Edge Detection

René Magritte, “Decalcomania”

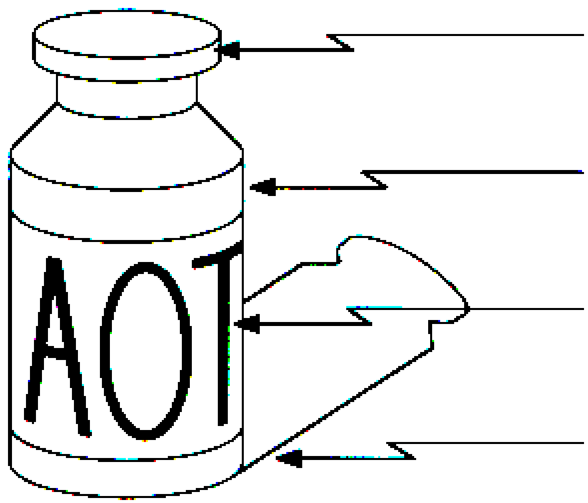
Many slides from
Derek Hoiem, Robert Collins

CSC320: Introduction to Visual Computing
Michael Guerzhoy

Discontinuities in Intensity

Source: Robert Collins

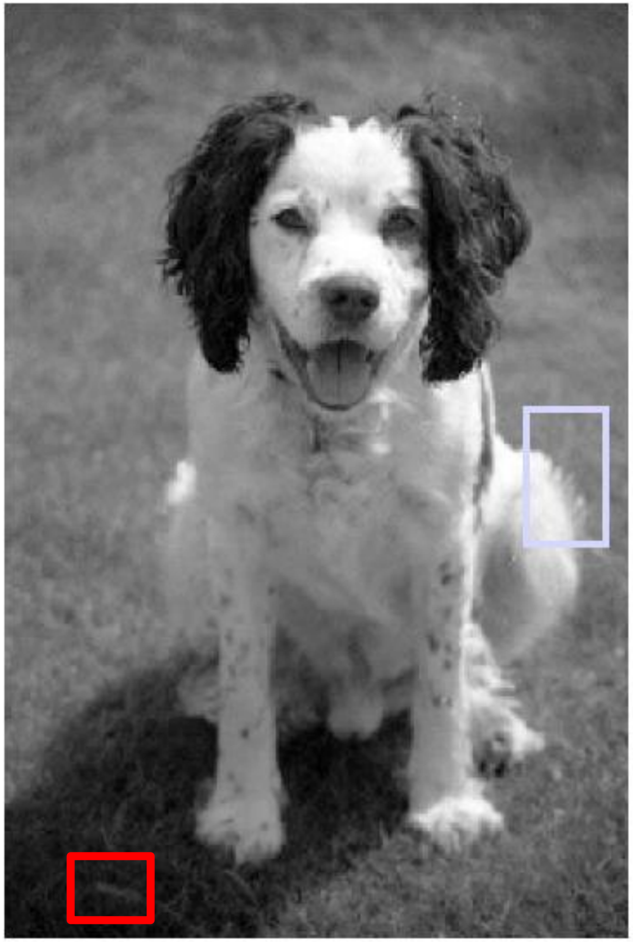
Origin of Edges

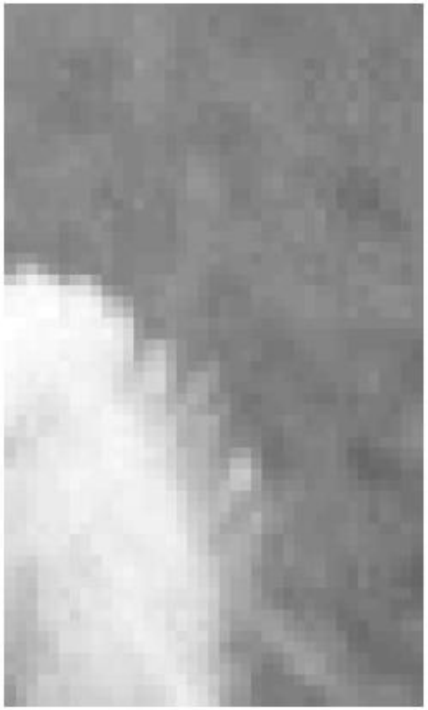


- Edges are caused by a variety of factors

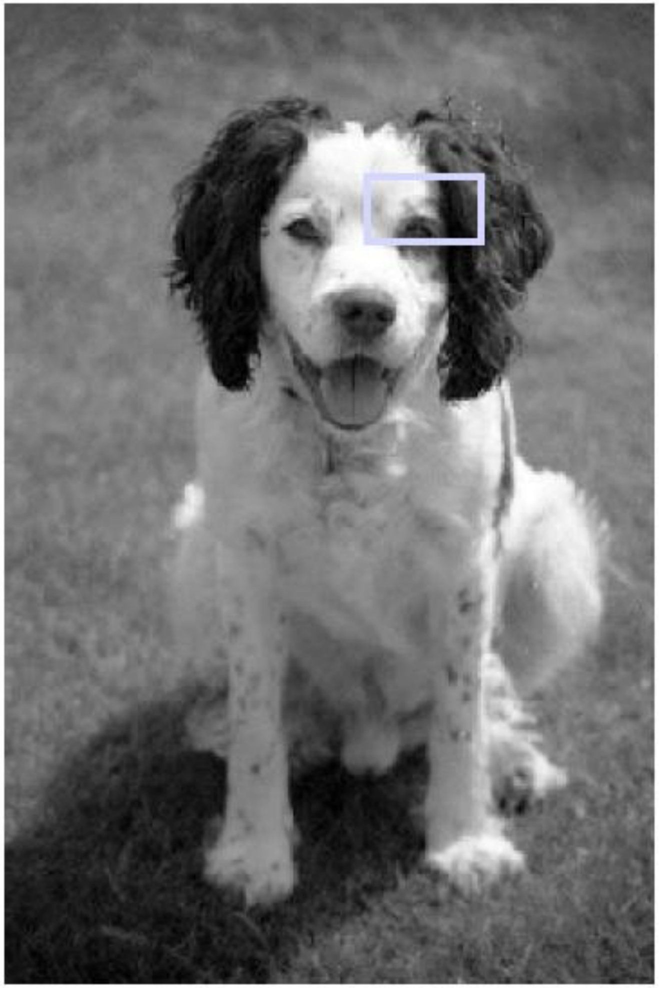
Source: Steve Seitz

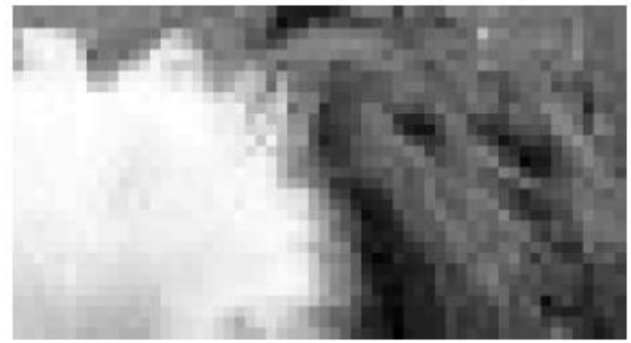
Boundaries of objects





Boundaries of material properties

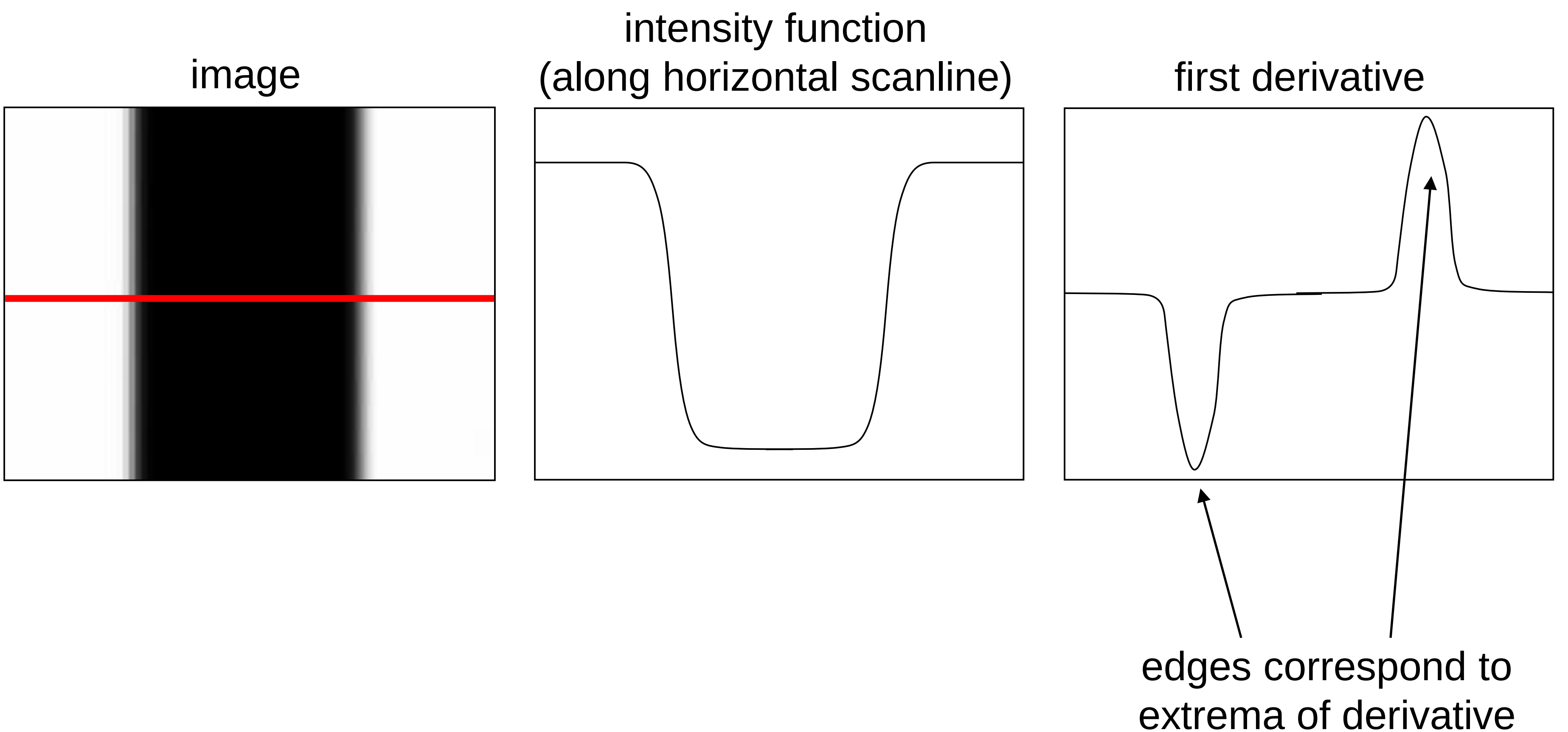




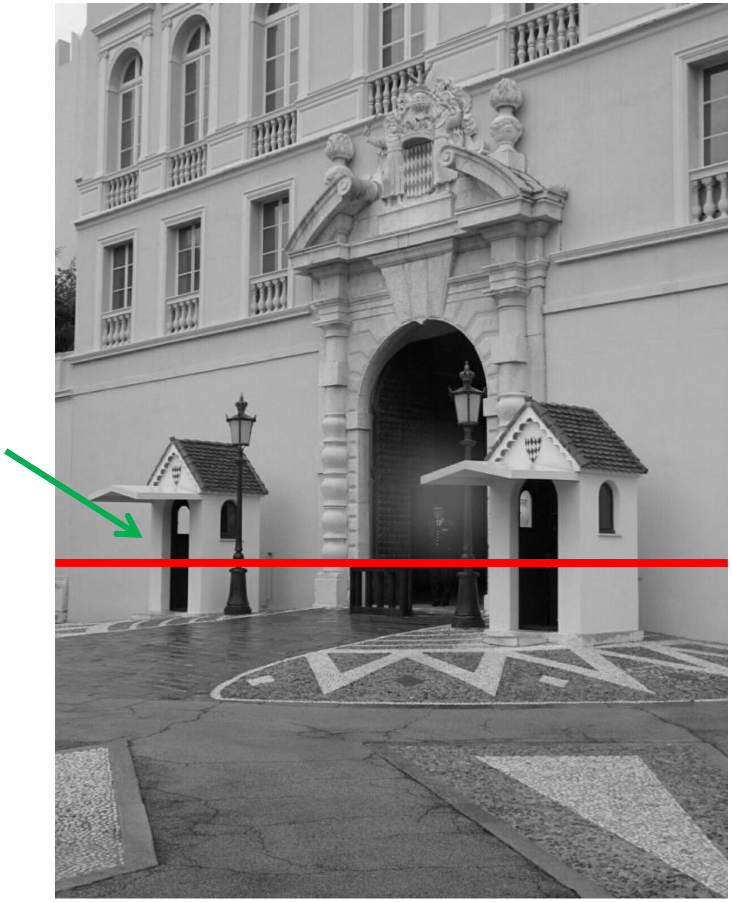
Boundaries of Lighting

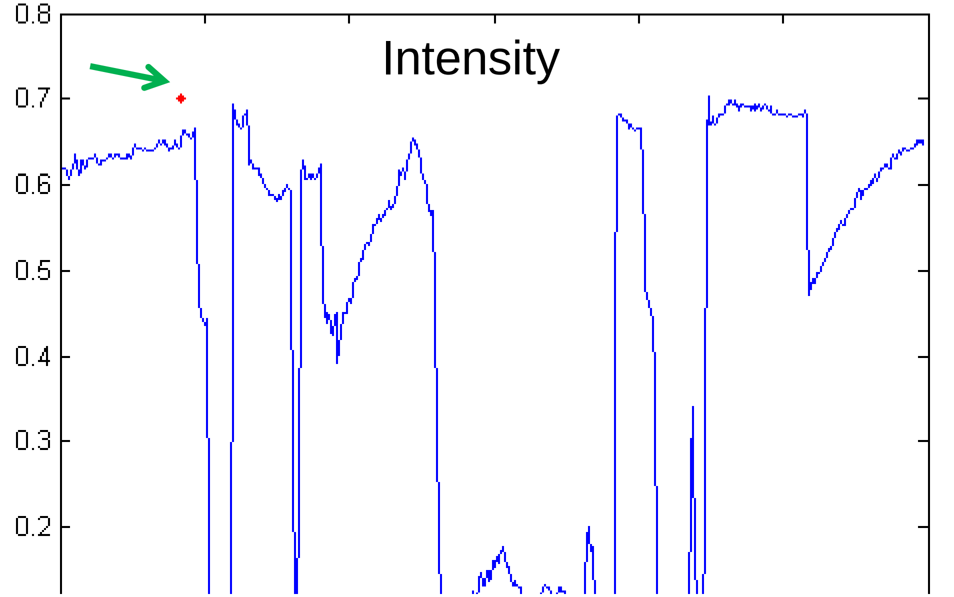
Characterizing edges

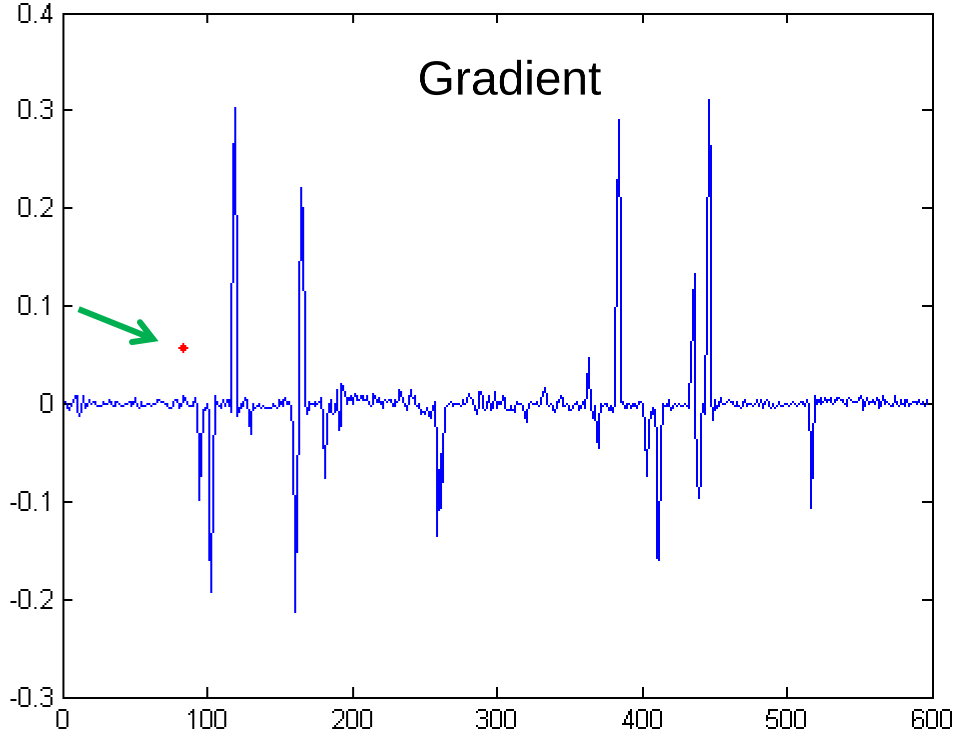
- An edge is a place of rapid change in the image intensity function



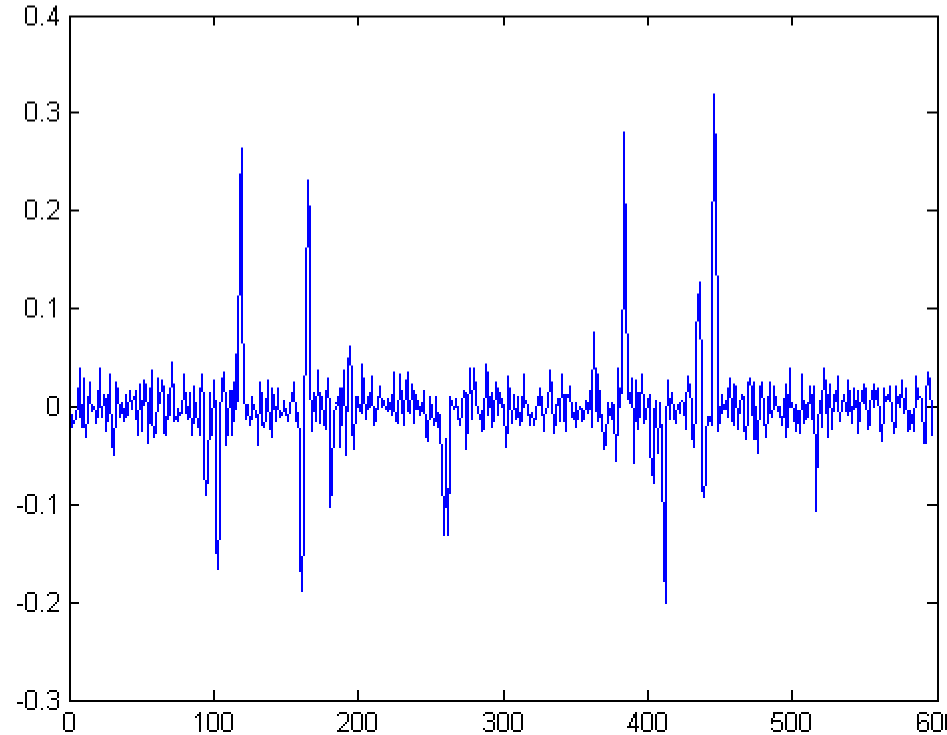
Intensity profile







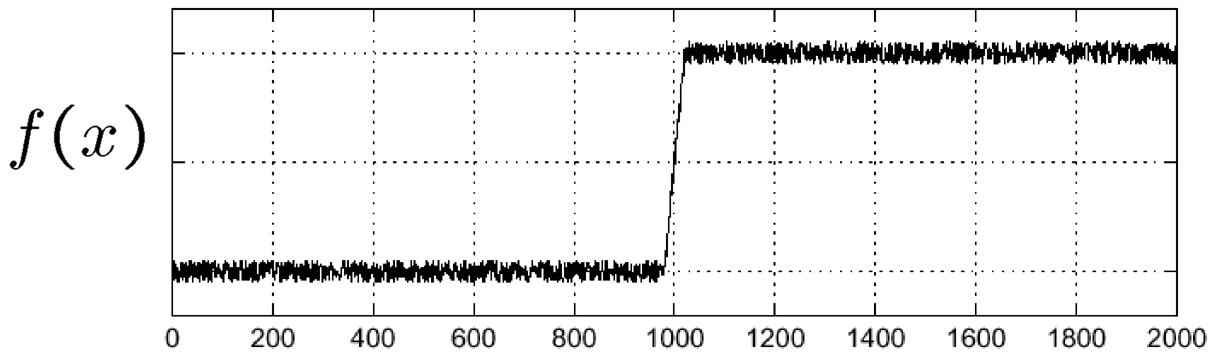
With a little Gaussian noise

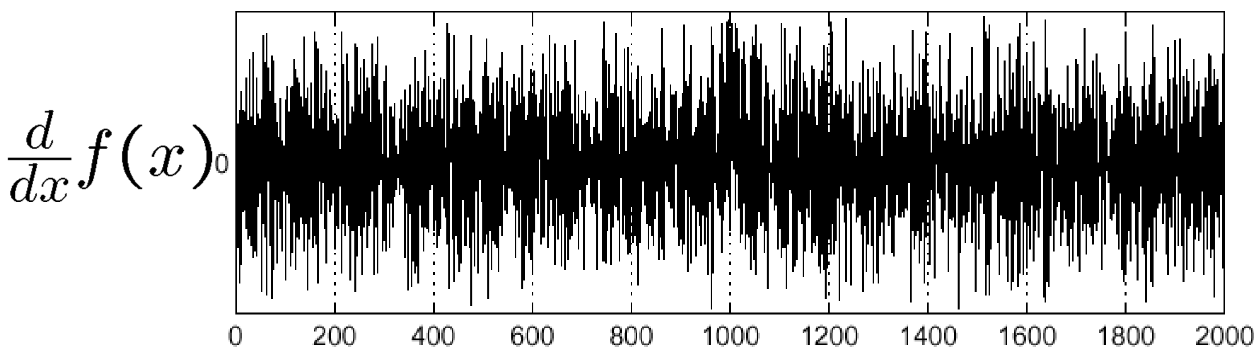


Gradient

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?

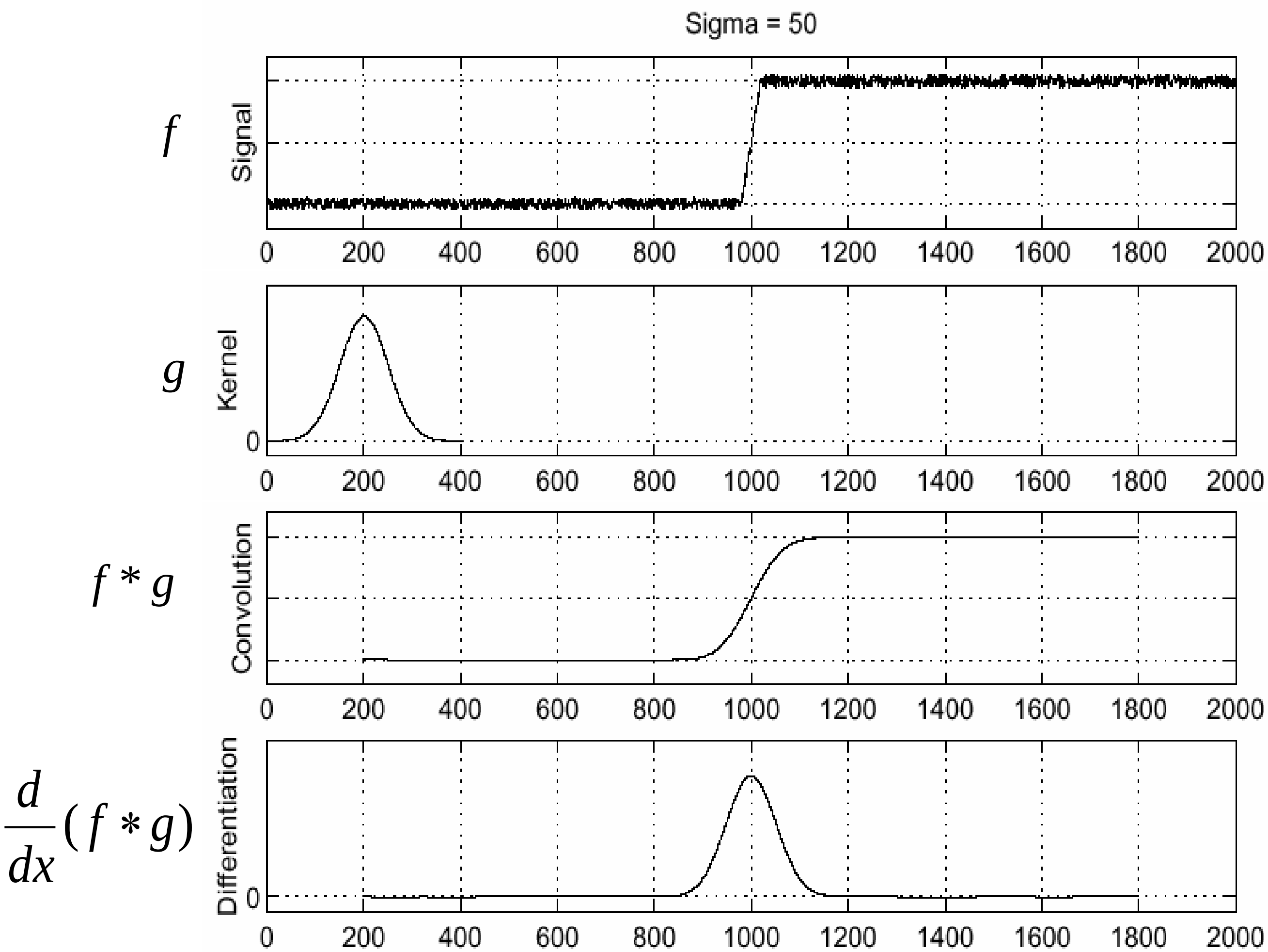
Source: S. Seitz

Effects of noise

- 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 can we do about it?

Source: D. Forsyth

Solution: smooth first



- 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:

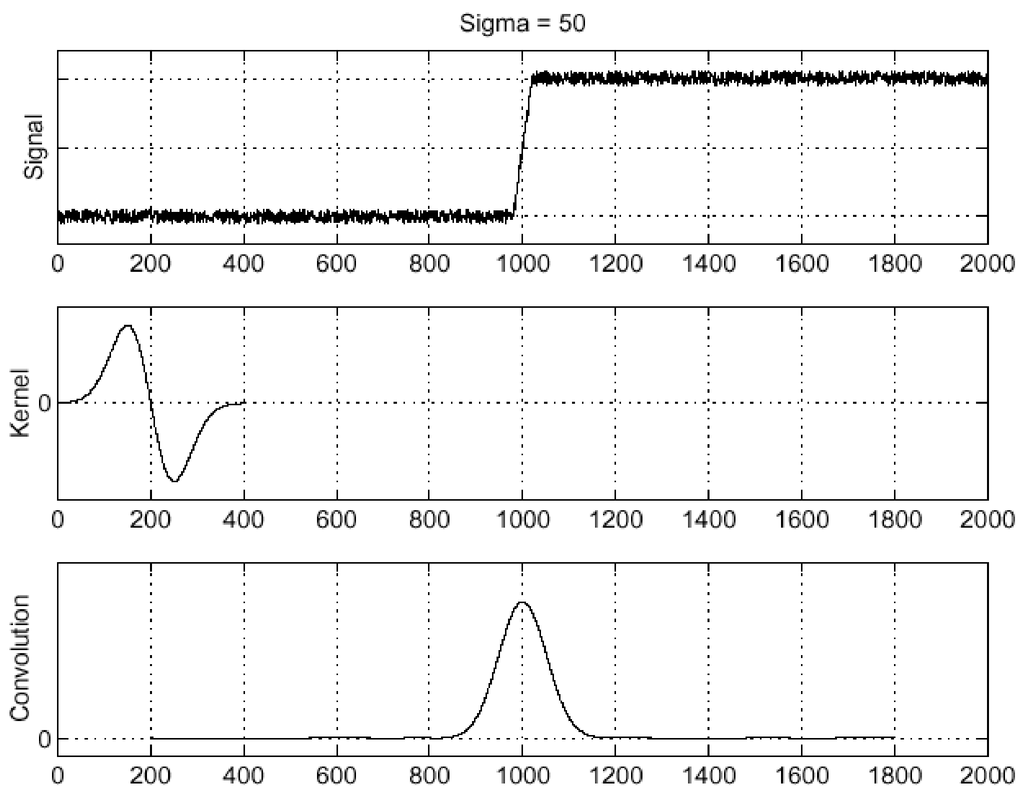
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

f

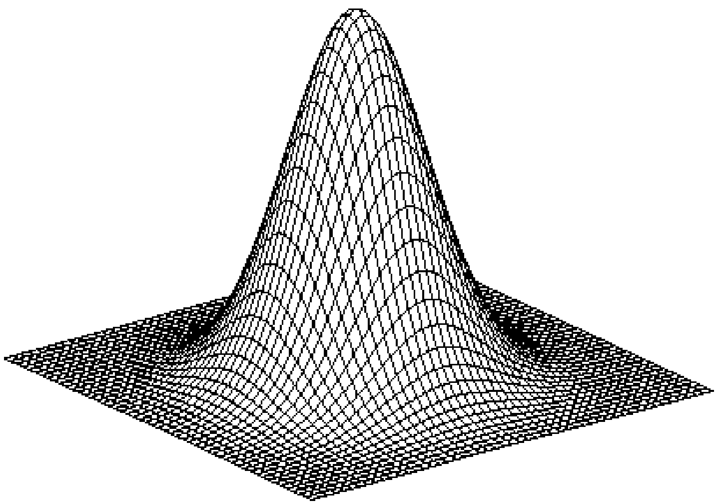
$\frac{d}{dx} g$

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

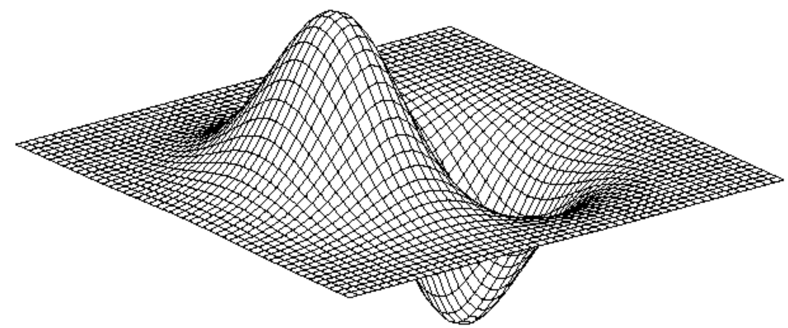


Source: S. Seitz

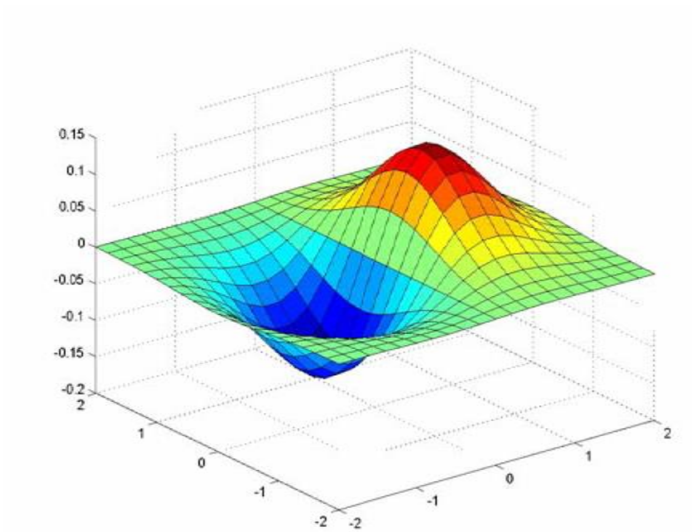
Derivative of Gaussian filter



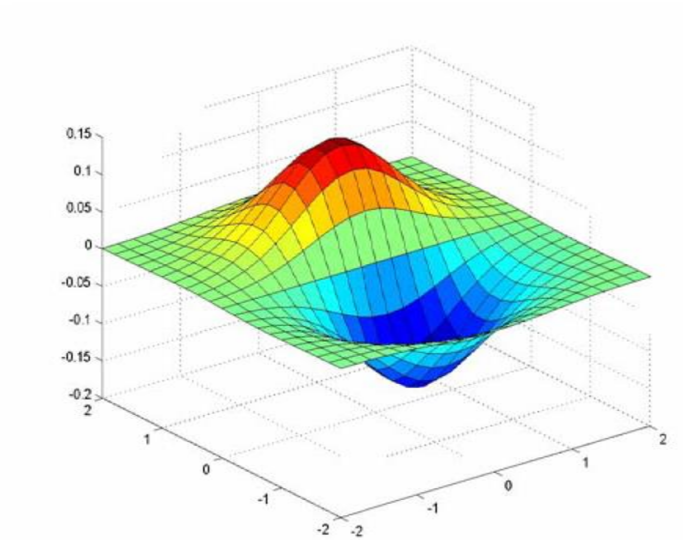
* [1 0 -1] =



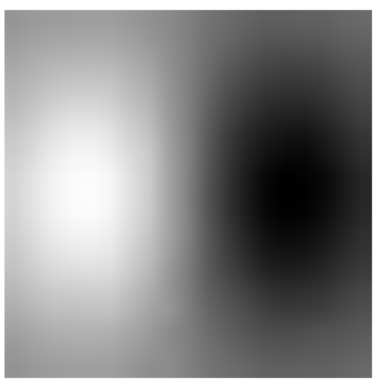
Derivative of Gaussian filter

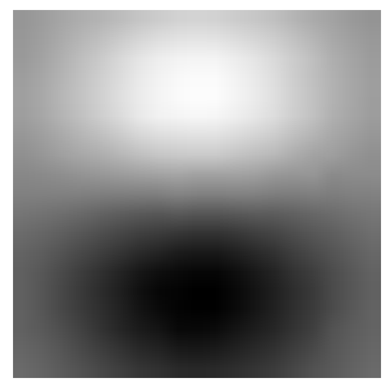


x-direction



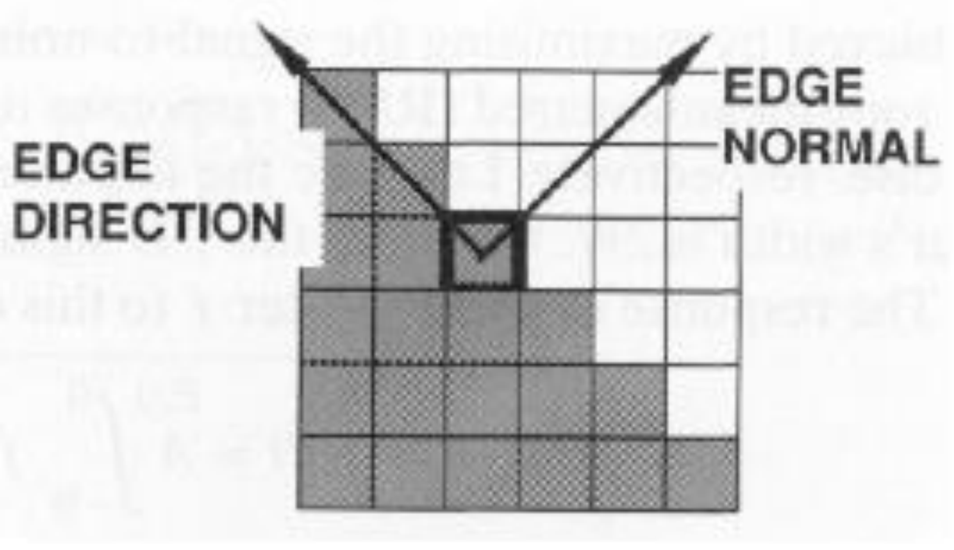
y-direction





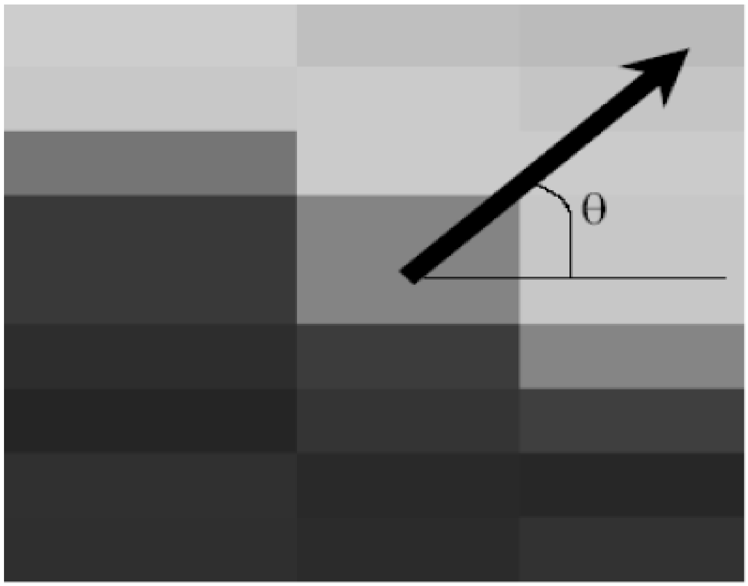
Terminology

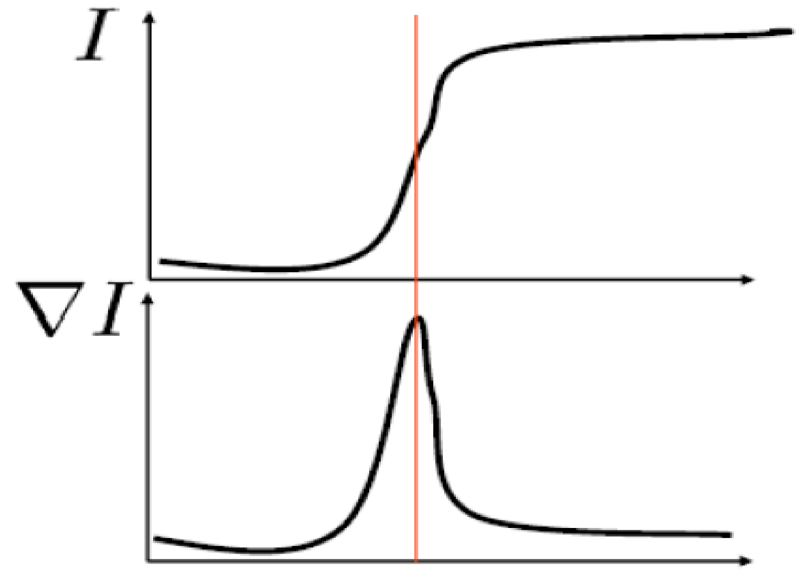
- Edge normal: unit vector in the direction of maximum intensity change.
- Edge direction: unit vector along edge (perpendicular to edge normal).
- Edge position or center: the image position at which the edge is located.
- Edge strength or magnitude: local image contrast along the normal.



If the edge direction is **(u, v)**, the edge normal is **(-v, u)** (or **(v, -u)**
Local image contrast at (x, y) along (u, -v): something like
|I(x+u, x-v)-I(x-u, x+v)|.

Gradients (derivatives) of 2D images





Edge pixels are at local maxima of gradient magnitude
Gradient direction is always perpendicular to edge direction

Gradient Vector: $\nabla I = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]^T$

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

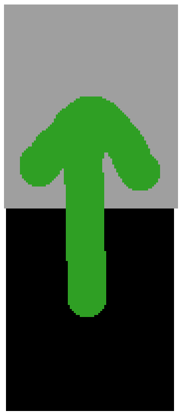
Magnitude:

$$\theta = atan2\left(\frac{\partial I}{\partial y}, \frac{\partial I}{\partial x}\right)$$

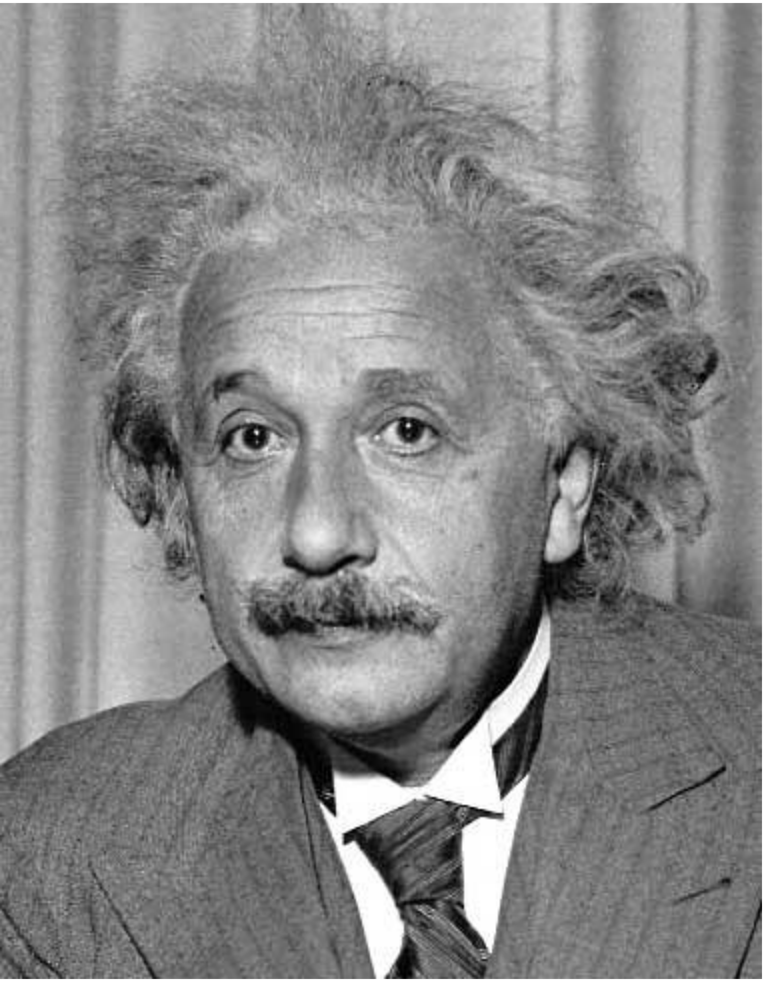
Orientation

Why?

- The direction of the gradient is always "uphill" on the surface
  - In in image, towards brightest point in the neighbourhood
  - If we need to go *mostly* in the x direction, $\left(\frac{\partial I}{\partial x}\right)$ will be larger than $\left(\frac{\partial I}{\partial y}\right)$



Reminder: How to Computer Image Derivatives? dI/dx

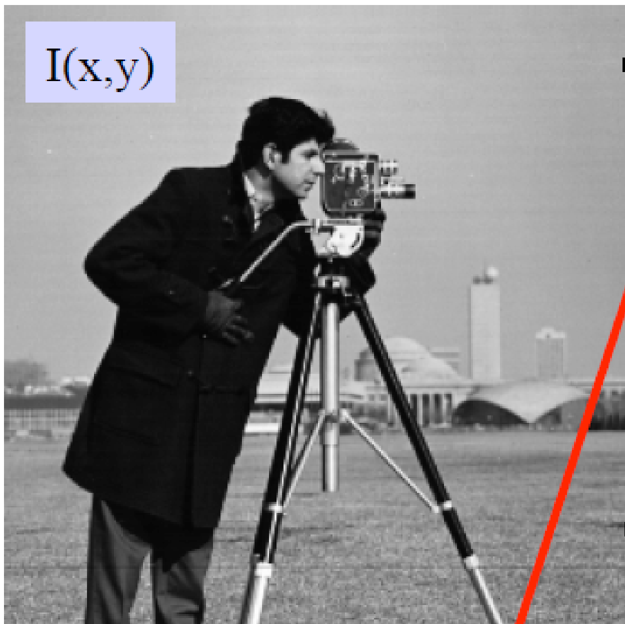


1	0	-1
2	0	-2
1	0	-1

Simple Edge Detection Using Gradients

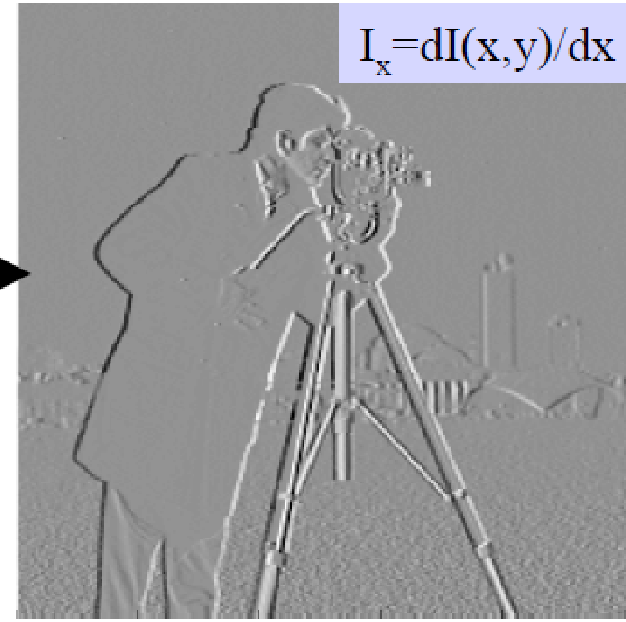
- **Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters**
- Compute gradient magnitude at each pixel
- If magnitude at a pixel exceeds a threshold, report a possible edge point.

Compute Spatial Image Gradients



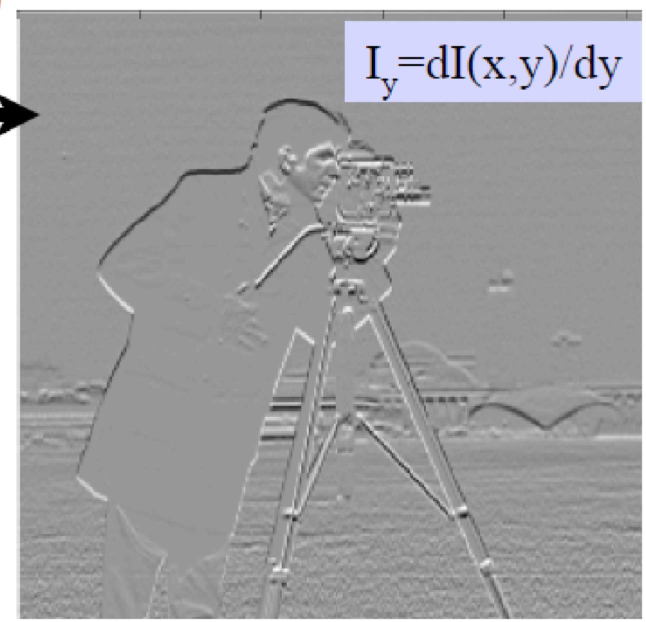
$$\frac{I(x+1,y) - I(x-1,y)}{2}$$

Partial derivative wrt x



$$\frac{I(x,y+1) - I(x,y-1)}{2}$$

Partial derivative wrt y

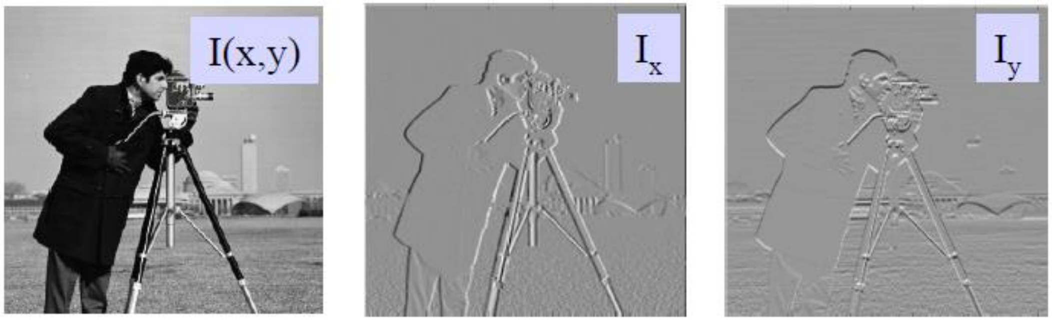


Replace with your favorite smoothing+derivative operator

Simple Edge Detection Using Gradients

- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters
- **Compute gradient magnitude at each pixel**
- If magnitude at a pixel exceeds a threshold, report a possible edge point.

Compute Gradient Magnitude



Magnitude of gradient
sqrt(Ix.^2 + Iy.^2)

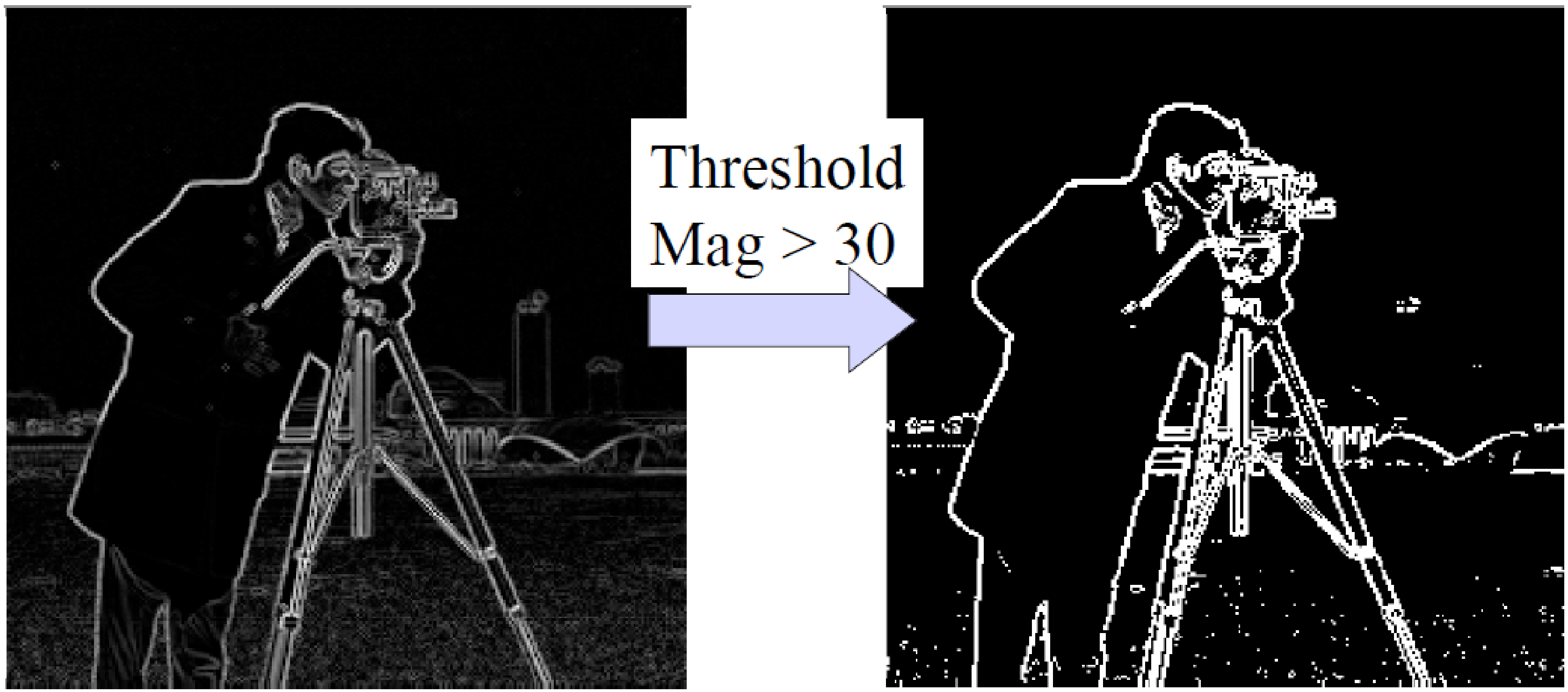
Measures steepness of slope at each pixel (= edge contrast)

Simple Edge Detection Using Gradients

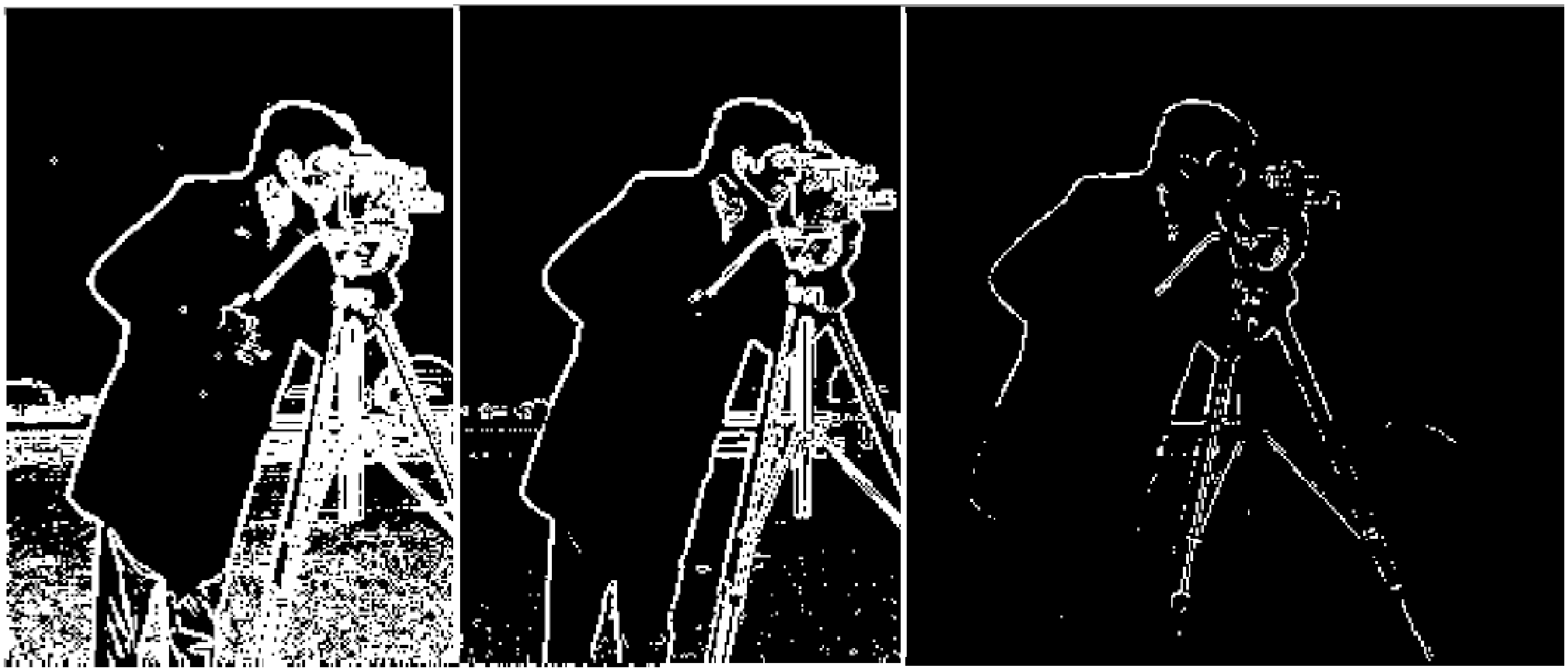
- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters
- Compute gradient magnitude at each pixel
- **If magnitude at a pixel exceeds a threshold, report a possible edge point.**

- Example – cont.:

Binary edge image



How to choose the threshold?



> 10 > 30 > 80

There is ALWAYS a tradeoff between smoothing and good edge localization!

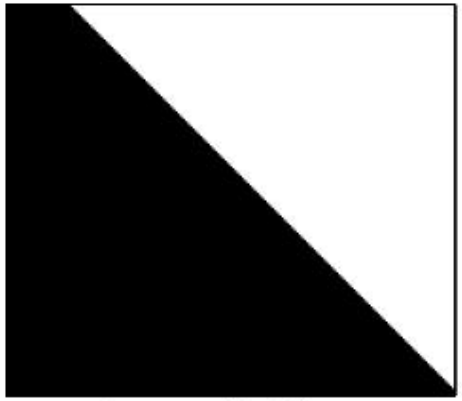
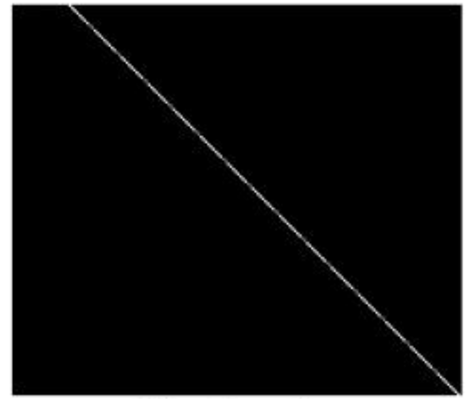


Image with Edge



Edge Location

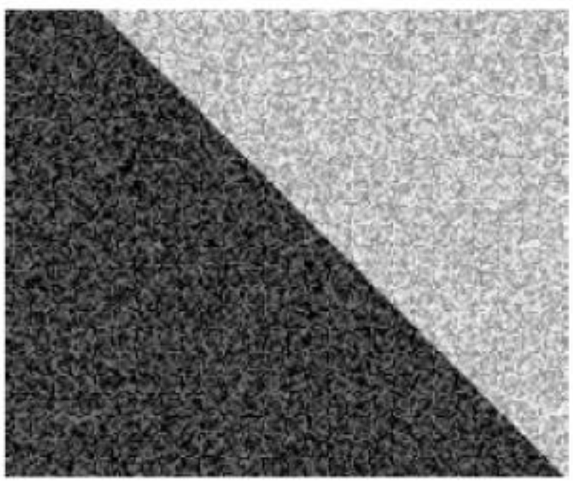
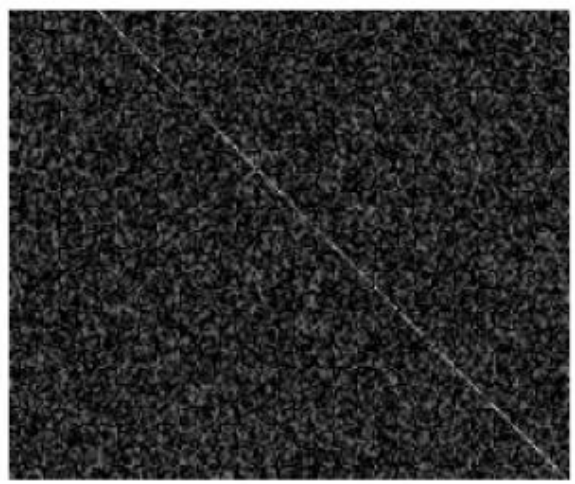


Image + Noise

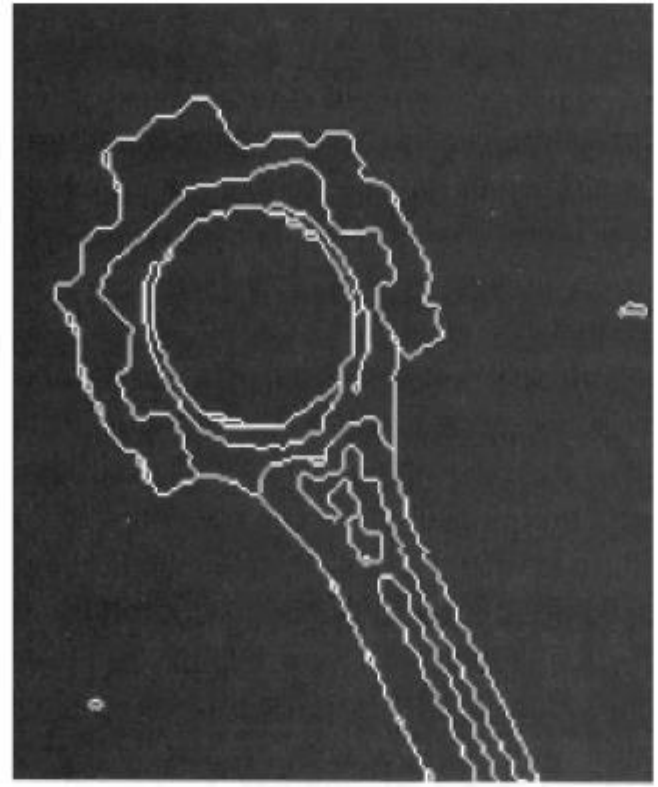


Derivatives detect
edge *and* noise

Smoothed derivative removes
noise, but blurs edge

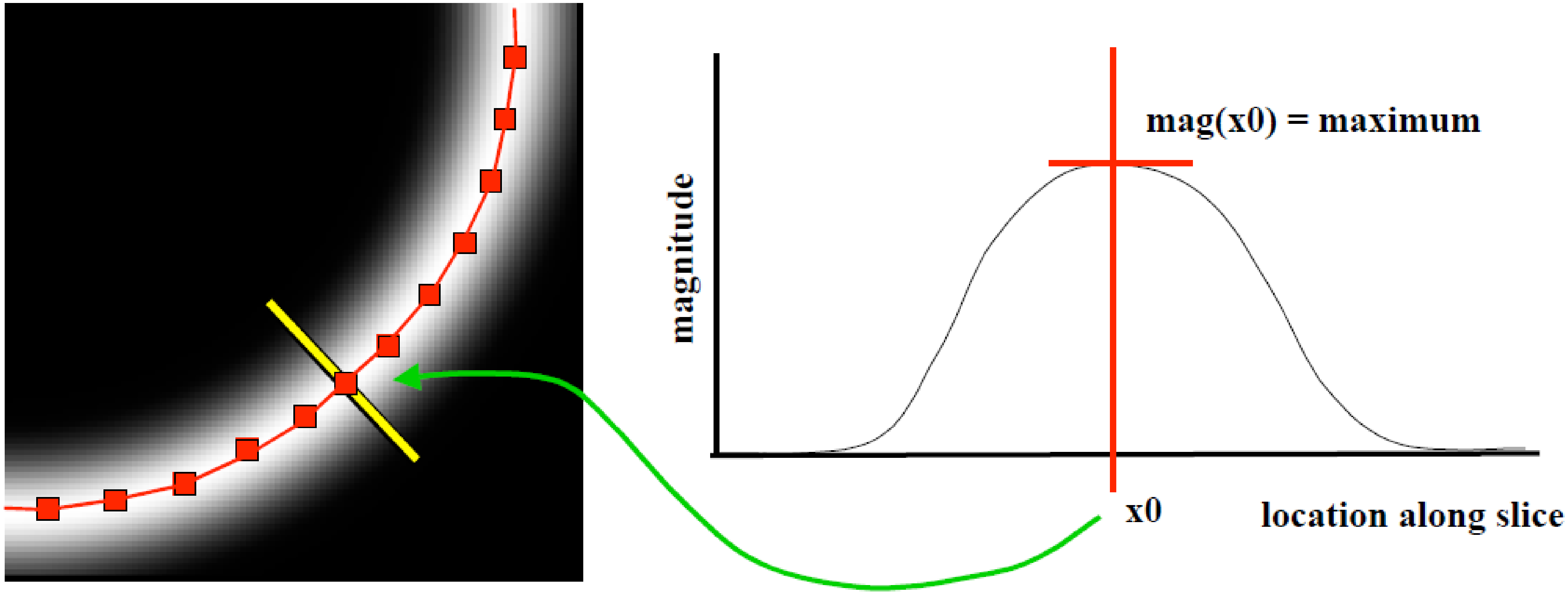
Edge Thinning and Linking

smoothing+thresholding gives us a binary mask with “thick” edges



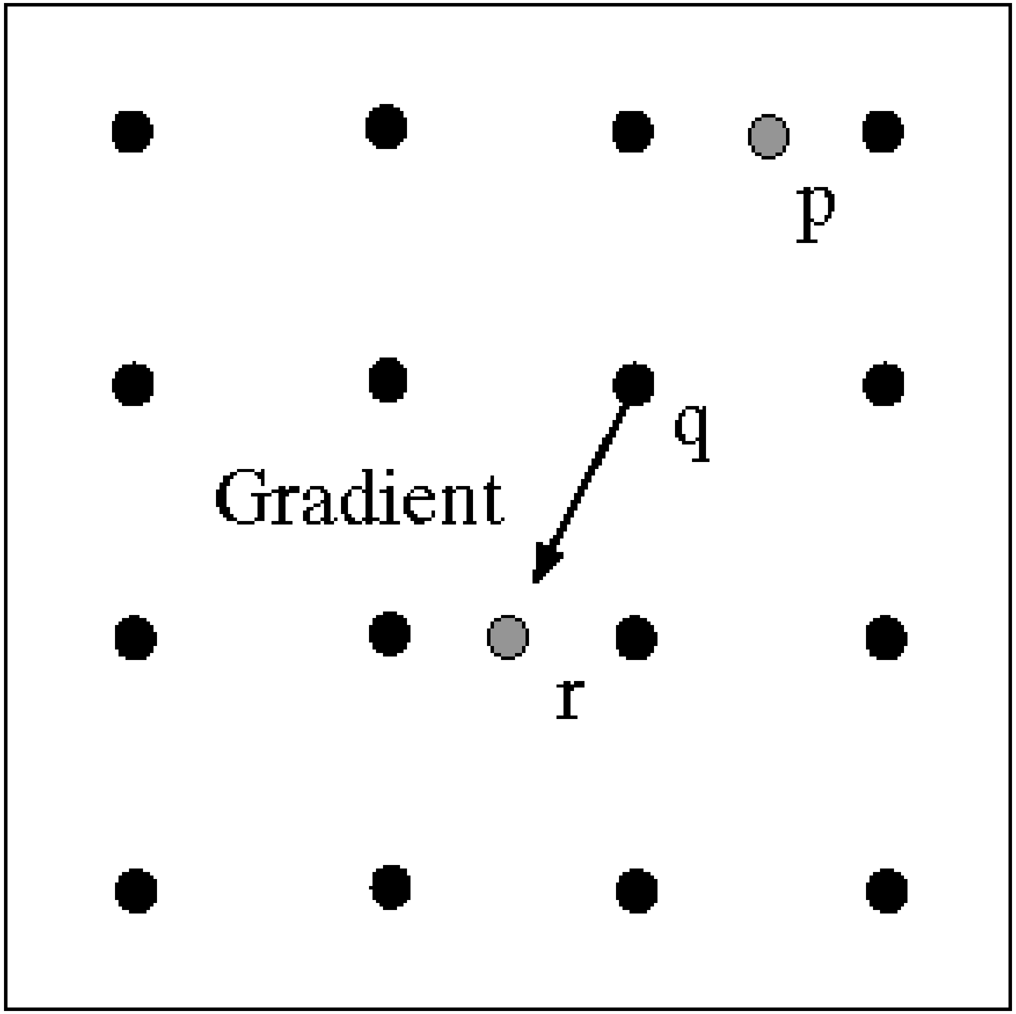
we want thin, one-pixel wide, connected contours

Canny edge detection: non-maximum suppression

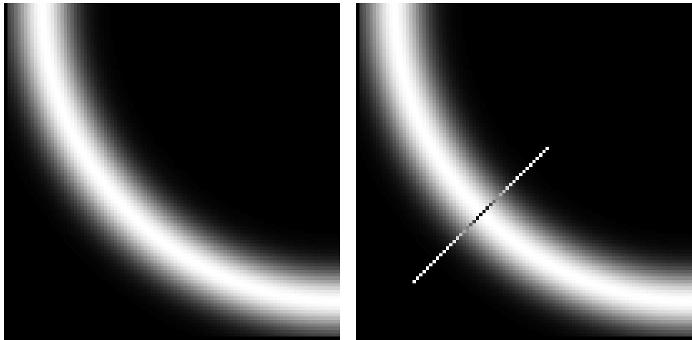


- We want to suppress points along the curve where the magnitude is non-maximal
- We do this by looking for a maximum along the *normal* to the edge (i.e., along the gradient

Non-maximum suppression for each orientation

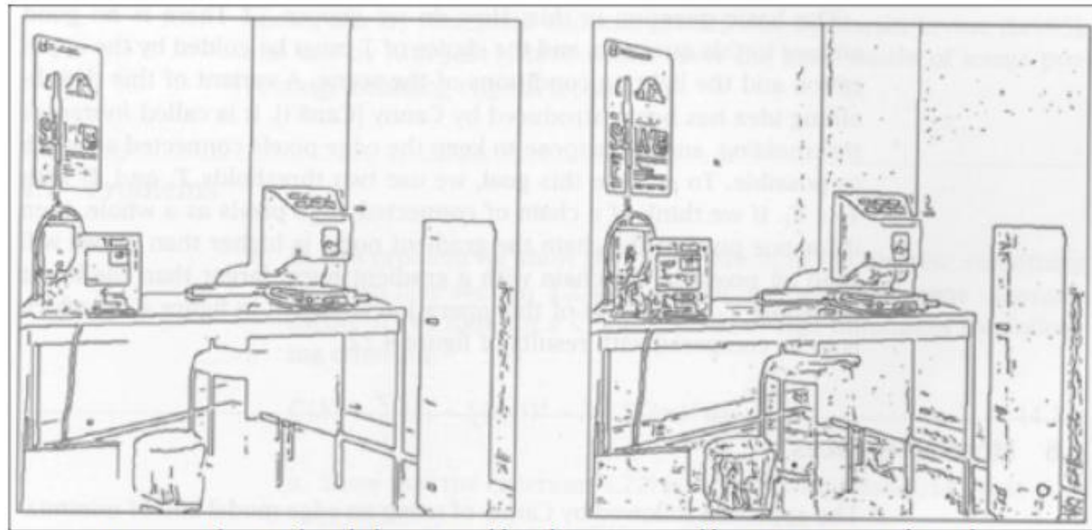


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Source: D. Forsyth

Which threshold to pick?



Two thresholds applied to gradient magnitude

T = 15 T = 5

- If the threshold is too high, we get gaps
- If the threshold is too low, extra edges

Canny: Hysteresis Thresholding

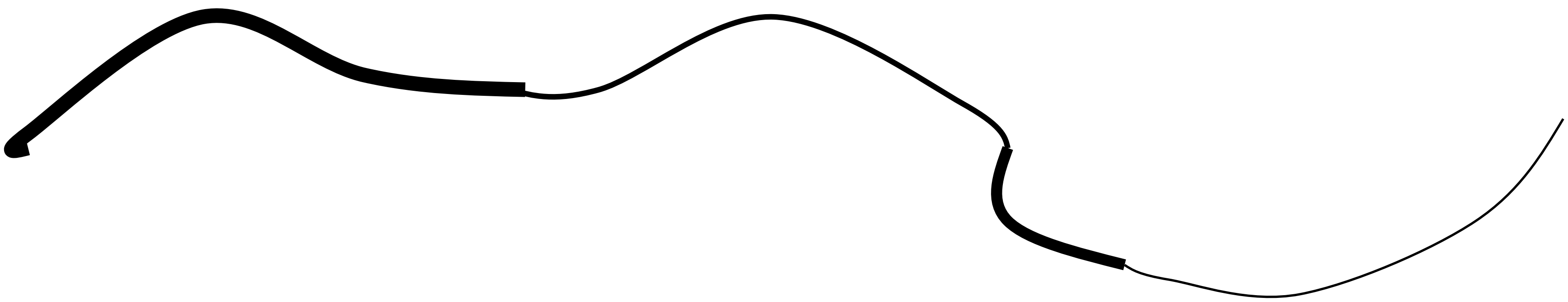
- Keep both a high threshold H and a low threshold L
- Any edges with strength <L are discarded
- Any edges with strength >H are kept
- An edge P with strength between L and H is kept only if there is a pth of edges with strength >L connecting P to an edge of strength >H

Aside

- ὑστέρησις: “Deficiency, lagging behind”
- Wikipedia: “[T]he dependence of the output of a system not only on its current input, but also on its history of past inputs”

Hysteresis thresholding

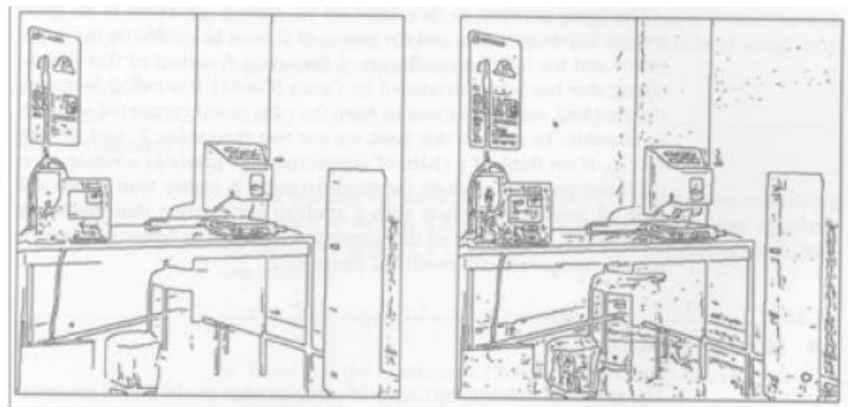
- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



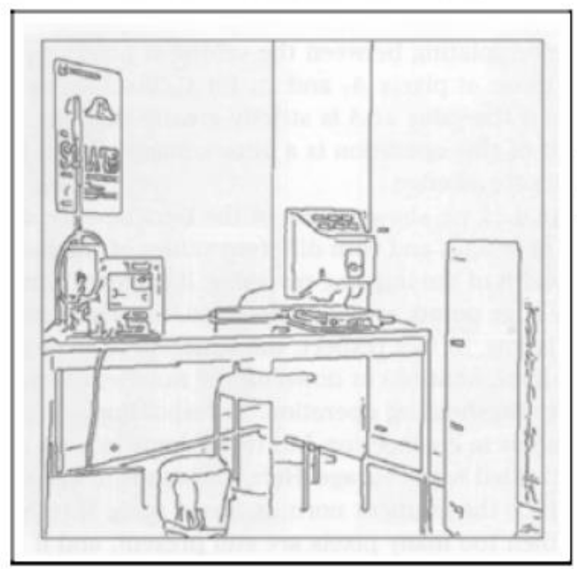
Source: S. Seitz

Hysteresis Thresholding

T=15



T=5



Hysteresis
T_h=15 T_l = 5

Hysteresis
thresholding

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

Source: S. Seitz

Canny edge detector

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute magnitude of gradient at every pixel

$$M(x,y) = |\nabla I| = \sqrt{I_x^2 + I_y^2}$$

3. Eliminate those pixels that are not local maxima of the magnitude in the direction of the gradient

4. Hysteresis Thresholding

   - Select the pixels such that $M > T_h$ (high threshold)

   - Collect the pixels such that $M > T_l$ (low threshold) that are neighbors of already collected edge points

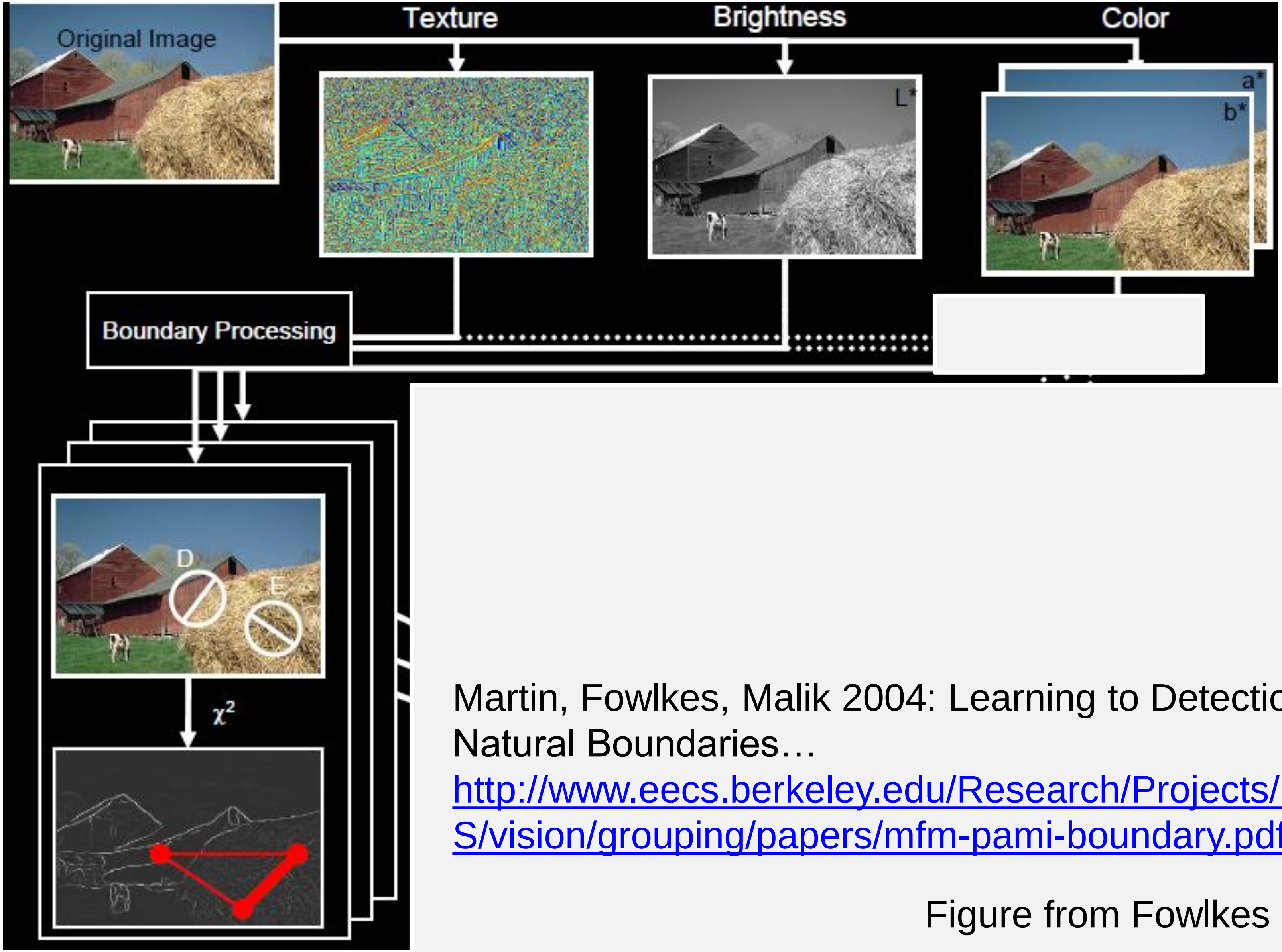
Source: D. Lowe, L. Fei-Fei

Learning to detect boundaries

image | human segmentation | gradient magnitude

- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

pB boundary detector



Martin, Fowlkes, Malik 2004: Learning to Detection Natural Boundaries…
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/mfm-pami-boundary.pdf

Figure from Fowlkes

pB Boundary Detector

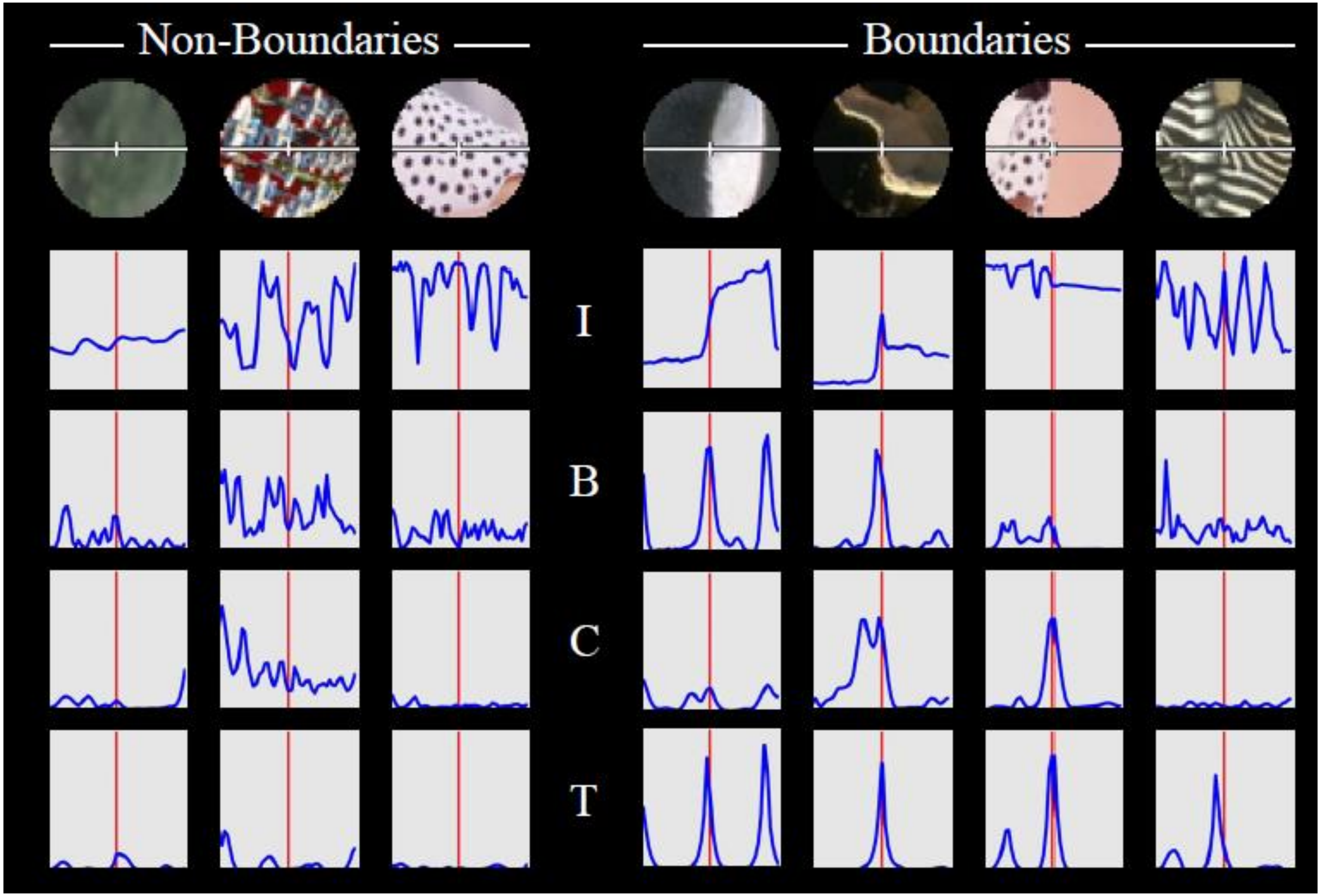


Figure from Fowlkes

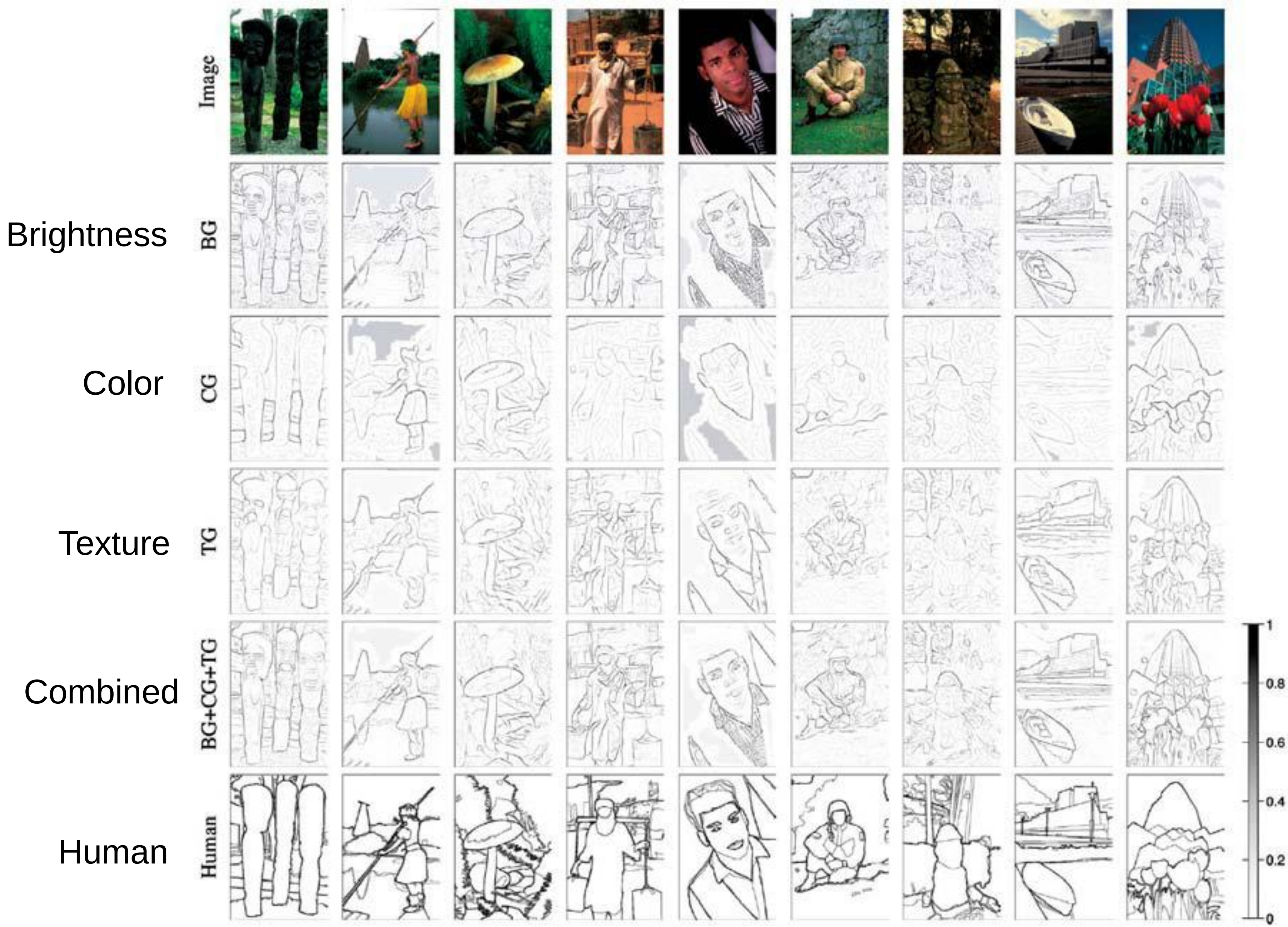
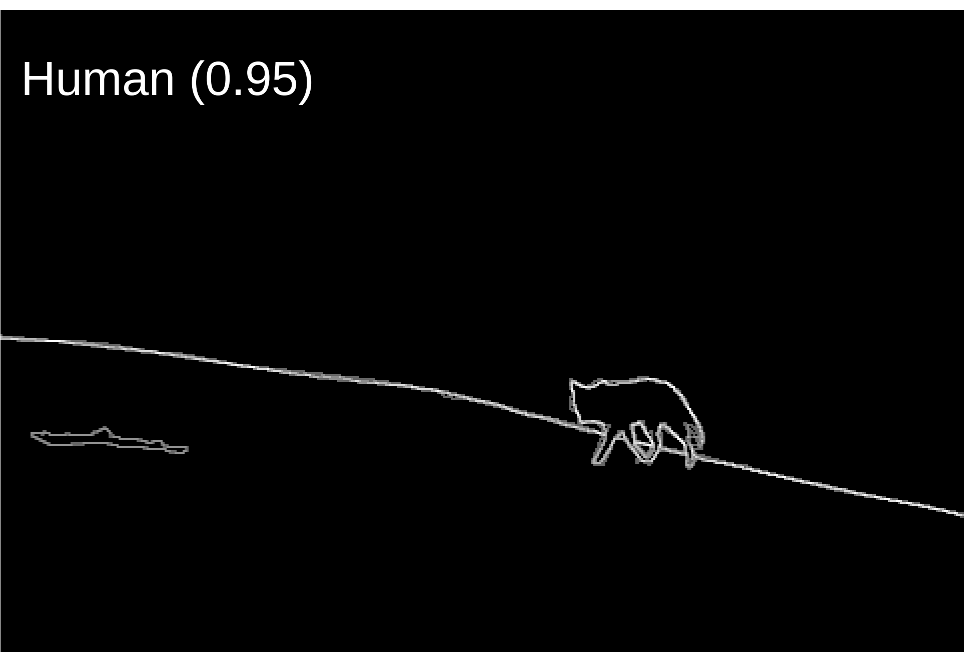
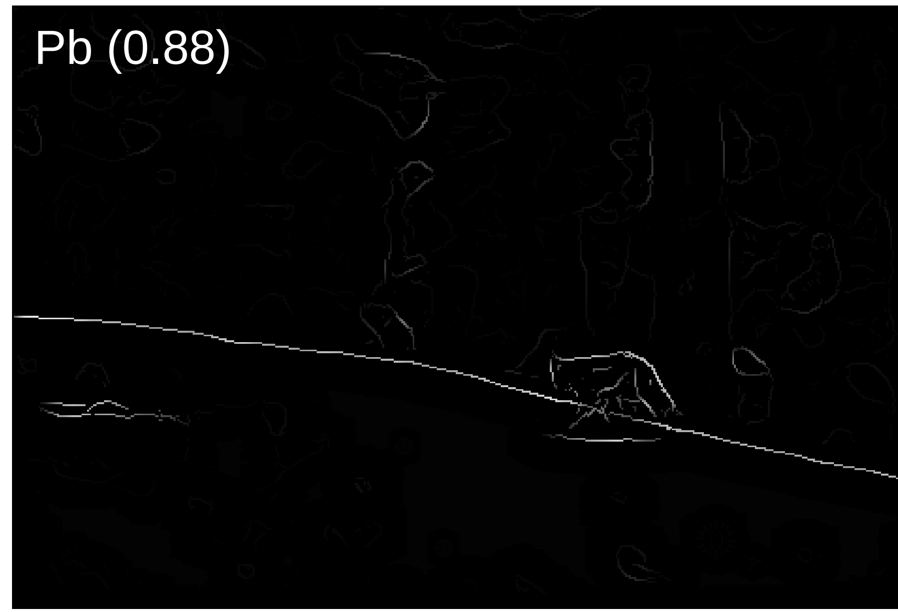


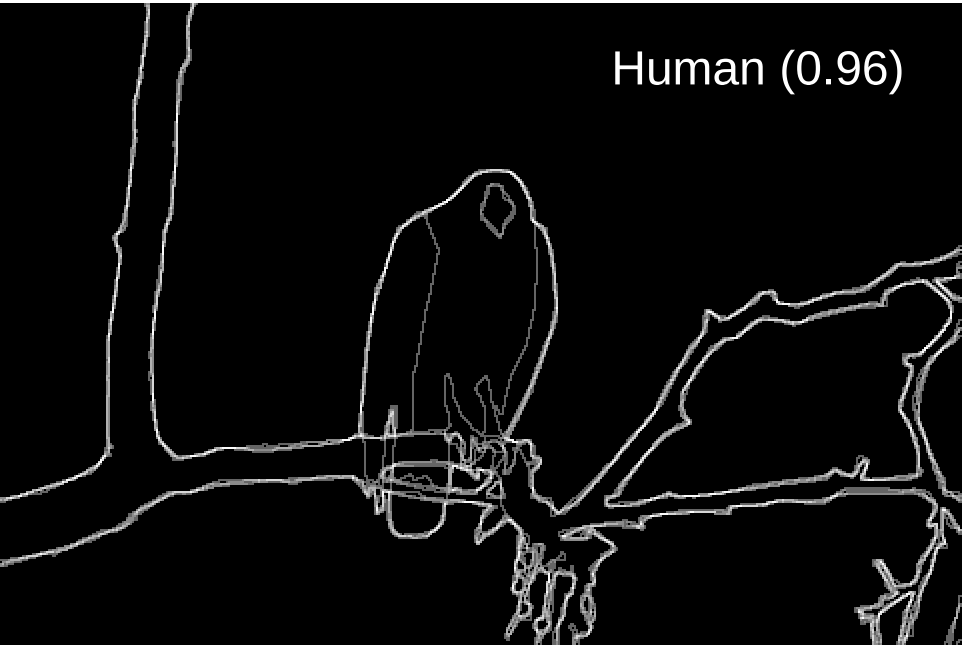
Image
Brightness
BG
Color
CG
Texture
TG
Combined
BG+CG+TG
Human
Human
1
0.8
0.6
0.4
0.2
0

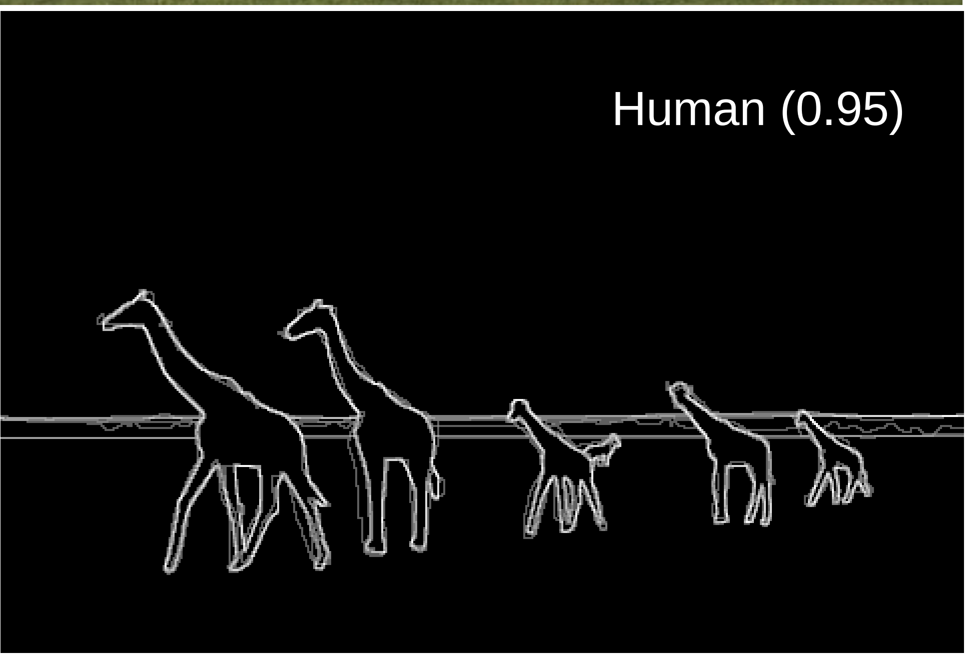
Results



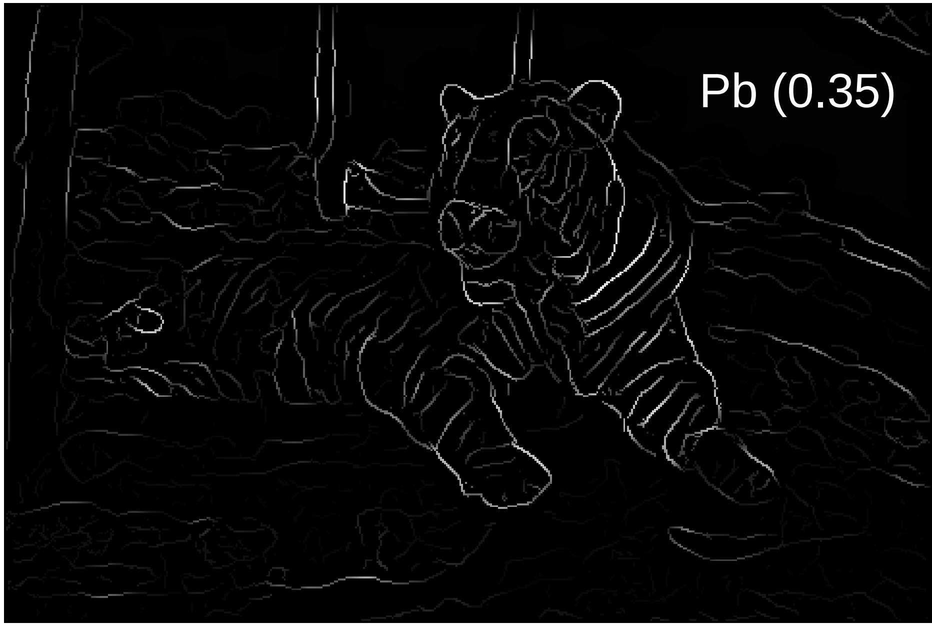


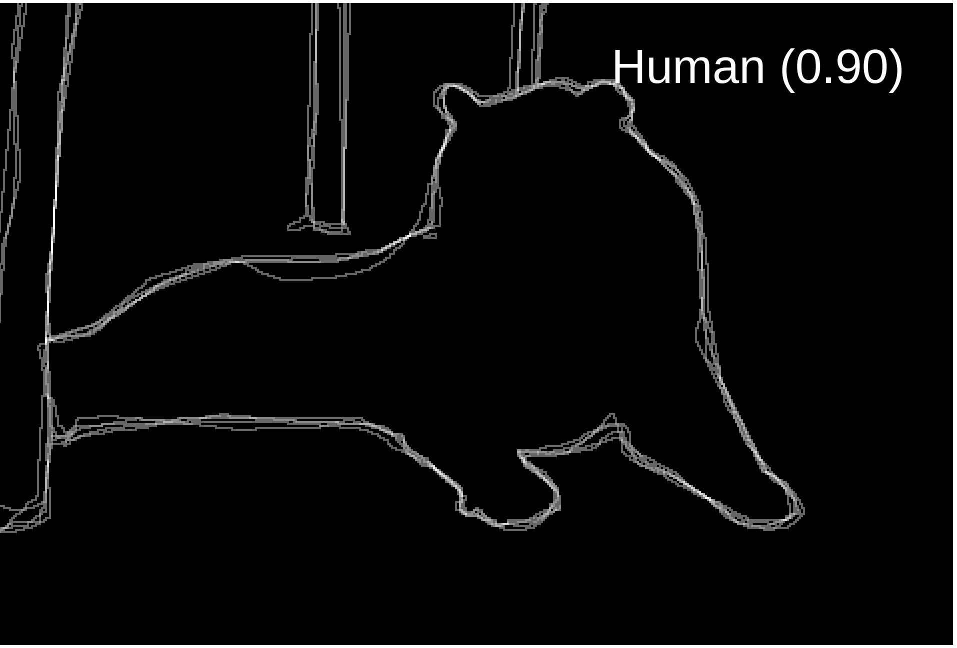
Results




Human (0.95)


Pb (0.63)





For more:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/108082-color.html

State of edge detection

- Local edge detection is mostly solved
  - Intensity **gradient**, color, **texture**
- Some methods to take into account longer contours, but could probably do better
- Poor use of object and high-level information