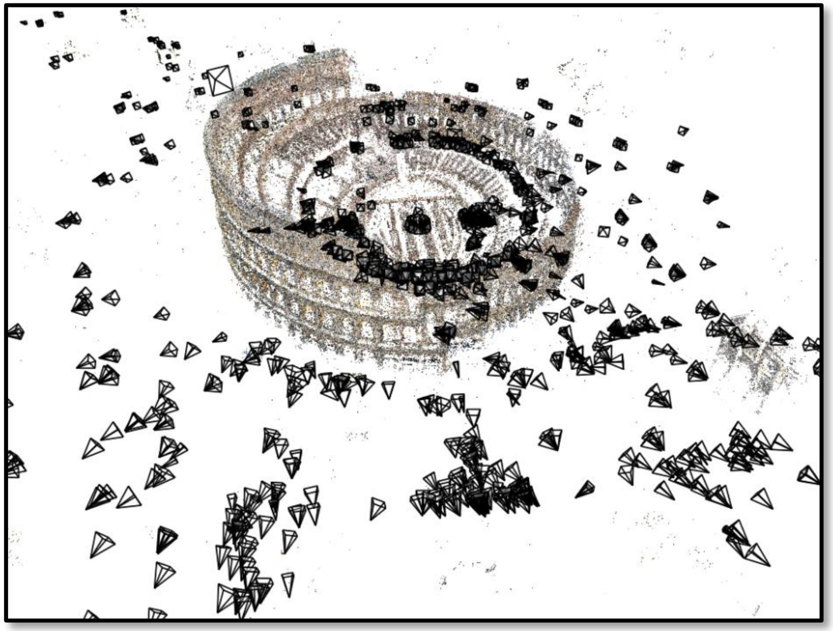
CS6670: Computer Vision

Noah Snavely

Lecture 21: Multiple-view geometry and structure from motion



Readings

- Szeliski, Chapter 7.1 – 7.4

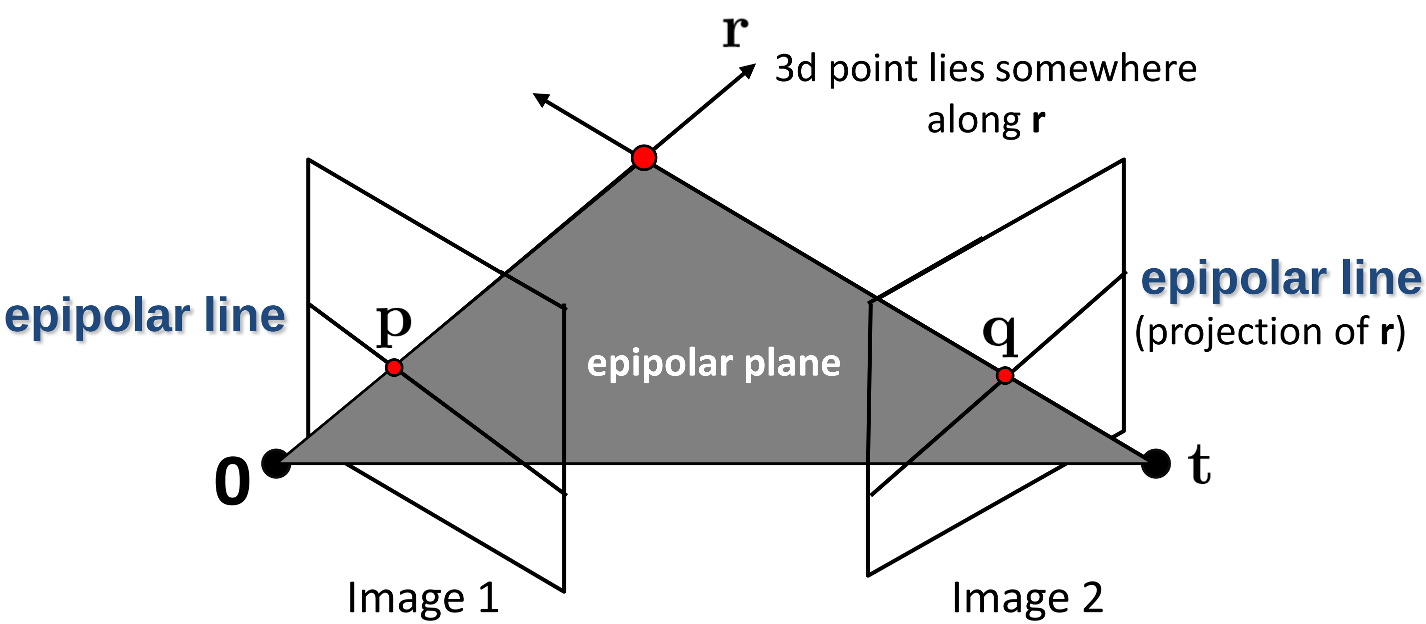
Announcements

- Project 2b due next Tuesday, Nov 2, by 10:59pm

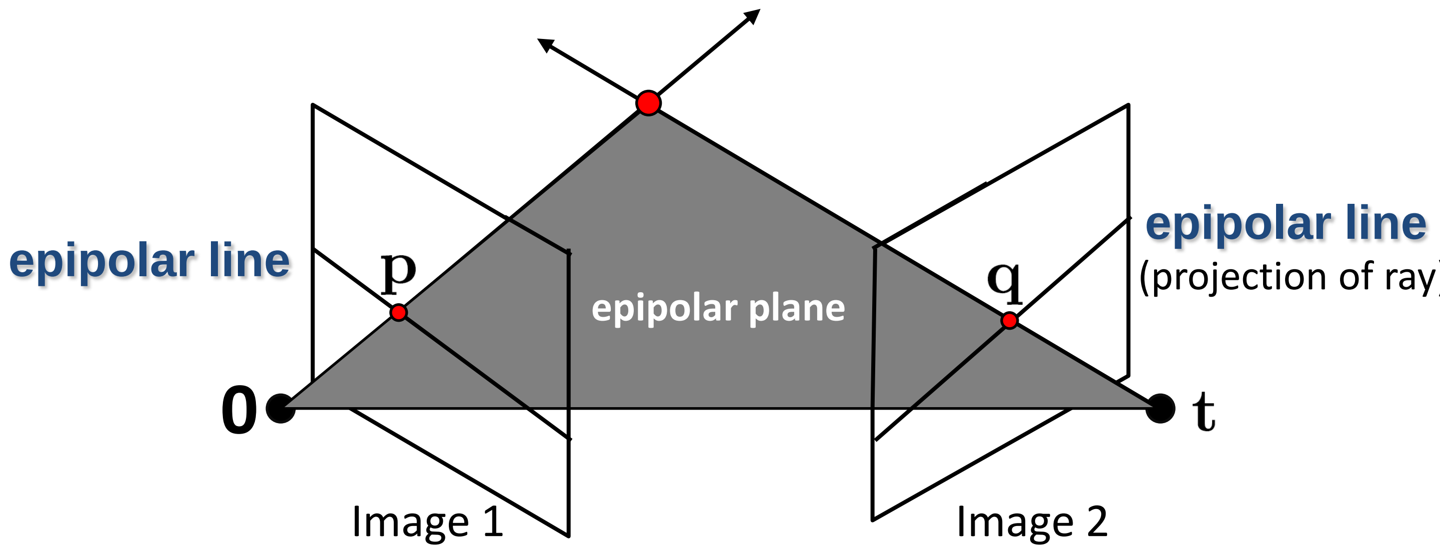
- Final project proposals due by this Friday, Oct 29, by 11:59pm
  - See project webpage for project ideas

Two-view geometry

- Where do epipolar lines come from?

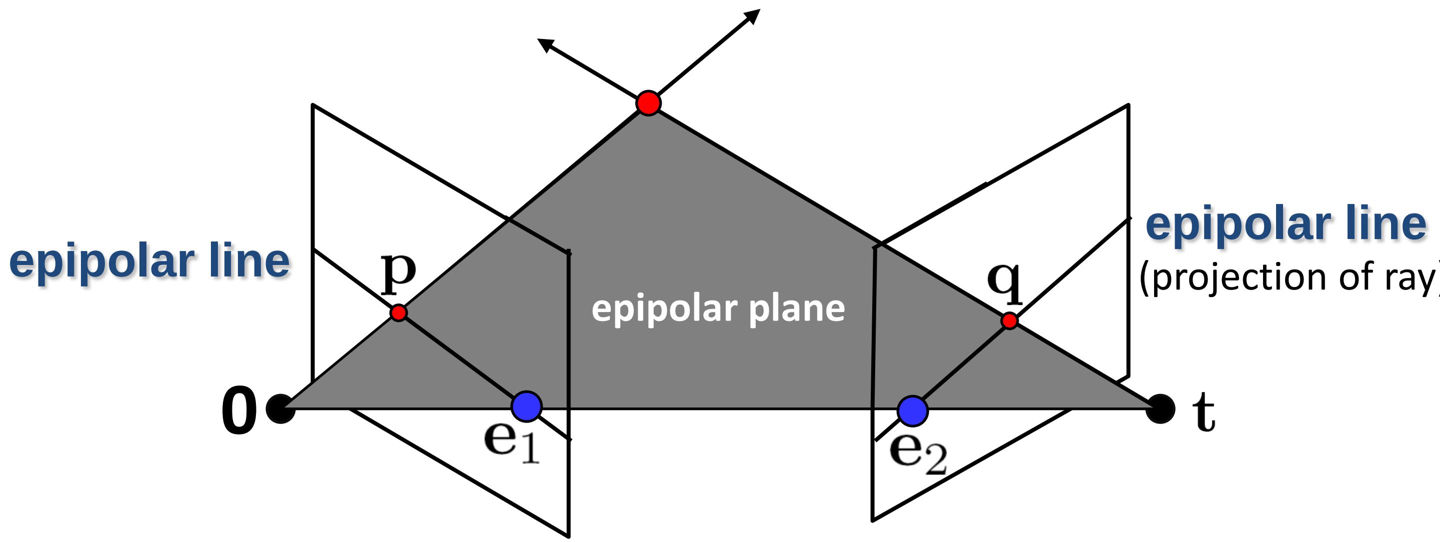


Fundamental matrix



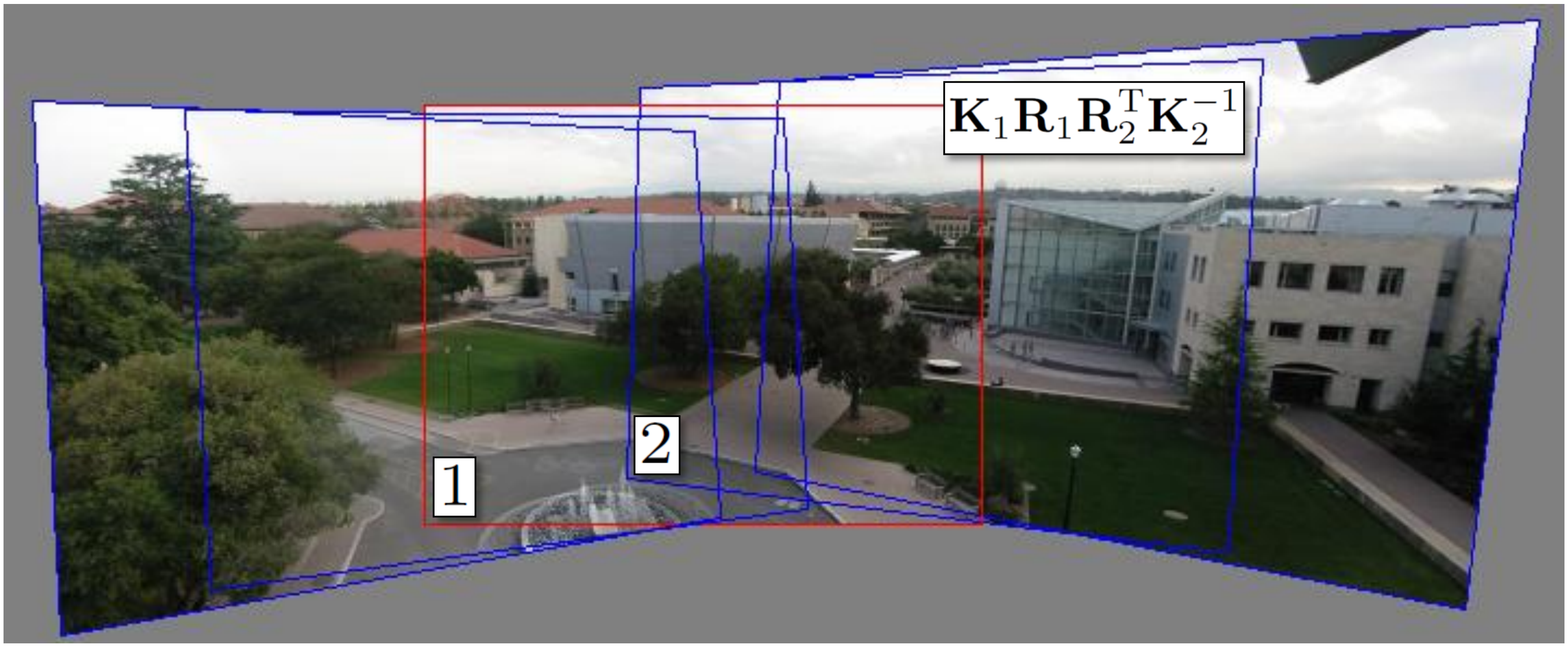
- This *epipolar geometry* of two views is described by a Very Special 3x3 matrix $\mathbf{F}$, called the *fundamental matrix*
- $\mathbf{F}$ maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point **p** is: $\mathbf{F}\mathbf{p}$
- *Epipolar constraint* on corresponding points: $\mathbf{q}^T\mathbf{F}\mathbf{p} = 0$

Fundamental matrix



- Two special points: $\mathbf{e}_1$ and $\mathbf{e}_2$ (the *epipoles*): projection of one camera into the other

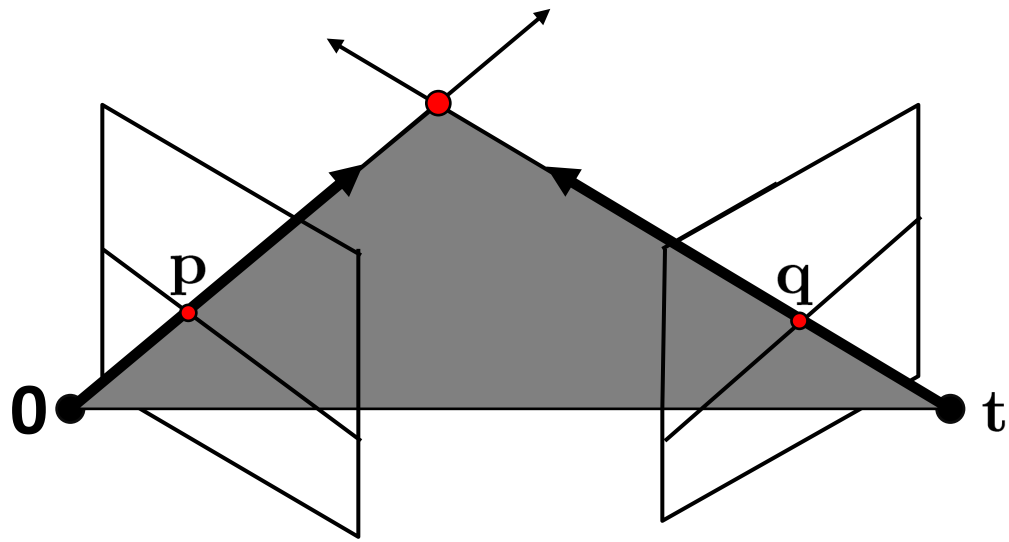
Relationship with homography?



Images taken from the same center of projection? Use a homography, and map points directly to points!

Images taken from different places? We need an **F**-matrix.

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

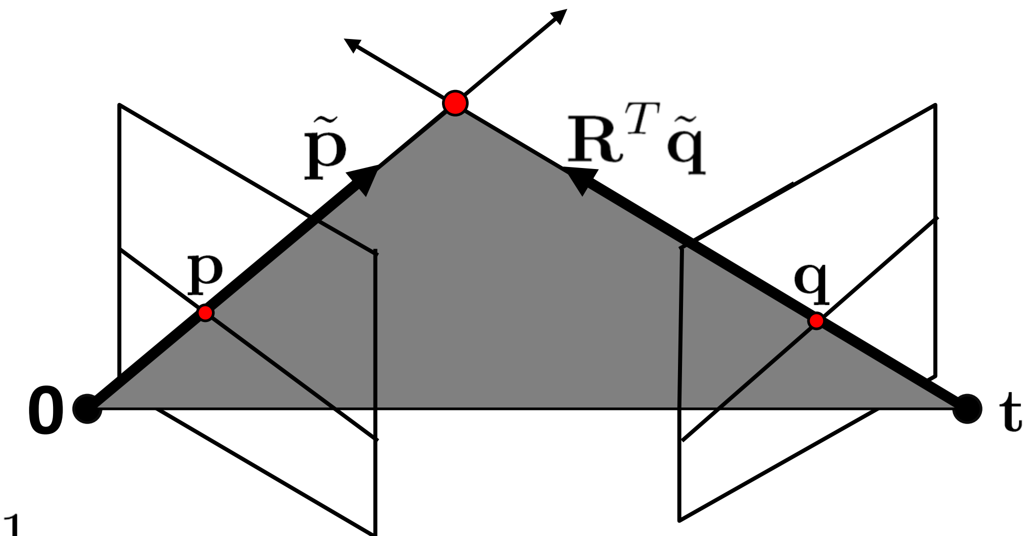
Cross-product as linear operator

Useful fact: Cross product with a vector **t** can be represented as multiplication with a (*skew-symmetric*) 3x3 matrix

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

$$\mathbf{t} \times \tilde{\mathbf{p}} = [\mathbf{t}]_\times \tilde{\mathbf{p}}$$

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

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

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

$\mathbf{E}$ ← the Essential matrix

Properties of the Fundamental Matrix

- $\mathbf{F}\mathbf{p}$ is the epipolar line associated with $\mathbf{p}$

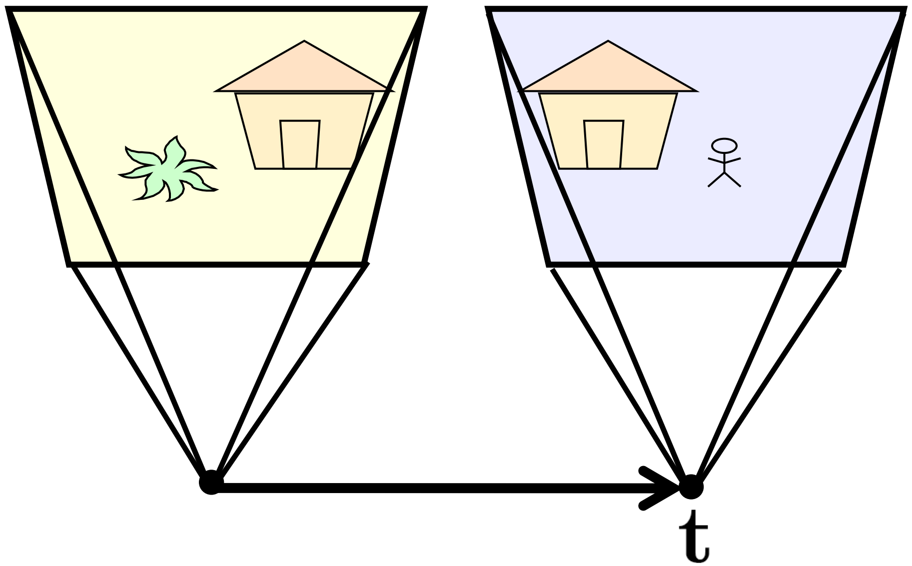
- $\mathbf{F}^T\mathbf{q}$ is the epipolar line associated with $\mathbf{q}$

- $\mathbf{F}\mathbf{e}_1 = \mathbf{0}$ and $\mathbf{F}^T\mathbf{e}_2 = \mathbf{0}$

- $\mathbf{F}$ is rank 2

- How many parameters does **F** have?

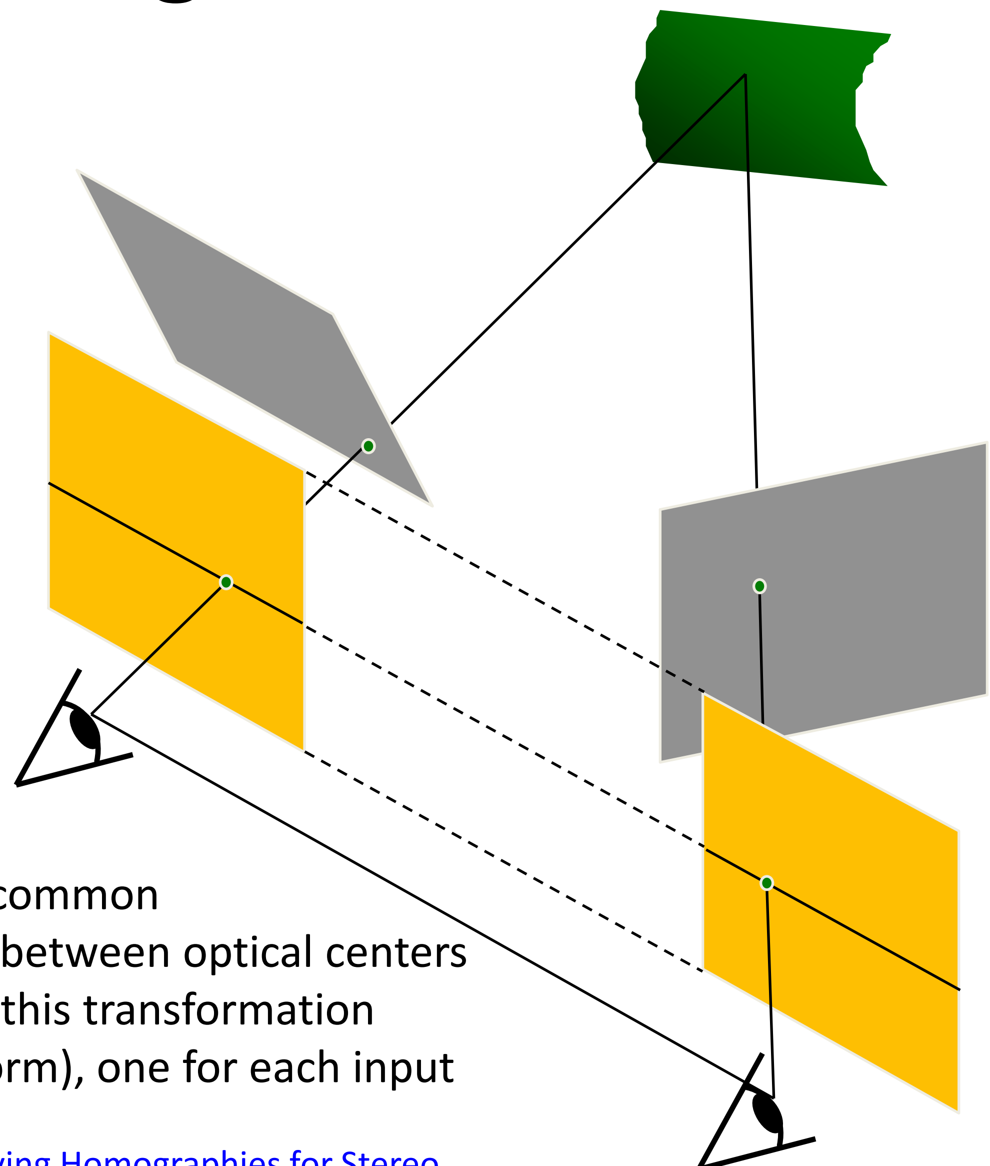
Rectified case



$$\mathbf{R} = \mathbf{I}_{3\times 3}$$
$$\mathbf{t} = [\begin{array}{ccc} 1 & 0 & 0 \end{array}]^T$$

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

Stereo image rectification

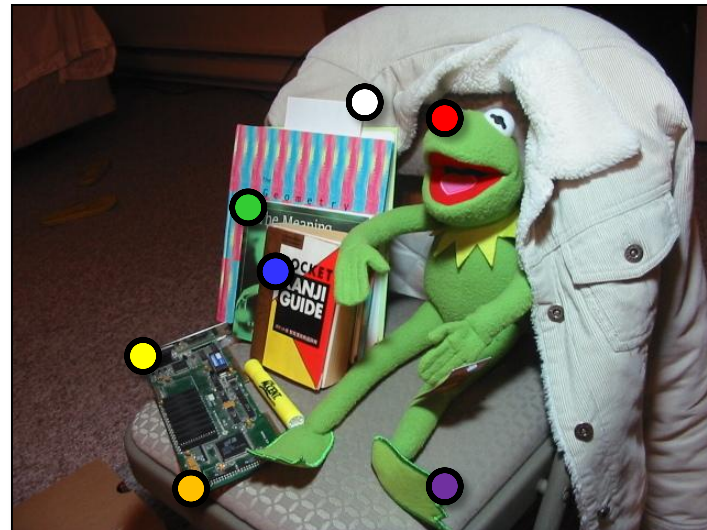


- reproject image planes onto a common
- plane parallel to the line between optical centers
- pixel motion is horizontal after this transformation
- two homographies (3x3 transform), one for each input image reprojection

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

Questions?

Estimating **F**



- If we don’t know $\mathbf{K}_1$, $\mathbf{K}_2$, $\mathbf{R}$, or $\mathbf{t}$, 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

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

- In reality, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$, least eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

8-point algorithm – Problem?

- **F** should have rank 2 – not a linear constraint
- 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.

8-point algorithm

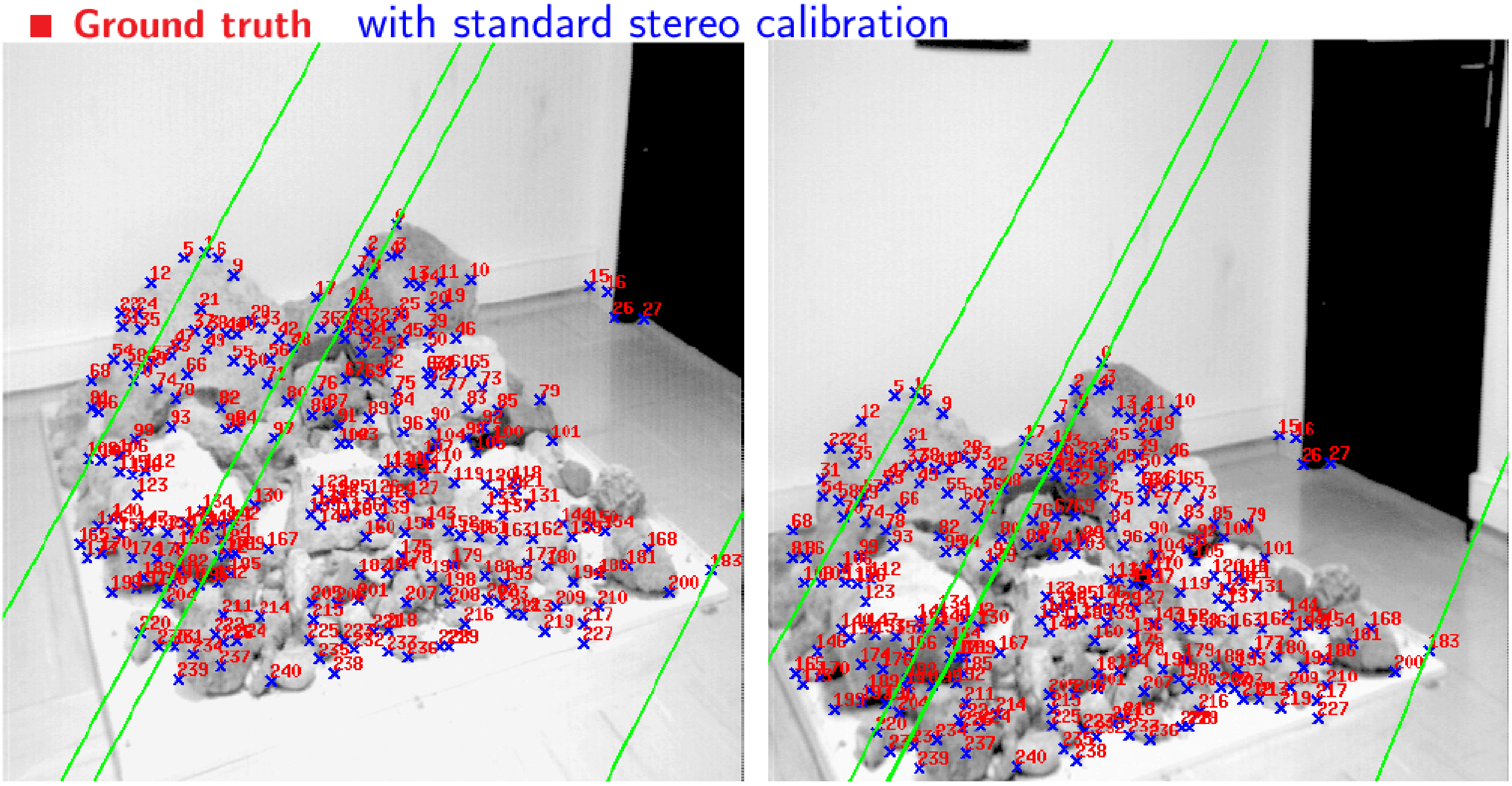
```
% Build the constraint matrix
   A = [x2(1,:)‘.*x1(1,:)'   x2(1,:)'.*x1(2,:)'  x2(1,:)' ...
        x2(2,:)'.*x1(1,:)'   x2(2,:)'.*x1(2,:)'  x2(2,:)' ...
        x1(1,:)'             x1(2,:)'            ones(npts,1) ];

   [U,D,V] = svd(A);

% Extract fundamental matrix from the column of V
% corresponding to the smallest singular value.
   F = reshape(V(:,9),3,3)';

% Enforce rank2 constraint
   [U,D,V] = svd(F);
   F = U*diag([D(1,1) D(2,2) 0])*V';
```

8-point algorithm

- Pros: it is linear, easy to implement and fast
- Cons: susceptible to noise

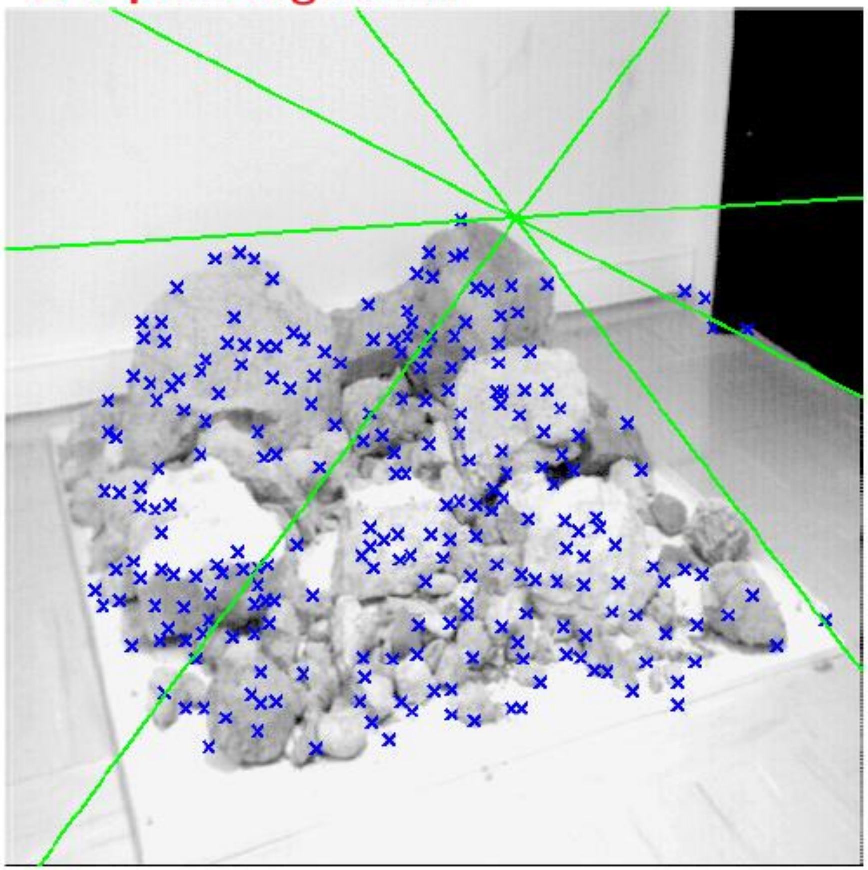
- In practice, we use the *normalized 8-point algorithm*
  - (See R. Hartley, “In Defense of the Eight-Point Algorithm”, PAMI ’97).

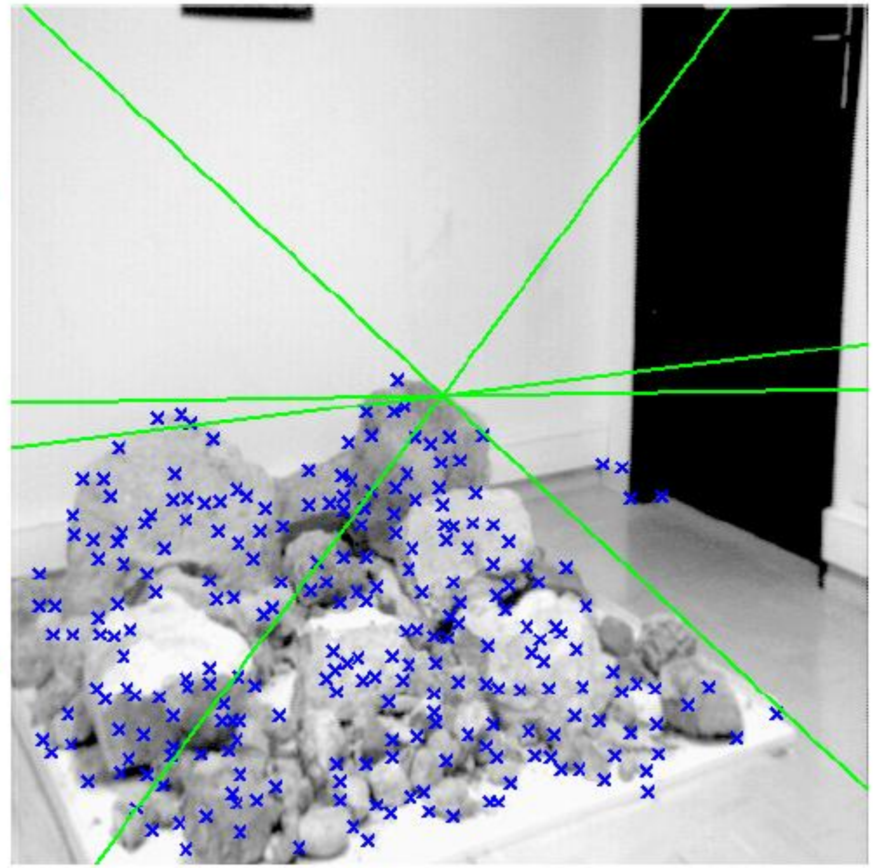
Results (ground truth)



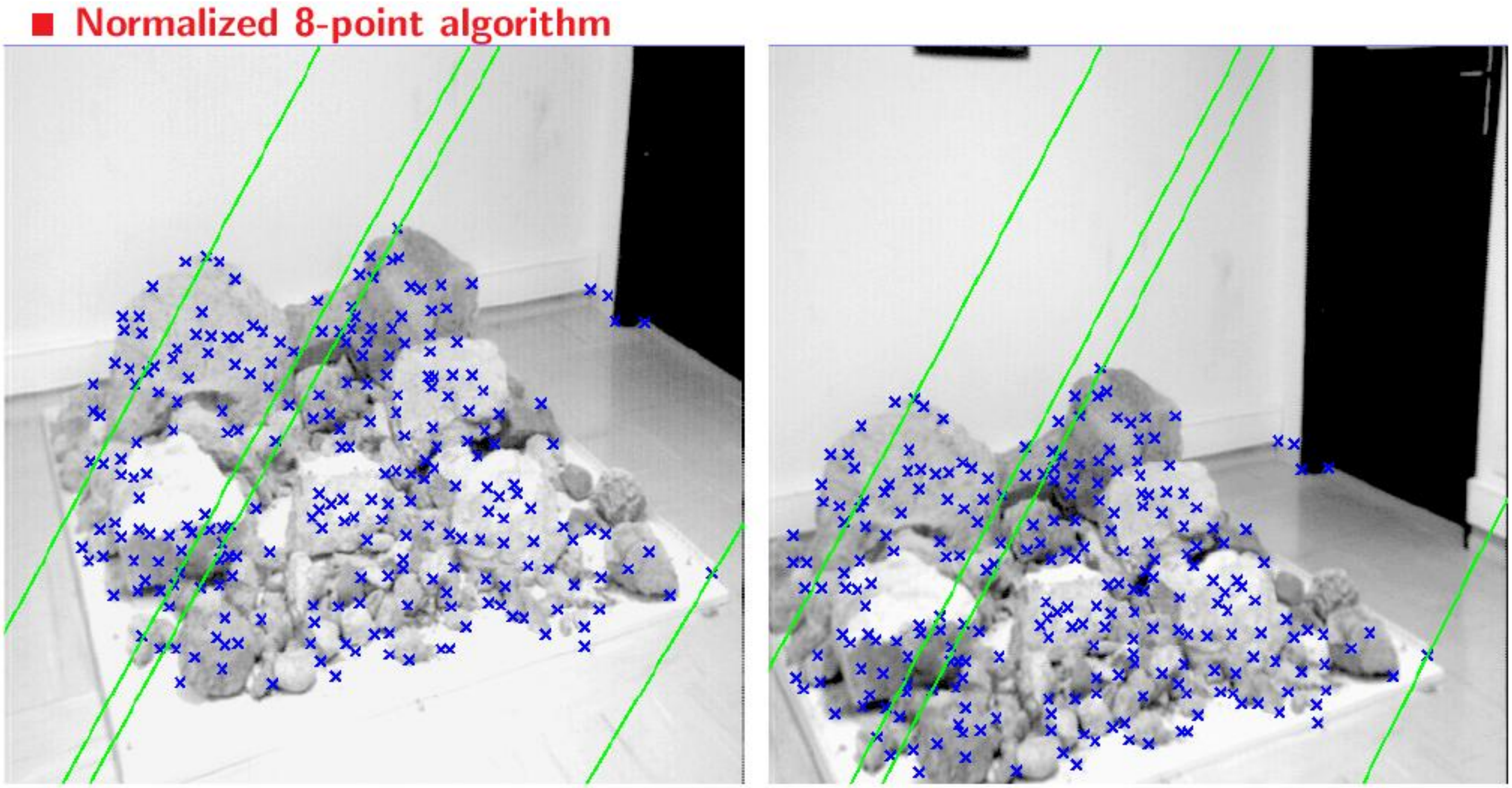
Results (8-point algorithm)

- 8-point algorithm





Results (normalized 8-point algorithm)



Estimating the **F**-matrix

- If we have more than eight points, we can solve a least-squares version of this problem

- What could go wrong?

- How can we fix this?

Estimating the **F**-matrix

- Extra bonus:
  - If we run RANSAC to find an **F**-matrix with the most inliers, we can throw out **inconsistent** matches (i.e., this is a “bad match filter”)
  - Might be useful later on…

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…