Feature Extraction: Edges

Images and Information

- “Vision is based on inference”
- What makes us see objects in an image?
- At one level
- Image is a matrix of numbers
  - Somehow we extract information from these
  - Obviously we do notuse all 10^5 to 10^6 numbers
- What makes us perceive objects in images?
  - Hypothesis: process images at low level
  - Extract “features”
  - Combine features with prior knowledge to classify objects in the image at a high-level
- What features should (or do) we use?

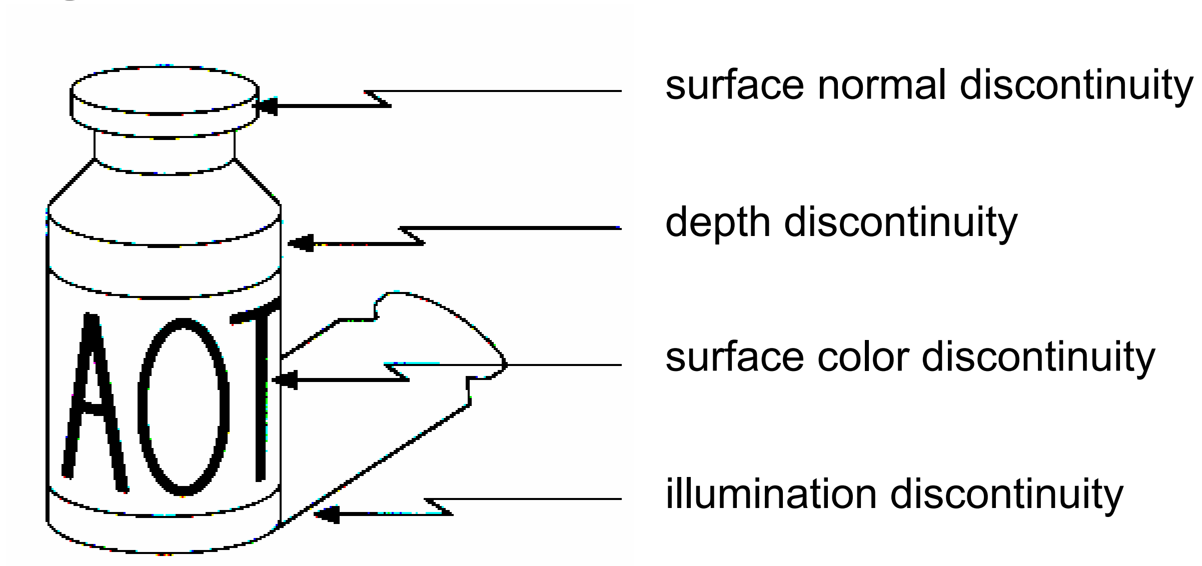
How do we decide which pieces form an object?

How can we mimic that processing on a machine?

Image Features

- Edges
- Corners and Junctions
- Texture
- Intensity gradients (distribution of ambient light)
- Correspondence of features across multiple images
  - Stereo
  - Flow
- First we will stay with features on a single image
- Goal: Define techniques that will “filter” out regions that have these features, and allow us to ignore most of the pixels.
- Use these features in higher level vision

Edges for inference


surface normal discontinuity
depth discontinuity
AOT
surface color discontinuity
illumination discontinuity

- Edges in an image have many causes
- An edge presents an opportunity to infer
- Looking at edges reduces information required
  - Look at a few pixels in a binary image as opposed to all pixels in a grayscale image
- Biological plausibility
  - initial stages of mammalian vision systems involve detection of edges and local features

Images as functions

- An image can be viewed as a function
  - For given pixel values i,j we have an intensity I_{ij}
- How can we decide from local pixel values where edges are?
- Need to recall math that studies changes of values of functions
  - Differential calculus
  - Filters and Signal processing

Derivative

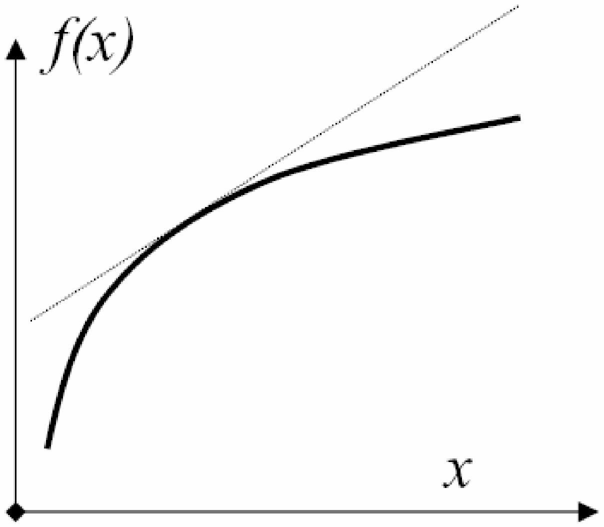
- In 1-D $\dfrac{df}{dx} = \lim_{h \to 0} \dfrac{f(x+h) - f(x)}{h}$
- Taylor series: for a continuous function

$$f(x+h) = f(x) + h \left.\frac{df}{dx}\right|_x + \frac{h^2}{2} \left.\frac{d^2 f}{dx^2}\right|_x + \cdots + \frac{h^n}{n!} \left.\frac{d^n f}{dx^n}\right|_x + \cdots$$

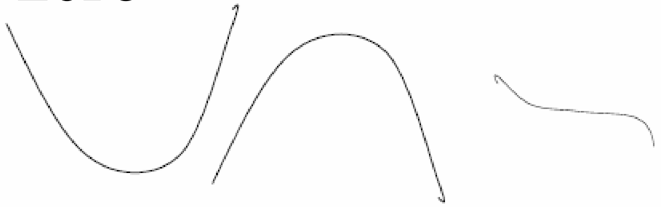
$$f(x-h) = f(x) - h \left.\frac{df}{dx}\right|_x + \frac{h^2}{2} \left.\frac{d^2 f}{dx^2}\right|_x + \cdots + (-1)^n \frac{h^n}{n!} \left.\frac{d^n f}{dx^n}\right|_x + \cdots$$

•Geometric interpretation

–Approximate smooth curve by values of tangent, curvature, etc.



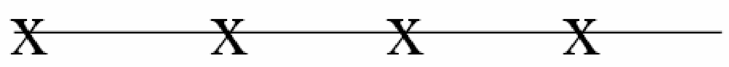
- Derivative measures rate of change
  - High when there is a sudden change
  - When there is no change it is zero



- *df/dx=0*
  - represents a minimum, maximum or saddle point of the curve *y=f(x)*
  - $d^2f/dx^2 > 0$ minimum, $d^2f/dx^2 < 0$ maximum
  - $d^2f/dx^2 = 0$ saddle point

- Second derivative measures rate of change of the rate of change
  - Zero at a maximum or a minimum of the function

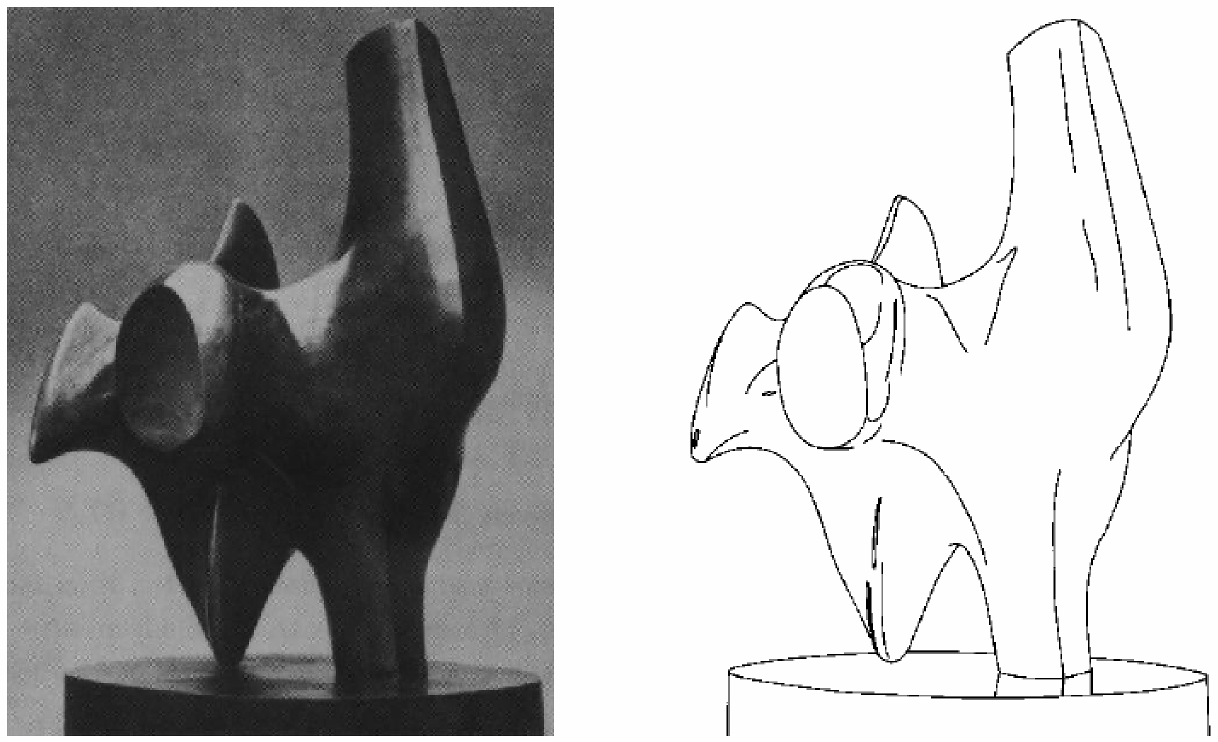
Finite Differences

- Images are digital and quantized
  - Derivatives are defined for continuous functions
  - Need to approximate derivative on images
- Truncate Taylor series and obtain an expression for the derivatives
- Forward differences: use value at the point and forward 
- Backward differences

$$\left.\frac{df}{dx}\right|_x = h^{-1}\left(f(x+h) - f(x)\right) - \frac{h}{2}\left.\frac{d^2 f}{dx^2}\right|_x + O\left(h^2\right)$$

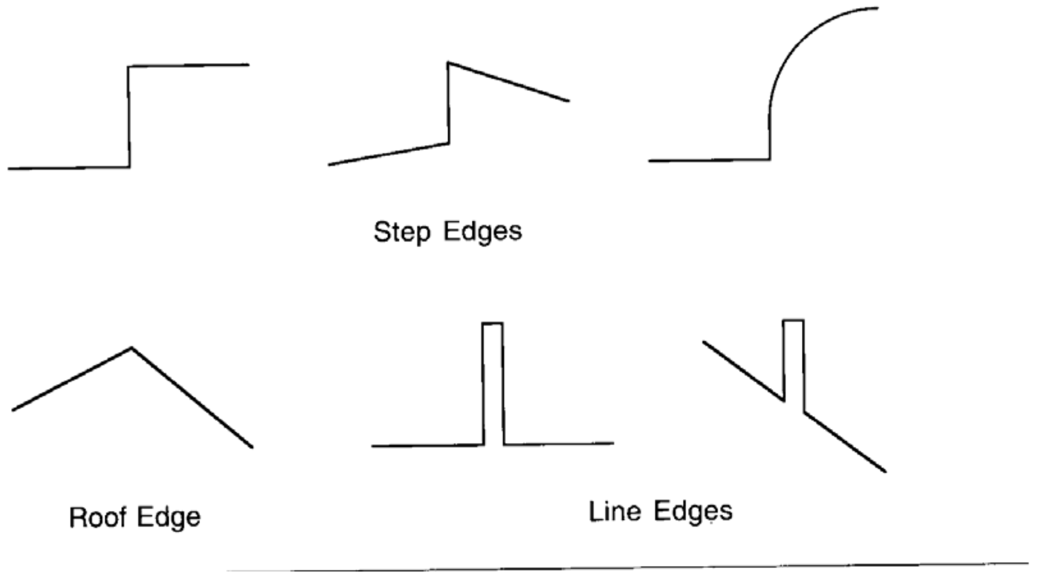
$$\left.\frac{df}{dx}\right|_x = h^{-1}\left(f(x) - f(x-h)\right) + \frac{h}{2}\left.\frac{d^2 f}{dx^2}\right|_x + O\left(h^2\right)$$

Edge detection



- How can you tell that a pixel is on an edge?

Profiles of image intensity edges



Edge detection

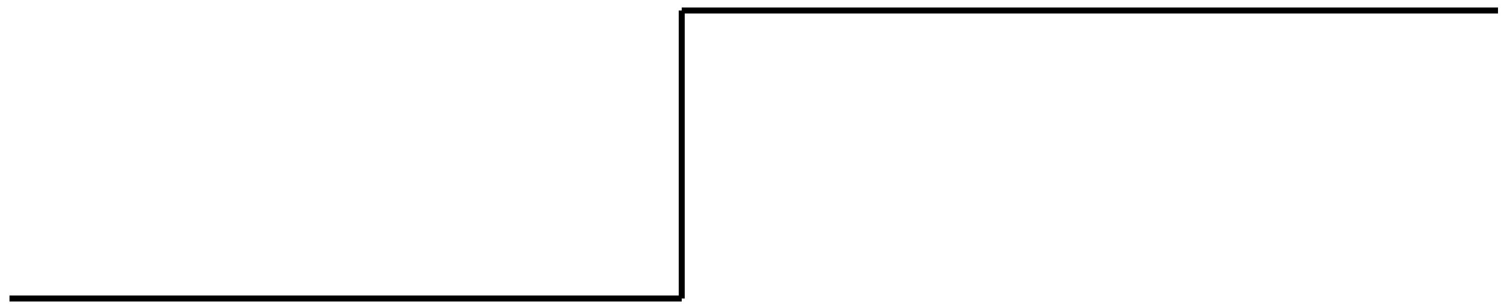
1. Detection of short linear edge segments (edgels)
2. Aggregation of edgels into extended edges
3. Possibly combine the edges

Edgel detection

- Difference operators
- Parametric-model matchers

Edge is Where Change Occurs

- Change is measured by derivative in 1D
- Biggest change, derivative has maximum magnitude
- Or 2nd derivative is zero.



Two and more dimensions

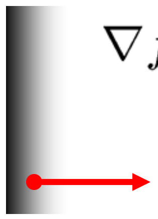
- Gradient $\nabla f = \frac{\partial f}{\partial x}\mathbf{e}_1 + \frac{\partial f}{\partial y}\mathbf{e}_2 = \frac{\partial f}{\partial x_i}\mathbf{e}_i$
- Directional derivative in the direction of a vector **n** $\nabla f \bullet \mathbf{n} = \frac{\partial f}{\partial x}\mathbf{e}_1 \bullet \mathbf{n} + \frac{\partial f}{\partial y}\mathbf{e}_2 \bullet \mathbf{n} = \frac{\partial f}{\partial x_i} n_i$

- Taylor series

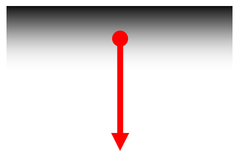
$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \mathbf{h} \bullet \nabla f(\mathbf{x}) + O(h^2)$$

Image gradient

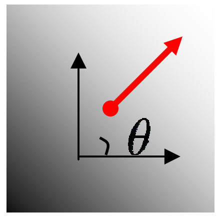
- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$
- The gradient points in the direction of most rapid change in intensity



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



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



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

- The gradient direction is given by:

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

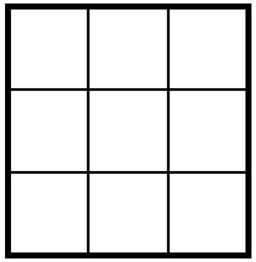
- how does this relate to the direction of the edge?
- The *edge strength* is given by the gradient magnitude $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

The discrete gradient

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

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

H



The Sobel operator

- Better approximations of the derivatives exist
  - The *Sobel* operators below are very commonly used

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

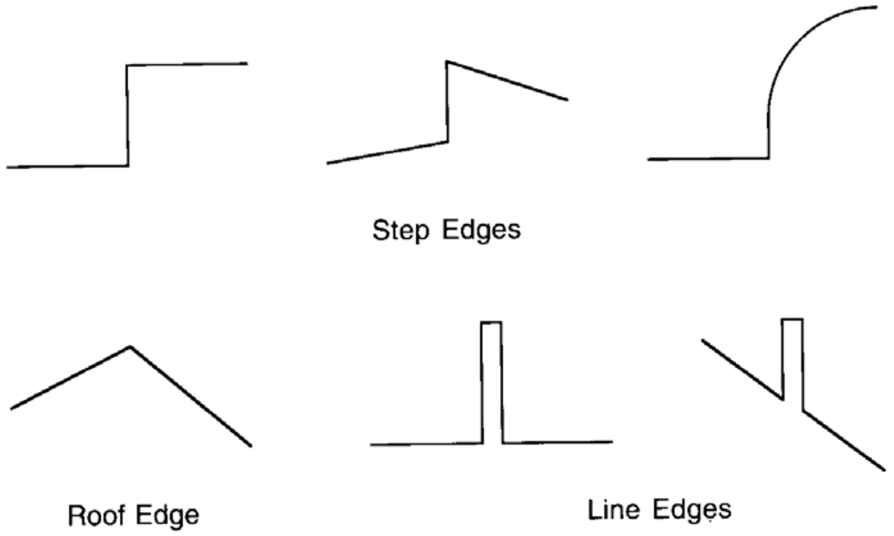
$$s_x \qquad\qquad\qquad s_y$$

  - 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 value, however

Threshholding and filtering

- Applying a “Sobel operator” to an image
  - using the Sobel discrete operator, loop over all non-boundary pixels,
  - compute the derivatives in the *x* and *y* directions
  - Compute magnitude and direction of edges
  - If (magnitude > threshhold)
    - put a 1 in the edge-image
  - Else
    - Put a zero
- Resulting image is a “binary” edge image

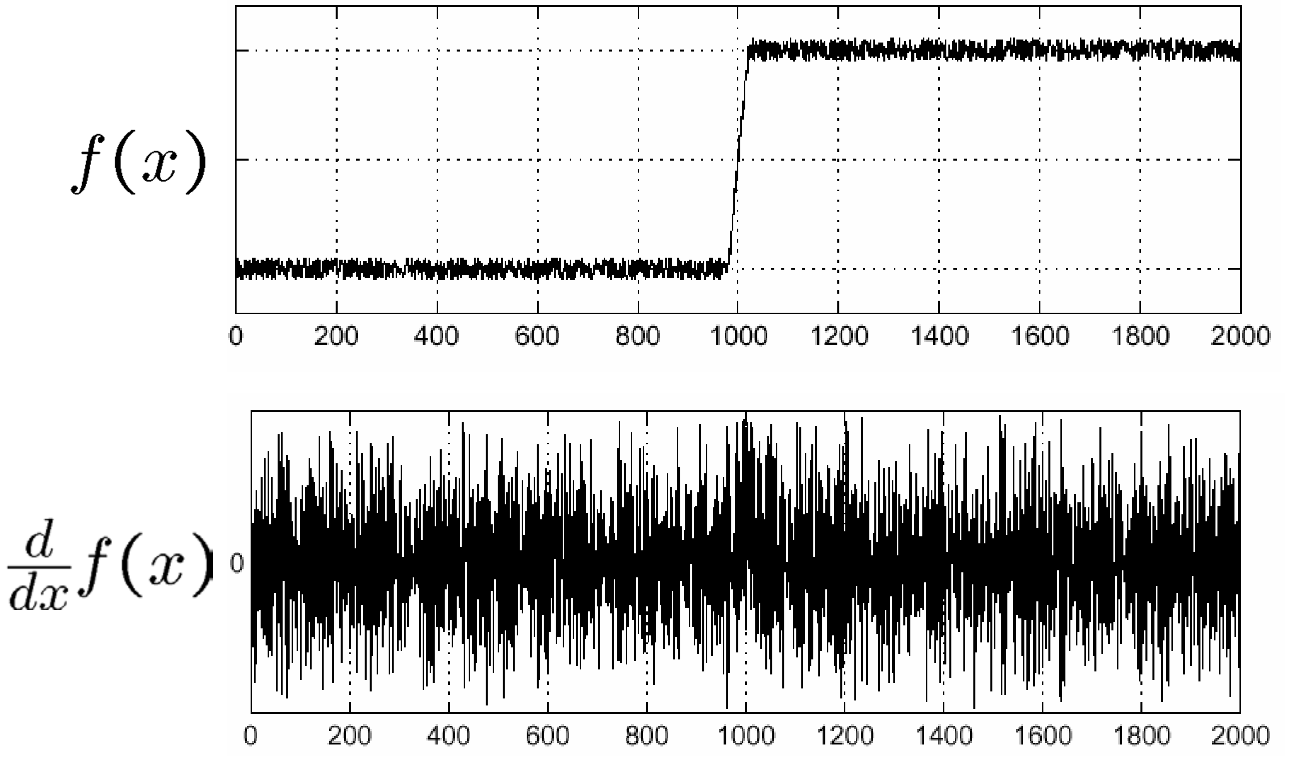
Non-maxima suppression



- Note that edges lie in the midst of gentler gradients
- So the Sobel operator will return non-zero values even in regions where there is no edge
- Threshholding can minimize “false-postives”, and only pick local maxima
- However it cannot account for noise

Effects of noise

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



- Why does this happen and how can we find the edge?
- Recall derivative definition

Noisy derivatives

- $\Delta f / \Delta x$
- Could become large due to a big change in *f* across a larger region
- Could become large due to a small change in *f* across a small region
- Noise: typically occurs across small scales
- Derivatives of noisy functions are hard to compute
- Somehow must remove these small scale noise-effects: smoothing

Smoothing - basic problems

- What function should be used to smooth or average the image before differentiation?
  - box filters or uniform smoothing
    - easy to compute
    - for large smoothing neighborhoods assigns too much weight to points far from an edge
  - Gaussian, or exponential, smoothing

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

Filters and Convolution

- box-smoother: In 1-D take an array h of length N say 5 pixels. Set values of filter coefficients to 1/5
- To compute smoothed image (running average)
- Apply filter to each pixel point and compute a new image
- This is called convolution of the filter with the signal or image.
- Convolution is indicated with a *
- $I=h*f$

Linear Filters

- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)
- Example: smoothing by averaging
  - form the average of pixels in a neighbourhood
- Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighbourhood
- Example: finding a derivative
  - form a weighted average of pixels in a neighbourhood

Convolution

- Represent these weights as an image, H
- H is usually called the **kernel**
- Operation is called **convolution**
  - it's associative
- Result is: $$R_{ij} = \sum_{u,v} H_{i-u,j-v} F_{uv}$$
- Notice weird order of indices
  - all examples can be put in this form
  - it's a result of the derivation expressing any shift-invariant linear operator as a convolution.

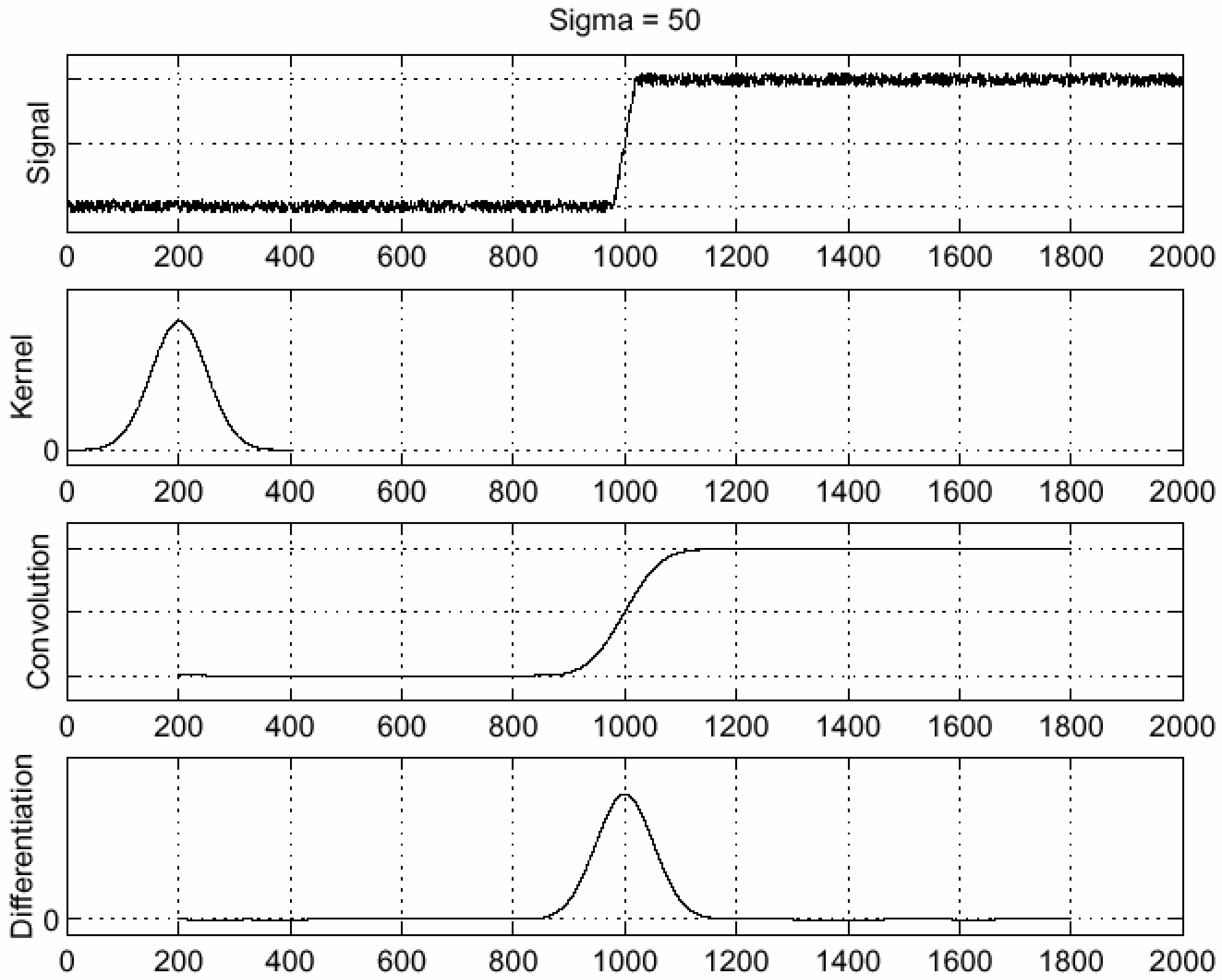
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



- Where is the edge?
- Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

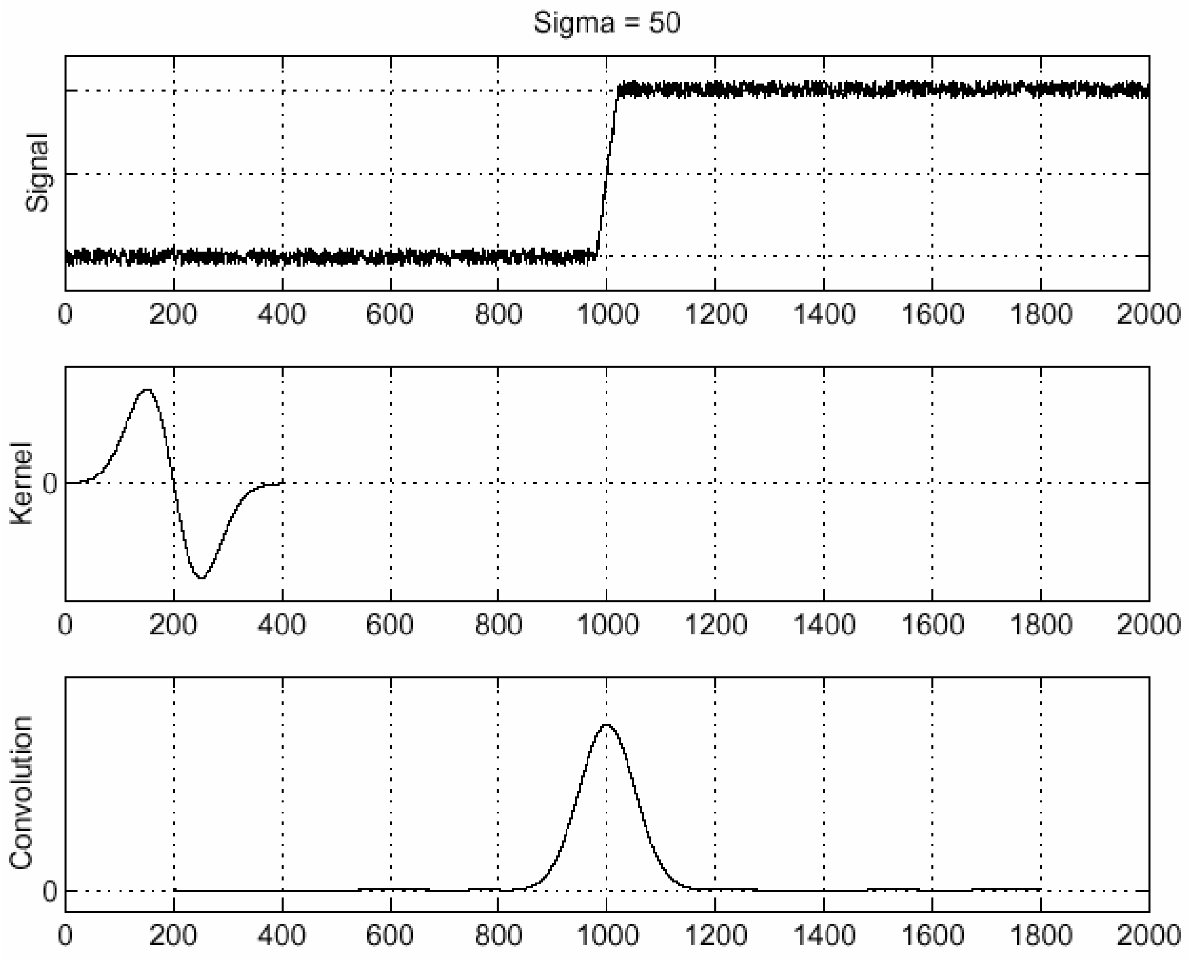
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

- This saves us one operation:

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



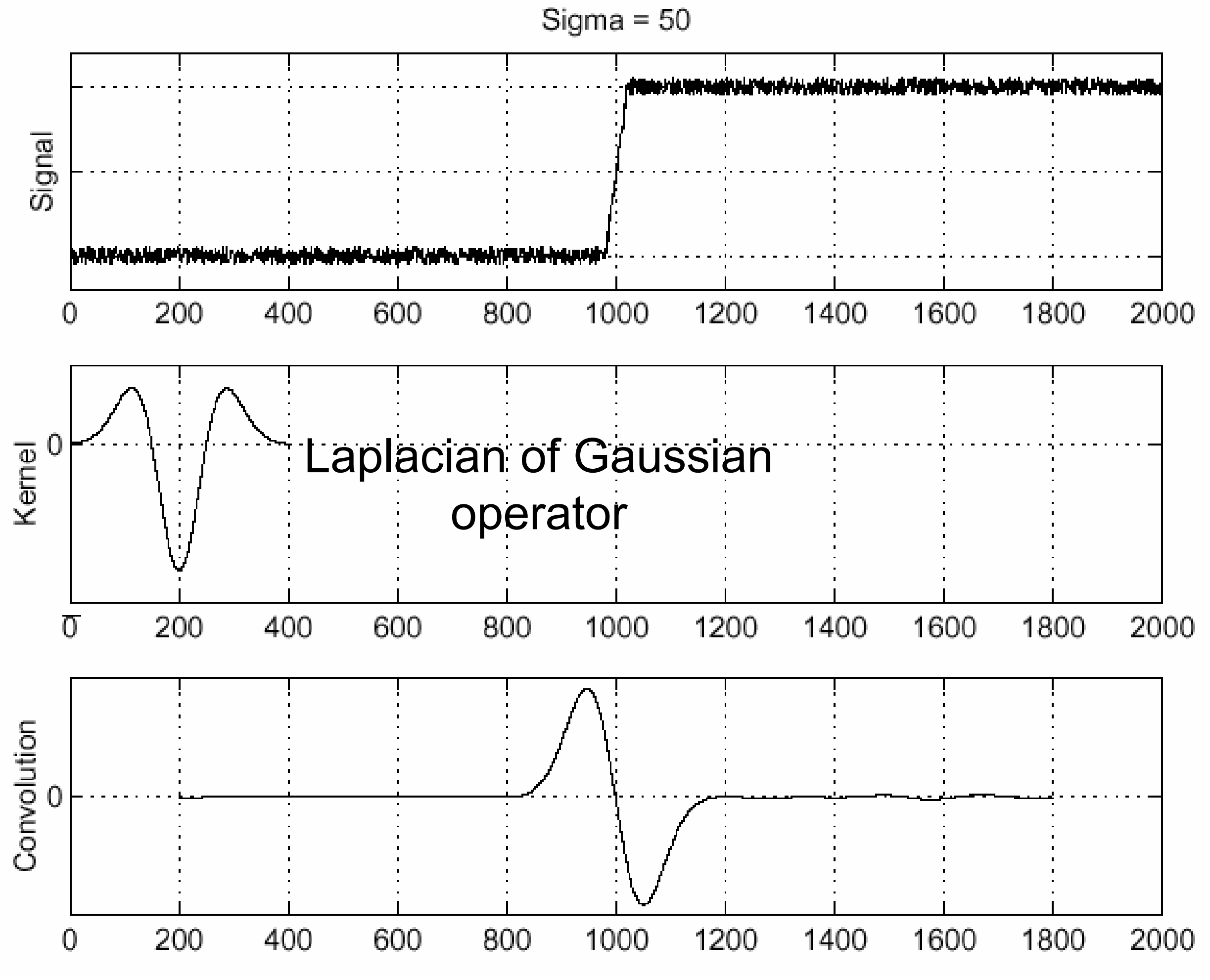
Laplacian of Gaussian

- Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

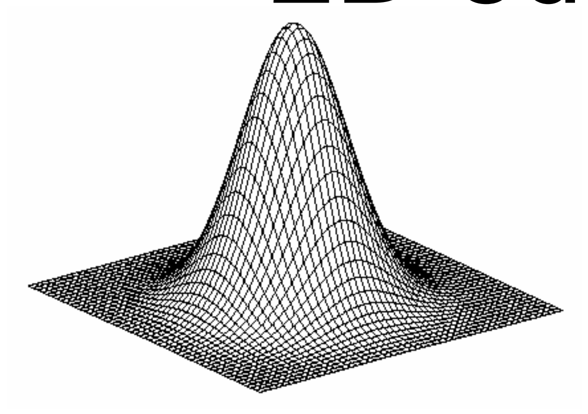
$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$



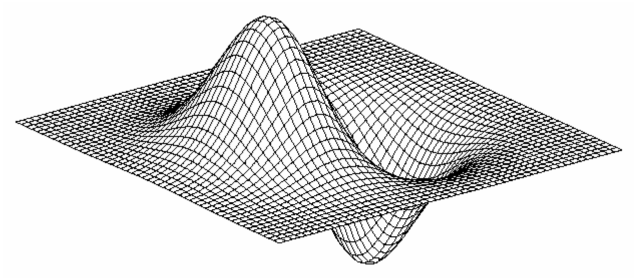
- Where is the edge?
- Zero-crossings of bottom graph

2D edge detection filters



Gaussian

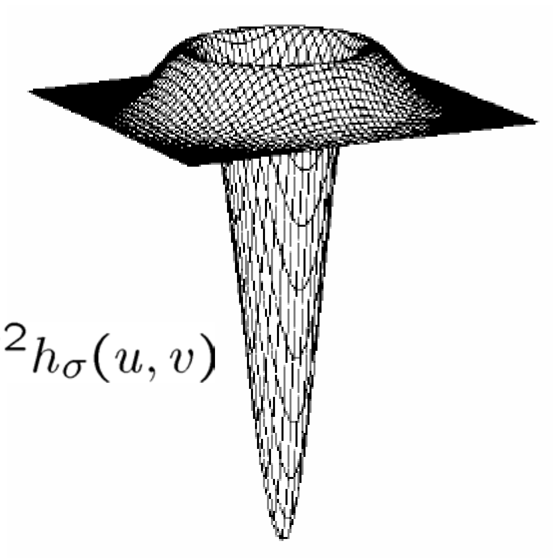
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

- ∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

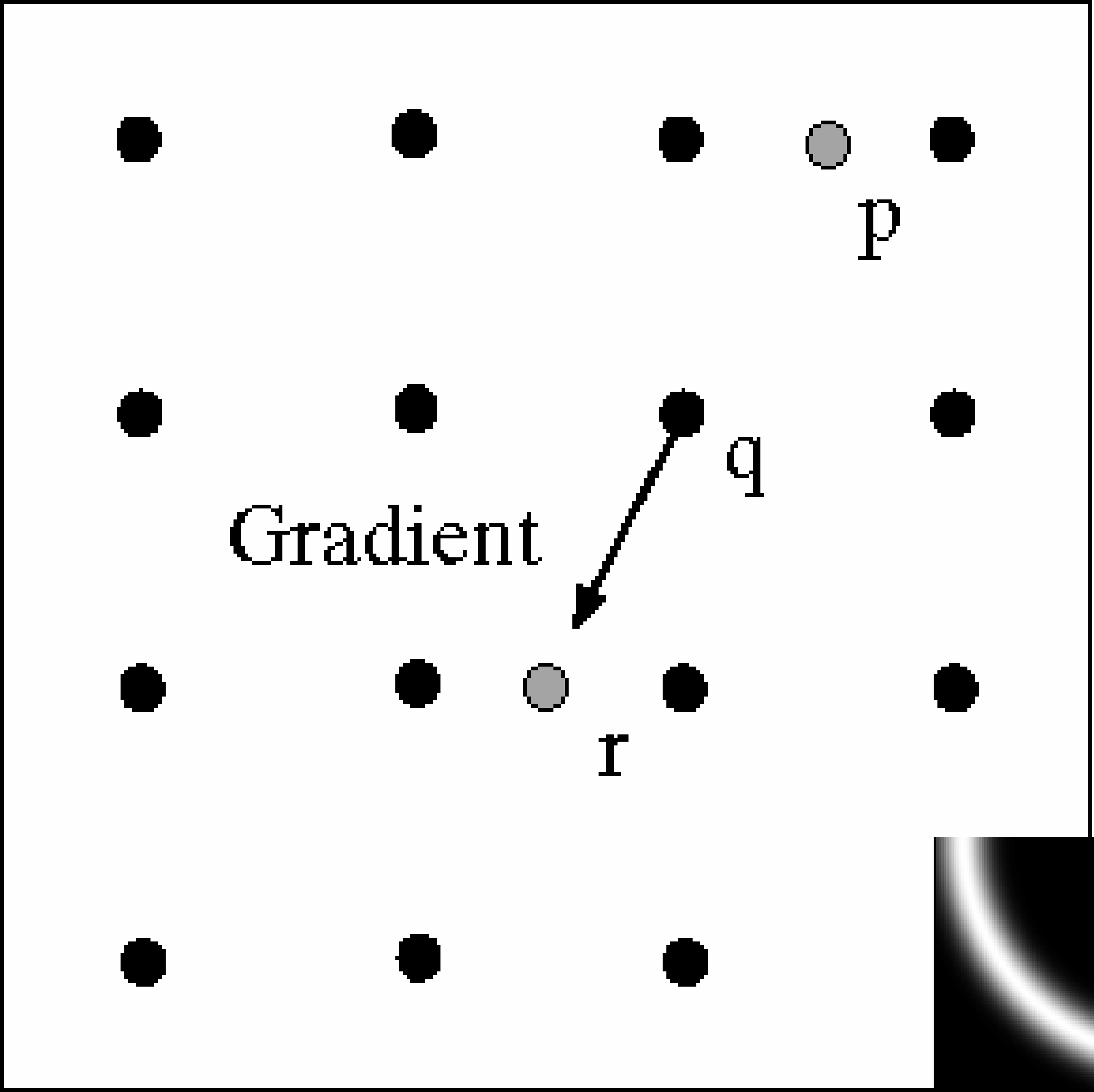
- Direction of the edge:
  - Edge has maximal change in direction normal to it.

Optimal Edge Detection: Canny

- Assume:
  - Linear filtering
  - Additive i.i.d Gaussian noise
- Edge detector should have:
  - Good Detection. Filter responds to edge, not noise.
  - Good Localization: detected edge near true edge.
  - Single Response: one per edge.

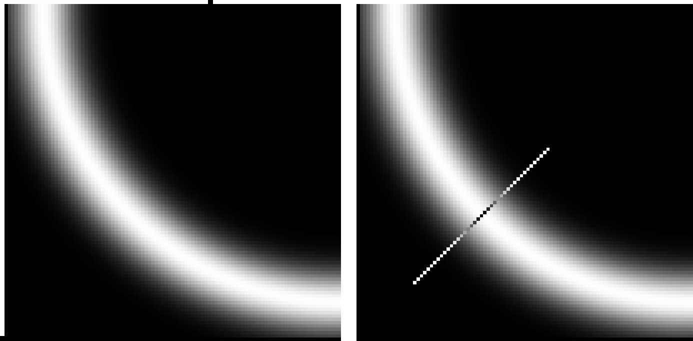
Optimal Edge Detection: Canny (continued)

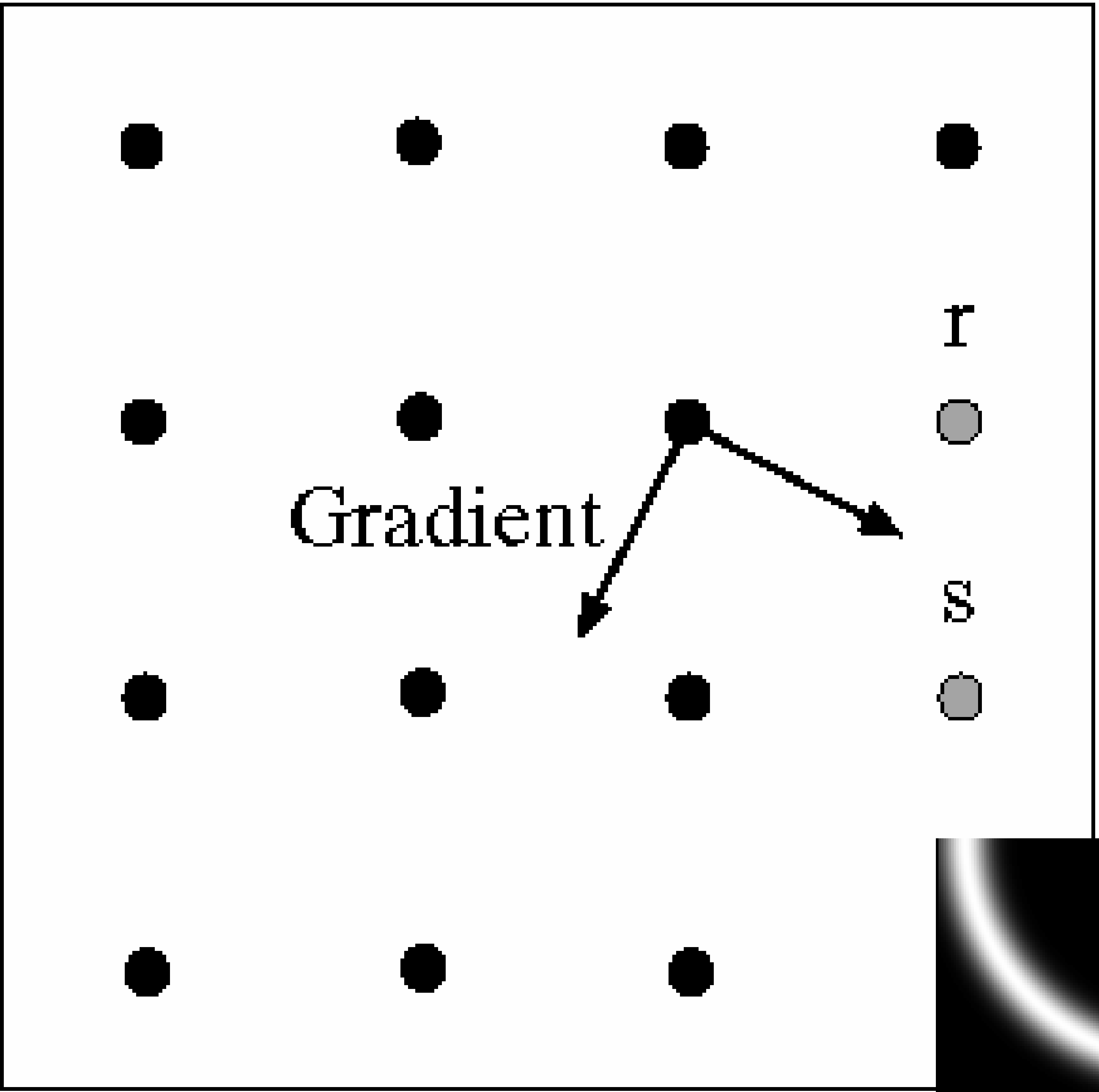
- Optimal Detector is approximately Derivative of Gaussian.
- Detection/Localization trade-off
  - More smoothing improves detection
  - But hurts localization.



Non-maximum suppression

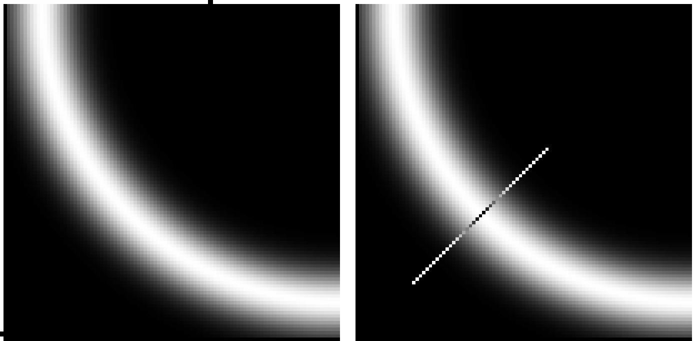
At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.





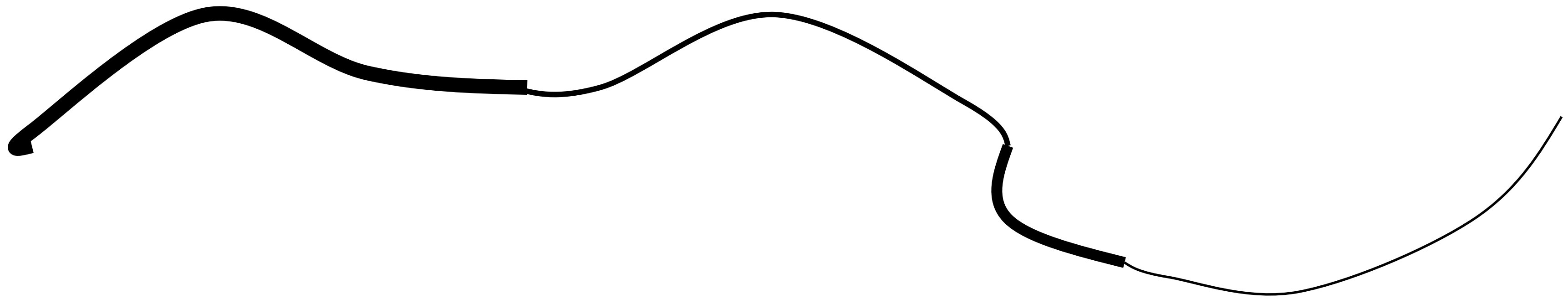
Predicting the next edge point

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).



Remaining issues

- 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.



Why is Canny so Dominant

- Still widely used after 20 years.

1. Theory is nice (but end result same).
2. Details good (magnitude of gradient).
3. Hysteresis an important heuristic.
4. Code was distributed.
5. Perhaps this is about all you can do with linear filtering.