Unit 6:

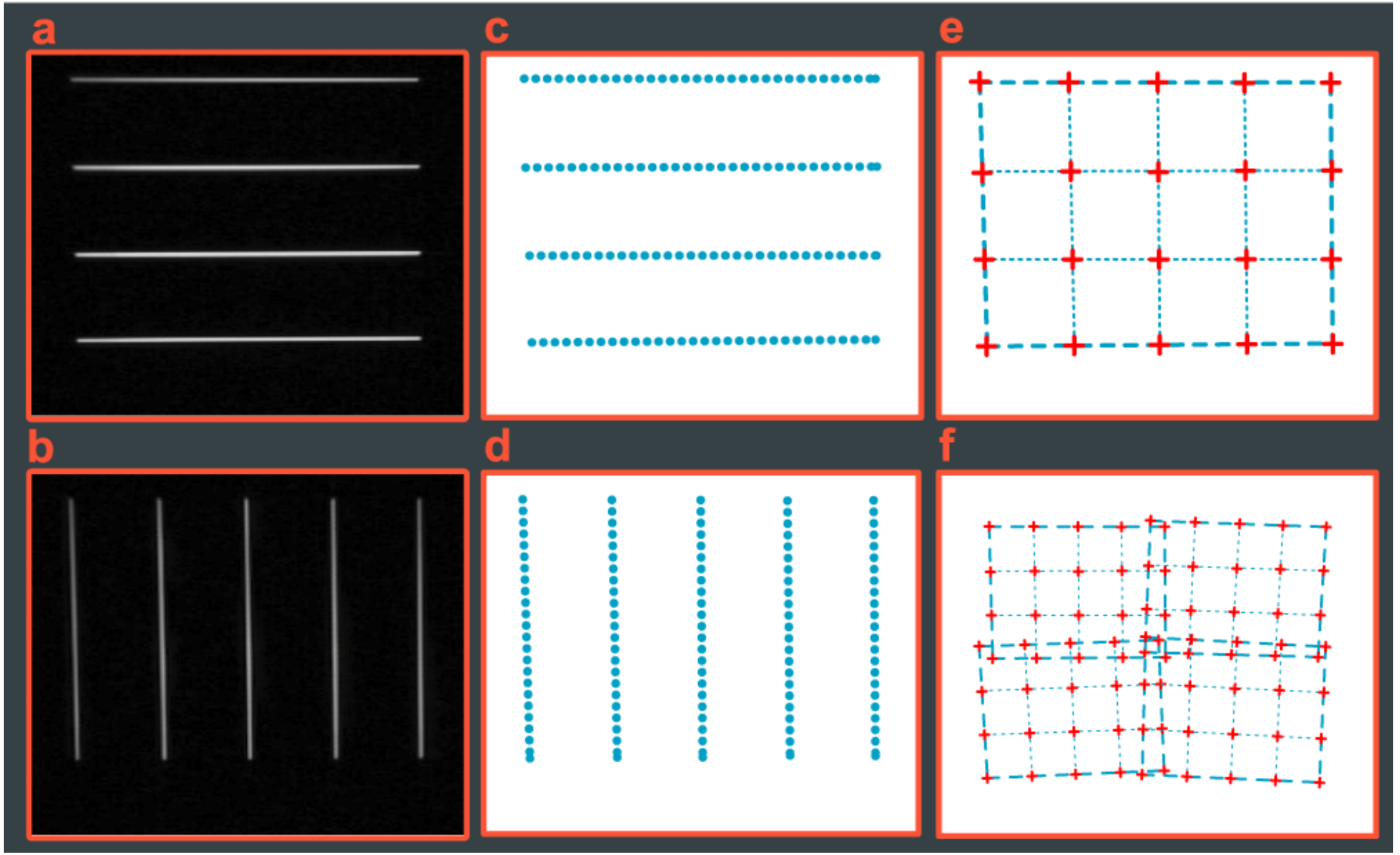
Model Fitting in CV

Shang-Hong Lai

Overview

- Motivation for model fitting
  - Finding geometric primitive shapes from images
  - Alignment of images
  - Camera calibration
  - Estimating fundamental matrix from point correspondences
- Hough transform
  - Fitting lines with the Hough transform
  - Sensitivity to noise & implementation concerns
- Model fitting from data with outliers
  - M-estimation
  - RANSAC

Line Fitting To Find Calibration Points



Fitting

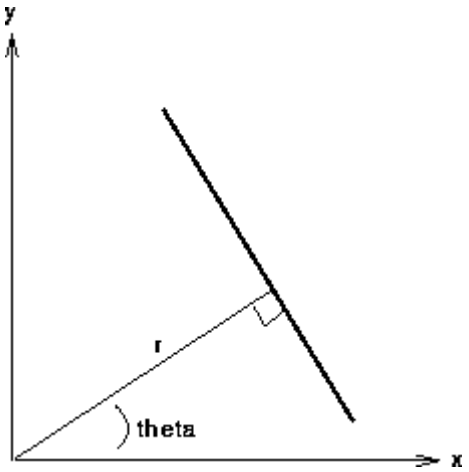
- Choose a parametric object/some objects to represent a set of tokens
- Most interesting case is when criterion is not local
  - can't tell whether a set of points lies on a line by looking only at each point and the next.

- 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...)

Fitting and the Hough Transform

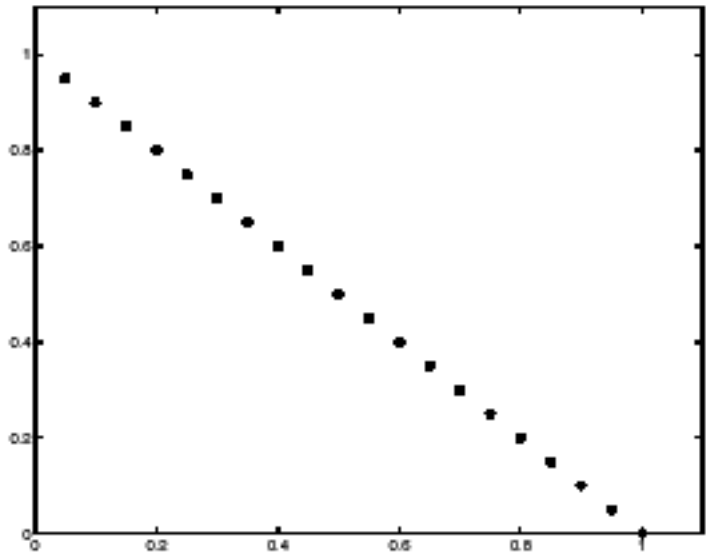
- Purports to answer all three questions
  - in practice, a good result requires good input.
- We explain for lines first
- One representation: a line is the set of points (x, y) such that:
  $(\cos \theta) X + (\sin \theta) Y + r = 0$

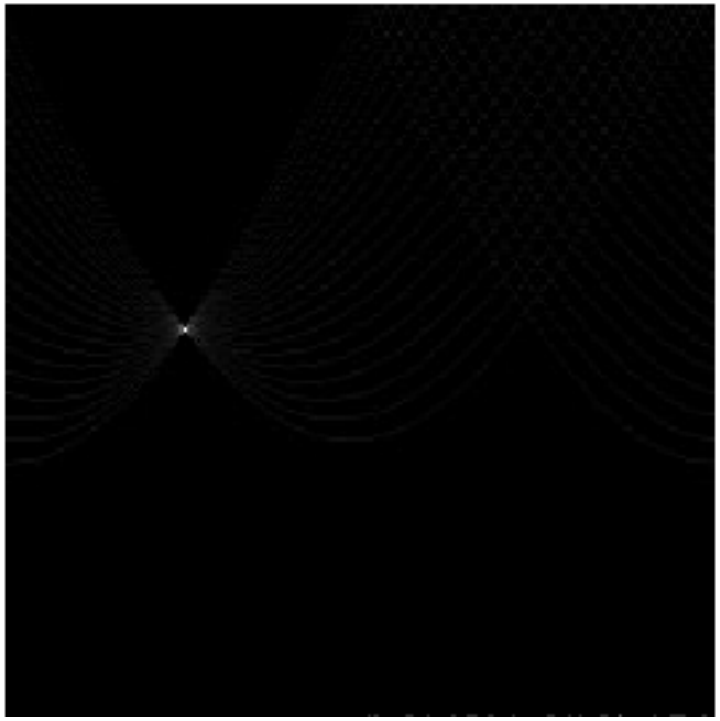


- Different choices of θ, $r>0$ give different lines
- For any token (x, y) there is a one parameter family of lines through this point, given by $(\cos \theta) X + (\sin \theta) Y + r = 0$
- Each point gets to vote for each line in the family; if there is a line that has lots of votes, that should be the line passing through the points

Votes

Tokens





r: 0 to 1.55

Theta = 45º = 0.785 rad
r = (1√2) / 2 = 0.707

Theta: 0 to 3.14 (rad)

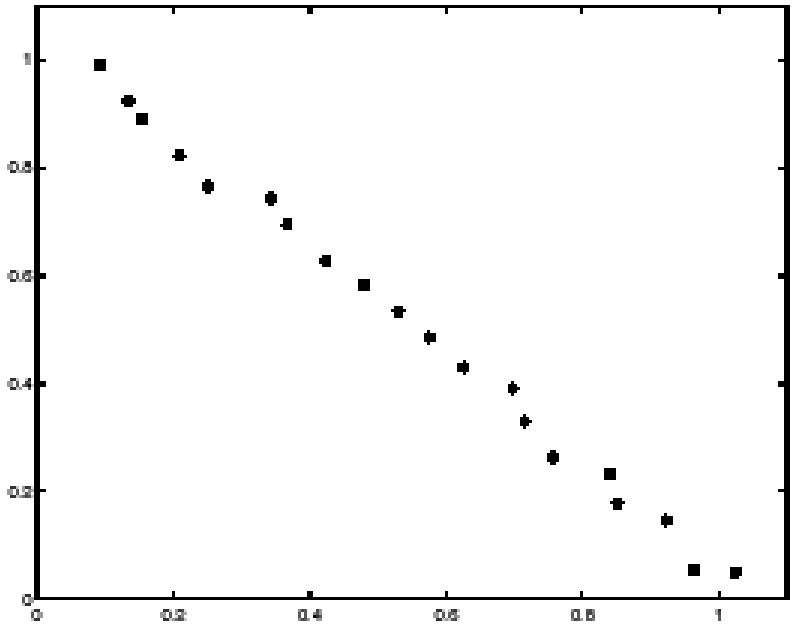
Brightest point = 20 votes

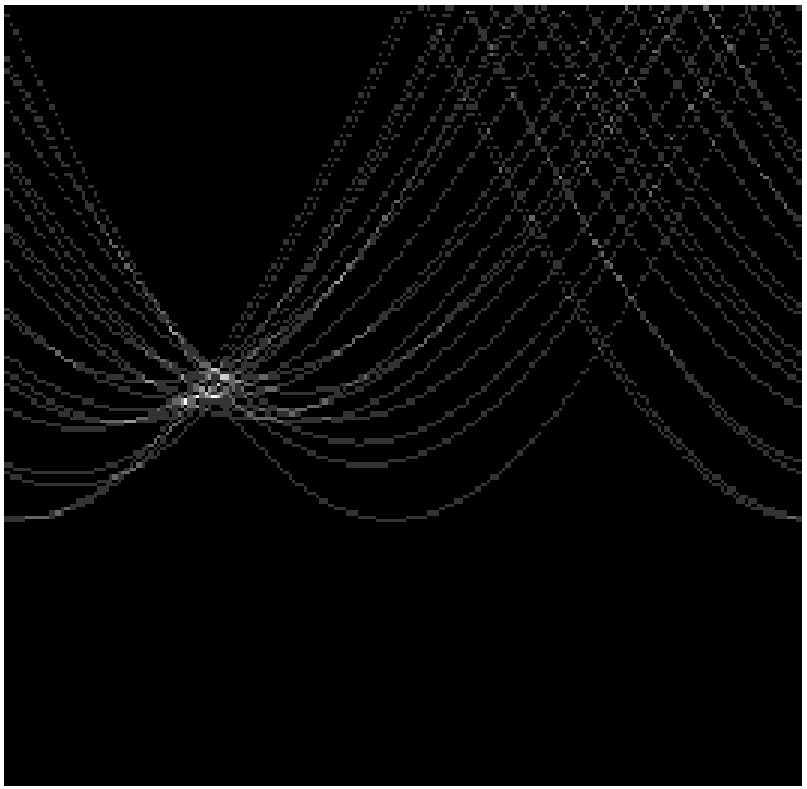
Mechanics of the Hough transform

- Construct a voting array representing θ, r
- For each point, render the curve (θ, r) 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

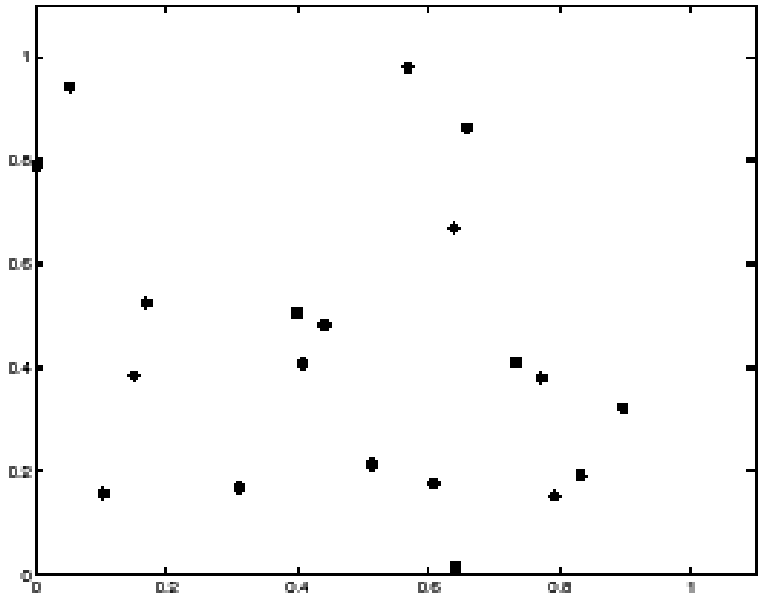
Votes

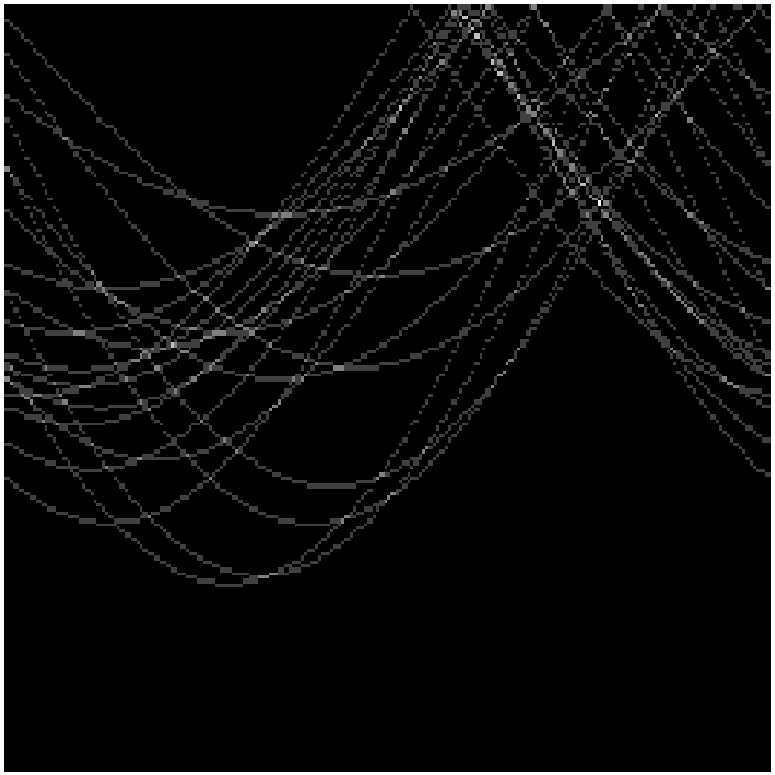
Tokens



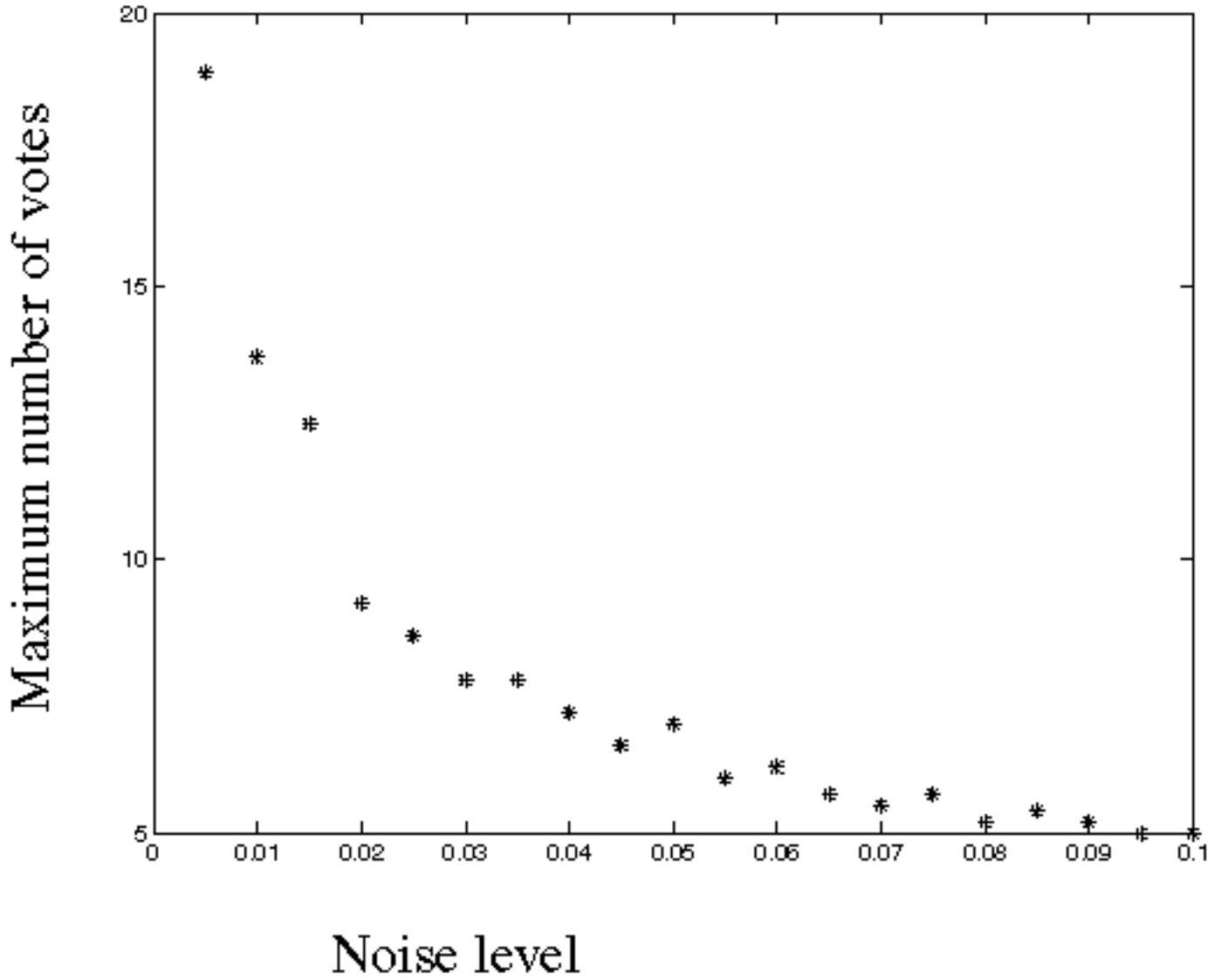


Brightest point = 6 votes

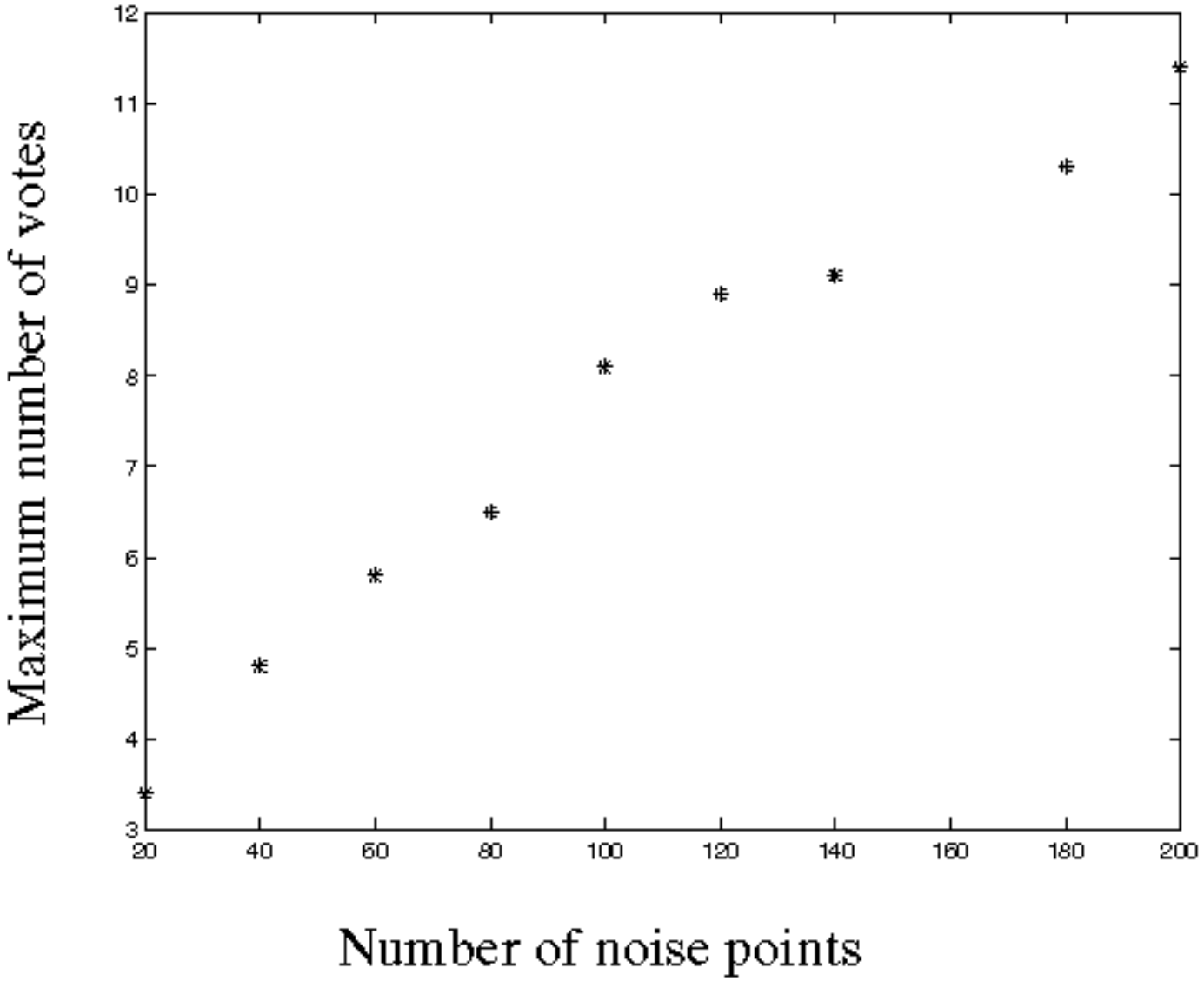

0
0.2
0.4
0.6
0.8
1
0
0.2
0.4
0.6
0.8
1



Noise Lowers the Peaks



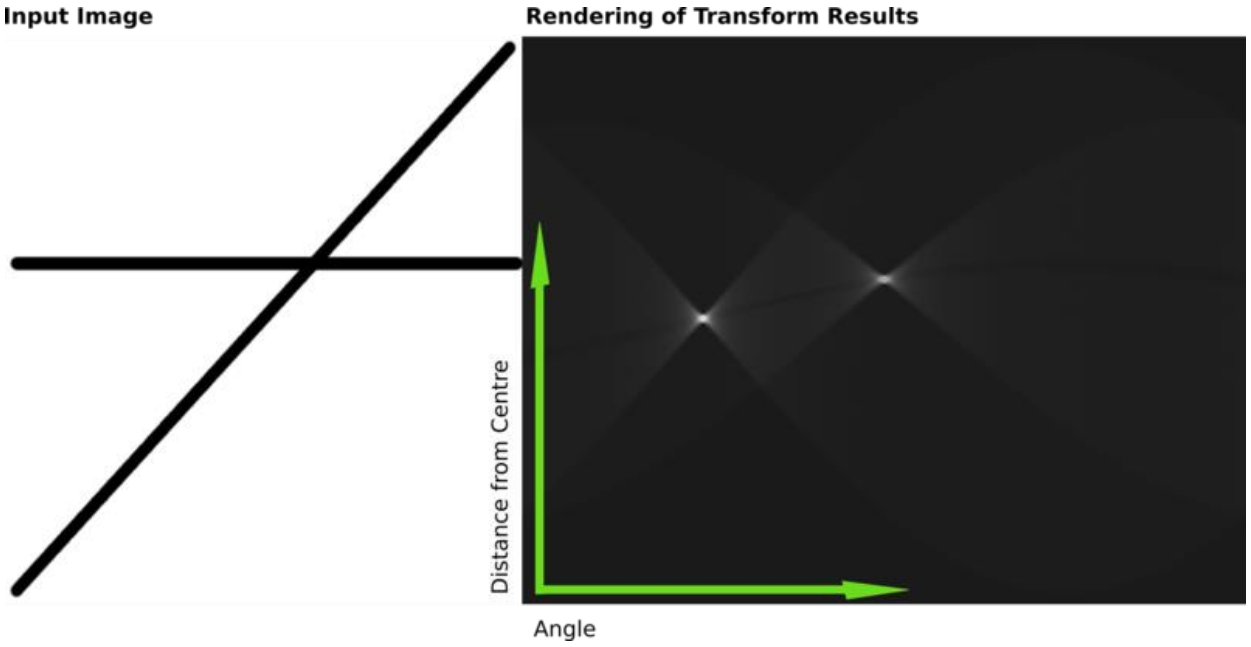
Noise Increases the Votes in Spurious Accumulator Elements



Optimization to the Hough Transform

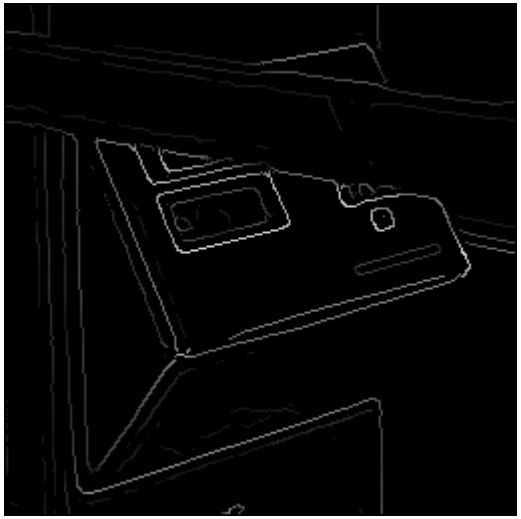
Noise: If the orientation of tokens (pixels) is known, only accumulator elements for lines with that general orientation are voted on. (Most edge detectors give orientation information.)

Speed: The accumulator array can be coarse, then repeated in areas of interest at a finer scale.

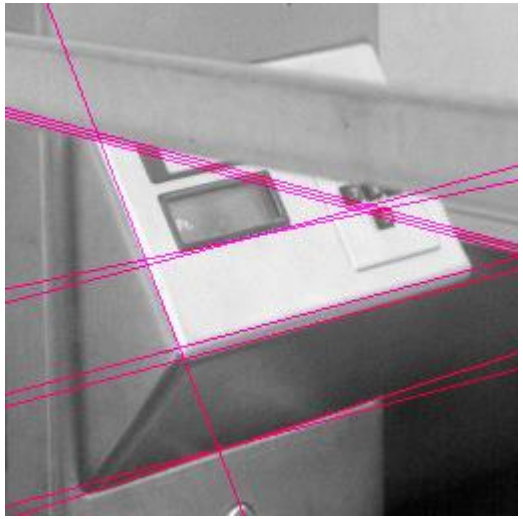


Real World Example

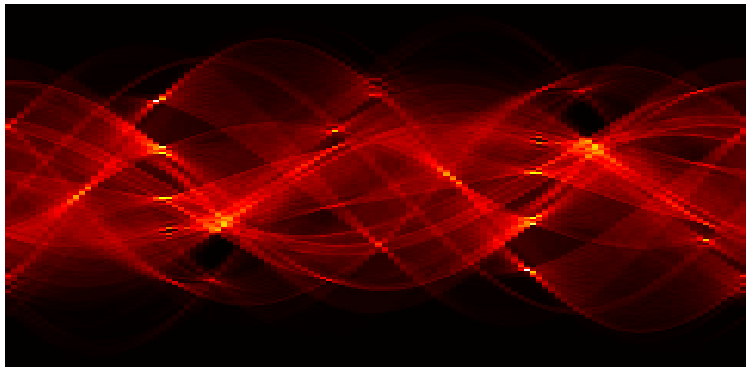
Original



Edge Detection



Found Lines



Hough Space

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

Least Square Line Fitting

we want to choose the line that minimizes

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

By differentiation, the line is given by the solution to the problem

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

To minimize the sum of perpendicular distances between points and lines, we need to minimize

$$\sum (ax_i + by_i + c)^2,$$

subject to $a^2 + b^2 = 1$. Now using a Lagrange multiplier λ, we have a solution if

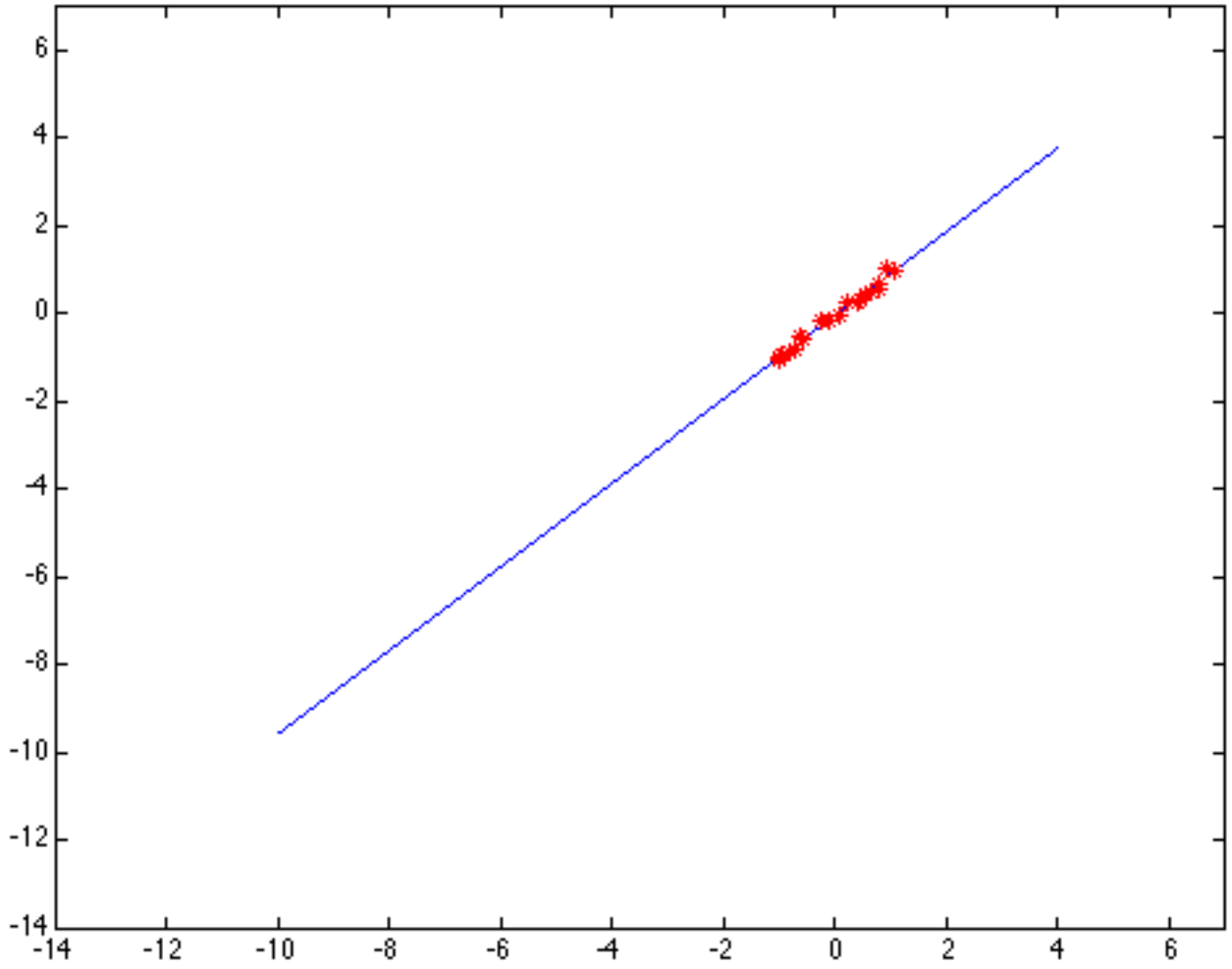
$$\begin{pmatrix} \overline{x^2} & \overline{xy} & \overline{x} \\ \overline{xy} & \overline{y^2} & \overline{y} \\ \overline{x} & \overline{y} & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \lambda \begin{pmatrix} 2a \\ 2b \\ 0 \end{pmatrix}.$$

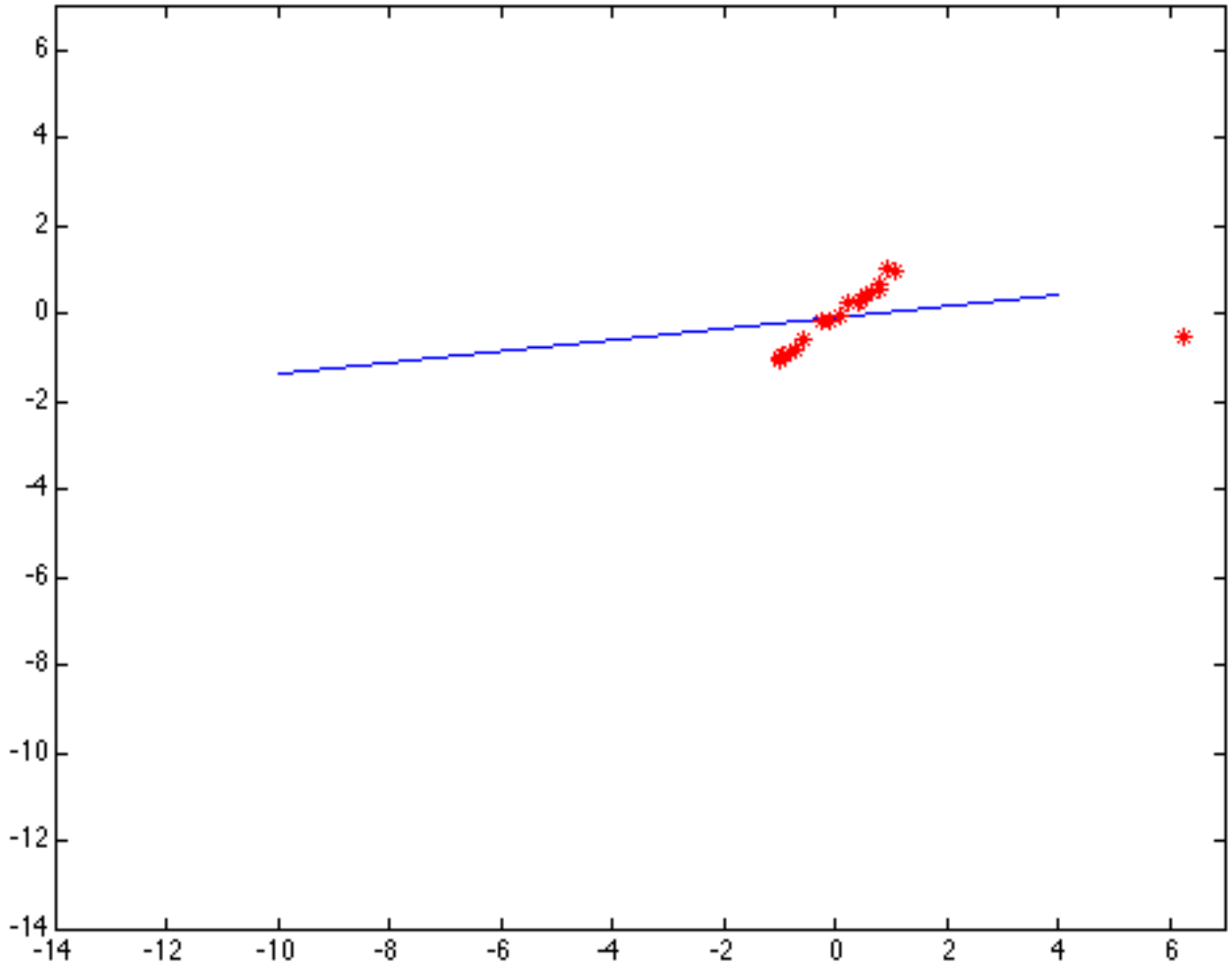
Algorithm 15.1: Incremental line fitting by walking along a curve, fitting a line to runs of pixels along the curve, and breaking the curve when the residual is too large

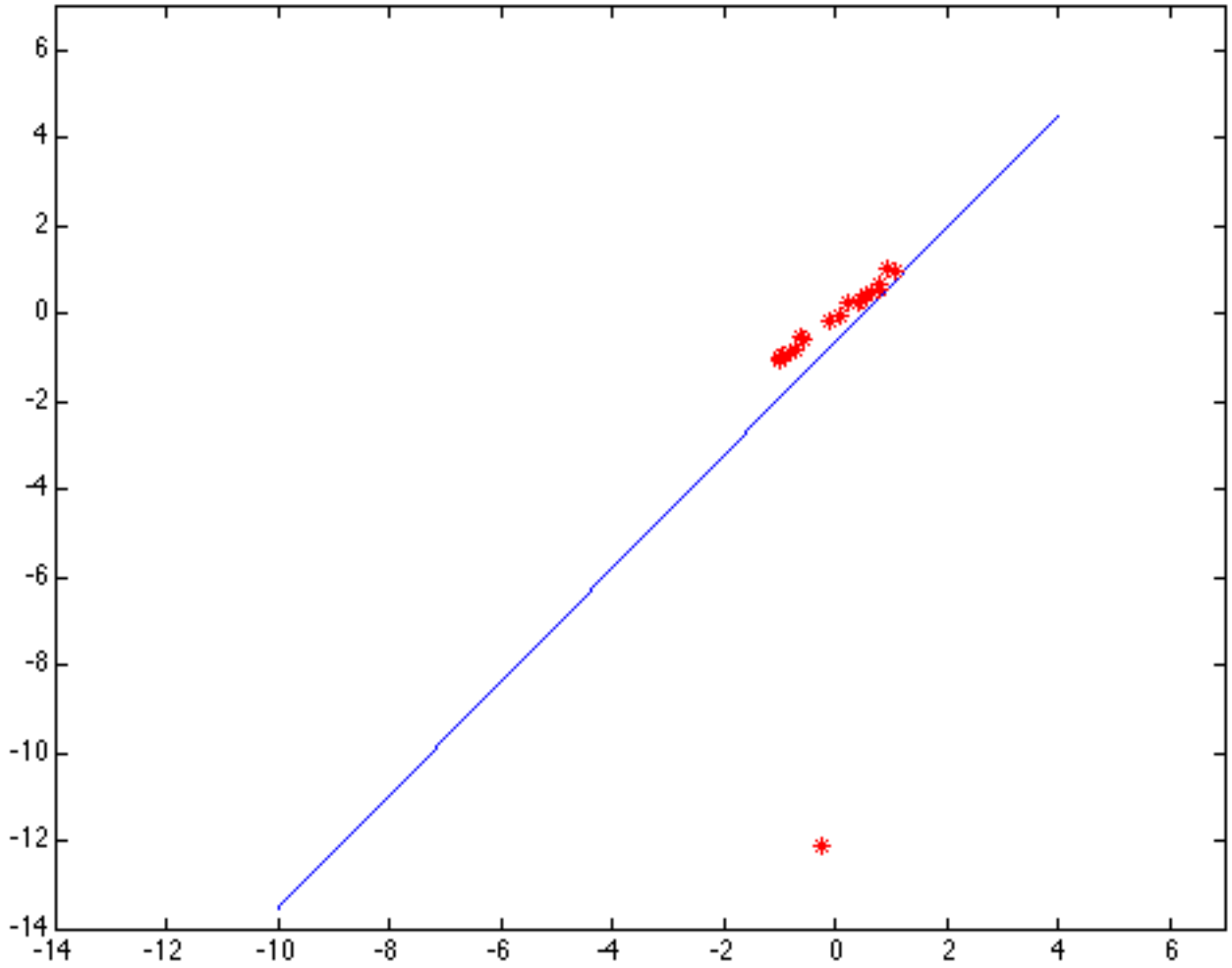
```
Put all points on curve list, in order along the curve
Empty the line point list
Empty the line list
Until there are too few points on the curve
   Transfer first few points on the curve to the line point list
   Fit line to line point list
   While fitted line is good enough
      Transfer the next 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
```

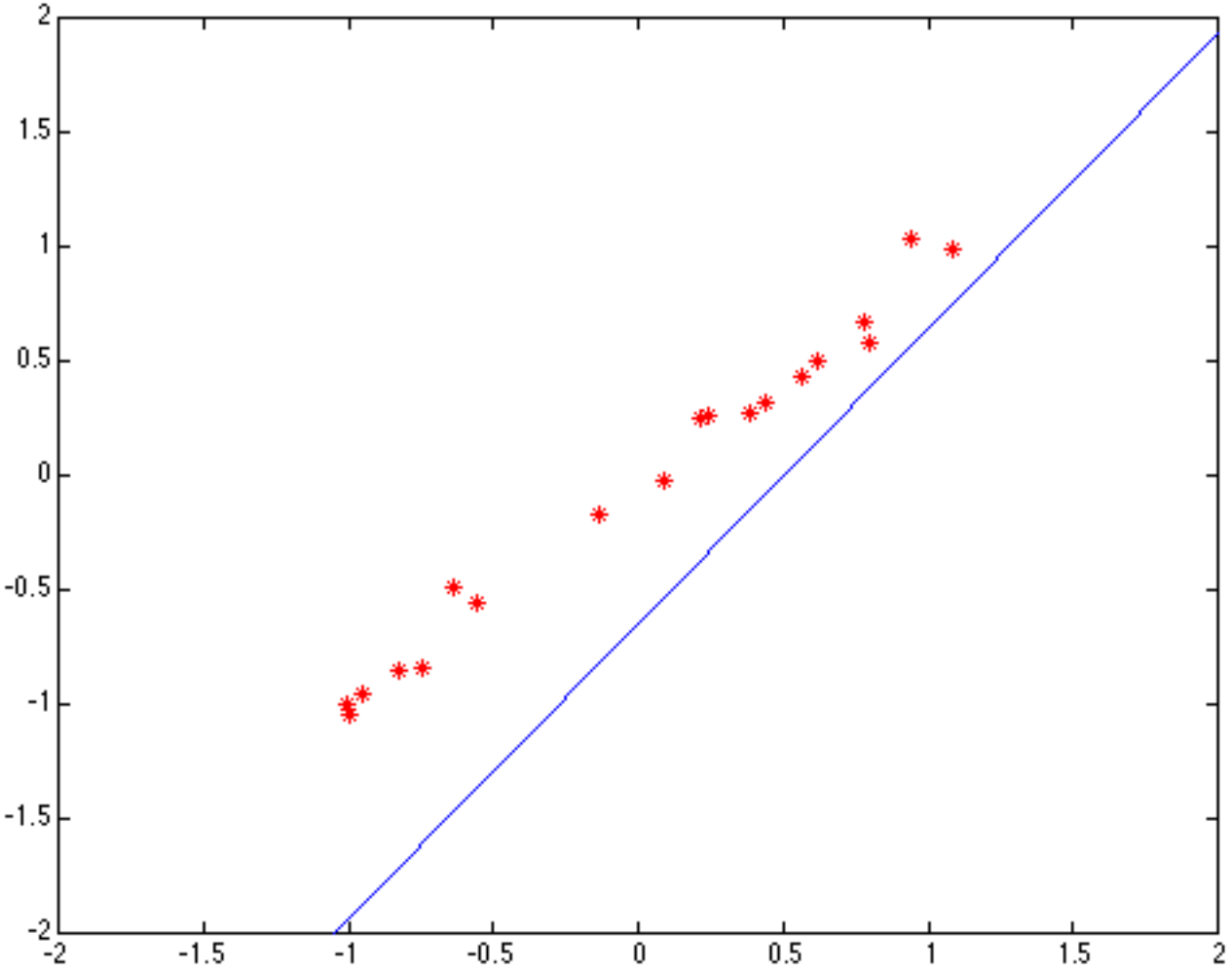
Robust Model Fitting

- Least square estimation can be a source of bias in the presence of high-level noises or outliers
- Robust estimation
  - M-estimator
    - trimmed least square estimation
      - Square nearby, threshold far away
  - RANSAC
    - Search for good points


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

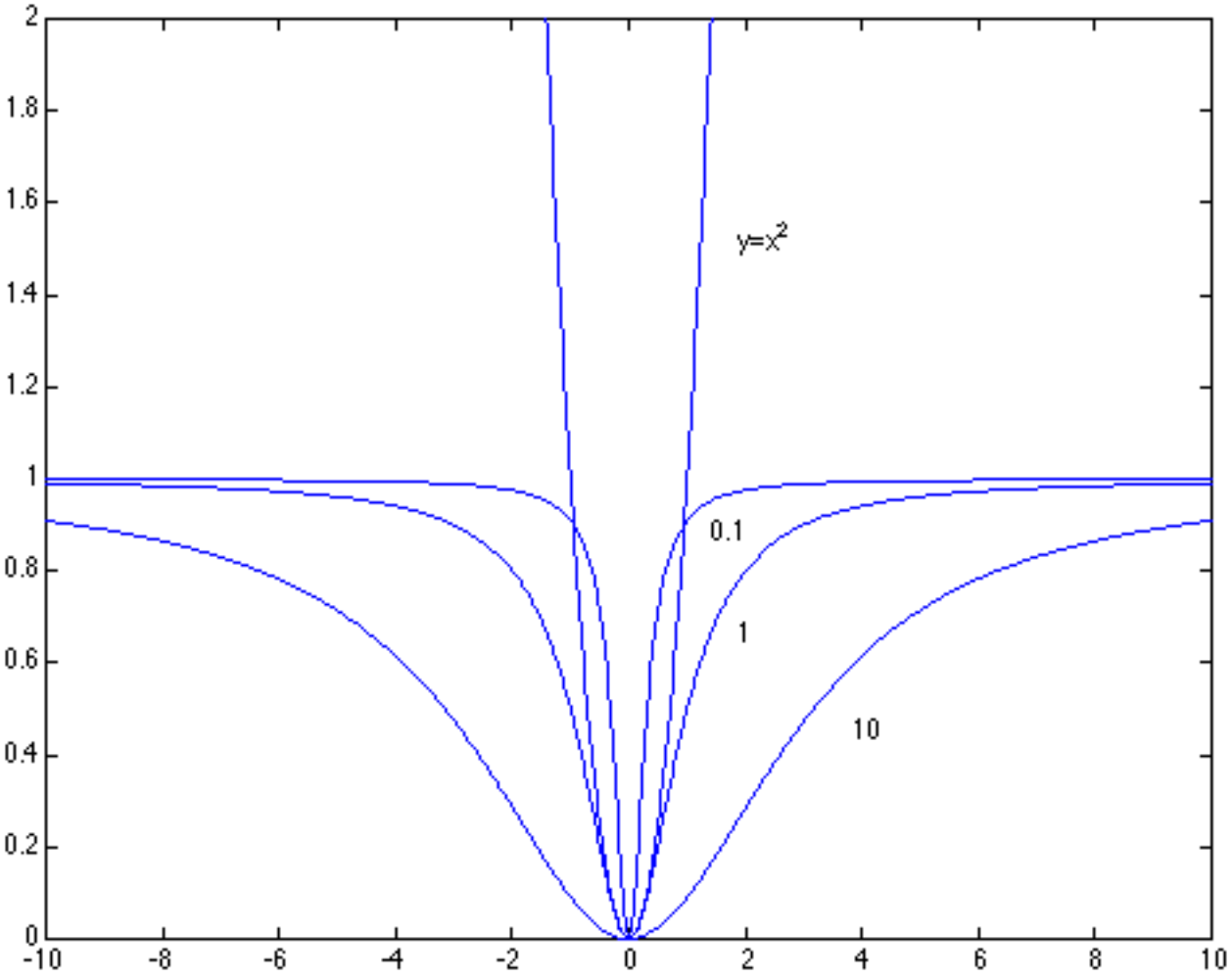

6
4
2
0
-2
-4
-6
-8
-10
-12
-14
-14
-12
-10
-8
-6
-4
-2
0
2
4
6


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

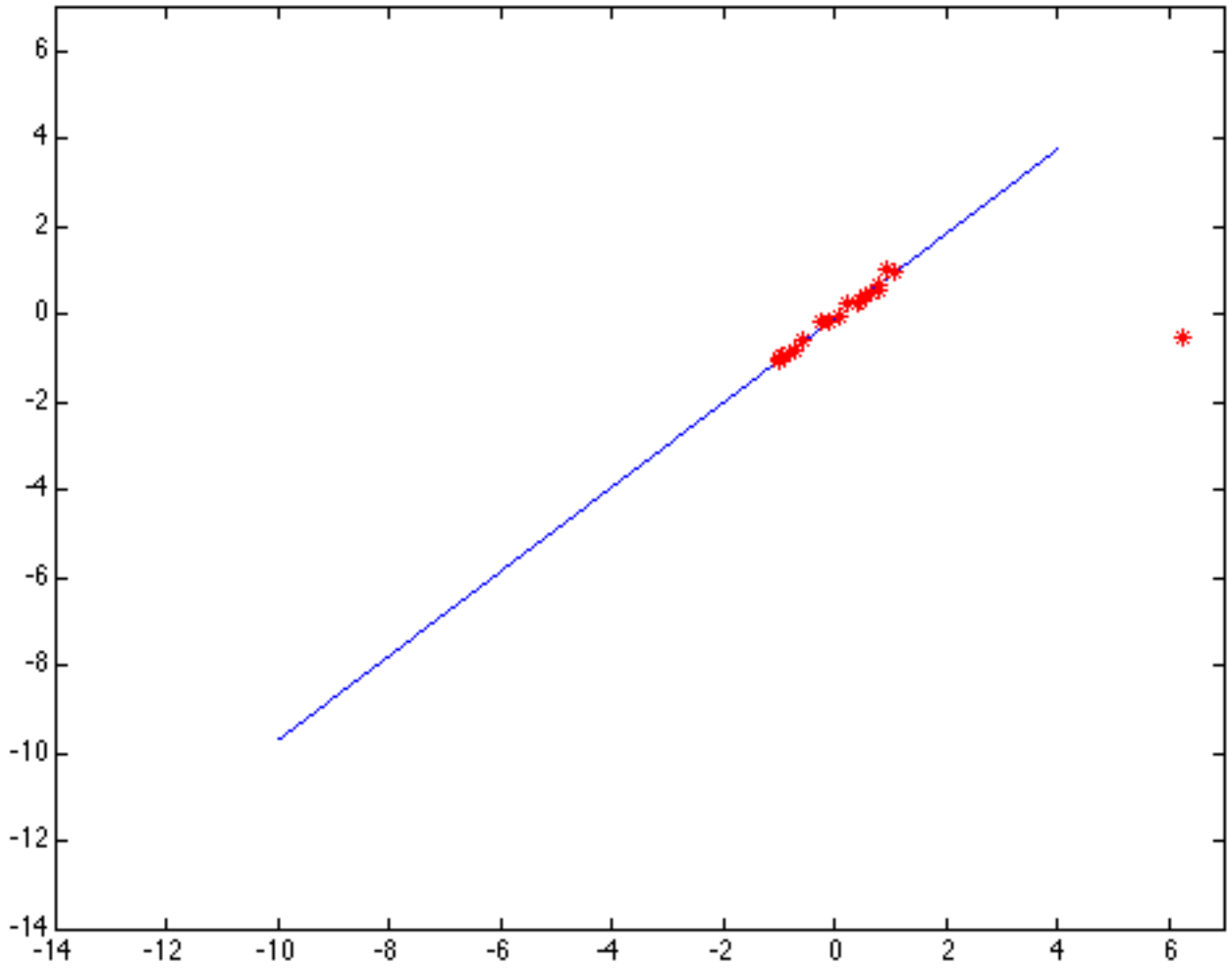

2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

LS Error Function vs. Robust Error Function

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

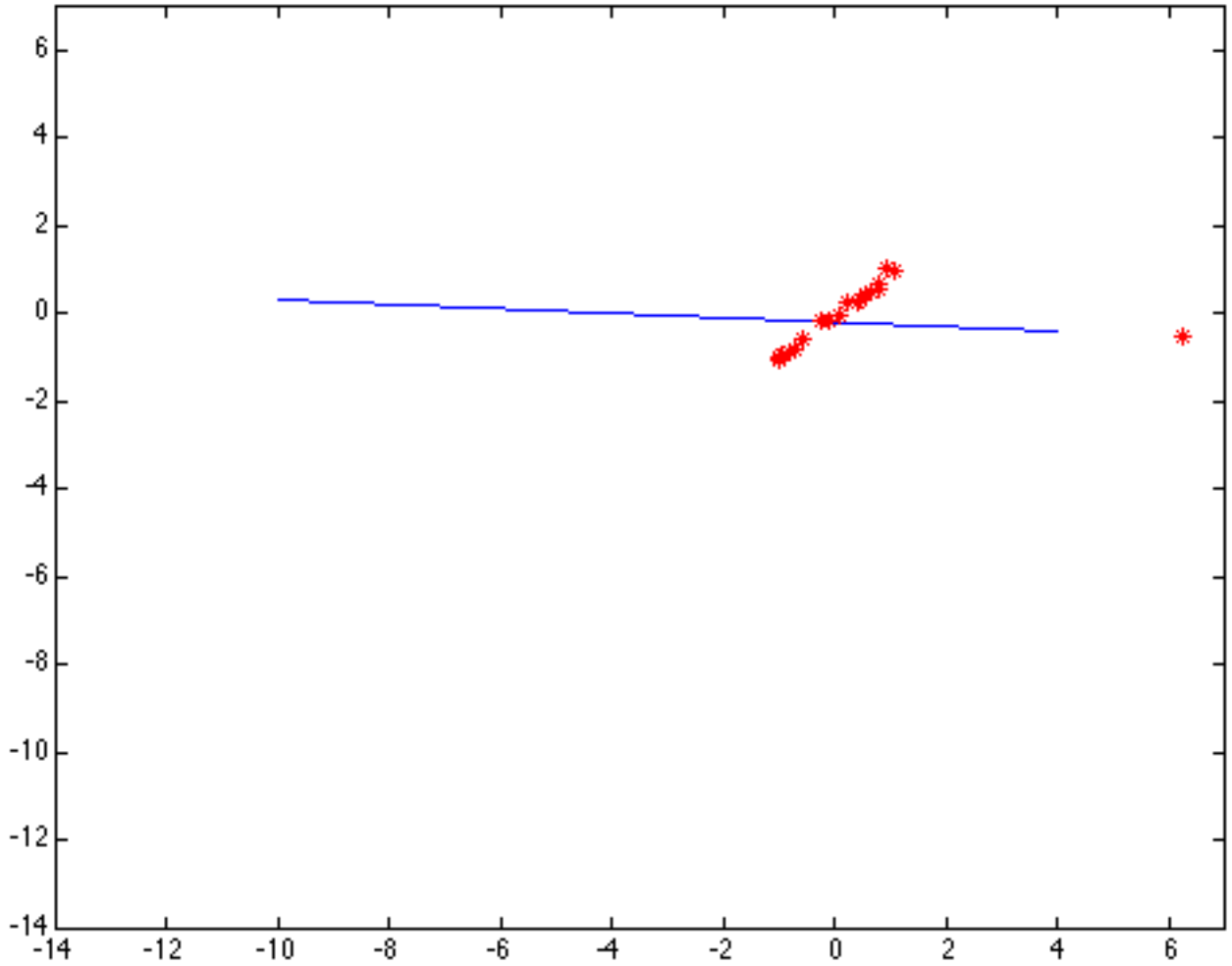


Line Fitting with Robust Error Function



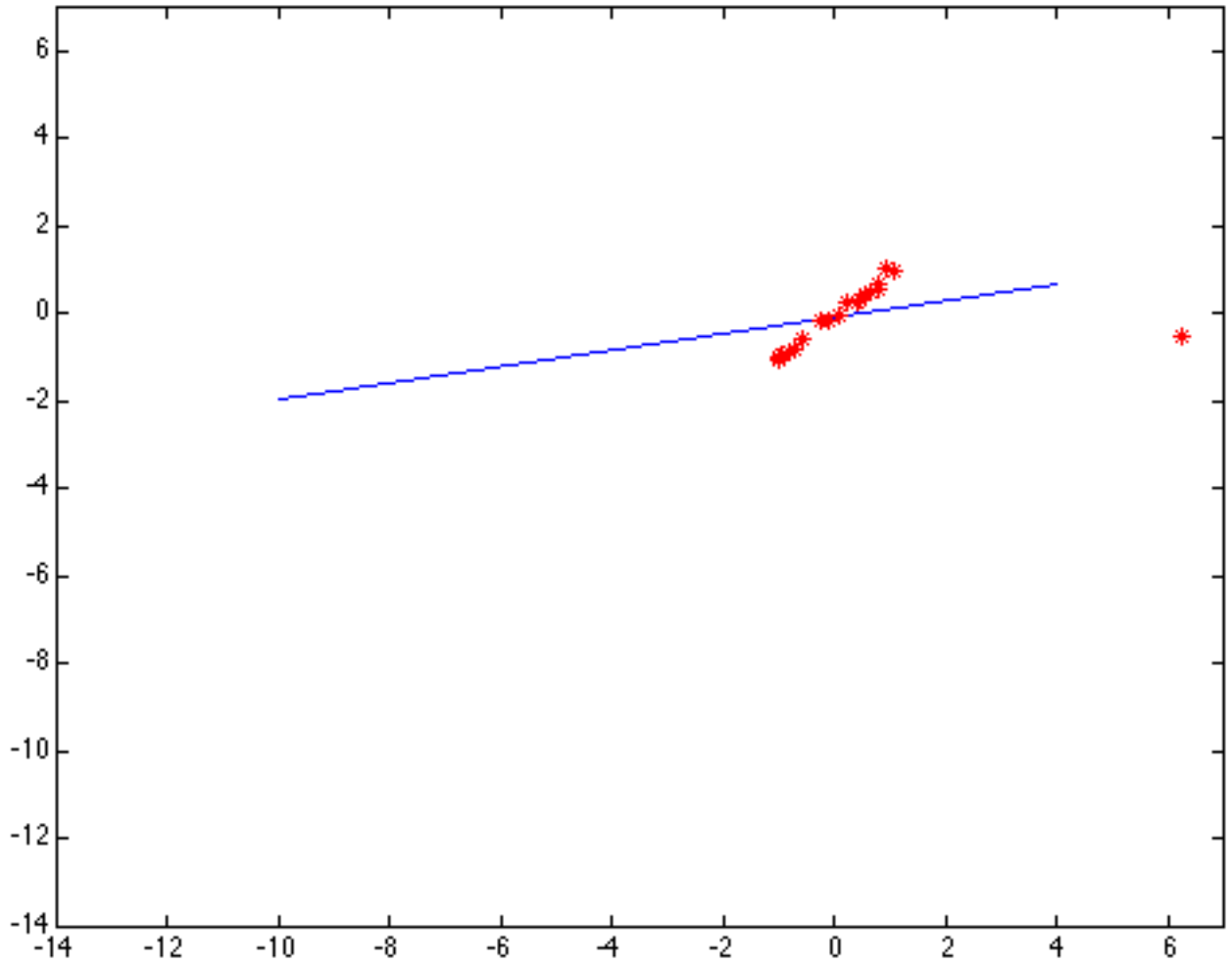
Line Fitting with Robust Error Function

Too small scale parameter σ



Line Fitting with Robust Error Function

Too large scale parameter σ



M-Estimator

An **M-estimator** estimates parameters by minimizing an expression of the form

$$\sum_i \rho(r_i(\boldsymbol{x}_i, \theta); \sigma)$$

where θ are the parameters of the model being fitted and $r_i(\boldsymbol{x}_i, \theta)$ is the residual

$$\sum_i \rho(r_i(\boldsymbol{x}_i, \theta); \sigma) \frac{\partial \rho}{\partial \theta} = 0$$

Solve the equation by **iteratively re-weighted least squares** estimation.

M-Estimator

For $s = 1$ to $s = k$
  draw a subset of r distinct points, chosen uniformly at random

  Fit to this set of points using maximum likelihood
  (usually least squares) to obtain θ_s^0

  estimate σ_s^0 using θ_s^0

  Until convergence (usually $|\theta_s^n - \theta_s^{n-1}|$ is small):
    take a minimising step using θ_s^{n-1}, σ_s^{n-1}
    to get θ_s^n
    now compute σ_s^n

report the best fit of this set, using the median of the
residuals as a criterion

Algorithm 17.3: *Using an M-estimator to fit a probabilistic model*

RANSAC

Random Sample Consensus

- 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

RANSAC

- RANdom Sample Consensus
- Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use those only.
- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

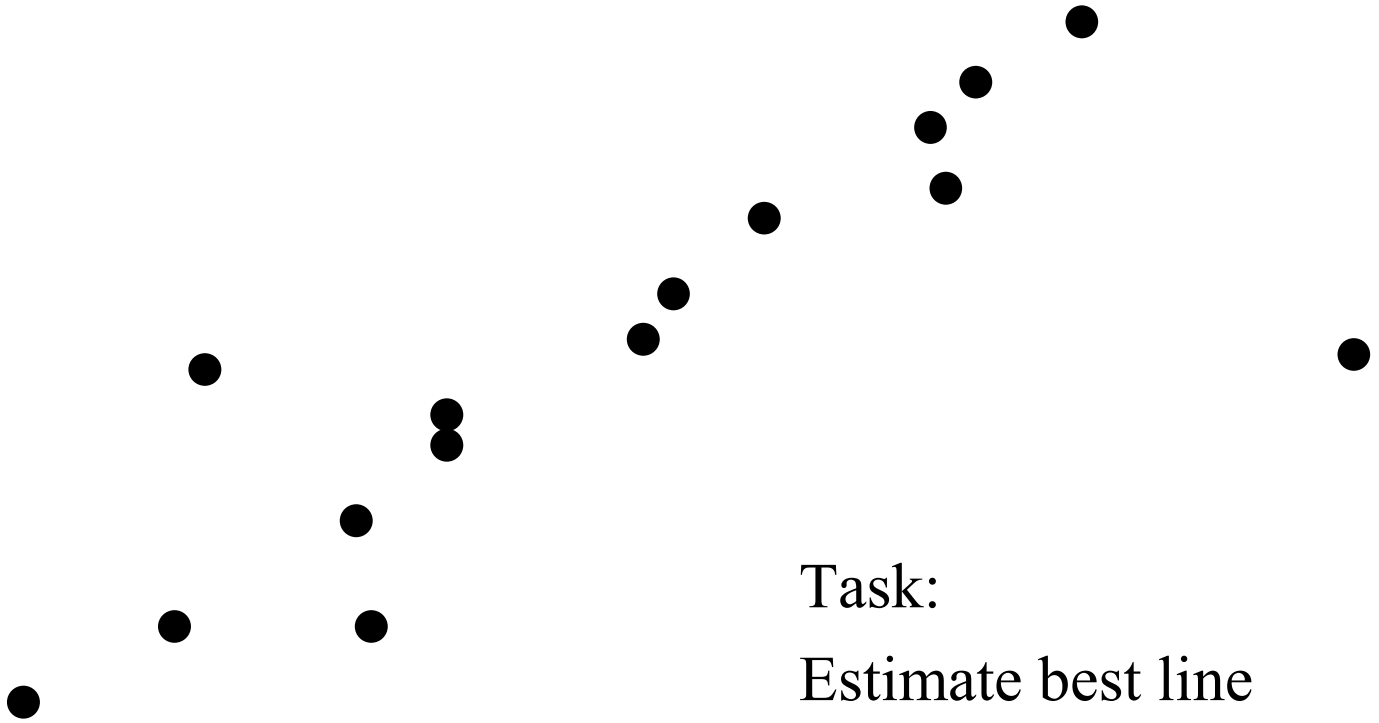
RANSAC

- RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

RANSAC Line Fitting Example

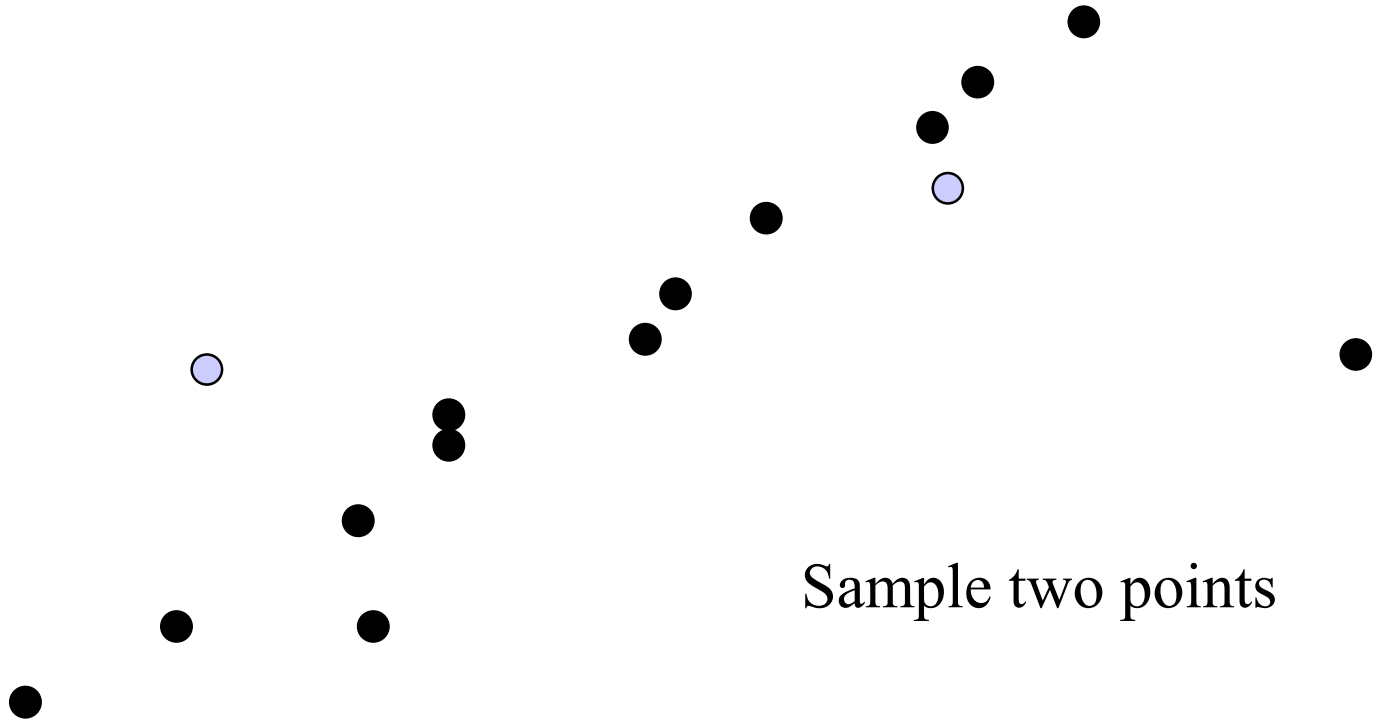


Task:

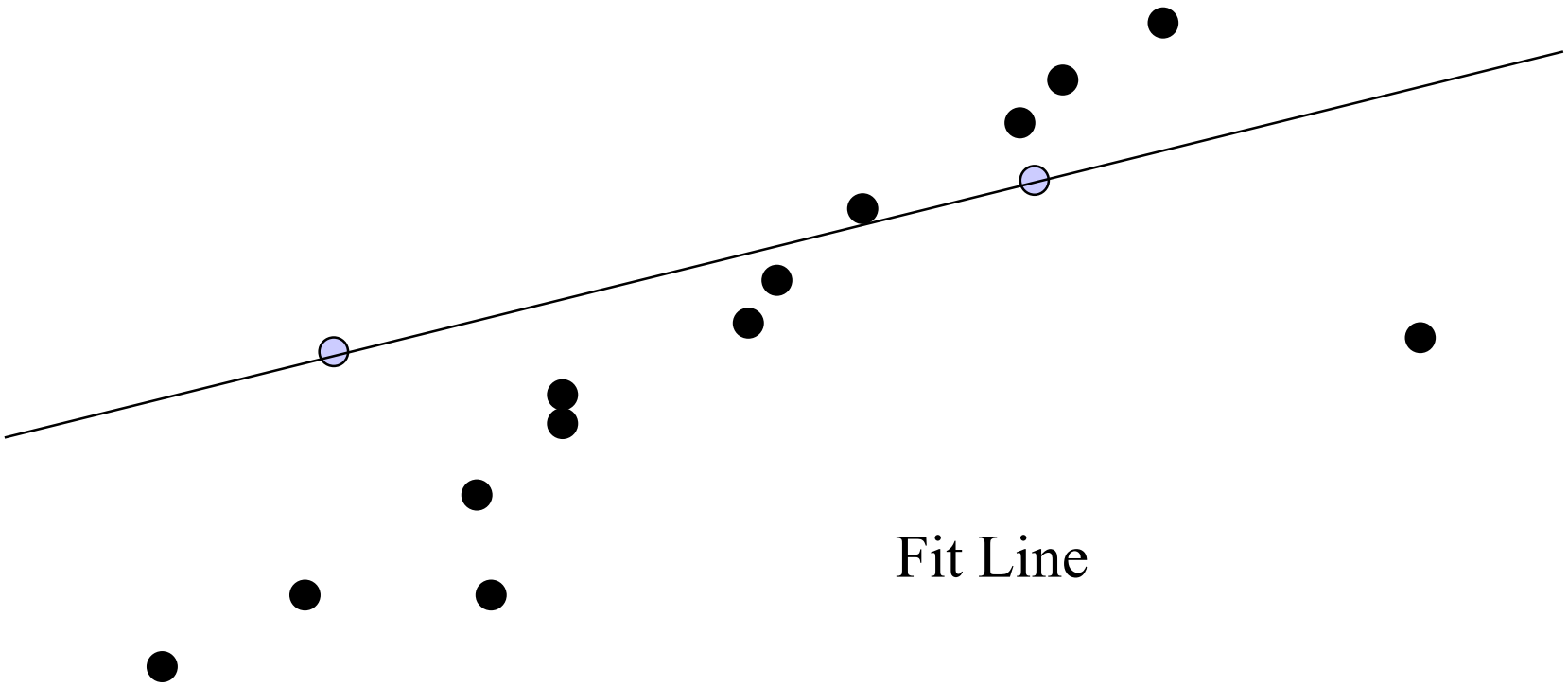
Estimate best line

Slide credit: Jinxiang Chai, CMU

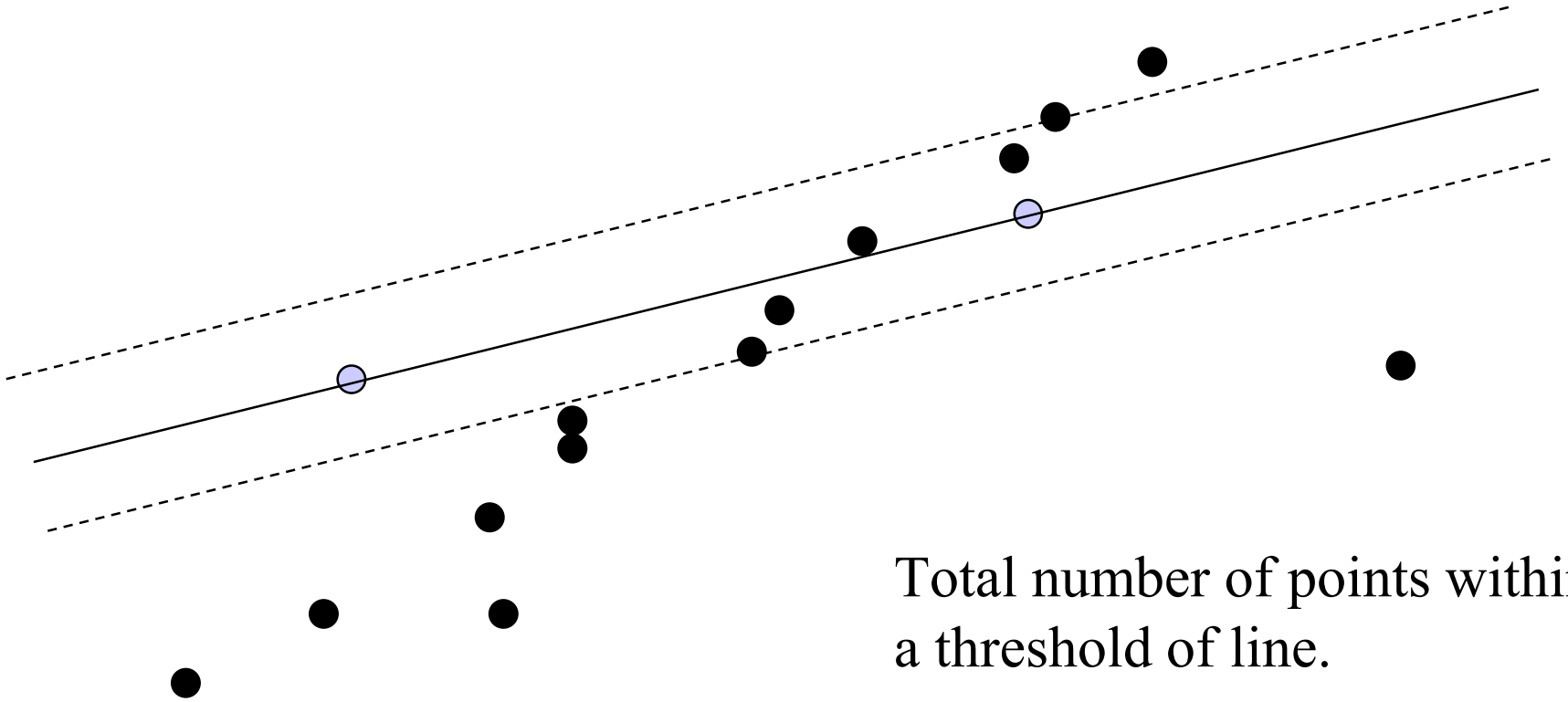
RANSAC Line Fitting Example



RANSAC Line Fitting Example

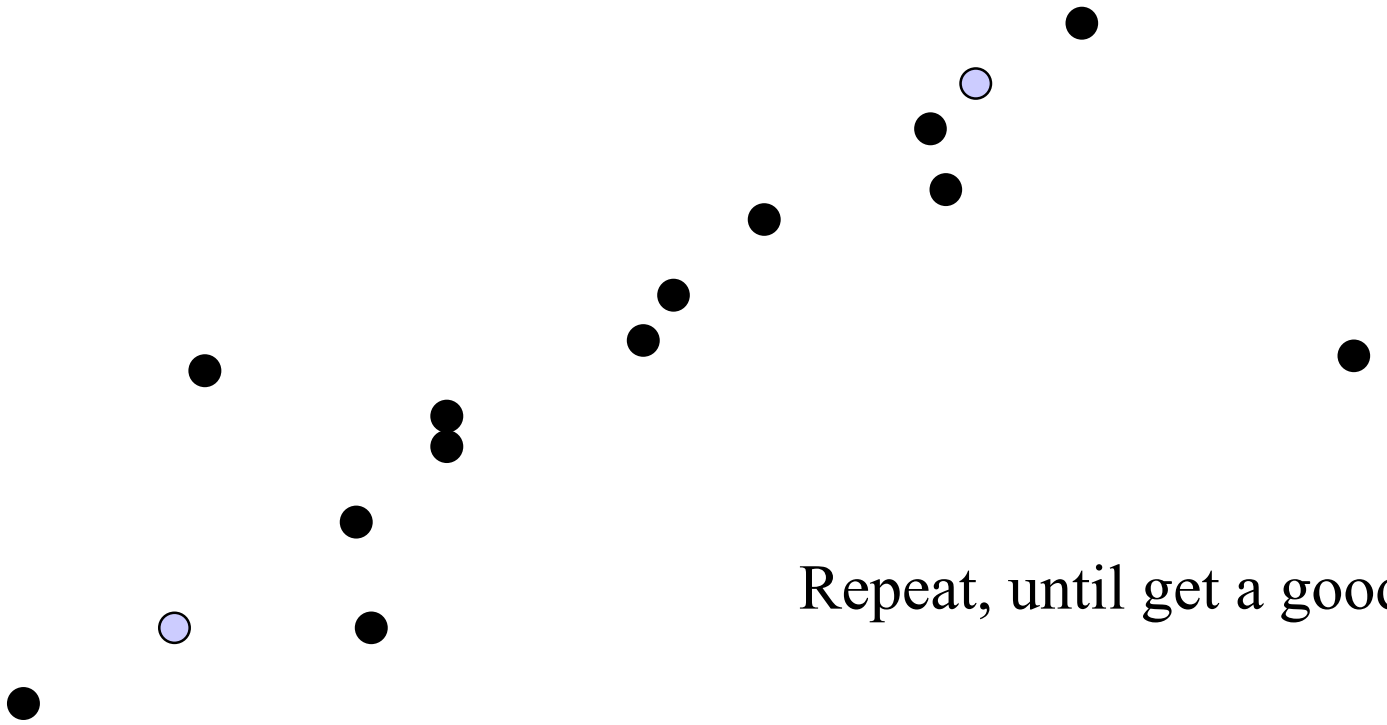


RANSAC Line Fitting Example



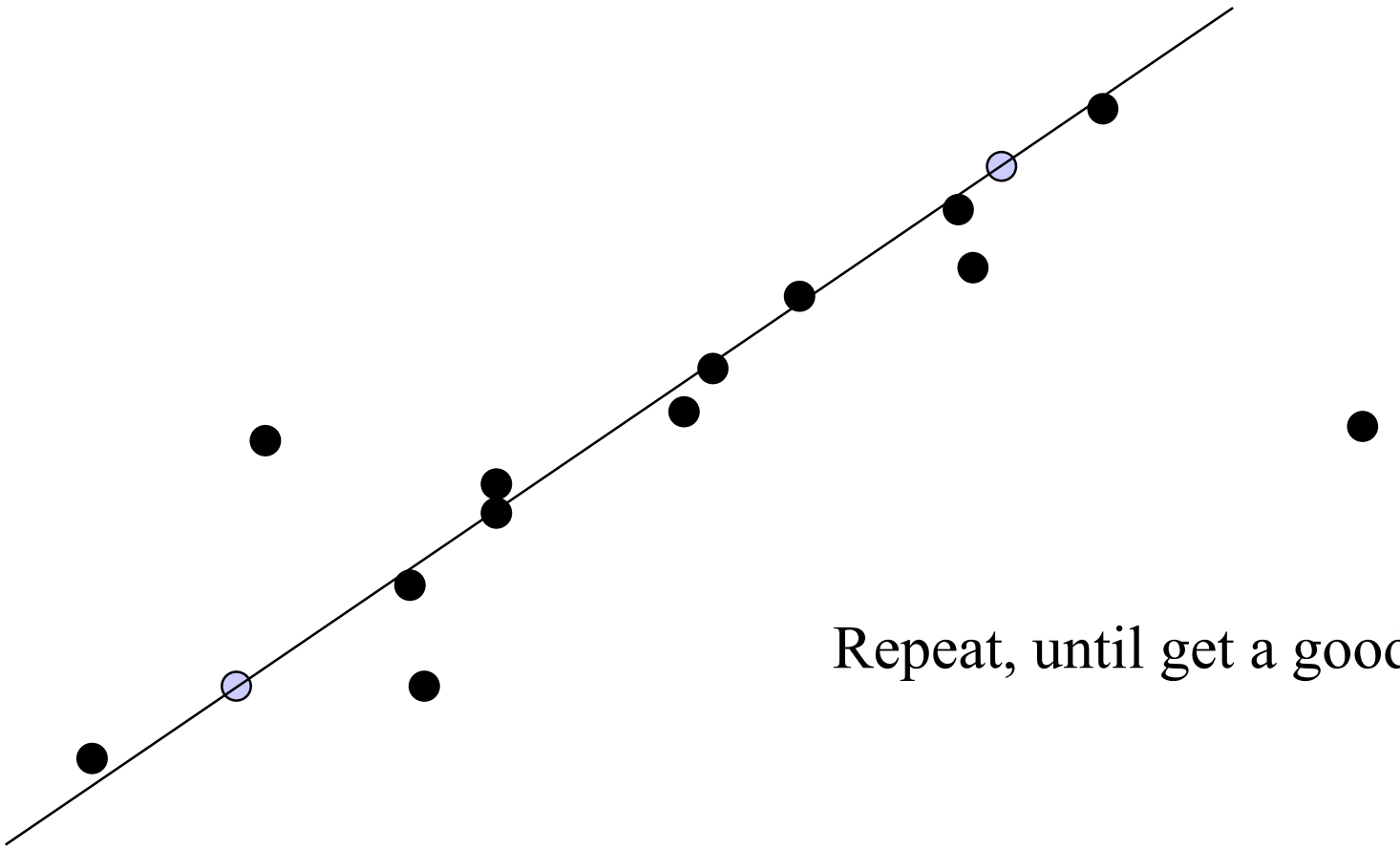
Total number of points within a threshold of line.

RANSAC Line Fitting Example



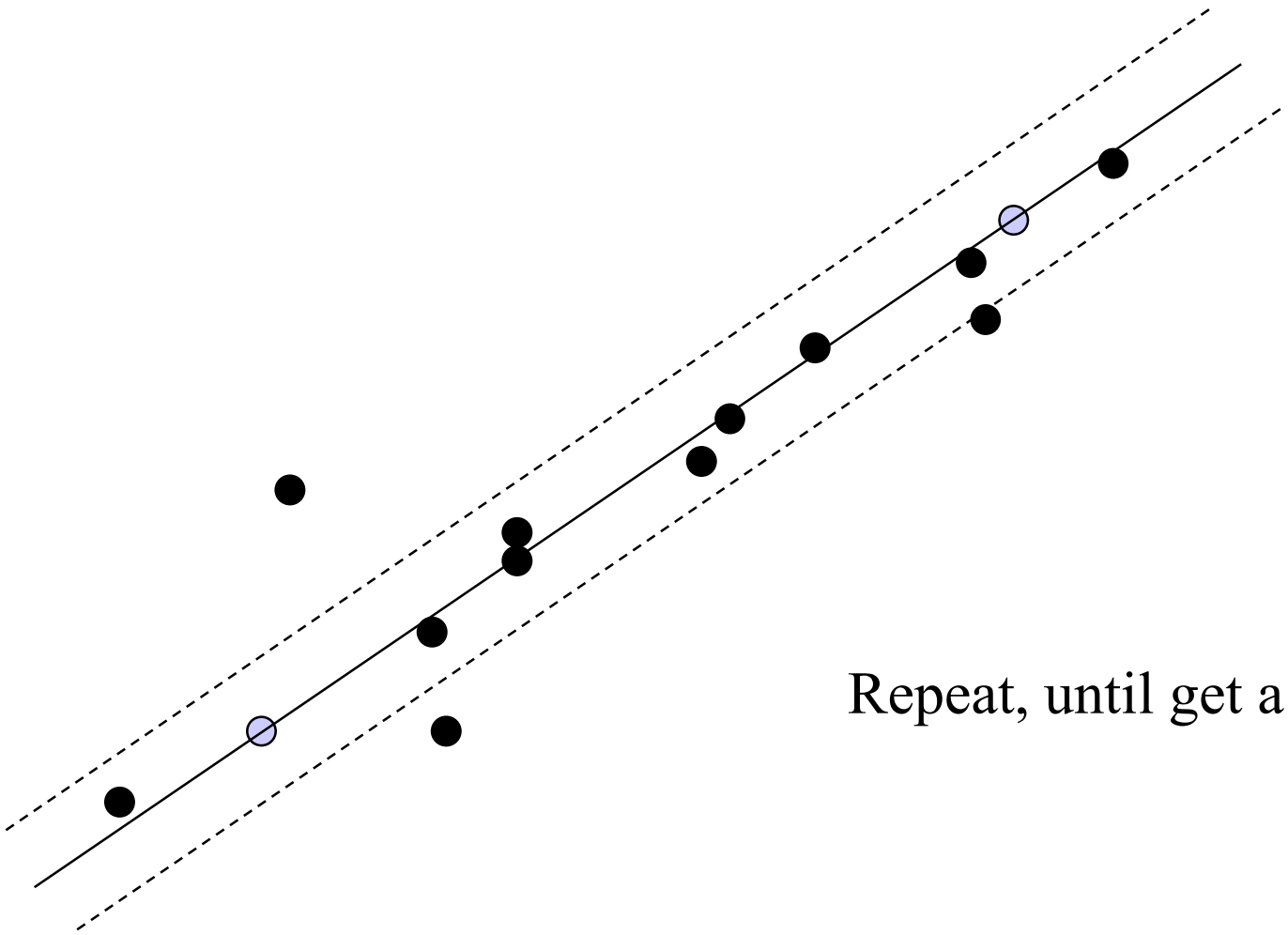
Repeat, until get a good result

RANSAC Line Fitting Example

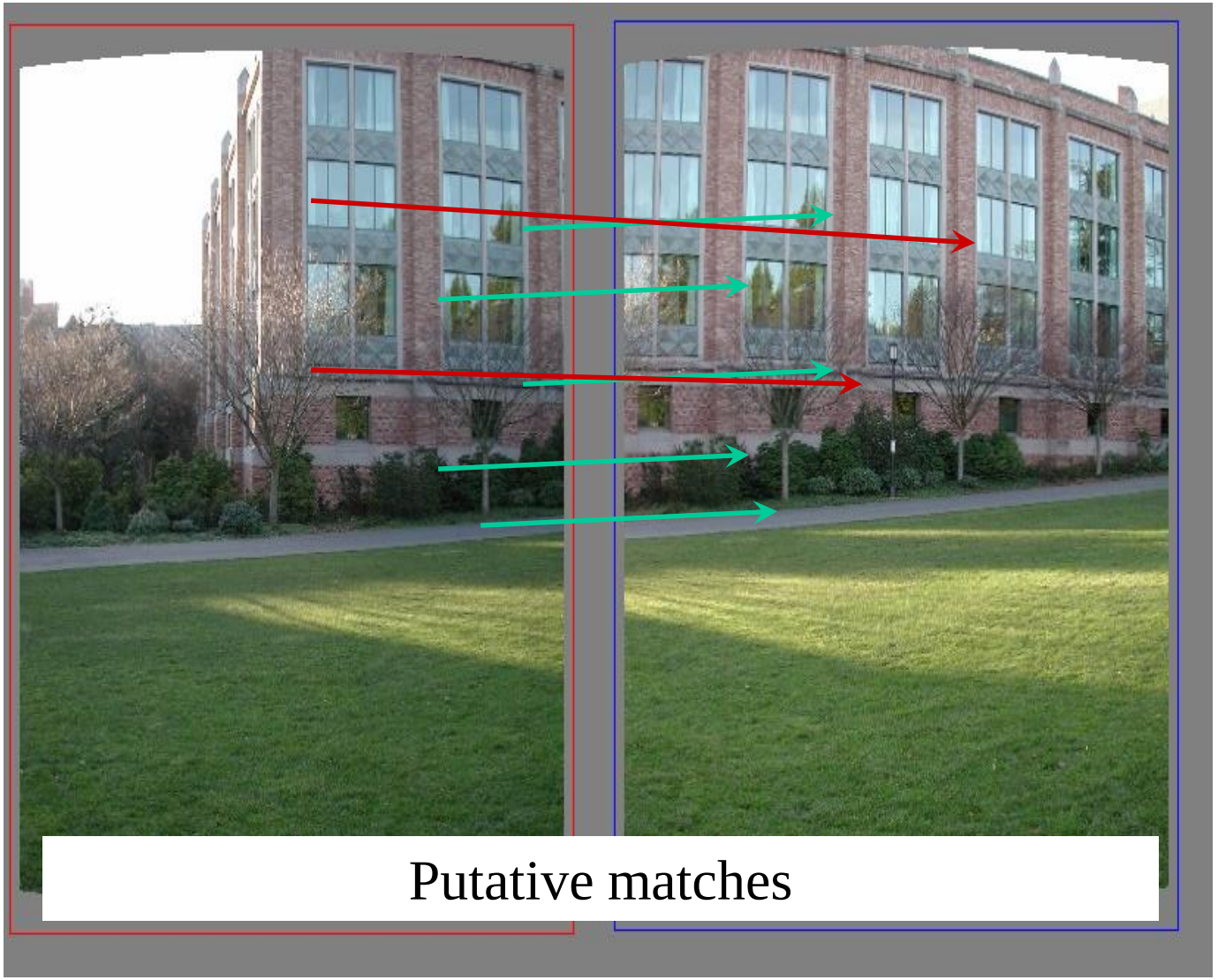


Repeat, until get a good result

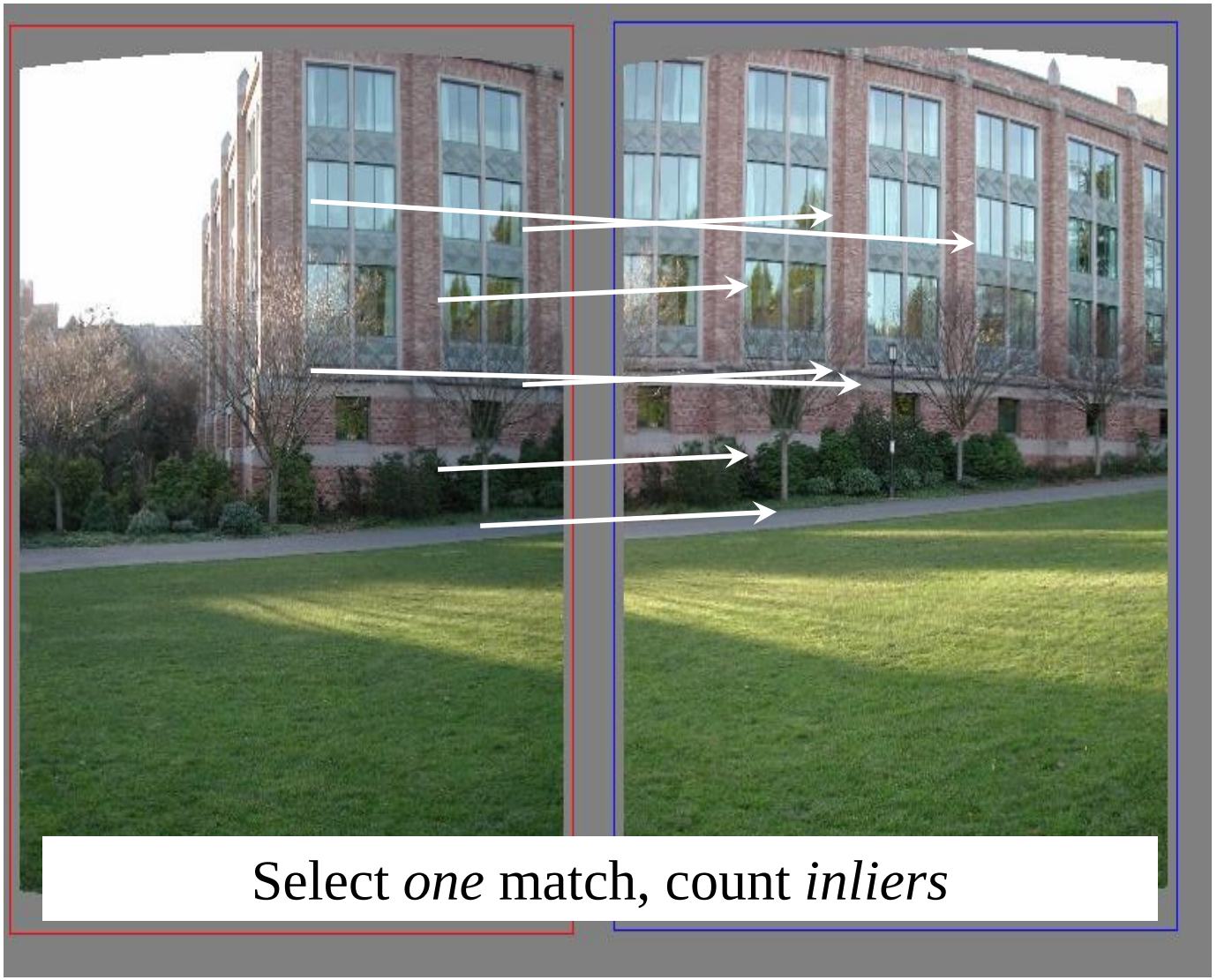
RANSAC Line Fitting Example

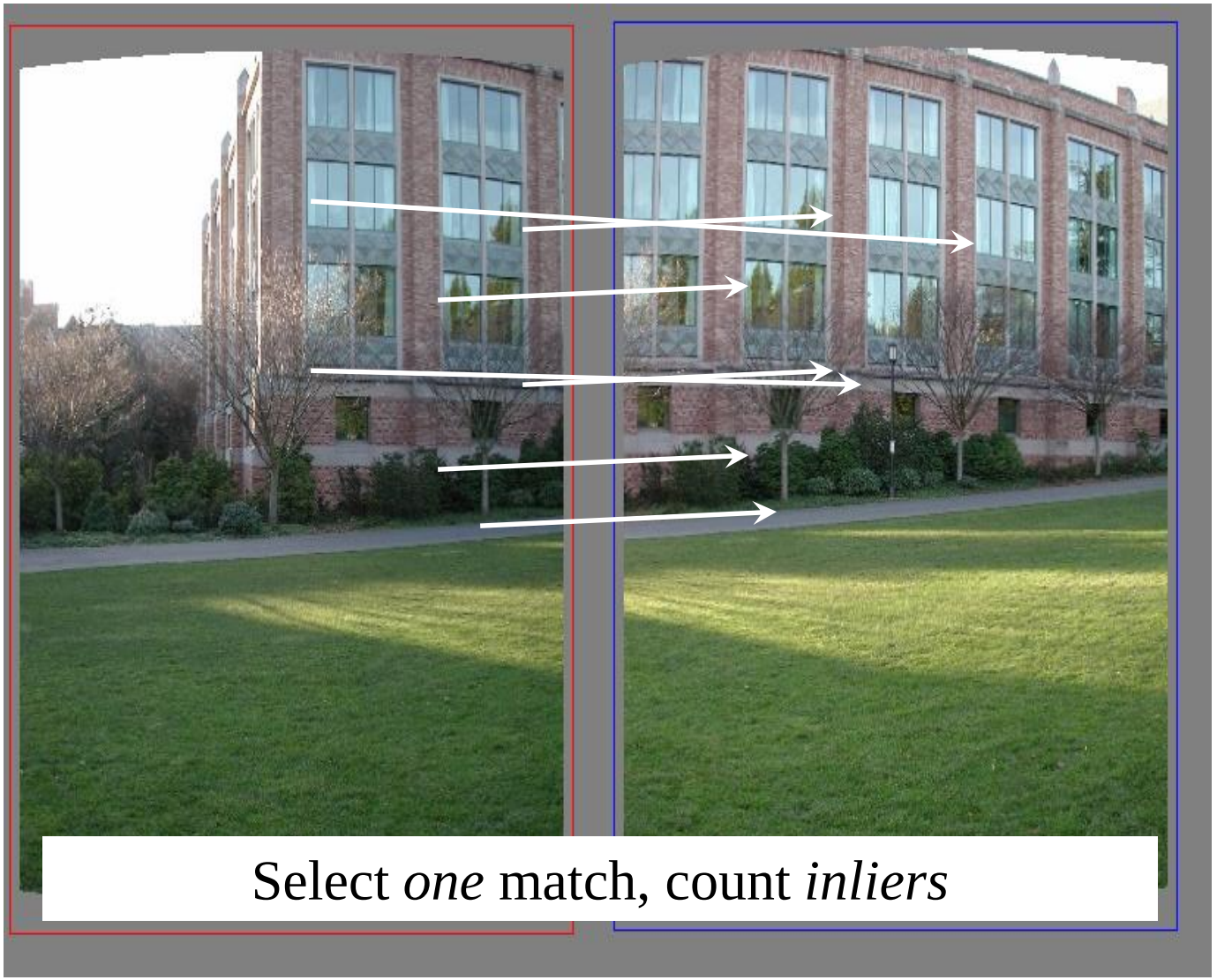


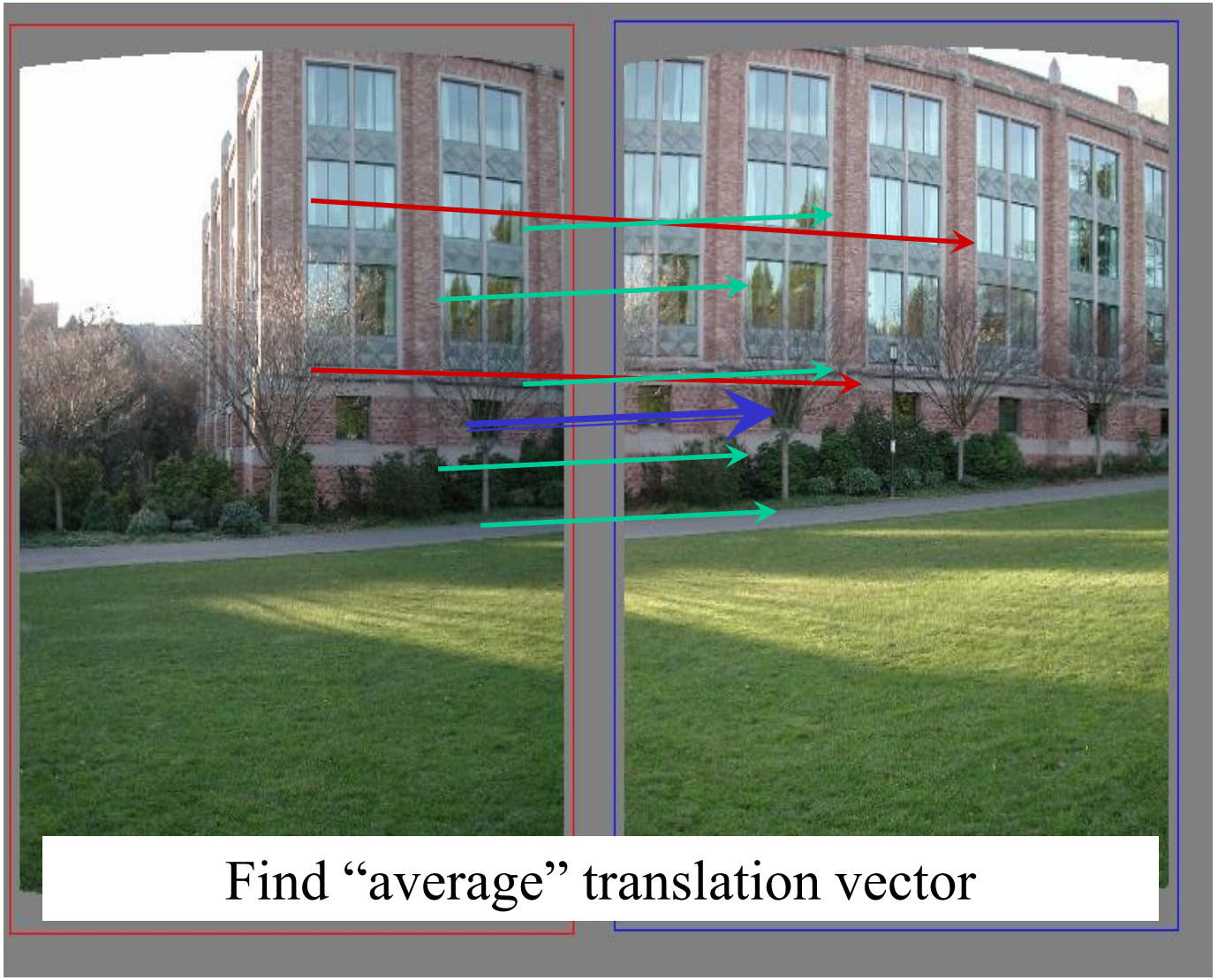
Repeat, until get a good result



Source: Rick Szeliski


Select *one* match, count *inliers*

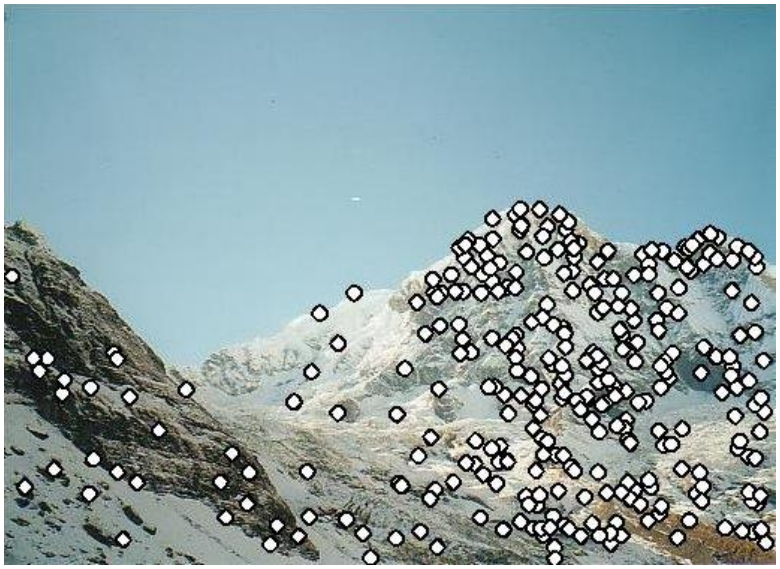

Select *one* match, count *inliers*

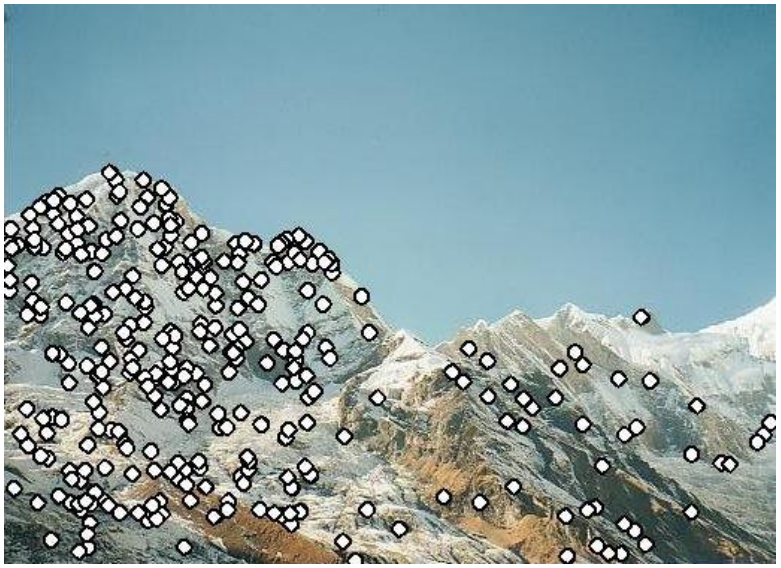

Find “average” translation vector

Feature-based alignment outline

Source: L. Lazebnik

Feature-based alignment outline





- Extract features

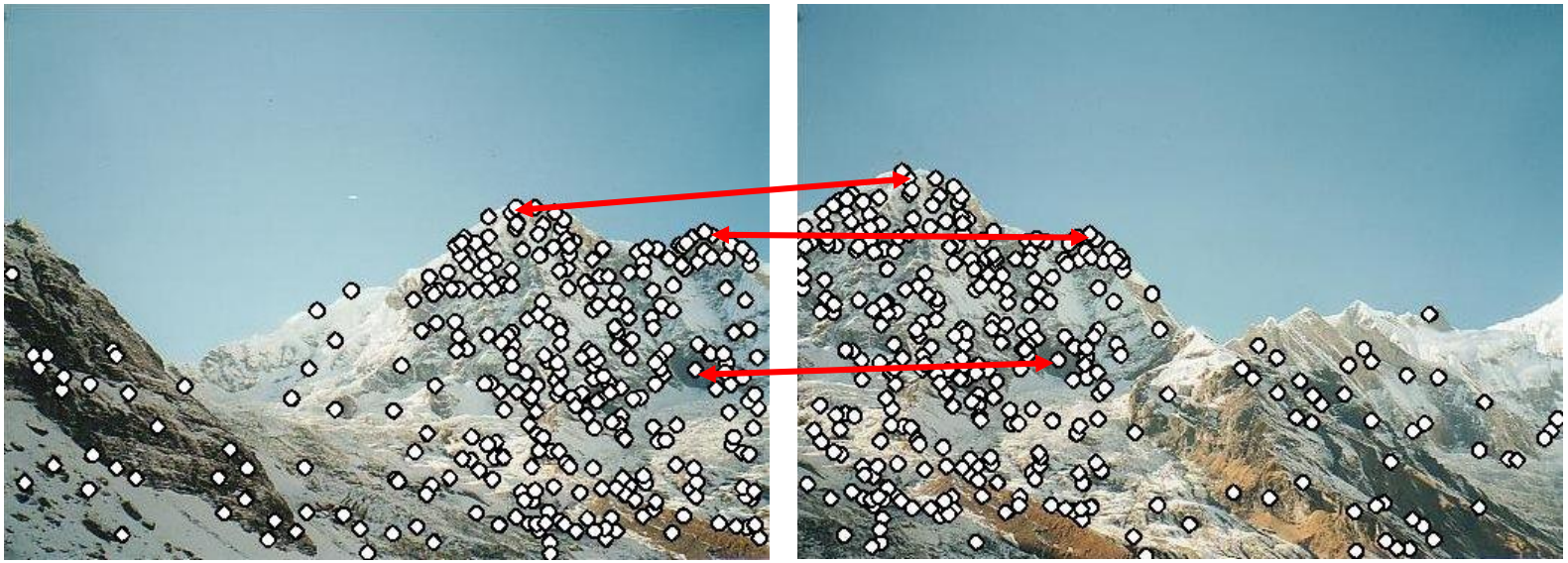
Source: L. Lazebnik

Feature-based alignment outline

- Extract features
- Compute *putative matches*

Source: L. Lazebnik

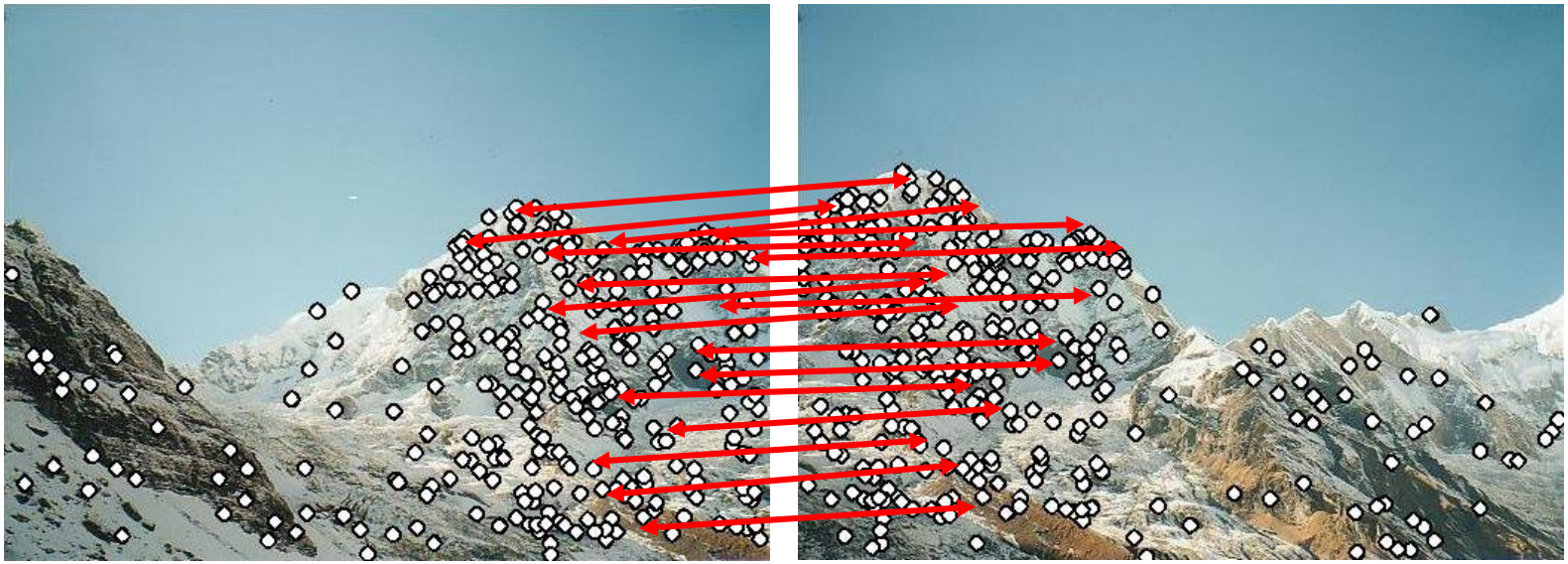
Feature-based alignment outline



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation T (small group of putative matches that are related by T)

Source: L. Lazebnik

Feature-based alignment outline



- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation T (small group of putative matches that are related by T)
  - *Verify* transformation (search for other matches consistent with T)

Source: L. Lazebnik

Feature-based alignment outline

- Extract features
- Compute *putative matches*
- Loop:
  - *Hypothesize* transformation *T* (small group of putative matches that are related by *T*)
  - *Verify* transformation (search for other matches consistent with *T*)

Source: L. Lazebnik

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

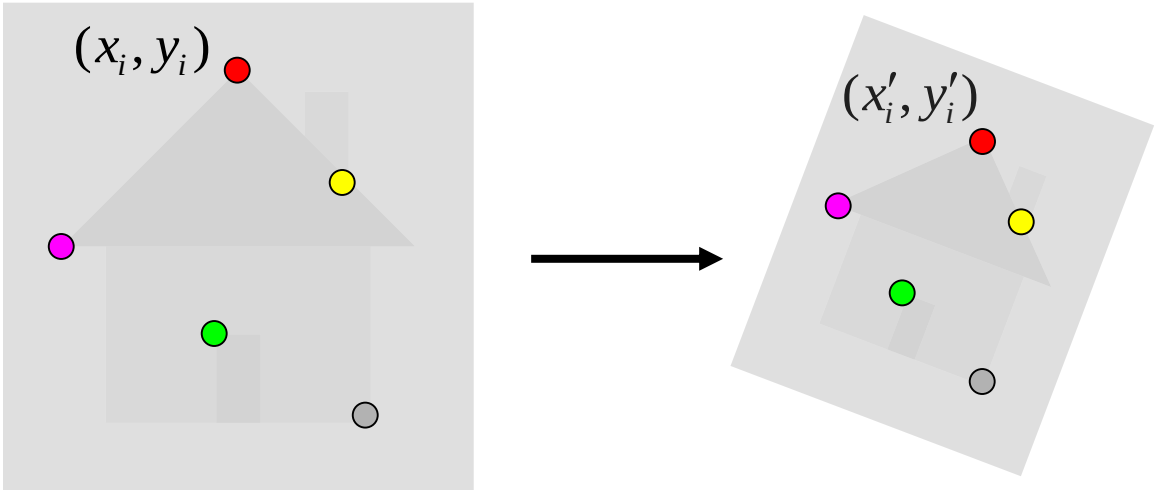
Fitting an affine transformation

Affine model approximates perspective projection of planar objects.

Figures from David Lowe, ICCV 1999

Fitting an affine transformation

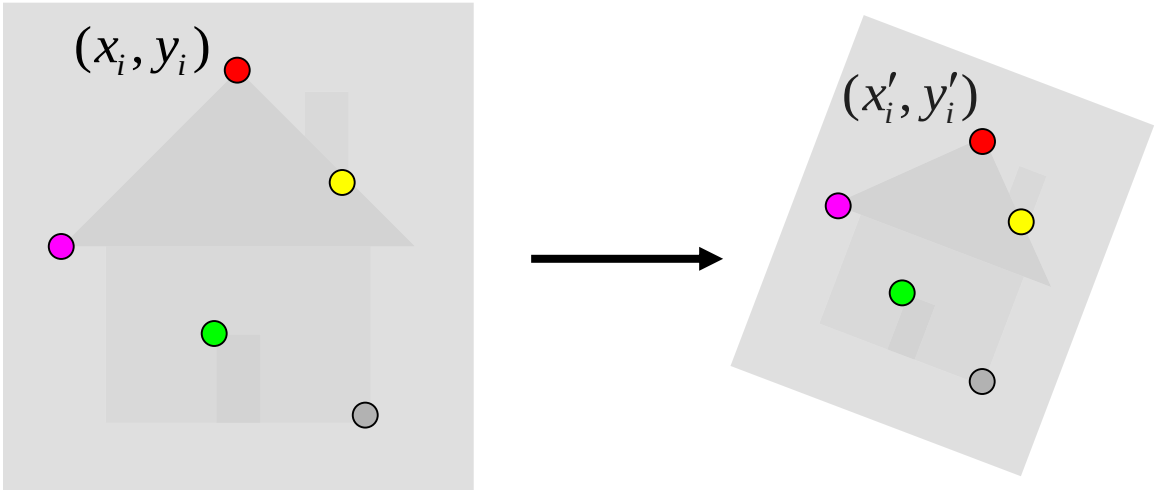
- Assuming we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

Fitting an affine transformation

- Assuming we know the correspondences, how do we get the transformation?



$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

$$\begin{bmatrix} & & \\ & & \\ & & \\ & & \\ & & \\ & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} & \\ & \\ & \\ & \\ & \\ & \\ & \\ & \end{bmatrix}$$

Fitting an affine transformation

$$\begin{bmatrix} & & \Lambda & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \Lambda & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \Lambda \\ x_i' \\ y_i' \\ \Lambda \end{bmatrix}$$

- How many matches (correspondence pairs) do we need to solve for the transformation parameters?
- Once we have solved for the parameters, how do we compute the coordinates of the corresponding point for (x_{new}, y_{new}) ?

Conclusion

- Finding lines and other parameterized objects is an important task for computer vision.
- The generalized Hough transform can detect arbitrary shapes from (edge detected) tokens.
- Success rate depends directly upon the noise in the edge image.
- The concept of Hough transform can be extended for robust model fitting.
- M-estimator
- RANSAC