Structure from Motion

Andrew Fitzgibbon

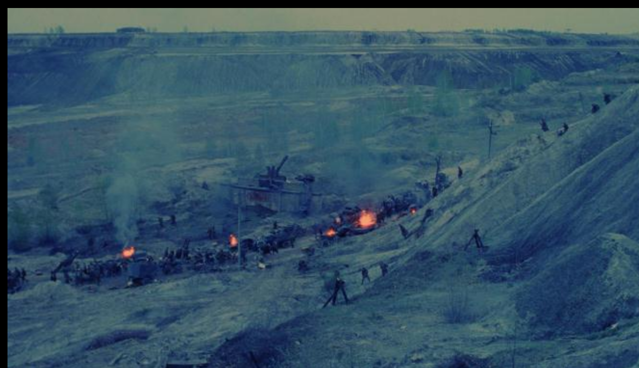
Microsoft Research

Computer graphics in the movies

Enemy at the Gates

Images courtesy of Mandalay Pictures, Pathé and Double Negative.

Computer graphics in the movies



Enemy at the Gates

Images courtesy of Mandalay Pictures, Pathé and Double Negative.

Computer graphics in the movies

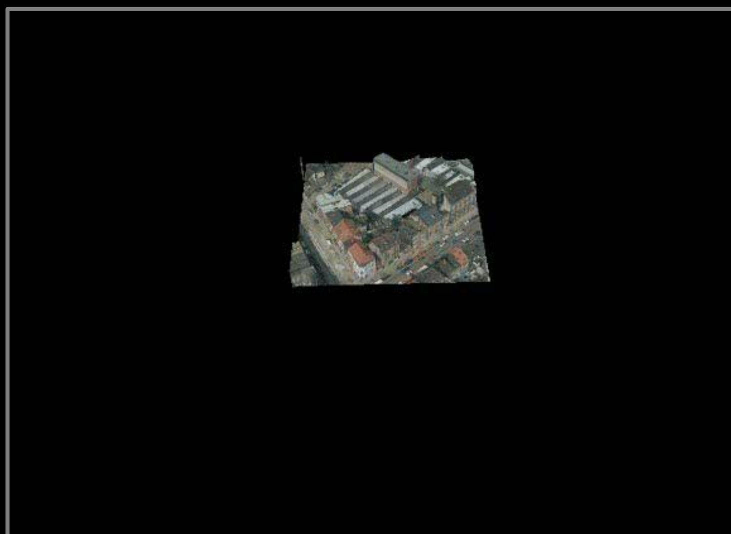
What was filmed…

Another example

Other “effects”: removing camera shake

3D Modelling from images

3D Modelling from images



A home-grown special effect

Boujou

- A computer program which makes inserting 3D objects easy
- Developed at Oxford University and at company "2d3"
- Now used in almost every movie
  - Lord of the Rings series
  - Harry Potter series
  - Bridget Jones's Diary

[demo]

Structure from motion: input data

If the sequence is m frames long then each *track* is represented as a stacked vector of 2D points:

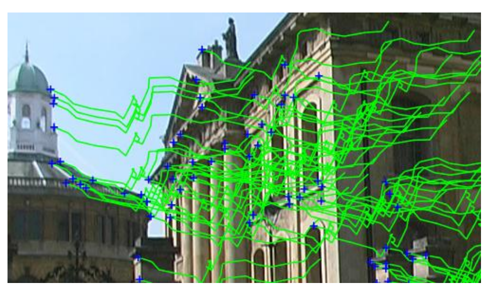
$$\underbrace{\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix}}_{2m \times 1}$$

Each 3D point generates a 2D *track*

If the sequence is m frames long then each *track* is represented as a stacked vector of 2D points:

$$\underbrace{\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix}}_{2m\times 1}$$

Many tracks: *measurements*



One track:

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix}$$

Concatenate n tracks, one per column:

$$\mathtt{M} = \begin{pmatrix} \mathbf{x}_{11} & \mathbf{x}_{11} & \cdots & \mathbf{x}_{1n} \\ \mathbf{x}_{21} & \mathbf{x}_{21} & \cdots & \mathbf{x}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{m1} & \mathbf{x}_{m1} & \cdots & \mathbf{x}_{mn} \end{pmatrix}$$

Measurement matrix

Structure from Motion: Problem statement

Given

- 2D point trajectories $\mathbf{x}$ in measurement matrix $\mathtt{M}$

Recover

- Structure: 3D point positions $\mathbf{X}$
- Motion: Camera projection matrices $\mathtt{P}$

All based on projection equation

$$\mathbf{x} = \mathtt{P}\mathbf{X}$$

Problem statement expanded

Given data: $\mathbf{x}$

Related by: $\mathbf{x} = \mathtt{P}\mathbf{X}$

Recover: $\mathtt{P}, \mathbf{X}$

Reminder: affine camera

Affine

$\tilde{\mathbf{w}} = \mathrm{P}_{aff}\tilde{\mathbf{X}}$. 8 degrees of freedom ($p_{34} = 1$). 4 points to calibrate.

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

1. Calibration

Reminder: Affine camera calibration

Given

3D points $\{\mathbf{X}_1, ..., \mathbf{X}_n\}$ and 2D points $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$

Compute $\mathtt{P}$ such that $\mathbf{x}_i = \mathbf{X}_i$ as closely as possible.

Reminder: Affine camera calibration

Given

3D points $\{\mathbf{X}_1, ..., \mathbf{X}_n\}$ and 2D points $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$

Compute $\mathtt{P}$ such that $\mathbf{x}_i = \mathbf{X}_i$ as closely as possible.

For $i = 1$

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{pmatrix} \begin{pmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{pmatrix}$$

2 equations, 8 unknowns $\implies$ no unique solution

Reminder: Affine camera calibration

Given

3D points $\{\mathbf{X}_1, ..., \mathbf{X}_n\}$ and 2D points $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$

Compute $\mathtt{P}$ such that $\mathbf{x}_i = \mathbf{X}_i$ as closely as possible.

For $i = 1..4$

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{pmatrix} \begin{pmatrix} X_1 & X_2 & X_3 & X_4 \\ Y_1 & Y_2 & Y_3 & Y_4 \\ Z_1 & Z_2 & Z_3 & Z_4 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$\mathtt{M} = \mathtt{P}\mathtt{B}^\top$$

$$\mathtt{P} = \mathtt{M}\mathtt{B}^{-\top}$$

8 equations, 8 unknowns $\implies$ unique solution (?)

Note $X^{-\top} = (X^{-1})^\top = (X^\top)^{-1}$

Reminder: Affine camera calibration

For $i = 1..n$

$$\begin{pmatrix} x_1 & x_2 & ... & x_n \\ y_1 & y_2 & ... & y_n \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \end{pmatrix} \begin{pmatrix} X_1 & X_2 & ... & X_n \\ Y_1 & Y_2 & ... & Y_n \\ Z_1 & Z_2 & ... & Z_n \\ 1 & 1 & ... & 1 \end{pmatrix}$$

$$\mathtt{M} = \mathtt{P}\mathtt{B}^\top$$

$$\mathtt{P} = \mathtt{M}(\mathtt{B}^*)^\top$$

$2n$ equations, 8 unknowns $\implies$ least-squares solution

$$X^* = (X^\top X)^{-1} X^\top$$

See that $M = PB^\top \implies MB = PB^\top B \implies MB(B^\top B)^{-1} = M\big((B^\top B)^{-\top} B^\top\big)^\top = M(B^*)^\top = P$, as $B^\top B$ symmetric.

2. Triangulation

2^{nd} reminder: affine triangulation

Given

2D points $\{\mathbf{x}_1, \mathbf{x}_2\}$,
projection matrices $\{\mathtt{P}_1, \mathtt{P}_2\}$.

Compute

$\mathbf{X}$ such that $\mathbf{x}_i = \mathtt{P}_i\mathbf{X}$ as closely as possible.

2nd reminder: affine triangulation

Given
2D points $\{\mathbf{x}_1, \mathbf{x}_2\}$,
projection matrices $\{\mathtt{P}_1, \mathtt{P}_2\}$.

Compute
$\mathbf{X}$ such that $\mathbf{x}_i = \mathtt{P}_i\mathbf{X}$ as closely as possible.

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{pmatrix} = \underbrace{\begin{pmatrix} \mathtt{P}_1 \\ \mathtt{P}_2 \end{pmatrix}}_{4\times 4} \mathbf{X}$$

$$\begin{aligned} \mathtt{M} &= \mathtt{A}\mathbf{X} \\ \Longrightarrow \quad \mathbf{X} &= \mathtt{A}^*\mathtt{M} \end{aligned}$$

Each 3D point generates a 2D *track*

If the sequence is m frames long then each *track* is represented as a stacked vector of 2D points:

$$\underbrace{\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix}}_{2m\times 1} = \begin{pmatrix} \mathtt{P}_1 \\ \mathtt{P}_2 \\ \vdots \\ \mathtt{P}_m \end{pmatrix} \mathbf{X}$$

2^{nd} reminder: affine triangulation

Given

2D points $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\}$,
projection matrices $\{\mathtt{P}_1, \mathtt{P}_2, ..., \mathtt{P}_m\}$.

Compute

$\mathbf{X}$ such that $\mathbf{x}_i = \mathtt{P}_i\mathbf{X}$ as closely as possible.

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} = \underbrace{\begin{pmatrix} \mathtt{P}_1 \\ \mathtt{P}_2 \\ \vdots \\ \mathtt{P}_m \end{pmatrix}}_{2m \times 4} \mathbf{X}$$

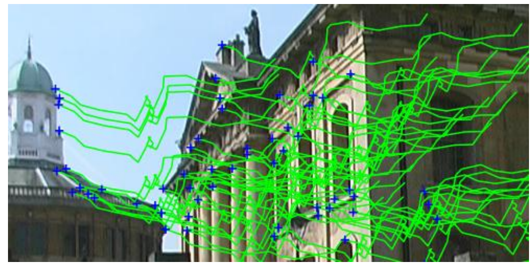
$$\mathtt{M} = \mathtt{A}\mathbf{X}$$
$$\implies \mathbf{X} = \mathtt{A}^*\mathtt{M}$$

3. Factorization

How do we get *both* **P** and **X**?

- One track:

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} = \begin{pmatrix} \mathtt{P}_1 \\ \mathtt{P}_2 \\ \vdots \\ \mathtt{P}_m \end{pmatrix} \mathbf{X}$$



- n tracks:

$$\begin{pmatrix} \mathbf{x}_{11} & \mathbf{x}_{11} & \cdots & \mathbf{x}_{1n} \\ \mathbf{x}_{21} & \mathbf{x}_{21} & \cdots & \mathbf{x}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{m1} & \mathbf{x}_{m1} & \cdots & \mathbf{x}_{mn} \end{pmatrix} = \begin{pmatrix} \mathtt{P}_1 \\ \mathtt{P}_2 \\ \vdots \\ \mathtt{P}_m \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{pmatrix}$$

The "factorization" problem

- n tracks:

$$\begin{pmatrix} \mathbf{x}_{11} & \mathbf{x}_{11} & \cdots & \mathbf{x}_{1n} \\ \mathbf{x}_{21} & \mathbf{x}_{21} & \cdots & \mathbf{x}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{m1} & \mathbf{x}_{m1} & \cdots & \mathbf{x}_{mn} \end{pmatrix} = \begin{pmatrix} \mathtt{P}_1 \\ \mathtt{P}_2 \\ \vdots \\ \mathtt{P}_m \end{pmatrix} \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{pmatrix}$$

$$M = PX$$

Given M, what are its factors P, X?

Factorization

- Factorization: $M = PX$
- Q. Suppose $M = 6$, what are P and X?
- Ambiguity:
  Factors P and X are not unique
  But they are restricted to a small set

Singular value decomposition (SVD)

- Any matrix M has a *singular value decomposition* of the form
  $$M = U\,S\,V^T$$
- For "simple" matrices U, S, and V:
  - S is diagonal
  - U is orthonormal: $U\,U^T = I$
  - V is orthonormal: V $V^T = I$

- Given SVD of M, i.e. M = U D V'
- And rank(M) = 4 (why?)
- Simply set
  - P = First 4 columns of U D
  - X = First 4 rows of V

This is one solution – could there be a better one?

Ambiguity (Perspective, not affine)

Ambiguity

How unique is a factorization?

Given matrix $\mathtt{M}$, for which we have already found factors so that

$$\mathtt{M} = \mathtt{A}\mathtt{B}^\top$$

we can generate new factors $\tilde{\mathtt{A}}$ and $\tilde{\mathtt{B}}$ as follows.

For any 4×4 invertible matrix $\mathtt{H}$, set

$$\tilde{\mathtt{A}} = \mathtt{A}\mathtt{H}$$
$$\tilde{\mathtt{B}} = \mathtt{B}\mathtt{H}^{-\top}$$

Then $\tilde{\mathtt{A}}\tilde{\mathtt{B}}^\top = \mathtt{A}\mathtt{H}(\mathtt{B}\mathtt{H}^{-\top})^\top = \mathtt{A}\mathtt{H}\mathtt{H}^{-1}\mathtt{B}^\top = \mathtt{A}\mathtt{B}^\top = \mathtt{M}$

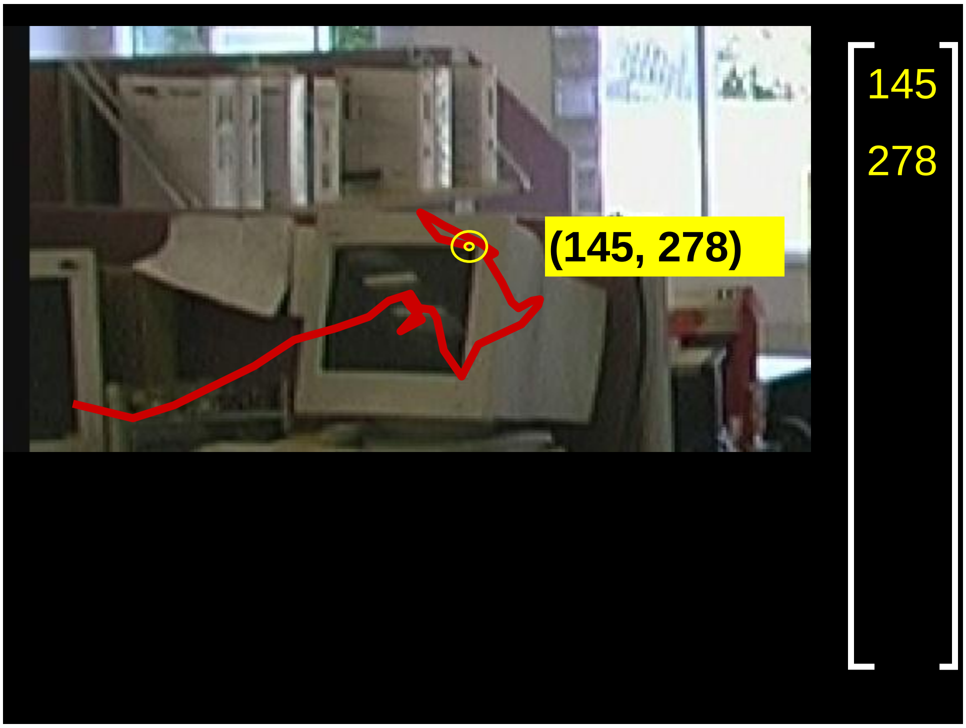
But…

- Factorization uses affine camera model, the real world is **perspective**
- Factorization computes minimum of error

  e(P,X) = norm(M – P X)

  Would like to optimize **weighted error**

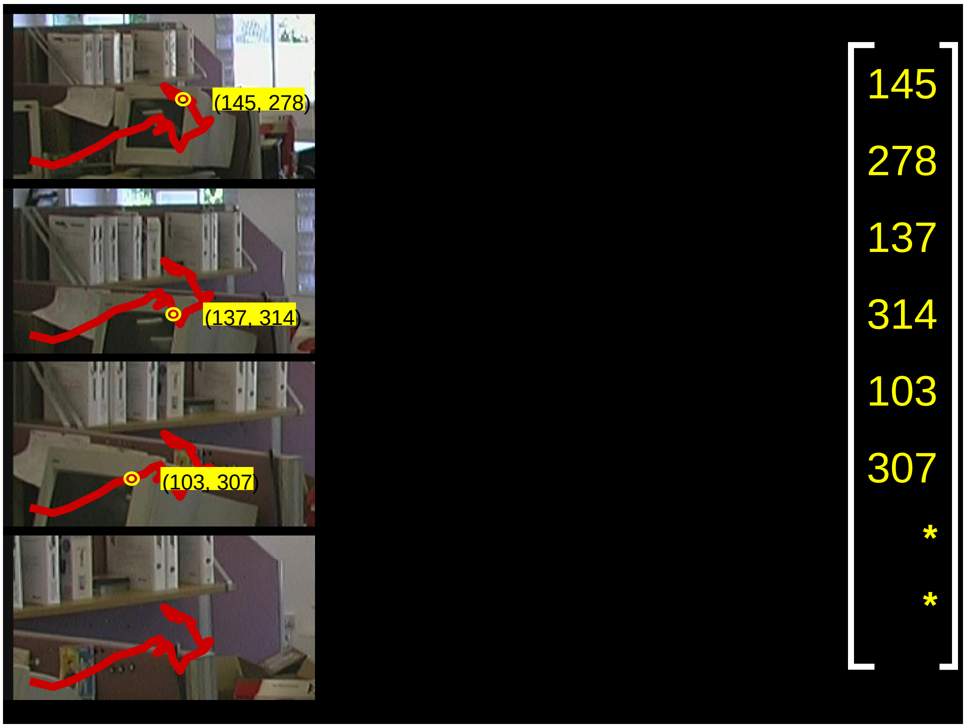
  e(P,X) = norm(W * (M – P X))
- Must deal with **missing data**

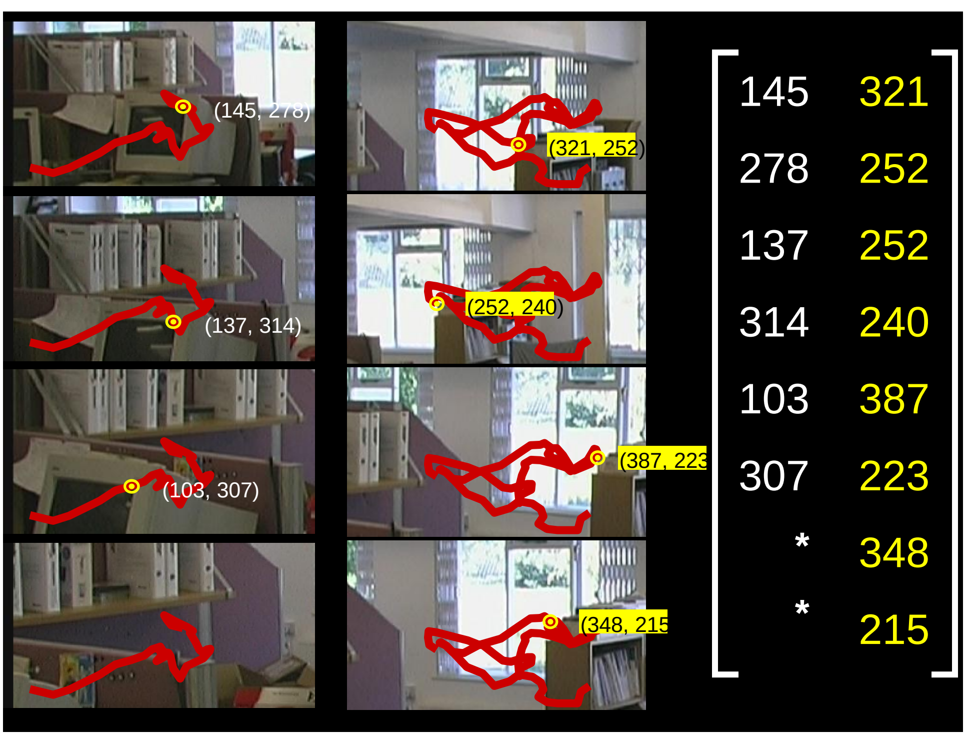

(145, 278)
145
278


(137, 314)
145
278
137
314

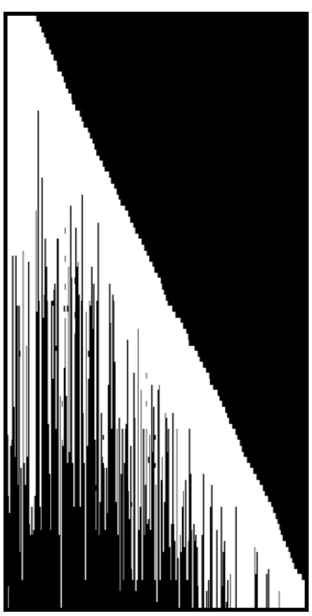

(103, 307)
145
278
137
314
103
307

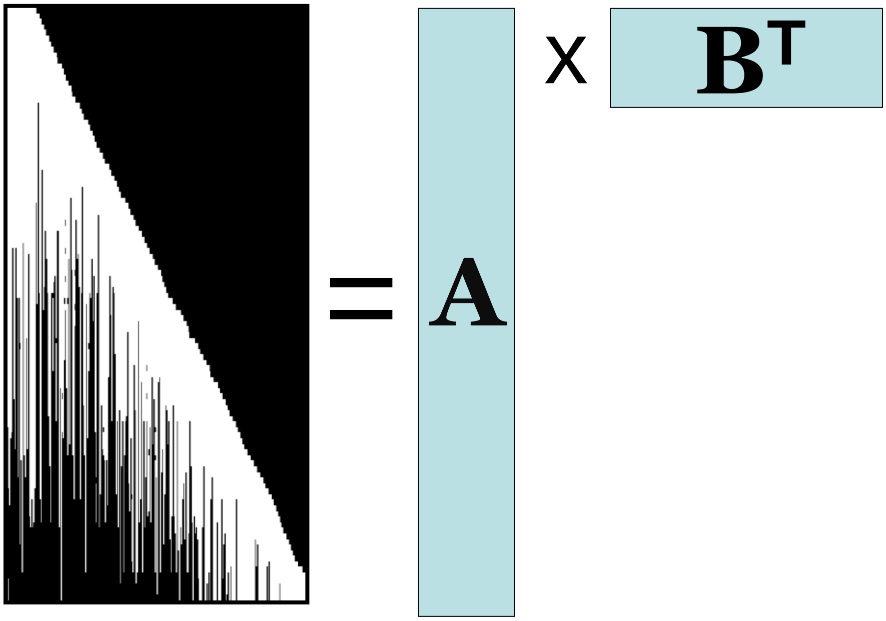

145
278
137
314
103
307
*
*


(145, 278)
(137, 314)
(103, 307)
145
278
137
314
103
307
*
*


(145, 278)
(137, 314)
(103, 307)
(321, 252)
(252, 240)
(387, 223
(348, 215
145 321
278 252
137 252
314 240
103 387
307 223
* 348
* 215

453	464	485	499	520	311	*	*	*	*	*	*	*	*	*
448	459	481	495	516	305	*	*	*	*	*	*	*	*	*
449	459	482	496	519	305	274	*	*	*	*	*	*	*	*
452	462	485	499	523	306	276	*	*	*	*	*	*	*	*
461	470	494	507	533	313	282	265	*	*	*	*	*	*	*
470	480	*	517	544	321	290	270	*	*	*	*	*	*	*
480	490	*	527	555	330	*	278	362	*	*	*	*	*	*
493	503	*	542	569	342	*	288	375	*	*	*	*	*	*
504	513	*	555	580	351	*	*	387	648	*	*	*	*	*
507	516	*	561	584	354	*	*	392	652	*	*	*	*	*
517	526	*	574	595	363	*	*	*	664	556	*	*	*	*
526	534	*	584	605	370	*	*	*	674	566	*	*	*	*
534	542	*	594	614	377	*	*	*	683	576	*	*	*	*
544	552	*	605	626	385	*	*	*	695	586	*	*	*	*
554	563	*	614	639	392	*	*	*	*	596	626	*	*	*
562	571	*	620	650	397	*	*	*	*	603	637	*	*	*
566	577	*	622	659	397	*	*	*	*	606	645	*	*	*
560	573	*	616	658	389	*	*	*	*	600	644	491	*	*
*	578	*	618	667	390	*	*	*	*	603	652	494	*	*
*	582	*	620	672	389	*	*	*	*	605	658	495	*	*
*	589	*	627	681	393	*	*	*	*	612	*	500	*	*
*	598	*	633	694	397	*	*	*	*	618	*	508	105	*
*	602	*	636	700	397	*	*	*	*	621	*	509	105	*
*	612	*	645	*	404	*	*	*	*	630	*	517	112	*
*	625	*	658	*	412	*	*	*	*	643	*	527	122	*
*	*	*	673	*	423	*	*	*	*	658	*	539	134	25
*	*	*	691	*	435	*	*	*	*	675	*	551	148	32
*	*	*	699	*	438	*	*	*	*	683	*	555	152	29
*	*	*	*	*	452	*	*	*	*	*	*	571	168	38
*	*	*	*	*	459	*	*	*	*	*	*	580	173	37
*	*	*	*	*	461	*	*	*	*	*	*	585	*	33
*	*	*	*	*	464	*	*	*	*	*	*	591	*	30
*	*	*	*	*	464	*	*	*	*	*	*	595	*	22
*	*	*	*	*	462	*	*	*	*	*	*	600	*	*
*	*	*	*	*	461	*	*	*	*	*	*	606	*	*





Matrix factorization with missing data

$$\min_{\mathbf{A},\mathbf{B}} \left\| \mathbf{W} \odot (\mathbf{M} - \mathbf{A}\mathbf{B}^\top) \right\|_\rho$$

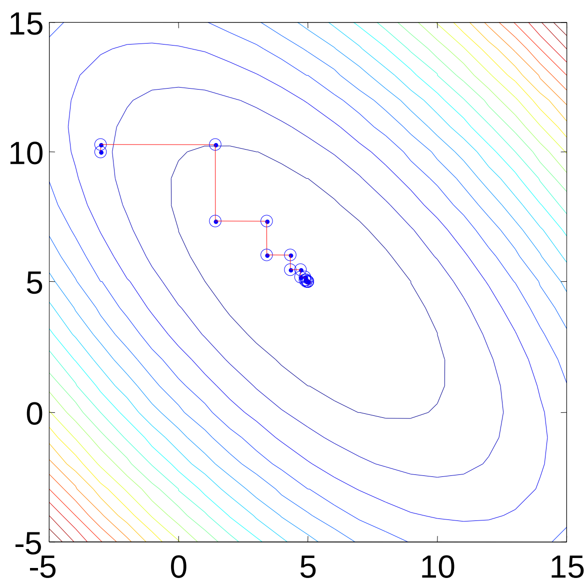
- Structure from motion
- PCA with missing data
- Shape from shading
- ...

$$\mathbf{P} \odot \mathbf{Q} = \mathbf{R} \Leftrightarrow r_{ij} = p_{ij} q_{ij}$$

Missing data algorithm 1: Alternation

If we know A…

Missing data algorithm 1: Alternation



Missing data algorithm 1: Alternation

But…

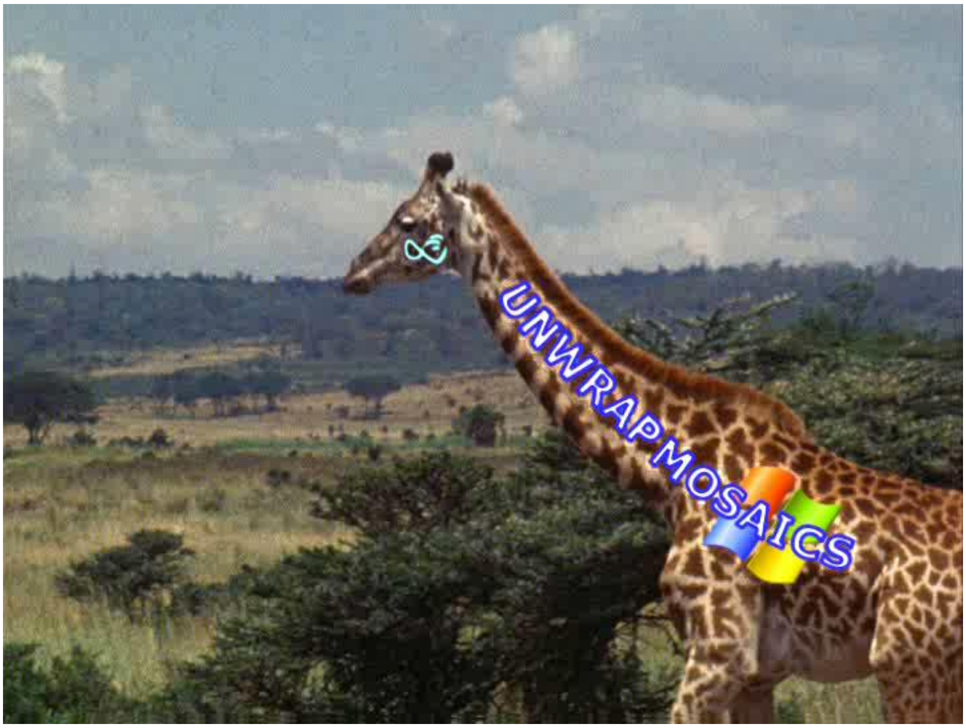
- Factorization uses affine camera model, the real world is **perspective**
- Factorization computes minimum of error

$$e(P,X) = \mathrm{norm}(M - P\,X)$$

  Would like to optimize **weighted error**

$$e(P,X) = \mathrm{norm}(W\ \bar{}\ (M - P\,X))$$

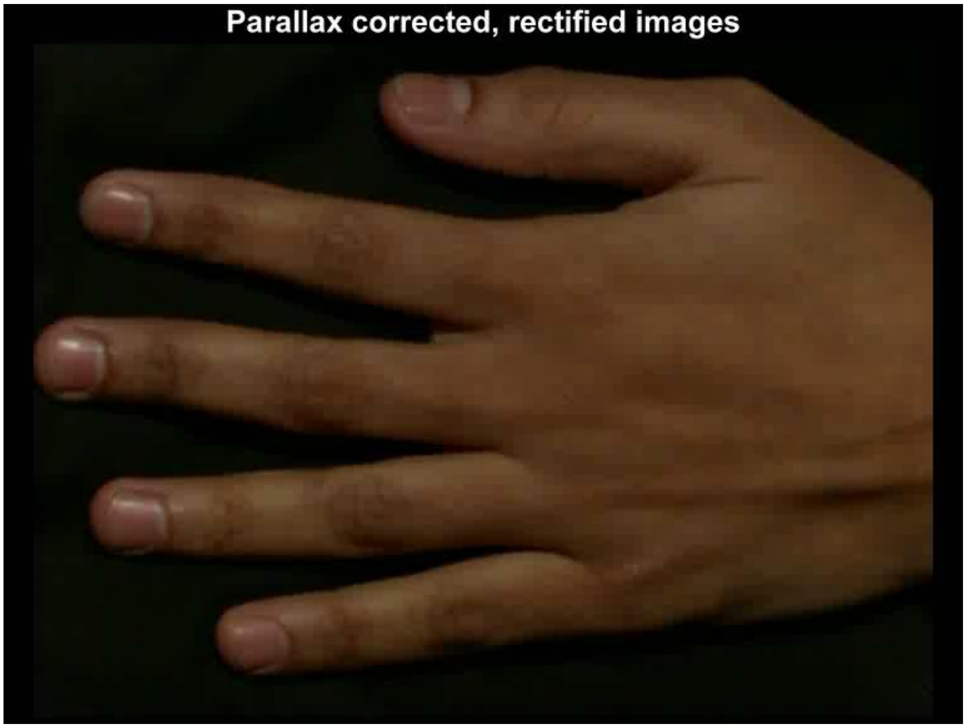
- Must deal with **missing data**


UNWRAP MOSAICS


unwrap mosaics
Rav-Acha | Kohli | Rother | Fitzgibbon
http://research.microsoft.com/unwrap

Photometric stereo

- Camera and flash mounted together
- Geometric target allows estimation of position of camera...
- ... and therefore of position of light


Parallax corrected, rectified images