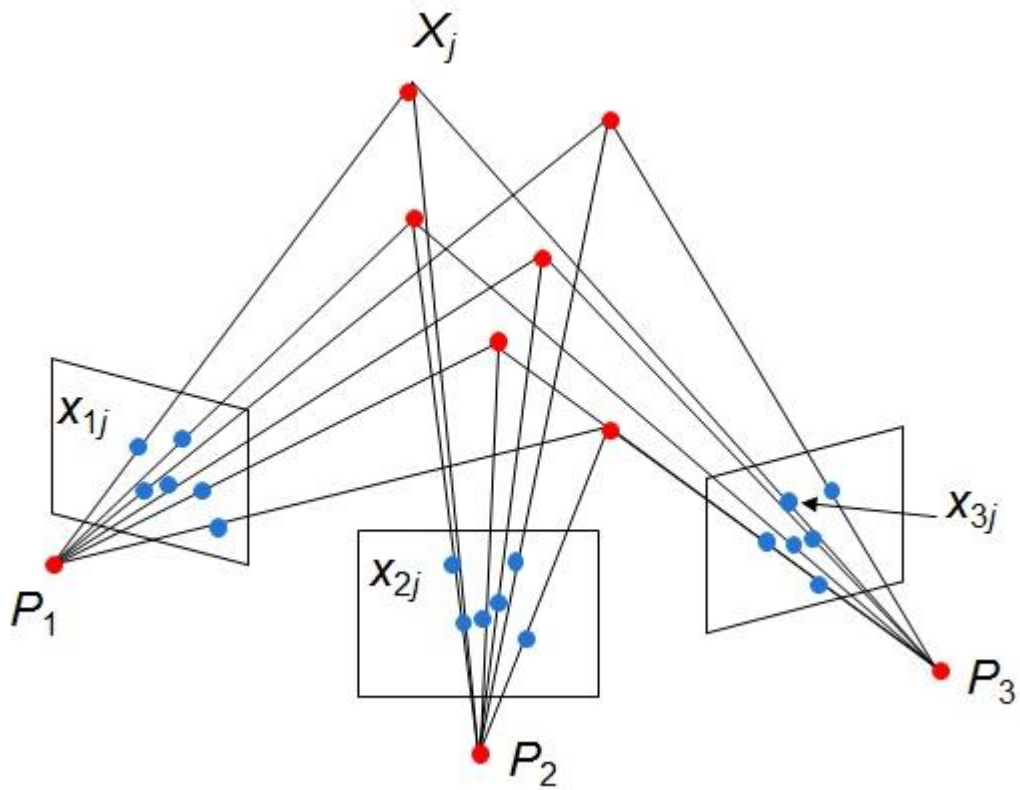
Structure from Motion



Jia-Bin Huang, Virginia Tech

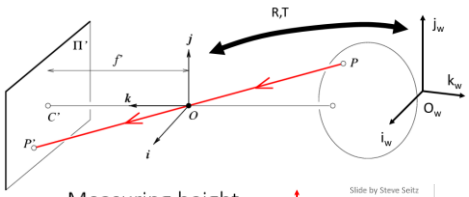
Many slides from S. Seitz, N Snavely, and D. Hoiem

Administrative stuffs

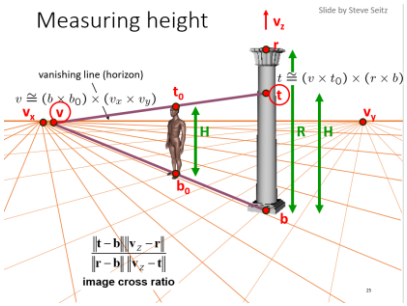
- HW 3 due 11:59 PM, Oct 25 (Wed)
- Submit your alignment results! [Link]
- HW 2 grades are out
  - Average: 113.7, median: 126
  - Excellent reports: Diya Li, Vikram Mohanty, Yihan Pang, Chang Sun, Yecheng Zhao

Perspective and 3D Geometry

- **Projective geometry and camera models**
  - Vanishing points/lines
  - $\mathrm{x} = \mathbf{K}[\mathbf{R}\ \ \mathbf{t}]\mathbf{X}$

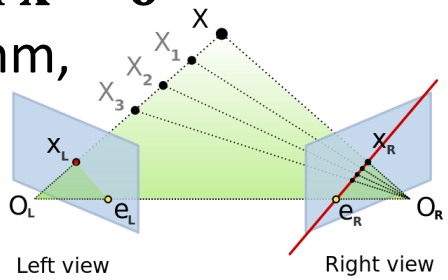


- **Single-view metrology and camera calibration**
  - Calibration using known 3D object or vanishing points
  - Measuring size using perspective cues



- **Photo stitching**
  - Homography relates rotating cameras $\mathbf{x}' = \mathbf{H}\mathbf{x}$
  - Recover homography using RANSAC + normalized DLT

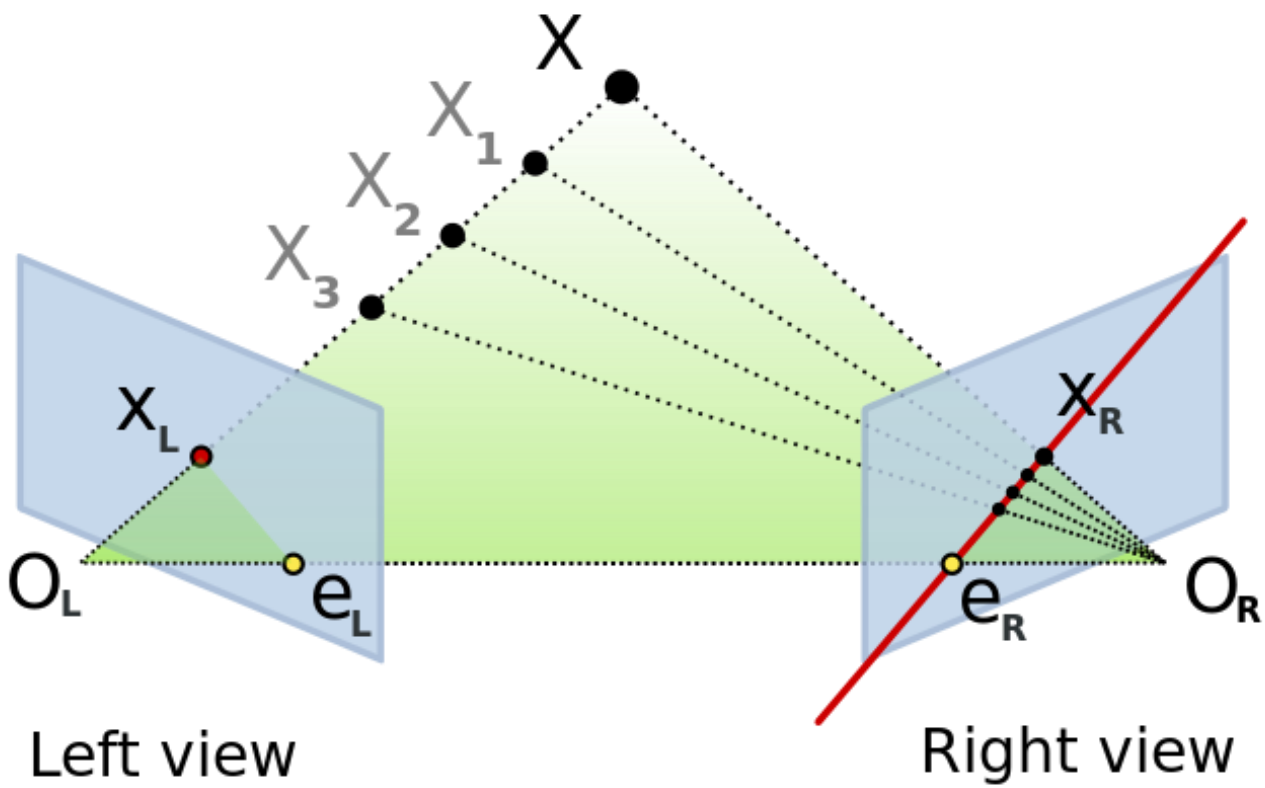
- **Epipolar Geometry and Stereo Vision**
  - Fundamental/essential matrix relates two cameras $\mathbf{x}'\mathbf{F}\mathbf{x} = \mathbf{0}$
  - Recover $\mathbf{F}$ using RANSAC + normalized 8-point algorithm, enforce rank 2 using SVD



- **Structure from motion** (this class)
  - How can we recover 3D points from multiple images?

Recap: Epipoles

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



Recap: Fundamental Matrix

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

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

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

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

Recap: Automatic Estimation of F

Assume we have matched points x ⇔ 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: $\mathbf{F} = \mathbf{T}'^T \tilde{\mathbf{F}} \mathbf{T}$

This class: Structure from Motion

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

Structure [ˈstrək(t)SHər]:

3D Point Cloud of the Scene

Motion [ˈmōSH(ə)n]:

Camera Location and Orientation

Structure from Motion (SfM)

Get the Point Cloud from Moving Cameras

SfM Applications – 3D Modeling

http://www.3dcadbrowser.com/download.aspx?3dmodel=40454

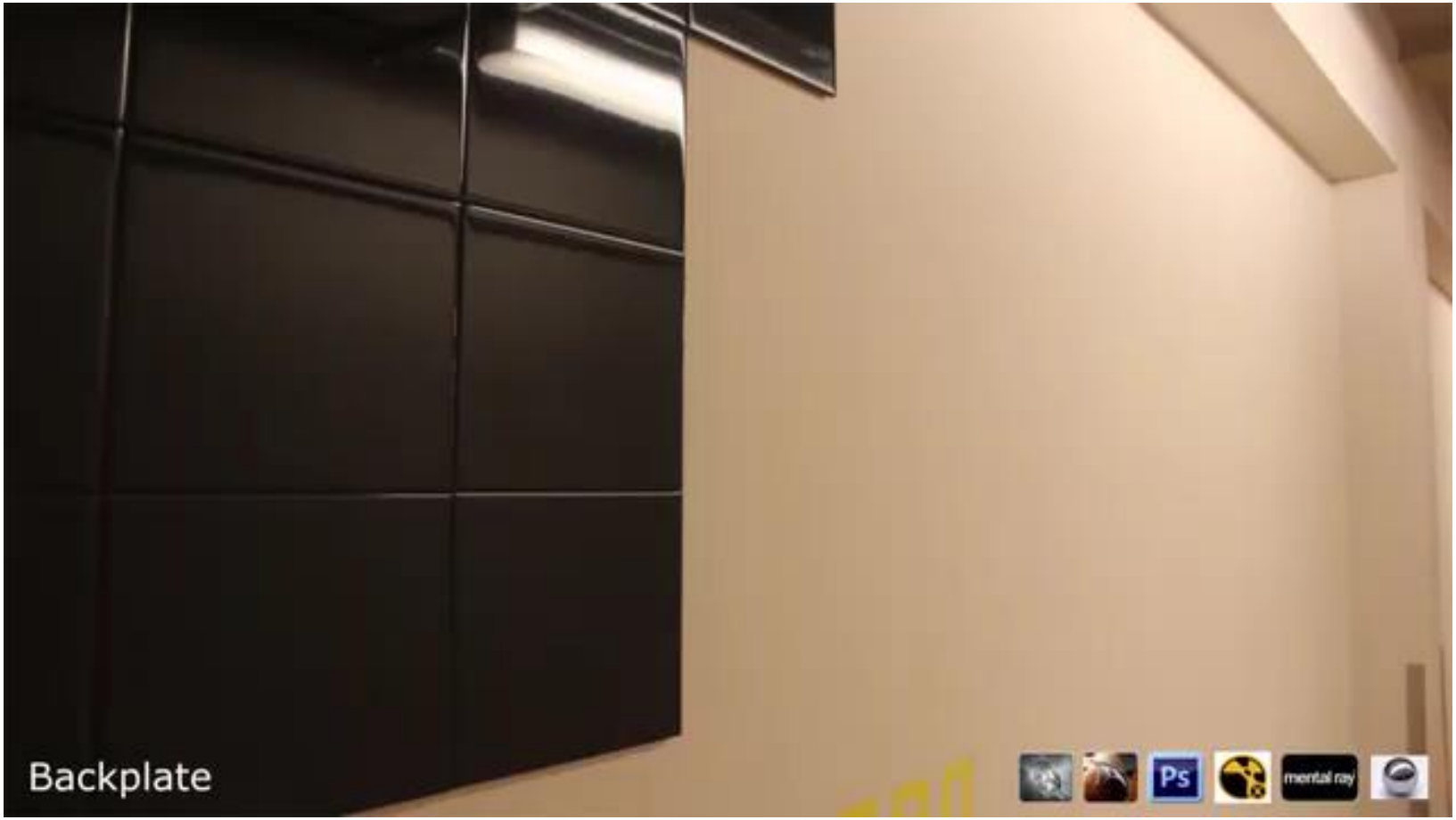
SfM Applications – Surveying cultural heritage structure analysis

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

SfM Applications – Robot navigation and mapmaking

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

SfM Applications – Visual effect (matchmove)



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

Steps

	Images → Points:	Structure from Motion
	Points → More points:	Multiple View Stereo
+	Points → Meshes:	Model Fitting
	Meshes → Models:	Texture Mapping
=	Images → Models:	Image-based Modeling

Slide credit: J. Xiao

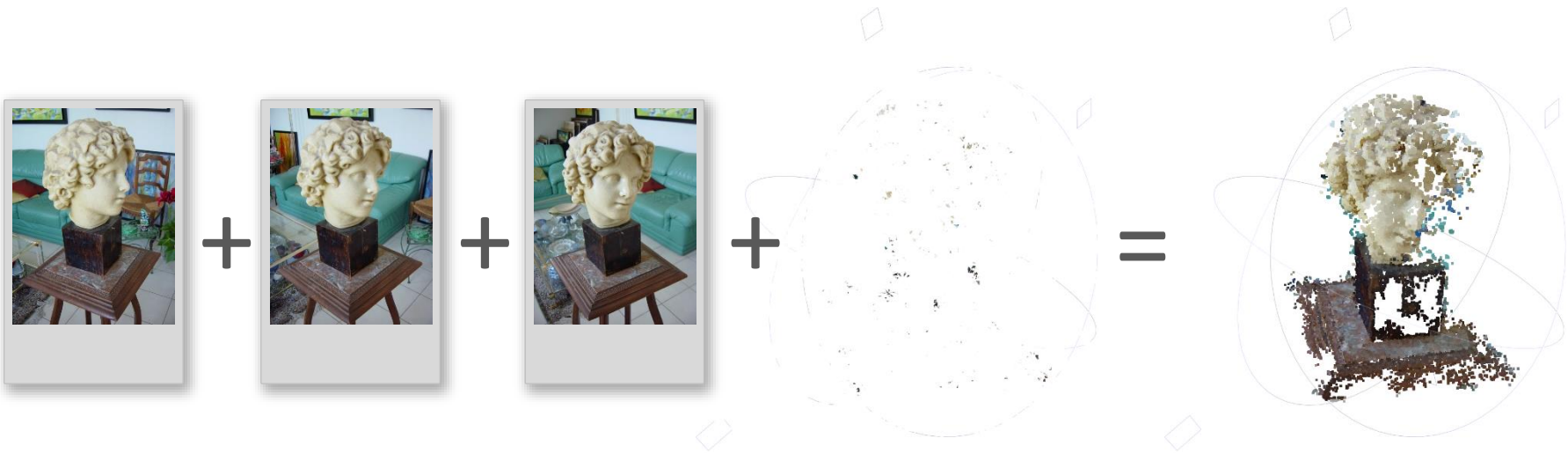
Steps

	Images → Points:	Structure from Motion
	Points → More points:	Multiple View Stereo
+	Points → Meshes:	Model Fitting
	Meshes → Models:	Texture Mapping
=	Images → Models:	Image-based Modeling

Slide credit: J. Xiao

Steps

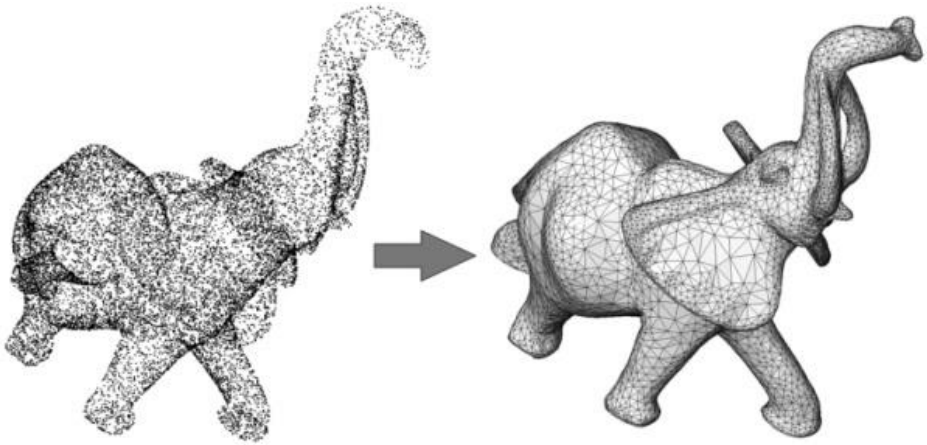
	Images → Points:	Structure from Motion
	Points → More points:	Multiple View Stereo
+	Points → Meshes:	Model Fitting
	Meshes → Models:	Texture Mapping
=	Images → Models:	Image-based Modeling



Slide credit: J. Xiao

Steps

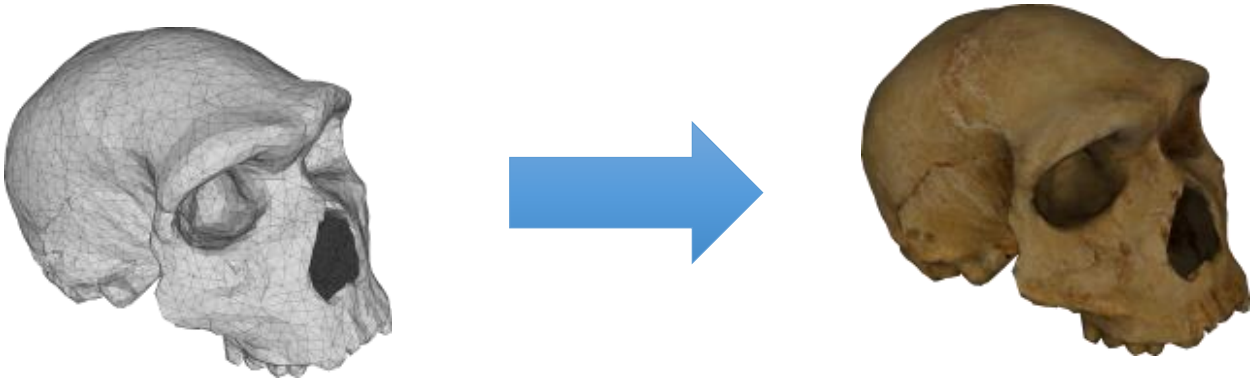
	Images → Points:	Structure from Motion
	Points → More points:	Multiple View Stereo
+	Points → Meshes:	Model Fitting
	Meshes → Models:	Texture Mapping
=	Images → Models:	Image-based Modeling



Slide credit: J. Xiao

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



Slide credit: J. Xiao

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

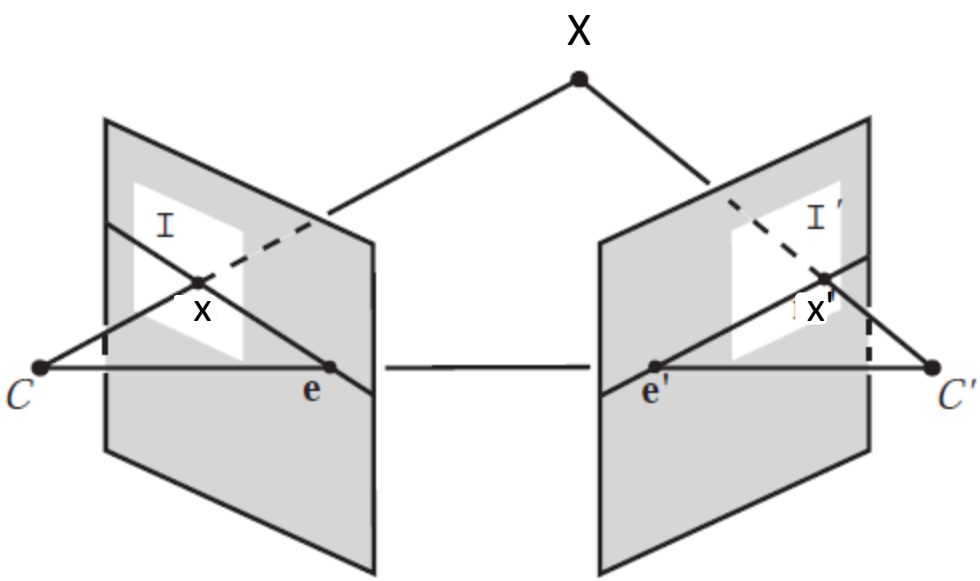
Example: https://photosynth.net/

Slide credit: J. Xiao

Triangulation: Linear Solution

- Generally, rays C→x and C'→x' will not exactly intersect

- Solve via SVD:
  A least squares solution a system of equations



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

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

Further reading: HZ p. 312-313

Triangulation: Linear Solution

$$\mathbf{x} = w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \mathbf{P} = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix}$$

$$\mathbf{x}' = w\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} \qquad \mathbf{P}' = \begin{bmatrix} \mathbf{p}_1'^T \\ \mathbf{p}_2'^T \\ \mathbf{p}_3'^T \end{bmatrix}$$

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

$$\mathbf{x} = w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{P}\boldsymbol{X} = \begin{bmatrix} \boldsymbol{p}_1^T \\ \boldsymbol{p}_2^T \\ \boldsymbol{p}_3^T \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} \boldsymbol{p}_1^T\boldsymbol{X} \\ \boldsymbol{p}_2^T\boldsymbol{X} \\ \boldsymbol{p}_3^T\boldsymbol{X} \end{bmatrix}$$

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u\boldsymbol{p}_3^T\boldsymbol{X} \\ v\boldsymbol{p}_3^T\boldsymbol{X} \\ \boldsymbol{p}_3^T\boldsymbol{X} \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_1^T\boldsymbol{X} \\ \boldsymbol{p}_2^T\boldsymbol{X} \\ \boldsymbol{p}_3^T\boldsymbol{X} \end{bmatrix}$$

$$\begin{aligned}
u\boldsymbol{p}_3^T\boldsymbol{X} - \boldsymbol{p}_1^T\boldsymbol{X} &= \left[u\boldsymbol{p}_3^T - \boldsymbol{p}_1^T\right]\boldsymbol{X} = \mathbf{0} \\
v\boldsymbol{p}_3^T\boldsymbol{X} - \boldsymbol{p}_2^T\boldsymbol{X} &= \left[v\boldsymbol{p}_3^T - \boldsymbol{p}_2^T\right]\boldsymbol{X} = \mathbf{0} \\
u'\boldsymbol{p}'^T_3\boldsymbol{X} - \boldsymbol{p}'^T_1\boldsymbol{X} &= \left[u'\boldsymbol{p}'^T_3 - \boldsymbol{p}'^T_1\right]\boldsymbol{X} = \mathbf{0} \\
v'\boldsymbol{p}'^T_3\boldsymbol{X} - \boldsymbol{p}'^T_2\boldsymbol{X} &= \left[v'\boldsymbol{p}'^T_3 - \boldsymbol{p}'^T_2\right]\boldsymbol{X} = \mathbf{0}
\end{aligned}$$

Triangulation: Linear Solution

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

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

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

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

Pros and Cons

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

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

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

Triangulation: Non-linear Solution

- Minimize projected error while satisfying

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

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

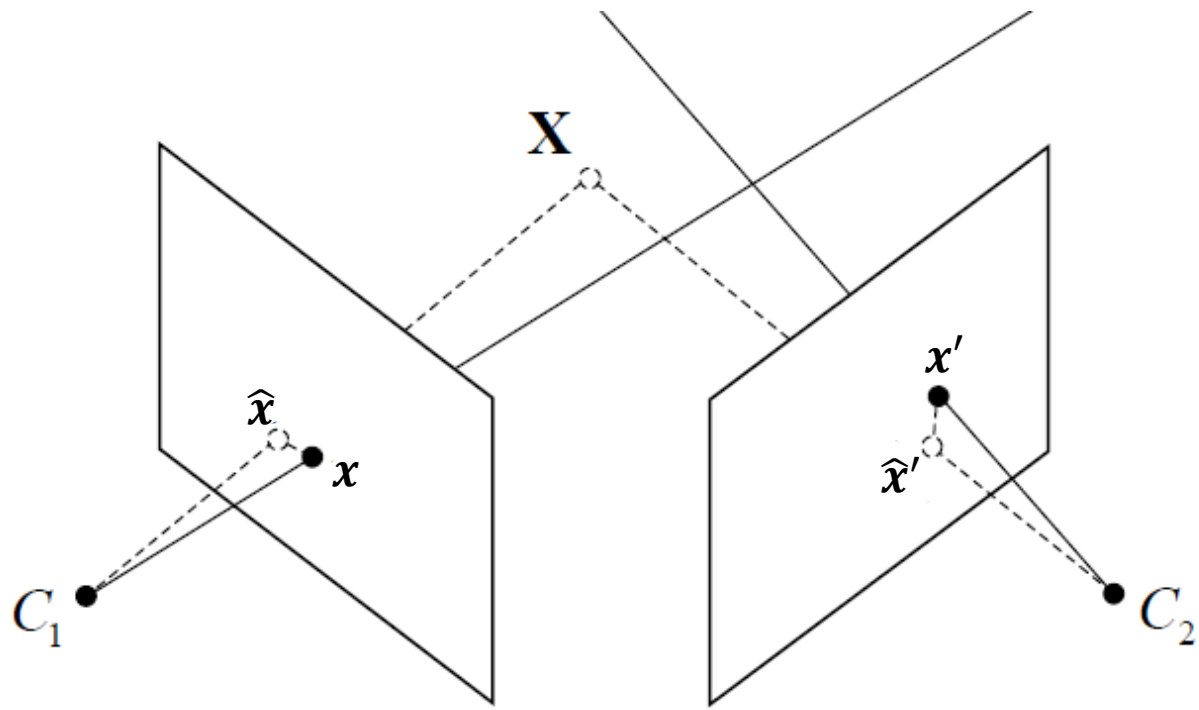


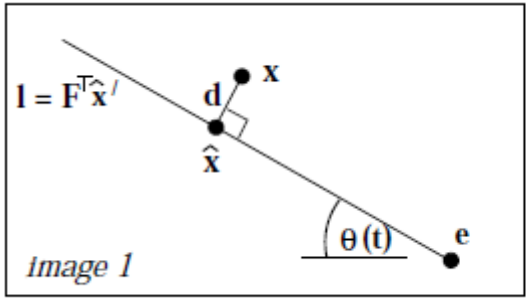
Figure source: Robertson and Cipolla (Chpt 13 of Practical Image Processing and Computer Vision)

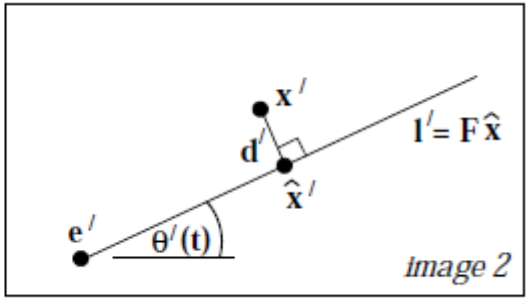
Triangulation: Non-linear Solution

- Minimize projected error while satisfying

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

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





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

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

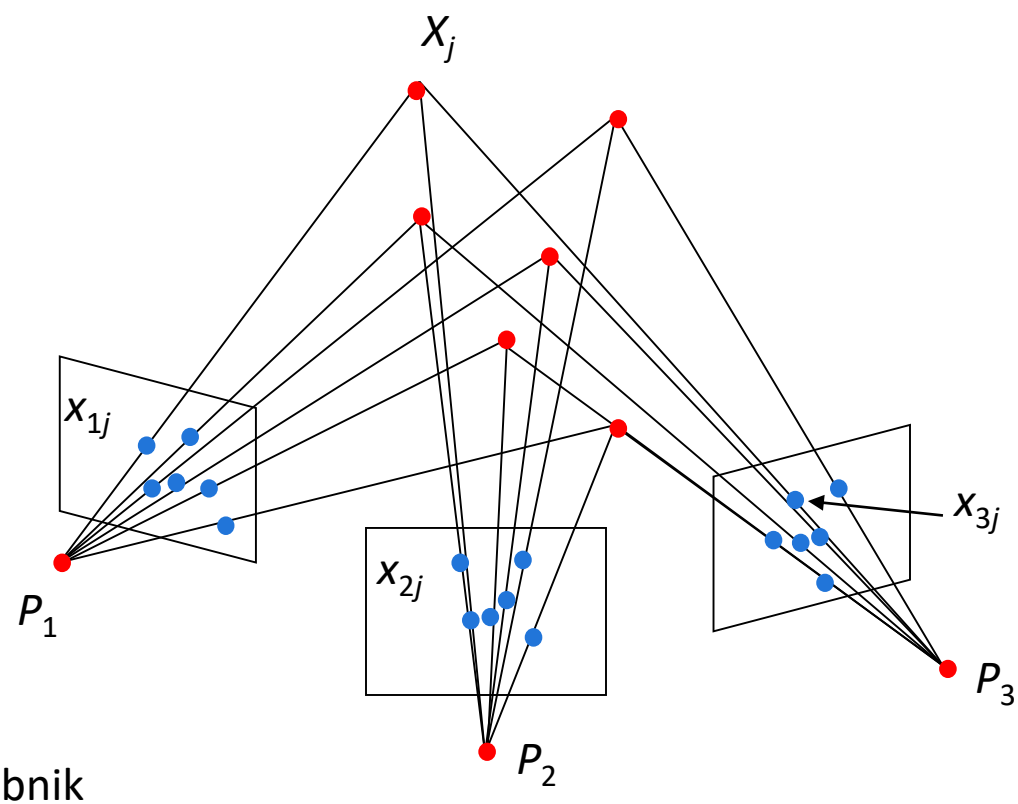
Further reading: HZ p. 318

Projective structure from motion

- Given: *m* images of *n* fixed 3D points

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

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



Slides from Lana Lazebnik

Projective structure from motion

- Given: *m* images of *n* fixed 3D points
  - $\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$
- 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 **Q**:
  - $\mathbf{X} \rightarrow \mathbf{QX}, \mathbf{P} \rightarrow \mathbf{PQ}^{-1}$
- We can solve for structure and motion when

$$2mn >= \boxed{11m} + \boxed{3n} - \boxed{15}$$

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

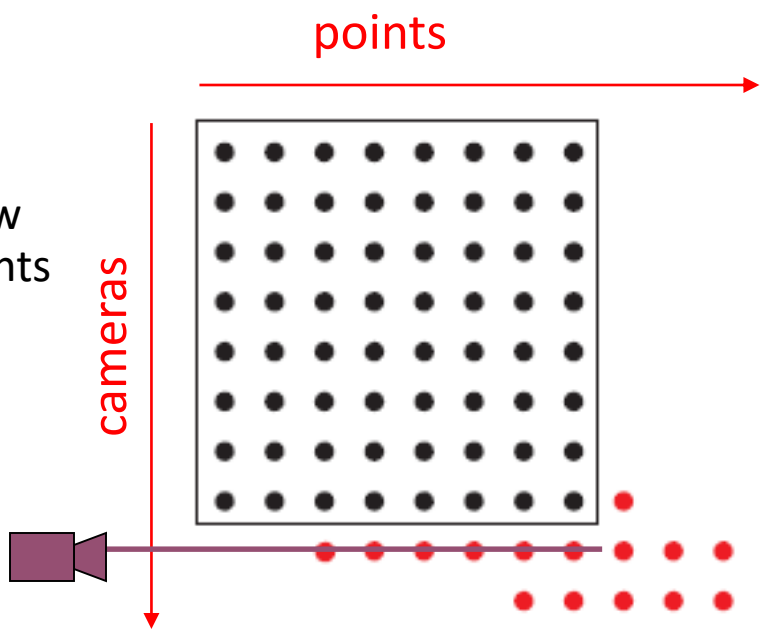
- For two cameras, at least 7 points are needed

Sequential structure from motion

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

- Initialize structure by triangulation

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



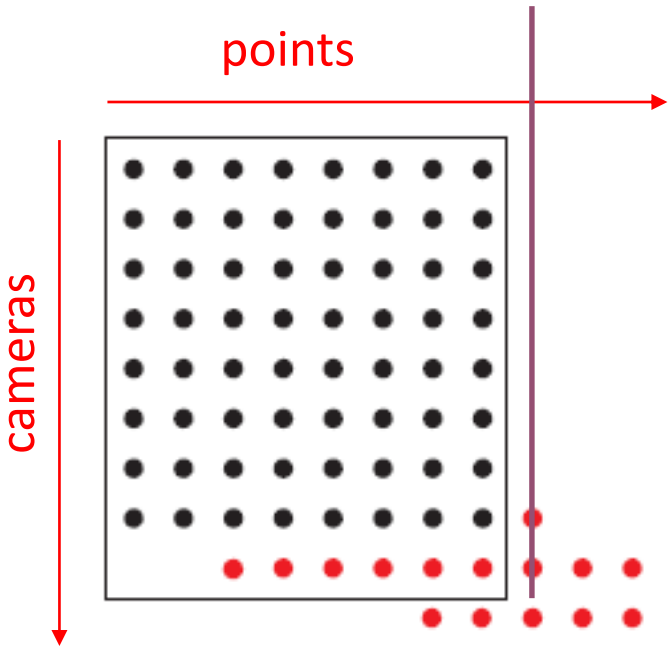
Sequential structure from motion

- Initialize motion from two images using fundamental matrix

- Initialize structure by triangulation

- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*

  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*



Sequential structure from motion

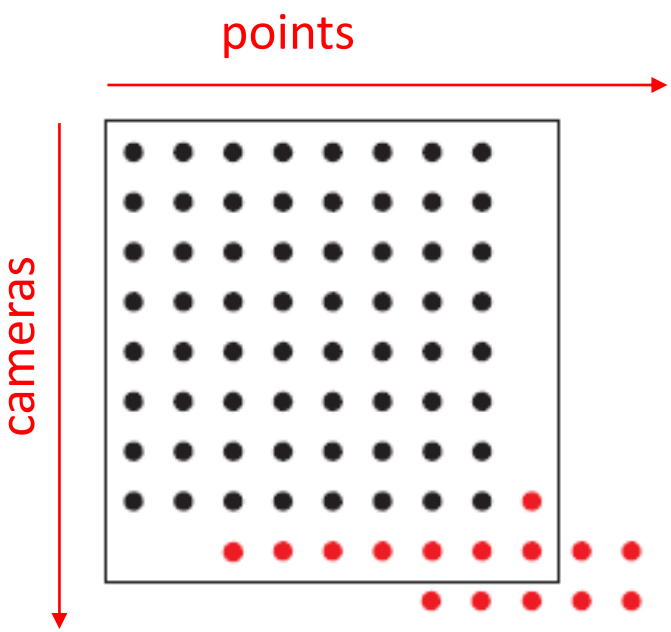
- Initialize motion from two images using fundamental matrix

- Initialize structure by triangulation

- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*

  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*

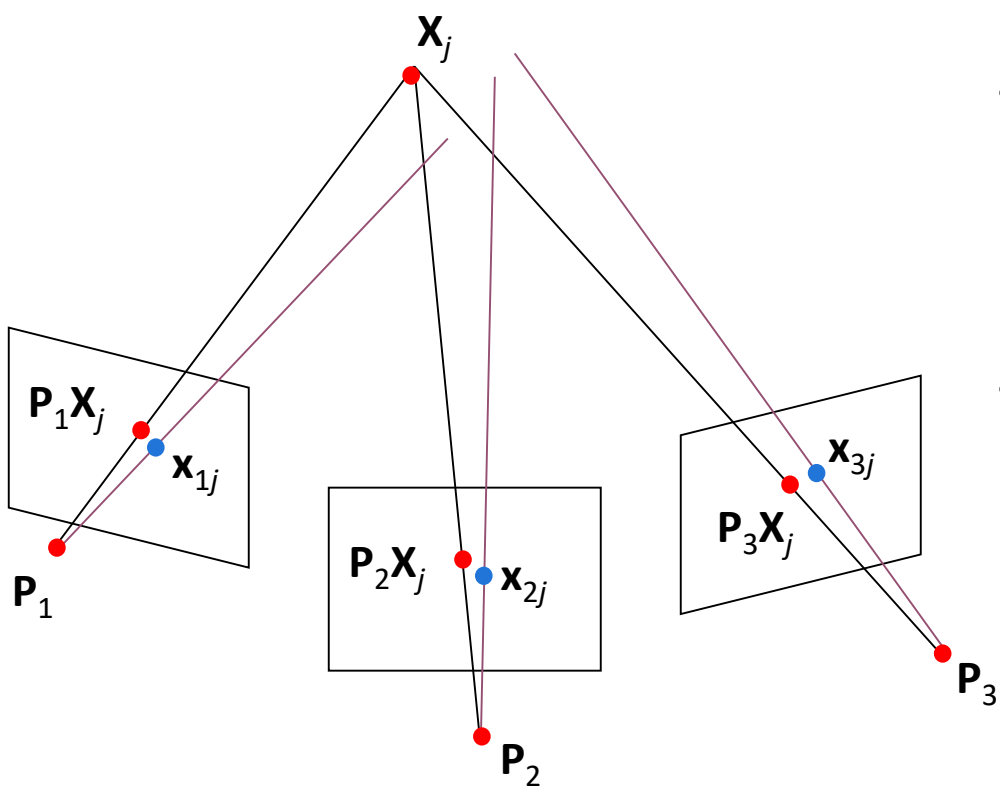
- Refine structure and motion: bundle adjustment



Bundle adjustment

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

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



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

Auto-calibration

- Auto-calibration: determining intrinsic camera parameters directly from uncalibrated images
- For example, we can use the constraint that a moving camera has a fixed intrinsic matrix
  - Compute initial projective reconstruction and find 3D projective transformation matrix **Q** such that all camera matrices are in the form $\mathbf{P}_i = \mathbf{K}\,[\mathbf{R}_i \mid \mathbf{t}_i]$
- Can use constraints on the form of the calibration matrix, such as zero skew

Summary so far

- 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)

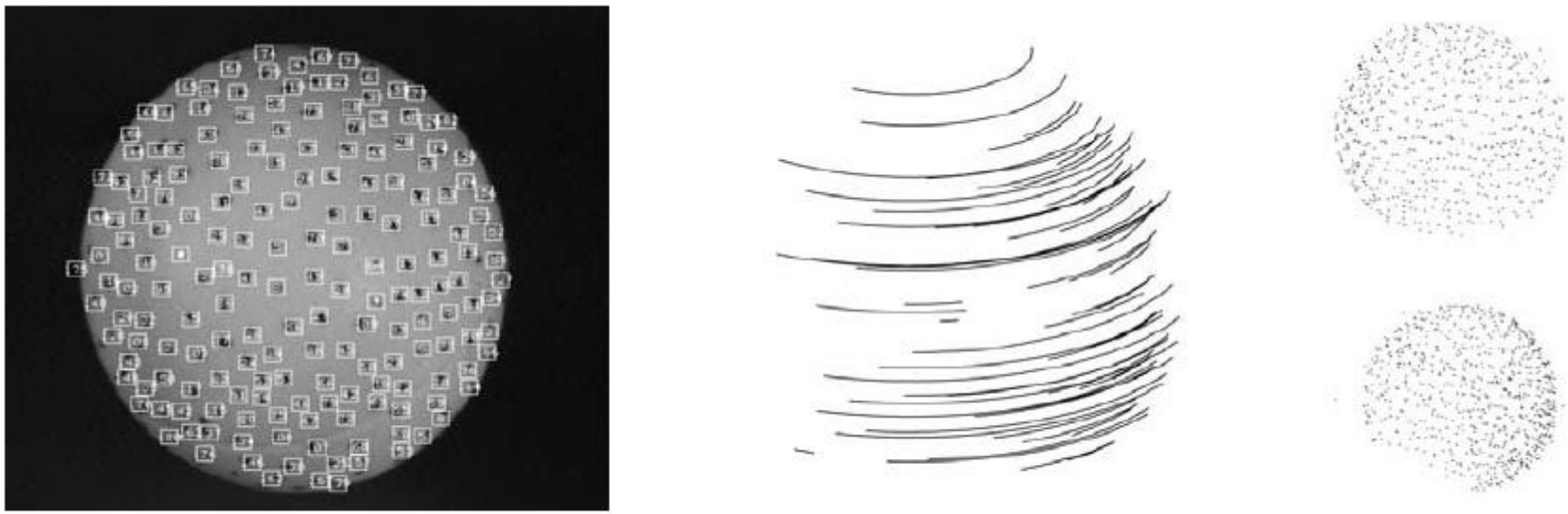
(best method with software available; also has good overview of recent methods)

Reconstruction of Cornell (Crandall et al. ECCV 2011)

3D from multiple images

Building Rome in a Day: Agarwal et al. 2009

Structure from motion under orthographic projection



(a) (b) (c)

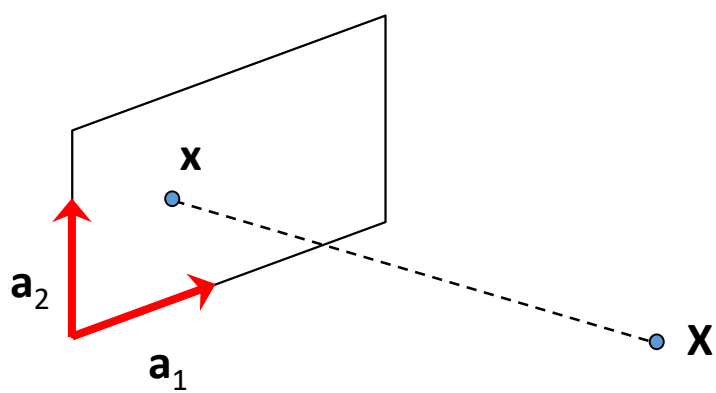
3D Reconstruction of a Rotating Ping-Pong Ball

- 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 - Examples

Orthographic projection for rotated/translated camera

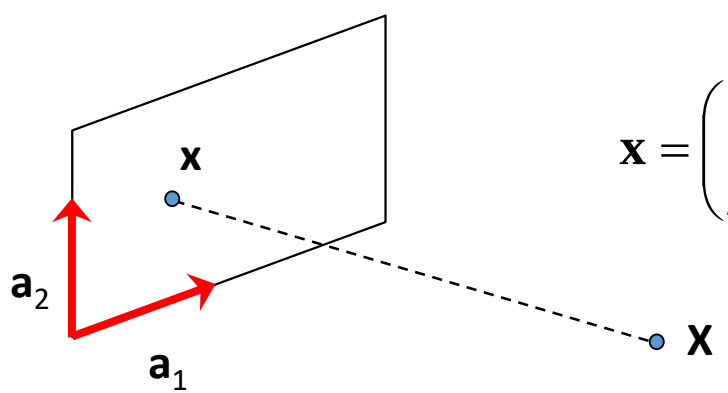


$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \begin{pmatrix} u_{fp} \\ v_{fp} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(R'_f \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix}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

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

Affine structure from motion

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



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

Projection of world origin

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

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

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

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

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

2d normalized point (observed)

3d normalized point

Linear (affine) mapping

Suppose we know 3D points and affine camera parameters …

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

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

3D Points (3xn)

Camera Parameters (2mx3)

What if we instead observe corresponding 2d image points?

Can we recover the camera parameters and 3d points?

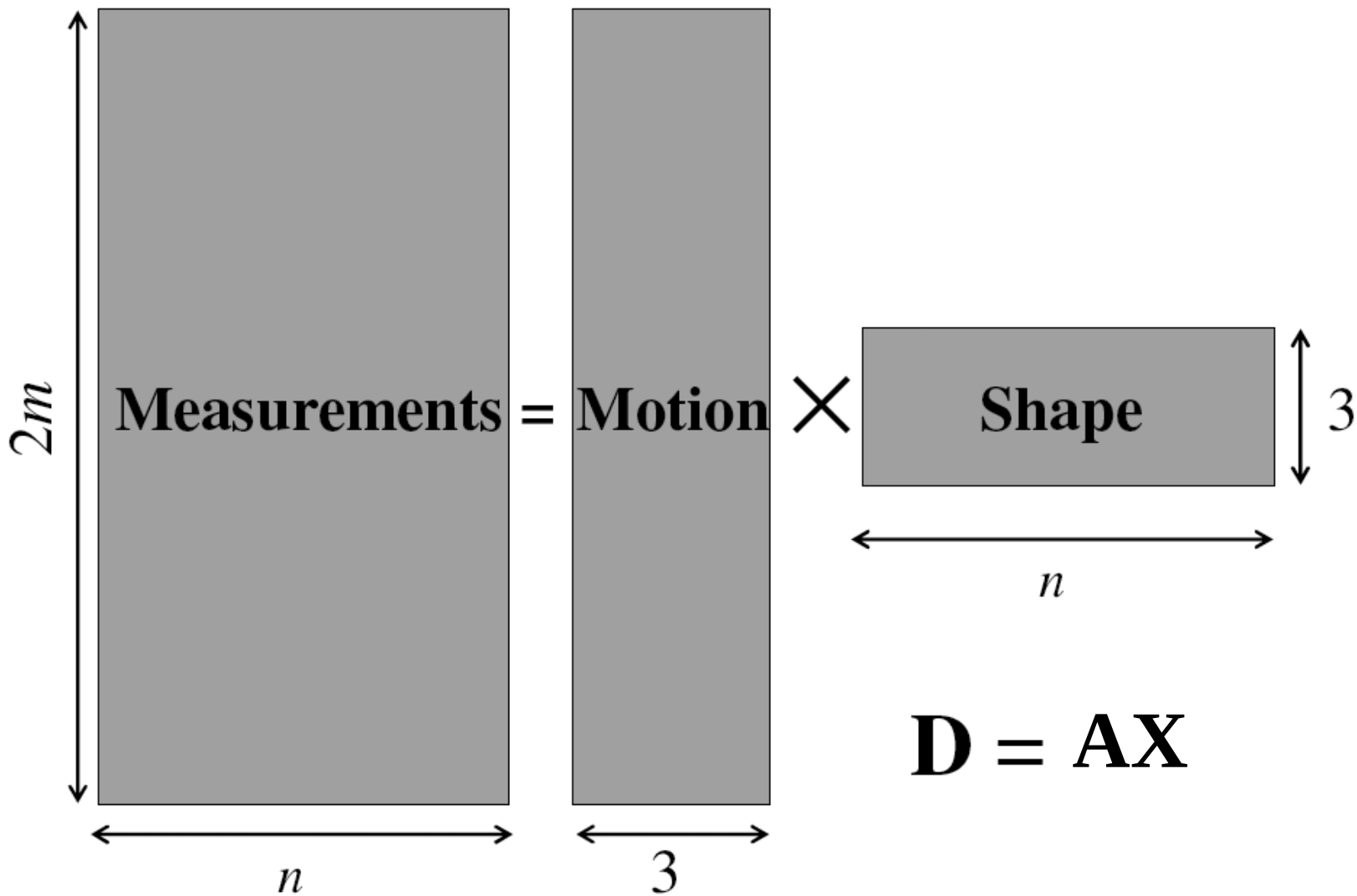
cameras ($2m$)

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

points (n)

What rank is the matrix of 2D points?

Factorizing the measurement matrix

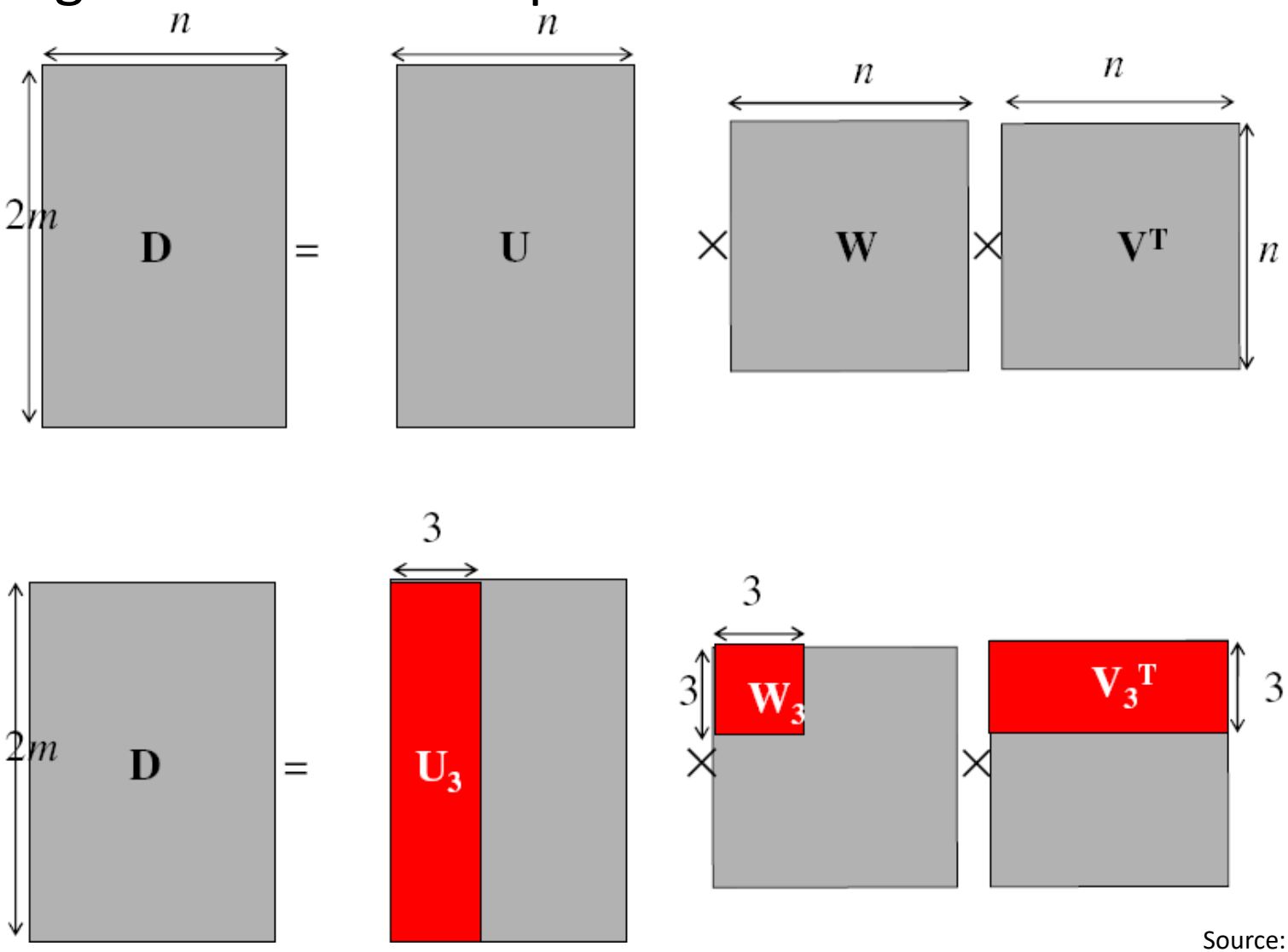


$$\mathbf{D} = \mathbf{AX}$$

Source: M. Hebert

Factorizing the measurement matrix

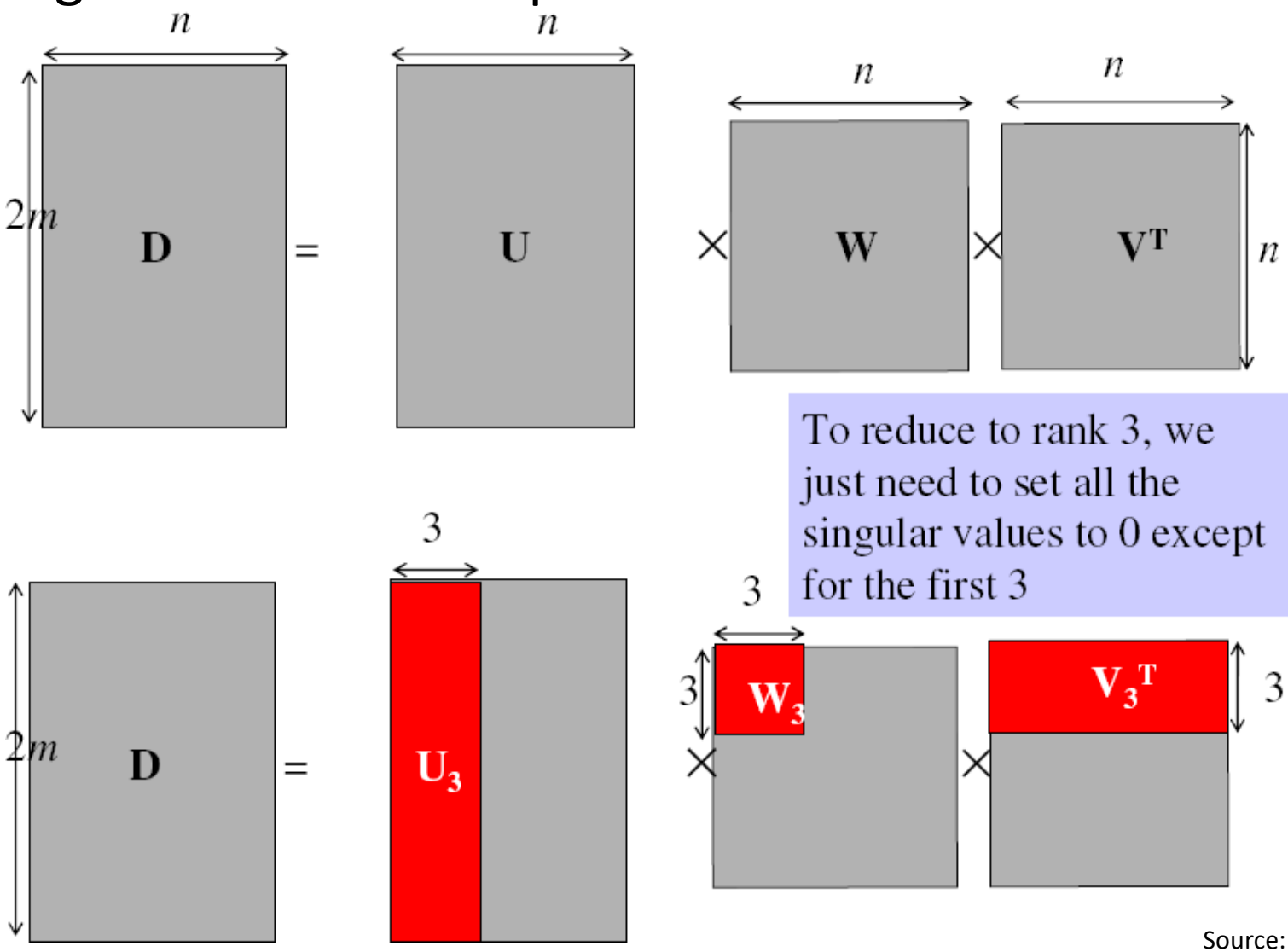
- Singular value decomposition of D:



Source: M. Hebert

Factorizing the measurement matrix

- Singular value decomposition of D:

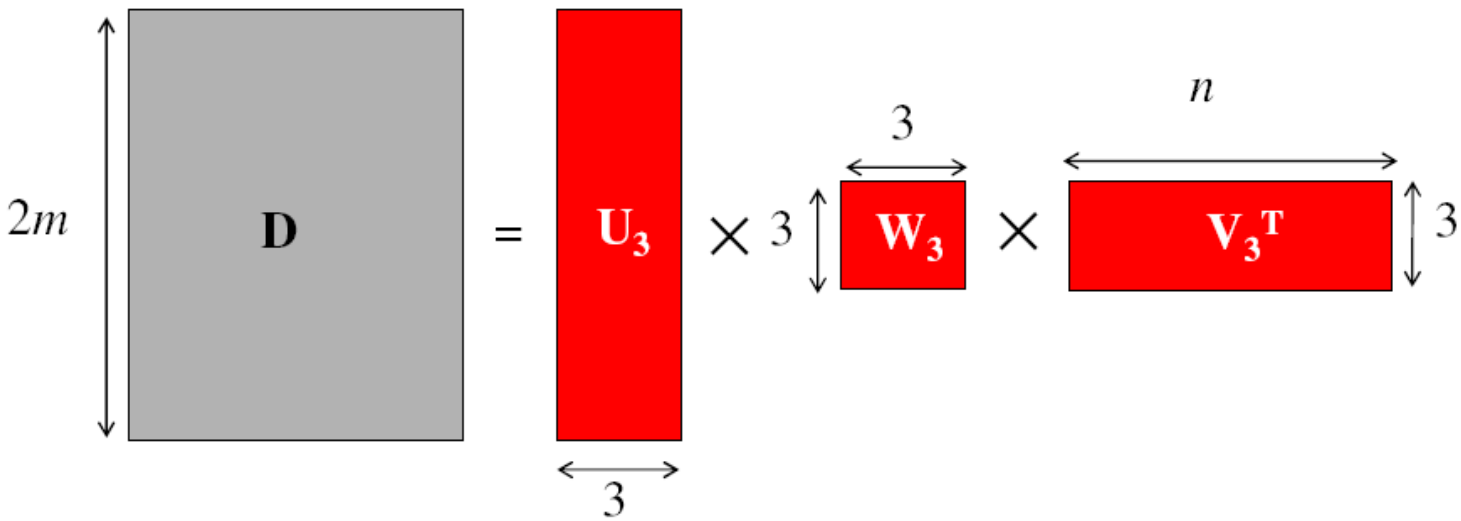


To reduce to rank 3, we just need to set all the singular values to 0 except for the first 3

Source: M. Hebert

Factorizing the measurement matrix

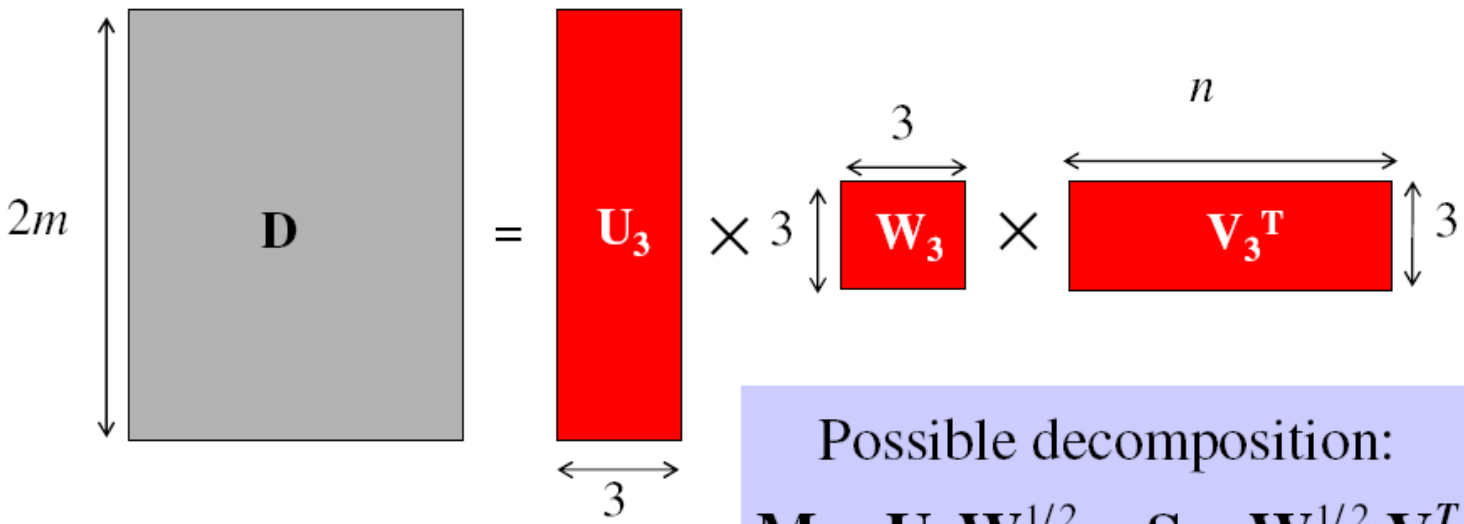
- Obtaining a factorization from SVD:



Source: M. Hebert

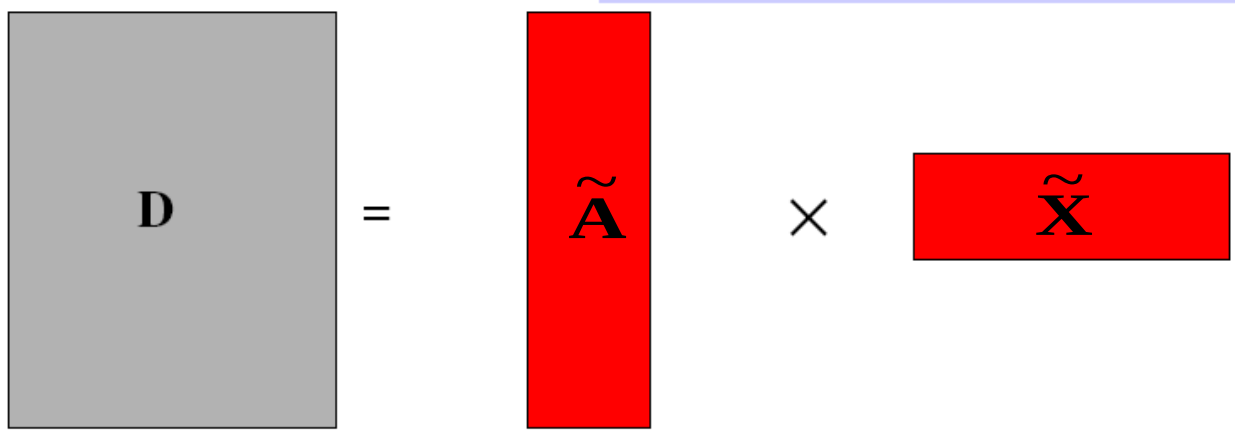
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



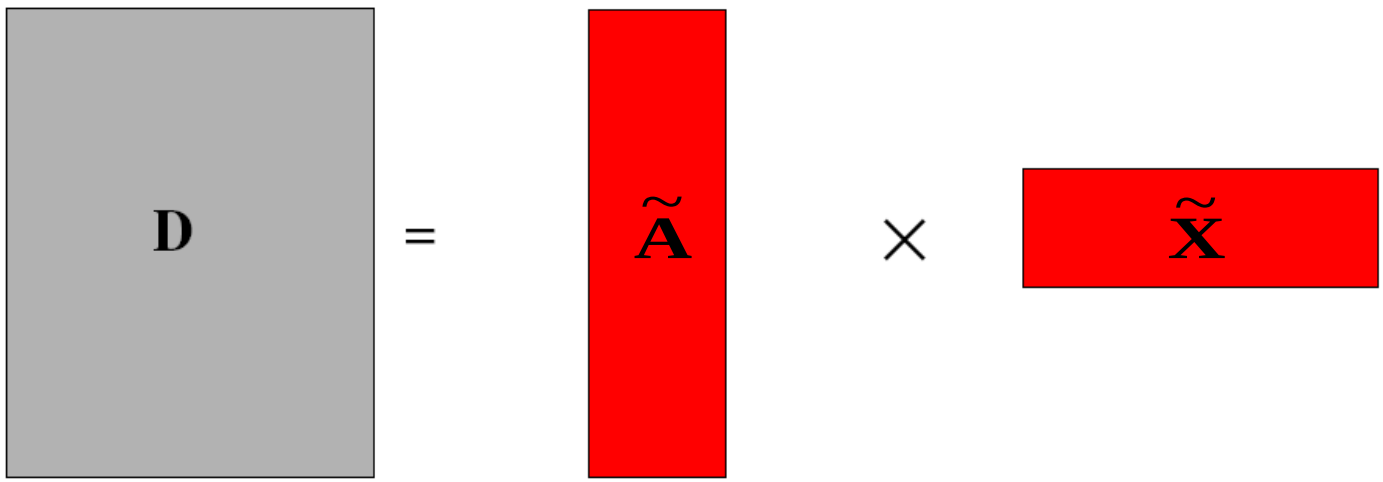
Possible decomposition:

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



Source: M. Hebert

Affine ambiguity



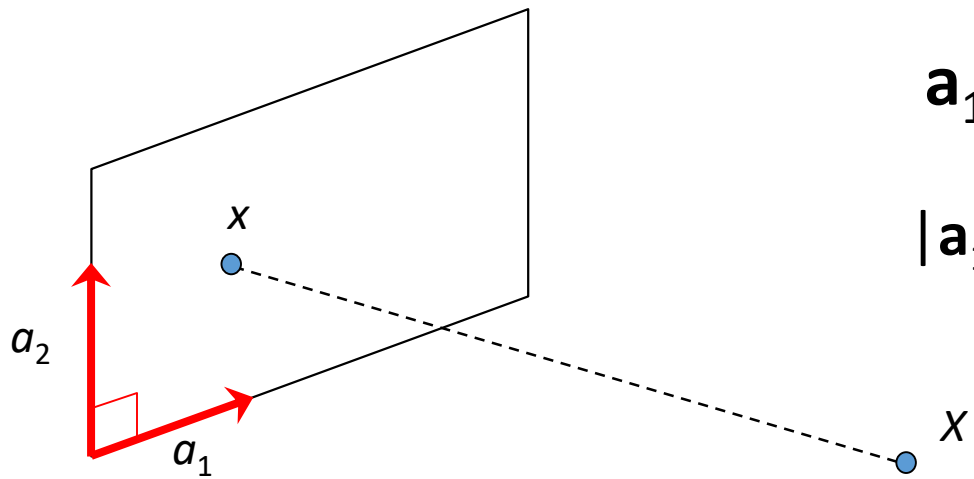
- The decomposition is not unique.
  We get the same **D** by using any 3×3 matrix **C** and applying the transformations **A → AC, X →C^{-1}X**

- Why?
  We have only an affine transformation and we have not enforced any Euclidean constraints (e.g., perpendicular image axes)

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

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

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

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

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

$$\Downarrow$$

$$[\qquad\qquad\qquad\qquad\qquad\qquad]\begin{bmatrix} L_{11} \\ L_{12} \\ L_{13} \\ L_{21} \\ L_{22} \\ L_{23} \\ L_{31} \\ L_{32} \\ L_{33} \end{bmatrix} = \mathrm{k}$$

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

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

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

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

Algorithm summary

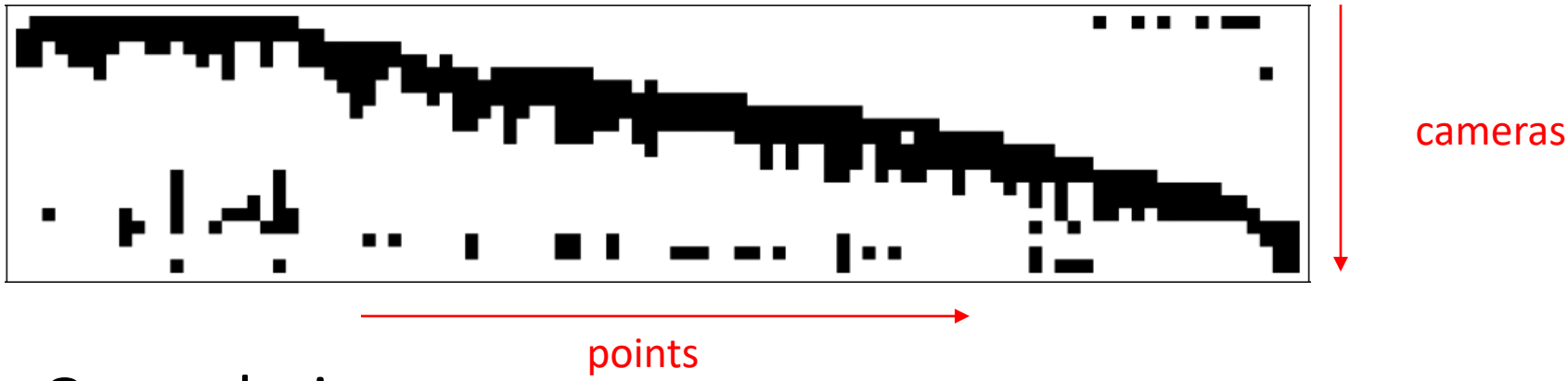
- Given: m images and n tracked features $\mathbf{x}_{ij}$
- For each image i, center the feature coordinates
- Construct a $2m \times n$ measurement matrix $\mathbf{D}$:
  - Column j contains the projection of point j in all views
  - Row i contains one coordinate of the projections of all the n points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

  $\mathbf{A} = \mathbf{U}_3\mathbf{W}_3^{½}$ and $\mathbf{S} = \mathbf{W}_3^{½}\,\mathbf{V}_3^{\mathrm{T}}$
- Eliminate affine ambiguity
  - Solve $\mathbf{L} = \mathbf{C}\mathbf{C}^{\mathbf{T}}$ using metric constraints
  - Solve $\mathbf{C}$ using Cholesky decomposition
  - Update $\mathbf{A}$ and $\mathbf{X}$: $\mathbf{A} = \mathbf{A}\mathbf{C}$, $\mathbf{S} = \mathbf{C}^{-1}\mathbf{S}$

Source: M. Hebert

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:



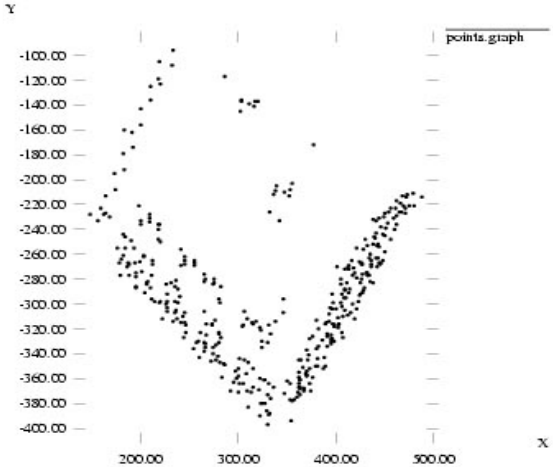
One solution:

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

Reconstruction results

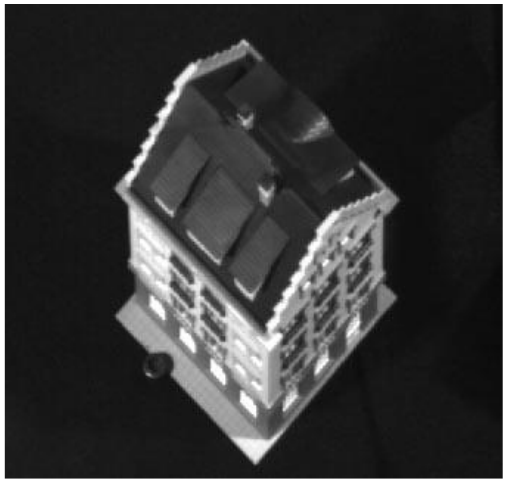
1

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

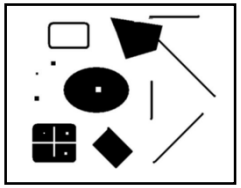
- Short explanation of Affine SfM: class notes from Lischinksi and Gruber

  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

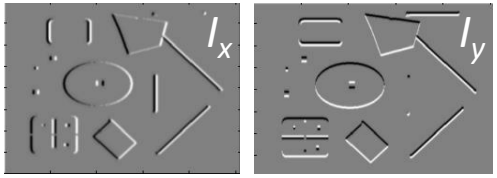
Review of Affine SfM from Interest Points

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

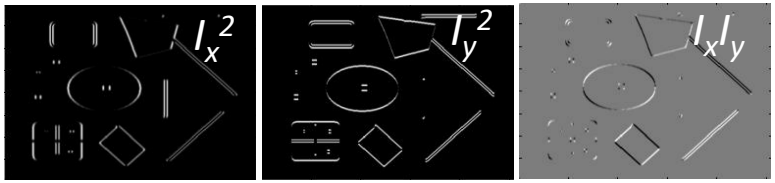


$$\mu(\sigma_I,\sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives



2. Square of derivatives



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

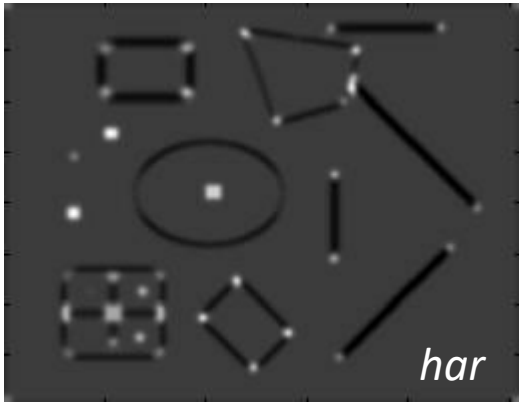
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

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

Original (x,y) position

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

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

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

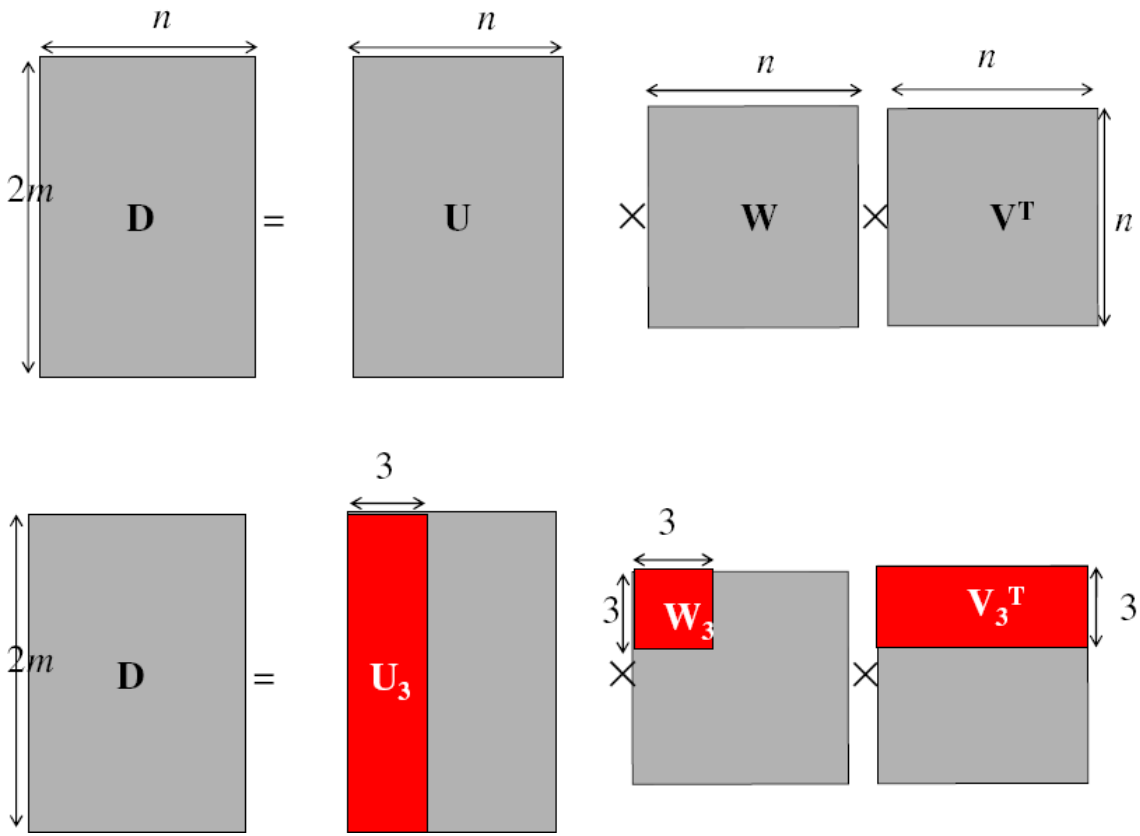
d) Recalculate I_t

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



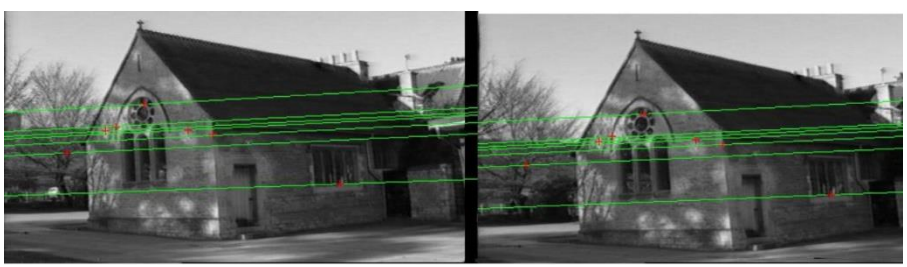
Solve for orthographic constraints

HW 3 – Part 1 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
```



8-Point Algorithm for Recovering F

- Correspondence Relation

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

1. Normalize image coordinates

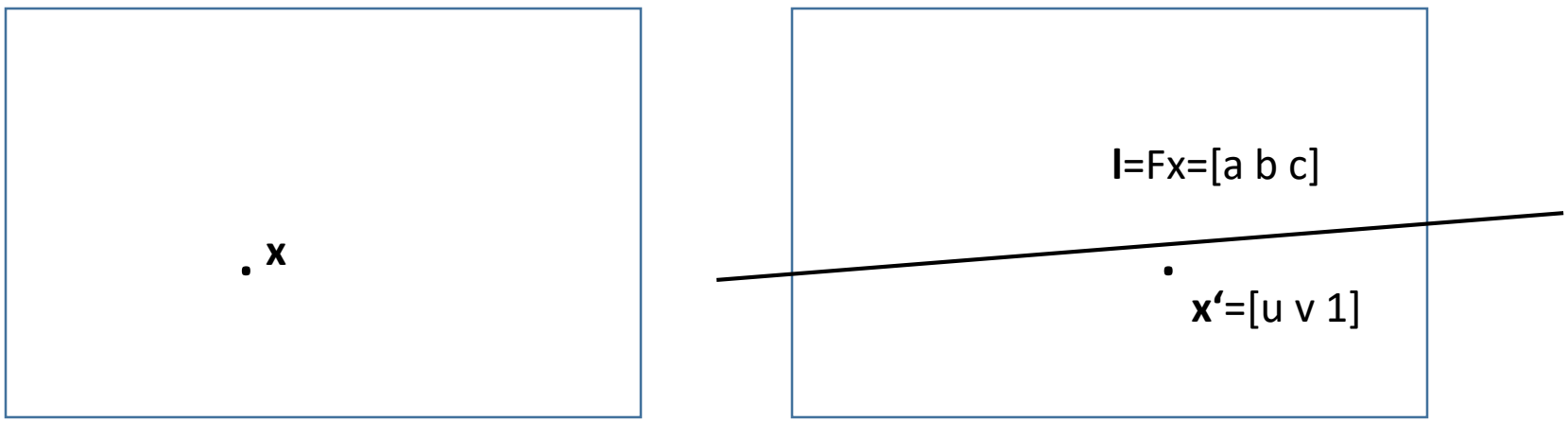
$$\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x} \quad \tilde{\mathbf{x}}' = \mathbf{T}'\mathbf{x}'$$

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: $\mathbf{F} = \mathbf{T}'^T \tilde{\mathbf{F}} \mathbf{T}$

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

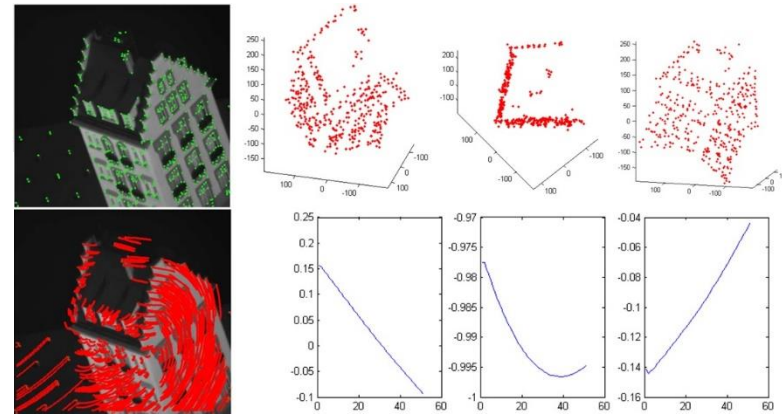
Distance of point to epipolar line



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

HW 3 – Part 2 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, 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: $\mathbf{D} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^T$
  - 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 $\mathbf{S} = \mathbf{W}_3^{1/2}\,\mathbf{V}_3^T$
- Eliminate affine ambiguity
  - Solve $\mathbf{L} = \mathbf{C}\mathbf{C}^T$ using metric constraints
  - Solve **C** using Cholesky decomposition
  - Update **A** and **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 diplay
motion and shape

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

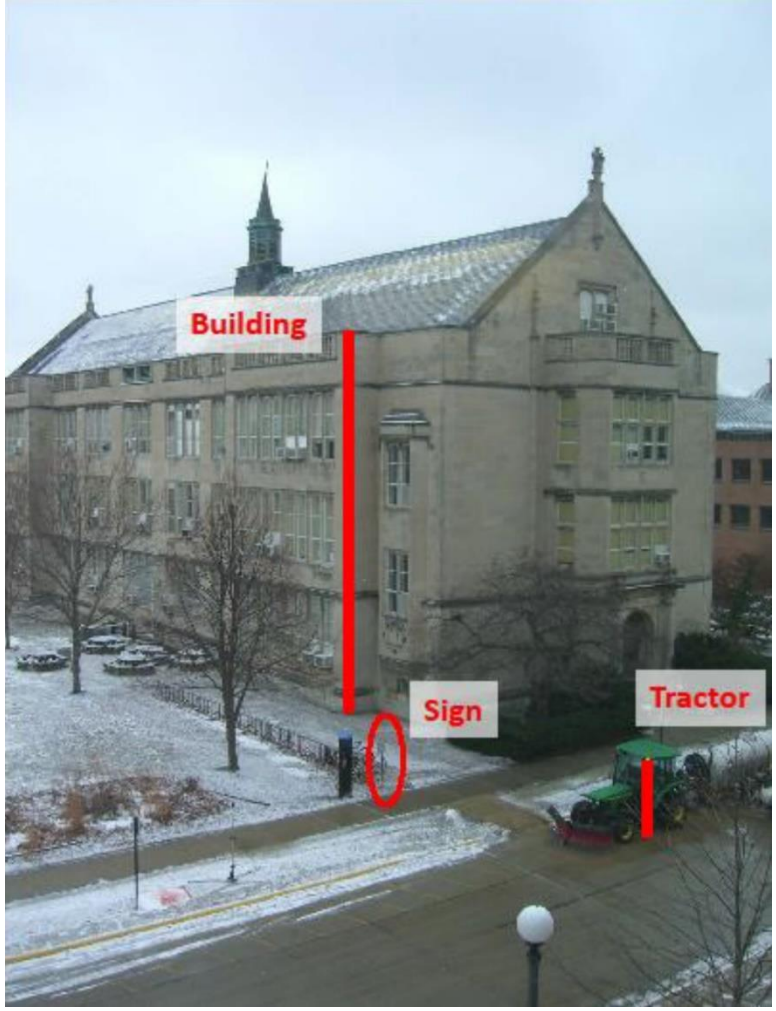
HW 3 – Part 2 Affine SfM

- 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 **L = CC^T**
  - L = reshape(A\b, [3,3]); % A - 3m x 9, b – 3m x 1
- Recover **C** from **L** by Cholesky decomposition: **L = CC^T**
- Update **A** and **X**: **A = AC, X = C^{-1}X**

HW 3 – Graduate credits
Single-view metrology

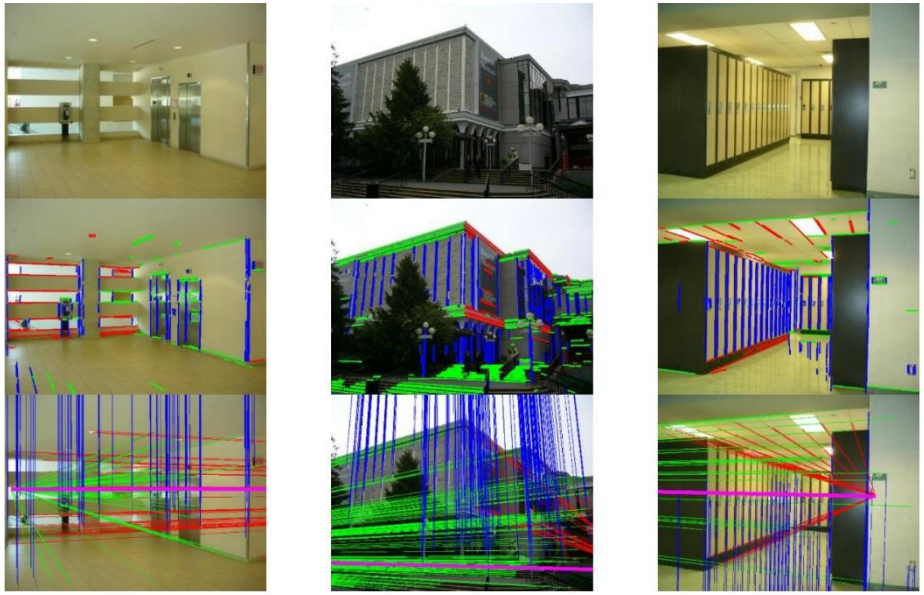


Assume Sign = 1.65m

Question: What's the heights of

- Building
- Tractor
- Camera

HW 3 – Graduate credits
Automatic vanishing point detection



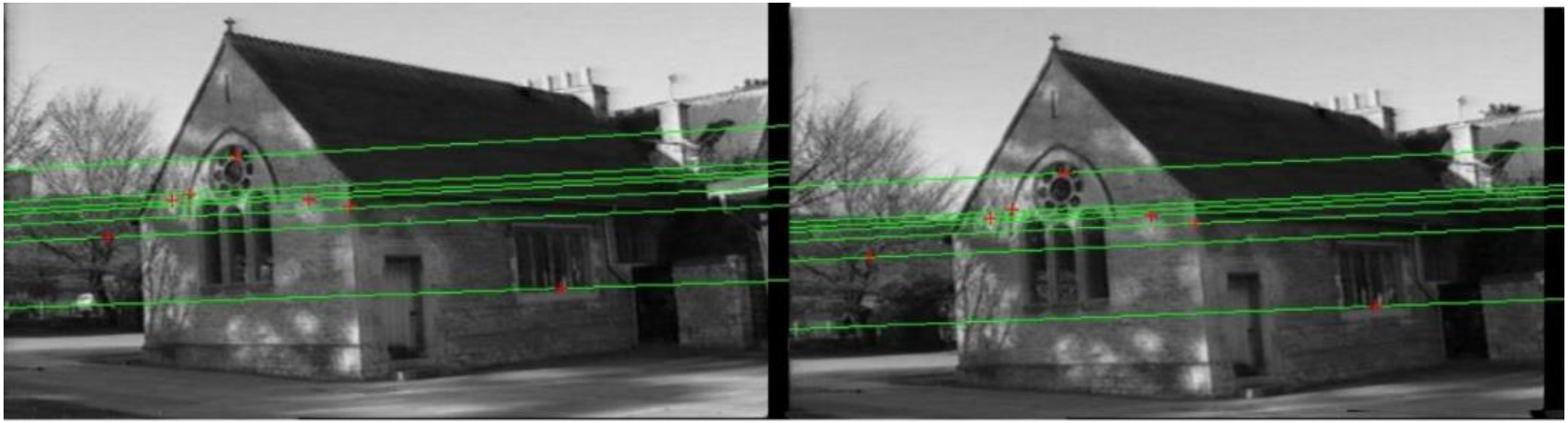
Input:

- lines: a matrix of size [NumLines x 5] where each row represents a line segment with (x1, y1, x2, y2, lineLength)

Output:

- VP: [2 x 3] each column corresponds to a vanishing point in the order of X, Y, Z
- lineLabel: [NumLine x 3] each column is a logical vector indicating which line segments correspond to the vanishing point.

HW 3 – Graduate credits
Epipolar Geometry



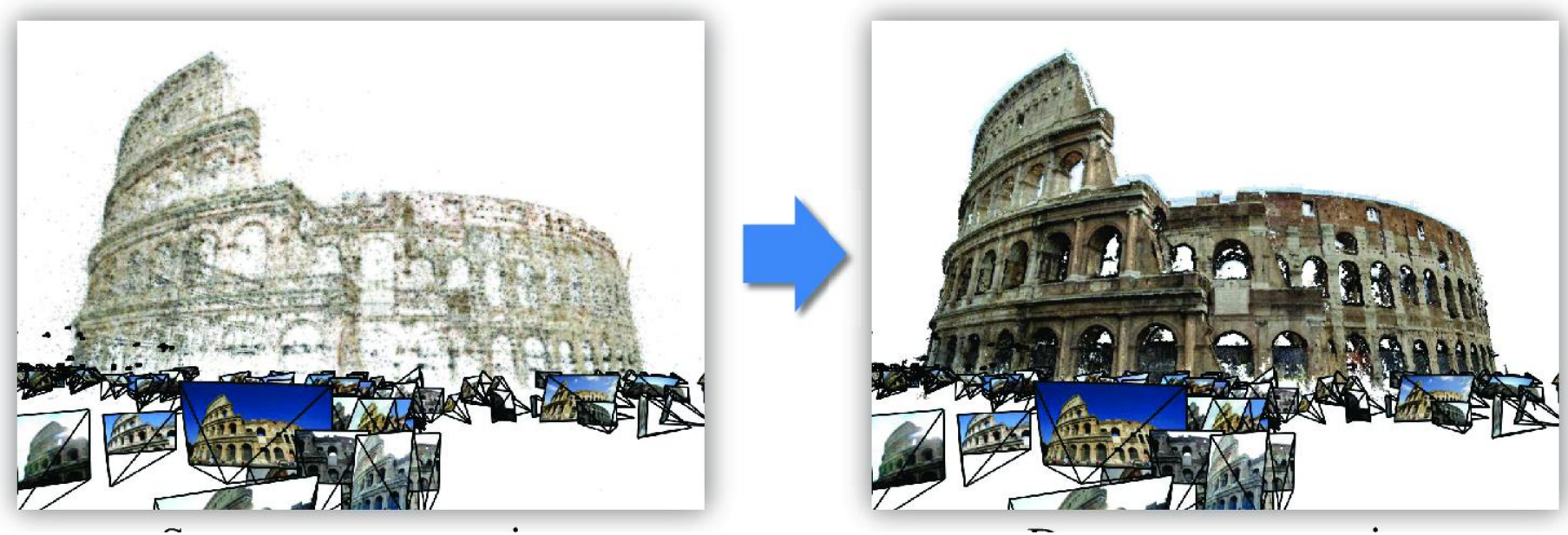
Try “un-normalized” 8-point algorithm.

Report and compare the accuracy with the normalized version

HW 3 – Graduate credits
Affine structure from motion

- Missing track completion.
- Some keypoints will fall out of frame, or come into frame throughout the sequence.
- Fill in the missing data and visualize the predicted positions of points that aren't visible in a particular frame.

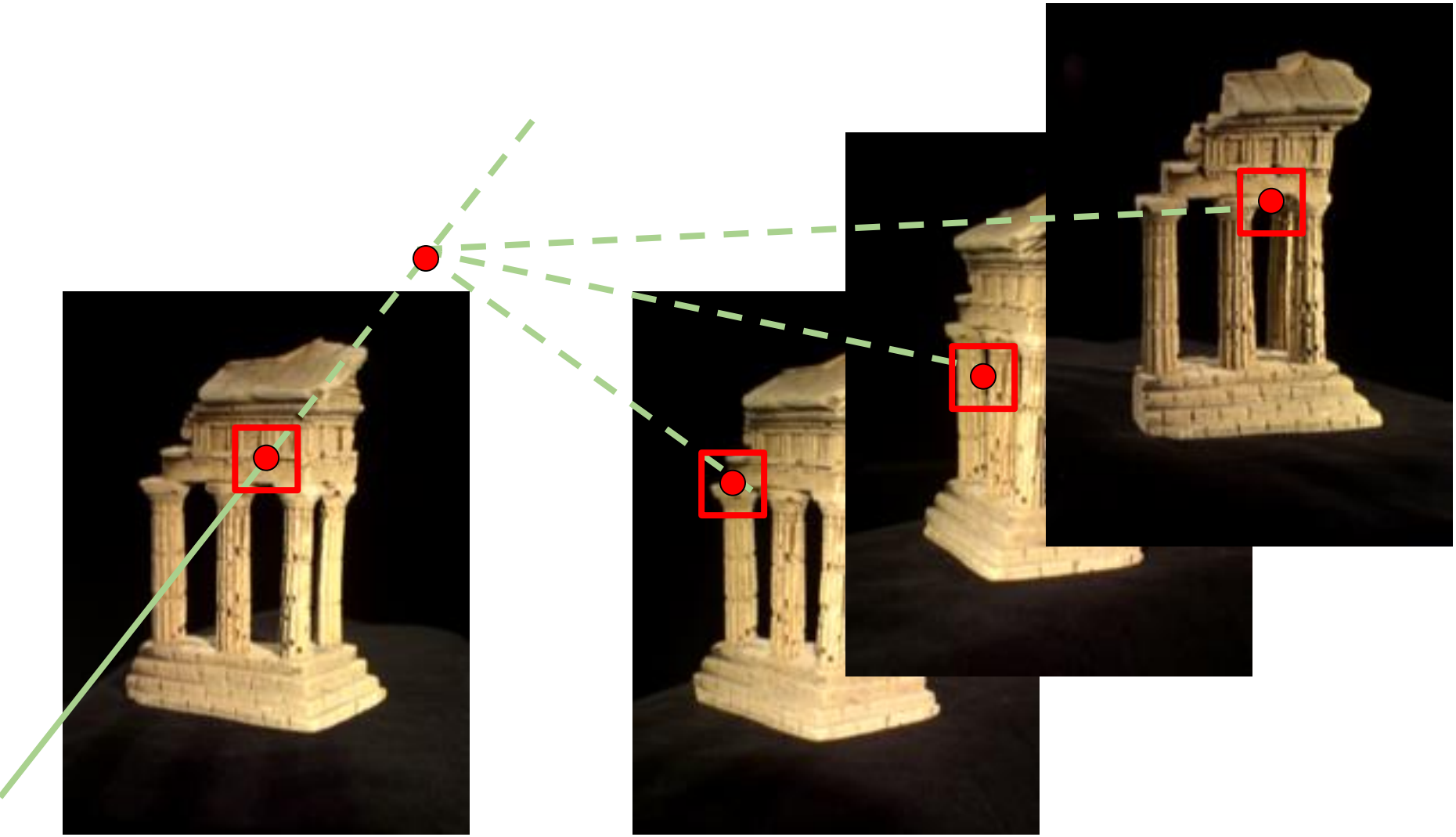
Multi-view stereo



Multi-view stereo

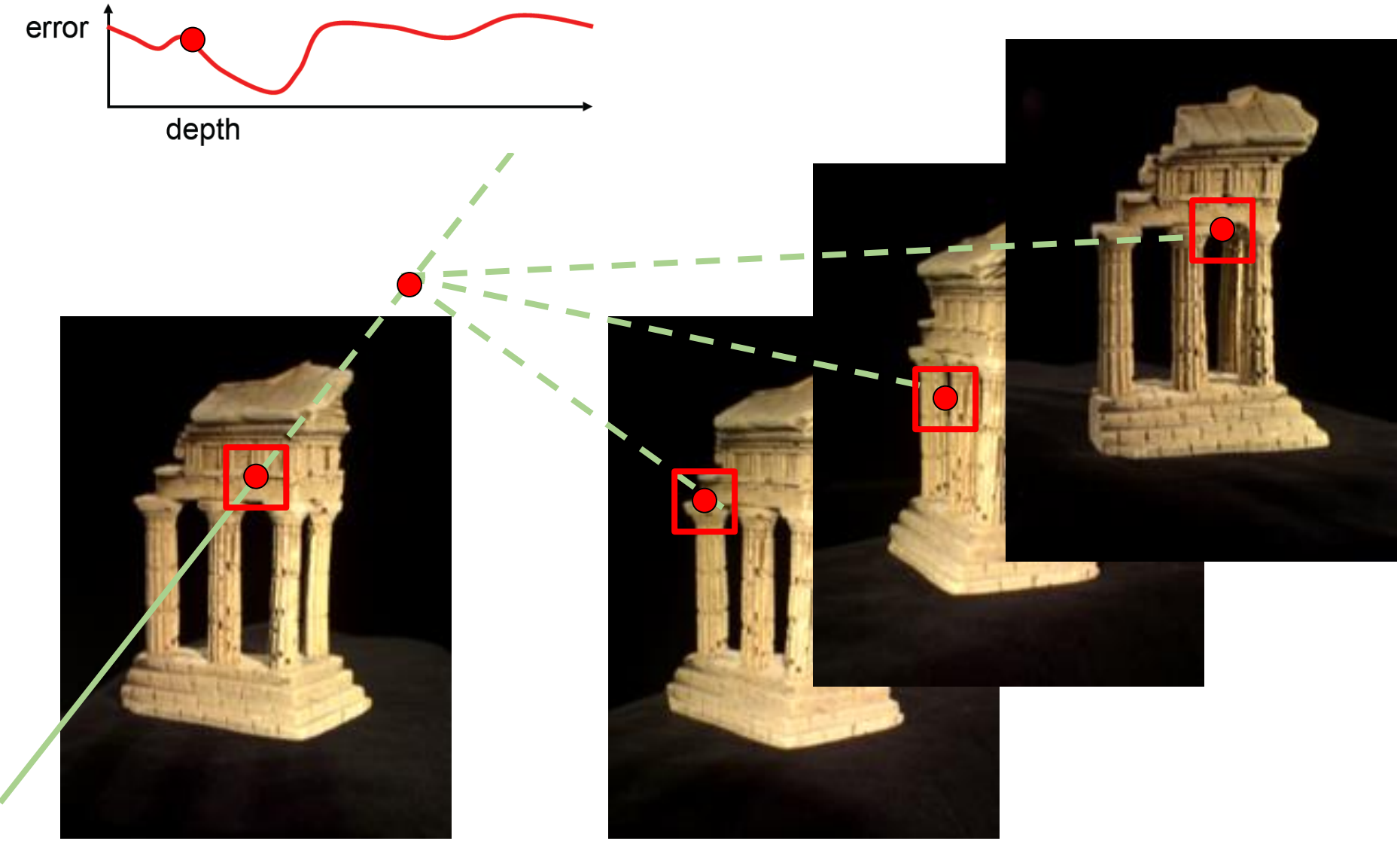
- Generic problem formulation: given several images of the same object or scene, compute a representation of its 3D shape
- "Images of the same object or scene"
  - Arbitrary number of images (from two to thousands)
  - Arbitrary camera positions (special rig, camera network or video sequence)
  - Calibration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

Multi-view stereo: Basic idea



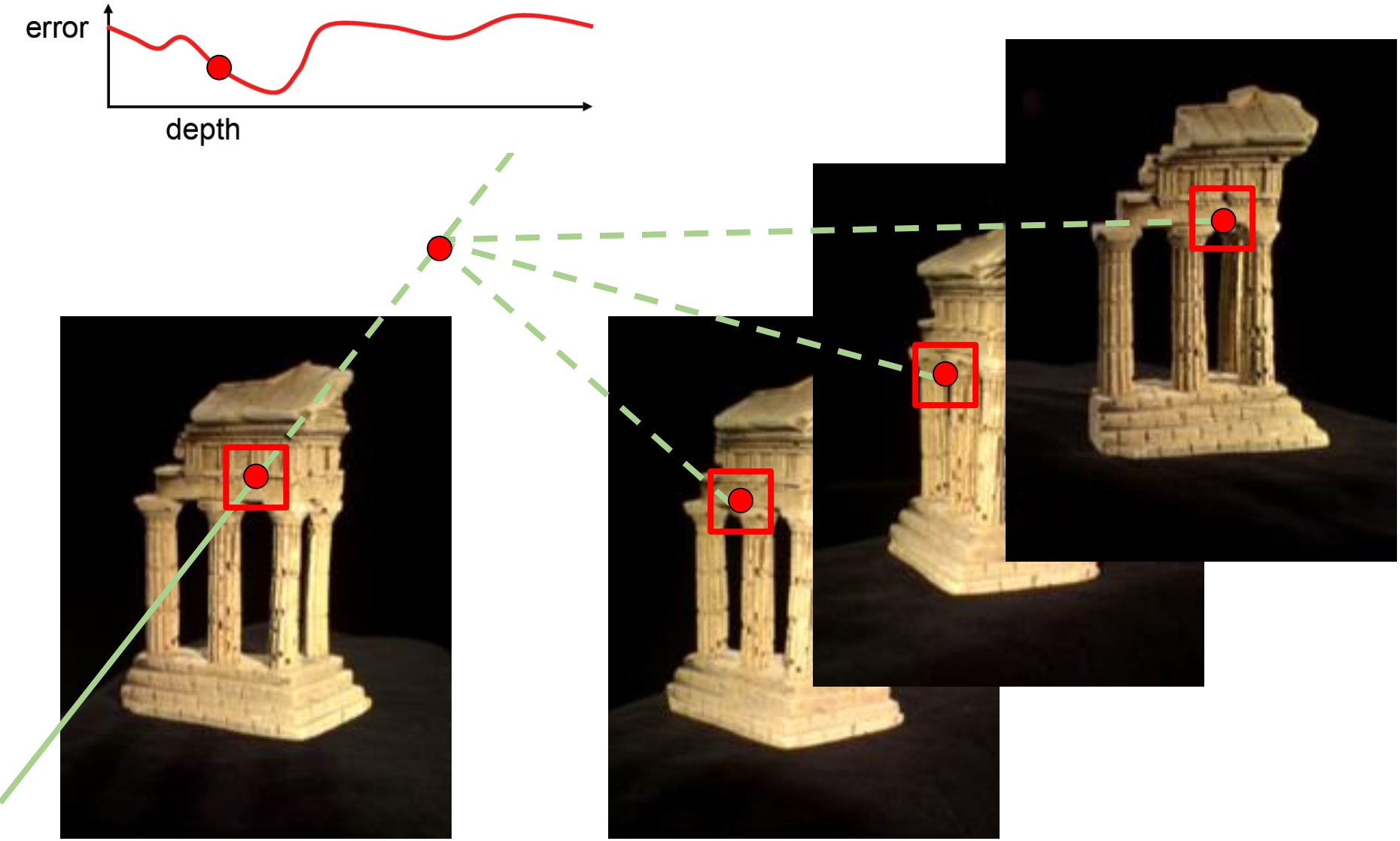
Source: Y. Furukawa

Multi-view stereo: Basic idea



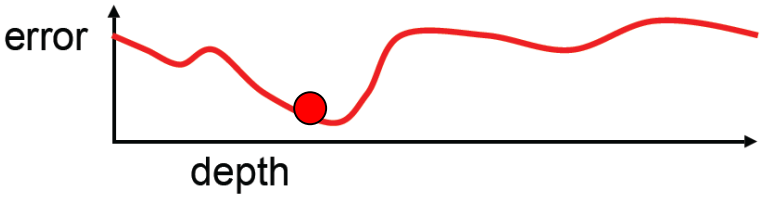
Source: Y. Furukawa

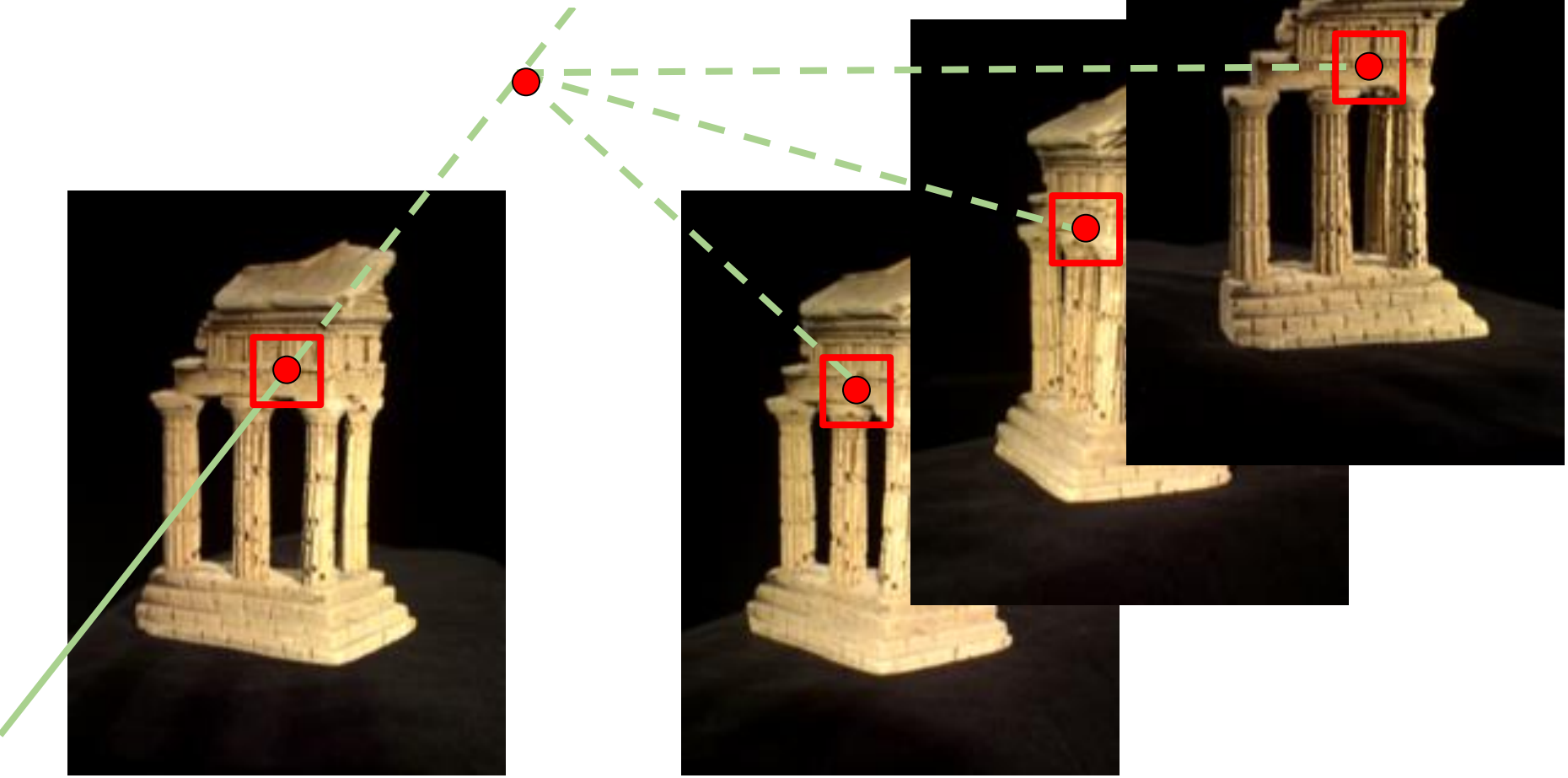
Multi-view stereo: Basic idea



Source: Y. Furukawa

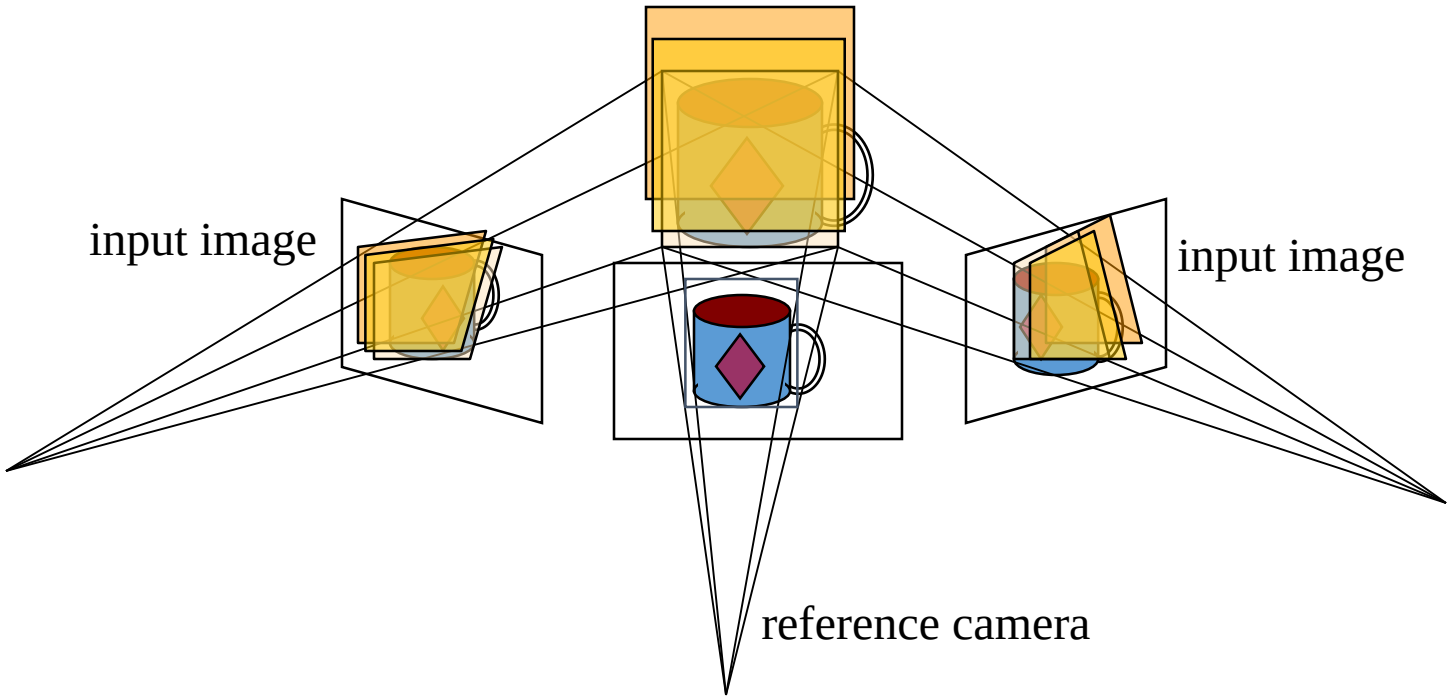
Multi-view stereo: Basic idea





Source: Y. Furukawa

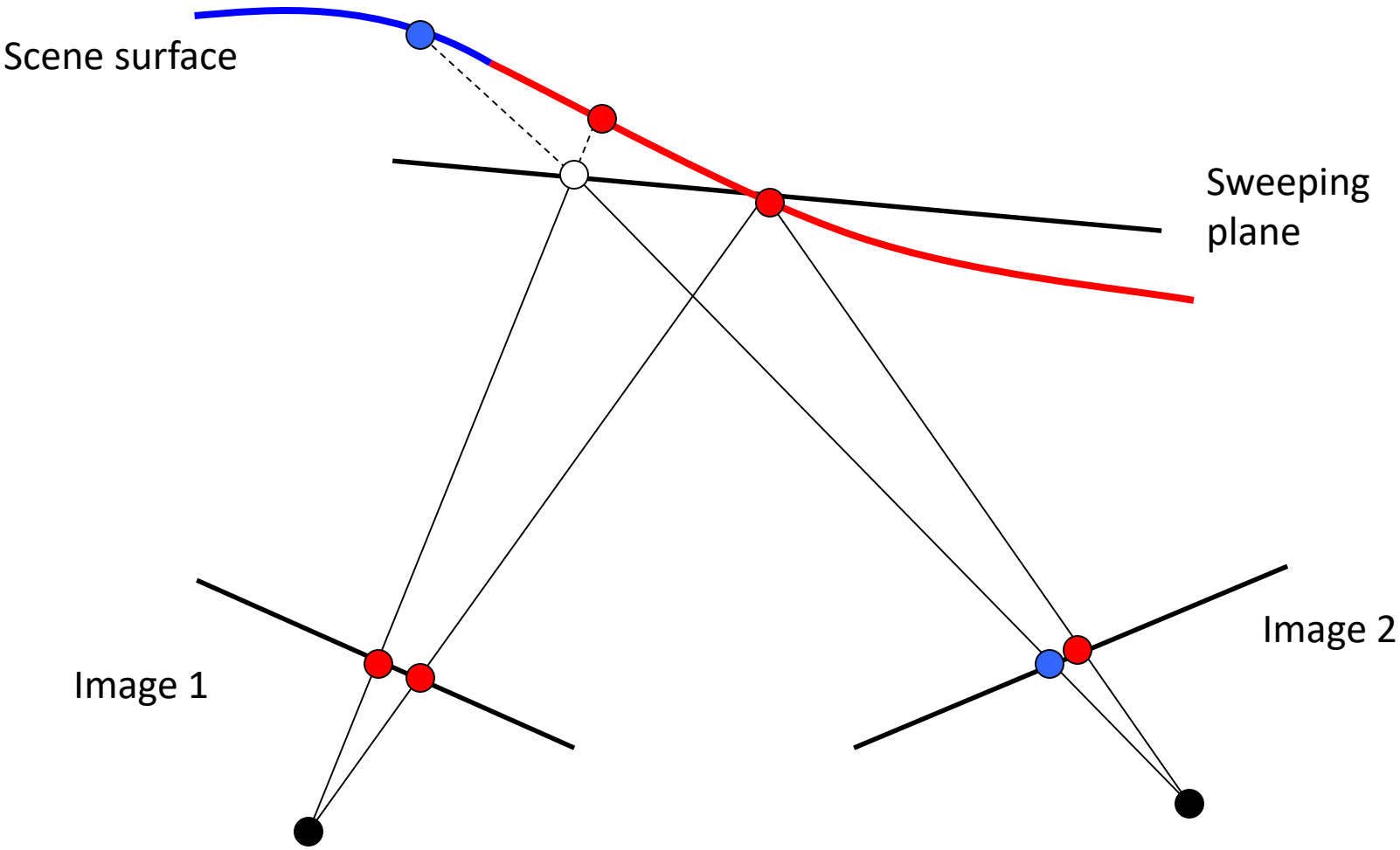
Plane Sweep Stereo



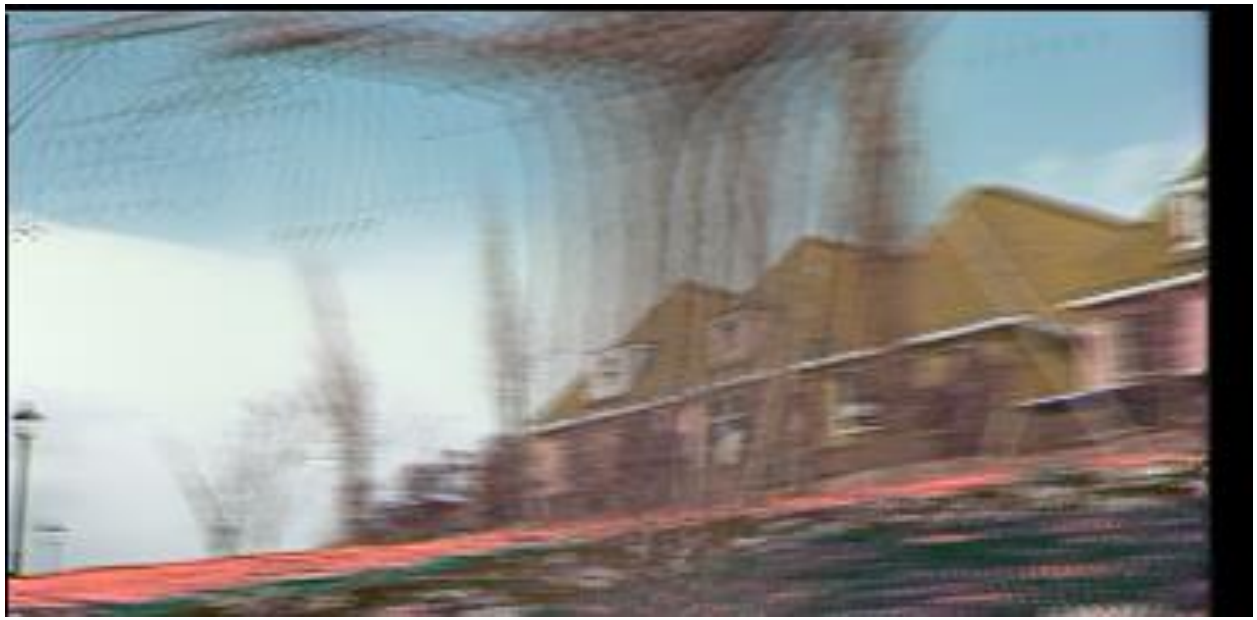
- Sweep family of planes at different depths w.r.t. a reference camera
- For each depth, project each input image onto that plane
- This is equivalent to a homography warping each input image into the reference view
- What can we say about the scene points that are at the right depth?

R. Collins. A space-sweep approach to true multi-image matching. CVPR 1996.

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

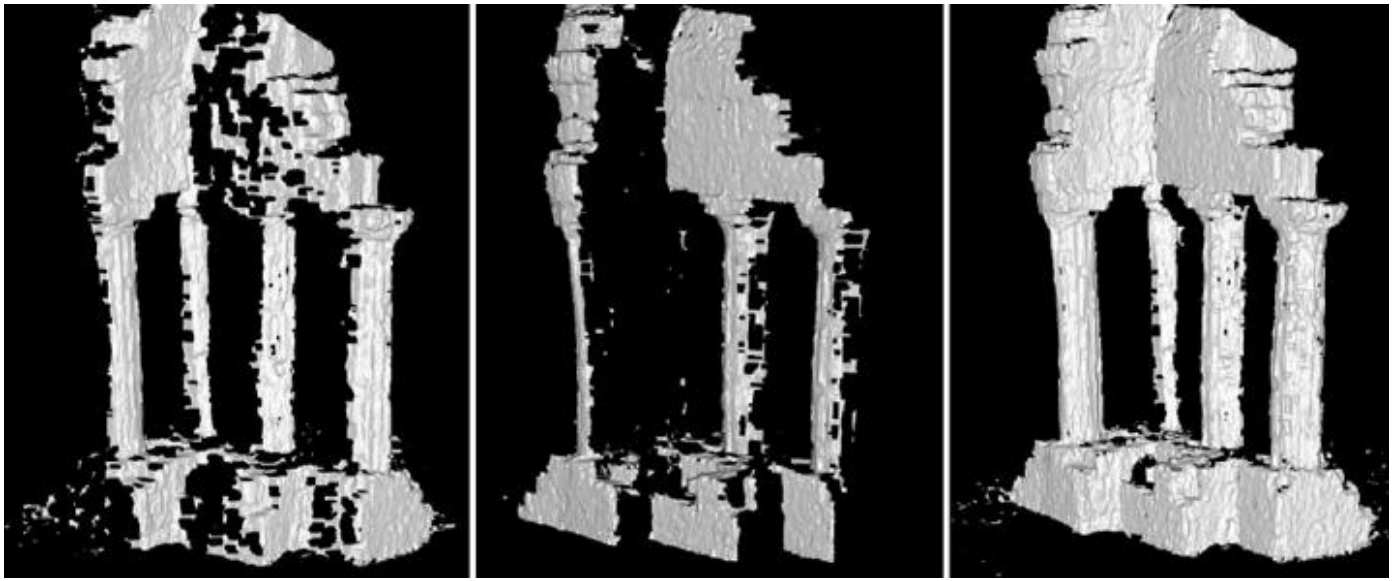
- Can be accelerated using graphics hardware

R. Yang and M. Pollefeys. *Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware*, CVPR 2003

Merging depth maps

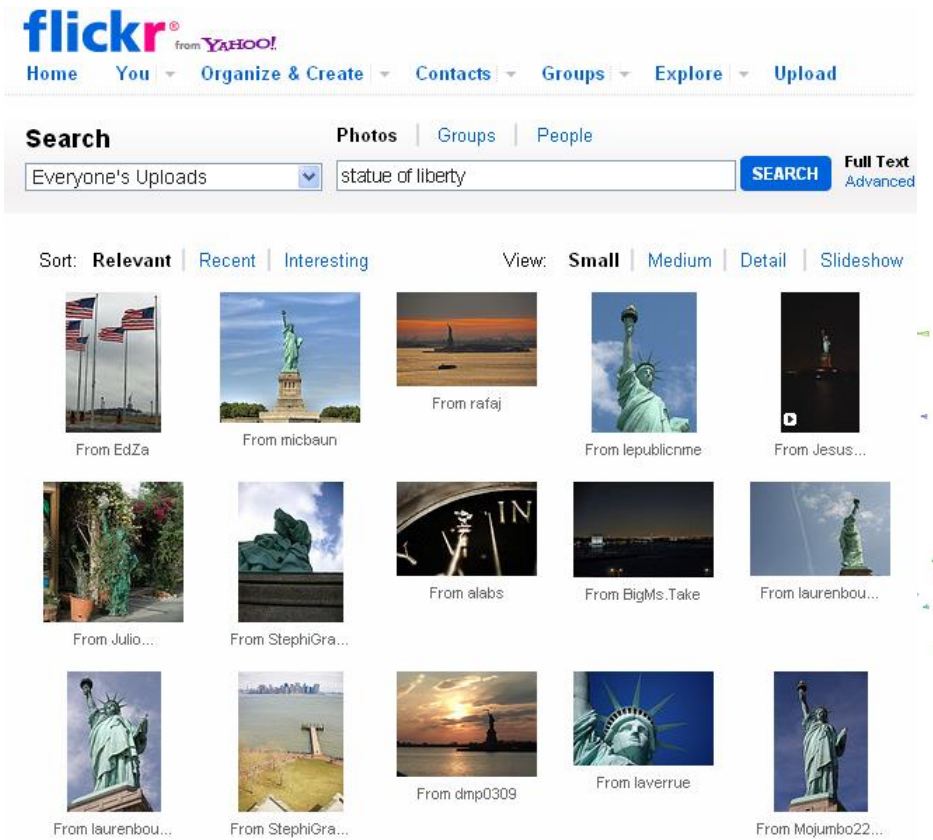
- Given a group of images, choose each one as reference and compute a depth map w.r.t. that view using a multi-baseline approach
- Merge multiple depth maps to a volume or a mesh (see, e.g., Curless and Levoy 96)

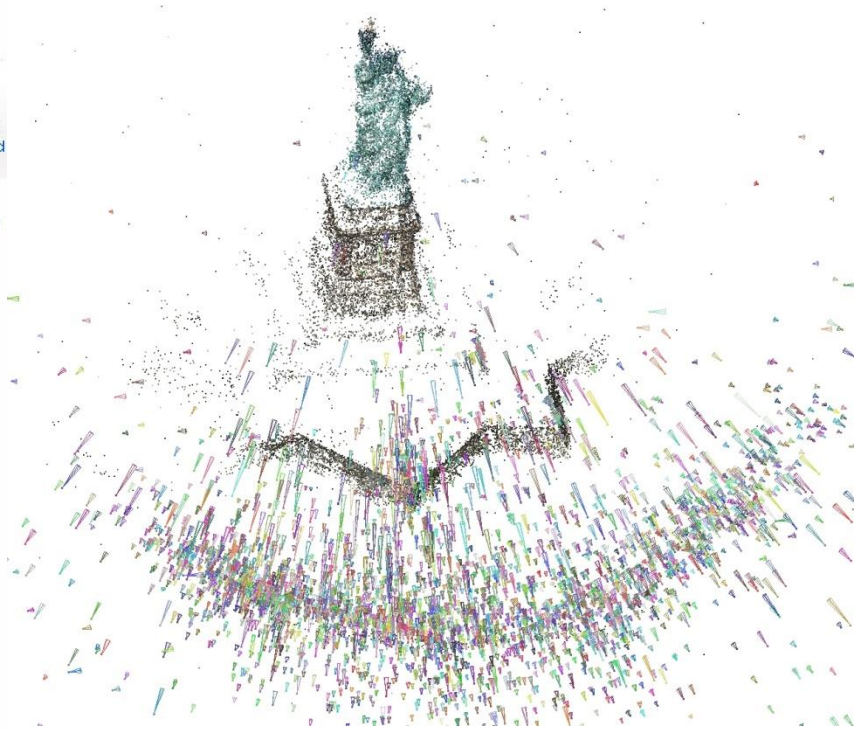
Map 1 Map 2 Merged

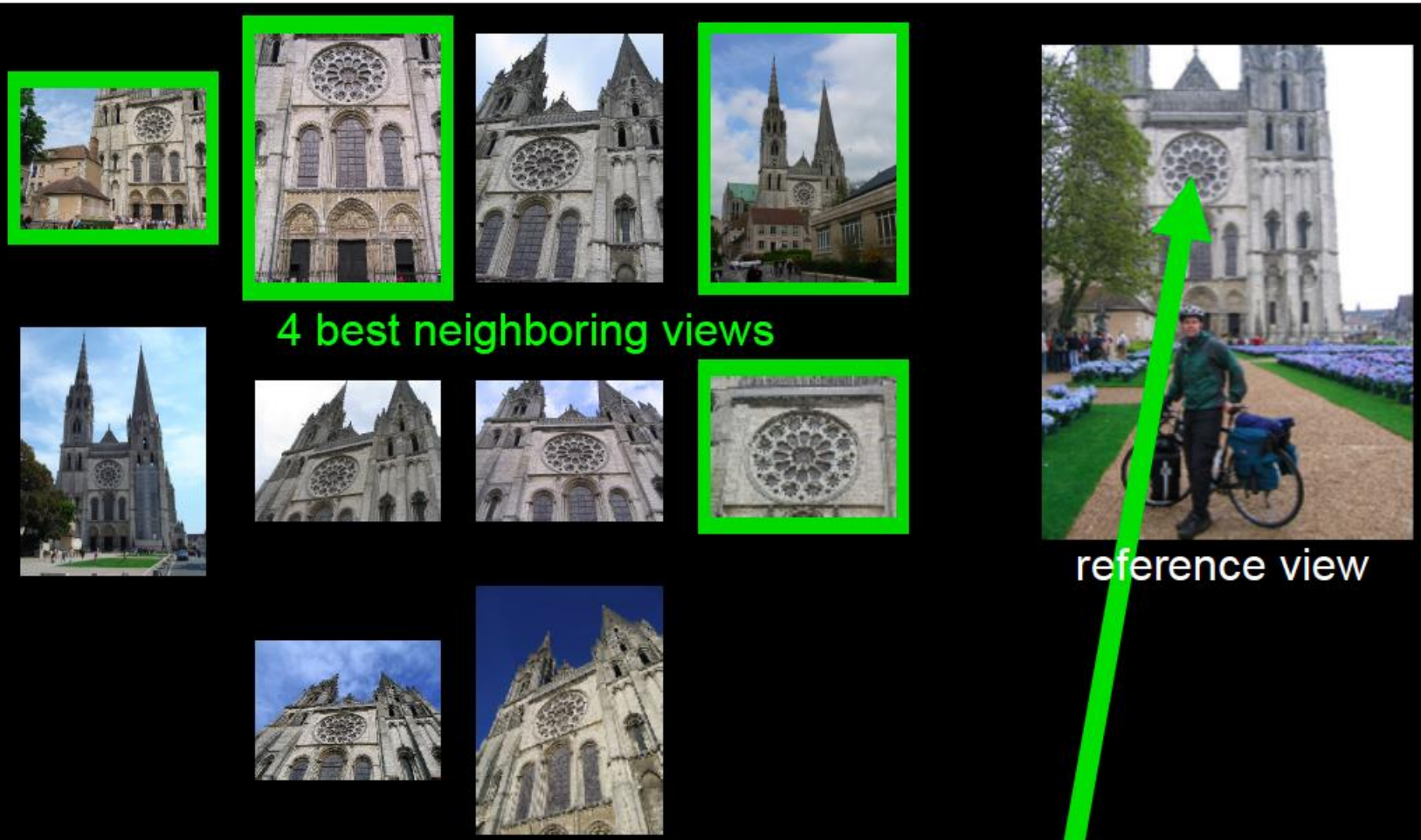


Stereo from community photo collections

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

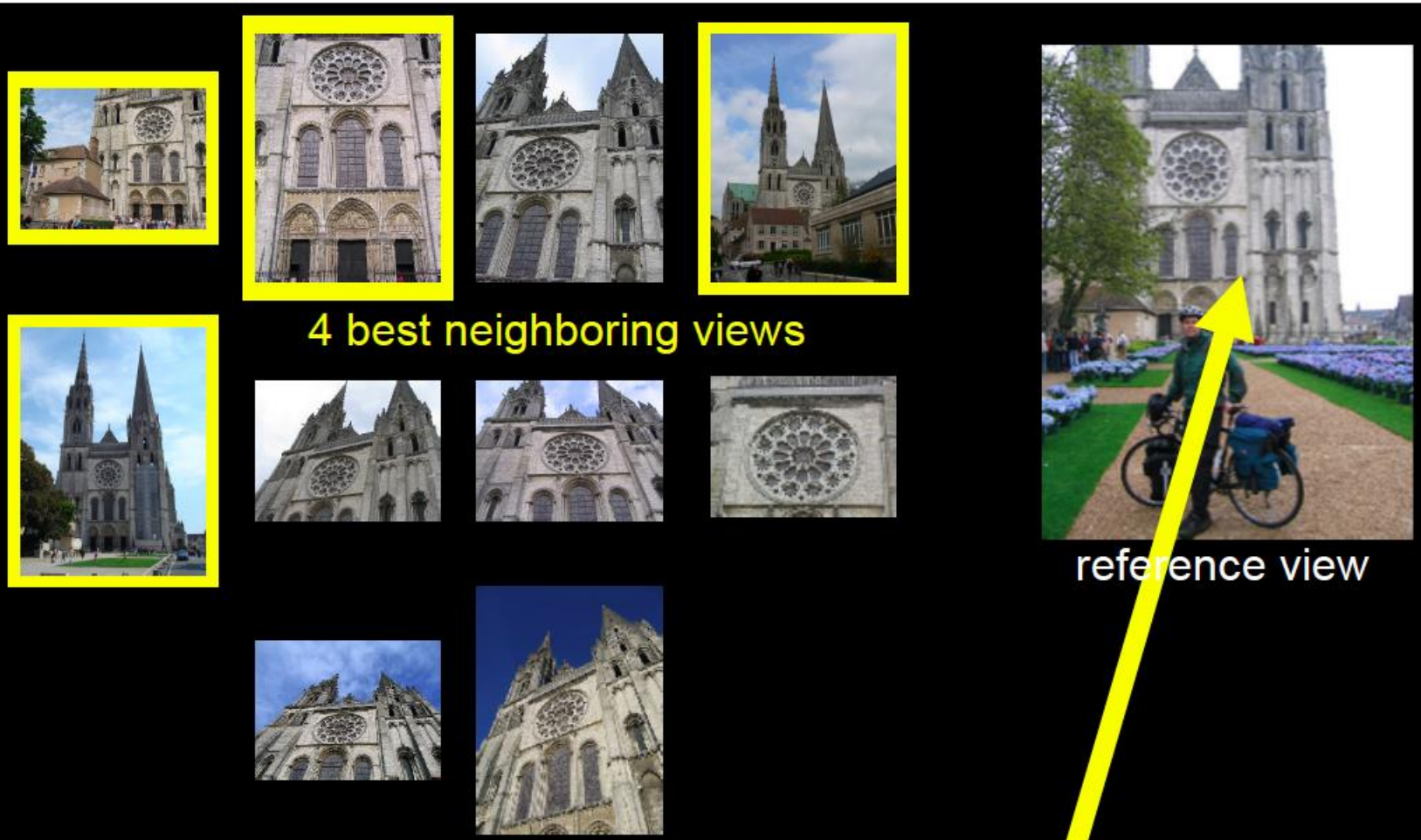






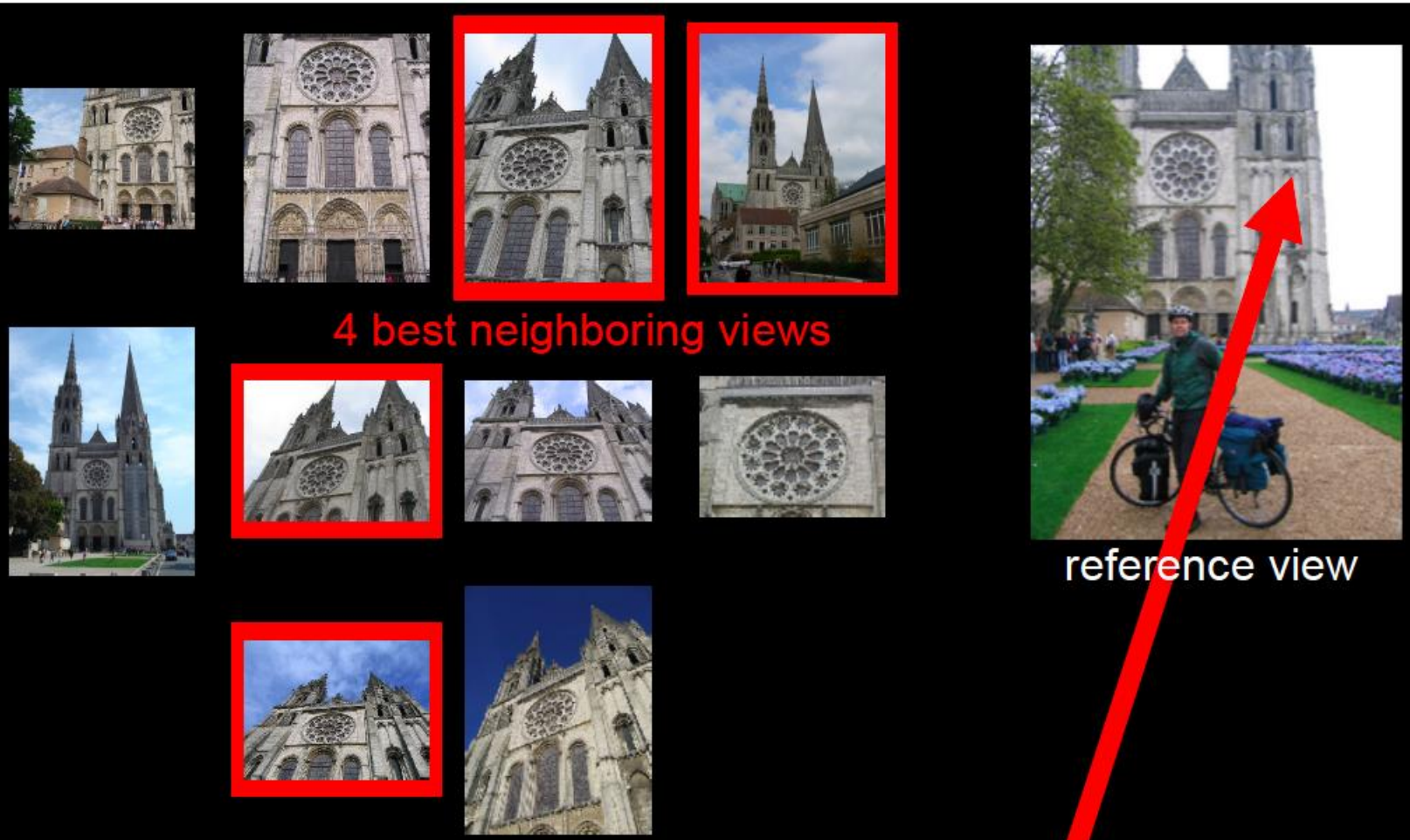
Local view selection

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



Local view selection

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



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

- 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=kyIzMr917Rc, 3D Computer Vision: Past, Present, and Future

Next class

- Grouping and Segmentation