Computer Vision I - Robust *Multi-View 3D Reconstruction*

Carsten Rother

28/11/2015

Roadmap for next four lectures

- Appearance-based Matching (sec. 4.1)

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

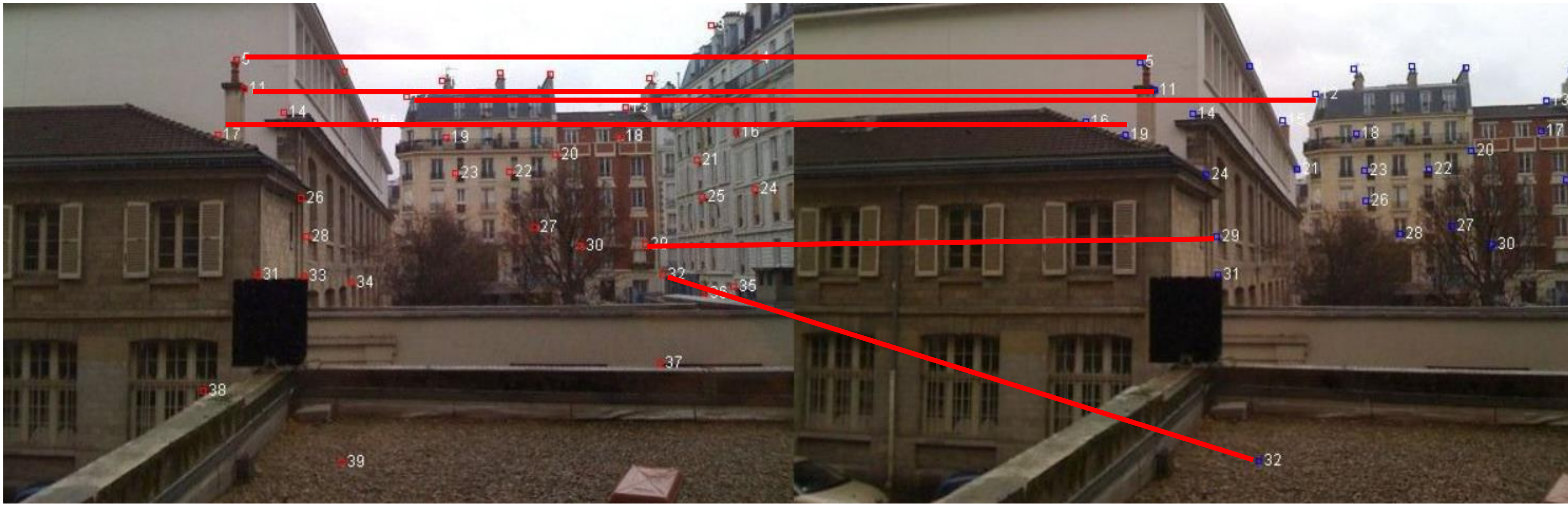
- Geometry of a Single Camera (sec 2.1.5, 2.1.6)
  - Camera versus Human Perception
  - The Pinhole Camera
  - Lens effects

- 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 cameras (sec. 6.1.4)

- Multi-View 3D reconstruction (sec. 7.3-7.4)
  - General scenario
  - From Projective to Metric Space
  - Special Cases

In last lecture we asked (for rotating camera)…



Question 1: If a match is completely wrong then $argmin_h ||A\boldsymbol{h}||$ is a bad idea

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

Robust model fitting

RANSAC:

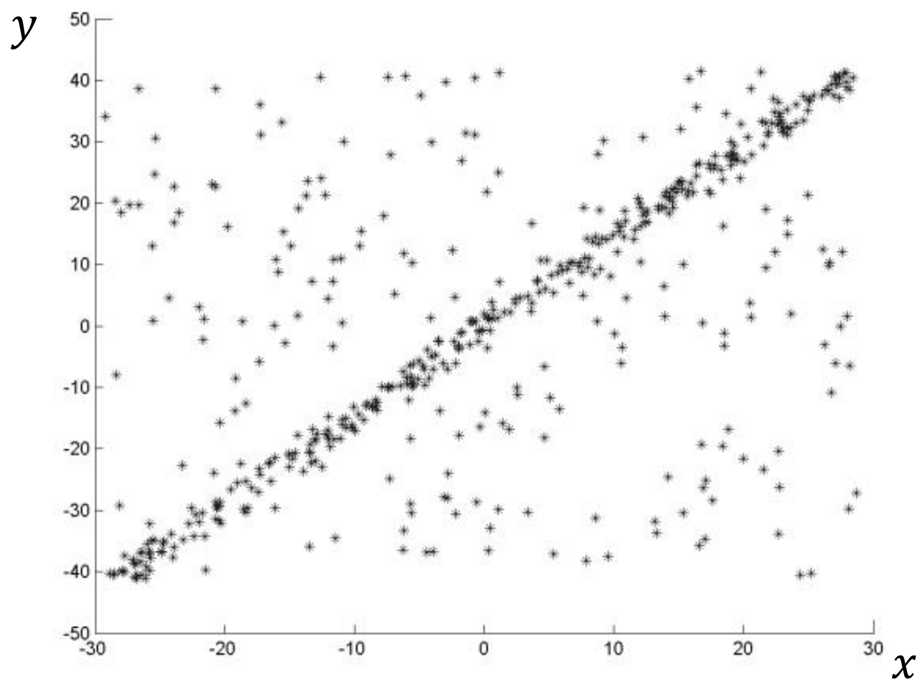
Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography

Martin A. Fischler and Robert C. Bolles (June 1981).

[Side credits: Dimitri Schlesinger]

Example Tasks

Search for a straight line in a clutter of points



i.e. search for parameters a and b for the model $ax + by = 1$

given a training set $\left((x^1, y^1), (x^2, y^2) \dots (x^i, y^i)\right)$

Example Tasks

Estimate the fundamental matrix F


i.e. parameters satisfying

$$[x_{l1}, x_{l2}, 1] \cdot \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \cdot \begin{bmatrix} x_{r1} \\ x_{r2} \\ 1 \end{bmatrix} = 0$$

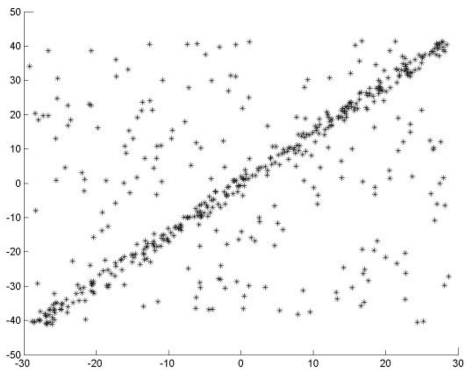
given a training set of correspondent pairs

$$\Big((x_l^1, x_r^1), (x_l^2, x_r^2) \dots (x_l^i, x_r^i) \Big)$$

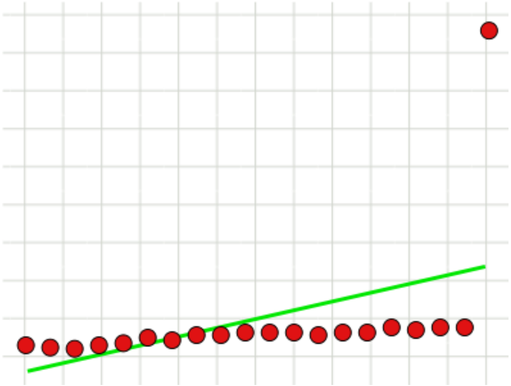
For Homography of rotating camera we have: $x_l^i H = x_r^i$

Two sources of errors

1. **Noise**: the coordinates deviate from the true ones according to some “rule” (probability) – the father away the less confident
2. **Outliers**: the data have nothing in common with the model to be estimated



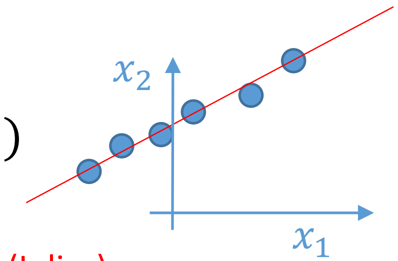
Ignoring outliers can lead to a wrong estimation.
→ The way out: find outliers explicitly, estimate the model from inliers only



Task formulation

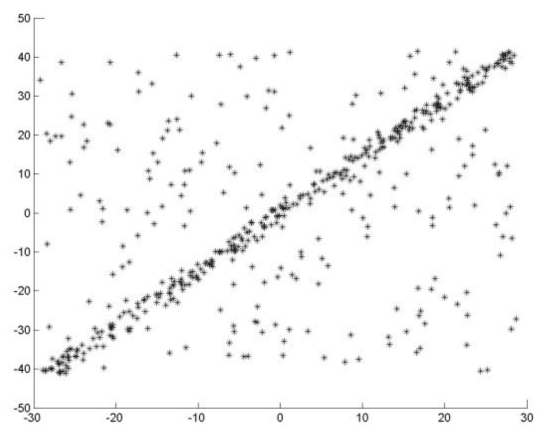
Let $x \in \mathcal{X}$ be the input space and $y \in \mathcal{Y}$ be the parameter space.
The training data consist of data points $L = (x^1, x^2 \ldots x^i),\ x^i \in \mathcal{X}$

Let an **evaluation function** $f : \mathcal{X} \times \mathcal{Y} \rightarrow \{0, 1\}$ be given that checks the consistency of a point x with a model y.



- Straight line $f(x_1, x_2, a, b) = \begin{cases} 0 & if\ |ax_1 + bx_2\ - 1\,| \le t\ (e.g.\ 0.1) \\ 1 & otherwise \end{cases}$ (Inlier) (Outlier)
- Fundamental matrix $f(x_l, x_r, F) = \begin{cases} 0 & if\ |\,x_l^t F x_r| \le t\ (e.g.\ 0.1) \\ 1 & otherwise \end{cases}$ (Inlier) (Outlier)

The task is to find the parameter that is consistent with the **majority** of the data points: $y^* = argmin_y \sum_i f(x^i, y)$



First Idea: 2D Line estimation

Question: How to compute: $y^* = argmin_y \sum_i f(x^i, y)$

<u>A naïve approach:</u> enumerate all parameter values

→ know as **Hough Transform** (very time consuming and not possible at all for many free parameters (i.e. high dimensional parameter space)

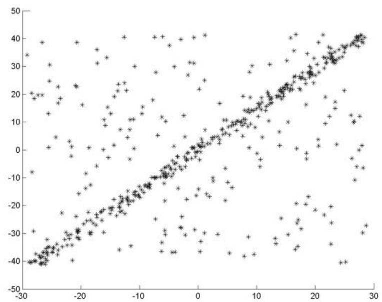
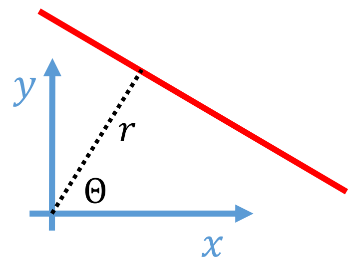


Image with points



Encode all line with two parameters (r, Θ)

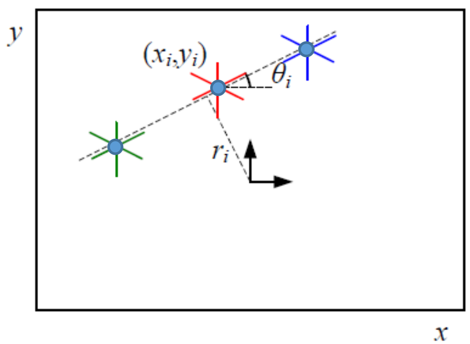
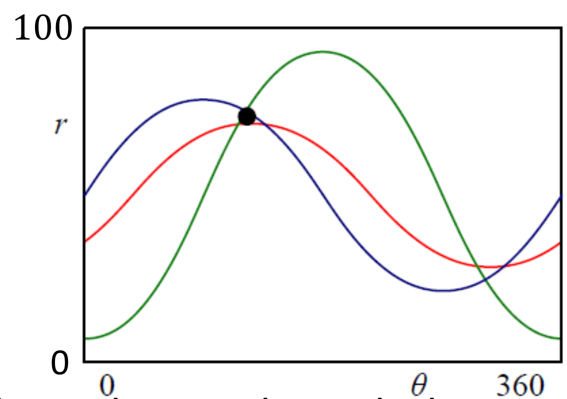


Image with just 3 points

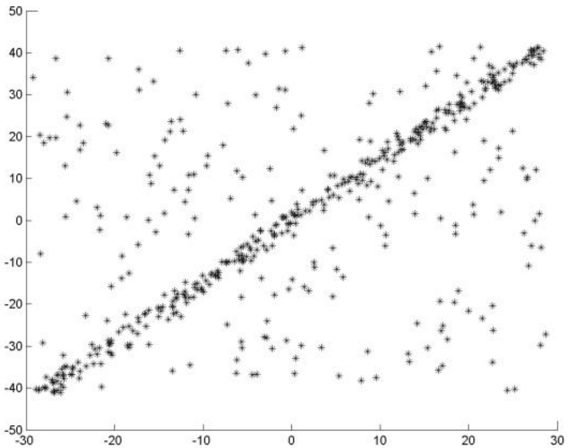
Hough transform



All lines that go through these 3 points

Goal: Find "point" in the figure where most lines meet

First Idea: 2D Line estimation



Hough transform

r

0	1	0	0	2	1	0	0
2	0	3	0	0	1	0	0
2	3	1	0	2	0	0	1
3	3	3	4	4	1	3	0
3	3	4	3	1	0	1	0
9	10	9	3	1	3	1	0
20	**200**	25	4	2	3	0	0
50	35	5	10	5	2	4	4

Θ

Accumulator space
number of inliers (sketched)

- Observation: The parameter space have very low counts
- Idea: do not try all values but only some of them. Which ones?

Data-driven Oracle

An **Oracle** is a function that predicts a parameter given the minimum amount of data points (d-tuple): $g : \mathcal{X}^d \rightarrow \mathcal{Y}$

Examples:

- Line can be estimated from $d = 2$ points
- Fundamental matrix from $d = 7$ or 8 points correspondences
- Homography can be computed from $d = 4$ points correspondences

First Idea: Do not enumerate all parameter values but all d-tuples of data points
That is then n^d number of tests, e.g. n^2 for lines (with n points)
The optimization is performed over a **discrete domain**.

$$y^* = argmin_y \sum_i f(x^i, y)$$

Second Idea: Do not try all subsets, but sample them randomly

RANSAC

[Random Sample Consensus, Fischler and Bolles 1981]

Basic RANSAC method:

Can be done in parallel!

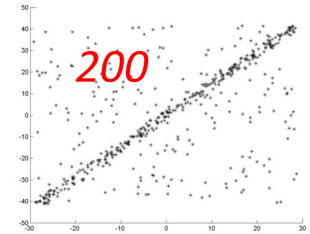
Repeat many times
    select d-tuple, e.g. (x^1, x^2) for lines
    compute parameter(s) y, e.g. line $y = g(x^1, x^2)$
    evaluate $f'(y) = \sum_i f(x^i, y)$
    If $f'(y) \leq f'(y^*)$
        set $y^* = y$ and keep value $f'(y^*)$

- 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?

Convergence

Observation: it is necessary to sample **any** d -tuple of inliers just **once** in order to estimate the model correctly.



$\varepsilon \sim 0.2$

1000 points overall

Let ε be the probability of outliers.

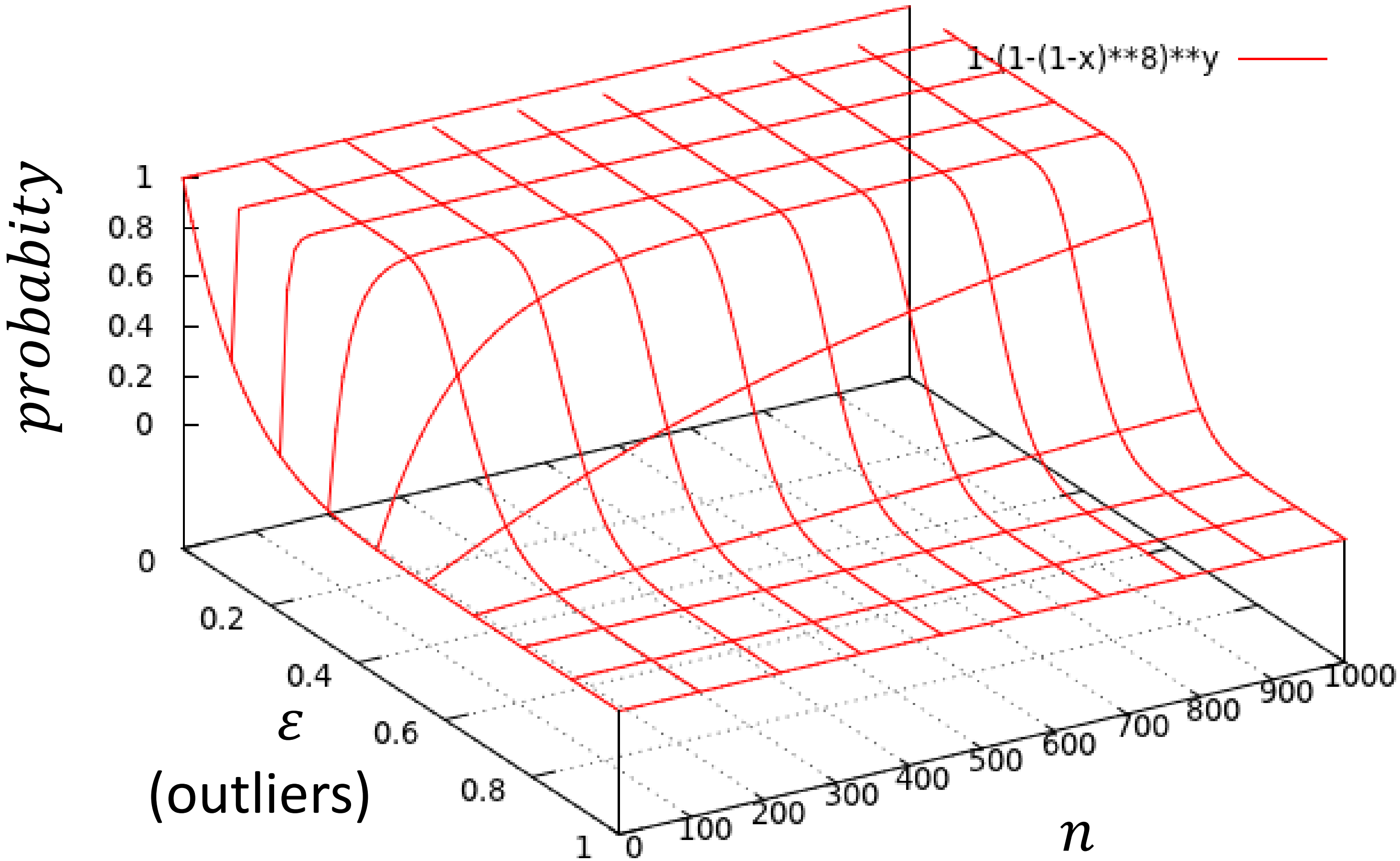
The probability to sample d inliers is $(1-\varepsilon)^d$ (here $0.8^2 = 0.64$)

The probability of a “wrong” d-tuple is $1-(1-\varepsilon)^d$ (here 0.36)

The probability to sample n times only wrong tuples is $(1-(1-\varepsilon)^d)^n$ (here $0.36^{20} = 0.000000013$)

The probability to sample the “right” tuple at least once during the process (i.e. to estimate the correct model according to assumptions) $1-(1-(1-\varepsilon)^d)^n$ (here 99.999999866 %)

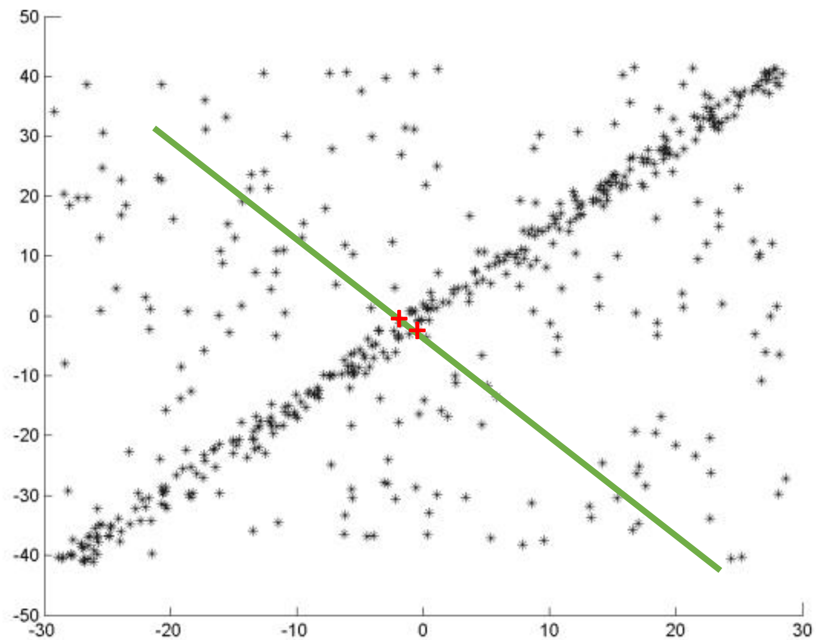
Convergence



$$1-(1-(1-\varepsilon)^d)^n,\ d=8,\ \varepsilon\in[0,1],\ n=1\ldots1000$$

Comment

- In our derivation for $p = 1-(1-(1-\varepsilon)^d)^n$ we were slightly optimistic since „degenerate“ inliers may give rise to bad lines



- However, these bad lines have little support wrt number of inliers
- We also define later a refinement procedure which can correct such bad lines

The choice of the **oracle** is crucial

Example – the fundamental matrix:

a) 8-point algorithm
Probability: 70% ($n = 300; \epsilon = 0.5; d = 8$)

b) 7-point algorithm
Probability: 90% ($n = 300; \epsilon = 0.5; d = 7$)

Number of trials to get p% accuracy (here 99%)

$$p = 1 - \left(1 - (1 - \varepsilon)^d\right)^n$$

$$n = \frac{\log(1 - p)}{\log(1 - (1 - \varepsilon)^d)}$$

d	proportion of outliers ε						
	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

The choice of **evaluation function** is crucial

- **Evaluation function**: $f(x_1, x_2, a, b) = \begin{cases} 1 & if\ |ax_1 + bx_2\ - 1| \leq t\ (e.g. 0.1) \\ 0 & otherwise \end{cases}$

  error function

- **Algebraic error**: Is a measure that has no geometric meaning

  Example: For a line: $d(x_1, x_2, a, b) = |ax_1 + bx_2\ - 1\ |$

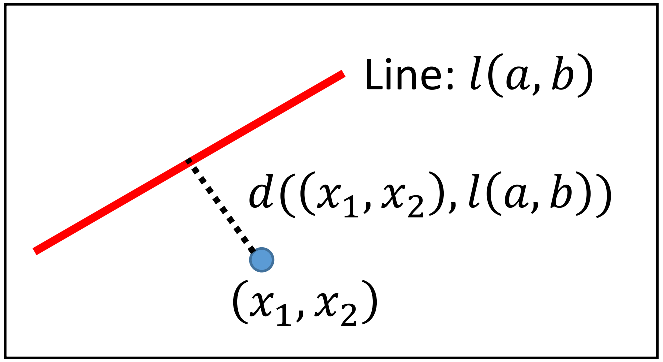
  For a homograpy: $d(x_1, x_2, a, b) = |\boldsymbol{Ah}|$

  (where $\boldsymbol{A}$ is 1×8 matrix derived as above

  For F-matrix: $d(x_l, x_r, F) = \left|\ x_l^t F x_r \right|$

- **Geometric error**: Is a measure that considers a distance in image plane

  Example: For a line: $d(x_1, x_2, a, b) = d(\ (x_1, x_2), l(a, b)\ )$

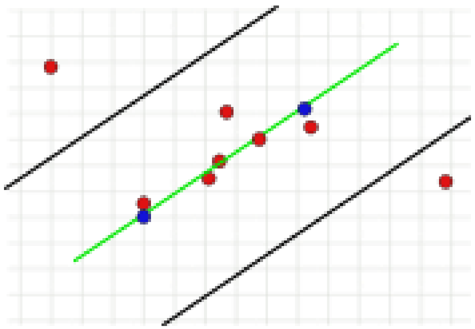


(d is Euclidean distance between point to line)

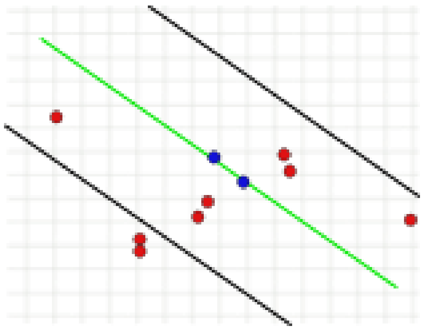
Geometric error: for homography and F-matrix to come

The choice of **confidence interval** is crucial

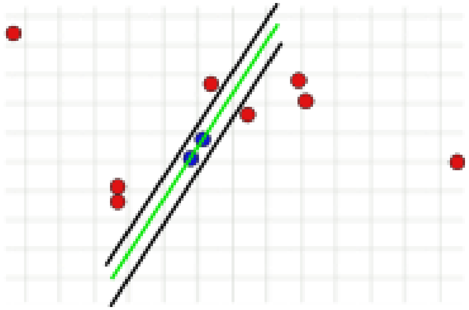
Examples:



Large confidence,
“right” model,
2 outliers



Large confidence,
“wrong” model,
2 outliers



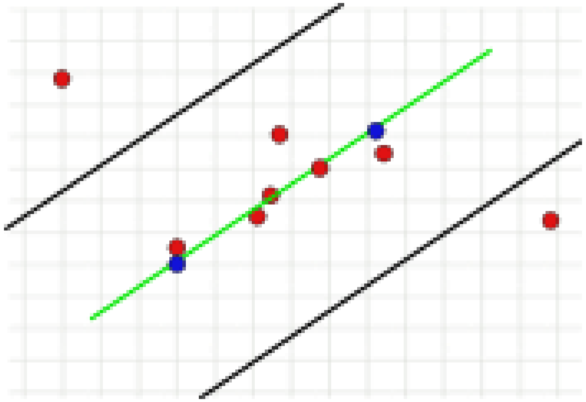
Small confidence,
Almost all points are outliers
(independent of the model)

Extension: Adaptive number of samples n

Choose n in an adaptive way:

1) Fix $p = 99.9\%$ (very large value)
2) Set $n = \infty$ and $\varepsilon = 0.9$ (large value for outlier)
3) During RANSAC adapt n, ε :
   1) Re-compute ε from current best solution
      ε = outliers / all points
   2) Re-Compute new n:

$$n = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^d)}$$



MSAC (**M**-Estimator **SA**mple **C**onsensus)

If a data point is an inlier the penalty is not 0, but it depends on the "distance" to the model.

Example for the fundamental matrix:

$$f(x_l, x_r, F) = \begin{cases} 0 & if \mid x_l^t F x_r \mid \leq t \ (e.g. 0.1) \\ 1 & otherwise \end{cases}$$

becomes

$$f(x_l, x_r, F) = \begin{cases} \mid x_l^t F x_r \mid & if \mid x_l^t F x_r \mid \leq t \ (e.g. 0.1) \\ t & otherwise \end{cases}$$

"robust function"

→ the task is to find the model with the minimum average penalty

$$f(x_l, x_r, F) = \min(|x_l^t F x_r|, t)$$

$$y^* = \arg\min_y \sum_i f(x^i, y)$$

[P.H.S. Torr und A. Zisserman 1996]

Randomized RANSAC

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

Randomized RANSAC:

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

1. Sample m points from L

2. If all of them are good, check all others as before

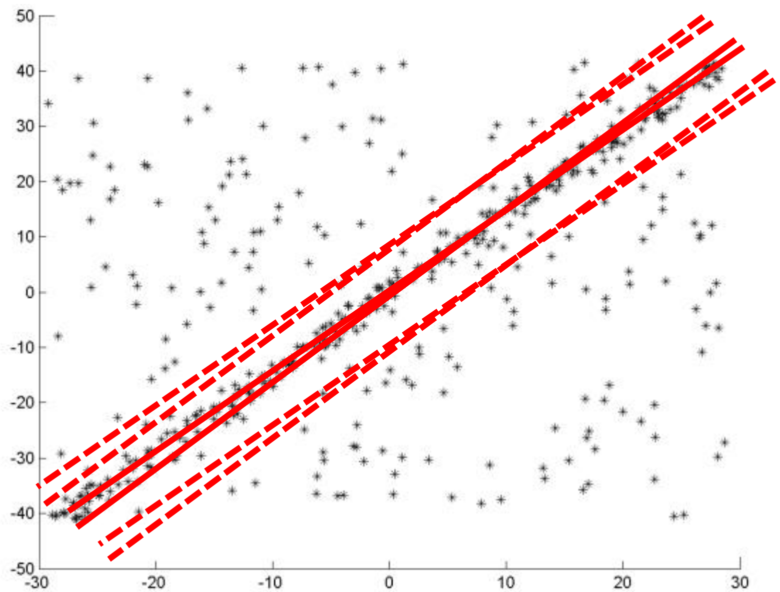
3. If there is at least one bad point, among m, reject the hypothesis

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).

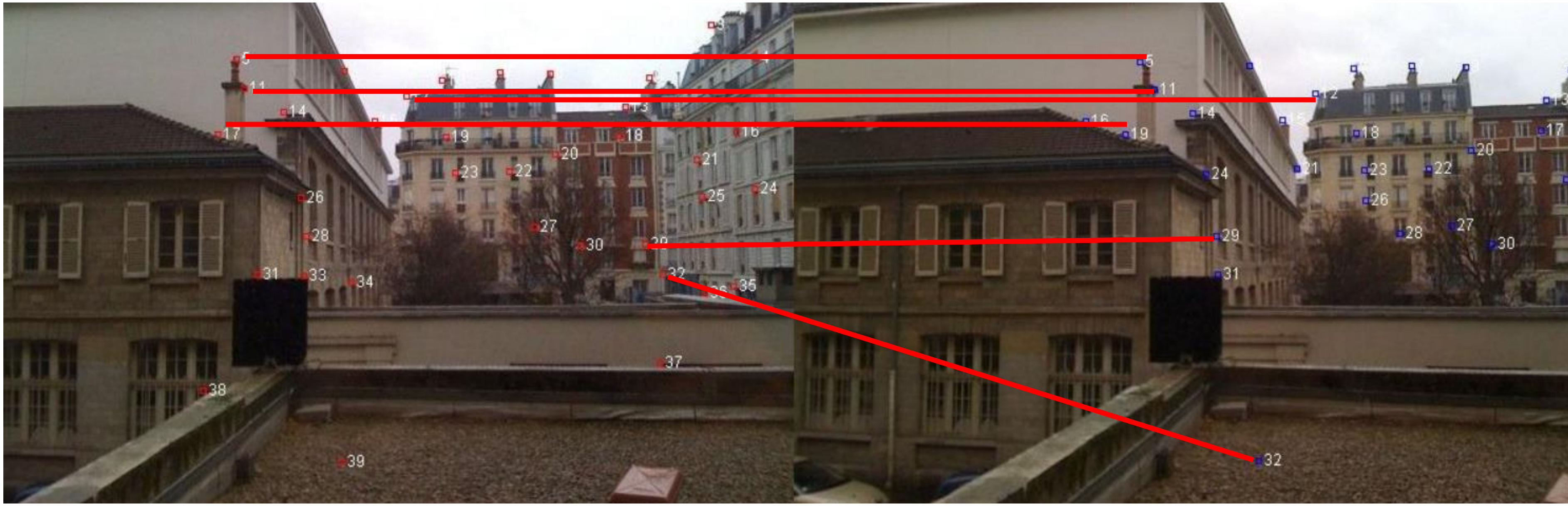
Refinement after RANSAC

<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)



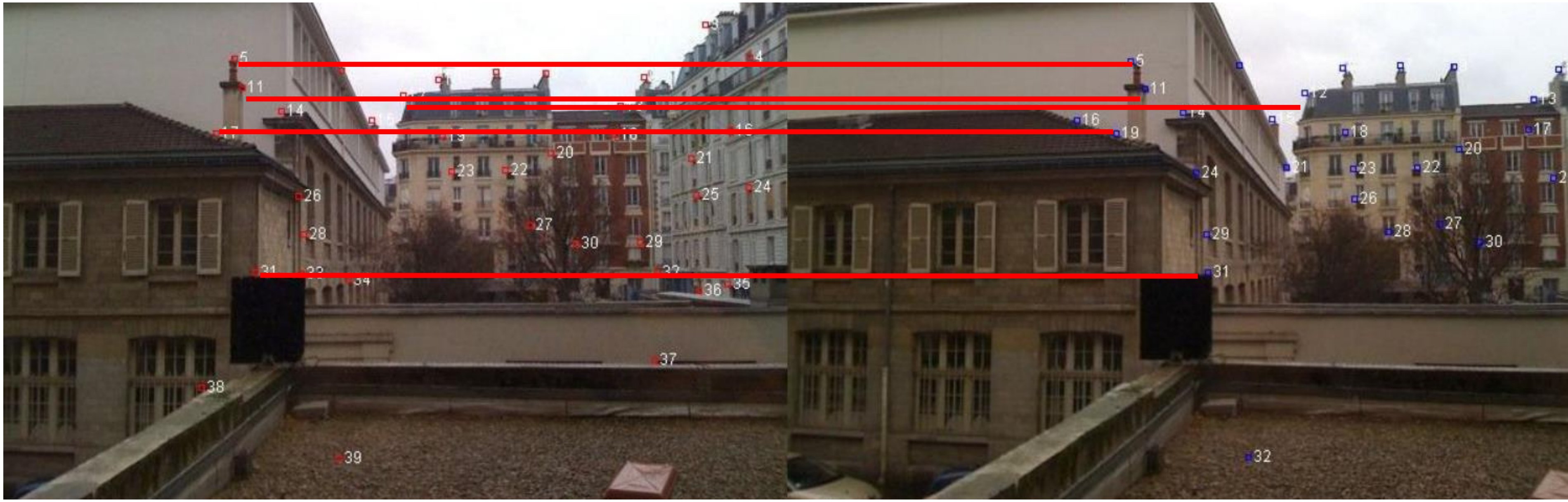
In last lecture we asked (for rotating camera)…



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

Reminder from last Lecture: Homography for rotating camera



Algorithm:

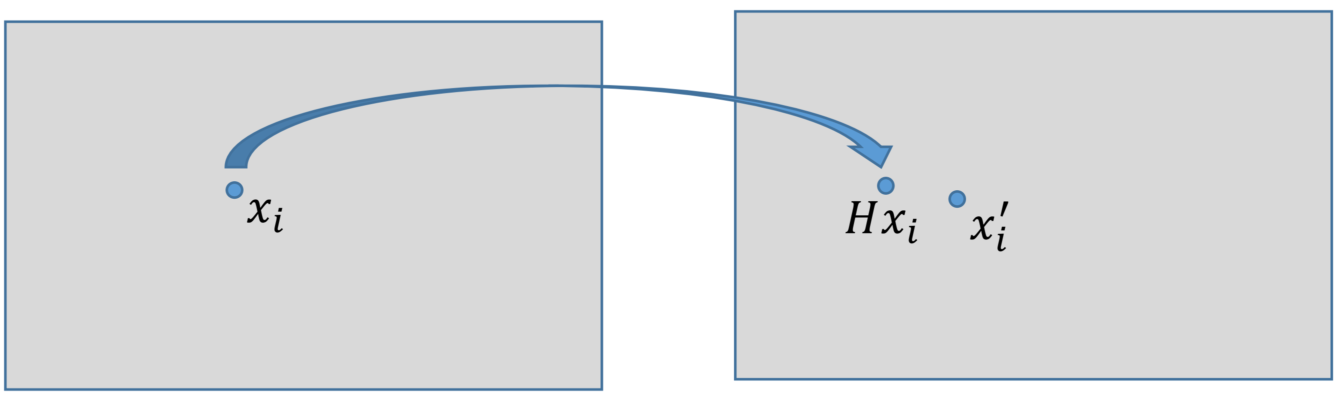
1) Take $m \geq 4$ point matches (x, x')
2) Assemble A with $A\boldsymbol{h} = 0$
3) compute $\boldsymbol{h}^* = argmin_{\boldsymbol{h}} ||A\boldsymbol{h}||$ subject to $||\boldsymbol{h}|| = 1$, use SVD to do this.

Refine Hypothesis H with inliers

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

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

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

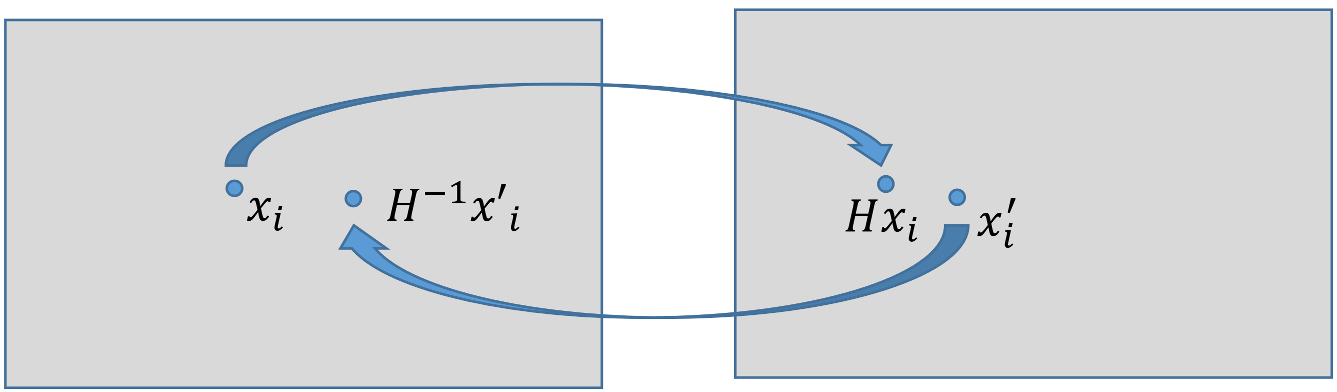


This is not symmetric

Refine Hypothesis H with inliers

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

1. Algebraic error: $argmin_h \|A\boldsymbol{h}\|$
2. *First* geometric error: $H^* = argmin_H \sum_i d(x_i', Hx_i)$
3. *Second,* symmetric geometric error: $H^* = argmin_H \sum_i d(x_i', Hx_i) + d(x_i, H^{-1}x'_i)$

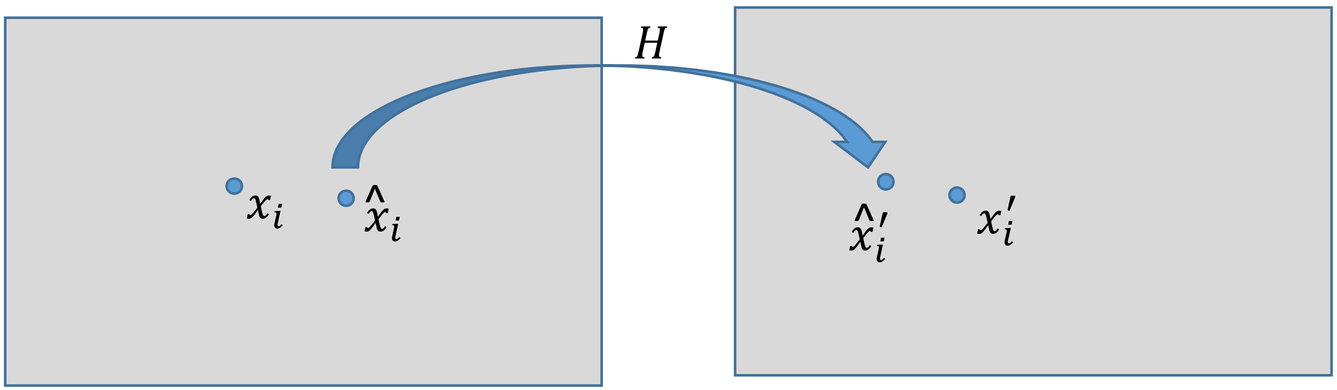


Refine Hypothesis H with inliers

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

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

<u>Comment:</u> 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)

Halfway Slide

1 Min Break

Full Homography Method (HZ page 123)

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–p95) using the Levenberg–Marquardt algorithm of section A6.2(p600).

(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.

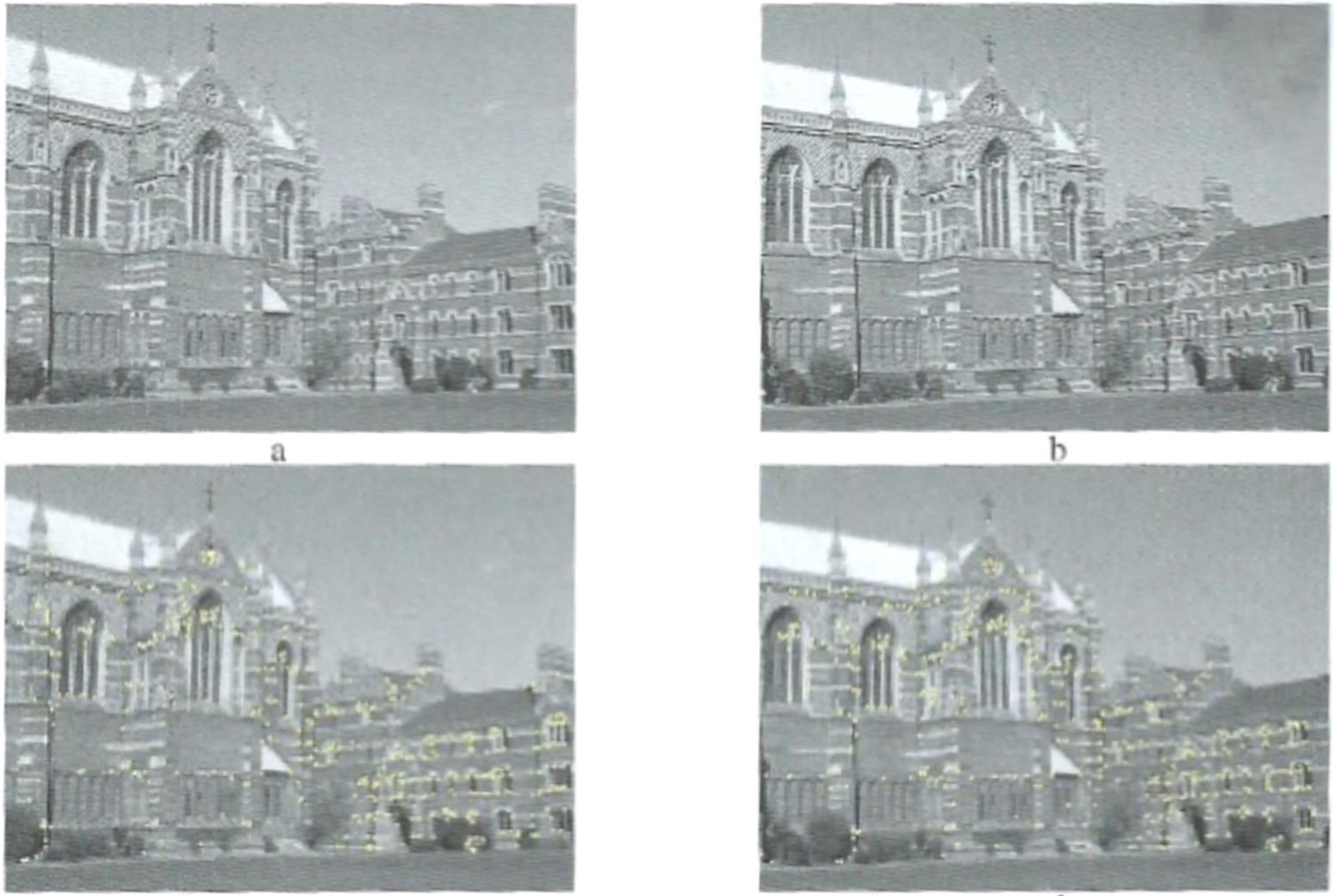
See next slide

This is the optimal geometric error

See next slides

[see details on page 114ff in HZ]

Example



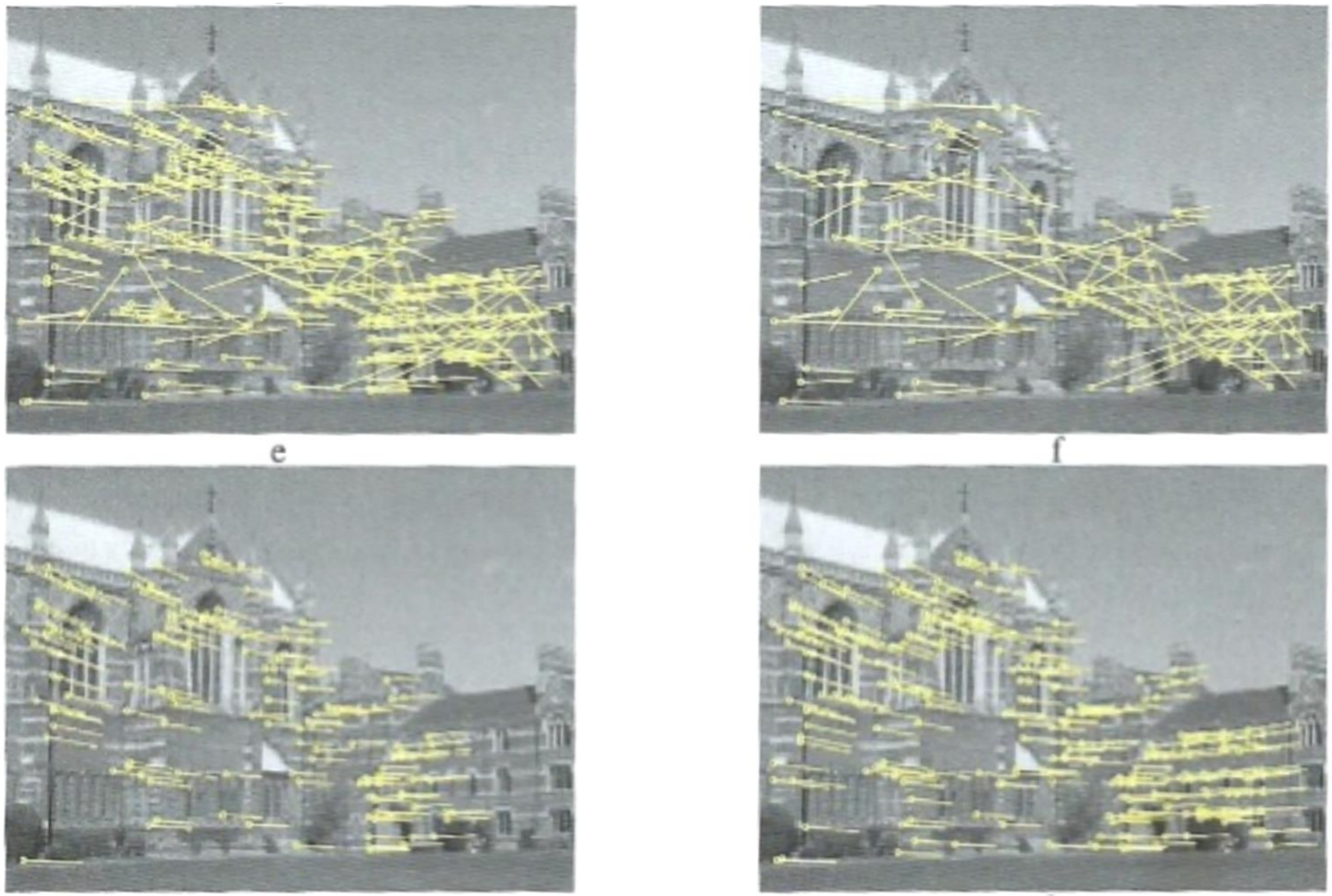
Input images

~500 interest points

Example

268 putative matches

117 outliers found



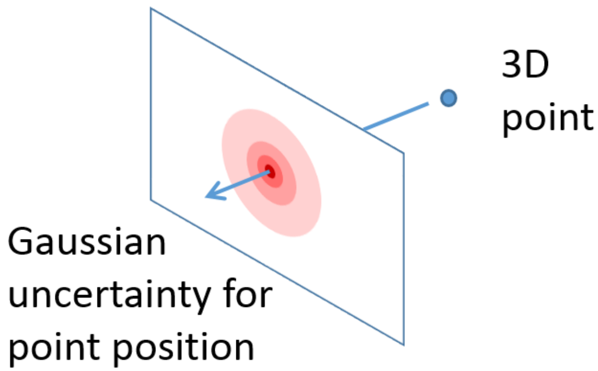
151 inliers found

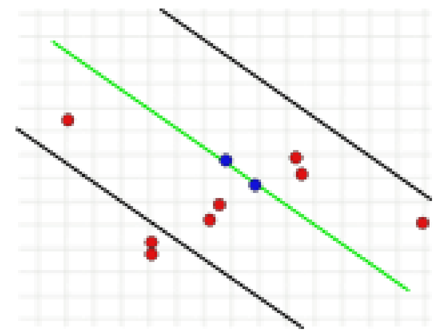
262 inliers after guided matching

Guided matching variant: 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:

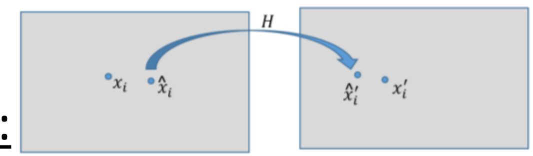
1. For a 2D line: $d(x, l) \leq \sigma\sqrt{3.84} = t$
2. For a Homography: $d(x_l, x_r, H) \leq \sigma\sqrt{5.99} = t$
3. For an F-matrix: $d(x_l, x_r, F) \leq \sigma\sqrt{3.84} = t$

(see page 119 HZ)

Methods for $F/E/H$ Matrix computation - Summary

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 $\hat{x}'^t_i F/E \hat{x}_i = 0$

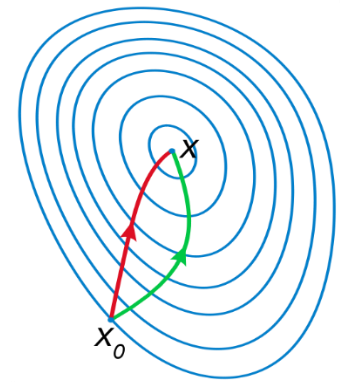
A few word on iterative continuous optimization

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

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

For non-linear (arbitrary) optimization problems:

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

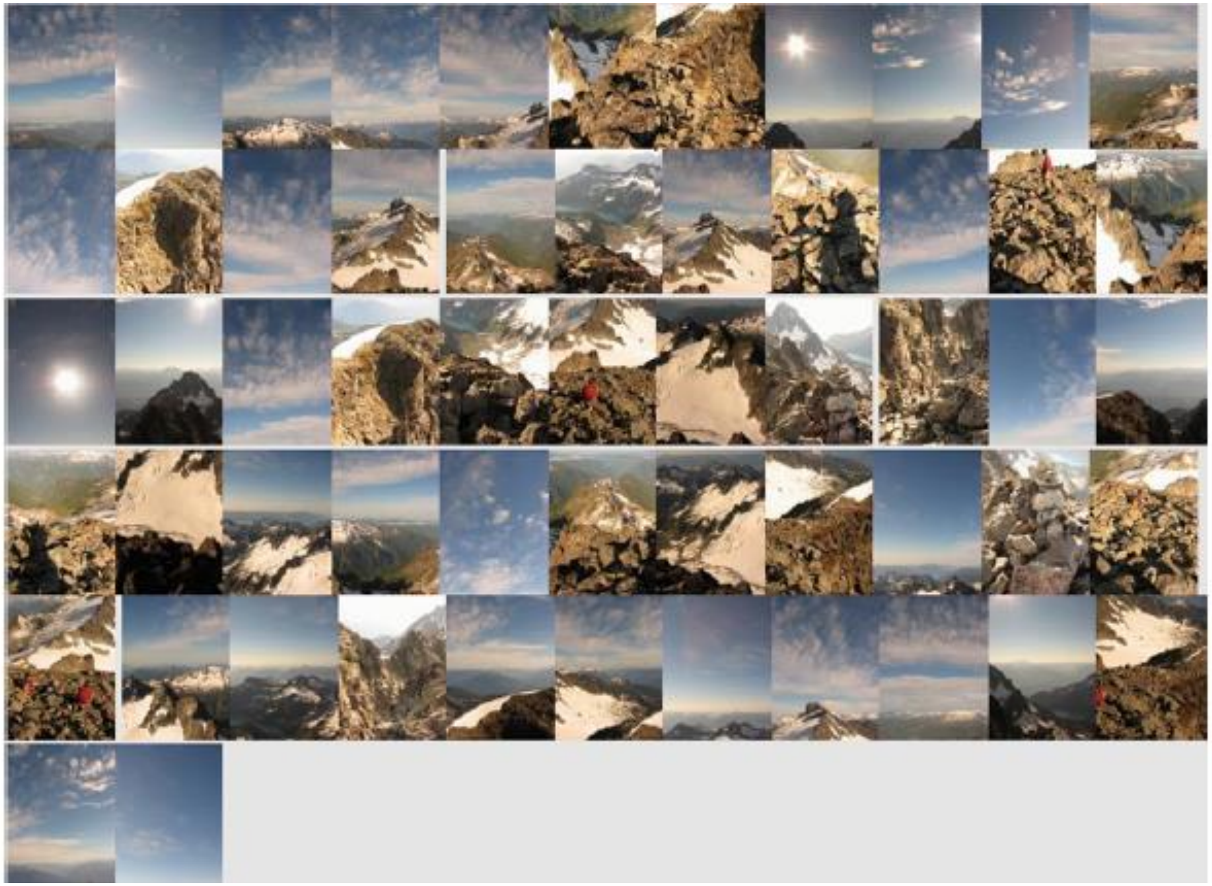


Red Newton's method; green gradient descent

- Iterative Estimation methods (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

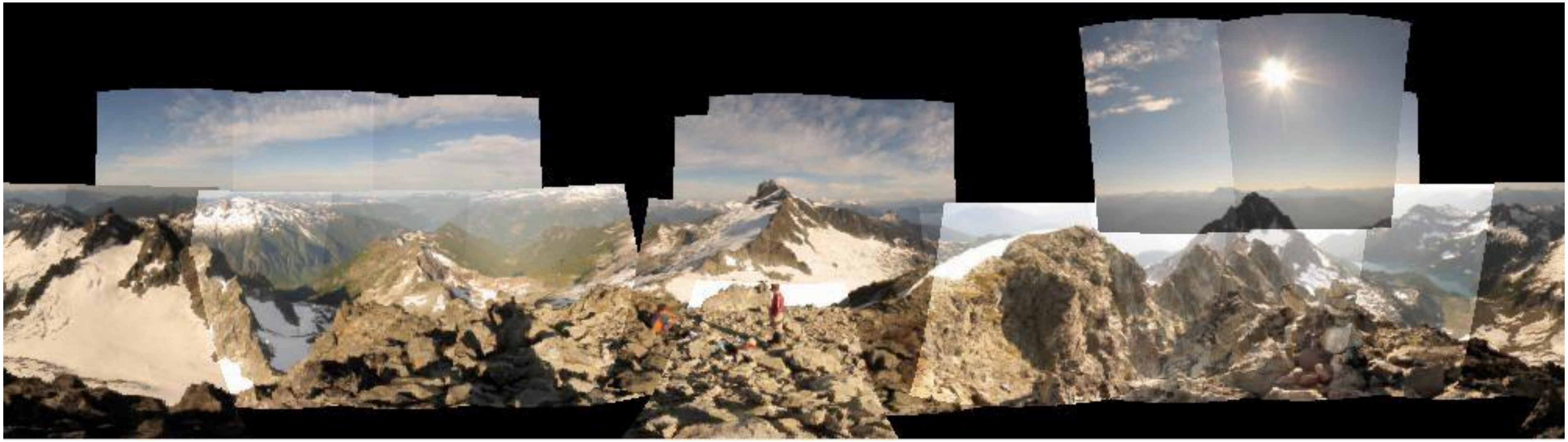
An unordered set of images:



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

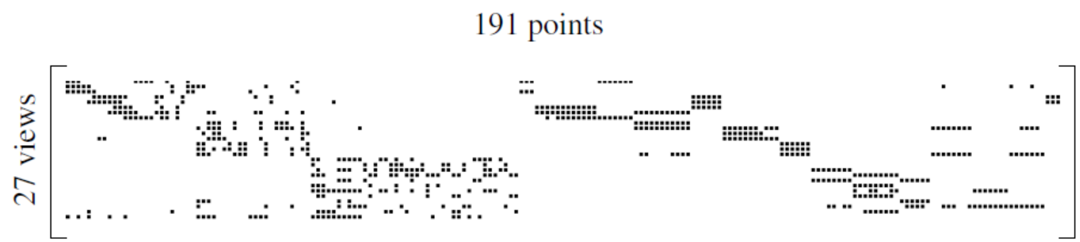
... automatically create a panorama

Roadmap for next four lectures

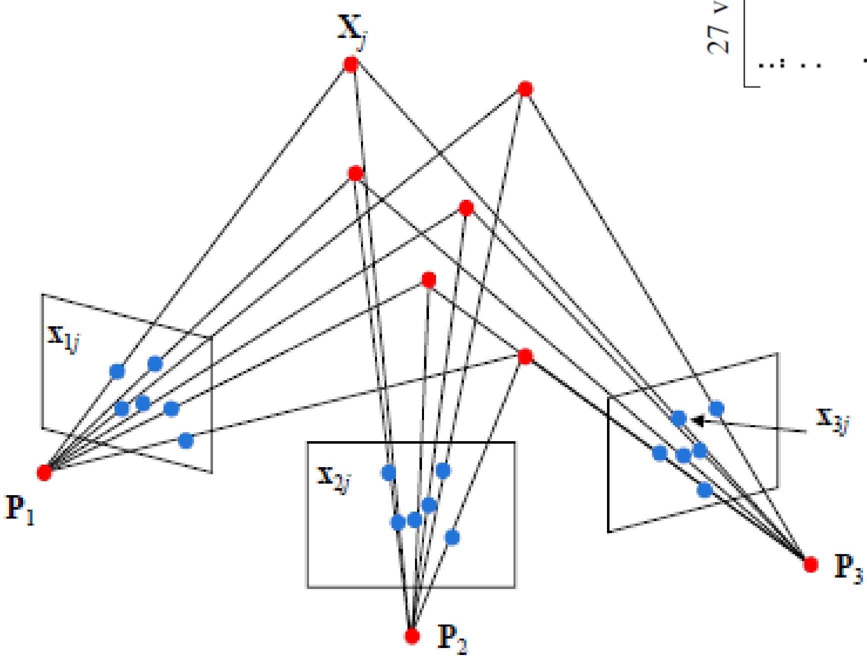
- Appearance-based Matching (sec. 4.1)
- Projective Geometry - Basics (sec. 2.1.1-2.1.4)
- Geometry of a Single Camera (sec 2.1.5, 2.1.6)
  - Camera versus Human Perception
  - The Pinhole Camera
  - Lens effects
- Geometry of two Views (sec. 7.2)
  - The Homography (e.g. rotating camera)
  - Camera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

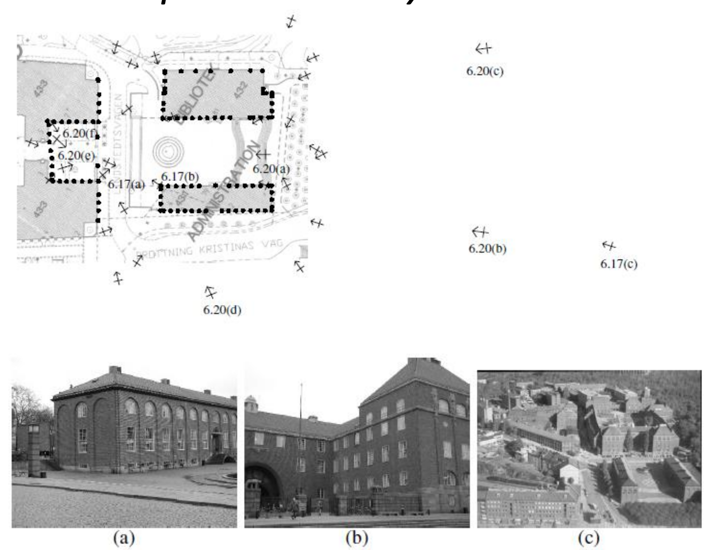
3D reconstruction: Problem definition

- Given image observations in m cameras of n static 3D points
- Formally: $x_{ij} = P_j X_i$ for $j = 1 \dots m; i = 1 \dots n$
- 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

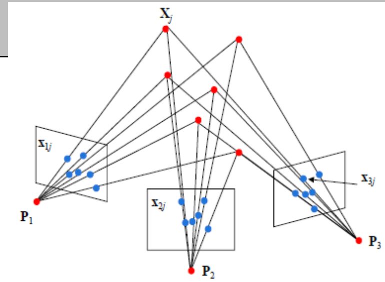


Example: "Visibility" matrix



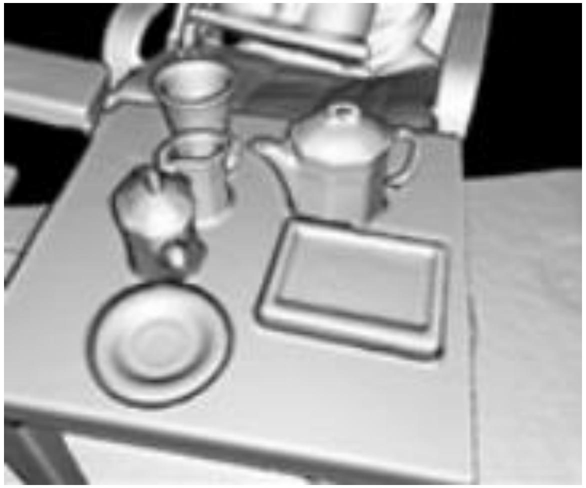


Names: 3D reconstruction



1) Sparse Structure from Motion (SfM)
   In Robotics it 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



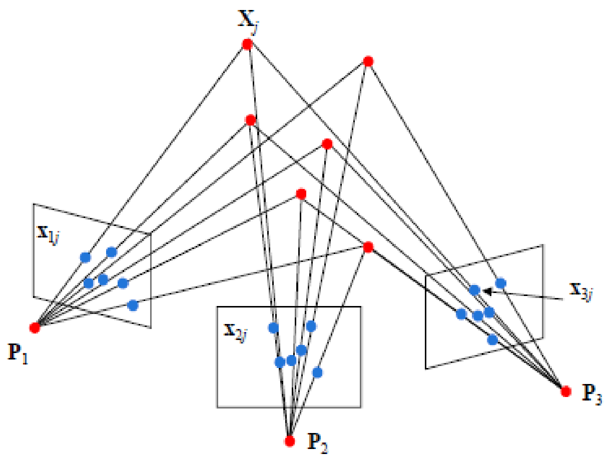
Example: Dense Reconstruction

[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]

Reconstruction Algorithm

<u>Generic Outline</u> (calibrated and un-calibrated cameras)

1) Compute robust F/E-matrix between each pair of neighboring views
2) Compute initial reconstruction of consecutive pair of views
3) Compute an initial full 3D reconstruction
4) Bundle-Adjustment to minimize overall geometric error
5) If cameras are not calibrated then perform auto-calibration (also known as self-calibration)



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

[See page 453 HZ]

Step 2: Compute initial reconstruction of consecutive pair of views

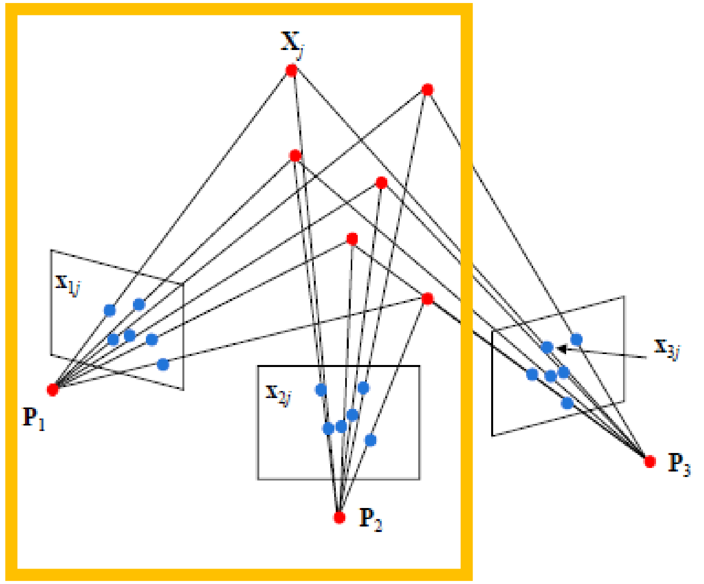
Input:

- Calibrated Cameras: E-matrix, K, K', 5+ matching points (x_i, x'_i)
- Un-calibration Cameras: F-matrices, 7+ matching points (x_i, x'_i)

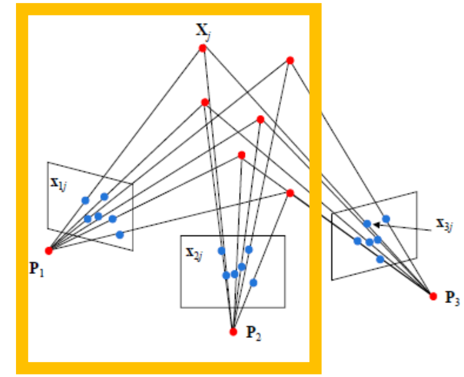
Output: $P, P', X_{i's}$ such that geometric error: PX_i *to* x_i and $P'X_i$ *to* x'_i is small

2-Step Method:

1. Derive P, P'
2. Compute $X_{i's}$ (called Triangulation)

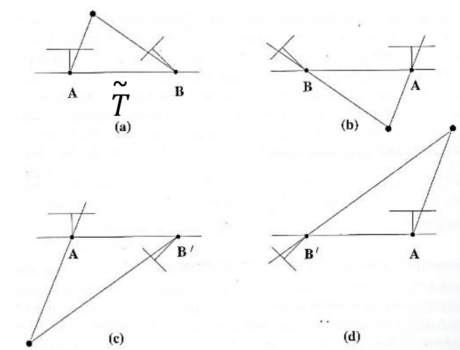


Derive P, P': calibrated case



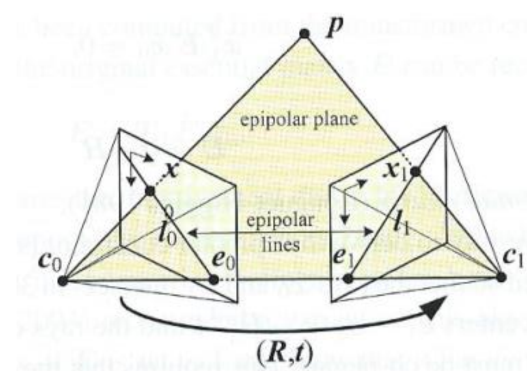
<u>We have done this already</u>:

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



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

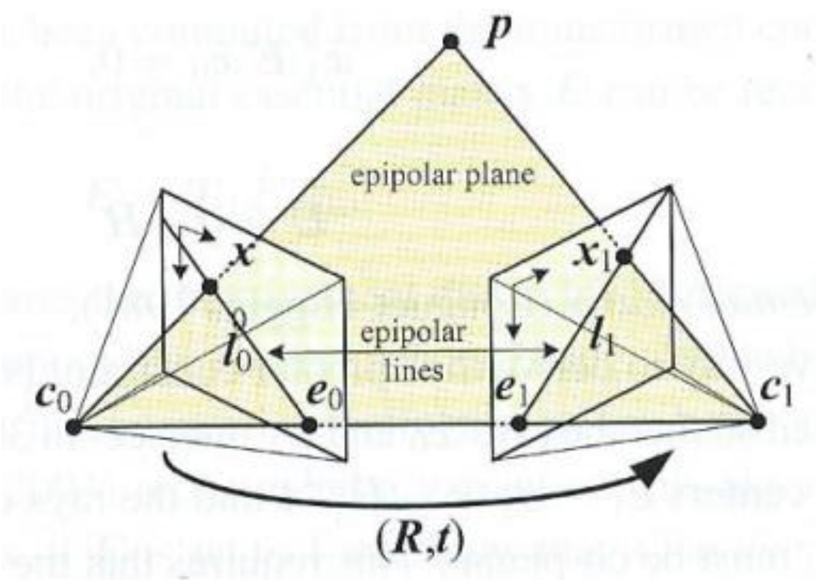
$$x_0 = \underbrace{K_0[I|0]}_{P}\, X \text{ and } x_1 = \underbrace{K_1 R^{-1}[I|-\tilde{T}]}_{P'}\, X$$

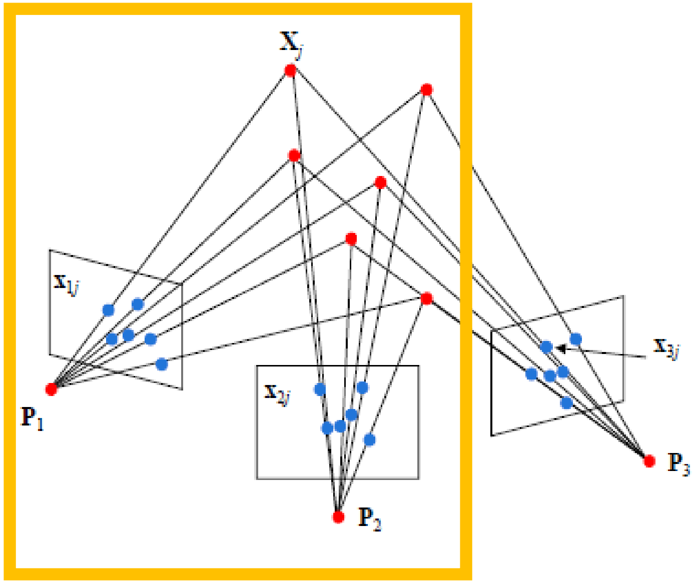


Derive P, P': un-calibrated case

- Derivation (blackboard) see HZ page 256

$$P = [I_{3\times3} \mid 0\,]; \quad P' = [\,[e']_\times F \mid e']$$





Derivation

we need P, P' such that

$x = PX, \quad x' = P'X \quad x'^T F x$ for all X

$$\Leftrightarrow \underset{1\times4}{X^T} \underset{4\times3}{P'^T} \underset{3\times3}{F} \underset{3\times4}{P} \underset{4\times1}{X} = 0 \quad (1)$$

choose $P = [I \mid 0] \quad P' = [SF \mid e']$

$\boxed{15 \text{ DoF fixed}}$

where $S = \left[\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix}\right]_x = \begin{bmatrix} 0 & -c' & b' \\ c & 0 & -a' \\ -b & a & 0 \end{bmatrix}$

we show that (1) holds for any X

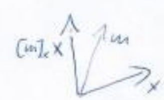
$$P'^T F P = \underset{4\times3}{\begin{bmatrix} F^T S^T \\ \hline e'^T \end{bmatrix}} F \underset{3\times4}{\left[\begin{array}{ccc|c} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{array}\right]} = \begin{bmatrix} F^T S^T \\ \hline e'^T \end{bmatrix} \left[F \,\middle|\, \begin{matrix} 0 \\ 0 \\ 0 \end{matrix}\right] = \left[\begin{array}{c|c} F^T S^T F & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline \underbrace{e'^T F}_{\text{left nullspace } 0} & 0 \end{array}\right]$$

$$= \left[\begin{array}{c|c} F^T S^T F & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline 0\;0\;0 & 0 \end{array}\right]$$

$\Rightarrow$ we have to show:

$$\underbrace{\underset{1\times4}{(x, y, z, 1)} \underset{4\times4}{\left[\begin{array}{c|c} F^T S^T F & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ \hline 0\;0\;0 & 0 \end{array}\right]}}_{[(x,y,z)F^T S^T F,\ 0]} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = 0 \Leftrightarrow (x, y, z)\, F^T S^T F \begin{pmatrix} x \\ y \\ z \end{pmatrix} = 0$$

This is true if $F^T S F = [m]_x$



$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} 0 & -c' & b' \\ c' & 0 & -a' \\ -b' & a' & 0 \end{bmatrix} \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix}$$

Derivation

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} -c'b+b'c & -c'e+b'f & -c'h+b'i \\ c'a-a'c & c'd-a'f & c'g-a'i \\ -b'a+a'b & -b'd+a'e & -b'g+a'h \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

① $-ac'b + ab'c + bc'a - ba'c - cb'a + ca'b = 0$

⑤ $-dc'e + db'f + ec'd - ea'f - fb'd + fa'e = 0$

$\cancel{⑨\; -gc'h + gb'i + ec'g - ea'i + f}$

⑨ $-gc'h + gb'i + hc'g - ha'i - b'gi + a'hi = 0$

② $-ac'e + ab'f + bc'd - ba'f - cb'd + ca'e =: m_1$

④ $-dc'b + db'c + ec'a - ea'c - fb'a + fa'b =: -m_1$

③ $-ac'h + ab'i + bc'g - ba'i - cb'g + ca'h =: m_2$

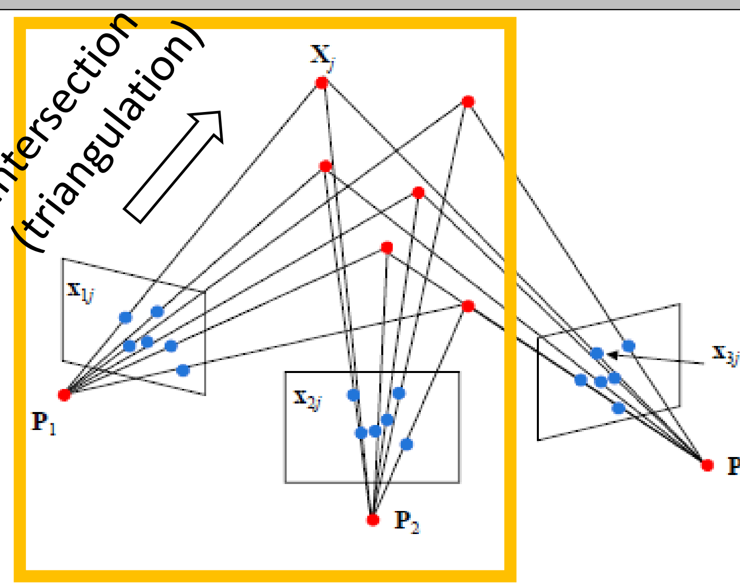
⑦ $-gc'b + gb'c + hc'a - ha'c - ib'a + ia'b =: -m_2$

⑥ $-dc'h + db'i + ec'g - ea'i - fb'g + fa'h =: m_3$

⑧ $-gc'e + gb'f + hc'd - ha'f - ib'd + ia'e =: -m_3$

Compute $X_{i's}$ (Triangulation)

- Input: x, x', P, P'
- Output: $X_{i's}$
- Triangulation is also called intersection
- Simple solution for algebraic error:



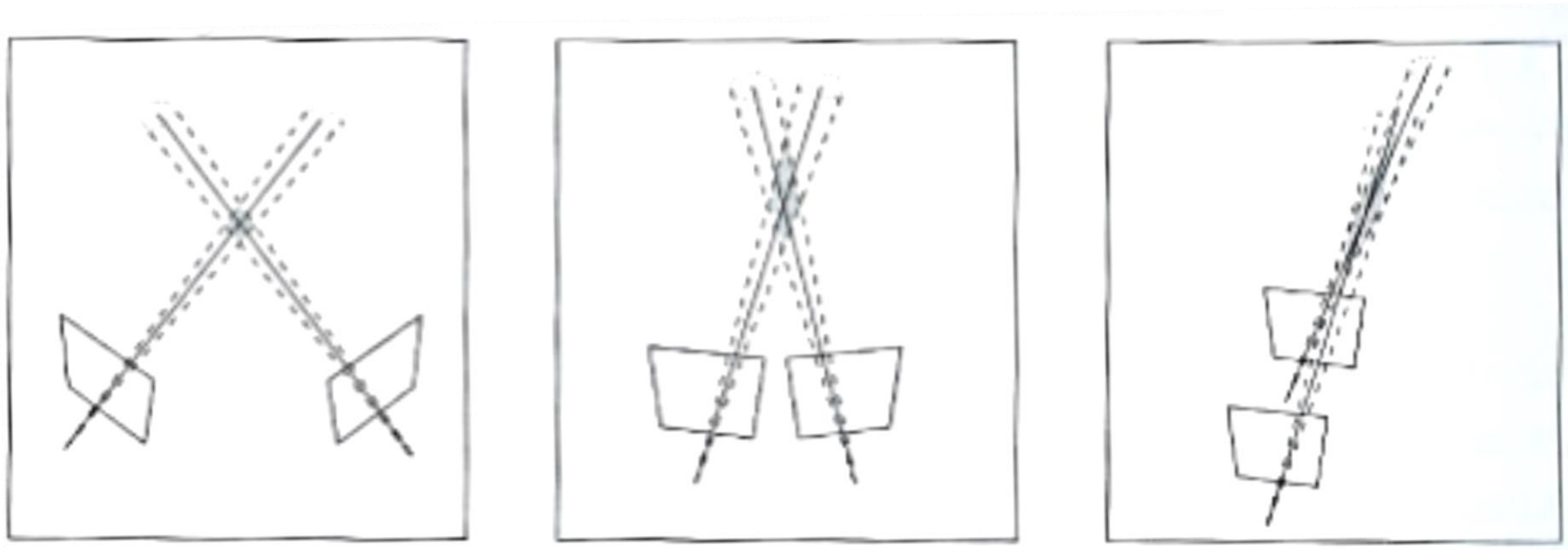
1) $\lambda x = P\ X$ and $\lambda' x' = P'\ X$
3x4 matrix

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