# Lecture 5:
# Edges and Smoothed Derivatives

Background Reading:
- T&V Section 4.1 and 4.2
- Forsyth&Ponce, Chapter 8
- Jain et.al. Chapter 5

# 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 (although an artist also relies on object-level knowledge)

Source: D. Lowe

# Simple Edge Detection Using Gradients

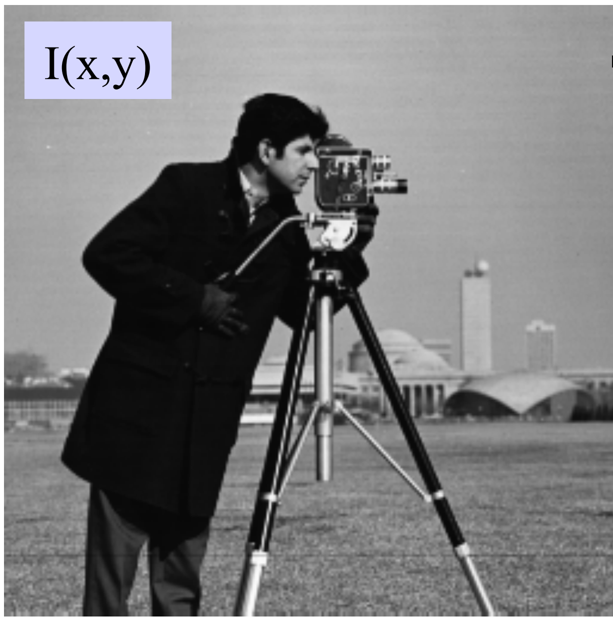
A simple edge detector using gradient magnitude

- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters

- Compute gradient magnitude at each pixel

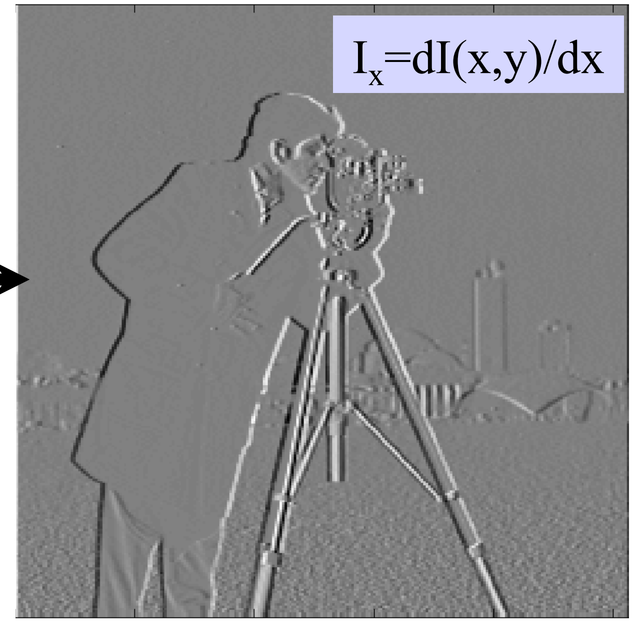
- If magnitude at a pixel exceeds a threshold, report a possible edge point.

# Compute Spatial Image Gradients



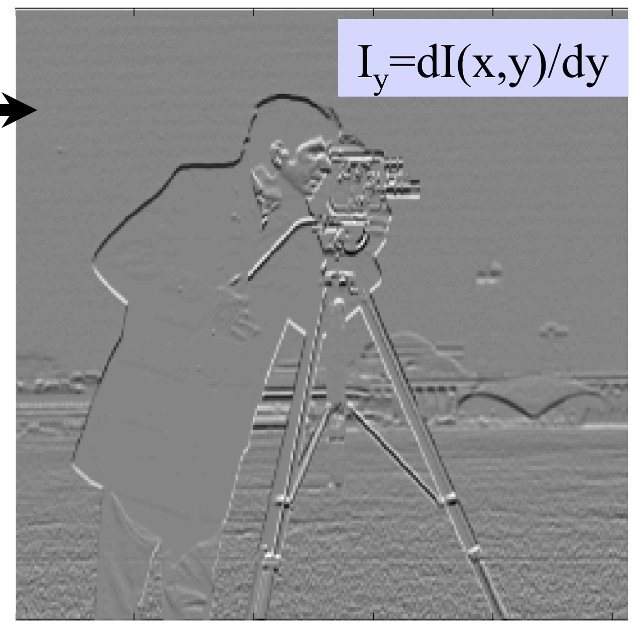

$$\frac{I(x+1,y) - I(x-1,y)}{2}$$

**Partial derivative wrt x**




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

**Partial derivative wrt y**



# Simple Edge Detection Using Gradients

A simple edge detector using gradient magnitude

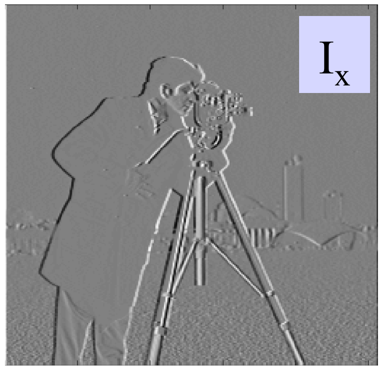
- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters

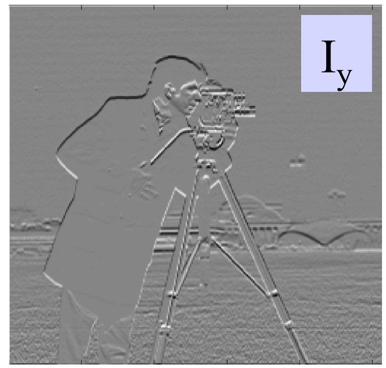
- Compute gradient magnitude at each pixel

- If magnitude at a pixel exceeds a threshold, report a possible edge point.

Robert Collins
CMPEN454

# 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

<u>A simple edge detector using gradient magnitude</u>

- Compute gradient vector at each pixel by convolving image with horizontal and vertical derivative filters

- Compute gradient magnitude at each pixel

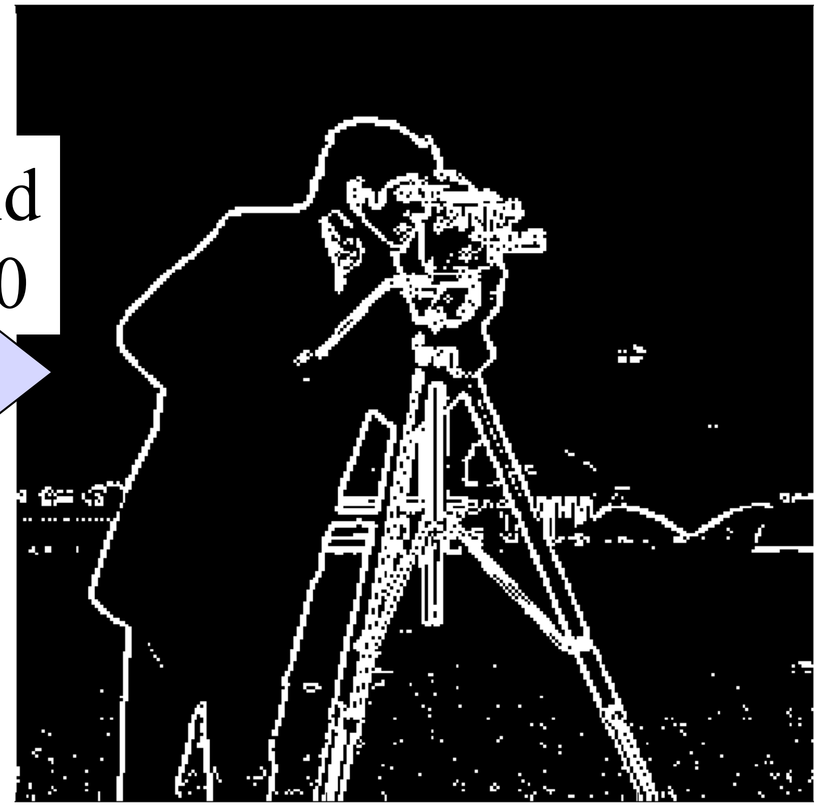
- If magnitude at a pixel exceeds a threshold, report a possible edge point.

# Threshold to Find Edge Pixels

- Example – cont.:

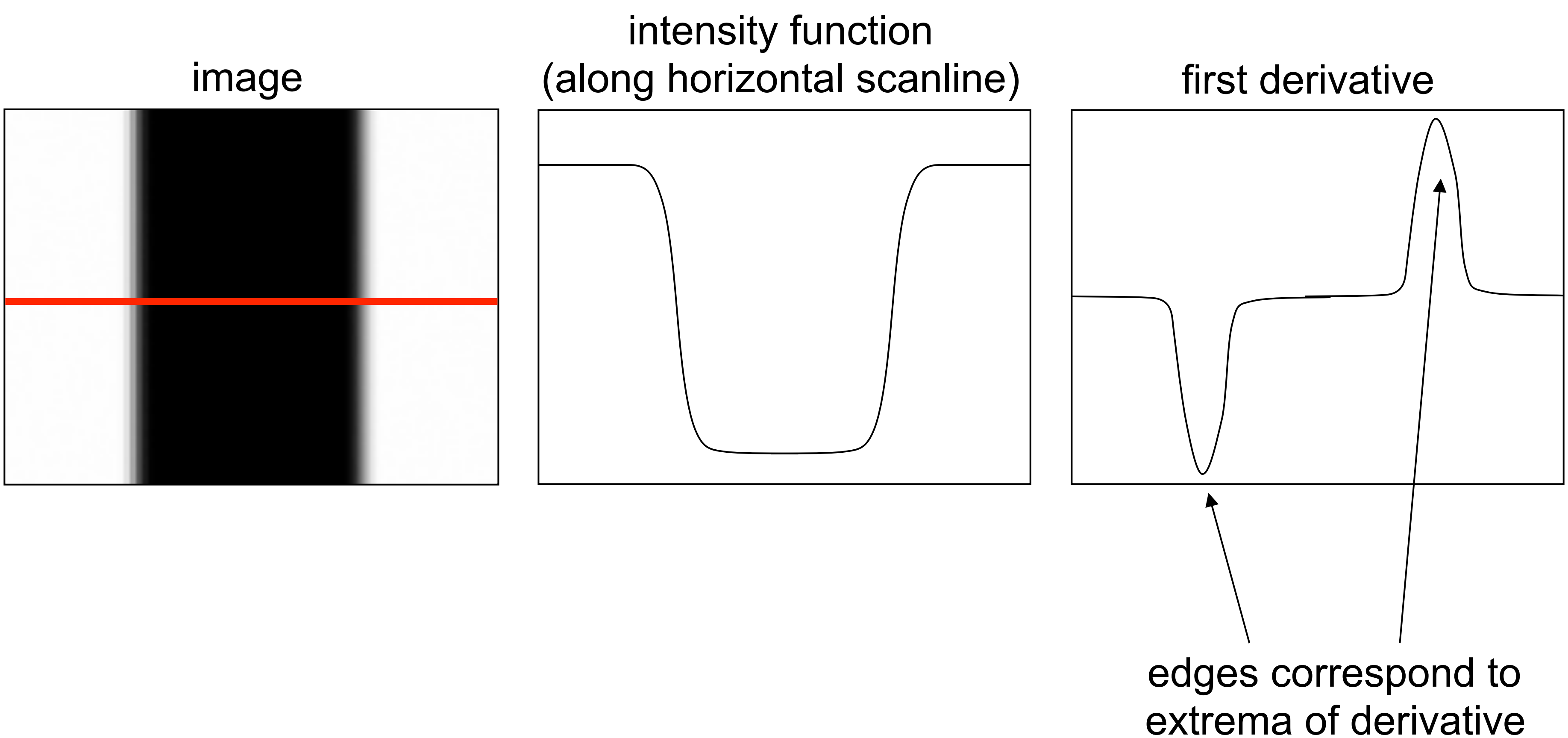
Binary edge image

Threshold
Mag > 30



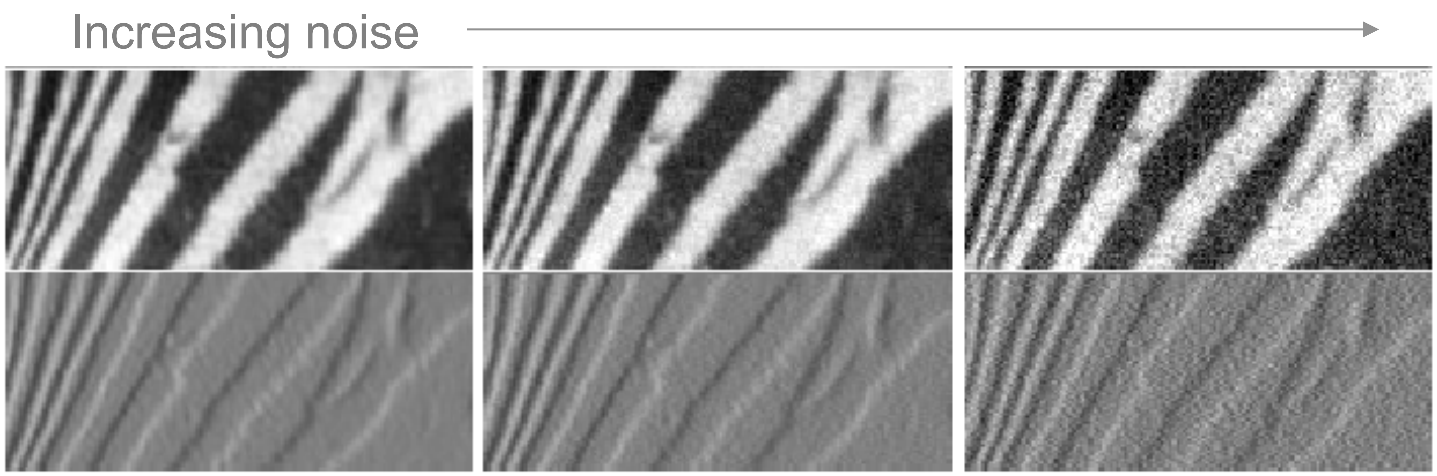
# Characterizing edges

- Locate edges as maxima/minima of first derivative



# Problem: Derivatives and Noise

- derivative operator is affected by noise



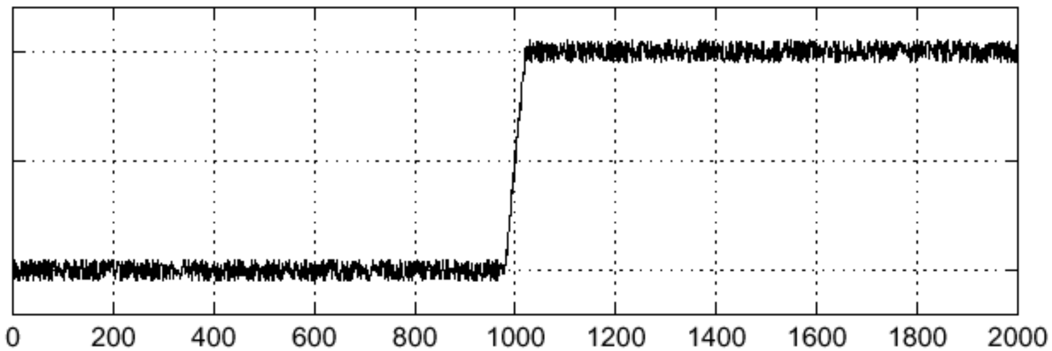

- Numerical derivatives can amplify noise!
  (particularly higher order derivatives)

Due to noise, our computed gradient vectors may be wrong (e.g. incorrect direction and magnitude)!

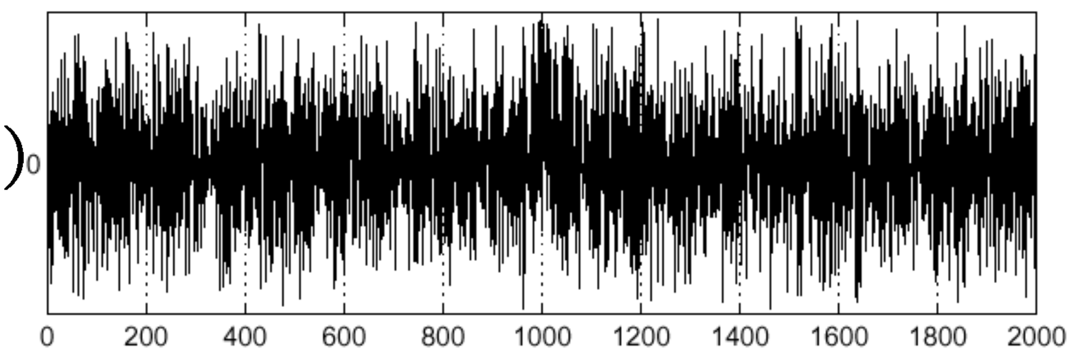
# Effects of noise

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

$f(x)$



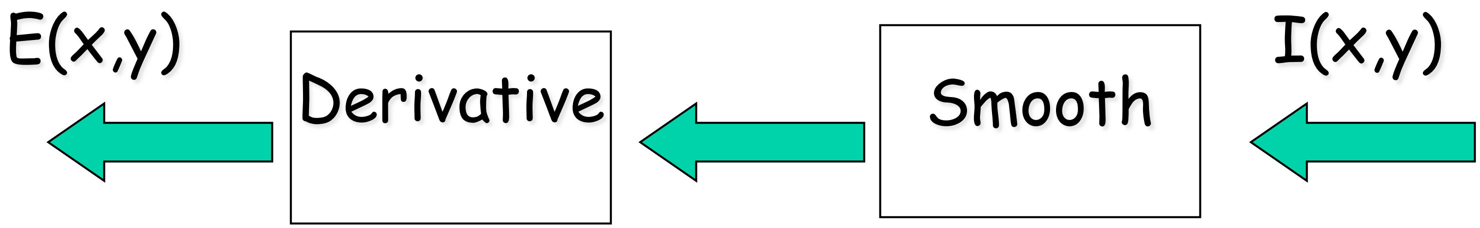

$\frac{d}{dx}f(x)$




Where is the edge?

Source: S. Seitz

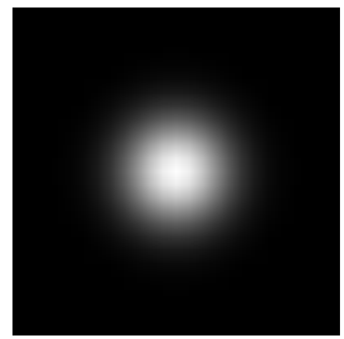
# Solution: Smooth before Applying Derivative Operator!



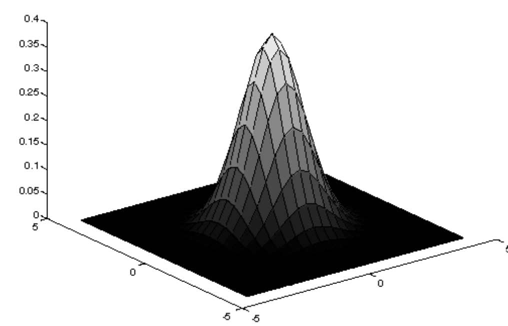

DerivFilter * (SmoothFilter * I)

# Recall: Gaussian Smoothing Filter

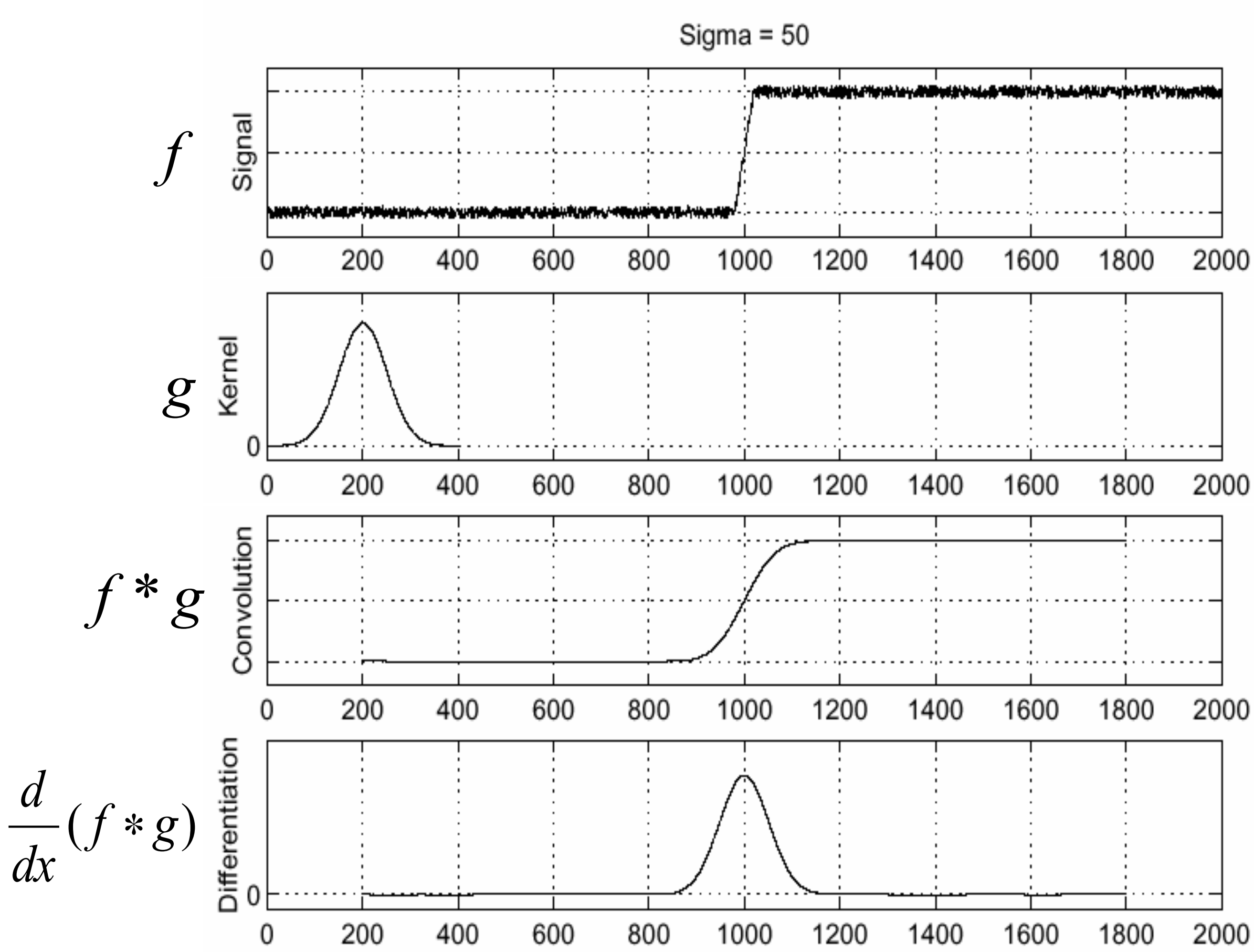
- Smoothing filter that does weighted averaging.
  - The coefficients are a 2D Gaussian.
  - Gives more weight at the central pixels and less weights to the neighbors.
  - The farther away the neighbors, the smaller the weight.



$$G_\sigma \equiv \frac{1}{2\pi\sigma^2}\exp\{-\frac{x^2+y^2}{2\sigma^2}\}$$



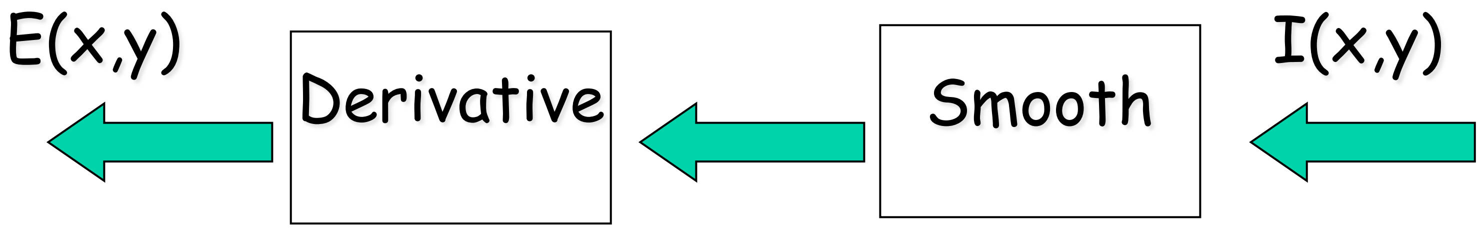
# Solution: smooth first




- To find edges, look for extrema of $\frac{d}{dx}(f * g)$

Source: S. Seitz

# Solution: Smooth before Applying Derivative Operator!




DerivFilter * (SmoothFilter * I)

Question: Do we have to apply two linear operations here (convolutions)?

# Math: Properties of Convolution

Commutative: f * g = g * f

Associative: (f * g) * h = f * (g * h)

Distributive: (f + g) * h = f * h + g * h

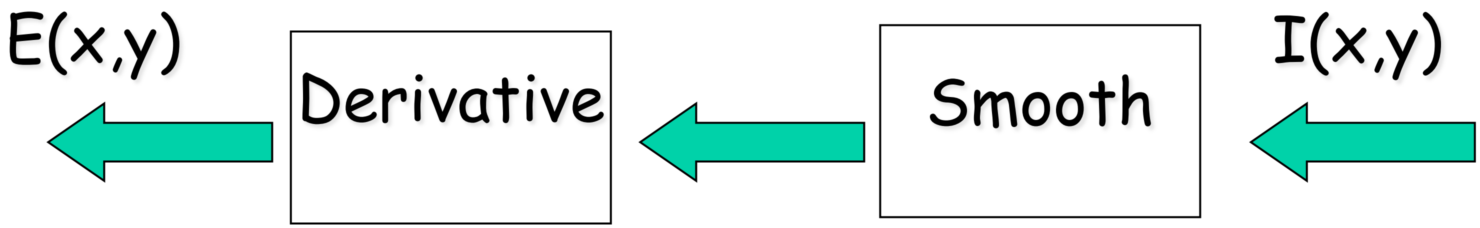
Linear: (a f + b g) * h = a f * h + b g * h

Shift Invariant: f(x+t) * h = (f * h)(x+t)

Differentiation rule:

$$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g$$

# Solution: Smooth before Applying Derivative Operator!




DerivFilter * (SmoothFilter * I)

Question: Do we have to apply two linear operations here (convolutions)?

# Smoothing and Differentiation

No, we can combine filters!

By associativity of convolution operator:

$$\text{DerivFilter} * (\text{SmoothFilter} * \text{I})$$
$$= \underbrace{(\text{DerivFilter} * \text{SmoothFilter})}_{\text{we can precompute this part as a single kernel to apply}} * \text{I}$$

# Recall: Convolution in Matlab

Imfilter(image,template{,option1,option2,…})

Boundary options: constant, symmetric, replicate, circular
Output size options: same as image, or full size (includes partial values computed when mask is off the image).
Corr or conv option: convolution rotates the template (as we have discussed). Correlation does not.

Pro tip: you typically want to use "full" size option when convolving two filters to get another filter.

# Example: Prewitt Edge Operator

Convolve with:

rot180

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

Noise Smoothing

Vertical Edge Detection

This filter is called the (vertical) Prewitt Edge Detector

Note: I am inventing notation here. Rot180 is meant to be used like the transpose operator, but it rotates by 180.

# Example: Prewitt Edge Operator

rot180

Convolve with:

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Noise Smoothing

Horizontal Edge Detection

This filter is called the (horizontal) Prewitt Edge Detector

# Example: Sobel Edge Operator

Convolve with: rot180

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

Gives more weight to the 4-neighbors

and rot180

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

# Important Observation

Note that a Prewitt operator is a box filter convolved with a derivative operator [using "full" option].

Simple box filter

Finite diff operator

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Also note: a Sobel operator is a [1 2 1] filter convolved with a derivative operator.

Simple Gaussian

Finite diff operator

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

# Generalize: Smooth Derivatives

- Solution: First smooth the image by a Gaussian $G_S$ and then take derivatives:

$$\frac{\partial f}{\partial x} \approx \frac{\partial (G_\sigma * f)}{\partial x}$$

- Applying the differentiation property of the convolution:

$$\frac{\partial f}{\partial x} \approx \frac{\partial G_\sigma}{\partial x} * f$$

- Therefore, taking the derivative in x of the image can be done by convolution with the derivative of a Gaussian:

$$G_\sigma^x = \frac{\partial G_\sigma}{\partial x} = xe^{-\frac{x^2+y^2}{2\sigma^2}}$$

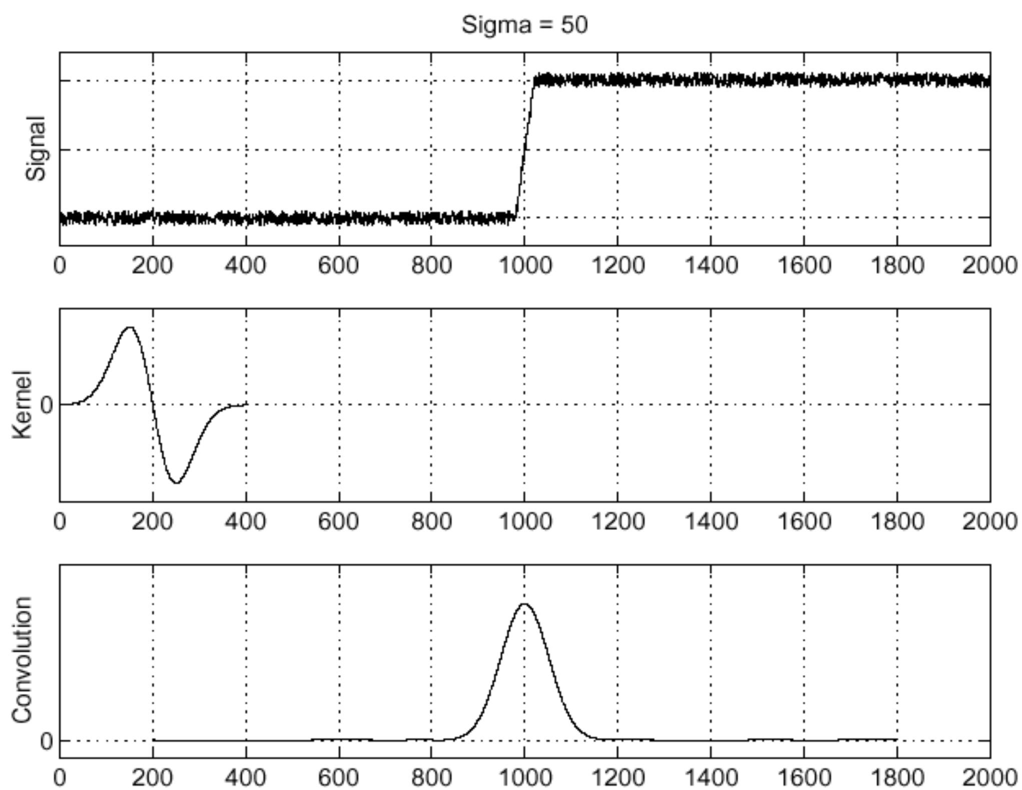
# Derivative theorem of convolution

- Differentiation is convolution, and convolution satisfies the differentiation rule : $\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$
- This saves us one operation:

$f$

$\frac{d}{dx} g$

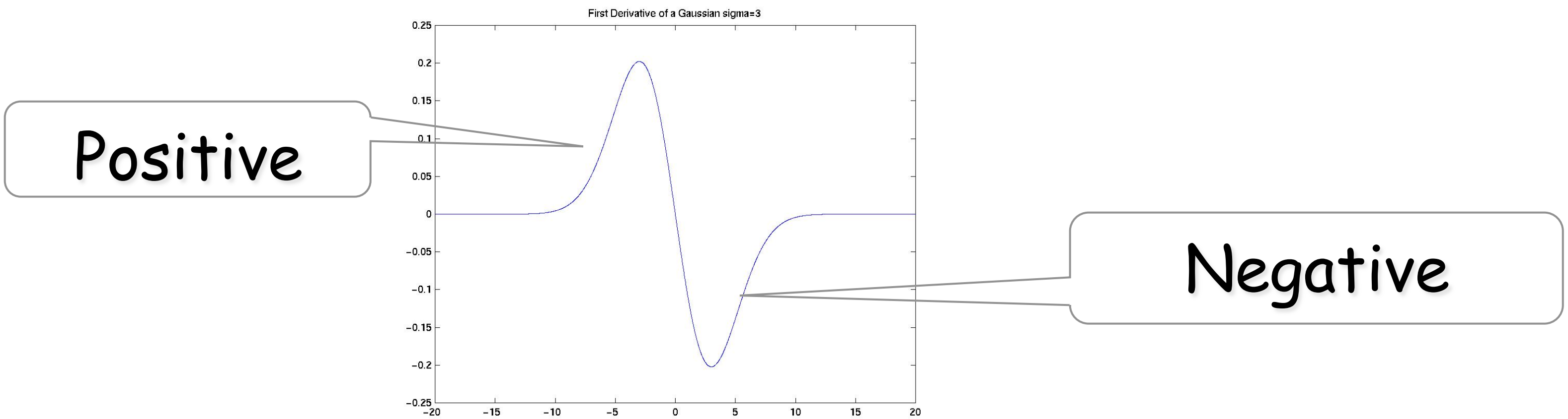
$f * \frac{d}{dx} g$




Source: S. Seitz

# First (partial) Derivative of a Gaussian

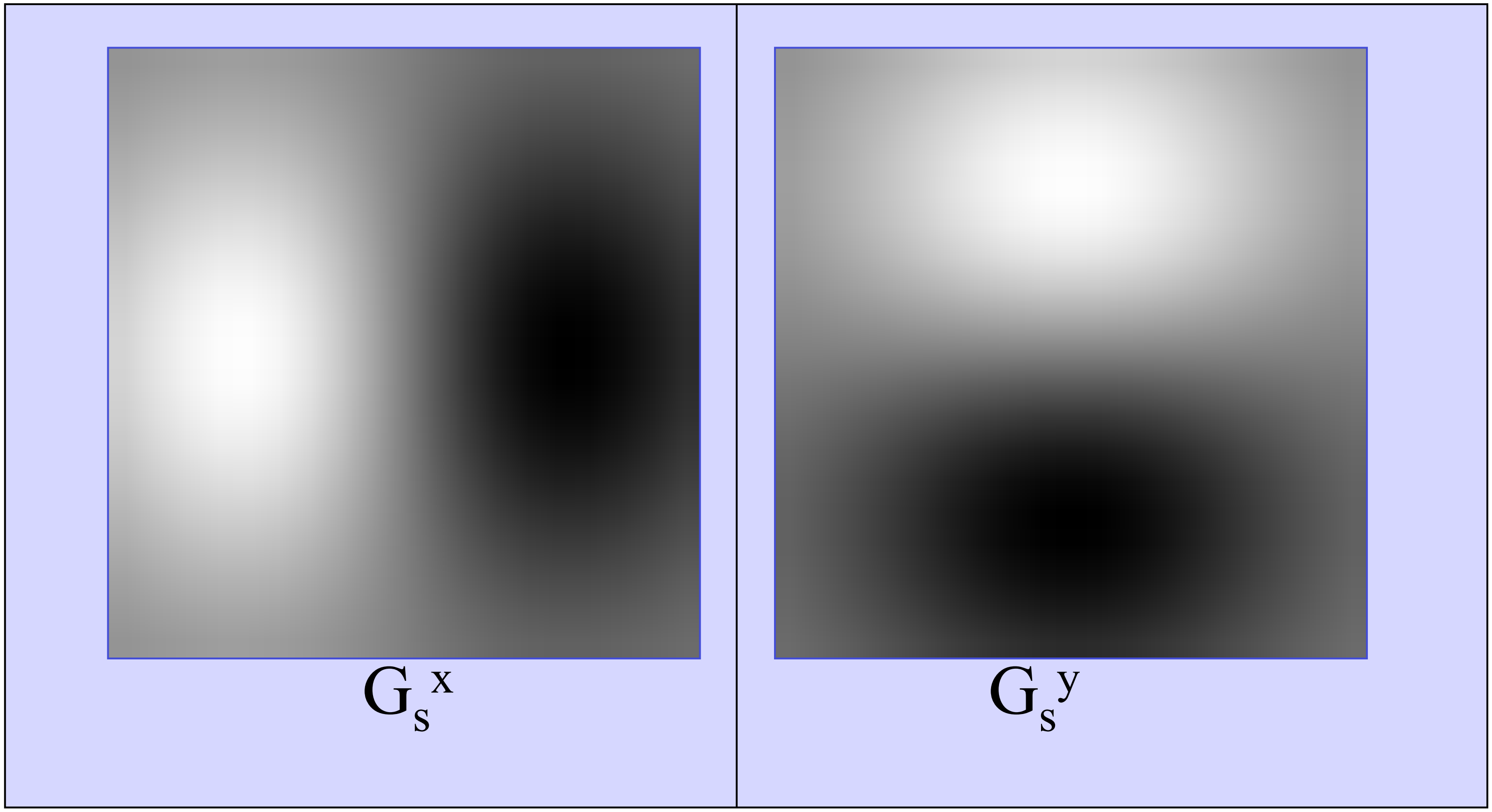
$$g'(x) = -\frac{1}{2\sigma^2} 2x e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$



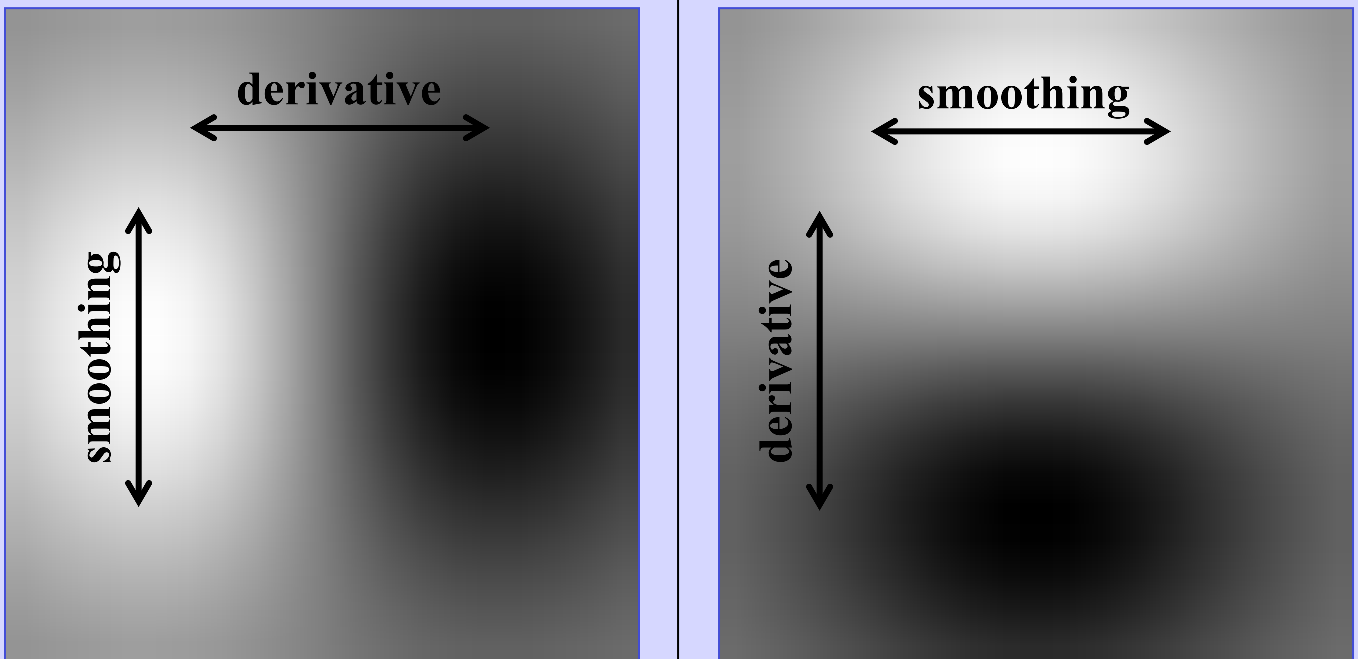

As a filter, it is also computing a difference (derivative)

Robert Collins
CMPEN454

# Derivative of Gaussian Filter in 2D



# Derivative of Gaussian Filter in 2D

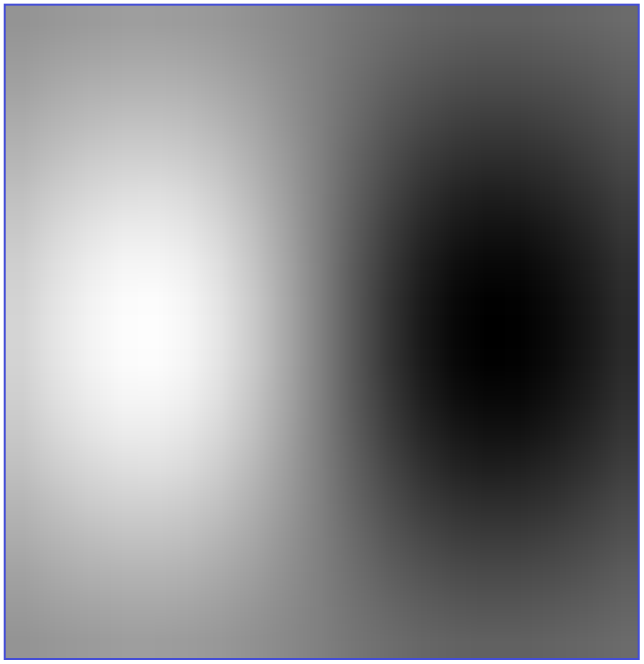



$G_s^x$ $G_s^y$

# Efficient Implementation

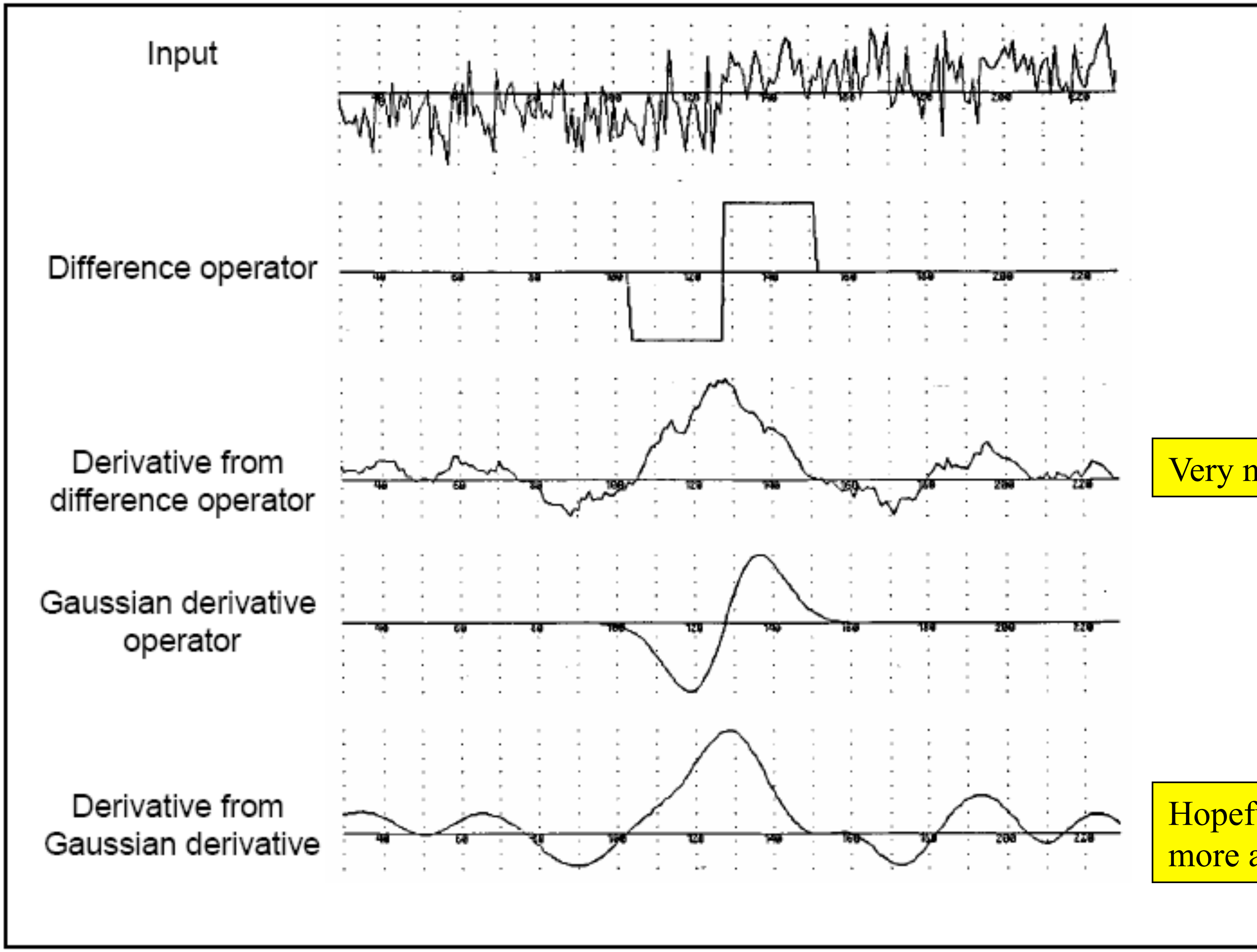
- Since 2D Gaussian filter is separable, derivative of Gaussian filter in 2D can also be separated into two 1D filters
- Example:
  - First convolve each row with a 1D Derivative of Gaussian
  - Then convolve each column with a 1D Gaussian

$$G_\sigma^x * f = g_\sigma^x * g_{\sigma\uparrow} * f$$



Robert Collins
CMPEN454

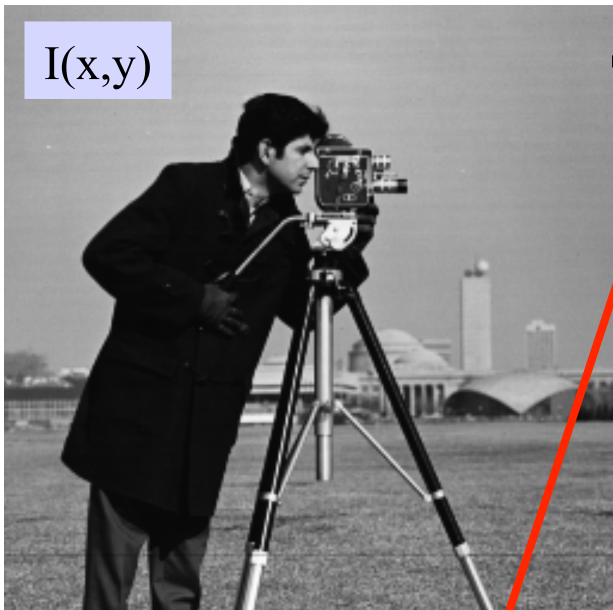
# Summary: Smooth Derivatives




Very noisy

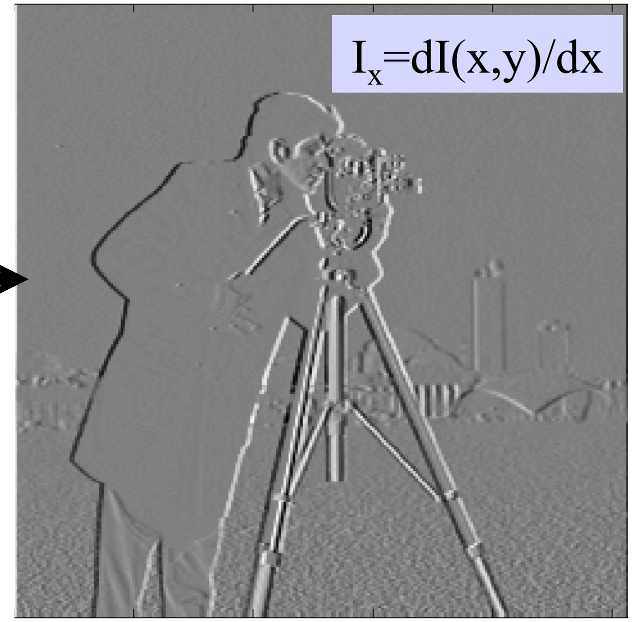
Hopefully more accurate

# Compute Spatial Image Gradients



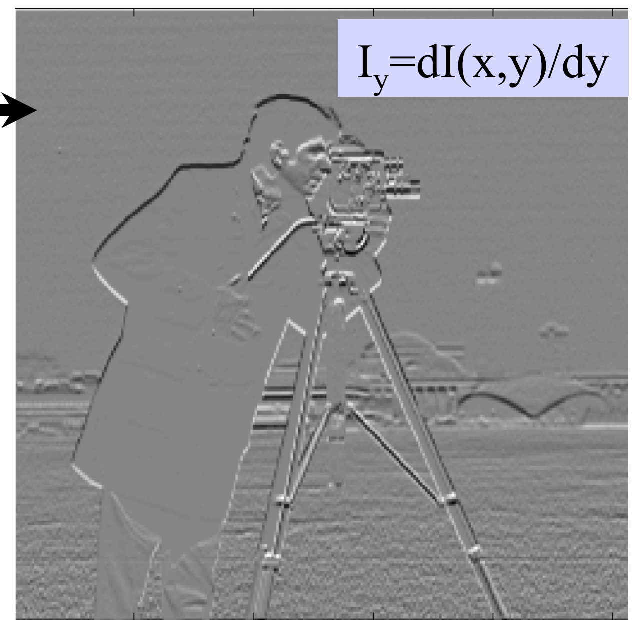

$$\frac{I(x+1,y) - I(x-1,y)}{2}$$

**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