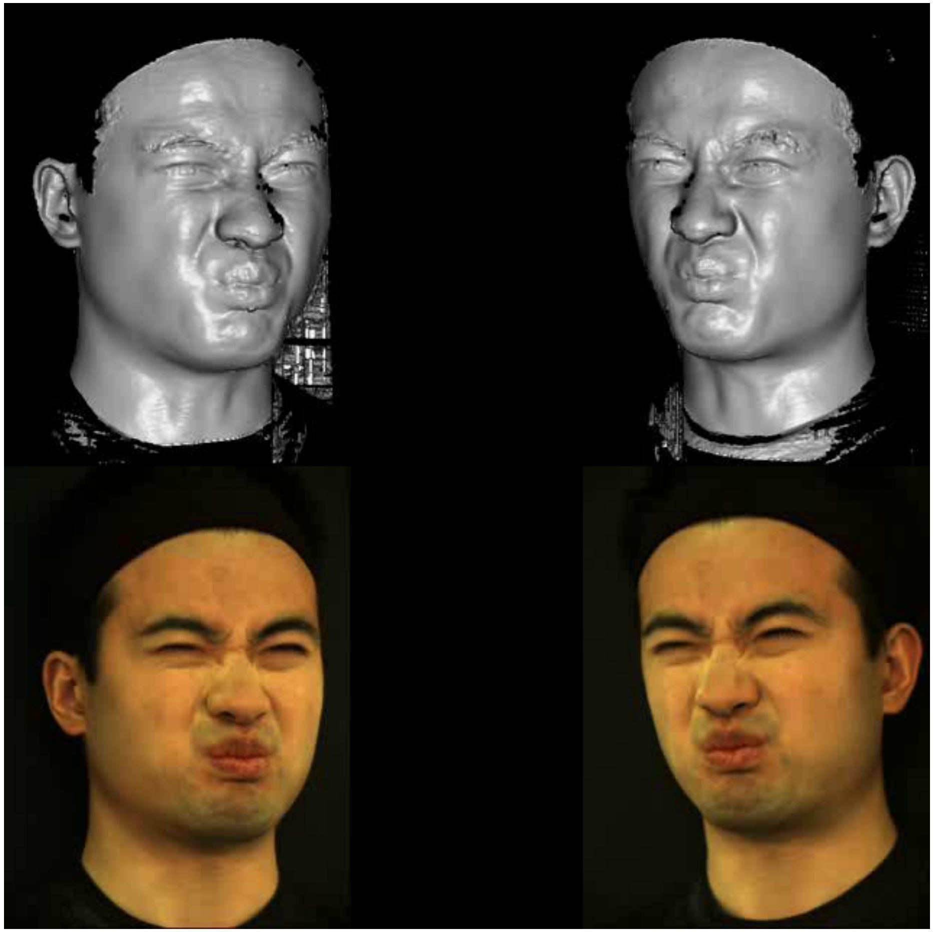
Last lecture

- Passive Stereo
- Spacetime Stereo



Today

- Structure from Motion:

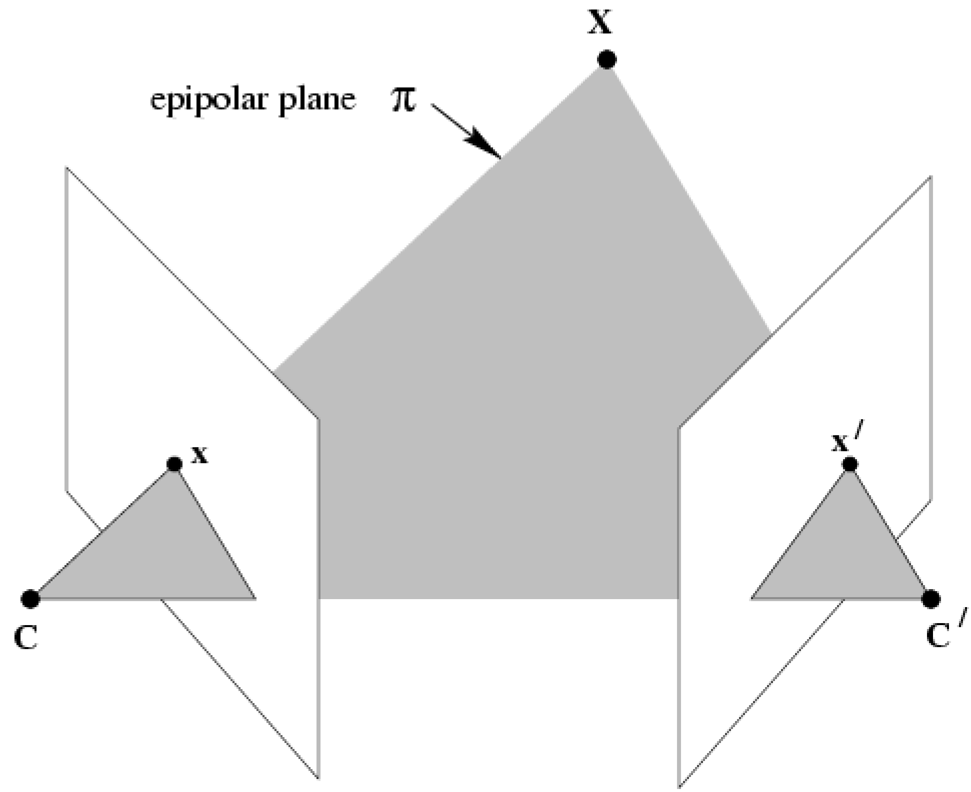
  Given pixel correspondences,

  how to compute 3D structure and camera motion?

Epipolar geometry & fundamental matrix

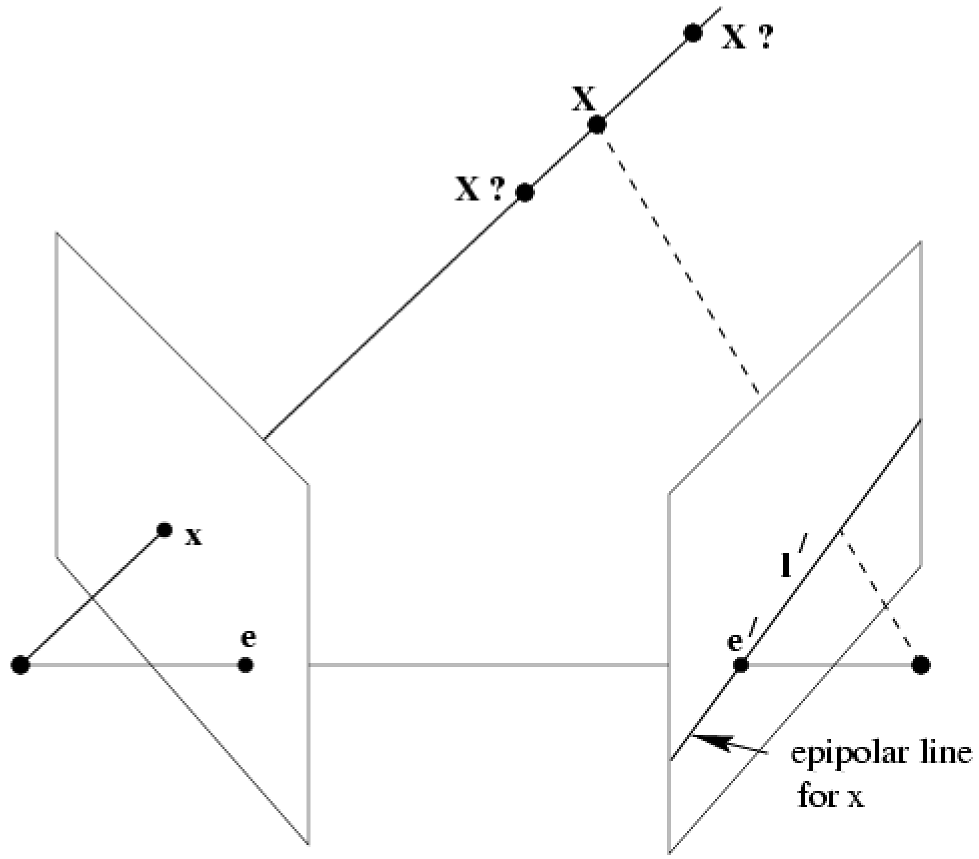
The epipolar geometry

epipolar geometry demo



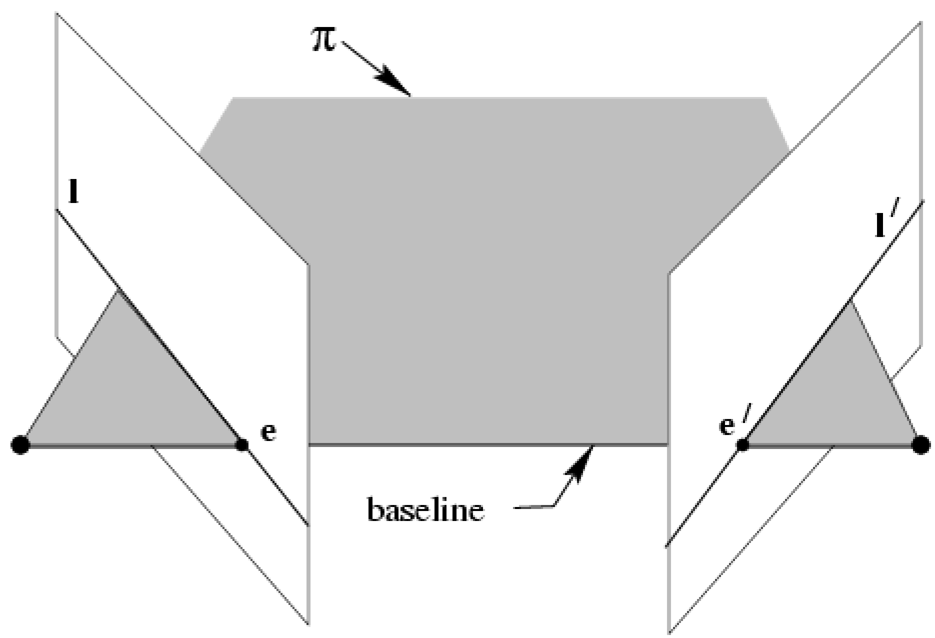
C, C', x, x' and X are coplanar

The epipolar geometry



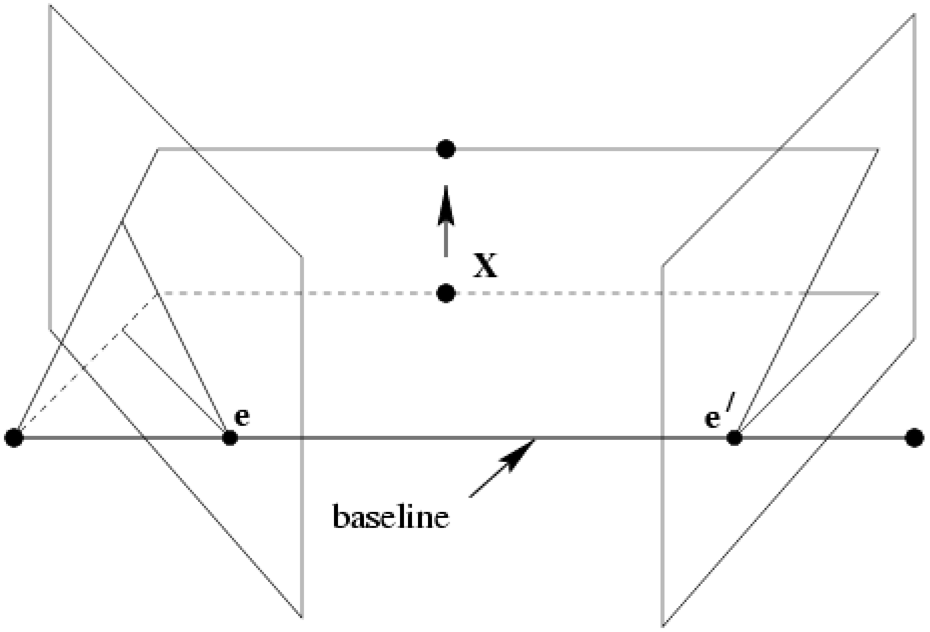
What if only C, C', x are known?

The epipolar geometry



All points on π project on *l* and *l'*

The epipolar geometry



Family of planes π and lines l and l' intersect at e and e'

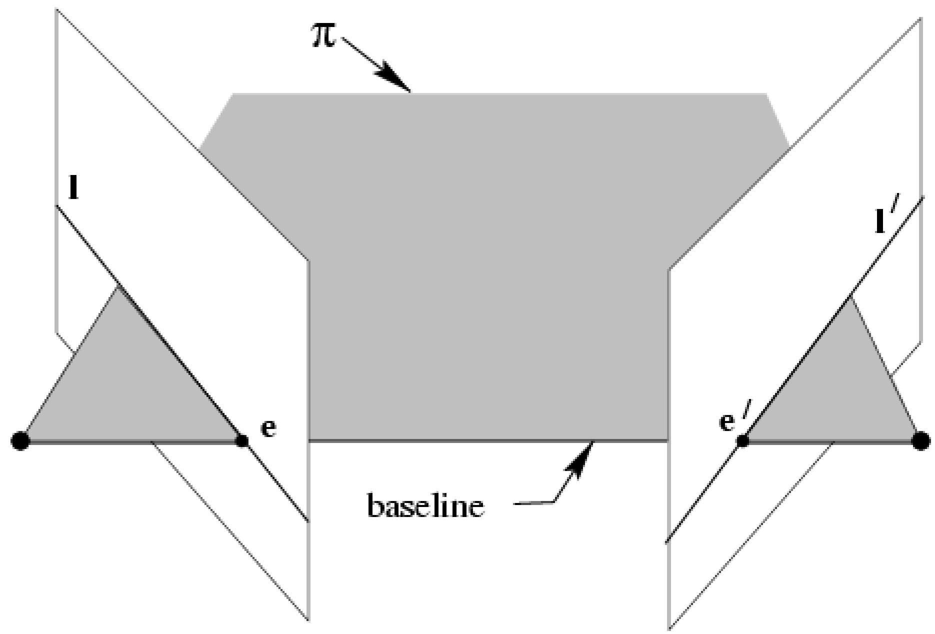
The epipolar geometry

epipolar pole

[epipolar geometry demo]

= intersection of baseline with image plane

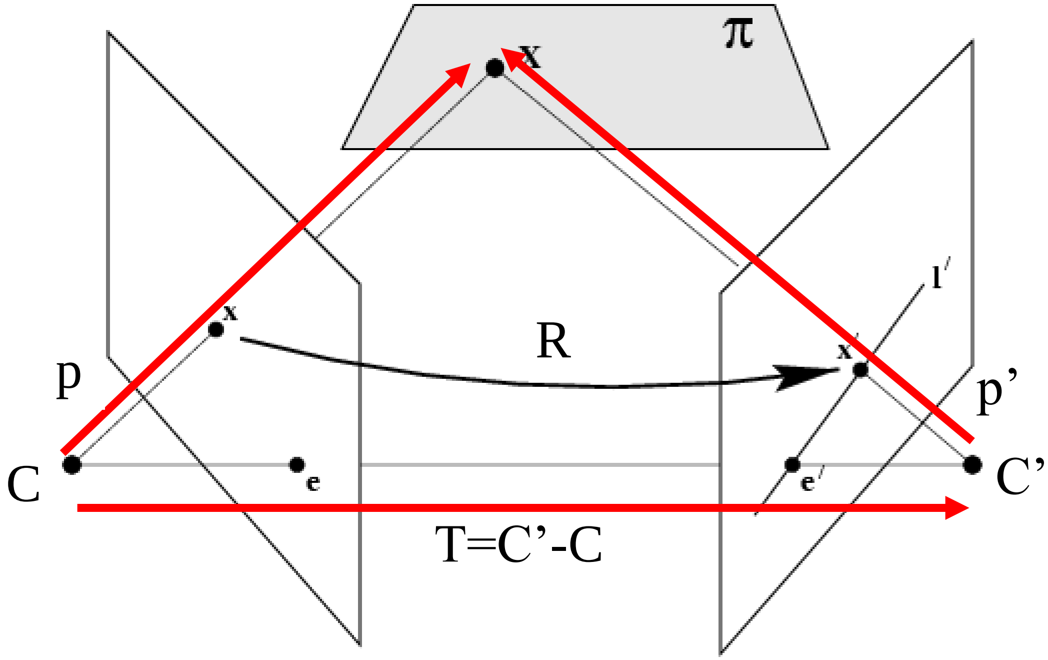
= projection of projection center in other image



epipolar plane = plane containing baseline

epipolar line = intersection of epipolar plane with image

The fundamental matrix F



Two reference frames are related via the extrinsic parameters

$$p' = R(p - T)$$

The equation of the epipolar plane through X is

$$\mathbf{X}^{T}(\mathbf{T}\times\mathbf{p}) = 0 \Rightarrow (\mathbf{R}^{T}\mathbf{p}' + \mathbf{T})^{T}(\mathbf{T}\times\mathbf{p}) = 0$$

The fundamental matrix F

$$(\mathbf{R}^{\mathrm{T}}\mathbf{p}')^{\mathrm{T}}(\mathbf{T}\times\mathbf{p}) = 0$$

$$\mathbf{T}\times\mathbf{p} = \mathbf{S}\mathbf{p}$$

$$\mathbf{S} = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$

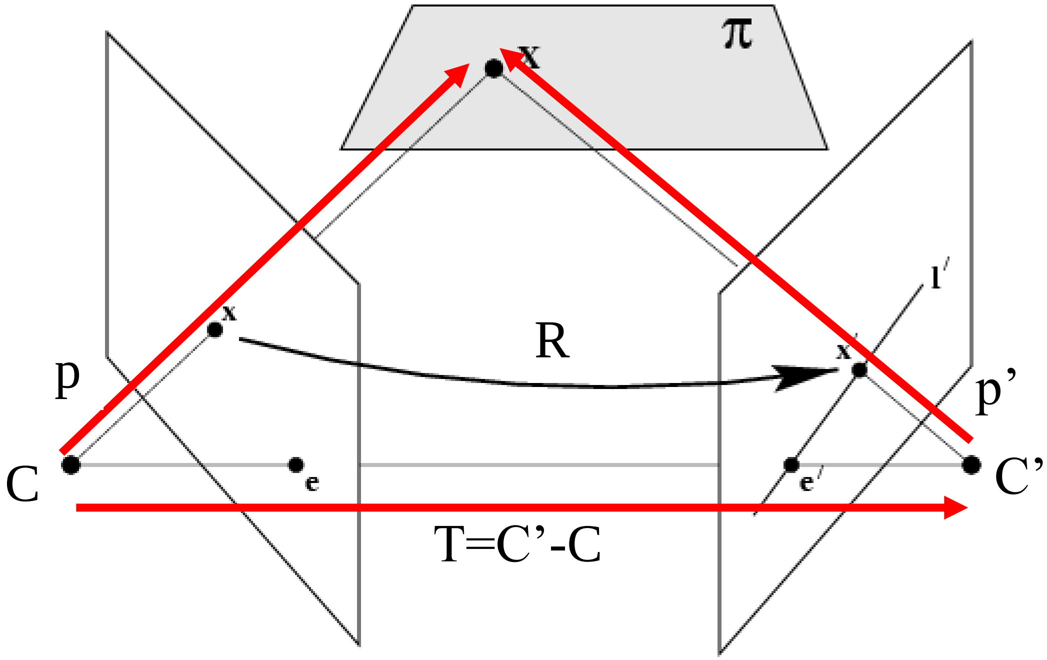
$$\Rightarrow \quad (\mathbf{R}^{\mathrm{T}}\mathbf{p}')^{\mathrm{T}}(\mathbf{S}\mathbf{p}) = 0$$

$$\Rightarrow \quad (\mathbf{p}'^{\mathrm{T}}\mathbf{R})(\mathbf{S}\mathbf{p}) = 0$$

$$\Rightarrow \quad \mathbf{p}'^{\mathrm{T}}\boxed{\mathbf{E}}\mathbf{p} = 0$$

essential matrix

The fundamental matrix F



$$\mathbf{p'}^{T}\,\mathbf{E}\mathbf{p} = 0$$

The fundamental matrix F

$$\mathbf{p'}^{\mathrm{T}}\mathbf{E}\mathbf{p} = 0$$

Let M and M' be the intrinsic matrices, then

$$\mathbf{p} = \mathbf{M}^{-1}\mathbf{x} \qquad \mathbf{p'} = \mathbf{M'}^{-1}\mathbf{x'}$$

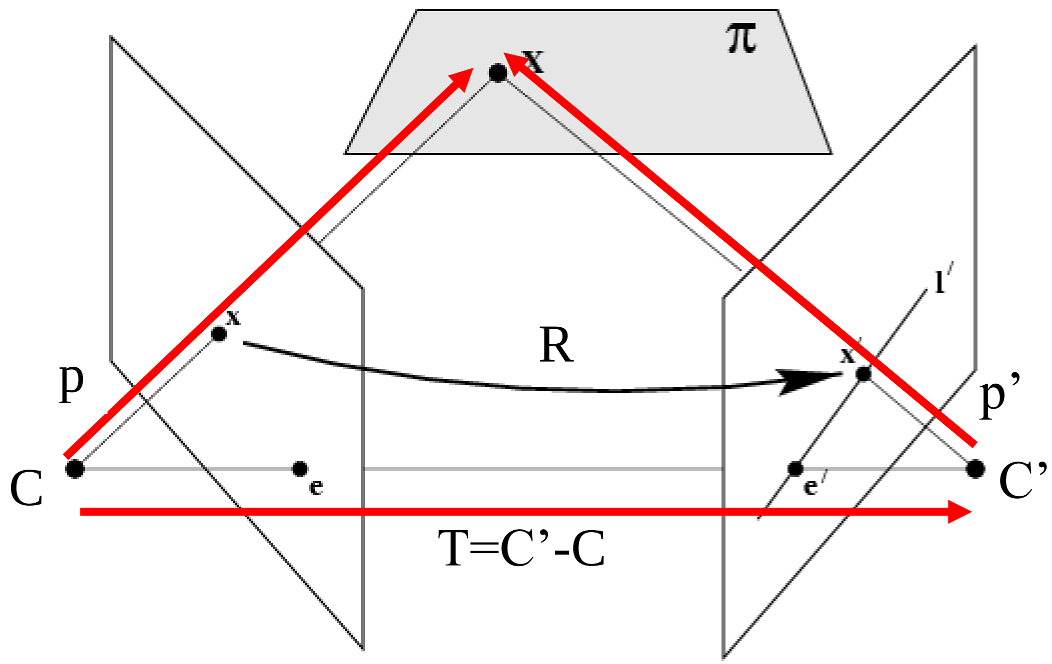
$$\Rightarrow (\mathbf{M'}^{-1}\mathbf{x'})^{\mathrm{T}}\mathbf{E}(\mathbf{M}^{-1}\mathbf{x}) = 0$$

$$\Rightarrow \mathbf{x'}^{\mathrm{T}}\boxed{\mathbf{M'}^{-\mathrm{T}}\mathbf{E}\mathbf{M}^{-1}}\mathbf{x} = 0$$

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

fundamental matrix

The fundamental matrix F



$$\mathbf{p'}^{\mathrm{T}}\,\mathbf{E}\mathbf{p} = 0$$

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

The fundamental matrix F

- The fundamental matrix is the algebraic representation of epipolar geometry

- The fundamental matrix satisfies the condition that for any pair of corresponding points $x \leftrightarrow x'$ in the two images

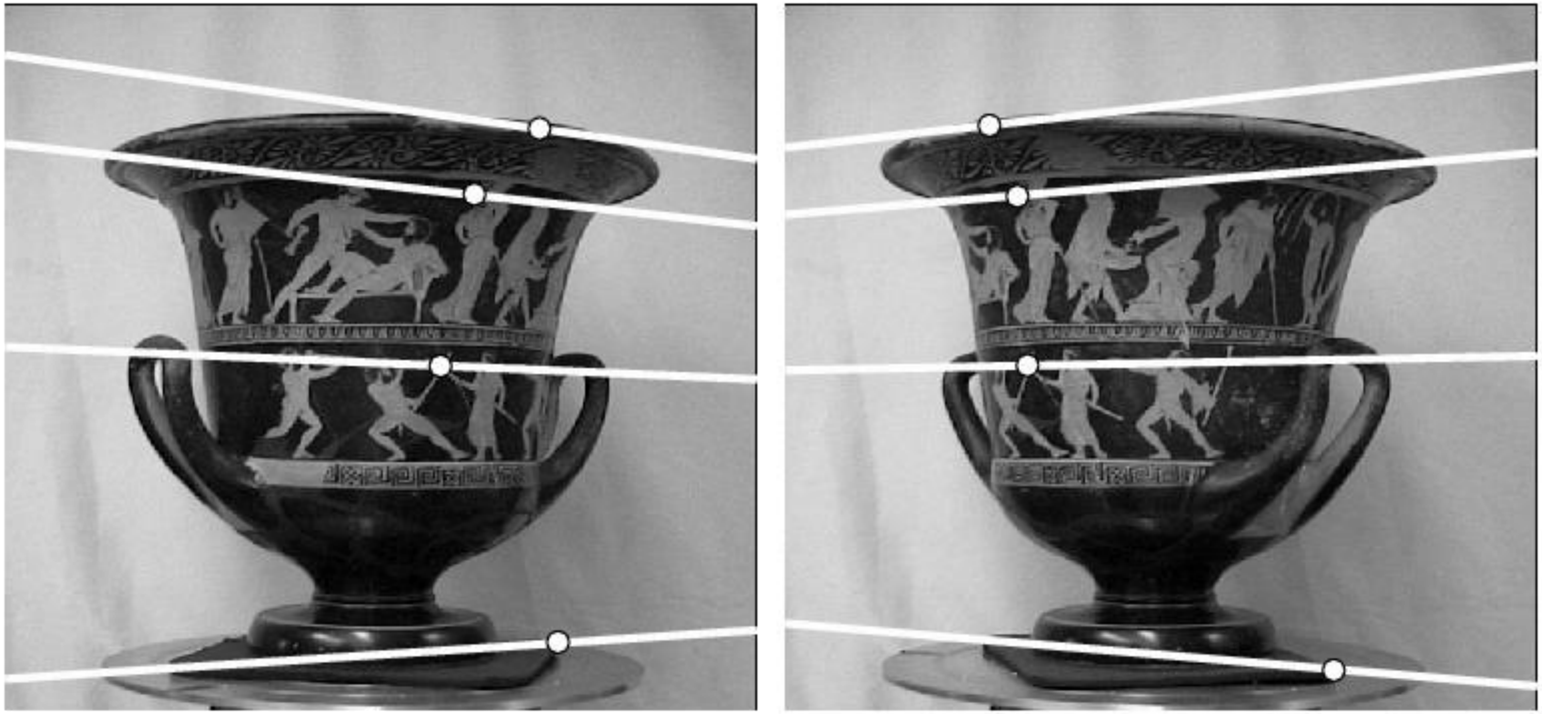
$$x'^{T} F x = 0 \qquad \left(x'^{T} l' = 0 \right)$$

The fundamental matrix F

F is the unique 3x3 rank 2 matrix that satisfies $x'^{T}Fx=0$ for all $x \leftrightarrow x'$

1. Transpose: if F is fundamental matrix for (P,P'), then F^{T} is fundamental matrix for (P',P)
2. Epipolar lines: $l'=Fx$ & $l=F^{T}x'$
3. Epipoles: on all epipolar lines, thus $e'^{T}Fx=0, \forall x$ $\Rightarrow e'^{T}F=0$, similarly $Fe=0$
4. F has 7 d.o.f. , i.e. 3x3-1(homogeneous)-1(rank2)
5. F is a correlation, projective mapping from a point x to a line $l'=Fx$ (not a proper correlation, i.e. not invertible)

The fundamental matrix F



- It can be used for
  - Simplifies matching
  - Allows to detect wrong matches

Estimation of 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 $\left\|\mathbf{Af}\right\|$, least eigenvector of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$

8-point algorithm

- To enforce that F is of rank 2, F is replaced by F' that minimizes $\|\mathbf{F}-\mathbf{F}'\|$ subject to $\det\mathbf{F}'=0$.

- It is achieved by SVD. Let $\mathbf{F}=\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$, 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 $\mathbf{F}'=\mathbf{U}\boldsymbol{\Sigma}'\mathbf{V}^{\mathrm{T}}$ 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

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

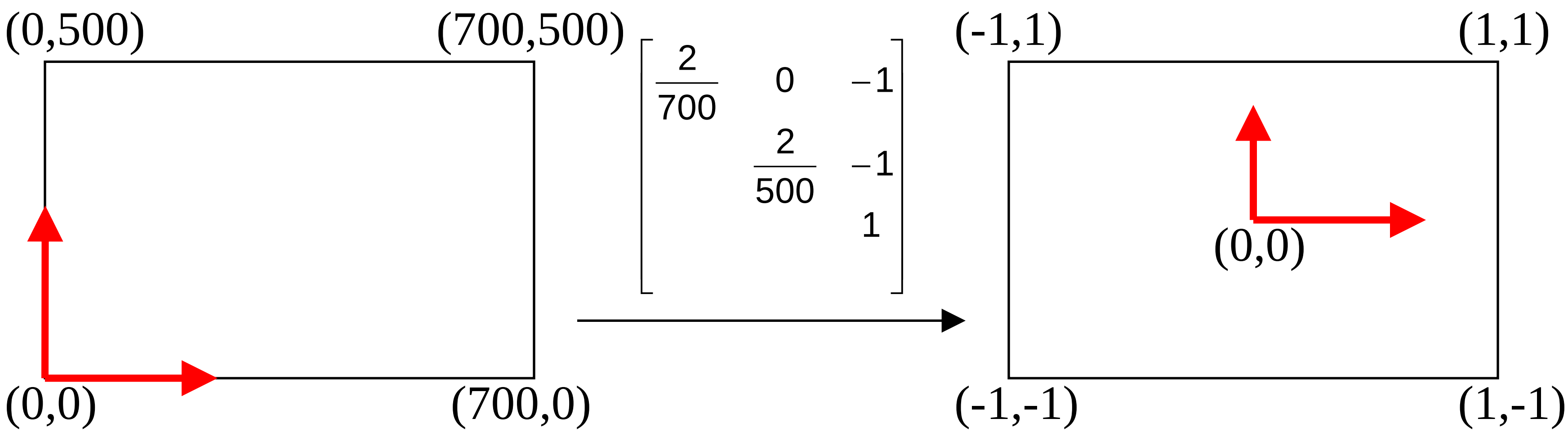
!

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. $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

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

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

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;
```

Normalization

```
function [newpts, T] = normalise2dpts(pts)

    c = mean(pts(1:2,:)')';   % Centroid
    newp(1,:) = pts(1,:)-c(1); % Shift origin to centroid.
    newp(2,:) = pts(2,:)-c(2);

    meandist = mean(sqrt(newp(1,:).^2 + newp(2,:).^2));
    scale = sqrt(2)/meandist;

    T = [scale      0    -scale*c(1)
           0     scale  -scale*c(2)
           0       0          1      ];
    newpts = T*pts;
```

RANSAC

repeat

select minimal sample (8 matches)

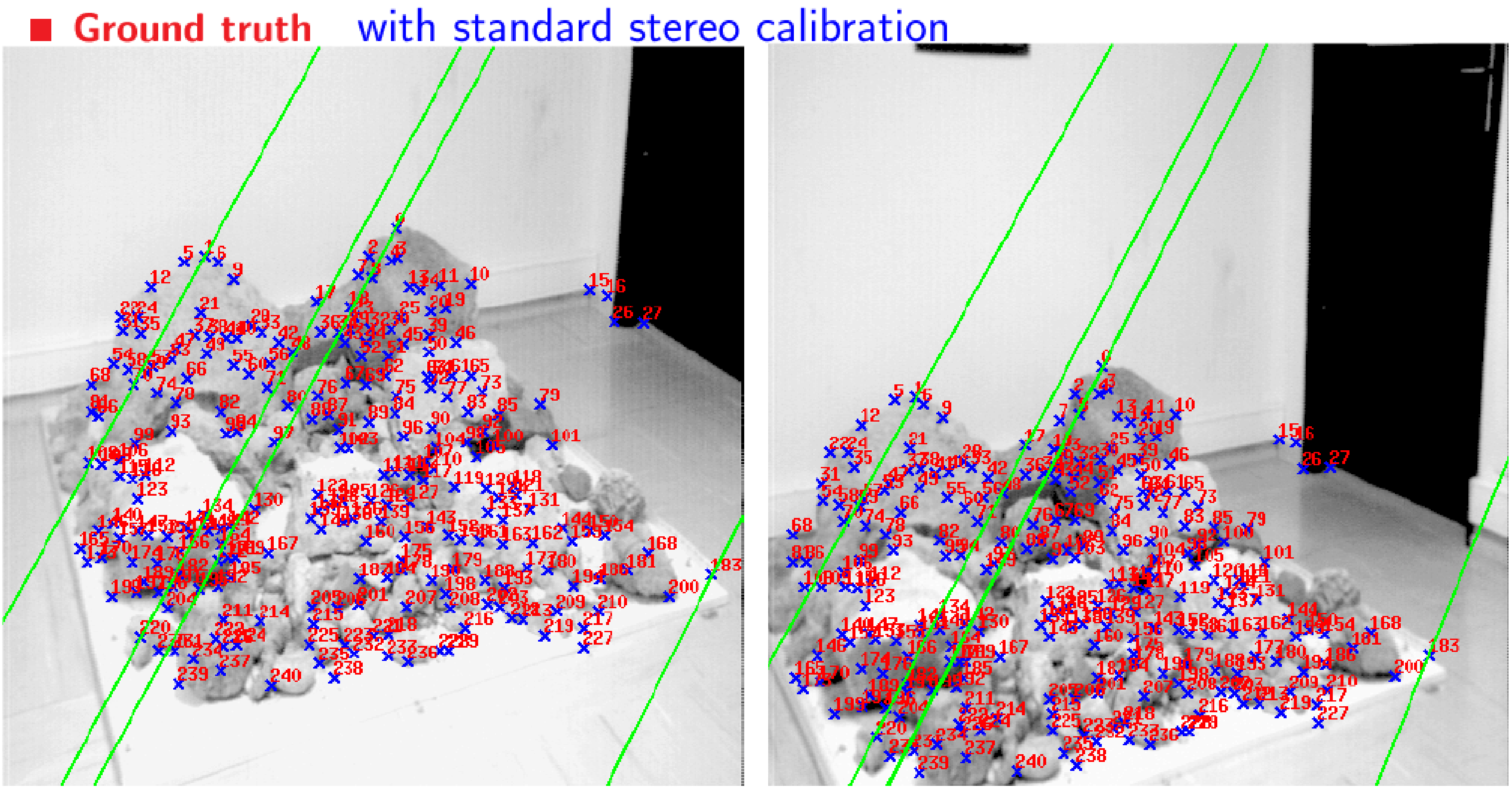
compute solution(s) for F

determine inliers

until Γ(#*inliers*,#*samples*)>95% or too many times

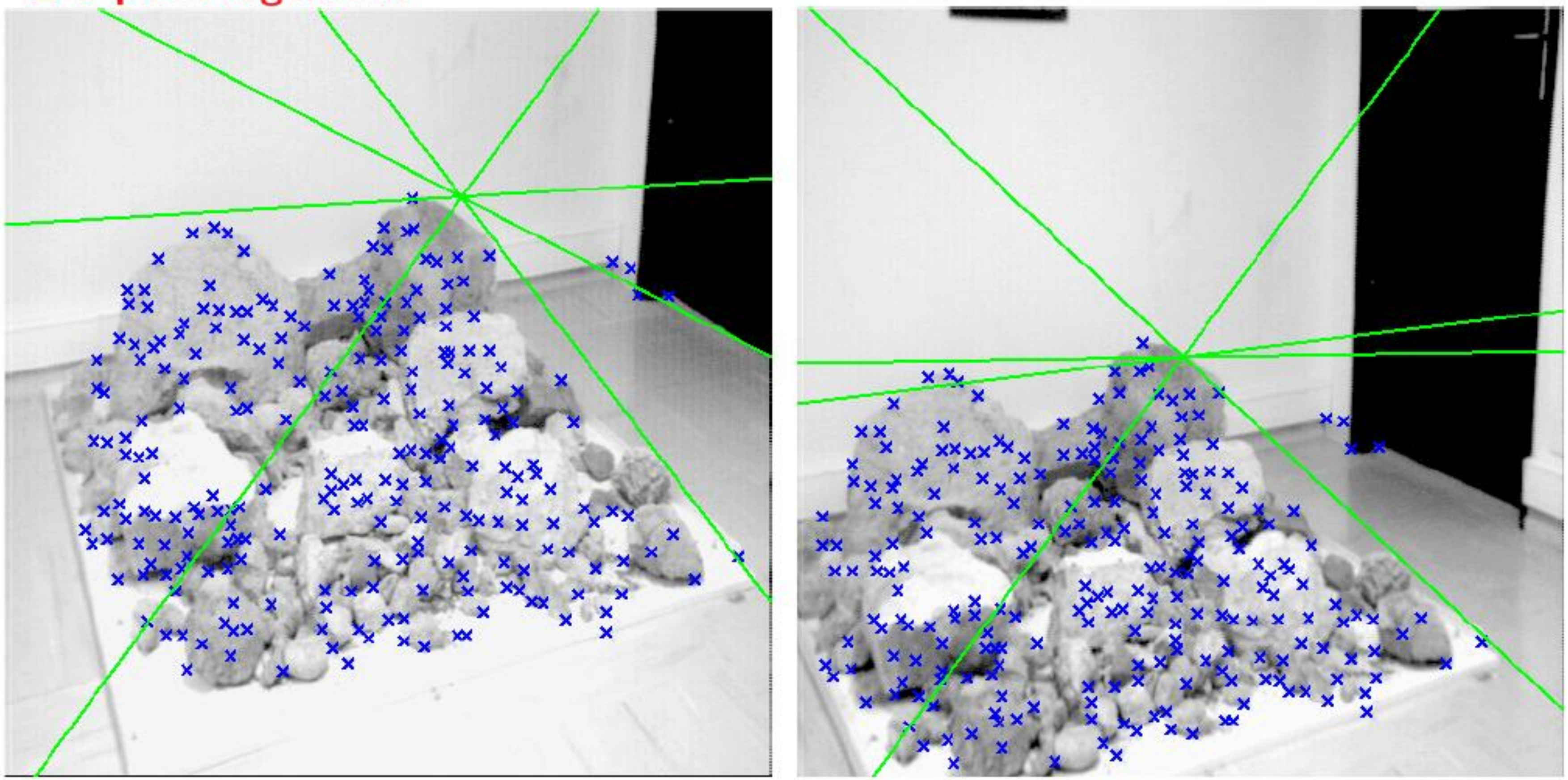
compute F based on all inliers

Results (ground truth)

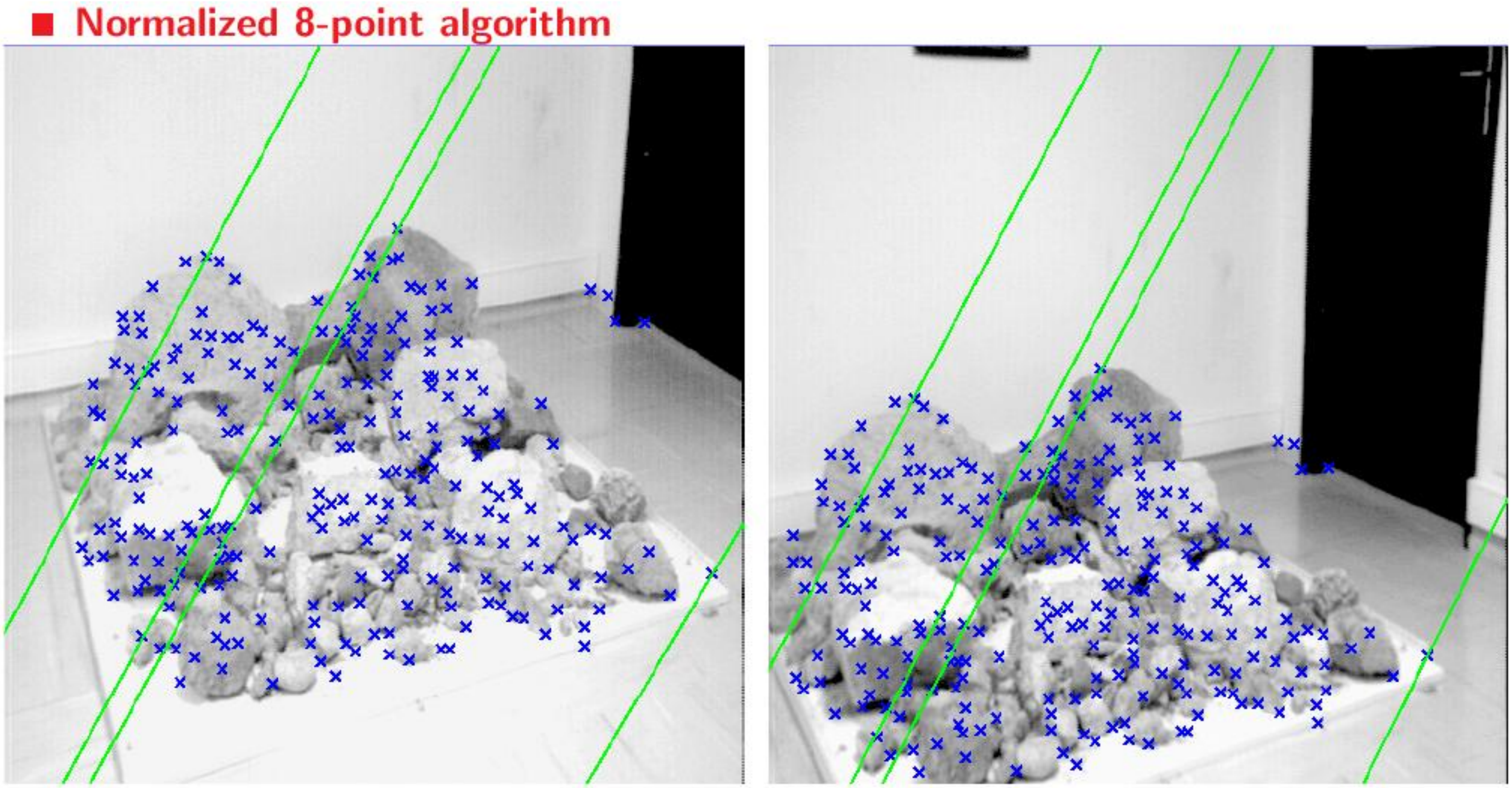


Results (8-point algorithm)

- 8-point algorithm



Results (normalized 8-point algorithm)



From F to R, T

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

$$\mathbf{x'}^{\mathrm{T}}\,\mathbf{M'}^{-\mathrm{T}}\,\mathbf{E}\mathbf{M}^{-1}\mathbf{x} = 0$$

$$\mathbf{E} = \mathbf{M'}^{\mathrm{T}}\,\mathbf{F}\mathbf{M}$$

If we know camera parameters

$$\mathbf{E} = \mathbf{R}[\mathbf{T}]_{\times}$$

Hartley and Zisserman, Multiple View Geometry, 2nd edition, pp 259

Triangulation

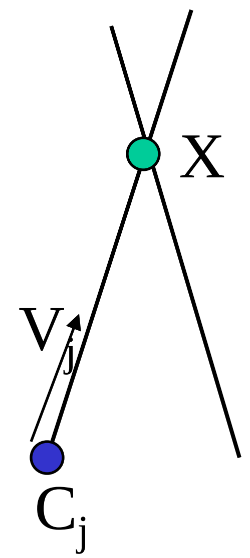
- Problem: Given some points in *correspondence* across two or more images (taken from calibrated cameras), $\{(u_j,v_j)\}$, compute the 3D location $\mathbf{X}$

Triangulation

- **Method I**: intersect viewing rays in 3D, minimize:

$$\arg\min_{\mathbf{X}} \sum_j \|\mathbf{C}_j + s\mathbf{V}_j - \mathbf{X}\|$$

  - **X** is the unknown 3D point
  - $\mathbf{C}_j$ is the optical center of camera *j*
  - $\mathbf{V}_j$ is the *viewing ray* for pixel (u_j,v_j)
  - s_j is unknown distance along $\mathbf{V}_j$
- Advantage: geometrically intuitive



Triangulation

- **Method II**: solve linear equations in **X**
  - advantage: very simple

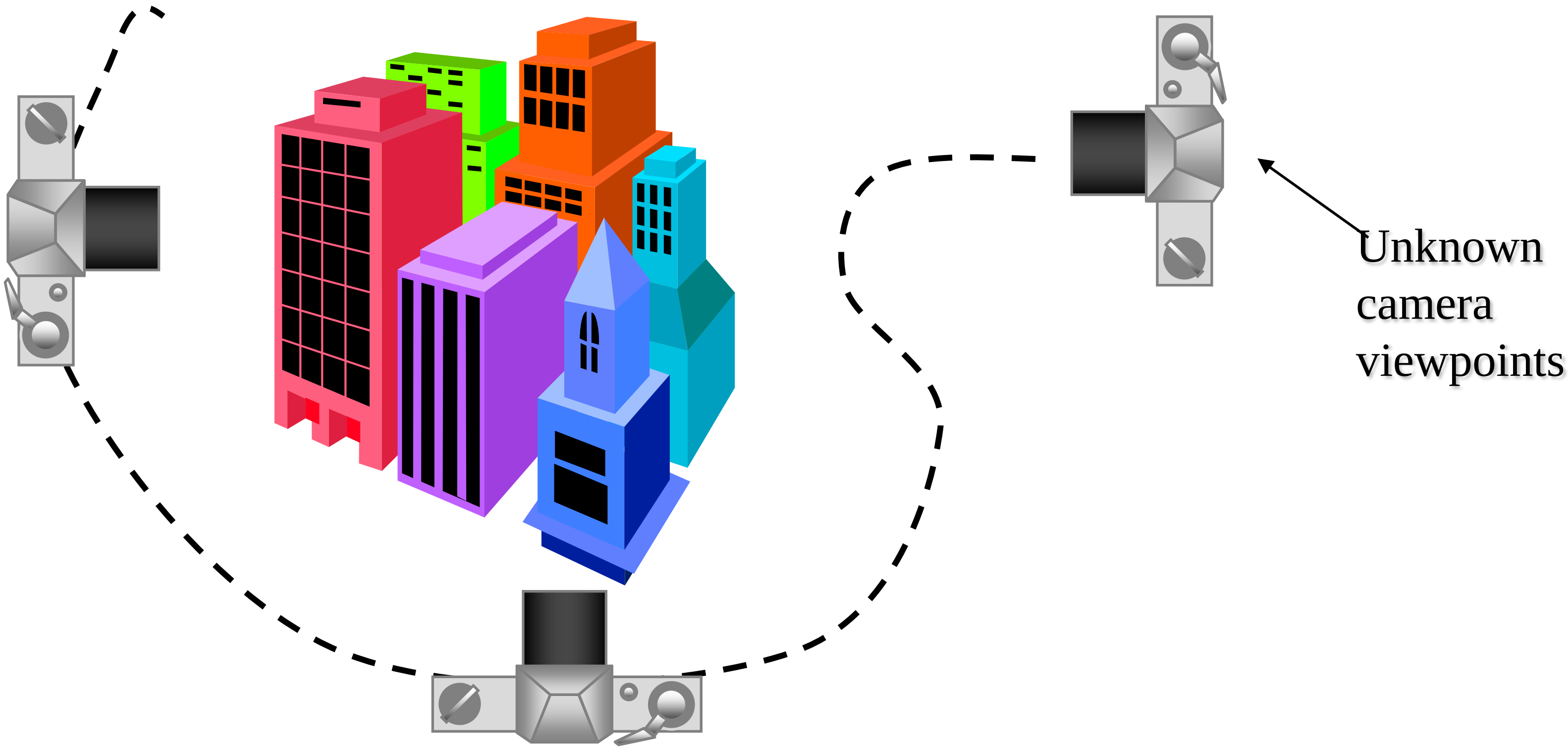
$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

- **Method III**: non-linear minimization
  - advantage: most accurate (image plane error)

Structure from motion

Structure from motion



structure for motion: automatic recovery of **camera motion** and **scene structure** from two or more images. It is a self calibration technique and called *automatic camera tracking* or *matchmoving*.

Applications

- For computer vision, multiple-view shape reconstruction, novel view synthesis and autonomous vehicle navigation.
- For film production, seamless insertion of CGI into live-action backgrounds

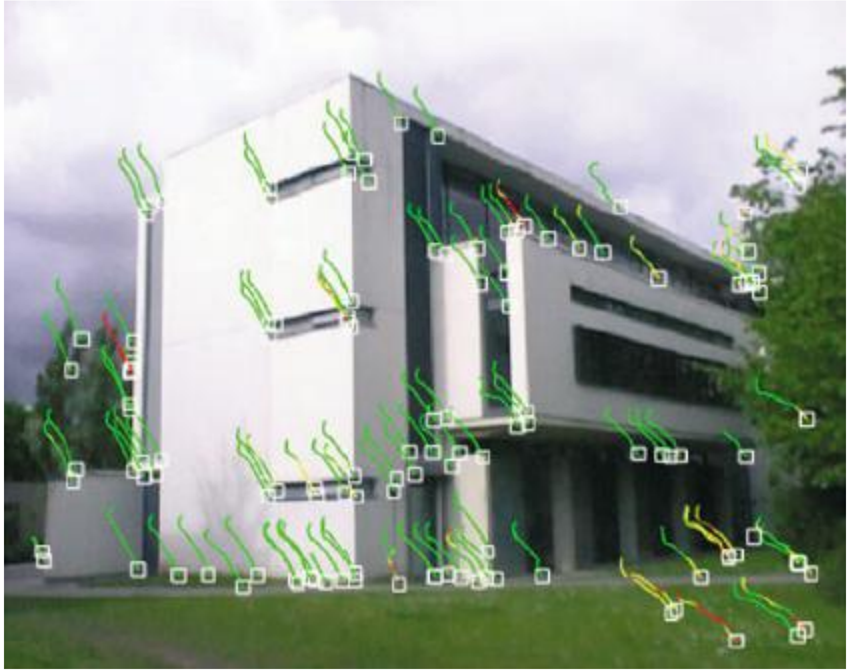
Structure from motion

2D feature tracking → 3D estimation → optimization (bundle adjust) → geometry fitting

SFM pipeline

Structure from motion

- Step 1: Track Features
  - Detect good features, Shi & Tomasi, SIFT
  - Find correspondences between frames
    - Lucas & Kanade-style motion estimation
    - window-based correlation
    - SIFT matching

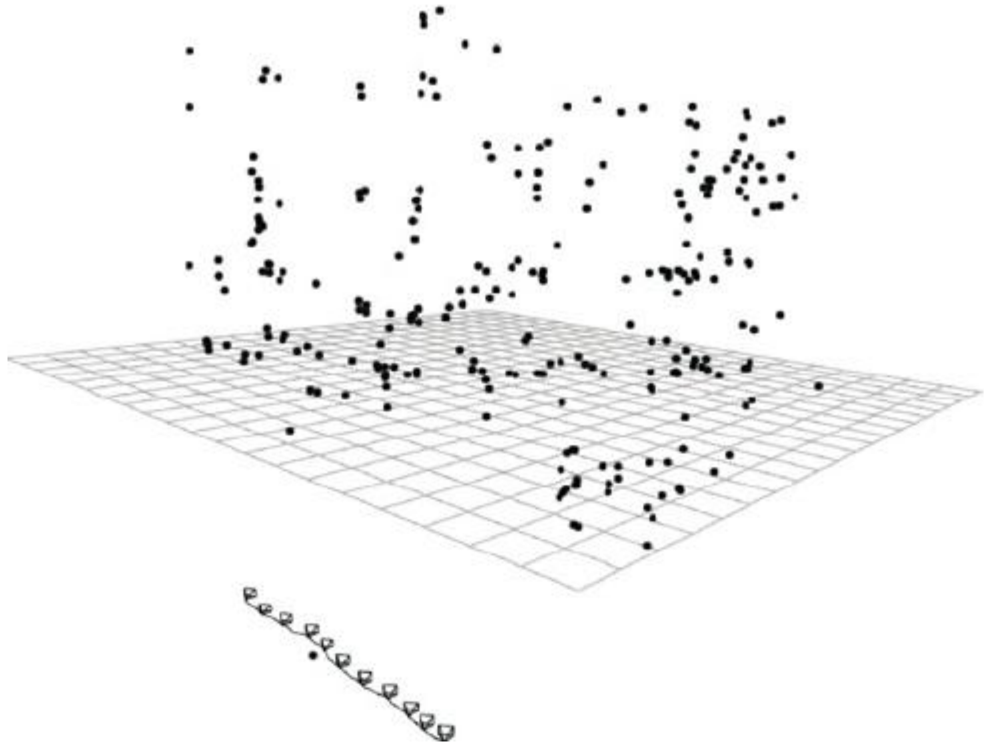


Structure from Motion

- Step 2: Estimate Motion and Structure
  - Simplified projection model, e.g., **[Tomasi 92]**
  - 2 or 3 views at a time **[Hartley 00]**

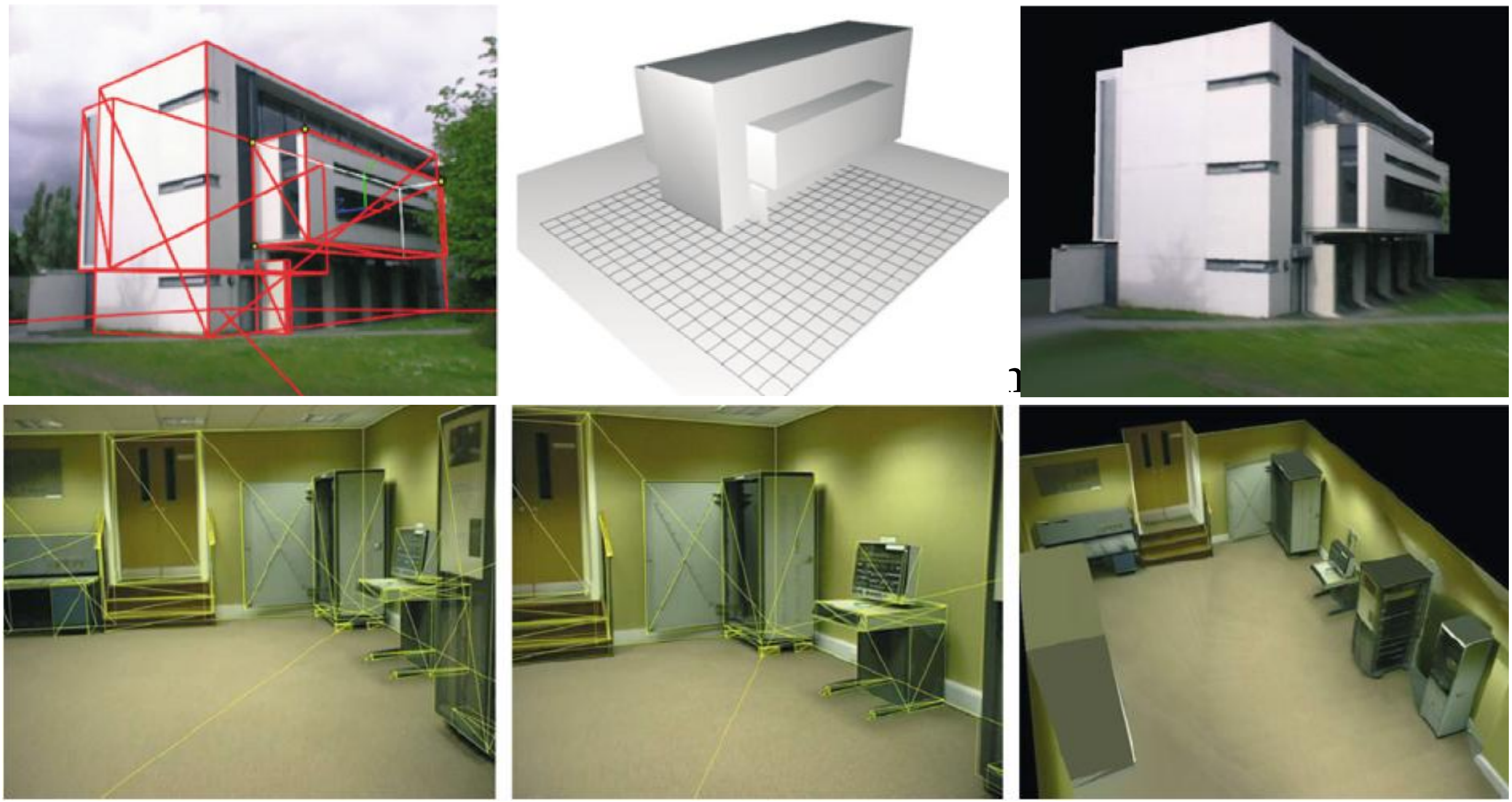
Structure from Motion

- Step 3: Refine estimates
  - “Bundle adjustment” in photogrammetry
  - Other iterative methods



Structure from Motion

- Step 4: Recover surfaces (image-based triangulation, silhouettes, stereo…)

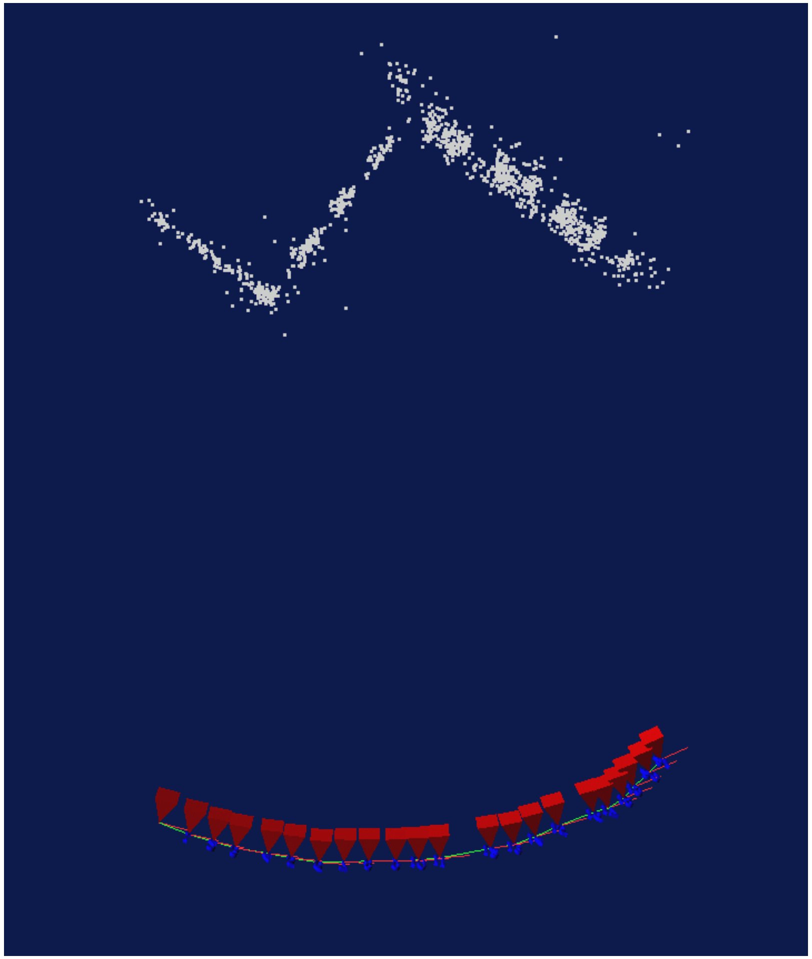


Example : Photo Tourism

(a) (b) (c)

Factorization methods

Problem statement



SFM under orthographic projection

2D image point

orthographic projection matrix

3D scene point

image offset

$$\underset{2\times 1}{\mathbf{q}} = \underset{2\times 3}{\Pi}\,\underset{3\times 1}{\mathbf{p}} + \underset{2\times 1}{\mathbf{t}}$$

- Trick
  - Choose scene origin to be centroid of 3D points
  - Choose image origins to be centroid of 2D points
  - Allows us to drop the camera translation:

$$\mathbf{q} = \Pi \mathbf{p}$$

factorization (Tomasi & Kanade)

projection of *n* features in one image:

$$\underset{2\times n}{\begin{bmatrix}\mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n\end{bmatrix}} = \underset{2\times 3}{\prod} \underset{3\times n}{\begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_n\end{bmatrix}}$$

projection of *n* features in *m* images

$$\underset{2m\times n}{\begin{bmatrix}\mathbf{q}_{11} & \mathbf{q}_{12} & \cdots & \mathbf{q}_{1n}\\ \mathbf{q}_{21} & \mathbf{q}_{22} & \cdots & \mathbf{q}_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{q}_{m1} & \mathbf{q}_{m2} & \cdots & \mathbf{q}_{mn}\end{bmatrix}} = \underset{2m\times 3}{\begin{bmatrix}\mathbf{\Pi}_1\\ \mathbf{\Pi}_2\\ \vdots\\ \mathbf{\Pi}_m\end{bmatrix}} \underset{3\times n}{\begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_n\end{bmatrix}}$$

W measurement M motion S shape

Key Observation: *rank*(W) <= 3

Factorization

$$\text{known} \; \text{—} \; \underset{2m\times n}{\mathbf{W}} = \boxed{\underset{2m\times 3}{\mathbf{M}} \; \underset{3\times n}{\mathbf{S}}} \; \text{—} \; \text{solve for}$$

- Factorization Technique
  - W is at most rank 3 (assuming no noise)
  - We can use *singular value decomposition* to factor W:

$$\underset{2m\times n}{\mathbf{W}} = \underset{2m\times 3}{\mathbf{M}'} \; \underset{3\times n}{\mathbf{S}'}$$

  - S' differs from S by a linear transformation *A*:

$$\mathbf{W} = \mathbf{M}'\mathbf{S}' = (\mathbf{M}\mathbf{A}^{-1})(\mathbf{A}\mathbf{S})$$

  - Solve for A by enforcing *metric* constraints on *M*

Metric constraints

- Orthographic Camera
  - Rows of Π are orthonormal: $$\Pi_i \Pi_i^{\mathrm{T}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
- Enforcing "Metric" Constraints
  - Compute **A** such that rows of **M** have these properties

$$\mathbf{M'A} = \mathbf{M}$$

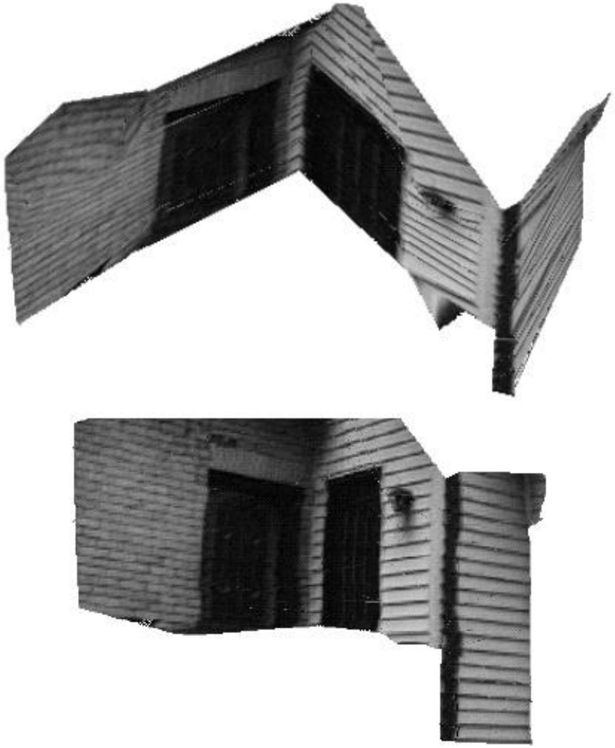
Trick (not in original Tomasi/Kanade paper, but in followup work)

- Constraints are linear in AA^T :

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \Pi_i \Pi_i^{\mathrm{T}} = \Pi'_i \mathbf{A} \left(\Pi_i'^{\mathrm{T}} \mathbf{A} \right)^T = \Pi'_i \mathbf{G} \Pi_i'^{\mathrm{T}} \qquad \textit{where } \mathbf{G} = \mathbf{A}\mathbf{A}^T$$

- Solve for G first by writing equations for every Π_i in M
- Then $G = AA^T$ by SVD

Results



Extensions to factorization methods

- Paraperspective [Poelman & Kanade, PAMI 97]
- Sequential Factorization [Morita & Kanade, PAMI 97]
- Factorization under perspective [Christy & Horaud, PAMI 96] [Sturm & Triggs, ECCV 96]
- Factorization with Uncertainty [Anandan & Irani, IJCV 2002]

Bundle adjustment

Structure from motion

$$\begin{aligned} \hat{u}_{ij} &= f(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i}) \\ \hat{v}_{ij} &= g(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i}) \end{aligned}$$

- How many points do we need to match?
- 2 frames:

  (***R***,***t***): 5 dof + $3n$ point locations $\leq$

  $4n$ point measurements $\Rightarrow$

  $n \geq 5$
- k frames:

  $6(k-1)-1 + 3n \leq 2kn$
- always want to use many more

Bundle Adjustment

$$\begin{aligned}\hat{u}_{ij} &= f(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i}) \\ \hat{v}_{ij} &= g(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i})\end{aligned}$$

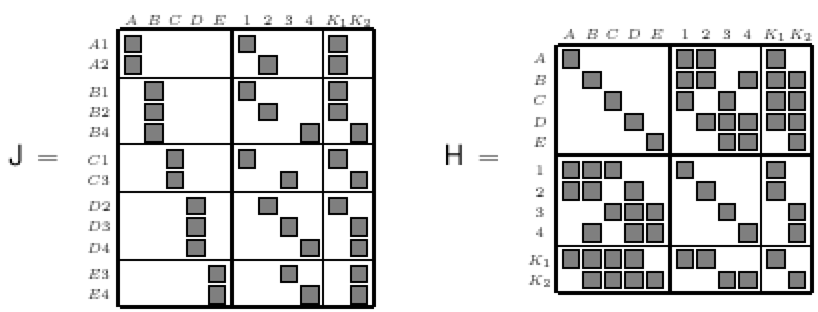
- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers

Lots of parameters: sparsity

$$
\begin{aligned}
\hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\
\hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)
\end{aligned}
$$

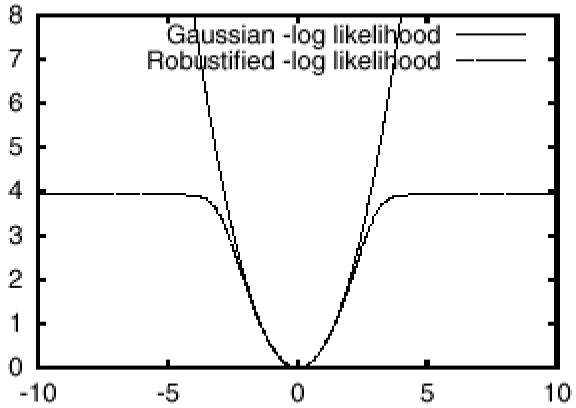
- Only a few entries in Jacobian are non-zero

$$
\frac{\partial \hat{u}_{ij}}{\partial \mathbf{K}}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{R}_j}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{t}_j}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{x}_i},
$$



Robust error models

- Outlier rejection
  - use robust penalty applied to each set of joint measurements

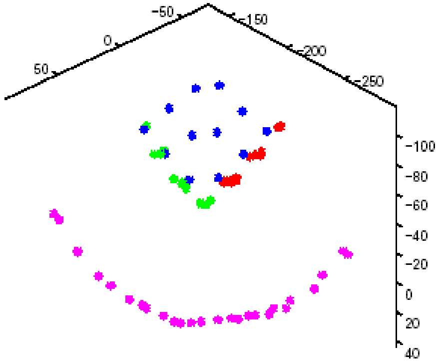


  - for extremely bad data, use random sampling [RANSAC, Fischler & Bolles, CACM'81]

$$\sum_i \sigma_i^{-2} \rho \left(\sqrt{(u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2} \right)$$

Structure from motion: limitations

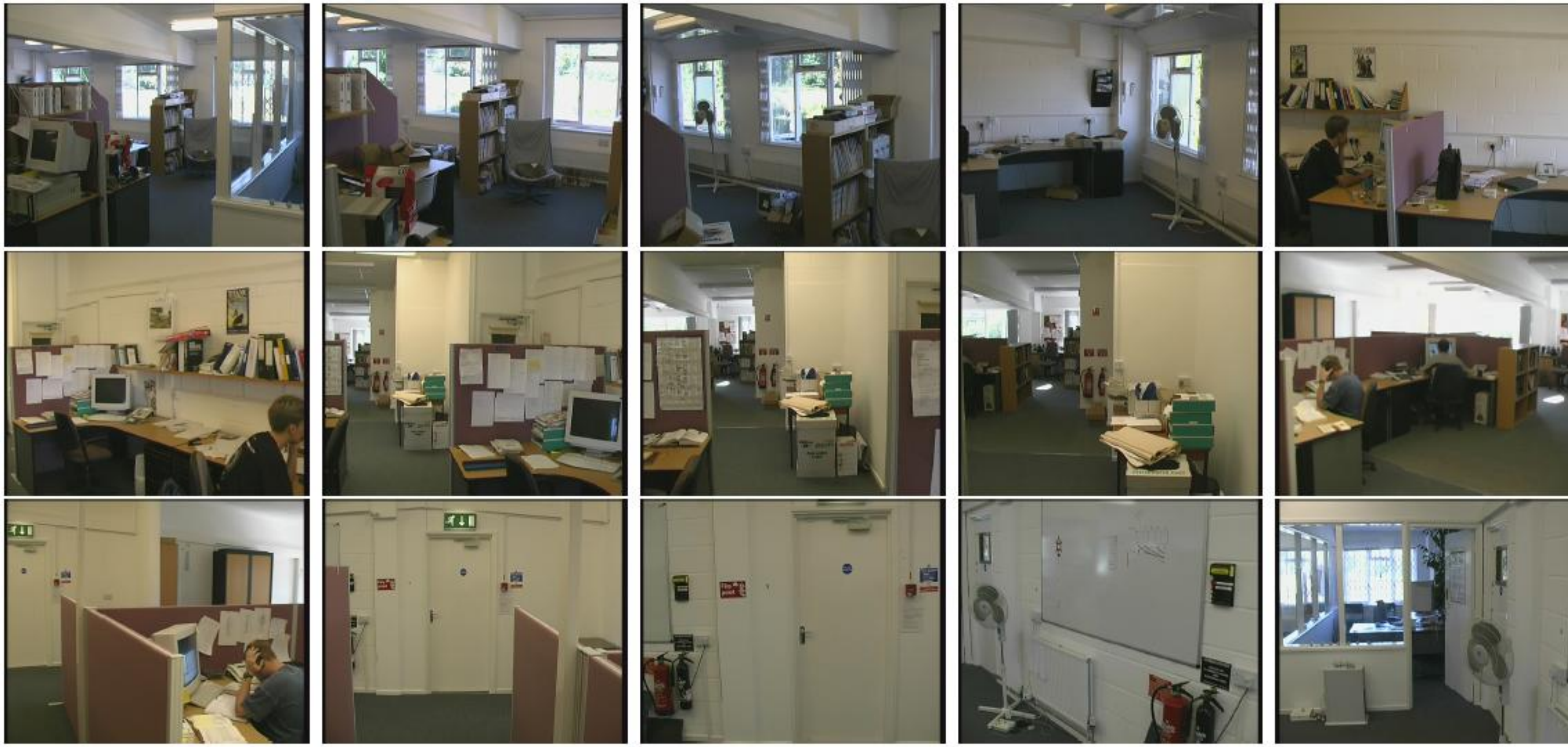
- Very difficult to reliably estimate ***metric*** structure and motion unless:
  - large (x or y) rotation *or*
  - large field of view and depth variation
- Camera calibration important for Euclidean reconstructions
- Need good feature tracker
- Lens distortion



Issues in SFM

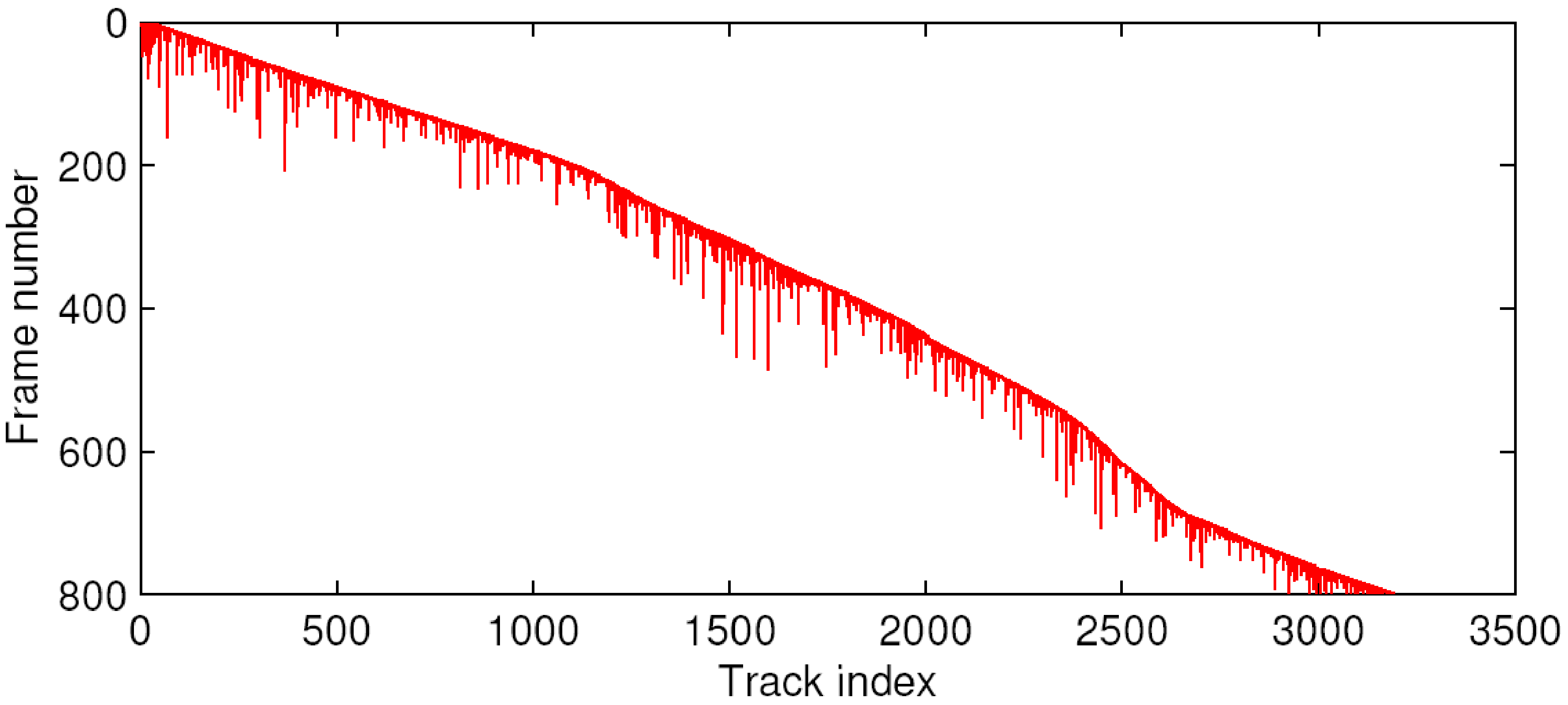
- Track lifetime
- Nonlinear lens distortion
- Degeneracy and critical surfaces
- Prior knowledge and scene constraints
- Multiple motions

Track lifetime



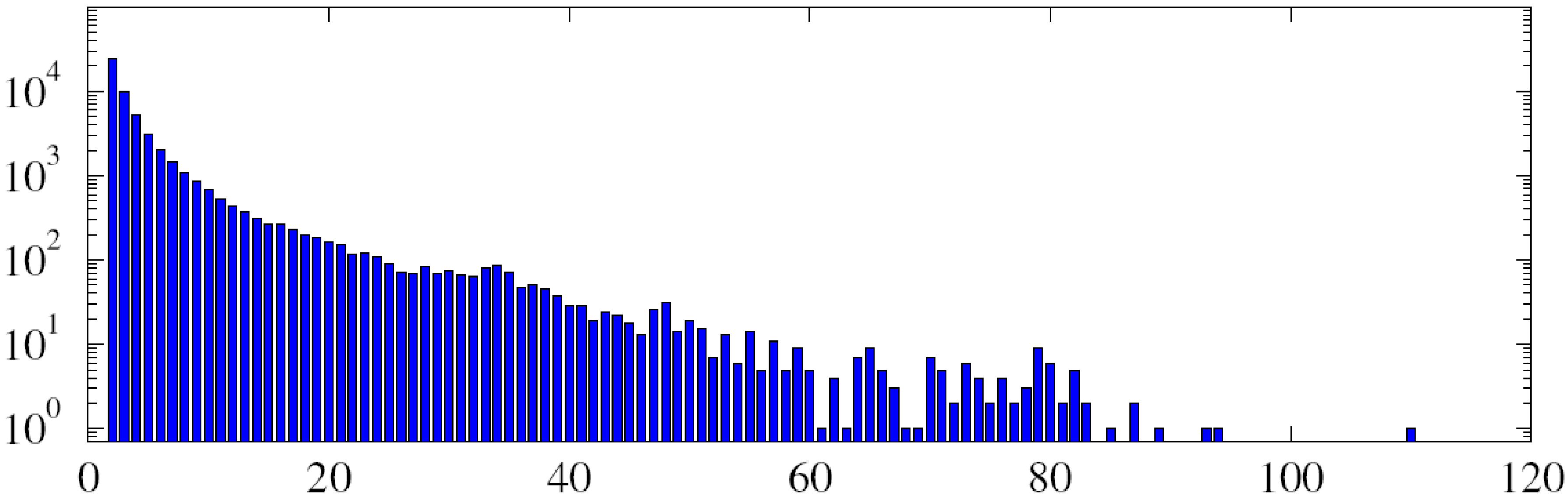
every 50th frame of a 800-frame sequence

Track lifetime



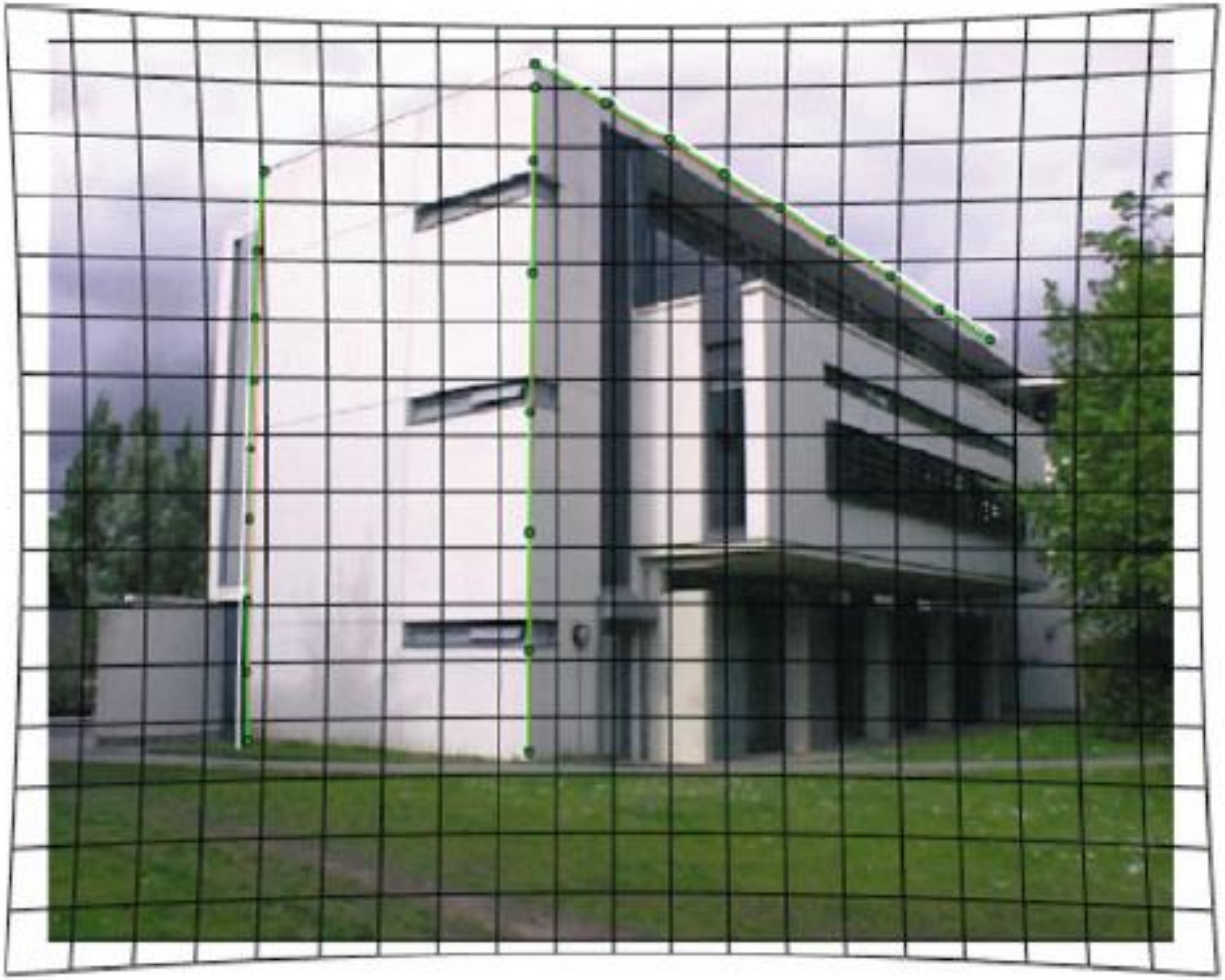
lifetime of 3192 tracks from the previous sequence

Track lifetime

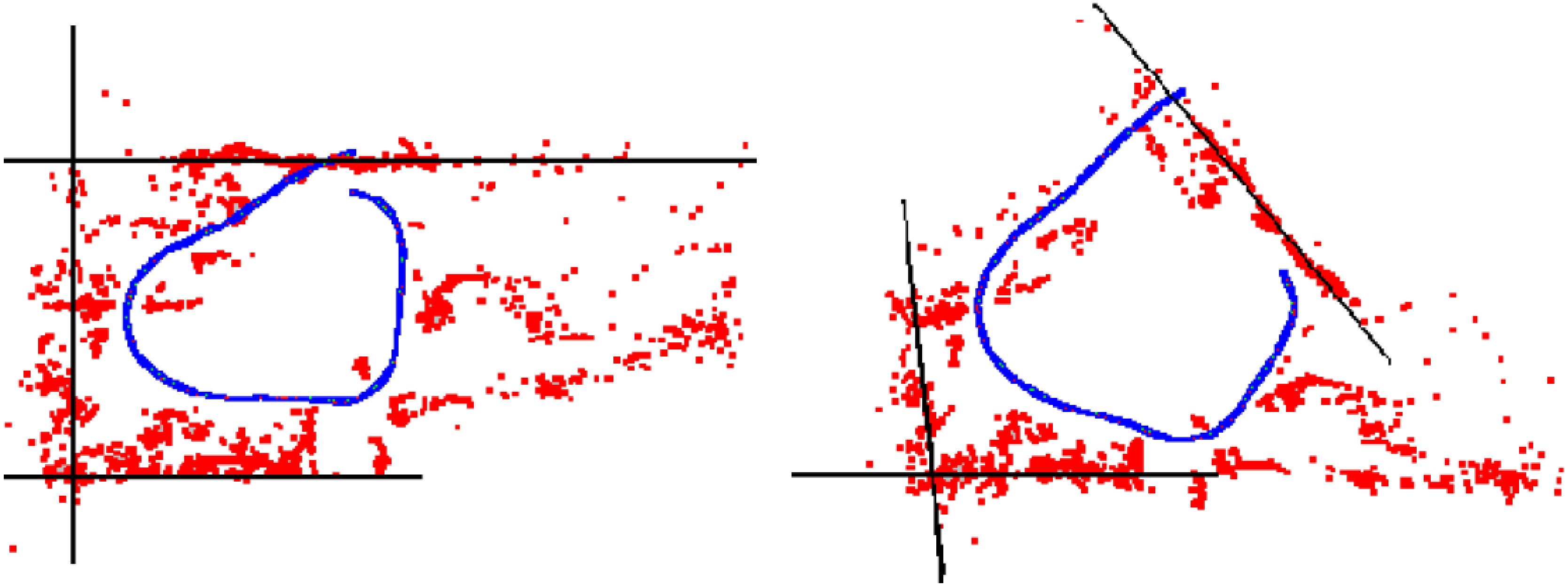


track length histogram

Nonlinear lens distortion



Nonlinear lens distortion

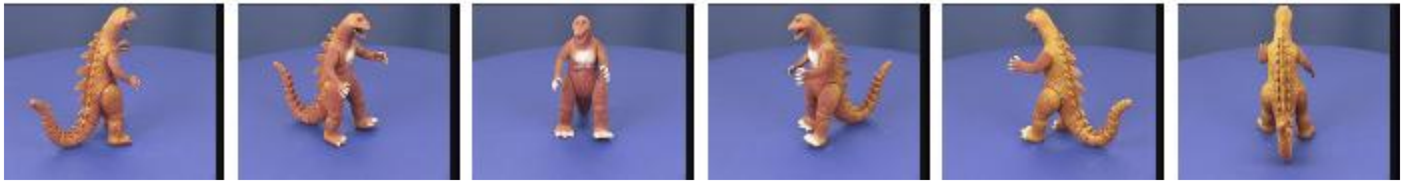


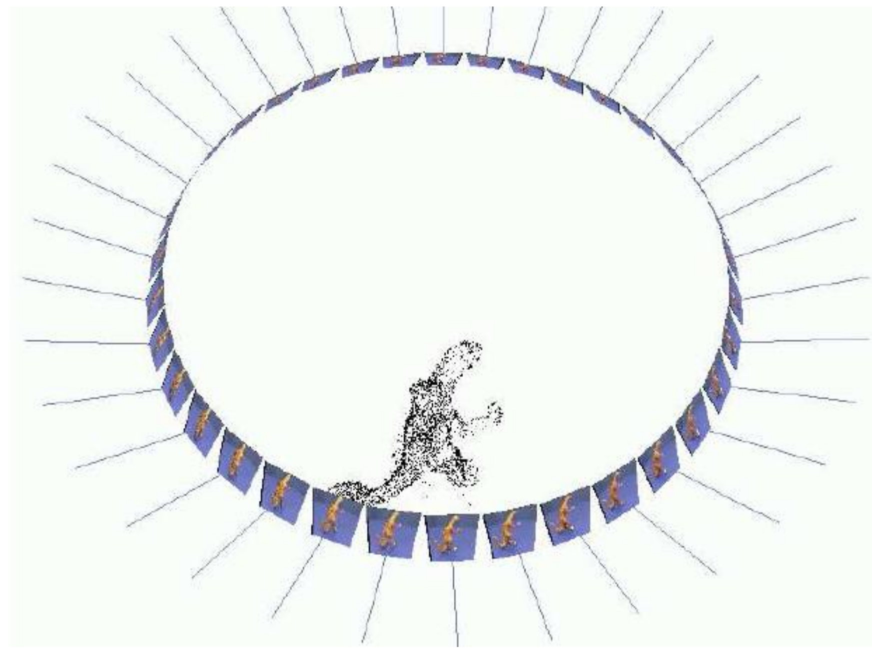
effect of lens distortion

Prior knowledge and scene constraints

add a constraint that several lines are parallel

Prior knowledge and scene constraints





add a constraint that it is a turntable sequence

Applications of Structure from Motion

Jurassic park

PhotoSynth

http://labs.live.com/photosynth/