Lecture 6: Multi-view Stereo & Structure from Motion

Prof. Rob Fergus

Many slides adapted from Lana Lazebnik and Noah Snavelly, who in turn adapted slides from Steve Seitz, Rick Szeliski, Martial Hebert, Mark Pollefeys, and others….

Overview

- Multi-view stereo
- Structure from Motion (SfM)
- Large scale Structure from Motion
- Kinect Fusion
- Dynamic Fusion

Multi-view Stereo

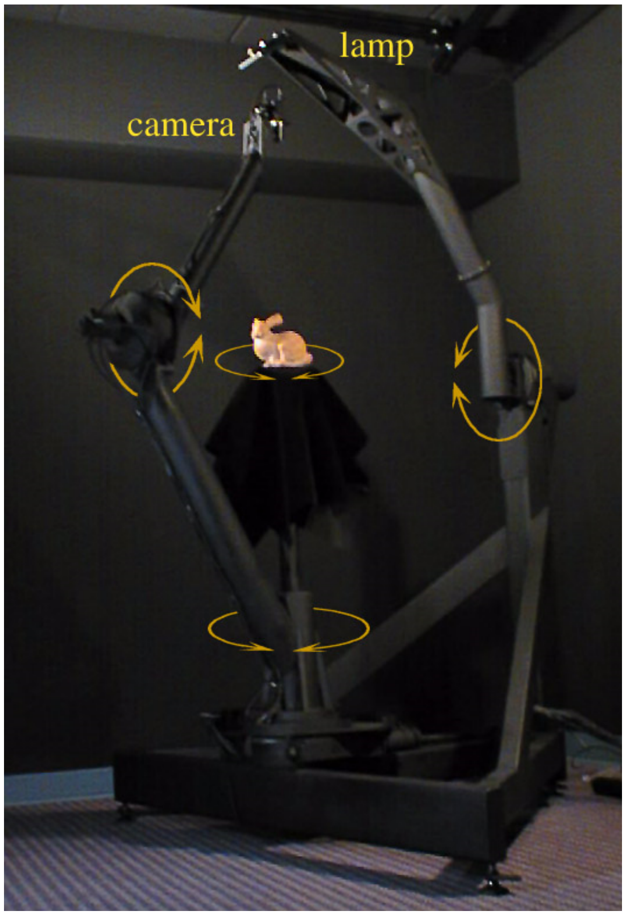
Point Grey's Bumblebee XB3

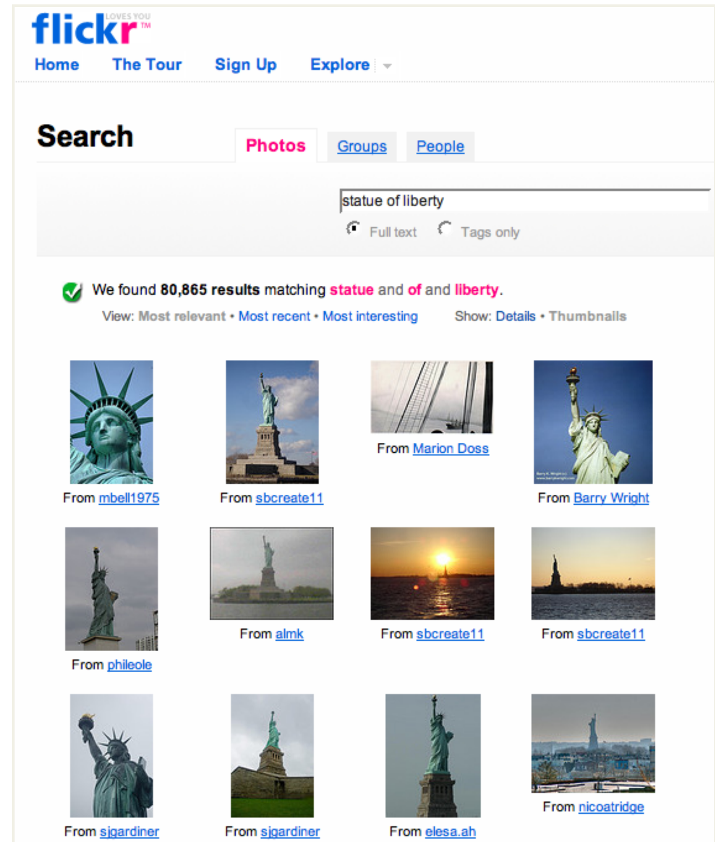
Point Grey's ProFusion 25

CMU's 3D Room

[Slide: N. Snavely]

Multi-view Stereo



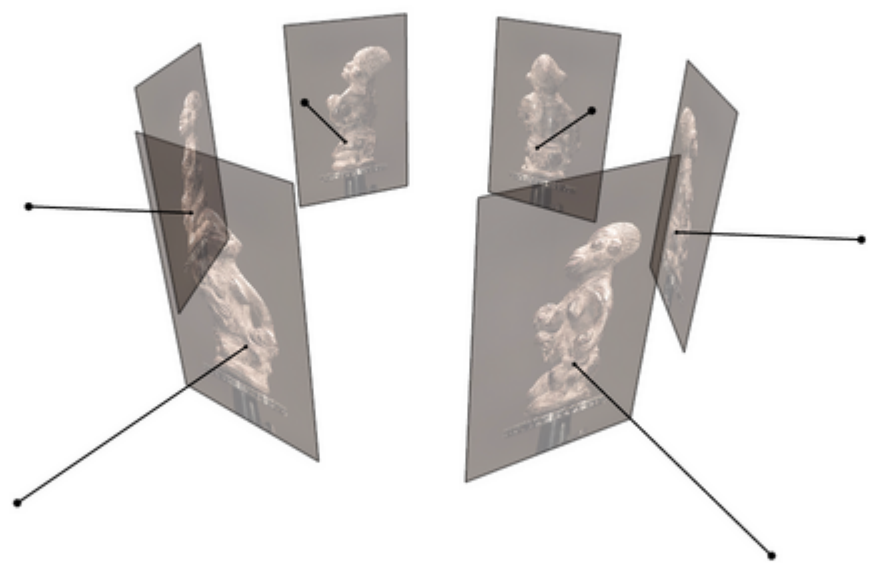


[Slide: N. Snavely]

Multi-view Stereo

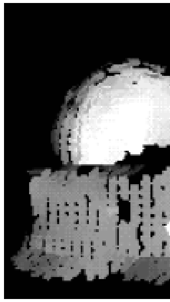
Input: calibrated images from several viewpoints

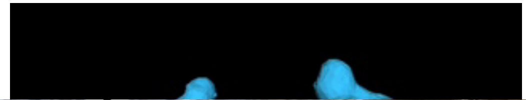
Output: 3D object model



Figures by Carlos Hernandez

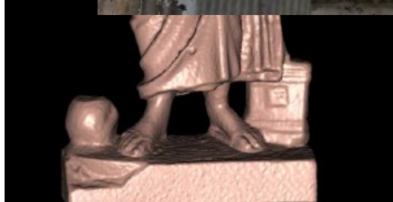
[Slide: N. Snavely]





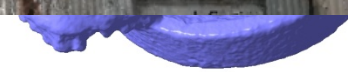
Faugeras, Keriven
1998

Furukawa et al., 2010



Hernandez, Schmitt
2004

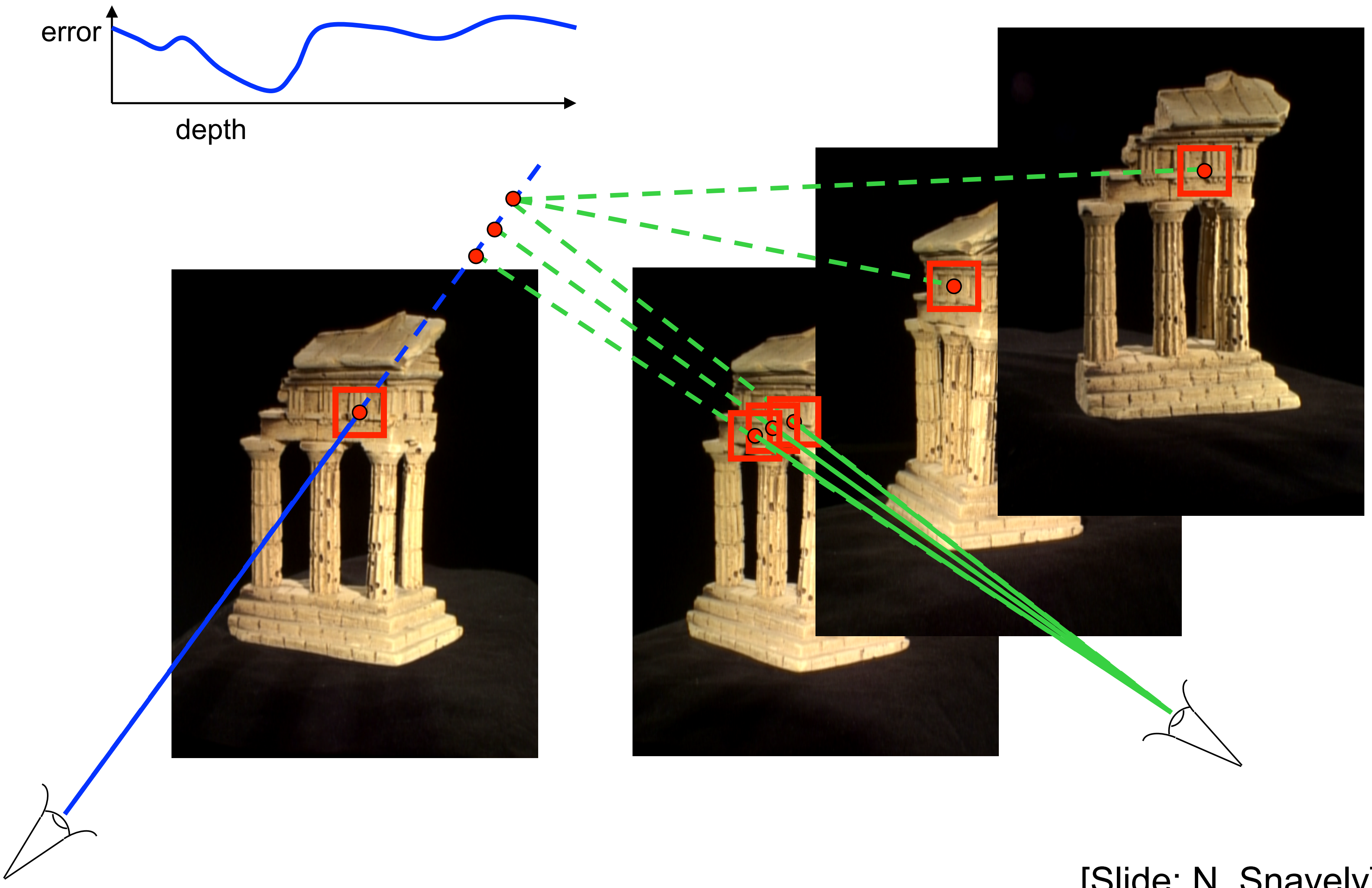
Pons, Keriven, Faugeras
2005



Furukawa, Ponce
2006

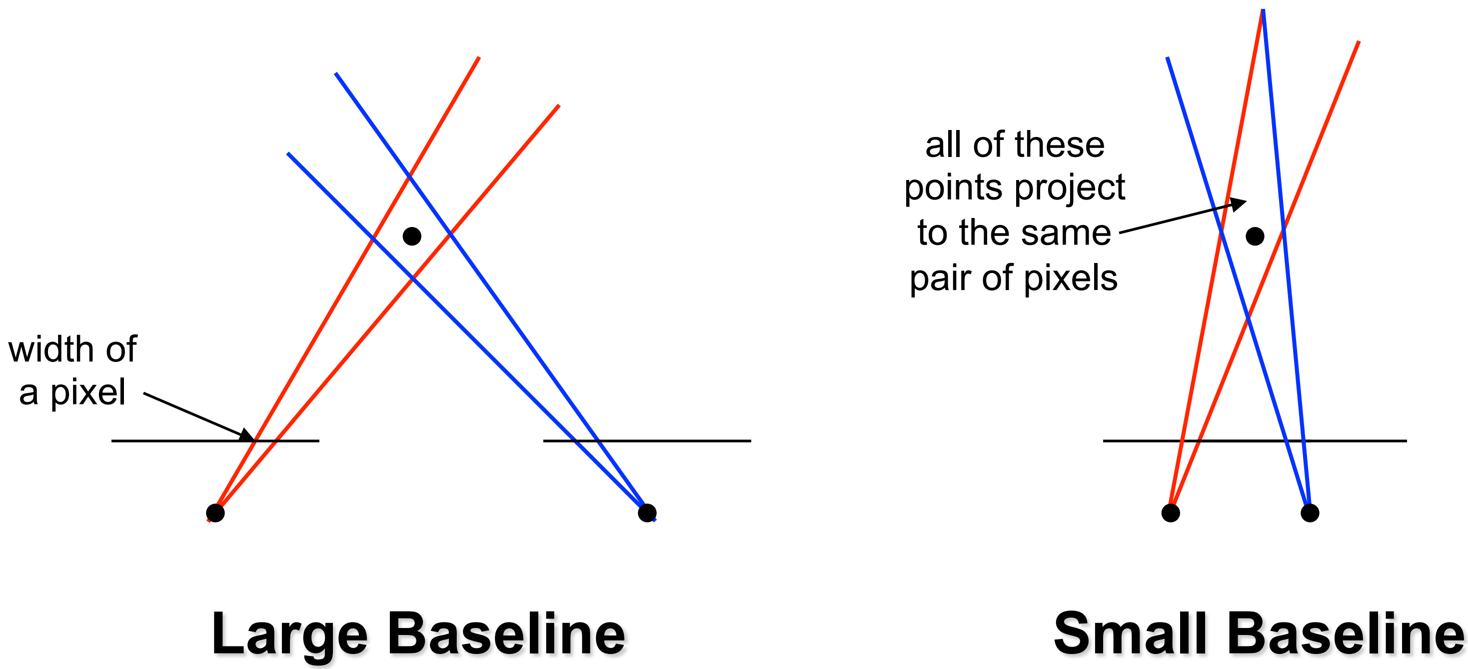
Goesele et al.
2007

Stereo: another view



[Slide: N. Snavely]

Choosing the stereo baseline



What's the optimal baseline?

- Too small: large depth error
- Too large: difficult search problem

[Slide: N. Snavely]

The Effect of Baseline on Depth Estimation

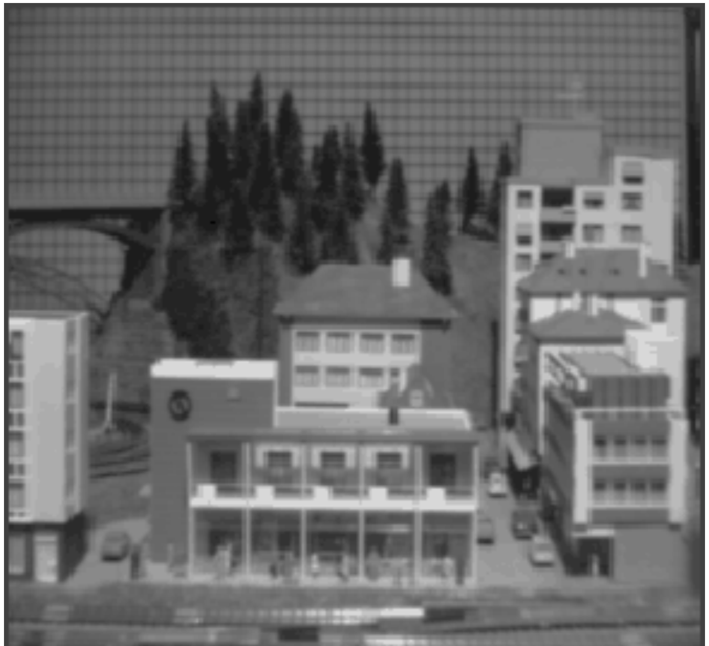
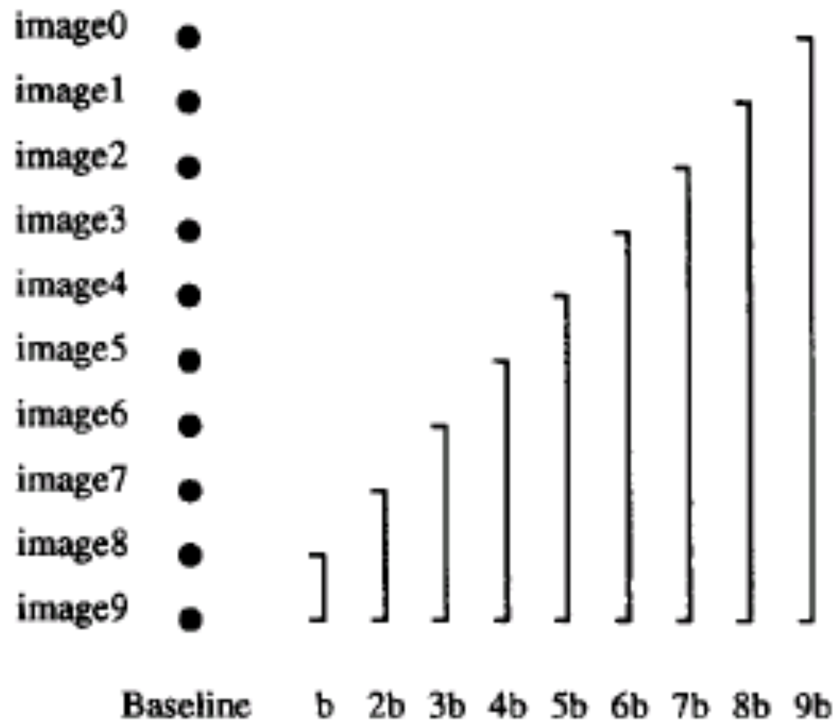
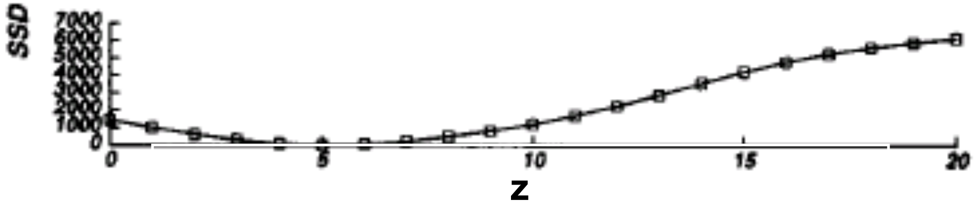
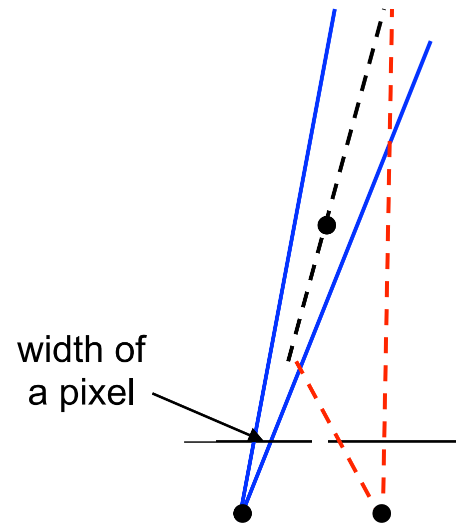


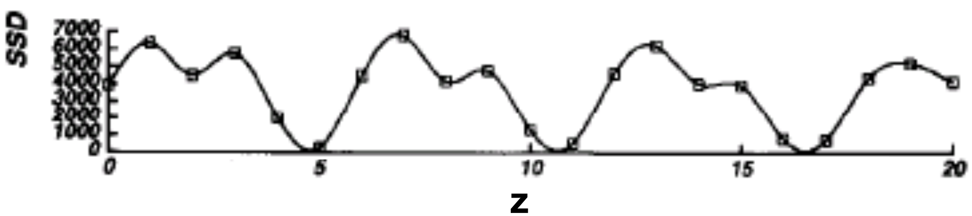
Figure 2: An example scene. The grid pattern in the background has ambiguity of matching.

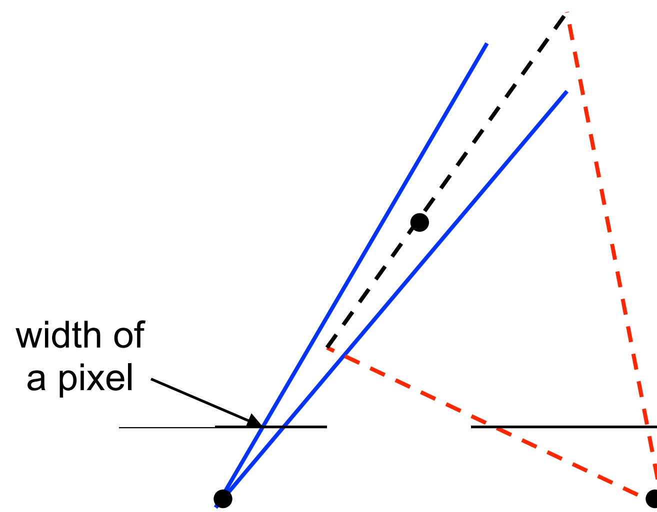






pixel matching score





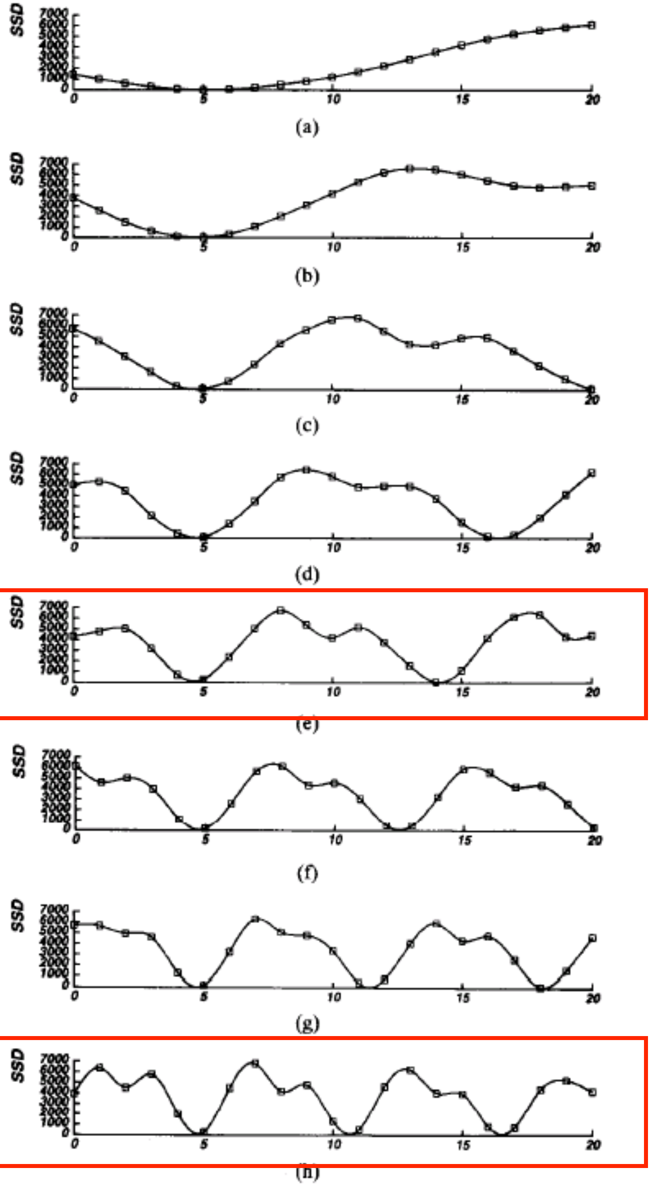


Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$.

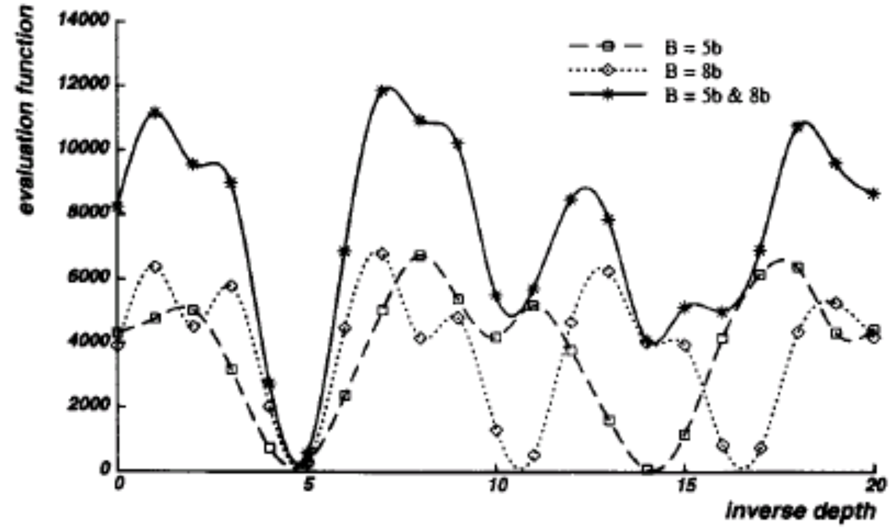


Fig. 6. Combining two stereo pairs with different baselines.

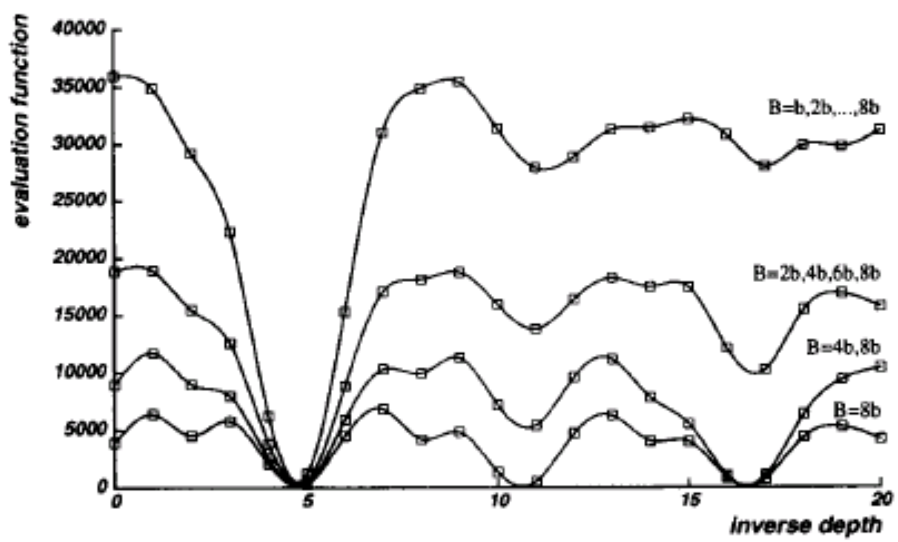


Fig. 7. Combining multiple baseline stereo pairs.

Multibaseline Stereo

Basic Approach

- Choose a reference view
- Use your favorite stereo algorithm BUT
  - > replace two-view SSD with SSSD over all baselines

Limitations

- Only gives a depth map (not an "object model")
- Won't work for widely distributed views:

Problem: *visibility*

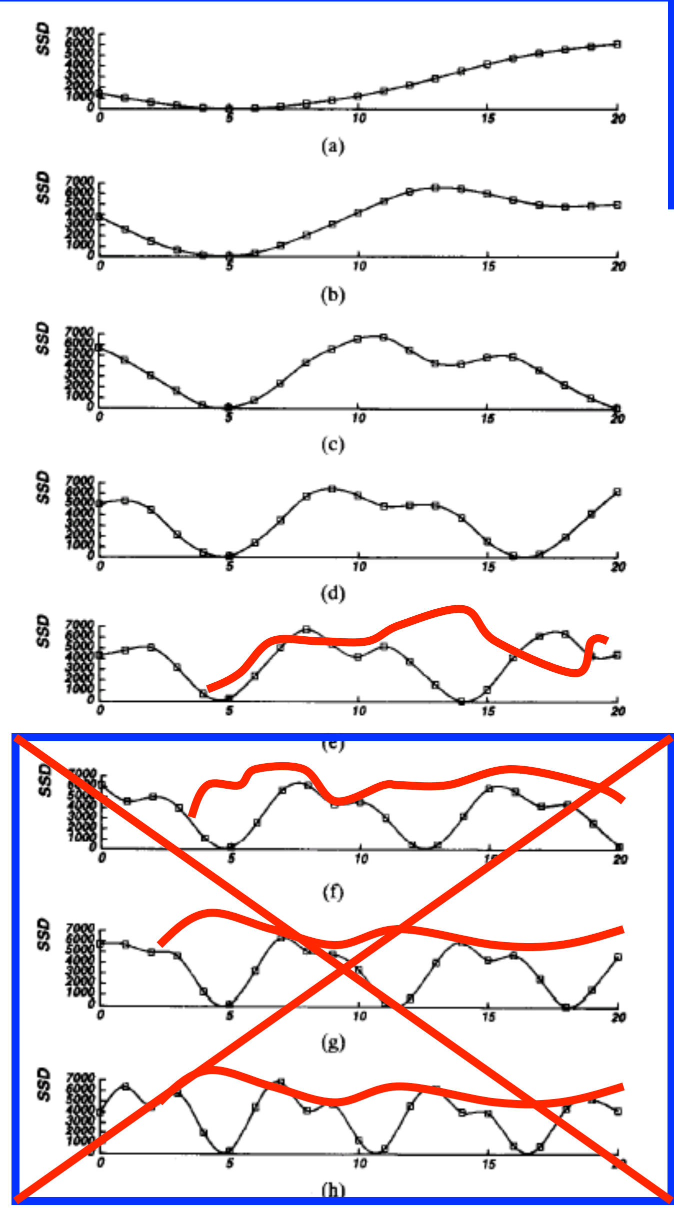


Fig. 5. SSD values versus inverse distance: (a) $B = b$; (b) $B = 2b$; (c) $B = 3b$; (d) $B = 4b$; (e) $B = 5b$; (f) $B = 6b$; (g) $B = 7b$; (h) $B = 8b$. The horizontal axis is normalized such that $8bF = 1$.

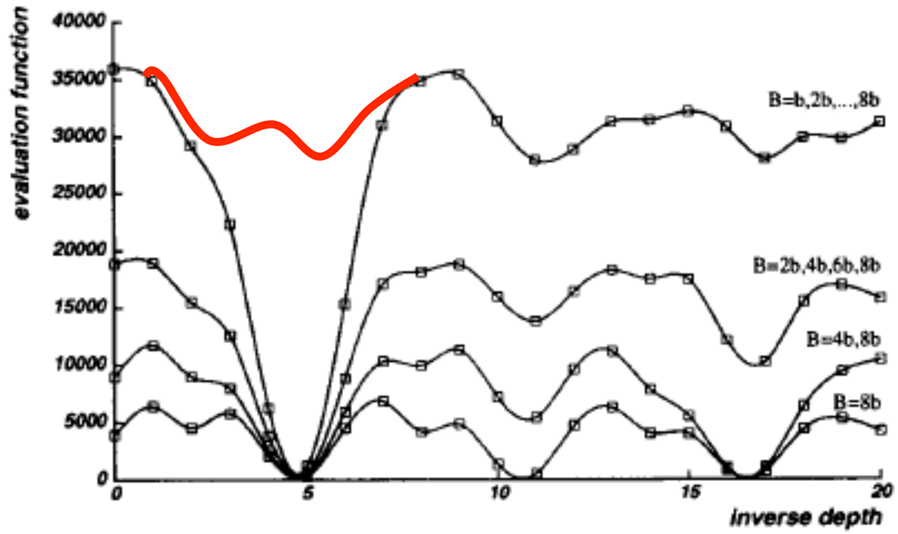


Fig. 7. Combining multiple baseline stereo pairs.

Some Solutions

- Match only nearby photos [Narayanan 98]
- Use NCC instead of SSD,
  Ignore NCC values > threshold
  [Hernandez & Schmitt 03]

Popular matching scores

- SSD (Sum Squared Distance)

$$\sum_{x,y} |W_1(x,y) - W_2(x,y)|^2$$

- NCC (Normalized Cross Correlation)

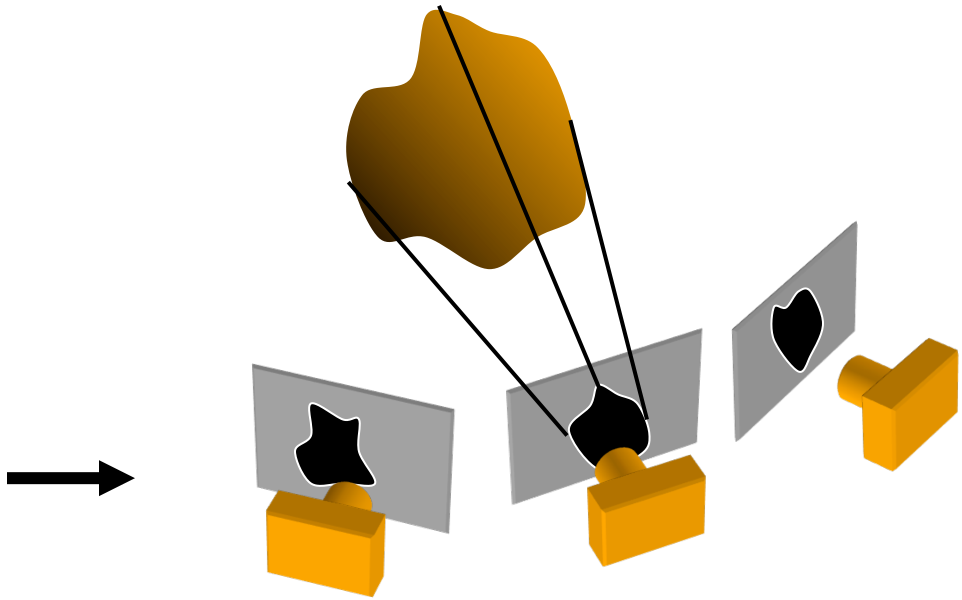
$$\frac{\sum_{x,y}(W_1(x,y) - \overline{W_1})(W_2(x,y) - \overline{W_2})}{\sigma_{W_1}\sigma_{W_2}}$$

  - where $\overline{W_i} = \frac{1}{n}\sum_{x,y} W_i \qquad \sigma_{W_i} = \sqrt{\frac{1}{n}\sum_{x,y}(W_i - \overline{W_i})^2}$
  - what advantages might NCC have?

[Slide: N. Snavely]

Reconstruction from Silhouettes

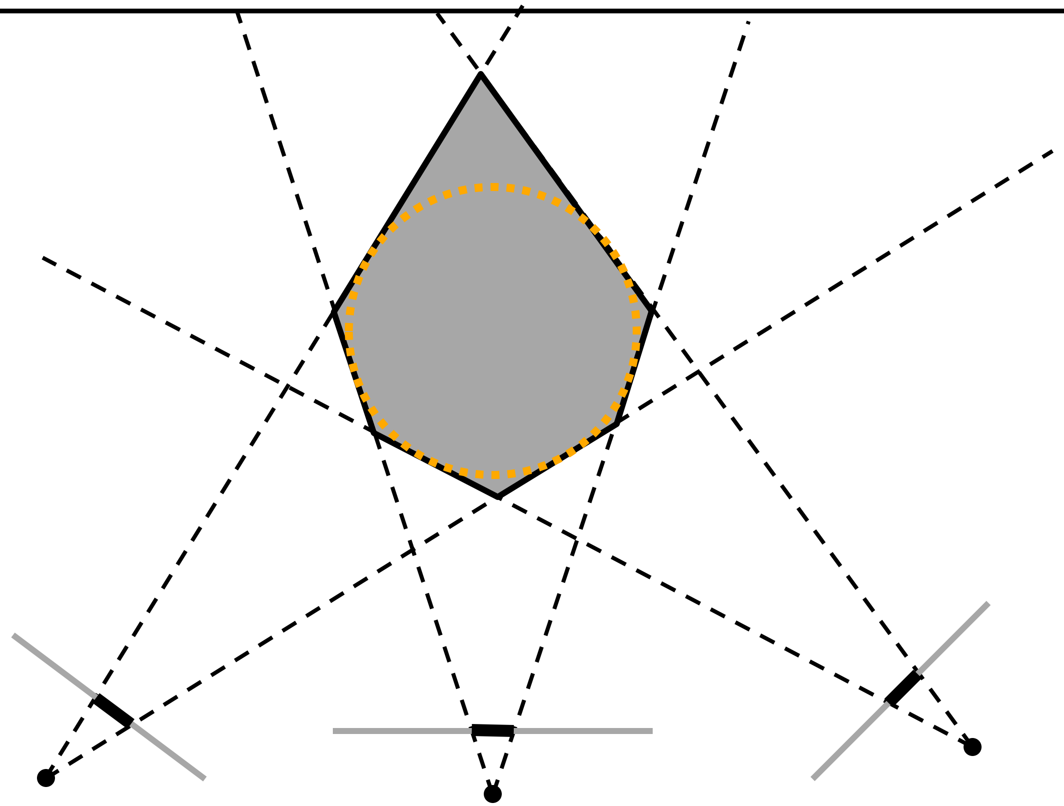
Binary Images



Approach:

- *Backproject* each silhouette
- Intersect backprojected volumes

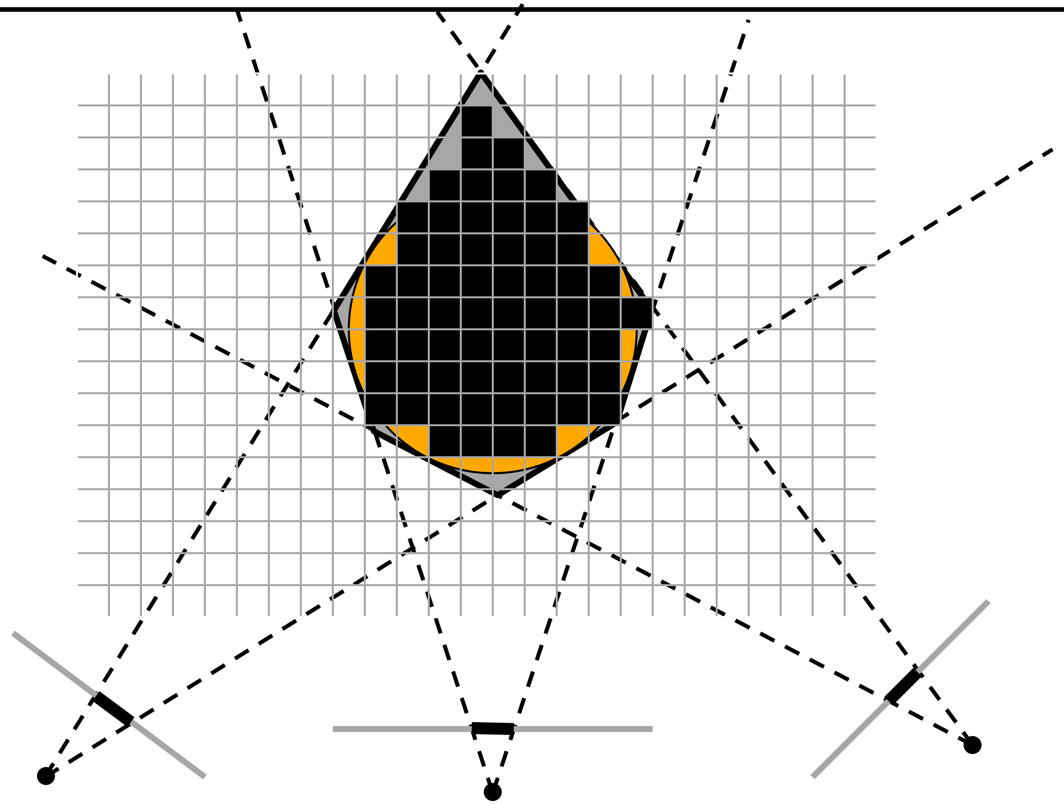
Volume intersection



Reconstruction Contains the True Scene

- But is generally not the same
- In the limit (all views) get *visual hull*
  - > Complement of all lines that don’t intersect S

Voxel algorithm for volume intersection



Color voxel black if on silhouette in every image

- $O(MN^3)$, for M images, N^3 voxels
- Don't have to search 2^{N^3} possible scenes!

Properties of Volume Intersection

Pros

- Easy to implement, fast
- Accelerated via octrees [Szeliski 1993] or interval techniques [Matusik 2000]

Cons

- No concavities
- Reconstruction is not photo-consistent
- Requires identification of silhouettes

Multi-view stereo: Summary

- Multiple-baseline stereo
  - Pick one input view as reference
  - Inverse depth instead of disparity
- Volumetric stereo
  - Photo-consistency
  - Space carving
- Shape from silhouettes
  - Visual hull: intersection of visual cones
- Carved visual hulls
- Feature-based stereo
  - From sparse to dense correspondences

All assume calibrated cameras!

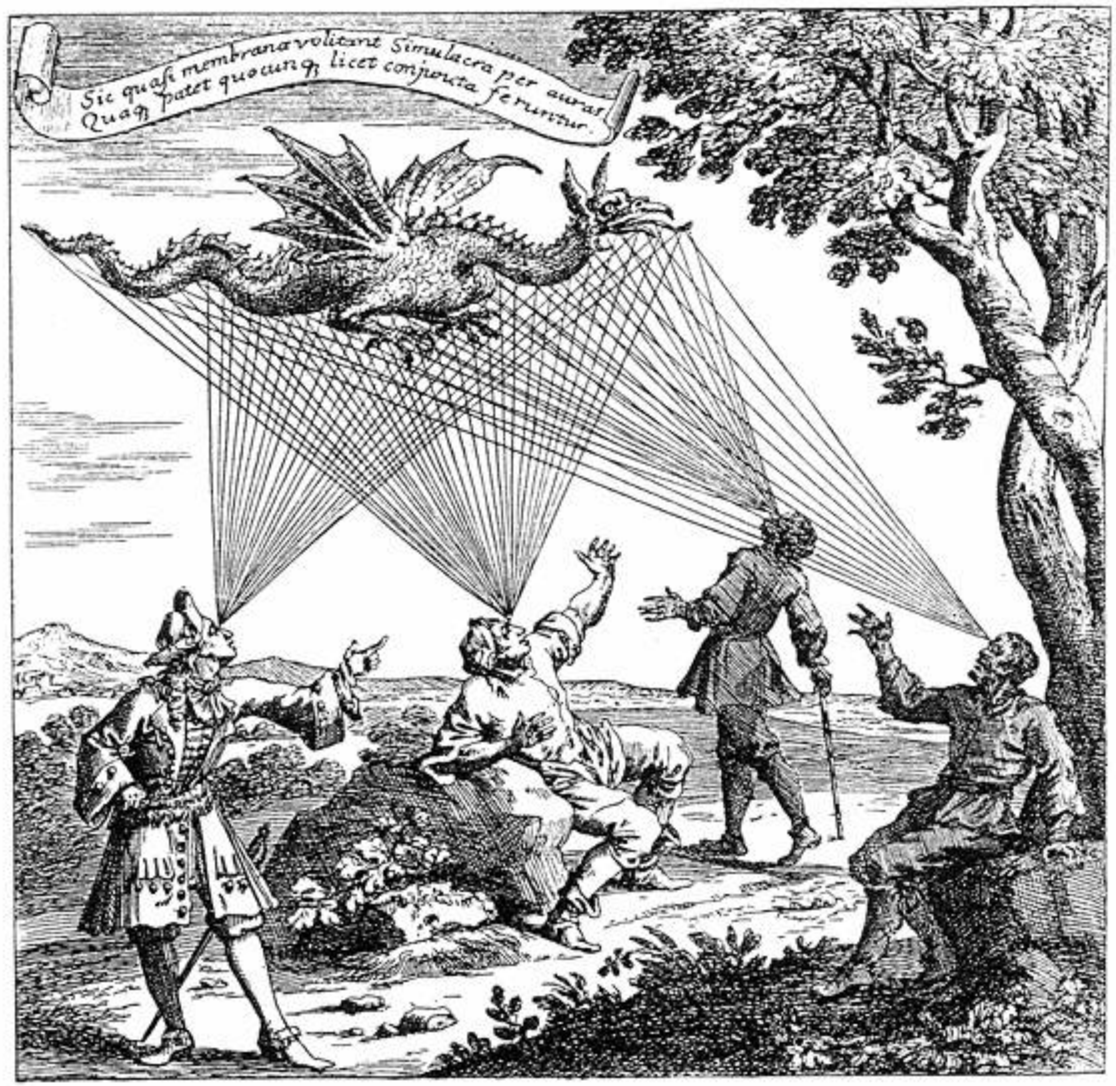
Overview

Multi-view stereo

Structure from Motion (SfM)

Large scale Structure from Motion

Structure from motion



Драконъ, видимый подъ различными углами зрѣнія

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

Multiple-view geometry questions

- **Scene geometry (structure):** Given 2D point matches in two or more images, where are the corresponding points in 3D?

- **Correspondence (stereo matching):** Given a point in just one image, how does it constrain the position of the corresponding point in another image?

- **Camera geometry (motion):** Given a set of corresponding points in two or more images, what are the camera matrices for these views?

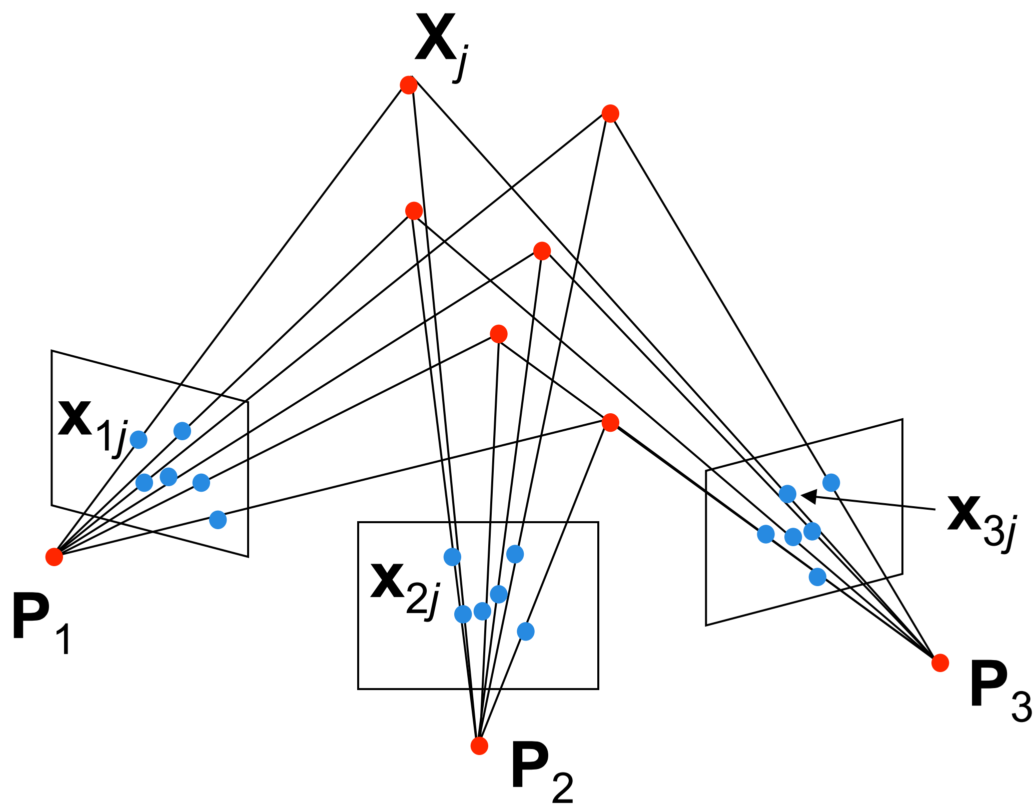
Slide: S. Lazebnik

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 $\mathbf{x}_{ij}$



Slide: S. Lazebnik

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!

Slide: S. Lazebnik

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

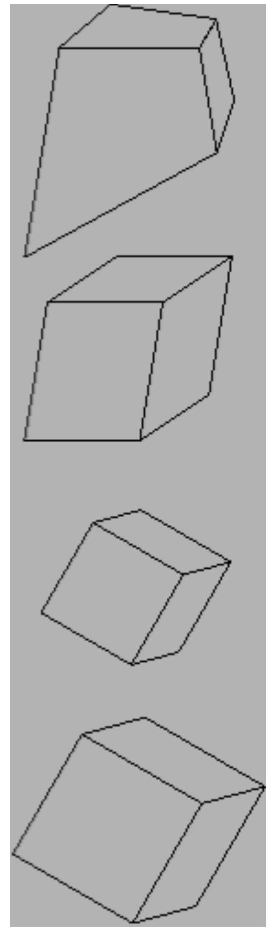
- More generally: if we transform the scene using a transformation **Q** and apply the inverse transformation to the camera matrices, then the images do not change

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

Slide: S. Lazebnik

Types of ambiguity

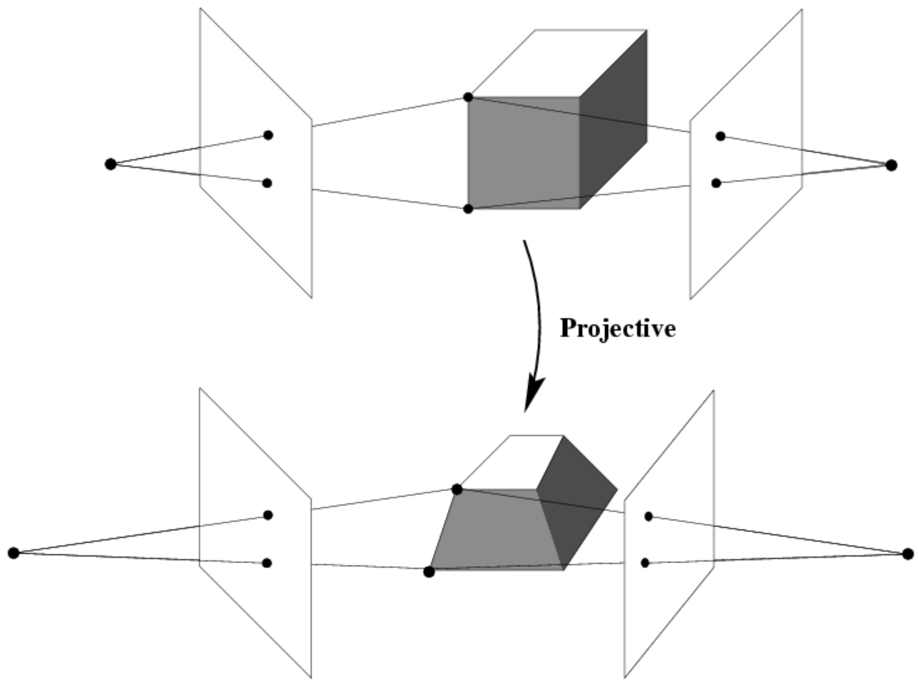
Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$	Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$	Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$	Preserves angles, ratios of length
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

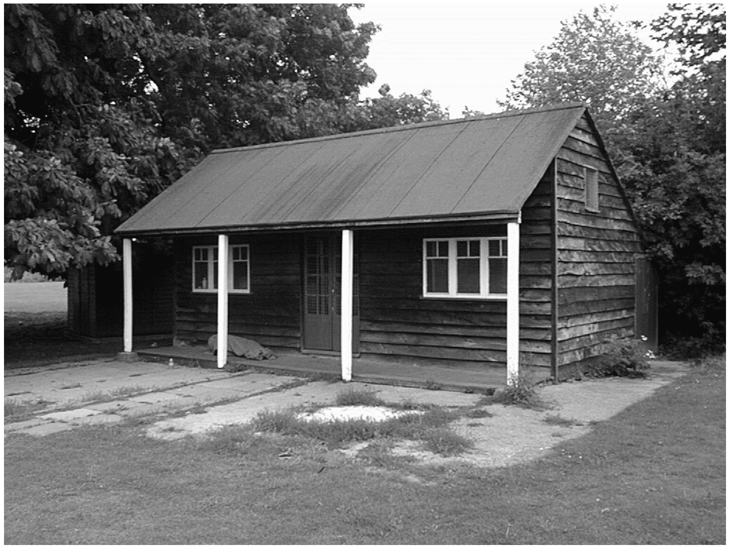
Slide: S. Lazebnik

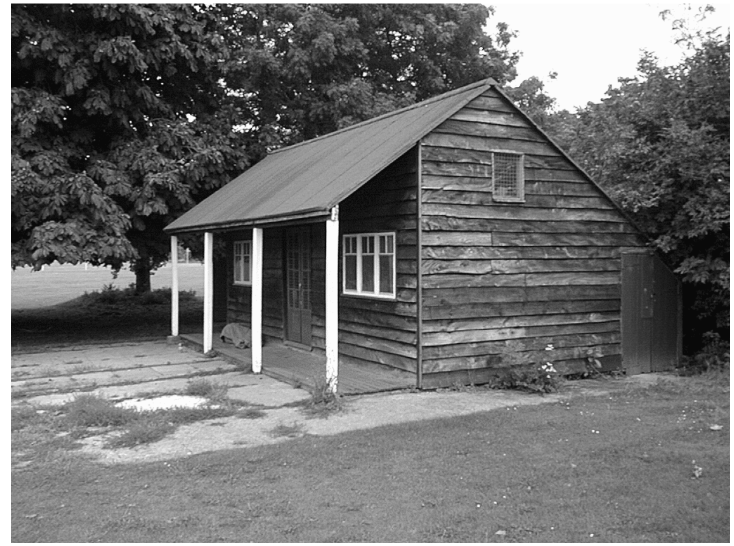
Projective ambiguity

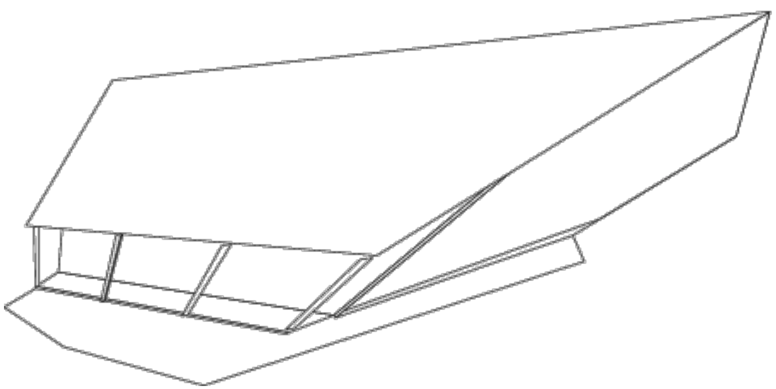


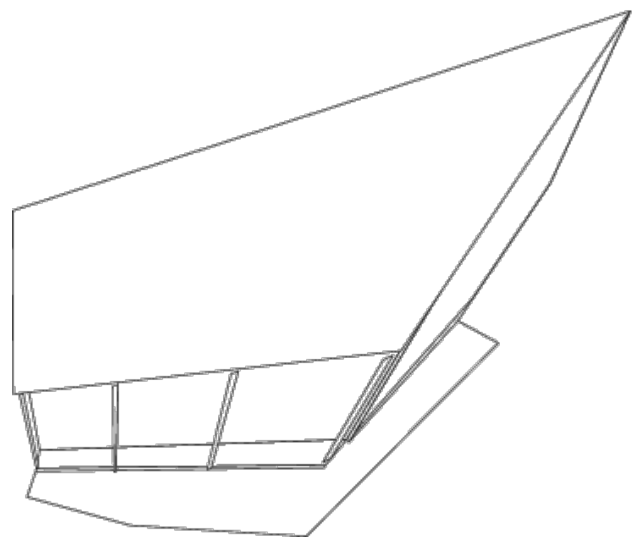
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_P^{-1}\right)\left(\mathbf{Q}_P\mathbf{X}\right)$$

Projective ambiguity

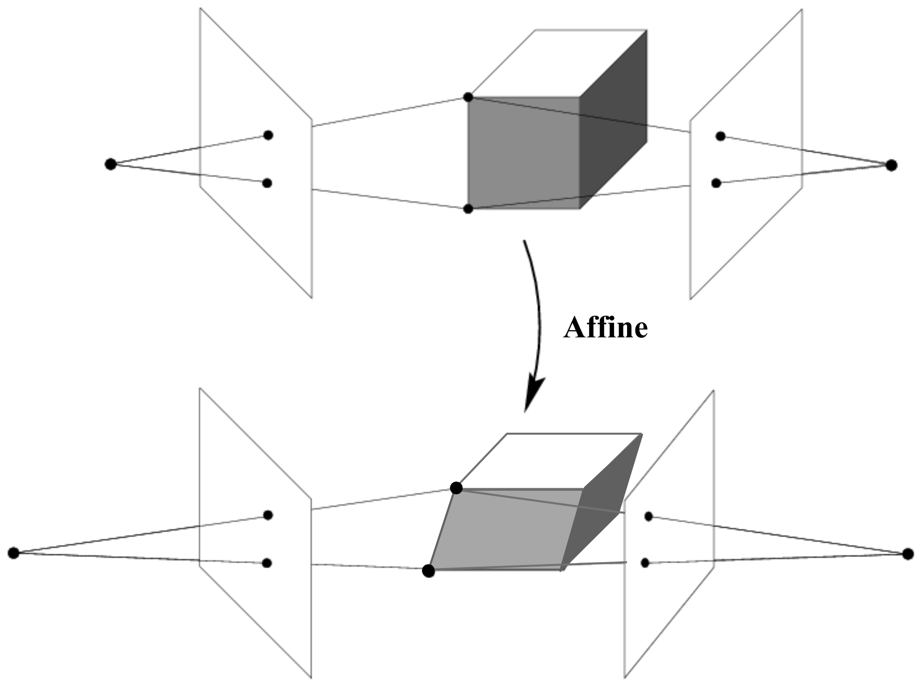






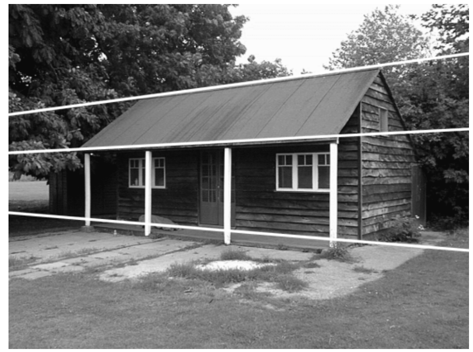


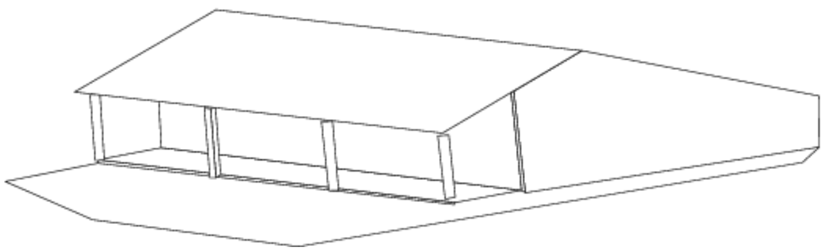
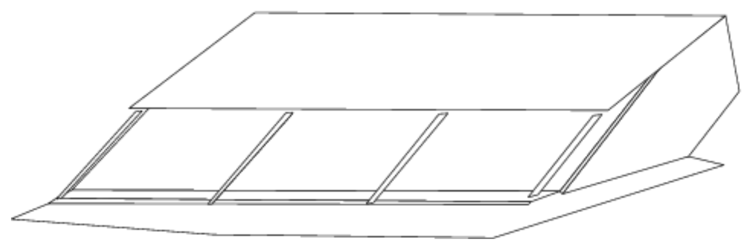
Affine ambiguity



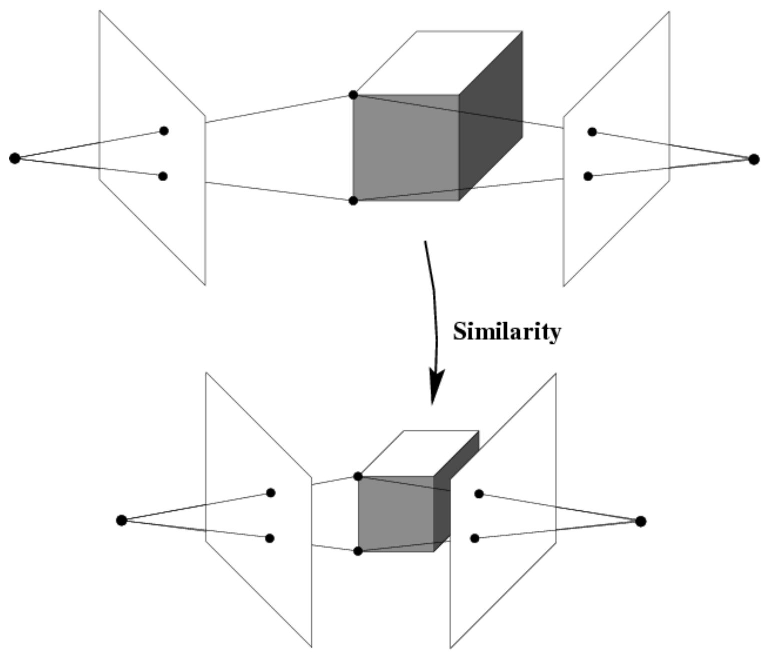
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_A^{-1}\right)\left(\mathbf{Q}_A\mathbf{X}\right)$$

Affine ambiguity

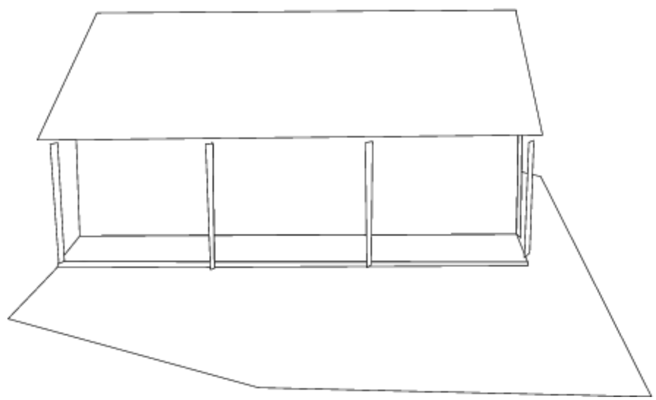
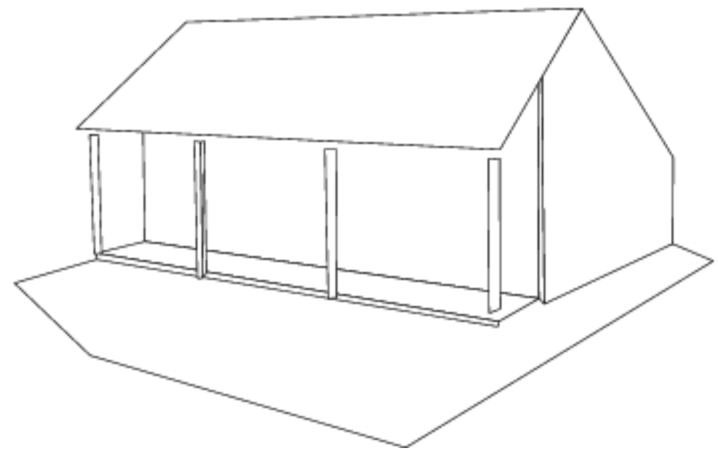
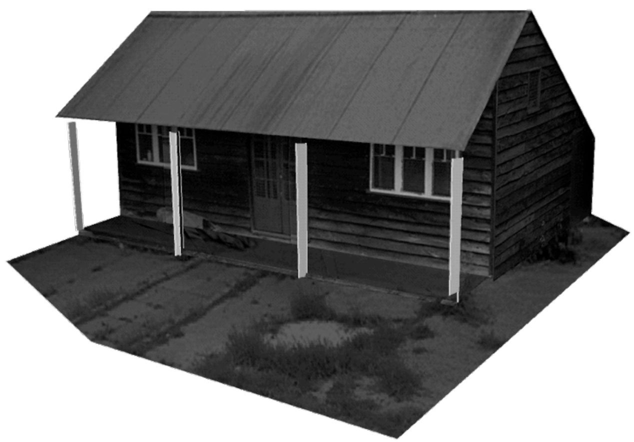



Similarity ambiguity



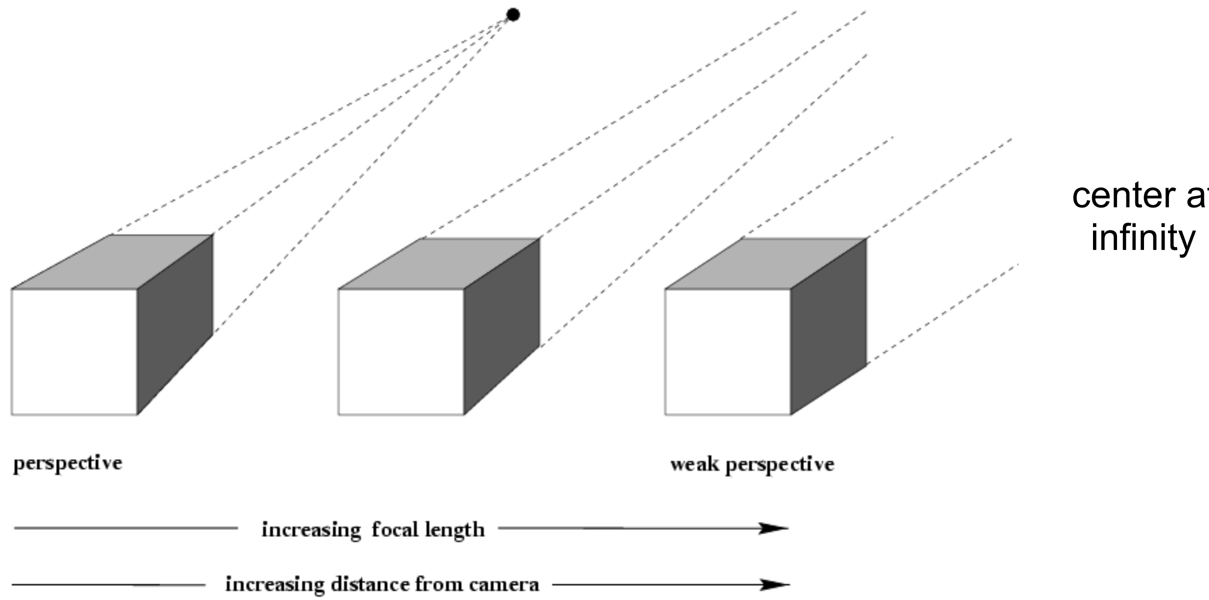
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}_S^{-1}\right)\left(\mathbf{Q}_S\mathbf{X}\right)$$

Similarity ambiguity


Structure from motion

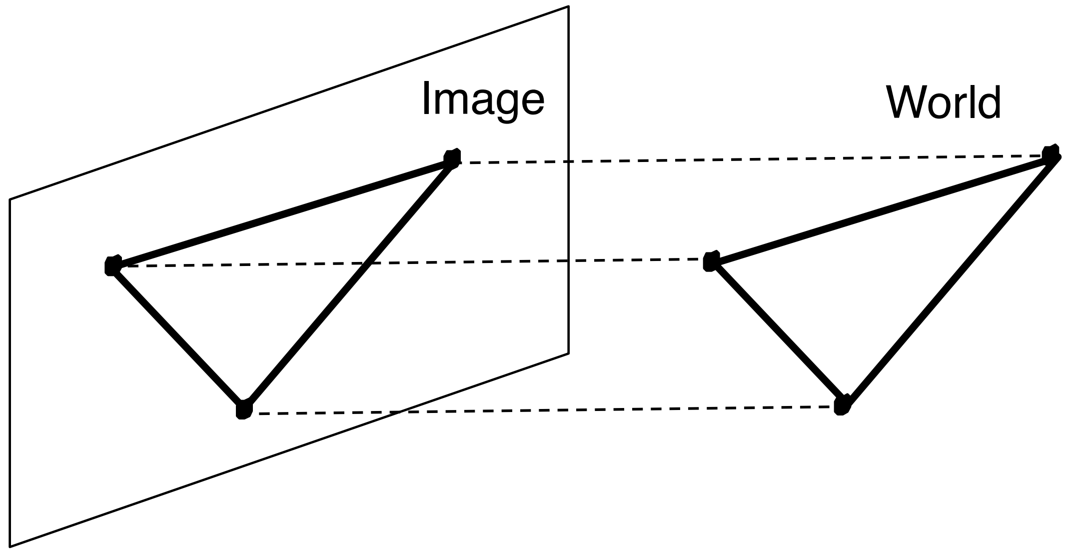
- Let's start with *affine cameras* (the math is easier)



Recall: Orthographic Projection

Special case of perspective projection

- Distance from center of projection to image plane is infinite



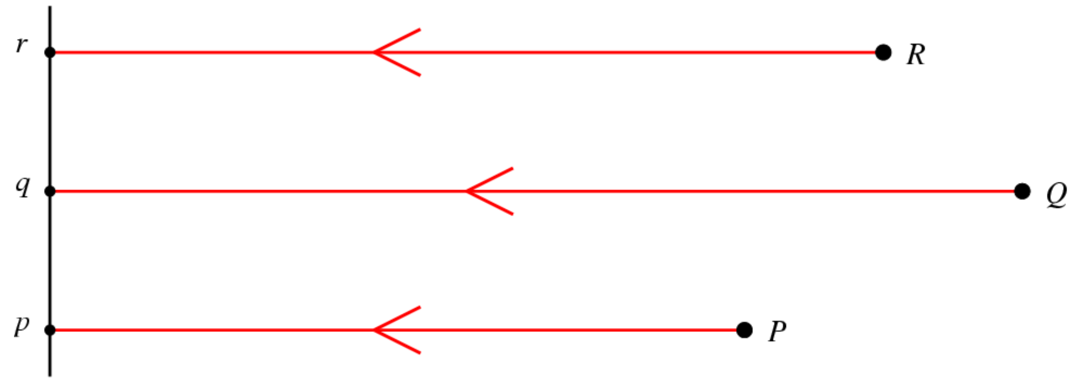
- Projection matrix:

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

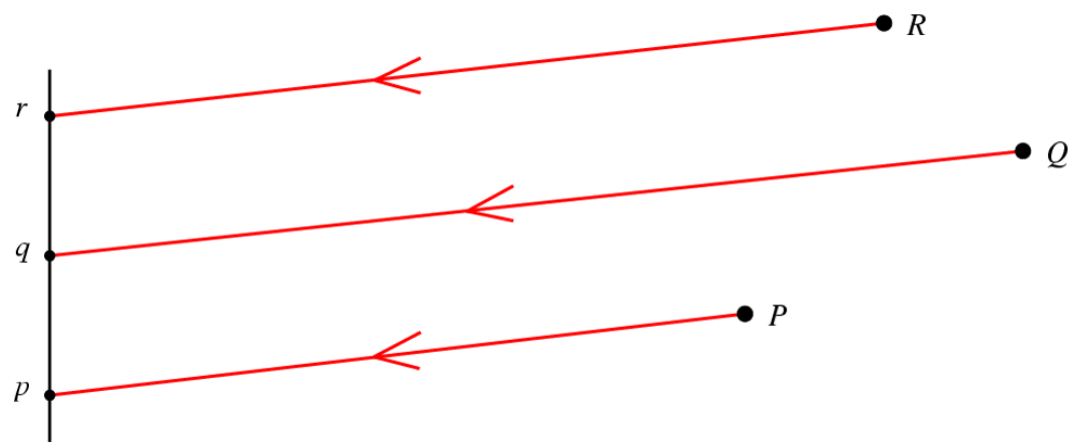
Slide by Steve Seitz

Affine cameras

Orthographic Projection



Parallel Projection

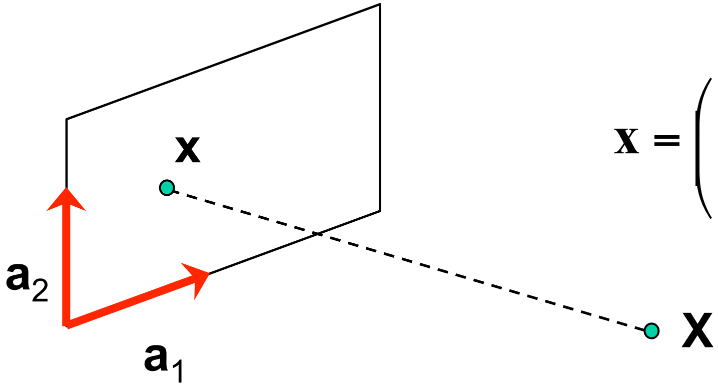


Affine cameras

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

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

- 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 $2mn$ knowns and $8m + 3n$ unknowns (minus 12 dof for affine ambiguity)
- Thus, we must have $2mn >= 8m + 3n – 12$
- 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 2*m* × *n* data (measurement) matrix:

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

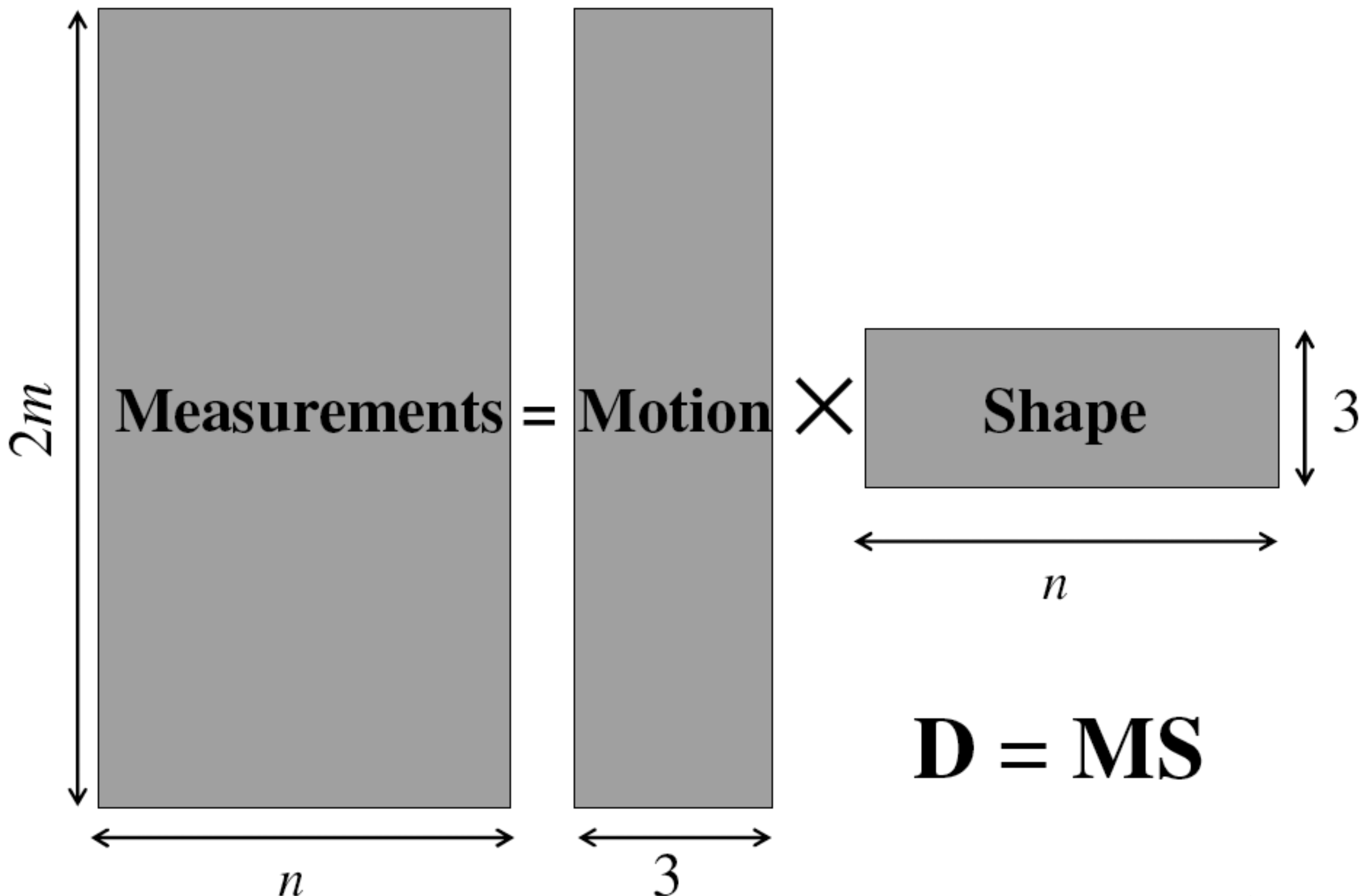
points (3 × *n*)

cameras (2*m* × 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

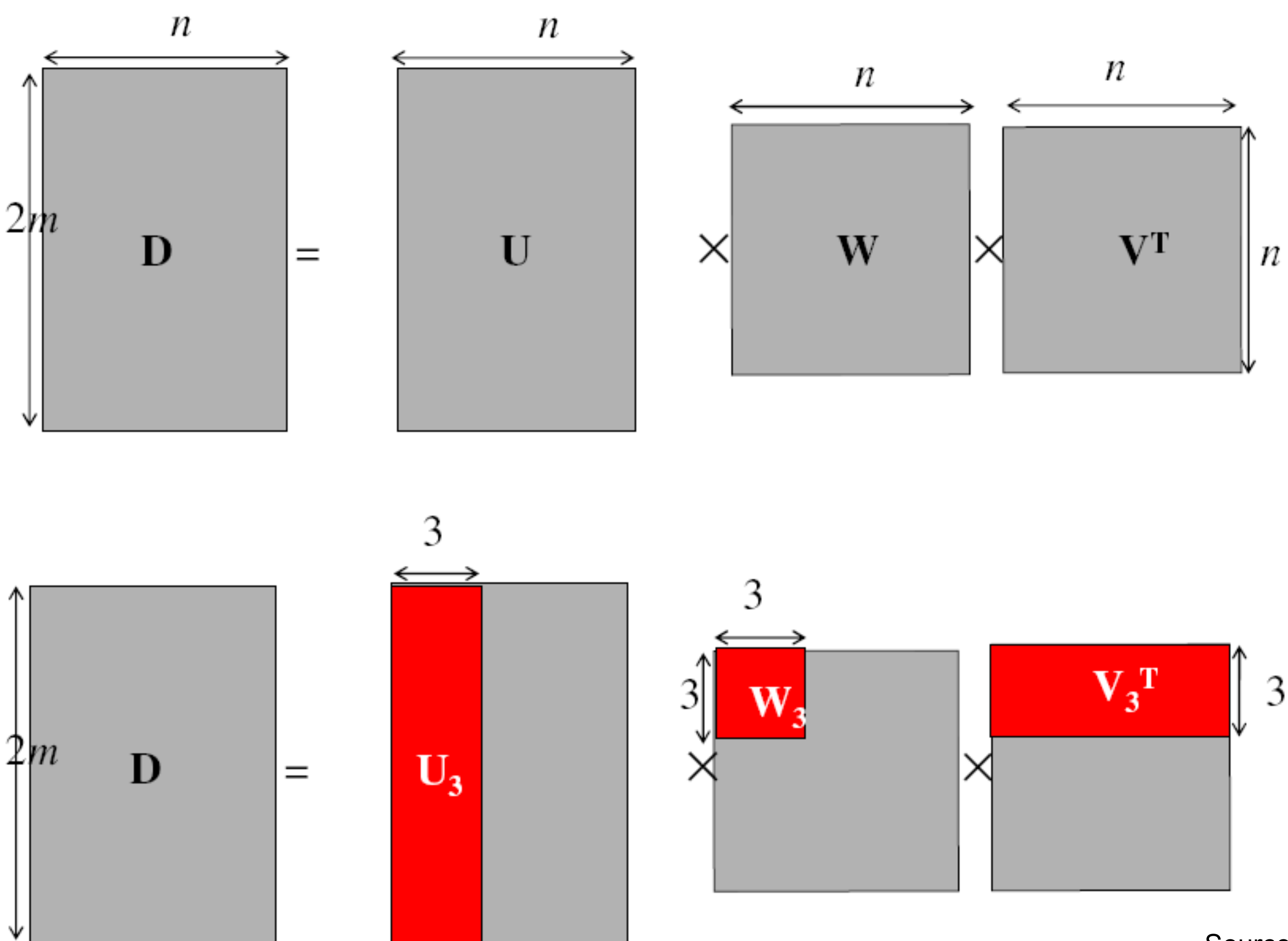


$$\mathbf{D} = \mathbf{MS}$$

Source: M. Hebert

Factorizing the measurement matrix

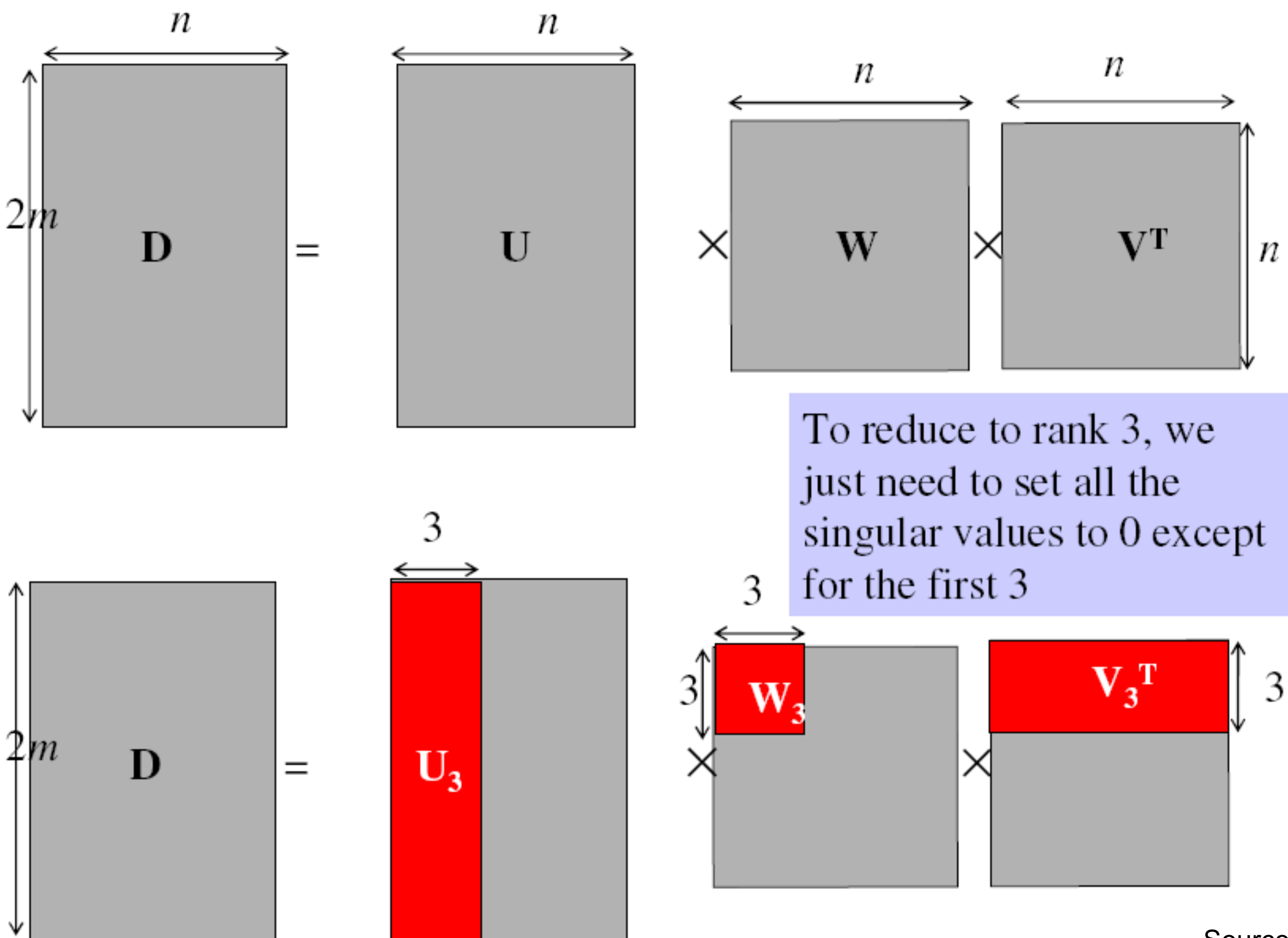
- Singular value decomposition of D:



Source: M. Hebert

Factorizing the measurement matrix

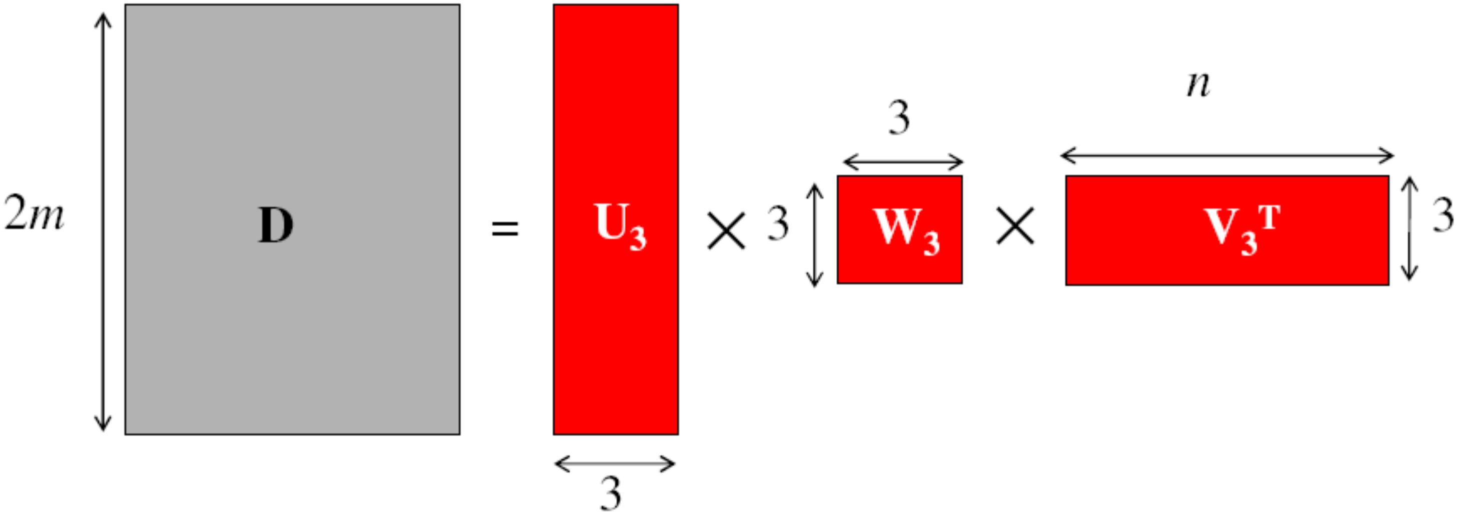
- Singular value decomposition of D:



Source: M. Hebert

Factorizing the measurement matrix

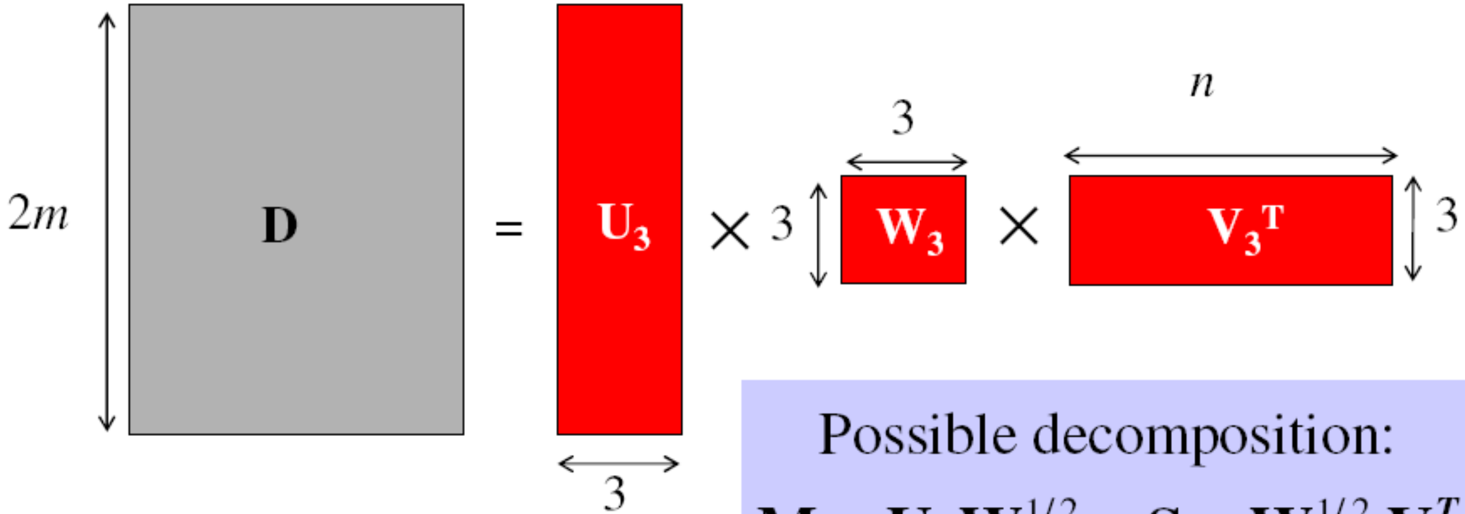
- Obtaining a factorization from SVD:



Source: M. Hebert

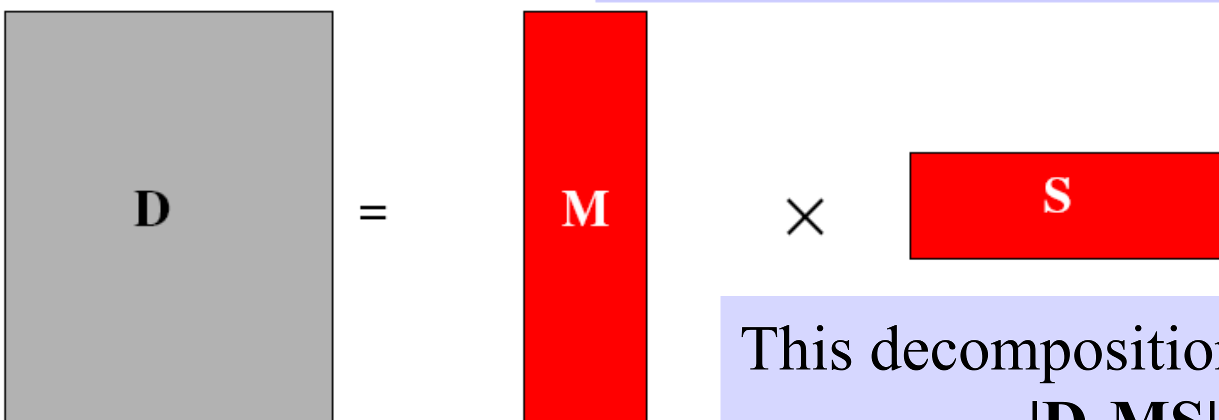
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



Possible decomposition:

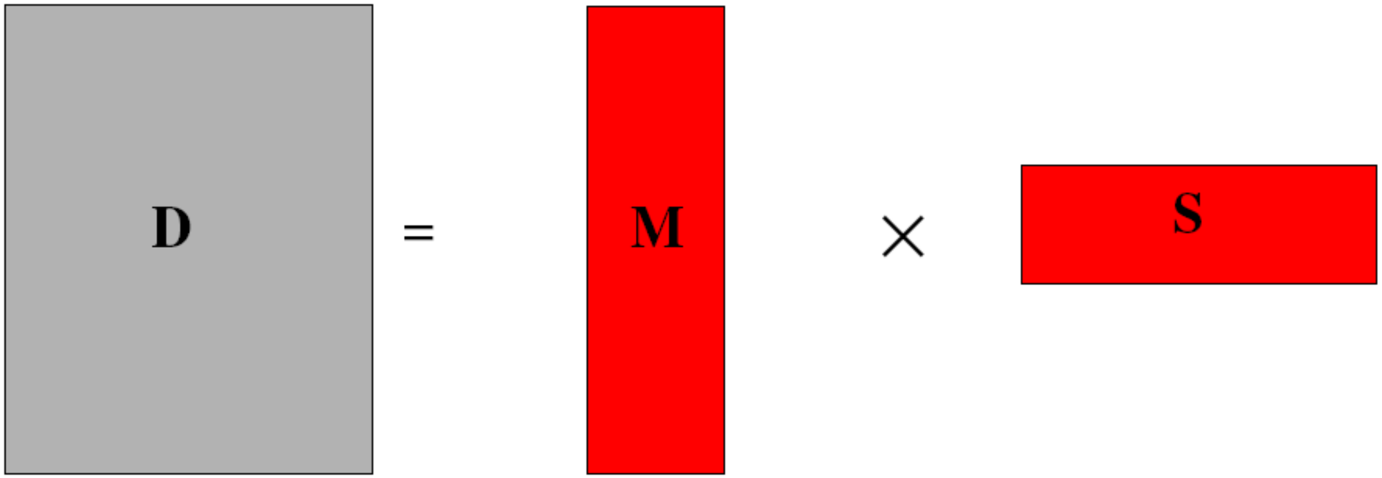
$$\mathbf{M} = \mathbf{U}_3 \mathbf{W}_3^{1/2} \qquad \mathbf{S} = \mathbf{W}_3^{1/2} \mathbf{V}_3^T$$



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

Source: M. Hebert

Affine ambiguity

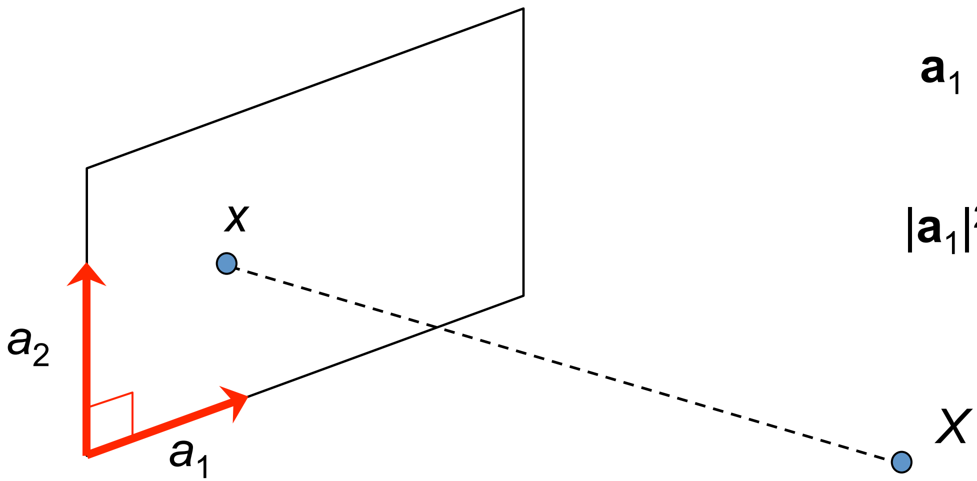


- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations **M** → **MC**, **S** → $\mathbf{C}^{-1}\mathbf{S}$
- That is because we have only an affine transformation and we have not enforced any Euclidean constraints (like forcing the image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

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

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

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

Slide: D. Hoiem

Algorithm summary

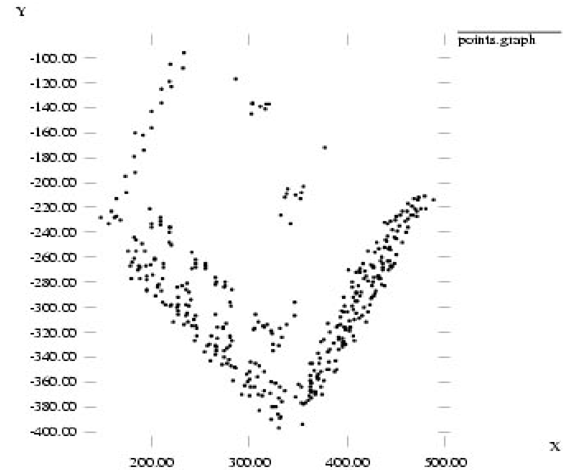
- Given: m images and n features $\mathbf{x}_{ij}$
- For each image i, center the feature coordinates
- Construct a $2m \times n$ measurement matrix $\mathbf{D}$:
  - Column j contains the projection of point j in all views
  - Row i contains one coordinate of the projections of all the n points in image 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 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

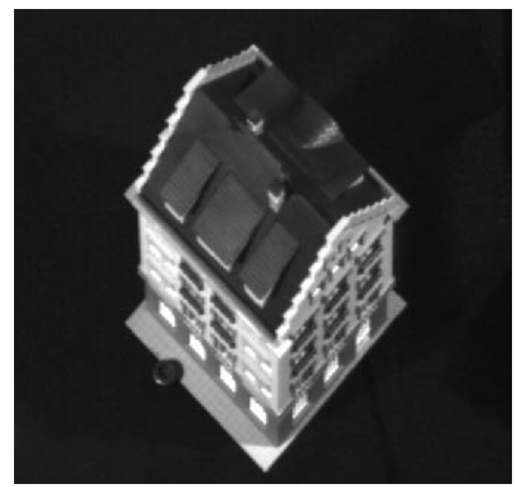
1

60



120

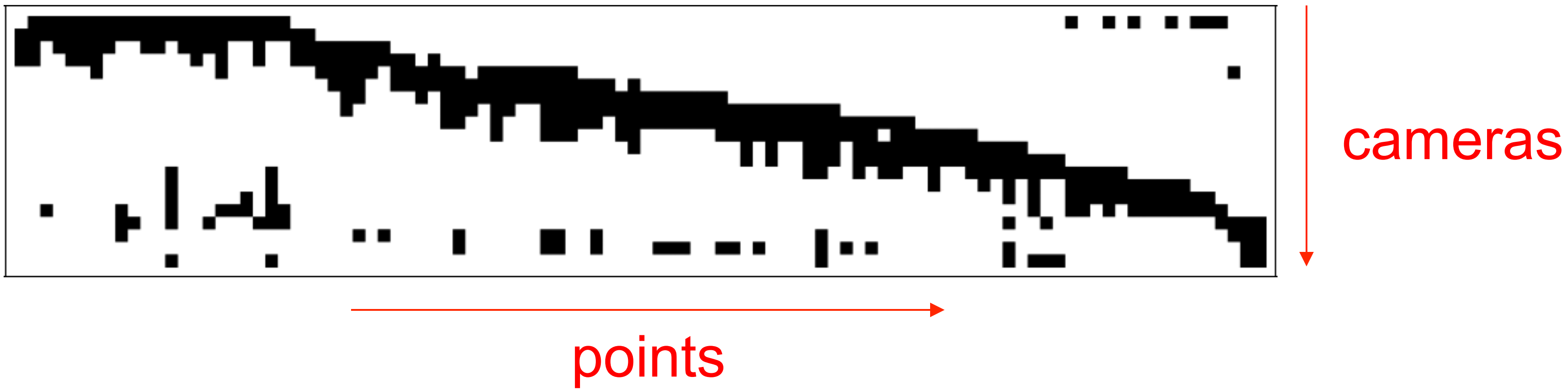
150



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

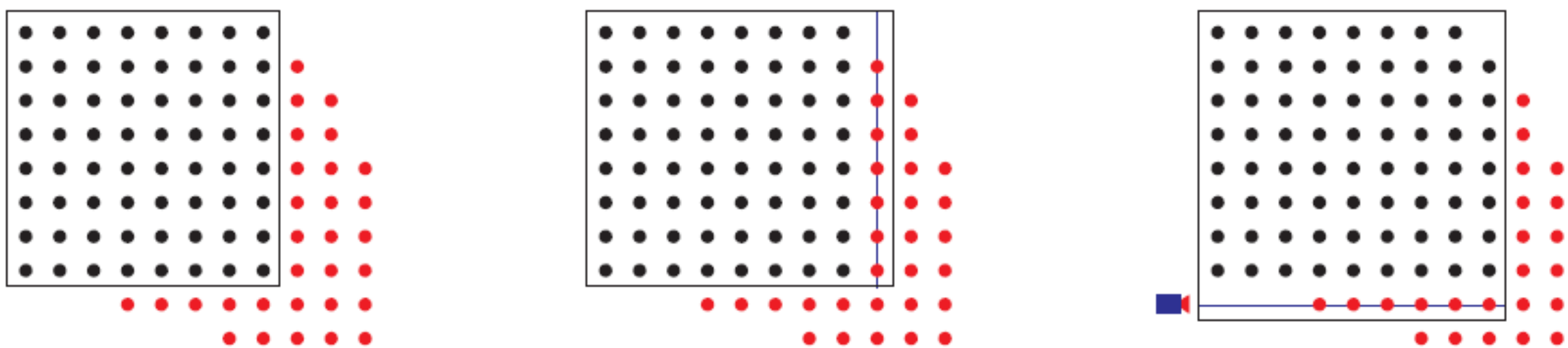
Dealing with missing data

- 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)

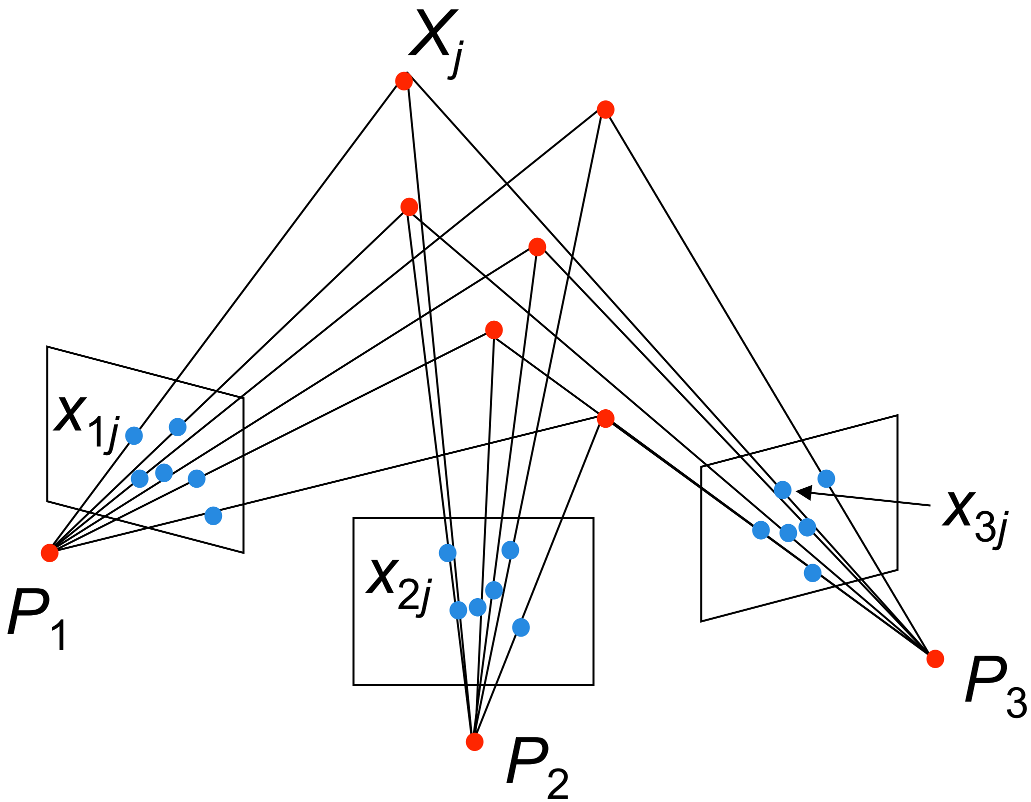
F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. [Segmenting, Modeling, and Matching Video Clips Containing Multiple Moving Objects.](#) PAMI 2007

Projective structure from motion

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

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

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



Projective structure from motion

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

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

- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* 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: $[\mathbf{I}|\mathbf{0}]$
- Second camera matrix: $[\mathbf{A}|\mathbf{b}]$
- Then $\mathbf{b}$ is the epipole ($\mathbf{F}^T\mathbf{b} = 0$), $\mathbf{A} = -[\mathbf{b}_\times]\mathbf{F}$

F&P sec. 13.3.1

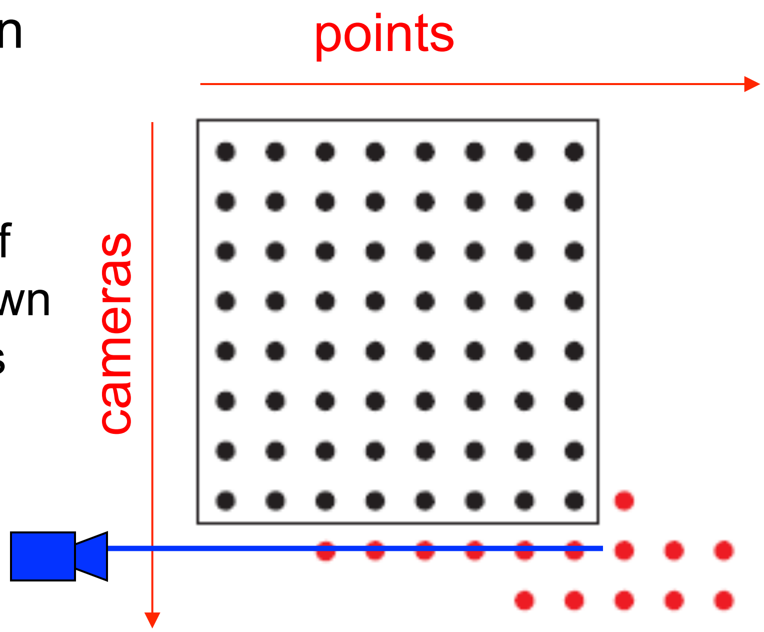
Sequential structure from motion

•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

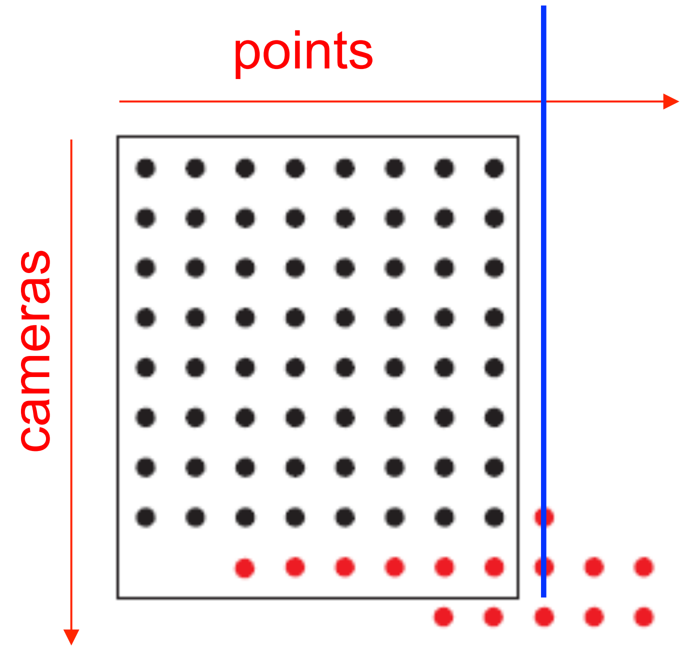
•Initialize motion from two images using fundamental matrix

•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

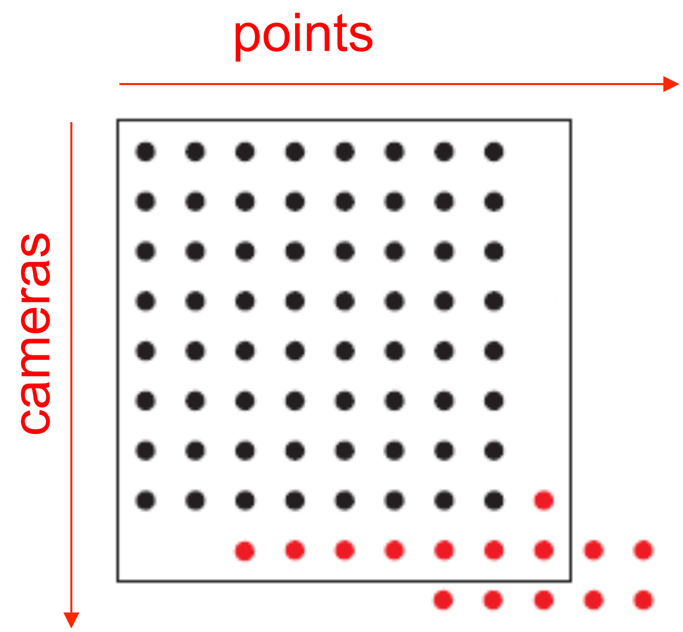
•Initialize motion from two images using fundamental matrix

•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*

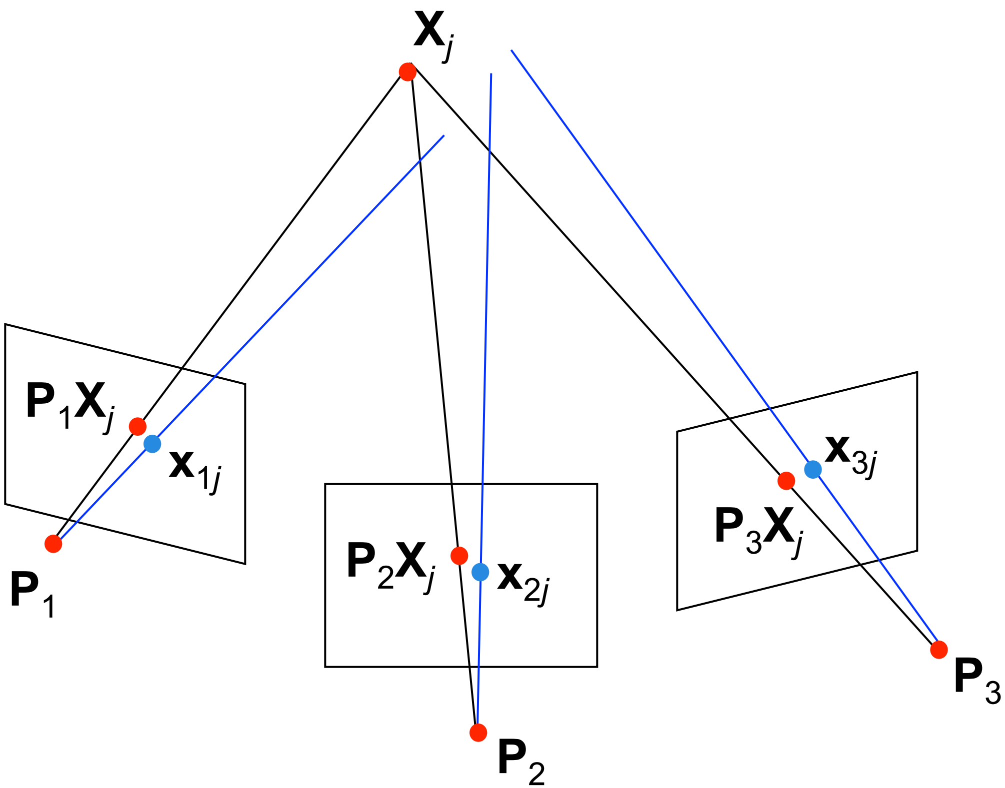


•Refine structure and motion: bundle adjustment

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

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

Summary: 3D geometric vision

- Single-view geometry
  - The pinhole camera model
    - Variation: orthographic projection
  - The perspective projection matrix
  - Intrinsic parameters
  - Extrinsic parameters
  - Calibration
- Multiple-view geometry
  - Triangulation
  - The epipolar constraint
    - Essential matrix and fundamental matrix
  - Stereo
    - Binocular, multi-view
  - Structure from motion
    - Reconstruction ambiguity
    - Affine SFM
    - Projective SFM

Overview

Multi-view stereo

Structure from Motion (SfM)

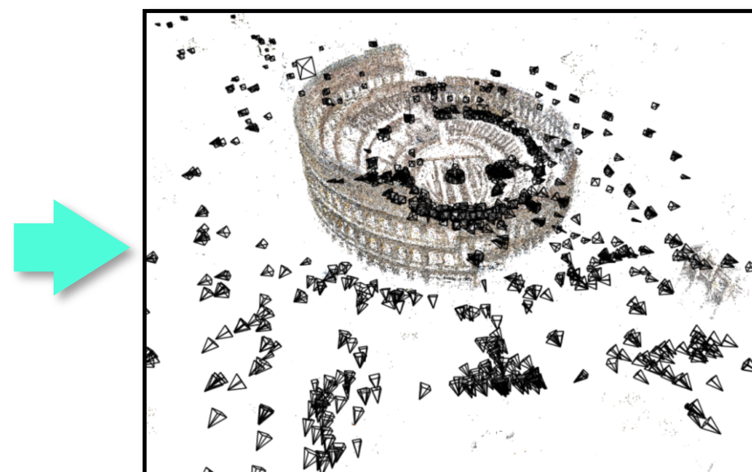
Large scale Structure from Motion

Large-scale Structure from motion

Given many images from photo collections how can we

a) figure out where they were all taken from?

b) build a 3D model of the scene?



This is (roughly) the **structure from motion** problem

Slides from N. Snavely

Challenges

appearance variation

resolution

massive collections

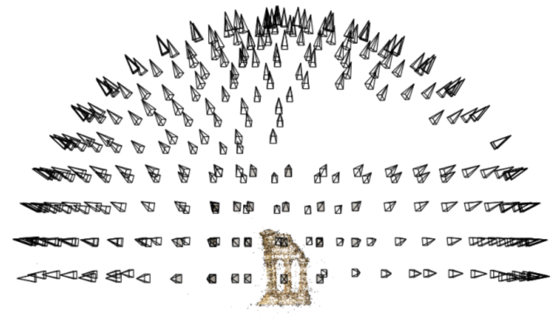
82,754 results for photos matching **notre** and **dame** and **paris**.

Large-scale structure from motion

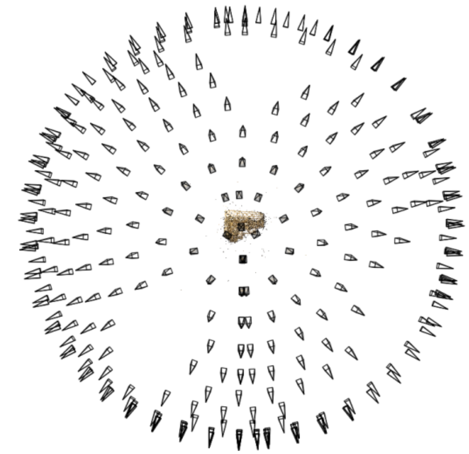
Dubrovnik, Croatia. 4,619 images (out of an initial 57,845).
Total reconstruction time: 23 hours
Number of cores: 352

Slide: N. Snavely

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*

Photo Tourism

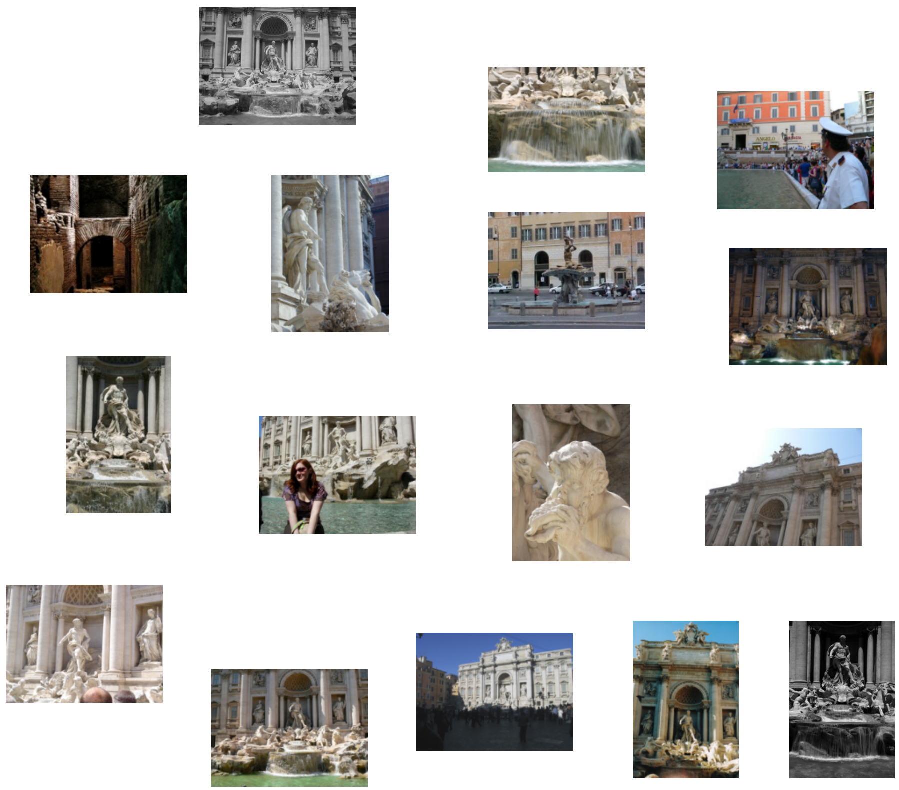
Slide: N. Snavely

First step: how to get correspondence?

Feature detection and matching

Feature detection

Detect features using SIFT [Lowe, IJCV 2004]

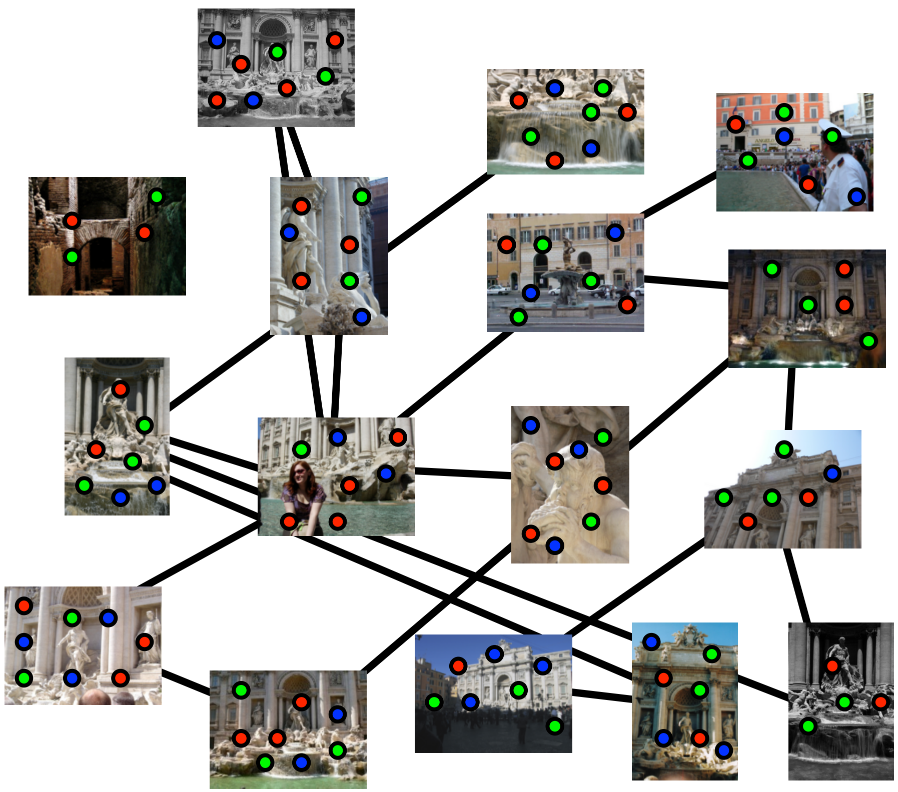


Feature detection

Detect features using SIFT [Lowe, IJCV 2004]

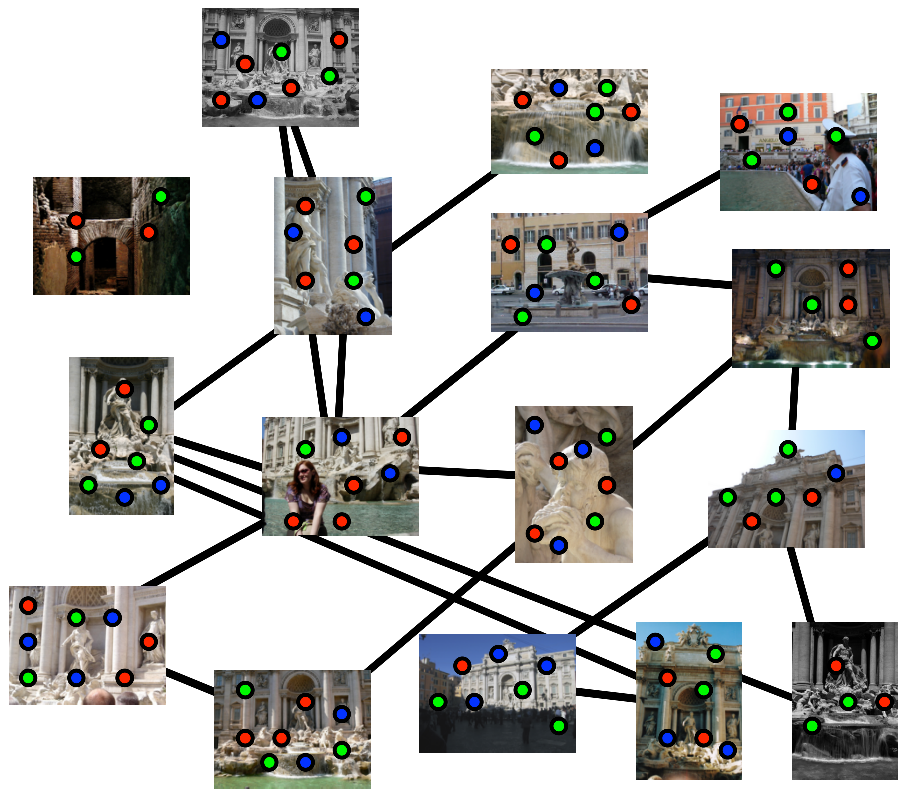
Feature matching

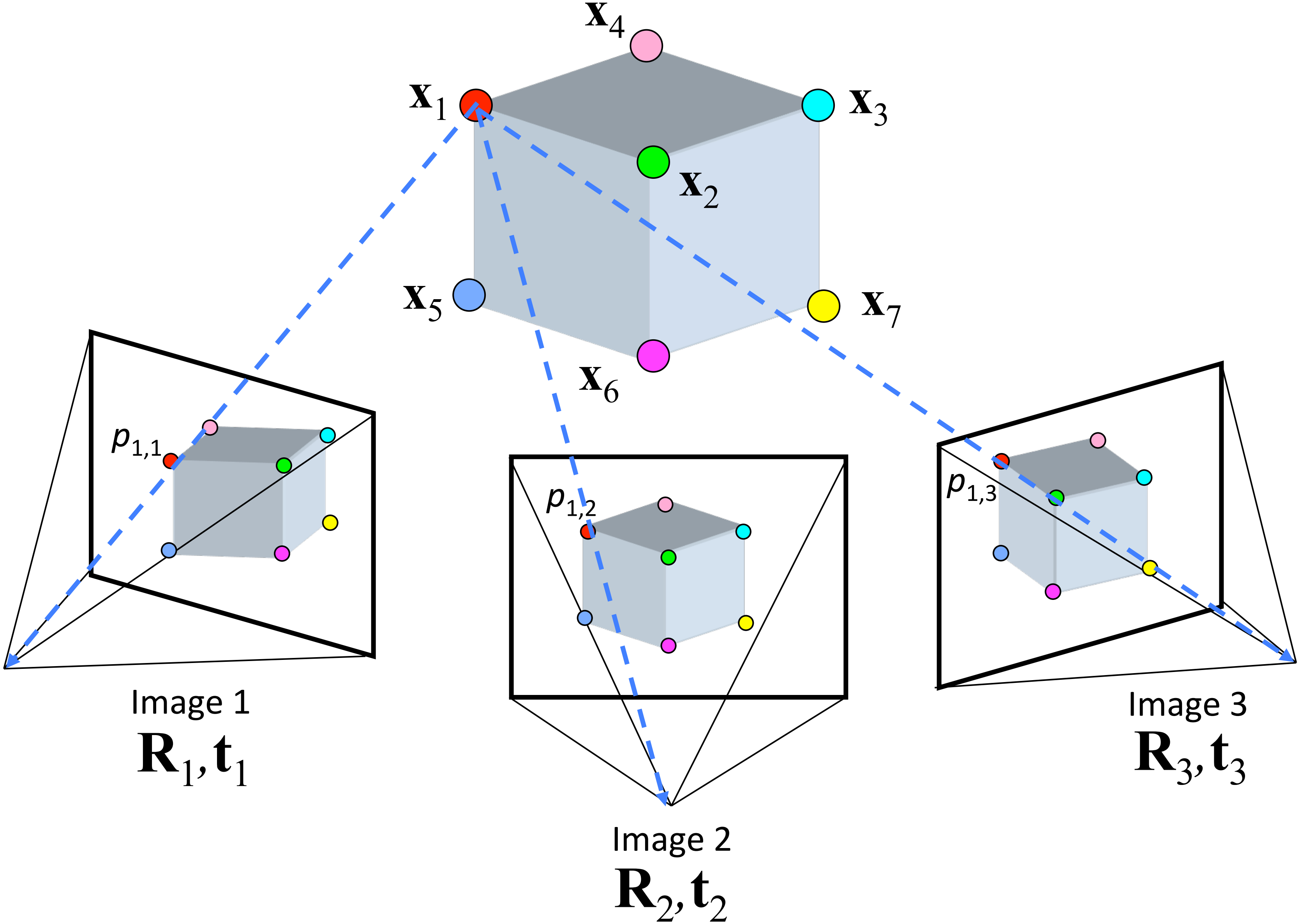
Match features between each pair of images



Feature matching

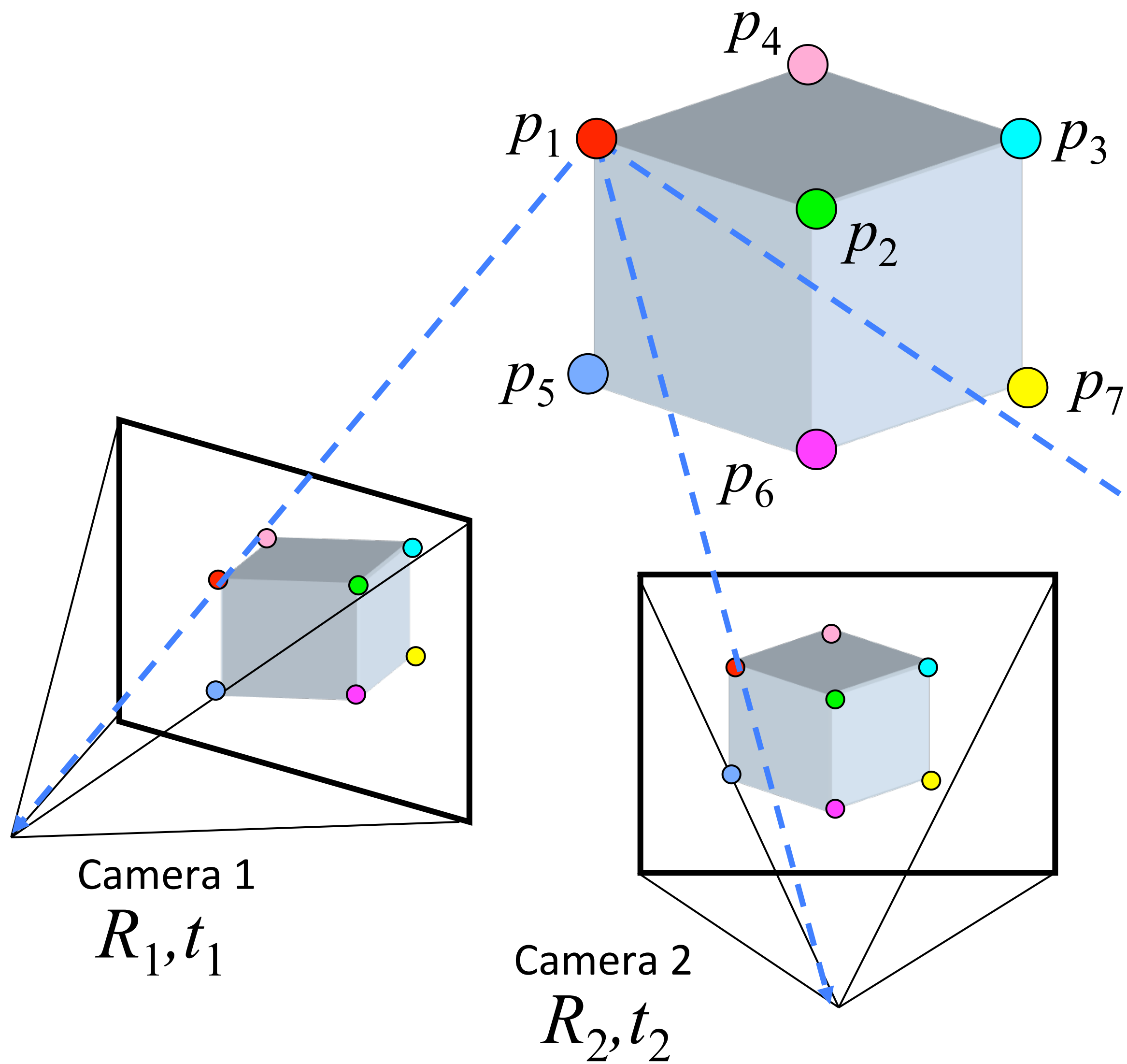
Refine matching using RANSAC to estimate fundamental matrix between each pair





Slide: N. Snavely

Structure from motion



minimize

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

Slide: N. Snavely

SfM objective function

Given point **x** and rotation and translation **R**, **t**

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \mathbf{R}\mathbf{x} + \mathbf{t} \qquad u' = \frac{fx'}{z'} \qquad v' = \frac{fy'}{z'} \qquad \begin{bmatrix} u' \\ v' \end{bmatrix} = \mathbf{P}(\mathbf{x}, \mathbf{R}, \mathbf{t})$$

Minimize sum of squared reprojection errors:

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

predicted
image location

observed
image location

Solving structure from motion

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)

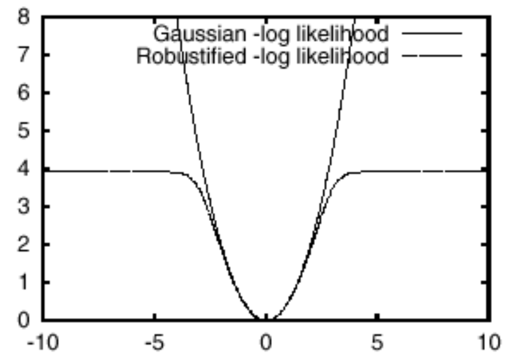
Many techniques use non-linear least-squares (NLLS) optimization (*bundle adjustment*)

- 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

Extensions to SfM

Can also solve for intrinsic parameters (focal length, radial distortion, etc.)

Can use a more robust function than squared error, to avoid fitting to outliers



For more information, see: Triggs, *et al*, "Bundle Adjustment – A Modern Synthesis", *Vision Algorithms* 2000.

Problem size

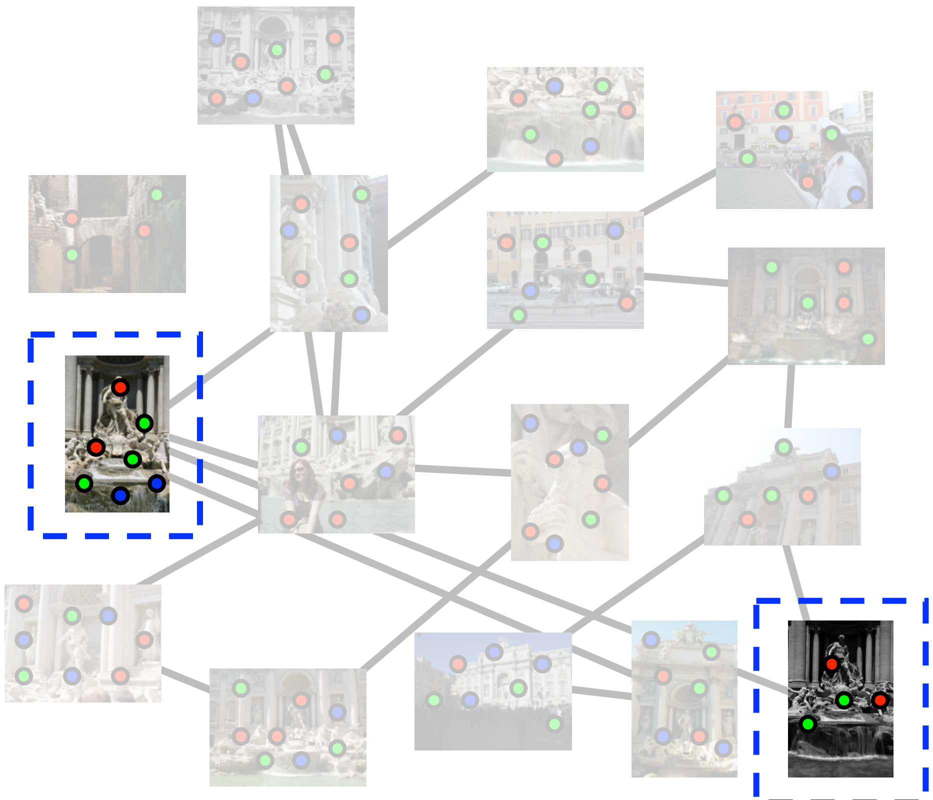
Trevi Fountain collection

466 input photos

+ > 100,000 3D points

= very large optimization problem

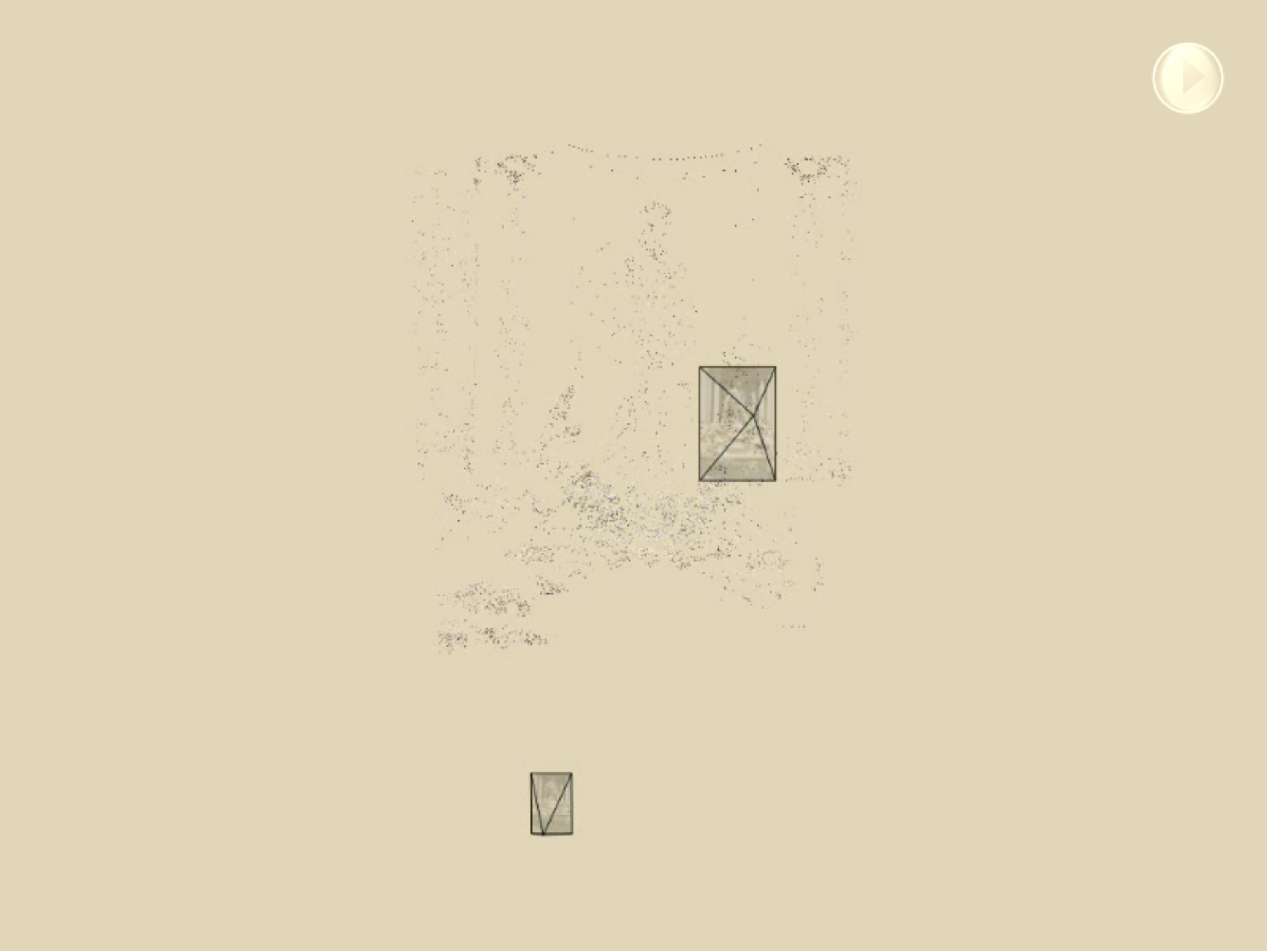
Incremental structure from motion



Incremental structure from motion

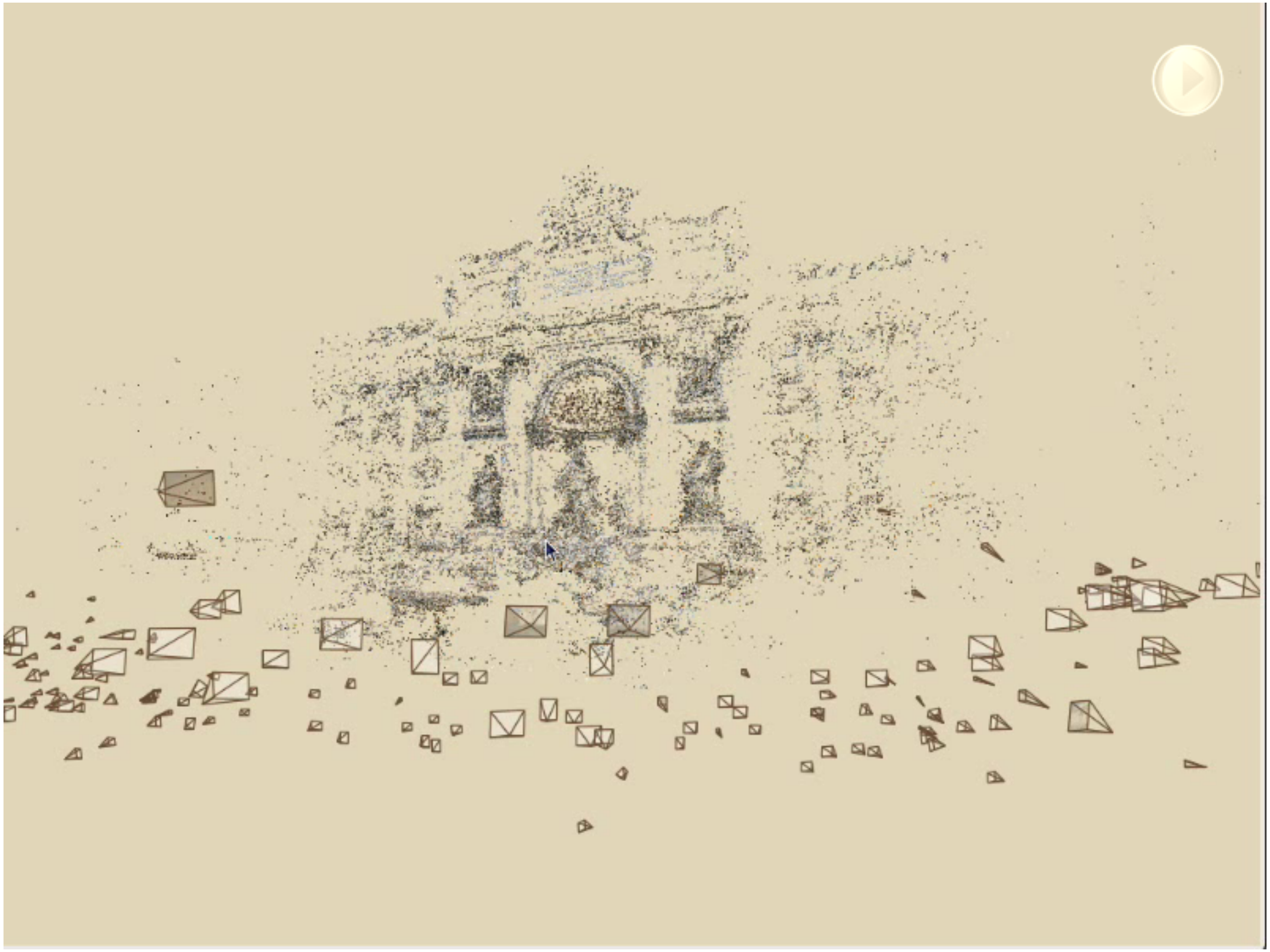
Slide: N. Snavely

Incremental structure from motion



Slide: N. Snavely

Photo Explorer



Slide: N. Snavely

KinectFusion: Real-Time Dense Surface Mapping and Tracking

by Richard.A Newcombe et al.

Presenting: Boaz Petersil

Video

Motivation

Augmented Reality

Robot Navigation

3d model scanning

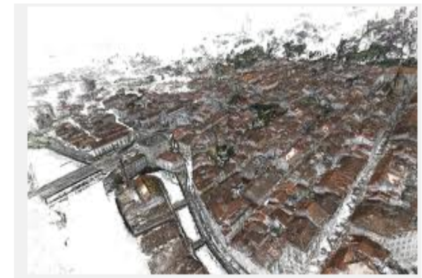
Etc..

[Slide: B. Petersil]

Related Work

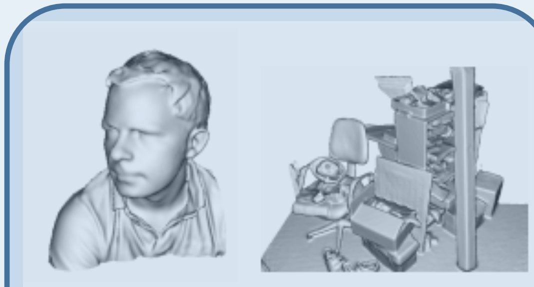
Tracking (&sparse Mapping)

PTAM

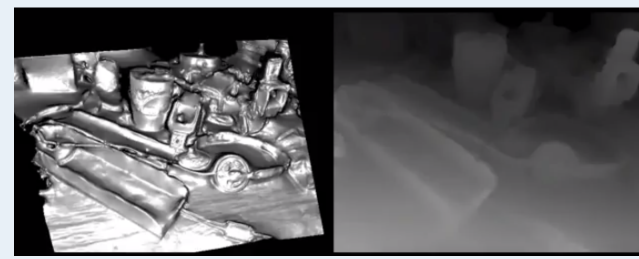


Bundle-adjustment(offline)

Tracking&Mapping



Kinect Fusion



DTAM (RGB cam!)

Dense Mapping

[Slide: B. Petersil]

Challenges

- Tracking Camera Precisely
- Fusing and De-noising Measurements
- Avoiding Drift
- Real-Time
- Low-Cost Hardware

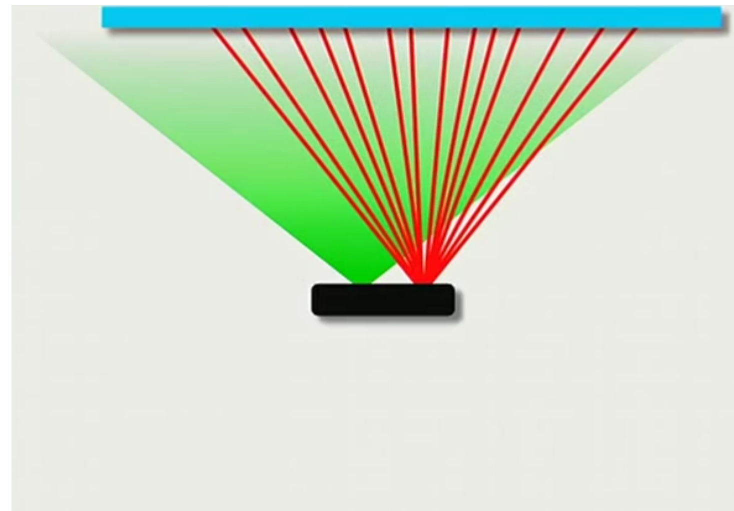
[Slide: B. Petersil]

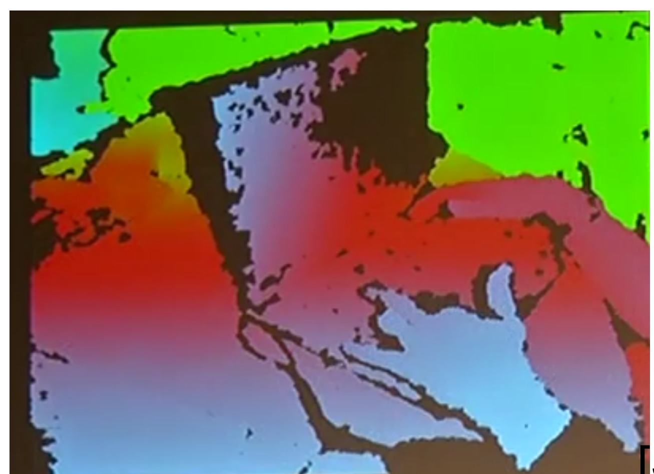
Proposed Solution

- Fast Optimization for Tracking, Due to High Frame Rate.
- Global Framework for fusing data
- Interleaving Tracking & Mapping
- Using Kinect to get Depth data (low cost)
- Using GPGPU to get Real-Time Performance (low cost)

[Slide: B. Petersil]

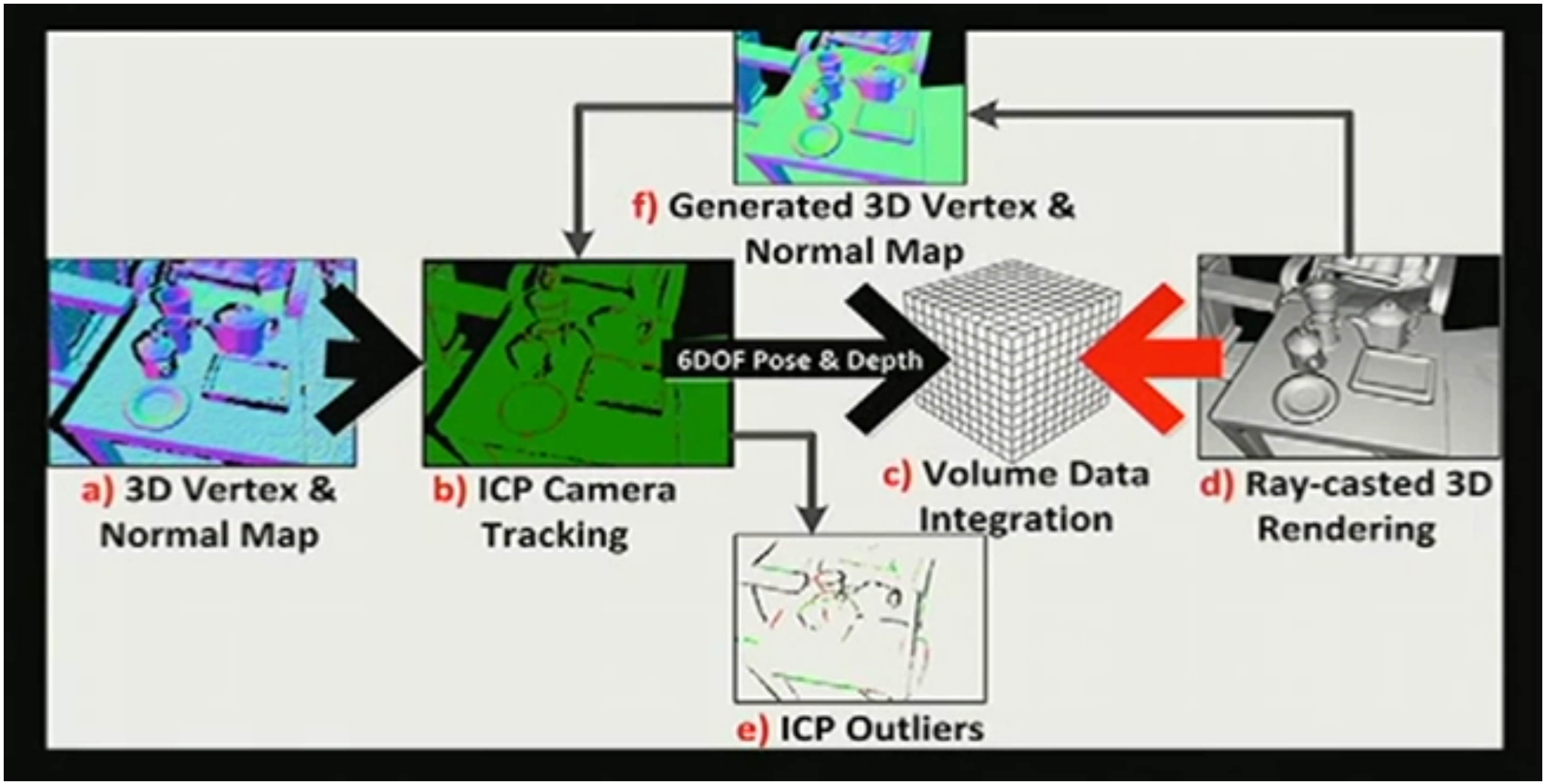
How does Kinect work?





[Slide: B. Petersil]

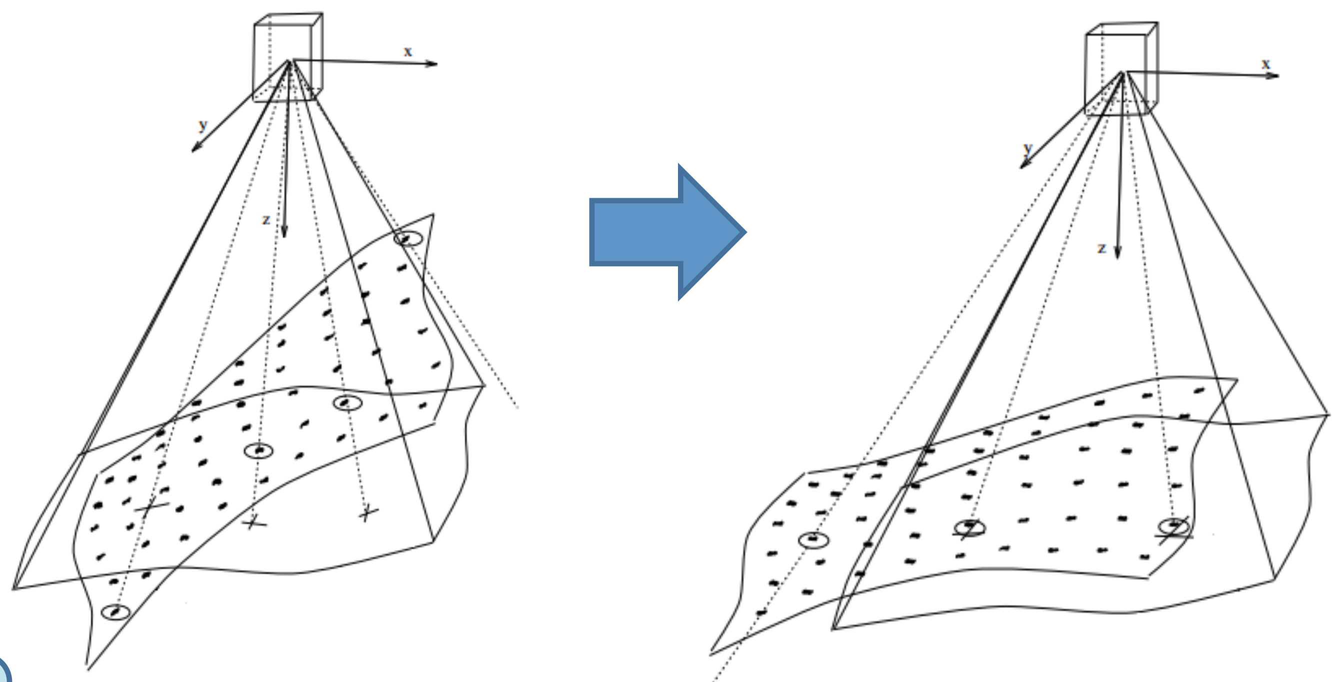
Method



[Slide: B. Petersil]

Tracking

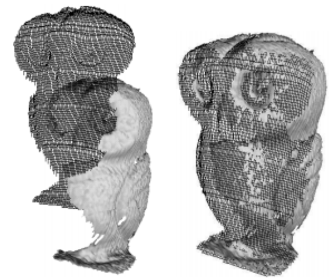
- Finding Camera position is the same as fitting frame’s Depth Map onto Model



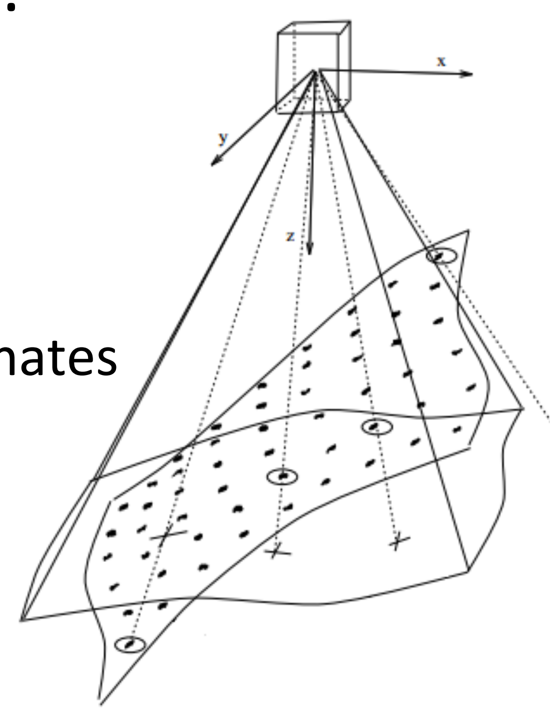
➔ Tracking
Mapping

[Slide: B. Petersil]

Tracking – ICP algorithm



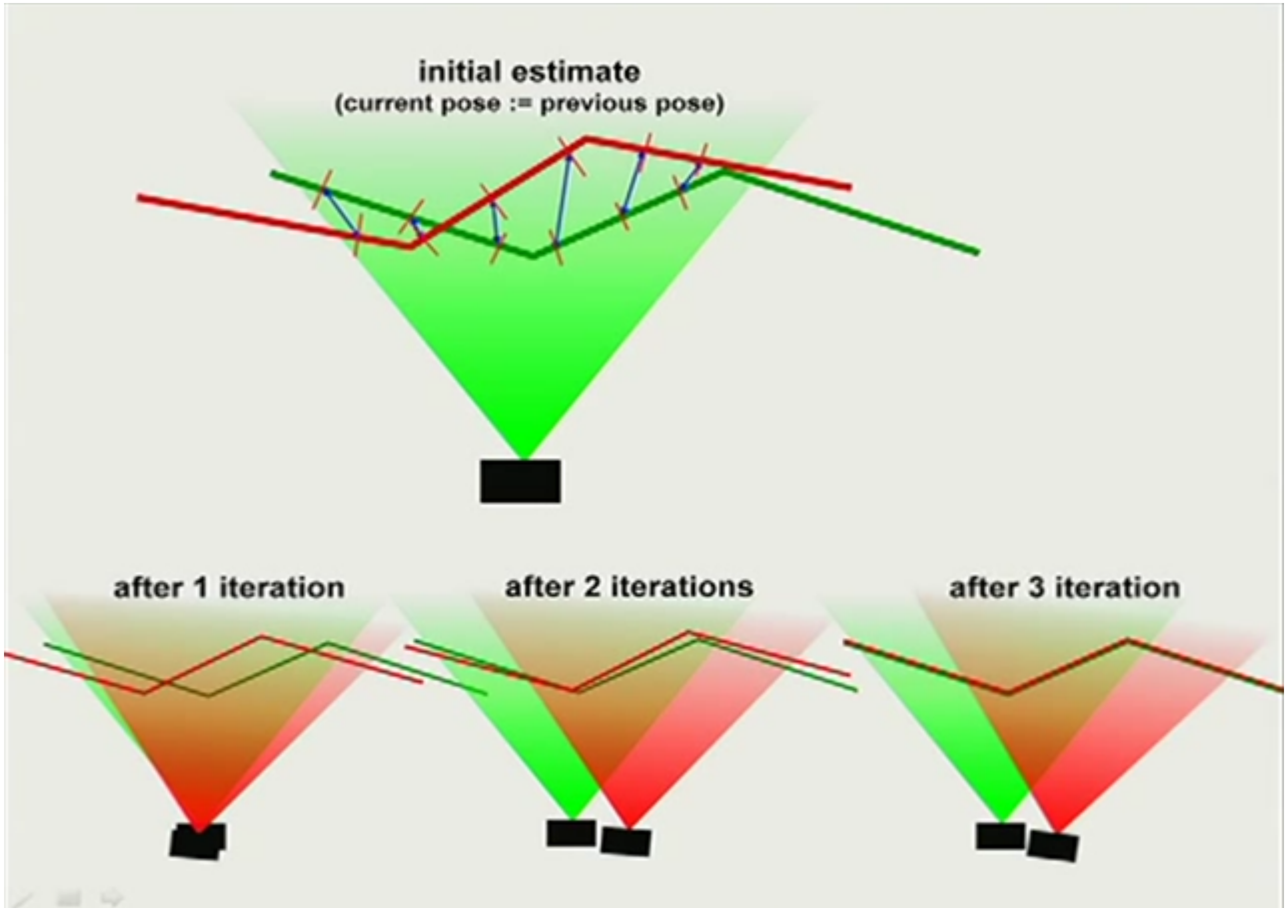
- icp = iterative closest point
- Goal: fit two 3d point sets
- Problem: What are the correspondences?
- Kinect fusion chosen solution:
  1) Start with T_0
  2) Project model onto camera
  3) Correspondences are points with same coordinates
  4) Find new T with Least - Squares
  5) Apply T, and repeat 2-5 until convergence



Tracking
Mapping

[Slide: B. Petersil]

Tracking – ICP algorithm



- Tracking
- Mapping

- Assumption: frame and model are roughly aligned.
- True because of high frame rate

[Slide: B. Petersil]

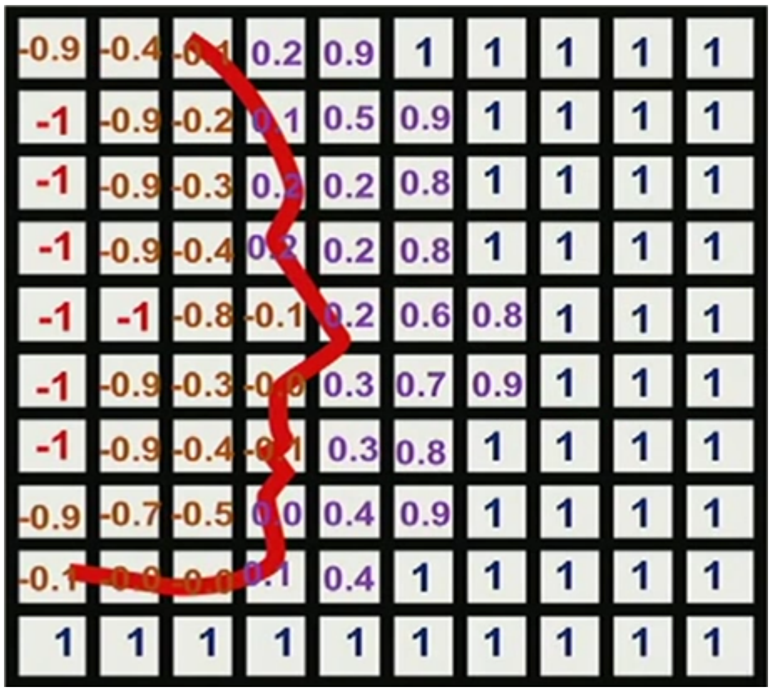
Mapping

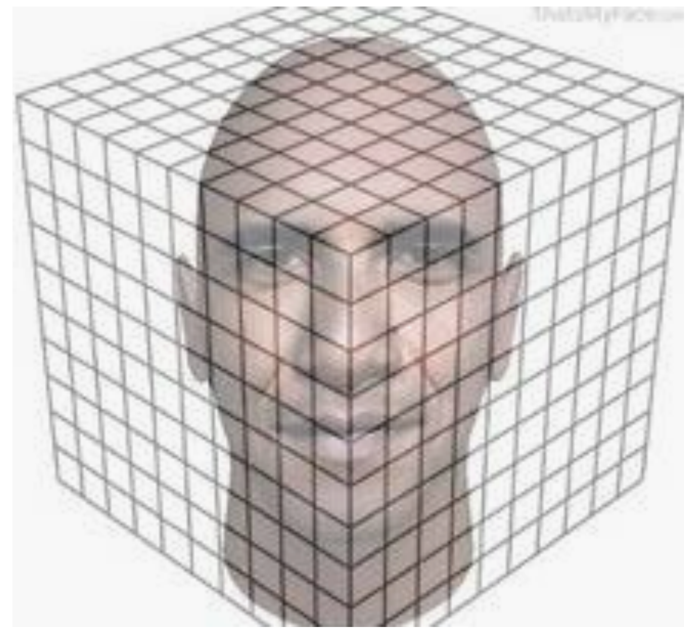
- Mapping is Fusing depth maps when camera poses are known
- Problems:
  - measurements are noisy
  - Depth maps have holes in them
- Solution:
  - using implicit surface representation
  - Fusing = estimating from all frames relevant

[Slide: B. Petersil]

Mapping – surface representation

- Surface is represented implicitly - using Truncated Signed Distance Function (TSDF)





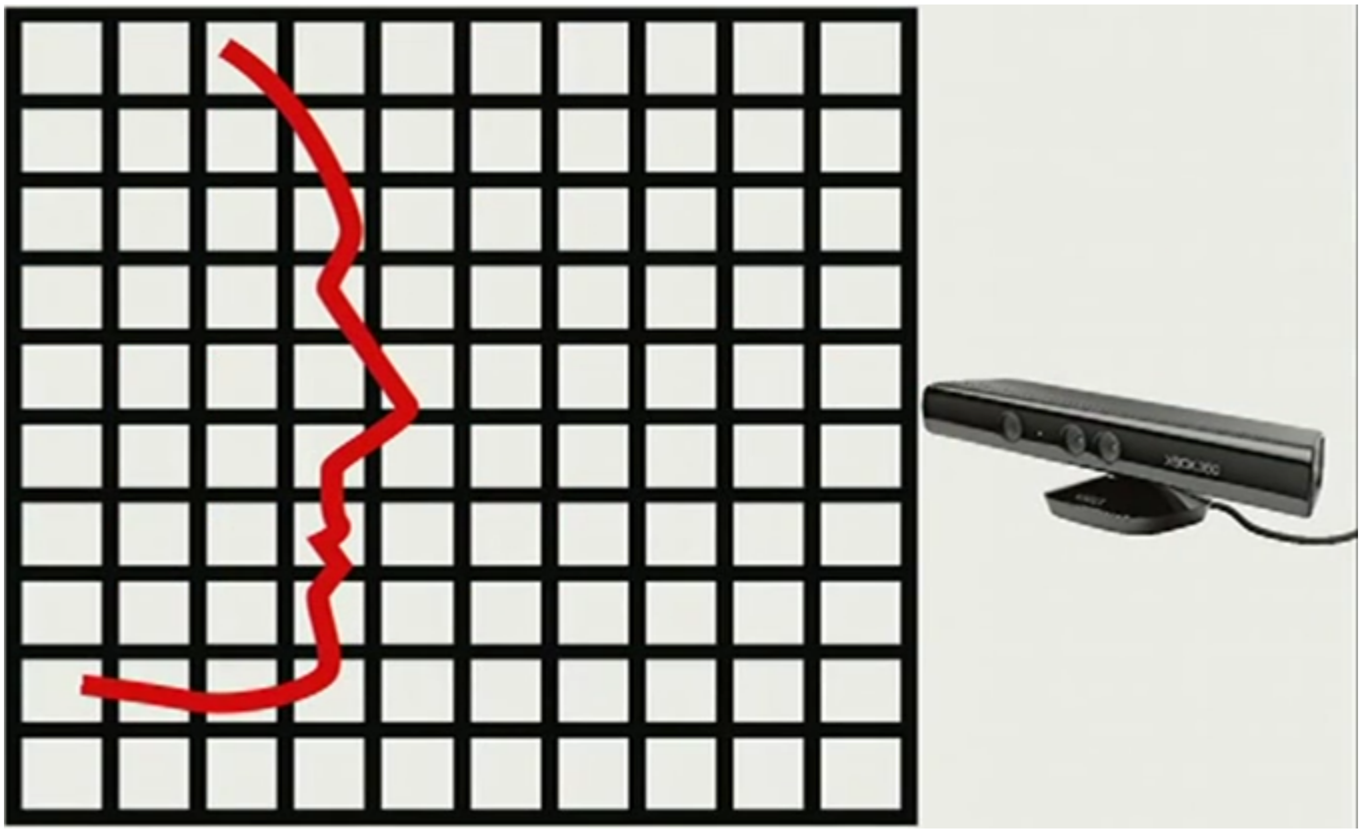
Voxel grid

Tracking
➔ Mapping

•Numbers in cells measure voxel distance to surface – D

[Slide: B. Petersil]

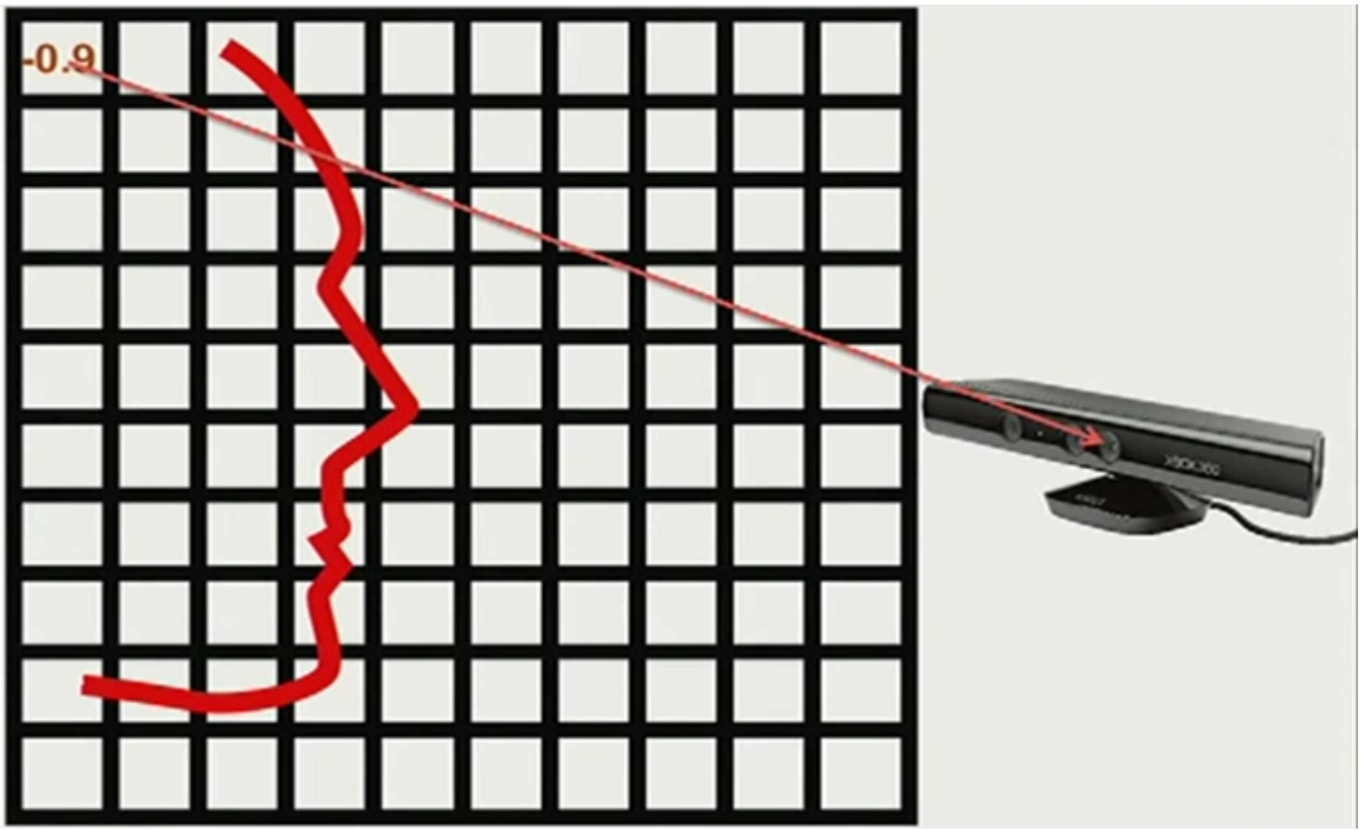
Mapping



Tracking
Mapping

[Slide: B. Petersil]

Mapping

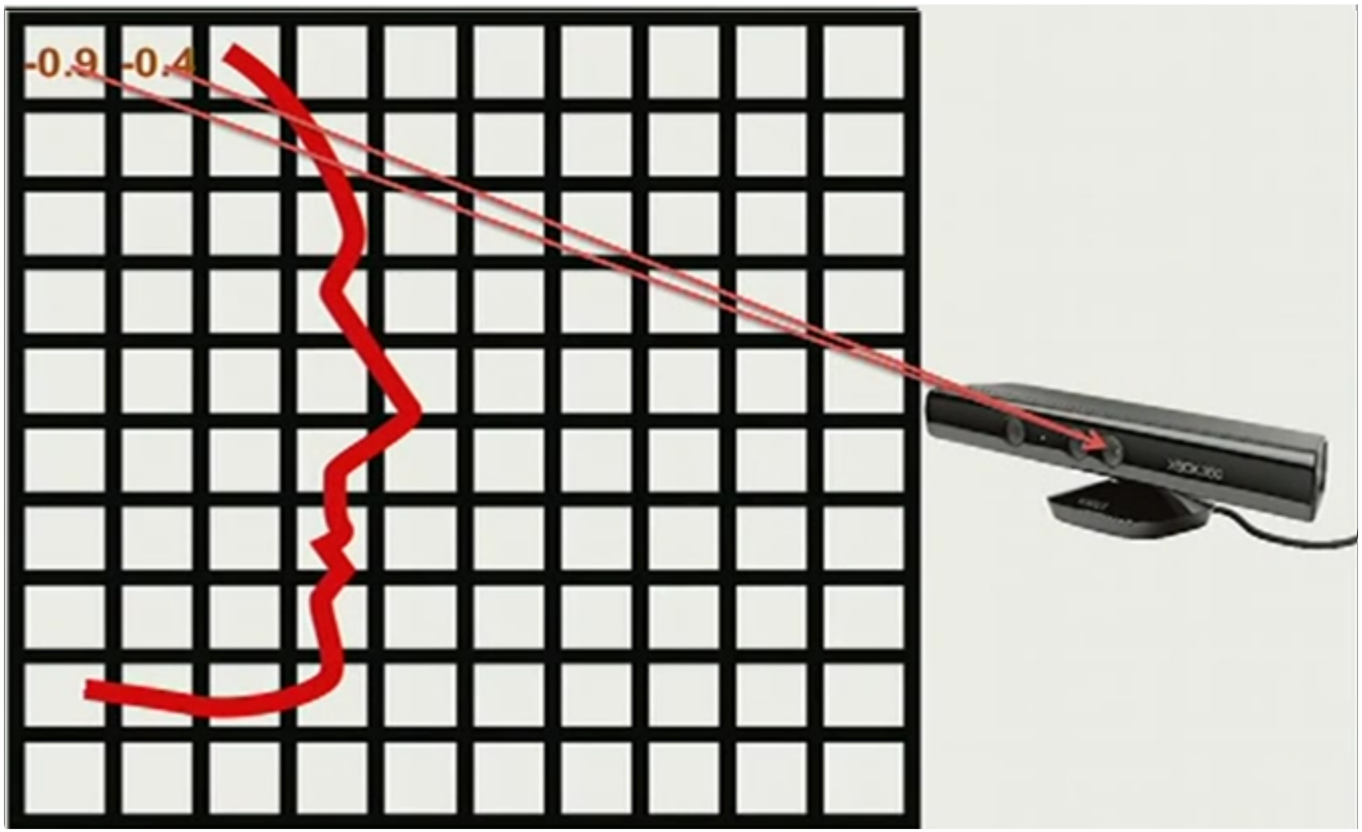


Tracking
➔ Mapping

d= [pixel depth] – [distance from sensor to voxel]

[Slide: B. Petersil]

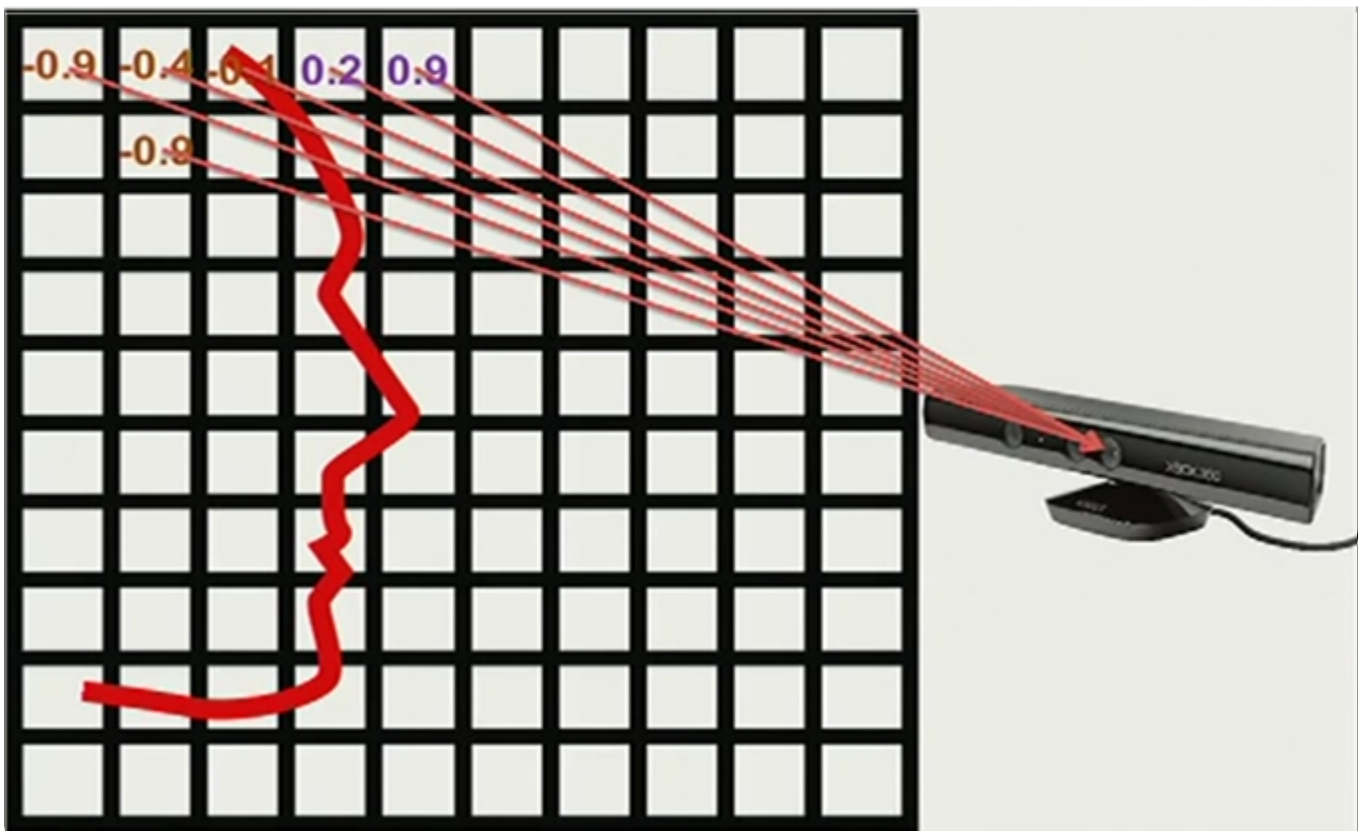
Mapping



Tracking
➔ Mapping

[Slide: B. Petersil]

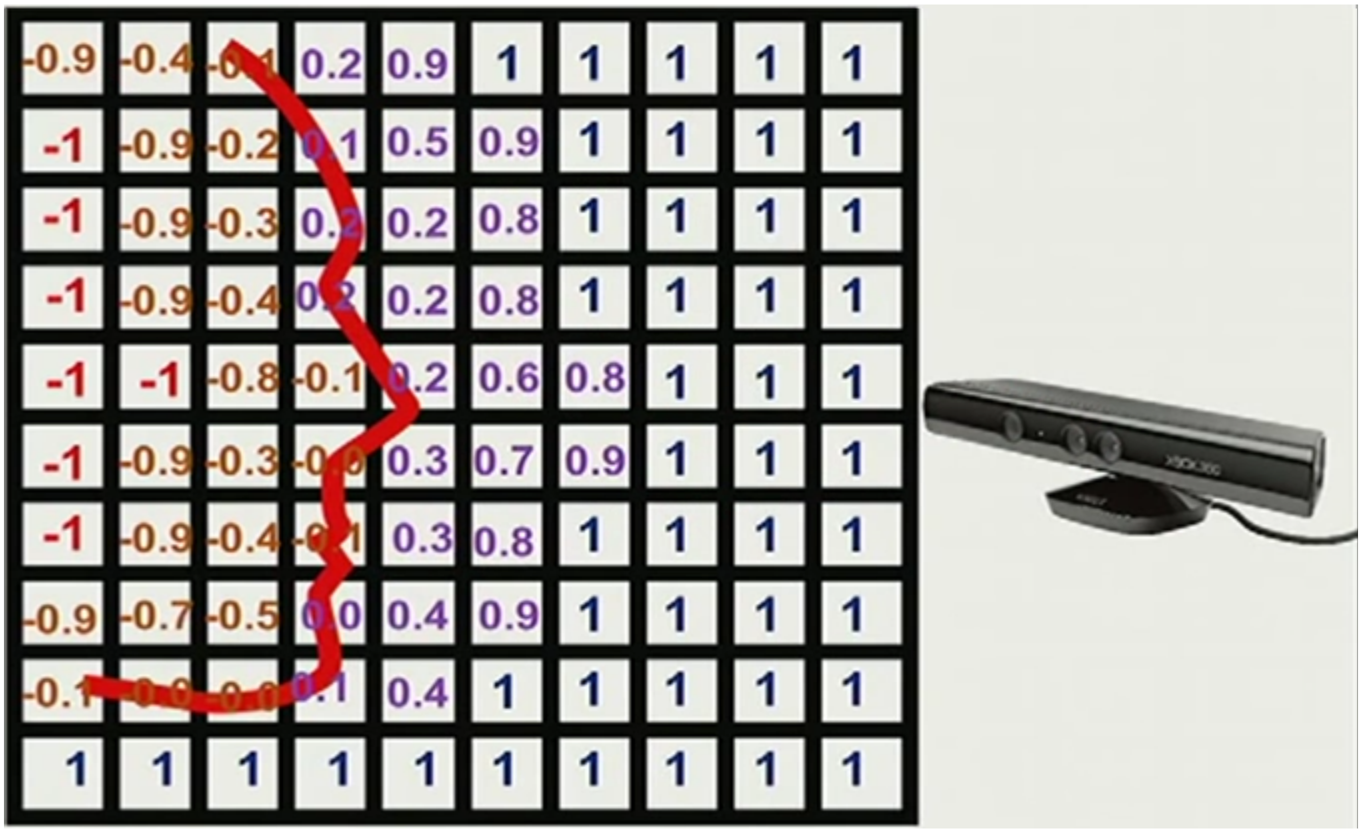
Mapping



Tracking
➔ Mapping

[Slide: B. Petersil]

Mapping

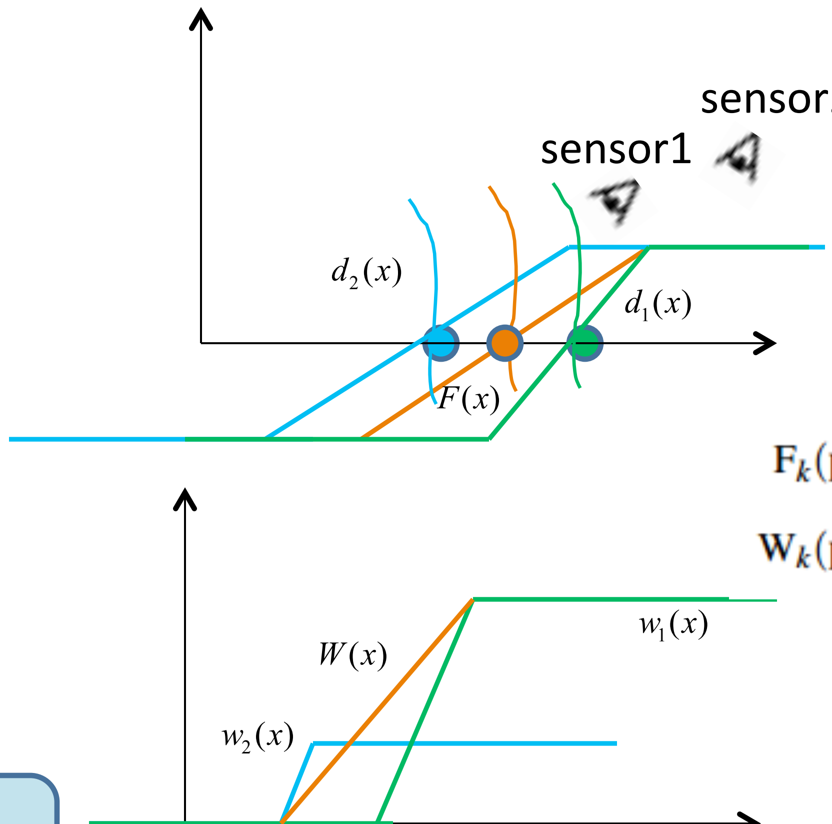


Tracking
➔ Mapping

[Slide: B. Petersil]

Mapping

- Each Voxel also has a weight W, proportional to grazing angle
- Voxel D is the weighted average of all measurements



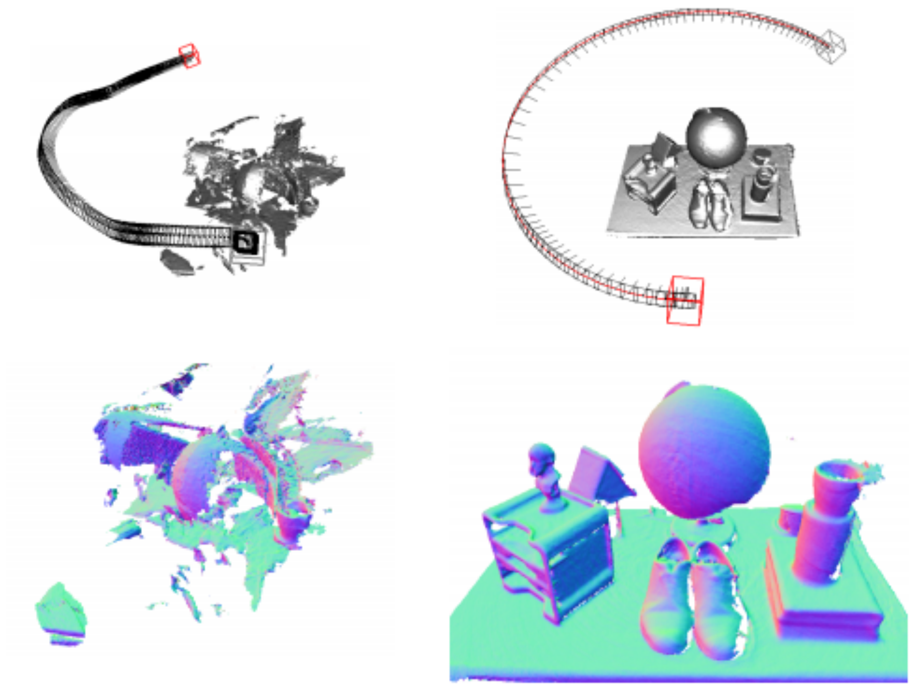
$$\mathbf{F}_k(\mathbf{p}) = \frac{\mathbf{W}_{k-1}(\mathbf{p})\mathbf{F}_{k-1}(\mathbf{p}) + \mathbf{W}_{\mathbf{R}_k}(\mathbf{p})\mathbf{F}_{\mathbf{R}_k}(\mathbf{p})}{\mathbf{W}_{k-1}(\mathbf{p}) + \mathbf{W}_{\mathbf{R}_k}(\mathbf{p})}$$

$$\mathbf{W}_k(\mathbf{p}) = \mathbf{W}_{k-1}(\mathbf{p}) + \mathbf{W}_{\mathbf{R}_k}(\mathbf{p})$$

[Slide: B. Petersil]

Handling drift

- Drift would have happened If tracking was done from frame to previous frame
- Tracking is done on built model



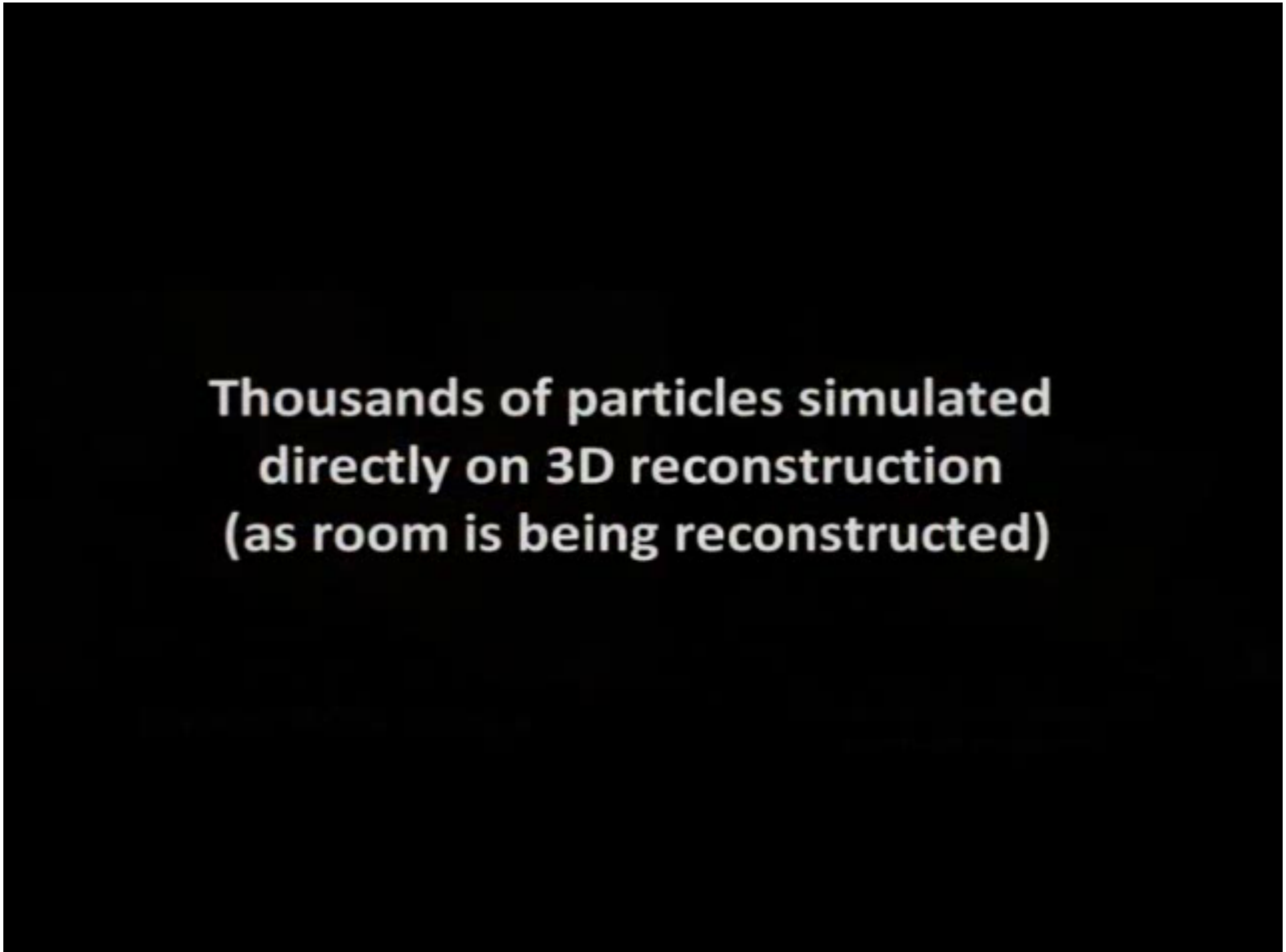
(a) Frame to frame tracking (b) Partial loop

➔ Tracking
➔ Mapping

[Slide: B. Petersil]

[Slide: B. Petersil]

Results & Applications



Pros & Cons

- Pros:
  - Really nice results!
    - Real time performance (30 HZ)
    - Dense model
    - No drift with local optimization
    - Robust to scene changes
  - Elegant solution
- Cons :
  - 3d grid can't be trivially up-scaled

[Slide: B. Petersil]

Limitations

- Doesn't work for large areas (Voxel-Grid)
- Doesn't work far away from objects (active ranging)
- Doesn't work out-doors (IR)
- Requires powerful Graphics card
- Uses lots of battery (active ranging)
- Only one sensor at a time

[Slide: B. Petersil]