03/07/17

Structure from Motion

Computer Vision

CS 543 / ECE 549

University of Illinois

Derek Hoiem

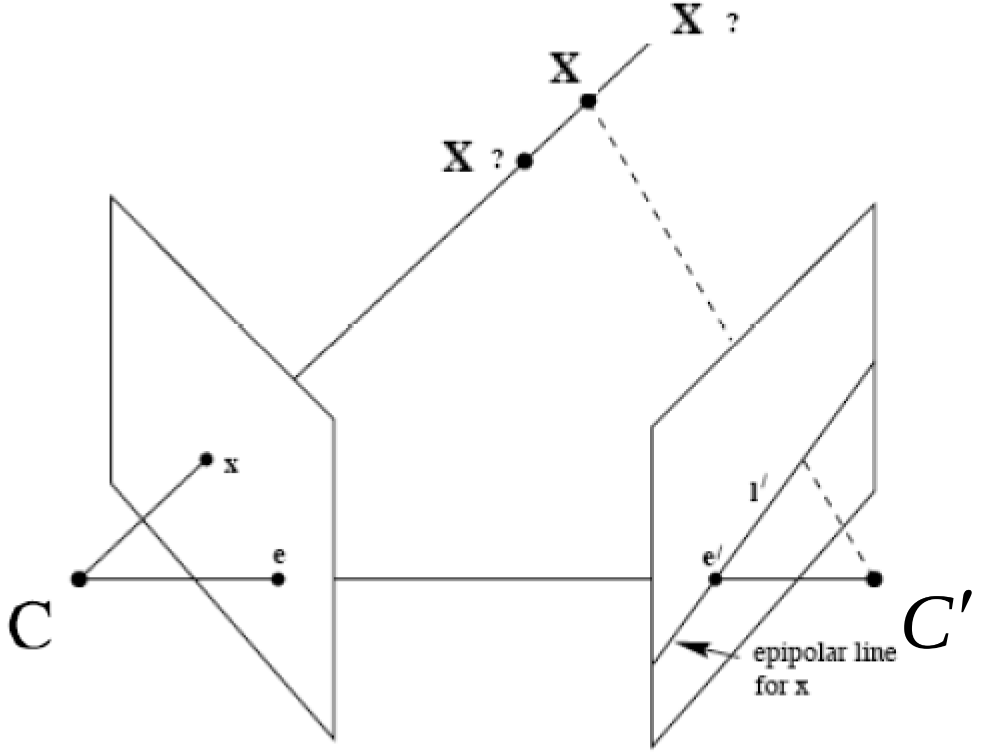
Affine SfM slides adapted from Martial Hebert

This class: structure from motion

- Incremental perspective structure from motion
- Global affine structure from motion

Last Class: Epipolar Geometry

- 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



Last Class: 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:

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

Incremental Structure from Motion (SfM)

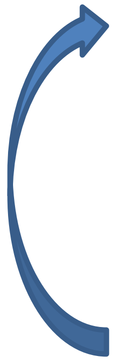
Goal: Solve for camera poses and 3D points in scene

Incremental SfM

1. Compute features

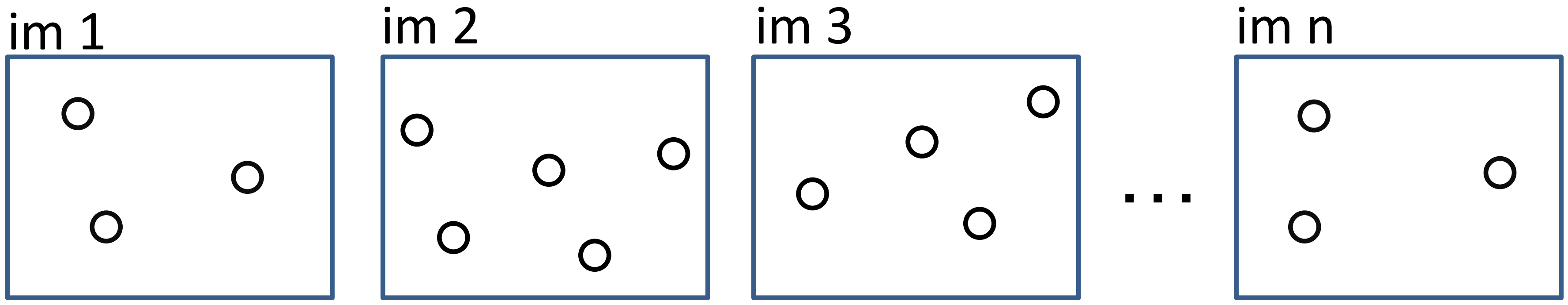
2. Match images

3. Reconstruct
    a) Solve for pose and 3D points in two cameras
    b) Solve for pose of additional camera(s) that observe reconstructed 3D points
    c) Solve for new 3D points that are viewed in at least two cameras
    d) Bundle adjust to minimize reprojection error



Incremental SFM: **detect features**

- Feature types: SIFT, ORB, Hessian-Laplacian, …



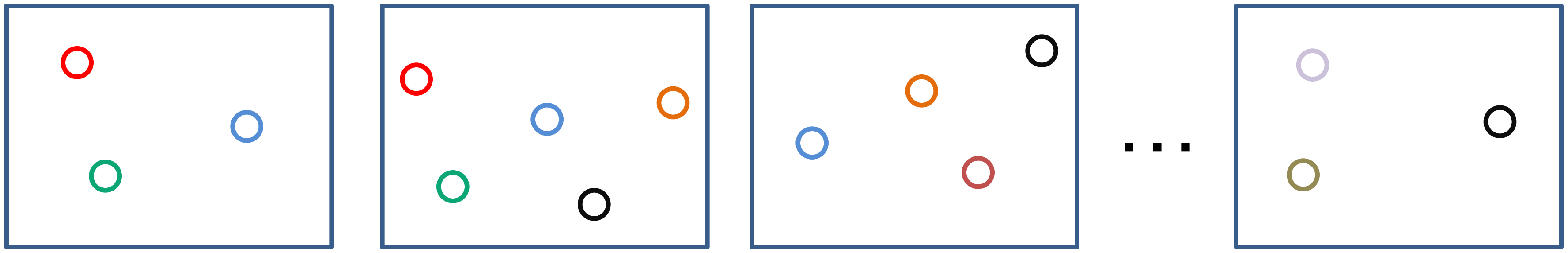
Each circle represents a set of detected features

Incremental SFM: **match features and images**

For each pair of images:

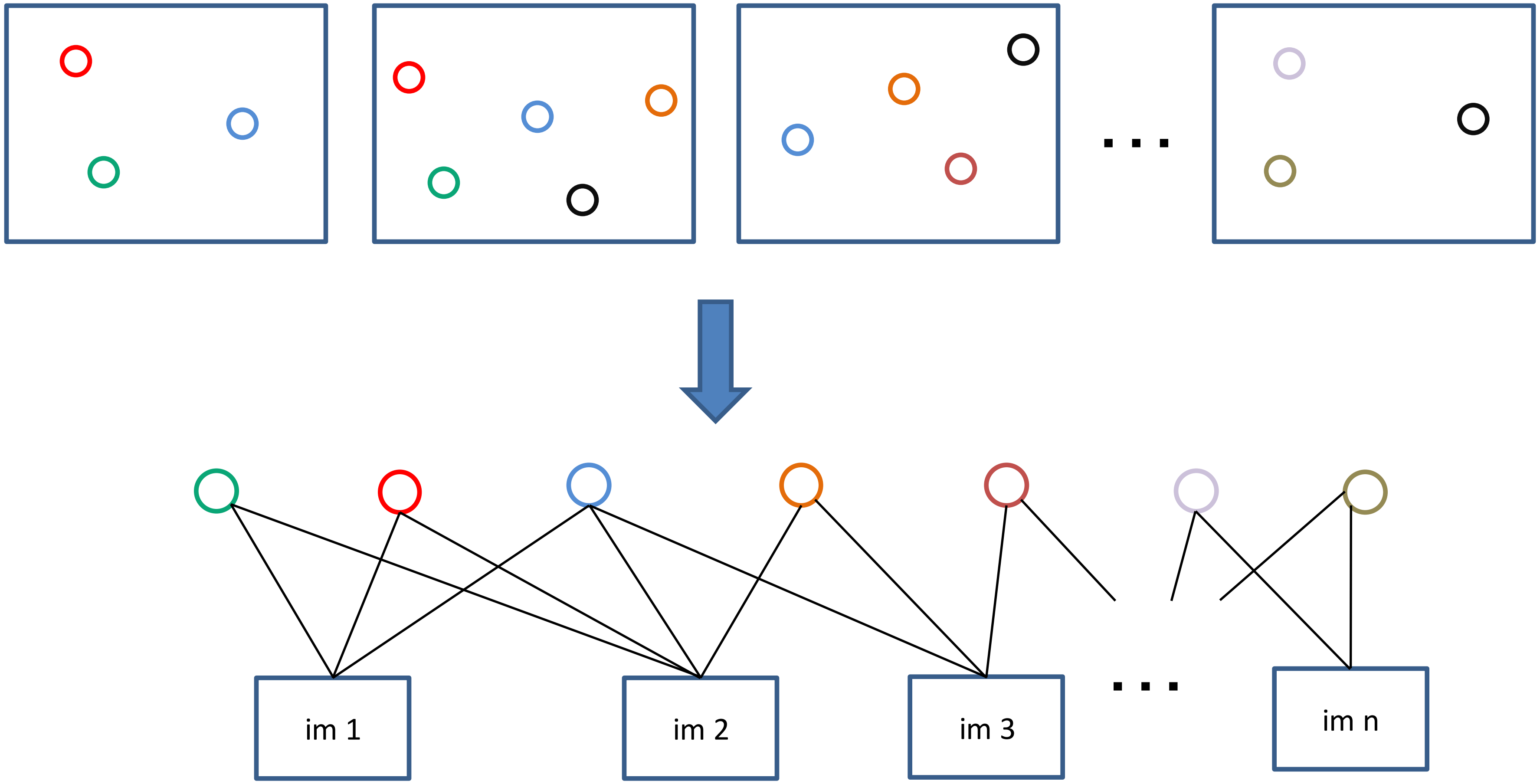
1. Match feature descriptors via approximate nearest neighbor
2. Solve for F and find inlier feature correspondences

- Speed tricks
  - Match only 100 largest features first
  - Use a bag-of-words method to find candidate matches
  - Perform initial filtering based on GPS coordinates, if available
  - Use known matches to predict new ones



Points of same color have been matched to each other

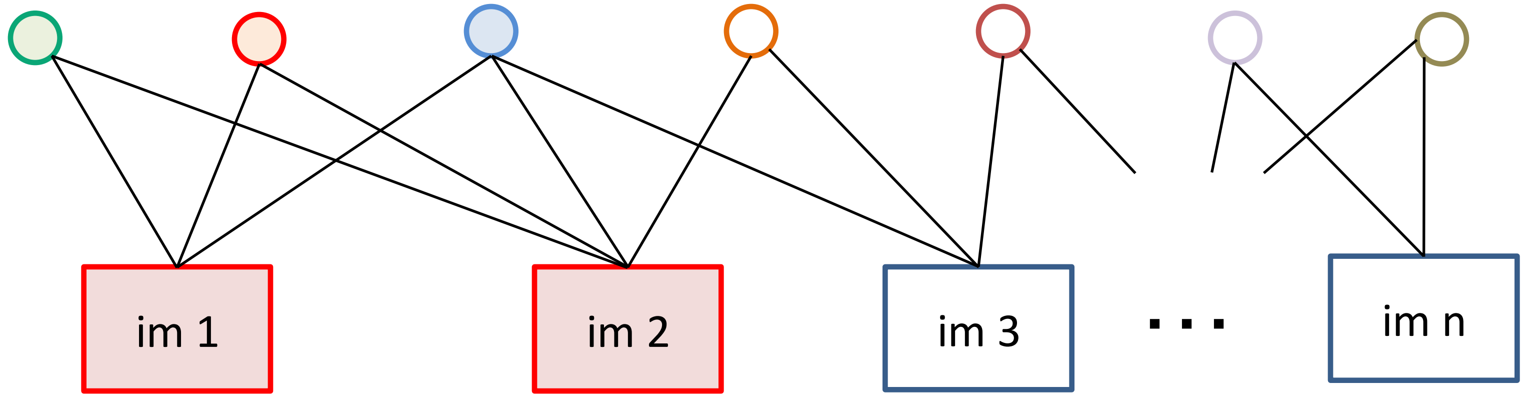
Incremental SFM: **create tracks graph**



tracks graph: bipartite graph between observed 3D points and images

Incremental SFM: **initialize reconstruction**

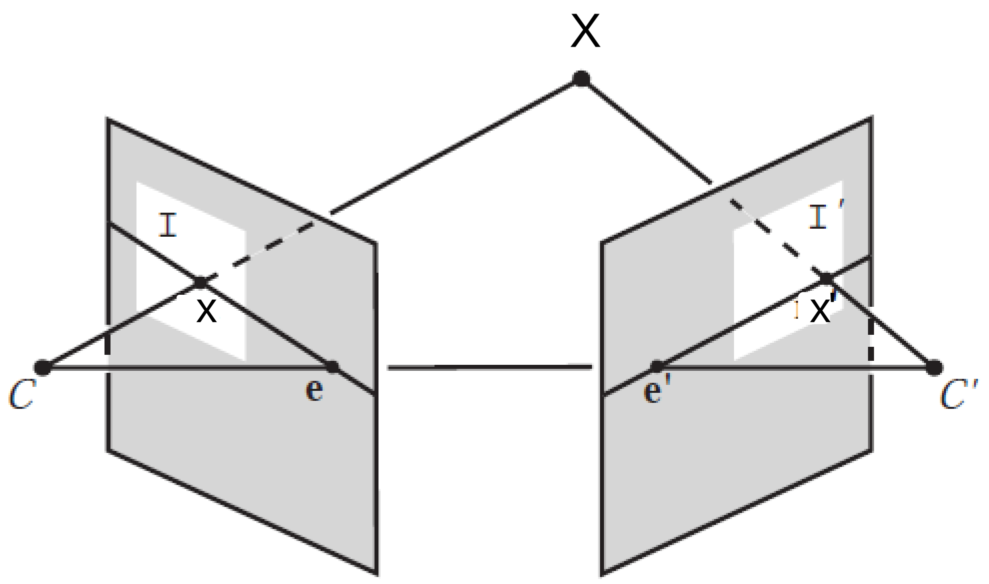
1. Choose two images that are likely to provide a stable estimate of relative pose
   - E.g., $\frac{\#\text{ inliers for }H}{\#\text{ inliers for }F} < 0.7$ and many inliers for F
2. Get focal lengths from EXIF, estimate essential matrix using 5-point algorithm, extract pose R_2, t_2 with $R_1 = \boldsymbol{I}, t_1 = \boldsymbol{0}$
3. Solve for 3D points given poses
4. Perform bundle adjustment to refine points and poses



filled circles = "triangulated" points
filled rectangles = "resectioned" images (solved pose)

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



$$\mathbf{x} = \mathbf{P}\mathbf{X} \qquad \mathbf{x}' = \mathbf{P}'\mathbf{X}$$

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

Further reading: HZ p. 312-313

Triangulation: Linear Solution

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)

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

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

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

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 $\hat{\boldsymbol{x}}'^{T}\boldsymbol{F}\hat{\boldsymbol{x}}=0$

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

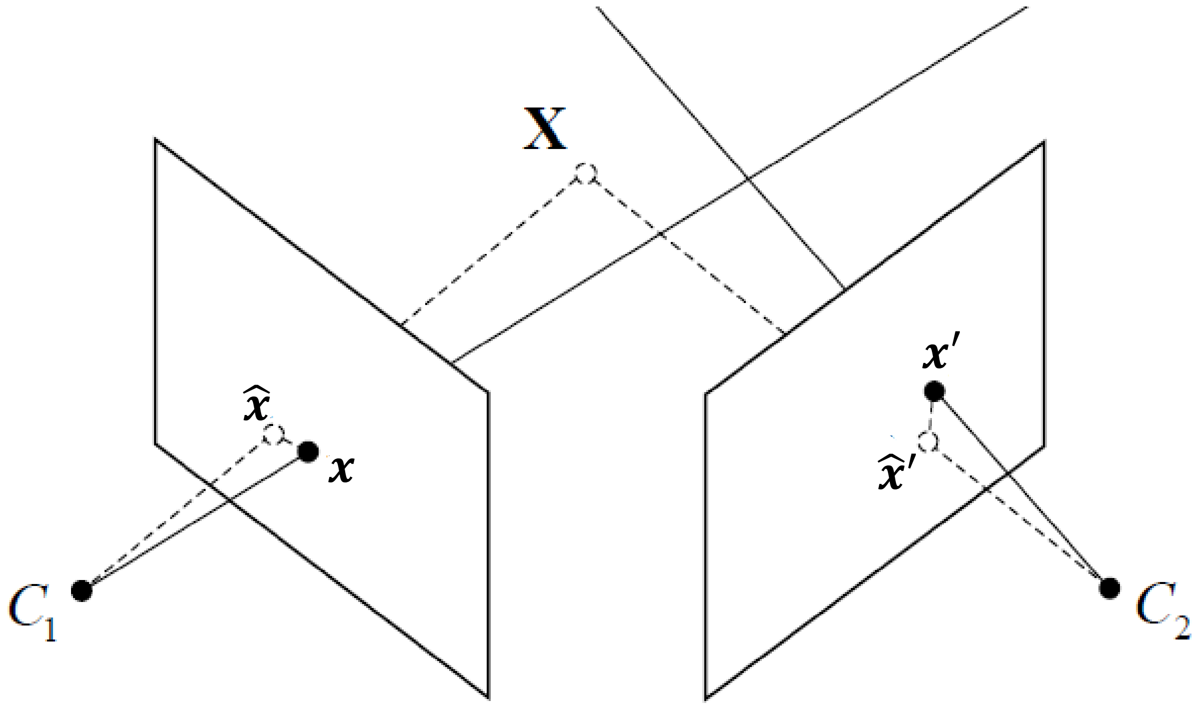
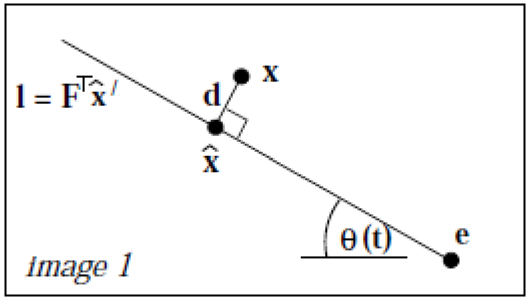


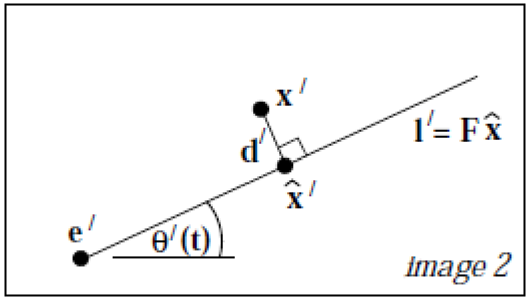
Figure source: Robertson and Cipolla (Chpt 13 of Practical Image Processing and Computer Vision)

Triangulation: Non-linear Solution

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

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





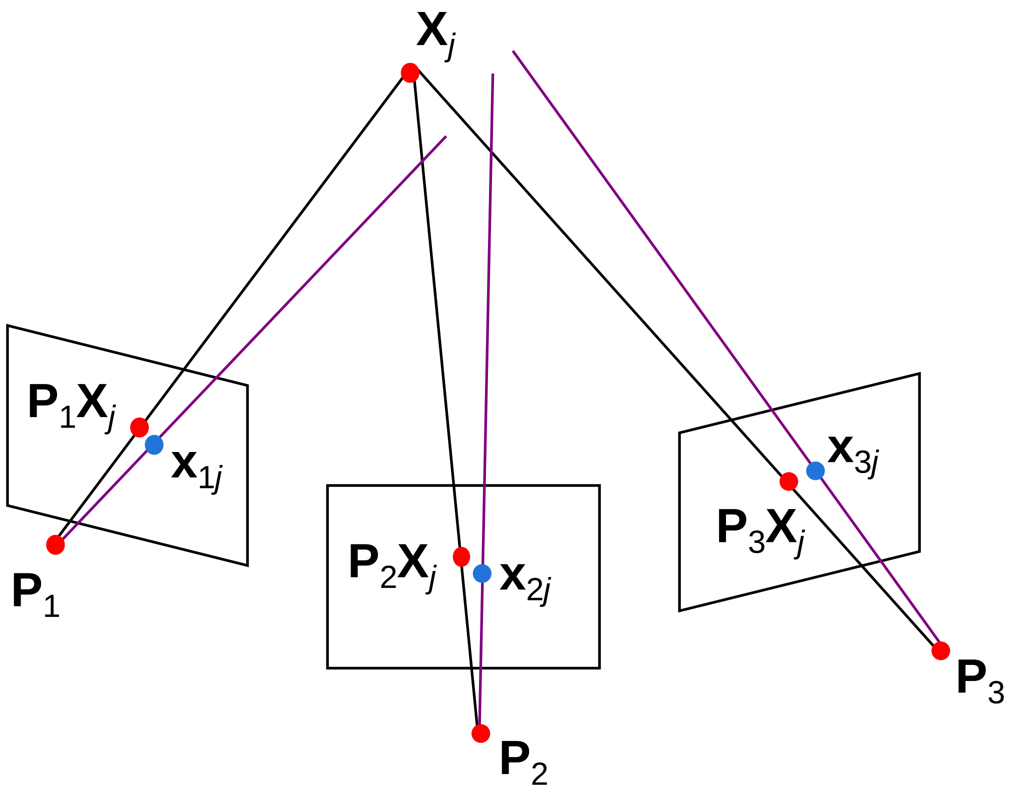
- Solution is a 6-degree polynomial of *t*, minimizing $d(\mathbf{x}, \mathbf{l}(t))^2 + d(\mathbf{x}', \mathbf{l}'(t))^2$

Further reading: HZ p. 318

Bundle adjustment

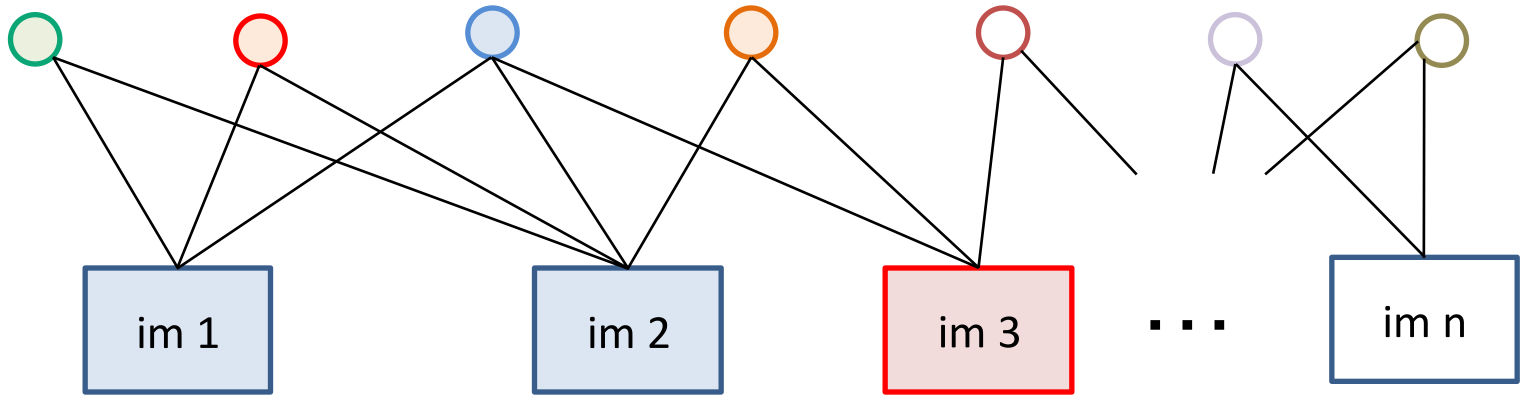
- Non-linear method for refining structure and motion
- Minimizing reprojection error

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



Incremental SFM: **grow reconstruction**

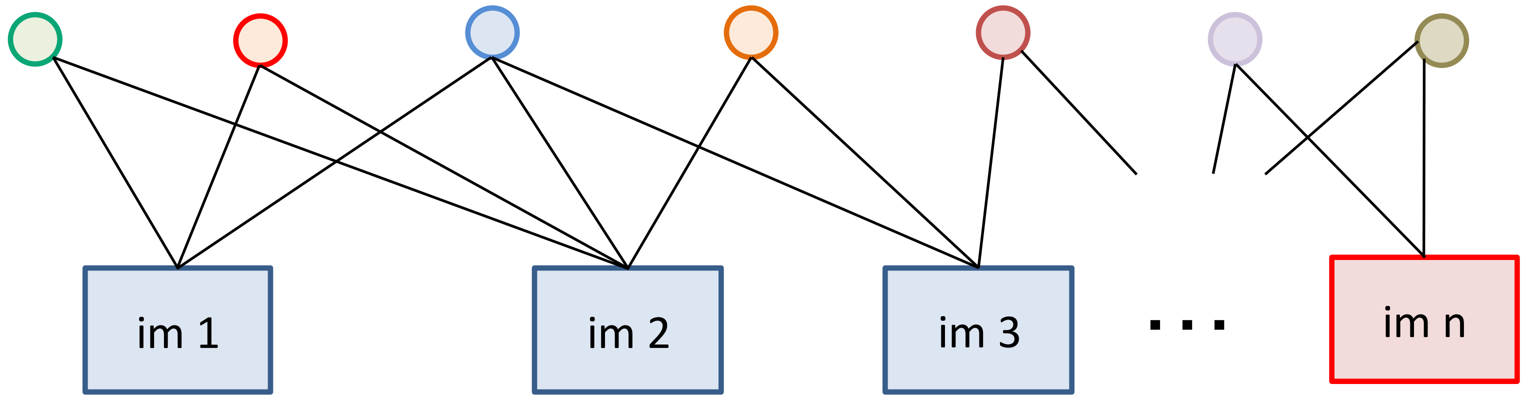
1. Resection: solve pose for image(s) that have the most triangulated points
2. Triangulate: solve for any new points that have at least two cameras
3. Remove 3D points that are outliers
4. Bundle adjust
   - For speed, only do full bundle adjust after some percent of new images are resectioned
5. Optionally, align with GPS from EXIF or ground control points (GCP)



filled circles = “triangulated” points
filled rectangles = “resectioned” images (solved pose)

Incremental SFM: **grow reconstruction**

1. Resection: solve pose for image(s) that have the most triangulated points
2. Triangulate: solve for any new points that have at least two cameras
3. Remove 3D points that are outliers
4. Bundle adjust
   - For speed, only do full bundle adjust after some percent of new images are resectioned
5. Optionally, align with GPS from EXIF or ground control points (GCP)



filled circles = “triangulated” points
filled rectangles = “resectioned” images (solved pose)

Important recent papers and methods for SfM

- OpenMVG
  - https://github.com/openMVG/openMVG
  - http://imagine.enpc.fr/~moulonp/publis/iccv2013/index.html (Moulin et al. ICCV 2013)
  - Software has global and incremental methods

- OpenSfM (software only): https://github.com/mapillary/OpenSfM
  - Basis for my description of incremental SfM

- Visual SfM: Visual SfM (Wu 2013)
  - Used to be the best incremental SfM software (but not anymore and closed source); paper still very good

Reconstruction of Cornell (Crandall et al. ECCV 2011)

Multiview Stereo (MVS)

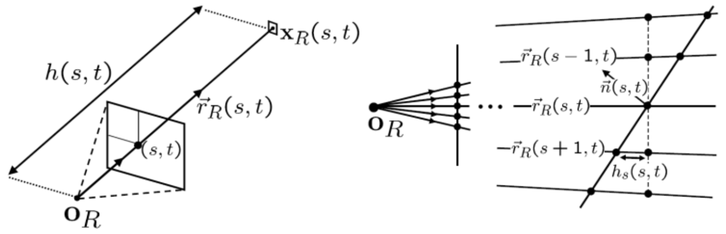
"Multiview Stereo: a tutorial" by Yasu
Furukawa
http://www.cse.wustl.edu/~furukawa/papers/fnt_mvs.pdf

Software:

- MVE: https://github.com/simonfuhrmann/mve

Main ideas:

- Initialize with SfM
- MVS: For each image, find 2+ other images with similar viewpoints but substantial baselines
  - Grow regions from sparse points in SfM
  - Create a patch around each pixel and solve for depth, surface normal, and relative intensity that is consistent with all images



Surface Reconstruction

Floating scale surface reconstruction:

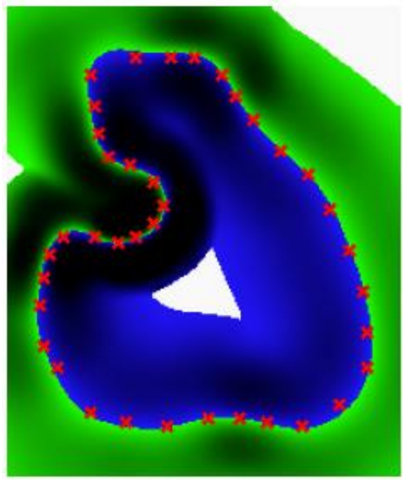
http://www.gcc.tu-darmstadt.de/home/proj/fssr/

Software:

- MVE:
  https://github.com/simonfuhrmann/mve

Main ideas:

- Initialize with MVS
- Merge 3D points from all depth images
- Estimate implicit surface function in octree and find zero crossings

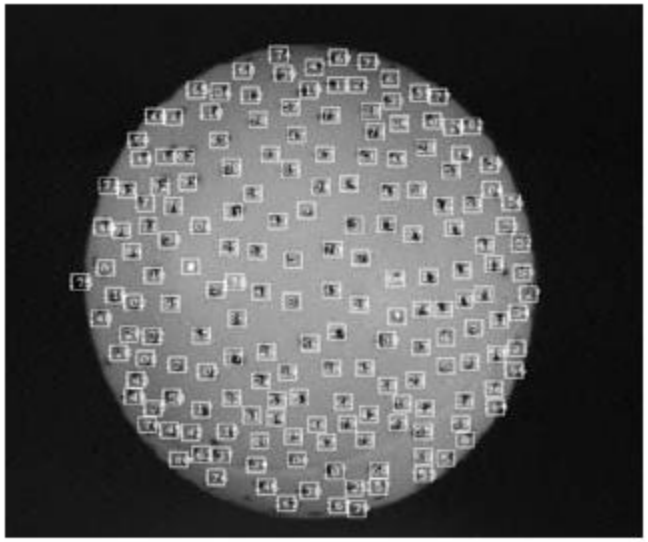


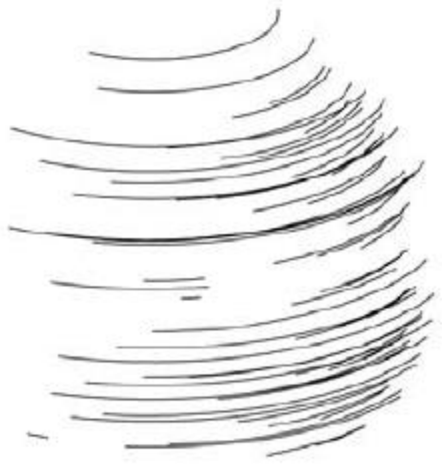
Implicit Surface Example

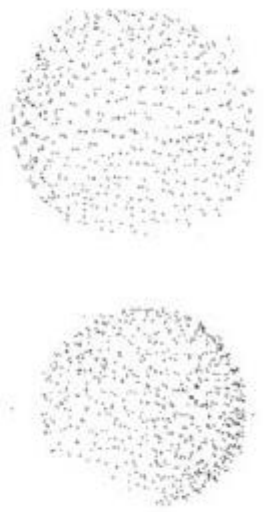
Where does SfM fail?

- Not enough images with enough overlap
  - Disconnected reconstructions
- Featureless or reflecting surfaces
  - No matches or bad matches
- Images with pure rotations
  - Recovery of “F” can fail or bad pose reconstruction
- Repeated structures (buildings or bridges)
  - Many consistent bad matches results in inconsistent reconstructions

Structure from motion under orthographic projection


(a)


(b)

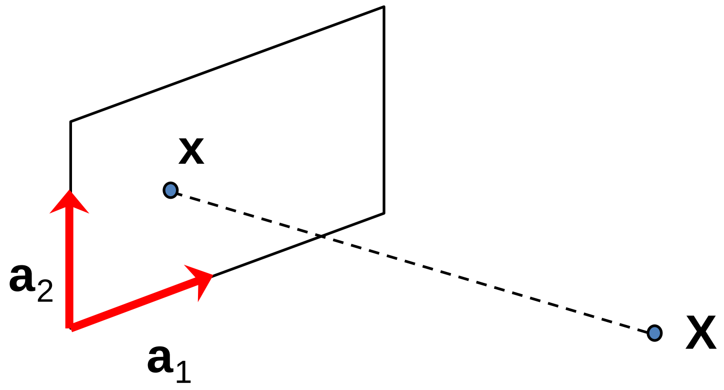

(c)

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

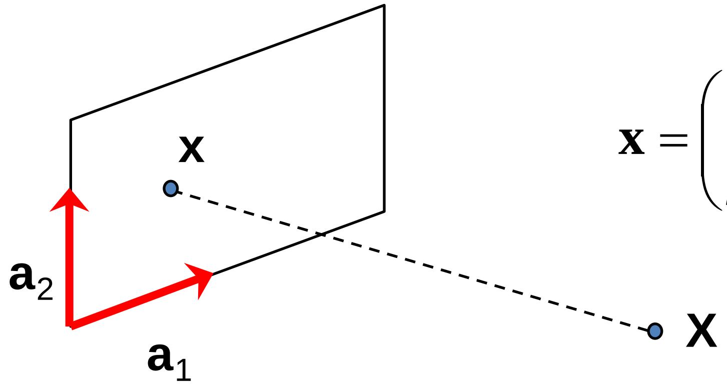


$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(R'_f \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix}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

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

Affine structure from motion

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



$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \mathbf{AX} + \underset{\substack{\uparrow \\ \text{Projection of world origin}}}{\mathbf{t}}$$

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

$$\underset{\substack{\uparrow \\ \text{Camera Parameters (2mx3)}}}{\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \underset{\substack{\uparrow \\ \text{3D Points (3xn)}}}{\cdots} & \mathbf{X}_n \end{bmatrix}$$

What if we instead observe corresponding 2d image points?

Can we recover the camera parameters and 3d points?

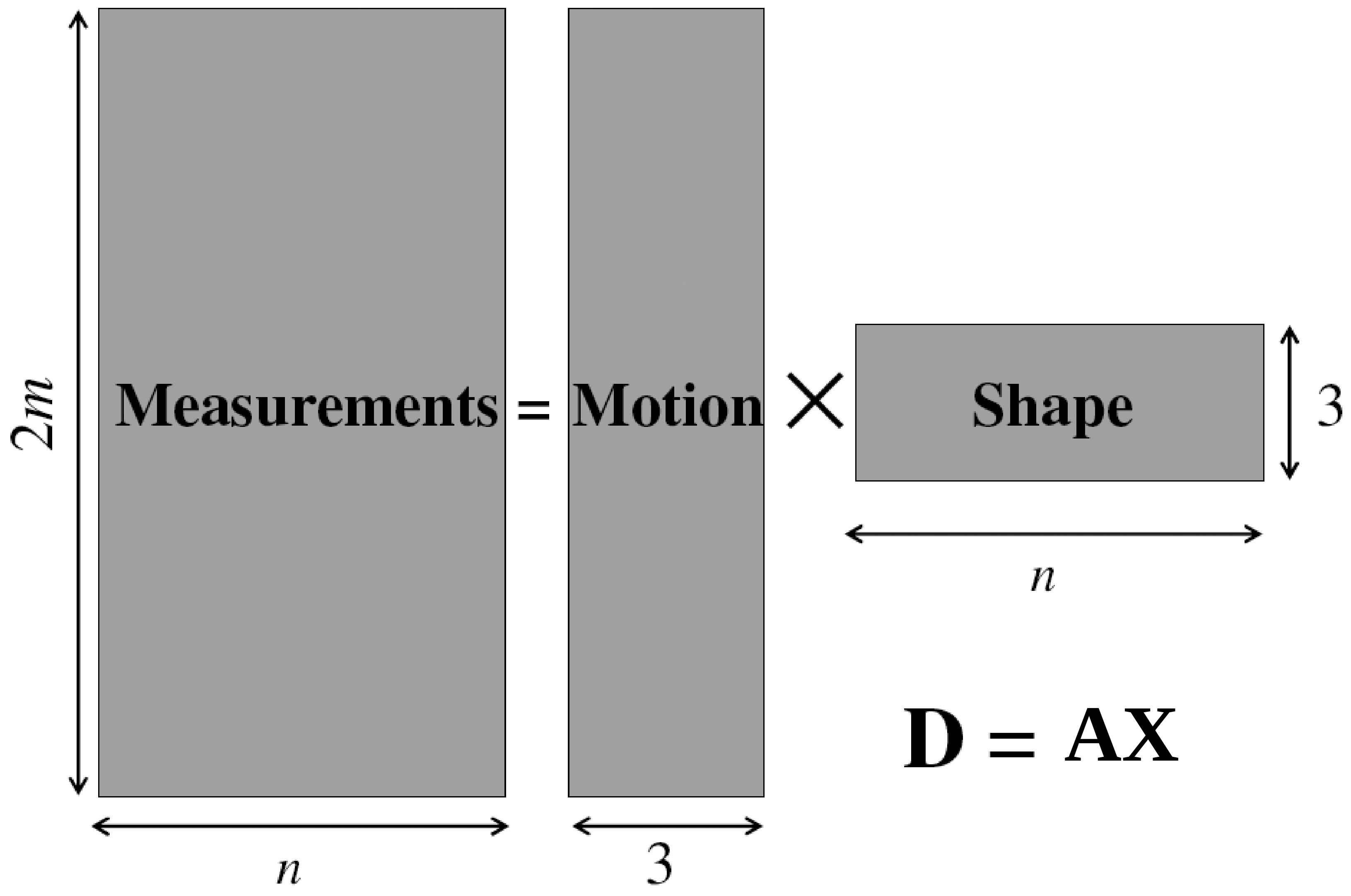
cameras ($2m$)

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

points (n)

What rank is the matrix of 2D points?

Factorizing the measurement matrix

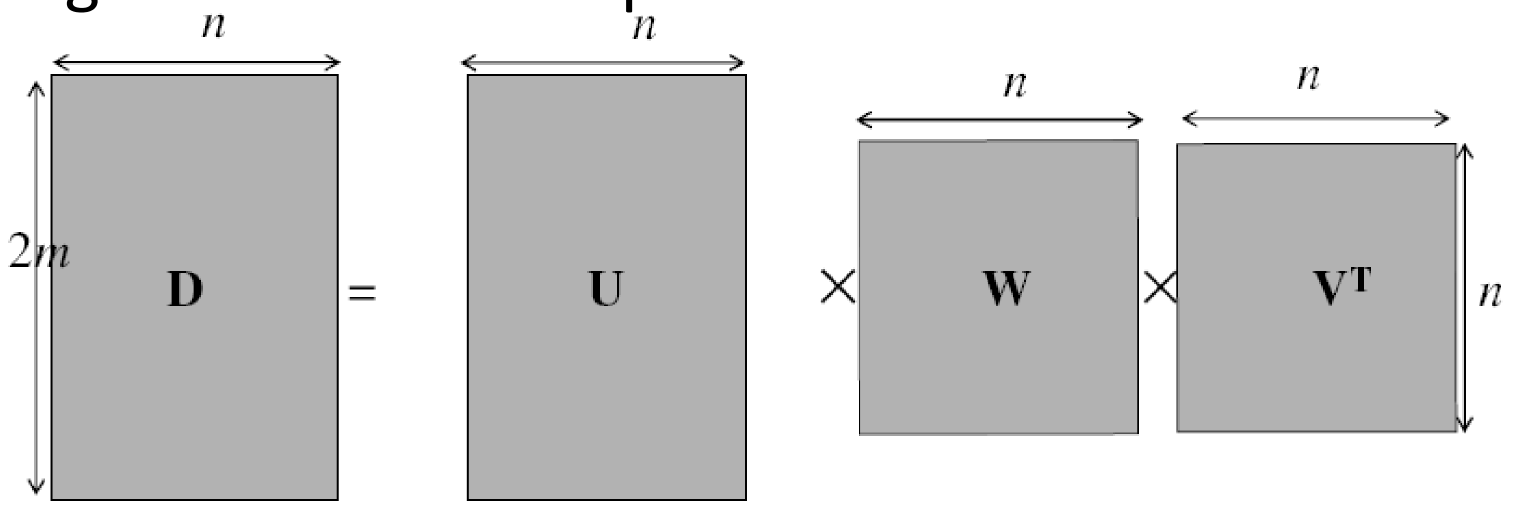


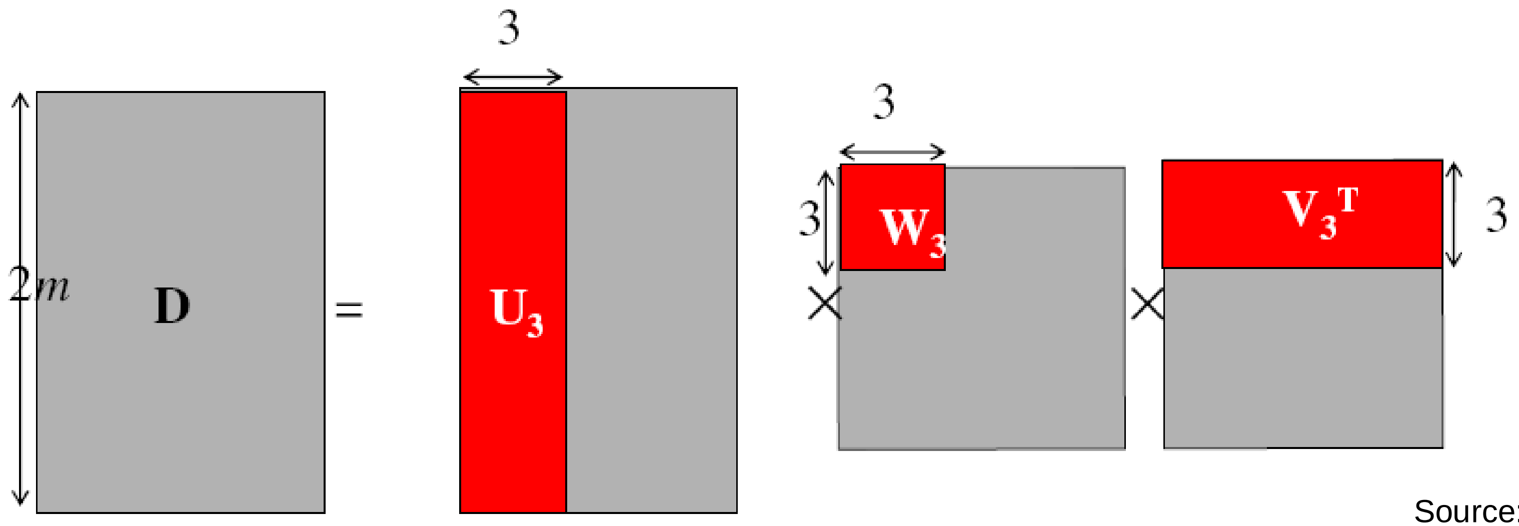
$$\mathbf{D} = \mathbf{A}\mathbf{X}$$

Source: M. Hebert

Factorizing the measurement matrix

- Singular value decomposition of D:

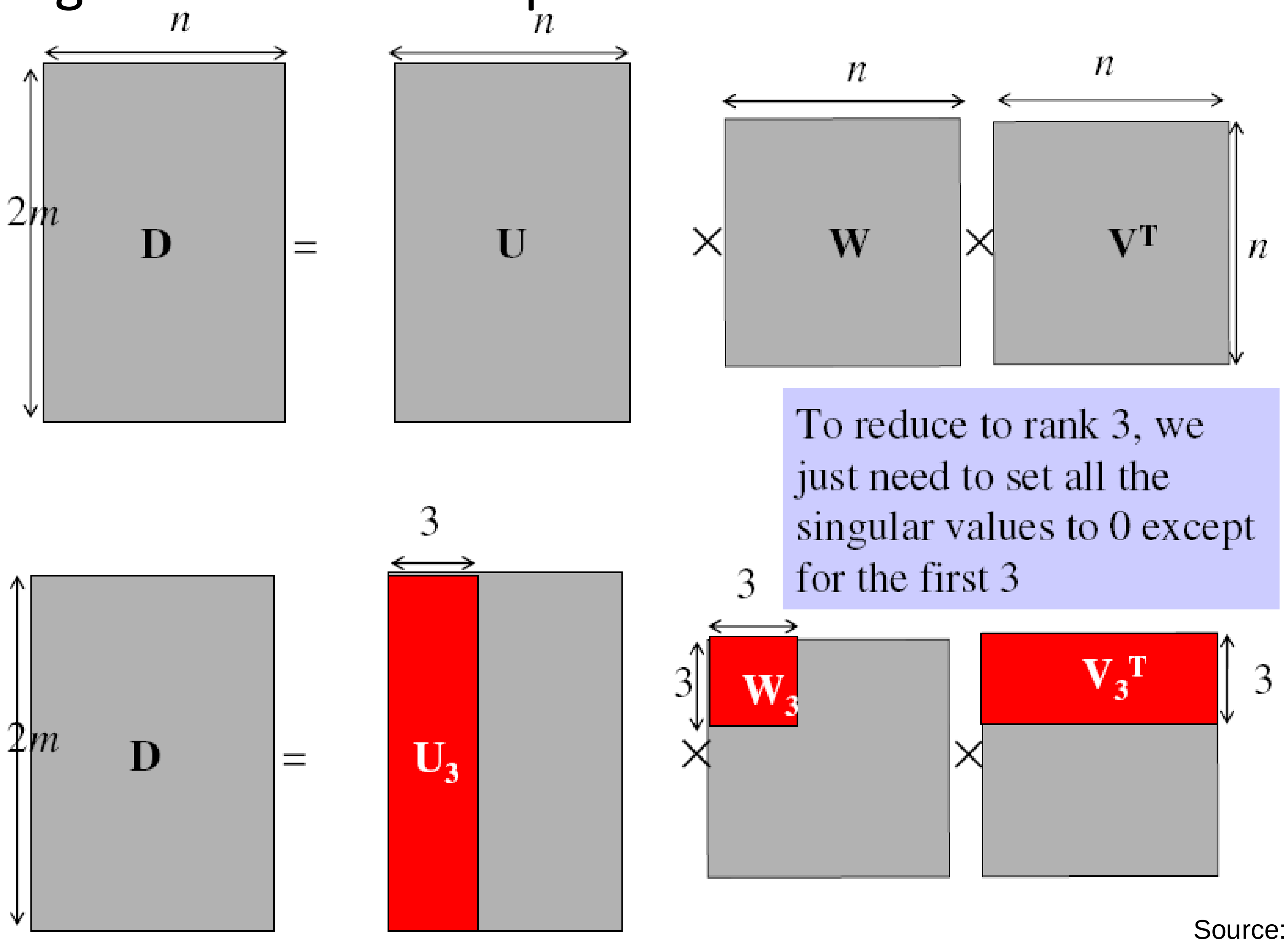




Source: M. Hebert

Factorizing the measurement matrix

- Singular value decomposition of D:

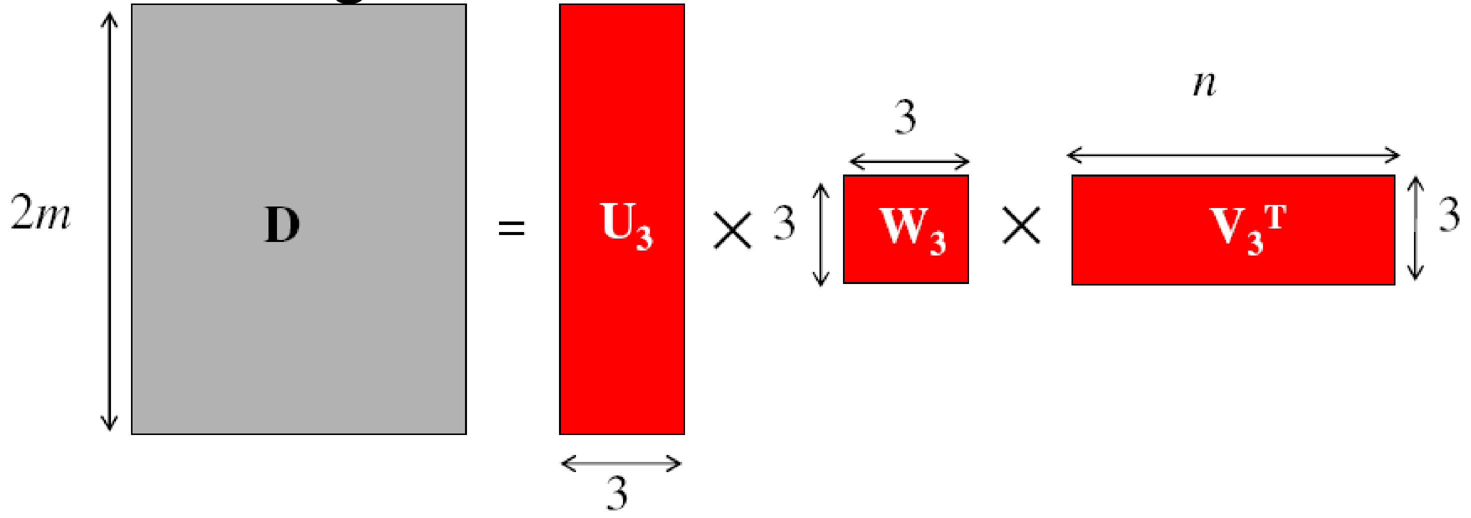


To reduce to rank 3, we just need to set all the singular values to 0 except for the first 3

Source: M. Hebert

Factorizing the measurement matrix

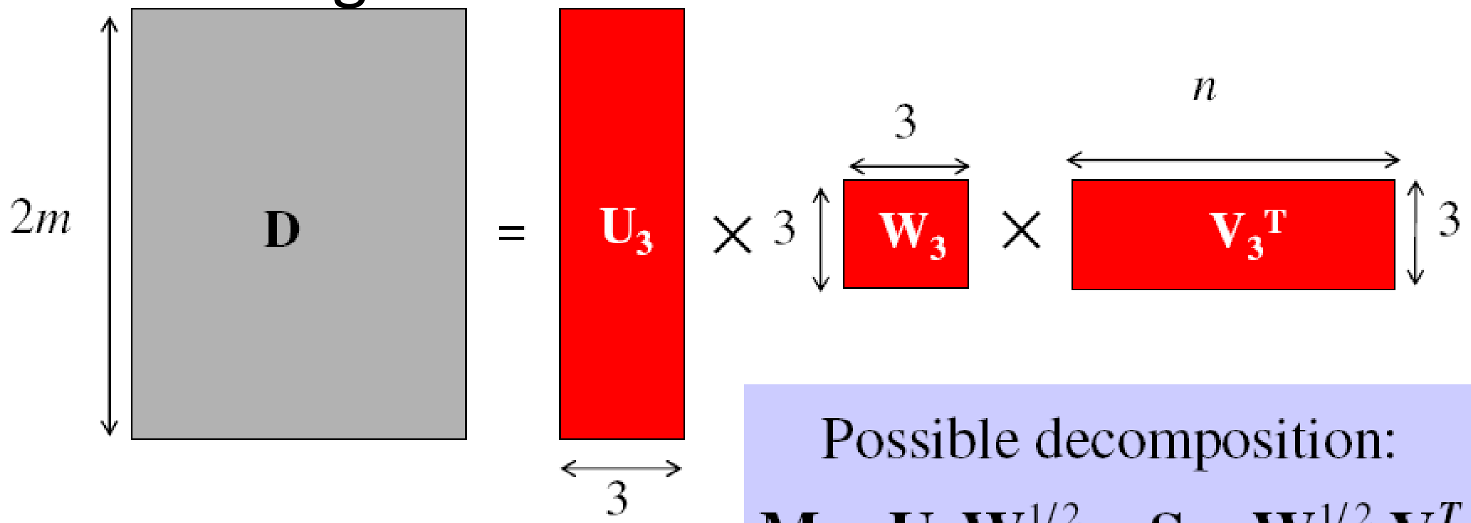
- Obtaining a factorization from SVD:



Source: M. Hebert

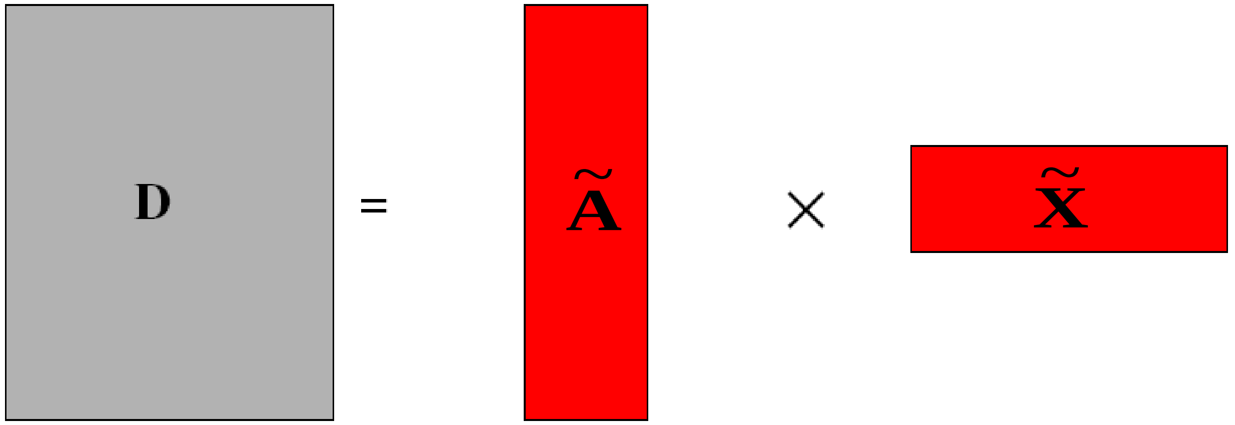
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



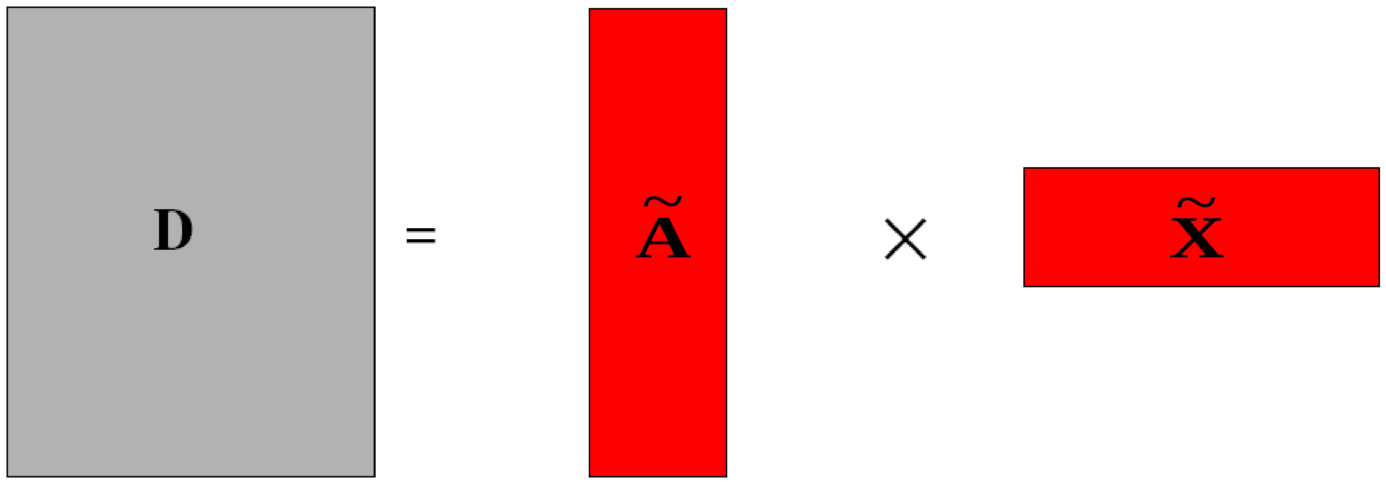
Possible decomposition:

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



Source: M. Hebert

Affine ambiguity

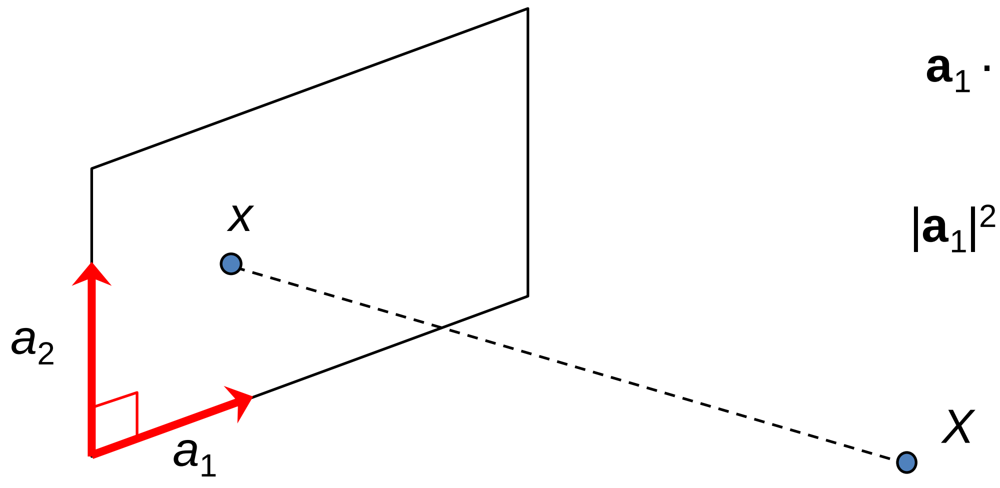


- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations $\mathbf{A} \rightarrow \mathbf{AC}$, $\mathbf{X} \rightarrow \mathbf{C}^{-1}\mathbf{X}$
- That is because we have only an affine transformation and we have not enforced any Euclidean 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 **L = CC**$^{\mathbf{T}}$
- Recover **C** from **L** by Cholesky decomposition: **L = CC**$^{\mathbf{T}}$
- Update **A** and **X**: **A = ÃC, X = C**$^{\mathbf{-1}}$**X̃**

Algorithm summary

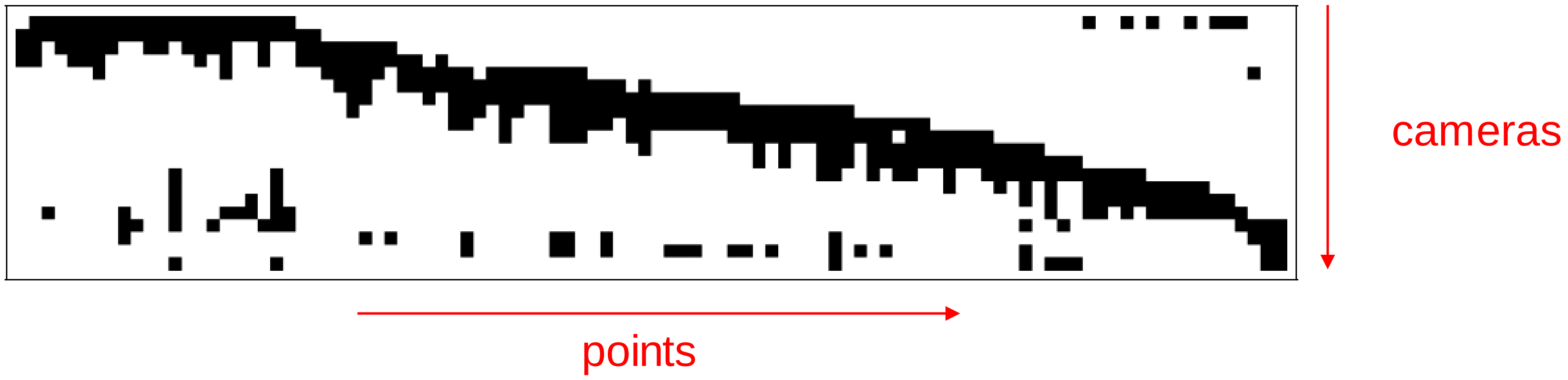
- Given: *m* images and *n* tracked features $\mathbf{x}_{ij}$
- For each image *i, c*enter the feature coordinates
- Construct a $2m \times n$ measurement matrix **D**:
  - Column *j* contains the projection of point *j* in all views
  - Row *i* contains one coordinate of the projections of all the *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:

  $\mathbf{A} = \mathbf{U}_3\mathbf{W}_3^{1/2}$ and $\mathbf{X} = \mathbf{W}_3^{1/2}\,\mathbf{V}_3^{T}$
- 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:



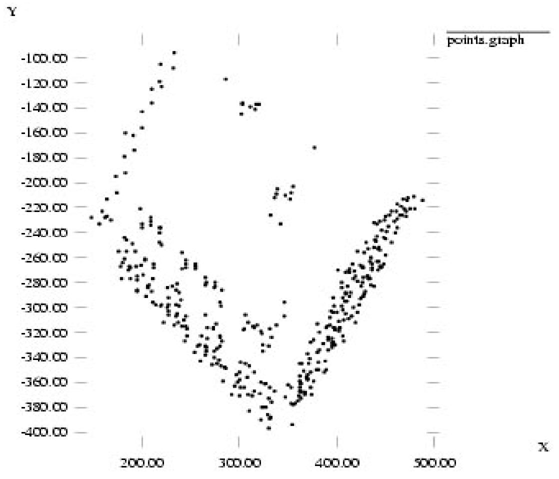
One solution:

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

Reconstruction results (your HW 3.4)

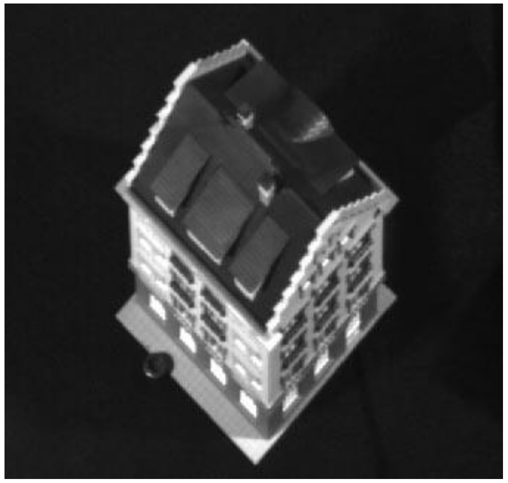
1

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.

Further reading

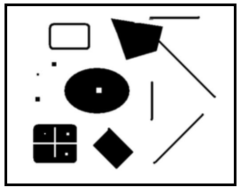
- Short explanation of Affine SfM: class notes from Lischinksi and Gruber

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

- Clear explanation of epipolar geometry and projective SfM
  - http://mi.eng.cam.ac.uk/~cipolla/publications/contributionToEditedBook/2008-SFM-chapters.pdf

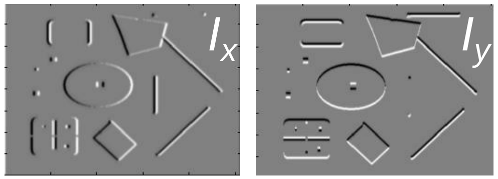
Review of Affine SfM from Interest Points

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

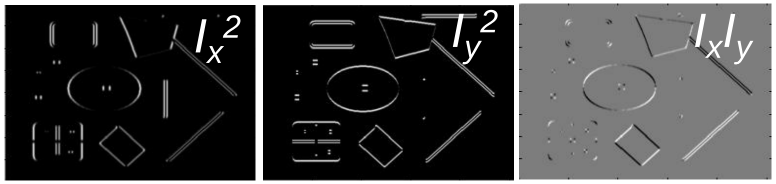


$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives

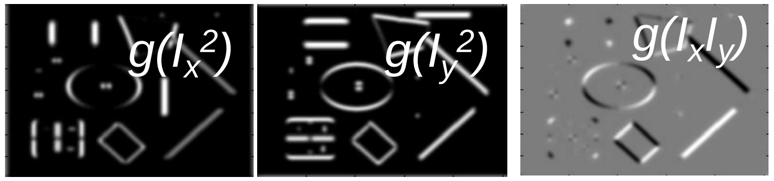


2. Square of derivatives



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

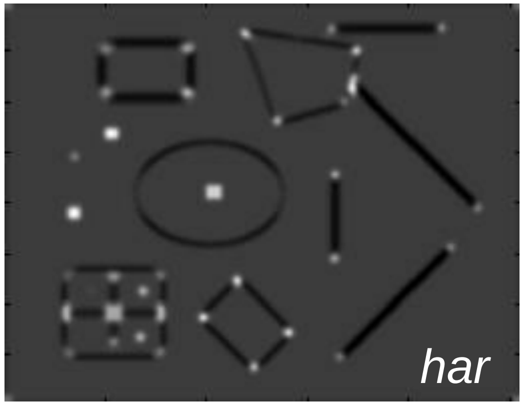
3. Gaussian filter $g(\sigma_I)$



4. Cornerness function – both eigenvalues are strong

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

5. Non-maxima suppression



Review of Affine SfM from Interest Points

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;`

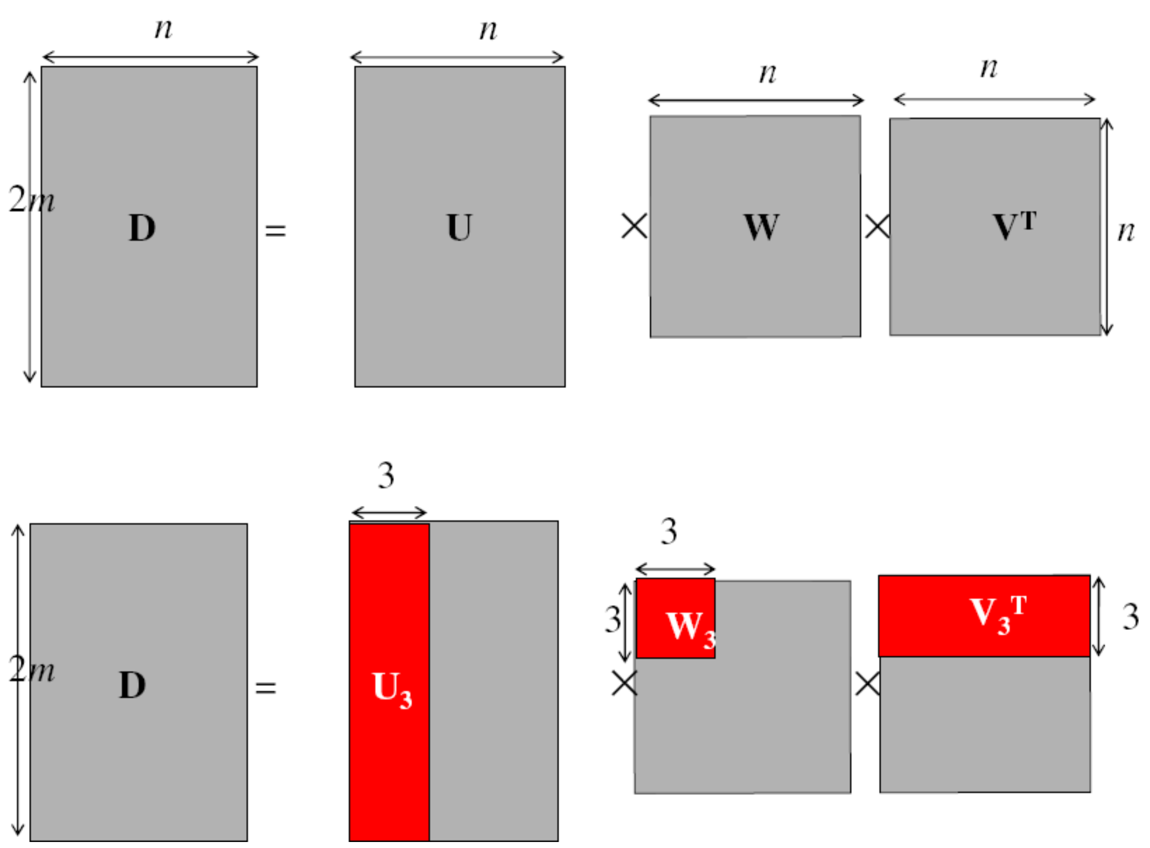
d) Recalculate I_t

e) Repeat steps 2-4 until small change

- Use interpolation for subpixel values

Review of Affine SfM from Interest Points

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

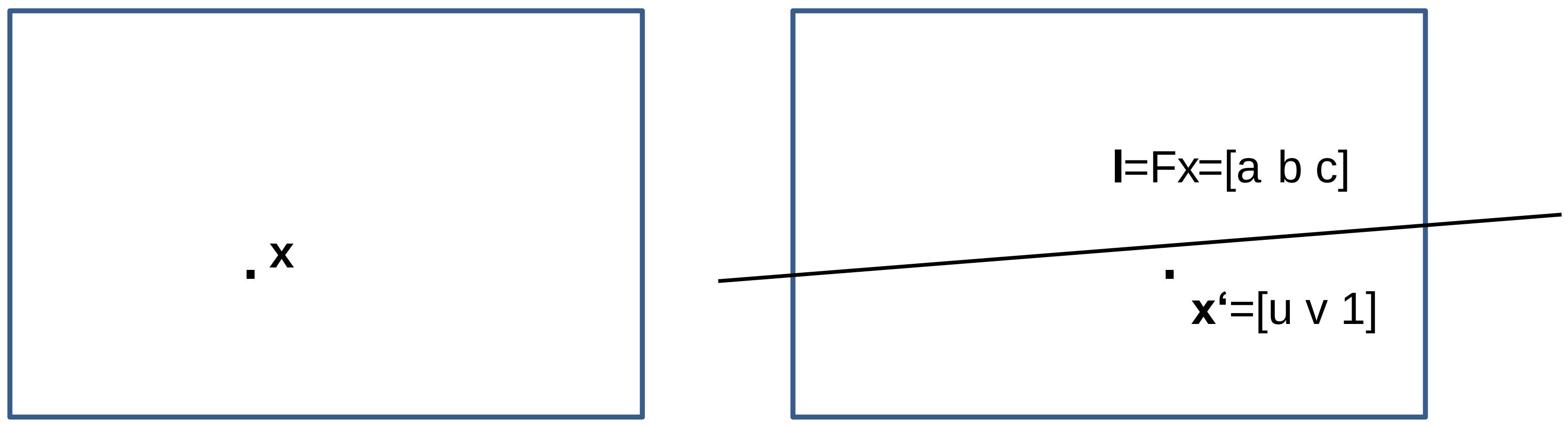


✚ Solve for orthographic constraints

Tips for HW 3

- Problem 1: vanishing points
  - Use lots of lines to estimate vanishing points
  - For estimation of VP from lots of lines, see single-view geometry chapter, or use robust estimator of a central intersection point
  - For obtaining intrinsic camera matrix, numerical solver (e.g., `fsolve` in matlab) may be helpful
- Problem 3: epipolar geometry
  - Use reprojection distance for inlier check (make sure to compute line to point distance correctly)
- Problem 4: structure from motion
  - Use Matlab's `chol` and `svd`
  - If you weren't able to get tracking to work from HW2 can use provided points

Distance of point to epipolar line



$$d(l, x') = \frac{|au + bv + c|}{\sqrt{a^2 + b^2}}$$

The Reading List

- "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
- "An efficient solution to the five-point relative pose problem", Nister, PAMI 2004
- "Accurate, dense, and robust multiview stereopsis", Furukawa and Ponce, CVPR 2007
- "Photo tourism: exploring image collections in 3d", ACM SIGGRAPH 2006
- "Building Rome in a day", Agarwal et al., ICCV 2009

(also see reading from earlier slides)

Next class

- Clustering and using clustered interest points for matching images in a large database