



COMPUTER VISION

Le Thanh Ha, Ph.D

Assoc. Prof. at University of Engineering and Technology,
Vietnam National University

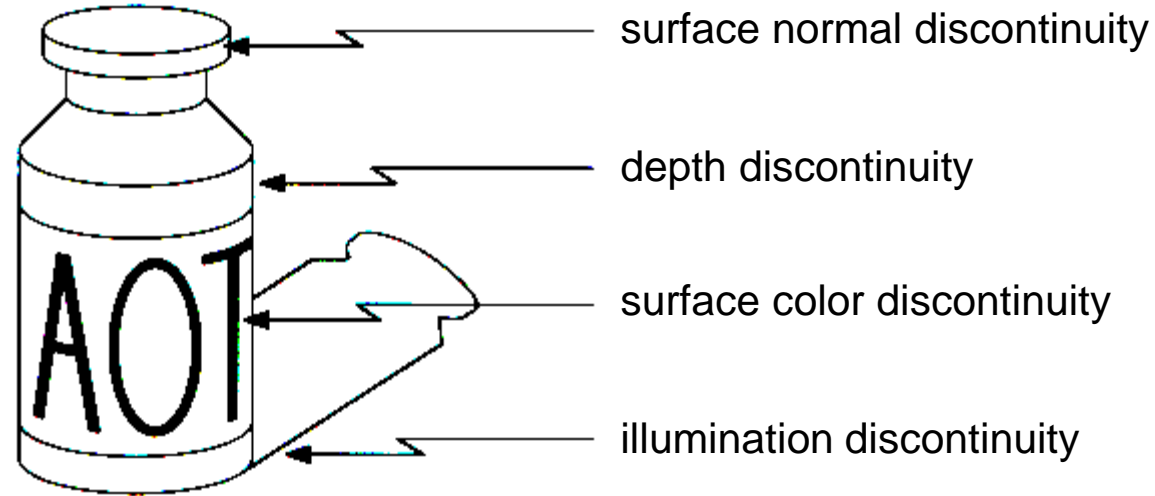
ltha@vnu.edu.vn; lthavnu@gmail.com; 0983 692 592

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)



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

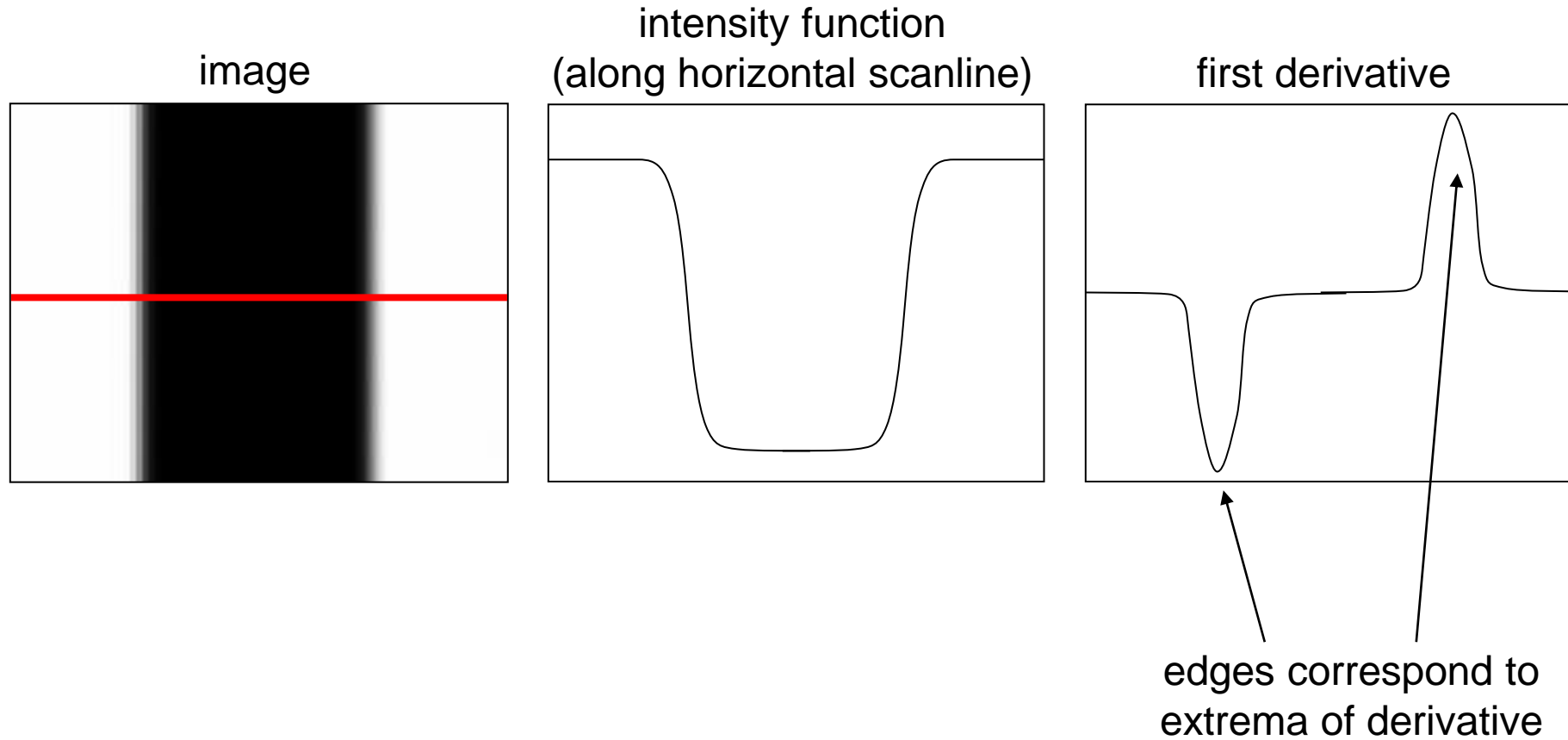
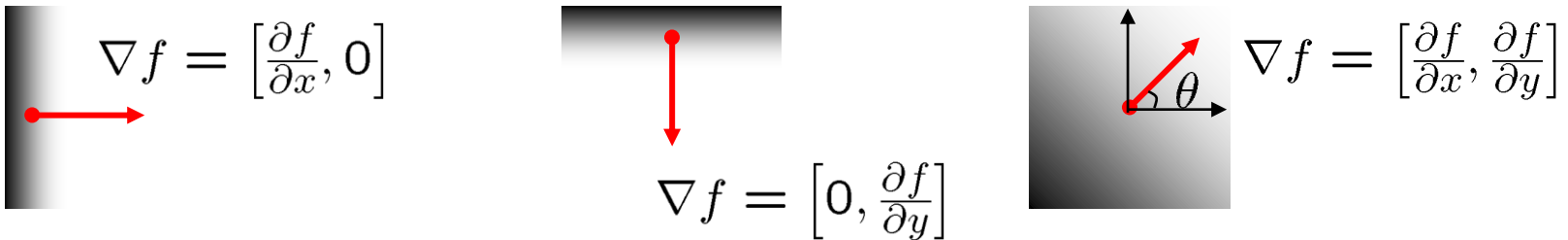


Image gradient

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

- 

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

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

Differentiation and convolution

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

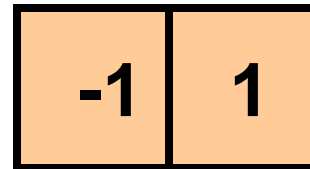
$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \rightarrow 0} \left(\frac{f(x + \varepsilon, y)}{\varepsilon} - \frac{f(x, y)}{\varepsilon} \right)$$

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

- We could approximate this as

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

- (which is obviously a convolution)



Finite difference filters

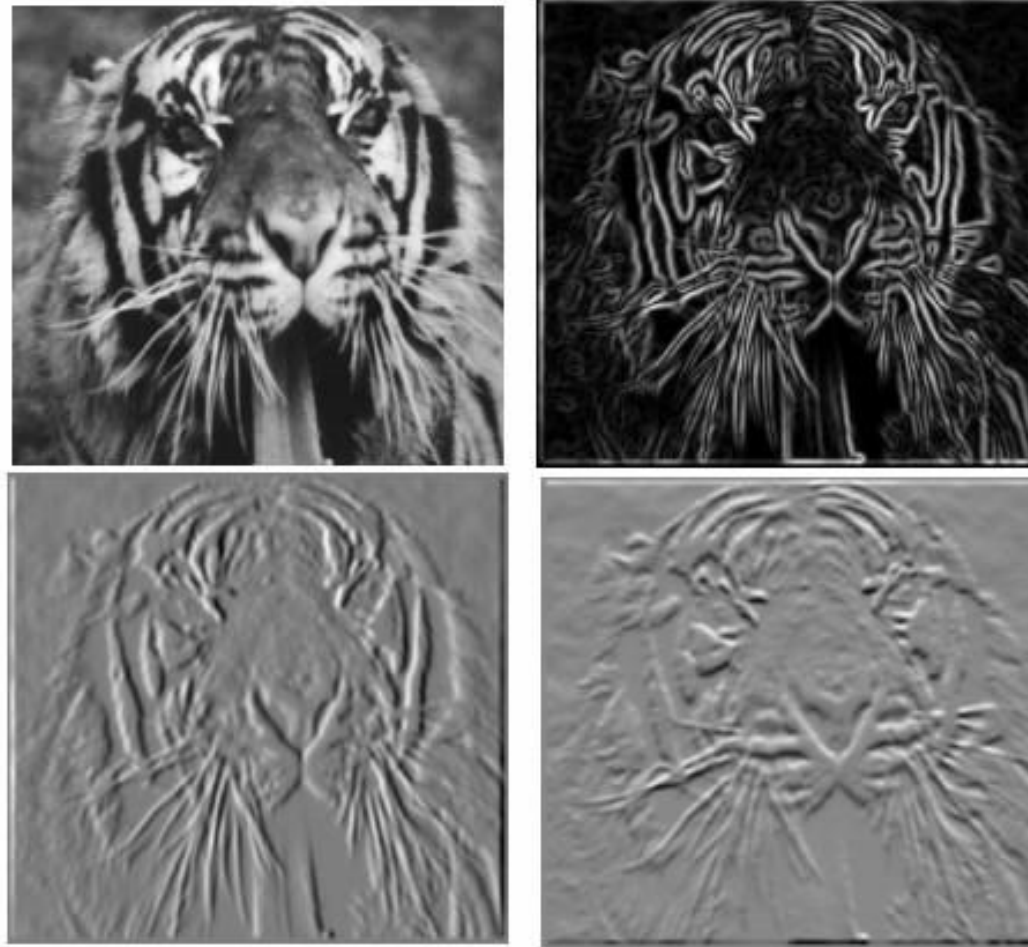
- Other approximations of derivative filters exist:

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

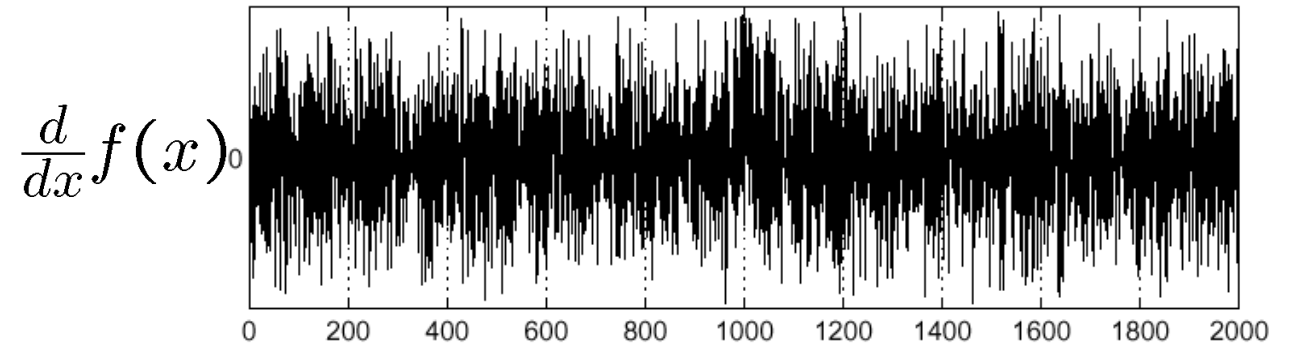
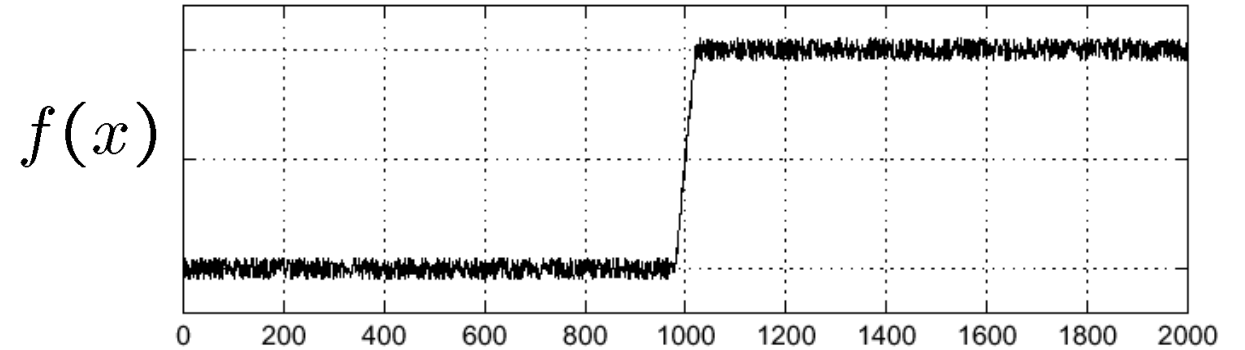
Finite differences: example



- Which one is the gradient in the x-direction (resp. y-direction)?

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?

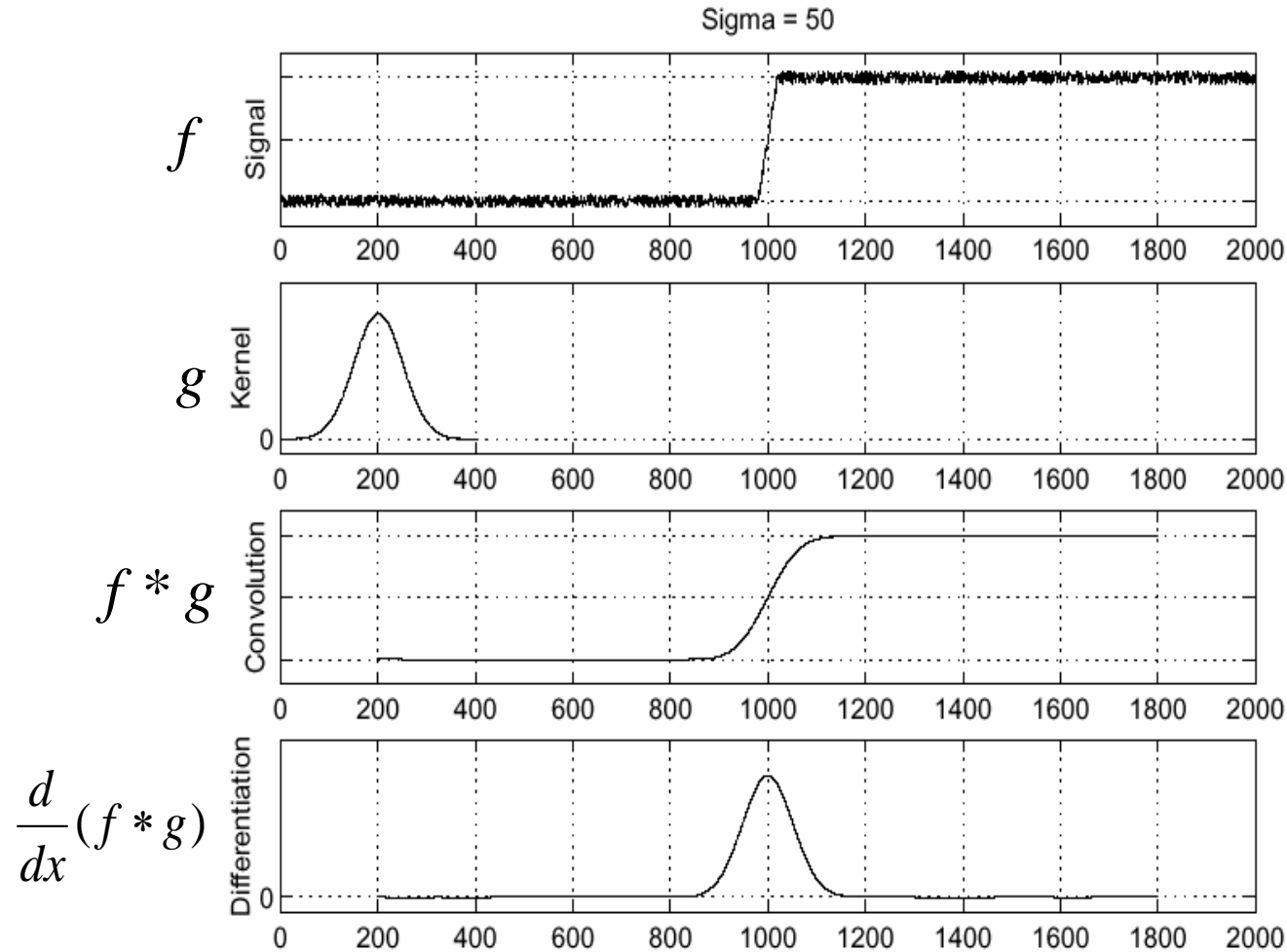
Effects of noise

- 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

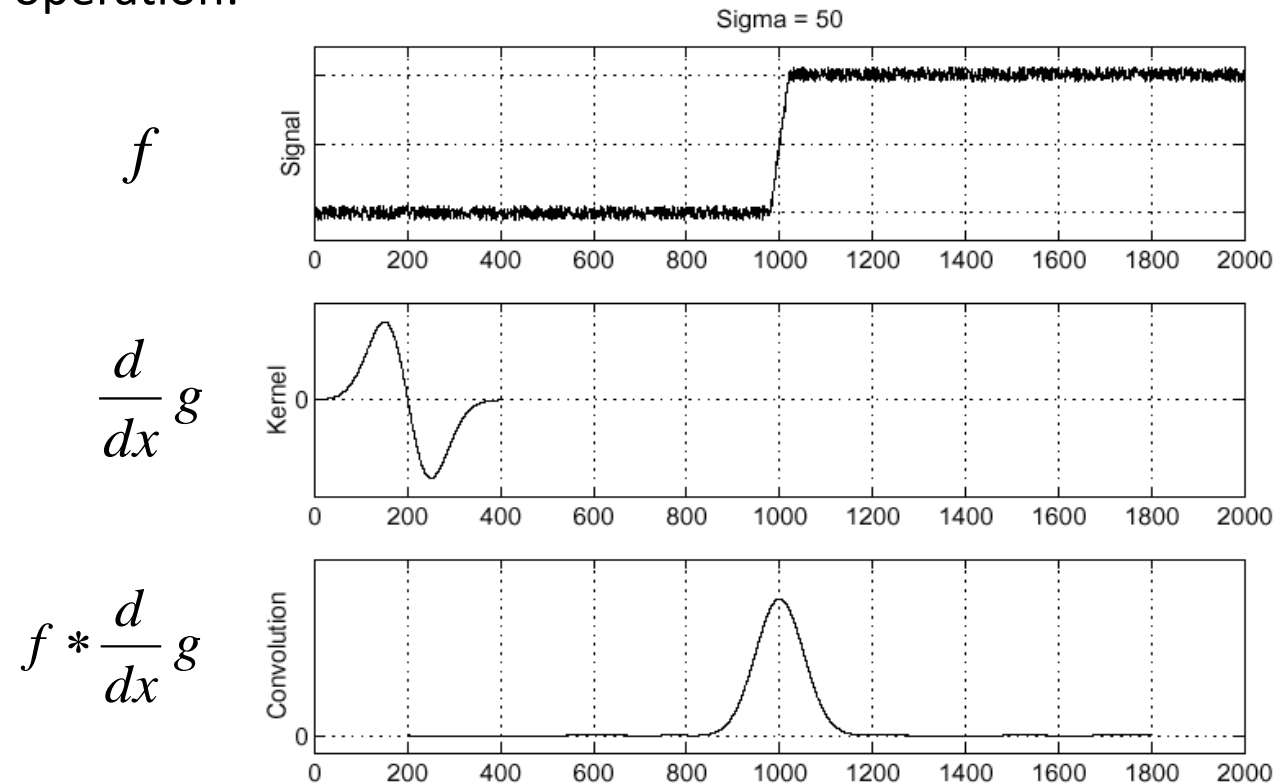
Solution: smooth first



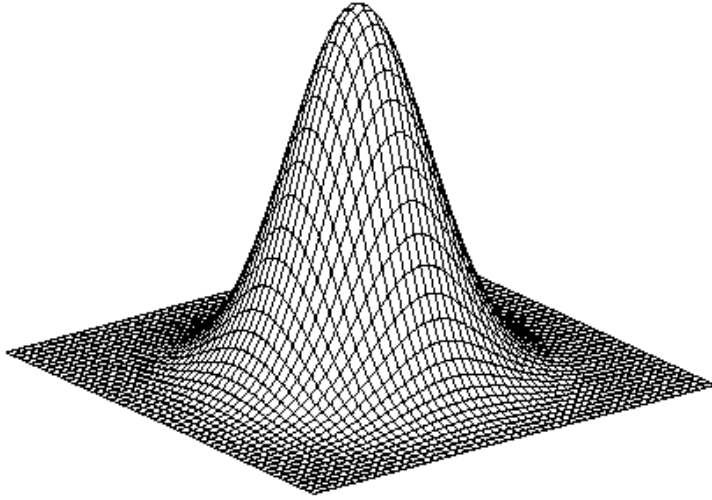
- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Derivative theorem of convolution

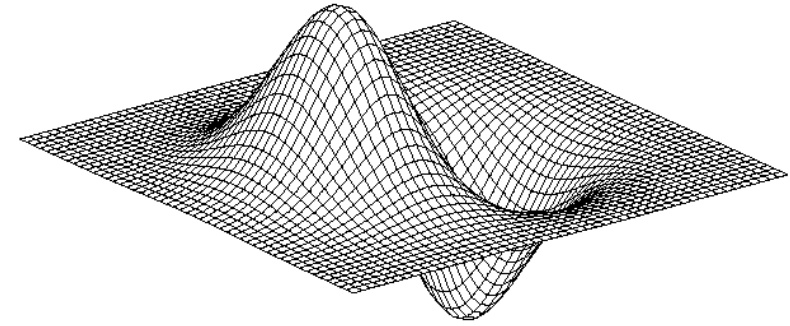
- Differentiation is convolution, and convolution is associative: $\frac{d}{dx}(f * g) = f * \frac{d}{dx}g$
- This saves us one operation:



Derivative of Gaussian filter

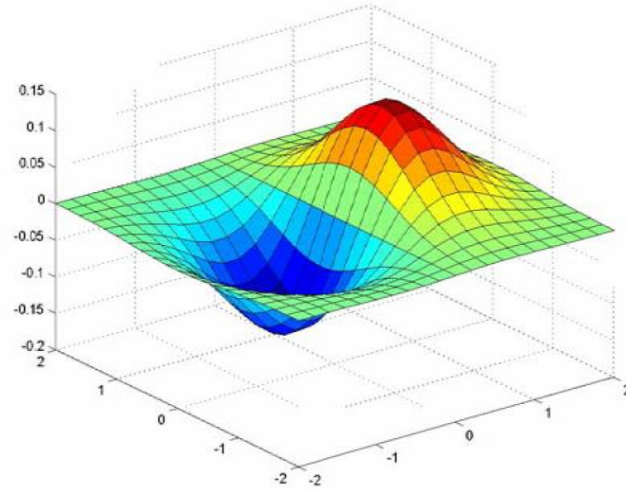


$$* \begin{bmatrix} 1 & -1 \end{bmatrix} =$$

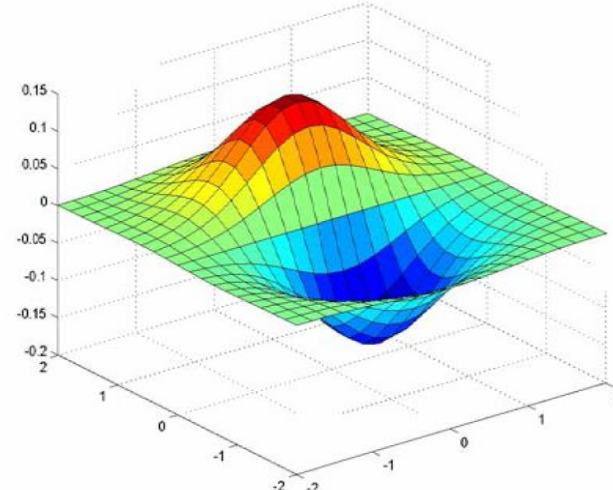
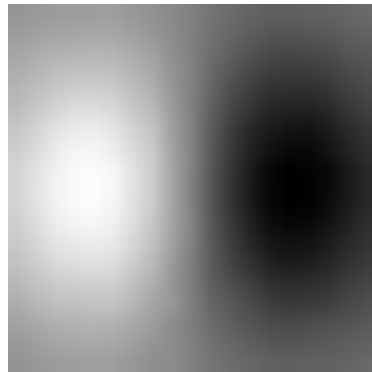


- Is this filter separable?

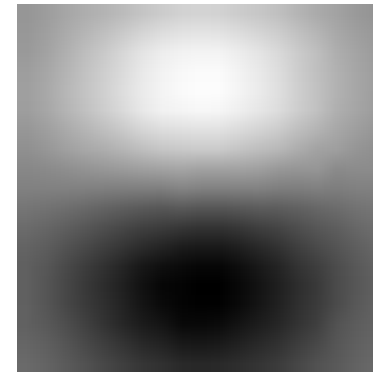
Derivative of Gaussian filter



x-direction

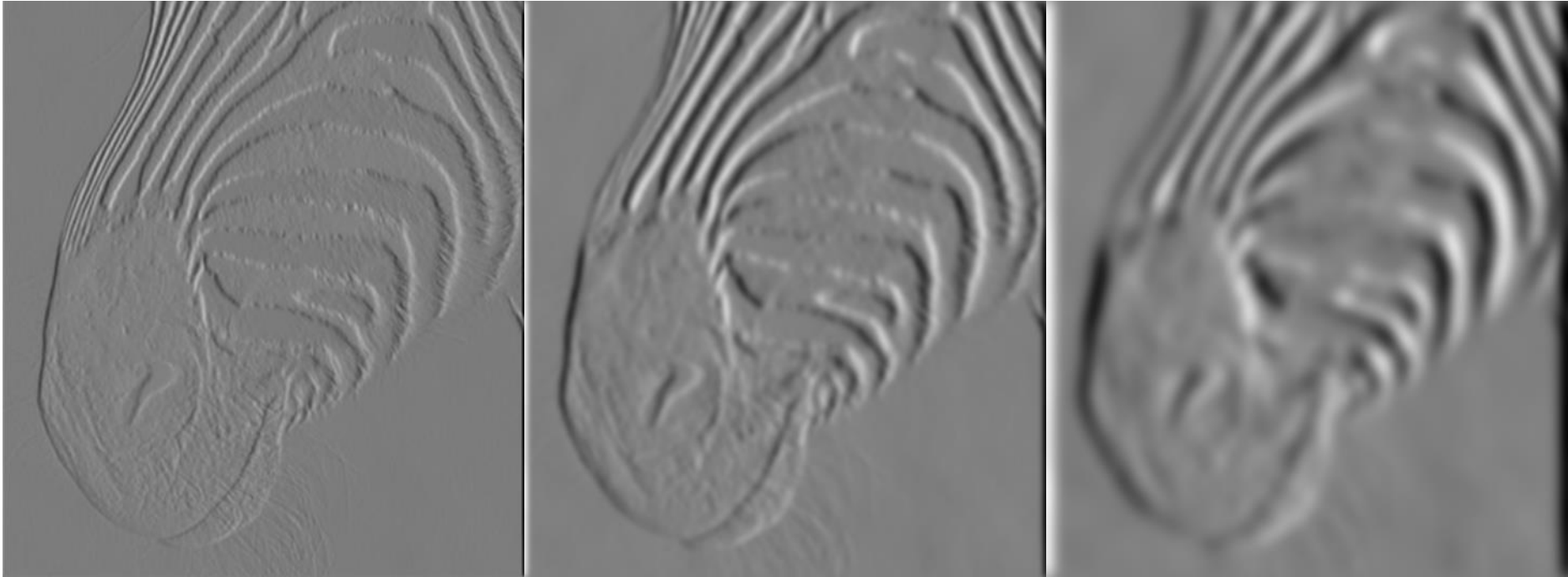


y-direction



- 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?

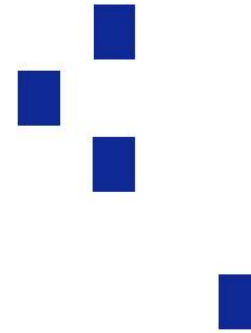


Designing an edge detector

- 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



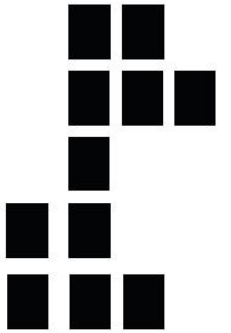
True
edge



Poor robustness
to noise



Poor
localization



Too many
responses

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.

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')`

Example



- original image (Lena)

Example



norm of the gradient

Example



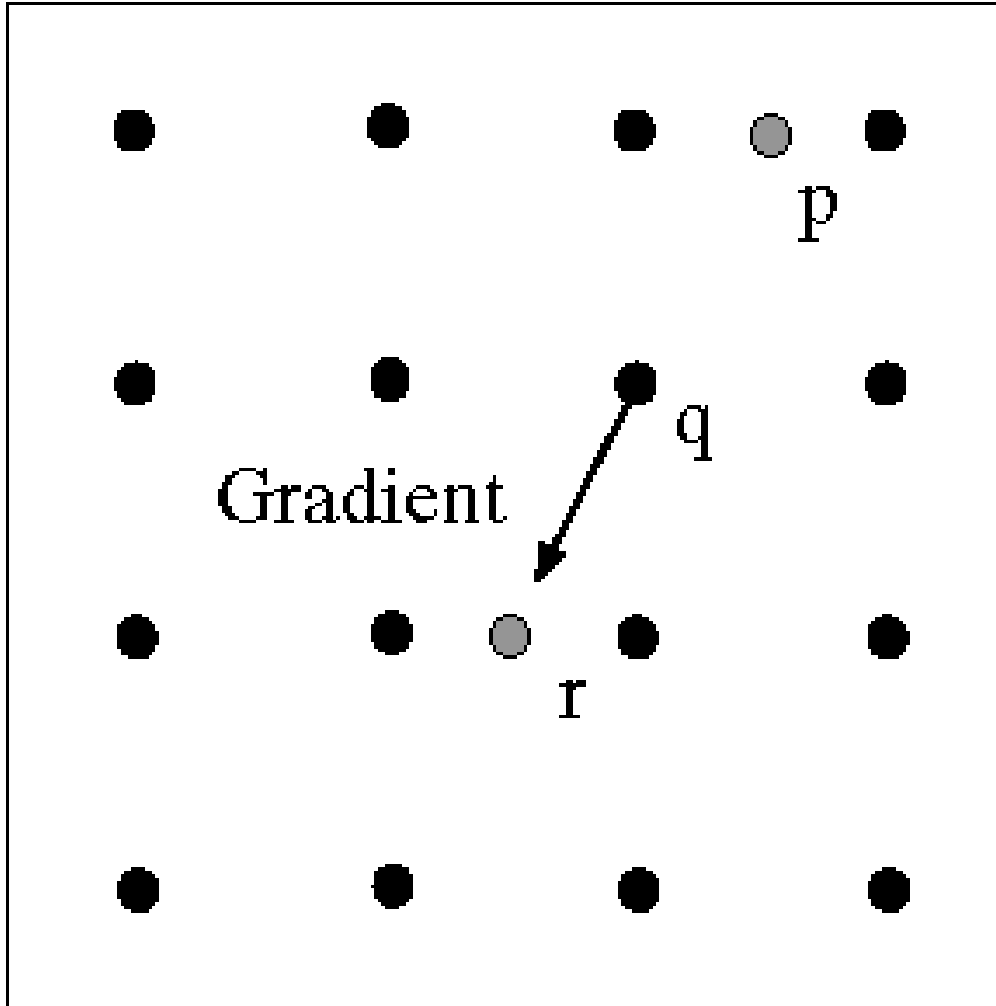
thresholding

Example

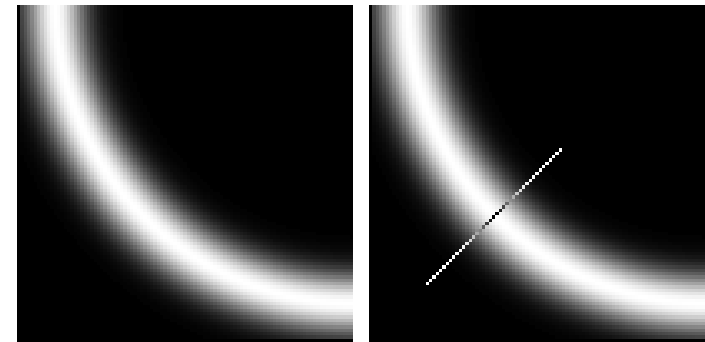


thinning
(non-maximum suppression)

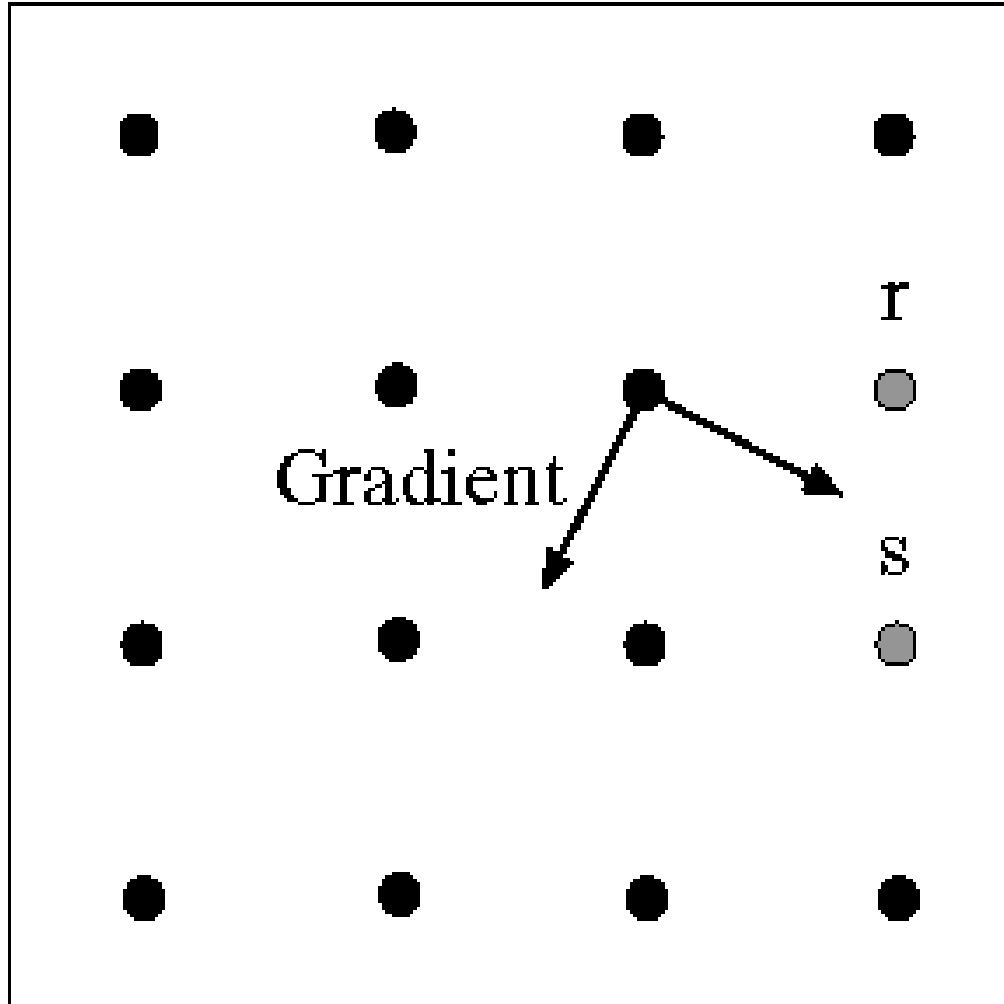
Non-maximum suppression



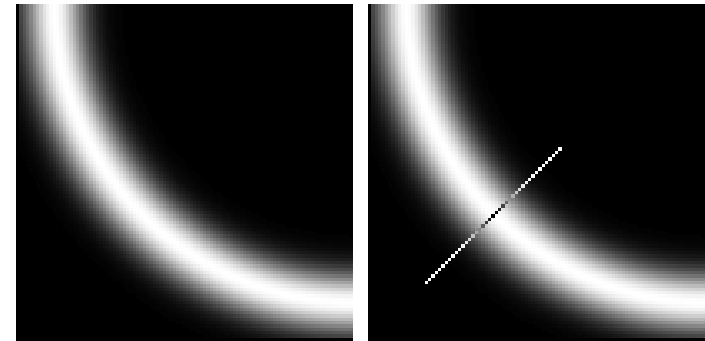
At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Edge linking

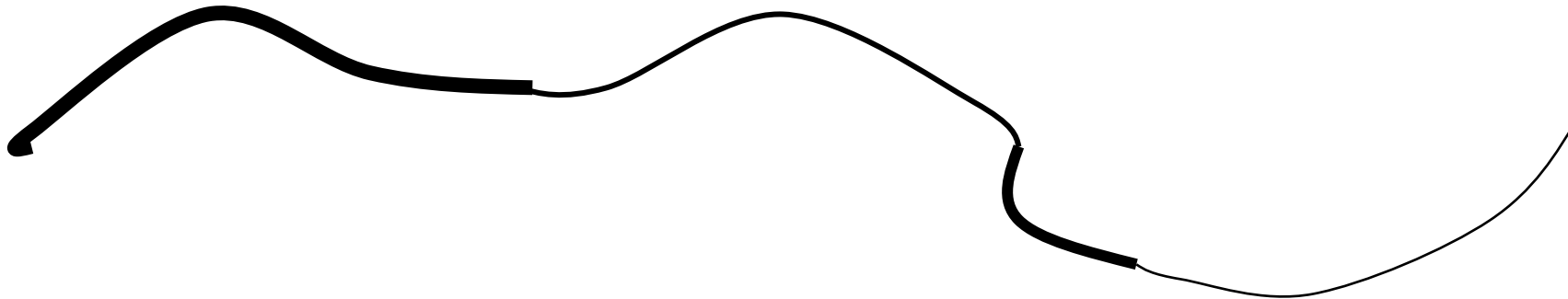


Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



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 image



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

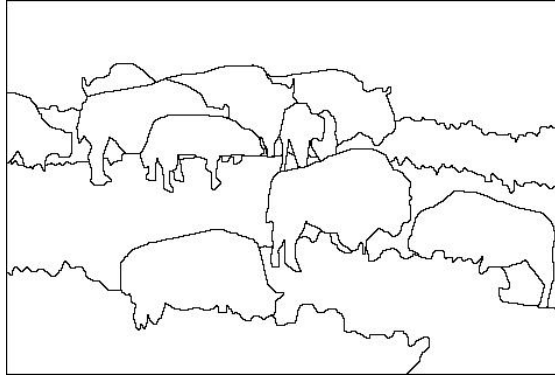
- large σ detects large scale edges
- small σ detects fine features

Edge detection is just the beginning...

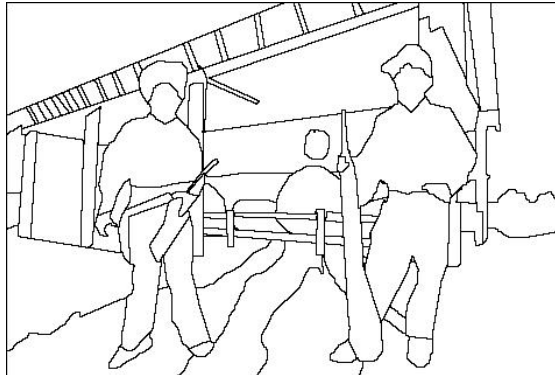
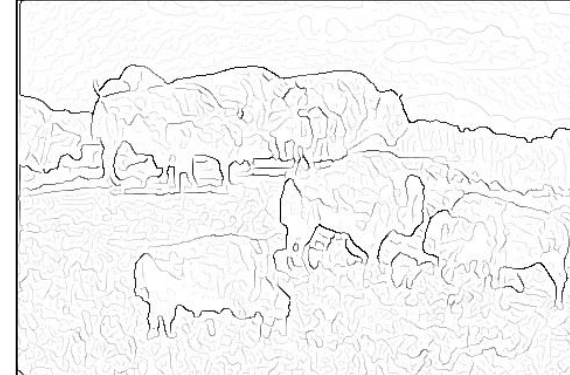
image



human segmentation



gradient magnitude



- Berkeley segmentation database:

<http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/>