CS5670: Computer Vision

Two-view geometry

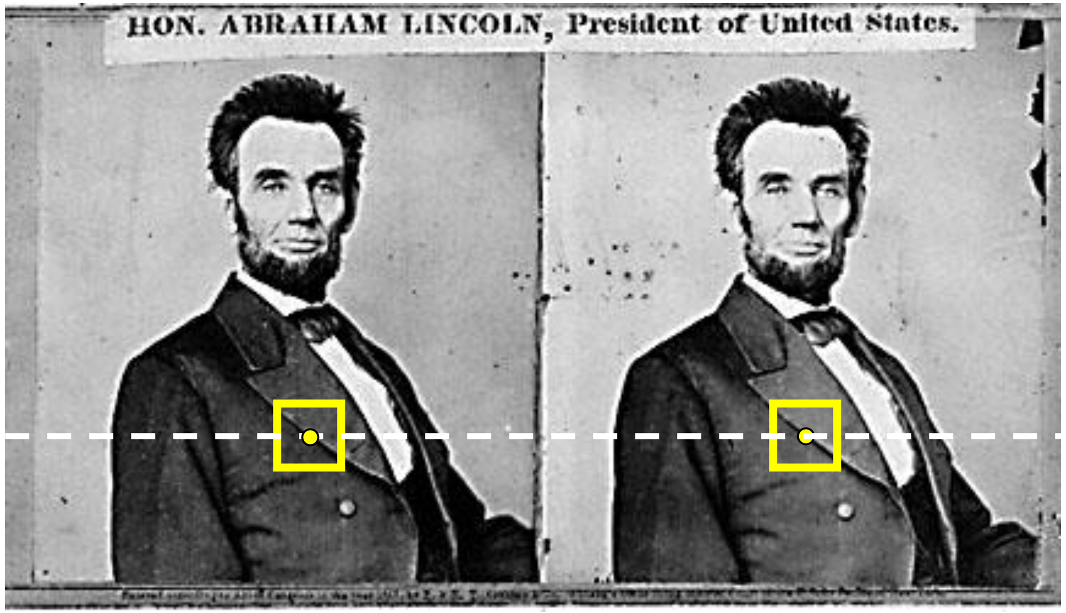
Reading

- Reading: Szeliski 2nd Edition, Ch. 11.3, 12.1

Fundamental matrix song

http://danielwedge.com/fmatrix/

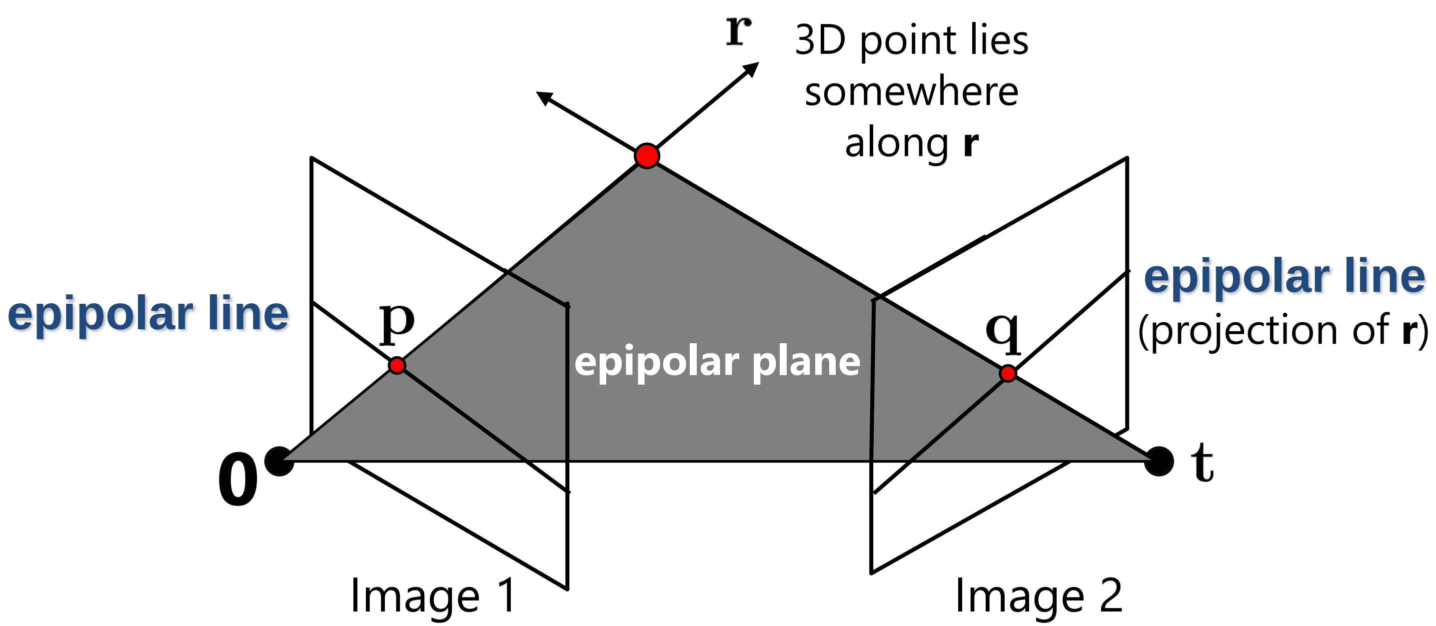
Back to stereo



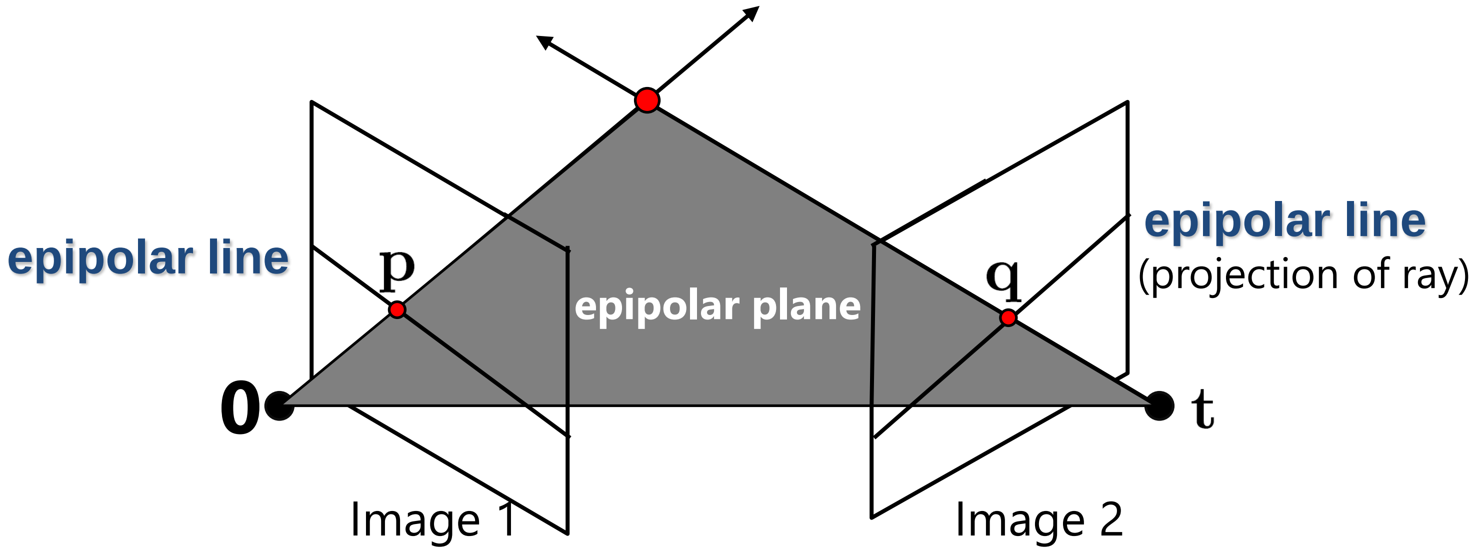
- Where do epipolar lines come from?

Two-view geometry

- Where do epipolar lines come from?

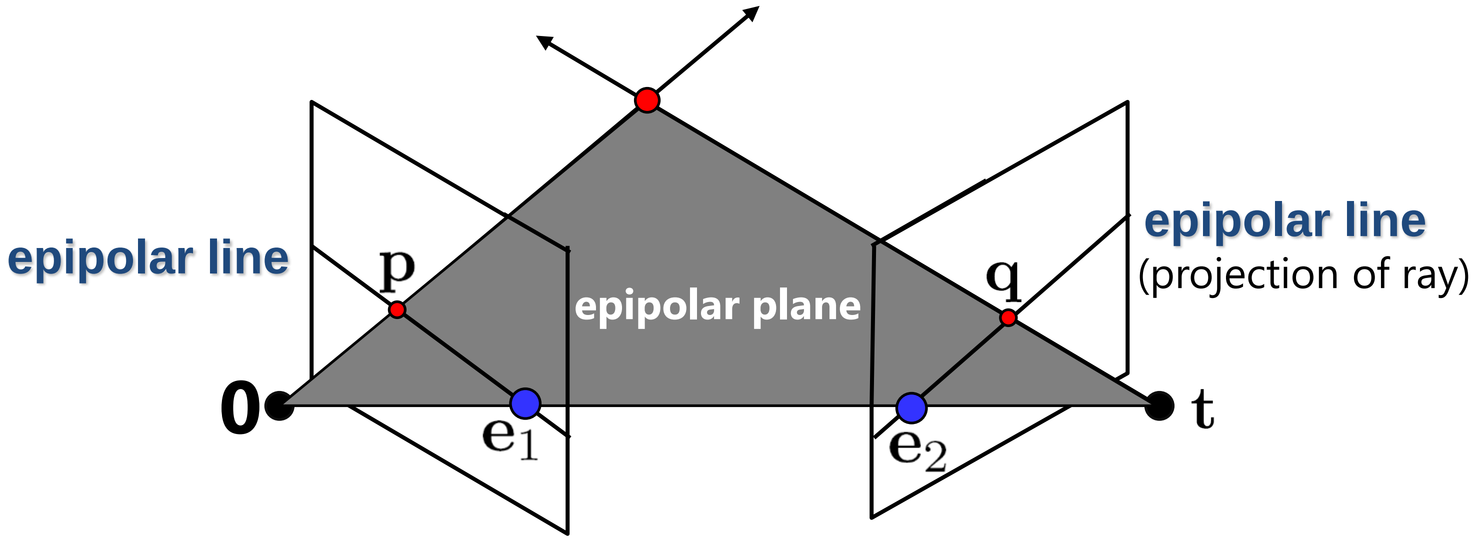


Fundamental matrix



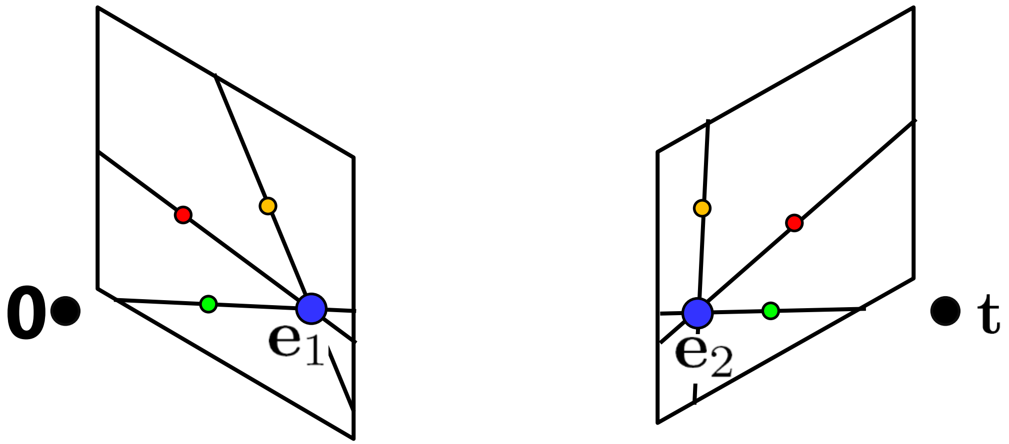
- This *epipolar geometry* of two views is described by a Very Special 3x3 matrix $\mathbf{F}$, called the *fundamental matrix*
- $\mathbf{F}$ maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point **p** $\mathbf{F}\mathbf{p}$
- *Epipolar constraint* on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

Fundamental matrix



- Two Special points: $\mathbf{e}_1$ and $\mathbf{e}_2$ (the *epipoles*): projection of one camera into the other
- All of the epipolar lines in an image pass through the epipole
- Epipoles may or may not be inside the image

Epipoles

Properties of the Fundamental Matrix

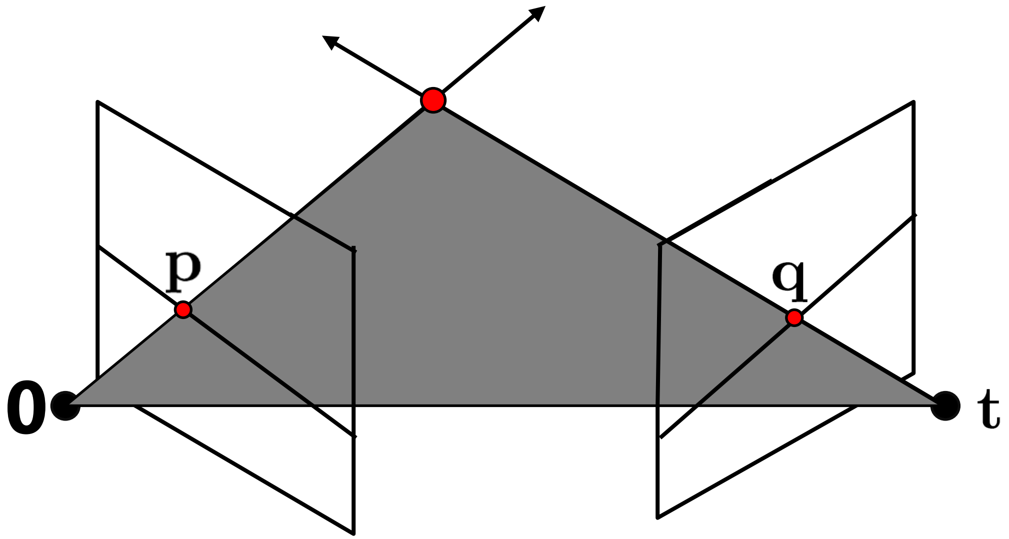
- $\mathbf{F}\mathbf{p}$ is the epipolar line associated with $\mathbf{p}$
- $\mathbf{F}^T\mathbf{q}$ is the epipolar line associated with $\mathbf{q}$
- $\mathbf{F}\mathbf{e}_1 = \mathbf{0}$ and $\mathbf{F}^T\mathbf{e}_2 = \mathbf{0}$
- $\mathbf{F}$ is rank 2
- How many parameters does $\mathbf{F}$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

Example

Demo

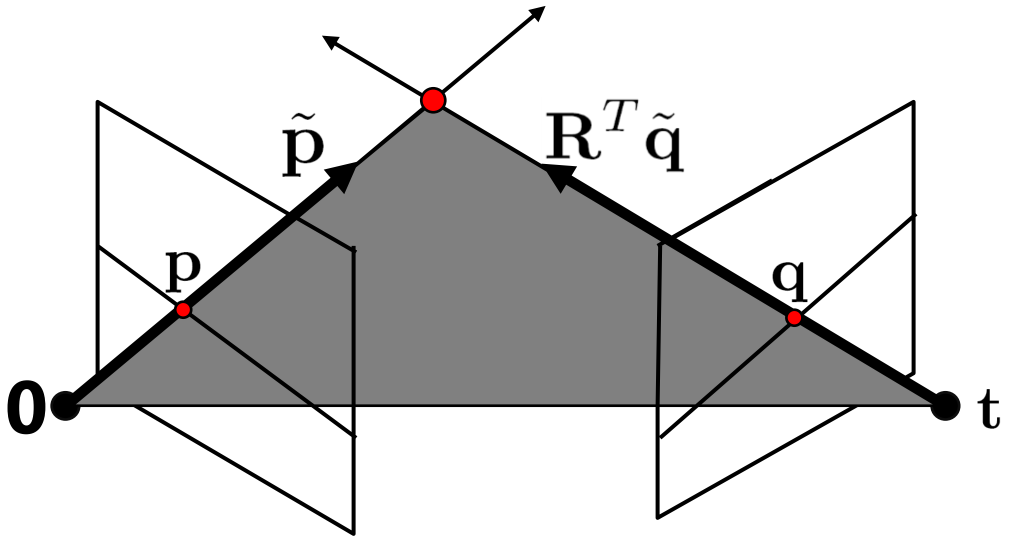
https://www.cs.cornell.edu/courses/cs5670/2022sp/demos/FundamentalMatrix/?demo=demo1

Fundamental matrix



- Why does **F** exist?
- Let’s derive it…

Fundamental matrix – calibrated case



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

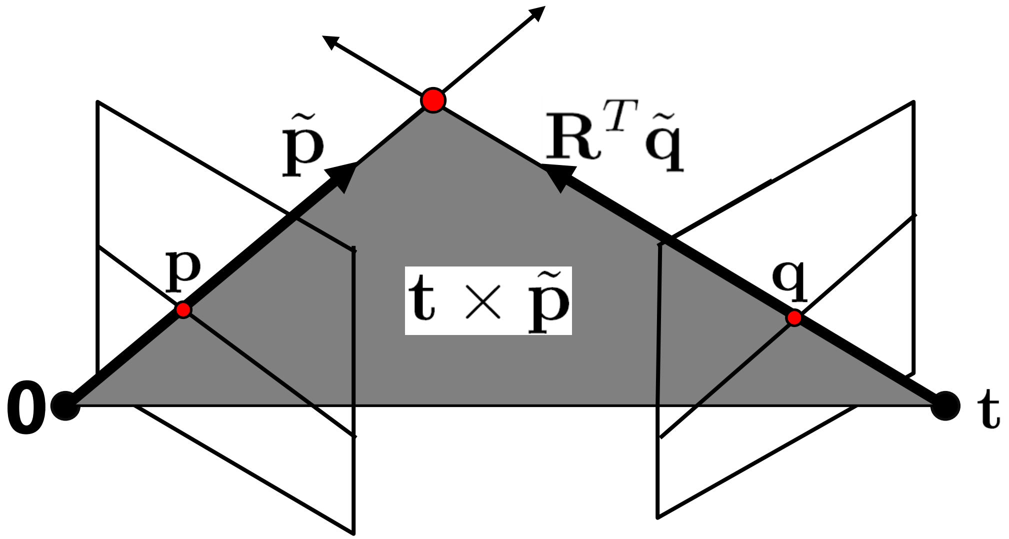
$\mathbf{K}_2$: intrinsics of camera 2

$\mathbf{R}$: 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

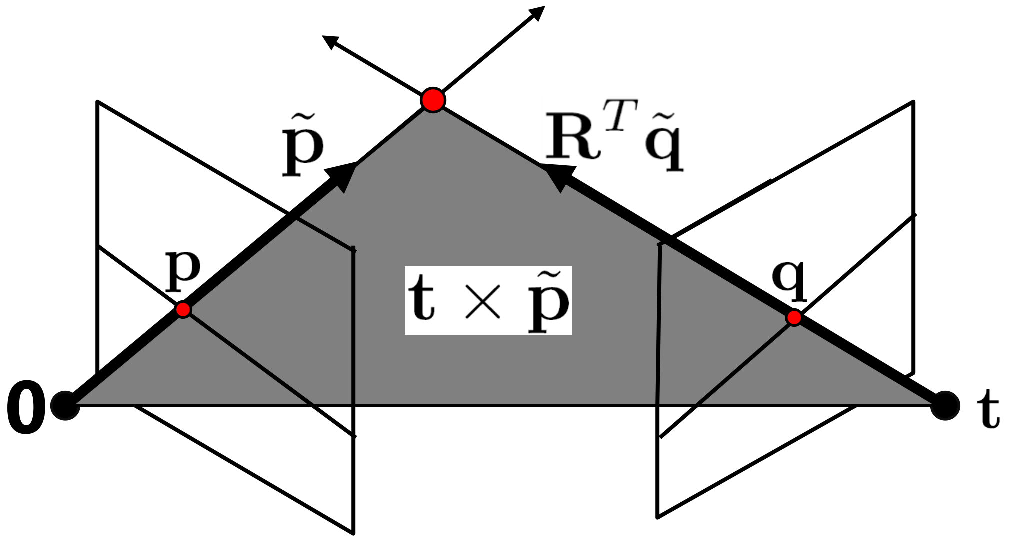
Fundamental matrix – calibrated case



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

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

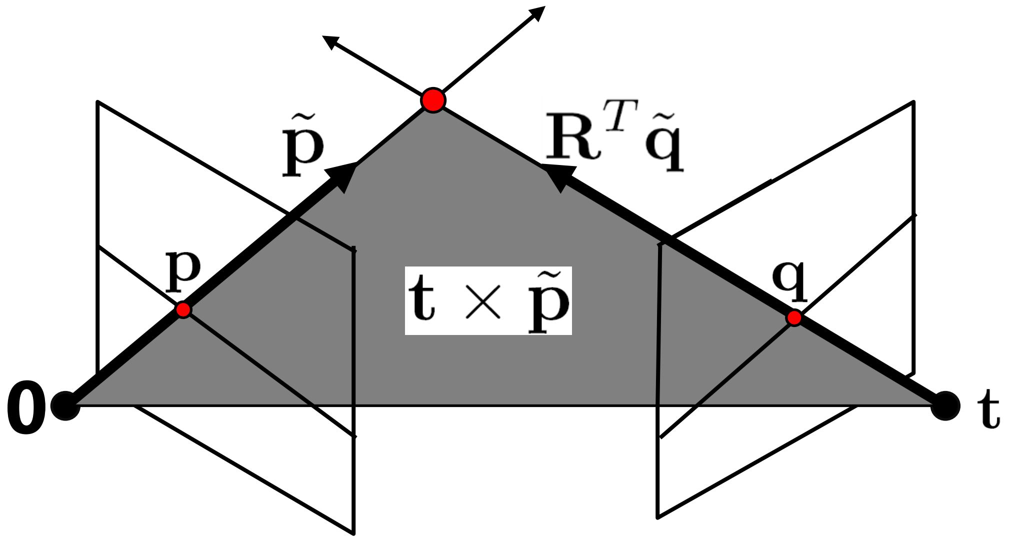
Fundamental matrix – calibrated case



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

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

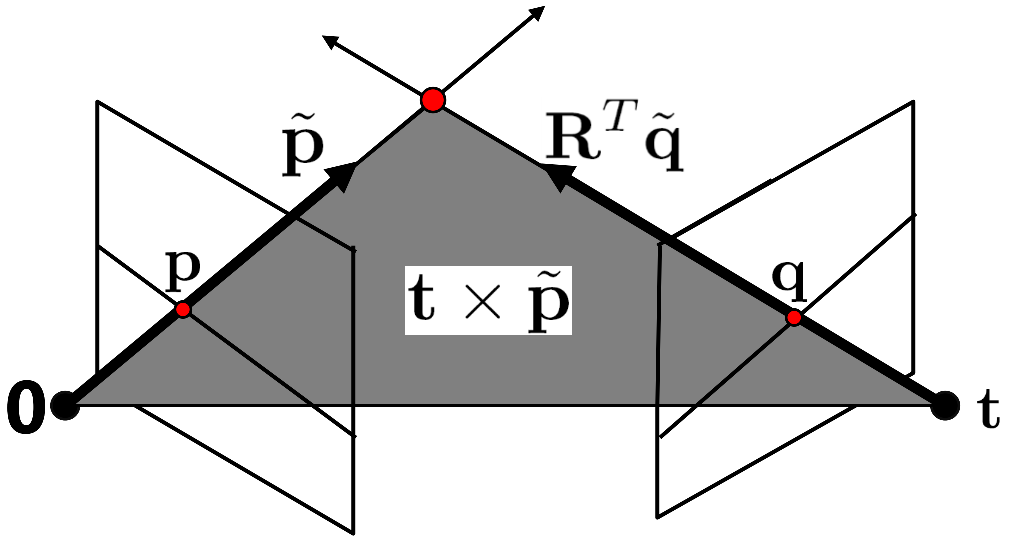
Fundamental matrix – calibrated case



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

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

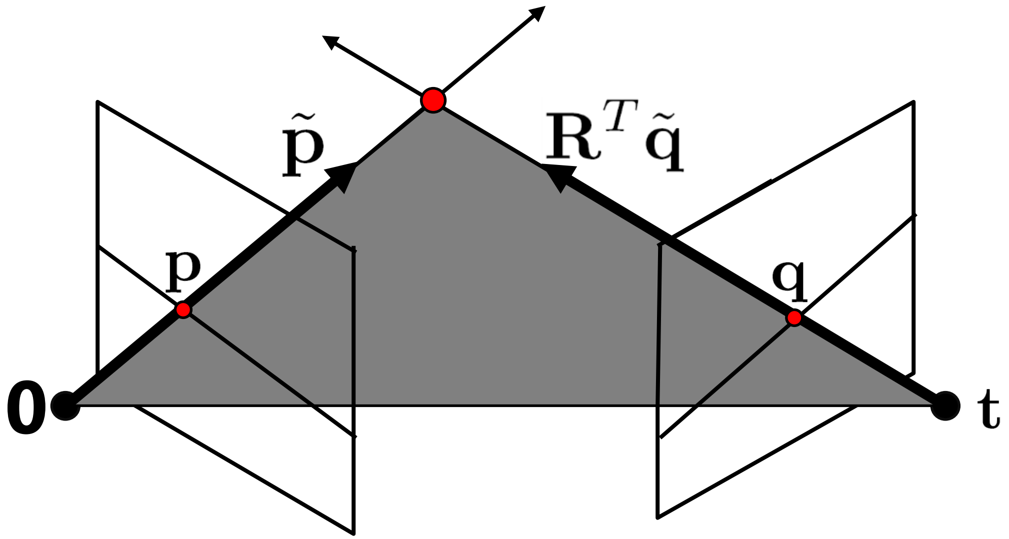
Fundamental matrix – calibrated case



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

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

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

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

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

$\mathbf{E}$ ← the Essential matrix

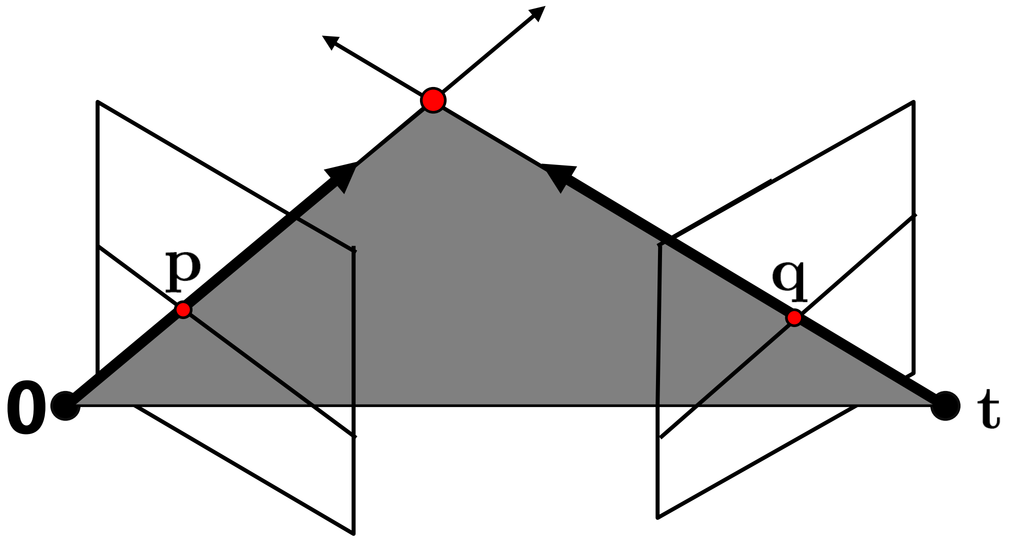
Cross-product as linear operator

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 – uncalibrated case



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

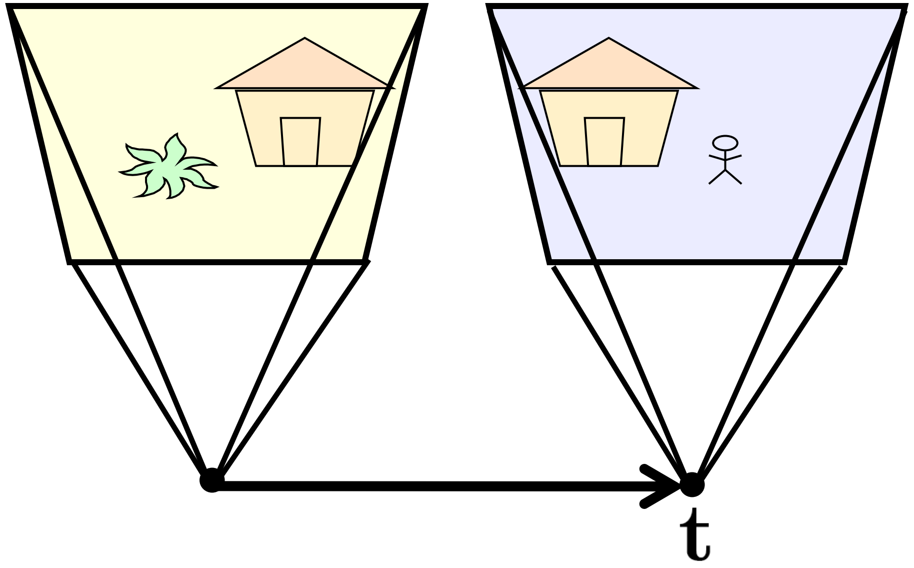
$\mathbf{K}_2$: intrinsics of camera 2

$\mathbf{R}$: 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

Rectified case



$$\mathbf{R} = \mathbf{I}_{3\times 3}$$
$$\mathbf{t} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$$

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

Working out the math

$$\begin{bmatrix} 0 & b & 0 \\ 0 & 2 & -1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ y \end{bmatrix} \begin{bmatrix} a \\ b \\ 1 \end{bmatrix} = 0$$

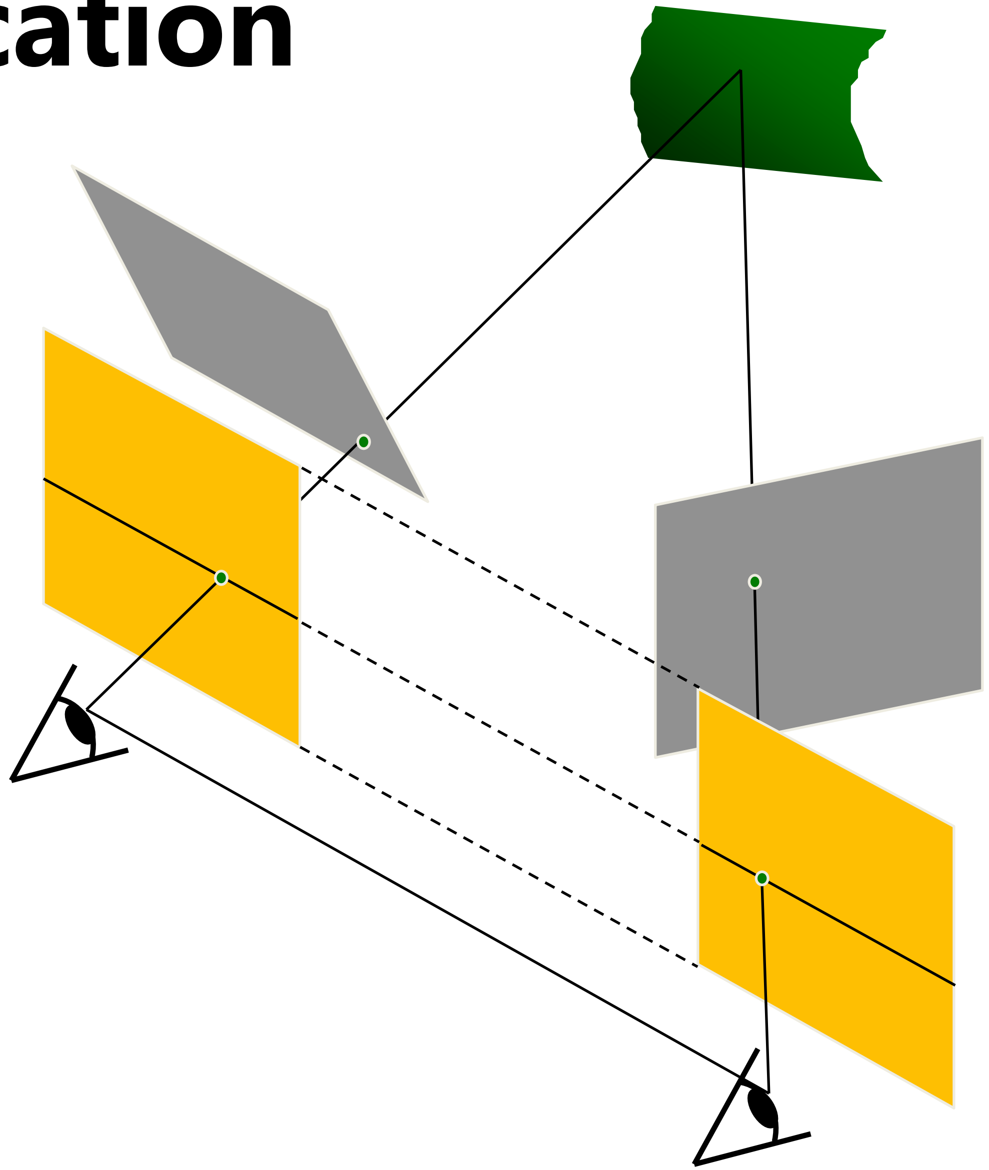
$$E$$

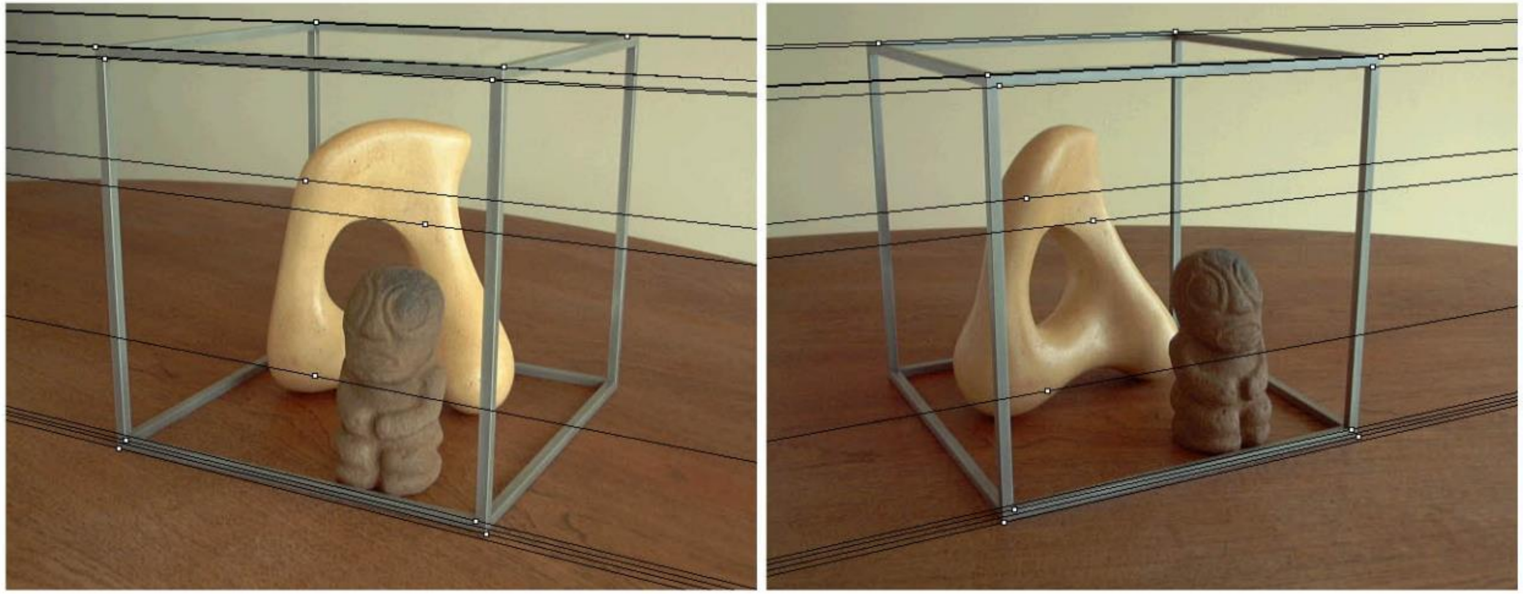
$$-b + y = 0$$

$$b = y$$

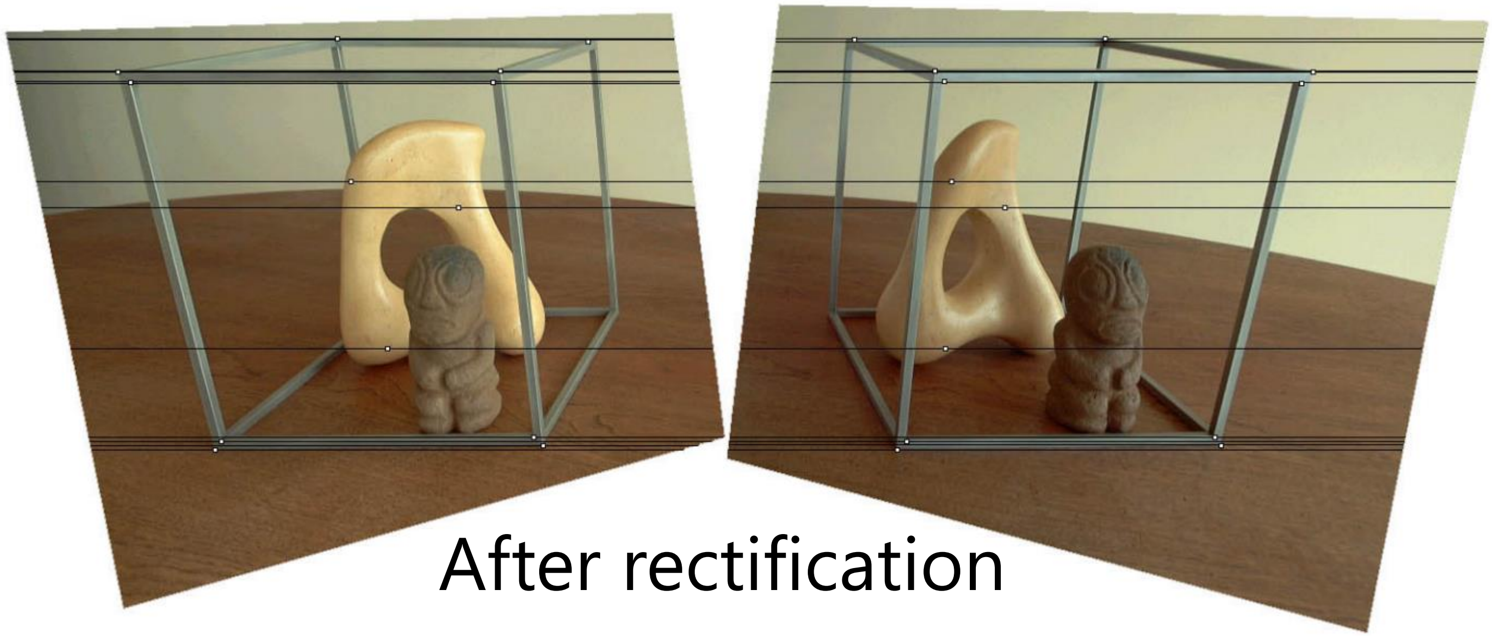
Stereo image rectification

- reproject image planes onto a common plane
  - plane parallel to the line between optical centers
- pixel motion is horizontal after this transformation
- two homographies, one for each input image reprojection
  - C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. CVPR 1999.



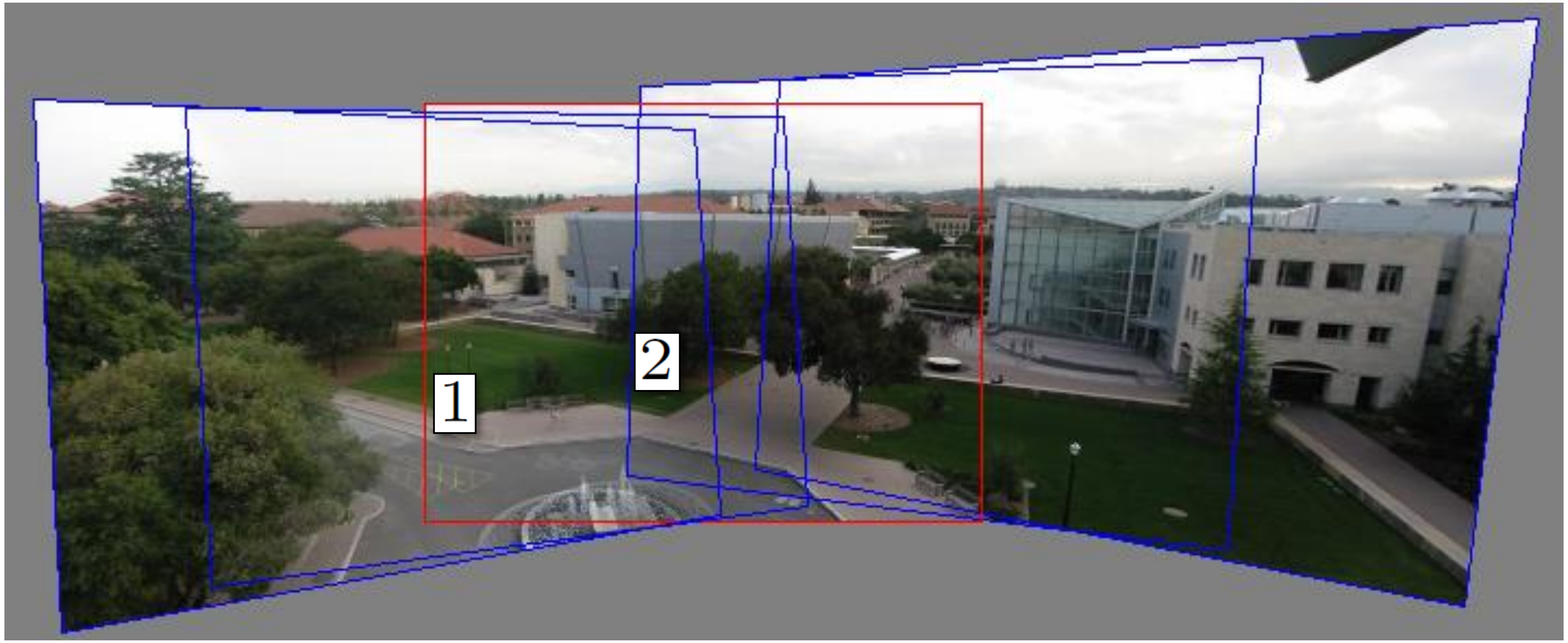


Original stereo pair



After rectification

Relationship between F matrix and homography?



Images taken from the same center of projection? Use a homography!

Questions?

Estimating F

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

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

- Like with homographies, instead of solving $\mathbf{Af} = 0$, we seek unit length $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$: least eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$.

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

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix},$$

Let

$$\Sigma' = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

then $\mathbf{F}' = \mathbf{U}\Sigma'\mathbf{V}^{\mathrm{T}}$ is the solution (closest rank-2 matrix to **F**)

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

Problem with 8-point algorithm

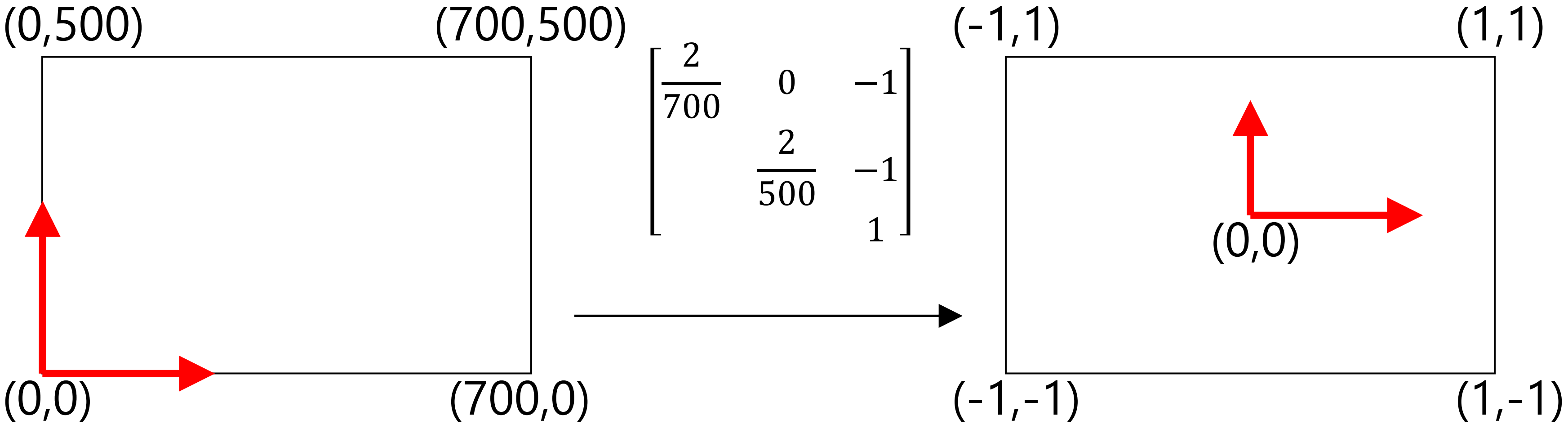
$$\begin{array}{l}
\begin{bmatrix}
u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\
u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1
\end{bmatrix}
\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \\
\begin{array}{ccccccccc}
\sim 10000 & \sim 10000 & \sim 100 & \sim 10000 & \sim 10000 & \sim 100 & \sim 100 & \sim 100 & 1
\end{array}
\end{array}$$

Orders of magnitude difference
between column of data matrix
$\rightarrow$ least-squares yields poor results

Normalized 8-point algorithm

normalized least squares yields good results

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



$$\begin{bmatrix} \frac{2}{700} & 0 & -1 \\ & \frac{2}{500} & -1 \\ & & 1 \end{bmatrix}$$

Normalized 8-point algorithm

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

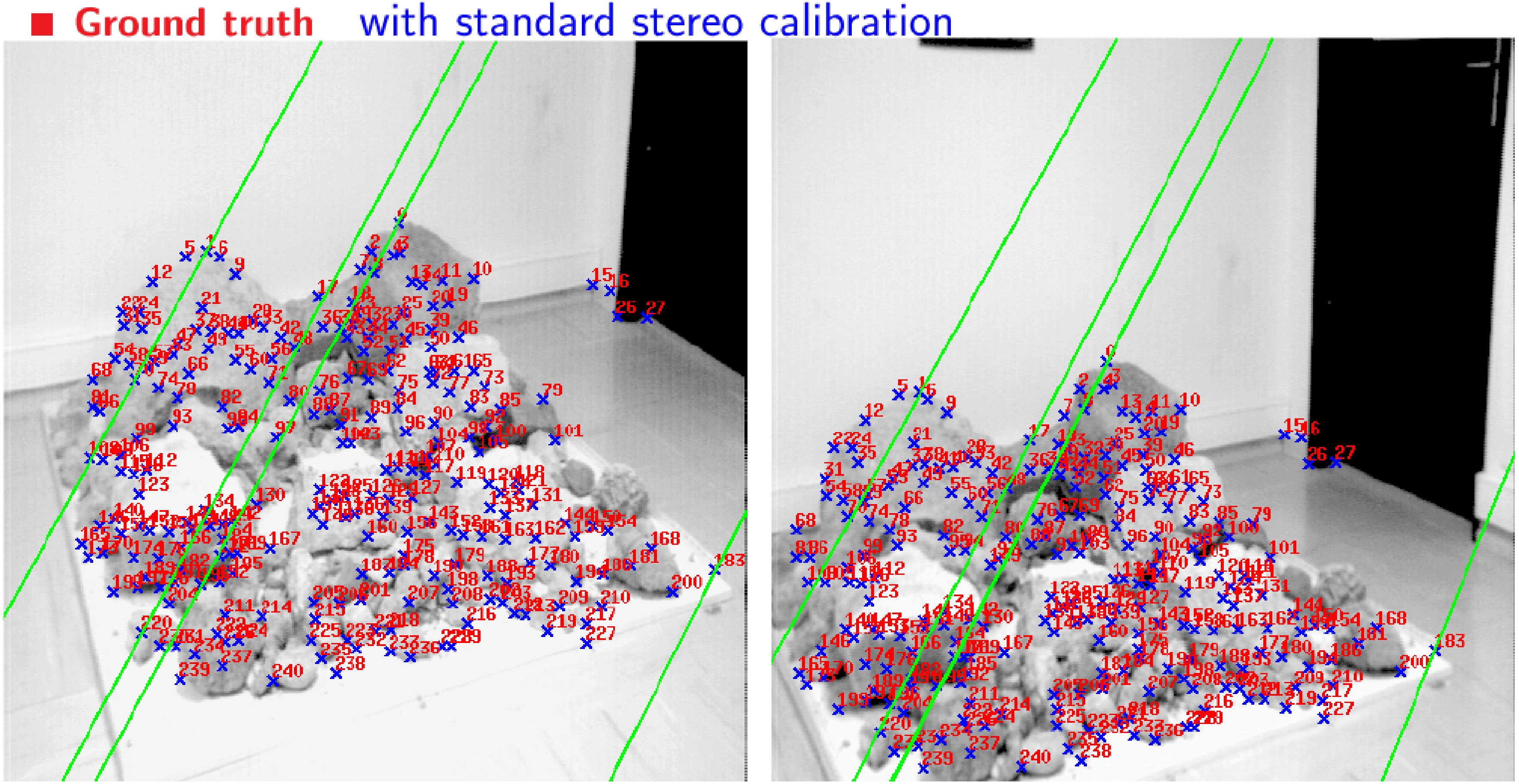
$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$$

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

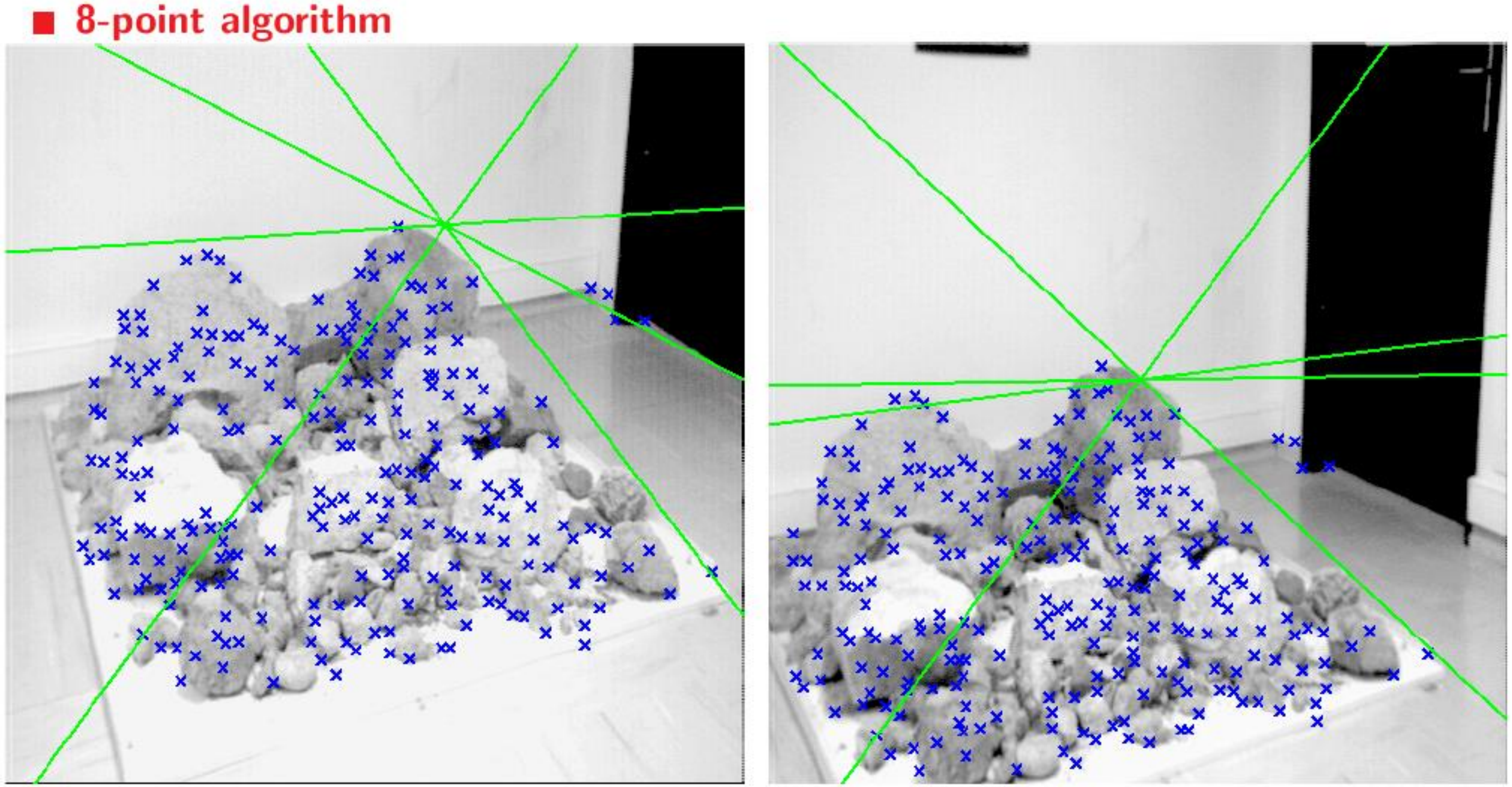
Normalized 8-point algorithm

```
[x1, T1] = normalise2dpts(x1);
[x2, T2] = normalise2dpts(x2);

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);

F = reshape(V(:,9),3,3)';

[U,D,V] = svd(F);
F = U*diag([D(1,1) D(2,2) 0])*V';


% Denormalise
F = T2'*F*T1;
```

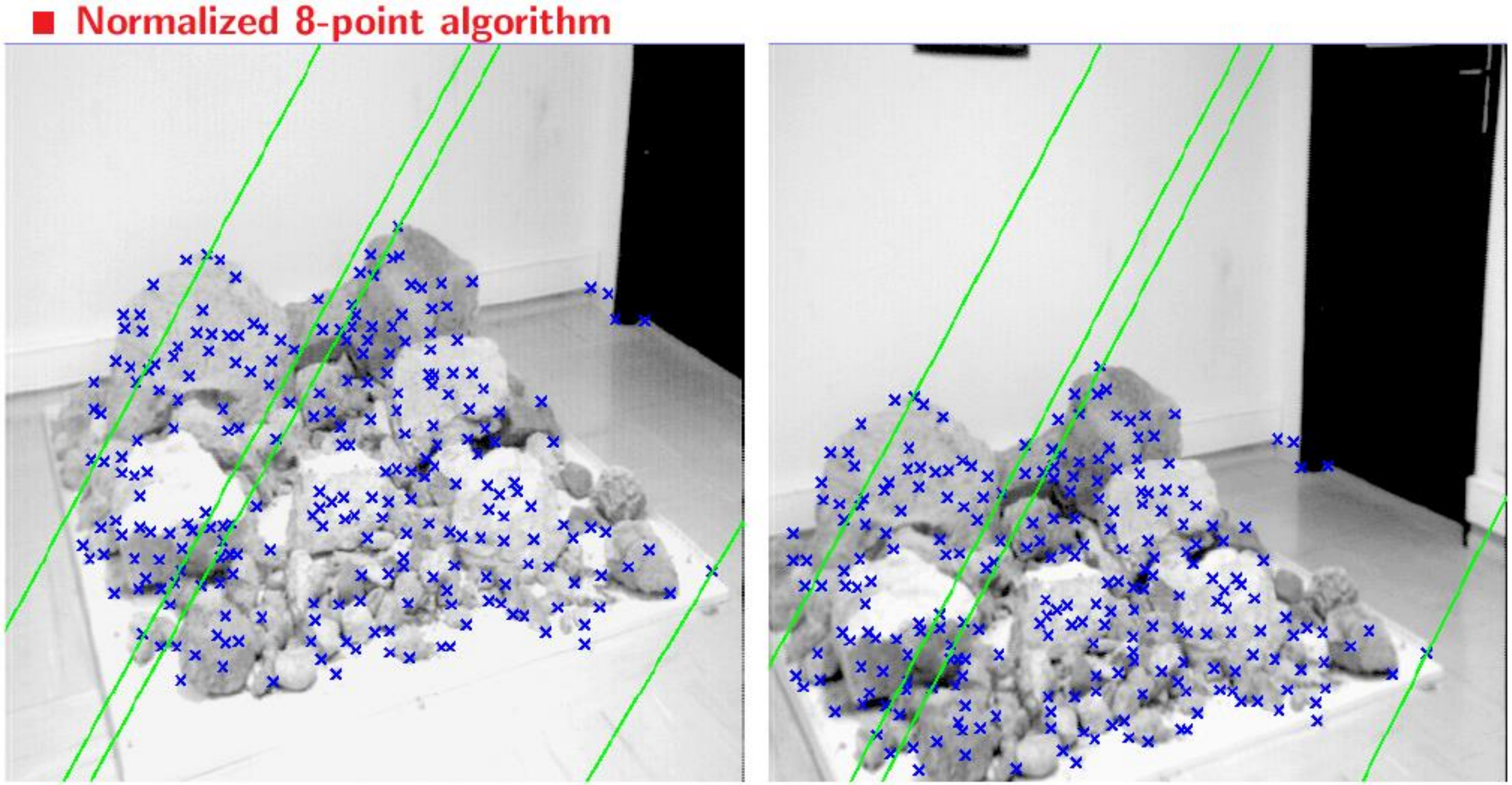
Results (ground truth)



Results (ground truth)



Results (normalized 8-point algorithm)



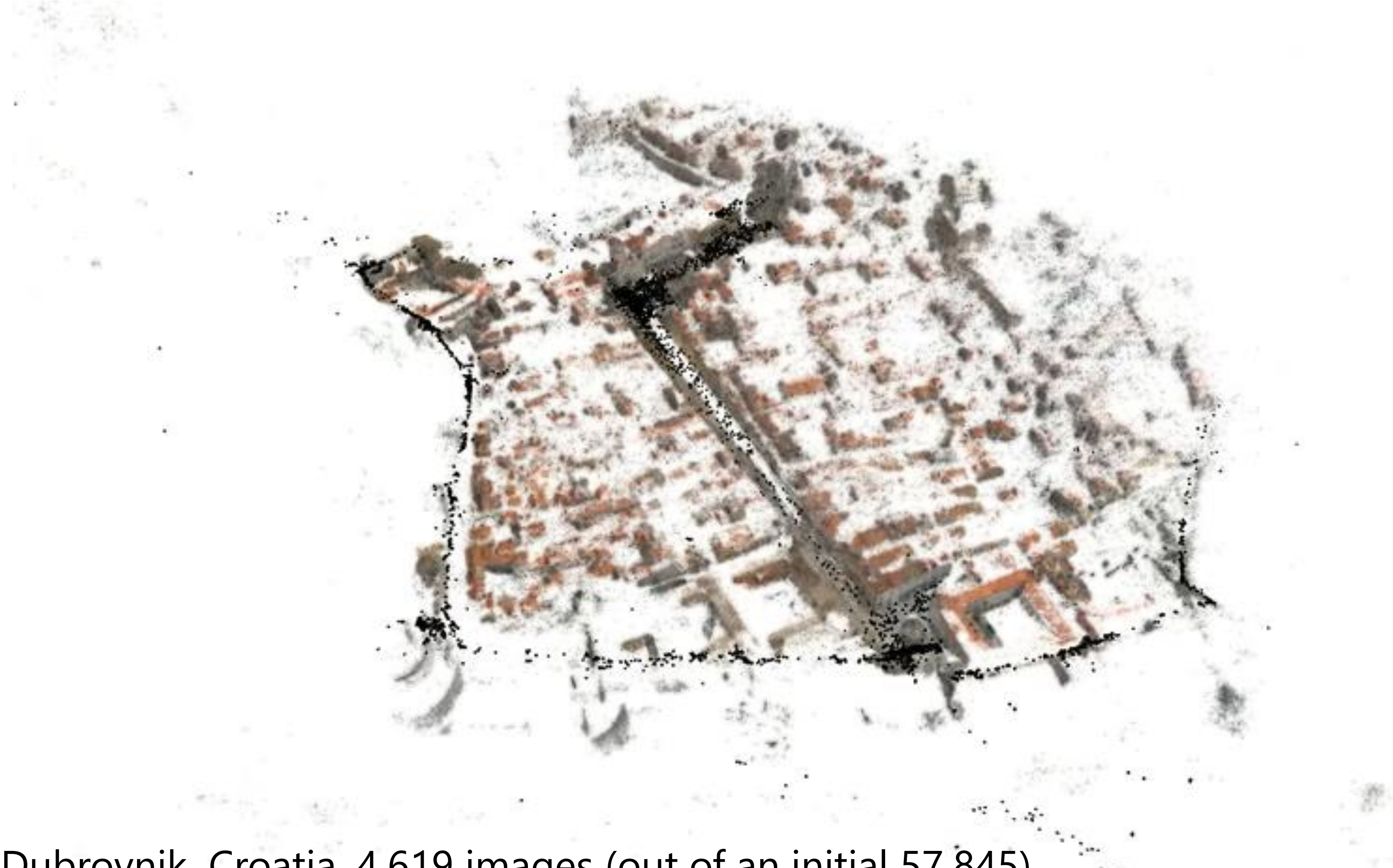
What about more than two views?

- 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…

Large-scale structure from motion



Dubrovnik, Croatia. 4,619 images (out of an initial 57,845).
Total reconstruction time: 23 hours
Number of cores: 352

Questions?