EEE-6512: Image Processing and Computer Vision

November 29, 2017

Lecture #11: Model-Fitting

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

Outline

- Overview
- **Fitting Lines**
- **Fitting Curves**
- Addressing Noise

Last Time

- Global Optimization / Search for Parameters
  - Ordinary Least Squares
  - Total Least Squares

- Hypothesize and Test
  - Hough Transform for Line Detection

Hough Transform Outline

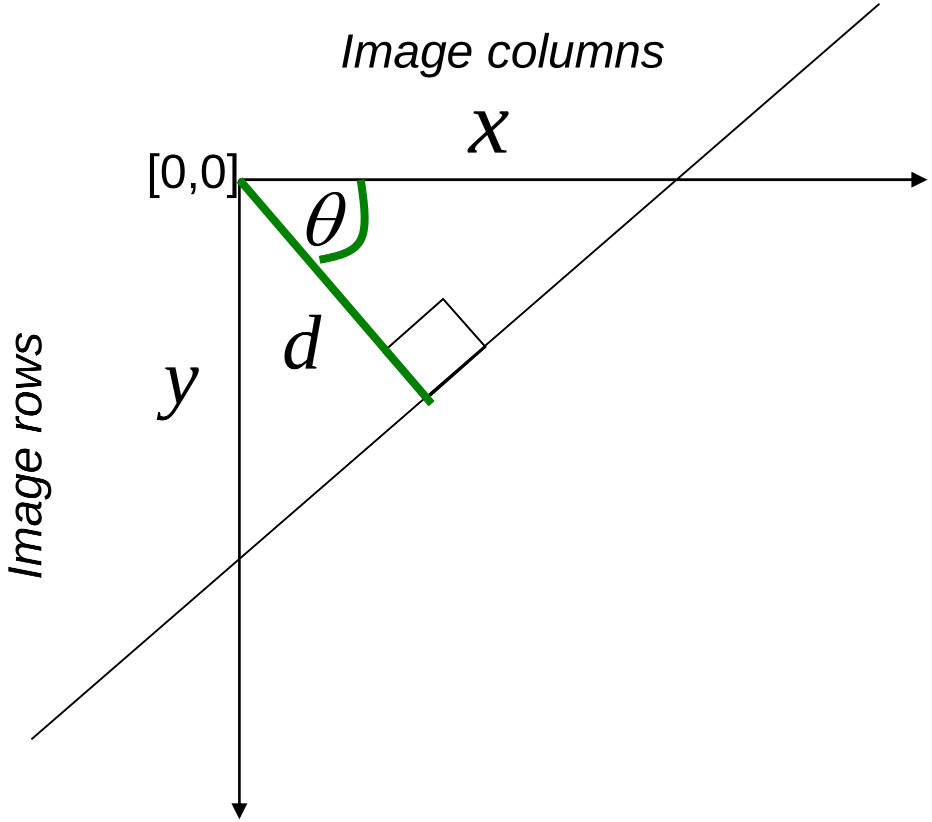
1. Create a grid of parameter values

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

3. Find maximum or local maxima in grid

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

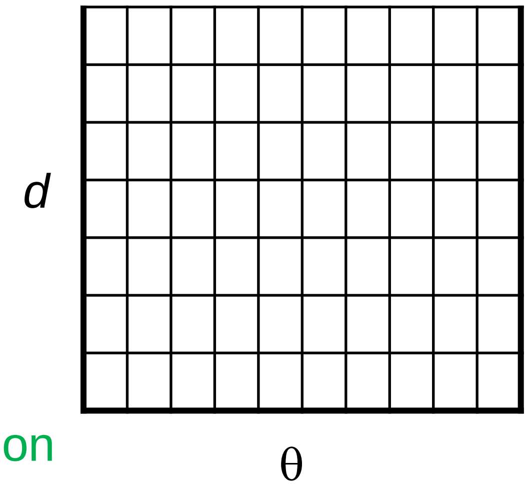
Kristen Grauman

Hough transform algorithm

Using the polar parameterization:

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

H: accumulator array (votes)



Basic Hough transform algorithm

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

   for θ = [θ_{min} to θ_{max}] // some quantization

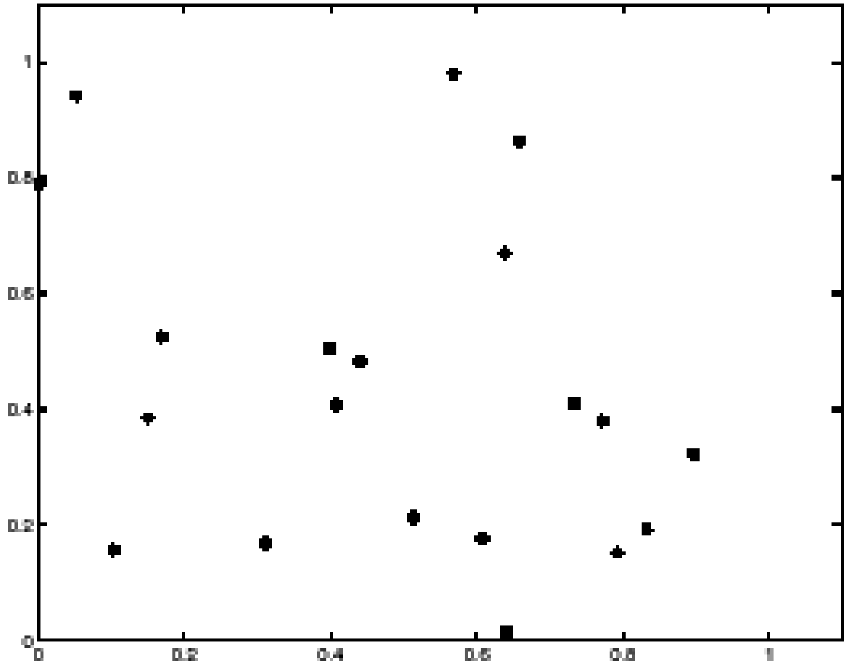
   $$d = x\cos\theta - y\sin\theta$$

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

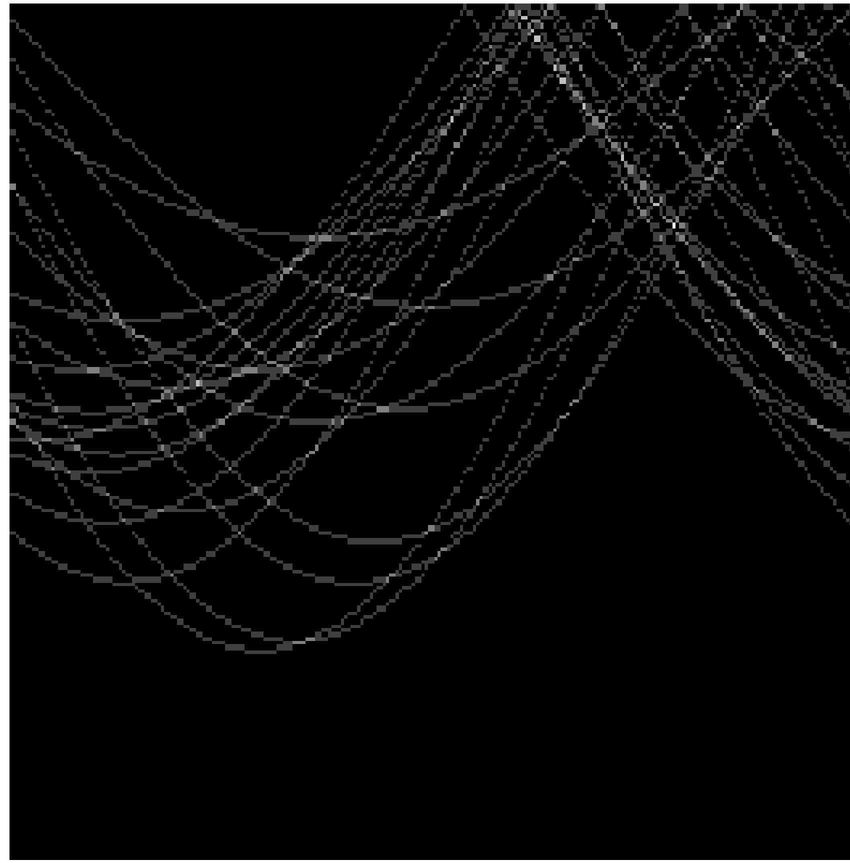
Image Parameter Spaces

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

Random points



features



votes

- Uniform noise can lead to spurious peaks in the array

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

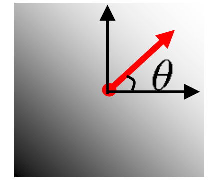
Hough Algorithm Extensions

Extensions

Extension 1: Use the image gradient

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)



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

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

Extensions

Extension 1: Use the image gradient

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

   compute unique (d, θ) based on image gradient at (x,y)

   $H[d, \theta]$ += 1
3. same
4. same

(Reduces degrees of freedom)

Extension 2

- give more votes for stronger edges (use magnitude of gradient)

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

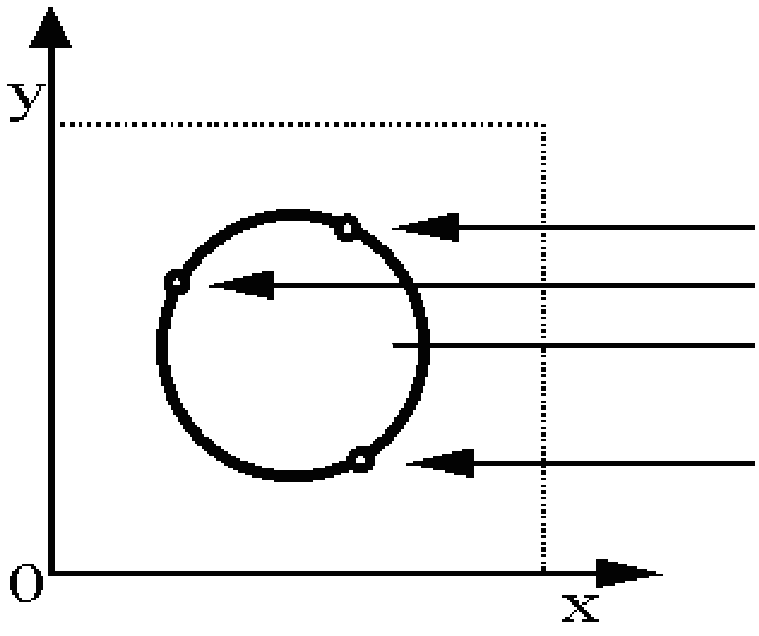
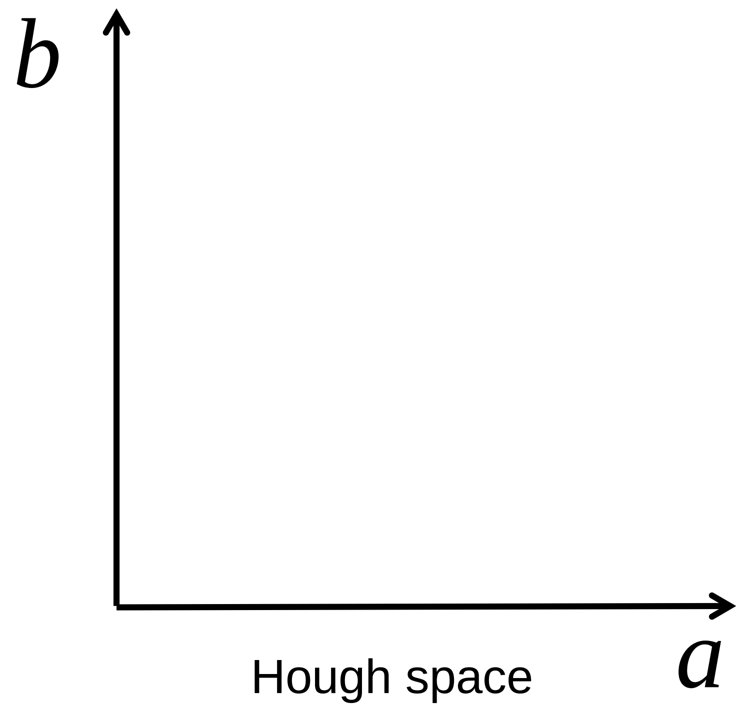


Image space



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

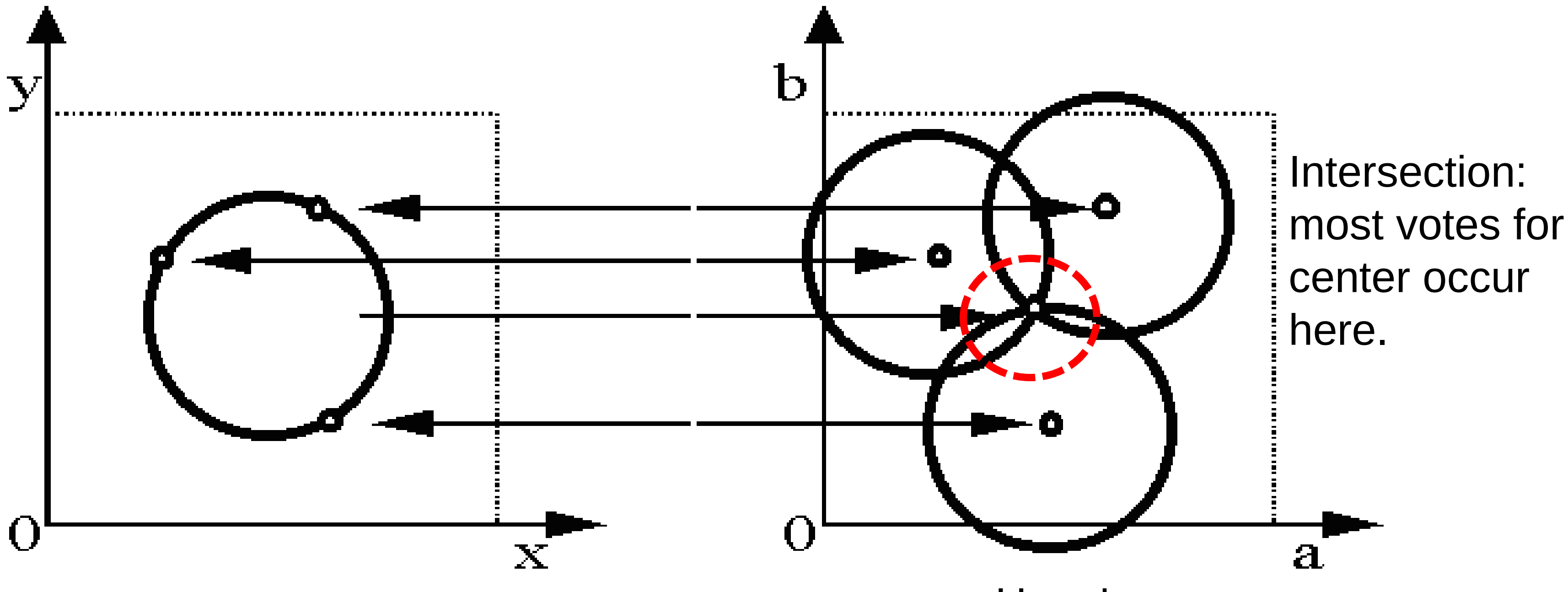


Image space

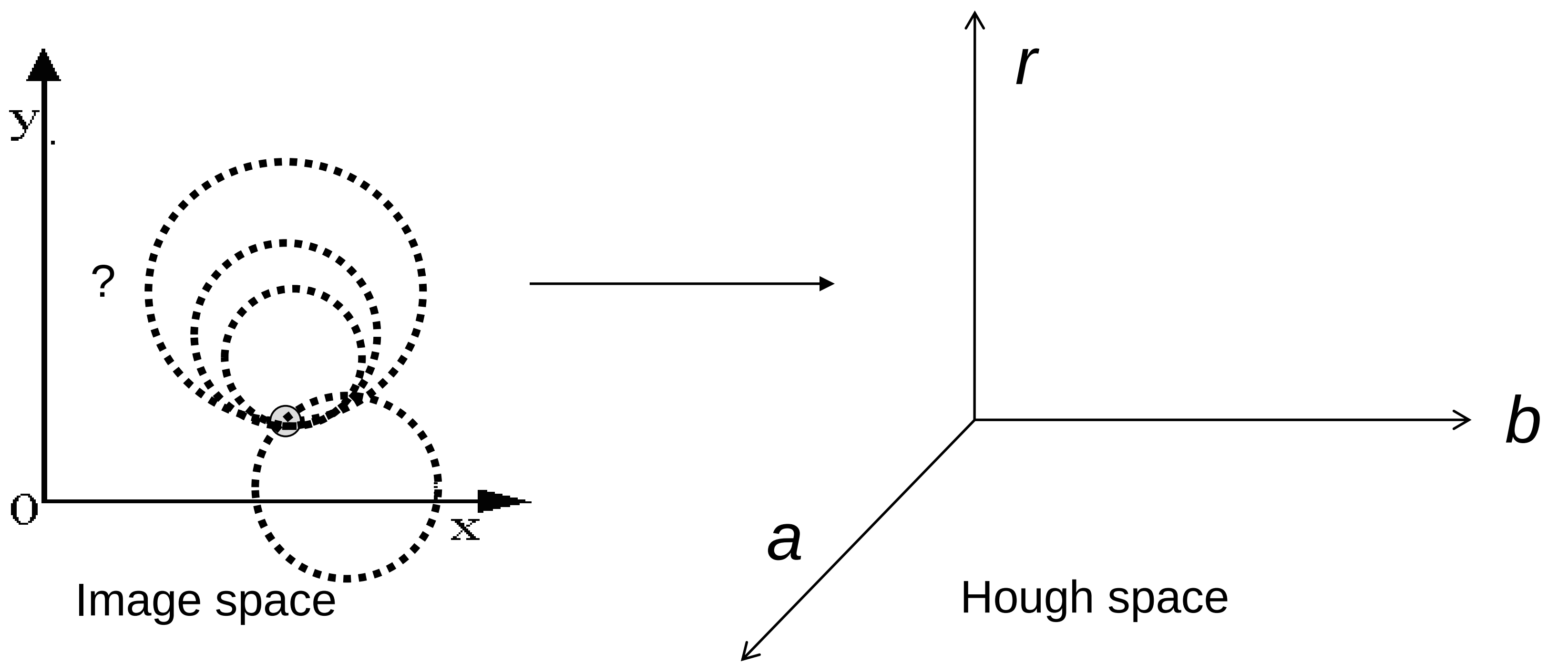
Hough space

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

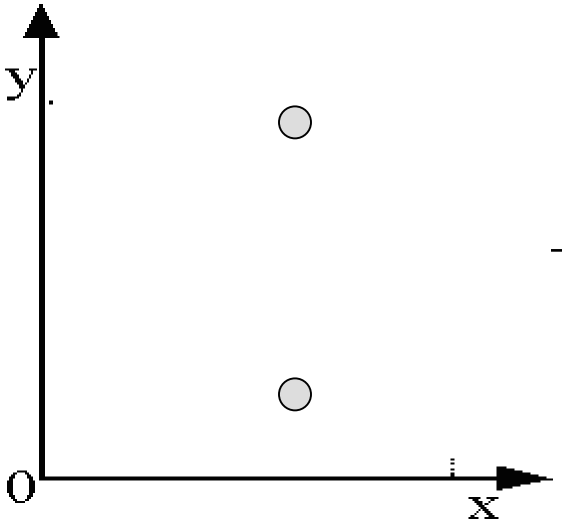
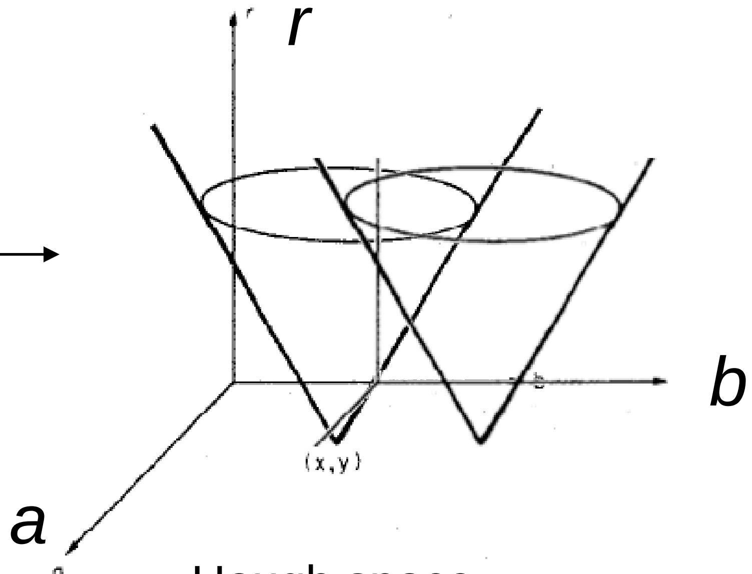


Image space



Hough space

Kristen Grauman

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

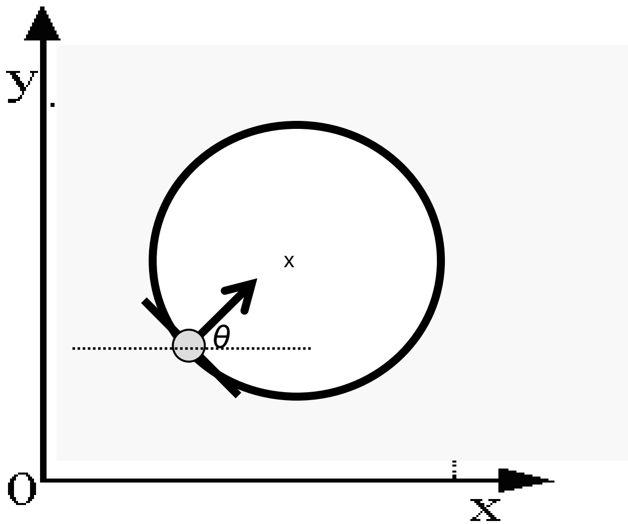
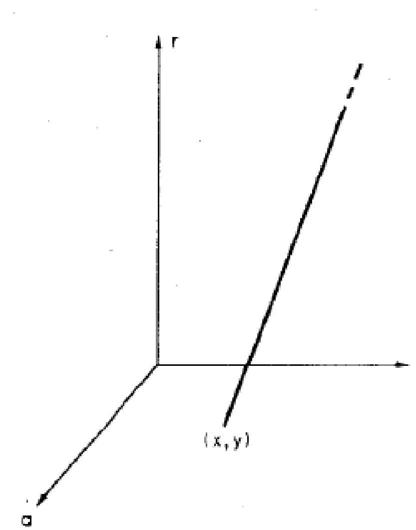


Image space



Hough space

Hough transform for circles

For every edge pixel (x,y) :

  For each possible radius value r:

    For each possible gradient direction θ:

    // or use estimated gradient at (x,y)

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

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

      $H[a,b,r] += 1$

  end

end

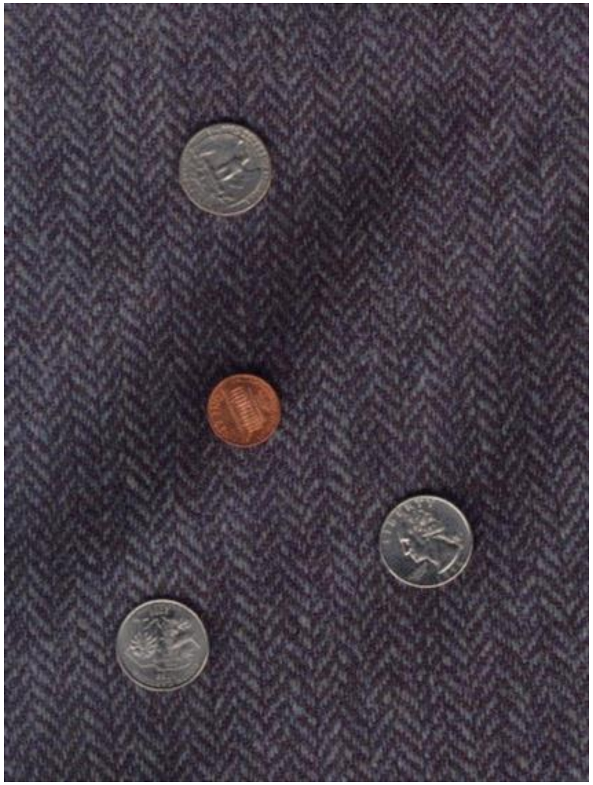
Optimization of Circle Hough Transform

Use Edge Direction (Eliminates Radius)

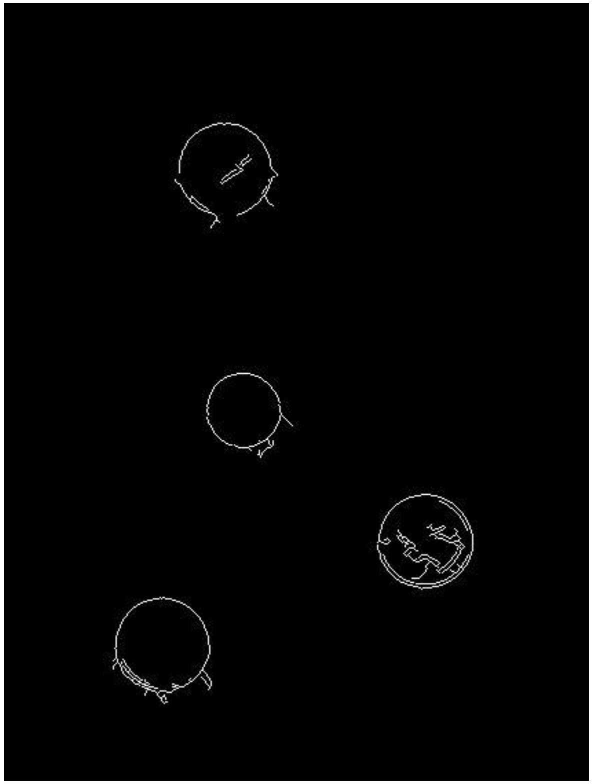
$$b = a * \tan(\theta) - x * \tan(\theta) + y$$

Example: detecting circles with Hough

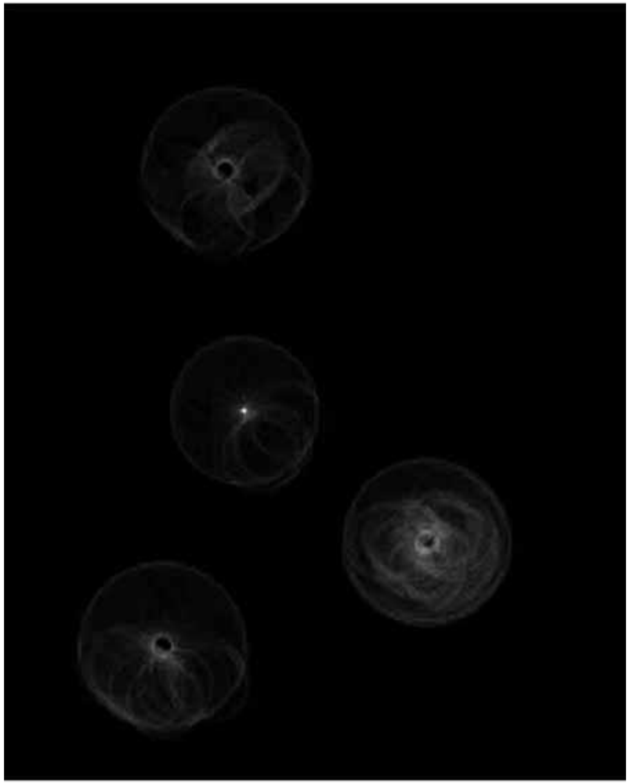
Original



Edges



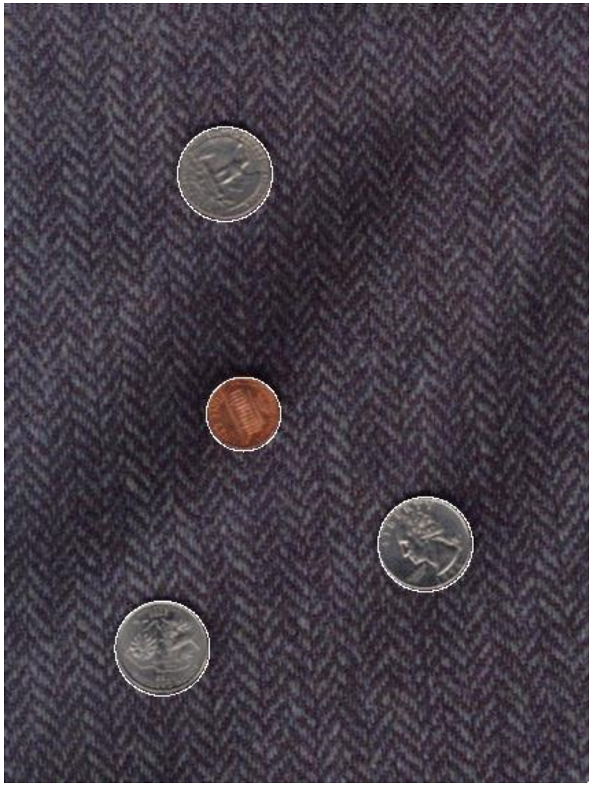
Votes: Penny



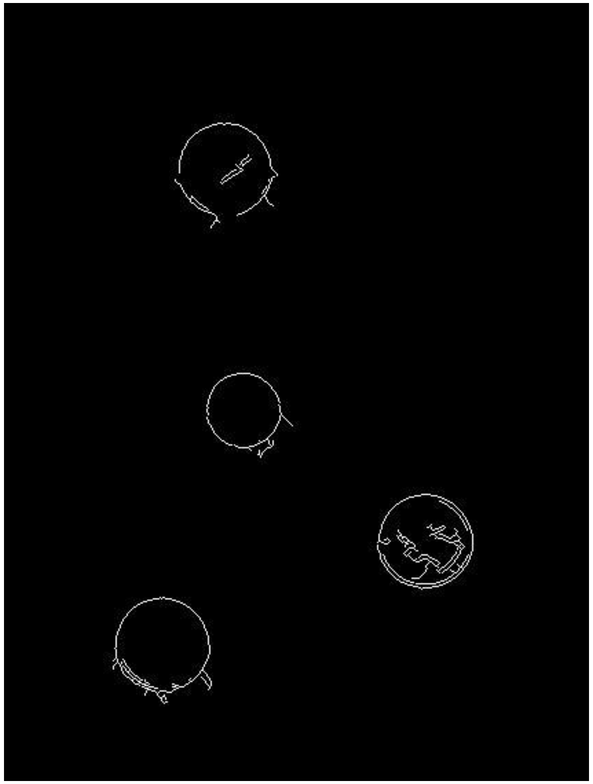
Note: a different Hough transform (with separate accumulators) was used for each circle radius (quarters vs. penny).

Example: detecting circles with Hough

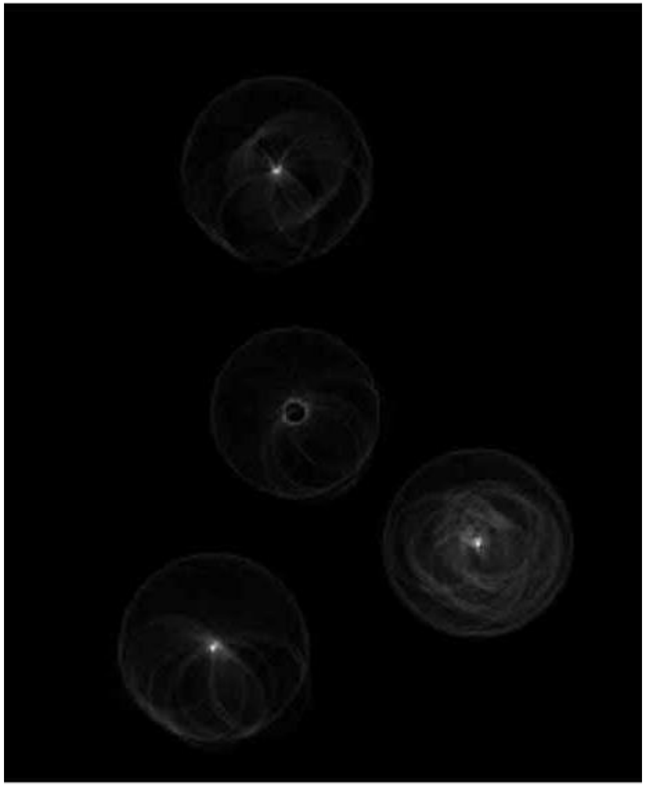
Original



Edges



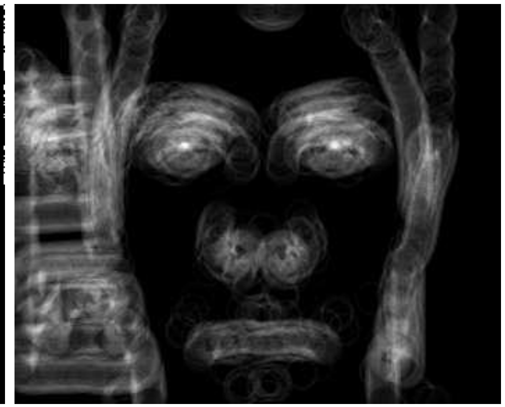
Votes: Quarter



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

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

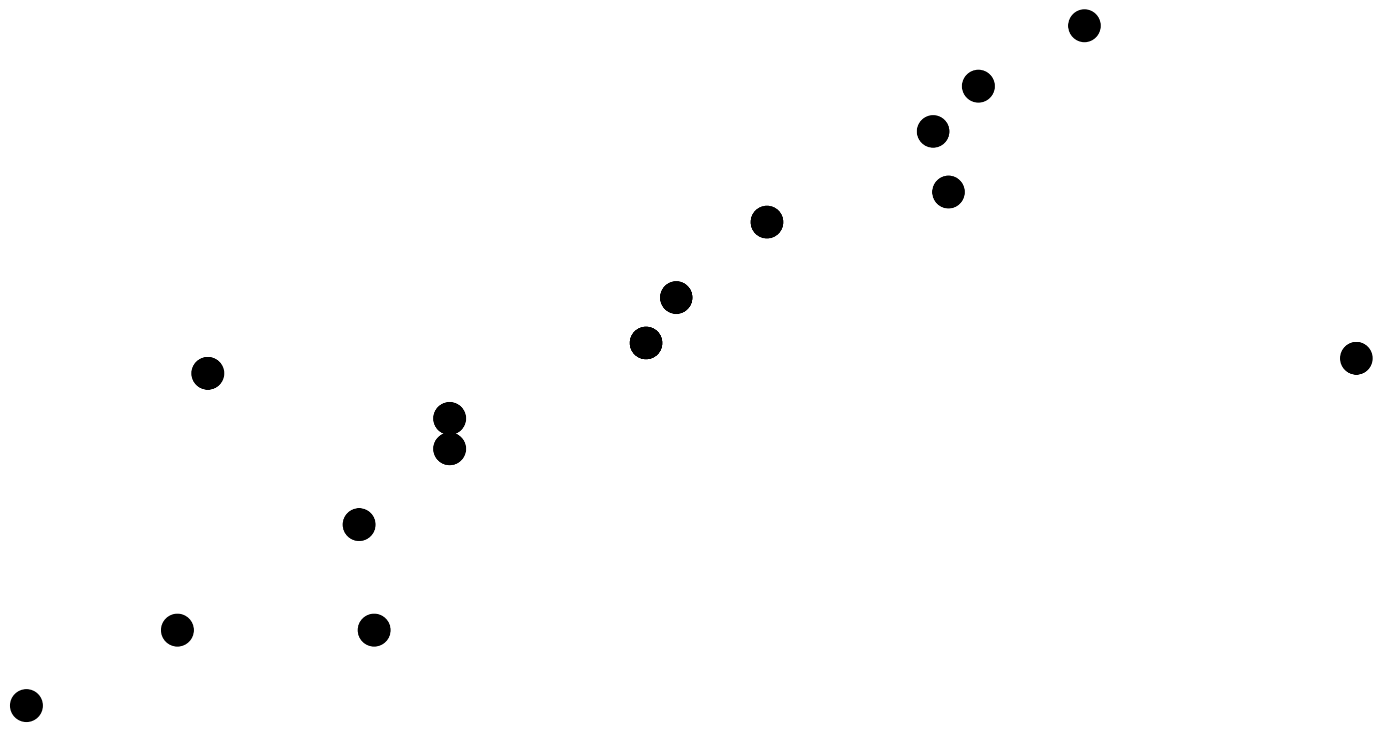
RANSAC loop:

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

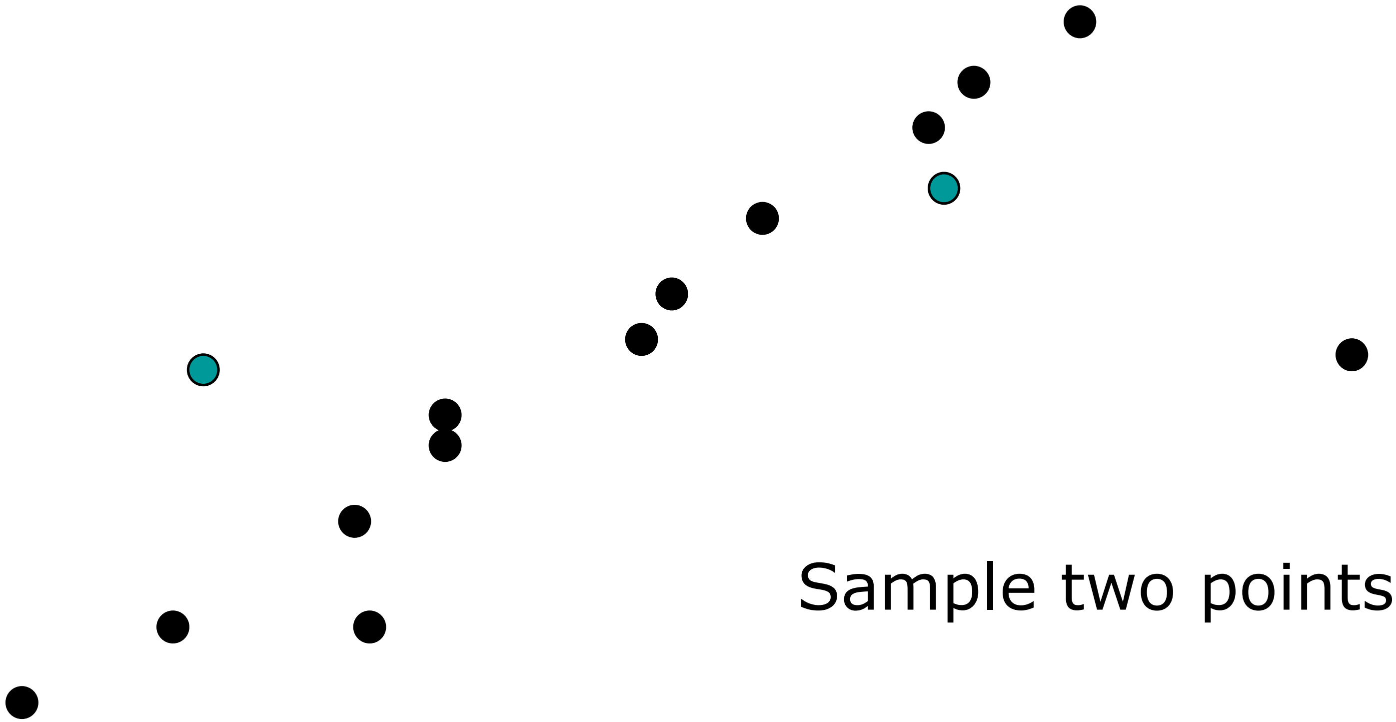
RANSAC Line Fitting Example

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



RANSAC Line Fitting Example

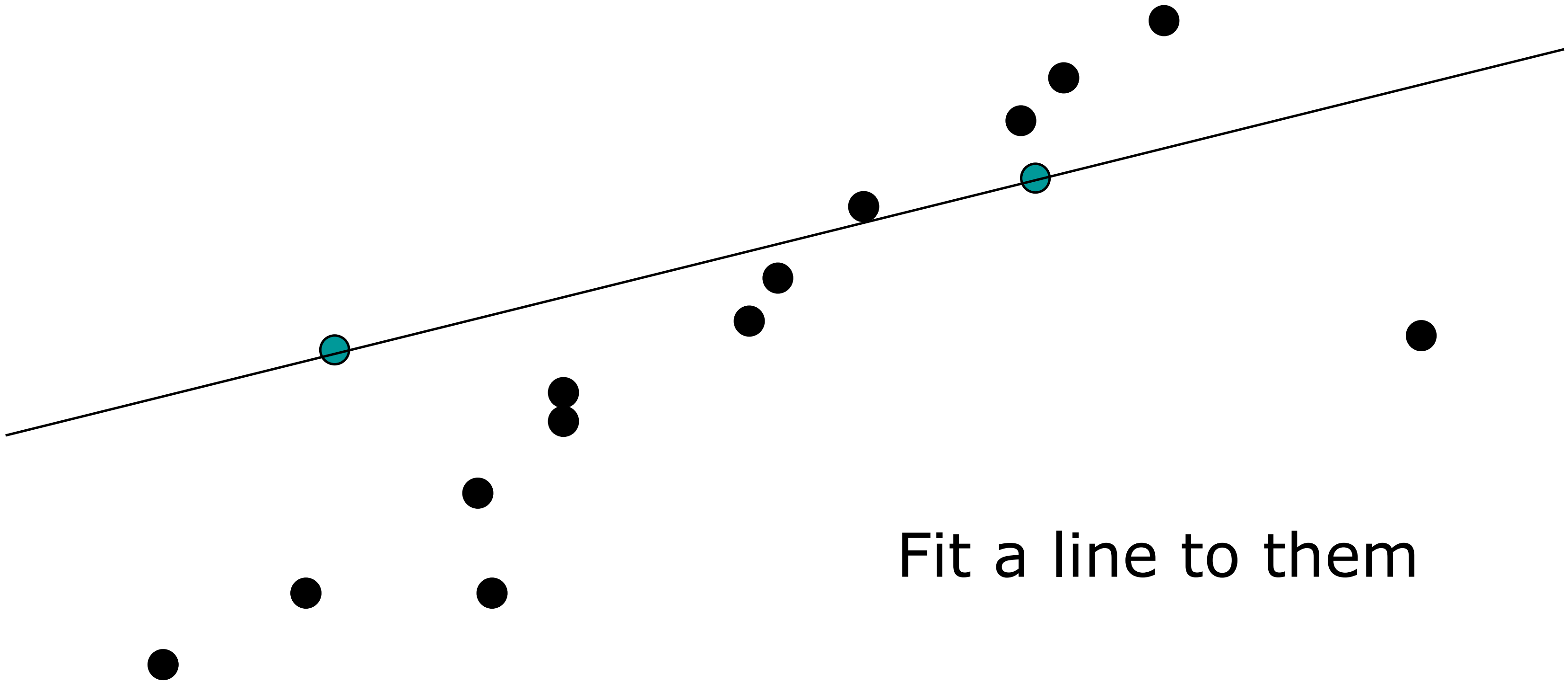
- Task: Estimate the best line



Sample two points

RANSAC Line Fitting Example

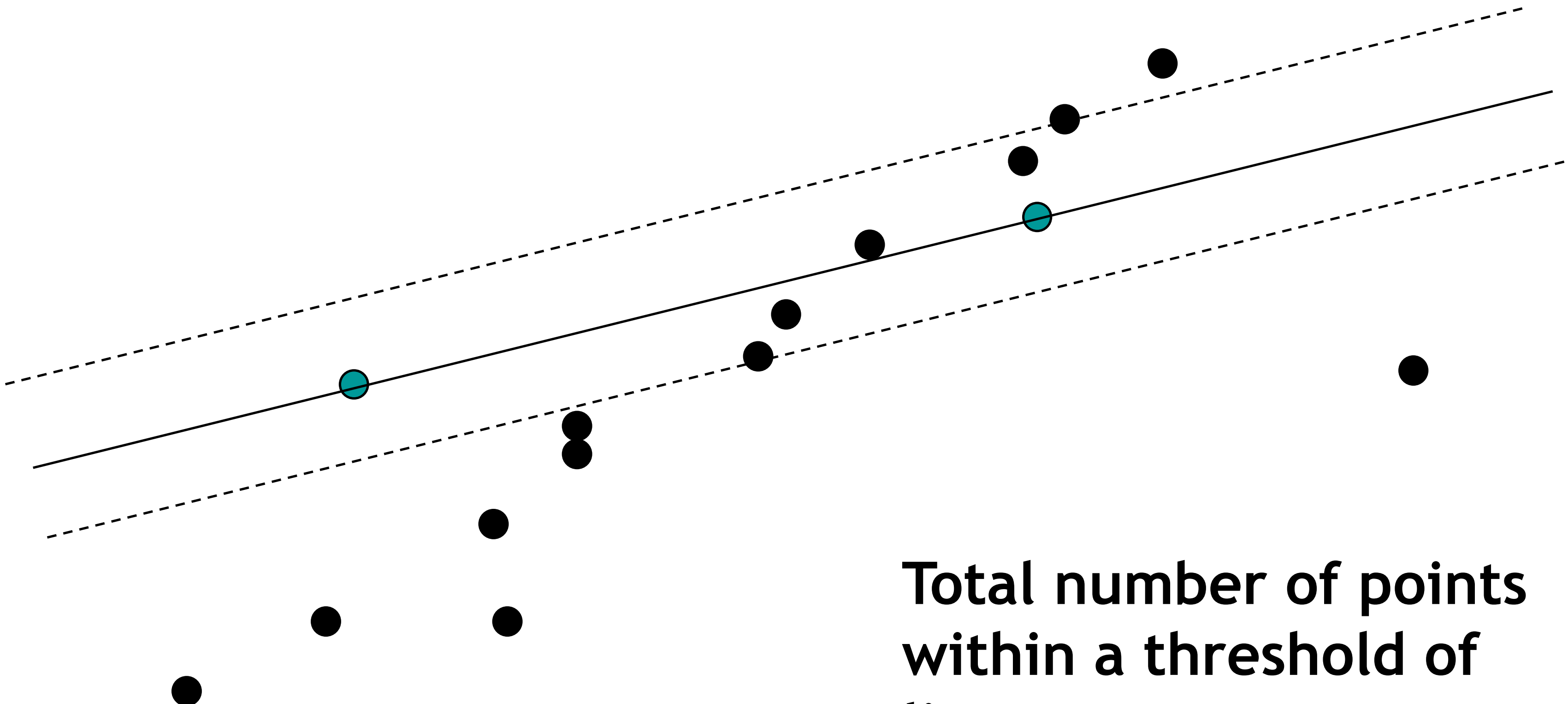
- Task: Estimate the best line



Fit a line to them

RANSAC Line Fitting Example

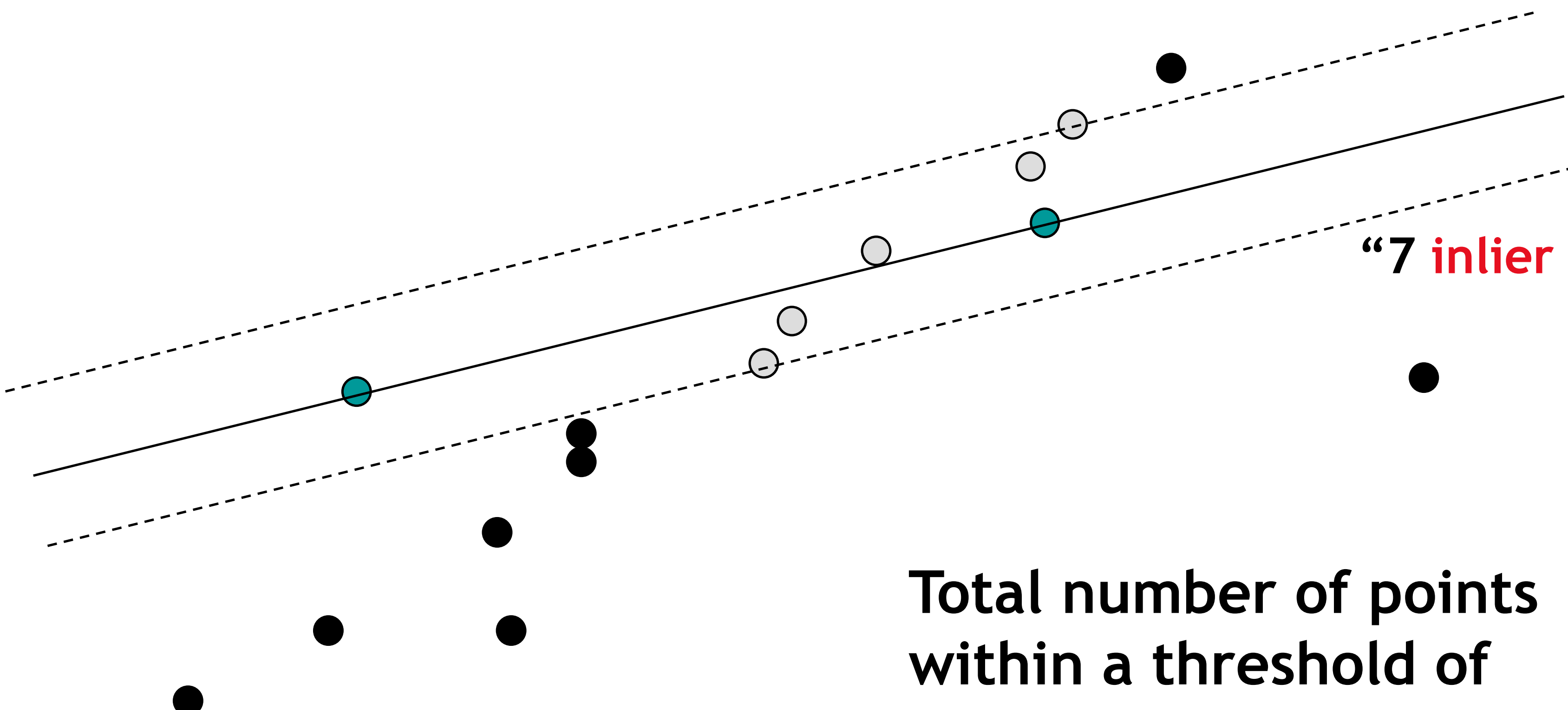
- Task: Estimate the best line



Total number of points within a threshold of line.

RANSAC Line Fitting Example

- Task: Estimate the best line

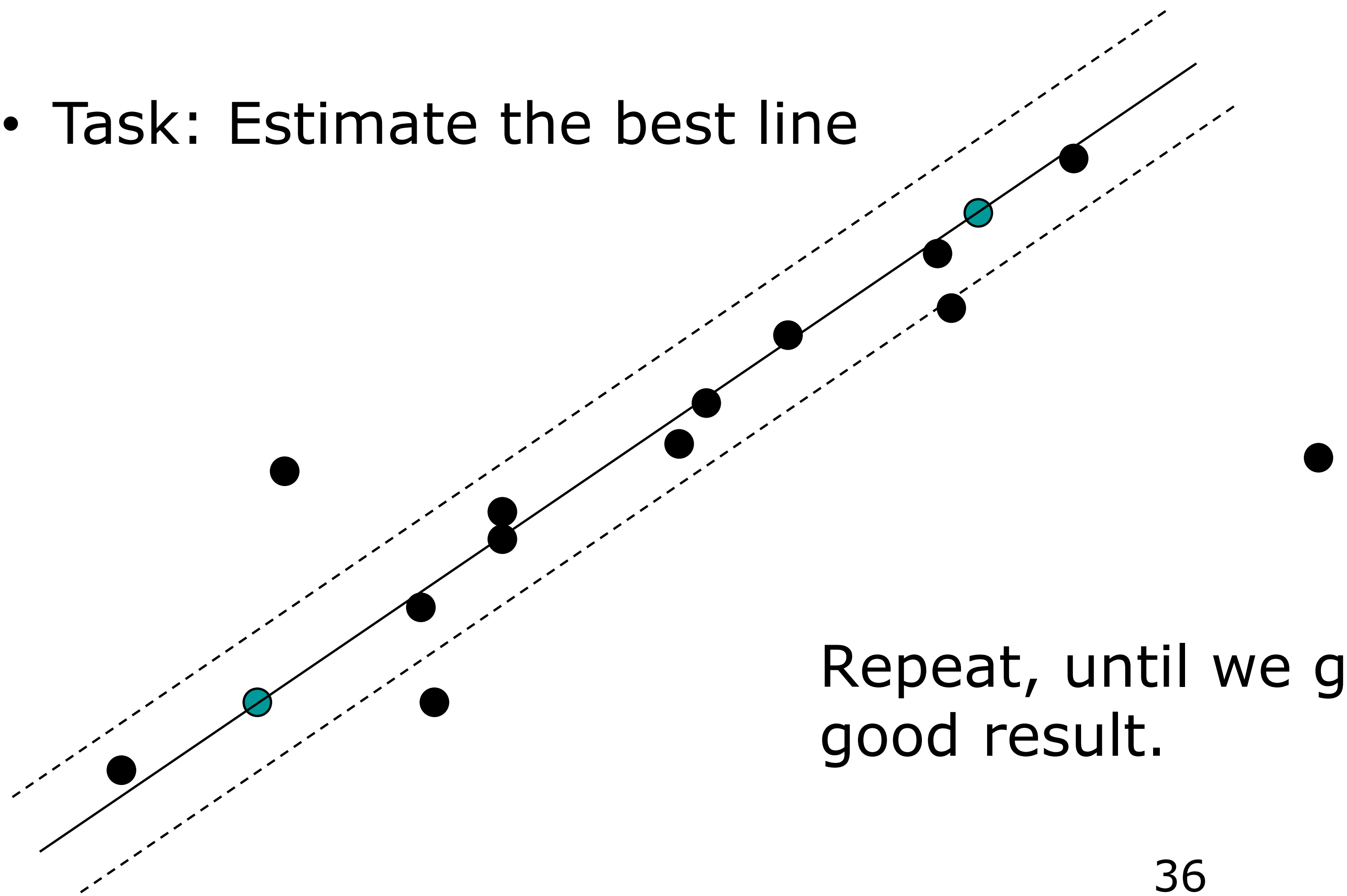


“7 inlier points”

Total number of points within a threshold of line.

RANSAC Line Fitting Example

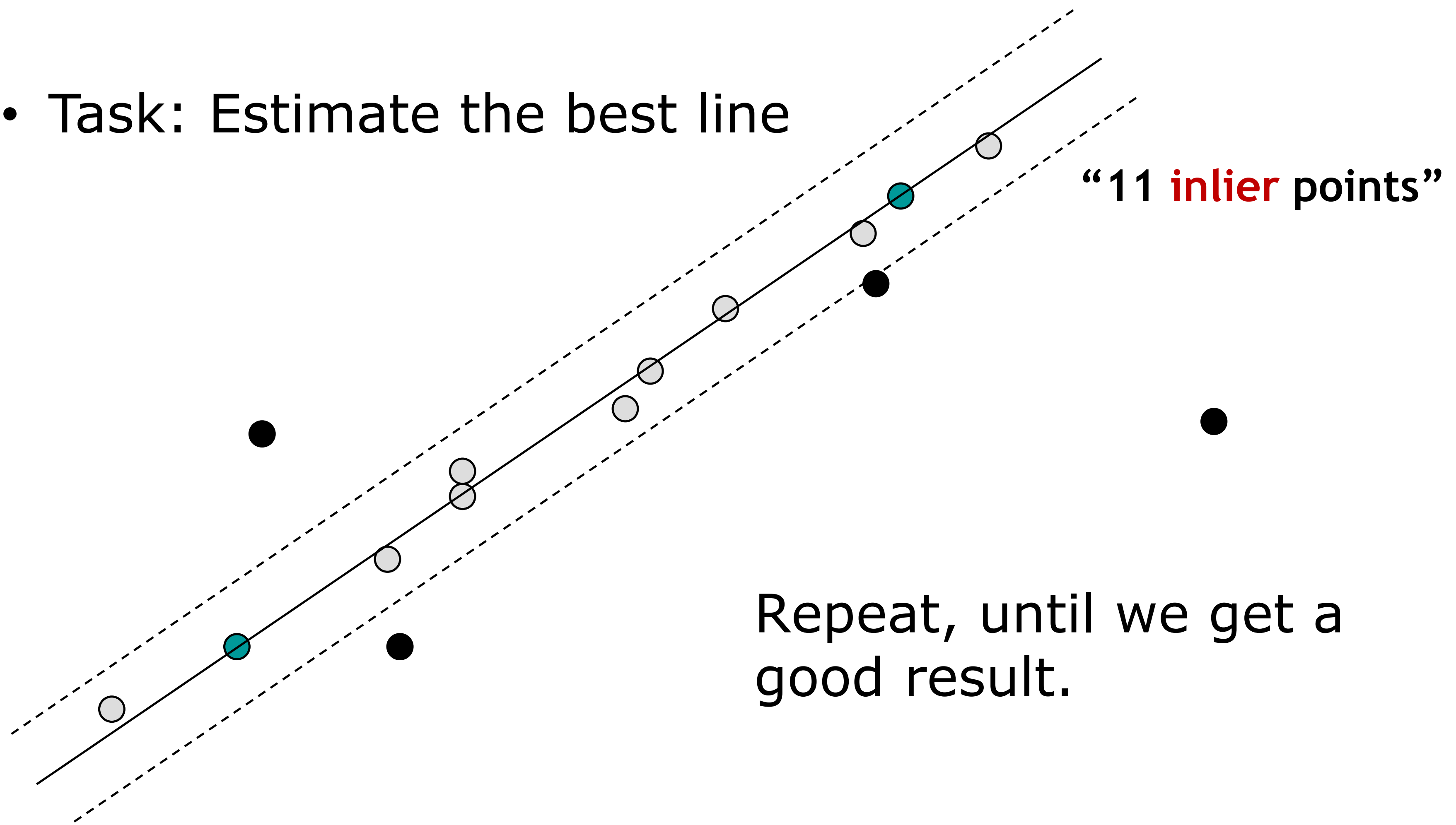
- Task: Estimate the best line



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?
  - 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: $(1-w^n)^k$

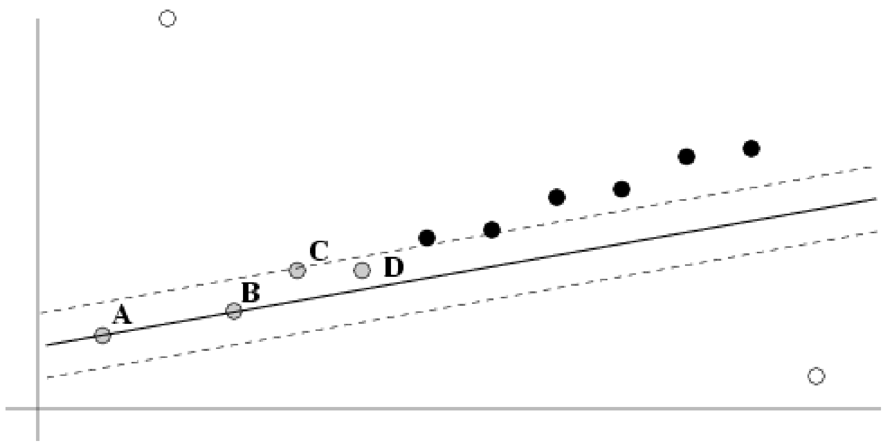
$\Rightarrow$ Choose k high enough to keep this below desired failure rate.

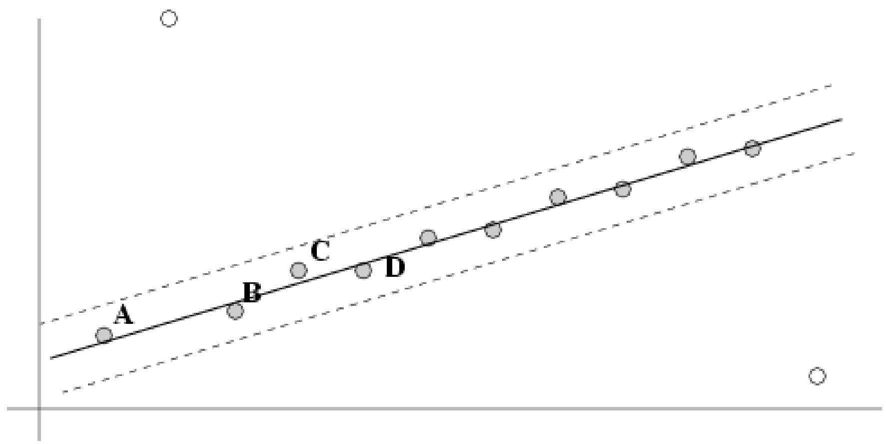
RANSAC: Computed k

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

Questions?

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