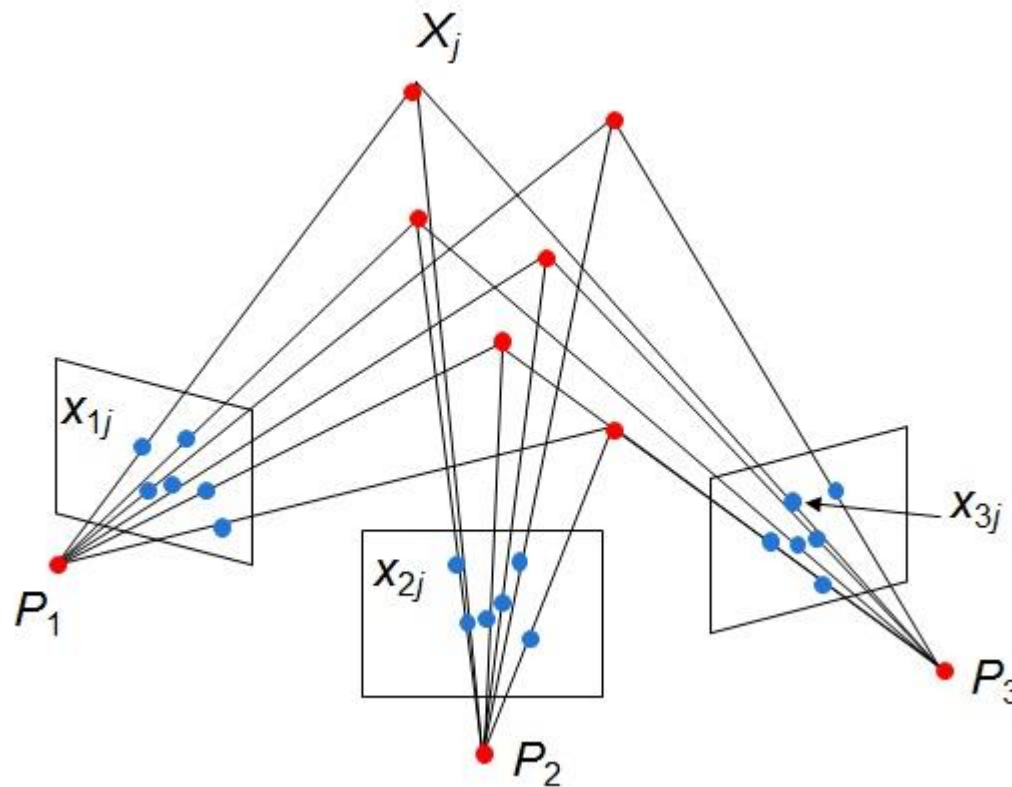


Structure from Motion



Jia-Bin Huang, Virginia Tech

Administrative stuffs

- HW 3 due 11:59 PM, Oct 17 (Monday)
- Top alignment results
 - Mengyu Song (5.03)
 - second moment ellipse + ICP
 - Badour AlBahar (5.41)
 - Tested the original image, multiple initializations
 - Sujay Yadawadkar (6.989)
 - Iterative closest point with affine transformation
- Feedback
 - Detailed discussions on HW assignment
 - More generous on hints

Perspective and 3D Geometry

- **Projective geometry and camera models**

- Vanishing points/lines
- $x = K[R \ t]X$

- **Single-view metrology and camera calibration**

- Calibration using known 3D object or vanishing points
- Measuring size using perspective cues

- **Photo stitching**

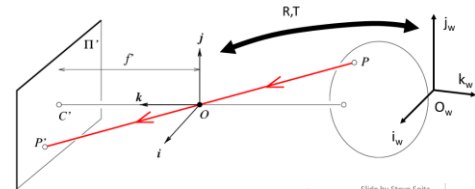
- Homography relates rotating cameras $x' = Hx$
- Recover homography using RANSAC + normalized DLT

- **Epipolar Geometry and Stereo Vision**

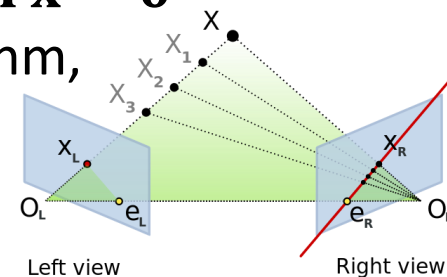
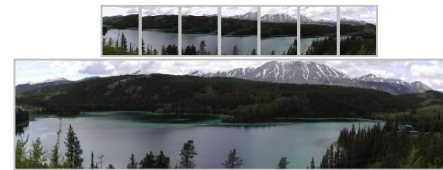
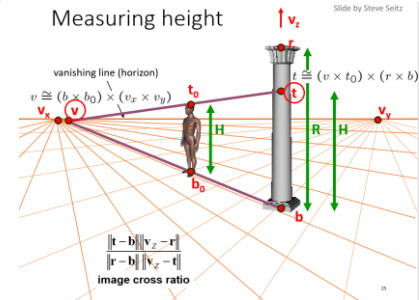
- Fundamental/essential matrix relates two cameras $x'Fx = 0$
- Recover F using RANSAC + normalized 8-point algorithm, enforce rank 2 using SVD

- **Structure from motion (this class)**

- How can we recover 3D points from multiple images?

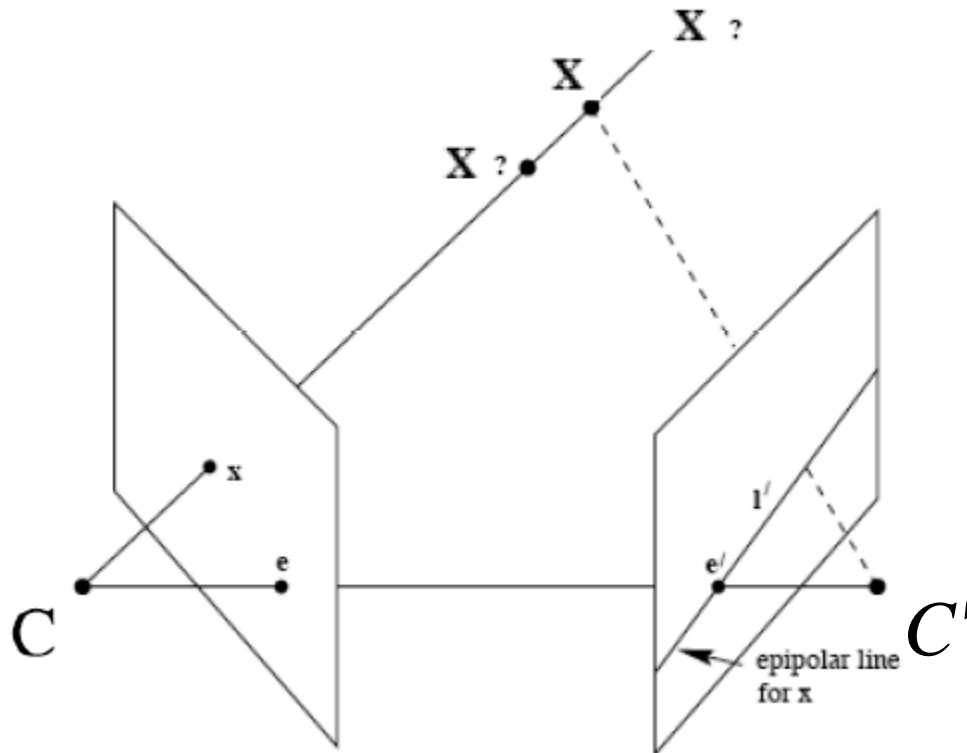


Measuring height



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: Automatic Estimation of F

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

8-Point Algorithm for Recovering F

- 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

- Randomly sample 8 points
- Compute F via least squares
- Enforce $\det(\tilde{\mathbf{F}}) = 0$ by SVD
- Repeat and choose F with most inliers

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

Recap

- We can get projection matrices P and P' up to a projective ambiguity (see HZ p. 255-256)

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

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];
```

This class: Structure from Motion

- Projective structure from motion
- Affine structure from motion
- HW 3
- Multi-view stereo (optional)

Structure ['stræk(t)SHər]:

3D Point Cloud of the Scene

Motion ['mōSH(ə)n]:

Camera Location and Orientation

Structure from Motion (SfM)

Get the Point Cloud from Moving
Cameras

SfM Applications – 3D Modeling



SfM Applications – Surveying cultural heritage structure analysis



SfM Applications – Robot navigation and mapmaking



SfM Applications – Visual effect (matchmove)



Steps

Images \rightarrow Points: Structure from Motion

Points \rightarrow More points: Multiple View Stereo

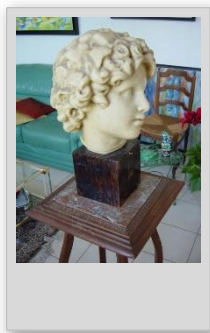
Points \rightarrow Meshes: Model Fitting

Meshes \rightarrow Models: Texture Mapping

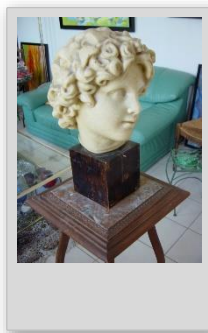
+

=

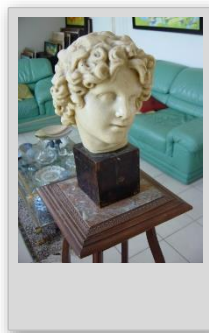
Images \rightarrow Models: Image-based Modeling



+



+



=



Steps

Images \rightarrow Points: Structure from Motion

Points \rightarrow More points: Multiple View Stereo

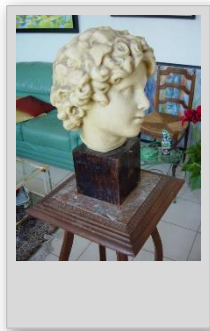
Points \rightarrow Meshes: Model Fitting

Meshes \rightarrow Models: Texture Mapping

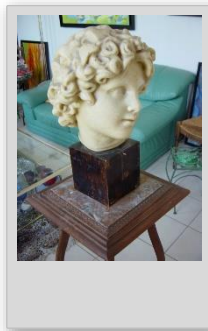
+

=

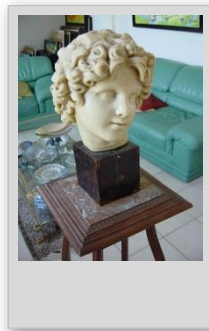
Images \rightarrow Models: Image-based Modeling



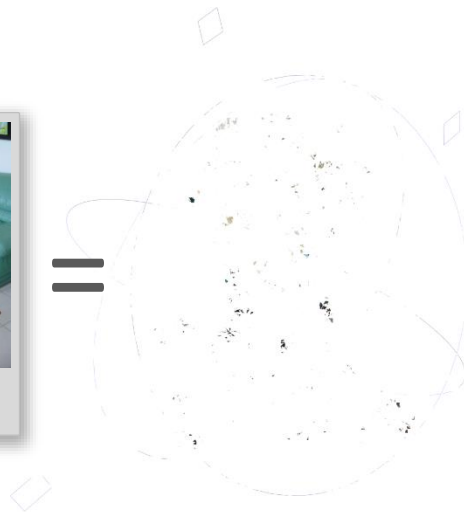
+



+



=



Steps

Images \rightarrow Points:

Structure from Motion

Points \rightarrow More points:

Multiple View Stereo

Points \rightarrow Meshes:

Model Fitting

Meshes \rightarrow Models:

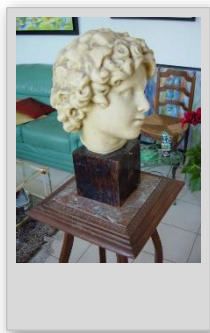
Texture Mapping

+

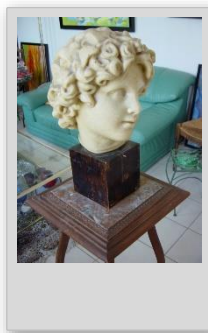
=

Images \rightarrow Models:

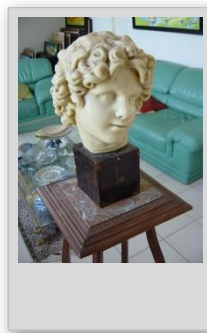
Image-based Modeling



+



+



+



=



Steps

Images \rightarrow Points:

Structure from Motion

Points \rightarrow More points:

Multiple View Stereo

Points \rightarrow Meshes:

Model Fitting

Meshes \rightarrow Models:

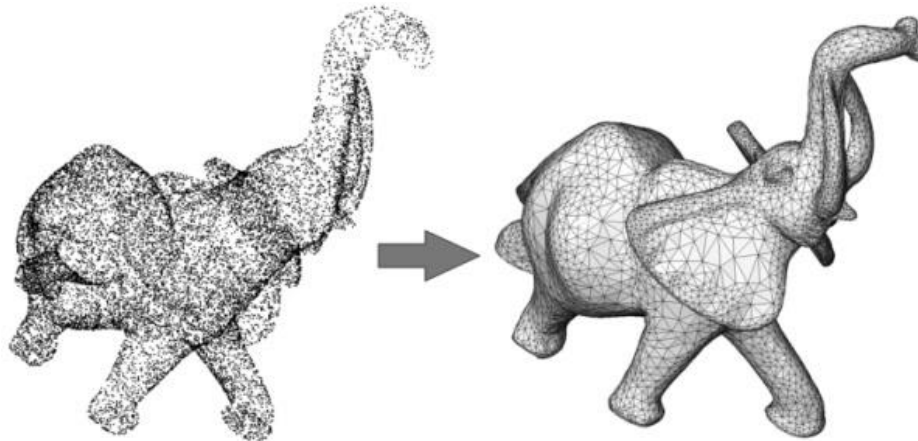
Texture Mapping

+

=

Images \rightarrow Models:

Image-based Modeling



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



Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

Meshes → Models: Texture Mapping

+

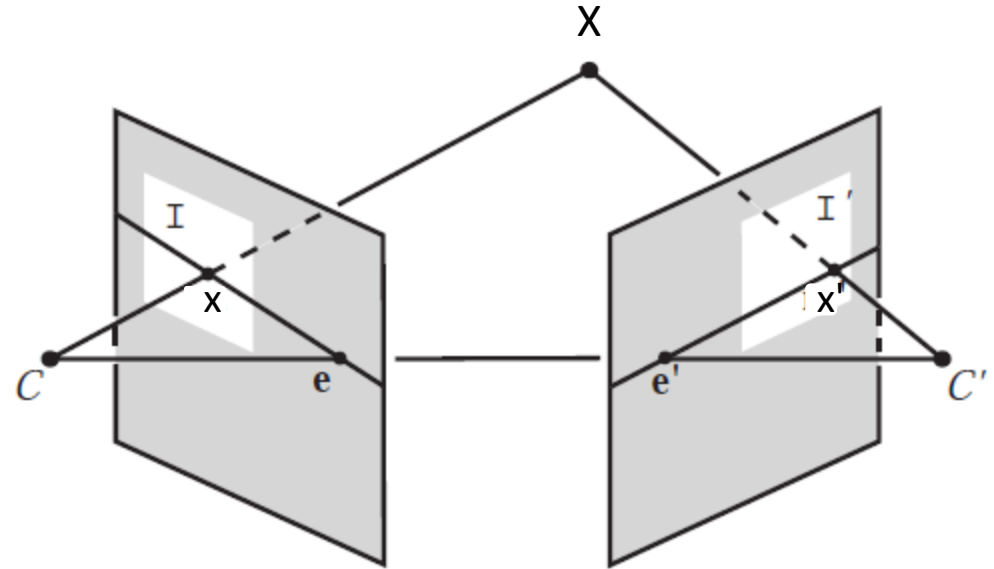
=

Images → Models: Image-based Modeling

Example: <https://photosynth.net/>

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



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

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



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

Triangulation: Linear Solution

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

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

Pros and Cons

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

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

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

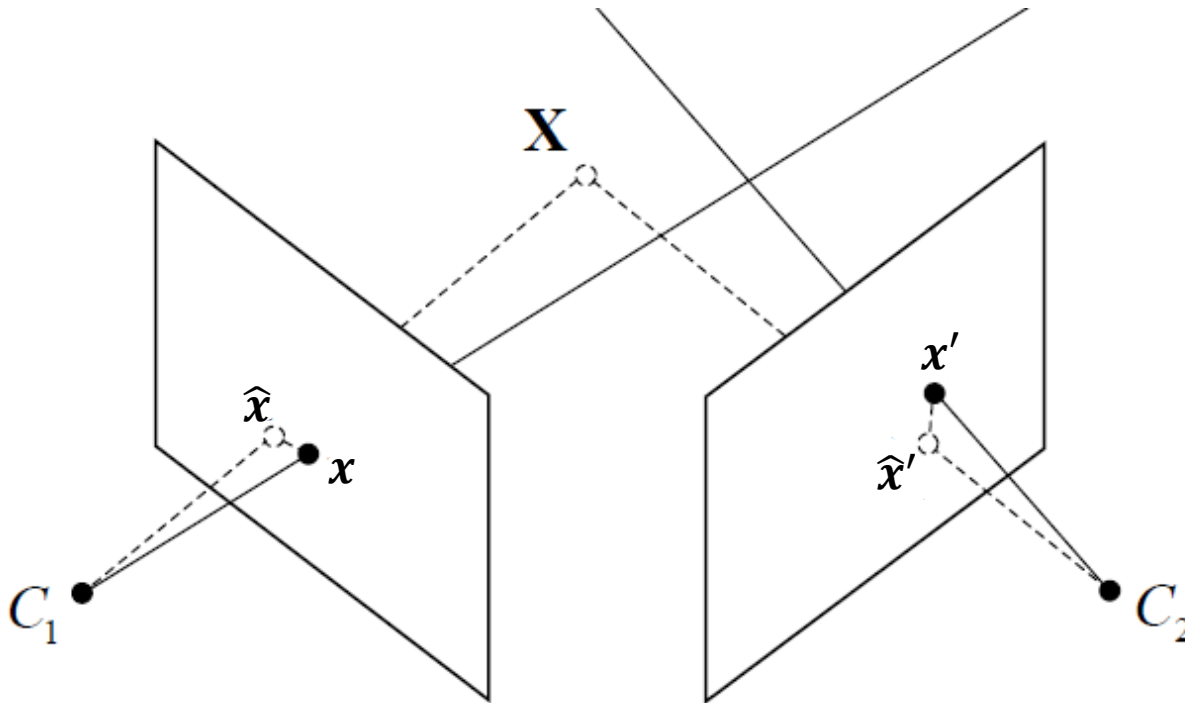
$$\mathbf{A} = \begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1^T \\ v\mathbf{p}_3^T - \mathbf{p}_2^T \\ u'\mathbf{p}'_3^T - \mathbf{p}'_1^T \\ v'\mathbf{p}'_3^T - \mathbf{p}'_2^T \end{bmatrix}$$

Triangulation: Non-linear Solution

- Minimize projected error while satisfying

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

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

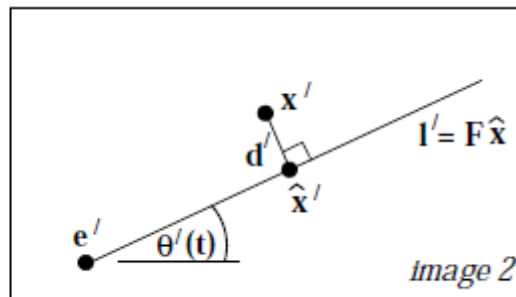
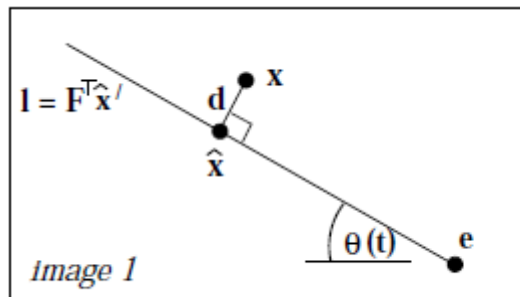


Triangulation: Non-linear Solution

- Minimize projected error while satisfying

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

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



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

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

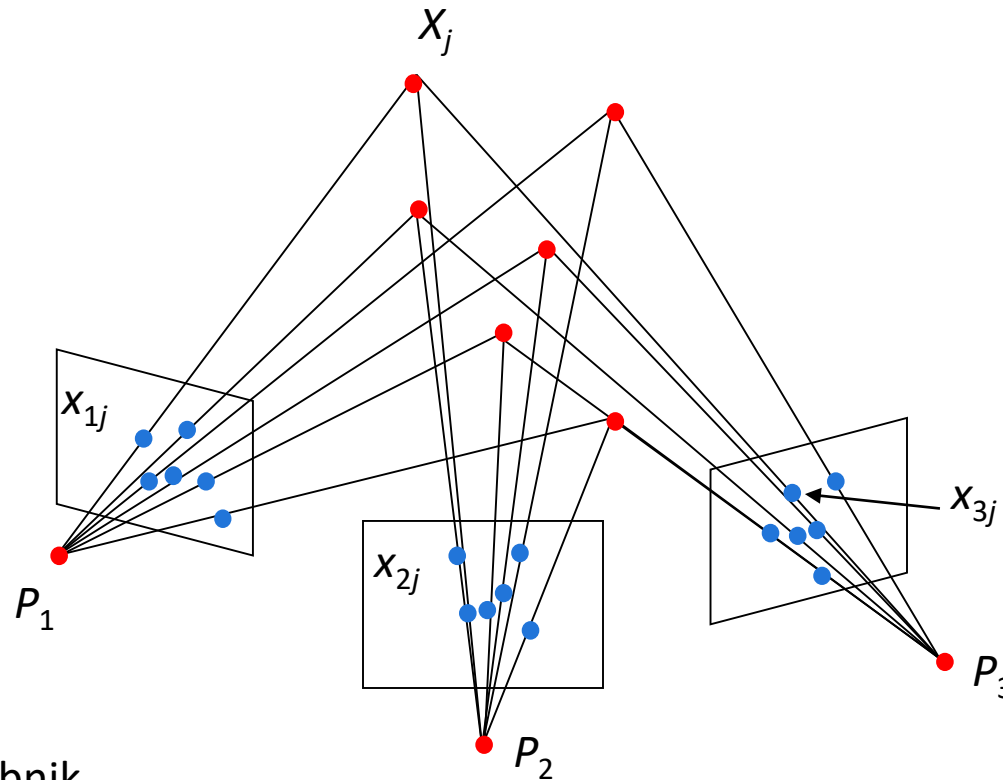
Further reading: HZ p. 318

Projective structure from motion

- Given: m images of n fixed 3D points

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

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



Projective structure from motion

- Given: m images of n fixed 3D points

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

- Problem:

- Estimate unknown m projection matrices \mathbf{P}_i and n 3D points \mathbf{X}_j from the known 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} :

$$\bullet \mathbf{X} \rightarrow \mathbf{QX}, \mathbf{P} \rightarrow \mathbf{PQ}^{-1}$$

- We can solve for structure and motion when

$$2mn \geq 11m + 3n - 15$$

DoF in \mathbf{P}_i

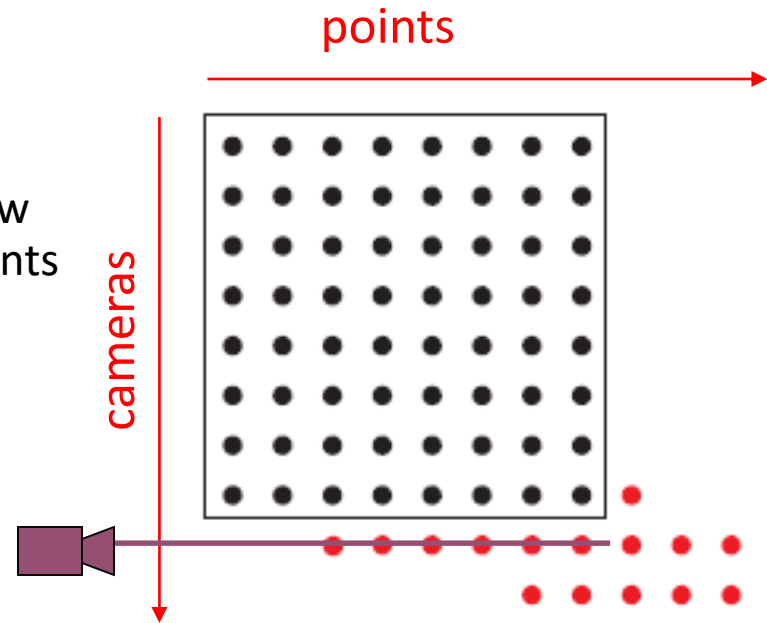
DoF in \mathbf{X}_j

Up to 4x4 projective tform \mathbf{Q}

- For two cameras, at least 7 points are needed

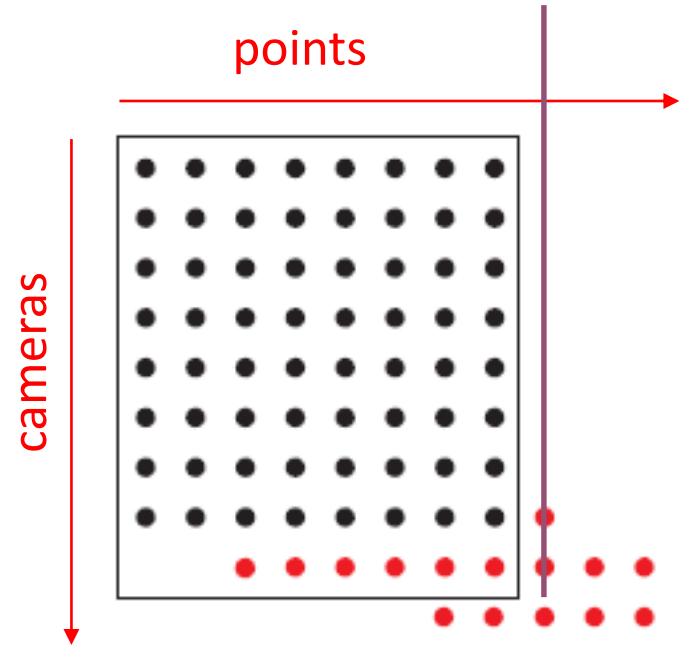
Sequential structure from motion

- 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*



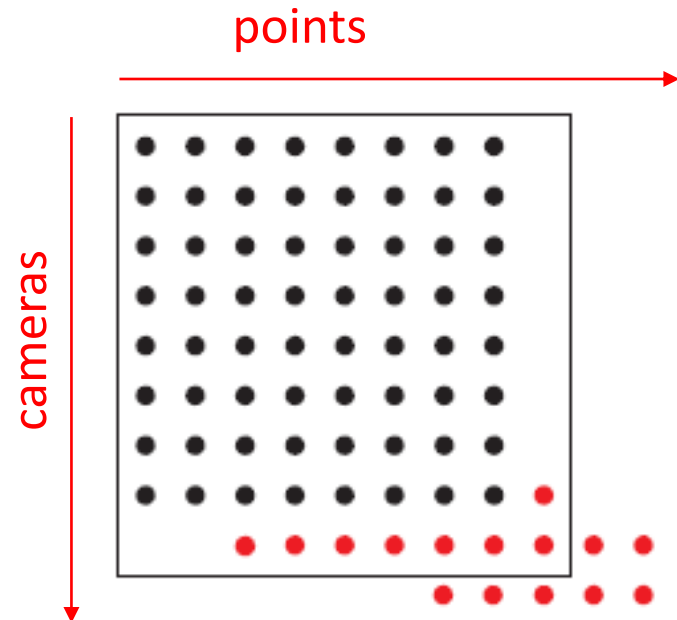
Sequential structure from motion

- 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

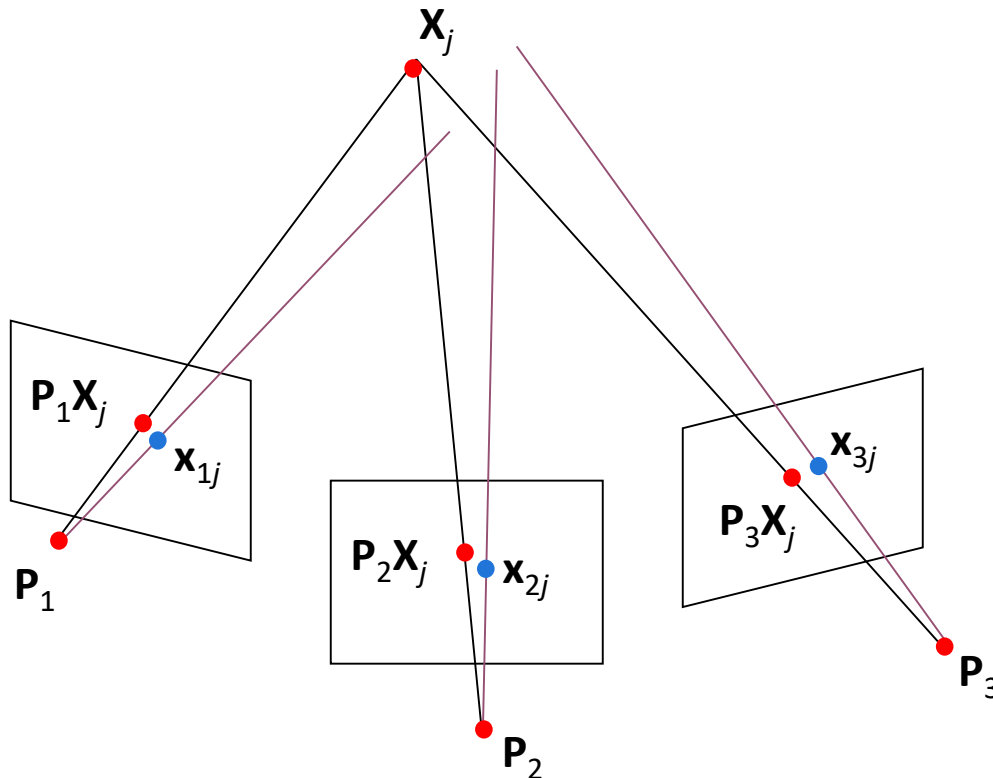
- Initialize motion from two images using fundamental matrix
- Initialize structure by triangulation
- For each additional view:
 - Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
 - Refine and extend structure: compute new 3D points,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

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



- 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 \mathbf{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

- From two images, we can:
 - Recover fundamental matrix F
 - Recover canonical camera projection matrix 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

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)

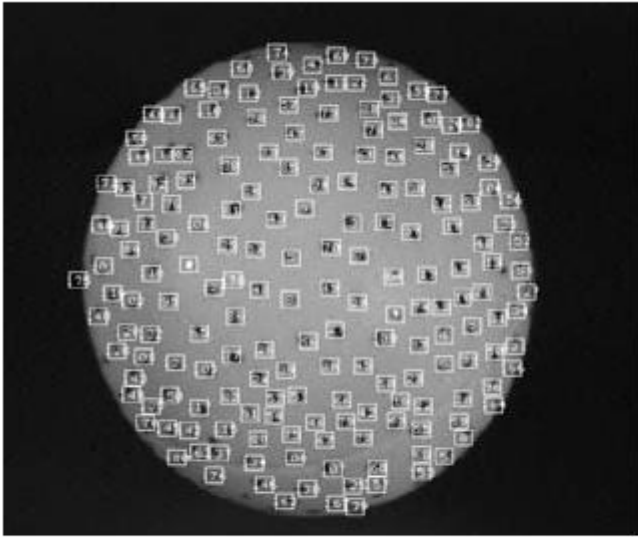
(best method with software available; also has good overview of recent methods)

[Reconstruction of Cornell](#) (Crandall et al. ECCV 2011)

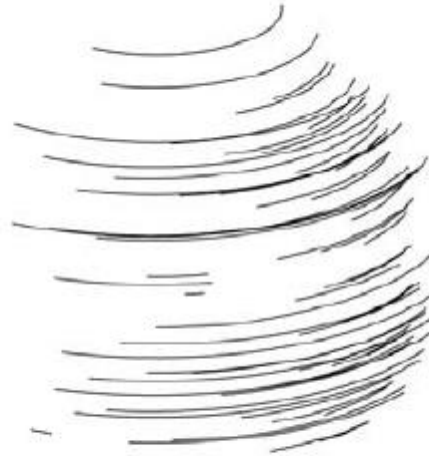
3D from multiple images



Structure from motion under orthographic projection



(a)



(b)



(c)

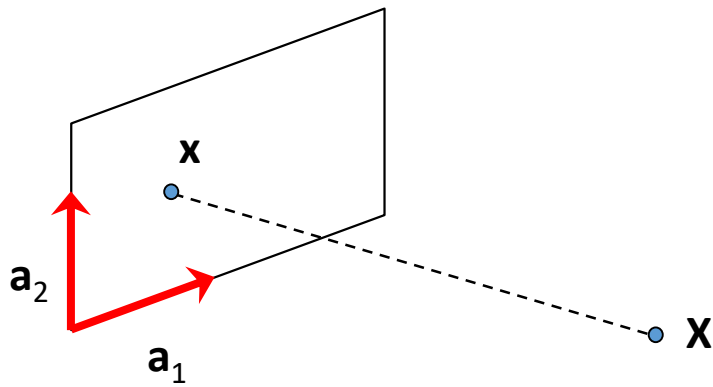
3D Reconstruction of a Rotating Ping-Pong Ball

- 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

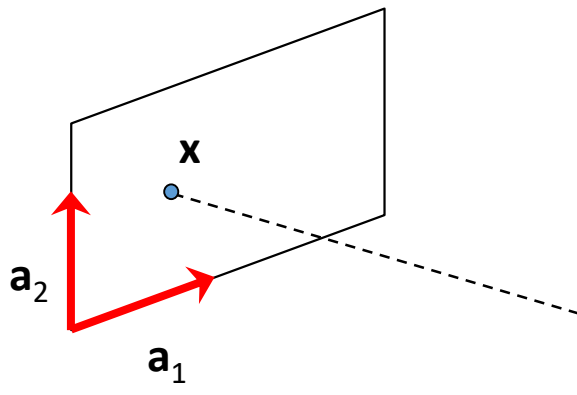


$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(R'_f \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix}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

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

Affine structure from motion

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



The diagram illustrates the affine projection process. On the left, a 3D plane is shown with two red vectors, \mathbf{a}_1 and \mathbf{a}_2 , originating from a point. A blue dot labeled \mathbf{x} is located on the plane. A dashed line extends from this point towards the right, where it meets another blue dot labeled \mathbf{X} . This represents the projection of the 3D point \mathbf{X} onto the 2D plane, resulting in the 2D point \mathbf{x} .

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

Projection of world origin

1. We are given corresponding 2D points (\mathbf{x}) in several frames
2. 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

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



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



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

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 (2mx3)

3D Points (3xn)

The diagram illustrates the computation of 2D positions from 3D points and camera parameters. It features a vertical column of camera parameter matrices $\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_m$ on the left, and a horizontal row of 3D point vectors $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n$ on the right. A blue arrow points from the label 'Camera Parameters (2mx3)' to the column of \mathbf{A} matrices. Another blue arrow points from the label '3D Points (3xn)' to the row of \mathbf{X} vectors. The two matrices are positioned to be multiplied together.

What if we instead observe corresponding 2d image points?

Can we recover the camera parameters and 3d points?

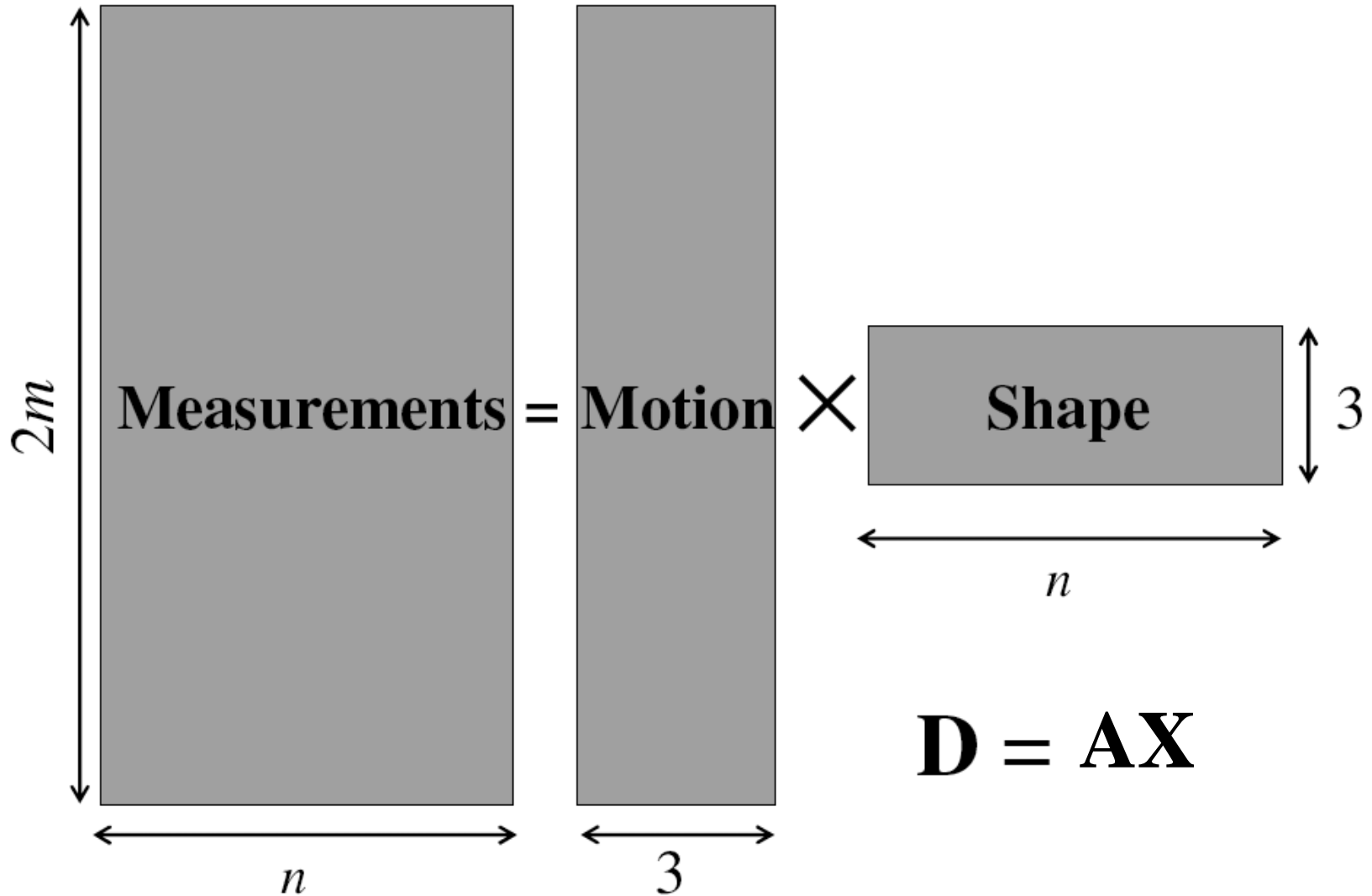
$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn}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

cameras (2m)

points (n)

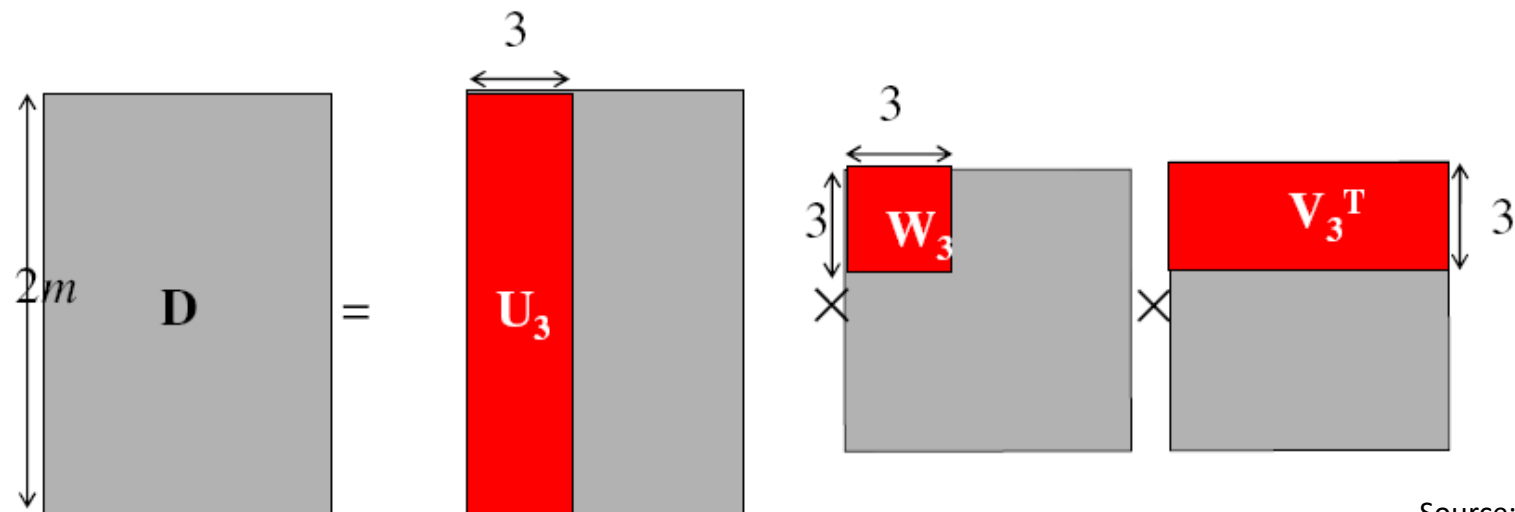
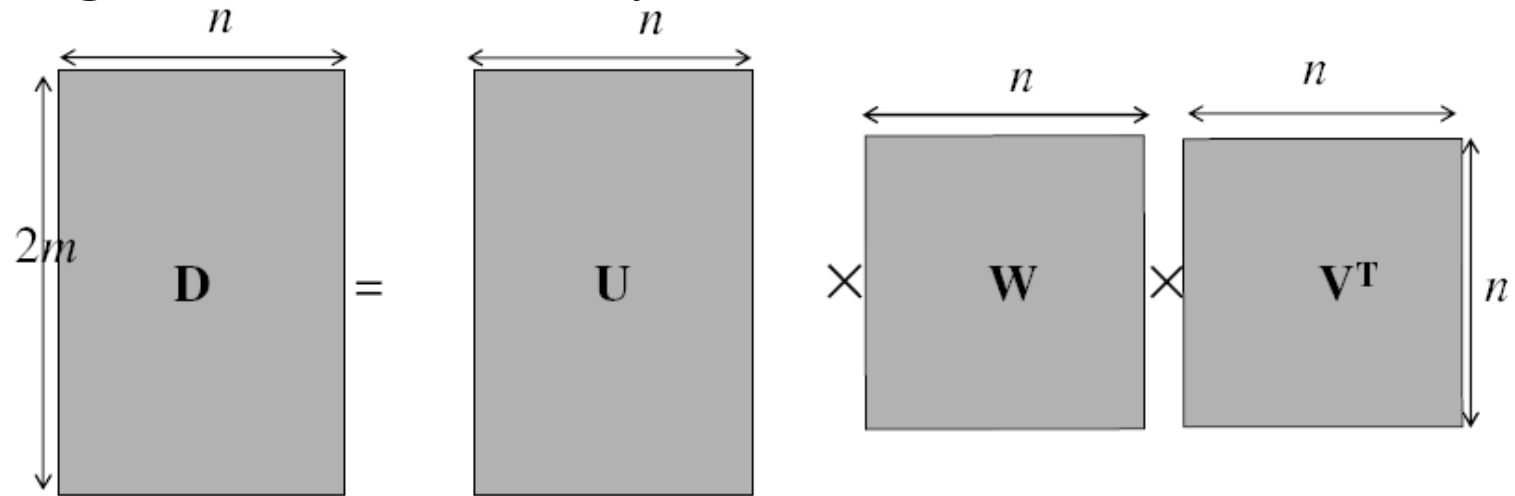
What rank is the matrix of 2D points?

Factorizing the measurement matrix



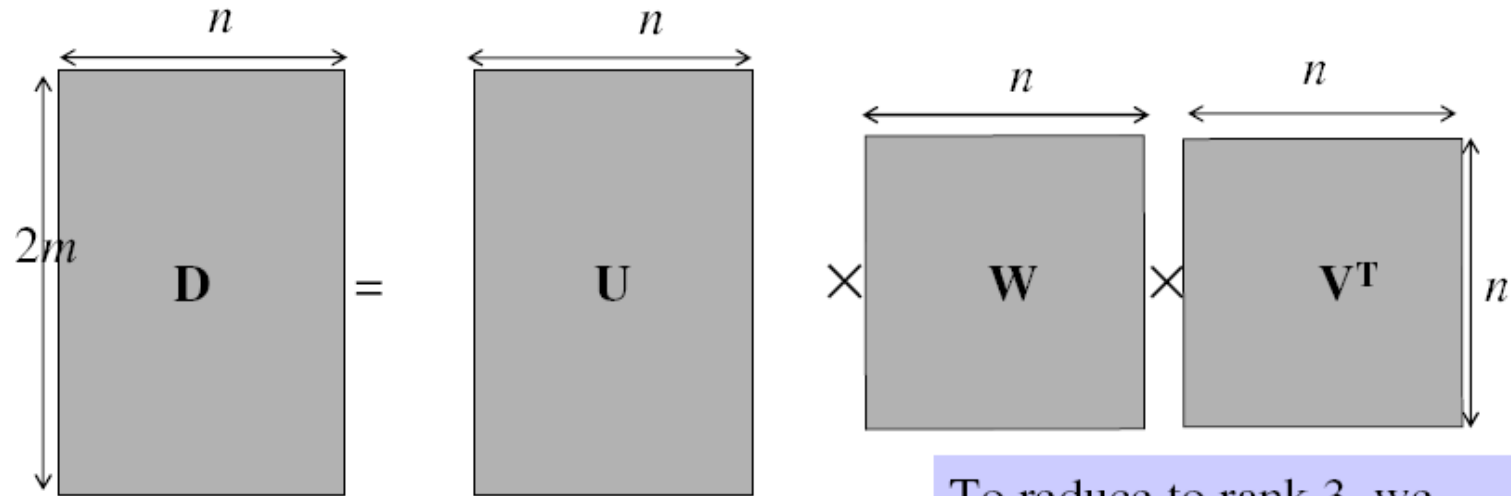
Factorizing the measurement matrix

- Singular value decomposition of D :

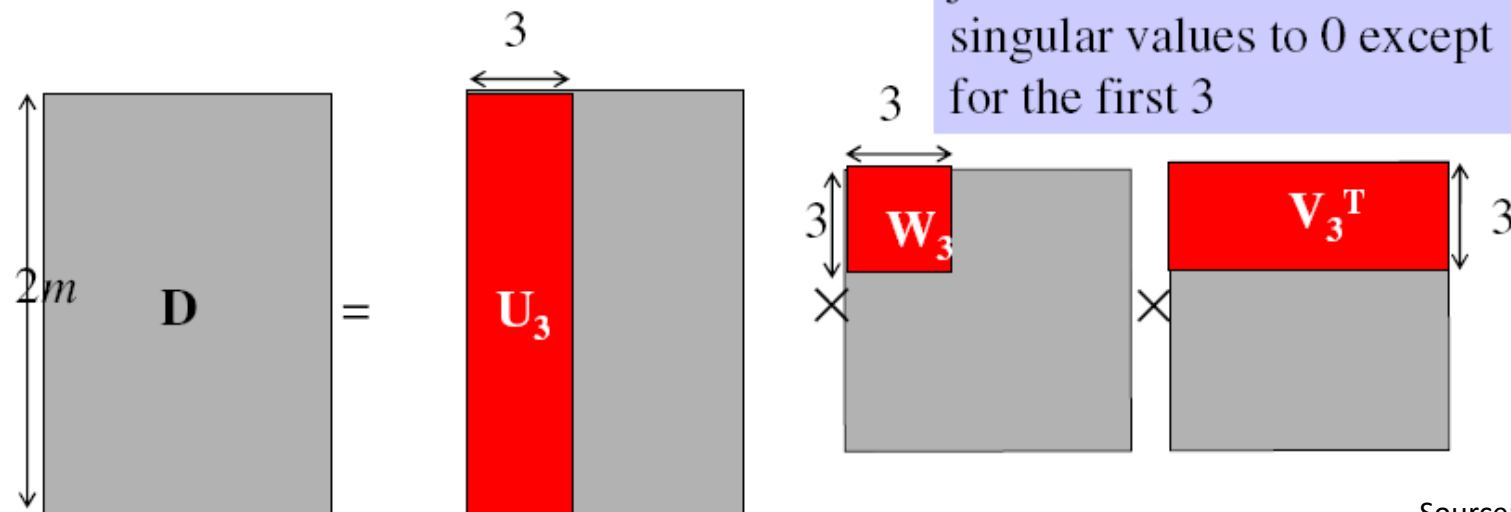


Factorizing the measurement matrix

- 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:

The diagram illustrates the SVD factorization of a measurement matrix D . The matrix D is represented by a gray rectangle with a vertical double-headed arrow to its left labeled $2m$ and the label D in the center. It is followed by an equals sign. To the right of the equals sign is a red rectangle labeled U_3 with a horizontal double-headed arrow below it labeled 3 . This is followed by a multiplication symbol \times and a vertical double-headed arrow labeled 3 . Then is a red rectangle labeled W_3 with a horizontal double-headed arrow above it labeled 3 and a vertical double-headed arrow to its left labeled 3 . This is followed by another multiplication symbol \times and a red rectangle labeled V_3^T with a horizontal double-headed arrow above it labeled n and a vertical double-headed arrow to its right labeled 3 .

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

Factorizing the measurement matrix

- Obtaining a factorization from SVD:

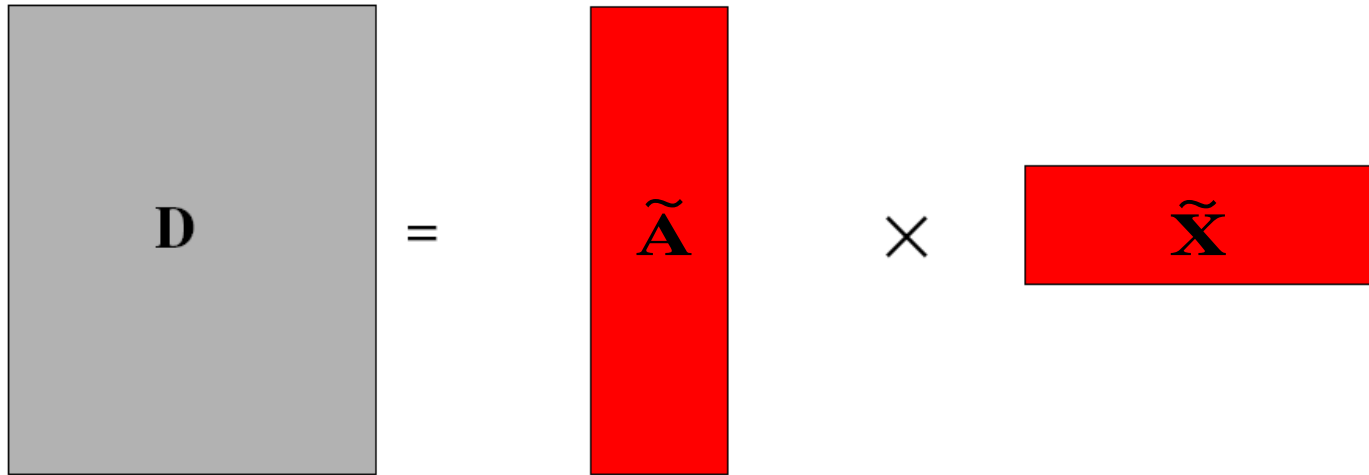
$$\begin{array}{c} 2m \\ \downarrow \\ \mathbf{D} \end{array} = \begin{array}{c} \mathbf{U}_3 \\ \leftarrow 3 \end{array} \times \begin{array}{c} 3 \\ \leftarrow \\ \mathbf{W}_3 \\ \rightarrow 3 \end{array} \times \begin{array}{c} n \\ \leftarrow \\ \mathbf{V}_3^T \\ \rightarrow 3 \end{array}$$

Possible decomposition:

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

$$\mathbf{D} = \begin{array}{c} \tilde{\mathbf{A}} \\ \leftarrow 3 \end{array} \times \begin{array}{c} \tilde{\mathbf{X}} \\ \leftarrow n \end{array}$$

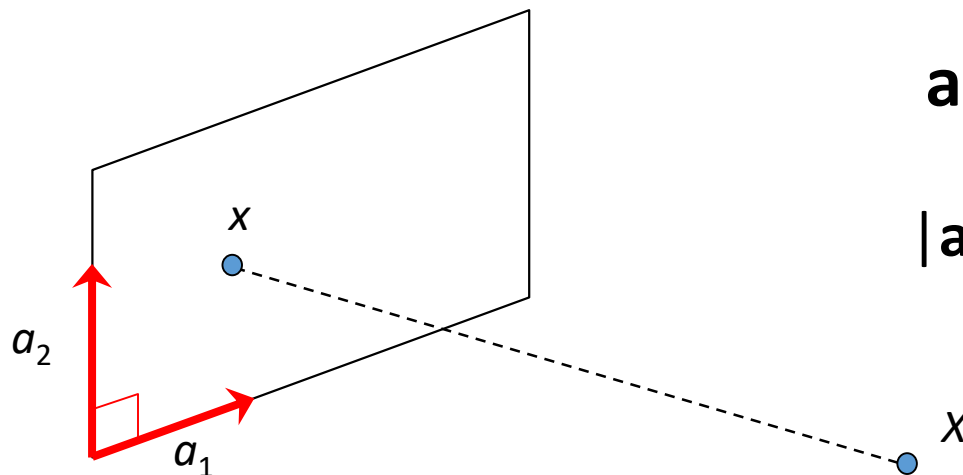
Affine ambiguity


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

- 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



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

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

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

How to solve $L = CC^T$?

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



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

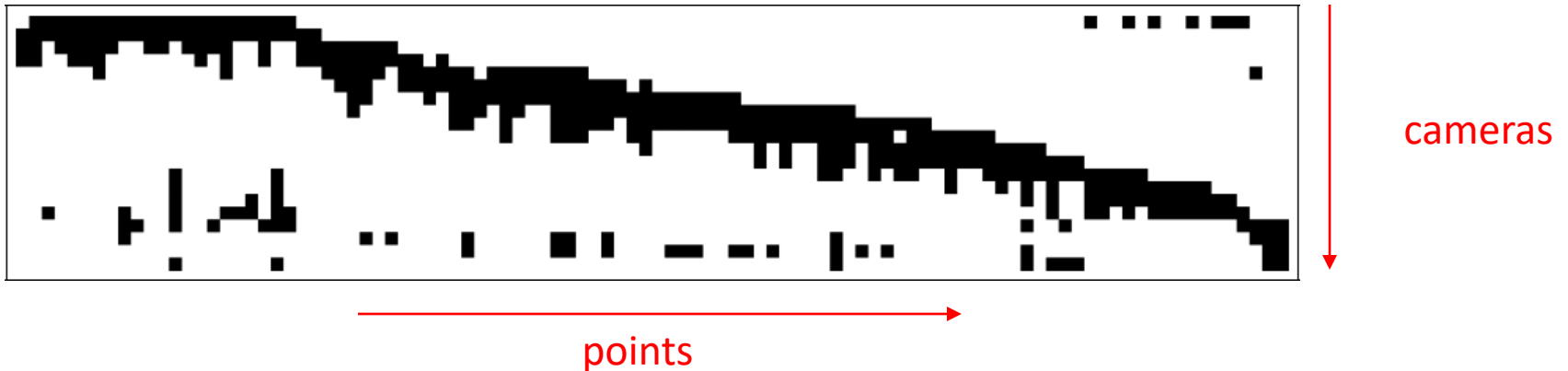
`reshape([a b c]'*[d e f], [1, 9])`

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: $\mathbf{D} = \mathbf{U} \mathbf{W} \mathbf{V}^T$
 - 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:
$$\mathbf{A} = \mathbf{U}_3 \mathbf{W}_3^{1/2} \text{ and } \mathbf{S} = \mathbf{W}_3^{1/2} \mathbf{V}_3^T$$
- Eliminate affine ambiguity
 - Solve $\mathbf{L} = \mathbf{C}\mathbf{C}^T$ using metric constraints
 - Solve \mathbf{C} using Cholesky decomposition
 - Update \mathbf{A} and \mathbf{X} : $\mathbf{A} = \mathbf{A}\mathbf{C}$, $\mathbf{S} = \mathbf{C}^{-1}\mathbf{S}$

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



1



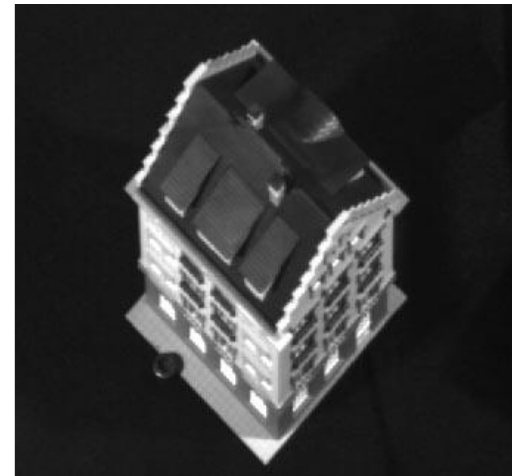
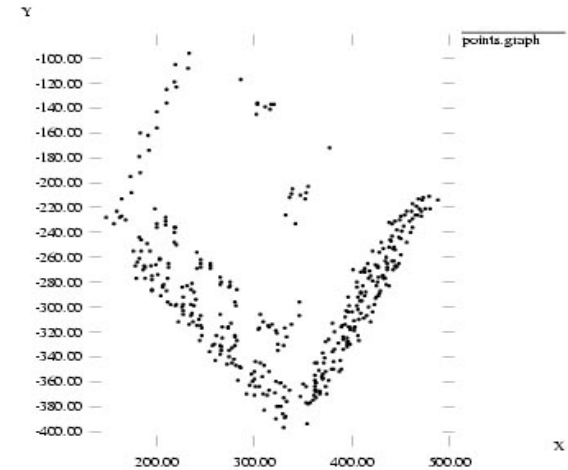
60



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 Lischinski and Gruber

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

- Clear explanation of epipolar geometry and projective SfM

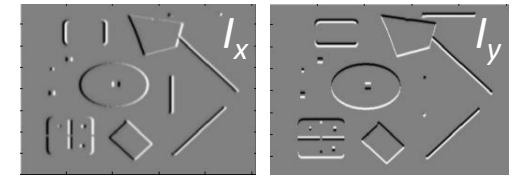
- <http://mi.eng.cam.ac.uk/~cipolla/publications/contributionToEditedBook/2008-SFM-chapters.pdf>

Review of Affine SfM from Interest Points

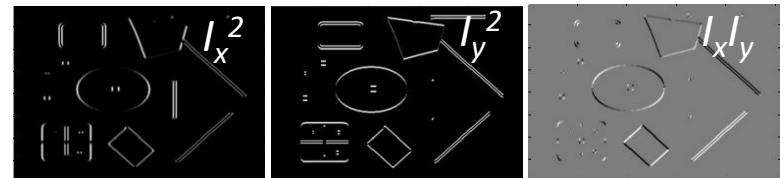
1. Detect interest points (e.g., Harris)

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

1. Image
derivatives



2. Square of
derivatives



3. Gaussian
filter $g(\sigma_I)$

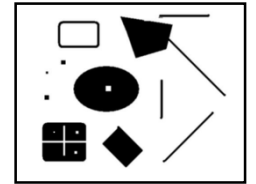
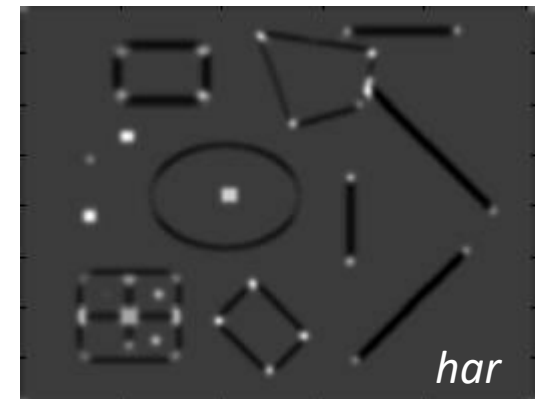


4. Cornerness function – both eigenvalues are strong

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

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

5. Non-maxima suppression



Review of Affine SfM from Interest Points

2. Correspondence via Lucas-Kanade tracking

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

b) Compute (u, v) by

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

2nd moment matrix for feature patch in first image

displacement

Original (x, y) position

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

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

Diagram illustrating the decomposition of matrix D into U , W , and V^T . Matrix D is a $2m \times n$ matrix. It is equal to matrix U (size $n \times n$) multiplied by matrix W (size $n \times n$) multiplied by matrix V^T (size $n \times n$).

Diagram illustrating the decomposition of matrix D into U_3 , W_3 , and V_3^T . Matrix D is a $2m \times n$ matrix. It is equal to matrix U_3 (size $3 \times n$, highlighted in red) multiplied by matrix W_3 (size 3×3 , highlighted in red) multiplied by matrix V_3^T (size $3 \times n$, highlighted in red).

 Solve for
orthographic
constraints

HW 3 – Part 1-A. Vanishing points

```
% Load the image
im = imread('new_classroom_building.jpg');

% Manually select at least three lines
(press q to stop)

vp = getVanishingPoint(im);
```

lines - $[3 \times N]$, $N \geq 3$
line equation (a, b, c): $au + bv + c = 0$

Problem: Solving the VPs using lines

1. Find points at the intersections of each pair of lines. Take the mean as your VP.
-> less accurate
2. Find a point that minimizes the sum of the distances to the lines. Solve for VP using $A \backslash b$:

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \\ a_3 & b_3 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

3. Maximum likelihood estimate to minimize averaged angular differences (L-M)



Write-up

- Plot the VPs and the lines used to estimate them on the image plane.
- Specify the three VPs (u,v) in the image plane
- Plot the ground horizon line and specify its normalized parameters: $au + bv + c = 0$



HW 3 – Part 1-B. Finding K

$$\mathbf{VPs} = [\mathbf{vp1}, \mathbf{vp2}, \mathbf{vp3}] - [3 \times 3]$$

Unknown camera parameters f, u_0, v_0

Problem: Solving intrinsic matrix \mathbf{K}

Orthogonality constraints $\mathbf{X}_i^\top \mathbf{X}_j = \mathbf{0}$

VP (3D)

VP (2D)

$$\mathbf{X}_i = \mathbf{R}^{-1} \mathbf{K}^{-1} \mathbf{p}_i$$

$$\mathbf{p}_i^\top (\mathbf{K}^{-1})^\top (\mathbf{K}^{-1}) \mathbf{p}_j = 0$$

$$\mathbf{K} = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$\mathbf{K}^{-1} = \begin{bmatrix} \frac{1}{f} & 0 & -\frac{u_0}{f} \\ 0 & \frac{1}{f} & -\frac{v_0}{f} \\ 0 & 0 & 1 \end{bmatrix}$$

Approach 1:

- Closed-form solution: solve u_0, v_0 first and then solve f

Approach 2:

- Use numerical solver, e.g., `fsolve`

Write-up

- Show the process of finding camera focal length and optical center
- Report the estimated camera focal length (f) and optical center (u_0, v_0).

HW 3 – Part 1-C. Finding R

$$\mathbf{VPs} = [\mathbf{vp1}, \mathbf{vp2}, \mathbf{vp3}] - [3 \times 3]$$

Problem: Solving rotation matrix \mathbf{R}

$$\text{Rotation matrix } \mathbf{R} = [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{r}_3]$$

$$\mathbf{p}_i = \mathbf{KRX}_i$$

Set directions of vanishing points

$$\mathbf{X}_1 = [\mathbf{1}, \mathbf{0}, \mathbf{0}]^T$$

$$\mathbf{X}_2 = [\mathbf{0}, \mathbf{1}, \mathbf{0}]^T$$

$$\mathbf{X}_3 = [\mathbf{0}, \mathbf{0}, \mathbf{1}]^T$$

$$\mathbf{p}_1 = \mathbf{K}\mathbf{r}_1$$

$$\mathbf{p}_2 = \mathbf{K}\mathbf{r}_2$$

$$\mathbf{p}_3 = \mathbf{K}\mathbf{r}_3$$

Special properties of \mathbf{R}

- $\text{inv}(\mathbf{R}) = \mathbf{R}^T$
- Each row and column of \mathbf{R} has unit length

Write-up

- Describe how to compute the camera's rotation matrix.
- Compute the rotation matrix for this image, vertical VP = $[0, 1, 0]$, the right-most VP = $[1, 0, 0]$, left-most VP = $[0, 0, 1]$.

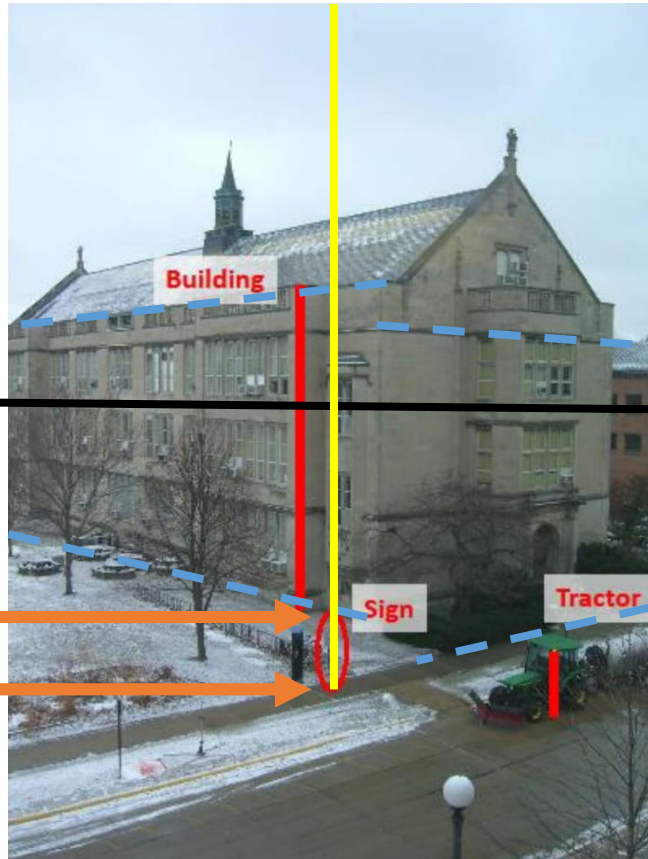
HW 3 – Part 1-D. Single-view metrology

Problem: Estimate the height of building, tractor, and camera

Height of sign = 1.65m

$$\frac{\|\mathbf{t} - \mathbf{b}\| \|\mathbf{v}_Z - \mathbf{r}\|}{\|\mathbf{r} - \mathbf{b}\| \|\mathbf{v}_Z - \mathbf{t}\|} = \frac{H}{R}$$

image cross ratio



Write-up

- Turn in an illustration that shows the horizon line, and the lines and measurements used to estimate the heights of the building, tractor, and camera.
- Report the estimated heights of the building, tractor, and camera in meters.

HW 3 – Part 2 Epipolar Geometry

Problem: recover F from matches with outliers

```
load matches.mat
```

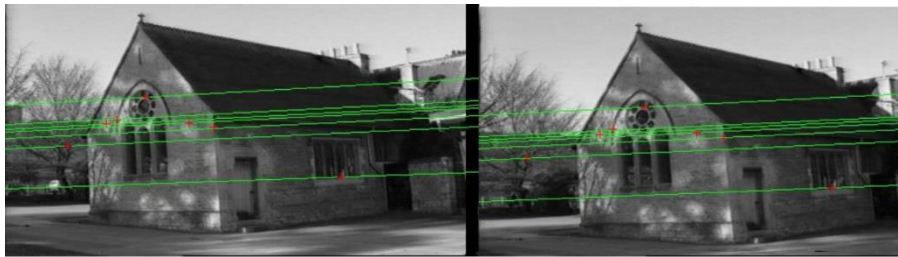
```
[c1, r1] - 477 x 2
```

```
[c2, r2] - 500 x 2
```

```
matches - 252 x 2
```

```
matches(:,1): matched point in im1
```

```
matches(:,2): matched point in im2
```



Write-up:

- Describe what test you used for deciding inlier vs. outlier.
- Display the estimated fundamental matrix F after normalizing to unit length
- Plot the outlier keypoints with green dots on top of the first image plot(x , y , 'g');
- Plot the corresponding epipolar lines

8-Point Algorithm for Recovering F

- Correspondence Relation

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

1. Normalize image coordinates

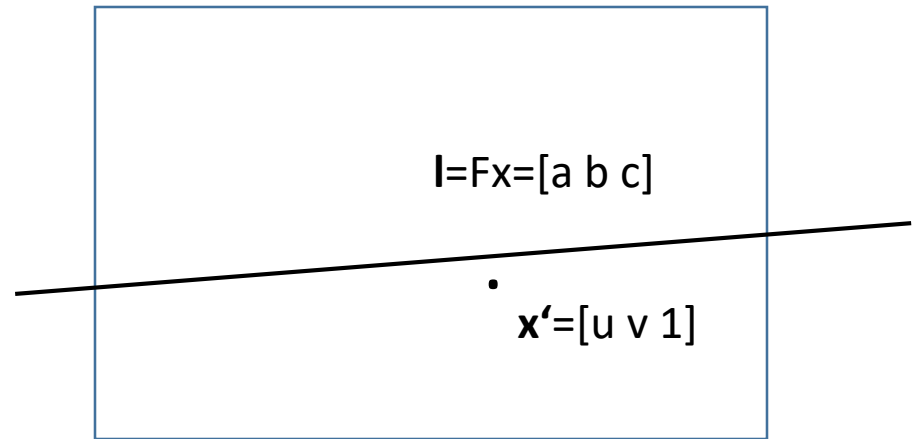
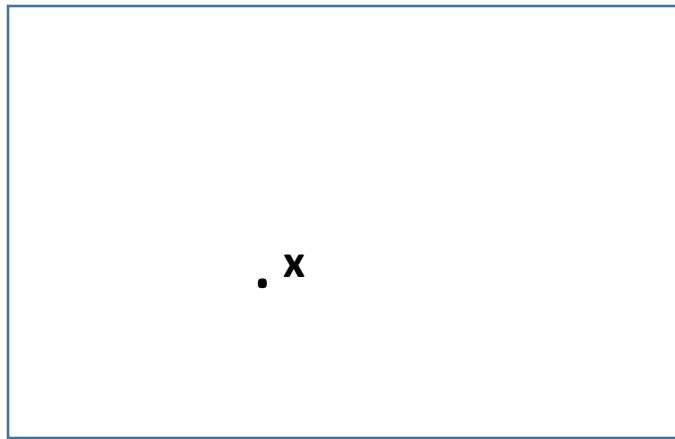
$$\tilde{\mathbf{x}} = \mathbf{T} \mathbf{x} \quad \tilde{\mathbf{x}}' = \mathbf{T}' \mathbf{x}'$$

2. RANSAC with 8 points

- Randomly sample 8 points
- Compute F via least squares
- Enforce $\det(\tilde{\mathbf{F}}) = 0$ by SVD
- Repeat and choose F with most inliers

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

Distance of point to epipolar line



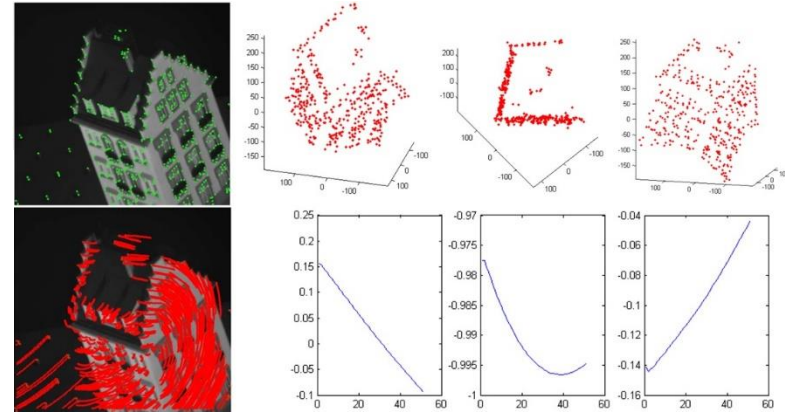
$$d(l, x') = \frac{|au + bv + c|}{\sqrt{a^2 + b^2}}$$

HW 3 – Part 3 Affine SfM

Problem: recover motion and structure

Affine Structure from motion

- 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:
 $\mathbf{A} = \mathbf{U}_3 \mathbf{W}_3^{1/2}$ and $\mathbf{S} = \mathbf{W}_3^{1/2} \mathbf{V}_3^T$
- Eliminate affine ambiguity
 - Solve $\mathbf{L} = \mathbf{C}\mathbf{C}^T$ using metric constraints
 - Solve \mathbf{C} using Cholesky decomposition
 - Update \mathbf{A} and \mathbf{X} : $\mathbf{A} = \mathbf{A}\mathbf{C}$, $\mathbf{S} = \mathbf{C}^{-1}\mathbf{S}$



```
load tracks.mat
```

```
track_x - [500 x 51]
```

```
track_y - [500 x 51]
```

Use `plotSfM(A, S)` to display motion and shape

```
A - [2m x 3] motion matrix
```

```
S - [3 x n]
```

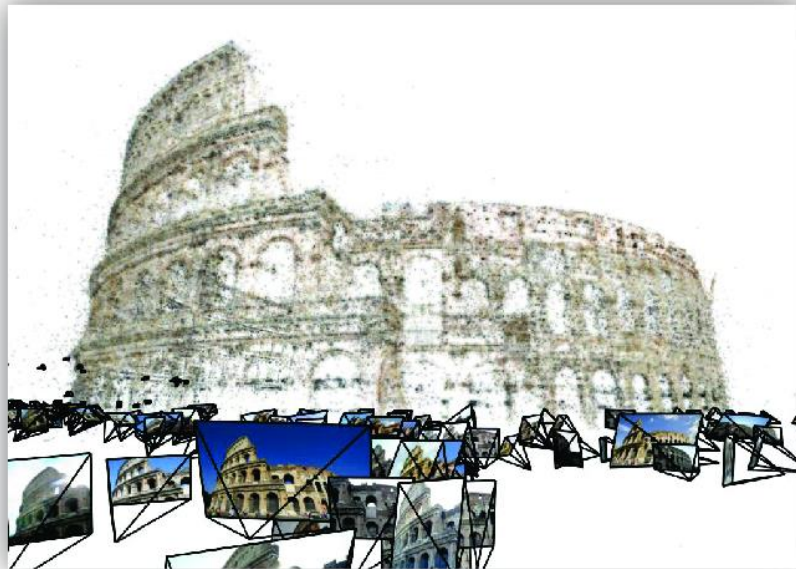
HW 3 – Part 3 Affine SfM

- Eliminate affine ambiguity

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

- Solve for $\mathbf{L} = \mathbf{C} \mathbf{C}^T$
 - `L = reshape(A\b, [3,3]); % A - 3m x 9, b - 3m x 1`
- Recover \mathbf{C} from \mathbf{L} by Cholesky decomposition: $\mathbf{L} = \mathbf{C} \mathbf{C}^T$
- Update \mathbf{A} and \mathbf{X} : $\mathbf{A} = \mathbf{A} \mathbf{C}$, $\mathbf{X} = \mathbf{C}^{-1} \mathbf{X}$

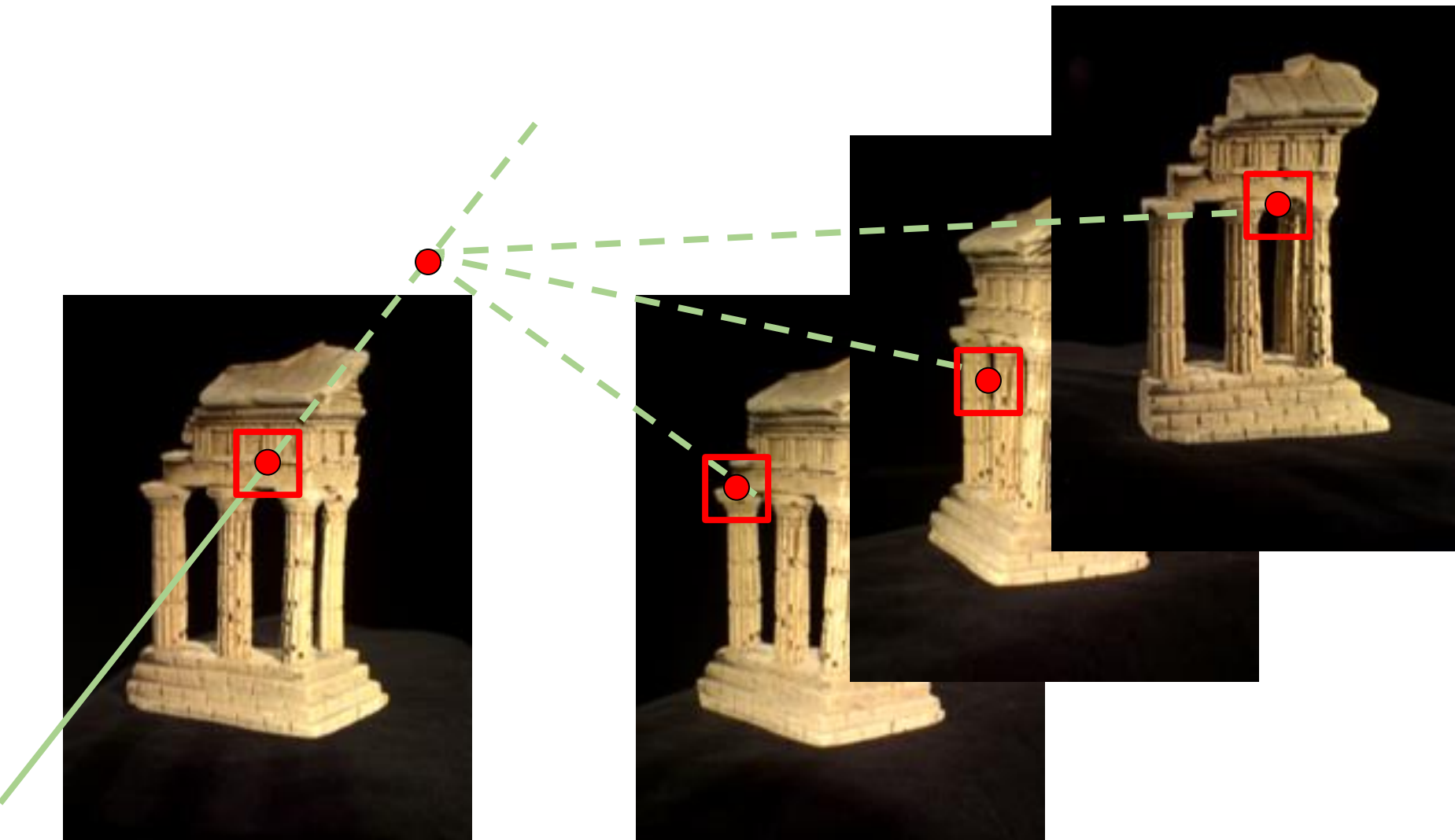
Multi-view stereo



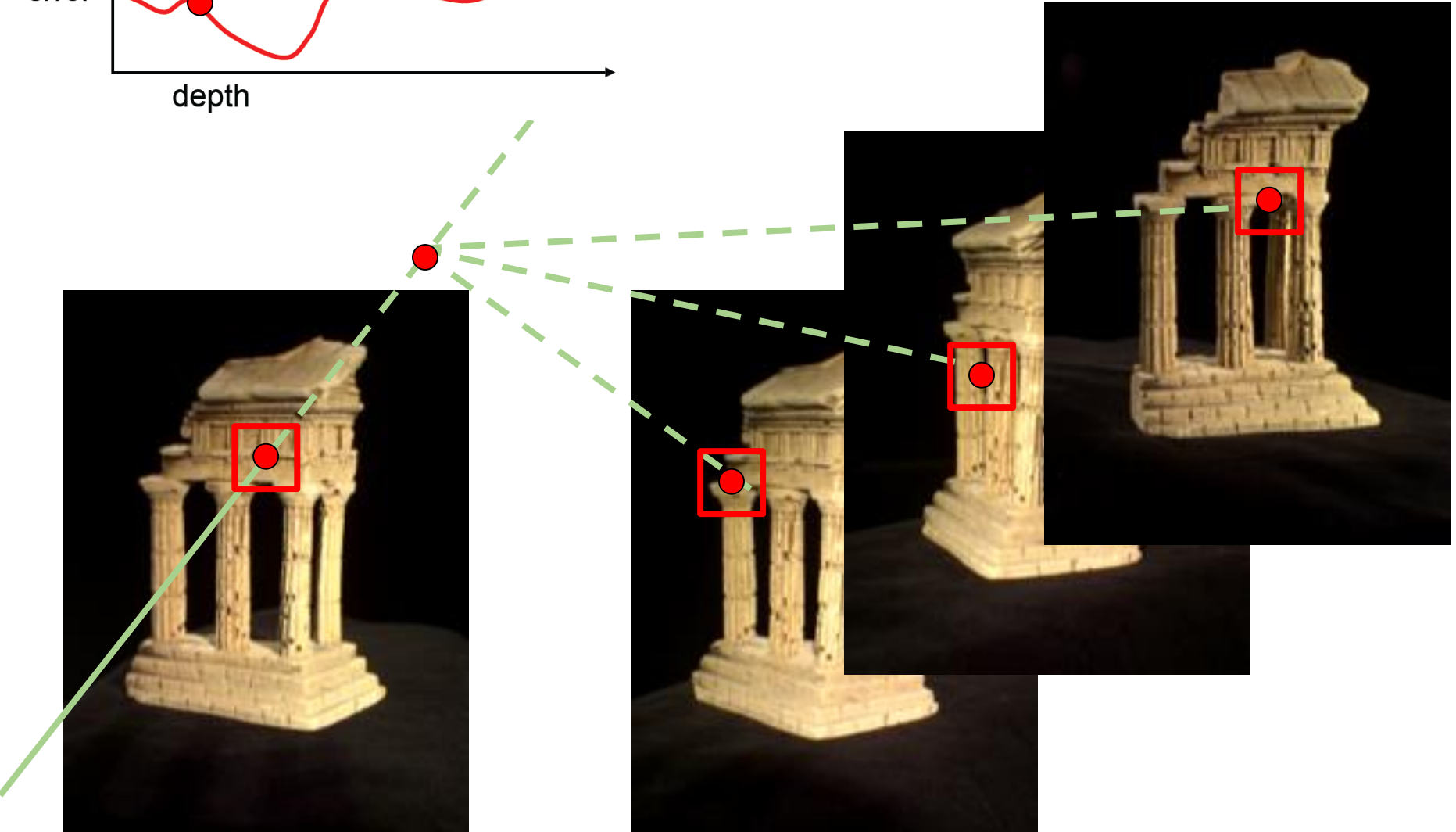
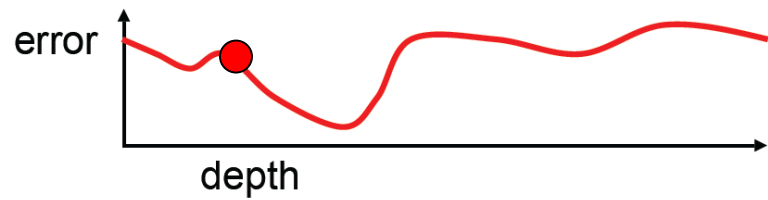
Multi-view stereo

- Generic problem formulation: given several images of the same object or scene, compute a representation of its 3D shape
- “Images of the same object or scene”
 - Arbitrary number of images (from two to thousands)
 - Arbitrary camera positions (special rig, camera network or video sequence)
 - Calibration may be known or unknown
- “Representation of 3D shape”
 - Depth maps
 - Meshes
 - Point clouds
 - Patch clouds
 - Volumetric models
 -

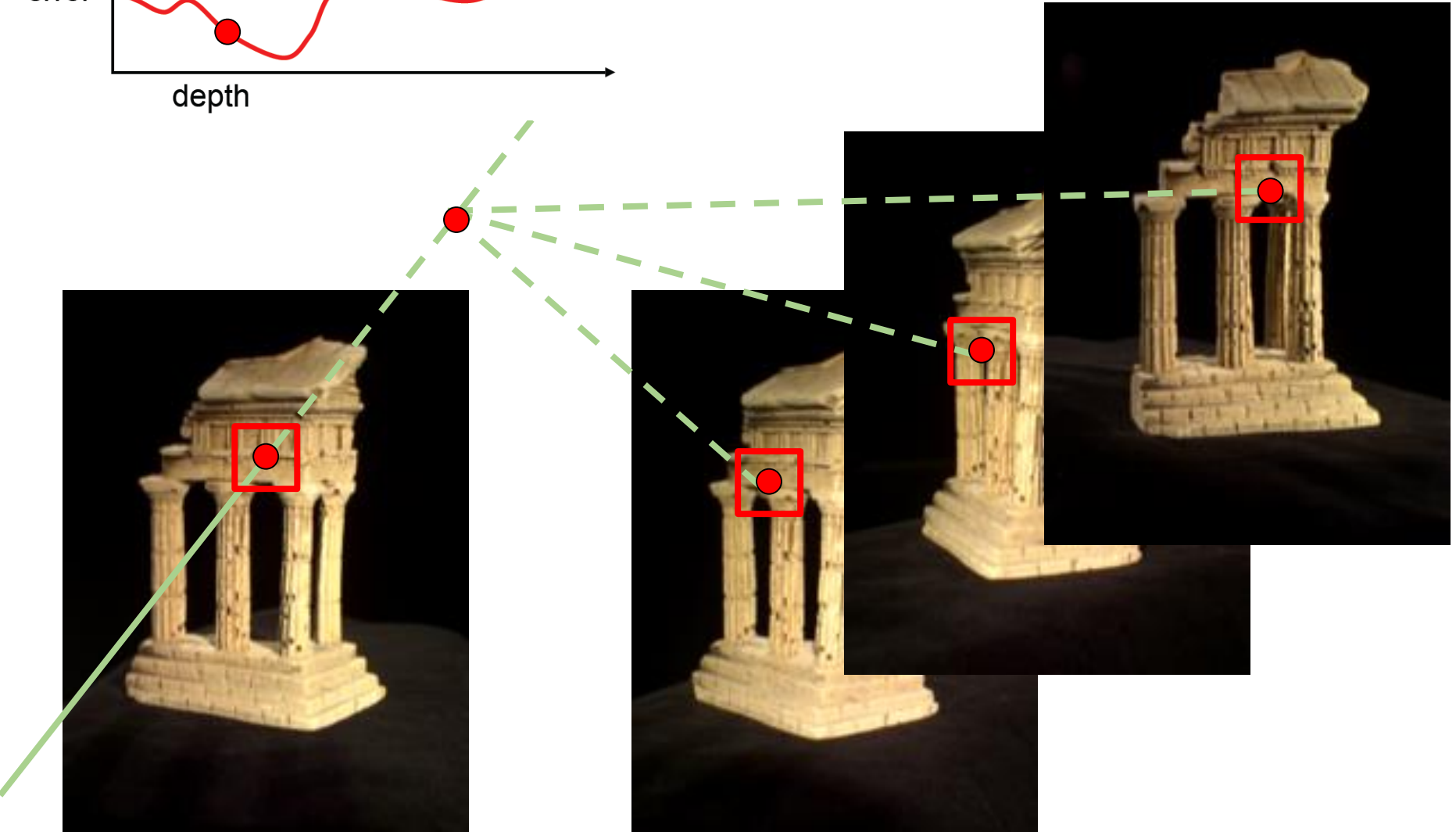
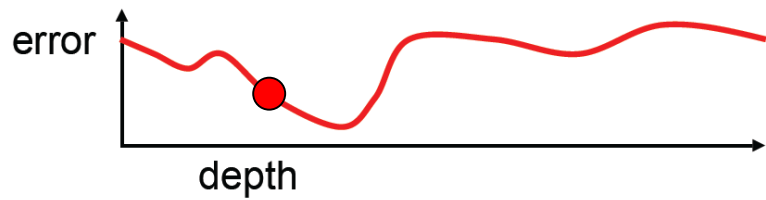
Multi-view stereo: Basic idea



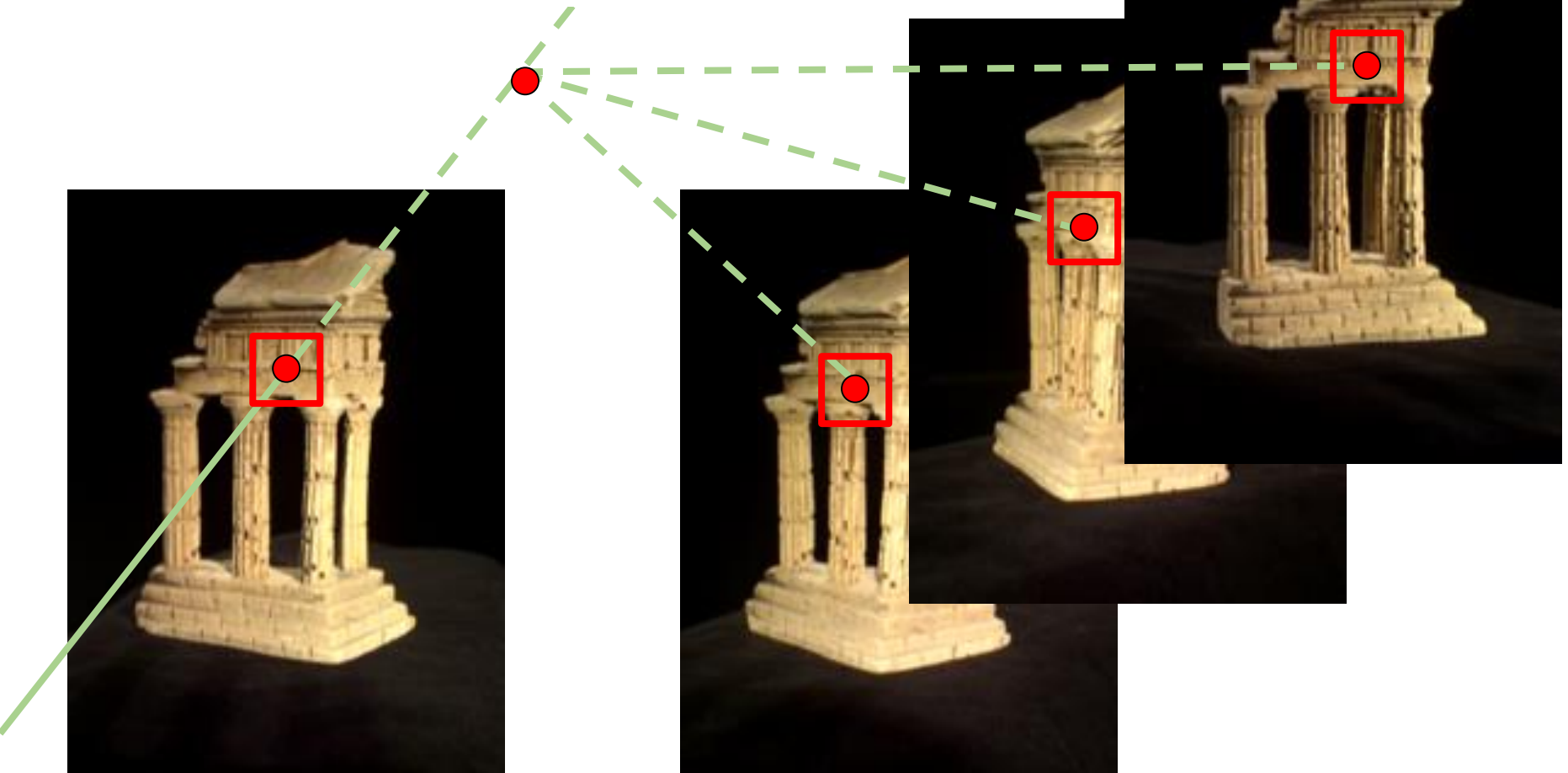
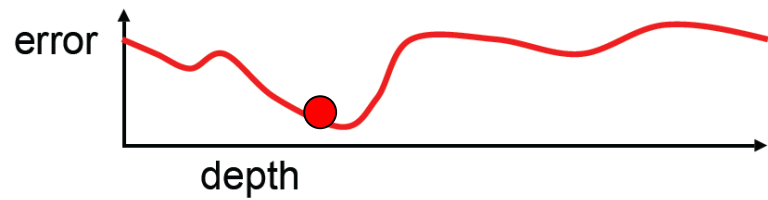
Multi-view stereo: Basic idea



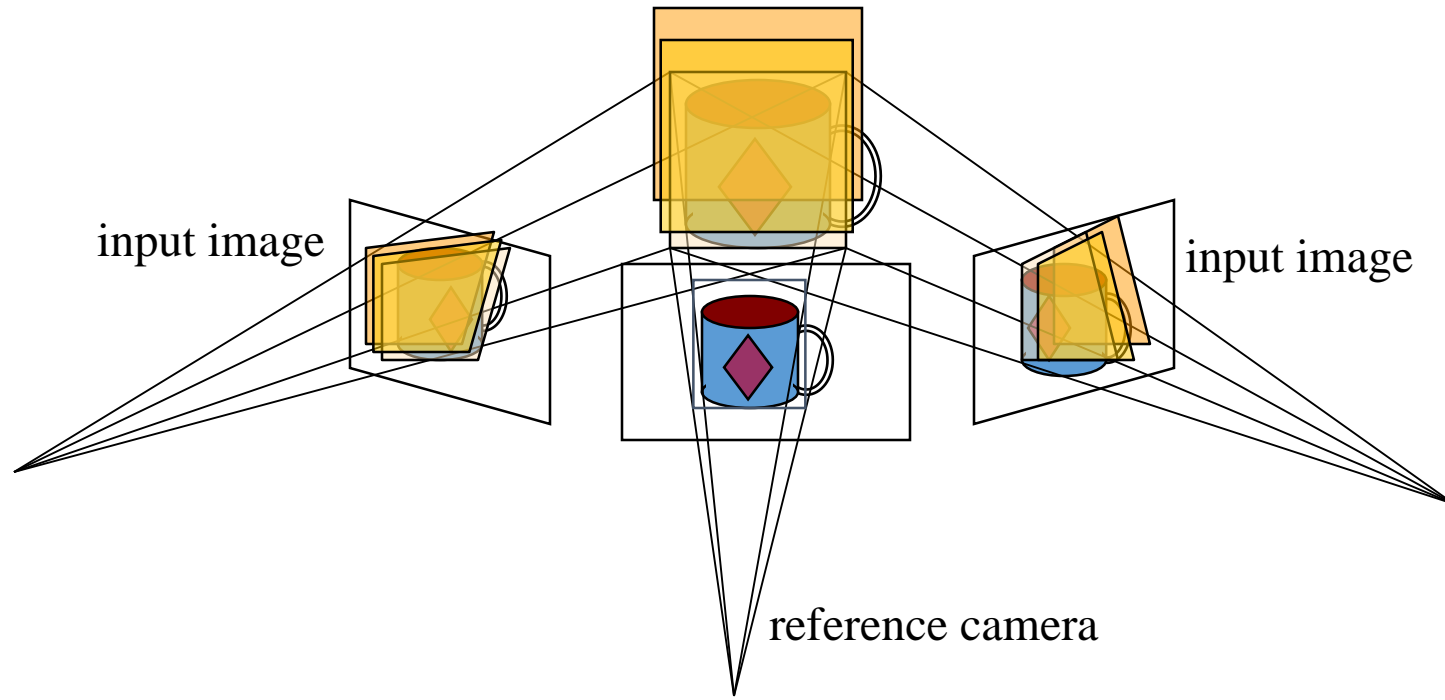
Multi-view stereo: Basic idea



Multi-view stereo: Basic idea

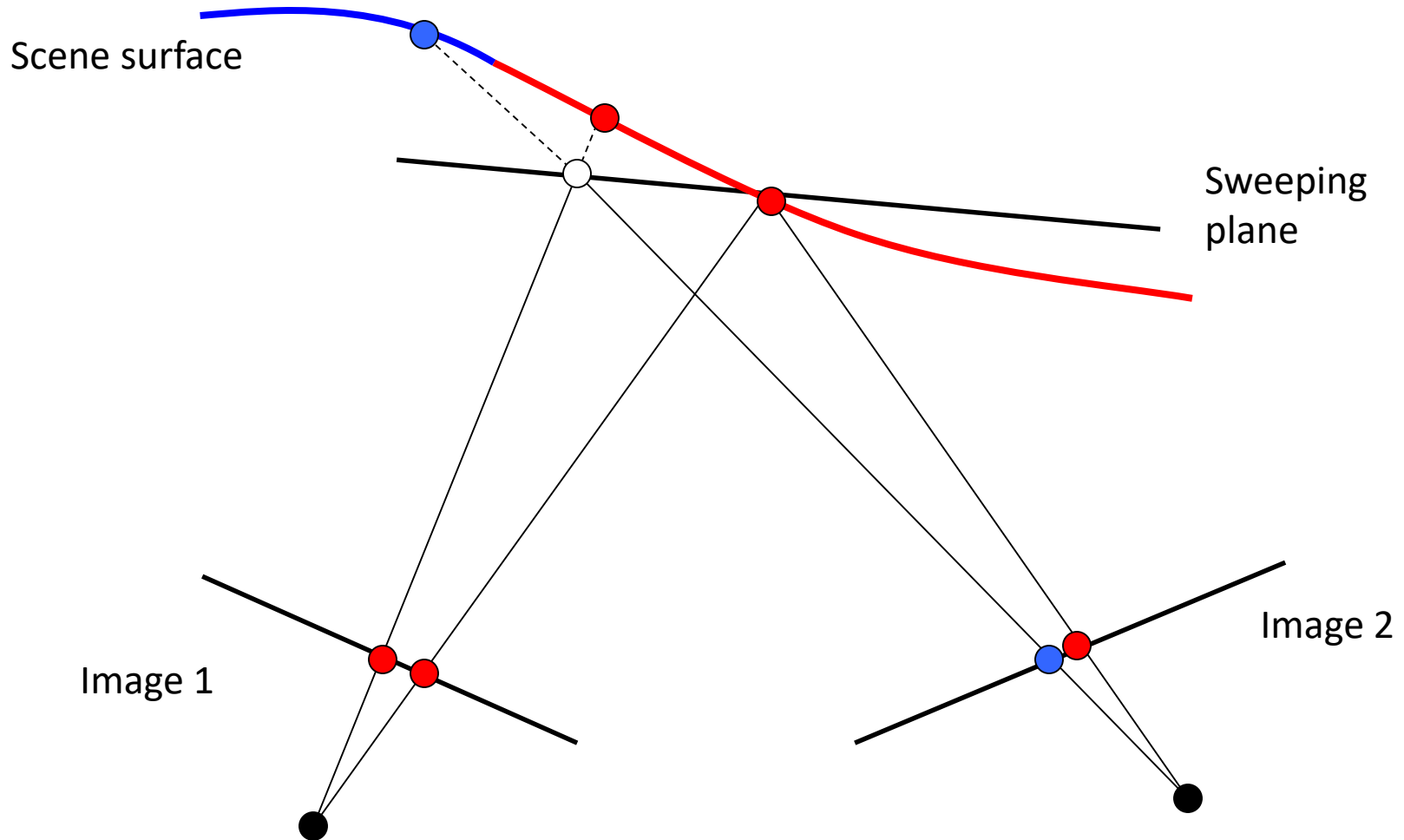


Plane Sweep Stereo



- Sweep family of planes at different depths w.r.t. a reference camera
- For each depth, project each input image onto that plane
- This is equivalent to a homography warping each input image into the reference view
- What can we say about the scene points that are at the right depth?

Plane Sweep Stereo



Plane Sweep Stereo



- For each depth plane
 - For each pixel in the composite image stack, compute the variance
- For each pixel, select the depth that gives the lowest variance
- Can be accelerated using graphics hardware

Merging depth maps

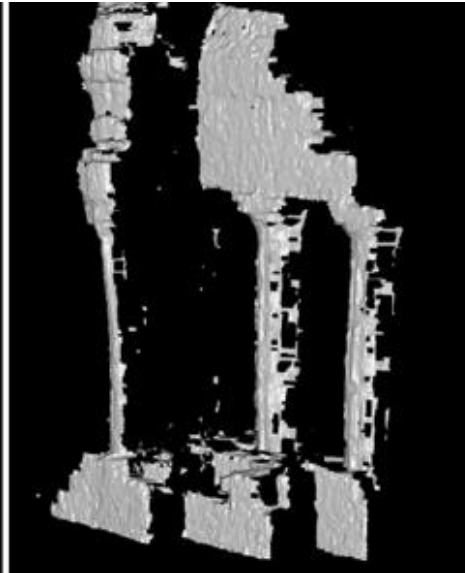


- Given a group of images, choose each one as reference and compute a depth map w.r.t. that view using a multi-baseline approach
- Merge multiple depth maps to a volume or a mesh (see, e.g., Curless and Levoy 96)

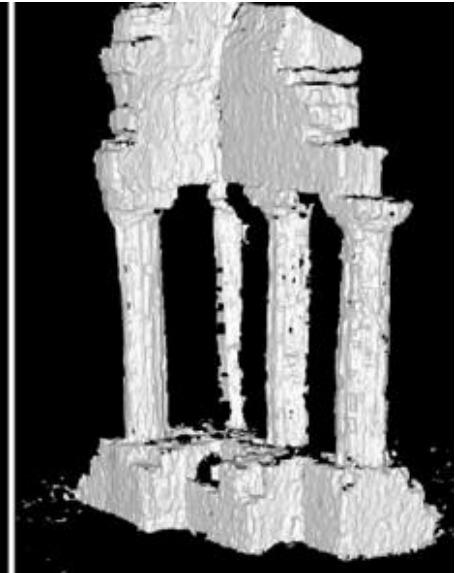
Map 1



Map 2



Merged

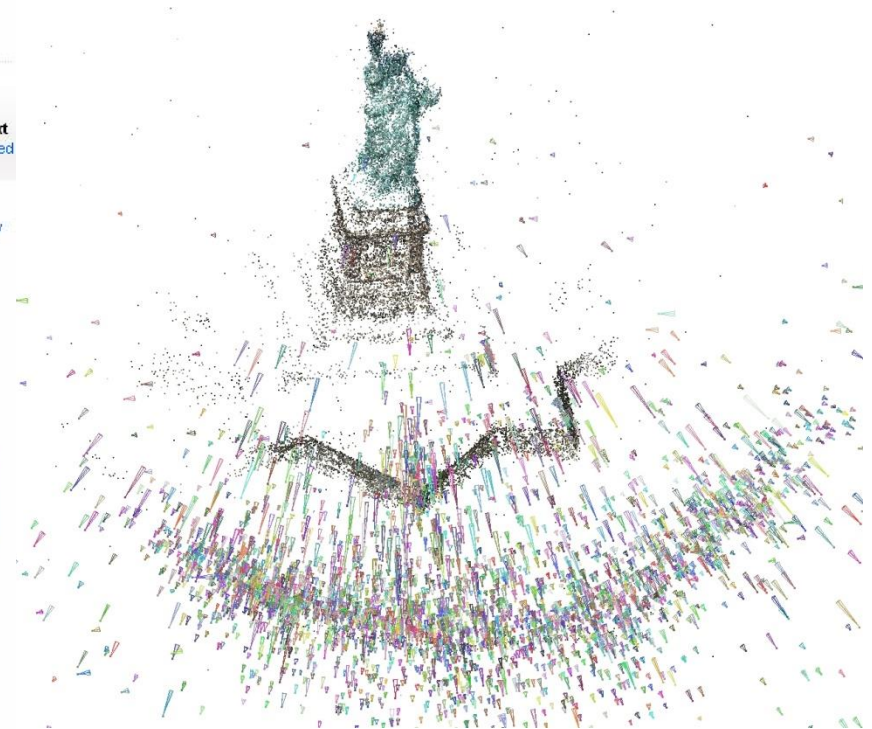


Stereo from community photo collections

- Need *structure from motion* to recover unknown camera parameters
- Need *view selection* to find good groups of images on which to run dense stereo

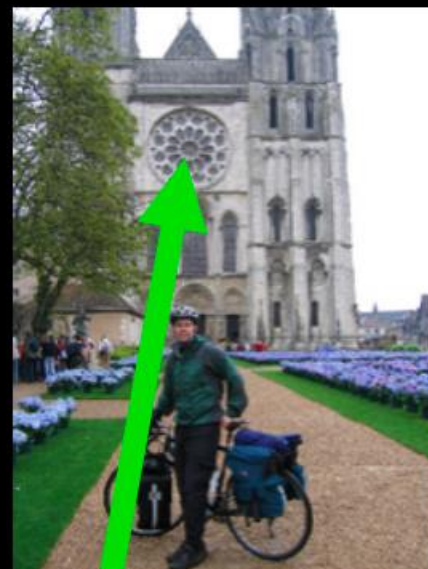


Sort: **Relevant** | Recent | Interesting View: **Small** | Medium | Detail | Slideshow





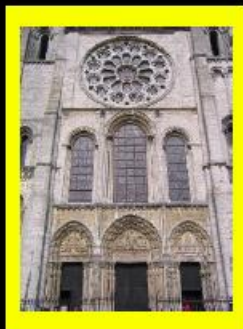
4 best neighboring views



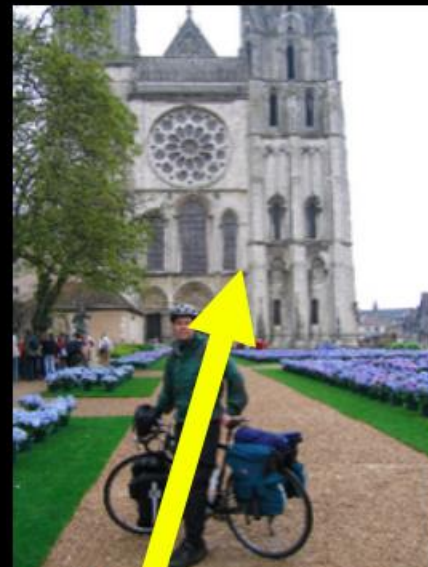
reference view

Local view selection

- Automatically select neighboring views for each **point** in the image
- Desiderata: good matches AND good baselines



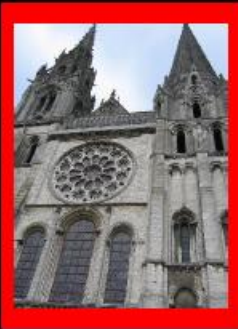
4 best neighboring views



reference view

Local view selection

- Automatically select neighboring views for each **point** in the image
- Desiderata: good matches AND good baselines



4 best neighboring views



reference view

Local view selection

- Automatically select neighboring views for each **point** in the image
- Desiderata: good matches AND good baselines

Towards Internet-Scale Multi-View Stereo



St. Peter's Basilica



Trevi Fountain



Colosseum



Dubrovnik



Piazza San Marco

- [YouTube video](#), [high-quality video](#)

Yasutaka Furukawa, Brian Curless, Steven M. Seitz and Richard Szeliski, [Towards Internet-scale Multi-view Stereo](#), CVPR 2010.

Internet-Scale Multi-View Stereo



The Visual Turing Test for Scene Reconstruction

Rendered Images (Right) vs. Ground Truth Images (Left)



Q. Shan, R. Adams, B. Curless, Y. Furukawa, and S. Seitz, ["The Visual Turing Test for Scene Reconstruction,"](#) 3DV 2013.

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
- <https://www.youtube.com/watch?v=kylzMr917Rc>, 3D Computer Vision: Past, Present, and Future

Next class

- Grouping and Segmentation

