Computer Vision

Fitting

Marc Pollefeys
COMP 256

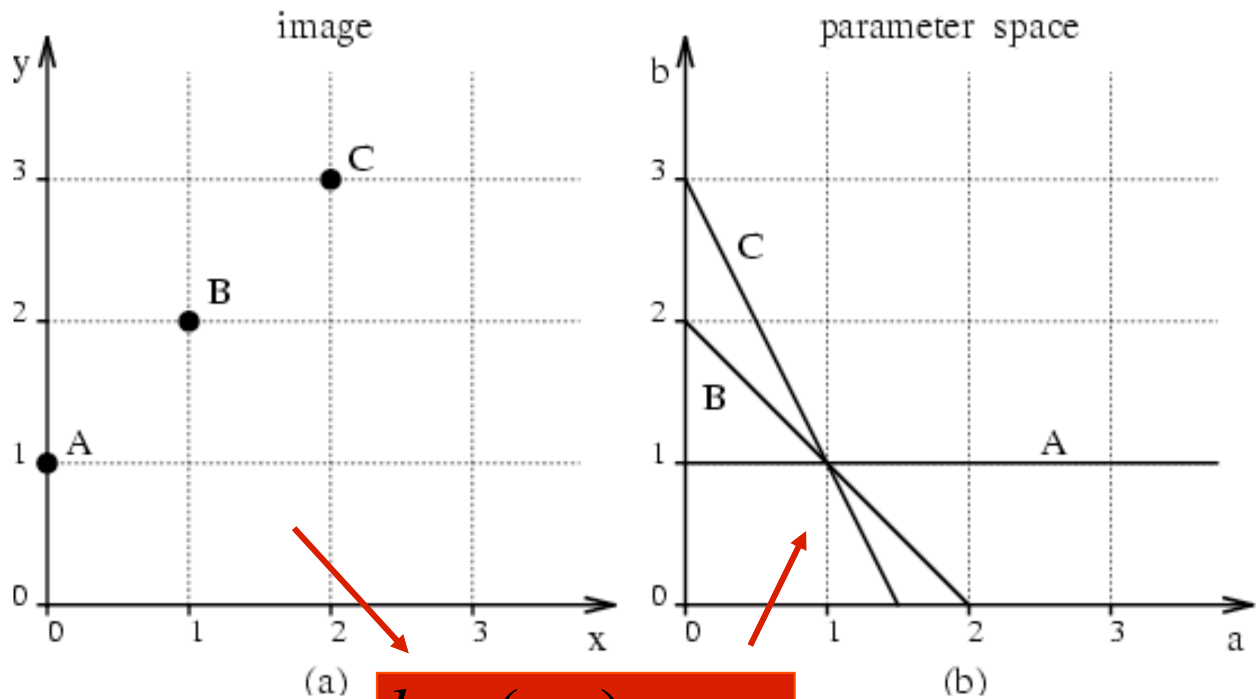
Some slides and illustrations from D. Forsyth, T. Darrel, A. Zisserman, ...

Fitting

- Choose a parametric object/some objects to represent a set of tokens
- Most interesting case is when criterion is not local
  - can't tell whether a set of points lies on a line by looking only at each point and the next.
- Three main questions:
  - what object represents this set of tokens best?
  - which of several objects gets which token?
  - how many objects are there?

(you could read line for object here, or circle, or ellipse or...)

Hough transform : straight lines



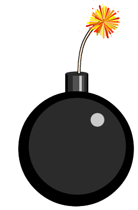
$$b = (-x)\,a + y$$

implementation :

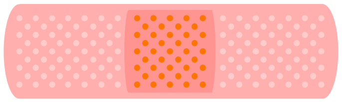
1. the parameter space is discretised
2. a counter is incremented at each cell where the lines pass
3. peaks are detected

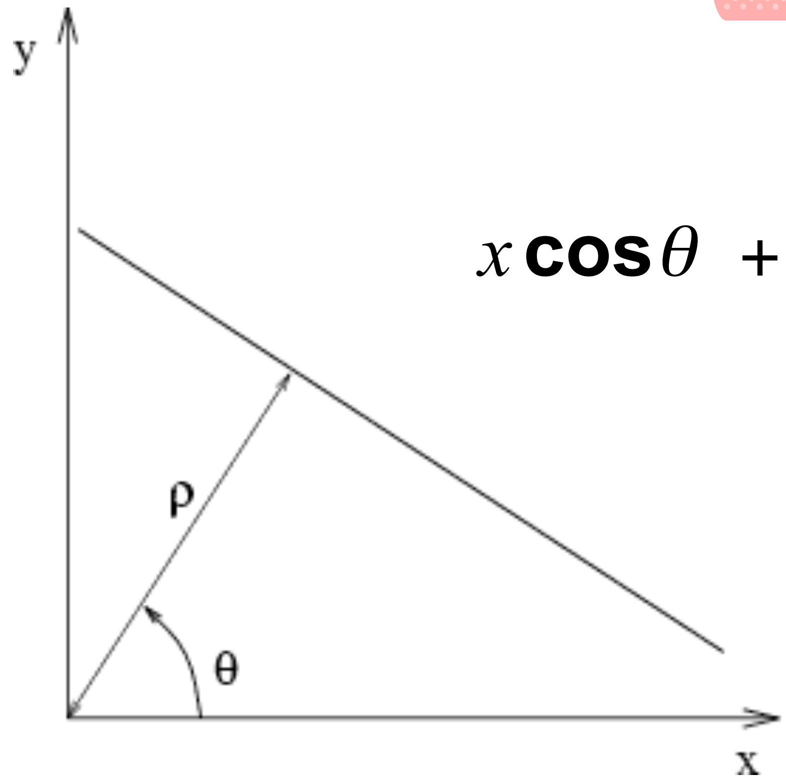
Hough transform : straight lines

problem : unbounded parameter domain
vertical lines require infinite *a*



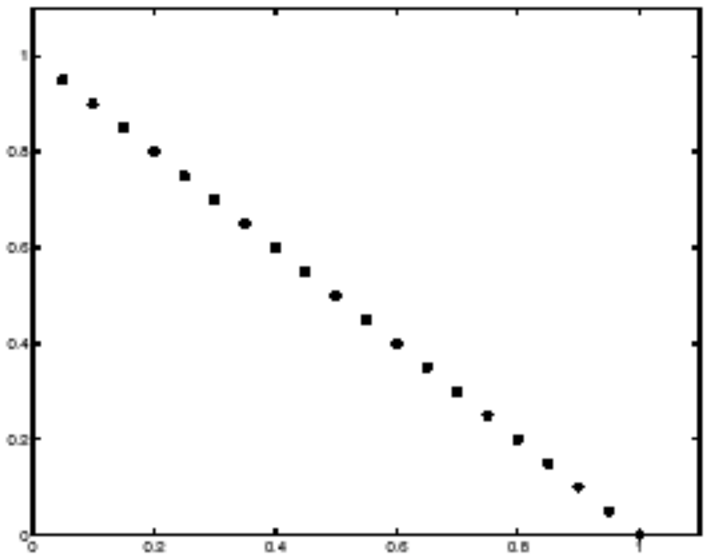
alternative representation:



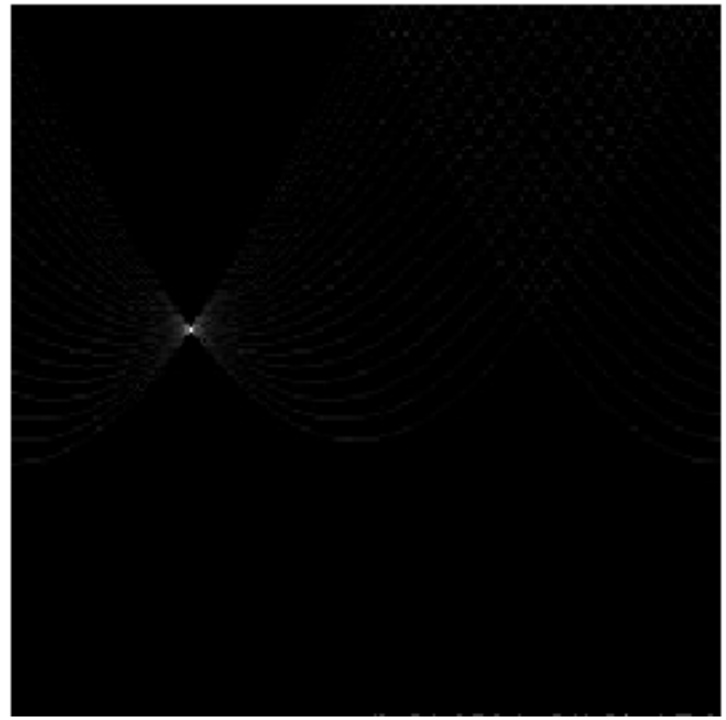


$$x\,\mathbf{cos}\,\theta + y\,\mathbf{sin}\,\theta = \rho$$

Each point will add a cosine function in the (θ,ρ) parameter space



tokens

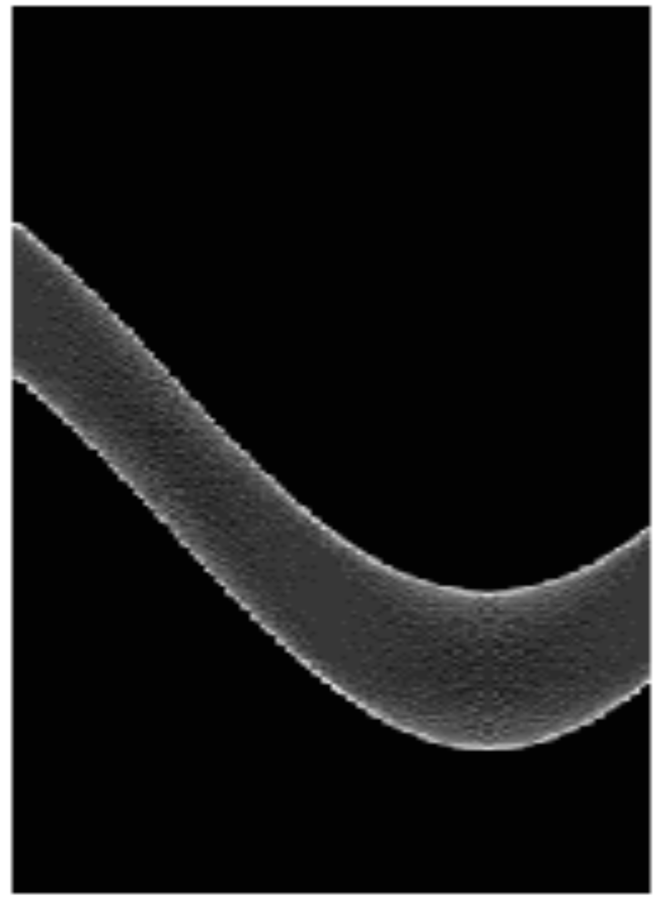


votes

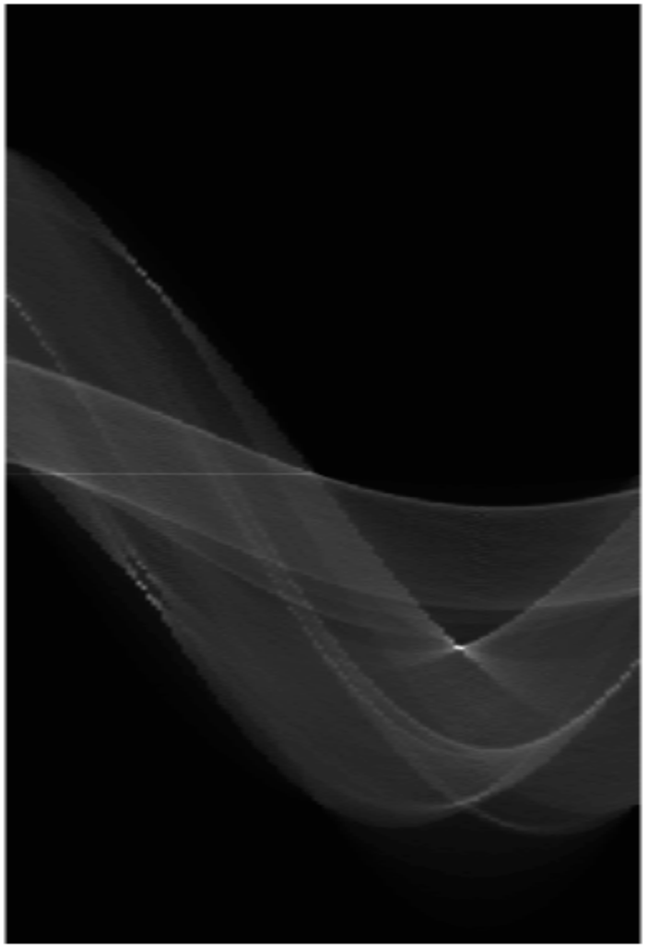
Hough transform : straight lines

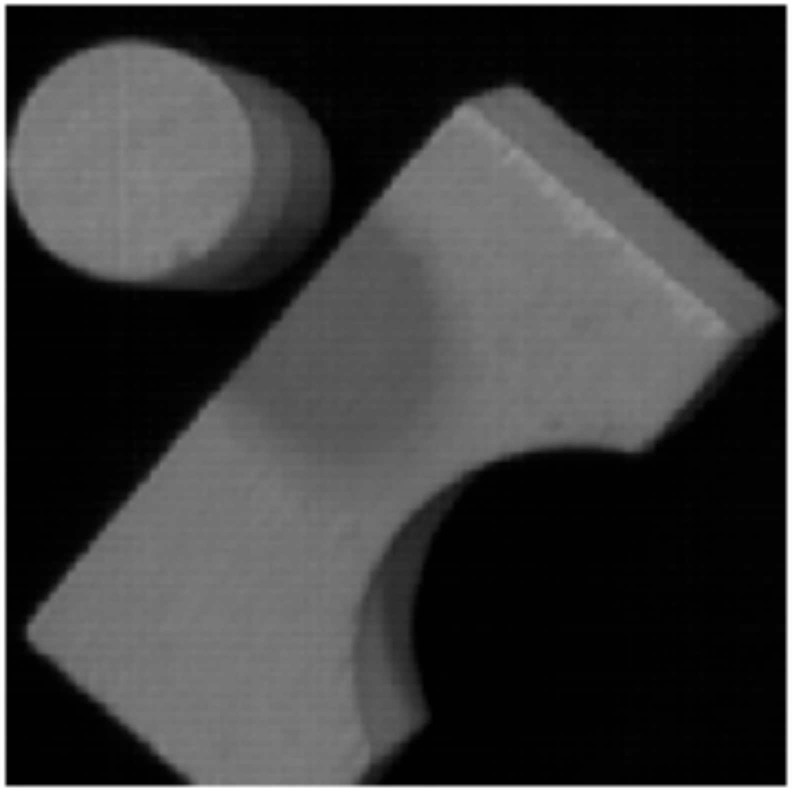
Square :

Circle :



Hough transform : straight lines



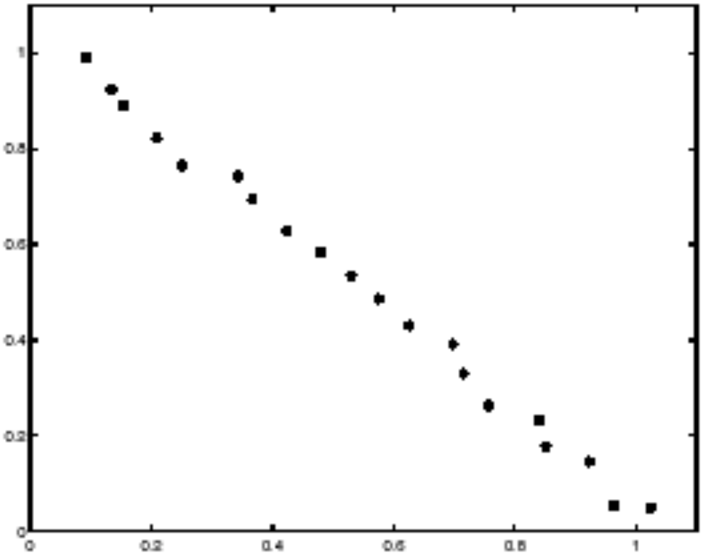


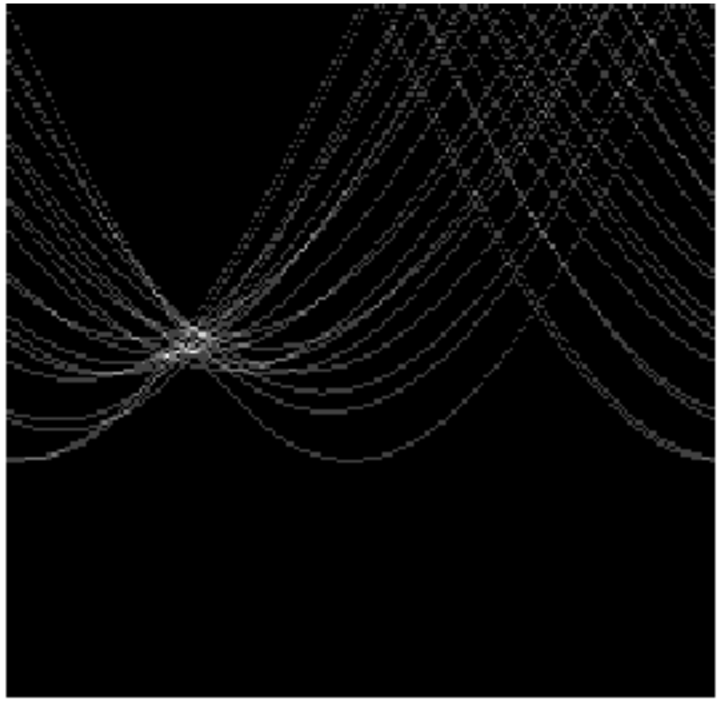
Mechanics of the Hough transform

- Construct an array representing θ, d
- For each point, render the curve (θ, d) into this array, adding one at each cell
- Difficulties
  - how big should the cells be? (too big, and we cannot distinguish between quite different lines; too small, and noise causes lines to be missed)

- How many lines?
  - count the peaks in the Hough array
- Who belongs to which line?
  - tag the votes

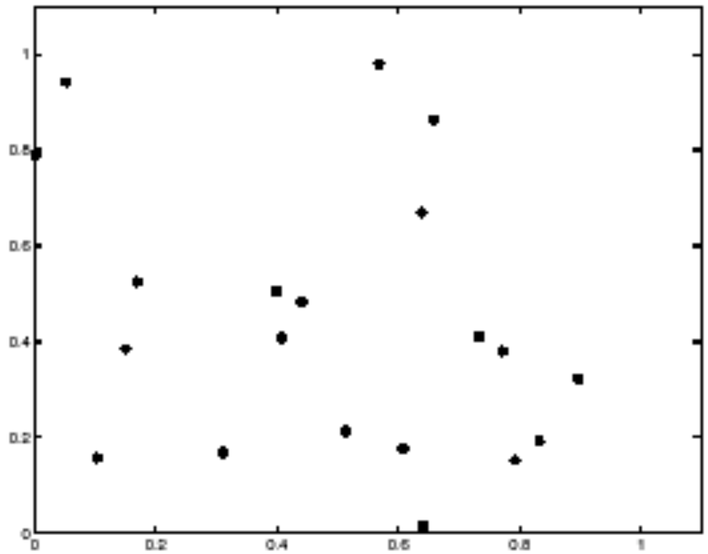
- Hardly ever satisfactory in practice, because problems with noise and cell size defeat it



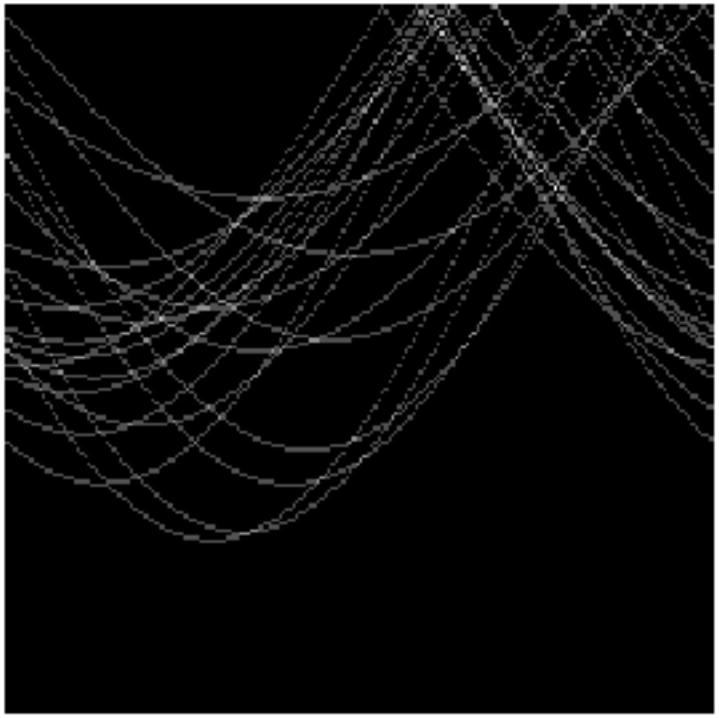


tokens

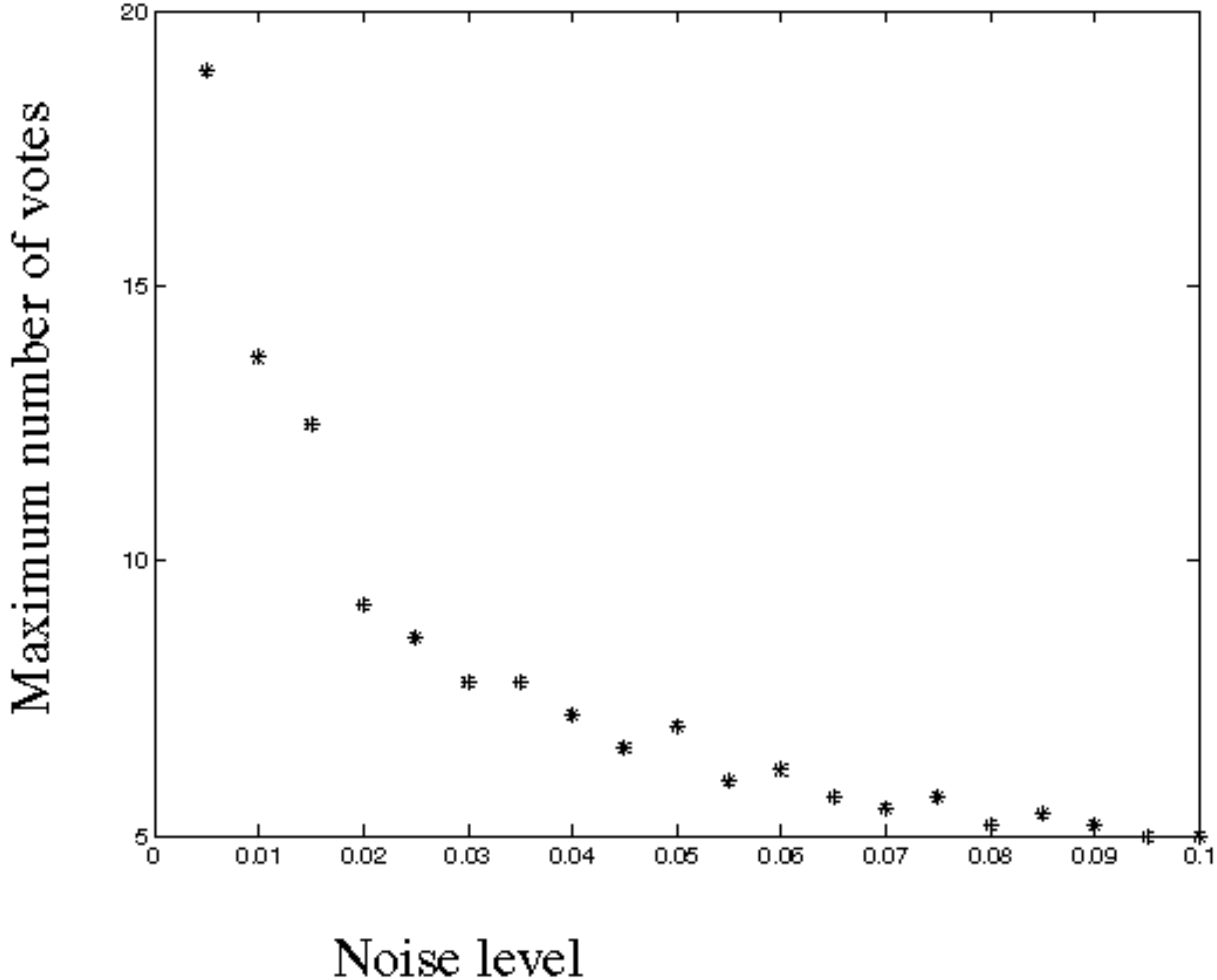
votes

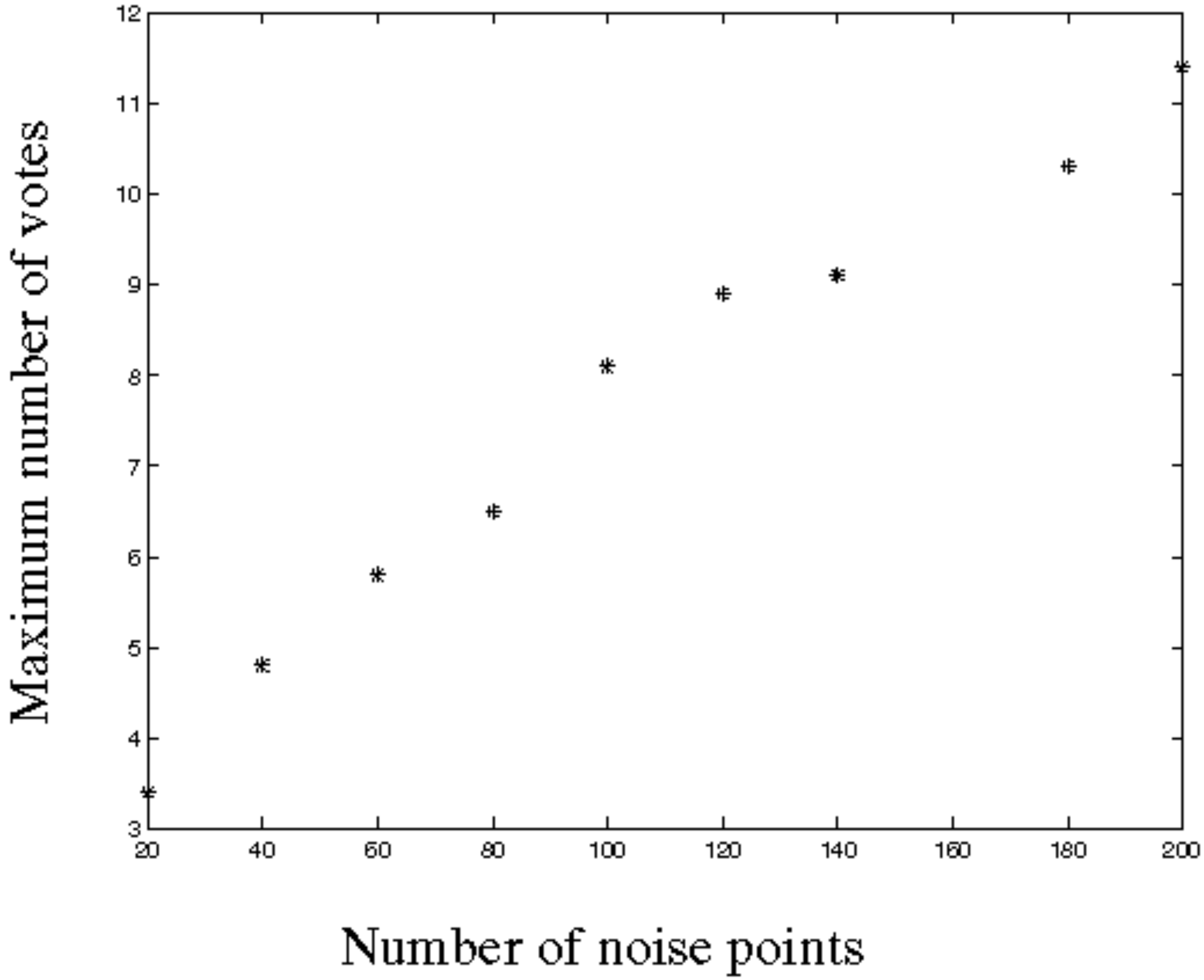


1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1



Computer Vision




Maximum number of votes
Number of noise points
12
11
10
9
8
7
6
5
4
3
20
40
60
80
100
120
140
160
180
200

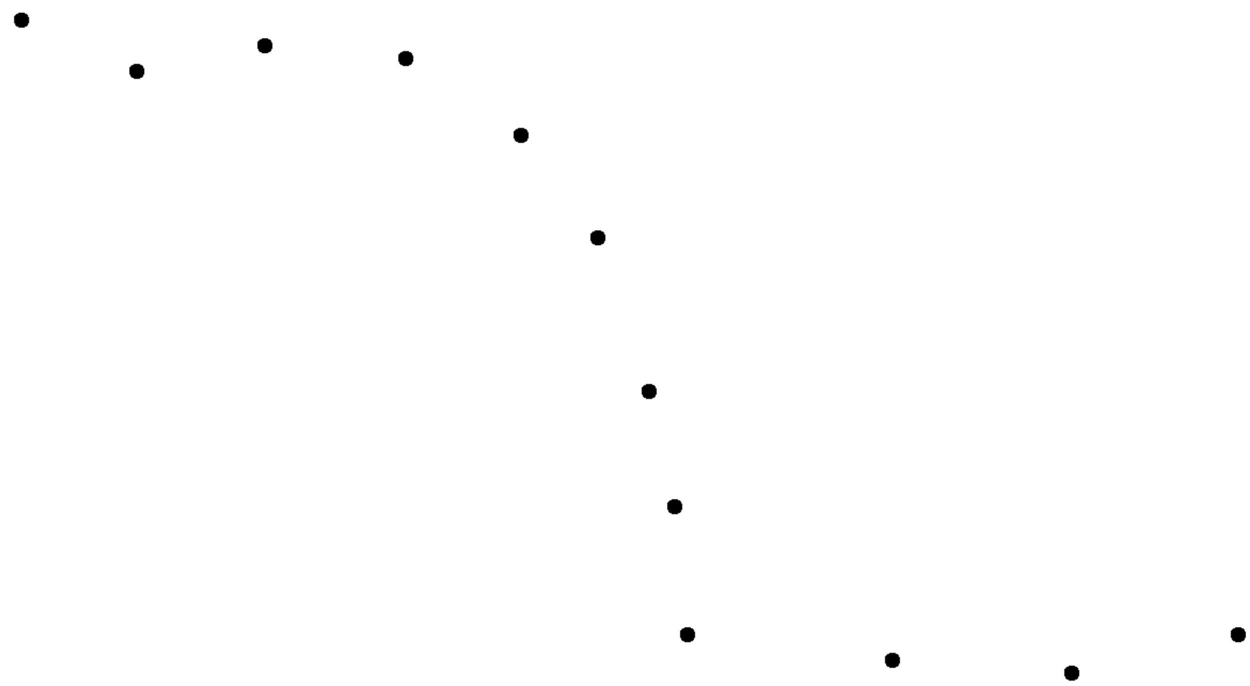
Who came from which line?

- Assume we know how many lines there are - but which lines are they?
  - easy, if we know who came from which line
- Three strategies
  - Incremental line fitting
  - K-means
  - Probabilistic (later!)

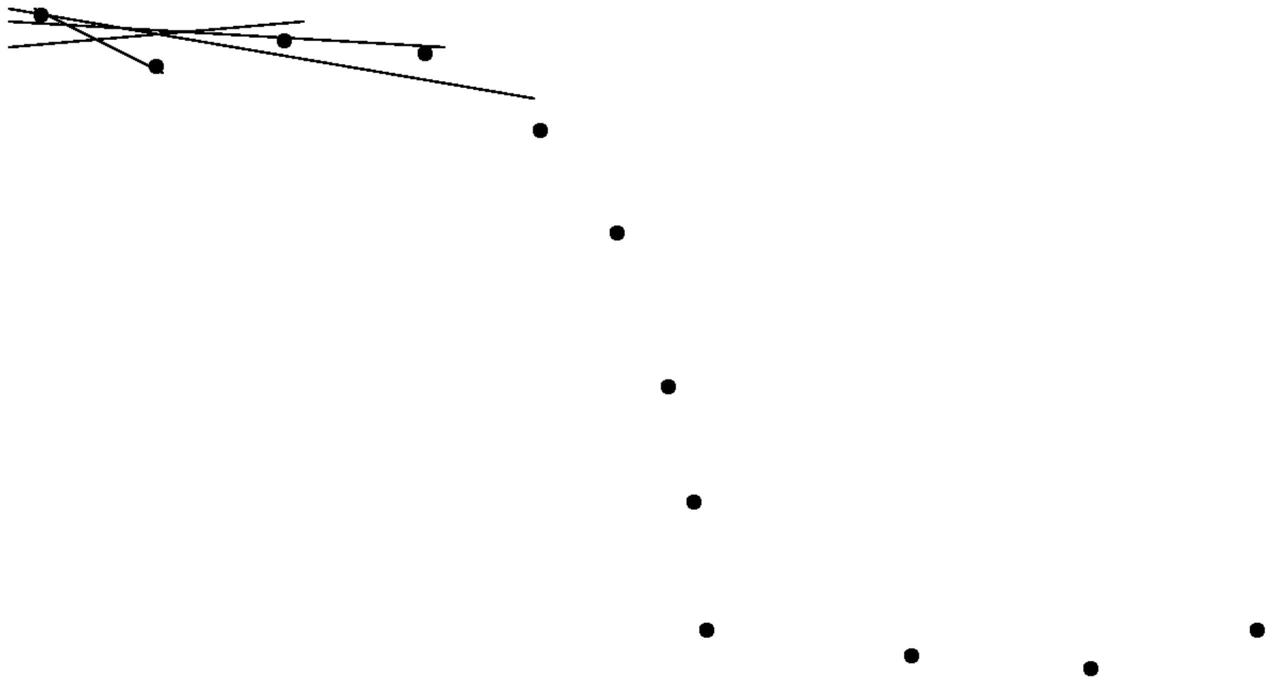
Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

```
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
```

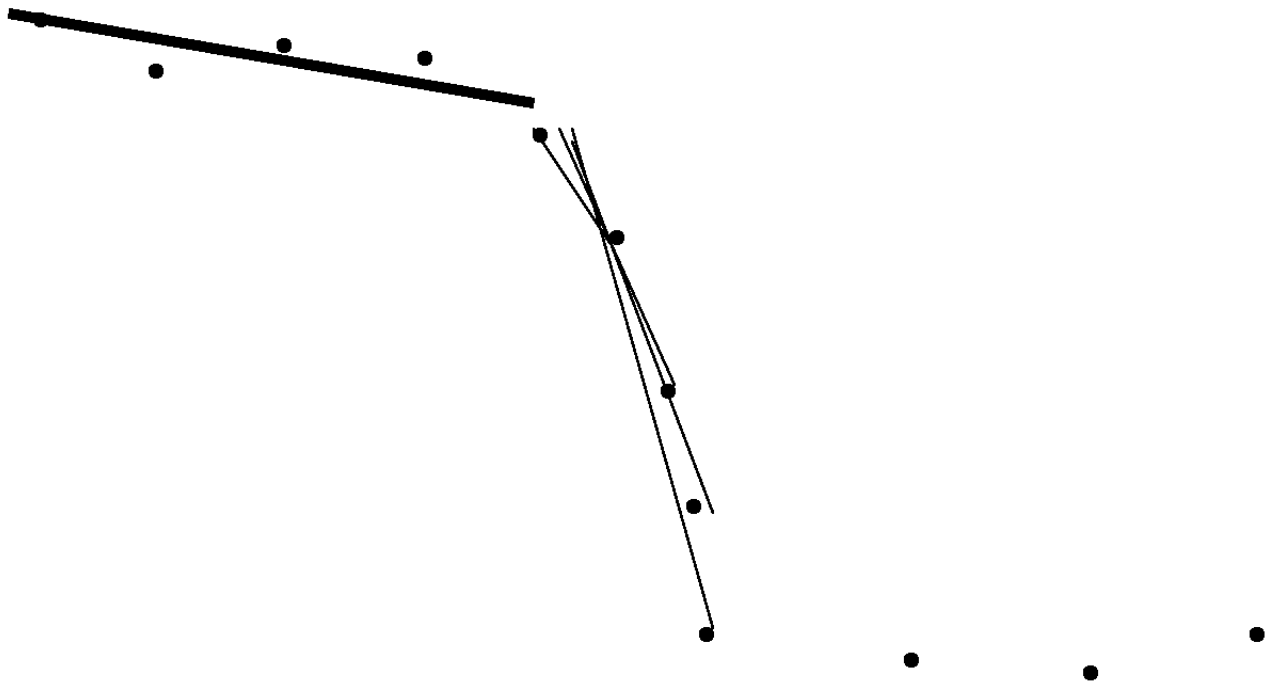
Incremental line fitting



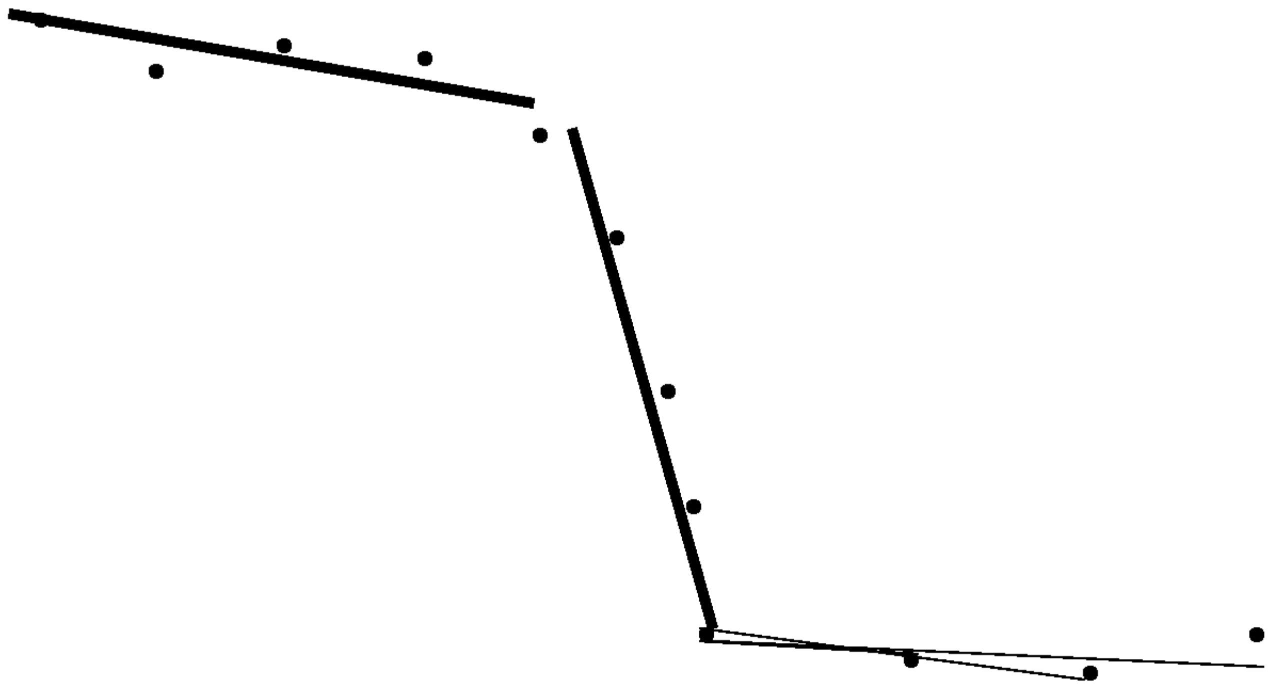
Incremental line fitting



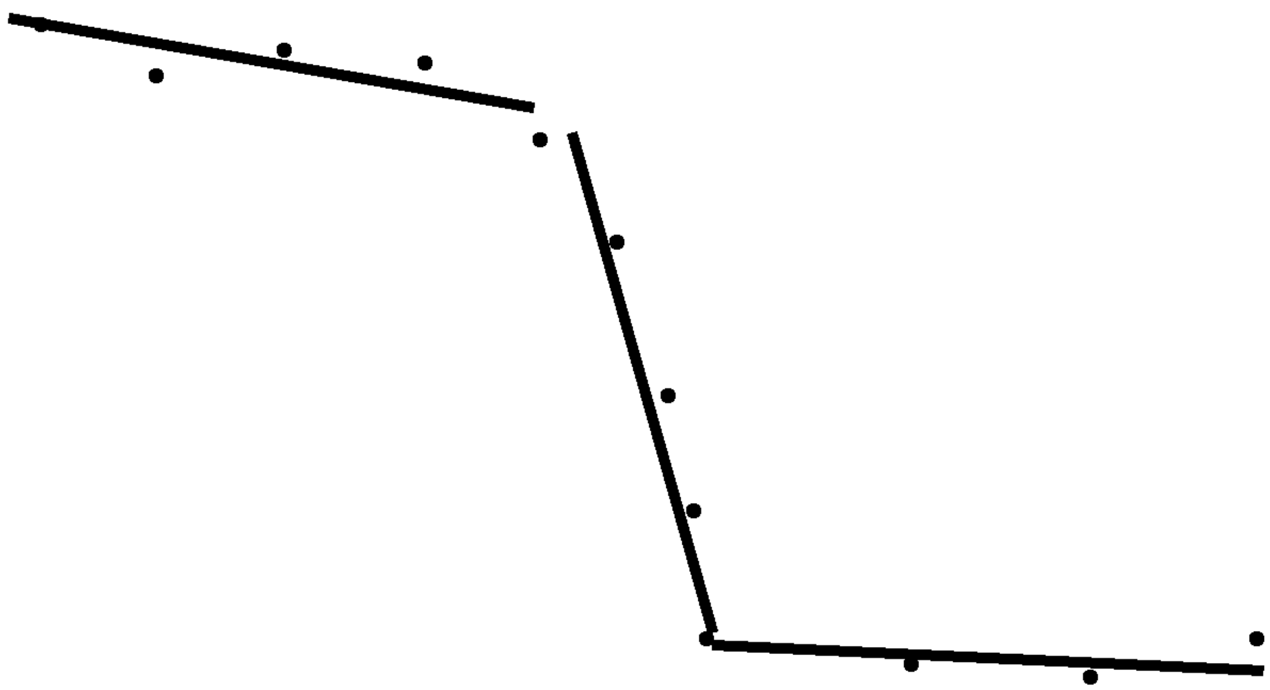
Incremental line fitting



Incremental line fitting



Incremental line fitting

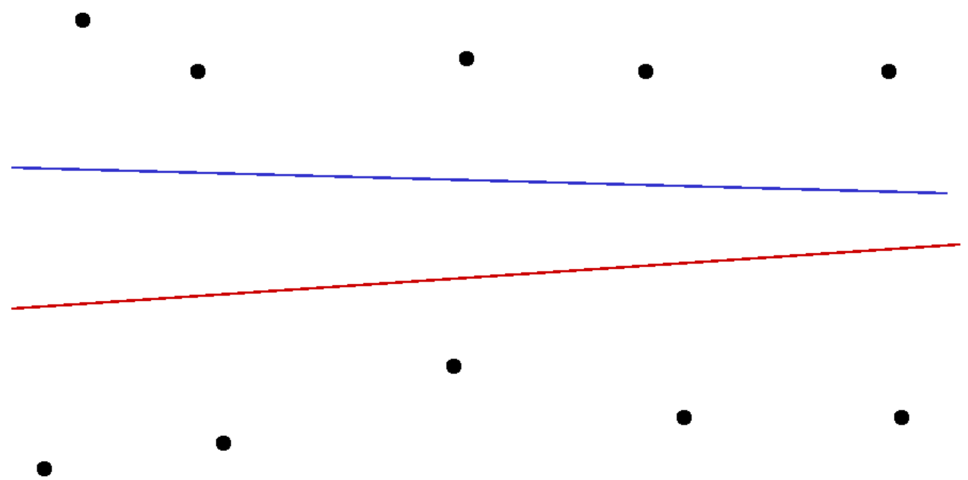


Algorithm 15.2: K-means line fitting by allocating points to the closest line and then refitting.

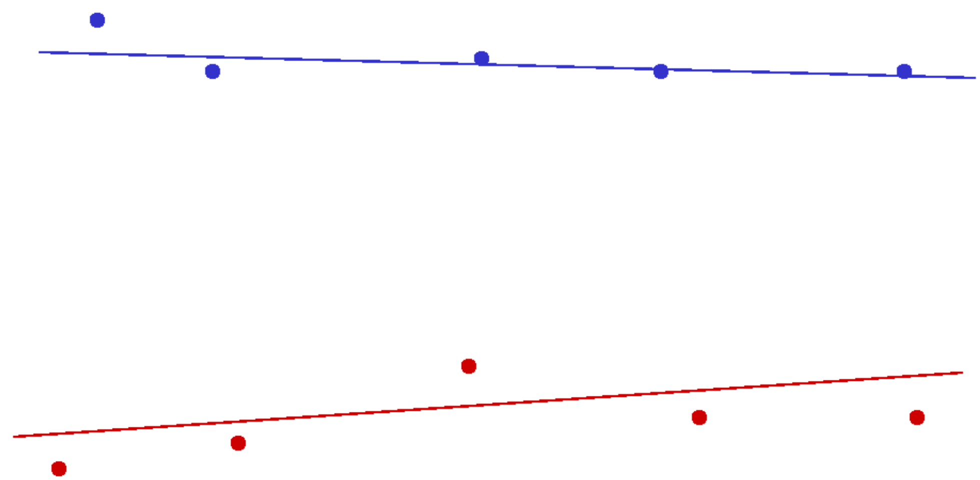
Hypothesize k lines (perhaps uniformly at random)
or
Hypothesize an assignment of lines to points
  and then fit lines using this assignment

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

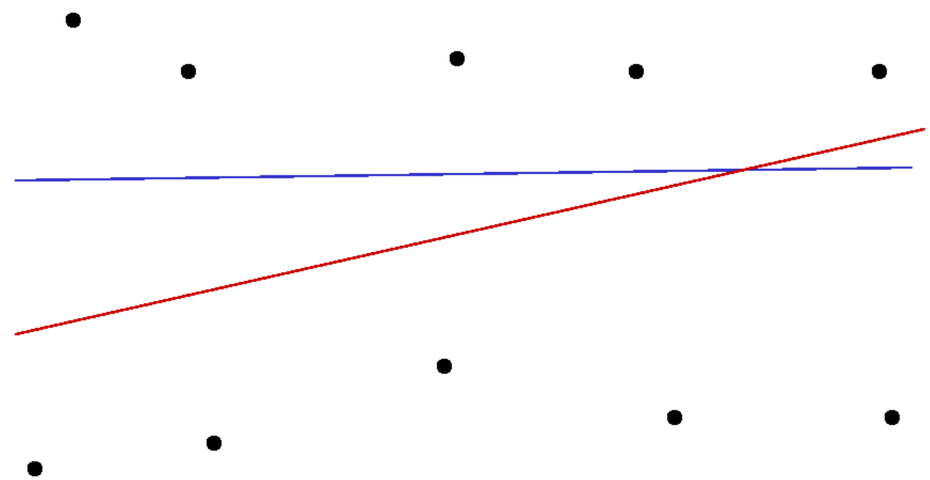
K-means line fitting



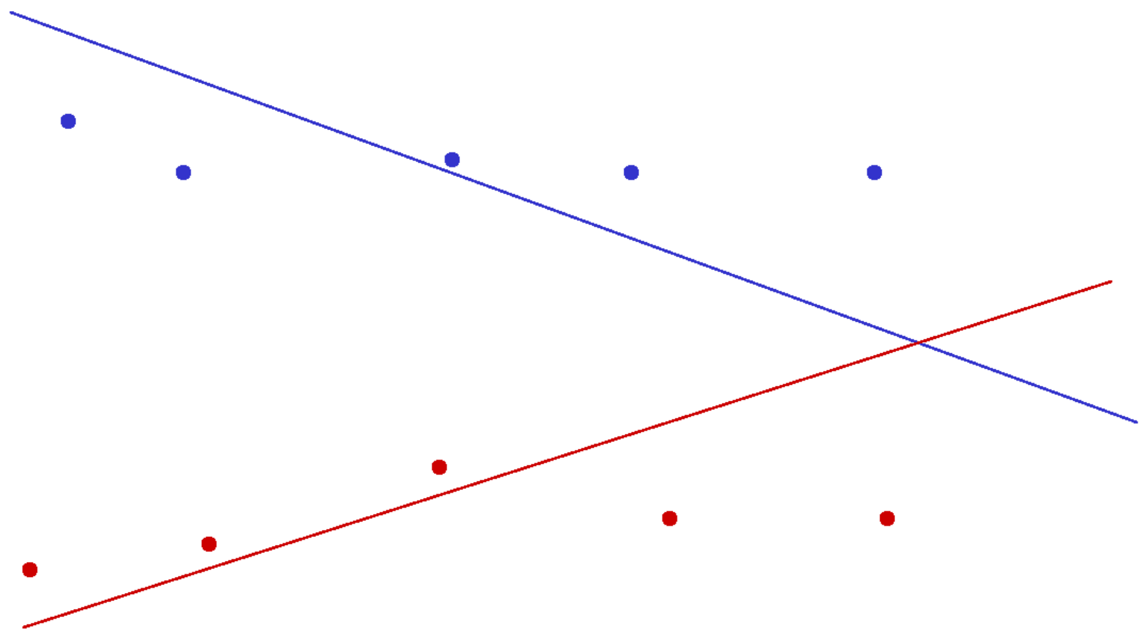
K-means line fitting



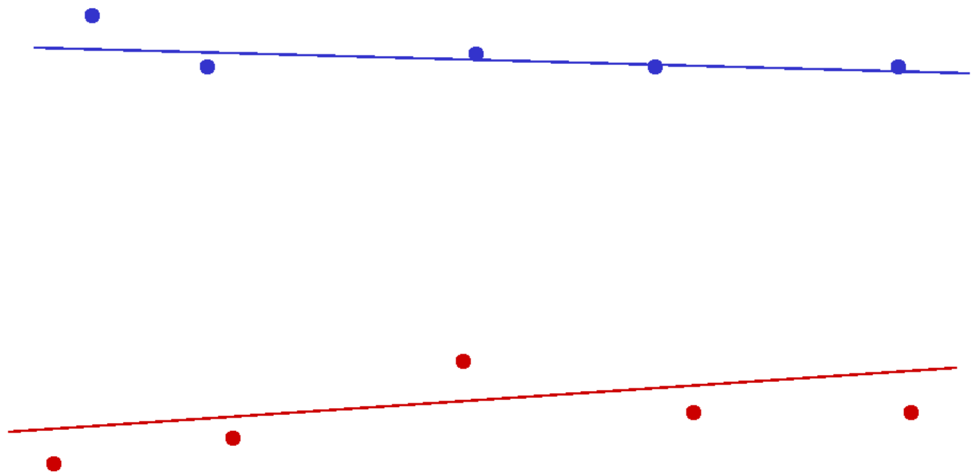
K-means line fitting



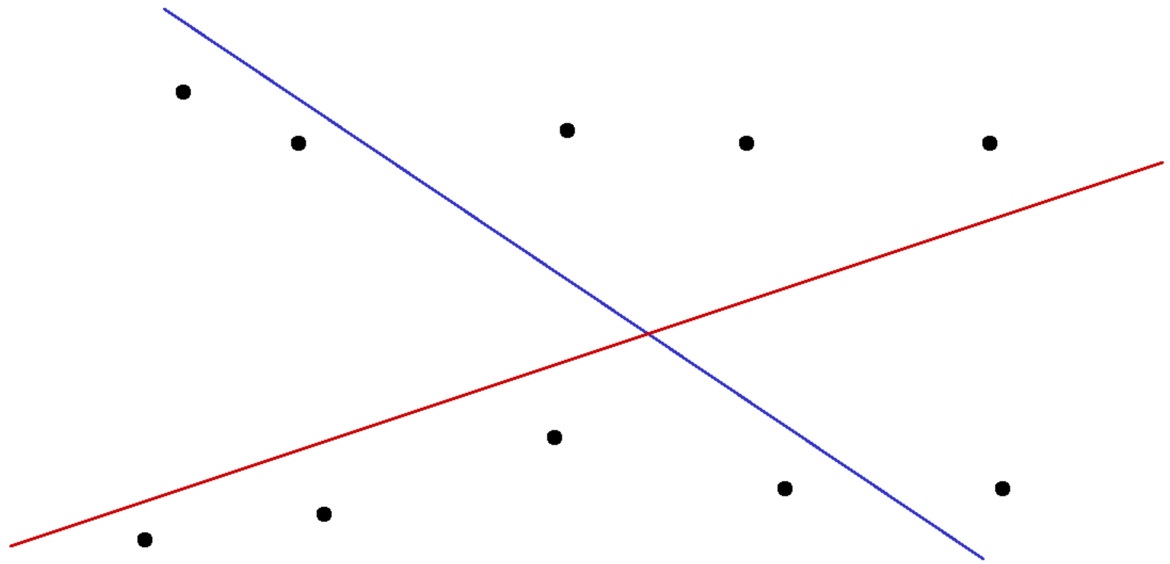
K-means line fitting



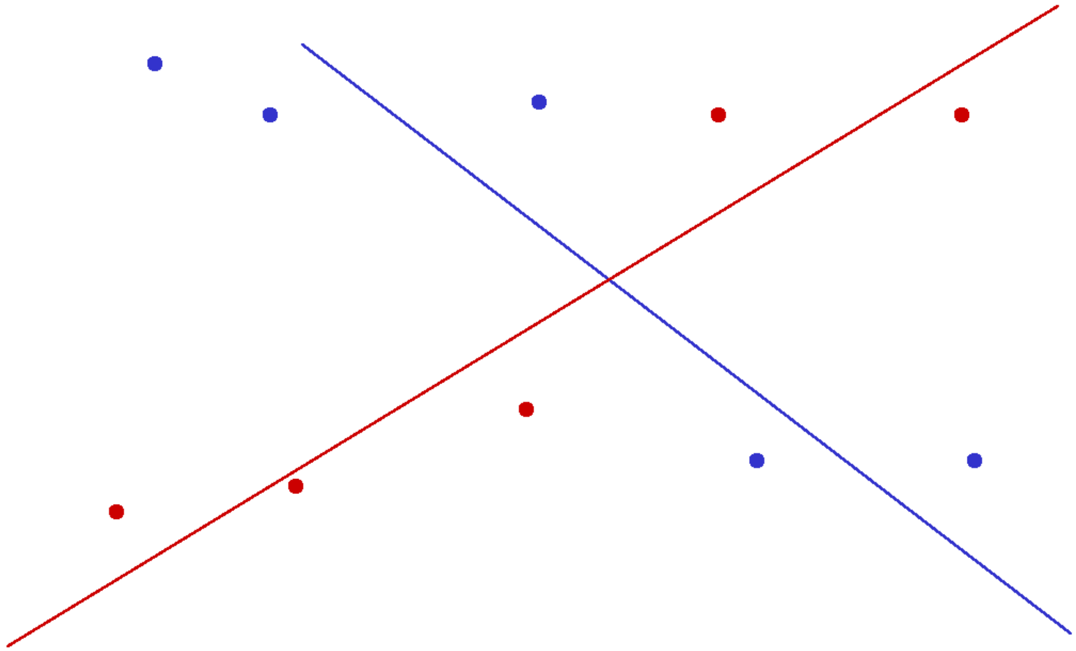
K-means line fitting



K-means line fitting



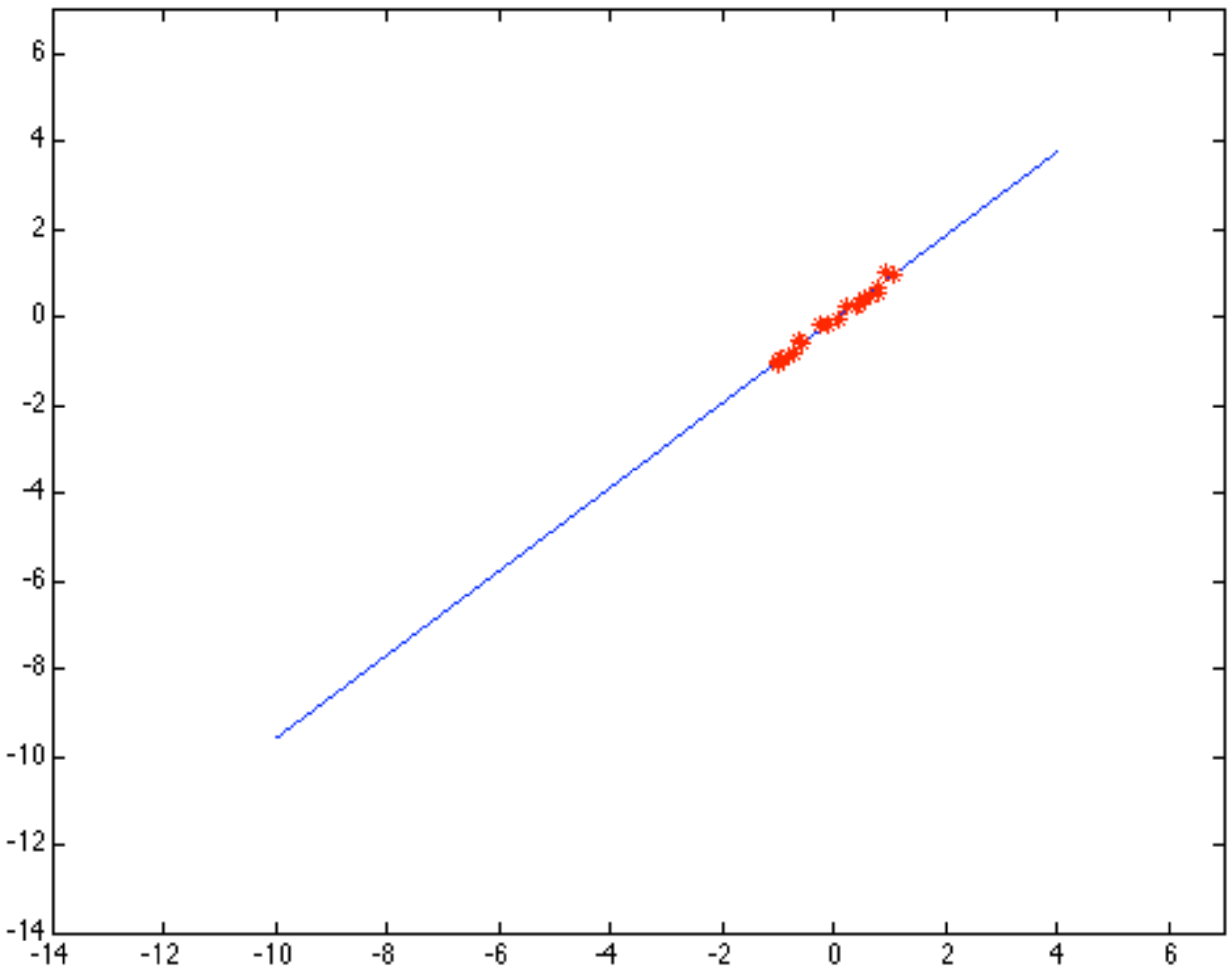
K-means line fitting

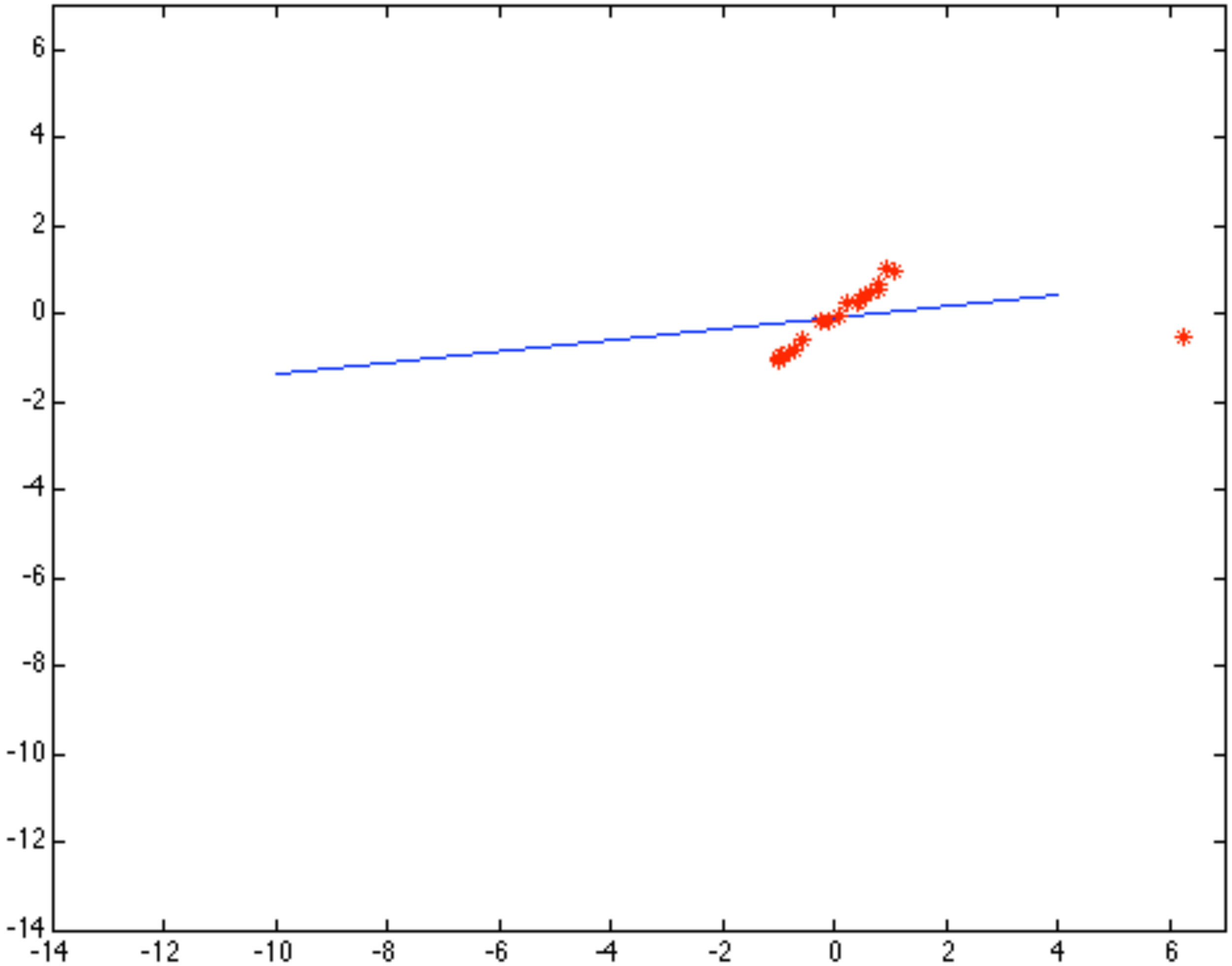


Robustness

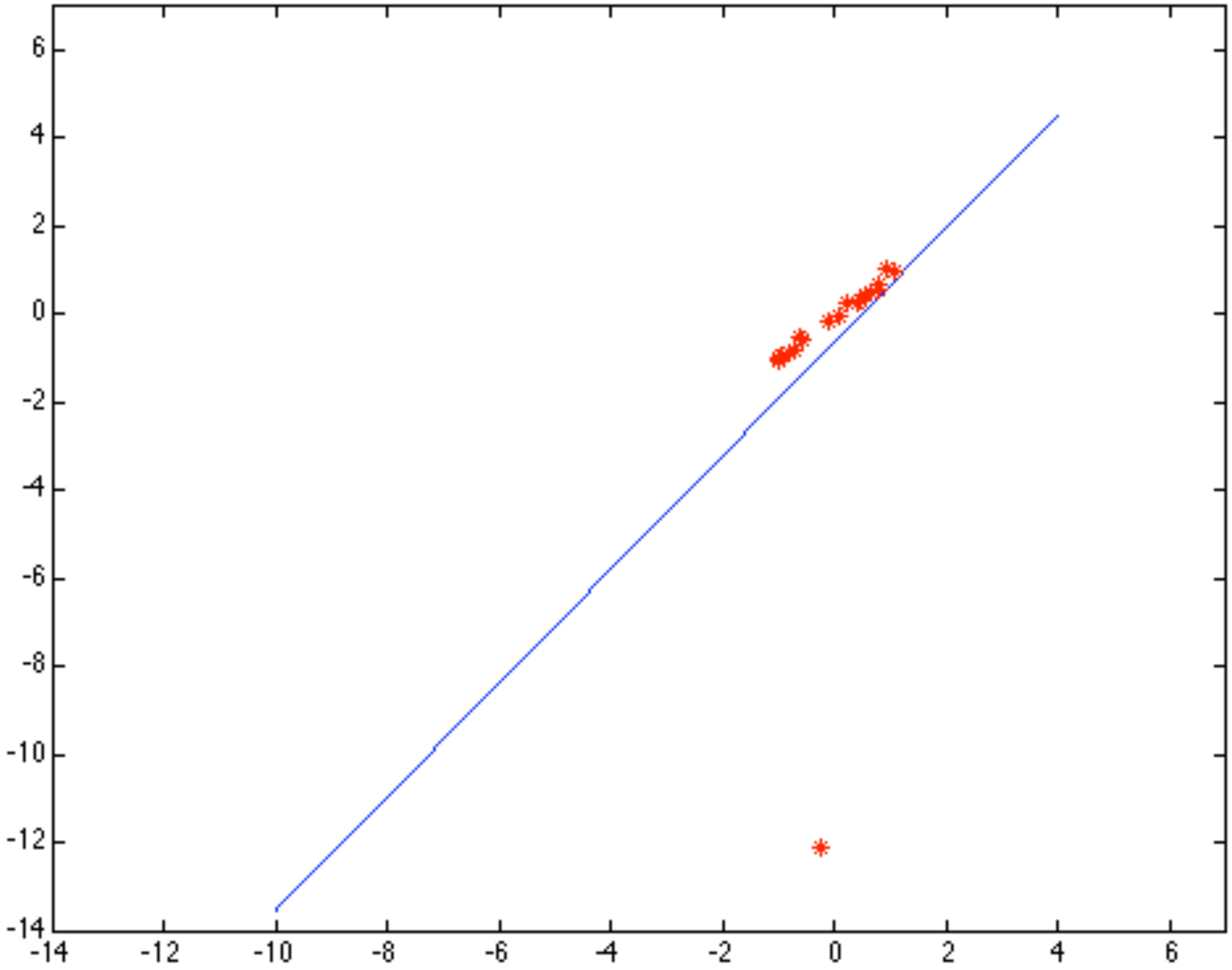
- As we have seen, squared error can be a source of bias in the presence of noise points
  - One fix is EM - we'll do this shortly
  - Another is an M-estimator
    - Square nearby, threshold far away
  - A third is RANSAC
    - Search for good points

Computer Vision

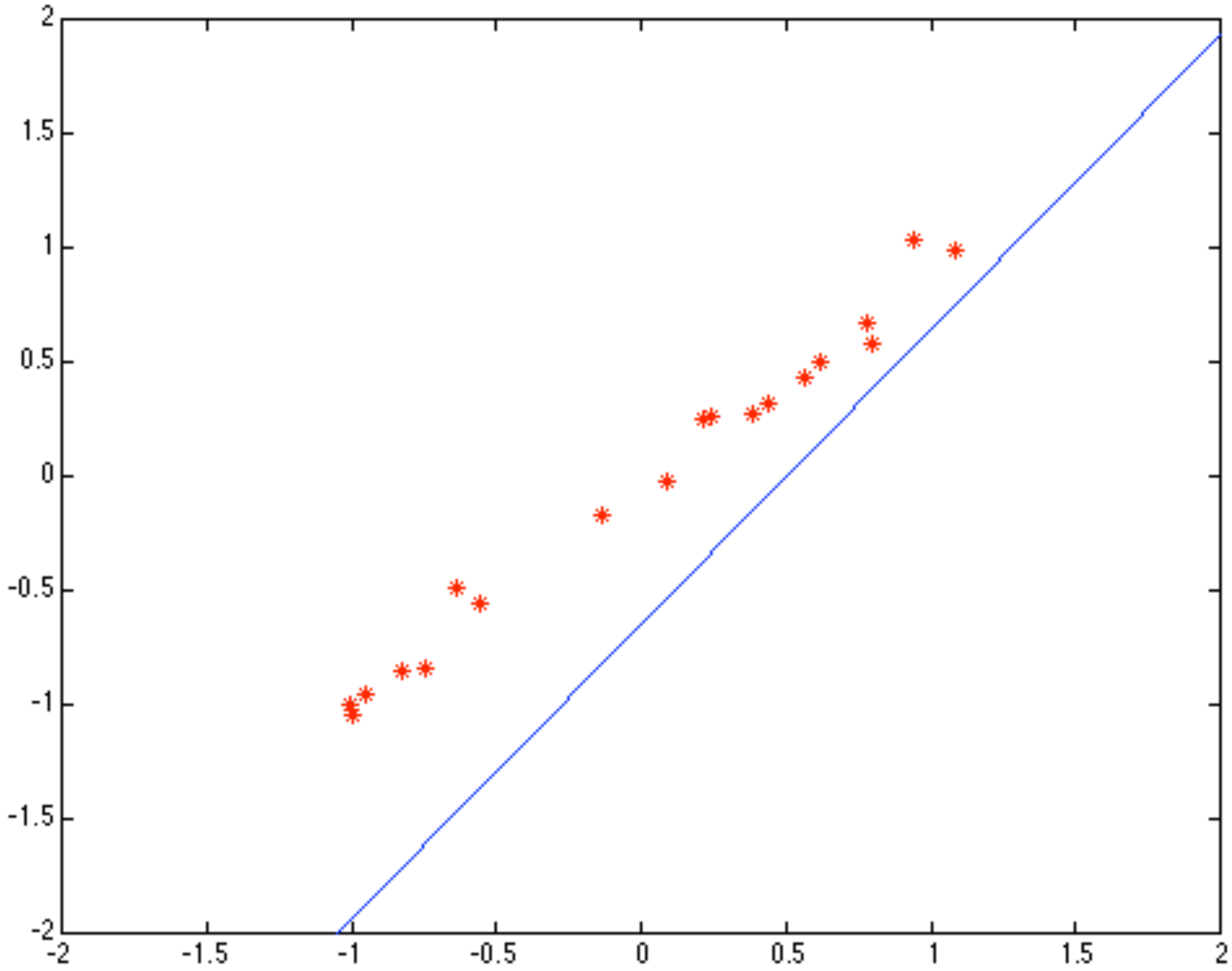



6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6
-4
-2
0
2
4
6

Computer Vision



Computer Vision



Computer Vision

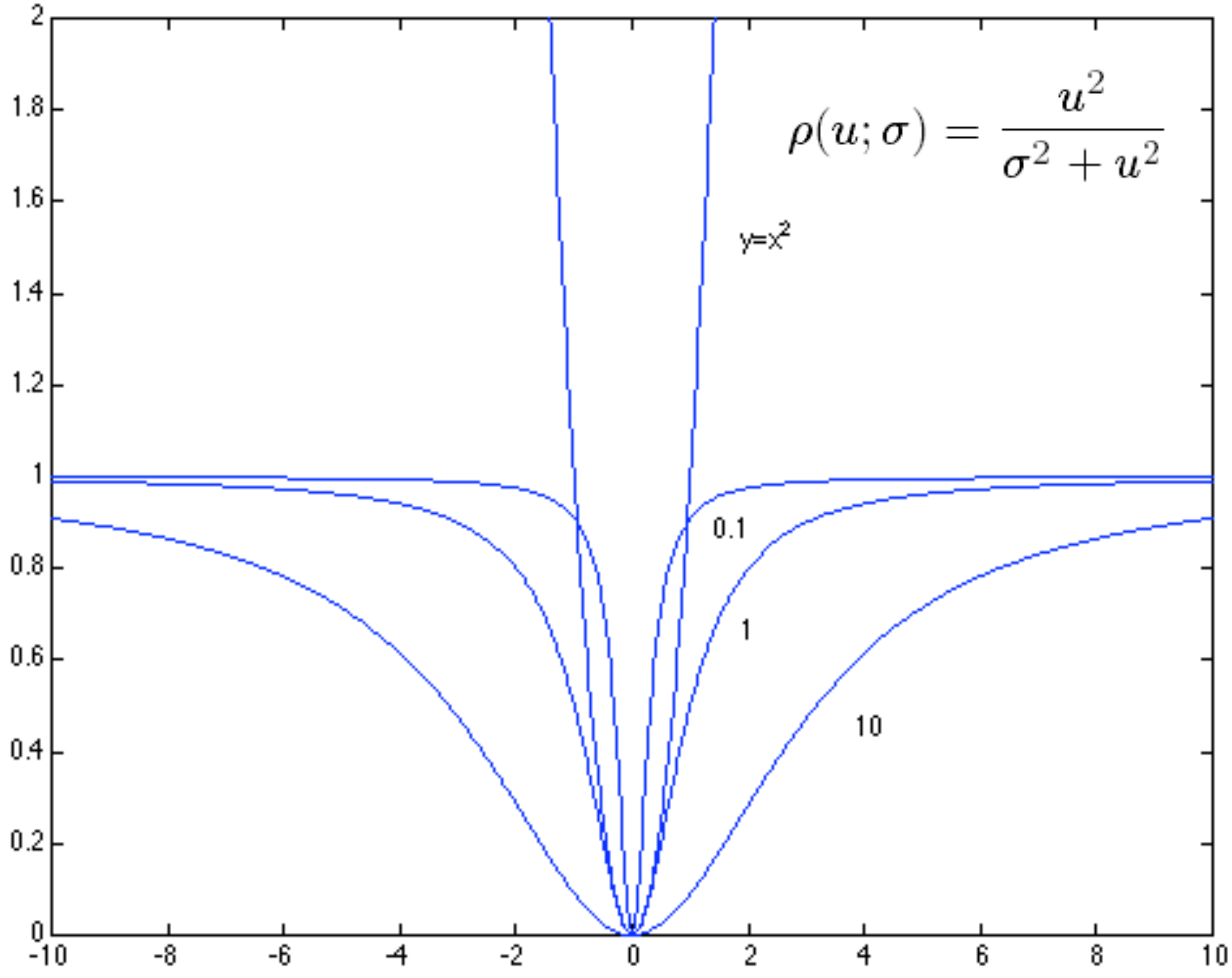
M-estimators

- Generally, minimize

$$\sum_i \rho\left(r_i\left(x_i,\theta\right);\sigma\right)$$

where $r_i\left(x_i,\theta\right)$ is the residual

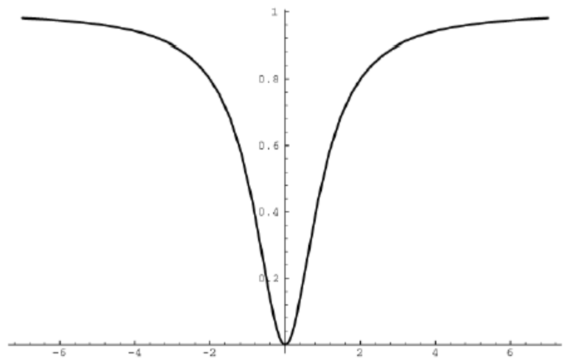
Computer Vision



Robust Estimation

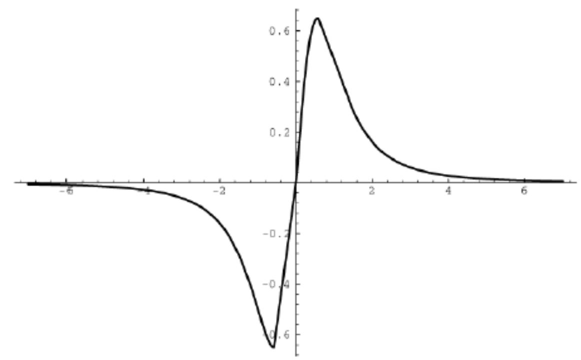
A quadratic ρ function gives too much weight to outliers

Instead, use robust norm:

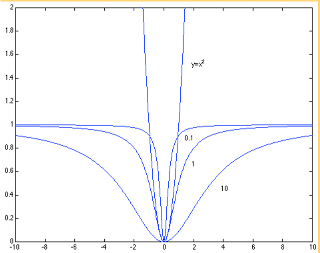


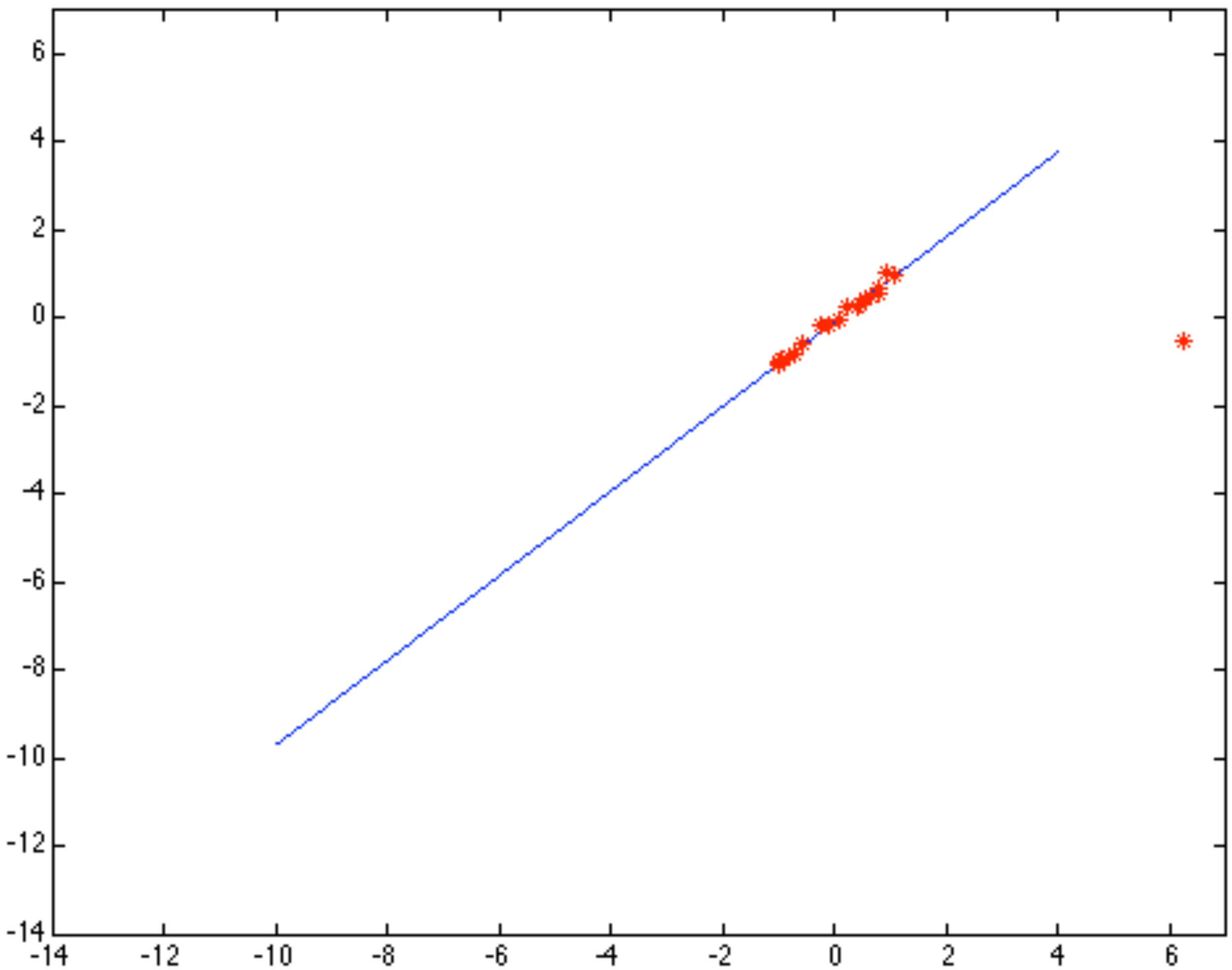
$$\rho(r,\sigma) = \frac{r^2}{\sigma^2 + r^2}$$

Influence function (d/dr of norm):

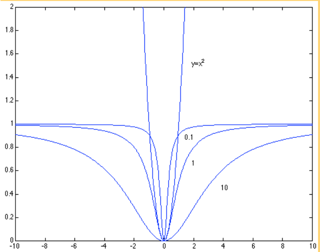


$$\psi(r,\sigma) = \frac{2r\sigma^2}{(\sigma^2 + r^2)^2}$$

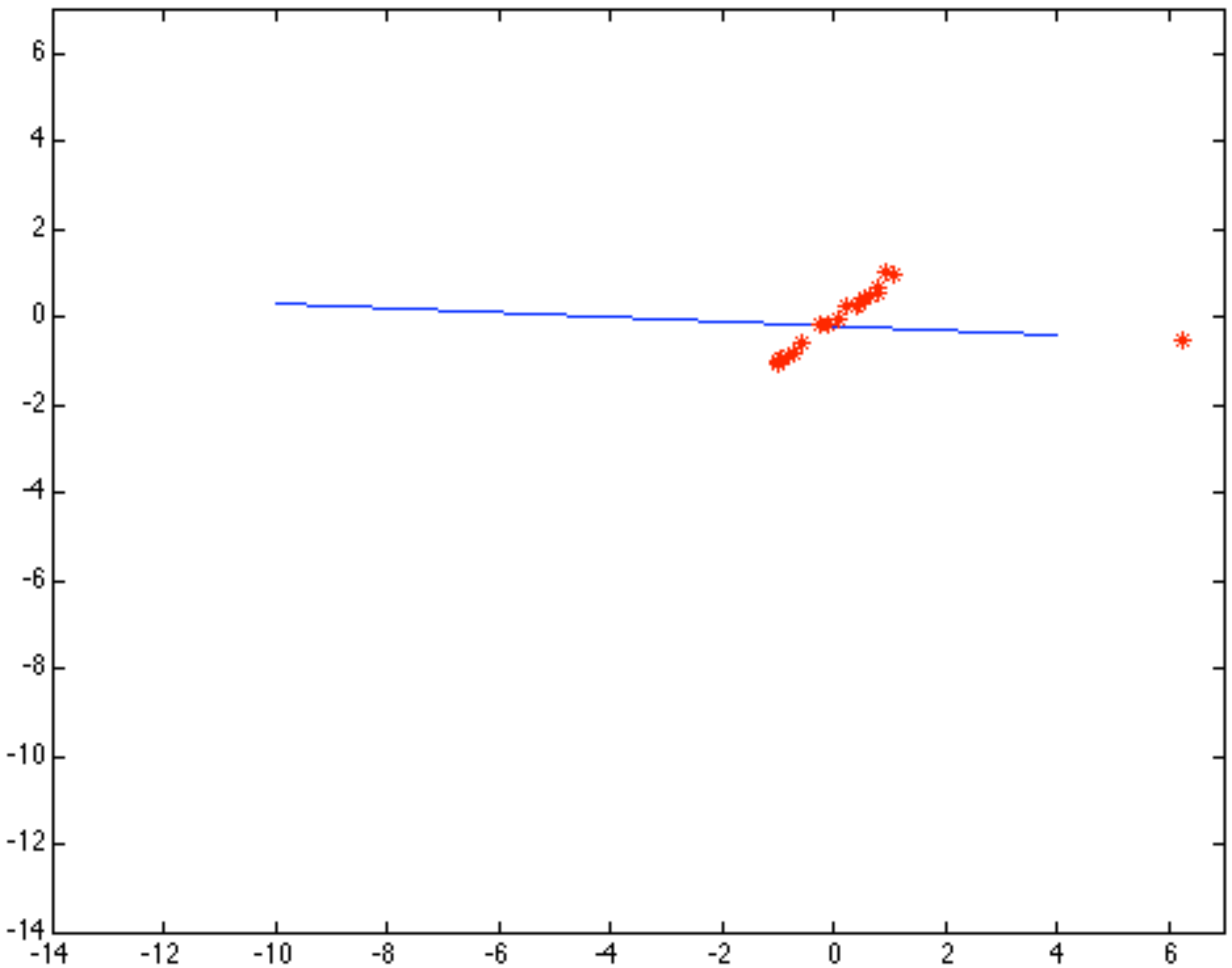




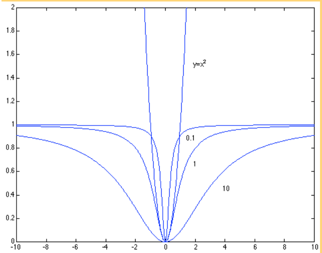
6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6
-4
-2
0
2
4
6



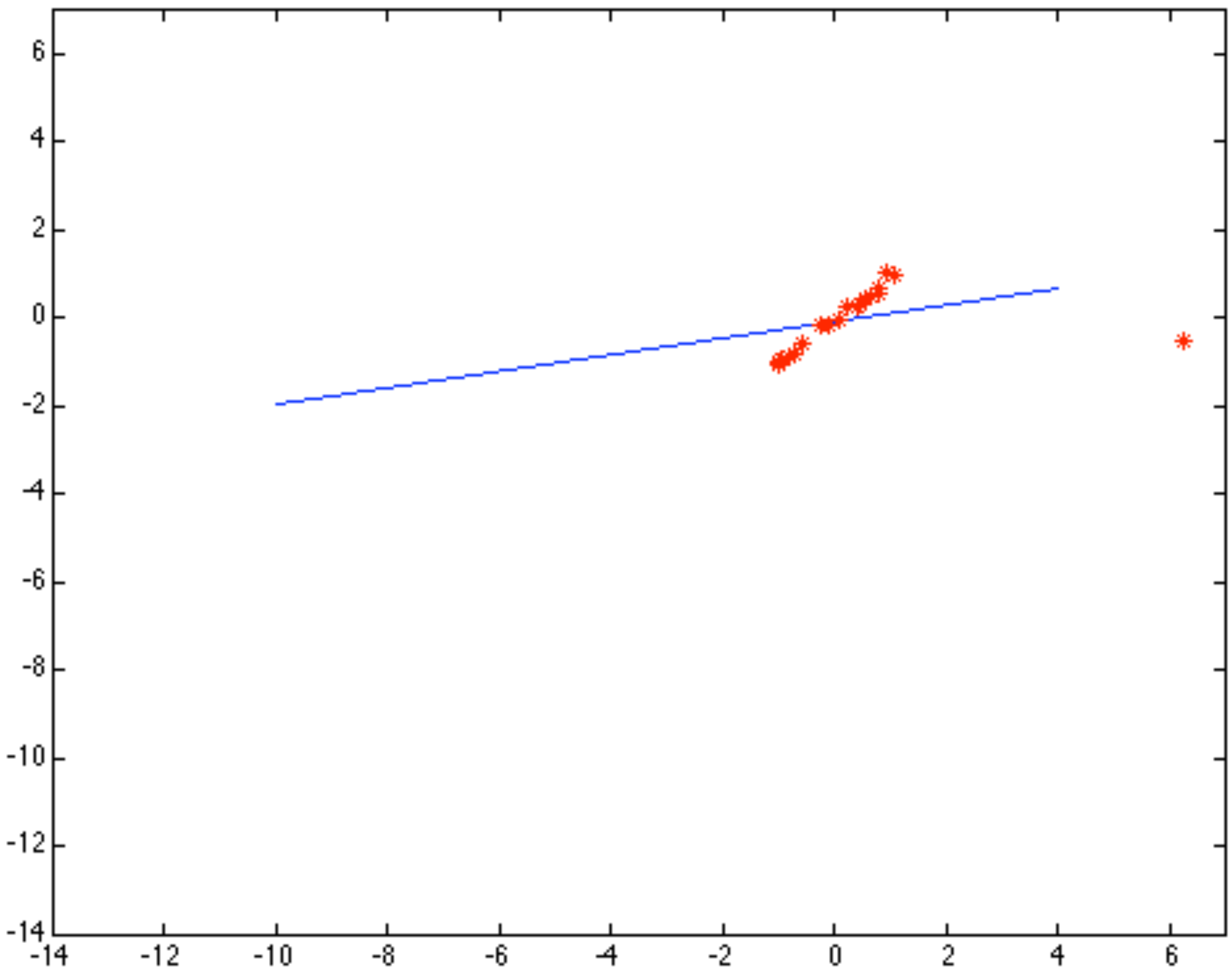
Too small



Computer Vision



Too large



Computer Vision

Robust scale

Scale is critical!

Popular choice:

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

RANSAC

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

- 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 it is unlikely to be all outliers

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

Distance threshold

Choose *t* so probability for inlier is α (e.g. 0.95)

- Often empirically
- Zero-mean Gaussian noise σ then $d_\perp^2$ follows χ_m^2 distribution with m=codimension of model

(dimension+codimension=dimension space)

Codimension	Model	t^2
1	line,F	$3.84\sigma^2$
2	H,P	$5.99\sigma^2$
3	T	$7.81\sigma^2$

Computer Vision

How many samples?

Choose N so that, with probability p, at least one random sample is free from outliers. e.g. p=0.99

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

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

	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

Acceptable consensus set?

- Typically, terminate when inlier ratio reaches expected ratio of inliers

$$T = (1 - e)n$$