Fitting

Computer Vision

CS 543 / ECE 549

University of Illinois

Slides from Derek Hoiem, Svetlana Lazebnik

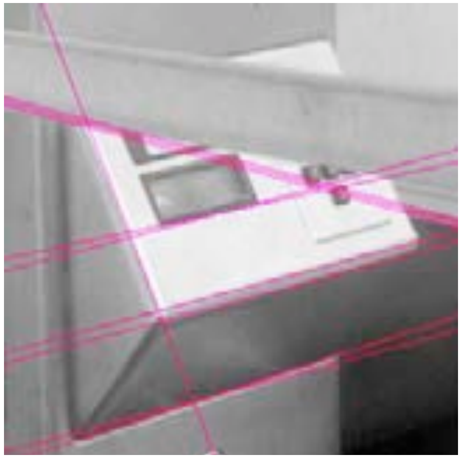
Fitting

- We've learned how to detect edges, corners, blobs. Now what?
- We would like to form a higher-level, more compact representation of the features in the image by grouping multiple features according to a simple model

Slide from L. Lazebnik

Fitting

- Choose a *parametric model* to represent a set of features



simple model: lines

simple model: circles

complicated model: car

Source: K. Grauman

Fitting Methods

- Global optimization / Search for parameters
  – Least squares fit
  – Robust least squares
- Hypothesize and test
  – Generalized Hough transform
  – RANSAC

Slide from D. Hoiem

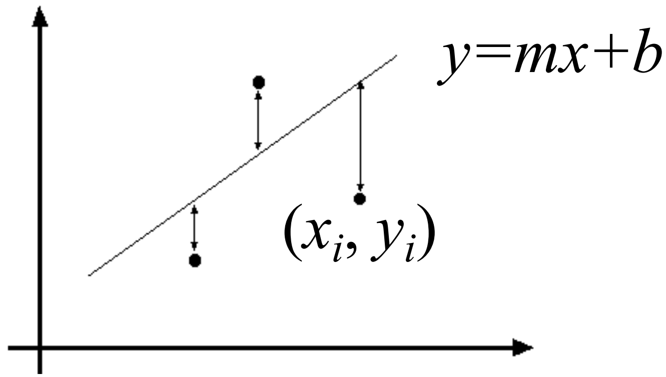
Simple example: Fitting a line

Slide from D. Hoiem

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

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



$$E = \sum_{i=1}^{n} \left(\begin{bmatrix} x_i & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - y_i \right)^2 = \left\| \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} - \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \right\|^2 = \|\mathbf{Ap} - \mathbf{y}\|^2$$

$$= \mathbf{y}^T \mathbf{y} - 2(\mathbf{Ap})^T \mathbf{y} + (\mathbf{Ap})^T (\mathbf{Ap})$$

$$\frac{dE}{dp} = 2\mathbf{A}^T \mathbf{Ap} - 2\mathbf{A}^T \mathbf{y} = 0$$

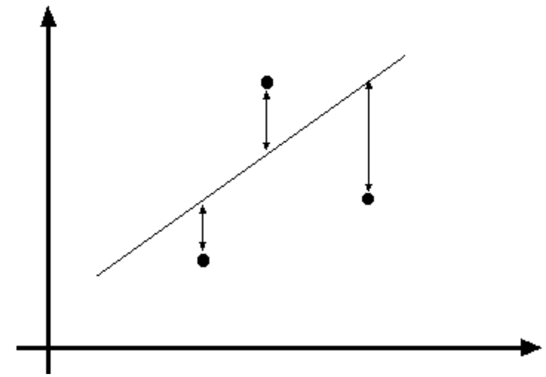
Matlab: `p = A \ y;`

$$\mathbf{A}^T \mathbf{Ap} = \mathbf{A}^T \mathbf{y} \Rightarrow \mathbf{p} = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{y}$$

Modified from S. Lazebnik

Problem with “vertical” least squares

- Not rotation-invariant
- Fails completely for vertical lines



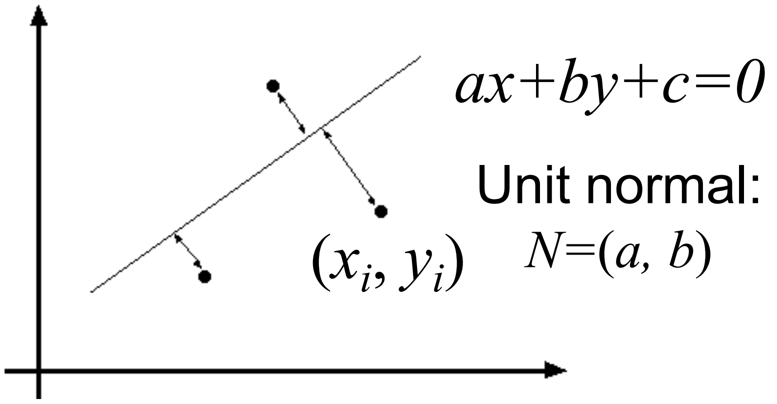
Slide from S. Lazebnik

Total least squares

If (a^2+b^2=1) then

Distance between point (x_i, y_i) and line $ax+by+c=0$ is $|ax_i + by_i + c|$

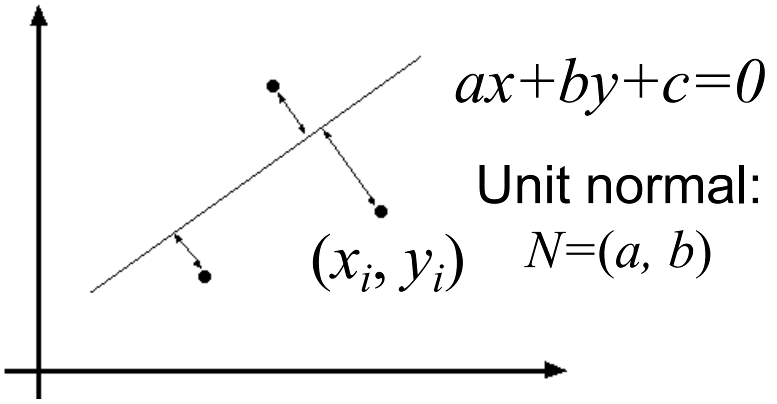
proof:
http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html



Total least squares

If ($a^2+b^2=1$) then

Distance between point (x_i, y_i) and line $ax+by+c=0$ is $|ax_i + by_i + c|$



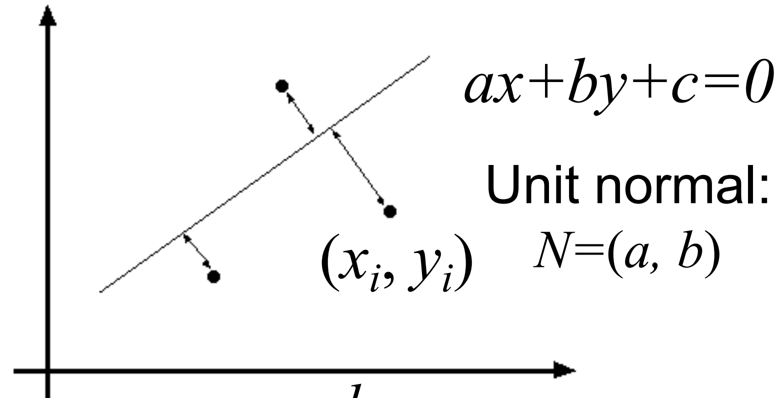
Find (a, b, c) to minimize the sum of squared perpendicular distances

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

Total least squares

Find (a, b, c) to minimize the sum of squared perpendicular distances

$$E = \sum_{i=1}^{n} (a x_i + b y_i + c)^2$$



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

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

$$\text{minimize}\, \mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p} \quad \text{s.t.} \quad \mathbf{p}^T \mathbf{p} = 1 \quad \Rightarrow \quad \text{minimize} \frac{\mathbf{p}^T \mathbf{A}^T \mathbf{A} \mathbf{p}}{\mathbf{p}^T \mathbf{p}}$$

Solution is eigenvector corresponding to smallest eigenvalue of $A^T A$

See details on Raleigh Quotient: http://en.wikipedia.org/wiki/Rayleigh_quotient

Recap: Two Common Optimization Problems

Problem statement

$$\text{minimize } \|\mathbf{Ax}-\mathbf{b}\|^2$$

$$\text{least squares solution to } \mathbf{Ax}=\mathbf{b}$$

Solution

$$\mathbf{x}=\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

$$\mathbf{x}=\mathbf{A}\setminus\mathbf{b} \quad \text{(matlab)}$$

Problem statement

$$\text{minimize } \mathbf{x}^T\mathbf{A}^T\mathbf{Ax} \quad \text{s.t. } \mathbf{x}^T\mathbf{x}=1$$

$$\text{minimize} \frac{\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}}{\mathbf{x}^T\mathbf{x}}$$

$$\text{non-trivial lsq solution to } \mathbf{Ax}=0$$

Solution

$$[\mathbf{v},\lambda]=\text{eig}(\mathbf{A}^T\mathbf{A})$$

$$\lambda_1<\lambda_{2..n}:\mathbf{x}=\mathbf{v}_1$$

Slide from D. Hoiem

Least squares (global) optimization

Good

- Clearly specified objective
- Optimization is easy

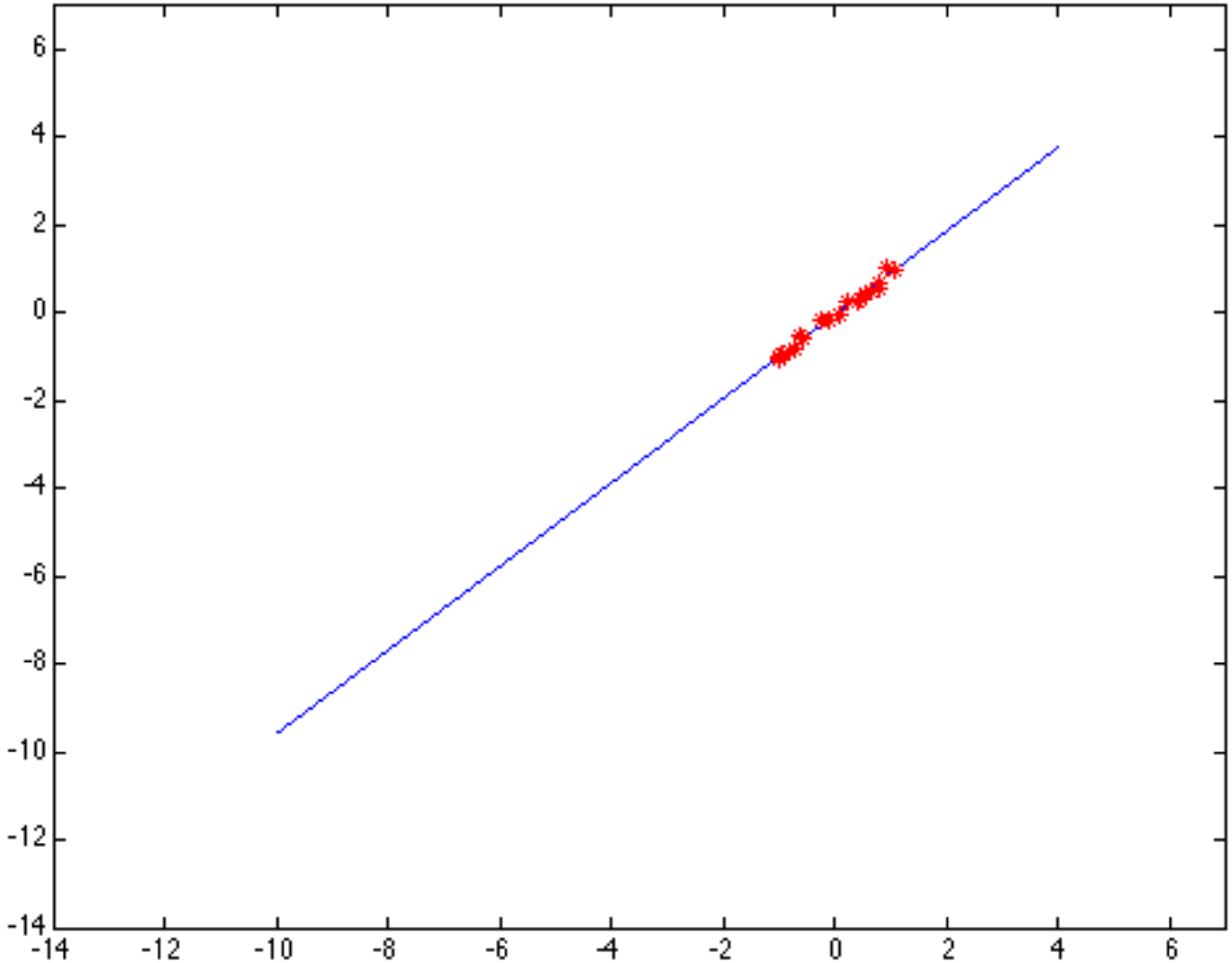
Bad

- May not be what you want to optimize
- Sensitive to outliers
  - Bad matches, extra points
- Doesn't allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

Slide from D. Hoiem

Least squares: Robustness to noise

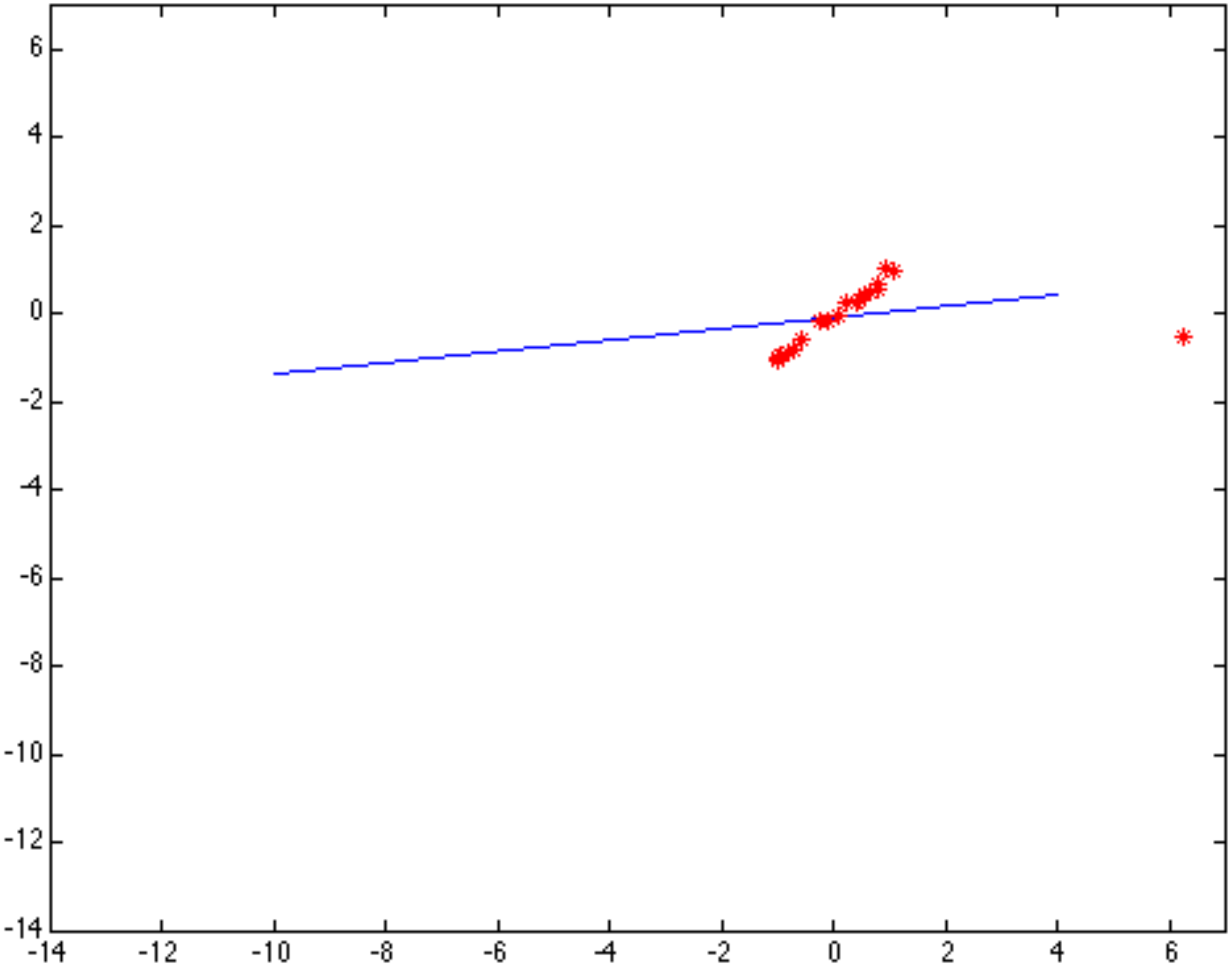
- Least squares fit to the red points:



Slide from L. Lazebnik

Least squares: Robustness to noise

- Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

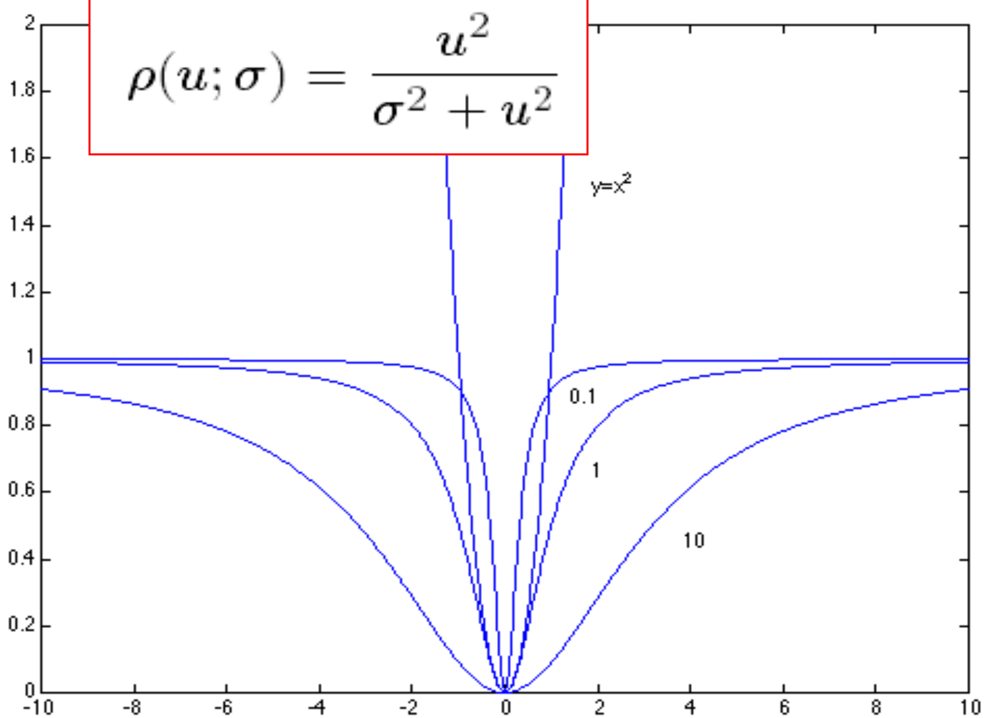
Slide from L. Lazebnik

Robust least squares (to deal with outliers)

General approach:

minimize $$\sum_{i} \rho\left(u_i\left(x_i,\theta\right);\sigma\right) \qquad u^2 = \sum_{i=1}^{n}(y_i - mx_i - b)^2$$

$u_i(x_i, \theta)$ – residual of i^th point w.r.t. model parameters ϑ
ρ – robust function with scale parameter σ



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

The robust function ρ

- Favors a configuration with small residuals
- Constant penalty for large residuals

Slide from S. Savarese

Robust Estimator

1. Initialize: e.g., choose θ by least squares fit and
   $\sigma = 1.5 \cdot \mathrm{median}(error)$

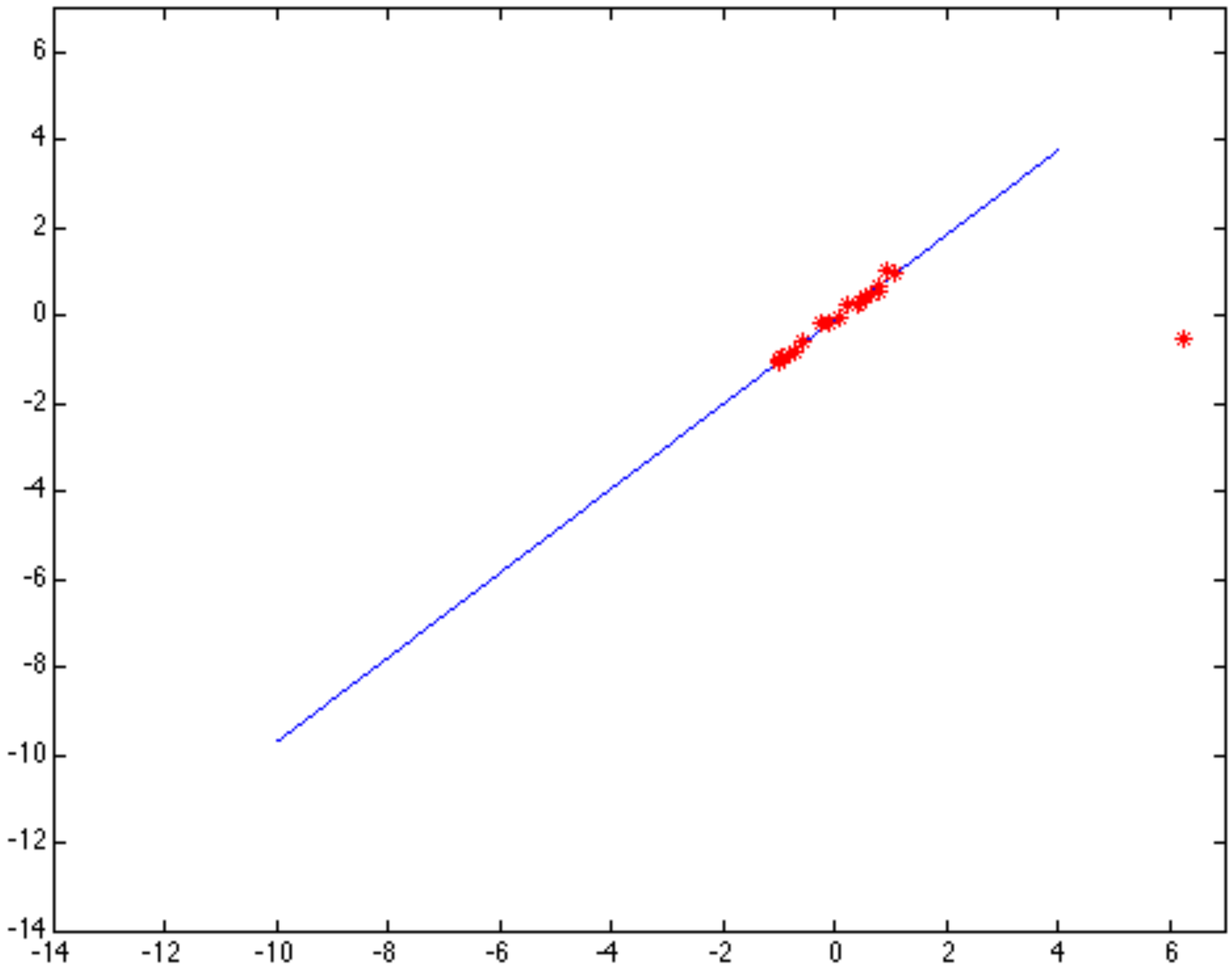
2. Choose params to minimize: $\sum_i \frac{error(\theta, data_i)^2}{\sigma^2 + error(\theta, data_i)^2}$
   – E.g., numerical optimization

3. Compute new $\sigma = 1.5 \cdot \mathrm{median}(error)$

4. Repeat (2) and (3) until convergence

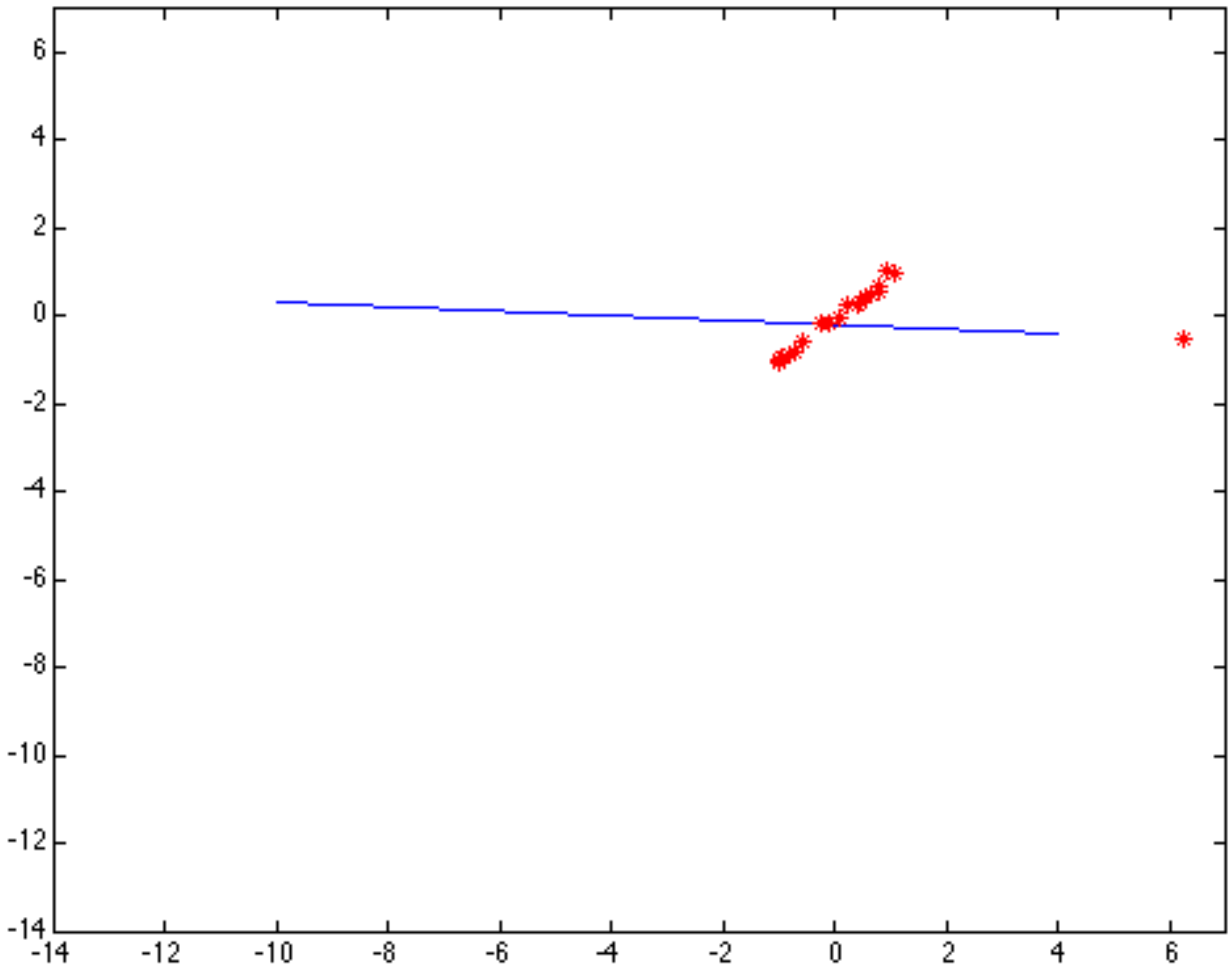
Slide from D. Hoiem

Choosing the scale: Just right



The effect of the outlier is minimized

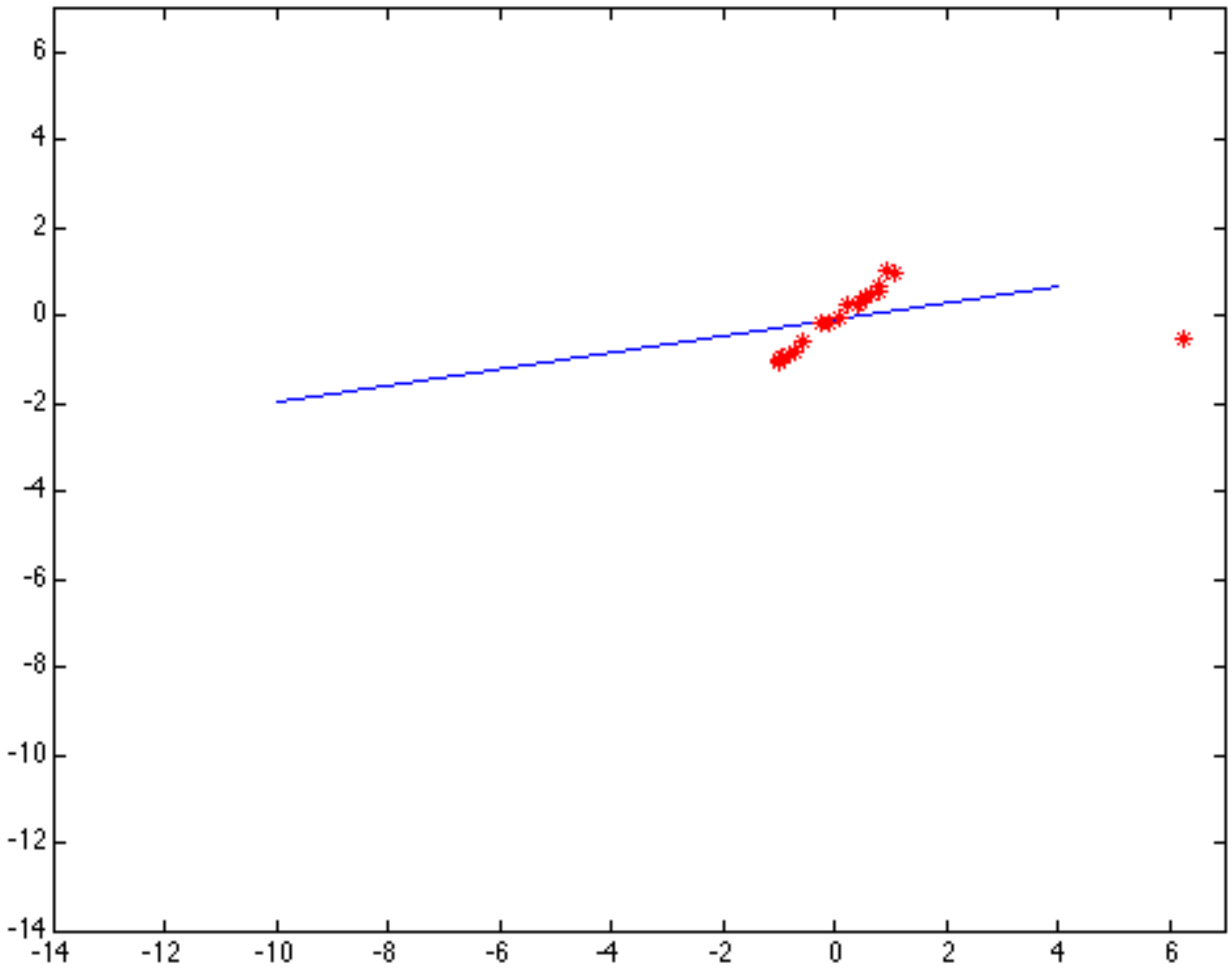
Choosing the scale: Too small



The error value is almost the same for every point and the fit is very poor

Slide from L. Lazebnik

Choosing the scale: Too large



Behaves much the same as least squares

Slide from L. Lazebnik

Other ways to search for parameters (for when no closed form solution exists)

- Line search
  1. For each parameter, step through values and choose value that gives best fit
  2. Repeat (1) until no parameter changes

- Grid search
  1. Propose several sets of parameters, evenly sampled in the joint set
  2. Choose best (or top few) and sample joint parameters around the current best; repeat

- Gradient descent
  1. Provide initial position (e.g., random)
  2. Locally search for better parameters by following gradient

Slide from D. Hoiem

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

Slide from D. Hoiem

Hough Transform: Outline

1. Create a grid of parameter values

2. Each point votes for a set of parameters, incrementing those values in grid

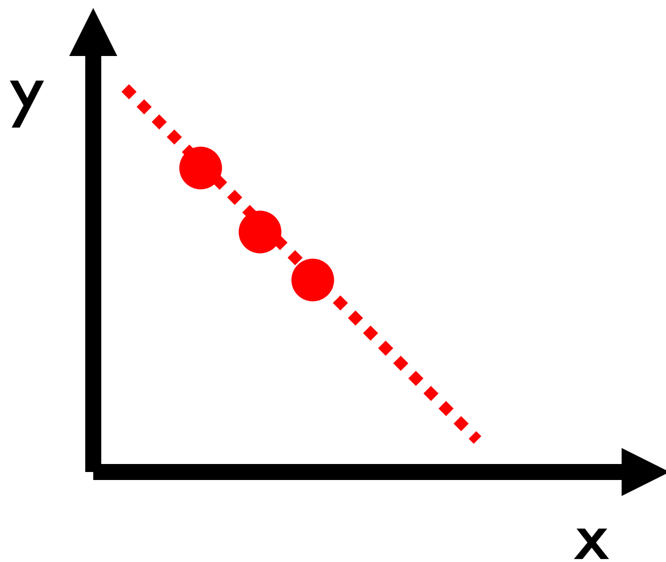
3. Find maximum or local maxima in grid

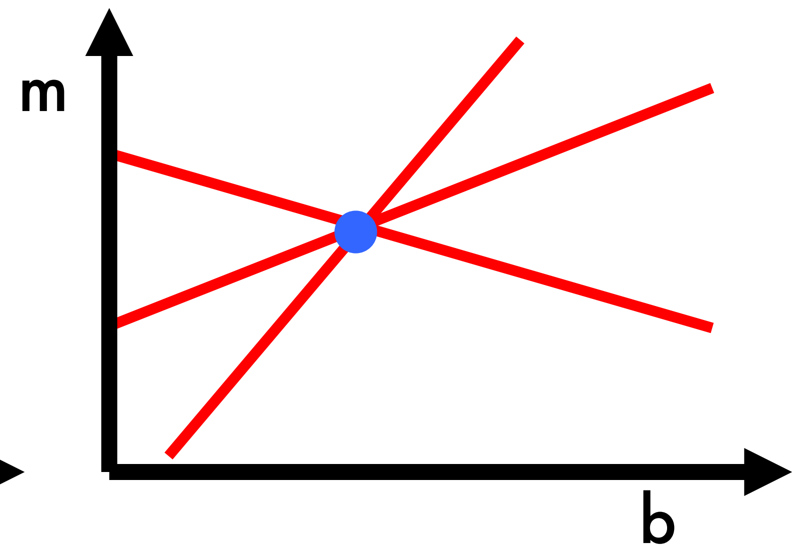
Slide from D. Hoiem

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



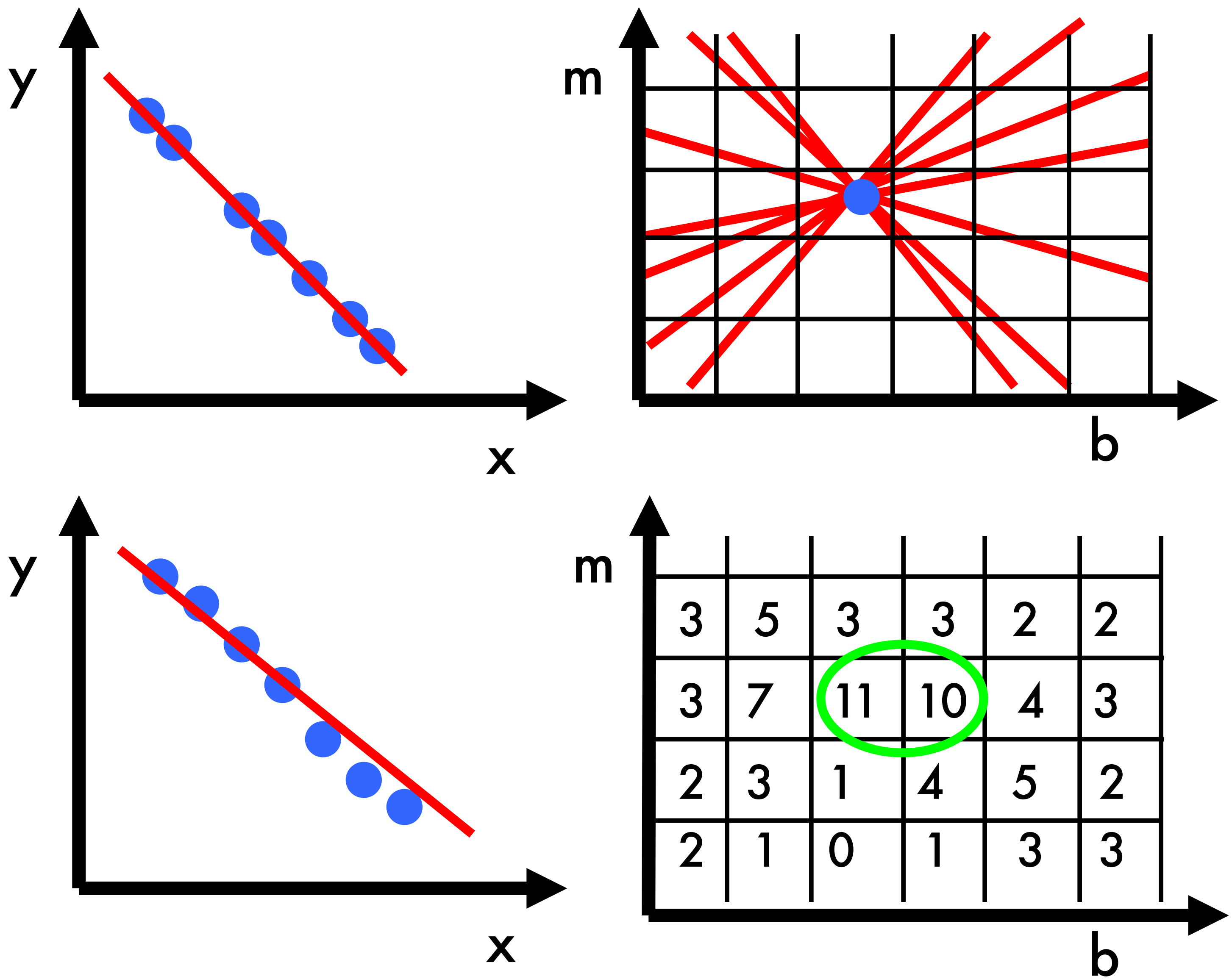


Hough space

$$y = m\,x + b$$

Slide from S. Savarese

Hough transform



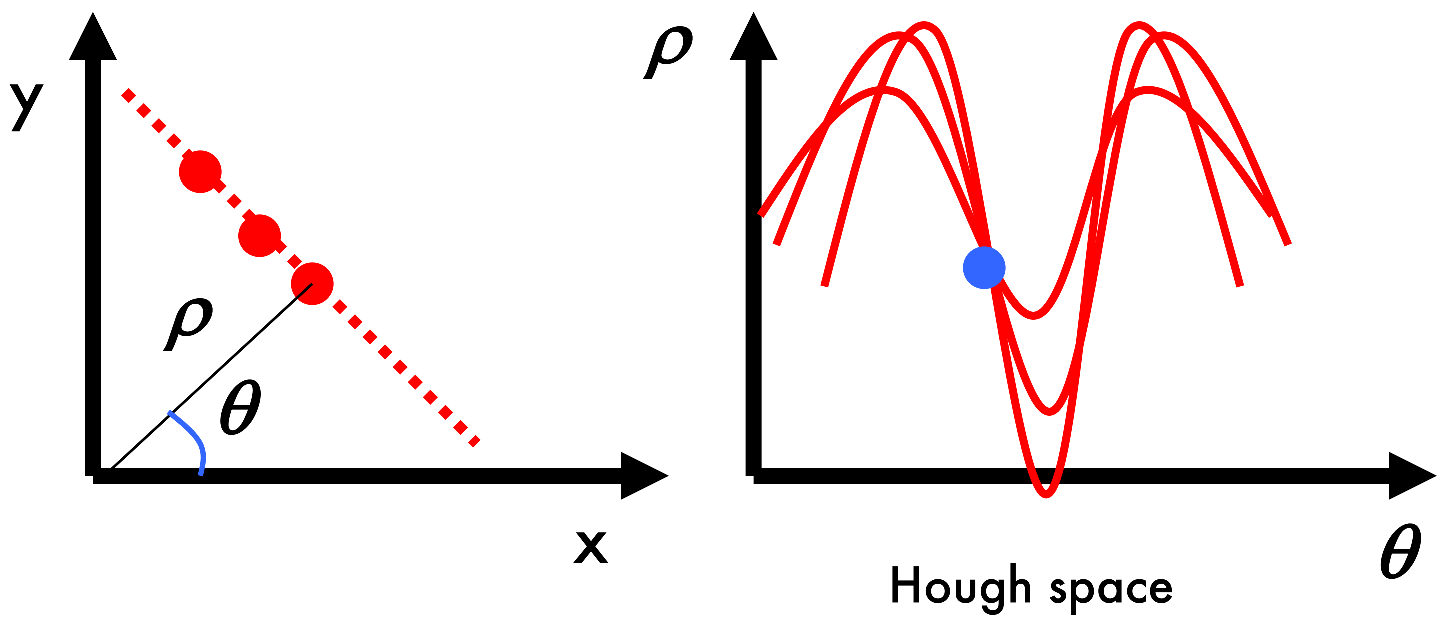
Slide from S. Savarese

Hough transform

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

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

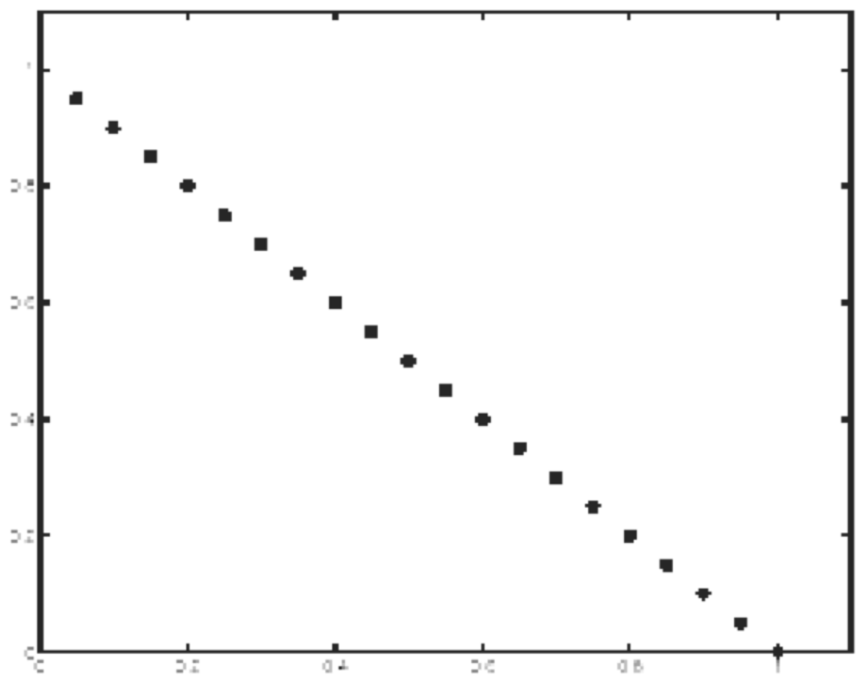
Use a polar representation for the parameter space



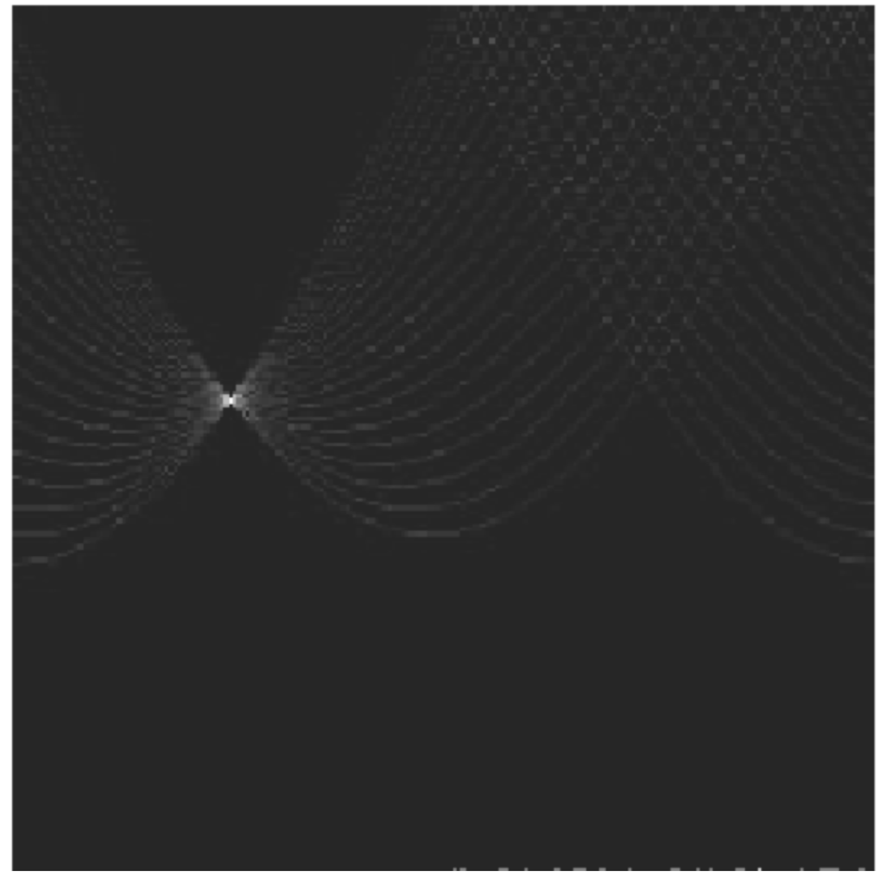
$$x\cos\theta + y\sin\theta = \rho$$

Slide from S. Savarese

Hough transform - experiments



features

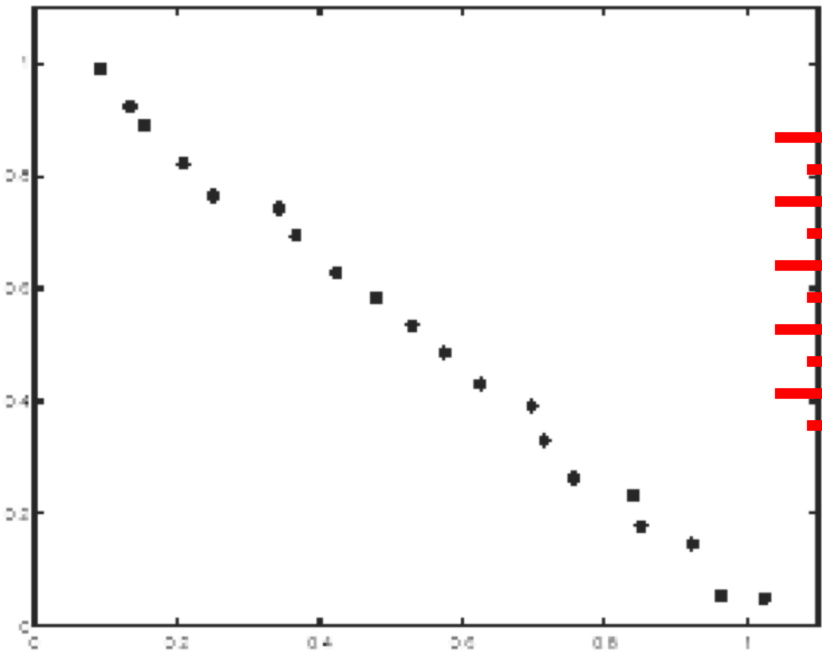


votes

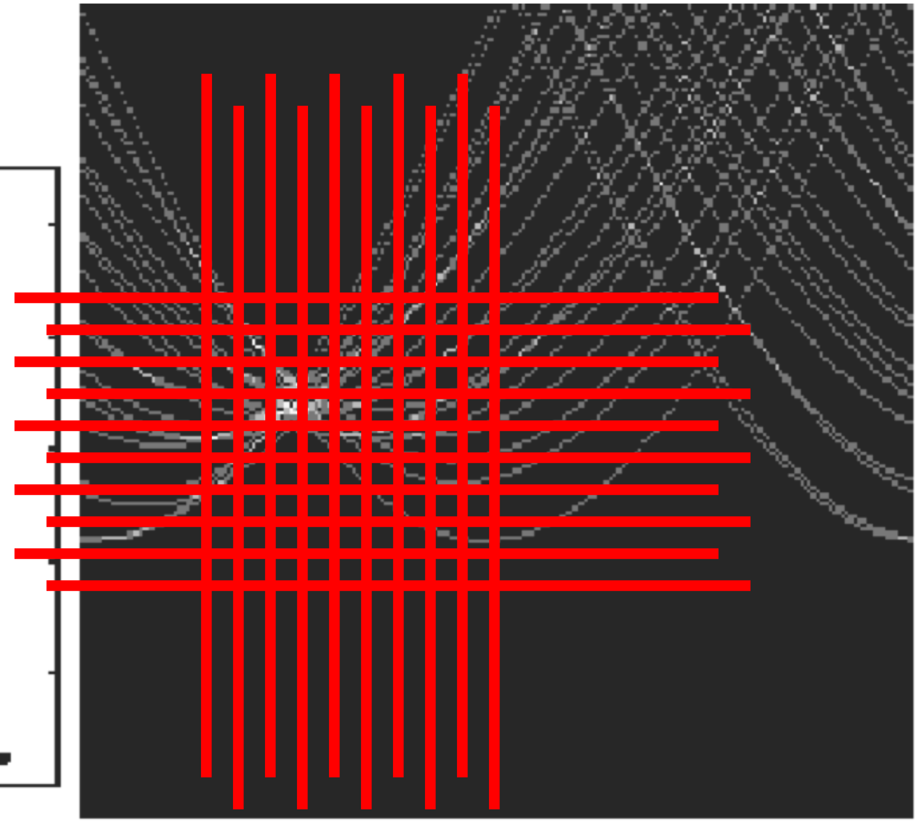
Slide from S. Savarese

Hough transform - experiments

Noisy data



features

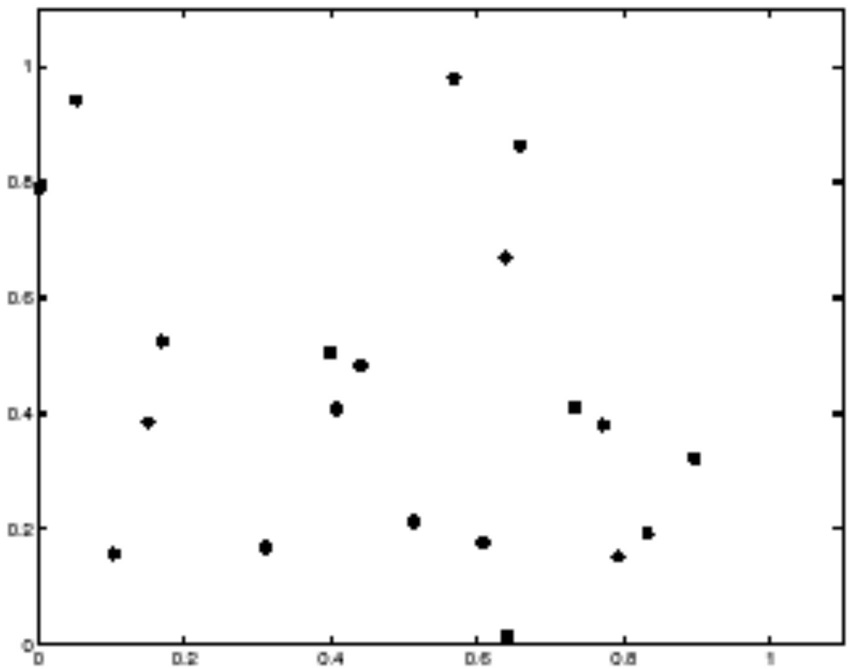


votes

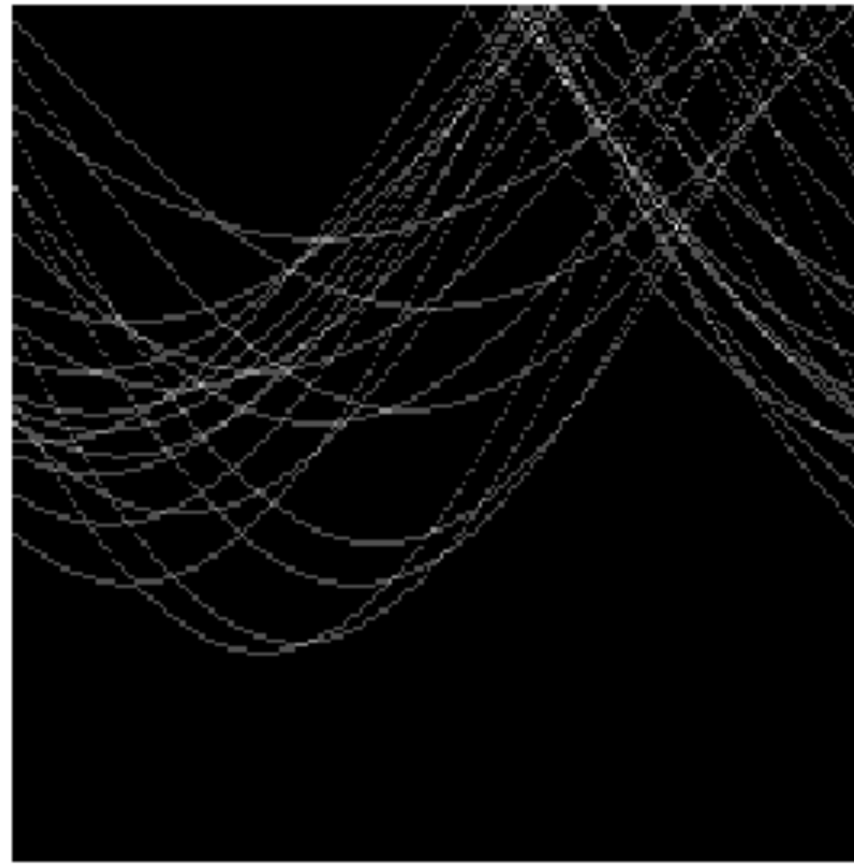
Need to adjust grid size or smooth

Slide from S. Savarese

Hough transform - experiments



features



votes

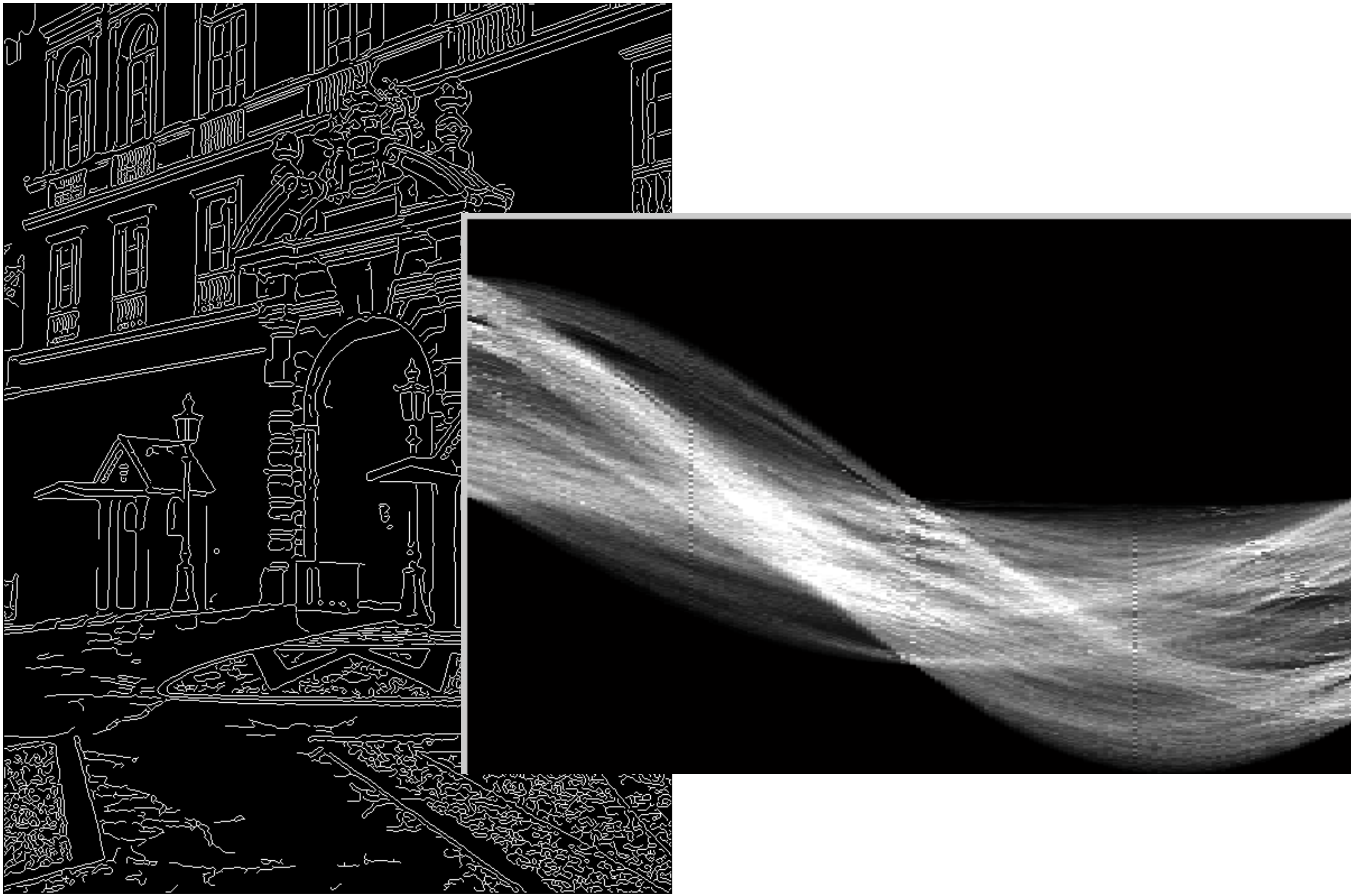
Issue: spurious peaks due to uniform noise

Slide from S. Savarese

1. Image → Canny

Slide from D. Hoiem

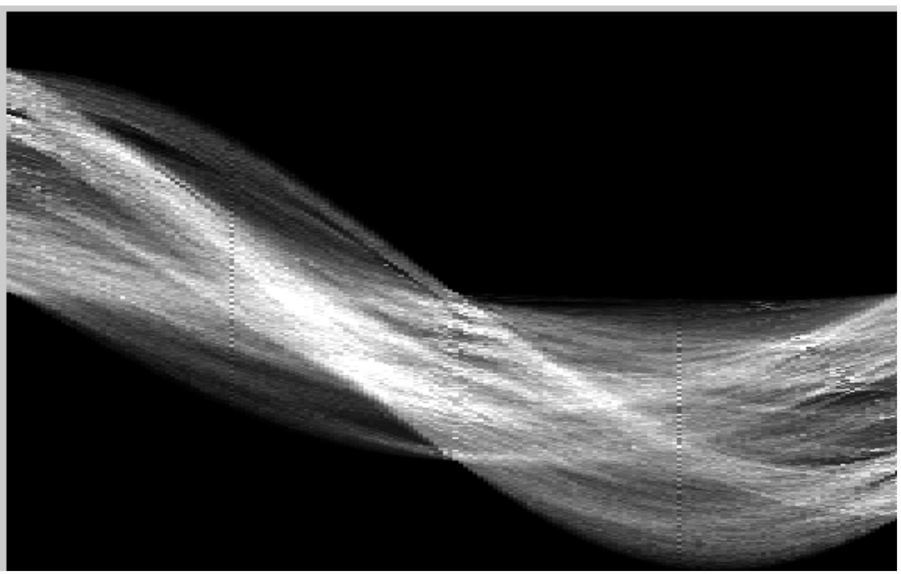
2. Canny → Hough votes



Slide from D. Hoiem

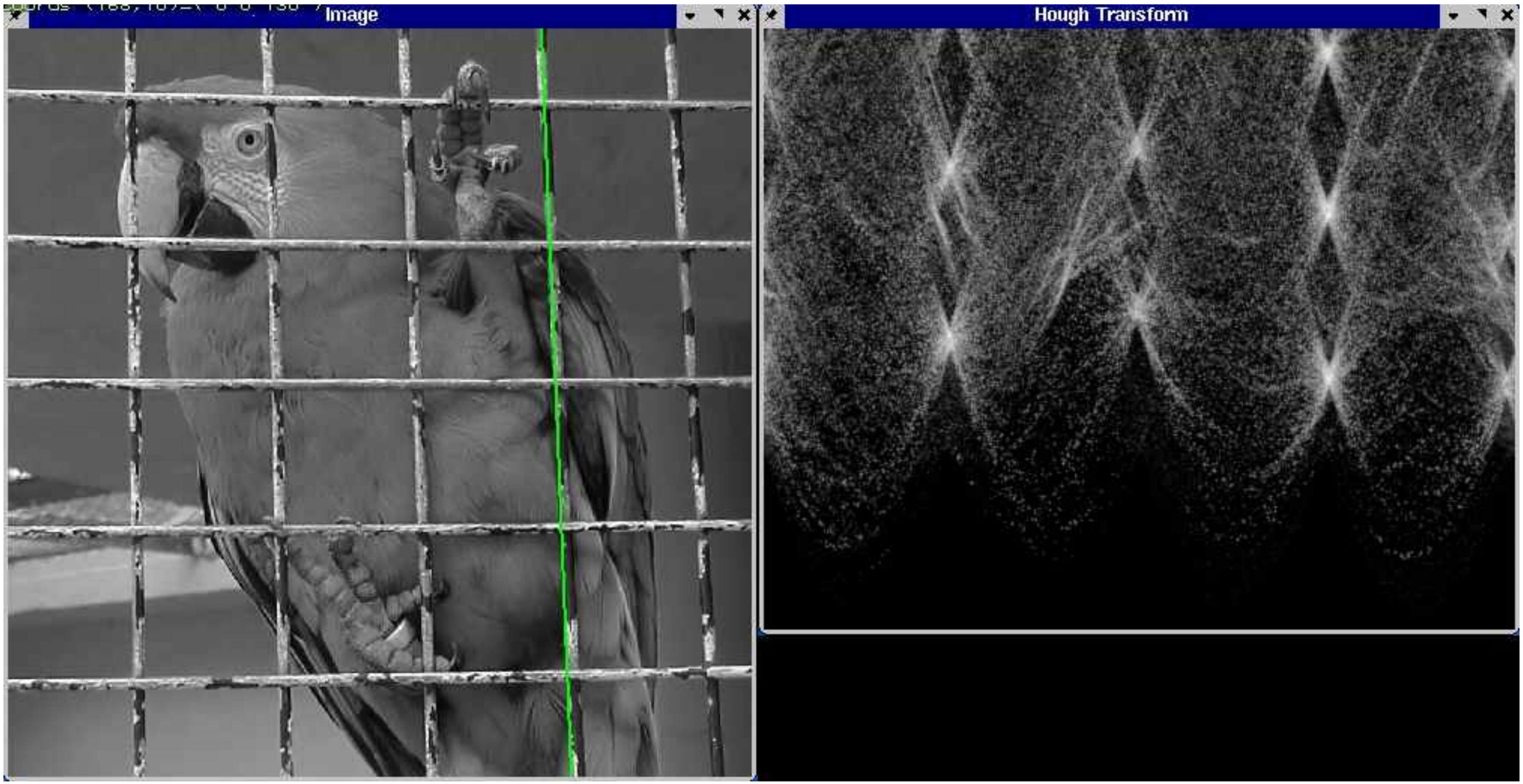
3. Hough votes → Edges

Find peaks and post-process



Slide from D. Hoiem

Hough transform example



Slide from D. Hoiem

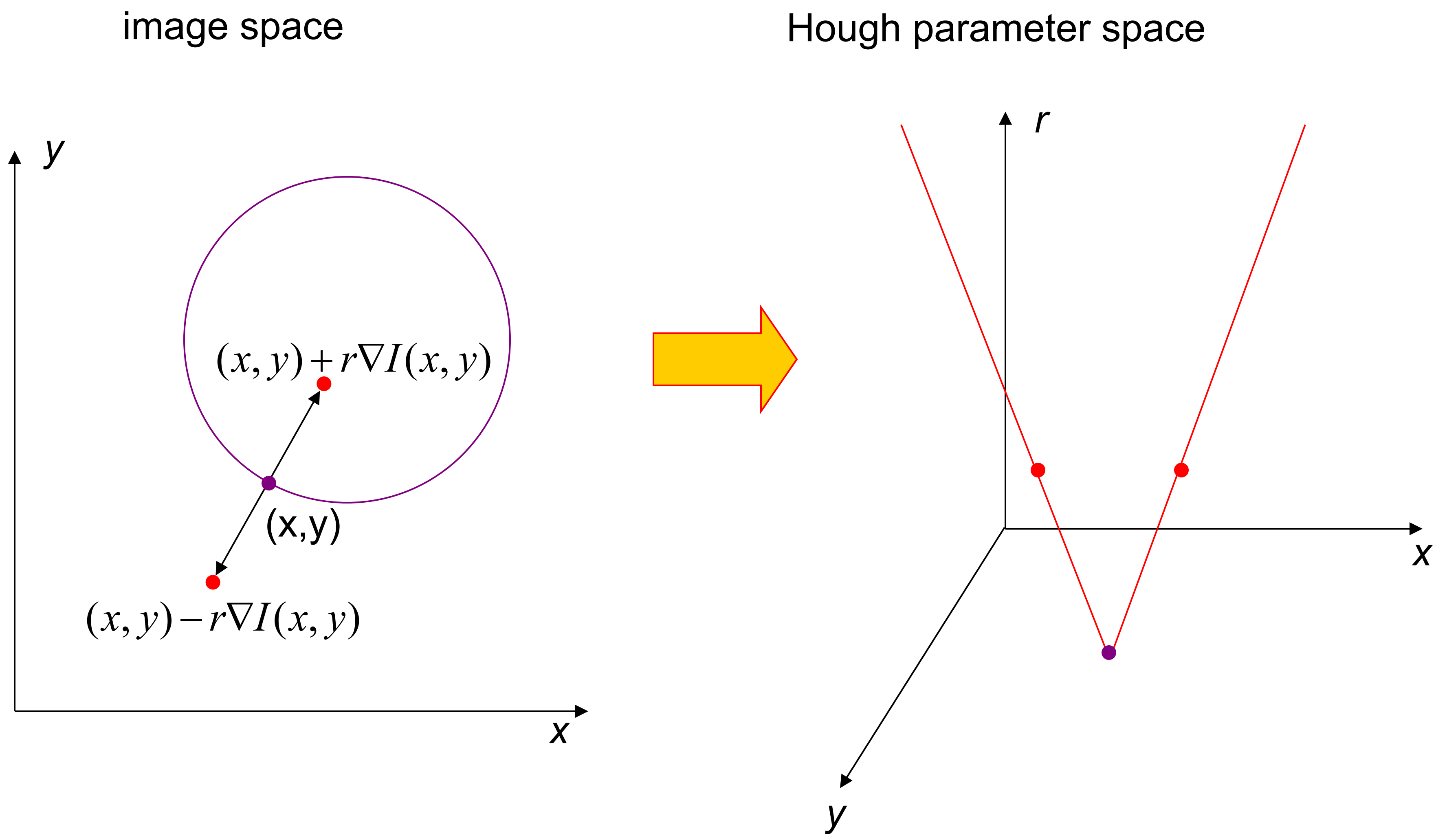
http://ostatic.com/files/images/ss_hough.jpg

Finding circles (x_0, y_0, r) using Hough transform

- Fixed r
- Variable r

Slide from D. Hoiem

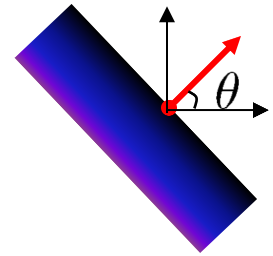
Hough transform for circles



Slide from L. Lazebnik

Incorporating image gradients

- When we detect an edge point, we also know its gradient orientation
- How does this constrain possible lines passing through the point?



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- Modified Hough transform:

- For each edge point (x,y)
  θ = gradient orientation at (x,y)
  ρ = x cos θ + y sin θ
  H(θ, ρ) = H(θ, ρ) + 1
  end

Slide from L. Lazebnik

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

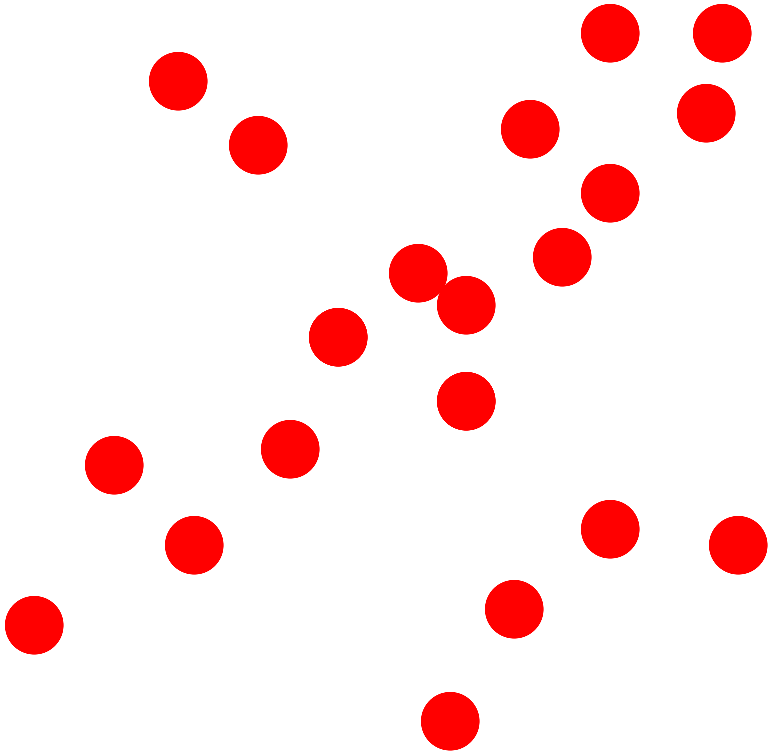
- Line fitting (also circles, ellipses, etc.)
- Object instance recognition (parameters are position/scale/orientation)
- Object category recognition (parameters are position/scale)

Slide from D. Hoiem

RANSAC

(RANdom SAmple Consensus) :

Fischler & Bolles in '81.



Algorithm:

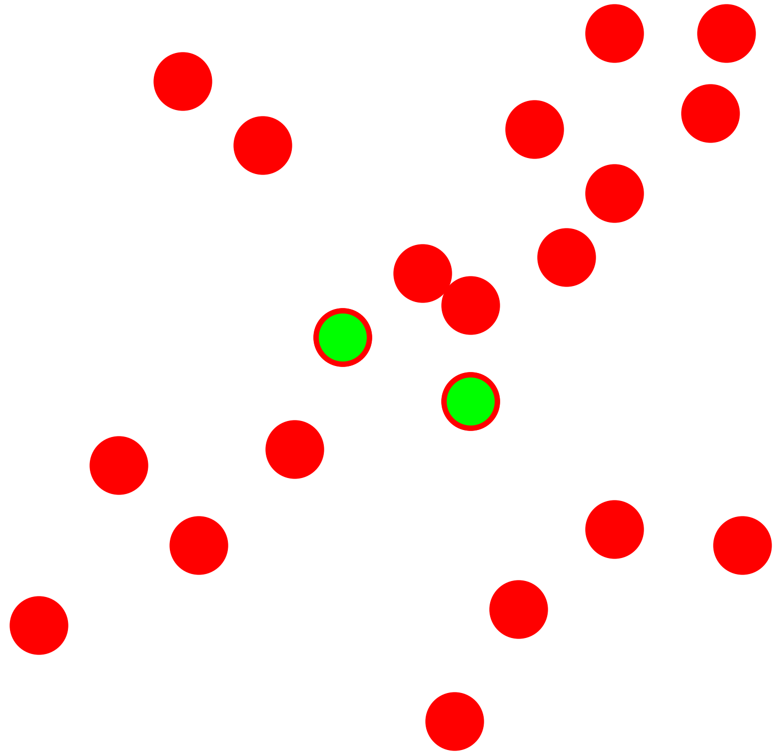
1. **Sample** (randomly) the number of points required to fit the model
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

RANSAC

Line fitting example



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

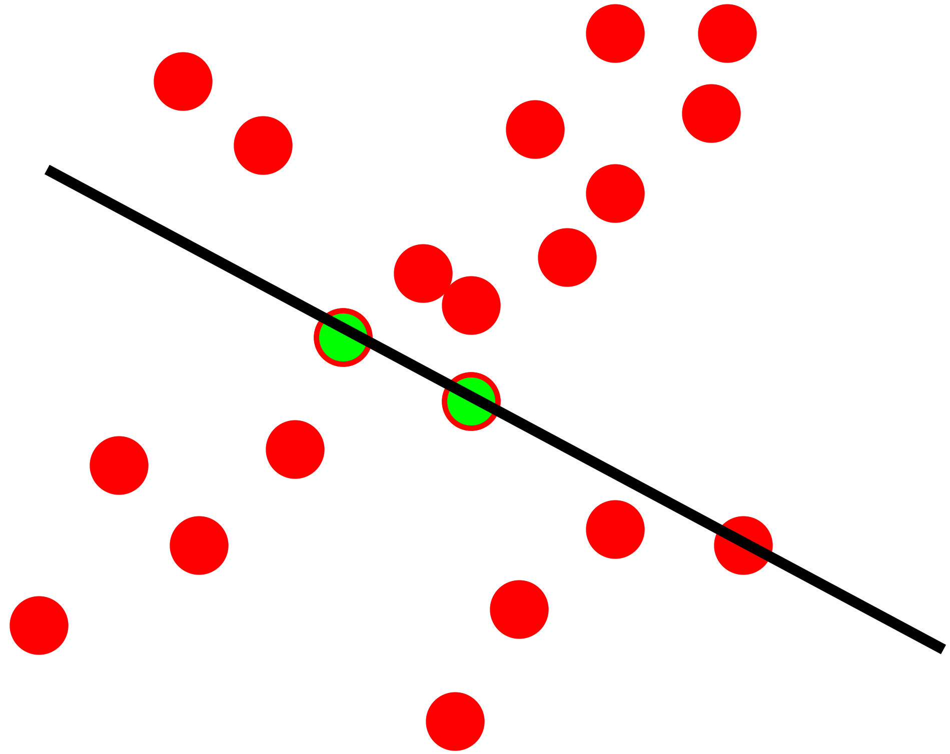
Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

Illustration by Savarese

RANSAC

Line fitting example



Algorithm:

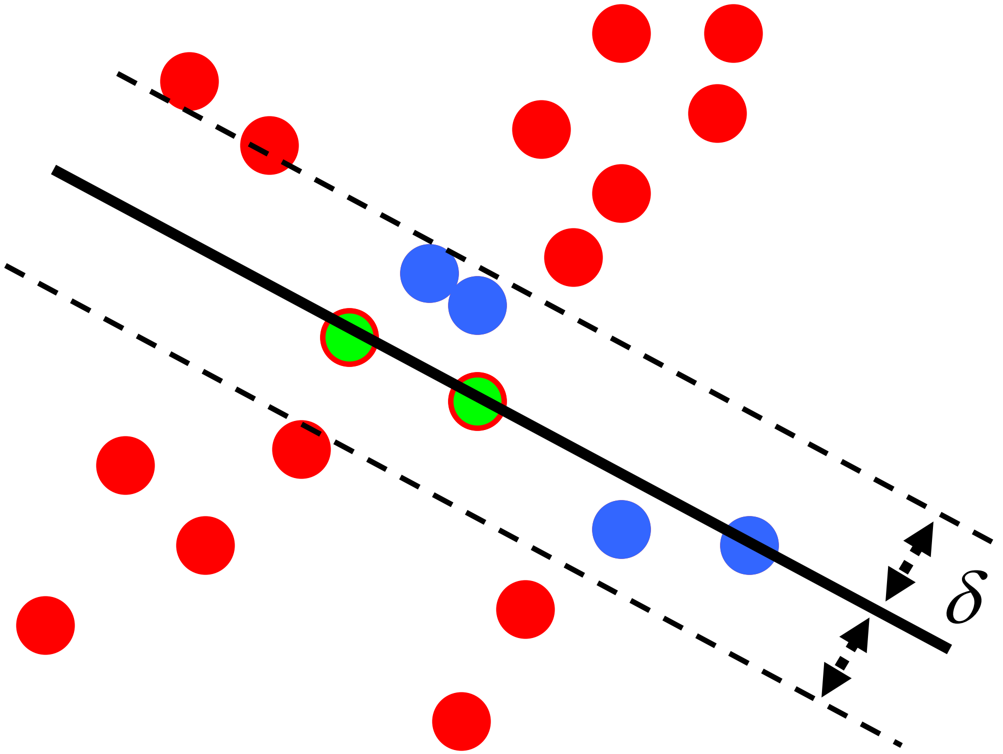
1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

RANSAC

Line fitting example



$$N_I = 6$$

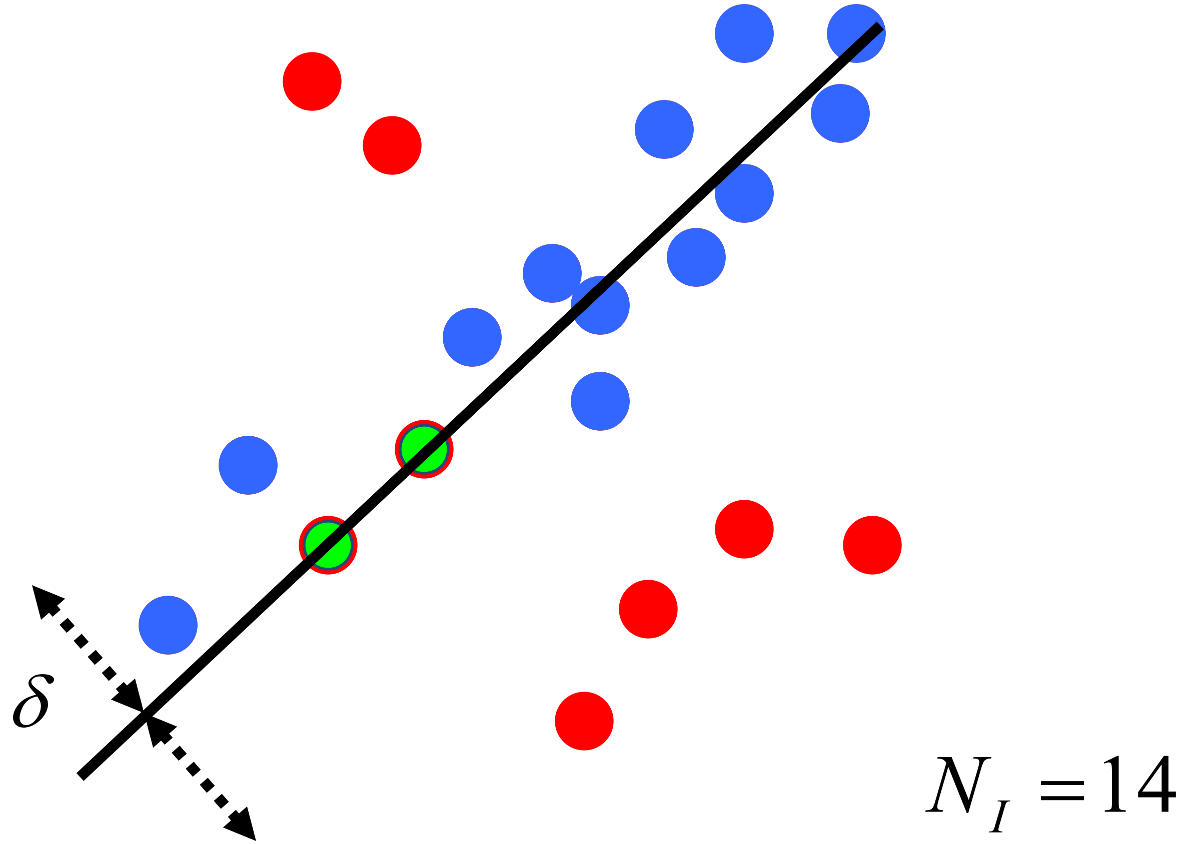
Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

RANSAC



Algorithm:

1. **Sample** (randomly) the number of points required to fit the model (#=2)
2. **Solve** for model parameters using samples
3. **Score** by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Slide from D. Hoiem

How to choose parameters?

- 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*)
- Number of sampled points *s*
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

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

	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

Slide from D. Hoiem

modified from M. Pollefeys

RANSAC conclusions

Good

- Robust to outliers
- Applicable for larger number of objective function parameters than Hough transform
- Optimization parameters are easier to choose than Hough transform

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not as good for getting multiple fits (though one solution is to remove inliers after each fit and repeat)

Common applications

- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

Slide from D. Hoiem

Fitting Summary

- Least Squares Fit
  - closed form solution
  - robust to noise
  - not robust to outliers
- Robust Least Squares
  - improves robustness to noise
  - requires iterative optimization
- Hough transform
  - robust to noise and outliers
  - can fit multiple models
  - only works for a few parameters (1-4 typically)
- RANSAC
  - robust to noise and outliers
  - works with a moderate number of parameters (e.g, 1-8)

Slide from D. Hoiem