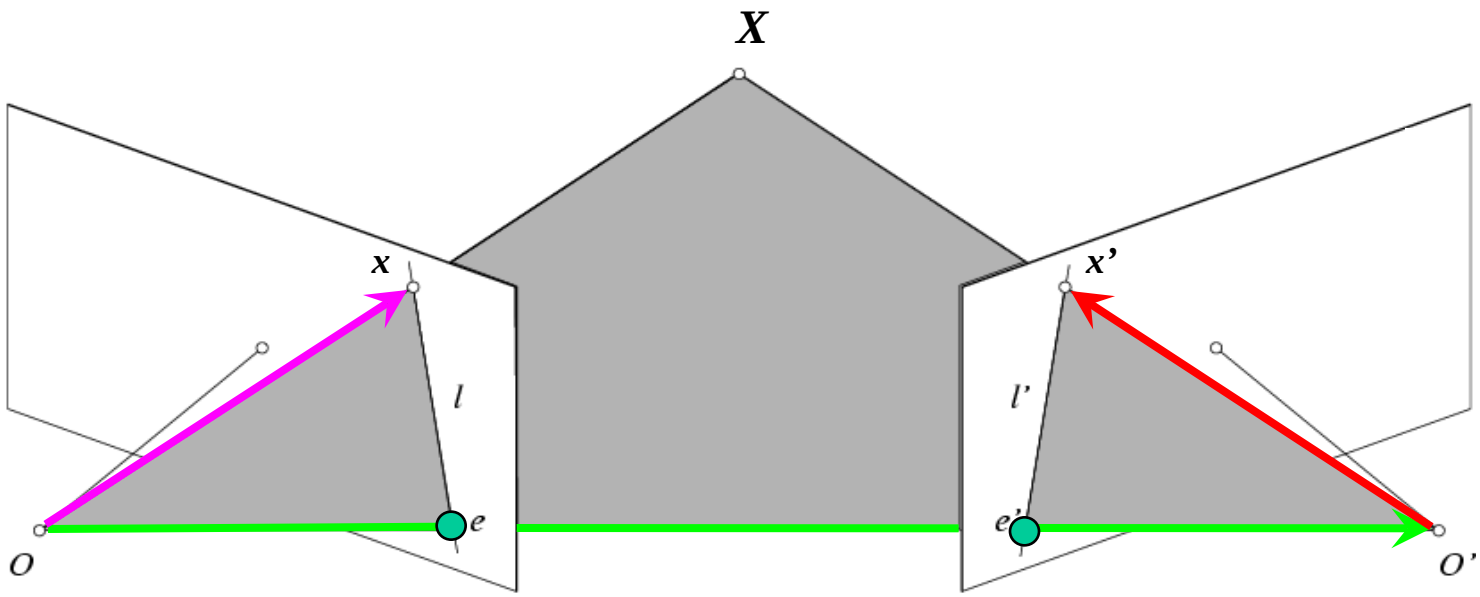
Structure from Motion

Vinay P. Namboodiri

Slide credit to Svetlana Lazebnik and Robert T. Collins

Epipolar constraint: Uncalibrated case



$$\hat{x}^T E \hat{x}' = 0 \quad \Rightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

$$\hat{x} = K^{-1} x$$

$$\hat{x}' = K'^{-1} x'$$

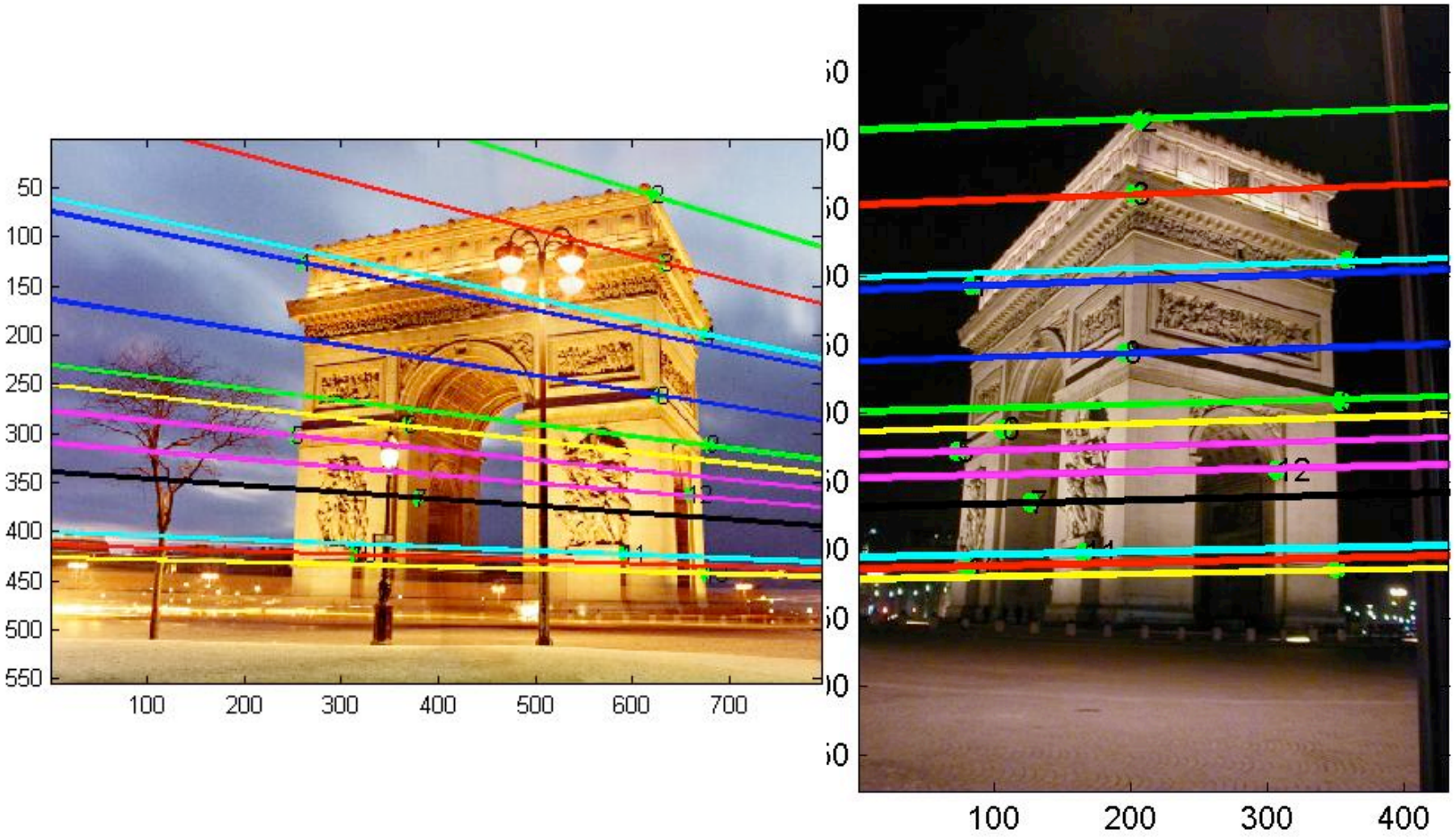
Fundamental Matrix
(Faugeras and Luong, 1992)

Robert Collins
CSE486, Penn State

Example

Robert Collins
CSE486, Penn State

Example



Robert Collins
CSE486, Penn State

Example

$$F = \begin{pmatrix} -0.00310695 & -0.0025646 & 2.96584 \\ -0.028094 & -0.00771621 & 56.3813 \\ 13.1905 & -29.2007 & -9999.79 \end{pmatrix}$$

Robert Collins
CSE486, Penn State

E/F Matrix Summary

Longuet-Higgins equation $p_r^T E p_l = 0$

Epipolar lines: $\tilde{p_r}^T \tilde{l_r} = 0$ $\tilde{p_l}^T \tilde{l_l} = 0$

$\tilde{l_r} = E p_l$ $\tilde{l_l} = E^T p_r$

Epipoles: $e_r^T E = 0$ $E e_l = 0$

E vs F: E works in film coords (calibrated cameras)
F works in pixel coords (uncalibrated cameras)

Computing F from Point Matches

- Assume that you have m correspondences
- Each correspondence satisfies:

$$\bar{p_r}_i^T F \bar{p}_{li} = 0 \quad i = 1, \ldots, m$$

- F is a 3x3 matrix (9 entries)
- Set up a **HOMOGENEOUS** linear system with 9 unknowns

Robert Collins
CSE486, Penn State

Computing F

$$\bar{p}_{li} = (x_i \;\; y_i \;\; 1)^T \quad \bar{p}_{ri} = (x'_i \;\; y'_i \;\; 1)^T$$

$$\bar{p_r}_i^T F \bar{p}_{li} = 0 \quad i = 1, \ldots, m$$

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

Robert Collins
CSE486, Penn State

Computing F

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

$$x_i x'_i f_{11} + x_i y'_i f_{21} + x_i f_{31} +$$
$$y_i x'_i f_{12} + y_i y'_i f_{22} + y_i f_{32} +$$
$$x'_i f_{13} + y'_i f_{23} + f_{33} = 0$$

Robert Collins
CSE486, Penn State

Computing F

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

Given m point correspondences…

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_mx'_m & x_my'_m & x_m & y_mx'_m & y_my'_m & y_m & x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0$$

Think: how many points do we need?

How Many Points?

Unlike a homography, where each point correspondence contributes two constraints (rows in the linear system of equations), for estimating the essential/fundamental matrix, each point only contributes one constraint (row). [because the Longuet-Higgins / Epipolar constraint is a scalar eqn.]

Thus need at least 8 points.

Hence: The Eight Point algorithm!

Algorithm EIGHT_POINT

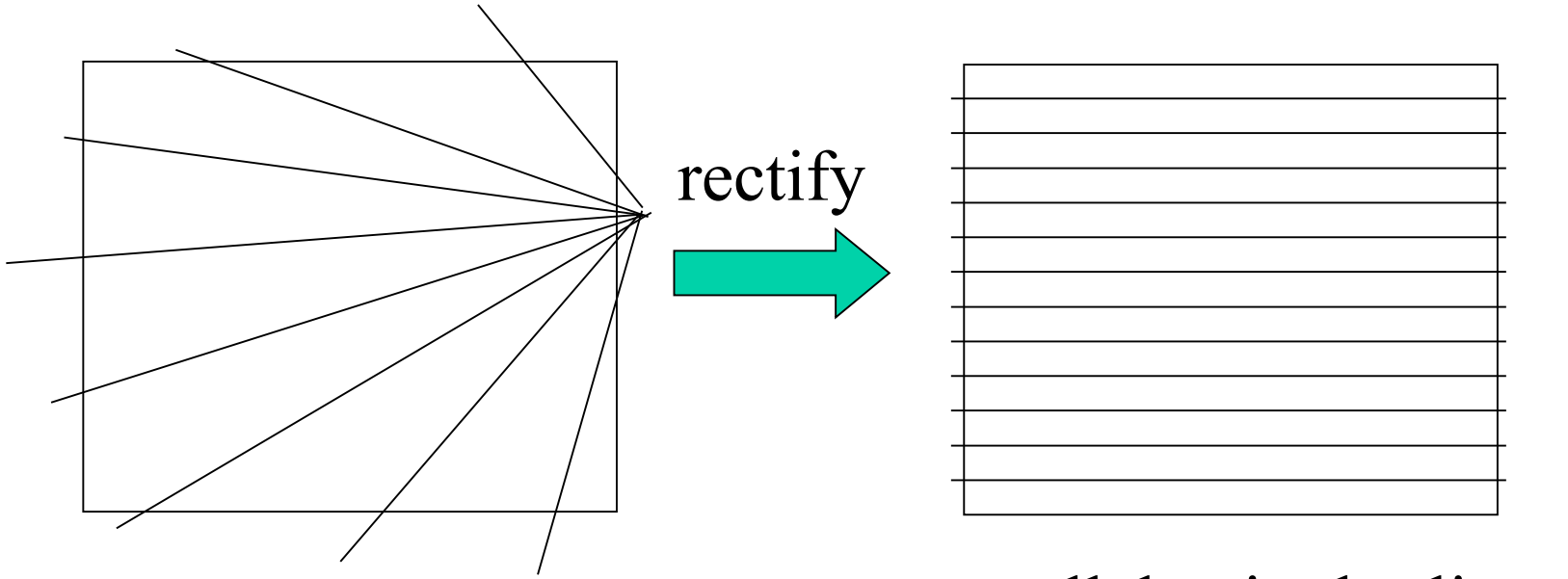
F must be singular (remember, it is rank 2, since it is important for it to have a left and right nullspace, i.e. the epipoles). To enforce rank 2 constraint:

- Find the SVD of F: $F = U_f D_f V_f^T$
- Set smallest s.v. of F to 0 to create D'_f
- Recompute F: $F = U_f D'_f V_f^T$

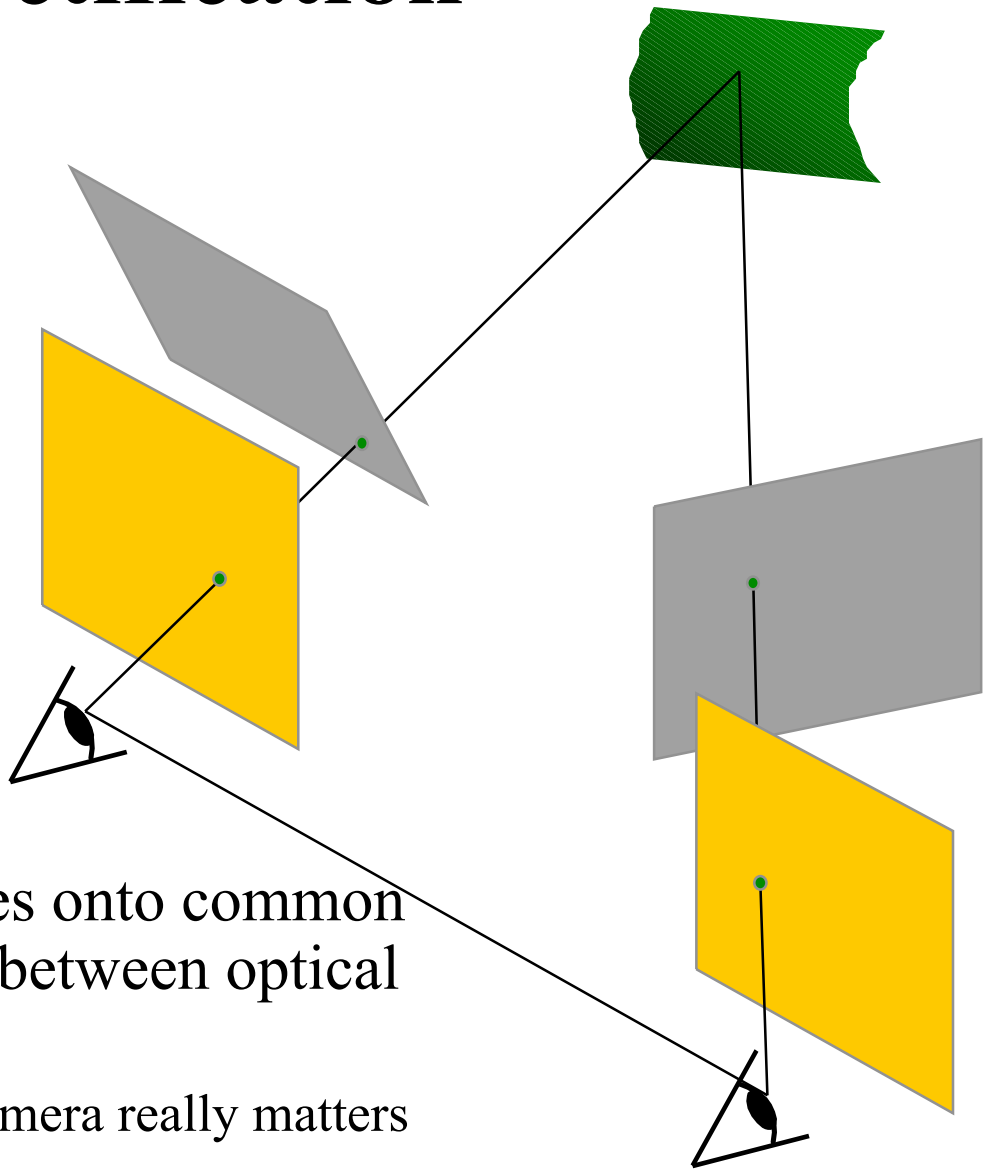
Robert Collins
CSE486, Penn State

A Practical Issue

How to "rectify" the images so that any scan-line stereo alorithm that works for simple stereo can be used to find dense matches (i.e. compute a disparity image for every pixel).



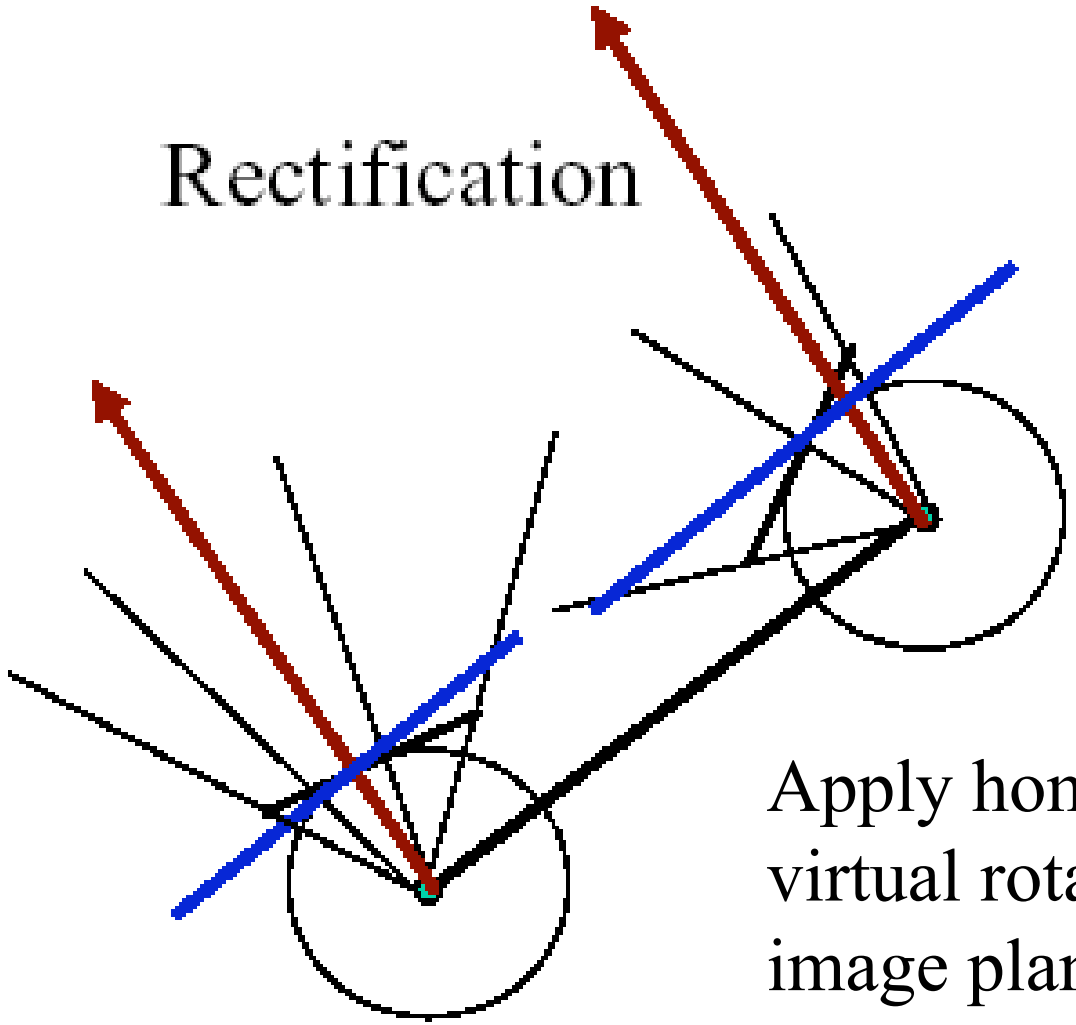
Stereo Rectification



- Image Reprojection
  - reproject image planes onto common plane parallel to line between optical centers
- Notice, only focal point of camera really matters

Robert Collins
CSE486, Penn State

General Idea

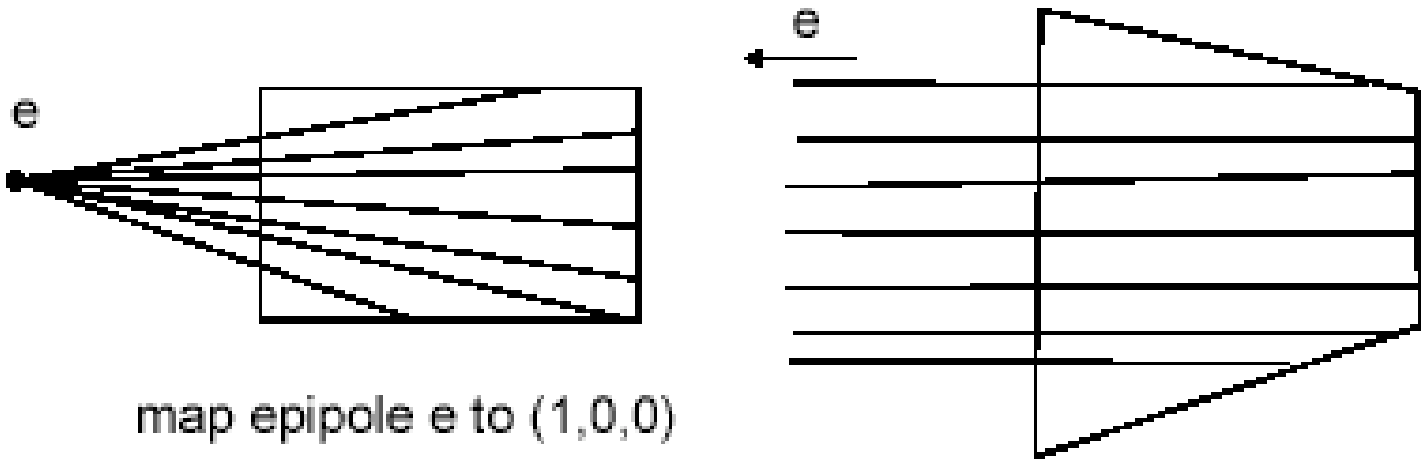


Apply homography representing a virtual rotation to bring the image planes parallel with the baseline (epipoles go to infinity).

Robert Collins
CSE486, Penn State

General Idea, continued

Apply projective transformation so that epipolar lines correspond to horizontal scanlines



$$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = He$$

map epipole e to (1,0,0)

try to minimize image distortion

Note that rectified images usually not rectangular

Robert Collins
CSE486, Penn State

Image Rectification

Build the rotation:

$$R_{rect} = \begin{bmatrix} \mathbf{e_1}^T \\ \mathbf{e_2}^T \\ \mathbf{e_3}^T \end{bmatrix}$$

with:

$$\mathbf{e_1} = \frac{\mathbf{T}}{||\mathbf{T}||}$$

where T is just a unit vector representing the epipole in the left image. We know how to compute this from E, from last class.

Robert Collins
CSE486, Penn State

Algorithm Rectification

- Build the matrix R_{rect}
- Set $R_l = R_{rect}$ and $R_r = R.R_{rect}$
- For each left point $p_l = (x,y,f)^T$
  - compute $R_l\, p_l = (x', y', z')^T$
  - Compute $p'_l = f/z' \; (x', y' \; z')^T$
- Repeat above for the right camera with R_r

Example

Robert Collins
CSE486, Penn State

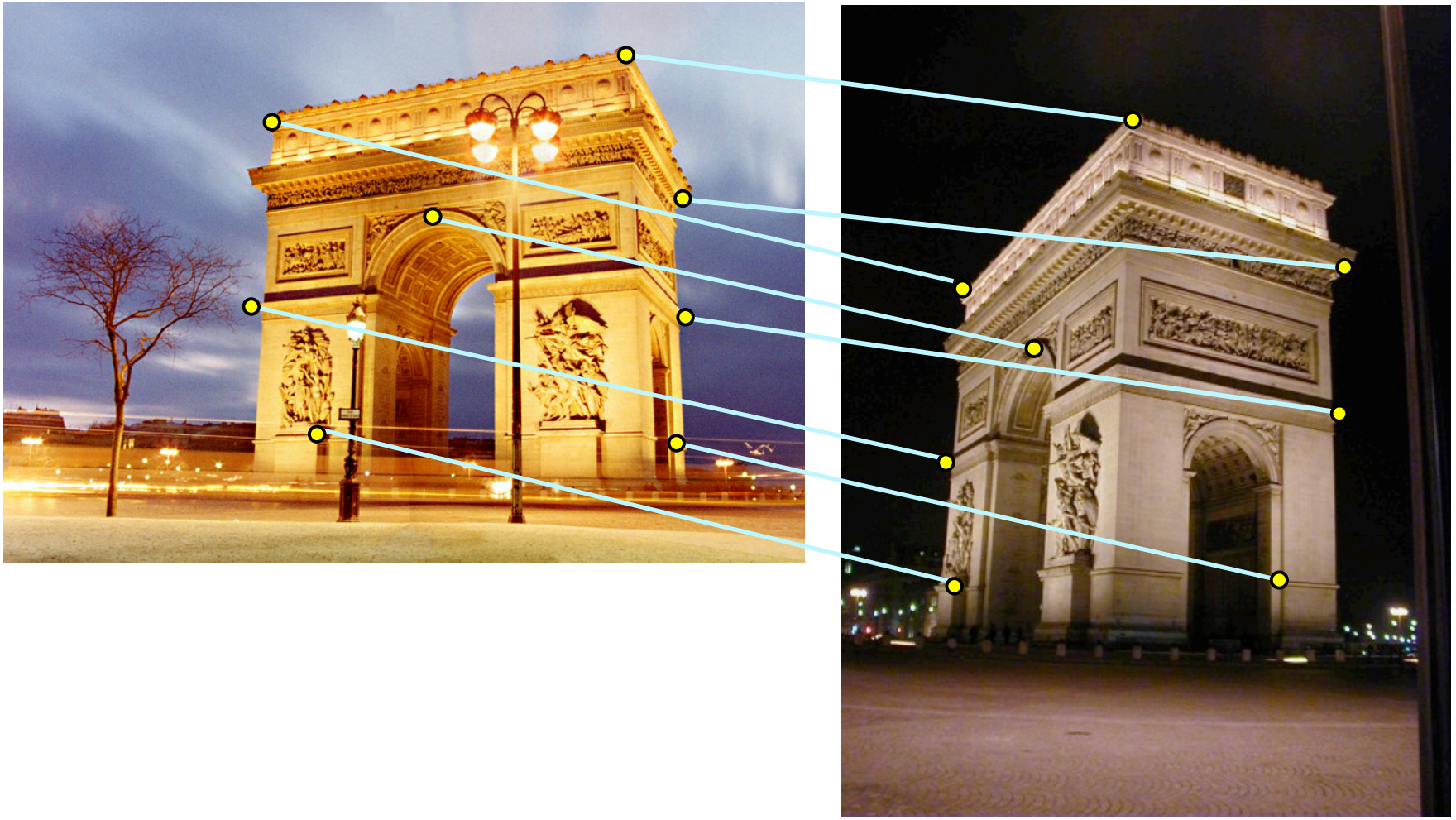
Example

Rectified Pair

Robert Collins
CSE486, Penn State

Steps to General Stereo

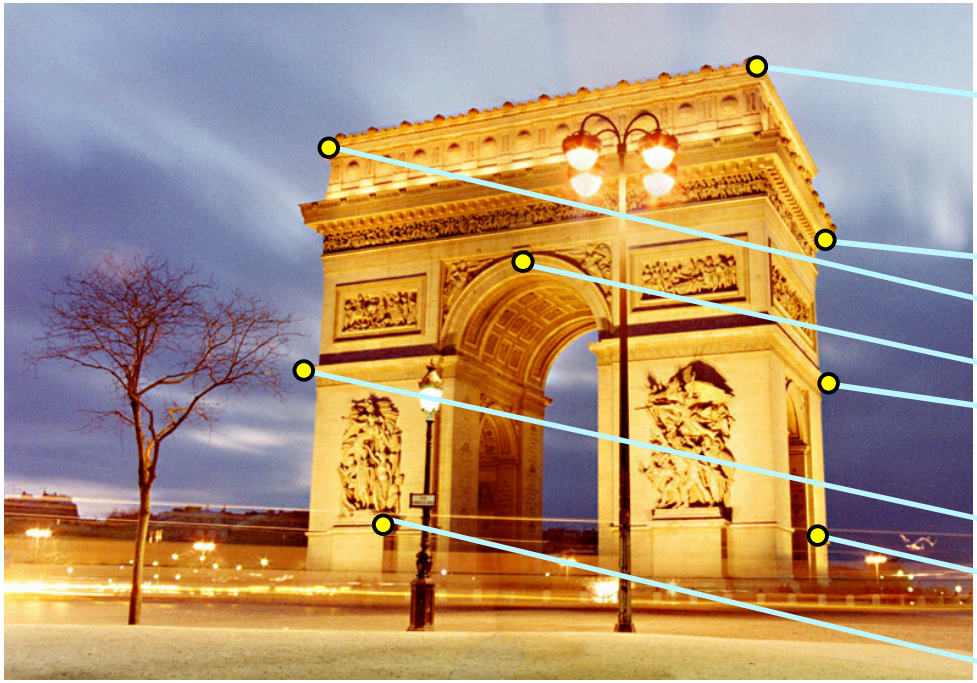
find 8 or more initial point matches (somehow)

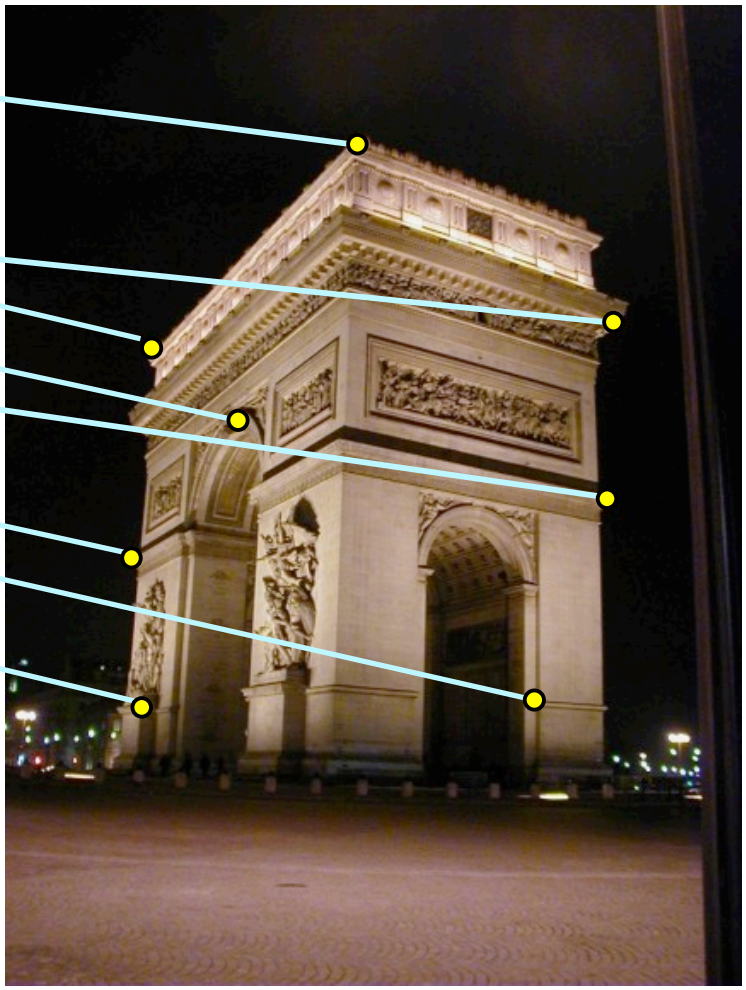


Robert Collins
CSE486, Penn State

Steps to General Stereo

Compute F matrix using 8-point algorithm



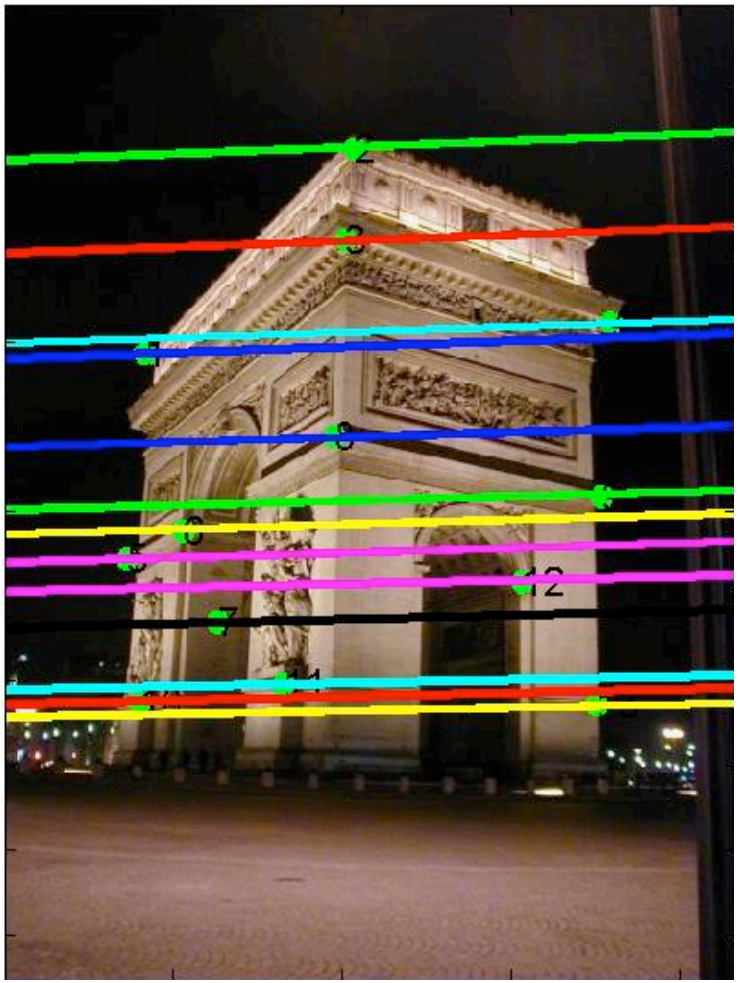


$$F = \begin{pmatrix} -0.00310695 & -0.0025646 & 2.96584 \\ -0.028094 & -0.00771621 & 56.3813 \\ 13.1905 & -29.2007 & -9999.79 \end{pmatrix}$$

Robert Collins
CSE486, Penn State

Steps to General Stereo

Infer epipolar geometry (epipoles, epipolar lines) from F.

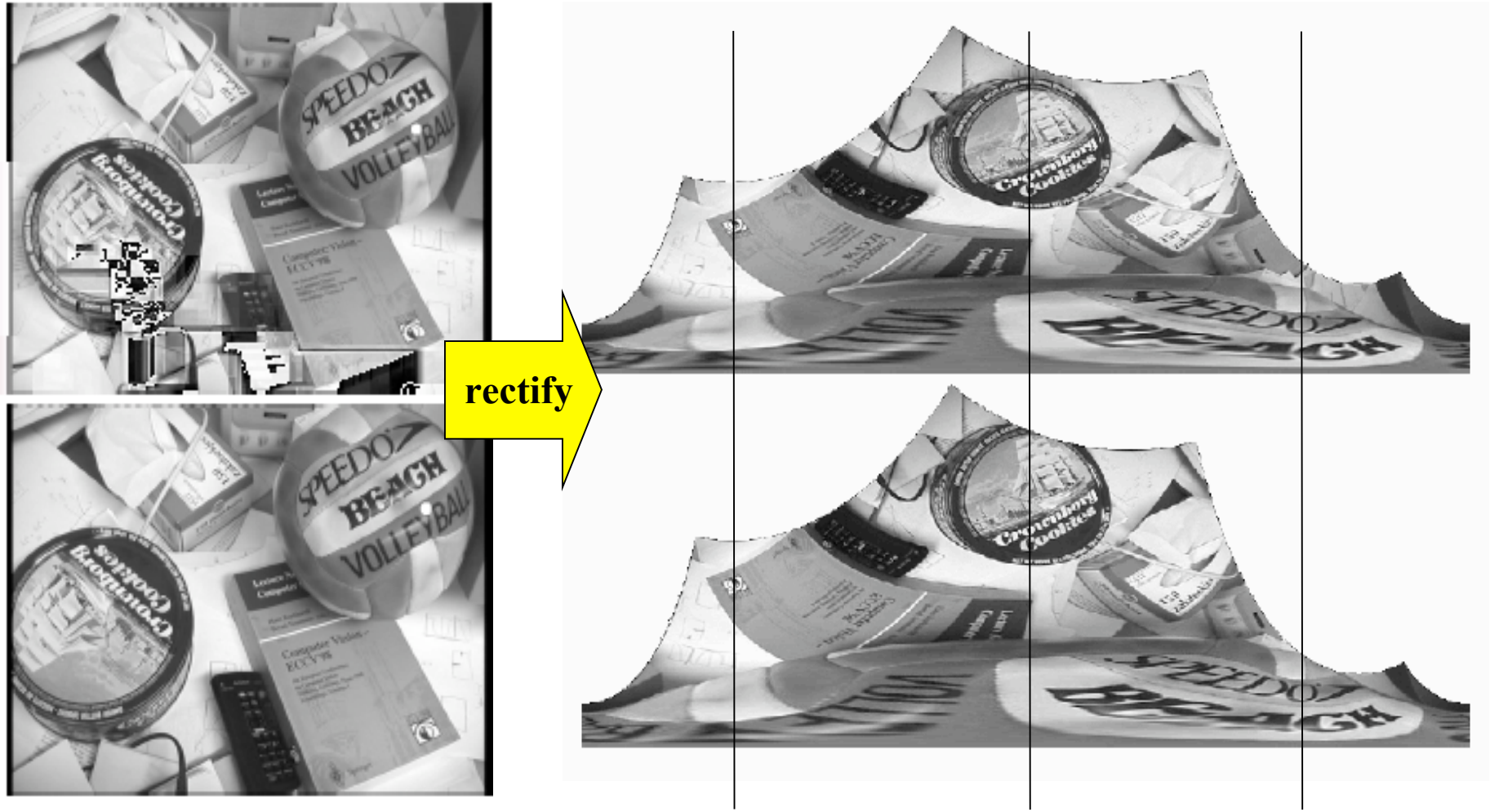


$$F = \begin{pmatrix} -0.00310695 & -0.0025646 & 2.96584 \\ -0.028094 & -0.00771621 & 56.3813 \\ 13.1905 & -29.2007 & -9999.79 \end{pmatrix}$$

Robert Collins
CSE486, Penn State

Steps to General Stereo

Rectify images to get simple scanline stereo pair.

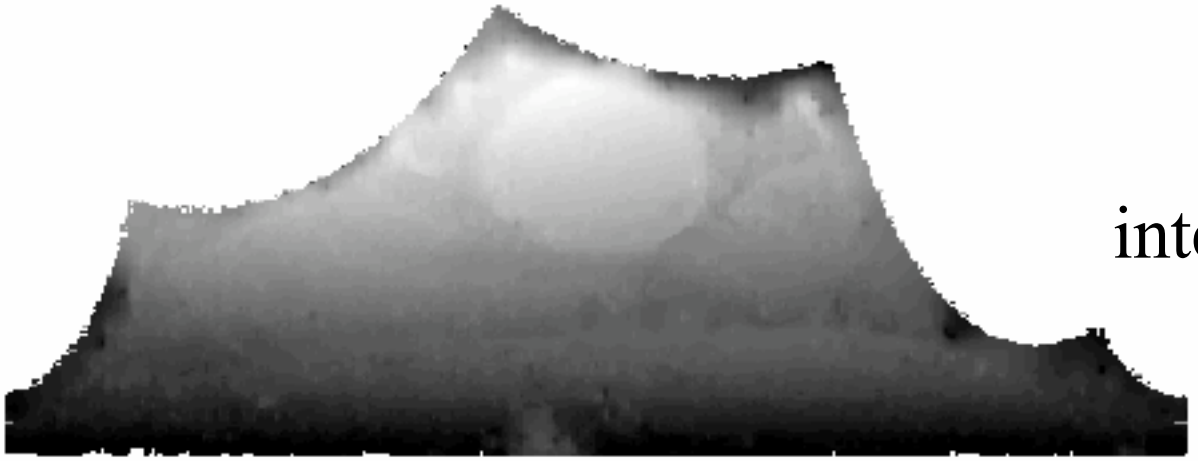


Robert Collins
CSE486, Penn State

Steps to General Stereo

Compute disparity map (correspondence matching)

sparse

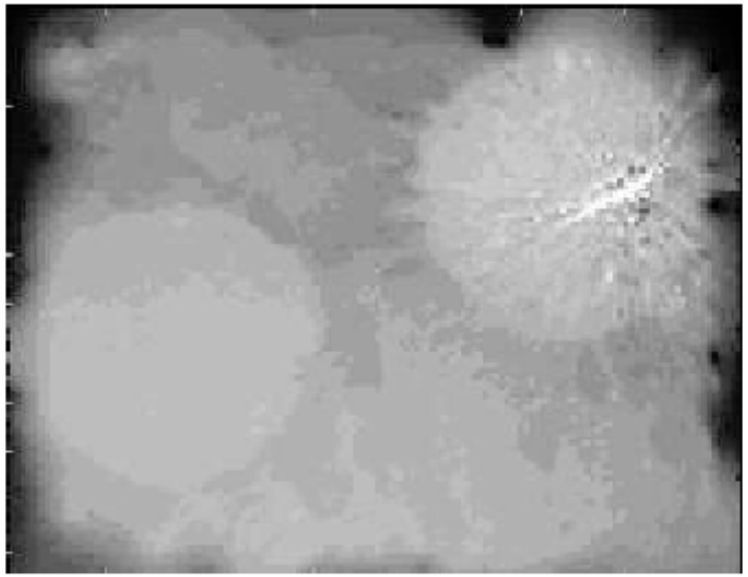


interpolated

Robert Collins
CSE486, Penn State

Steps to General Stereo

Compute 3D surface geometry from disparity map



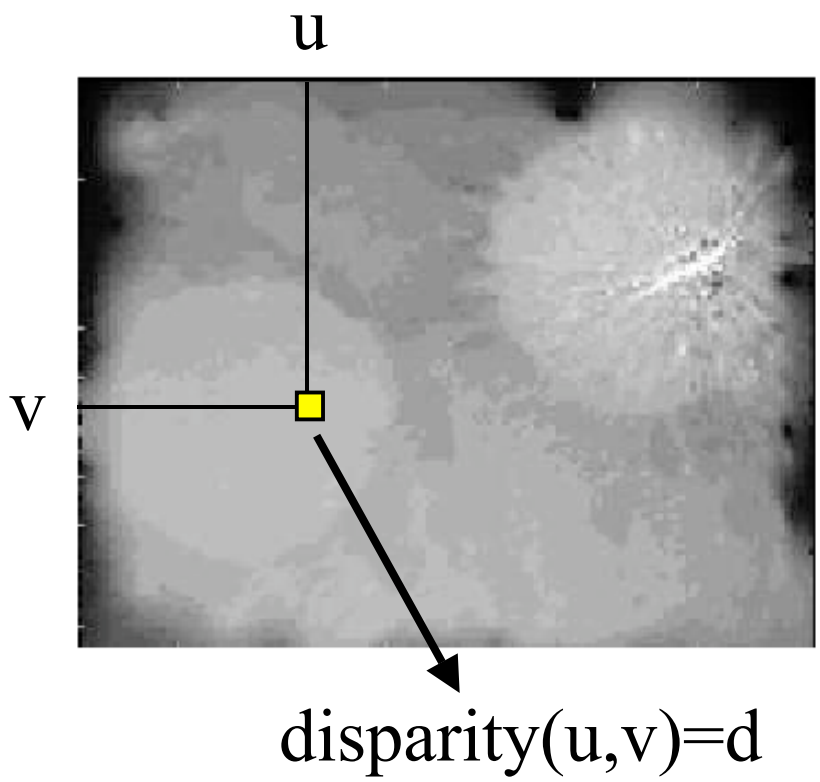
disparity map in
pixel coords

Views of texture mapped
depth surface

Robert Collins
CSE486, Penn State

Reconstruction

Even though we have a dense disparity map, when talking about recovering 3D scene structure, we will consider it to just be a set of point matches.



point match:

(u,v) in left image
matches
(u-d,v) in right image

Robert Collins
CSE486, Penn State

Stereo Reconstruction

Given point correspondences, how to compute 3D point positions using triangulation.

Results depend on how calibrated the system is:

1) Intrinsic and extrinsic parameters known
   Can compute metric 3D geometry
2) Only intrinsic parameters known
   Unknown scale factor
3) Neither intrinsic nor extrinsic known
   Recover structure up to an unknown projective transformation of the scene

Fully Calibrated Stereo

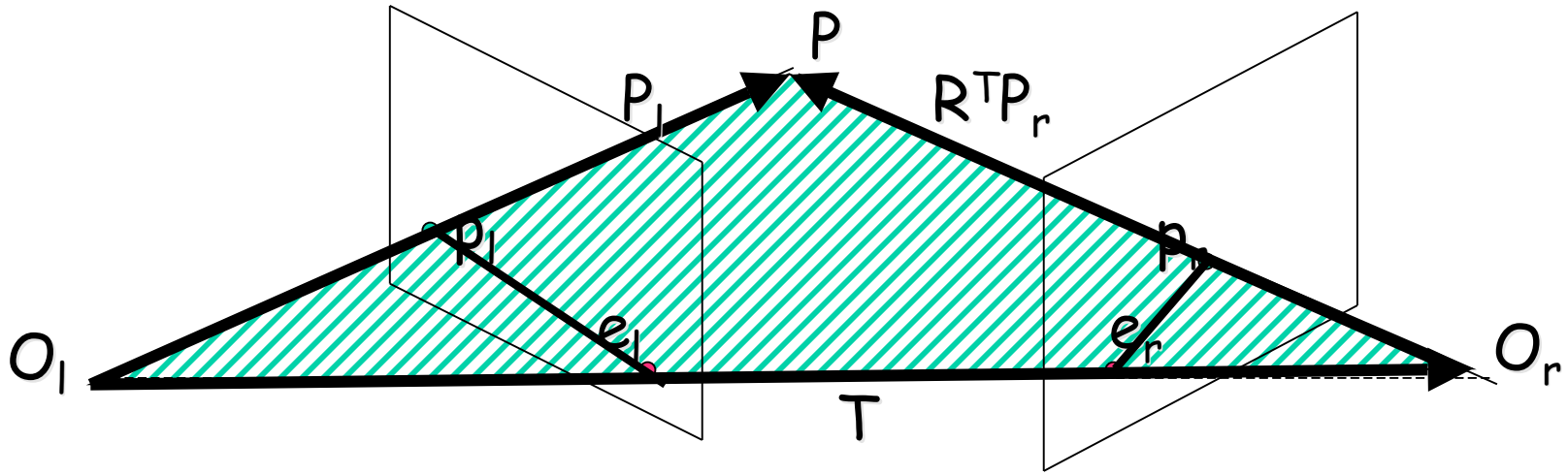
Known intrinsics -- can compute viewing rays in camera coordinate system

Know extrinsics -- know how rays from both cameras are positioned in 3D space

Reconstruction: triangulation of viewing rays

Robert Collins
CSE486, Penn State

Calibrated Triangulation



ideally, P is the point of intersection of two 3D rays:

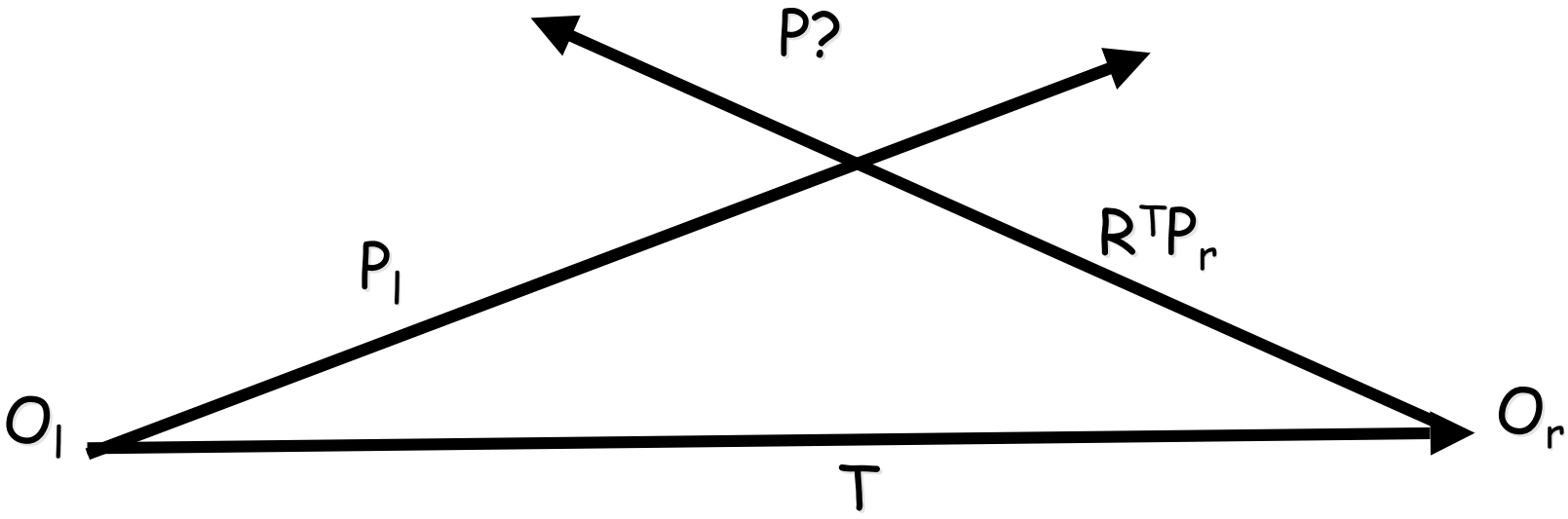
ray through O_l with direction P_l

ray through O_r with direction R^TP_r

Robert Collins
CSE486, Penn State

Triangulation with Noise

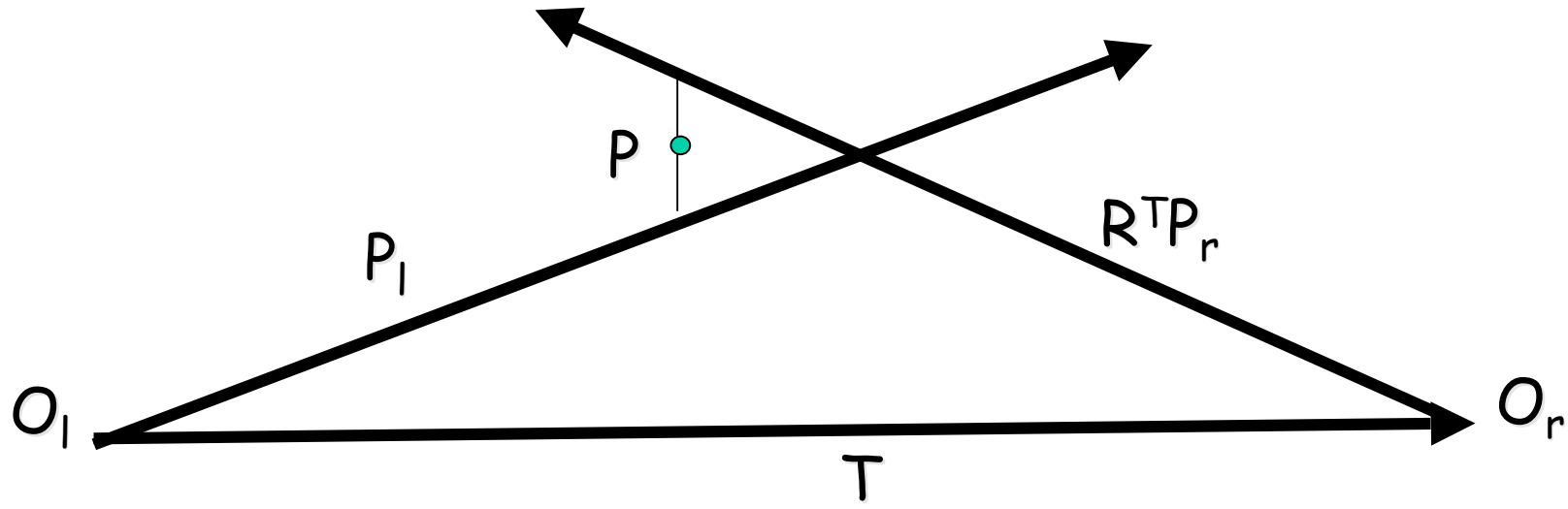
Unfortunately, these rays typically don't intersect due to noise in point locations and calibration params



Robert Collins
CSE486, Penn State

Triangulation with Noise

Unfortunately, these rays typically don't intersect due to noise in point locations and calibration params



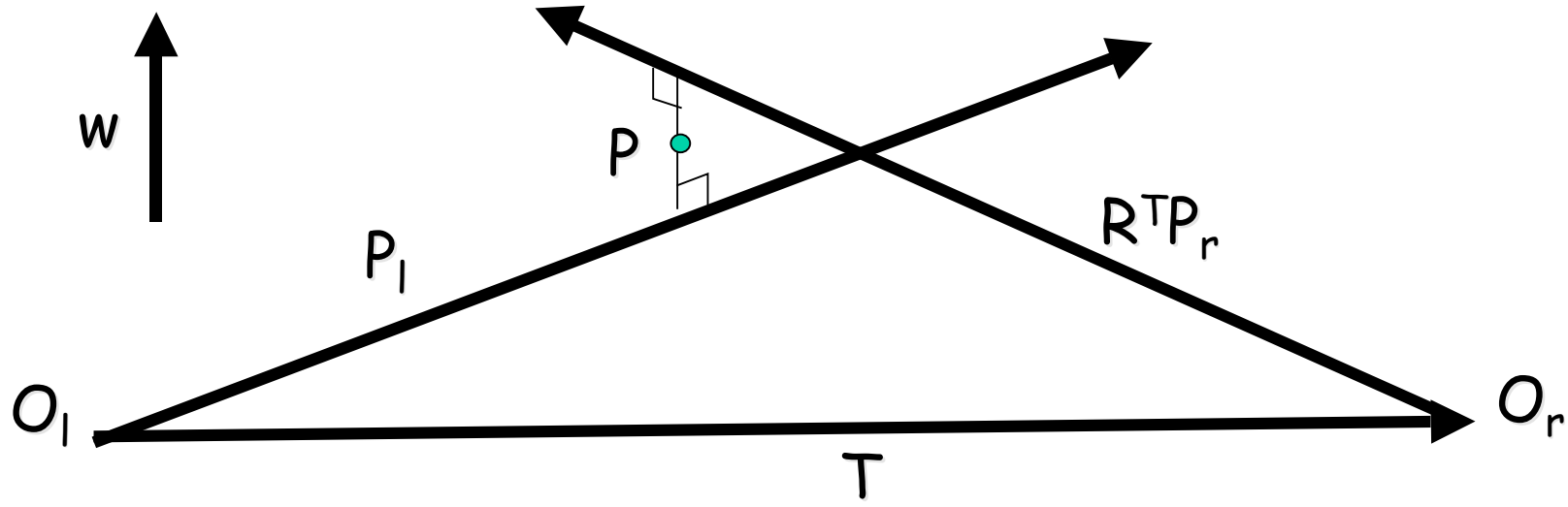
Solution: Choose P as the "pseudo-intersection point". This is point that minimizes the sum of squared distance to both rays. (The SSD is 0 if the rays exactly intersect)

Robert Collins
CSE486, Penn State

Solution from T&V Book

P is midpoint of the segment perpendicular to P_l and $R^T P_r$

Let $w = P_l \times R^T P_r$ (this is perpendicular to both)



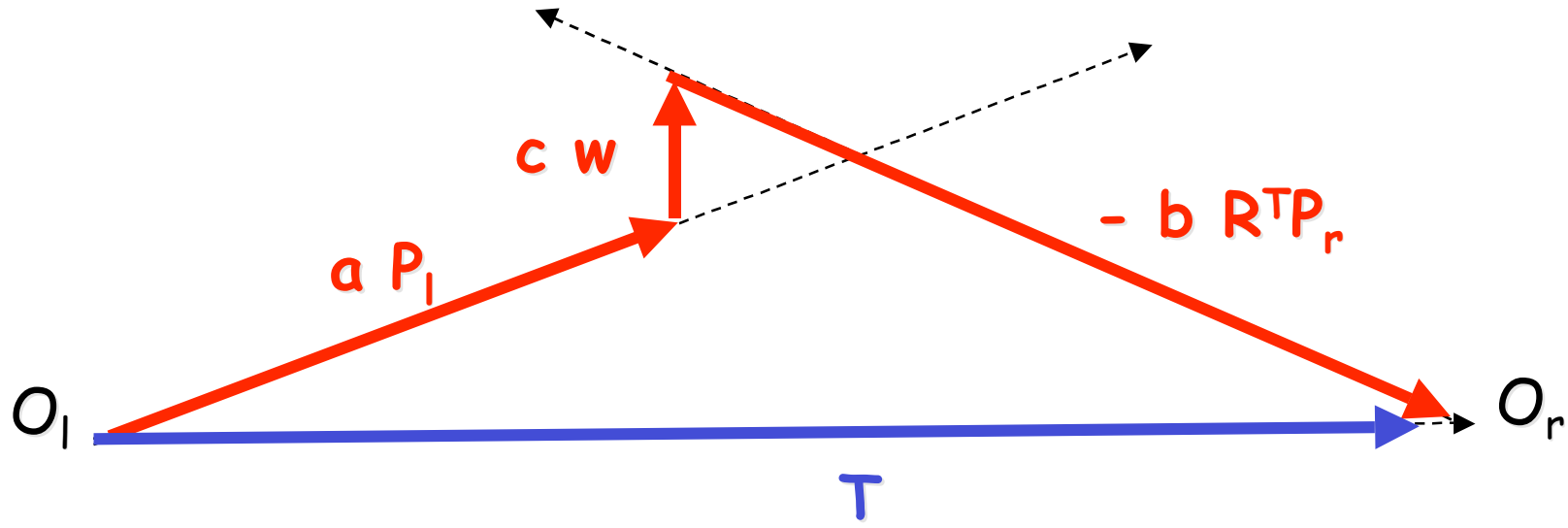
Introducing three unknown scale factors a,b,c we note we can write down the equation of a "circuit"

Robert Collins
CSE486, Penn State

Solution from T&V Book

Writing vector "circuit diagram" with unknowns a,b,c

$$a\,P_l + c\,(P_l X\, R^T P_r) - b\,R^T P_r = T$$

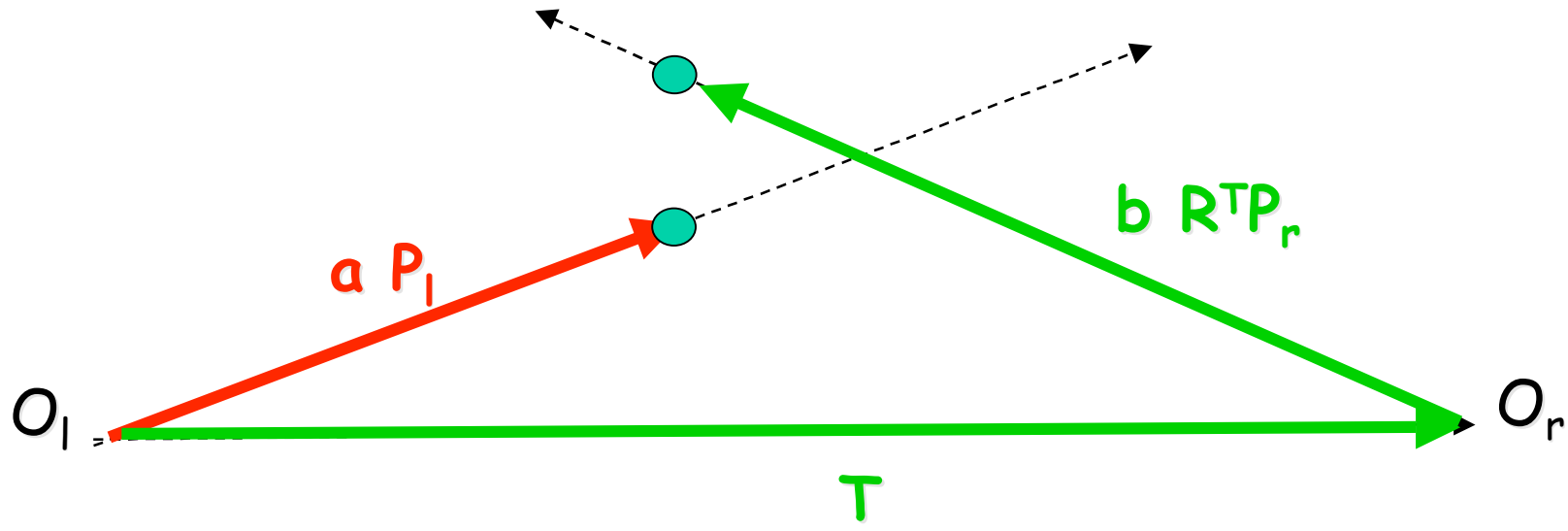


note: this is three linear equations in three unknowns a,b,c
=> can solve for a,b,c

Robert Collins
CSE486, Penn State

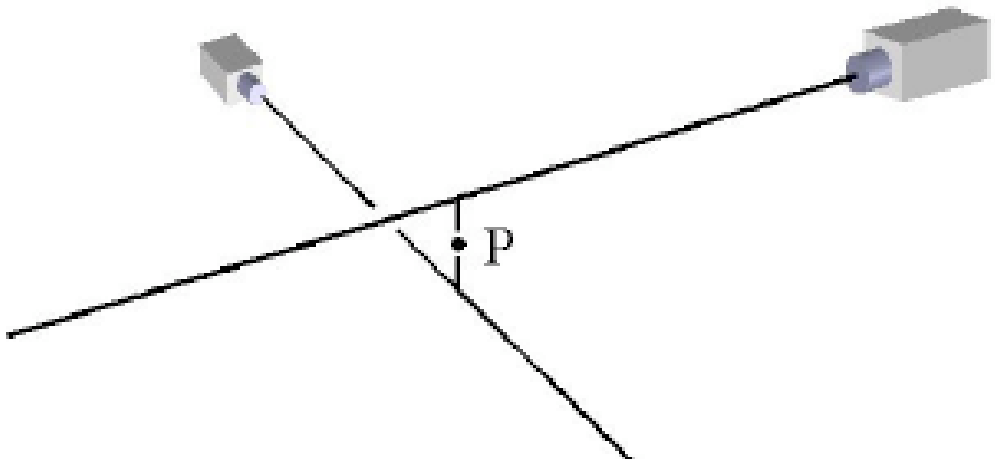
Solution from T&V Book

After finding a,b,c , solve for midpoint of line segment between points $O_l + a P_l$ and $O_l + T + b R^T P_r$



Alternate Solution

I prefer an alternate solution based on using least squares to solve for an unknown point P that minimizes SSD to viewing rays. Why? it generalizes readily to N cameras.



$$[\sum_{i}^{n} w_i(I - u_i u_i')]\, P \;=\; \sum_{i}^{n} w_i(I - u_i u_i')c_i$$

Robert Collins
CSE486, Penn State

Stereo Reconstruction

Given point correspondences, how to compute 3D point positions using triangulation.

Results depend on how calibrated the system is:

1) Intrinsic and extrinsic parameters known
   Can compute metric 3D geometry
2) Only intrinsic parameters known
   Unknown scale factor
3) Neither intrinsic nor extrinsic known
   Recover structure up to an unknown projective transformation of the scene

Only Intrinsic Params Known

General outline of solution (see book for details)

Use knowledge that E = R S to solve for R and T, then use previous triangulation method.

Note: since E is only defined up to a scale factor, we can only determine the direction of T, not its length. So... 3D reconstruction will have an unknown scale.

Only Intrinsic Params Known

Using E to solve for extrinsic params R and T

$E = R\ S$ where elements of S are functions of T

Then $E^T\ E = S^T\ R^T\ R\ S = S^T\ S$ (because $R^T R = I$)

Thus $E^T\ E$ is only a function of T.
Solve for elements of T assuming it is a unit vector.

After determining T, plug back into $E = R\ S$ to determine R.

Only Intrinsic Params Known

Unfortunately, four different solutions for (R,T) are possible (due to choice of sign of E, and choice of sign of T when solving for it).

However, only one choice will give consistent solutions when used for triangulation, where consistent means reconstructed points are in front of the cameras (positive Z coordinates).

So, check all four solutions, choose the correct one, and you are done.

Robert Collins
CSE486, Penn State

Stereo Reconstruction

Given point correspondences, how to compute 3D point positions using triangulation.

Results depend on how calibrated the system is:

1) Intrinsic and extrinsic parameters known
   Can compute metric 3D geometry
2) Only intrinsic parameters known
   Unknown scale factor
3) Neither intrinsic nor extrinsic known
   Recover structure up to an unknown projective transformation of the scene

Stereo when "Nothing" is Known

What if we don't known intrinsic nor extrinsic params? (we just look at two pictures of the scene with no prior information)

Can we recover any 3D information?

It wasn't clear that you could, but then in 1992...

- Faugeras "What can be seen in three dimensions from an uncalibrated stereo rig", ECCV 1992
- Hartley et.al., "Stereo from Uncalibrated Cameras", CVPR 1992
- Mohr et.al., "Relative 3D Reconstruction using Multiple Uncalibrated Images, LIFIA technical report, 1992

Stereo when "Nothing" is Known

Of course, we don't really know "nothing".

We know point correspondences, and because of that, we can compute the fundamental matrix F

Sketch of solution: use knowledge of F and use 5 points in the scene to define an arbitrary projective coordinate system. These points will have coordinates:

(1 0 0 0) (0 1 0 0) (0 0 1 0) (0 0 0 1) (1 1 1 1)

Result: You can recover 3D locations of other points with respect to that projective coordinate system.

Stereo when "Nothing" is Known

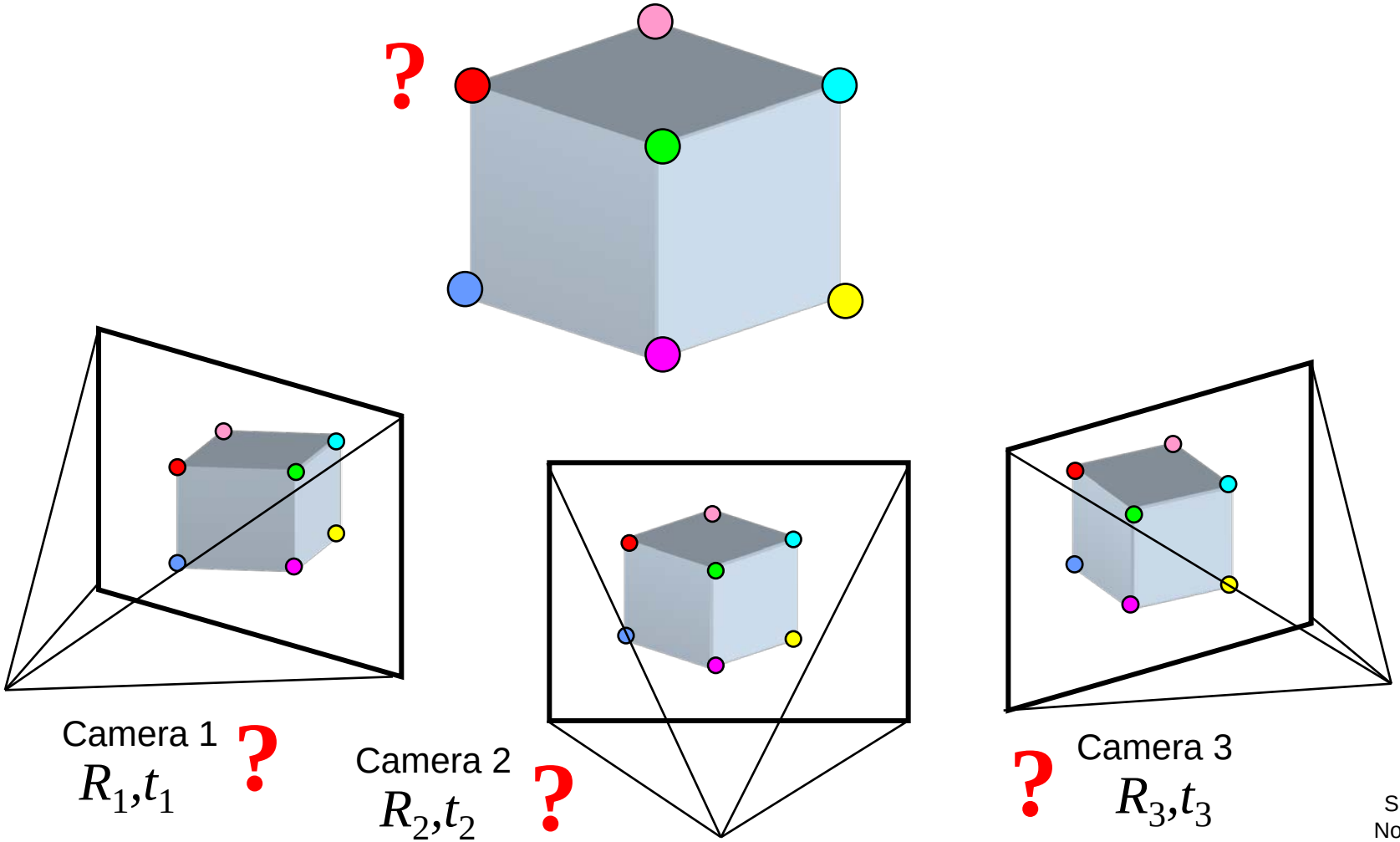
Result: You can recover 3D locations of other points with respect to that projective coordinate system.

Why would that be practical?

Often, you can then use other information to determine how your arbitrary projective coordinate system relates to the "real" Euclidean scene coordinates system.
e.g. use prior knowledge of lengths and angles
of some items in the world (like a house)

Structure from motion

- Given a set of corresponding points in two or more images, compute the camera parameters and the 3D point coordinates



Slide credit: Noah Snavely

Overview: Structure from motion

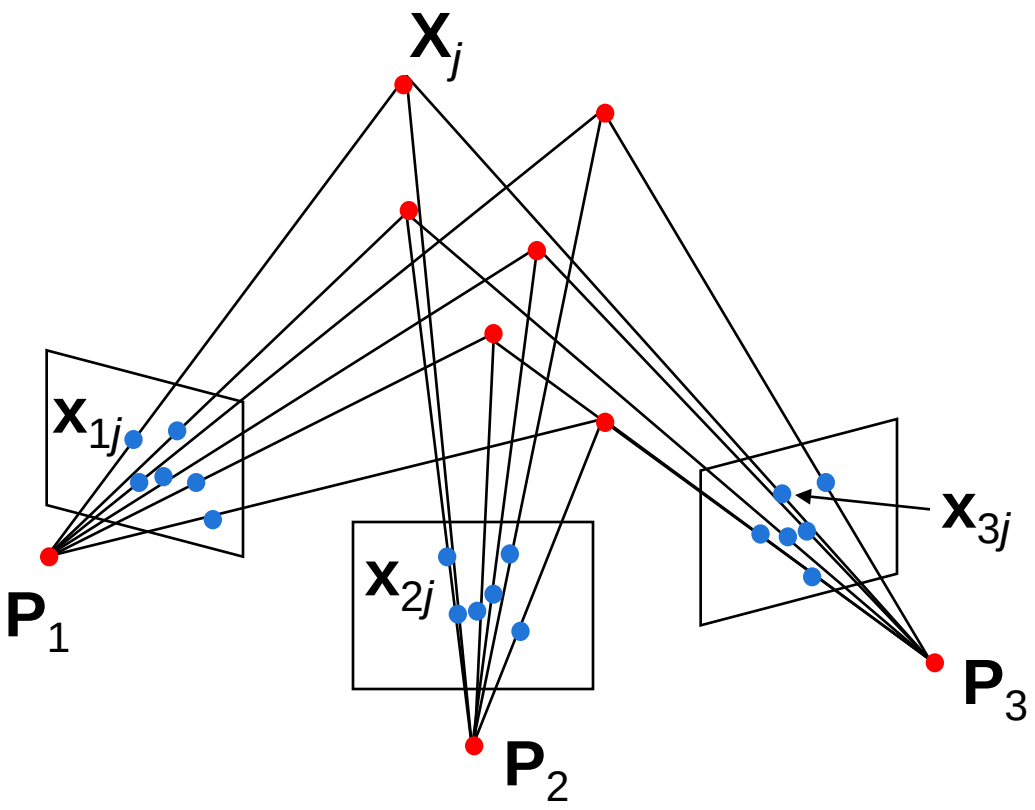
- Ambiguity
- Affine structure from motion
  - Factorization
- Dealing with missing data
  - Incremental structure from motion
- Projective structure from motion
  - Bundle adjustment
  - Self-calibration

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* 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!

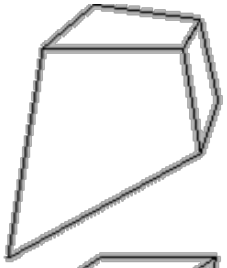
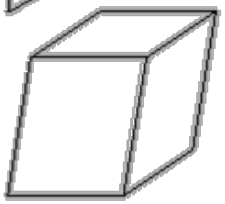
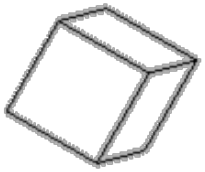
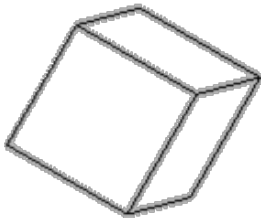
Structure from motion ambiguity

- 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:

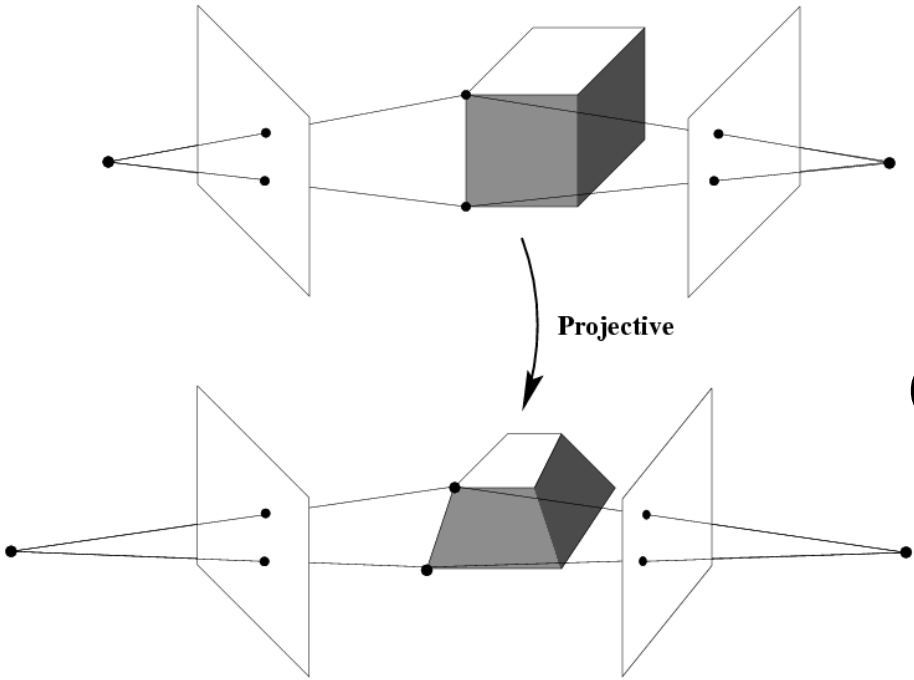
$$\mathbf{x} = \mathbf{PX} = \left(\mathbf{PQ}^{-1}\right)\left(\mathbf{QX}\right)$$

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

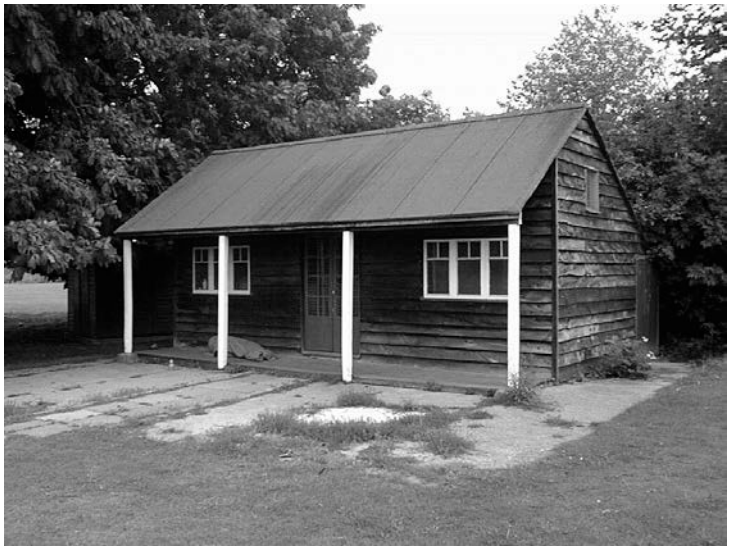
Projective ambiguity

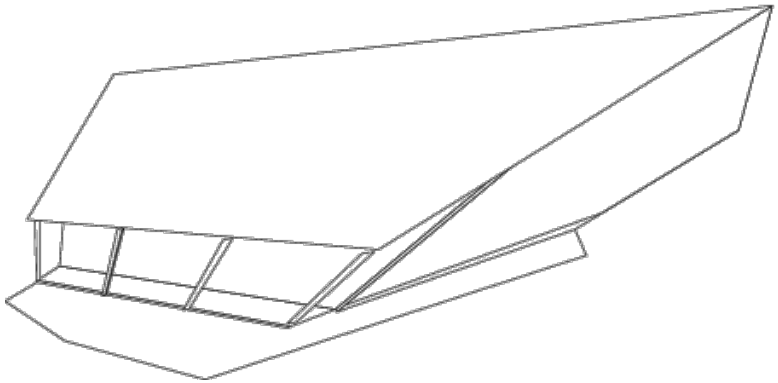


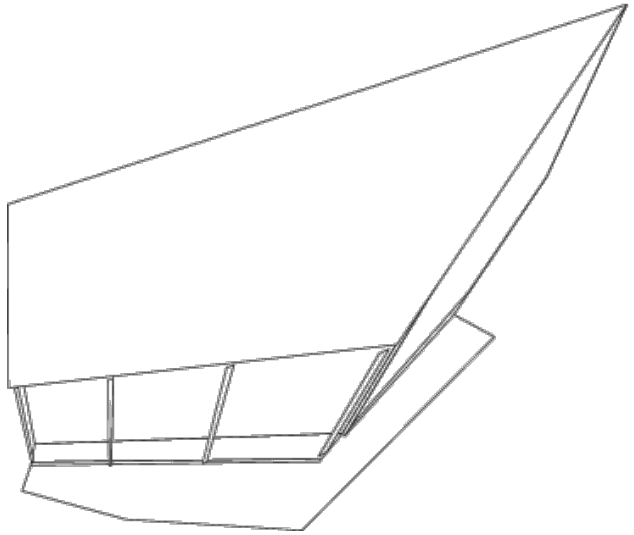
$$\mathbf{Q_p} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ \mathrm{v}^{\mathsf{T}} & v \end{bmatrix}$$

$$\mathbf{x} = \mathbf{PX} = \left(\mathbf{PQ_P^{-1}}\right)\left(\mathbf{Q_P\,X}\right)$$

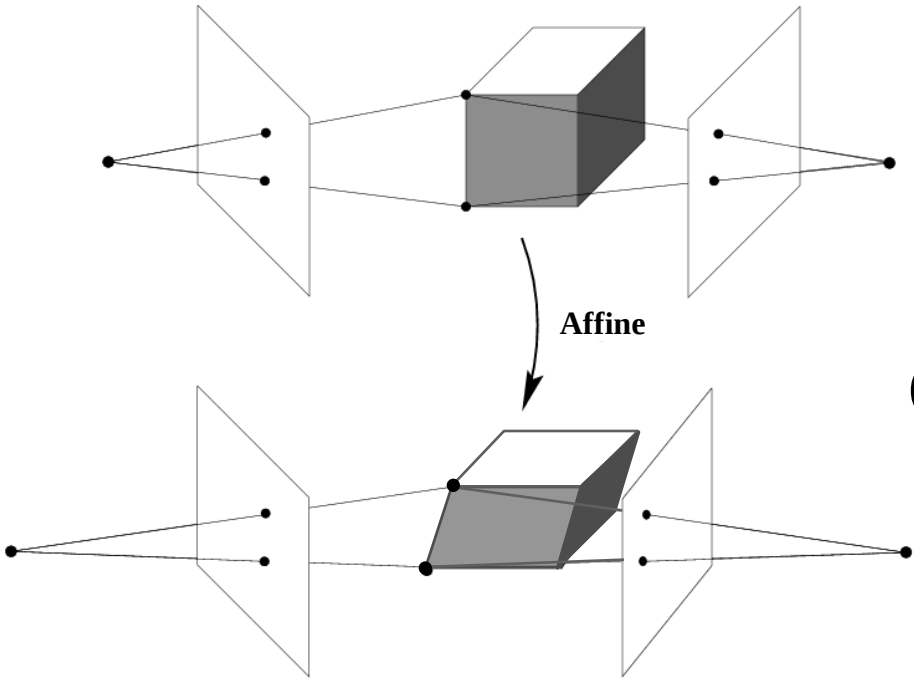
Projective ambiguity







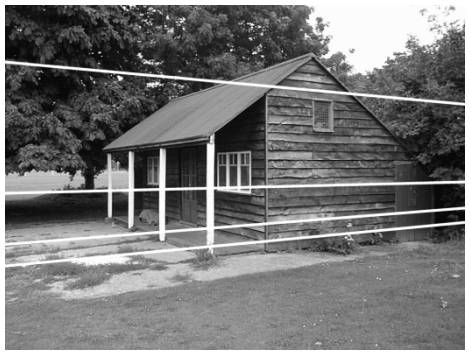
Affine ambiguity

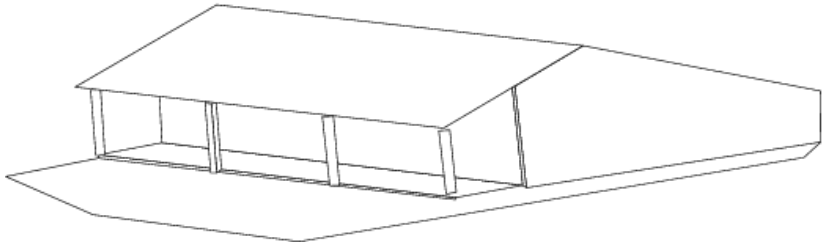


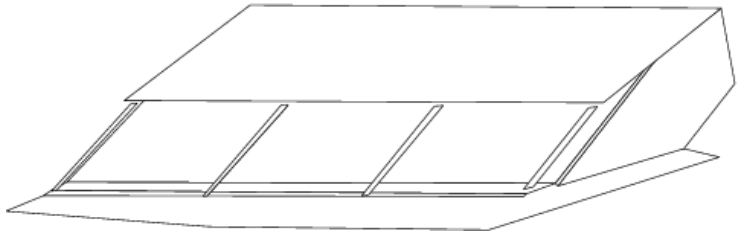
$$\mathbf{Q_A} = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{PX} = \left(\mathbf{PQ_A^{-1}}\right)\left(\mathbf{Q_A X}\right)$$

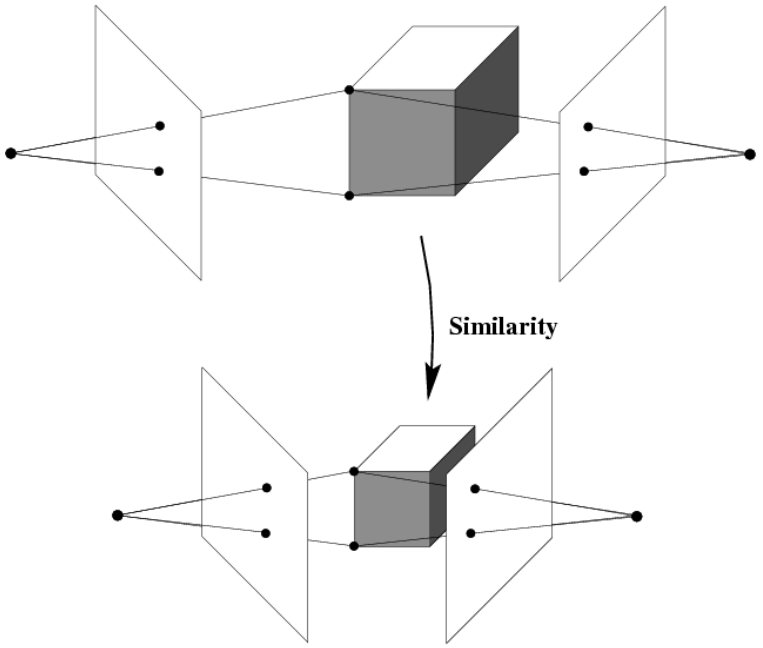
Affine ambiguity







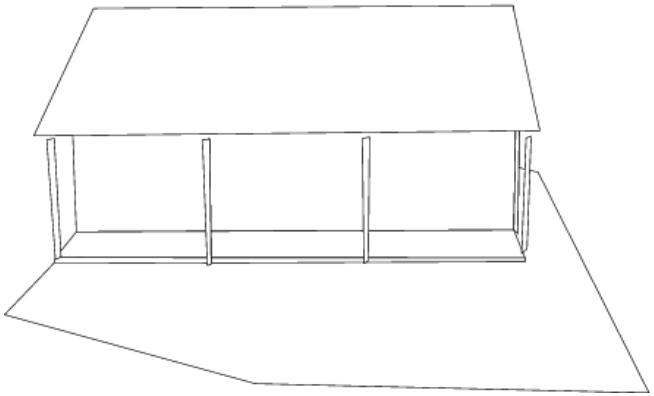
Similarity ambiguity

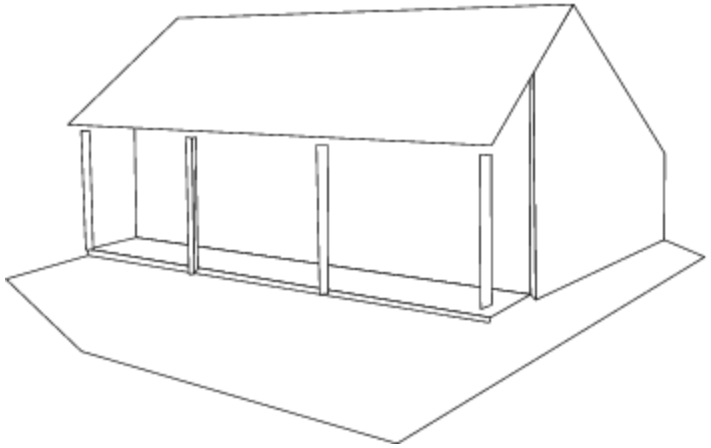


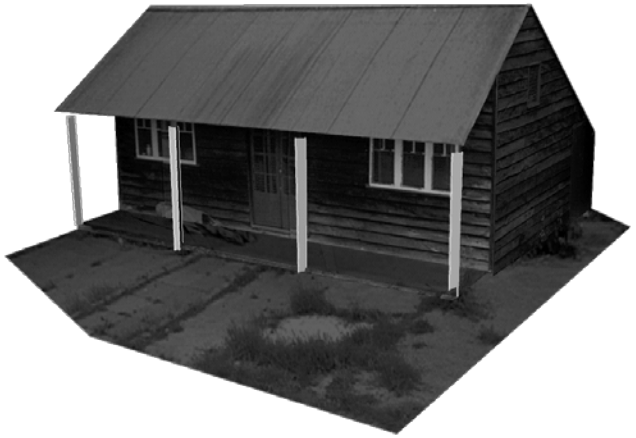
$$\mathbf{Q}_s = \begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$$

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

Similarity ambiguity

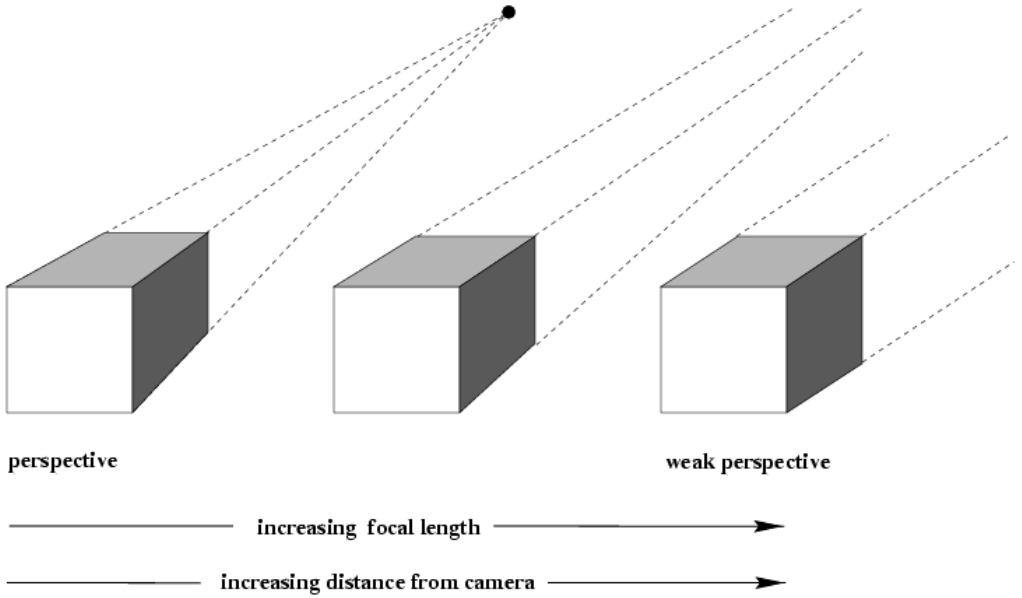






Structure from motion

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

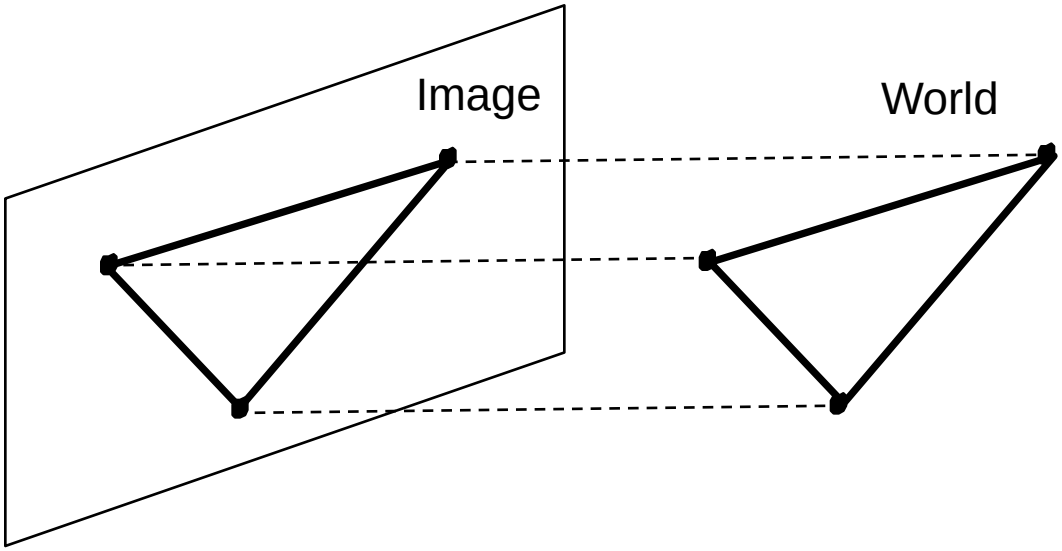


center at infinity

Recall: Orthographic Projection

Special case of perspective projection

- Distance from center of projection to image plane is infinite



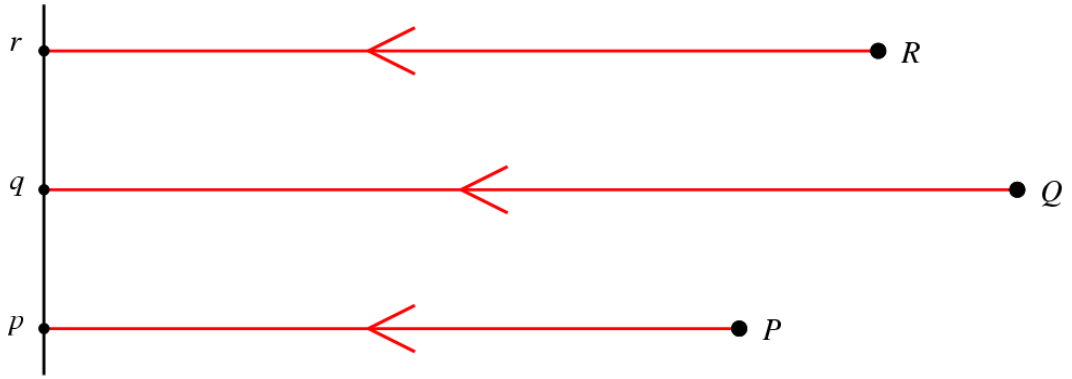
- Projection matrix:

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

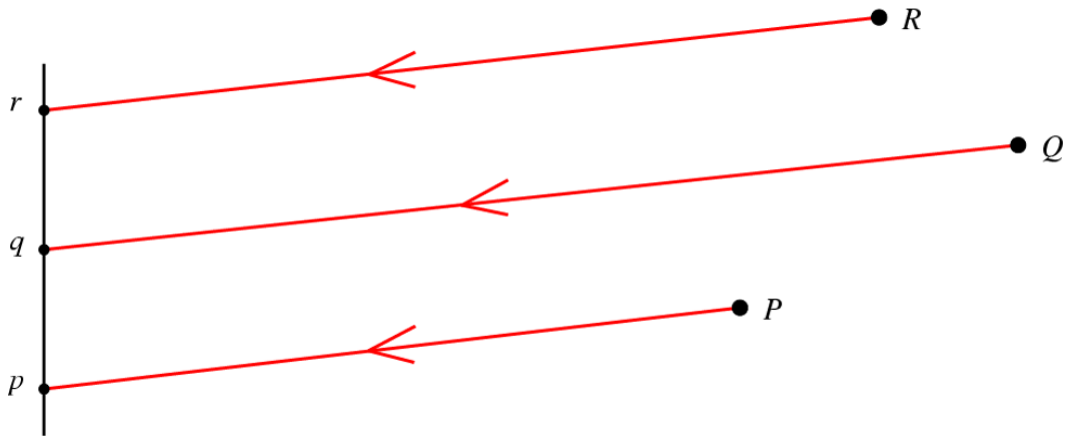
Slide by Steve Seitz

Affine cameras

Orthographic Projection



Parallel Projection

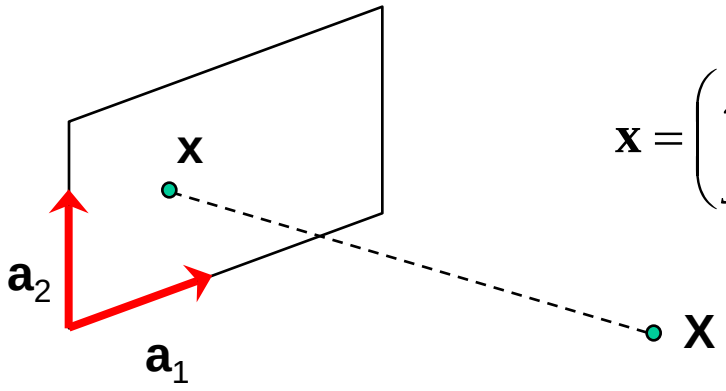


Affine cameras

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

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

- 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:

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

- 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:

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

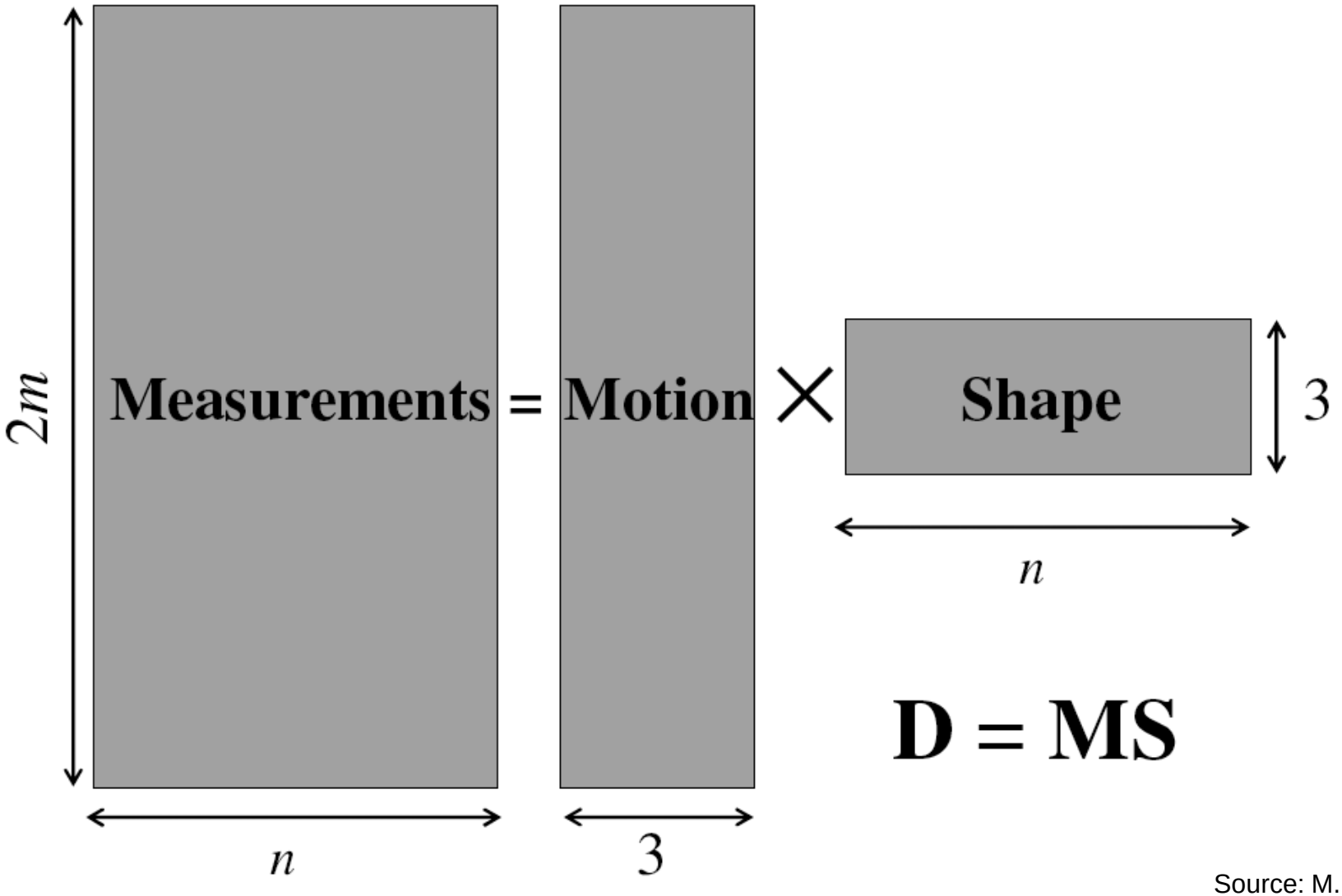
points ($3 \times n$)

cameras ($2m \times 3$)

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.

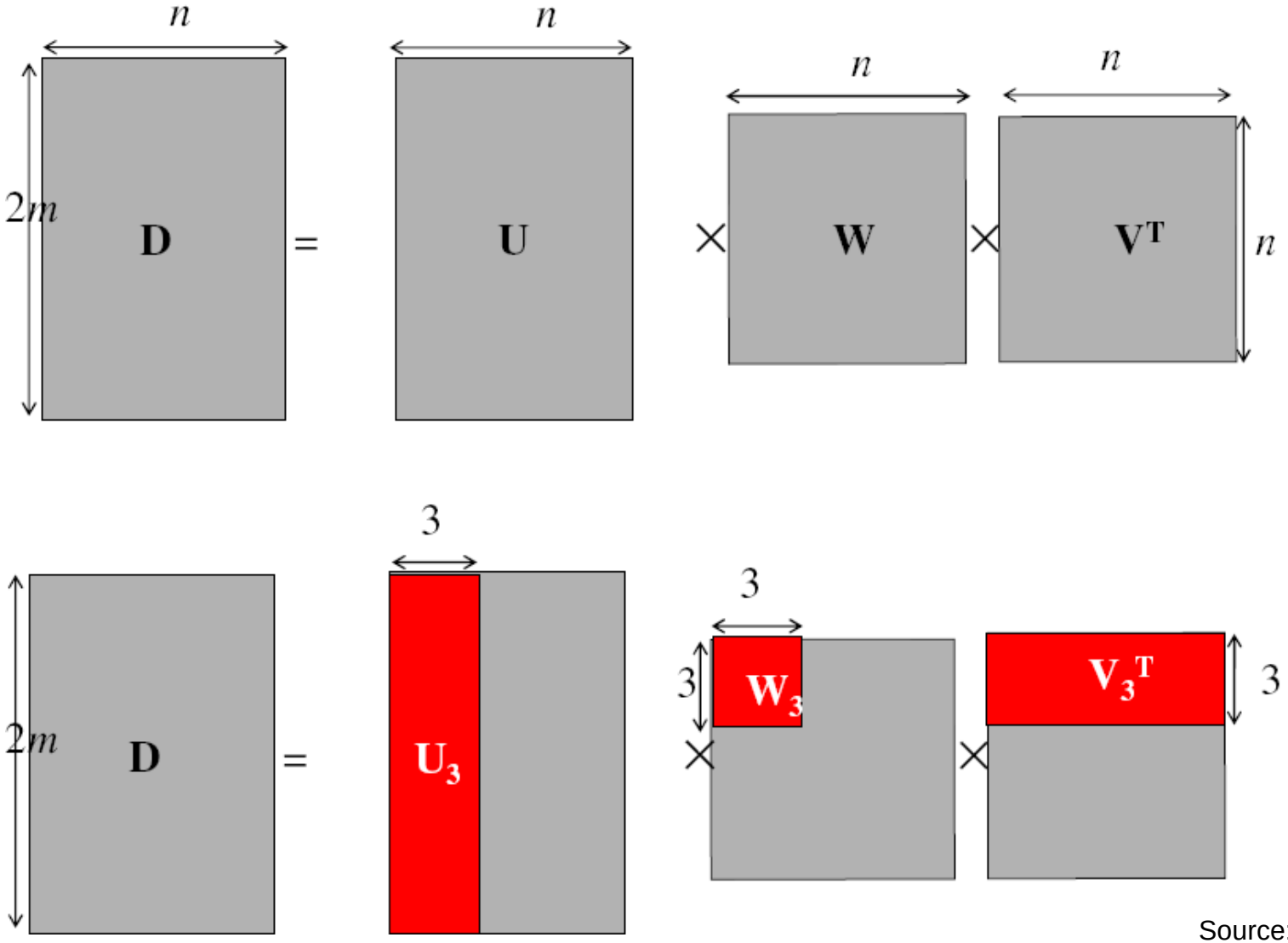
Factorizing the measurement matrix



Source: M. Hebert

Factorizing the measurement matrix

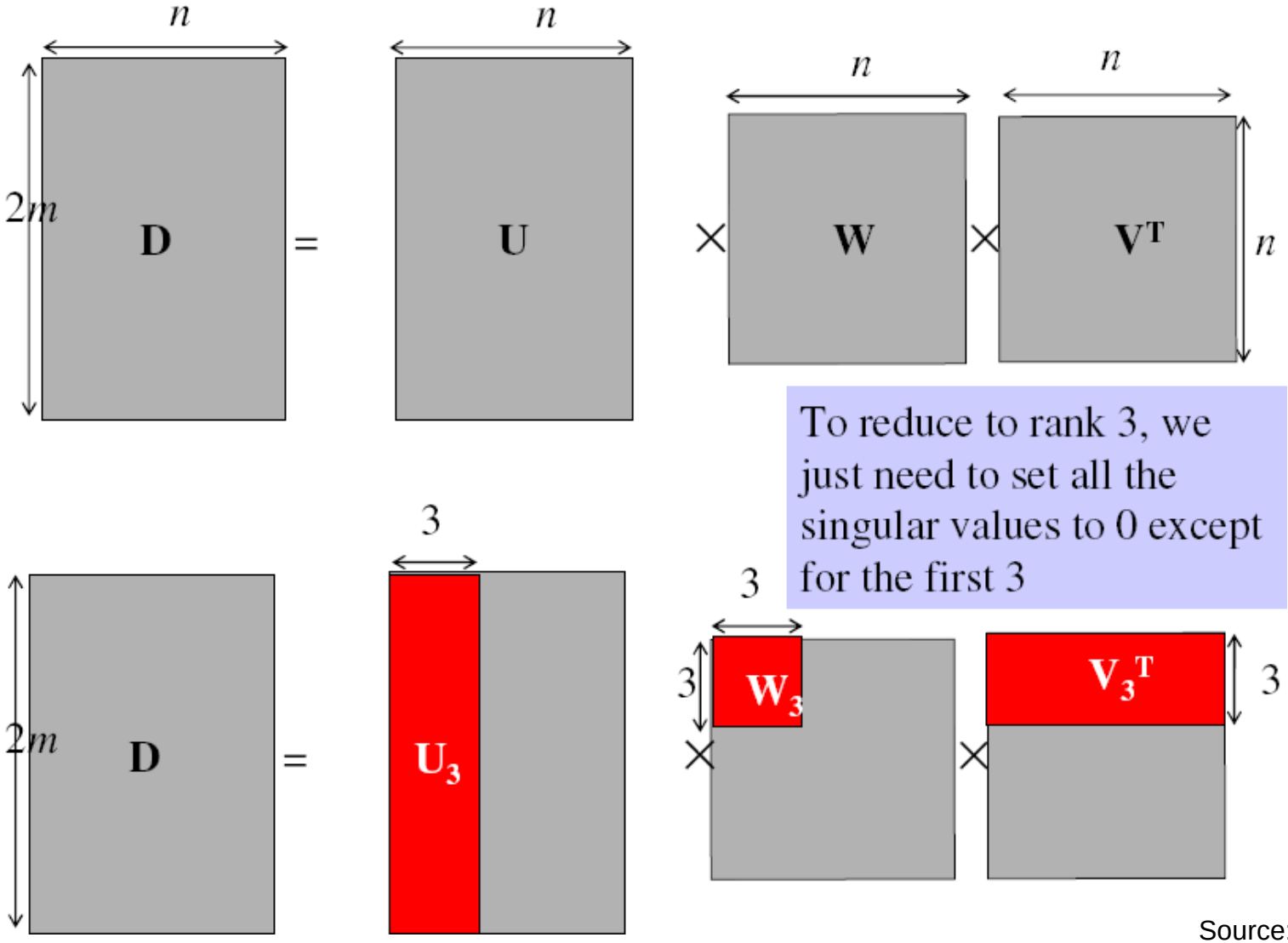
- Singular value decomposition of D:



Source: M. Hebert

Factorizing the measurement matrix

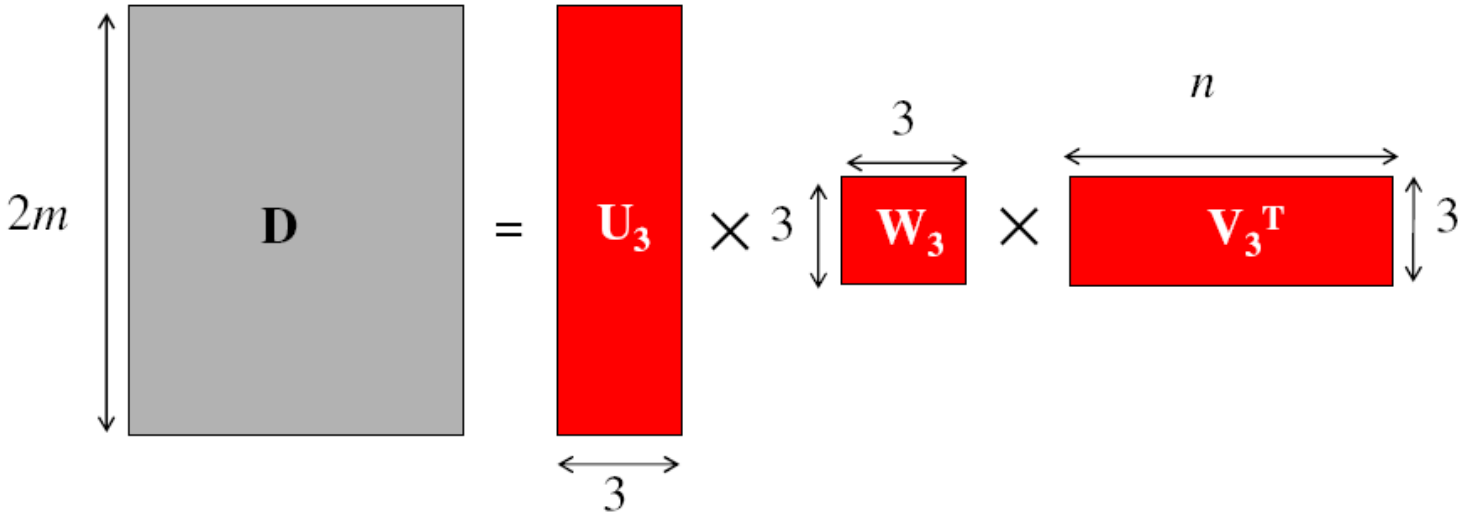
- Singular value decomposition of D:



Source: M. Hebert

Factorizing the measurement matrix

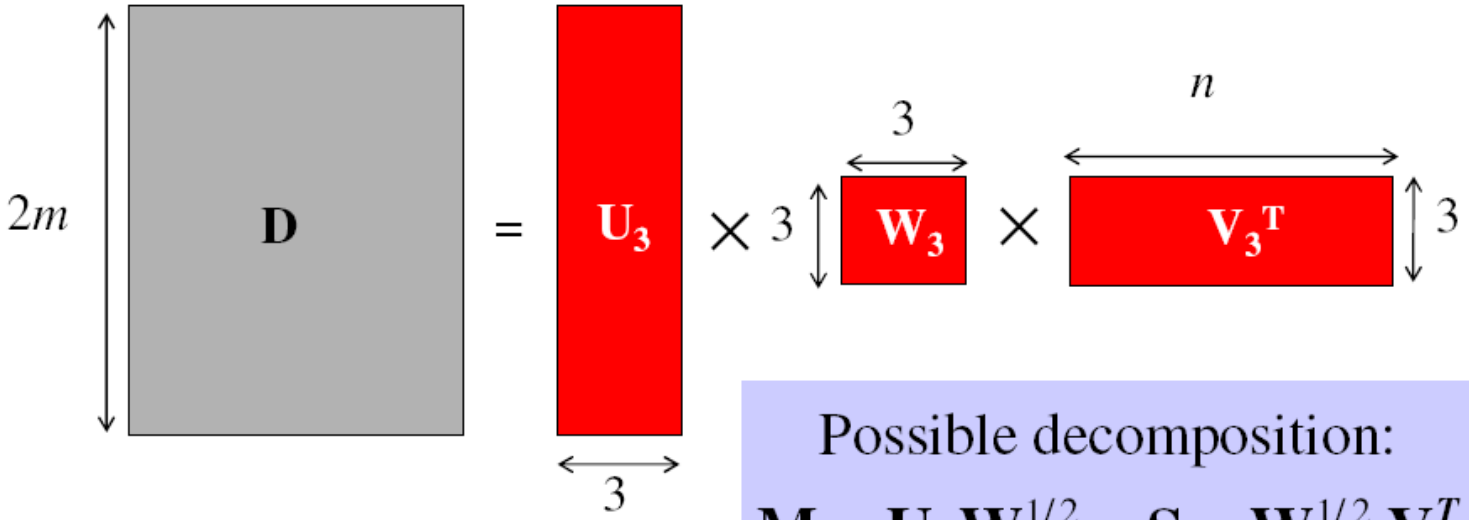
- Obtaining a factorization from SVD:



Source: M. Hebert

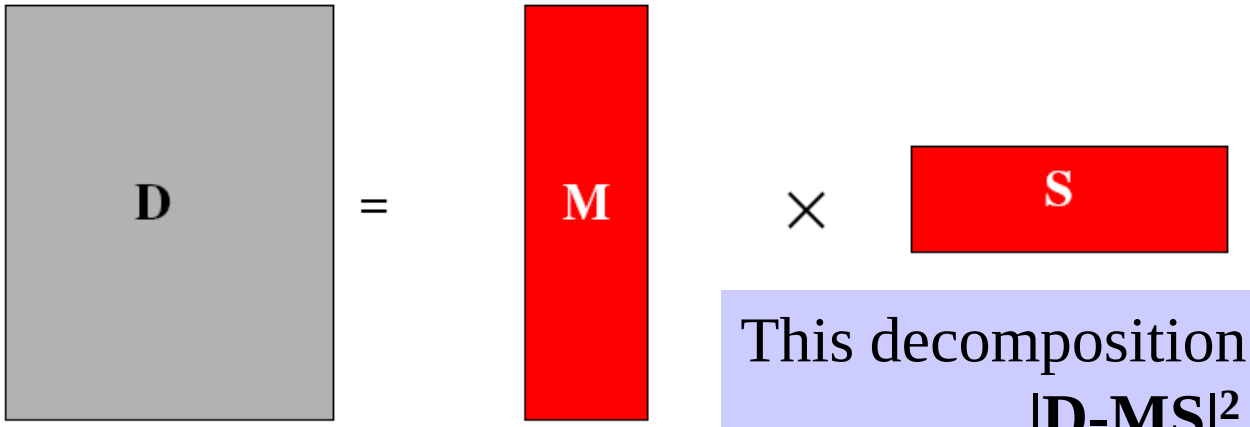
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



Possible decomposition:

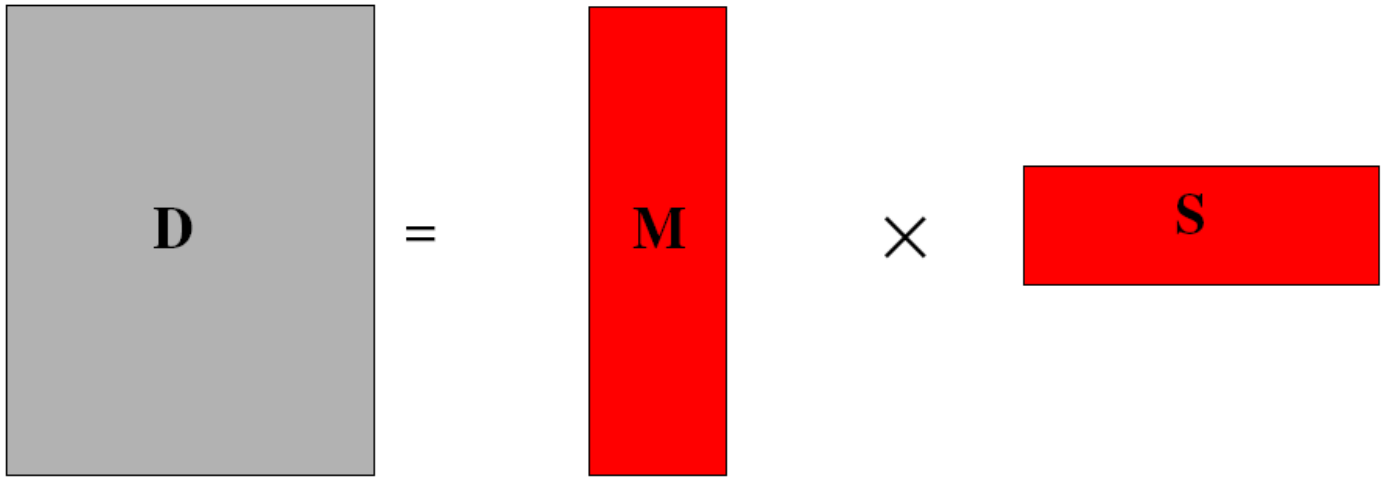
$$\mathbf{M} = \mathbf{U}_3 \mathbf{W}_3^{1/2} \qquad \mathbf{S} = \mathbf{W}_3^{1/2} \mathbf{V}_3^T$$



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

Source: M. Hebert

Affine ambiguity

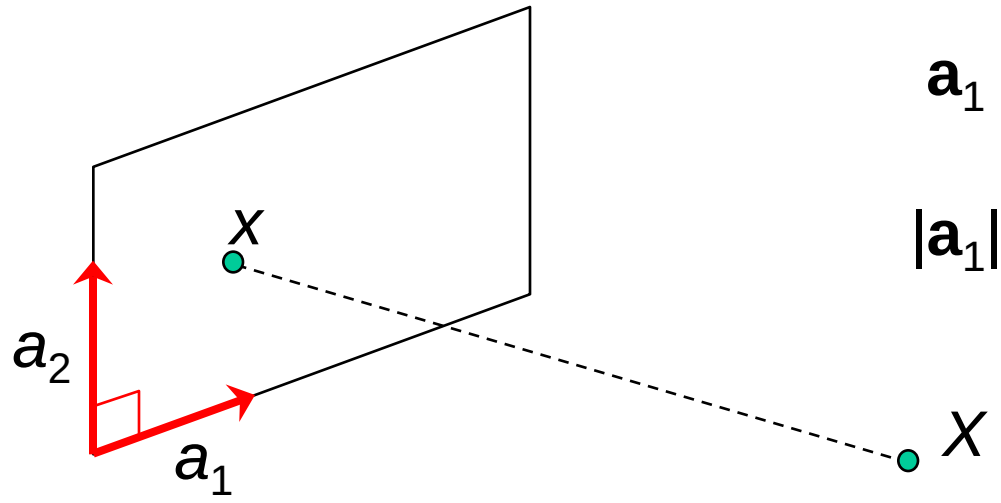


- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations **M** → **MC, S** →$\mathbf{C^{-1}S}$
- That is because we have only an affine transformation and we have not enforced any Euclidean constraints (like forcing the image axes to be perpendicular, for example)

Source: M. Hebert

Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and scale is 1



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

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

- This translates into 3*m* equations in $\mathbf{L} = \mathbf{C}\mathbf{C}^T$:

$$\mathbf{A}_i \mathbf{L} \mathbf{A}_i^T = \mathbf{Id}, \qquad i = 1, \ldots, m$$

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

Source: M. Hebert

Algorithm summary

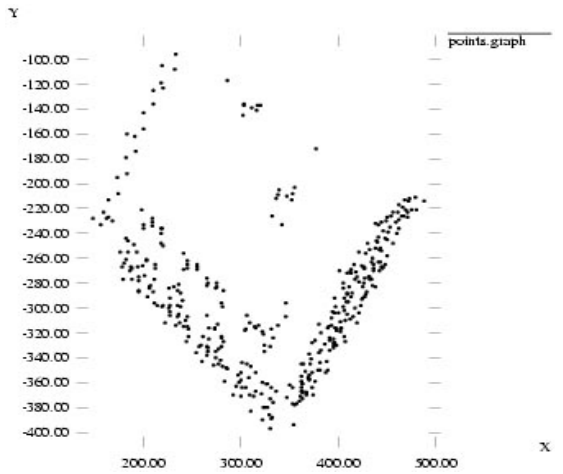
- Given: *m* images and *n* features $\mathbf{x}_{ij}$
- For each image *i, c*enter the feature coordinates
- Construct a $2m \times n$ measurement matrix **D**:
  - Column *j* contains the projection of point *j* in all views
  - Row *i* contains one coordinate of the projections of all the *n* points 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 $\mathbf{S} = \mathbf{W}_3^{½}\,\mathbf{V}_3^{T}$ (**or** $\mathbf{M} = \mathbf{U}_3$ and $\mathbf{S} = \mathbf{W}_3\mathbf{V}_3^{T}$)
- Eliminate affine ambiguity

Source: M. Hebert

Reconstruction results

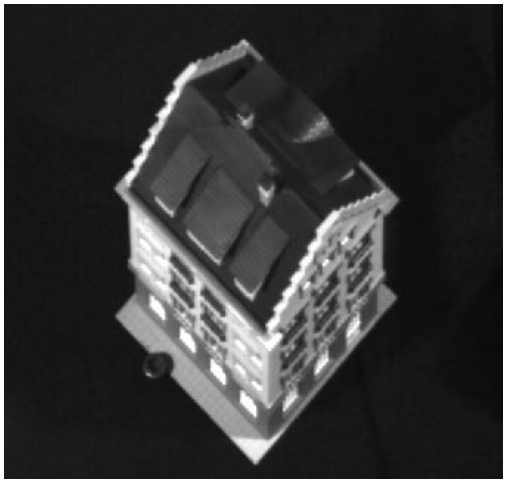
1

60



120

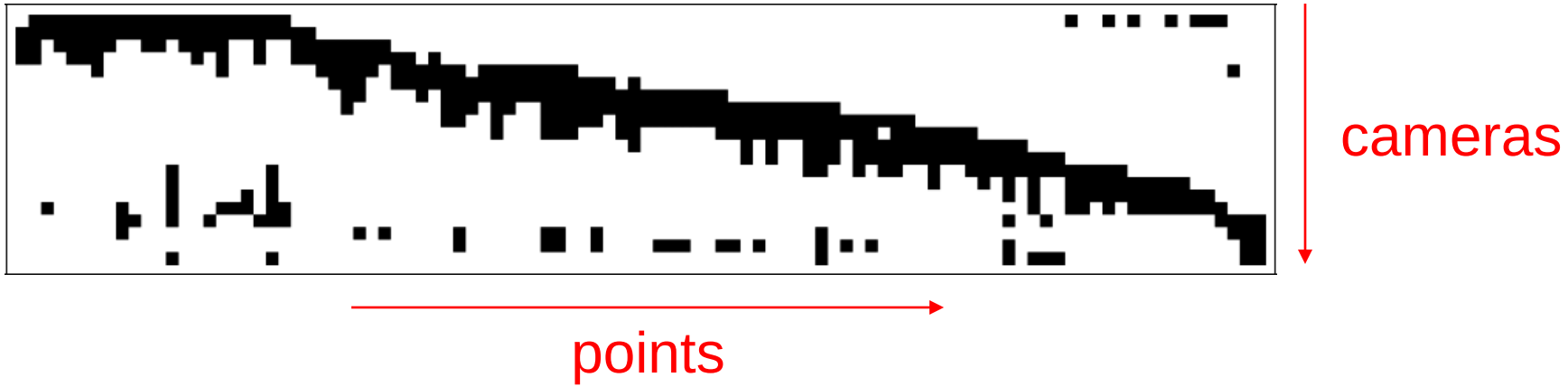
150



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

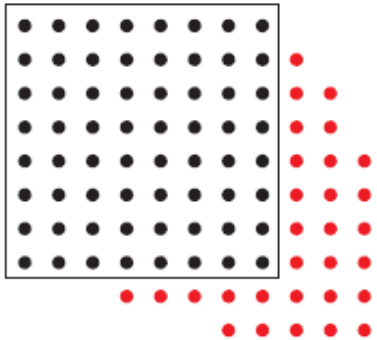
Dealing with missing data

- So far, we have assumed that all points are visible in all views
- In reality, the measurement matrix typically looks something like this:

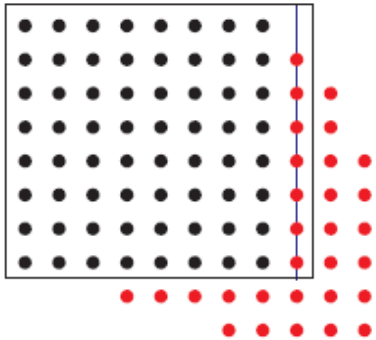


Dealing with missing data

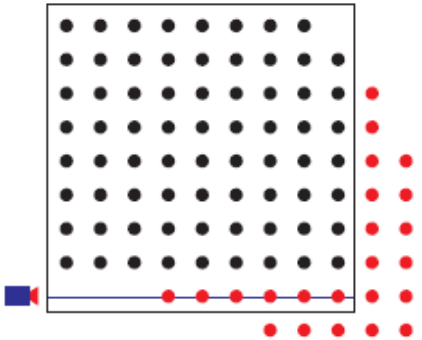
- Possible solution: decompose matrix into dense sub-blocks, factorize each sub-block, and fuse the results
  - Finding dense maximal sub-blocks of the matrix is NP-complete (equivalent to finding maximal cliques in a graph)
- Incremental bilinear refinement



(1) Perform factorization on a dense sub-block



(2) Solve for a new 3D point visible by at least two known cameras (linear least squares)



(3) Solve for a new camera that sees at least three known 3D points (linear least squares)

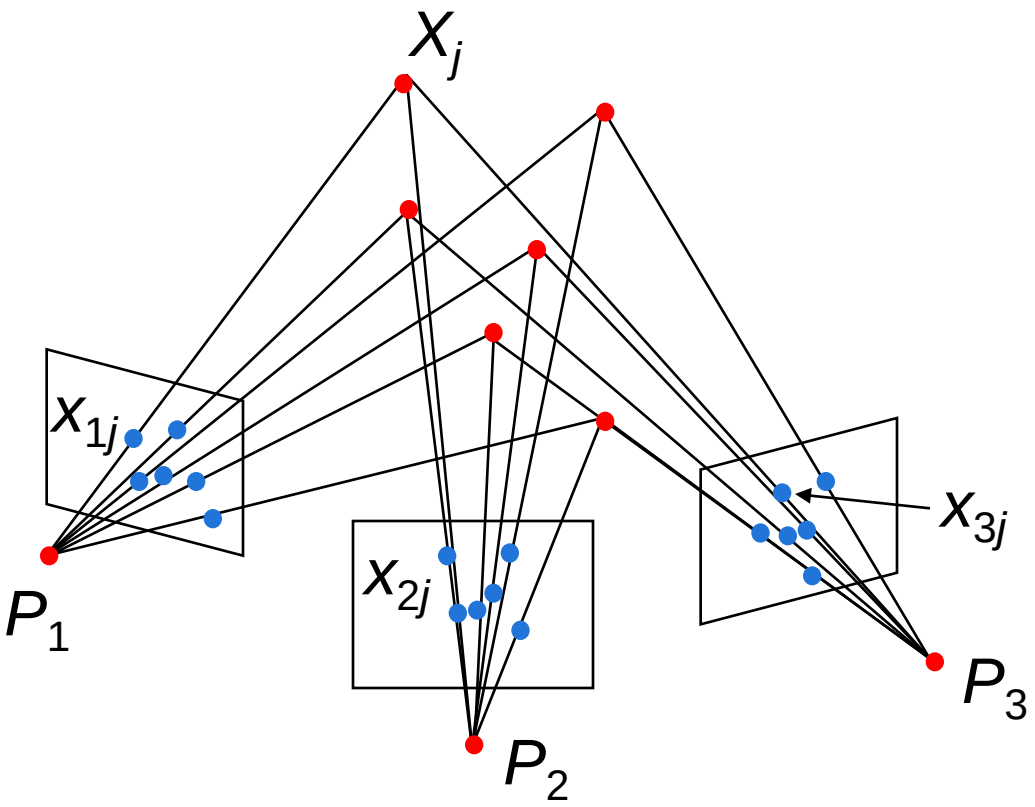
F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, Modeling, and Matching Video Clips Containing Multiple Moving Objects. PAMI 2007.

Projective structure from motion

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

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

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



Projective structure from motion

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

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

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

$$2mn >= 11m + 3n - 15$$

- For two cameras, at least 7 points are needed

Projective SFM: Two-camera case

- Compute fundamental matrix $\mathbf{F}$ between the two views
- First camera matrix: $\mathbf{[I|0]}$
- Second camera matrix: $\mathbf{[A|b]}$
- Then $\mathbf{b}$ is the epipole ($\mathbf{F}^T\mathbf{b} = 0$), $\mathbf{A} = -[\mathbf{b}_\times]\mathbf{F}$

F&P sec. 8.3.2

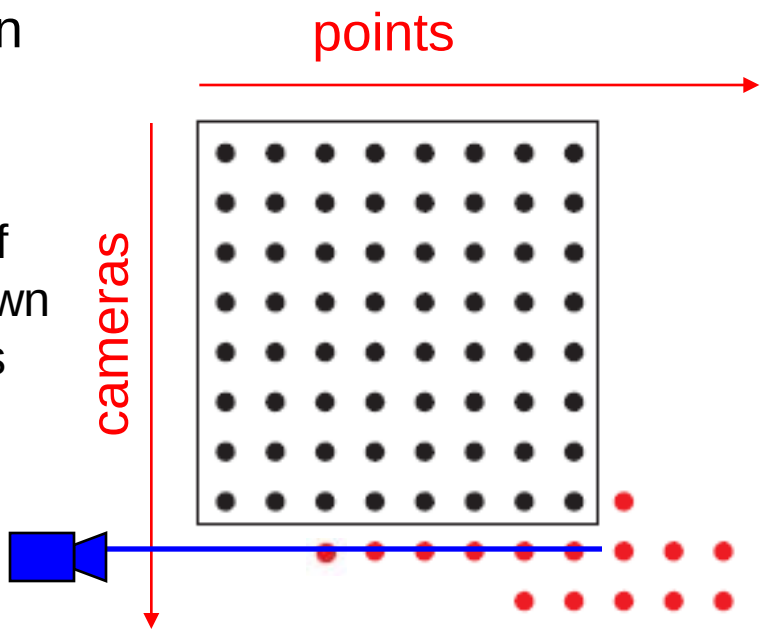
Sequential structure from motion

•Initialize motion from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

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



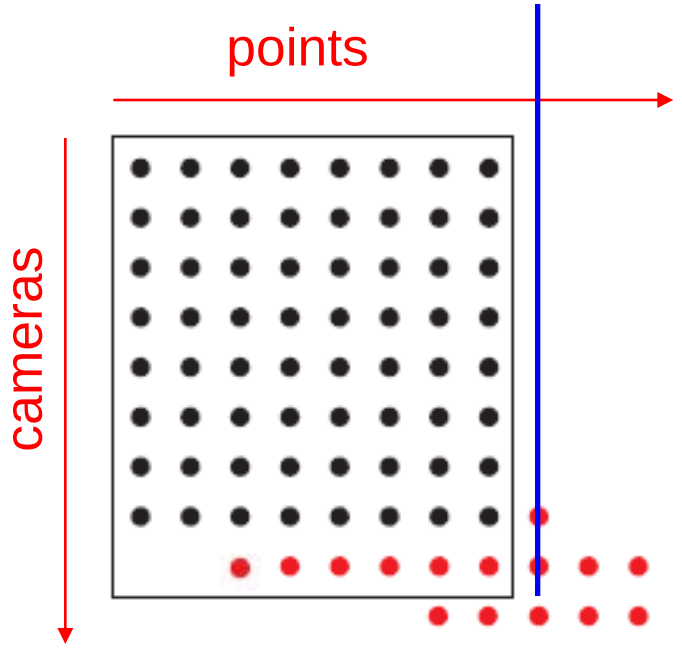
Sequential structure from motion

•Initialize motion from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
- Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*



Sequential structure from motion

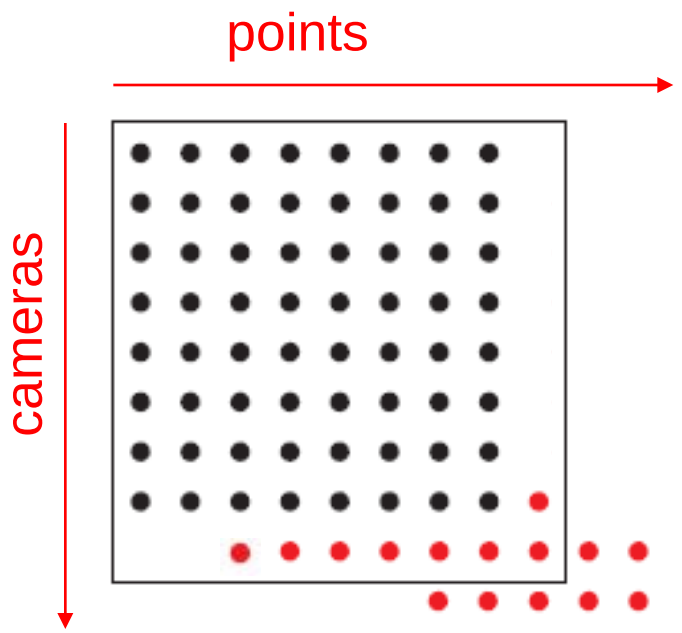
•Initialize motion from two images using fundamental matrix

•Initialize structure by triangulation

•For each additional view:

- Determine projection matrix of new camera using all the known 3D points that are visible in its image – *calibration*
- Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera – *triangulation*

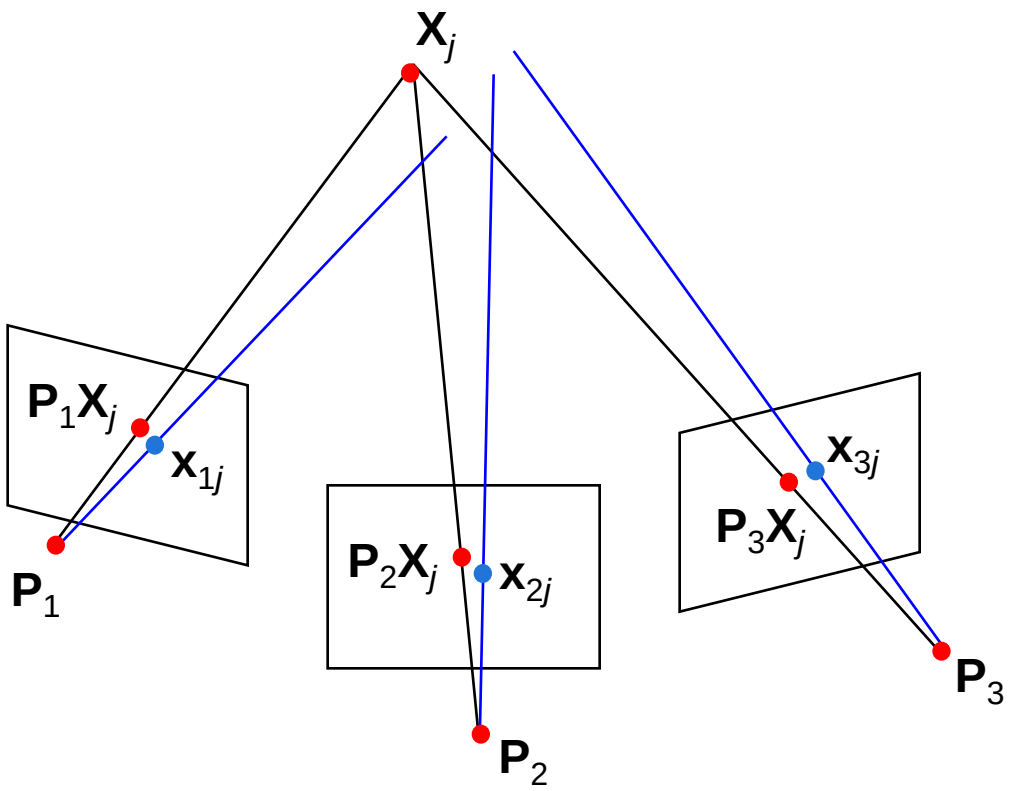
•Refine structure and motion: bundle adjustment



Bundle adjustment

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

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



Self-calibration

- Self-calibration (auto-calibration) is the process of determining intrinsic camera parameters directly from uncalibrated images
- For example, when the images are acquired by a single moving camera, we can use the constraint that the intrinsic parameter matrix remains fixed for all the images
  - 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: zero skew
- Can use vanishing points

Review: Structure from motion

- Ambiguity
- Affine structure from motion
  - Factorization
- Dealing with missing data
  - Incremental structure from motion
- Projective structure from motion
  - Bundle adjustment
  - Self-calibration