Convolution and Filtering

COS 429: Computer Vision

PRINCETON UNIVERSITY

Figure credits: S. Lazebnik, S. Seitz, K. Grauman, and M. Hebert

Local Neighborhoods

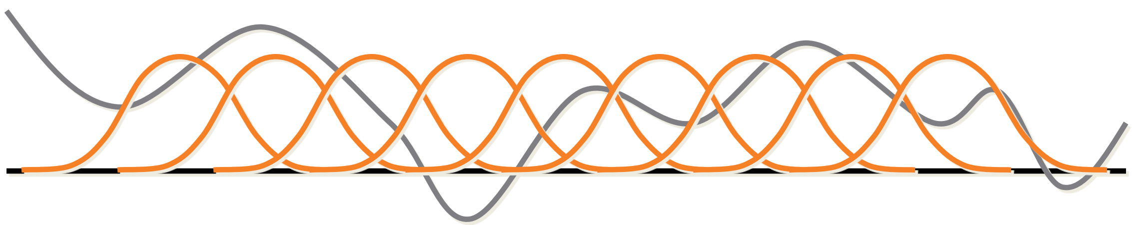
- Hard to tell anything from a single pixel
  - Example: you see a reddish pixel. Is this the object's color? Illumination? Noise?
- The next step in order of complexity is to look at local neighborhood of a pixel

Linear Filters

- Given an image $In(x,y)$ generate a new image $Out(x,y)$:
  - For each pixel (x,y), $Out(x,y)$ is a *specific* linear combination of pixels in the neighborhood of $In(x,y)$
- This algorithm is
  - Linear in input values (intensities)
  - Shift invariant

Discrete Convolution

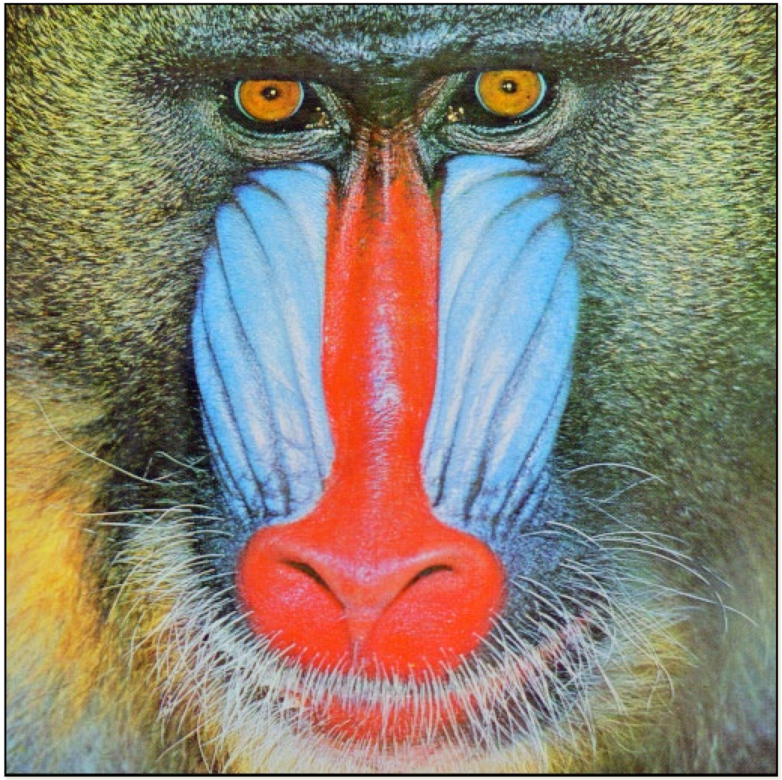
- This is the discrete analogue of convolution



$$f(x) * g(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt$$

- Pattern of weights = "filter kernel"
- Will be useful in smoothing, edge detection

Example: Smoothing



Original: Mandrill

Smoothed with Gaussian kernel

Gaussian Filters

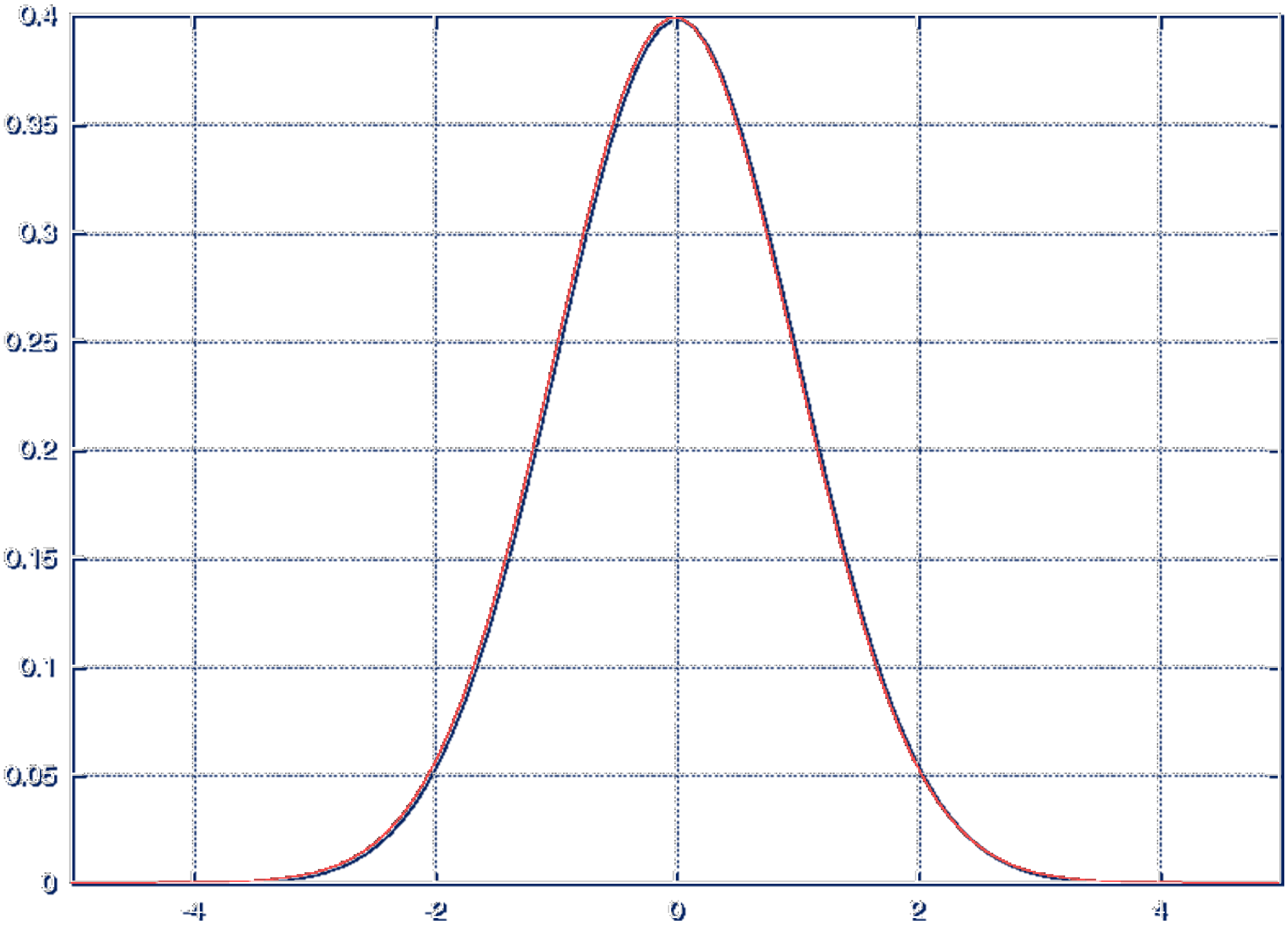
- One-dimensional Gaussian

$$G_1(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

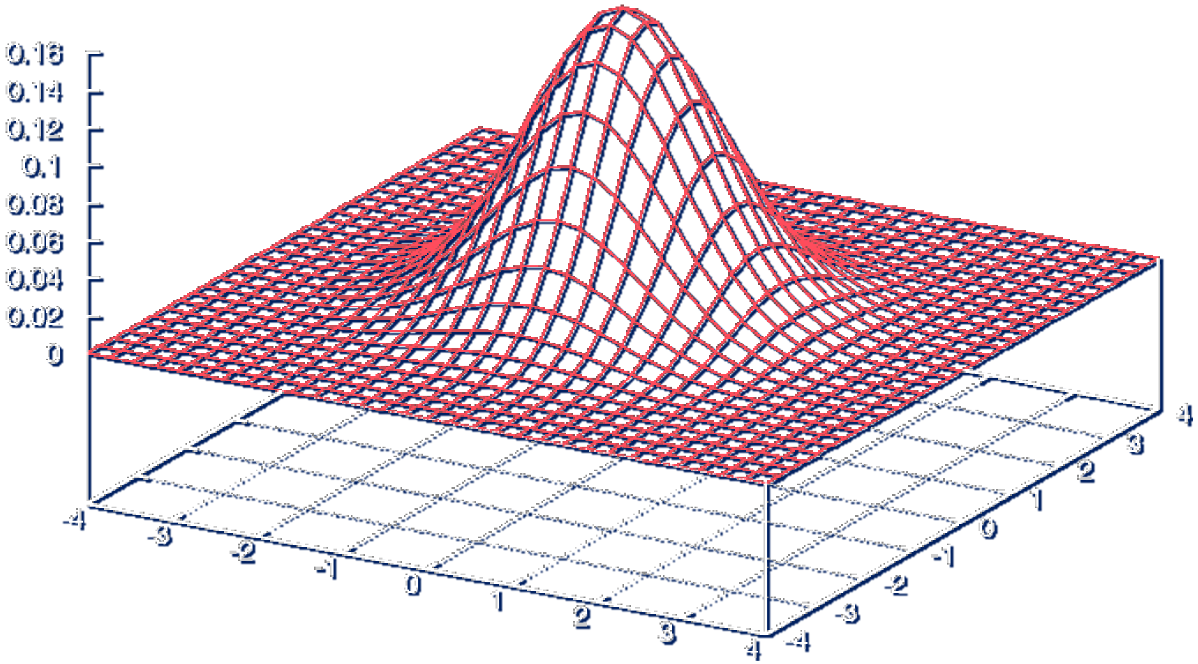
- Two-dimensional Gaussian

$$G_2(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian Filters



Gaussian Filters



Gaussian Filters

- Gaussians are used because:
  - Smooth
  - Decay to zero rapidly
  - Simple analytic formula
  - Central limit theorem: limit of applying (most) filters multiple times is some Gaussian
  - Separable:

$$G_2(x, y) = G_1(x)\, G_1(y)$$

Computing Discrete Convolutions

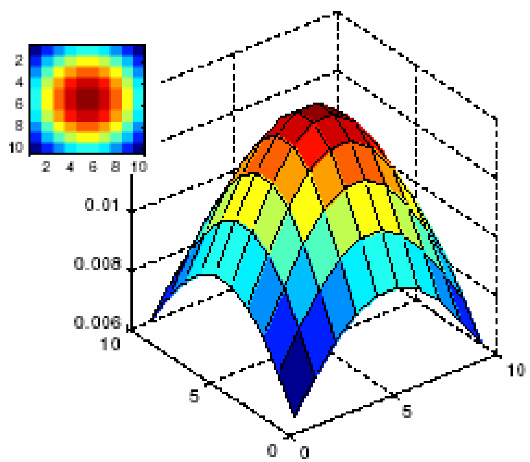
$$Out(x,y) = \sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} f(i,j)\, In(x-i, y-j)$$

- What happens near edges of image?
  - Ignore (Out is smaller than In)
  - Pad with zeros (edges get dark)
  - Replicate edge pixels
  - Wrap around
  - Reflect
  - Change filter

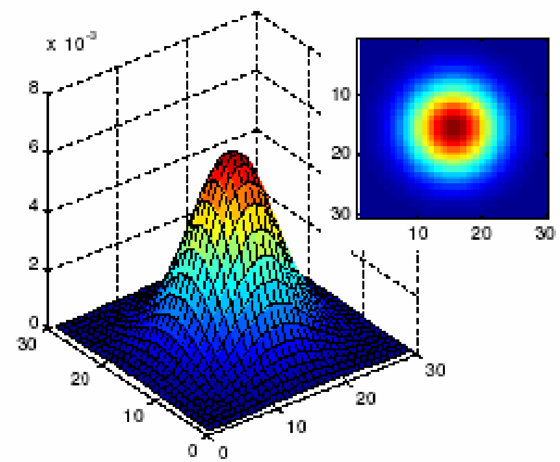
Computing Discrete Convolutions

$$Out(x,y) = \sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} f(i,j)\, In(x-i, y-j)$$

- What happens if kernel is infinite?
  - Truncate when filter falls off to near zero
  - For Gaussian, typical support between 2σ and 3σ



σ = 5 with 10x10 kernel



σ = 5 with 30x30 kernel

Source: K. Grauman

Computing Discrete Convolutions

$$Out(x,y) = \sum_{i=i_{min}}^{i_{max}} \sum_{j=j_{min}}^{j_{max}} f(i,j)\, In(x-i, y-j)$$

- How long does it take?
  - If In is $n \times n$, f is $m \times m$, naive computation takes time $O(m^2 n^2)$
  - OK for small filter kernels, bad for large ones

Fourier Transforms

- Define *Fourier transform* of function *f* as

$$F(\omega) = \mathcal{F}\big(f(x)\big) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(x)\, e^{i\omega x}\, dx$$

- F is a function of frequency – describes how much of each frequency is contained in f
- Fourier transform is invertible

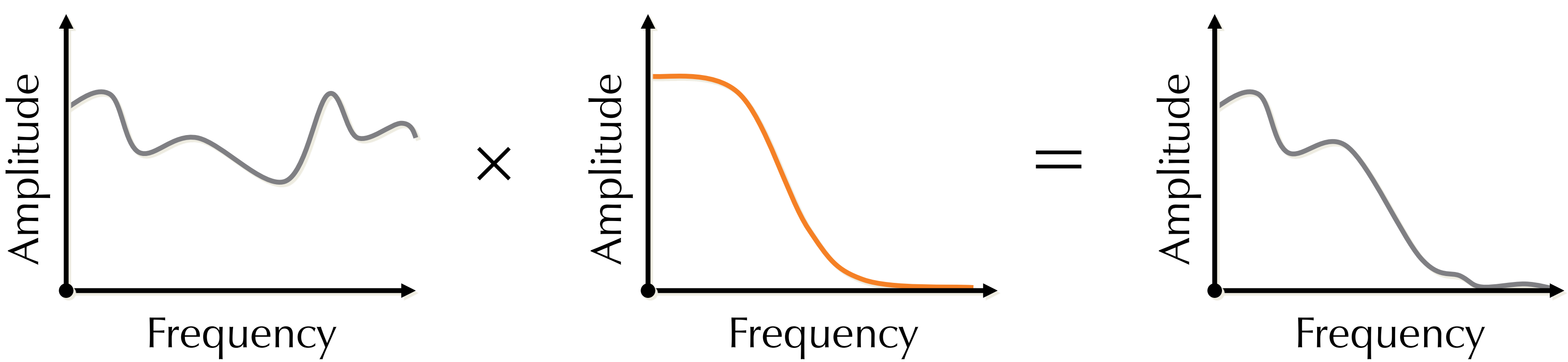
Fourier Transform and Convolution

- Fourier transform turns convolution into multiplication:

$$\mathcal{F}\big(f(x) * g(x)\big) = \mathcal{F}\big(f(x)\big)\,\mathcal{F}\big(g(x)\big)$$

Fourier Transform and Convolution

- Useful application #1: Use frequency space to understand effects of filters
  - Example: Fourier transform of a Gaussian is a Gaussian
  - Thus: attenuates high frequencies



Fourier Transform and Convolution

- Useful application #2: Efficient computation
  - Fast Fourier Transform (FFT) takes time

$$O(n \log n)$$

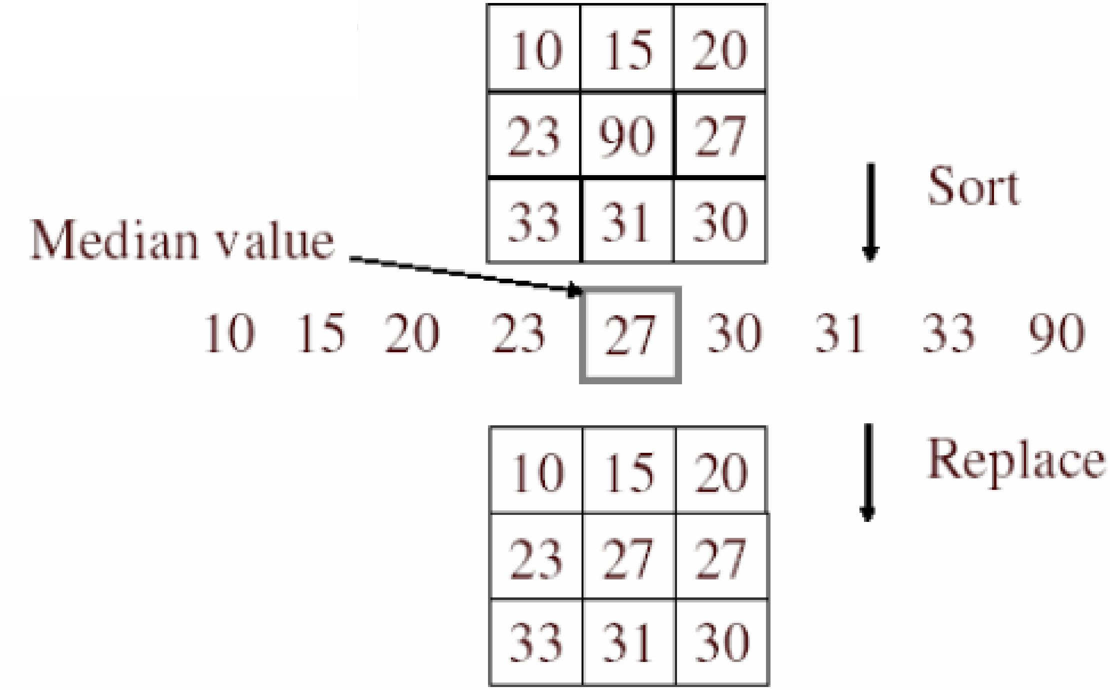
  - Thus, convolution can be performed in time

$$O(n \log n + m \log m)$$

  - Greatest efficiency gains for large filters ($m \sim n$)

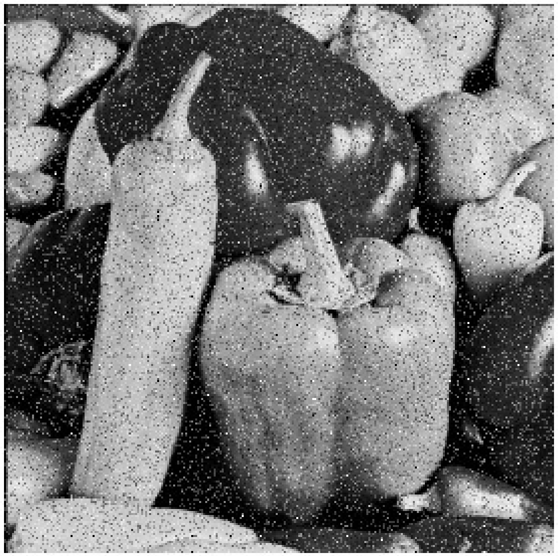
Alternative: Median Filtering

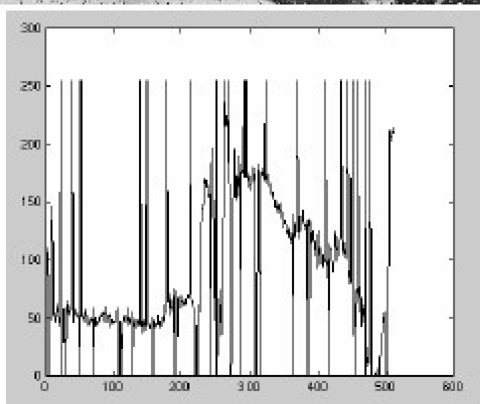
- A median filter operates over a window by selecting the median intensity in the window



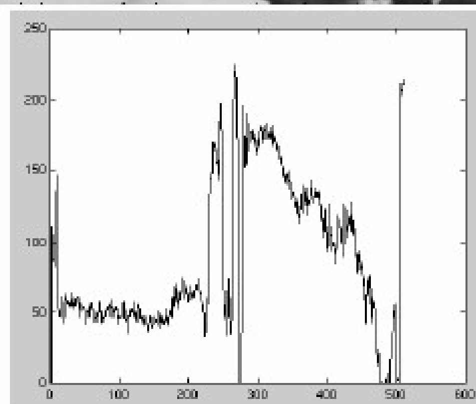
Credit: K. Grauman

Median Filter





Salt-and-pepper noise



Median filtered

Credit: M. Hebert

Gaussian vs. Median filtering

3x3 5x5 7x7

Gaussian

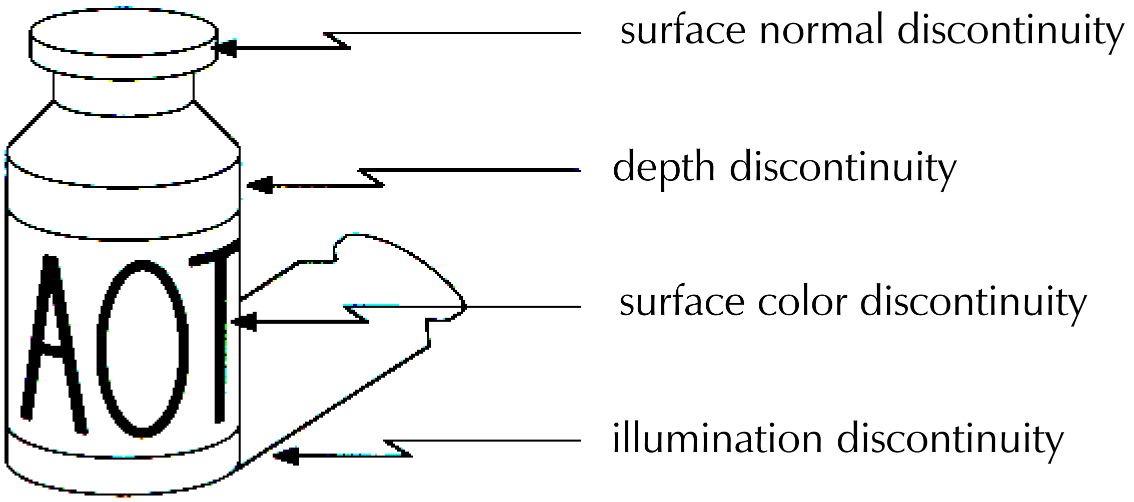
Median

Edge Detection

Winter in Kraków photographed by Marcin Ryczek

Origin of Edges

- Edges are caused by a variety of factors:



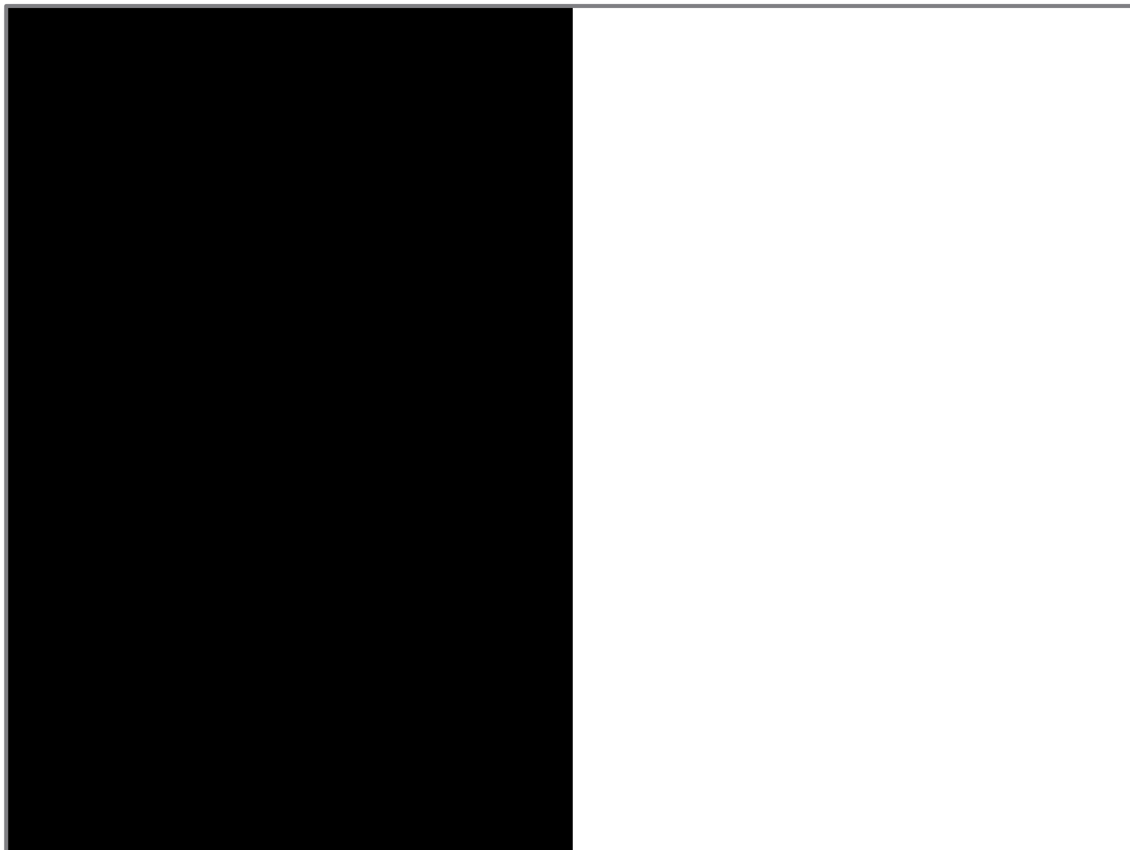
Credit: Steve Seitz

Edge Detection

- Intuitively, much of semantic and shape information is available in the edges
- Ideal: artist's line drawing (but artist is also using object-level knowledge)
- But what, mathematically, is an edge?

Credit: D. Lowe

What is an Edge?

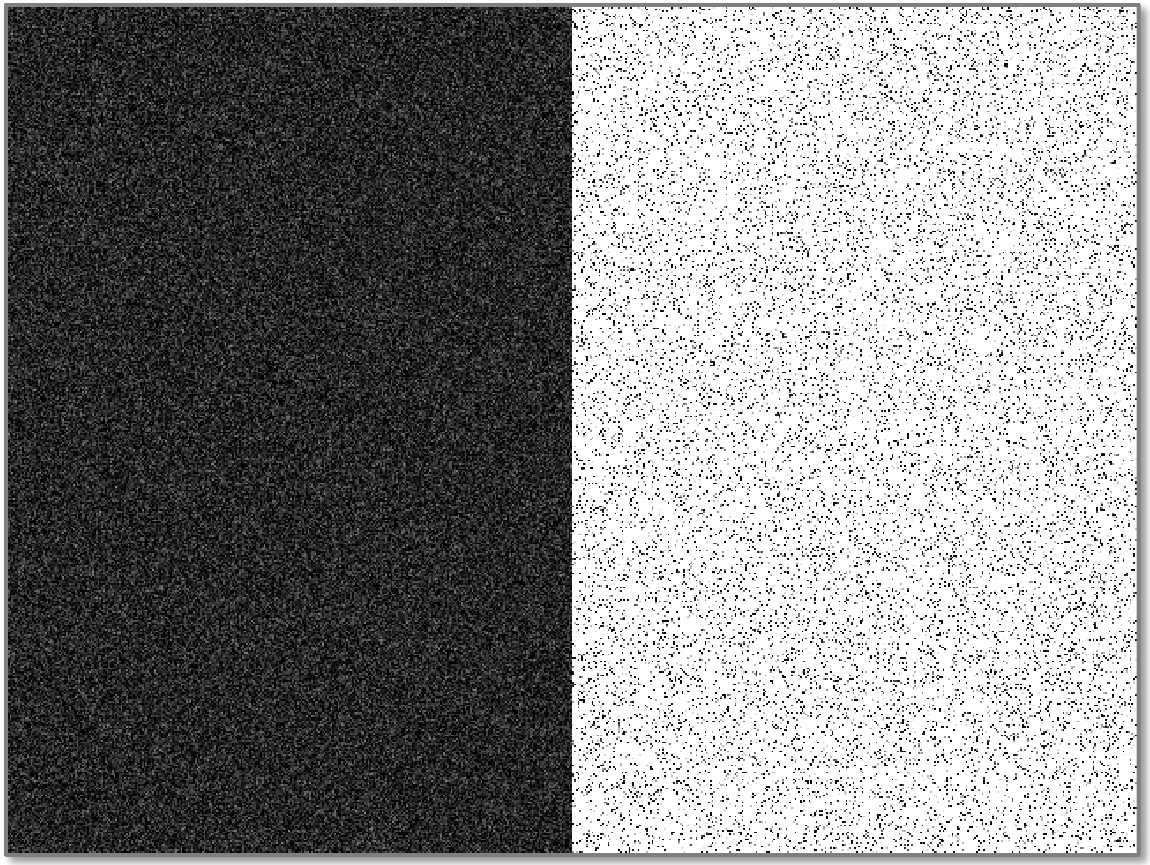


Edge easy to find

What is an Edge?

Where is edge? Single pixel wide or multiple pixels?

What is an Edge?

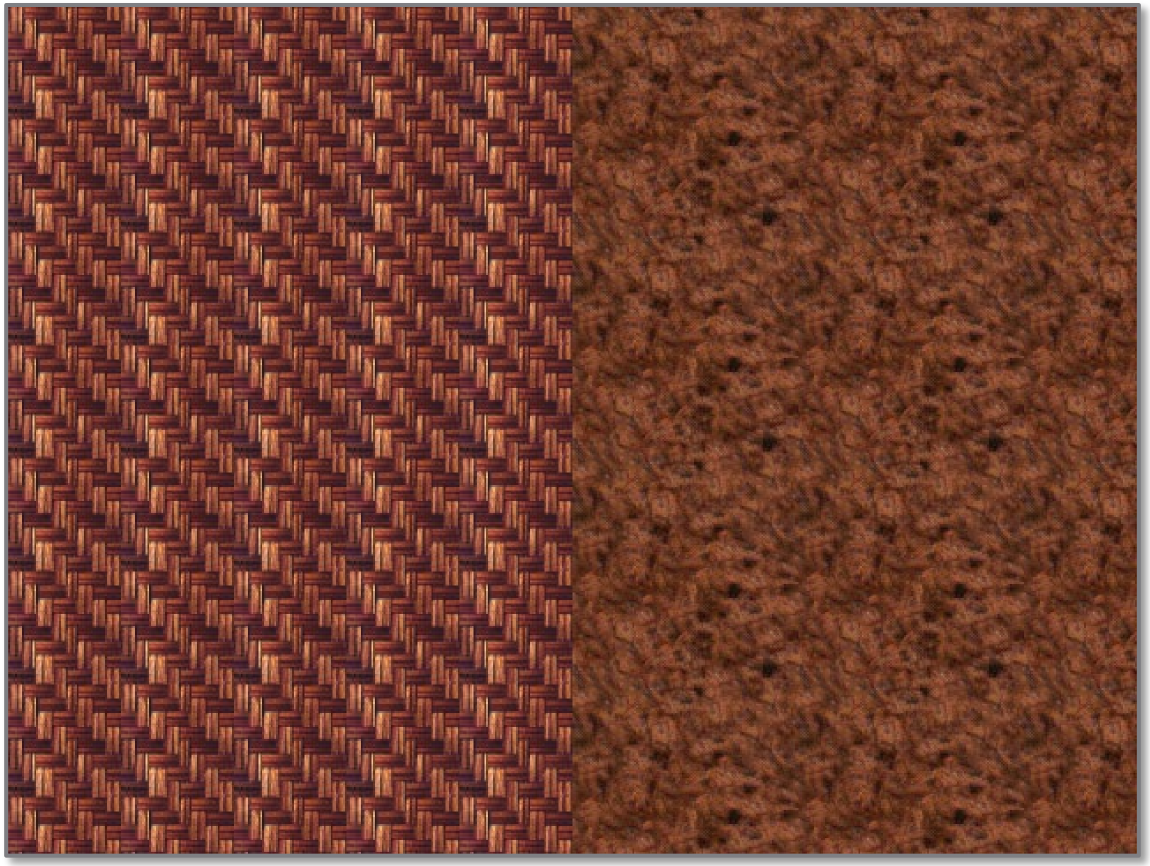


Noise: have to distinguish noise from actual edge

What is an Edge?

Is this one edge or two?

What is an Edge?



Texture discontinuity

Formalizing Edge Detection

- Look for strong step edges

$$\frac{dI}{dx} > \tau$$

- One pixel wide: look for *maxima* in $dI\ /\ dx$
- Noise rejection: smooth (with a Gaussian) over a neighborhood of size σ

Canny Edge Detector

- Smooth
- Find derivative
- Find maxima
- Threshold

Canny Edge Detector

- First, smooth with a Gaussian of some width σ

Canny Edge Detector

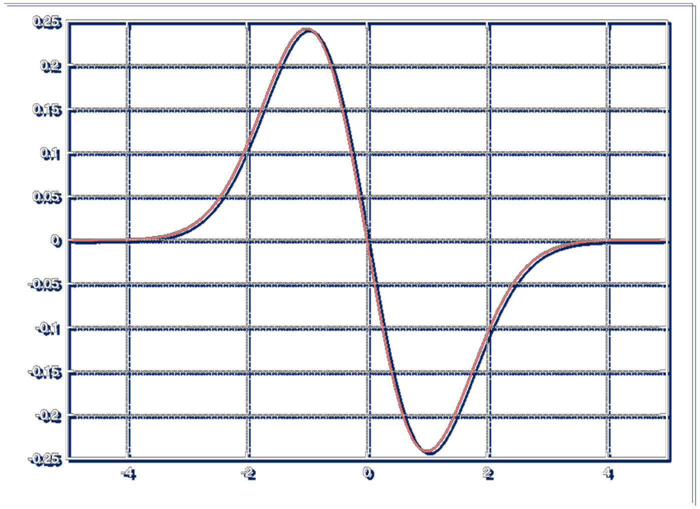
- Next, find "derivative"
- What is derivative in 2D? Gradient:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)$$

Canny Edge Detector

- Useful fact #1: differentiation "commutes" with convolution

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



- Useful fact #2: Gaussian is separable:

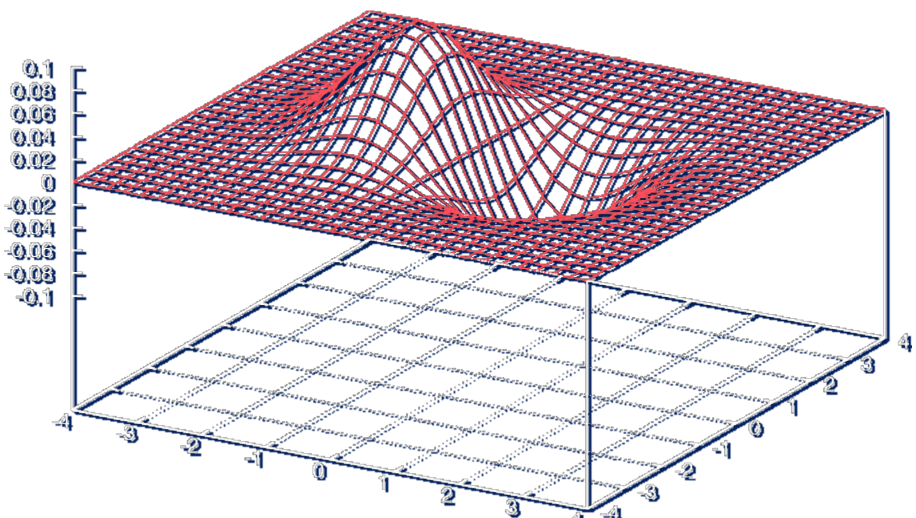
$$G_2(x, y) = G_1(x)\, G_1(y)$$

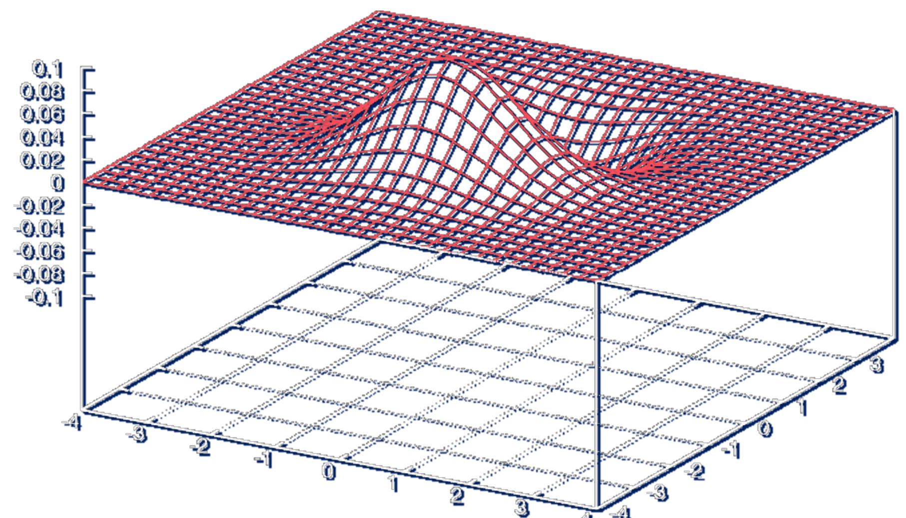
Canny Edge Detector

- Thus, combine first two stages of Canny:

$$\nabla\big(f(x,y) * G_2(x,y)\big) = \begin{bmatrix} f(x,y) * \big(G_1'(x)G_1(y)\big) \\ f(x,y) * \big(G_1(x)G_1'(y)\big) \end{bmatrix}$$

$$= \begin{bmatrix} f(x,y) * G_1'(x) * G_1(y) \\ f(x,y) * G_1(x) * G_1'\,(y) \end{bmatrix}$$





Canny Edge Detector

Original Image

Smoothed Gradient Magnitude

Canny Edge Detector

- Nonmaximum suppression
  - Eliminate all but local maxima in gradient magnitude (sqrt of sum of squares of x and y components)
  - At each pixel p look along direction of gradient: if either neighbor is bigger, set p to zero
  - In practice, quantize direction to horizontal, vertical, and two diagonals
  - Result: "thinned edge image"

Canny Edge Detector

- Final stage: thresholding
- Simplest: use a single threshold
- Better: use two thresholds
  - Find chains of touching edge pixels, all $\geq \tau_{low}$
  - Each chain must contain at least one pixel $\geq \tau_{high}$
  - Helps eliminate dropouts in chains, without being too susceptible to noise
  - "Thresholding with hysteresis"

Canny Edge Detector

Original Image

Edges

Faster Edge Detectors

- Can build simpler, faster edge detector by omitting some steps:
  - No nonmaximum suppression
  - No hysteresis in thresholding
  - Simpler filters (approx. to gradient of Gaussian)
    - Sobel: $\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$
    - Roberts: $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$

Second-Derivative-Based Edge Detectors

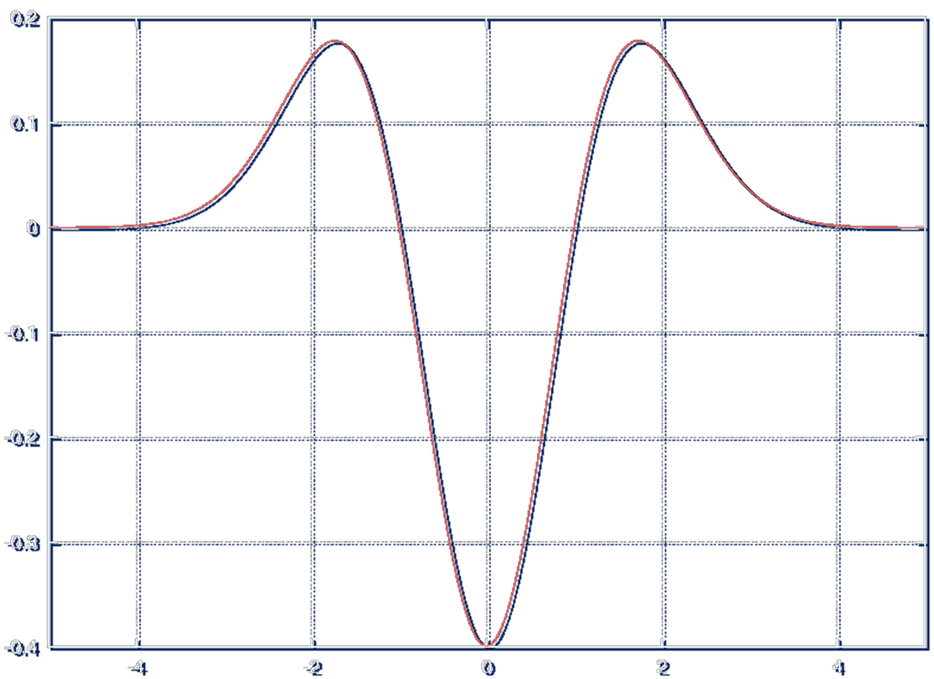
- To find local maxima in derivative, look for zeros in second derivative
- Analogue in 2D: Laplacian

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

- Marr-Hildreth edge detector

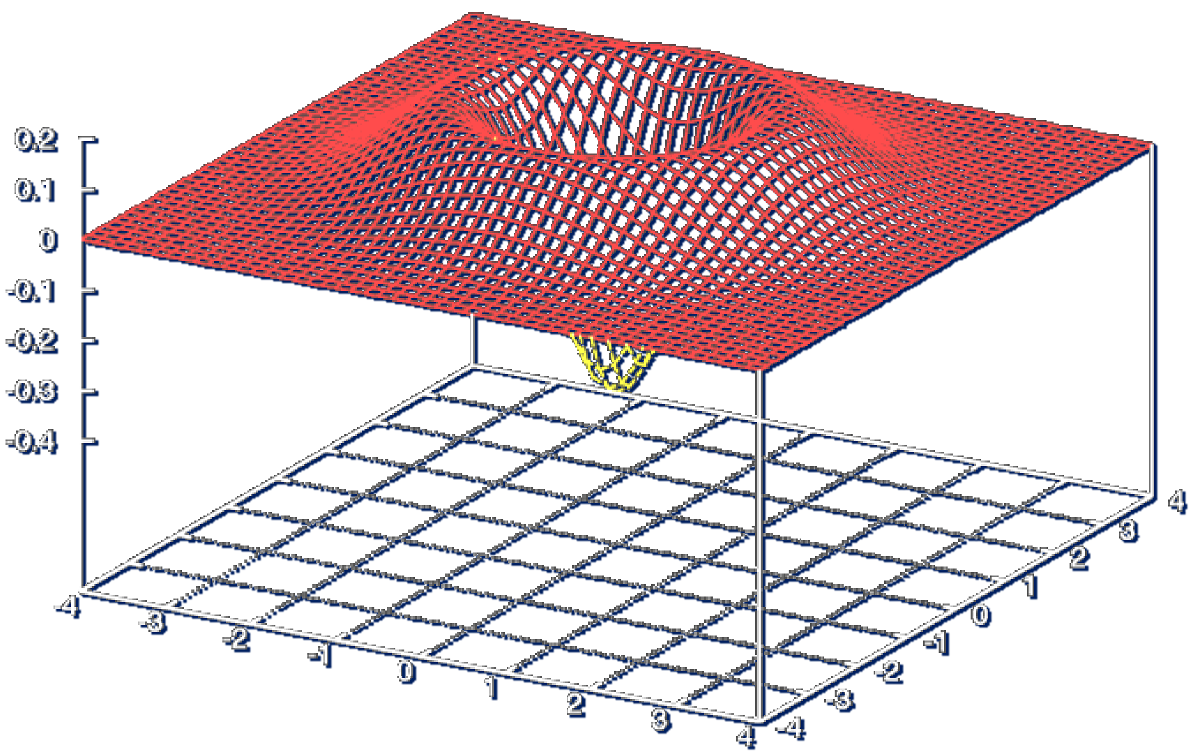
LOG

- As before, combine Laplacian with Gaussian smoothing: Laplacian of Gaussian (LOG)



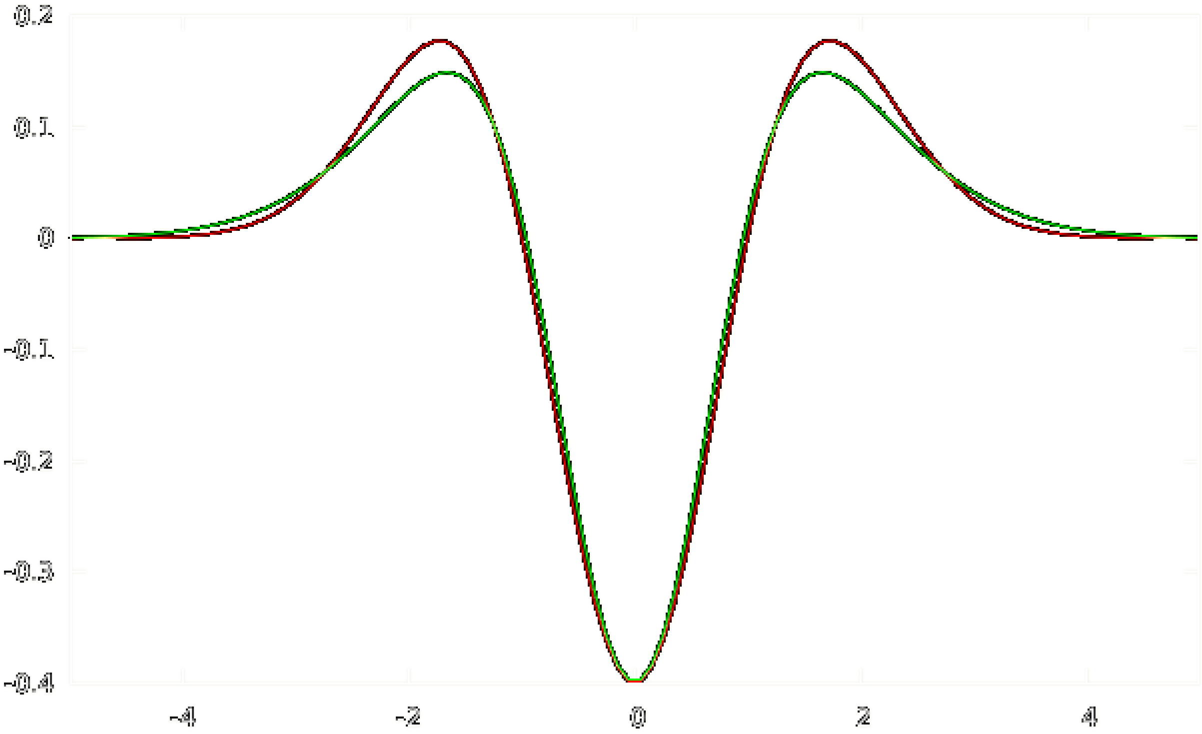
LOG

- As before, combine Laplacian with Gaussian smoothing: Laplacian of Gaussian (LOG)



LOG vs. DOG

- Laplacian of Gaussian sometimes approximated by Difference of Gaussians



$\nabla^2 G_1(x, \sigma)$

$\sqrt{2}\left[G_1(x, \sigma\sqrt{2}) - G_1(x, \sigma/\sqrt{2})\right]$

Problems with Laplacian Edge Detectors

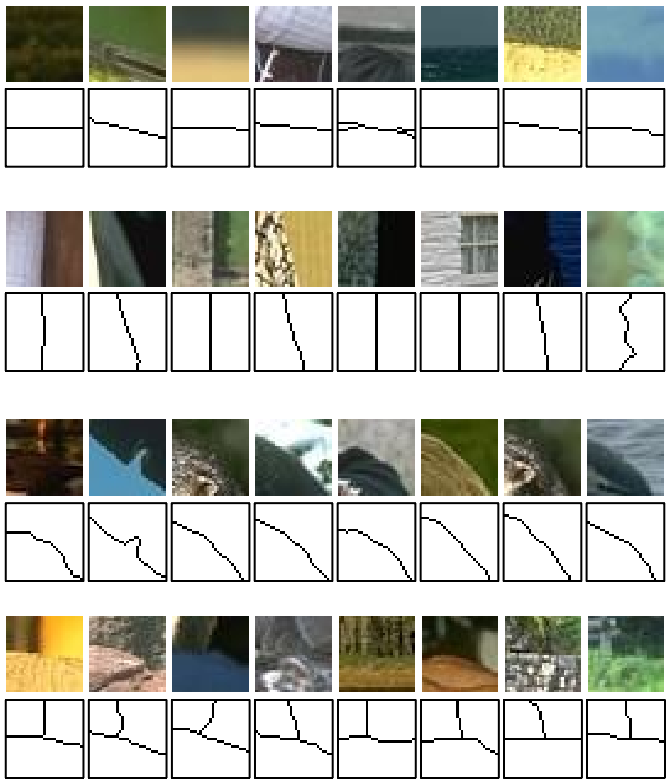
- Distinguishing local minimum vs. maximum
- Symmetric – poor performance near corners
- Sensitive to noise
  - Higher-order derivatives = greater noise sensitivity
  - Combines information *along* edge, not just perpendicular

Image gradients vs. meaningful contours

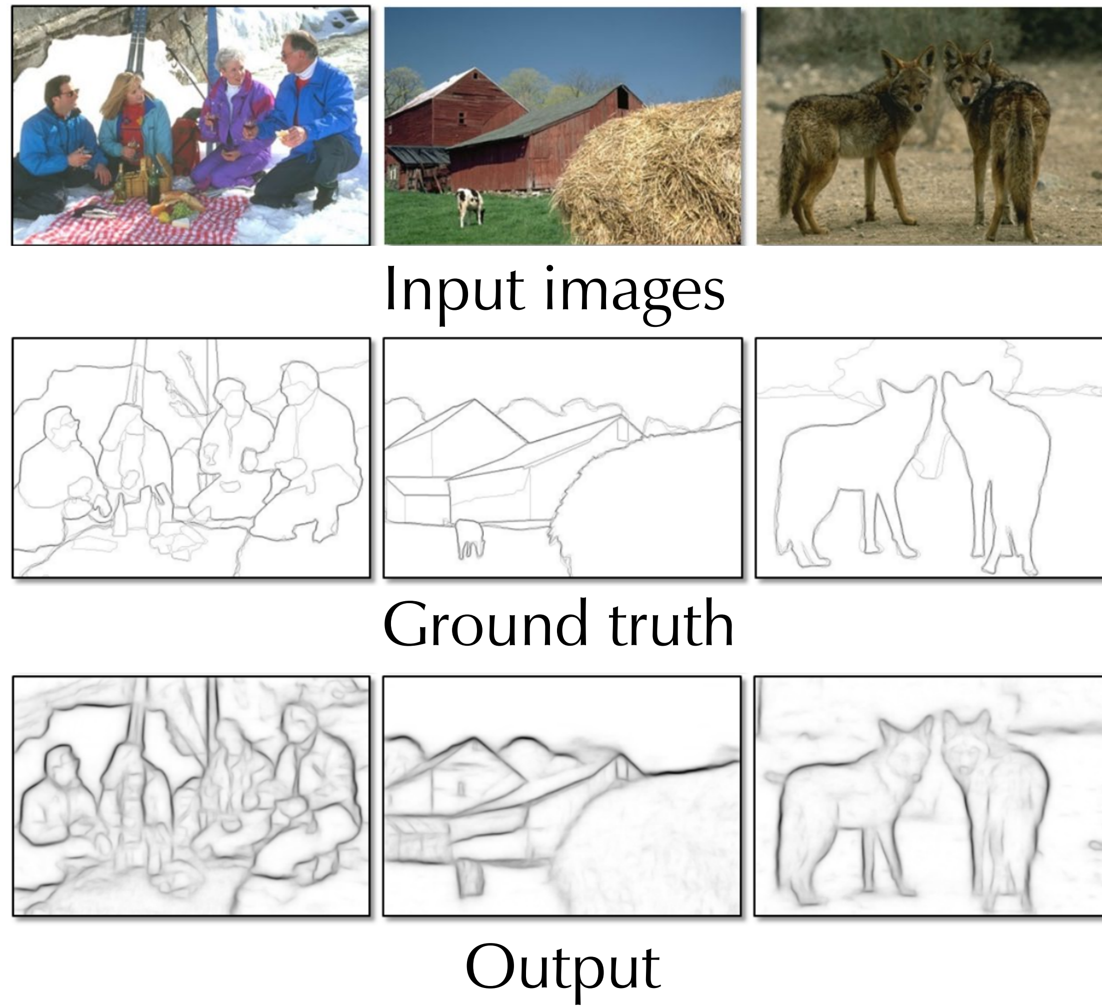
- Berkeley segmentation database:
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

image | human segmentation | gradient magnitude

Data-Driven Edge Detection



Training data



P. Dollar and L. Zitnick, [Structured forests for fast edge detection](), ICCV 2013