Chapter 7

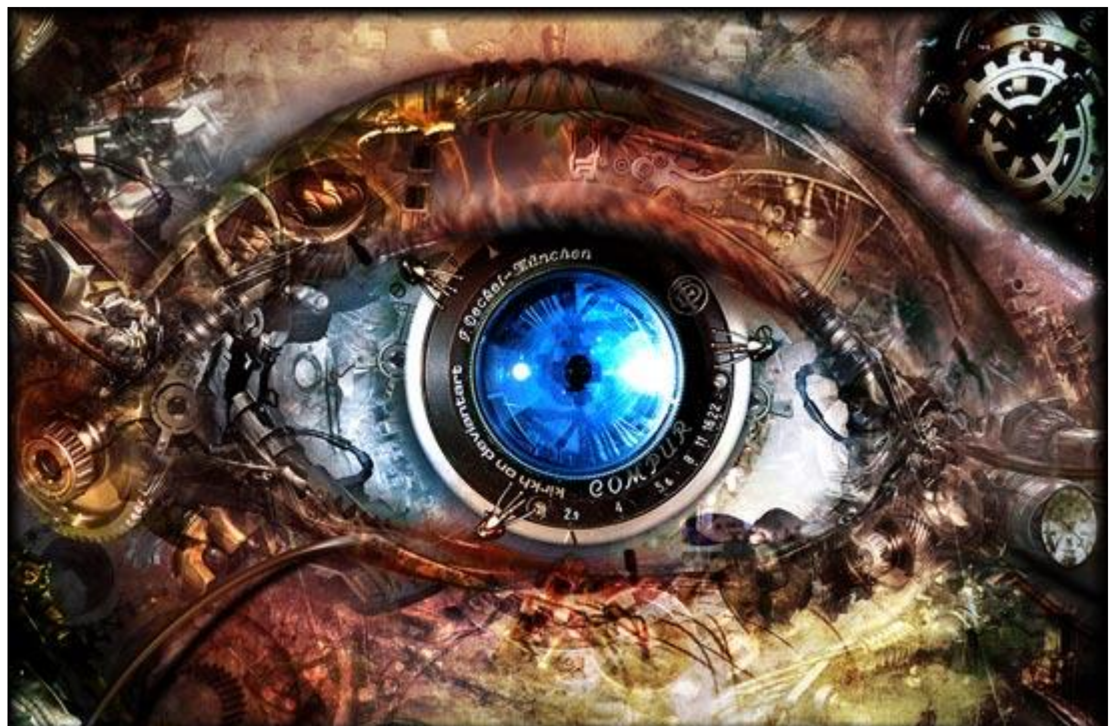
Edge Detection

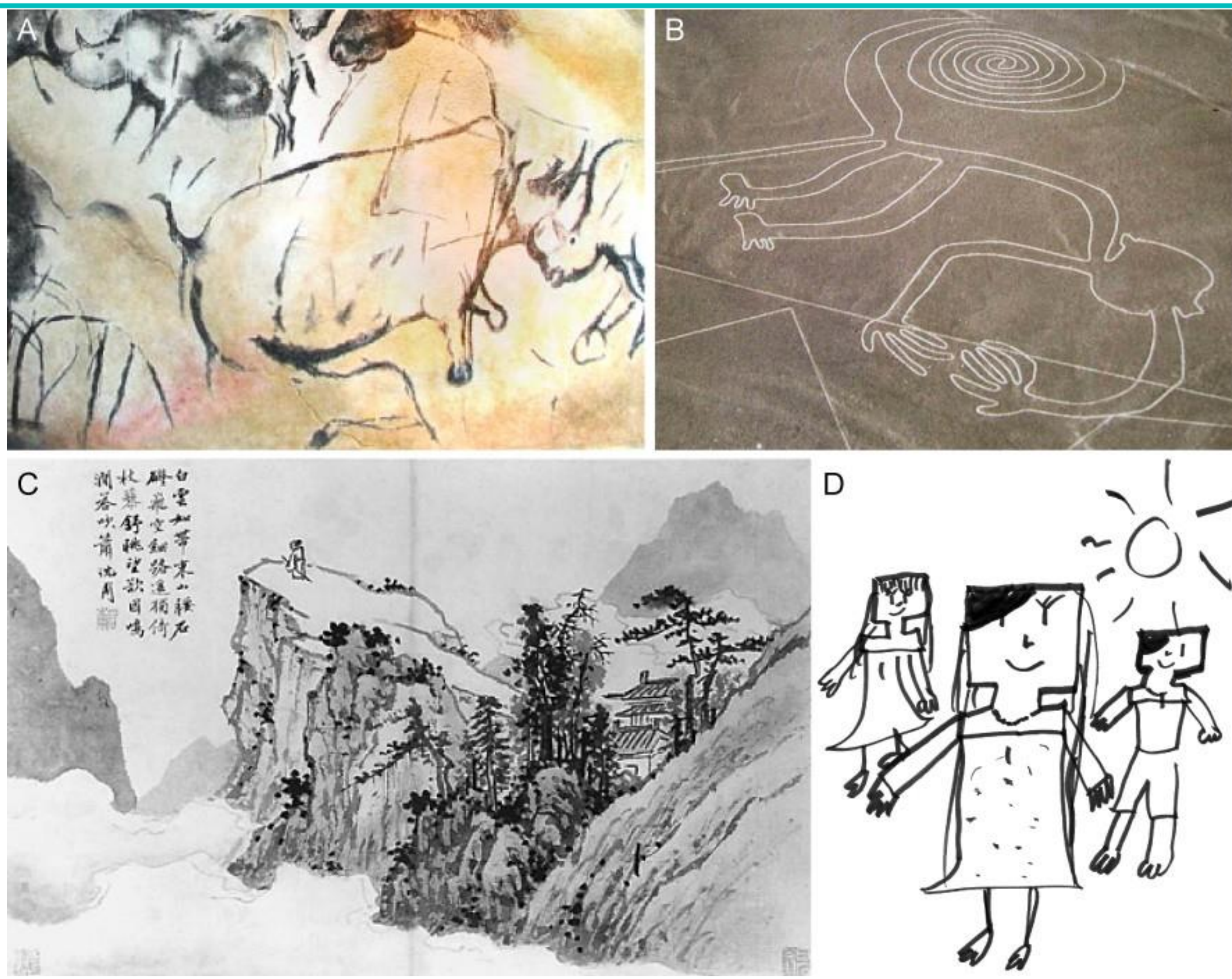
Prof. Fei-Fei Li, Stanford University

Contents

Edge detection

- Canny edge detector





(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 geoplyphs, 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

152 Biederman

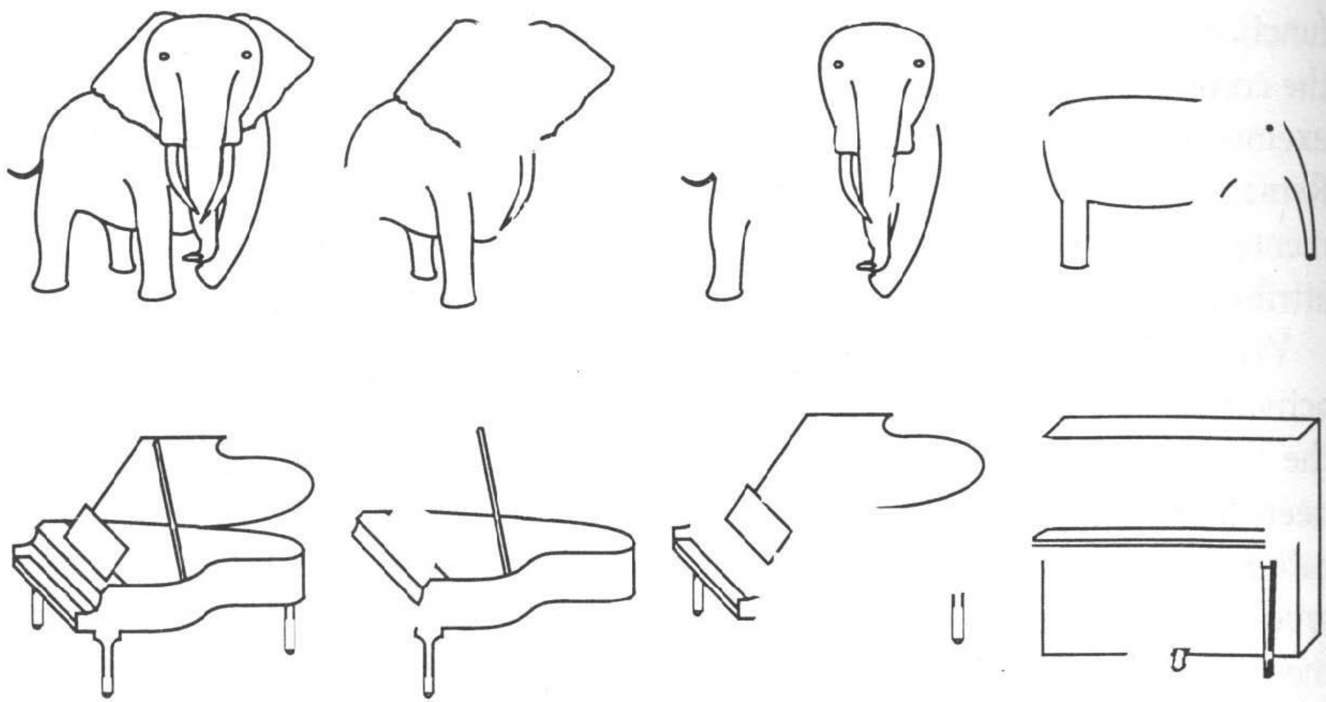
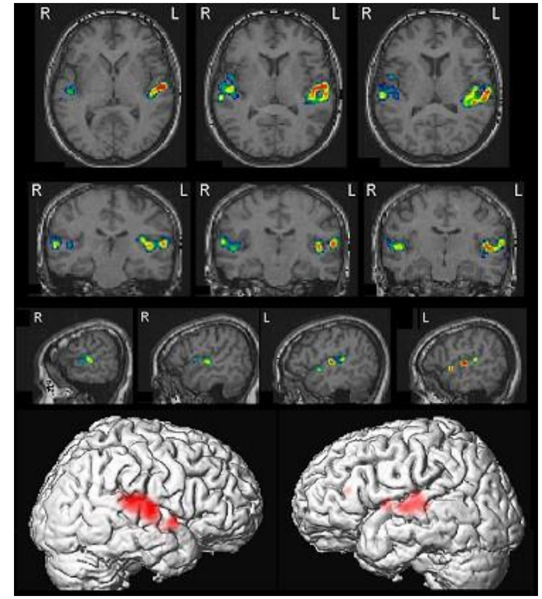
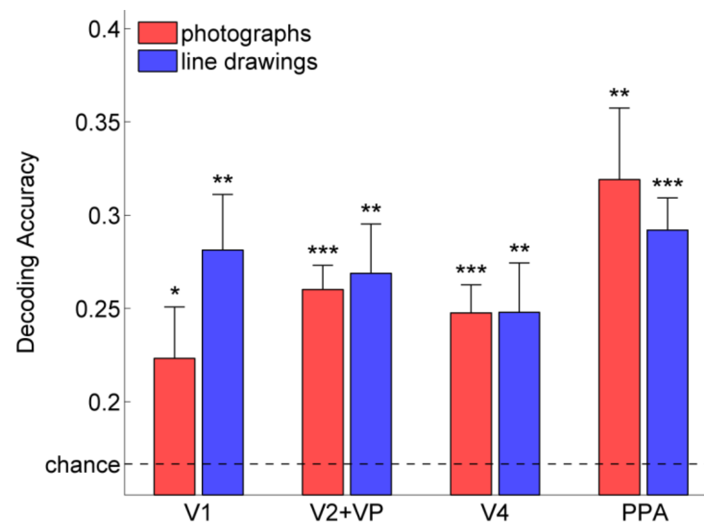


Figure 4.14
Complementary-part images. From an original intact image (left column), two complemen-





Walther, Chai, Caddigan, Beck & Fei–Fei, PNAS, 2011

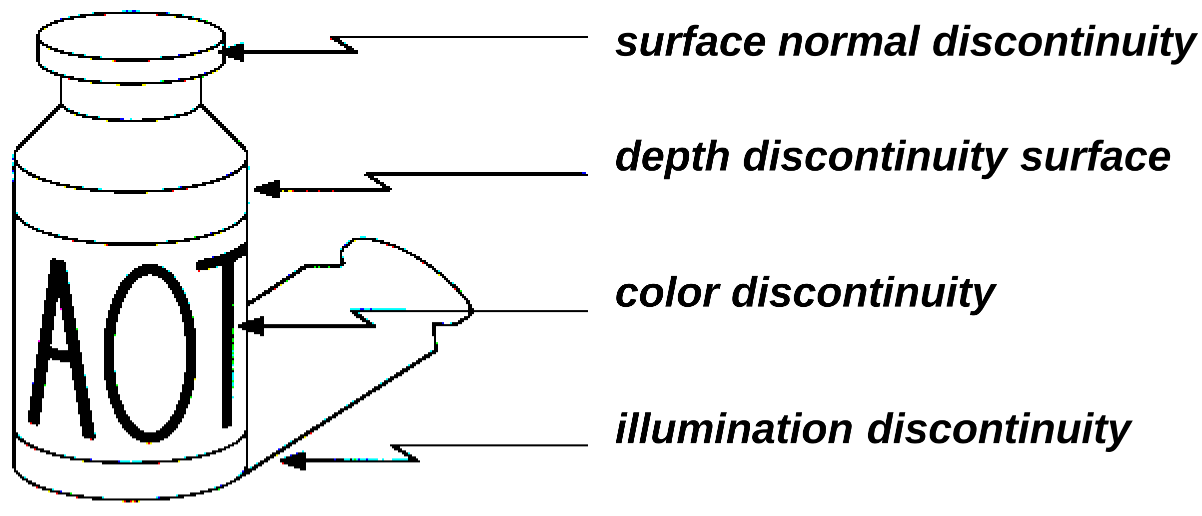
Edge detection

- ***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

Origin of Edges



- ***Edges are caused by a variety of factors***

Characterizing edges

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

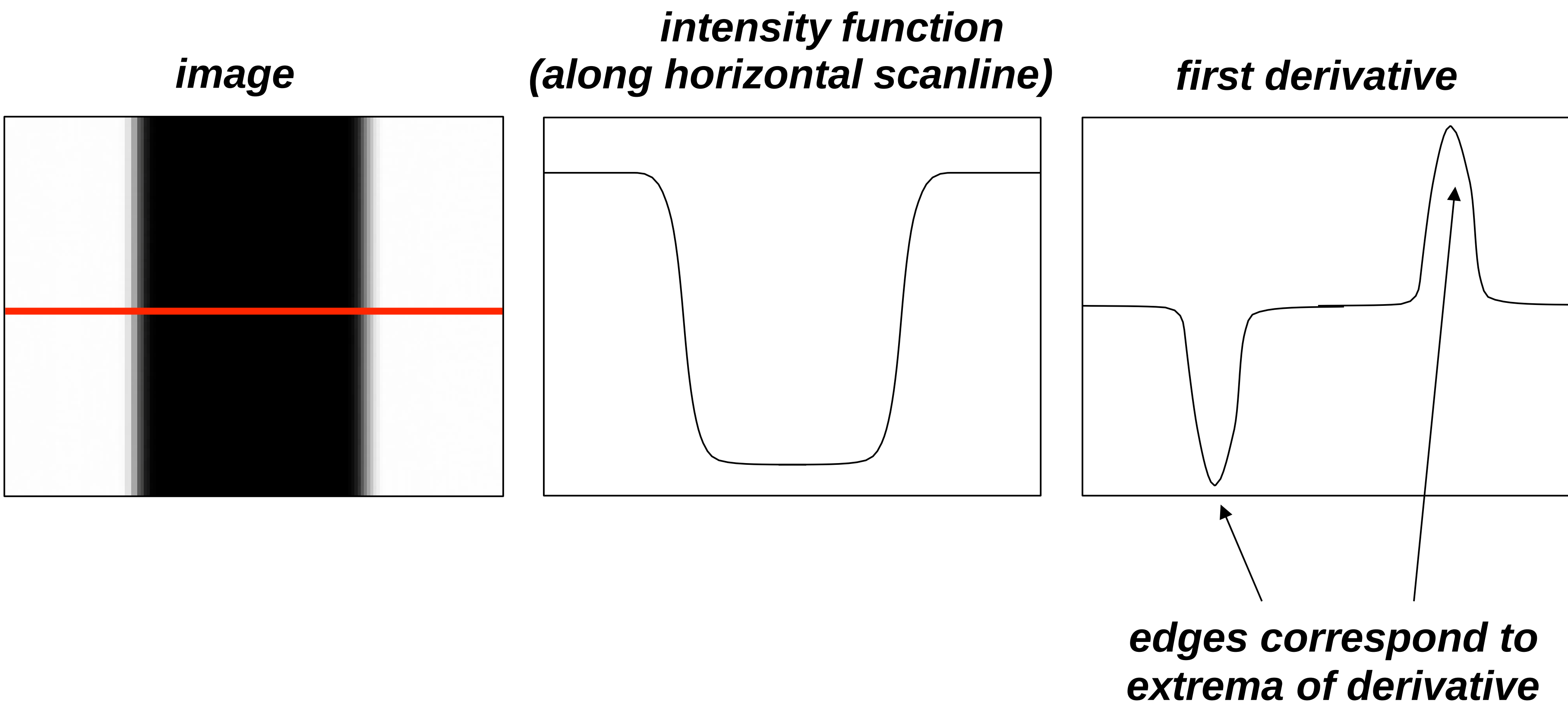
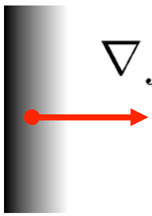
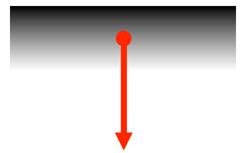


Image gradient

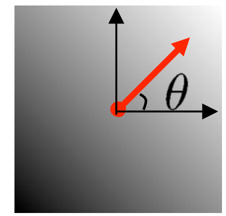
- ***The gradient of an image:*** $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



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

The gradient 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

Differentiation and convolution

- ***Recall, for 2D function, f(x,y):***

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon} \right)$$

- ***This is linear and shift invariant, so must be the result of a convolution.***

- ***We could approximate this as***

$$\frac{\partial f}{\partial x} \approx \left(\frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x} \right)$$

- ***(which is obviously a convolution)***

-1	**1**

Source: D. Forsyth, D. Lowe

Finite difference filters

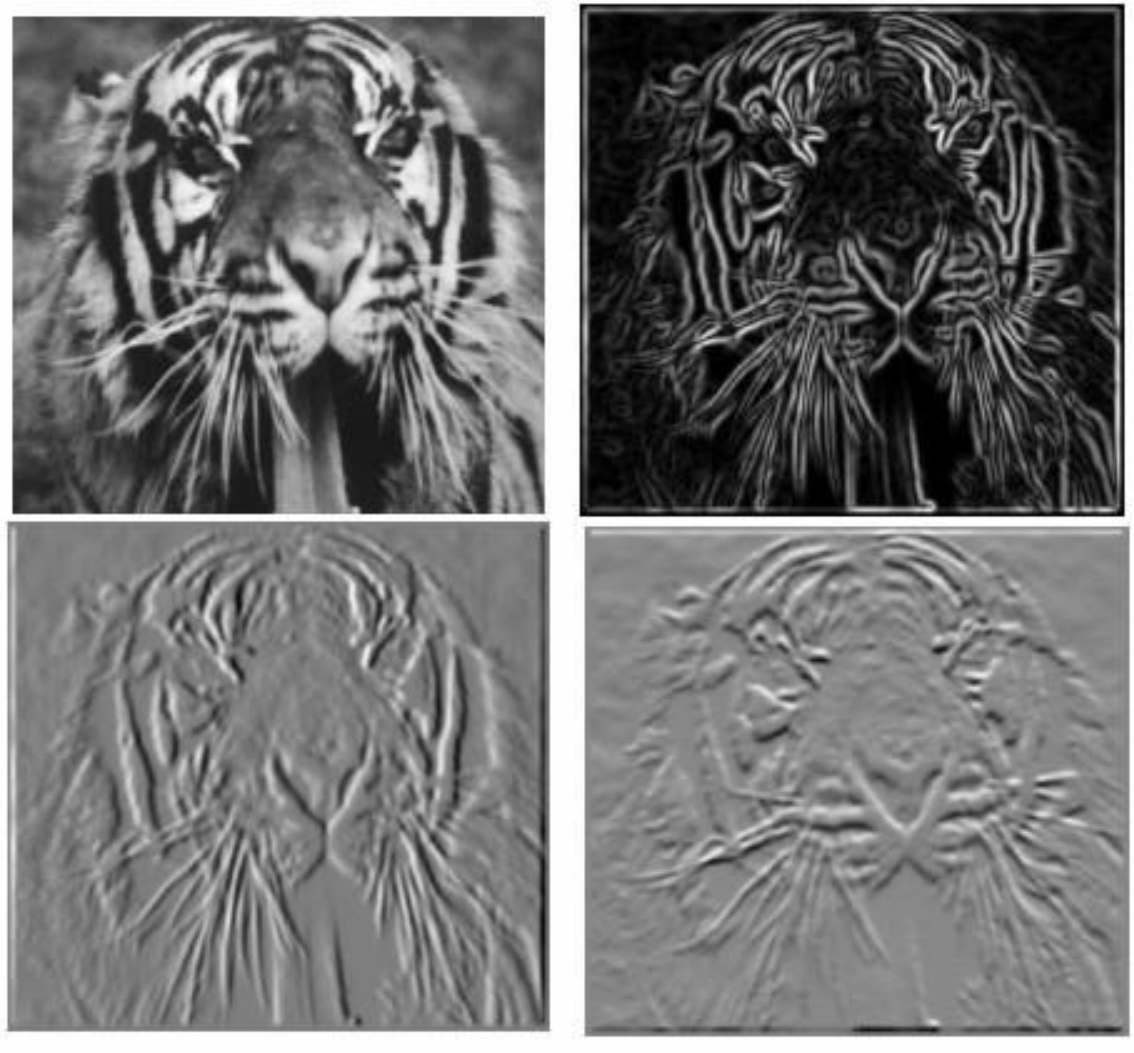
- ***Other approximations of derivative filters***

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Finite differences: example

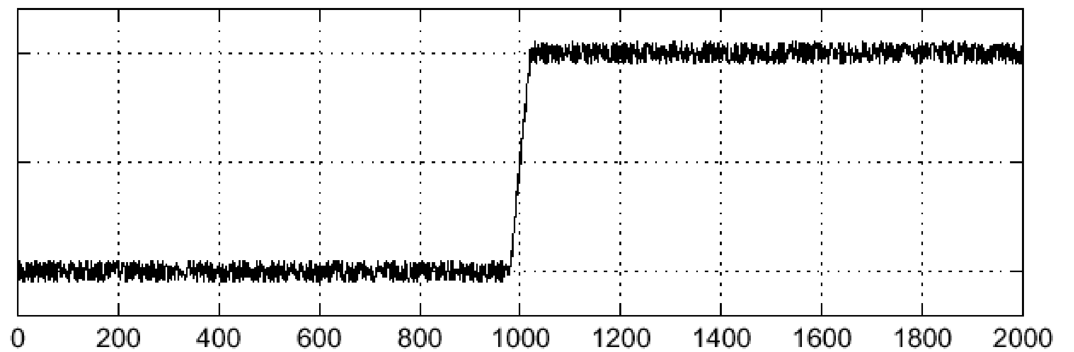


Which one is the gradient in the x-direction? How about y-direction?

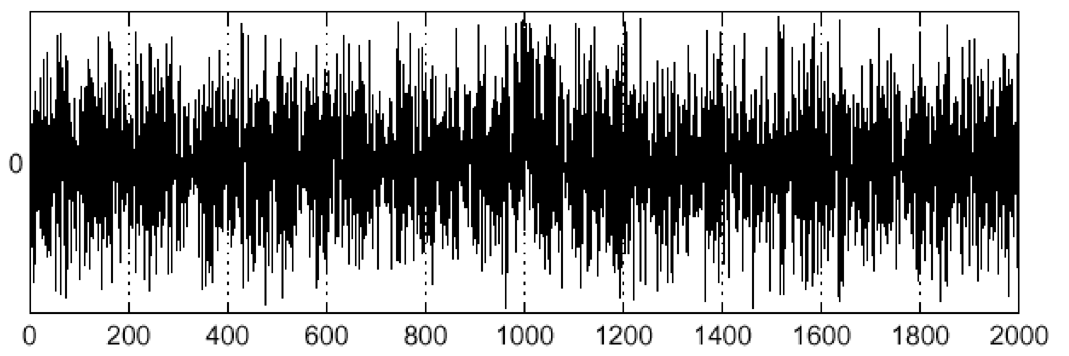
Effects of noise

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

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

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?***

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*

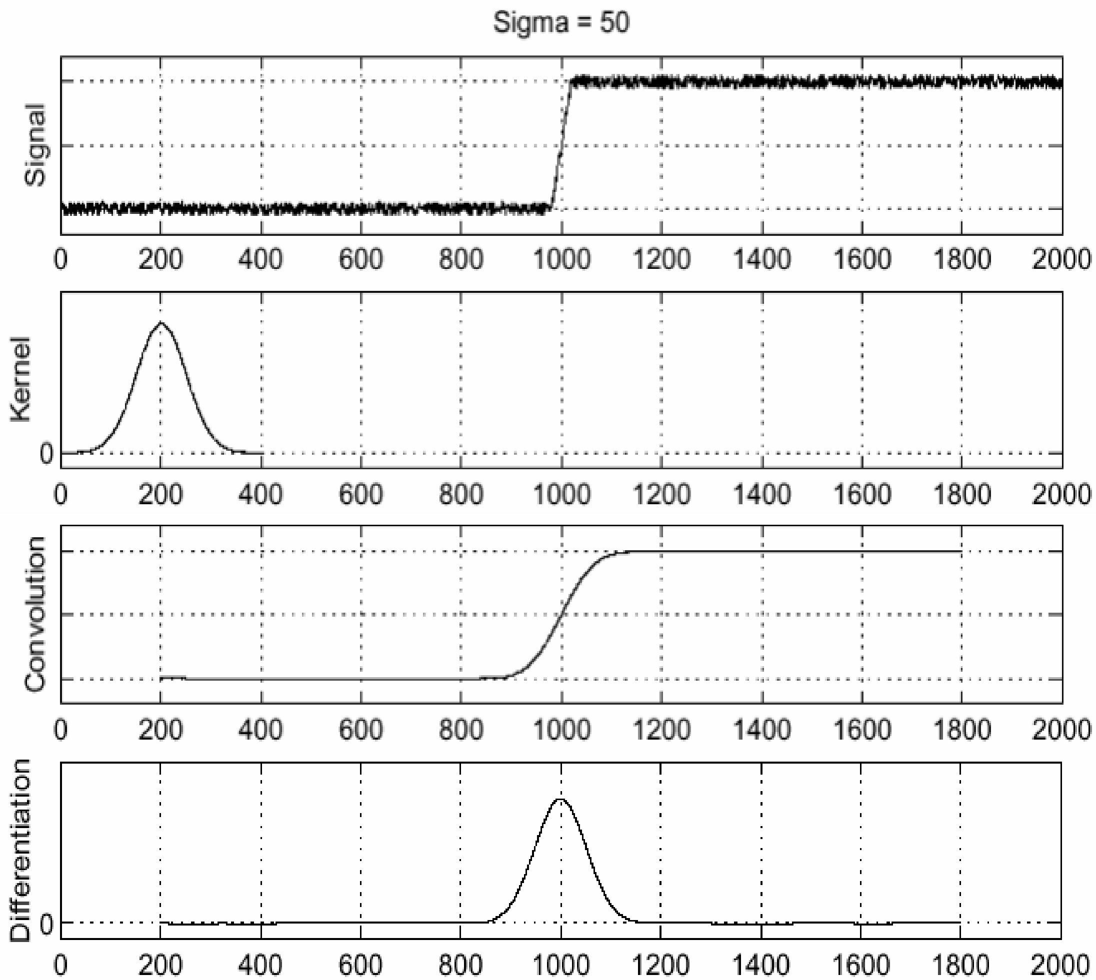
Solution: smooth first

f

g

$f * g$

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



- ***To find edges, look for peaks in*** $\frac{d}{dx}(f * g)$

Source: S. Seitz

Derivative theorem of convolution

- ***Differentiation is convolution, and convolution is associative:***

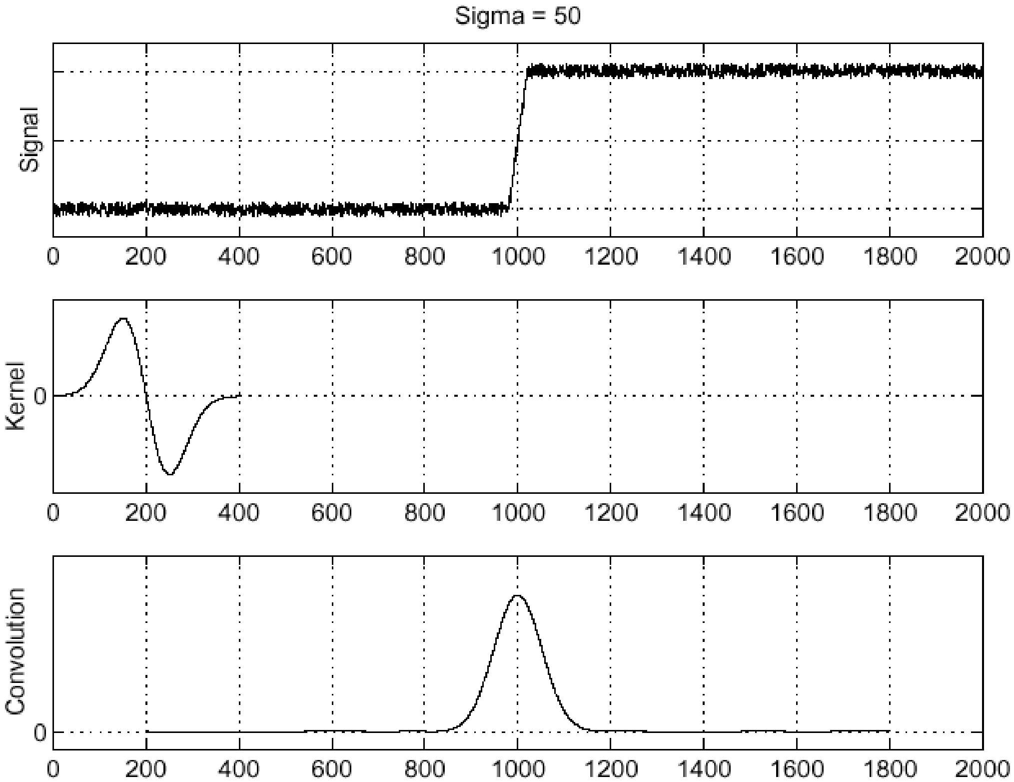
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- ***This saves us one operation:***

f

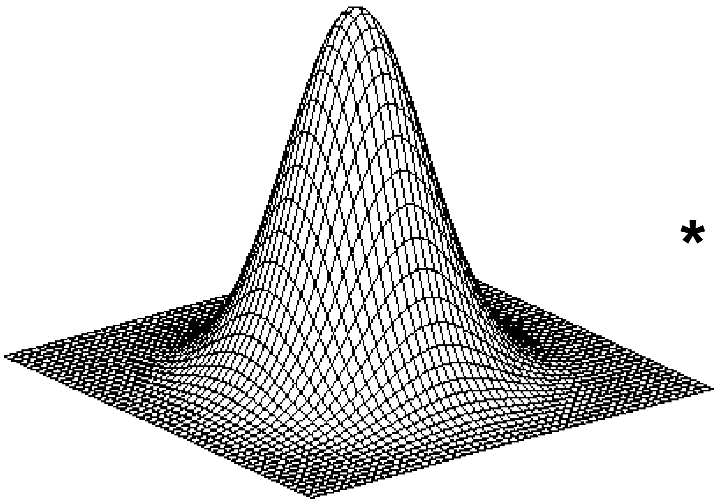
g

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

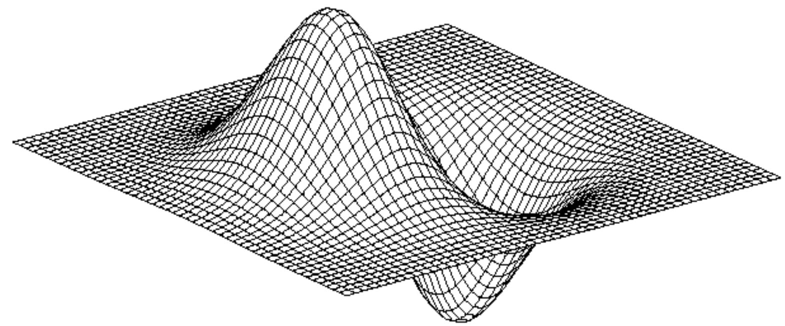


Source: S. Seitz

Derivative of Gaussian filter

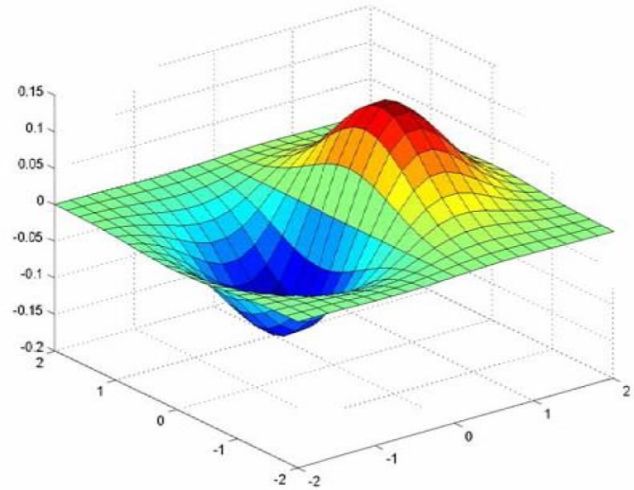


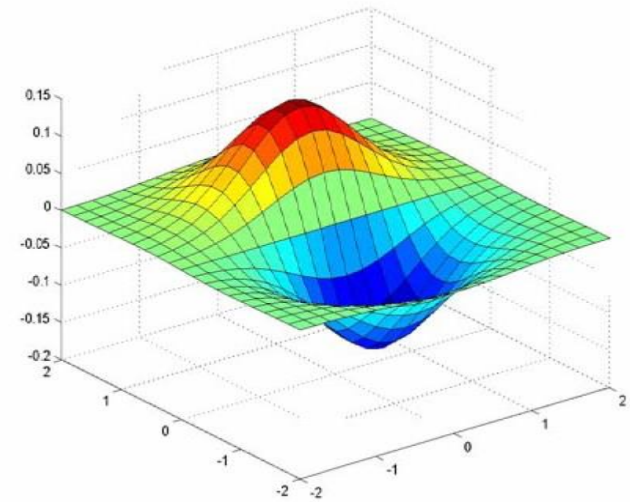
** [1 -1] =*



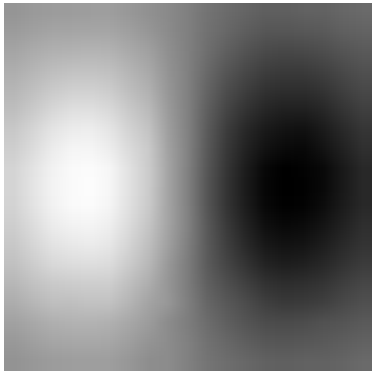
- ***Is this filter separable?***

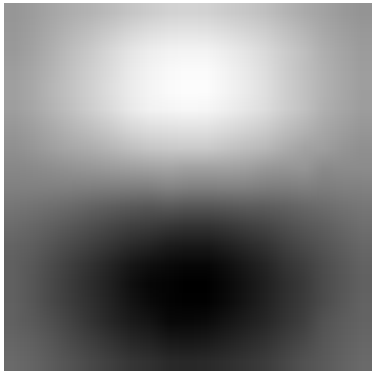
Derivative of Gaussian filter





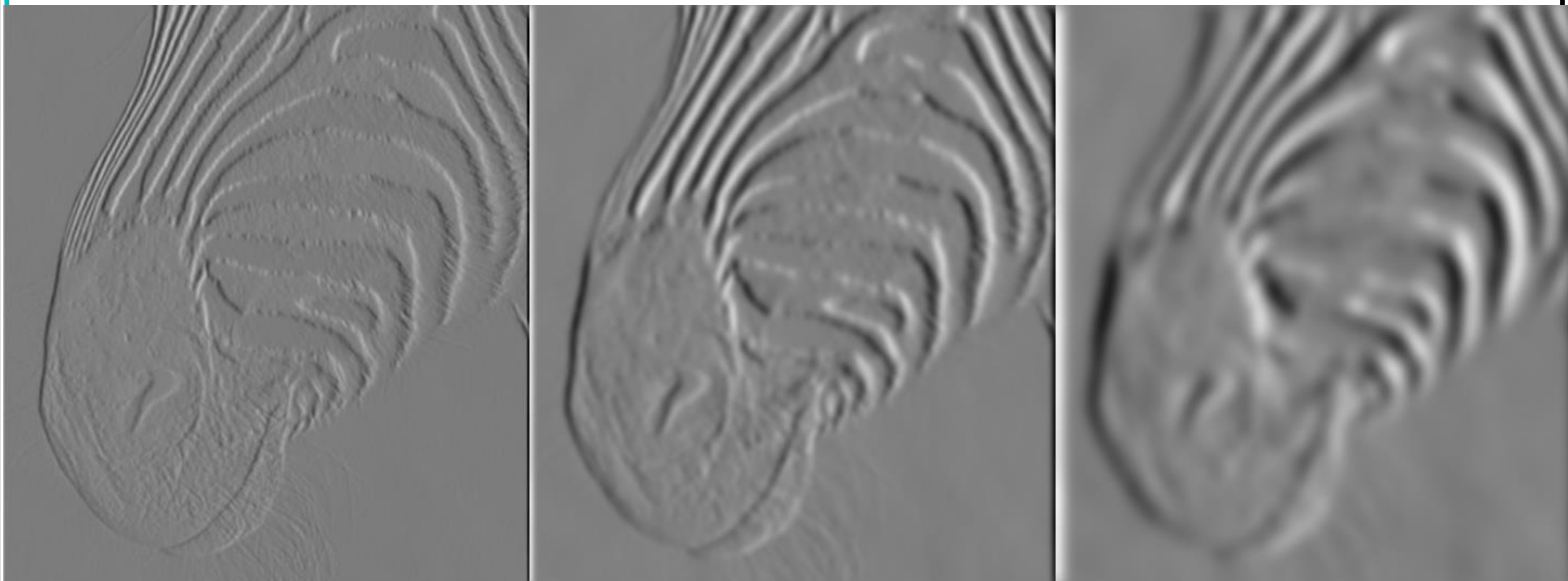
x-direction ***y-direction***





- ***Which one finds horizontal/vertical edges?***

Tradeoff between smoothing and localization



1 pixel ***3 pixels*** ***7 pixels***

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

Implementation issues

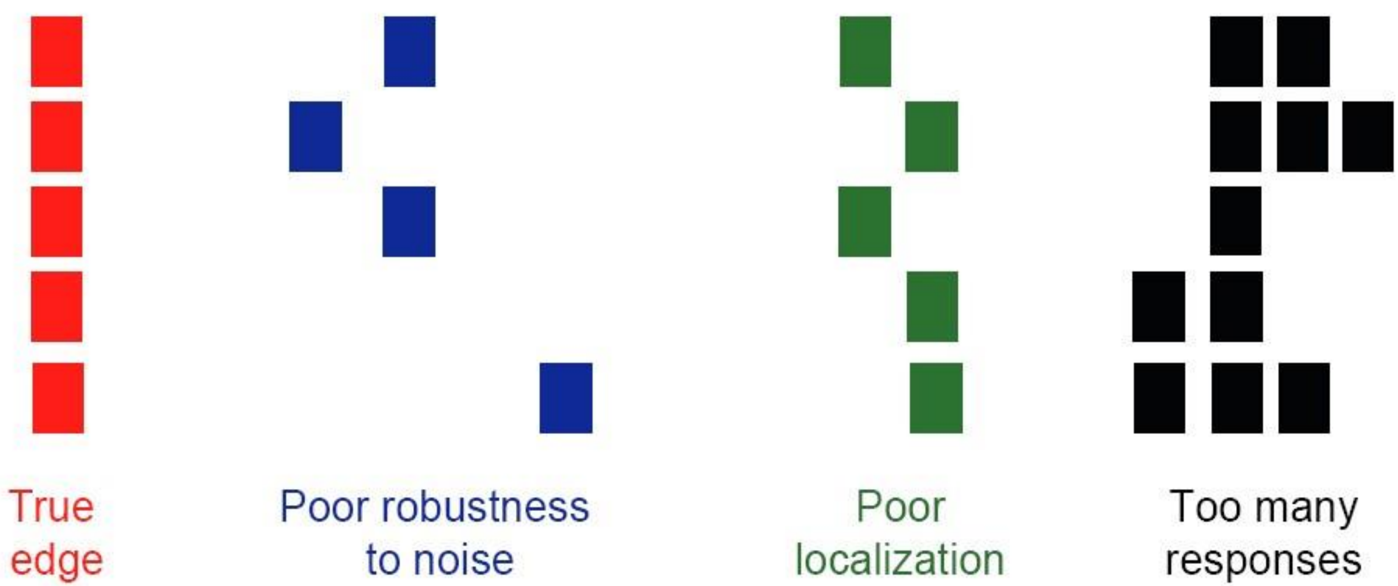
- ***The gradient magnitude is large along a thick "trail" or "ridge," so how do we identify the actual edge points?***
- ***How do we link the edge points to form curves?***

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*

Designing an edge detector

- ***Criteria for an "optimal" edge detector:***
  - ***Good detection***
  - ***Good localization***
  - ***Single response***



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. Papern Analysis and Machine Intelligence, 8:679-714, 1986.

Canny edge detector

1. *Filter image with derivative of Gaussian*
2. *Find magnitude and orientation of gradient*
3. *Non-maximum suppression:*
   – *Thin multi-pixel wide "ridges" down to single pixel width*
4. *Linking and thresholding (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')*
- *EmguCV: Image<Gray, Byte> cannyEdges = Binary_Image.Canny(cannyThreshold, cannyThresholdLinking);*

Example

- ***Original image (Lena)***

Example

Norm of the gradient

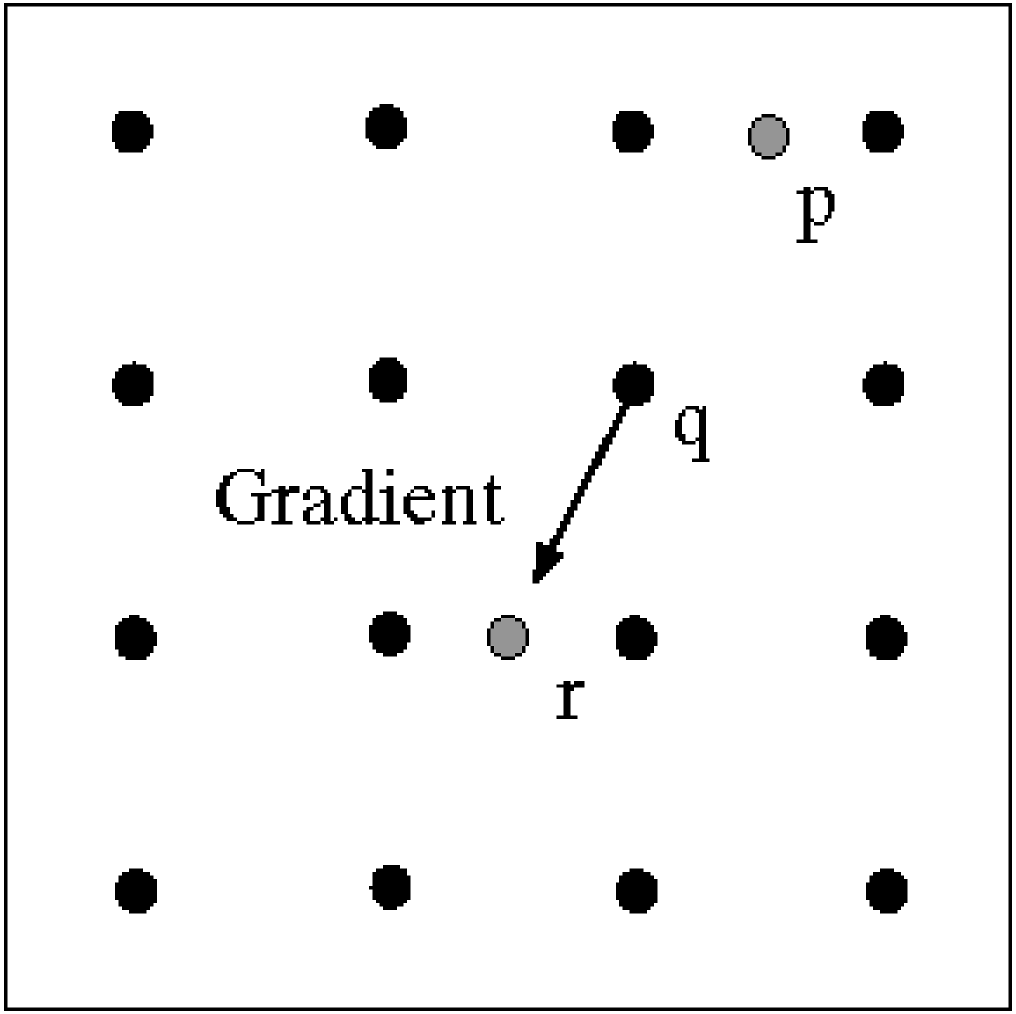
Example

Thresholding

Example

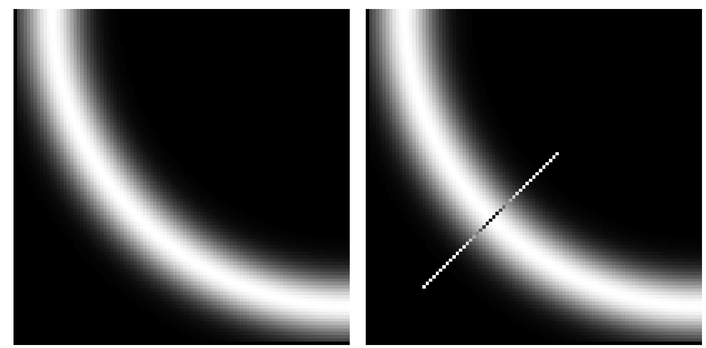
***Thinning** (non-maximum suppression)*

Non-Maximum Suppression



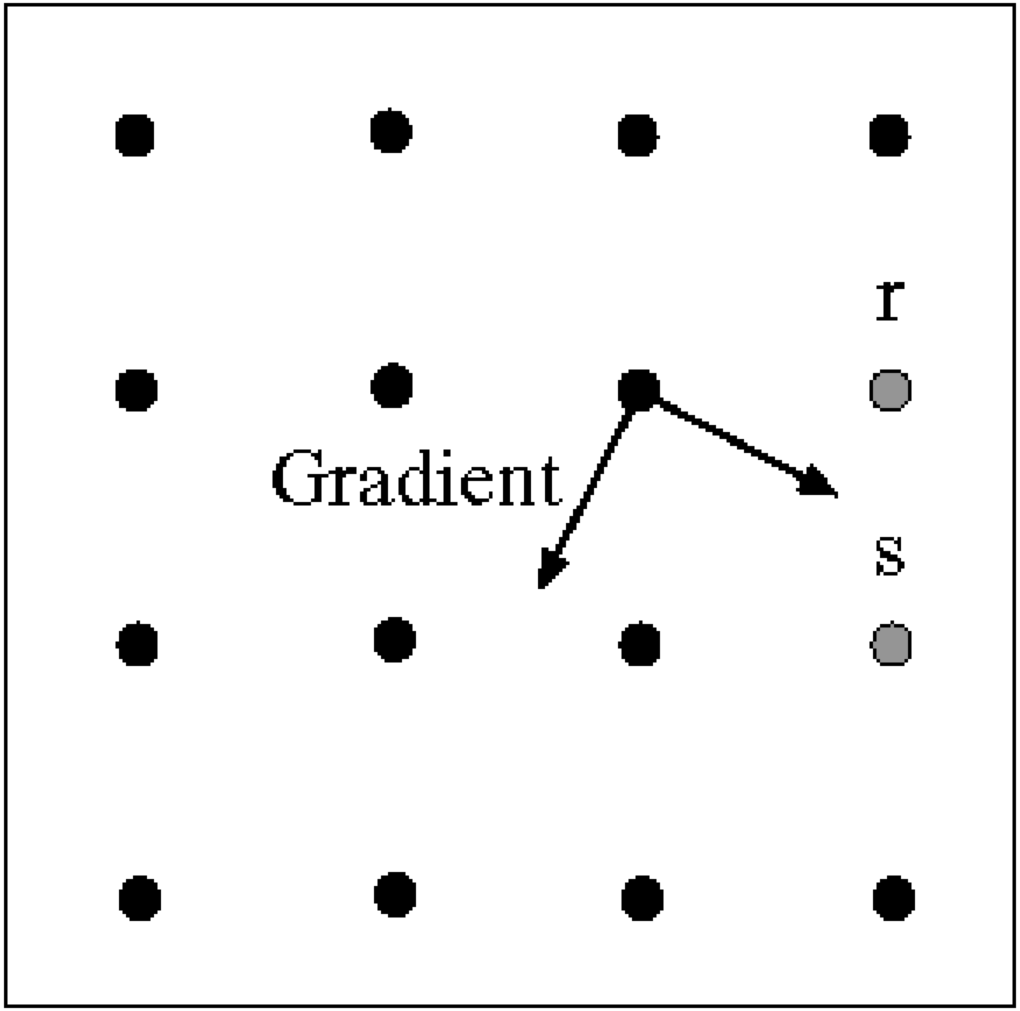
At q, we have a maximum if the value is larger than those at both p and r.

Interpolate to get these values.

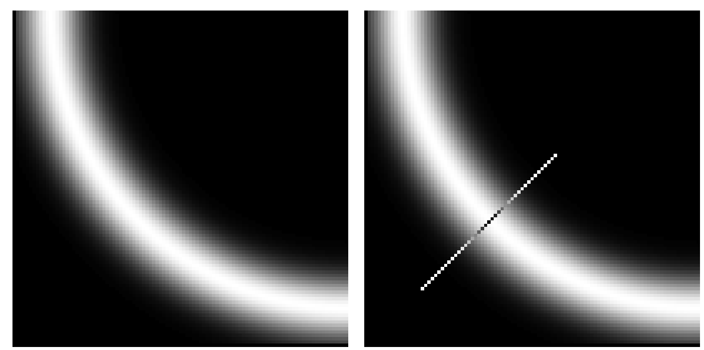


Source: D. Forsyth

Edge linking

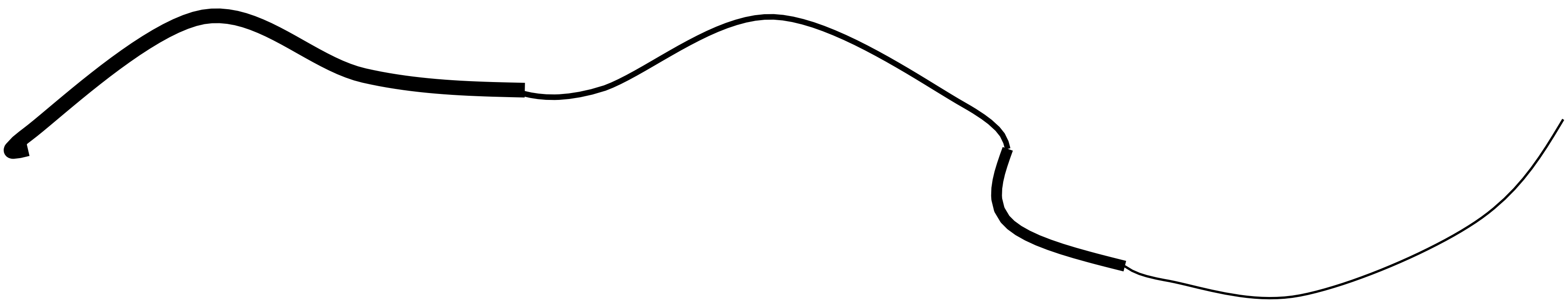


Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



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.***



Hysteresis thresholding

original

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Effect of σ (Gaussian kernel spread/size)

original

Canny with $\sigma = 1$

Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- ***Large σ detects large scale edges***
- ***Small σ detects fine features***

Edge detection is just the beginning…

image ***human segmentation*** ***gradient magnitude***

- ***Berkeley segmentation database:***
  [*hp p ://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/*]()