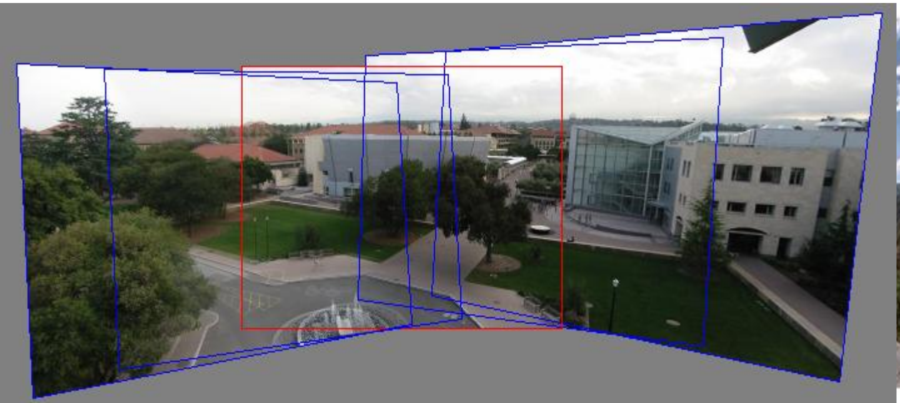
计算机视觉

Computer Vision

Lecture 7: Feature Matching and Model Fitting

张 超

信息科学技术学院 智能科学系

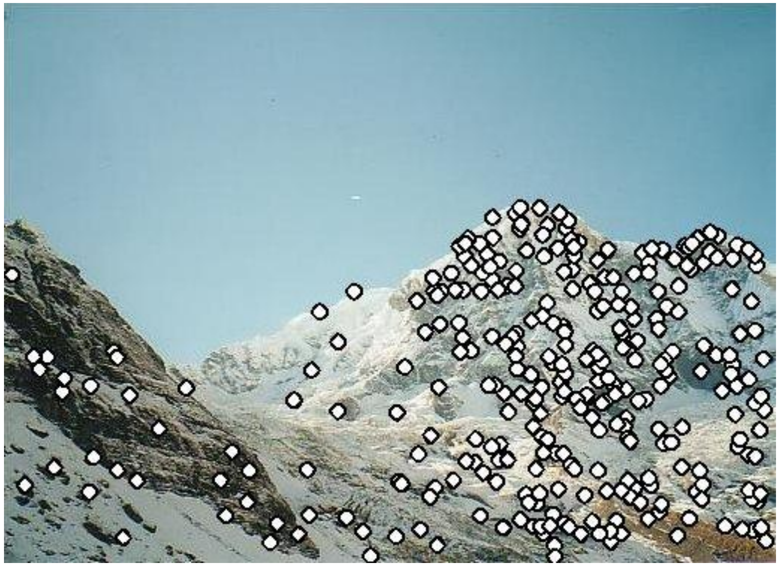


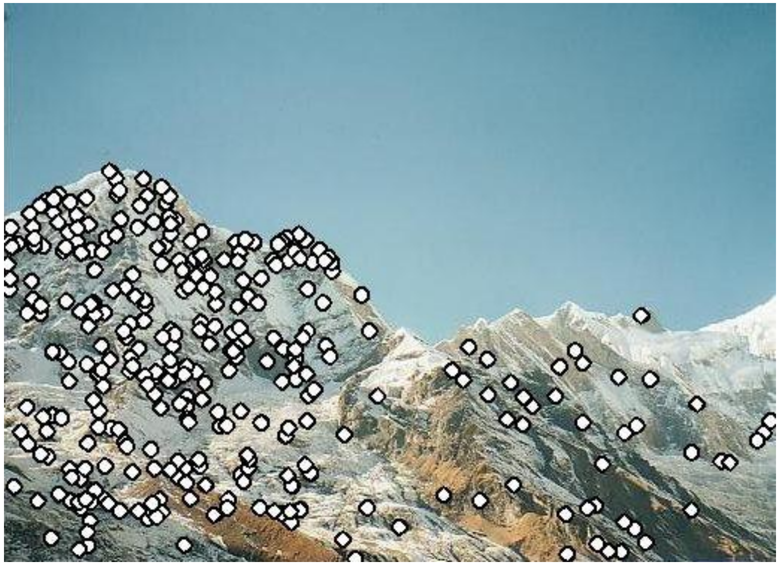
How do we build panorama?

- **We need to match (align) images**

Matching with Features

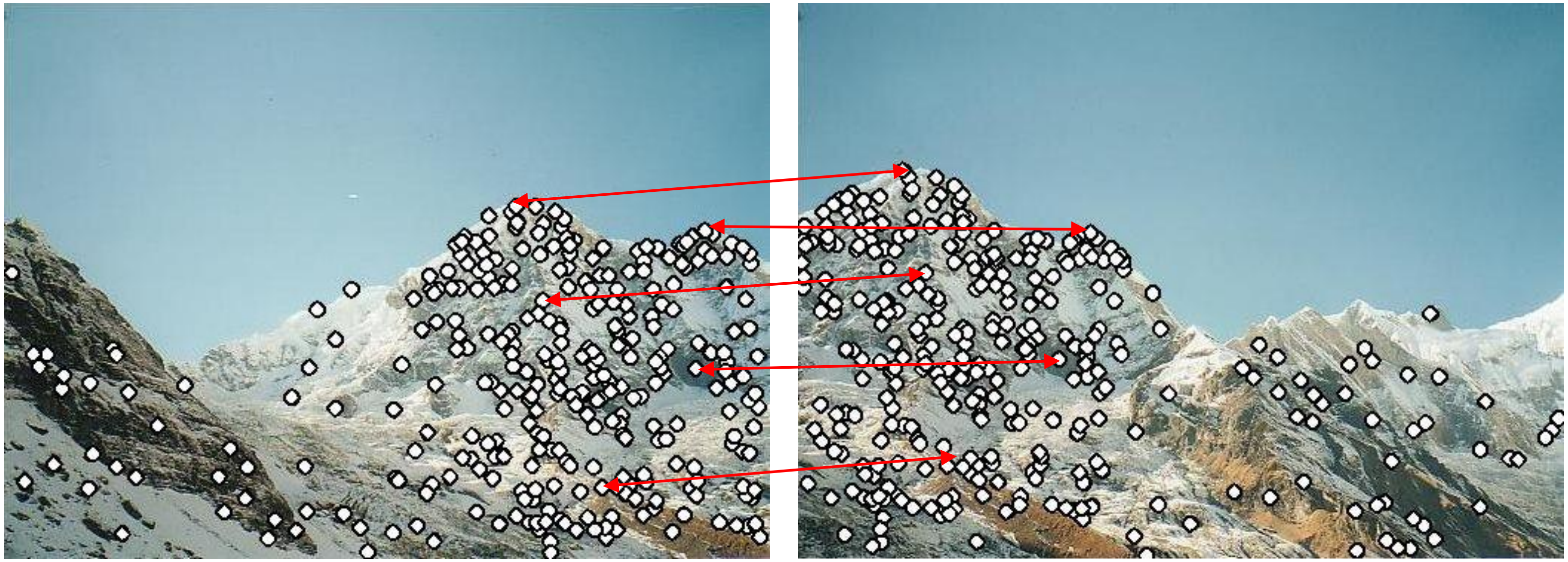
- Detect feature points in both images





Matching with Features

- Detect feature points in both images
- Find corresponding pairs



Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Matching with Features

- Detect feature points in both images
- Find corresponding pairs
- Use these pairs to align images

Feature matching

Feature matching

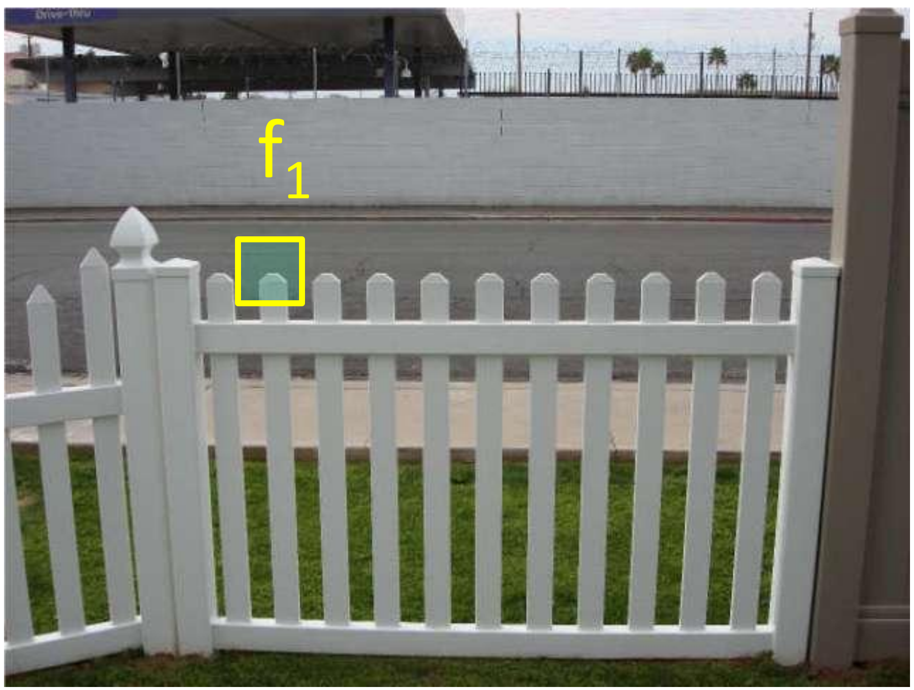
Given a feature in I_1, how to find the best match in I_2?

1. Define distance function that compares two descriptors
2. Test all the features in I_2, find the one with min distance

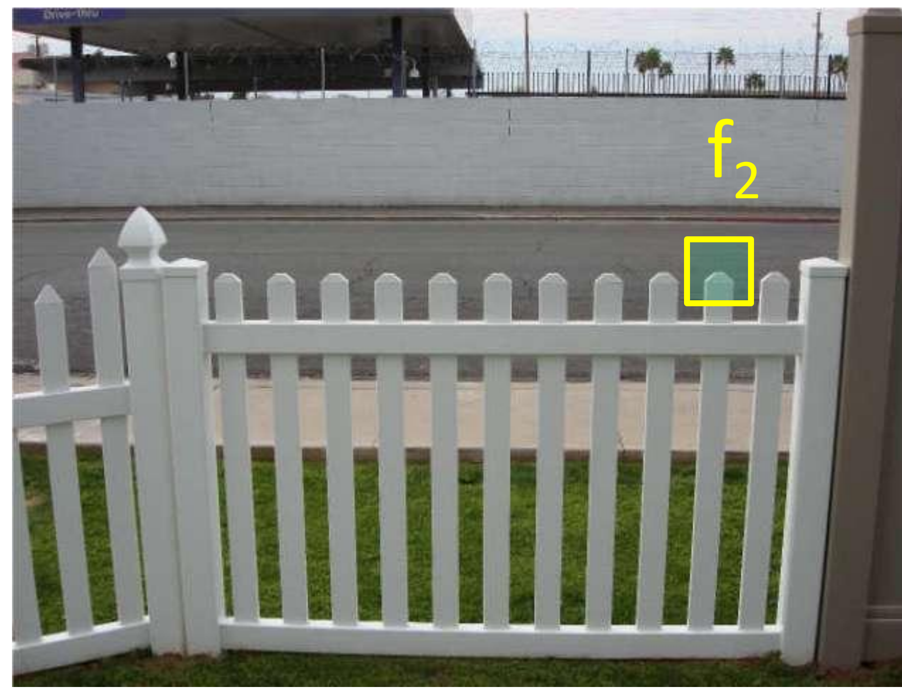
Feature distance

How to define the difference between two features f_1, f_2?

- **Simple approach: L_2 distance, $||f_1 - f_2||$**
- **can give good scores to ambiguous (incorrect) matches**



I_1

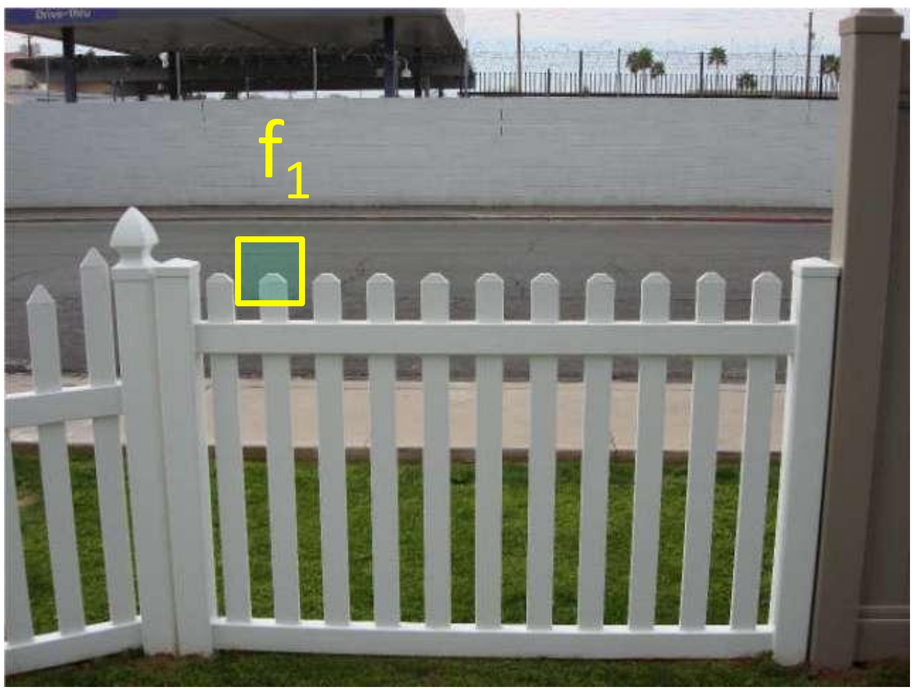


I_2

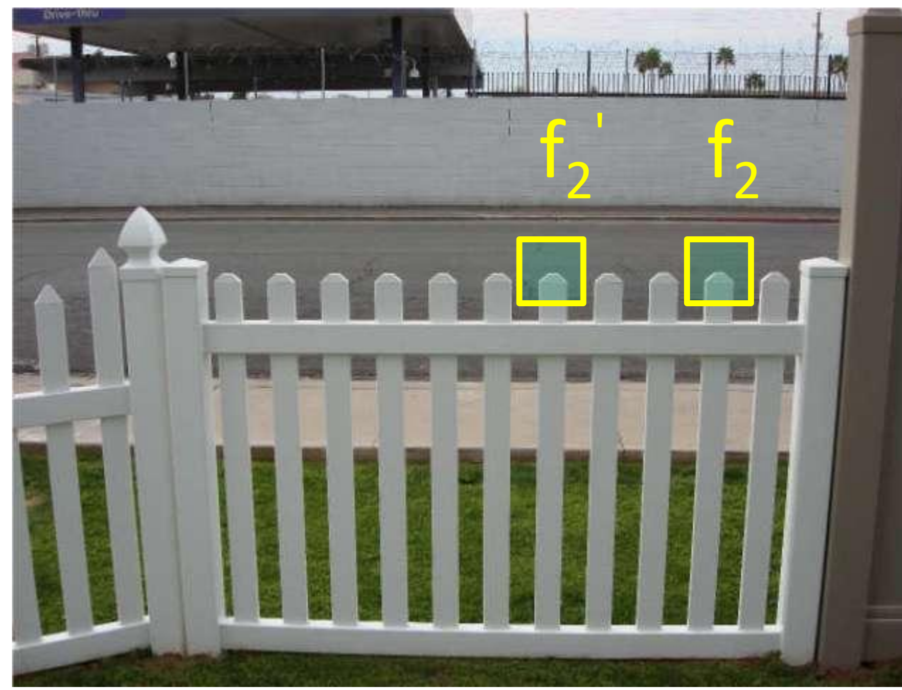
Feature distance

How to define the difference between two features f_1, f_2?

- Better approach: ratio distance = $||f_1 - f_2|| / || f_1 - f_2' ||$
  - f_2 is best SSD match to f_1 in I_2
  - f_2' is 2nd best SSD match to f_1 in I_2
  - gives large values for ambiguous matches



I_1



I_2

Matching Cost of Two Features

- Euclidean distance:

$$d(\mathbf{f_1}, \mathbf{f_2}) = \|\mathbf{f_1} - \mathbf{f_2}\|^2$$

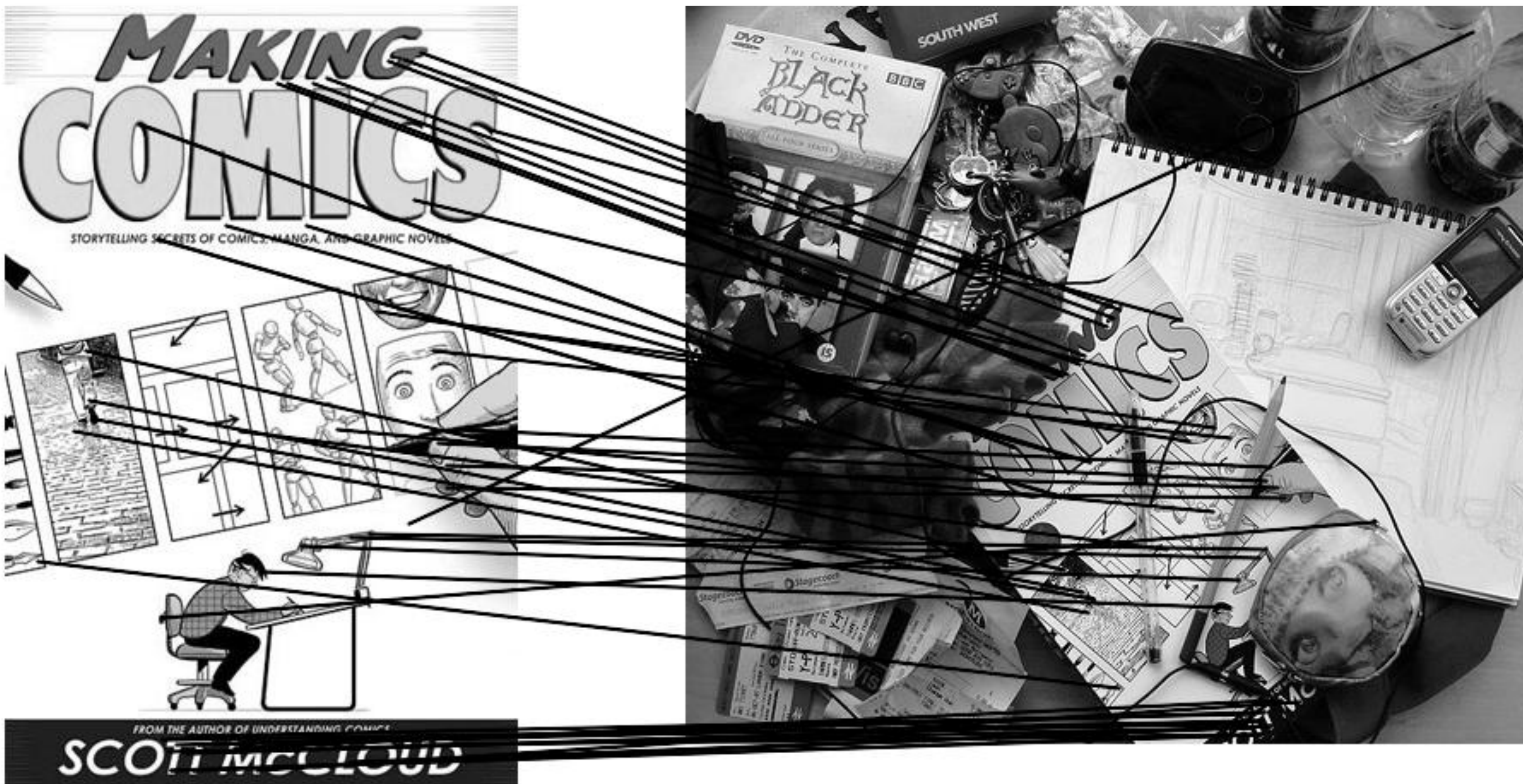
- Chi-squared distance:

$$d(\mathbf{f_1}, \mathbf{f_2}) = \sum_{\mathbf{i}} \frac{(\mathbf{f_1(i)} - \mathbf{f_2(i)})^2}{\mathbf{f_1(i)} + \mathbf{f_2(i)}}$$

- Hamming distance (only for binary features):

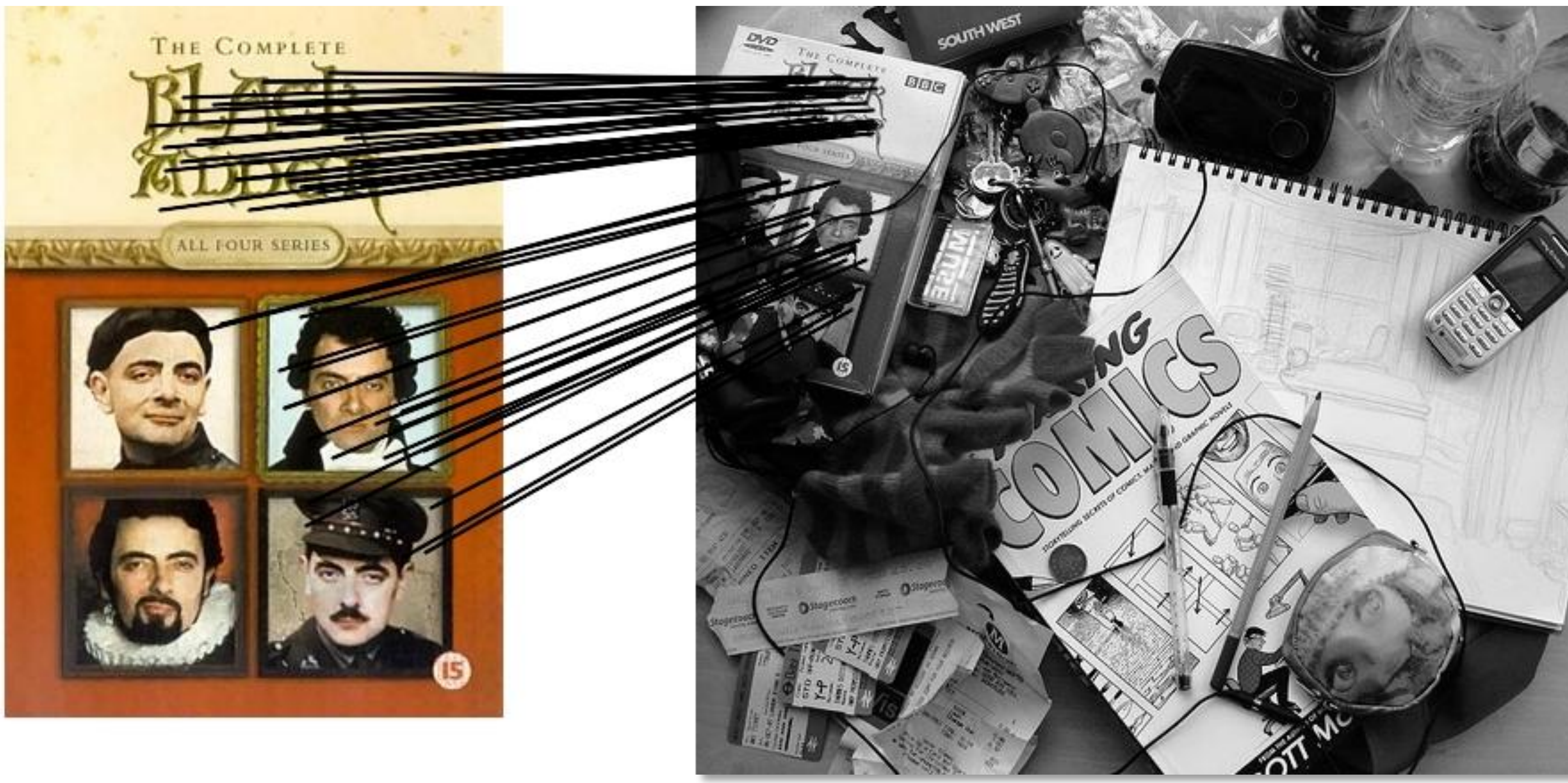
$$d(\mathbf{f_1}, \mathbf{f_2}) = \sum_{\mathbf{i}} \mathbf{1}(\mathbf{f_1(i)} \neq \mathbf{f_2(i)})$$

Feature matching example



51 matches

Feature matching example

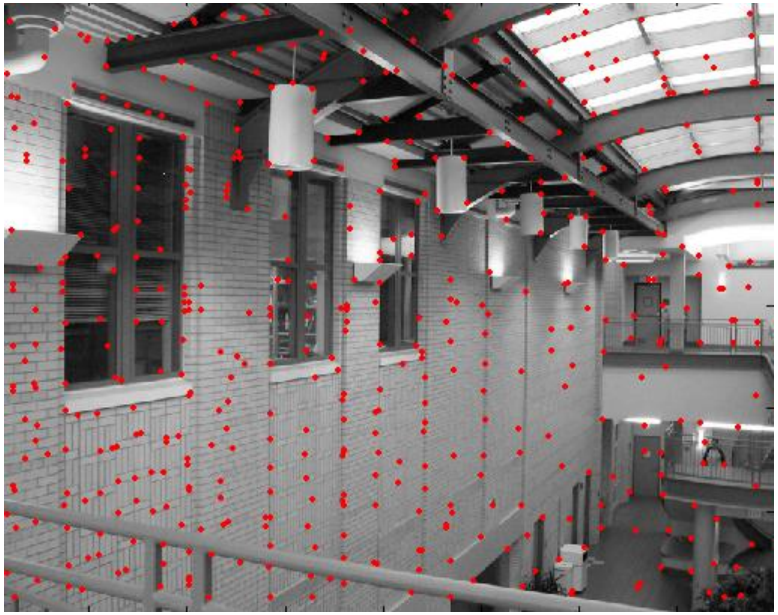


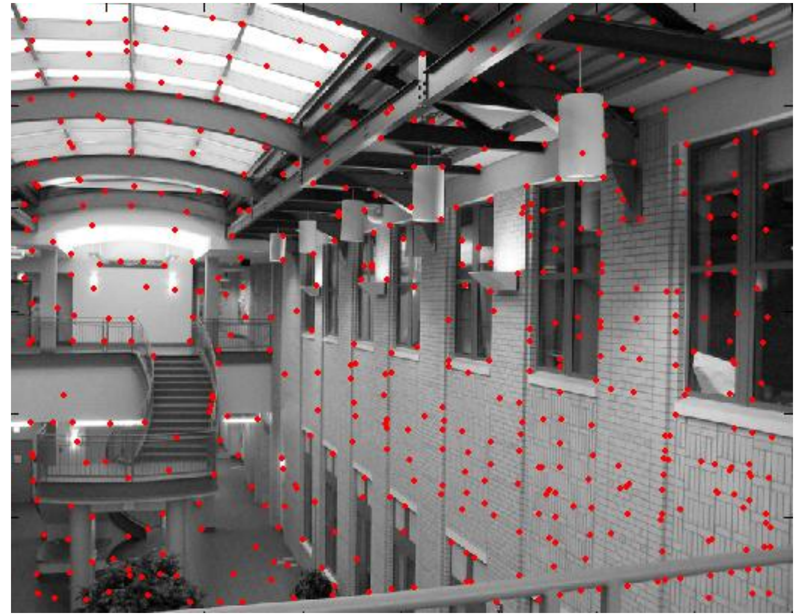
58 matches

Feature Matching

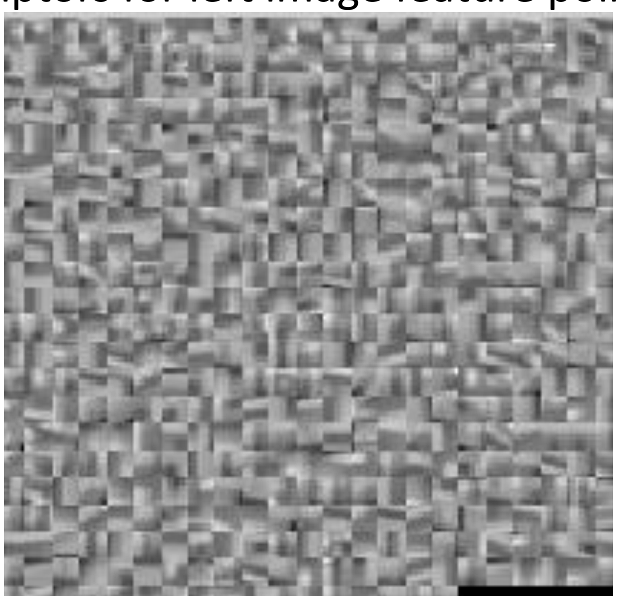
- Simple criteria: One feature matches to another if those features are nearest neighbors and their distance is below some threshold.
- Problems:
  - Threshold is difficult to set
  - Non-distinctive features could have lots of close matches, only one of which is correct

Feature matching



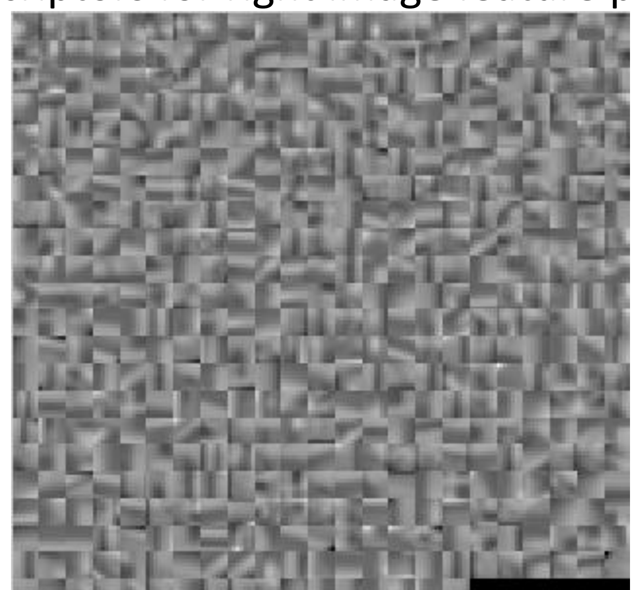


descriptors for left image feature points



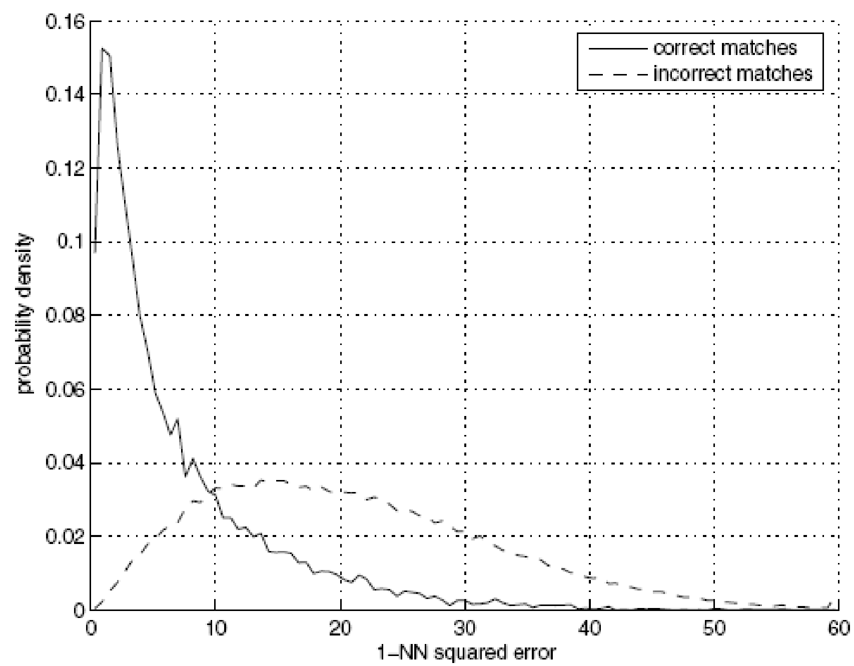
?

descriptors for right image feature points



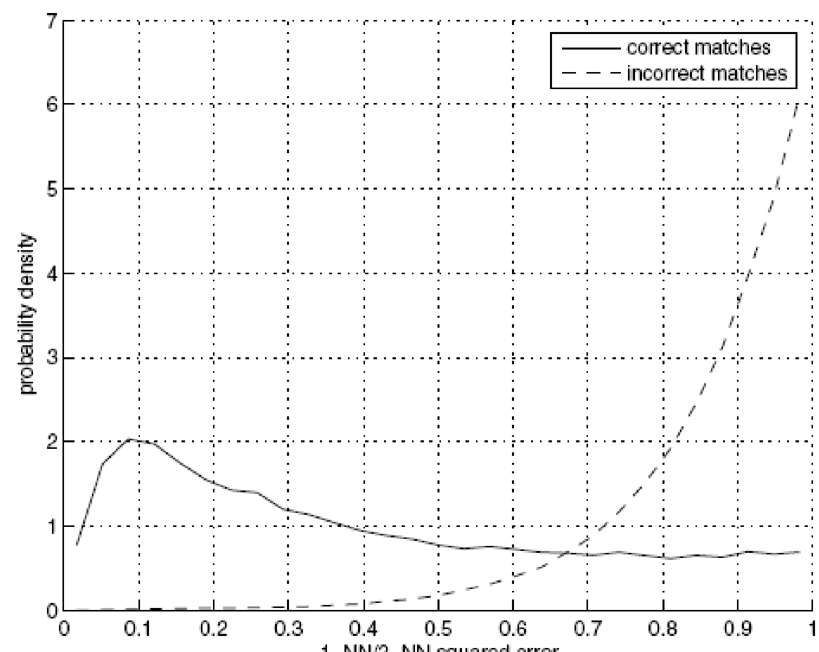
(a) Feature-space outlier rejection

- Let’s not match all features, but only these that have “similar enough” matches?
- How can we do it?
  - SSD(patch1,patch2) < threshold
  - How to set threshold?
    Not so easy.

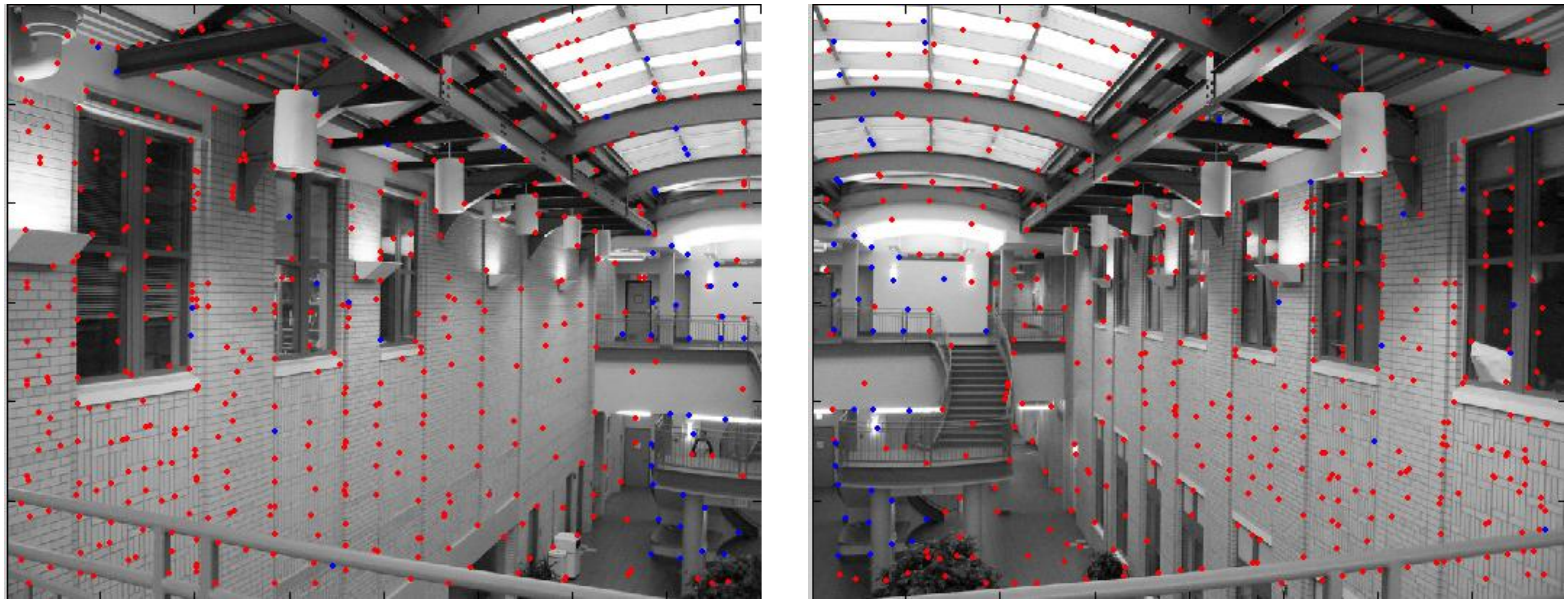


Feature-space outlier rejection

- **A better way [Lowe, 1999]:**
  - **1-NN: SSD of the closest match**
  - **2-NN: SSD of the <u>second-closest</u> match**
  - **Look at how much better 1-NN is than 2-NN, e.g. 1-NN/2-NN**
  - **That is, is our best match so much better than the rest?**



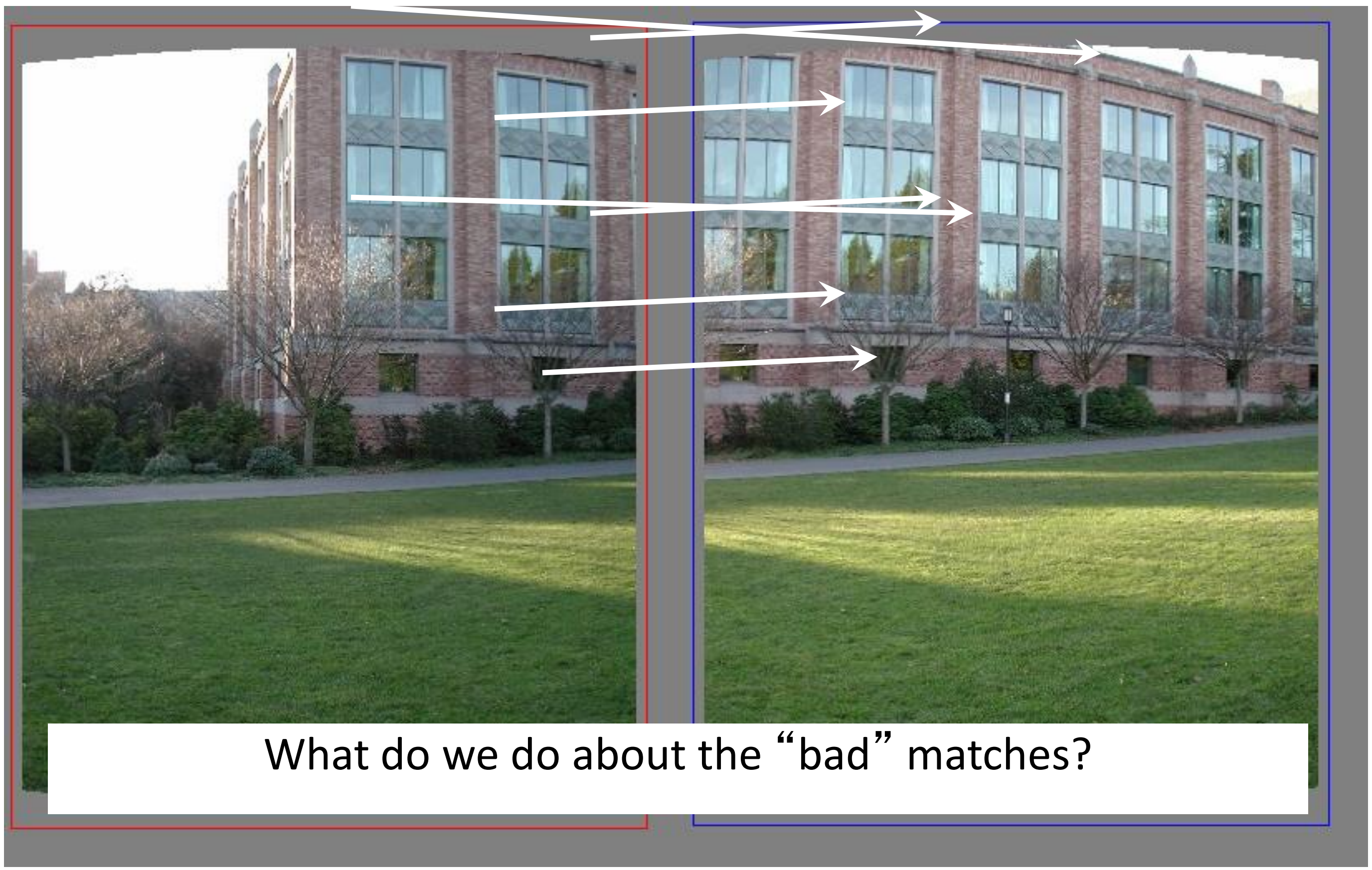
Feature-space outlier rejection



- **Can we now compute H from the blue points?**
  - **No! Still too many outliers…**
  - **What can we do?**

(b) Matching many features--looking for a good homography

Simplified illustration with translation instead of homography



What do we do about the “bad” matches?

Note: at this point we don’t know which ones are good/bad

Fitting

Fitting: find the parameters of a model that best fit the data

Goals:

- **Choose a parametric model to fit a certain quantity from data**
- **Estimate model parameters**

Fitting

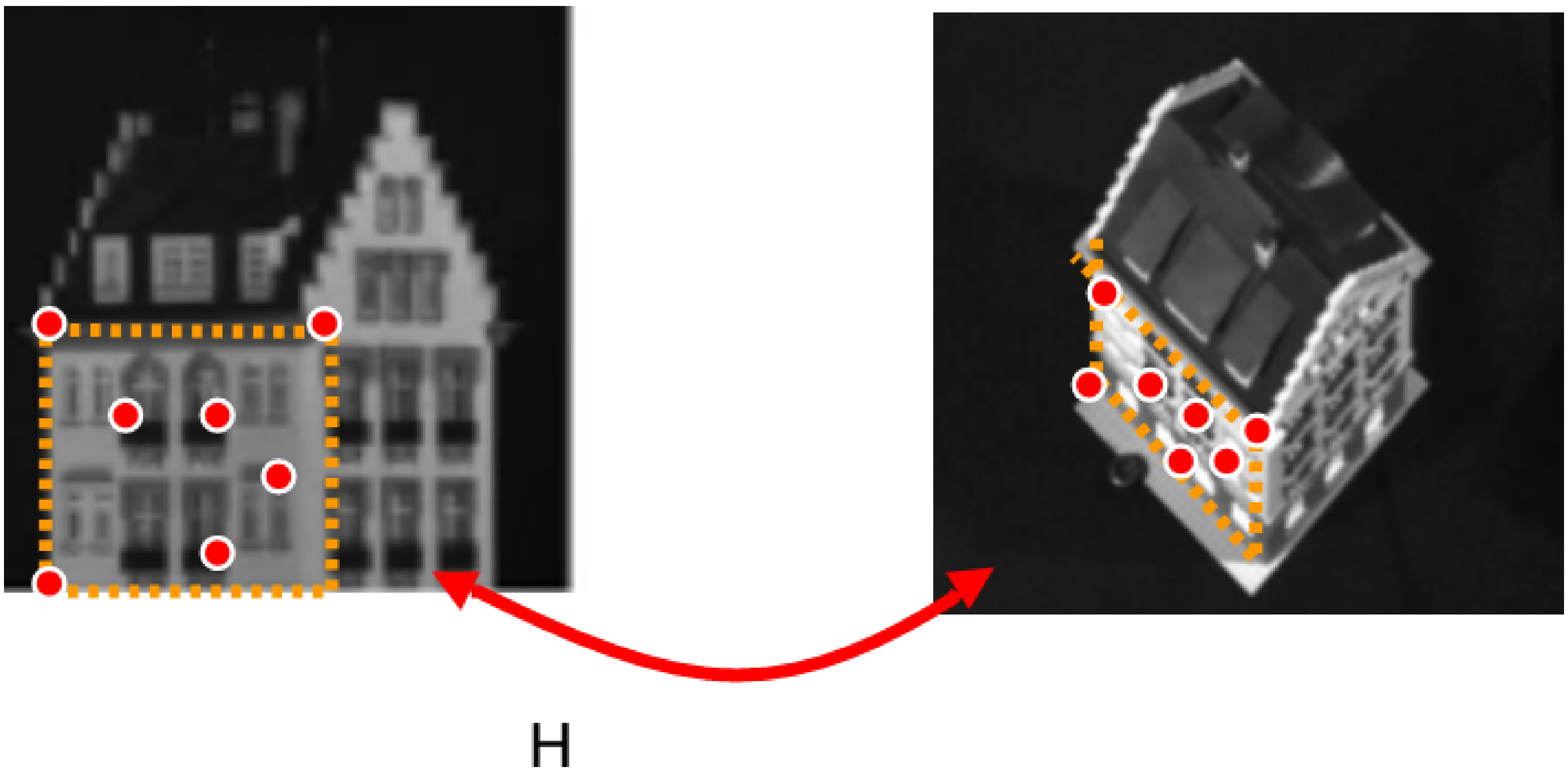
Goals:

- Choose a parametric model to fit a certain quantity from data
- Estimate model parameters

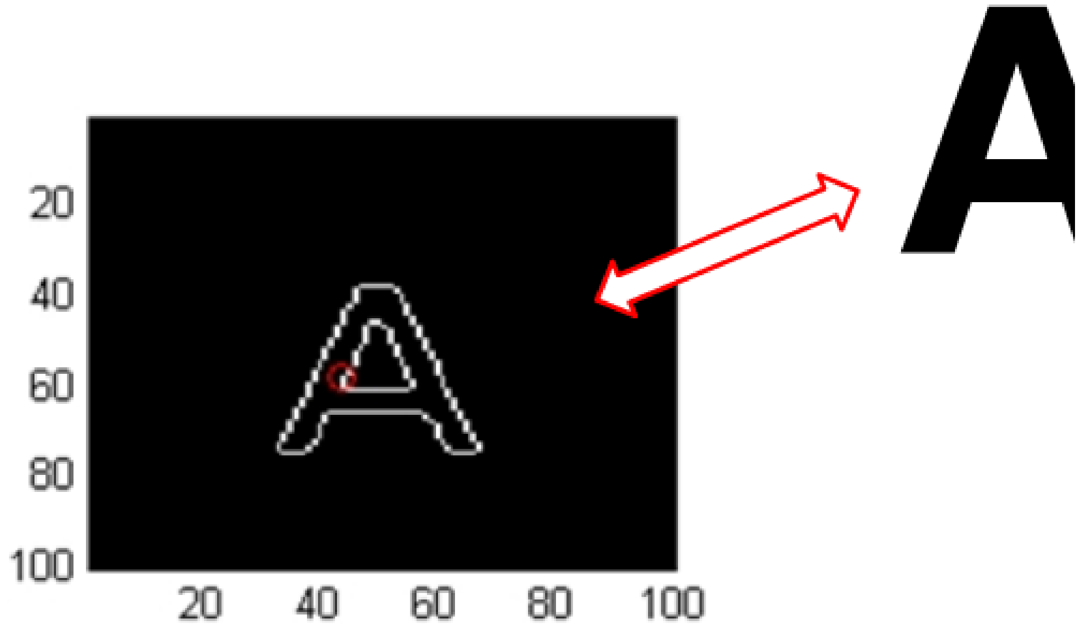
- Lines
- Curves
- Homographic transformation
- Fundamental matrix
- Shape model

Example: fitting lines
(for computing vanishing points)

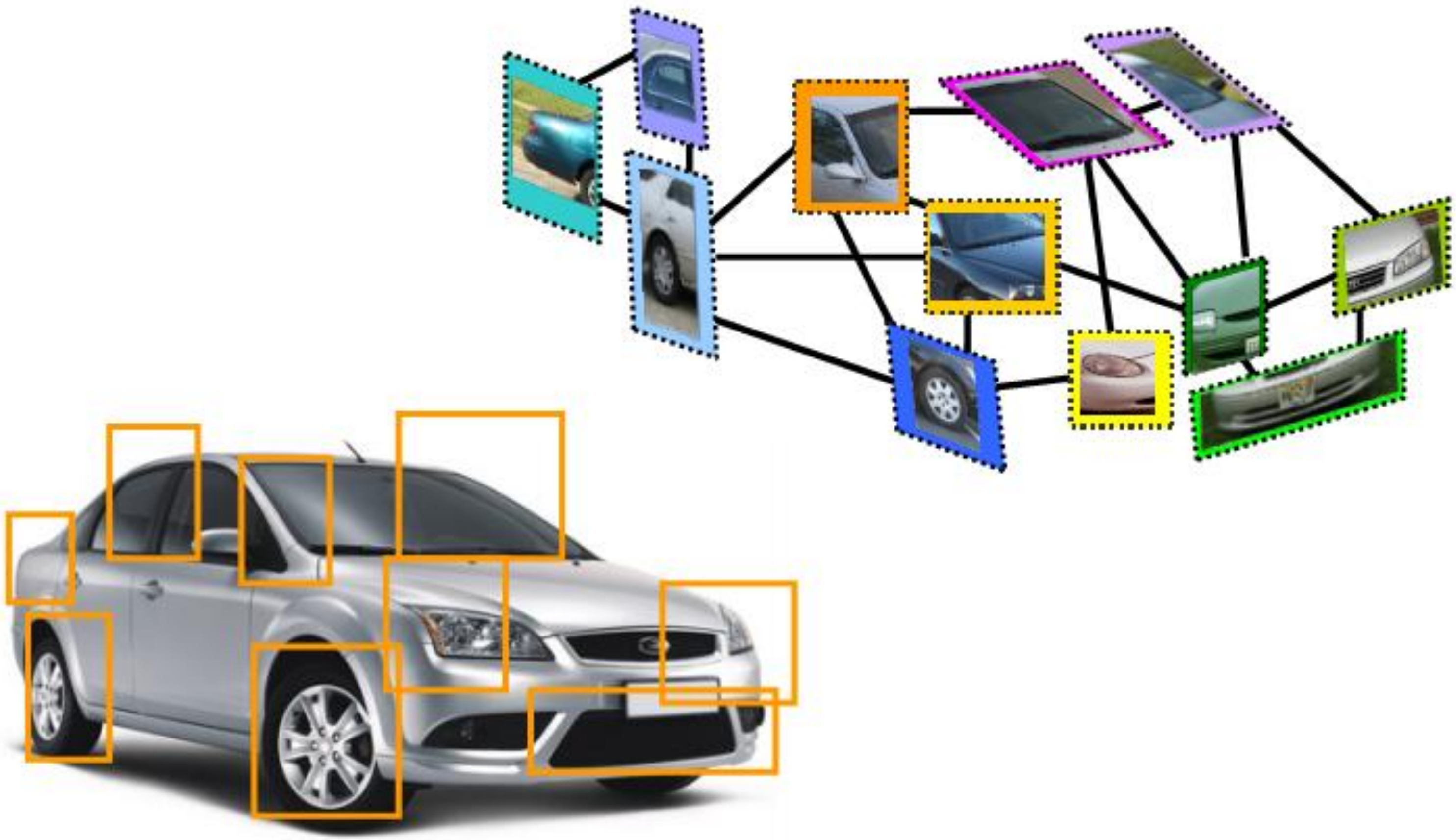
Example: Estimating an homographic transformation



Example: fitting a 2D shape template



Example: fitting a 3D object model



Fitting

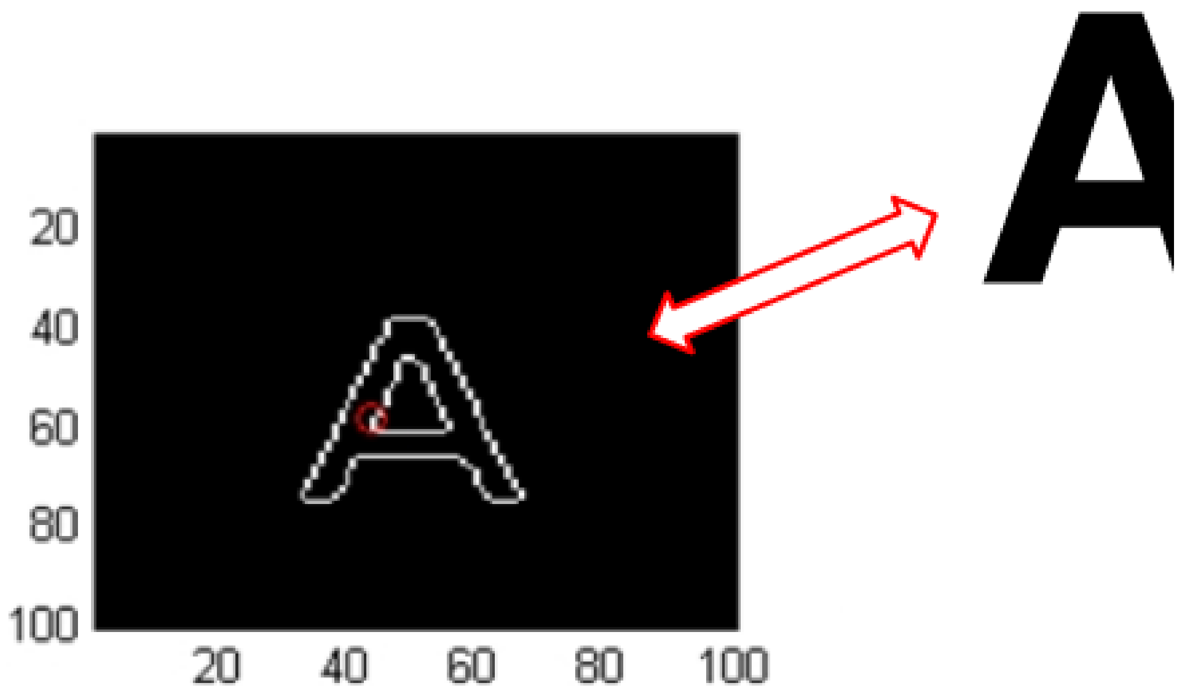
- Fitting, matching and recognition are interconnected problems

Fitting

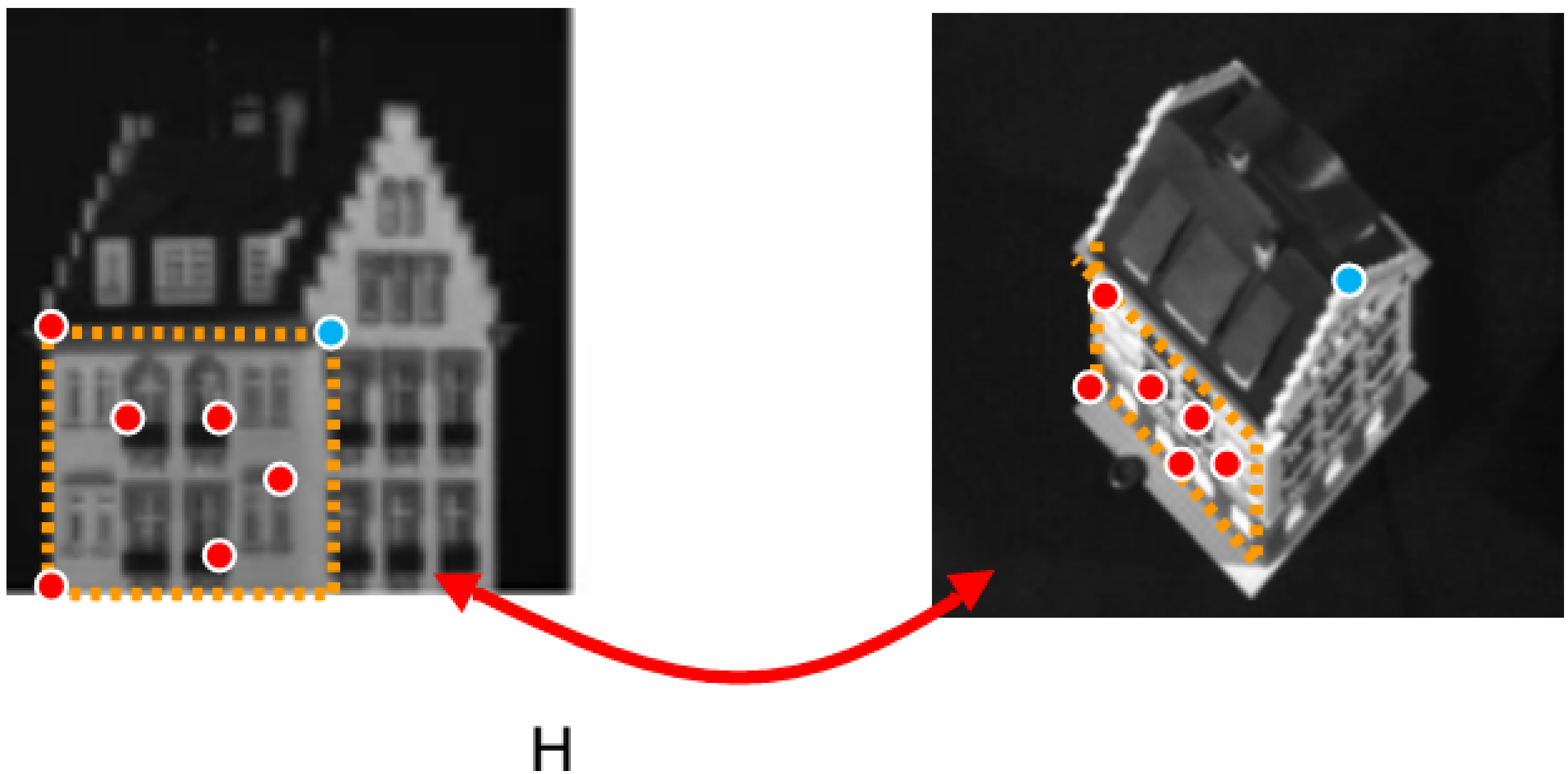
- **Critical issues:**
  - noisy data
  - outliers
  - missing data

Critical issues: noisy data

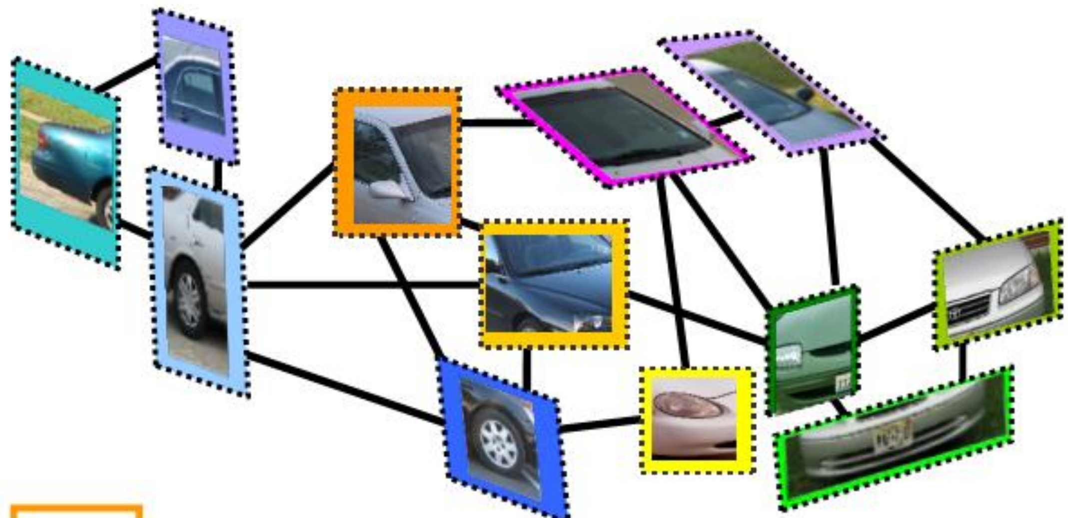
Critical issues: noisy data (intra-class variability)

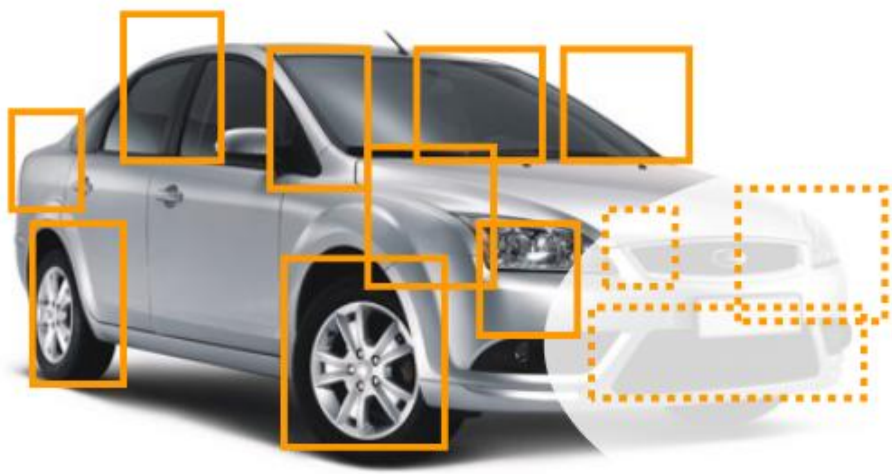


Critical issues: outliers



Critical issues: missing data (occlusions)





Fitting

Goal: Choose a parametric model to fit a certain quantity from data

Techniques:

– Least square methods
– Robust estimation
– Hough transform
– RANSAC
– EM (Expectation Maximization) [not covered]

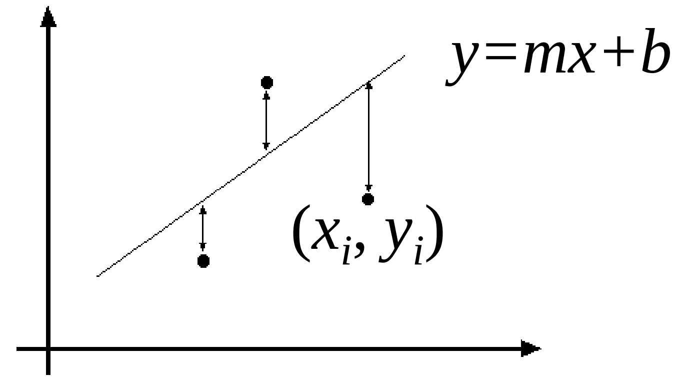
Fitting

- If we know which points belong to the line, how do we find the "optimal" line parameters?
  - Least squares
- What if there are outliers?
  - Robust fitting, RANSAC
- What if there are many lines?
  - Voting methods: RANSAC, Hough transform
- What if we're not even sure it's a line?
  - Model selection

Least squares line fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m\,x_i + b$
- Find (m, b) to minimize

$$\boxed{E = \sum_{i=1}^{n} (y_i - m x_i - b)^2}$$



$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

$$E = \|Y - XB\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0$$

$$\boxed{X^T XB = X^T Y}$$

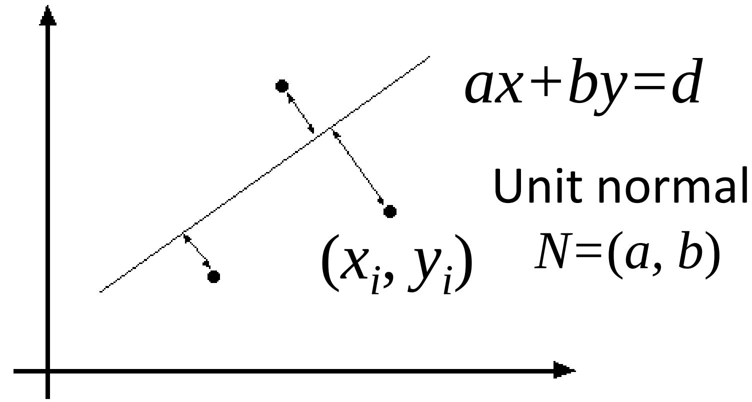
Normal equations: least squares solution to $XB = Y$

Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines

Total least squares

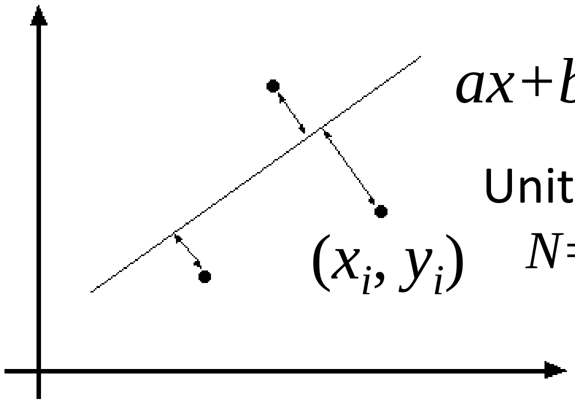
- **Distance between point (x_i, y_i) and line $ax+by=d$ ($a^2+b^2=1$): $|ax_i + by_i - d|$**



Total least squares

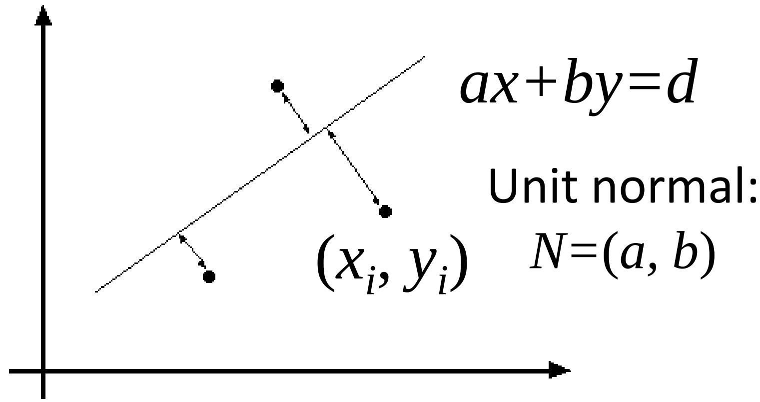
- **Distance between point (x_i, y_i) and line $ax+by=d$ ($a^2+b^2=1$): $|ax_i + by_i - d|$**
- **Find (a, b, d) to minimize the sum of squared perpendicular distances**

$$\boxed{E = \sum_{i=1}^{n} (ax_i + by_i - d)^2}$$



Total least squares

- Distance between point (x_i, y_i) and line $ax+by=d$ $(a^2+b^2=1)$: $|ax_i + by_i - d|$
- Find (a, b, d) to minimize the sum of squared perpendicular distances



$$\boxed{E = \sum_{i=1}^{n} (ax_i + by_i - d)^2}$$

$$\frac{\partial E}{\partial d} = \sum_{i=1}^{n} -2(ax_i + by_i - d) = 0 \qquad d = \frac{a}{n}\sum_{i=1}^{n} x_i + \frac{b}{n}\sum_{i=1}^{n} y_i = a\bar{x} + b\bar{y}$$

$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = (UN)^T (UN)$$

$$\frac{dE}{dN} = 2(U^T U)N = 0$$

Solution to $(U^T U)N = 0$, subject to $\|N\|^2 = 1$: eigenvector of $U^T U$ associated with the smallest eigenvalue (least squares solution to *homogeneous linear system* $UN = 0$)

Recap: Two Common Optimization Problems

Problem statement	Solution
$\text{minimize } \lVert \mathbf{A}\mathbf{x} - \mathbf{b} \rVert^2$	$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$
$\text{least squares solution to } \mathbf{A}\mathbf{x} = \mathbf{b}$	$\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$ (matlab)

Problem statement	**Solution**
$\text{minimize } \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} \quad \text{s.t. } \mathbf{x}^T\mathbf{x} = 1$	$[\mathbf{v}, \lambda] = \text{eig}(\mathbf{A}^T\mathbf{A})$
$\text{minimize} \dfrac{\mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}}$	$\lambda_1 < \lambda_{2..n} : \mathbf{x} = \mathbf{v}_1$
$\text{non-trivial lsq solution to } \mathbf{M}\mathbf{x} = 0$	

Search / Least squares conclusions

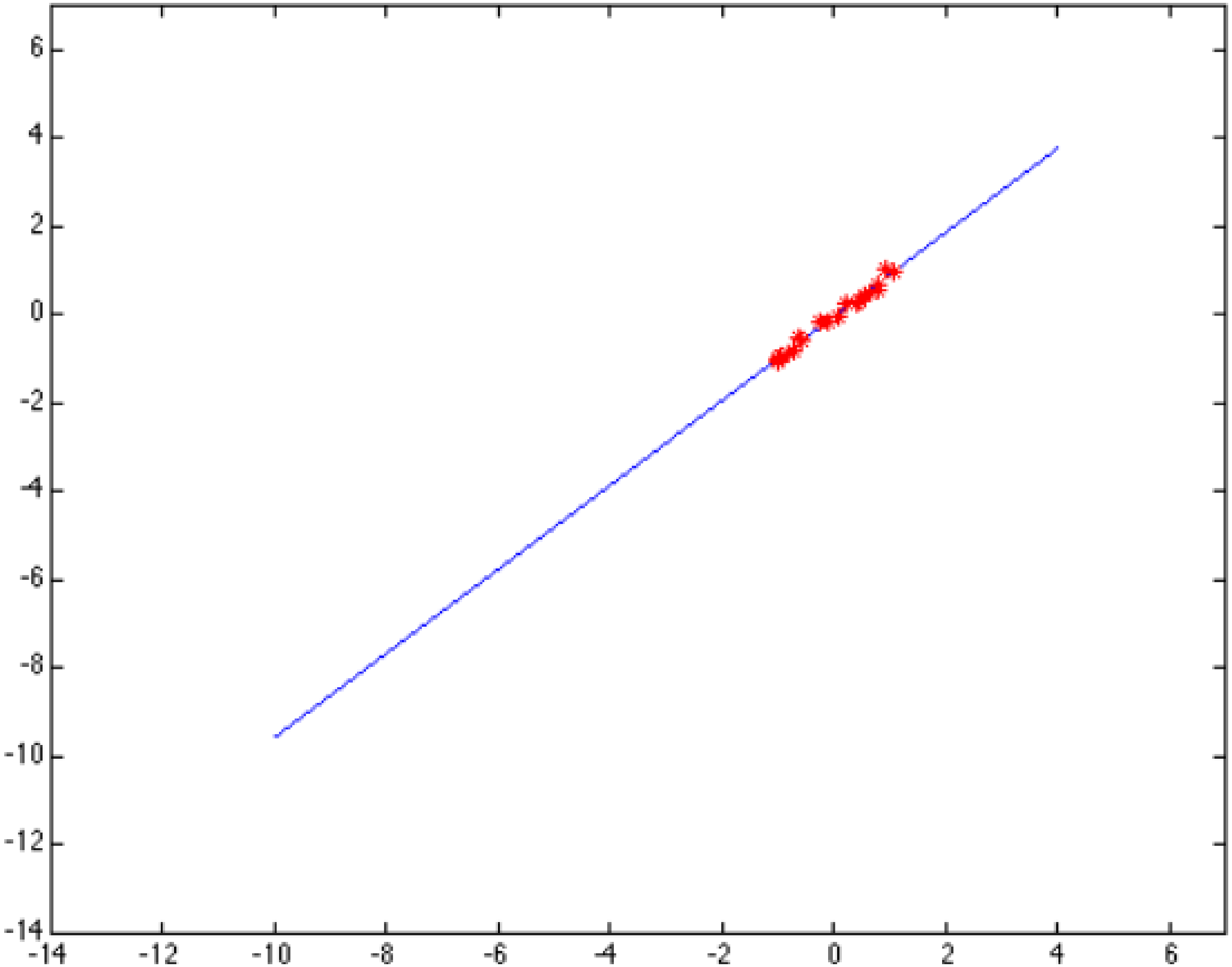
Good

- Clearly specified objective
- Optimization is easy (for least squares)

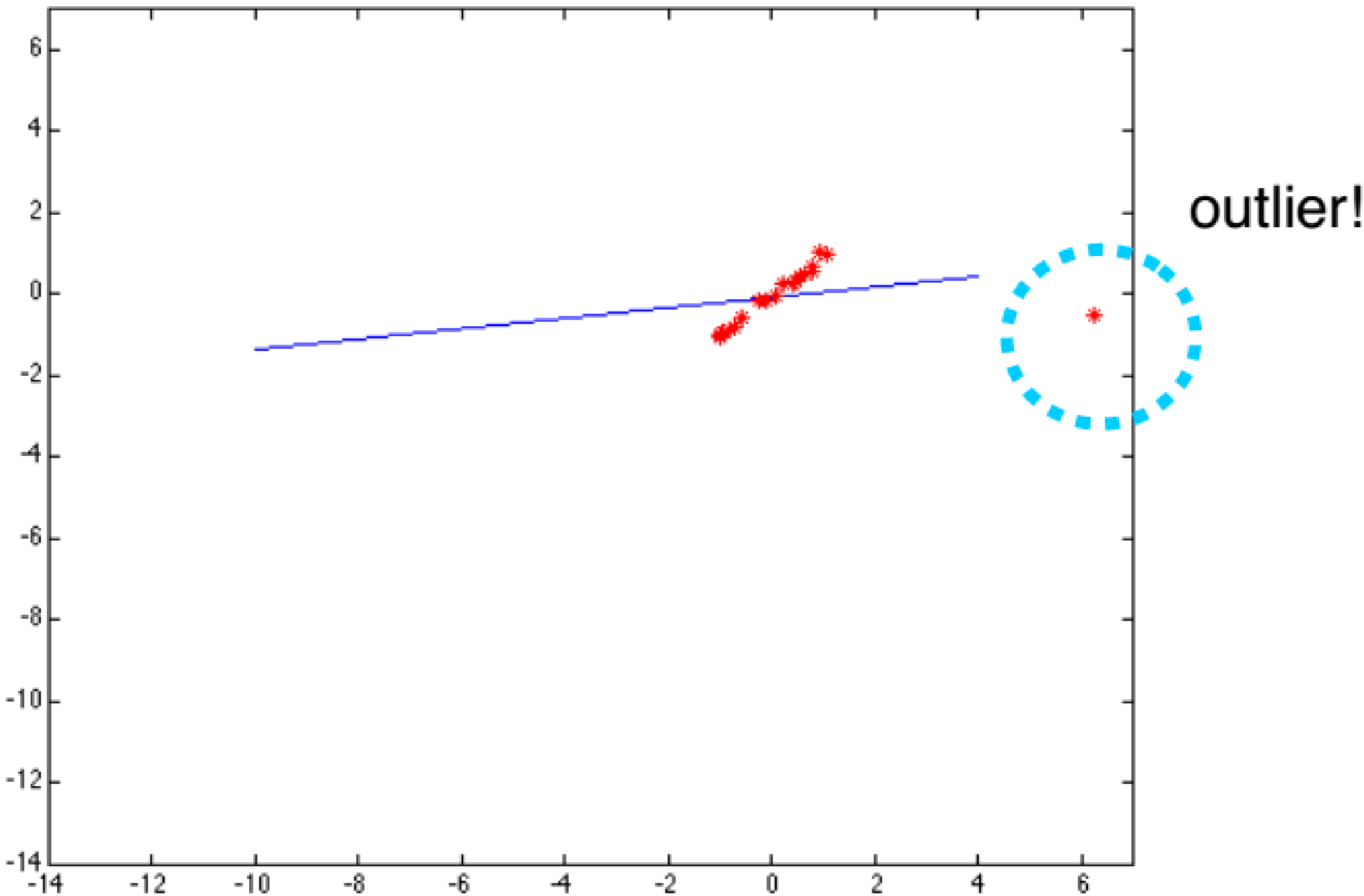
Bad

- Not appropriate for non-convex objectives
  - May get stuck in local minima
- Sensitive to outliers
  - Bad matches, extra points
- Doesn't allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

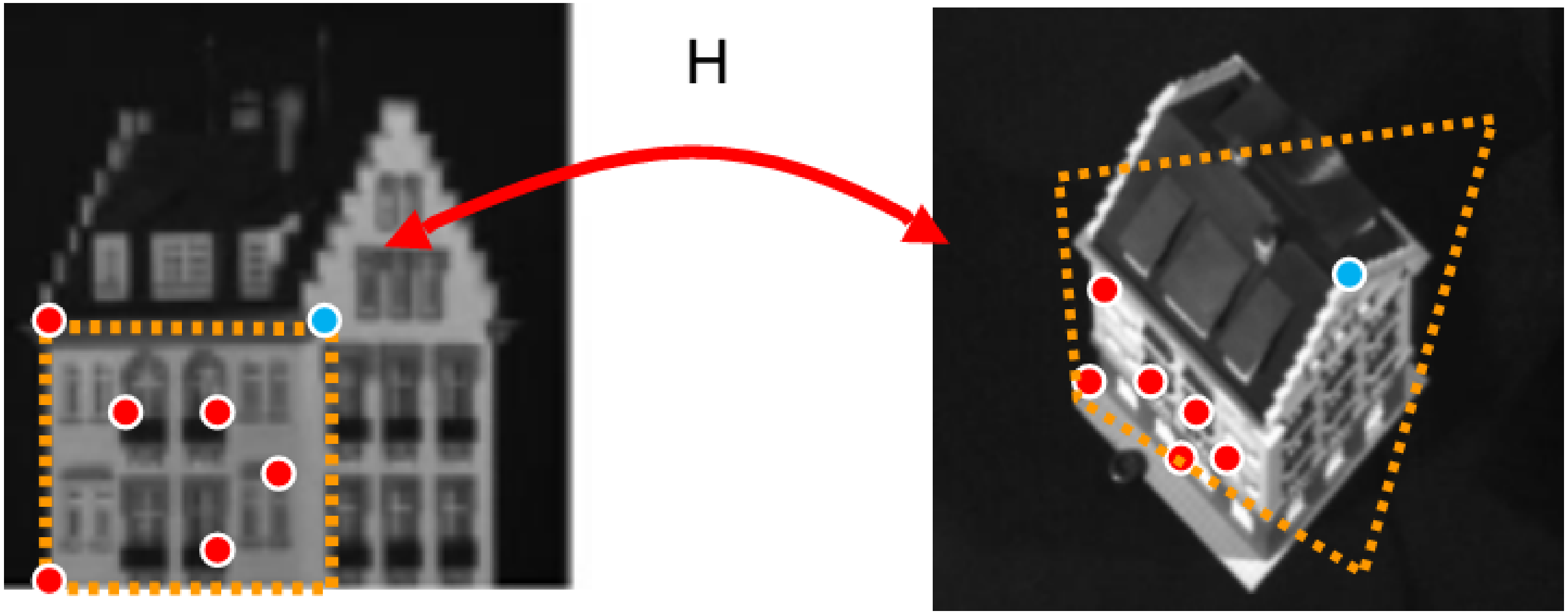
Least squares: Robustness to noise



Least squares: Robustness to noise



Critical issues: outliers



CONCLUSION: Least square is not robust w.r.t. outliers

Robust Statistics

- Recover the best fit to the majority of the data.
- Detect and reject outliers.

Total least squares

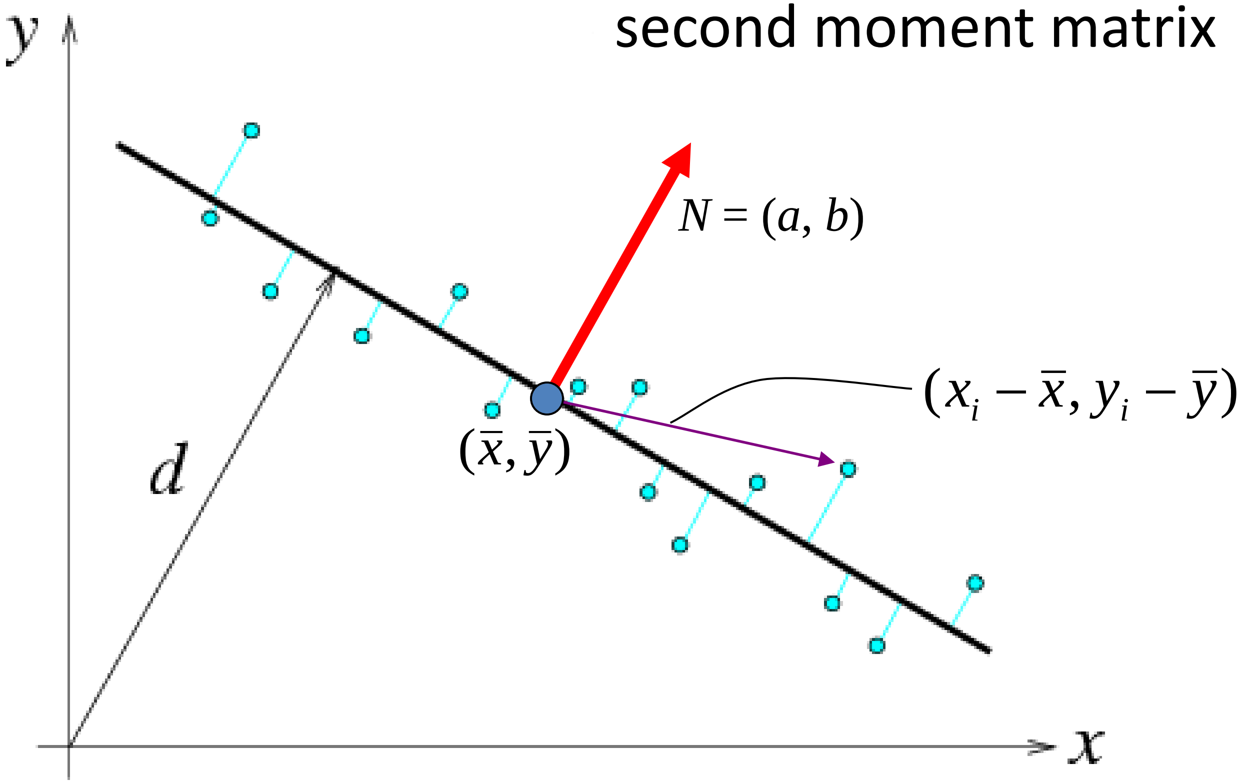
$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \quad U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2 \end{bmatrix}$$

second moment matrix

Total least squares

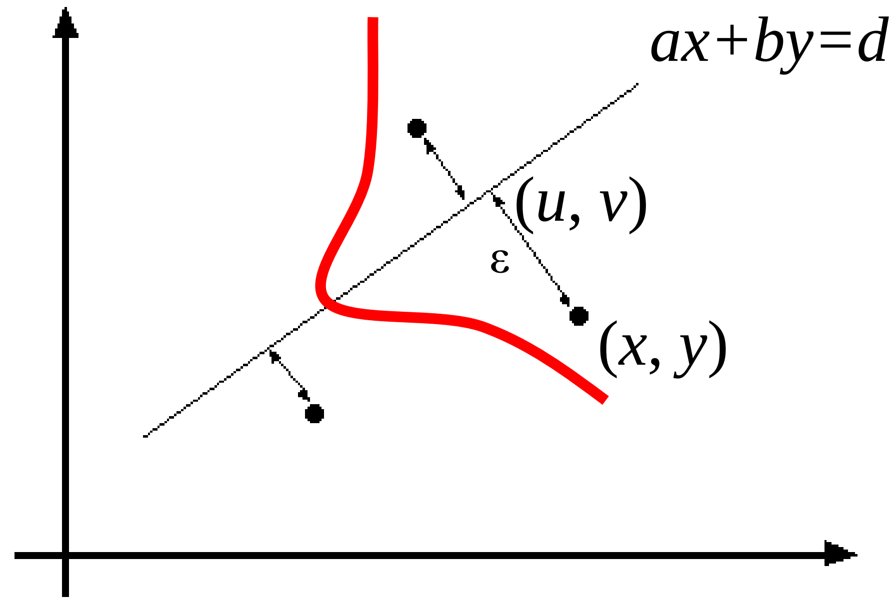
$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \quad U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2 \end{bmatrix}$$

second moment matrix



Least squares as likelihood maximization

- **Generative model: line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line**



$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \varepsilon \begin{pmatrix} a \\ b \end{pmatrix}$$

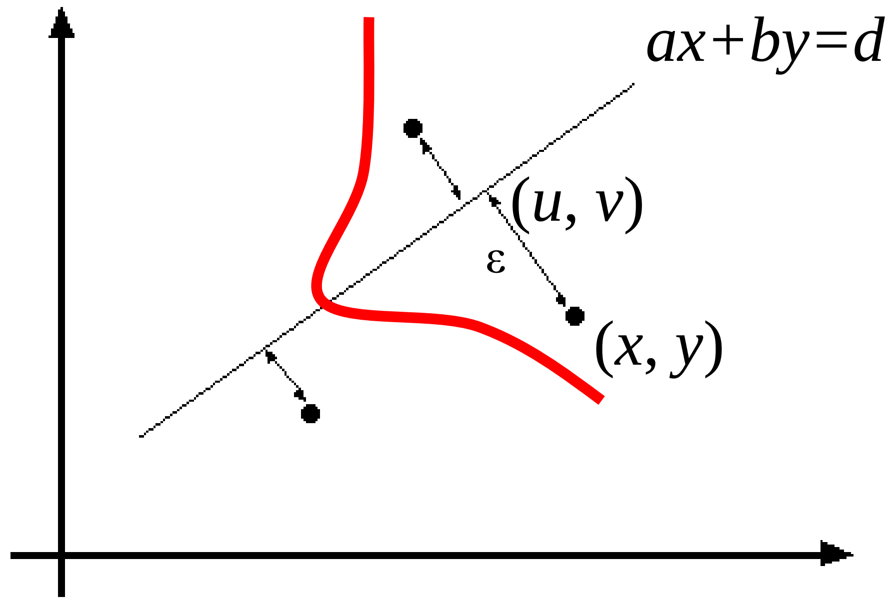
point on the line

noise: sampled from zero-mean Gaussian with std. dev. σ

normal direction

Least squares as likelihood maximization

- **Generative model**: line points are sampled independently and corrupted by Gaussian noise in the direction perpendicular to the line



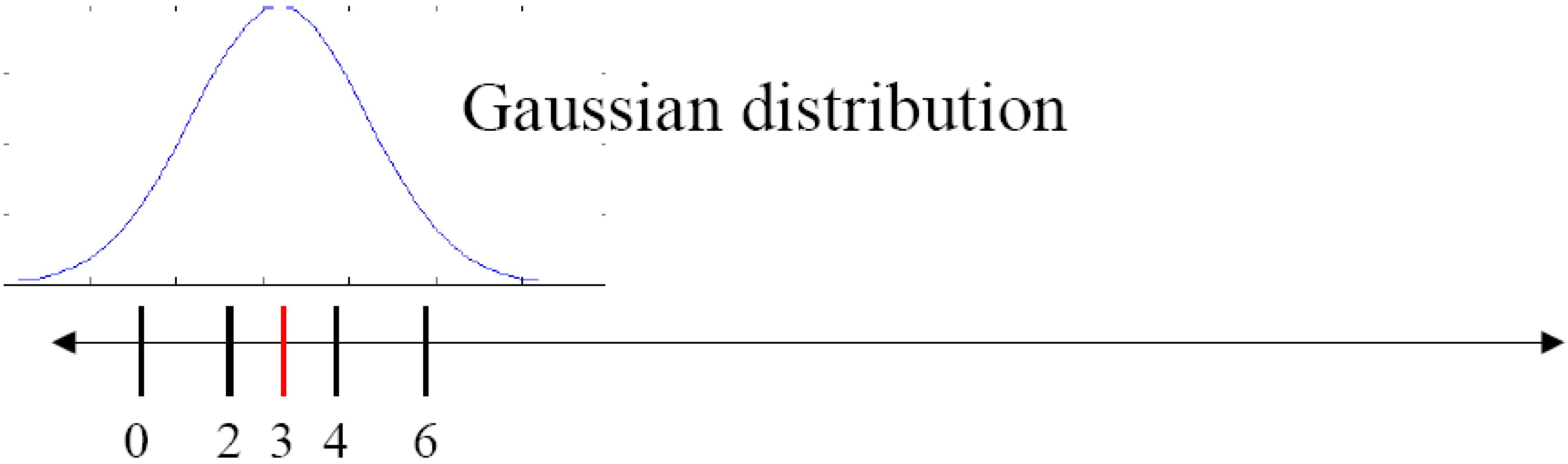
$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} + \varepsilon \begin{pmatrix} a \\ b \end{pmatrix}$$

Likelihood of points given line parameters (*a*, *b*, *d*):

$$P(x_1, y_1, \ldots, x_n, y_n \mid a, b, d) = \prod_{i=1}^{n} P(x_i, y_i \mid a, b, d) \propto \prod_{i=1}^{n} \exp\left(-\frac{(ax_i + by_i - d)^2}{2\sigma^2} \right)$$

Log-likelihood: $L(x_1, y_1, \ldots, x_n, y_n \mid a, b, d) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (ax_i + by_i - d)^2$

Estimating the mean



Gaussian distribution

Mean is the optimal solution to:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} d_i$$

$$\min_{\mu} \sum_{i=1}^{N} (\underbrace{d_i - \mu}_{\text{residual}})^2$$

Estimating the mean

The mean maximizes this likelihood:

$$\max_{\mu} p(d_i \mid \mu) = \frac{1}{\sqrt{2\pi\sigma}} \prod_{i=1}^{N} \exp(-\frac{1}{2}(d_i - \mu)^2 / \sigma^2)$$

The negative log gives (with sigma=1):

$$\min_{\mu} \sum_{i=1}^{N} (d_i - \mu)^2$$

"least squares" estimate

Estimating the mean

Estimating the mean

What happens if we change just one measurement?

$$\mu' = \mu + \frac{\Delta}{N}$$

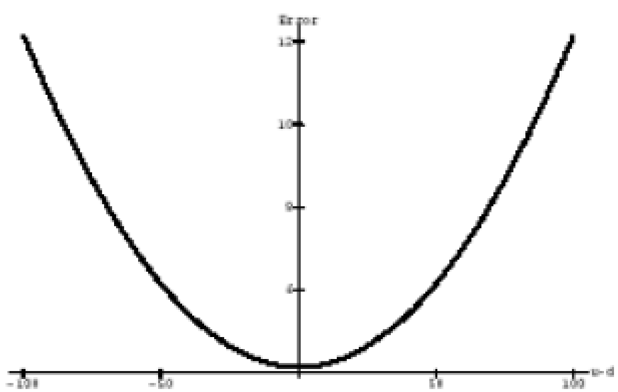
With a single "bad" data point I can move the mean arbitrarily far.

Influence

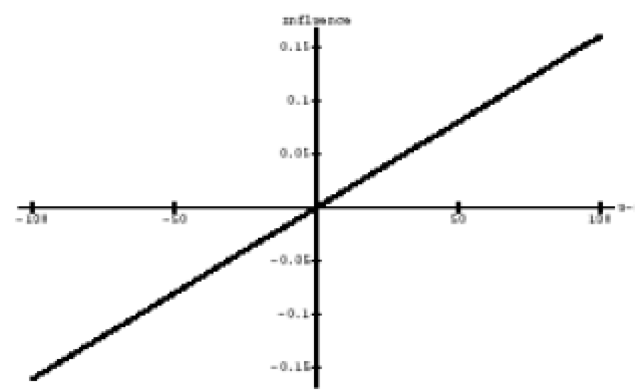
What's Wrong?

$$\min_{\mu} \sum_{i=1}^{N} (d_i - \mu)^2$$

Outliers (large residuals) have too much influence.



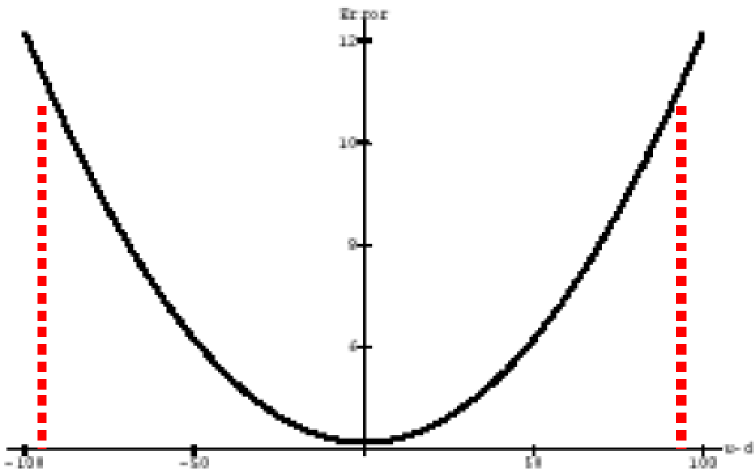
$$\rho(x) = x^2$$

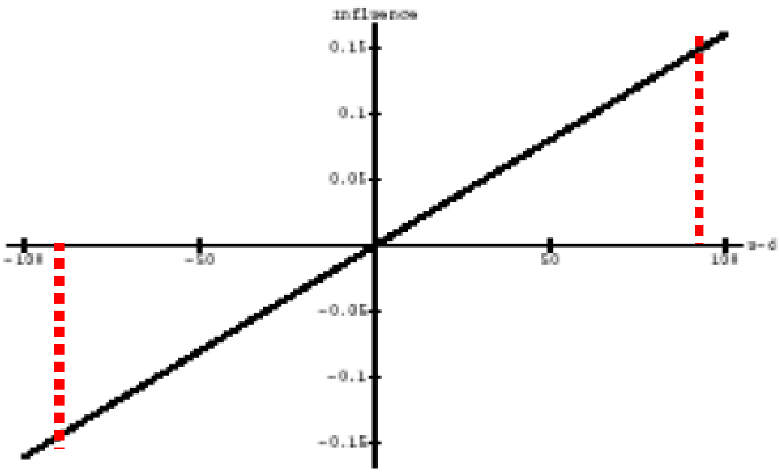


$$\psi(x) = 2x$$

Approach

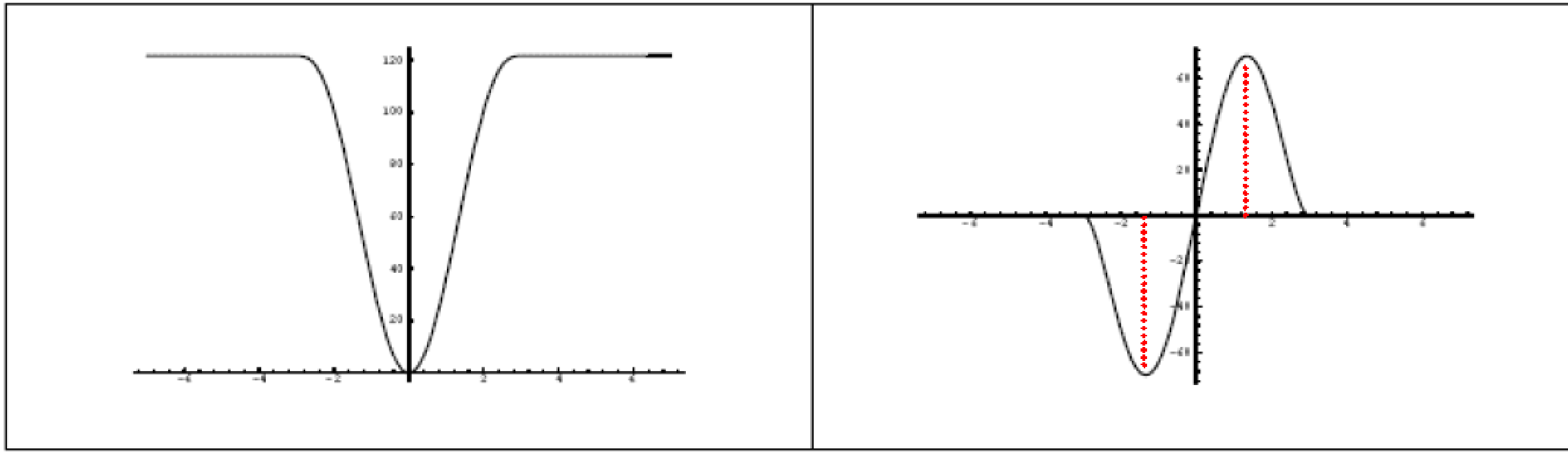
Influence is proportional to the derivative of the ρ function.





Want to give less influence to points beyond some value.

Redescending Function



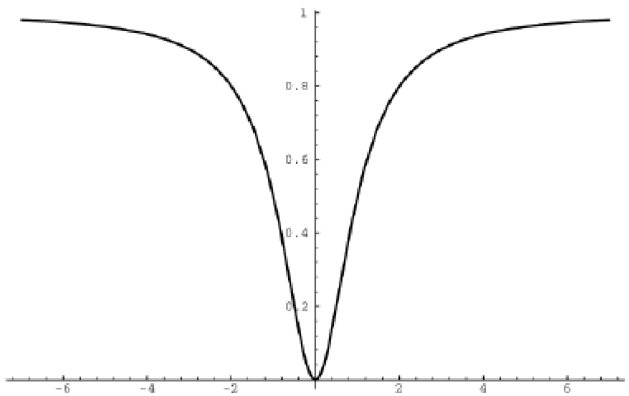
Tukey's biweight.

Beyond a point, the influence begins to decrease.

Beyond where the second derivative is zero – outlier points

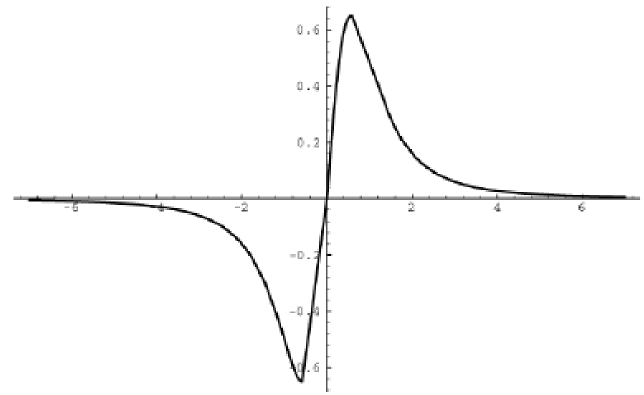
Robust Estimation

Geman-McClure function works well.
Twice differentiable, redescending.

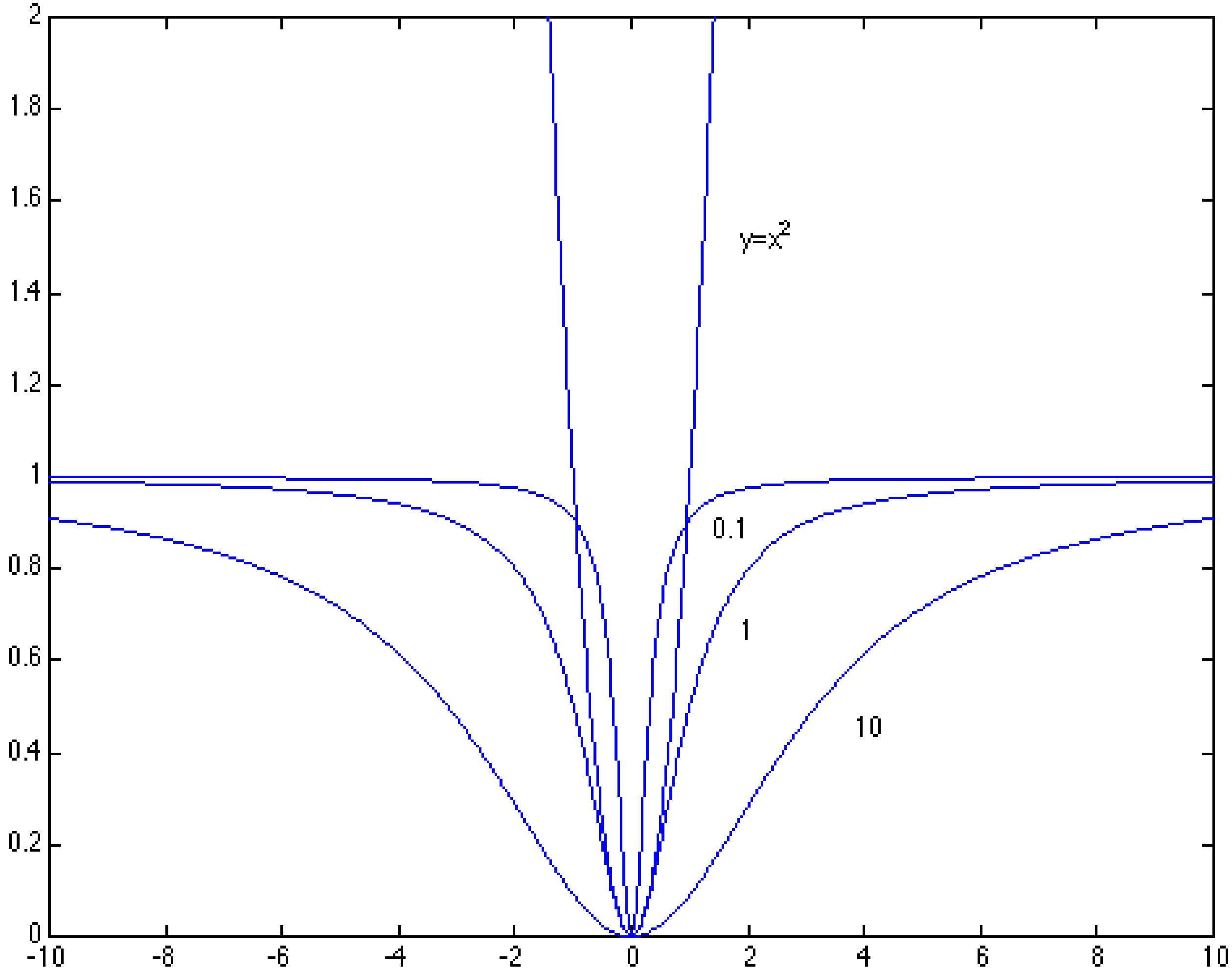


$$\rho(r,\sigma) = \frac{r^2}{\sigma^2 + r^2}$$

Influence function
(d/dr of norm):



$$\psi(r,\sigma) = \frac{2r\sigma^2}{(\sigma^2 + r^2)^2}$$



2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-10
-8
-6
-4
-2
0
2
4
6
8
10
$y=x^2$
0.1
1
10

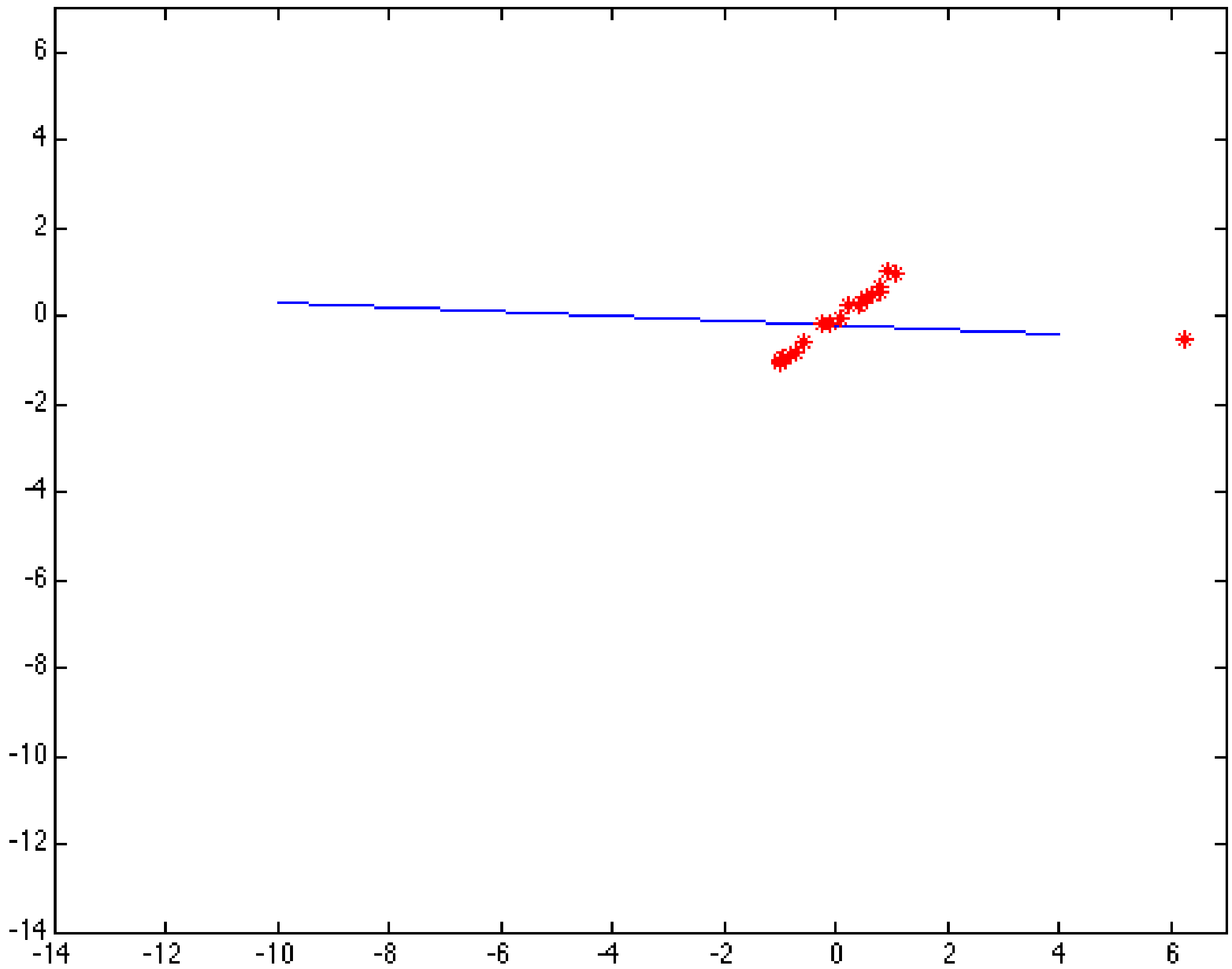
Robust scale

Scale is critical!

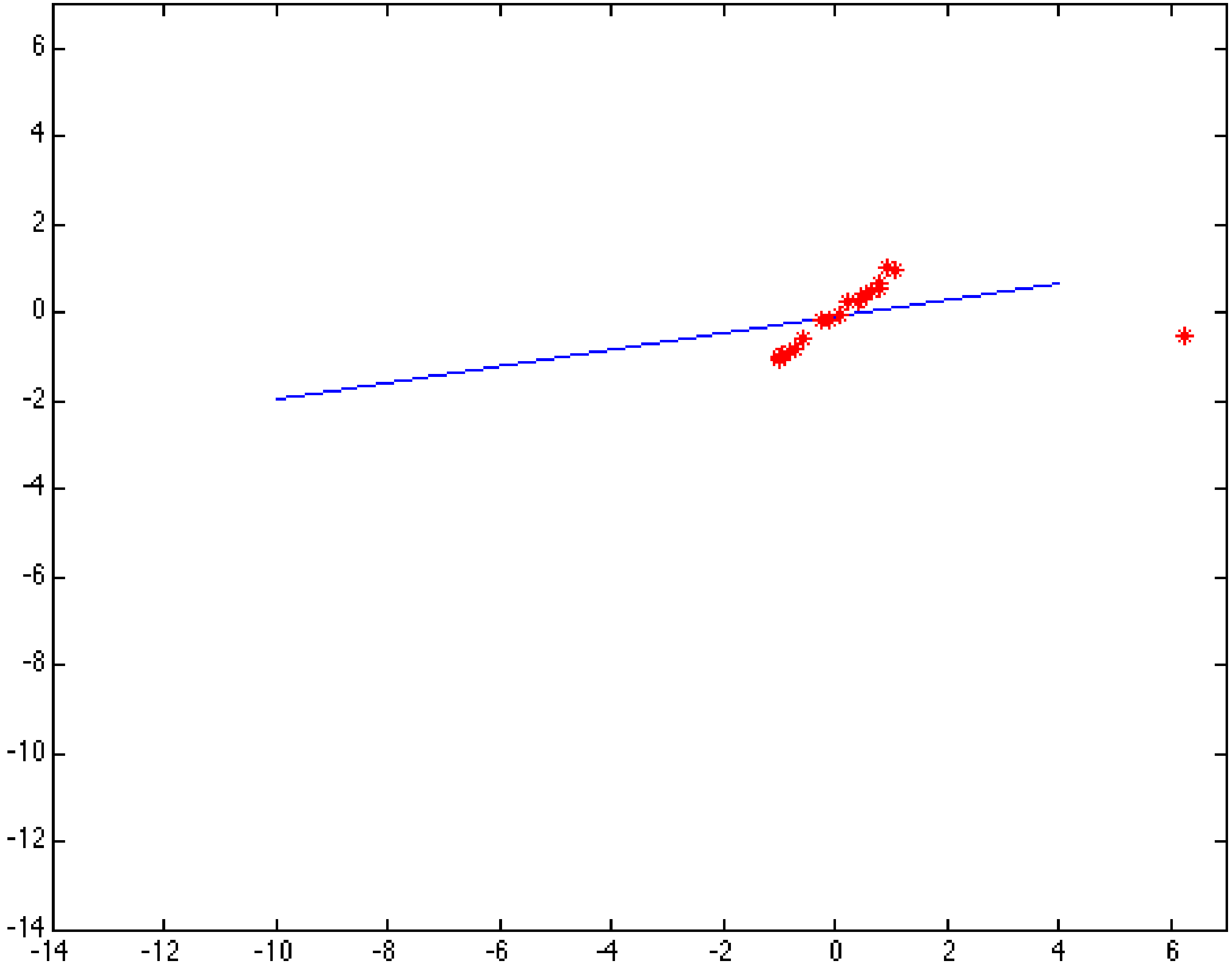
Popular choice:

$$\sigma^{(n)} = 1.4826 \text{ median}_i \; |r_i^{(n)}(x_i; \theta^{(n-1)})|$$

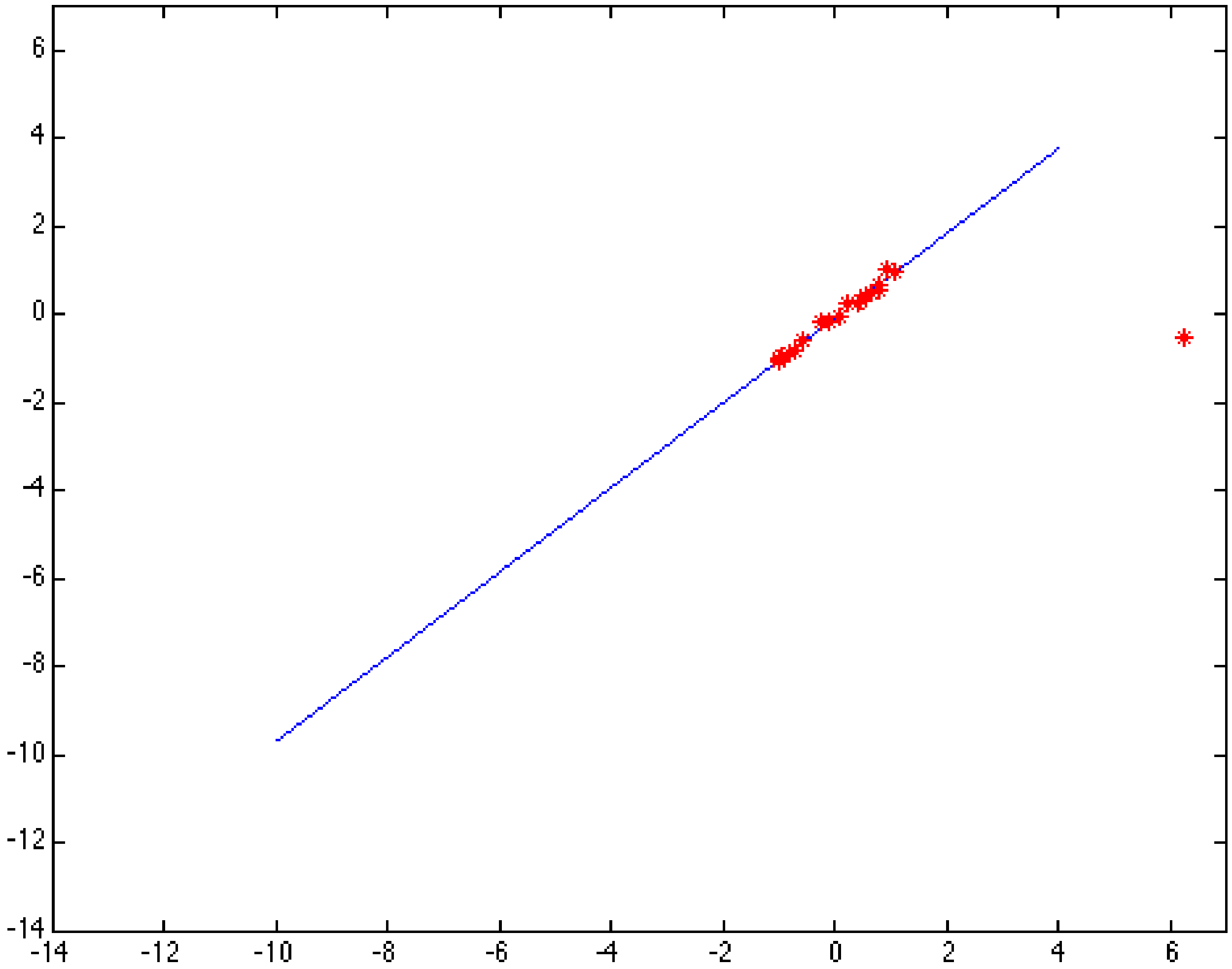
Too small



Too large



Just right



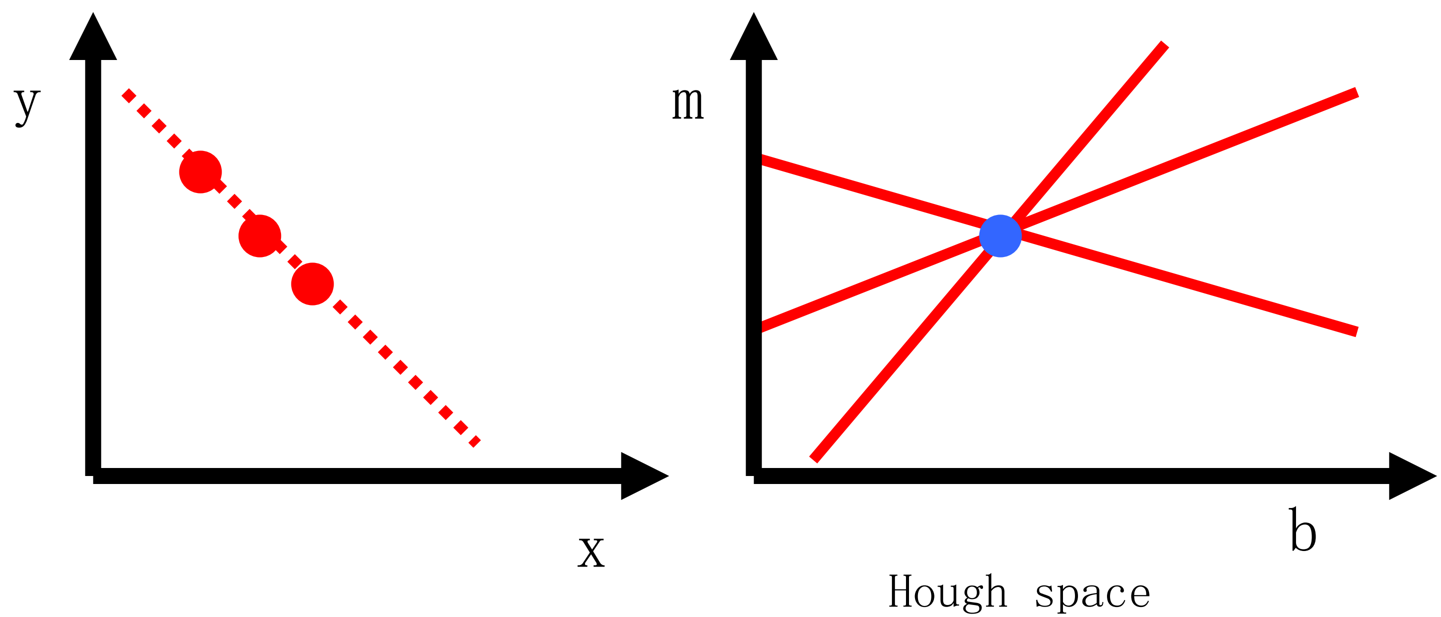
Robust estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Least squares solution can be used for initialization
- Adaptive choice of scale: approx. 1.5 times median residual (F&P, Sec. 15.5.1)

Hough transform

P.V.C. Hough, *Machine Analysis of Bubble Chamber Pictures,* Proc. Int. Conf. High Energy Accelerators and Instrumentation, 1959

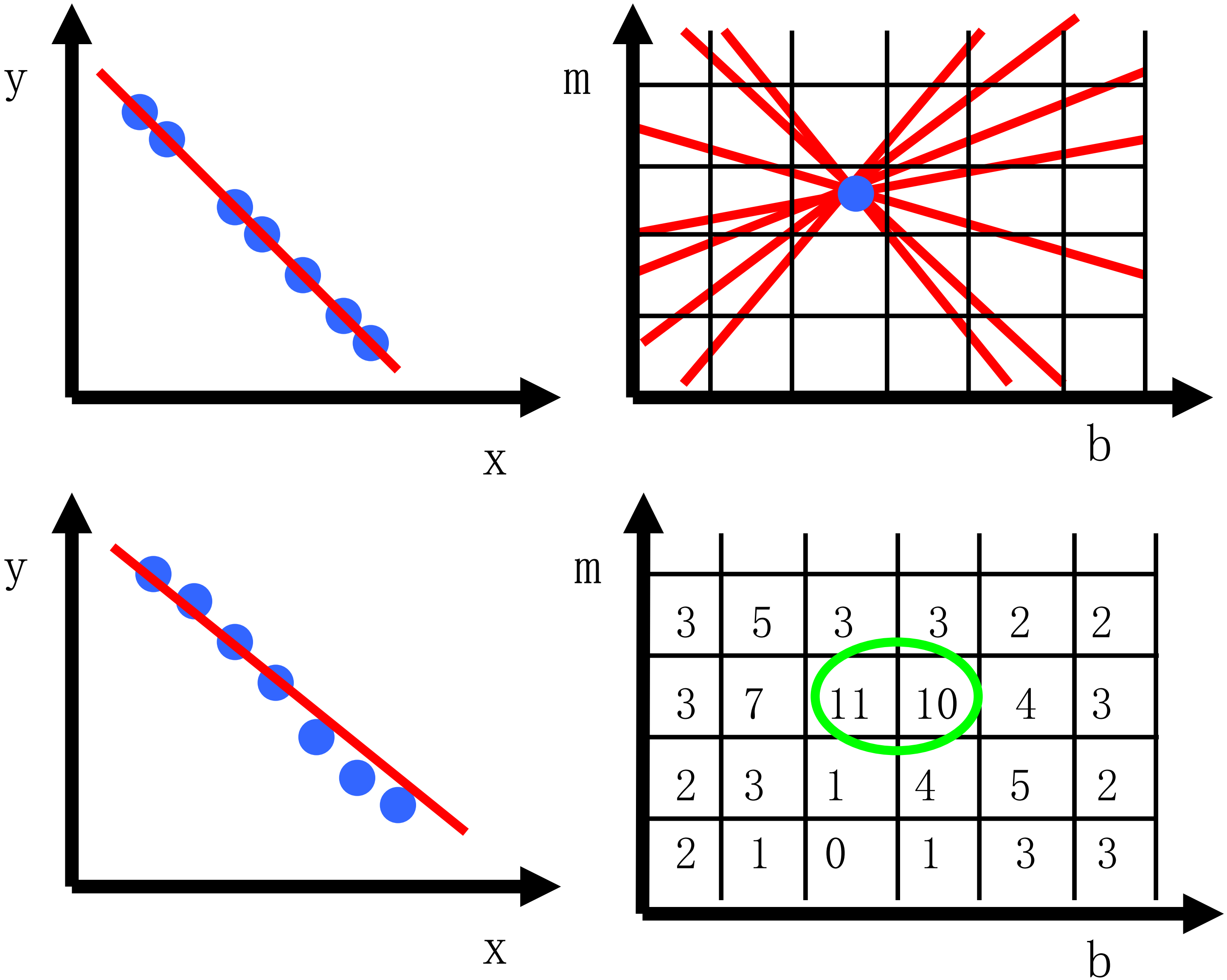
Given a set of points, find the curve or line that explains the data points best



$$y = m\ x + b$$

Slide from S. Savarese

Hough transform

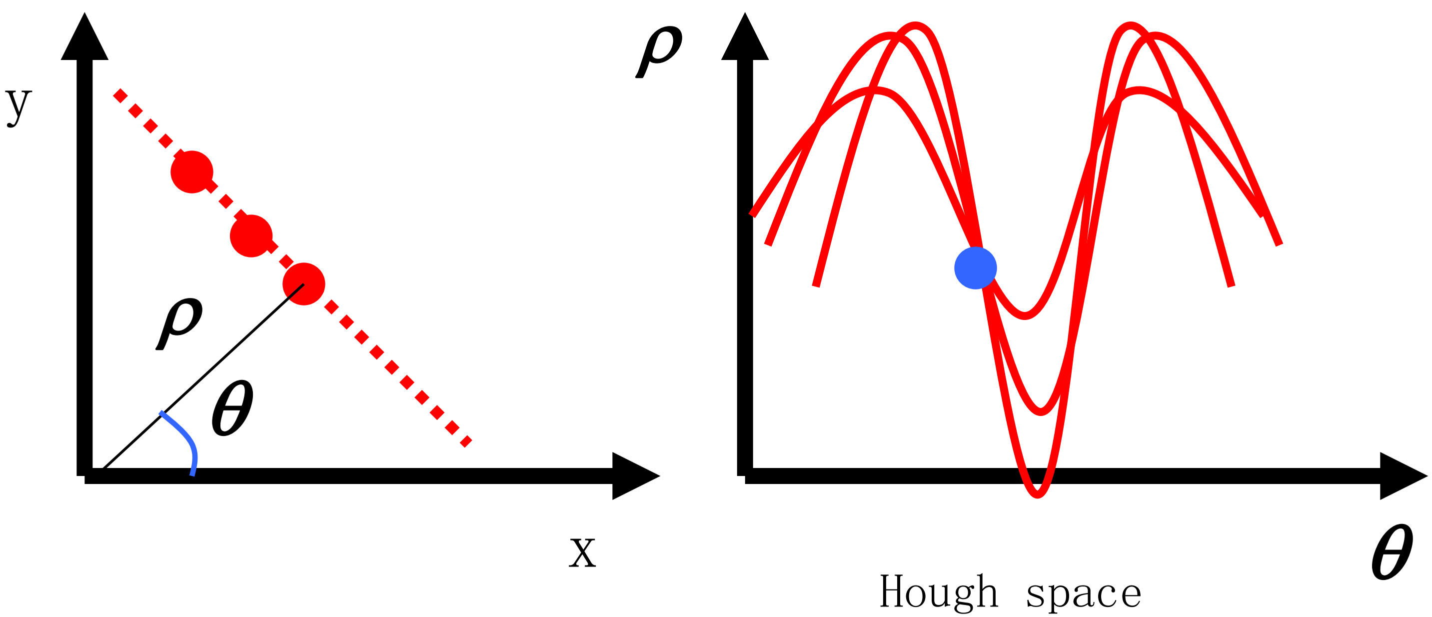


Slide from S. Savarese

Hough transform

Issue : parameter space [m,b] is unbounded...

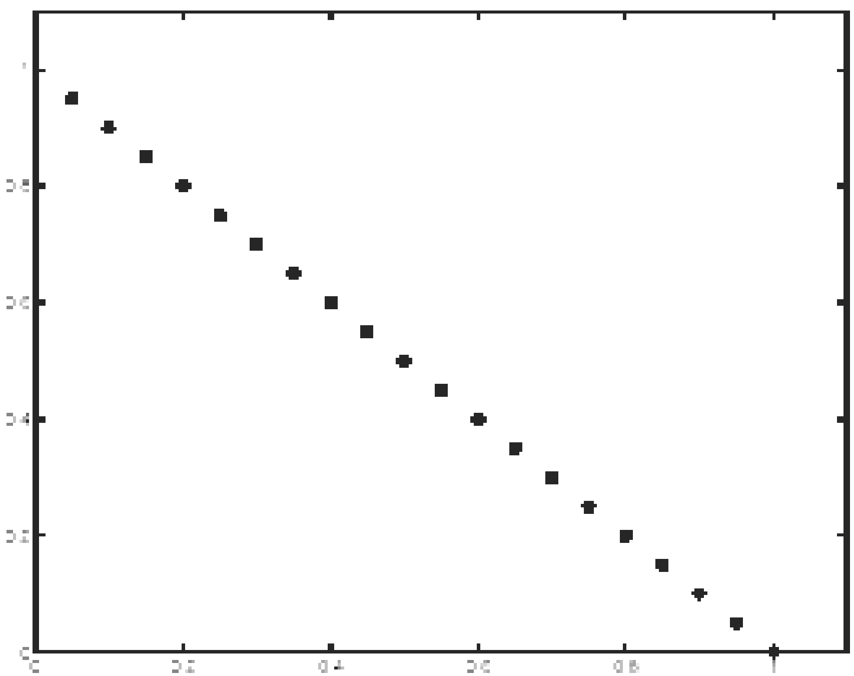
Use a polar representation for the parameter space



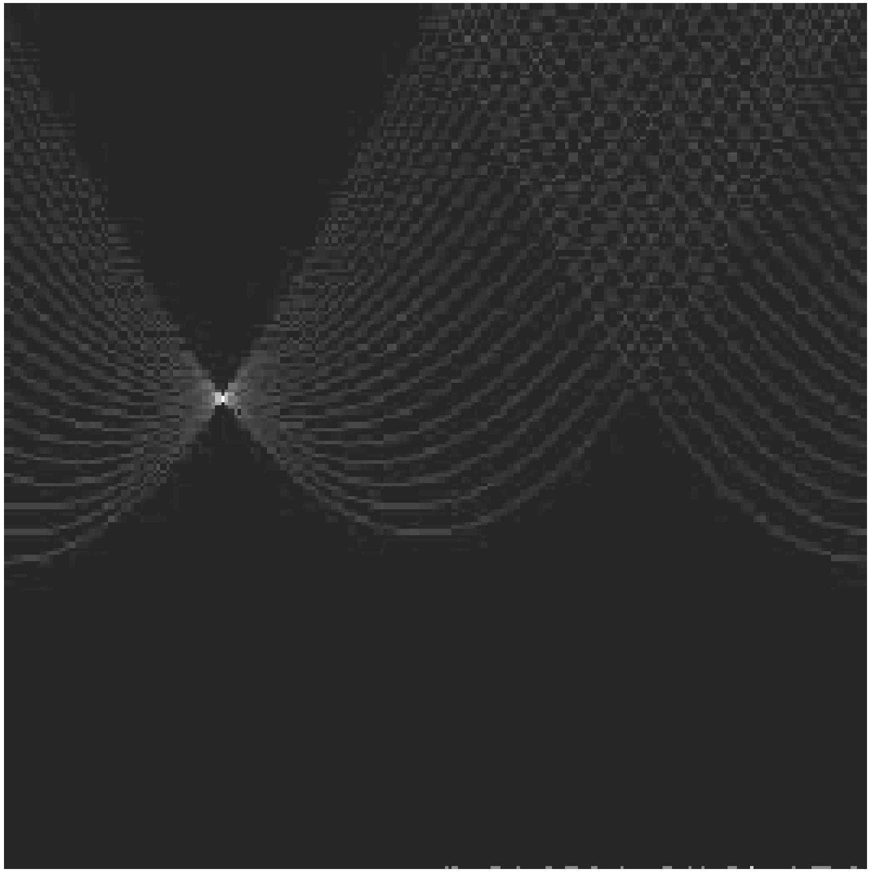
$$x\cos\theta + y\sin\theta = \rho$$

Slide from S. Savarese

Hough transform - experiments



features

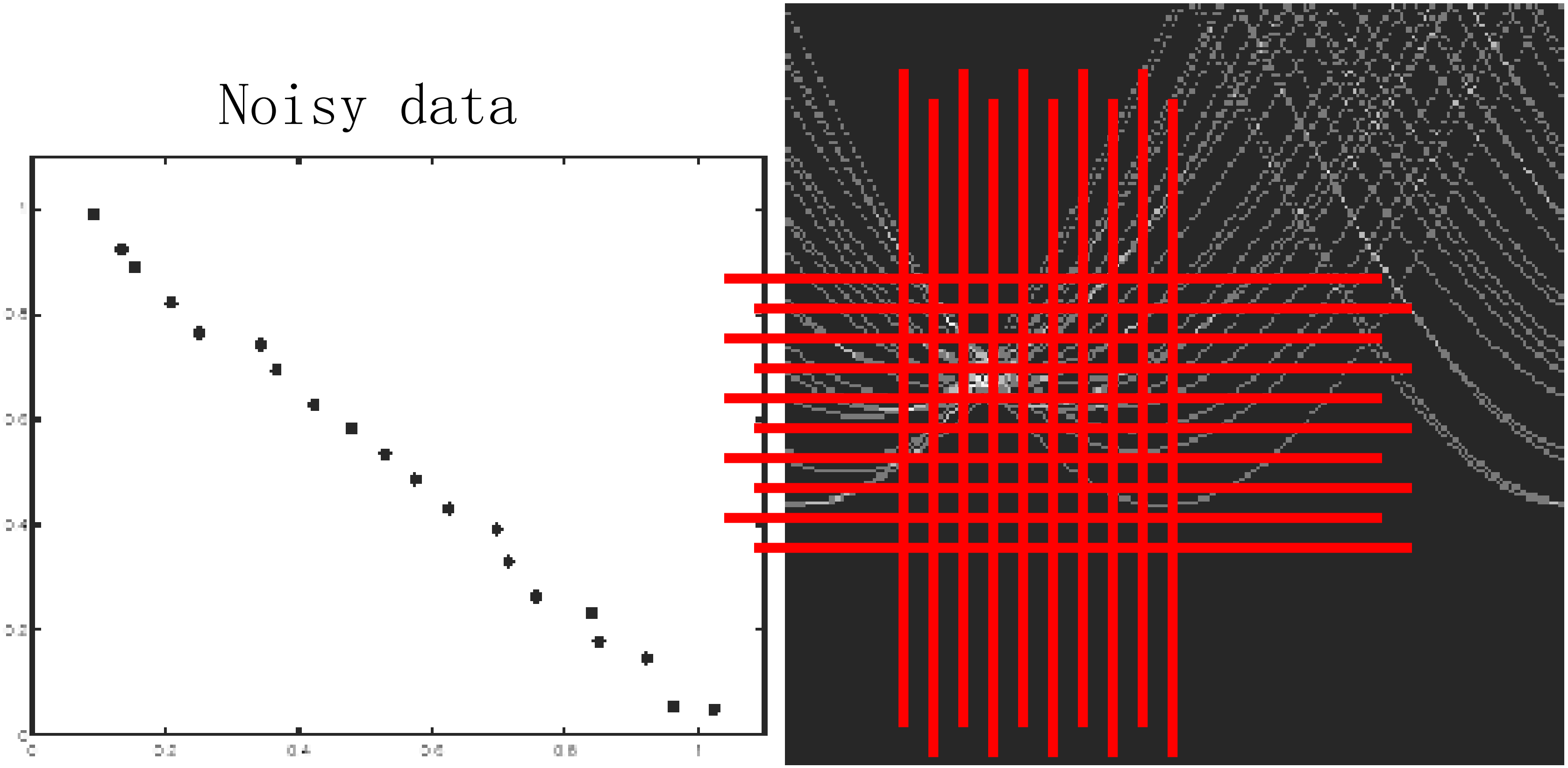


votes

Slide from S. Savarese

Hough transform – experiments

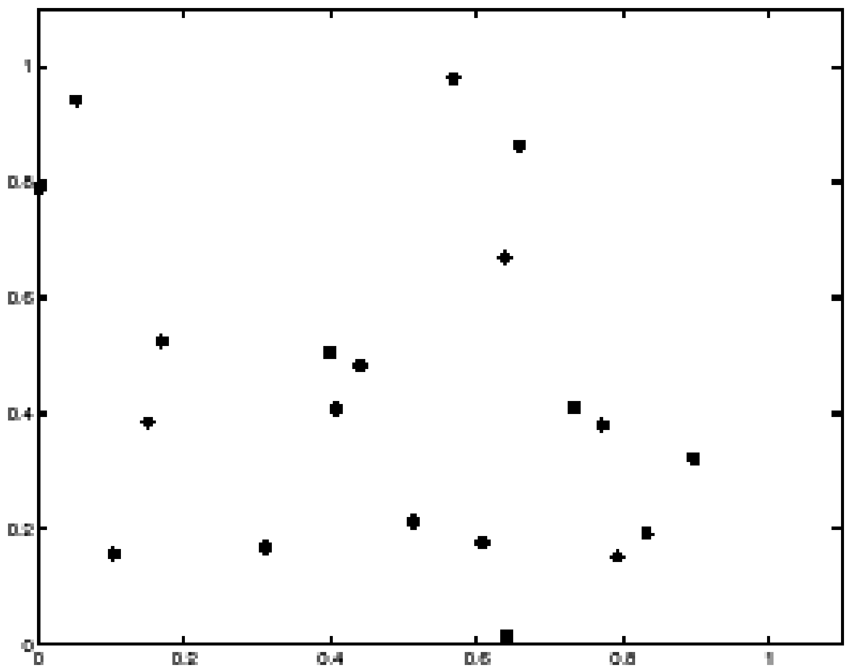
Noisy data



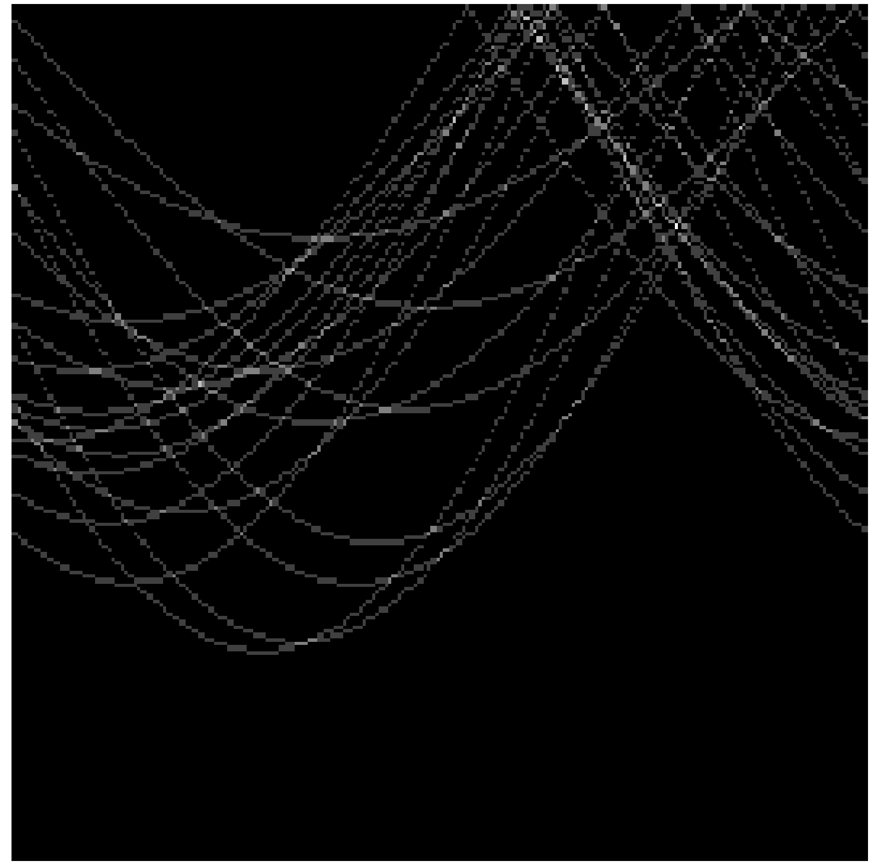
features votes

Issue: Grid size needs to be adjusted…

Hough transform – experiments



features



votes

Issue: spurious peaks due to uniform noise

Hough transform

- **Fitting a circle (x, y, r)**

Hough transform conclusions

Good

- Robust to outliers: each point votes separately
- Fairly efficient (much faster than trying all sets of parameters)
- Provides multiple good fits

Bad

- Some sensitivity to noise
- Bin size trades off between noise tolerance, precision, and speed/memory
  - Can be hard to find sweet spot
- Not suitable for more than a few parameters
  - grid size grows exponentially

Common applications

- Line fitting (also circles, ellipses, etc.)
- Object instance recognition (parameters are affine transform)
- Object category recognition (parameters are position/scale)

How do we find the best line?

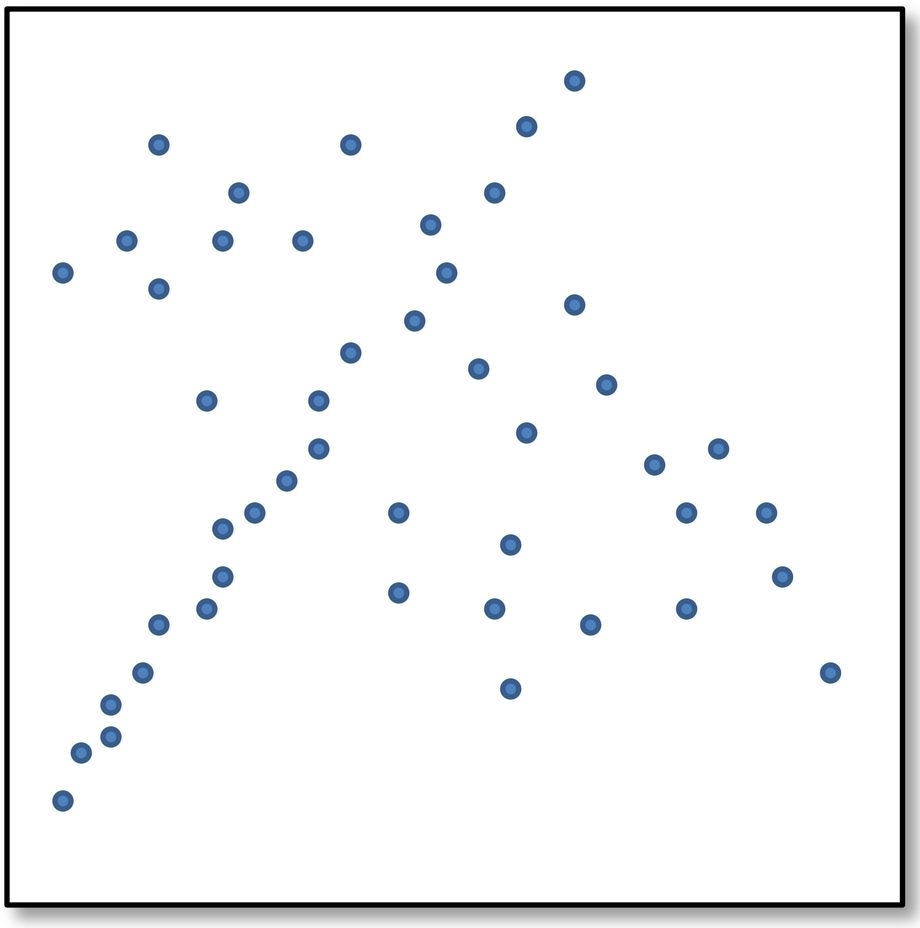
- Unlike least-squares, no simple closed-form solution
- Hypothesize-and-test
  - Try out many lines, keep the best one
  - Which lines?

Another Idea

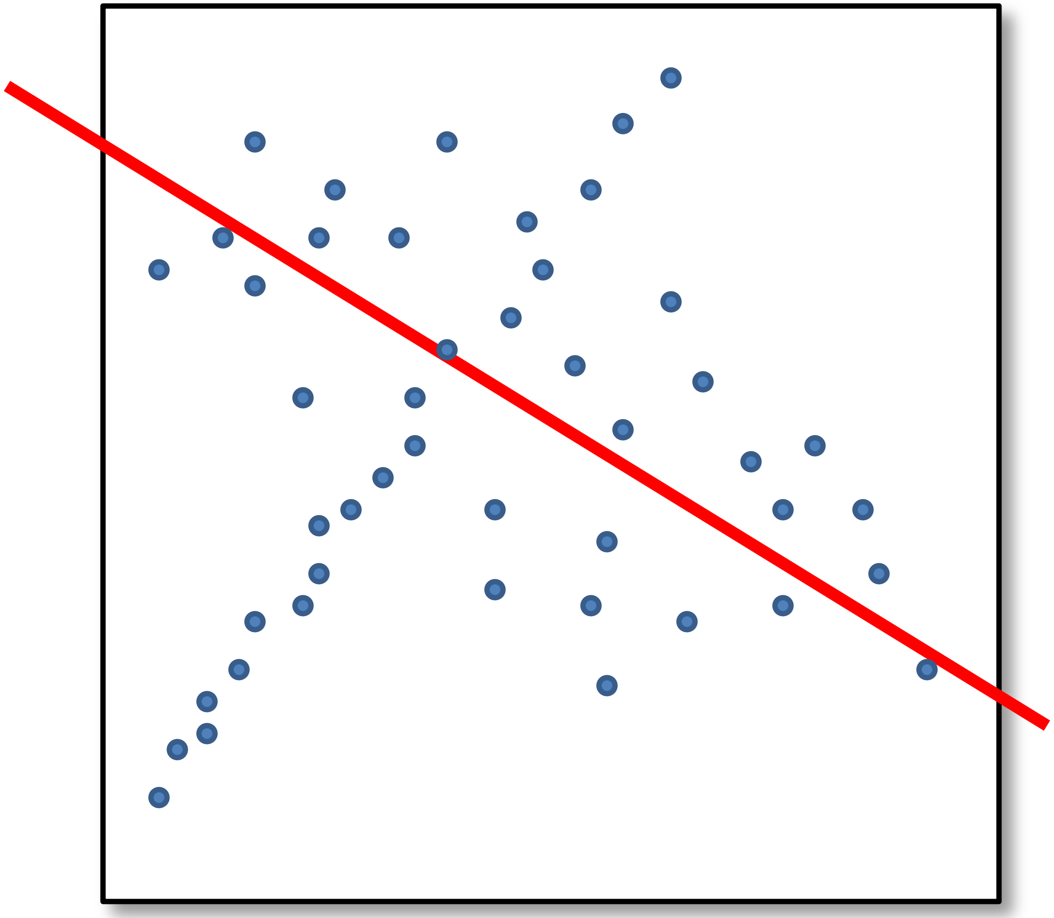
- Given a hypothesized line
- Count the number of points that "agree" with the line
  - "Agree" = within a small distance of the line
  - I.e., the inliers to that line

- For all possible lines, select the one with the largest number of inliers

Counting inliers

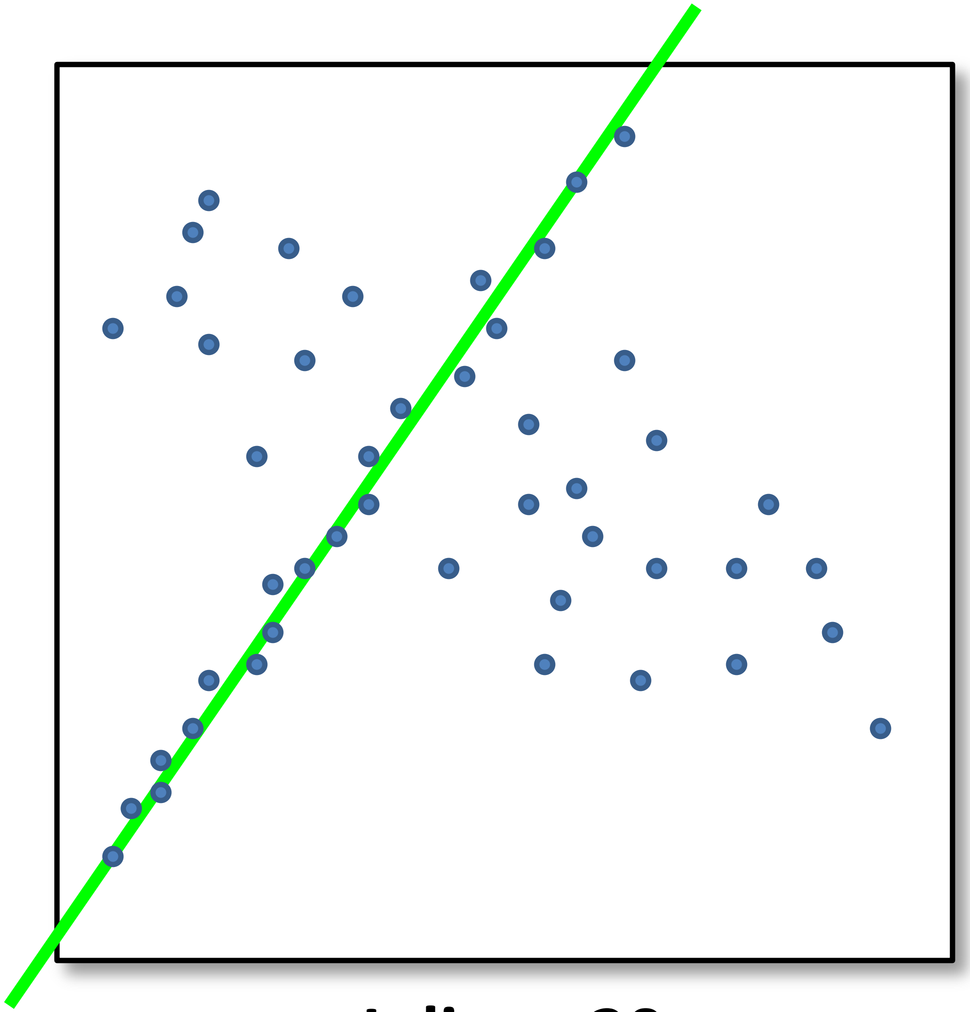


Counting inliers



Inliers: 3

Counting inliers



Inliers: 20

Hypothesize and test

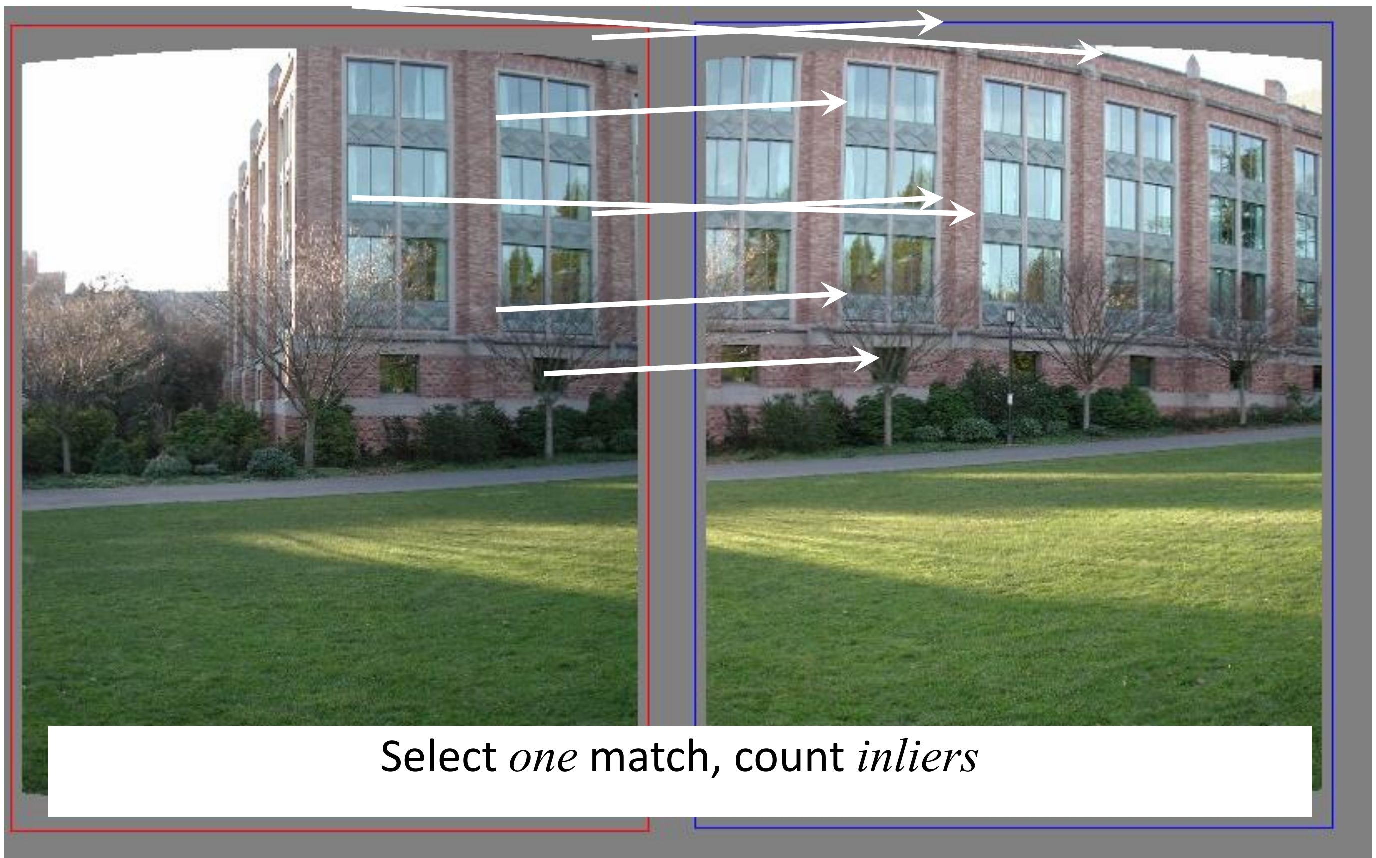
1. **Propose parameters**
   - Try all possible
   - Each point votes for all consistent parameters
   - Repeatedly sample enough points to solve for parameters

2. **Score the given parameters**
   - Number of consistent points, possibly weighted by distance

3. **Choose from among the set of parameters**
   - Global or local maximum of scores

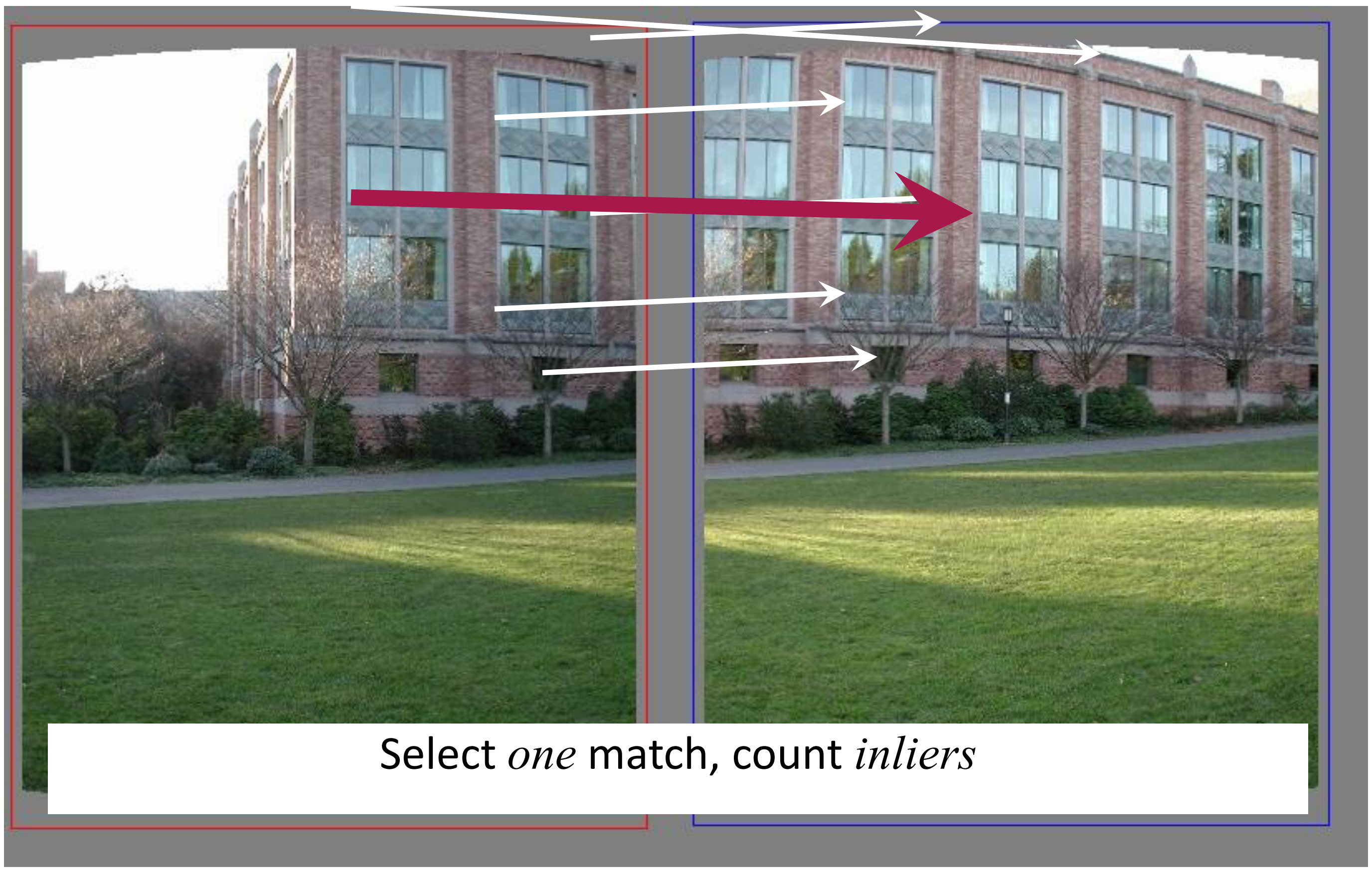
4. **Possibly refine parameters using inliers**

RAndom SAmple Consensus
RANSAC



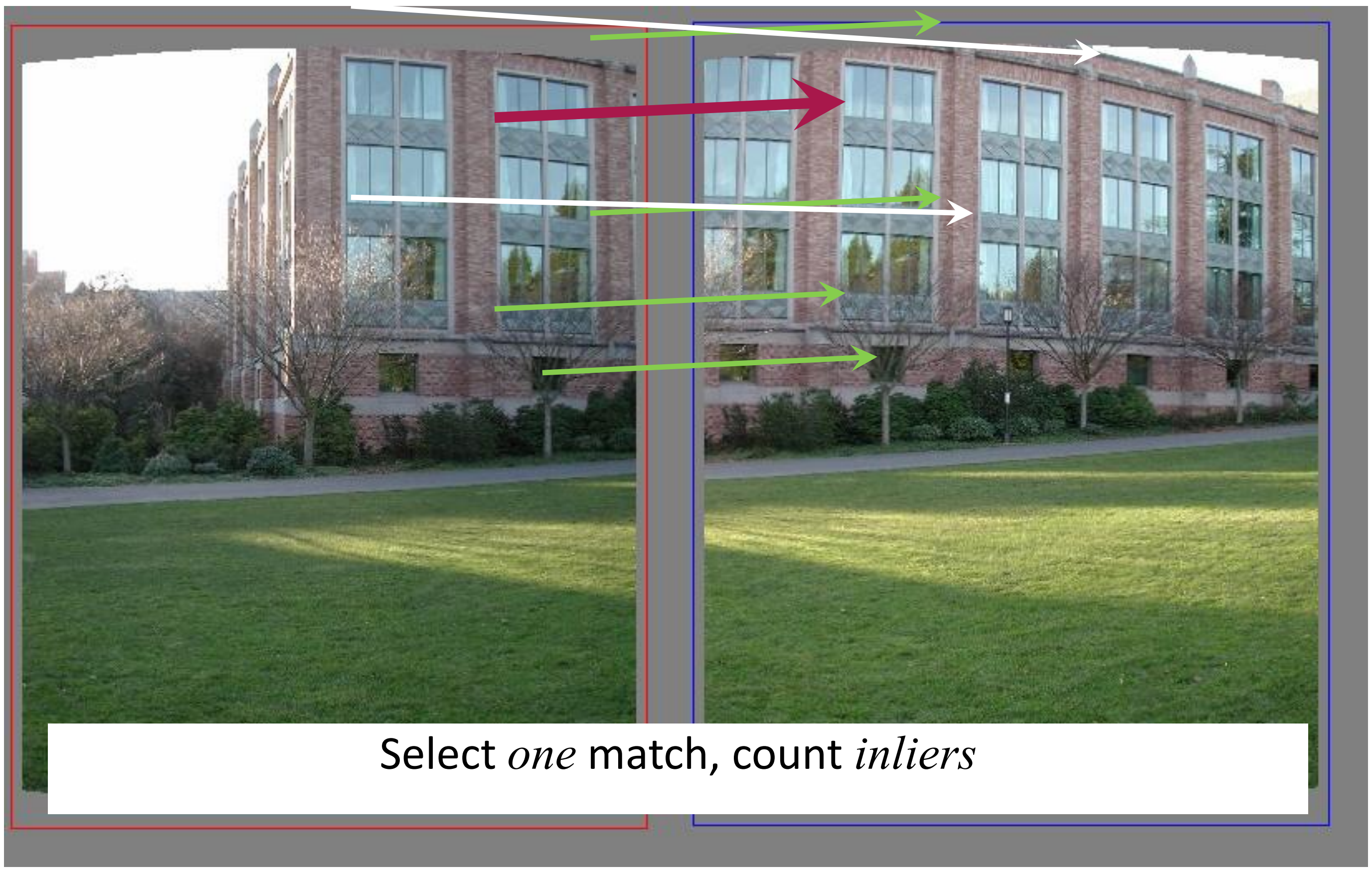
Select *one* match, count *inliers*

RAndom SAmple Consensus



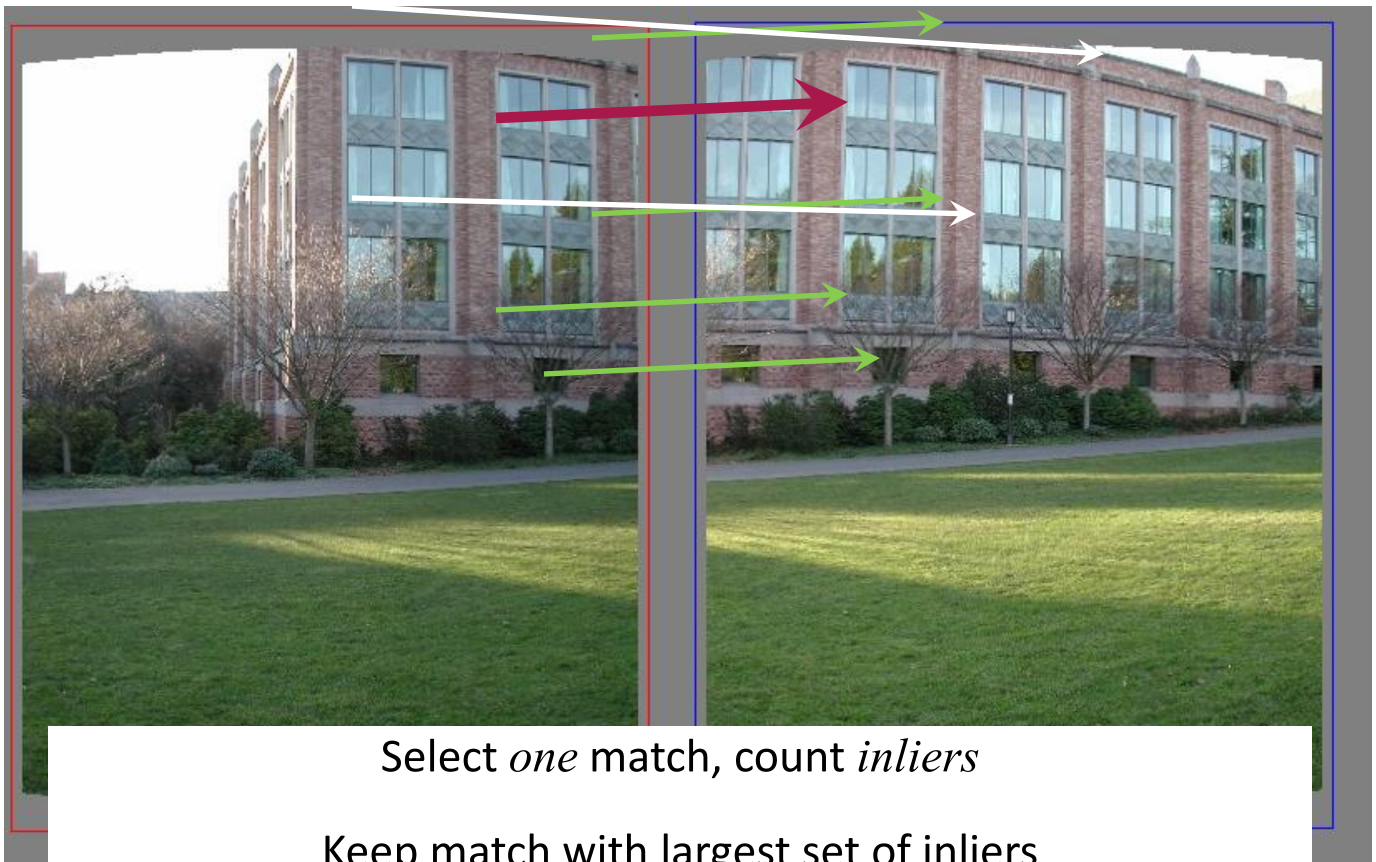
0 inliers

<u>RA</u>ndom <u>SA</u>mple <u>C</u>onsensus



4 inliers

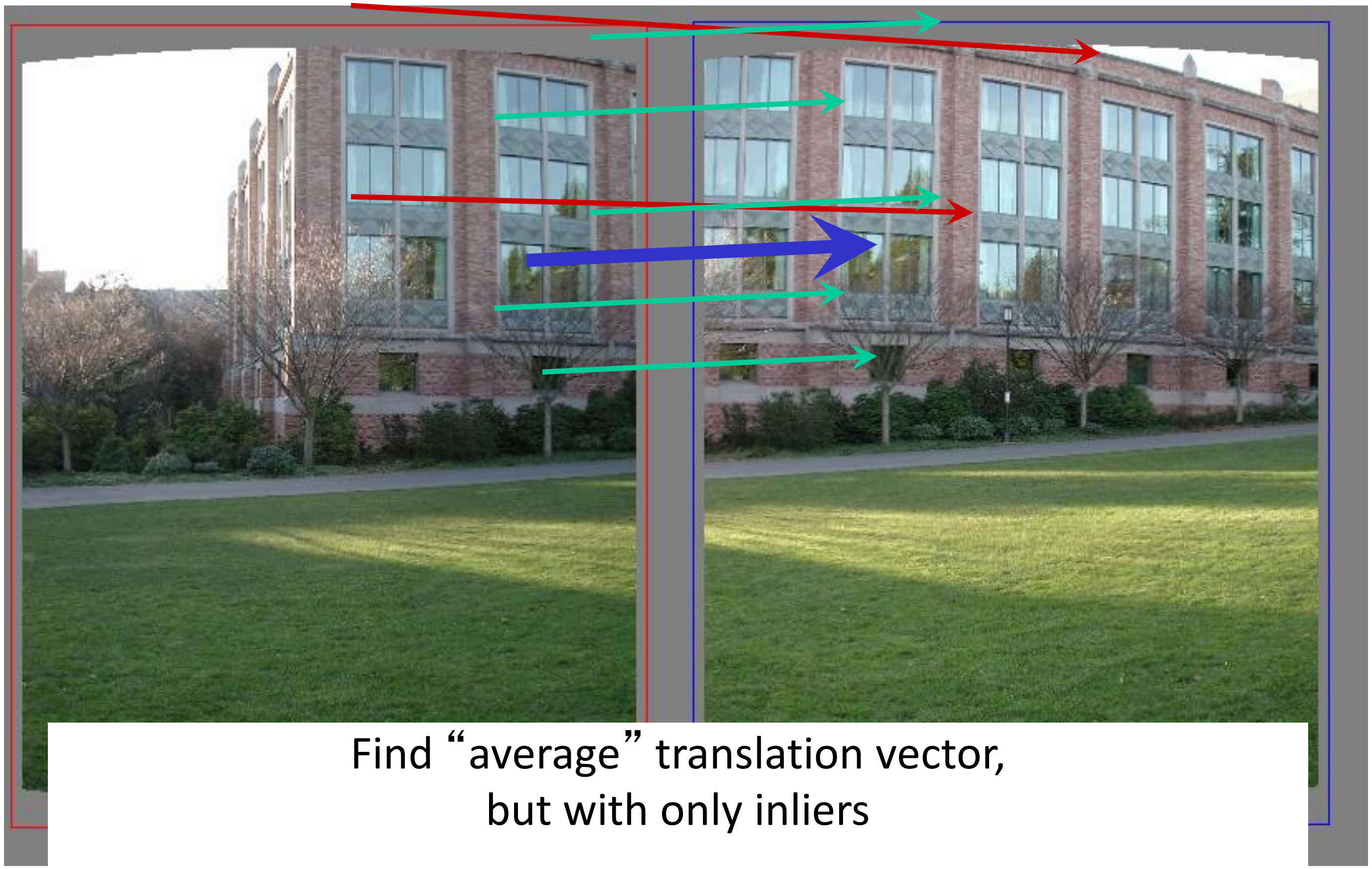
RAndom SAmple Consensus



Select *one* match, count *inliers*

Keep match with largest set of inliers

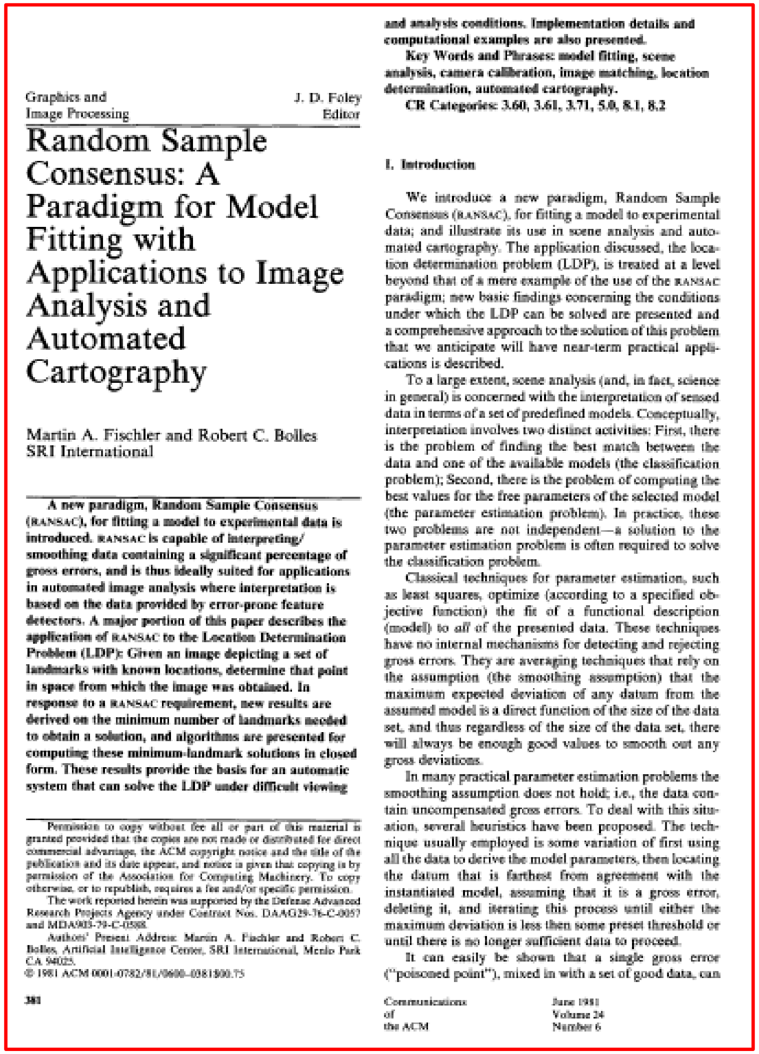
At the end: Least squares fit



Find “average” translation vector,
but with only inliers

Reference

- **M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.**
- http://portal.acm.org/citation.cfm?id=358692



Graphics and Image Processing

J. D. Foley Editor

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

Martin A. Fischler and Robert C. Bolles
SRI International

A new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data is introduced. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and is thus ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors. A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): Given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. In response to a RANSAC requirement, new results are derived on the minimum number of landmarks needed to obtain a solution, and algorithms are presented for computing these minimum-landmark solutions in closed form. These results provide the basis for an automatic system that can solve the LDP under difficult viewing and analysis conditions. Implementation details and computational examples are also presented.

Key Words and Phrases: model fitting, scene analysis, camera calibration, image matching, location determination, automated cartography.

CR Categories: 3.60, 3.61, 3.71, 5.0, 8.1, 8.2

The work reported herein was supported by the Defense Advanced Research Projects Agency under Contract Nos. DAAG29-76-C-0057 and MDA903-79-C-0588.

Authors' Present Address: Martin A. Fischler and Robert C. Bolles, Artificial Intelligence Center, SRI International, Menlo Park CA 94025.

I. Introduction

We introduce a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data; and illustrate its use in scene analysis and automated cartography. The application discussed, the location determination problem (LDP), is treated at a level beyond that of a mere example of the use of the RANSAC paradigm; new basic findings concerning the conditions under which the LDP can be solved are presented and a comprehensive approach to the solution of this problem that we anticipate will have near-term practical applications is described.

To a large extent, scene analysis (and, in fact, science in general) is concerned with the interpretation of sensed data in terms of a set of predefined models. Conceptually, interpretation involves two distinct activities: First, there is the problem of finding the best match between the data and one of the available models (the classification problem); Second, there is the problem of computing the best values for the free parameters of the selected model (the parameter estimation problem). In practice, these two problems are not independent—a solution to the parameter estimation problem is often required to solve the classification problem.

Classical techniques for parameter estimation, such as least squares, optimize (according to a specified objective function) the fit of a functional description (model) to *all* of the presented data. These techniques have no internal mechanisms for detecting and rejecting gross errors. They are averaging techniques that rely on the assumption (the smoothing assumption) that the maximum expected deviation of any datum from the assumed model is a direct function of the size of the data set, and thus regardless of the size of the data set, there will always be enough good values to smooth out any gross deviations.

In many practical parameter estimation problems the smoothing assumption does not hold; i.e., the data contain uncompensated gross errors. To deal with this situation, several heuristics have been proposed. The technique usually employed is some variation of first using all the data to derive the model parameters, then locating the datum that is farthest from agreement with the instantiated model, assuming that it is a gross error, deleting it, and iterating this process until either the maximum deviation is less then some preset threshold or until there is no longer sufficient data to proceed.

It can easily be shown that a single gross error ("poisoned point"), mixed in with a set of good data, can

RANSAC

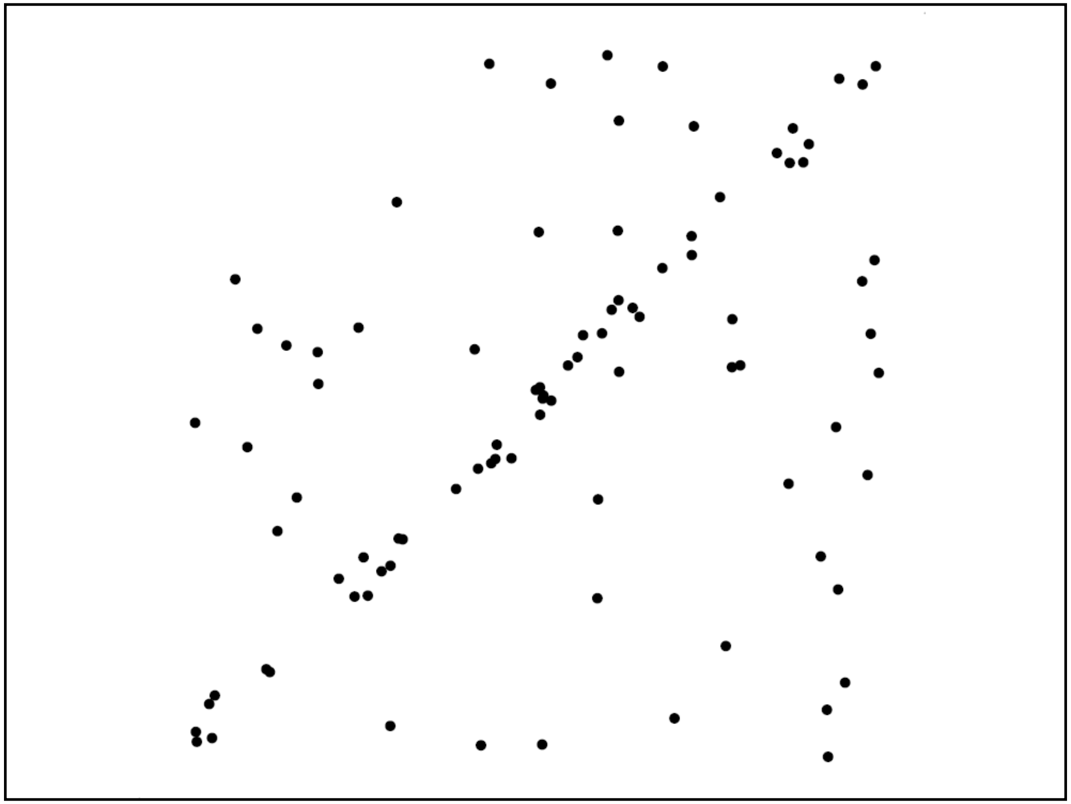
- **Idea:**
  - **All the inliers will agree with each other on the translation vector; the (hopefully small) number of outliers will (hopefully) disagree with each other**
    - **RANSAC only has guarantees if there are < 50% outliers**
  - **"All good matches are alike; every bad match is bad in its own way."**

– Tolstoy via Alyosha Efros

RANSAC

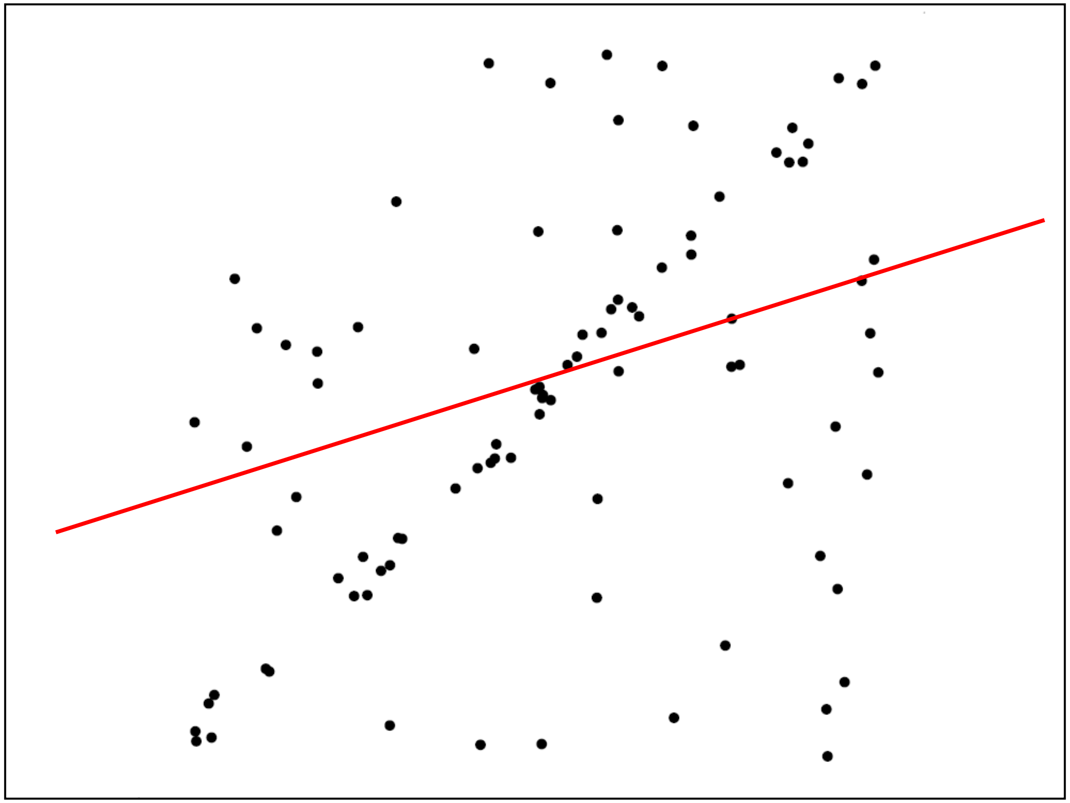
- **Inlier threshold related to the amount of noise we expect in inliers**
  - **Often model noise as Gaussian with some standard deviation (e.g., 3 pixels)**
- **Number of rounds related to the percentage of outliers we expect, and the probability of success we'd like to guarantee**
  - **Suppose there are 20% outliers, and we want to find the correct answer with 99% probability**
  - **How many rounds do we need?**

RANSAC for line fitting example



Source: R. Raguram

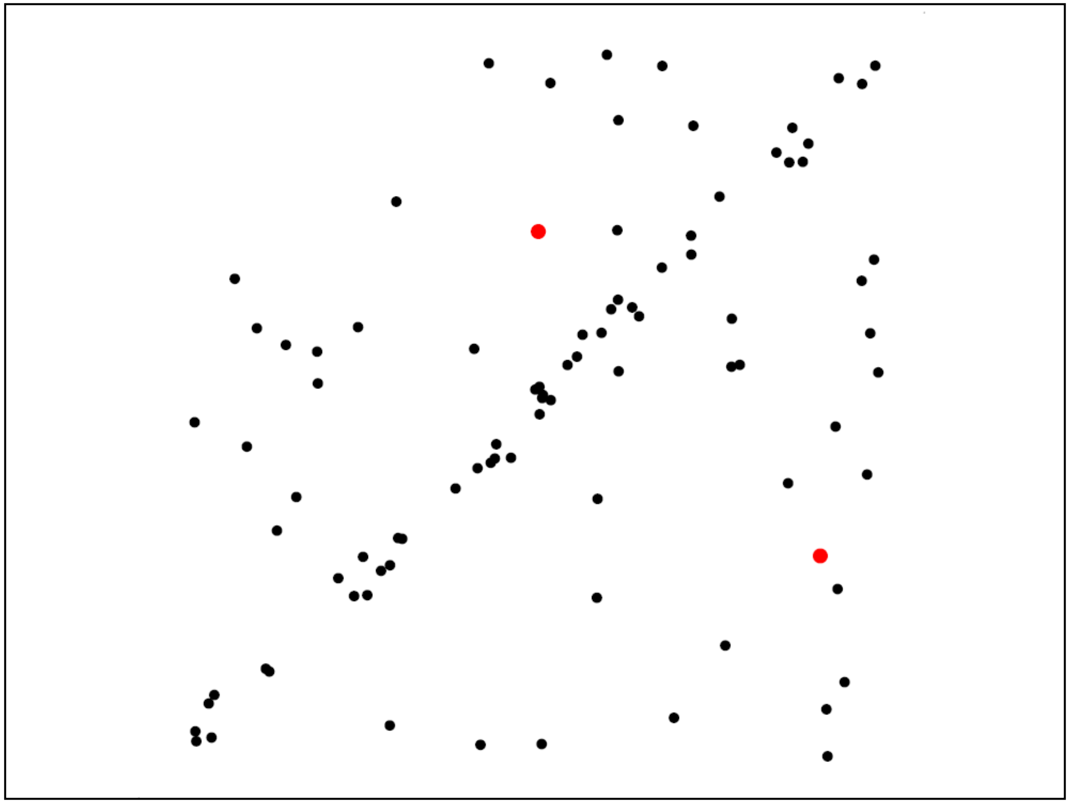
RANSAC for line fitting example



Least-squares fit

Source: R. Raguram

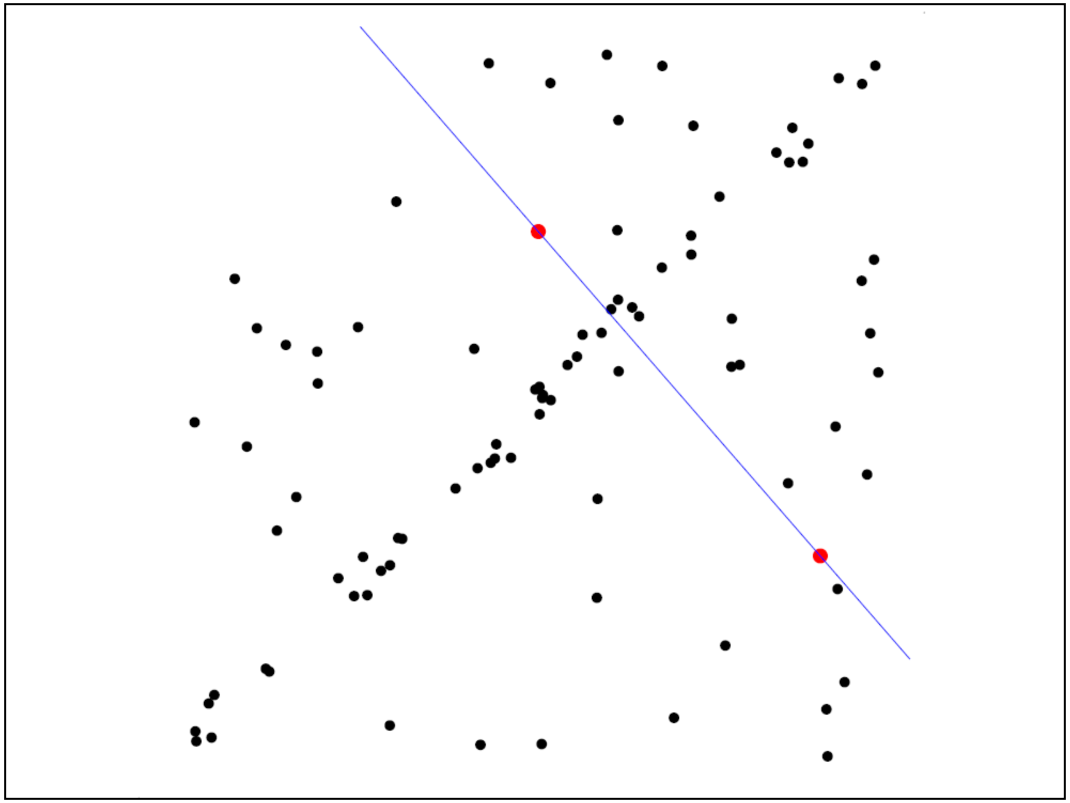
RANSAC for line fitting example



1. Randomly select minimal subset of points

Source: R. Raguram

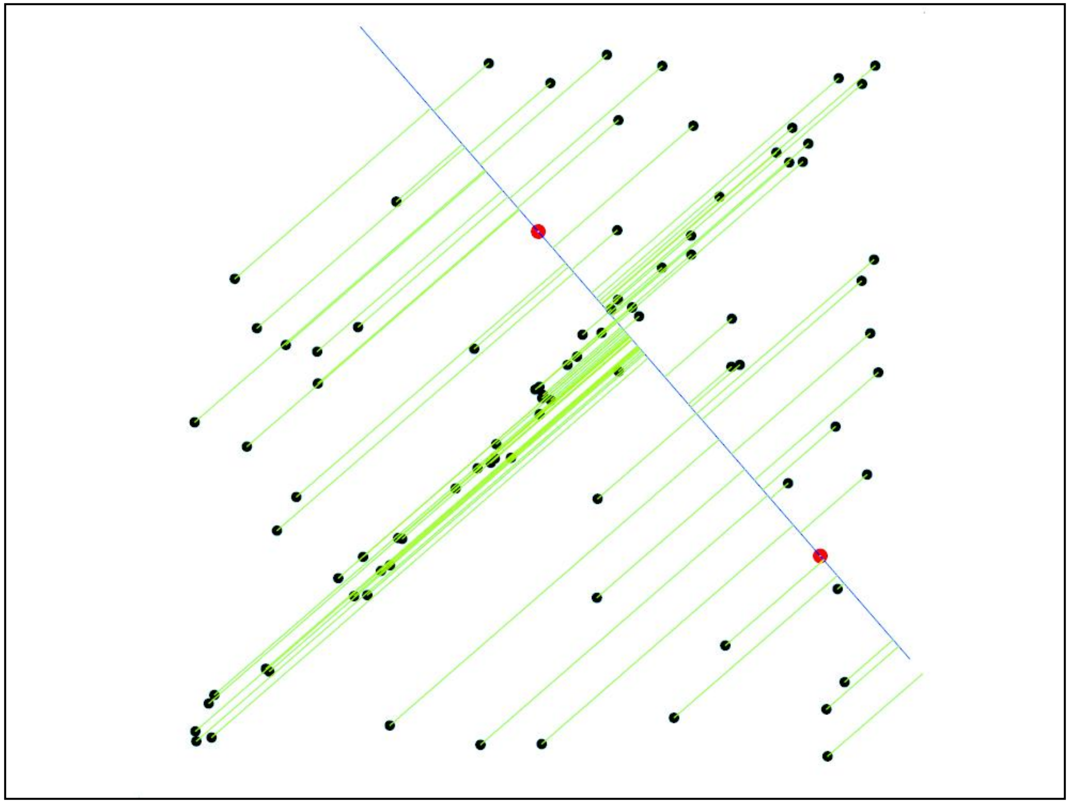
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

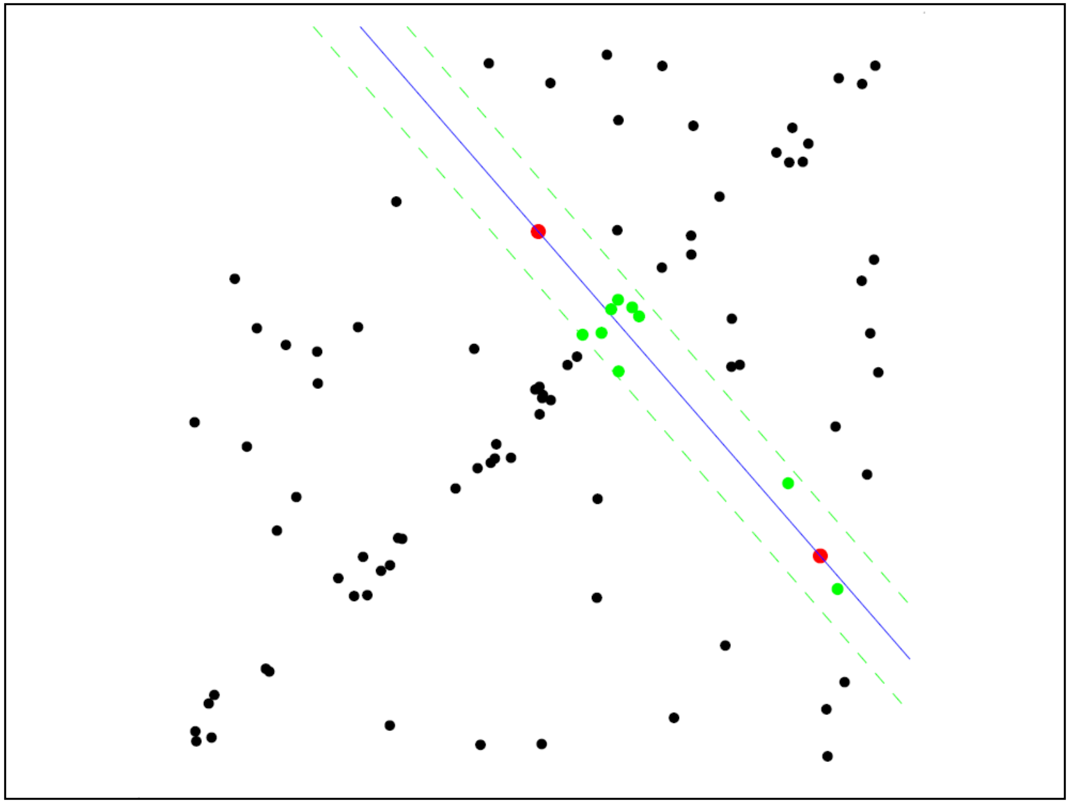
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

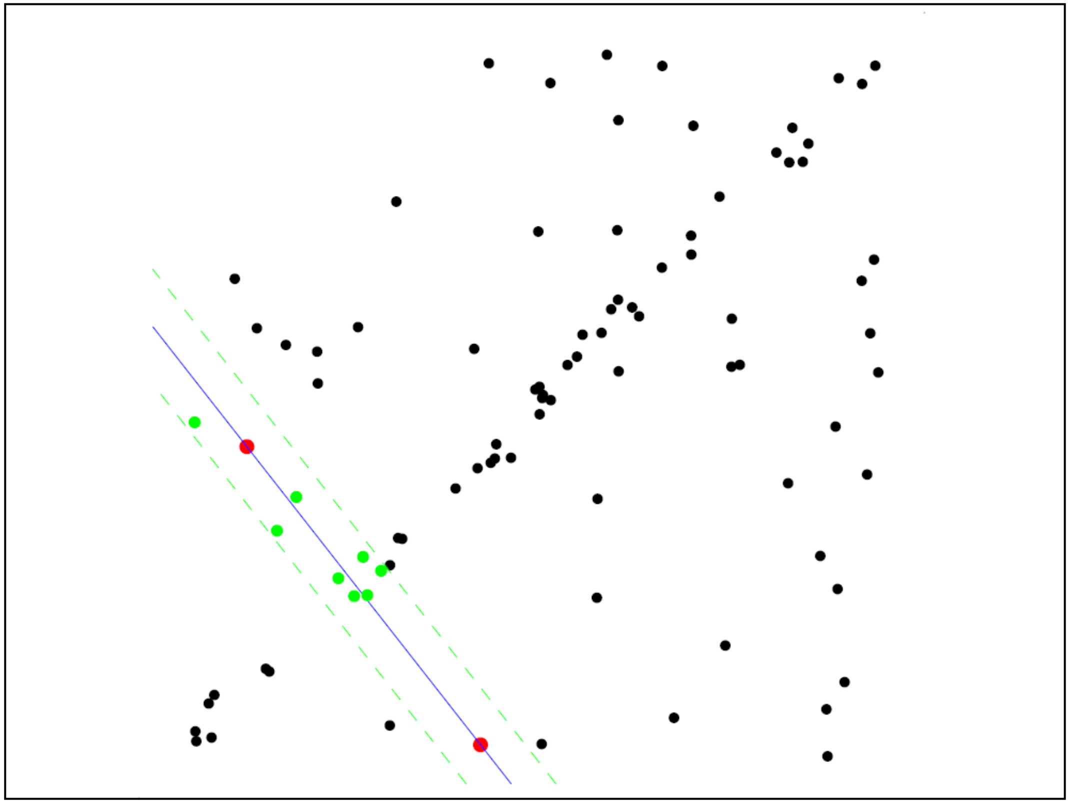
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

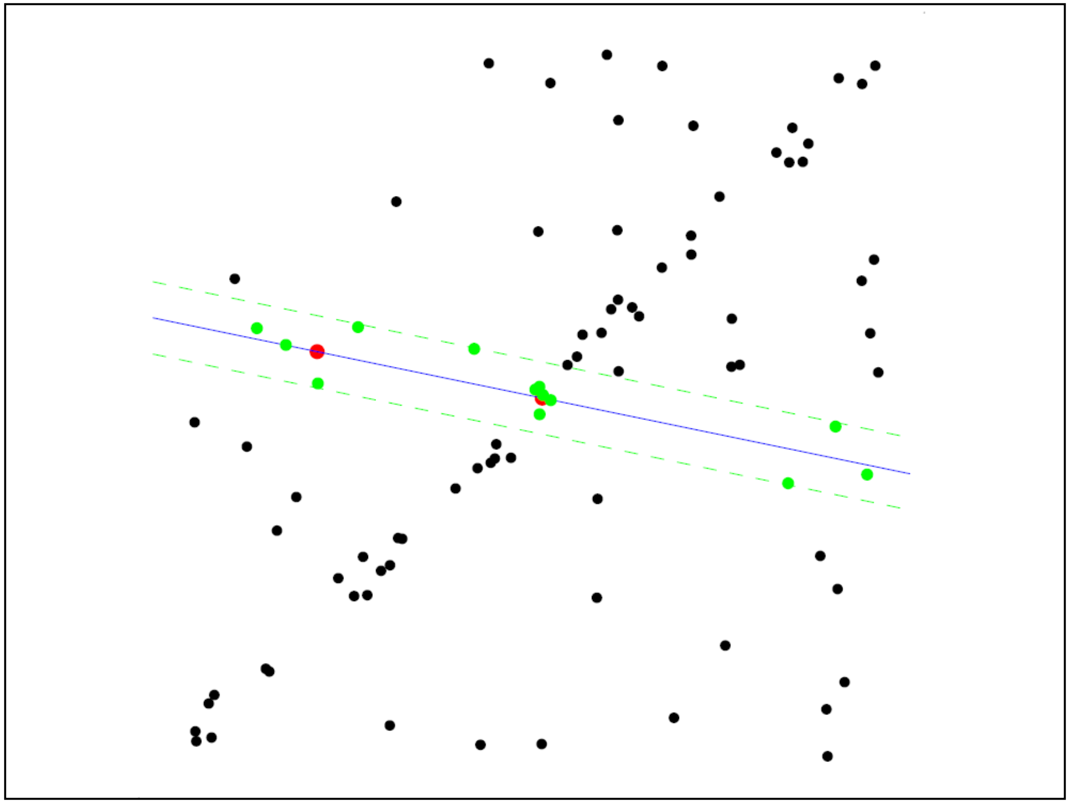
RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

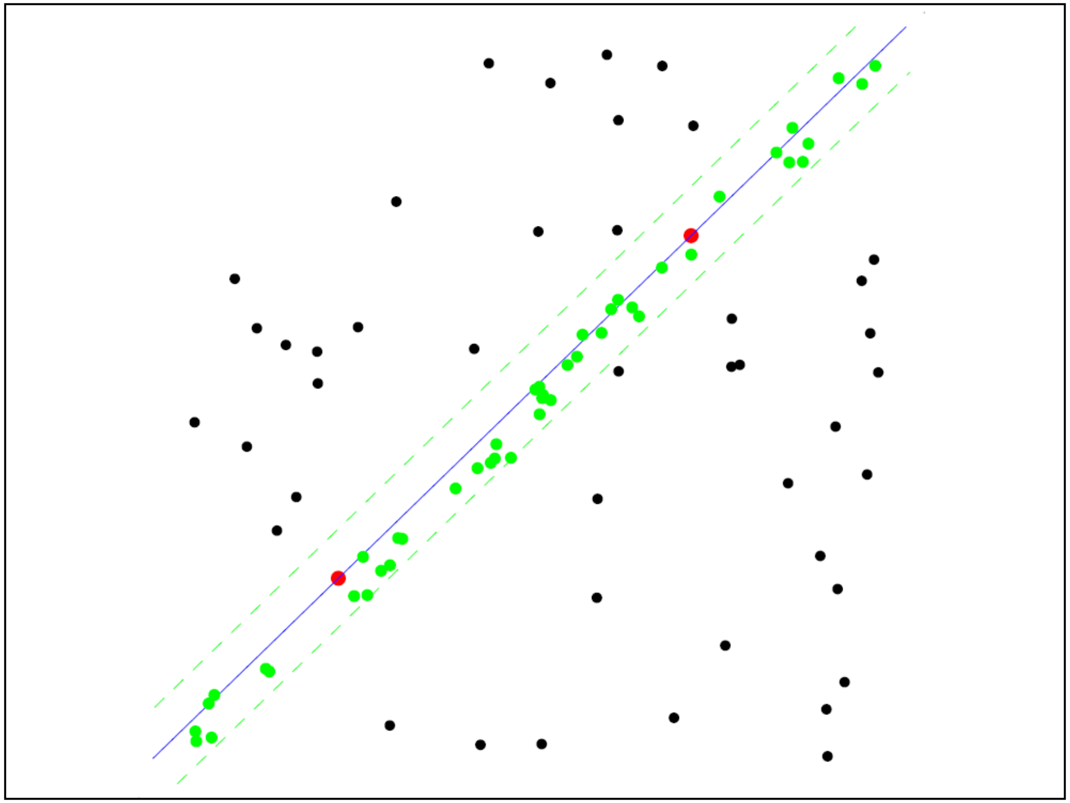


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example

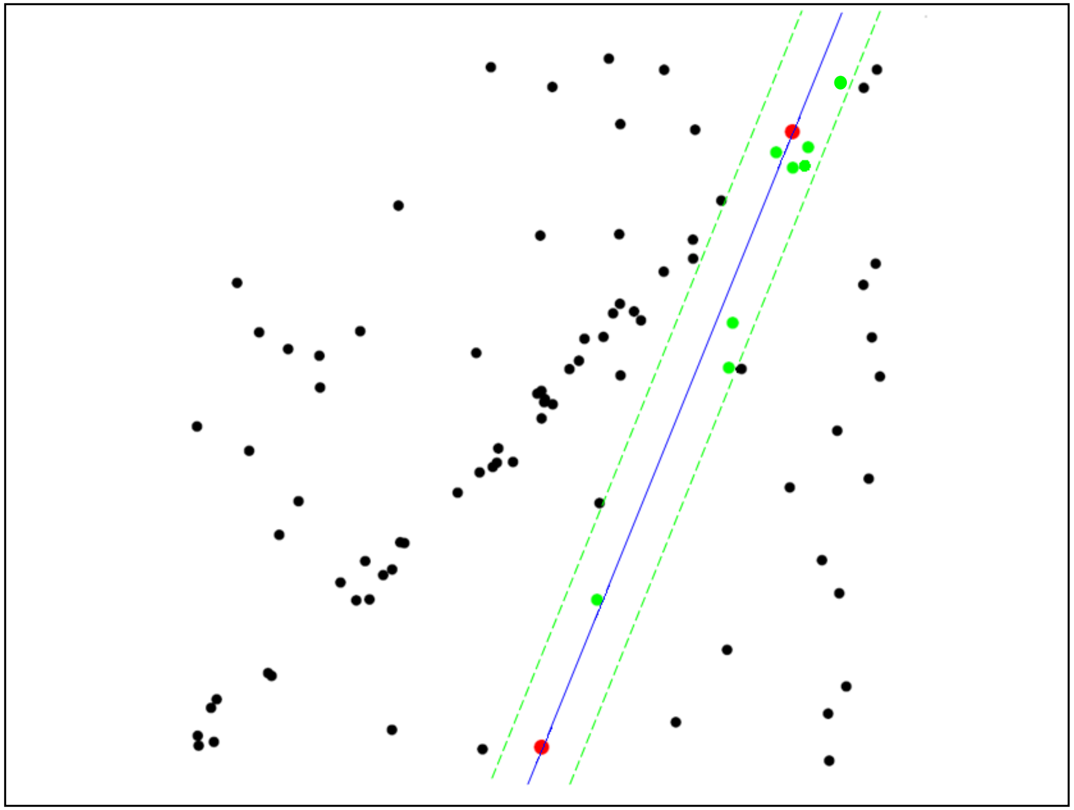
Uncontaminated sample



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat *hypothesize-and-verify* loop

Source: R. Raguram

RANSAC for line fitting

- Repeat ***N*** times:
- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find *inliers* to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Choosing the parameters

- **Initial number of points *s***
  - **Typically minimum number needed to fit the model**
- **Distance threshold *t***
  - **Choose *t* so probability for inlier is *p* (e.g. 0.95)**
  - **Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$**
- **Number of samples *N***
  - **Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)**

Source: M. Pollefeys

Choosing the parameters

- Initial number of points ***s***
  - Typically minimum number needed to fit the model
- Distance threshold ***t***
  - Choose ***t*** so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples ***N***
  - Choose ***N*** so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\boxed{\left(1-\left(1-e\right)^s\right)^N = 1-p}$$

$$N = \log\left(1-p\right)/\log\left(1-\left(1-e\right)^s\right)$$

	proportion of outliers e						
s	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

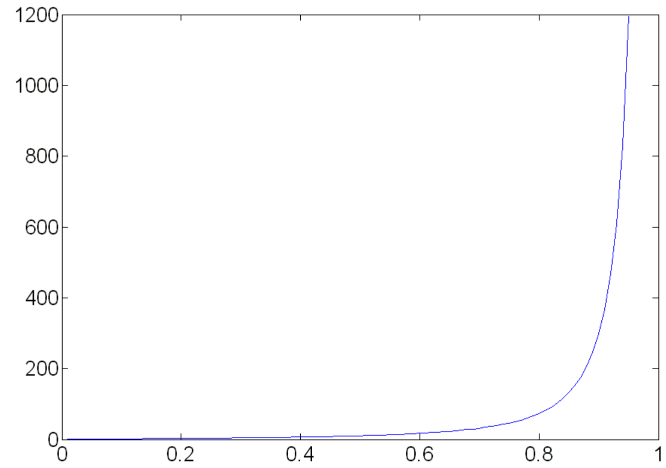
Source: M. Pollefeys

Choosing the parameters

- Initial number of points ***s***
  - Typically minimum number needed to fit the model
- Distance threshold ***t***
  - Choose ***t*** so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples ***N***
  - Choose ***N*** so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\boxed{\left(1-\left(1-e\right)^s\right)^N = 1-p}$$

$$N = \log\left(1-p\right)/\log\left(1-\left(1-e\right)^s\right)$$



Source: M. Pollefeys

Choosing the parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Consensus set size *d*
  - Should match expected inlier ratio

Source: M. Pollefeys

Adaptively determining the number of samples

- **Outlier ratio *e* is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield *e*=0.2**
- **Adaptive procedure:**
  - ***N*=∞, *sample_count* =0**
  - **While *N* >*sample_count***
    - **Choose a sample and count the number of inliers**
    - **If inlier ratio is highest of any found so far, set e = 1 – (number of inliers)/(total number of points)**
    - **Recompute *N* from *e:***

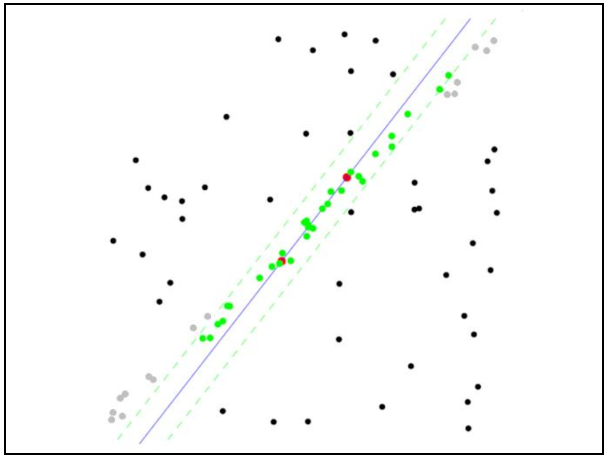
$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

    - **Increment the *sample_count* by 1**

Source: M. Pollefeys

RANSAC pros and cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice
- Cons
  - Lots of parameters to tune
  - Doesn't work well for low inlier ratios (too many iterations, or can fail completely)
  - Can't always get a good initialization of the model based on the minimum number of samples



Readings

- **Reading chapter 10.1-4**