

Structure from Motion

Computer Vision

CS 543 / ECE 549

University of Illinois

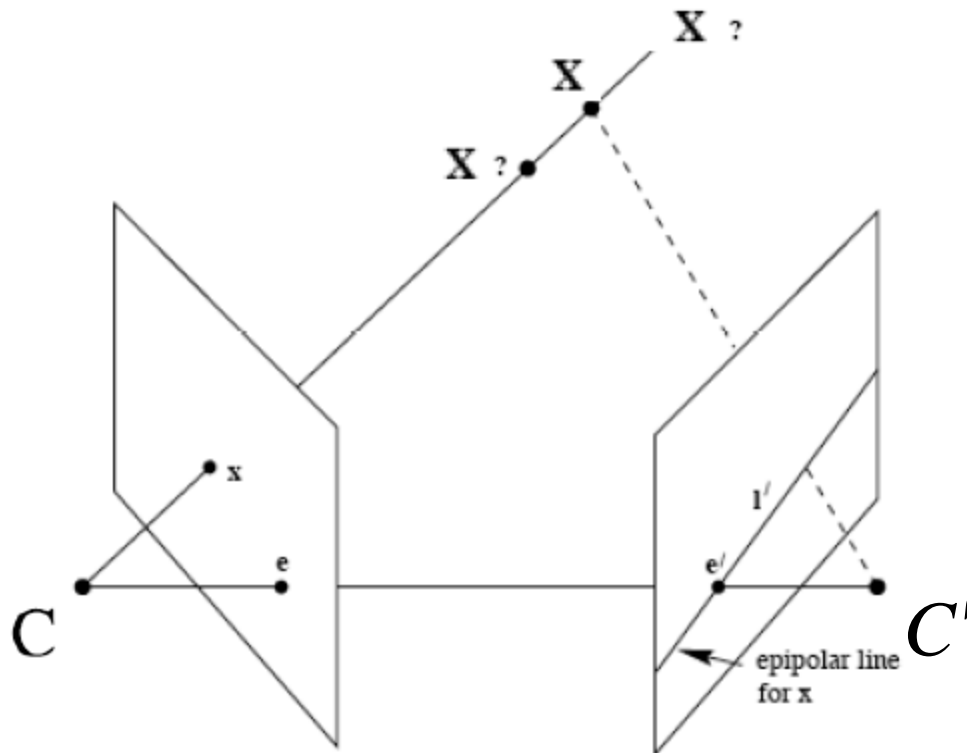
Derek Hoiem

This class: structure from motion

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

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

- If \mathbf{x} and \mathbf{x}' correspond to the same 3d point \mathbf{X} :

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

Recap: Automatically Relating Projections

Assume we have matched points $\mathbf{x} \leftrightarrow \mathbf{x}'$ with outliers

Homography (No Translation)

- Correspondence Relation

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \Rightarrow \mathbf{x}' \times \mathbf{H}\mathbf{x} = \mathbf{0}$$

1. Normalize image coordinates

$$\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x} \quad \tilde{\mathbf{x}}' = \mathbf{T}'\mathbf{x}'$$

2. RANSAC with 4 points

3. De-normalize: $\mathbf{H} = \mathbf{T}'^{-1}\tilde{\mathbf{H}}\mathbf{T}$

Fundamental Matrix (Translation)

- Correspondence Relation

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

1. Normalize image coordinates

$$\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x} \quad \tilde{\mathbf{x}}' = \mathbf{T}'\mathbf{x}'$$

2. RANSAC with 8 points

3. Enforce $\det(\tilde{\mathbf{F}}) = 0$ by SVD

4. De-normalize: $\mathbf{F} = \mathbf{T}'^{-1}\tilde{\mathbf{F}}\mathbf{T}$

Recap

- We can get projection matrices \mathbf{P} and \mathbf{P}' up to a projective ambiguity

$$\mathbf{P} = [\mathbf{I} \mid \mathbf{0}] \quad \mathbf{P}' = [[\mathbf{e}']_{\times} \mathbf{F} \mid \mathbf{e}'] \quad \mathbf{e}'^T \mathbf{F} = 0$$

See HZ p. 255-256

- [Code:](#)

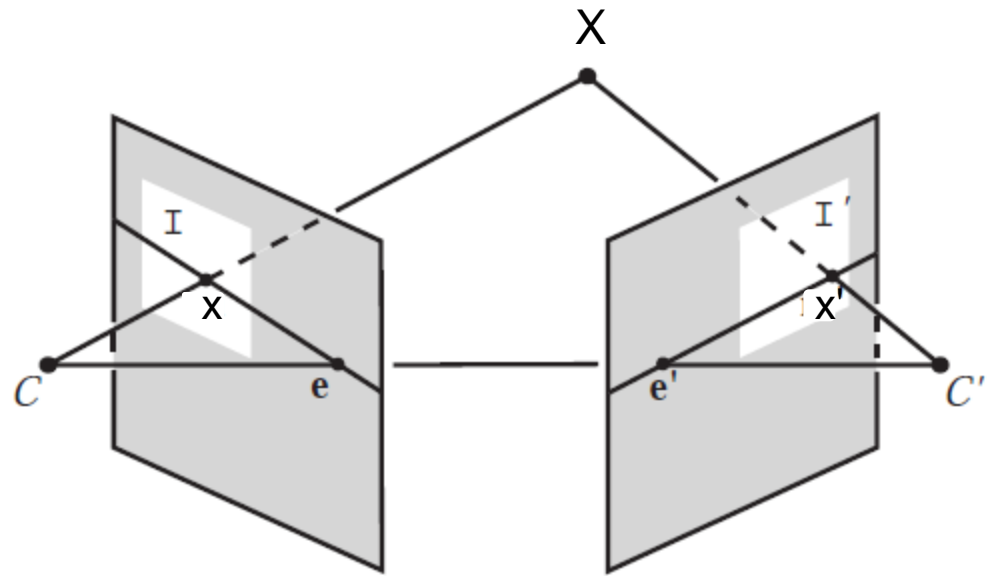
```
function P = vgg_P_from_F(F)
[U,S,V] = svd(F);
e = U(:,3);
P = [-vgg_contreps(e)*F e];
```

Recap

- Fundamental matrix song

Triangulation: Linear Solution

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



$$\mathbf{x} \times (\mathbf{P}\mathbf{X}) = 0$$

$$\mathbf{x}' \times (\mathbf{P}'\mathbf{X}) = 0$$



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

Triangulation: Linear Solution

Given \mathbf{P} , \mathbf{P}' , \mathbf{x} , \mathbf{x}'

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

1. Precondition points and projection matrices
2. Create matrix \mathbf{A}
3. $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \text{svd}(\mathbf{A})$
4. $\mathbf{X} = \mathbf{V}(:, \text{end})$

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

Pros and Cons

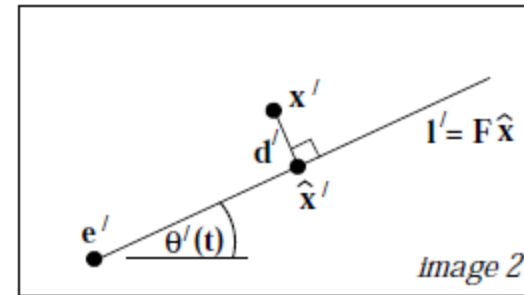
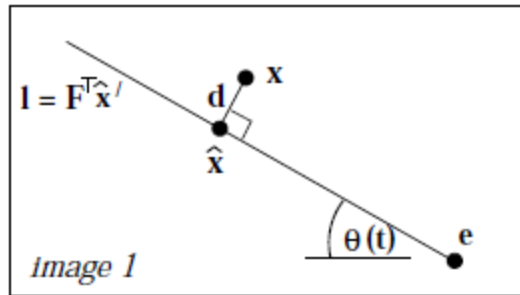
- Works for any number of corresponding images
- Not projectively invariant

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

Triangulation: Non-linear Solution

- Minimize projected error while satisfying $\mathbf{x}^\top \mathbf{F} \mathbf{x} = 0$

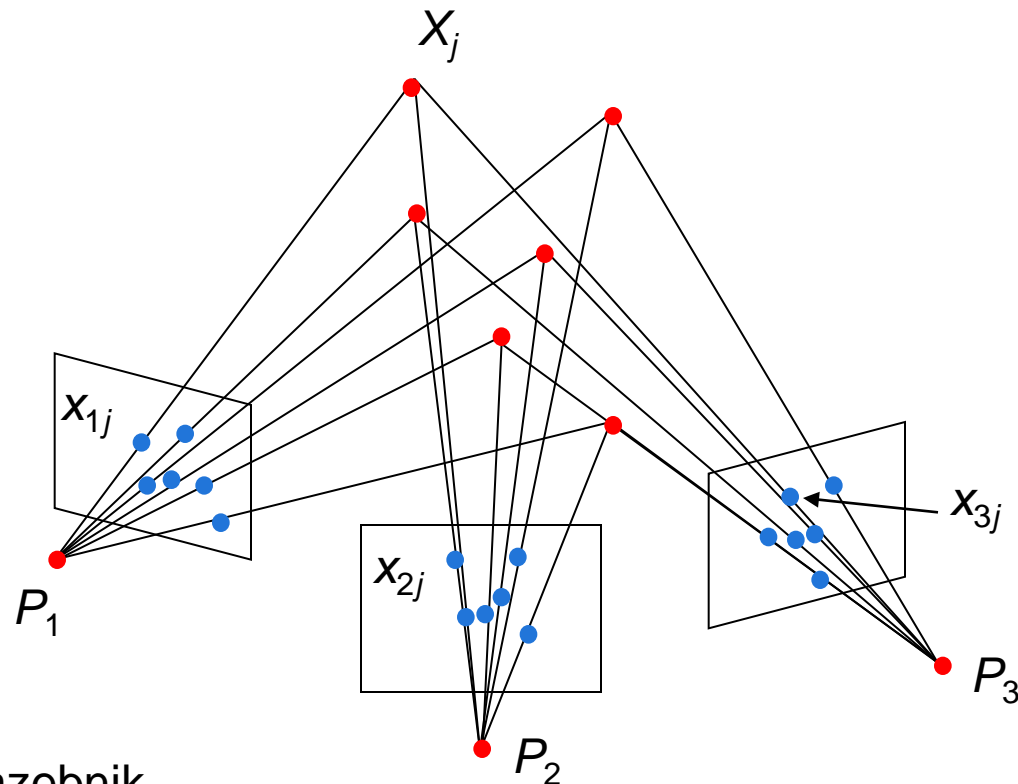
$$\mathcal{C}(\mathbf{X}) = d(\mathbf{x}, \hat{\mathbf{x}})^2 + d(\mathbf{x}', \hat{\mathbf{x}}')^2$$



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

Projective structure from motion

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

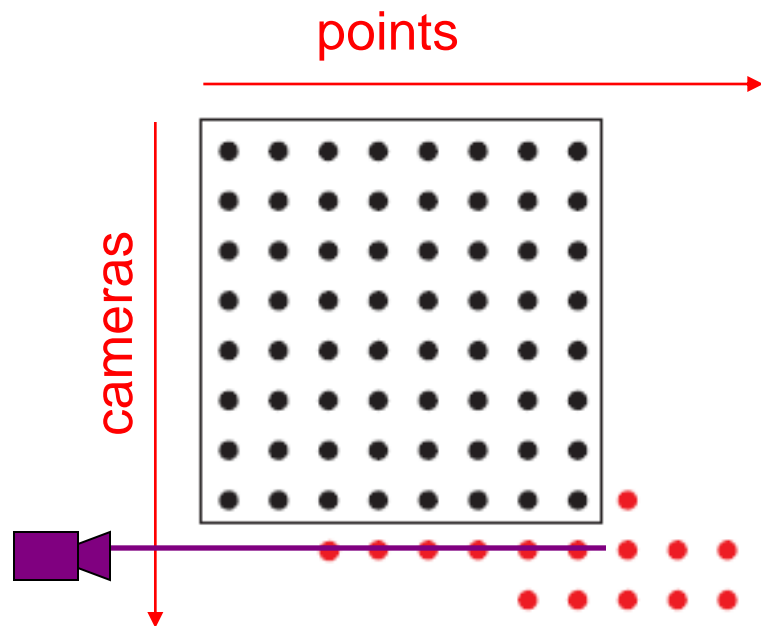


Projective structure from motion

- Given: m images of n fixed 3D points
 - $\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n$
- 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 \mathbf{Q} :
 - $\mathbf{X} \rightarrow \mathbf{QX}, \mathbf{P} \rightarrow \mathbf{PQ}^{-1}$
- We can solve for structure and motion when
 - $2mn \geq 11m + 3n - 15$
- For two cameras, at least 7 points are needed

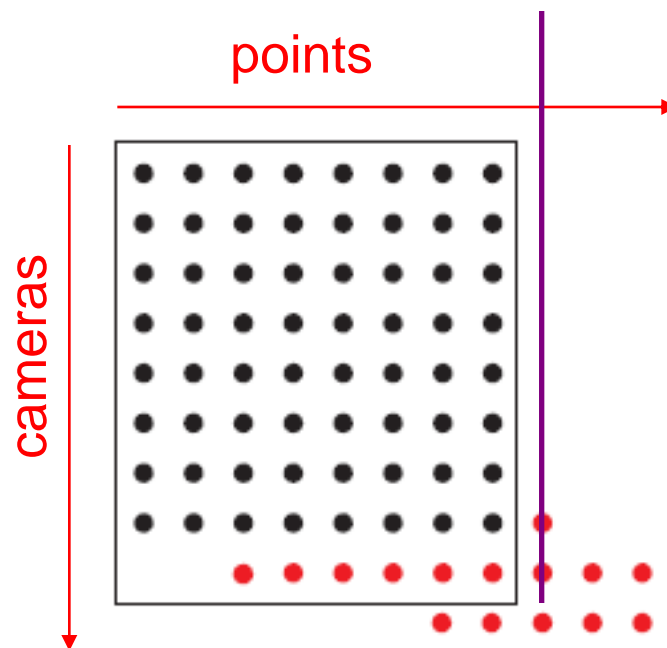
Sequential structure from motion

- 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*



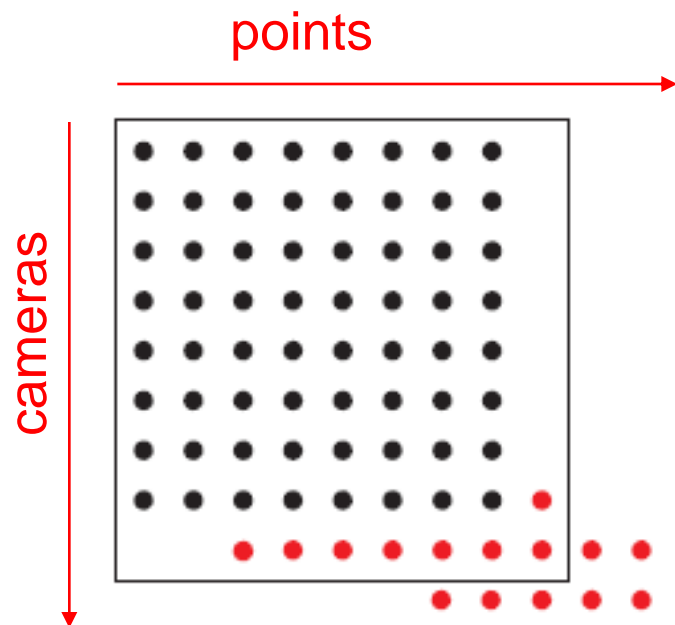
Sequential structure from motion

- 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

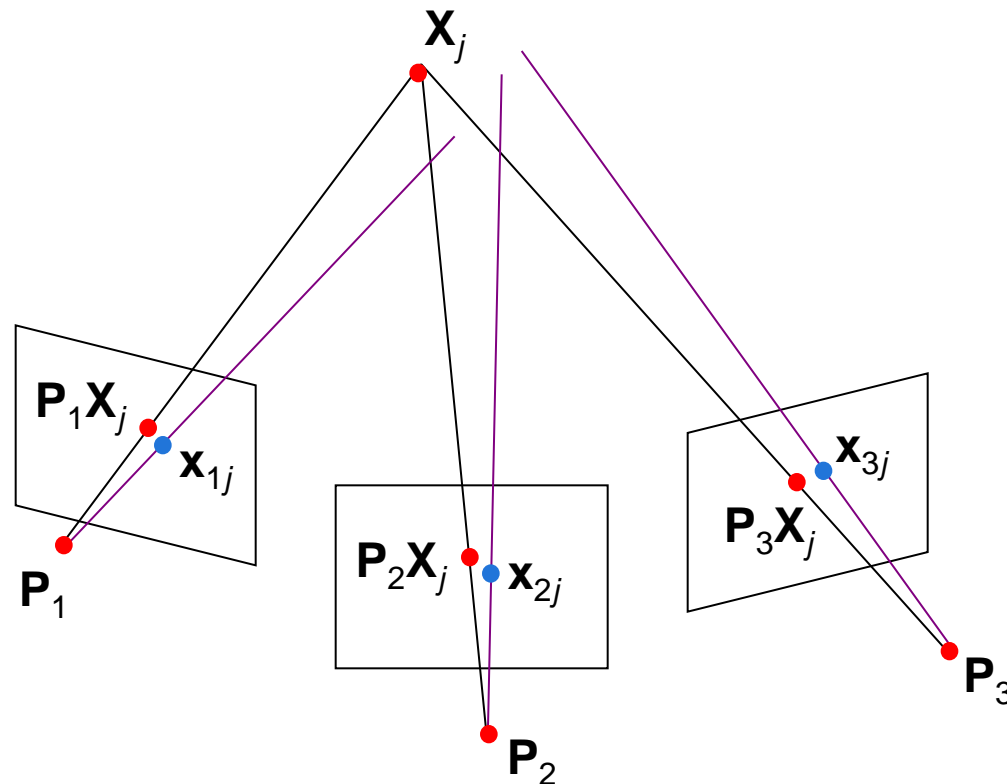
- Initialize motion from two images using fundamental matrix
- Initialize structure by triangulation
- For each additional view:
 - Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
 - Refine and extend structure: compute new 3D points, 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

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



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 \mathbf{Q} such that all camera matrices are in the form $\mathbf{P}_i = \mathbf{K} [\mathbf{R}_i | \mathbf{t}_i]$
- Can use constraints on the form of the calibration matrix, such as zero skew

Summary so far

- From two images, we can:
 - Recover fundamental matrix F
 - Recover canonical cameras P and P' from F
 - Estimate 3D positions (if K is known) that correspond to each pixel
- For a moving camera, we can:
 - Initialize by computing F , P , X for two images
 - Sequentially add new images, computing new P , refining X , and adding points
 - Auto-calibrate assuming fixed calibration matrix to upgrade to similarity transform

Photo synth

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

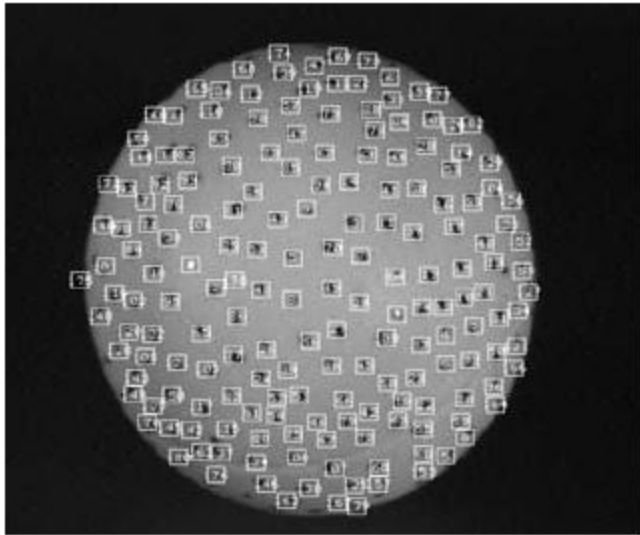


<http://photosynth.net/>

3D from multiple images



Structure from motion under orthographic projection



(a)



(b)



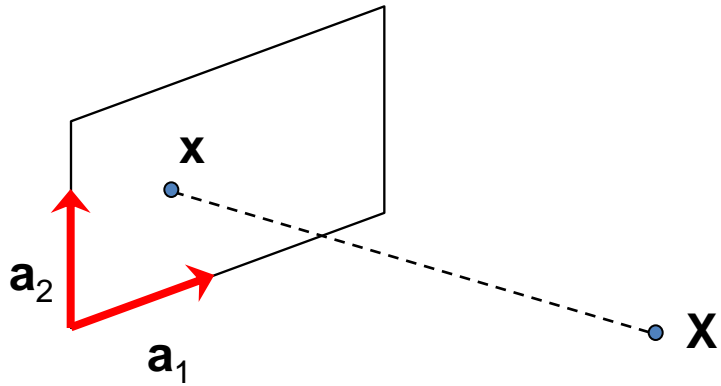
(c)

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.

Affine projection for rotated/translated camera

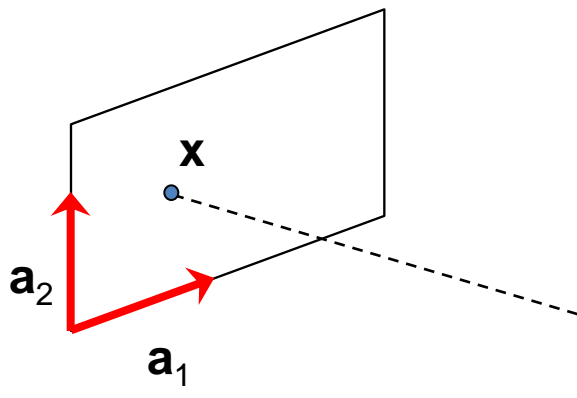


$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(R'_f \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix}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

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

Affine structure from motion

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



The diagram illustrates the affine projection of a 3D point \mathbf{X} onto a 2D plane. A 3D point \mathbf{X} (blue dot) is shown in a 3D coordinate system. A dashed line represents the projection ray from \mathbf{X} through the origin of the 2D plane. The 2D plane is defined by two red vectors, \mathbf{a}_1 and \mathbf{a}_2 , which form a basis for the plane. The projection of \mathbf{X} onto this plane is a 2D point \mathbf{x} (blue dot).

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

Projection of world origin

- We are given corresponding 2D points (\mathbf{x}) in several frames
- We want to estimate the 3D points (\mathbf{X}) and the affine parameters of each camera (\mathbf{A})

Step 1: Simplify by getting rid of \mathbf{t} : shift to centroid of points for each camera

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



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



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

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

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

Camera Parameters (2m x 3)

3D Points (3 x n)

The diagram illustrates the computation of 2D positions from 3D points and camera parameters. It shows a vertical stack of camera parameter matrices $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_m$ multiplied by a horizontal row of 3D point vectors $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$. A blue arrow points from the label 'Camera Parameters (2m x 3)' to the first matrix \mathbf{A}_1 . Another blue arrow points from the label '3D Points (3 x n)' to the first point vector \mathbf{X}_1 .

What if we instead observe corresponding 2d image points?

Can we recover the camera parameters and 3d points?

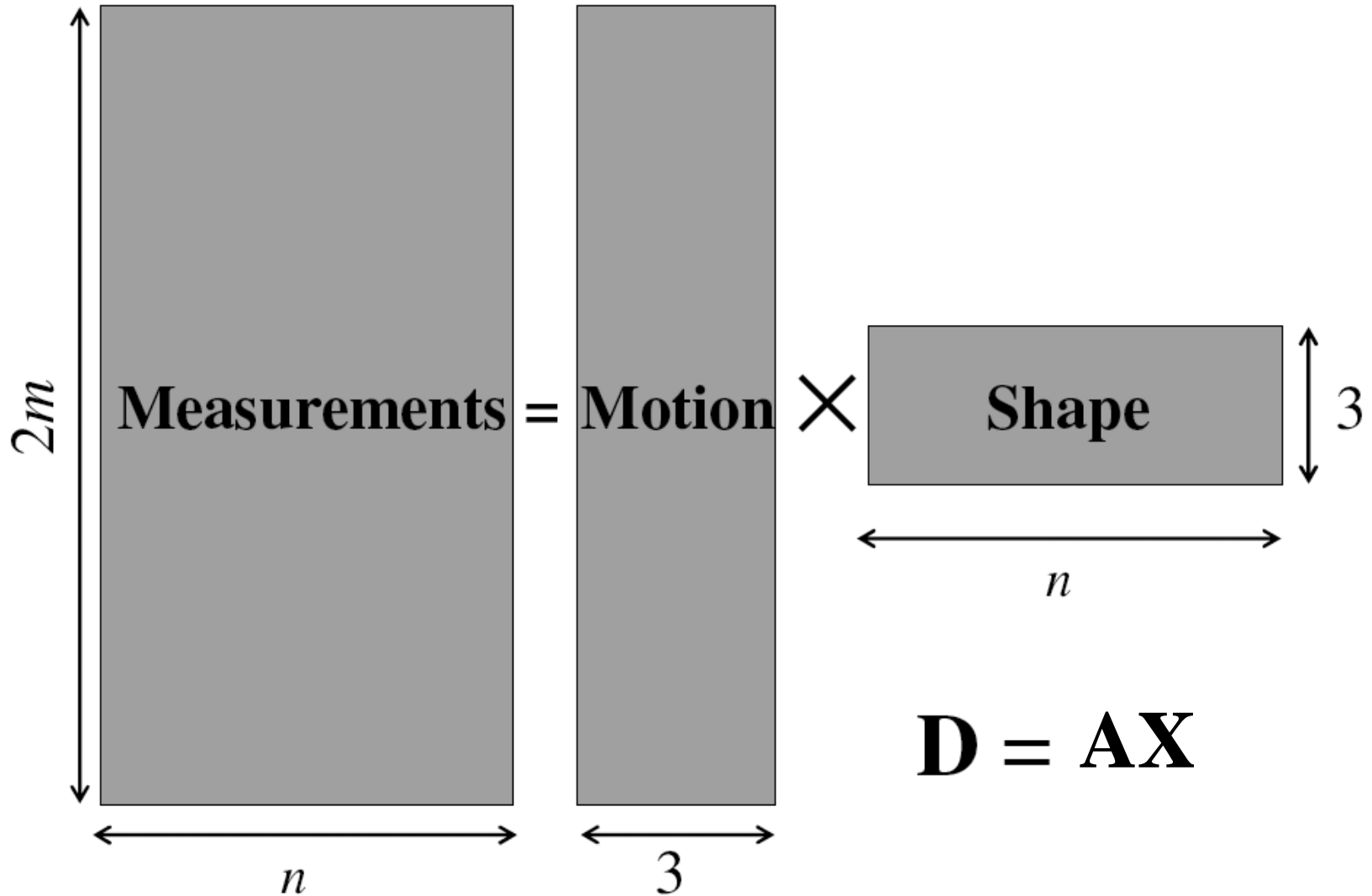
$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix}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

cameras ($2m$)

points (n)

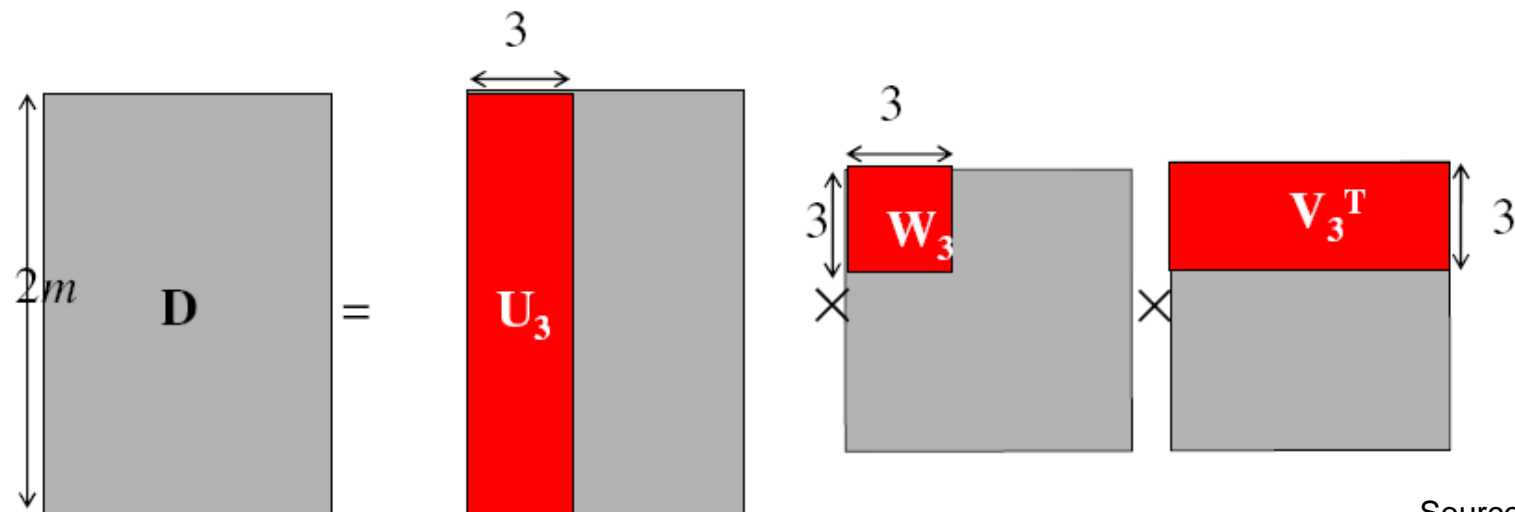
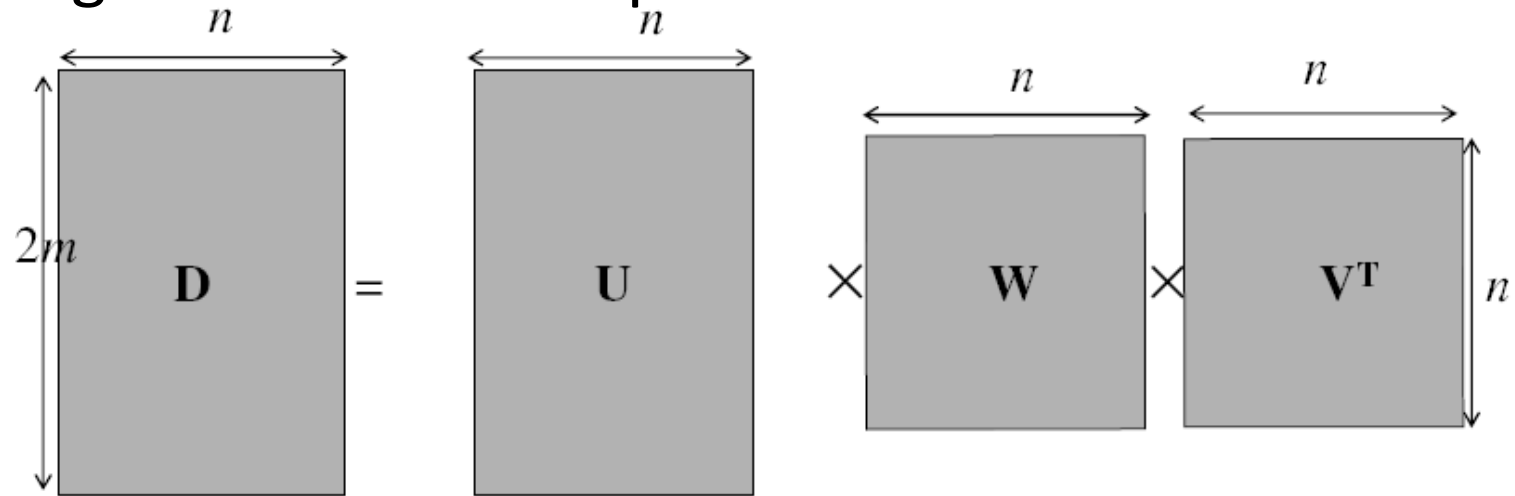
What rank is the matrix of 2D points?

Factorizing the measurement matrix



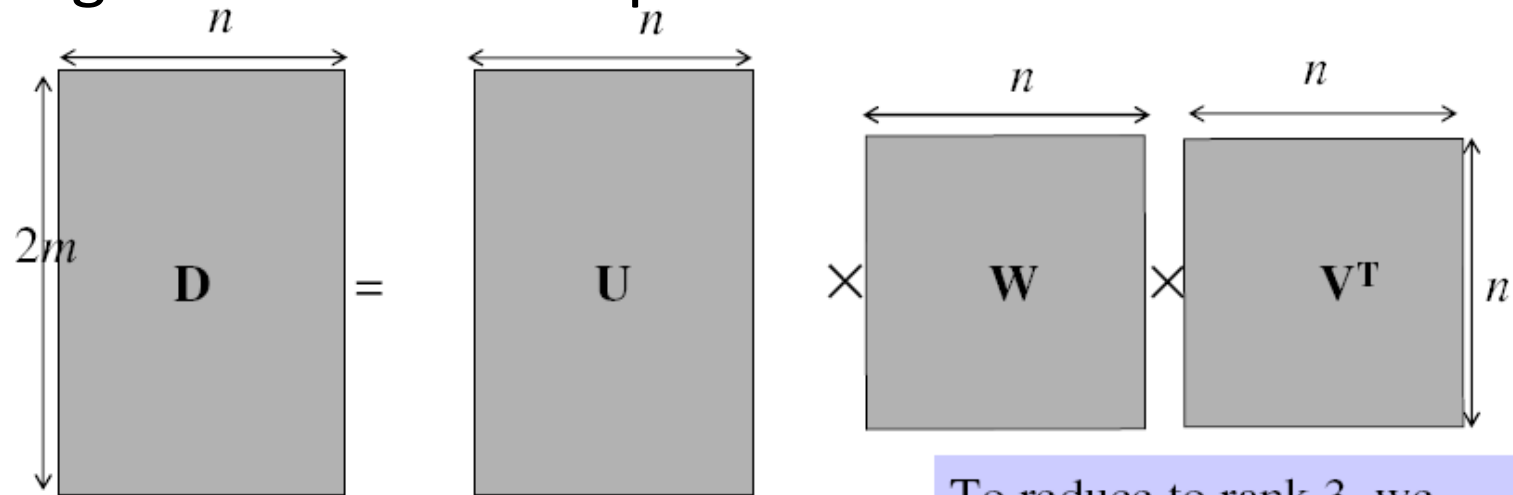
Factorizing the measurement matrix

- Singular value decomposition of D :

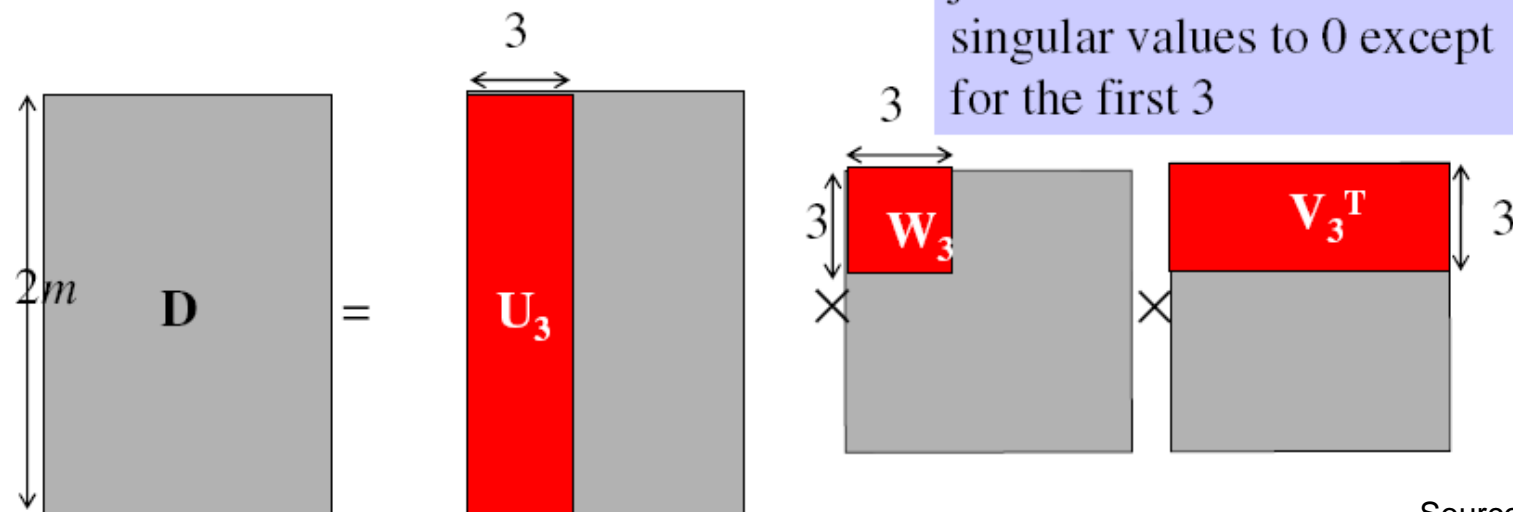


Factorizing the measurement matrix

- Singular value decomposition of D :

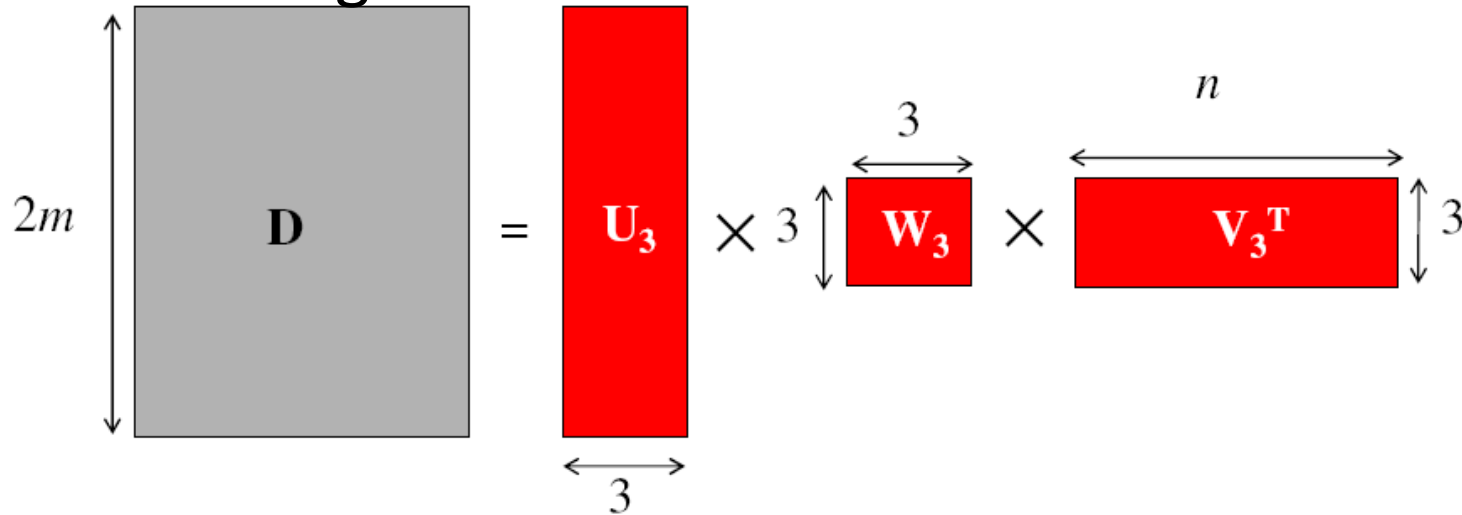


To reduce to rank 3, we just need to set all the singular values to 0 except for the first 3



Factorizing the measurement matrix

- Obtaining a factorization from SVD:



Factorizing the measurement matrix

- Obtaining a factorization from SVD:

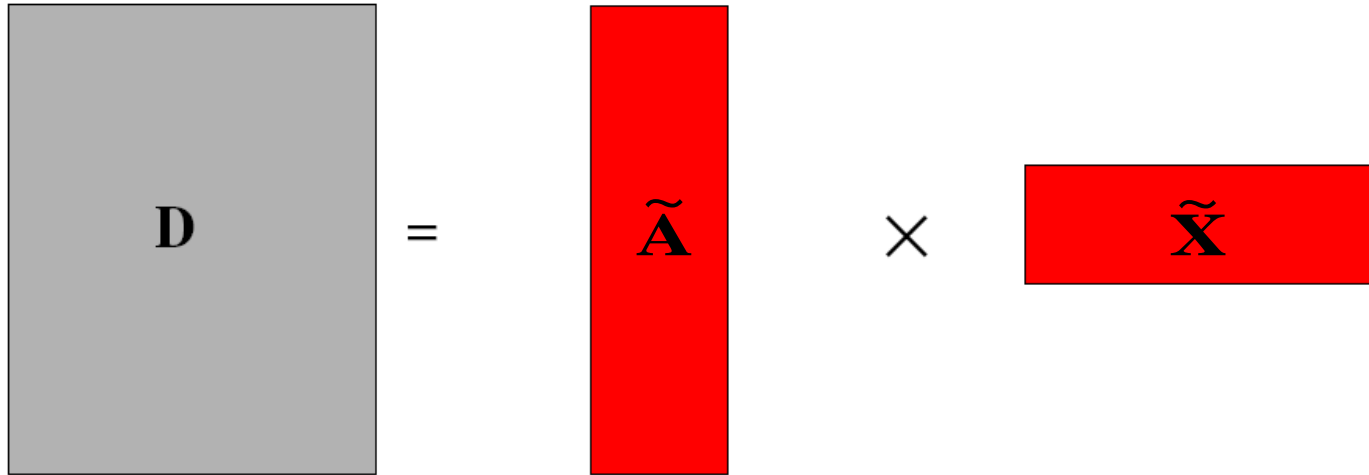
$$\begin{array}{c} \text{\scriptsize } 2m \\ \updownarrow \\ \text{\textbf{D}} \end{array} = \begin{array}{c} \text{\textbf{U}}_3 \\ \text{\scriptsize } 3 \\ \leftarrow \end{array} \times \begin{array}{c} \text{\scriptsize } 3 \\ \leftarrow \end{array} \begin{array}{c} \text{\textbf{W}}_3 \\ \updownarrow \text{\scriptsize } 3 \end{array} \times \begin{array}{c} \text{\scriptsize } n \\ \leftarrow \end{array} \begin{array}{c} \text{\textbf{V}}_3^T \\ \updownarrow \text{\scriptsize } 3 \end{array}$$

Possible decomposition:

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

$$\text{\textbf{D}} = \text{\textbf{\tilde{A}}} \times \text{\textbf{\tilde{X}}}$$

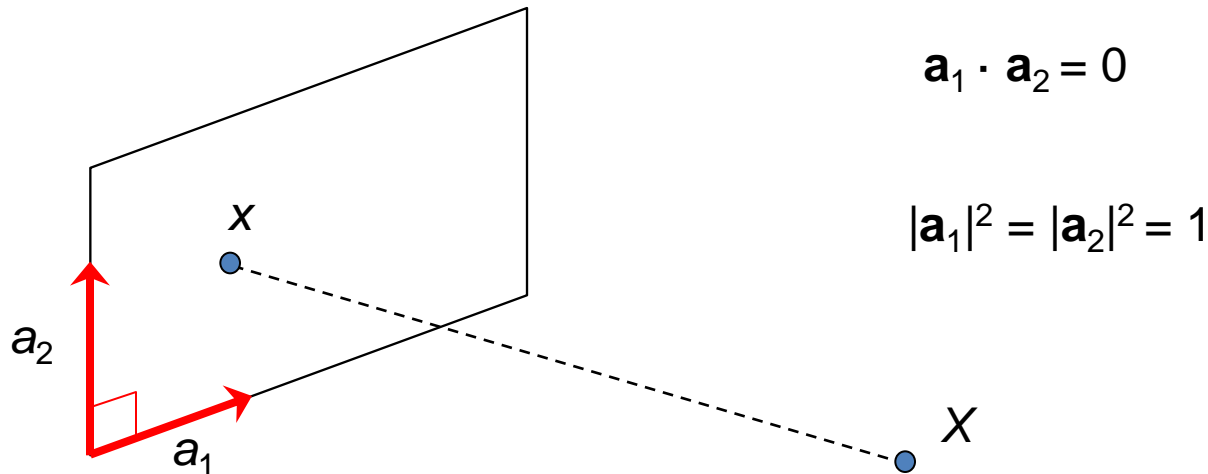
Affine ambiguity


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

- The decomposition is not unique. We get the same \mathbf{D} by using any 3×3 matrix \mathbf{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)

Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and of unit length



Solve for orthographic constraints

Three equations for each image i

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

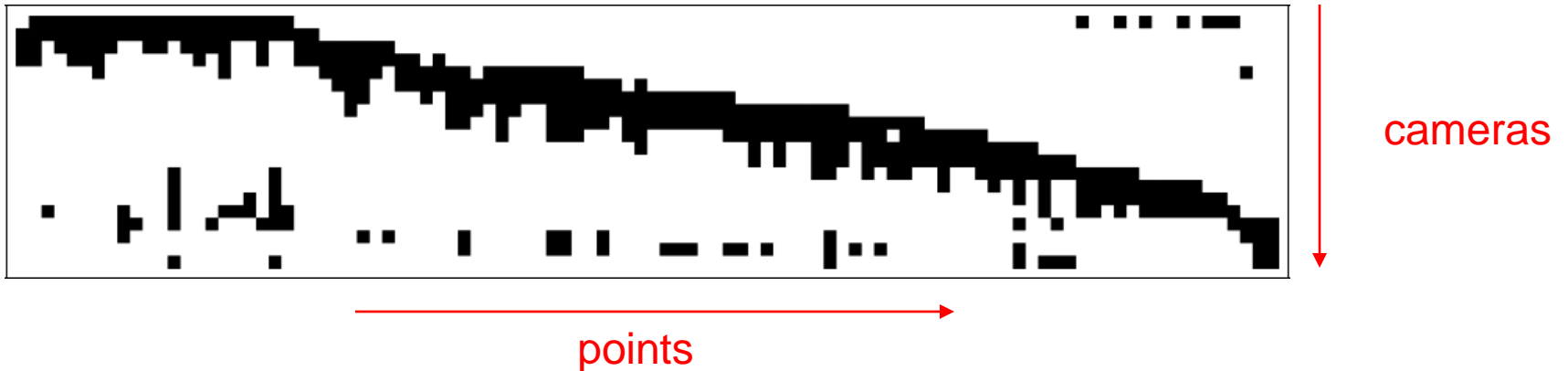
- Solve for $\mathbf{L} = \mathbf{C} \mathbf{C}^T$
- Recover \mathbf{C} from \mathbf{L} by Cholesky decomposition:
 $\mathbf{L} = \mathbf{C} \mathbf{C}^T$
- Update \mathbf{A} and \mathbf{X} : $\mathbf{A} = \tilde{\mathbf{A}} \mathbf{C}$, $\mathbf{X} = \mathbf{C}^{-1} \tilde{\mathbf{X}}$

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: $\mathbf{D} = \mathbf{U} \mathbf{W} \mathbf{V}^T$
 - 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:
$$\mathbf{A} = \mathbf{U}_3 \mathbf{W}_3^{1/2} \text{ and } \mathbf{X} = \mathbf{W}_3^{1/2} \mathbf{V}_3^T$$
- Eliminate affine ambiguity

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 Lischinski and Gruber

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

Reconstruction results (your HW 4)



1



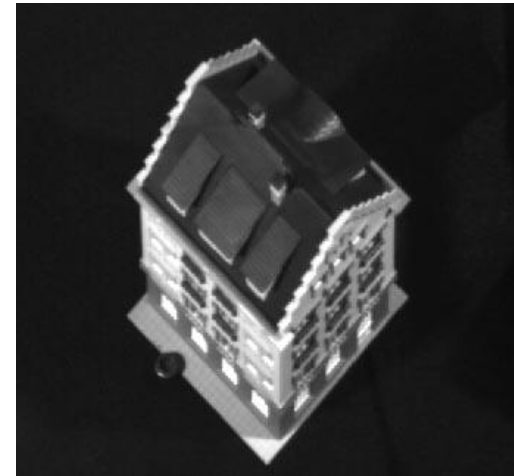
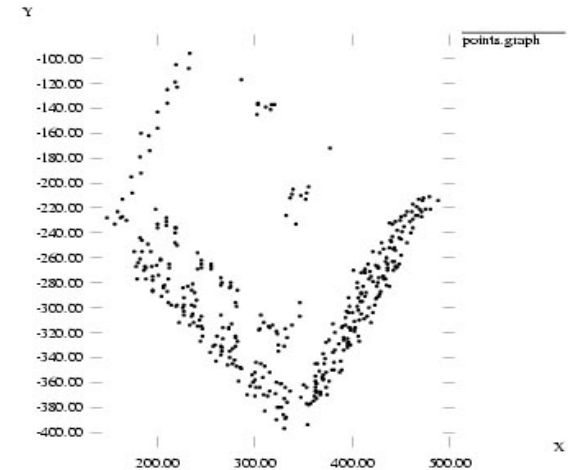
60



120



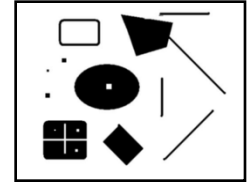
150



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

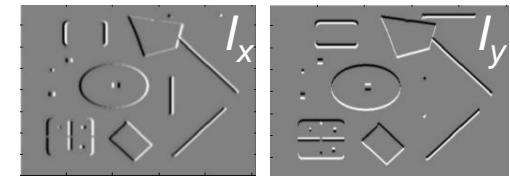
HW 4: Problem 2 summary

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

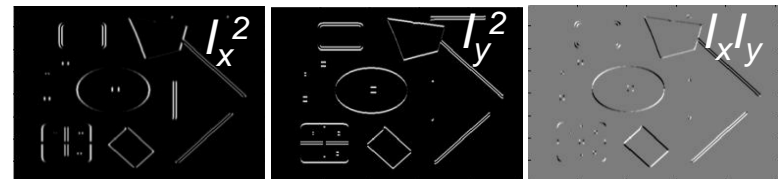


$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives



2. Square of derivatives



3. Gaussian filter $g(\sigma_I)$



$$\det M = \lambda_1 \lambda_2$$

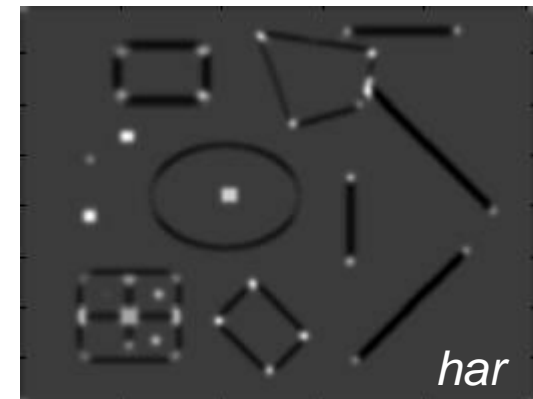
$$\text{trace } M = \lambda_1 + \lambda_2$$

4. Cornerness function – both eigenvalues are strong

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha [\text{trace}(\mu(\sigma_I, \sigma_D))]^2 =$$

$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha [g(I_x^2) + g(I_y^2)]^2$$

5. Non-maxima suppression



HW 4: Problem 2 summary

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) : $x' = x' + u$; $y' = y' + v$;

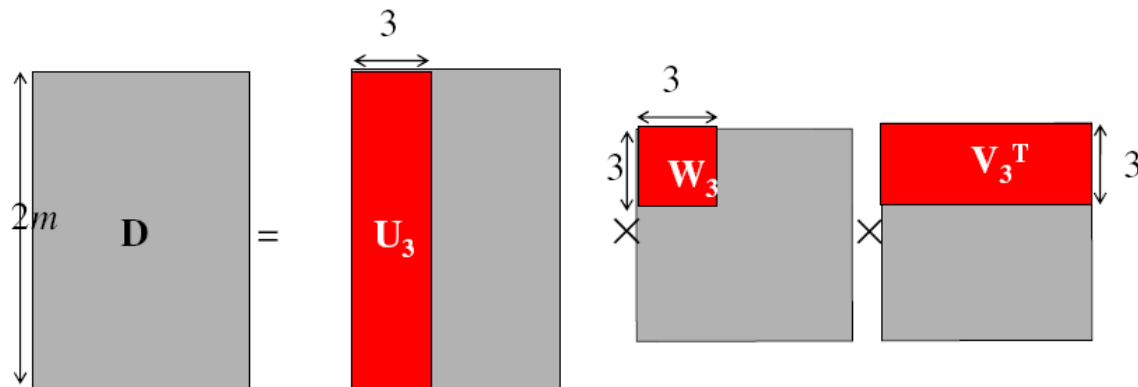
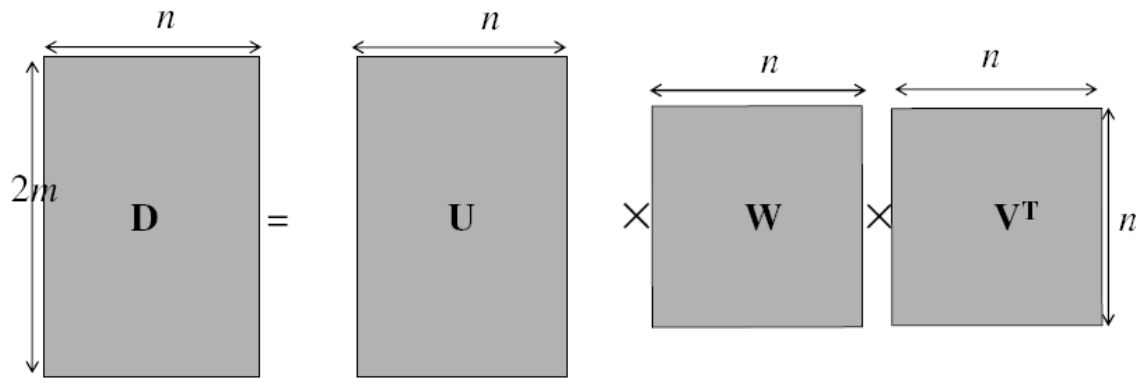
d) Recalculate I_t


e) Repeat steps 2-4 until small change

- Use interpolation for subpixel values

HW 4: Problem 2 summary

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



 Solve for orthographic constraints

HW 4: Problem 2 summary

- 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)
 - If you get stuck on one part, can the included intermediate results
 - Get tracking working on one point for a few frames before trying to get it working for all points
- Extra problems
 - Either for fun, or if you weren't able to complete earlier homeworks
 - Affine verification
 - Missing track completion
 - Optical flow
 - Coarse-to-fine tracking

See you next week

- Object tracking
- Action recognition