Robot Vision

TTK4255

Lecture 04 – Feature based Alignment and Pose Estimation

Annette Stahl

(Annette.Stahl@ntnu.no)

Department of Engineering Cybernetics – ITK

NTNU, Trondheim

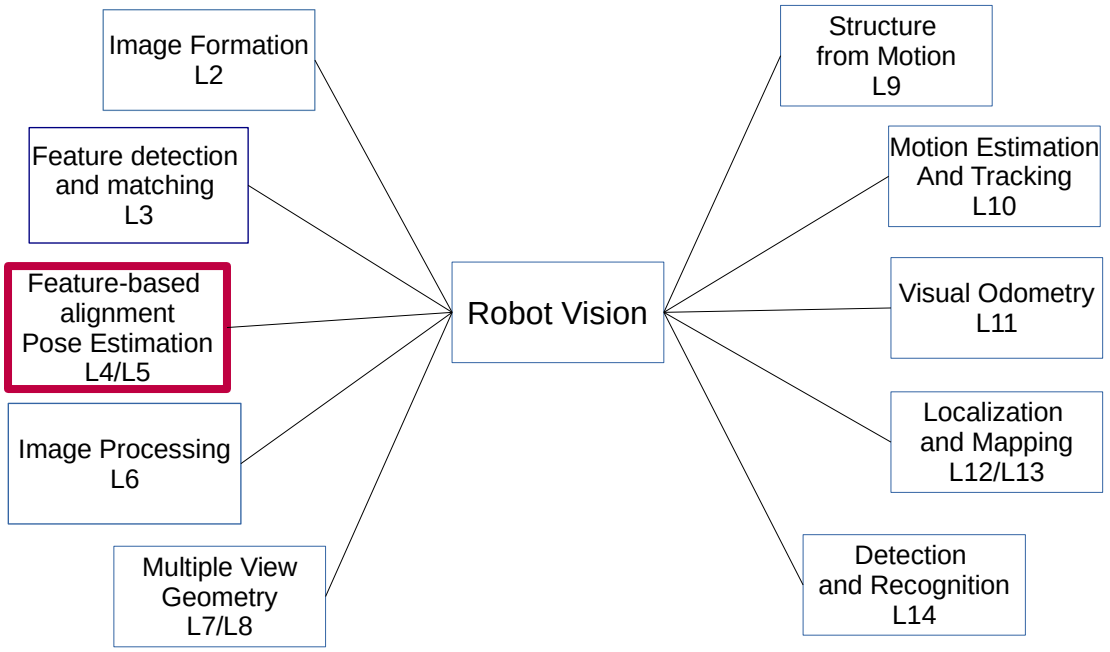
Spring Semester

27. January 2020

Lecture 04 – Feature based Alignment and Pose Estimation

Annette Stahl (Annette.Stahl@ntnu.no)

Simen Haugo (Simen.Haugo@ntnu.no)



Outline of the fourth lecture:

- Homography
- Direct Linear Transformation
- RANSAC
- Singular Value Decomposition
- Pose Estimation
- PnP Problem

Pose Estimation / Localization

Extract Local Features

Establish 2D-3D Matches

Camera Pose Estimation:
RANSAC + n-Point-Pose Algorithm

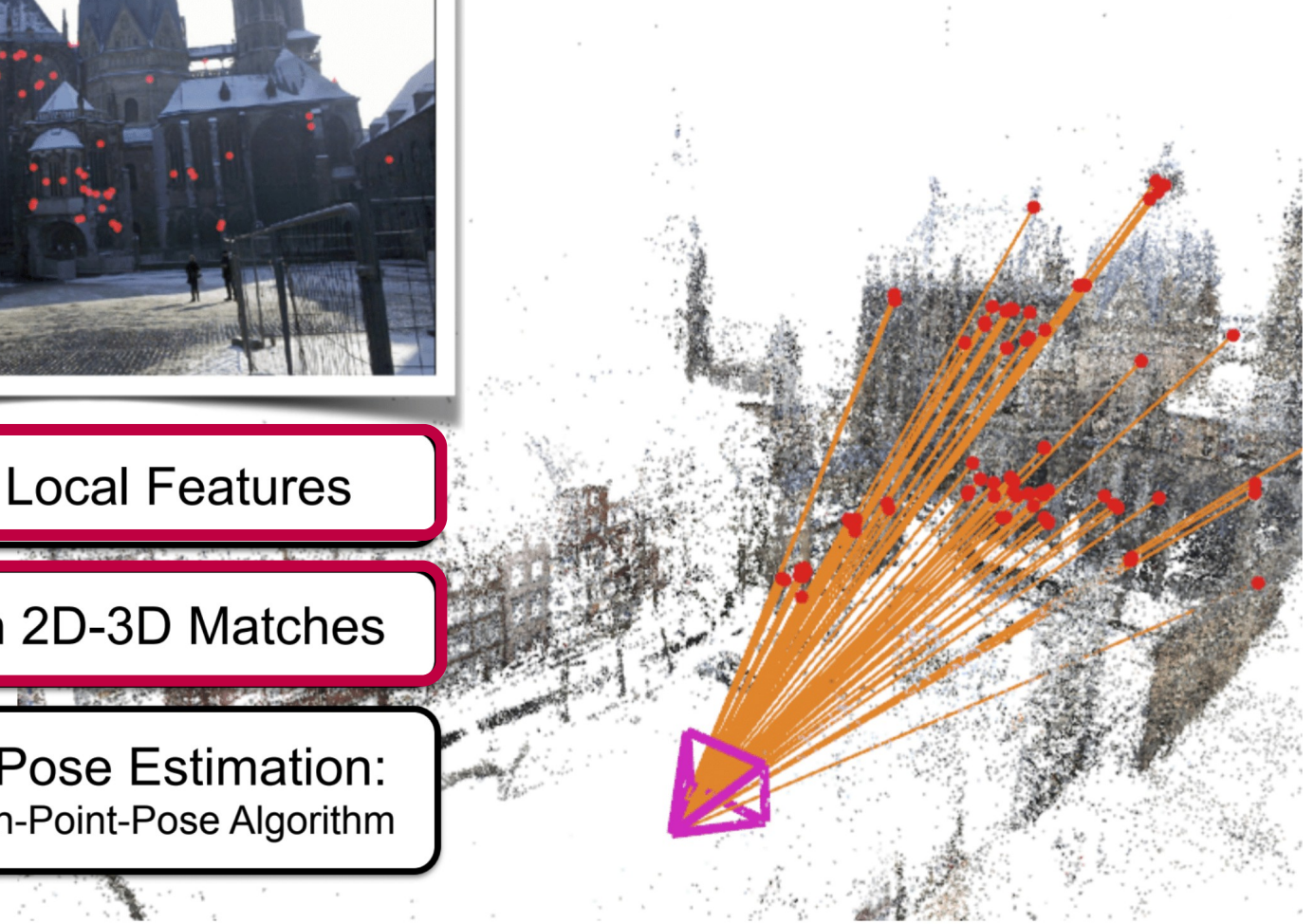


Image Courtesy: Torstein Sattler, CVPR 2015 Tutorial on Large-Scale Visual Place Recognition and Image-Based Localization

Recap L03 – Extract Local Features

Feature points should be

- distinctive (i.e., features, keypoints, salient points), and
- repeatable (i.e., can be re-detected from other views)

Feature descriptors should be invariant to:

– geometric changes: rotation, scale, view point

– photometric changes: illumination

Most feature descriptors are designed to be invariant to

- 2D translation,
- 2D rotation,
- Scale
- Some can handle
  - Small view-point invariance (SIFT, SURF, ORB, BRISK; SIFT works with up to 50 degrees of viewpoint changes!)
  - Affine illumination changes (SIFT, SURF, ORB, BRISK)

Recap and Addition to L03

Detector	Localization Accuracy of the detector	Descriptor that can be used	Efficiency	Relocalization & Loop closing
Harris	++++	Patch SIFT BRIEF ORB BRISK FREAK	+++ + ++++ ++++ +++ ++++	+ +++++ +++ ++++ +++ ++++
Shi-Tomasi	++++	Patch SIFT BRIEF ORB BRISK FREAK	++ + ++++ ++++ +++ ++++	+ +++++ +++ ++++ +++ ++++
FAST	++++	Patch SIFT BRIEF ORB BRISK FREAK	++++ + ++++ ++++ +++ ++++	+++ +++++ +++ ++++ +++ ++++
SIFT	+++	SIFT	+	++++
SURF	+++	SURF	++	++++

Courtesy: D Scaramuzza, UZH, ETH

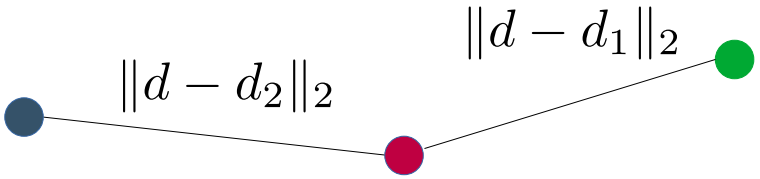
Recap L03 Feature Matching

Nearest-neighbor Search

- Not every nearest neighbor is correct
- Use ratio test to reject wrong / ambiguous matches

  [Lowe, IJCV'04]: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf
- Only accept match if

$$\frac{\|d - d_1\|_2}{\|d - d_2\|_2} < 0.8$$



In General:

- Typical datasets: 3-10k query features, >1M points
- Exhaustive (linear) nearest neighbor search is prohibitive
- Challenge of dimensionality: No exact search method that is faster than linear search

Nearest Neighbor Search

To extend the list of already listed methods from L03:

Multiple fast approximate nearest neighbor search methods:

- **KD-trees**

  [Muja & Lowe, PAMI'14] https://www.cs.ubc.ca/research/flann/uploads/FLANN/flann_pami2014.pdf

- **Hierarchical K-means trees**

  [Muja & Lowe, PAMI'14]

- **Product quantization**

  [Jégou et al., PAMI'11] https://hal.inria.fr/file/index/docid/825085/filename/jegou_pq_postprint.pdf

  [Kalantidis & Avrithis,CVPR'14] https://hal.inria.fr/file/index/docid/825085/filename/jegou_pq_postprint.pdf

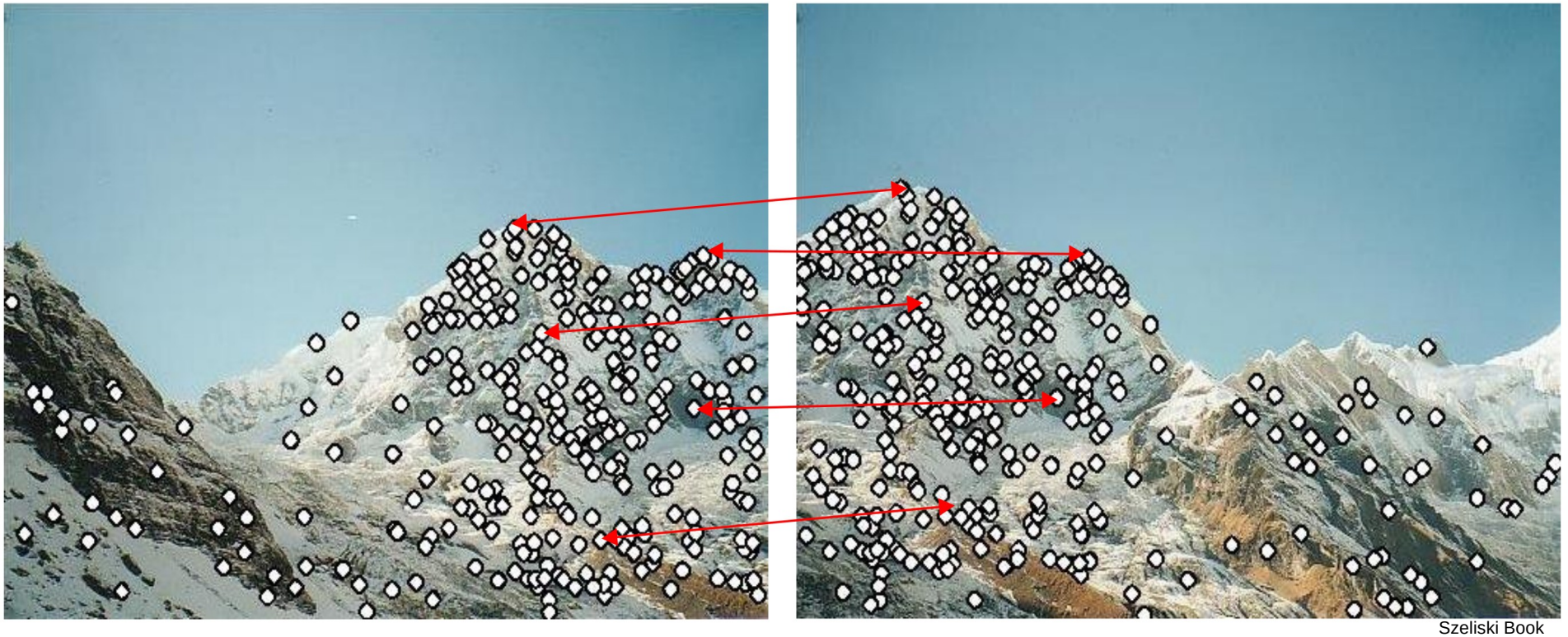
- **Inverted multi-index**

  [Babenko & Lempitsky, CVPR'12] http://arbabenko.github.io/MultiIndex/

  http://cache-default99i.cdn.yandex.net/download.yandex.ru/company/cvpr2012.pdf

- **Hashing techniques**

2D to 2D Point Correspondences



Szeliski Book

n 2D Point correspondences $\mathbf{x_i} \leftrightarrow \mathbf{x'_i}, i = 1, 2, ..., n$ are found.

→ How to perform **feature-based alignment** of the two images with respect to the **found point correspondences**?

Matching with Features

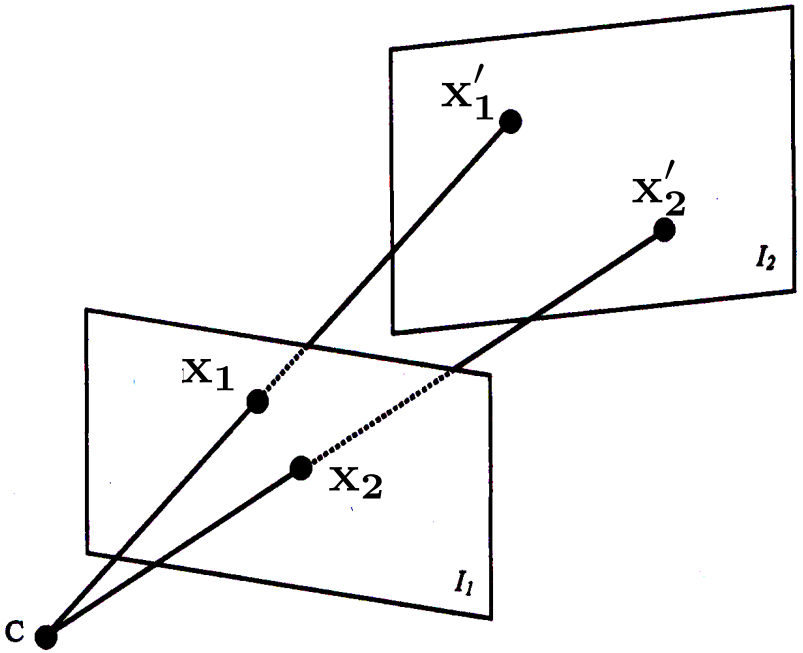
Szeliski Book

Recap L01: Homography in $\mathbb{P}^2$

Problem: Given a set of n points $\mathbf{x_i}$ in $\mathbb{P}^2$and a corresponding set of points $\mathbf{x_i'}$ likewise in $\mathbb{P}^2$

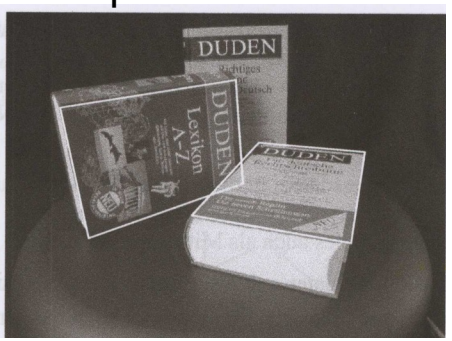
Compute the projective transformation that takes each $\mathbf{x_i}$ from one image into another image

$\mathbf{x_i'},\ i = 1, 2, ..., n$, where each image being considered as a projective plane $\mathbb{P}^2$



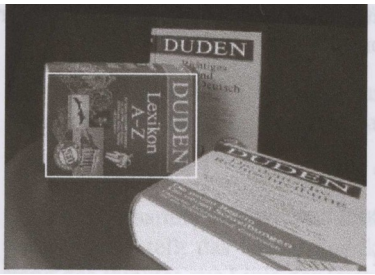
Courtesy: O. Schreer. Stereoanalyse und Bildsynthese. Springer, 2005

Example:



Original Image

Quadrilateral of one book is mapped to a rectangle (of pre-defined size)



First book

Second Book

Recap L01: Homography

Demo in L01

Recap L01 Projective Transformation

How to perform feature-based alignment of the two images with respect to the found point correspondences?

Estimate the 2D Homogrpahy.

2D Homography: A planar projective transformation is a linear transformation on homogeneous 3-vectors represented by a non-singular 3x3 matrix

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\mathbf{x}' = H\mathbf{x}$$

→ A homography is an invertible mapping h from $\mathbb{P}^2$ to itself such that three points $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}$ lie on the same line if and only if $h(\mathbf{x_1}), h(\mathbf{x_2}), h(\mathbf{x_3})$ do.

Homographies form a group since the inverse of a homography is also a homography and so the composition of two homographies.

Estimating the Homography

Determine H with the **Direct Linear Transformation (DLT)** algorithm :

Given is a set of four 2D to 2D point correspondences $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$

The transformation is given by the linear transformation

$$\mathbf{x}' = H\mathbf{x}$$

involving homogeneous vectors. → Note: the 3-vectors on both sides of the equation have the same direction but might differ in magnitude by a non-zero scale factor!

Using the vector cross product $\mathbf{x}' \times H\mathbf{x} = 0$ we compute for each point correspondence $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$:

$$\mathbf{x_i'} \times H\mathbf{x_i} = \begin{bmatrix} x_i' \\ y_i' \\ w_i' \end{bmatrix} \times \begin{bmatrix} h^{1\top}\mathbf{x_i} \\ h^{2\top}\mathbf{x_i} \\ h^{3\top}\mathbf{x_i} \end{bmatrix} = \begin{bmatrix} y_i' h^{3\top}\mathbf{x_i} - w_i' h^{2\top}\mathbf{x_i} \\ w_i' h^{1\top}\mathbf{x_i} - x_i' h^{3\top}\mathbf{x_i} \\ x_i' h^{2\top}\mathbf{x_i} - y_i' h^{1\top}\mathbf{x_i} \end{bmatrix} = 0$$

Where the j-th row of the matrix H is denoted by $h^{j\top}$ and $\mathbf{x_i}' = (x_i', y_i', w_i')^\top$

We can derive a linear solution for H.

Estimating the Homography

This gives a set of three equations in the entries of H

$$\begin{bmatrix} y_i' h^{3\top}\mathbf{x_i} - w_i' h^{2\top}\mathbf{x_i} \\ w_i' h^{1\top}\mathbf{x_i} - x_i' h^{3\top}\mathbf{x_i} \\ x_i' h^{2\top}\mathbf{x_i} - y_i' h^{1\top}\mathbf{x_i} \end{bmatrix} = \begin{bmatrix} y_i'\mathbf{x_i}^\top h^3 - w_i'\mathbf{x_i}^\top h^2 \\ w_i'\mathbf{x_i}^\top h^1 - x_i'\mathbf{x_i}^\top h^3 \\ x_i'\mathbf{x_i}^\top h^2 - y_i'\mathbf{x_i}^\top h^1 \end{bmatrix}$$

$$= \begin{bmatrix} 0^\top & -w_i'\mathbf{x_i}^\top & y_i'\mathbf{x_i}^\top \\ w_i'\mathbf{x_i}^\top & 0^\top & -x_i'\mathbf{x_i}^\top \\ -y_i'\mathbf{x_i}^\top & x_i'\mathbf{x_i}^\top & 0^\top \end{bmatrix} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = 0$$

Estimating the Homography

$$\underbrace{\begin{bmatrix} 0^\top & -w_i'\mathbf{x_i}^\top & y_i'\mathbf{x_i}^\top \\ w_i'\mathbf{x_i}^\top & 0^\top & -x_i'\mathbf{x_i}^\top \\ -y_i'\mathbf{x_i}^\top & x_i'\mathbf{x_i}^\top & 0^\top \end{bmatrix}}_{A_i} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = 0$$

These equations are of the form $A_i h = \mathbf{0}$

Where A_i is a 3 x9 matrix and h is a 9-vector made up of the entries of the matrix H

$$h = \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix}, \qquad H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

is an equation linear in the unknown h.

Estimating the Homography

We can rewrite

$$\underbrace{\begin{bmatrix} 0^\top & -w_i'\mathbf{x_i}^\top & y_i'\mathbf{x_i}^\top \\ w_i'\mathbf{x_i}^\top & 0^\top & -x_i'\mathbf{x_i}^\top \\ -y_i'\mathbf{x_i}^\top & x_i'\mathbf{x_i}^\top & 0^\top \end{bmatrix}}_{A_i} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = 0$$

as

$$\begin{bmatrix} 0 & 0 & 0 & -w_1'x_1 & -w_1'y_1 & -w_1'w_1 & y_1'x_1 & y_1'y_1 & y_1'w_1 \\ w_1'x_1 & w_1'y_1 & w_1'w_1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1'w_1 \\ -y_1'x_1 & -y_1'y_1 & -y_1'w_1 & x_1'x_1 & x_1'y_1 & x_1'w_1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0$$

Note: **three** equations where only **two** are **linearly independent.**

(third row of A is a linear combination of the first and second row and obtained, up to scale from the sum of $-x_i'$ times the first row and $-y_i'$ times the second row)

Estimating the Homography

Thus each point correspondence gives two entities of H.

$$\begin{bmatrix} 0 & 0 & 0 & -w_1'x_1 & -w_1'y_1 & -w_1'w_1 & y_1'x_1 & y_1'y_1 & y_1'w_1 \\ w_1'x_1 & w_1'y_1 & w_1'w_1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1'w_1 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0$$

Meaning: The third equation is omitted in solving for H:

$$\underbrace{\begin{bmatrix} 0^\top & -w_i'\mathbf{x_i}^\top & y_i'\mathbf{x_i}^\top \\ w_i'\mathbf{x_i}^\top & 0^\top & -x_i'\mathbf{x_i}^\top \end{bmatrix}}_{A_i} \begin{bmatrix} h^1 \\ h^2 \\ h^3 \end{bmatrix} = 0$$

where A_i is now the 2x9 matrix

$$A_i h = 0$$

Estimating the Homography

→ Each point correspondence leads to two independent equations

Given a set of n such point correspondences we obtain

$$Ah = 0$$

where $A = [A_1 \ \cdots \ A_n]^\top$ is a $2n$x9 matrix.

Recall H (and thus h) is homogeneous, we have 8DoF and thus need the matrix A to have rank 8 in order to determine h up to scale.

→ 4 point correspondences are sufficient (3 points are non-collinear!)

A solution to the equation $Ah = 0$ can be computed by utilizing the **Singular Value Decomposition (SVD)** of A:

$$A = USV^\top$$

Singular Value Decomposition - SVD

- SVD is one of the most useful matrix decompositions.
- Most common application is in the solution of over-determinded systems
- SVD is a factorization of a mxn real matrix A,

  where U and V are orthogonal matrices and D is a diagonal matrix with non-negative entries.

  The diagonal entries of S are the singular values of A and the columns of U and V are the left and right singular vectors of A respectively.

  The nullspace of A is the span of the right singular vectors $\mathbf{v_i}$ that corresponds to a zero singular value $\mathbf{s_i}$ (or does not have a corresponding singular value)

Example $A\mathbf{x} = \mathbf{0}$:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$U = \begin{bmatrix} -0.3863 & -0.9224 \\ -0.9224 & 0.3863 \end{bmatrix} \quad S = \begin{bmatrix} 9.5080 & 0 & 0 \\ 0 & 0.7729 & 0 \end{bmatrix} \quad V = \begin{bmatrix} -0.4287 & 0.8060 & 0.4082 \\ -0.5663 & 0.1124 & -0.8165 \\ -0.7039 & -0.5812 & 0.4082 \end{bmatrix}$$

Singular Value Decomposition - SVD

Matlab

[U,S,V] = svd(A);

Right singular vectors are columns in V

OpenCV

cv::SVD::compute(A, S, U, Vtranspose, cv::SVD::FULL_UV);

Right singular vectors are rows in Vtranspose

Direct Linear Transformation

The resulting algorithm is the **Direct Linear Transformation (DLT):**

Given are $n \geq 4$ 2D to 2D point correspondences $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}, i = 1, 2, ..., n$, determine the 2D homograpy matrix H such that $\mathbf{x_i}' = H\mathbf{x_i}$

1. For each correspondence $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}$ compute the matrix A_i

2.Assemble the n 2x9 matrices A_i into a single 2nx9 matrix A

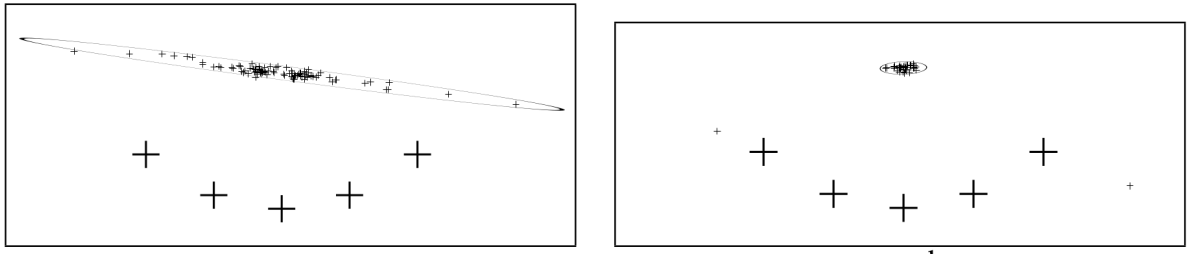
$$\mathrm{A} = \begin{bmatrix} 0 & 0 & 0 & -w_1'x_1 & -w_1'y_1 & -w_1'w_1 & y_1'x_1 & y_1'y_1 & y_1'w_1 \\ w_1'x_1 & w_1'y_1 & w_1'w_1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1'w_1 \\ 0 & 0 & 0 & -w_2'x_2 & -w_2'y_2 & -w_2'w_2 & y_2'x_2 & y_2'y_2 & y_2'w_2 \\ w_2'x_2 & w_2'y_2 & w_2'w_2 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2'w_2 \\ \vdots & & & \vdots & & & \vdots & & \\ 0 & 0 & 0 & -w_n'x_n & -w_n'y_n & -w_n'w_n & y_n'x_n & y_n'y_n & y_n'w_n \\ w_n'x_n & w_n'y_n & w_n'w_n & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n'w_n \end{bmatrix}$$

3. Obtain the SVD of A: $A = USV^\top$

4. Finally, h is then the last column of V and H can be determined from h.

Normalized DLT for 2D Homographies

Data normalization becomes important for less well conditioned problems such as DLT computation of the Fundamental Matrix (L07).



Hartley: Multiple View Geometry, p109

This leads us to the **normalized Direct Linear Transformation (DLT):**

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the 2D homography matrix $\mathtt{H}$ such that $\mathbf{x}'_i = \mathtt{H}\mathbf{x}_i$.

Algorithm

(i) **Normalization of $\mathbf{x}$:** Compute a similarity transformation $\mathtt{T}$, consisting of a translation and scaling, that takes points $\mathbf{x}_i$ to a new set of points $\tilde{\mathbf{x}}_i$ such that the centroid of the points $\tilde{\mathbf{x}}_i$ is the coordinate origin $(0,0)^\mathsf{T}$, and their average distance from the origin is $\sqrt{2}$.

(ii) **Normalization of $\mathbf{x}'$:** Compute a similar transformation $\mathtt{T}'$ for the points in the second image, transforming points $\mathbf{x}'_i$ to $\tilde{\mathbf{x}}'_i$.

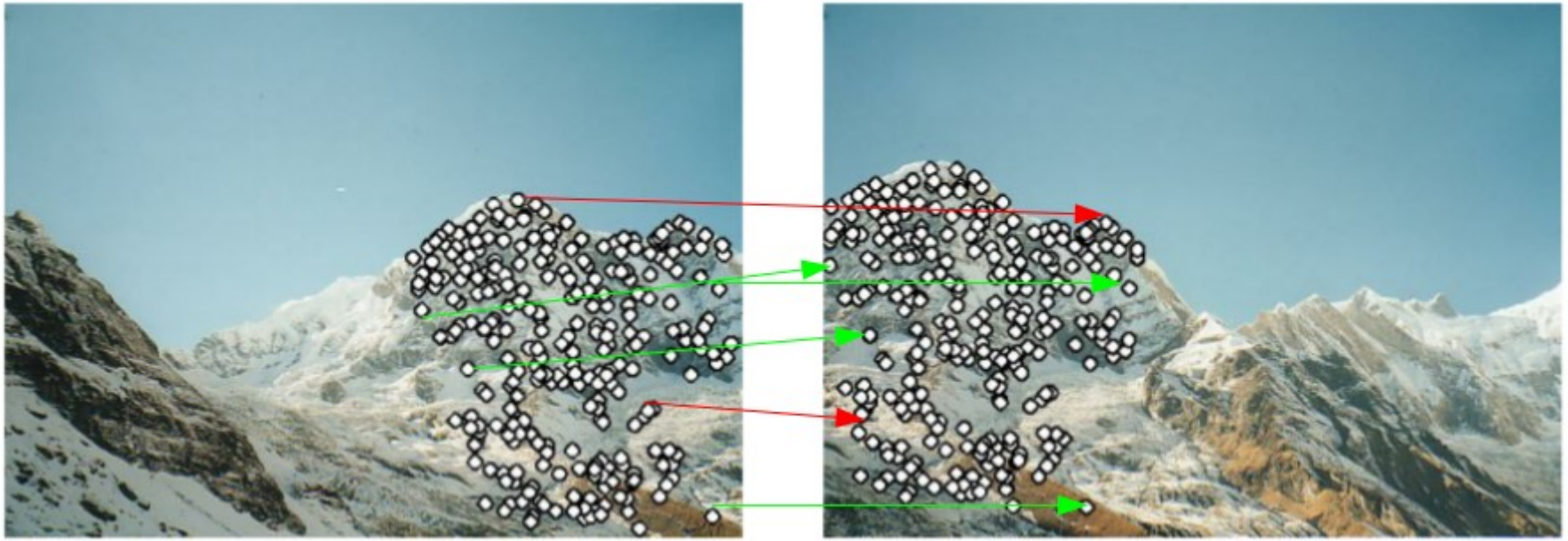
(iii) **DLT:** Apply algorithm 4.1(p91) to the correspondences $\tilde{\mathbf{x}}_i \leftrightarrow \tilde{\mathbf{x}}'_i$ to obtain a homography $\widetilde{\mathtt{H}}$.

(iv) **Denormalization:** Set $\mathtt{H} = \mathtt{T}'^{-1}\widetilde{\mathtt{H}}\mathtt{T}$.

Robust Estimation of the Homography

→ We need 4 corresponding point-pairs to be able to estimate H

(arbitrary 4 points will likely fail. RANSAC helps to remove wrong matching candidates.)

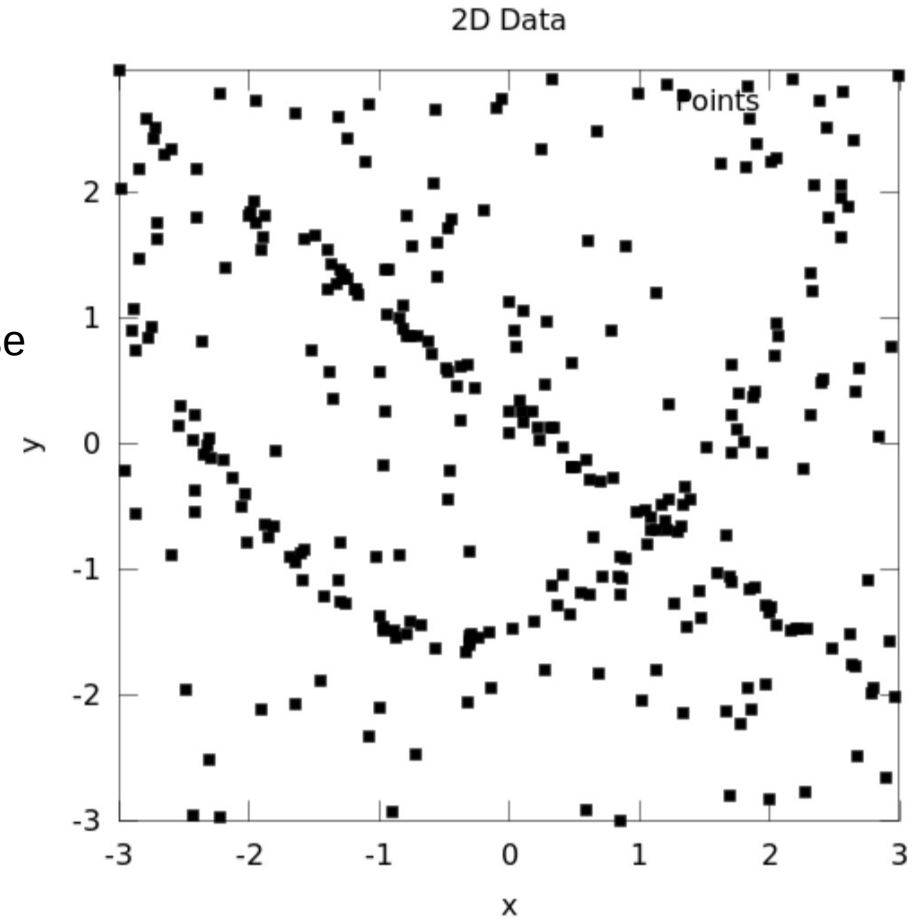


RRANdom SAmple Consensus (RANSAC)

- Many Robotic Vision problems require the estimation of model parameters, like Homographies, Fundamental Matrix, etc.
- We have to deal with many outliers

Basic Idea

- Measurements with many outliers
- Here we have mixed data from 2 models + noise
- Which do you see?



M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM , 24(6):381–395, 1981.

RRANdom SAmple Consensus (RANSAC)

- Linear model might be present

  $y = mx + b$

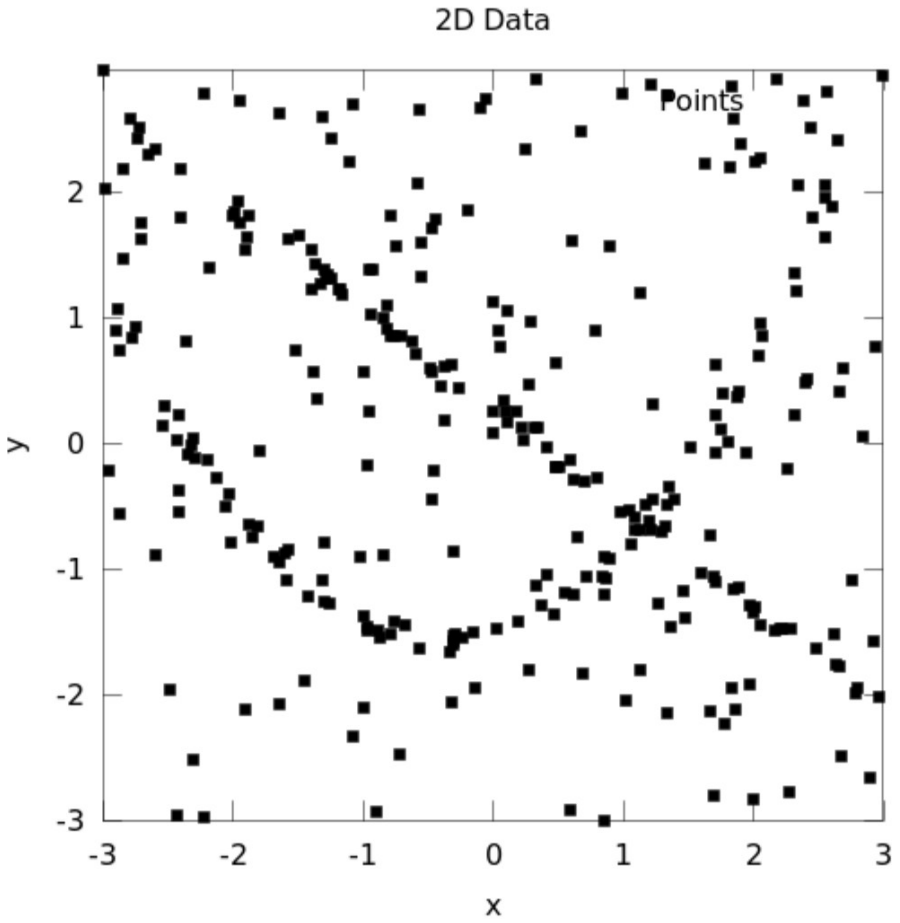
  (2 parameters)

- Quadratic model might be present

  $y = ax^2 + bx + c$

  (3 parameters)

- You need to pick one model!

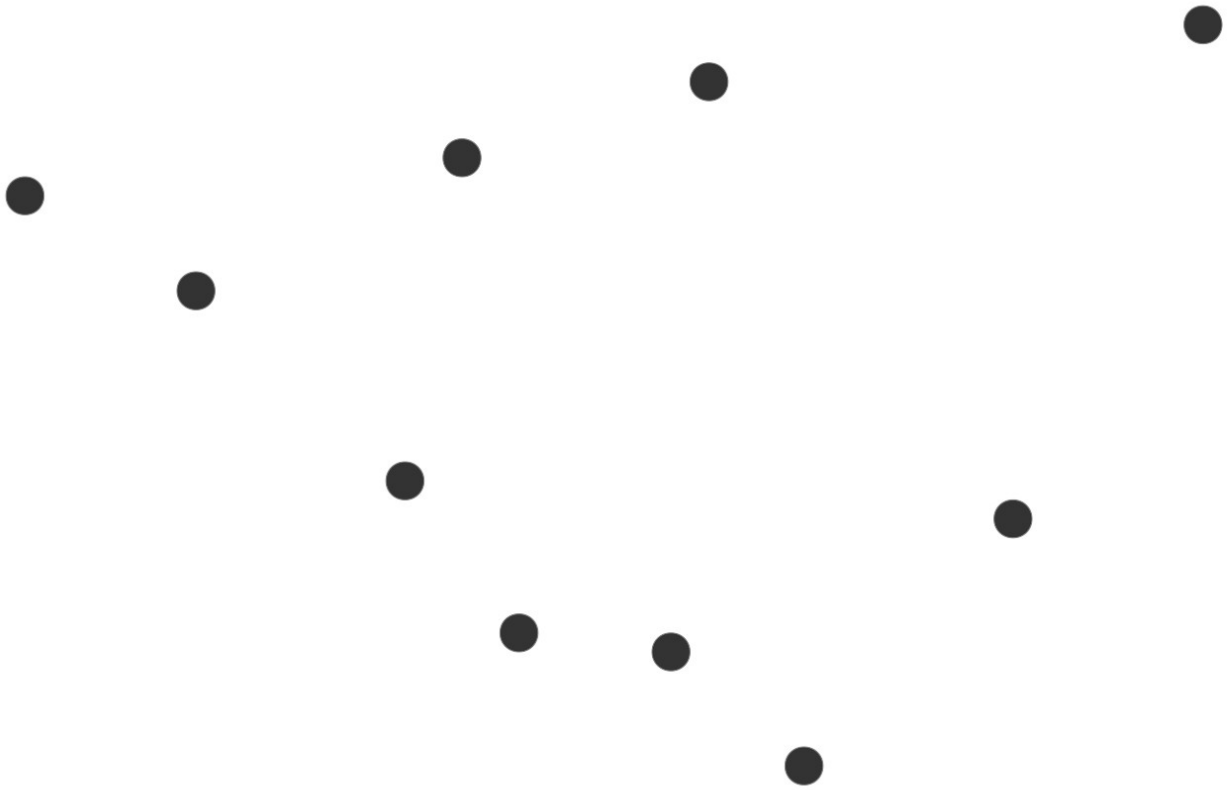


M.A. Fischler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM , 24(6):381–395, 1981.

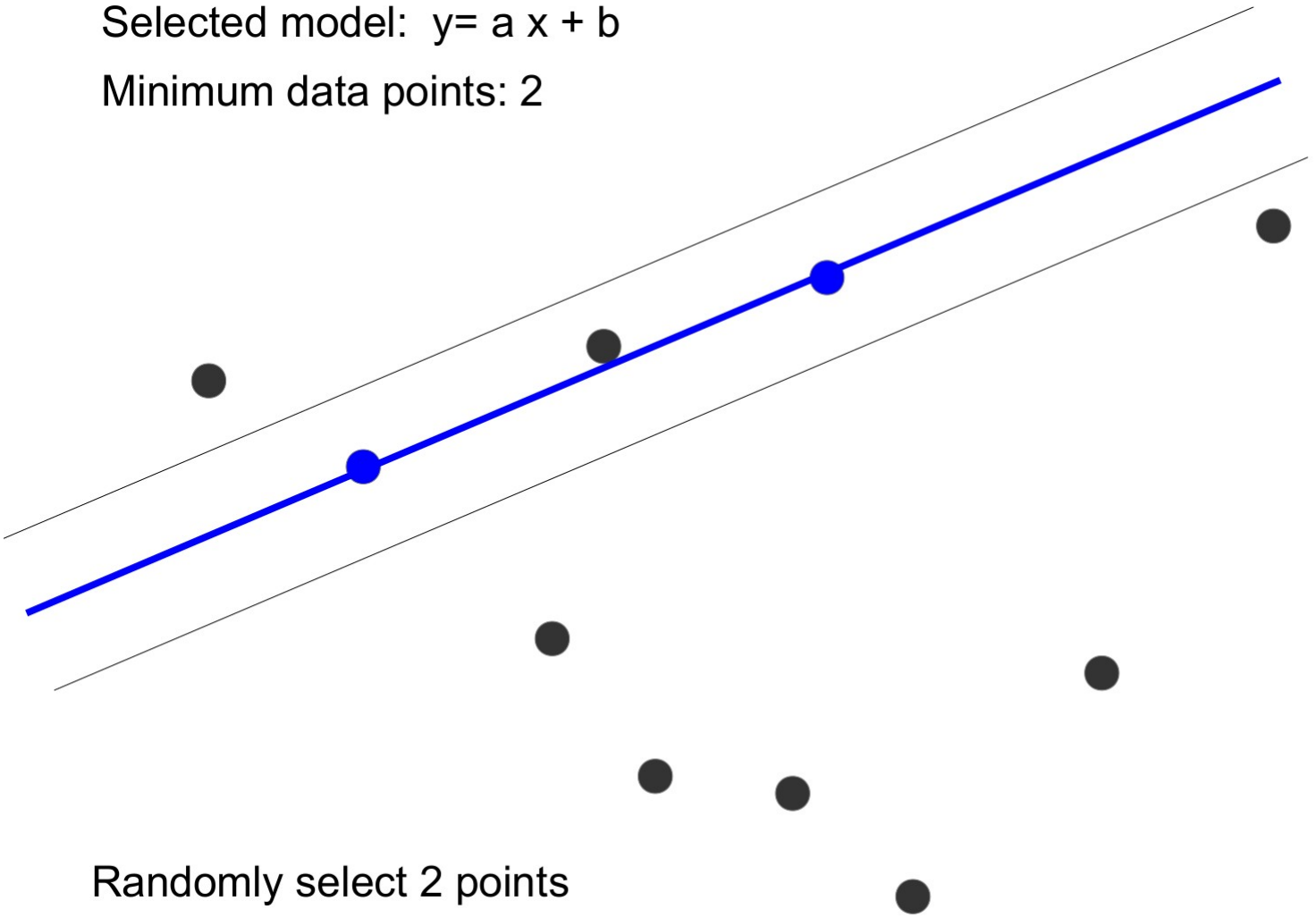
RANSAC: Line Estimation

Selected model: $y = a x + b$

Minimum data points: 2



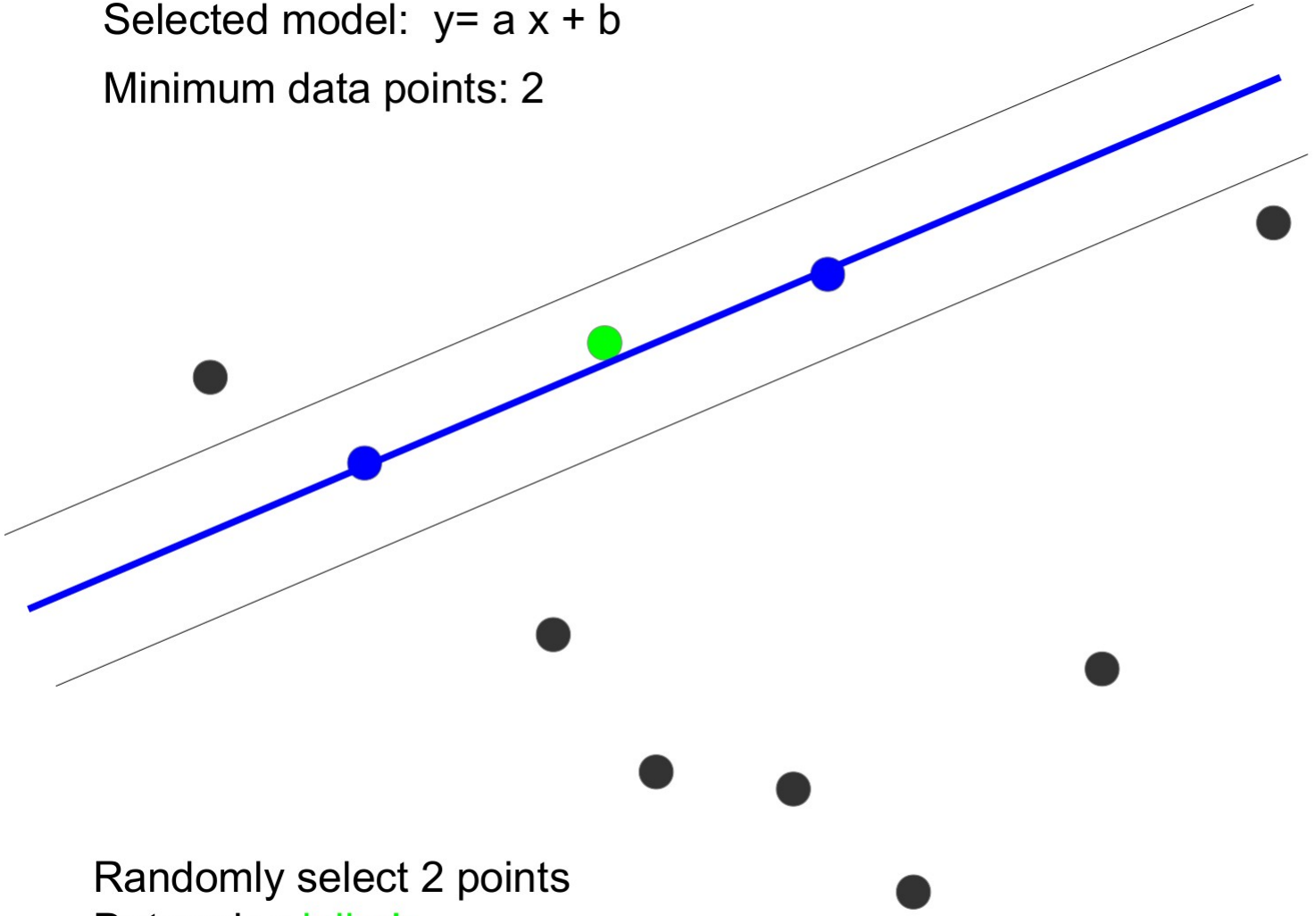
RANSAC: Line Estimation



RANSAC: Line Estimation

Selected model: y= a x + b

Minimum data points: 2

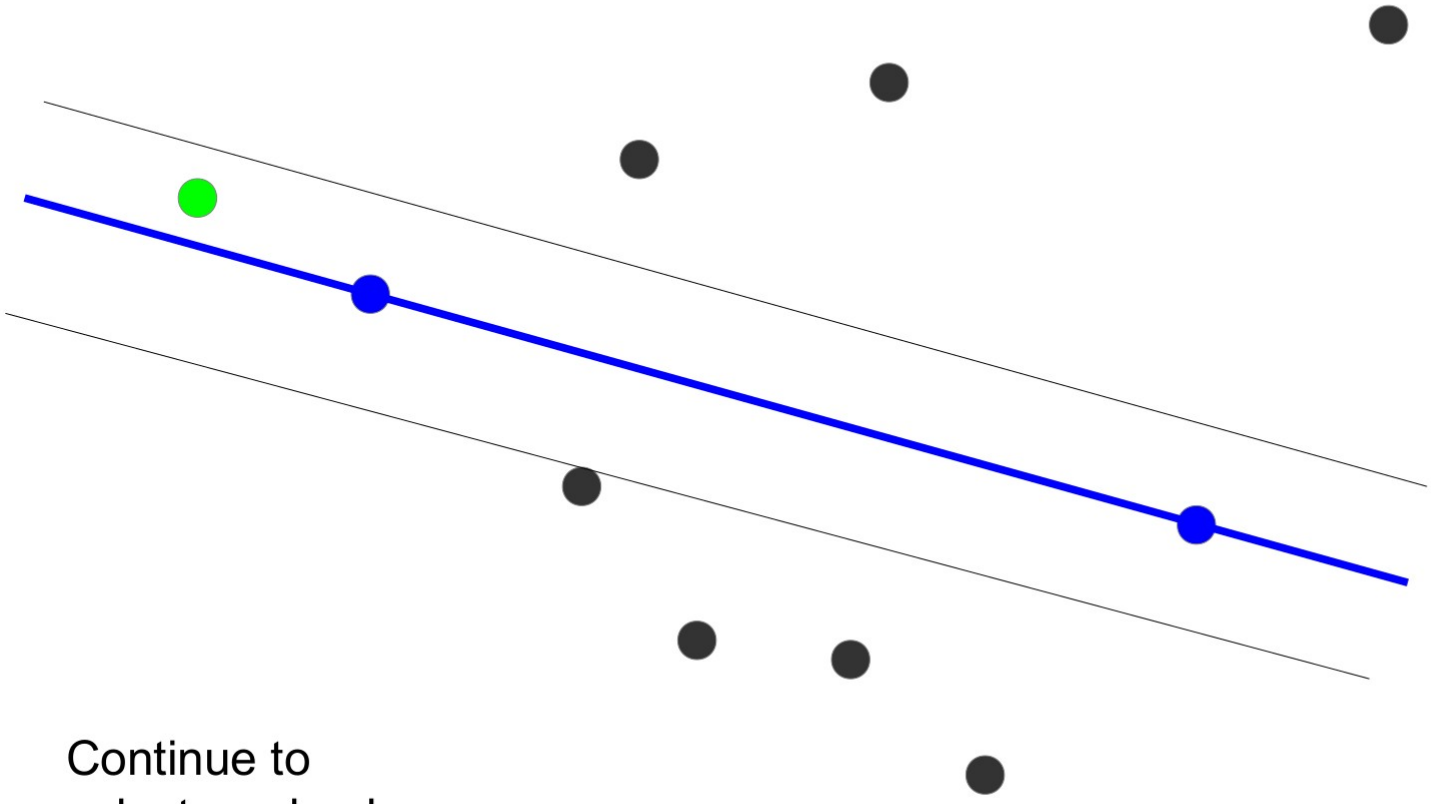


Randomly select 2 points
Determine inlier's

RANSAC: Line Estimation

Selected model: y= a x + b

Minimum data points: 2

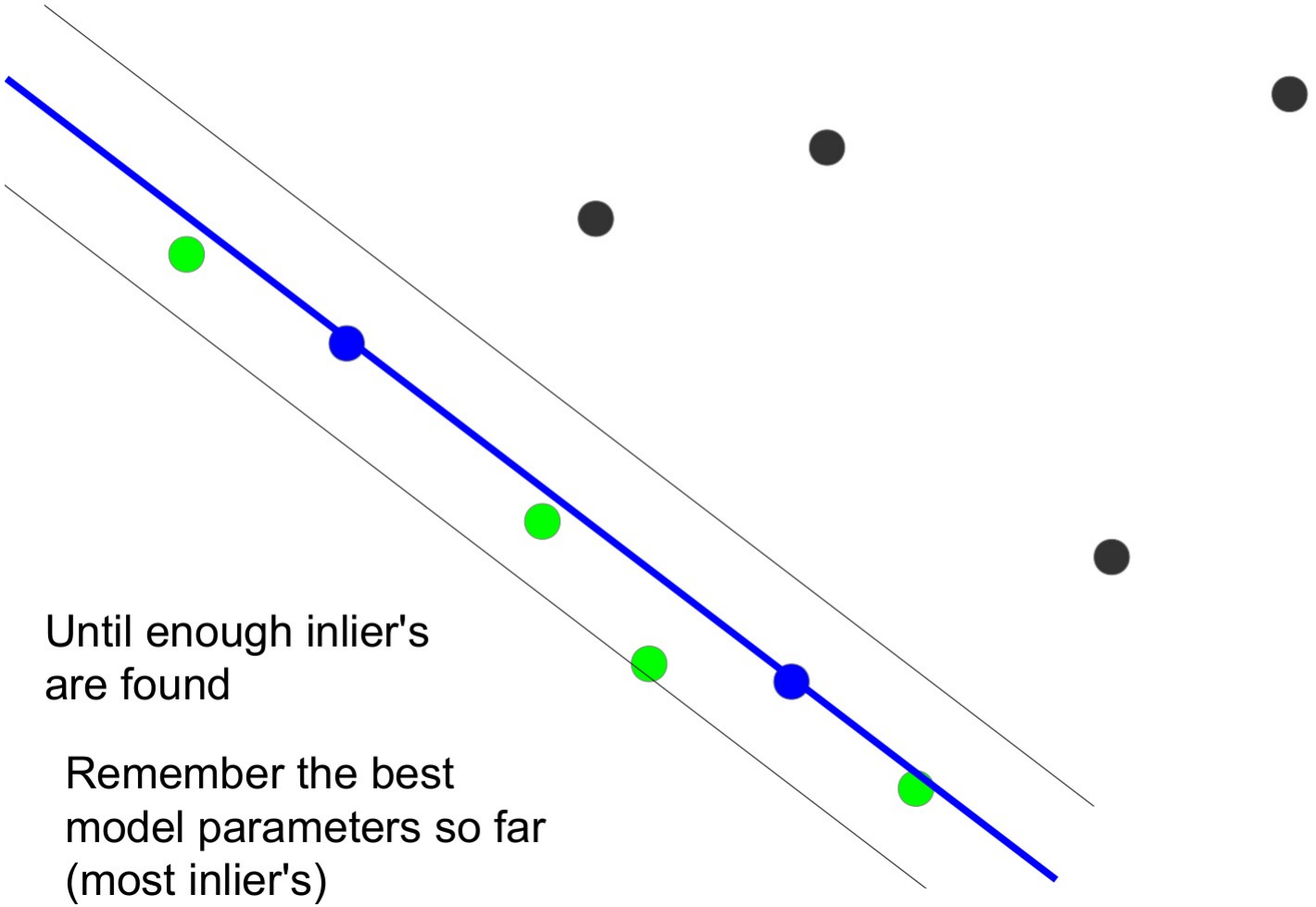


Continue to
select randomly
two points ...

RANSAC: Line Estimation

Selected model: y= a x + b

Minimum data points: 2



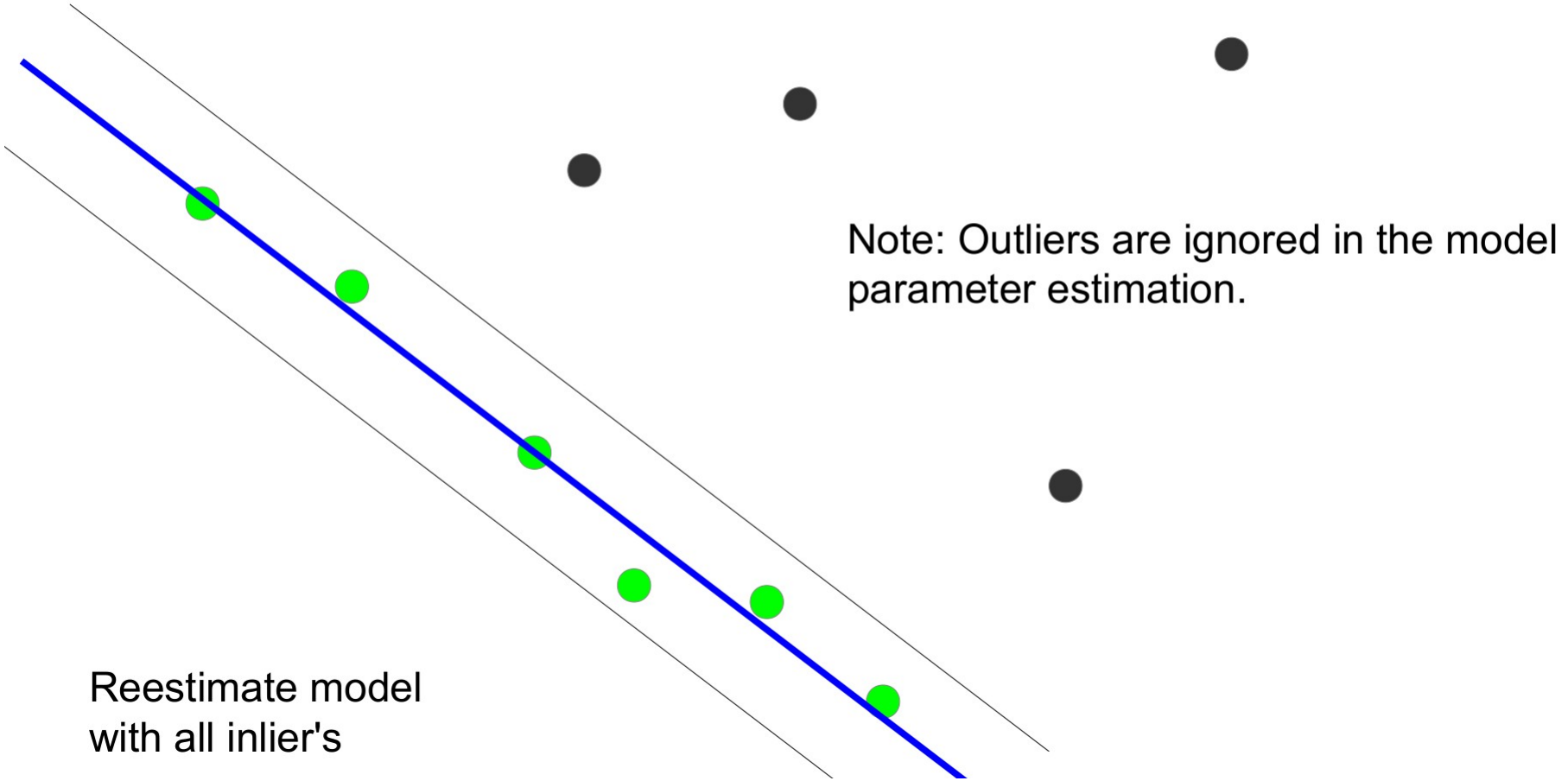
Until enough inlier's are found

Remember the best model parameters so far (most inlier's)

RANSAC: Line Estimation

Selected model: y= a x + b

Minimum data points: 2



Note: Outliers are ignored in the model parameter estimation.

Reestimate model with all inlier's

RANSAC Algorithm

1. Randomly choose minimum number of points required to determine model parameters.

2. Solve for the parameters of the model.

3. Compute how many of all points agree to this parameters with a predefined tolerance

4. If the fraction of the number of inlier's over the total number points in the set exceeds a predefined threshold , re-estimate the model parameters using all the identified inlier's and terminate.

5. Otherwise, repeat steps 1 through 4 (maximum of N times)

Note
+ RANSAC can handle large amount of outliers (50%)
- Computes parameters for one model
+ Robust and accurate estimations

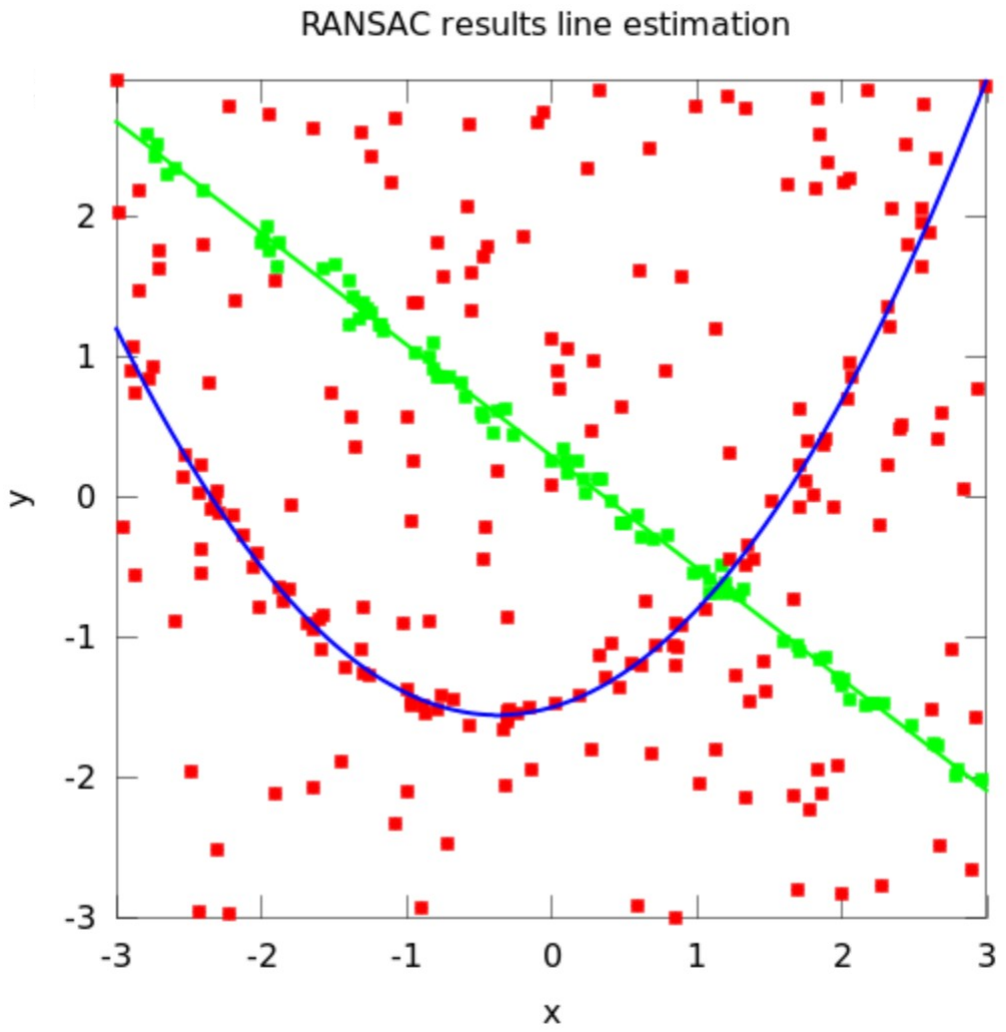
For a point correspondence (and homography H) we can choose from several **errors**

- **Algebraic error** $\varepsilon_i = \|A_i h\|$
- **Geometric errors (Reprojection error)**

$$\varepsilon_i = d(H\mathbf{x_i}, \mathbf{x_i'}) + d(\mathbf{x_i}, H^{-1}\mathbf{x_i'})$$

Found Estimates

For the quadratic model you randomly select 3 points to estimate the parameters!



How many Samples do we need?

How often to draw random samples of s points?

- w : probability that single point is inlier.

  → $e=1\text{-}w$ *:* probability to be an outlier

- $p=0.99$: probability that one sample is outlier-free

$$N = \frac{\log(1-p)}{\log(1(1-e)^s)}$$

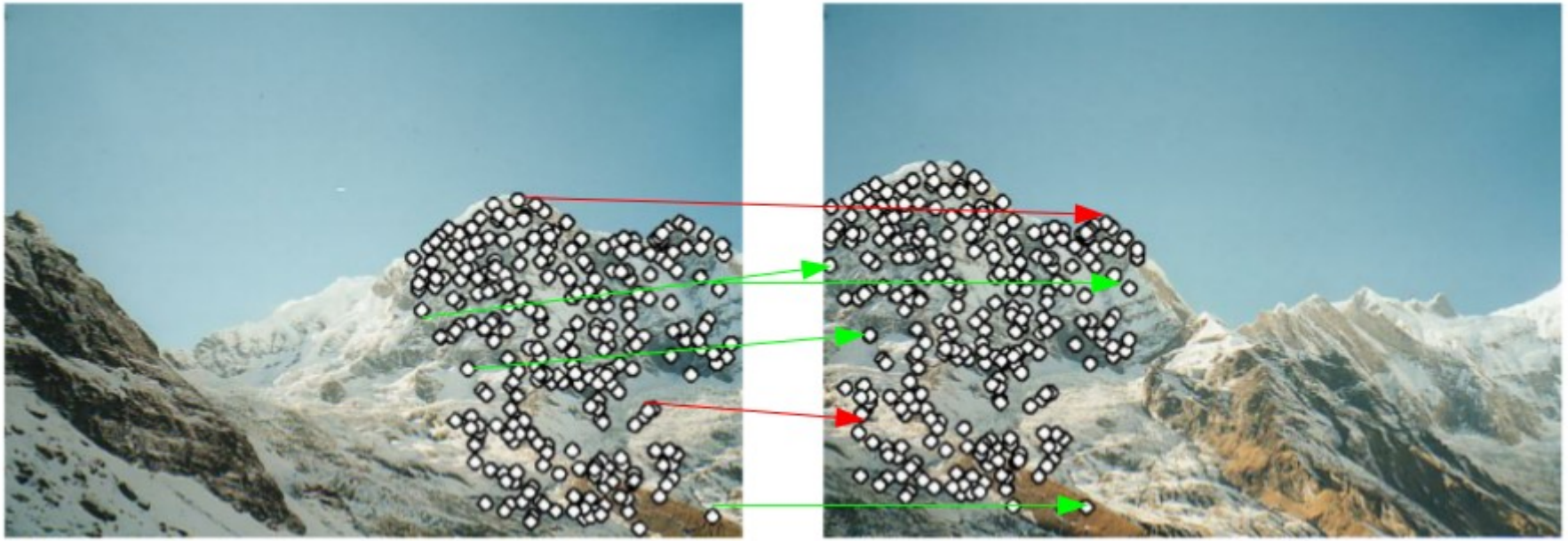
Note: As e is most often unknown, N can be computed adaptively

(Multiple View Geometry, Hartley,Zisserman, p. 105)

Robust Estimation of the Homography

→ We need 4 corresponding point-pairs to be able to estimate H

(Arbitrary 4 points will likely fail. RANSAC helps to remove wrong matching candidates.)



Task is to automatically find 4 correct mappings ...

Estimate Homography

- Compute interest points
- Compute initial correspondences (use any heuristics)

  - RANSAC (iterate): select randomly 4 point-pairs
    - Compute homography H
    - Compute distance d for each point-pair candidate $\varepsilon_i = d(H\mathbf{x_i}, \mathbf{x'_i}) + d(\mathbf{x_i}, H^{-1}\mathbf{x'_i})$
    - Count inlier's; repeat RANSAC steps

- Re-estimate H based on inliers:
  - Normalized DLT
  - Minimize $\varepsilon = \sum \varepsilon_i$ in an additional iterative optimization step (i.e. method like Levenberg Marquardt (→ L05))

  Remark: Essentially the same method enables the automatic computation of the “Fundamental” matrix (→ 3D reconstruction L07)

Example

- Estimate homography $\mathbf{x}' = H\mathbf{x}$

  OpenCV

```
#include "opencv2/calib3d.hpp"
cv::findHomography(srcPoints, dstPoints, CV_RANSAC);
```

  Matlab

```
tform = estimateGeometricTransform(srcPoints,dstPoints,'projective');
```

- Represent the images in common coordinates (Note the additional translation)

  OpenCV

```
#include "opencv2/calib3d.hpp"
cv::warpPerspective(img1, img2, H, output_size);
```

  Matlab

```
img2 = imwarp(img1,tform);
```

- Merge Images
- Blending + Histogram Equalization

Example Image Stitching

Find a Transformation for known point correspondences.

left image

Corresponding Points:

A: 431 210	B: 174 250
A: 269 315	B: 12 357
A: 301 345	B: 45 387
A: 487 230	B: 226 270
A: 336 229	B: 82 268
A: 318 274	B: 63 314
A: 391 320	B: 136 358
A: 293 262	B: 35 302
A: 268 387	B: 4 433
A: 457 308	B: 196 345

right image

Mosaic

left image

$$H = \begin{pmatrix} 1.02e+00 & 3.49e-02 & 2.45e+02 \\ 1.55e-02 & 9.87e-01 & -3.76e+01 \\ 1.90e-19 & 2.17e-19 & 1.00e+00 \end{pmatrix}$$

right image

3D to 2D Correspondence

	Structure (scene geometry)	Motion (camera geometry)	Measurements
Pose estimation	calibrated Camera	uncalibrated camera	3D to 2D correspondences
Triangulation, Stereo	uncalibrated camera	calibrated camera	3D to 2D correspondences
Reconstruction, Structure from Motion	uncalibrated camera	uncalibrated camera	3D to 2D correspondences

Pose Estimation / Localization

Extract Local Features

Establish 2D-3D Matches

Camera Pose Estimation:
RANSAC + n-Point-Pose Algorithm

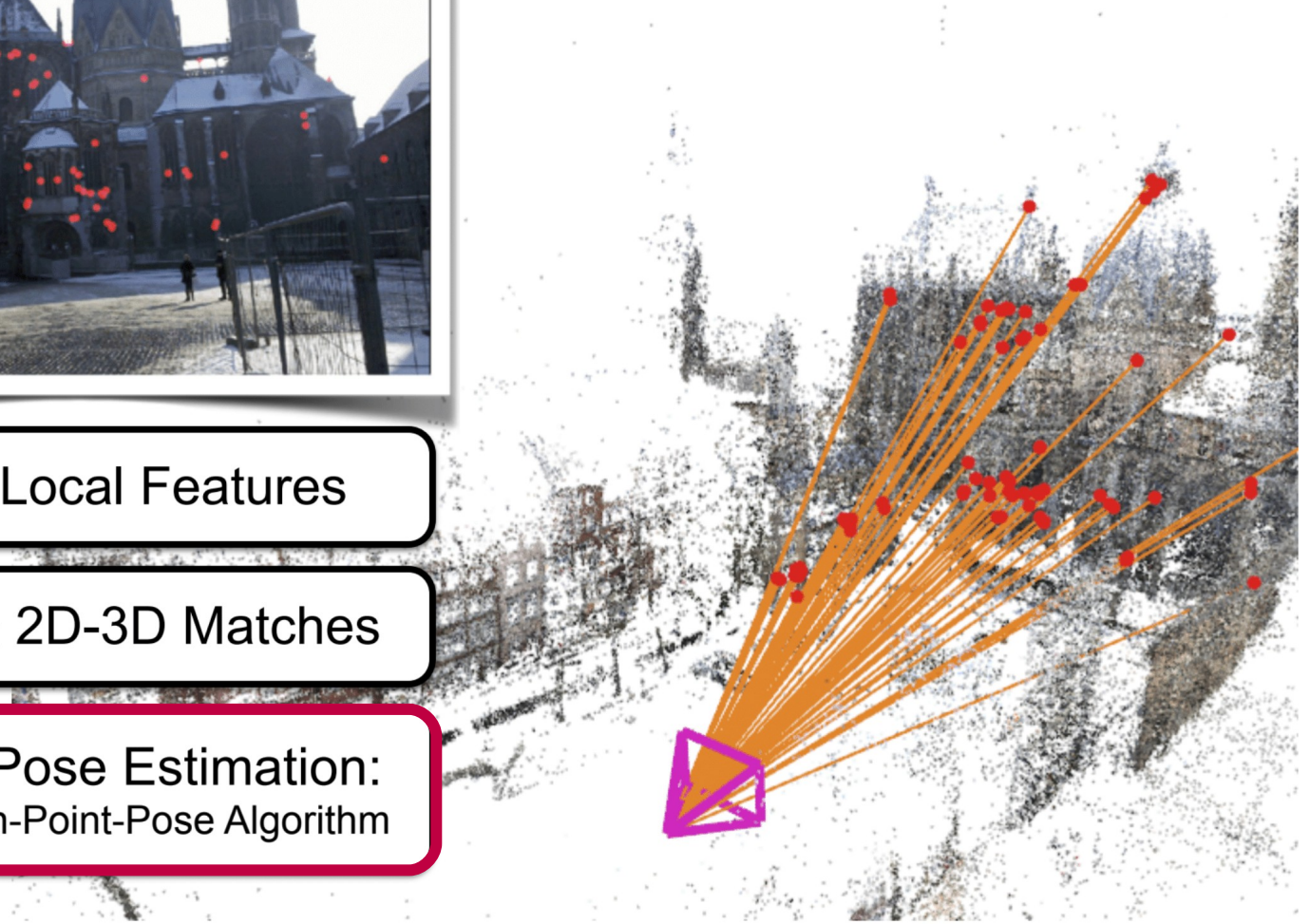


Image Courtesy: Torstein Sattler, CVPR 2015 Tutorial on Large-Scale Visual Place Recognition and Image-Based Localization

Pose Estimation

- Given known 3D landmark positions in the world frame and
- Given their image correspondences in the camera frame,
- Estimate the 6Dof 3D pose of the camera in the world frame (including intrinsic parameters if uncalibrated).

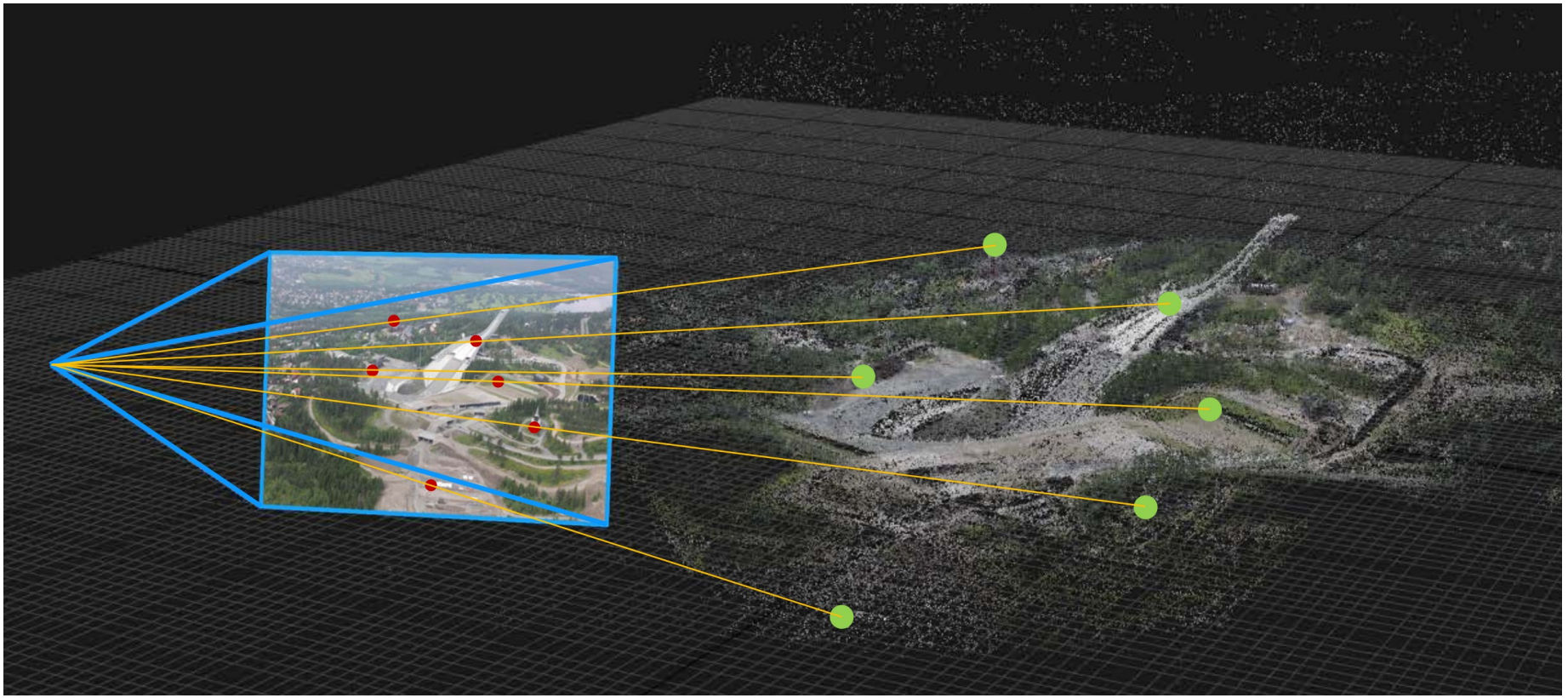
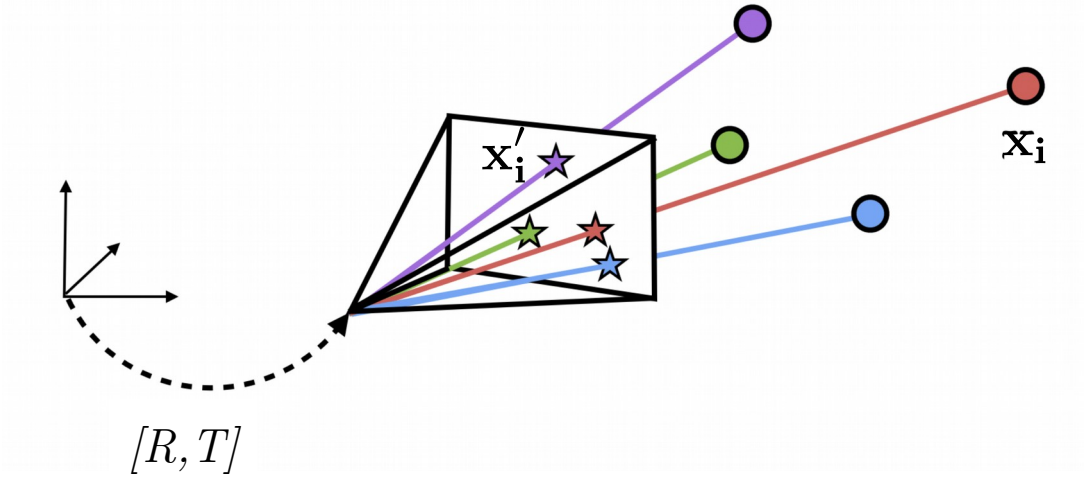


Image courtesy: T. Haavardsholm

Perspective from *n* Point (PnP) Problem



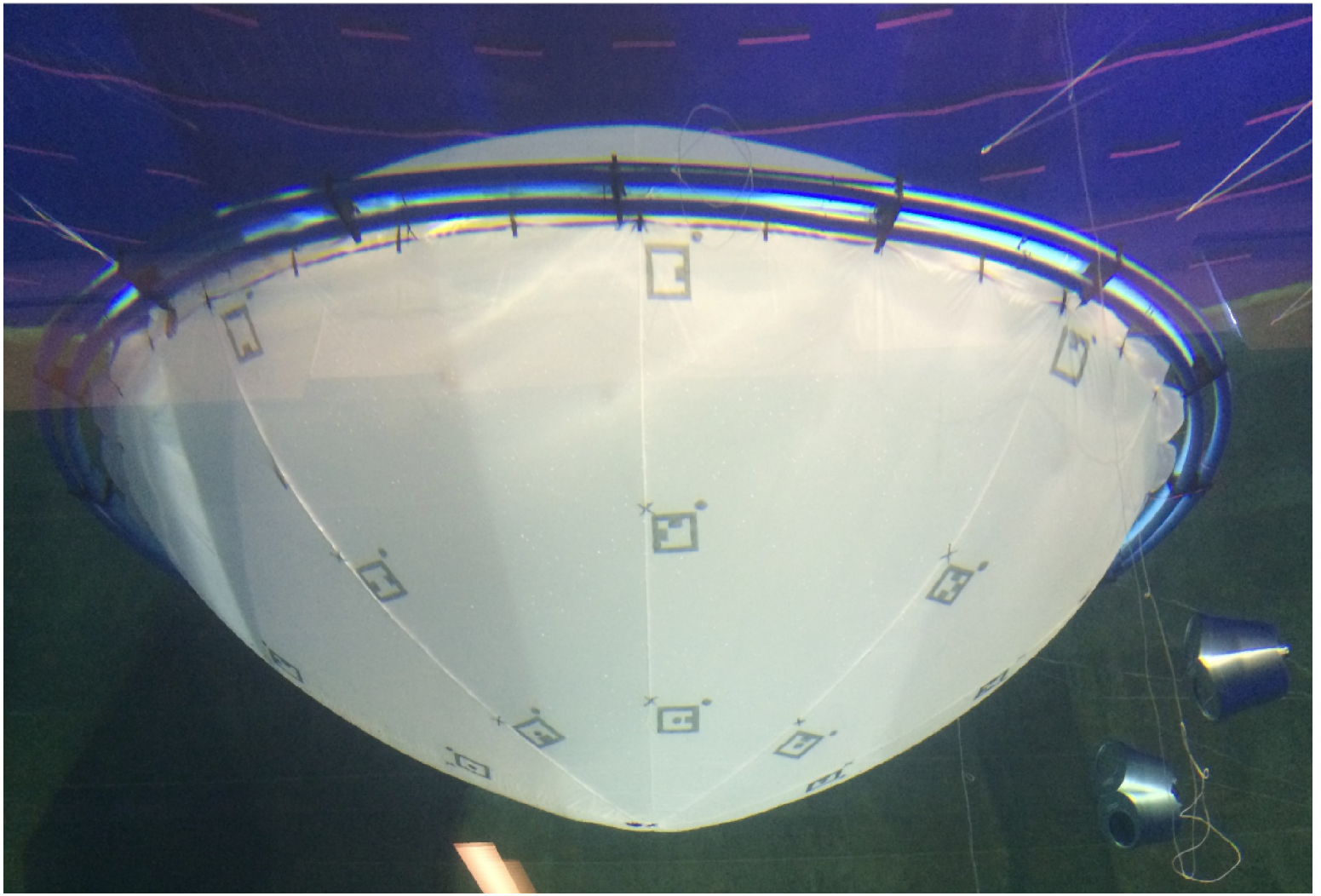
- Given: n 2D-3D correspondences $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}, i = 1, 2, ..., n$
- Compute pose $[R/T]$ s.t. $K[R/T] \quad \mathbf{x_i}' = H\mathbf{x_i}$
- Optionally: Also estimate internal camera matrix K
  - In form of individual parameters
  - In form of projection matrix $P = K[R/T]$

Image Courtesy: Torstein Sattler, CVPR 2015 Tutorial on Large-Scale Visual Place Recognition and Image-Based Localization

Perspective from *n* Point (PnP) Problem

Calibrated Camera (intrinsics are known)	Uncalibrated Camera (intrinsics are unknown)
Valid for any 3D point configurations	DLT
Minimum number of points: 3 P3P (Perspective form three points)	Minimum number of points: 4 coplanar points 6 points if non coplanar

P4P real world example



Zsolt Volent, Jens Birkevold, Annette Stahl, Andreas Lien, Leif Magne Sunde, Pål Lader, Experimental study of installation procedure and volume estimation of tarpaulin for chemical treatment of fish in floating cages, Aquacultural Engineering, Volume 78, Part B, 2017, Pages 105-113, ISSN 0144-8609, https://doi.org/10.1016/j.aquaeng.2017.05.003

Pose estimation - P6P

Problem: Given at least 6 known 3D landmark positions $\mathbf{x_i}$ in the world frame and

Given their 2D image correspondences $\mathbf{x_i'}$ in the camera frame, $\mathbf{x_i} \leftrightarrow \mathbf{x_i'}, i = 1, 2, ..., n$

Estimate the 6Dof 3D pose of the camera in the world frame including intrinsic parameters.

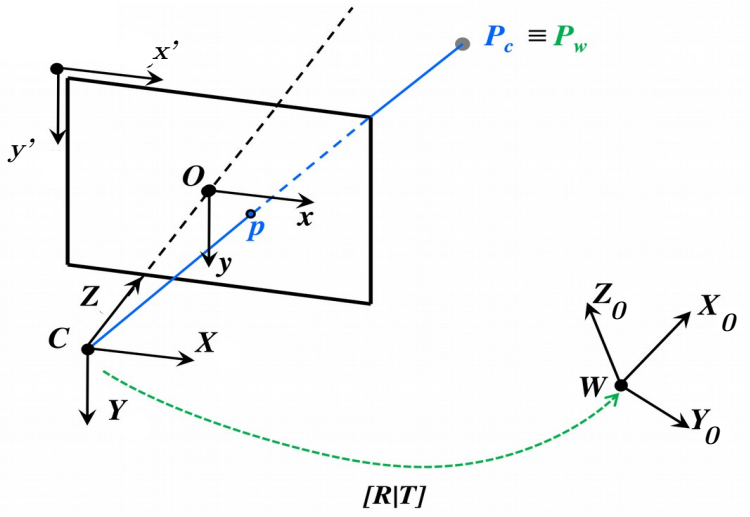
Solution: We can estimate a general **projection matrix** P that satisfies the perspective projection model

$$\lambda \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{x}}_\mathbf{i}' = K[R|T]\tilde{\mathbf{x}}_\mathbf{i}$$

$$\boxed{\tilde{\mathbf{x}}_\mathbf{i}' = P\tilde{\mathbf{x}}_\mathbf{i}}$$

where $P = K[R|T]$



Pose Estimation

The **algorithm** consists of two main parts:

1. Compute $P \in \mathbb{R}^{3\times 4}$

2. Decompose P into $K,\ R,$ and T using the **RQ decomposition**

Pose Estimation

1. Compute $P \in \mathbb{R}^{3\times 4}$: $\quad \tilde{\mathbf{x}}_\mathbf{i}' = P\tilde{\mathbf{x}}_\mathbf{i}$

Each correspondence generates two equations

$$x_i' = \frac{p_{11}x_i + p_{12}y_i + p_{13}z_i + p_{14}w_i}{p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34}w_i} \qquad y_i' = \frac{p_{21}x_i + p_{22}y_i + p_{23}z_i + p_{24}w_i}{p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34}w_i}$$

Multiplying the denominator gives equations linear in the elements of P

$$p_{11}x_i + p_{12}y_i + p_{13}z_i + p_{14}w_i + (p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34}w_i)x_i' = 0$$
$$p_{21}x_i + p_{22}y_i + p_{23}z_i + p_{24}w_i + (p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34}w_i)y_i' = 0$$

Rearrange in matrix form leads to a 2x12 matrix form

$$A_i p = \mathbf{0}$$

where $p = [p_{11}\ p_{12} \ldots p_{34}]^\top$ is a vector containing the matrix elements of P; solve with DLT

Pose Estimation

1. Alternative to DLT:

   Directly minimize the set of equations

$$x_i' = \frac{p_{11}x_i + p_{12}y_i + p_{13}z_i + p_{14}w_i}{p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34}w_i} \qquad y_i' = \frac{p_{21}x_i + p_{22}y_i + p_{23}z_i + p_{24}w_i}{p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34}w_i}$$

   by using non-linear least squares (→ L05)

Pose Estimation

The **algorithm** consists of two main parts:

1. Compute $P \in \mathbb{R}^{3\times 4}$

2. Decompose P into $K,\ R,$ and T using the **RQ decomposition**

Decompose P

2. Decompose P into $K,\ R,$ and T using the **RQ decomposition**

$$\begin{aligned} P &= K[R|T] \\ &= K[R|-RC] \\ &= [KR|-KRC] \\ &= [M|-MC] \end{aligned}$$

Where $M = KR$ and C is the camera center with respect to the world coordinate system.

2.1. Compute camera center

$$PC = 0$$

→ SVD of P:

$$P = USV^{\top}$$

(Note: C defines the null space of P)

→ C is the right singular vector, corresponding to the smallest singular value.

RQ Decomposition

Decompose M into KR

- RQ-factorization is a decomposition of a matrix M into the product $M = RQ$ where R is an upper triangular matrix and Q is an orthogonal matrix
- The RQ in ‘’RQ-decomposition’’ corresponds to KR in the expression $M = KR,$
  - Upper triangular matrix R = Camera matrix K
  - Orthogonal matrix Q = Rotation matrix R
- In addition:
  - K has a positive diagonal: $(KD)(D^{-1}R)$ where

$$\det(R) = 1 \rightarrow \quad \det(M) > 0$$

QR Decomposition

If RQ decomposition in not available in library documentation:

$$M = (QR)^{-1} = R^{-1}Q^{-1}$$

Matlab:

```
function [R,Q] = rq(M)

[Q,R] = qr(rot90(M,3));

R = rot90(R,2)';

Q = rot90(Q);
```

Example

```
%Estimate P by basic DLT.
est.P = estimateCameraMatrix_DLT(u,X);
est.P = est.P * sign(det(est.P(1:3, 1:3)));

% Estimate C.
[U,S,V] = svd(est.P);
est.C = V(1:3,end) / V(end, end);

% Estimating K and R by RQ decomposition.
[est.K,est.R] = rq(est.P(1:3,1:3));

% Enable positive diagonal of K by changing signs in both est.K and est.R.
D = diag(sign(diag(est.K)));
est.K = est.K*D;
est.R = D*est.R;
est.K = est.K/est.K(end,end);

%Determine t from the estimated C.
est.t = -est.R*est.C;
```

Pose Estimation

1. Extract local features
2. Find 3D to 2D correspondences
3. Estimate P from $\tilde{\mathbf{x}}_\mathbf{i}' = P\tilde{\mathbf{x}}_\mathbf{i}$
4. Decompose P into, $P = K[R|T]$ where $M = KR$ is a 3 × 3 matrix
5. Normalize P by multiplying all elements with $sign(\det(M))$
6. Compute the camera center C by SVD
7. RQ-decomposition of M gives us $M = K'R'$

   where K' is an upper triangular like K and R' is an orthogonal matrix like R
8. Define $D = diag(sign(K'_{11}), sign(K'_{22}), sign(K'_{33}))$
9. Then $K = K'D$ and $R = DR'$ and $T = -RC$

Pose Estimation / Localization

Extract Local Features

Establish 2D-3D Matches

Camera Pose Estimation:
RANSAC + n-Point-Pose Algorithm

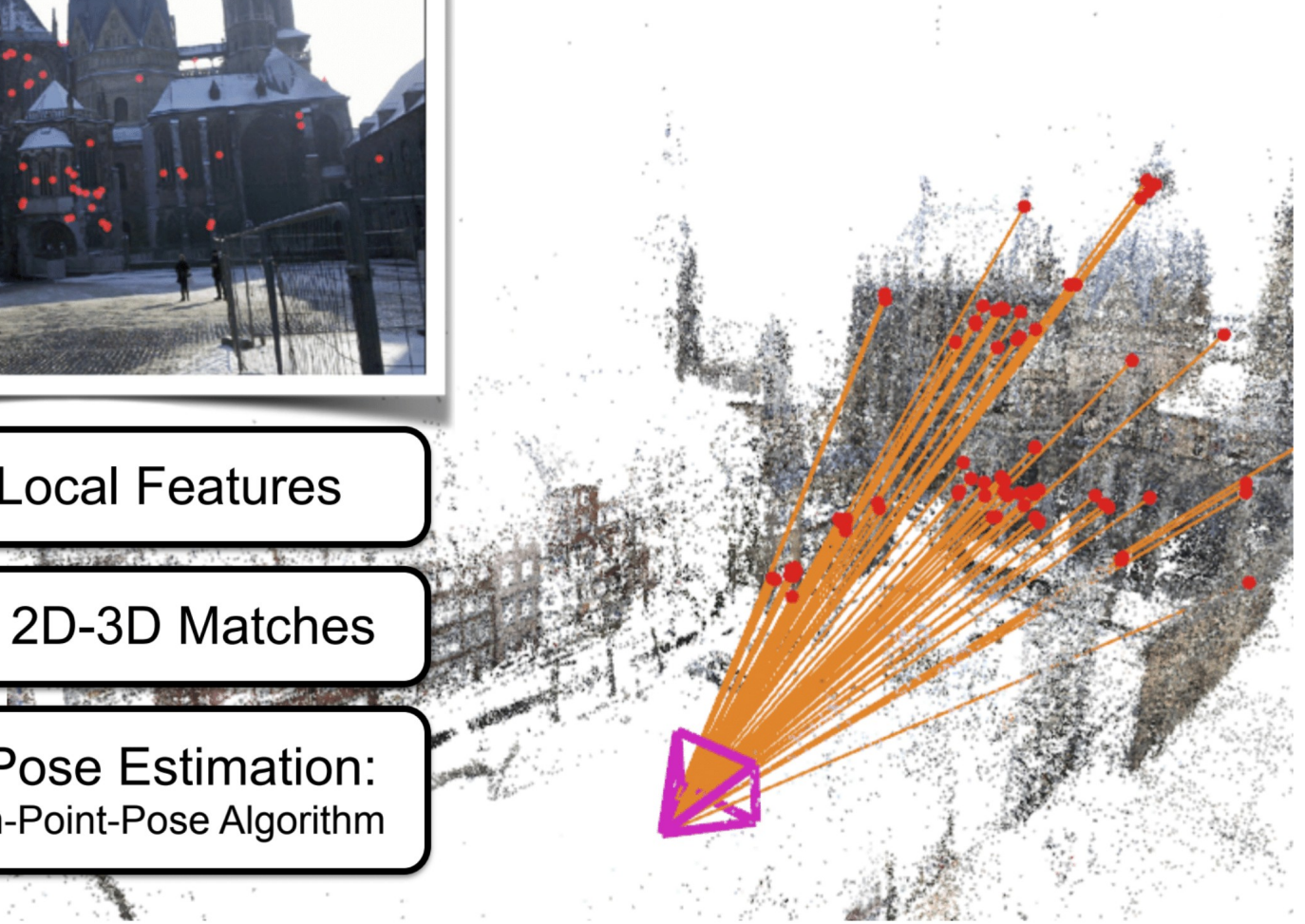


Image Courtesy: Torstein Sattler, CVPR 2015 Tutorial on Large-Scale Visual Place Recognition and Image-Based Localization

Alternative Methods

- Iterative methods
  - Initialize with P from DLT, clamp known parameters
  - Minimize geometric error
  - Better: Soft constraints, for example:

- Use Perspective-n-Point (PnP) Problem (several different methods available)
  - Typically fast non-iterative methods
  - Minimal in number of points
  - Accuracy comparable to iterative methods

Perspective from *n* Point (PnP) Problem

How many points are needed?

- n = 1 → P1P: infinitely many solutions
- n = 2 → P2P: infinitely many solutions, but bounded.

  In case the orientation of the 3D points is known a unique solution exist

  (points lying on a curve of known geometry [1])
- n = 3 → P3P: (no 3 collinear) finitely many solutions (up to 4)
- n = 4 → P4P: unique solution for 4 co-planar points (no 3 collinear)

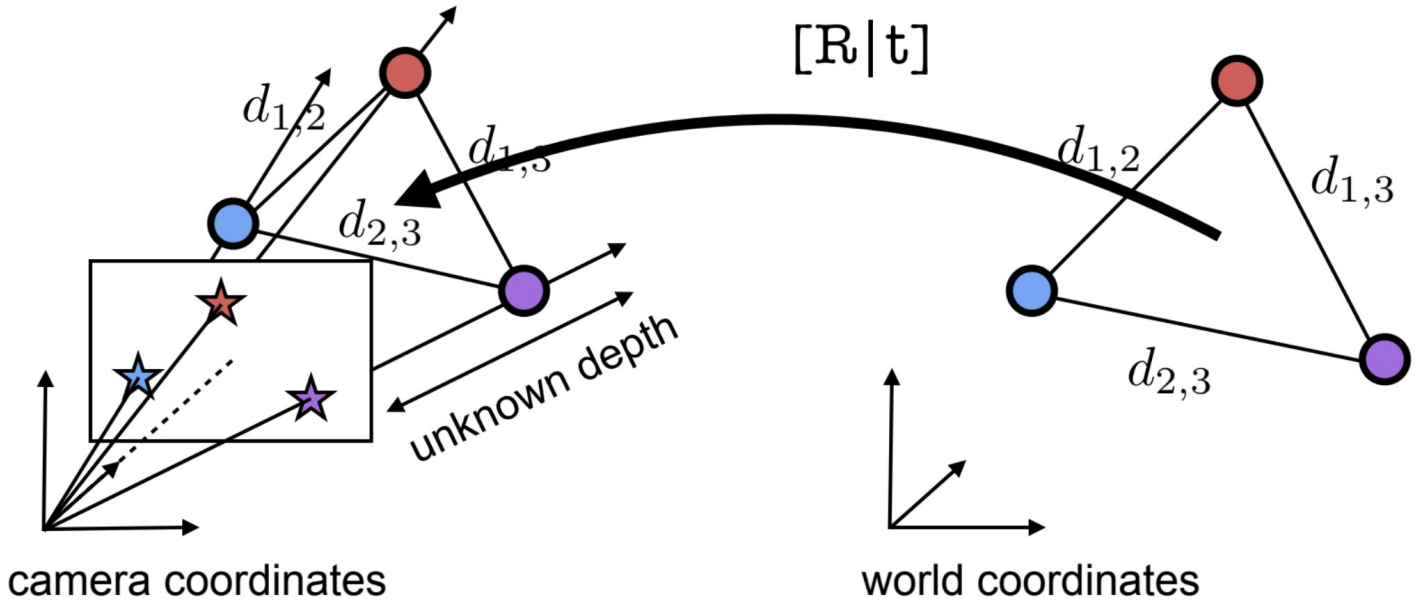
  case 4 non-coplanar points [2,3]

Examples

- P3P: Estimate pose, known K
- P4Pf: Estimate pose and focal length
- P6P: DLT!
- R6P: Estimate pose with rolling shutter

[1] Camera Pose Estimation Using First-Order Curve Differential Geometry
Ricardo Fabbri, Benjamin B. Kimia, and Peter J. Giblin, Brown University.

P3P Problem



- Case: Intrinsic calibration known
  [Haralick et al., ICVJ'94] http://www.haralick.org/journals/three_point_perspective.pdf
- Recover depths: Solve 4th degree univariate polynomial
  [Fischler, Bolles, CACM'91]https://dl.acm.org/doi/10.1145/358669.358692
- Recover pose by aligning local and global point positions
- Very efficient: ~2µs total [Kneip et al., CVPR'11]
- Up to four solutions: Disambiguate using 4th point

P4Pf – Unknown Focal Length

Estimate focal length and pose for 4 matches

http://cmp.felk.cvut.cz/~bujnam1/publications/Bujnak-et-al-CVPR-2008-final.pdf

- Solve system of multivariate polynomials
- Recover variables as Eigenvectors of 10x10 matrix
- Public available solver slow (~100μs)
  - Too slow if execution times are important
  - Faster variants exist

    [Bujnak et al., CVPR'12] http://cmp.felk.cvut.cz/~bujnam1/publications/Bujnak-Kukelova-Pajdla-CVPR-2012.pdf

    [Kukelova et al., ACCV'14] http://cmp.felk.cvut.cz/~kukelova/publications/Kukelova-etal-ACCV-2014.pdf
- Usually returns multiple solutions
  - Disambiguate using 5th point

Alternatives

- **P6P:** DLT, Linearly estimate full projection matrix from 6 matches [Hartley, Zisserman, 2004]
  - Very efficient to compute: Solve 12x12 linear system
  - Single solution!
  - Can obtain least-squares solutions using >6 matches

- **P5Pfr:** Non-minimal 5-point solver [Kukelova et al., ICCV'13]
  - As fast as P3P, only linear algebra operations
  - Estimates up to 3 radial distortion parameters

- **P3P(f):** Focal length sampling + P3P [Sattler et al., ECCV'14][code]
  - In each RANSAC iteration: Sample focal length value, estimate pose using P3P, update focal length distribution
  - Can be faster than P5Pfr+RANSAC
  - Similar accuracy as P4Pf+RANSAC

RANSAC Variants

Many different variants that improve upon standard RANSAC

- MSAC / MLESAC [Torr & Zisserman, CVIU'00]
- Td,d Test [Chum & Matas, BMVC'02]
- LO-RANSAC [Chum et al., DAGM'03] [Lebeda et al., BMVC'12][code]
- PROSAC [Chum & Matas, CVPR'05]
- Randomized RANSAC [Chum & Matas, PAMI'08]
- SCRAMSAC [Sattler et al., ICCV'09]
- USAC [Raguram et al., PAMI'13] (good overview, nice implementation)

Summary L04

Homography

- Automatic point-correspondences
- Wrong correspondences are common
- RANSAC estimation
  - Basic DLT (Direct Linear Transform) on 4 random correspondences
  - Inliers determined from the reprojection

- Improve estimate by normalized DLT
- inliers or iterative methods for an even better estimate

Pose estimation

- RQ Decomposition
- PnP Problem
- Alternative methods

Literature

Hartley & Zisserman Chapter 2.3 and Chapter 4.1