

# Lecture 5: Edge Detection

Dr. Juan Carlos Niebles  
Stanford AI Lab

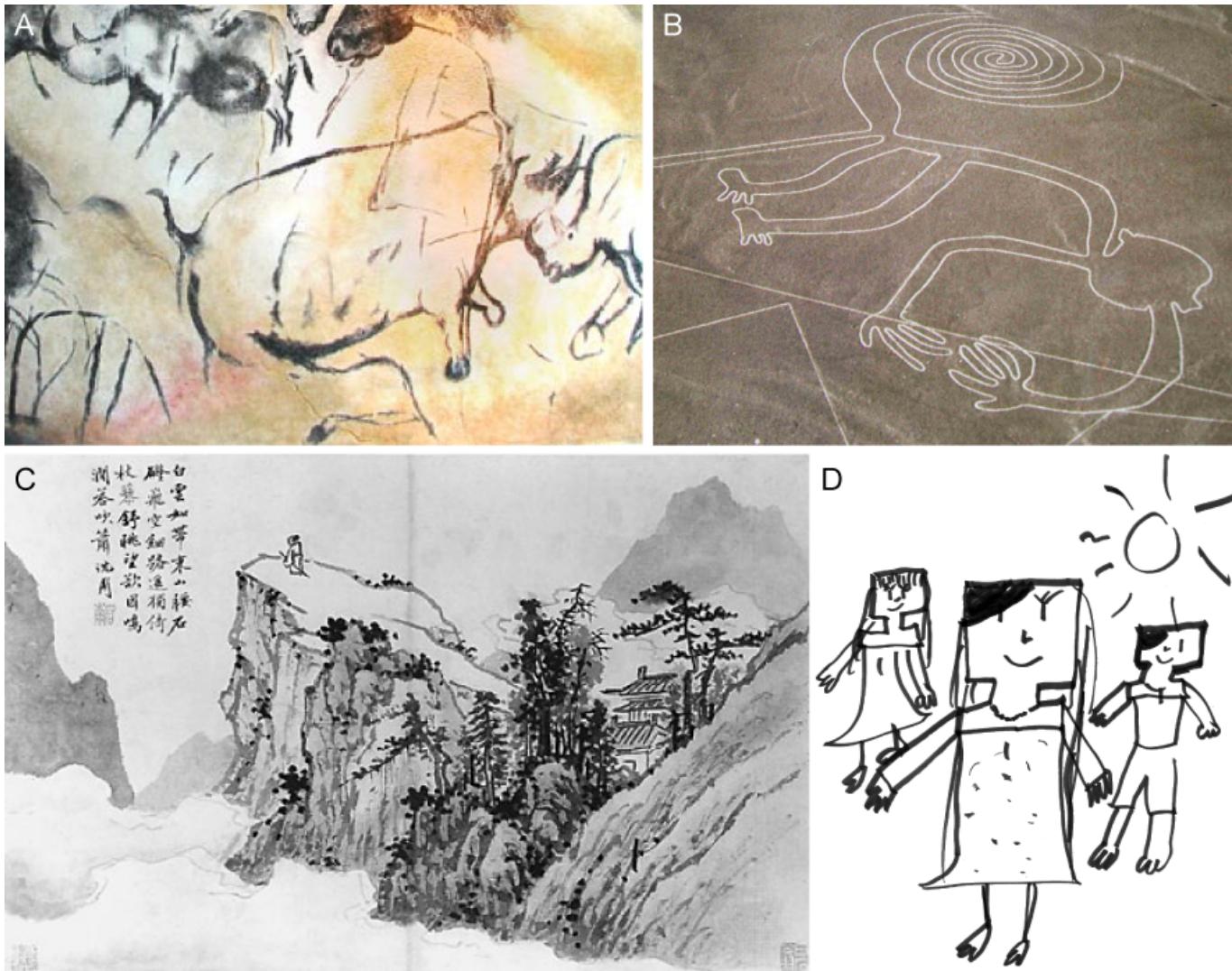
Professor Fei-Fei Li  
Stanford Vision Lab

# What we will learn today

- Edge detection
- A simple edge detector
- Canny edge detector
- A model fitting method for edge detection
  - RANSAC

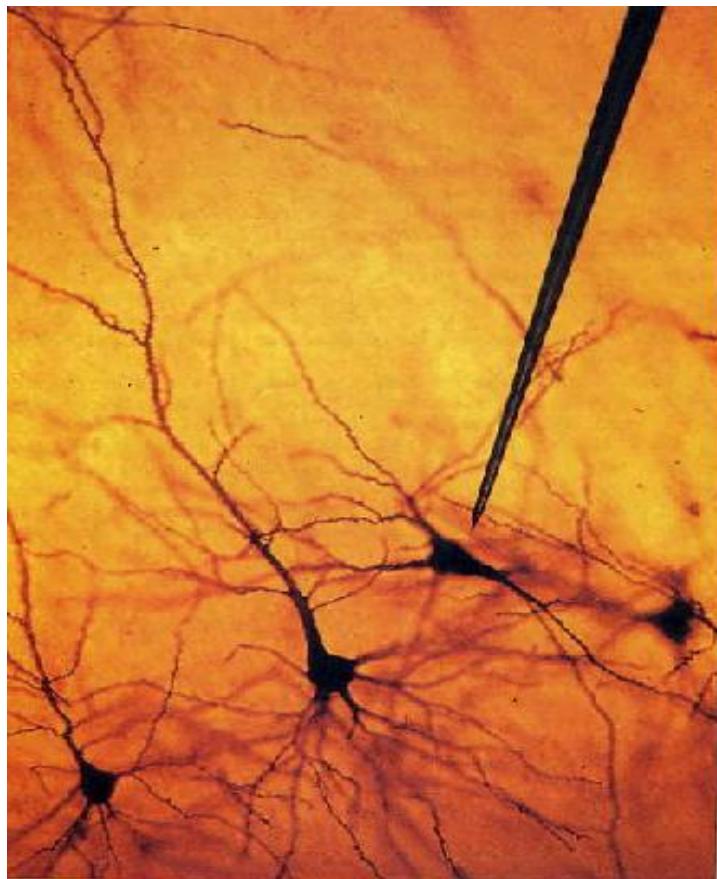
Some background reading:

Forsyth and Ponce, Computer Vision, Chapter 8, Chapter 15.5.2

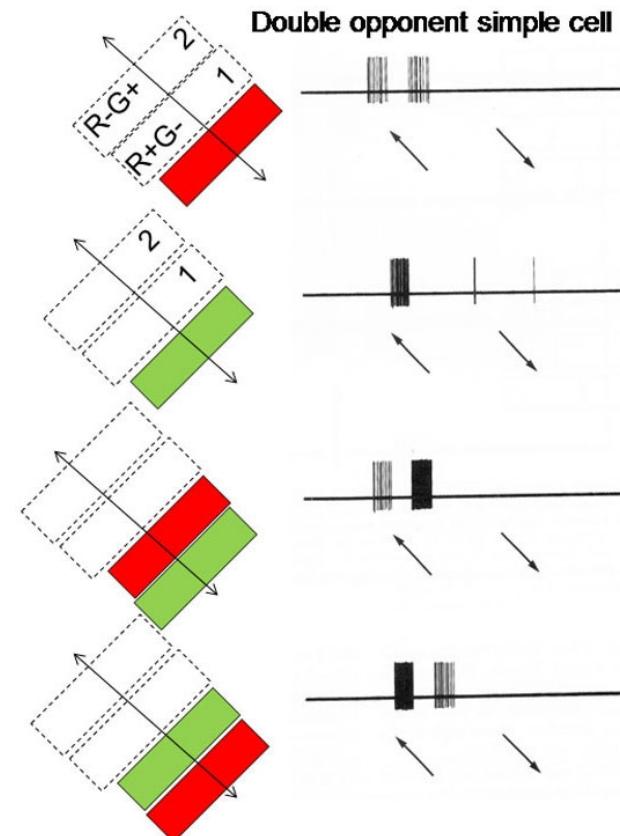


- (A) Cave painting at Chauvet, France, about 30,000 B.C.;
- (B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;
- (C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
- (D) Line drawing by 7-year old I. Lleras (2010 A.D.).

# We know edges are special from human (mammalian) vision studies

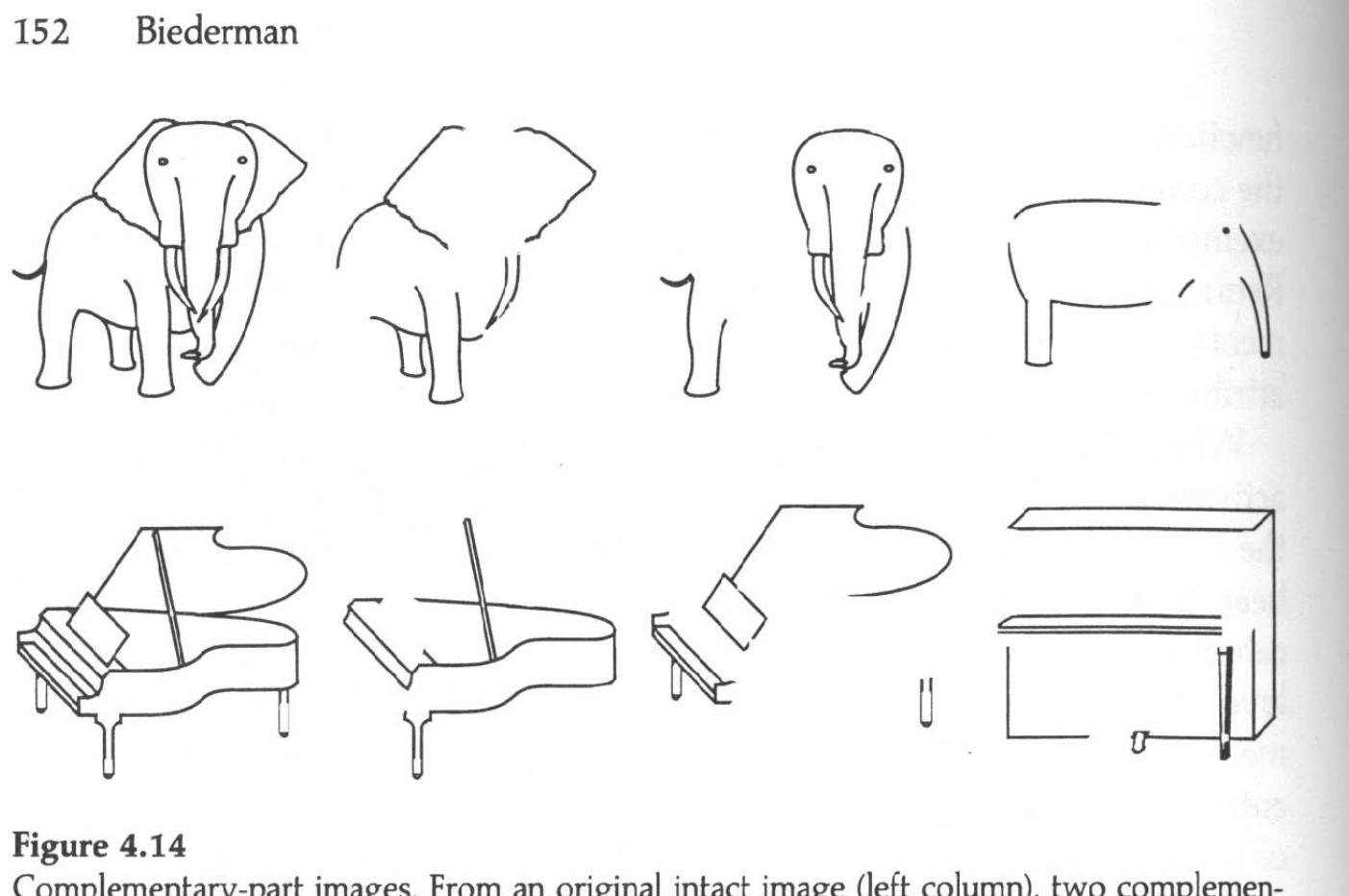


Hubel & Wiesel, 1960s



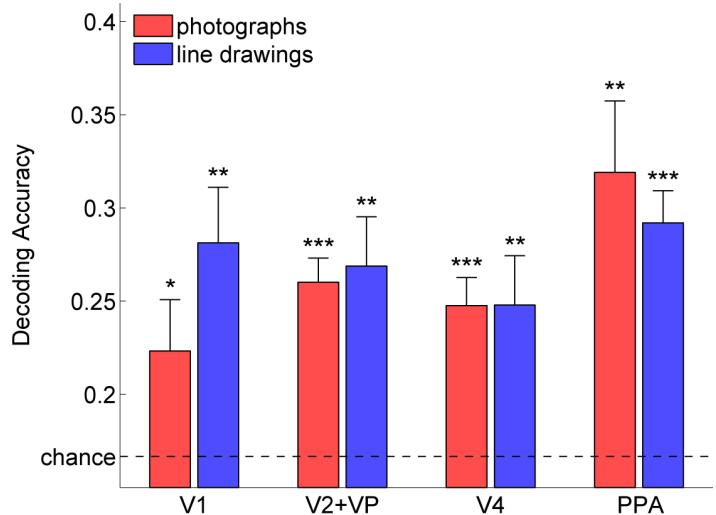
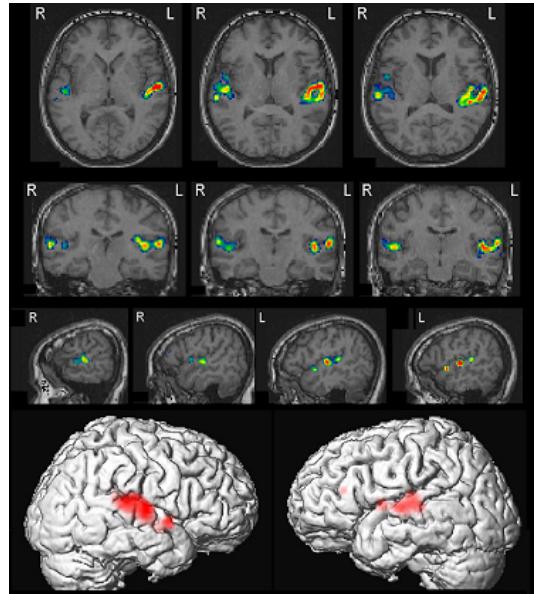
# We know edges are special from human (mammalian) vision studies

152 Biederman



**Figure 4.14**

Complementary-part images. From an original intact image (left column), two complemen-



Walther, Chai, Caddigan, Beck & Fei-Fei, *PNAS*, 2011

# Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)



Source: D. Lowe

# Why do we care about edges?

- Extract information,  
recognize objects



- Recover geometry and  
viewpoint



# Origins of edges



surface normal discontinuity

depth discontinuity

surface color discontinuity

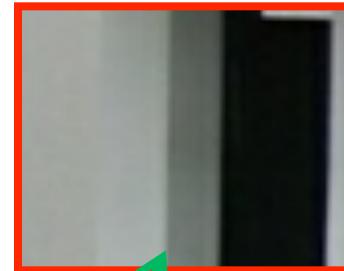
illumination discontinuity

Source: D. Hoiem

# Closeup of edges



Surface normal discontinuity

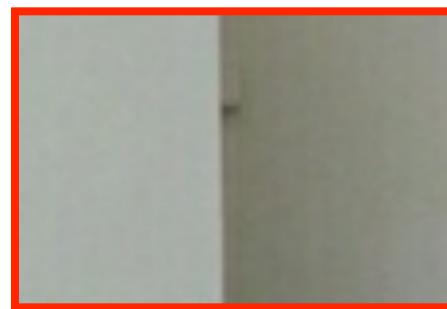


Source: D. Hoiem

# Closeup of edges



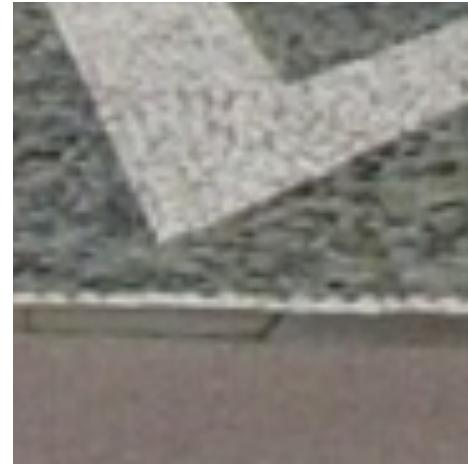
Depth discontinuity



# Closeup of edges



Surface color discontinuity

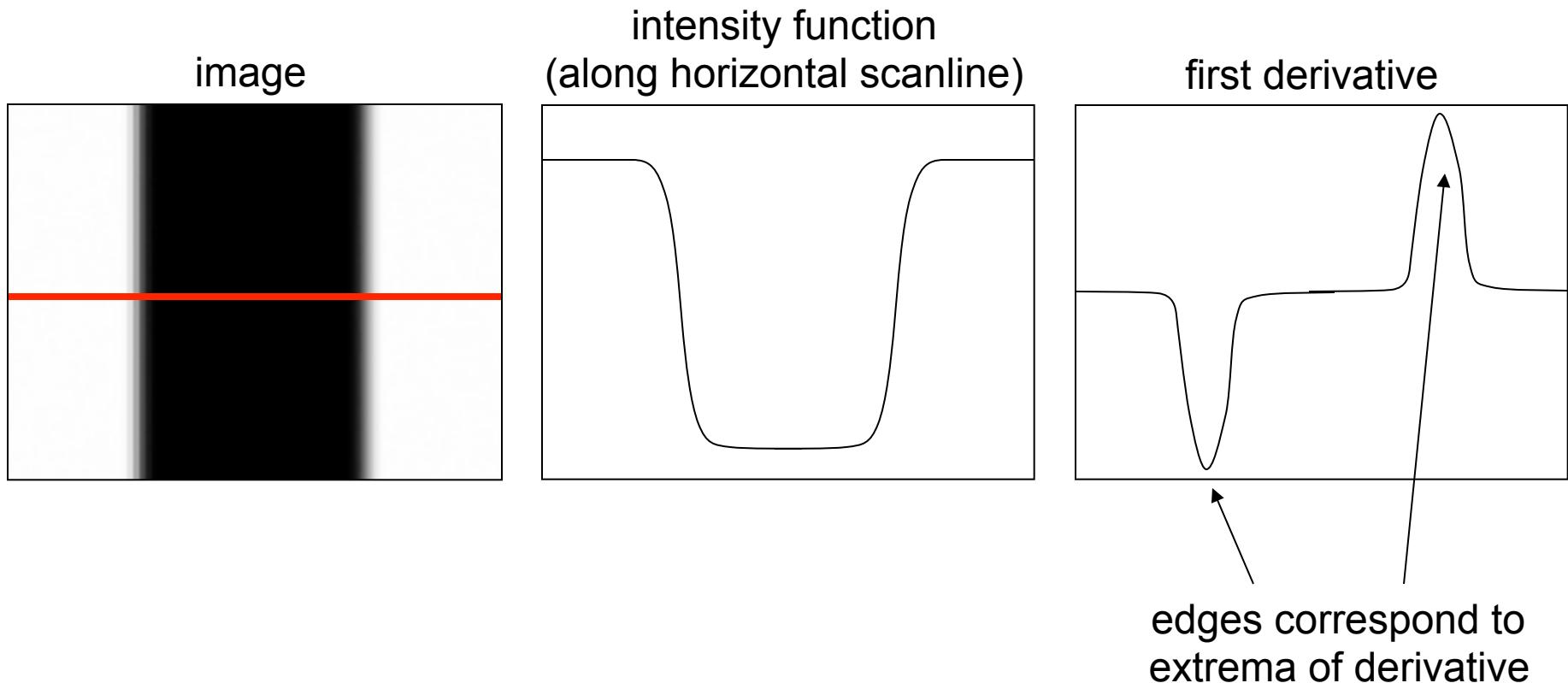


# What we will learn today

- Edge detection
- A simple edge detector
- Canny edge detector
- A model fitting method for edge detection
  - RANSAC

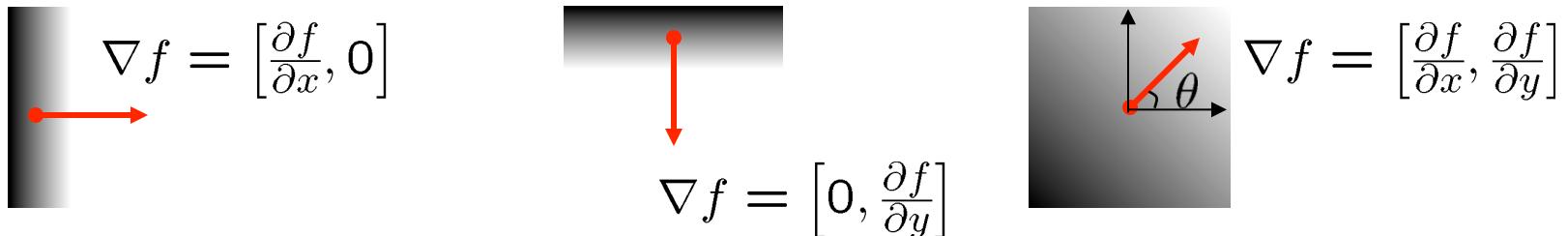
# Characterizing edges

- An edge is a place of rapid change in the image intensity function



# Image gradient

- The gradient of an image:  $\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$



The gradient vector points in the direction of most rapid increase in intensity

The gradient direction is given by  $\theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$

- how does this relate to the direction of the edge?

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

# Finite differences: example

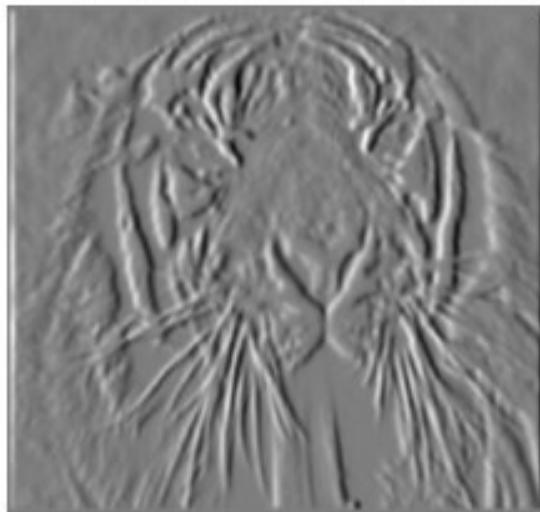
Original  
Image



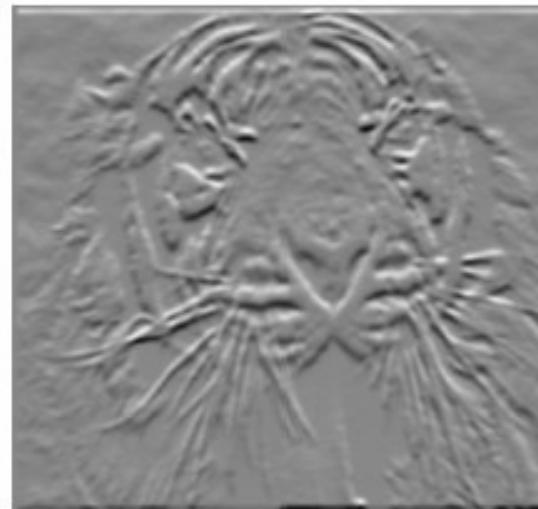
Gradient  
magnitude



x-direction

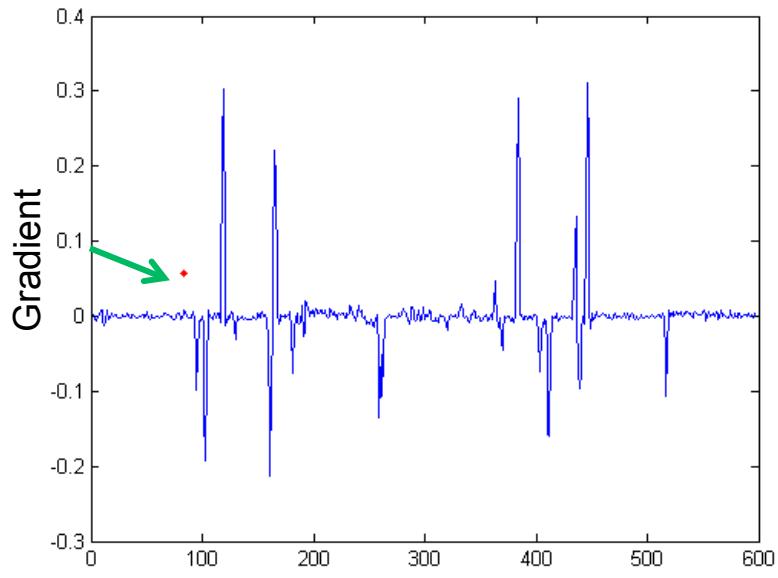
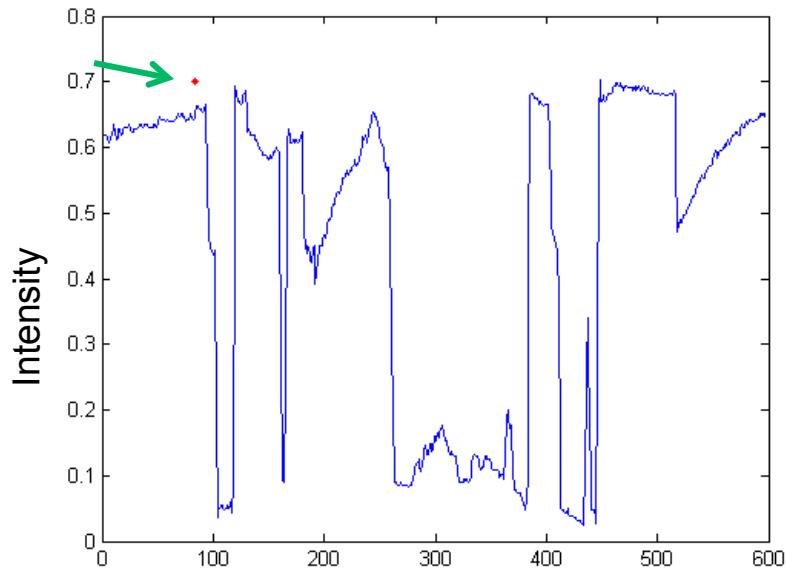
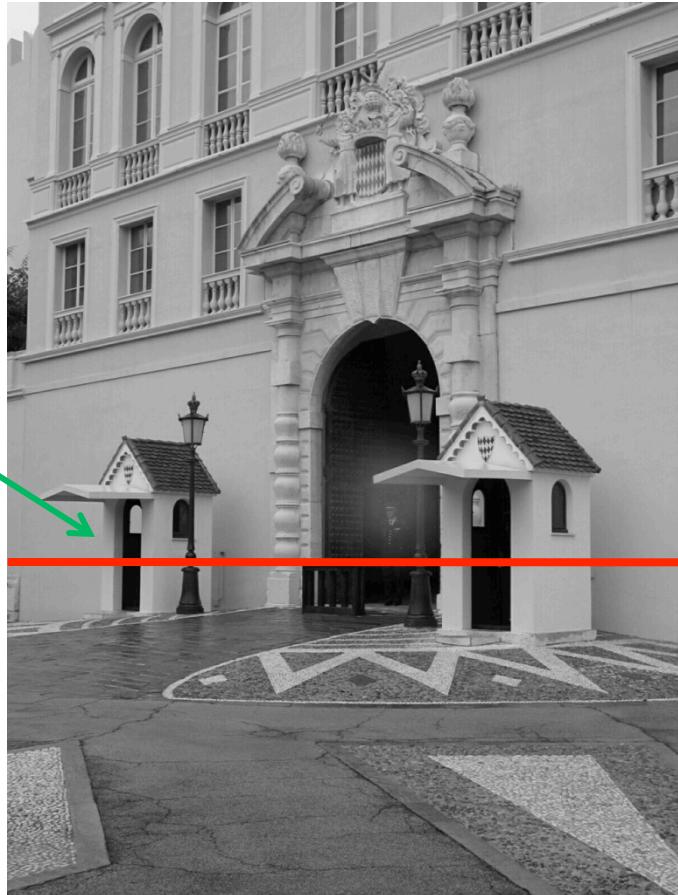


y-direction



- Which one is the gradient in the x-direction? How about y-direction?

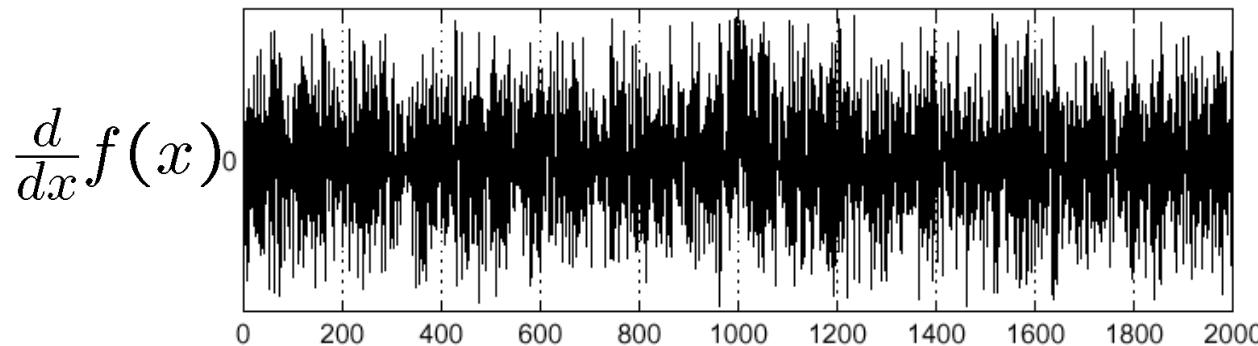
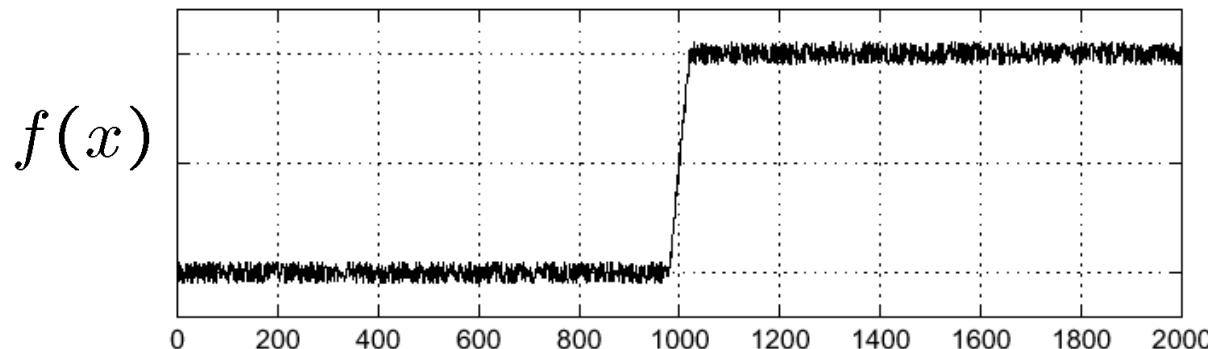
# Intensity profile



Source: D. Hoiem

# Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



Where is the edge?

Source: S. Seitz

# Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?

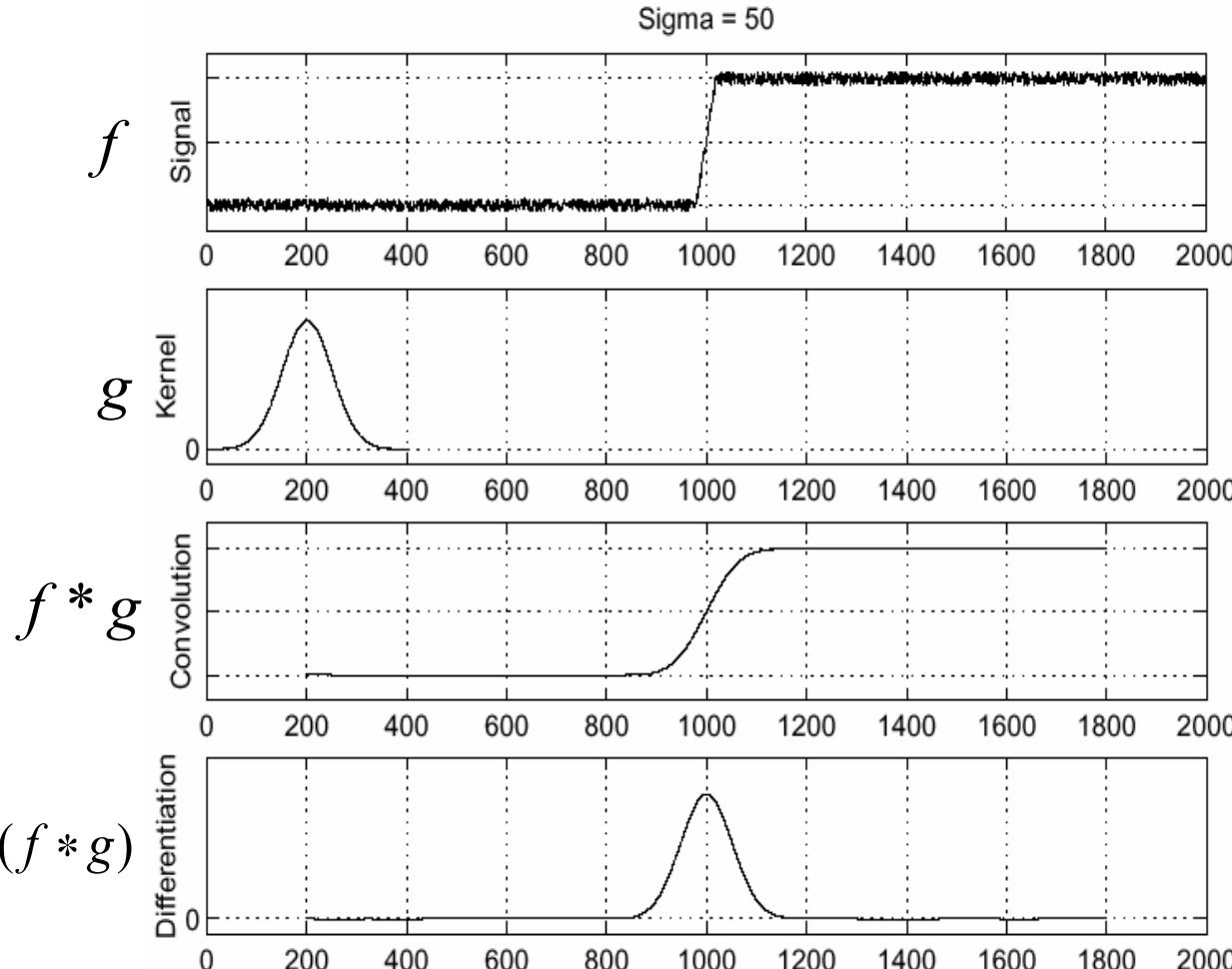
Source: D. Forsyth

# Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth

# Solution: smooth first



- To find edges, look for peaks in

$$\frac{d}{dx}(f * g)$$

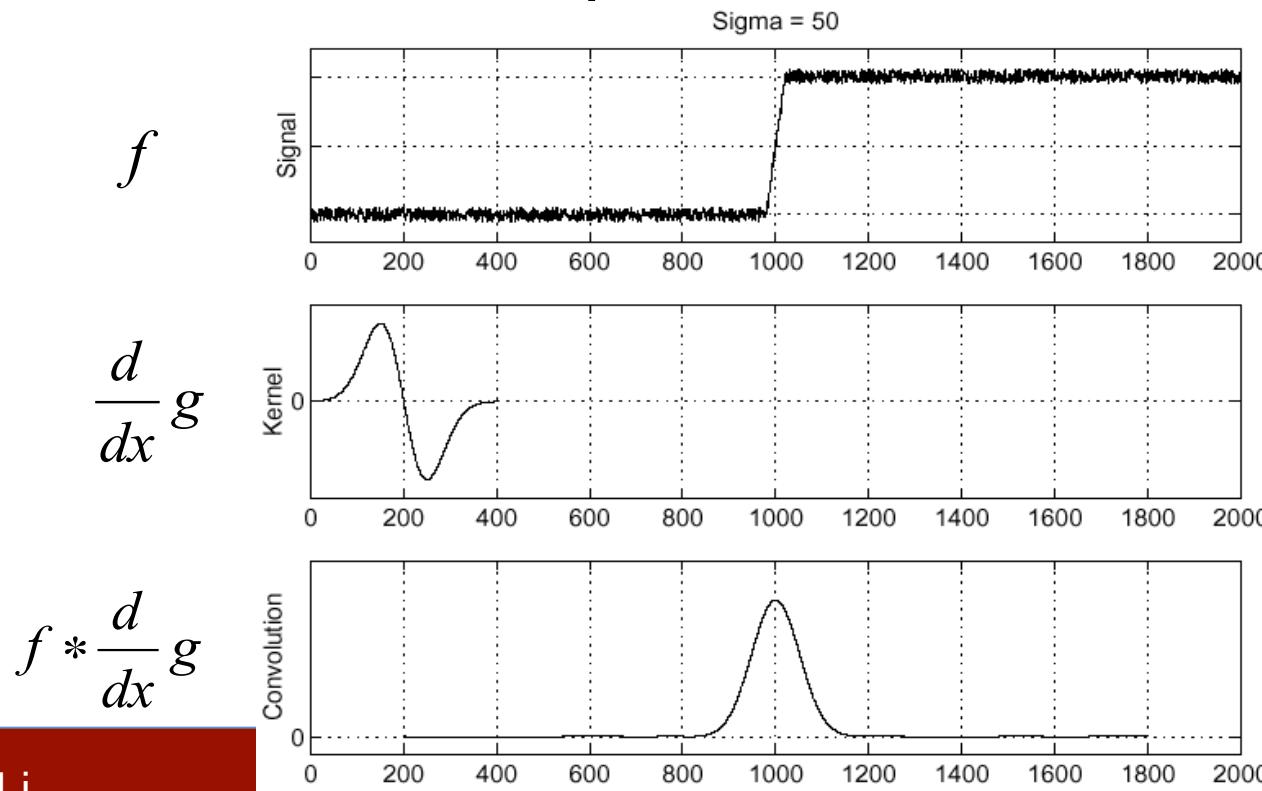
Source: S. Seitz

# Derivative theorem of convolution

- This theorem gives us a very useful property:

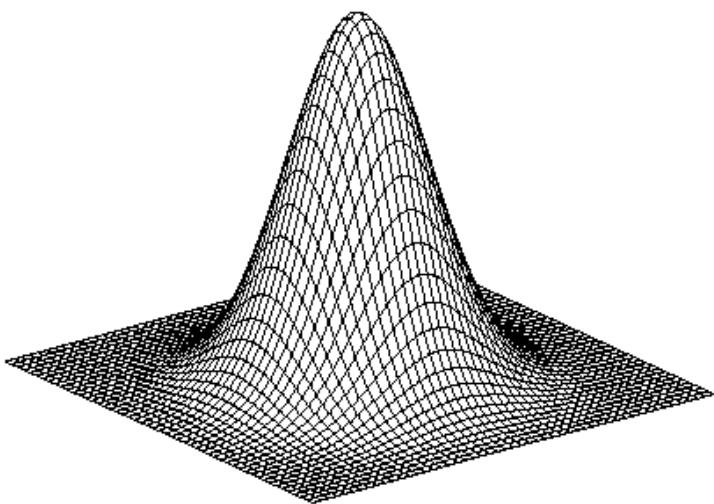
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$$

- This saves us one operation:



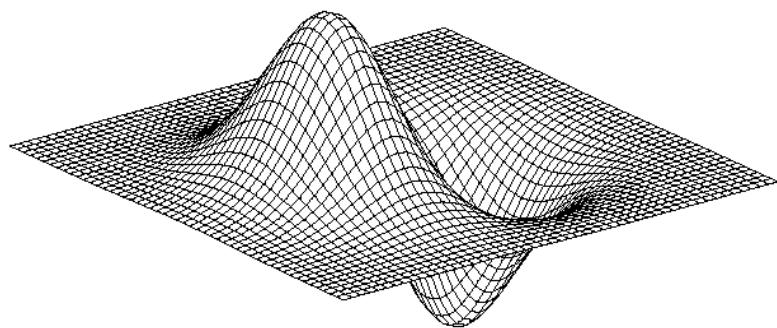
Source: S. Seitz

# Derivative of Gaussian filter



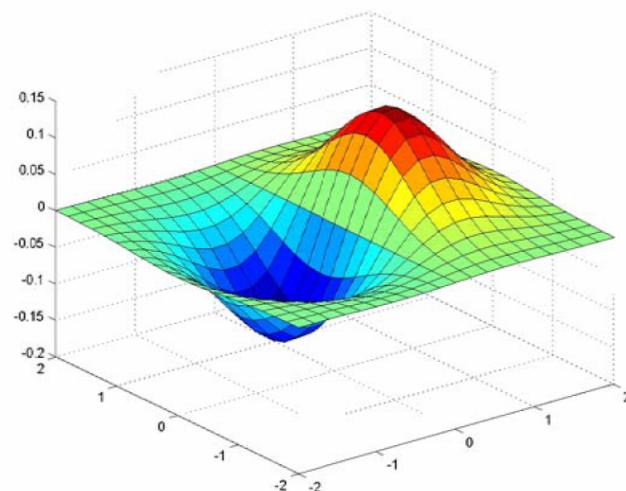
2D-gaussian

$$* [1 \ -1] =$$

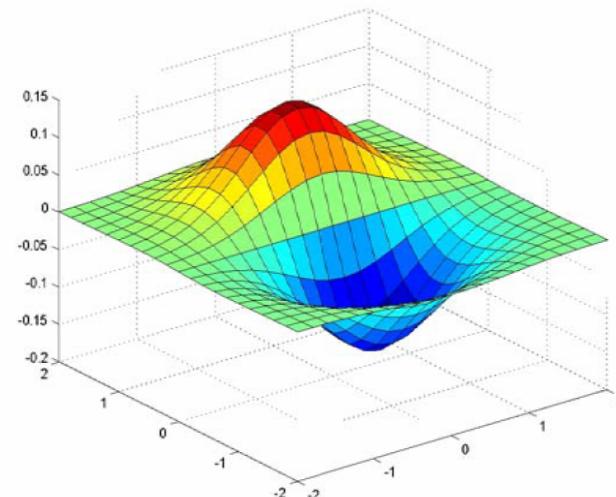


x - derivative

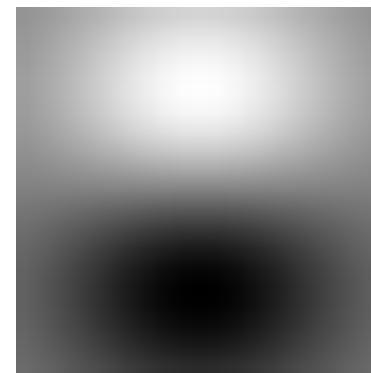
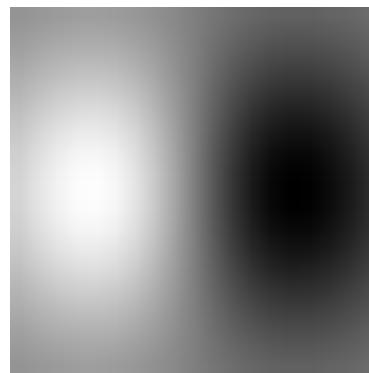
# Derivative of Gaussian filter



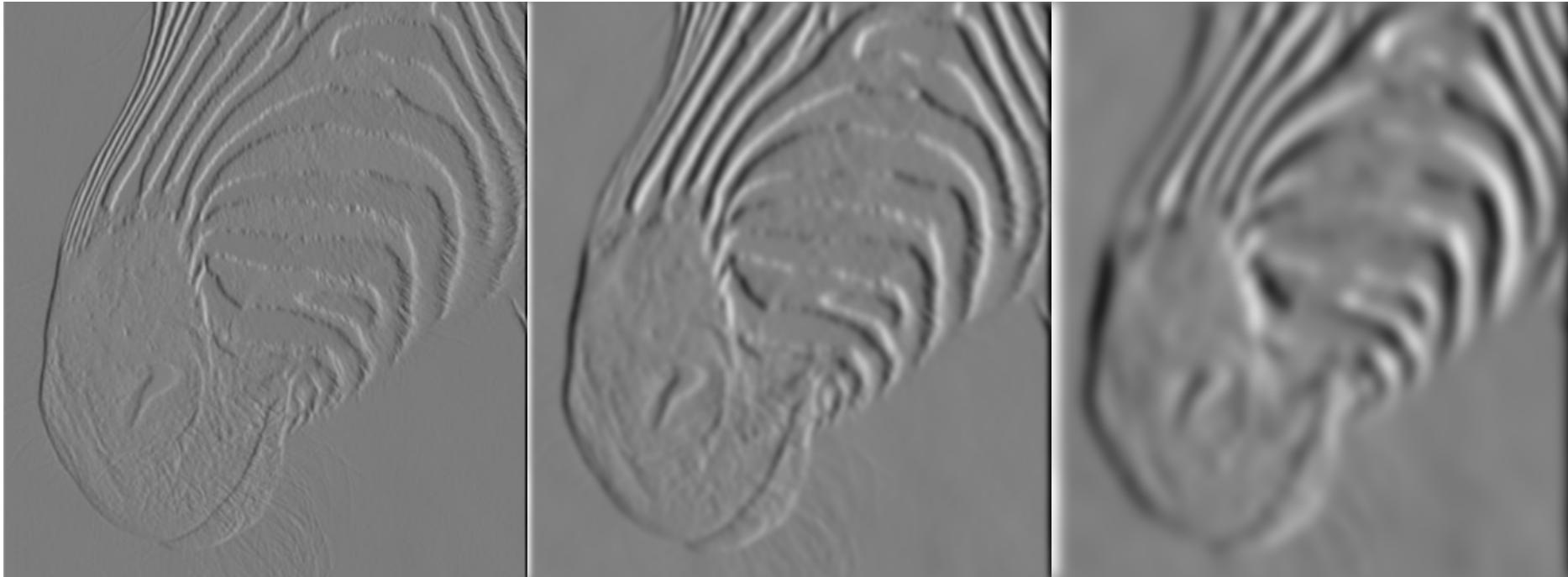
x-direction



y-direction



# Tradeoff between smoothing and localization



1 pixel

3 pixels

7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth

# Implementation issues

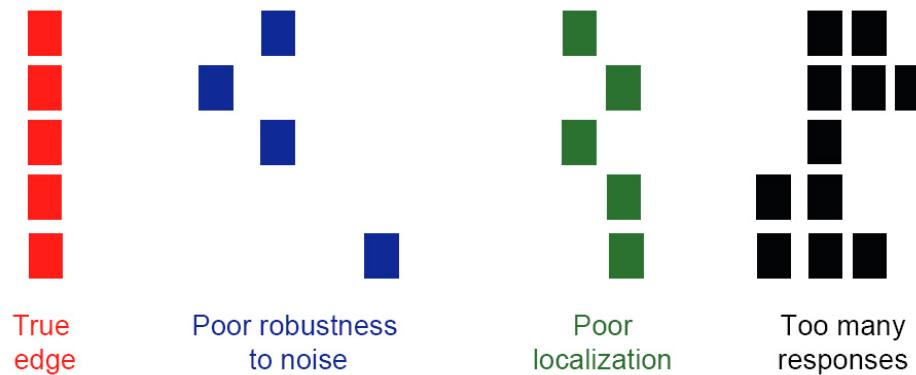


- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth

# Designing an edge detector

- Criteria for an “optimal” edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



# What we will learn today

- Edge detection
- A simple edge detector
- **Canny edge detector**
- A model fitting method for edge detection
  - RANSAC

# Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

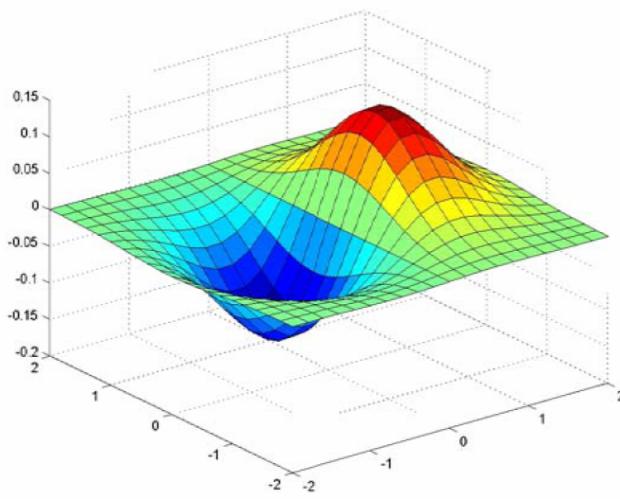
J. Canny, [\*A Computational Approach To Edge Detection\*](#), IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

# Example

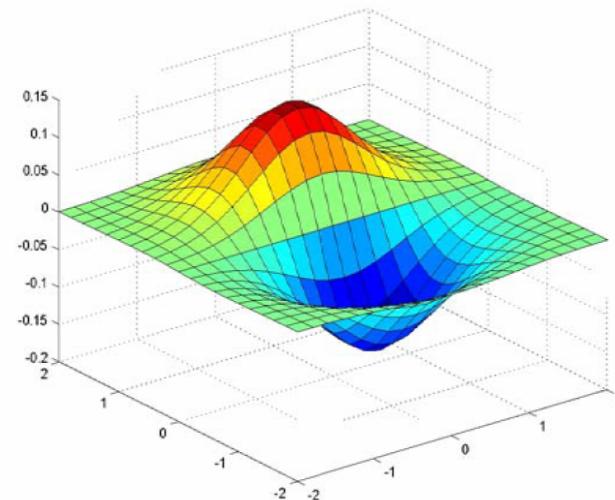


- original image (Lena)

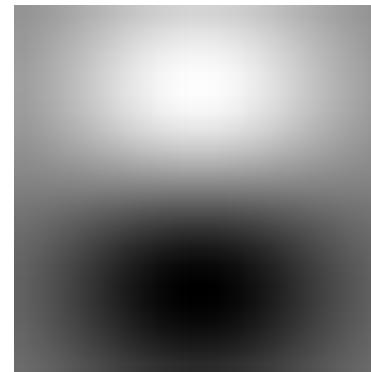
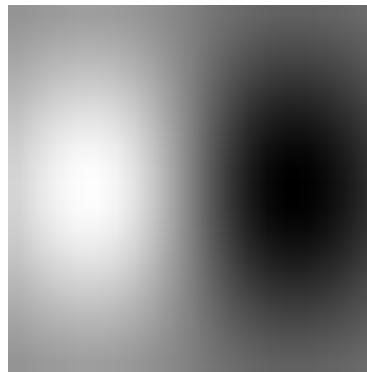
# Derivative of Gaussian filter



x-direction



y-direction



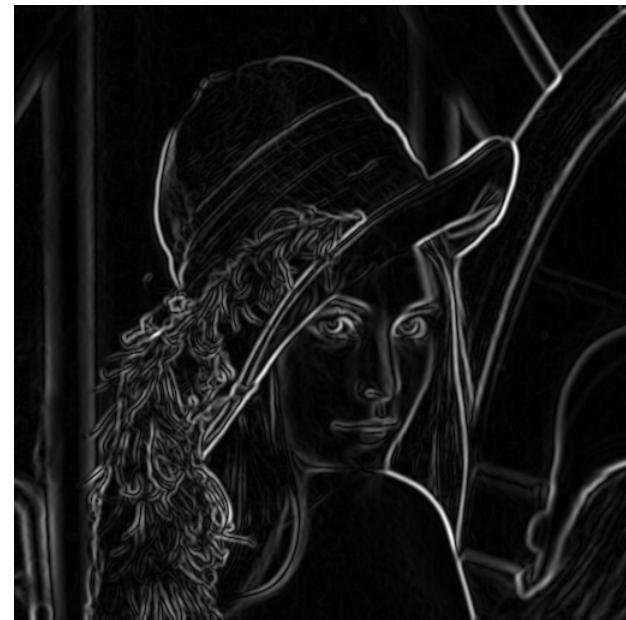
# Compute Gradients (DoG)



X-Derivative of Gaussian



Y-Derivative of Gaussian



Gradient Magnitude

# Get Orientation at Each Pixel

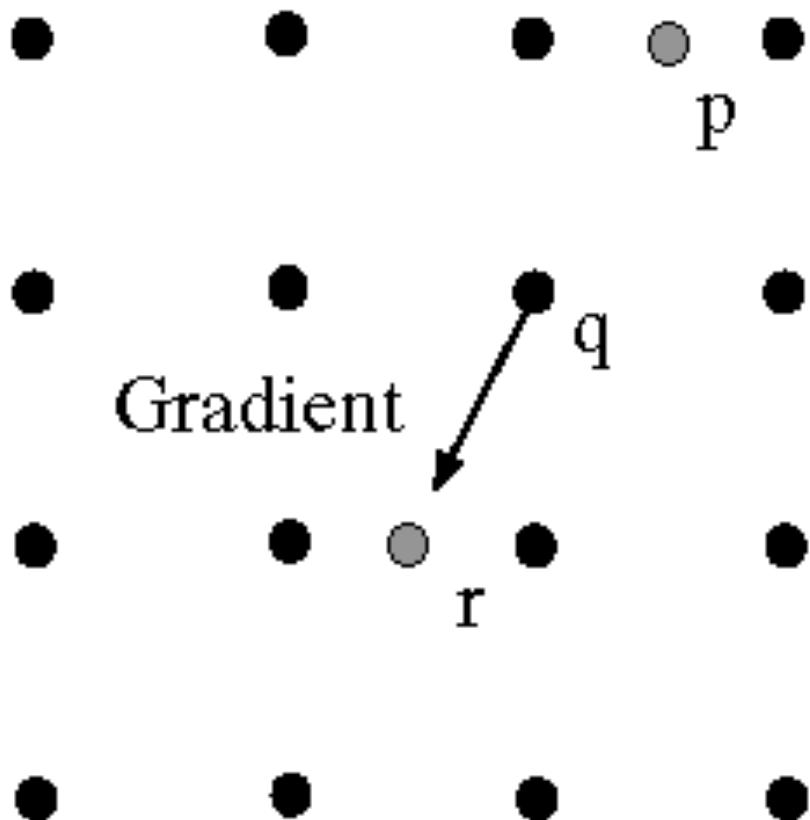
- Threshold at minimum level



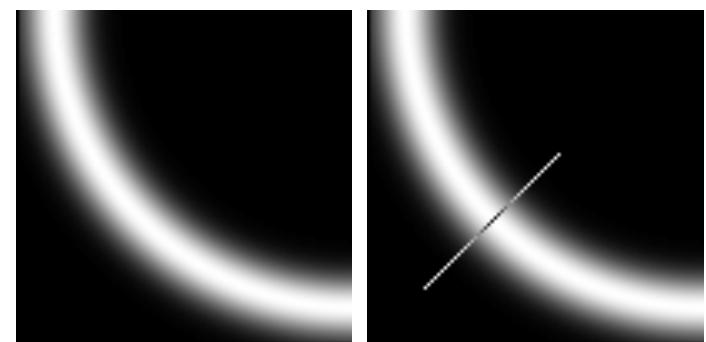
$\theta = \text{atan2}(gy, gx)$

Source: J. Hayes

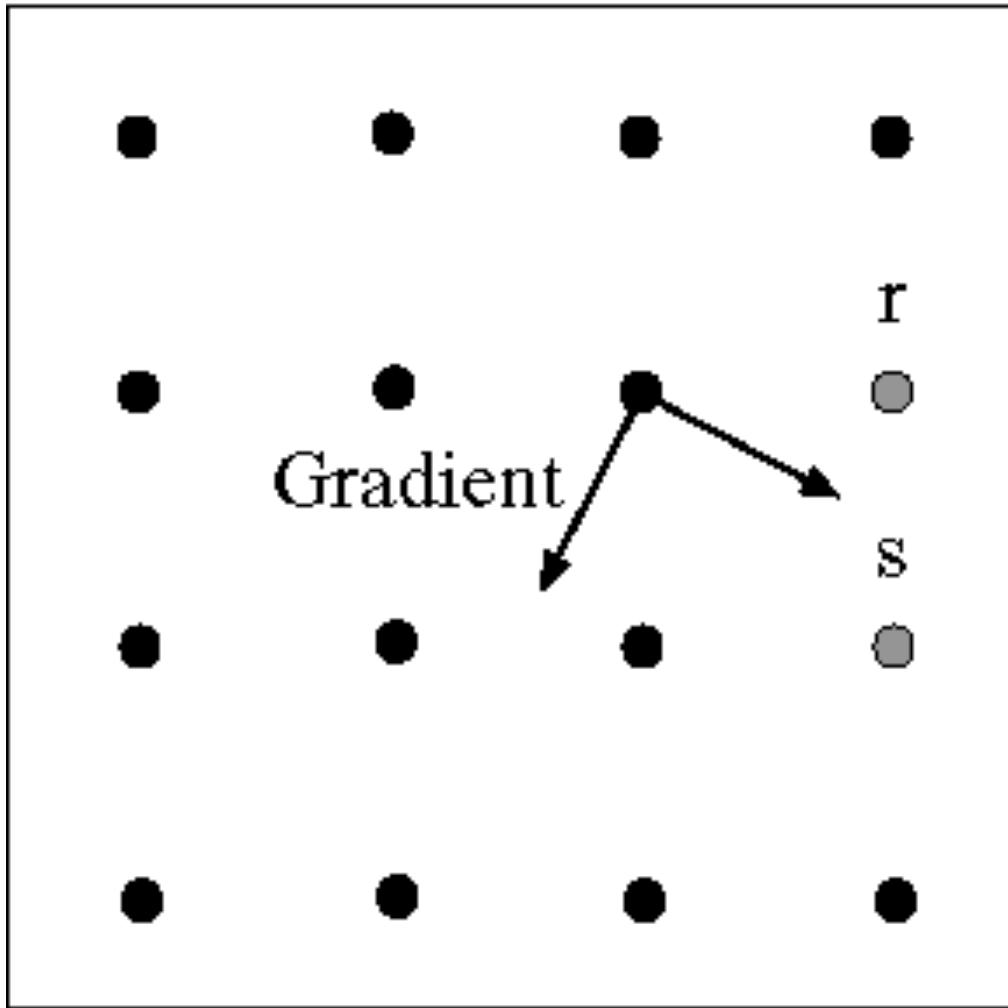
# Non-maximum suppression



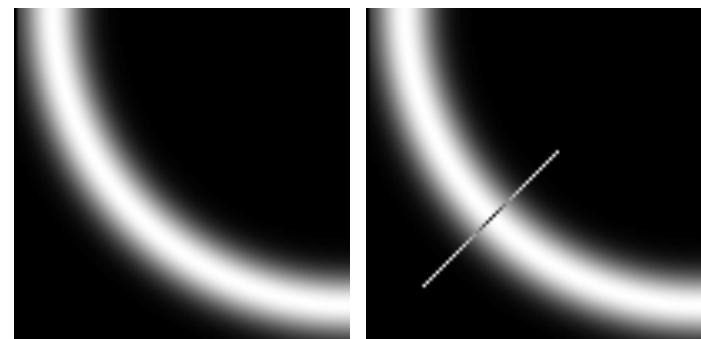
At q, we have a maximum if the value is larger than those at both p and at r.  
Interpolate to get these values.



# Edge linking



Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Source: D. Forsyth

# Before Non-max Suppression



Source: J. Hayes

# After non-max suppression



Source: J. Hayes

# Hysteresis thresholding

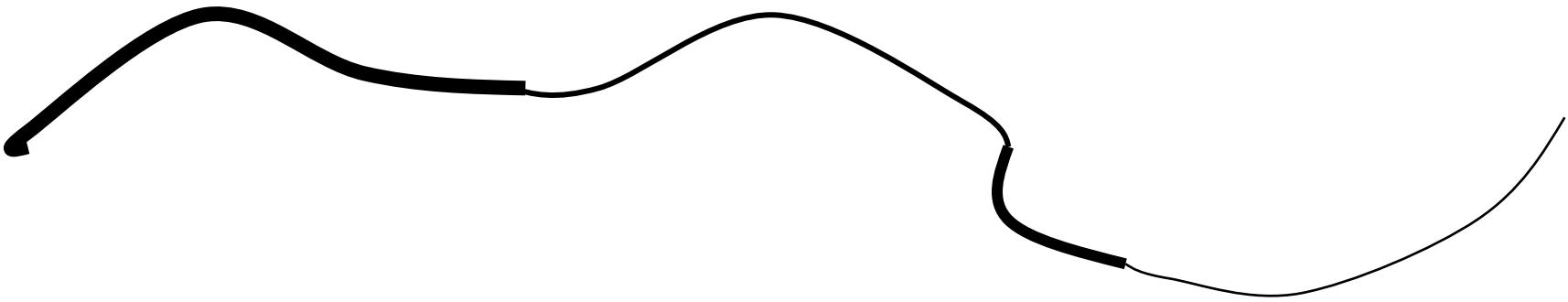
- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels



Source: J. Hayes

# Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Source: S. Seitz

# Final Canny Edges



# Canny edge detector

1. Filter image with x, y derivatives of Gaussian
  2. Find magnitude and orientation of gradient
  3. Non-maximum suppression:
    - Thin multi-pixel wide “ridges” down to single pixel width
  4. Thresholding and linking (hysteresis):
    - Define two thresholds: low and high
    - Use the high threshold to start edge curves and the low threshold to continue them
- 
- MATLAB: `edge(image, ‘canny’ )`

# Effect of $\sigma$ (Gaussian kernel spread/size)



original



Canny with  $\sigma = 1$

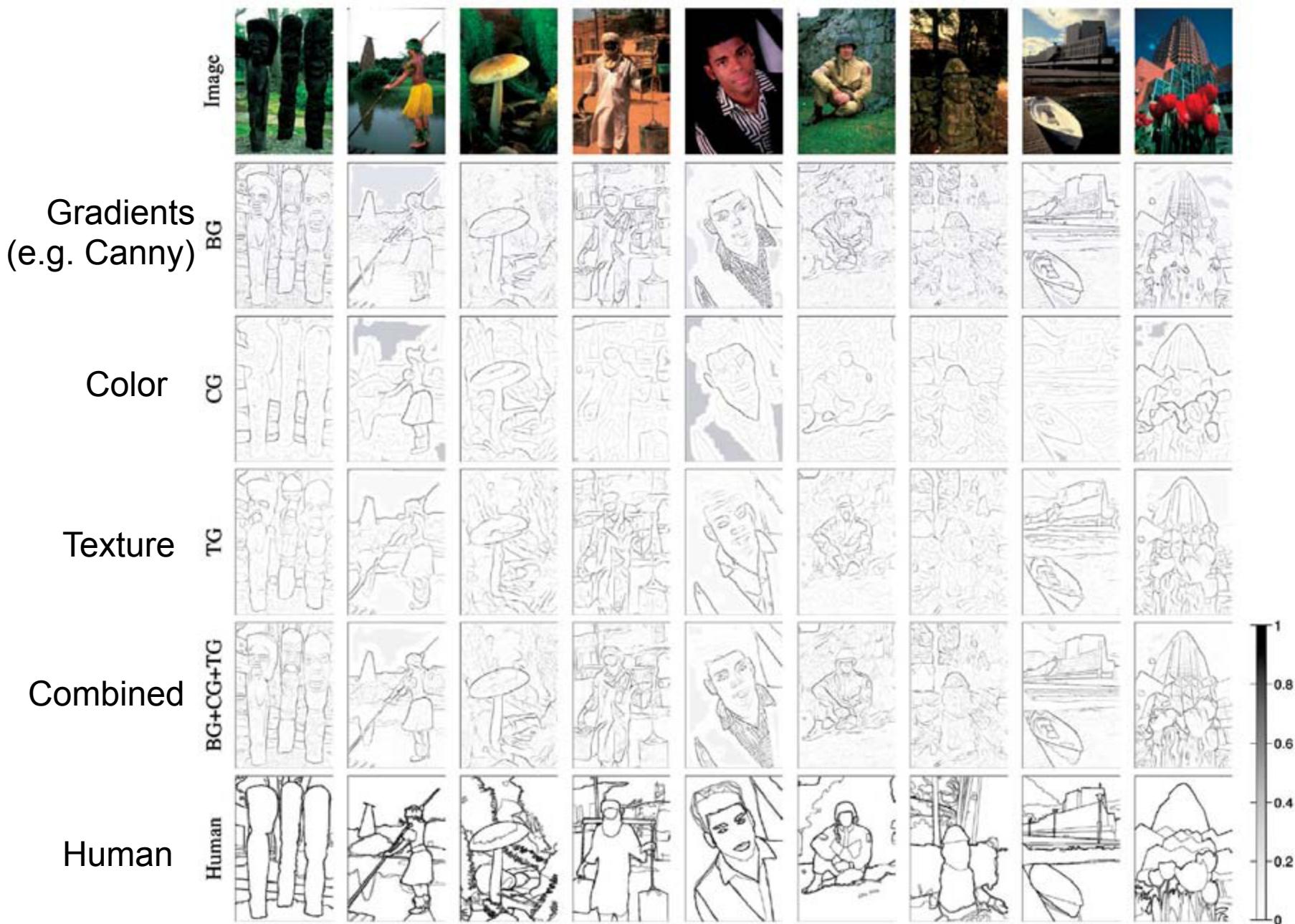


Canny with  $\sigma = 2$

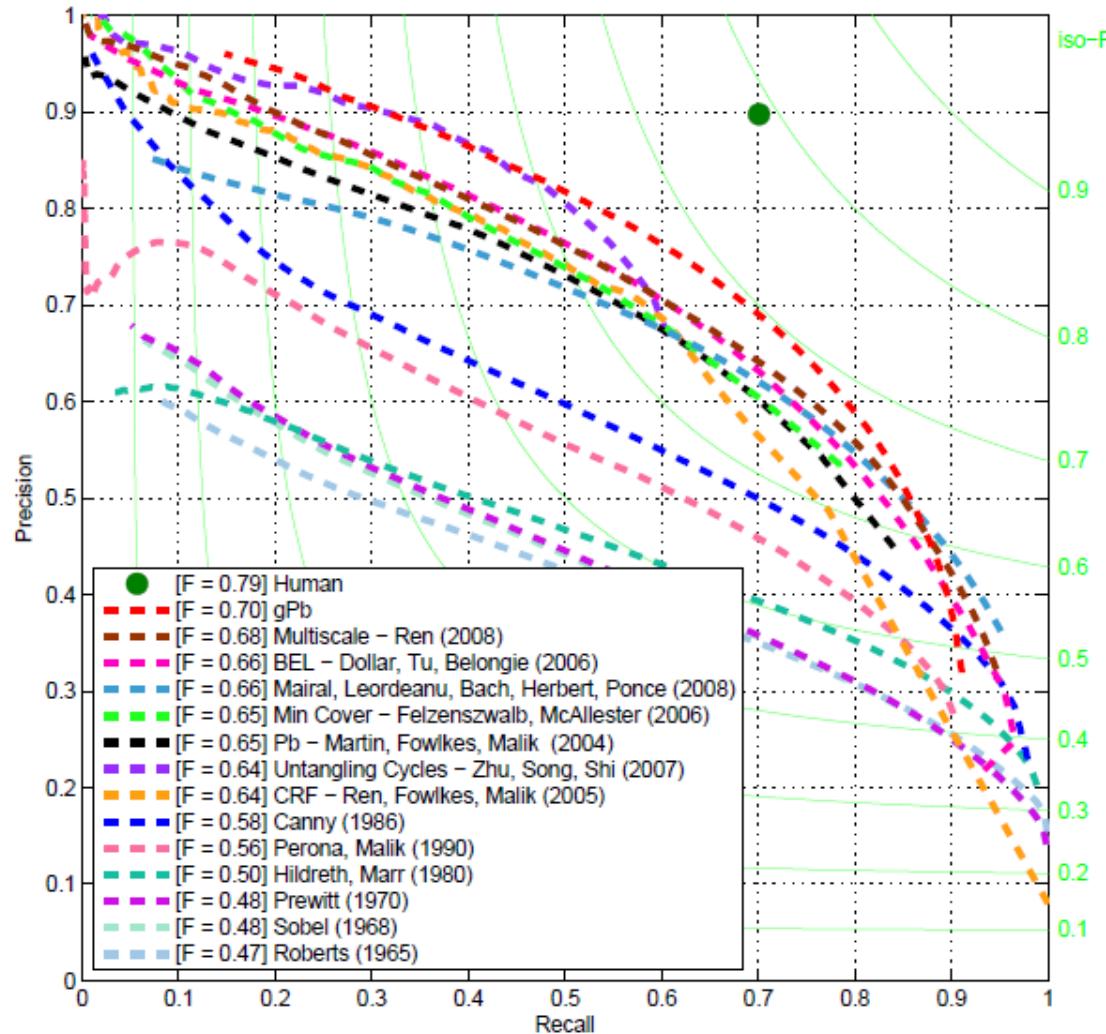
The choice of  $\sigma$  depends on desired behavior

- large  $\sigma$  detects large scale edges
- small  $\sigma$  detects fine features

Source: S. Seitz



# 45 years of boundary detection



Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)

# What we will learn today

- Edge detection
- A simple edge detector
- Canny edge detector
- A model fitting method for edge detection
  - RANSAC

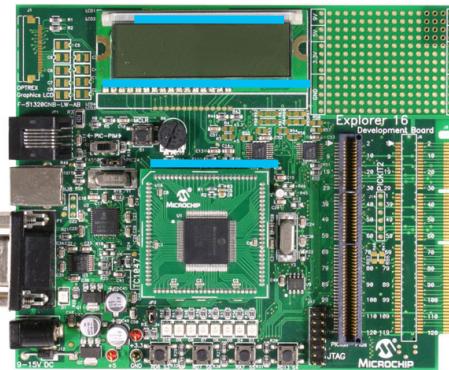
# Fitting as Search in Parametric Space

- 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

Source: L. Lazebnik

# Example: Line Fitting

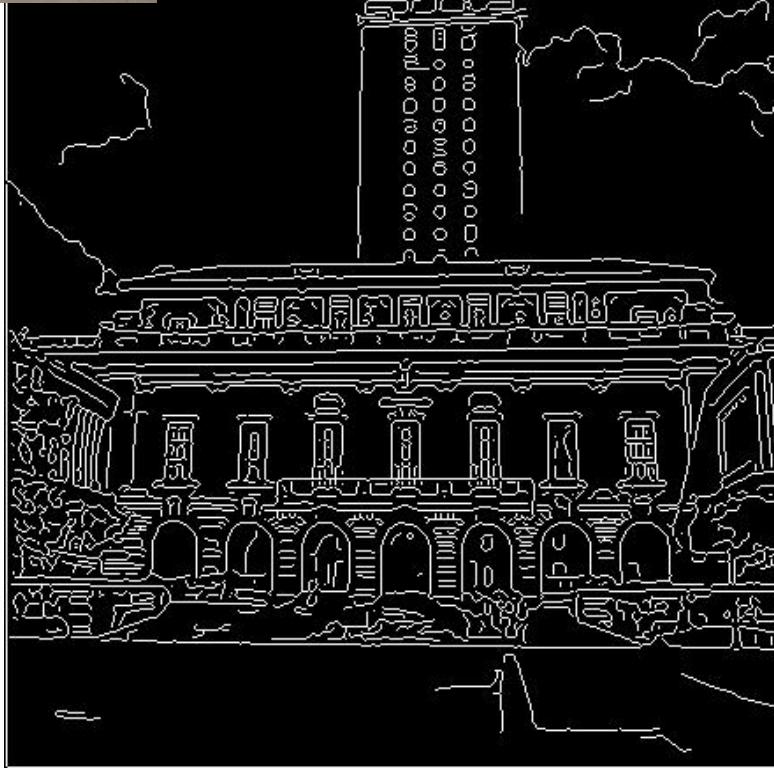
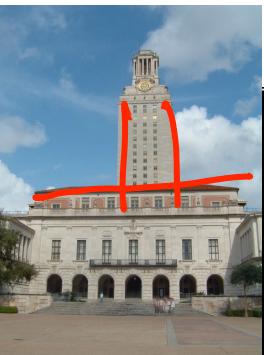
- Why fit lines?  
Many objects characterized by presence of straight lines



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

Slide credit: Kristen Grauman

# 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?

Slide credit: Kristen Grauman

# Voting

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- Voting is a general technique where we let the features vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, *but* typically their votes should be inconsistent with the majority of “good” features.
- Ok if some features not observed, as model can span multiple fragments.

Slide credit: Kristen Grauman

# RANSAC [Fischler & Bolles 1981]

- 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

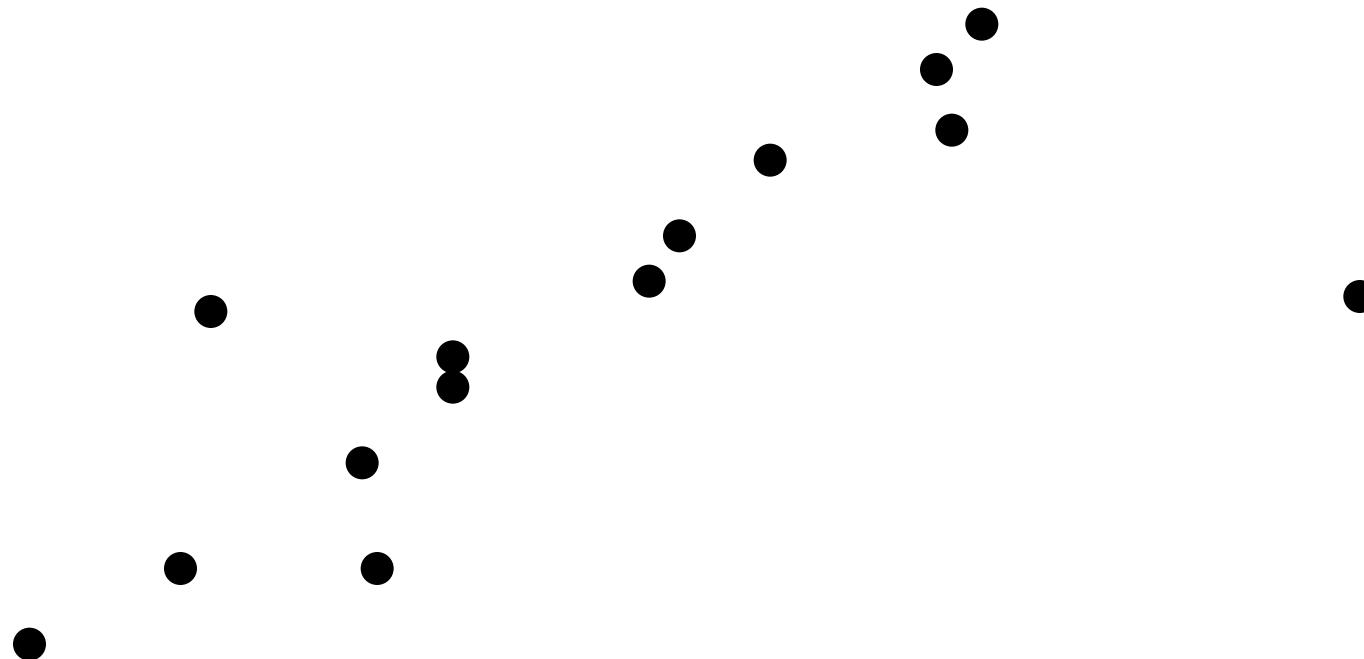
[Fischler & Bolles 1981]

## 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

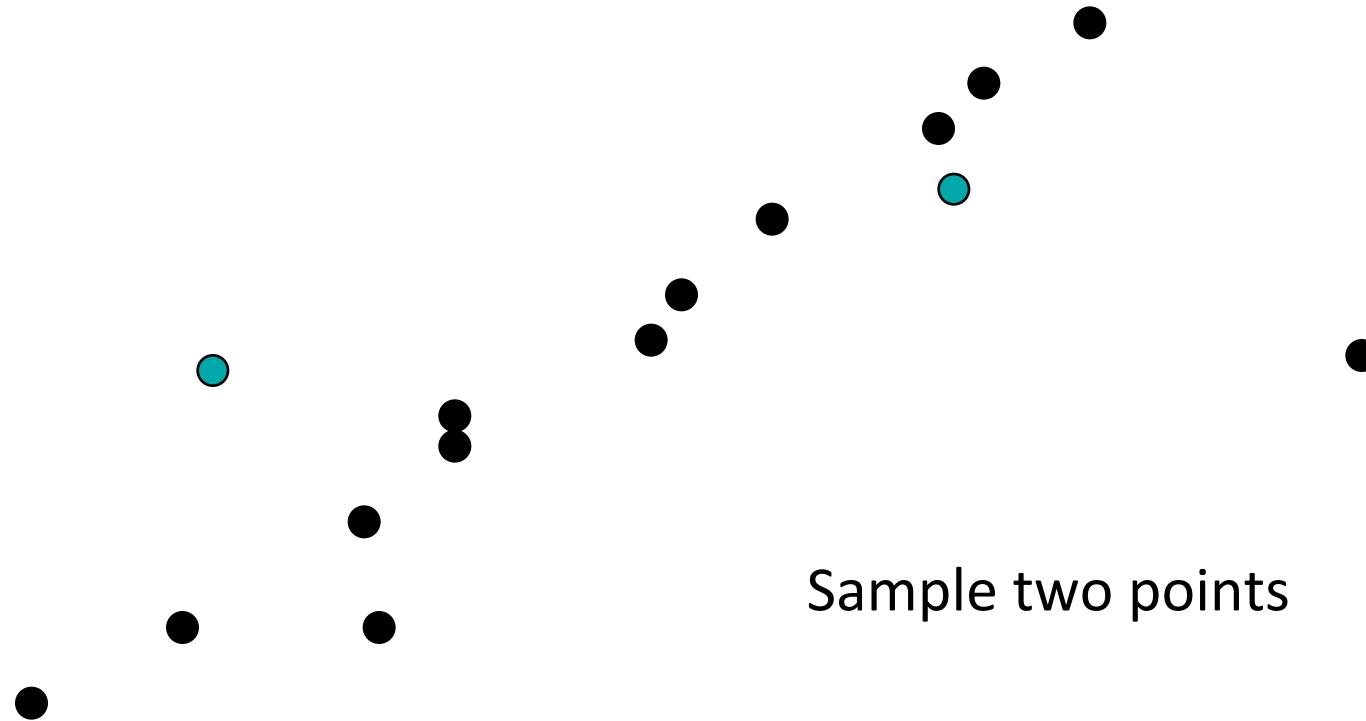
# RANSAC Line Fitting Example

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



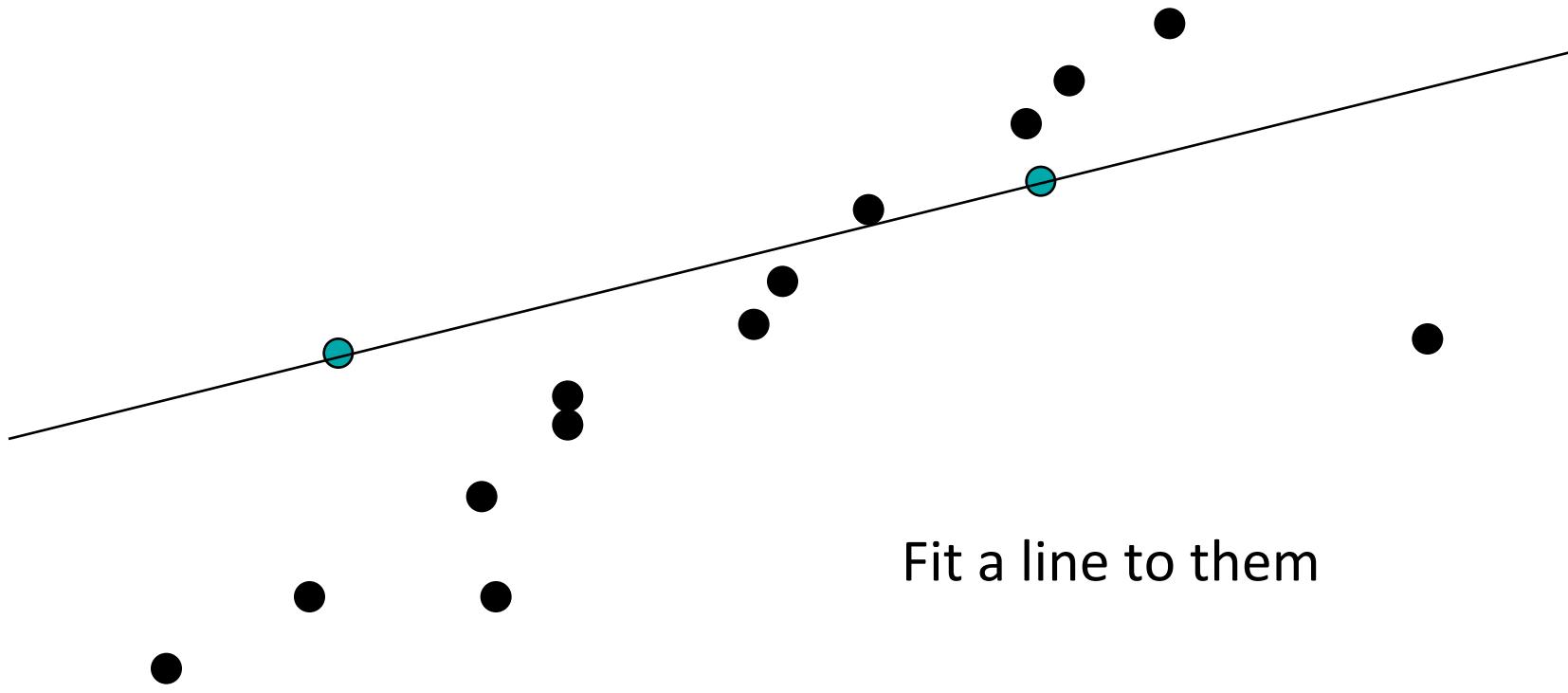
# RANSAC Line Fitting Example

- Task: Estimate the best line



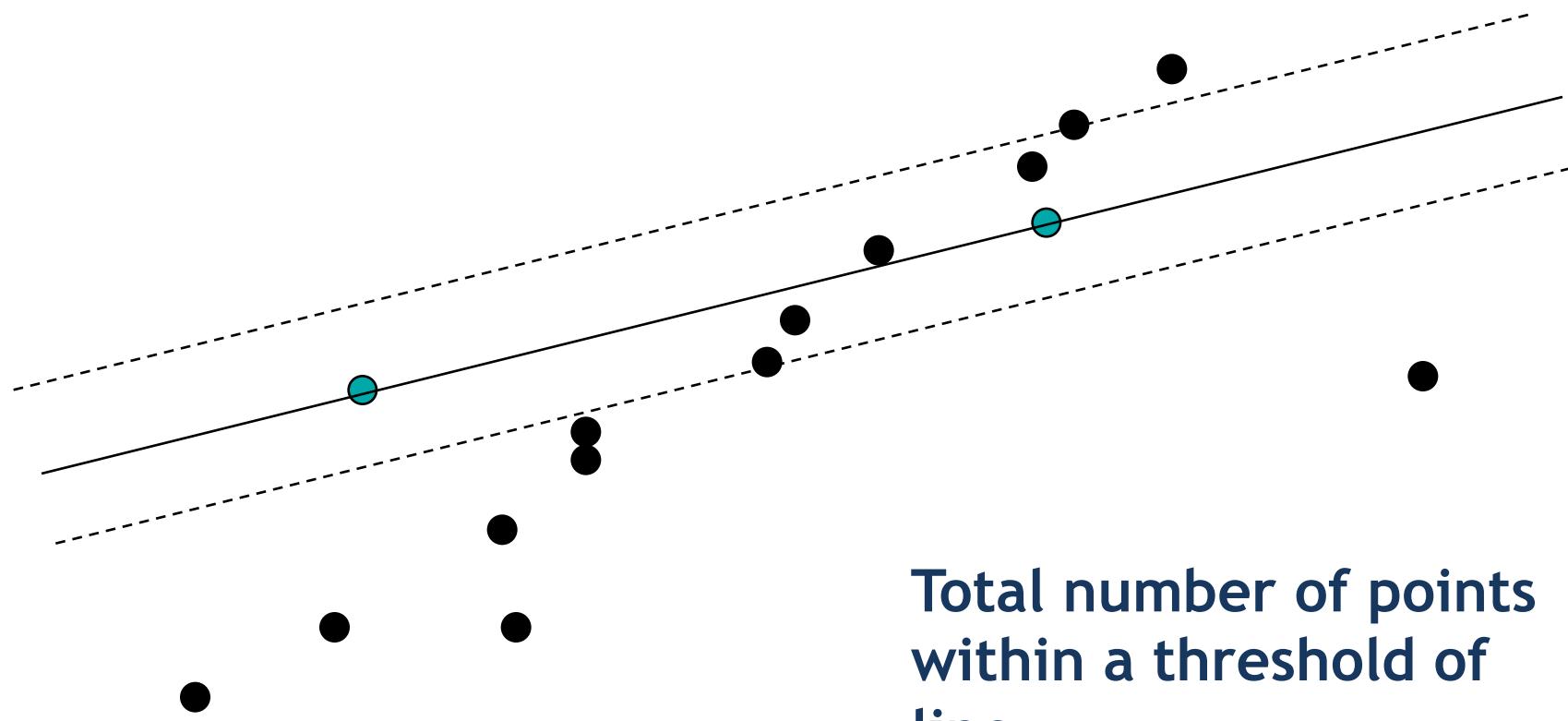
# RANSAC Line Fitting Example

- Task: Estimate the best line



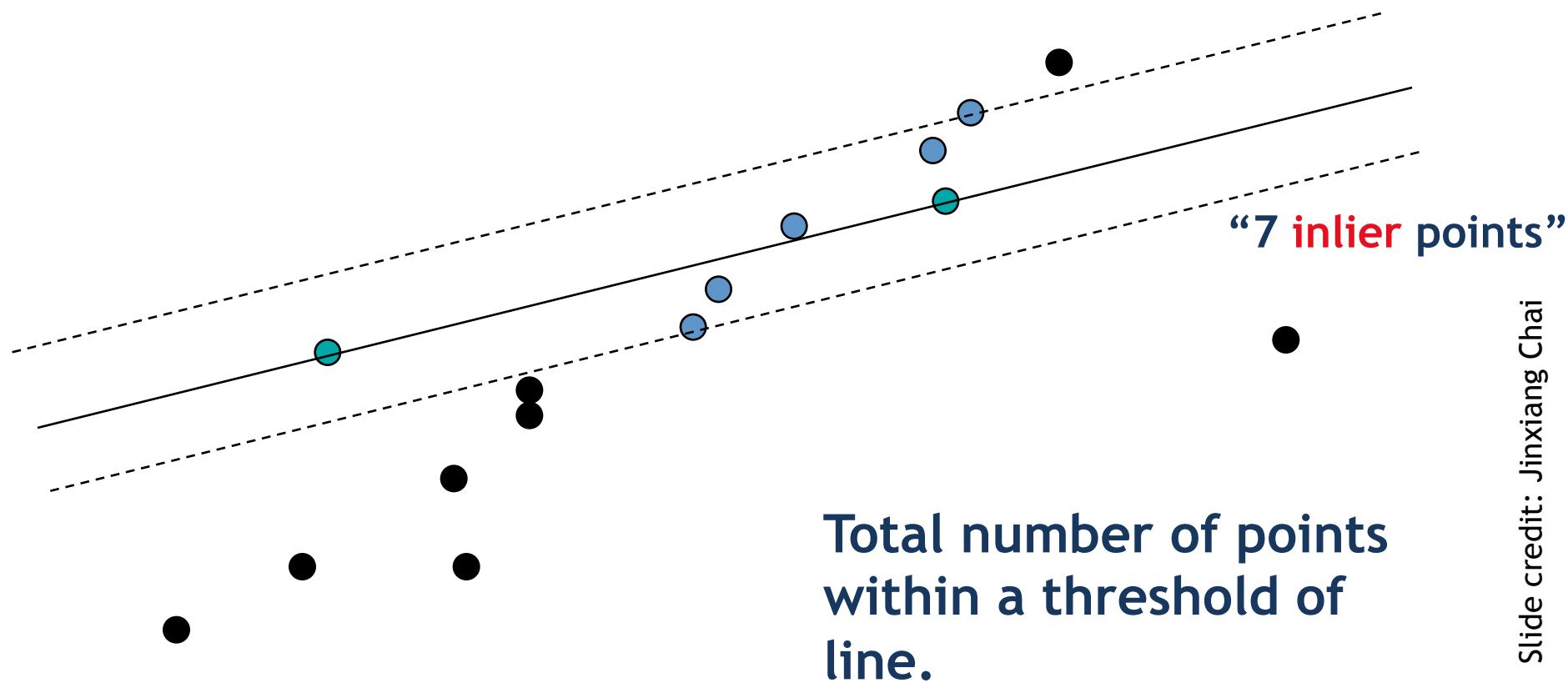
# RANSAC Line Fitting Example

- Task: Estimate the best line



# RANSAC Line Fitting Example

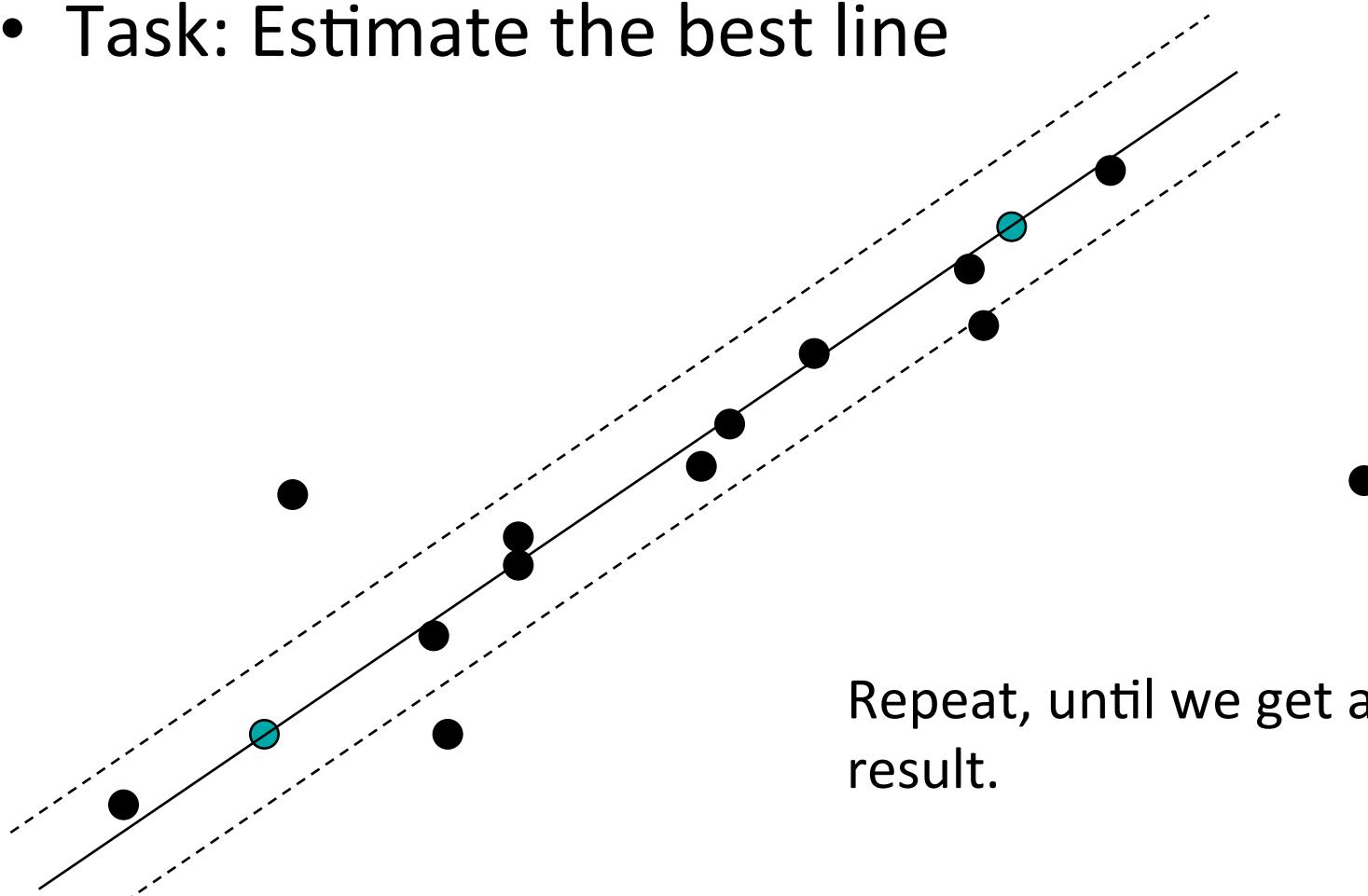
- Task: Estimate the best line



Slide credit: Jinxiang Chai

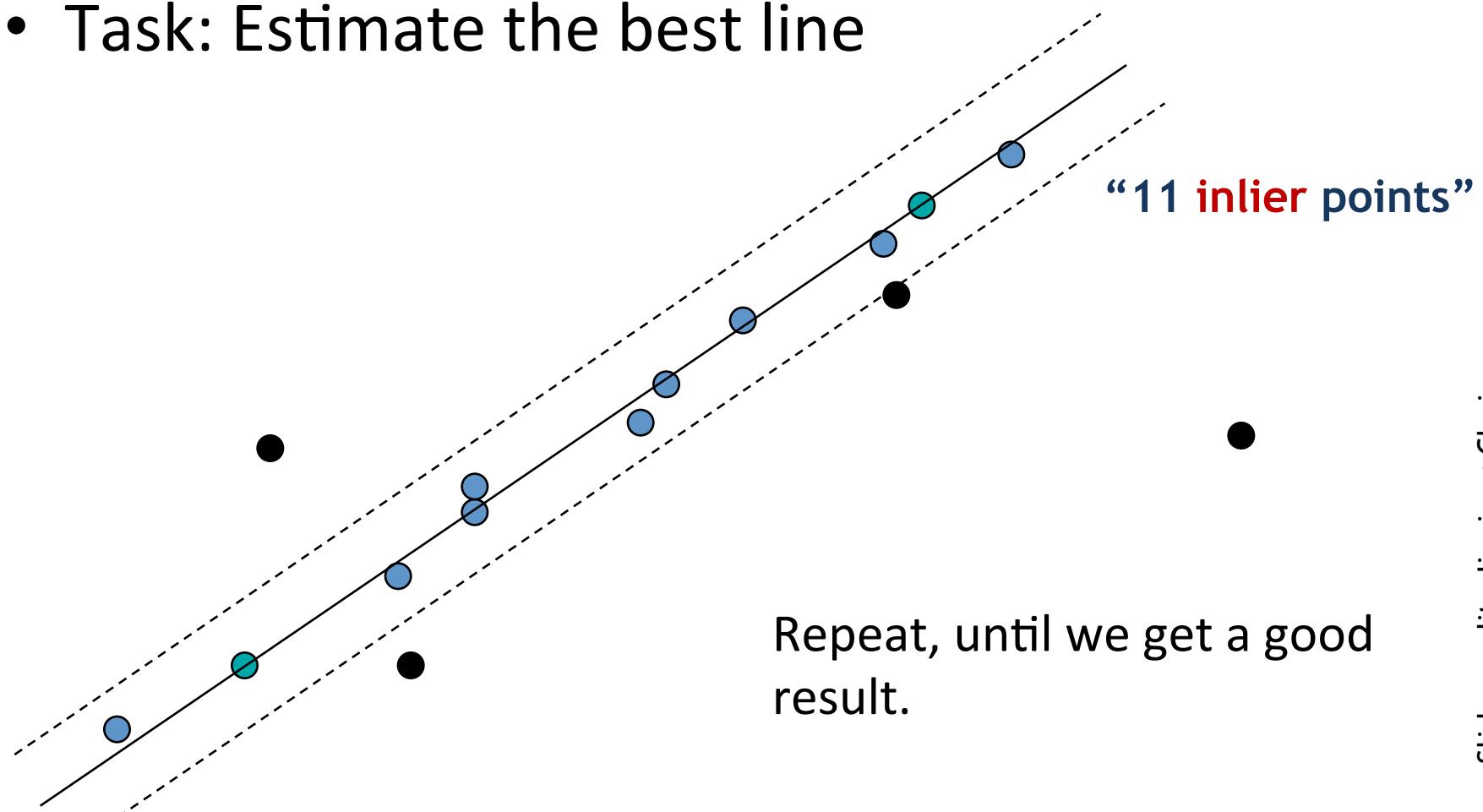
# RANSAC Line Fitting Example

- Task: Estimate the best line



# RANSAC Line Fitting Example

- Task: Estimate the best line



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

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

Slide credit: David Lowe

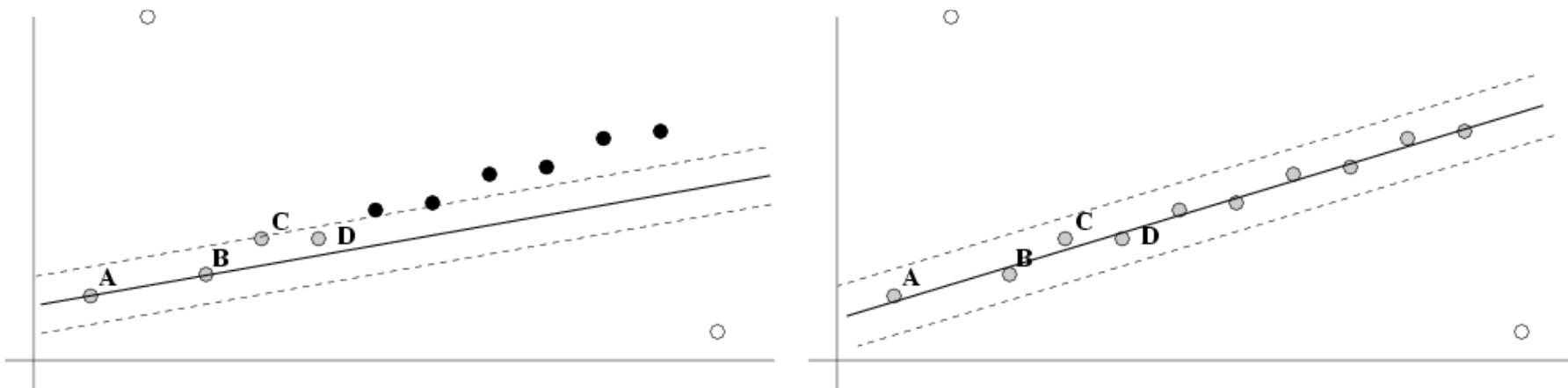
# RANSAC: Computed k ( $p=0.99$ )

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

Slide credit: David Lowe

# 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.



Slide credit: David Lowe

# RANSAC: Pros and Cons

- **Pros:**
  - General method suited for a wide range of model fitting problems
  - Easy to implement and easy to calculate its failure rate
- **Cons:**
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)
- A voting strategy, The Hough transform, can handle high percentage of outliers

# What we have learned today

- Edge detection
- A simple edge detector
- Canny edge detector
- A model fitting method for edge detection
  - RANSAC

Some background reading:

Forsyth and Ponce, Computer Vision, Chapter 8, Chapter 15.5.2