Computer Vision I - *Multi-View 3D reconstruction*

Carsten Rother

06/01/2017

Roadmap for next five lectures

- Appearance-based Matching (sec. 4.1)
- Projective Geometry - Basics (sec. 2.1.1-2.1.4)
- Geometry of a Single Camera (sec 2.1.5, 2.1.6)
  - Camera versus Human Perception
  - The Pinhole Camera
  - Lens effects
- Geometry of two Views (sec. 7.2)
  - The Homography (e.g. rotating camera)
  - Camera Calibration (3D to 2D Mapping)
  - The Fundamental and Essential Matrix (two arbitrary images)
- Robust Geometry estimation for two views (sec. 6.1.4)
- Accurate Geometry estimation for two views
- Multi-View 3D reconstruction (sec. 7.3-7.4)

RANSAC

[Random Sample Consensus, Fischler and Bolles 1981]

Basic RANSAC method:

Can be done in parallel!

Repeat many times
  select d-tuple, e.g. (x^1, x^2) for lines
  compute parameter(s) y, e.g. line $y = g(x^1, x^2)$
  evaluate $f'(y) = \sum_i f(x^i, y)$
  If $f'(y) \leq f'(y^*)$
    set $y^* = y$ and keep value $f'(y^*)$

- Sometimes we get a discrete set of intermediate solutions y. For example for F-matrix computation from 7 points we have up to 3 solutions. The we simply evaluate $f'(y)$ for all solutions.

- How many times do you have to sample in order to reliable estimate the true model?

Randomized RANSAC

Evaluation of a hypothesis y , i.e. $\sum_i f(x^i, y)$ often time consuming

<u>Randomized</u> <u>RANSAC:</u>

instead of checking all data points $x^i \in L$

1. Sample m points from L

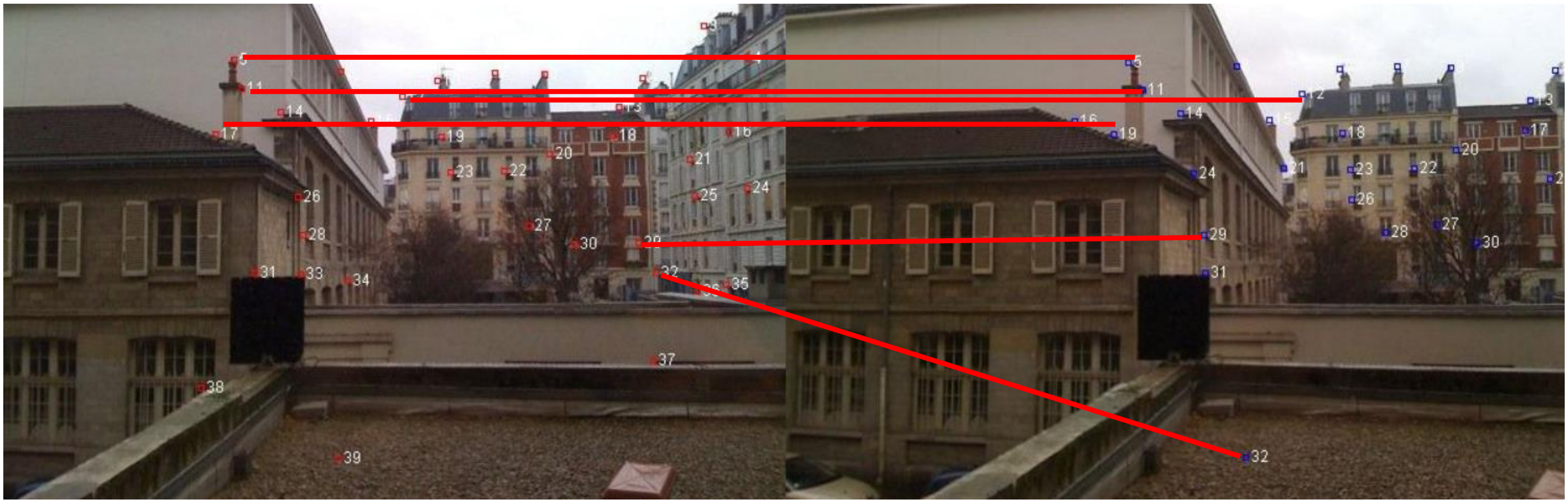
2. If all of them are inliers, check all others as before, i.e. evaluate hypothesis. But, if there is at least one bad point, among m, reject the hypothesis

It is possible that good hypotheses are rejected.
However it saves time (bad hypotheses are recognized fast)
→ one can sample more often
→ overall often profitable (depends on application).

Roadmap for next five lectures

- Appearance-based Matching (sec. 4.1)
- Projective Geometry - Basics (sec. 2.1.1-2.1.4)
- Geometry of a Single Camera (sec 2.1.5, 2.1.6)
  - Camera versus Human Perception
  - The Pinhole Camera
  - Lens effects
- Geometry of two Views (sec. 7.2)
  - The Homography (e.g. rotating camera)
  - Camera Calibration (3D to 2D Mapping)
  - The Fundamental and Essential Matrix (two arbitrary images)
- Robust Geometry estimation for two views (sec. 6.1.4)
- Accurate Geometry estimation for two views
- Multi-View 3D reconstruction (sec. 7.3-7.4)

In last lecture we asked (for rotating camera)…



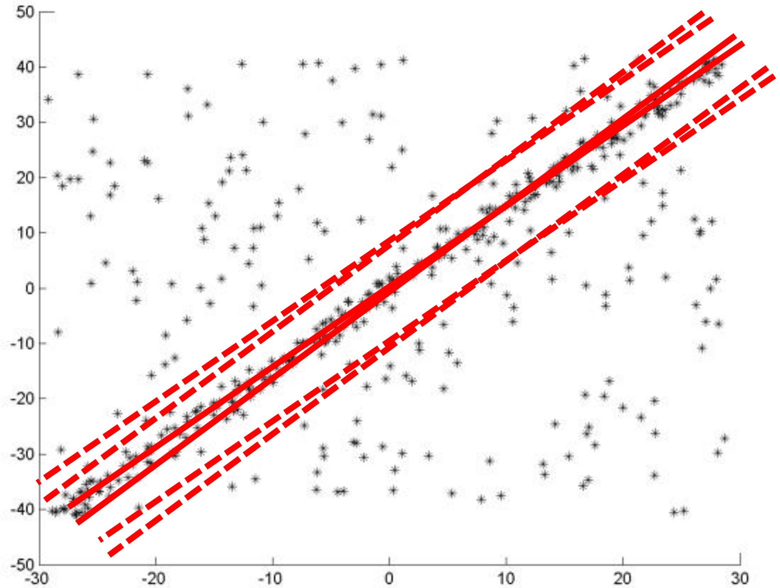
Question 1: If a match is completly wrong then $argmin_h \|A\boldsymbol{h}\|$ is a bad idea
Answer: RANSAC with $d = 4$

Question 2: If a match is slighly wrong then $argmin_h \|A\boldsymbol{h}\|$ might not be perfect.
Better might be a geometric error: $argmin_h \|\boldsymbol{H}x - x'\|$
Answer: see next slides

Refinement after RANSAC

<u>Typical procedure:</u>

1. RASNAC: compute model y in a robust way
2. Find all inliers $x_{inliers}$
3. Refine model y from inliers $x_{inliers}$
4. Go to Step 2.
   (until numbers of inliers or model does not change much)



Method to compute F, E, H for 2 Views

Procedure (as mentioned above):

1. RASNAC: compute model F, E, H in a robust way
2. Find all inliers $x_{inliers}$ (with potential relaxed criteria)
3. Refine model $F/E/H$ from inliers $x_{inliers}$
4. Go to Step 2.
   (until numbers of inliers or model does not change much)

Next questions:

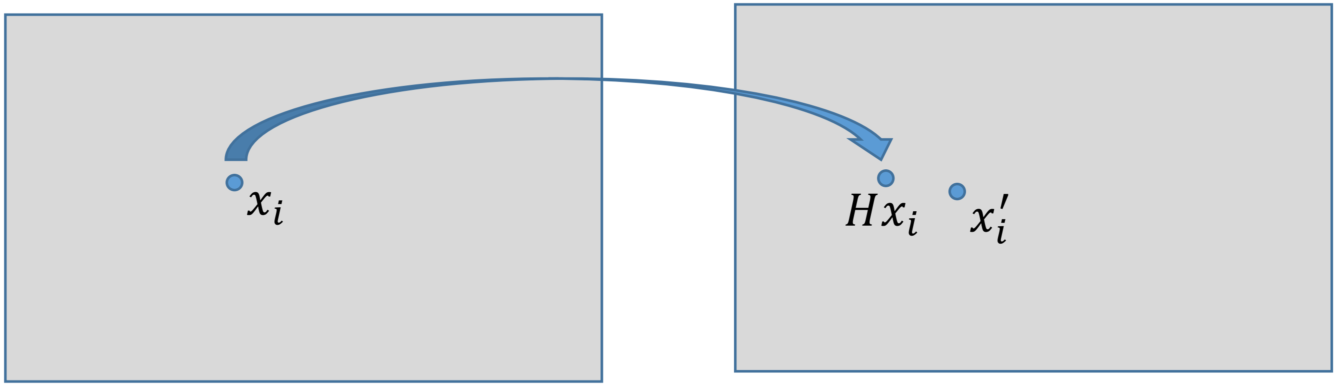
1) What is the best error measure for model computation in step 1 and 3?
2) How to do step 3?

Error function

1. Algebraic error: $argmin_h \|A\boldsymbol{h}\|$

where $d(a, b)$ is 2D geometric distance $\|a - b\|^2$

2. *First* geometric error: $H^* = argmin_H \sum_i d(x_i', Hx_i)$



This is not symmetric

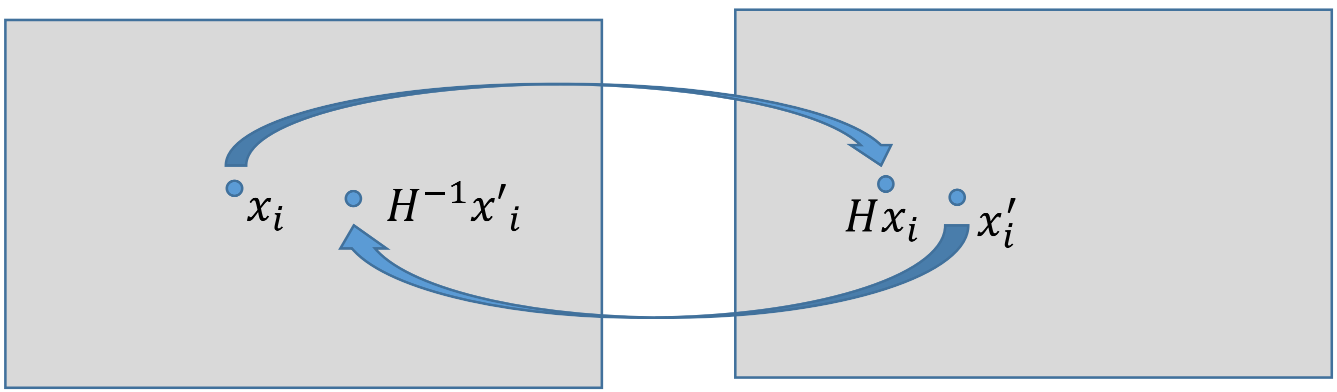
Error function

1. Algebraic error: $argmin_h \|A\mathbf{h}\|$

where $d(a, b)$ is 2D geometric distance $\|a - b\|^2$

2. *First* geometric error: $H^* = argmin_H \sum_i d(x_i', Hx_i)$

3. *Second,* symmetric geometric error: $H^* = argmin_H \sum_i d(x_i', Hx_i) + d(x_i, H^{-1}x'_i)$



Error function

1. Algebraic error: $argmin_h \|A\mathbf{h}\|$

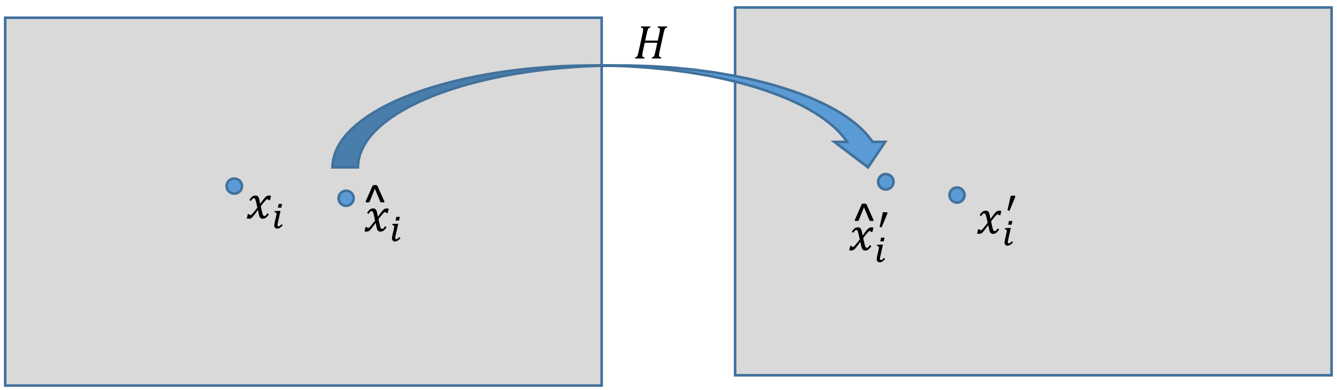
where $d(a,b)$ is 2D geometric distance $\|a-b\|^2$

2. *First* geometric error: $H^* = argmin_H \sum_i d(x_i', Hx_i)$

3. *Second,* symmetric geometric error: $H^* = argmin_H \sum_i d(x_i', Hx_i) + d(x_i, H^{-1}x'_i)$

4. *Third, optimal* geometric error (gold standard error):

$$\{H^*, \hat{x}_i, \hat{x}'_i\} = \underset{H, \hat{x}_i, \hat{x}'_i}{argmin} \sum_i d(x_i, \hat{x}_i) + d(x'_i, \hat{x}'_i) \qquad subject\ to\ \hat{x}_i' = H\hat{x}_i$$



the true 3D points $\hat{X}$ are searched for

Comment: This is optimal in the sense that it is the maximum-likelihood (ML) estimation under isotropic Gaussian noise assumption for $\hat{x}$ (see page 103 HZ)

Method to compute H for 2 Views - Details

Objective

Compute the 2D homography between two images.

Algorithm

(i) **Interest points:** Compute interest points in each image.
(ii) **Putative correspondences:** Compute a set of interest point matches based on proximity and similarity of their intensity neighbourhood.
(iii) **RANSAC robust estimation:** Repeat for N samples, where N is determined adaptively as in algorithm 4.5:
  (a) Select a random sample of 4 correspondences and compute the homography $\mathtt{H}$.
  (b) Calculate the distance $d_\perp$ for each putative correspondence.
  (c) Compute the number of inliers consistent with $\mathtt{H}$ by the number of correspondences for which $d_\perp < t = \sqrt{5.99}\,\sigma$ pixels.

  Choose the $\mathtt{H}$ with the largest number of inliers. In the case of ties choose the solution that has the lowest standard deviation of inliers.
(iv) **Optimal estimation:** re-estimate $\mathtt{H}$ from all correspondences classified as inliers, by minimizing the ML cost function (4.8–*p*95) using the Levenberg–Marquardt algorithm of section A6.2(*p*600).
(v) **Guided matching:** Further interest point correspondences are now determined using the estimated $\mathtt{H}$ to define a search region about the transferred point position.

The last two steps can be iterated until the number of correspondences is stable.

Algorithm 4.6. *Automatic estimation of a homography between two images using RANSAC.*

we discussed : Harries corner detector

we mentioned : Kd-tree to make it fast

This is a geometric error (for fixed H, see next slides). Depending on runtime one can choose different once.

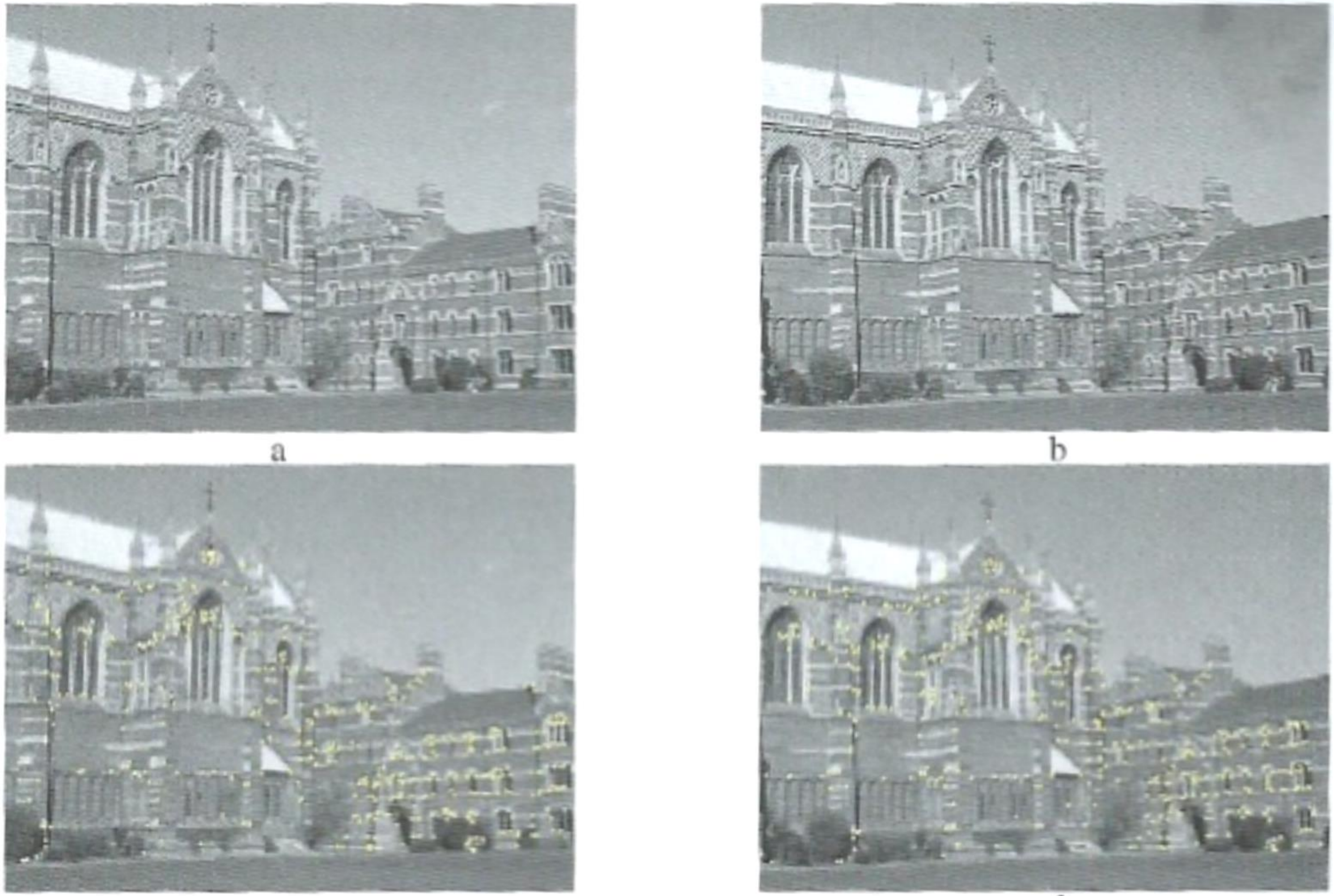
See next slide

This is the optimal geometric error

Refinement, see next slides

[see details on page 114ff in HZ]

Example



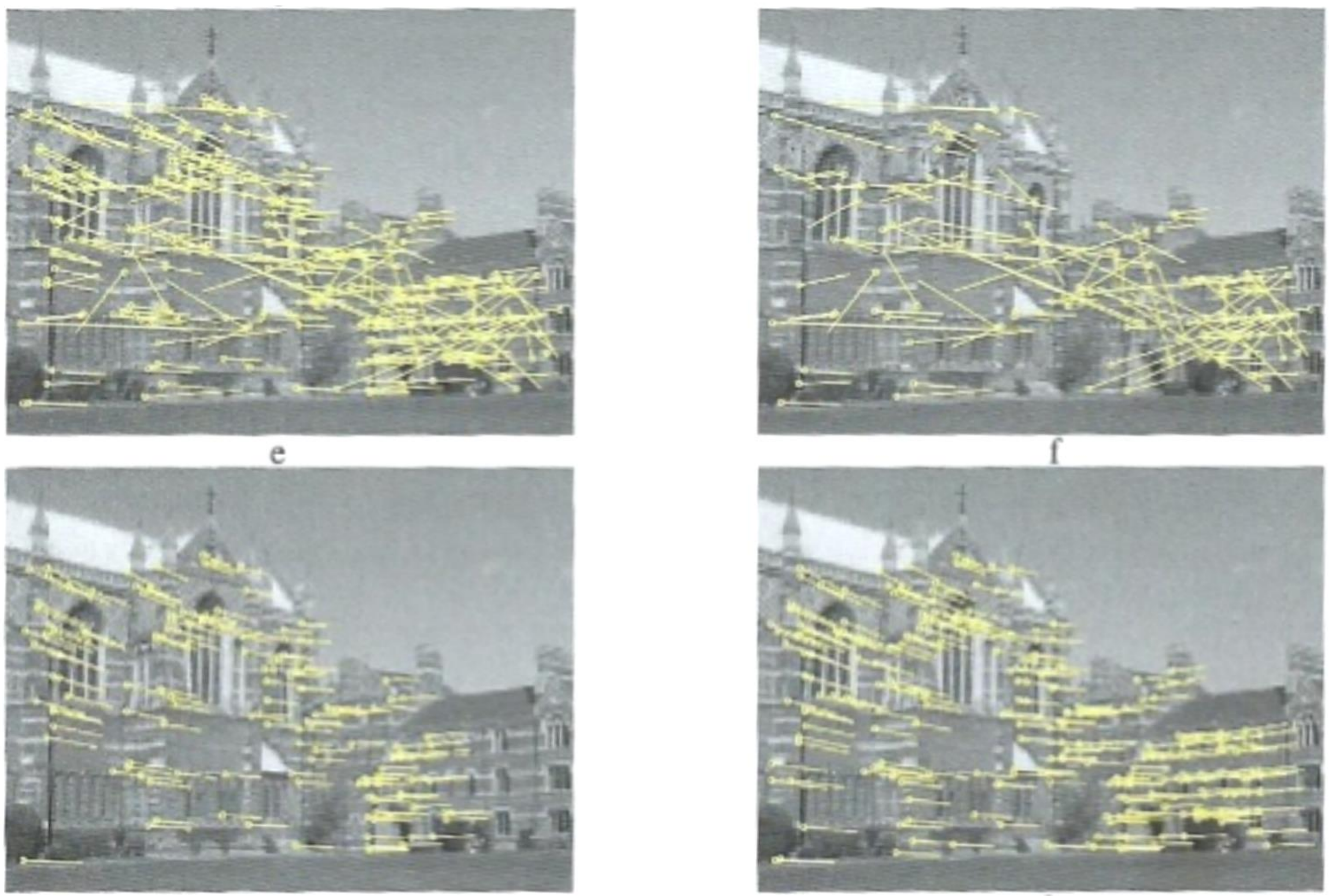
Input images

~500 interest points

Example

268
putative
matches

117
outliers
found



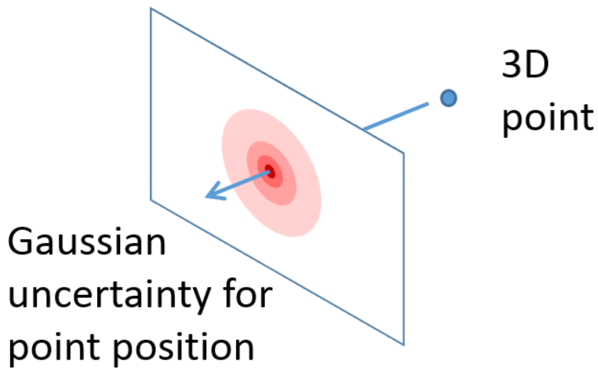
151
inliers
found

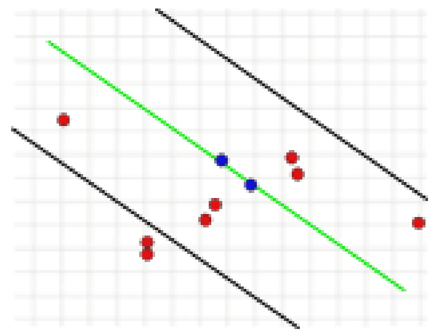
262
inliers
after
guided
matching

<u>Guided matching variant:</u> use given H and look for new inliers. Here we also double the threshold on appearance feature matches to get more inliers.

Geometric derivation of confidence interval

Assume Gaussian noise for a point with σ standard deviation and 0 mean:





To have a 95% chance that an inlier is inside the confidence interval, we require:

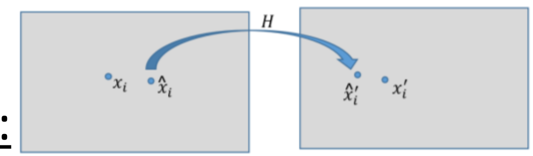
1. For a 2D line: $d(x, l) \leq \sigma\sqrt{3.84} = t$
2. For a Homography: $d(x_l, x_r, H) \leq \sigma\sqrt{5.99} = t$
3. For an F-matrix: $d(x_l, x_r, F) \leq \sigma\sqrt{3.84} = t$

(see page 119 HZ)

Method to compute H, E, F for 2 Views - Details

Procedure (as mentioned above):

1. RASNAC: compute model $F/E/H$ in a robust way
2. Find all inliers $x_{inliers}$ (with potential relaxed criteria)
3. Refine model $F/E/H$ from inliers $x_{inliers}$
4. Go to Step 2.
   (until numbers of inliers or model does not change much)



We need geometric error for a ***fixed*** model $F/E/H$ (RANSAC):

1. For a Homography: $d(x, x', H) = \min_{\hat{x}, \hat{x}'} [d(x, \hat{x}) + d(x', \hat{x}')]$ subject to $\hat{x}' = H\hat{x}$
2. For an F/E-matrix: $d(x, x', F/E) = \min_{\hat{x}, \hat{x}'} [d(x, \hat{x}) + d(x', \hat{x}')]$ subject to $\hat{x}'^t F/E \hat{x} = 0$

We need geometric error for ***model refinement*** $F/E/H$:

1. For a Homography: $\{H^*, \hat{x}_i, \hat{x}_i'\} = \underset{H, \hat{x}_i, \hat{x}'_i}{argmin} \sum_i d(x_i, \hat{x}_i) + d(x'_i, \hat{x}'_i)$ subject to $\hat{x}'_i = H\hat{x}_i$
2. For an F/E-matrix: $\{F^*/E^*, \hat{x}_i, \hat{x}_i'\} = \underset{F/E, \hat{x}_i, \hat{x}'_i}{argmin} \sum_i d(x_i, \hat{x}_i) + d(x'_i, \hat{x}'_i)$ sbj. to $\hat{x}'^t_i F/E \hat{x}_i = 0$

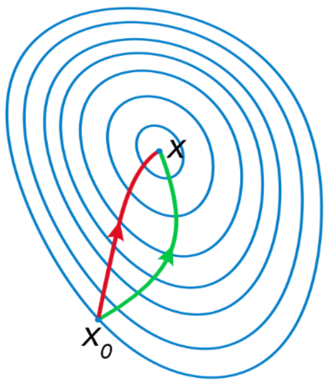
A few word on iterative continuous optimization

So far we had linear (least square) optimization problems:

$$x^* = argmin_x \|Ax\|_2$$

For non-linear (arbitrary) optimization problems:

$$x^* = argmin_x f(x)$$

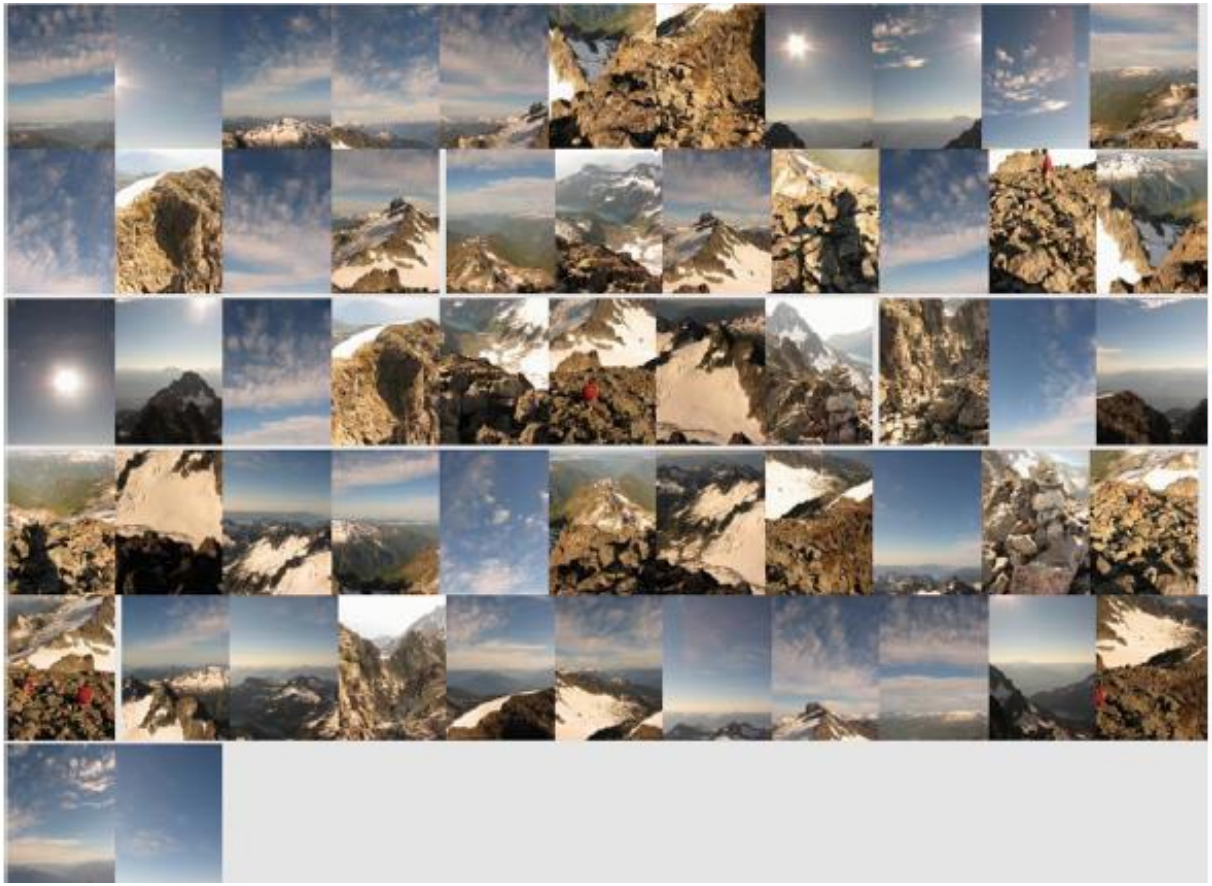


Red Newton's method; green gradient descent

- <u>Iterative Estimation methods</u> (see Appendix 6 in HZ; page 597ff)
  - Gradient Descent Method
    (good to get roughly to solution)
  - Newton Methods (e.g. Gauss-Newton):
    second order Method (Hessian). Good to find accurate result
  - Levenberg – Marquardt Method:
    mix of Newton method and Gradient descent

Application: Automatic Panoramic Stitching

An unordered set of images:



Run Homography search between all pairs of images

Application: Automatic Panoramic Stitching

... automatically create a panorama

Application: Automatic Panoramic Stitching

... automatically create a panorama

Application: Automatic Panoramic Stitching

... automatically create a panorama

Roadmap for next five lectures

- Appearance-based Matching (sec. 4.1)

- Projective Geometry - Basics (sec. 2.1.1-2.1.4)

- Geometry of a Single Camera (sec 2.1.5, 2.1.6)
  - Camera versus Human Perception
  - The Pinhole Camera
  - Lens effects

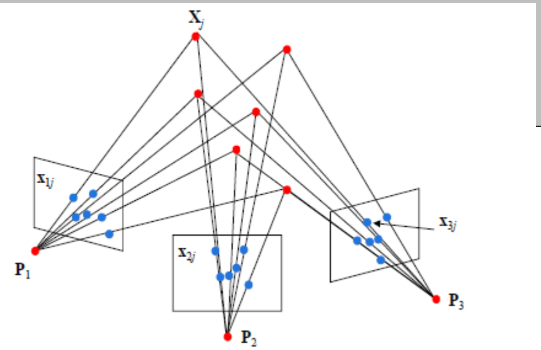
- Geometry of two Views (sec. 7.2)
  - The Homography (e.g. rotating camera)
  - Camera Calibration (3D to 2D Mapping)
  - The Fundamental and Essential Matrix (two arbitrary images)

- Robust Geometry estimation for two views (sec. 6.1.4)

- Accurate Geometry estimation for two views

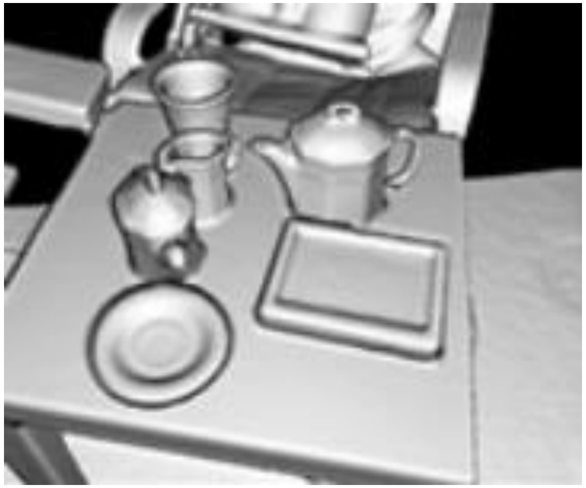
- Multi-View 3D reconstruction (sec. 7.3-7.4)

Names: 3D reconstruction



1) Sparse Structure from Motion (SfM)
   In Robotics this is known as SLAM (Simultaneous Localization and Mapping): "Place a robot in an unknown location in an unknown environment and have the robot incrementally build a map of this environment while simultaneously using the map to compute the vehicle location"

2) Dense Multi-view reconstruction



Example: Sparse Reconstruction

Building Rome in a day from People's Photos

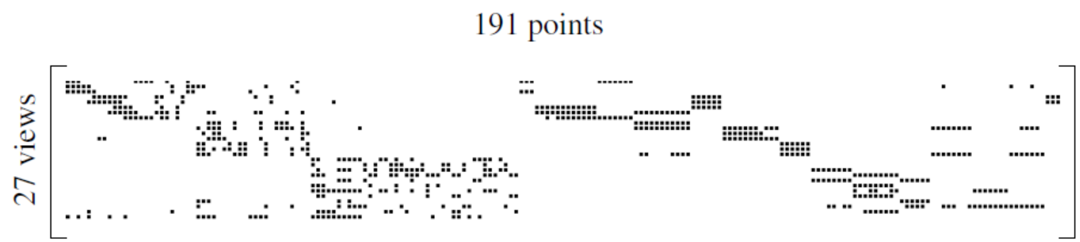
[Agarwal, Snavely, Simon, Seitz, Szeliski; ICCV 2009]

Example: Dense Reconstruction

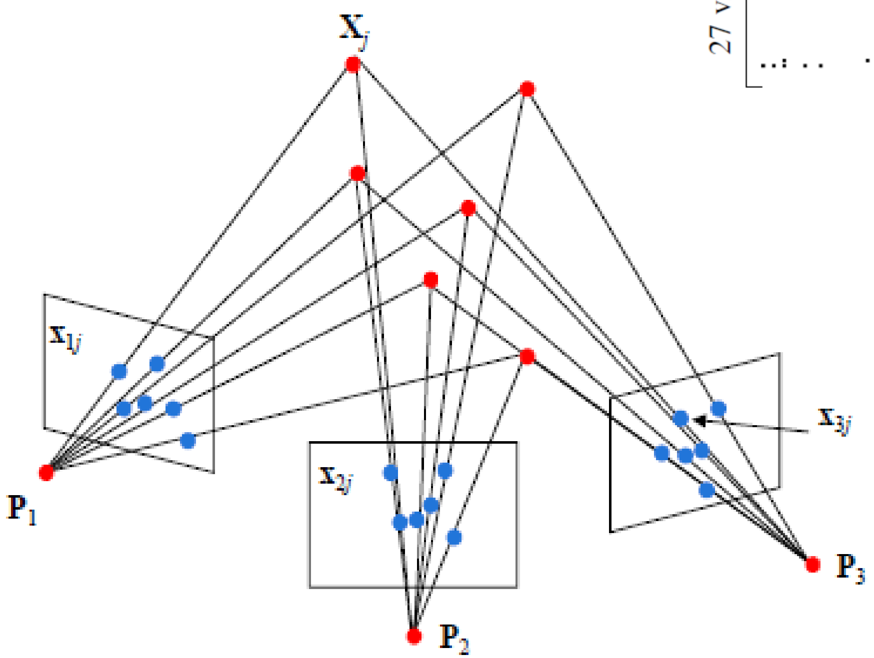
[KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera, Izadi et al ACM Symposium on User Interface Software and Technology, October 2011]

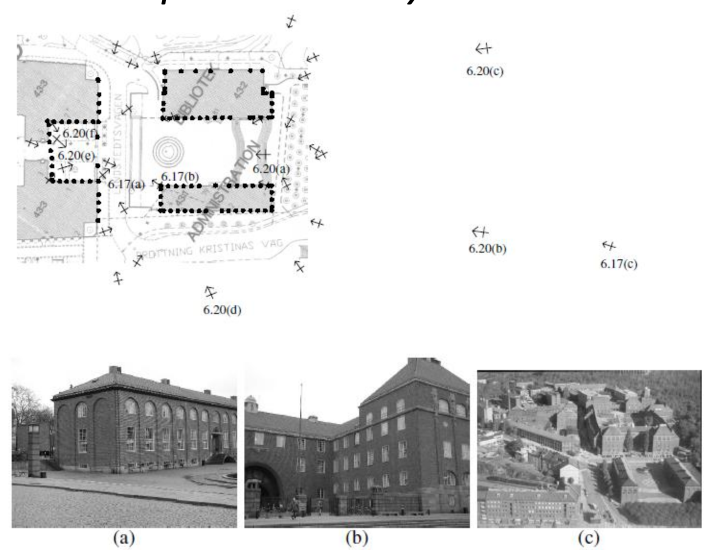
3D Reconstruction: Problem definition

- Given image observations in m cameras of n static 3D points
- Formally: $x_{ij} = P_j X_i$ for $j = 1 \ldots m; i = 1 \ldots n$
- Important: In practice we do not have all points visible in all views, i.e. the number of $x_{ij} \leq mn$ (this is captured by the “visibilty matrix”)
- Goal: find all P_j’s and X_i’s



Example: “Visibility” matrix

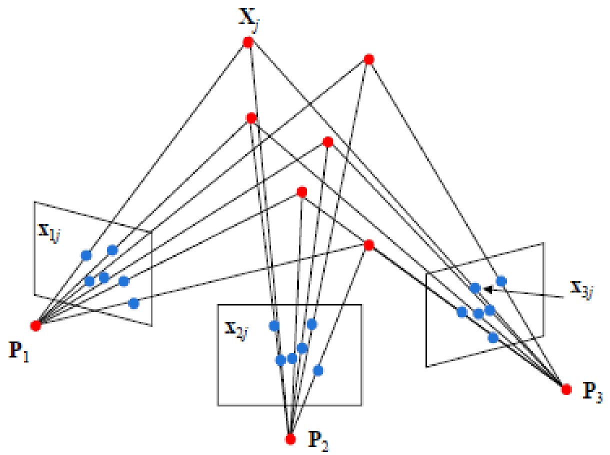




Reconstruction Algorithm

Procedure: (calibrated and un-calibrated cameras)

1) Compute accurate F, E-matrix between each pair of neighboring views
2) Uncalibrated case: derive intrinsic camera parameters for each pair
3) Compute initial reconstruction of each pair of neighboring views
4) Compute an initial full 3D reconstruction
5) Bundle-Adjustment to minimize overall geometric error



Reconstruct in step 2): (P_1, P_2); (P_2, P_3); (P_3, P_4) …

[See page 453 HZ]

Reconstruction Algorithm – Historic View

Procedure:
(calibrated and un-calibrated cameras)

1) Compute accurate F, E-matrix between each pair of neighboring views
2) Uncalibrated case: derive intrinsic camera parameters for each pair
3) Compute initial reconstruction of each pair of neighboring views
4) Compute an initial full 3D reconstruction
5) Bundle-Adjustment to minimize overall geometric error

"Modern"
(since it works as well as historic procedure)

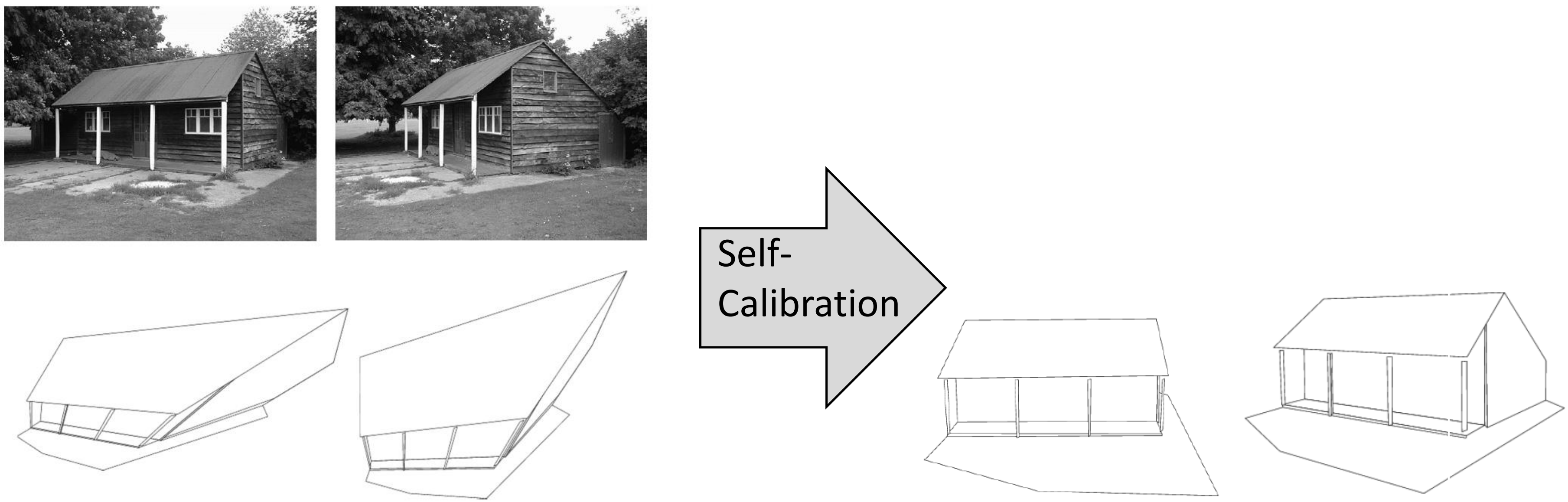
Procedure:
(calibrated and un-calibrated cameras)

1) Compute accurate F-matrix between each pair of neighboring views
2) -
3) Compute initial reconstruction of each pair/triplets of neighboring views (more complex)
4) Compute an initial full 3D reconstruction
5) Bundle-Adjustment to minimize overall geometric error
6) Uncalibrated case: Self-calibration. Determine a 4×4 Matrix to bring the reconstruction form projective to Euclidian space

"Historic"
(10+ years research on uncalibrated cameras)

Reconstruction Algorithm – Historic View

Uncalibrated case: Self-calibration. Determine a 4×4 Matrix to bring the reconstruction form projective to Euclidian space



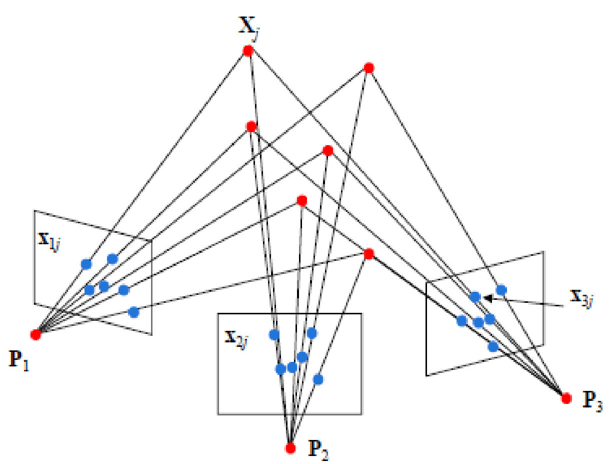
Correct reconstruction
(up to 3D projective ambiguity)

Correct reconstruction
(up to 3D Eucledian ambiguity)

Reconstruction Algorithm

Procedure: (calibrated and un-calibrated cameras)

1) Compute accurate F, E-matrix between each pair of neighboring views
2) Uncalibrated case: derive intrinsic camera parameters for each pair
3) Compute initial reconstruction of each pair of neighboring views
4) Compute an initial full 3D reconstruction
5) Bundle-Adjustment to minimize overall geometric error



Reconstruct in step 2): (P_1, P_2); (P_2, P_3); (P_3, P_4) …

[See page 453 HZ]

Derive Intrinsic Camera parameters form F

- Formulas: $\boldsymbol{x} = P\,\boldsymbol{X}, \boldsymbol{x} = K\,R\,(I_{3\times3} \mid -\tilde{\boldsymbol{C}})\,\boldsymbol{X}$, $K = \begin{bmatrix} f & s & p_x \\ 0 & mf & p_y \\ 0 & 0 & 1 \end{bmatrix}$
- Given F we would like to derive K_0, K_1 for both views
- Guess $s = 0, m = 1, p_x, p_y$ image centre (later refined in bundle adjustment)
- Compute f_0, f_1:
  1. Adjust K to have $p_x = 0, p_y = 0$: $T = \begin{bmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{bmatrix}$ then $TK = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$
  2. Between two views we have the so-called Kruppa equations:
     (see explanation in HZ ch. 19.4)
     $$\frac{u_1^T (K_0\,K_0^T) u_1}{\sigma_0^2 v_0^T (K_1\,K_1^T) v_0} = \frac{u_0^T (K_0\,K_0^T) u_1}{\sigma_0 \sigma_1 v_0^T (K_1\,K_1^T) v_1} = \frac{u_0^T (K_0\,K_0^T) u_0}{\sigma_1^2 v_1^T (K_1\,K_1^T) v_1}$$
     $$\text{where SVD of } F = [u_0\; u_1\; e_1] \begin{bmatrix} \sigma_0 & 0 & 0 \\ 0 & \sigma_1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_0^T \\ v_1^T \\ e_0^T \end{bmatrix}$$
  3. This can be solved for f_0, f_1 in closed form (see next slide)

p.s. There is lots of additional theory and concepts for reconstruction form uncalibrated cameras (skipped here, see lectures of previous years)

The solution for f_0, f_1

$$\frac{a + b f_0^2}{c + d f_1^2} \overset{①}{=} \frac{a' + b' f_0^2}{c' + d' f_1^2} \overset{②}{=} \frac{a'' + b'' f_0^2}{c'' + b'' f_1^2}$$

(change cond)

$$\Rightarrow a + b f_1^2 + c f_0^2 + d f_0^2 f_1^2 = 0 \quad ①$$

$$a' + b' f_1^2 + c' f_0^2 + d' f_0^2 f_1^2 = 0 \quad ②$$

set $x := f_0^2$ $\quad y := f_1^2$

$$① \quad a + by + cx + dxy = 0 \Rightarrow x = \frac{-a-by}{c+dy}$$

$$② \quad a' + b'y + c'x + d'xy = 0$$

put ① in ②

$$a' + b'y + \frac{c'(-a-by)}{c+dy} + d'y\left(\frac{-a-by}{c+dy}\right) = 0 \quad \Big| \cdot (c+dy)$$

$$\Rightarrow a + by + cy^2 = 0$$

$$\Rightarrow y = \pm\sqrt{d}$$

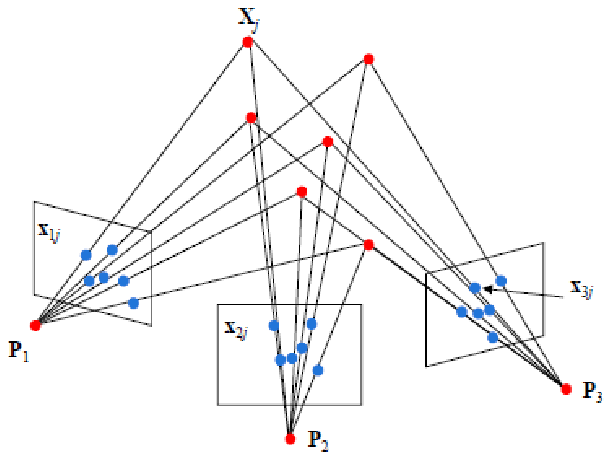
$$\Rightarrow f_1 = \pm\sqrt{\pm\sqrt{d}} = d^{1/4} \quad \text{since } f_1 \text{ positive}$$

from ①: $\quad a + b f_0^2 = 0 \Rightarrow f_0 = \sqrt{-\frac{a}{b}}$

Reconstruction Algorithm

Procedure: (calibrated and un-calibrated cameras)

1) Compute accurate F, E-matrix between each pair of neighboring views
2) Uncalibrated case: derive intrinsic camera parameters for each pair
3) Compute initial reconstruction of each pair of neighboring views
4) Compute an initial full 3D reconstruction
5) Bundle-Adjustment to minimize overall geometric error

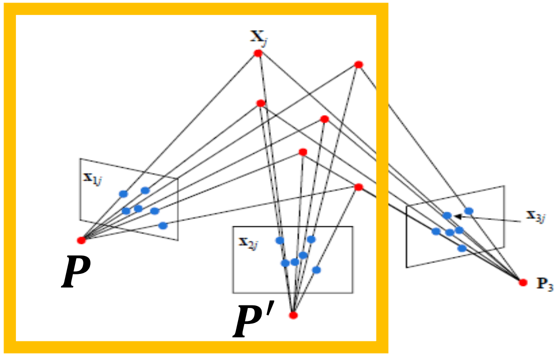


Reconstruct in step 2): (P_1, P_2); (P_2, P_3); (P_3, P_4) …

[See page 453 HZ]

Compute both Camera Matrices

X_i, P, P'



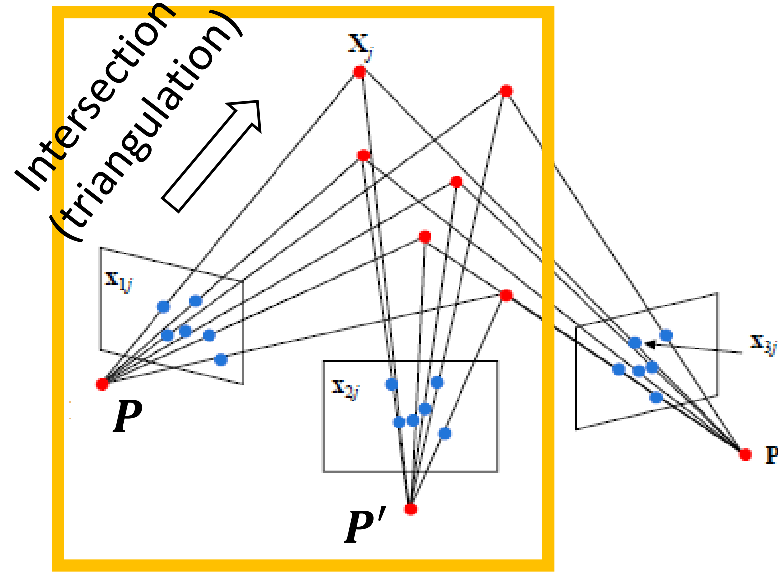
- We have seen that we can get: $R, \tilde{T}$(up to scale) from E (1 solution for 6+ points)

- We have set in previous lecture the camera matrices to:

$$x_0 = \underbrace{K_0[I|0]}_{P}\, X \text{ and } x_1 = \underbrace{K_1 R^{-1}[I|-\tilde{T}]}_{P'}\, X$$

Compute $X_{i's}$

- Input: x, x', P, P'
- Output: $X_{i's}$
- Process called Triangulation or "Intersection"
- Simple solution for algebraic error:



1) $\lambda x = P\ X$ and $\lambda' x' = P'\ X$

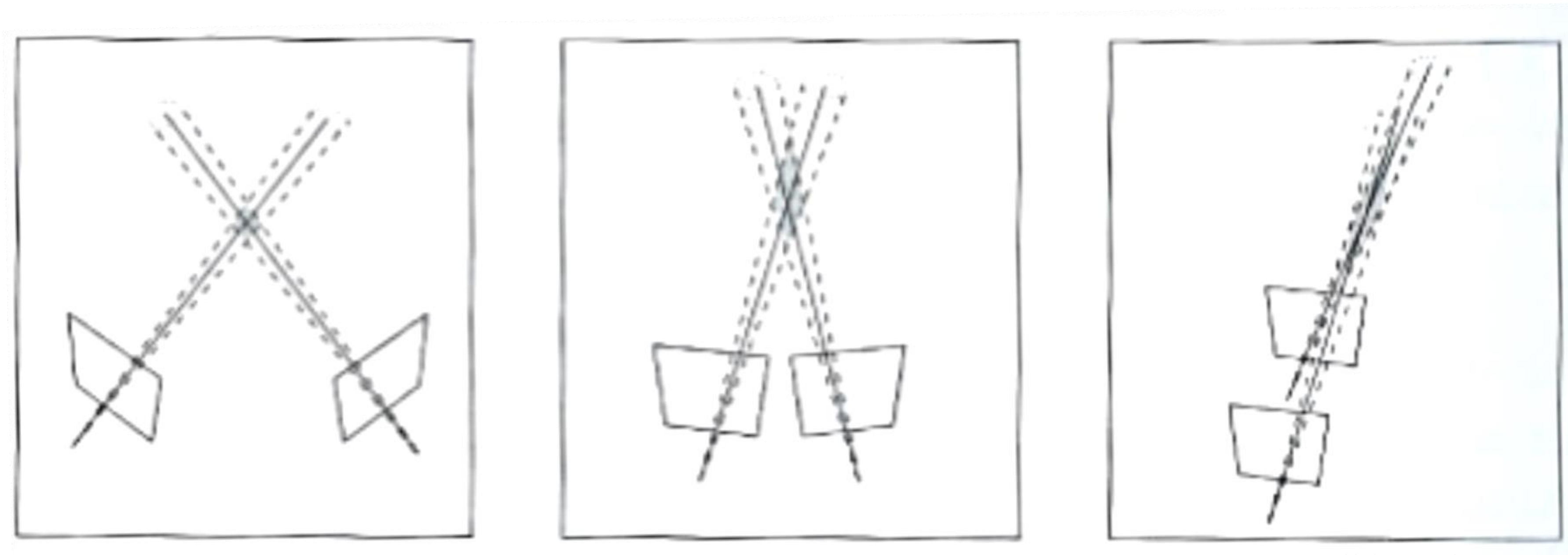
3x4 matrix

2) Eliminate λ by taking ratios. This gives 2x2 linear-independent equations for 4 unknowns: $X = (X_1, X_2, X_3, X_4)$, and we want: $\|X\| = 1$.
(remember X is a homogenous 4D vector, hence scale has to be fixed)

An example ratio is: $\frac{x_1}{x_2} = \frac{p_1\ X_1 + p_2 X_2 + p_3 X_3 + p_4 X_4}{p_5\ X_1 + p_6 X_2 + p_7 X_3 + p_8 X_4}$

3) This gives (as usual) a least square optimization problem:
$A\ X = 0$ with $\|X\| = 1$ where A is of size 4×4.
This can be solved in closed-form using SVD.

Triangulation: Uncertainty



Large baseline
Smaller uncertainty area

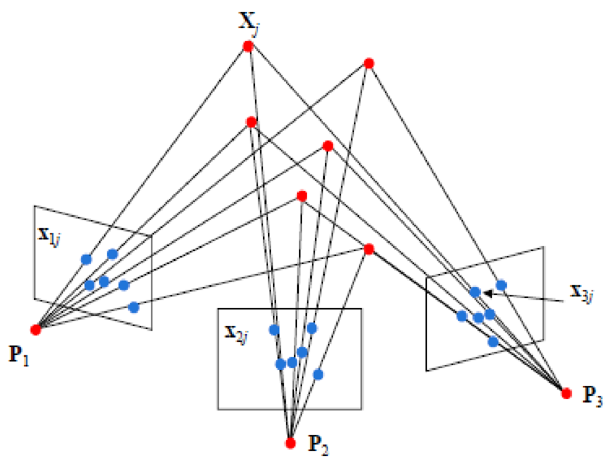
Smaller baseline
Larger uncertainty area

Very small baseline
Very large
uncertainty area

Reconstruction Algorithm

Procedure: (calibrated and un-calibrated cameras)

1) Compute accurate F, E-matrix between each pair of neighboring views
2) Uncalibrated case: derive intrinsic camera parameters for each pair
3) Compute initial reconstruction of each pair of neighboring views
4) Compute an initial full 3D reconstruction
5) Bundle-Adjustment to minimize overall geometric error



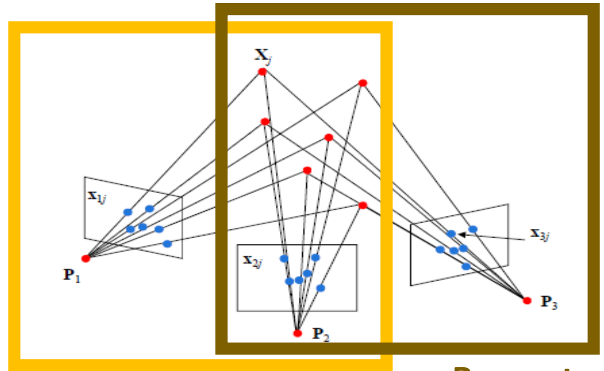
Reconstruct in step 2): (P_1, P_2); (P_2, P_3); (P_3, P_4) …

[See page 453 HZ]

Stitch Pairs of Views together

Three views of a camera:

X_i, P_1, P_2



X_i', P_2', P_3'

Reconstruct Points and Camera 1 and 2

Reconstruct Points and Camera 2 and 3 (denote with a dash)

- Both reconstructions share: 5+ 3D points and one camera (here P_2, P_2').
  (We denote the second reconstruction with a dash)

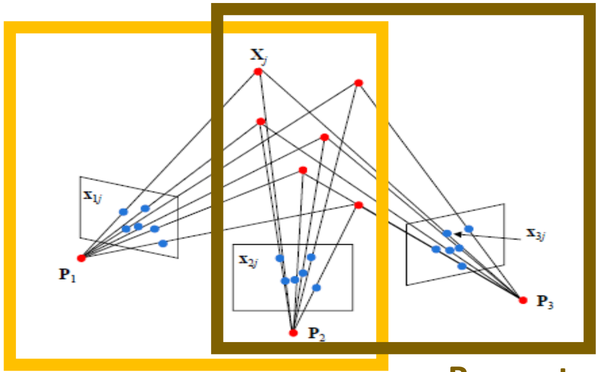
- Why are $X_\mathrm{i}, X_i{}'$ not the same?
  In general we have the following ambiguity: $x_{ij} = P_j X_i = P_j Q^{-1} Q X_i = P_j' X_i'$

- **Our Goal:** make $X_i = X_i'$ and $P_2 = P_2'$ such that $x_{ij} = P_j X_i$ and $x'_{ij} = P'_j X'_i$
  (remember all mean "=" mean equal up to scale. All elements, x, X and P are defined up to scale)

Stitch Pairs of Views together

Three views of a camera:

X_i, P_1, P_2



X'_i, P'_2, P'_3

Reconstruct Points and Camera 1 and 2

Reconstruct Points and Camera 2 and 3 (denote with a dash)

Method:

- Compute Q such that $X_{1-5} = QX'_{1-5}$ (up to scale)
- This can be done from 5+ 3D points in usual least-square sense ($\|AQ\|$), since each point gives 3 equations and Q has 15 DoF.

  An example ratio is: $\frac{X^1}{X^2} = \frac{Q_{11}X^{1\prime}+Q_{12}X^{2\prime}+Q_{13}X^{3\prime}+Q_{14}X^{4\prime}}{Q_{21}X^{1\prime}+Q_{22}X^{2\prime}+Q_{23}X^{3\prime}+Q_{24}X^{4}}$

  for $X_1 = (X^1, X^2, X^3, X^4)$; $X'_1 = (X^{1\prime}, X^{2\prime}, X^{3\prime}, X^{4\prime})$

- Convert the second (dashed) reconstruction into the first one:

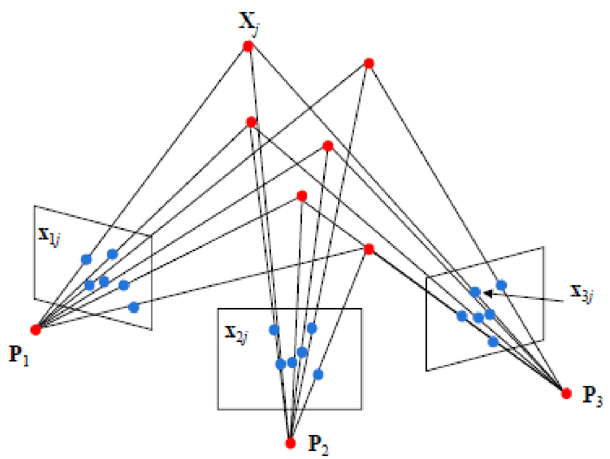
  $P'_{2,3}(new) = P'_{2,3}Q^{-1}$; $\quad X'_i(new) = QX'_i$ (note: $x_{ij} = P_jX_i = P_jQ^{-1}QX_i$)

- In this way you can "zip" all reconstructions into a single one, in sequential fashion.

Reconstruction Algorithm

Procedure: (calibrated and un-calibrated cameras)

1) Compute accurate F, E-matrix between each pair of neighboring views
2) Uncalibrated case: derive intrinsic camera parameters for each pair
3) Compute initial reconstruction of each pair of neighboring views
4) Compute an initial full 3D reconstruction
5) Bundle-Adjustment to minimize overall geometric error



Reconstruct in step 2): (P_1, P_2); (P_2, P_3); (P_3, P_4) …

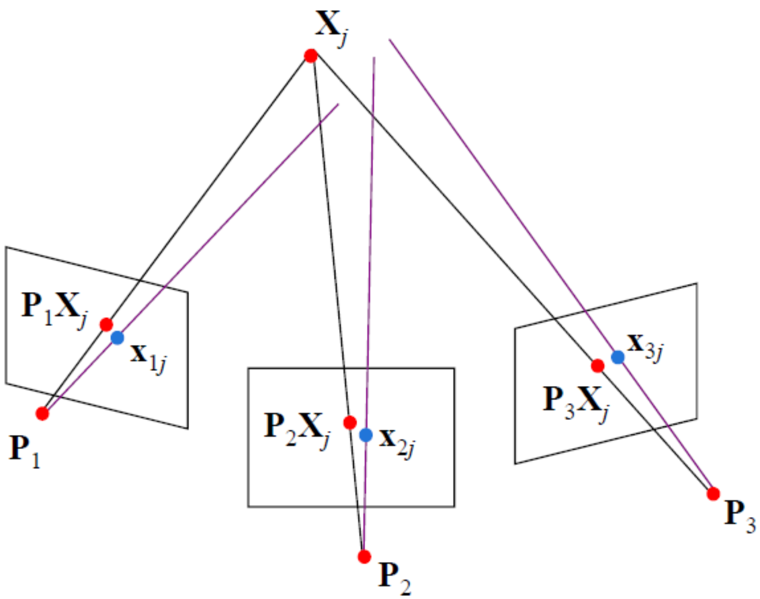
[See page 453 HZ]

Bundle adjustment

- Global refinement of jointly structure (points) and cameras
- Minimize geometric error: $argmin_{\{P_j, X_i\}} \sum_j \sum_i \alpha_{ij}\, d(P_j X_i, x_{ij})$

  here α_{ij} is 1 if X_j visible in view P_j (otherwise 0)

- Non-linear optimization with e.g. Levenberg-Marquard



Example – Reconstruction from a Video