

# Edges and Features

The content of most slides are from Minh Haoi Nguyen (SBU)

# A biref anlaogy wtih txet

- Waht mttares is waht hpapens on wrod baounrdies
- Mocpera iwht htsi sesm

# Primal Sketch

- Early vision: invariants, moments, pattern recognition
- David Marr, late'70s
  - Inspiration from biological vision
  - Image representation in terms of `sketch'
- Sketch components
  - Edges
  - Ridges
  - Blobs
  - Junctions

# Feature Extraction

- Features: local meaningful detectable
  - Points
  - Edges
    - Step edges
    - Line edges
  - Contours
    - Closed contours are boundaries
  - Regions

# Goals for low-level image representation

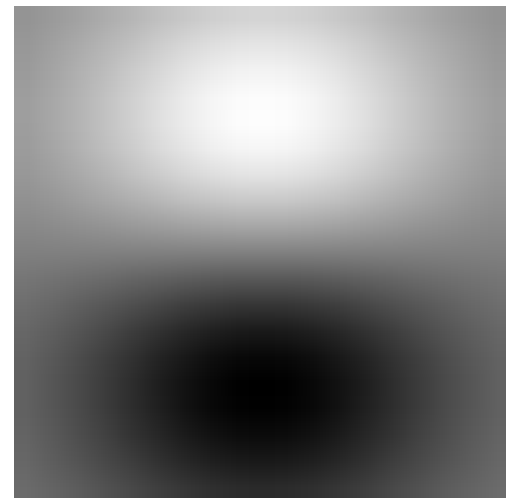
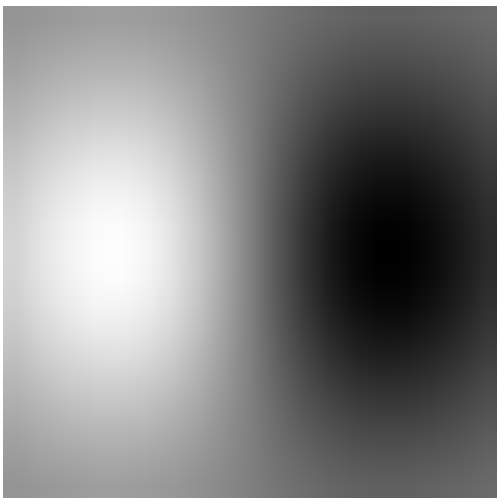
- Compact
- Generic: same for all tasks, objects, images considered
- Sufficient: no need to look back into the image

# Purpose

- Extract compact, generic, representation of image that carries sufficient information for higher-level processing tasks
- Essentially what area V1 does in our visual cortex.

# Filters are templates

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector
- Filtering the image is a set of dot products
- Insight
  - filters look like the effects they are intended to find
  - filters find effects they look like



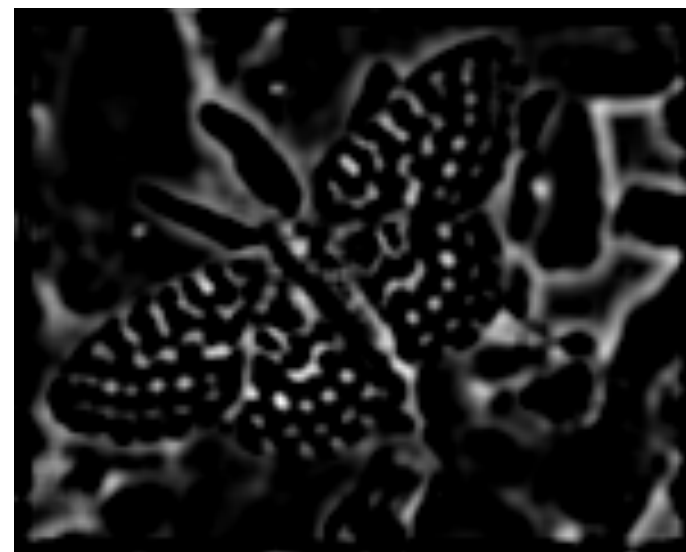
# Normalized correlation

- Think of filters of a dot product
  - now measure the angle
  - i.e normalized correlation output is filter output, divided by root sum of squares of values over which filter lies
- Tricks:
  - ensure that filter has a zero response to a constant region (helps reduce response to irrelevant background)
  - subtract image average when computing the normalizing constant (i.e. subtract the image mean in the neighborhood)
  - absolute value deals with contrast reversal





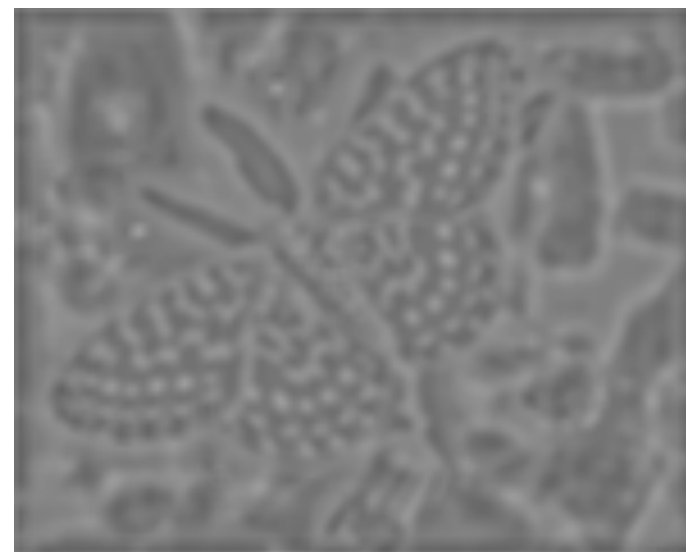
Input image and filter



Positive responses



Zero mean image, -1:1 scale



Zero mean image, -max:max scale



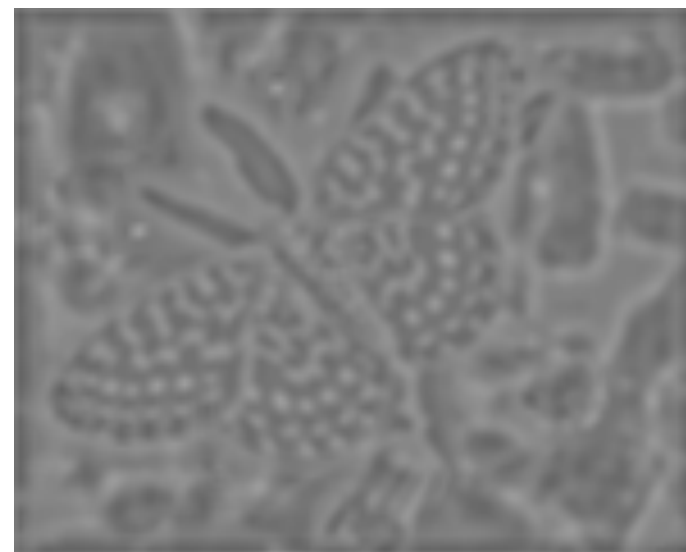
Input image and filter



Positive responses



Zero mean image, -1:1 scale



Zero mean image, -max:max scale

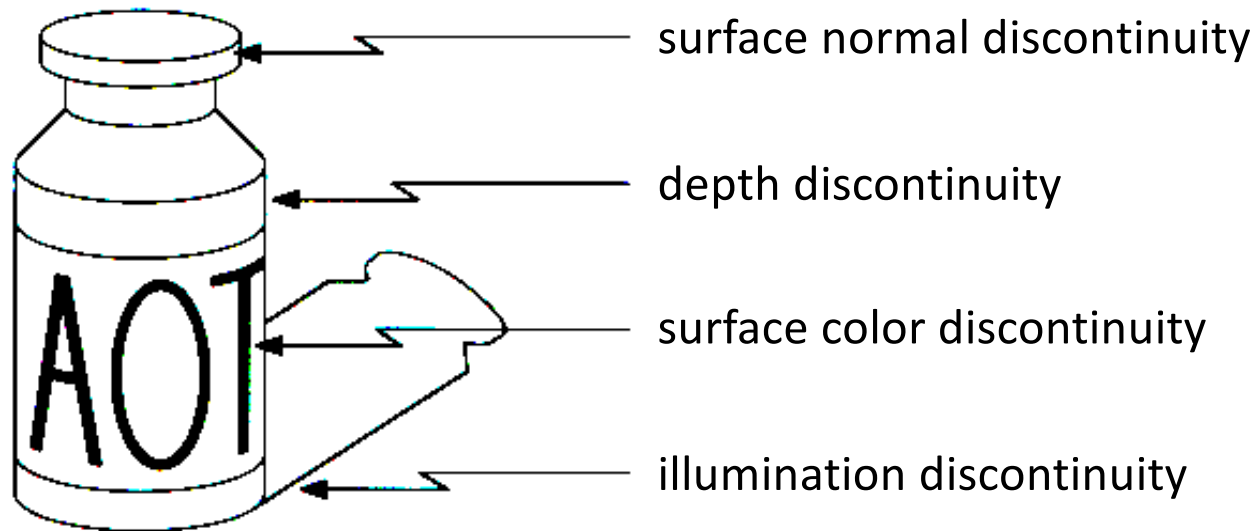
# Edge detection

- 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)



# Edges and Gradients

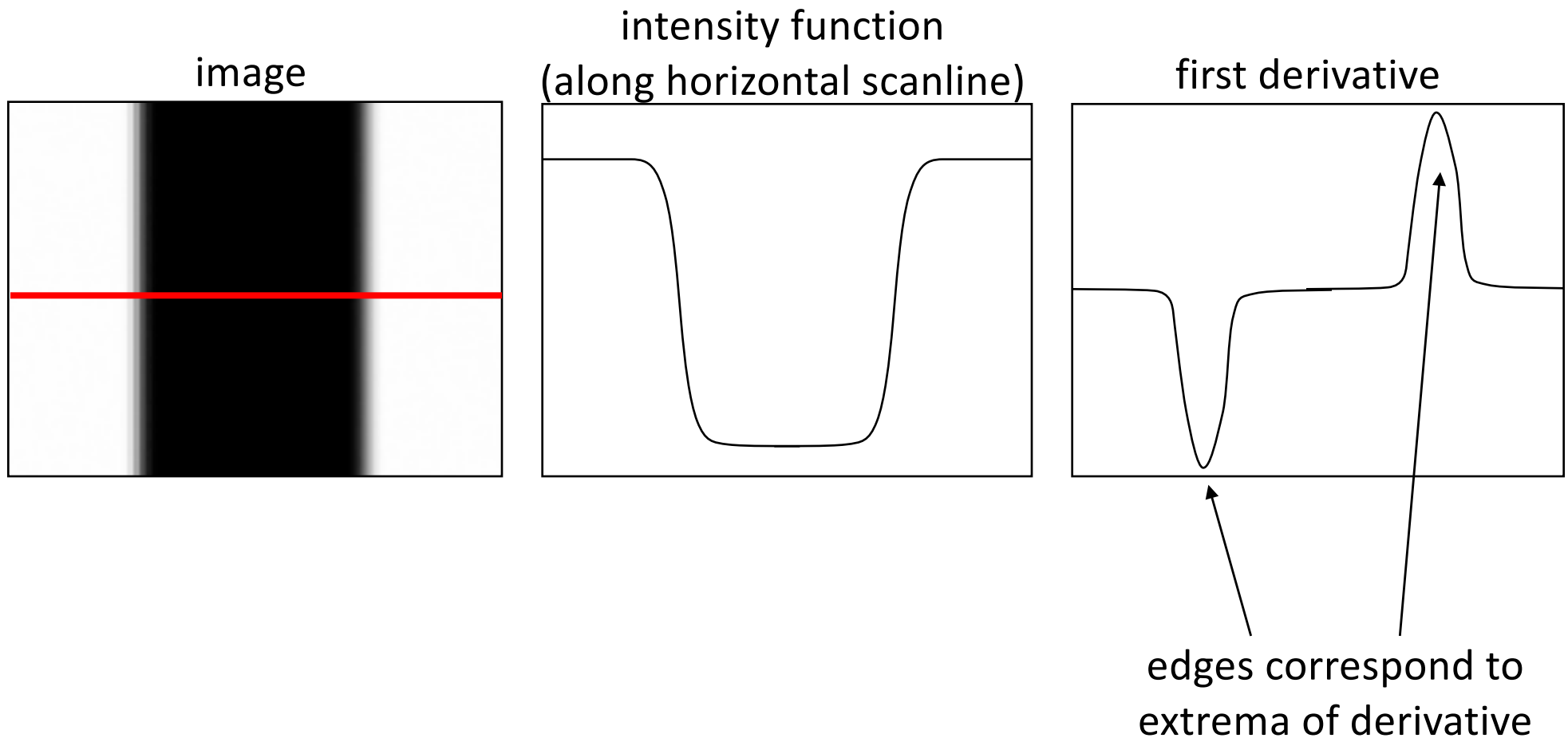
- Edges are caused by a variety of factors



- General strategy for edge detection
  - determine image gradient
  - now mark points where gradient magnitude is particularly large wrt neighbors (ideally, curves of such points).

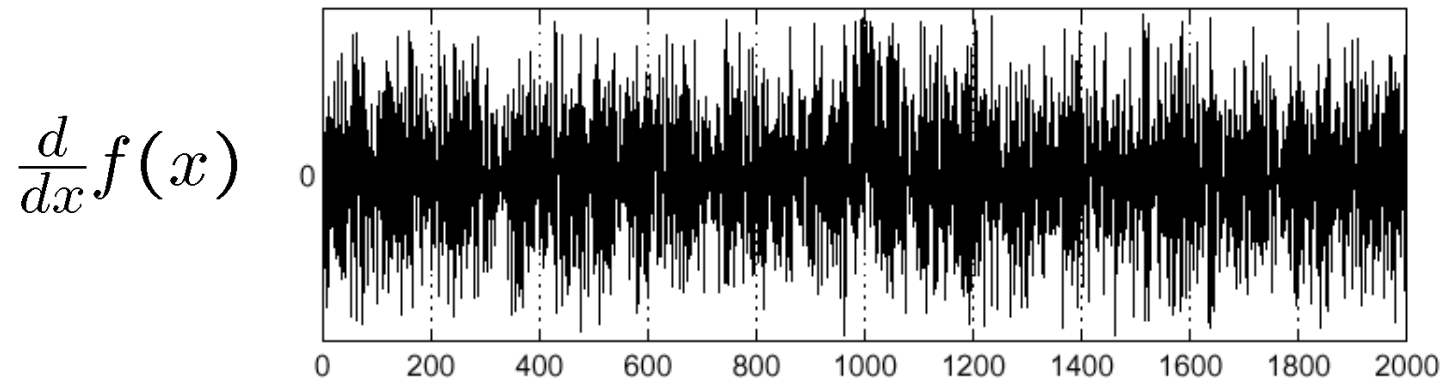
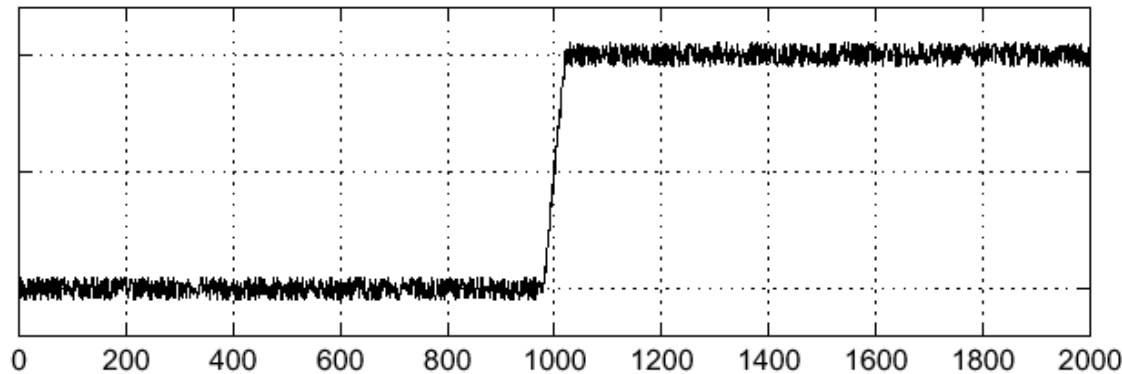
# Characterizing edges

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



# 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?

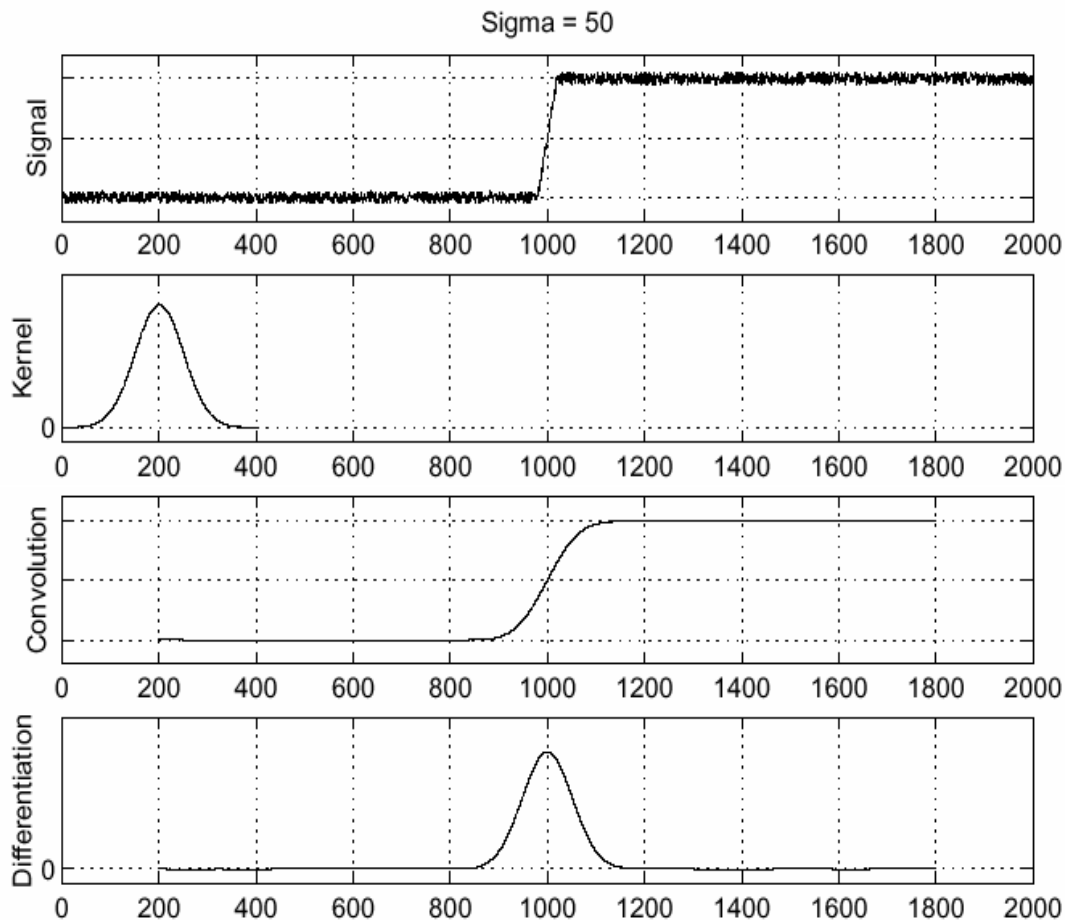
Source: S. Seitz

# 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?
  - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

# 1D Edge Detection

- Convolve the 1-D signal with a Gaussian kernel to give  $s(x)$ .
- Compute derivate of resulting smoothed signal to give  $s'(x)$ .
- Find maxima/minima of  $s'(x)$  and threshold



$$g_{\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

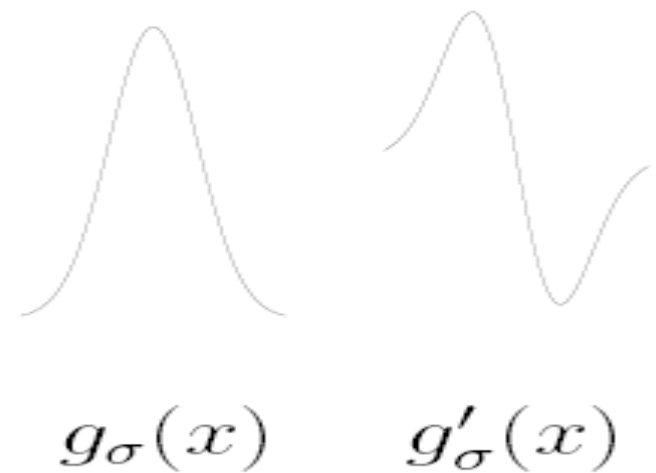
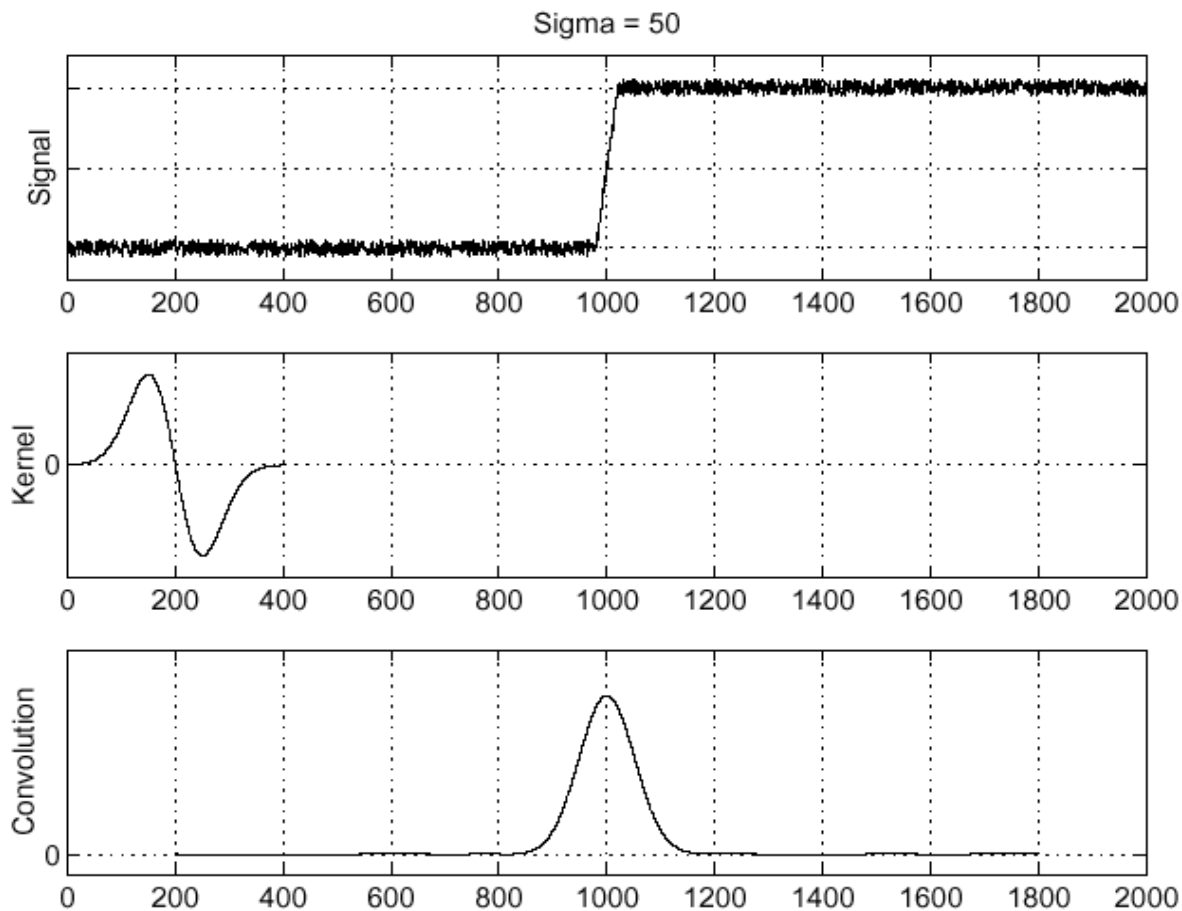
$$\begin{aligned} s(x) &= g_{\sigma}(x) * I(x) = \int_{-\infty}^{+\infty} g_{\sigma}(u) I(x-u) du \\ &= \int_{-\infty}^{+\infty} g_{\sigma}(x-u) I(u) du \end{aligned}$$

Slide: R. Cipolla, S. Seitz



# 1D Edge Detection

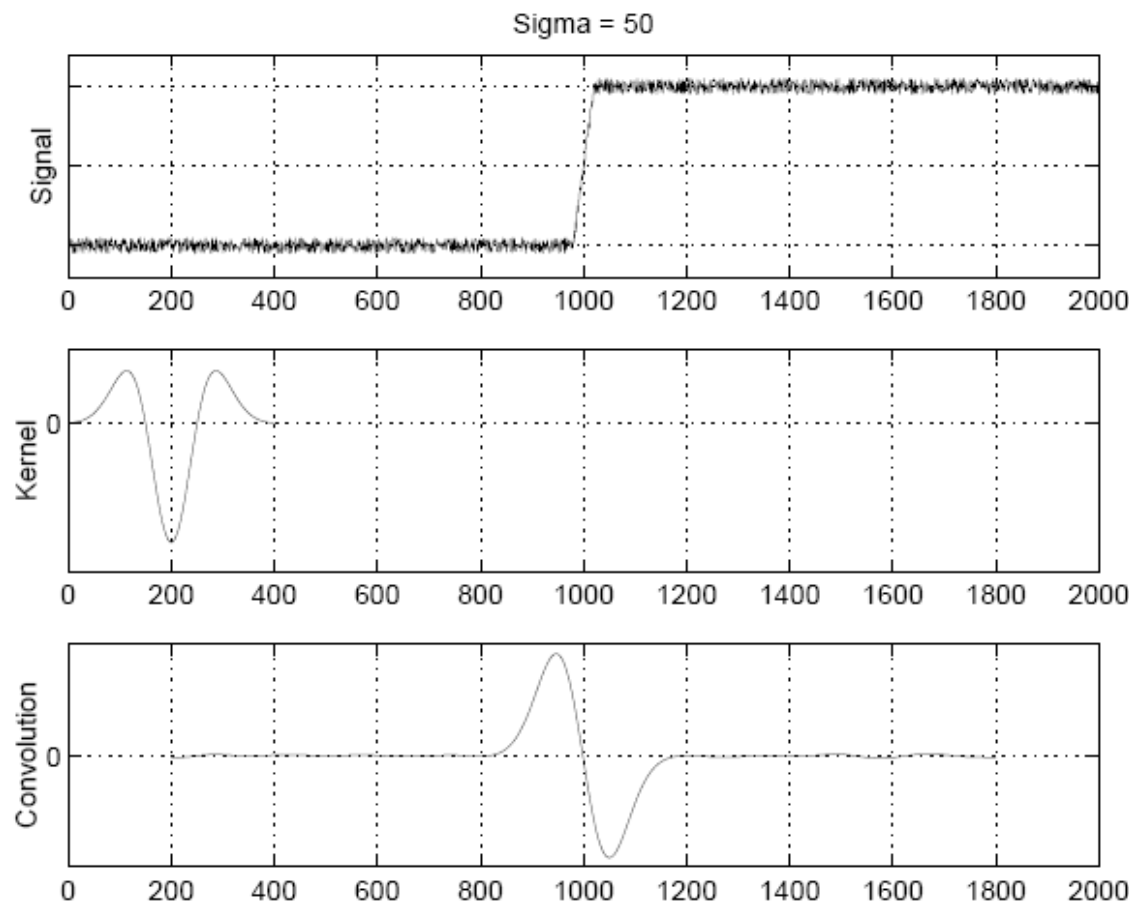
- Note that  $s'(x) = \frac{d}{dx} [g_\sigma(x) * I(x)] = g'_\sigma(x) * I(x)$



$$s''(x) = g''_\sigma(x) * I(x)$$

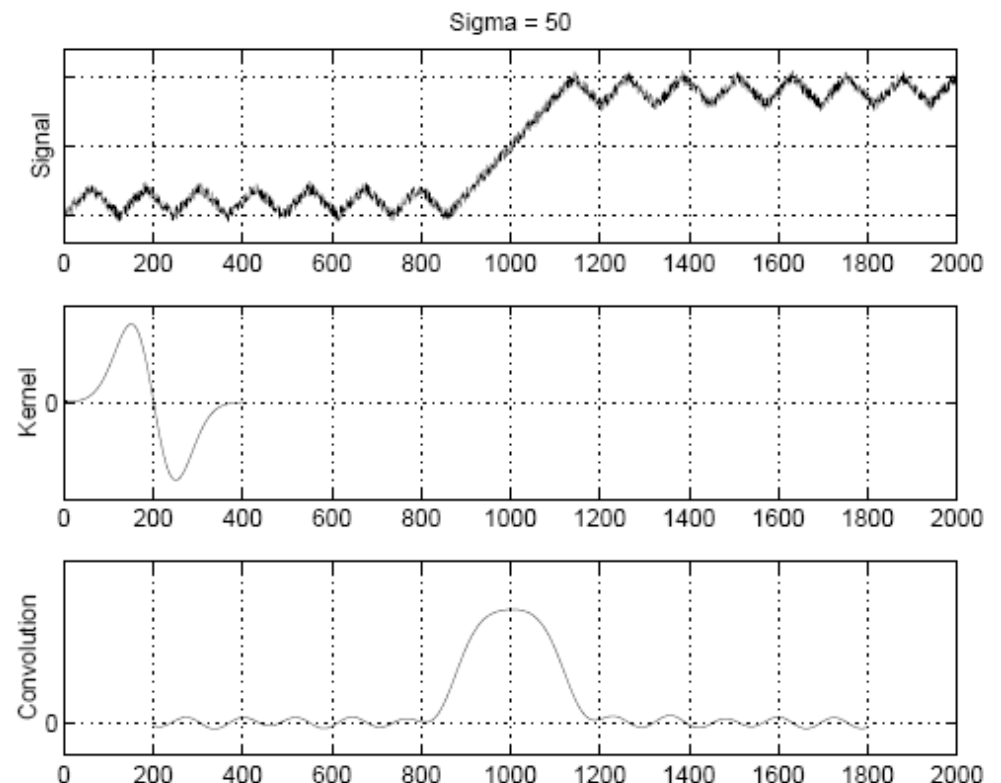
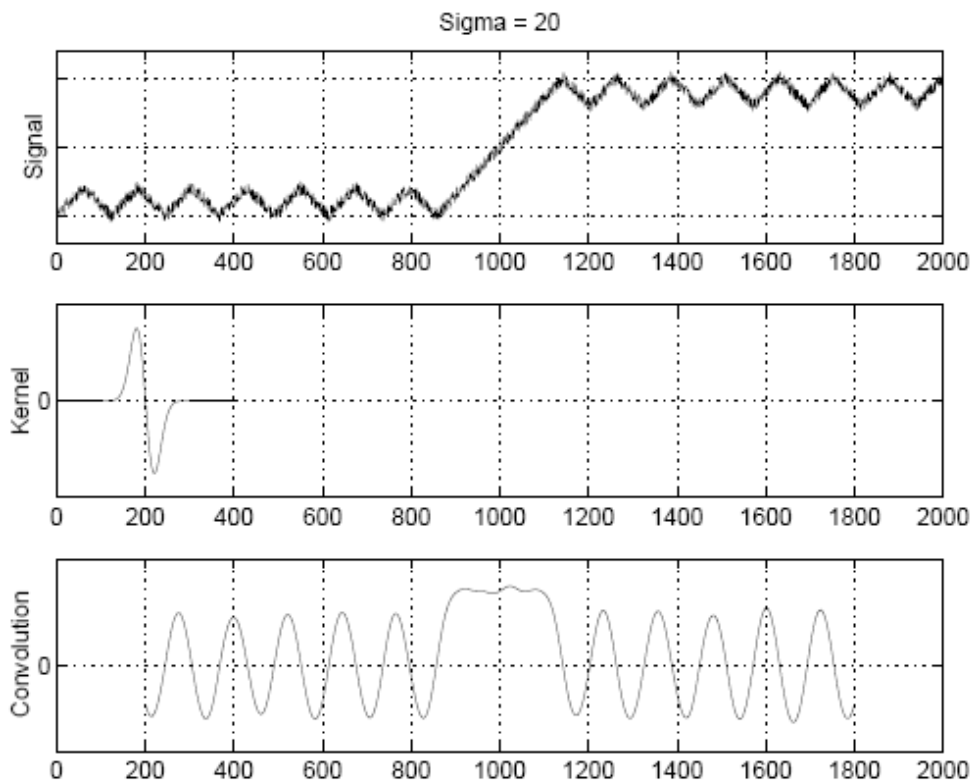
# 1D Edge Detection

- Looking for maxima/minima of  $s'(x)$  is same as zero-crossings of  $s''(x)$ , so easier to convolve with Laplacian of Gaussian,  $g_\sigma''(x)$ :



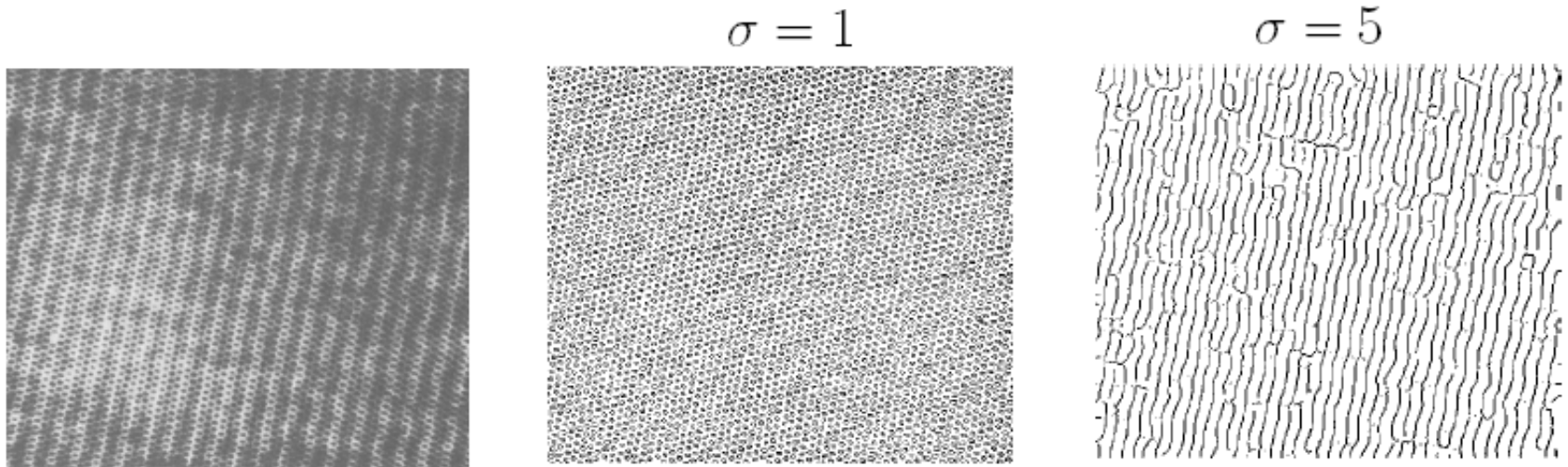
# Multi-scale 1D edge detection

- As  $\sigma$  increases, the smoothing increases and only the central edge survives.



# Multi-scale Edge Detection

- Degree of smoothing controls the scale at which we analyze the image
- Image of a dish cloth:



- Fine scales are sensitive to noise

# Derivatives of a 2D function

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

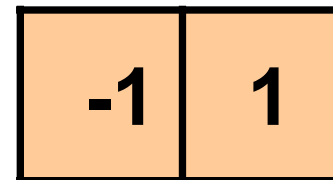
$$\frac{\partial f}{\partial x} = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}$$

- We could approximate this as

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

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

- It is obviously a convolution



# Finite difference filters

- Other approximations of derivative filters exist:

**Prewitt:**  $M_x =$ 

-1	0	1
-1	0	1
-1	0	1

 ;  $M_y =$ 

1	1	1
0	0	0
-1	-1	-1

**Sobel:**  $M_x =$ 

-1	0	1
-2	0	2
-1	0	1

 ;  $M_y =$ 

1	2	1
0	0	0
-1	-2	-1

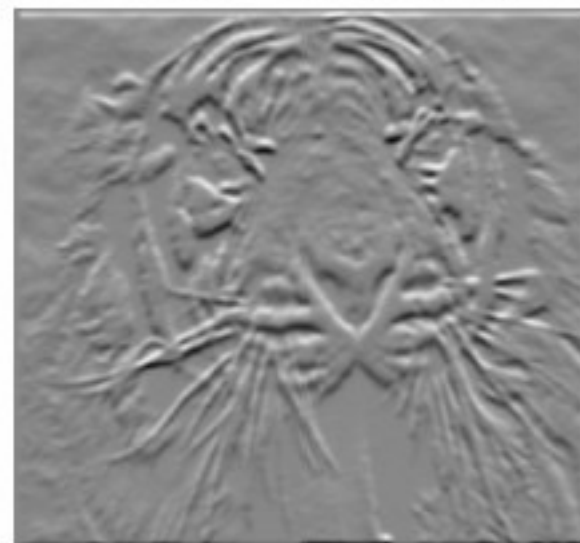
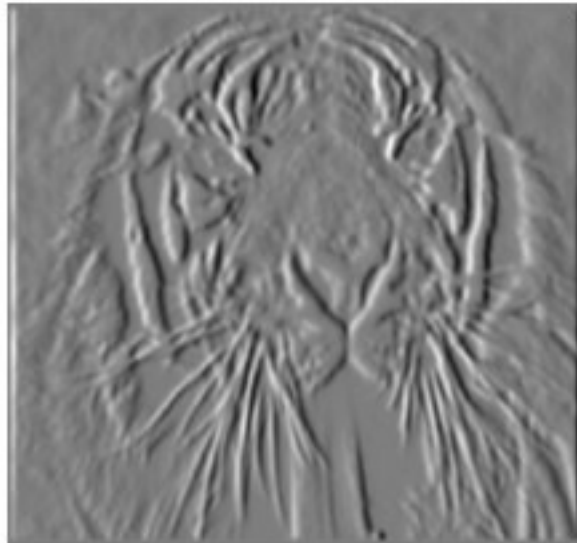
**Roberts:**  $M_x =$ 

0	1
-1	0

 ;  $M_y =$ 

1	0
0	-1

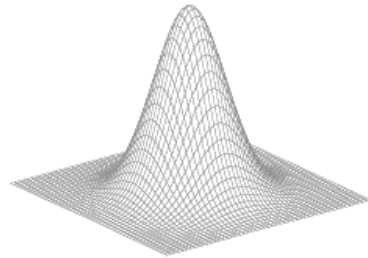
# Finite differences: example



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

# 2D Gaussian Filter

- Use 2D Gaussian kernel to smooth image



$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp - \left( \frac{x^2 + y^2}{2\sigma^2} \right)$$

- Now use 2D convolution operation

$$\begin{aligned} S(x, y) &= G_{\sigma}(x, y) * I(x, y) \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} G_{\sigma}(u, v) I(x - u, y - v) du dv \end{aligned}$$



# Blur with Gaussian filters

- $\sigma$  controls the blurring on an image:



Unsmoothed

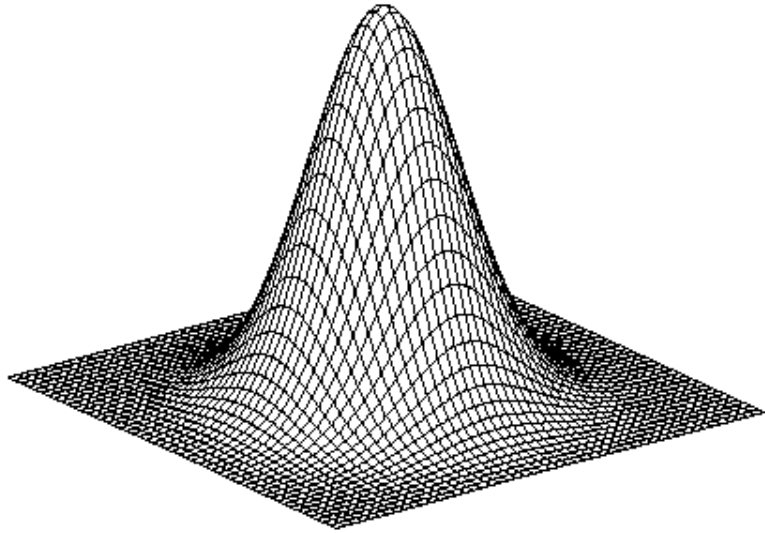


$\sigma = 3$  pixels

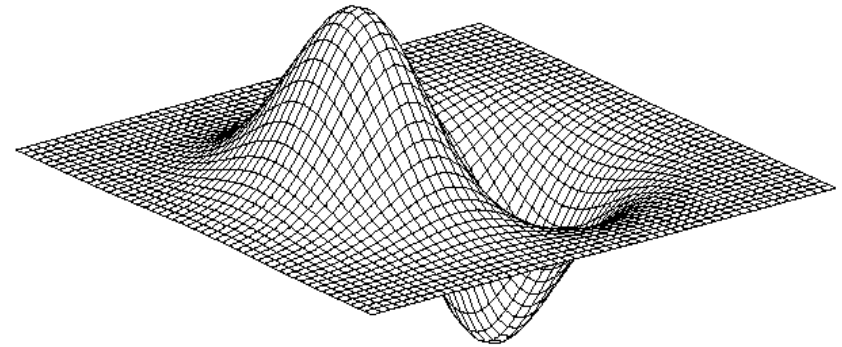


$\sigma = 4$  pixels

# Derivative of Gaussian filter

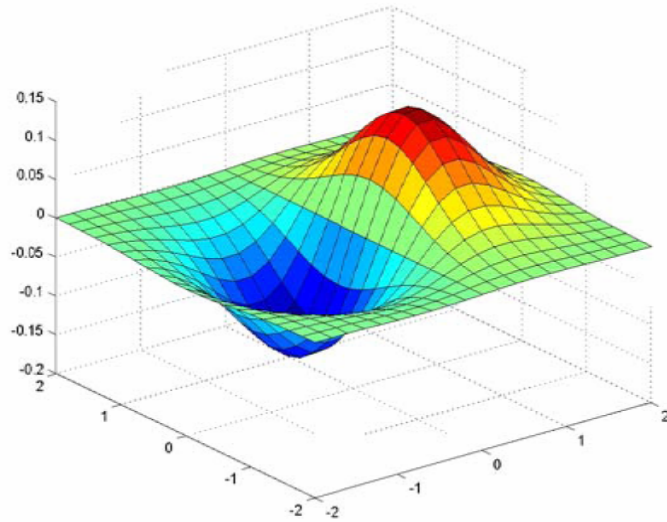


$$* \begin{bmatrix} 1 & -1 \end{bmatrix} =$$

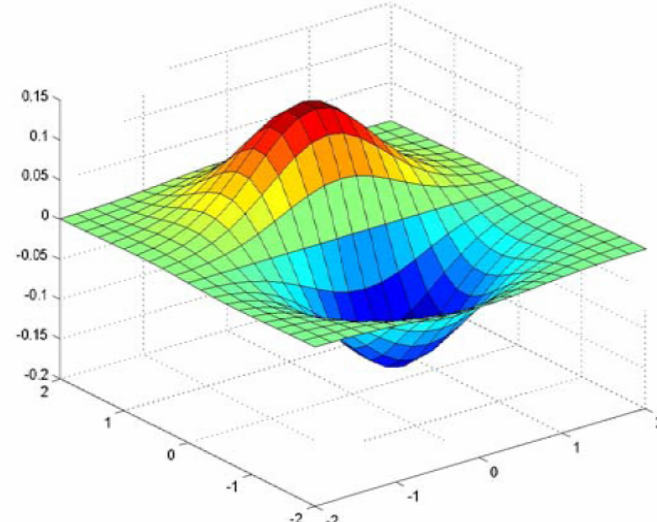
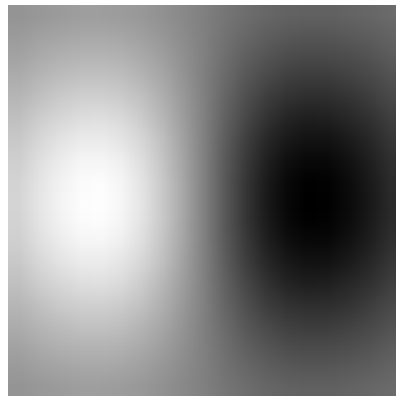


- Is this filter separable?

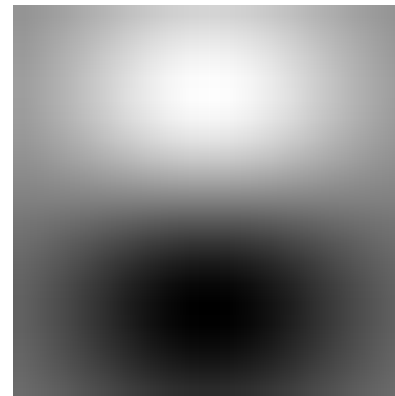
# Derivative of Gaussian filter



x-direction



y-direction



- Which one finds horizontal/vertical edges?

# 2D Edge Detection - Canny

- Now find gradient of the smoothed image  $S(x,y)$  at every pixel:

$$\begin{aligned}\nabla S &= \nabla(G_\sigma * I) \\ &= \begin{bmatrix} \frac{\partial(G_\sigma * I)}{\partial x} \\ \frac{\partial(G_\sigma * I)}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial G_\sigma}{\partial x} * I \\ \frac{\partial G_\sigma}{\partial y} * I \end{bmatrix}\end{aligned}$$



(a) Original image



(b) Edge strength  $|\nabla S|$

# 2D Edge Detection - Canny

- Apply non-maximal suppression. Edge elements or *edgels* are placed at locations where  $|\nabla S|$  is greater than local values of  $|\nabla S|$  in the directions  $\pm \nabla S$
- Then threshold



(c) Non-maximal suppression



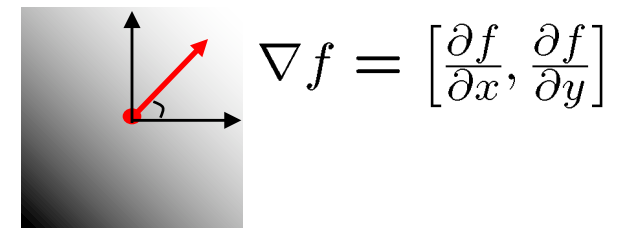
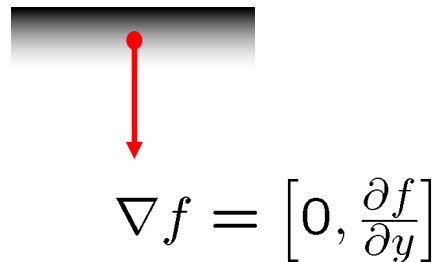
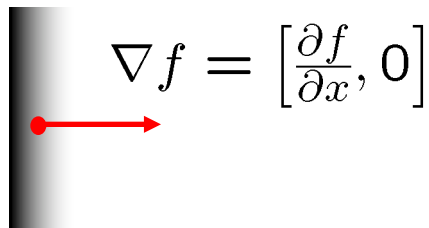
(d) Thresholding

- In Matlab, Canny edge detector is implemented in “edge”.

J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679-698, 1986.

# Image gradient

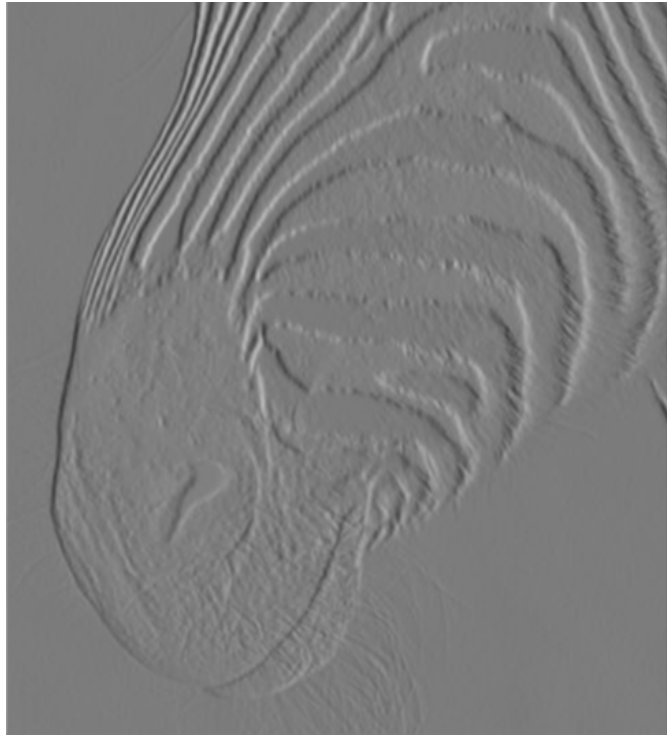
- The gradient of an image  $\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$



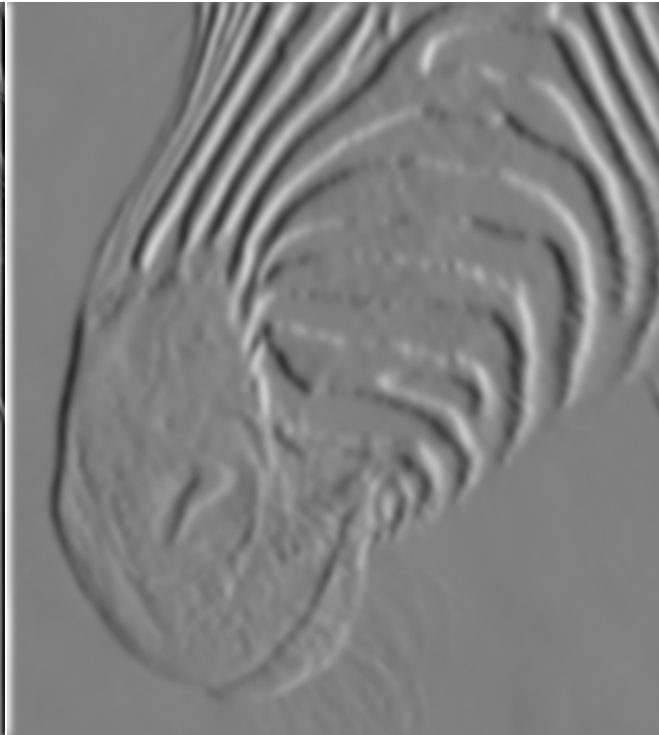
- The gradient points in the direction of most rapid increase in intensity
- The gradient direction is given by:  $\theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$
- The edge strength:  $\|\nabla f\| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2}$



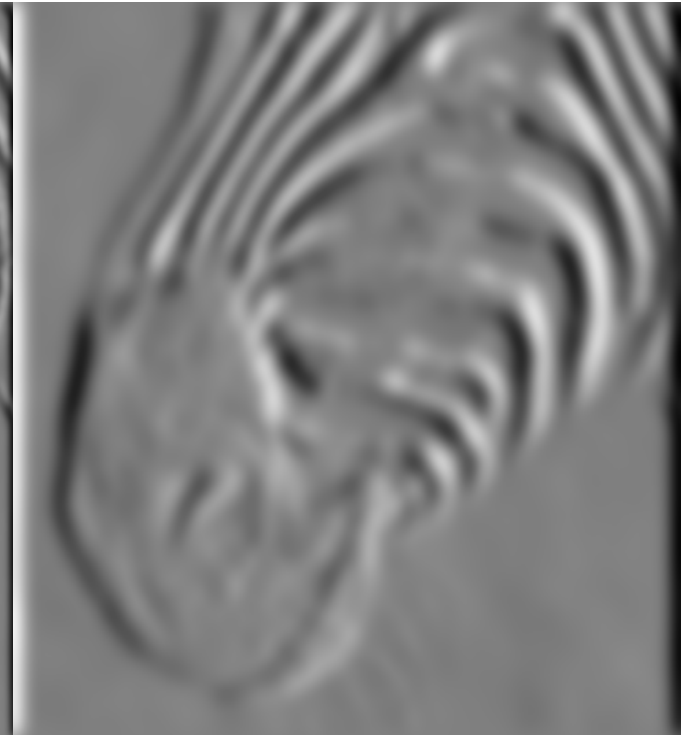
# Tradeoff: smoothing and localization



1 pixel



3 pixels



7 pixels

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

Source: D. Forsyth

# Implementation issues



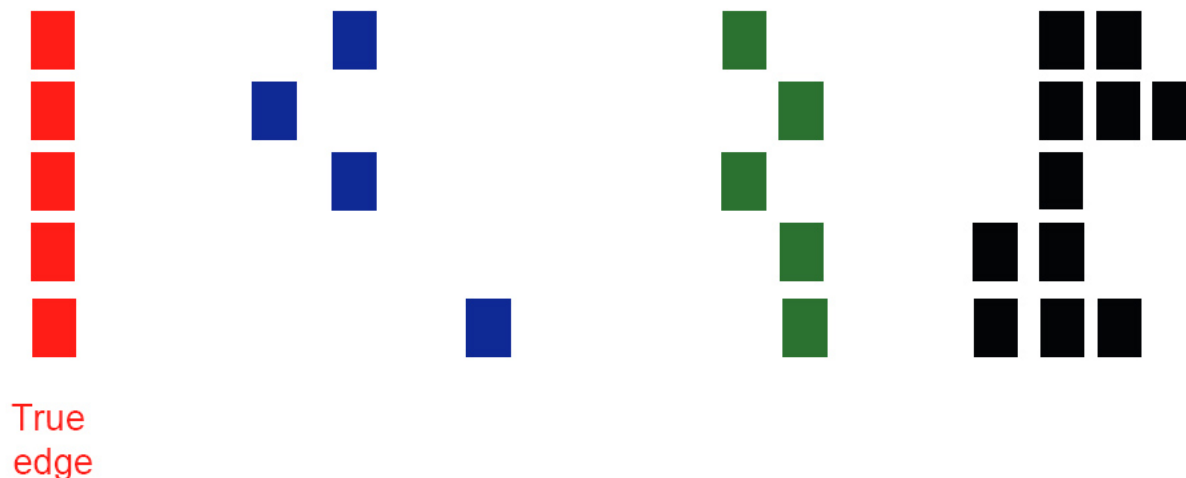
- The gradient magnitude is large along a thick “trail” or “ridge,” so how do we identify the actual edge points?
- How do we link the edge points to form curves?

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



Source: L. Fei-Fei

# 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



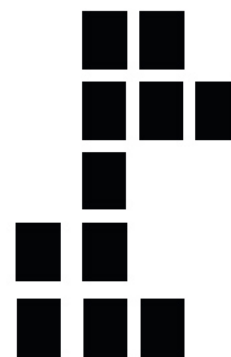
True  
edge



Poor robustness  
to noise



Poor  
localization



Too many  
responses

Source: L. Fei-Fei

# Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Requirements
  - Good localization
  - Few false positives
- Canny: optimization problem
  - Numerical optimization (the product of signal-to-noise ratio and localization) w.r.t filter shape
  - Derivative of Gaussian approximates optimal operator  $z$

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
5. MATLAB: `edge(image, 'canny')`

Source: D. Lowe, L. Fei-Fei

# Example



original image (Lena)

# Example



Norm of gradient

# Example



Thresholding

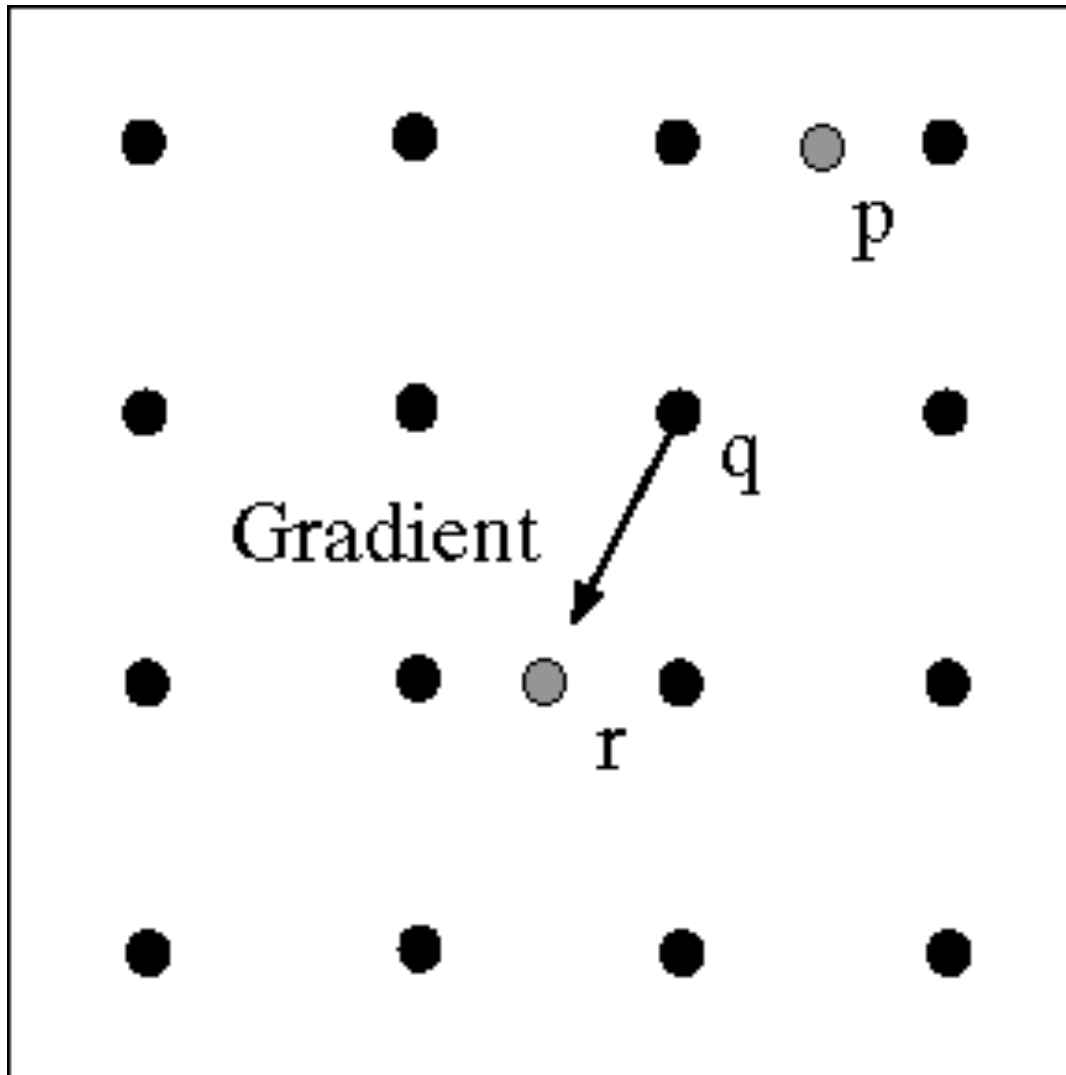
# Example



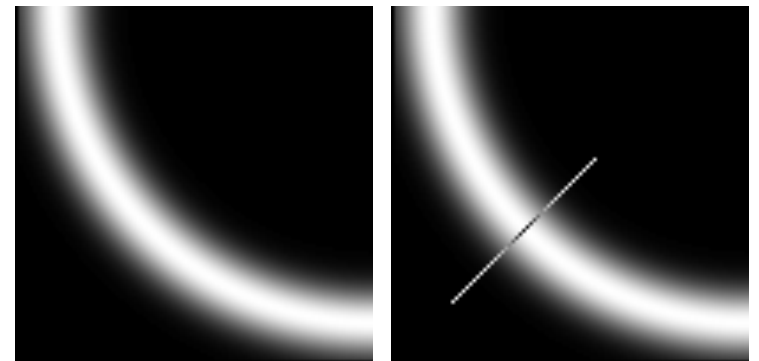
Thinning  
(non-maximum suppression)



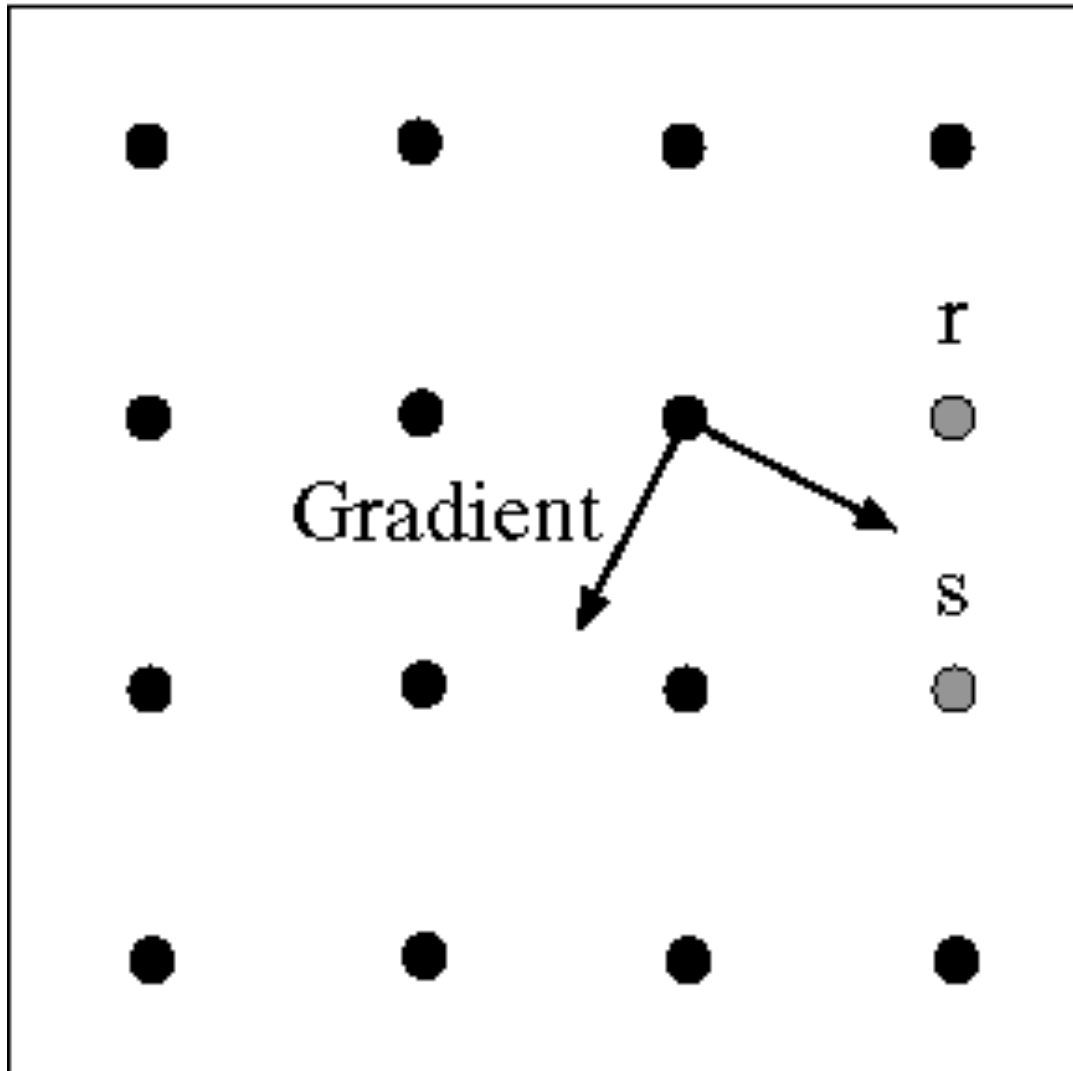
# Non-maximum suppression



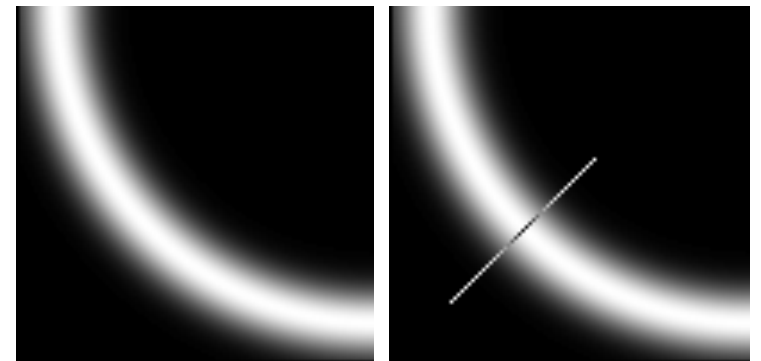
At  $q$ , we have a maximum if the value is larger than those at both  $p$  and at  $r$ . Interpolate to get these values.



# Edge linking

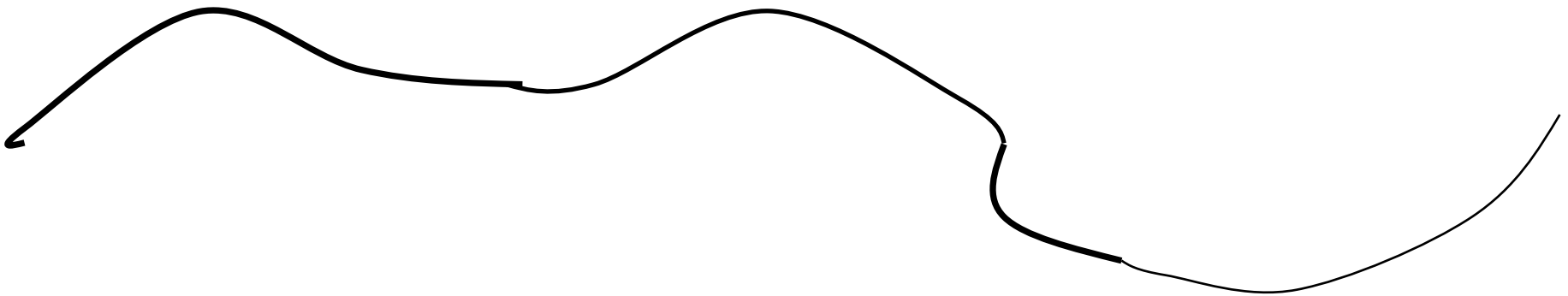


Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



# 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.



# Example



$K_{low} = 0.2$



$K_{high} = 0.5$

Hysteresis thresholding



# Example



original image



high threshold  
(strong edges)



low threshold  
(weak edges)



hysteresis threshold

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



original



Canny with  $\sigma = 1$



Canny with

The choice of  $\sigma$  depends on desired behavior

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

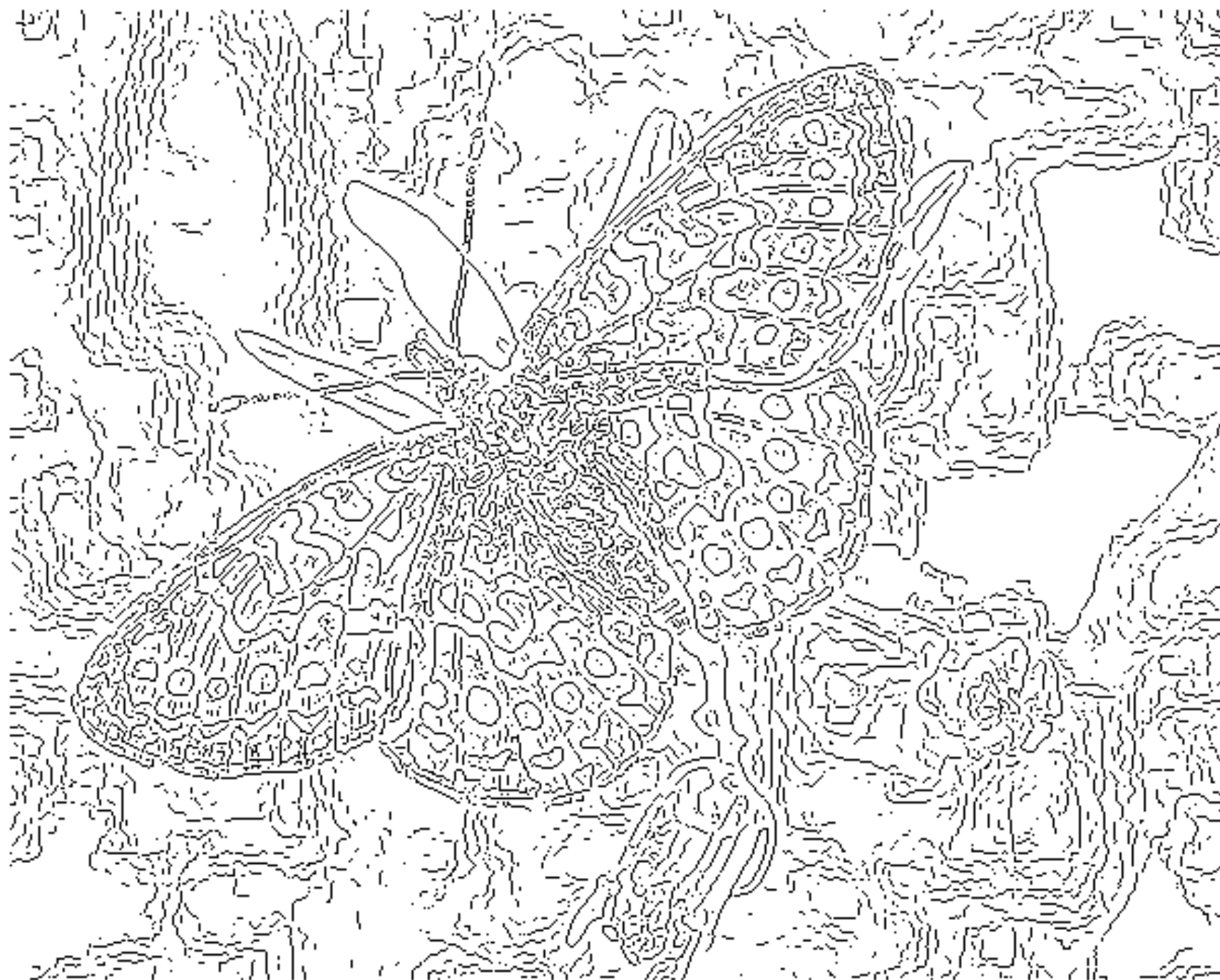
Source: S. Seitz

# Notice

- Something nasty is happening at corners
- Scale affects contrast
- Edges aren't bounding contours







fine scale  
high  
threshold



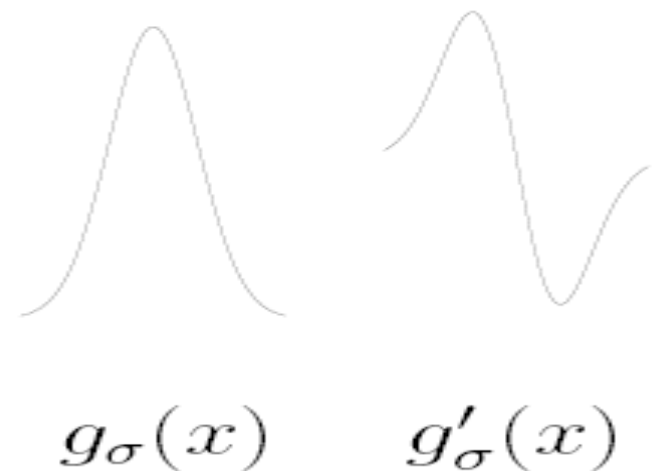
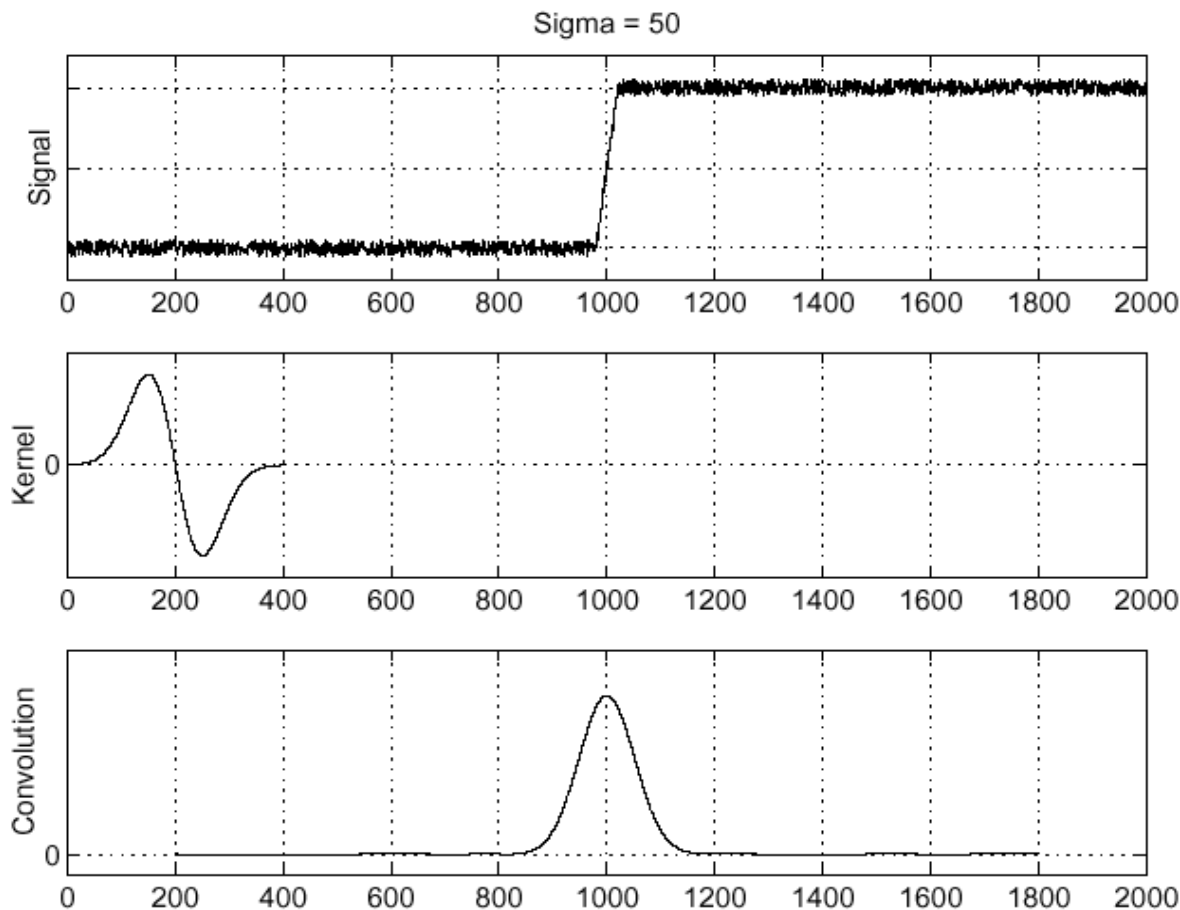
coarse  
scale,  
high  
threshold



coarse  
scale  
low  
threshold

# Review: Edge = local extreme

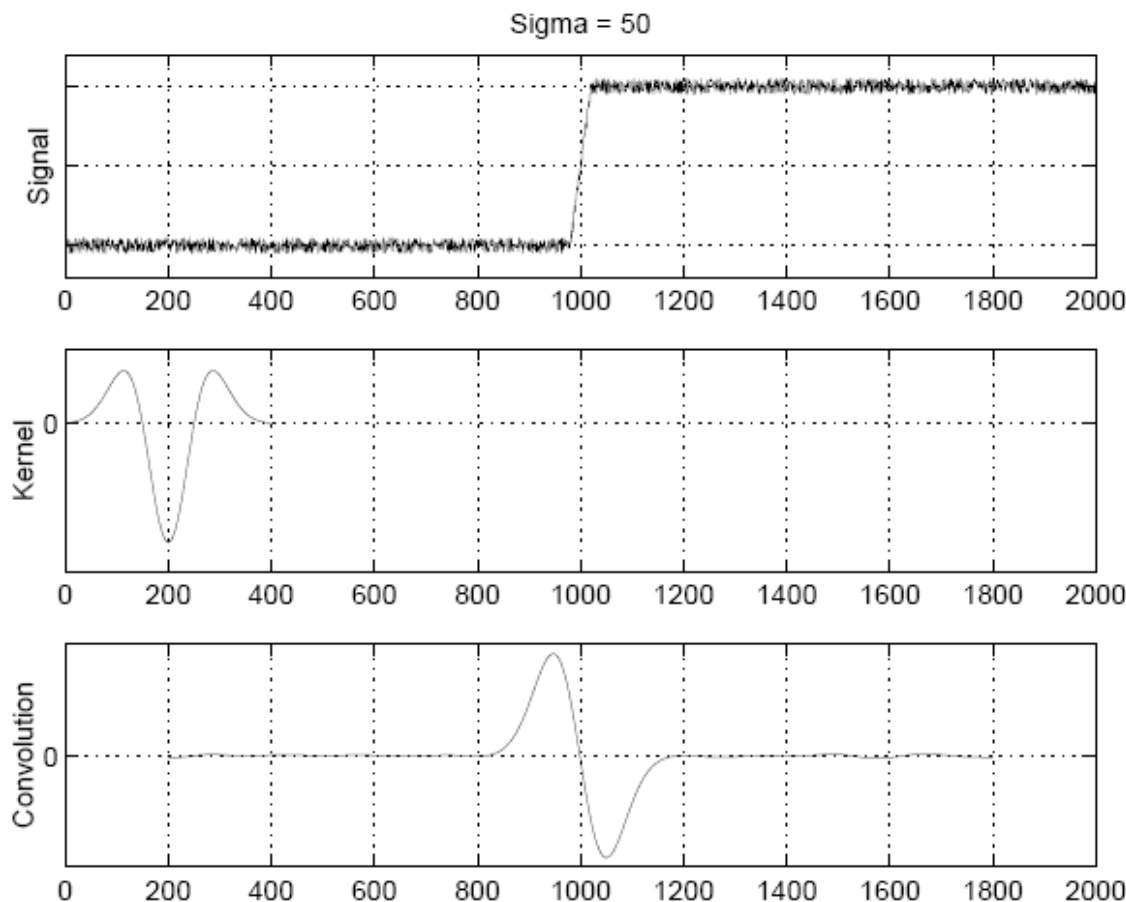
- Note that  $s'(x) = \frac{d}{dx} [g_\sigma(x) * I(x)] = g'_\sigma(x) * I(x)$



$$s''(x) = g''_\sigma(x) * I(x)$$

# Review: Laplacian operator

- Looking for maxima/minima of  $s'(x)$  is same as zero-crossings of  $s''(x)$ , so easier to convolve with Laplacian of Gaussian,  $g_\sigma''(x)$ :



# Laplacian Operator

- Laplacian operator  $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
- Apply Laplacian operator on an image:

$$\begin{aligned}\frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} &= [-1, 2, 1] * I + [-1, 2, 1]^T * I \\ &= \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * I\end{aligned}$$

- Some Laplacian filters

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \frac{1}{3} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

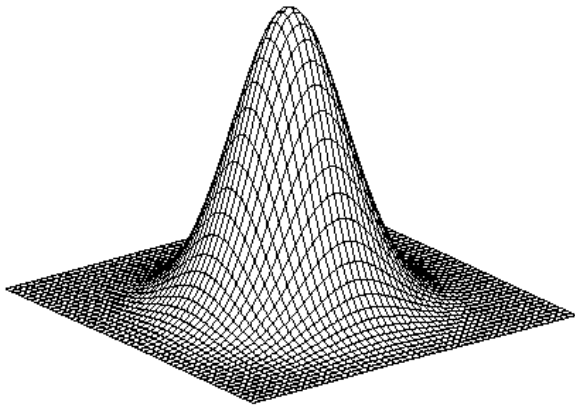
- In Matlab: `fspecial('Laplacian', alpha)`

# Laplacian of Gaussian

- Apply Laplacian operator on a Gaussian-blurred image is equivalent to convolving with Laplacian of Gaussian:

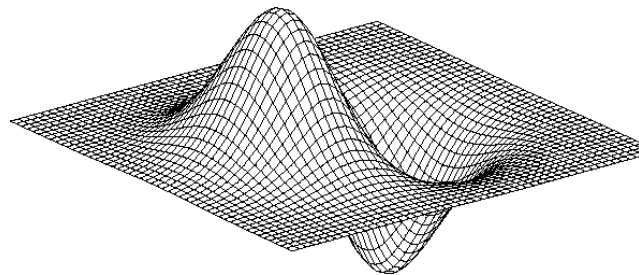
$$\nabla^2(G * I) = \frac{\partial^2(G * I)}{\partial x^2} + \frac{\partial^2(G * I)}{\partial y^2} = \left( \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} \right) * I$$

Laplacian of Gaussian

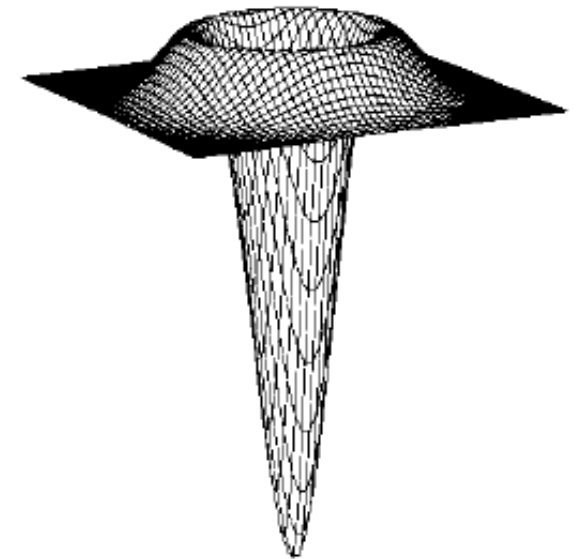


Gaussian

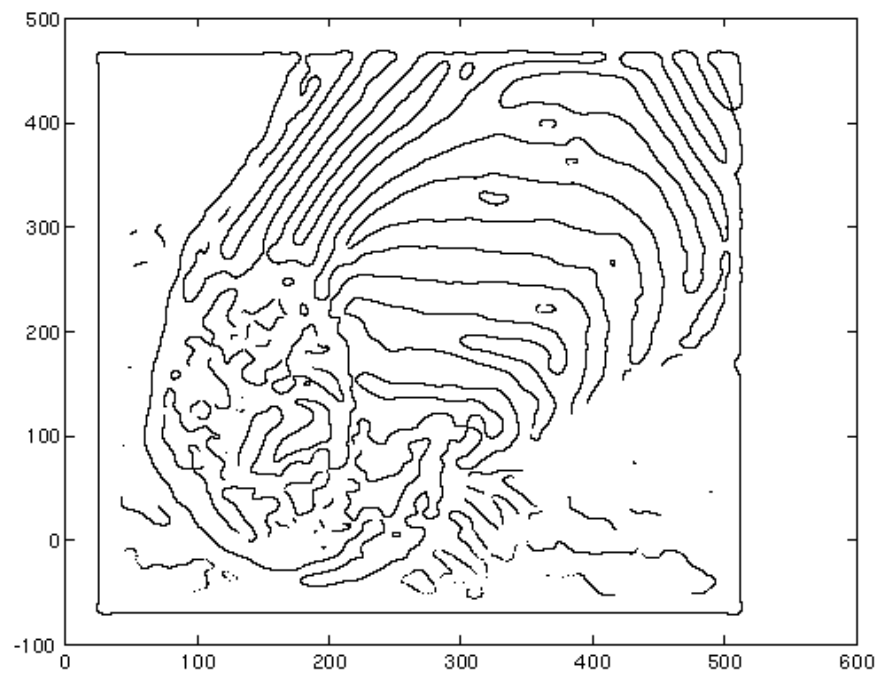
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



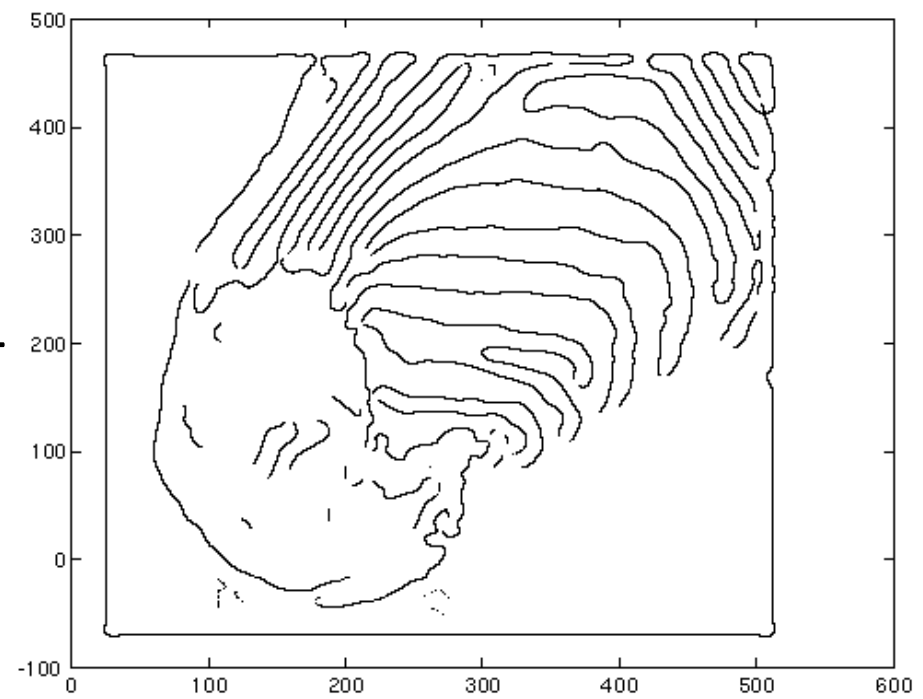
Derivative of Gaussian



$$\nabla^2 G(x, y, \sigma) = \left( \frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2} \right) G(x, y, \sigma)$$

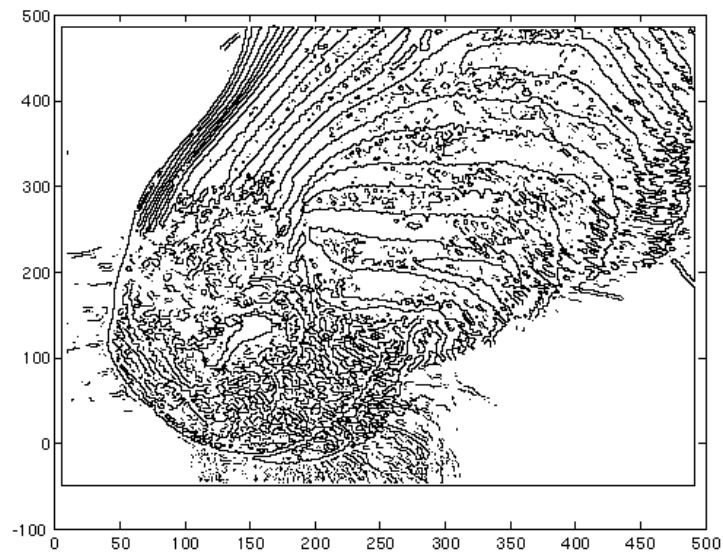


sigma=4



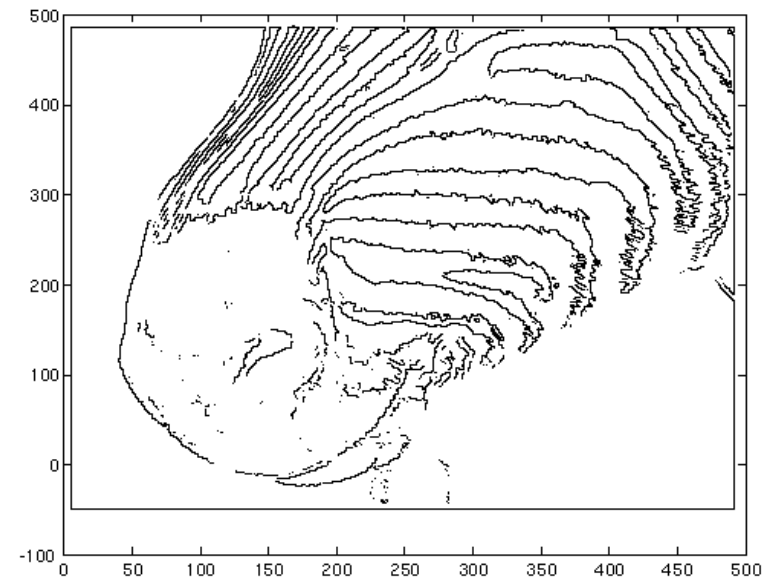
LOG zero crossings

contrast=1



sigma=2

contrast=4





# Orientation representations

- The gradient magnitude is affected by illumination changes
  - but it's direction isn't
- We can describe image patches by the swing of the gradient orientation
- Important types:
  - constant window
    - small gradient mags
  - edge window
    - few large gradient mags in one direction
  - flow window
    - many large gradient mags in one direction
  - corner window
    - large gradient mags that swing

# Disclaimer

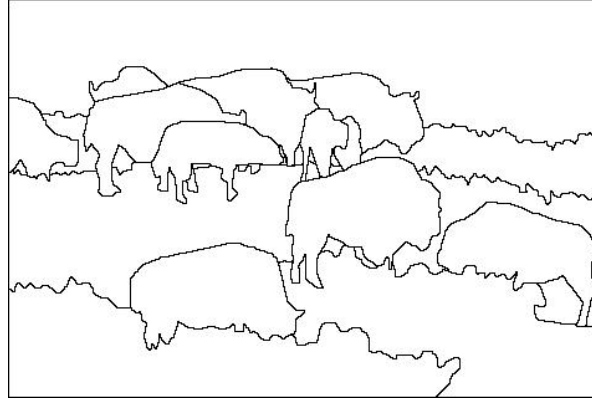
- No edge detection scheme is going to work perfectly in all cases. This is due to the fact that our notion of what constitutes a salient edge in the image is actually somewhat subtle.

# Edge detection is just the beginning...

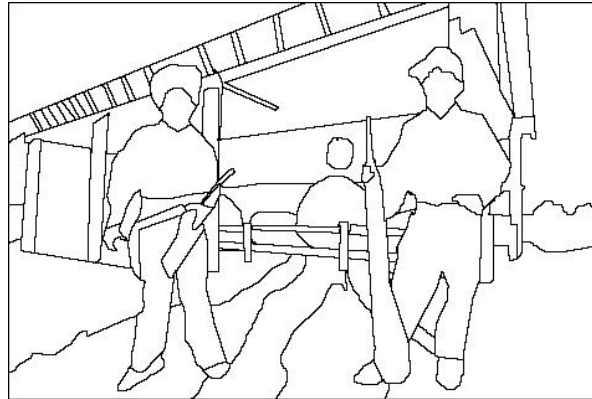
image



human segmentation



gradient magnitude



- Berkeley segmentation database:

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

# Lecture summary

- Edge detection
- Convolution with Gaussian derivative
- Laplacian of Gaussian (LOG)
- Canny edge detector

# Some references

- Read: Szeliski 4.1-4.3
- Some good explanation of Laplacian operator
  - <https://www.youtube.com/watch?v=EW08rD-GFh0>
  - <https://www.youtube.com/watch?v=XbCvGRjjzgg>
  - <https://www.youtube.com/watch?v=AlXVrAOls-8>