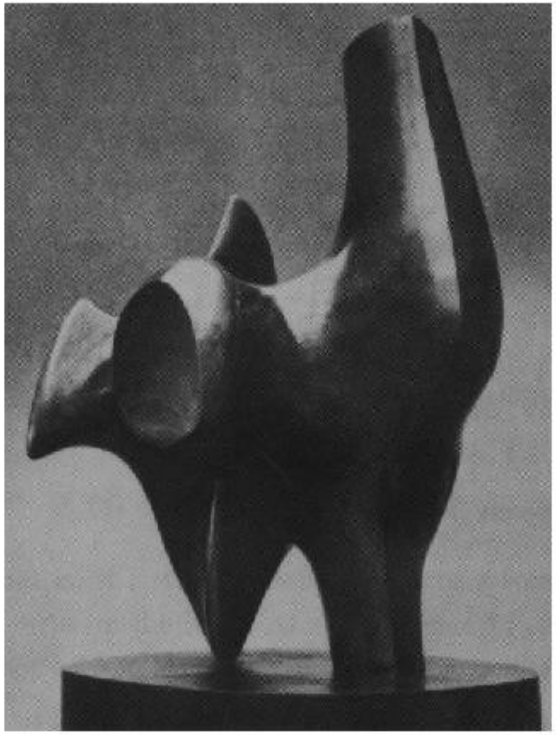
CS5670: Computer Vision

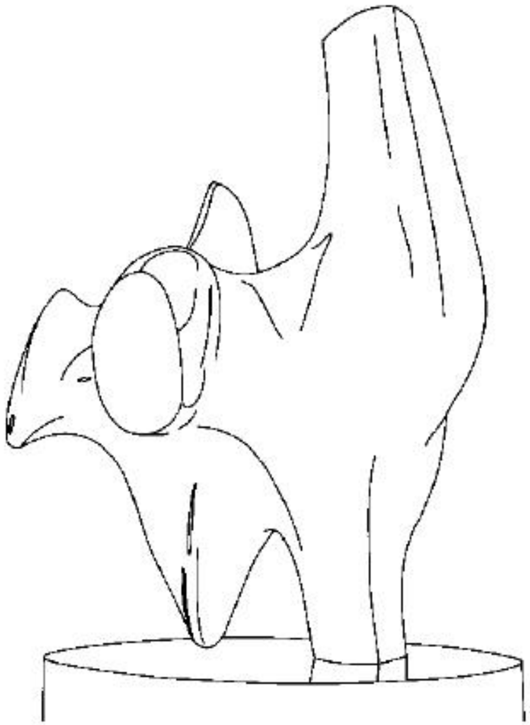
Noah Snavely

Lecture 2: Edge detection

From Sandlot Science

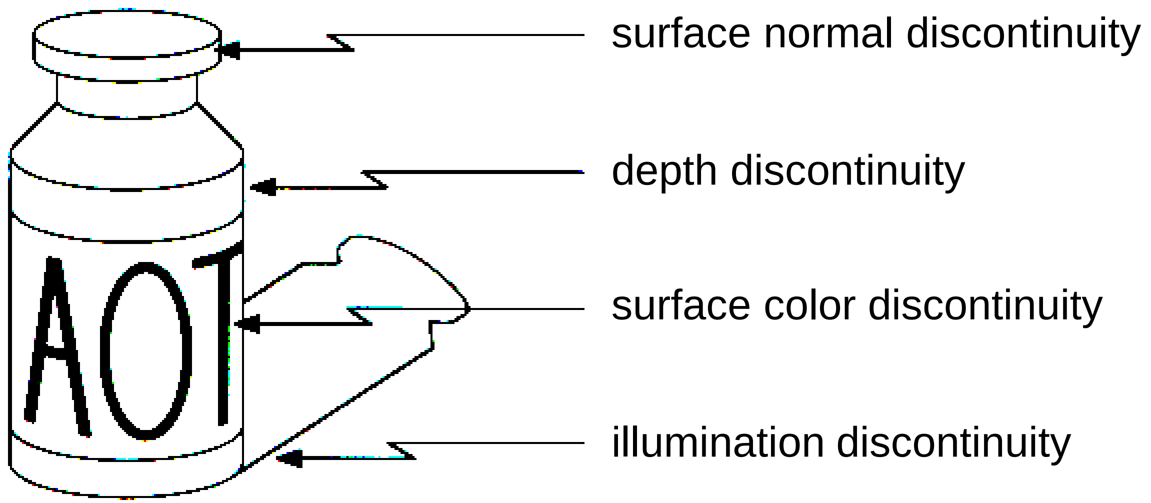
Edge detection





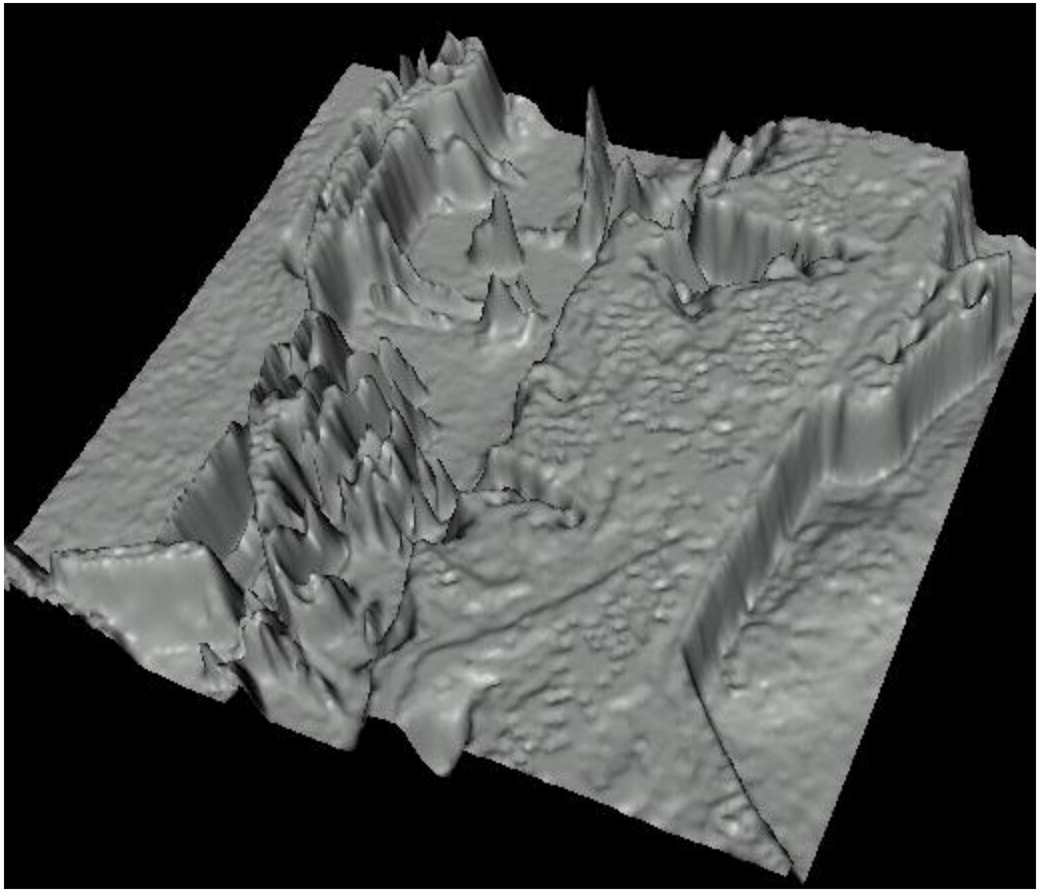
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Origin of Edges



- Edges are caused by a variety of factors

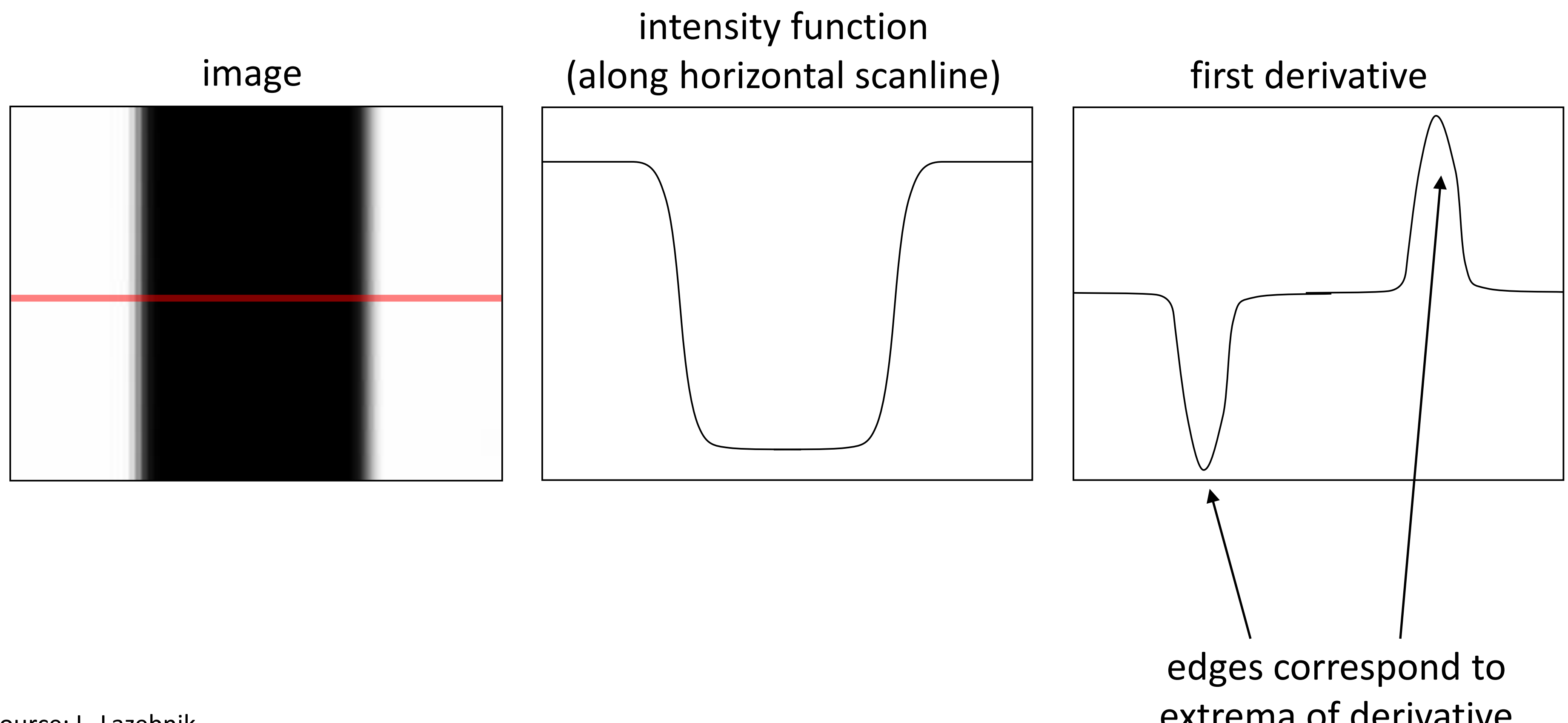
Images as functions…



- Edges look like steep cliffs

Characterizing edges

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



Source: L. Lazebnik

Image derivatives

- How can we differentiate a *digital* image F[x,y]?
  - Option 1: reconstruct a continuous image, *f,* then compute the derivative
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

How would you implement this as a linear filter?

$\frac{\partial f}{\partial x}$:

H_x

$\frac{\partial f}{\partial y}$:

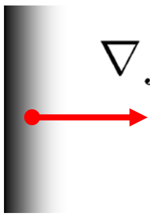
H_y

Source: S. Seitz

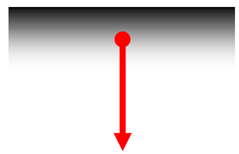
Image gradient

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

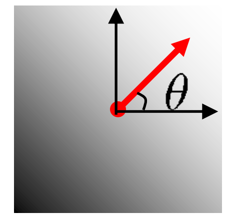
The gradient points in the direction of most rapid increase in intensity



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



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



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

The *edge strength* is given by the gradient magnitude:

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

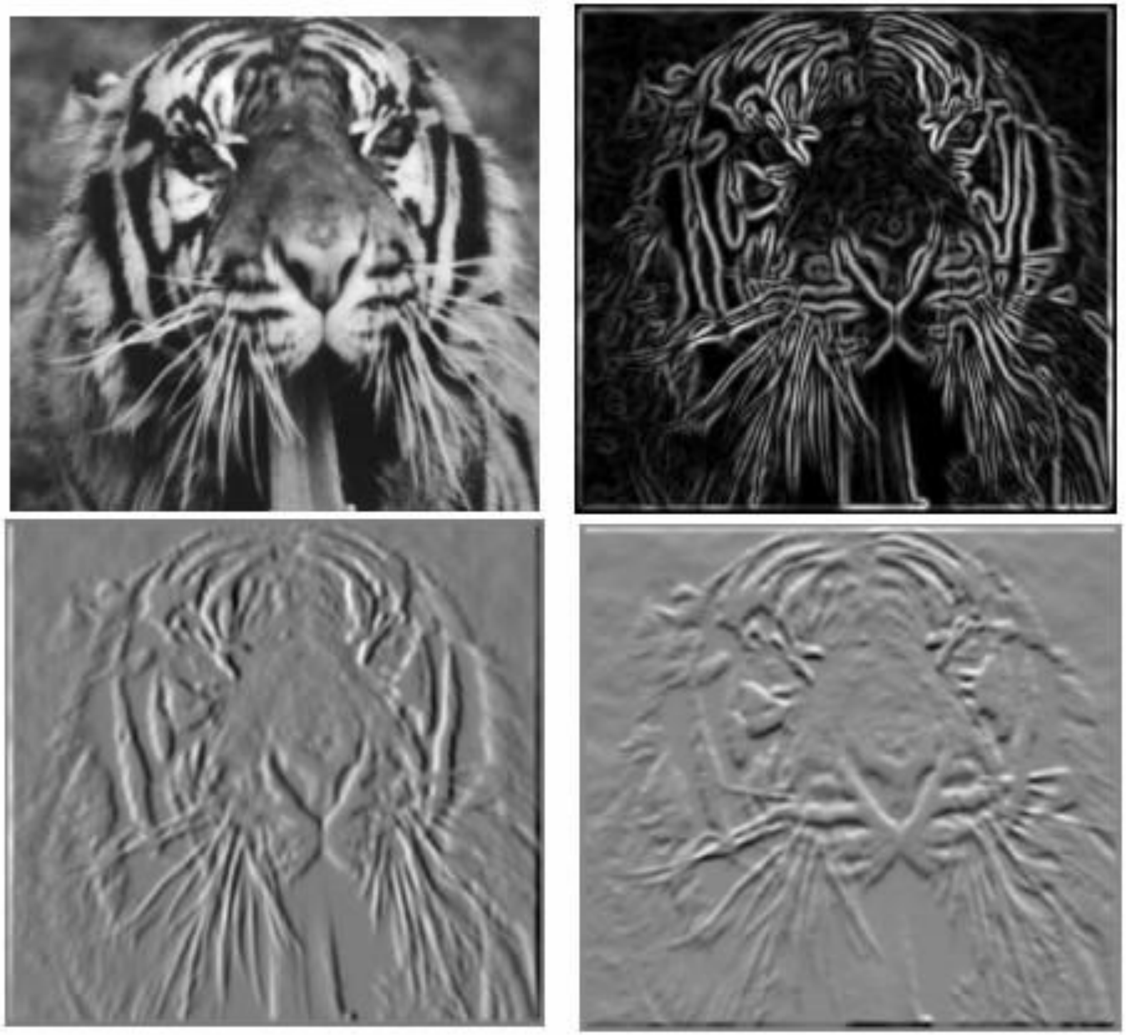
The gradient direction is given by:

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

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

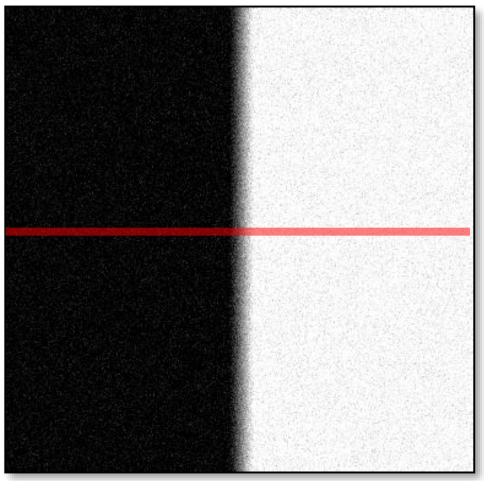
Source: Steve Seitz

Image gradient

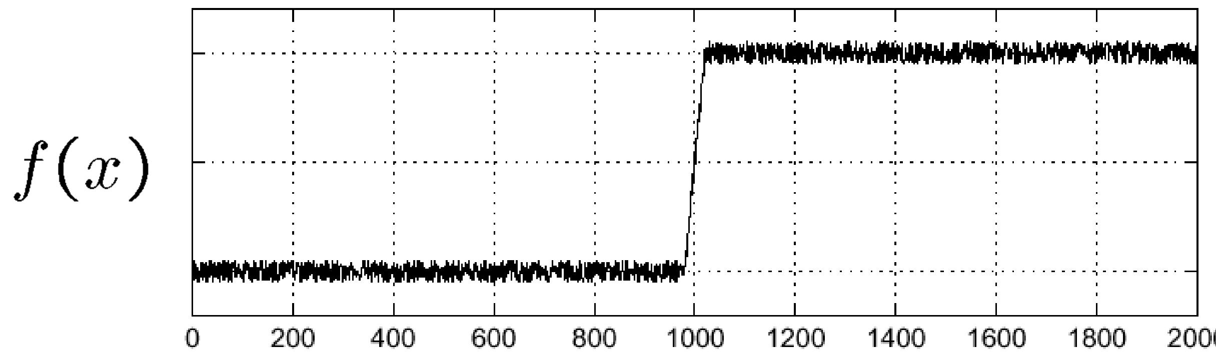


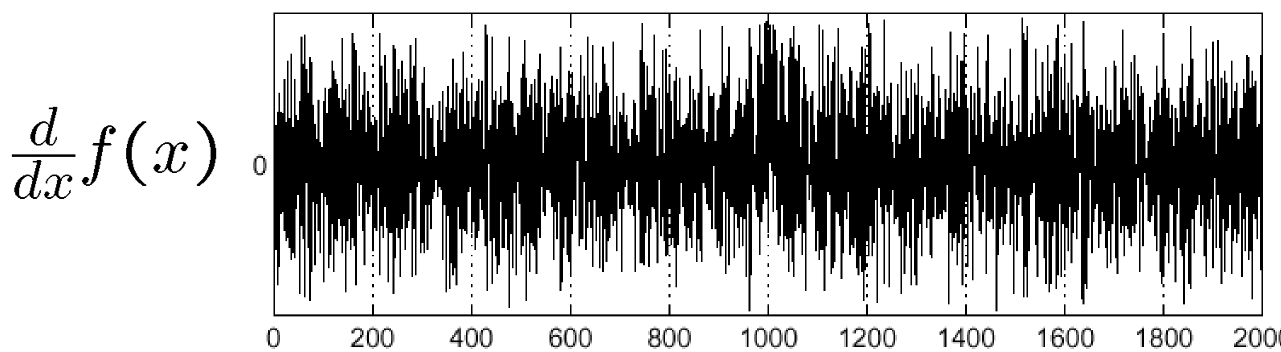
Source: L. Lazebnik

Effects of noise



Noisy input image

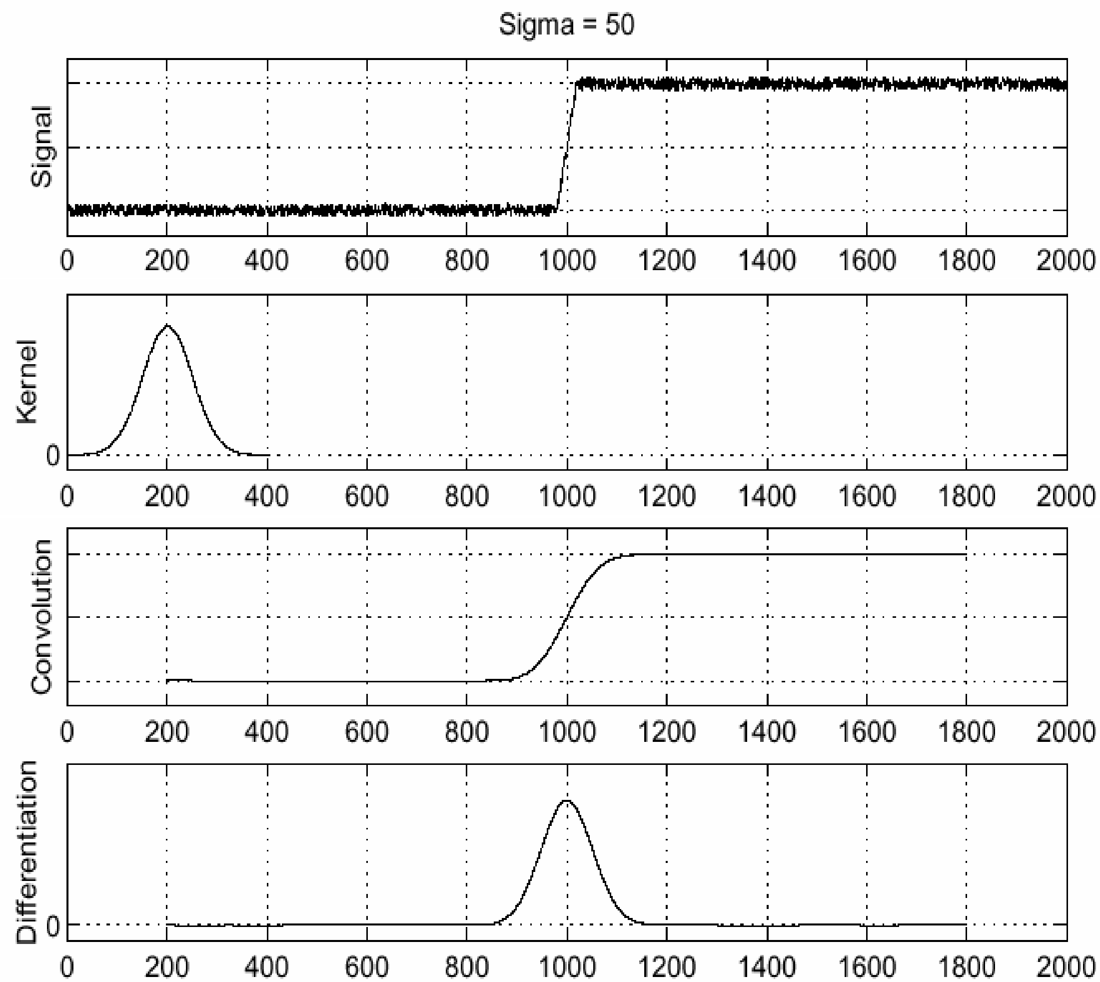




Where is the edge?

Source: S. Seitz

Solution: smooth first



f

h

$f * h$

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

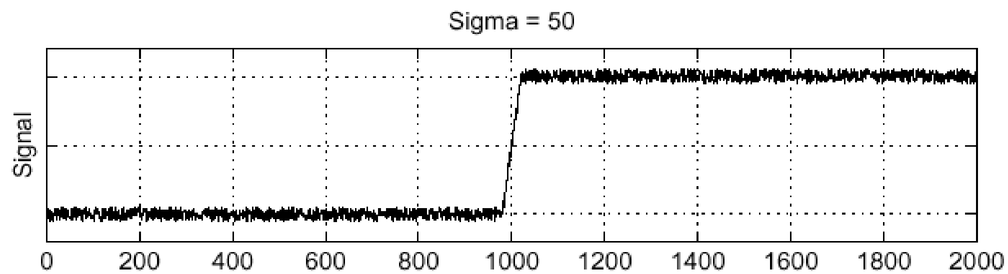
To find edges, look for peaks in $\frac{d}{dx}(f * h)$

Source: S. Seitz

Associative property of convolution

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

f

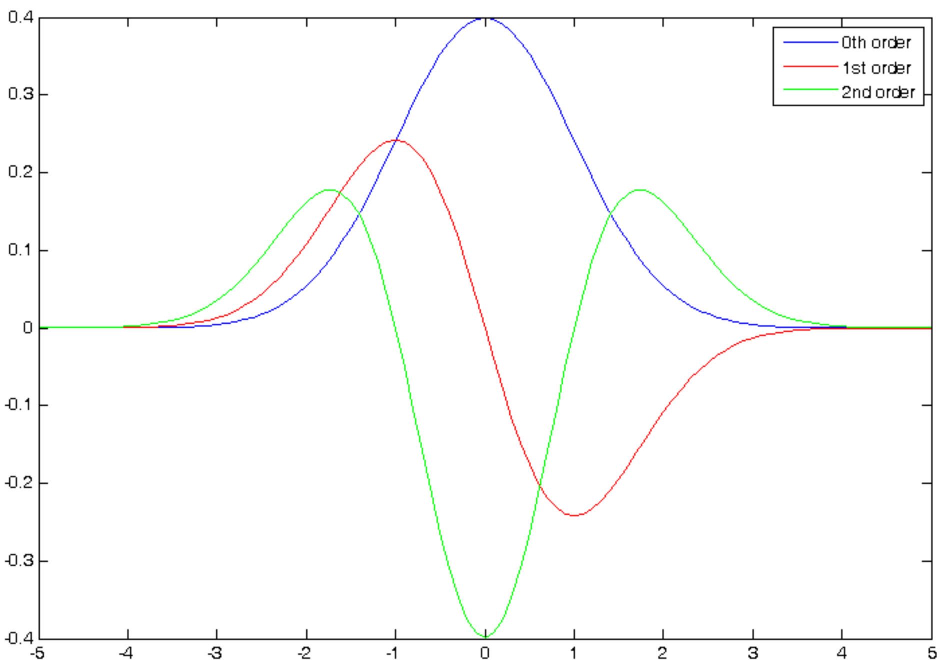


Source: S. Seitz

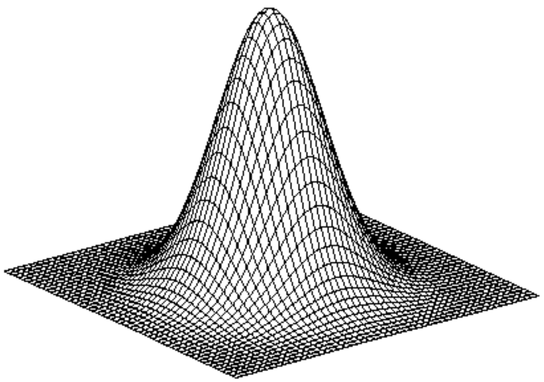
The 1D Gaussian and its derivatives

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma}\left(\frac{x}{\sigma}\right) G_\sigma(x)$$

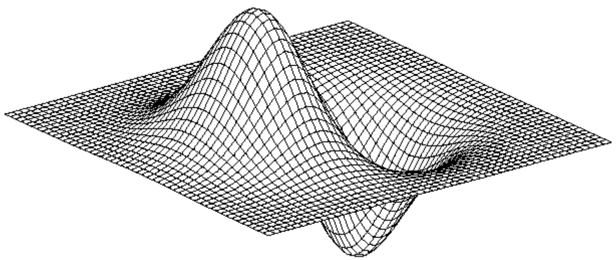


2D edge detection filters



Gaussian

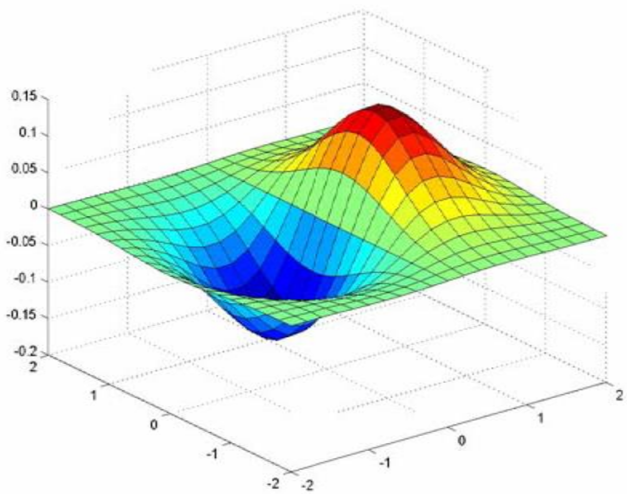
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



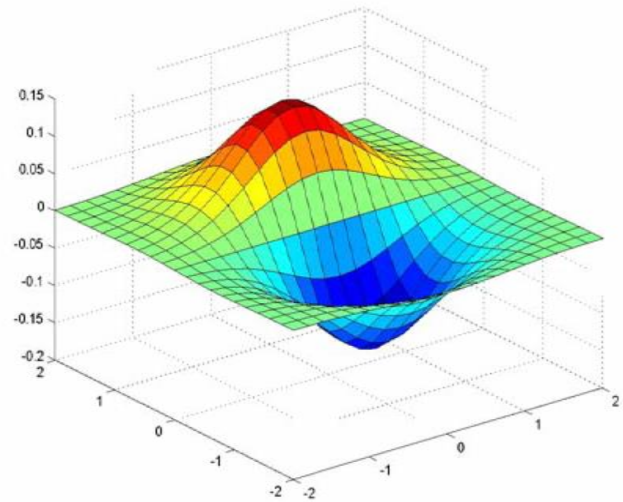
derivative of Gaussian (*x*)

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

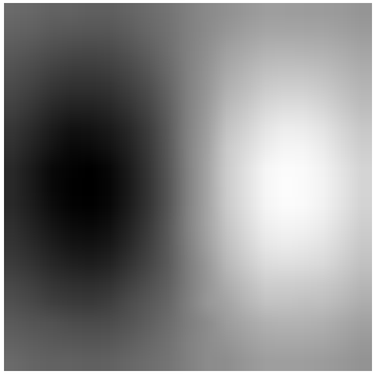
Derivative of Gaussian filter

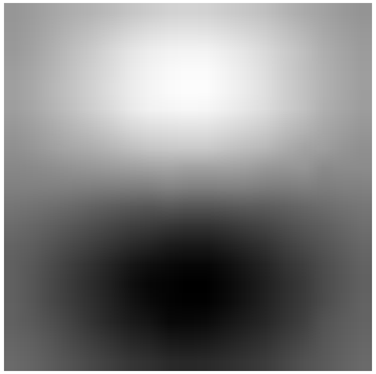


x-direction



y-direction





The Sobel operator

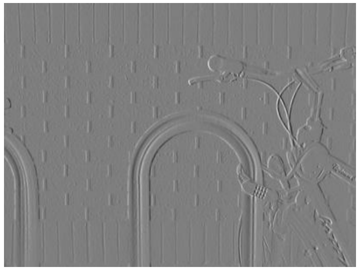
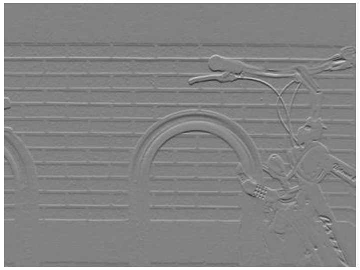
- Common approximation of derivative of Gaussian

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$s_x \qquad\qquad\qquad\qquad s_y$$

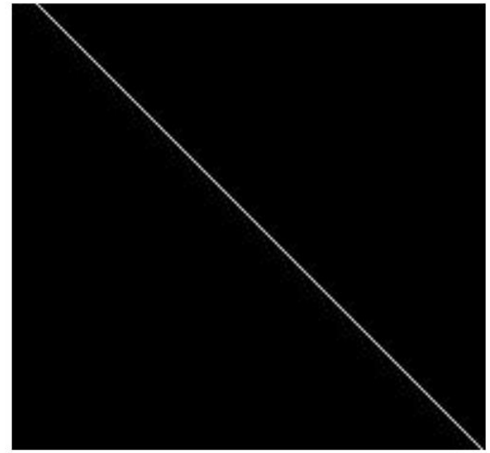
- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient magnitude

Sobel operator: example


Source: Wikipedia

Image with Edge



Edge Location

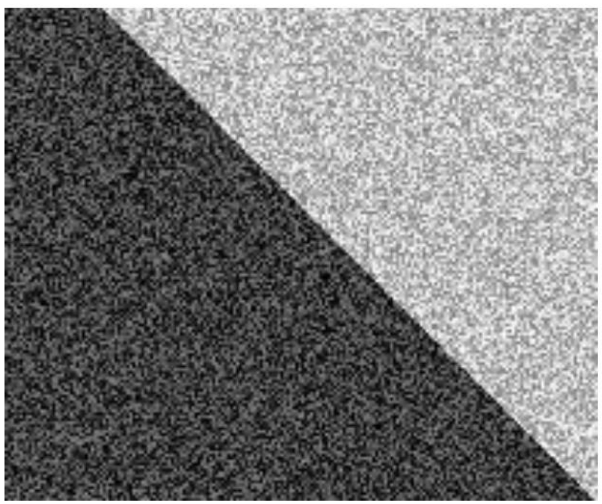
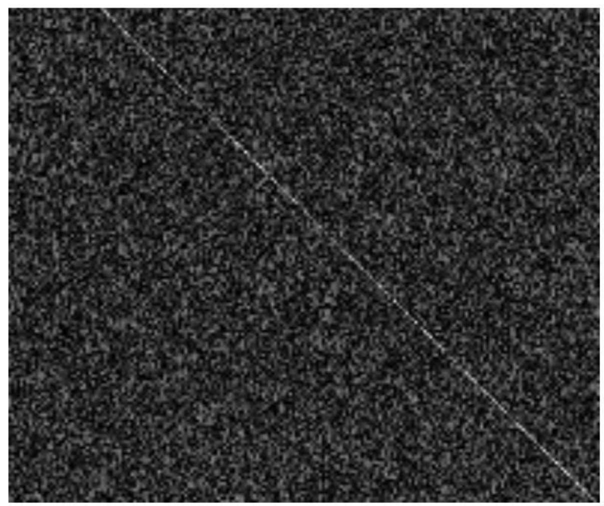
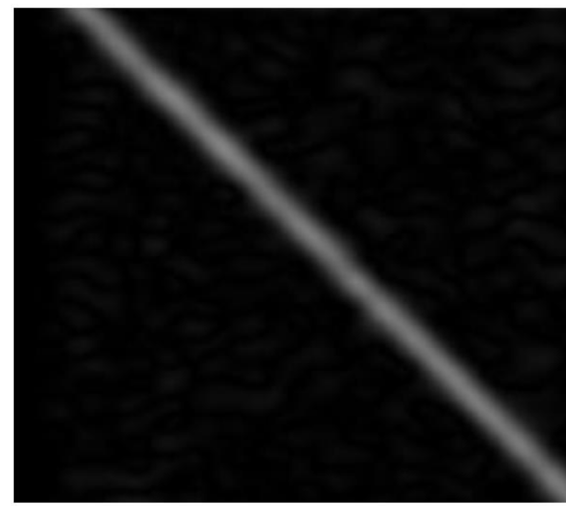


Image + Noise



Derivatives detect
edge *and* noise



Smoothed derivative removes
noise, but blurs edge

Example

- original image (Lena)

Finding edges

gradient magnitude

Finding edges

where is the edge?

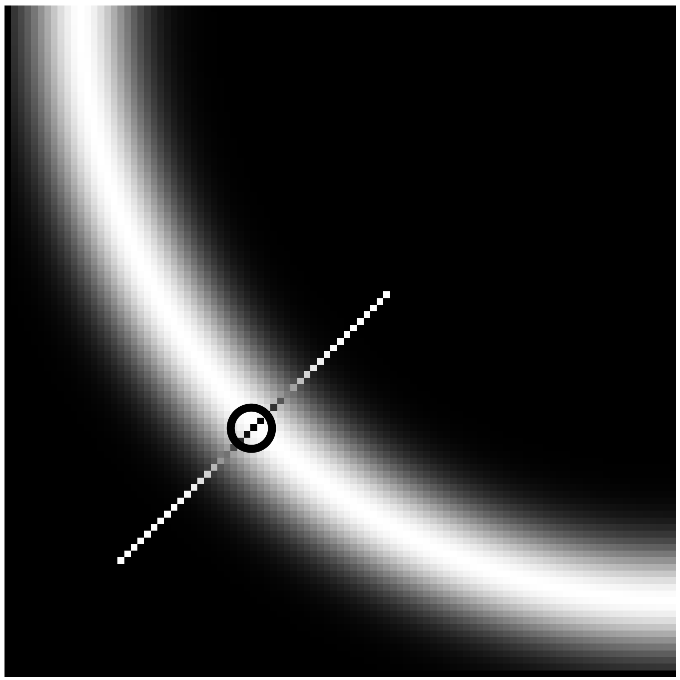
thresholding

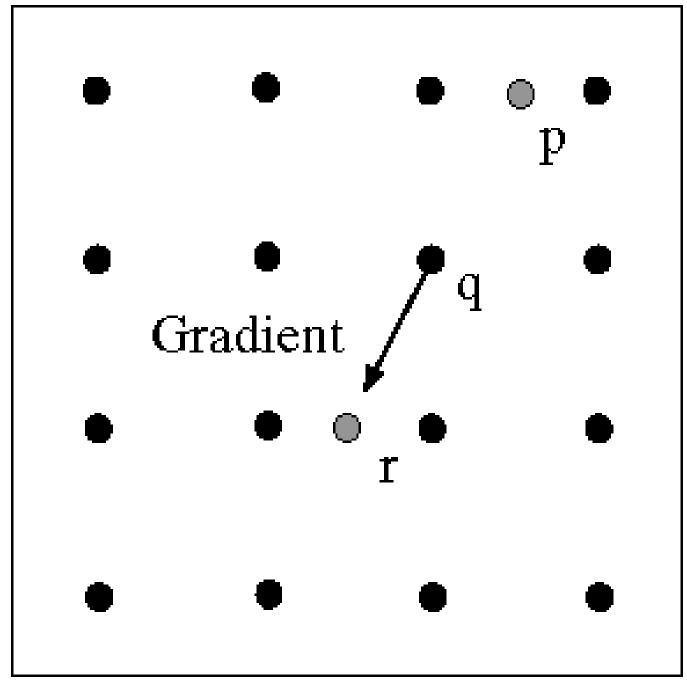
Get Orientation at Each Pixel

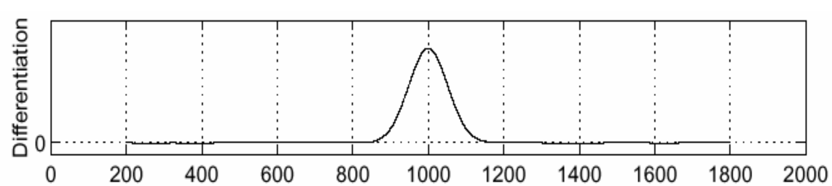
- Threshold at minimum level
- Get orientation

theta = atan2(gy, gx)

Non-maximum supression







- Check if pixel is local maximum along gradient direction
  - requires *interpolating* pixels p and r

Before Non-max Suppression

After Non-max Suppression

Finding edges

thresholding

Finding edges

thinning

(non-maximum suppression)

Canny edge detector

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

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

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

Source: D. Lowe, L. Fei-Fei

Canny edge detector

- Still one of the most widely used edge detectors in computer vision

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

- Depends on several parameters:

σ : width of the Gaussian blur

high threshold
low threshold

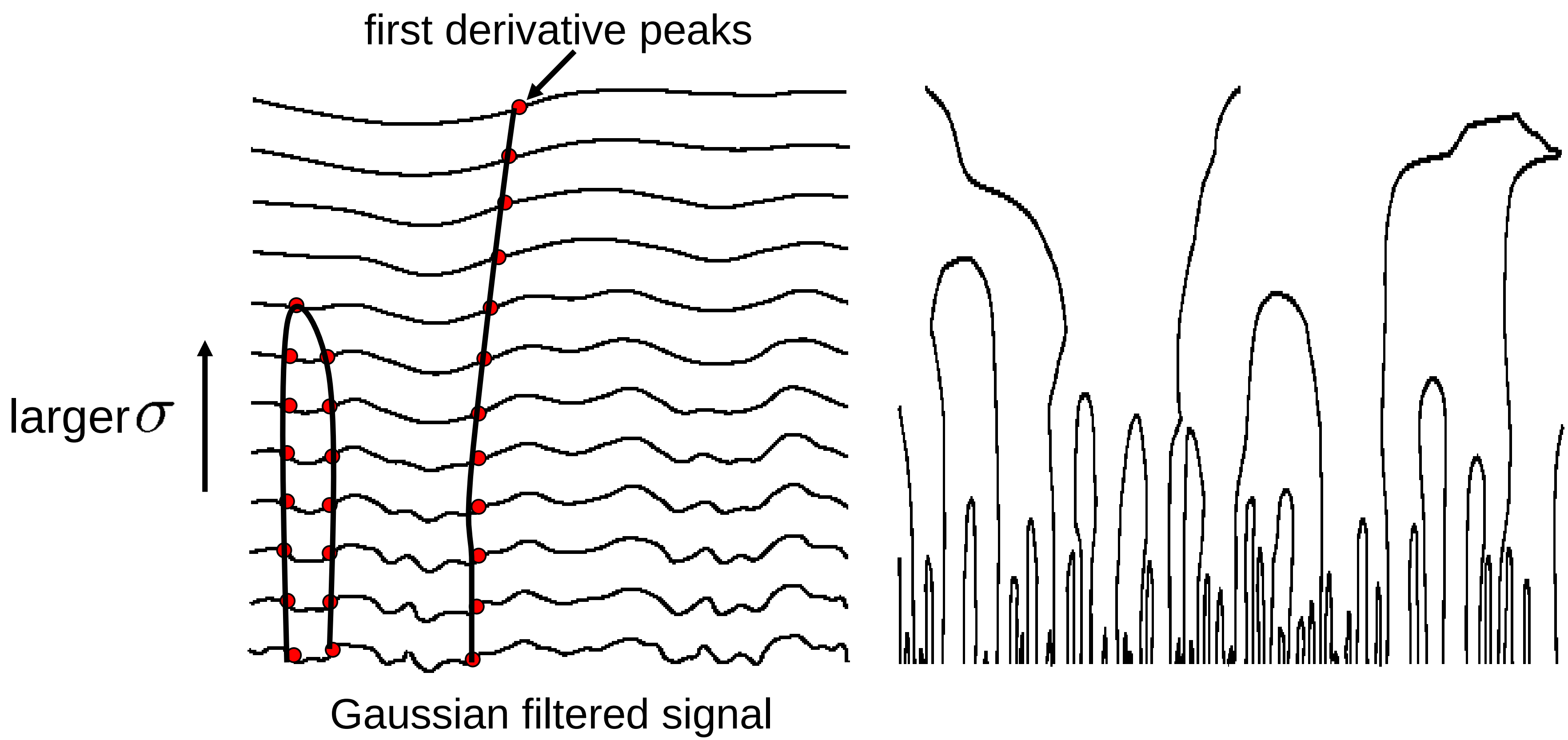
Canny edge detector

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 edges

Source: S. Seitz

Scale space (Witkin 83)



- Properties of scale space (w/ Gaussian smoothing)
  - edge position may shift with increasing scale (σ)
  - two edges may merge with increasing scale
  - an edge may ***not*** split into two with increasing scale

Questions?