7. Structure from Motion (SfM)

Estimating 3D structure (rigid) from multiple images

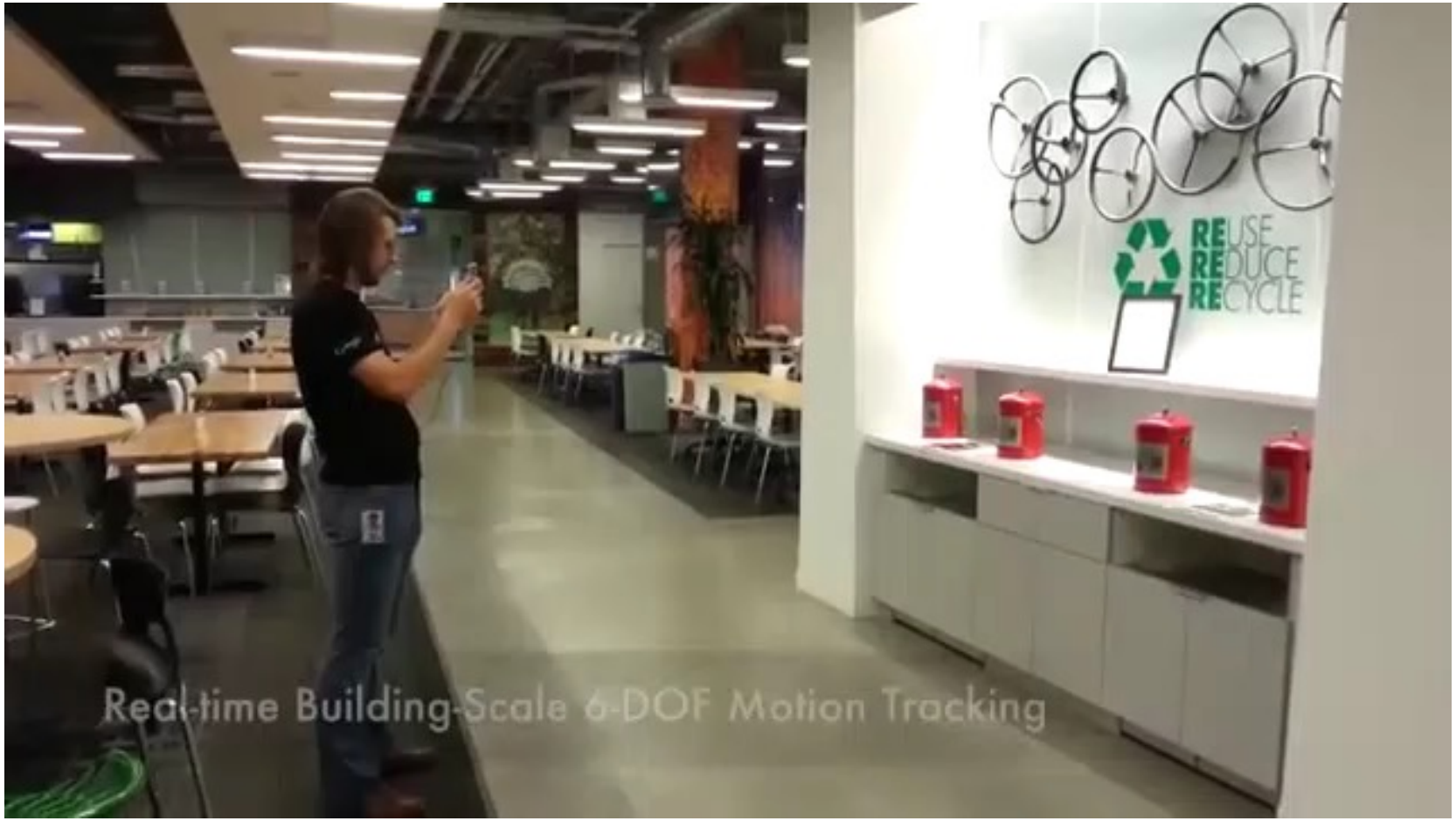
Many Slides from Prof. S. Seitz and Prof. J. B. Huang

SfM Applications – 3D Modeling

SfM Applications – Surveying cultural heritage structure analysis

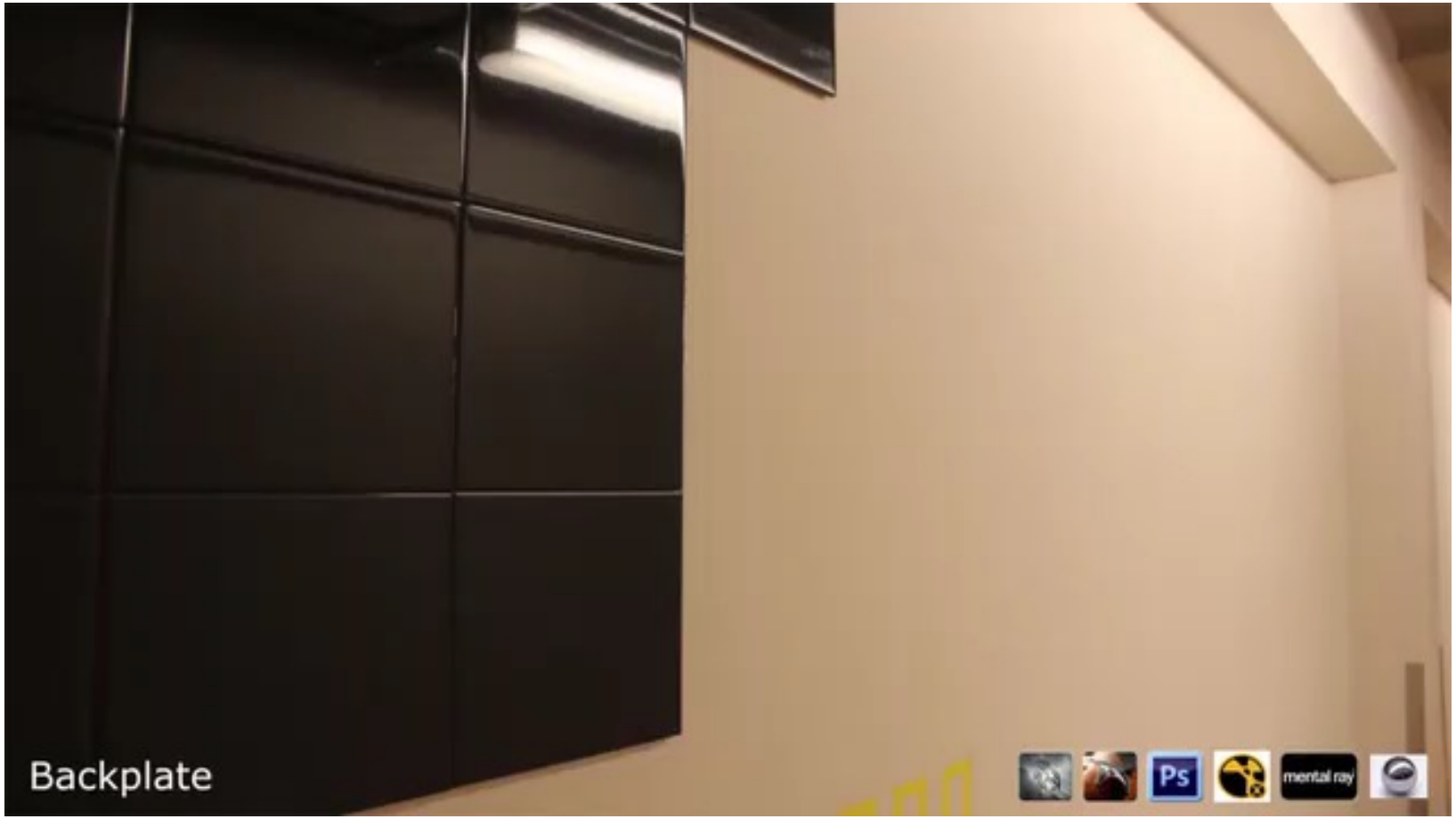
Guidi et al. High-accuracy 3D modeling of cultural heritage, 2004

SfM Applications – Robot navigation and mapmaking



https://www.youtube.com/watch?v=1HhOmF22oYA

SfM Applications – Visual effect (matchmove)



https://www.youtube.com/watch?v=bK6vCPcFkfk

Steps

Images → Points: Structure from Motion

\+

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

Meshes → Models: Texture Mapping

=

Images → Models: Image-based Modeling

Can use Agisoft

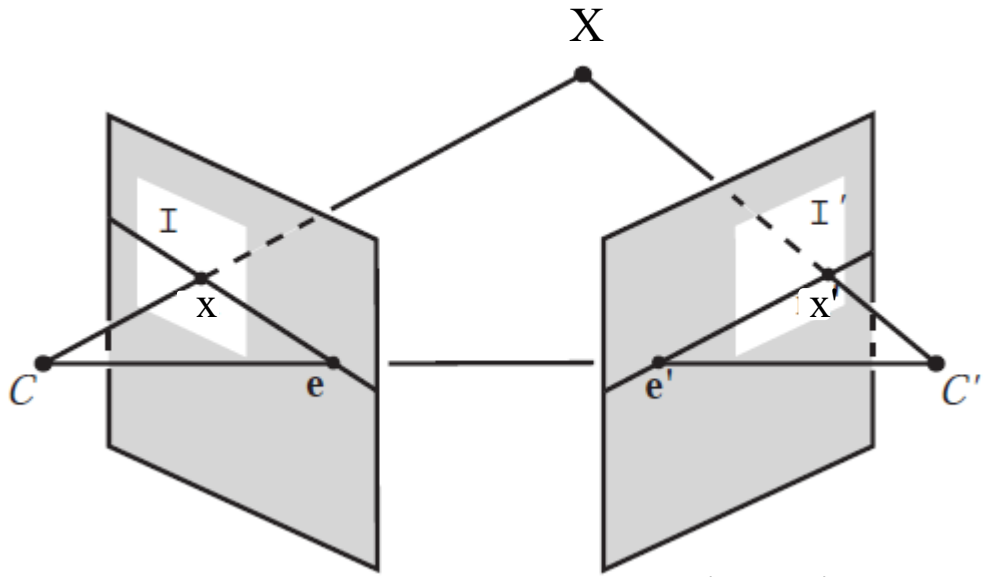
Slide credit: J. Xiao

Triangulation: Linear Solution

- Generally, rays C→x and C'→x' will not exactly intersect
- Can solve via SVD, finding a least squares solution to a system of equations

$\mathbf{x} = (u,v)$, $\mathbf{x}' = (u',v')$: Image points

$\mathbf{X} = (x,y,z)$



$\mathbf{x} = \mathbf{PX}$ $\quad \mathbf{x}' = \mathbf{P}'\mathbf{X}$

$$x(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) = 0$$
$$y(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{2T}\mathbf{X}) = 0$$
$$x(\mathbf{p}^{2T}\mathbf{X}) - y(\mathbf{p}^{1T}\mathbf{X}) = 0$$

where $\mathbf{p}^{iT}$ are the rows of P. These equations are *linear* in the components of $\mathbf{X}$.

$$\mathbf{AX} = \mathbf{0} \quad \mathbf{A} = \begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1^T \\ v\mathbf{p}_3^T - \mathbf{p}_2^T \\ u'\mathbf{p}_3'^T - \mathbf{p}_1'^T \\ v'\mathbf{p}_3'^T - \mathbf{p}_2'^T \end{bmatrix}$$

Further reading: HZ p. 312-313

Triangulation: Linear Solution

$$\mathbf{x} = w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \mathbf{x}' = w\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$$

Given **P**, **P'**, **x**, **x'**

1. Precondition points and projection matrices
2. Create matrix **A**
3. [U, S, V] = svd(A)
4. **X** = V(:, end)

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} \qquad \mathbf{P}' = \begin{bmatrix} \mathbf{p}_1'^T \\ \mathbf{p}_2'^T \\ \mathbf{p}_3'^T \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1^T \\ v\mathbf{p}_3^T - \mathbf{p}_2^T \\ u'\mathbf{p}_3'^T - \mathbf{p}_1'^T \\ v'\mathbf{p}_3'^T - \mathbf{p}_2'^T \end{bmatrix}$$

Pros and Cons

- Works for any number of corresponding images
- Not projectively invariant

Code: http://www.robots.ox.ac.uk/~vgg/hzbook/code/vgg_multiview/vgg_X_from_xP_lin.m

Triangulation: Non-linear Solution

- Minimize projected error while satisfying $\hat{\boldsymbol{x}}'^{T}\boldsymbol{F}\hat{\boldsymbol{x}}=0$

$$cost(\boldsymbol{X}) = dist(\boldsymbol{x}, \hat{\boldsymbol{x}})^2 + dist(\boldsymbol{x}', \hat{\boldsymbol{x}}')^2$$

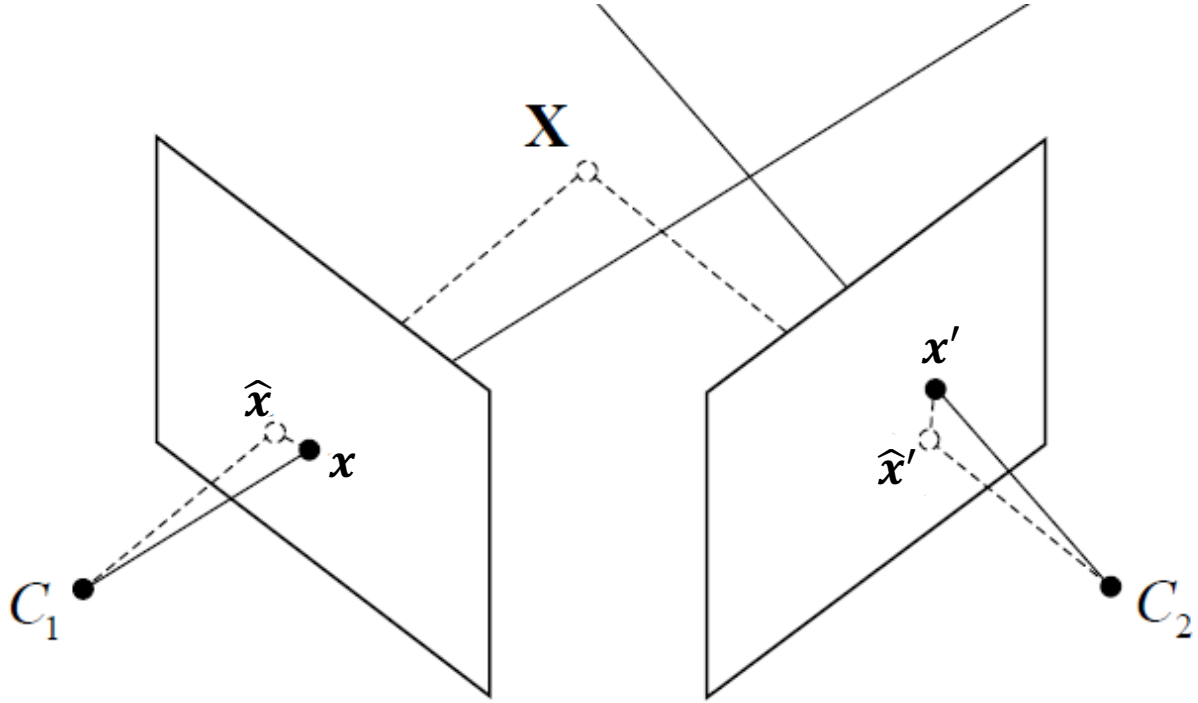


Figure source: Robertson and Cipolla (Chpt 13 of Practical Image Processing and Computer Vision)

Triangulation: Non-linear Solution

- Minimize projected error while satisfying $\hat{\boldsymbol{x}}'^{T}\boldsymbol{F}\hat{\boldsymbol{x}}=0$

$$cost(\boldsymbol{X}) = dist(\boldsymbol{x}, \hat{\boldsymbol{x}})^2 + dist(\boldsymbol{x}', \hat{\boldsymbol{x}}')^2$$

- Solution is a 6-degree polynomial of *t*, minimizing

$$d(\mathbf{x}, \mathbf{l}(t))^2 + d(\mathbf{x}', \mathbf{l}'(t))^2$$

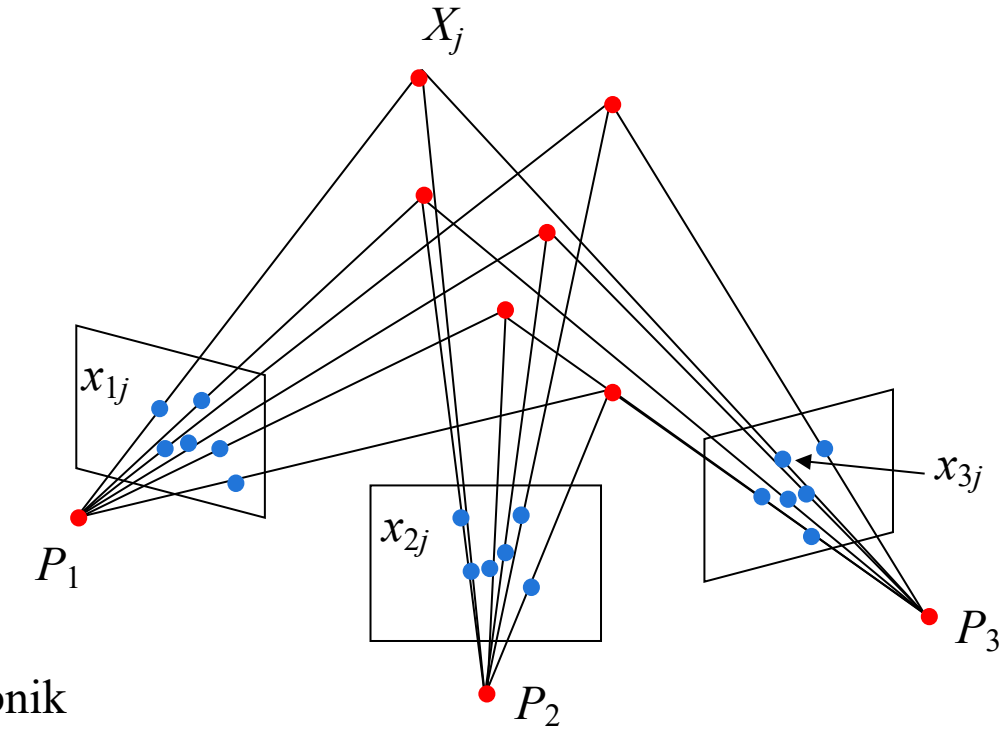
Further reading: HZ p. 318

Projective structure from motion

- Given: m images of n fixed 3D points

$$\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- Problem: estimate m projection matrices $\mathbf{P}_i$ and n 3D points $\mathbf{X}_j$ from the mn corresponding 2D points $\mathbf{x}_{ij}$



Projective structure from motion

- Given: m images of n fixed 3D points

$$\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

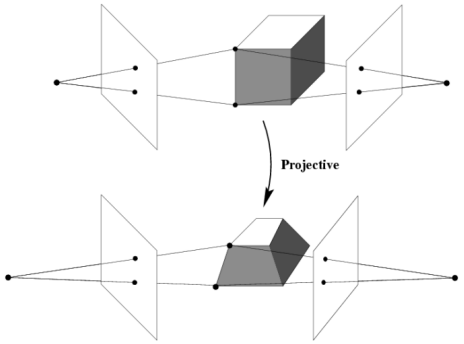
- Problem: estimate m projection matrices $\mathbf{P}_i$ and n 3D points $\mathbf{X}_j$ from the mn corresponding 2D points $\mathbf{x}_{ij}$
- With no calibration info, cameras and points can only be recovered up to a 4x4 projective transformation **Q**:
  - $\mathbf{X} \rightarrow \mathbf{QX}, \mathbf{P} \rightarrow \mathbf{PQ}^{-1}$
- We can solve for structure and motion when

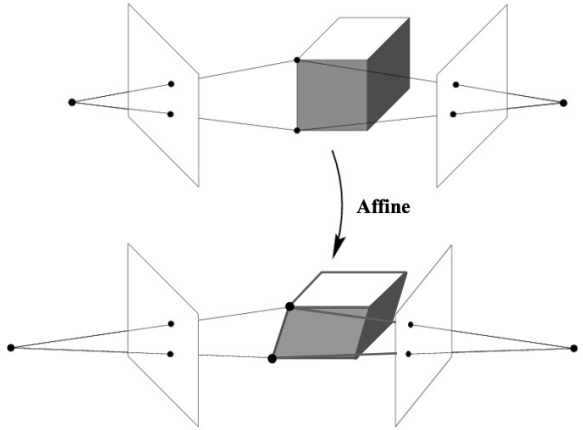
$$2mn >= 11m + 3n - 15$$

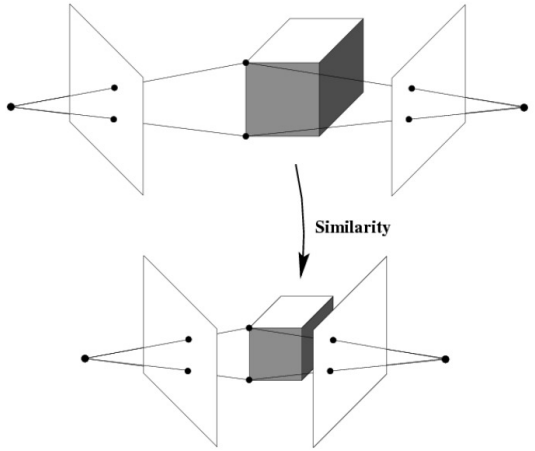
DoF in $\mathbf{P}_i$ | DoF in $\mathbf{X}_j$ | Up to 4x4 projective tform Q

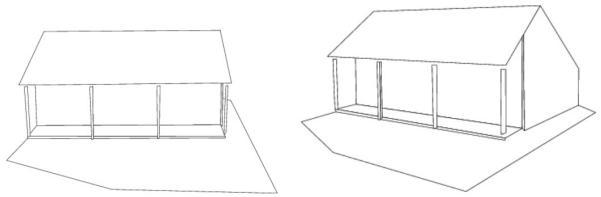
- For two cameras, at least 7 points are needed

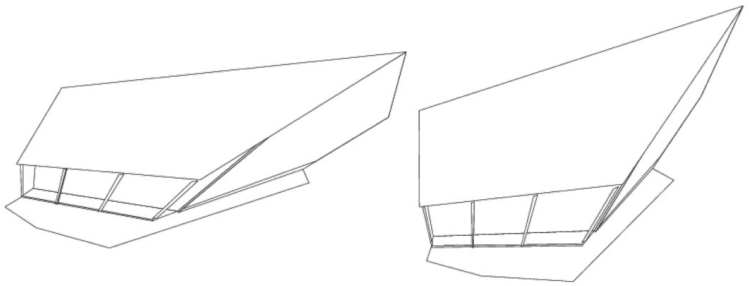
Ambiguities in Q

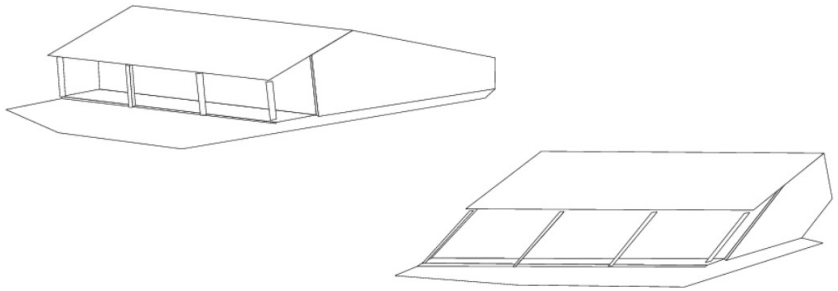


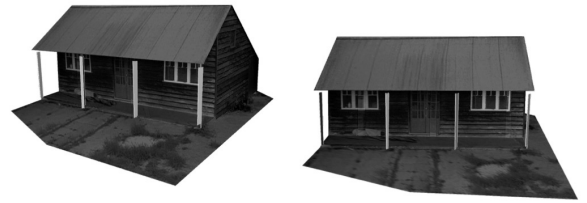












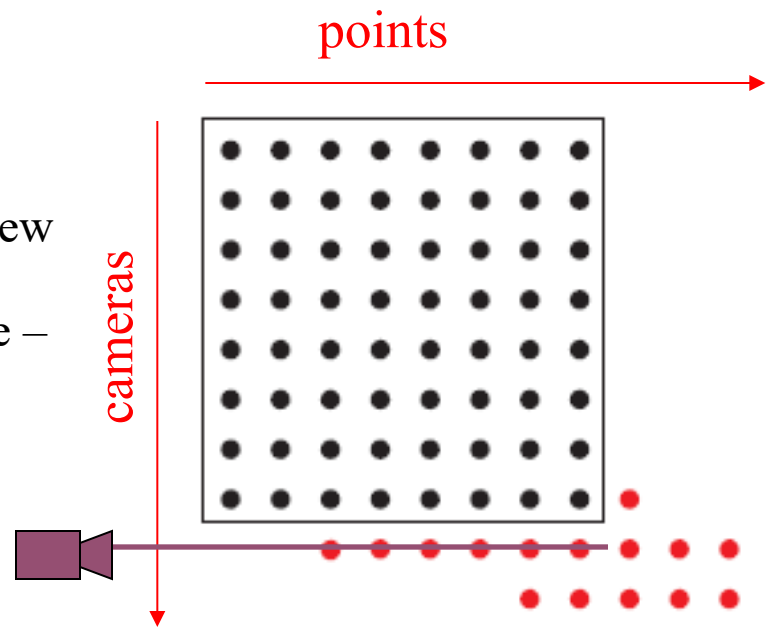
Sequential structure from motion

•Initialize motion (calibration) from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration/resectioning*



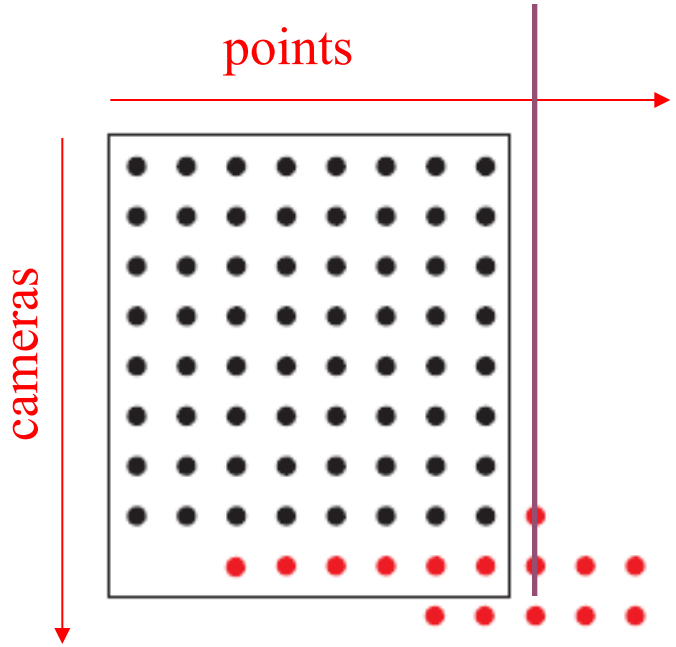
Sequential structure from motion

•Initialize motion from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
- Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*



Sequential structure from motion

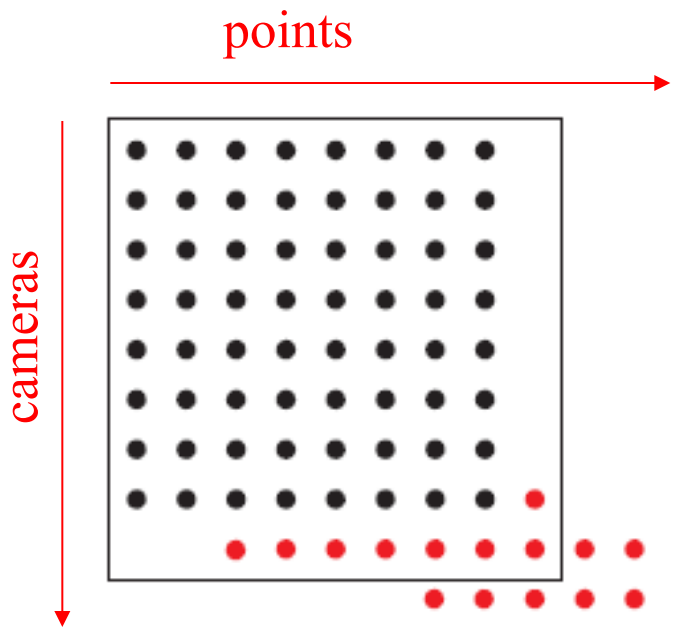
•Initialize motion from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
- Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*

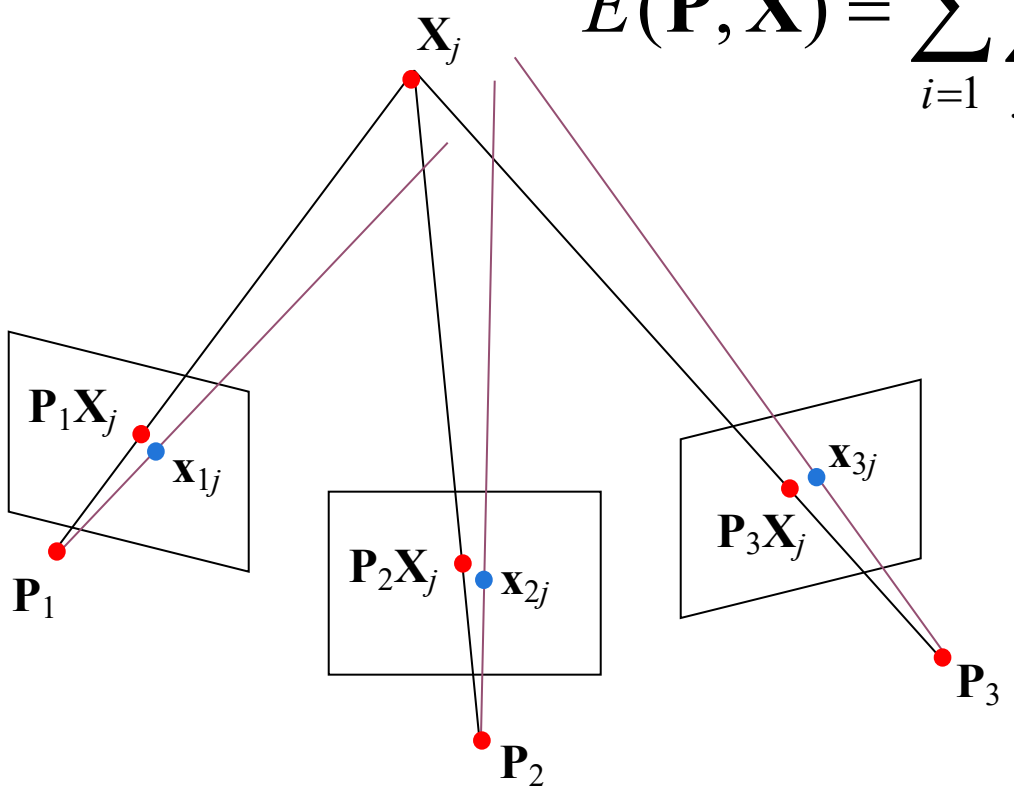
•Refine structure and motion: bundle adjustment



Bundle adjustment

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

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



- Theory:
  The Levenberg–Marquardt algorithm
- Practice:
  The Ceres-Solver from Google

Auto-calibration

- Auto-calibration: determining intrinsic camera parameters directly from uncalibrated images

- For example, we can use the constraint that a moving camera has a fixed intrinsic matrix
  - Compute initial projective reconstruction and find 3D projective transformation matrix **Q** such that all camera matrices are in the form $\mathbf{P}_i = \mathbf{K}[\mathbf{R}_i \mid \mathbf{t}_i]$

- Can use constraints on the form of the calibration matrix, such as zero skew

Summary so far

Projective SFM: Two-camera case

1. Estimate fundamental matrix $\boldsymbol{F}$ between the two views
2. Set first camera matrix to $[\boldsymbol{I} \mid \boldsymbol{0}]$
3. Then the second camera matrix is given by $[\boldsymbol{A} \mid \boldsymbol{t}]$ where $\boldsymbol{t}$ is the epipole ($\boldsymbol{F}^T\boldsymbol{t} = \boldsymbol{0}$) and $\boldsymbol{A} = -[\boldsymbol{t}_\times]\boldsymbol{F}$

- In practice, SFM pipelines use guesses of intrinsic parameters and the five-point algorithm

Slide by Prof. S Lazebnik

Recent work in SfM

- Reconstruct from many images by efficiently finding subgraphs
  - http://www.cs.cornell.edu/projects/matchminer/ (Lou et al. ECCV 2012)

- Improving efficiency of bundle adjustment or
  - http://vision.soic.indiana.edu/projects/disco/ (Crandall et al. ECCV 2011)
  - http://imagine.enpc.fr/~moulonp/publis/iccv2013/index.html (Moulin et al. ICCV 2013)

(best method with software available; also has good overview of recent methods)

Reconstruction of Cornell (Crandall et al. ECCV 2011)

3D from multiple images

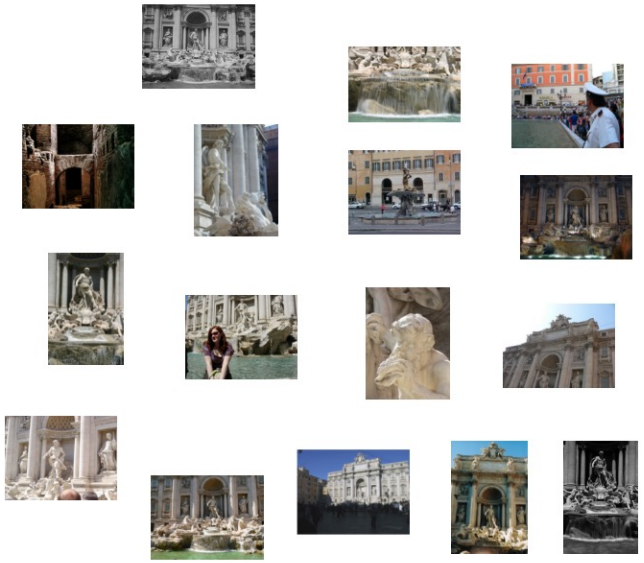
Building Rome in a Day: Agarwal et al. 2009

Modern SfM Pipeline

N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. SIGGRAPH 2006
http://phototour.cs.washington.edu/

Modern SfM Pipeline: Feature Detection

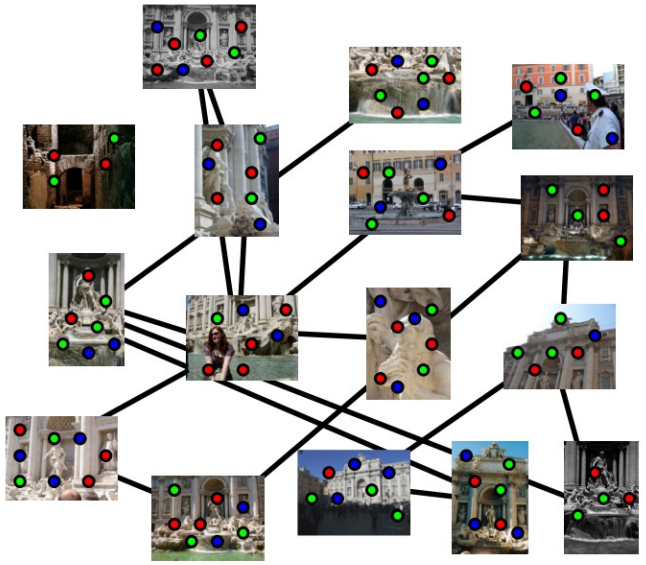
Detect SIFT features



Source: N. Snavely

Modern SfM Pipeline: Feature Matching

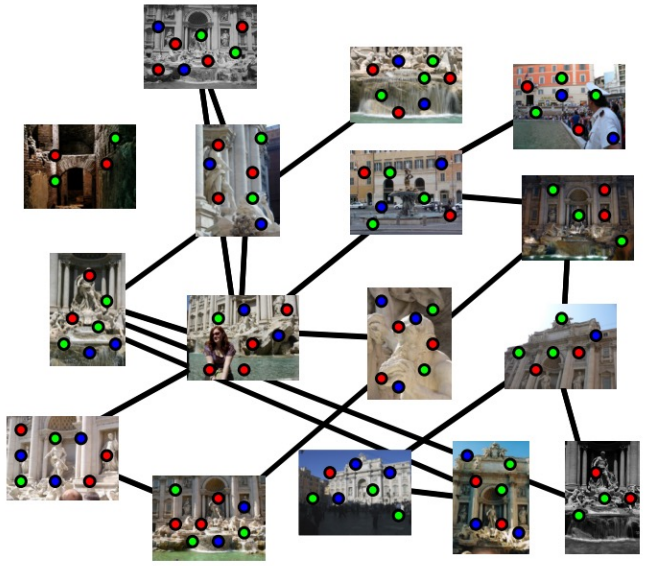
Match features between each pair of images



Source: N. Snavely

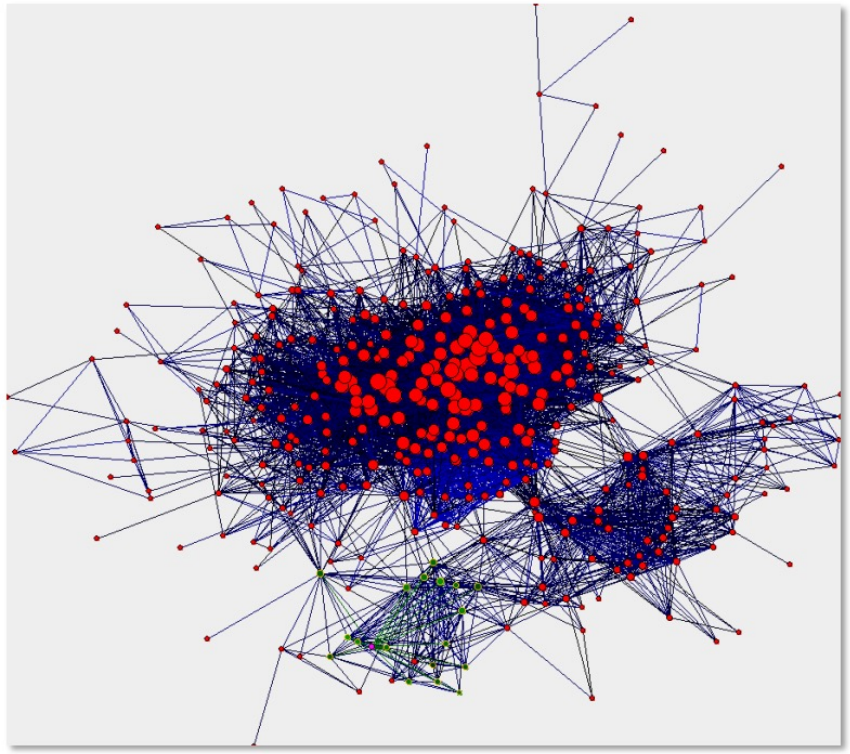
Modern SfM Pipeline: Feature Matching

Use RANSAC to estimate fundamental matrix between each pair



Source: N. Snavely

Modern SfM Pipeline: Image Connectivity Graph



(graph layout produced using the Graphviz toolkit: http://www.graphviz.org/)

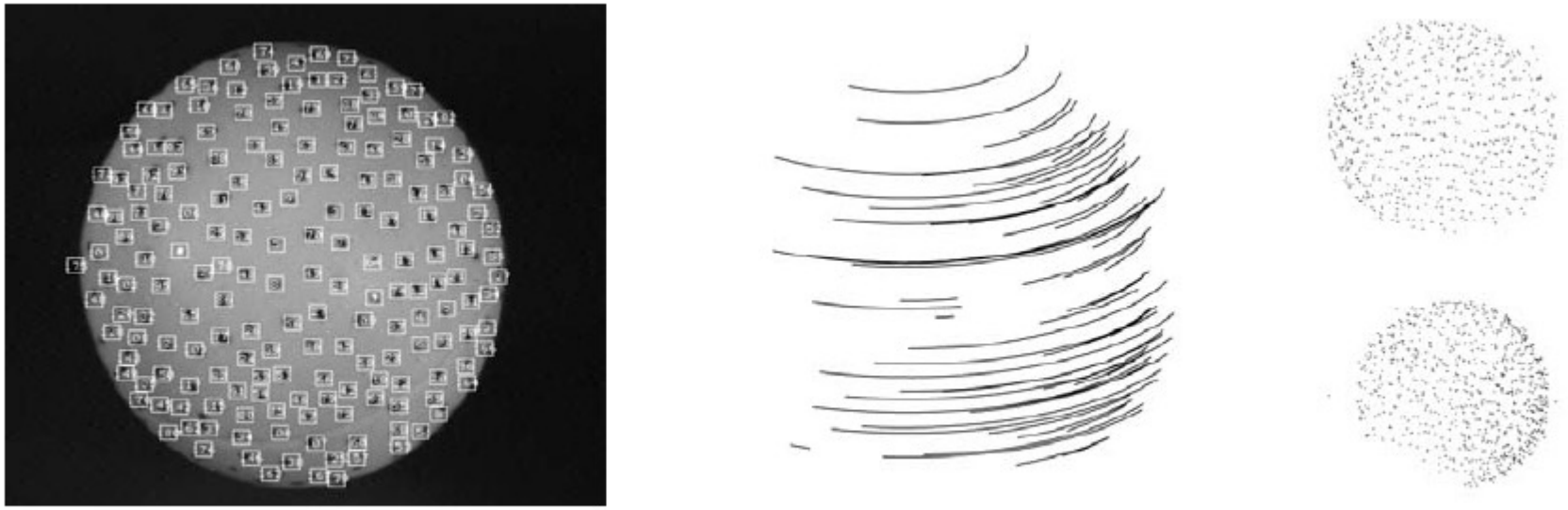
Source: N. Snavely

Modern SfM Pipeline

- Pick a pair of images with lots of inliers (and preferably, good EXIF data)
  - Initialize intrinsic parameters (focal length, principal point) from EXIF
  - Estimate extrinsic parameters ($\boldsymbol{R}$ and $\boldsymbol{t}$) using five-point algorithm
  - Use triangulation to initialize model points
- While remaining images exist
  - Find an image with many feature matches with images in the model
  - Run RANSAC on feature matches to register new image to model
  - Triangulate new points
  - Perform bundle adjustment to re-optimize everything
  - Optionally, align with GPS from EXIF data or ground control points

Slide by Prof. S Lazebnik

Structure from motion under orthographic projection



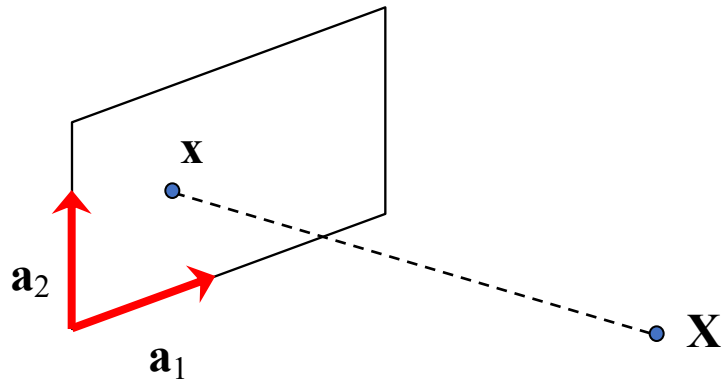
(a) (b) (c)

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.

Orthographic projection for rotated/translated camera

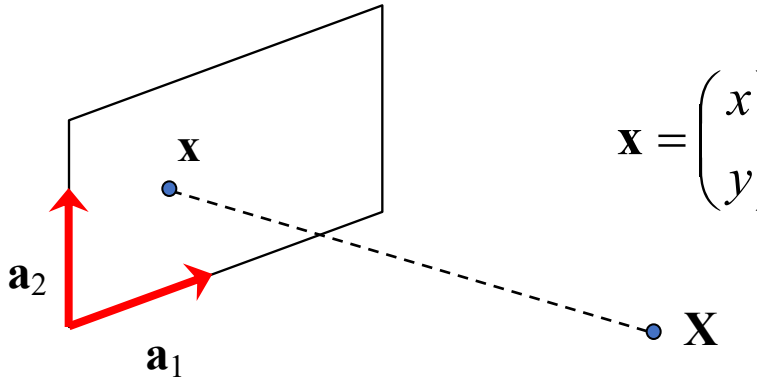


$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(R'_f \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix}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

$$R_f = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R'_f \qquad \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = R_f \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix} + t_f$$

Affine structure from motion

- Affine projection is a linear mapping + translation in in homogeneous 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**)

Step 1: Simplify by getting rid of **t**: shift to centroid of points for each camera

$$\mathbf{x}_i = \mathbf{A}_i\mathbf{X} + \mathbf{t}_i \qquad \hat{\mathbf{x}}_{ij} = \mathbf{x}_{ij} - \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_{ik}$$

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

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

2d normalized point (observed)

3d normalized point

Linear (affine) mapping

Suppose we know 3D points and affine camera parameters …

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

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

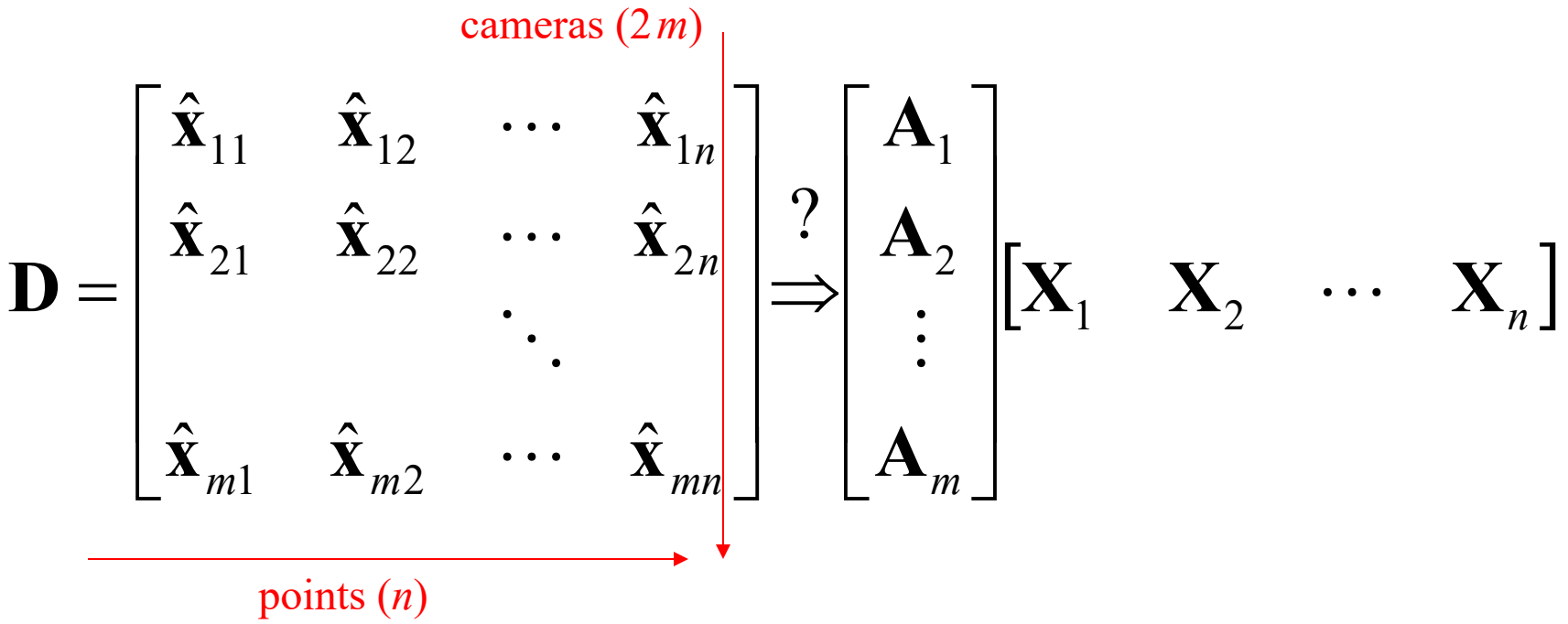
3D Points (3xn)

Camera Parameters (2mx3)

2D Image Points (2mxn)

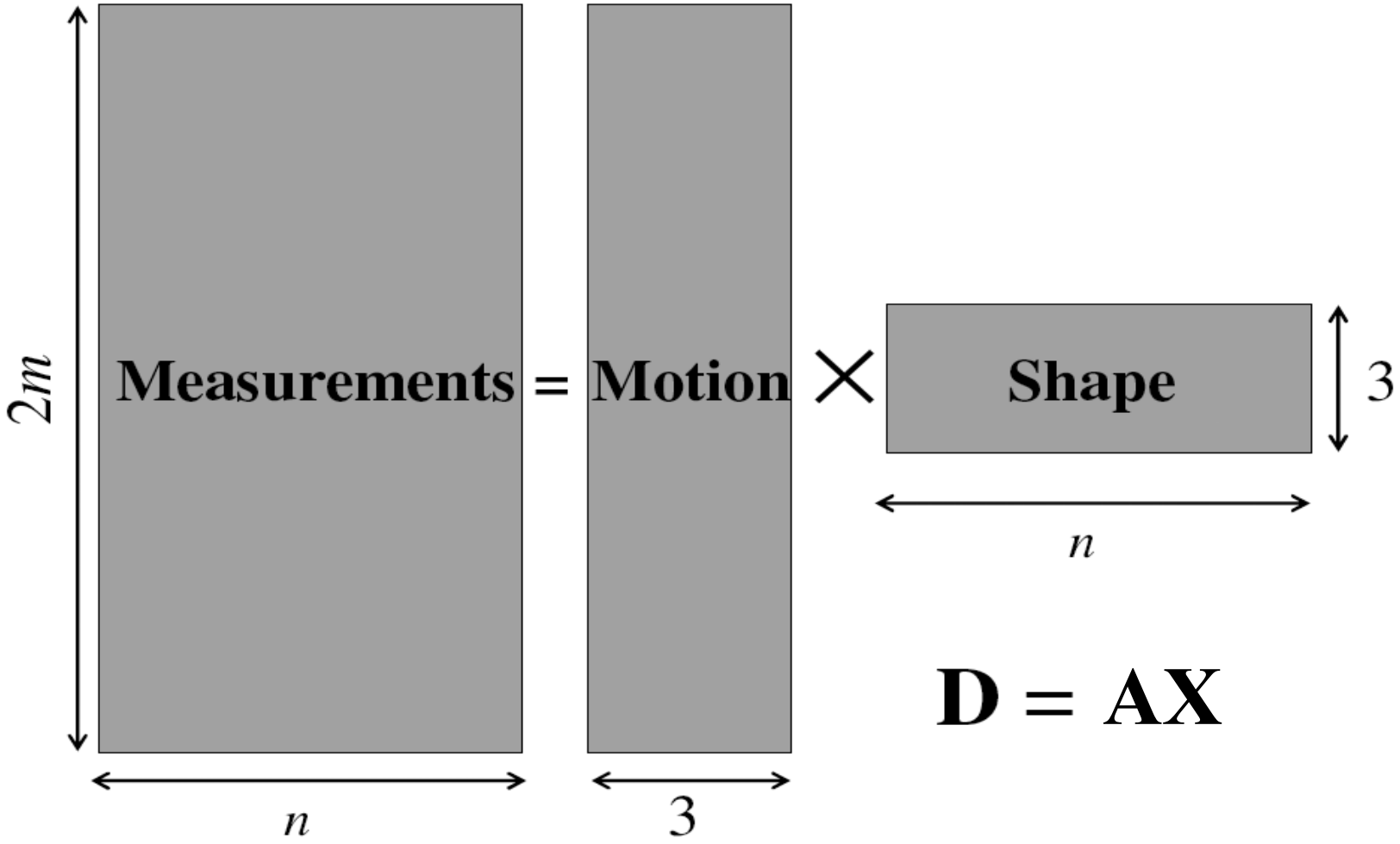
What if we instead observe corresponding 2d image points?

Can we recover the camera parameters and 3d points?



What rank is the matrix of 2D points?

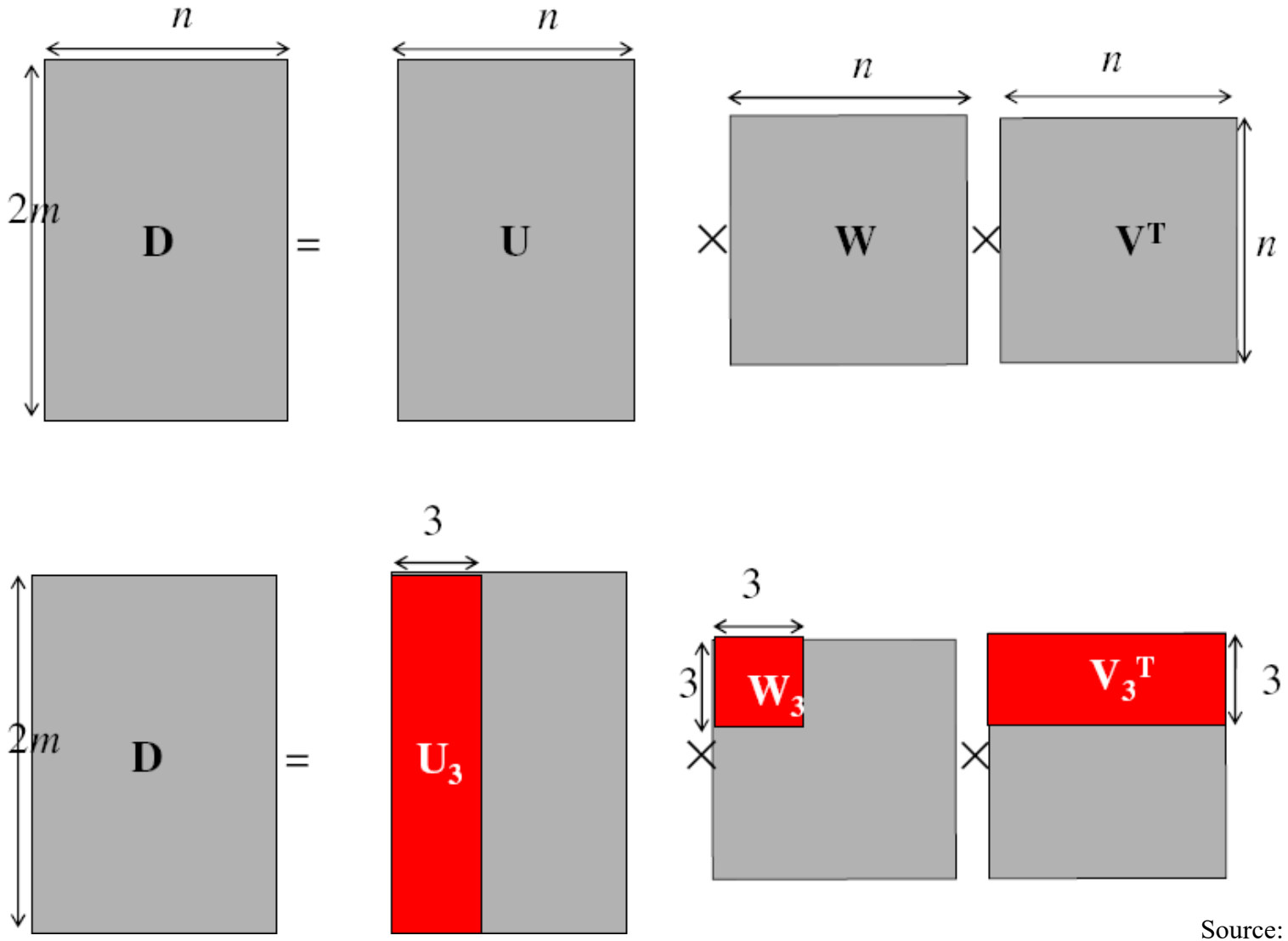
Factorizing the measurement matrix



$$\mathbf{D = AX}$$

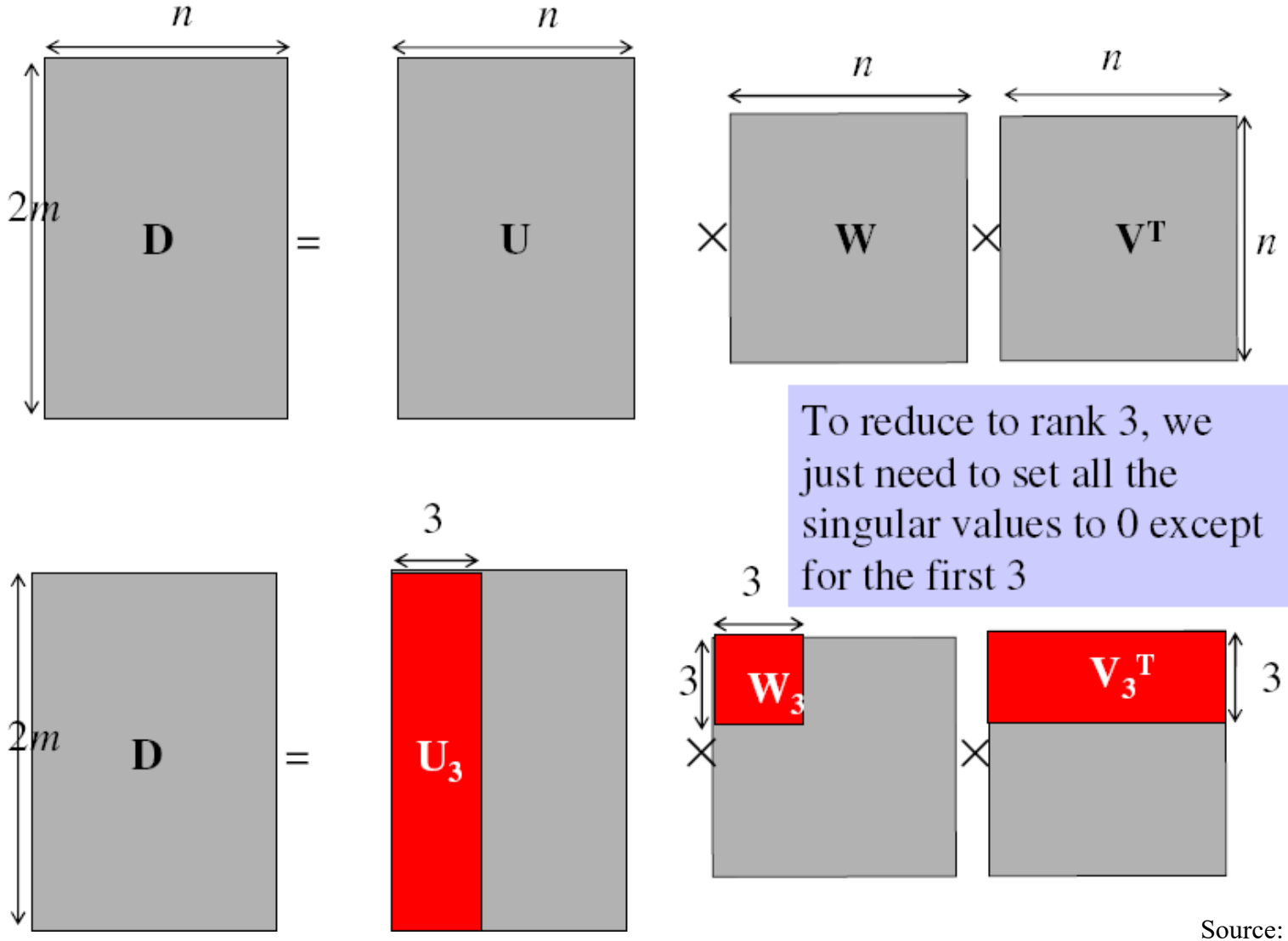
Source: M. Hebert

Factorizing D using SVD



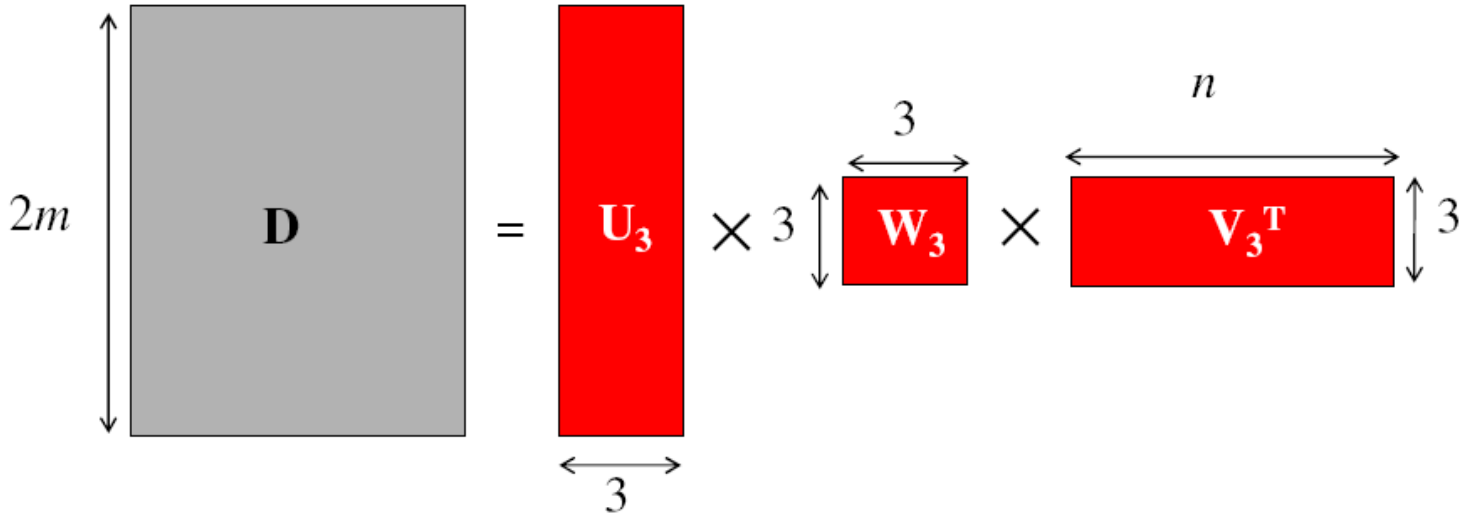
Source: M. Hebert

Factorizing D using SVD



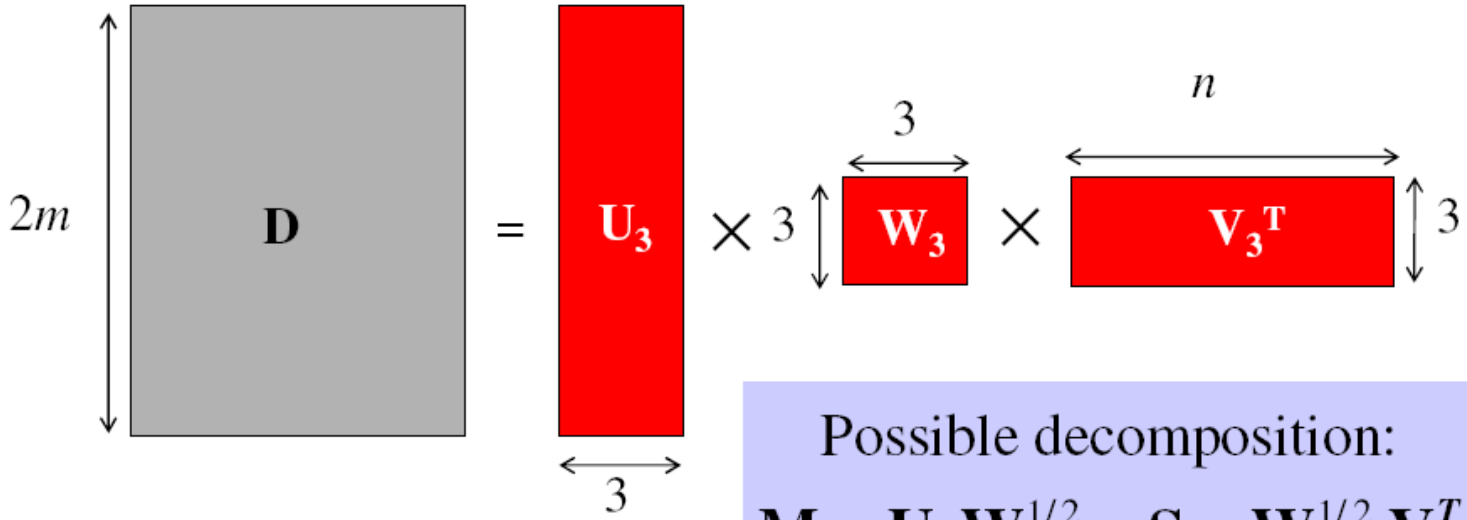
Source: M. Hebert

Factorizing D using SVD



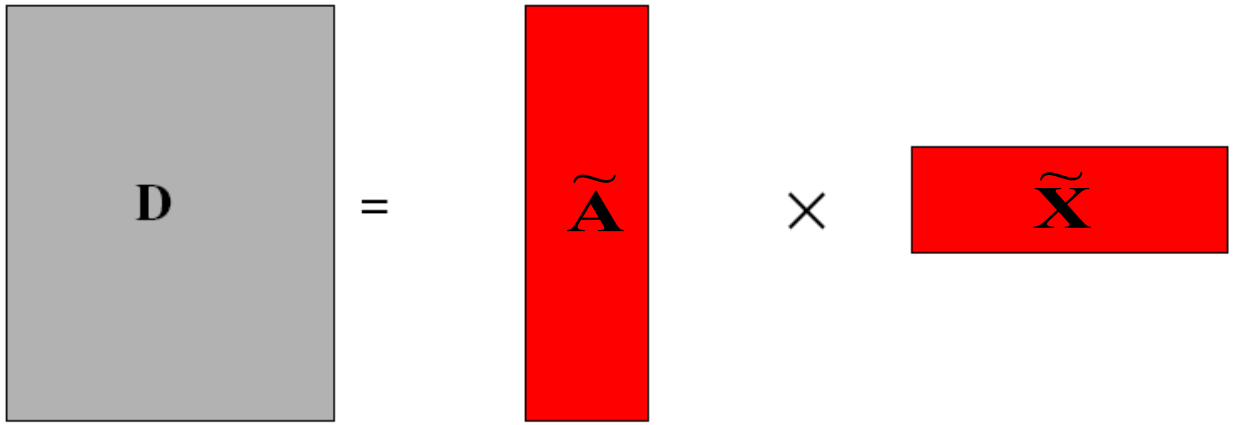
Source: M. Hebert

Factorizing D using SVD



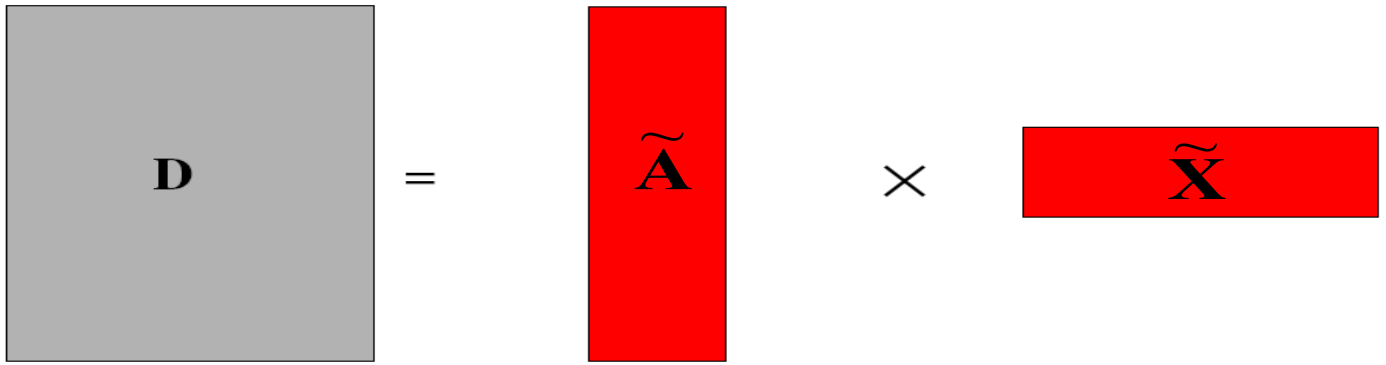
Possible decomposition:

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



Source: M. Hebert

Affine ambiguity

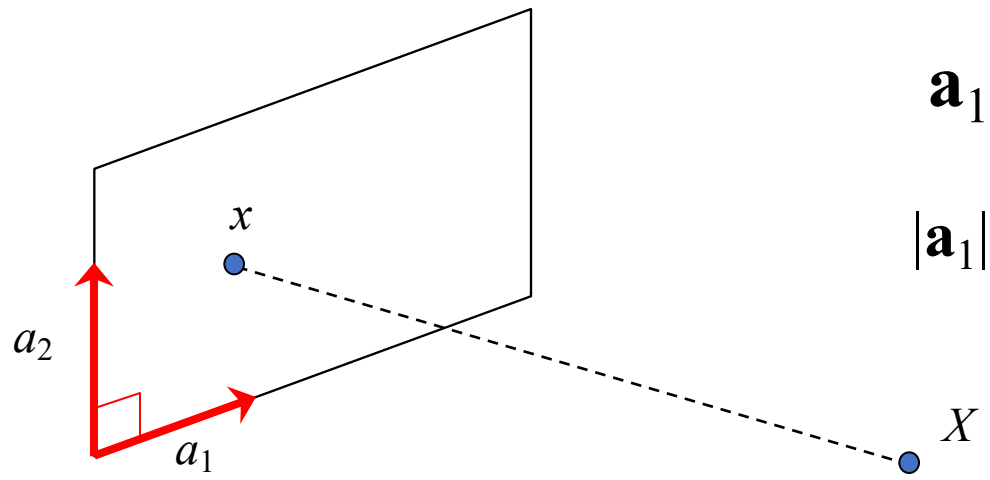


- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations $\mathbf{A} \rightarrow \mathbf{AC}$, $\mathbf{X} \rightarrow \mathbf{C^{-1}X}$
- That is because we have only an affine transformation and we have not enforced any Euclidean constraints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

Source: M. Hebert

Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and of unit length



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

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

Source: M. Hebert

Solve for orthographic constraints

Three equations for each image i

$$\begin{aligned}\tilde{\mathbf{a}}_{i1}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i1} &= 1 \\ \tilde{\mathbf{a}}_{i2}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i2} &= 1 \\ \tilde{\mathbf{a}}_{i1}^T \mathbf{C}\mathbf{C}^T \tilde{\mathbf{a}}_{i2} &= 0\end{aligned} \quad \text{where} \quad \tilde{\mathbf{A}}_i = \begin{bmatrix} \tilde{\mathbf{a}}_{i1}^T \\ \tilde{\mathbf{a}}_{i2}^T \end{bmatrix}$$

- Solve for $\mathbf{L} = \mathbf{C}\mathbf{C}^\mathbf{T}$
- Recover $\mathbf{C}$ from $\mathbf{L}$ by Cholesky decomposition: $\mathbf{L} = \mathbf{C}\mathbf{C}^\mathbf{T}$
- Update $\mathbf{A}$ and $\mathbf{X}$: $\mathbf{A} = \mathbf{A}\mathbf{C}$, $\mathbf{X} = \mathbf{C}^{-1}\mathbf{X}$

How to solve $\mathbf{L = CC^T}$?

$$[a \quad b \quad c]\begin{bmatrix} L_{11} & L_{21} & L_{31} \\ L_{12} & L_{22} & L_{32} \\ L_{13} & L_{23} & L_{33} \end{bmatrix}\begin{bmatrix} d \\ e \\ f \end{bmatrix} = k$$

$$[ad \;\; bd \;\; cd \;\; ae \;\; be \;\; ce \;\; af \;\; bf \;\; cf]\begin{bmatrix} L_{11} \\ L_{12} \\ L_{13} \\ L_{21} \\ L_{22} \\ L_{23} \\ L_{31} \\ L_{32} \\ L_{33} \end{bmatrix} = \mathrm{k}$$

reshape([a b c]'*[d e f], [1, 9])

Algorithm summary

- Given: m images and n tracked features $\mathbf{x}_{ij}$
- For each image i, center the feature coordinates
- Construct a $2m \times n$ measurement matrix $\mathbf{D}$:
  - 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 $\mathbf{D}$:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^{\mathbf{T}}$
  - Create $\mathbf{U}_3$ by taking the first 3 columns of $\mathbf{U}$
  - Create $\mathbf{V}_3$ by taking the first 3 columns of $\mathbf{V}$
  - Create $\mathbf{W}_3$ by taking the upper left 3×3 block of $\mathbf{W}$
- Create the motion (affine) and shape (3D) matrices:

  $$\mathbf{A} = \mathbf{U}_3\mathbf{W}_3^{1/2} \text{ and } \mathbf{S} = \mathbf{W}_3^{1/2}\,\mathbf{V}_3^{\mathrm{T}}$$

- Eliminate affine ambiguity
  - Solve $\mathbf{L} = \mathbf{C}\mathbf{C}^{\mathbf{T}}$ using metric constraints
  - Solve $\mathbf{C}$ using Cholesky decomposition
  - Update $\mathbf{A}$ and $\mathbf{X}$: $\mathbf{A} = \mathbf{A}\mathbf{C}$, $\mathbf{S} = \mathbf{C}^{-1}\mathbf{S}$

Source: M. Hebert

The Reading List (optional)

- "A computer algorithm for reconstructing a scene from two images", Longuet-Higgins, Nature 1981
- "Shape and motion from image streams under orthography: A factorization method." C. Tomasi and T. Kanade, *IJCV*, 9(2):137-154, November 1992
- "In defense of the eight-point algorithm", Hartley, PAMI 1997
- "Building Rome in a day", Agarwal et al., ICCV 2009
- https://www.youtube.com/watch?v=kyIzMr917Rc, 3D Computer Vision: Past, Present, and Future