CS5670 : Computer Vision

Noah Snavely

Lecture 14: Two-view geometry

Announcements

- Midterm graded
  - Total: 95 points
  - Mean: 79.4
  - Median: 82
- Project 3 released

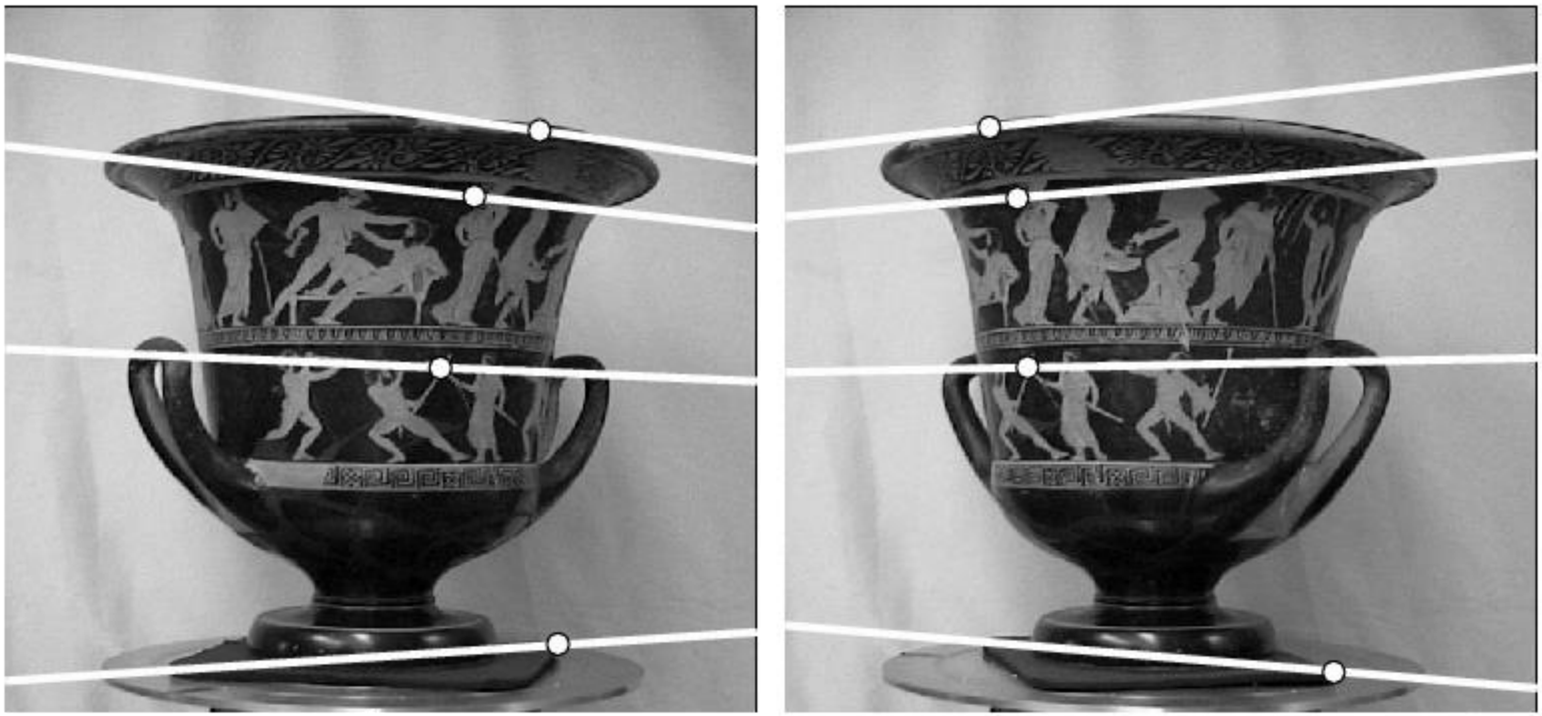
Announcements

- Reading: Szeliski, Ch. 7.2

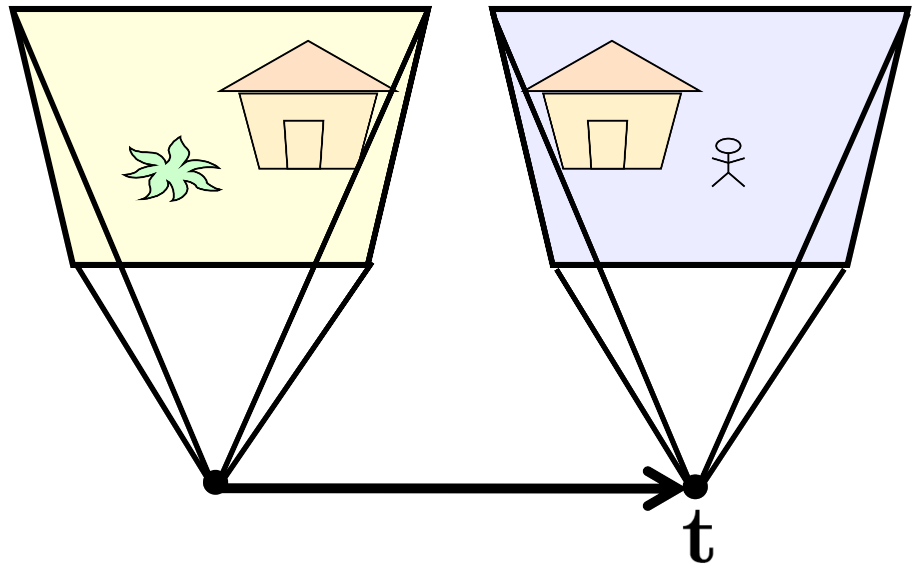
Fundamental matrix song

https://www.youtube.com/watch?v=DgGV3l82NTk

Epipolar geometry

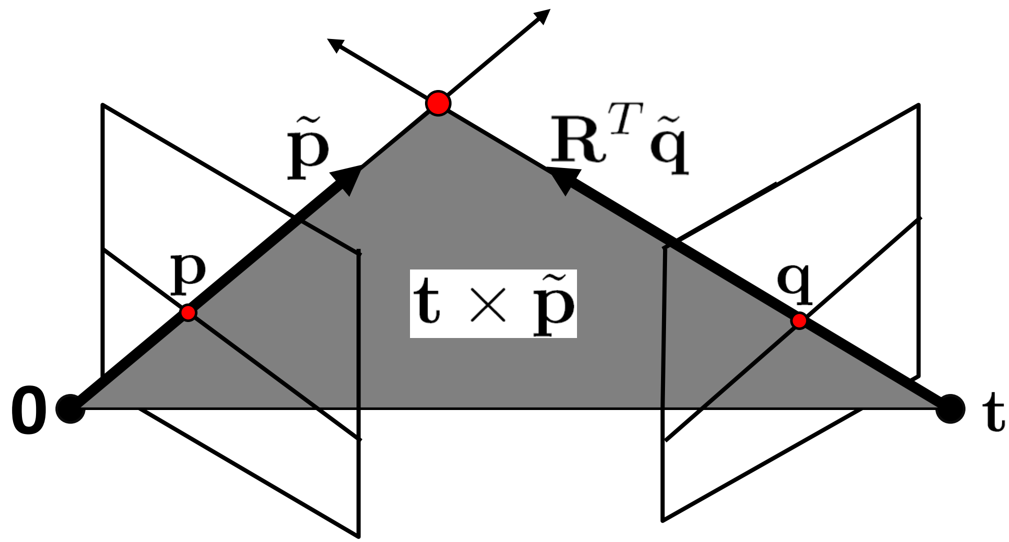


Rectified case



- Images have the same orientation, **t** parallel to image planes
- Where are the epipoles?

Fundamental matrix – calibrated case

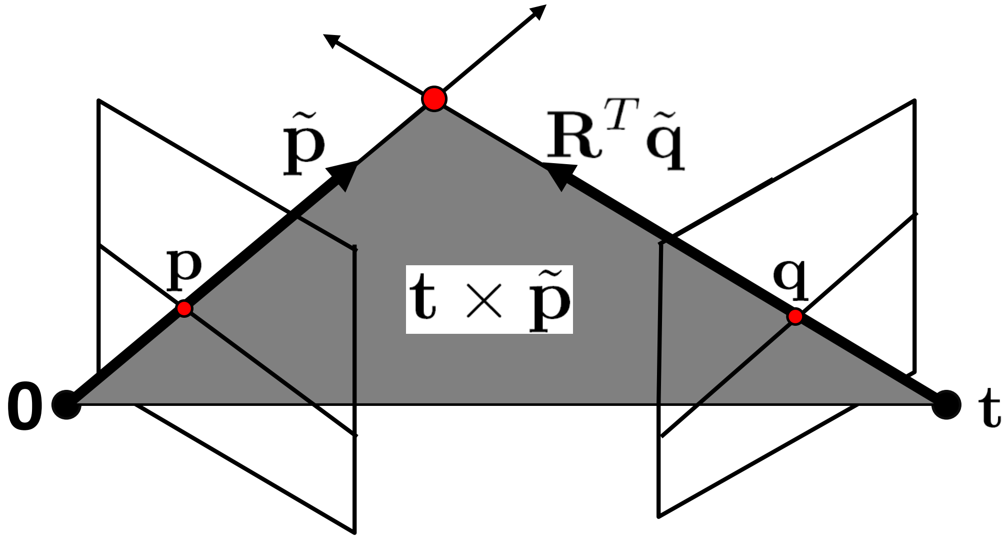


$$\tilde{\mathbf{q}}^T \underbrace{\mathbf{R}\,[\mathbf{t}]_\times}_{\mathbf{E}}\, \tilde{\mathbf{p}} = 0$$

the Essential matrix

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

Fundamental matrix – uncalibrated case



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

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

$\mathbf{F}$ ← the Fundamental matrix

Properties of the Fundamental Matrix

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

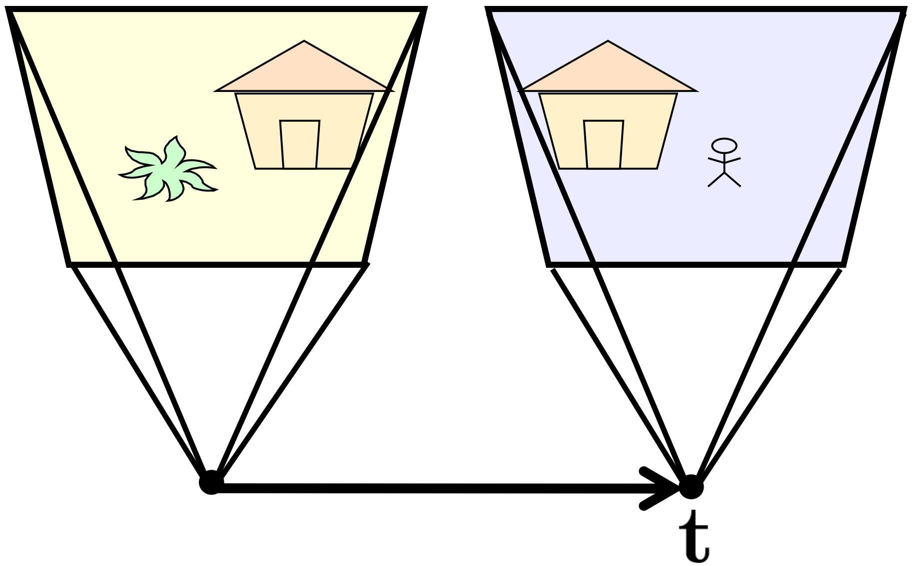
- $\mathbf{F}^T\mathbf{q}$ is the epipolar line associated with $\mathbf{q}$

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

- $\mathbf{F}$ is rank 2

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

Rectified case

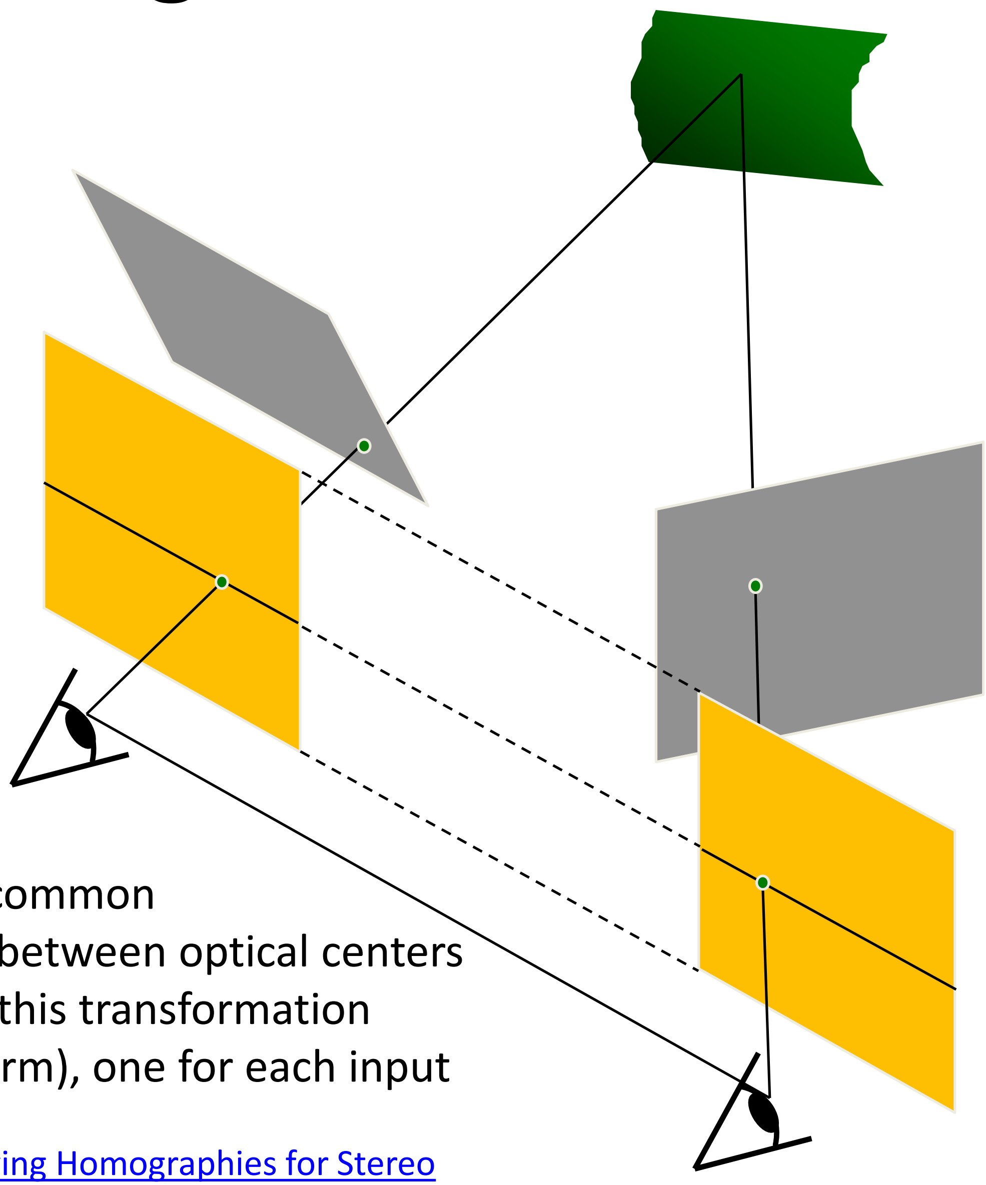


$$\mathbf{R} = \mathbf{I}_{3\times 3}$$

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

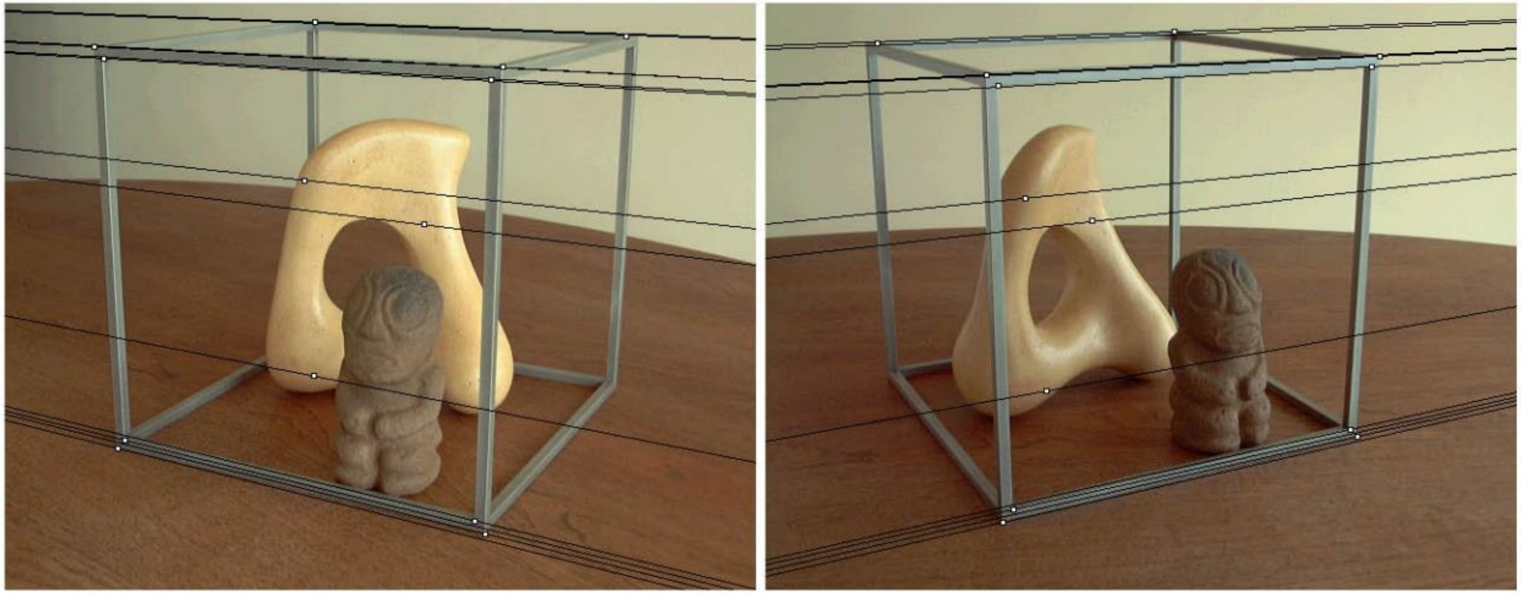
$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

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

Questions?

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

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

- 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
- 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

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

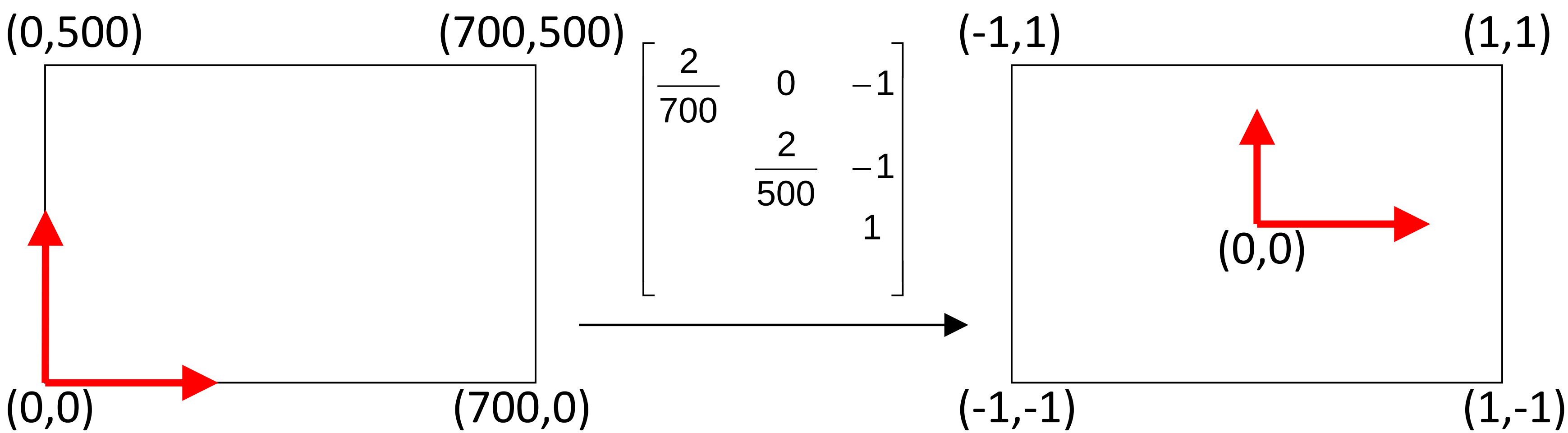
!

Orders of magnitude difference
between column of data matrix
$\rightarrow$ 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 $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{T}\mathbf{x}_\mathbf{i}$, $\hat{\mathbf{x}}_\mathbf{i}' = \mathbf{T}\mathbf{x}_\mathbf{i}'$
2. Call 8-point on $\hat{\mathbf{x}}_\mathbf{i}, \hat{\mathbf{x}}_\mathbf{i}'$ to obtain $\hat{\mathbf{F}}$
3. $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

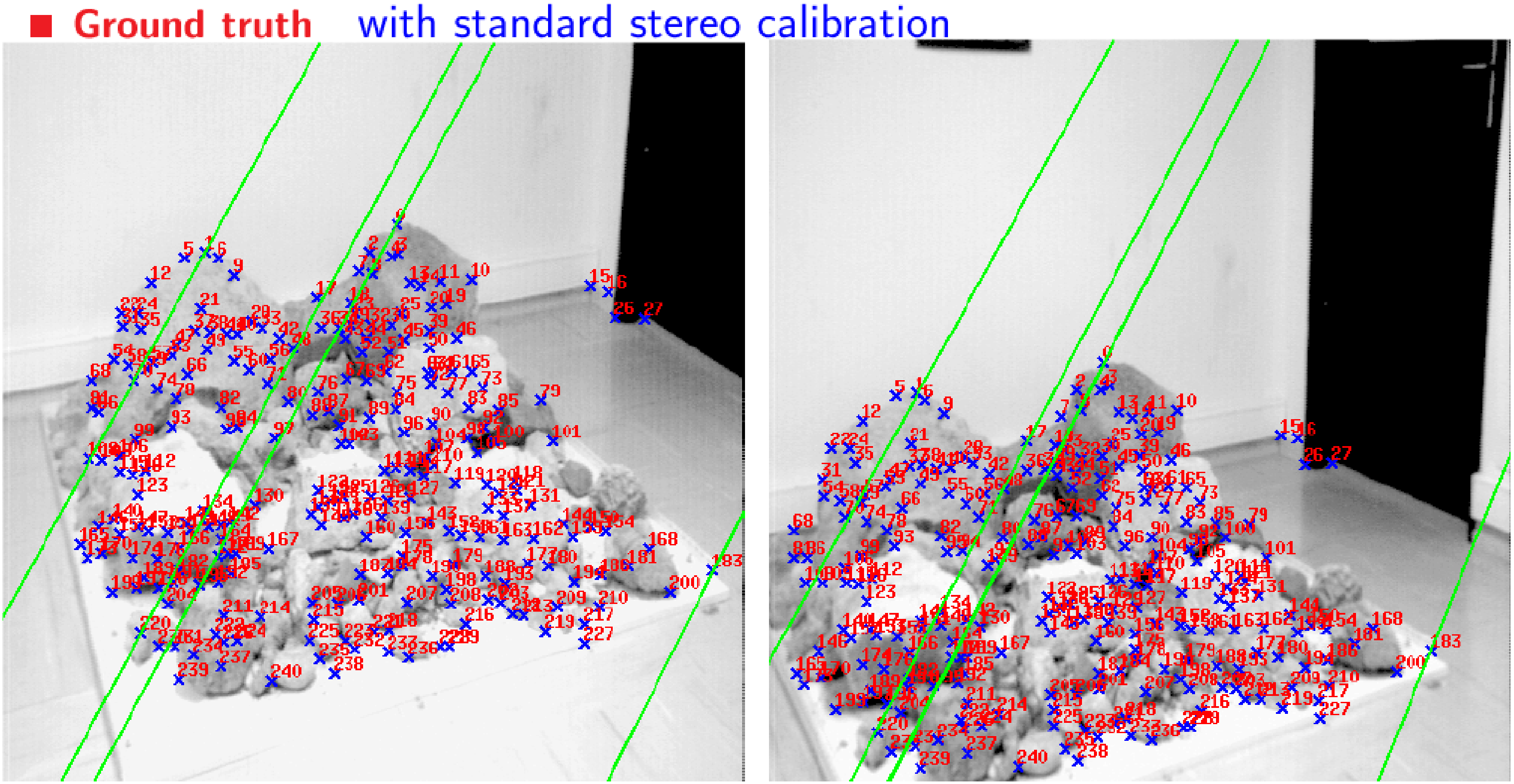
$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$$

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

Normalized 8-point algorithm

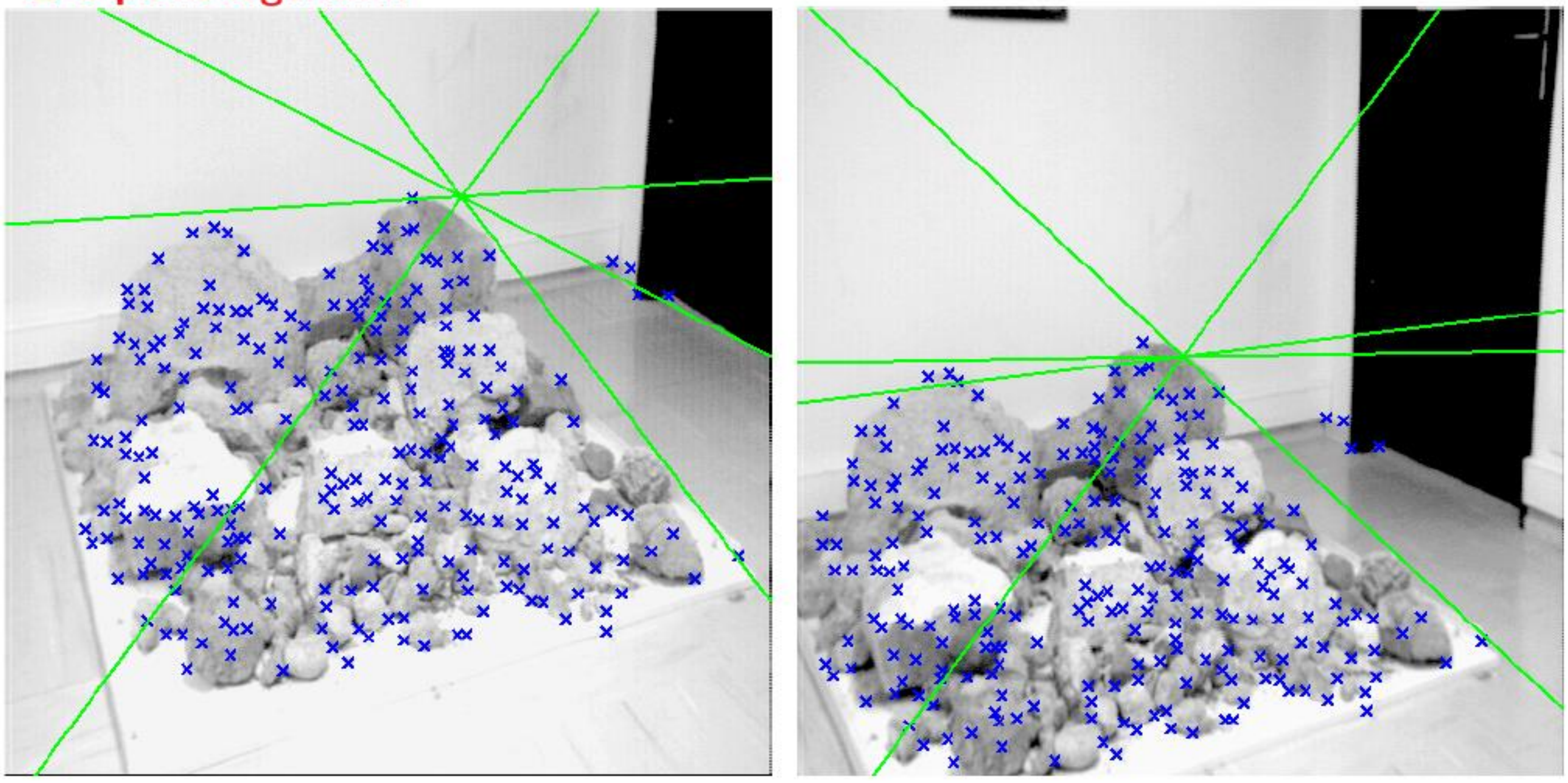
```
[x1, T1] = normalise2dpts(x1);
[x2, T2] = normalise2dpts(x2);

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)



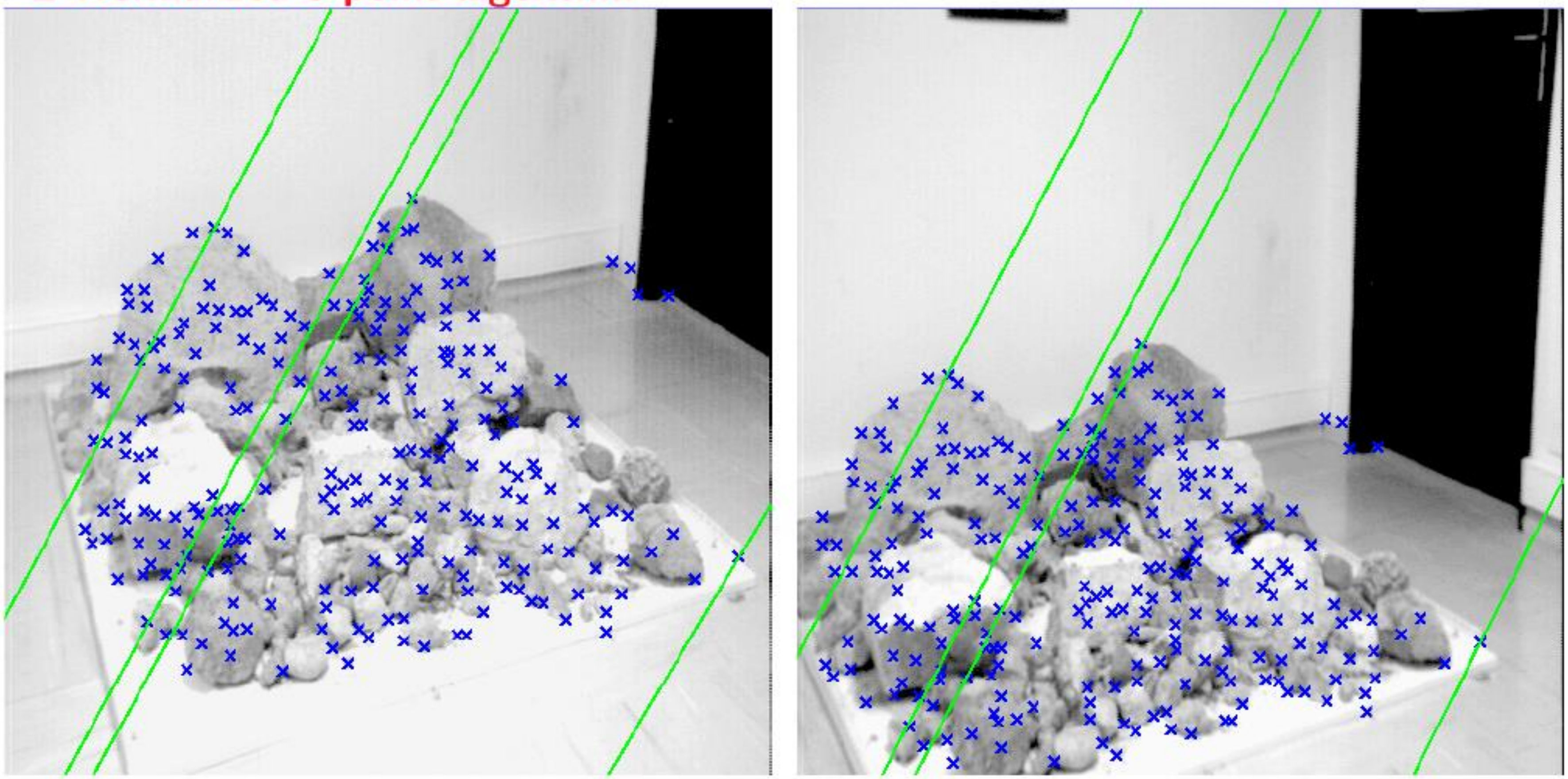
Results (8-point algorithm)

- 8-point algorithm



Results (normalized 8-point algorithm)

- 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