EEE-6512: Image Processing and Computer Vision

November 13, 2018

Lecture #12: Model Fitting

Damon L. Woodard, Ph.D.
Dept. of Electrical and Computer Engineering
dwoodard@ece.ufl.edu

Fitting

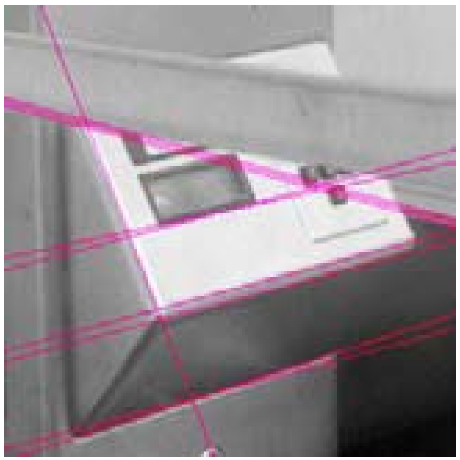
- We've learned how to detect edges, and corners. Now what?
- We would like to form a higher-level, more compact representation of the features in the image by grouping multiple features according to a simple model

Fitting: Main idea

- **Choose a parametric model to represent a set of features**
- **Membership criterion is not local**
  - Can’t tell whether a point belongs to a given model just by looking at that point
- **Three main questions:**
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- **Computational complexity is important**
  - It is infeasible to examine every possible set of parameters and every possible combination of features

Fitting

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



simple model: lines

simple model: circles

complicated model: face shape

complicated model: car

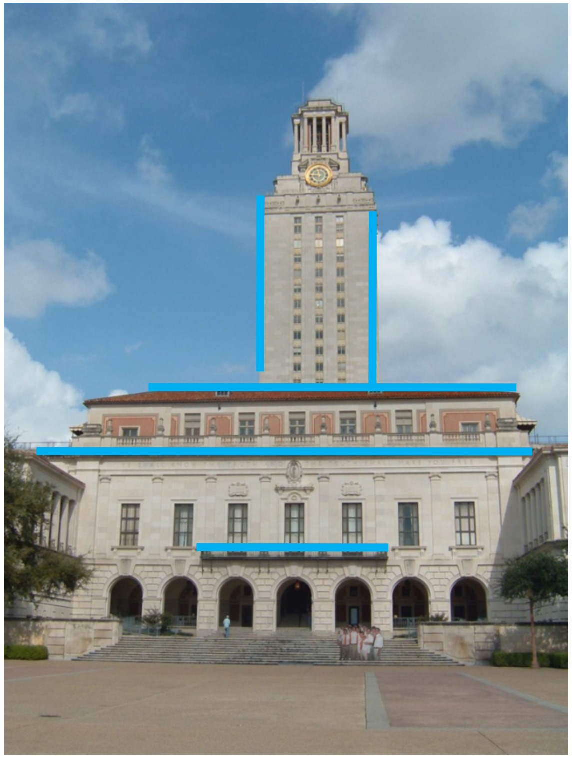
Fitting -Design challenges

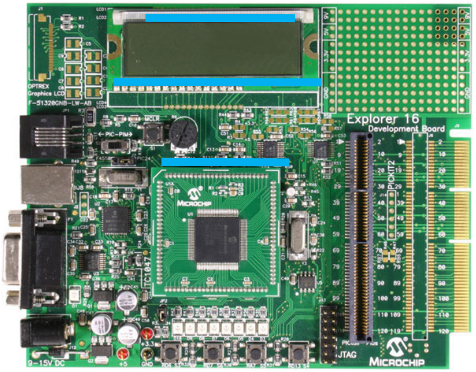
- Design a suitable **goodness of fit** measure
  - Similarity should reflect application goals
  - Encode robustness to outliers and noise

- Design an **optimization** method
  - Avoid local optima
  - Find best parameters quickly

Example: Line fitting

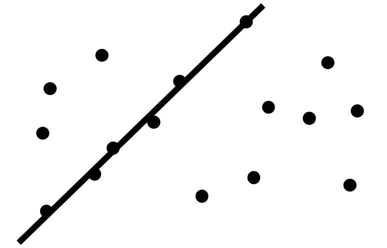
- Why fit lines?
  Many objects characterized by presence of straight lines

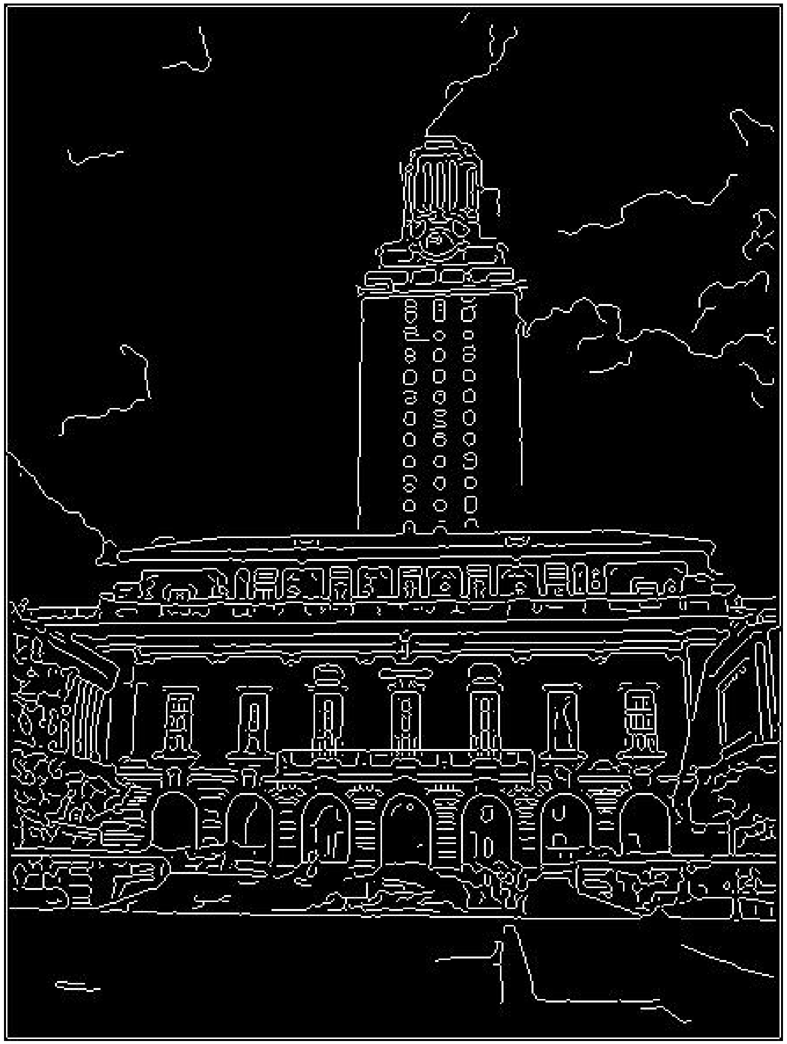




- Wait, why aren't we done just by running edge detection?

Difficulty of line fitting





- **Extra** edge points (clutter), multiple models:
  - which points go with which line, if any?
- Only some parts of each line detected, and some parts are **missing:**
  - how to find a line that bridges missing evidence?
- **Noise** in measured edge points, orientations:
  - how to detect true underlying parameters?

Fitting: Overview (cont.)

- **If we know which points belong to the line, how do we find the "optimal" line parameters?**
  Least squares

- **What if there are outliers?**
  RANSAC

- **What if there are many lines?**
  Voting methods: RANSAC, Hough transform

Line Fitting

Fitting: Methods

- Global optimization / Search for parameters
  - Ordinary Least Squares
  - Total Least Squares

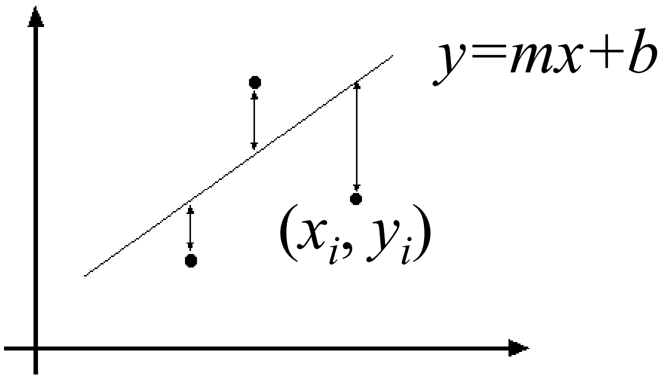
- Hypothesize and test
  - Hough transform
  - RANSAC

Ordinary Least Squares / Total Least Squares

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

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



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

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

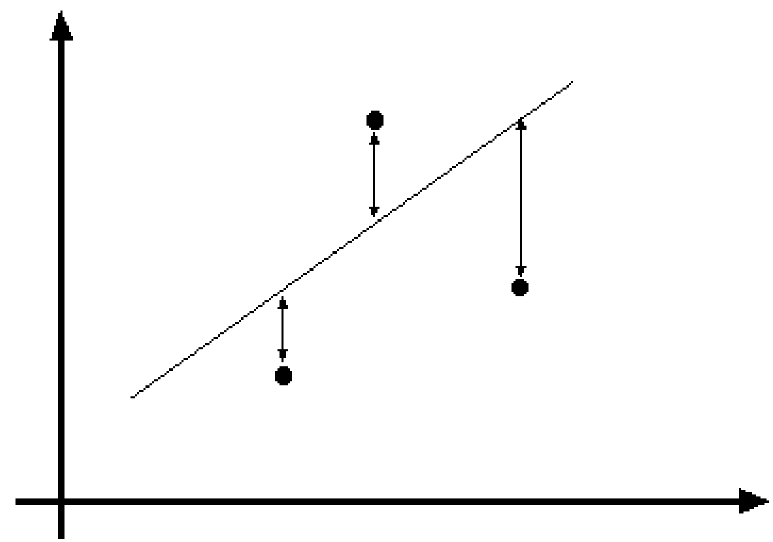
$$\frac{dE}{dp} = 2\mathbf{A}^T \mathbf{Ap} - 2\mathbf{A}^T \mathbf{y} = 0$$

Matlab: `p = A \ y;`

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

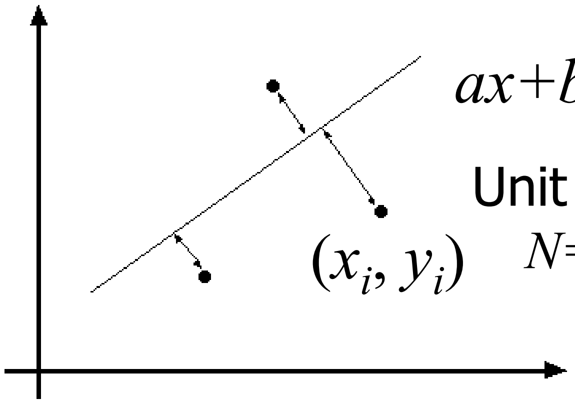
Problem with "vertical" least squares

- Not rotation-invariant
- Fails completely for vertical lines



Total least squares

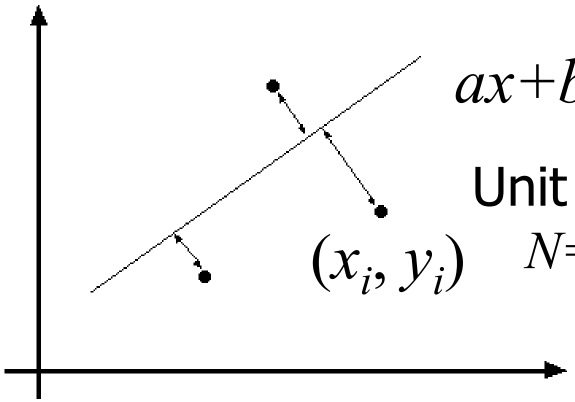
- Distance between point (x_n, y_n) and line $ax+by=d$ ($a^2+b^2=1$): $|ax + by - d|$
- Find (a, b, d) to minimize the sum of squared perpendicular distances



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

Total least squares

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



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

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

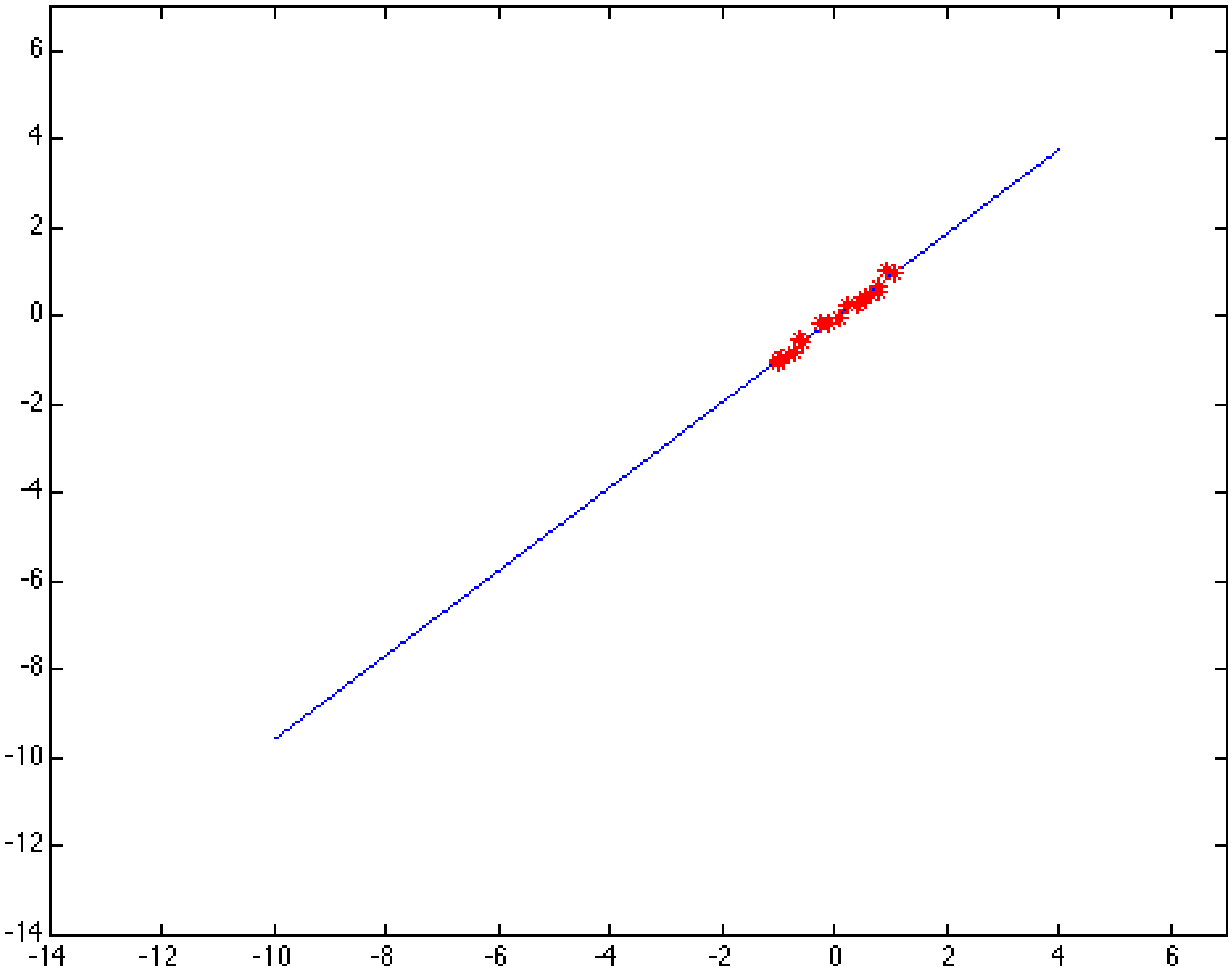
$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = (UN)^T (UN)$$

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

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* $UN = 0$)

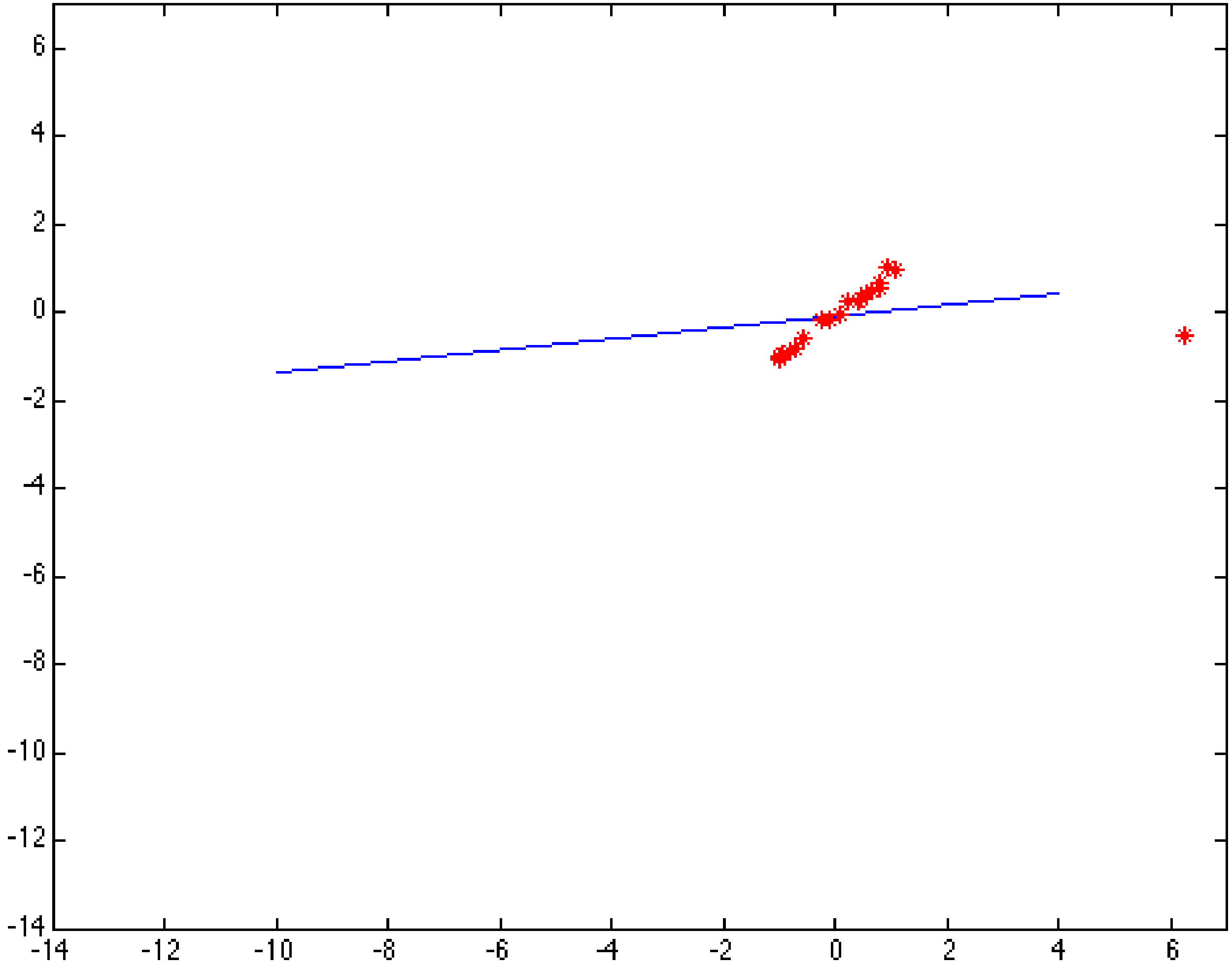
Least squares: Robustness to noise

Least squares fit to the red points:



Least squares: Robustness to noise

Least squares fit with an outlier:



Problem: squared error heavily penalizes outliers

Least squares (global) optimization

Good

- Clearly specified objective
- Optimization is easy

Bad

- Sensitive to outliers
  - Bad matches, extra points
- Doesn't allow you to get multiple good fits
  - Detecting multiple objects, lines, etc.

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

Hypothesize and Test/Hough Transform

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**

Hough Transform: Outline

1. Create a grid of parameter values

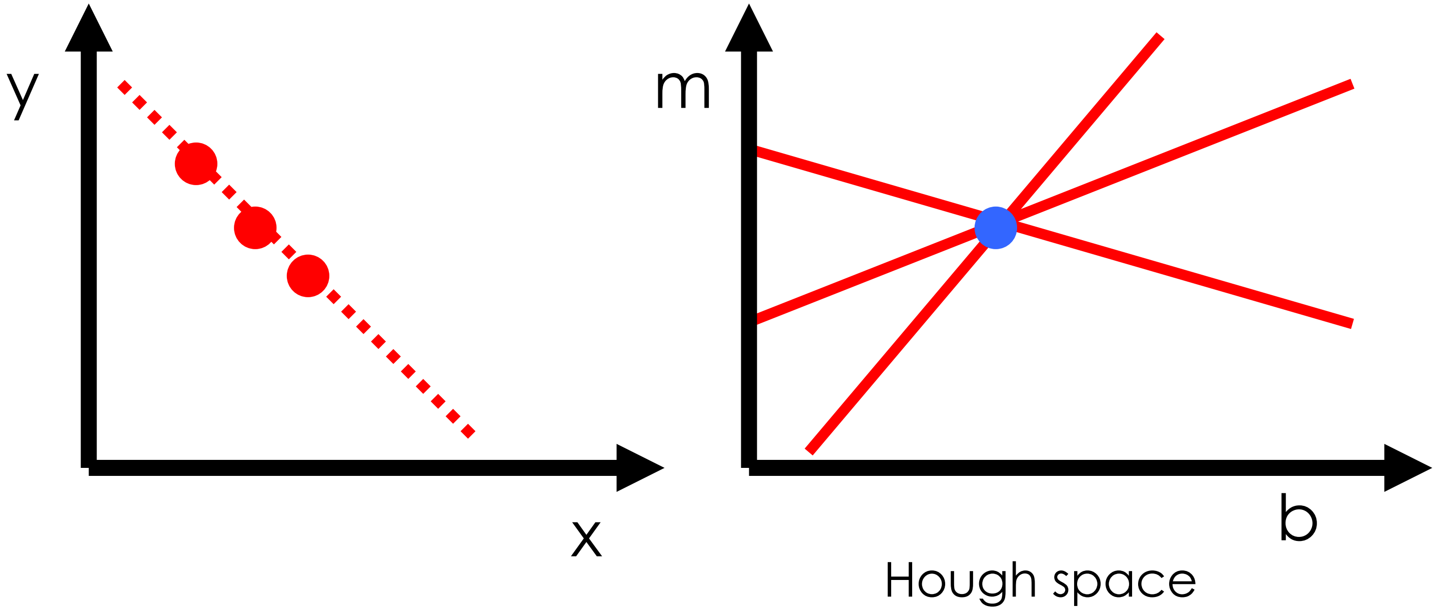
2. Each point votes for a set of parameters, incrementing those values in grid

3. Find maximum or local maxima in grid

Connection Between Image and Hough Spaces

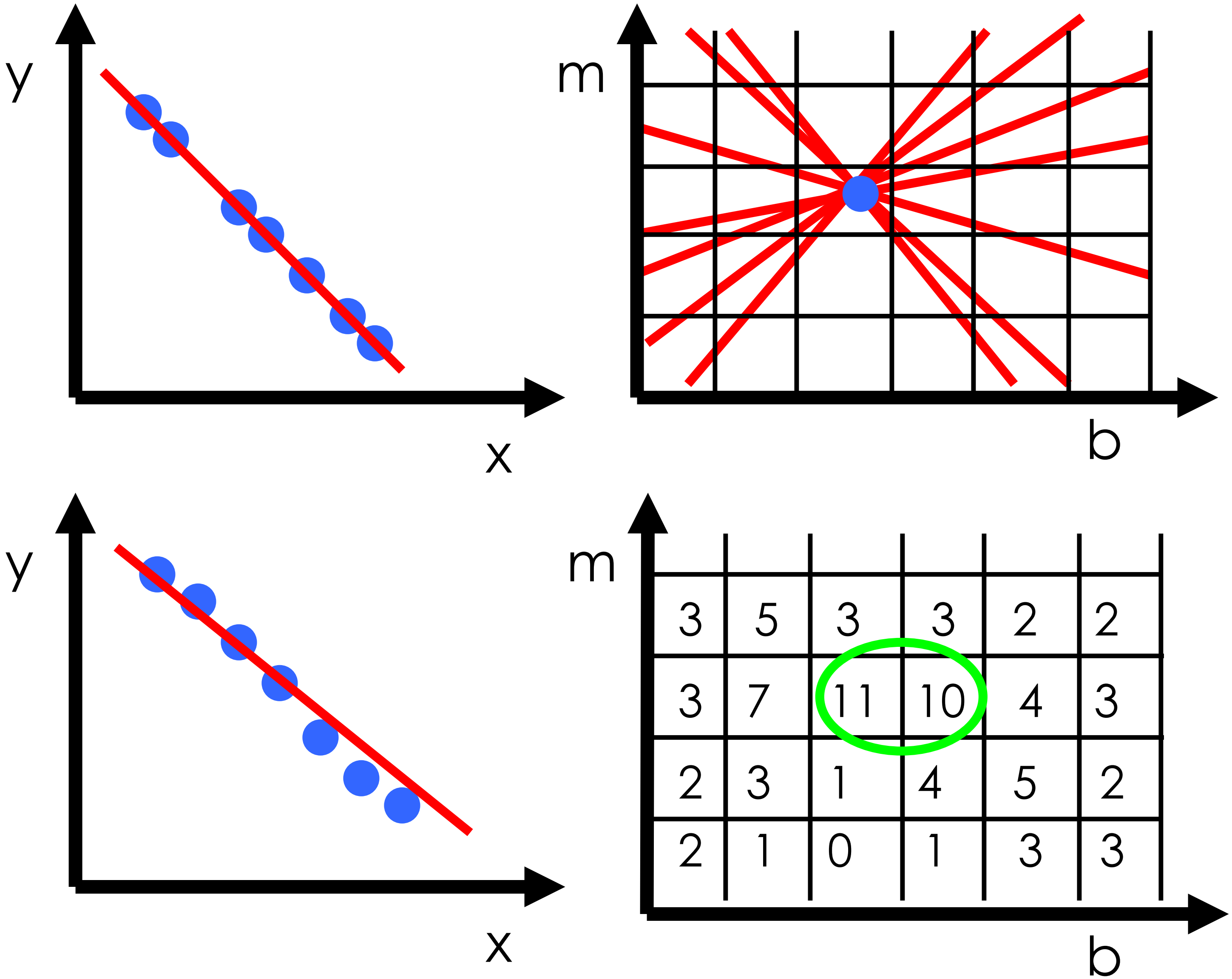
Hough transform

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



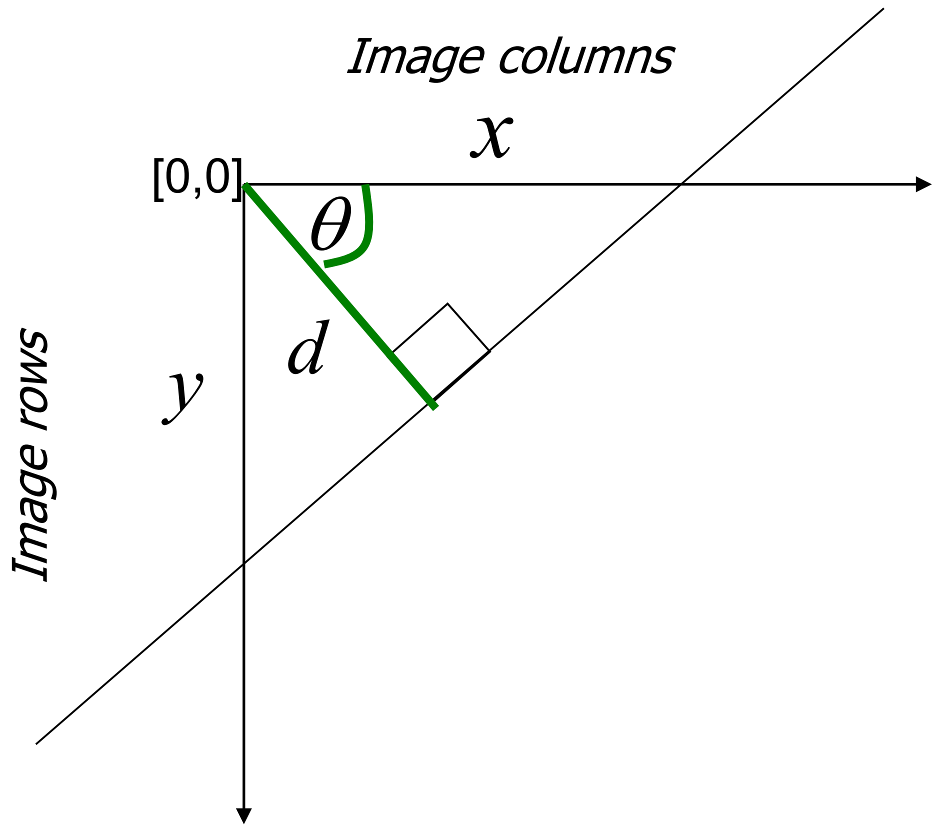
$$y = m\,x + b$$

Hough transform



Polar representation for lines

Issues with usual (*m,b*) parameter space: can take on infinite values, undefined for vertical lines.



d : perpendicular distance from line to origin

θ : angle the perpendicular makes with the x-axis

$$x\cos\theta - y\sin\theta = d$$

Point in image space → sinusoid segment in Hough space

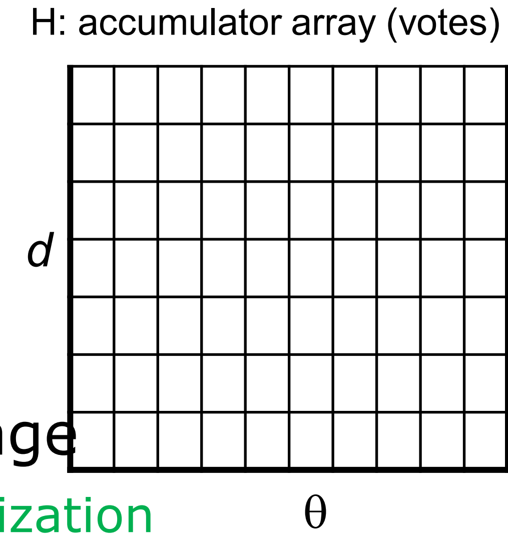
Image Parameter Spaces

Image Space	Parameter Space
Lines	Points
Points	Lines/Sinusoid
Collinear Points	Intersecting Lines

Hough transform algorithm

Using the polar parameterization:

$$x\cos\theta - y\sin\theta = d$$



Basic Hough transform algorithm

1. Initialize $H[d, \theta]=0$
2. for each edge point $I[x,y]$ in the image

   for $\theta = [\theta_{min}$ to $\theta_{max}]$ // some quantization

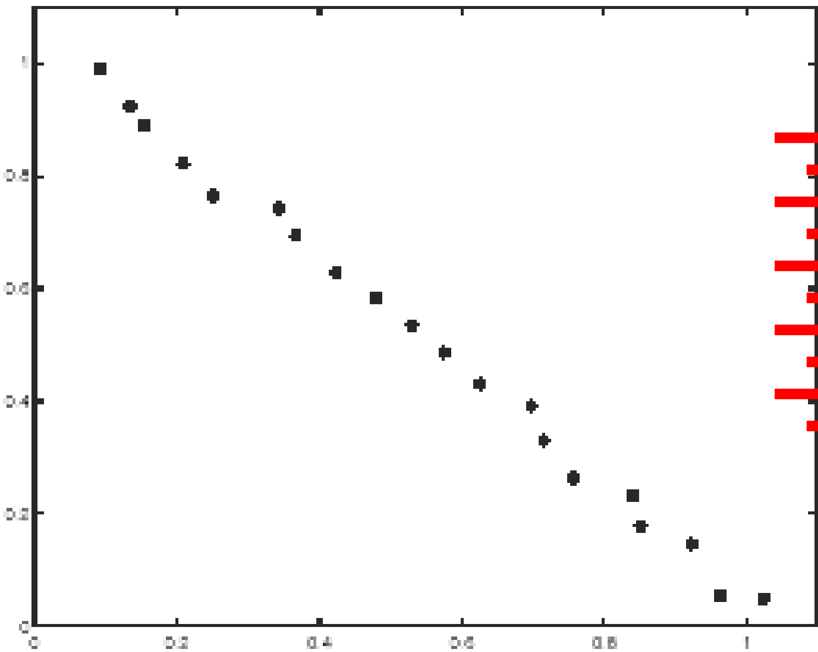
   $$d = x\cos\theta - y\sin\theta$$

   $H[d, \theta]$ += 1
3. Find the value(s) of (d, θ) where $H[d, \theta]$ is maximum
4. The detected line in the image is given by

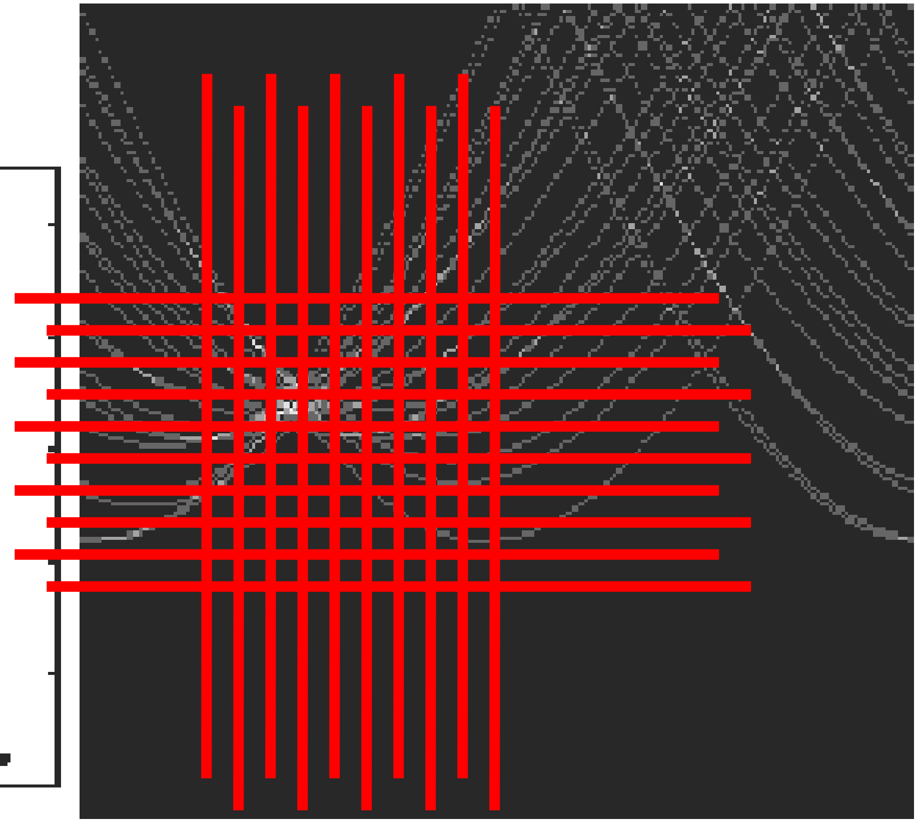
$$d = x\cos\theta - y\sin\theta$$

Hough transform - experiments

Noisy data



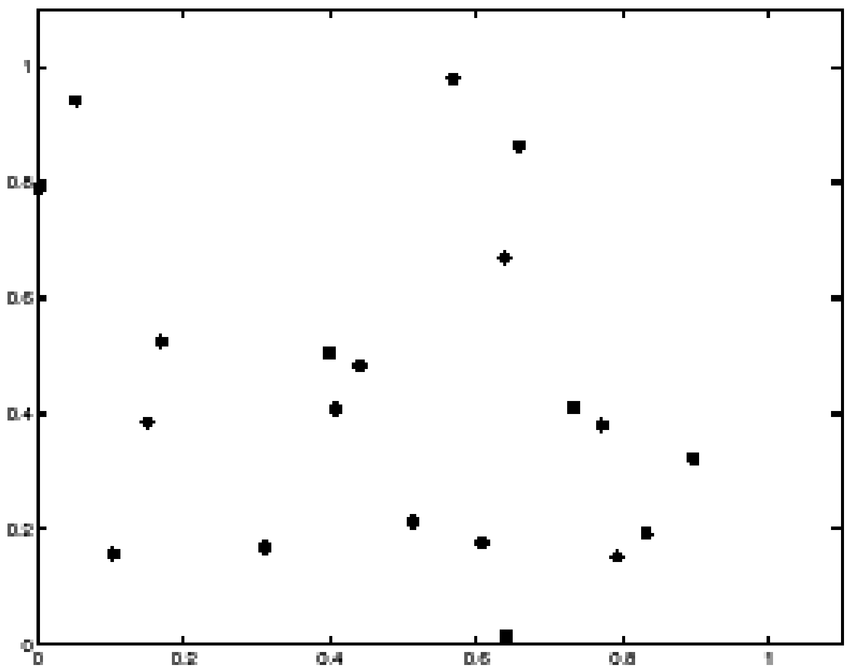
features



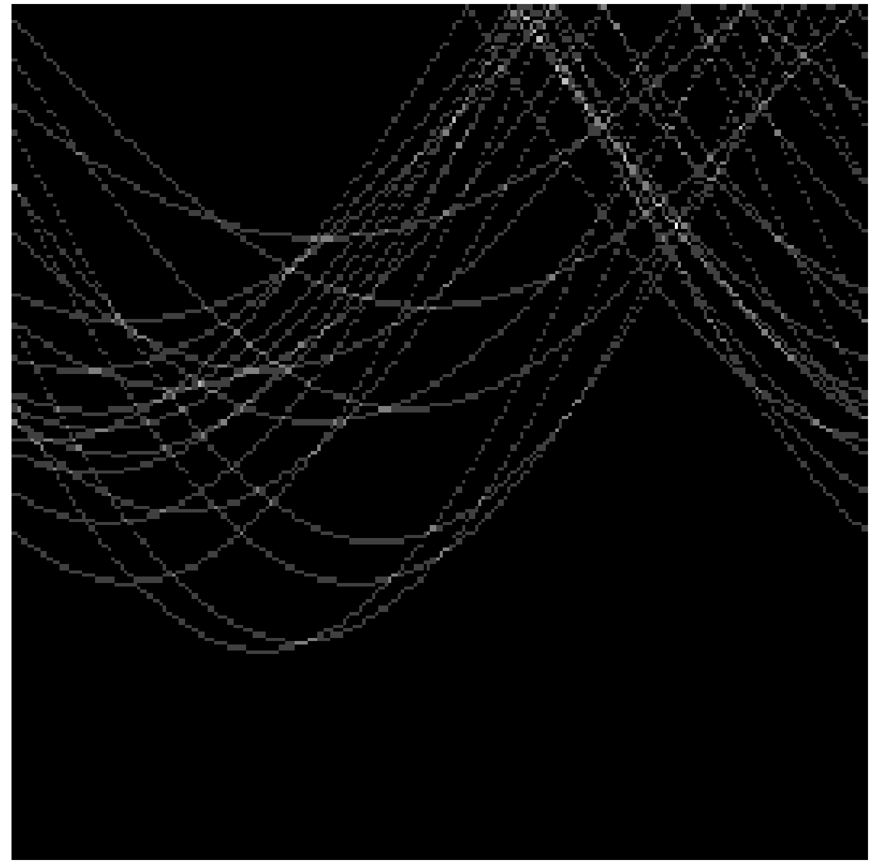
votes

Need to adjust grid size or smooth

Hough transform - experiments



features



votes

Issue: spurious peaks due to uniform noise

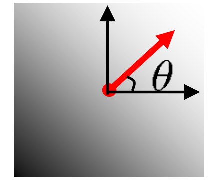
Dealing with noise

- **Choose a good grid / discretization**
  - Too coarse: large votes obtained when too many different lines correspond to a single bucket
  - Too fine: miss lines because some points that are not exactly collinear cast votes for different buckets
- **Increment neighboring bins (smoothing in accumulator array)**
- **Try to get rid of irrelevant features**
  - Take only edge points with significant gradient magnitude

Hough Algorithm Extensions

Extensions

Extension 1: Use the image gradient



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

1. same
2. for each edge point I[x,y] in the image

   θ = gradient at (x,y)

   $$d = x\cos\theta - y\sin\theta$$

   H[d, θ] += 1

3. same
4. same

(Reduces degrees of freedom)

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

Extension 2

- give more votes for stronger edges

Extension 3

- change the sampling of (d, θ) to give more/less resolution

Extension 4

- The same procedure can be used with circles, squares, or any other shape

Hough Transform for Circles

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For a fixed radius r, unknown gradient direction

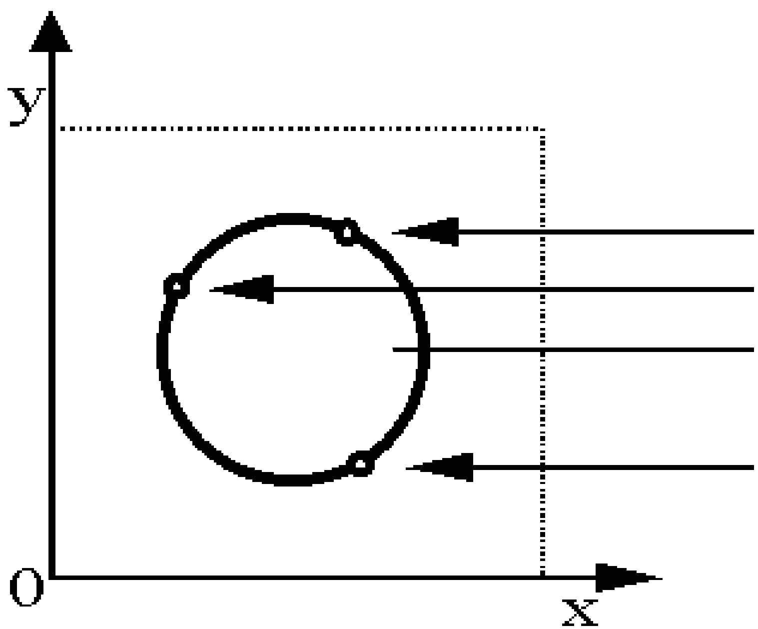
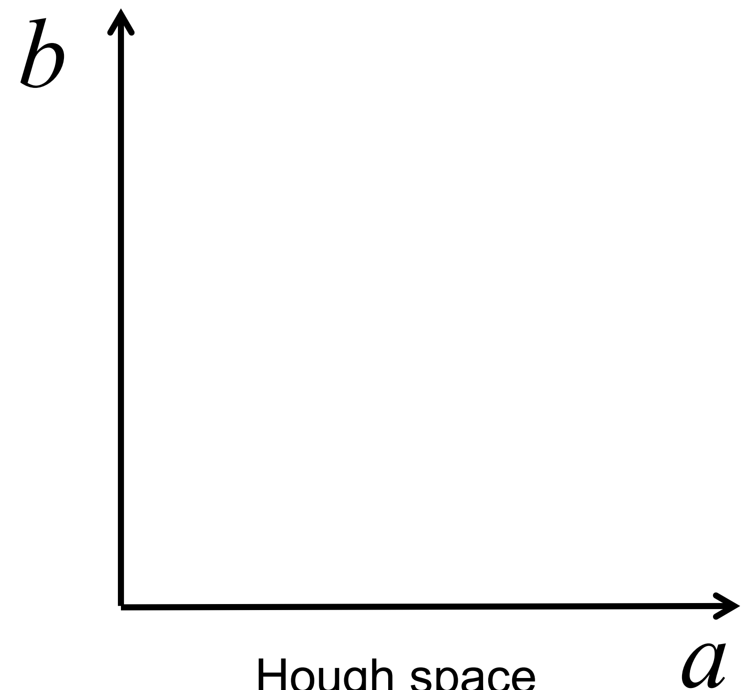


Image space



Hough space

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For a fixed radius r, unknown gradient direction

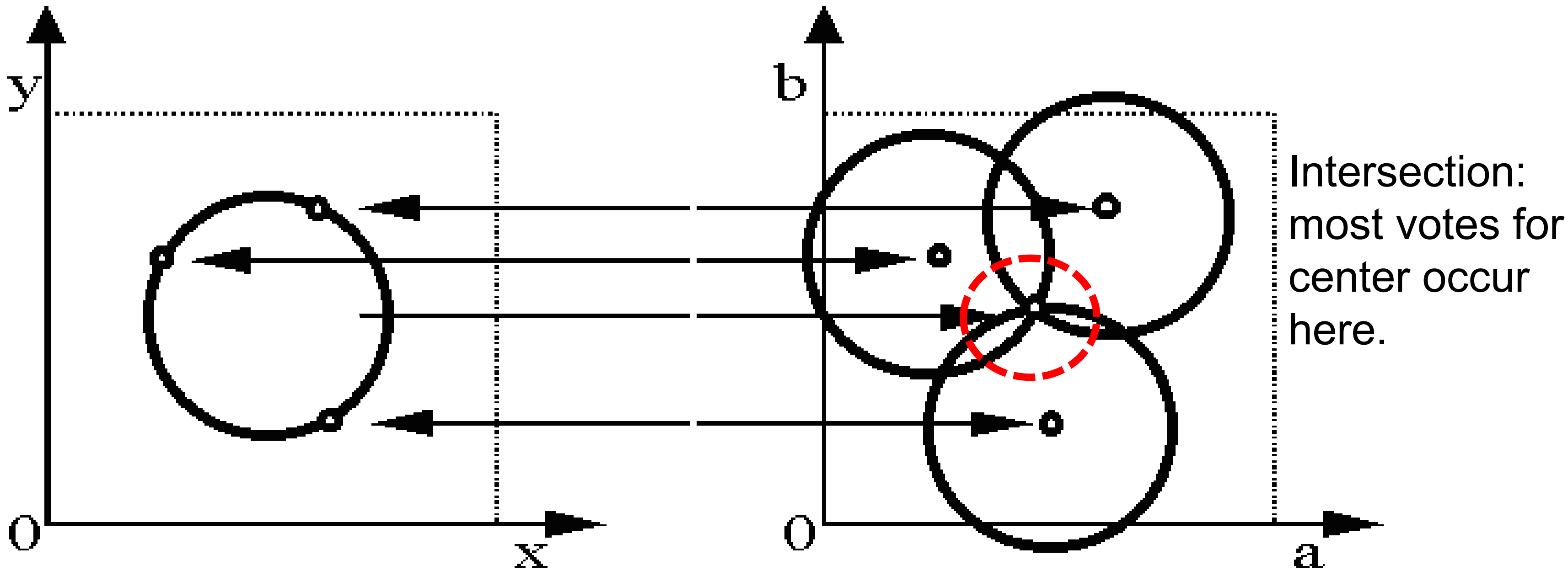


Image space

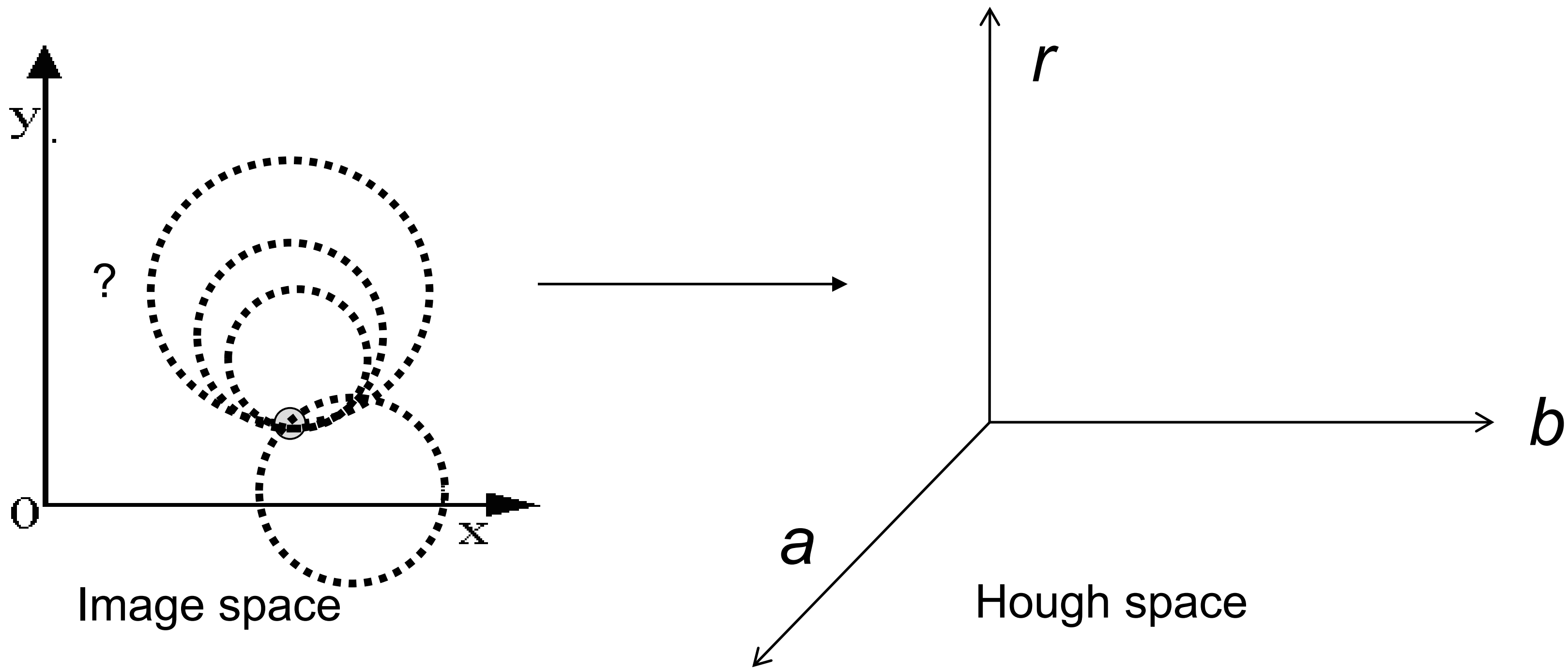
Hough space

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, unknown gradient direction



Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, unknown gradient direction

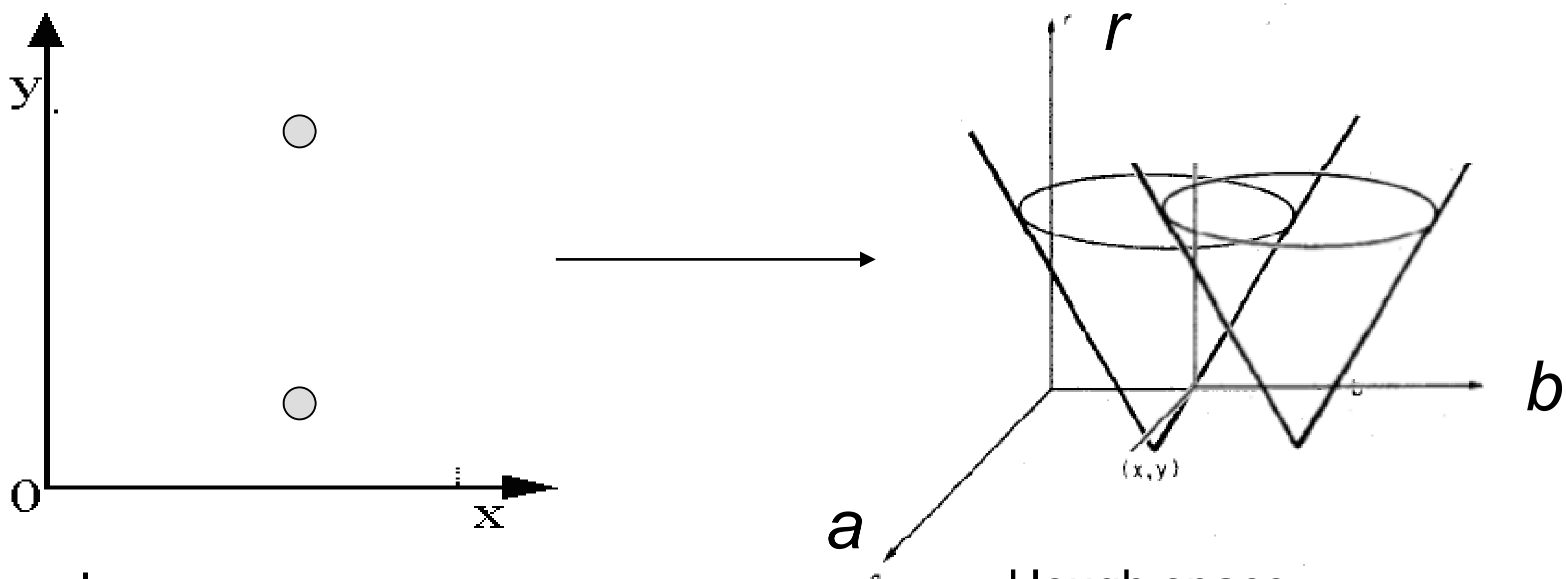


Image space

Hough space

Hough transform for circles

- Circle: center (a,b) and radius r

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

- For an unknown radius r, **known** gradient direction

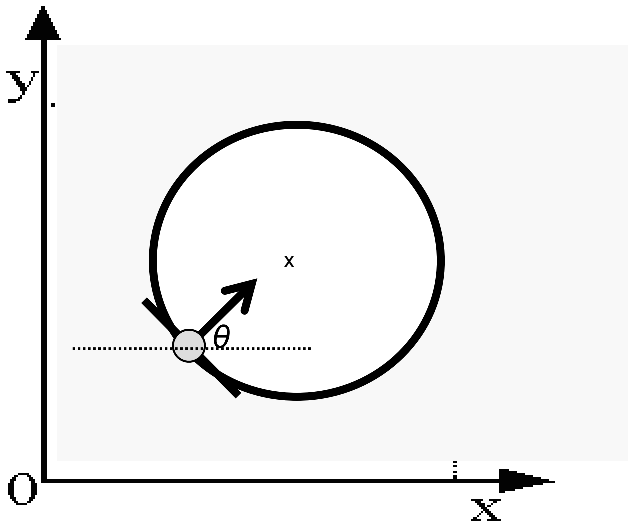
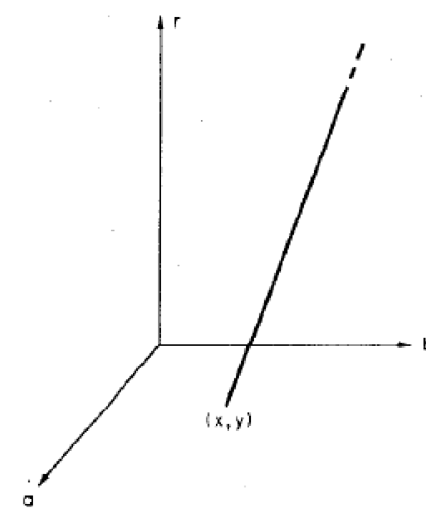


Image space



Hough space

Hough transform for circles

For every edge pixel (x,y) :

  For each possible radius value r:

    For each possible gradient direction θ:

      $a = x - r\cos(\theta)$ // column

      $b = y - r\sin(\theta)$ // row

      H[a,b,r] += 1

  end

end

Optimization of Circle Hough Transform

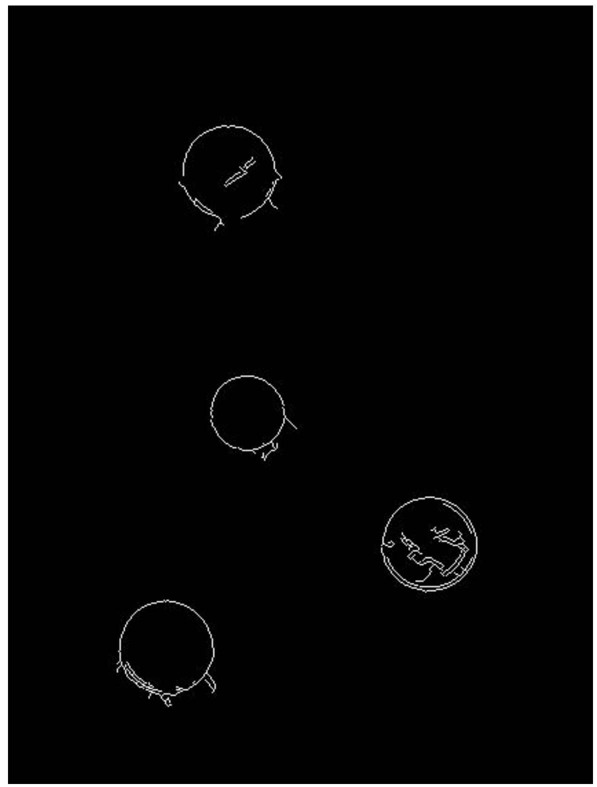
Use Edge Direction (Eliminates Radius)

$$b = a * \tan(\theta) - x * \tan(\theta) + y$$

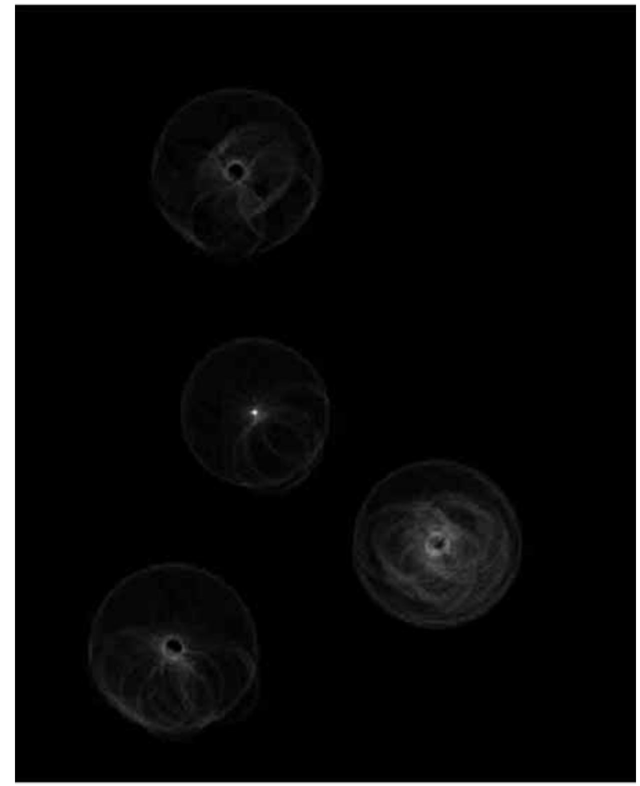
Example: detecting circles with Hough

Original

Edges



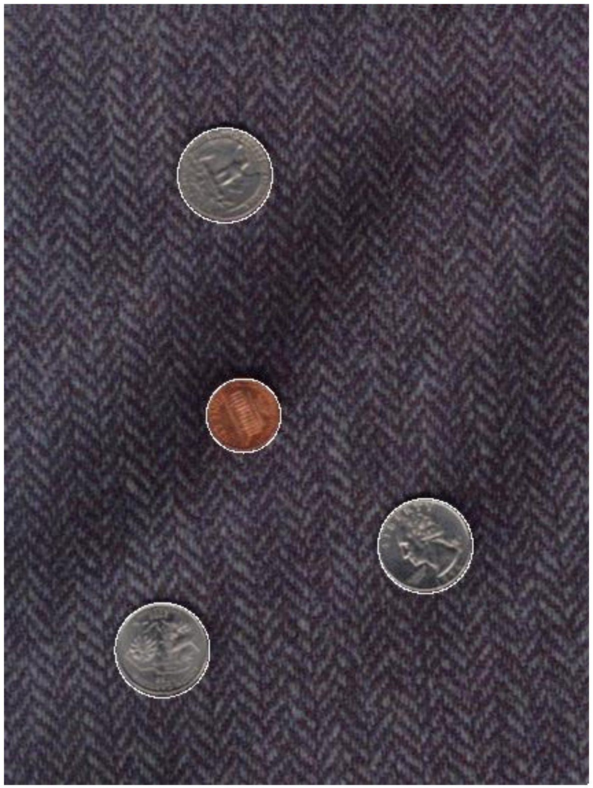
Votes: Penny



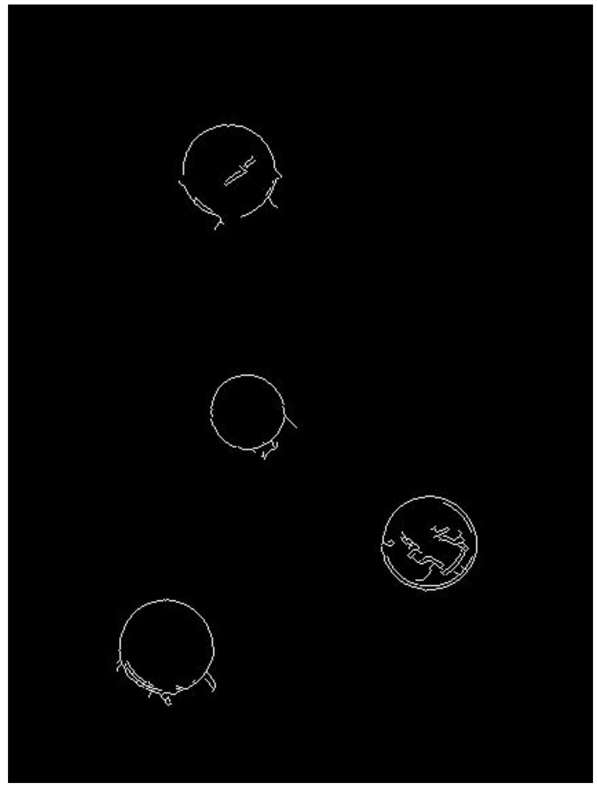
Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Example: detecting circles with Hough

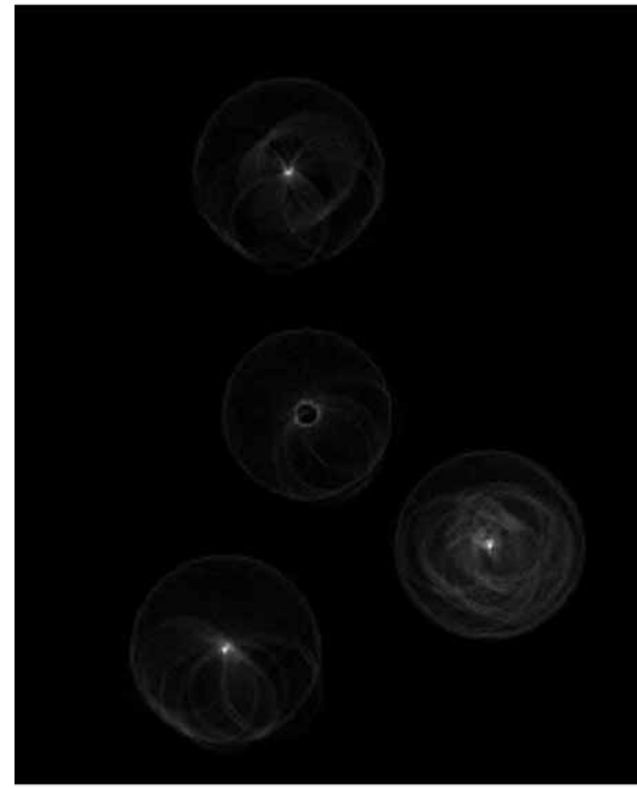
Original



Edges



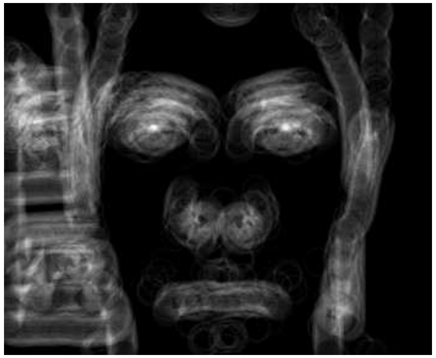
Votes: Quarter



Combined detections

Example: iris detection

Gradient+threshold



Hough space
(fixed radius)

Max detections

Voting: practical tips

- Minimize irrelevant tokens first
- Choose a good grid / discretization

Too fine ? Too coarse

- Vote for neighbors, also (smoothing in accumulator array)
- Use direction of edge to reduce parameters by 1
- To read back which points voted for “winning” peaks, keep tags on the votes.

Parameters for analytic curves

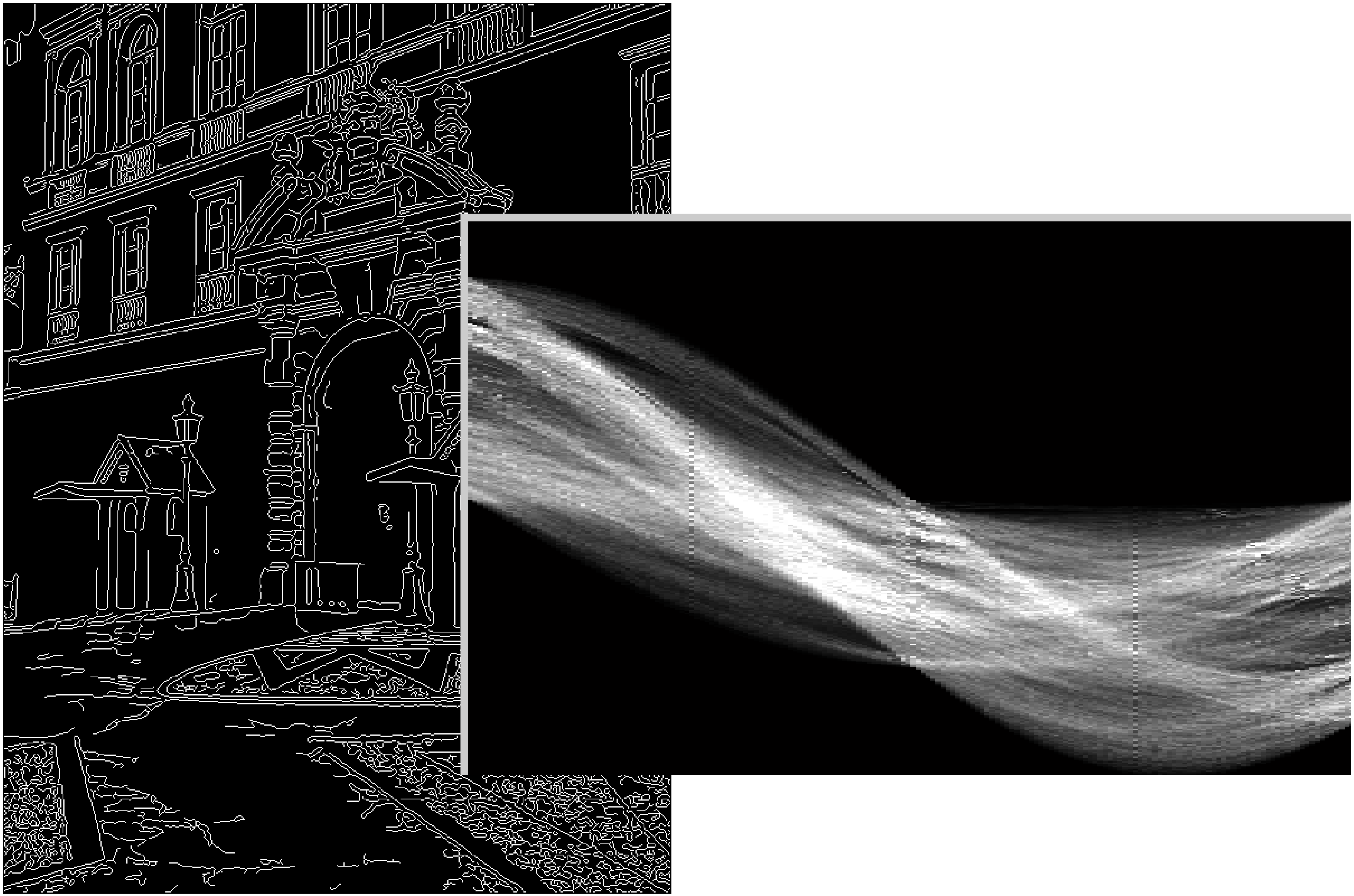
Analytic Form	Parameters	Equation
Line	ρ, θ	$x\cos\theta + y\sin\theta = \rho$
Circle	x_0, y_0, ρ	$(x-x_o)^2+(y-y_0)^2=r^2$
Parabola	x_0, y_0, ρ, θ	$(y-y_0)^2=4\rho(x-x_o)$
Ellipse	x_0, y_0, a, b, θ	$(x-x_o)^2/a^2+(y-y_0)^2/b^2=1$

Speed of Hough Transform

The computation time grows exponentially as the number of parameters increases

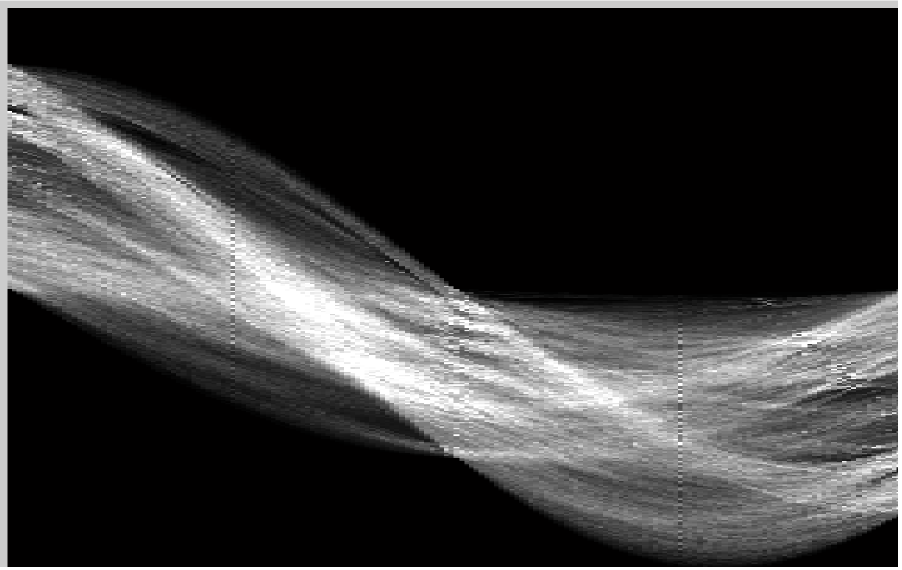
1. Image → Canny

2. Canny → Hough votes



3. Hough votes → Edges

Find peaks and post-process



Hough transform: pros and cons

Pros

- All points are processed independently, so can cope with occlusion, gaps
- Some robustness to noise: noise points unlikely to contribute *consistently* to any single bin
- Can detect multiple instances of a model in a single pass

Cons

- Complexity of search time increases exponentially with the number of model parameters
- Non-target shapes can produce spurious peaks in parameter space
- Quantization: can be tricky to pick a good grid size

Addressing Noise: RANSAC

RANSAC

- RANdom SAmple Consensus

- Approach: we want to avoid the impact of outliers, so let’s look for “inliers”, and use only those.

- 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

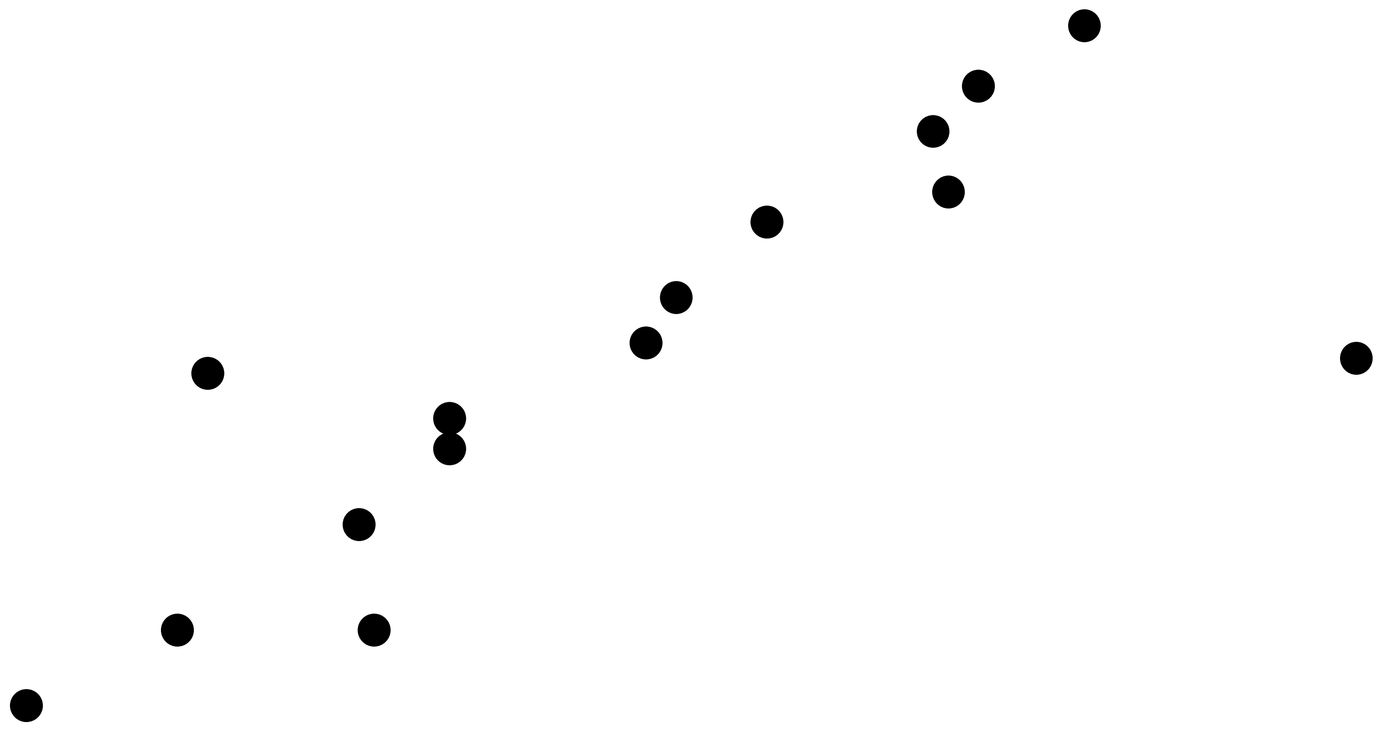
RANSAC loop:

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

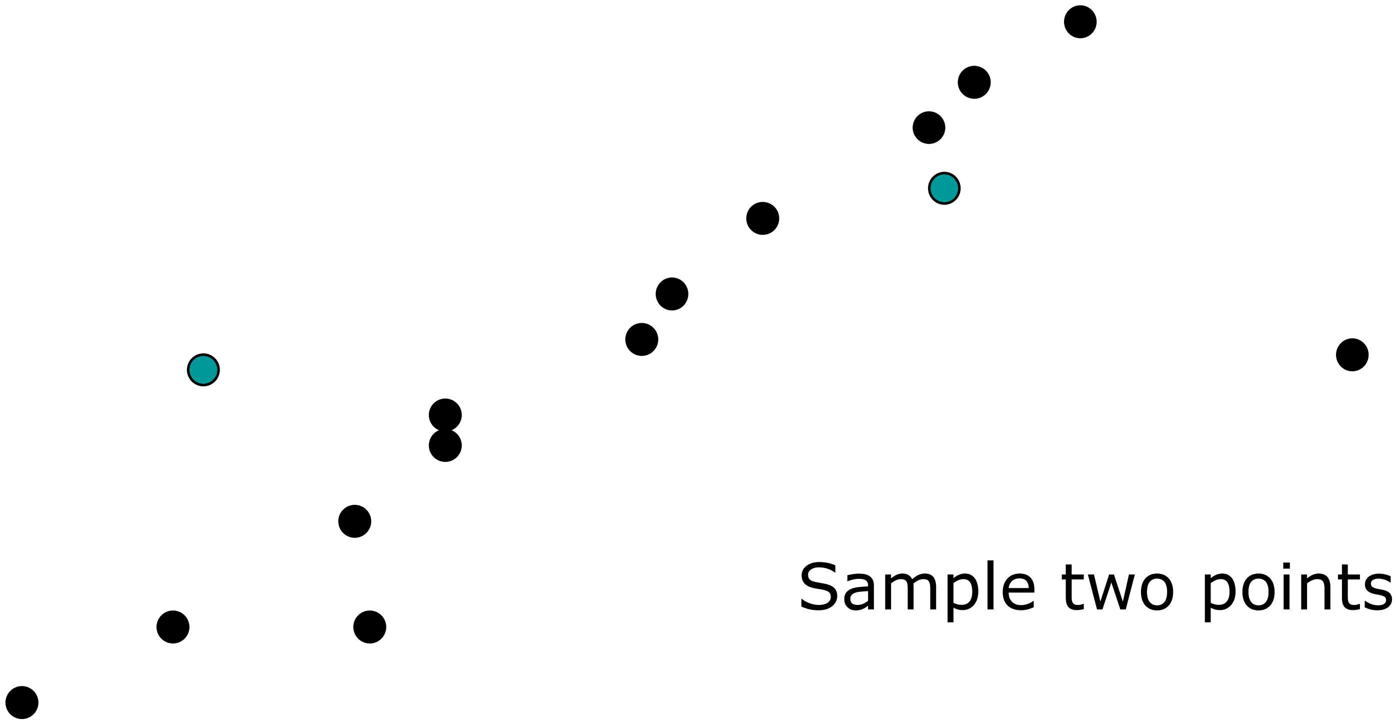
RANSAC Line Fitting Example

- Task: Estimate the best line
  - *How many points do we need to estimate the line?*



RANSAC Line Fitting Example

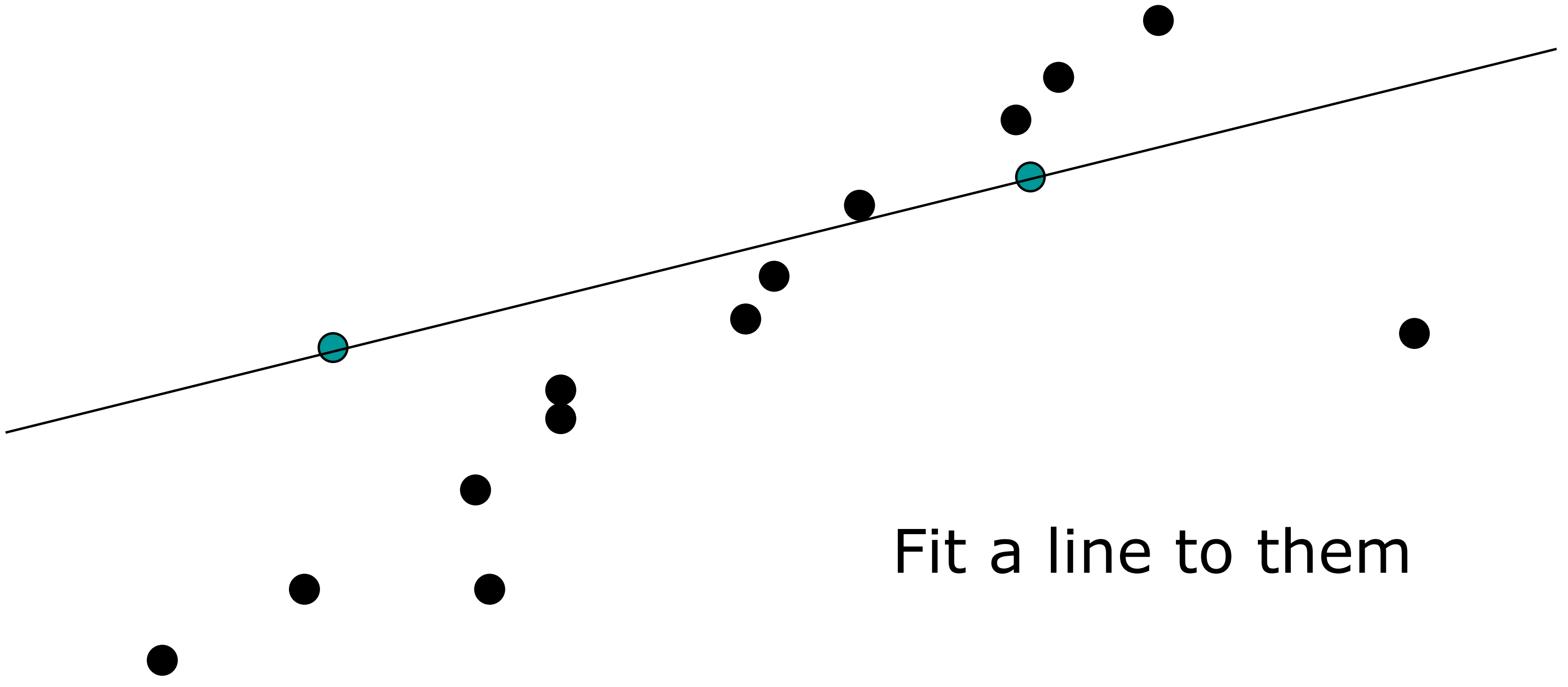
- Task: Estimate the best line



Sample two points

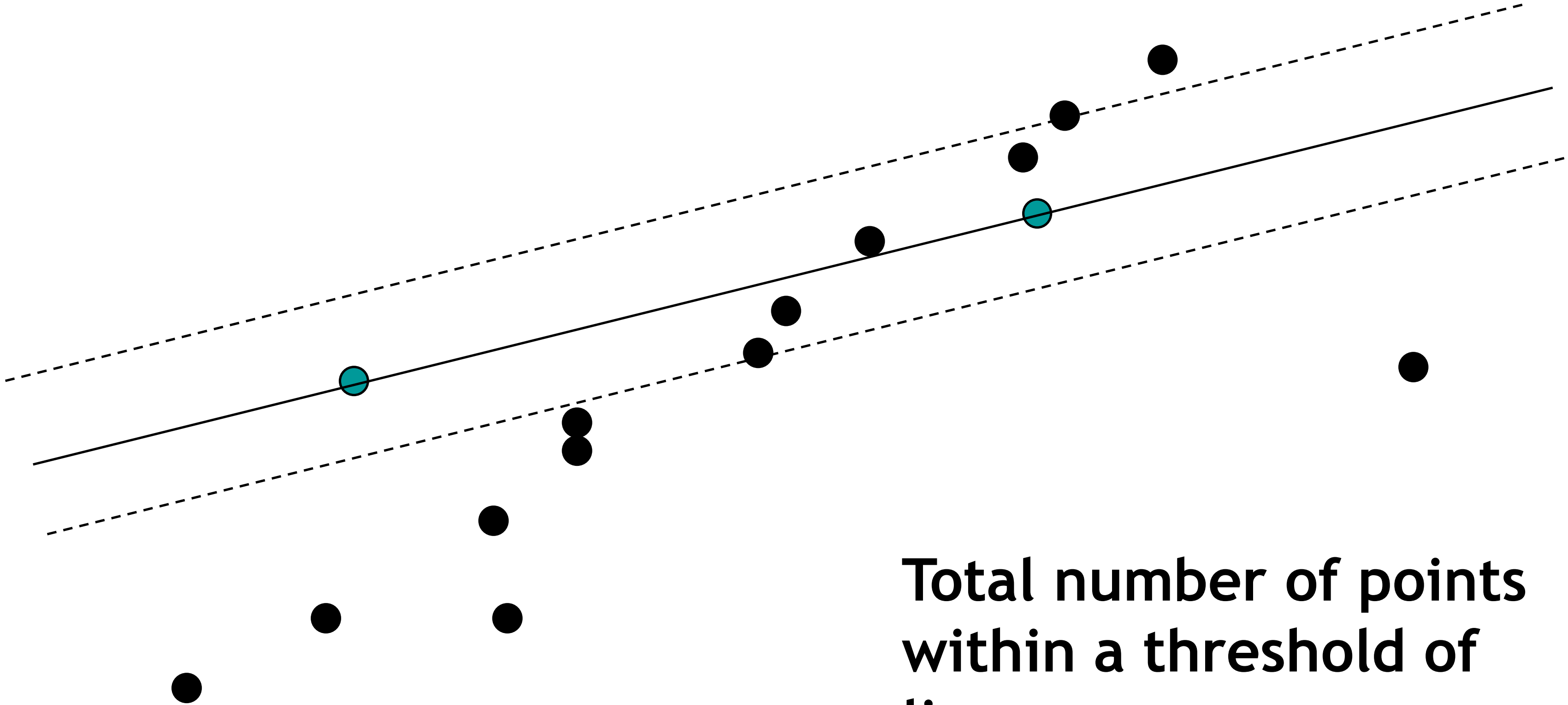
RANSAC Line Fitting Example

- Task: Estimate the best line



RANSAC Line Fitting Example

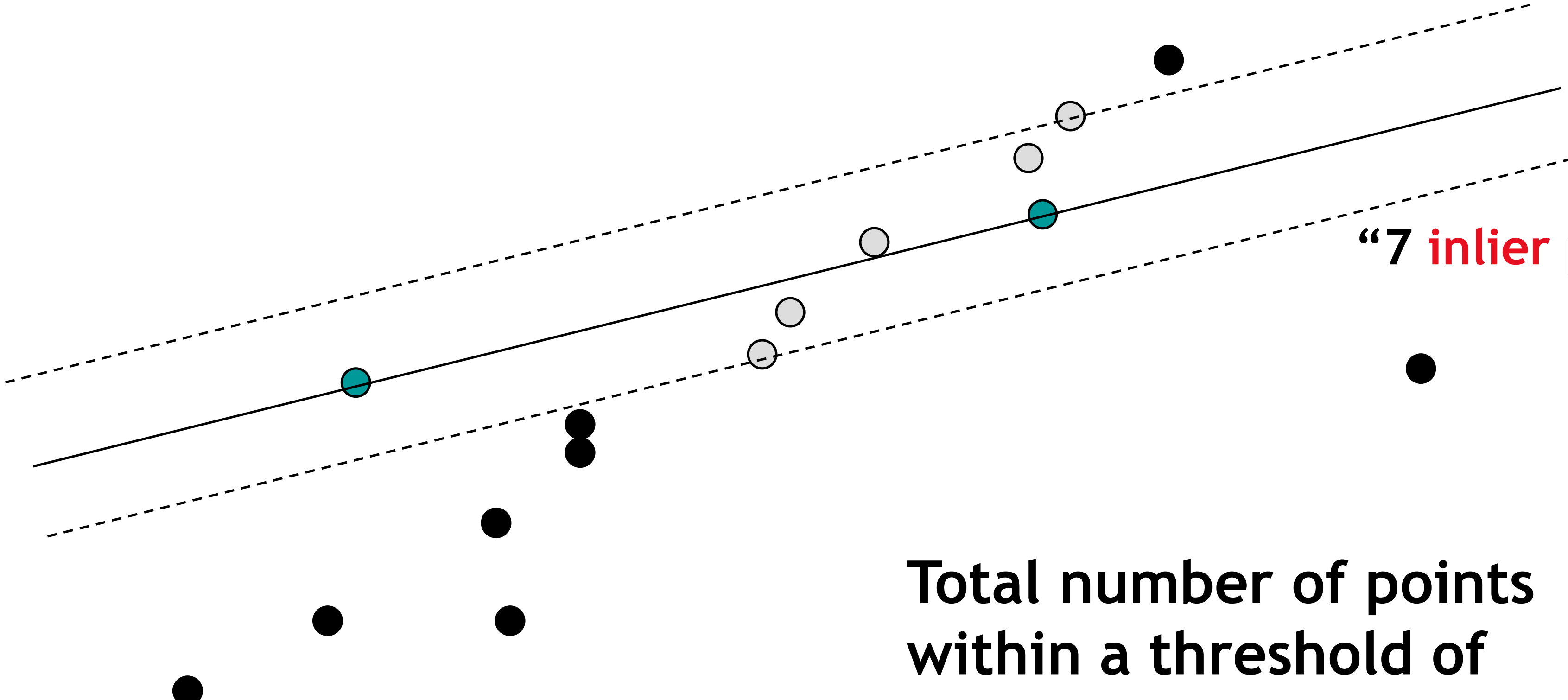
- Task: Estimate the best line



Total number of points within a threshold of line.

RANSAC Line Fitting Example

- Task: Estimate the best line

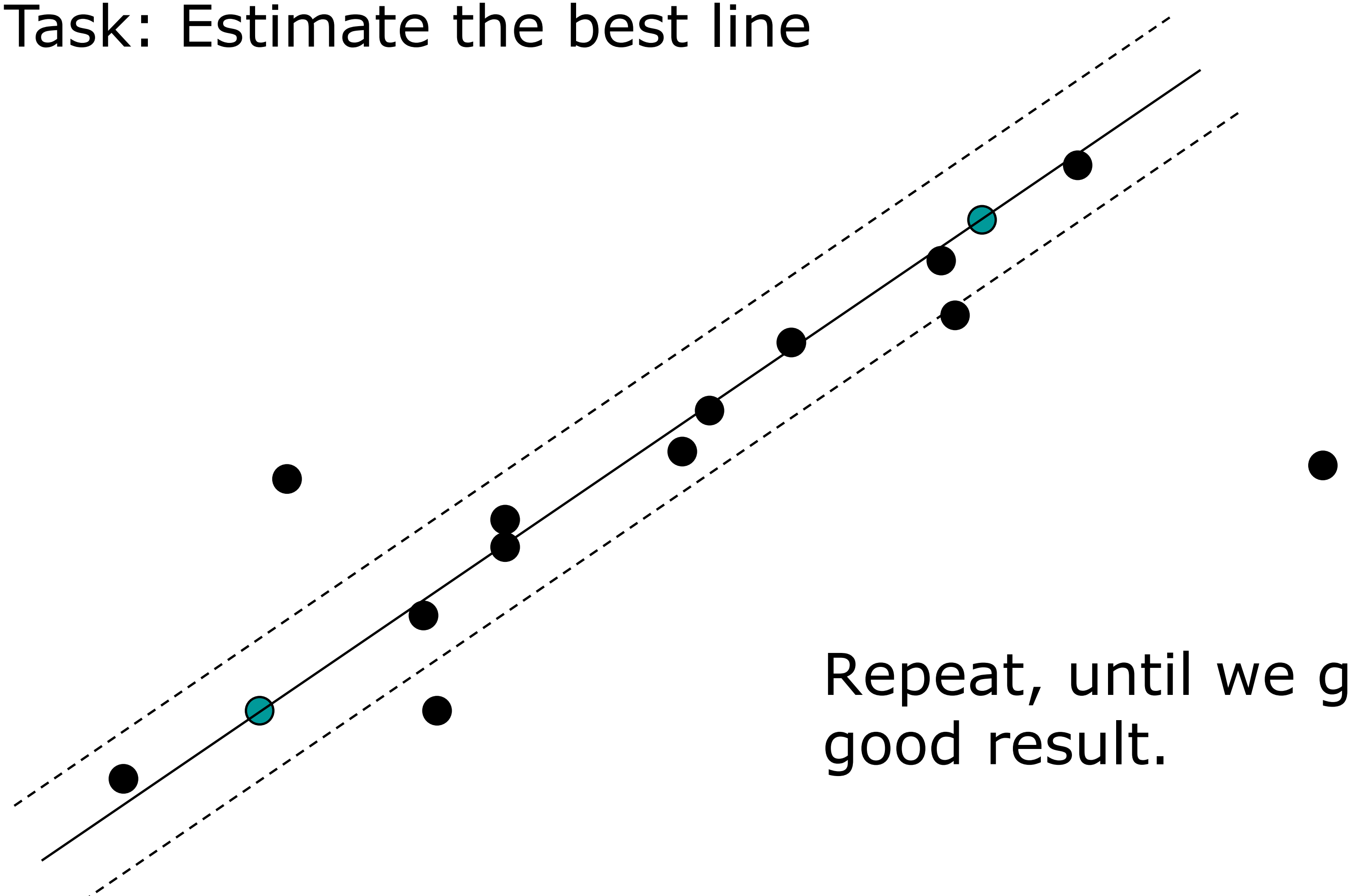


“7 inlier points”

Total number of points within a threshold of line.

RANSAC Line Fitting Example

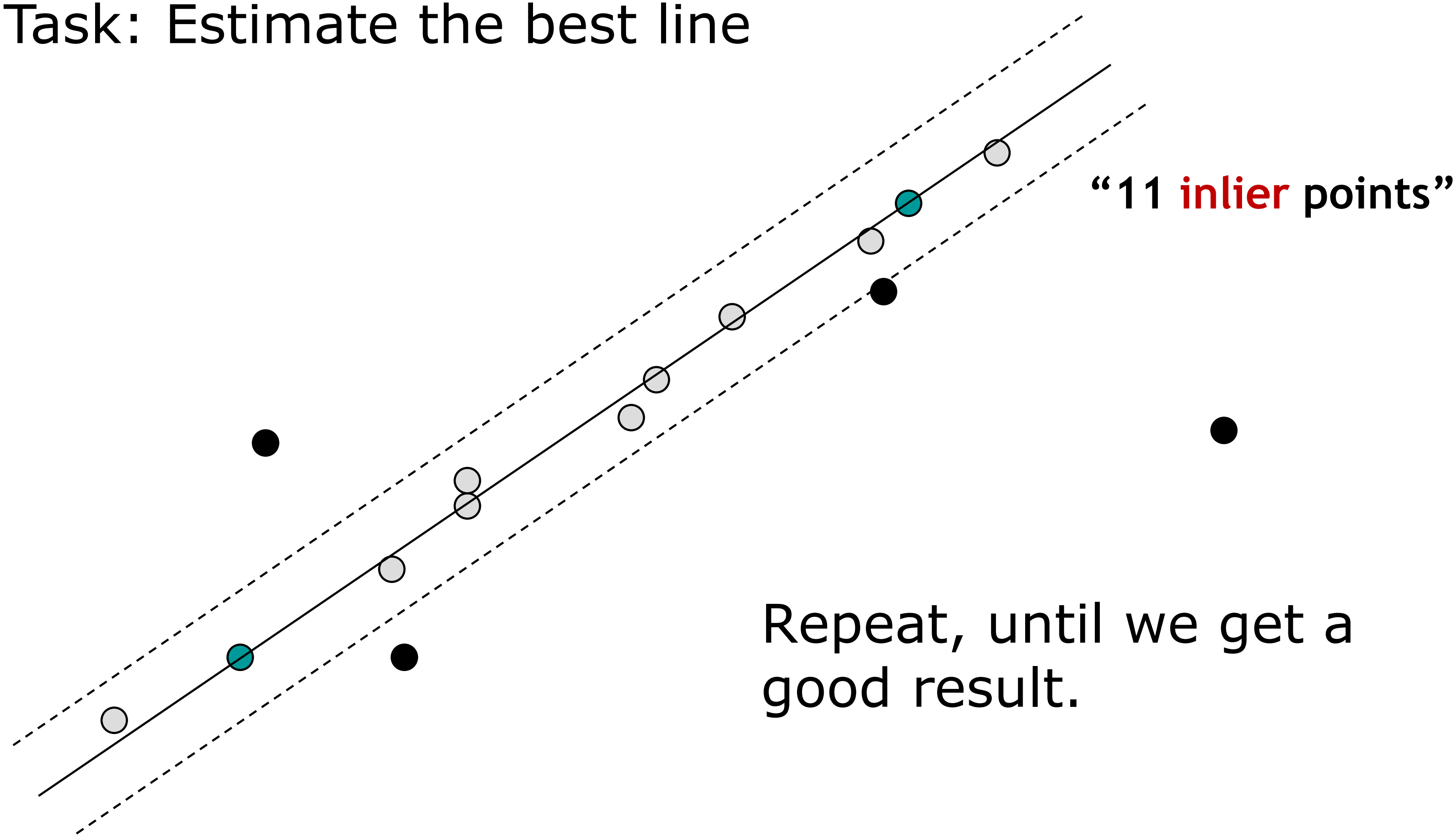
- Task: Estimate the best line



Repeat, until we get a good result.

RANSAC Line Fitting Example

- Task: Estimate the best line



Repeat, until we get a good result.

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: How many samples?

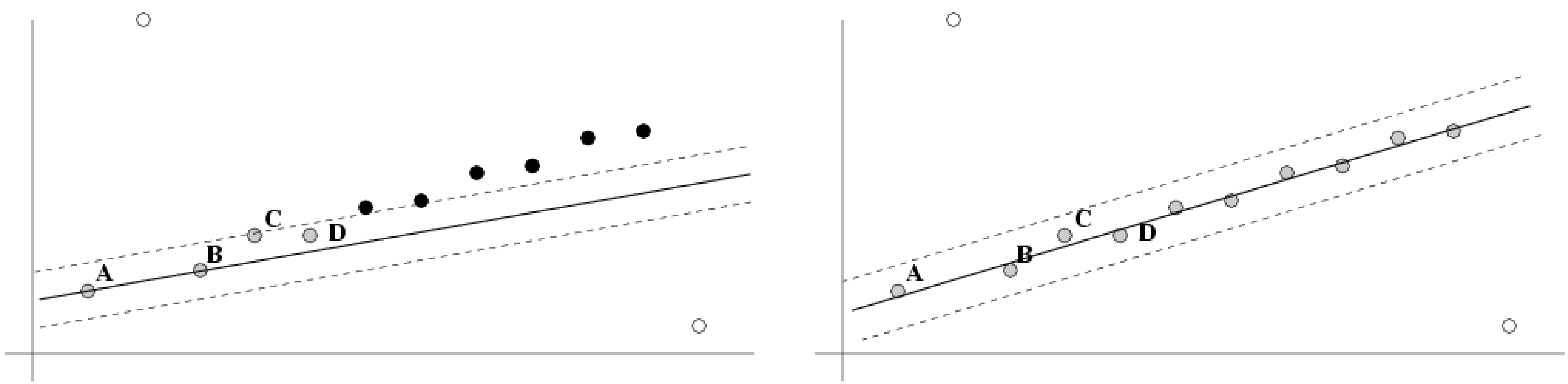
- How many samples are needed?
  - p is the probability of success
  - Suppose w is fraction of inliers (points from line).
  - n points needed to define hypothesis (2 for lines)
  - k samples chosen.
- Prob. that a single sample of n points is correct: w^n
- Prob. that all k samples fail is: $(1-w^n)^k$
- Choose k high enough to keep this below desired failure rate. $k = \frac{\log(1-p)}{\log(1-w^n)}$

RANSAC: Computed k (p=0.99)

Sample size n	Proportion of outliers						
	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**

After RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.



RANSAC: Pros and Cons

- **Pros**:
  - Robust against outliers
  - A general method that can be applied to most cases
  - Fast in huge data sets
  - Easy to implement
- **Cons**:
  - Has a certain probability of success
  - Requires prior knowledge about the data
  - Number of iterations increases logarithmically with outlier percentage

Summary

- **Fitting** problems require finding any supporting evidence for a model, even within clutter and missing features.
  - associate features with an explicit model

- If we know which points belong to the line, how do we find the “optimal” line parameters?
  - Least squares approaches

- What if there are outliers?
  - RANSAC

- What if there are many lines?
  - Voting methods: RANSAC, Hough transform
  - Voting approaches, such as the Hough transform, make it possible to find likely model parameters without searching all combinations of features.
  - Hough transform approaches for lines, circles, and other shapes

Next Time: Classification

Slide Credits

Some slides from Stanley Birchfield, Kristen Grauman, Svetlana Lazebnik, Jia-Bin Huang, Silvio Savarese, Steve Seitz, James Hays, Derek Hoiem, David Forsyth, Fei-Fei Li. Vivek Kwatra, Jinxiang Chai, David Lowe

Questions?