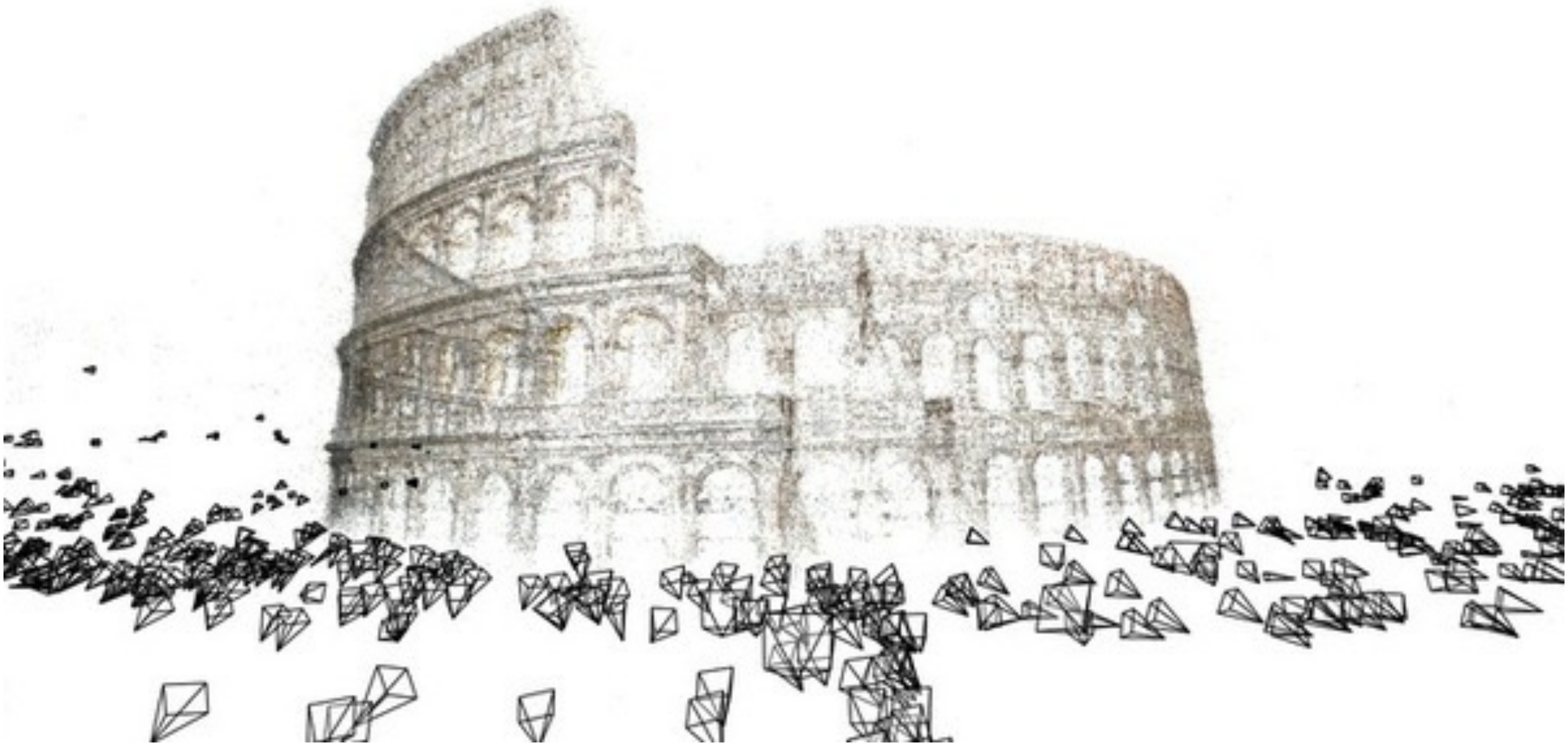
Structure from motion



16-385 Computer Vision
Spring 2020, Lecture 29

http://www.cs.cmu.edu/~16385/

Overview of today's lecture

- A note on normalization.
- Two-view structure from motion.
- Ambiguities in structure from motion.
- Affine structure from motion.
- Multi-view structure from motion.
- Large-scale structure from motion.

Slide credits

Many of these slides were adapted from:

- Kris Kitani (16-385, Spring 2017).
- Noah Snavely (Cornell University).
- Rob Fergus (New York University).

A note on normalization

Estimating F – 8-point algorithm

- The fundamental matrix F is defined by

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

for any pair of matches x and x' in two images.

- Let x=($u,v,1$)$^{\mathrm{T}}$ and x'=($u',v',1$)$^{\mathrm{T}}$,

$$\mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

each match gives a linear equation

$$uu' f_{11} + vu' f_{12} + u' f_{13} + uv' f_{21} + vv' f_{22} + v' f_{23} + uf_{31} + vf_{32} + f_{33} = 0$$

Problem with 8-point algorithm

$$
\begin{array}{c}
\begin{bmatrix}
u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\
u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1
\end{bmatrix} \\
\begin{array}{ccccccccc}
\sim 10000 & \sim 10000 & \sim 100 & \sim 10000 & \sim 10000 & \sim 100 & \sim 100 & \sim 100 & 1
\end{array}
\end{array}
\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0
$$

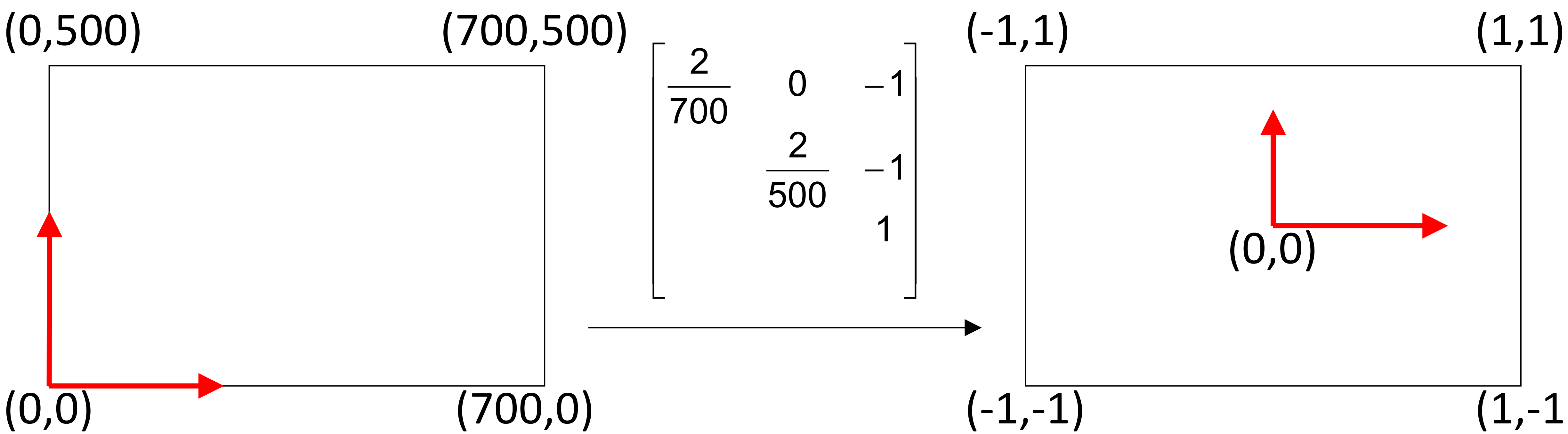
!

Orders of magnitude difference
between column of data matrix
$\rightarrow$ least-squares yields poor results

Normalized 8-point algorithm

normalized least squares yields good results

Transform image to ~[-1,1]x[-1,1]



$$\begin{bmatrix} \frac{2}{700} & 0 & -1 \\ & \frac{2}{500} & -1 \\ & & 1 \end{bmatrix}$$

Normalized 8-point algorithm

1. Transform input by $\hat{\mathbf{x}}_{\mathbf{i}} = \mathbf{T}\mathbf{x}_{\mathbf{i}}$, $\hat{\mathbf{x}}_{\mathbf{i}}' = \mathbf{T}\mathbf{x}_{\mathbf{i}}'$
2. Call 8-point on $\hat{\mathbf{x}}_{\mathbf{i}}, \hat{\mathbf{x}}_{\mathbf{i}}'$ to obtain $\hat{\mathbf{F}}$
3. $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

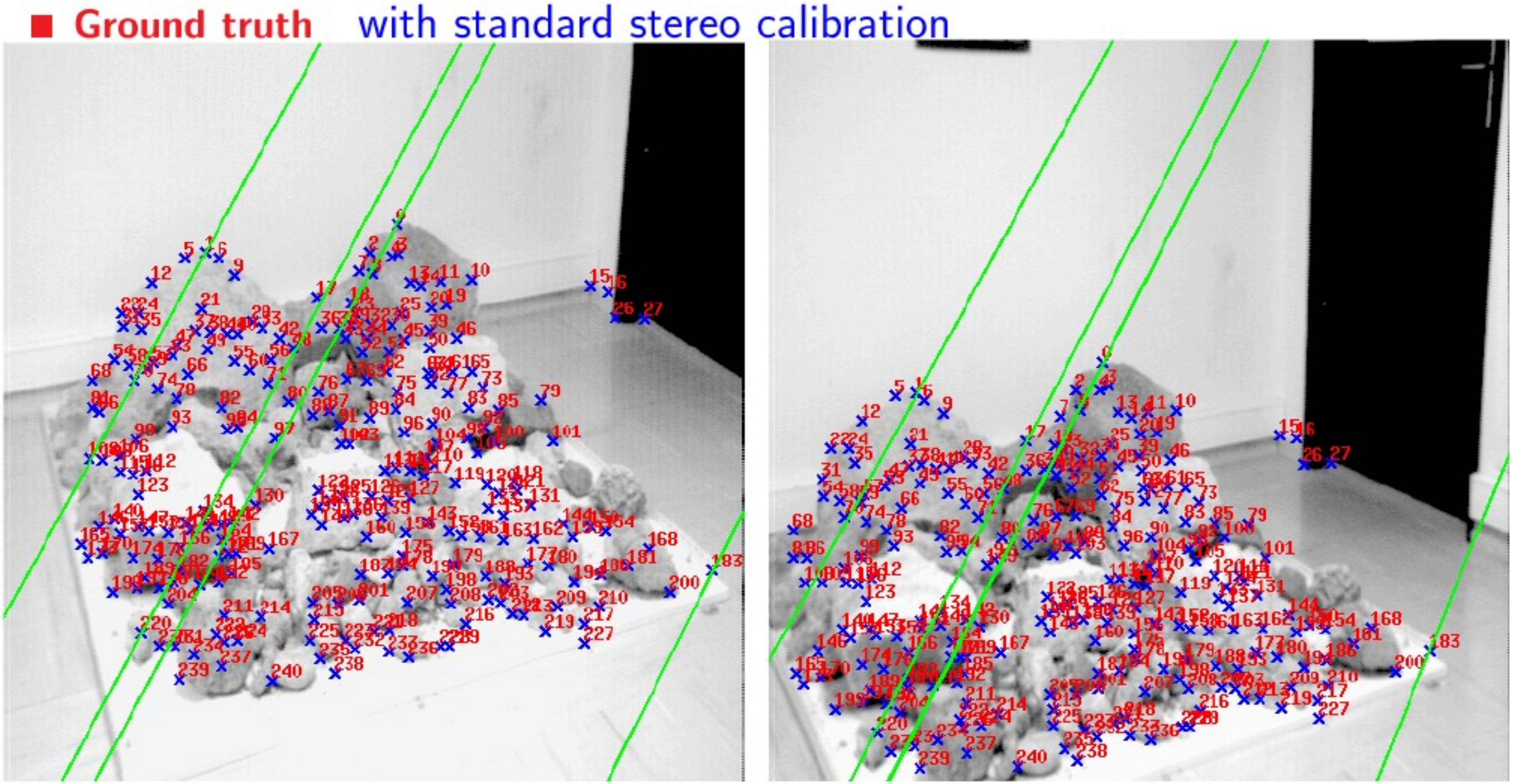
$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$$

$$\hat{\mathbf{x}}'^{\mathrm{T}}\underbrace{\mathbf{T}'^{-\mathrm{T}}\mathbf{F}\mathbf{T}^{-1}}_{\hat{\mathbf{F}}}\hat{\mathbf{x}} = 0$$

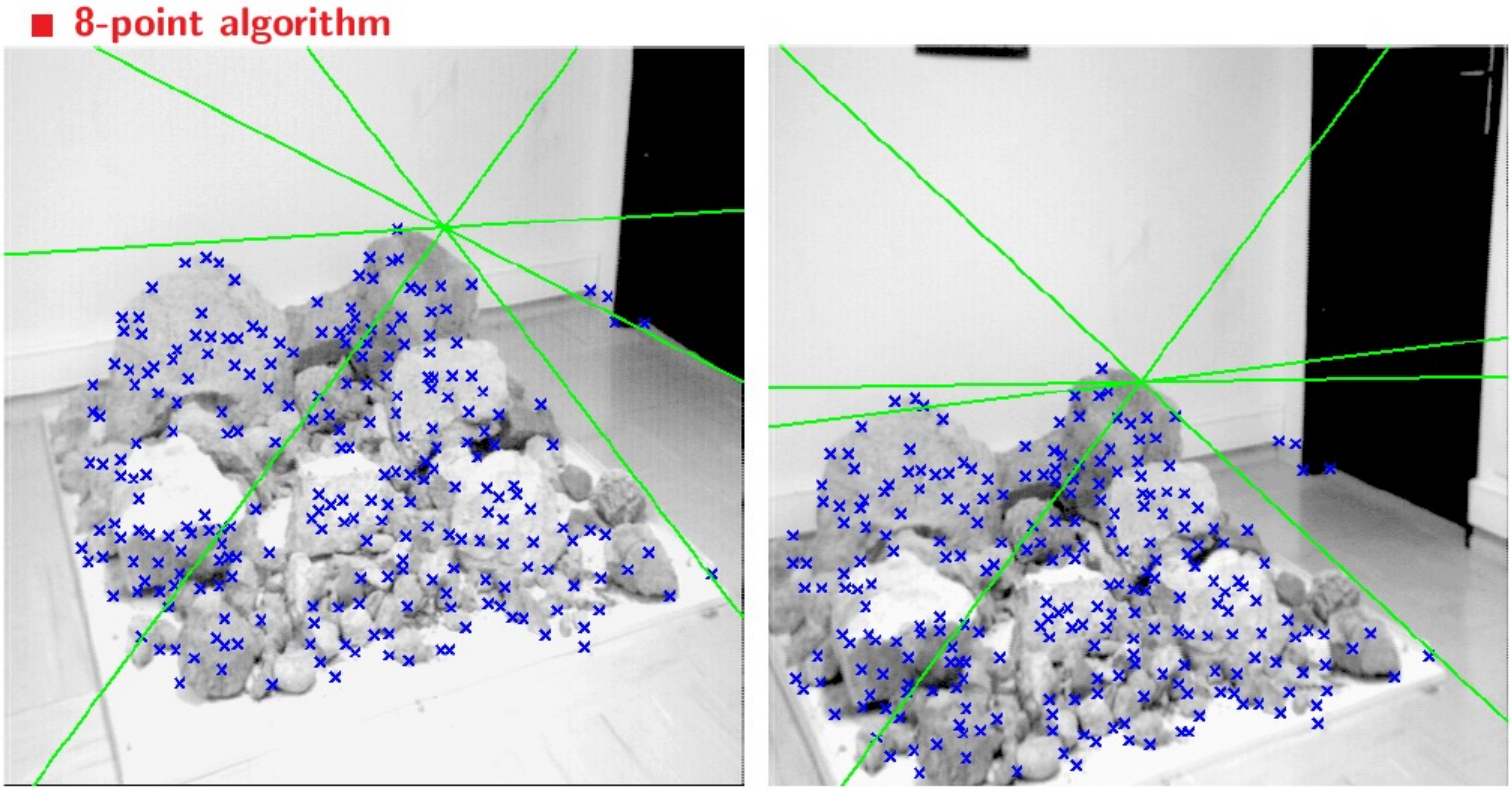
Normalized 8-point algorithm

```
[x1, T1] = normalise2dpts(x1);
[x2, T2] = normalise2dpts(x2);

A = [x2(1,:)'.*x1(1,:)'   x2(1,:)'.*x1(2,:)'  x2(1,:)' ...
     x2(2,:)'.*x1(1,:)'   x2(2,:)'.*x1(2,:)'  x2(2,:)' ...
     x1(1,:)'             x1(2,:)'            ones(npts,1) ];

[U,D,V] = svd(A);

F = reshape(V(:,9),3,3)';

[U,D,V] = svd(F);
F = U*diag([D(1,1) D(2,2) 0])*V';


% Denormalise
F = T2'*F*T1;
```

Results (ground truth)

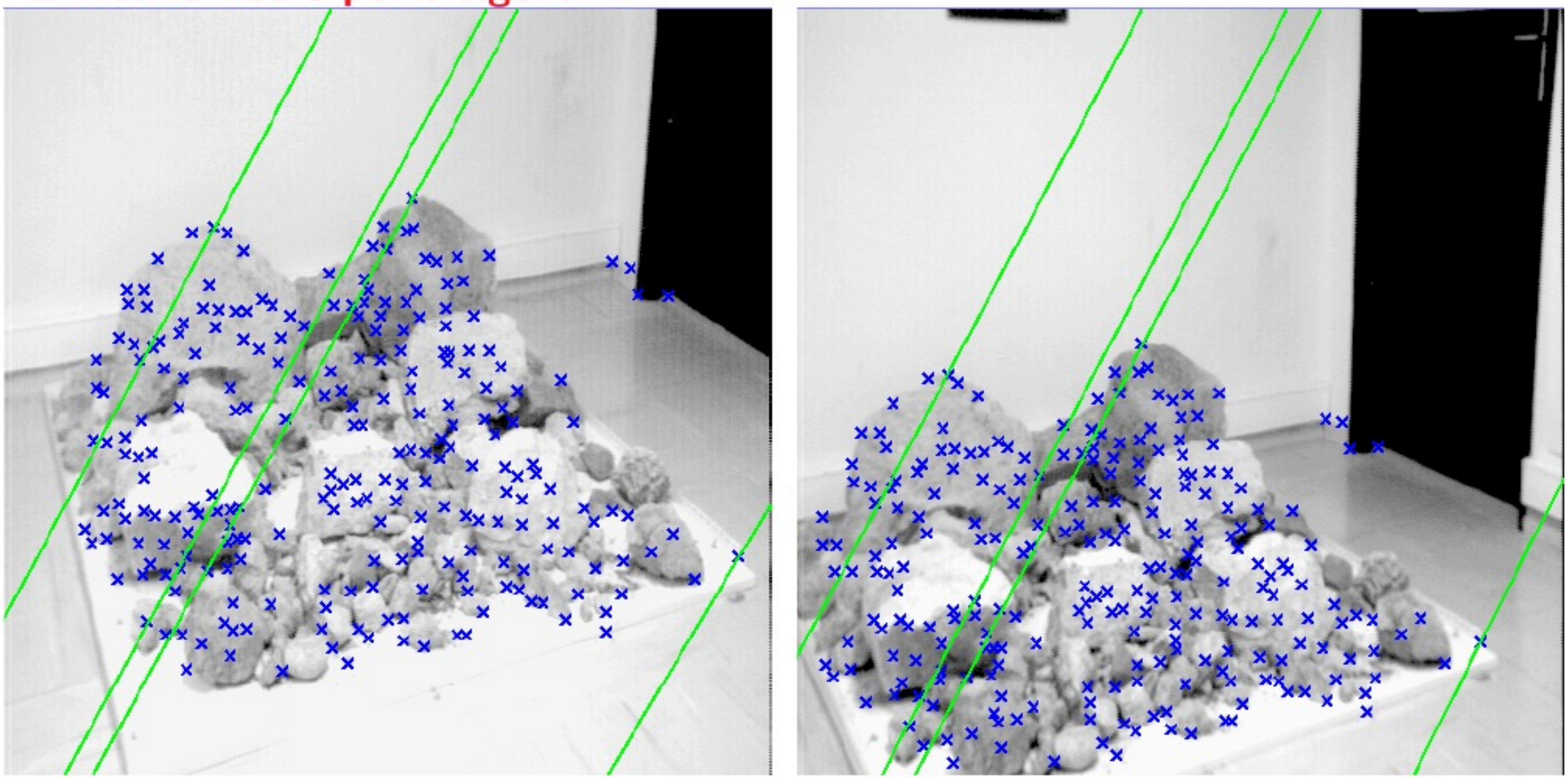


Results (8-point algorithm)



Results (normalized 8-point algorithm)

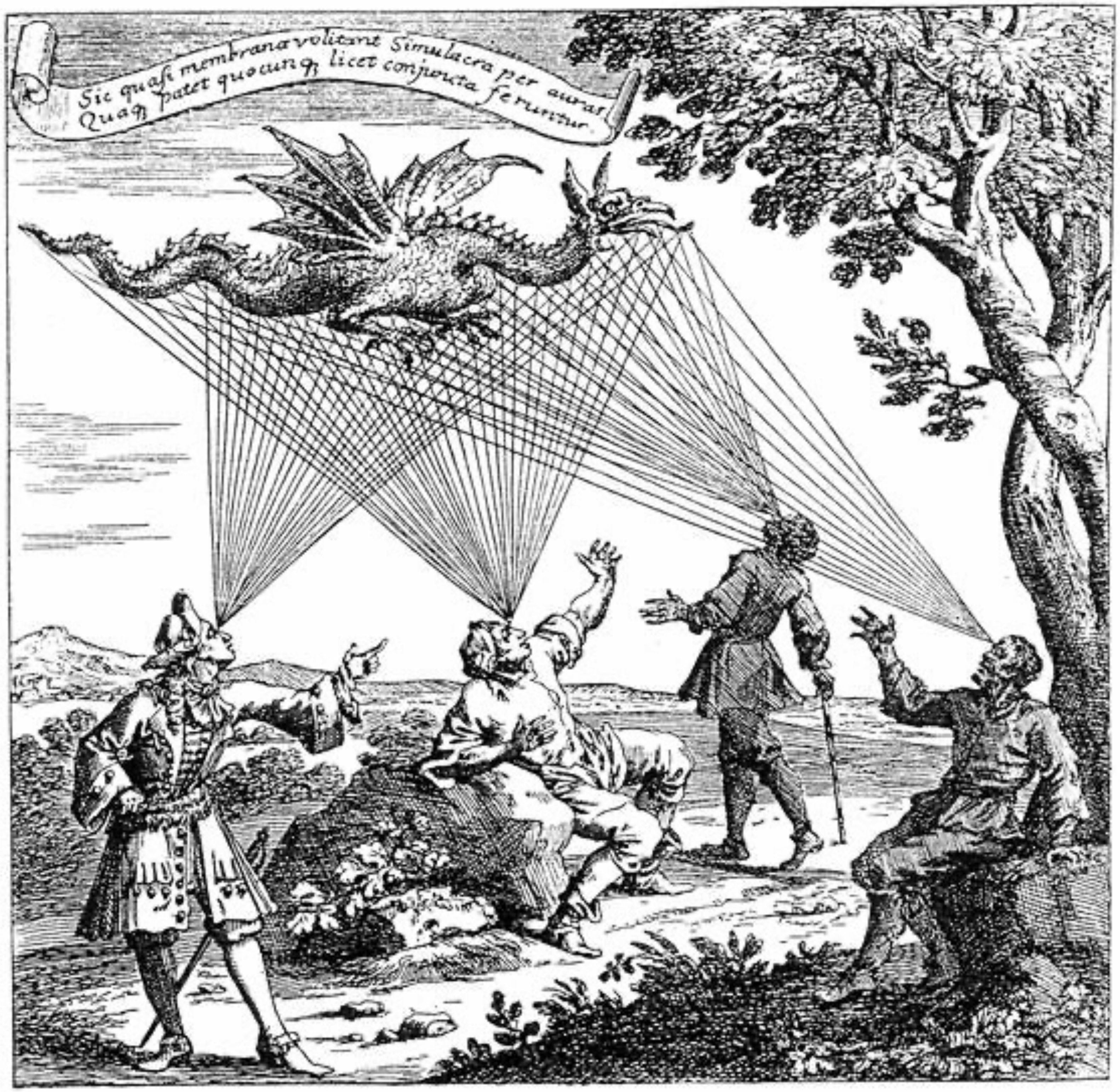
- Normalized 8-point algorithm



Two-view structure from motion

	Structure (scene geometry)	Motion (camera geometry)	Measurements
Pose Estimation	known	estimate	3D to 2D correspondences
Triangulation	estimate	known	2D to 2D coorespondences
Reconstruction	estimate	estimate	2D to 2D coorespondences

Structure from motion



Драконъ, видимый подъ различными углами зрѣнія

По гравюрѣ на мѣди изъ „Oculus artificialis teledioptricus" Цана. 1702 года.

Camera calibration & triangulation

- Suppose we know 3D points
    - And have matches between these points and an image
    - How can we compute the camera parameters?

- Suppose we have know camera parameters, each of which observes a point
    - How can we compute the 3D location of that point?

Structure from motion

- SfM solves both of these problems *at once*
- A kind of chicken-and-egg problem
  - (but solvable)

Reconstruction

(2 view structure from motion)

Given a set of matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

Estimate the camera matrices

$$\mathbf{P}, \mathbf{P}'$$

Estimate the 3D point

$$\mathbf{X}$$

Reconstruction

(2 view structure from motion)

Given a set of matched points

$$\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$$

Estimate the camera matrices

$$\mathbf{P}, \mathbf{P}'$$

'motion'

(of the cameras)

Estimate the 3D point

$$\mathbf{X}$$

'structure'

Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences

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

Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

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

Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

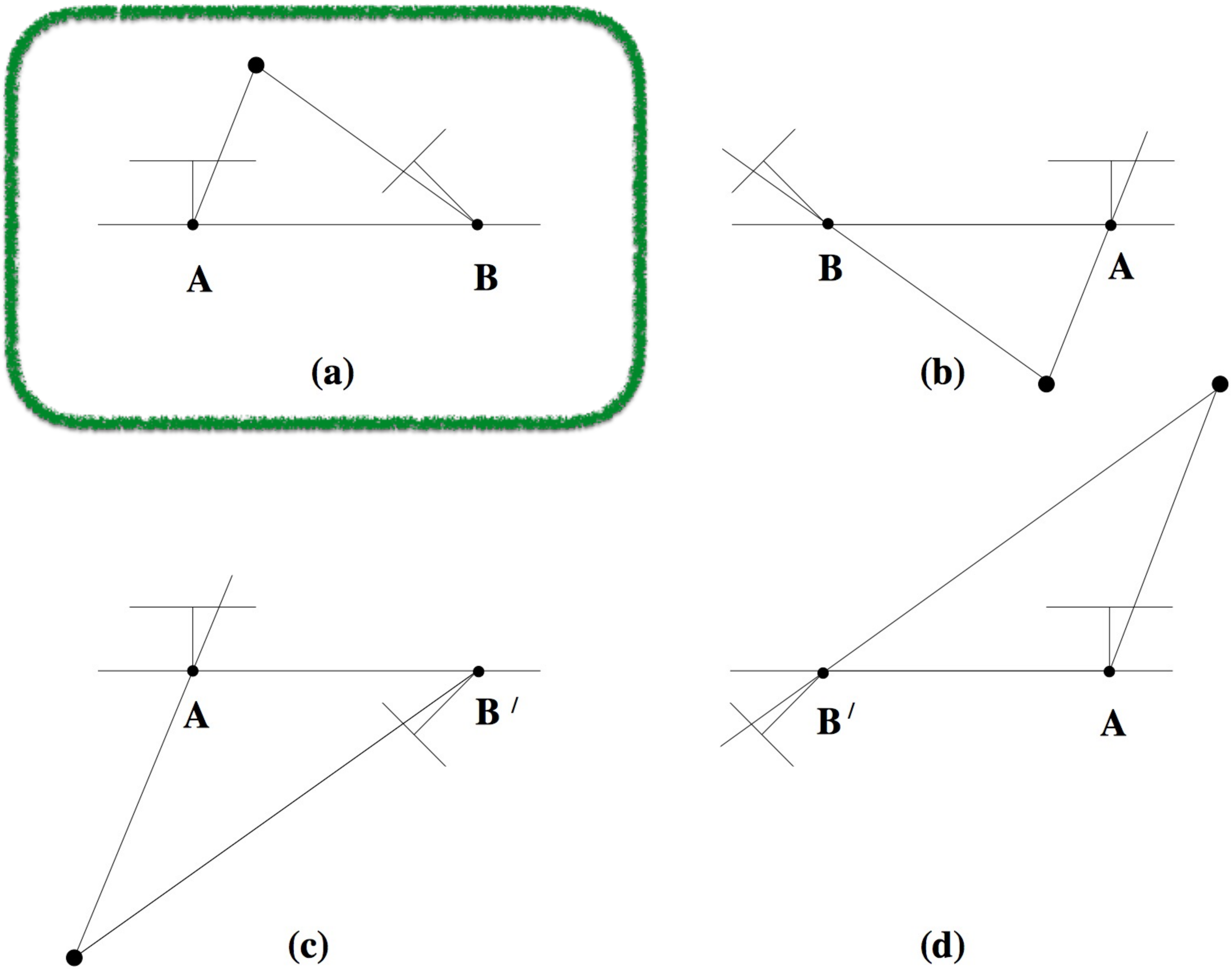
2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\ \mathbf{I} \mid \mathbf{0}\ ]$ and $\mathbf{P}' = [\ [\mathbf{e}_{x}]\mathbf{F} \mid \mathbf{e}'\ ]$

Camera matrices corresponding to the fundamental matrix **F** may be chosen as

$$\mathbf{P} = [\mathbf{I}|\mathbf{0}] \qquad \mathbf{P}' = [[\boldsymbol{e}_{\times}]\mathbf{F}|\boldsymbol{e}']$$

(See Hartley and Zisserman C.9 for proof)

Find the configuration where the points is in front of both cameras



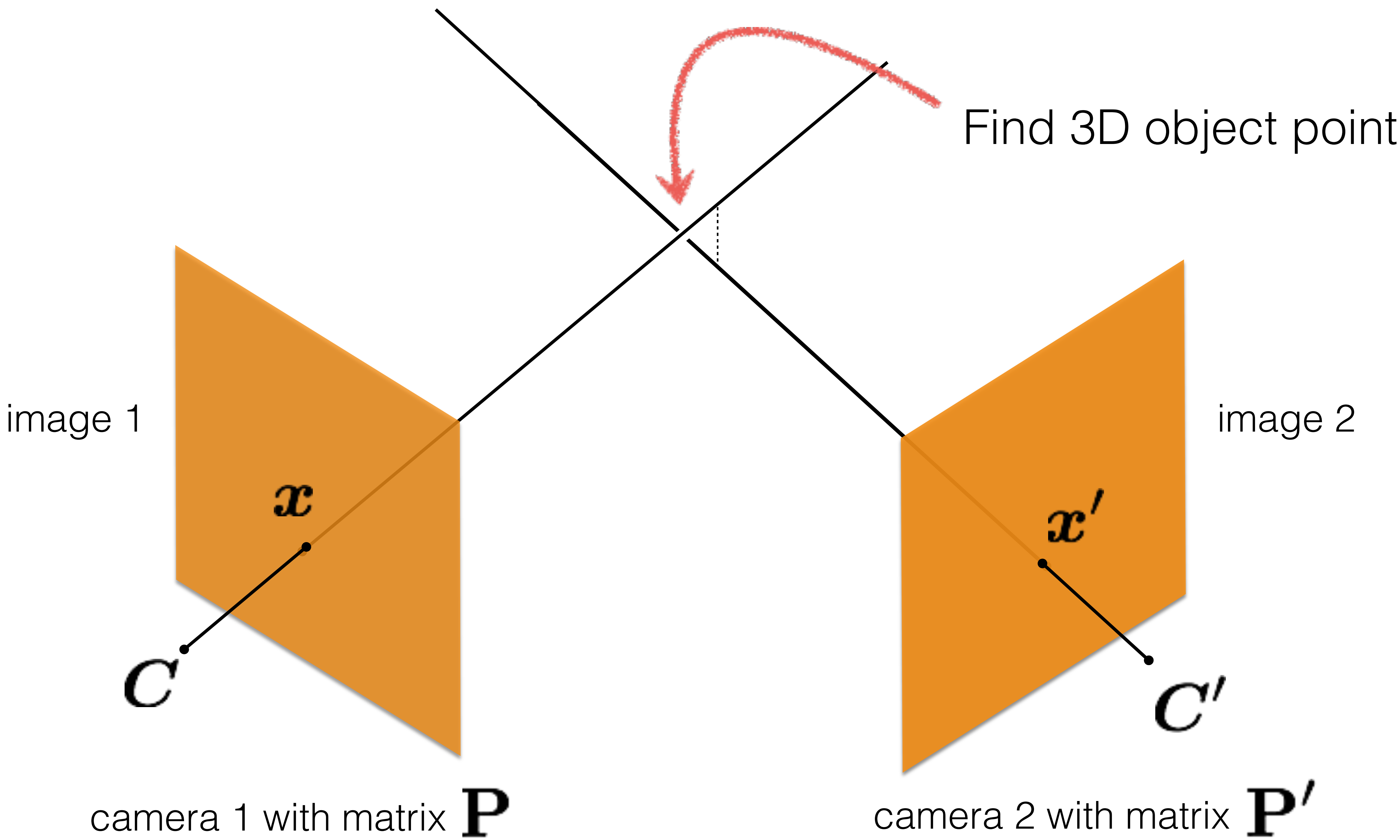
Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\ \mathbf{I}\ |\ \mathbf{0}\ ]$ and $\mathbf{P}' = [\ [\mathbf{e}'_{\times}]\mathbf{F}\ |\ \mathbf{e}'\ ]$

3. For each point correspondence, compute the point **X** in 3D space (triangulation)
   Triangulation with $x = \mathbf{P}\,X$ and $x' = \mathbf{P}'\,X$

Triangulation



Two-view SfM

1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm

2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\ \mathbf{I} \mid \mathbf{0}\ ]$ and $\mathbf{P'} = [\ [\mathbf{e'}_{\times}]\mathbf{F} \mid \mathbf{e'}\ ]$

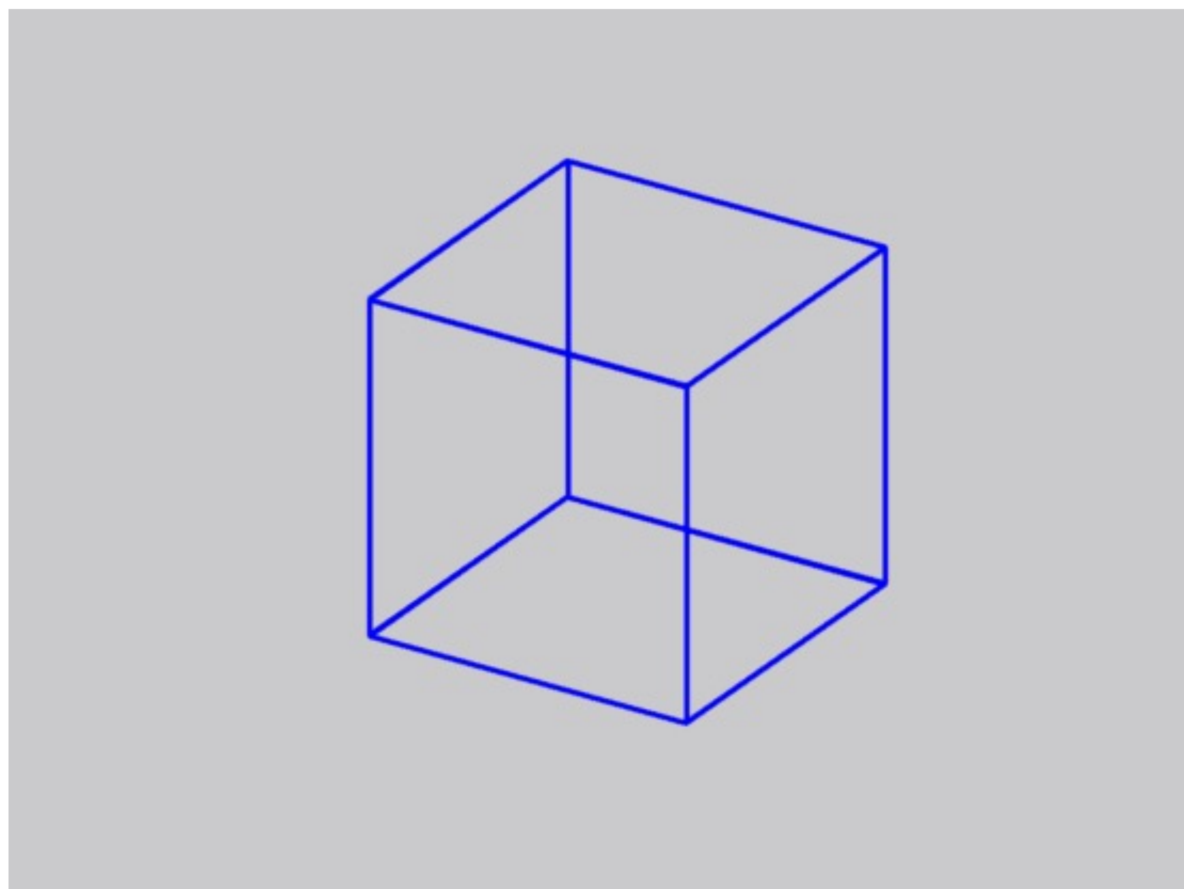
3. For each point correspondence, compute the point **X** in 3D space (triangularization)
   DLT with $x = \mathbf{P}\,X$ and $x' = \mathbf{P'}\,X$

Is SfM always uniquely solvable?

Ambiguities in structure from motion

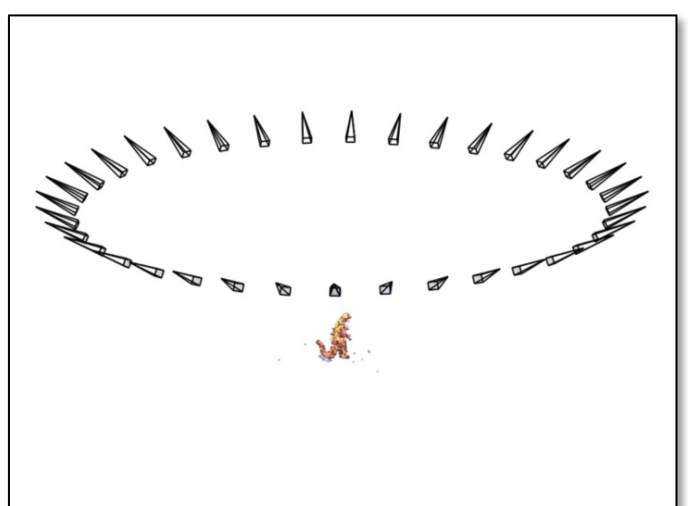
Is SfM always uniquely solvable?

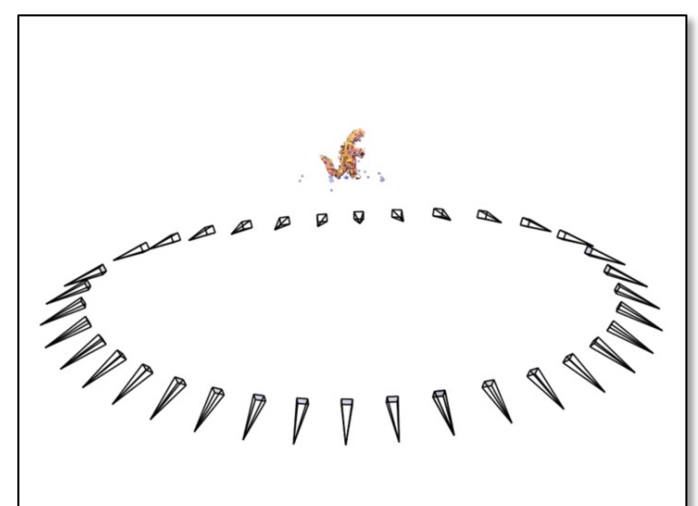
- No…



SfM – Failure cases

- Necker reversal





Projective Ambiguity

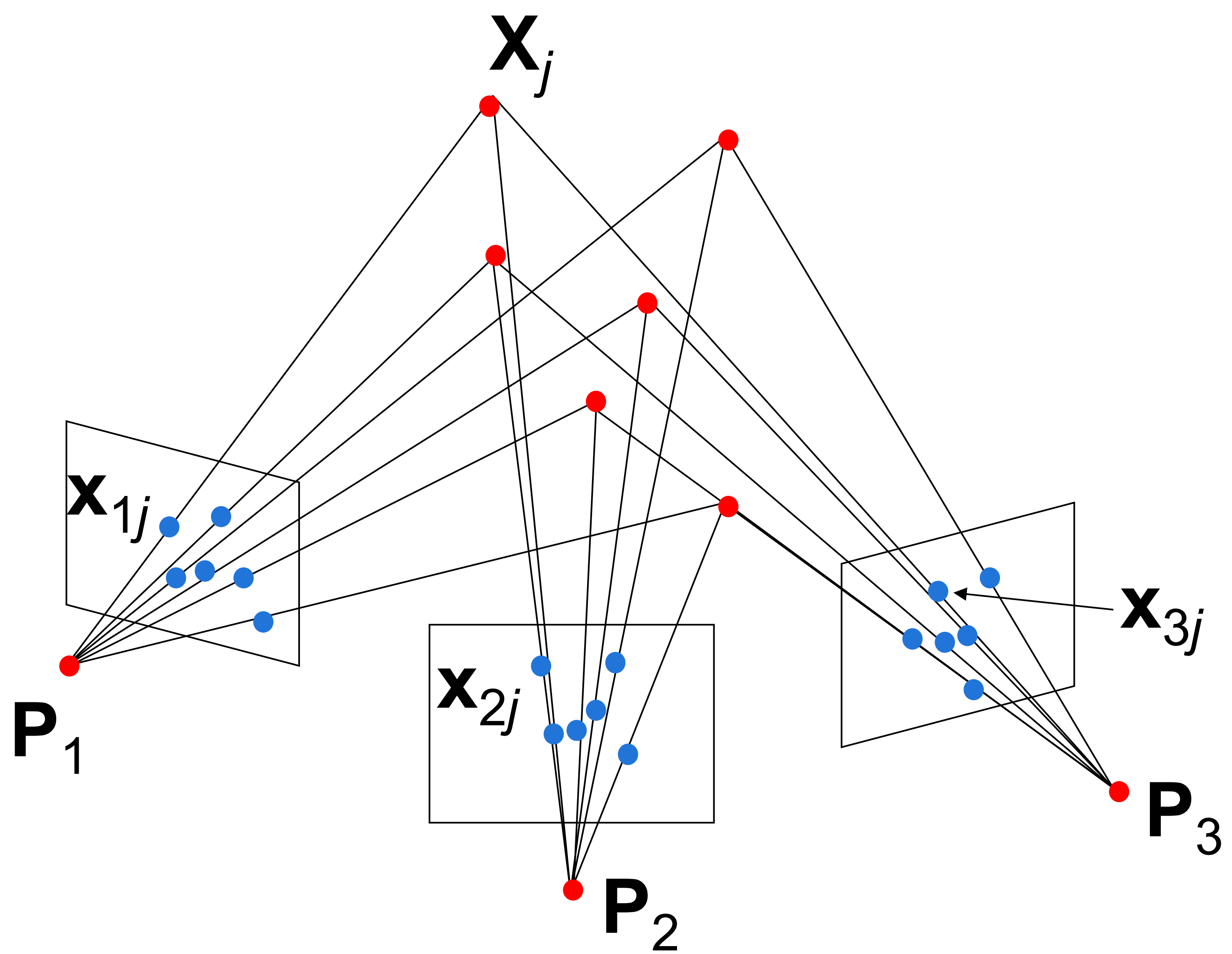
- Reconstruction is ambiguous by an arbitrary 3D projective transformation without prior knowledge of camera parameters

Structure from motion

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

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

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



Structure from motion ambiguity

- If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same:

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\frac{1}{k}\mathbf{P}\right)(k\mathbf{X})$$

It is impossible to recover the absolute scale of the scene!

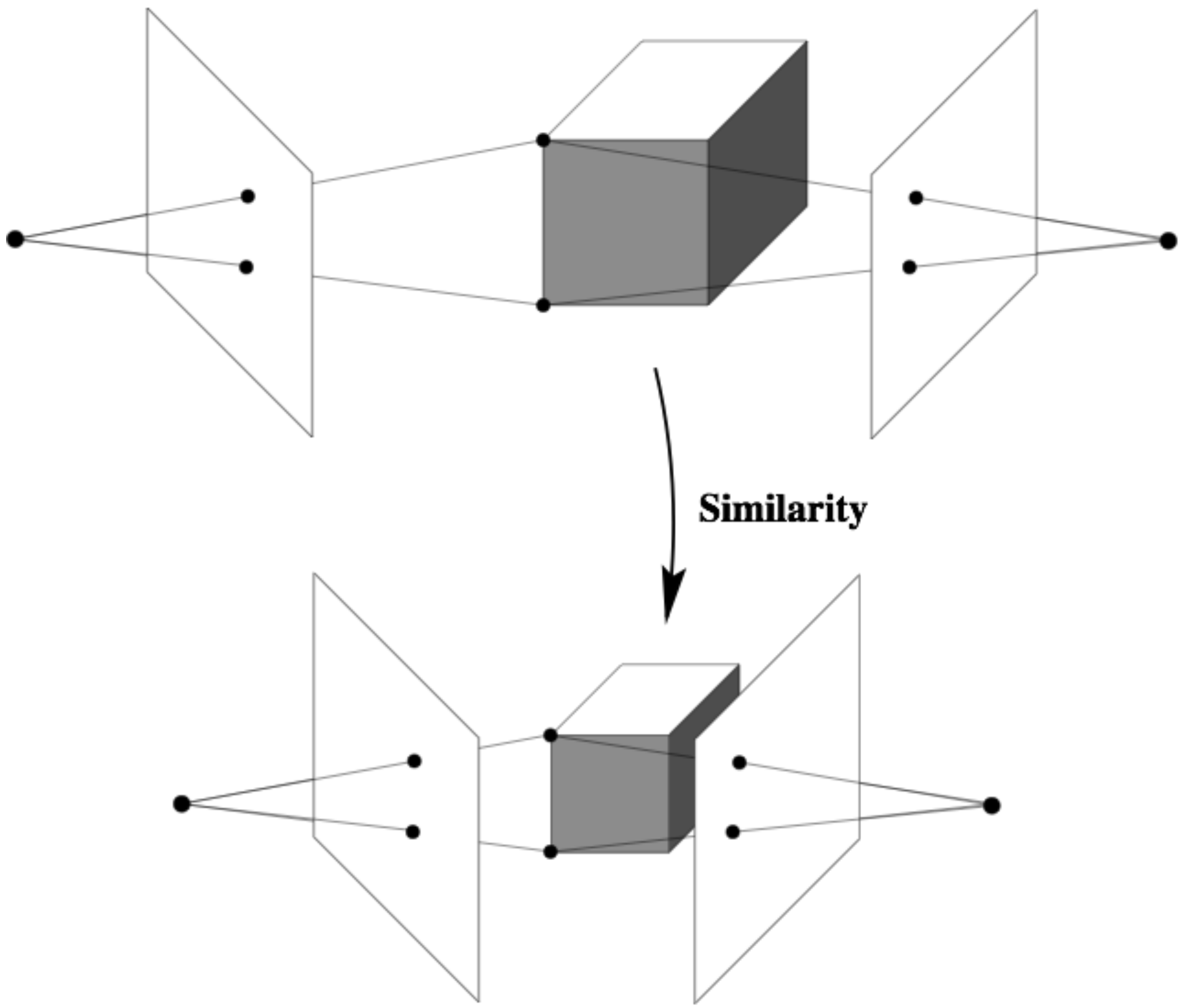
Structure from motion ambiguity

- If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same

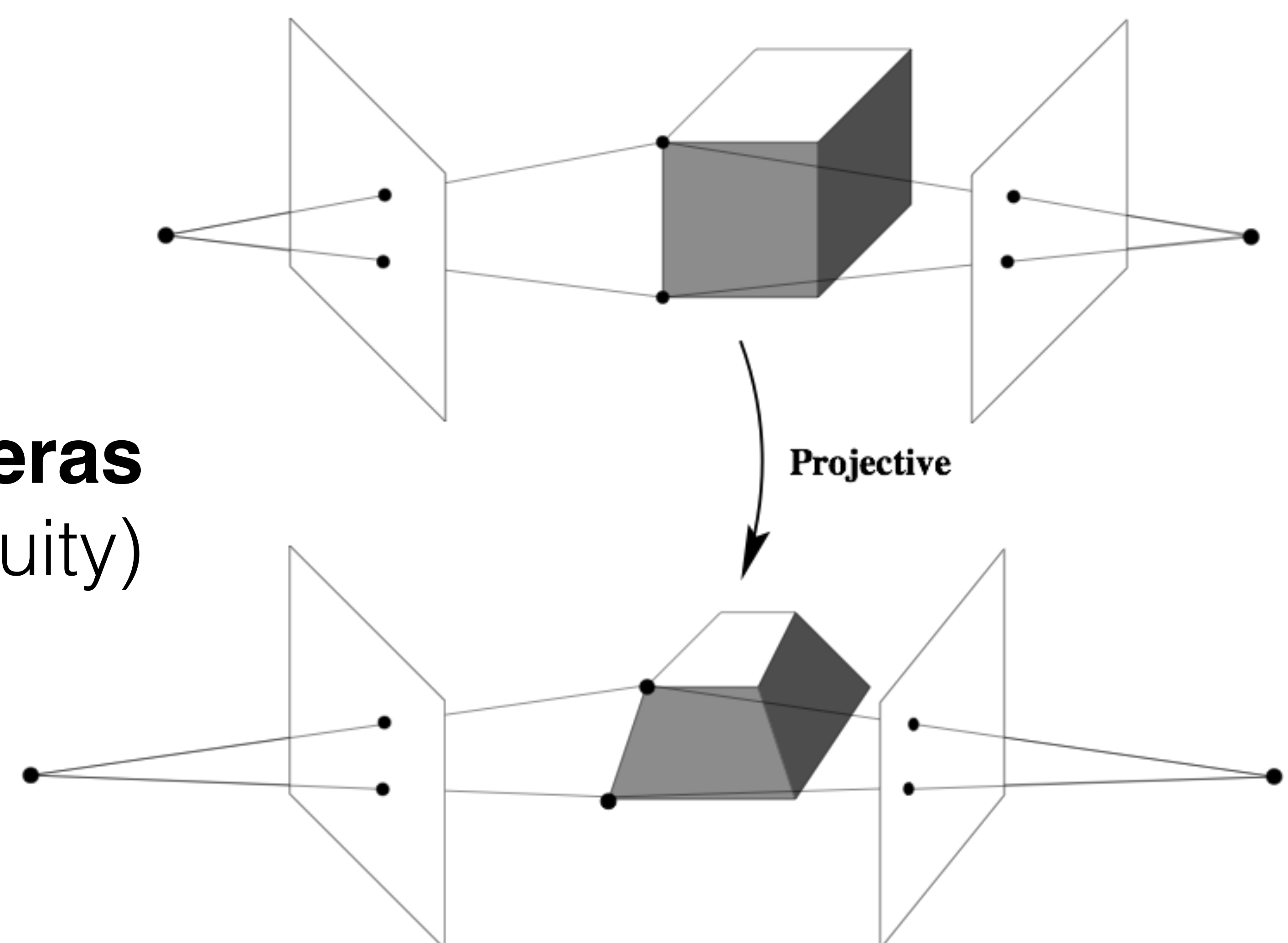
- More generally: if we transform the scene using a transformation **Q** and apply the inverse transformation to the camera matrices, then the images do not change

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)\left(\mathbf{Q}\mathbf{X}\right)$$

Calibrated cameras
(similarity projection ambiguity)



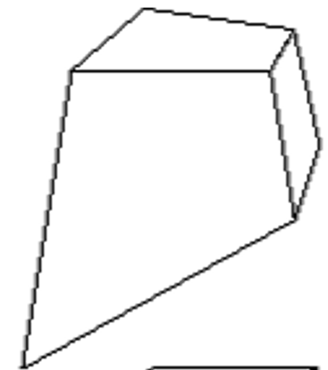
Uncalibrated cameras
(projective projection ambiguity)



Types of ambiguity

Projective
15dof

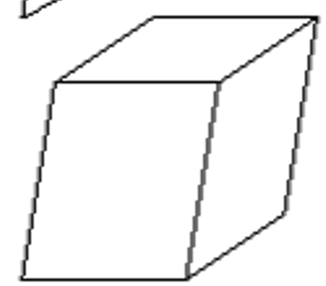
$$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$$



Preserves intersection and tangency

Affine
12dof

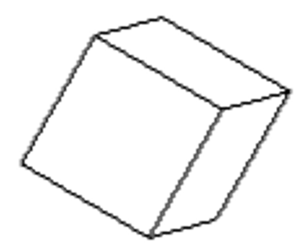
$$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$$



Preserves parallellism, volume ratios

Similarity
7dof

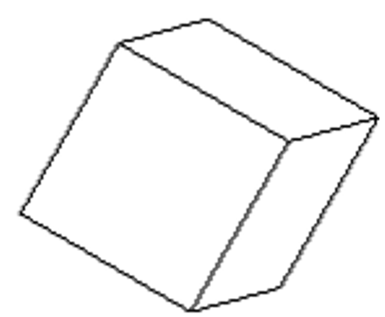
$$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$$



Preserves angles, ratios of length

Euclidean
6dof

$$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$

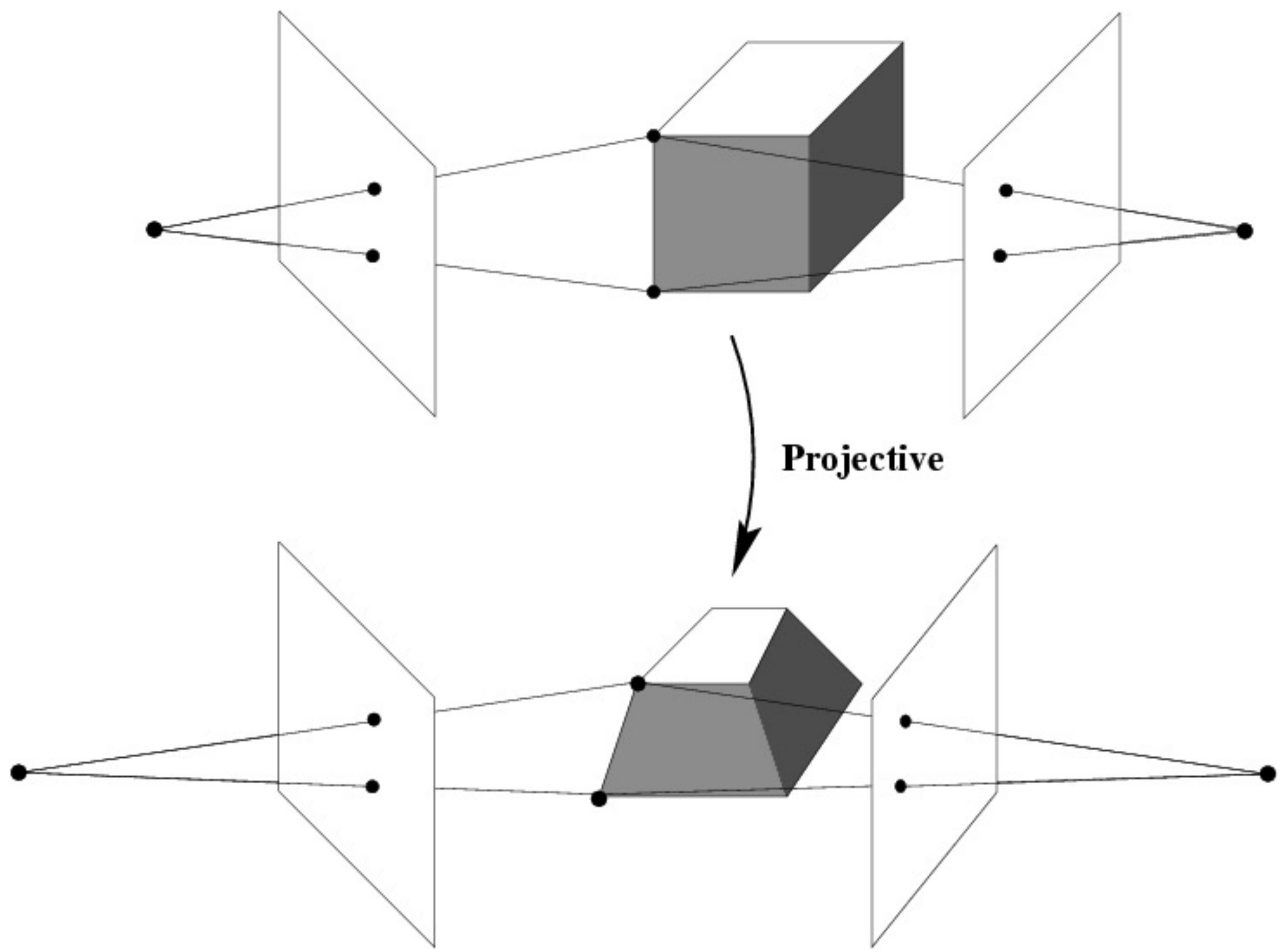


Preserves angles, lengths

- With no constraints on the camera calibration matrix or on the scene, we get a *projective* reconstruction
- Need additional information to *upgrade* the reconstruction to affine, similarity, or Euclidean

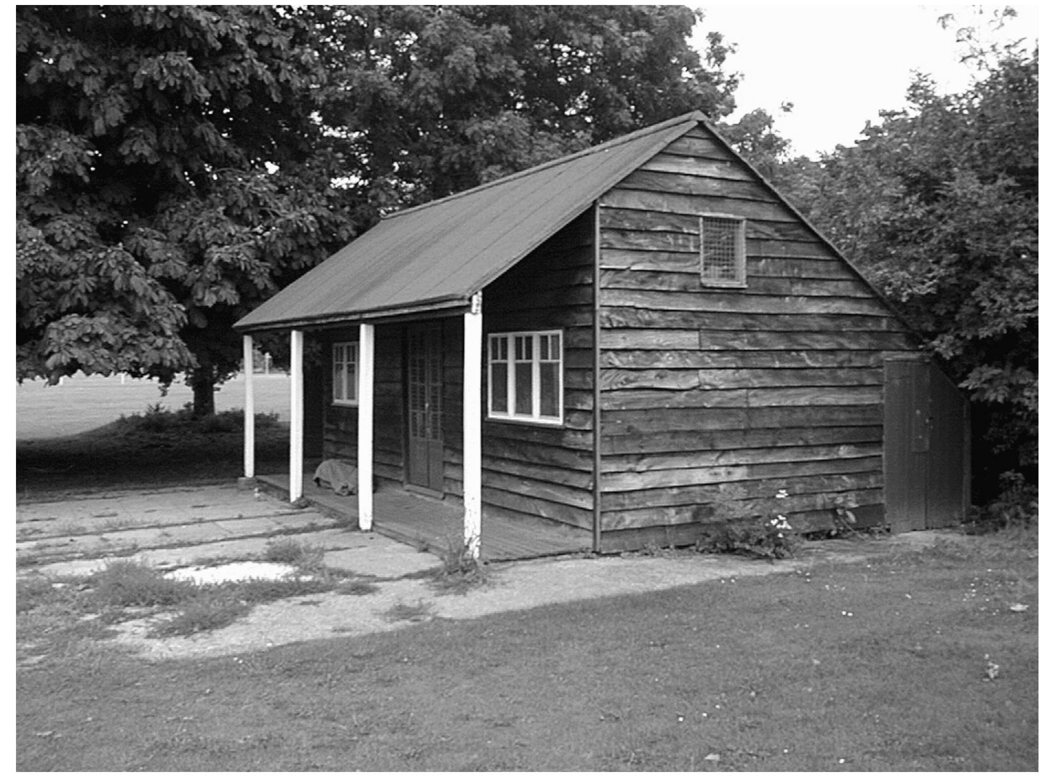
Slide: S. Lazebnik

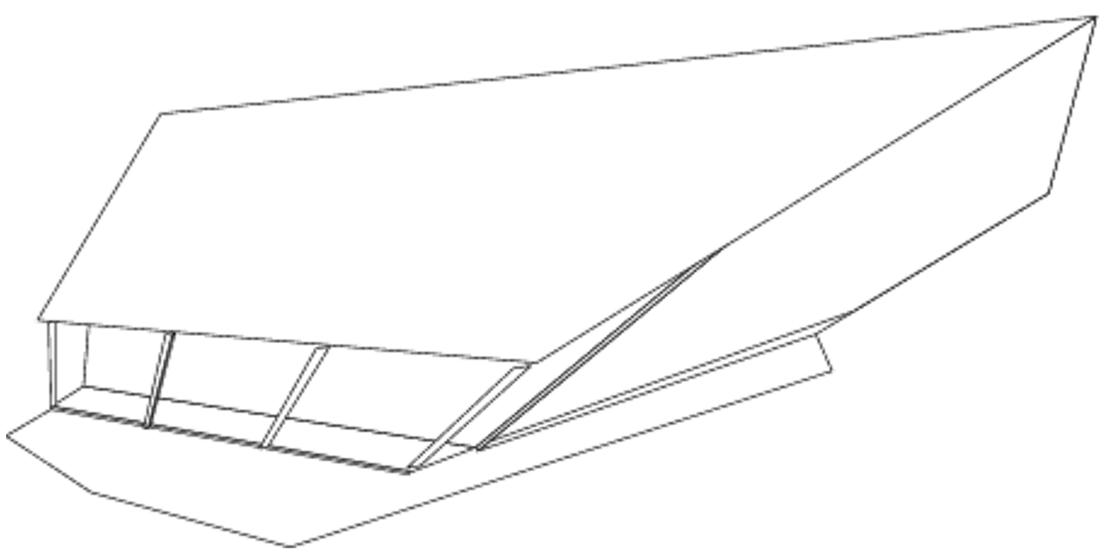
Projective ambiguity

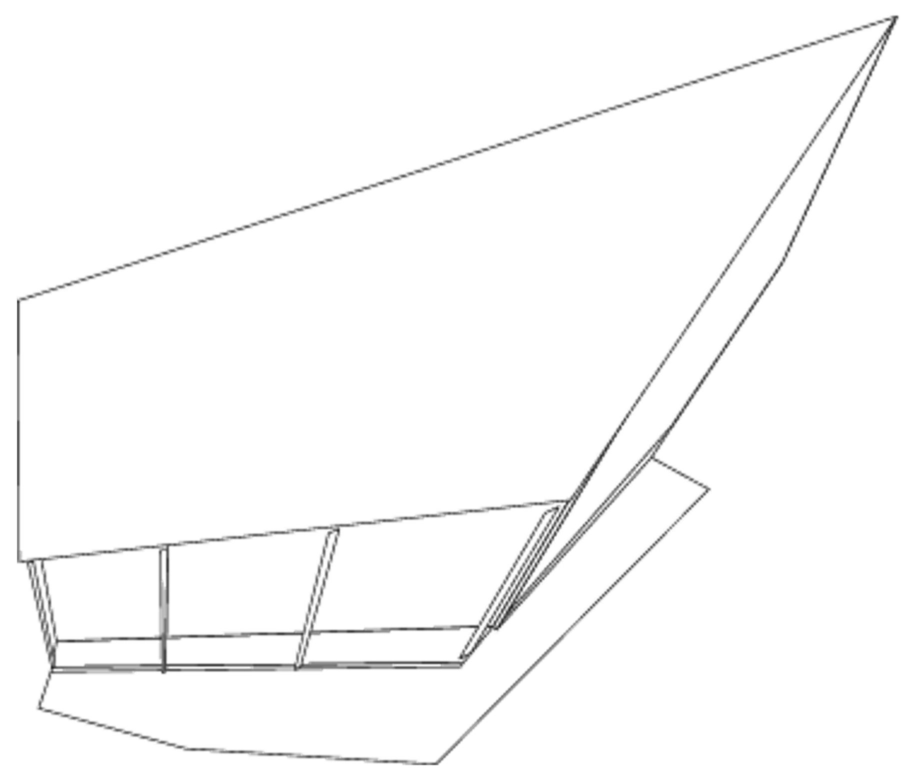


$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_{\mathbf{P}}^{-1}\right)\left(\mathbf{Q}_{\mathbf{P}}\mathbf{X}\right)$$

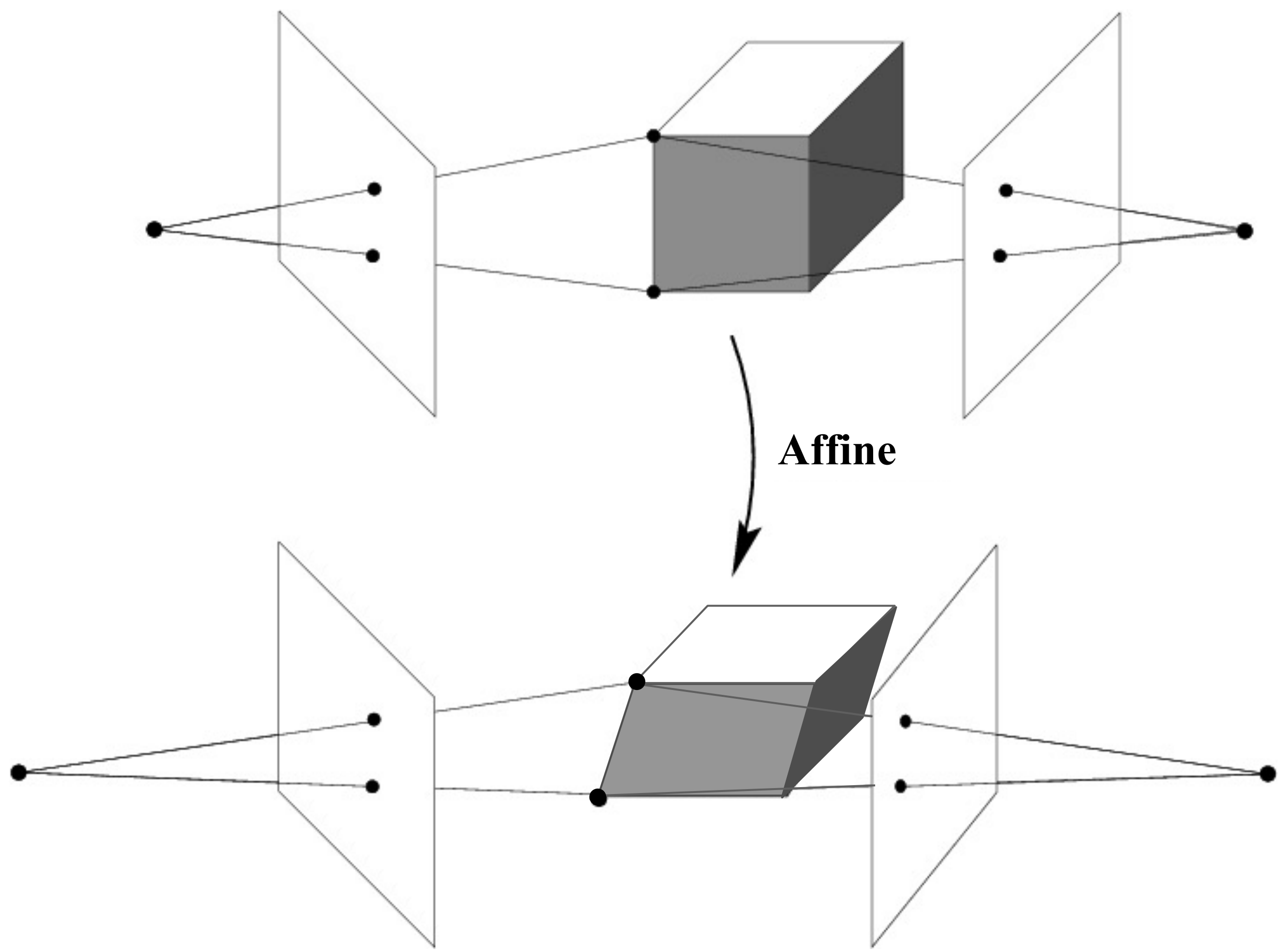
Projective ambiguity





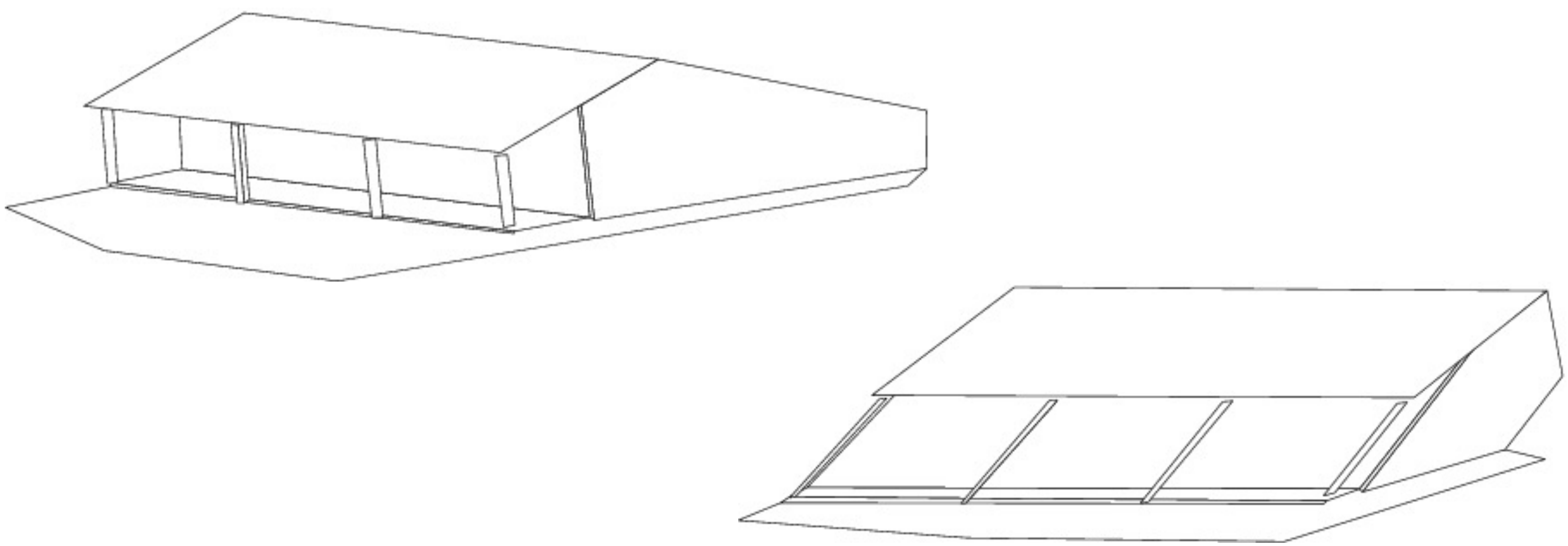


Affine ambiguity

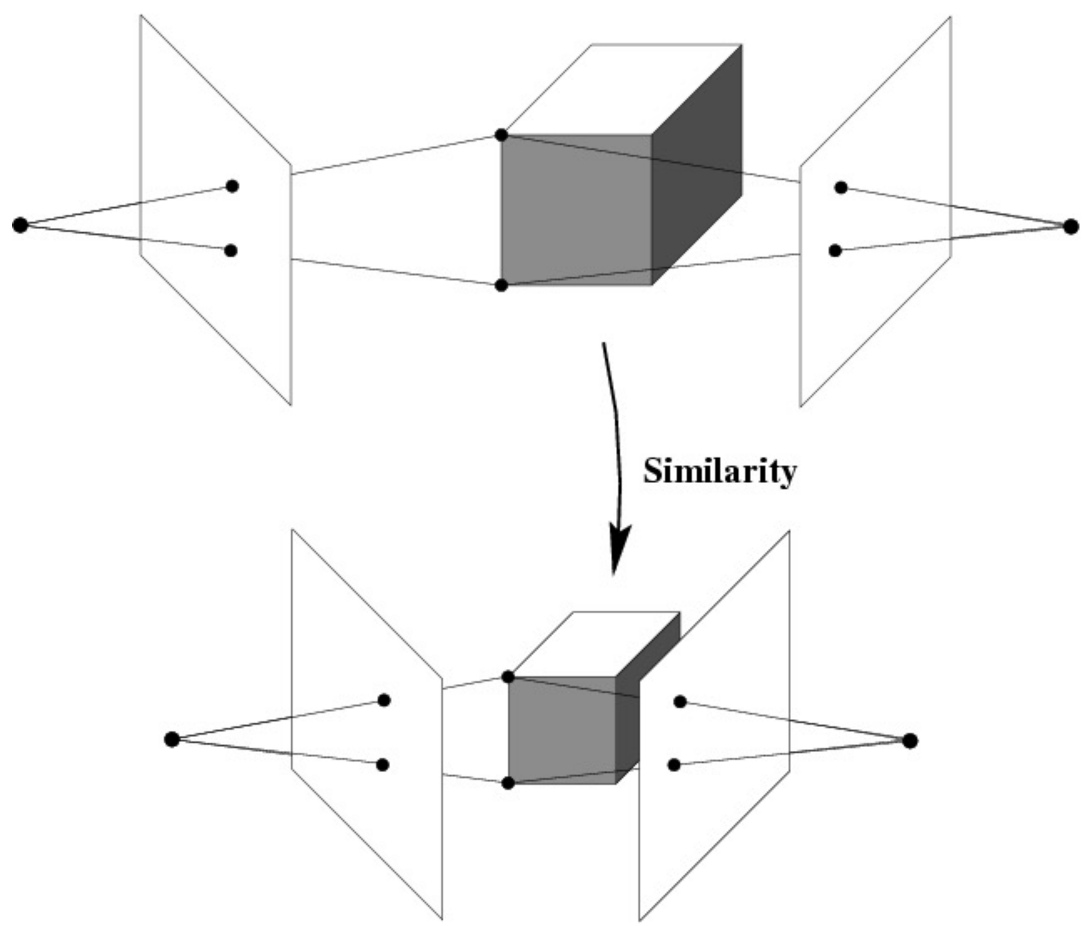


$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_{\mathbf{A}}^{-1}\right)\left(\mathbf{Q}_{\mathbf{A}}\,\mathbf{X}\right)$$

Affine ambiguity

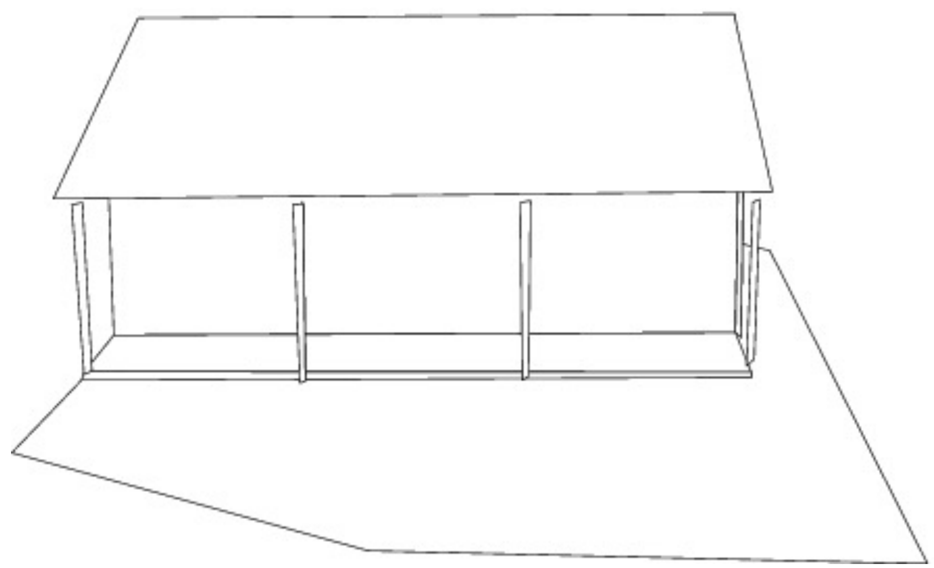


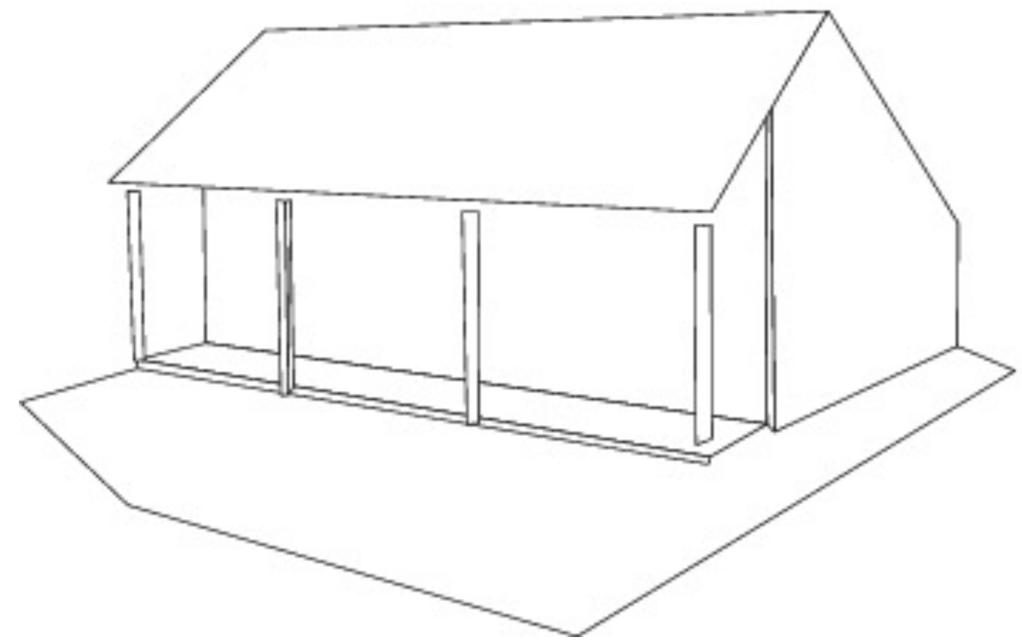
Similarity ambiguity

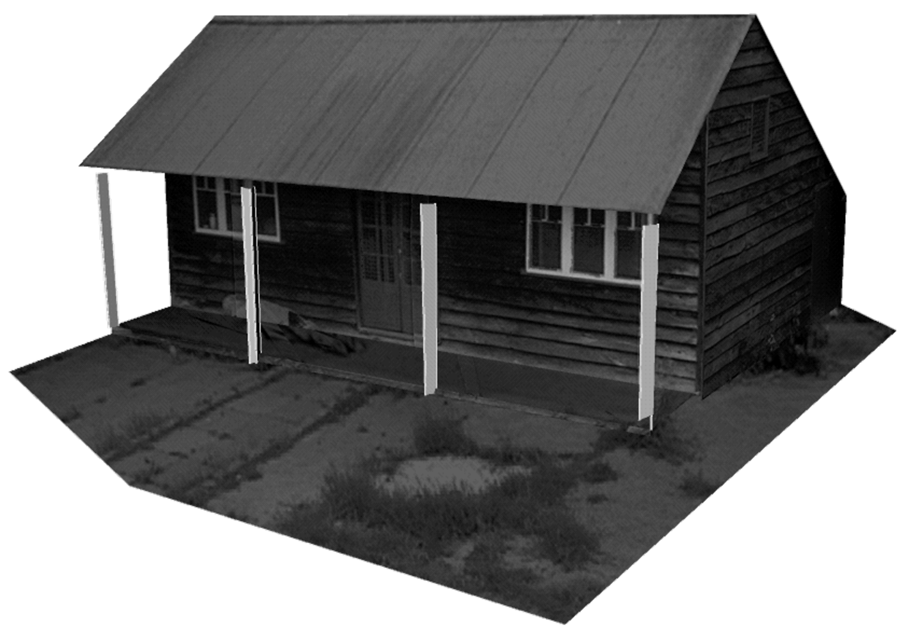


$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_S^{-1}\right)\left(\mathbf{Q}_S\mathbf{X}\right)$$

Similarity ambiguity





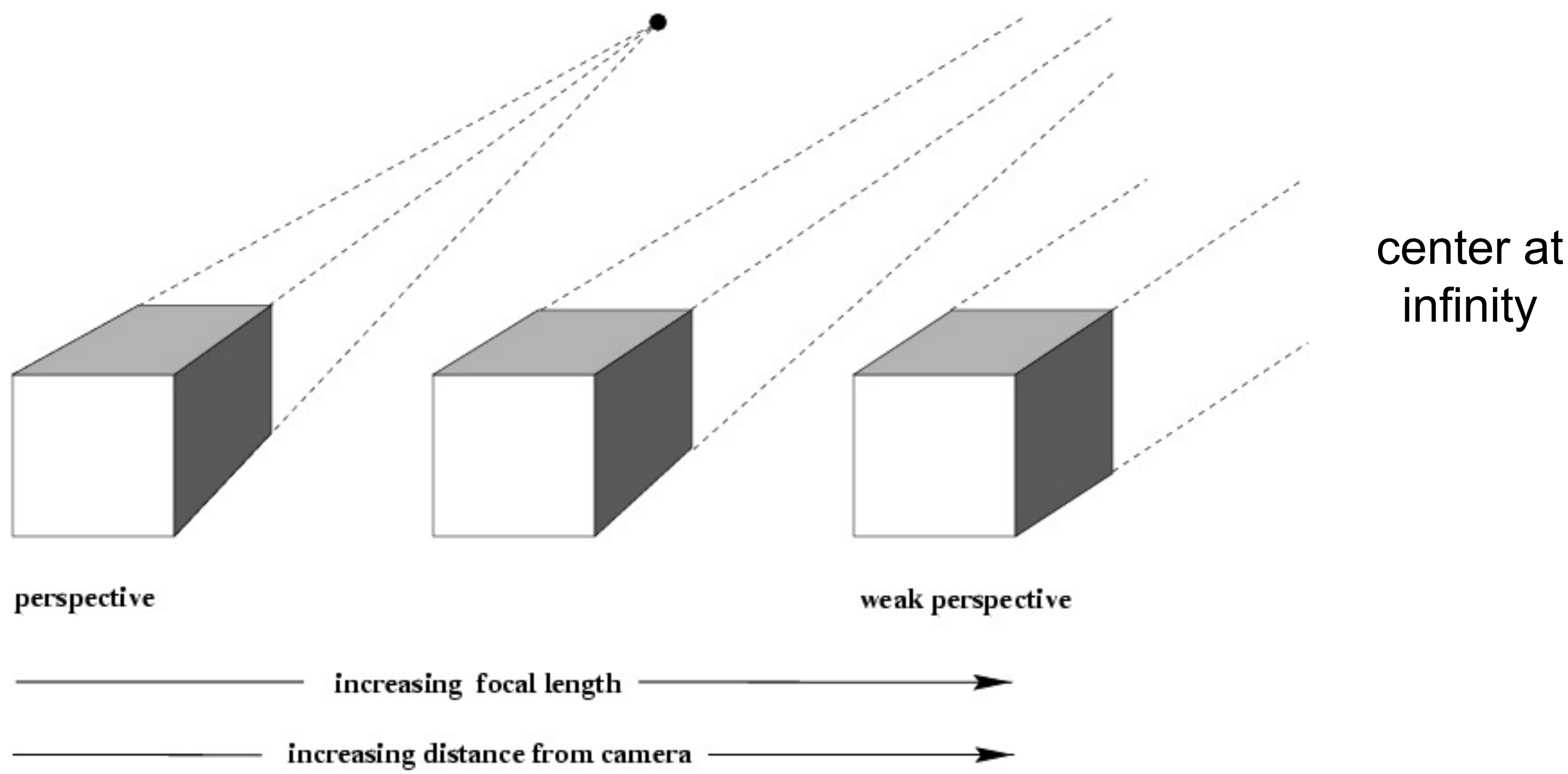


What can we do to remove ambiguities?

Affine structure from motion

Structure from motion

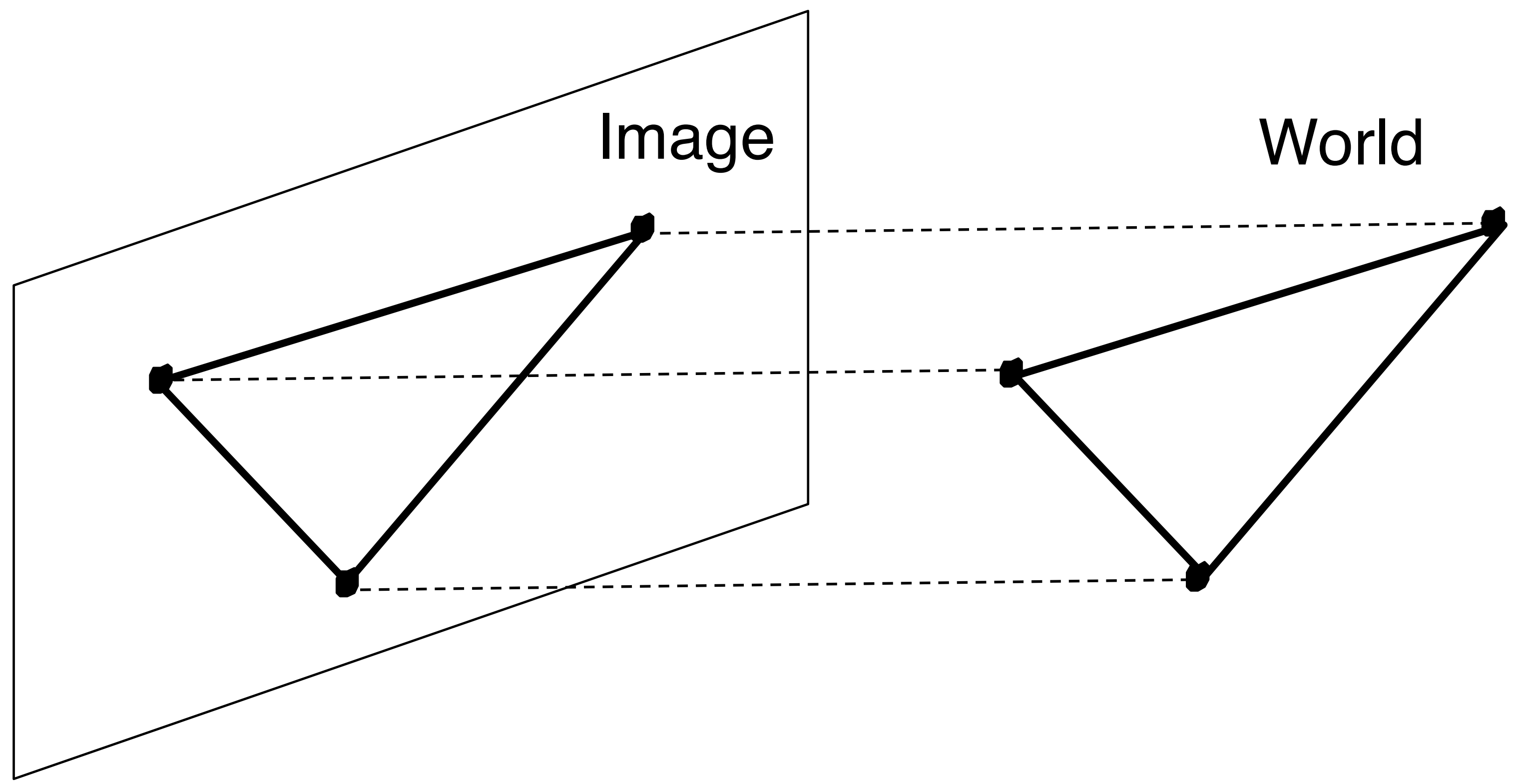
- Let's start with *affine cameras* (the math is easier)



Recall: Orthographic Projection

Special case of perspective projection

- Distance from center of projection to image plane is infinite

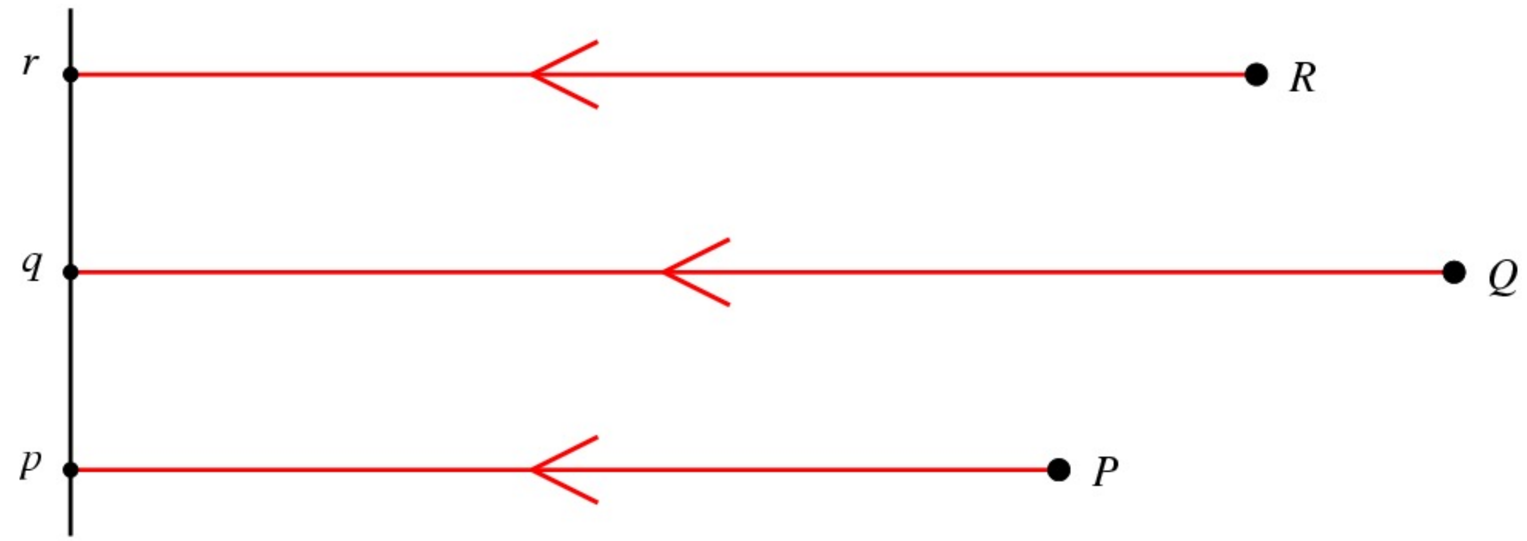


- Projection matrix:

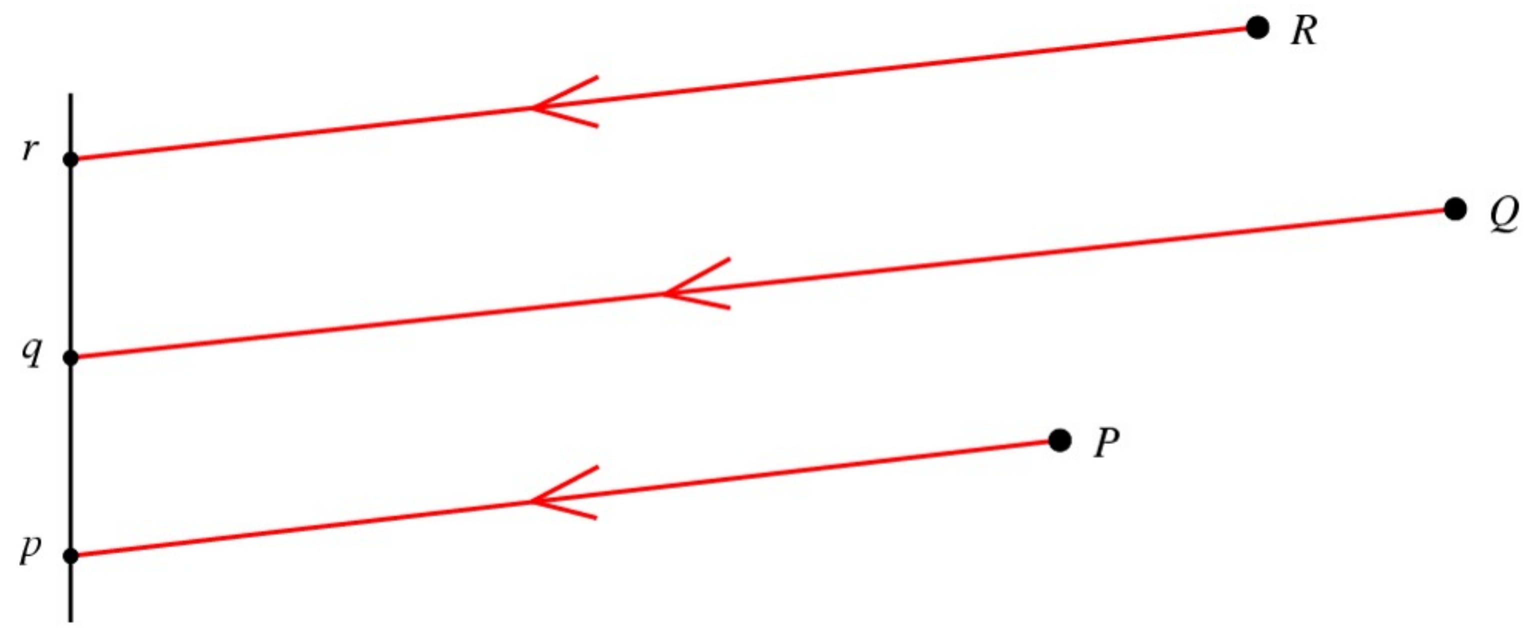
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

Affine cameras

Orthographic Projection



Parallel Projection

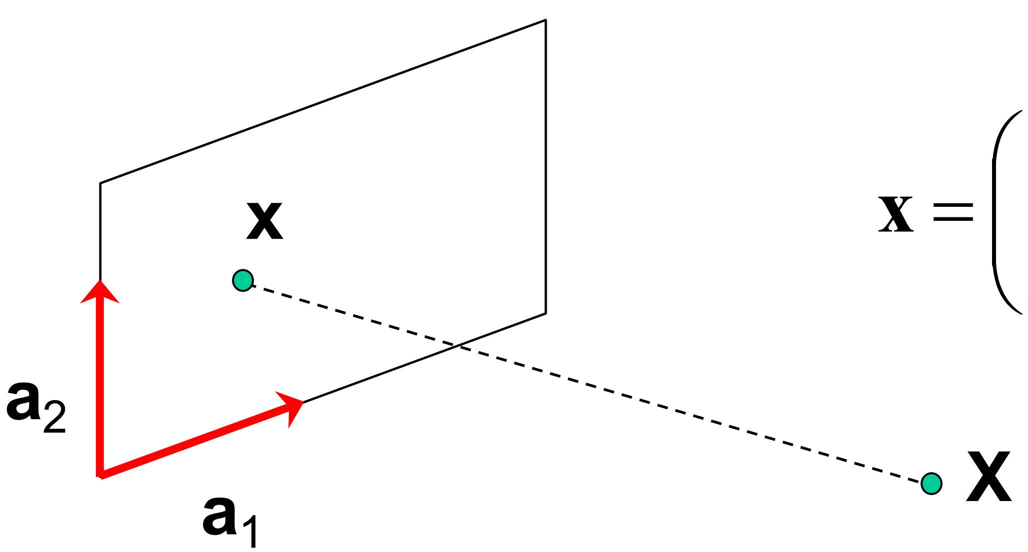


Affine cameras

- A general affine camera combines the effects of an affine transformation of the 3D space, orthographic projection, and an affine transformation of the image:

$$\mathbf{P} = [3\times 3\,\text{affine}]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}[4\times 4\,\text{affine}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

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



$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \mathbf{AX} + \mathbf{b}$$

Projection of world origin

Affine structure from motion

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

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

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \rightarrow \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$$

- We have 2*mn* knowns and 8*m* + 3*n* unknowns (minus 12 dof for affine ambiguity)
- Thus, we must have 2*mn* >= 8*m* + 3*n* – 12
- For two views, we need four point correspondences

Affine structure from motion

- Centering: subtract the centroid of the image points

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

$$= \mathbf{A}_i\left(\mathbf{X}_j - \frac{1}{n}\sum_{k=1}^{n}\mathbf{X}_k\right) = \mathbf{A}_i\hat{\mathbf{X}}_j$$

- For simplicity, assume that the origin of the world coordinate system is at the centroid of the 3D points
- After centering, each normalized point $\mathbf{x}_{ij}$ is related to the 3D point $\mathbf{X}_i$ by

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

Affine structure from motion

- Let's create a $2m \times n$ data (measurement) matrix:

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix}$$

cameras ($2m$)

points (n)

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

Affine structure from motion

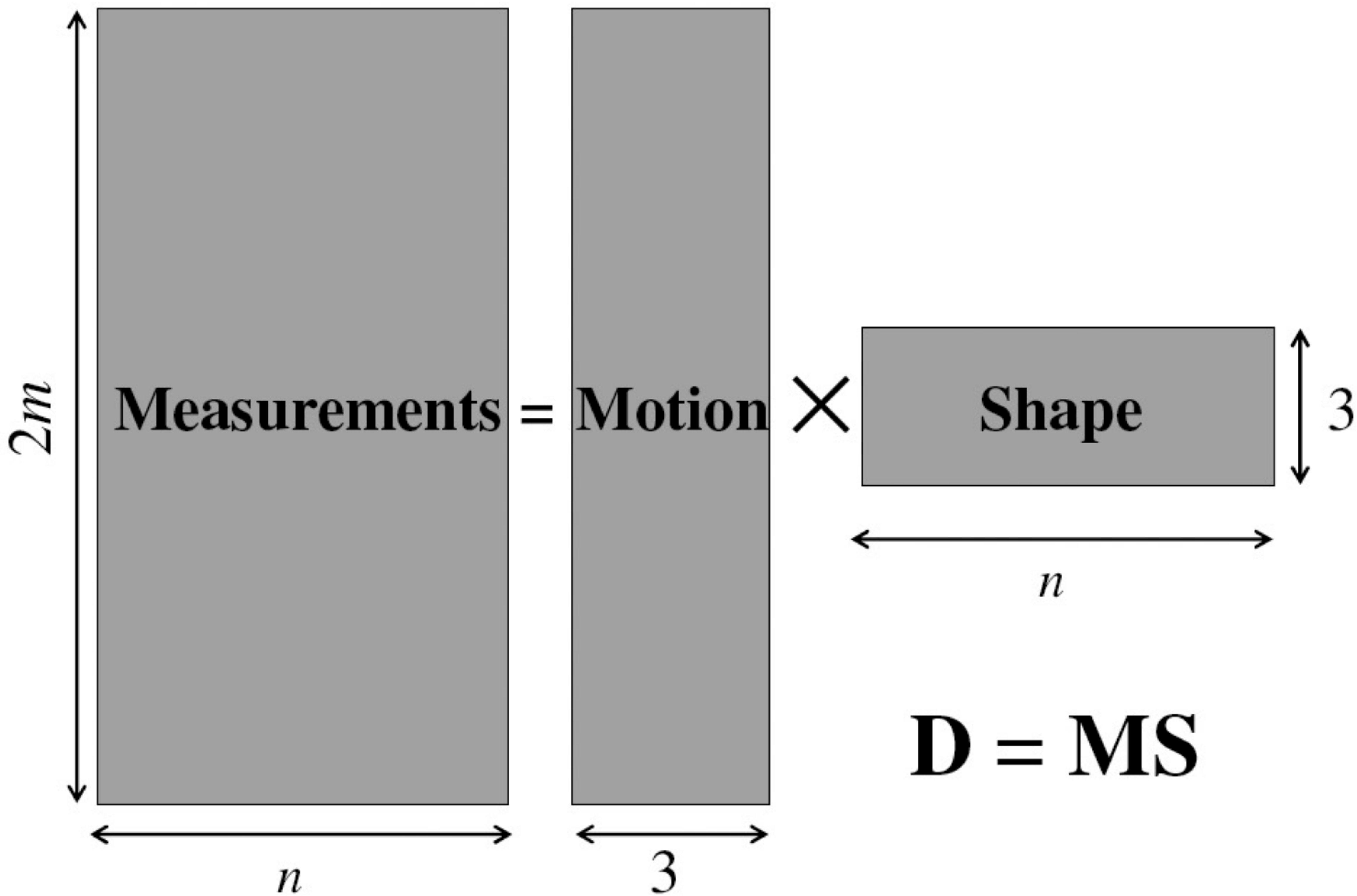
- Let's create a $2m \times n$ data (measurement) matrix:

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}}_{\text{cameras } (2m \times 3)} \underbrace{\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}}_{\text{points } (3 \times n)}$$

The measurement matrix $\mathbf{D} = \mathbf{MS}$ must have rank 3!

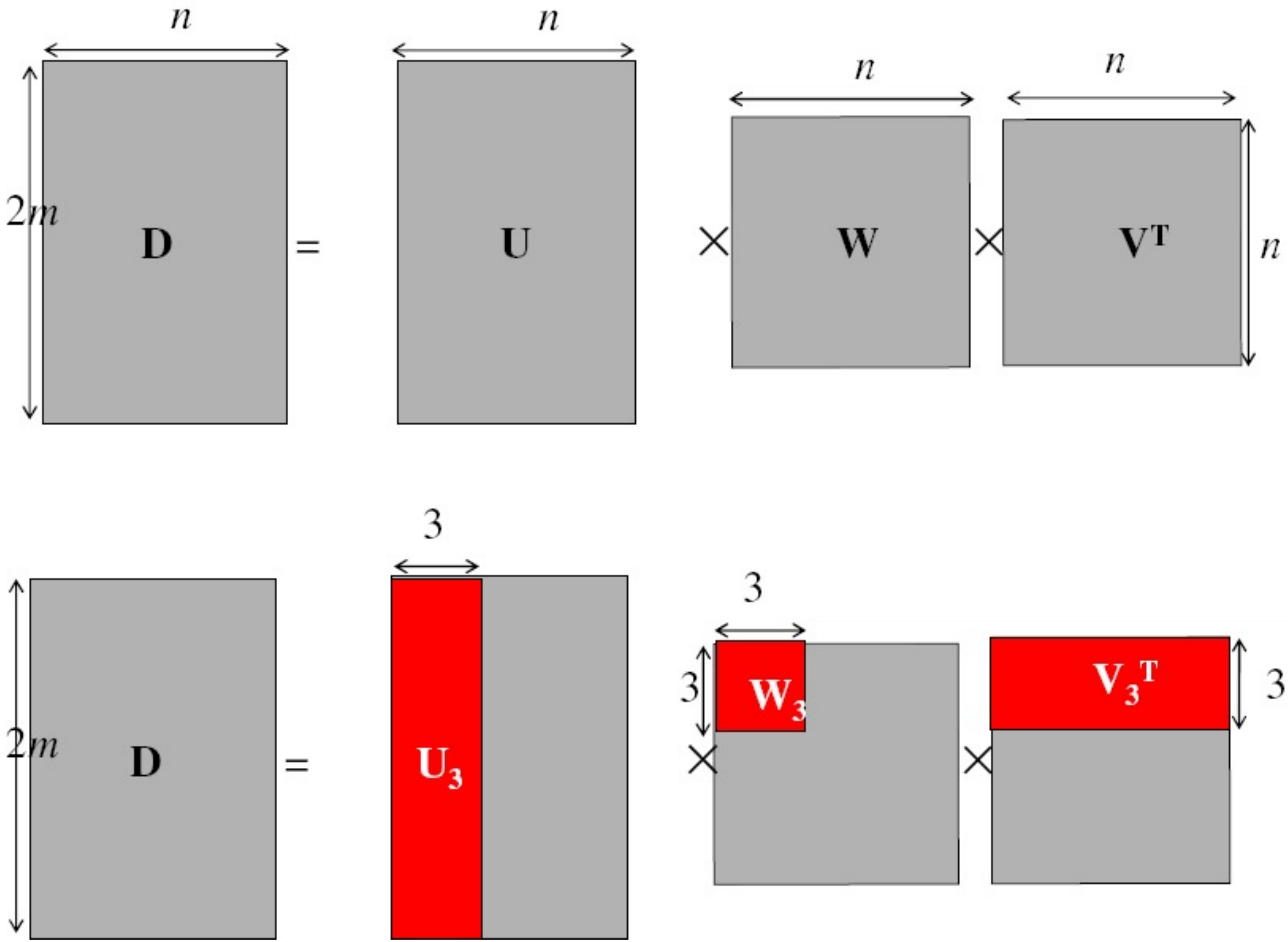
C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2):137-154, November 1992.

Factorizing the measurement matrix



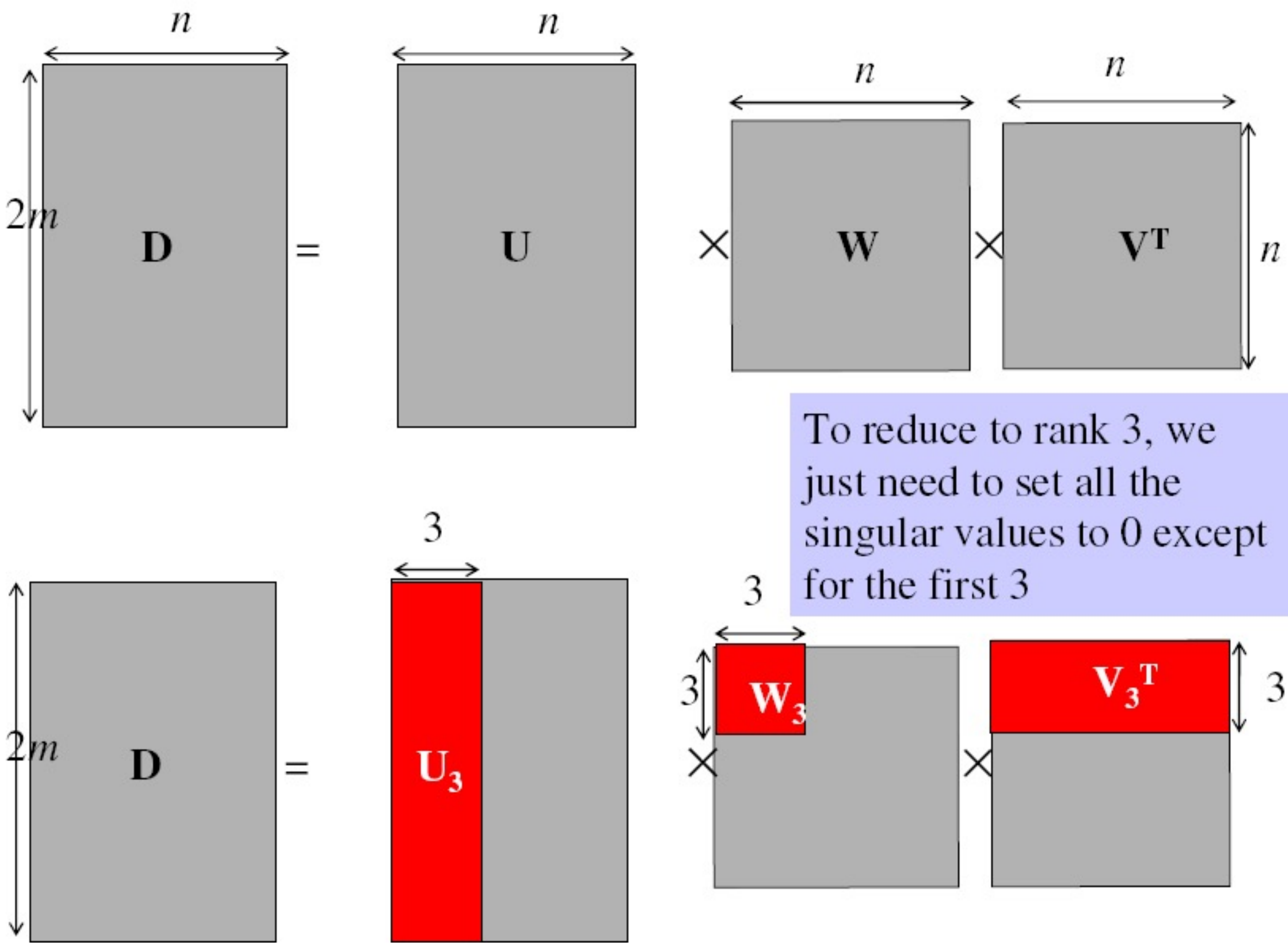
Factorizing the measurement matrix

- Singular value decomposition of D:



Factorizing the measurement matrix

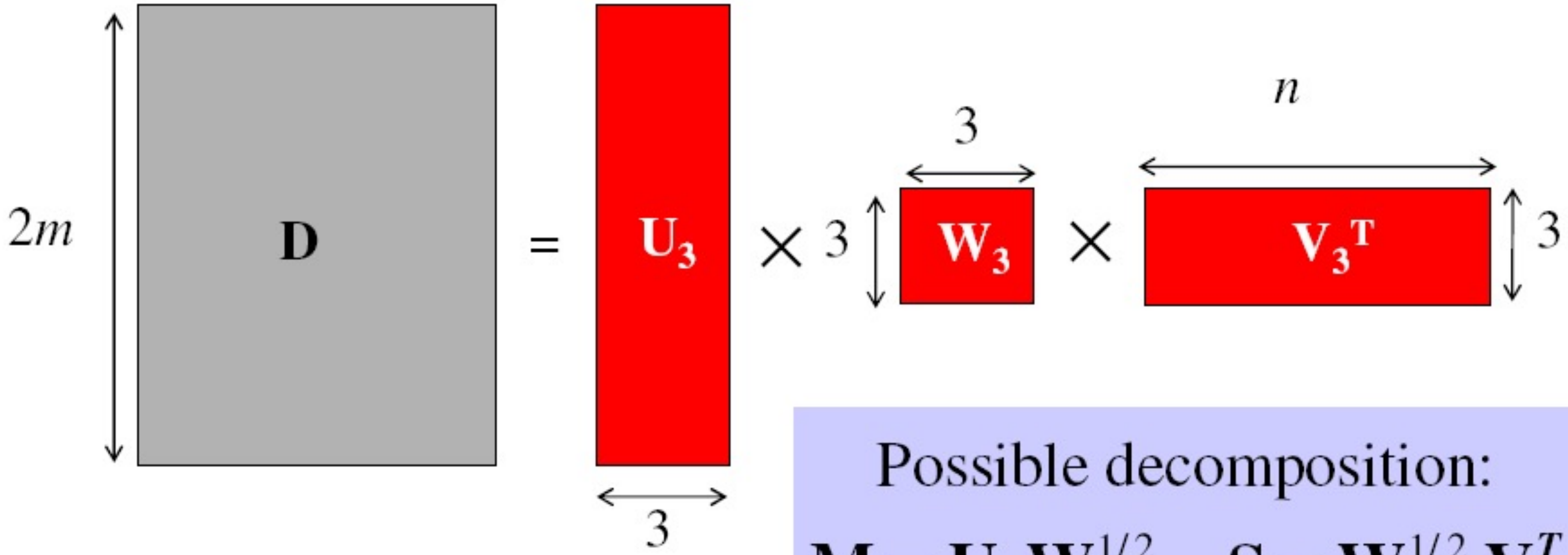
- Singular value decomposition of D:



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

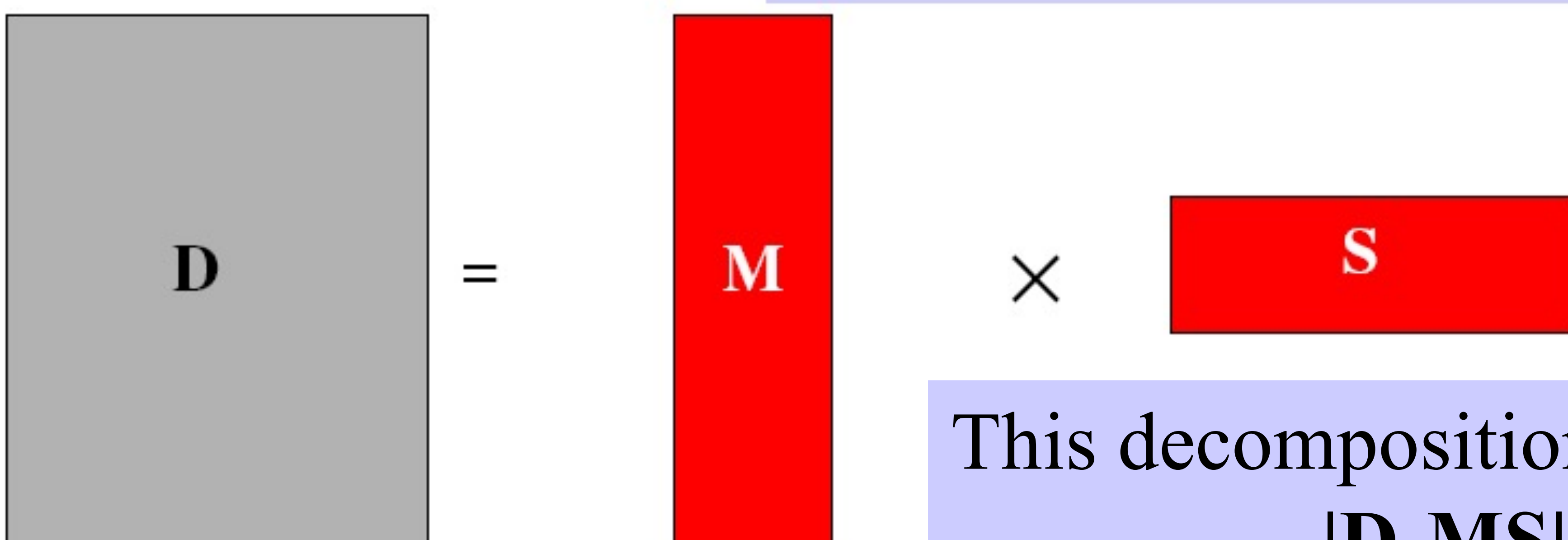
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



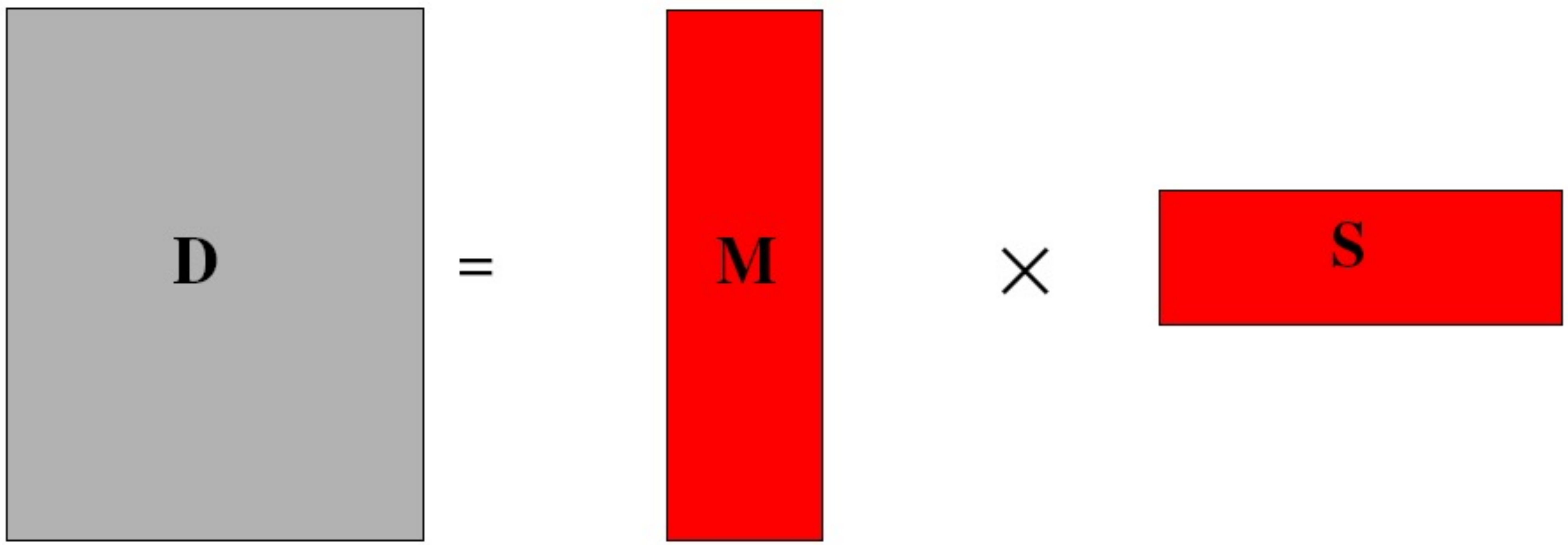
Possible decomposition:

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



This decomposition minimizes
$|\mathbf{D}\text{-}\mathbf{MS}|^2$

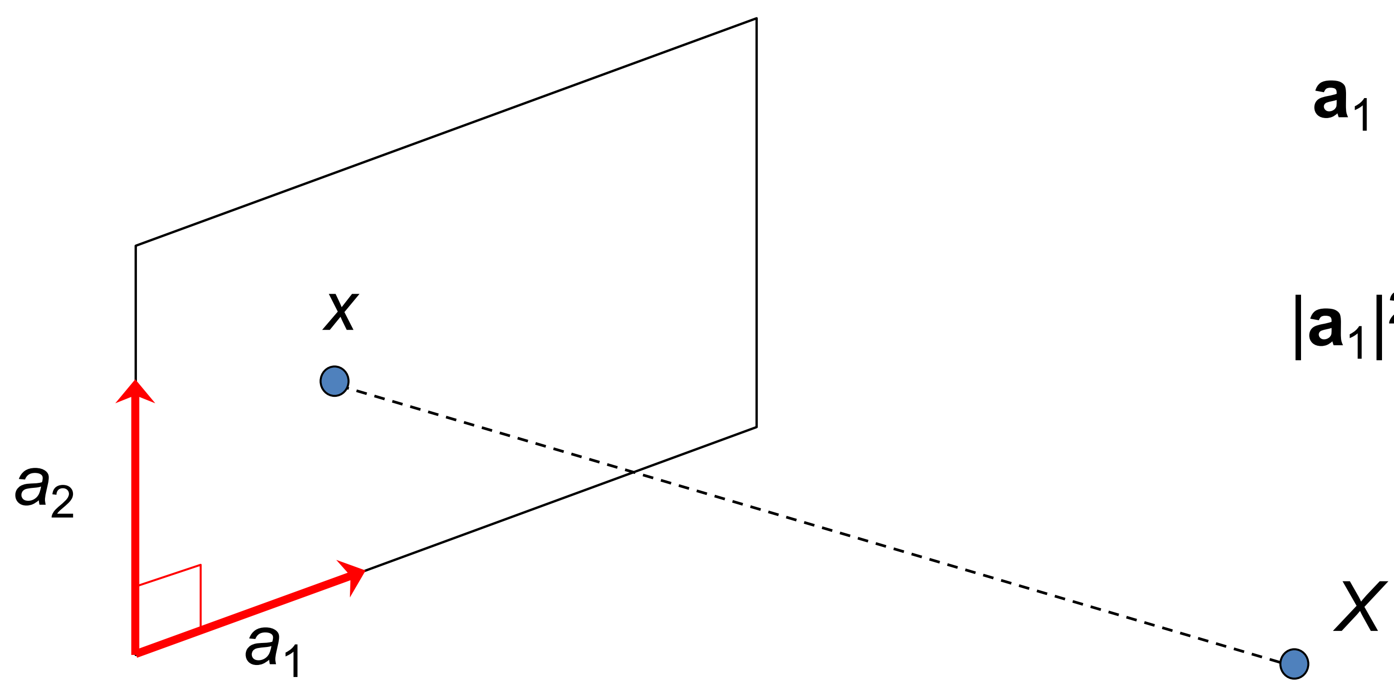
Affine ambiguity



- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations **M** → **MC**, **S** → $\mathbf{C}^{-1}\mathbf{S}$
- 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

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

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

Slide: D. Hoiem

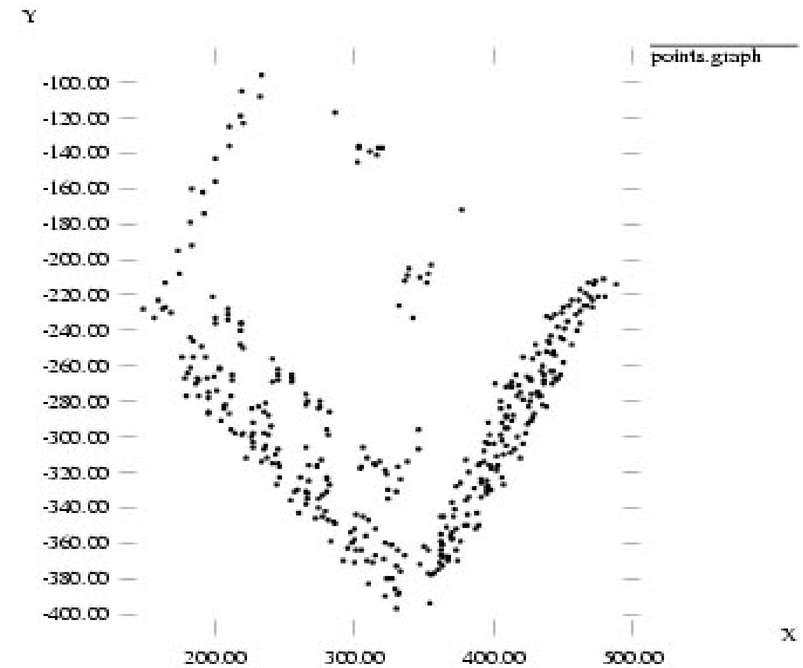
Algorithm summary

- Given: *m* images and *n* features $\mathbf{x}_{ij}$
- For each image *i, c*enter the feature coordinates
- Construct a $2m \times n$ measurement matrix **D**:
  - Column *j* contains the projection of point *j* in all views
  - Row *i* contains one coordinate of the projections of all the *n* points in image *i*
- Factorize **D**:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\ \mathbf{W}\ \mathbf{V}^{\mathbf{T}}$
  - Create $\mathbf{U}_3$ by taking the first 3 columns of **U**
  - Create $\mathbf{V}_3$ by taking the first 3 columns of **V**
  - Create $\mathbf{W}_3$ by taking the upper left 3×3 block of **W**
- Create the motion and shape matrices:
  - $\mathbf{M} = \mathbf{U}_3\mathbf{W}_3^{½}$ and $\mathbf{S} = \mathbf{W}_3^{½}\ \mathbf{V}_3^{\mathrm{T}}$ (**or** $\mathbf{M} = \mathbf{U}_3$ and $\mathbf{S} = \mathbf{W}_3\mathbf{V}_3^{\mathrm{T}}$)
- Eliminate affine ambiguity

Reconstruction results

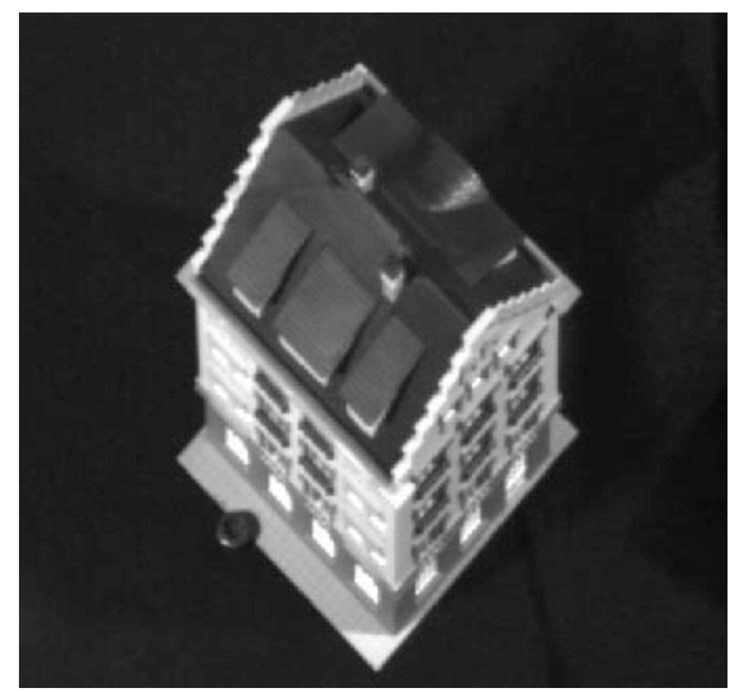
1

60



120

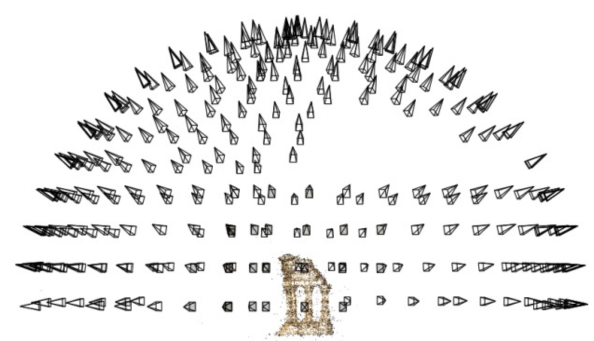
150



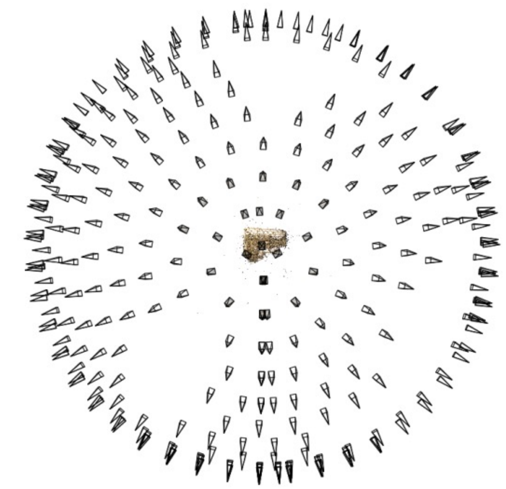
C. Tomasi and T. Kanade. [Shape and motion from image streams under orthography: A factorization method.](#) *IJCV*, 9(2):137-154, November 1992.

Large-scale structure from motion

Structure from motion



Reconstruction (side)



(top)

- Input: images with points in correspondence
  $p_{i,j} = (u_{i,j}, v_{i,j})$

- Output
  - structure: 3D location $\mathbf{x}_i$ for each point p_i
  - motion: camera parameters $\mathbf{R}_j$, $\mathbf{t}_j$ possibly $\mathbf{K}_j$

- Objective function: minimize *reprojection error*

15,464

37,383

76,389

Standard way to view photos

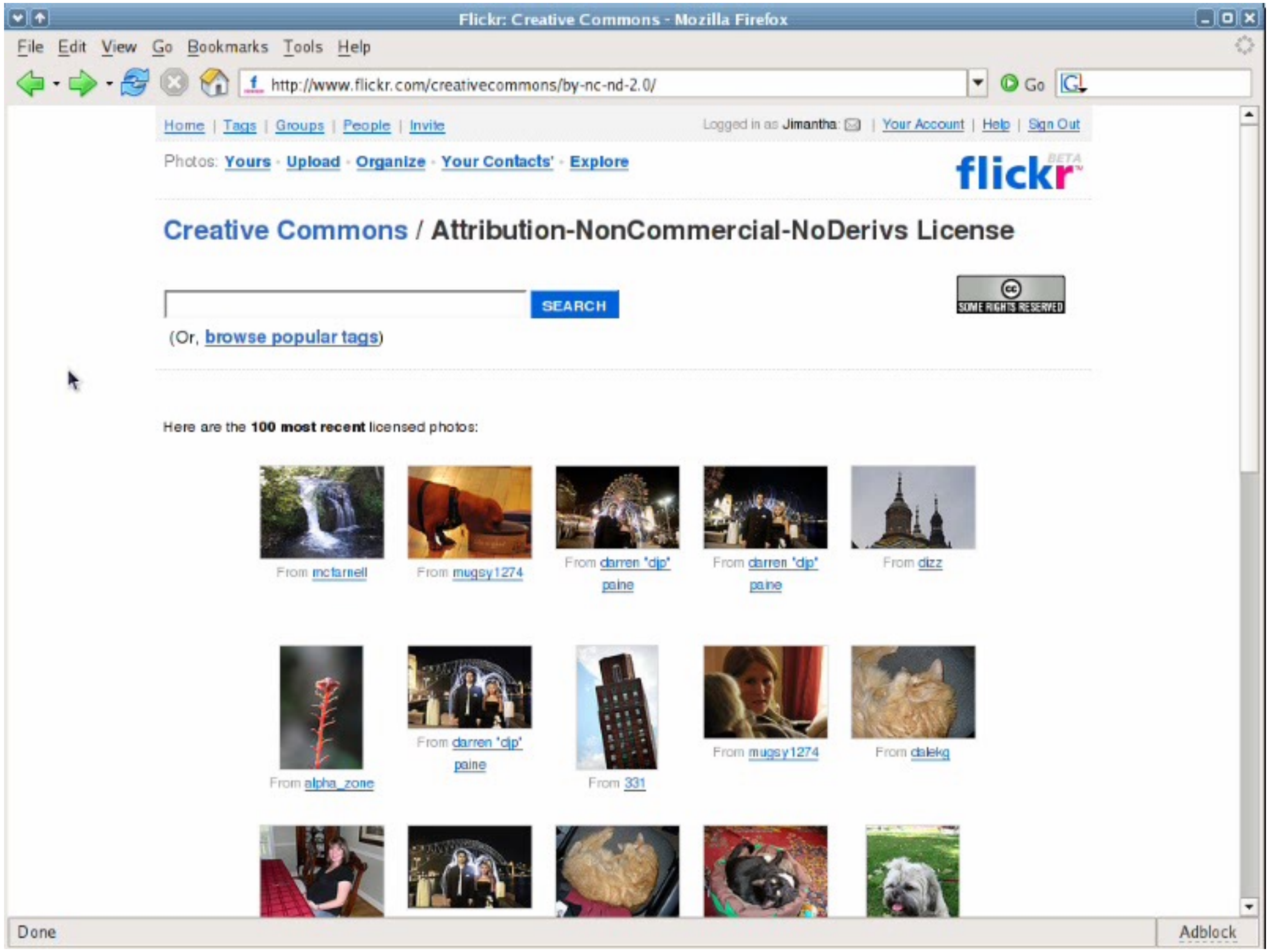
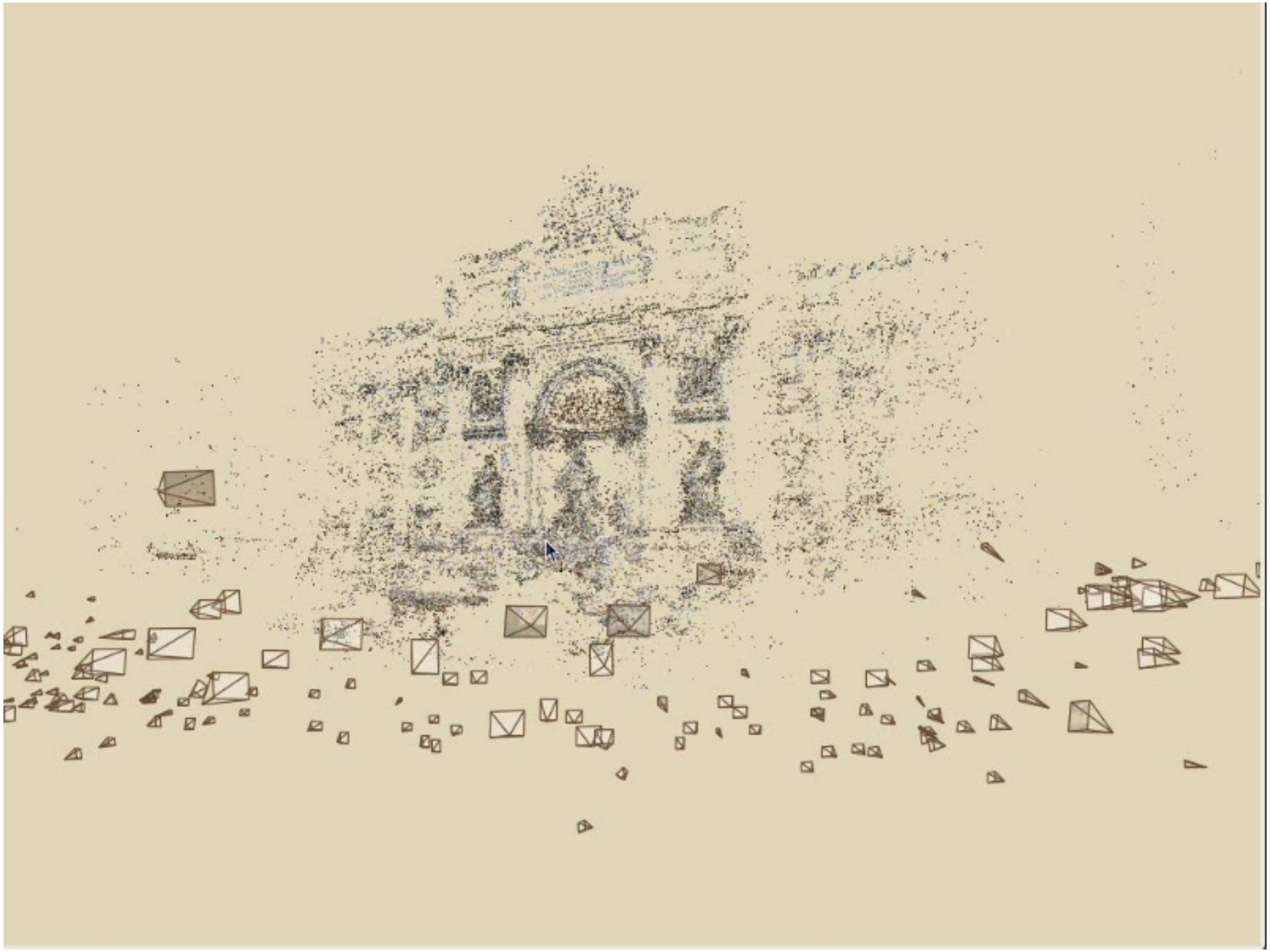
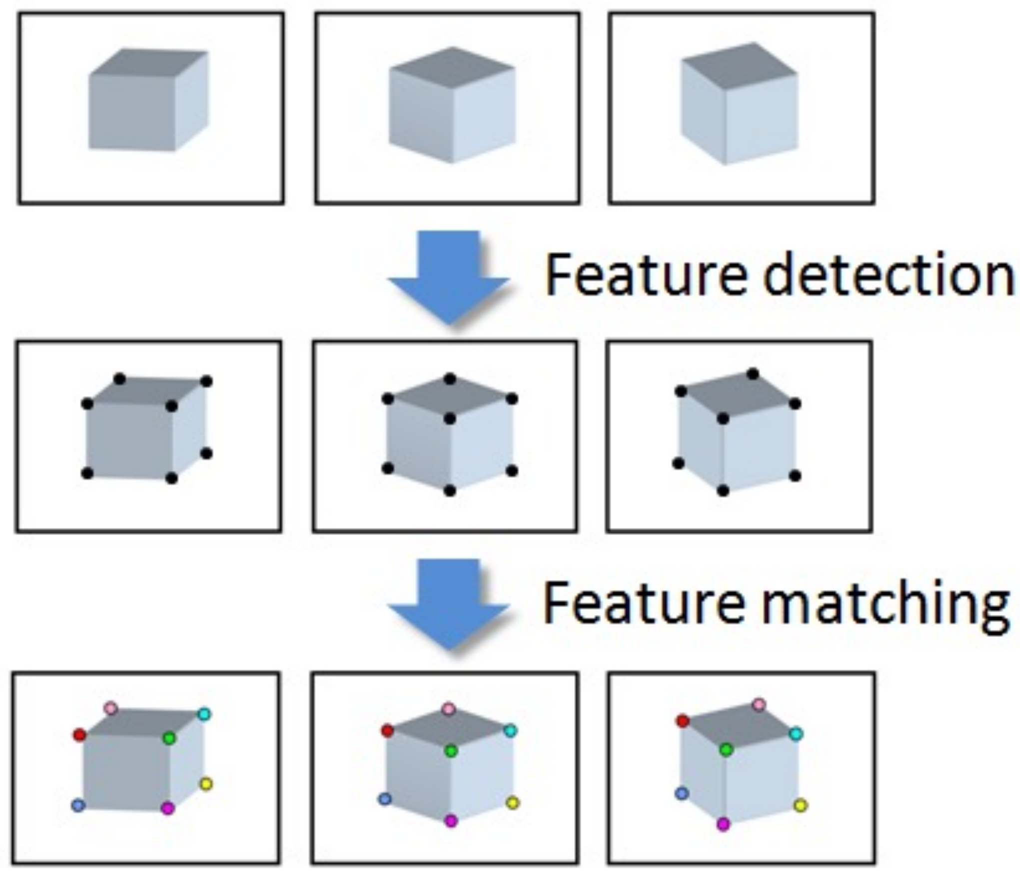


Photo Tourism

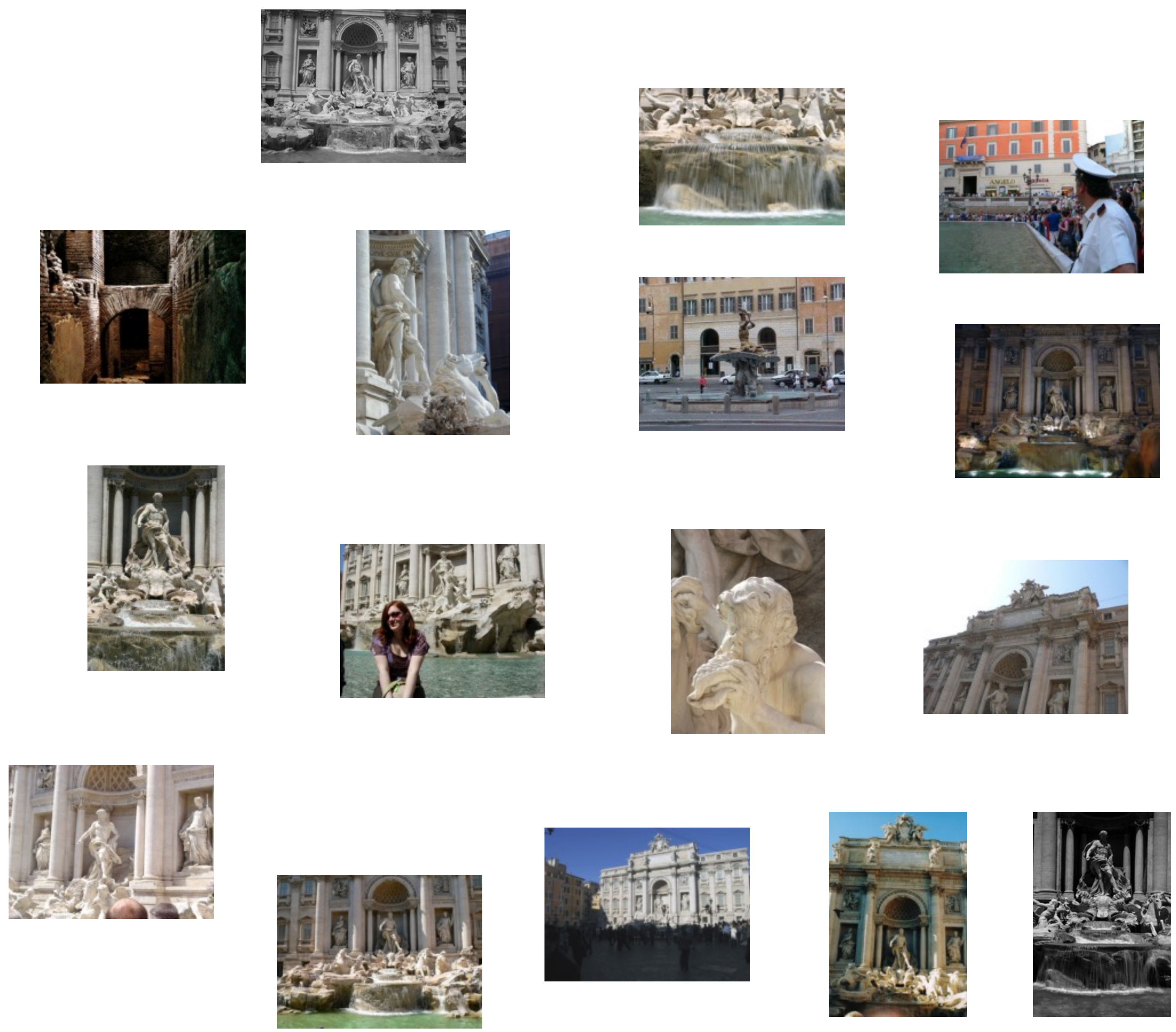


Input: Point correspondences



Feature detection

Detect features using SIFT [Lowe, IJCV 2004]

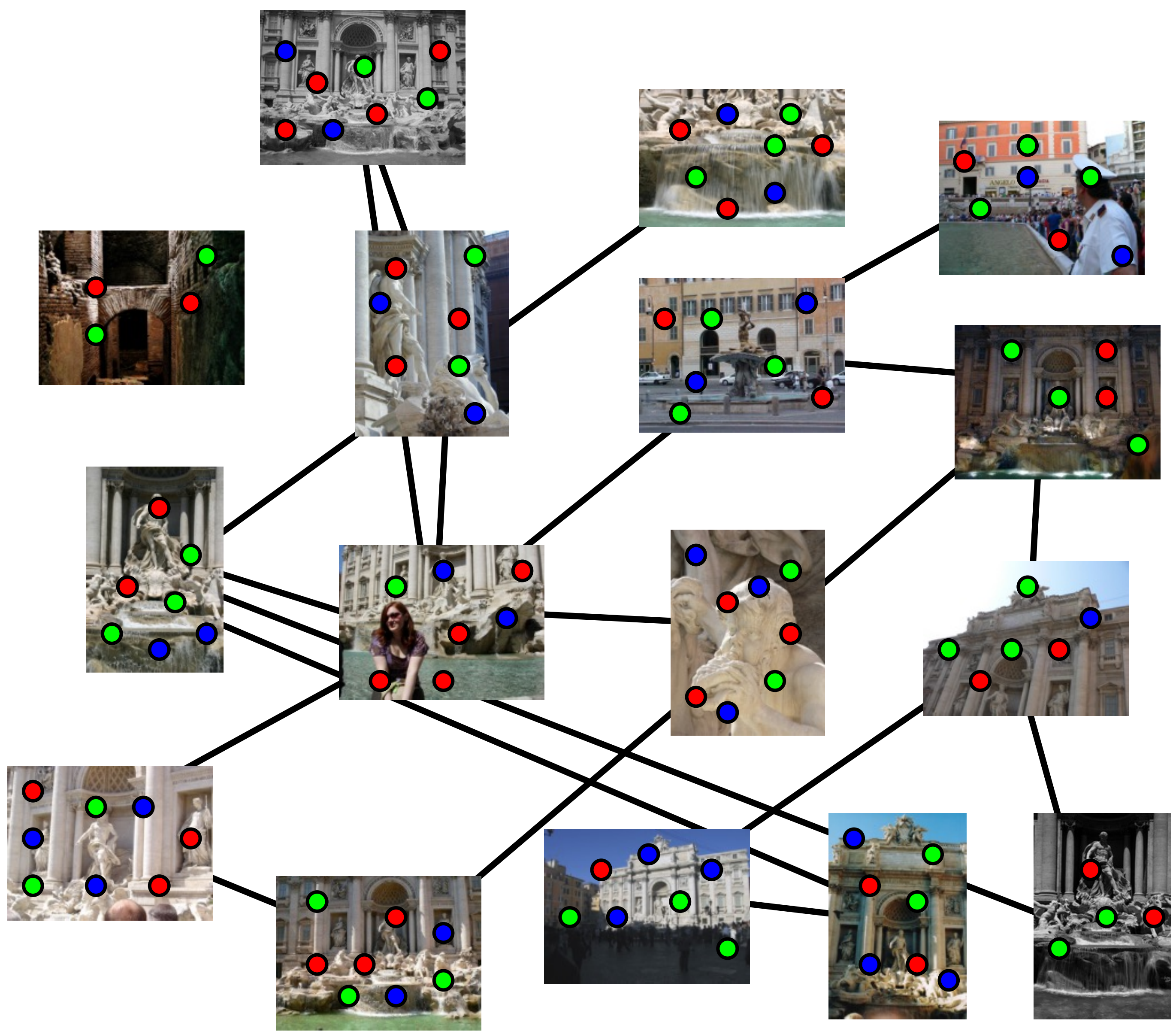


Feature description

Describe features using SIFT [Lowe, IJCV 2004]

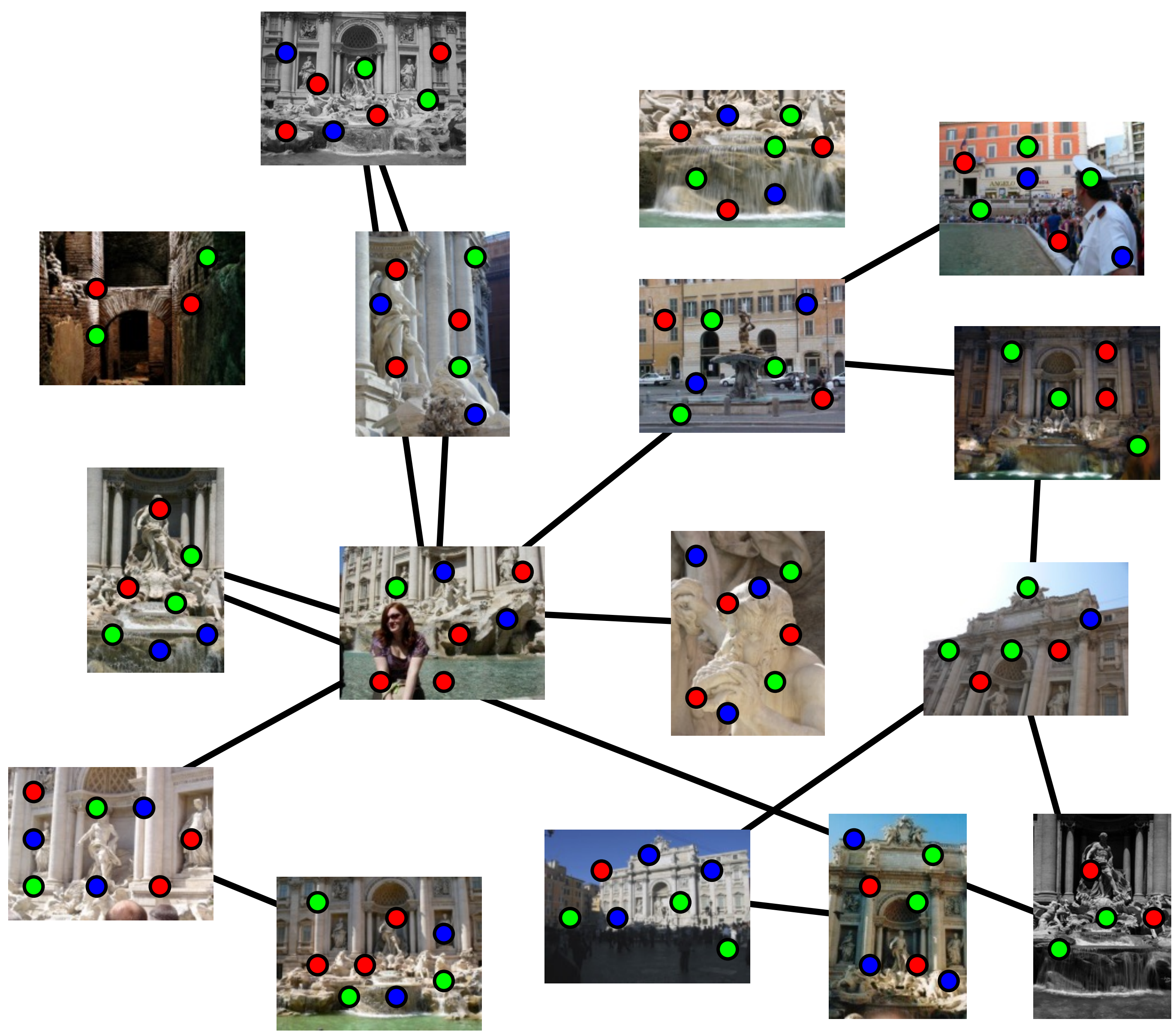
Feature matching

Match features between each pair of images



Feature matching

Refine matching using RANSAC to estimate fundamental matrix between each pair



Correspondence estimation

- Link up pairwise matches to form connected components of matches across several images

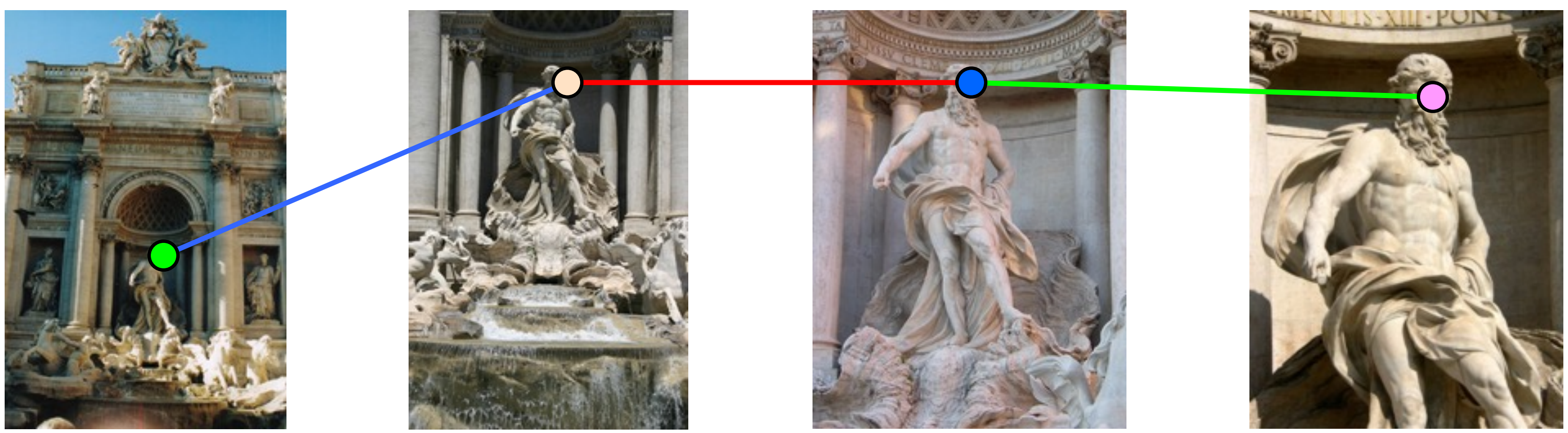
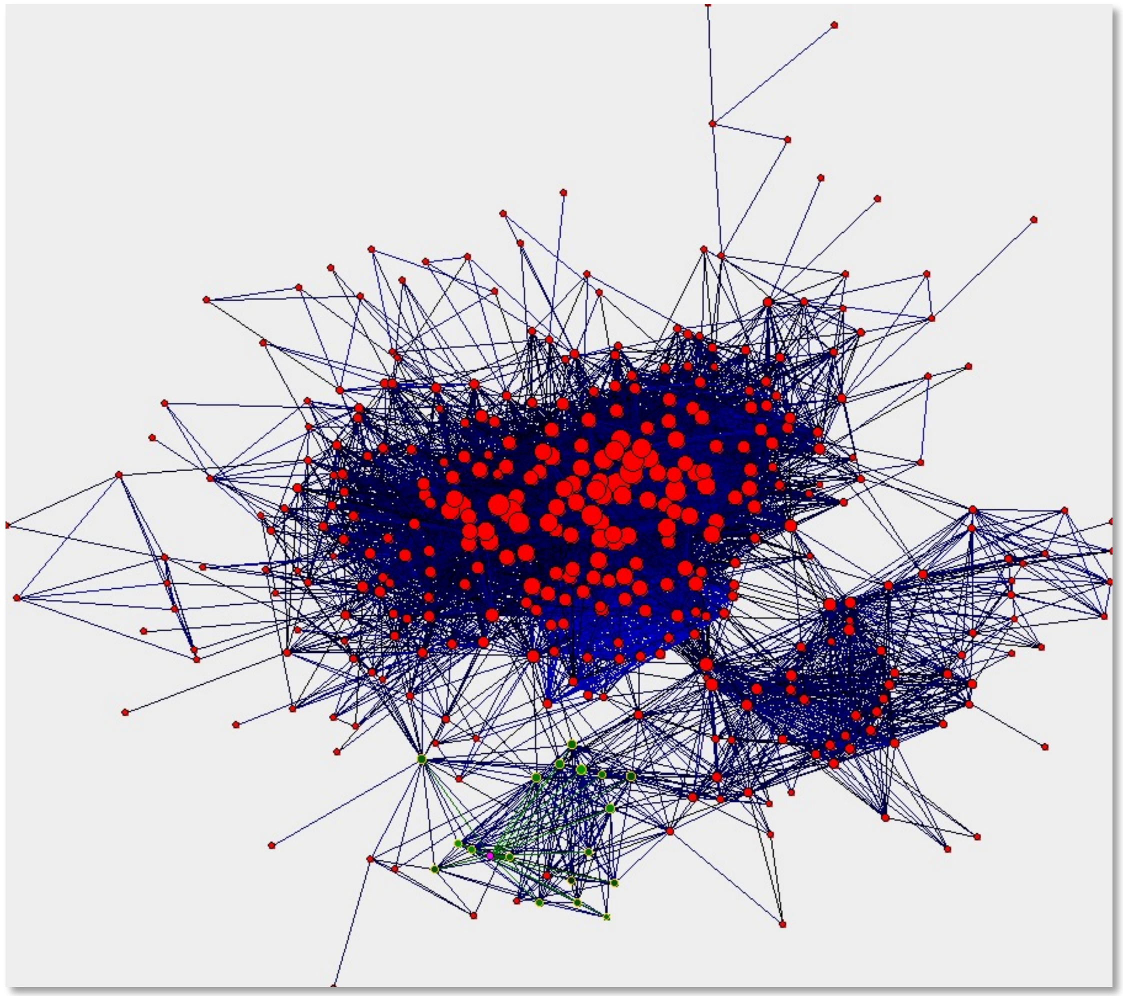


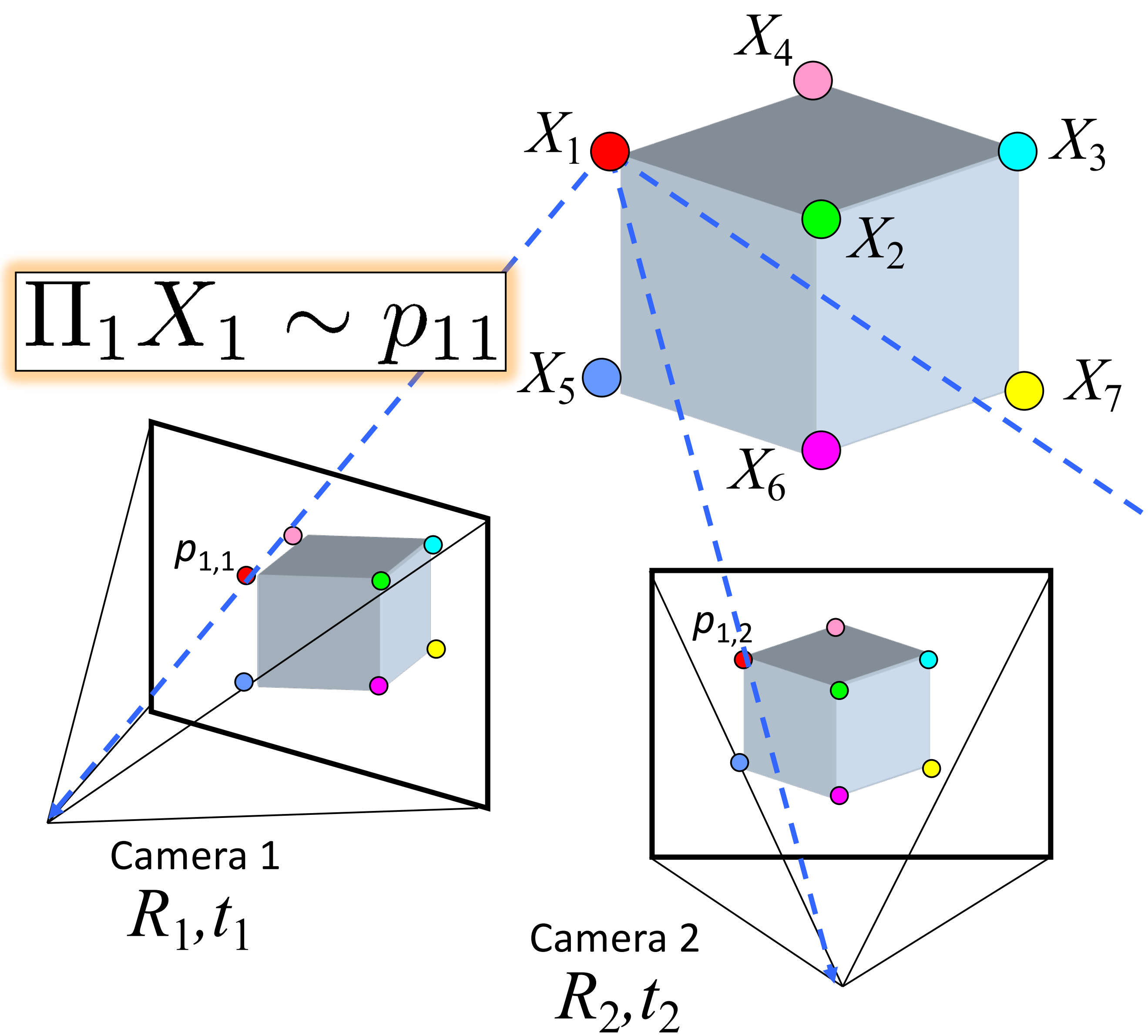
Image 1 Image 2 Image 3 Image 4

Image connectivity graph



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

Structure from motion



minimize

$$g(\mathbf{R}, \mathbf{T}, \mathbf{X})$$

non-linear least squares

Global structure from motion

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \underbrace{w_{ij}}_{\substack{\textit{indicator variable}: \\ \text{is point } i \text{ visible in image } j\text{ ?}}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\textit{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\textit{observed} \\ \text{image location}}} \right\|^2$$

- Minimizing this function is called *bundle adjustment*
  – Optimized using non-linear least squares, e.g. Levenberg-Marquardt

Problem size

- What are the variables?
- How many variables per camera?
- How many variables per point?

- Trevi Fountain collection

    466 input photos

    > 100,000 3D points

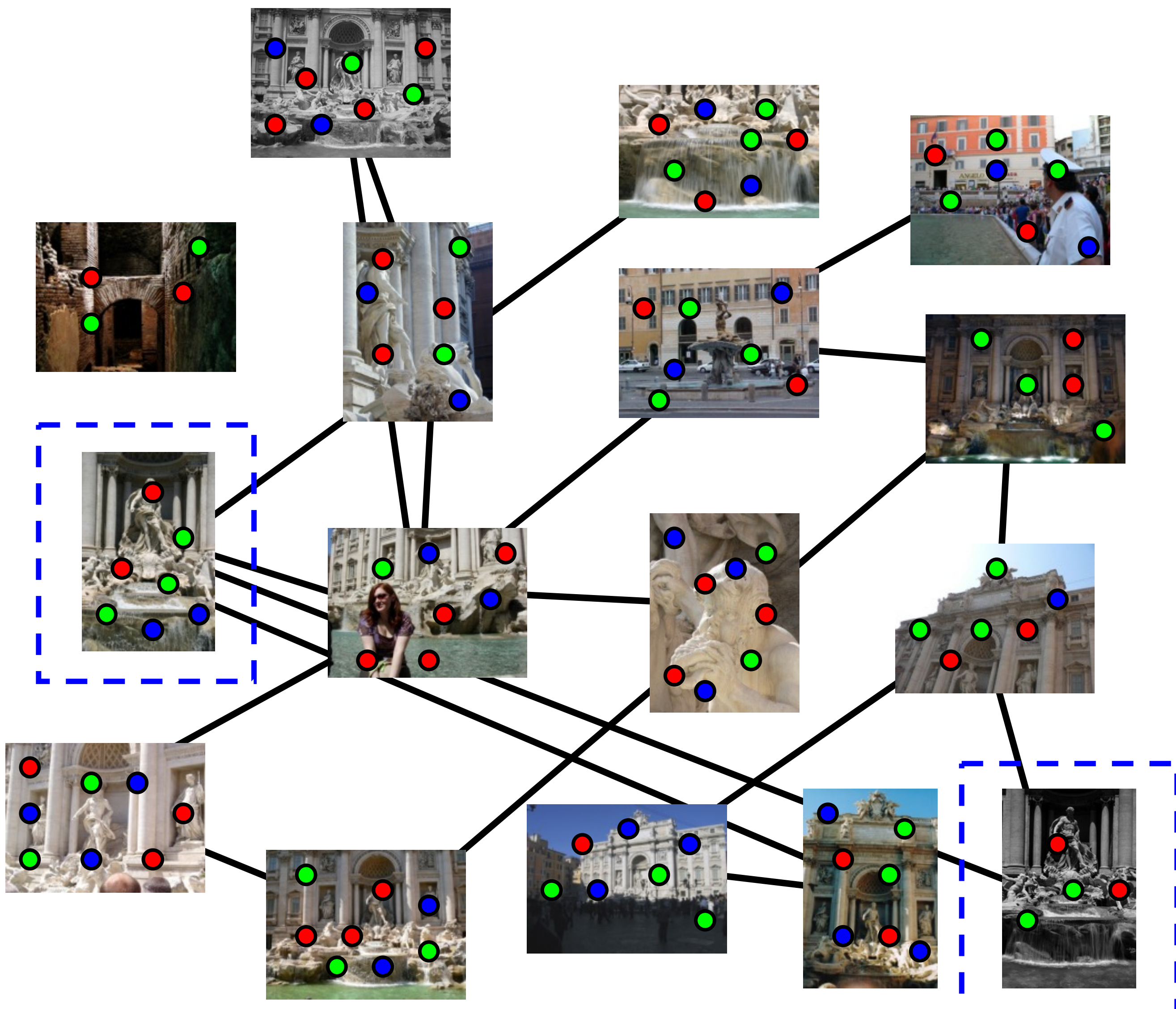
    = very large optimization problem

Doing bundle adjustment

- Minimizing *g* is difficult
  - *g* is non-linear due to rotations, perspective division
  - lots of parameters: 3 for each 3D point, 6 for each camera
  - difficult to initialize
  - gauge ambiguity: error is invariant to a similarity transform (translation, rotation, uniform scale)

- Many techniques use non-linear least-squares (NLLS) optimization (*bundle adjustment*)
  - Levenberg-Marquardt is one common algorithm for NLLS
  - Lourakis, **The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm**, http://www.ics.forth.gr/~lourakis/sba/
  - http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm

Initialization: Incremental structure from motion



Incremental structure from motion

Final reconstruction

More examples

More examples

More examples

Even larger scale SfM

City-scale structure from motion

- “Building Rome in a day”

http://grail.cs.washington.edu/projects/rome/

SfM applications

- 3D modeling
- Surveying
- Robot navigation and mapmaking
- Visual effects ("Match moving")
  - https://www.youtube.com/watch?v=RdYWp70P_kY

Applications – Photosynth

Applications – Hyperlapse

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

References

Basic reading:

- Szeliski textbook, Chapter 7.
- Hartley and Zisserman, Chapter 18.