Structure from motion

Digital Visual Effects, Spring 2005

Yung-Yu Chuang

2005/4/20

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

Announcements

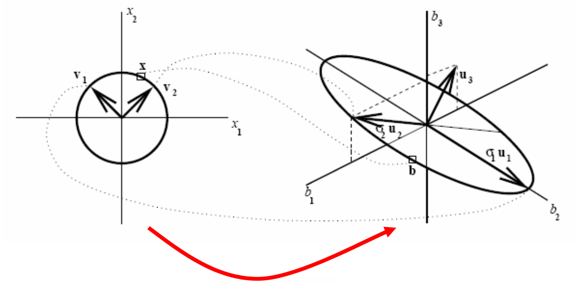
- Project #1 winning artifacts.
- I have a Canon G2 for project #2 artifacts.
- Project #2 is extended until next Tuesday. Basic requirements first, SIFT (you can refer Matlab implementation) and BA later.

Outline

- Singular value decomposition
- Epipolar geometry and fundamental matrix
- Structure from motion
- Applications
- Factorization methods
  - Orthogonal
  - Missing data
  - Projective
  - Projective with missing data

Singular value decomposition (SVD)

Every matrix represents a transformation



$$\mathbf{b} = A\mathbf{x}$$

$$A = \frac{1}{\sqrt{2}} \begin{bmatrix} \sqrt{3} & \sqrt{3} \\ -3 & 3 \\ 1 & 1 \end{bmatrix}$$

http://www.uwlax.edu/faculty/will/svd/index.html

Singular value decomposition

Theorem 3.2.1 *If A is a real $m \times n$ matrix then there exist orthogonal matrices*

$$U = [\, \mathbf{u}_1 \ \cdots \ \mathbf{u}_m \,] \in \mathcal{R}^{m \times m}$$
$$V = [\, \mathbf{v}_1 \ \cdots \ \mathbf{v}_n \,] \in \mathcal{R}^{n \times n}$$

such that

$$U^T A V = \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_p) \in \mathcal{R}^{m \times n}$$

where $p = \min(m, n)$ and $\sigma_1 \geq \ldots \geq \sigma_p \geq 0$. Equivalently,

$$A = U \Sigma V^T .$$

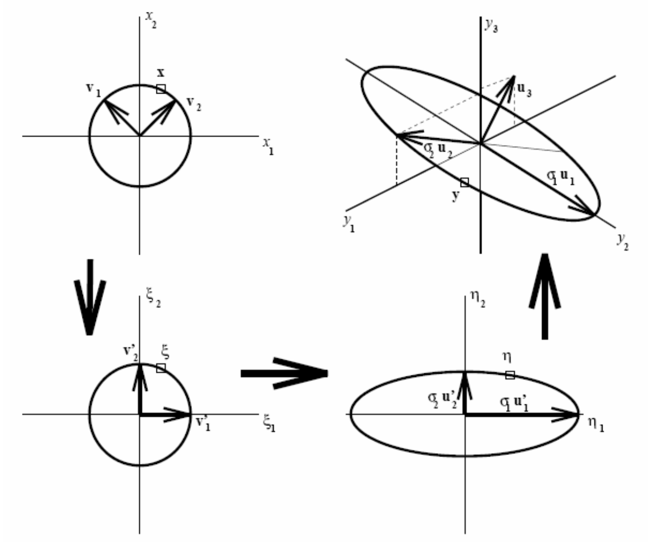
The SVD reveals a great deal about the structure of a matrix. If we define r by

$$\sigma_1 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = 0 ,$$

that is, if σ_r is the smallest nonzero singular value of A, then

$$\operatorname{rank}(A) = r$$

Singular value decomposition



Pseudoinverse

Theorem 3.3.1 *The minimum-norm least squares solution to a linear system $A\mathbf{x} = \mathbf{b}$, that is, the shortest vector $\mathbf{x}$ that achieves the*

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\| ,$$

is unique, and is given by

$$\hat{\mathbf{x}} = V \Sigma^{\dagger} U^T \mathbf{b}$$

where

$$\Sigma^{\dagger} = \begin{bmatrix} 1/\sigma_1 & & & & 0 & \cdots & 0 \\ & \ddots & & & & & \\ & & 1/\sigma_r & & \vdots & & \vdots \\ & & & 0 & & & \\ & & & & \ddots & & \\ & & & & 0 & 0 \cdots & 0 \end{bmatrix}$$

is an $n \times m$ diagonal matrix.

The matrix

$$A^{\dagger} = V \Sigma^{\dagger} U^T$$

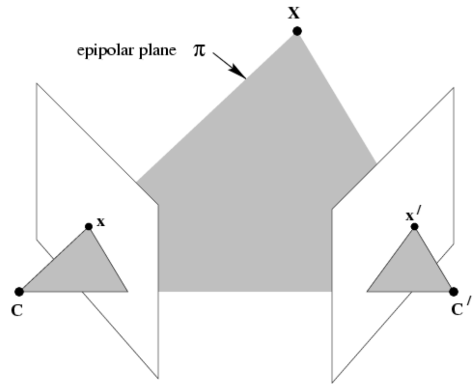
is called the *pseudoinverse* of A.

Epipolar geometry & fundamental matrix

The epipolar geometry

DigiVFX

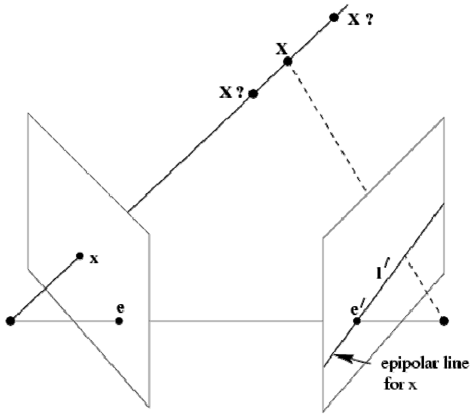
epipolar geometry demo



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

The epipolar geometry

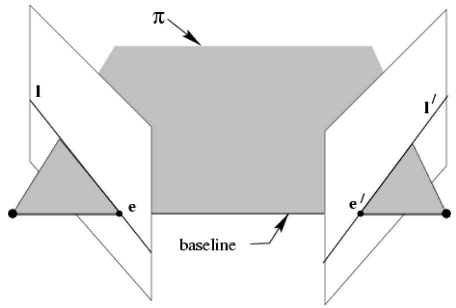
DigiVFX



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

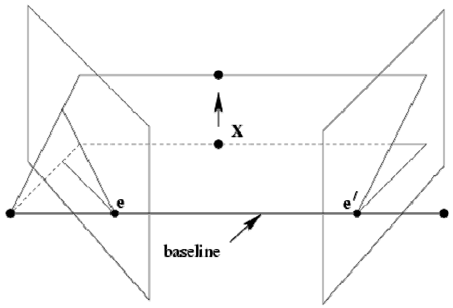
The epipolar geometry

DigiVFX



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'

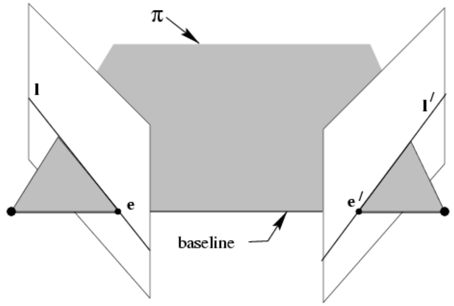
The epipolar geometry

epipolar pole

epipolar geometry demo

= intersection of baseline with image plane

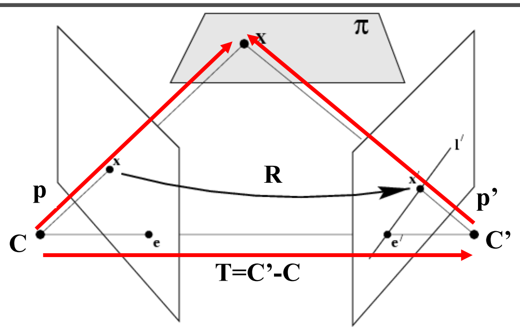
= projection of projection center in other image



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

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

The fundamental matrix F

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

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

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

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

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

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

essential matrix

The fundamental matrix F

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

Let M and M' be the intrinsic parameters, then

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

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

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

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

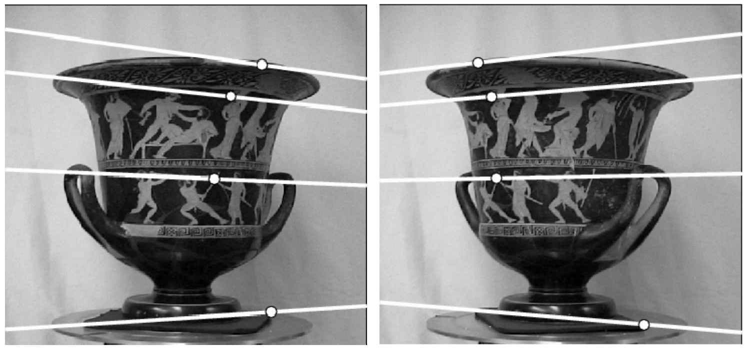
$$x'^{\mathrm{T}} F x = 0 \qquad \left(x'^{\mathrm{T}} l' = 0\right)$$

The fundamental matrix F

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

1. **Transpose:** if F is fundamental matrix for (P,P'), then F^{T} is fundamental matrix for (P',P)
2. **Epipolar lines:** $l'=Fx$ & $l=F^{\mathrm{T}}x'$
3. **Epipoles:** on all epipolar lines, thus $e'^{\mathrm{T}}Fx=0, \forall x$ $\Rightarrow e'^{\mathrm{T}}F=0$, 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 **f** to minimize $\|\mathbf{Af}\|$, 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 $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$, 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

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

~10000 ~10000 ~100 ~10000 ~10000 ~100 ~100 ~100 1

! Orders of magnitude difference between column of data matrix → least-squares yields poor results

Normalized 8-point algorithm

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

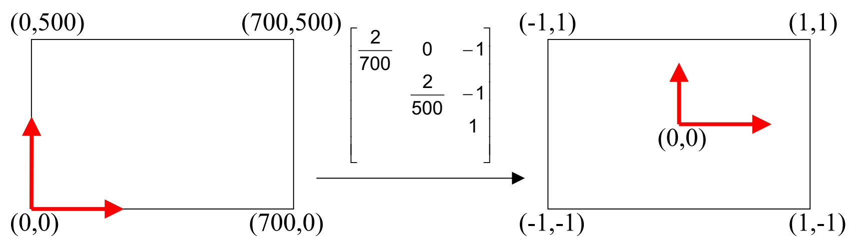
$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$$

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

Normalized 8-point algorithm

normalized least squares yields good results

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



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

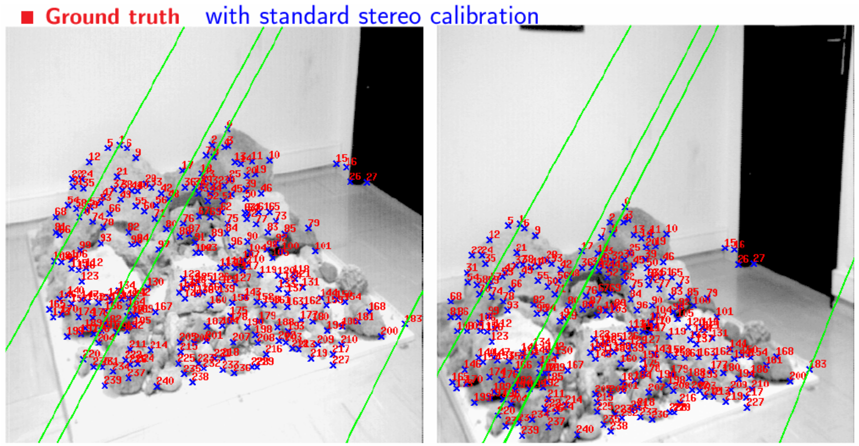
repeat

- select minimal sample (8 matches)
- compute solution(s) for F
- determine inliers

until Γ(*#inliers*,*#samples*)<95% || too many times

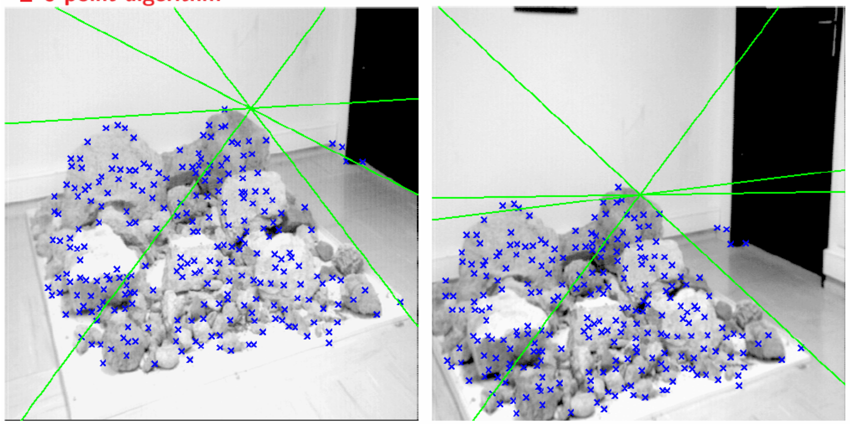
compute F based on all inliers

Results (ground truth)



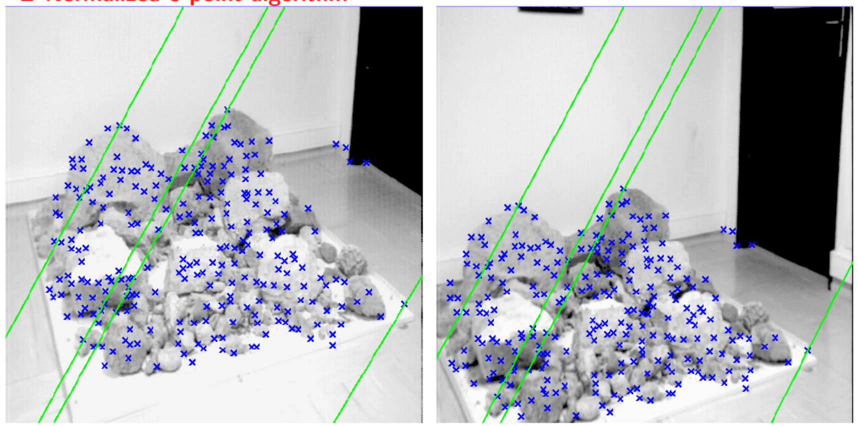
Results (8-point algorithm)

- 8-point algorithm



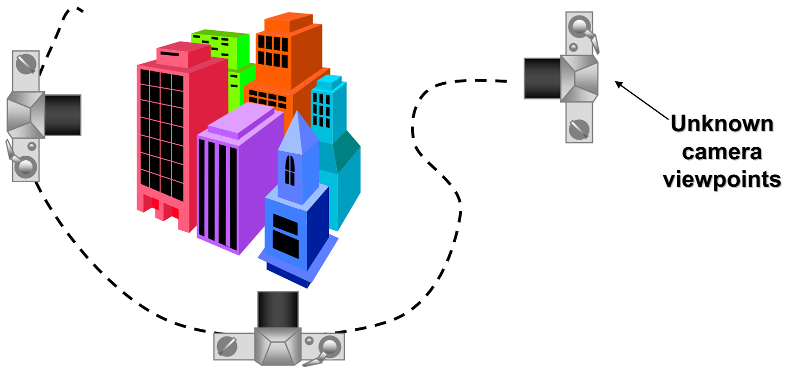
Results (normalized 8-point algorithm)

- 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

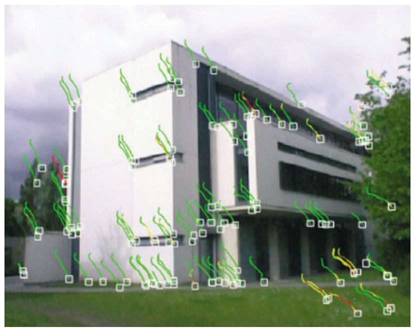
Structure from motion

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



KLT tracking

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

SIFT tracking (matching actually)

Frame 0 → Frame 10

SIFT tracking

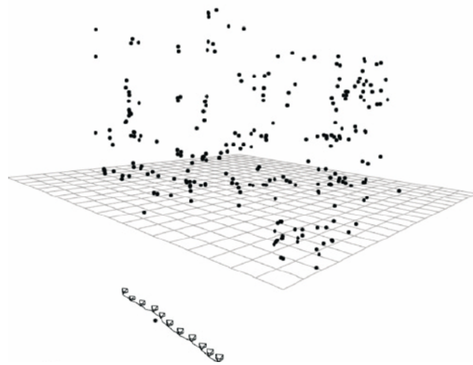
Frame 0 → Frame 200

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]**

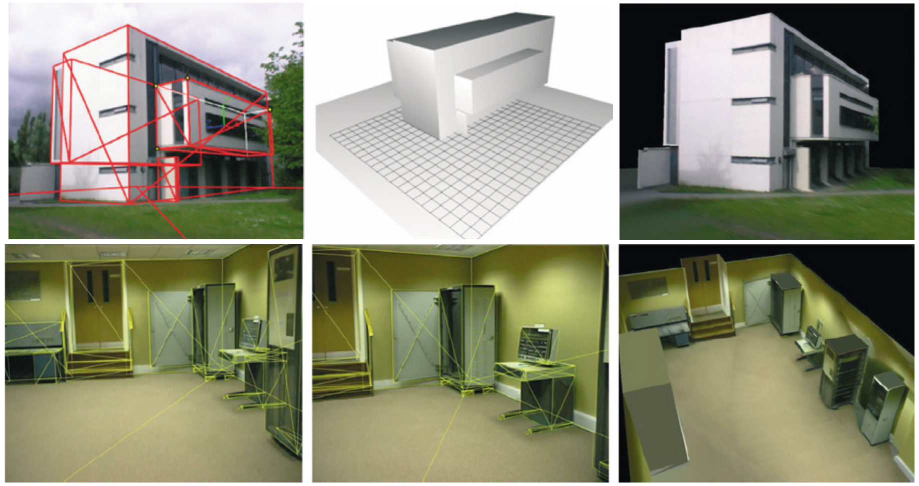
Structure from Motion

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



Structure from Motion

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



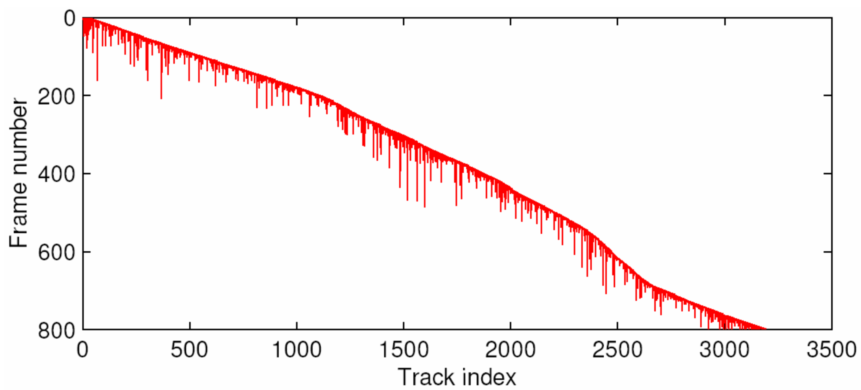
Issues in SFM

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

Track lifetime

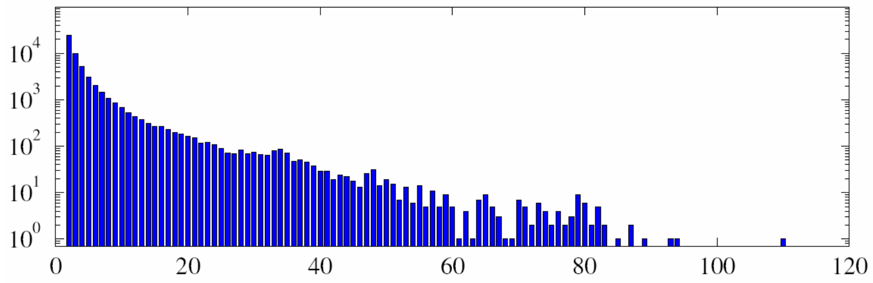
every 50th frame of a 800-frame sequence

Track lifetime



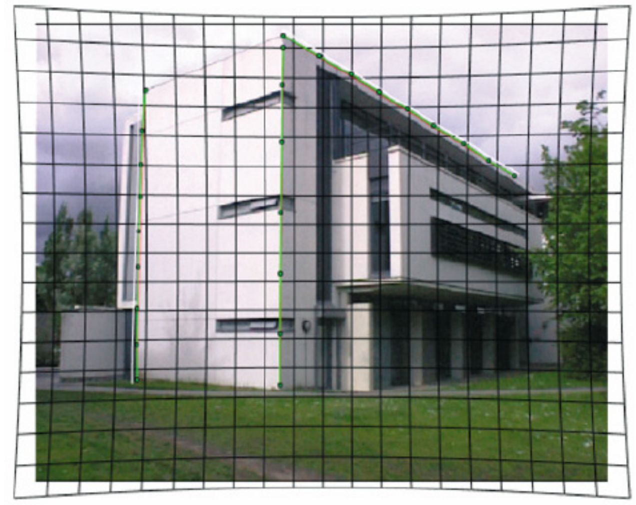
lifetime of 3192 tracks from the previous sequence

Track lifetime

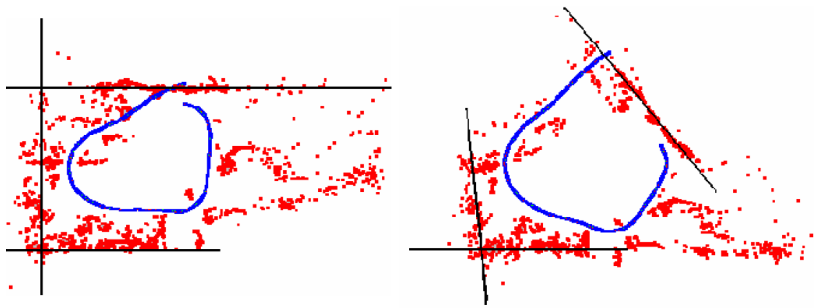


track length histogram

Nonlinear lens distortion



Nonlinear lens distortion

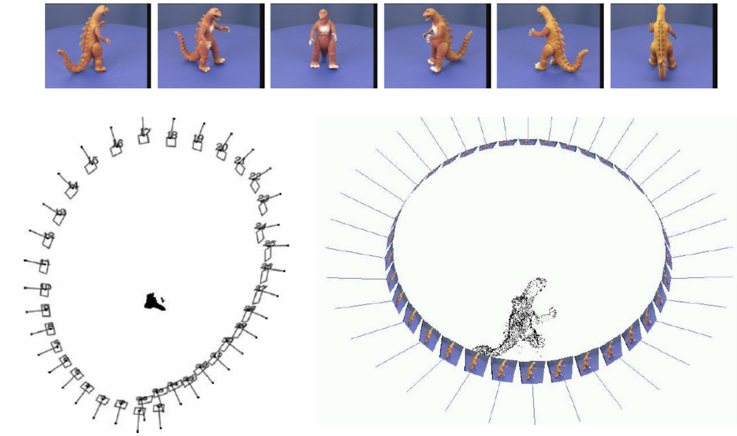


effect of lens distortion

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 matchmove

Applications of matchmove

2d3 boujou

Enemy at the Gate, Double Negative

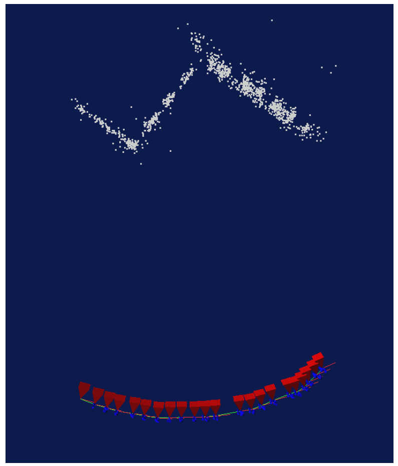
2d3 boujou

Enemy at the Gate, Double Negative

Jurassic park

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}

$$\mathbf{q}_{ij} = \pi(\Pi_j \mathbf{p}_i) \qquad \pi(x,y,z) = (x/z, y/z)$$

$$\lambda_{ij} = z$$

Structure from motion

- Estimate M_i and 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

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

SFM under orthographic projection

2D image point · orthographic projection matrix · 3D scene point · image offset

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

- 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:

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

factorization (Tomasi & Kanade)

projection of *n* features in one image:

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

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

Metric constraints

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

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

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

- Constraints are linear in $\mathbf{AA}^T$:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \Pi \Pi^T = \Pi' \mathbf{A} (\mathbf{A} \Pi')^T = \Pi' \mathbf{G} \Pi'^T \qquad \textit{where} \quad \mathbf{G} = \mathbf{AA}^T$$

- Solve for **G** first by writing equations for every Π_i in **M**
- Then $\mathbf{G} = \mathbf{AA}^T$ by SVD (since **U** = **V**)

Factorization with noisy data

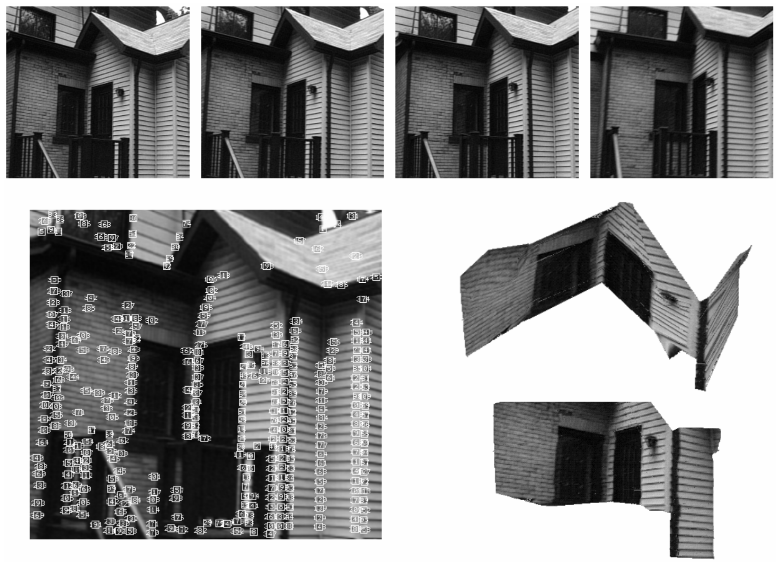
$$\underset{2m \times n}{\mathbf{W}} = \underset{2m \times 3}{\mathbf{M}} \; \underset{3 \times n}{\mathbf{S}} + \underset{2m \times n}{\mathbf{E}}$$

- SVD gives this solution
  - Provides optimal rank 3 approximation **W'** of **W**

$$\underset{2m \times n}{\mathbf{W}} = \underset{2m \times n}{\mathbf{W}'} + \underset{2m \times n}{\mathbf{E}}$$

- Approach
  - Estimate **W'**, then use noise-free factorization of **W'** as before
  - Result minimizes the SSD between positions of image features and projection of the reconstruction

Results



Extensions to factorization methods

DigiVFX

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

Reference

DigiVFX

- Carlo Tomasi, The Singular Value Decomposition, Mathematical Modeling of Continuous Systems course note, 2004.
- Richard Hartley, In Defense of the 8-point Algorithm, ICCV, 1995.
- Andrew W. Fitzgibbon and Andrew Zisserman, Automatic Camera Tracking, Video Registration, 2003.
- Carlo Tomasi and Takeo Kanade, Shape and Motion from Image Streams: A Factorization Method, Proceedings of Natl. Acad. Sci., 1993.