Structure from Motion

Computer Vision

CS 143, Brown

James Hays

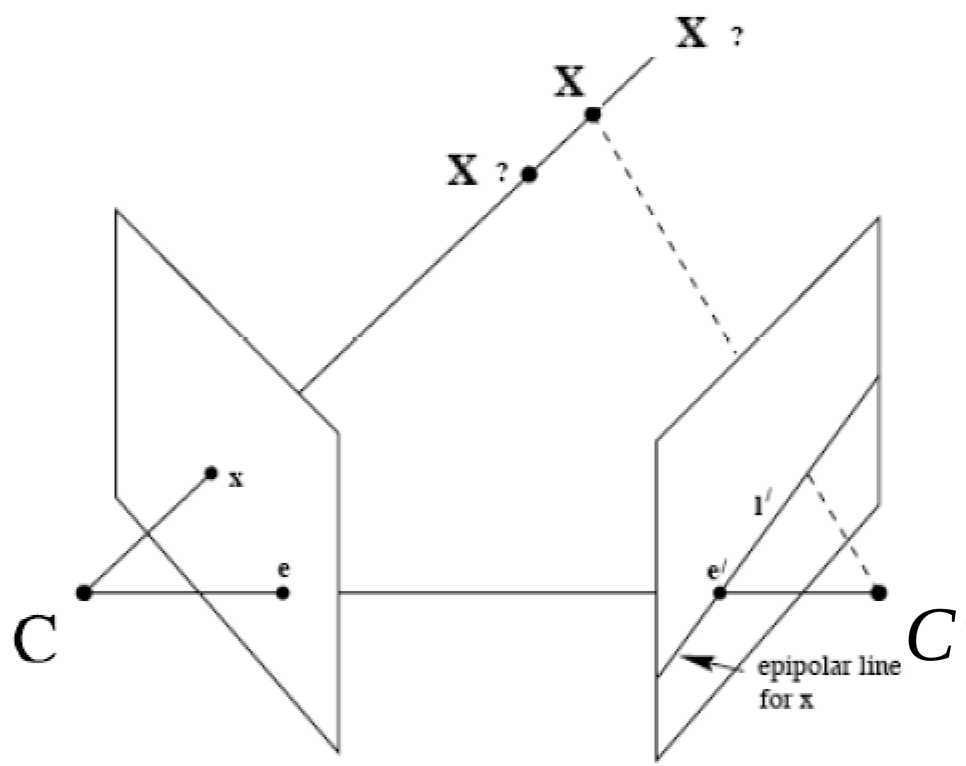
Many slides adapted from Derek Hoiem, Lana Lazebnik, Silvio Saverese, Steve Seitz, and Martial Hebert

This class: structure from motion

- Recap of epipolar geometry
  - Depth from two views
- Affine structure from motion

Recap: Epipoles

- Point x in left image corresponds to **epipolar line** l’ in right image
- Epipolar line passes through the epipole (the intersection of the cameras’ baseline with the image plane



Recap: Fundamental Matrix

- Fundamental matrix maps from a point in one image to a line in the other

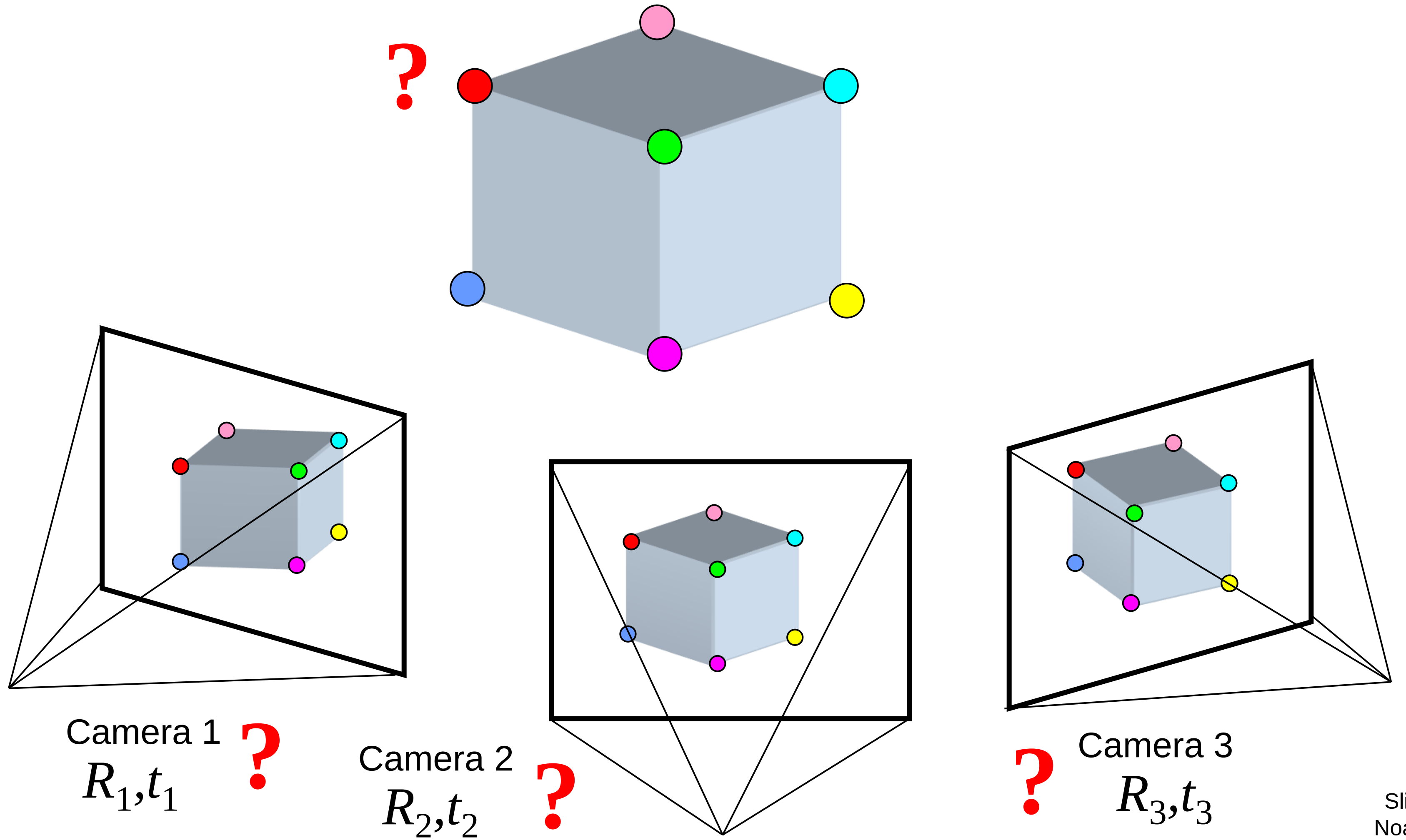
$$\mathbf{l}' = \mathbf{F}\mathbf{x} \qquad \mathbf{l} = \mathbf{F}^\top \mathbf{x}'$$

- If x and x' correspond to the same 3d point X:

$$\mathbf{x}'^\top \mathbf{F}\mathbf{x} = 0$$

Structure from motion

- Given a set of corresponding points in two or more images, compute the camera parameters and the 3D point coordinates



Slide credit: Noah Snavely

Structure from motion ambiguity

- If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same:

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\frac{1}{k}\mathbf{P}\right)(k\mathbf{X})$$

It is impossible to recover the absolute scale of the scene!

How do we know the scale of image content?


Photography
Your Eyes Deserve It
TOTAL
TOTAL RECLAIMS
TOTAL RECLAIMS

Structure from motion ambiguity

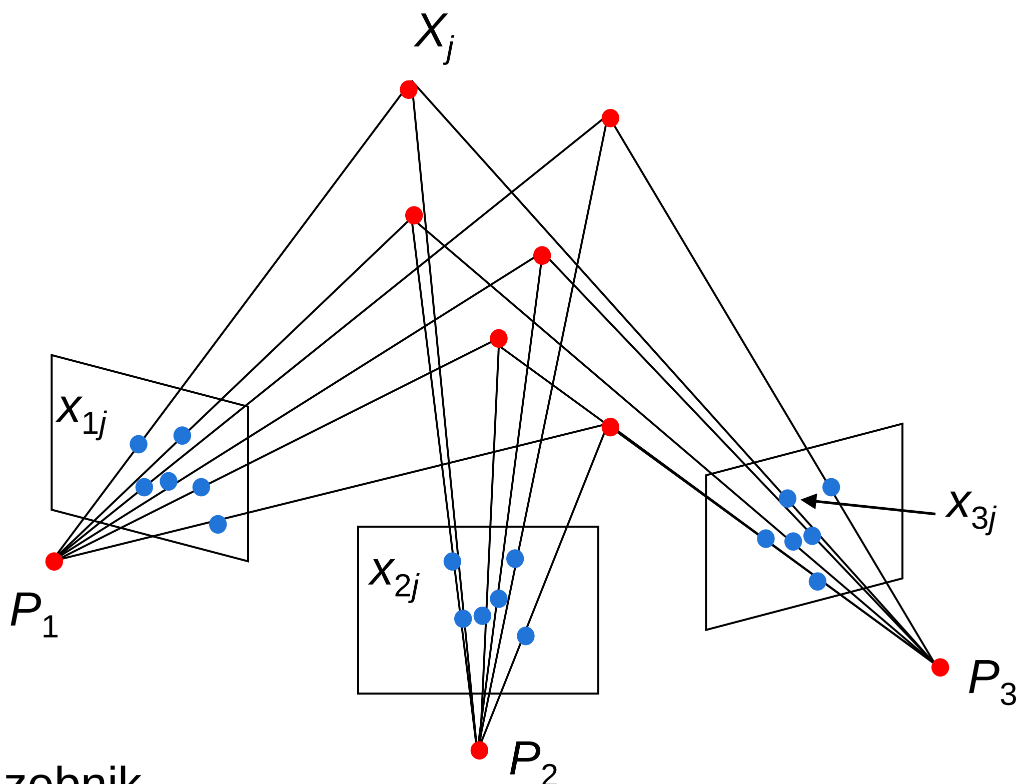
- If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same

- More generally: if we transform the scene using a transformation **Q** and apply the inverse transformation to the camera matrices, then the images do not change

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)\left(\mathbf{Q}\mathbf{X}\right)$$

Projective structure from motion

- Given: *m* images of *n* fixed 3D points
  - $\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j \,, \; i = 1, \ldots, m, \quad j = 1, \ldots, n$
- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* corresponding points $\mathbf{x}_{ij}$

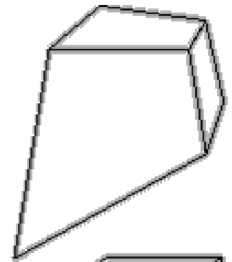
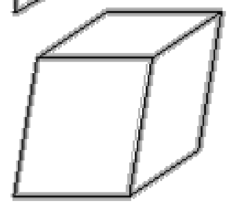
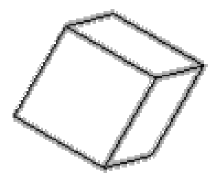
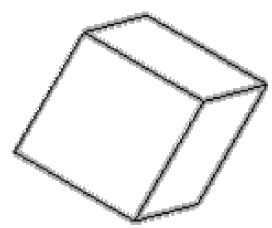


Slides from Lana Lazebnik

Projective structure from motion

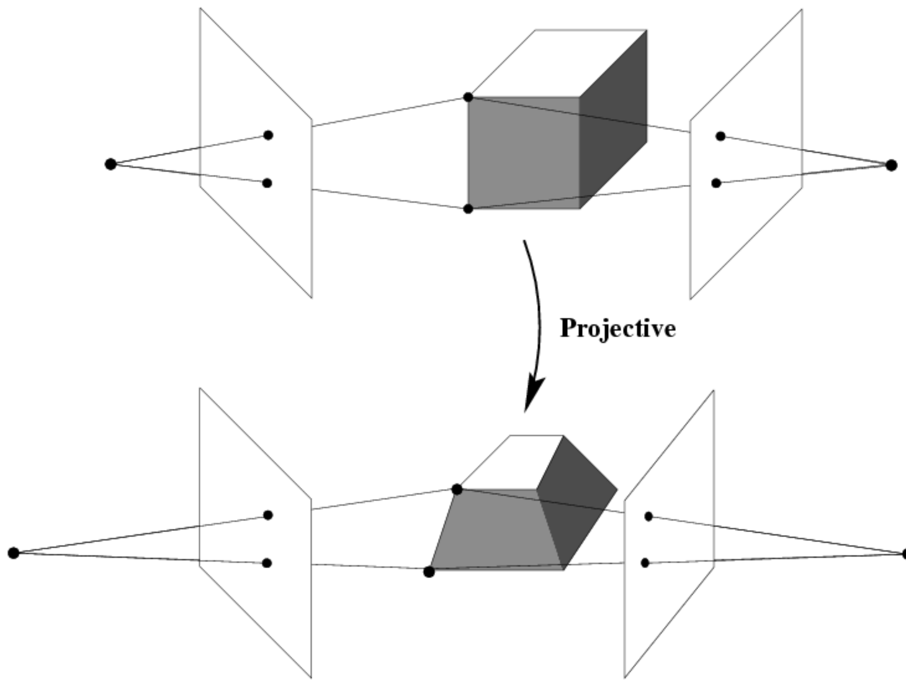
- Given: m images of n fixed 3D points
  - $\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j , \qquad i = 1, \dots , m, \quad j = 1, \dots , n$
- Problem: estimate m projection matrices $\mathbf{P}_i$ and n 3D points $\mathbf{X}_j$ from the mn corresponding points $\mathbf{x}_{ij}$
- With no calibration info, cameras and points can only be recovered up to a 4x4 projective transformation **Q**:
  - $\mathbf{X} \rightarrow \mathbf{QX}, \mathbf{P} \rightarrow \mathbf{PQ}^{-1}$
- We can solve for structure and motion when
  - $2mn >= 11m + 3n - 15$
- For two cameras, at least 7 points are needed

Types of ambiguity

Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} s\,R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, ratios of length
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, lengths

- With no constraints on the camera calibration matrix or on the scene, we get a *projective* reconstruction
- Need additional information to *upgrade* the reconstruction to affine, similarity, or Euclidean

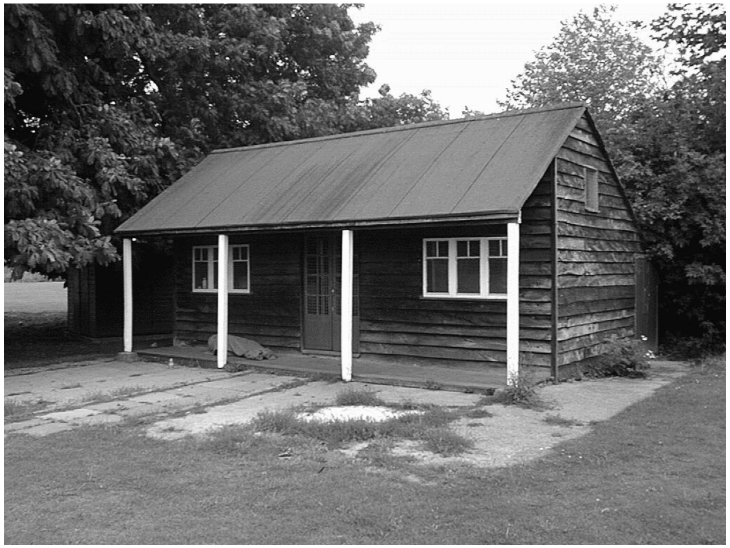
Projective ambiguity

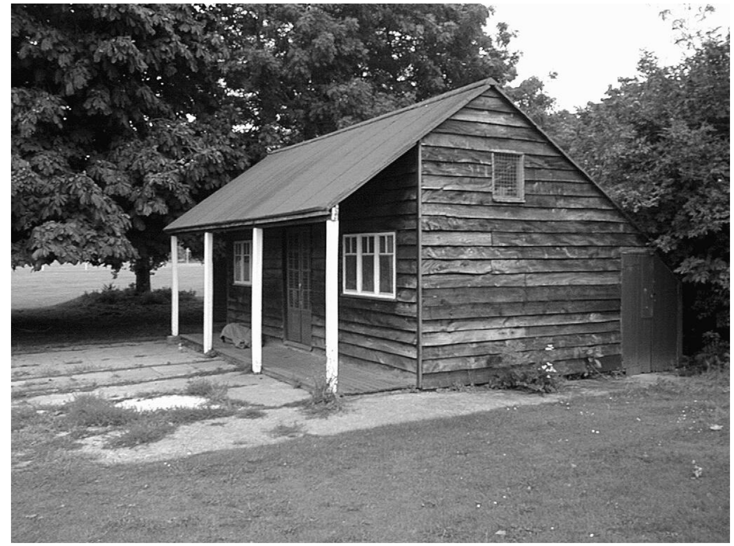


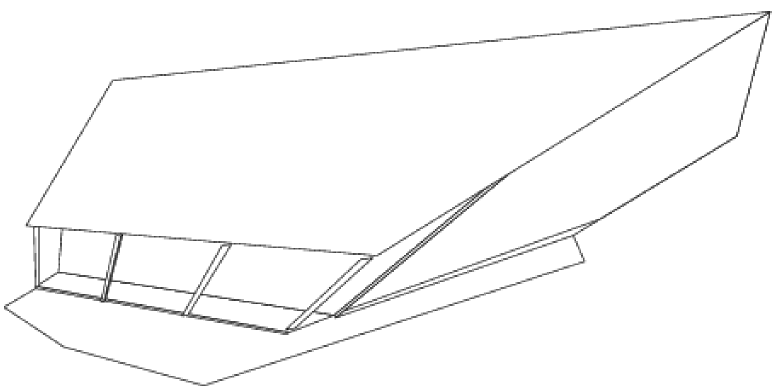
$$\mathbf{Q_p} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ \mathrm{v}^{\mathsf{T}} & v \end{bmatrix}$$

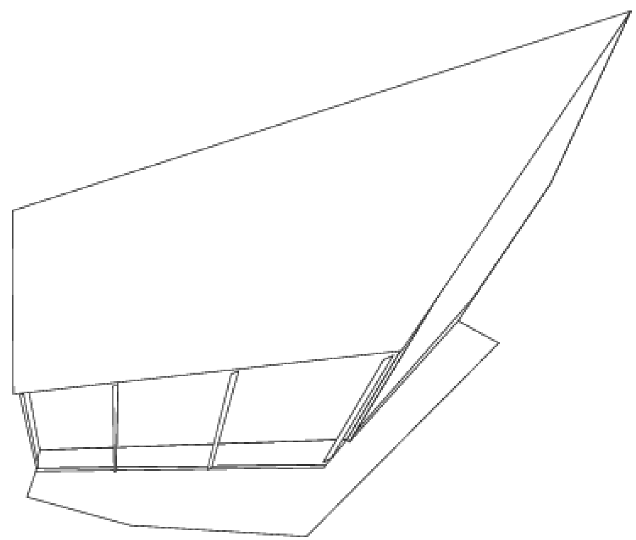
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_\mathbf{P}^{-1}\right)\left(\mathbf{Q}_\mathbf{P}\mathbf{X}\right)$$

Projective ambiguity

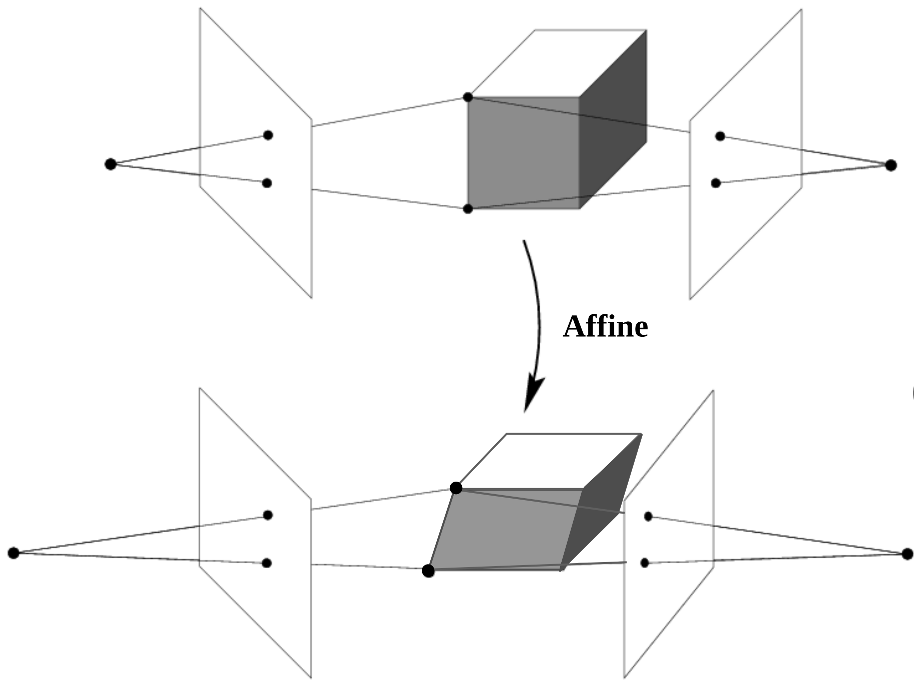








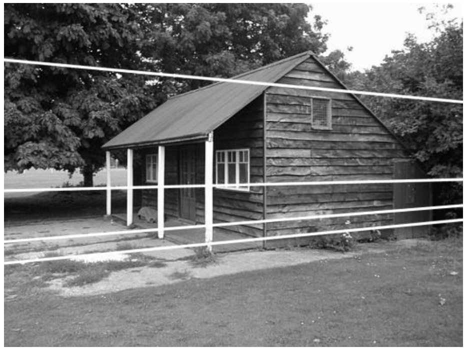
Affine ambiguity

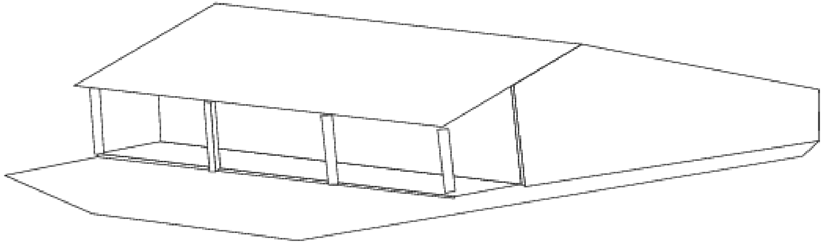


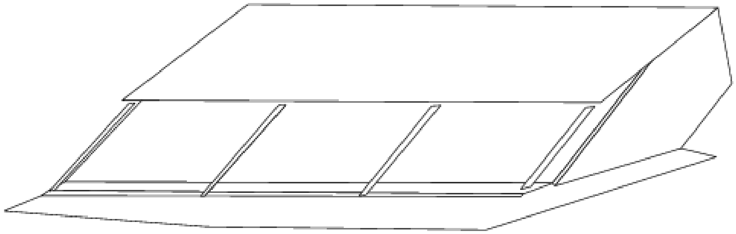
$$\mathbf{Q}_A = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_A^{-1}\right)\left(\mathbf{Q}_A\mathbf{X}\right)$$

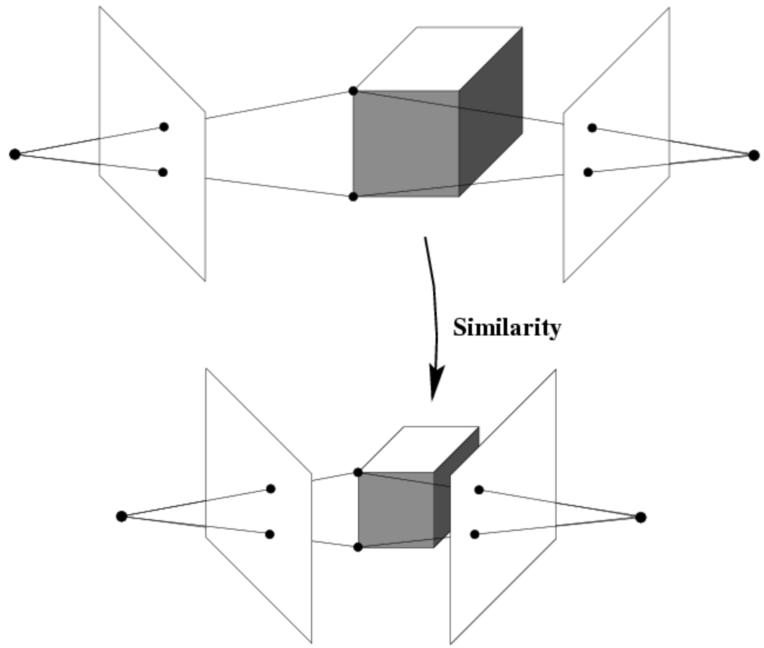
Affine ambiguity







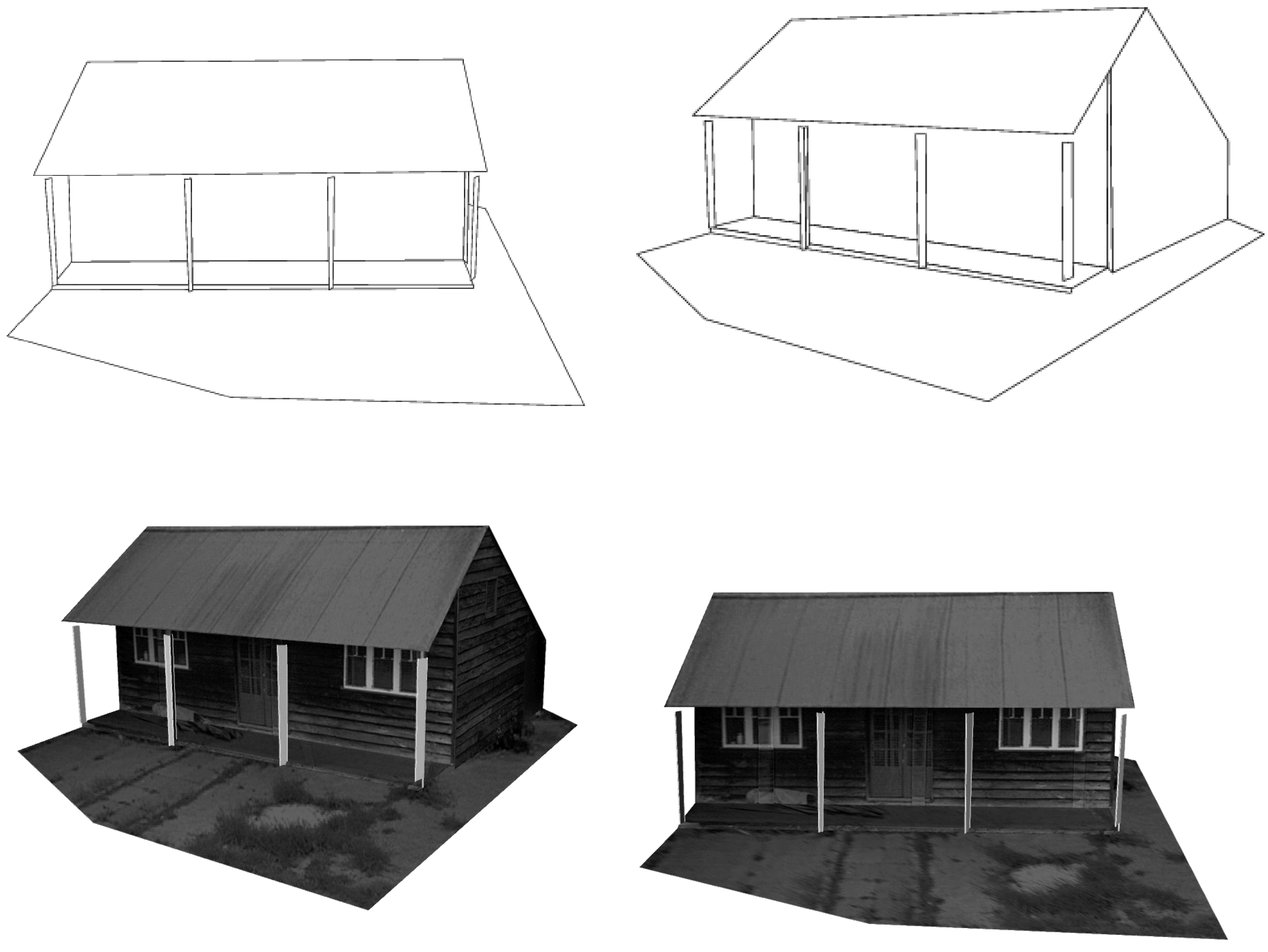
Similarity ambiguity



$$\mathbf{Q}_s = \begin{bmatrix} s\mathrm{R} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_S^{-1}\right)\left(\mathbf{Q}_S\mathbf{X}\right)$$

Similarity ambiguity



Bundle adjustment

- Non-linear method for refining structure and motion
- Minimizing reprojection error

$$E(\mathbf{P},\mathbf{X}) = \sum_{i=1}^{m}\sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, \mathbf{P}_i\mathbf{X}_j\right)^2$$

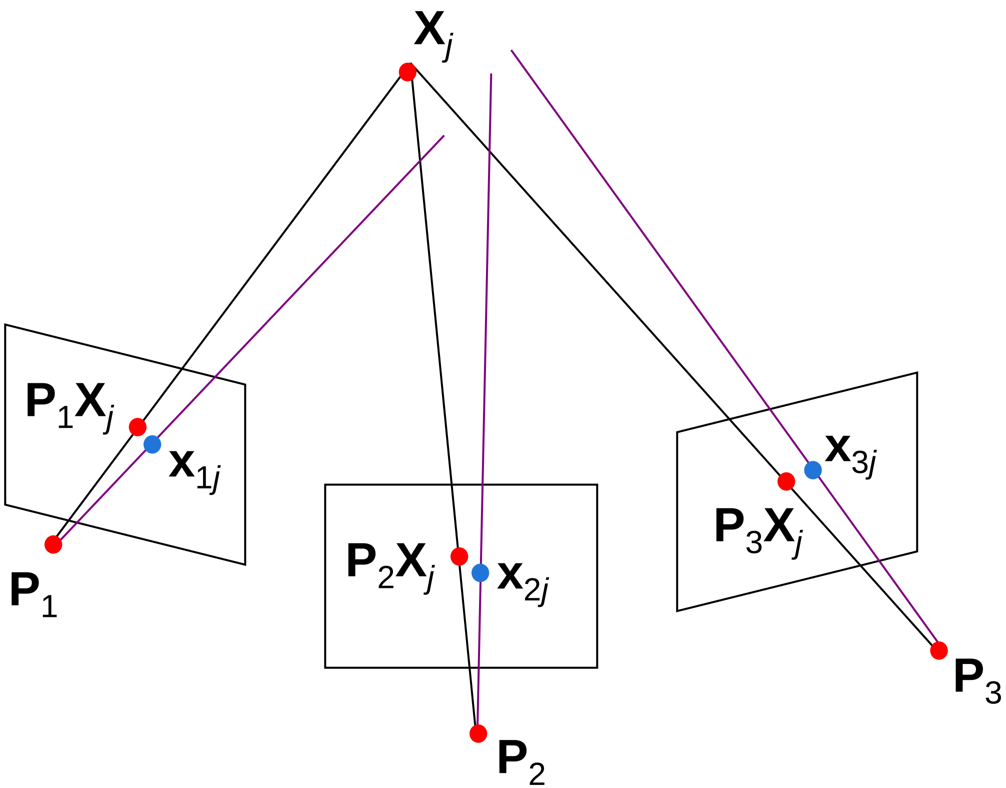
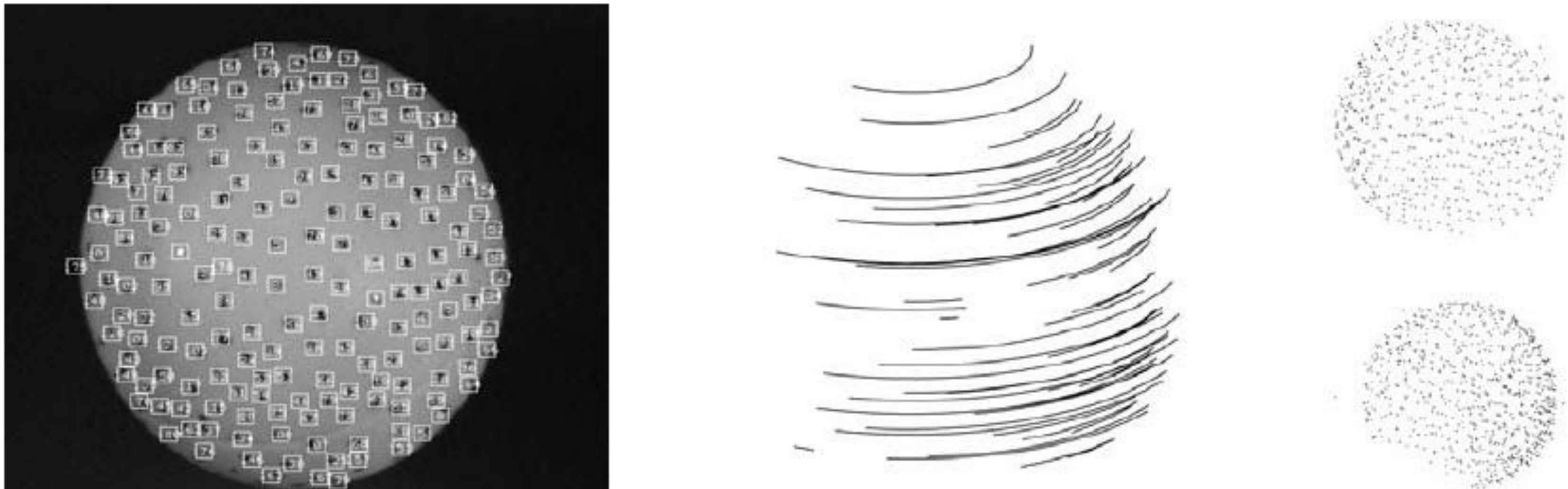


Photo synth

Noah Snavely, Steven M. Seitz, Richard Szeliski, "Photo tourism: Exploring photo collections in 3D," SIGGRAPH 2006

http://photosynth.net/

Structure from motion under orthographic projection



(a) (b) (c)

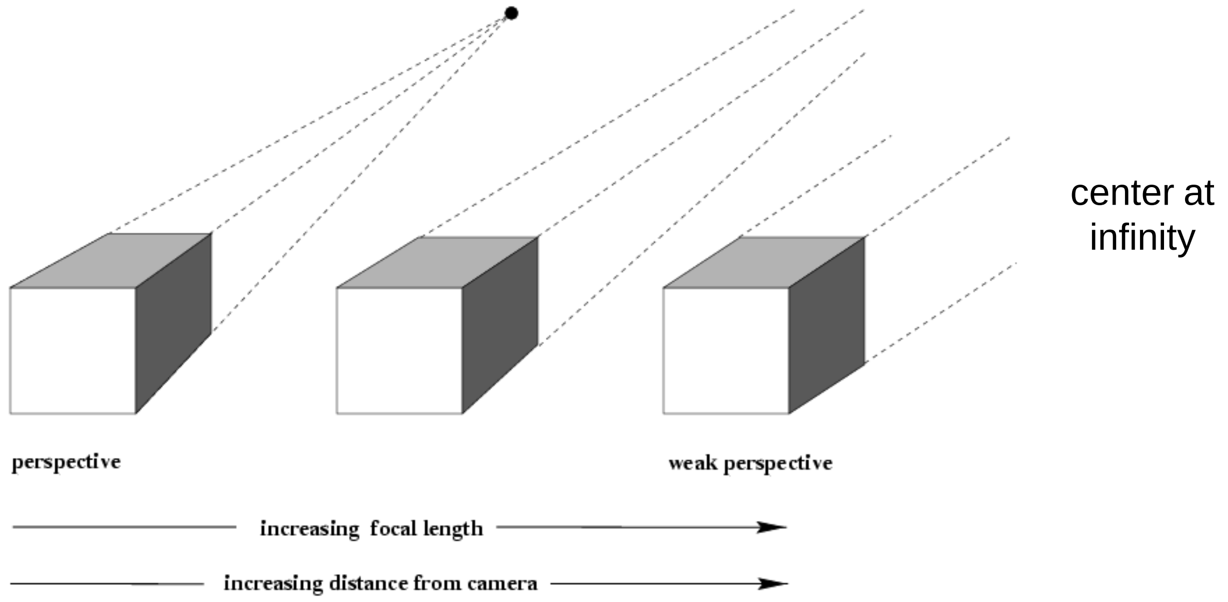
3D Reconstruction of a Rotating Ping-Pong Ball

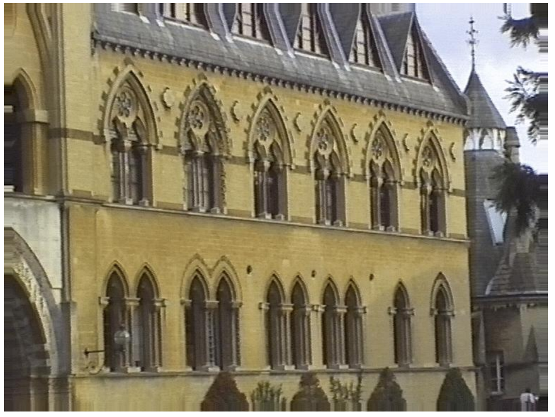
- Reasonable choice when
  - Change in depth of points in scene is much smaller than distance to camera
  - Cameras do not move towards or away from the scene

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137-154, November 1992.

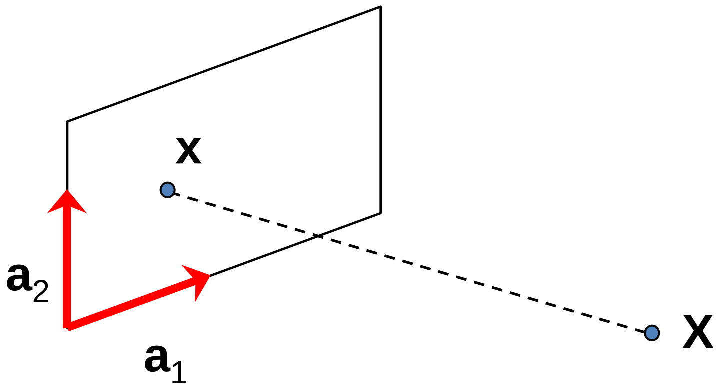
Structure from motion

- Let's start with *affine cameras* (the math is easier)





Affine projection for rotated/translated camera



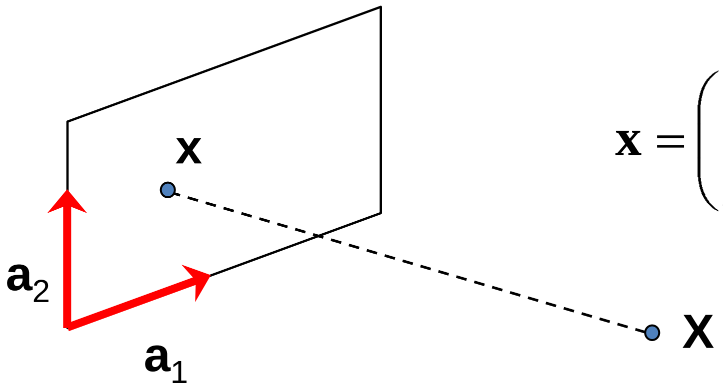
$$\begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(R'_f \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix} + t_f \right)$$

$$R_f = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R'_f$$

$$\begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = R_f \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix} + t_f$$

Affine structure from motion

- Affine projection is a linear mapping + translation in inhomogeneous coordinates



$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \mathbf{AX} + \mathbf{t}$$

Projection of world origin

1. We are given corresponding 2D points (**x**) in several frames
2. We want to estimate the 3D points (**X**) and the affine parameters of each camera (**A**)

Affine structure from motion

- Centering: subtract the centroid of the image points

$$\hat{\mathbf{x}}_{ij} = \mathbf{x}_{ij} - \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_{ik} = \mathbf{A}_i\mathbf{X}_j + \mathbf{b}_i - \frac{1}{n}\sum_{k=1}^{n}\left(\mathbf{A}_i\mathbf{X}_k + \mathbf{b}_i\right)$$

$$= \mathbf{A}_i\left(\mathbf{X}_j - \frac{1}{n}\sum_{k=1}^{n}\mathbf{X}_k\right) = \mathbf{A}_i\hat{\mathbf{X}}_j$$

- For simplicity, assume that the origin of the world coordinate system is at the centroid of the 3D points
- After centering, each normalized point $\mathbf{x}_{ij}$ is related to the 3D point $\mathbf{X}_i$ by

$$\hat{\mathbf{x}}_{ij} = \mathbf{A}_i\mathbf{X}_j$$

Suppose we know 3D points and affine camera parameters …

then, we can compute the observed 2d positions of each point

$$\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

3D Points (3xn)

Camera Parameters (2mx3)

What if we instead observe corresponding 2d image points?

Can we recover the camera parameters and 3d points?

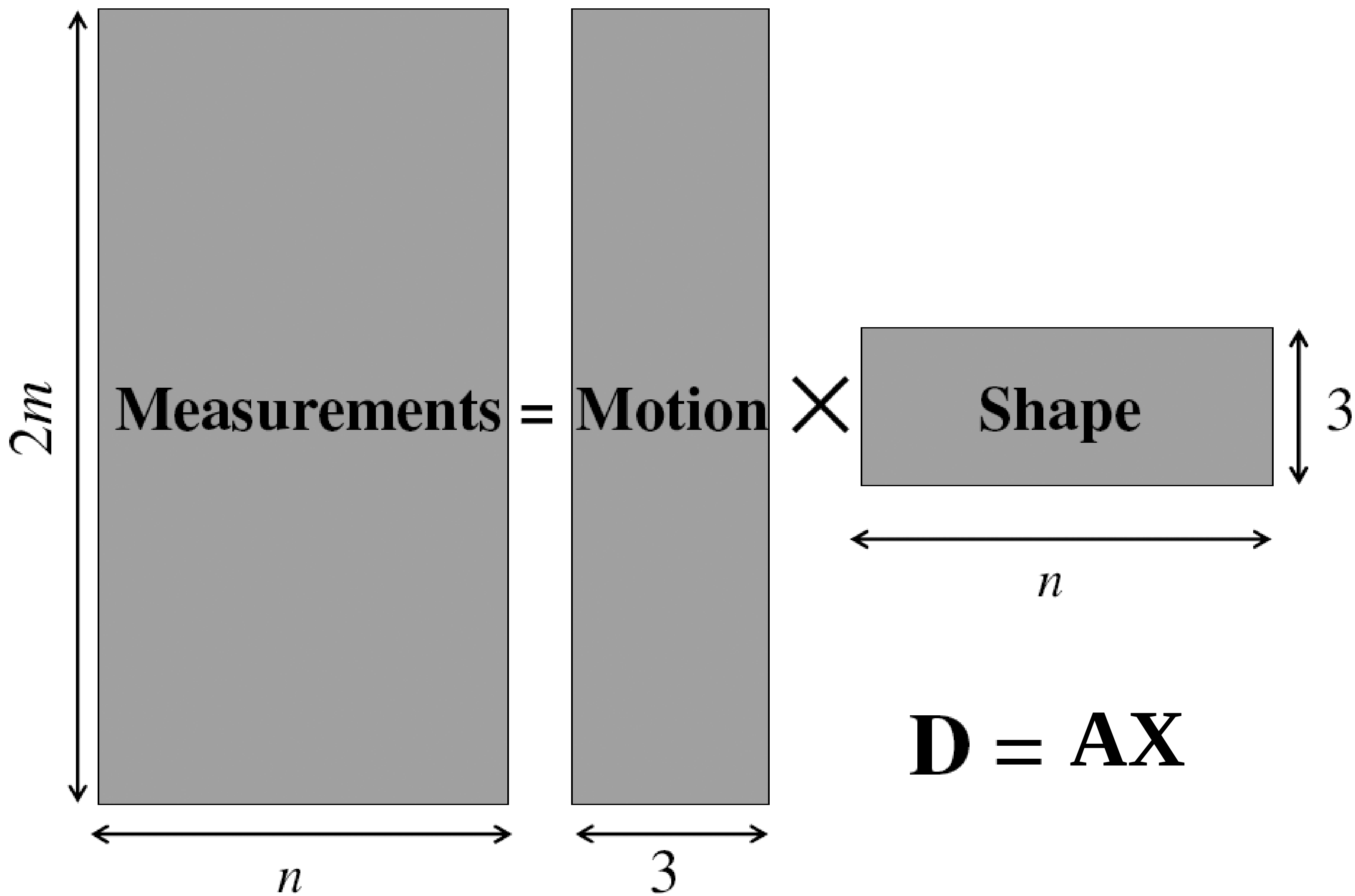
cameras ($2m$)

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix} \overset{?}{\Rightarrow} \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

points (n)

What rank is the matrix of 2D points?

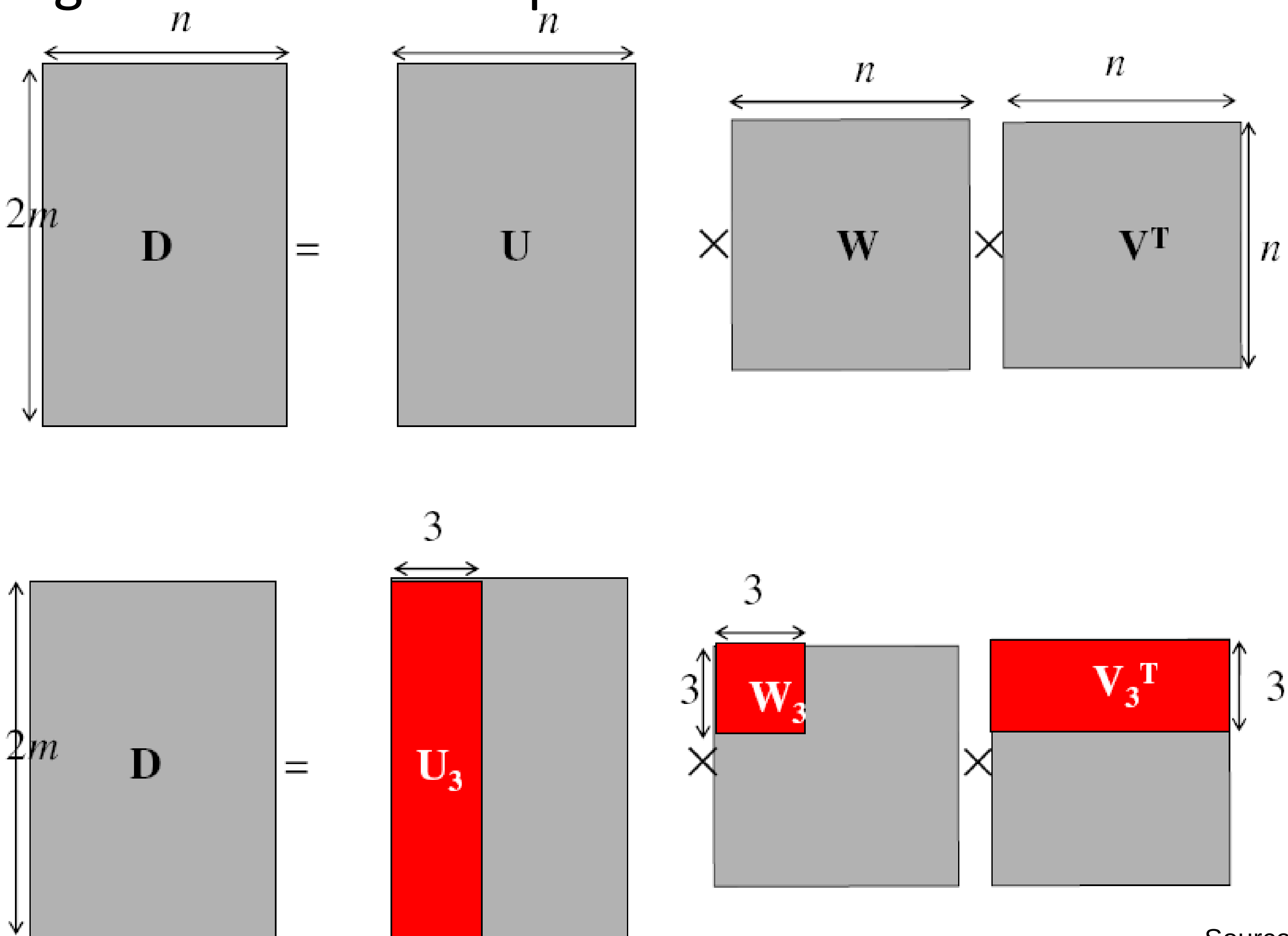
Factorizing the measurement matrix



Source: M. Hebert

Factorizing the measurement matrix

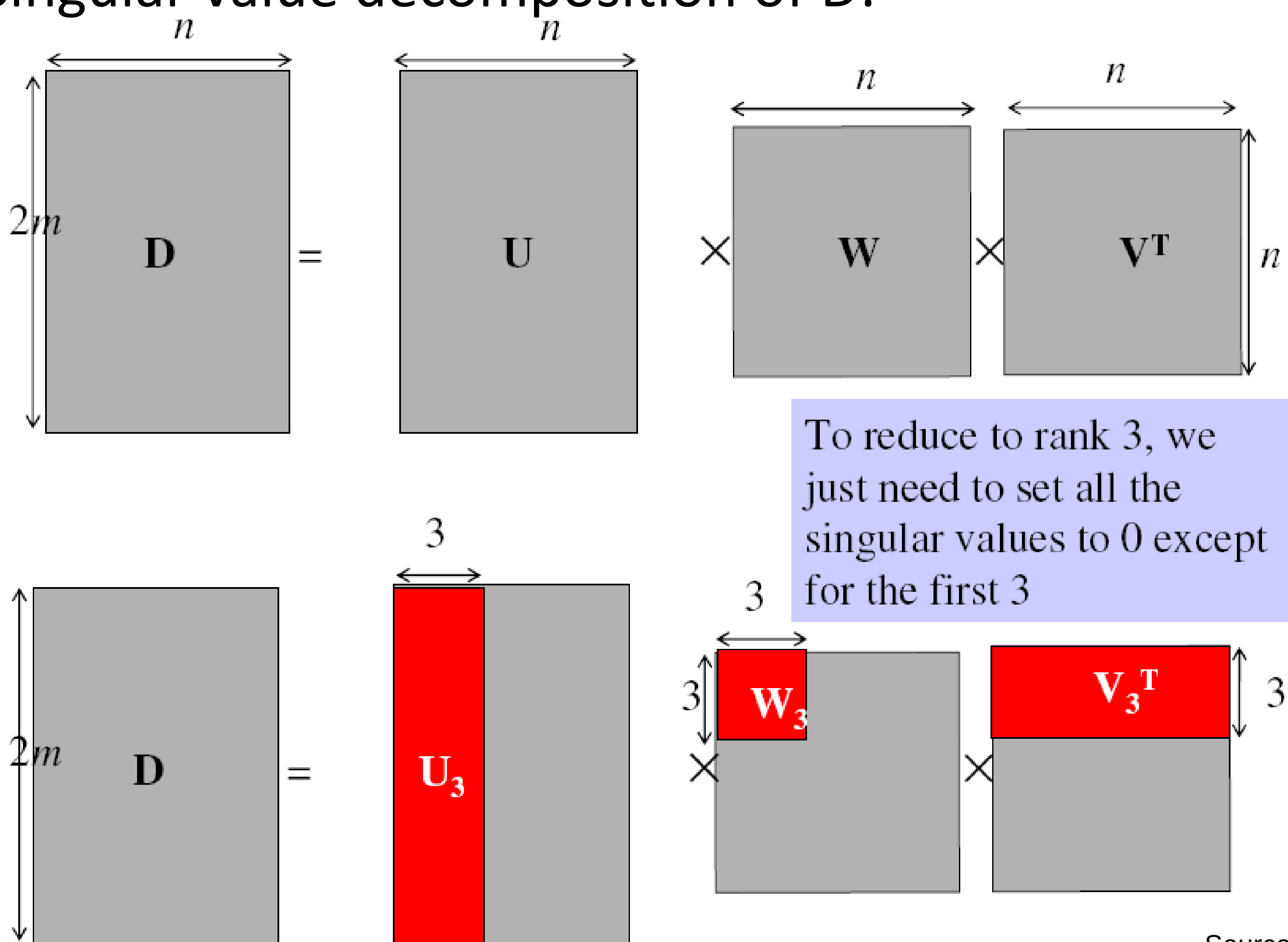
- Singular value decomposition of D:



Source: M. Hebert

Factorizing the measurement matrix

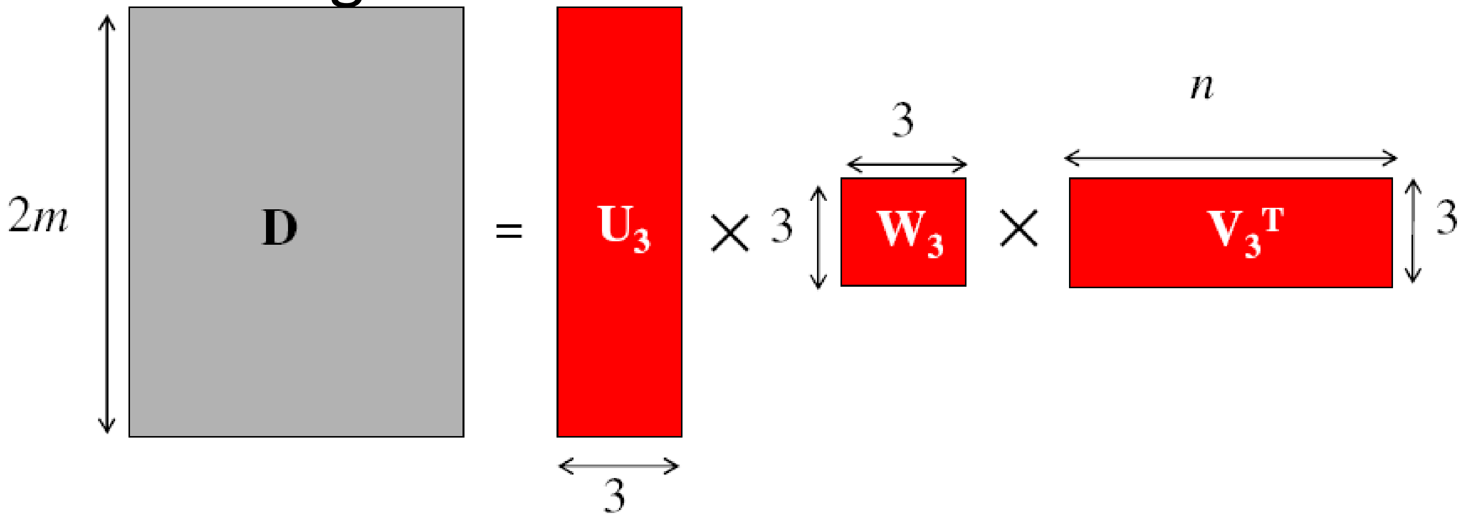
- Singular value decomposition of D:



Source: M. Hebert

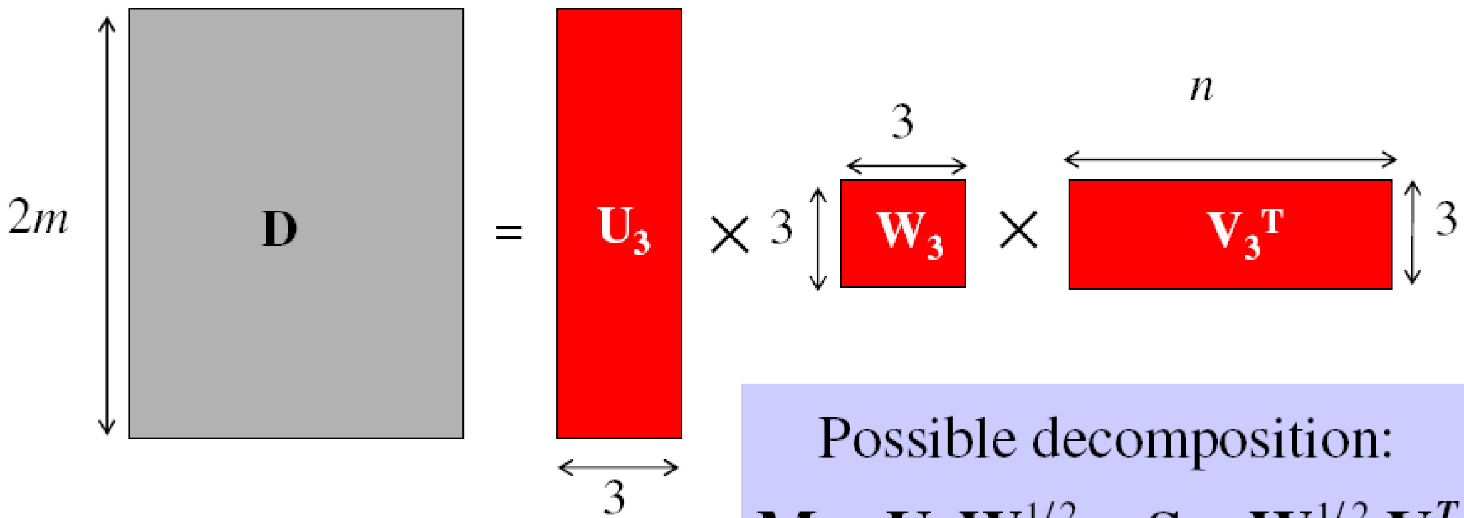
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



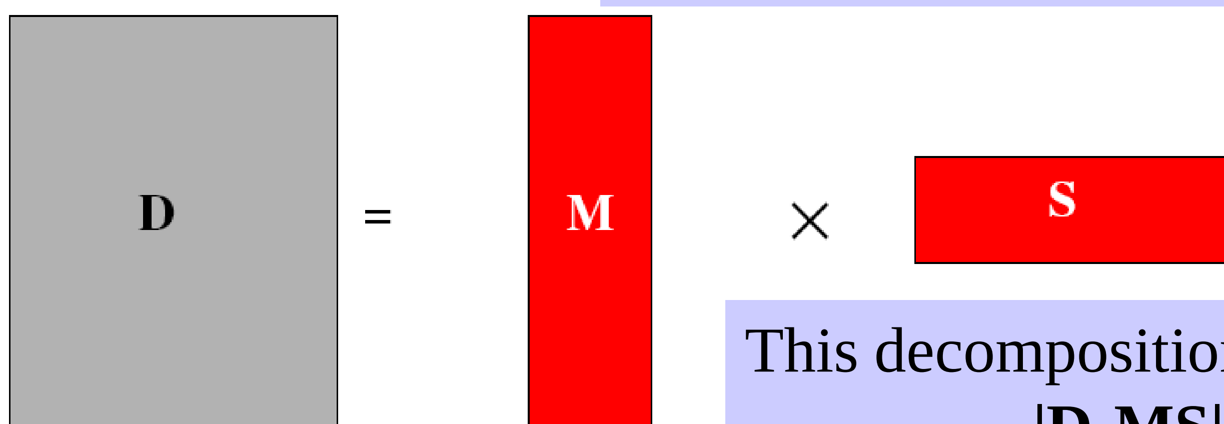
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



Possible decomposition:

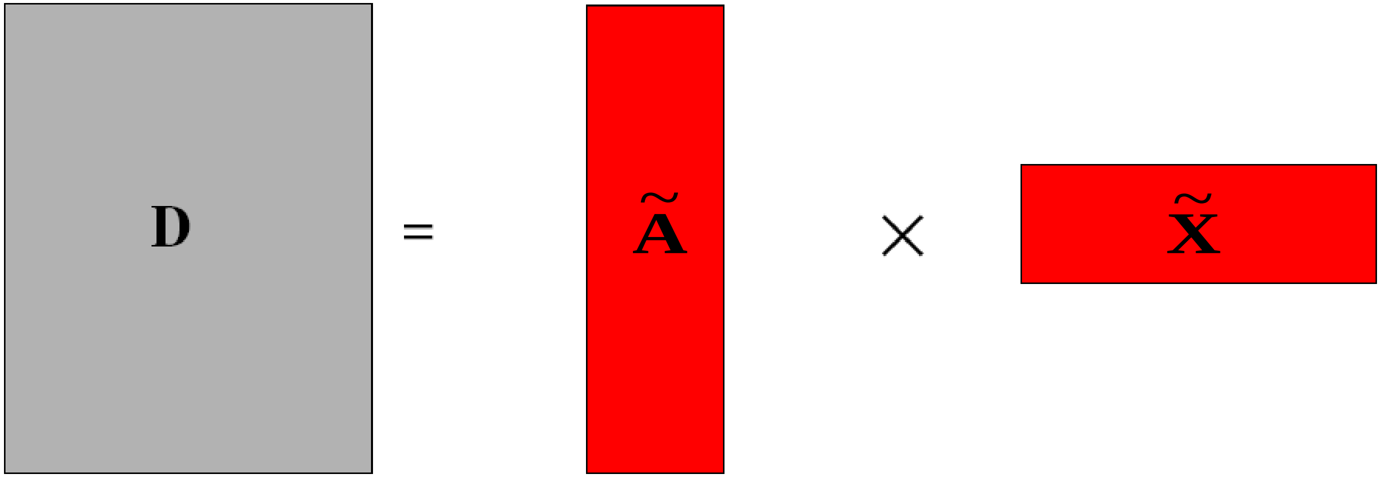
$$\mathbf{M} = \mathbf{U}_3 \mathbf{W}_3^{1/2} \qquad \mathbf{S} = \mathbf{W}_3^{1/2}\,\mathbf{V}_3^T$$



This decomposition minimizes $|\mathbf{D}\text{-}\mathbf{MS}|^2$

Source: M. Hebert

Affine ambiguity

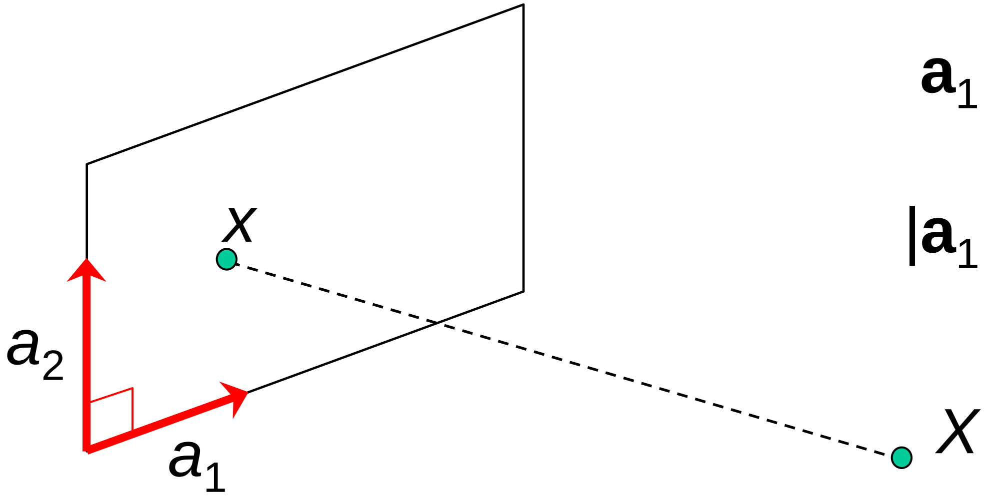


- The decomposition is not unique. We get the same **D** by using any 3 3 matrix **C** and applying the transformations **A → AC, X → C^{-1}X**
- That is because we have only an affine transformation and we have not enforced any Euclidean constraints (like forcing the image axes to be perpendicular, for example)

Source: M. Hebert

Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and scale is 1



$$\mathbf{a}_1 \cdot \mathbf{a}_2 = 0$$

$$|\mathbf{a}_1|^2 = |\mathbf{a}_2|^2 = 1$$

- This translates into $3m$ equations in $\mathbf{L} = \mathbf{C}\mathbf{C}^T$:

$$\mathbf{A}_i \, \mathbf{L} \, \mathbf{A}_i^T = \mathbf{Id}, \qquad i = 1, \ldots, m$$

  - Solve for **L**
  - Recover **C** from **L** by Cholesky decomposition: $\mathbf{L} = \mathbf{C}\mathbf{C}^T$
  - Update **M** and **S**: $\mathbf{M} = \mathbf{MC}, \ \mathbf{S} = \mathbf{C}^{-1}\mathbf{S}$

Source: M. Hebert

Algorithm summary

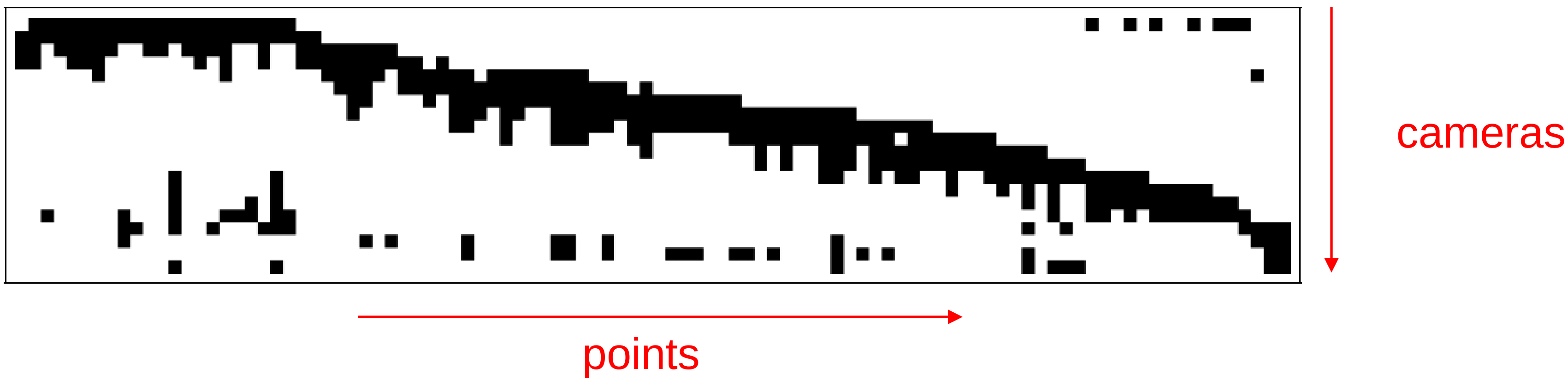
- Given: *m* images and *n* tracked features $\mathbf{x}_{ij}$
- For each image *i, c*enter the feature coordinates
- Construct a 2*m*   *n* measurement matrix **D**:
  - Column *j* contains the projection of point *j* in all views
  - Row *i* contains one coordinate of the projections of all the *n* points in image *i*
- Factorize **D**:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\ \mathbf{W}\ \mathbf{V}^{\mathbf{T}}$
  - Create $\mathbf{U}_3$ by taking the first 3 columns of **U**
  - Create $\mathbf{V}_3$ by taking the first 3 columns of **V**
  - Create $\mathbf{W}_3$ by taking the upper left 3   3 block of **W**
- Create the motion (affine) and shape (3D) matrices:

  $\mathbf{A} = \mathbf{U}_3\mathbf{W}_3^{½}$ and $\mathbf{X} = \mathbf{W}_3^{½}\ \mathbf{V}_3^{T}$
- Eliminate affine ambiguity

Source: M. Hebert

Dealing with missing data

- So far, we have assumed that all points are visible in all views
- In reality, the measurement matrix typically looks something like this:



One solution:

- – solve using a dense submatrix of visible points
- – Iteratively add new cameras

A nice short explanation

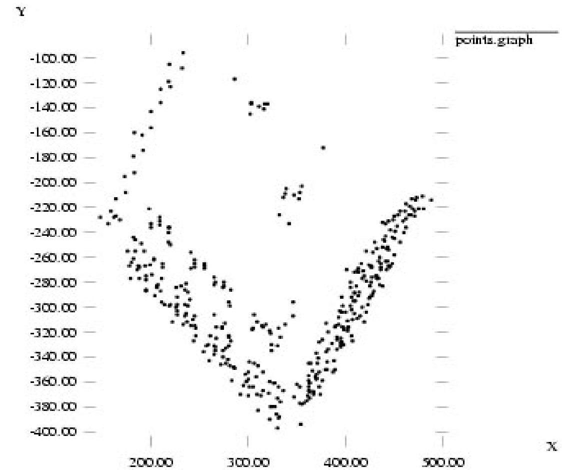
- Class notes from Lischinksi and Gruber

  http://www.cs.huji.ac.il/~csip/sfm.pdf

Reconstruction results (project 5)

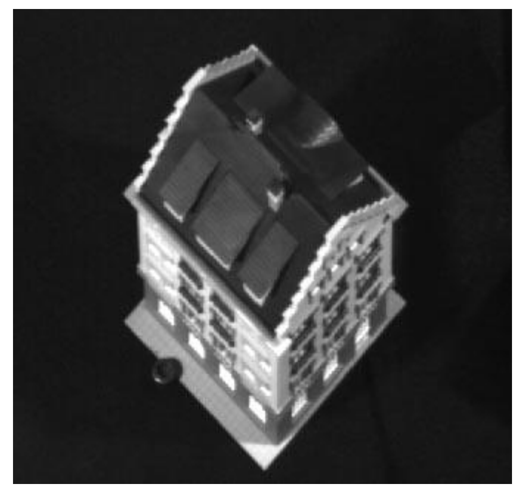
1

60



120

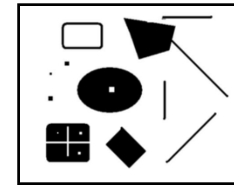
150



C. Tomasi and T. Kanade. [Shape and motion from image streams under orthography: A factorization method.] *IJCV*, 9(2):137-154, November 1992.

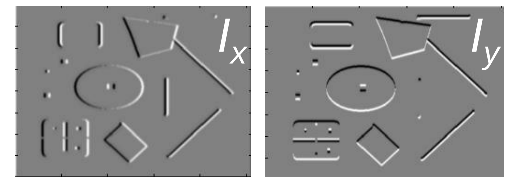
Project 5

1. Detect interest points (e.g., Harris)



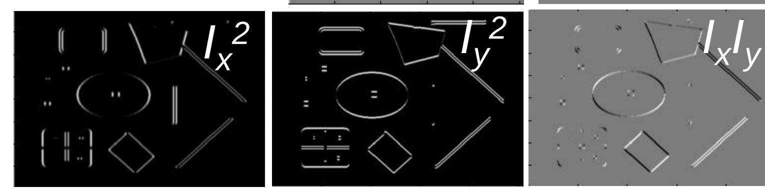
$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives



$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

2. Square of derivatives



3. Gaussian filter $g(\sigma_I)$

4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\operatorname{trace}(\mu(\sigma_I, \sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_xI_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression

Project 5

2. Correspondence via Lucas-Kanade tracking

a) Initialize (x',y') = (x,y)

b) Compute (u,v) by

Original (x,y) position

$I_t = I(x', y', t+1) - I(x, y, t)$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

2nd moment matrix for feature patch in first image

displacement

c) Shift window by (u, v): `x'=x'+u;  y'=y'+v;`

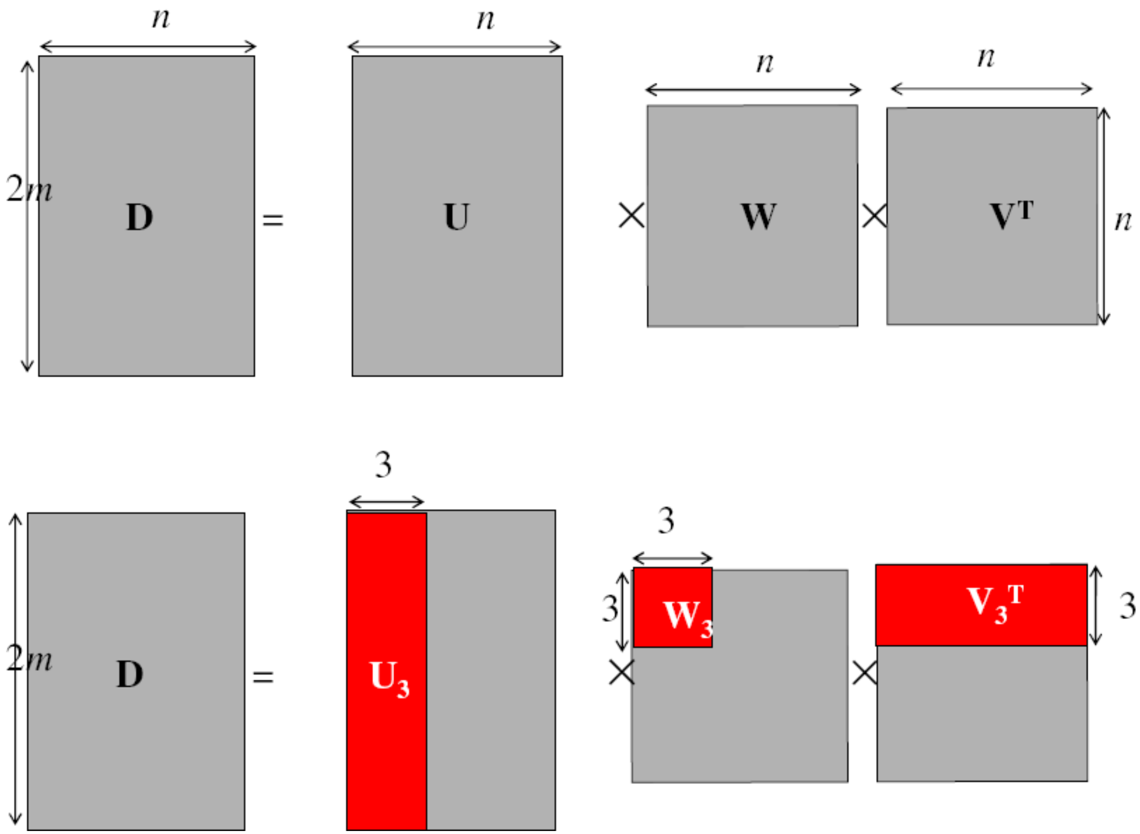
d) *(extra credit)* Recalculate I_t

e) *(extra credit)* Repeat steps 2-4 until small change

- Use interpolation for subpixel values

Project 5

3. Get Affine camera matrix and 3D points using Tomasi-Kanade factorization



+ Solve for orthographic constraints

Project 5

- Tips
  - Helpful matlab functions: interp2, meshgrid, ordfilt2 (for getting local maximum), svd, chol
  - When selecting interest points, must choose appropriate threshold on Harris criteria or the smaller eigenvalue, or choose top N points
  - Vectorize to make tracking fast (interp2 will be the bottleneck)
  - Get tracking working on one point for a few frames before trying to get it working for all points