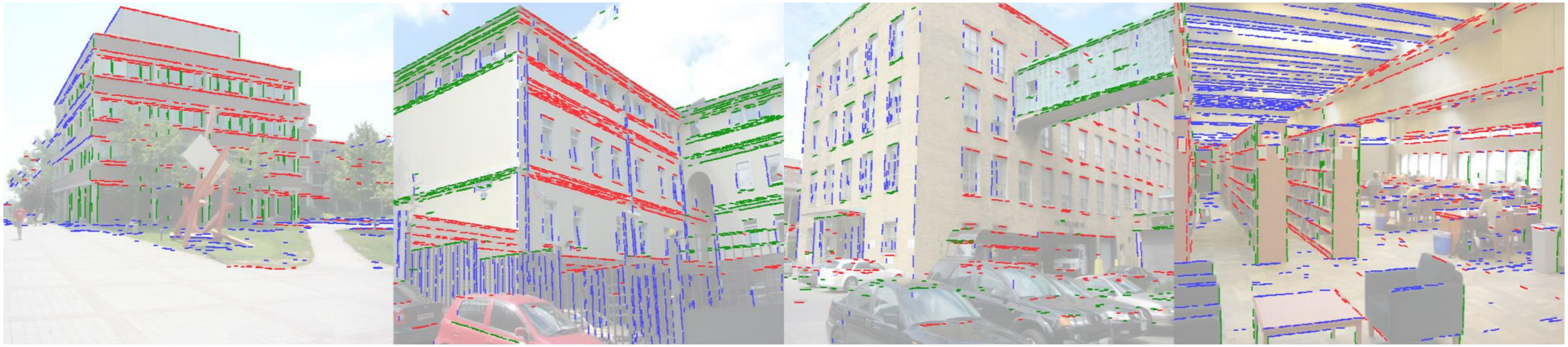
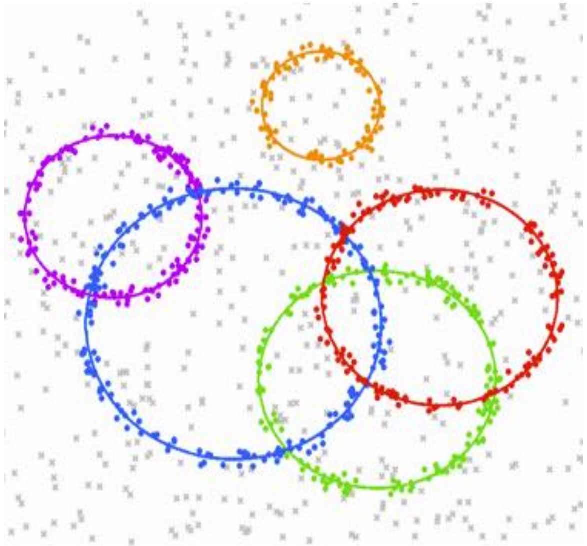
CS484/684 Computational Vision

Geometric Model Fitting





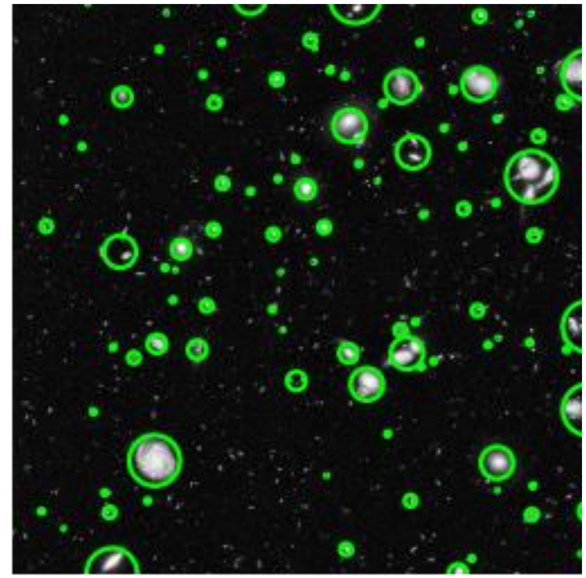
with some slides stolen from
Steve Seitz and Rick Szeliski

Geometric Model Fitting

- Feature matching $(\mathbf{p}_i, \mathbf{p}'_i)$
- Model fitting (e.g. homography estimation for panoramas)
  - How many points to choose?
  - Least square model fitting
  - RANSAC (robust method for model fitting)
- Multi-model fitting problems

Flashbacks: feature **detectors**

Harris corners



Dog

python code from “FeaturePoints.ipynb”

```
from skimage.feature import corner_harris, corner_subpix, corner_peaks

hc_filter = corner_harris(image_gray)
peaks = corner_peaks(hc_filter)
```

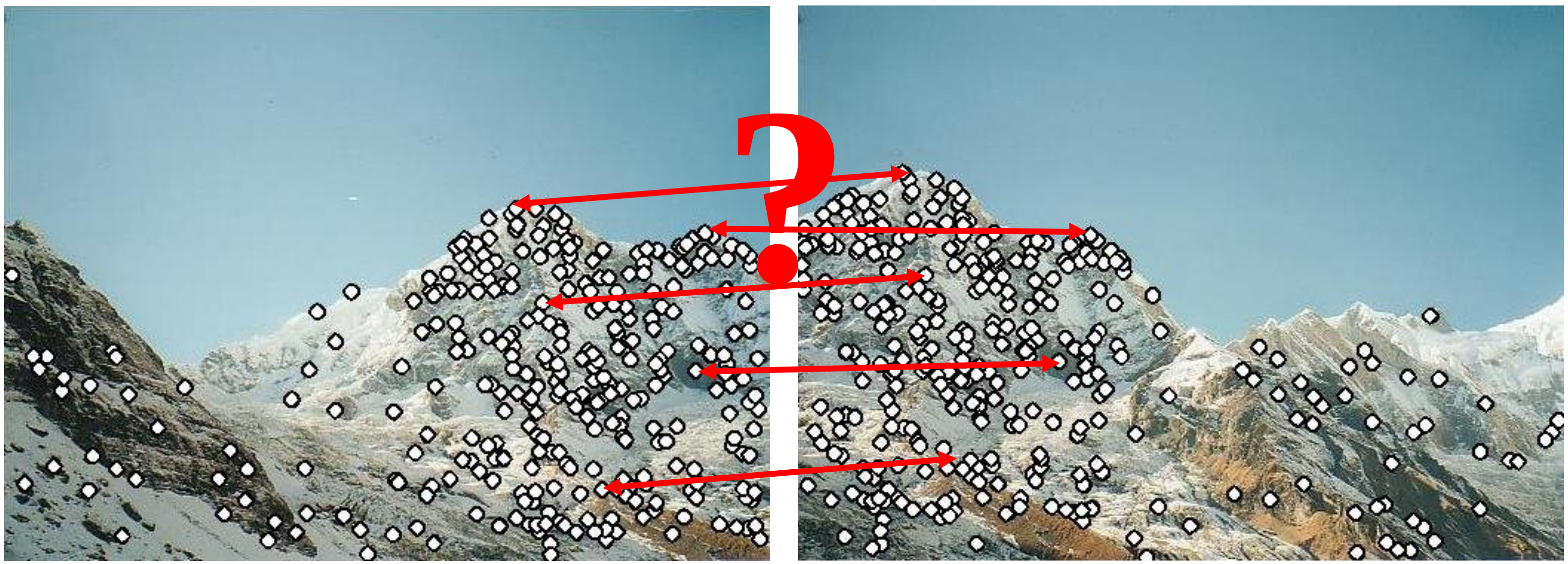
```
from skimage.feature import blob_dog

blobs = blob_dog(image_gray)
```

Flashbacks: feature **descriptors**

We know how to detect points

Next question: **How to match them?**



need **point descriptors** that should be

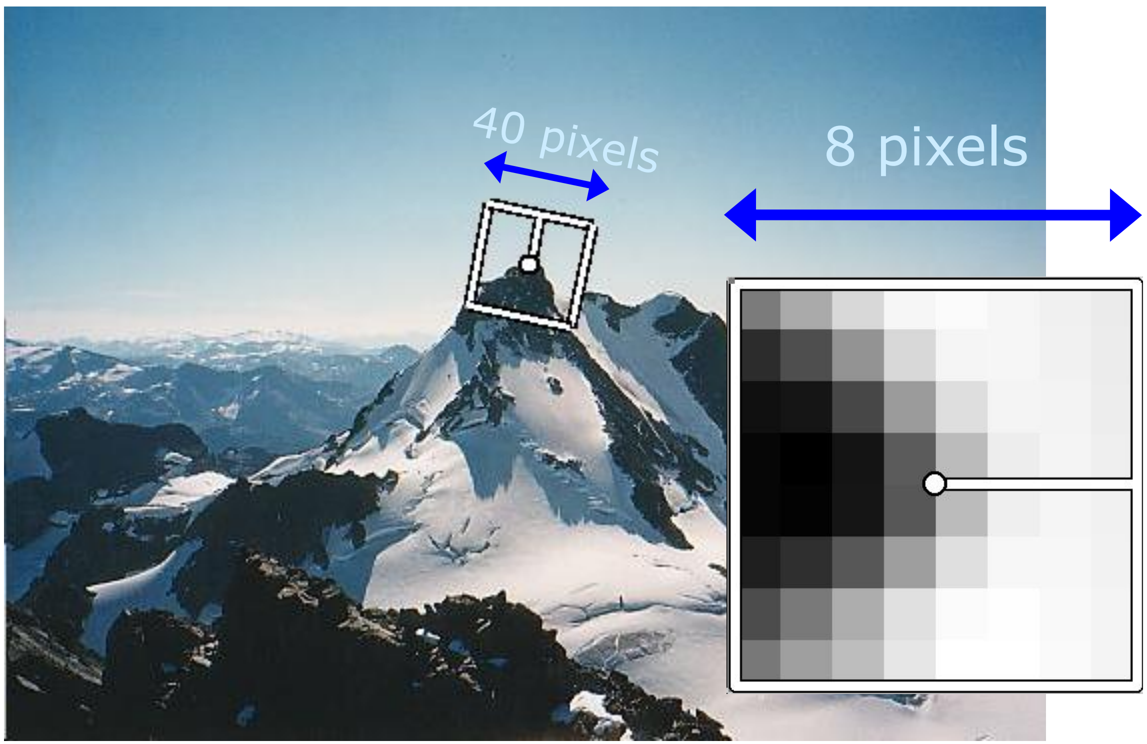
- Invariant (e.g. to gain/bias, rotation, projection, etc)
- Distinctive (to avoid false matches)

Flashbacks: MOPS descriptor

8x8 oriented patch

- Sampled at 5 x scale

Bias/gain normalization: $I' = (I - \mu)/\sigma$



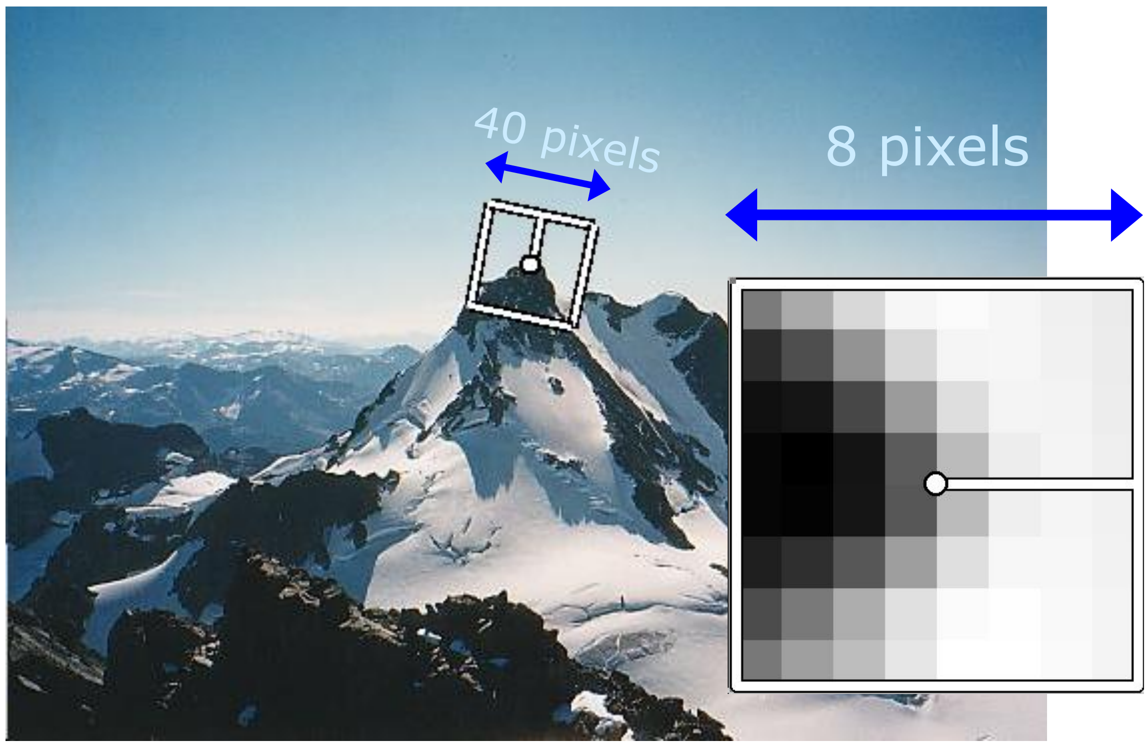
Another popular idea (SIFT): use gradient orientations inside the patch as a descriptor (also invariant to gain/bias)

Flashbacks: MOPS descriptor

8x8 oriented patch

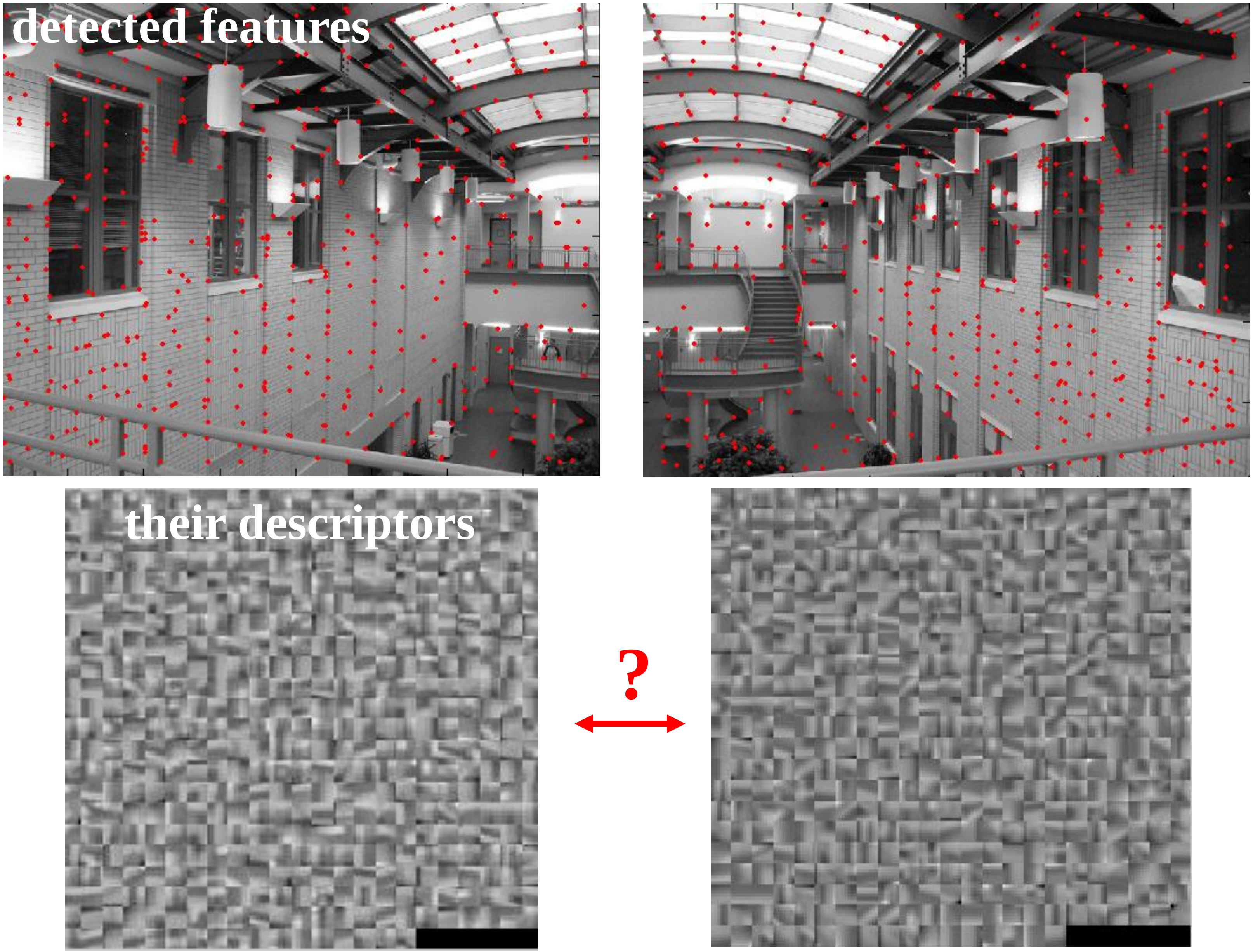
- Sampled at 5 x scale

Bias/gain normalization: $I' = (I - \mu)/\sigma$



Popular descriptors: MOPS, SIFT, SURF, HOG, BRIEF, many more…

Feature matching



Feature matching

Optimal matching:

- Bipartite matching, quadratic assignment (QA) problems
  - too expensive

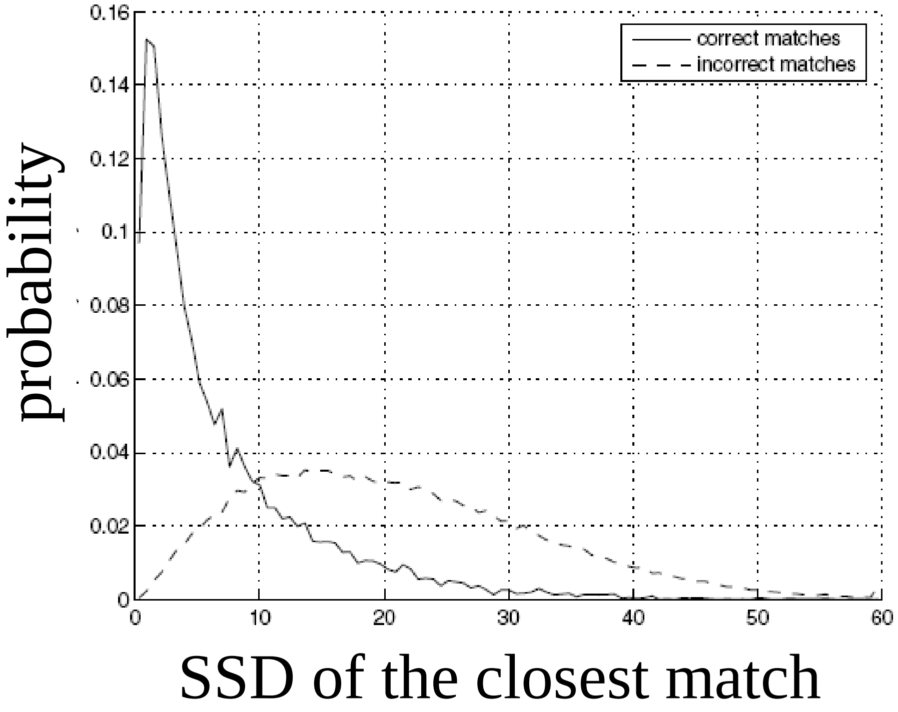
Common simple approach:

- use SSD (sum of squared differences) between two descriptors (patches).
- for each feature in image 1 find a feature in image 2 with the lowest SSD
- accept a match if SSD(patch1,patch2) < T (threshold)

Feature matching

SSD(patch1,patch2) < T

How to set threshold T?

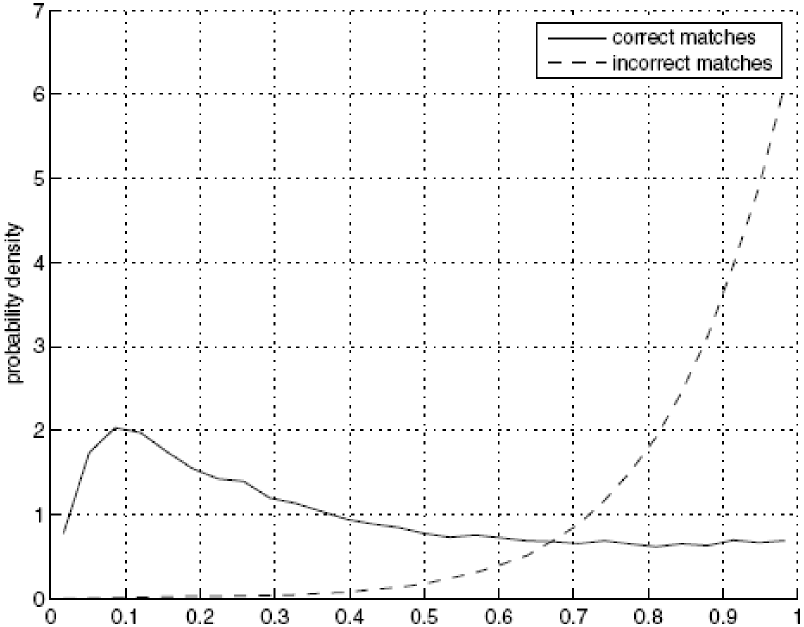


no threshold T is good for separating correct and incorrect matches

Feature matching

A better way [Lowe, 1999]:

- SSD of the closest match (SSD1)
- SSD of the second-closest match (SSD2)
- Accept the best match if it is much better than the second-best match (and the rest of the matches)



(SSD1) / (SSD2)

easier to select threshold T for decision test

(SSD1) / (SSD2) < T

Python example (BRIEF descriptor)

```
from skimage.feature import (corner_harris, corner_peaks, plot_matches, BRIEF, match_descriptors)

keypointsL = corner_peaks(corner_harris(imL), threshold_rel=0.0005, min_distance=5)
keypointsR = corner_peaks(corner_harris(imR), threshold_rel=0.0005, min_distance=5)

extractor = BRIEF()

extractor.extract(imL, keypointsL)
keypointsL = keypointsL[extractor.mask]
descriptorsL = extractor.descriptors

extractor.extract(imR, keypointsR)
keypointsR = keypointsR[extractor.mask]
descriptorsR = extractor.descriptors

matchesLR = match_descriptors(descriptorsL, descriptorsR, cross_check=True)
```

find the closest match p' for any feature p

crosscheck: keep pair (p,p') only if p is the best match for p'

How to fit a homorgaphy???

What problems do you see for homography estimation?

How to fit a homorgaphy???

What problems do you see for homography estimation?

Issue 1: the number of matches $(\mathbf{p}_i, \mathbf{p}'_i)$ is more than 4

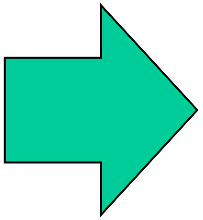
Answer: model fitting via "least squares" (later, slide 21)

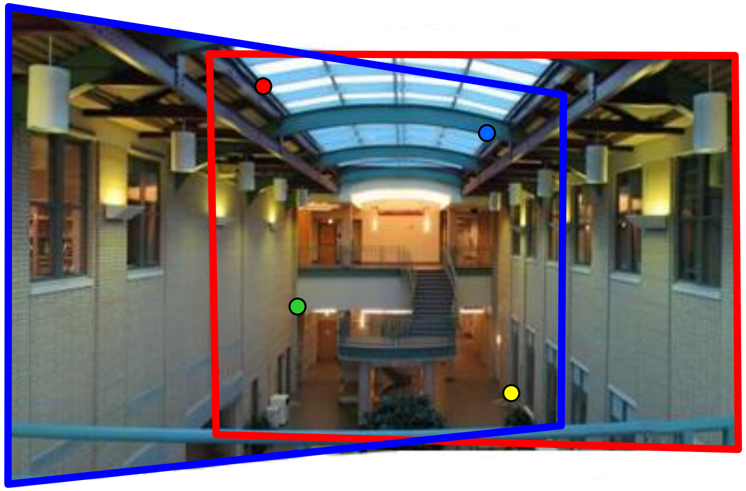
Issue 2: too many *outliers* or wrong matches $(\mathbf{p}_i, \mathbf{p}'_i)$

Answer: robust model fitting via RANSAC (later, slide 35)

Recall: Homography from 4 points

(from Topic 5)





Recall: Homography from 4 points

Consider one match (point-correspondence) $p = (x, y) \rightarrow p' = (x', y')$

$$\mathbf{p}' = \mathbf{H}\mathbf{p} \qquad \begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

After eliminating $w = gx + hy + i$ **:**

$$\Rightarrow \qquad \begin{aligned} ax + by + c - gxx' - hyx' - ix' &= 0 \\ dx + ey + f - gxy' - hyy' - iy' &= 0 \end{aligned}$$

Two equations linear w.r.t unknown coefficients of matrix H and quadratic w.r.t. known point coordinates (x,y,x',y')

Recall: Homography from 4 points

Consider 4 point-correspondences $p_i = (x_i, y_i) \rightarrow p'_i = (x'_i, y'_i)$

$$\mathbf{p}'_i = \mathbf{H}\mathbf{p}_i \qquad \begin{bmatrix} w_i x'_i \\ w_i y'_i \\ w_i \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

for i=1,2,3,4

$$\Rightarrow \quad \begin{aligned} ax_i + by_i + c - gx_i x'_i - hy_i x'_i - ix'_i = 0 \\ dx_i + ey_i + f - gx_i y'_i - hy_i y'_i - iy'_i = 0 \end{aligned}$$

Special case of DLT method (see p.89 in Hartley and Zisserman)

Can be written as matrix multiplication $\mathbf{A}_i \cdot \mathbf{h} = \mathbf{0}$ **for i=1,2,3,4**

where $\mathbf{h} = [\,a\; b\; c\; d\; e\; f\; g\; h\; i\,]^T$ **is a vector of unknown coefficients in** $\mathbf{H}$

and $\mathbf{A}_i$ **is a 2x9 matrix based on known point coordinates** x_i, y_i, x'_i, y'_i

Recall: Homography from 4 points

Consider 4 point-correspondences $p_i = (x_i, y_i) \rightarrow p'_i = (x'_i, y'_i)$

$$\mathbf{p}'_i = \mathbf{H}\mathbf{p}_i \quad \Rightarrow \quad \underset{2x9}{\mathbf{A}_i} \cdot \underset{9x1}{\mathbf{h}} = \underset{2x1}{\mathbf{0}} \quad \text{for i=1,2,3,4}$$

All four matrix equations can be "stacked up" as

$$\underset{8x9}{\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \mathbf{A}_3 \\ \mathbf{A}_4 \end{bmatrix}} \cdot \mathbf{h} = \underset{8x1}{\mathbf{0}}$$

or

$$\boxed{\underset{8x9}{\mathbf{A}} \cdot \underset{9x1}{\mathbf{h}} = \underset{8x1}{\mathbf{0}}}$$

iClilcker **Q:** how many solutions for h? A: none B: one C: many

Recall: Homography from 4 points

Consider 4 point-correspondences $p_i = (x_i, y_i) \;\rightarrow\; p'_i = (x'_i, y'_i)$

homogeneous linear equations

$$\mathbf{p}'_i = \mathbf{H}\mathbf{p}_i \quad \Rightarrow \quad \boxed{\mathbf{A} \cdot \mathbf{h} = \mathbf{0}} \qquad (*)$$

8x9 9x1 8x1

for i=1,2,3,4

8 linear equations, 9 unknowns: trivial solution h=0?

All solutions h form the (right) null space of A of dimension 1, but they represent the same transformation (as homographies can be scaled)

To find one specific solution h, for now fix one element, *e.g.* ***i* = 1**

as discussed in topic 5, **this may not work** more generally, should fix norm $\|\mathbf{h}\|=1$ (later)

$$\Rightarrow \quad \boxed{\mathbf{A}_{1:8} \cdot \mathbf{h}_{1:8} = -\mathbf{A}_9}$$

8x8 8x1 8x1

first 8 columns of A **first 8 rows of h** **9th columns of A**

Homography from more than 4 points

Consider N point-correspondences $p_i = (x_i, y_i) \rightarrow p'_i = (x'_i, y'_i)$

$$\mathbf{p'}_i = \mathbf{H}\mathbf{p}_i$$

for $i = 1, \ldots, N$

Questions:

Are there any benefits from knowing more point correspondences?
What if 4 points correspondences are known with error?

over-constrained system

$$\Rightarrow \quad \boxed{\mathbf{A}_{1:8} \cdot \mathbf{h}_{1:8} = -\mathbf{A}_9}$$

2Nx8 8x1 2Nx1

First, consider a simpler model fitting problem…

Simpler example: line fitting

Assume a set of data points $(X_1, X'_1), (X_2, X'_2), (X_3, X'_3)$, ...

(e.g. person's height vs. weight)

We want to fit a linear model (a, b) to predict X' from X

$$a \cdot X + b = X'$$

How many pairs (X_i, X'_i) do we need to find a and b?

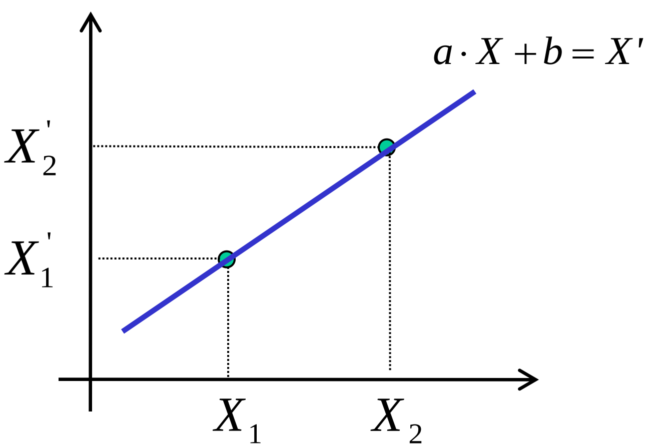
$$X_1 a + b = X'_1$$

$$X_2 a + b = X'_2$$

$$\begin{bmatrix} X_1 & 1 \\ X_2 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} X'_1 \\ X'_2 \end{bmatrix}$$

$$A \cdot x = B$$

$$x = A^{-1} \cdot B$$



Simpler example: line fitting

Assume a set of data points $(X_1, X_1'), (X_2, X_2'), (X_3, X_3'), \ldots$

(e.g. person's height vs. weight)

We want to fit a linear model to predict X' from X

$$a \cdot X + b = X'$$

What if the data points (X_i, X_i') are noisy?

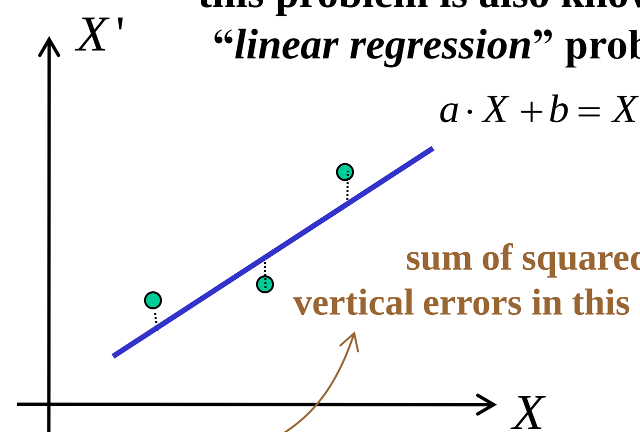
over-constrained

$$\begin{bmatrix} X_1 & 1 \\ X_2 & 1 \\ X_3 & 1 \\ \ldots & \ldots \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} X_1' \\ X_2' \\ X_3' \\ \ldots \end{bmatrix}$$

$$A \cdot x = B$$

$$\min_x \|Ax - B\|^2$$ **(least-squares)**

this problem is also known as "*linear regression*" problem



sum of squared vertical errors in this example

$$x = A^{-1} \cdot B$$

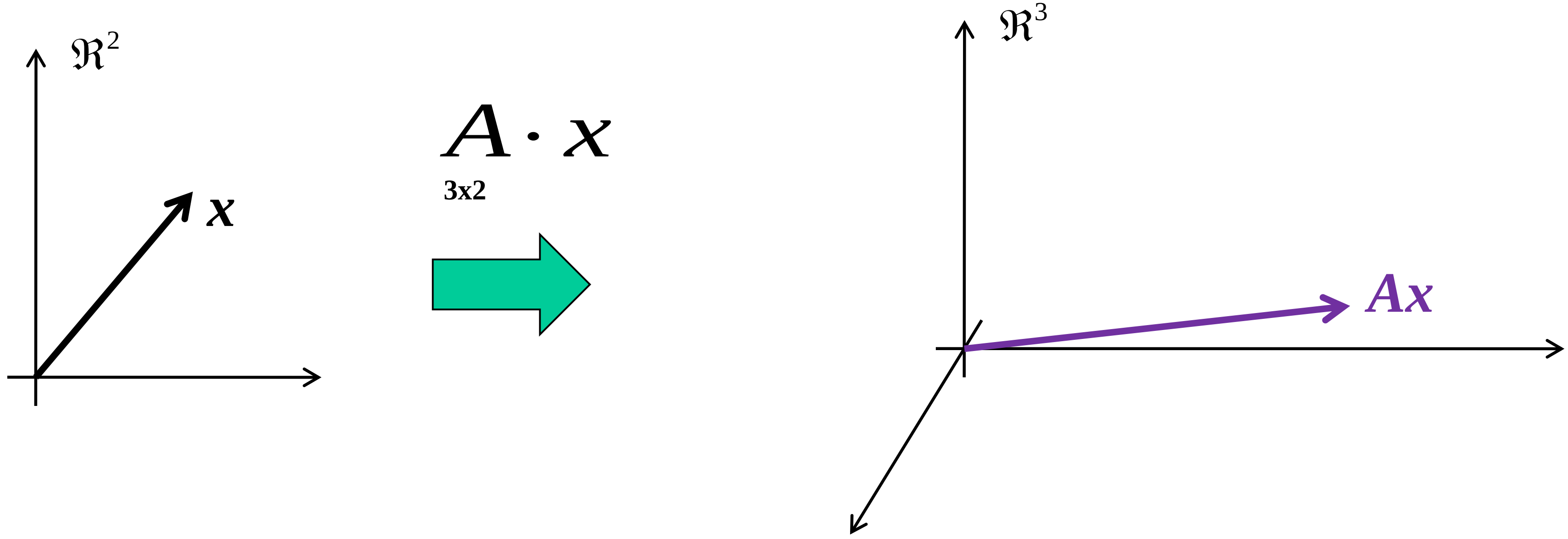
where $A^{-1} \equiv V \cdot W^{-1} \cdot U^T$ **is a *pseudo-inverse***
based on SVD decomposition $A = U \cdot W \cdot V^T$
(in python, one can use *svd* function in library *numpy.linalg*)

SVD: rough idea

$$A = U \cdot W \cdot V^T$$

M ≥ N: **MxN** **MxN** **NxN** **NxN**

where U and V are matrices with ortho-normal columns and W is diagonal with elements $w_i \geq 0$
(see "Numerical Recipes in C", edition 2, Sec. 2.6)



iClicker Moment: **where are all points from $\mathscr{R}^2$ mapped to?**

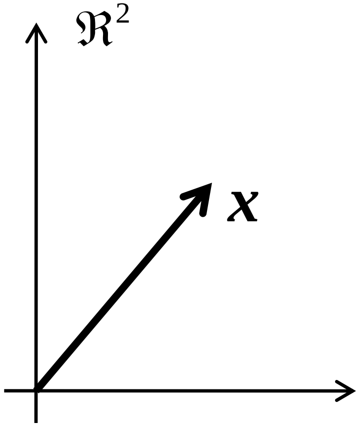
A: point B: line C: plane D: whole R^3

SVD: rough idea

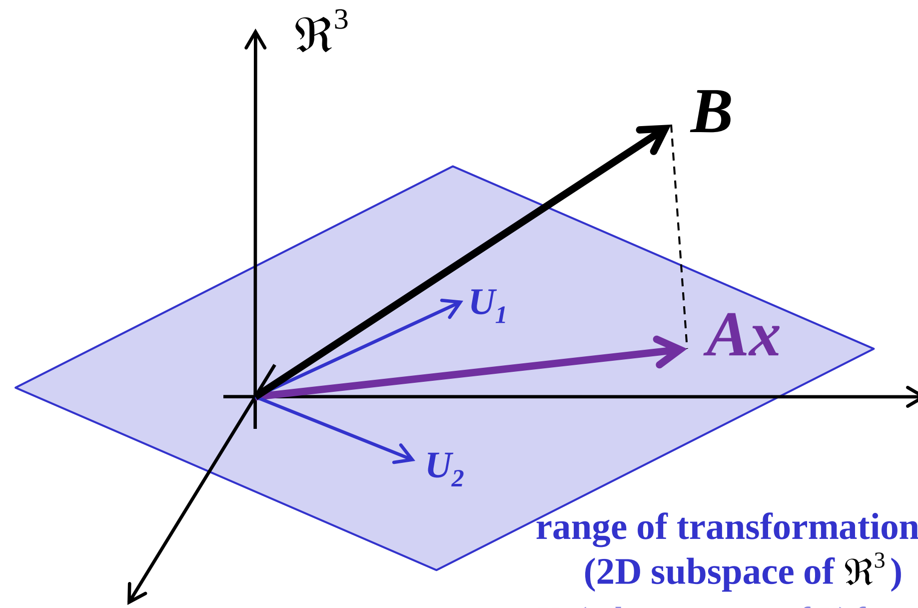
$$\underset{\text{MxN}}{A} = \underset{\text{MxN}}{\overset{embed}{U}} \cdot \underset{\text{NxN}}{\overset{scale}{W}} \cdot \underset{\text{NxN}}{\overset{rotate}{V^T}}$$

M ≥ N:

where U and V are matrices with ortho-normal columns and W is diagonal with elements $w_i \geq 0$
(see "Numerical Recipes in C", edition 2, Sec. 2.6)



$$\underset{3x2}{A \cdot x}$$



range of transformation *A* (2D subspace of $\Re^3$)

U_i (column-vectors of U) form the basis of this subspace

How does SVD help to solve *least-squares* ?

$$\min_x \|Ax - B\|^2$$

<u>projection of *B* onto range of *A*</u>

$$x = A^{-1} \cdot B \equiv V \cdot W^{-1} \cdot U^T \cdot B$$

Equivalent (fast to compute) expression

$$\underset{\text{Nx1}}{x} = \underset{\text{NxN}}{(A^TA)^{-1}} \cdot \underset{\text{NxM}}{A^T} \cdot \underset{\text{Mx1}}{B}$$

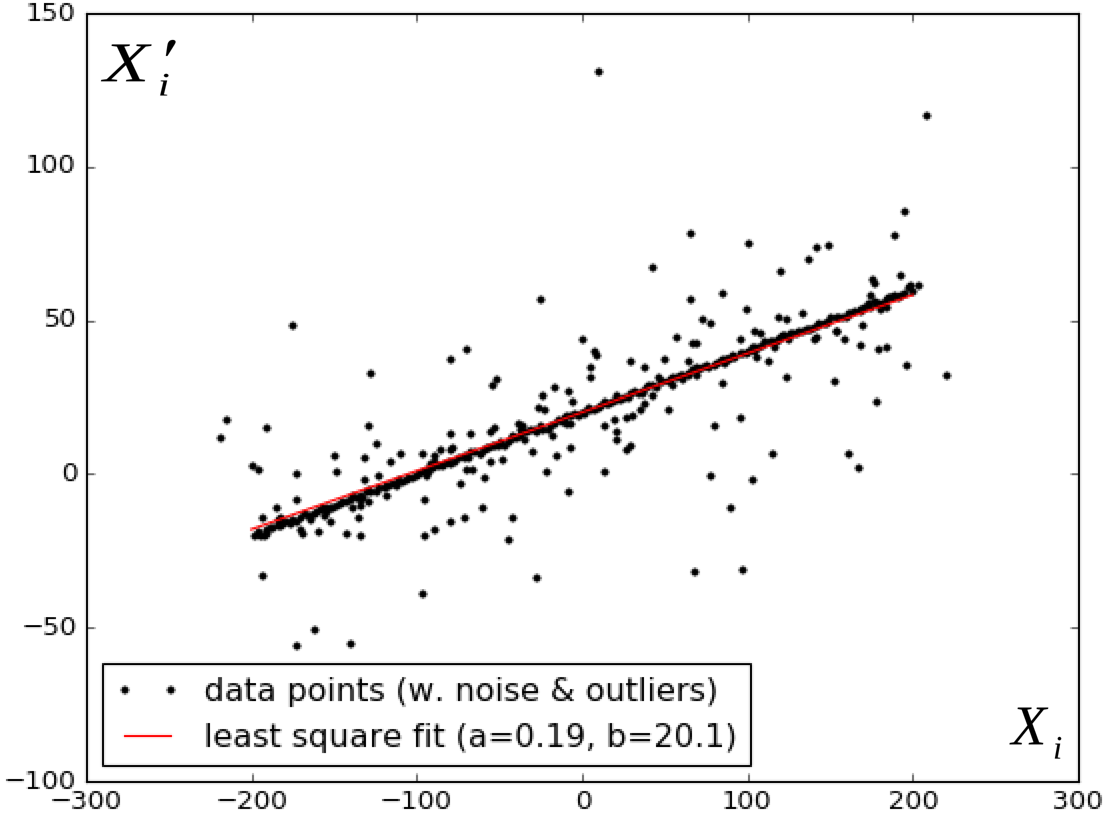
Indeed: $A^TA = VWU^TUWV^T = VW^2V^T$

so $(A^TA)^{-1} \cdot A^T = VW^{-2}V^T \cdot VWU^T = VW^{-1}U^T$

If M>>N computing inverse of *positive semi-definite* NxN matrix A^TA can be faster than SVD of MxN matrix A

Least squares line fitting

Data generated as $X'_i = a\ X_i + b + \delta X_i$ for Normal noise δX_i



$$\min_{x=(a,b)^\top} \|Ax - B\|^2 \longrightarrow x = A^{-1}B$$

least squares

Homography from N ≥ 4 points

Consider N point correspondences $p_i = (x_i, y_i) \rightarrow p'_i = (x'_i, y'_i)$

$$\mathbf{p}'_i = \mathbf{H}\mathbf{p}_i \quad \Rightarrow \quad \underset{2N\text{x}9}{\mathbf{A}} \cdot \underset{9\text{x}1}{\mathbf{h}} = \underset{2N\text{x}1}{\mathbf{0}} \qquad (*)$$

for $i = 1,\ldots,N$

over-constrained system

Approach 1: **add constraint *i* =1**. So, there are only 8 unknowns.

Set up a system of linear equations for vector of unknowns $\boldsymbol{h}_{1:8} = [a,b,c,d,e,f,g,h]^T$

$$\underset{2N\text{x}8}{\mathbf{A}_{1:8}} \cdot \underset{8\text{x}1}{\mathbf{h}_{1:8}} = \mathbf{B} = -\underset{2N\text{x}1}{\mathbf{A}_9}$$

solve

$$\min_{h_{1:8}} \left\| A_{1:8} h_{1:8} - B \right\|^2$$

(*least-squares*)

compute inverse for $\mathbf{A}_{1:8}^T \mathbf{A}_{1:8}$ as in line fitting, then $\boldsymbol{h}_{1:8} = (\mathbf{A}_{1:8}{}^T \cdot \mathbf{A}_{1:8})^{-1} \cdot \mathbf{A}_{1:8}{}^T \cdot (-\mathbf{A}_9)$

Homography from N ≥ 4 points

Consider N point correspondences $p_i = (x_i, y_i) \;\to\; p'_i = (x'_i, y'_i)$

$$\mathbf{p}'_i = \mathbf{H}\mathbf{p}_i \quad \Rightarrow \quad \boxed{\mathbf{A} \cdot \mathbf{h} = \mathbf{0}} \quad (*)$$

for $i = 1,\ldots,$N

2Nx9 9x1 2Nx1

over-constrained system

Approach 2**:** **add constraint ||h||=1**

solve
$$\min_{h:\|h\|=1} \|A\,\mathbf{h}\|^2$$
(*homogeneous least-squares*)

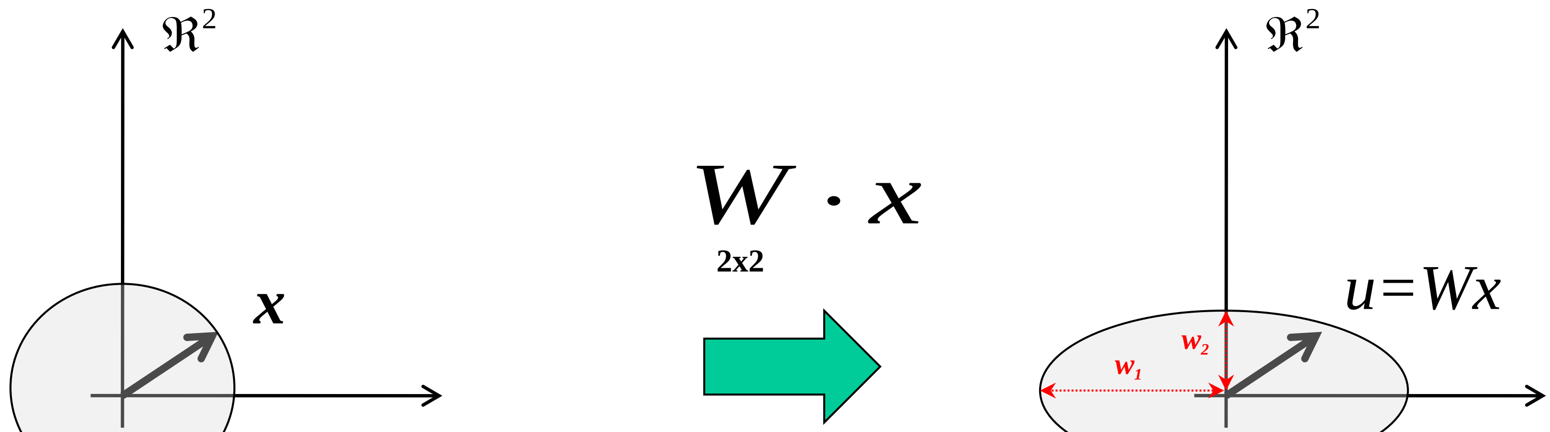
Solution: (unit) eigenvector of $\mathbf{A}^T\mathbf{A}$
corresponding to the smallest eigen-value
(use SVD, see next slide)

DLT method
(see p.91
in Hartley and
Zisserman)

Simple motivating example:

Consider 2x2 diagonal matrix $W = \begin{bmatrix} w_1 & 0 \\ 0 & w_2 \end{bmatrix}$

solve: $\min_{x:\|x\|=1} \| W \cdot x \|$



unit circle in $\Re^2$
$\|x\| = 1$ or $x^T \cdot x = 1$

Ellipsoid in $\Re^2$

$$x \cdot x^T = 1 \;\Leftrightarrow\; u^T W^{-2} u = 1$$

$$\frac{u_1^2}{w_1^2} + \frac{u_2^2}{w_2^2} = 1$$

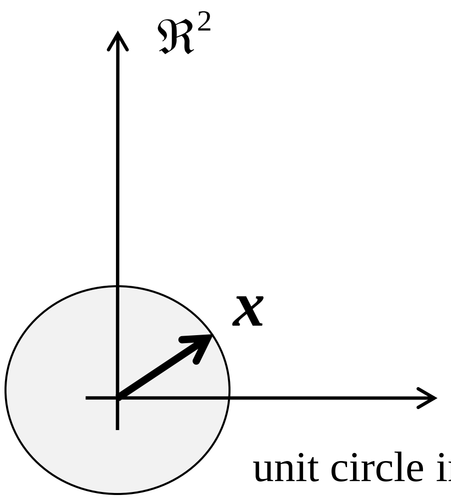
Solution: $x = (1,0)$ if $w_1 < w_2$
$x = (0,1)$ if $w_2 < w_1$

$\Leftarrow$ equivalently, solve $\min_{u \in Ellips} \| u \|$

General case: use SVD (rough idea)

$$\underset{M\times N}{A} = \overset{embed}{\underset{M\times N}{U}} \cdot \overset{scale}{\underset{N\times N}{W}} \cdot \overset{rotate}{\underset{N\times N}{V^T}} \qquad M \geq N$$

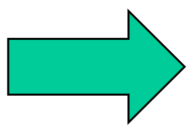
where U and V are matrices with ortho-normal columns and W is diagonal with elements $w_i \geq 0$ (see "Numerical Recipes in C", edition 2, Sec. 2.6)

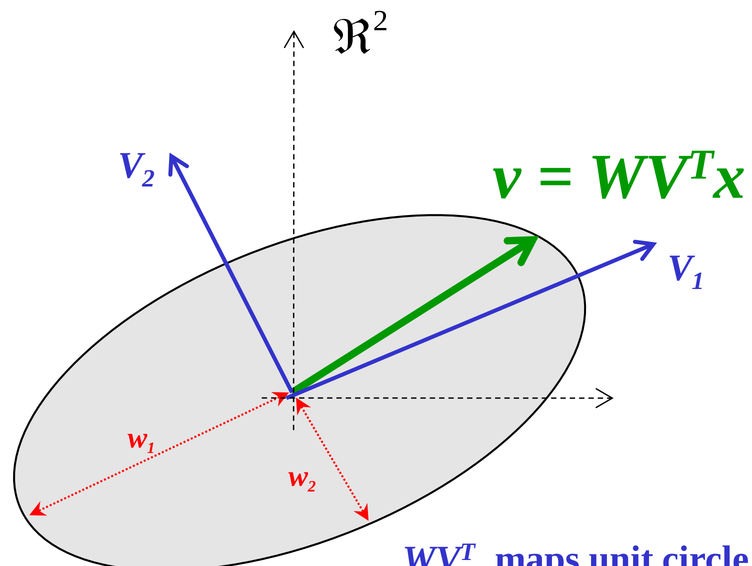


unit circle in $\Re^2$

$||x|| = 1$ or $x^T \cdot x = 1$

$$\underset{2x2}{W}\,\underset{2x2}{V^T} \cdot x$$





WV^T maps unit circle $x^T \cdot x = 1$
onto ellipsoid $v^T W^{-2} v = 1$
in basis $\{V_1, V_2\}$

check that $x = V_i$ is mapped to point
$W V^T x = w_i\, \mathbf{e}_i$

vector with zeros
and 1 in the i-th position

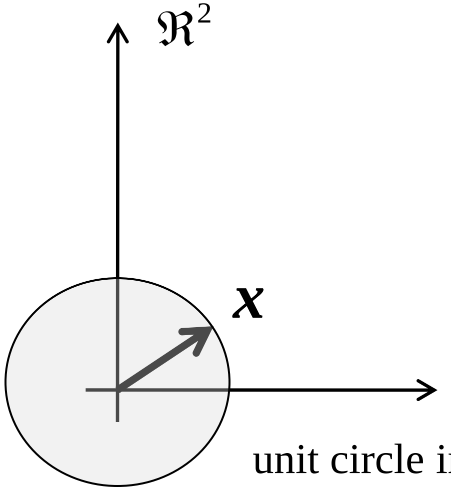
- **can interpret multiplication by V^T as change of ortho-normal basis**
  (or as space rotation, but illustration should be modified)
- **interpret multiplication by W as anisotropic scaling/deformation**

General case: use SVD (rough idea)

$$\underset{\textit{embed}}{A = U} \cdot \underset{\textit{scale}}{W} \cdot \underset{\textit{rotate}}{V^T}$$

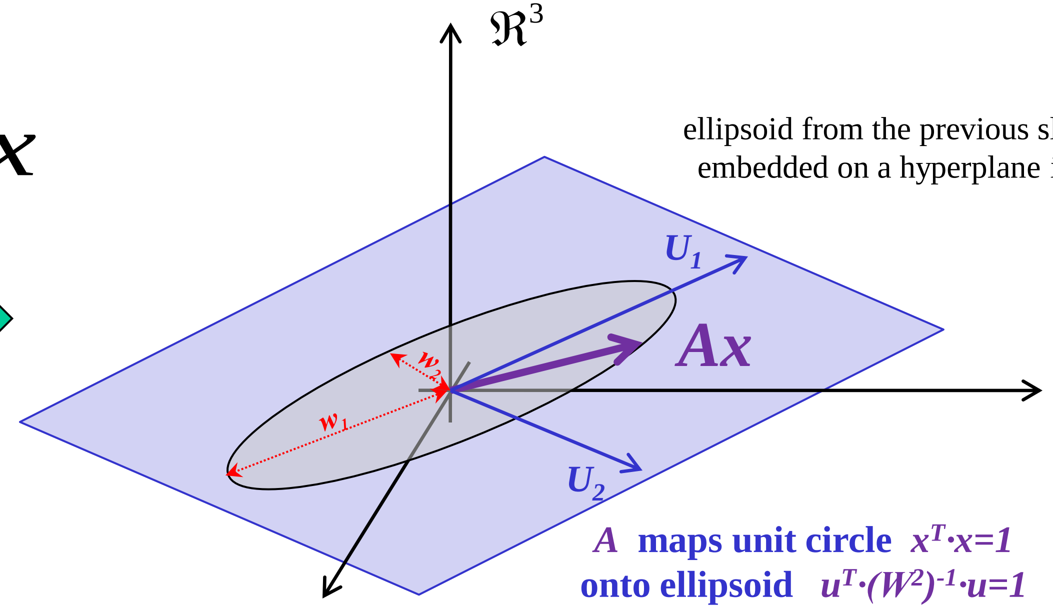
M ≥ N: MxN MxN NxN NxN

where U and V are matrices with ortho-normal columns and W is diagonal with elements $w_i \geq 0$ (see "Numerical Recipes in C", edition 2, Sec. 2.6)



unit circle in $\Re^2$
$||x|| = 1$ or $x^T \cdot x = 1$

$A \cdot x$
3x2



ellipsoid from the previous slide embedded on a hyperplane in $\Re^3$

A maps unit circle $x^T \cdot x = 1$ onto ellipsoid $u^T \cdot (W^2)^{-1} \cdot u = 1$ in the range of A (2D subspace of $\Re^3$)

Indeed, the coordinates of vector u are in basis $\{U_1, U_2\}$, i.e. $u = WV^T \cdot x$ and $x = VW^{-1} \cdot u$

How does SVD help to solve $\min_{x:||x||=1} ||\mathbf{A} \cdot x||$ **?**

$$A \cdot V_i = UWV^T \cdot V_i = UW \cdot e_i = w_i \cdot (U \cdot e_i) = w_i \cdot U_i \quad \Rightarrow \quad ||A \cdot V_i|| = w_i$$

vector $x = V_i$ corresponding to the least w_i solves the problem

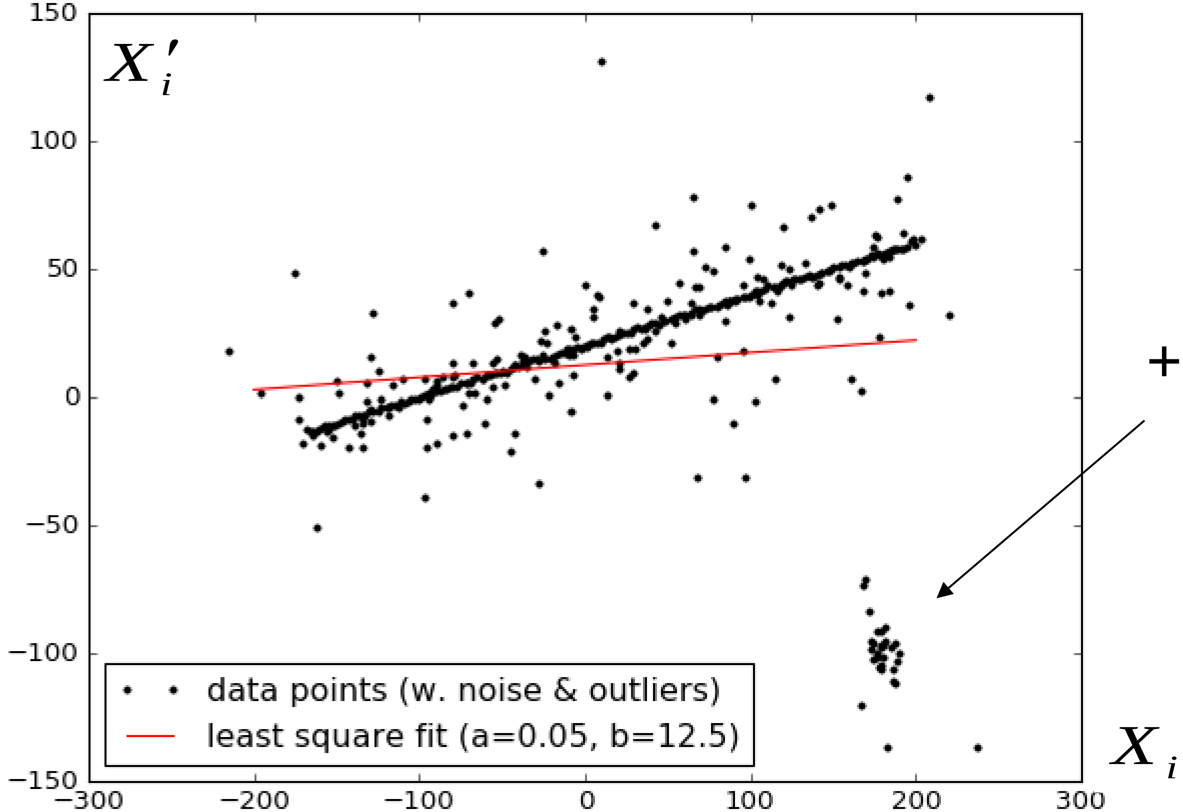
Check that V_i and $(w_i)^2$ are eigen vectors/values for matrix $A^T A$ (note: ellipsoid $x^T \cdot (A^T A) \cdot x = 1$ maps onto circle $u^T \cdot u = 1$)

$$A^T A \cdot V_i = VW^2 V^T \cdot V_i = VW^2 \cdot e_i = w_i^2 \cdot V_i$$

=> **can use eigen decomposition of $A^T A$ instead of SVD of A.**

Least squares fail in **presence of outliers**

Data generated as $X_i' = a\ X_i + b + \delta X_i$ for Normal noise δX_i



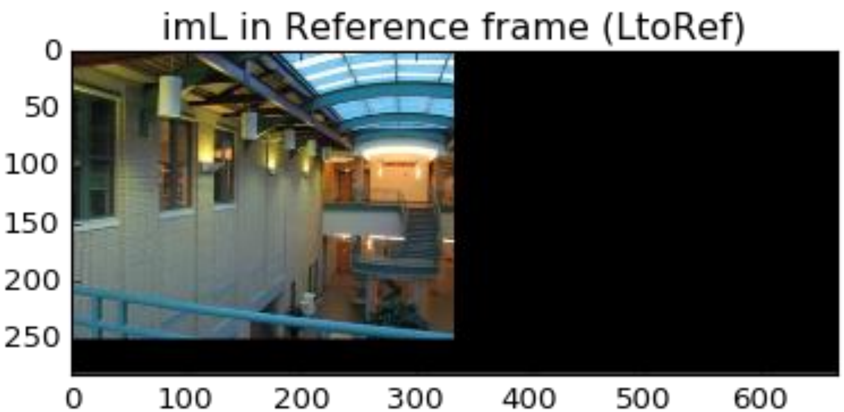
+ outliers

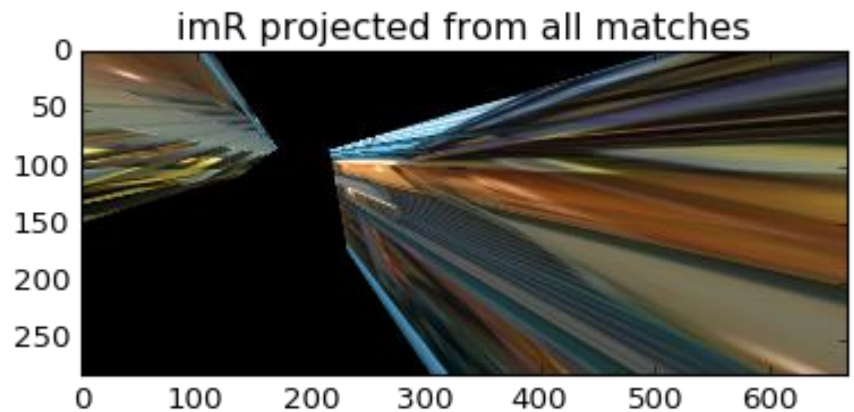
$$\min_{x=(a,b)^\top} \|A\,x - B\|^2 \longrightarrow x = A^{-1}B$$

least squares

Least squares fail in **presence of outliers**

In presence of outliers we need more robust methods (e.g. RANSAC, soon)

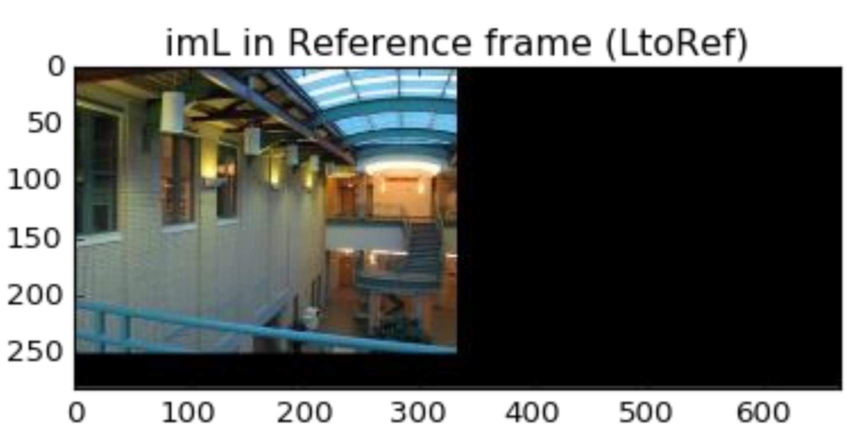


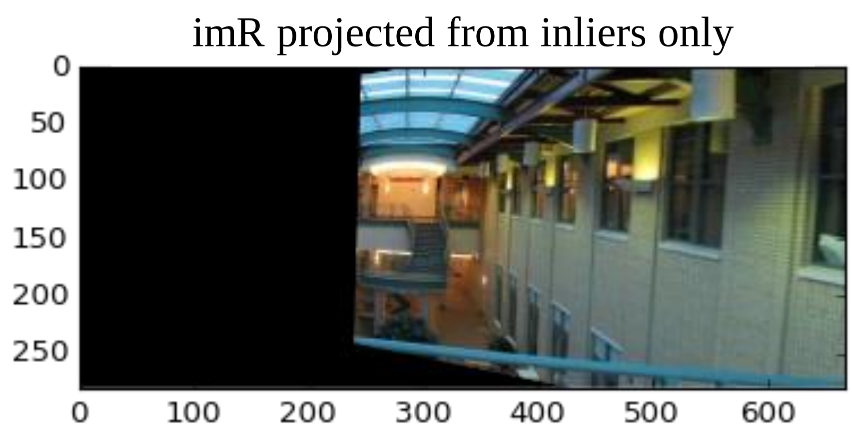


$$\min_{h:\|h\|=1} \|A\,\mathbf{h}\|^2 \quad \Longrightarrow \quad \mathbf{h} = V_i \quad \text{for } A = UWV^\top \text{ and } i = \arg\min w_i$$

homogeneous least squares

Least squares work if using "inliers" only (detecting these? – soon)





$$\min_{h:\|h\|=1} \|A\,\mathbf{h}\|^2 \quad \longrightarrow \quad \mathbf{h} = V_i$$

for $A = UWV^\top$
and $i = \arg\min w_i$

homogeneous least squares

Least squares work if using "inliers" only (detecting these? – soon)

larger errors in the area with no matches

"algebraic errors"

we minimized

$$ax + by + c - gxx' - hyx' - ix' \approx 0$$
$$dx + ey + f - gxy' - hyy' - iy' \approx 0$$

using ***"least squares"***

"geometric errors"

observable in the image

$$x' - \frac{ax + by + c}{gx + hy + i} \approx 0$$
$$y' - \frac{dx + ey + f}{gx + hy + i} \approx 0$$

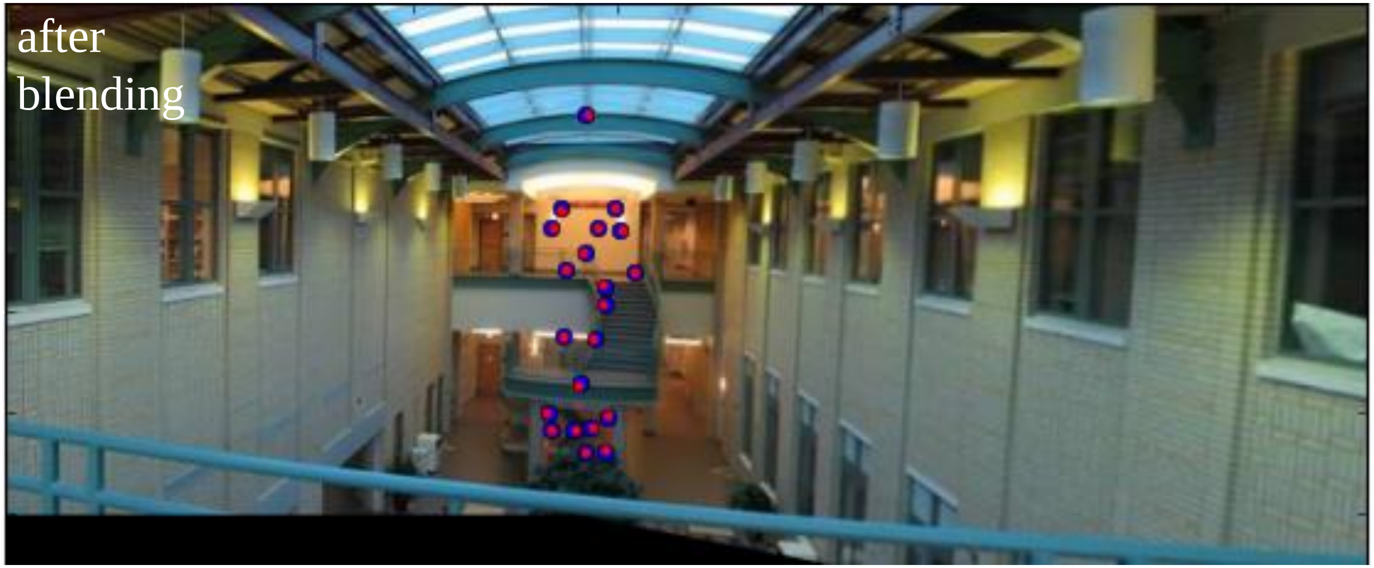
harder to minimize

"***non-linear least squares***"

"errors" among inliers $\|p' - Hp\|$

Did we actually minimize these errors?

Least squares work if using "inliers" only (detecting these? – soon)



Question: how to remove outliers automatically?

Model fitting robust to outliers

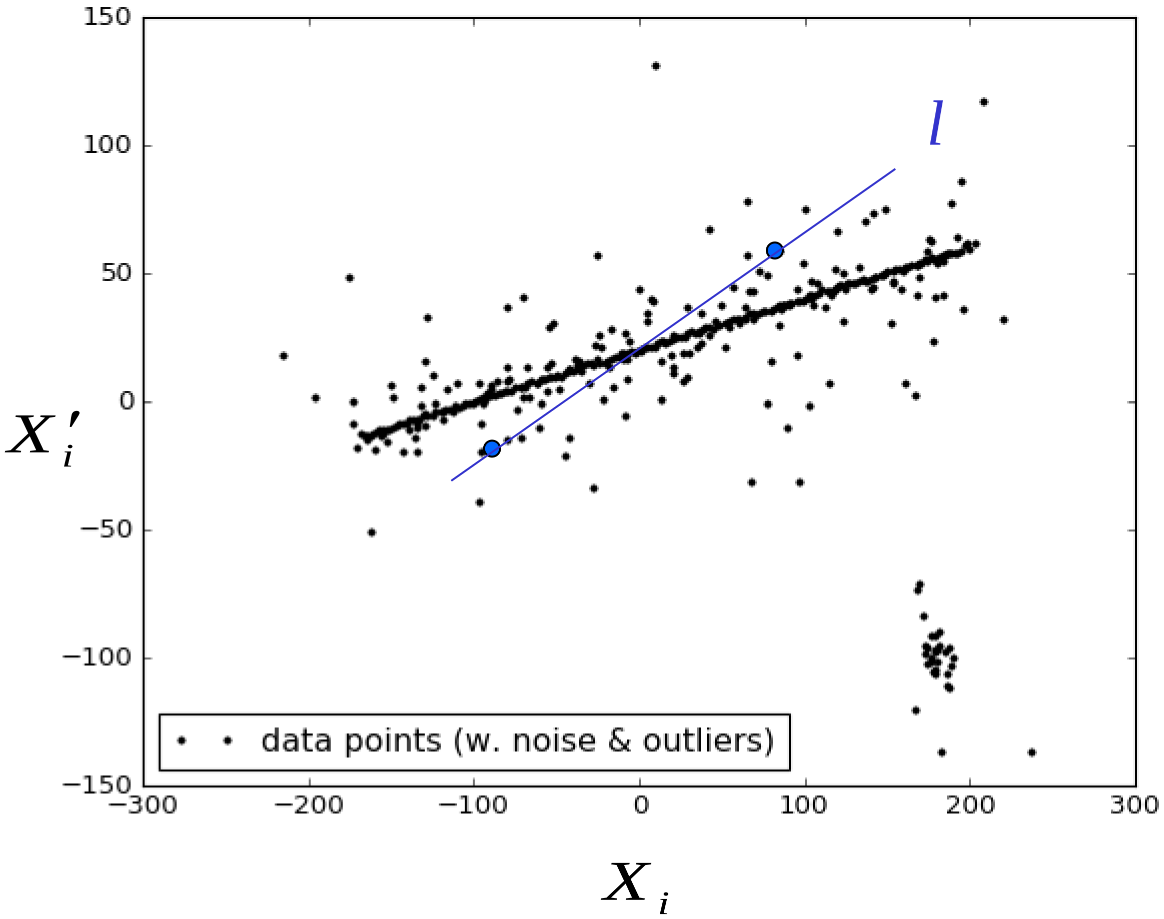
We need a method that can separate inliers from outlliers

RANSAC

random sampling consensus

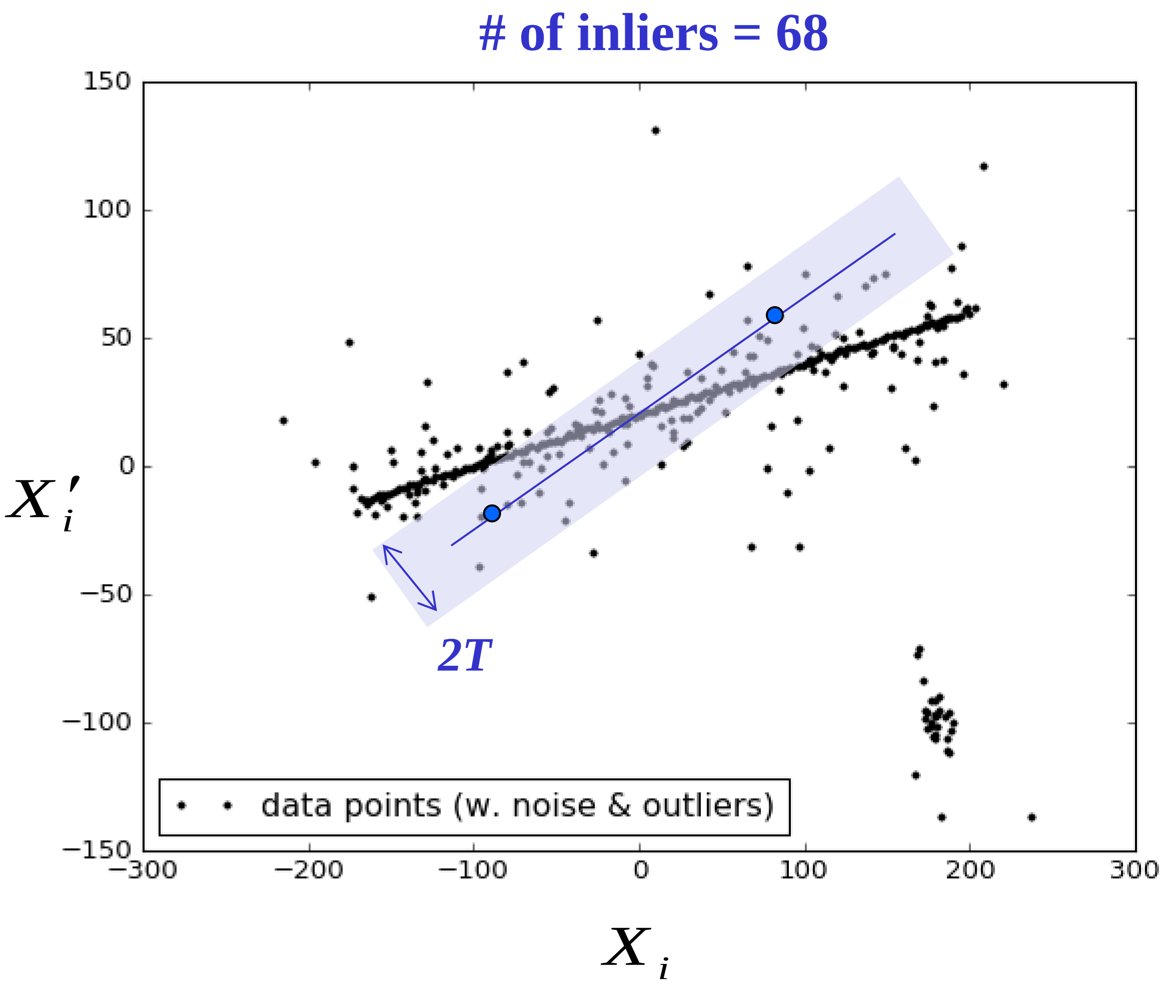
[Fischler and Bolles, 1981]

RANSAC (for line fitting example)



1. **randomly sample two points from the set, get a line**

RANSAC (for line fitting example)



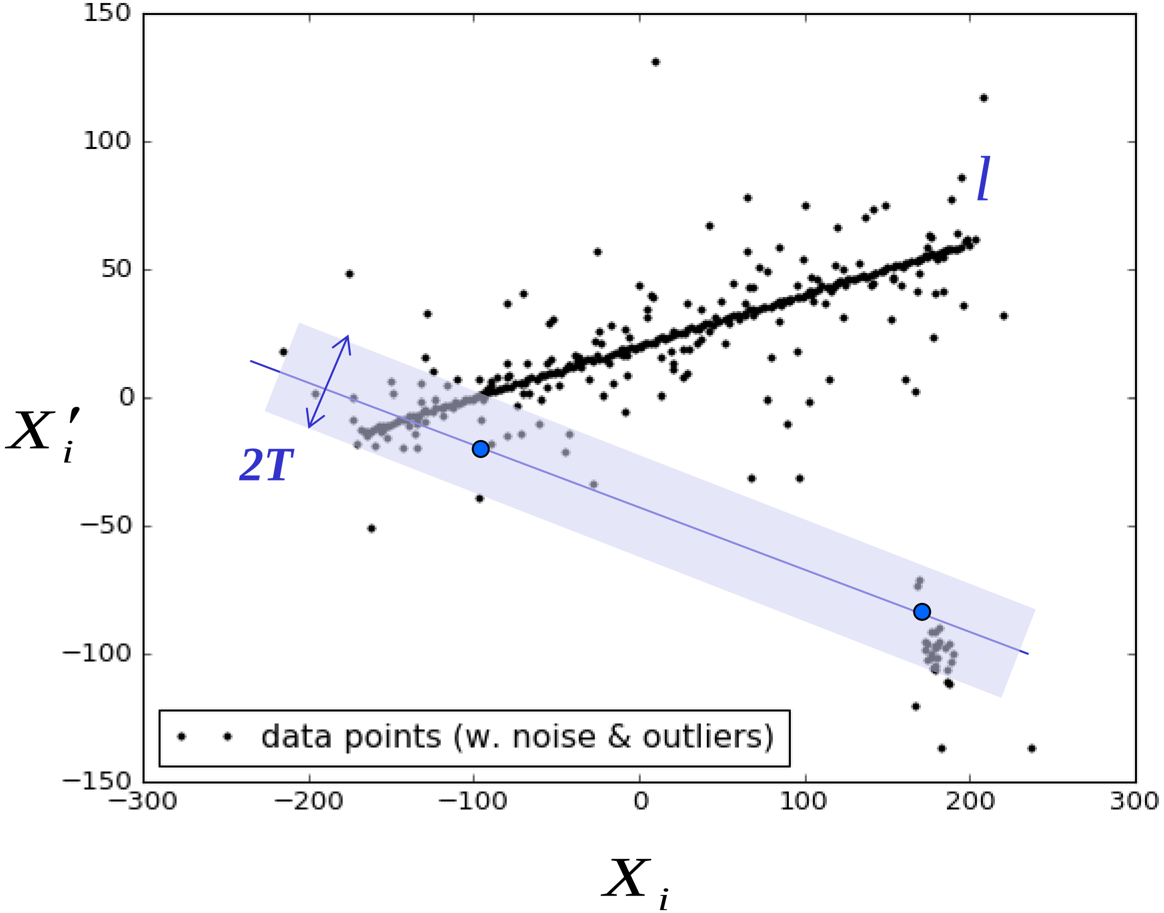
1. randomly sample **two points** from the set, get a **line**

2. count **inliers** p for **threshold** T

$$\| p - l \| \leq T$$

RANSAC (for line fitting example)

of inliers = 24

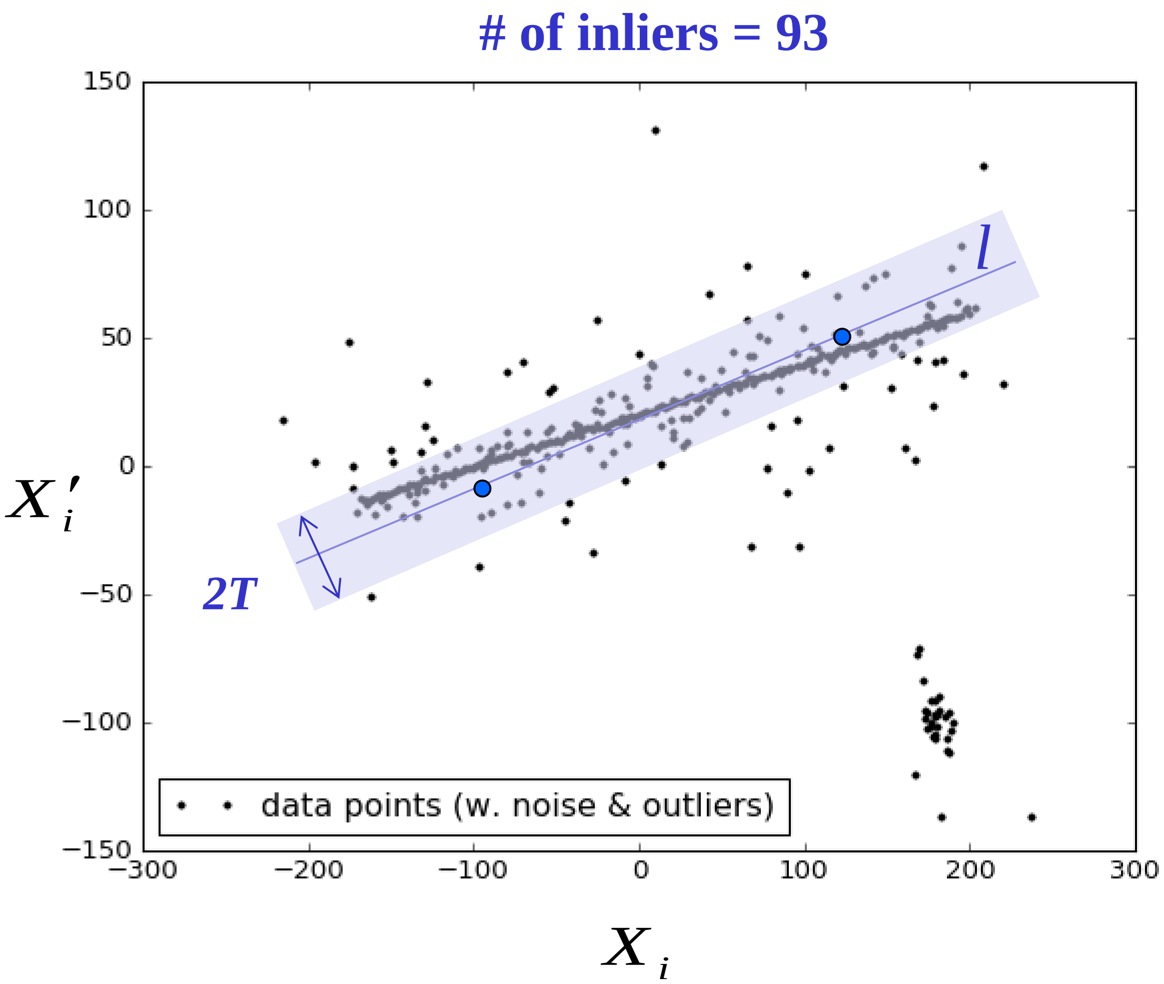


1. randomly sample **two points** from the set, get a **line**
2. count **inliers** p for **threshold** T

$$\| p - l \| \leq T$$

3. repeat

RANSAC (for line fitting example)



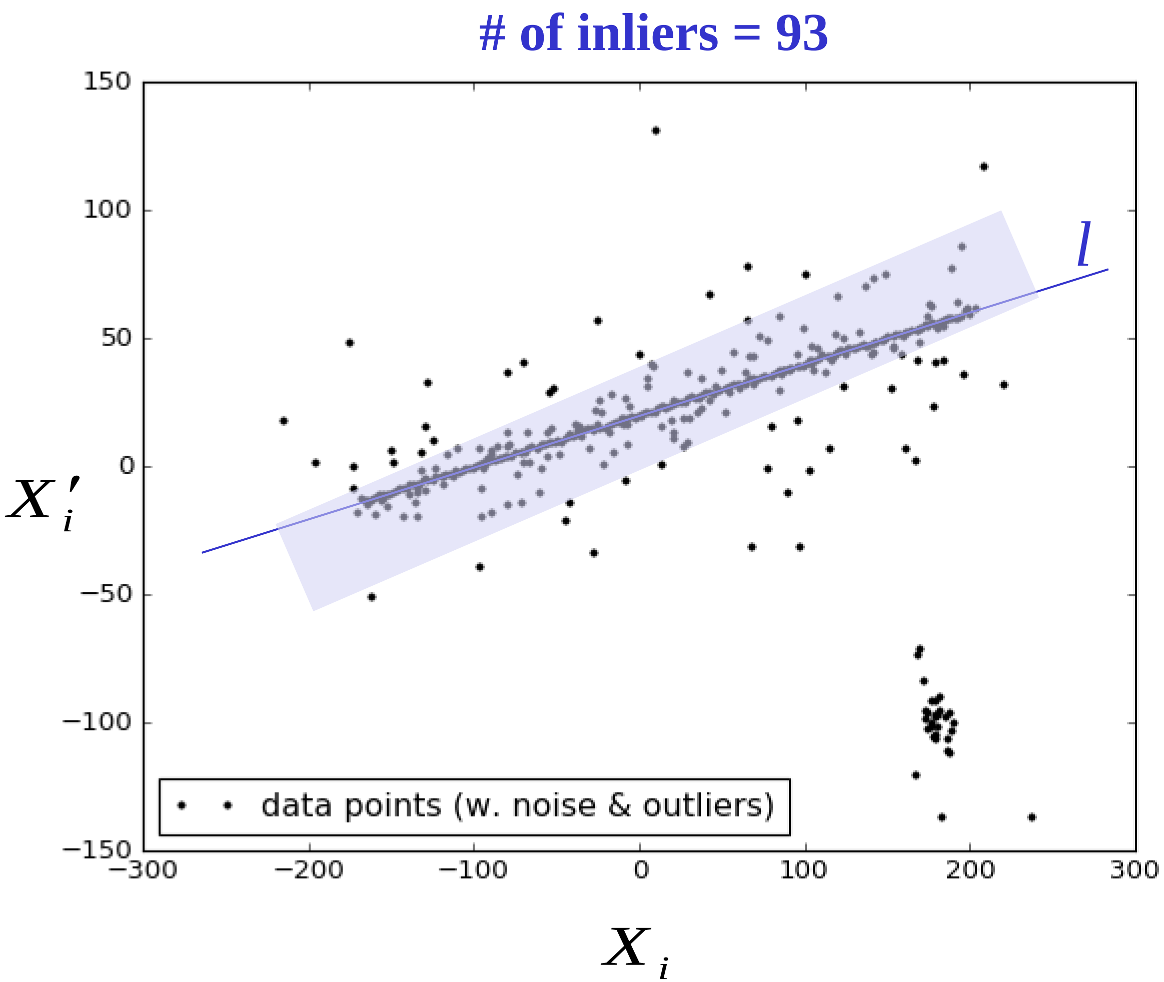
1. randomly sample **two points** from the set, get a **line**

2. count **inliers** p for <u>threshold</u> T

$$\| p - l \| \leq T$$

3. repeat N times and select model with **most inliers**

RANSAC (for line fitting example)



1. **randomly sample two points from the set, get a line**

2. **count inliers p for threshold T**

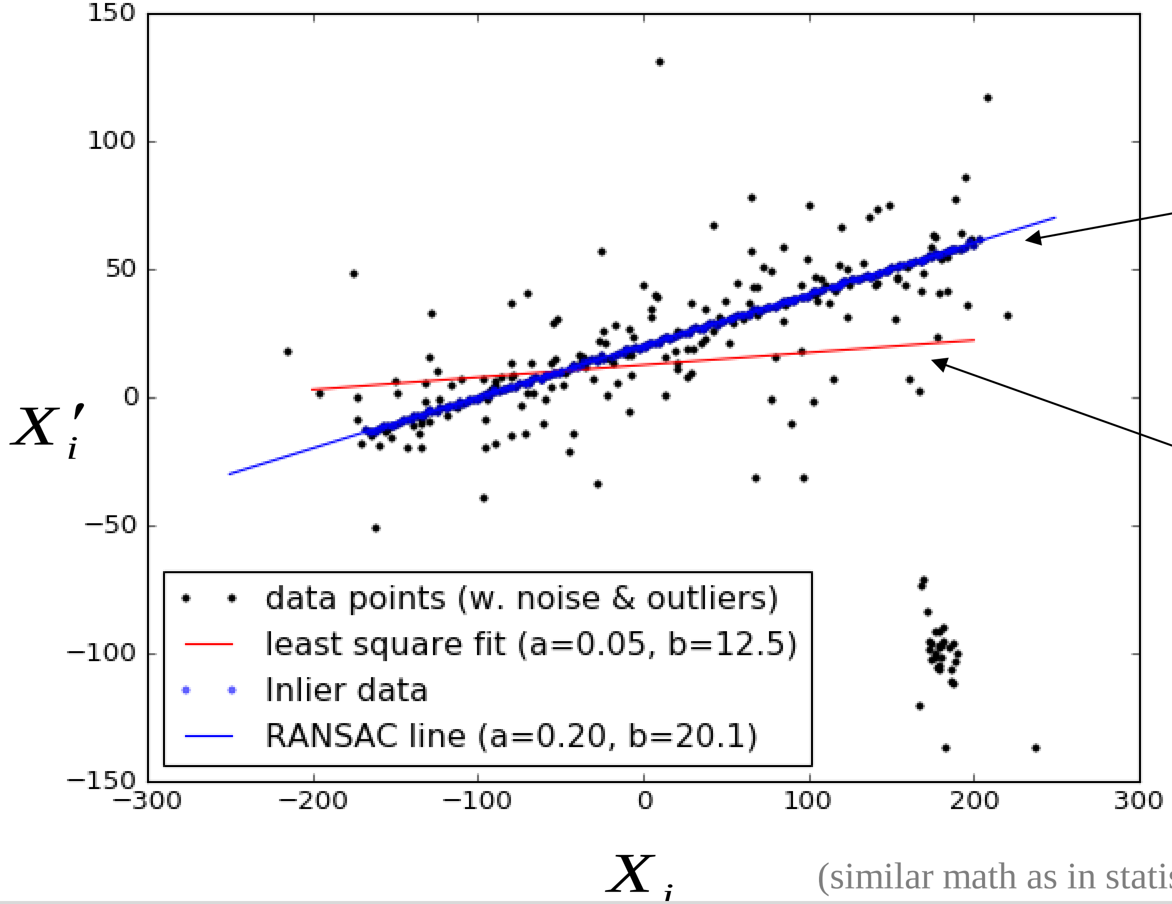
$$\| p - l \| \leq T$$

3. **repeat N times and select model with most inliers**

4. **Use *least squares* to fit a model (line) to this largest set of inliers**

Q: Assume know percentage of outliers in the data.
How many pairs of points (N) **should be sampled to have high confidence** (e.g. 95%) **that at least one pair consist of two inliers?** [Fischler and Bolles, 1981]

RANSAC (for line fitting example)



line model reliably estimated via RANSAC with only N=10 samples
(*least squares* fit to the largest set of inliers)

least squares **line model fit to all points**

(similar math as in statistical analysist of RANSAC success rate)

Birthday Paradox: in a group of random 23 people the probability that at least two have same birthday is 50.7%

RANSAC for robust homography fitting

only two differences: **1. need to randomly sample four pairs** (p, p')

the minimum number of matches to estimate a homography H

2. pair (p, p') **counts as an inlier for a given homography H if**

$$\| p' - Hp \| \leq T$$

RANSAC for robust homography fitting

Homography for corrupted <u>four</u> matches is likely to have only a few inliers (p, p')

(randomly sampled)

$$\| p' - Hp \| \leq T$$

RANSAC for robust homography fitting

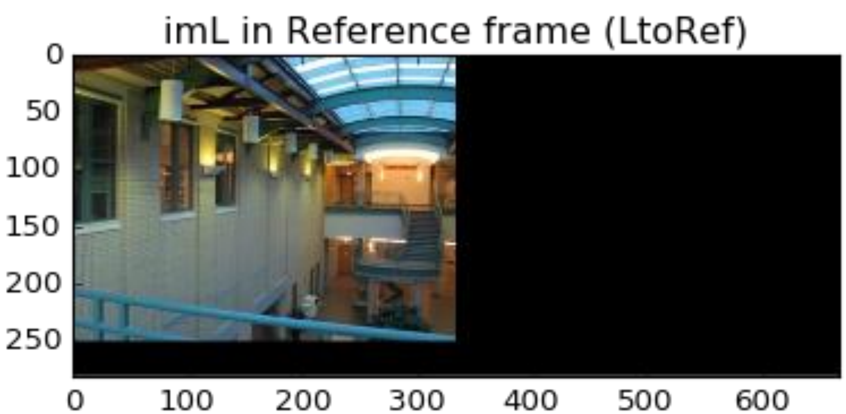
Homography for good <u>four</u> matches has 21 inliers (p, p')

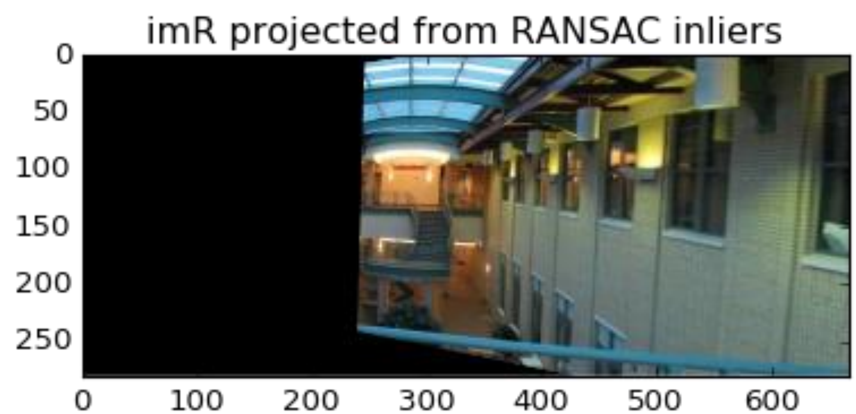
$$\| p' - Hp \| \leq T$$

(randomly sampled)

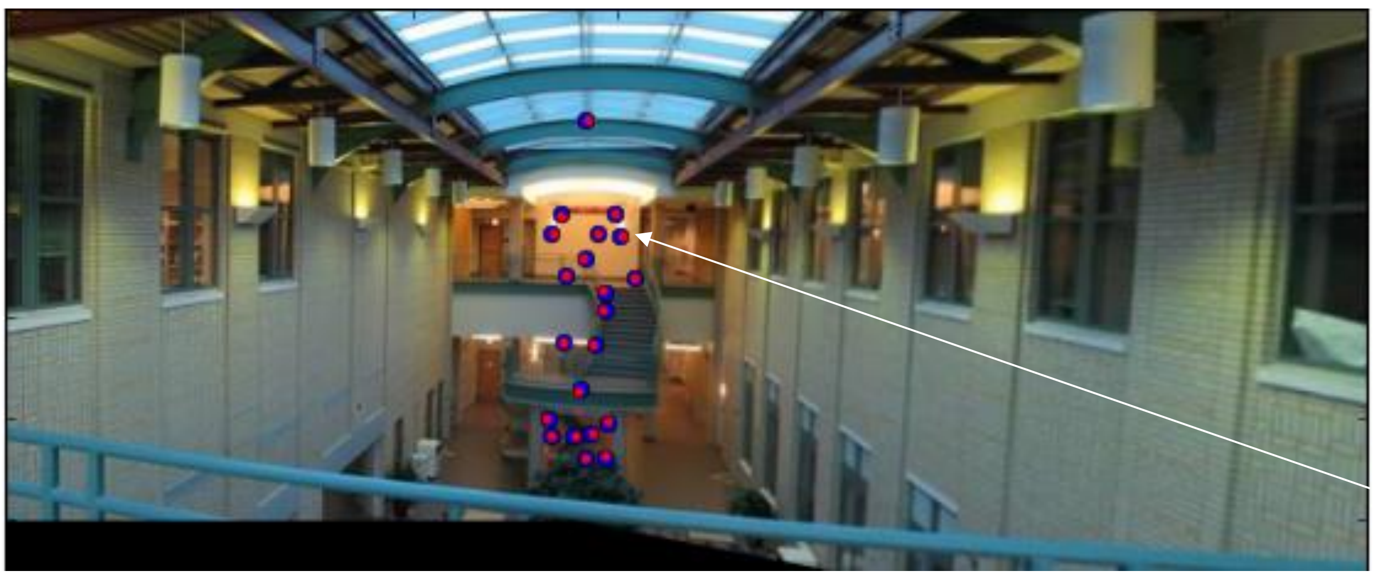
RANSAC for robust homography fitting

Inliers for the randomly sampled homography with the largest inlier set





RANSAC for robust homography fitting



matched inliers

$\| p' - Hp \|$

The final automatic panorama result

RANSAC for robust model fitting

In general (for other models):
always sample the smallest number
of points/matches needed to estimate a model

RANSAC loop:

1. Select <u>four</u> feature pairs (at random)
2. Compute homography H (exact)
3. Count *inliers* (p, p') where $\| p' - Hp \| \le T$

   e.g. geometric errors

4. Iterate N times (steps 1-3). Keep the largest set of inliers.
5. Re-compute <u>least-squares</u> H estimate on all of the inliers

   e.g. for algebraic errors
   (for simplicity)

Other examples of geometric model fitting

images from **different** view points (optical centers)

Merton College III data

from Oxford's Visual Geometry Group

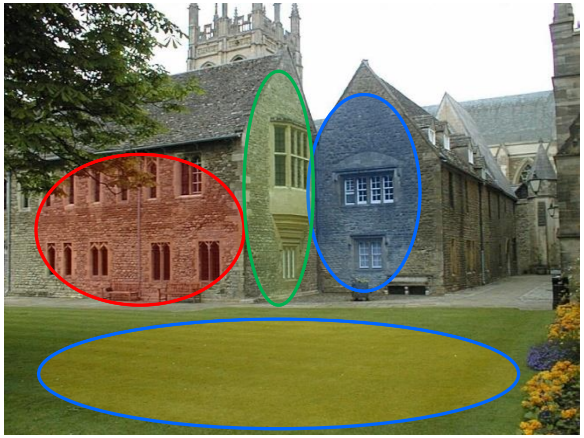
http://www.robots.ox.ac.uk/~vgg/data/data-mview.html

Question: Is it possible to create a panorama from these images?
(or, is there a *homography* that can match overlap in these images?)

Can a *homography* map/warp **a part** of the left image onto **a part** of the right image?

Other examples of geometric model fitting

images from **<u>different</u>** view points (optical centers)

Merton College III data
from Oxford's Visual Geometry Group
http://www.robots.ox.ac.uk/~vgg/data/data-mview.html

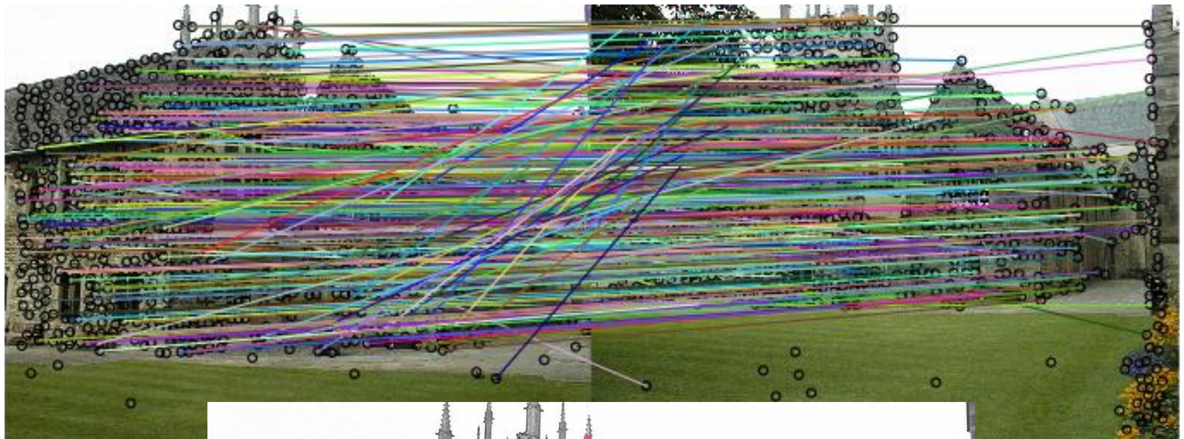
There should be a *homography* for each plane in the scene (Why?)

Question: How can we detect such *homographies*?

What do these multiple *homographies* give us?

Other examples of geometric model fitting

matched features (p,p'), as earlier



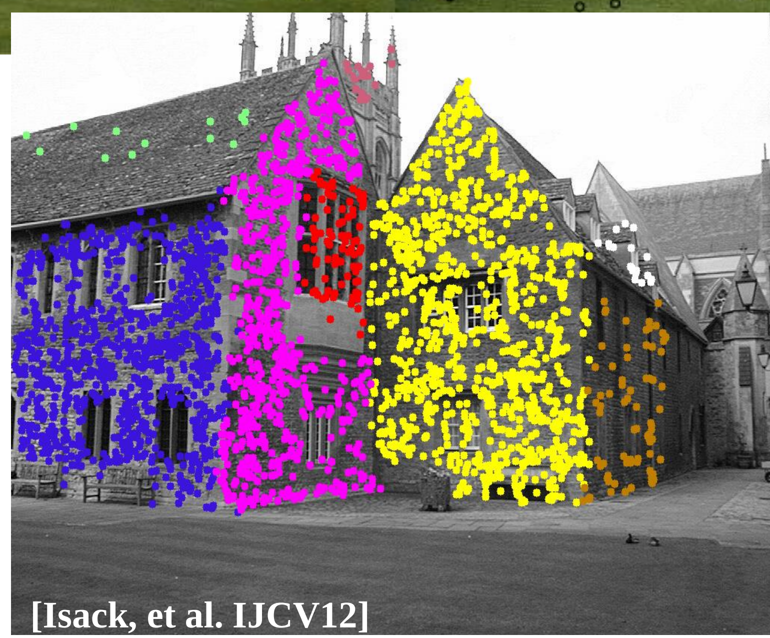
NOTE:
good matches can be used for reconstructing 3D points if camera positions & orientations are known:

$$(\boldsymbol{p},\boldsymbol{p}') \rightarrow \boldsymbol{X_p} \in \boldsymbol{R^3}$$

(Triangulation, see next topic)

1. Allow to remove bad matches (outliers)

2. Correspondences can be estimated with subpixel precision $(\boldsymbol{p},\boldsymbol{p}') \rightarrow (\boldsymbol{p},\boldsymbol{Hp})$



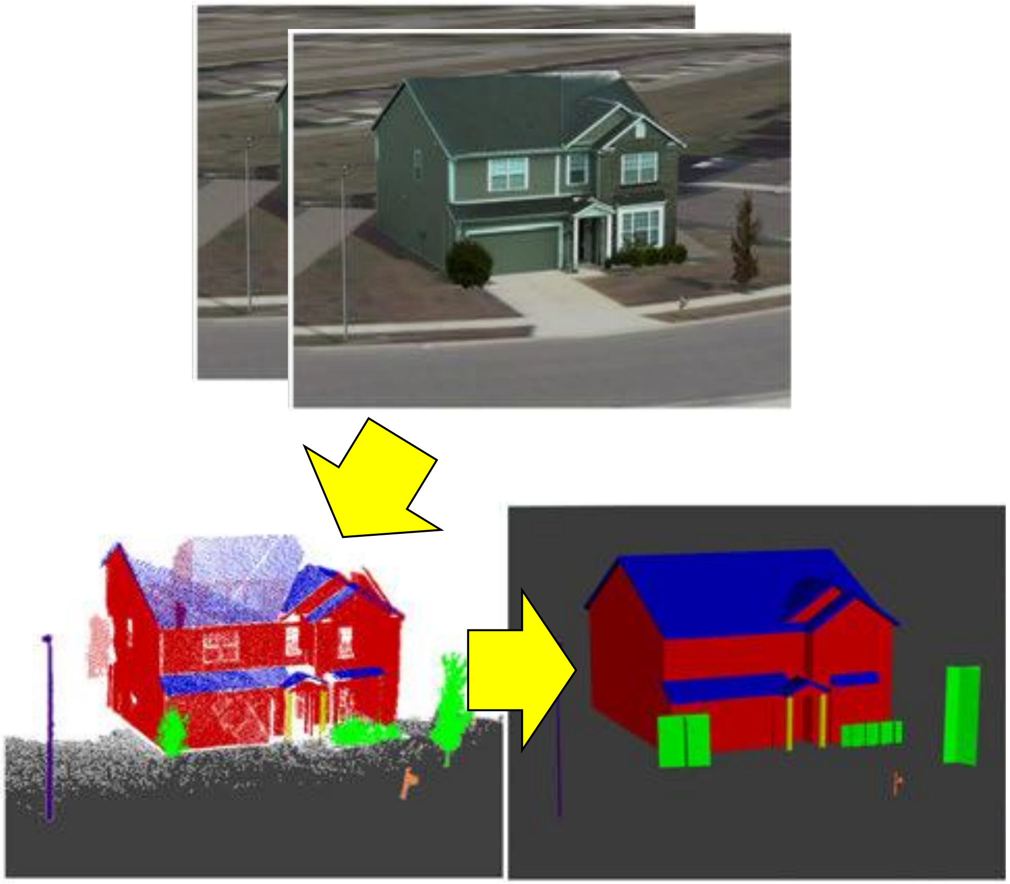
3. Inliers allow to segment planar regions

4. Segmentation allows to extend correspondence from inliers to any point $\boldsymbol{p}$ inside segment: $(\boldsymbol{p},\boldsymbol{Hp})$

5. Piece-wise planar 3D scene reconstruction

What do these multiple *homographies* give us?

Other examples of geometric model fitting



could be used for

3D piecewise planar reconstruction

from multiple views

Other K-model estimation problems in vision



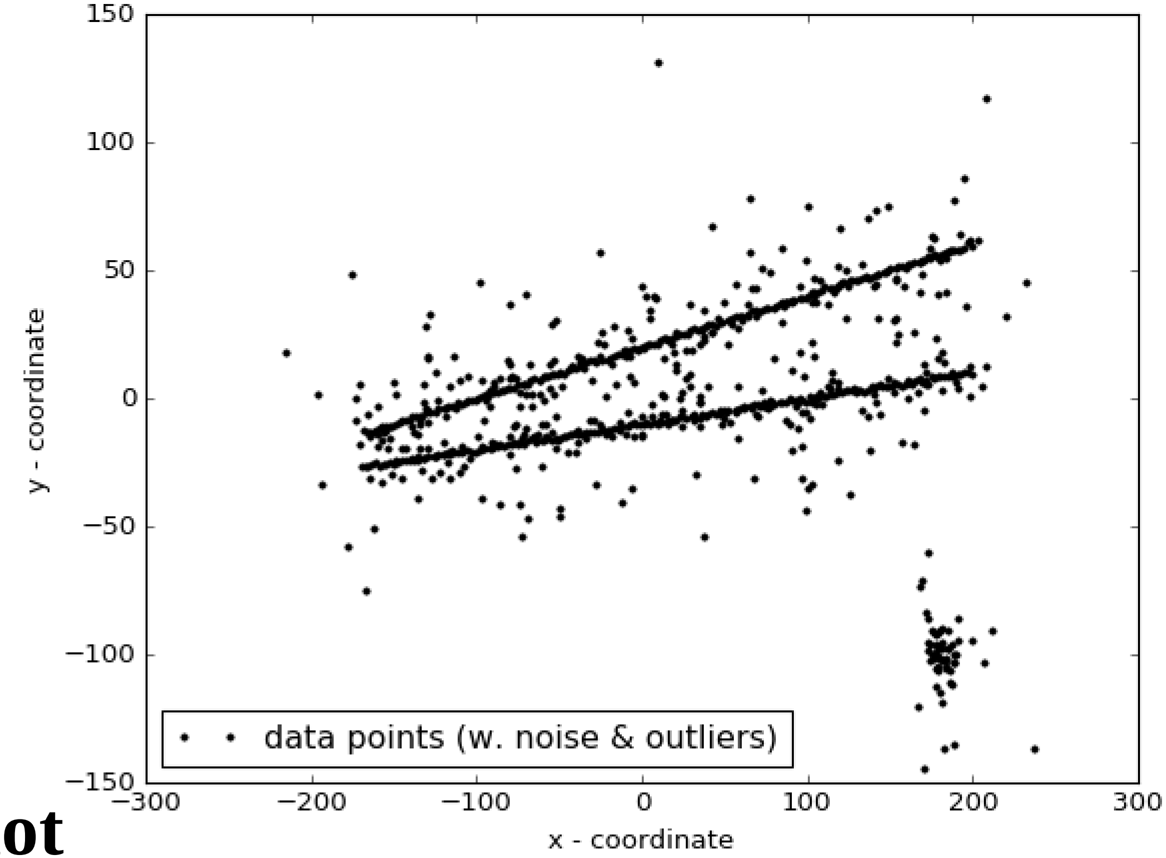

- Geometric structures
  - lines, planes, rigid motions (fundamental matrices), flex. models (pose)
- Stereo or multi-view reconstruction — Topics 7,8,12
  - projection matrices (camera positions), points and surfaces in 3D
- Object models (segmentation, classification) — Topics 9,10,11,12
  - appearance models, boundary surface models, (linear & non-linear) discriminative models

Computer vision - can answer *what* and *where*

based on “model” estimation

model formulation + loss function (data & regularization) + optimization

So, how do we fit multiple geometric models?

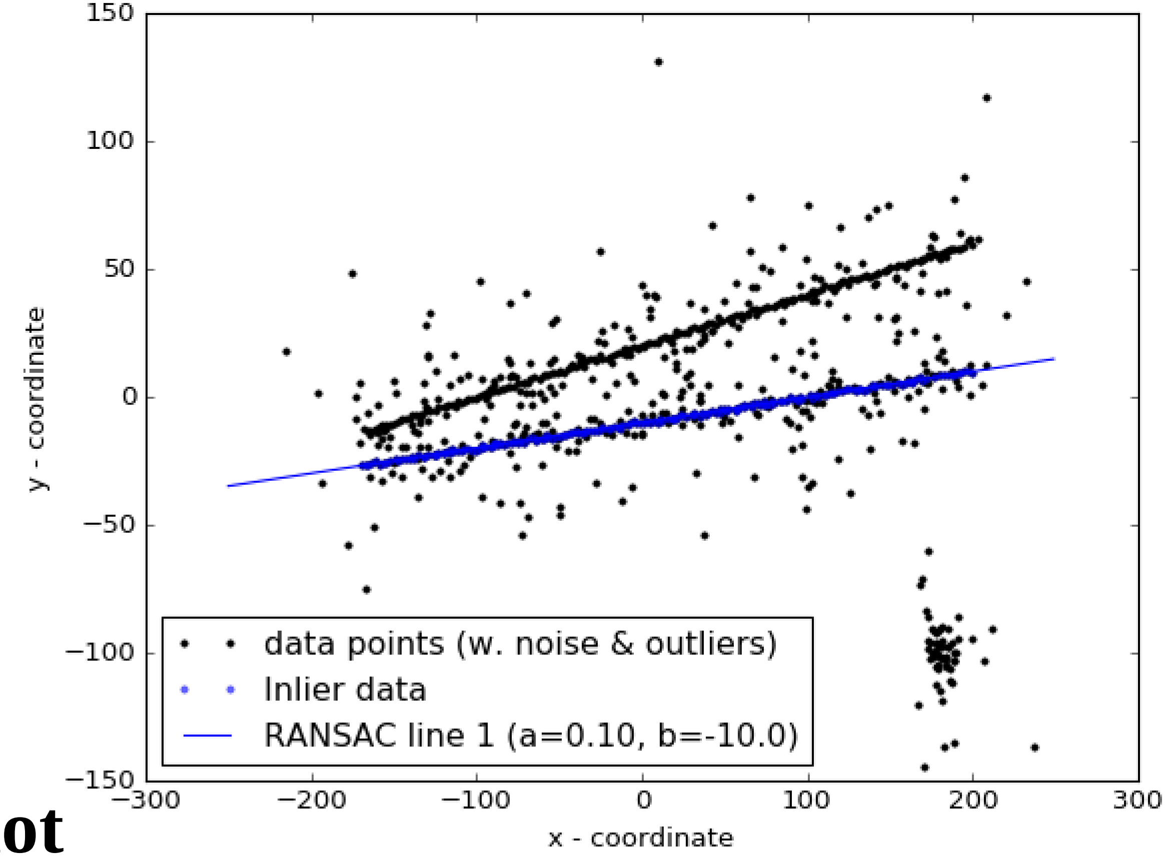


Why not

RANSAC

again?

So, how do we fit multiple geometric models?

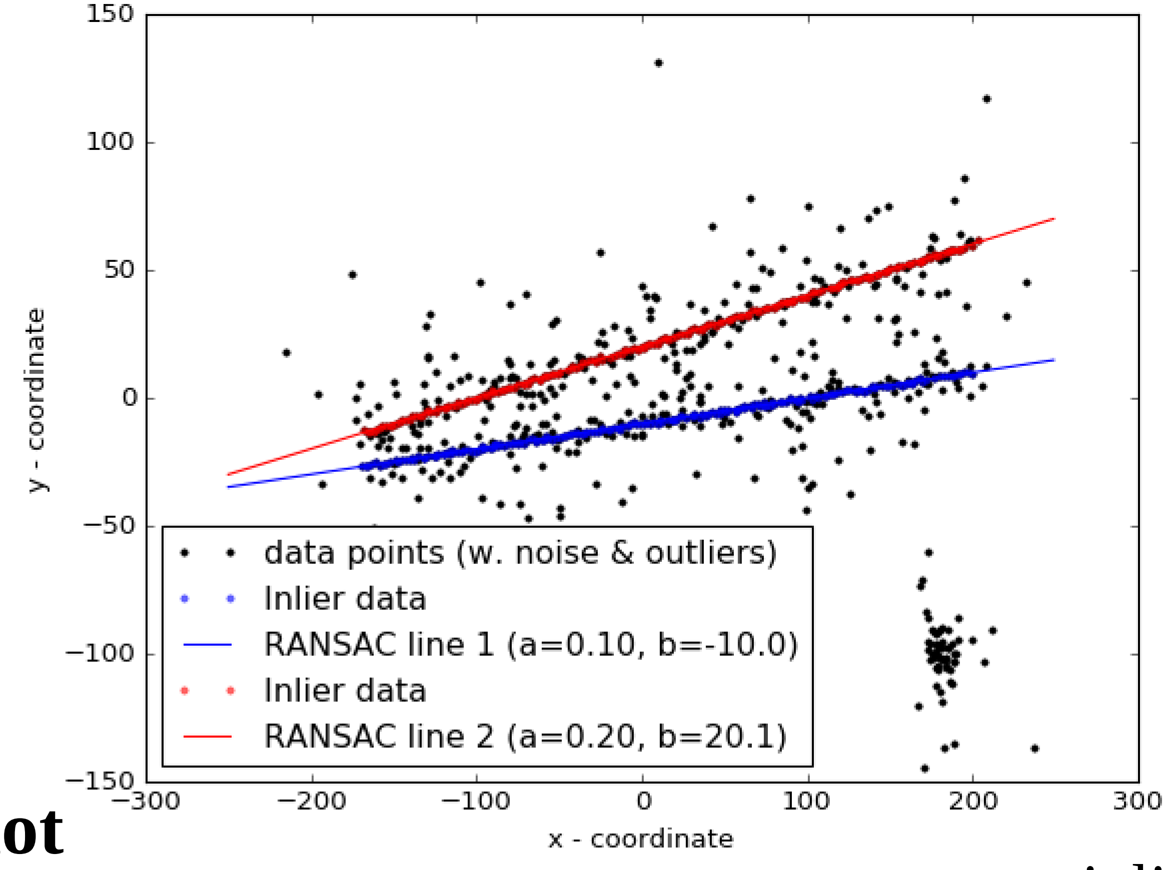


Why not

RANSAC

again?

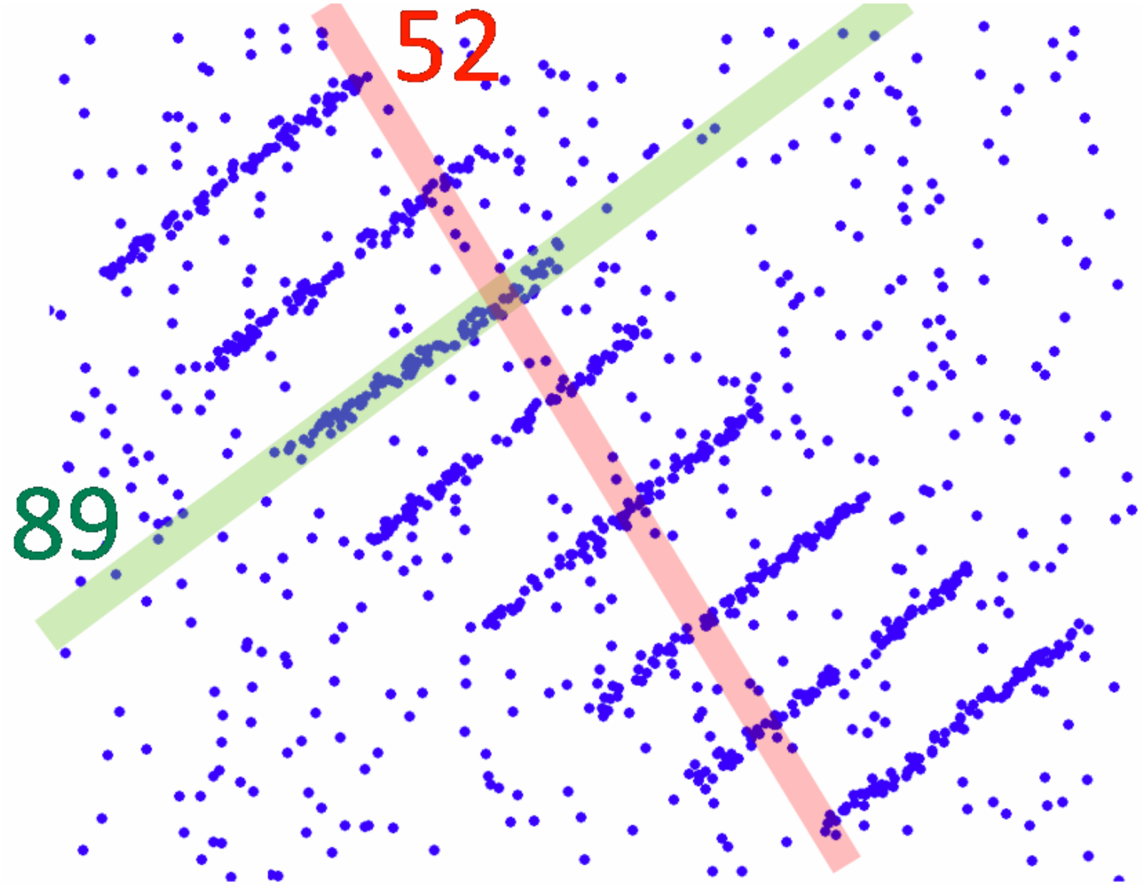
So, how do we fit multiple geometric models?



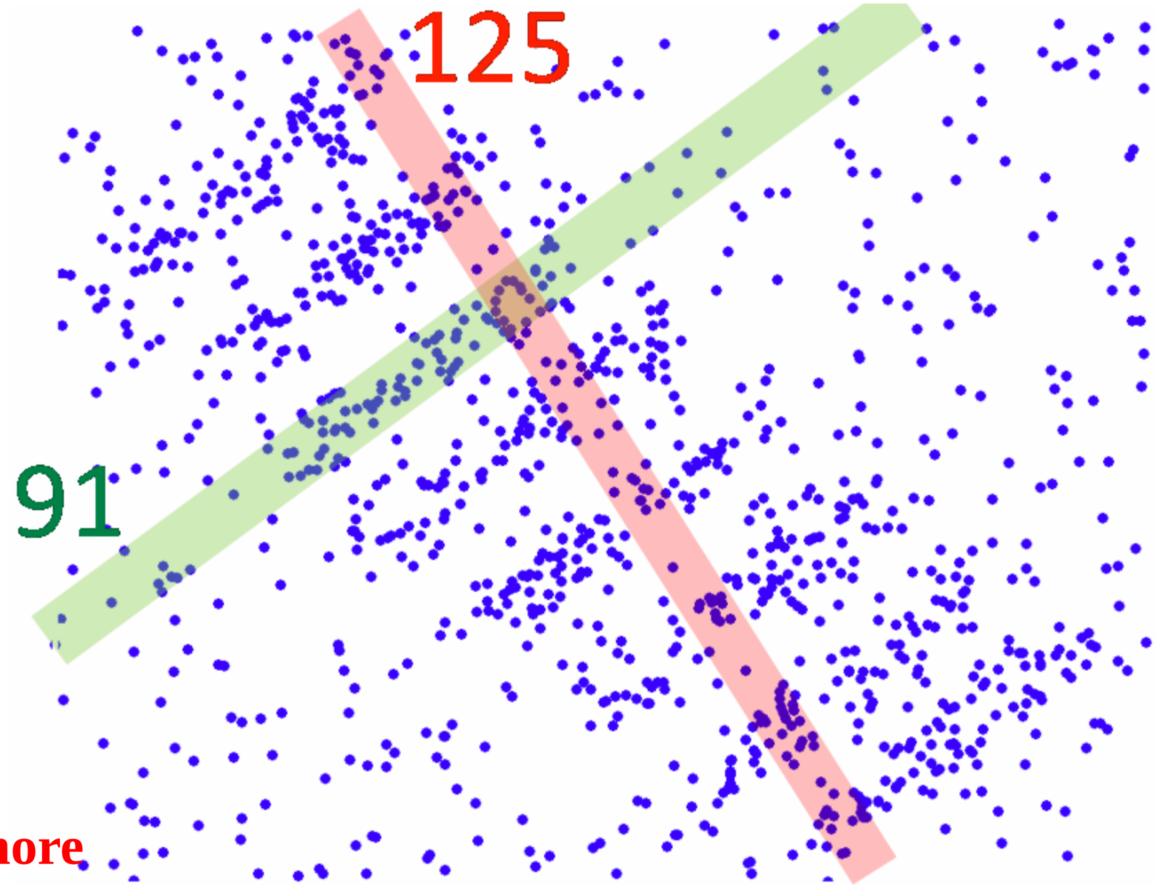
Why not
RANSAC
again?

remove inliers for line 1 and use RANSAC again (**sequential RANSAC**)

Multiple models and many outliers



Multiple models and many outliers



Higher noise

We need a more principled (objective) approach evaluating the whole solution

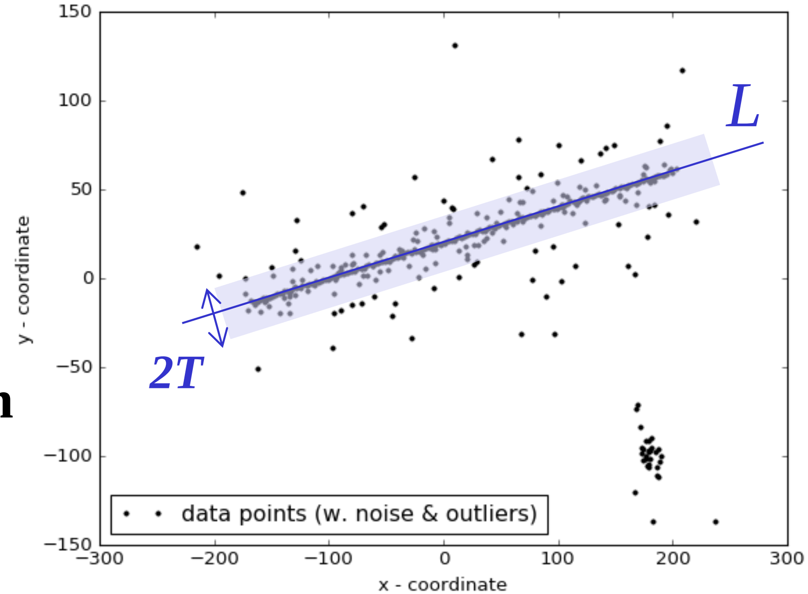
(greedy) maximization of inliers often fails for
multiple models + many outliers + large noise

RANSAC as optimization (of certain objective)

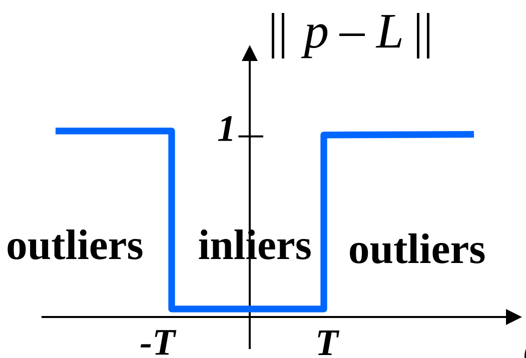
treat $\|p - L\|$ as an "operator" $\| \cdot - \cdot \|$ representing some error penalty function *Loss (distance (p , L))*

$$E(L) = \sum_p \|p - L\|$$

- find optimal label *L* (line)
minimizing loss function *E(L)*



RANSAC minimizes error (loss) function
$\| p - L \|$ counting outliers
(equivalent to maximizing inliers)



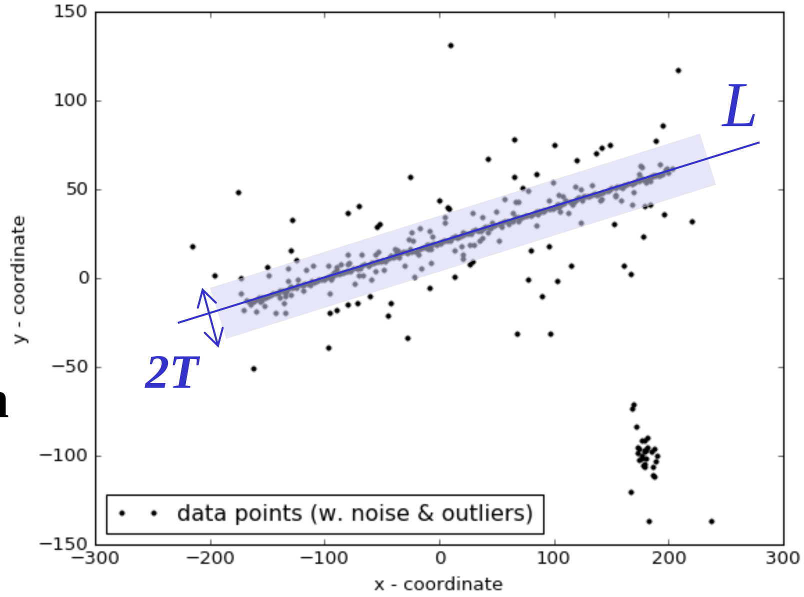
$dist(p, L)$ - **errors** (distances between point *p* and line *L*)

RANSAC as optimization (of certain objective)

$$E(L) = \sum_p ||p - L||$$

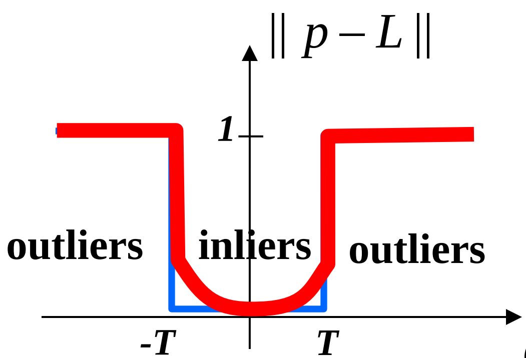
treat $||p - L||$ as an "operator" $|| \cdot - \cdot ||$ representing some error penalty function *Loss* (*distance* (*p* , *L*))

- **find optimal label** L (line) **minimizing loss function** $E(L)$



RANSAC minimizes error (loss) function $|| p - L||$ counting outliers

(equivalent to maximizing inliers)

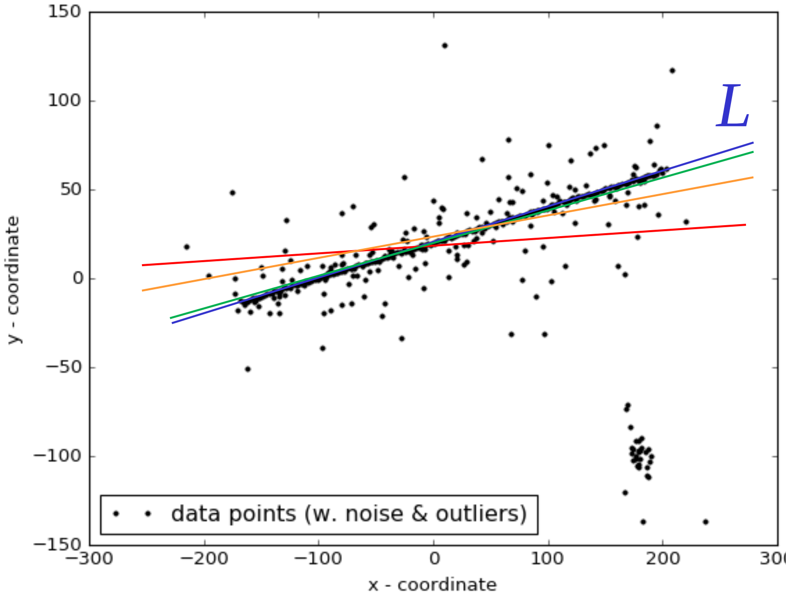


$dist(p, L)$ - **errors** (distances between point p and line L)

The use of least squares among inliers can be interpreted as quadratic loss for *errors* < *T*

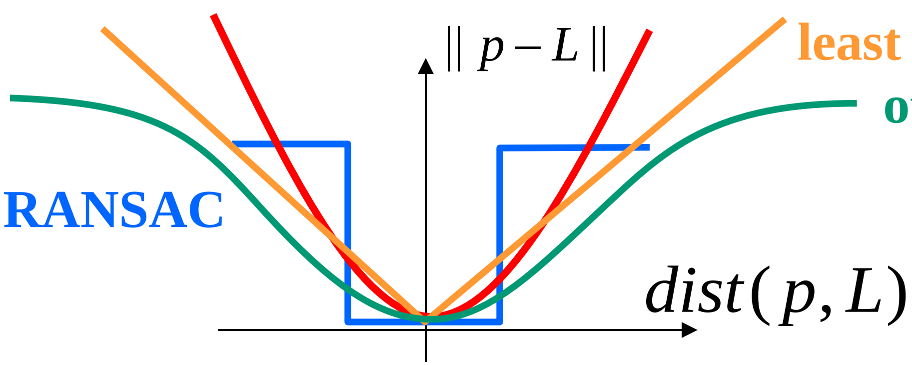
General error functions (overview)

$$E(L) \;=\; \sum_p ||p - L||$$



Can use different losses
***i.e.* error functions $||\, p - L \,||$**

least squares



least absolute deviations

other robust error measure

robust error/loss functions have bounded response effectively ignoring larger errors (outliers)

non convex => harder to optimize

e.g. see *iterative reweighted least squares* (IRLS)

Optimization-based multi-model fitting

Error / loss function

distances to lines

$$E(\mathbf{L}) \;=\; \sum_p ||p - L_p||$$

<u>If multiple models</u>

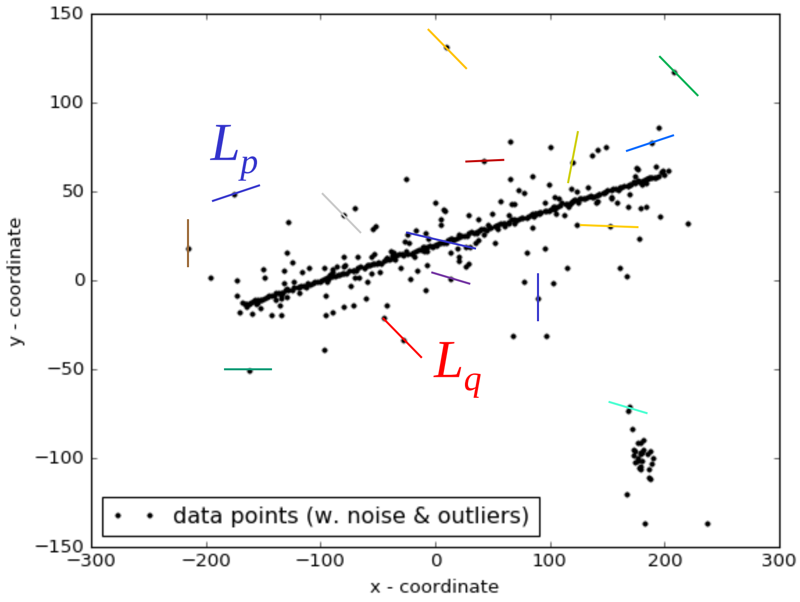
- assign different models (labels L_p) to every point p

- <u>find optimal labeling</u>

$\mathbf{L} = \{ L_1 , L_2 , \ldots , L_n \}$

$$\forall p, \quad L_p \in \Lambda$$

set of used models (lines)



too "complex" - too many lines

(perfect line model for every point)

need a "simpler" solution, i.e.

<u>few</u> lines "explaining" all data

Optimization-based multi-model fitting

Error / loss function — *distances to lines*

Penalize the number of models *(solution "complexity")*

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

our first example of ***regularization***

[Delong, Isack, et al. 2012]

<u>If multiple models</u>

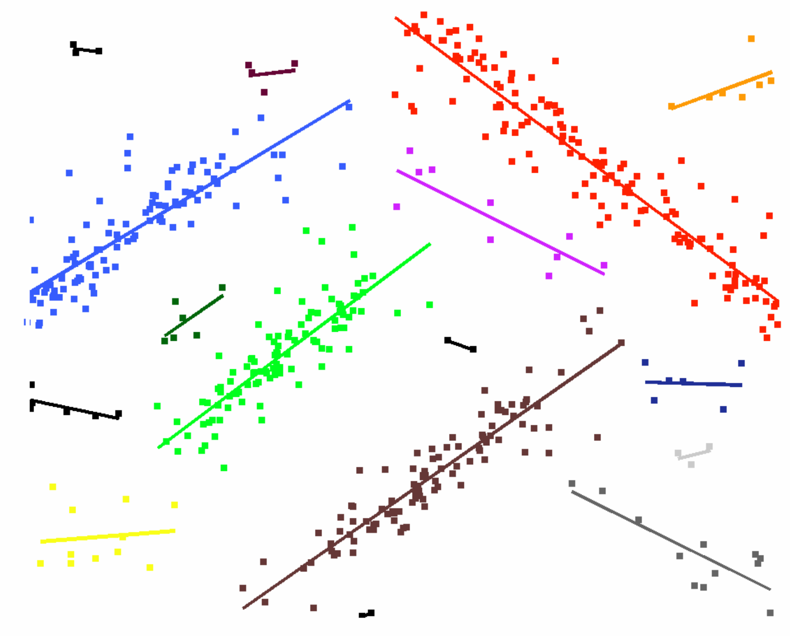
- assign different models (labels L_p) to every point p

- <u>find optimal labeling</u>

$$\mathbf{L} = \{ L_1 , L_2 , \ldots , L_n \}$$

$$\forall p, \quad L_p \in \Lambda$$

set of used models (lines)



<u>few</u> lines "explaining" all data

NOTE: loss above relates to *K-means* (topic 9) with <u>sparsity regularization prior</u> (a.k.a. AIC/BIC)

Optimization-based multi-model fitting

Error / loss function — *distances to lines*

Penalize the number of models — *(solution "complexity")*

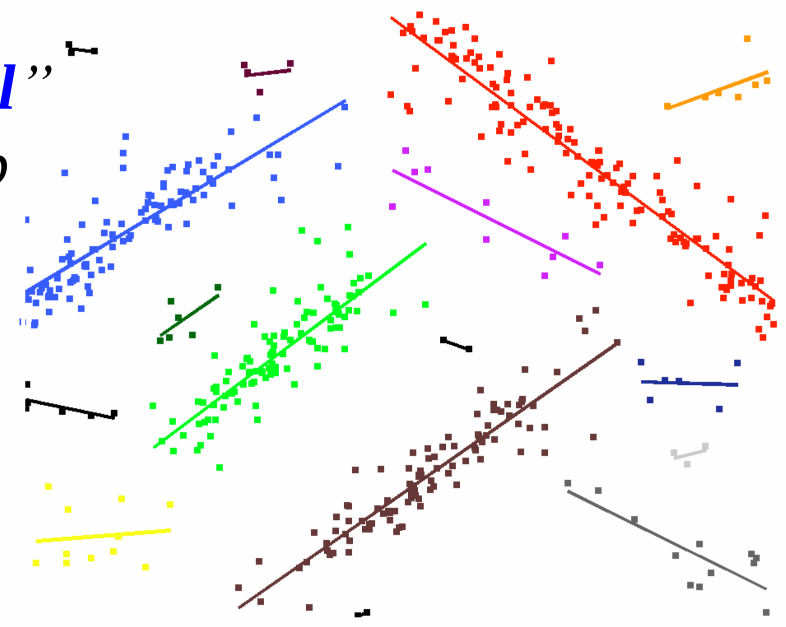
$$E(\mathbf{L},\Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

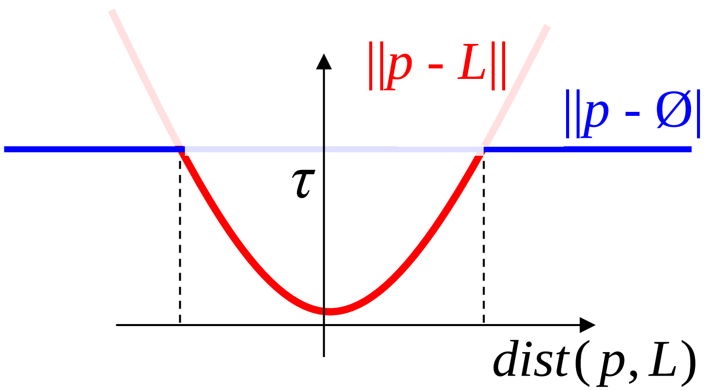
our first example of ***regularization***

[Delong, Isack, et al. 2012]

Extra trick: add to Λ one special model Ø representing "***outlier model***" with fixed cost $||p$ - Ø$|| = \tau$ for any p

Then any point p prefers to be assigned "outlier model" Ø if its closest line L has $||p - L|| > \tau$





Optimization-based multi-model fitting

Error / loss function — *distances to lines*

Penalize the number of models *(solution "complexity")*

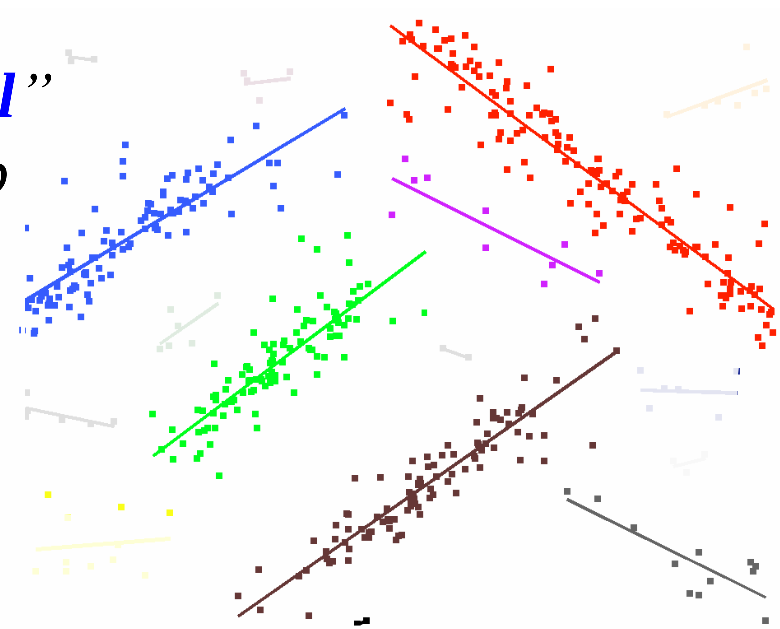
$$E(\mathbf{L},\Lambda) = \sum_p ||p - L_p|| \;+\; \gamma \cdot |\Lambda|$$

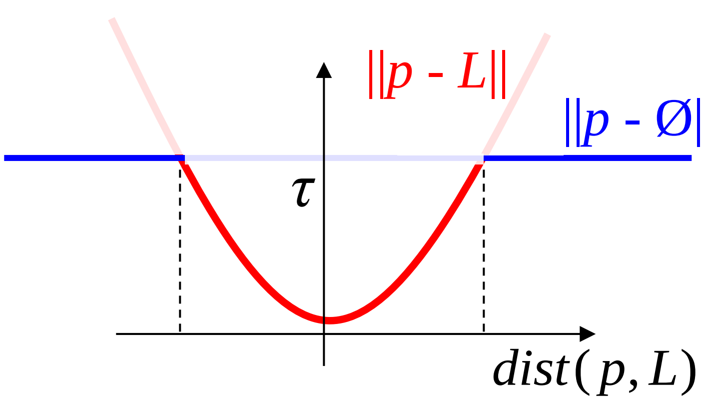
our first example of ***regularization***

[Delong, Isack, et al. 2012**]**

Extra trick: add to Λ one special model Ø representing "***outlier model***" with fixed cost ||*p* - Ø|| = *τ* for any *p*

Then any point *p* prefers to be assigned "outlier model" Ø if its closest line *L* has ||*p* - *L*|| > *τ*

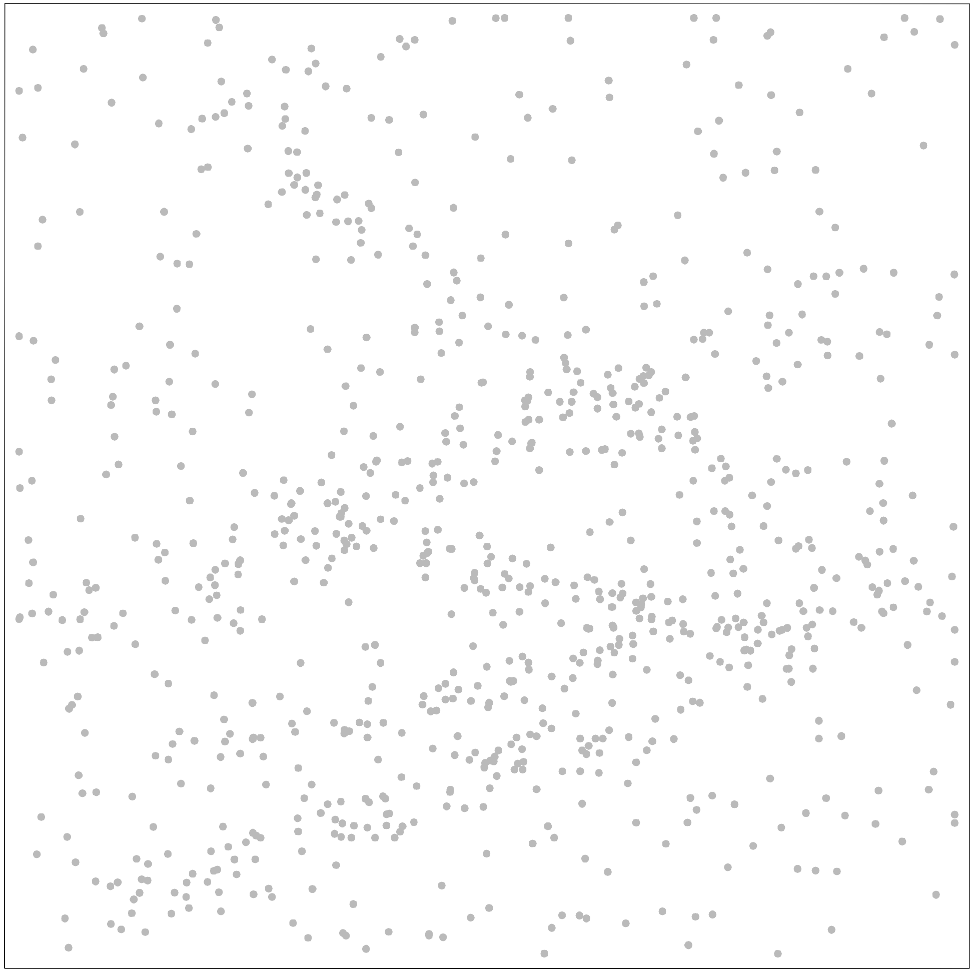




Also, *weak* models with only a few inliers $N < \frac{\gamma}{\tau}$ are not worth having. Its inliers should be assigned Ø. Why?

Optimization problem

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

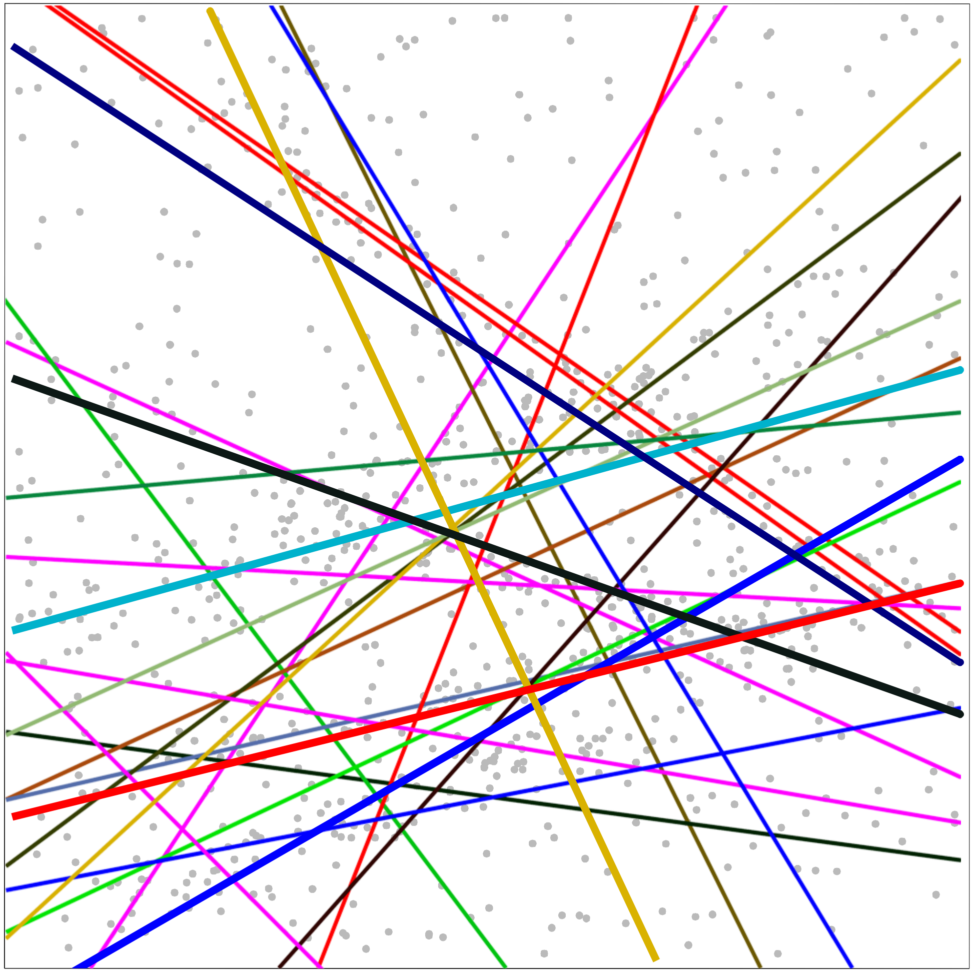


data points

Optimization problem

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

1. Initialization of Λ:
randomly sample
K lines from points
(some very large K)



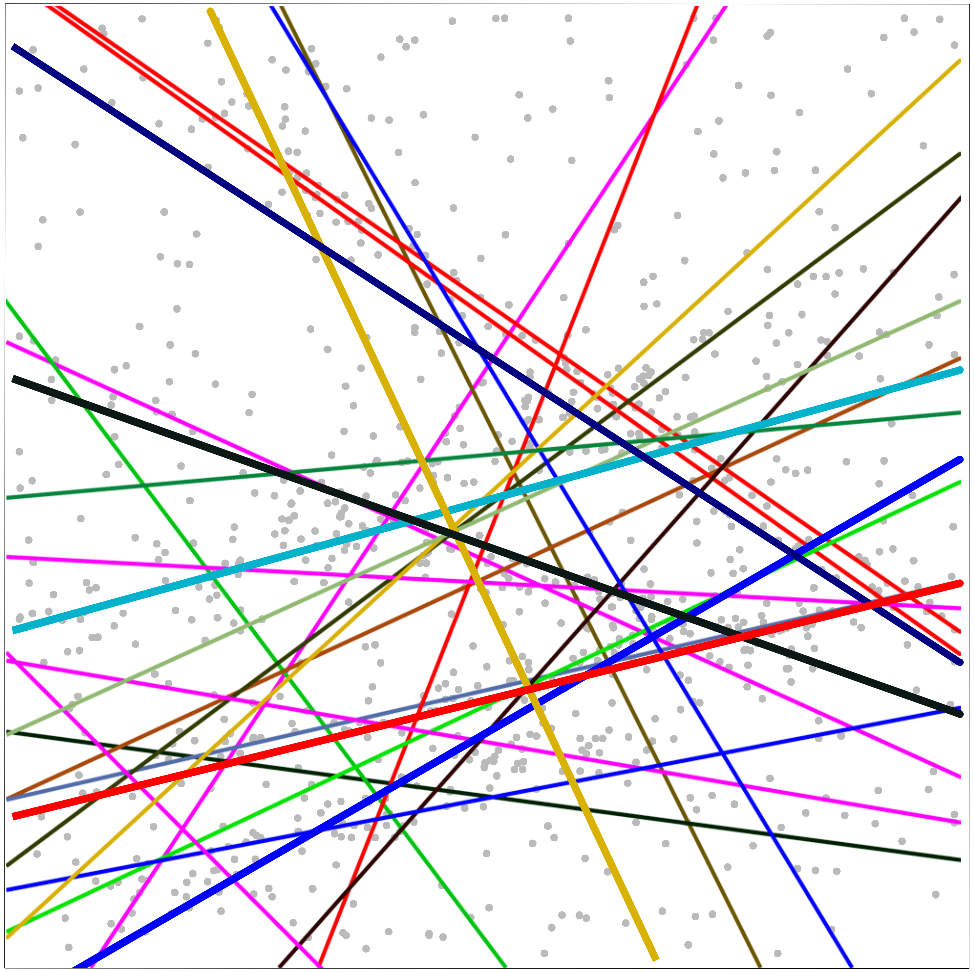
data points + randomly sampled lines

Optimization problem

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

1. Initialization of Λ:
randomly sample
K lines from points
(some very large K)

2a. Assign each point to
model in **Λ** with lowest || ||



data points + randomly sampled lines

Optimization problem

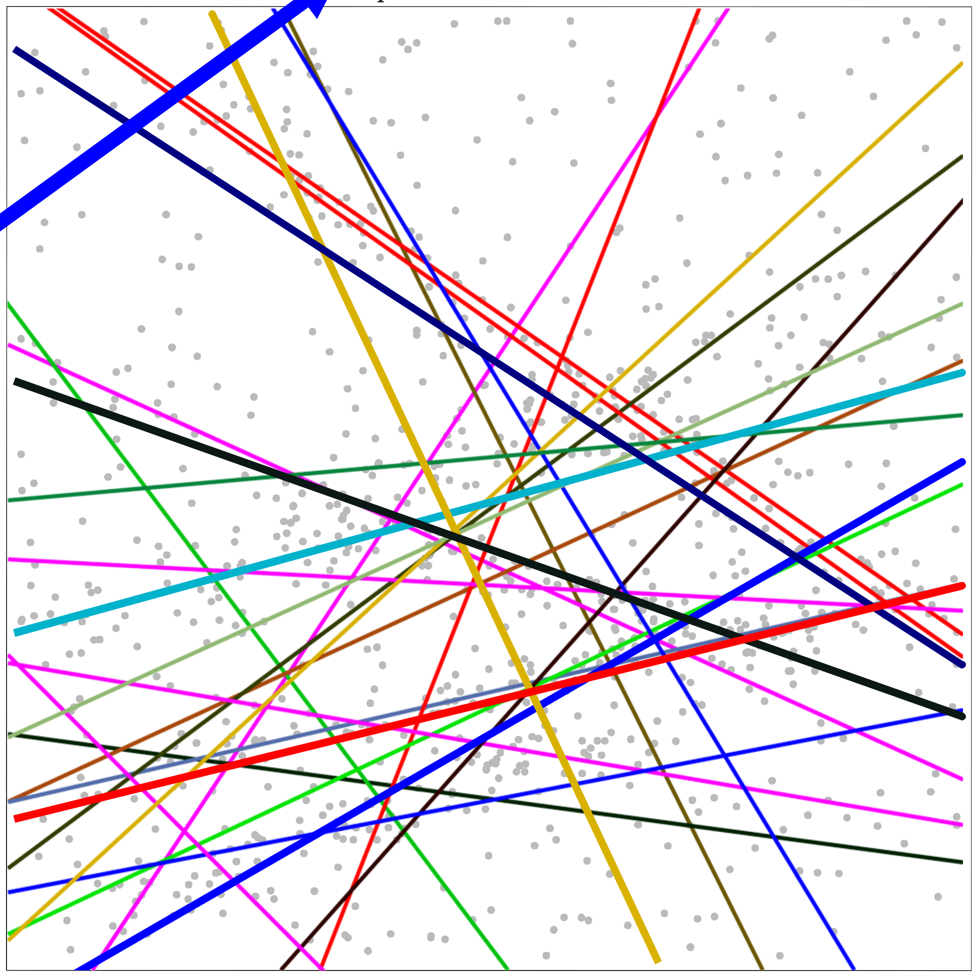
$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

1. Initialization of $\mathbf{\Lambda}$:
randomly sample
K lines from points
(some very large K)

2a. Assign each point to model in $\mathbf{\Lambda}$ with lowest || ||

2b. Re-estimate lines parameters minimizing errors || || among inliers
(e.g. least squares)

addresses the first term of the objective assuming the number of used models is fixed:
$|\Lambda| = const$



data points + randomly sampled lines

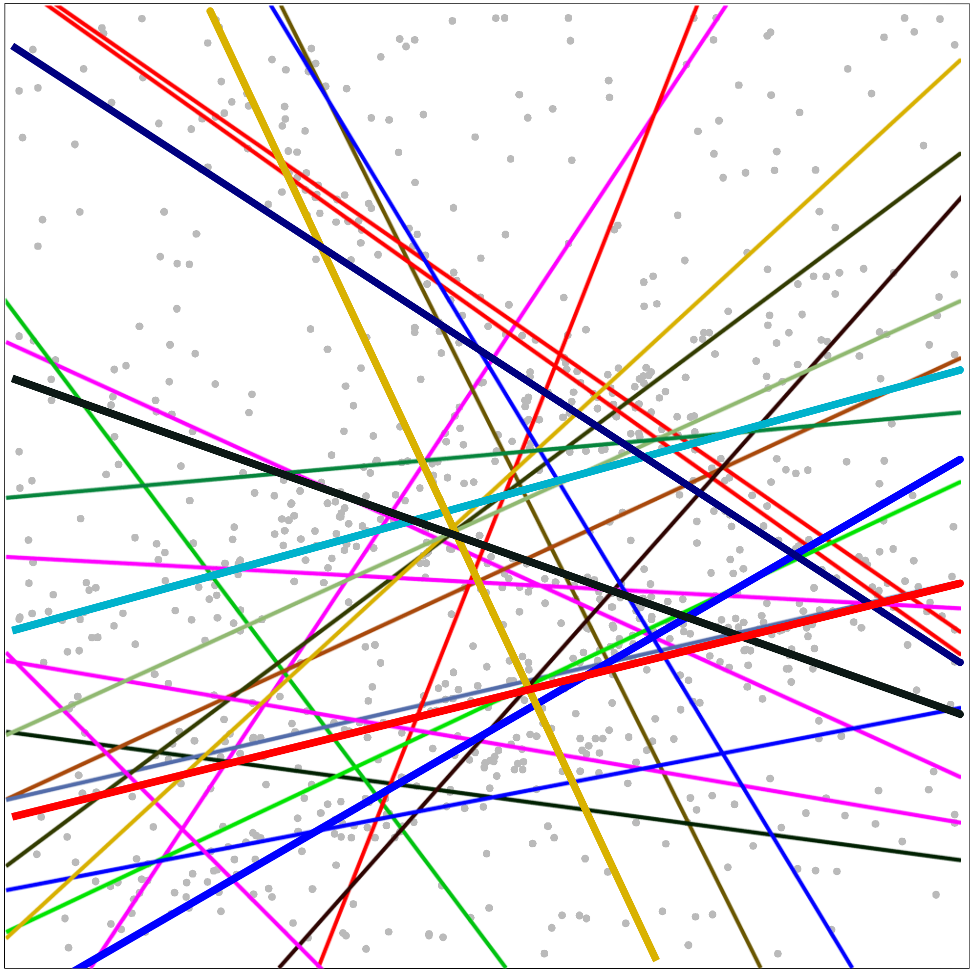
As a function of a set of used models Λ,
E **relates to the objective for *Uncapacitated Facility Location* (UFL) problem**

Optimization problem

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

1. Initialization of Λ:
randomly sample
K lines from points
(some very large K)

2a. Assign each point to model in **Λ** with lowest || ||
2b. Re-estimate lines parameters minimizing errors || || among inliers
(e.g. least squares)



data points + randomly sampled lines

As a function of a set of used models Λ,
E **relates to the objective for** ***Uncapacitated Facility Location*** **(UFL) problem**

Optimization problem

$$E(\mathbf{L},\Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

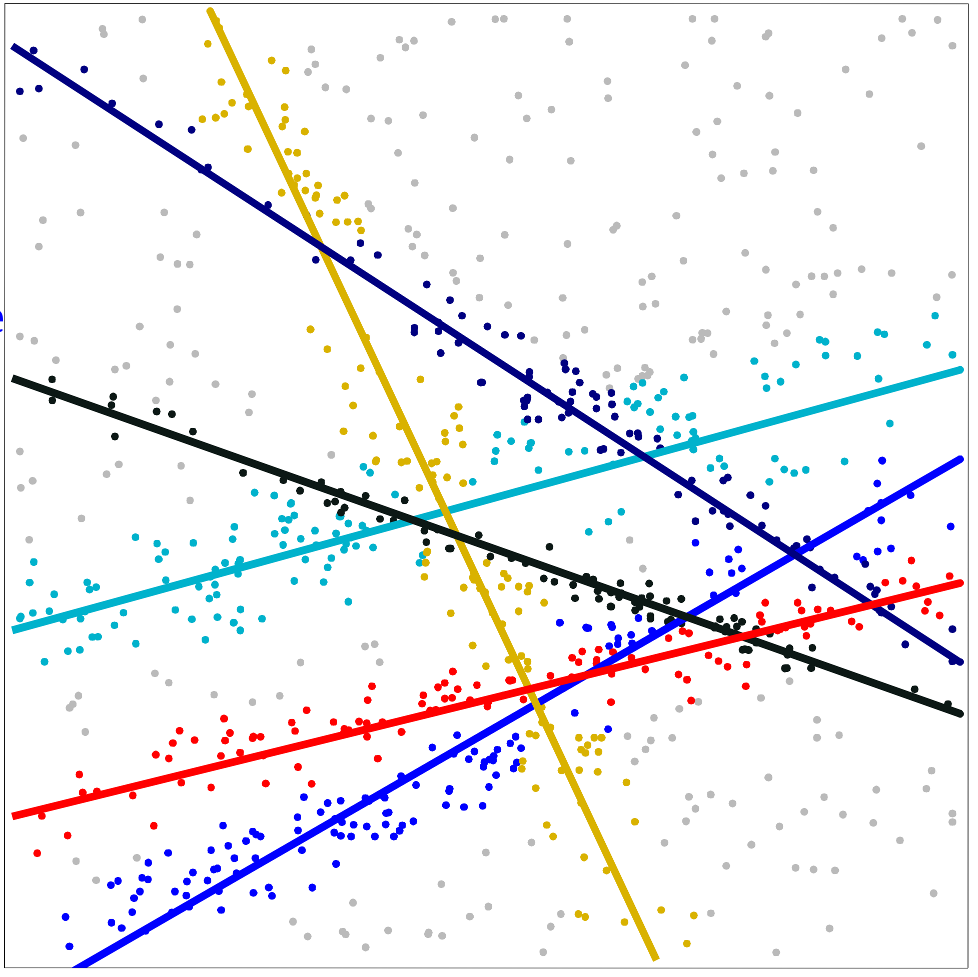
1. Initialization of Λ:
randomly sample
K lines from points
(some very large K)

2. Assign points to the closest model, **re-estimate** line parameters using inliers (e.g. least squares)

3. (UFL heuristic)
clean Λ by removing lines L that are not worth keeping:

$$\sum_{p:L_p=L} \min_{l\in\Lambda\setminus L} ||p-l|| < \sum_{p:L_p=L} ||p-L|| + \gamma$$

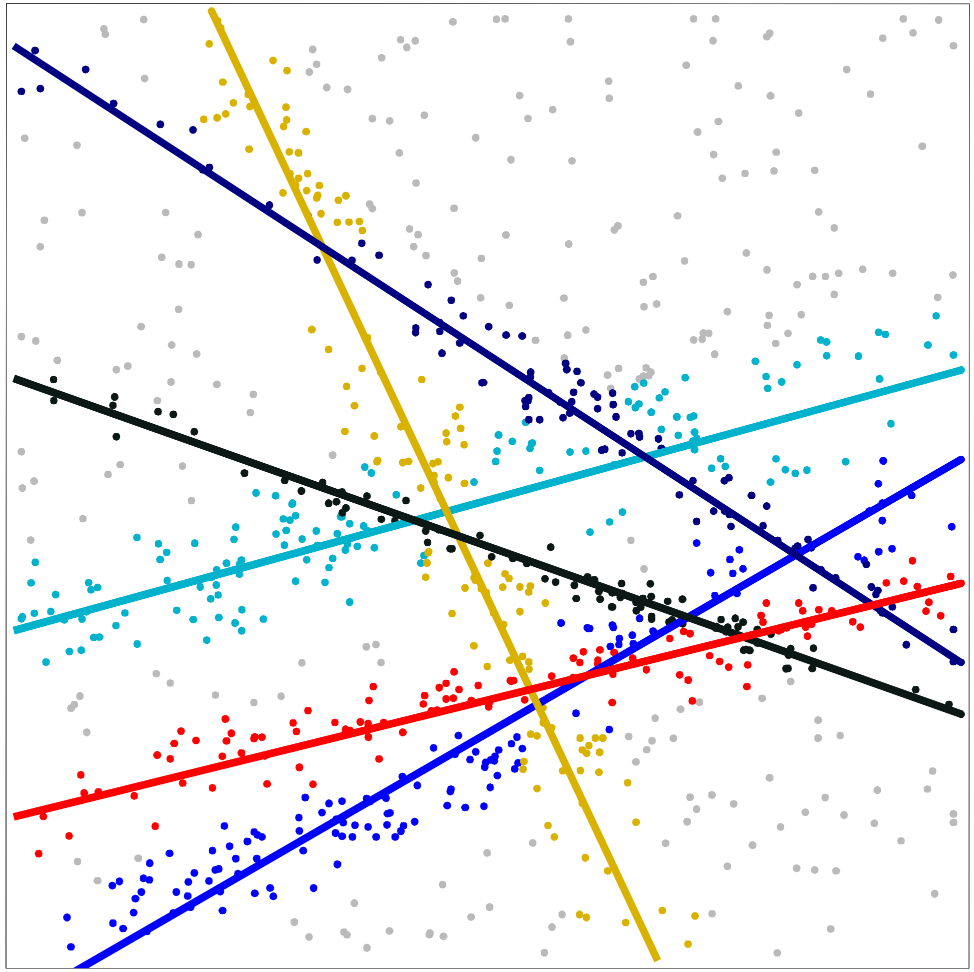
4. Iterate steps 2-3 until convergence

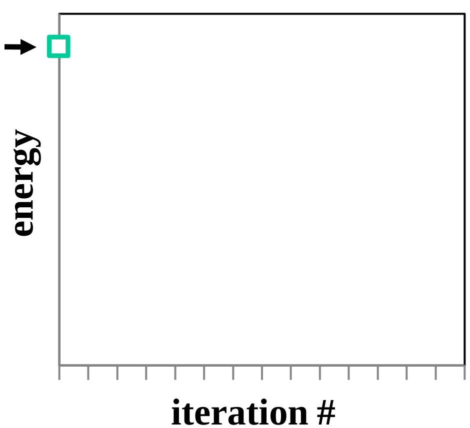


data points & assigned models (lines or ∅)

Optimization problem (algorithm illustration)

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

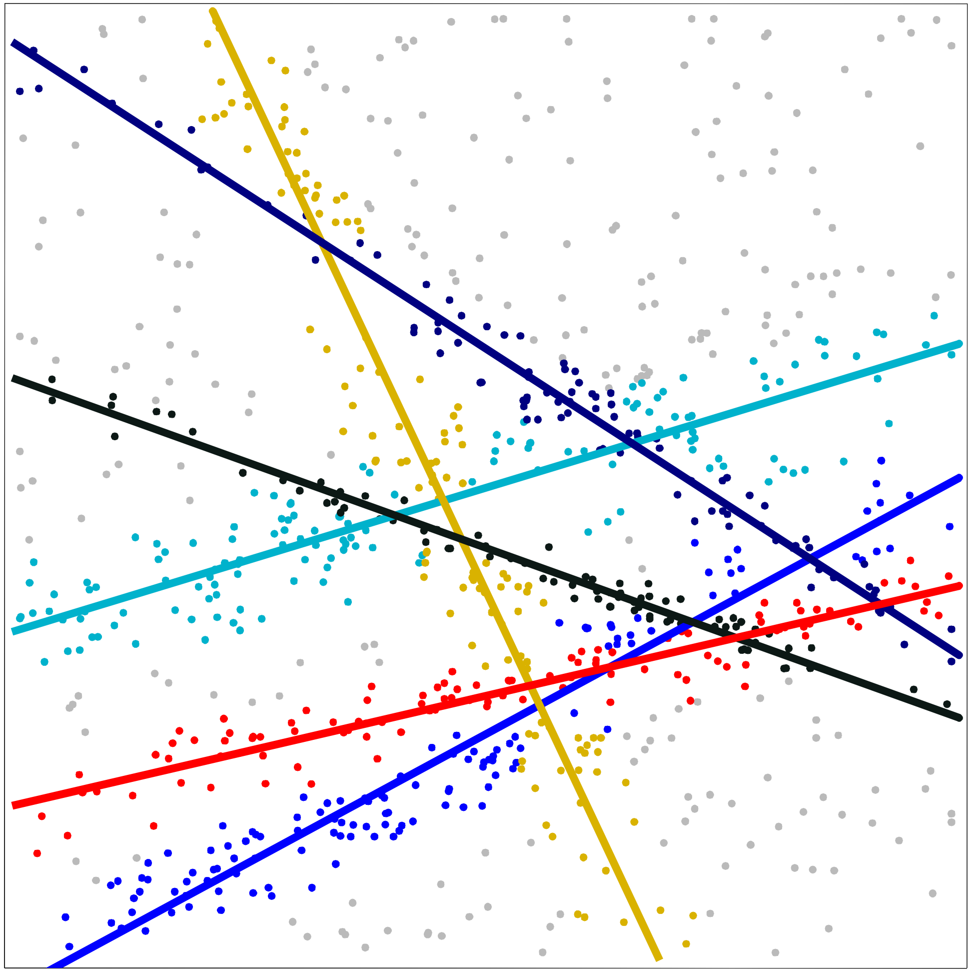




Beginning of iteration 2: points assigned to the closest model that survived iteration 1

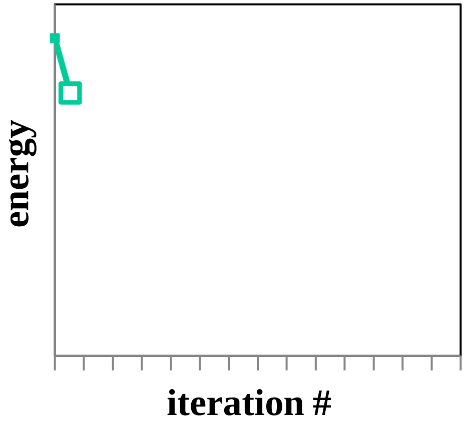
Optimization problem (algorithm illustration)

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$



Re-estimating lines in Λ from their inliers

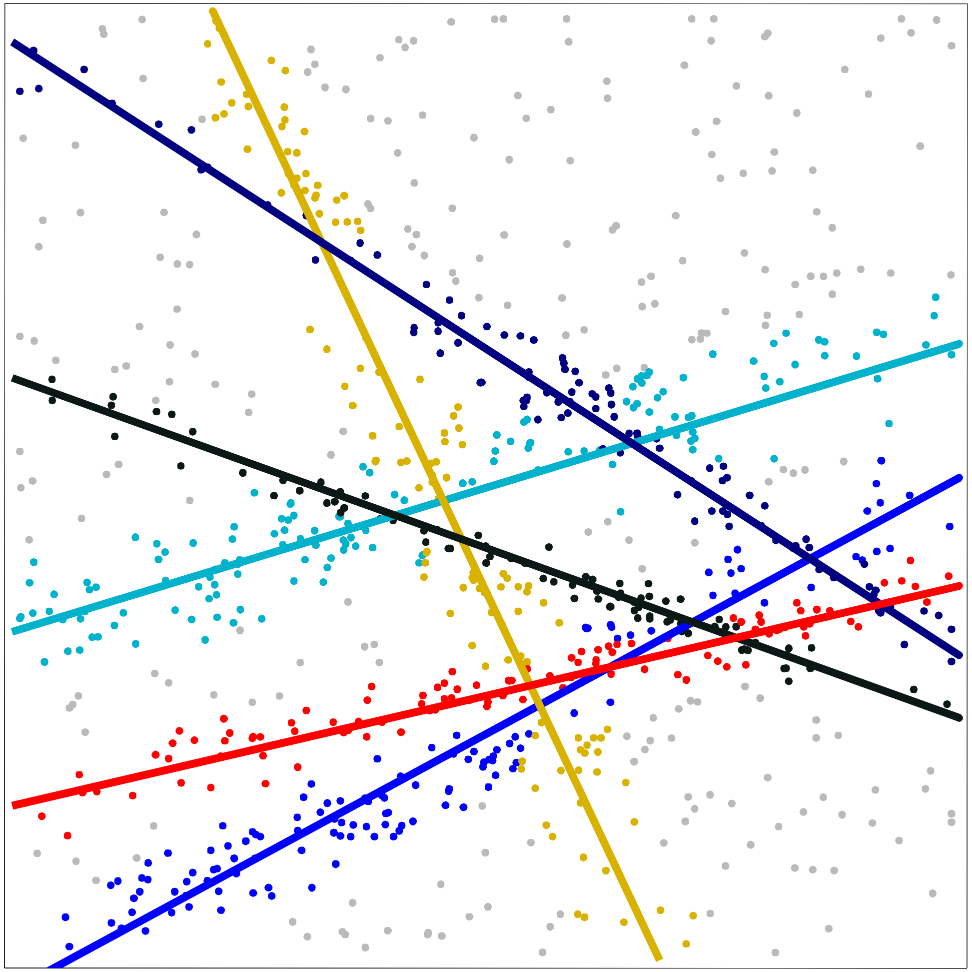
Remove suboptimal lines, if any (UFL heuristic)



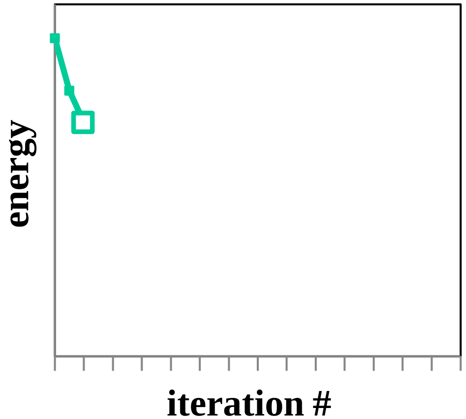
iteration 2: re-estimate model parameters

Optimization problem (algorithm illustration)

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$



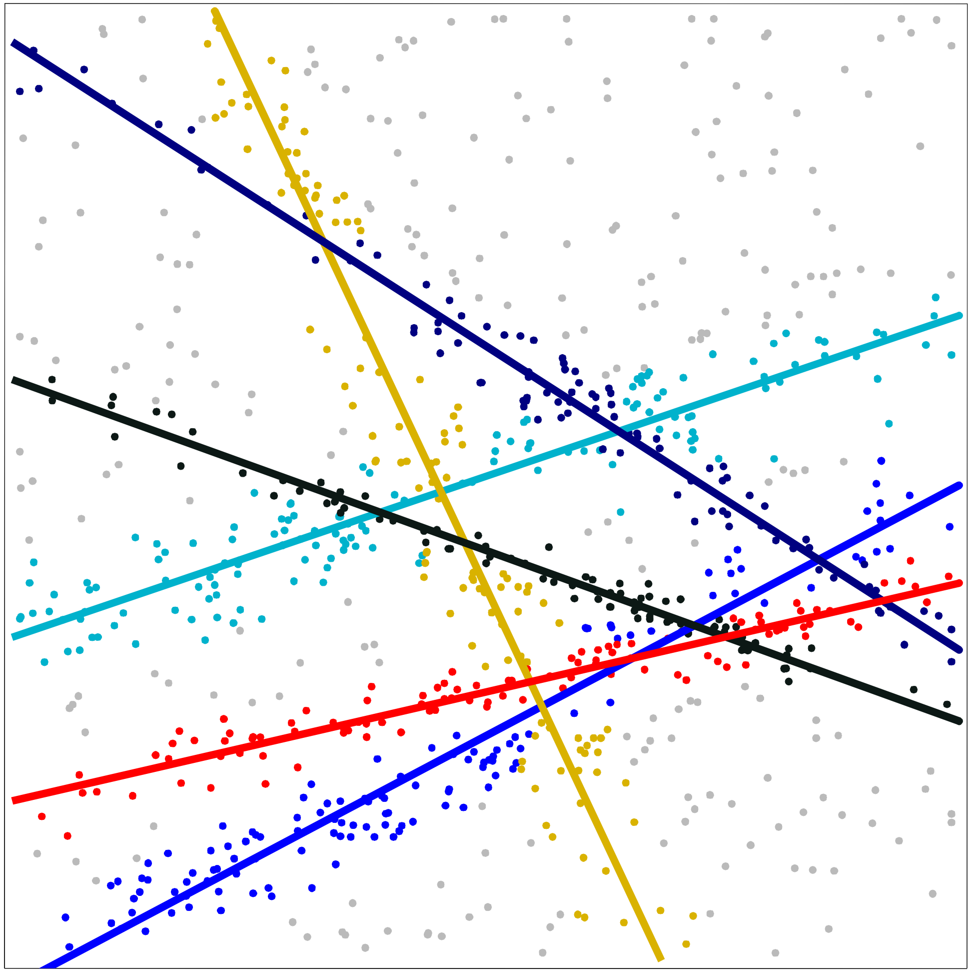
Re-assign points to closest lines



iteration 3: optimize points labeling *L*

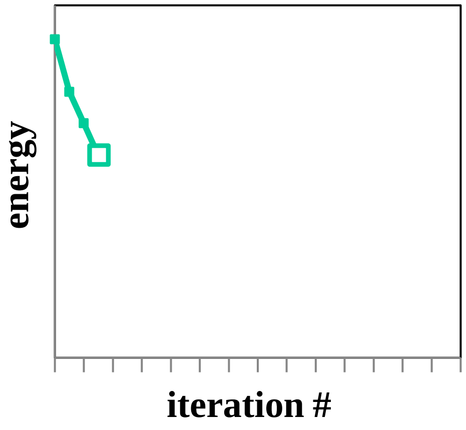
Optimization problem (algorithm illustration)

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$



Re-estimating lines in from their inliers

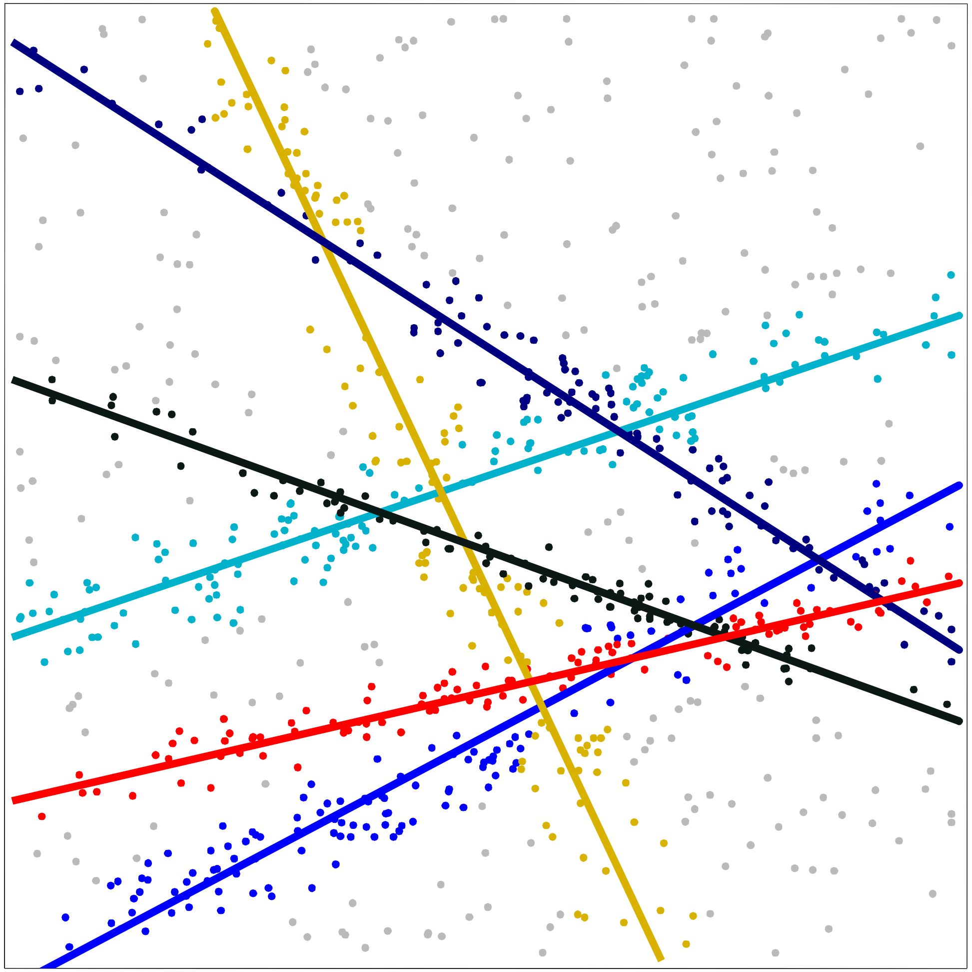
Remove suboptimal lines, if any (UFL heuristic)



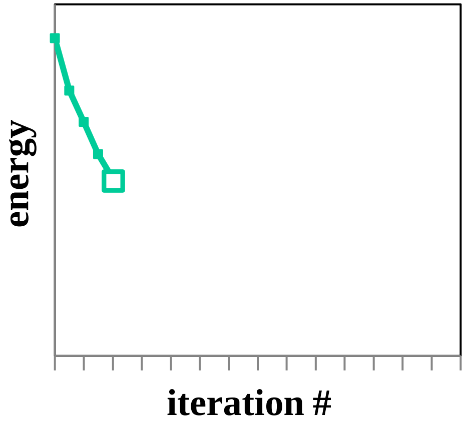
iteration 3: re-estimate model parameters

Optimization problem (algorithm illustration)

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$



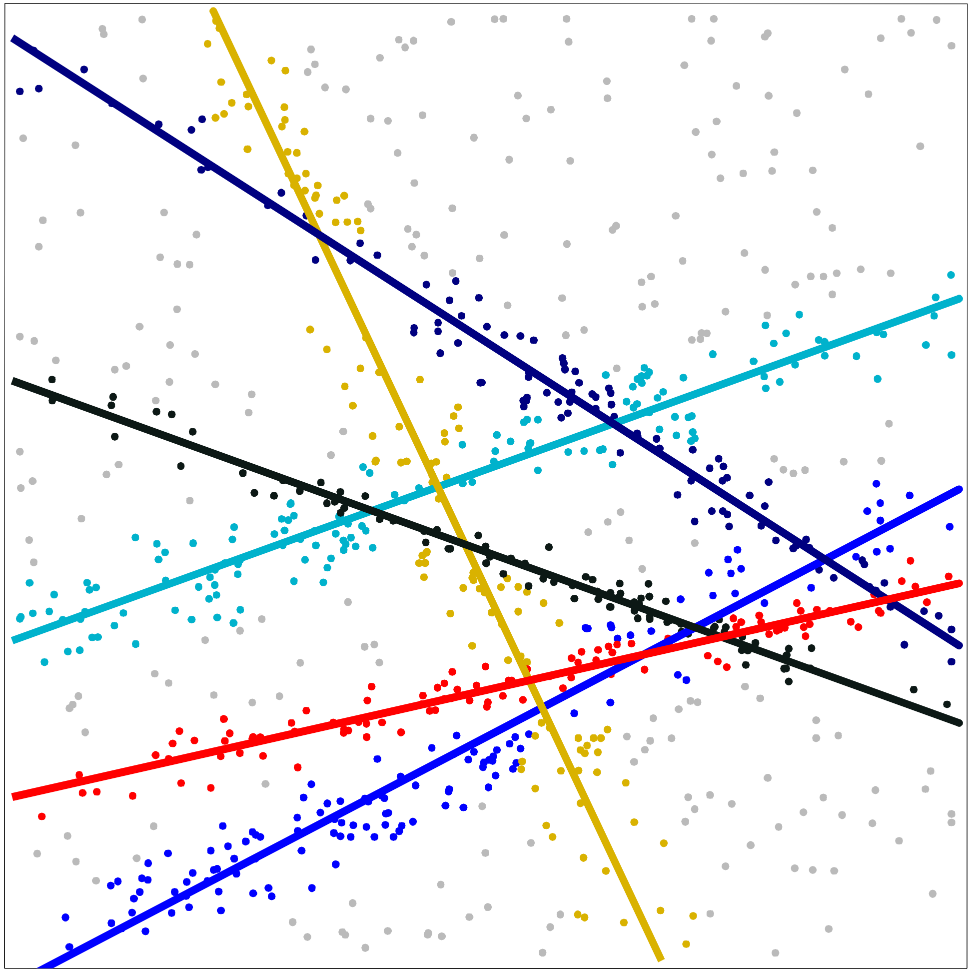
Re-assign points to closest lines

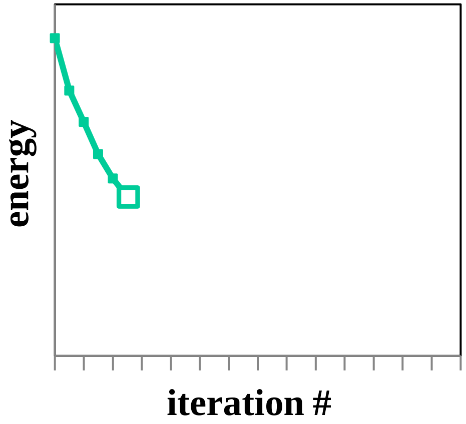


iteration 4: optimize points labeling *L*

Optimization problem (algorithm illustration)

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$



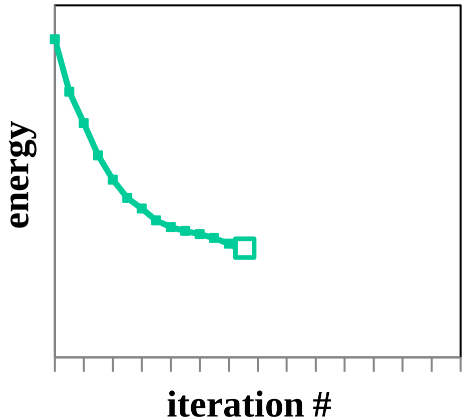


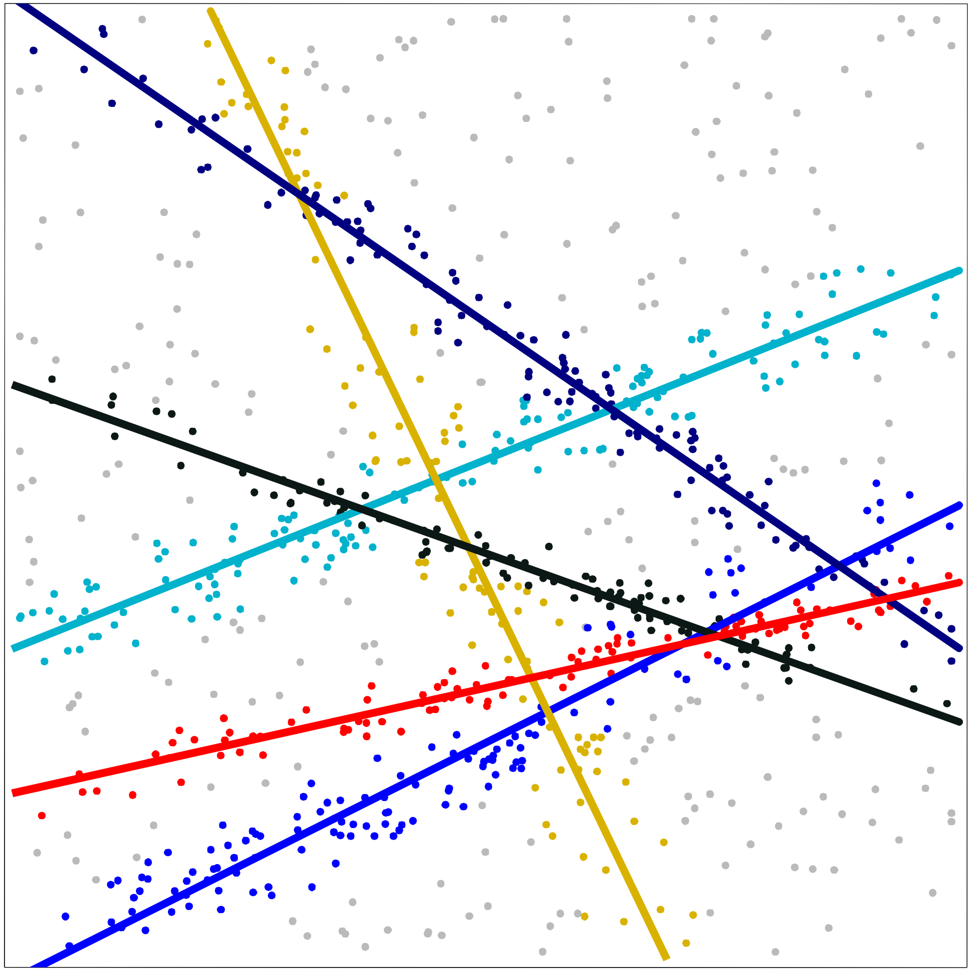
Re-estimating lines in from their inliers

iteration 4: re-estimate model parameters

Optimization problem (algorithm illustration)

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$

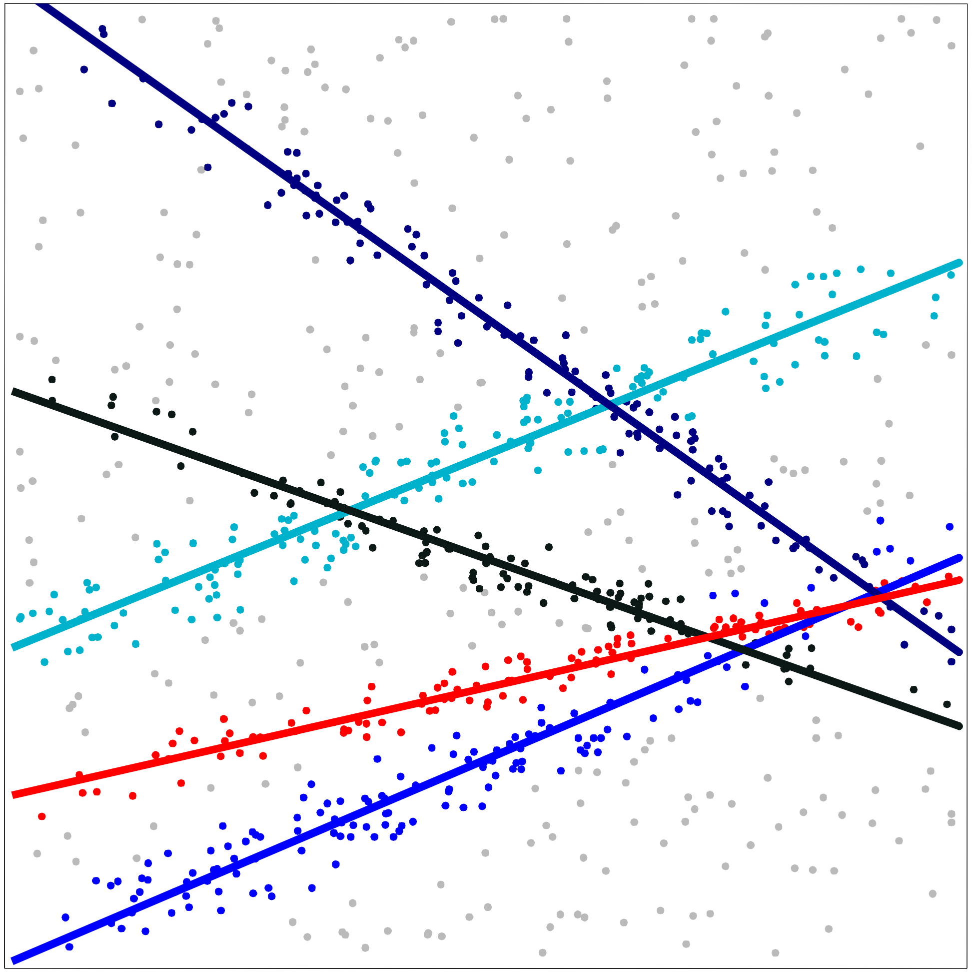




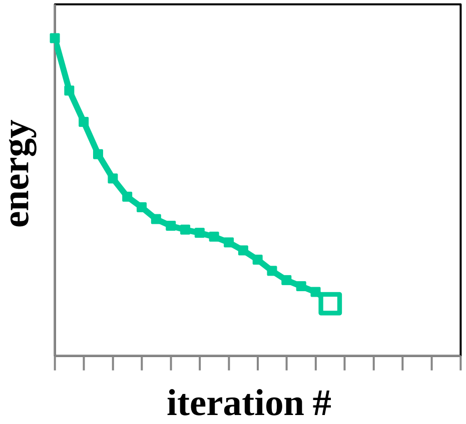
iteration 7...

Optimization problem (algorithm illustration)

$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$



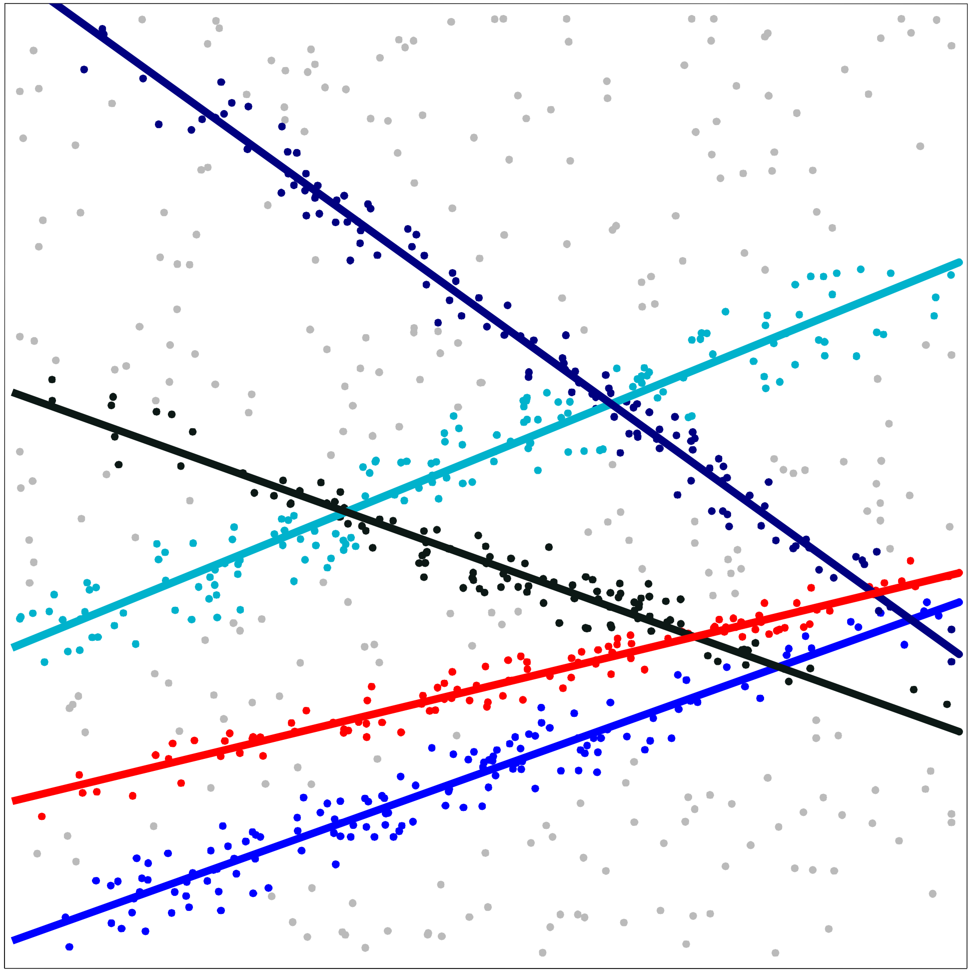
NOTE: yellow line became unnecessary

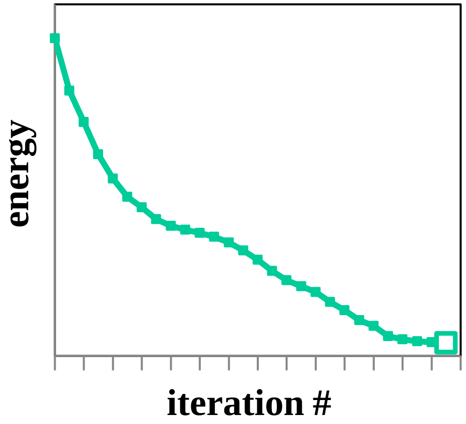


iteration 10...

Optimization problem (algorithm illustration)

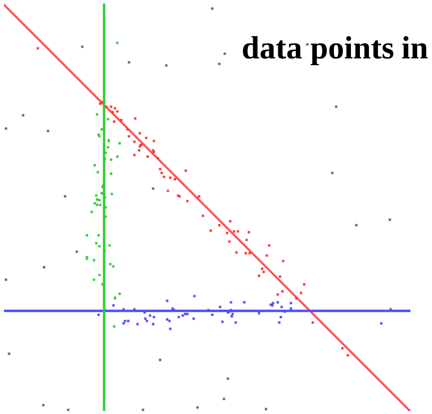
$$E(\mathbf{L}, \Lambda) = \sum_p ||p - L_p|| + \gamma \cdot |\Lambda|$$





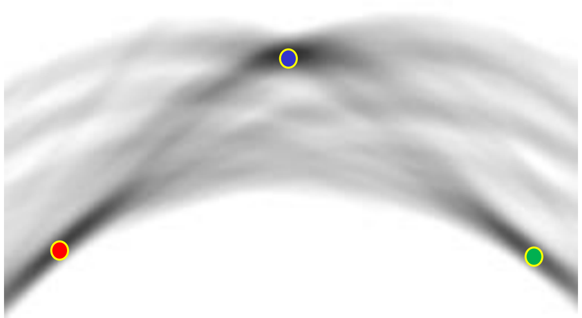
iteration 15... converged.

What's going on inside the space of labels (lines/models)



space of all lines

Hough space

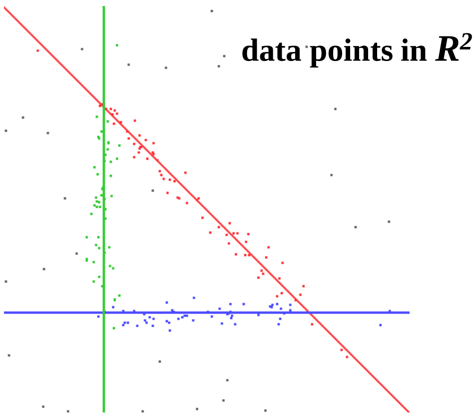


Hough Transform

[Szeliski Sec.4.3.2]

- **also R^2, each point here defines line parameters (a,b)**
- **darkness indicates the number of inliers for a line**

What's going on inside the space of labels (lines/models)

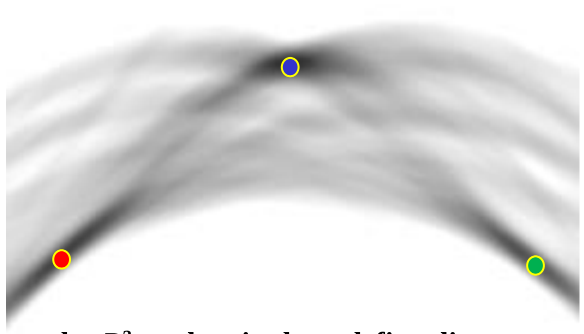


Hough Transform

[Szeliski Sec.4.3.2]

space of all lines

Hough space



- also R^2, each point here defines line parameters (a,b)

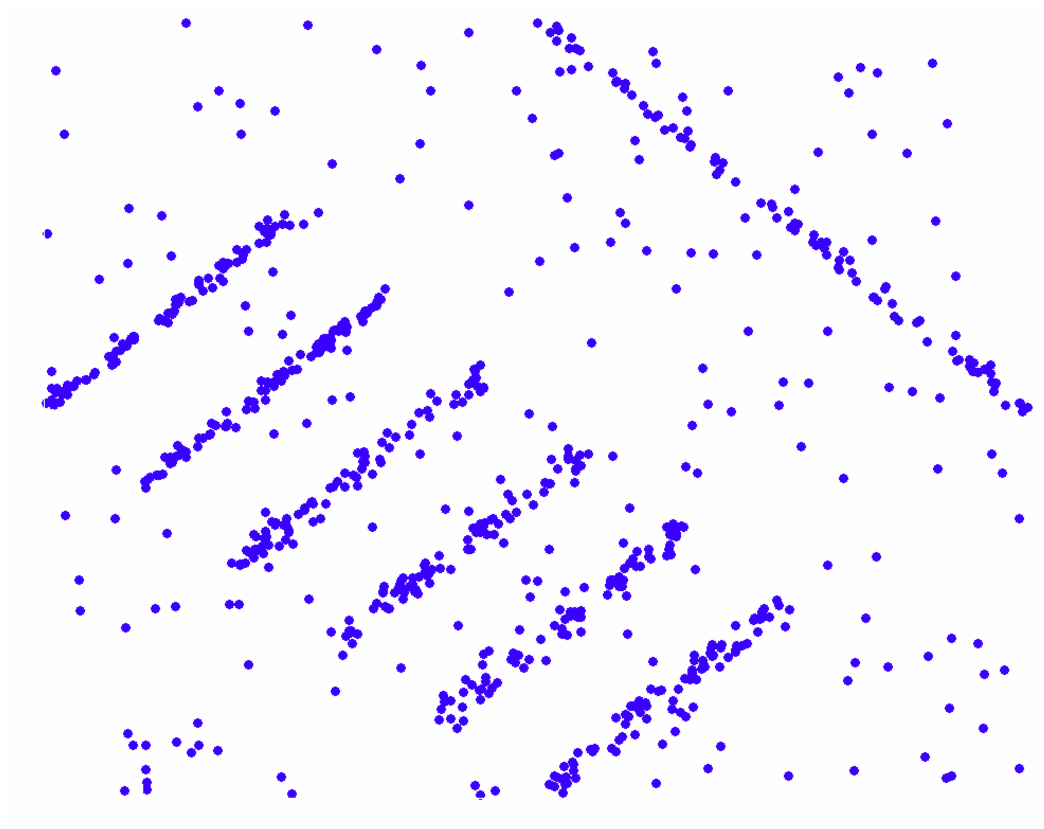
- darkness indicates the number of inliers for a line

Q: why not just look for *modes* in the Hough space?

(strong local maxima)

Note: RANSAC searches the maxima by exploring a (large) sample of lines randomly sampled from the "line density" in the Hough space.

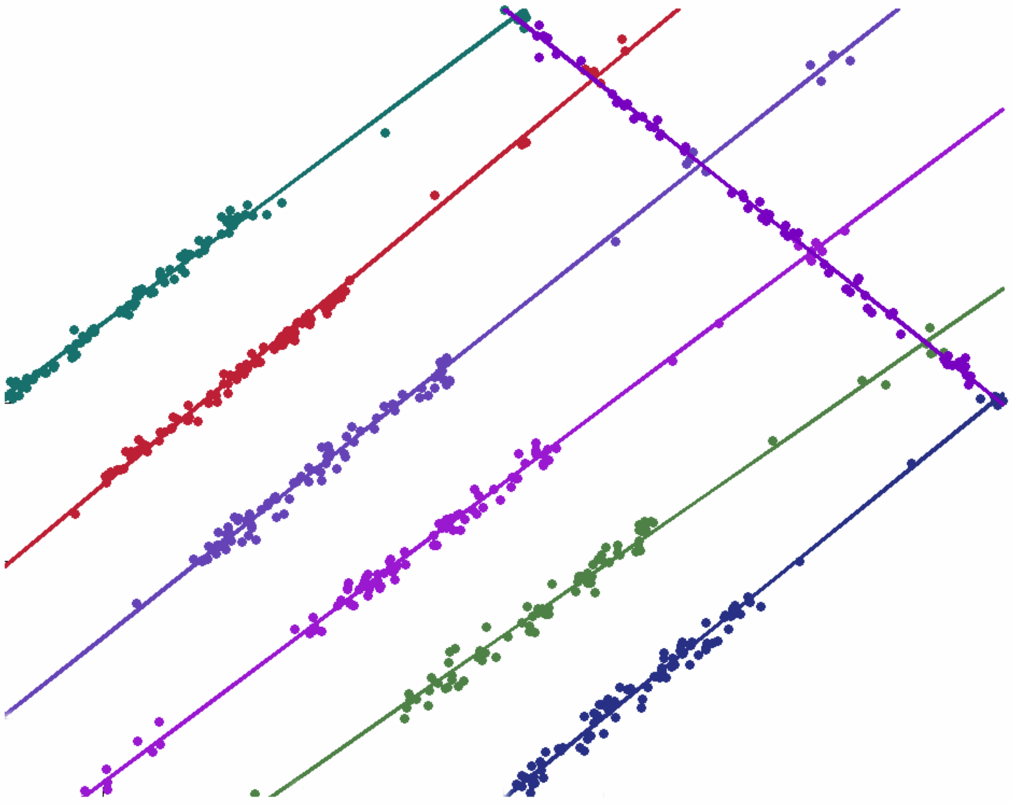
Comparison



Low noise

original data points

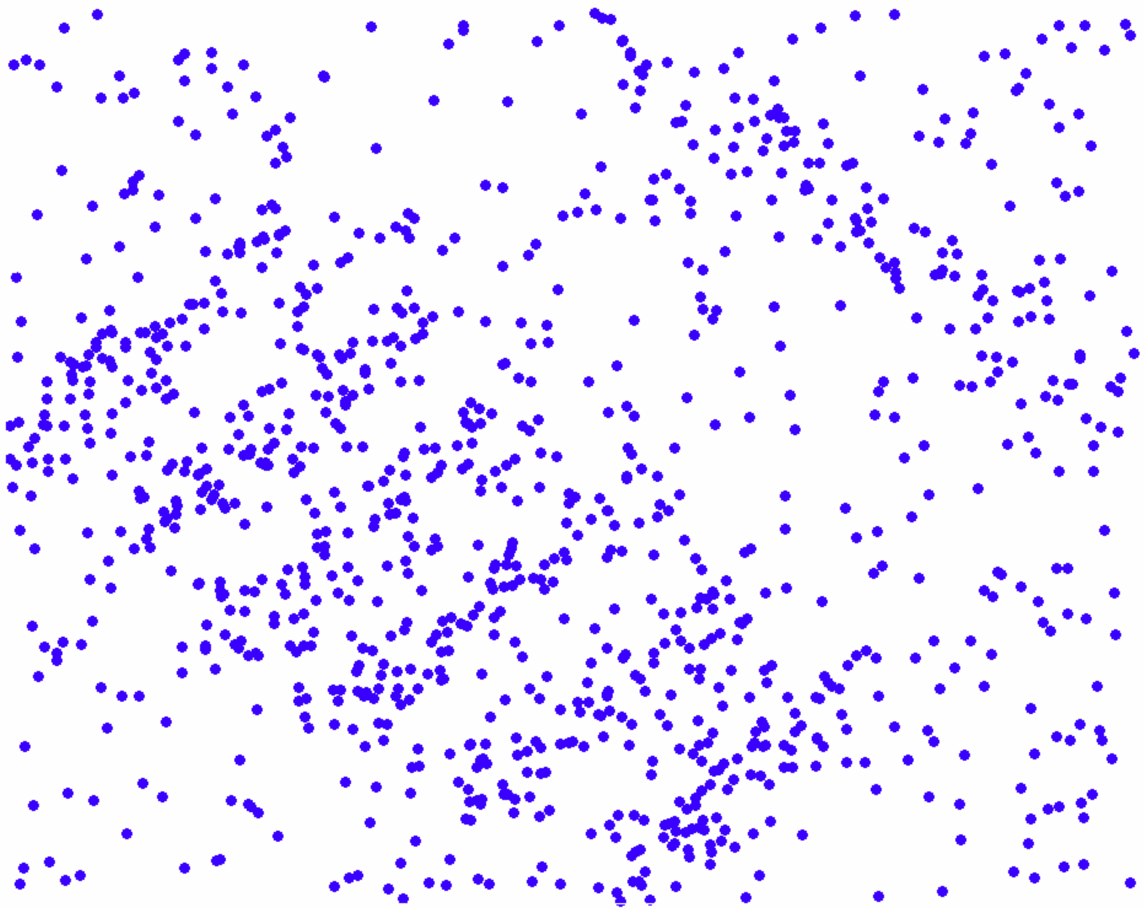
Comparison



Low noise

sequential RANSAC, modes in Hough space, UFL approach

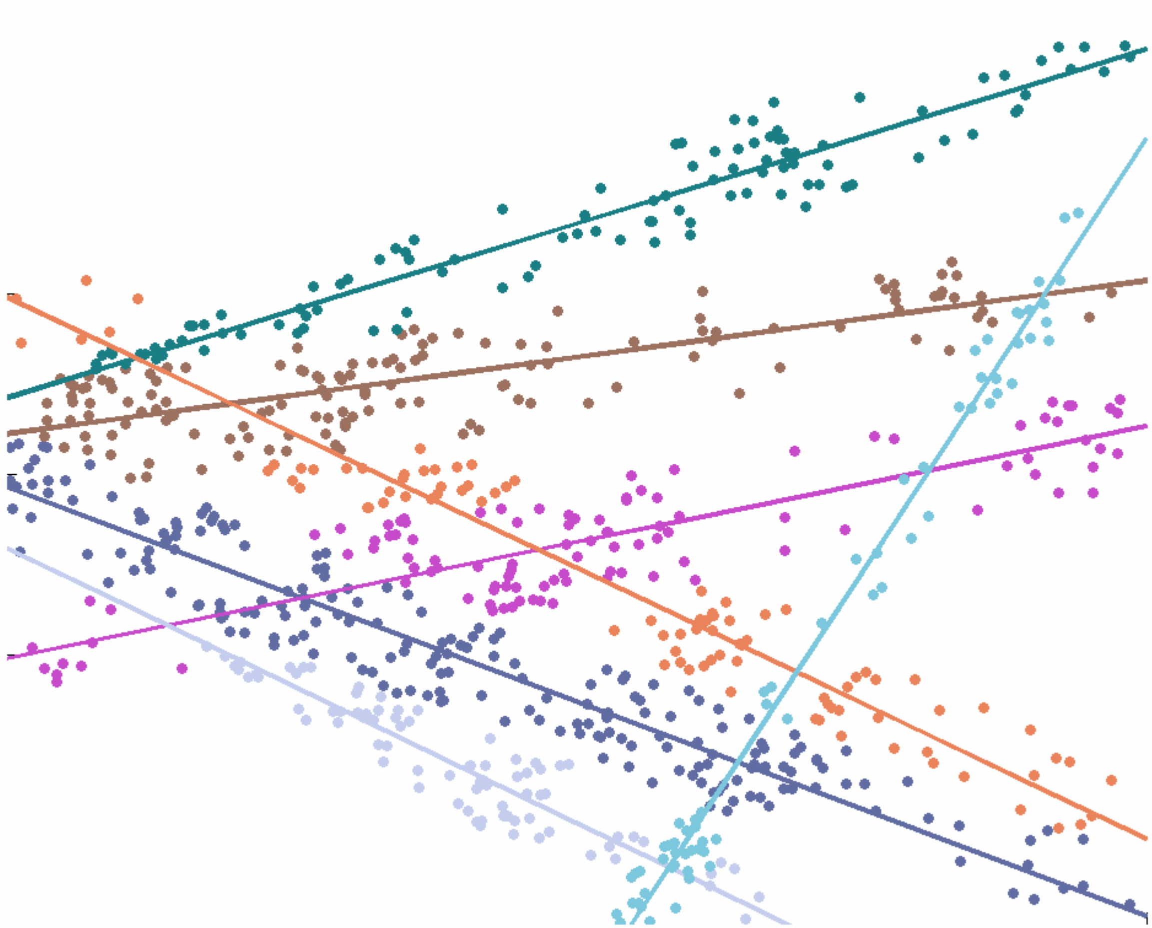
Comparison



High noise

original data points

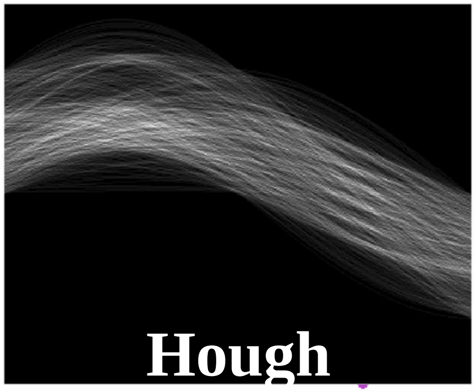
Comparison

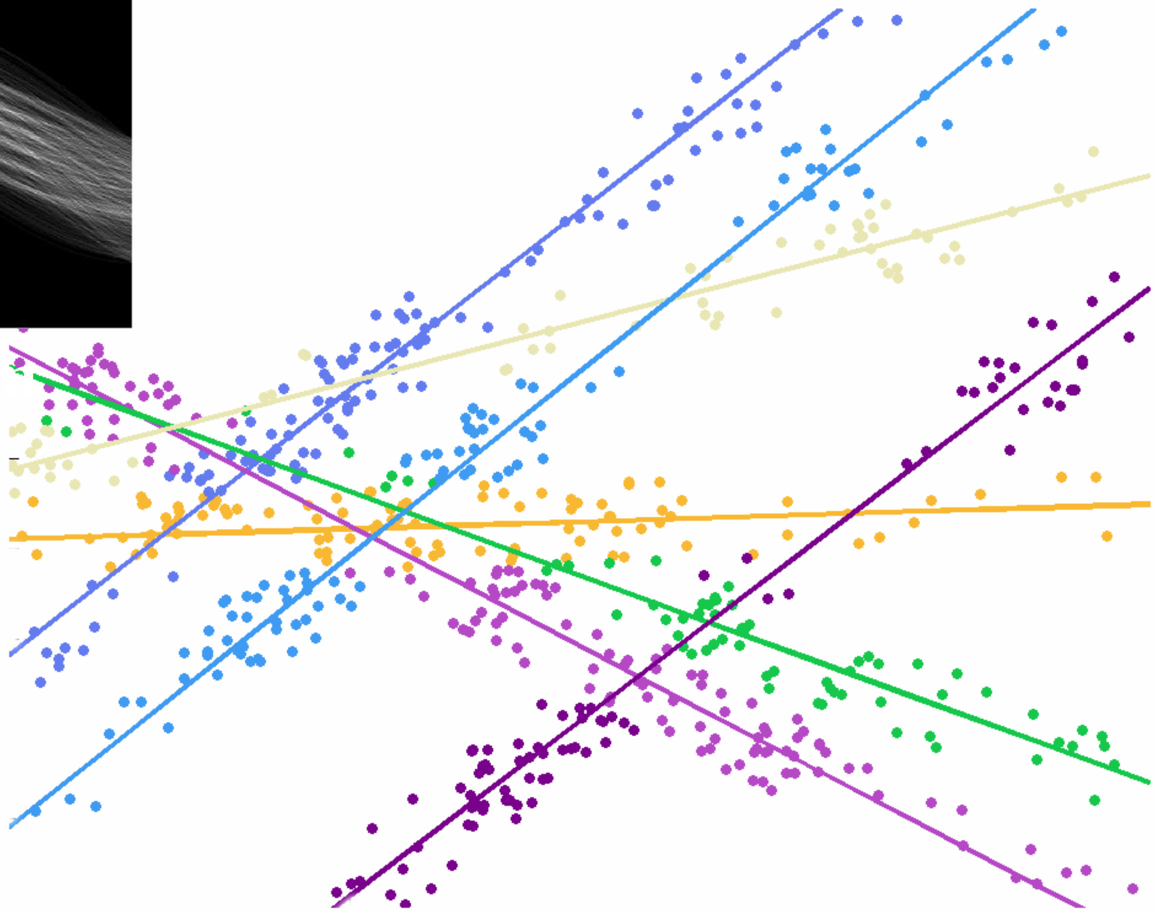


High noise

sequential RANSAC

Comparison

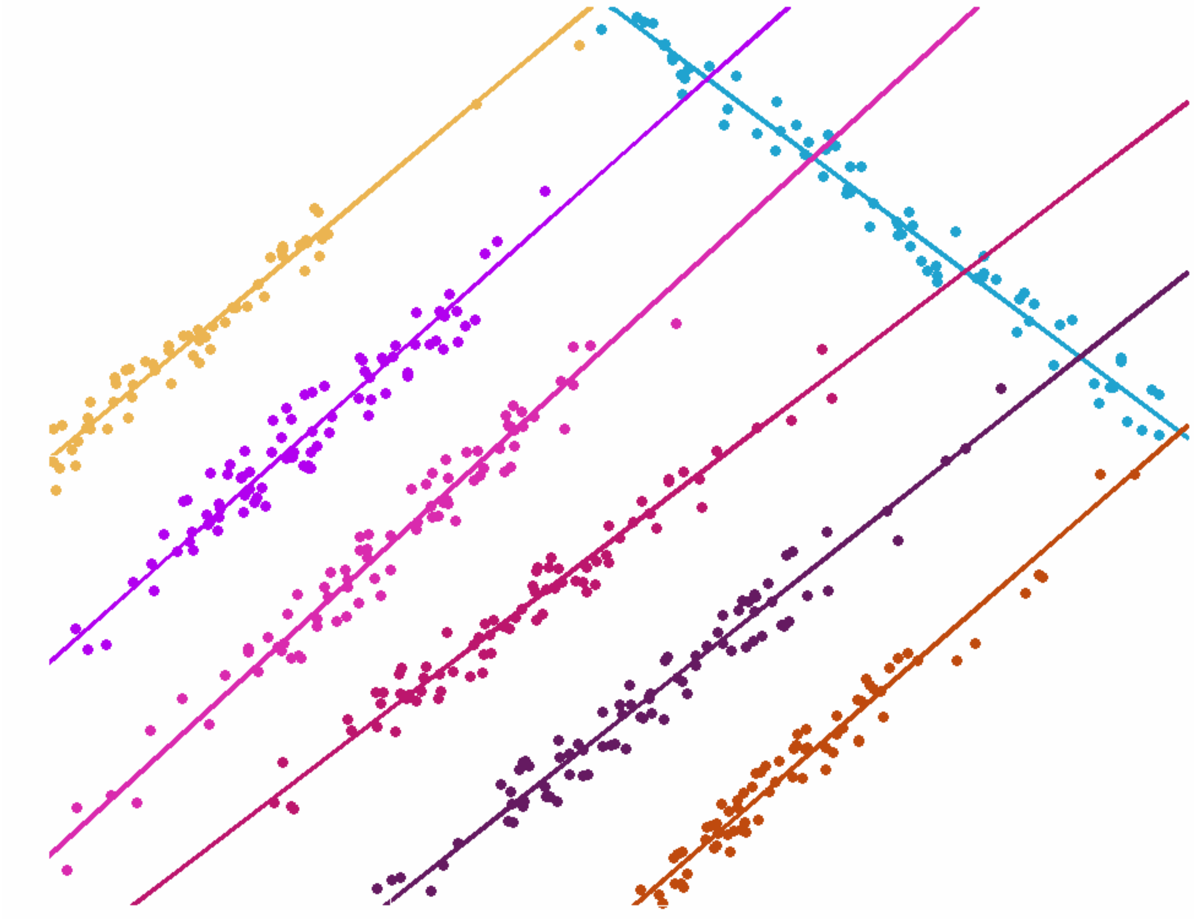




High noise

Finding modes in Hough-space

Comparison

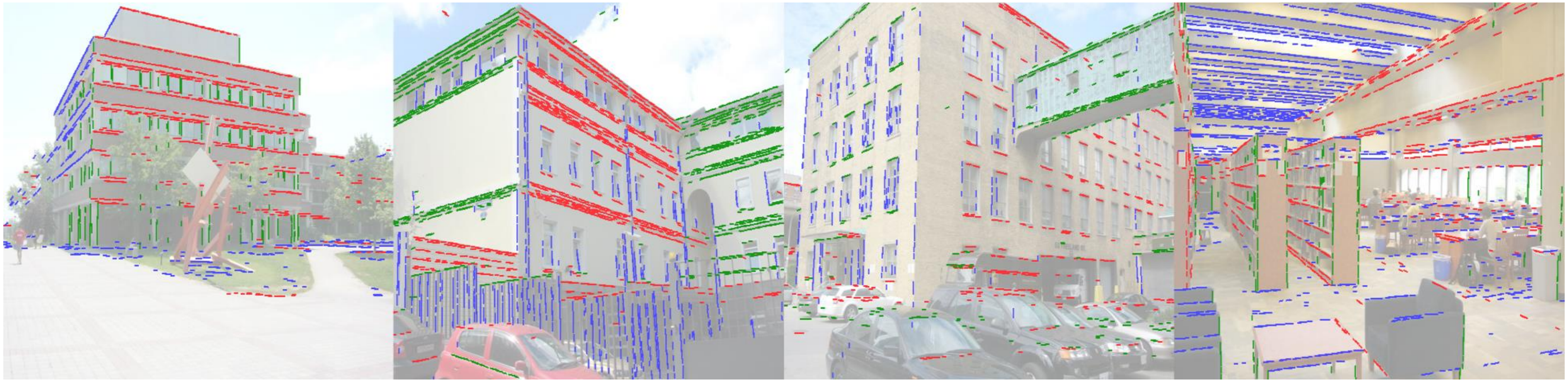


High noise

UFL-based approach
[Delong et al. IJCV12]

$$E(\mathbf{L}, \Lambda) = \sum_{p} ||p - L_p|| + \gamma \cdot |\Lambda|$$

Line fitting on real image data



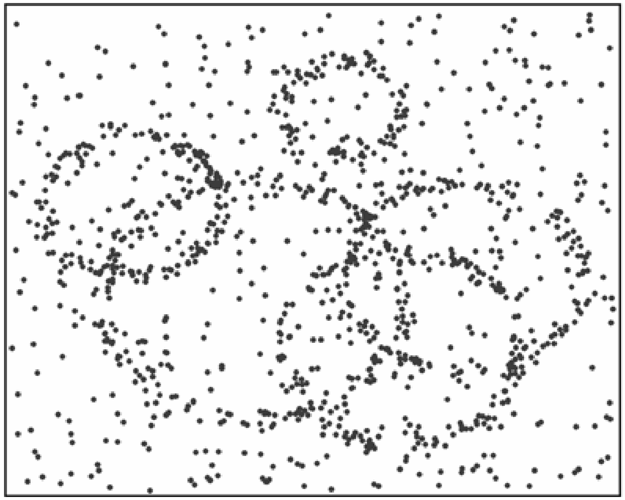
data points are "Canny edges"

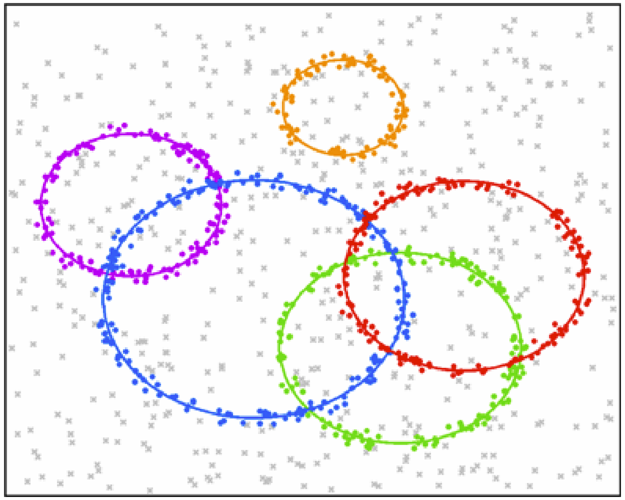
[Delong et al. NIPS 2012]

Note: color indicates clusters of lines with common *vanishing point*

Fitting other geometric models

$$\| p-\theta \| \ := \ \left((x_p - c_x)^2 - (y_p - c_y)^2 - r^2\right)^2 \qquad \text{for} \qquad \theta = \{c_c, c_y, r\}$$





Model fitting for arbitrary geometric models $\boldsymbol{\theta}$

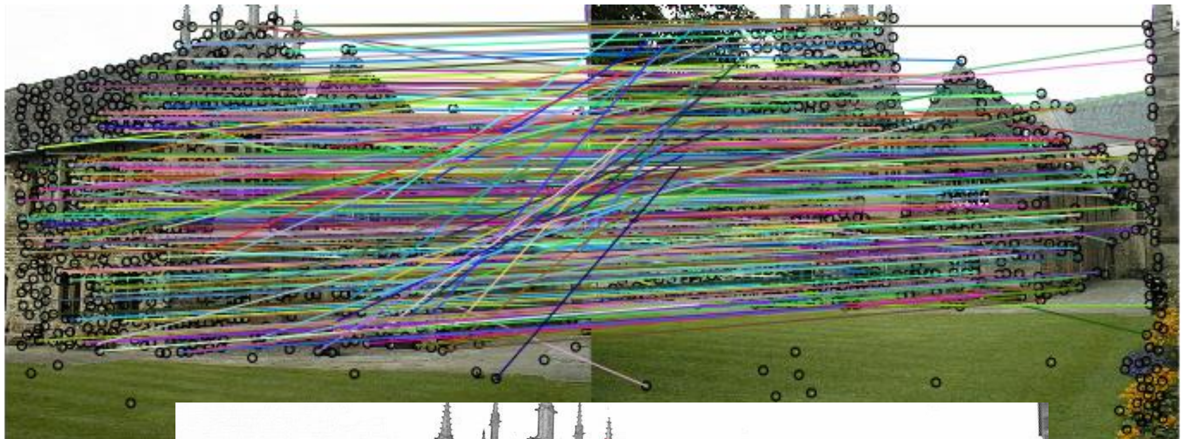
Need: 1) define an error measure w.r.t. model parameters $\| p-\theta \|$

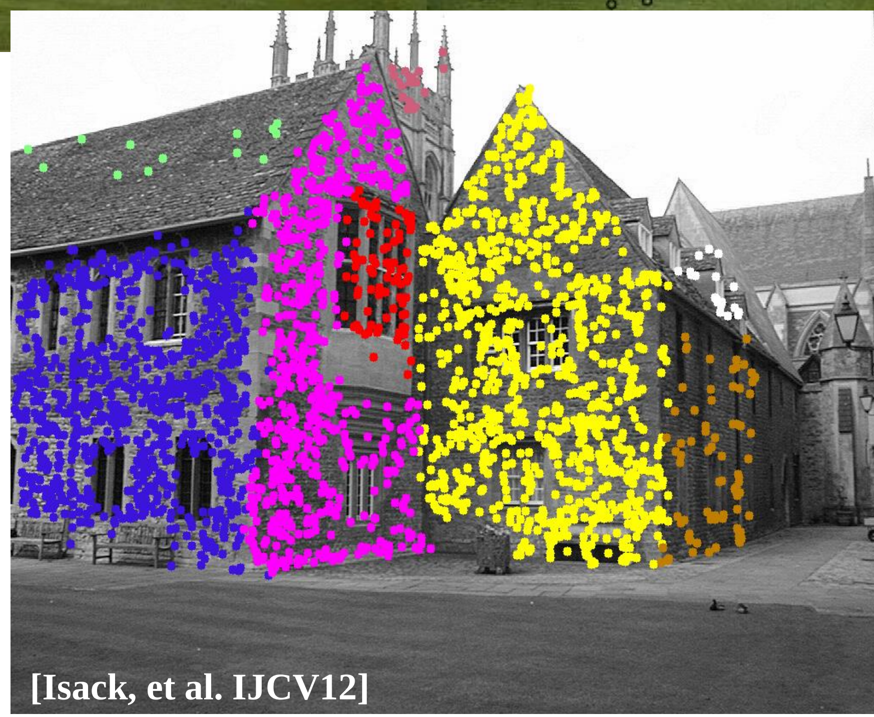
2) efficient method for minimizing the sum of errors among inliers w.r.t. model parameters $\boldsymbol{\theta}$

$$\min_{\theta} \sum_{p \in S_\theta} \| p-\theta \|$$

Fitting multiple homographies (e.g. planes)

matched features (p,p’), as earlier





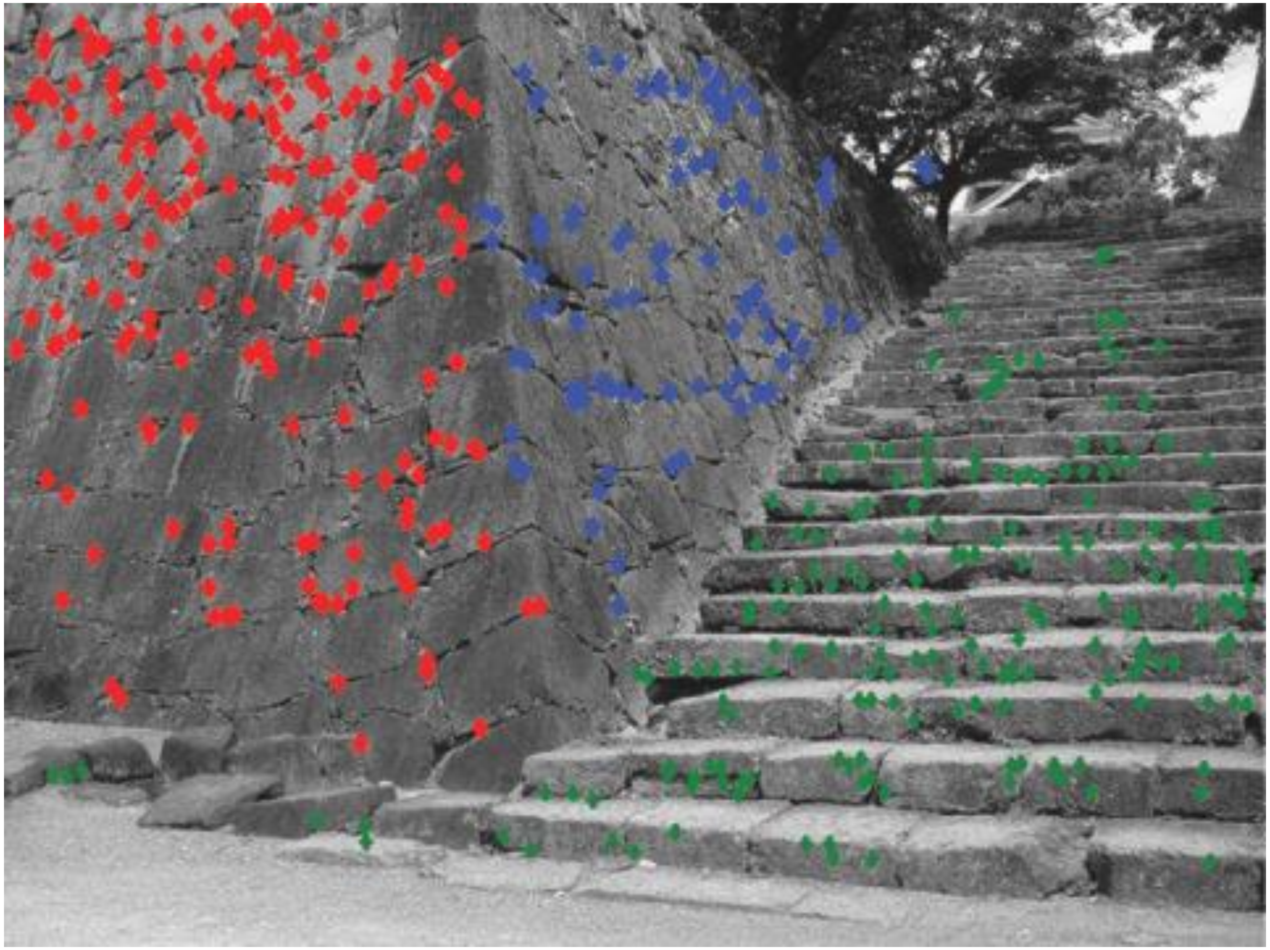
[Isack, et al. IJCV12]

using
symmetric
re-projection errors

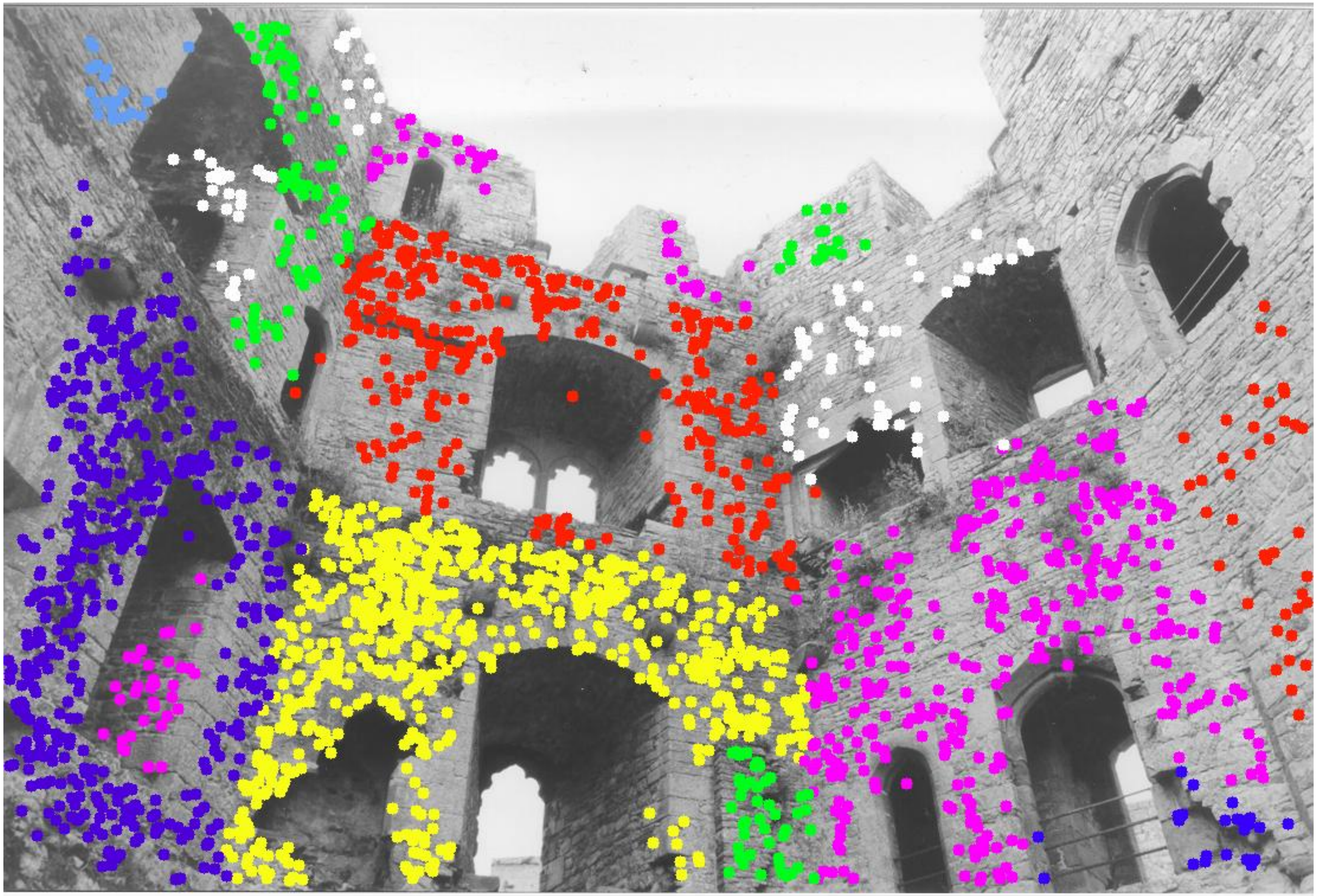
$$\| p' - Hp \| + \| p - H^{-1} p' \|$$

as an error measure between match (*p*, *p’*) and homography *H*

Fitting multiple homographies (e.g. planes)

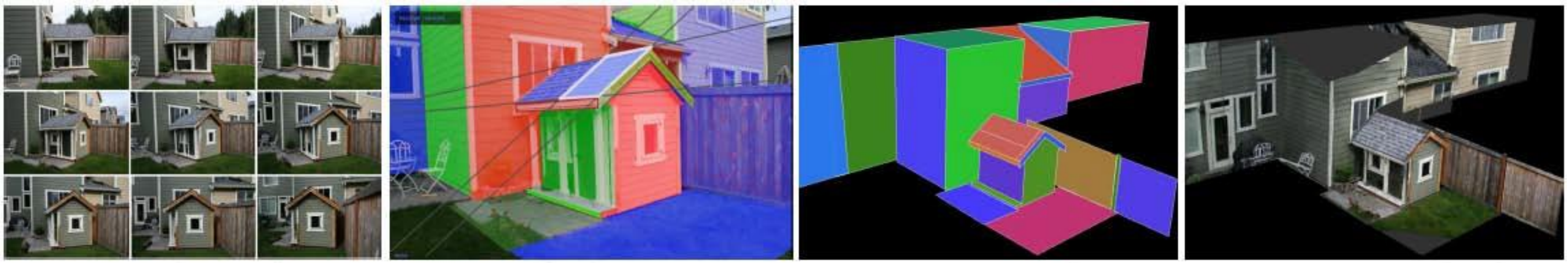


Fitting multiple homographies (e.g. planes)



same scene from a different view point…

Note very small steps between each floor



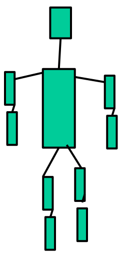
Input Photographs | 2D Sketching Interface | Geometric Model | Texture-mapped model

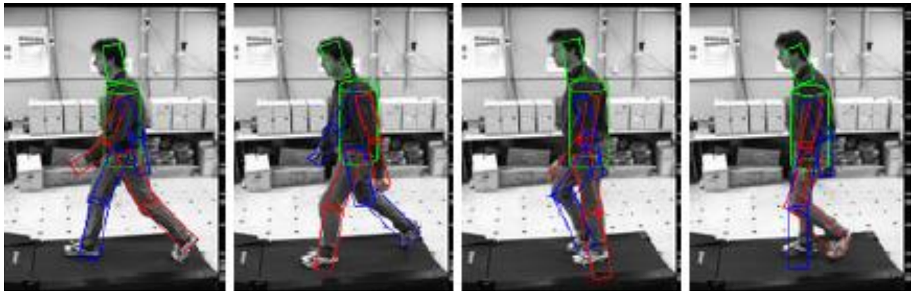
user can help with rough initialization instead of random sampling

[Sinha et al. 2008]

Fitting **dependent** models (parts)

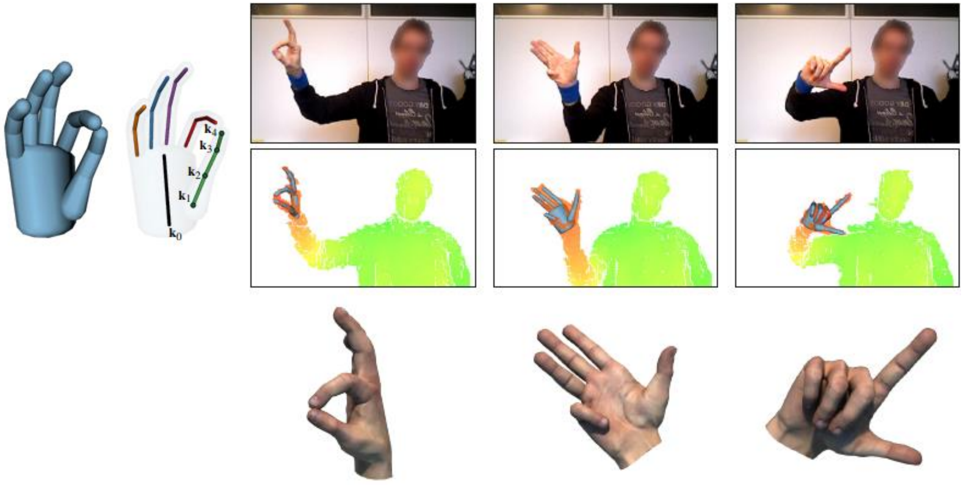
- **Pictorial structures** [Felzenswalb & Huttenlocher, 2005]





- **Articulated model tracking**

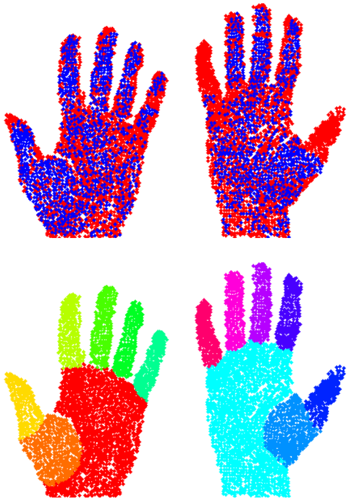
  hand model fitting to 3D point cloud



[Andrea Tagliasacchi et al., 2013]

- **Finding low dim. subspaces**

  e.g. in the space of point tracks



[Carl Olsson et al.,2017]

Fitting **dependent** models (parts)

Can use **physics** to define objectives (losses):

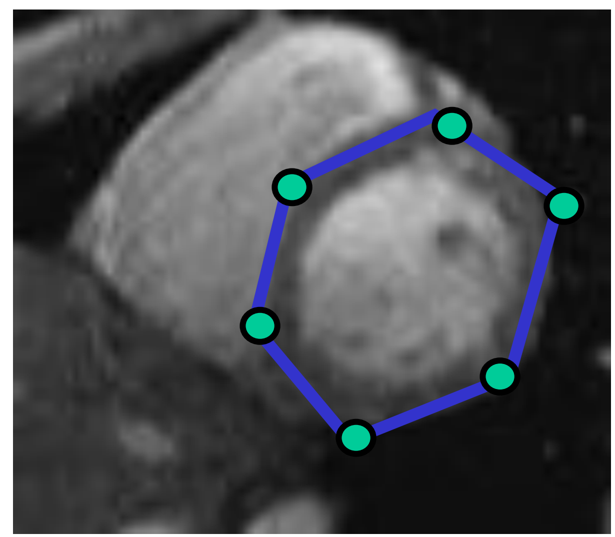
- parts interactions (e.g. spring-like)
- kinematics
- attraction/repulsion to image features

▪ **Articulated model tracking**

hand model fitting to 3D point cloud

[Andrea Tagliasacchi et al., 2013]

▪ **Active contours (snakes)**



[Kass Witkin Terzopoulos, 1988]

Geometric model fitting in vision

MODELS: lines, planes, homographies, affine transformations, projection matrices, fundamental/essential matrices, etc.

next topic

- **single models** (e.g. panorama stitching, camera projection matrix)
- **multiple models** (e.g. multi-plane reconstruction, multiple rigid motion)

FIRST STEP: detect some features (corners, LOGS, etc)
and compute their descriptors (SIFT, MOPS, etc.)

SECOND STEP: match or **track**

THIRD STEP: fit models
(minimization or errors/losses)