KECE471 Computer Vision

Segmentation by Fitting a Model

Chang-Su Kim

Chapter 15, Computer Vision by Forsyth and Ponce

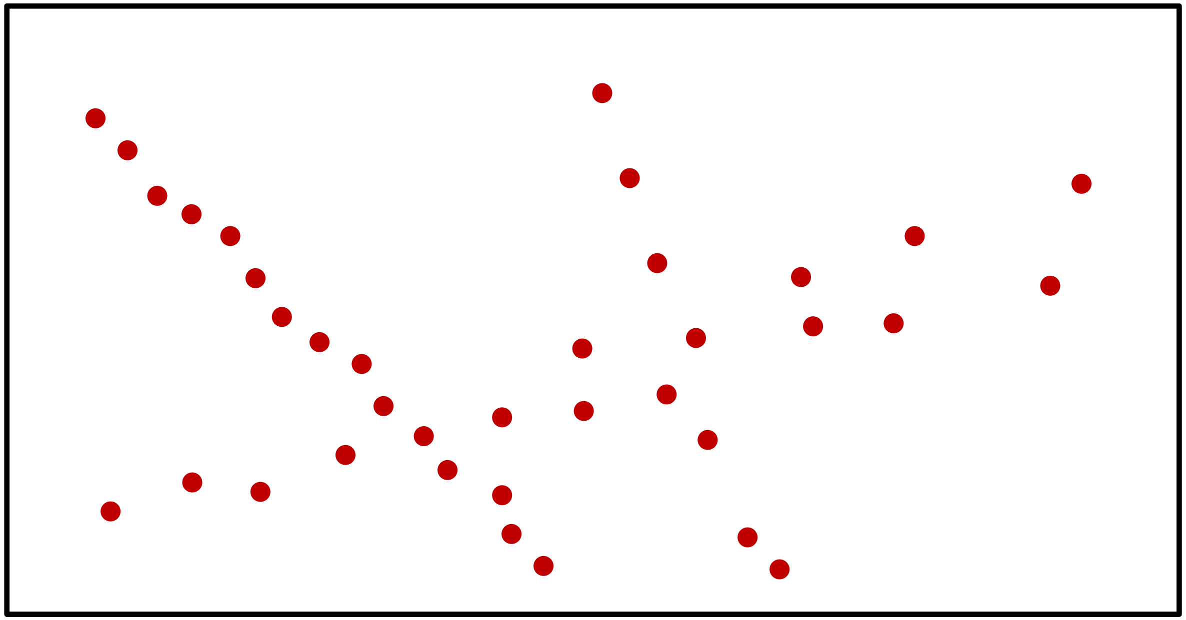
Note: Dr. Forsyth's notes are partly used.
Jun-Sung Kim in Korea University made the first draft of these slides

Fitting

- Choose an object (or model) to represent a set of tokens
  - Objects : line, circle, ellipse, and etc
  - e.g. Find a line that best describes a set of points

- Three main questions
  - What object represents this set of tokens best?
  - Which objects are associated with which tokens?
  - How many objects are there?

Line Fitting



- Three main questions
  - What line represents this set of points best?
  - Which lines gets which tokens?
  - How many lines are there?

Hough Transform for Line Fitting

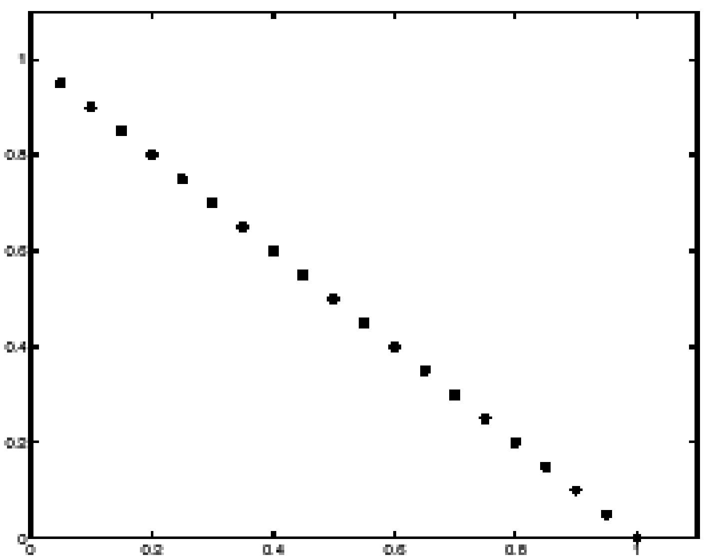
Hough Transform for Line Fitting

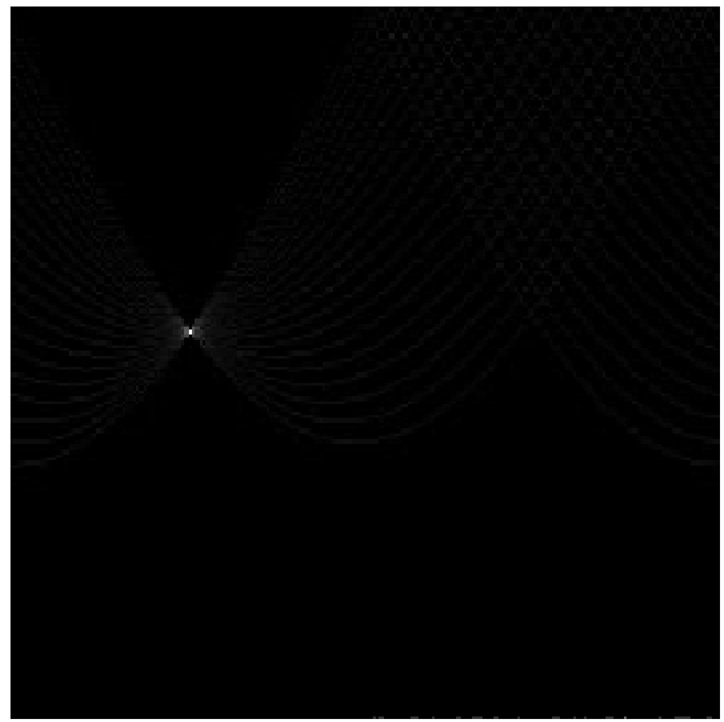
- Hough Transform
  - It may answer all three questions
  - A line is the set of point (x, y) such that
    $x\cos\theta + y\sin\theta + r = 0$

1. For a point (x_0, y_0), there is a family of lines through the point
   - Different choices of θ give different lines
2. Each point casts a vote for each line in the corresponding family
3. If there is a line that has many votes, that should be the line passing through many points

Hough Transform for Line Fitting

- Example : The Hough transform array
  - form a line



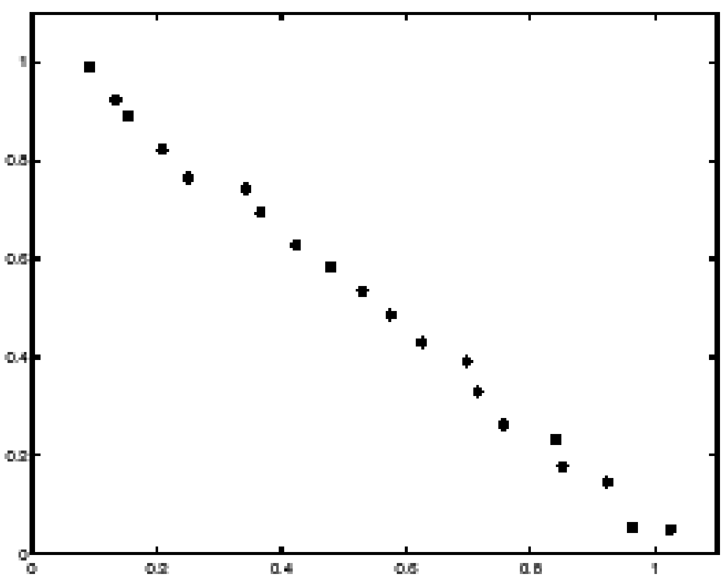


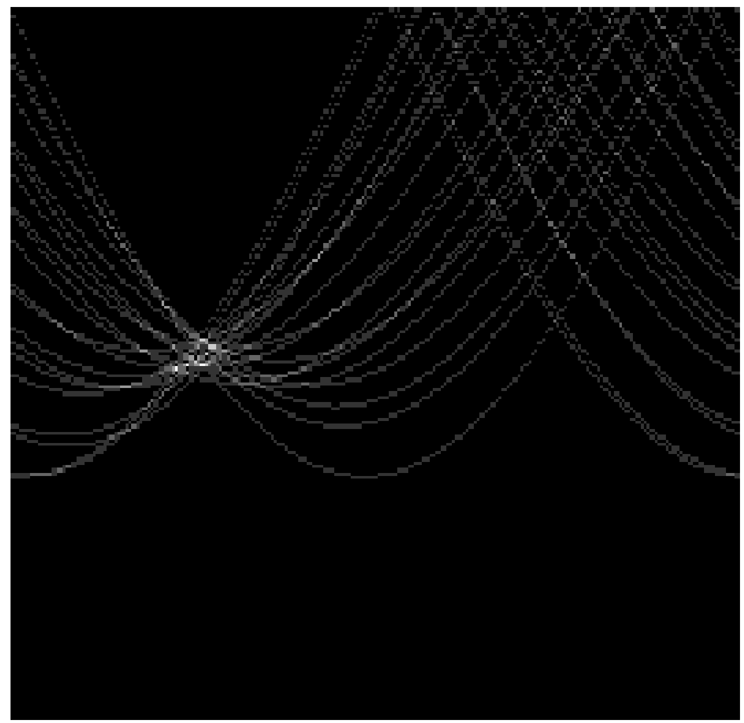
Hough Transform for Line Fitting

- Mechanics of the Hough transform
  - Construct an array representing θ, r
  - For each point, render the curve (θ, r) into this array, casting one vote to each cell
  - Difficulties
    - How big should the cells be?
      - If too big, we cannot distinguish between quite different lines
      - If too small, noise causes lines to be missed
  - How many lines?
    - Count the peaks in the array
  - Which points belong to which lines?
    - Tag the votes

Hough Transform for Line Fitting

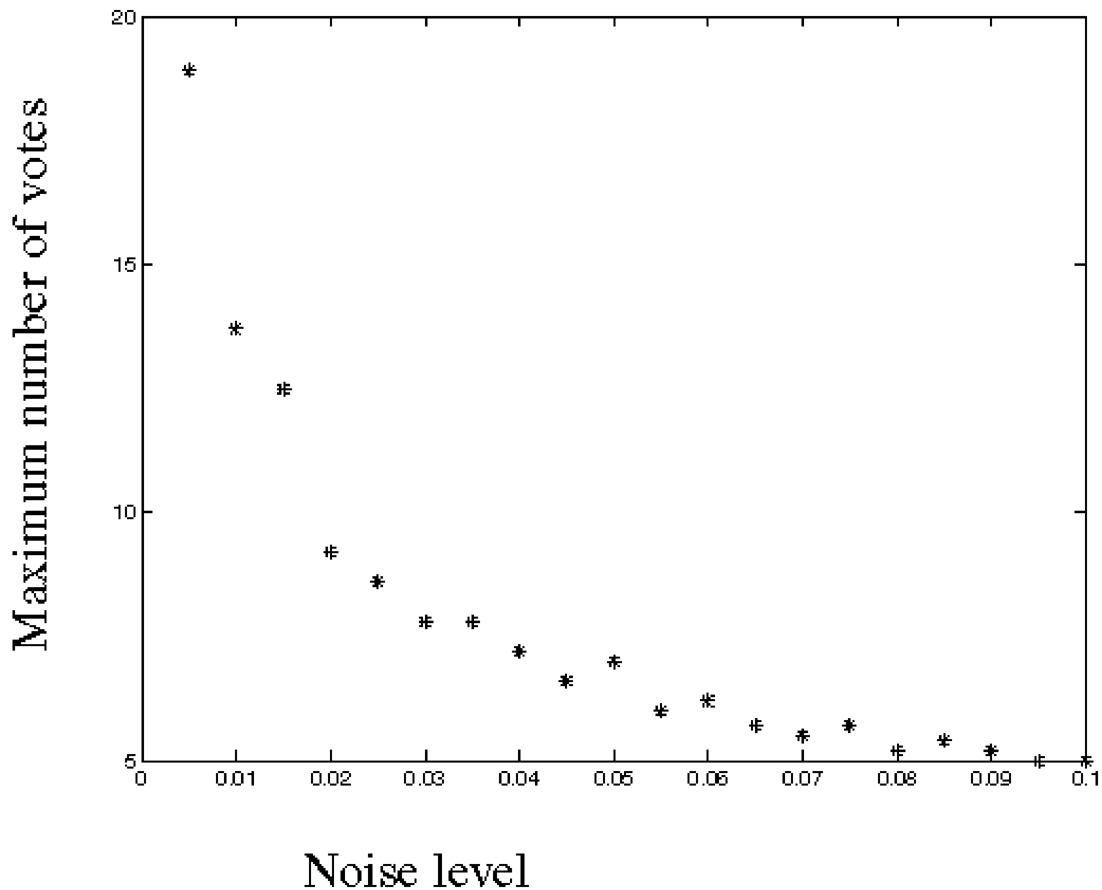
- Example : The Hough transform array
  - for a line with noises in the range [0, 0.05]





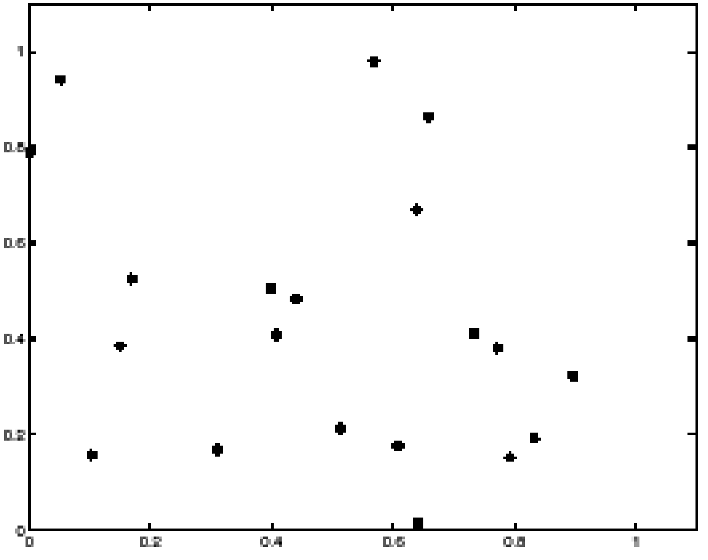
Hough Transform for Line Fitting

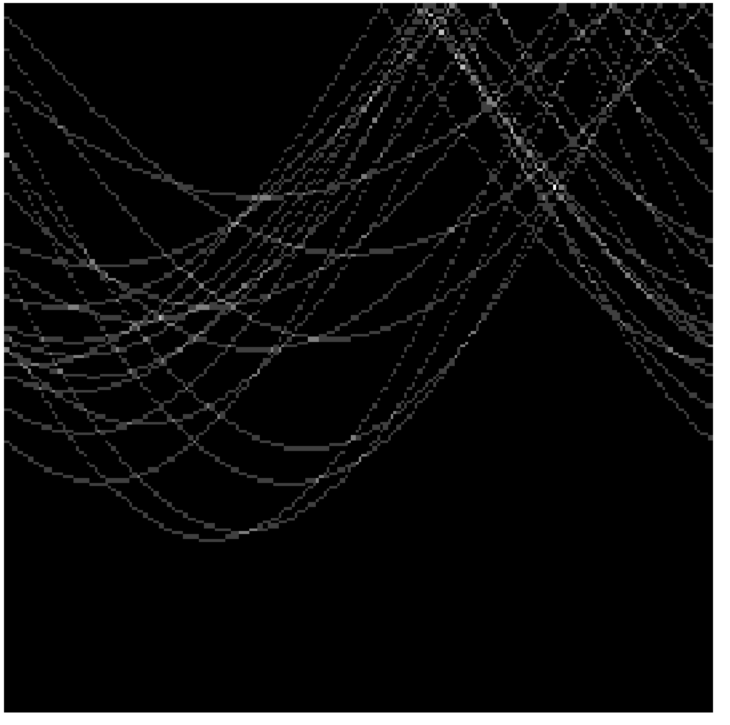
- # of votes with increasing noise level



Hough Transform for Line Fitting

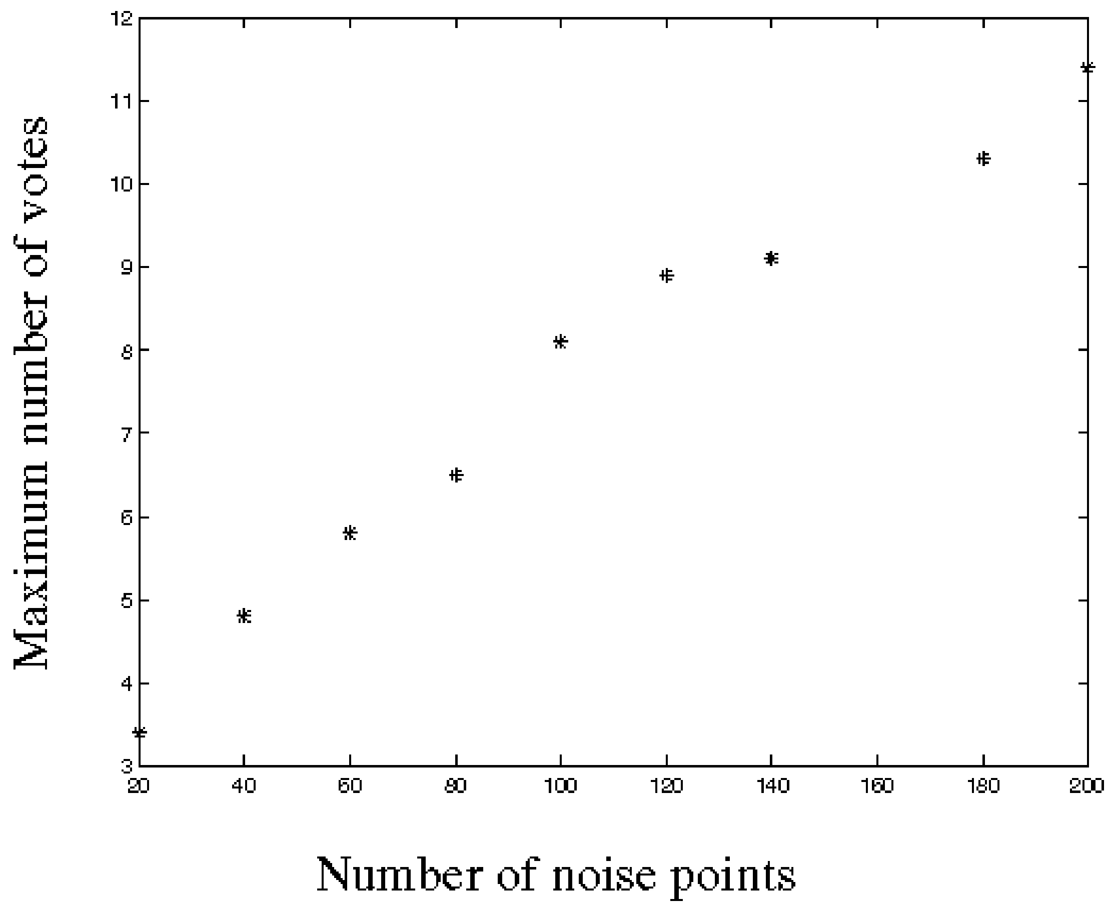
- Example : The Hough transform array
  - for random points





Hough Transform for Line Fitting

- # of votes with increasing random points

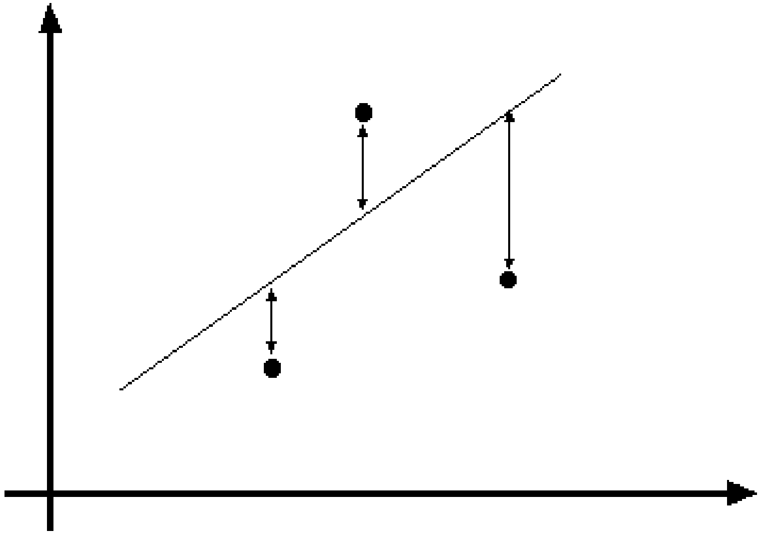


Line Fitting

Line Fitting with Least Squares

- To choose the line that minimizes

$$\sum_i (y_i - ax_i - b)^2$$

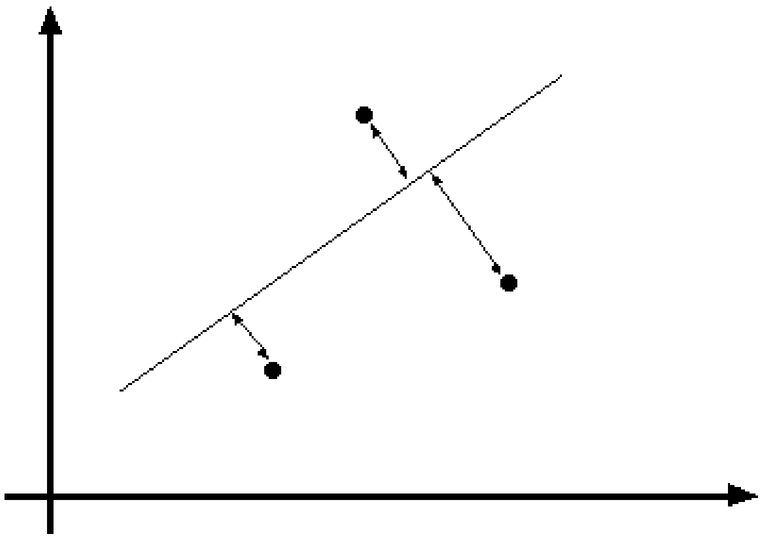


$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \overline{x^2} & \overline{x} \\ \overline{x} & 1 \end{pmatrix}^{-1} \begin{pmatrix} \overline{xy} \\ \overline{y} \end{pmatrix}$$

Line Fitting with Total Least Squares

- To choose the line that minimizes

$$\sum_i (ax_i + by_i + c)^2 \quad \text{where} \quad a^2 + b^2 = 1$$



$$\begin{pmatrix} \overline{x^2} - \overline{x}\cdot\overline{x} & \overline{xy} - \overline{x}\cdot\overline{y} \\ \overline{xy} - \overline{x}\cdot\overline{y} & \overline{y^2} - \overline{y}\cdot\overline{y} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \mu \begin{pmatrix} a \\ b \end{pmatrix}$$

$$c = -a\overline{x} - b\overline{y}$$

Which points are on which lines?

- If we know the set of points for a line, the line fitting is not difficult
- But, finding the set is difficult
- We learn two strategies
  - Incremental line fitting
  - K-means

Incremental Line Fitting

Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
   Transfer first few points on the curve to the line point list
   Fit line to line point list
   While fitted line is good enough
      Transfer the next point on the curve
         to the line point list and refit the line
   end
   Transfer last point(s) back to curve
   Refit line
   Attach line to line list
end
```

K-Means Line Fitting

Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

Hypothesize k lines (perhaps uniformly at random)
or
Hypothesize an assignment of lines to points
  and then fit lines using this assignment

Until convergence
  Allocate each point to the closest line
  Refit lines
end

- Note we are minimizing $\sum_{l_i \in \text{lines}} \sum_{x_j \in \text{data from } i\text{th line}} \text{dist}(l_i, x_j)^2$

Curve Fitting

- In principle, an easy generalization from line fitting

- In practice, rather hard
  - It is generally difficult to compute the distance between a point and a curve

Dealing with Outliers

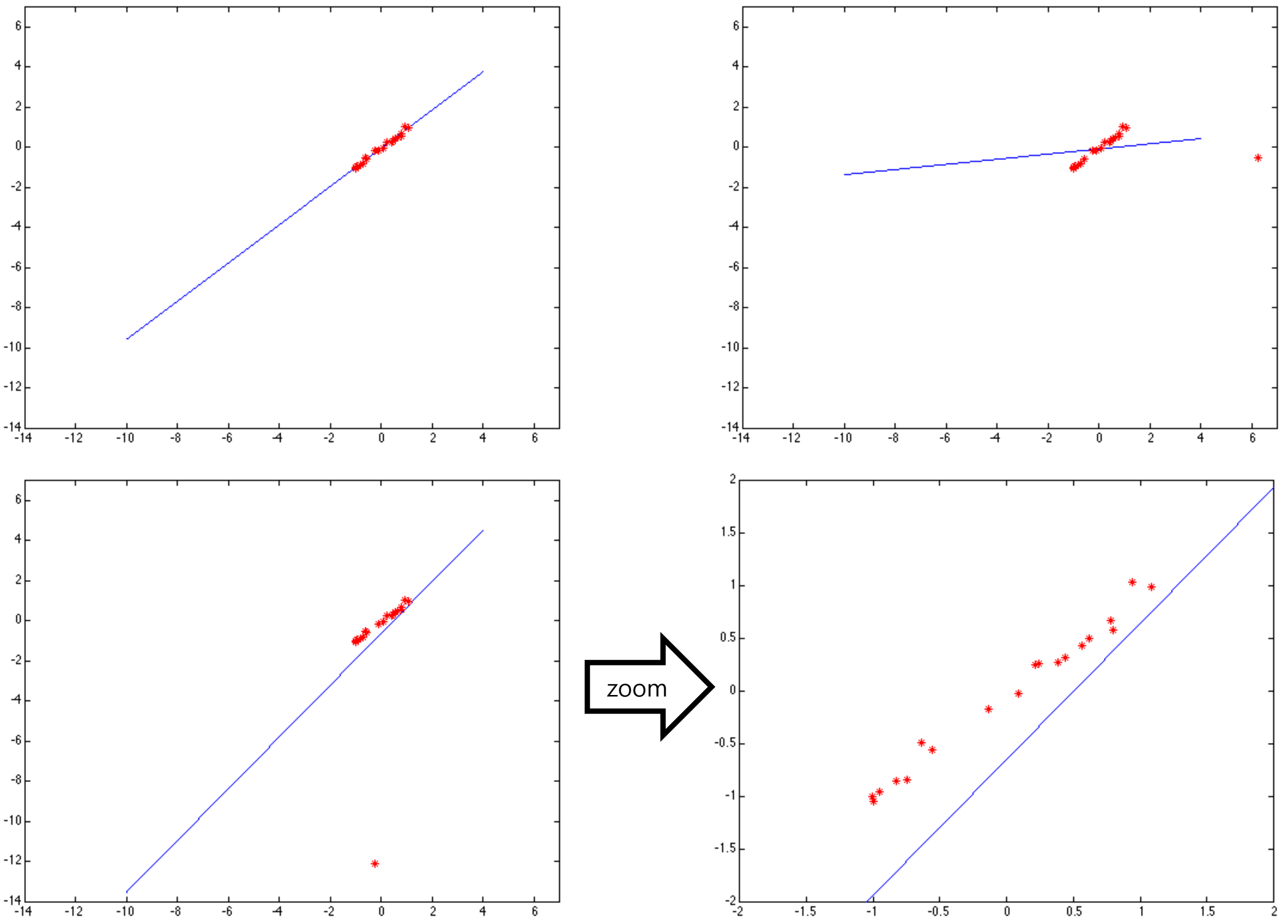
Robustness

- Poor fits in practice
  - Line fitting methods involve squared error terms
  - Squared errors can cause bias in the presence of noise points

- Robustness
  - M-estimators
    - Square nearby points but threshold far away points
  - RANSAC
    - Search for good points

Robustness

- Least-squares is sensitive to outliers



Robust M-Estimator

- It estimates parameters by minimizing

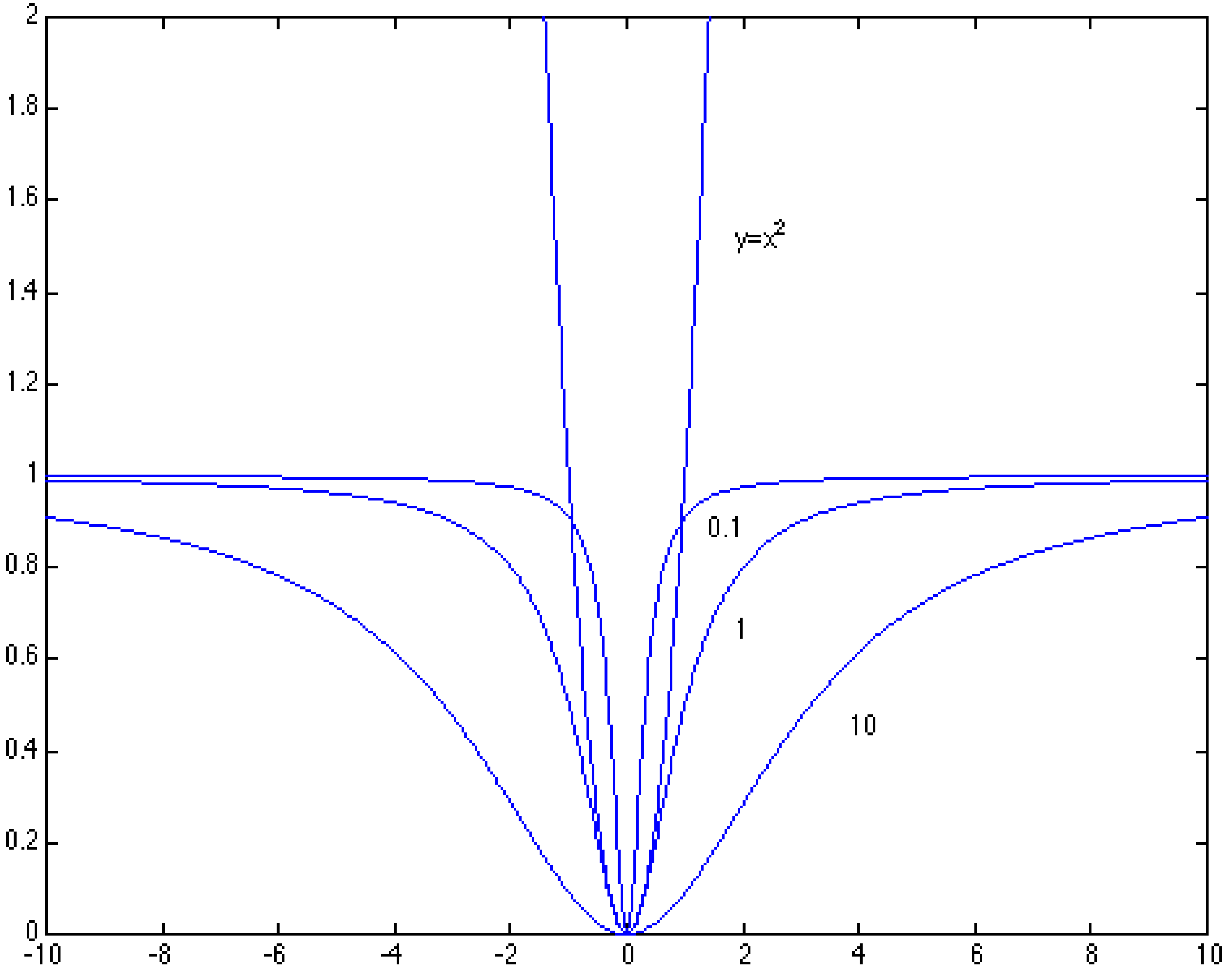
  $$\sum_i \rho(d_i;\sigma) \quad \text{instead of} \quad \sum_i d_i^{\,2}$$

- Common choice

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

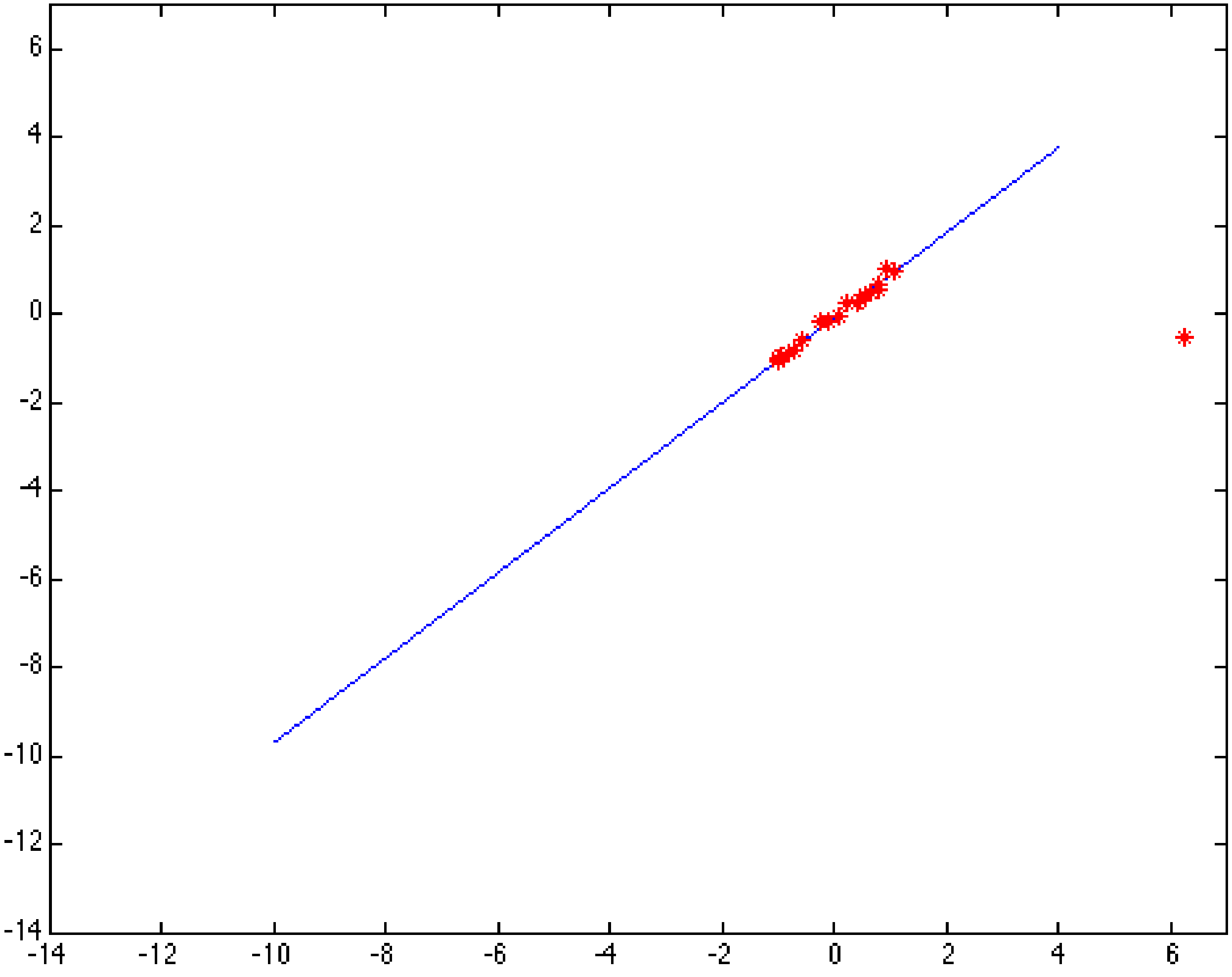
  – The scale parameter σ controls the point at which the function flattens out

Robust M-Estimator



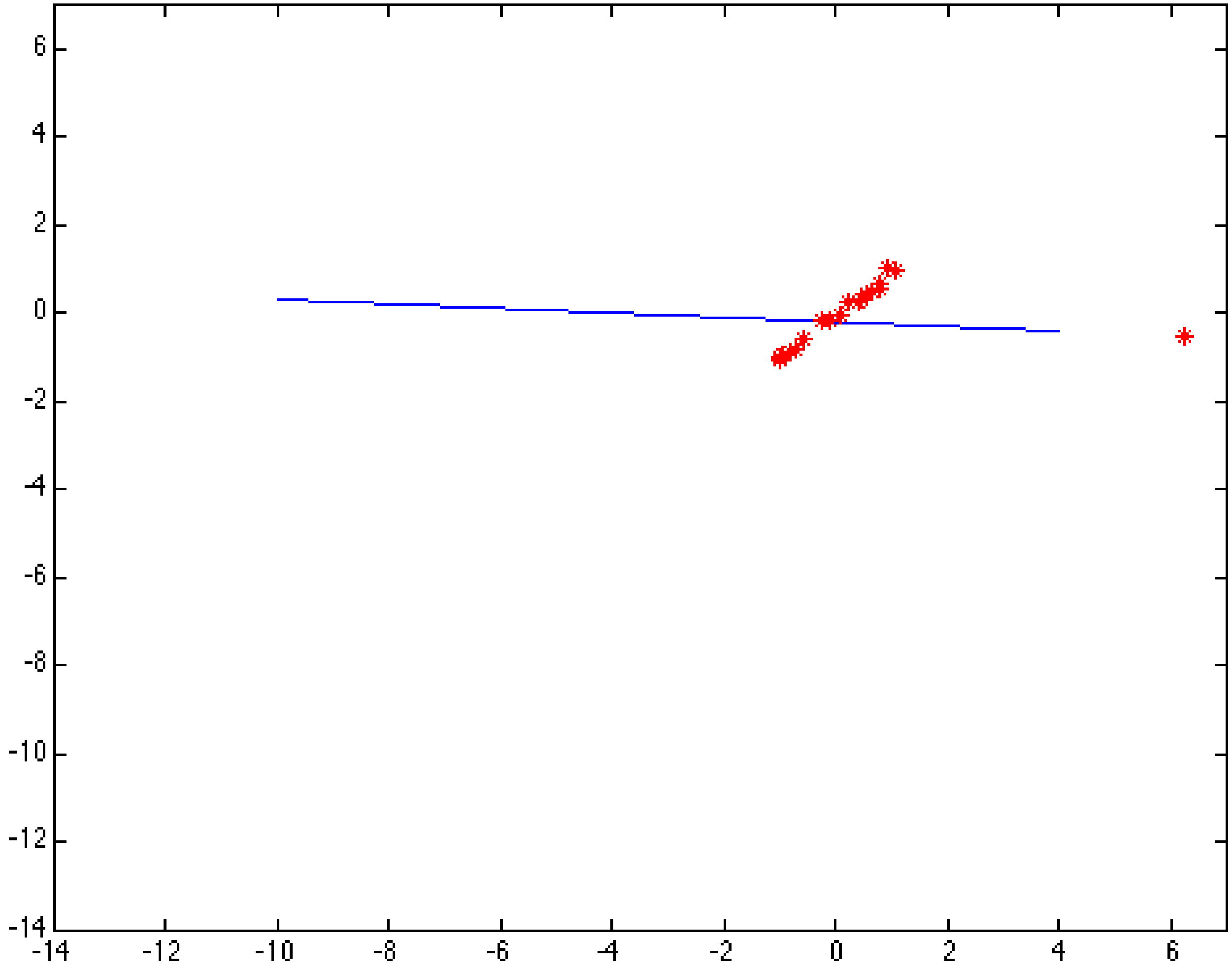
σ^2 = 0.1, 1, 10

Robustness



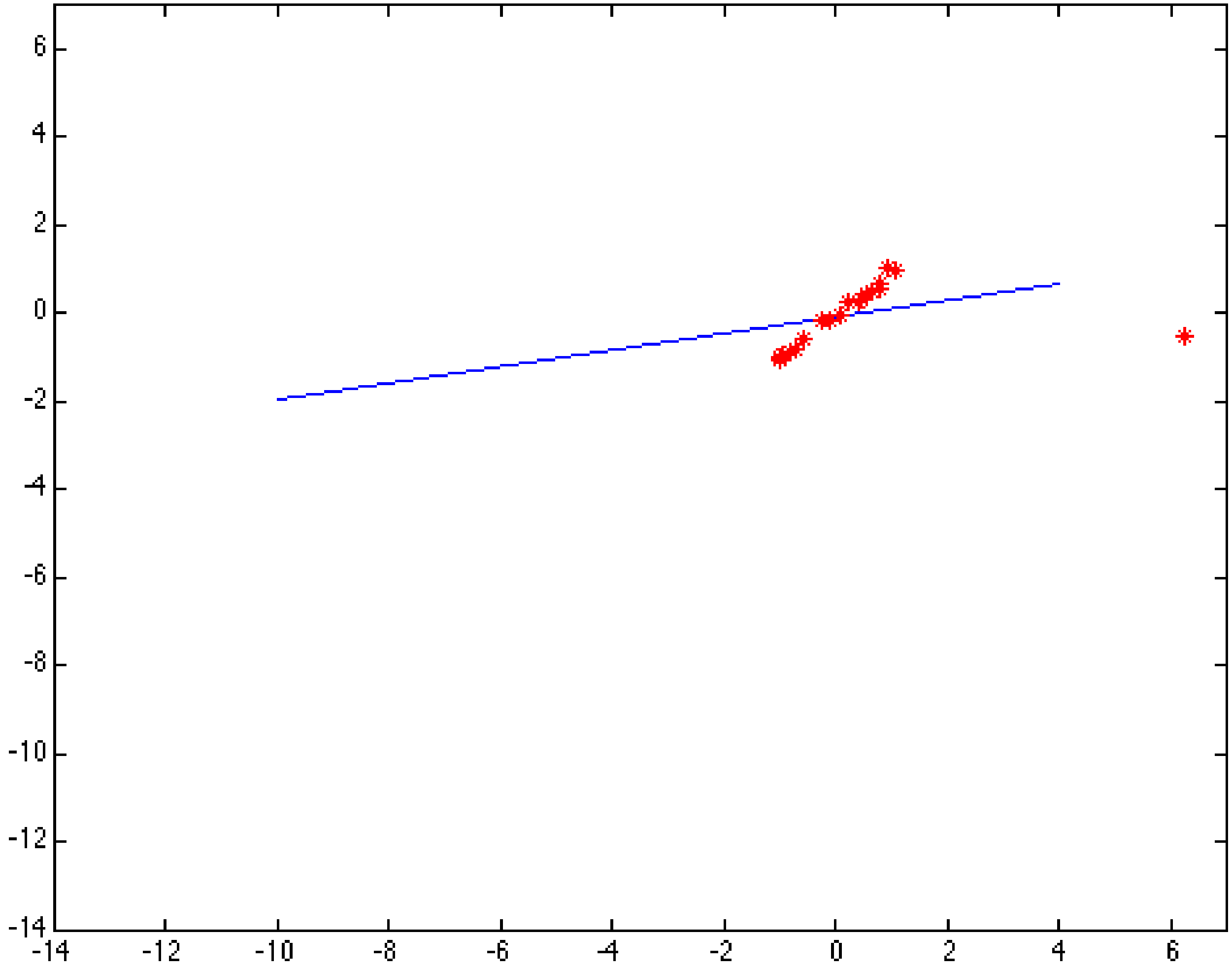
with an appropriate choice of σ

Robustness



too small σ

Robustness



too large σ

RANSAC

- Choose a small subset uniformly at random
- Fit to that
- Anything that is close to the result is signal; all others are noises
- Refit
- Do this many times and choose the best

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
  n — the smallest number of points required
  k — the number of iterations required
  t — the threshold used to identify a point that fits well
  d — the number of nearby points required
    to assert a model fits well
Until k iterations have occurred
  Draw a sample of n points from the data
    uniformly and at random
  Fit to that set of n points
  For each data point outside the sample
    Test the distance from the point to the line
      against t; if the distance from the point to the line
      is less than t, the point is close
  end
  If there are d or more points close to the line
    then there is a good fit. Refit the line using all
    these points.
end
Use the best fit from this collection, using the
  fitting error as a criterion

RANSAC

- RANdom SAmple Consensus
  - Issues
    - How many times?
      - Often enough that we are likely to have a good line
    - How big a subset?
      - Smallest possible
    - What does ‘close’ mean?
      - Depends on the problem
    - What is a good line?
      - One where the number of nearby points is so big that it is unlikely to be all outliers

RANSAC

- How many times (k)?
  - Method 1
    - The first successful subset is found at the r-th iteration
    - w =probability of non-outlier

$$E[r] = w^{-n}, \quad \mathrm{std}[r] = \frac{\sqrt{1-w^n}}{w^n}$$

    - $k \geq E[r] + c \cdot \mathrm{std}[r]$
  - Method 2
    - $q =$ the allowed probability of experiencing only bad iterations
    - $(1 - w^n)^k \leq q$