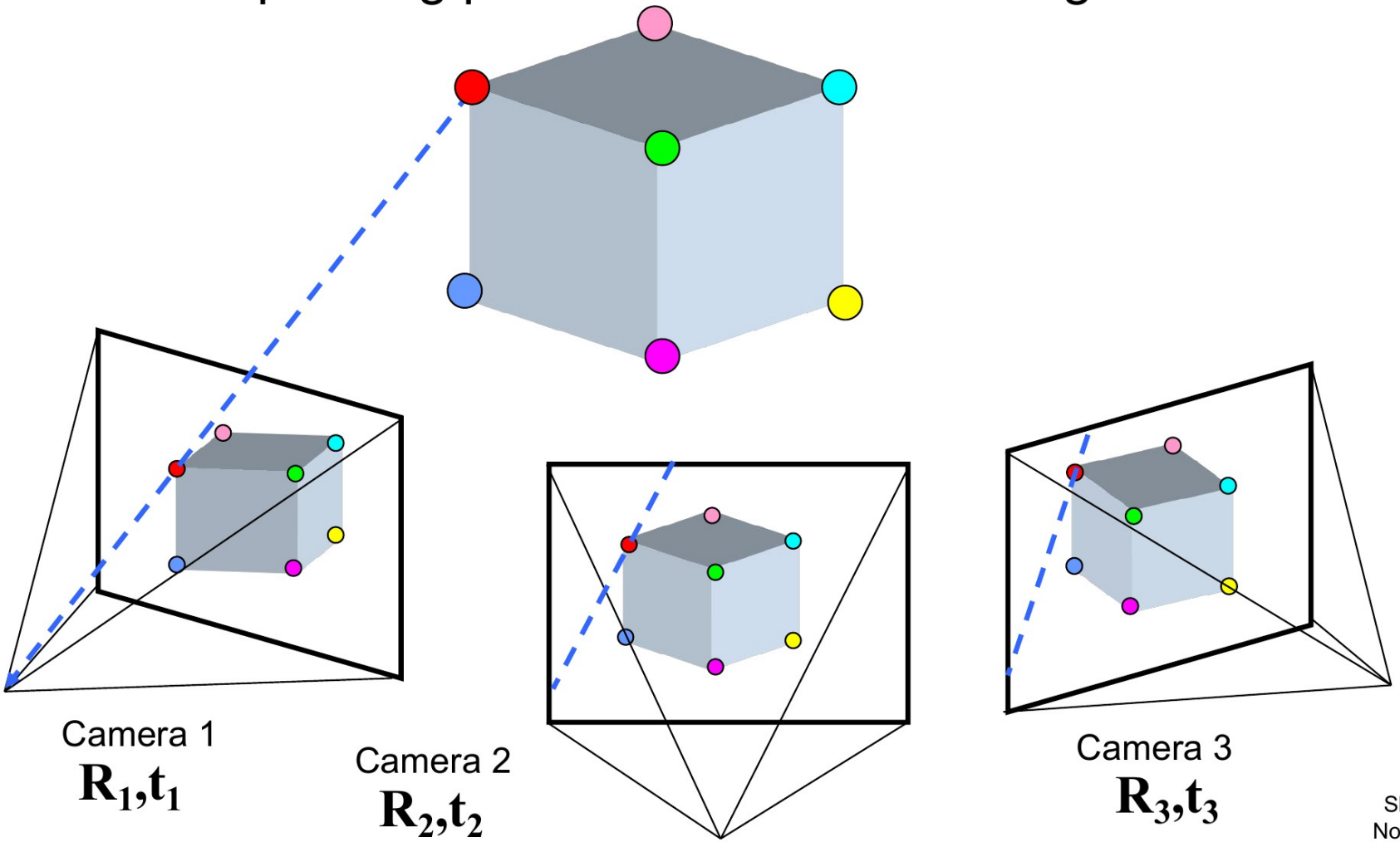
Camera 3

$\mathbf{R}_3,\mathbf{t}_3$?

Slide credit:
Noah Snavely

Multi-view geometry problems

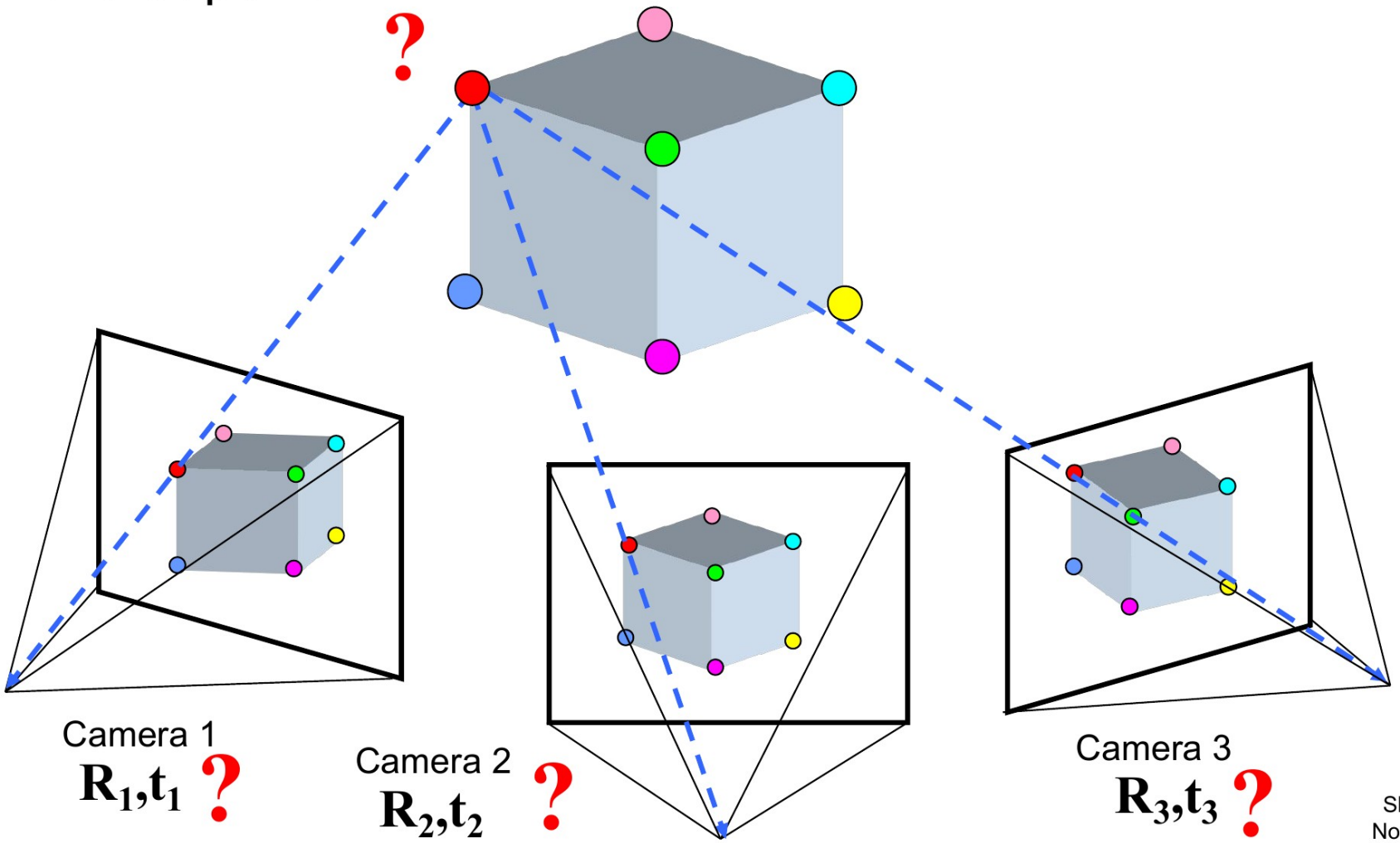
- **Stereo correspondence:** Given known camera parameters and a point in one of the images, where could its corresponding points be in the other images?



Slide credit:
Noah Snavely

Multi-view geometry problems

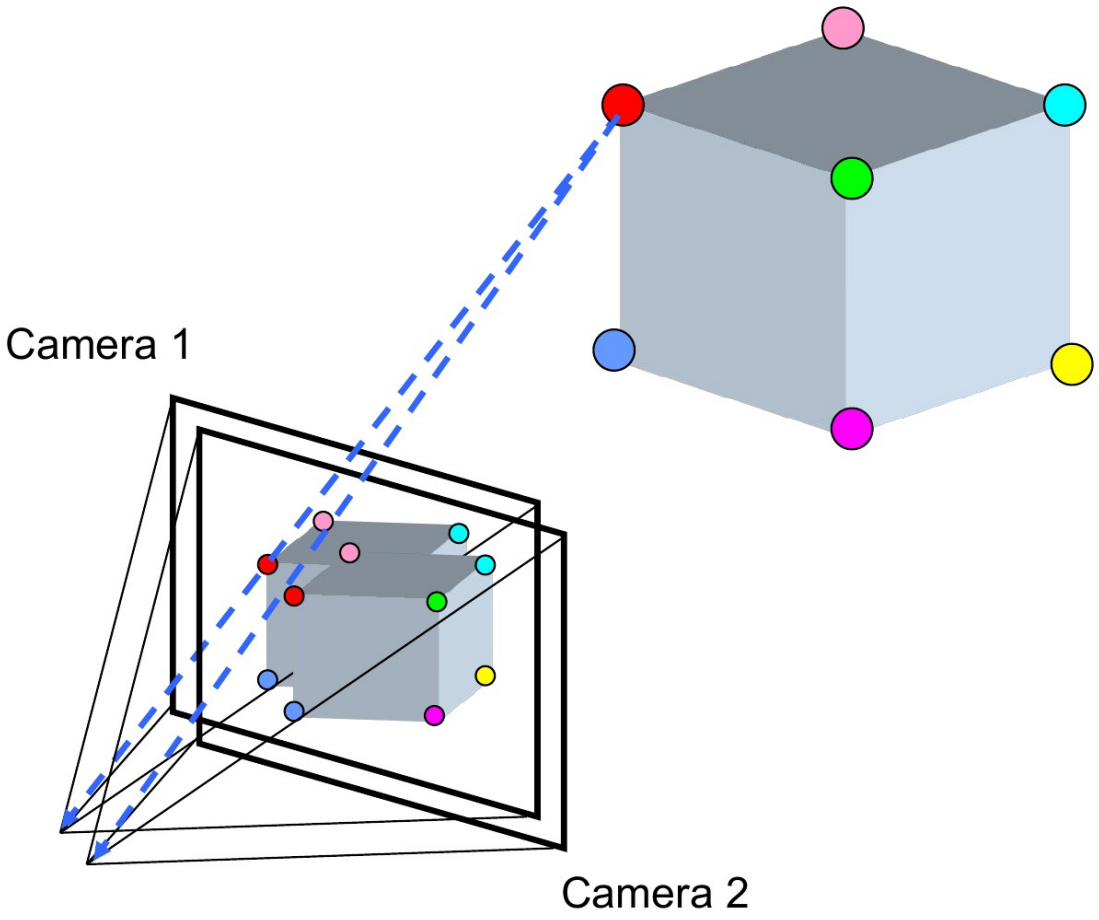
- **Structure from Motion:** Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point



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.

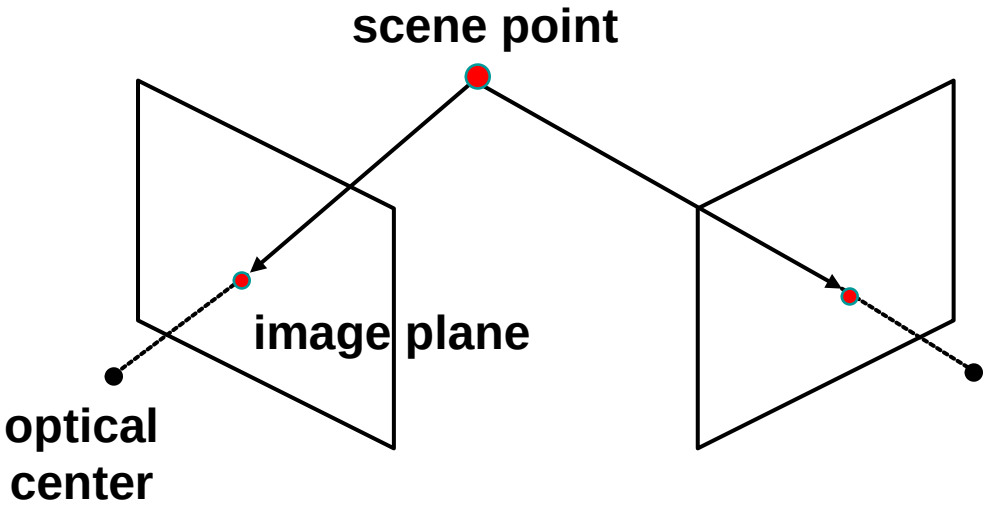


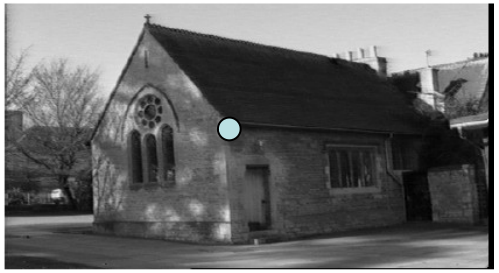
Estimating depth with stereo

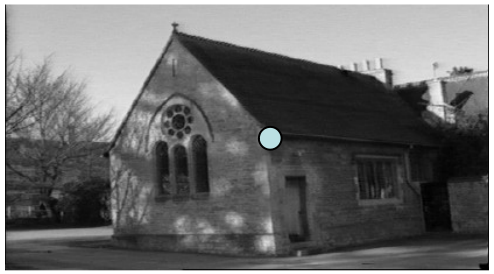
Stereo: shape from “change” between two views

We’ll need to consider:

- Info on camera pose
- Ideally intrinsics/calibration
- Image point correspondences





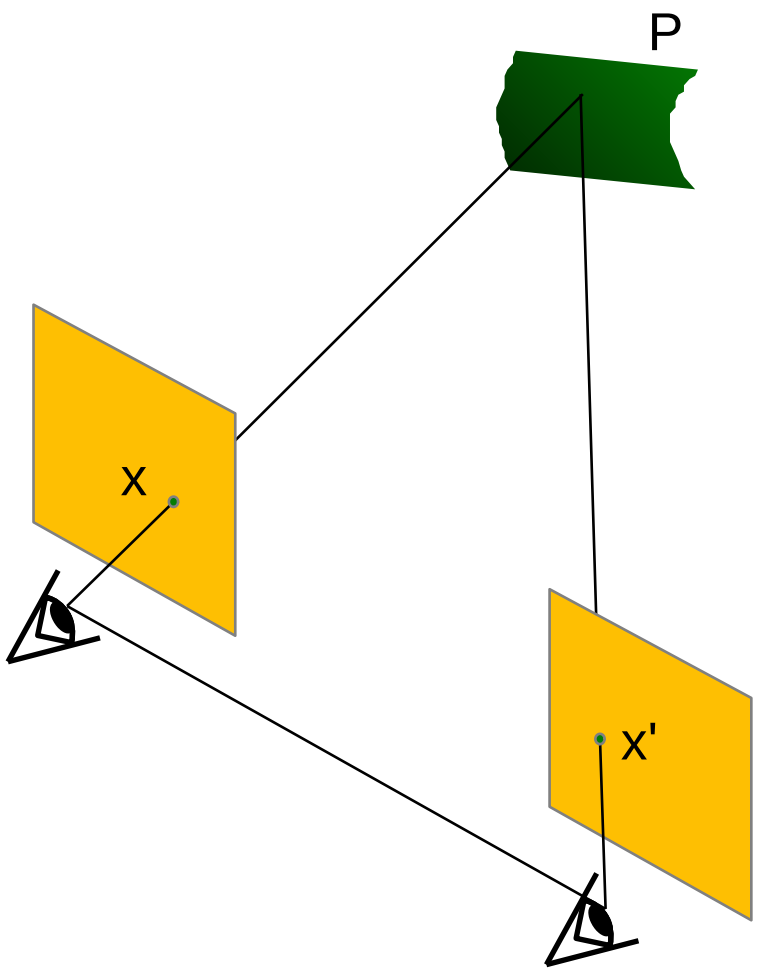


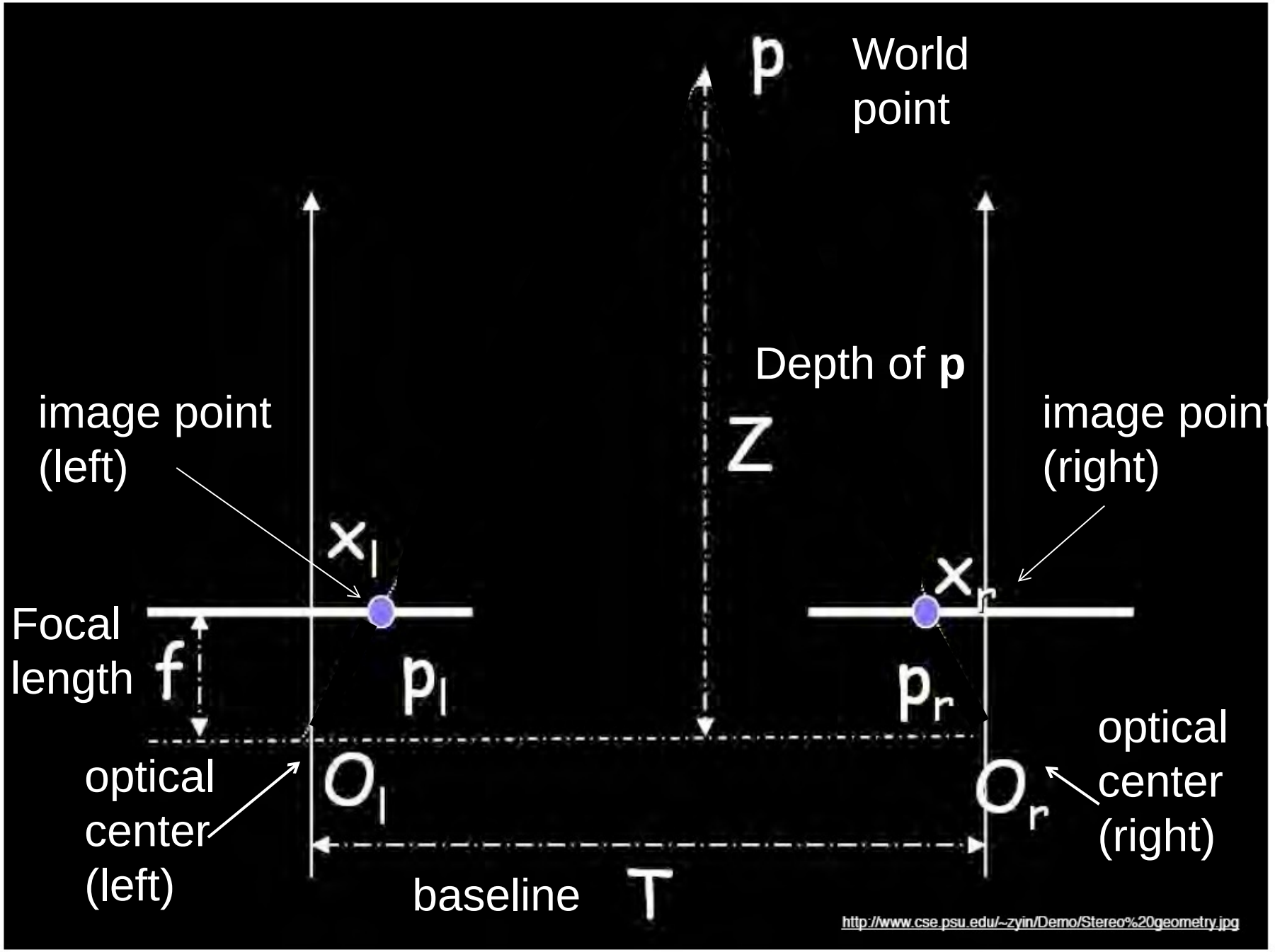
Geometry for a simple stereo system

Assume:

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

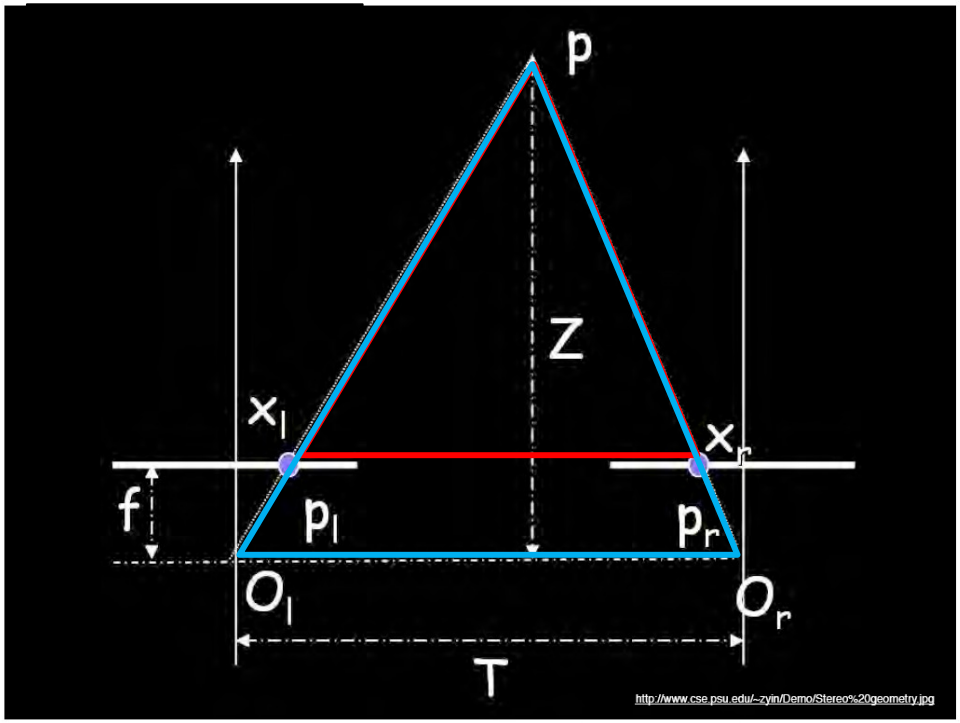
Goal: recover depth of Z by finding image coordinate x’ that corresponds to x




p
World point
Depth of p
Z
image point (left)
image point (right)
x_l
x_r
Focal length
f
p_l
p_r
optical center (left)
O_l
O_r
optical center (right)
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)

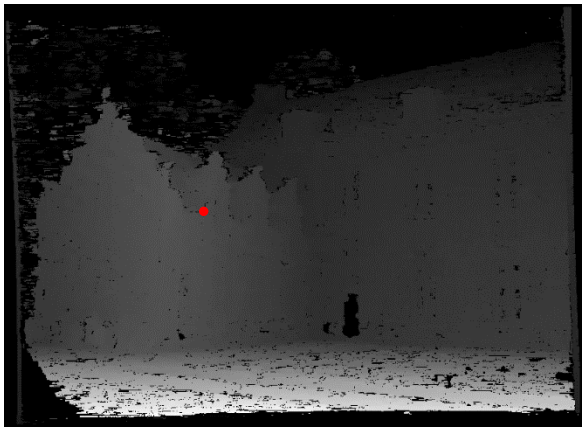


image I´(x´,y´)

$(x´,y´)=(x+D(x,y), y)$

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

What do we need to know?

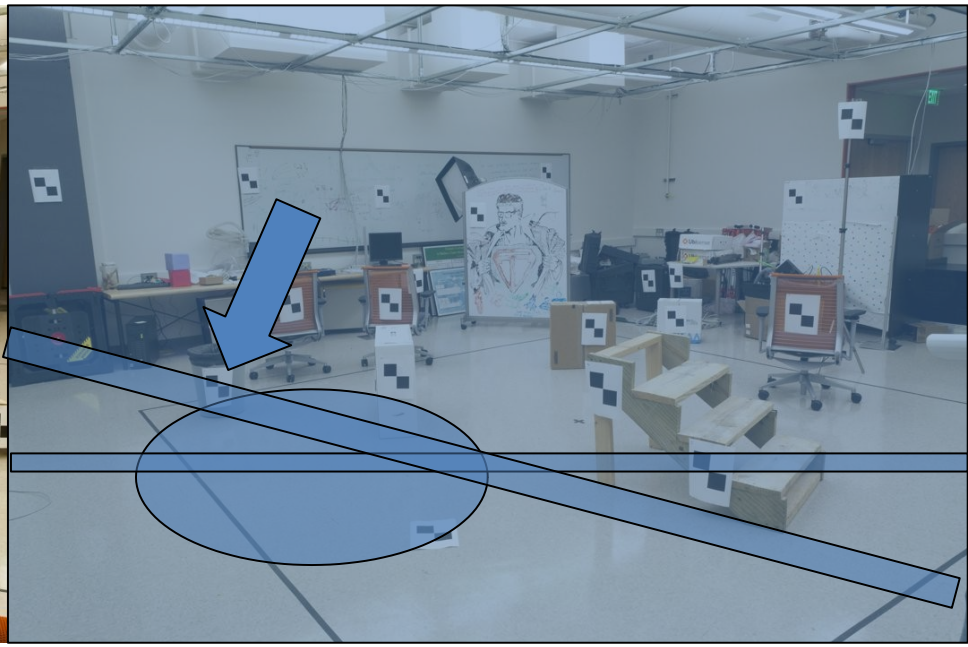
1. Calibration for the two cameras.
    1. 1. Intrinsic matrices for both cameras (e.g., f)
    2. 2a. Baseline distance T in parallel camera case
    3. 2b. R, t in non-parallel case

2. Correspondence for every pixel:

Like feature matching in assignment 2, but assignment 2 is “sparse”.

We need “dense” correspondence!

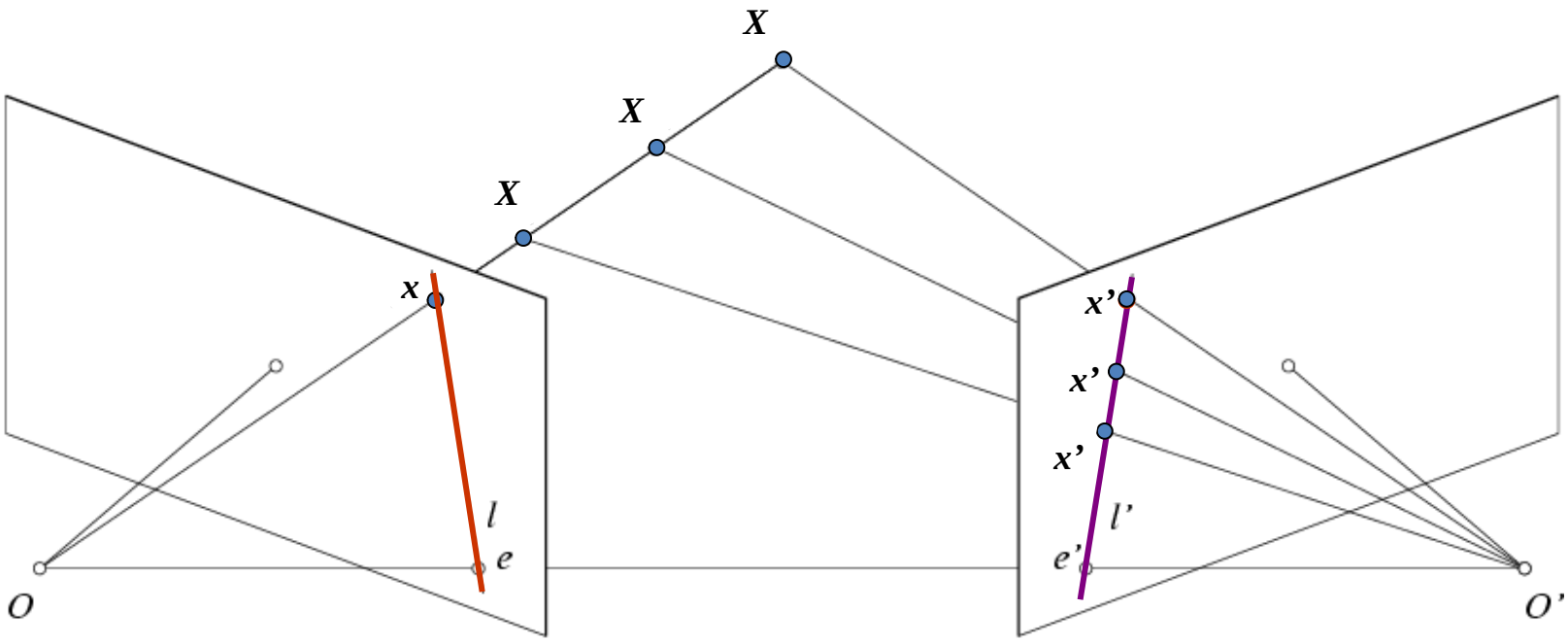
Correspondence for every pixel. Where do we need to search?



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

Epipolar geometry
Constrains 2D search to 1D

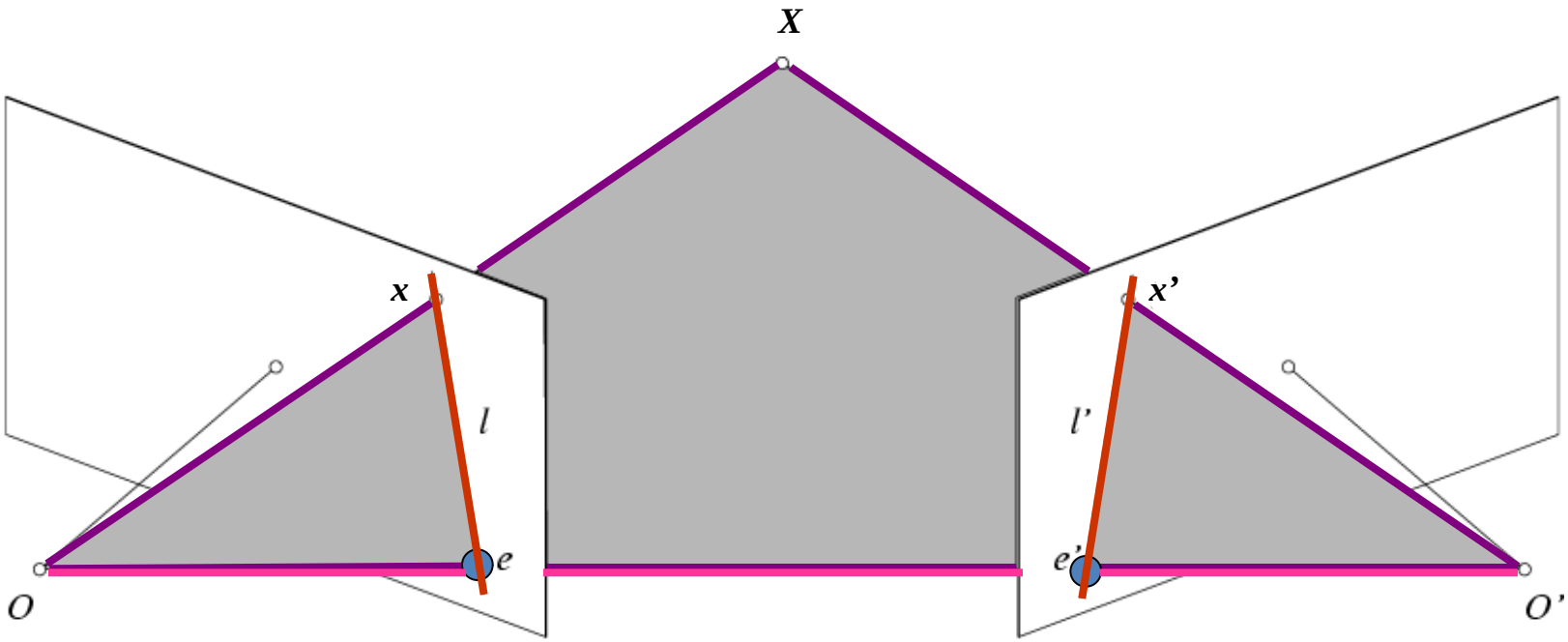
Key idea: Epipolar constraint



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



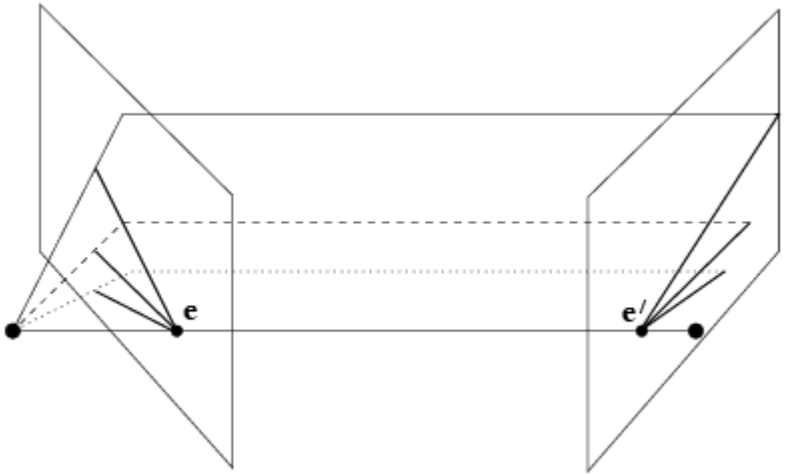
Baseline – line connecting the two camera centers

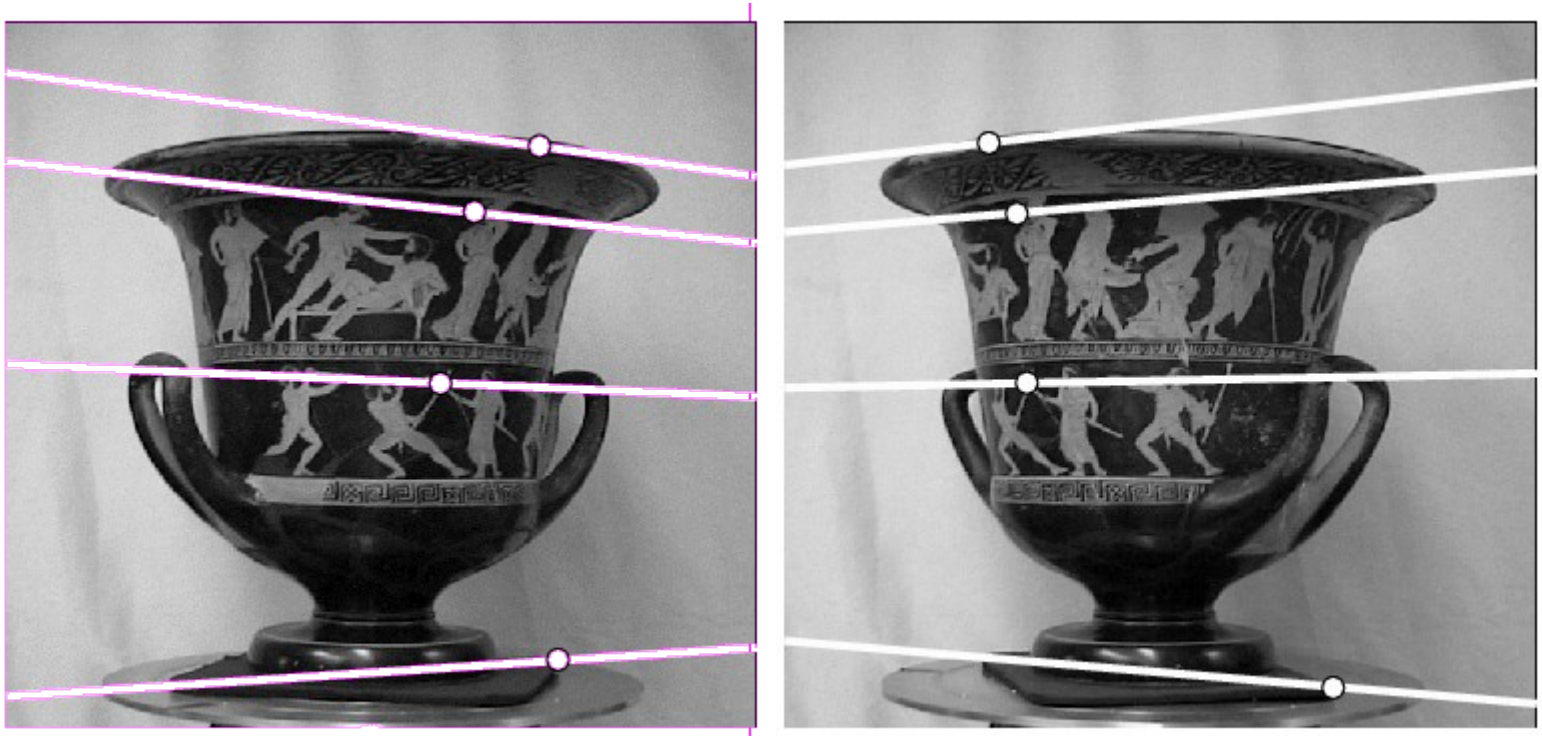
Epipoles
= intersections of baseline with image planes
= projections of the other camera center

Epipolar Plane – plane containing baseline (1D family)

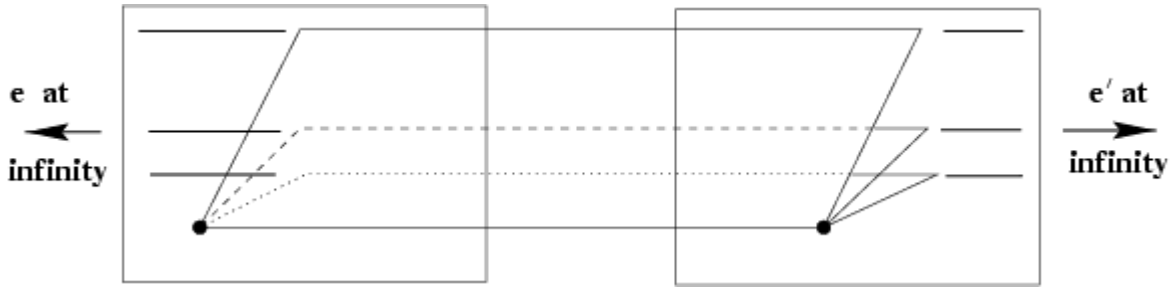
Epipolar Lines - intersections of epipolar plane with image planes (always come in corresponding pairs)

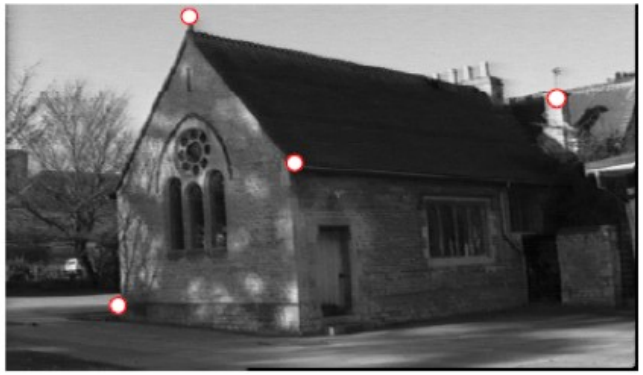
Example: Converging cameras

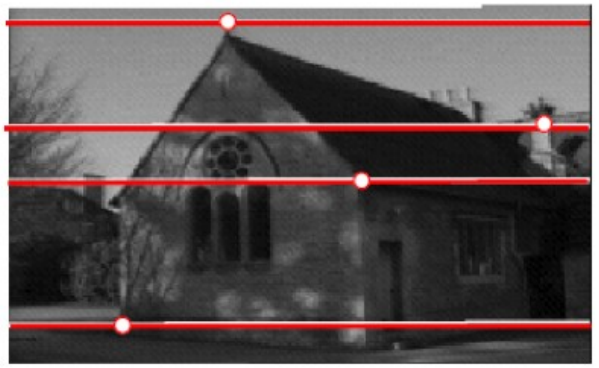




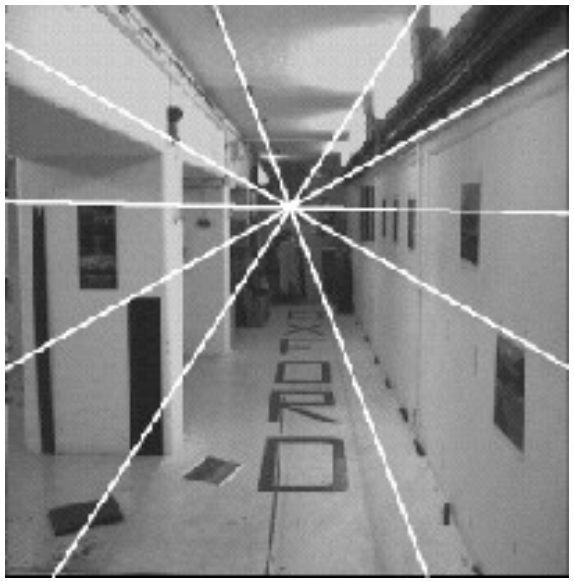
Example: Motion parallel to image plane

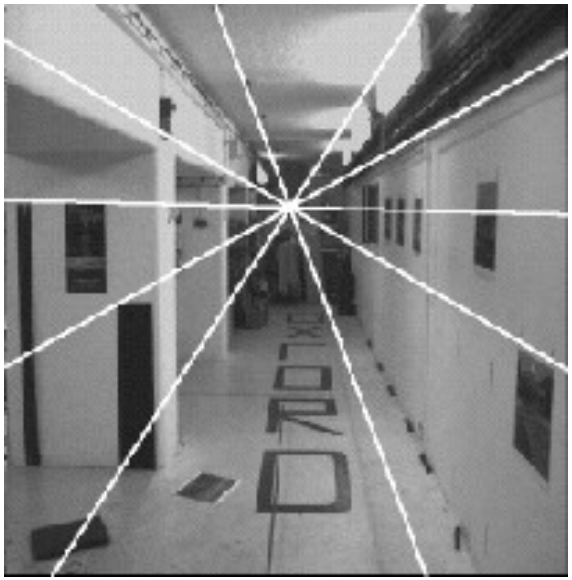


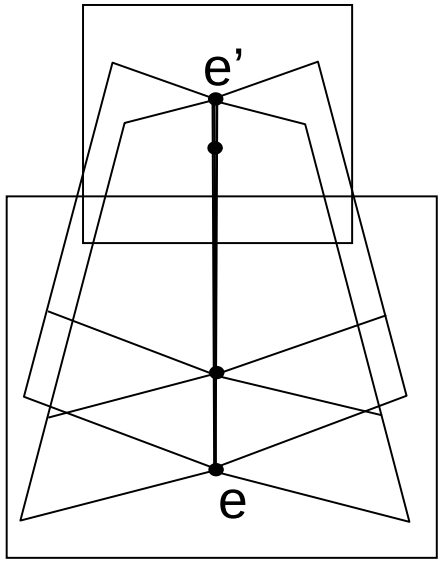




Example: Forward motion

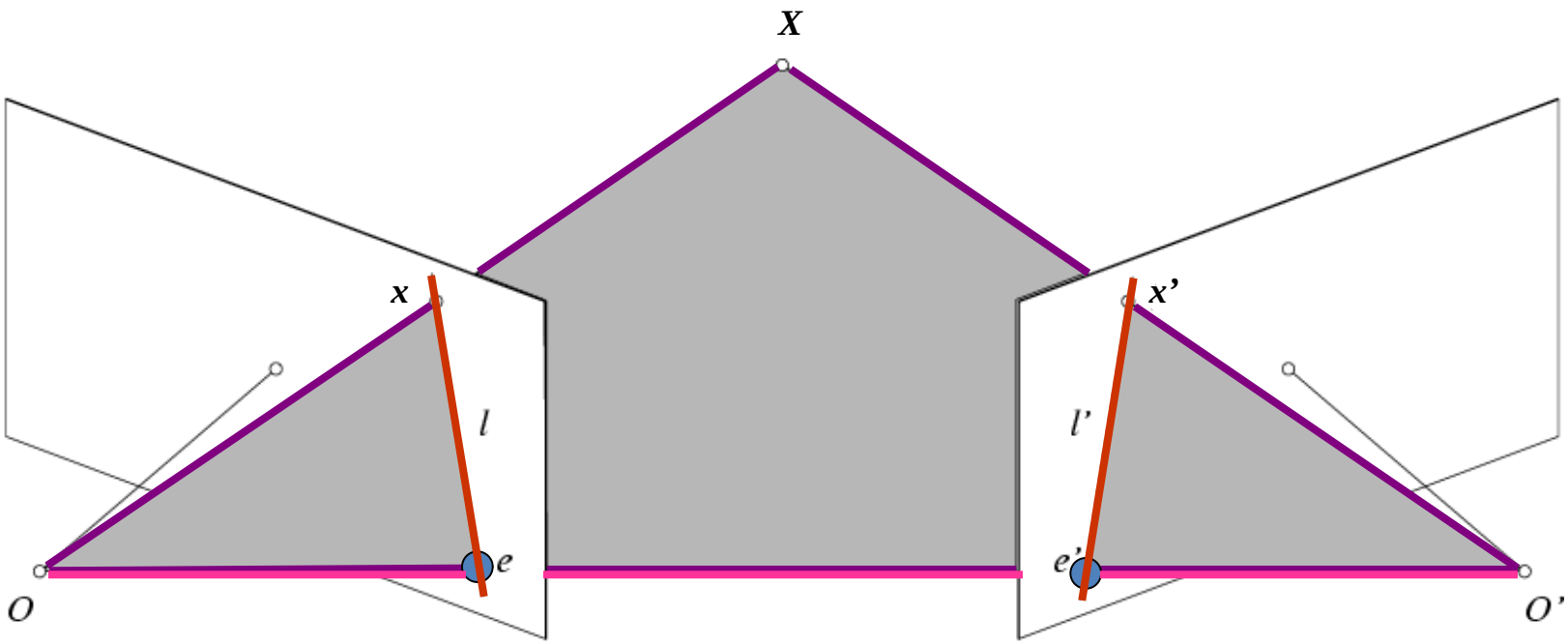






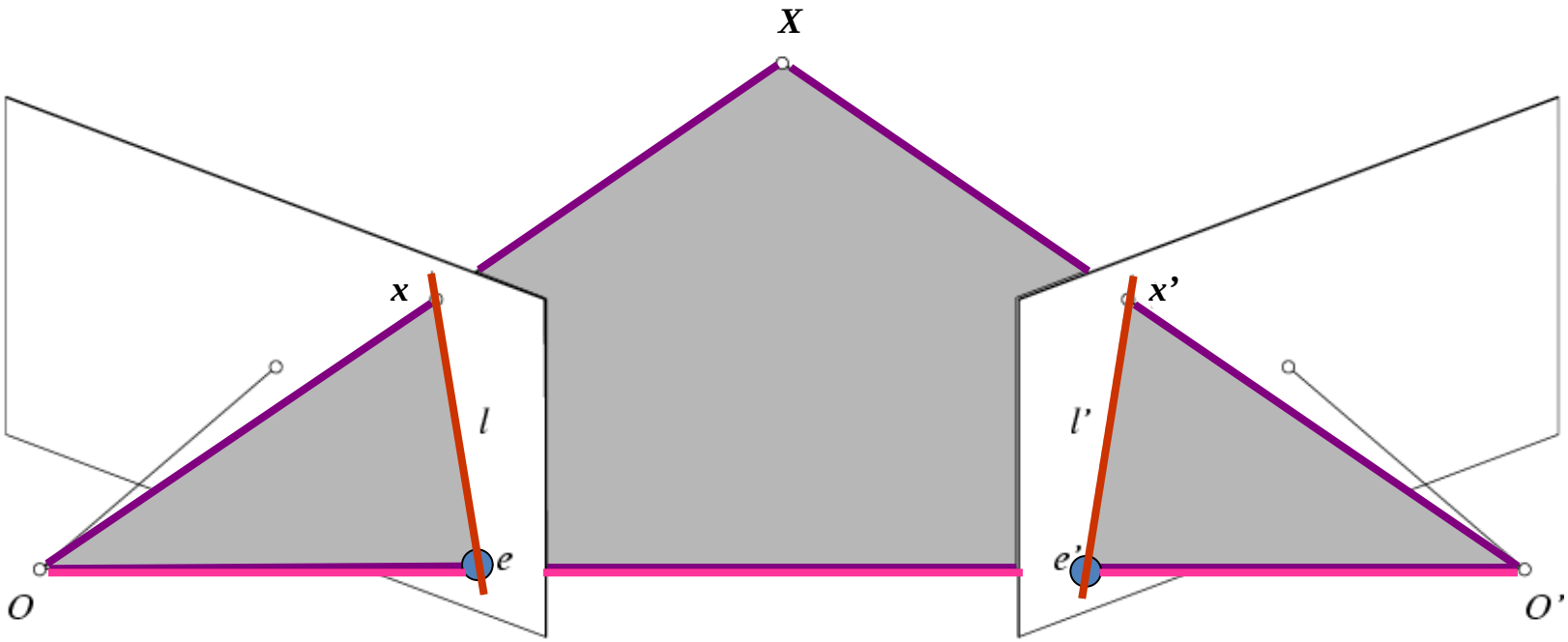
Epipole has same coordinates in both images.
Points move along lines radiating from e: “Focus of expansion”

What is this useful for?



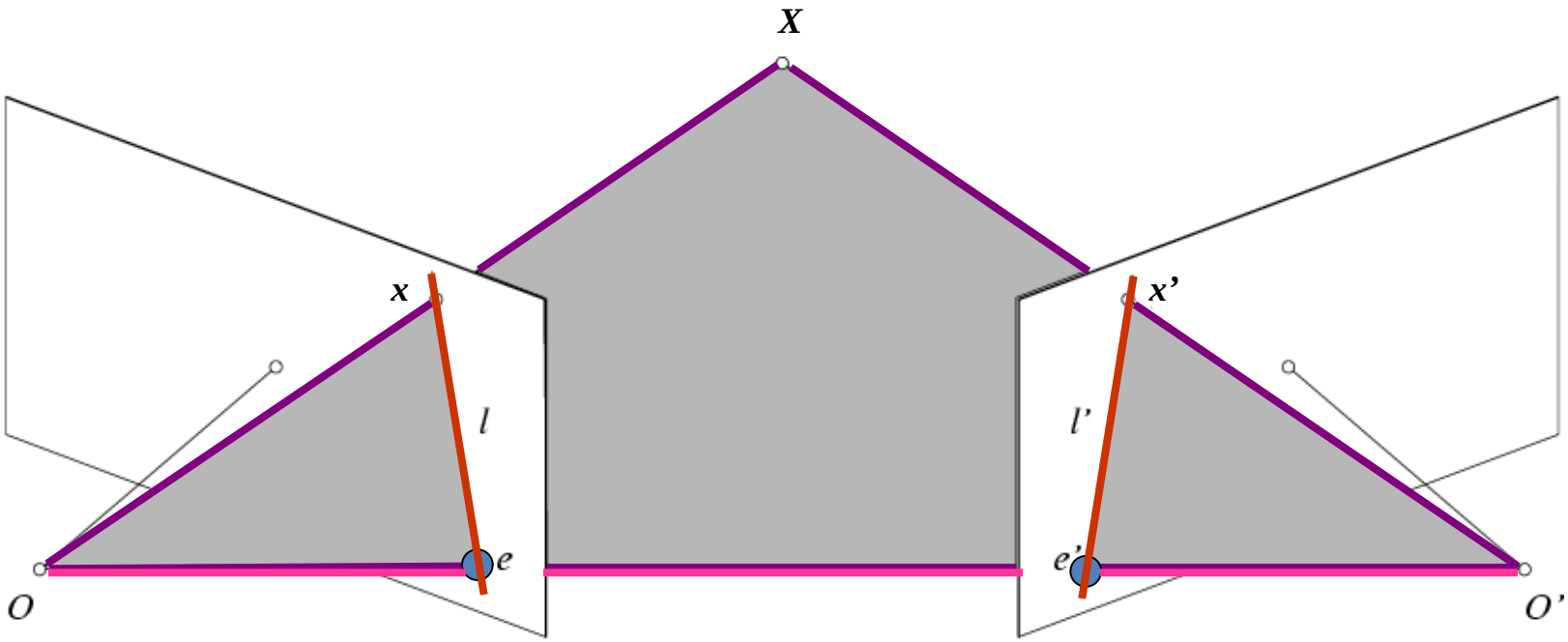
- Reduce search space for stereo disparity estimation.
- Help find *x'*: If I know *x*, and have calibrated cameras (known intrinsics K,K' and extrinsic relationship), I can restrict *x'* to be along *l'*.

What is this useful for?



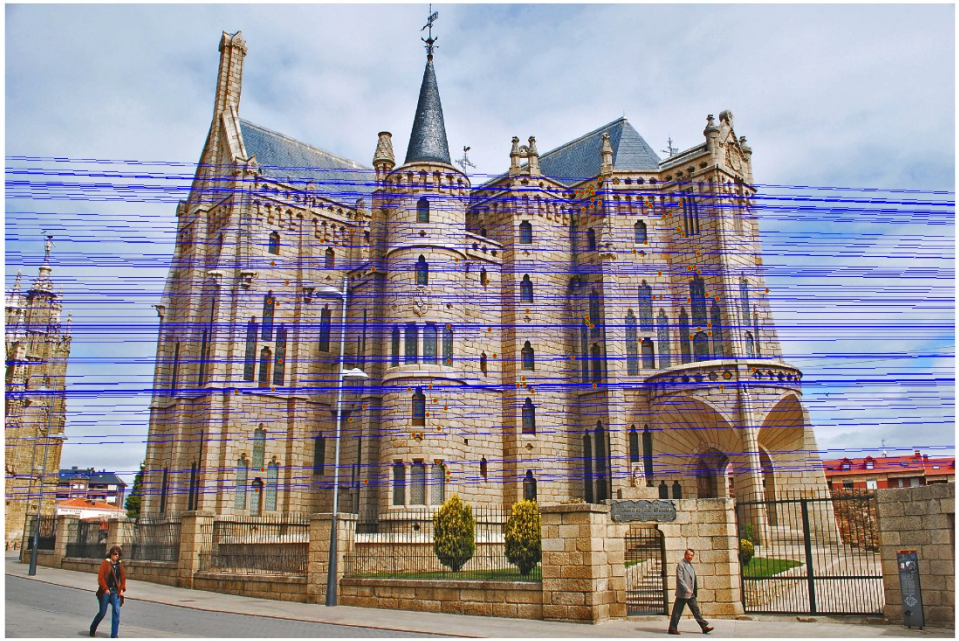
If we have enough *x*, *x'* correspondences, we can estimate relative position and orientation between the cameras and the 3D position of corresponding image points.

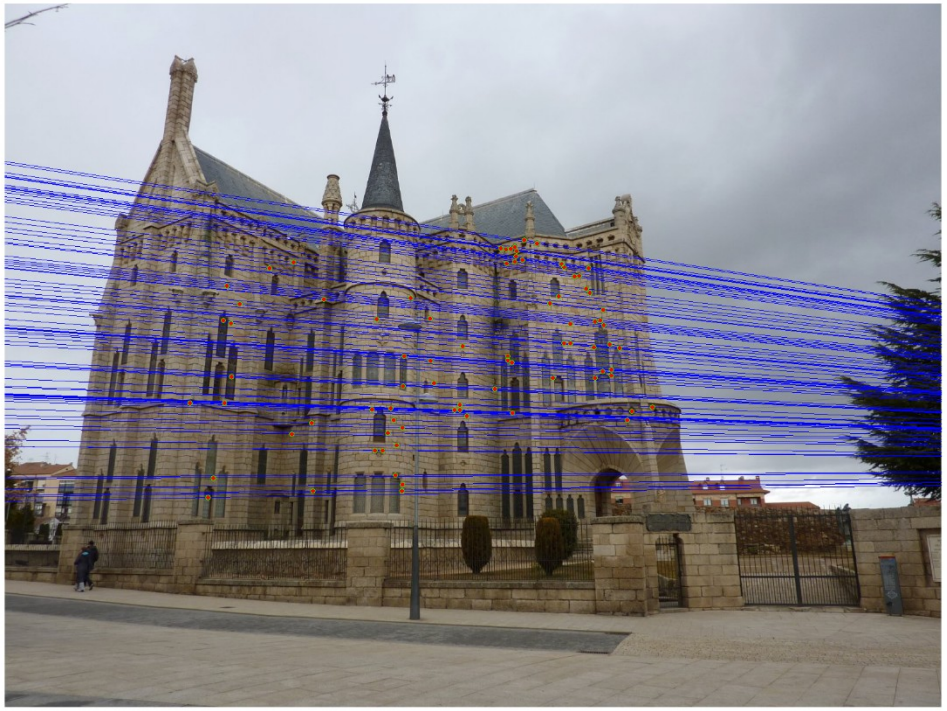
What is this useful for?



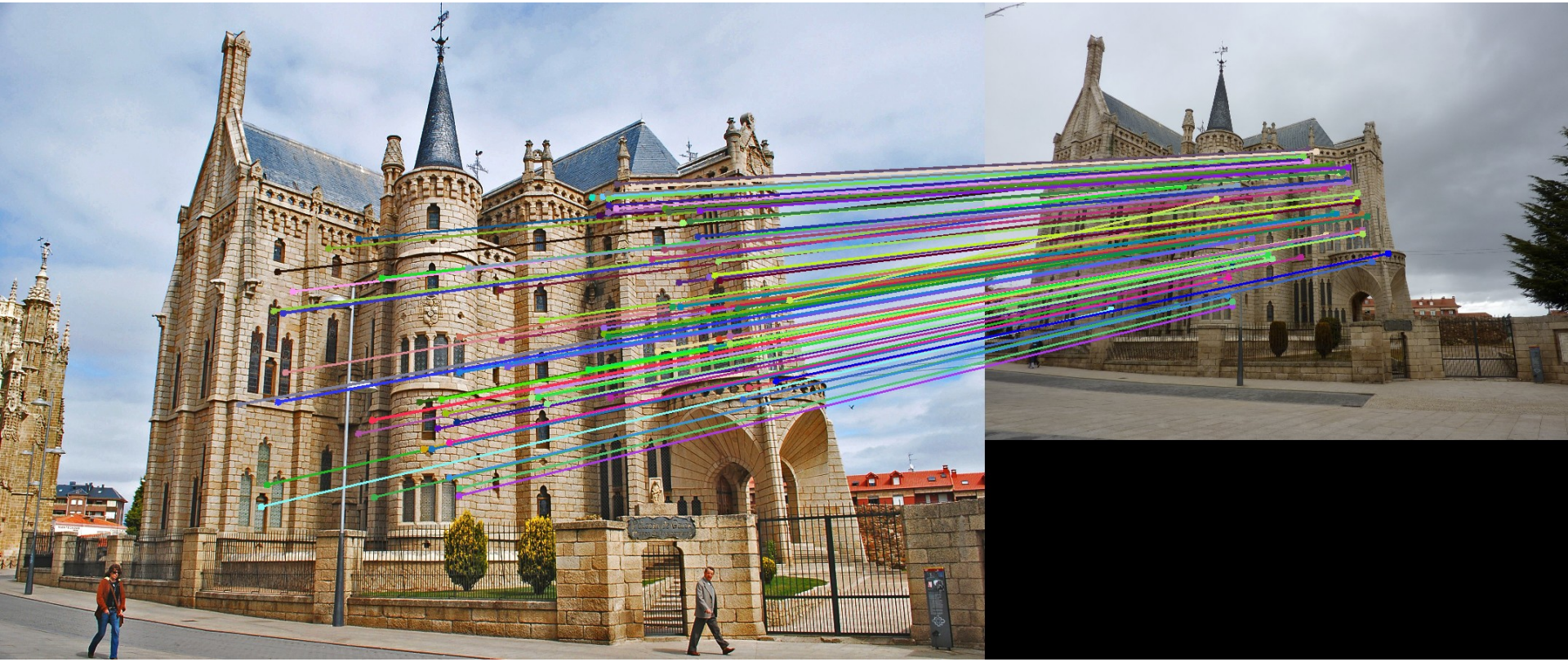
- Model fitting: see if candidate *x*, *x'* correspondences fit estimated projection models of cameras 1 and 2.

Epipolar lines

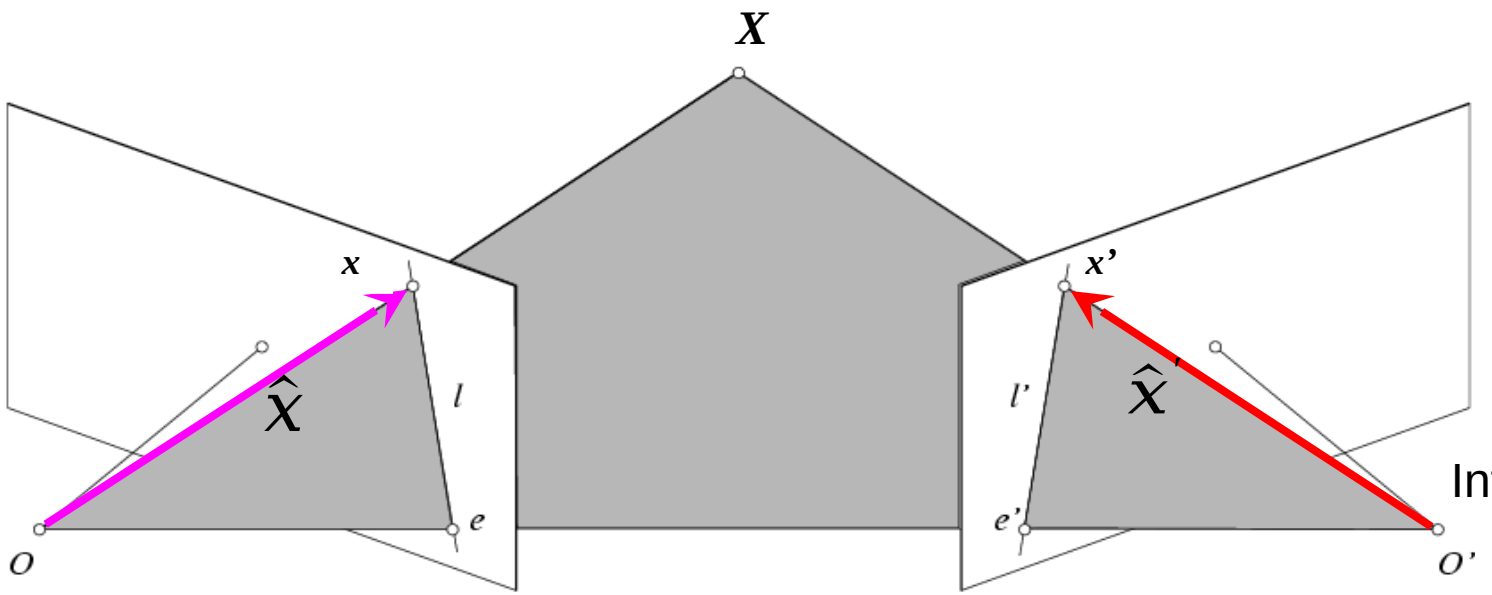




Keep only the matches at are “inliers” with respect to the “best” fundamental matrix



Epipolar constraint: Calibrated case



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

Homogeneous 2d point (3D ray towards X)

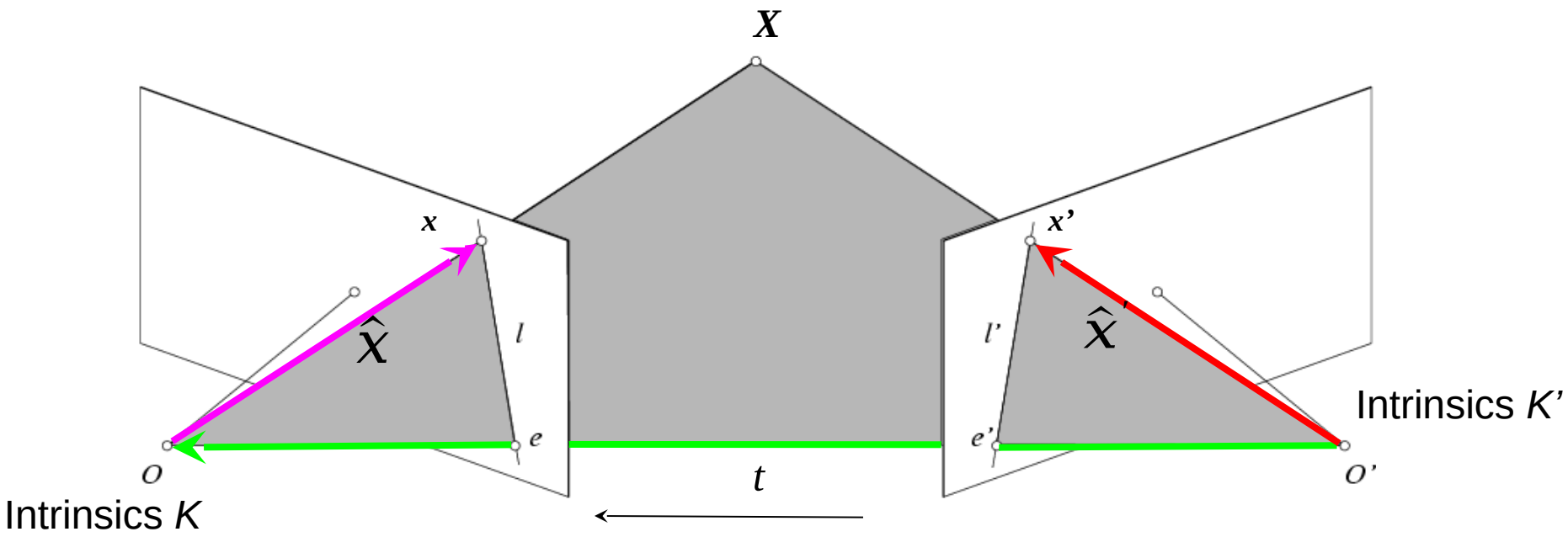
2D pixel coordinate (homogeneous)

3D scene point

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

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

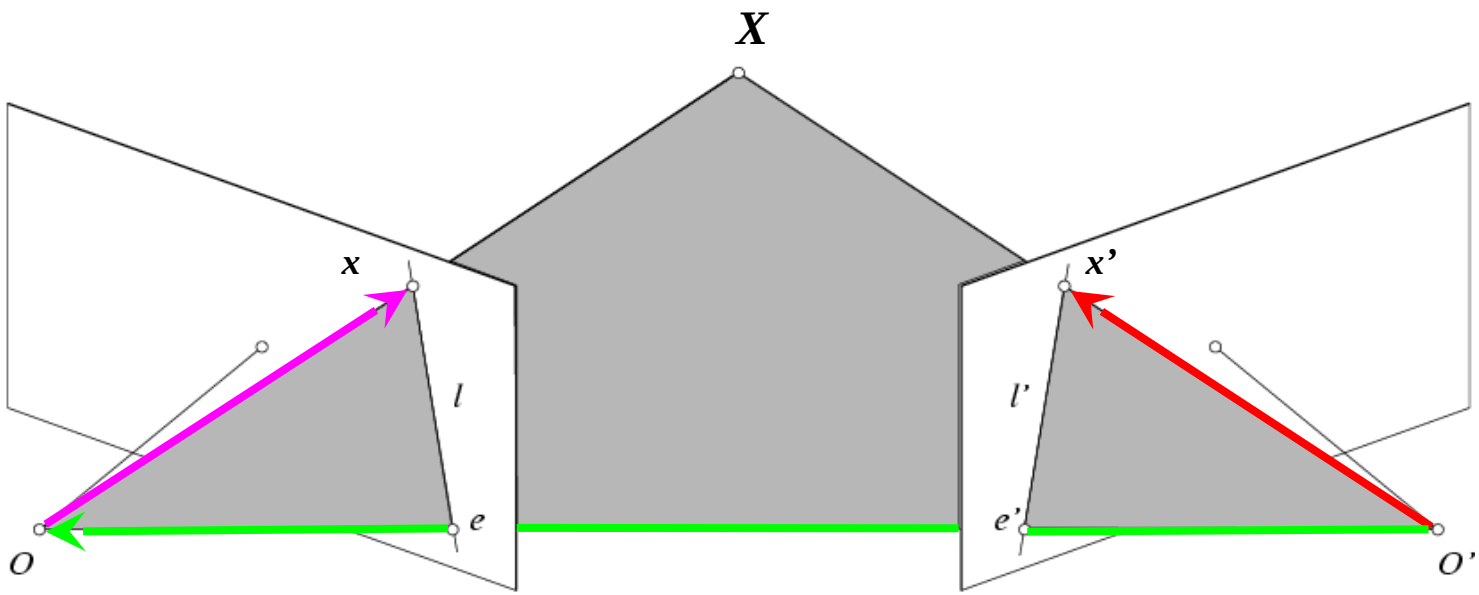
$$\hat{x}' = K'^{-1}x' = X'$$

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

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

(because and are co-planar)

Essential matrix



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

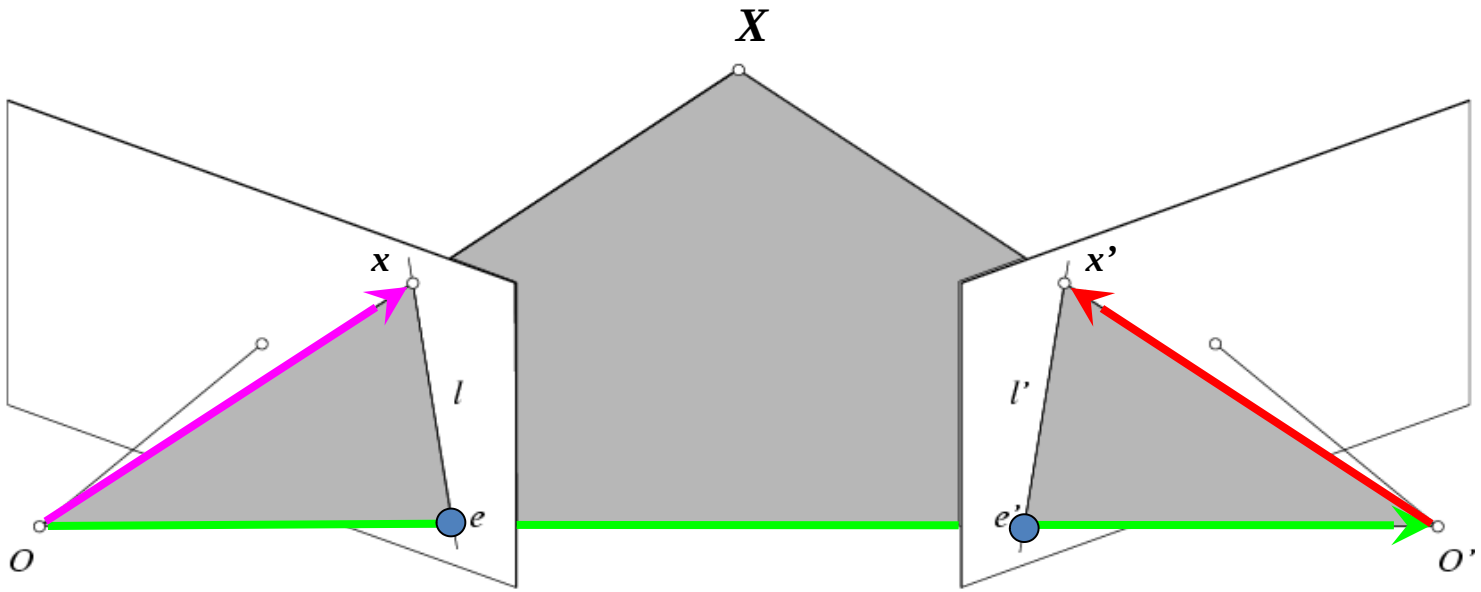
E is a 3x3 matrix which relates corresponding pairs of normalized homogeneous image points across pairs of images – for intrinsic *K* calibrated cameras.

Essential Matrix
(Longuet-Higgins, 1981)

Estimates relative position/orientation.

Note: $[t]_\times$ is the skew symmetrix matrix associated with the vector t

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:

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

The Fundamental Matrix

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

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

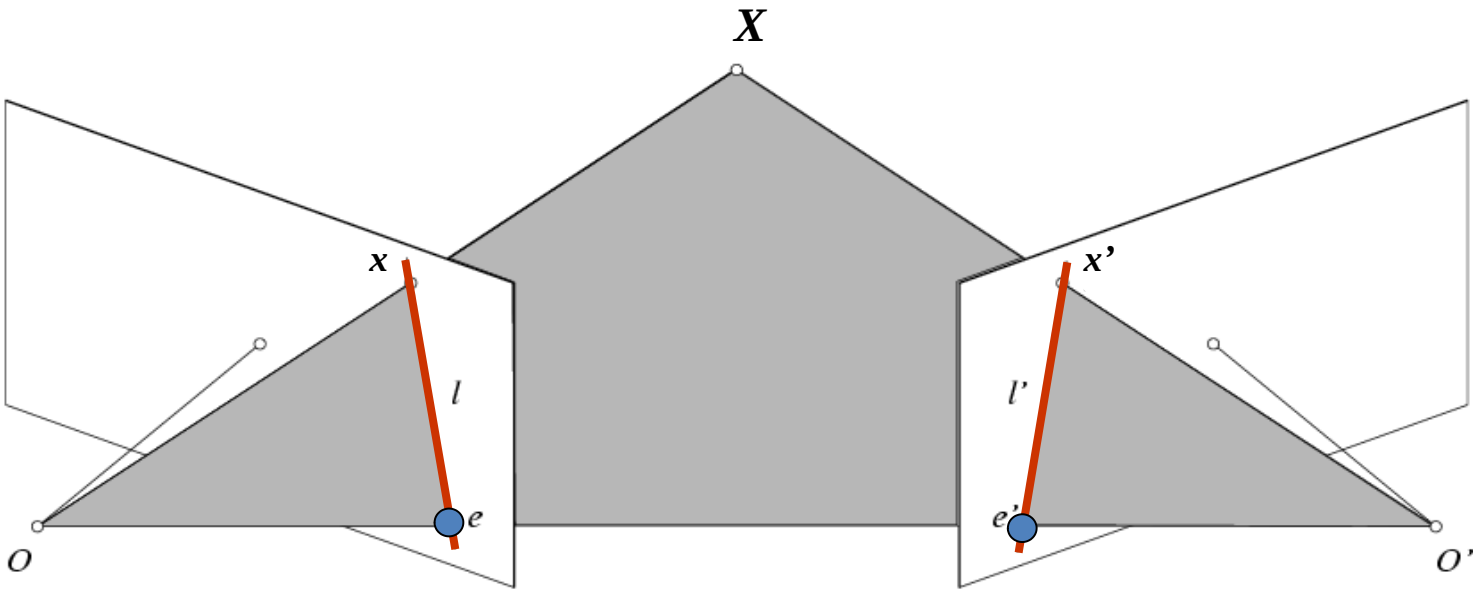
$$\hat{x} = K^{-1} x$$

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

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

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'(when you left multiply by [u v 1])*~~
~~*$F^T x$ = 0 is the epipolar line l' associated with x (when you left multiply by [u' v' 1])*~~

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

Properties of the Fundamental matrix (again), if you want to follow Hartley and Zisserman's book

- $\mathtt{F}$ is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence**: If $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image points, then $\mathbf{x}'^{\mathsf{T}}\mathtt{F}\mathbf{x} = 0$.

Note a slight change in which image is 'first'. I.e. this F is the transpose of the one we called F in the slides

- **Epipolar lines**:
  - $\mathbf{l}' = \mathtt{F}\mathbf{x}$ is the epipolar line corresponding to $\mathbf{x}$.
  - $\mathbf{l} = \mathtt{F}^{\mathsf{T}}\mathbf{x}'$ is the epipolar line corresponding to $\mathbf{x}'$.
- **Epipoles**:
  - $\mathtt{F}\mathbf{e} = \mathbf{0}$.
  - $\mathtt{F}^{\mathsf{T}}\mathbf{e}' = \mathbf{0}$.

If we assume camera 1 is at the world origin:

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

- **Computation from camera matrices** $\mathtt{P}, \mathtt{P}'$:
  - General cameras,
    $\mathtt{F} = [\mathbf{e}']_\times \mathtt{P}'\mathtt{P}^+$, where $\mathtt{P}^+$ is the pseudo-inverse of $\mathtt{P}$, and $\mathbf{e}' = \mathtt{P}'\mathbf{C}$, with $\mathtt{P}\mathbf{C} = \mathbf{0}$.
  - Canonical cameras, $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = [\mathtt{M} \mid \mathbf{m}]$,
    $\mathtt{F} = [\mathbf{e}']_\times \mathtt{M} = \mathtt{M}^{-\mathsf{T}}[\mathbf{e}]_\times$, where $\mathbf{e}' = \mathbf{m}$ and $\mathbf{e} = \mathtt{M}^{-1}\mathbf{m}$.
  - Cameras not at infinity $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$,
    $\mathtt{F} = \mathtt{K}'^{-\mathsf{T}}[\mathbf{t}]_\times \mathtt{R}\mathtt{K}^{-1} = [\mathtt{K}'\mathbf{t}]_\times \mathtt{K}'\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}\mathtt{K}^{\mathsf{T}}[\mathtt{K}\mathtt{R}^{\mathsf{T}}\mathbf{t}]_\times$.

F summary

- 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

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 - use Homography instead!.

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:

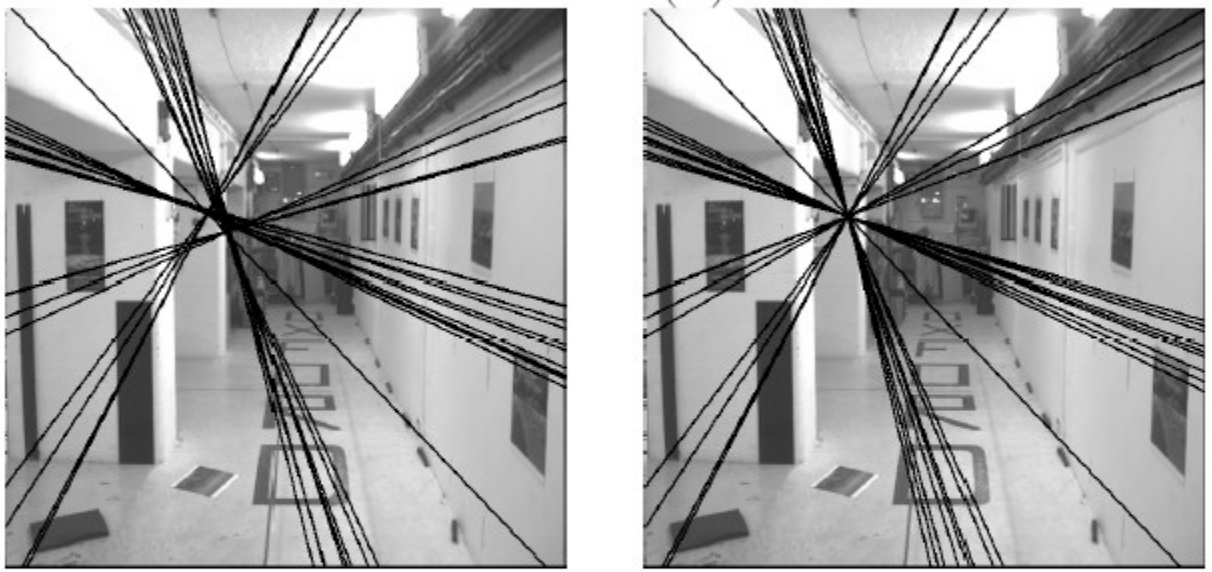
```
[U, S, V] = svd(A);
f = V(:, end);
F = reshape(f, [3 3])';
```

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 : $\det(\mathbf{F}) = 0$.



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

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

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)

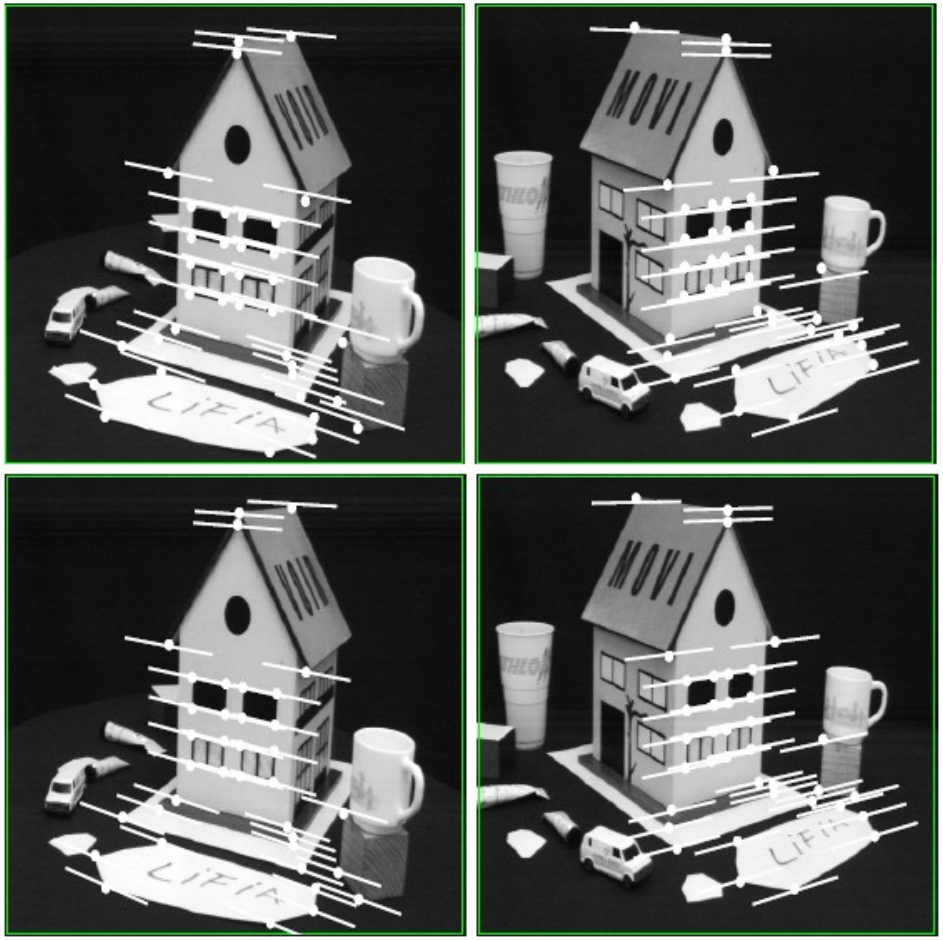
Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels

Use the eight-point algorithm to compute ***F*** from the normalized points

Enforce the rank-2 constraint (for example, take SVD of ***F*** and throw out the smallest singular value)

Transform fundamental matrix back to original units: if ***T*** and ***T'*** are the normalizing transformations in the two images, than the fundamental matrix in original coordinates is $\boldsymbol{T'^{T} F T}$

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…

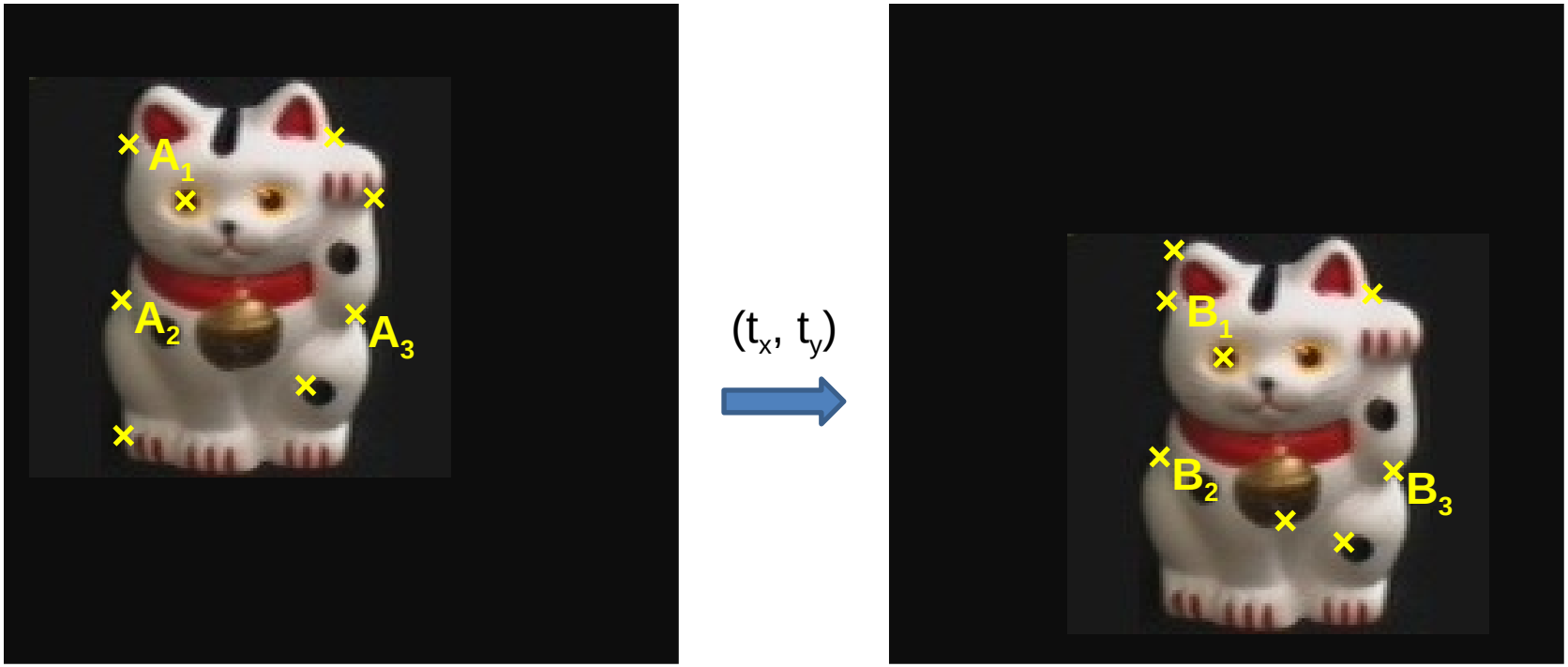
- Fundamental matrix song

- http://danielwedge.com/fmatrix/

Throwback: In Assignment 2 we did feature matching and we said that there is a way to figure out if our matches are correct or not.

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

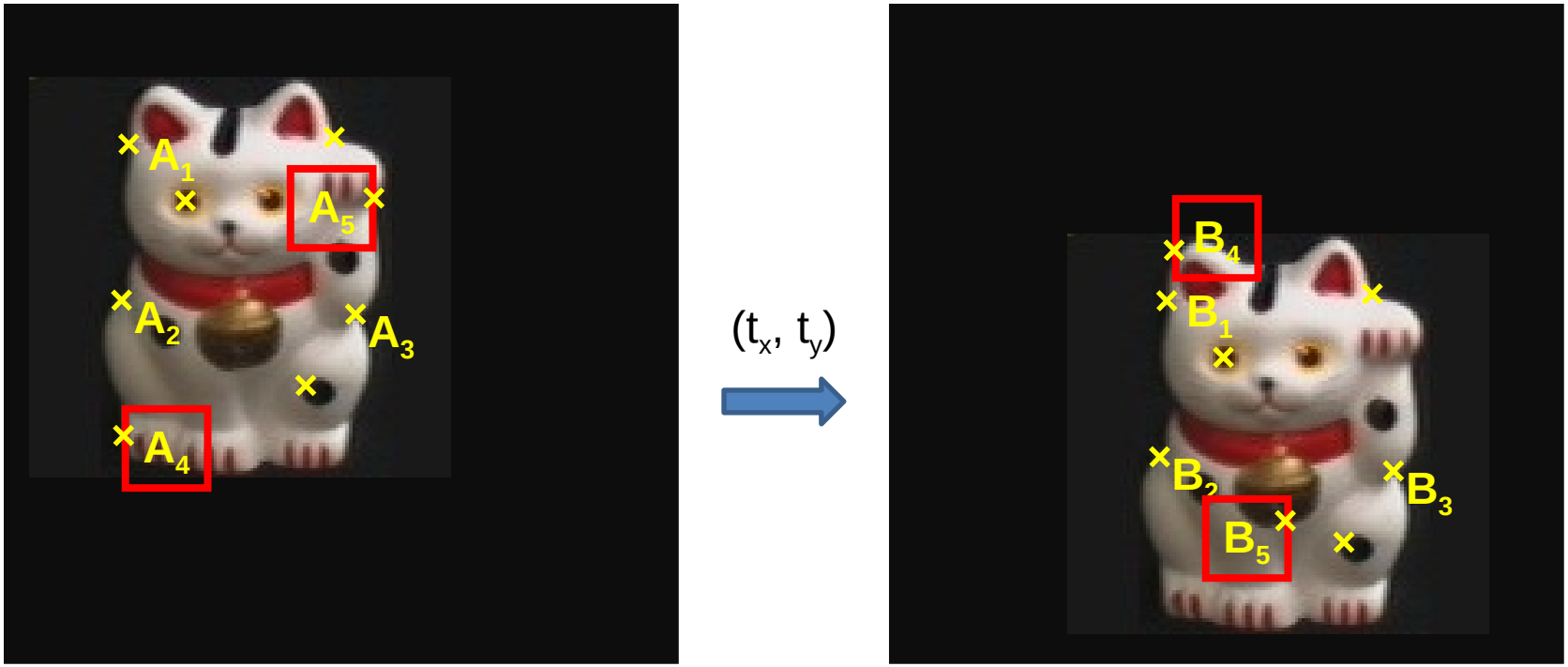
Example: solving for translation



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

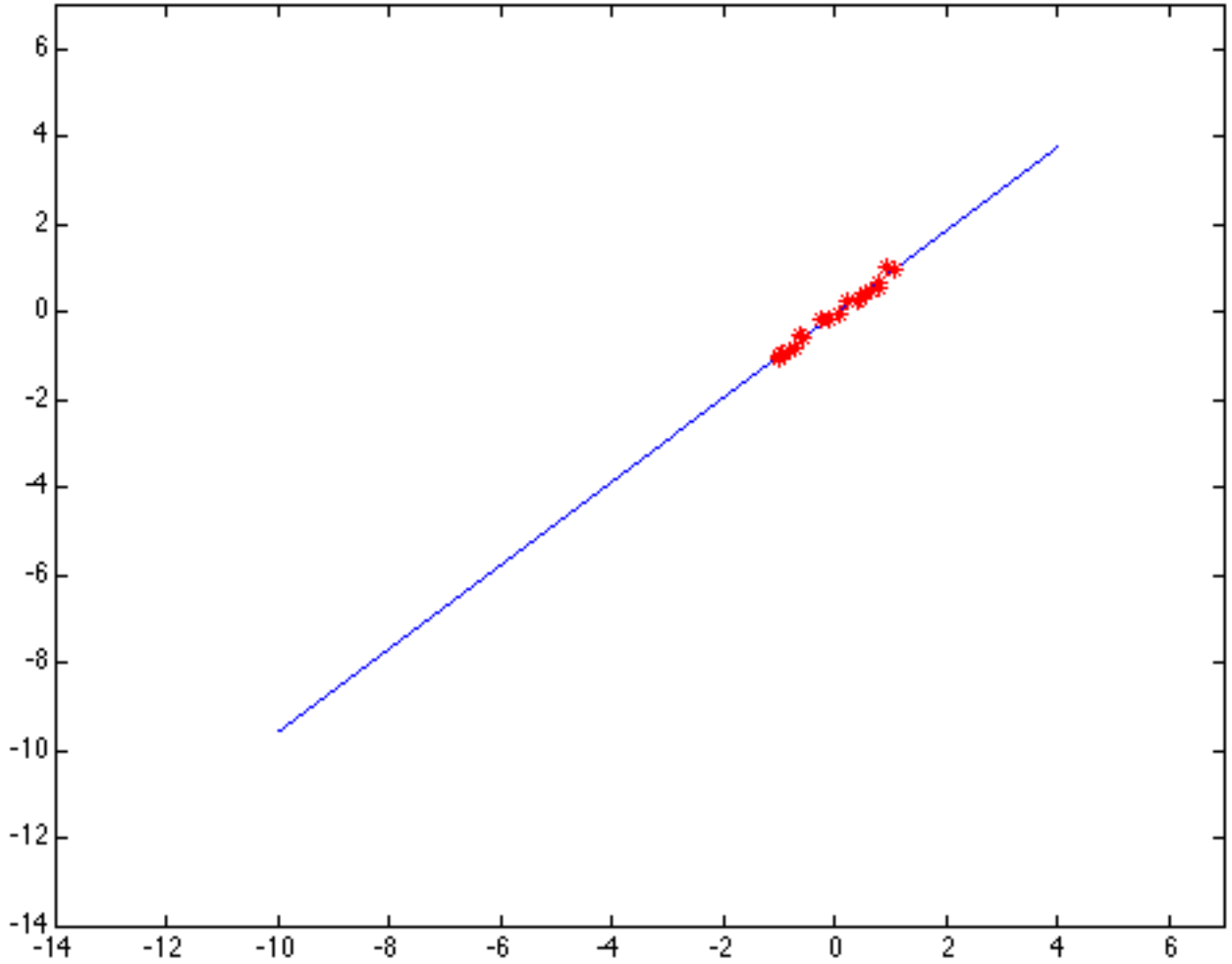


Problem: outliers A_4-B_4 and A_5-B_5 which *incorrectly* correspond

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

Least squares: Robustness to noise

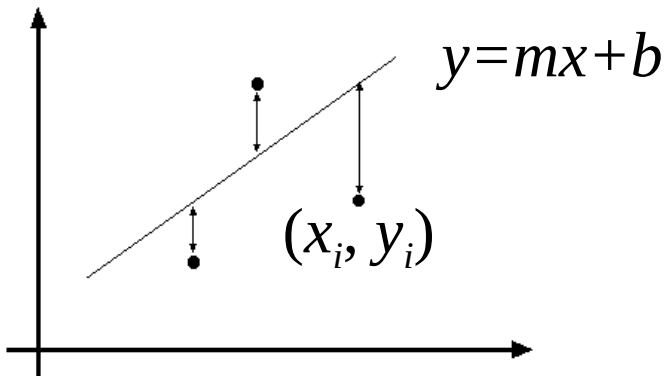
- Least squares fit to the red points:



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

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

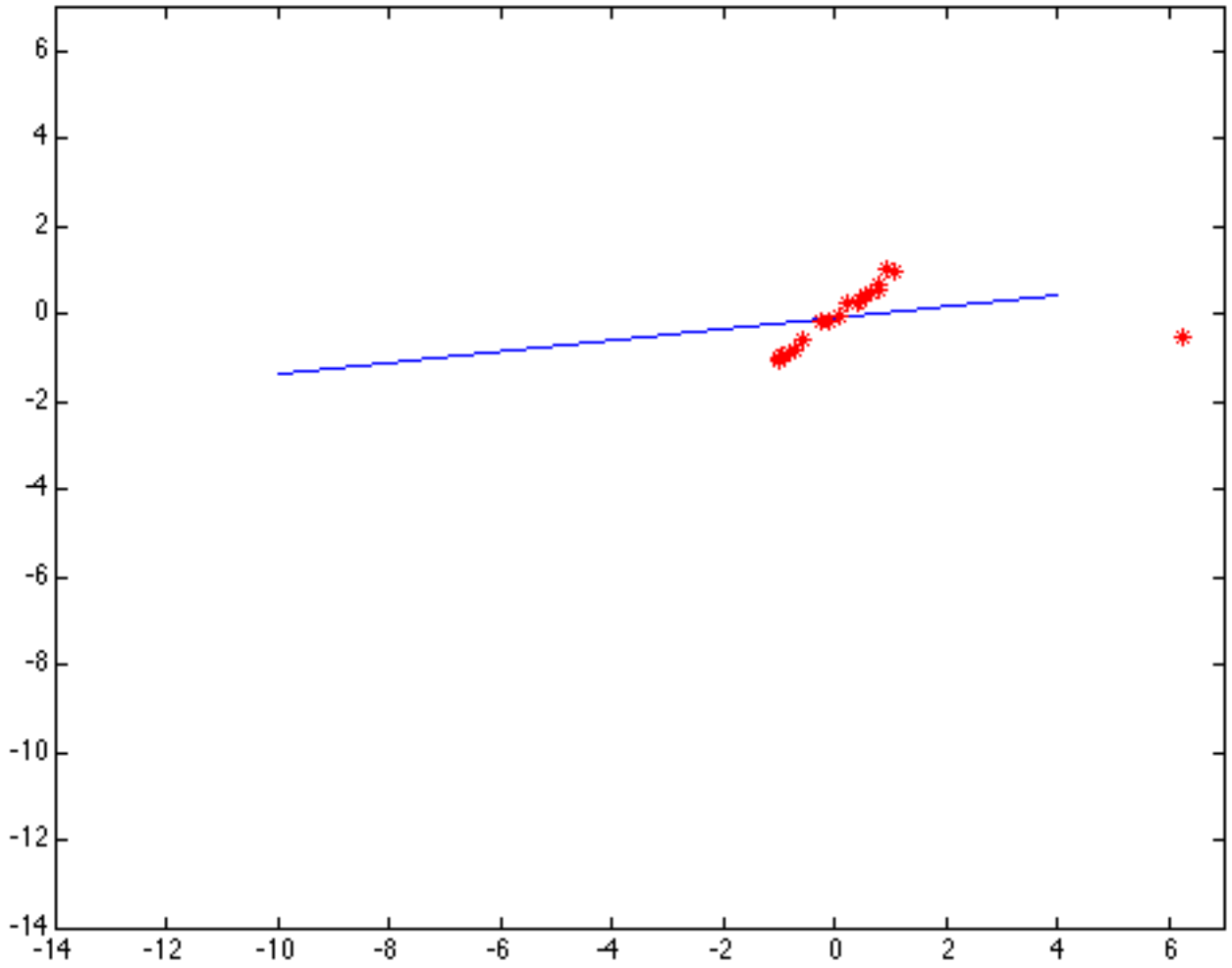


Matlab: `p = A \ y;`
Python: `p = np.linalg.lstsq(A,y)[0]`

(Closed form solution)

Least squares: Robustness to noise

- Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

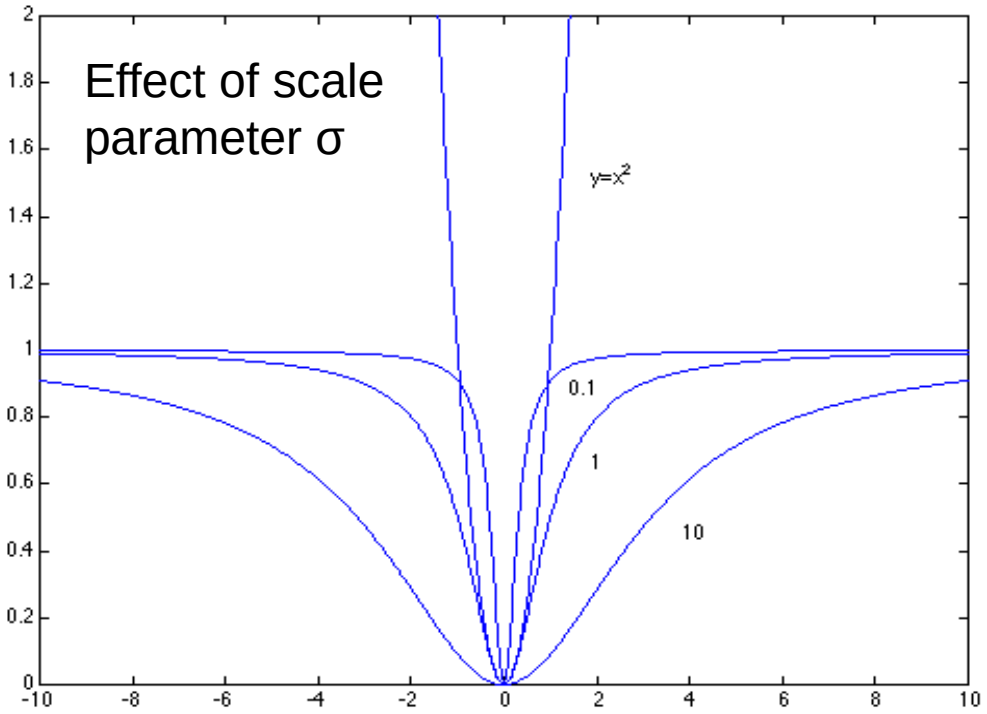
Robust least squares (to deal with outliers)

General approach:

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

$u_i(x_i, \theta)$ – residual of ith point w.r.t. model parameters θ

ρ – robust function with scale parameter σ



The robust function ρ

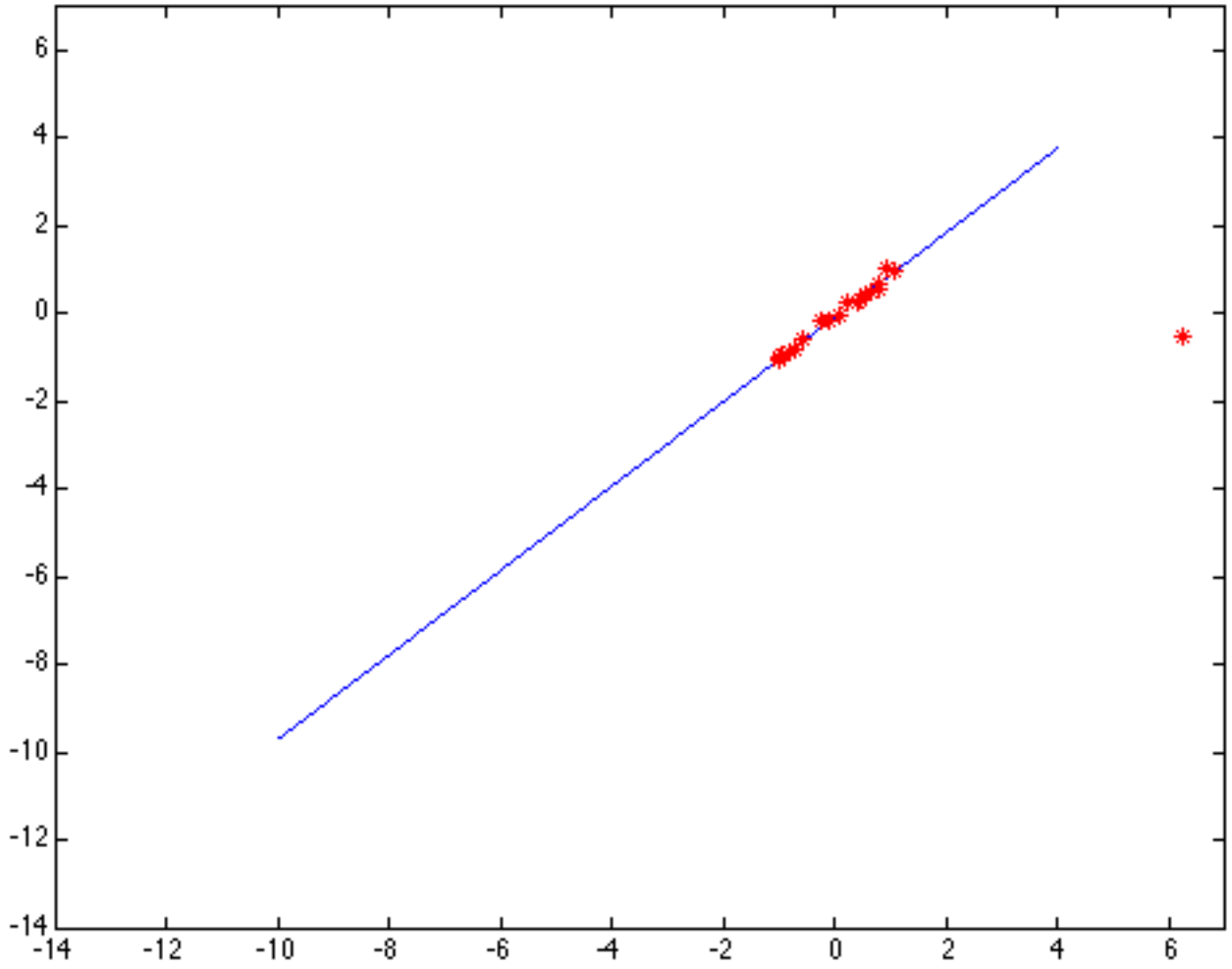
Favors a configuration with small residuals

Constant penalty for large residuals

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

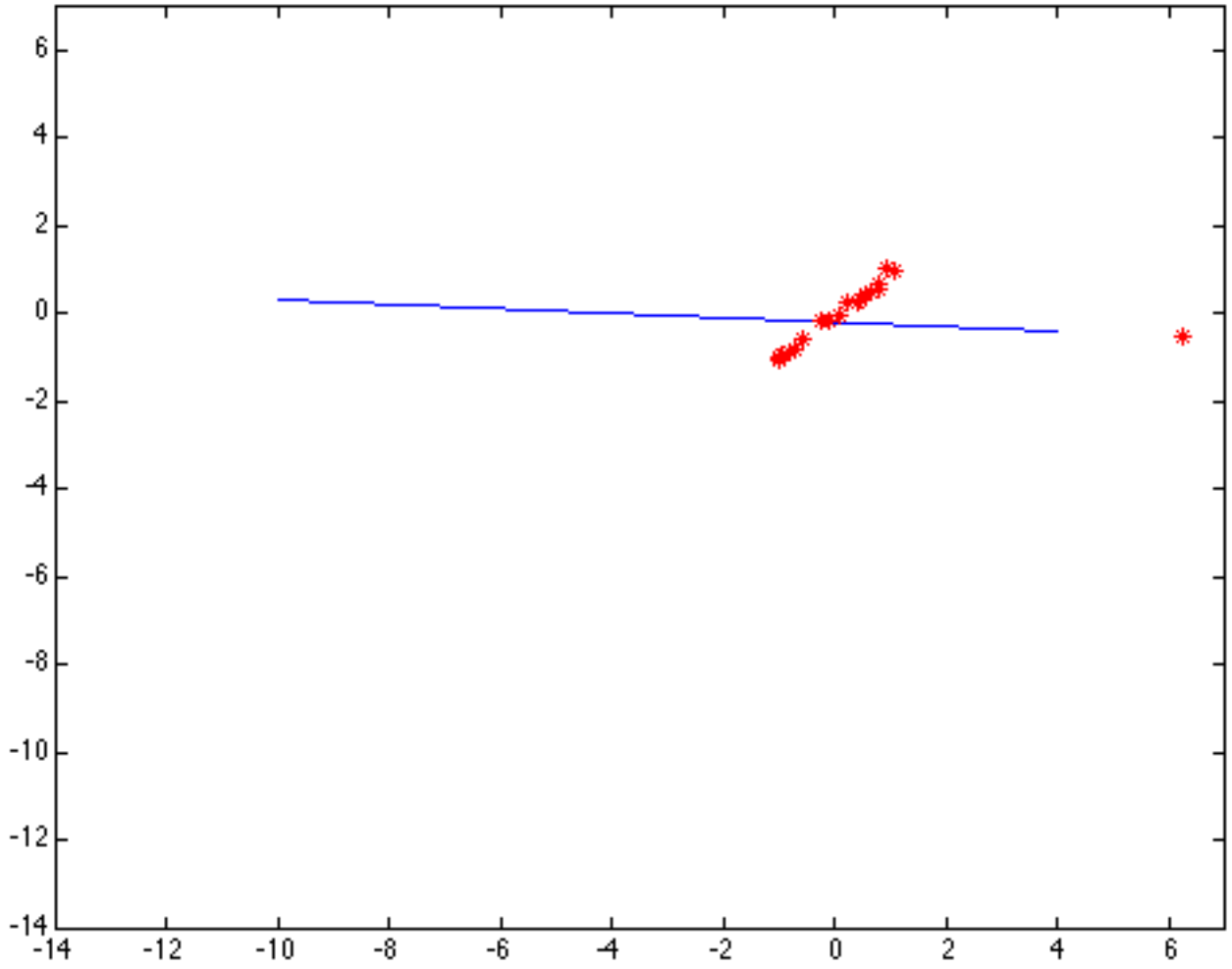
Slide from S. Savarese

Choosing the scale: Just right



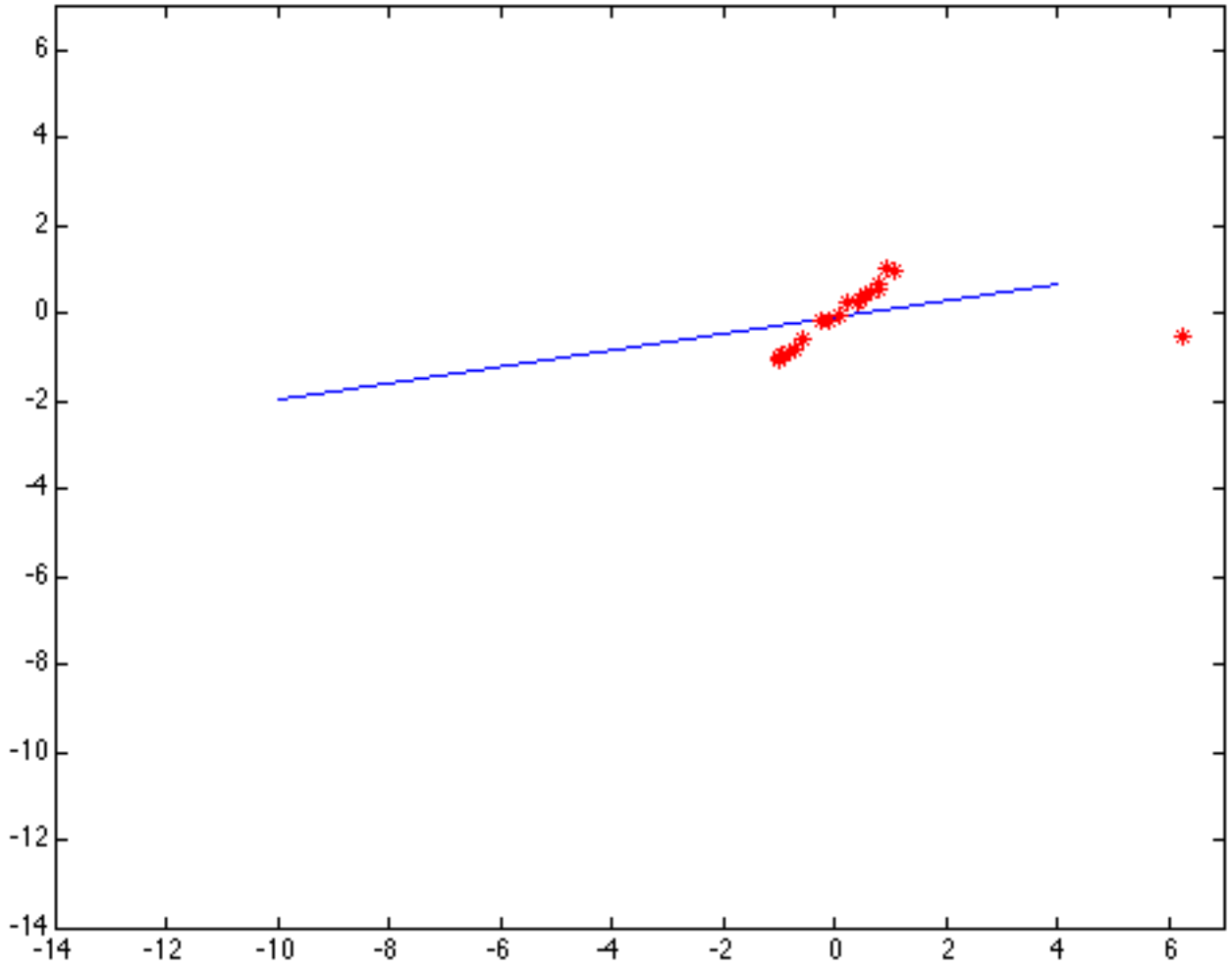
The effect of the outlier is minimized

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



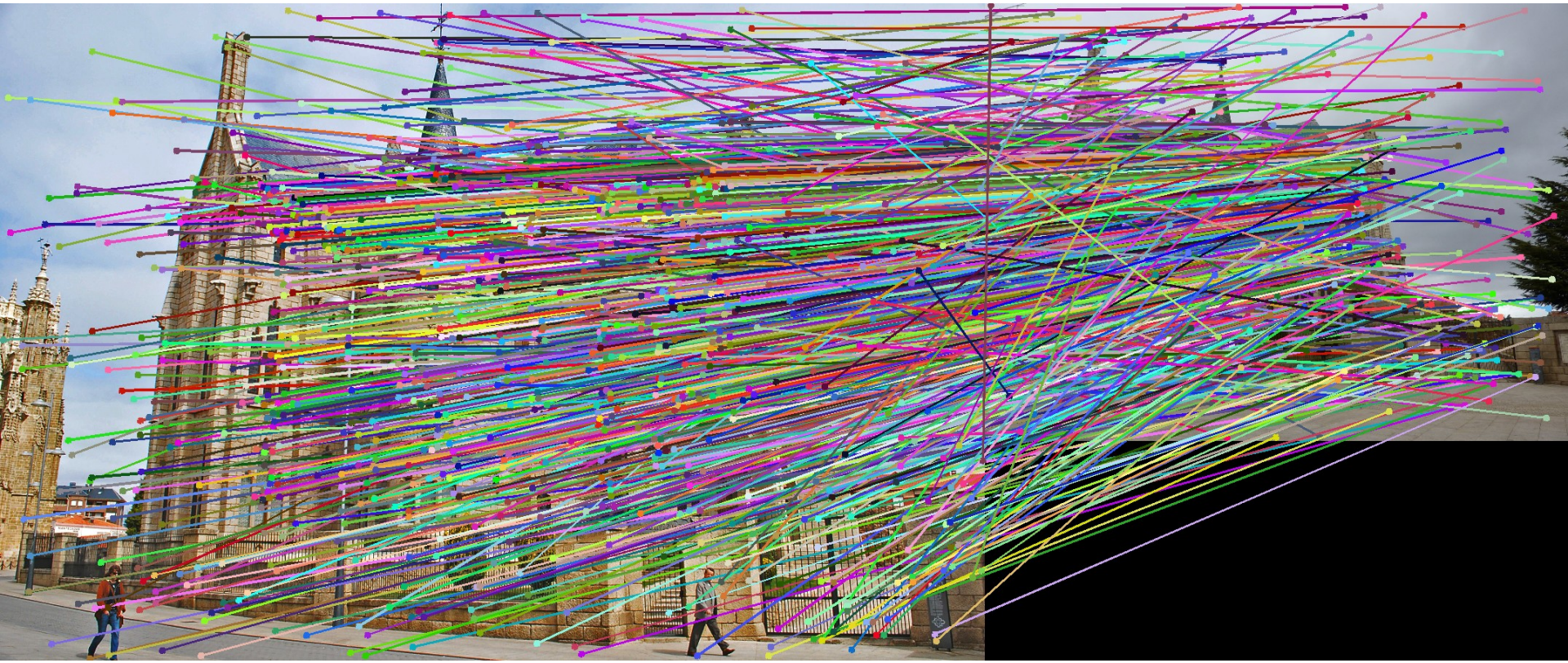
Behaves much the same as least squares

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

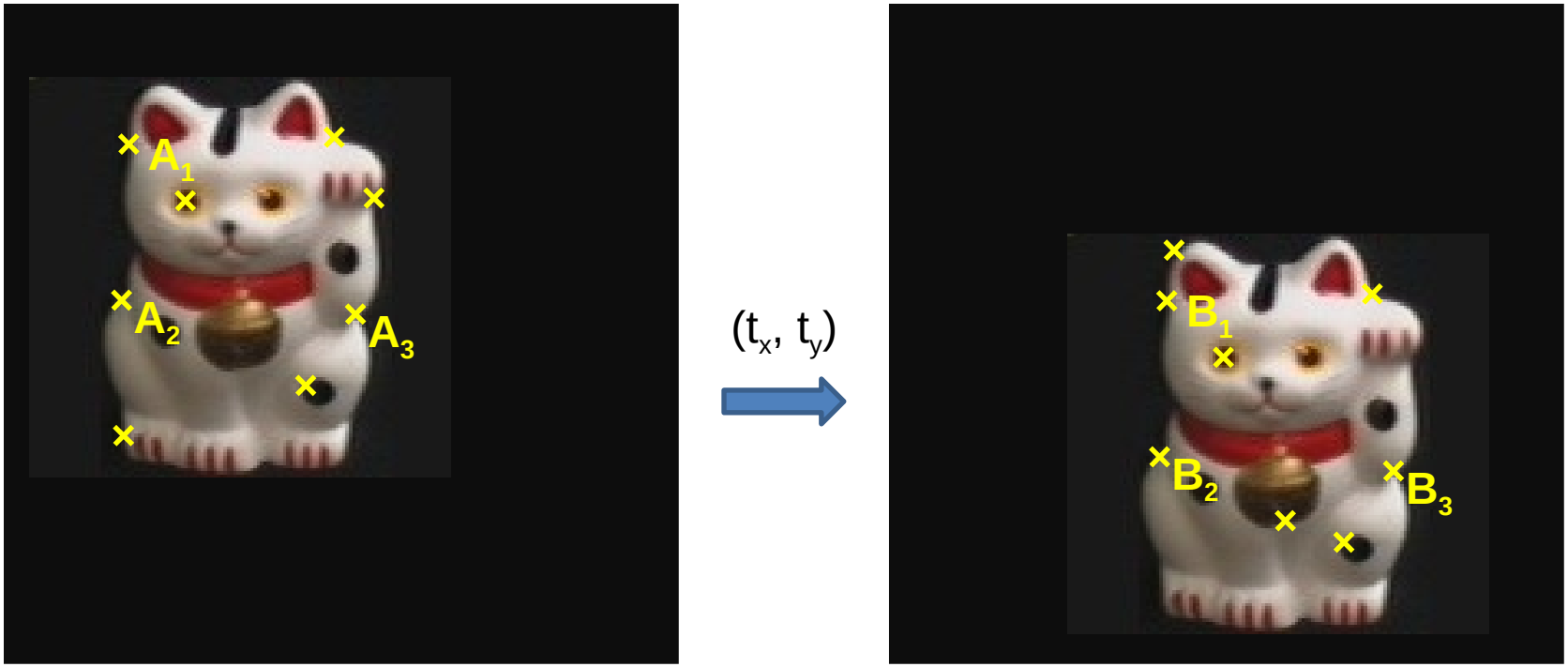
What if I have *many* outliers*?*

Episcopal Gaudi image pair



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

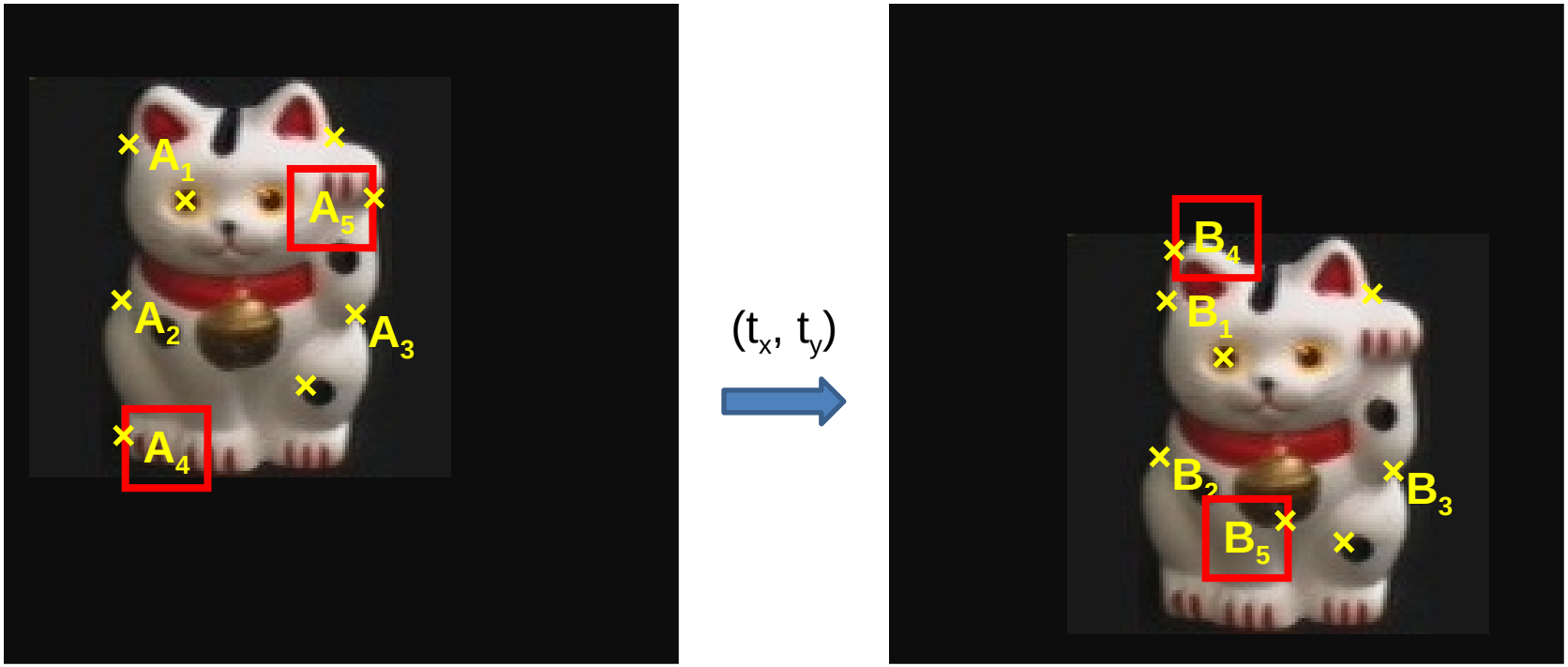
Example: solving for translation



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

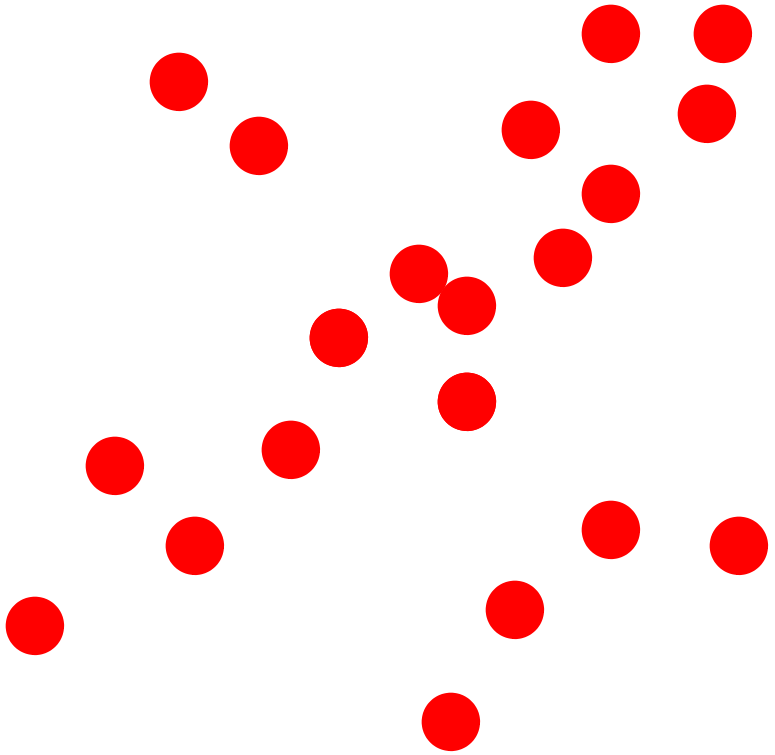
(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.

RANSAC

(RANdom SAmple Consensus) :

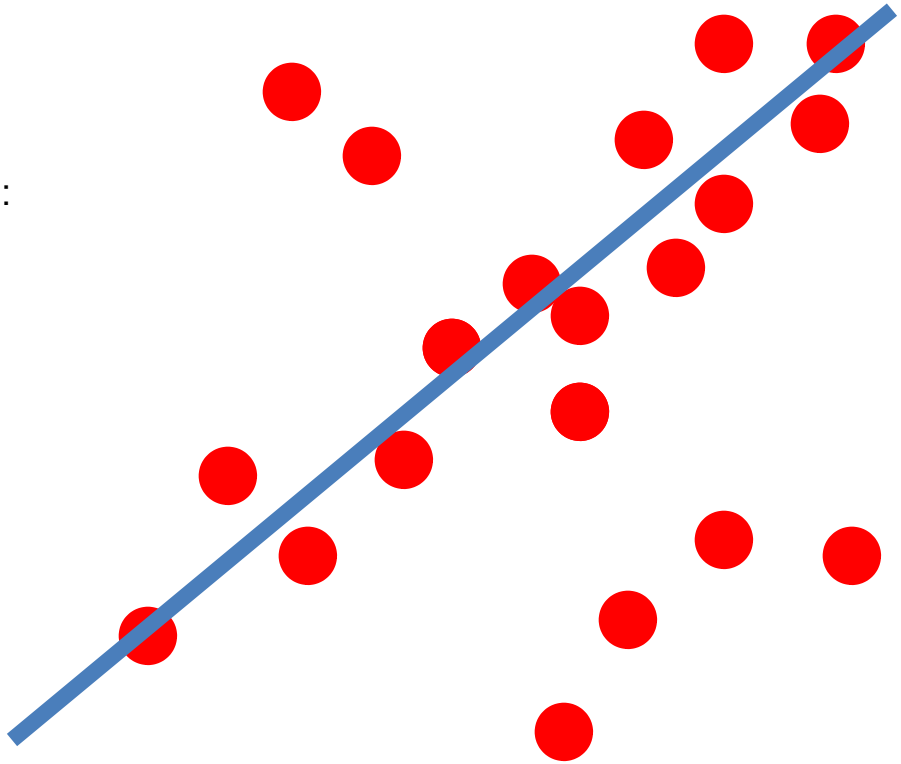
Fischler & Bolles in ‘81.



RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.

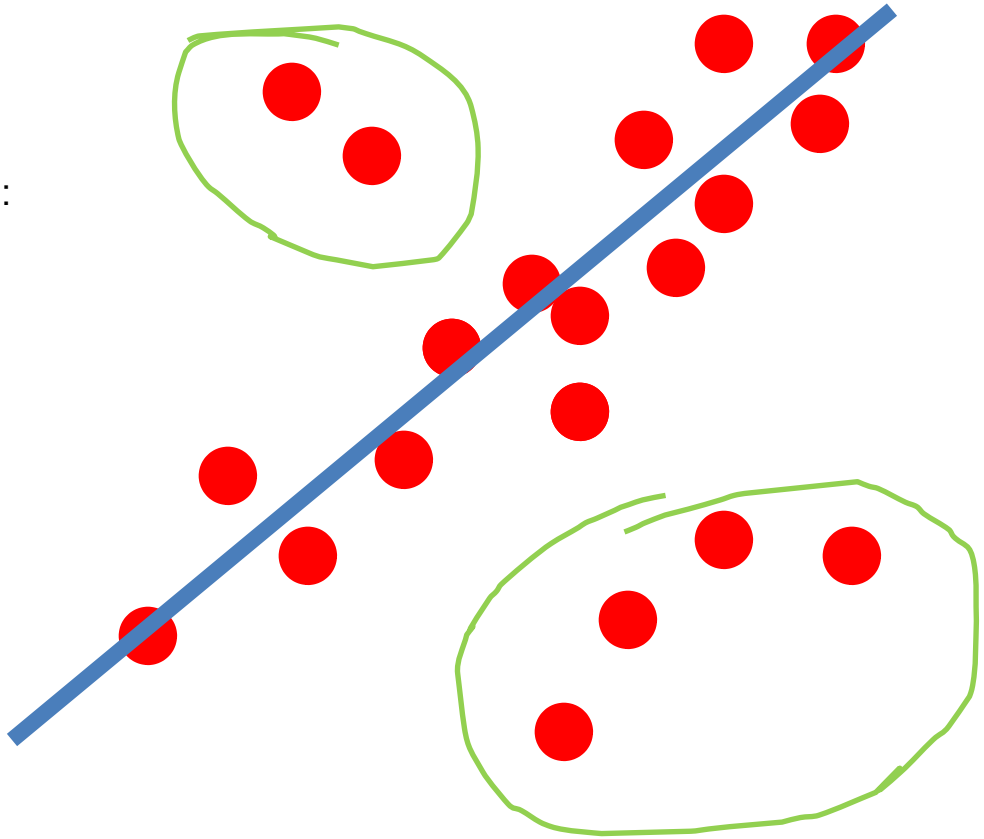


This data is noisy, but we expect a good fit to a known model.

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in ‘81.



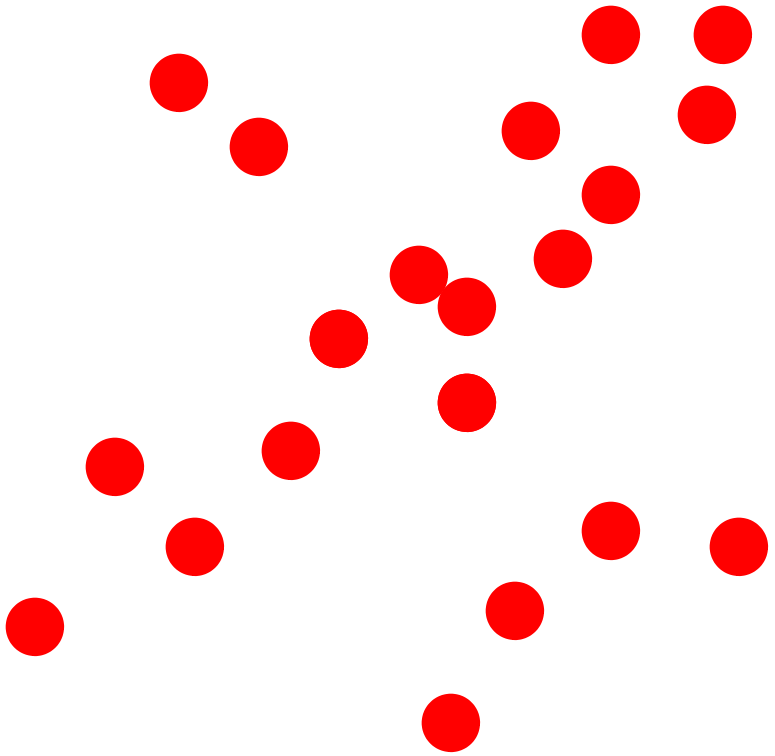
This data is noisy, but we expect a good fit to a known model.

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.



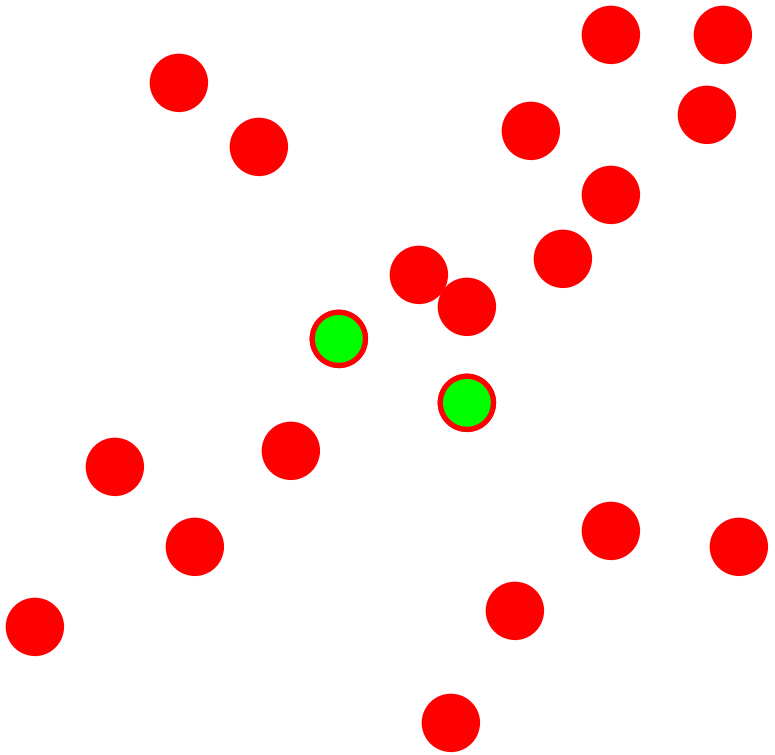
Algorithm:

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:

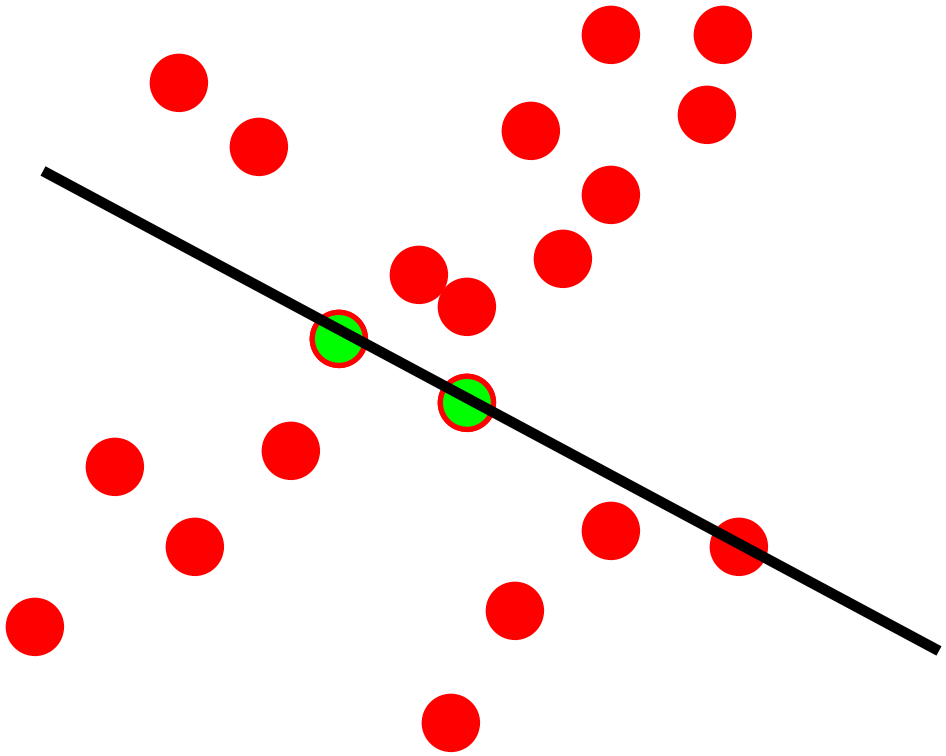
1. **Sample** (randomly) the number of points required to fit the model (s=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

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

Illustration by Savarese

RANSAC

Line fitting example



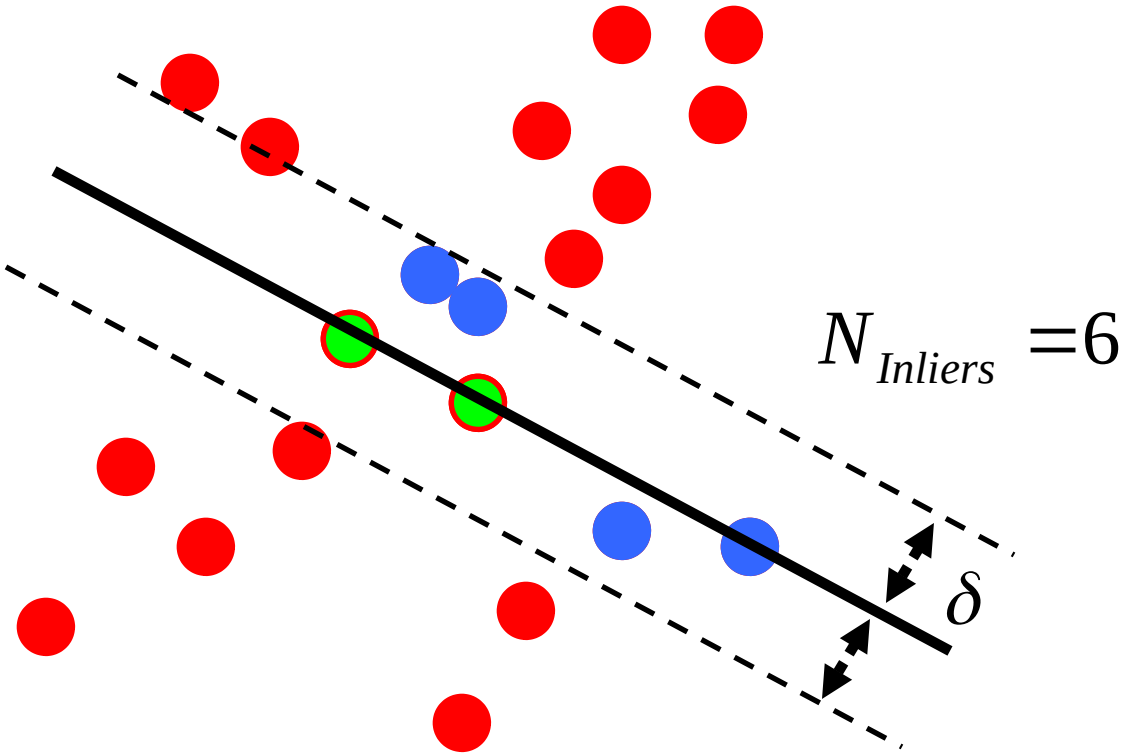
Algorithm:

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

Line fitting example

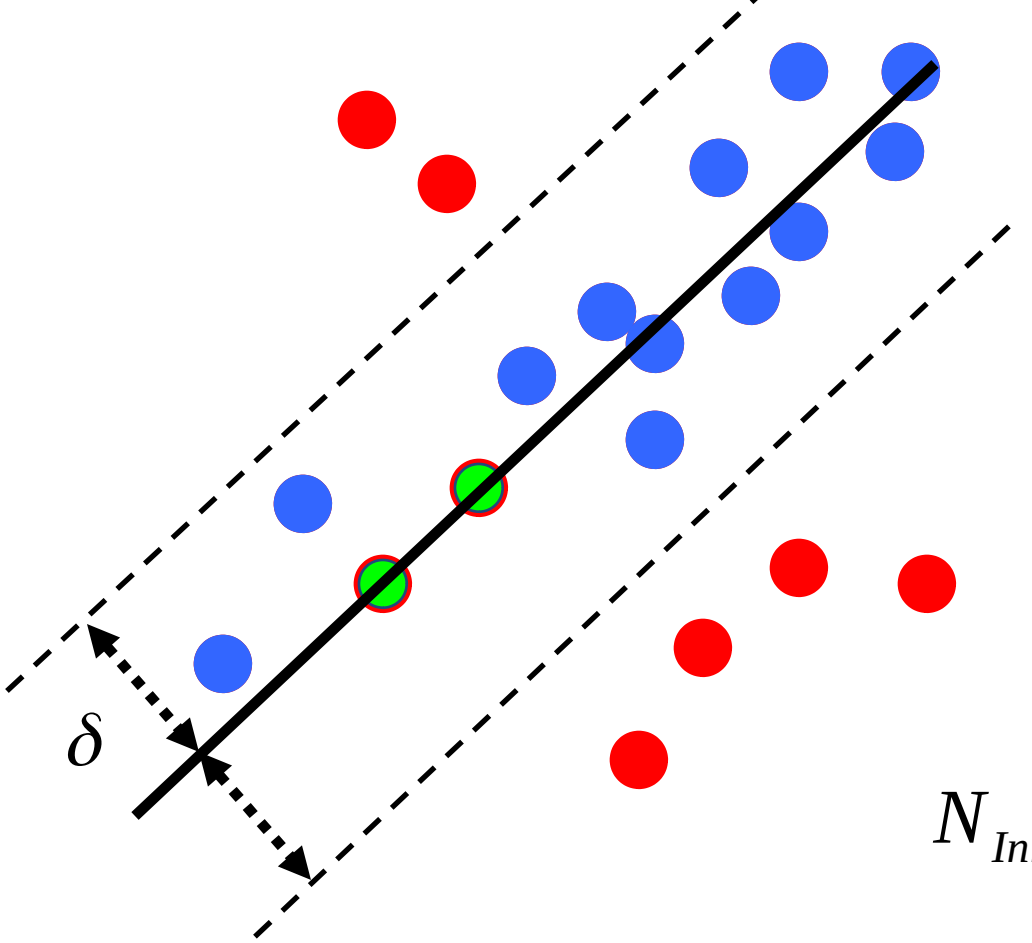


Algorithm:

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



$N_{Inliers} = 14$

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 sampled points *s*
  - Minimum number needed to fit the model

- 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 = expected percentage of outliers in your data, empircally tuned based on your expectations of how good the data is.*)

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 $p = 0.99$

How to choose parameters?

- Distance threshold *t*
  - *Usually chosen empirically*
  - *But…when measurement error is known to be Gaussian with mean 0 and variance* σ^2*:*
  - Choose ***t*** so that a point within ***t*** has a high probability of being an inlier (prob being Inlier = 0.95).
  - For fundamental matrices
    - estimate F few times, compute the Zero-mean Gaussian noise of some samples to obtain the std. dev. σ.
    - Then the threshold for RANSAC is $t^2=3.84\sigma^2$

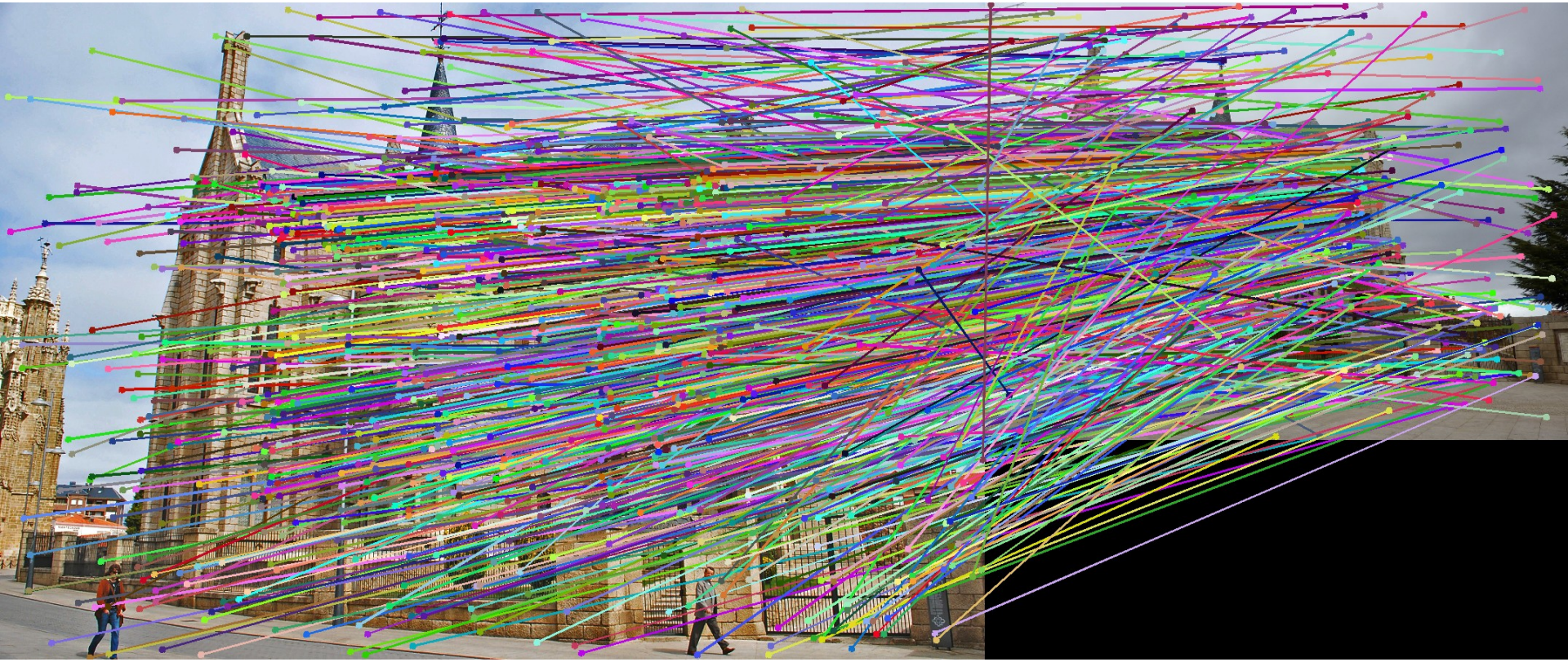
How many iterations do I need?

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

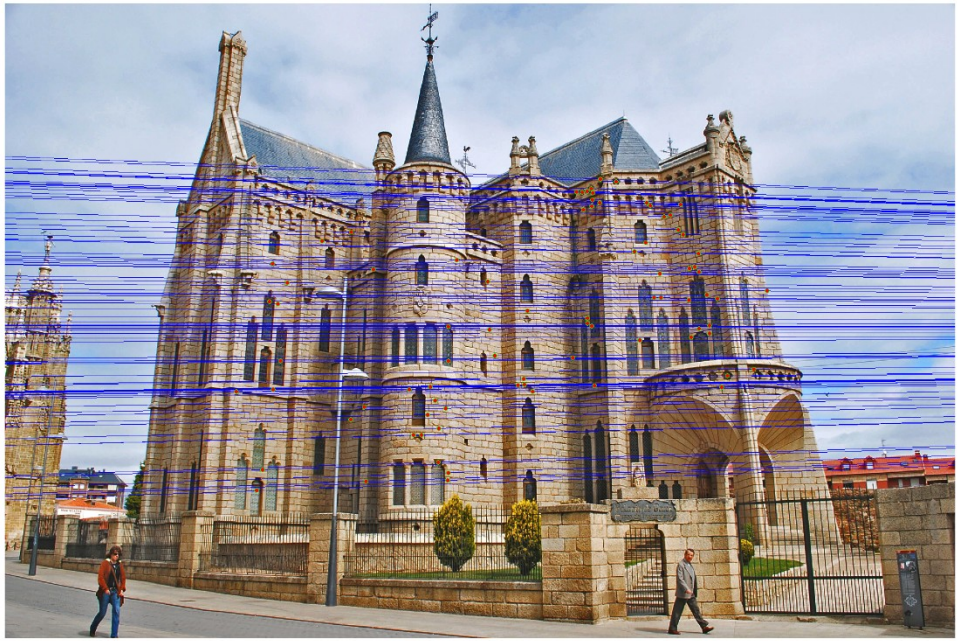
	Proportion of outliers e						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

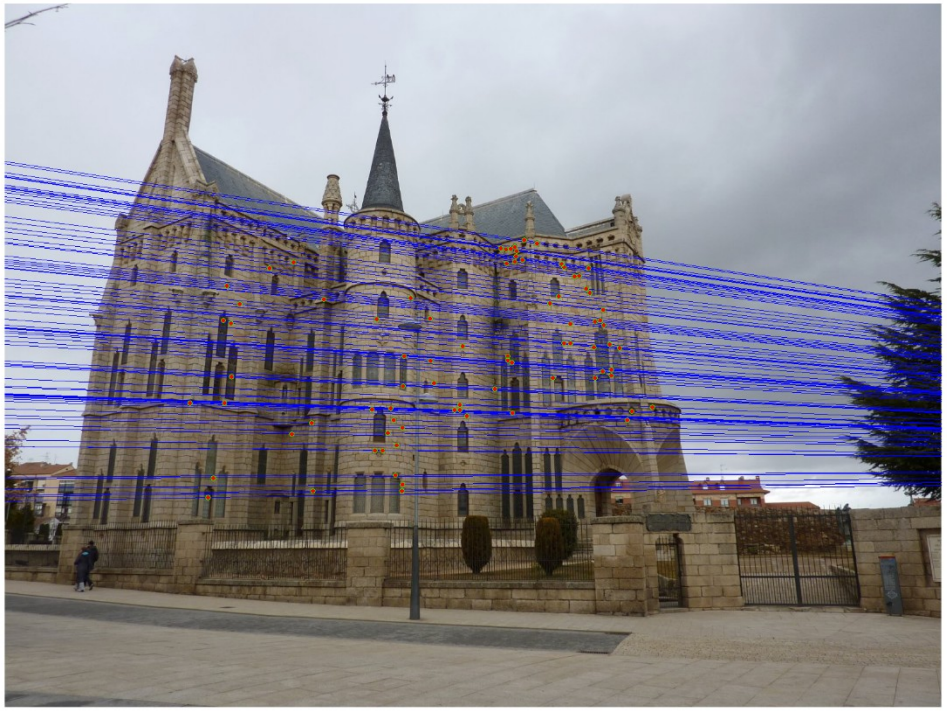
For p = 0.99

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

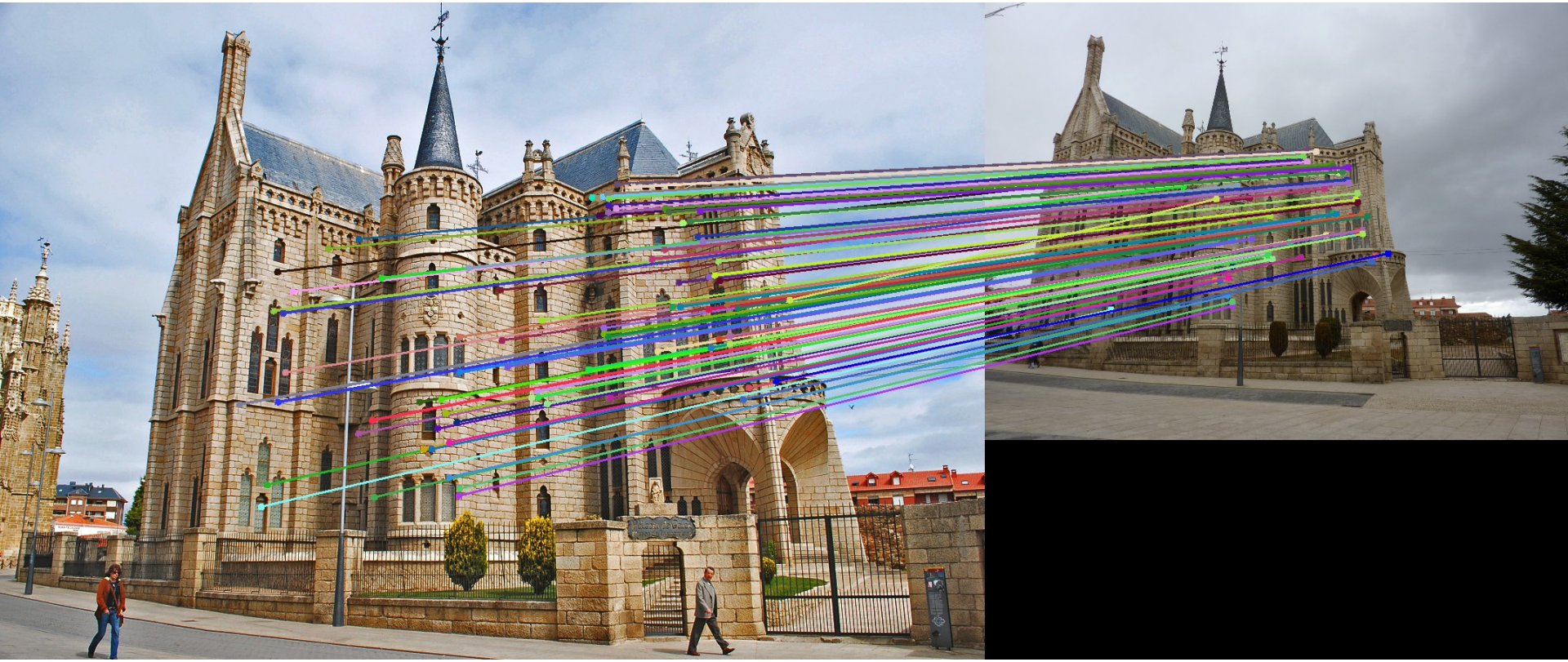


Epipolar lines





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



RANSAC conclusions

Good

- Robust to outliers
- Applicable for large number of objective function parameters
- Optimization parameters are easier to choose

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)

RANSAC spin offs

- There are dozens of RANSAC spinoffs

- DSAC - MLESAC - WaldSAC - PROSAC, Lo-RANSAC, ARRSAC, MAGSAC, MAPSAC, KALMANSAC, and many many more