Structure from motion

Digital Visual Effects

Yung-Yu Chuang

with slides by Richard Szeliski, Steve Seitz, Zhengyou Zhang and Marc Pollefyes

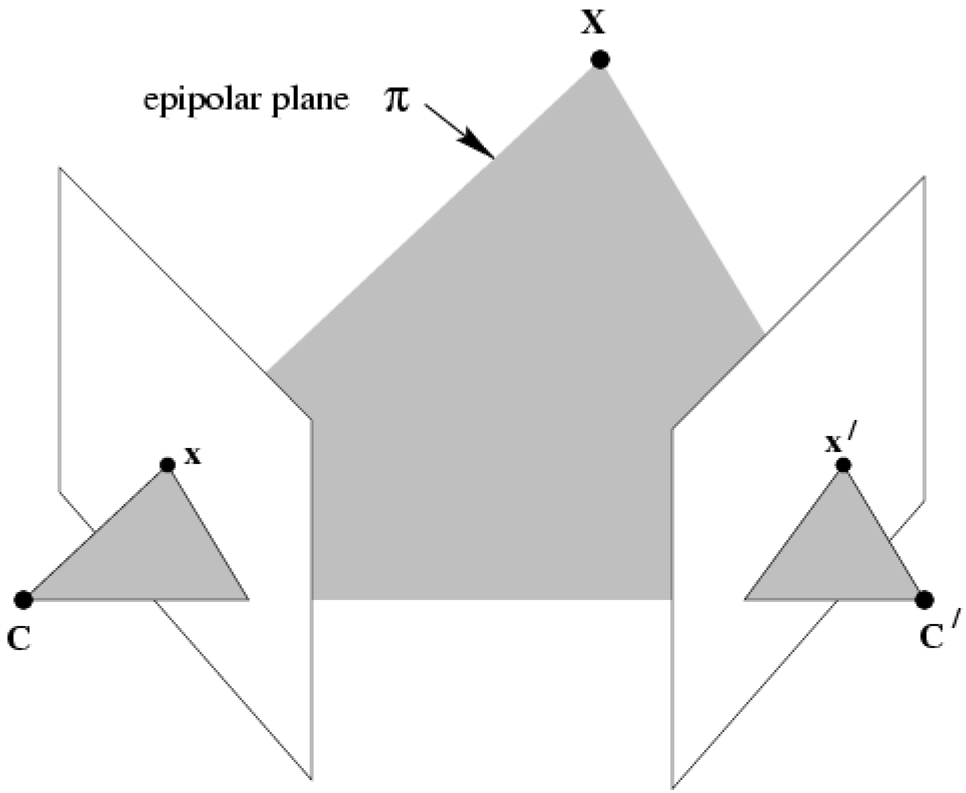
Outline

- Epipolar geometry and fundamental matrix
- Structure from motion
- Factorization method
- Bundle adjustment
- Applications

Epipolar geometry & fundamental matrix

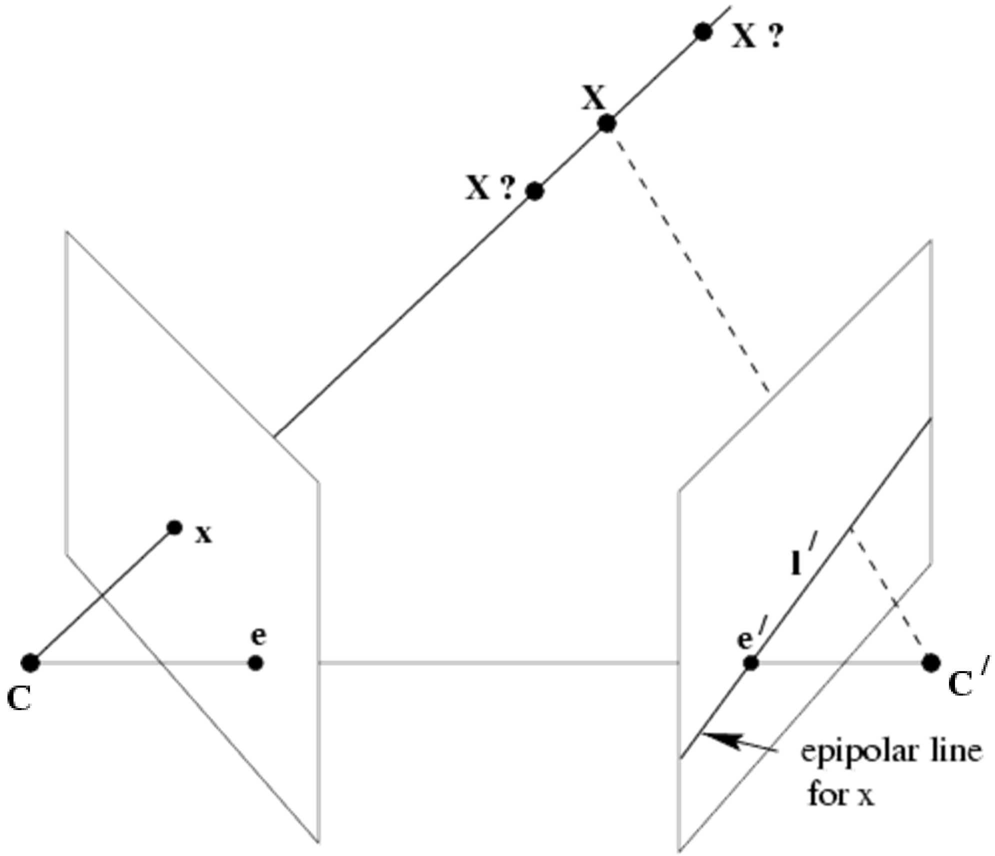
The epipolar geometry

epipolar geometry demo



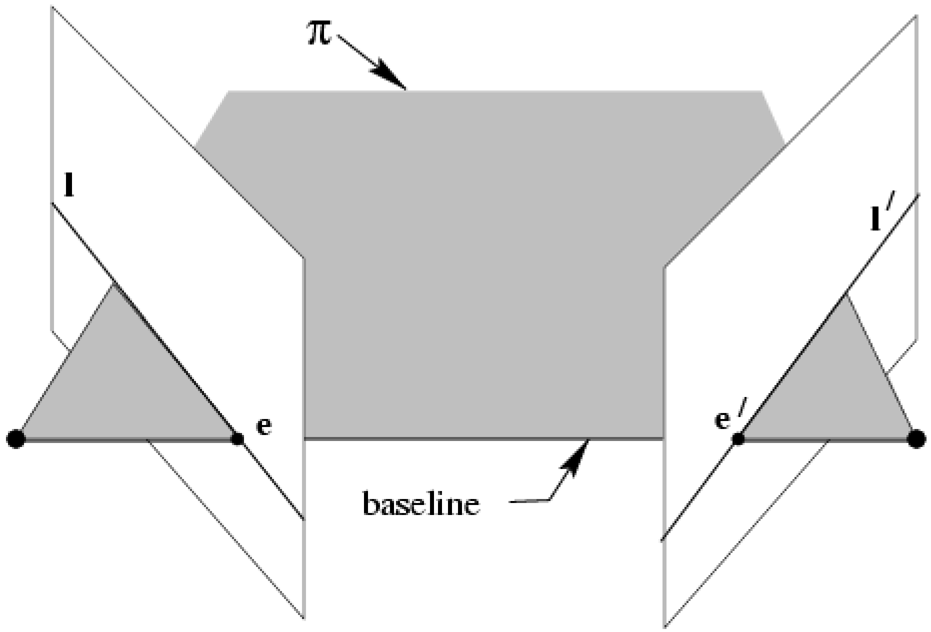
C, C', x, x' and X are coplanar

The epipolar geometry



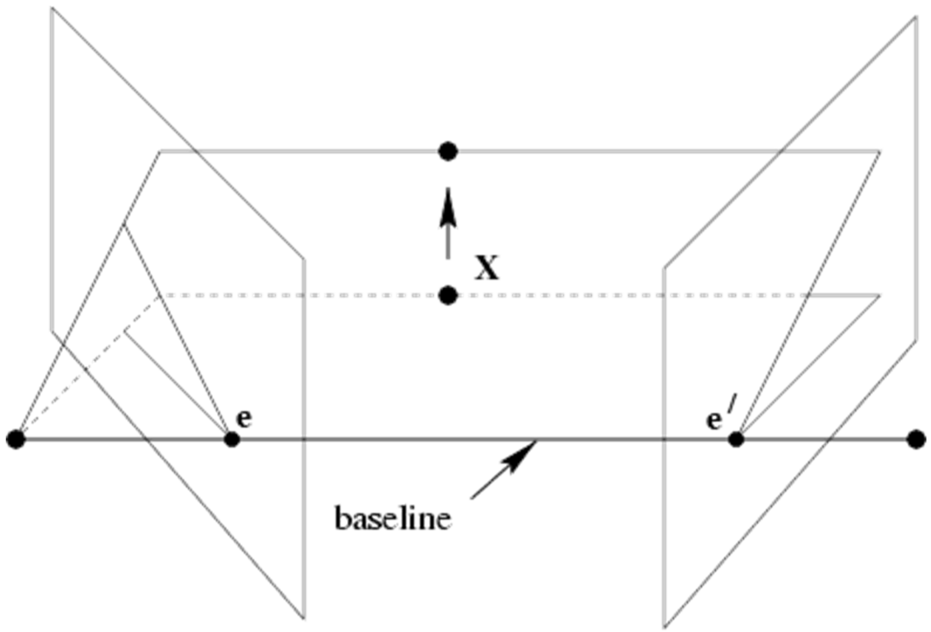
What if only C,C',x are known?

The epipolar geometry



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

The epipolar geometry



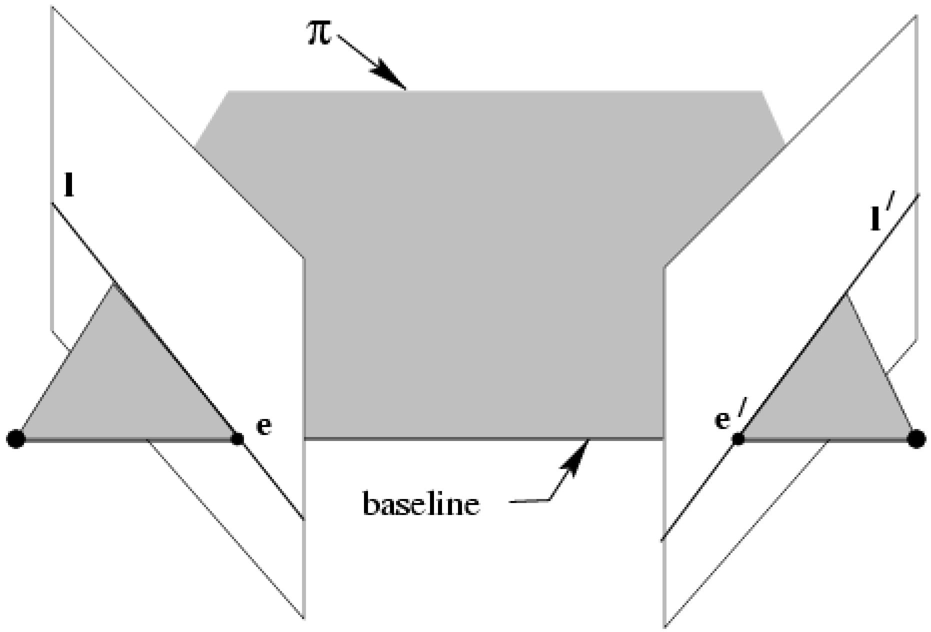
Family of planes π and lines l and l' intersect at e and e'

The epipolar geometry

epipolar pole [epipolar geometry demo]

= intersection of baseline with image plane

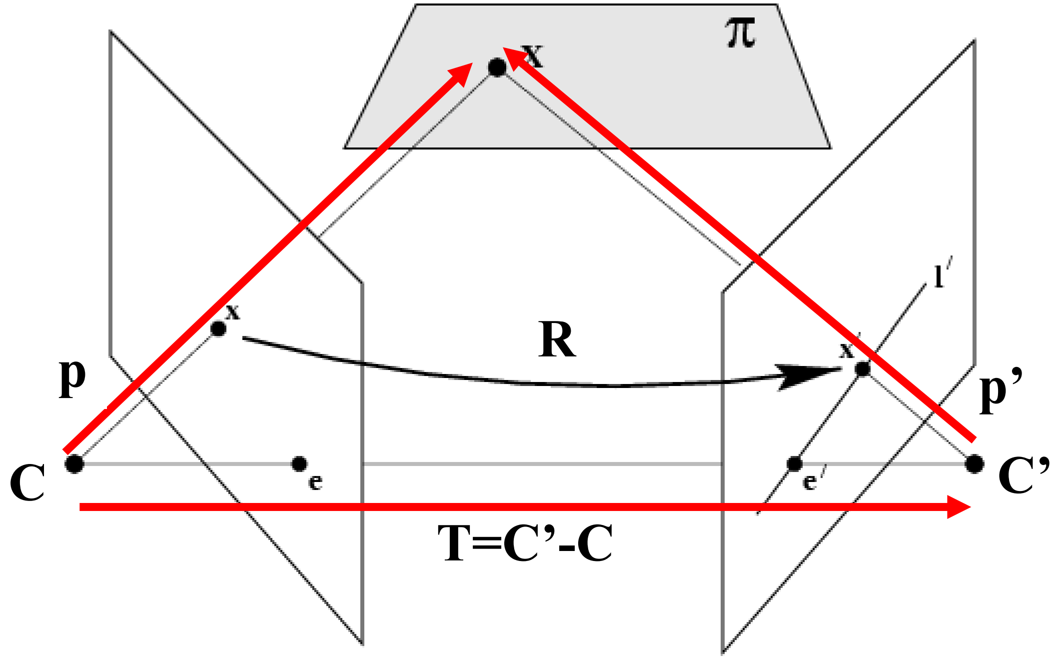
= projection of projection center in the other image



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

$$\mathbf{p} = \mathbf{R}\mathbf{p}' + \mathbf{T}$$

The fundamental matrix F

$$\mathbf{p} = \mathbf{R}\mathbf{p}' + \mathbf{T}$$

Multiply both sides by $\mathbf{p}^{\mathrm{T}}[\mathbf{T}]_{\times}$

$$[\mathbf{T}]_{\times} = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$

$$\mathbf{p}^{\mathrm{T}}[\mathbf{T}]_{\times}\mathbf{p} = \mathbf{p}^{\mathrm{T}}[\mathbf{T}]_{\times}(\mathbf{R}\mathbf{p}' + \mathbf{T})$$

$$\Rightarrow \quad 0 = \mathbf{p}^{\mathrm{T}}\boxed{[\mathbf{T}]_{\times}\mathbf{R}}\mathbf{p}'$$

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

essential matrix

The fundamental matrix F

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

Let $\mathbf{M}$ and $\mathbf{M}'$ be the intrinsic matrices, then

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

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

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

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

fundamental matrix

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↔x’ in the two images

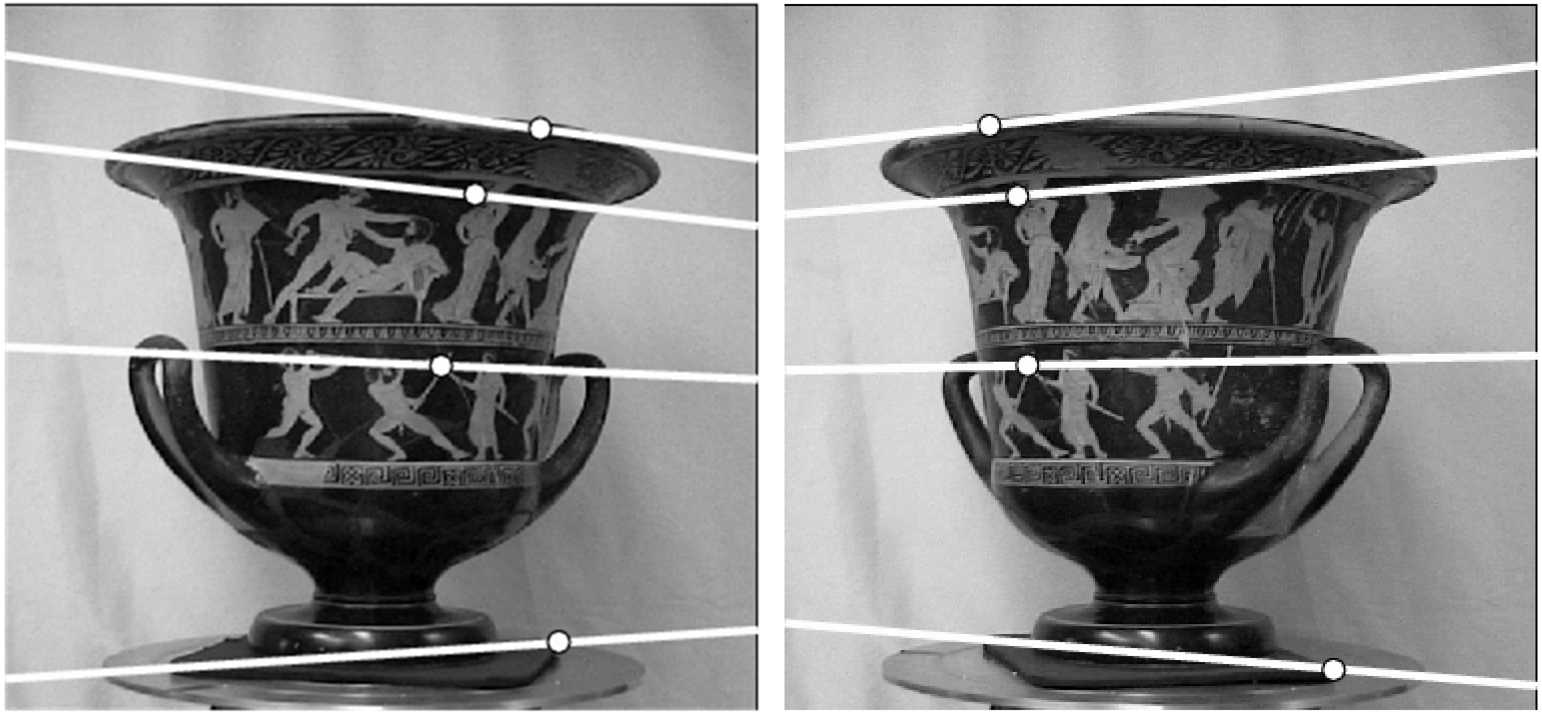
$$\mathrm{x}^{\mathrm{T}}\mathrm{F}\mathrm{x}' = 0 \qquad \left(\mathrm{x}^{\mathrm{T}}\mathrm{l} = 0\right)$$

The fundamental matrix F

F is the unique 3x3 rank 2 matrix that satisfies $x^TFx'=0$ for all $x \leftrightarrow x'$

1. **Transpose:** if F is fundamental matrix for (x,x'), then F^T is fundamental matrix for (x',x)
2. **Epipolar lines:** $l=Fx'$ & $l'=F^Tx$
3. **Epipoles:** on all epipolar lines, thus $e^TFx'=0, \forall x'$ $\Rightarrow e^TF=0$, 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 $\mathbf{x}=(u,v,1)^{\mathrm{T}}$ and $\mathbf{x}'=(u',v',1)^{\mathrm{T}}$, $\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$

each match gives a linear equation

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

8-point algorithm

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

- In reality, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$ subj. $\|\mathbf{f}\| = 1$. Find the vector corresponding to the least singular value.

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 $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$, where

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

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

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

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

Normalized 8-point algorithm

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

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

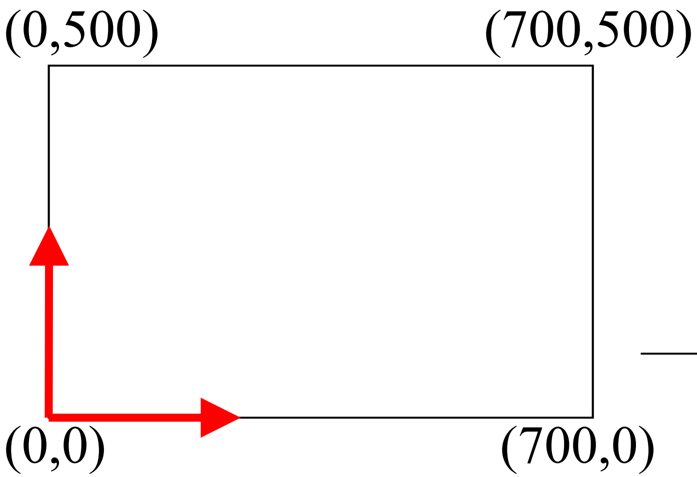
$$\boxed{\hat{\mathbf{x}}'^{\mathrm{T}}\mathbf{T}'^{-\mathrm{T}}}\,\mathbf{F}\,\boxed{\mathbf{T}^{-1}\hat{\mathbf{x}}} = 0$$

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

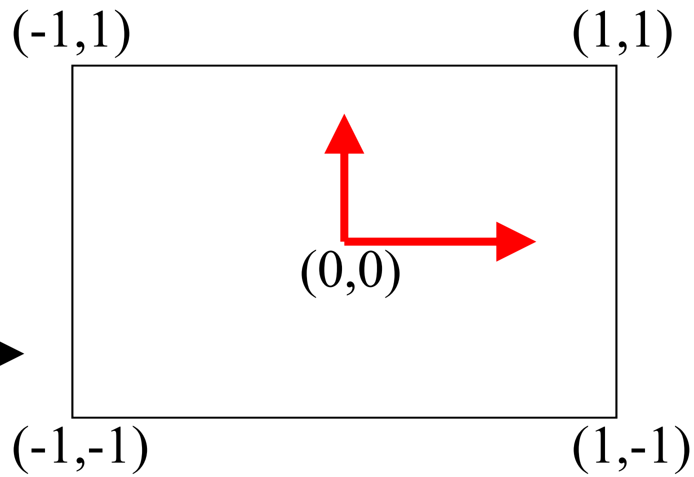
Normalized 8-point algorithm

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

```
[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

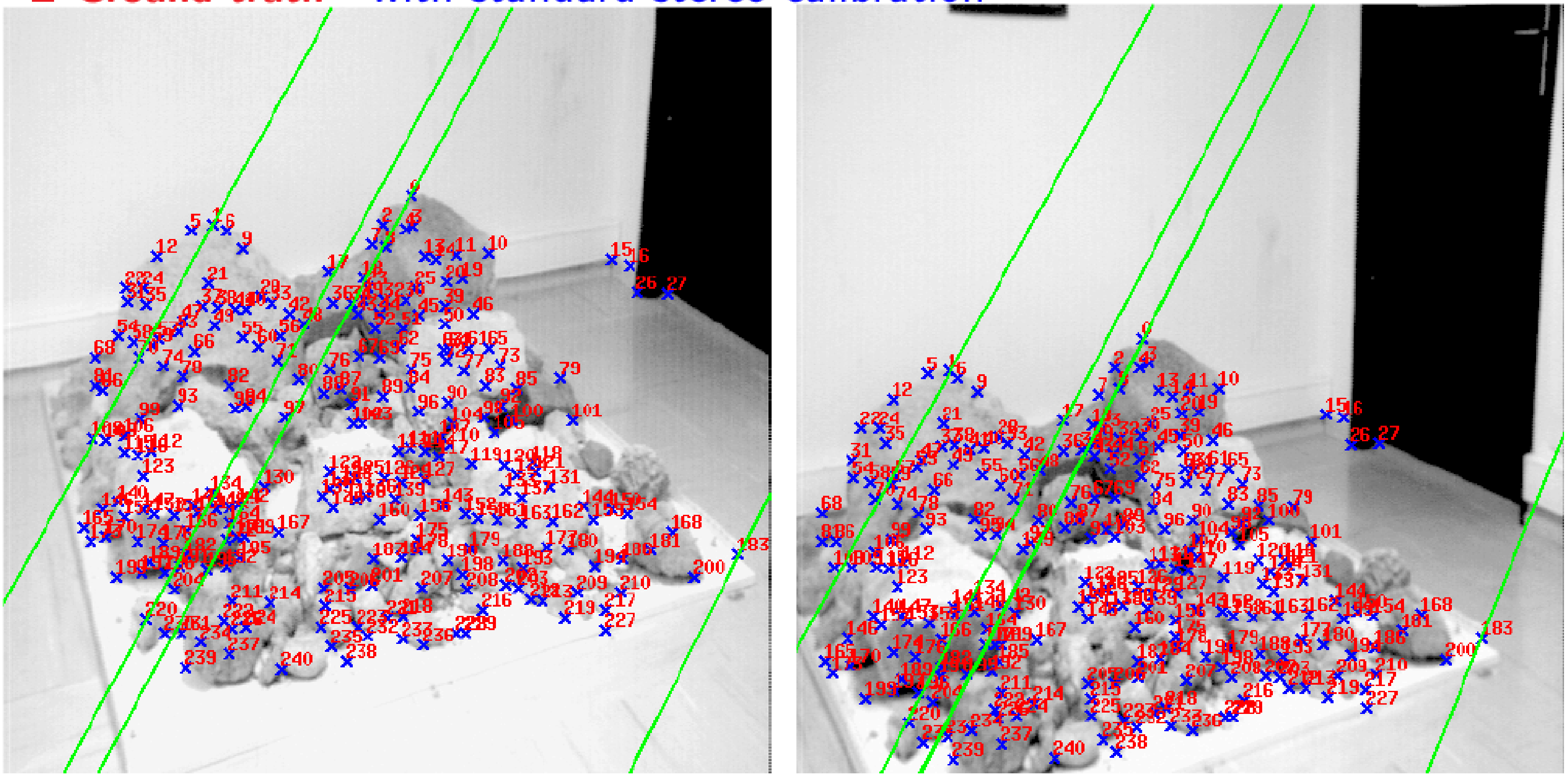
determine inliers

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

compute F based on all inliers

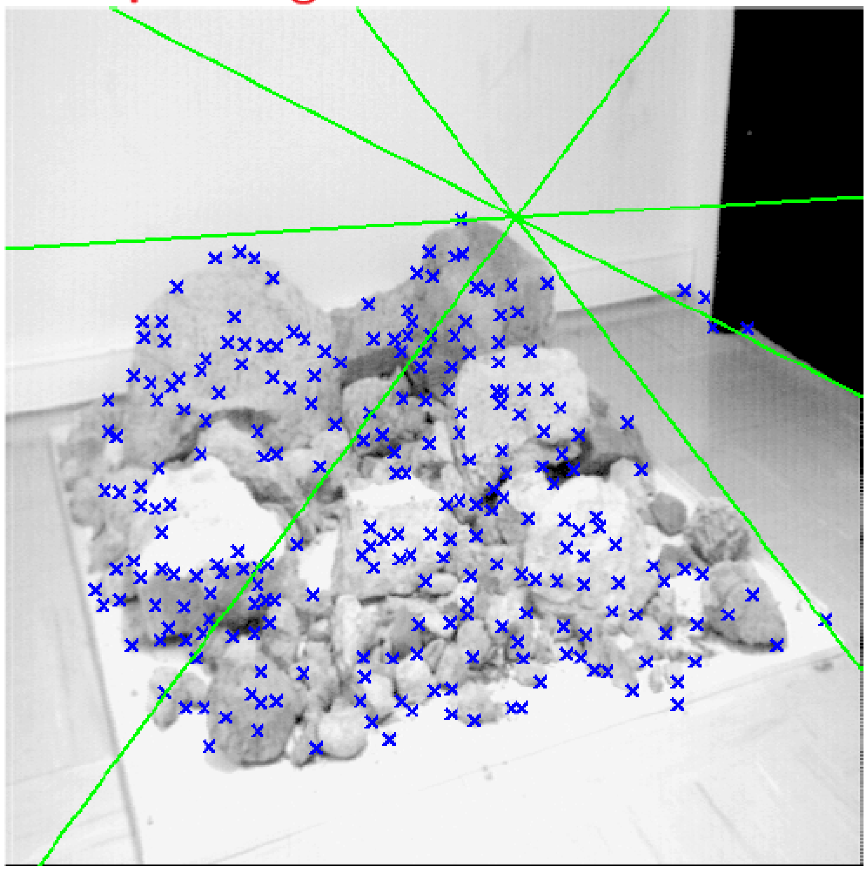
Results (ground truth)

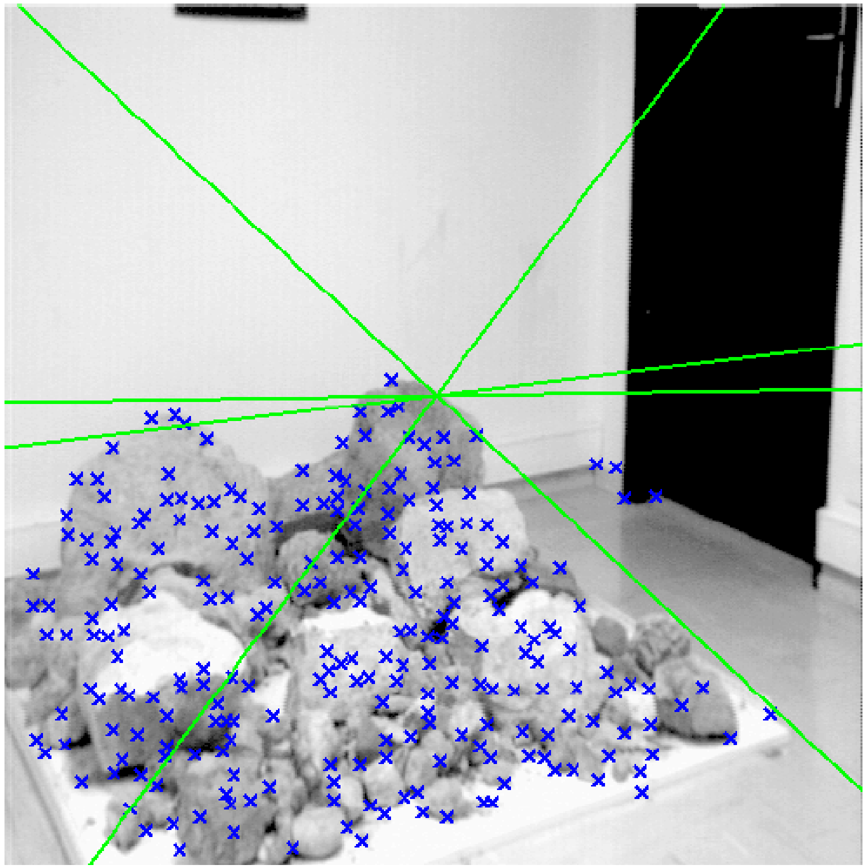
- Ground truth with standard stereo calibration



Results (8-point algorithm)

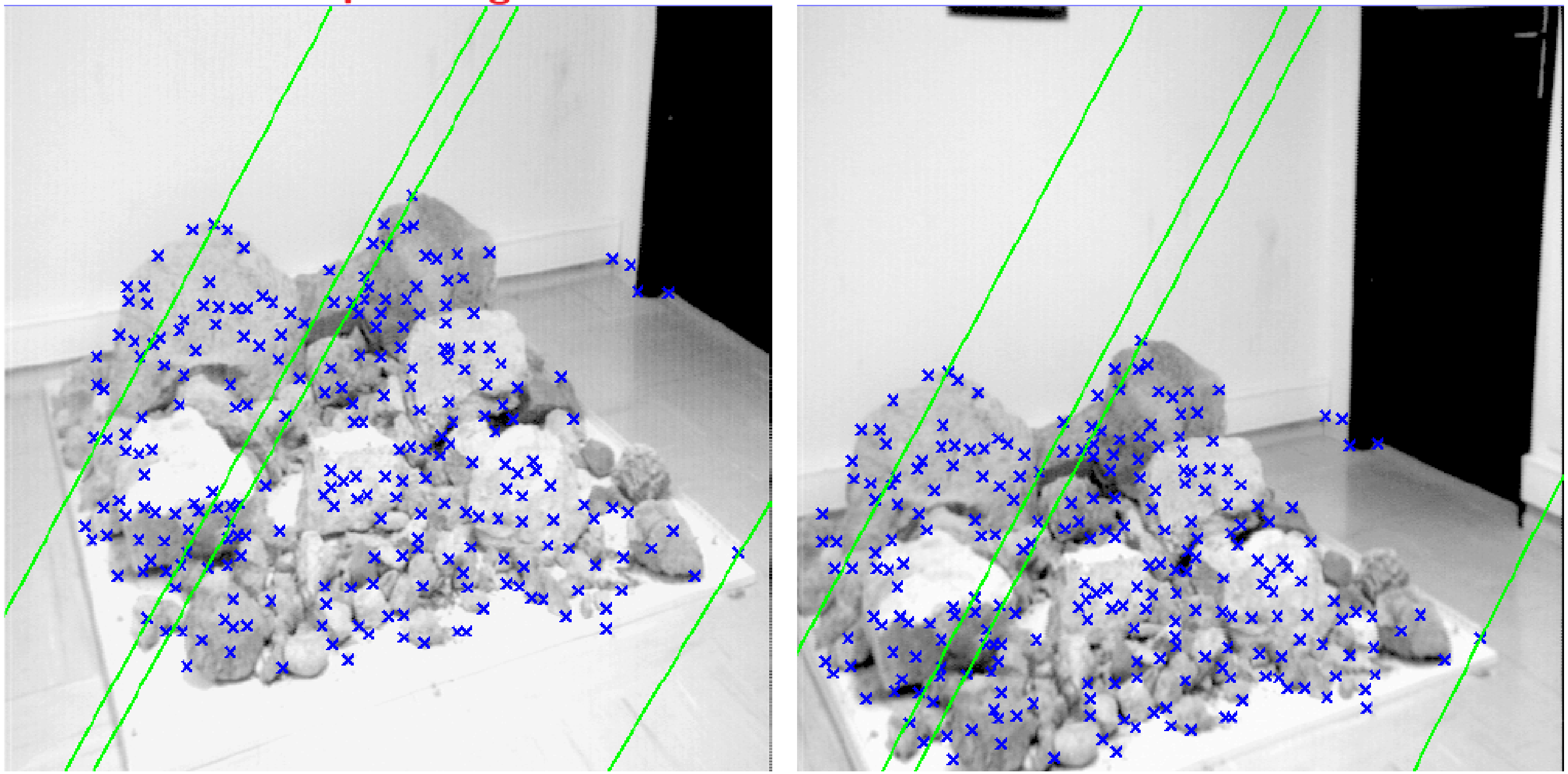
- 8-point algorithm





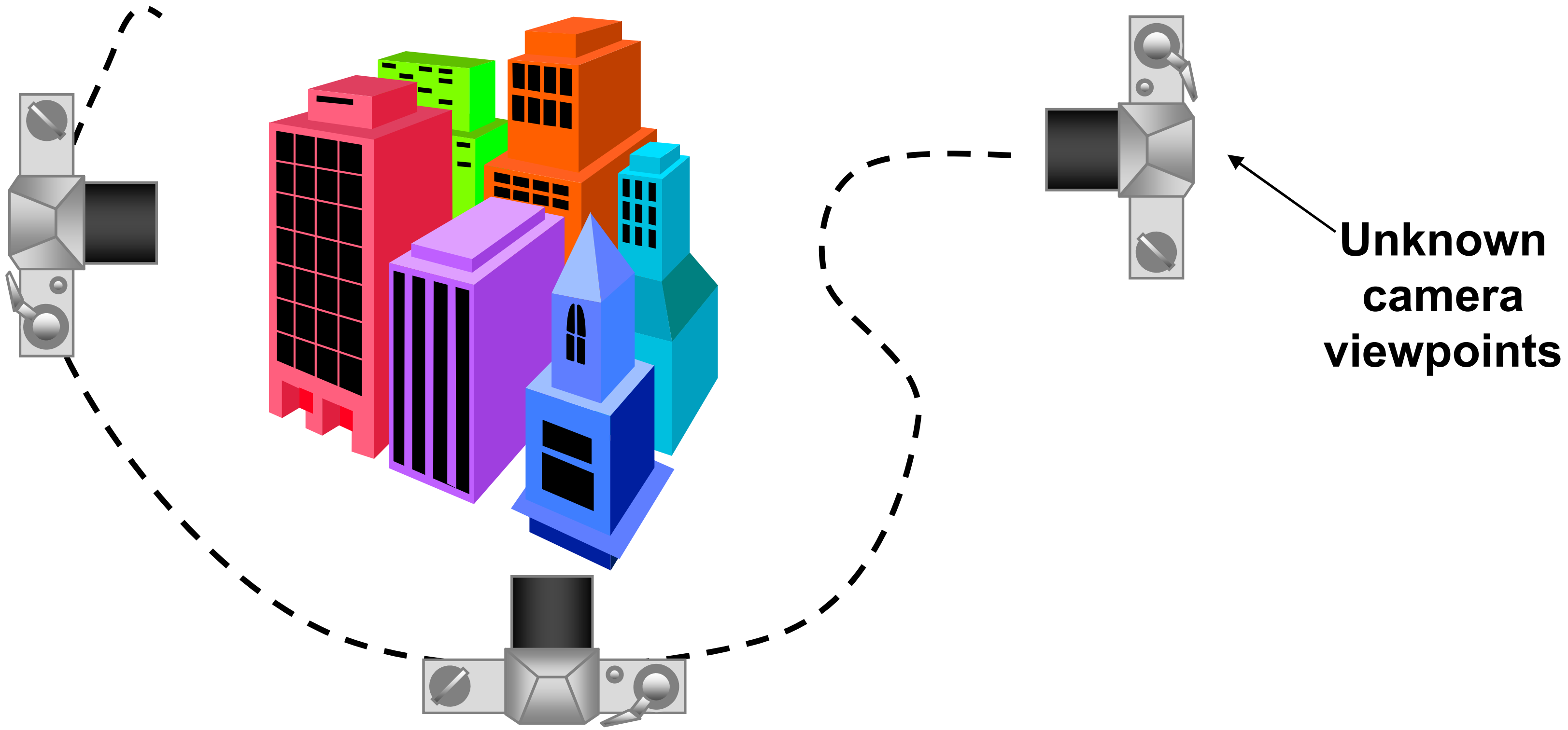
Results (normalized 8-point algorithm)

- Normalized 8-point algorithm



Structure from motion

Structure from motion

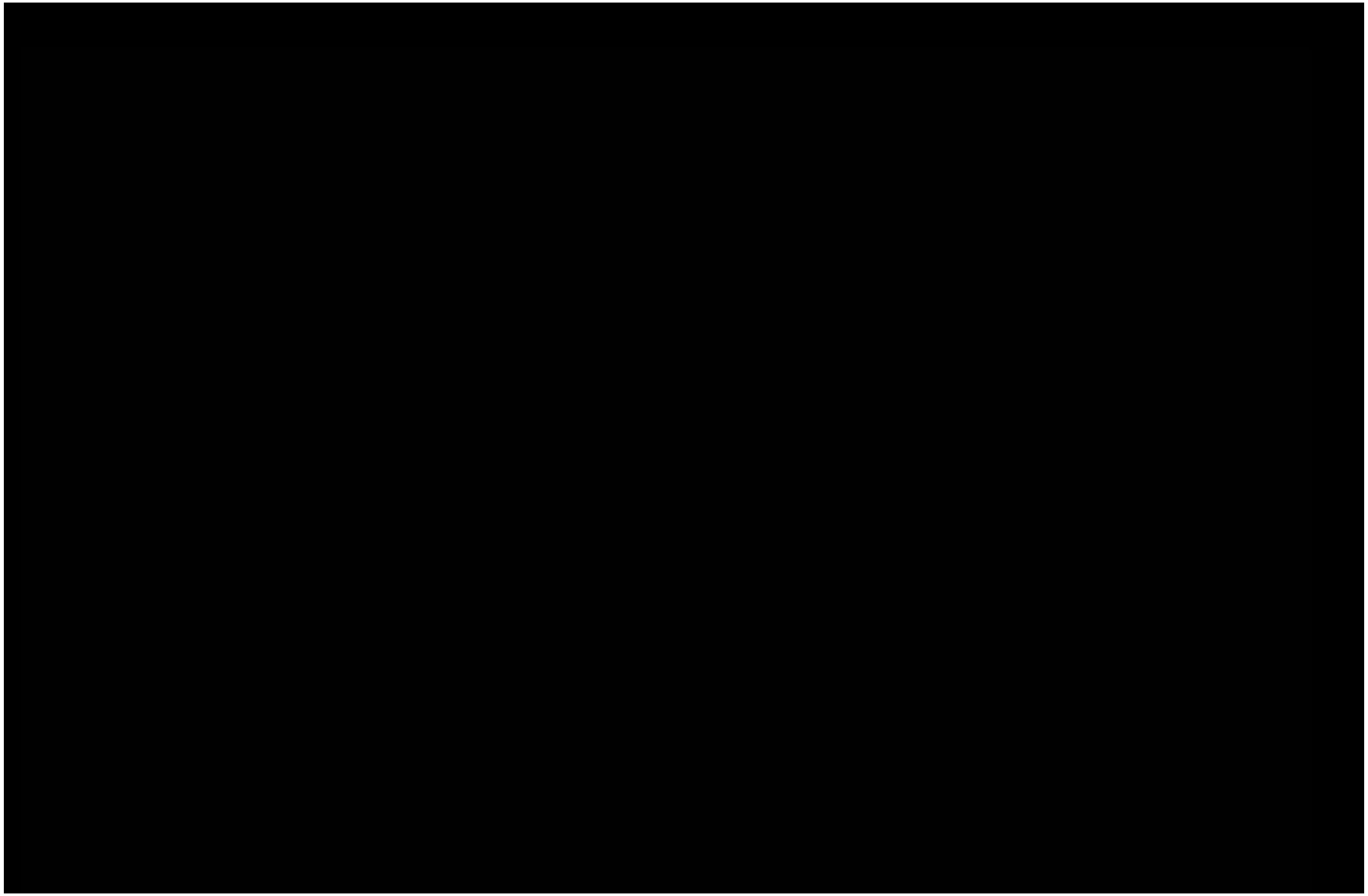


structure for motion: automatic recovery of **camera motion** and **scene structure** from two or more images. It is a self calibration technique and called *automatic camera tracking* or *matchmoving*.

Applications

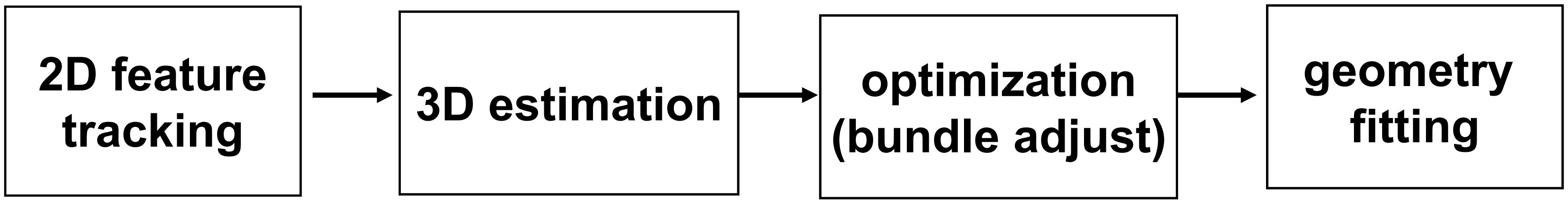
- For computer vision, multiple-view shape reconstruction, novel view synthesis and autonomous vehicle navigation.
- For film production, seamless insertion of CGI into live-action backgrounds

Matchmove



example #1 example #2 example #3

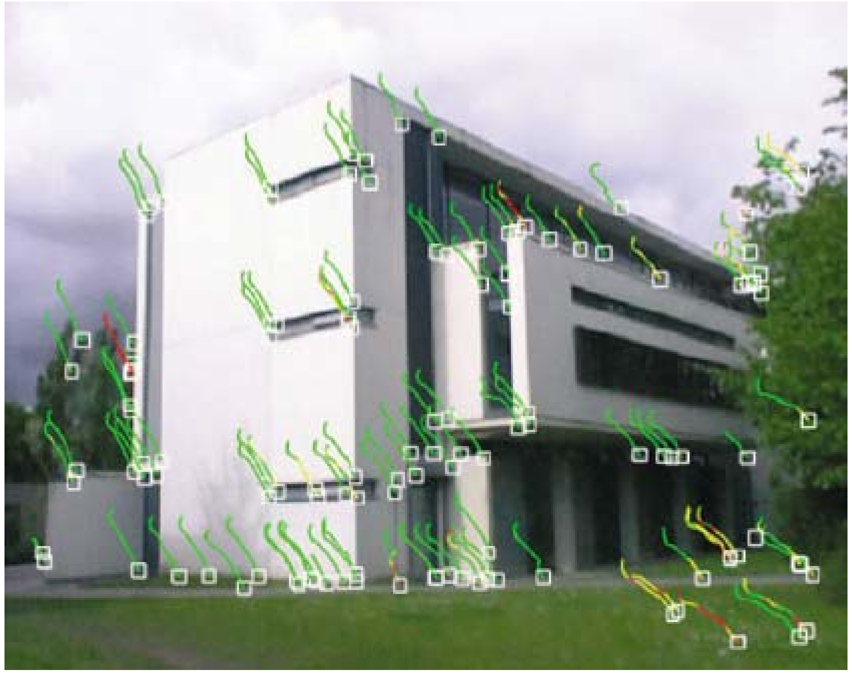
Structure from motion



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



KLT tracking

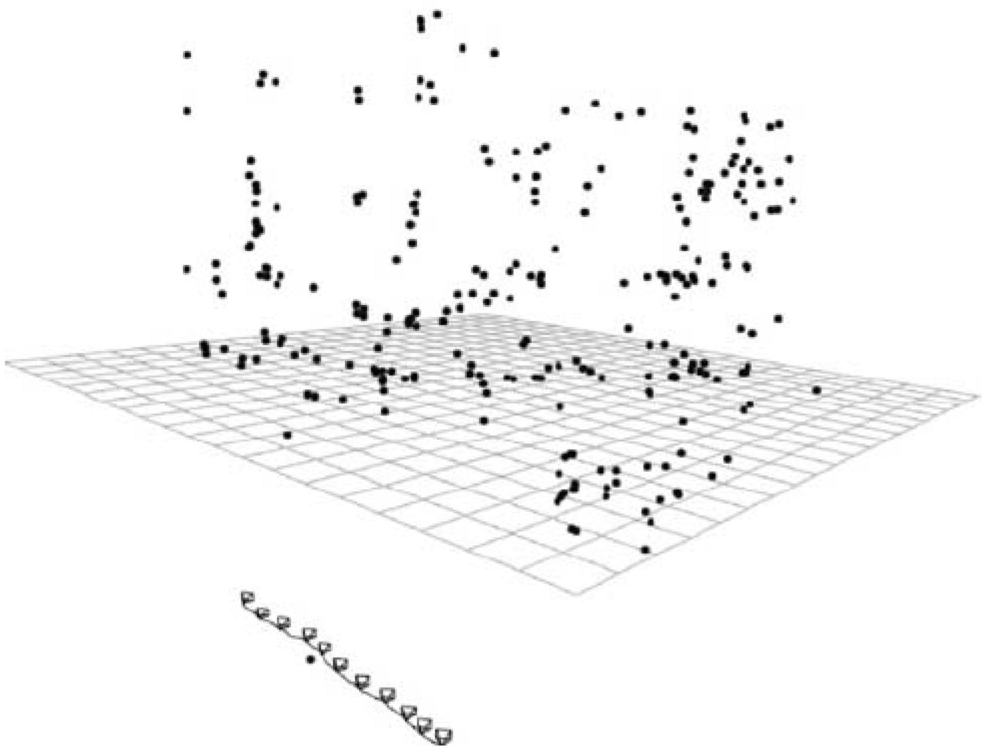
http://www.ces.clemson.edu/~stb/klt/

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

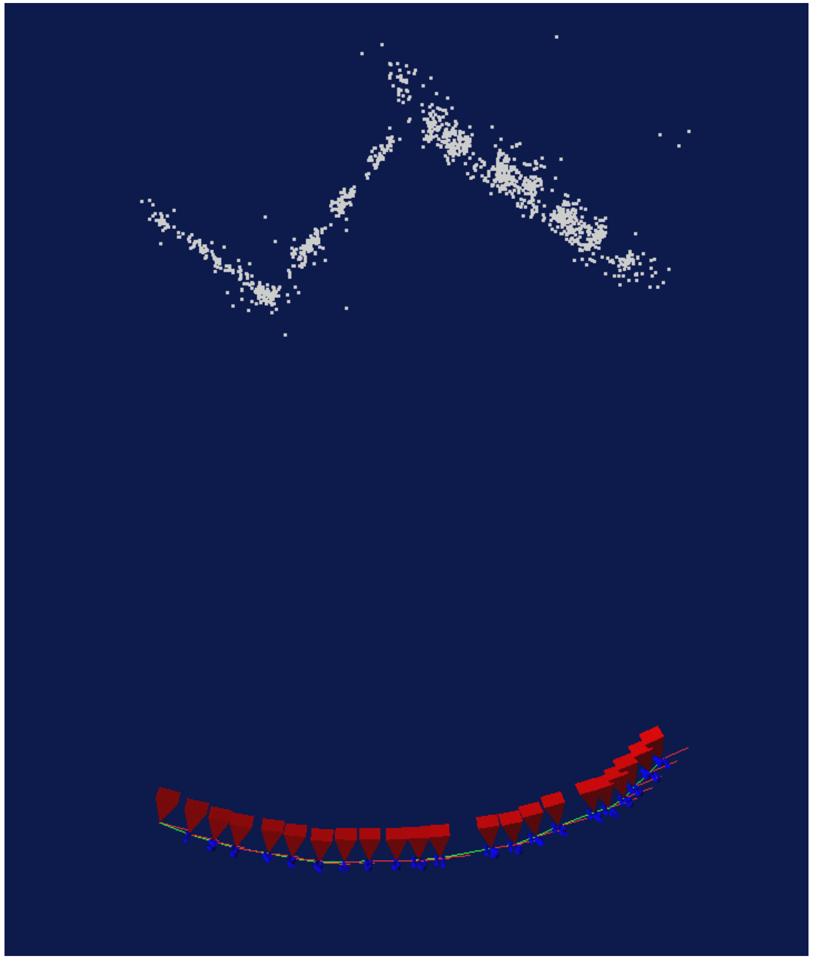


Structure from Motion

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

Factorization methods

Problem statement



Notations

- n 3D points are seen in m views
- **q**=*(u,v,1): 2D image point*
- **p**=*(x,y,z,1): 3D scene point*
- Π: projection matrix
- π: projection function
- q_{ij} is the projection of the i-th point on image j
- λ_{ij} projective depth of q_{ij}

$$\mathbf{q}_{ij} = \pi(\Pi_j \mathbf{p}_i)$$

$$\pi(x, y, z) = (x/z, y/z)$$

$$\lambda_{ij} = z$$

Structure from motion

- Estimate $\prod_j$ and $\mathbf{p}_i$ to minimize

$$\varepsilon(\mathbf{\Pi}_1,\cdots,\mathbf{\Pi}_m,\mathbf{p}_1,\cdots,\mathbf{p}_n)=\sum_{j=1}^{m}\sum_{i=1}^{n} w_{ij}\log P(\pi(\mathbf{\Pi}_j\mathbf{p}_i);\mathbf{q}_{ij})$$

$$w_{ij}=\begin{cases}1 & \text{if } p_i \text{ is visible in view j}\\ 0 & \text{otherwise}\end{cases}$$

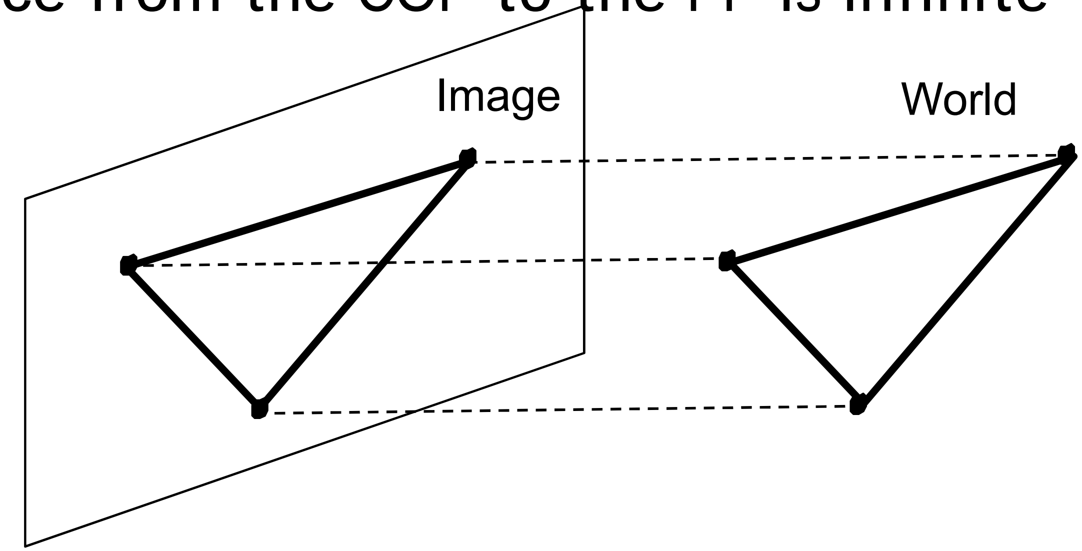
- Assume isotropic Gaussian noise, it is reduced to

$$\varepsilon(\mathbf{\Pi}_1,\cdots,\mathbf{\Pi}_m,\mathbf{p}_1,\cdots,\mathbf{p}_n)=\sum_{j=1}^{m}\sum_{i=1}^{n} w_{ij}\left\|\pi(\mathbf{\Pi}_j\mathbf{p}_i)-\mathbf{q}_{ij}\right\|^2$$

- Start from a simpler projection model

Orthographic projection

- Special case of perspective projection
  - Distance from the COP to the PP is infinite



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

  - Also called “parallel projection”: $(x, y, z) \rightarrow (x, y)$

SFM under orthographic projection

2D image point

Orthographic projection incorporating 3D rotation

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

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

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

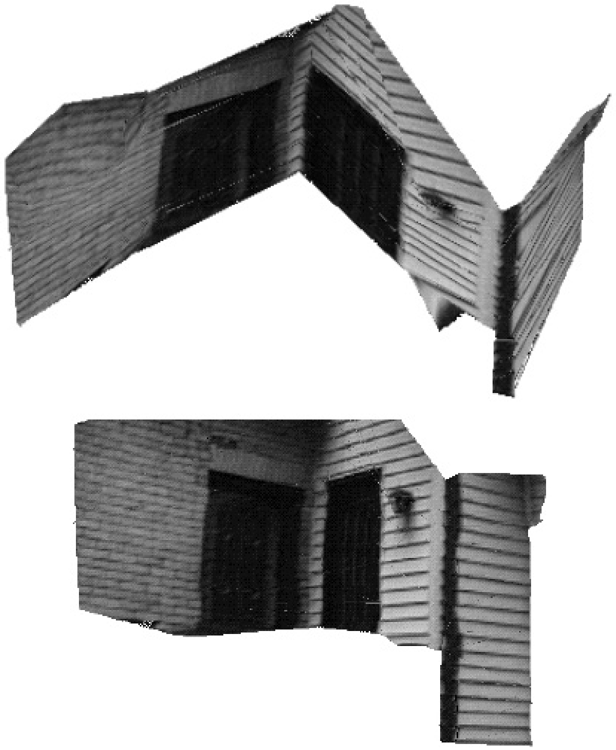
$$\underset{2m\times n}{\mathbf{W}} = \underset{2m\times 3}{\mathbf{M}'}\ \underset{3\times n}{\mathbf{S}'}$$

  - **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**

Results



Extensions to factorization methods

- Projective projection
- With missing data
- Projective projection with missing data

Bundle adjustment

Bundle adjustment

- n 3D points are seen in m views
- x_{ij} is the projection of the i-th point on image j
- a_j is the parameters for the j-th camera
- b_i is the parameters for the i-th point
- BA attempts to minimize the projection error

$$\min_{\mathbf{a}_j, \mathbf{b}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} d(\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i), \mathbf{x}_{ij})^2$$

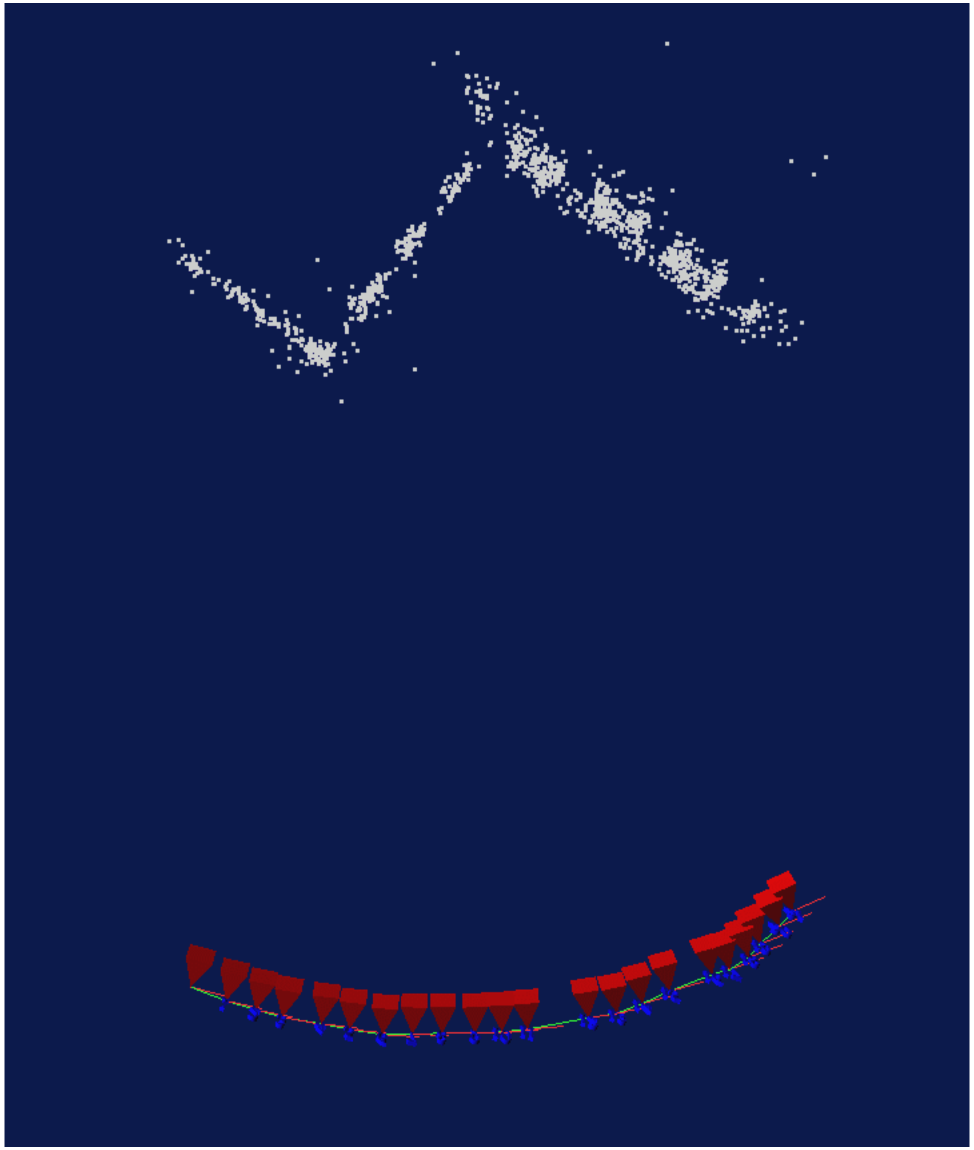
predicted projection

Euclidean distance

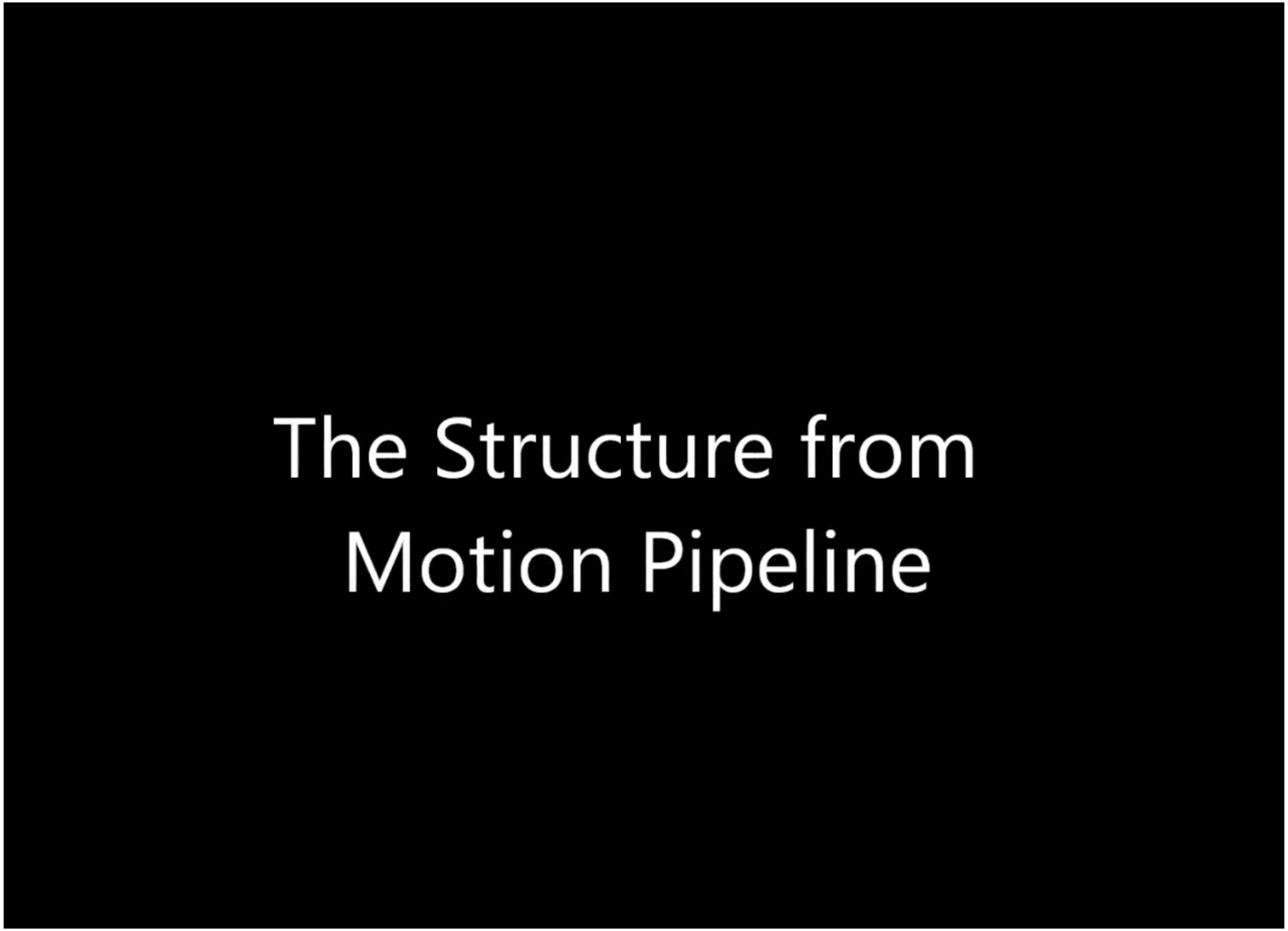
Levenberg-Marquardt method

- LM can be thought of as a combination of steepest descent and the Newton method. When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge. When the current solution is close to the correct solution, it becomes a Newton's method.

Bundle adjustment



MatchMove



Applications of matchmove

Jurassic park

2d3 boujou

Enemy at the Gate, Double Negative

2d3 boujou

Enemy at the Gate, Double Negative

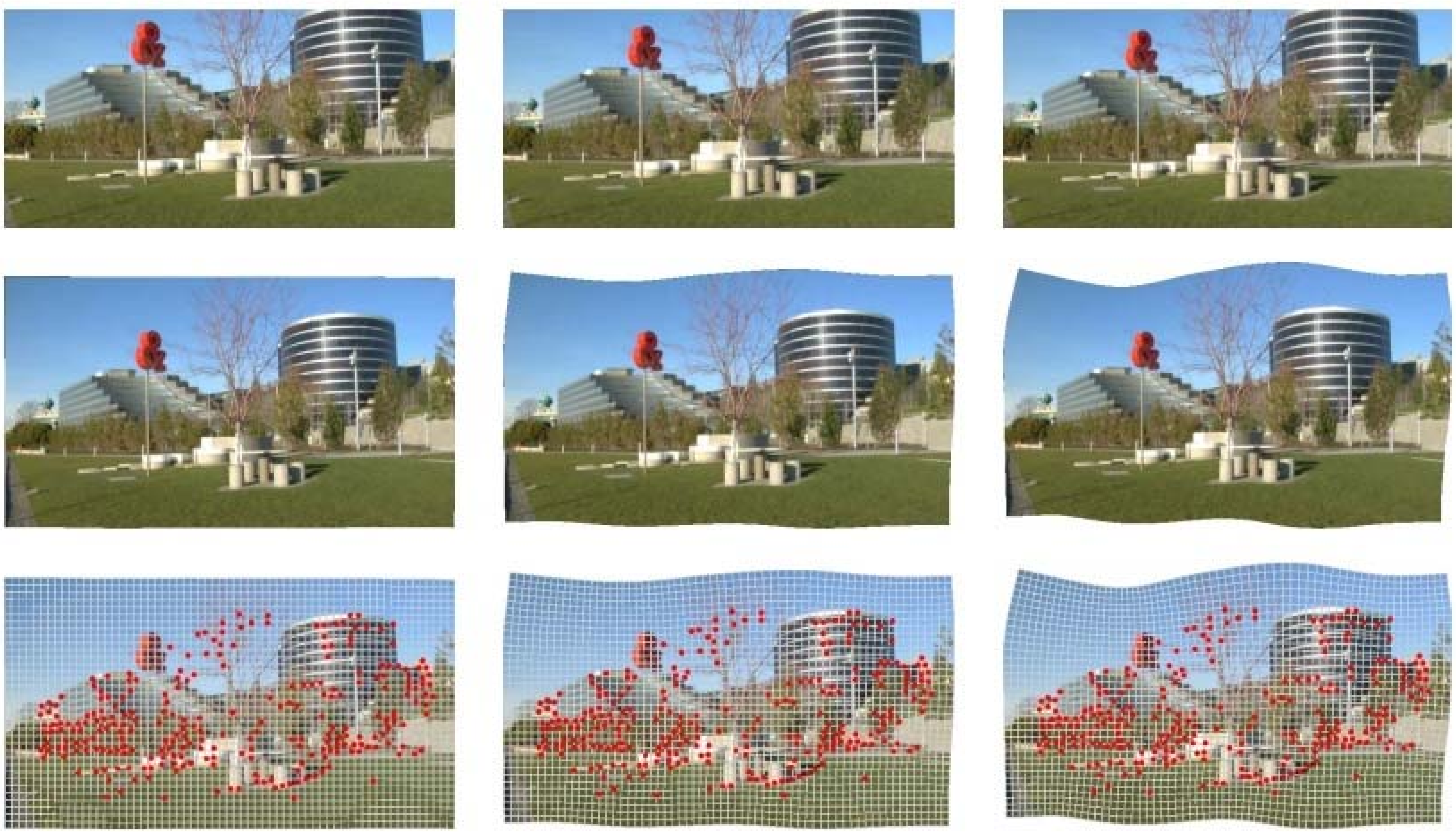
Photo Tourism

(a) (b) (c)

VideoTrace

http://www.acvt.com.au/research/videotrace/

Video stabilization



References

- Richard Hartley, In Defense of the 8-point Algorithm, ICCV, 1995.
- Carlo Tomasi and Takeo Kanade, Shape and Motion from Image Streams: A Factorization Method, Proceedings of Natl. Acad. Sci., 1993.
- Manolis Lourakis and Antonis Argyros, The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm, FORTH-ICS/TR-320 2004.
- N. Snavely, S. Seitz, R. Szeliski, Photo Tourism: Exploring Photo Collections in 3D, SIGGRAPH 2006.
- A. Hengel et. al., VideoTrace: Rapid Interactive Scene Modelling from Video, SIGGRAPH 2007.

Project #3 MatchMove

- It is more about using tools in this project
- You can choose either calibration or structure from motion to achieve the goal
- Calibration
- Voodoo/Icarus

- Examples from previous classes, #1, #2
- https://www.youtube.com/user/theActionMovieKid/videos