CS 558: Computer Vision
8^{th} Set of Notes

Instructor: Philippos Mordohai
Webpage: www.cs.stevens.edu/~mordohai
E-mail: Philippos.Mordohai@stevens.edu
Office: Lieb 215

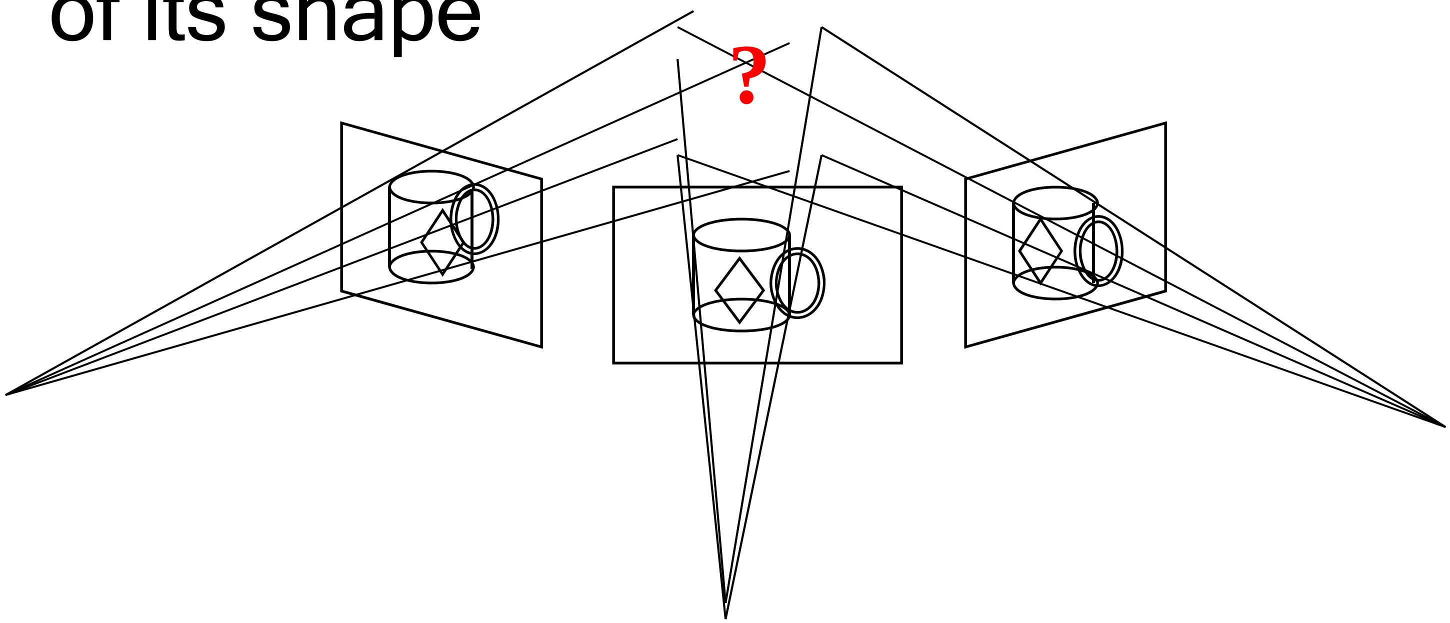
Overview

- Stereo Matching
  - Partially based on slides by M. Bleyer, P. Fua, S. Seitz and R. Szeliski
- Structure from Motion
  - Partially based on slides by S. Lazebnik, S. Setiz, N. Snavely and R. Szeliski

Stereo Matching

Stereo Matching

- Given two or more images of the same scene or object, compute a representation of its shape

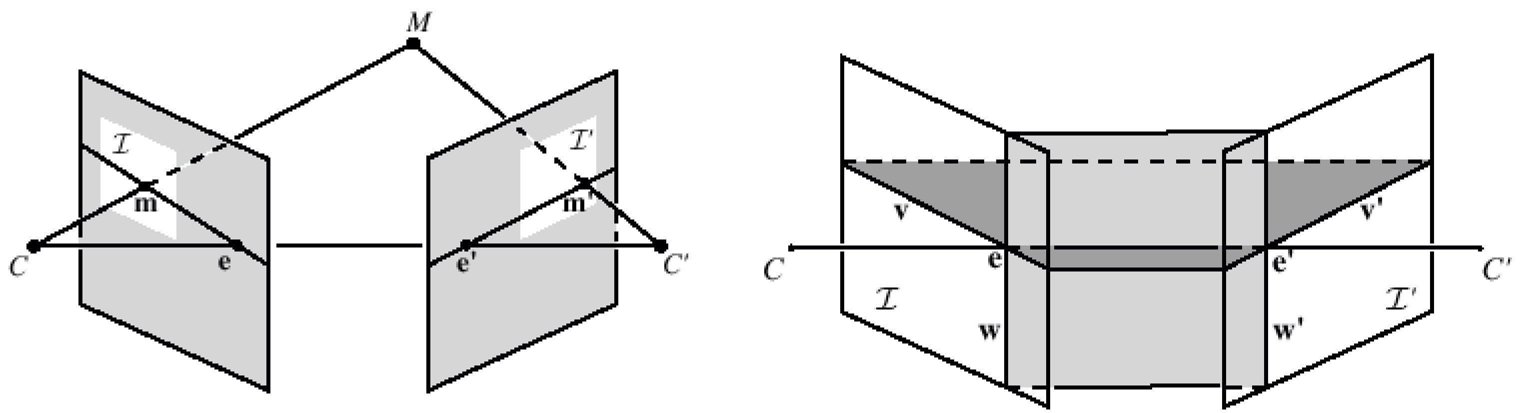


Stereo Matching

- What are some possible algorithms?
  - match "features" and interpolate
  - match edges and interpolate
  - match all pixels with windows

Rectification

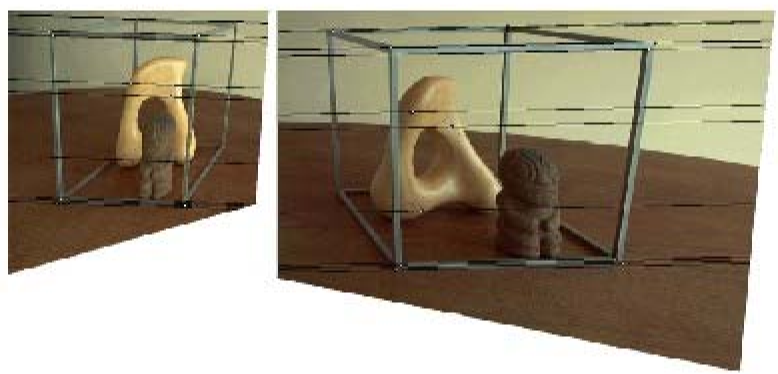
- Project each image onto same plane, which is parallel to the baseline
- Resample lines (and shear/stretch) to place lines in correspondence, and minimize distortion



- Take rectification for granted in this course

Rectification

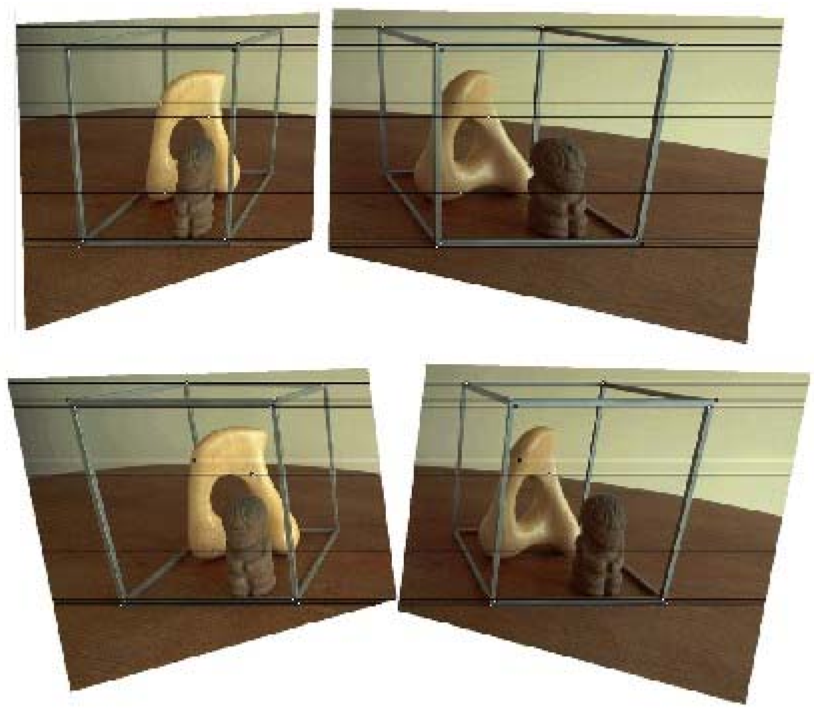
(a) Original image pair overlayed with several epipolar lines.



(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

BAD!

Rectification



(c) Image pair transformed by the similarity $\mathbf{H}_r$ and $\mathbf{H}'_r$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

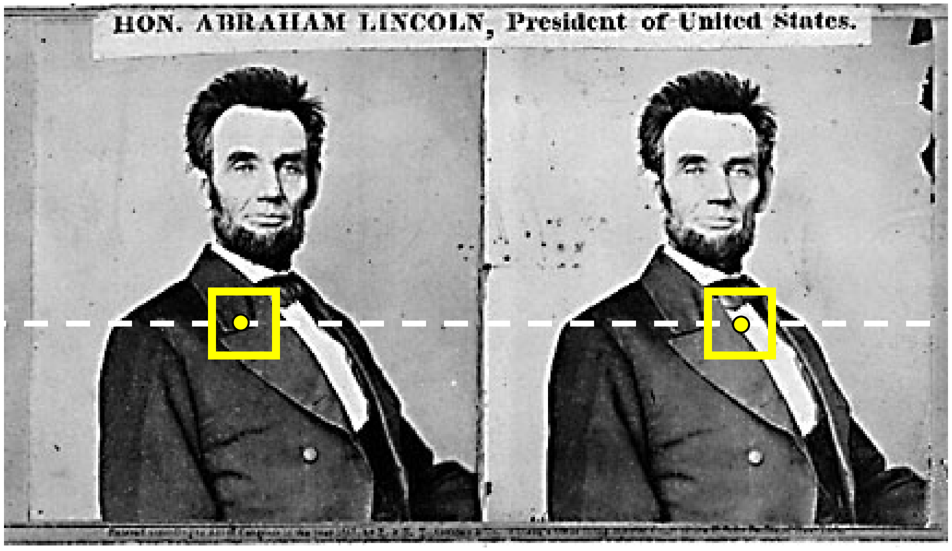
(d) Final image rectification after shearing transform $\mathbf{H}_s$ and $\mathbf{H}'_s$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

GOOD!

Finding Correspondences

- Apply feature matching criterion at *all* pixels simultaneously
- Search only over epipolar lines (many fewer candidate positions)

Basic Stereo Algorithm



For each epipolar line

For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match ***windows***

Disparity

- Disparity d is the difference between the x coordinates of corresponding pixels in the left and right image

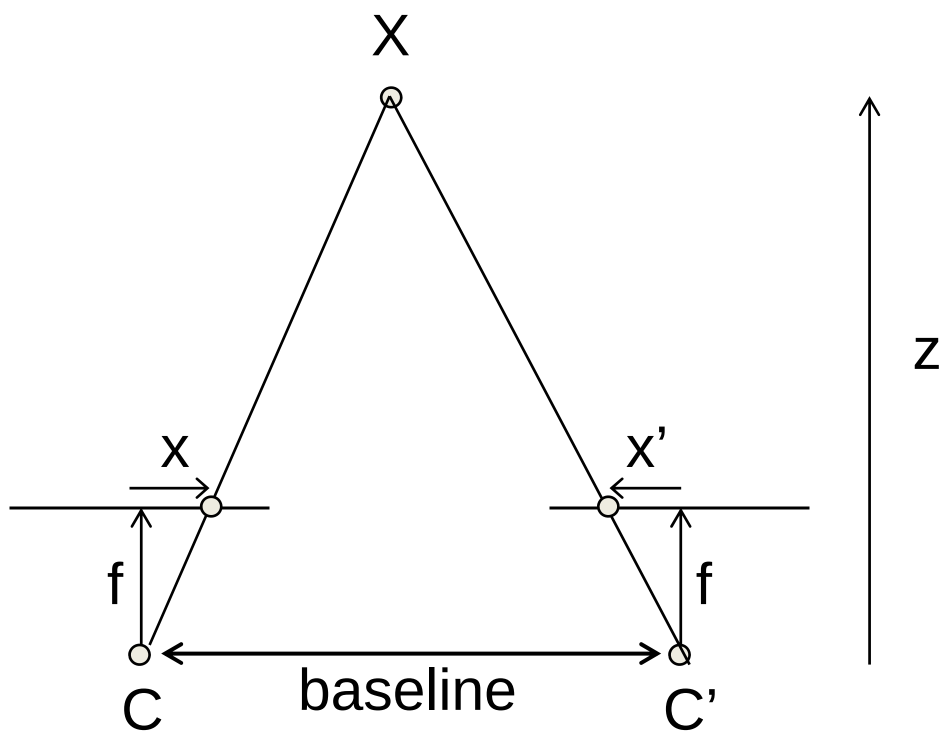
$$d = x_L - x_R$$

- Disparity is inversely proportional to depth

$$Z = \frac{bf}{d}$$

Stereo Reconstruction

$$Z = \frac{bf}{d}$$

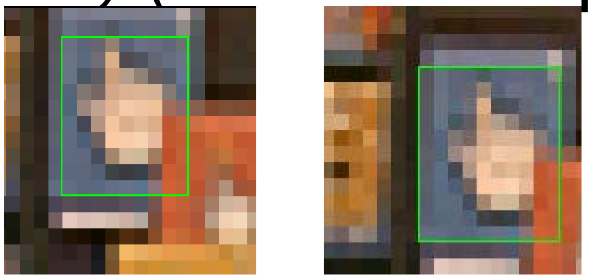


Finding Correspondences

- How do we determine correspondences?
  - *block matching* or *SSD* (sum squared differences)

$$SSD(x, y; d) = \sum_{(x',y')\in N(x,y)} [\, I_L(x', y') - I_R(x' - d, y')]^2$$

  - *d* is the *disparity* (horizontal displacement)

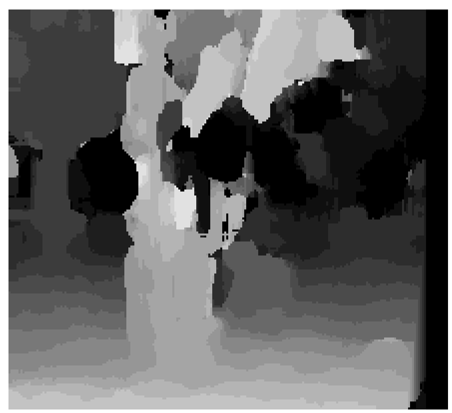


- How big should the neighborhood be?

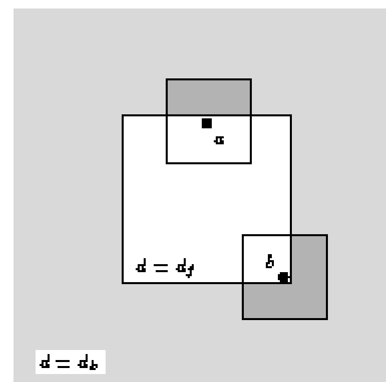
Neighborhood size

- Smaller neighborhood: more details
- Larger neighborhood: fewer isolated mistakes

w = 3



w = 20



Challenges

- Ill-posed inverse problem
  - Recover 3-D structure from 2-D information
- Difficulties
  - Uniform regions
  - Half-occluded pixels
  - Repeated patterns

Pixel Dissimilarity

- Sum of Squared Differences of intensities (SSD)

$$SSD(x, y; d) = \sum_{(x',y') \in \mathrm{N}(x,y)} [I_L(x', y') - I_R(x' - d, y')]^2$$

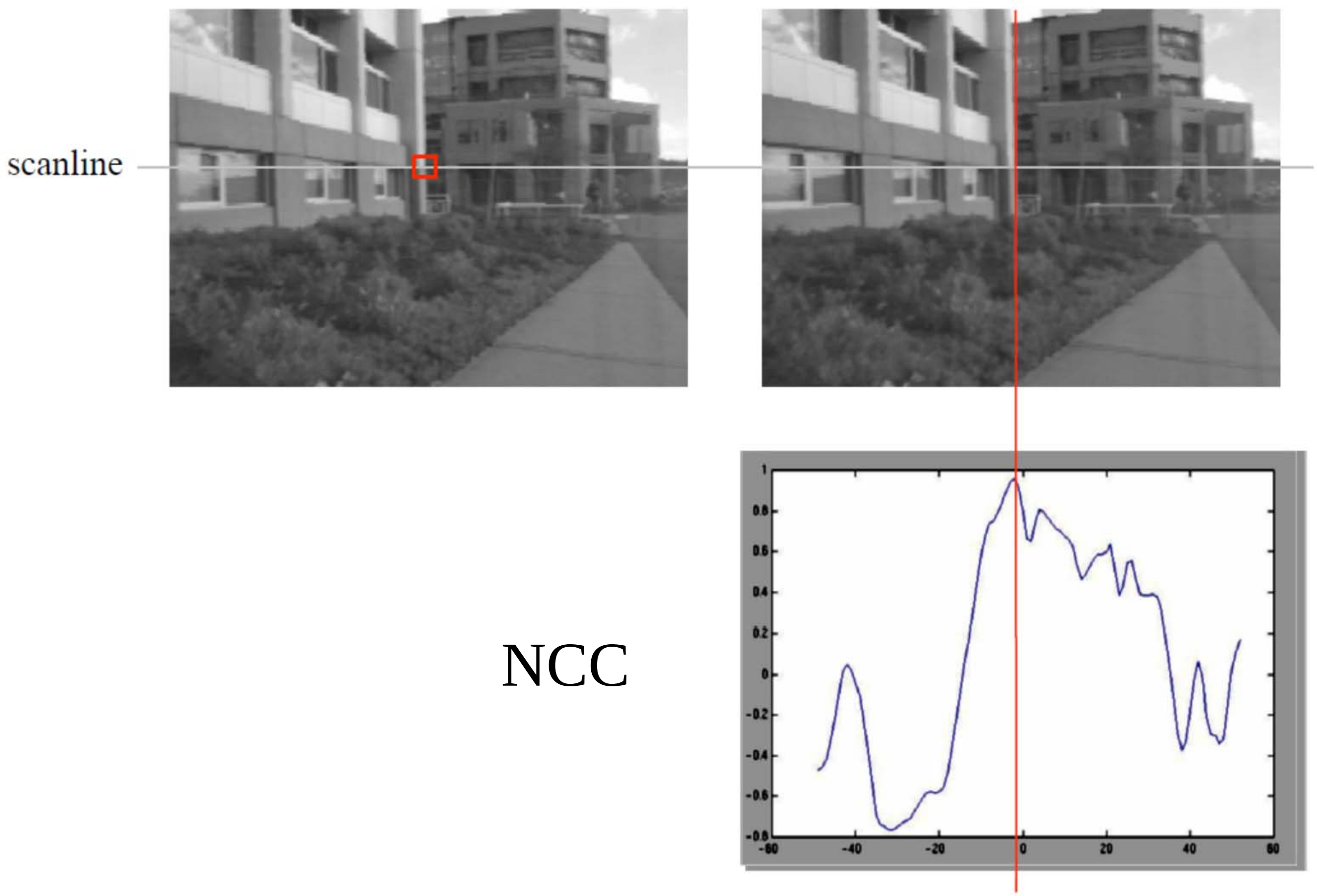
- Sum of Absolute Differences of intensities (SAD)

$$SAD(x, y; d) = \sum_{(x',y') \in \mathrm{N}(x,y)} |I_L(x', y') - I_R(x' - d, y')|$$

- Zero-mean Normalized Cross-correlation (NCC)

$$NCC(x, y, d) = \frac{\sum_{i \in W} (I_L(x_i, y_i) - \mu_L)(I_R(x_i - d, y_i) - \mu_R)}{\sigma_L \sigma_R}$$

Cost/Score Curve



Locally Adaptive Support

Apply weights to contributions of neighboring pixels according to similarity and proximity

(a) left support window (b) right support window (c) color difference between (a) and (b)

Locally Adaptive Support

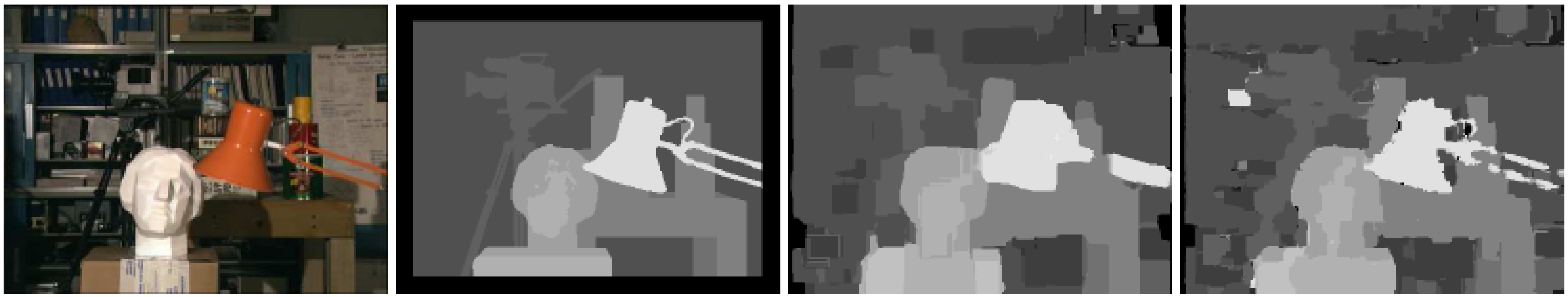
- Similarity in CIE Lab color space:

$$\Delta c_{pq} = \sqrt{(L_p - L_q)^2 + (a_p - a_q)^2 + (b_p - b_q)^2}$$

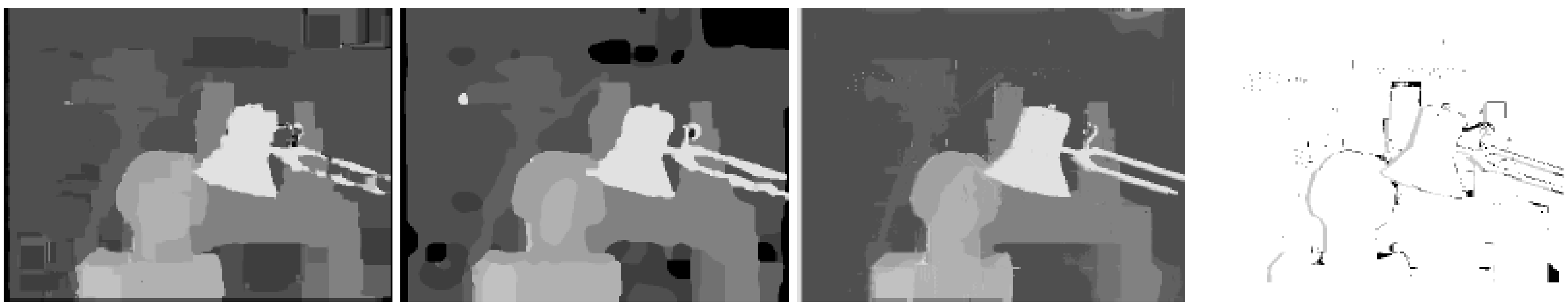
- Proximity: Euclidean distance

- Weights: $w(p,q) = k \cdot \exp\left(-(\frac{\Delta c_{pq}}{\gamma_c} + \frac{\Delta g_{pq}}{\gamma_p})\right)$

Locally Adaptive Support: Results



(a) left image (b) ground truth (c) shiftable win. [7] (d) compact win. [3]



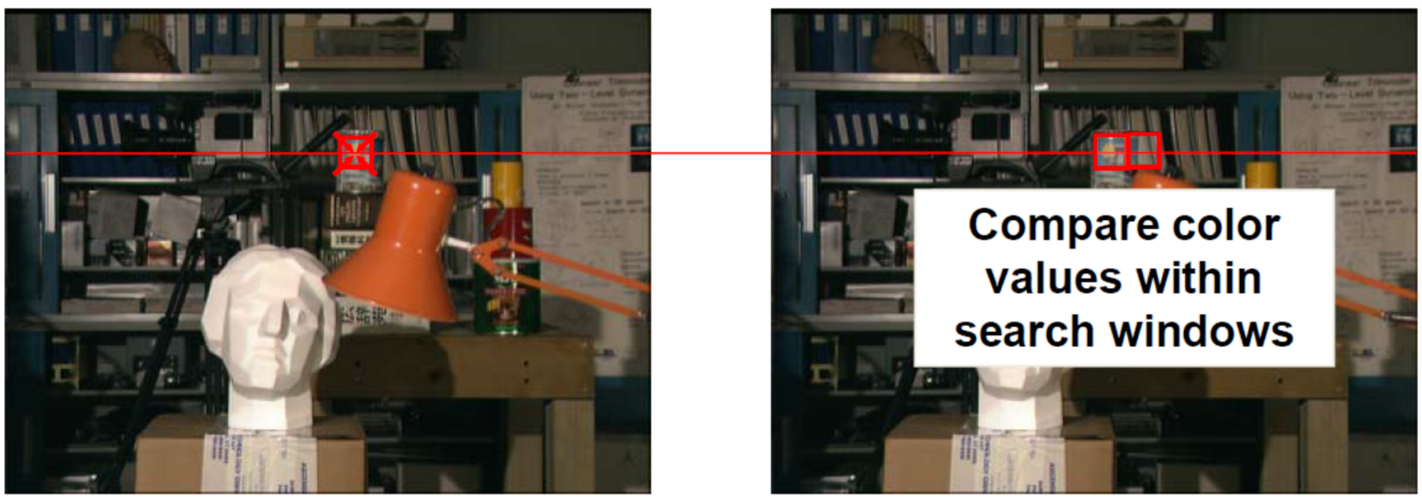
(e) variable win. [4] (f) Bay. diff. [19] (g) our result (h) bad pixels (error > 1)

Naïve Stereo Algorithm

- For each pixel p of the left image:
  - Compare color of p against the color of each pixel on the same horizontal scanline in the right image.
  - Select the pixel of most similar color as matching point

Window-Based Matching

- Instead of matching single pixels, center a small window on a pixel and match the whole window in the right image



(a) Left image (b) Right image

Window-Based Matching

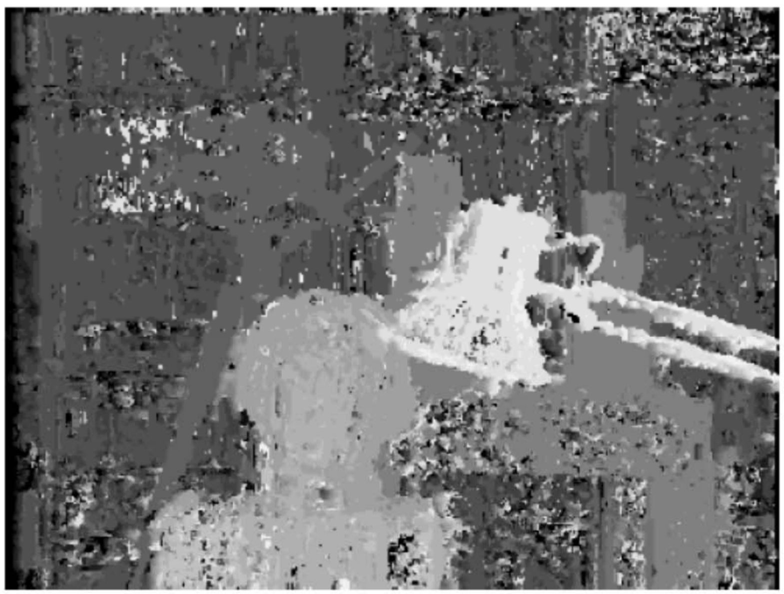
- the disparity d_p of a pixel p in the left image is computed as

$$d_p = \underset{0 \leq d \leq d\max}{\arg\min} \sum_{q \in Wp} c(q, q - d)$$

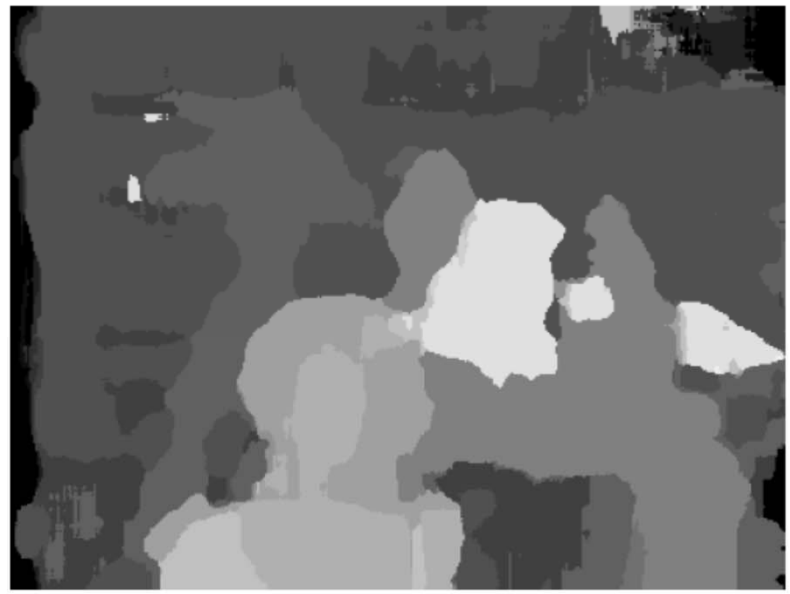
- where
  - argmin returns the value at which the function takes a minimum
  - d_{max} is a parameter defining the maximum disparity (search range)
  - W_p is the set of all pixels inside the window centered on p
  - c(p,q) is a function that computes the color difference between a pixel p of the left and a pixel q of the right image

Results

- The window size is a crucial parameter



Window size = 3x3 pixels



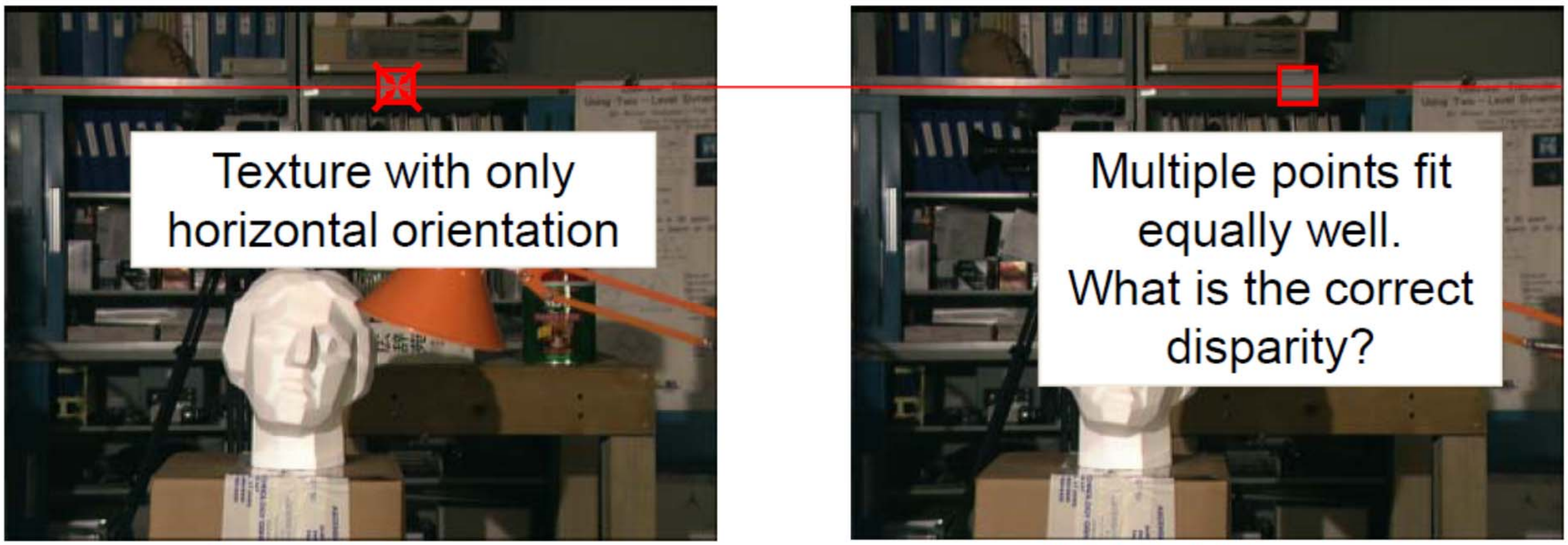
Window size = 21x21 pixels

Untextured Regions

(a) Left image (b) Right image

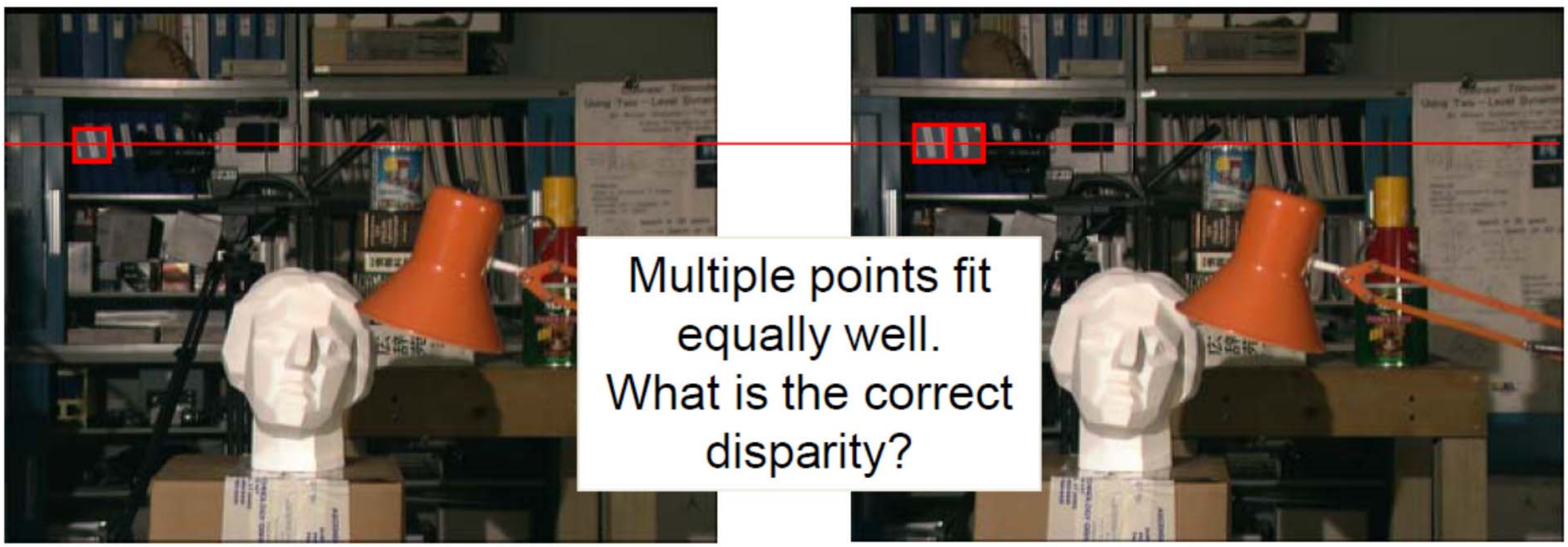
Aperture Problem

- There needs to be a certain amount of texture with vertical orientation



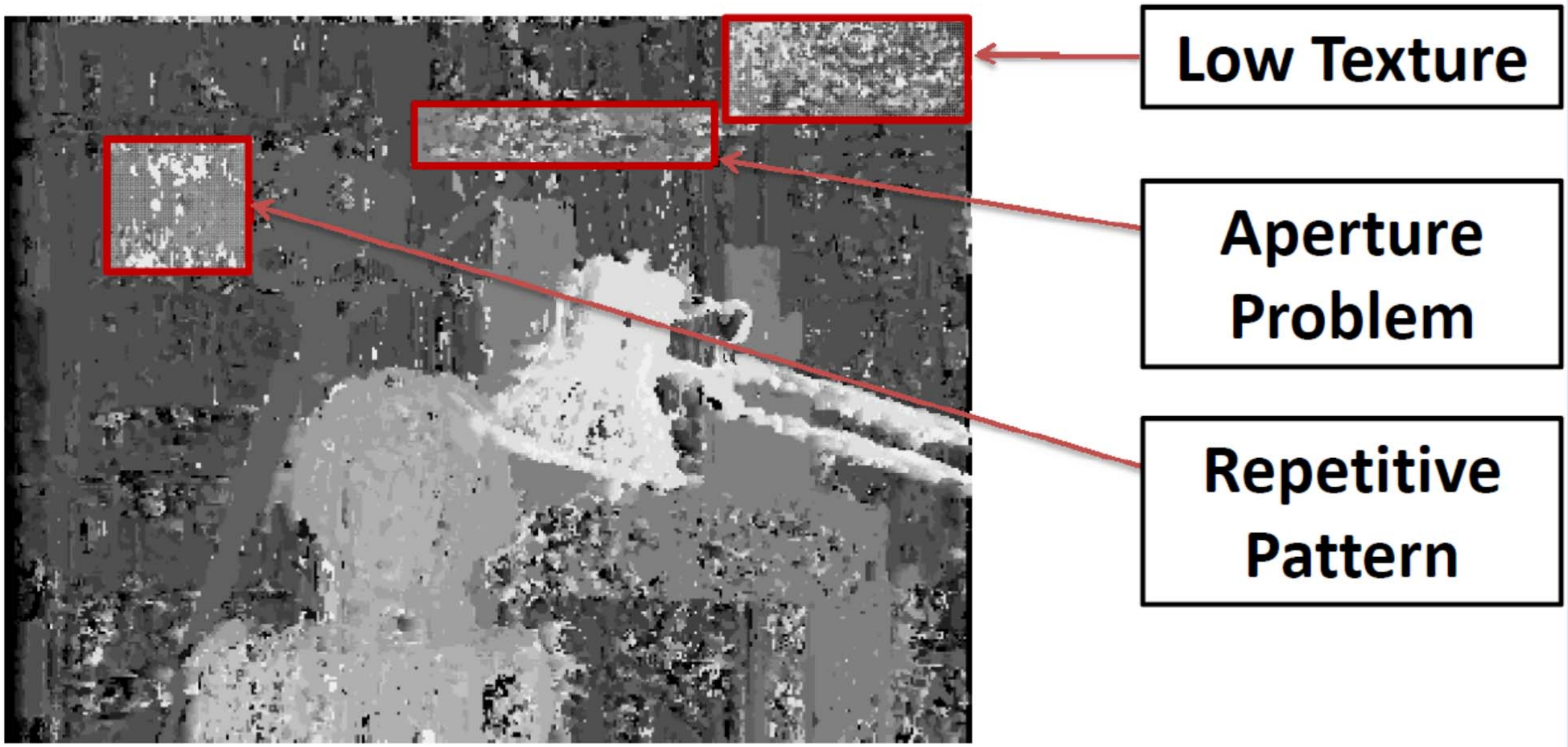
(a) Left image (b) Right image

Repetitive Patterns



(a) Left image (b) Right image

Effects of these Problems



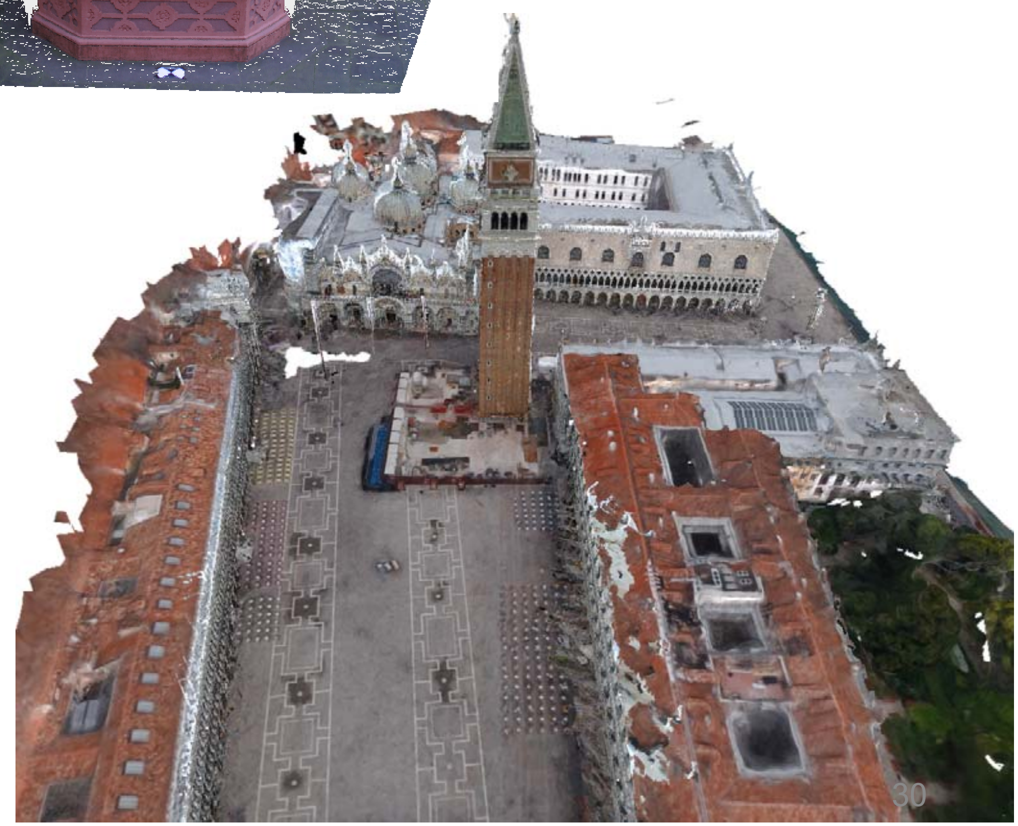
Window size = 3x3 pixels

Stereo Matching Summary

- One of fundamental computer vision problems
- A large variety of methods have been published
- Key idea: use global optimization to take into account more information than individual pixels
- See
  - http://vision.middlebury.edu/stereo/eval3/
  - http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo

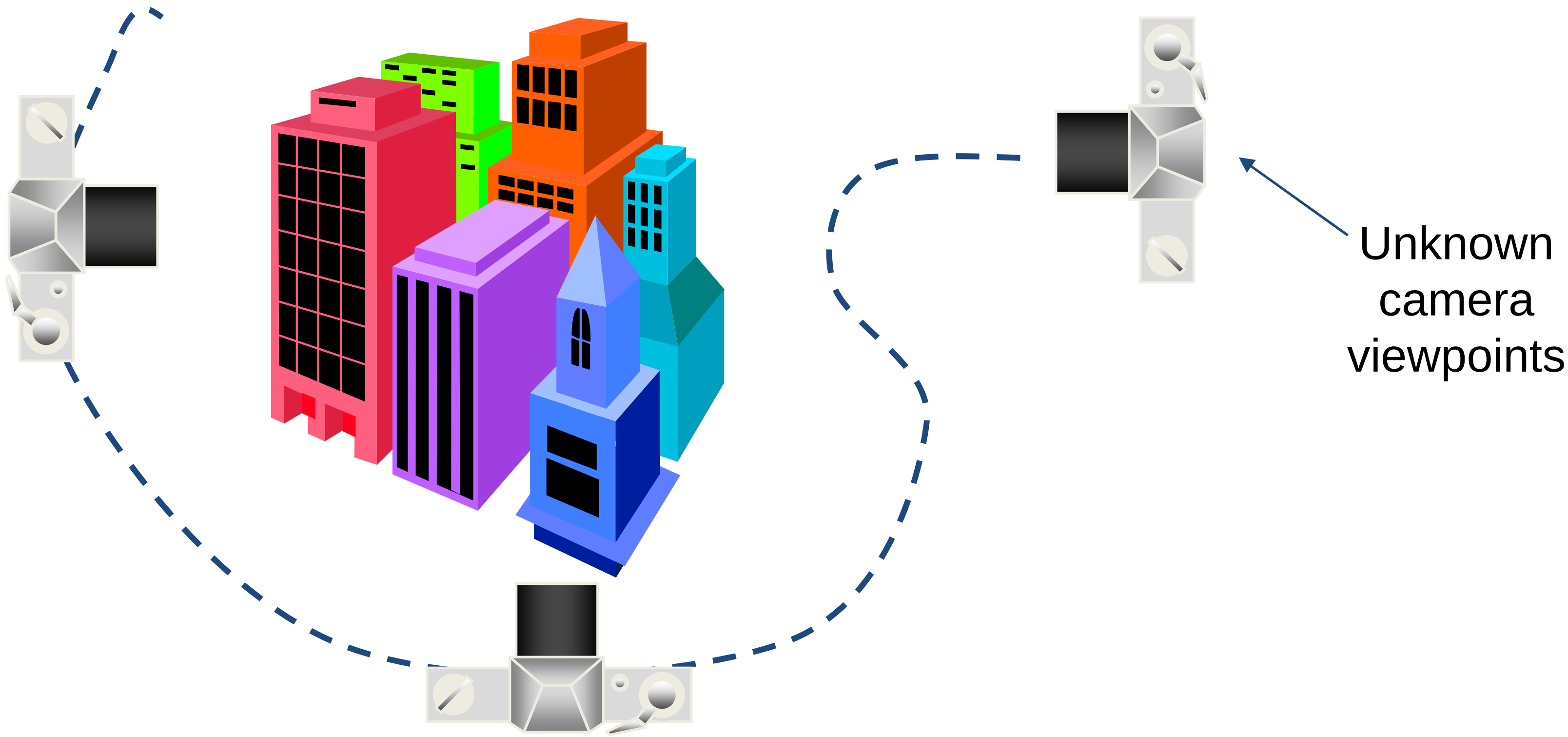
Multi-View Stereo

- See CS 532



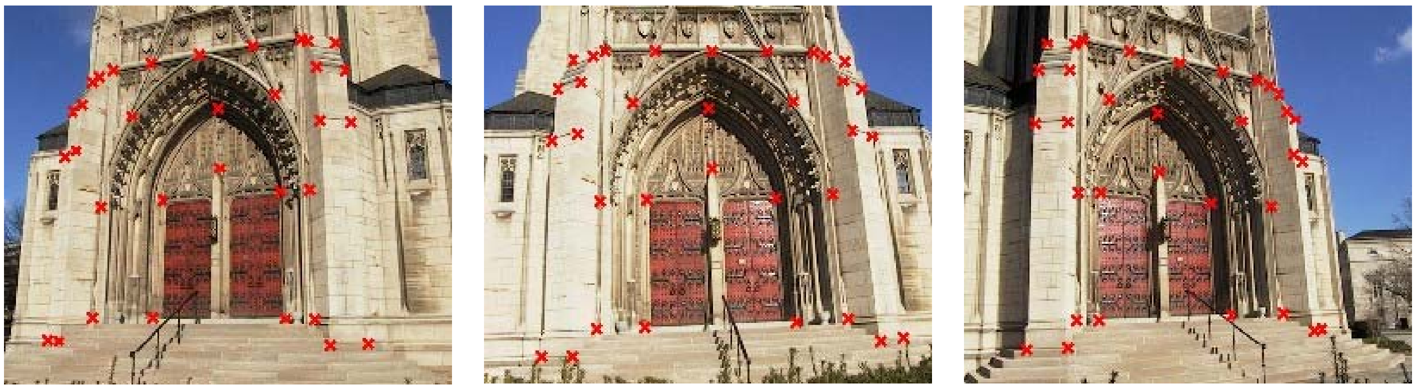
Structure from Motion

Structure from Motion



- Reconstruct
  - Scene geometry
  - Camera motion

Input: Feature Tracks



- Detect good features
  - corners, line segments
- Find correspondences between frames
  - Lucas & Kanade-style motion estimation
  - window-based correlation

Structure from Motion

- Given many points in *correspondence* across several images, $\{(u_{ij}, v_{ij})\}$, simultaneously compute the 3D location $\mathbf{x}_i$ and camera (or *motion*) parameters $(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j)$

$$\begin{aligned} \hat{u}_{ij} &= f(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i}) \\ \hat{v}_{ij} &= g(\boxed{\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i}) \end{aligned}$$

- Two main variants: calibrated, and uncalibrated (sometimes associated with Euclidean and projective reconstructions)

Number of Constraints

$$\hat{u}_{ij} = f(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$
$$\hat{v}_{ij} = g(\mathrm{K}, \mathrm{R}_j, \mathrm{t}_j, \mathrm{x}_i)$$

- How many points do we need to match?
- 2 frames:

  (**R**,**t**): 5 dof + 3n point locations $\leq$
  4n point measurements $\Rightarrow$ n $\geq$ 5

- k frames:

  6(k–1)-1 + 3n $\leq$ 2kn
- always want to use many more

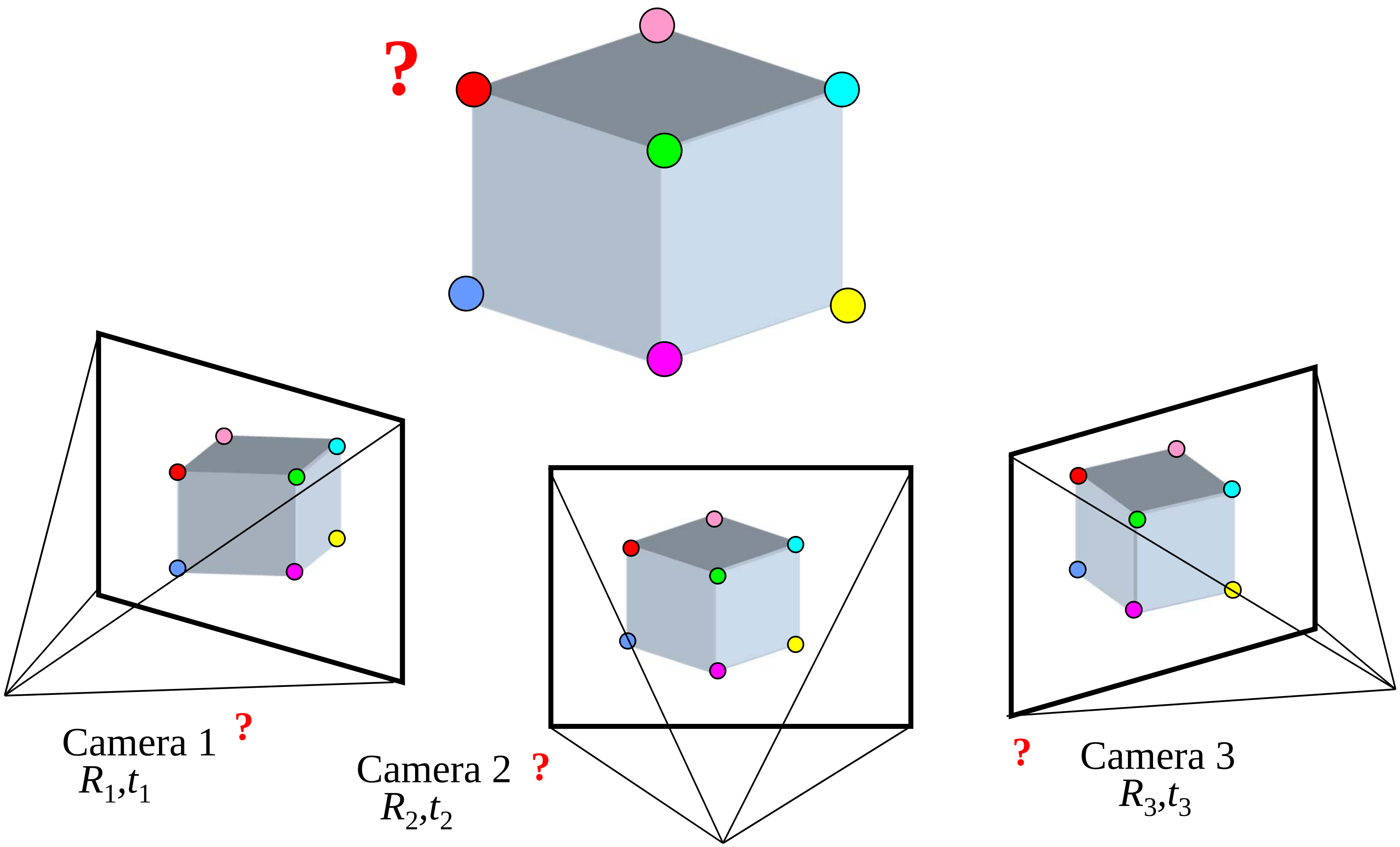
=> why 5 dof for 2 cameras and 6(k-1)-1 for k cameras?

Bundle Adjustment

- What makes this non-linear minimization hard?
  - many parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

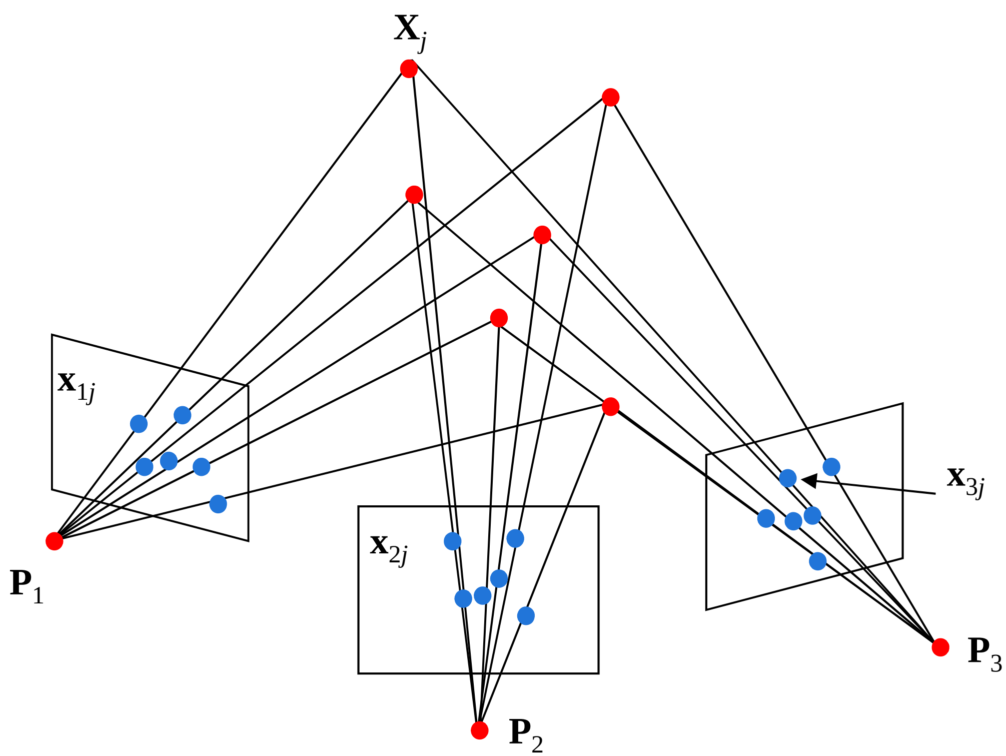
Structure from Motion

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



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:

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

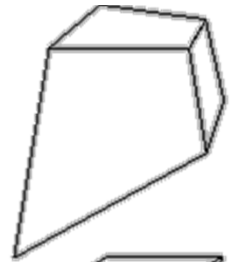
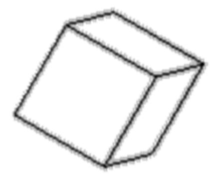
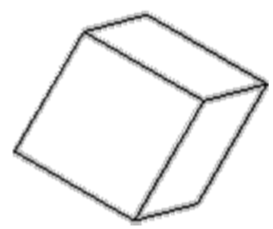
It is impossible to recover the absolute scale of the scene!

Structure from Motion Ambiguity

- 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

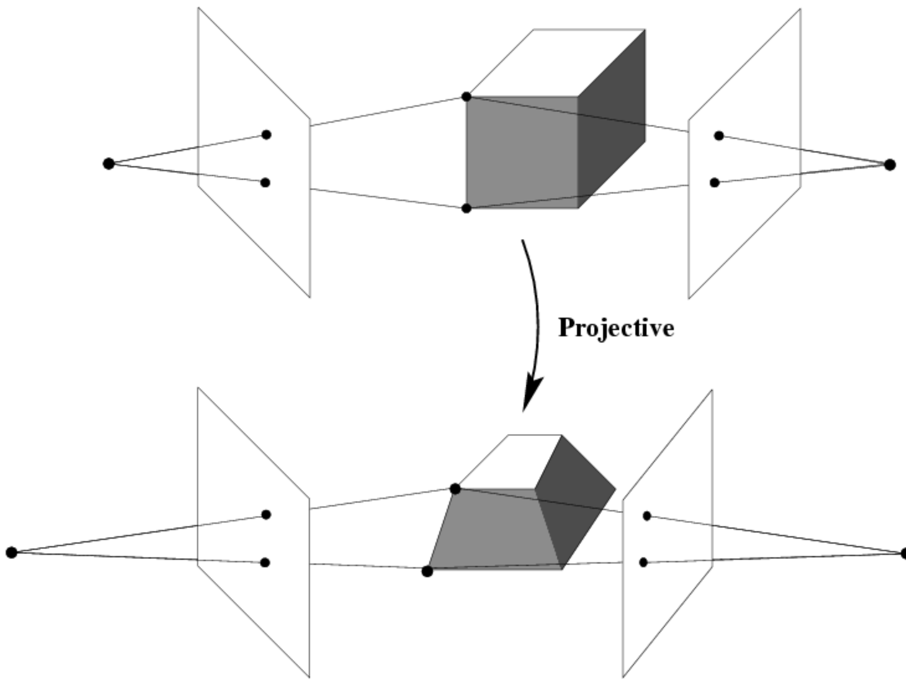
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)\left(\mathbf{Q}\mathbf{X}\right)$$

Types of Ambiguity

Projective 15dof	$\begin{bmatrix} \mathrm{A} & \mathrm{t} \\ \mathrm{v}^{\mathsf{T}} & v \end{bmatrix}$		Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} \mathrm{A} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$		Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} s\,\mathrm{R} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$		Preserves angles, ratios of length
Euclidean 6dof	$\begin{bmatrix} \mathrm{R} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$		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

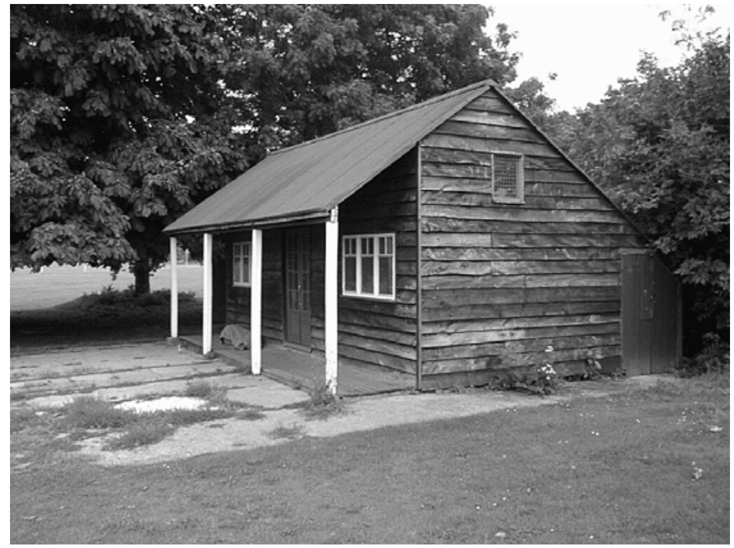
Projective Ambiguity

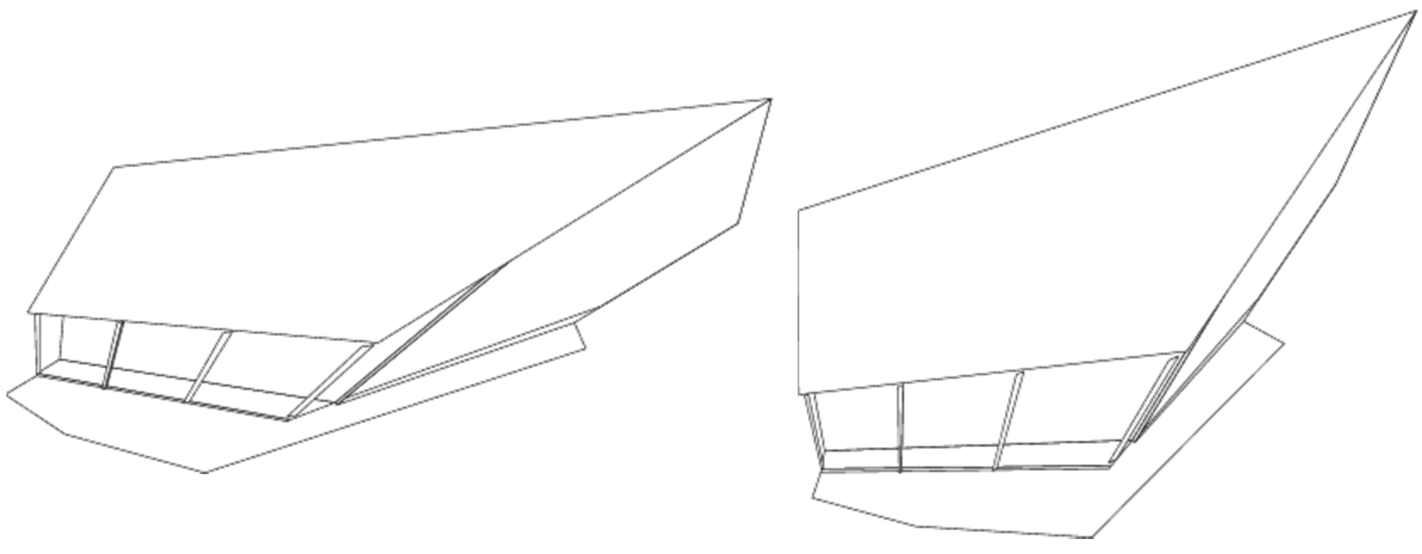


$$\mathbf{Q_p} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ \mathrm{v}^\mathsf{T} & v \end{bmatrix}$$

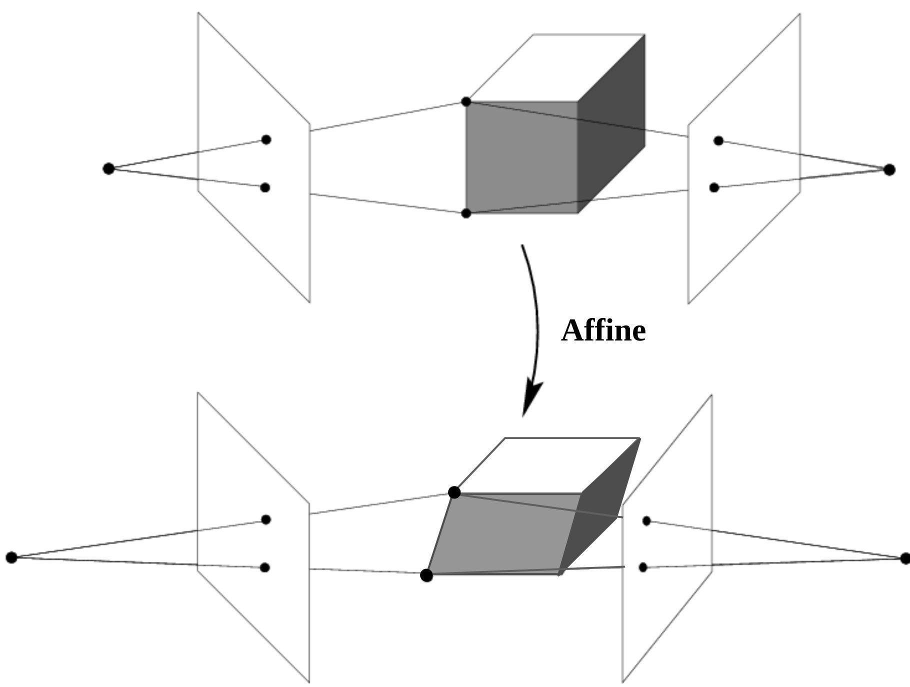
$$\mathbf{x} = \mathbf{PX} = \left(\mathbf{PQ_P^{-1}}\right)\left(\mathbf{Q_P\,X}\right)$$

Projective Ambiguity





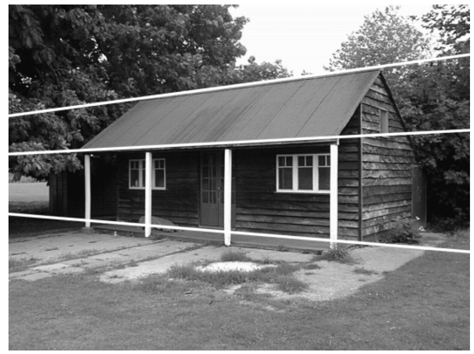
Affine Ambiguity

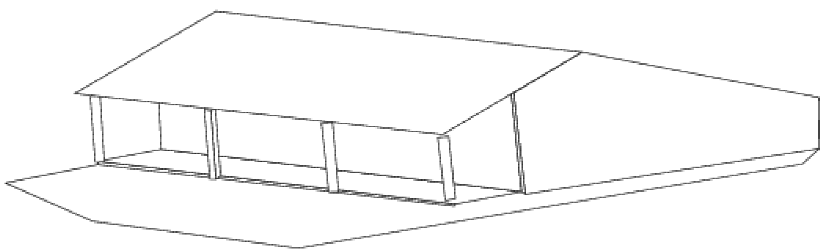


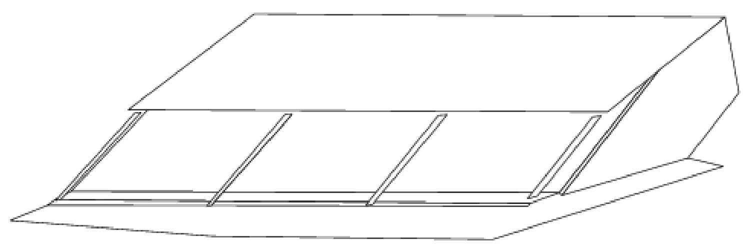
$$\mathbf{Q_A} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$$

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

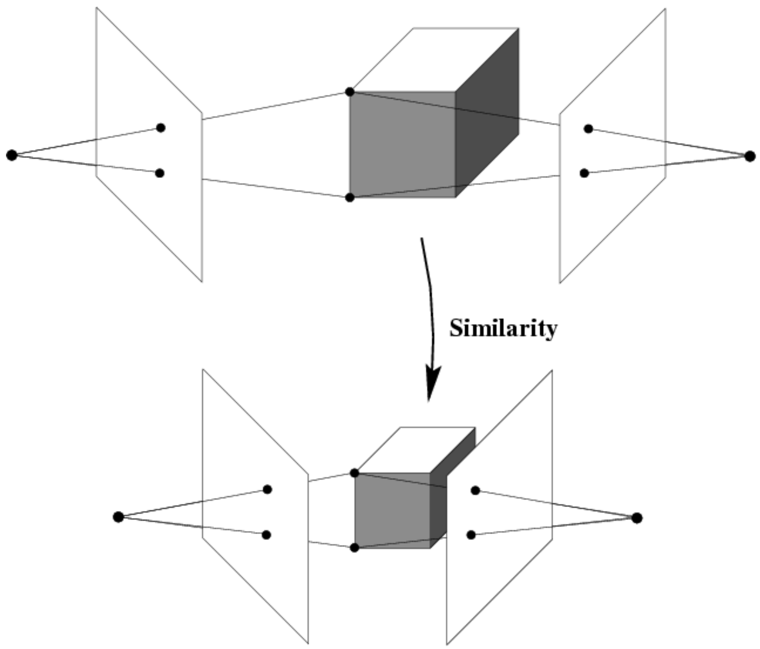
Affine Ambiguity







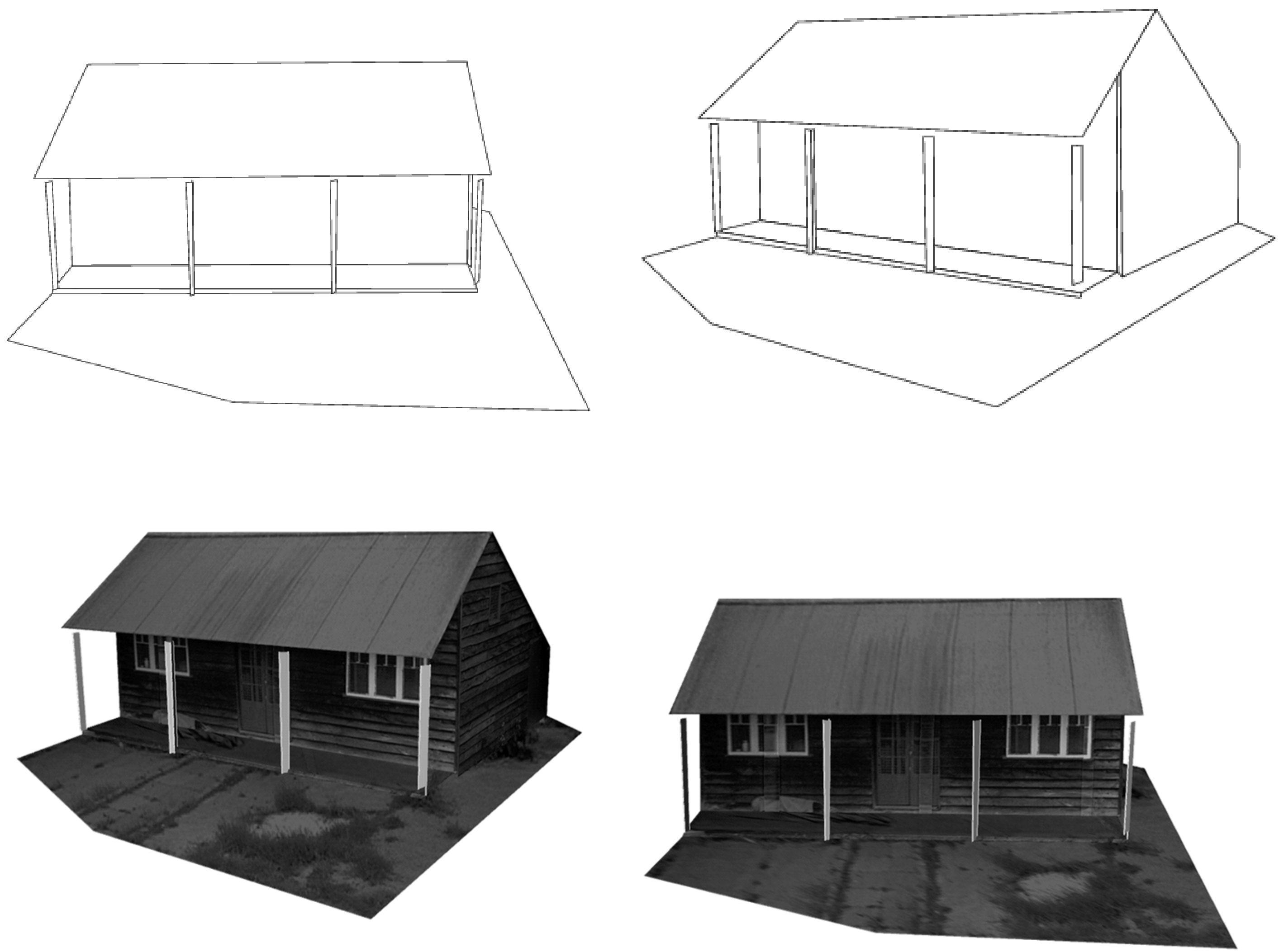
Similarity Ambiguity



$$\mathbf{Q}_s = \begin{bmatrix} s\mathrm{R} & \mathrm{t} \\ 0^\mathsf{T} & 1 \end{bmatrix}$$

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

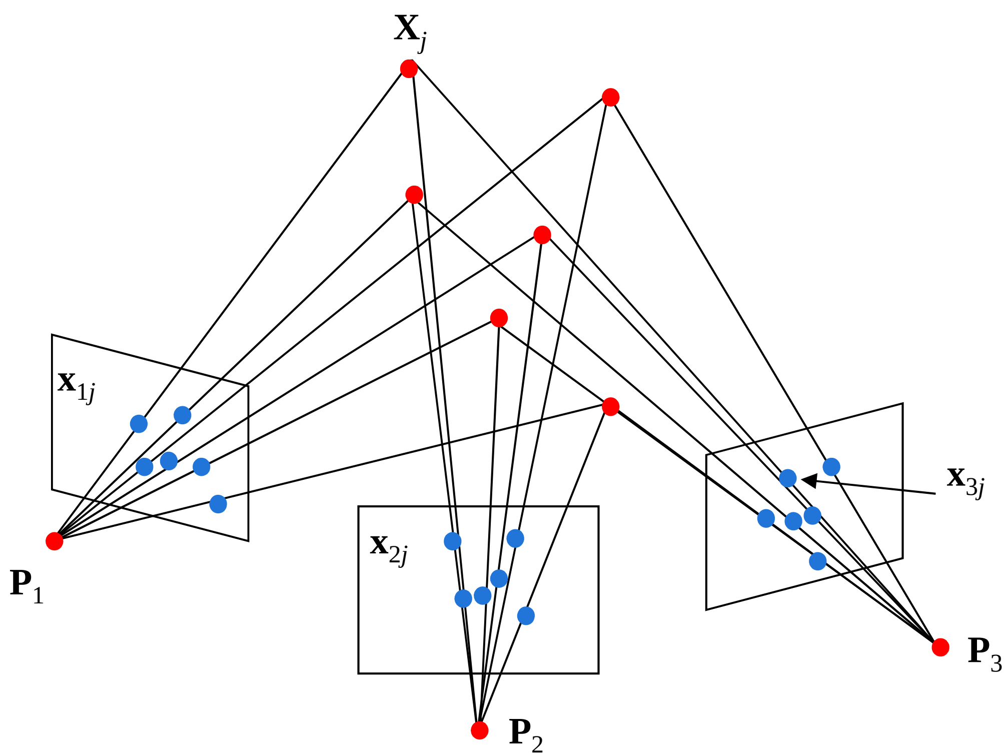
Similarity Ambiguity



Structure from Motion: Perspective Cameras

Projective Structure from Motion

- Given: *m* images of *n* fixed 3D points
  - $\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n$
- Problem: estimate *m* projection matrices $\mathbf{P}_i$ and *n* 3D points $\mathbf{X}_j$ from the *mn* correspondences $\mathbf{x}_{ij}$



Projective Structure from Motion

- Given: m images of n fixed 3D points
  - $z_{ij}\, \mathbf{x}_{ij} = \mathbf{P}_i\, \mathbf{X}_j\,, \qquad i = 1,\dots, m, \quad j = 1, \dots, n$
- Problem: estimate m projection matrices P_i and n 3D points 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:
  - $\mathrm{X} \rightarrow \mathrm{QX}, \mathrm{P} \rightarrow \mathrm{PQ}^{-1}$
- 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

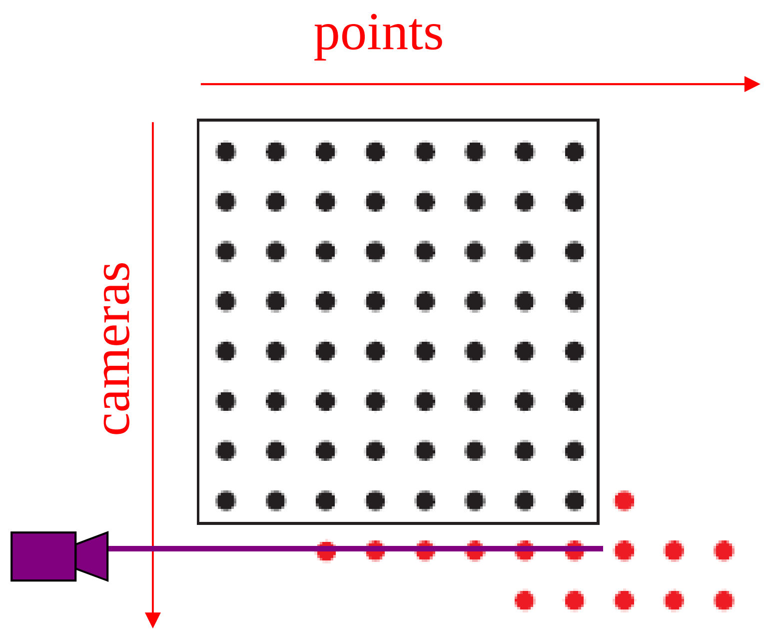
- Compute fundamental matrix $\mathbf{F}$ between the two views
- First camera matrix: $[\mathbf{I}|\mathbf{0}]$
- Second camera matrix: $[\mathbf{A}|\mathbf{b}]$
- Then $\mathbf{b}$ is the epipole ($\mathbf{F}^{\mathrm{T}}\mathbf{b} = 0$), $\mathbf{A} = -[\mathbf{b}_{\times}]\mathbf{F}$

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

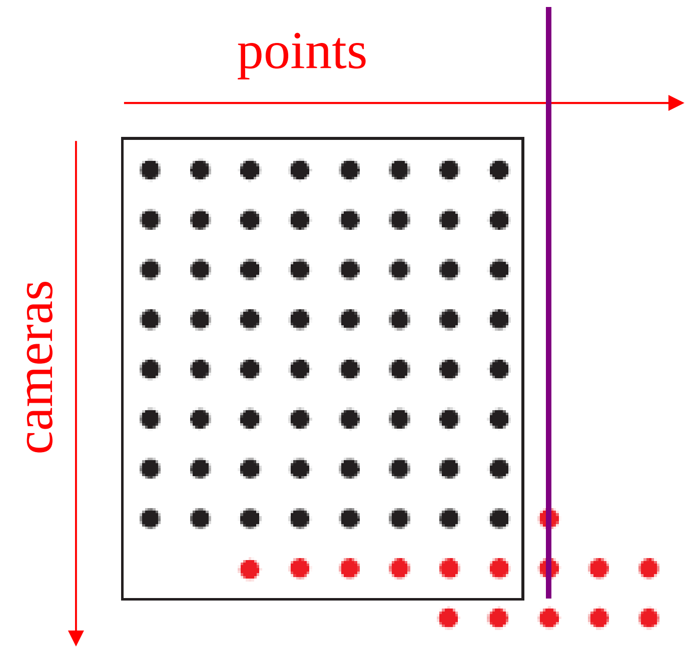
•Initialize motion from two images using fundamental matrix

•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*

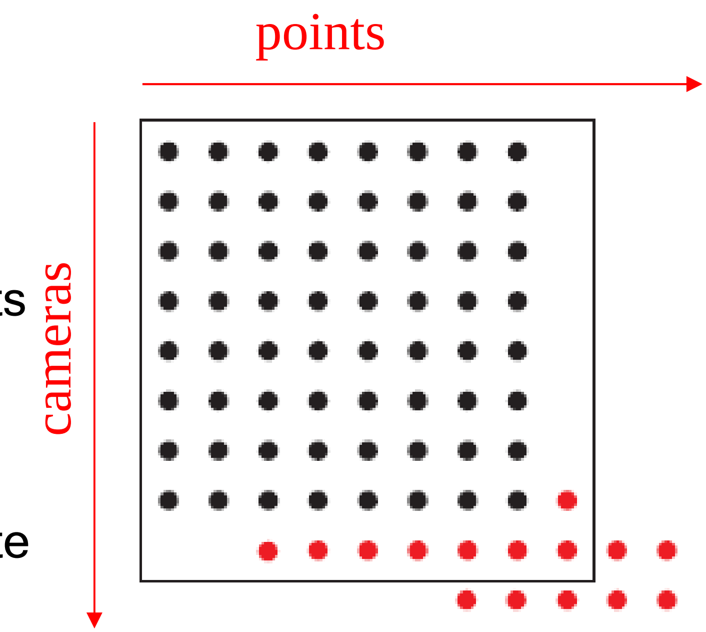


Sequential Structure from motion

- 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*

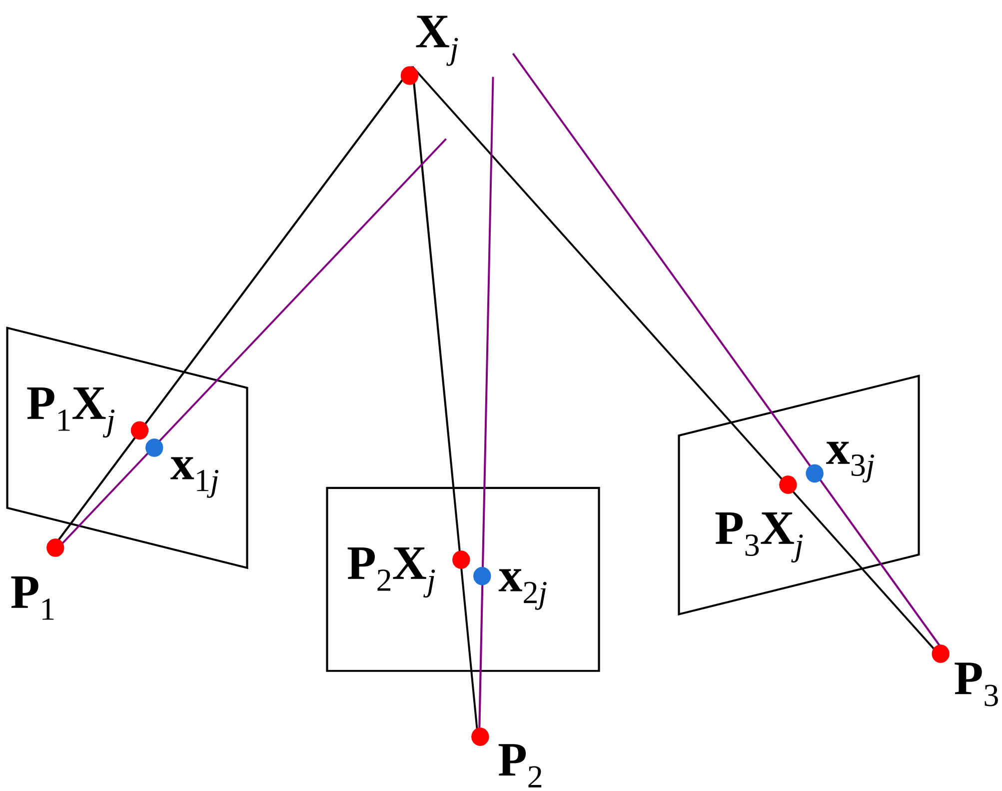


- Refine structure and motion: bundle adjustment

Bundle Adjustment

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

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

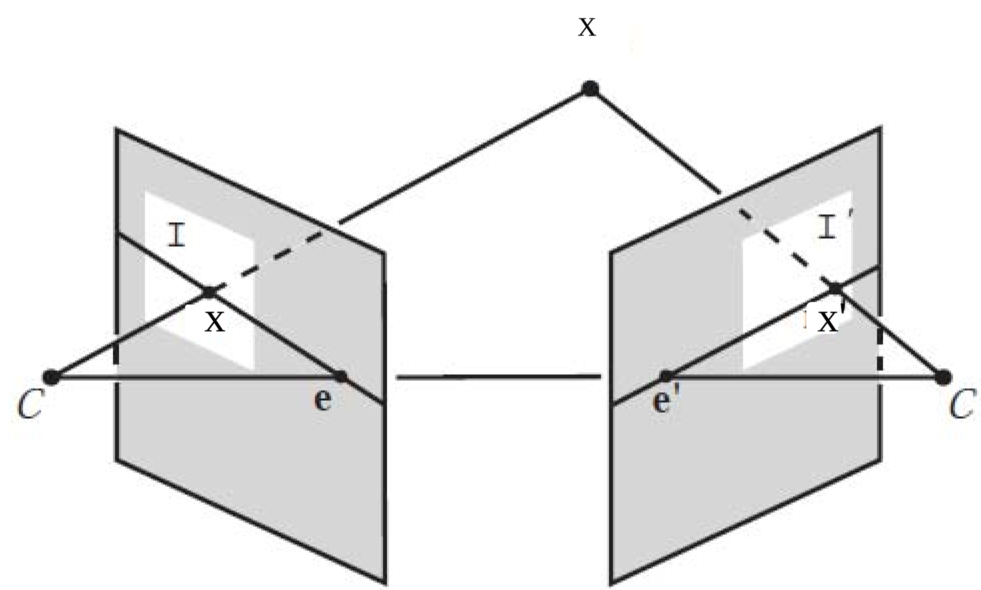


Self-calibration

- 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 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

Triangulation: Linear Solution

- Generally, rays C→x and C'→x' will not exactly intersect
- Can solve via SVD, finding a least squares solution to a system of equations



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

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

From x×PX = 0 and x'×PX' = 0

Triangulation: Linear Solution

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

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

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

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

Pros and Cons

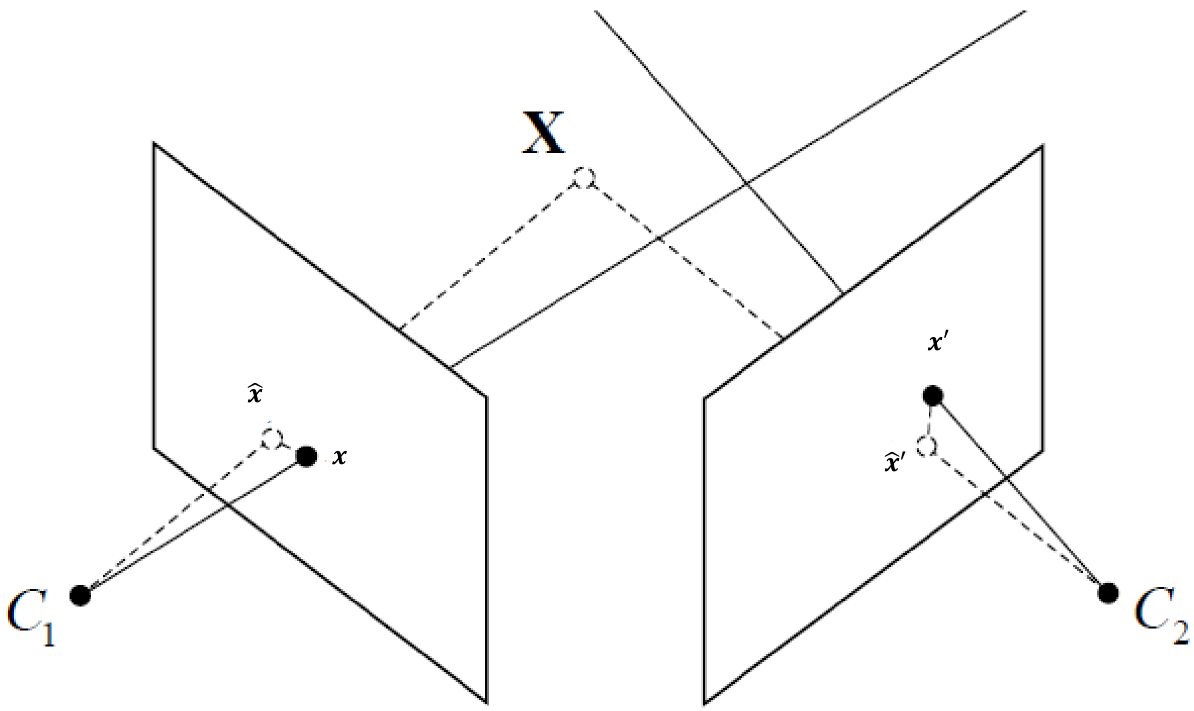
- Works for any number of corresponding images
- Not projectively invariant

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

Triangulation: Non-linear Solution

- Minimize projected error while satisfying $\hat{\boldsymbol{x}}'^{T}\boldsymbol{F}\hat{\boldsymbol{x}}=0$

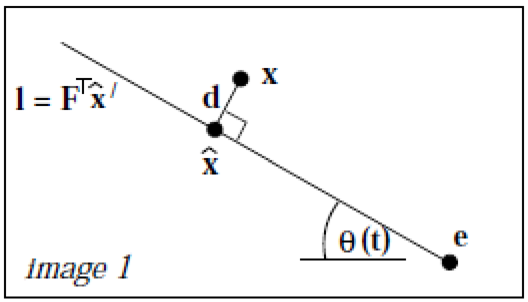
$$cost(\boldsymbol{X}) = dist(\boldsymbol{x}, \hat{\boldsymbol{x}})^2 + dist(\boldsymbol{x}', \hat{\boldsymbol{x}}')^2$$

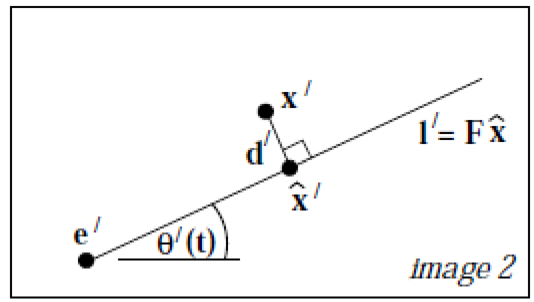


Triangulation: Non-linear Solution

- Minimize projected error while satisfying $\hat{\boldsymbol{x}}'^T \boldsymbol{F} \hat{\boldsymbol{x}}=0$

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





- Solution is a 6-degree polynomial of *t*, minimizing $d(\mathbf{x}, \mathbf{l}(t))^2 + d(\mathbf{x}', \mathbf{l}'(t))^2$

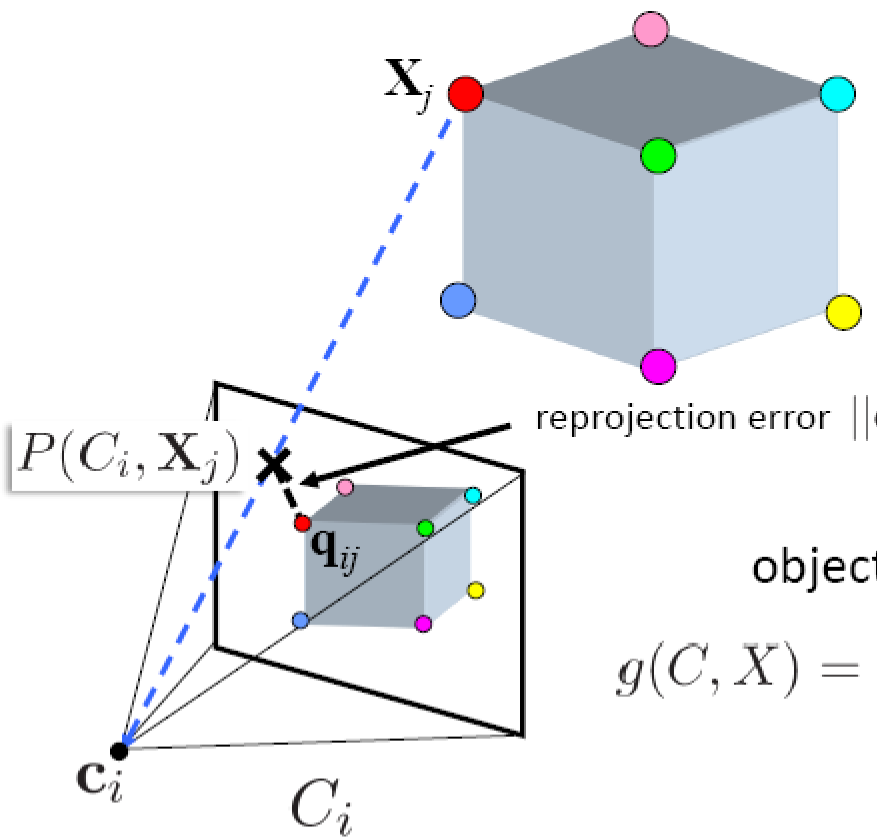
Bundle Adjustment

Bundle Adjustment

- Refines a visual reconstruction to produce jointly optimal 3D structure and viewing parameters

- *'Bundle'* refers to the bundle of light rays leaving each 3D feature and converging on each camera center.

Reprojection Error



reprojection error $||\mathbf{q}_{ij} - P(C_i, \mathbf{X}_j)||$

objective function:

$$g(C, X) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} ||\mathbf{q}_{ij} - P(C_i, \mathbf{X}_j)||^2$$

indicator variable:
1 if point *j* is visible in camera *i*
0 otherwise

Notation

- Structure and Cameras being parameterized by a single large vector **x**
- Small displacement in **x** represented by $\partial \boldsymbol{x}$
- Observations denoted by $\underline{z}$
- Predicted values at parameter value **x**, denoted by z = z(x)
- Residual prediction error, $\Delta z(\boldsymbol{x}) = \underline{z} - z(\boldsymbol{x})$

Objective Function

- Minimization of weighted sum of squared error (SSE) cost function:

$$\mathsf{f}(\mathsf{x}) \equiv \frac{1}{2} \sum_i \triangle \mathsf{z}_i(\mathsf{x})^\top \mathsf{W}_i \, \triangle \mathsf{z}_i(\mathsf{x}), \qquad \triangle \mathsf{z}_i(\mathsf{x}) \equiv \underline{\mathsf{z}}_i - \mathsf{z}_i(\mathsf{x})$$

Optimization Techniques

- Gradient Descent Method
- Newton-Raphson Method
- Gauss – Newton Method
- Levenberg – Marquardt Method

Additional Material and Software

- Open Source Structure-from-Motion tutorial at CVPR 2015
  - http://www.kitware.com/cvpr2015-tutorial.html
- Advanced notes on bundle adjustment
- Tutorials on several popular open source SfM packages

Slide Credits

- This set of sides contains contributions kindly made available by the following authors
  - Michael Bleyer
  - Pascal Fua
  - Svetlana Lazebnik
  - Steve Seitz
  - Noah Snavely
  - Richard Szeliski