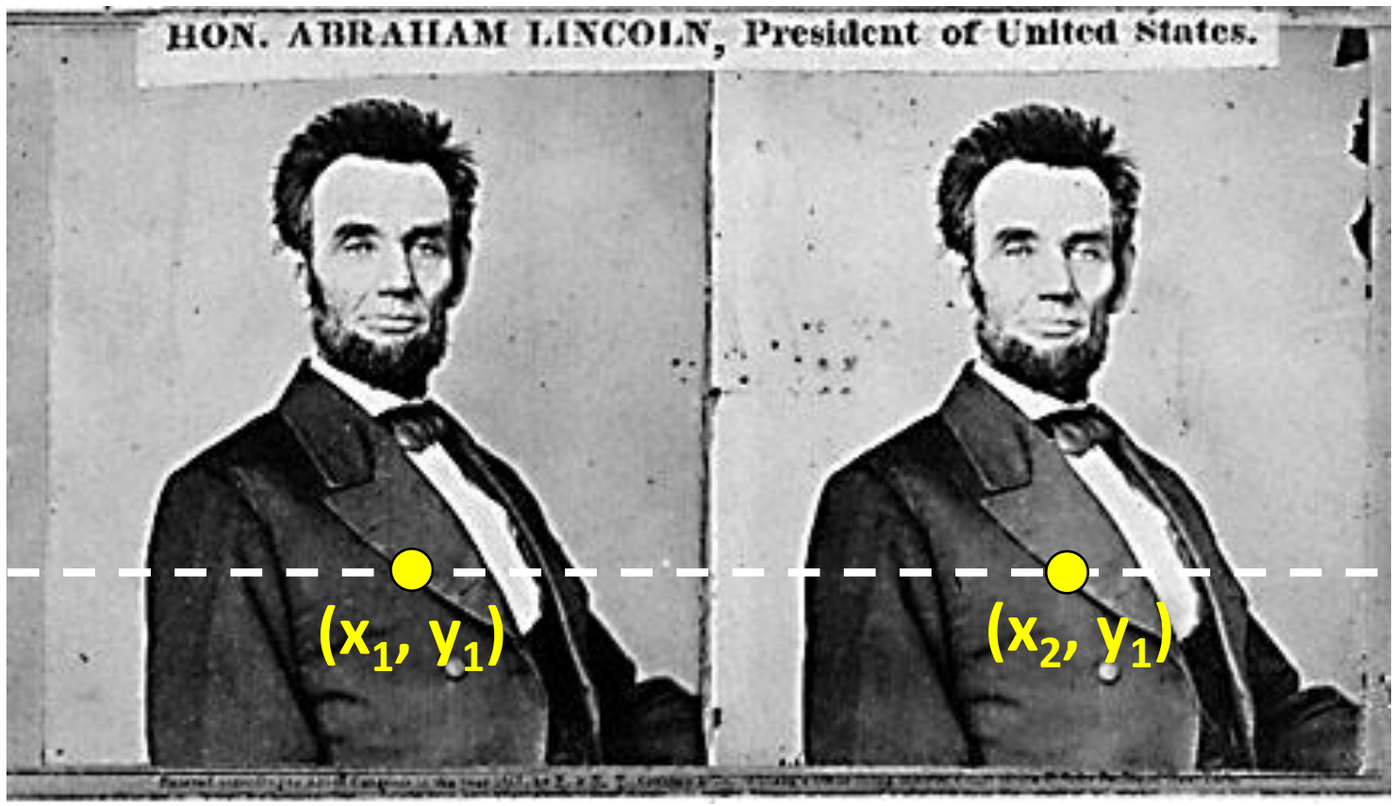
Geometry and Structure from Motion

Computer Vision
Fall 2019
Columbia University

Stereo

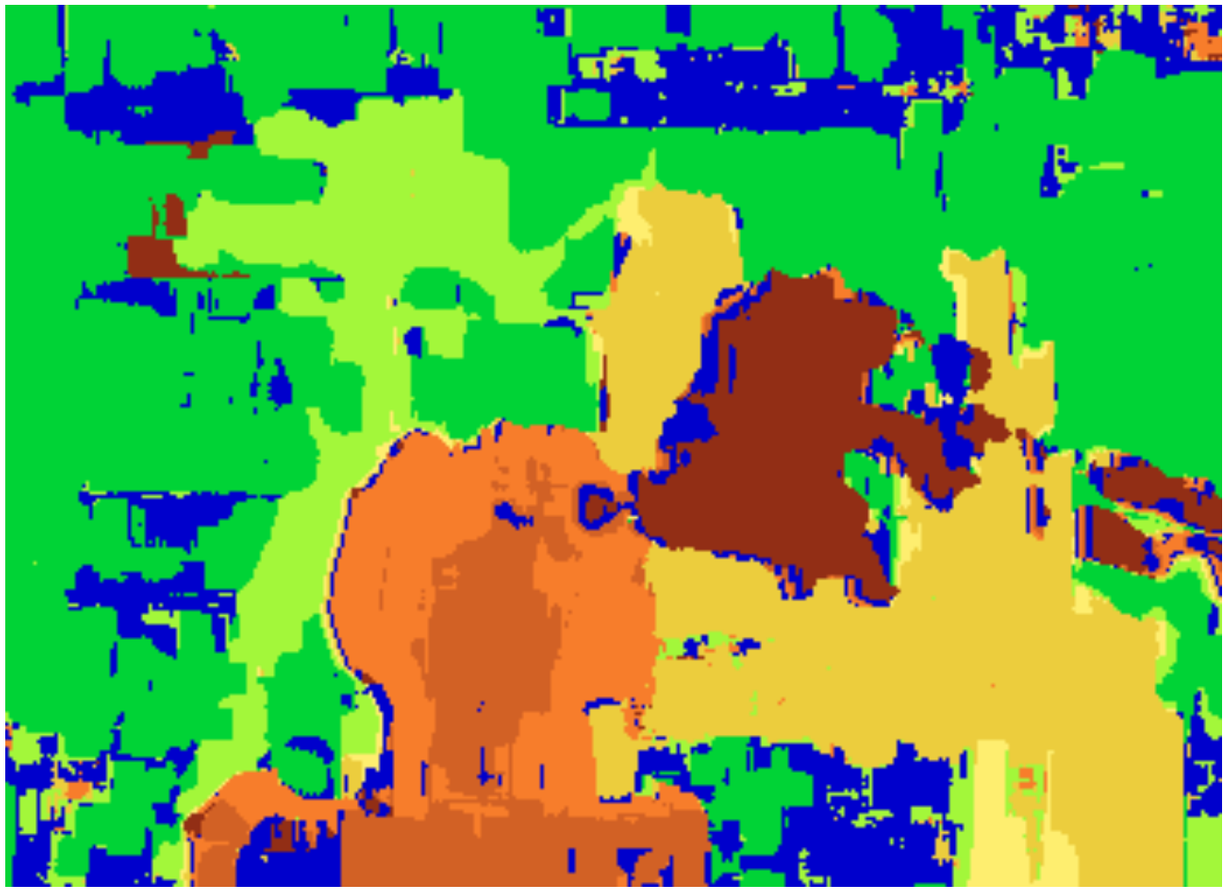


Two images captured by a purely horizontal translating camera (*rectified* stereo pair)

$x_2 - x_1$ = the *disparity* of pixel (x_1, y_1)

Slide credit: Noah Snavely

Results with window search



Window-based matching
(best window size)

Ground truth

Slide credit: Noah Snavely

Stereo as energy minimization

Simple pixel / window matching: choose the minimum of each column in the DSI independently:

$$d(x,y) = \arg\min_{d'} C(x,y,d')$$

Slide credit: Noah Snavely

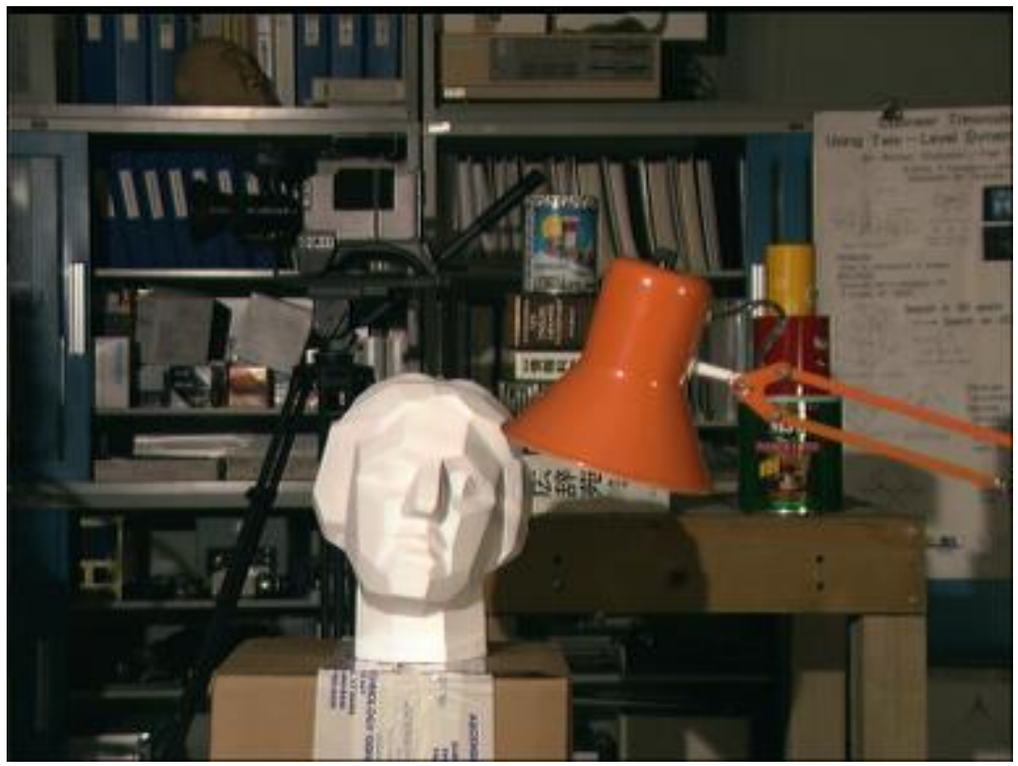
Stereo as energy minimization

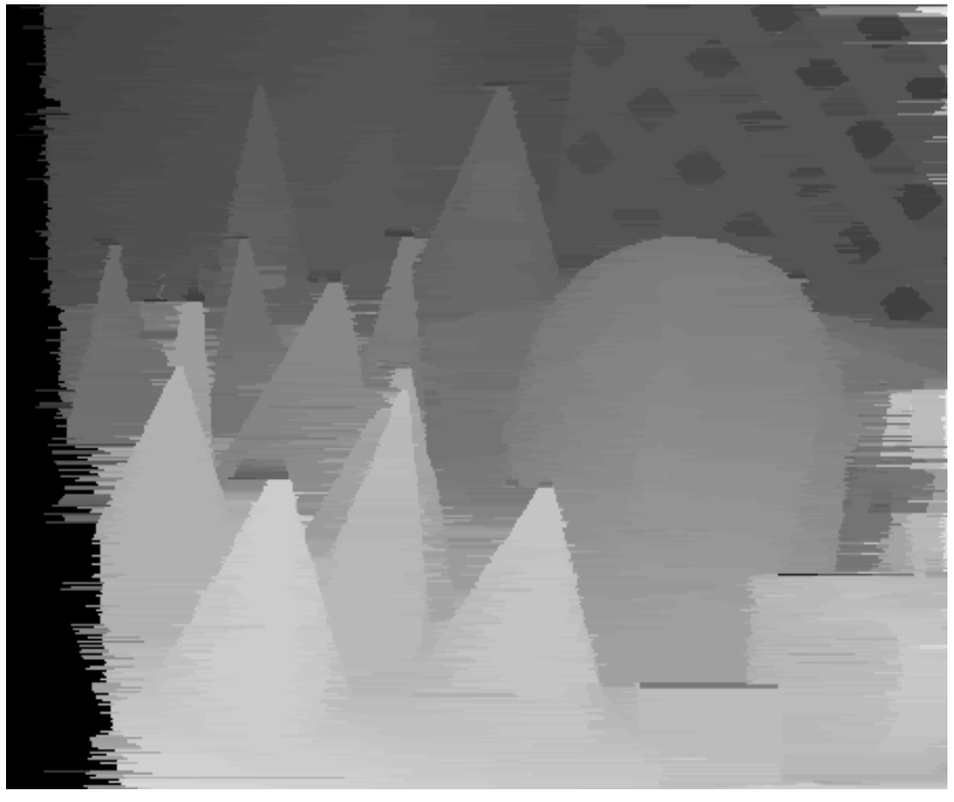
- Finds “smooth”, low-cost path through DPI from left to right

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

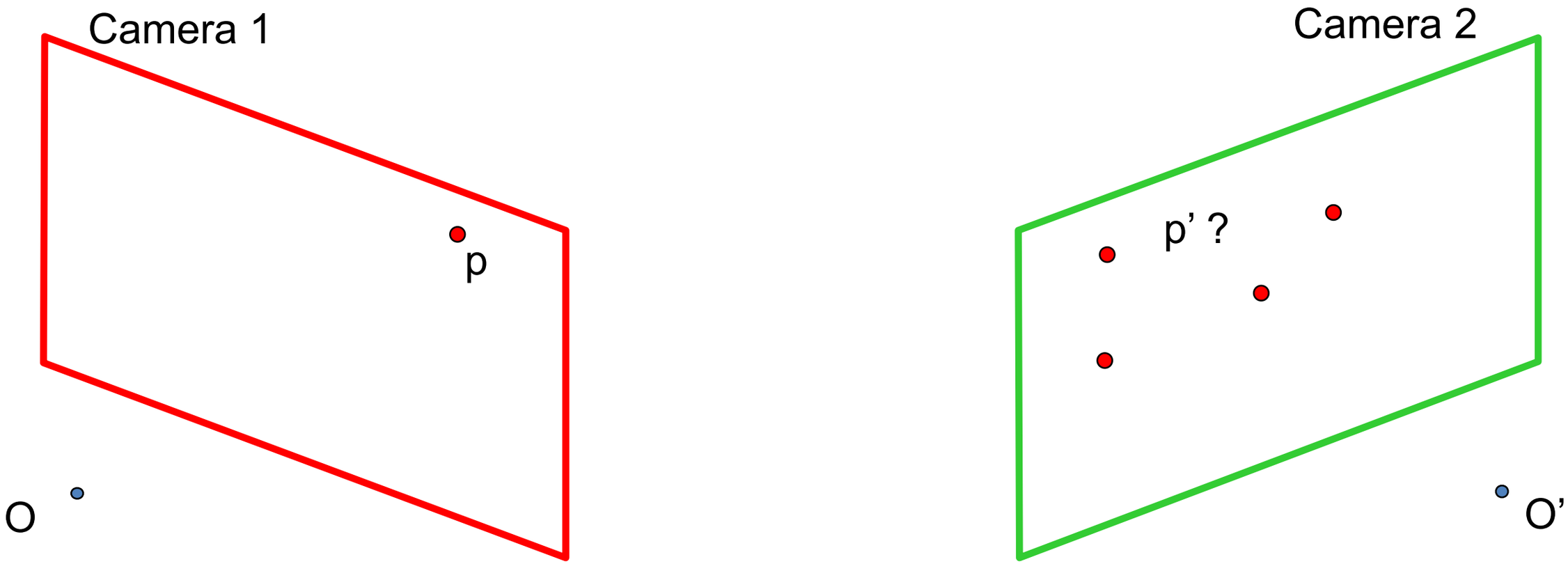
Slide credit: Noah Snavely

Dynamic Programming





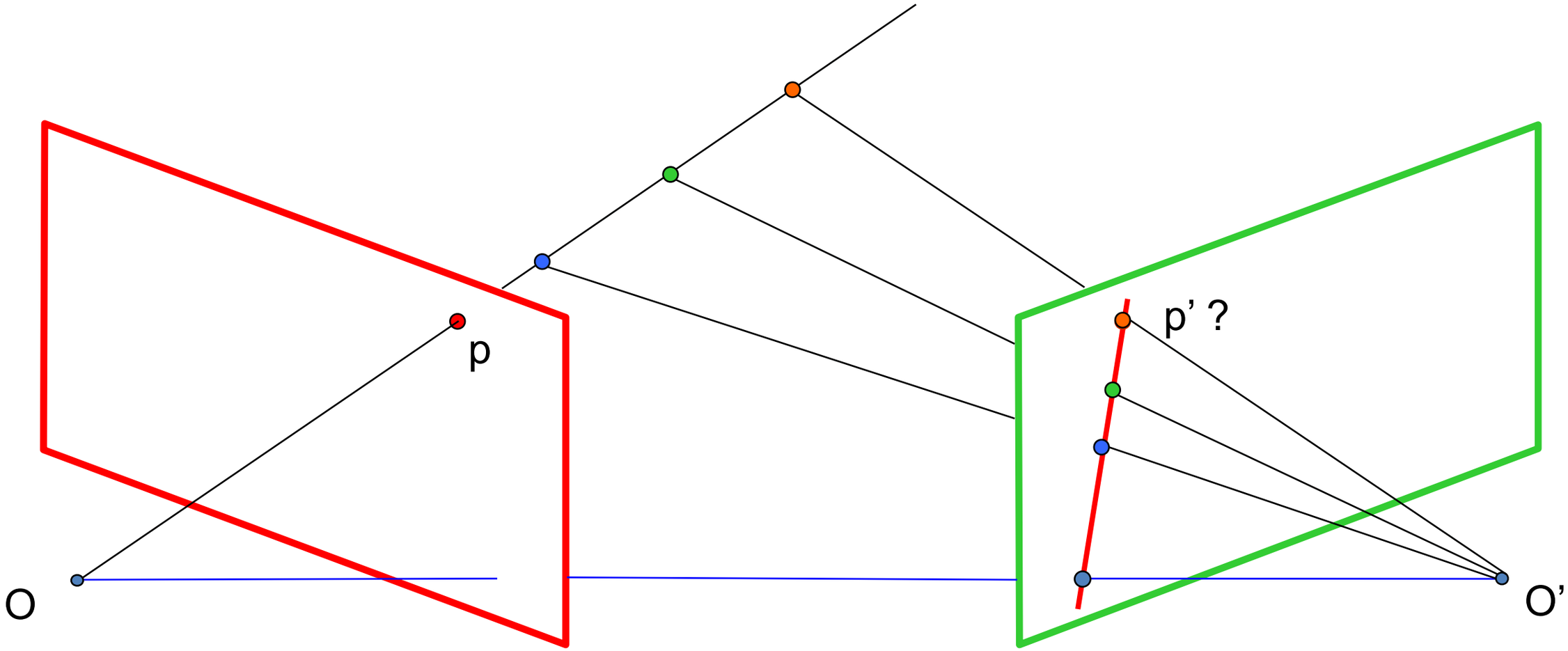
Stereo correspondence constraints



If we see a point in camera 1, are there any constraints on where we will find it on camera 2?

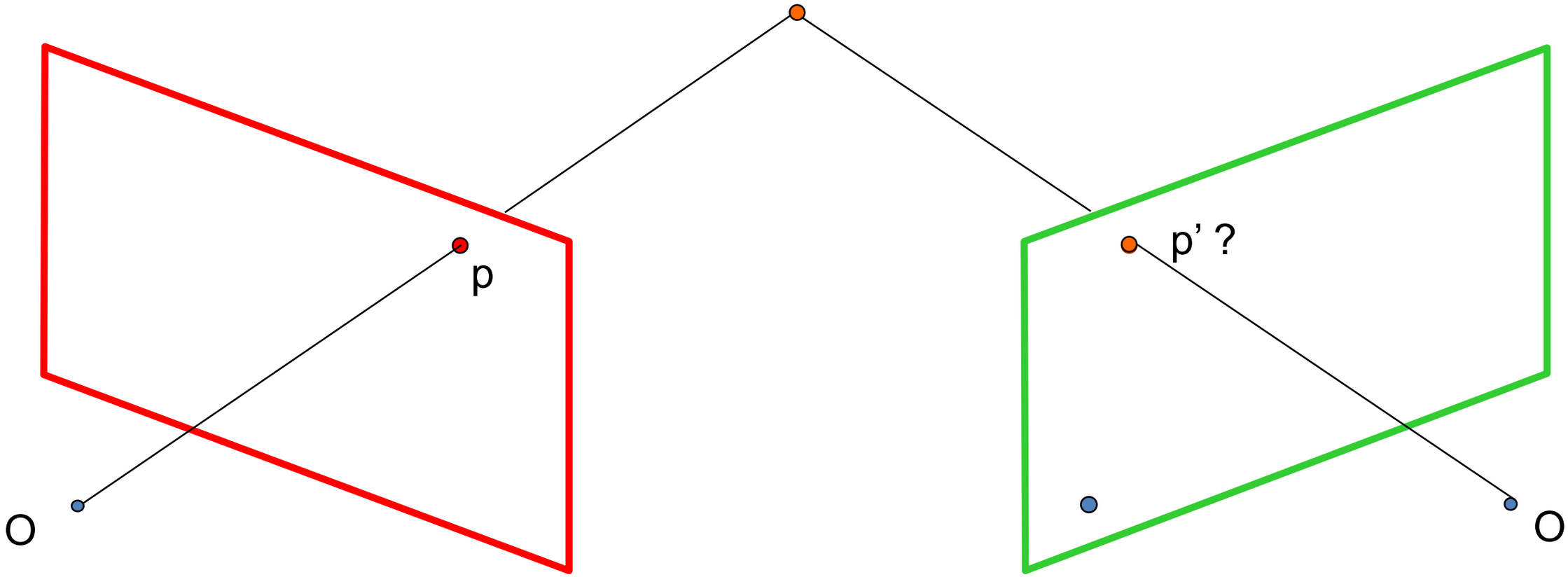
Slide credit: Antonio Torralba

Epipolar constraint



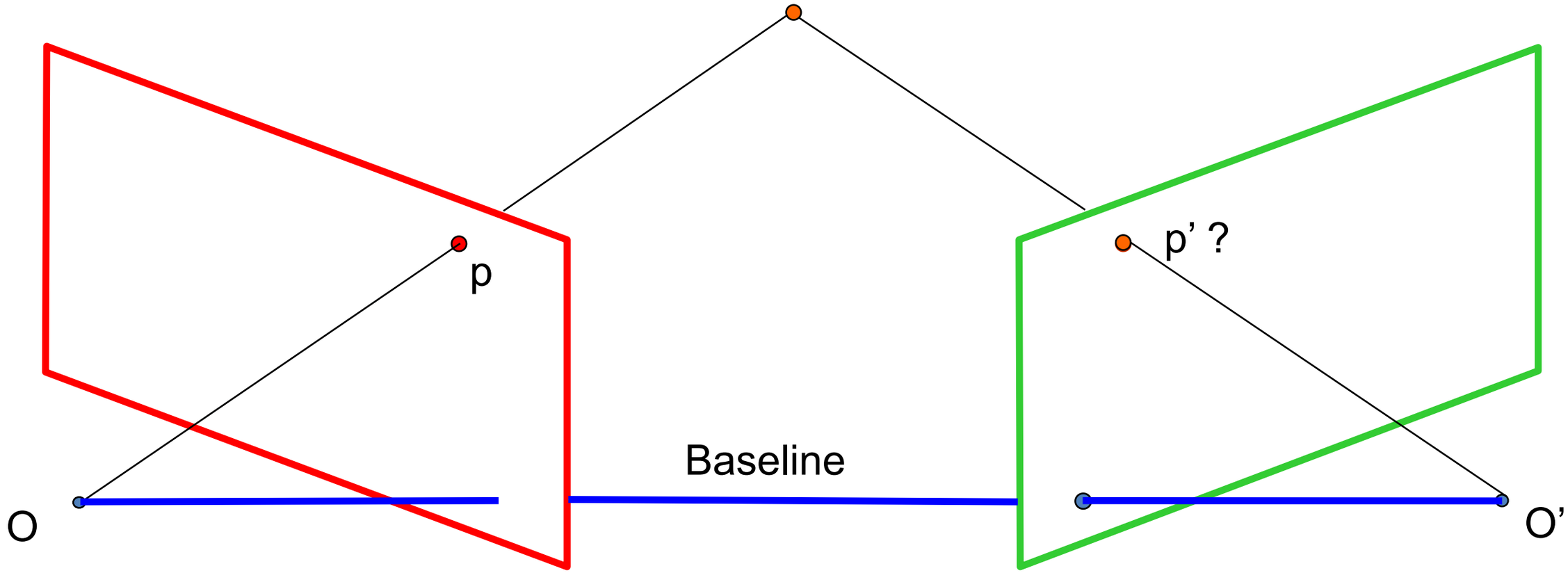
Slide credit: Antonio Torralba

Some terminology



Slide credit: Antonio Torralba

Some terminology

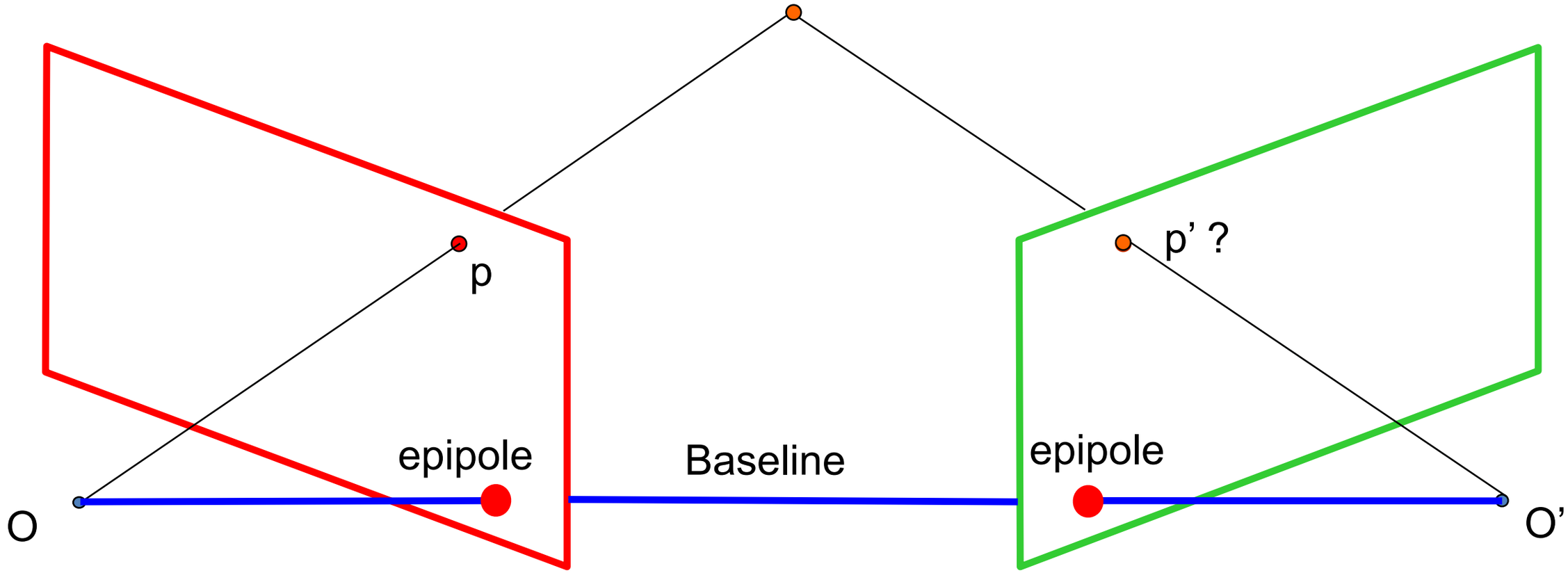


Baseline: the line connecting the two camera centers

Epipole: point of intersection of *baseline* with the image plane

Slide credit: Antonio Torralba

Some terminology

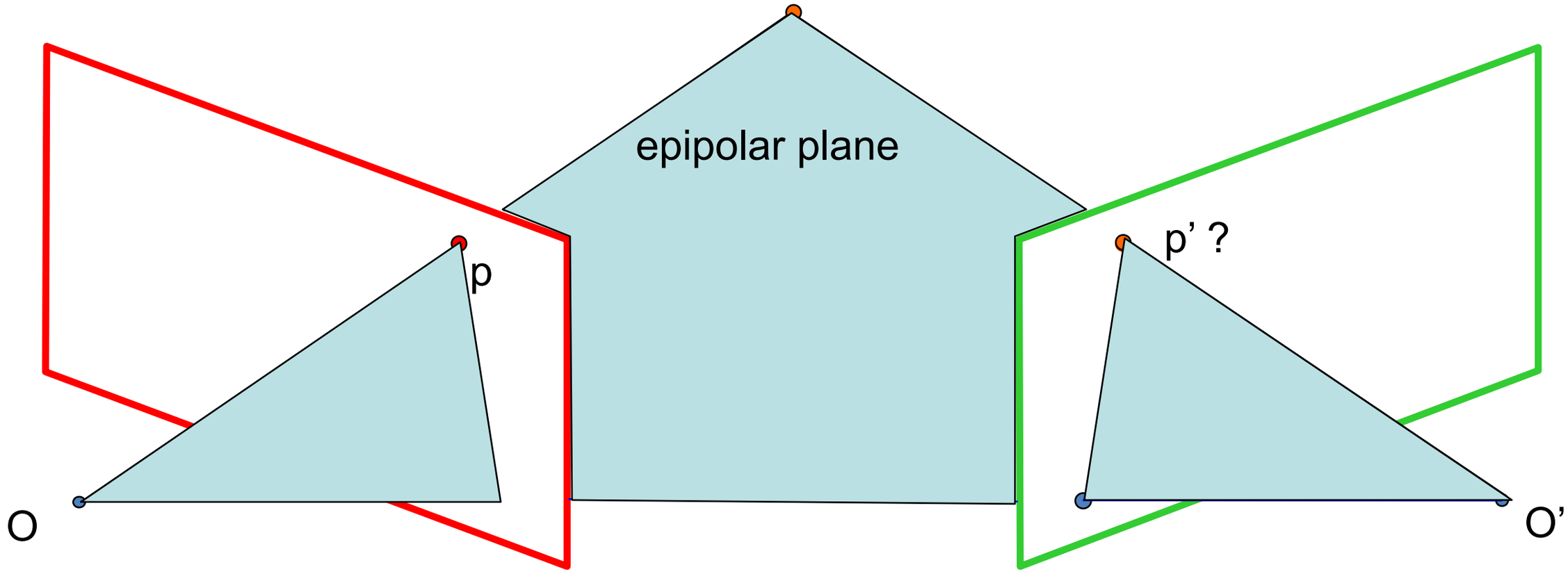


Baseline: the line connecting the two camera centers

Epipole: point of intersection of *baseline* with the image plane

Slide credit: Antonio Torralba

Some terminology



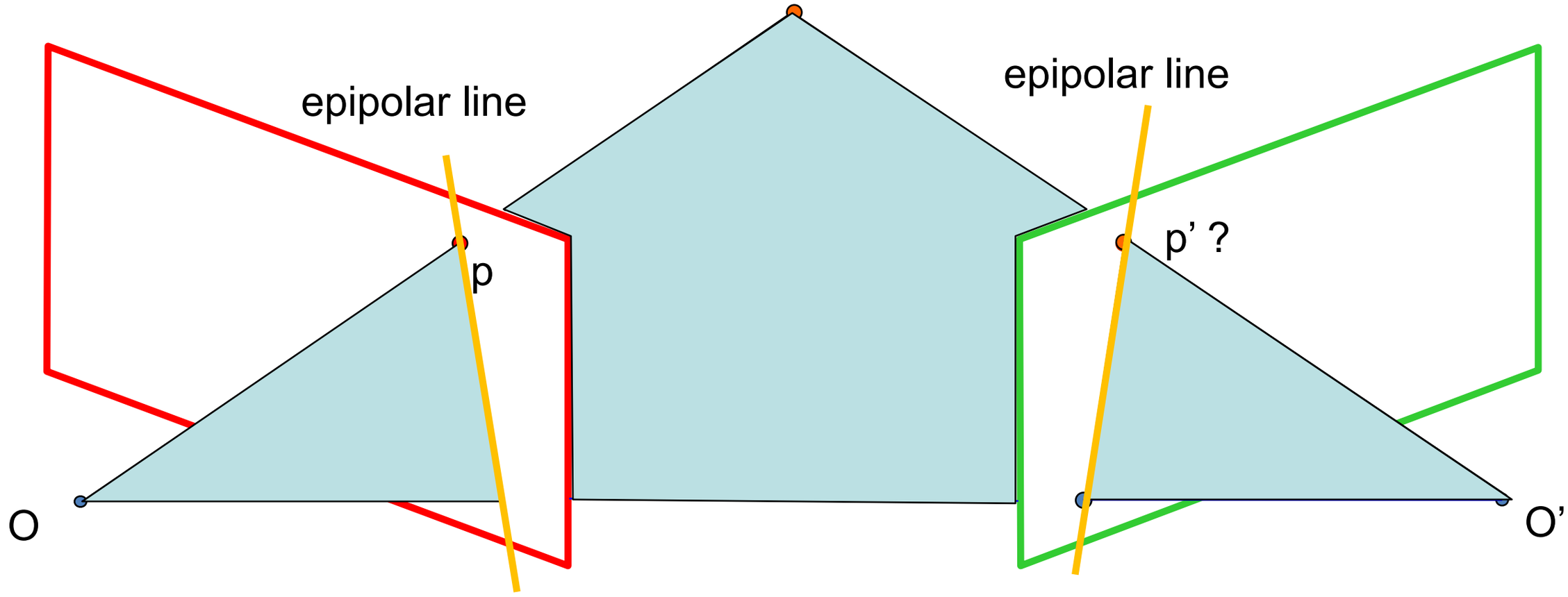
Baseline: the line connecting the two camera centers

Epipole: point of intersection of *baseline* with the image plane

Epipolar plane: the plane that contains the two camera centers and a 3D point in the world

Slide credit: Antonio Torralba

Some terminology



Baseline: the line connecting the two camera centers

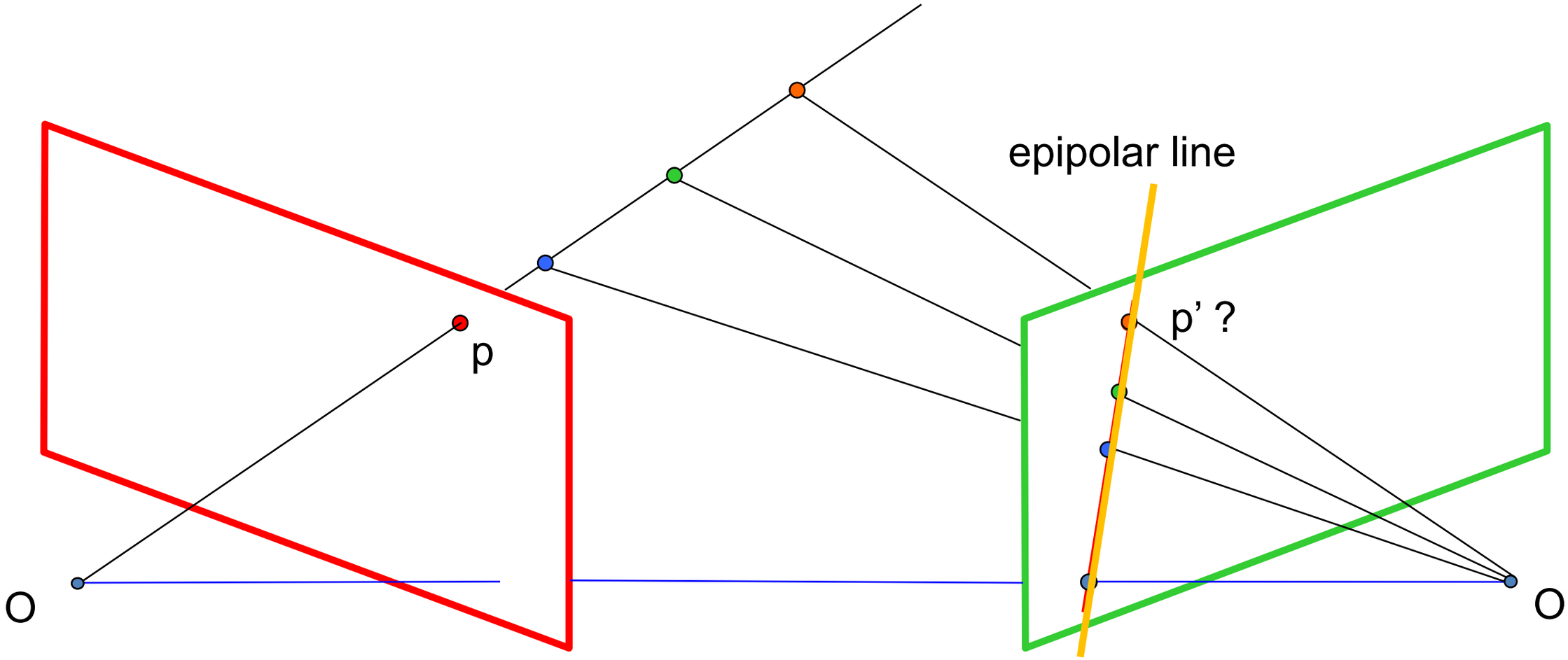
Epipole: point of intersection of *baseline* with the image plane

Epipolar plane: the plane that contains the two camera centers and a 3D point in the world

Epipolar line: intersection of the *epipolar plane* with each image plane

Slide credit: Antonio Torralba

Epipolar constraint

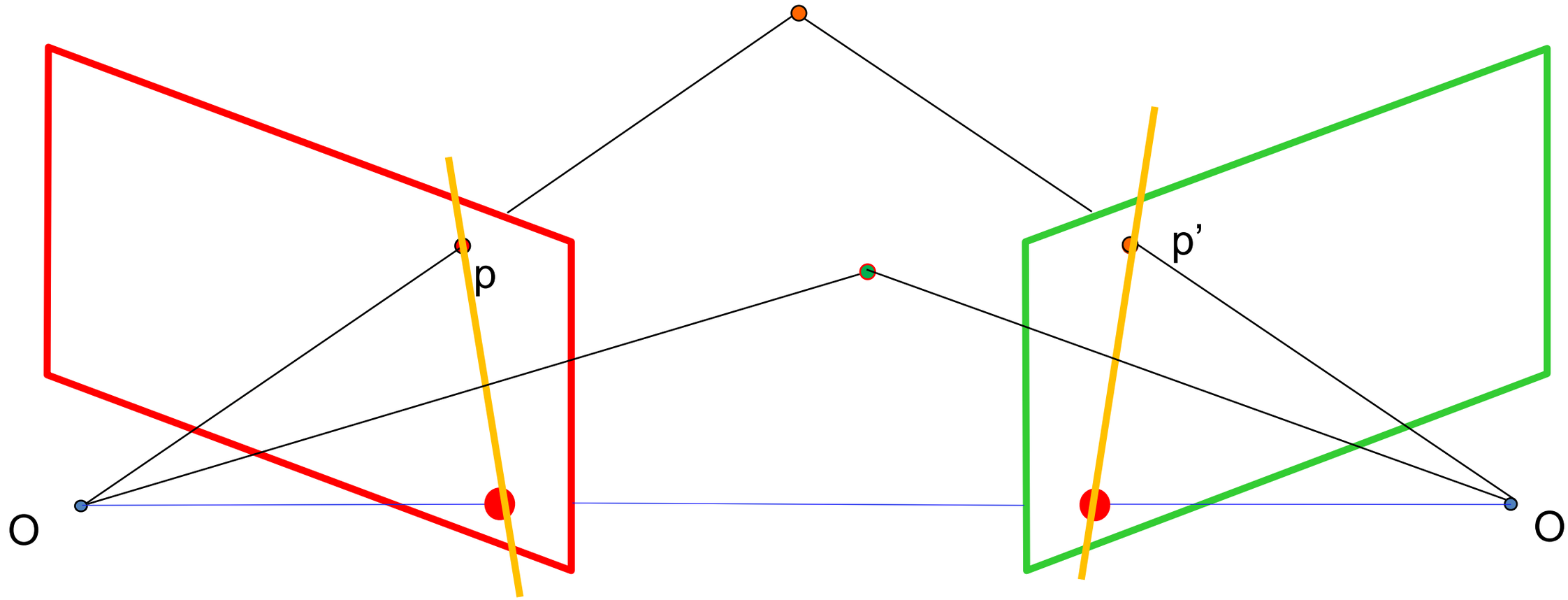


We can search for matches across epipolar lines

All epipolar lines intersect at the epipoles

Slide credit: Antonio Torralba

The essential matrix



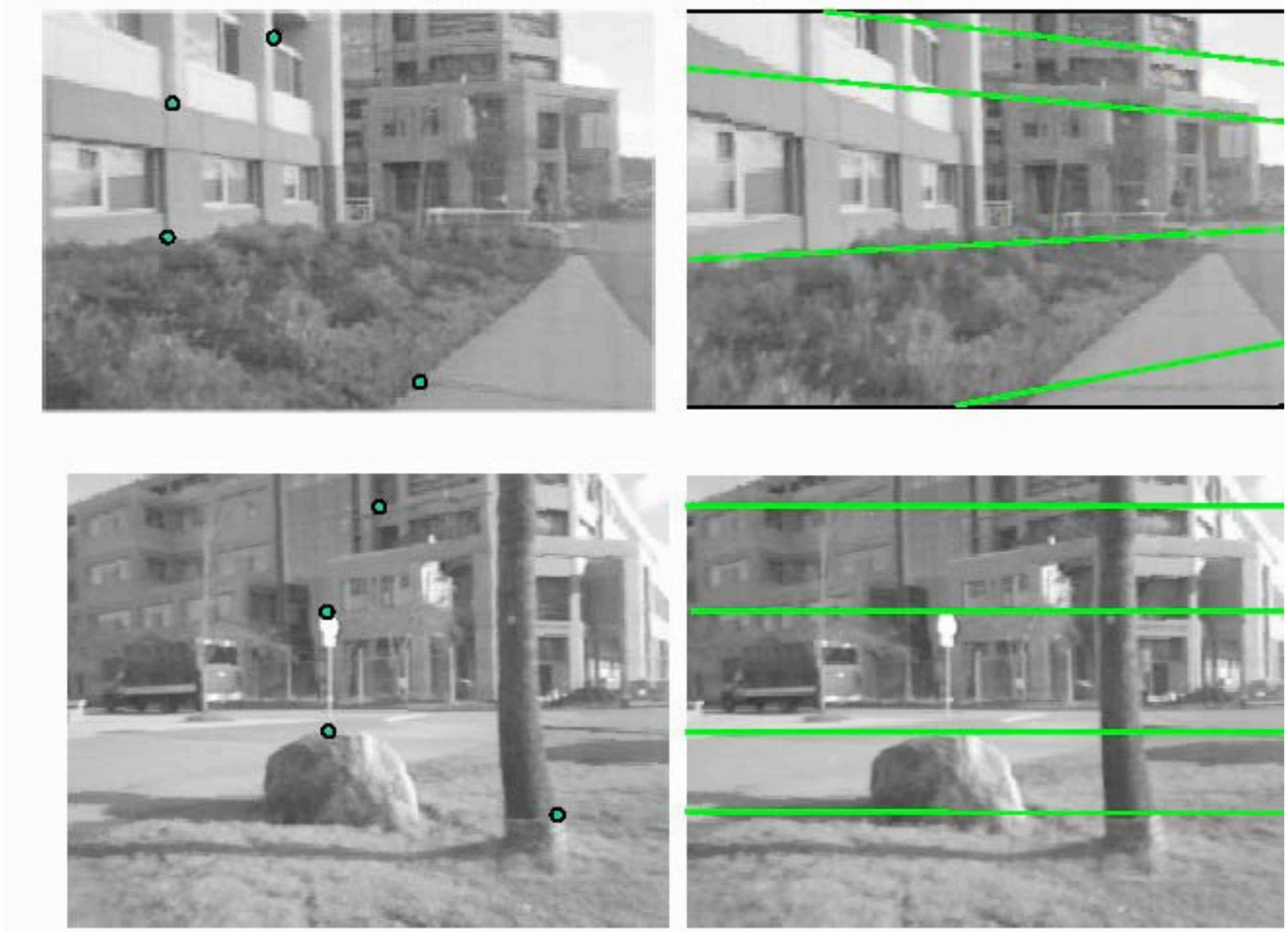
If we observe a point in one image, its position in the other image is constrained to lie on line defined by above.

$$p^T E p' = 0$$

E: essential matrix
p, p’: image points in homogeneous coordinates

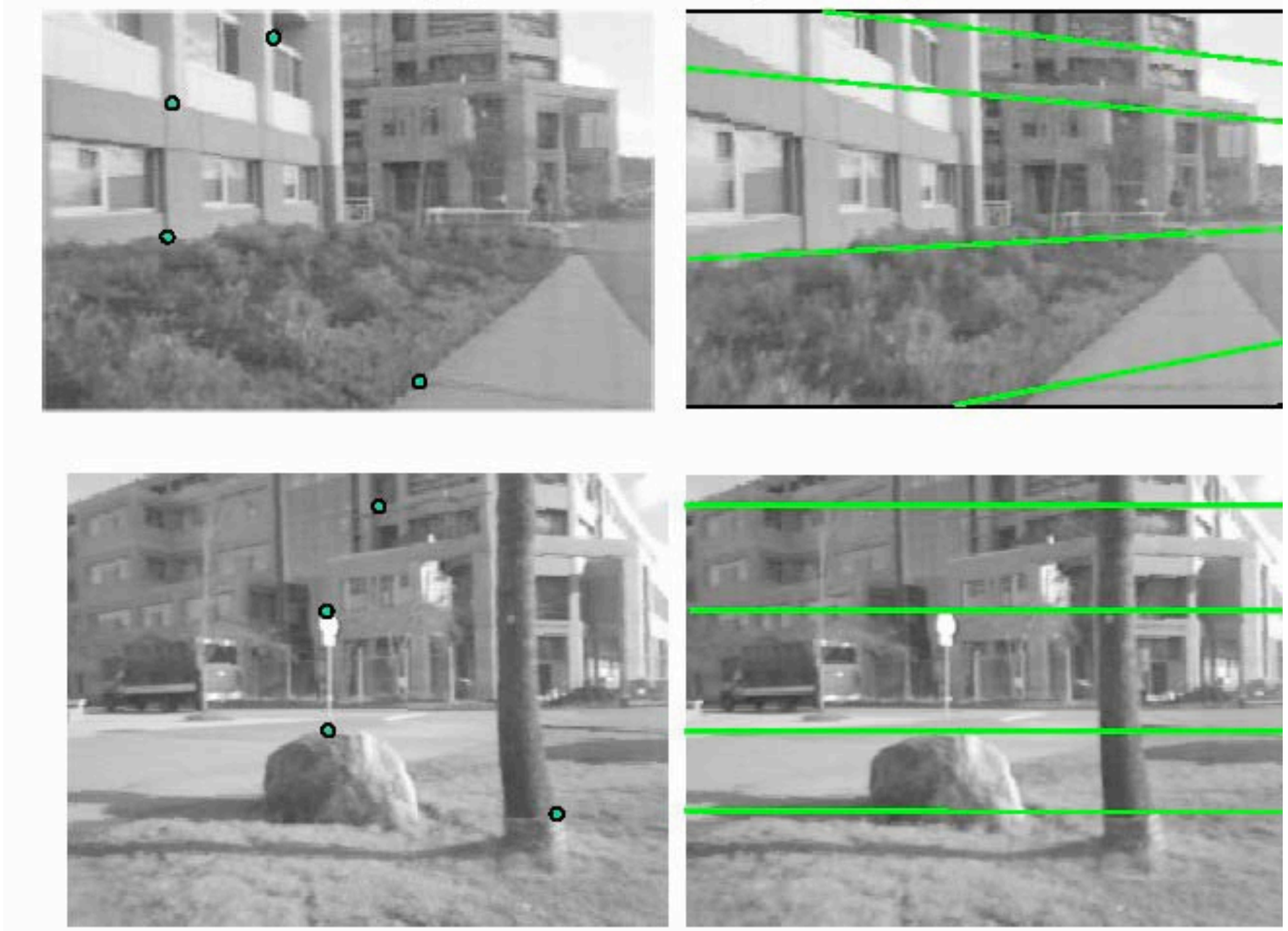
Slide credit: Antonio Torralba

Epipolar Examples



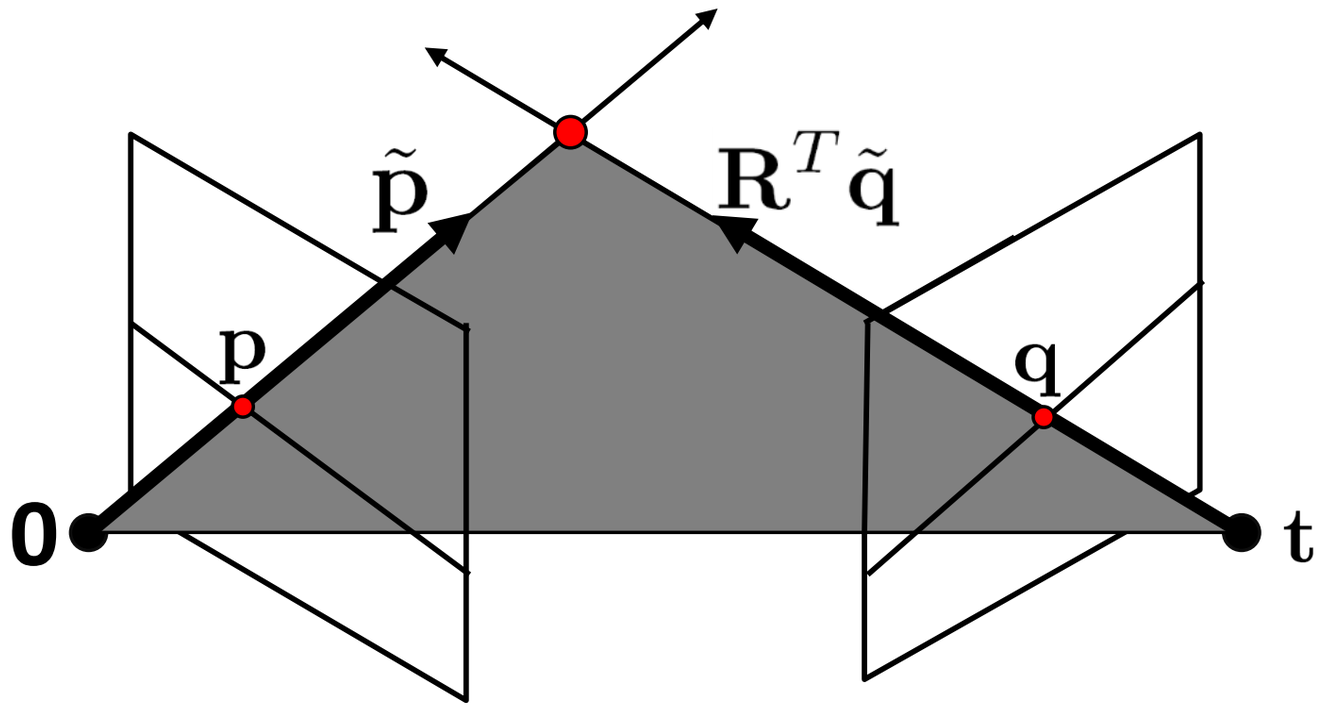
Source: S. Lazebnik

Where do they come from?



Source: S. Lazebnik

Fundamental matrix – calibrated case



$\mathbf{K}_1$: intrinsics of camera 1

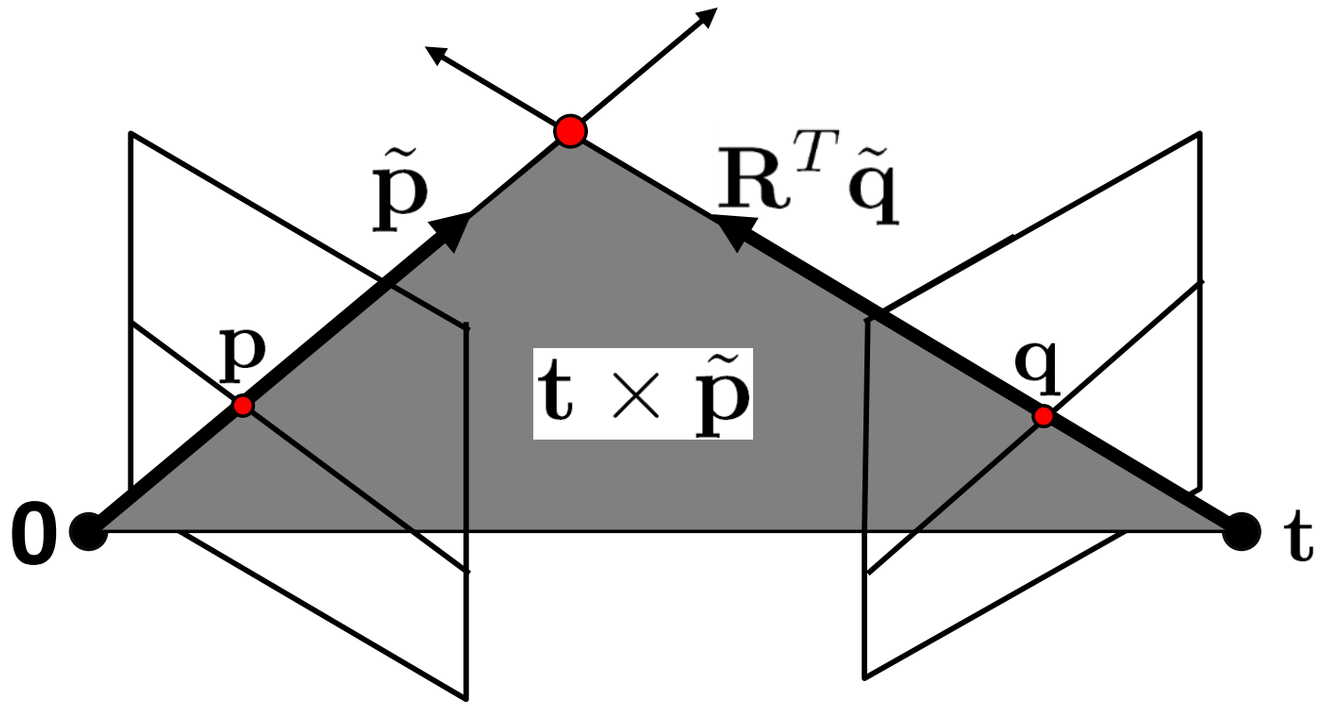
$\mathbf{K}_2$: intrinsics of camera 2

$\mathbf{R}$: rotation of image 2 w.r.t. camera 1

$\tilde{\mathbf{p}} = \mathbf{K}_1^{-1}\mathbf{p}$: ray through **p** in camera 1's (and world) coordinate system

$\tilde{\mathbf{q}} = \mathbf{K}_2^{-1}\mathbf{q}$: ray through **q** in camera 2's coordinate system

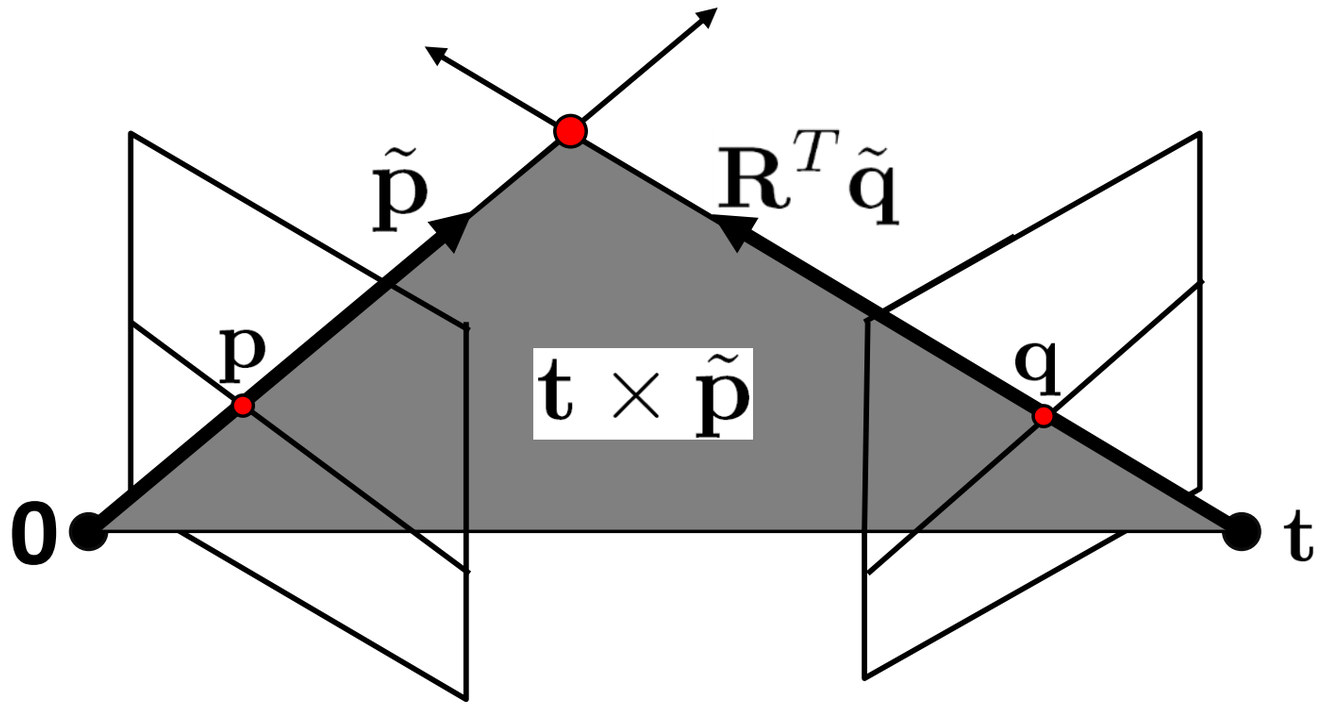
Fundamental matrix – calibrated case



- $\tilde{\mathbf{p}}$, $\mathbf{R}^T\tilde{\mathbf{q}}$, and $\mathbf{t}$ are coplanar
- epipolar plane can be represented as $\mathbf{t}\times\tilde{\mathbf{p}}$

$$(\mathbf{R}^T\tilde{\mathbf{q}})^T(\mathbf{t}\times\tilde{\mathbf{p}})=0$$

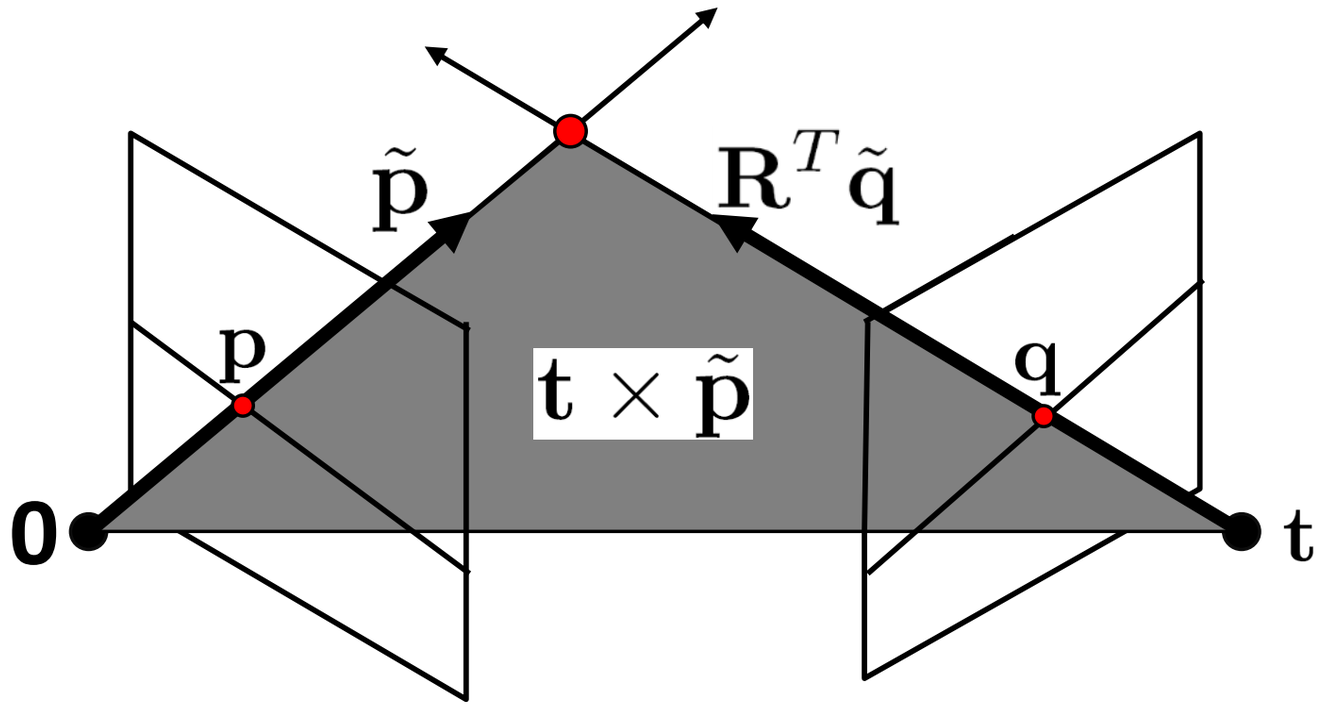
Fundamental matrix – calibrated case



- One more substitution:
  - Cross product with **t** can be represented as a 3x3 matrix

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \qquad \mathbf{t}\times\tilde{\mathbf{p}} = [\mathbf{t}]_\times \tilde{\mathbf{p}}$$

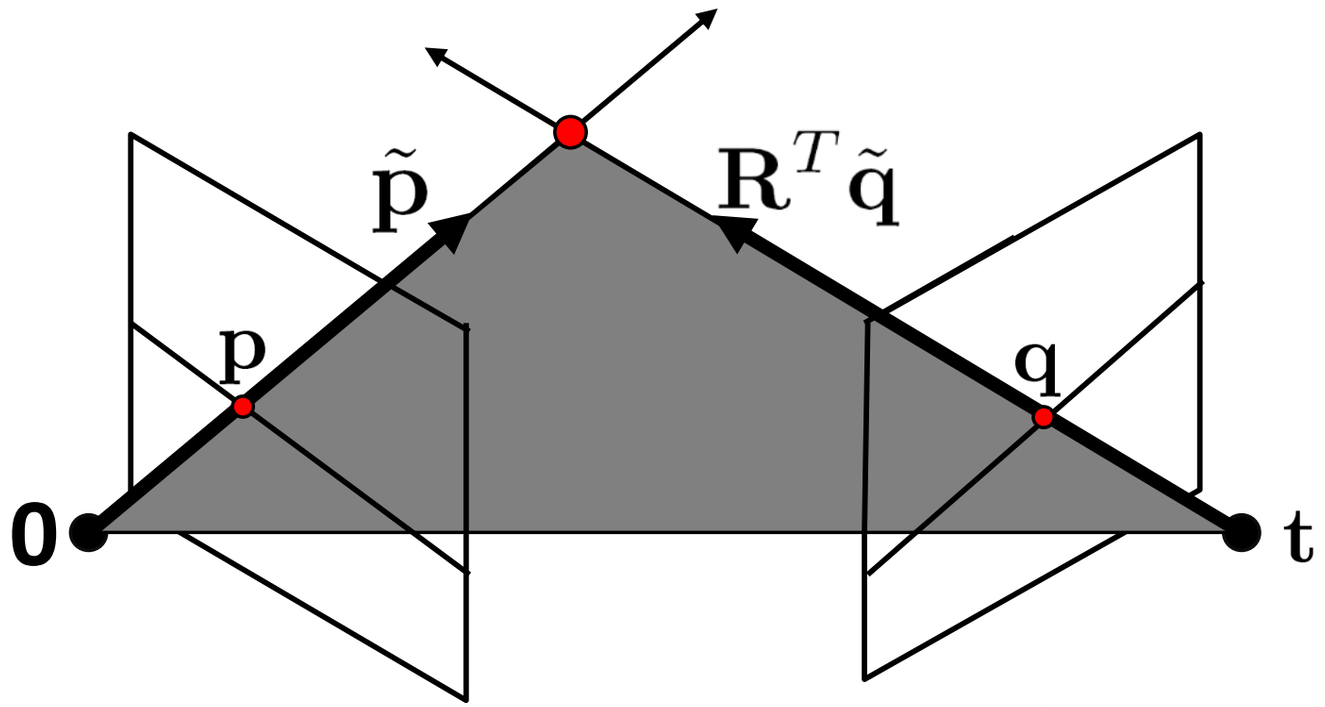
Fundamental matrix – calibrated case



$$\tilde{\mathbf{q}}^T\mathbf{R}(\mathbf{t}\times\tilde{\mathbf{p}})=0$$

$$\tilde{\mathbf{q}}^T\mathbf{R}\,[\mathbf{t}]_\times\,\tilde{\mathbf{p}}=0$$

Fundamental matrix – calibrated case



$\tilde{\mathbf{p}} = \mathbf{K}_1^{-1}\mathbf{p}$: ray through **p** in camera 1's (and world) coordinate system

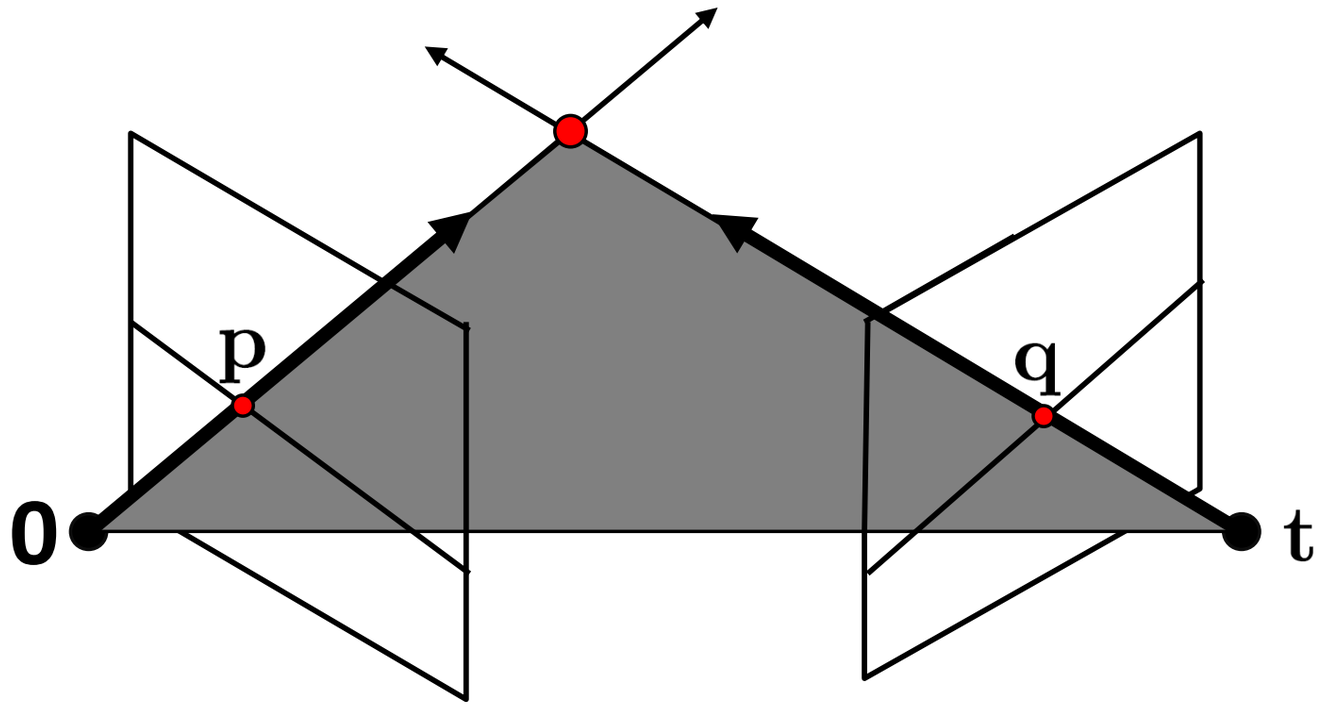
$\tilde{\mathbf{q}} = \mathbf{K}_2^{-1}\mathbf{q}$: ray through **q** in camera 2's coordinate system

$$\tilde{\mathbf{q}}^T \underbrace{\mathbf{R}\,[\mathbf{t}]_\times}_{\mathbf{E}} \tilde{\mathbf{p}} = 0$$

$\mathbf{E}$ ← the Essential matrix

$$\tilde{\mathbf{q}}^T \mathbf{E} \tilde{\mathbf{p}} = 0$$

Fundamental matrix – uncalibrated case



$\mathbf{K}_1$: intrinsics of camera 1

$\mathbf{K}_2$: intrinsics of camera 2

$\mathbf{R}$: rotation of image 2 w.r.t. camera 1

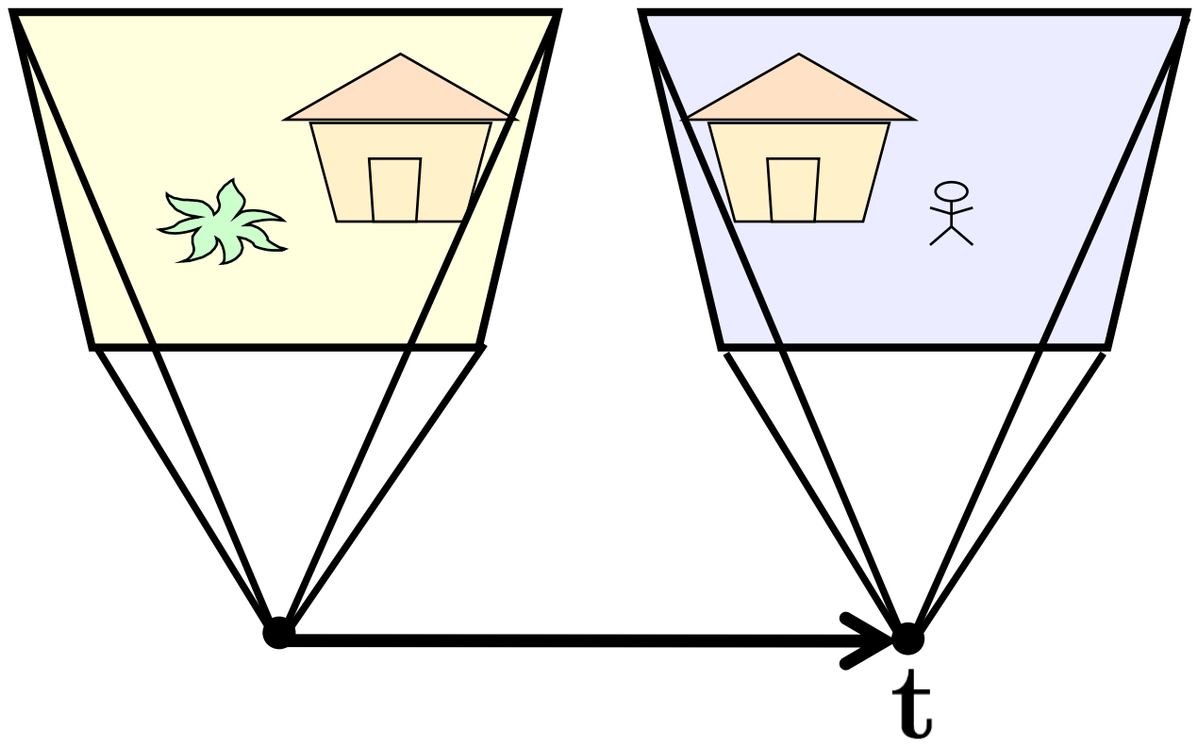
$$\mathbf{q}^T \underbrace{\mathbf{K}_2^{-T} \mathbf{R} \left[\mathbf{t}\right]_\times \mathbf{K}_1^{-1}}_{\mathbf{F}} \mathbf{p} = 0$$

$\mathbf{F}$ ← the Fundamental matrix

Properties of the Fundamental Matrix

- $\mathbf{F}\mathbf{p}$ is the epipolar line associated with $\mathbf{p}$
- $\mathbf{F}^T\mathbf{q}$ is the epipolar line associated with $\mathbf{q}$
- $\mathbf{F}\mathbf{e}_1 = \mathbf{0}$ and $\mathbf{F}^T\mathbf{e}_2 = \mathbf{0}$
- $\mathbf{F}$ is rank 2
- How many parameters does **F** have?

Rectified case

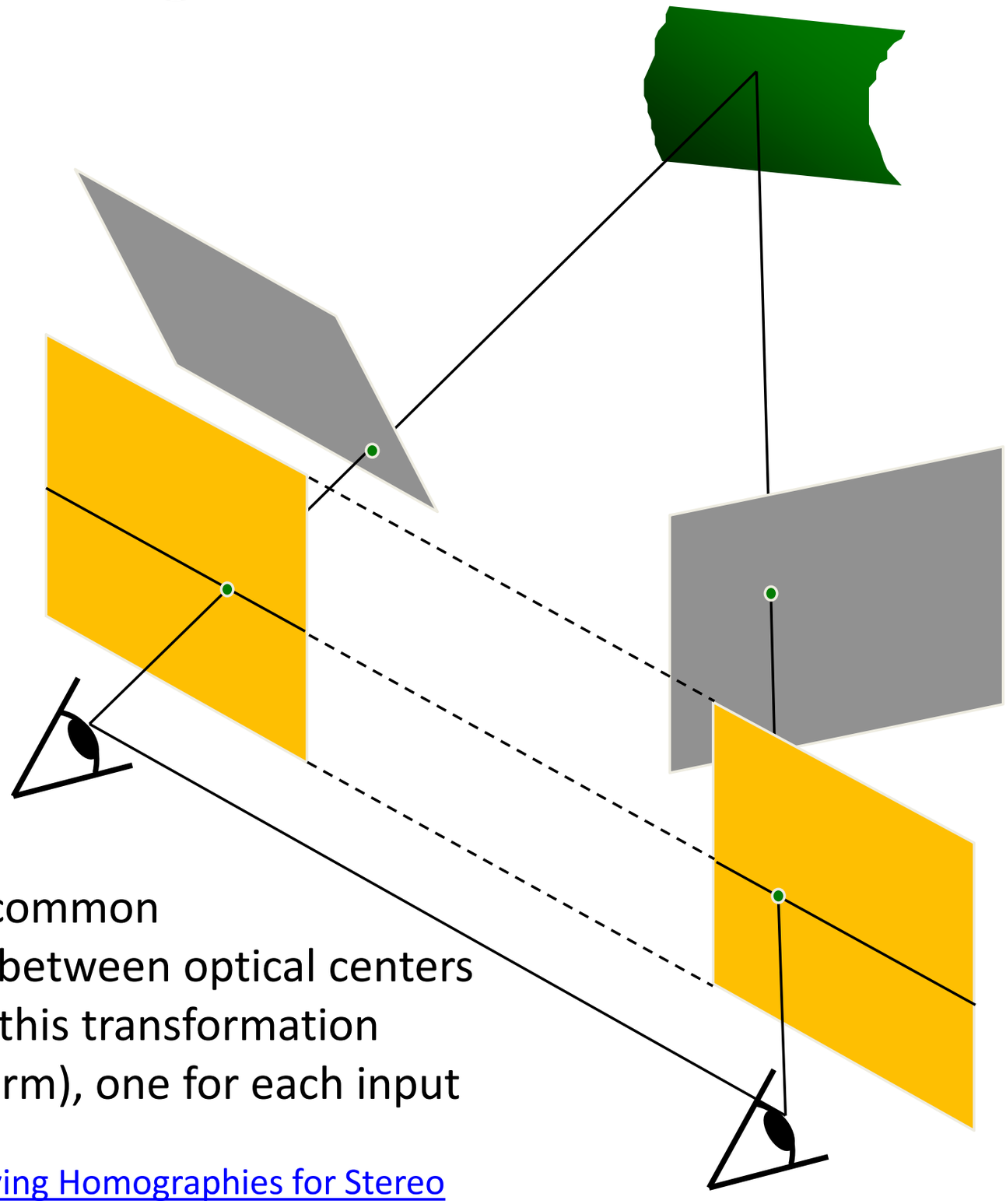


$$\mathbf{R} = \mathbf{I}_{3\times 3}$$

$$\mathbf{t} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$$

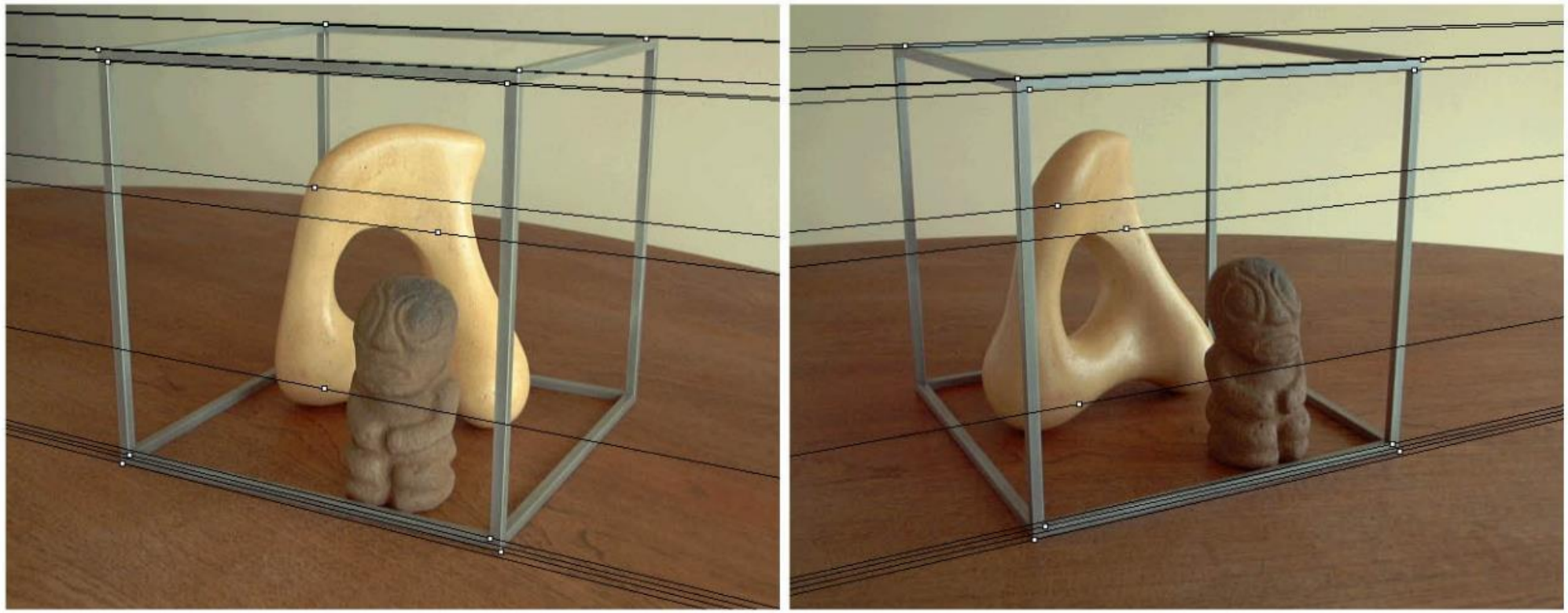
$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Stereo image rectification

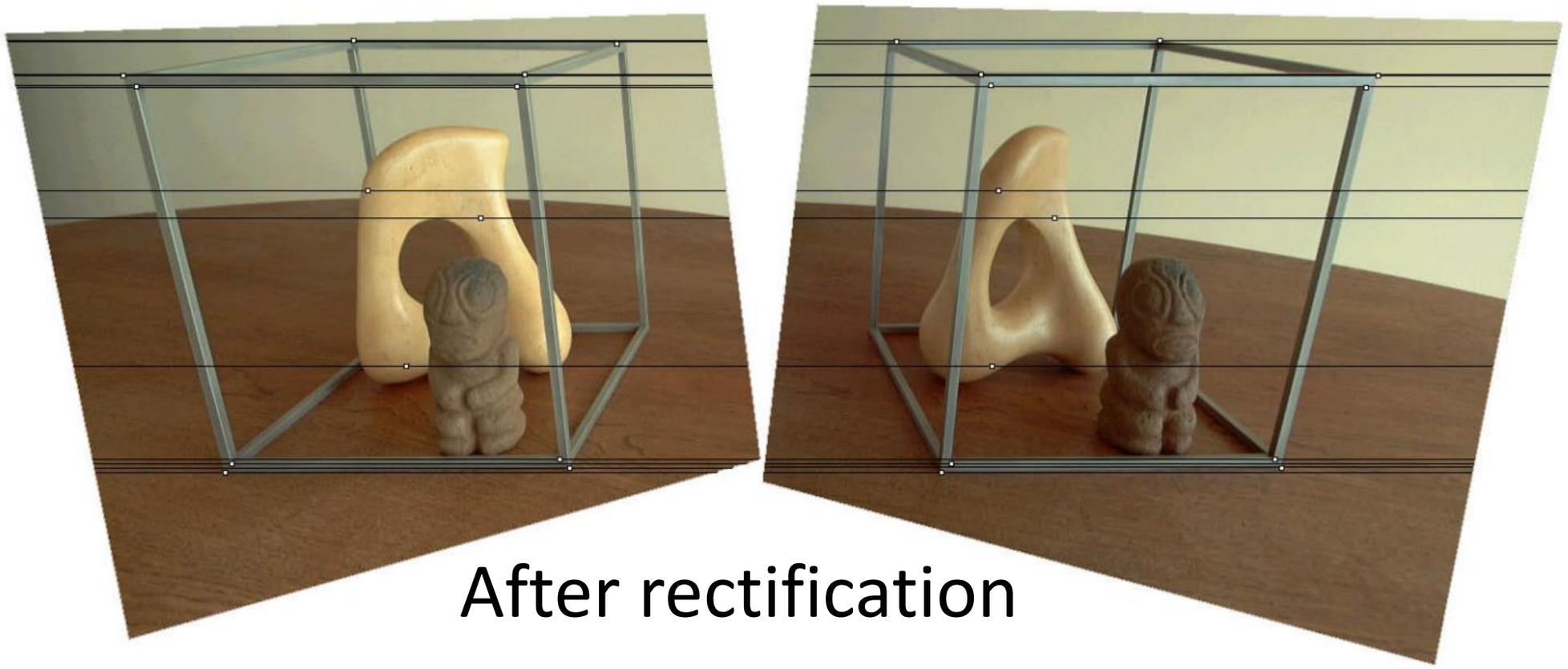


- reproject image planes onto a common
- plane parallel to the line between optical centers
- pixel motion is horizontal after this transformation
- two homographies (3x3 transform), one for each input image reprojection

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



Original stereo pair



After rectification

Estimating **F**

- If we don't know $\mathbf{K}_1$, $\mathbf{K}_2$, **R**, or **t**, can we estimate **F** for two images?

- Yes, given enough correspondences

Estimating F – 8-point algorithm

- The fundamental matrix F is defined by

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

for any pair of matches x and x’ in two images.

- Let x=(*u,v,1*)$^{\mathrm{T}}$ and x’=(*u’,v’,1*)$^{\mathrm{T}}$,

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

each match gives a linear equation

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

8-point algorithm

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

We want to solve the linear system: $Af = 0$

But, this has a trivial solution of f = 0.

8-point algorithm

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

We want to solve the linear system: $Af = 0$

The solution f is the eigenvector corresponding to the smallest eigenvalue of A^TA

8-point algorithm – Problem?

- **F** should have rank 2
- To enforce that **F** is of rank 2, F is replaced by F' that minimizes $\|\mathbf{F} - \mathbf{F}'\|$ subject to the rank constraint.
- This is achieved by SVD. Let $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$, where

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix}, \text{ let } \quad \Sigma' = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

then $\mathbf{F}' = \mathbf{U}\Sigma'\mathbf{V}^{\mathrm{T}}$ is the solution.

Problem with 8-point algorithm

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

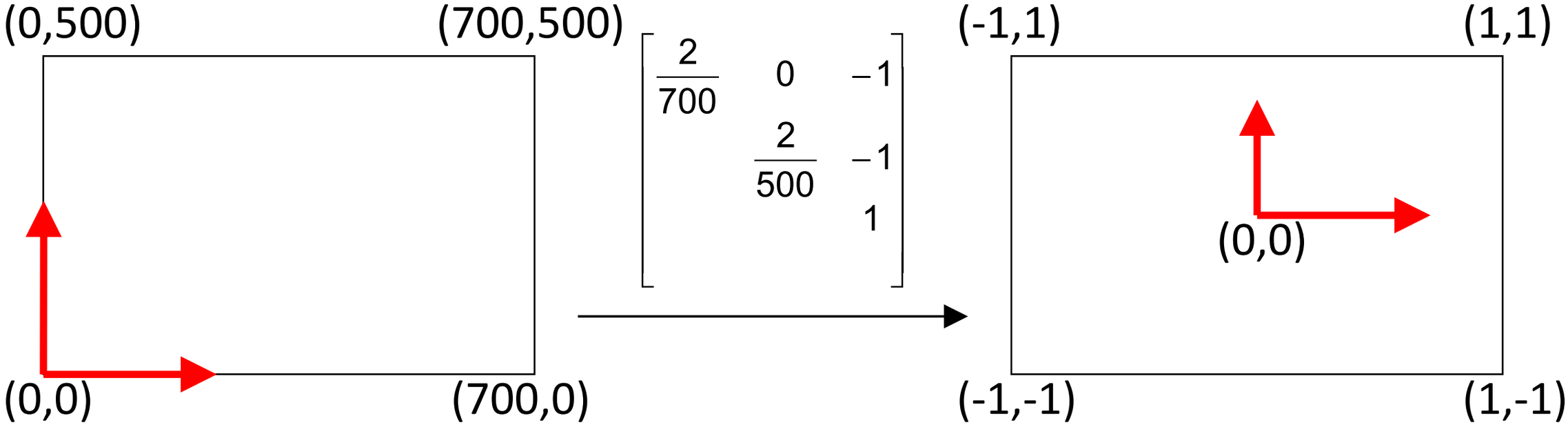
~10000 ~10000 ~100 ~10000 ~10000 ~100 ~100 ~100 1

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

Normalized 8-point algorithm

normalized least squares yields good results

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



Normalized 8-point algorithm

1. Transform input by $\hat{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i$, $\hat{\mathbf{x}}'_i = \mathbf{T}\mathbf{x}'_i$
2. Call 8-point on $\hat{\mathbf{x}}_i, \hat{\mathbf{x}}'_i$ to obtain $\hat{\mathbf{F}}$
3. $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

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

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

$$\hat{\mathbf{F}}$$

What about more than two views?

- The geometry of three views is described by a 3 x 3 x 3 tensor called the *trifocal tensor*

- The geometry of four views is described by a 3 x 3 x 3 x 3 tensor called the *quadrifocal tensor*

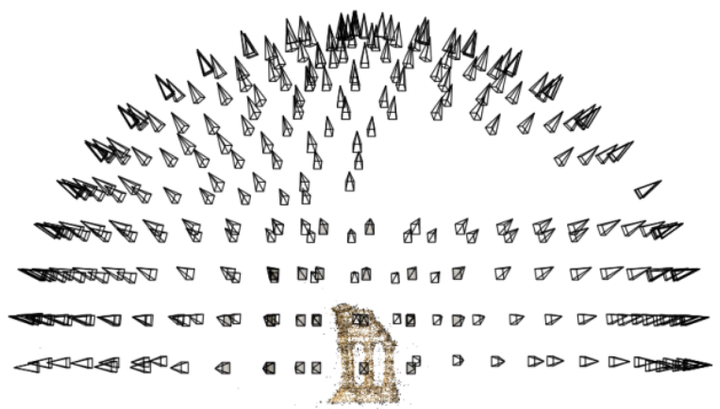
- After this it starts to get complicated…

Structure from motion

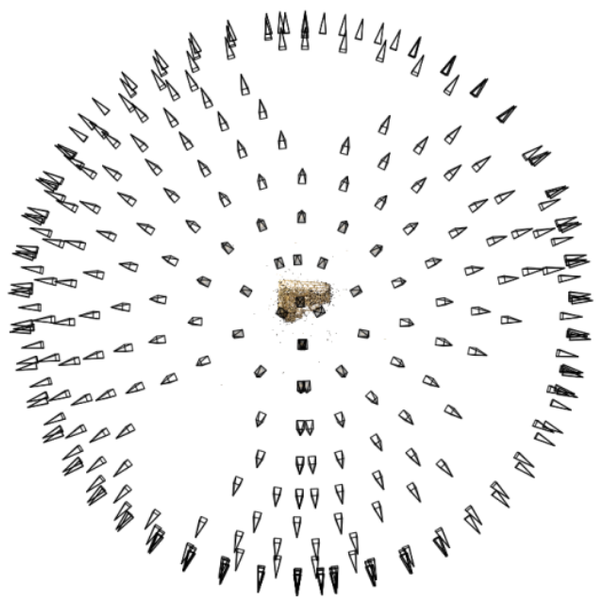
- Given many images, how can we
  a) figure out where they were all taken from?
  b) build a 3D model of the scene?

This is (roughly) the **structure from motion** problem

Structure from motion



Reconstruction (side)



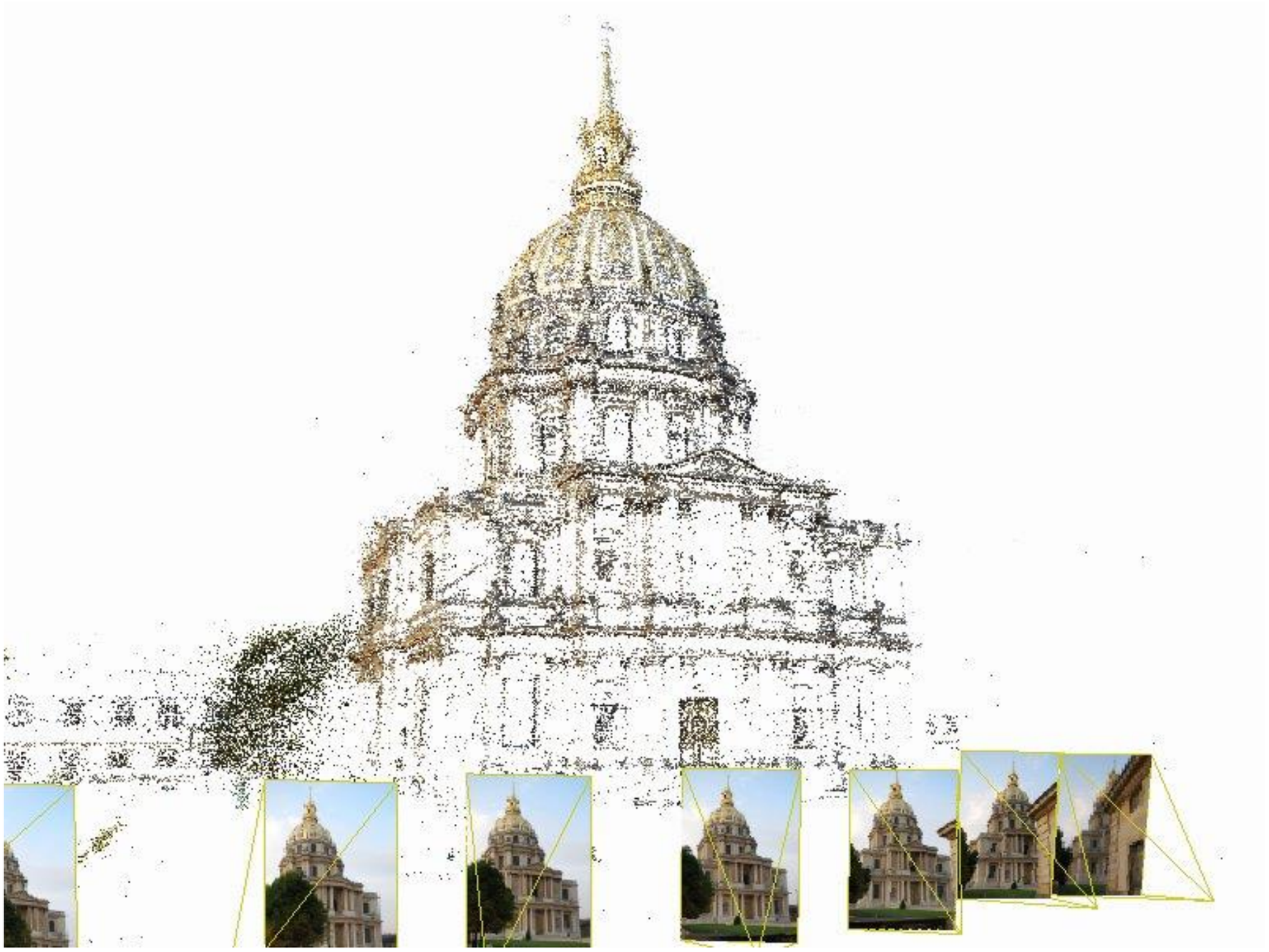
(top)

- Input: images with points in correspondence
  $p_{i,j} = (u_{i,j}, v_{i,j})$

- Output
  - structure: 3D location $\mathbf{x}_i$ for each point p_i
  - motion: camera parameters $\mathbf{R}_j$, $\mathbf{t}_j$ possibly $\mathbf{K}_j$

- Objective function: minimize *reprojection error*

Also doable from video



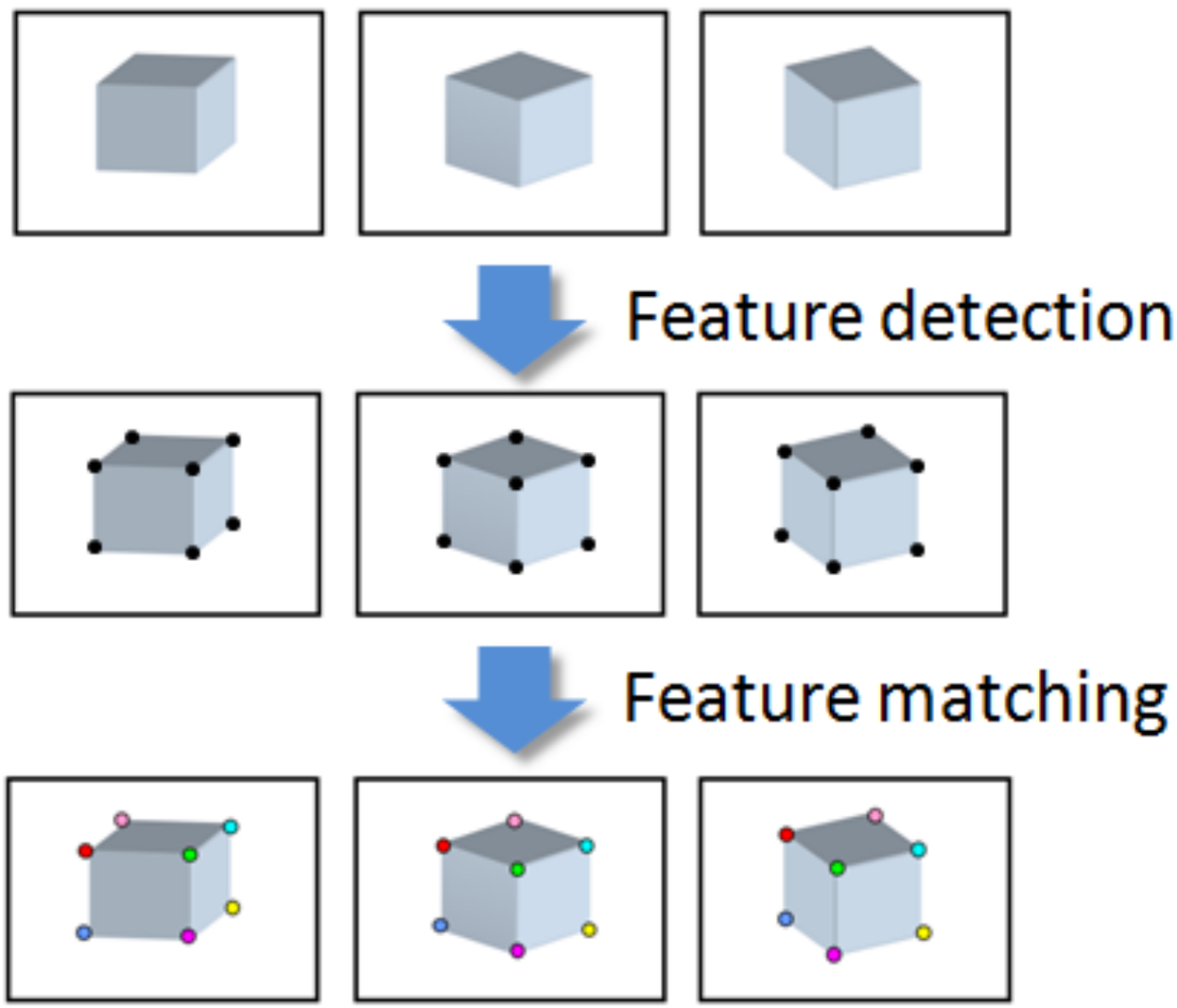
What we've seen so far…

- 2D transformations between images
  - Translations, affine transformations, homographies…

- Fundamental matrices
  - Still represent relationships between 2D images

- **What's new:** Explicitly representing 3D geometry of cameras *and points*

Input



Camera calibration and triangulation

- Suppose we know 3D points
  - And have matches between these points and an image
  - How can we compute the camera parameters?

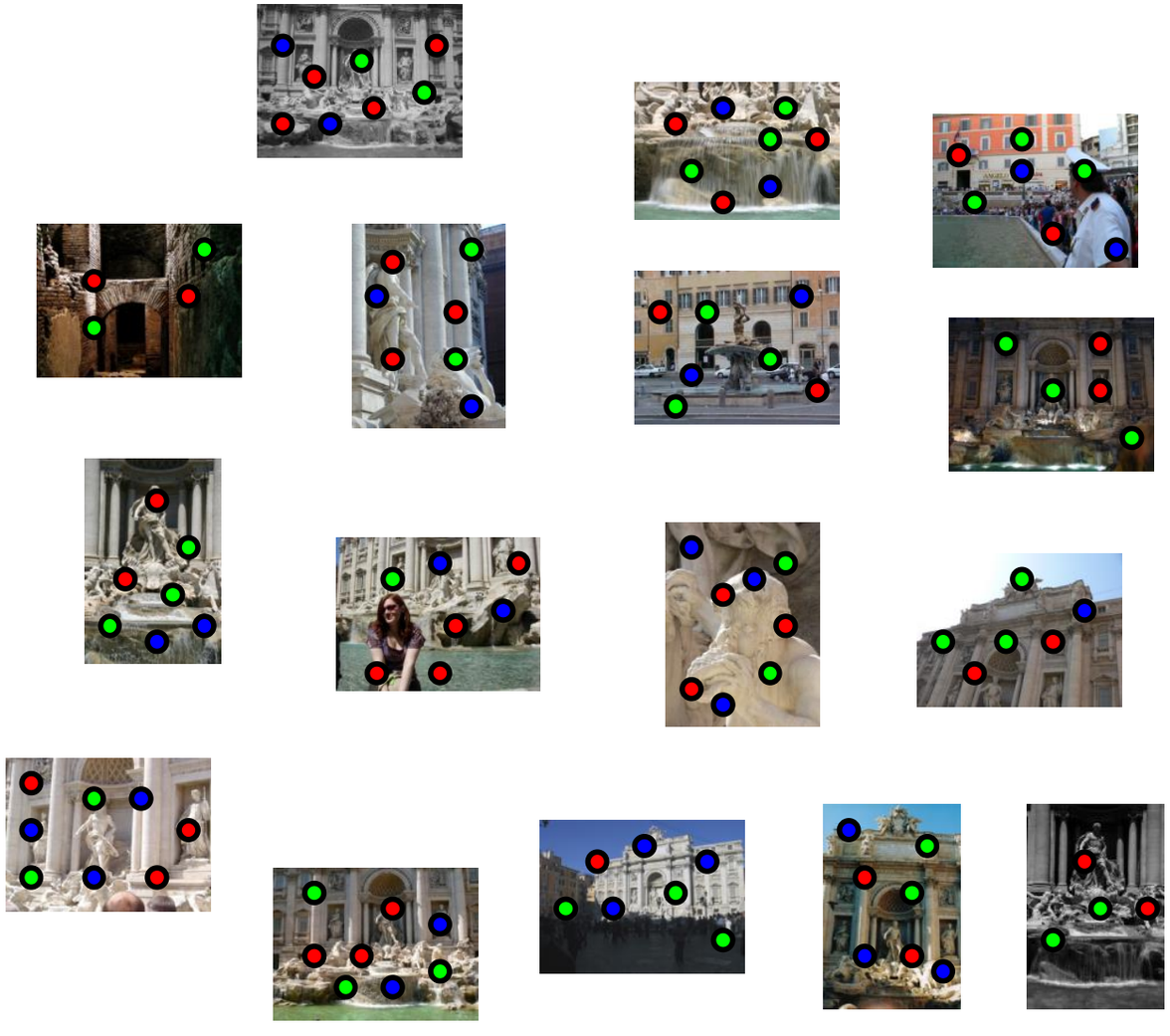
- Suppose we have know camera parameters, each of which observes a point
  - How can we compute the 3D location of that point?

Structure from motion

- SfM solves both of these problems *at once*
- A kind of chicken-and-egg problem
  - (but solvable)

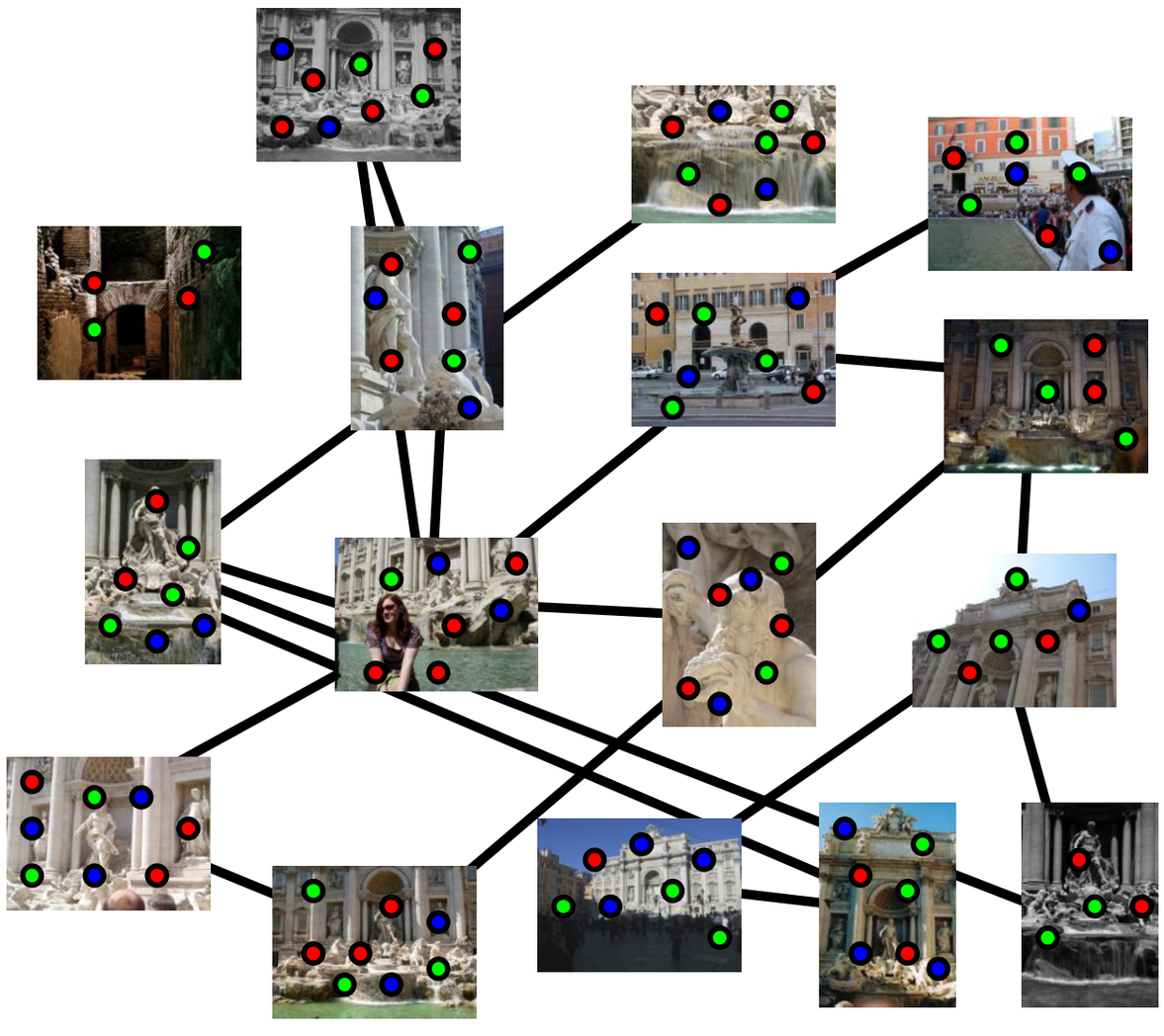
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



Feature matching

Match features between each pair of images



Feature matching

Refine matching using RANSAC to estimate fundamental matrix between each pair

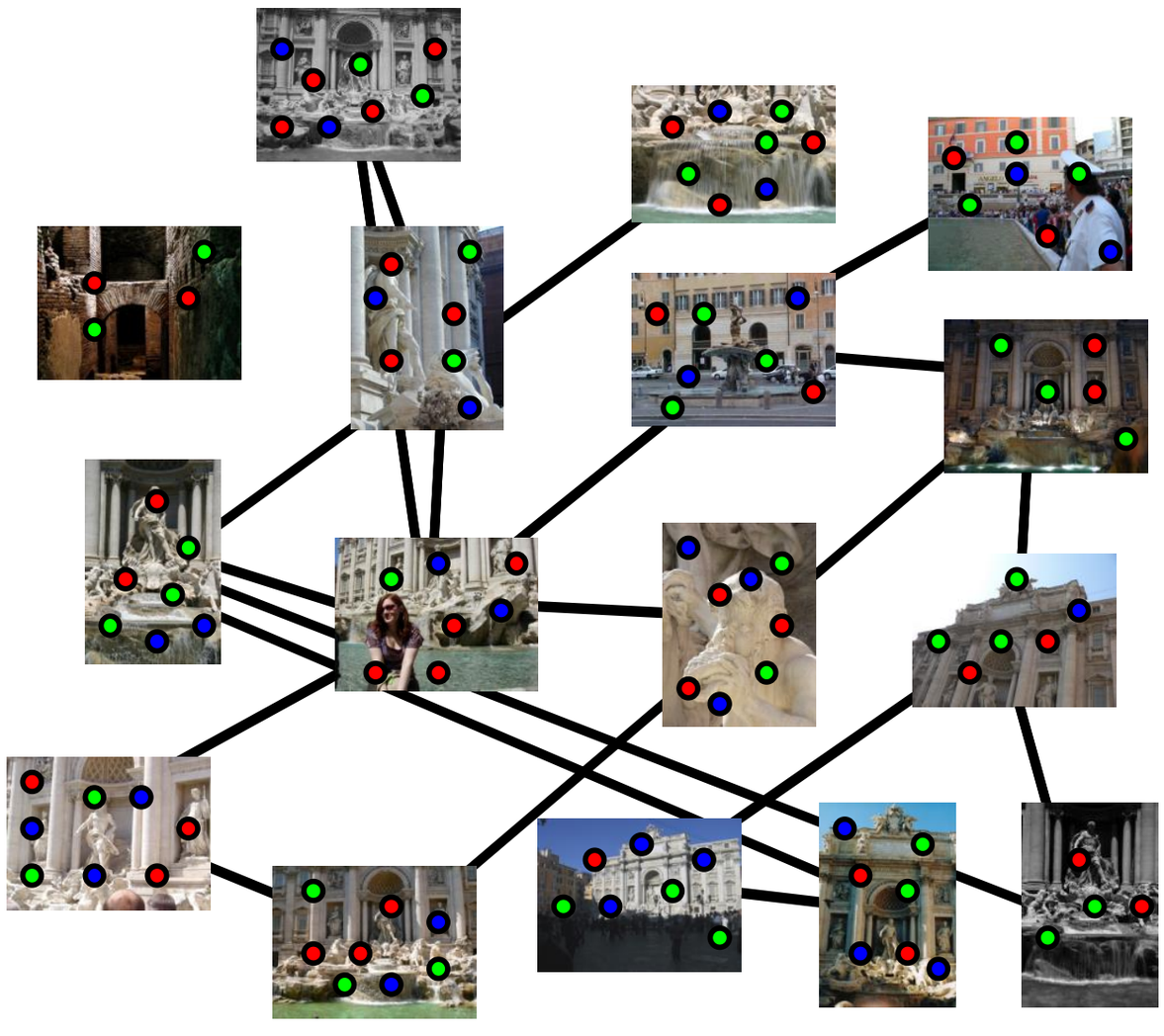
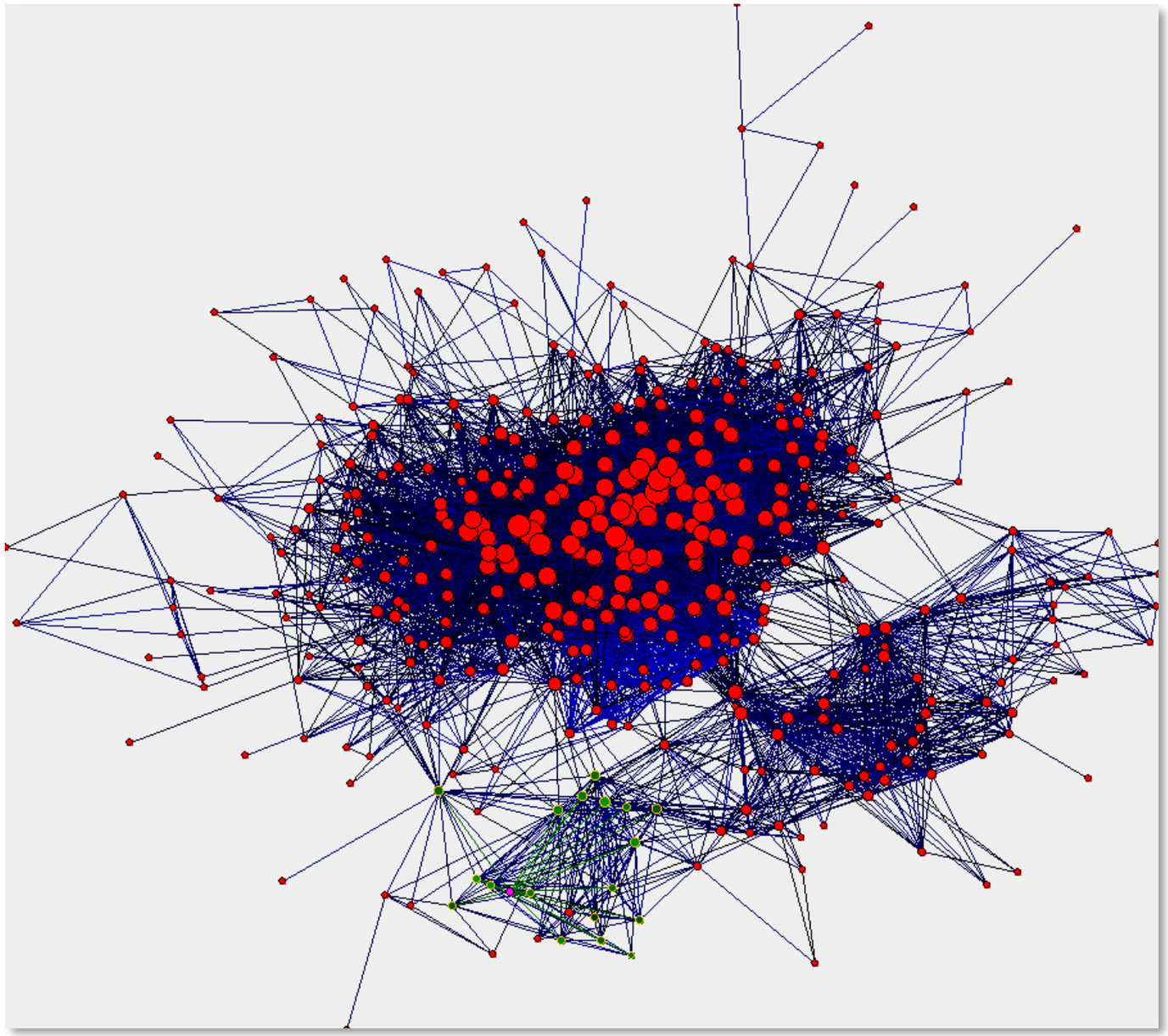


Image connectivity graph



(graph layout produced using the Graphviz toolkit: http://www.graphviz.org/)

Correspondence estimation

- Link up pairwise matches to form connected components of matches across several images

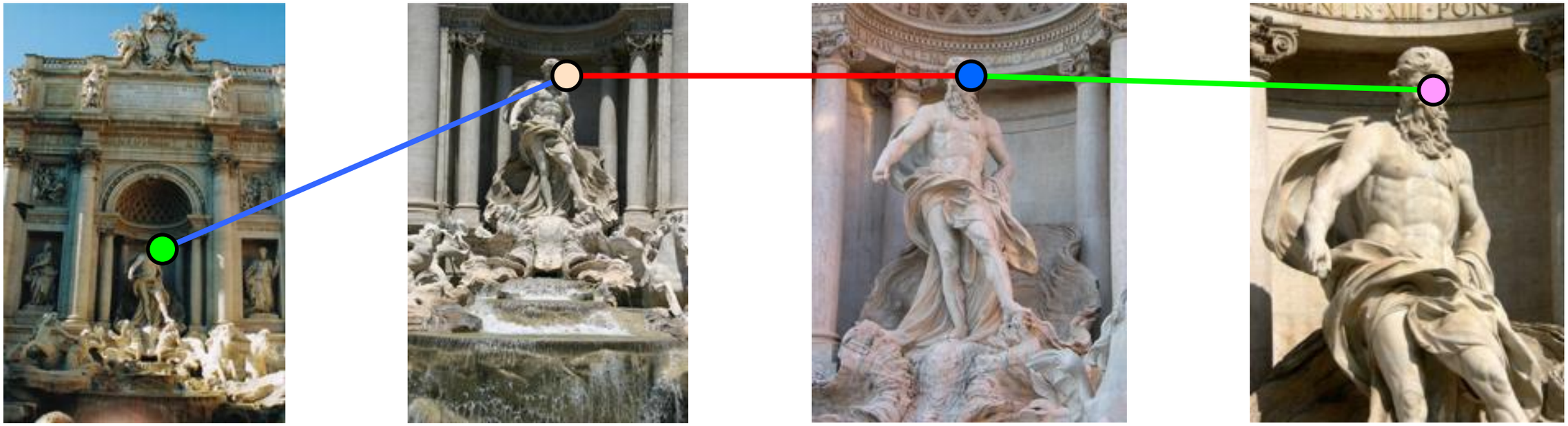
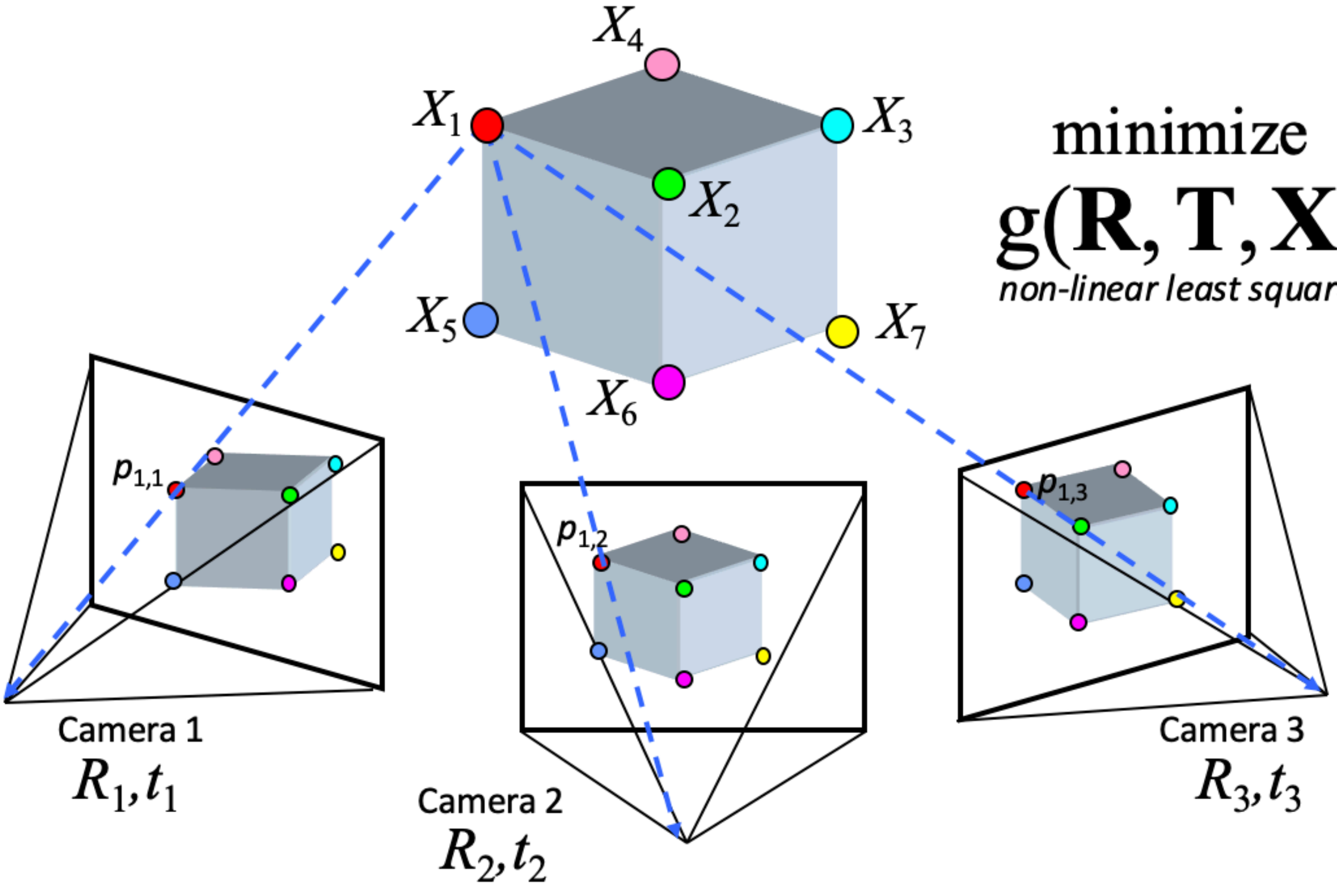


Image 1 Image 2 Image 3 Image 4

Structure from motion



minimize

$$g(\mathbf{R}, \mathbf{T}, \mathbf{X})$$

non-linear least squares

Structure from motion

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X},\mathbf{R},\mathbf{T}) = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j) - \begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} \right\|^2$$

w_{ij}: *indicator variable*: is point *i* visible in image *j* ?

$\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)$: *predicted* image location

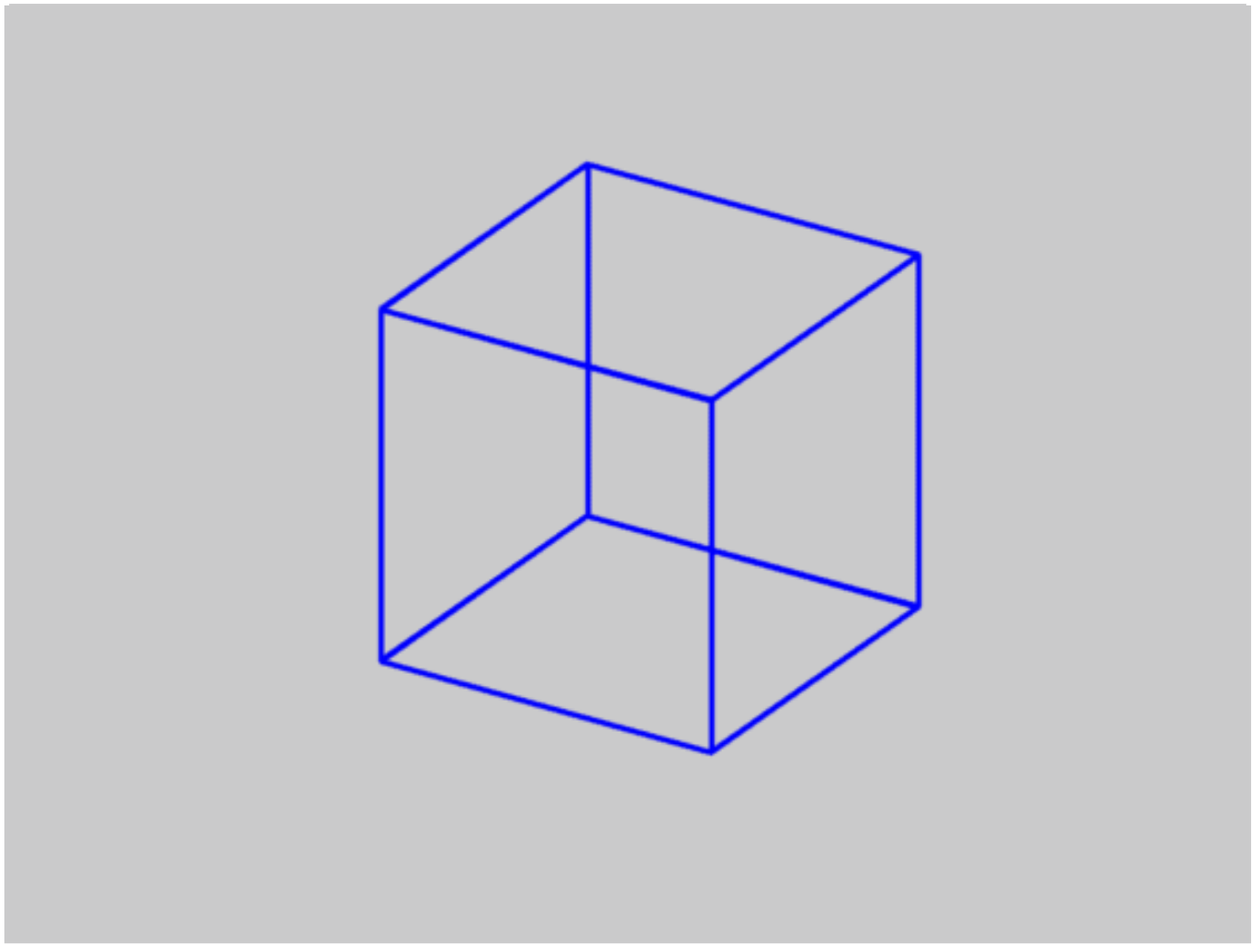
$\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}$: *observed* image location

- Minimizing this function is called *bundle adjustment*
  - Optimized using non-linear least squares, e.g. Levenberg-Marquardt

Is SfM always uniquely solvable?

Is SfM always uniquely solvable?

- No…



50 images of Rome

200 images of Rome

800 images of Rome


20,000 images of Rome

1,000,000 images of Rome

Building Rome in a Day

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, Richard Szeliski

Building Rome in a Day

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, Richard Szeliski

Building Rome in a Day

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz, Richard Szeliski

First-person Hyperlapse Videos

Johannes Kopf Michel F. Cohen Richard Szeliski
Microsoft Research

research.microsoft.com/hyperlapse