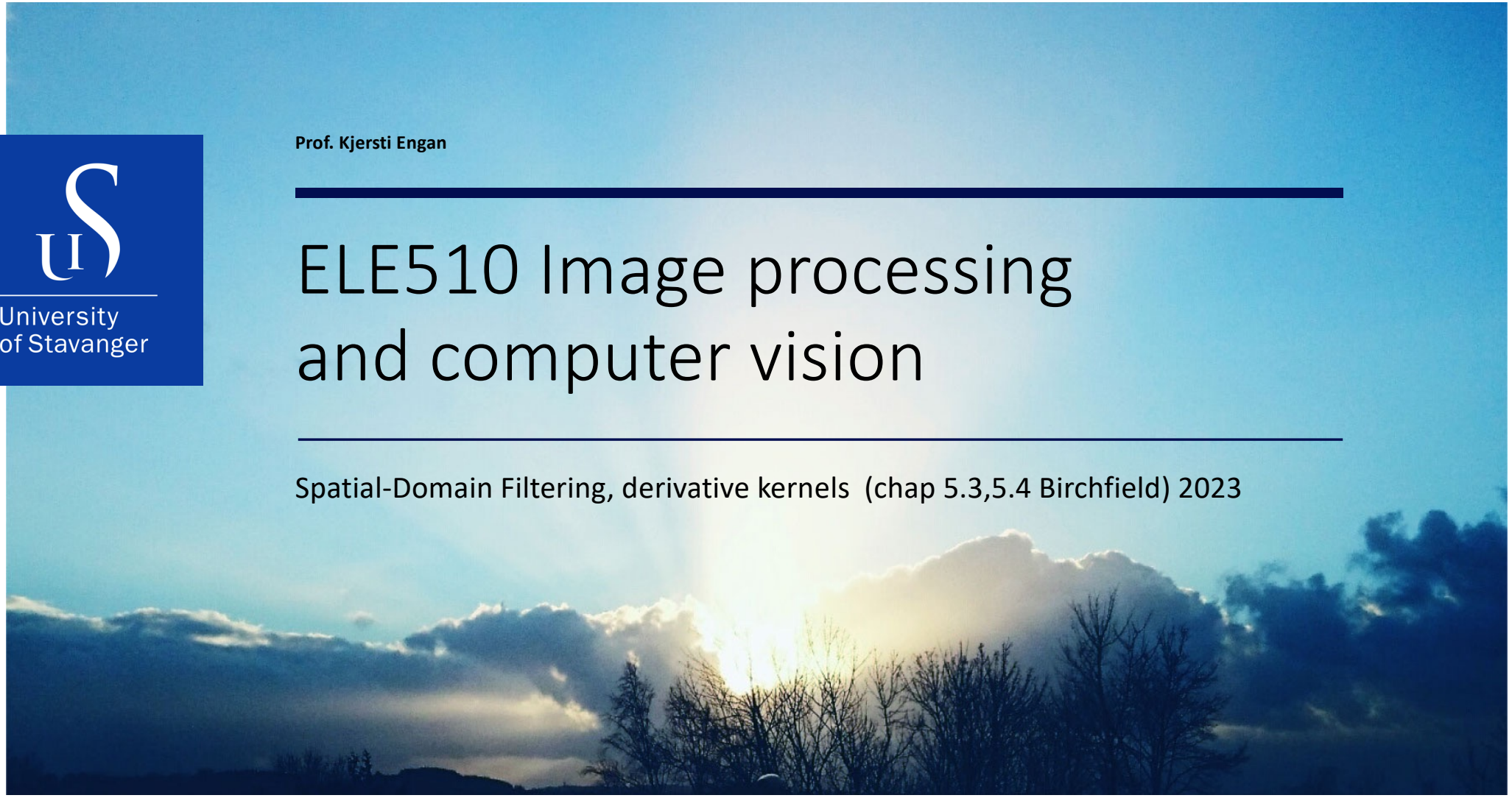


University of Stavanger

Prof. Kjersti Engan

ELE510 Image processing and computer vision

Spatial-Domain Filtering, derivative kernels (chap 5.3,5.4 Birchfield) 2023

Spatial domain filtering – linear filters – derivative kernels (5.3)

Three points from the topic:

1. How is the derivatives (1st. and 2.nd order) connected to important information in images?
2. How (and why) can we smooth before taking derivatives?
3. How can we do edge detection?

Noise

- Additive noise: In(x,y) = I(x,y) +n(x,y)
- **Additive gaussian noise**: the noise follows a random gaussian distribution with some sigma and typically mean=0.
- **Salt-and-pepper noise**: each pixel is set to either the minimum ("pepper") or maximum ("salt") possible gray level, or it remains unchanged:

$$I'(x,y) = \begin{cases} 0 & \text{if } 0 \le \xi < p \\ 255 & \text{if } p \le \xi < p+q \\ I(x,y) & \text{otherwise} \end{cases} \qquad \xi \sim U(0,1)$$

(5.3) Derivative Kernels

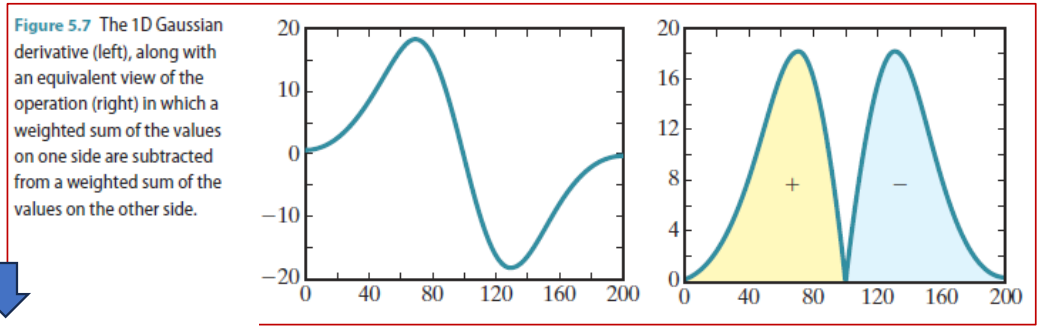
- We have talked about Gaussian filters and box filters, which are smoothing filters.
- Opposite of smoothing is enhancing edges/ local differences, and compute derivatives.
- The simplest approach to estimating the derivative is to compute **finite differences**.
  - Subtract one value in the signal from another.
  - Equivalent to convolving with the kernel [1 -1]

All signals and images have some noise. Therefore it is a good strategy to first smooth the signal somewhat, thereafter find the derivative.

Gaussian is most used smoothing filter -> Derivative of Gaussian

$$\frac{d}{dx} gauss_{\sigma^2}(x) = \frac{d}{dx}\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-x^2}{2\sigma^2}}\right)$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}}\left(\frac{-2x}{2\sigma^2}\right) \cdot e^{\frac{-x^2}{2\sigma^2}} = \frac{-x}{\sigma^2} gauss_{\sigma^2}(x) = \dot{g}auss_{\sigma^2}(x)$$

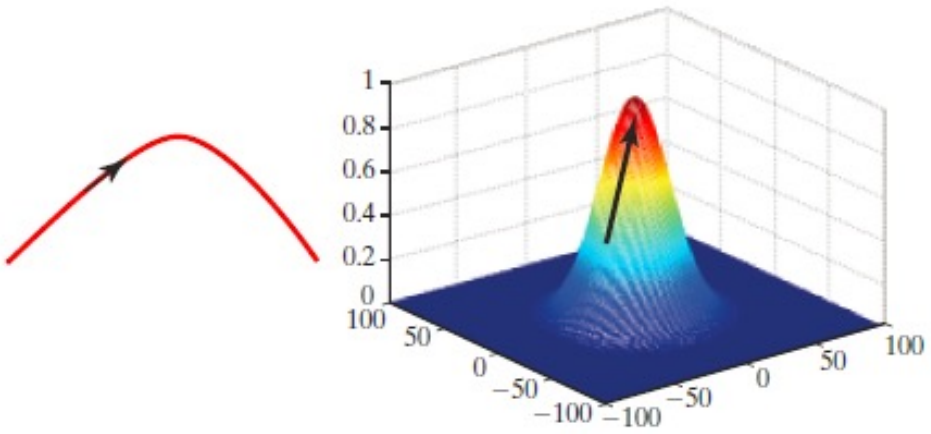


To construct a derivative kernel: sample the continous gaussian derivative and *normalize*.
Normalization: convolution with a ramp should give the slope of the ramp (the derivative!)

Image gradient

- The generalization of the derivative to 2D is the **gradient**.
- The vector whose elements are the partial derivatives of the function along the two axes:

$$\nabla f(x, y) = \left[\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}\right]^{\mathsf{T}}$$



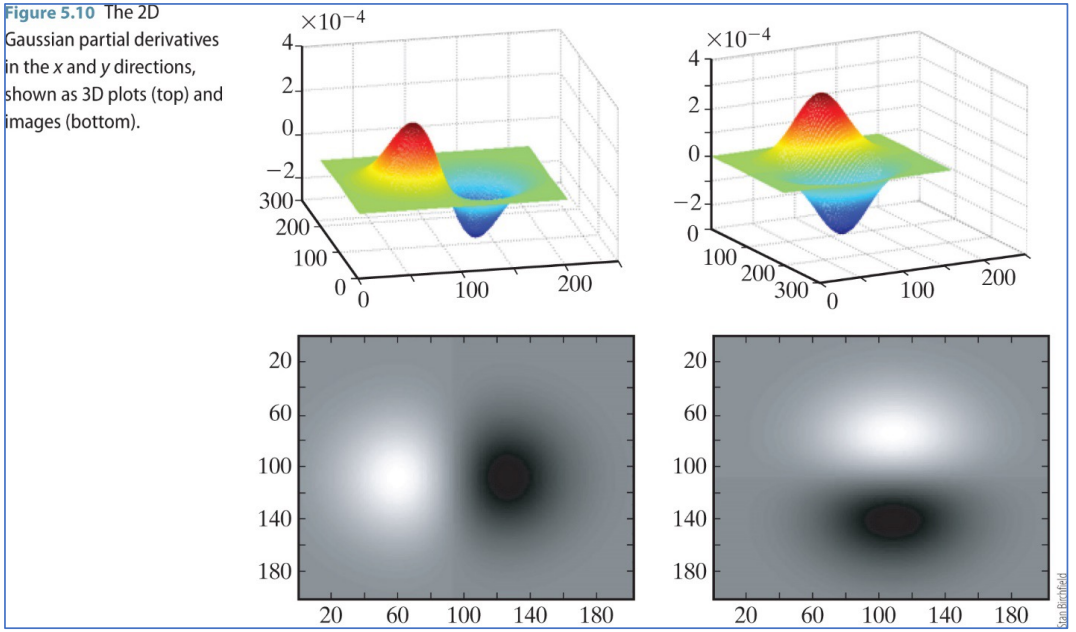


Figure 5.10 The 2D Gaussian partial derivatives in the *x* and *y* directions, shown as 3D plots (top) and images (bottom).

Gaussian in 2D:

$$Gauss(x, y) = C \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

1st. derivative in x and y direction, gradients:

$$g_x(x, y) = C_2 \cdot x \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

$$g_y(x, y) = C_3 \cdot y \cdot e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

What if we convolve with a gaussian to reduce noise, thereafter partial derivative (as in 1D example) :

- Once we have computed the gradient of the image it is often desirable to compute the magnitude of the gradient and the phase of the gradient.

$$|\nabla f| = \sqrt{f_x^2 + f_y^2} \approx |f_x| + |f_y| \approx max(|f_x|, |f_y|)$$

Euclidean manhattan chessboard

$$\phi(x, y) = arctan \frac{f_y}{f_x}$$

Prewitt operator

- The simplest 2D differentiating kernel is the **Prewitt operator**.
  - Obtained by convolving a 1D Gaussian derivative kernel with a 1D box filter in the orthogonal direction:

$$Prewitt_x = \frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} \circledast \frac{1}{2}\begin{bmatrix}1 & 0 & -1\end{bmatrix} = \frac{1}{6}\begin{bmatrix}1 & 0 & -1\\1 & 0 & -1\\1 & 0 & -1\end{bmatrix}$$

$$Prewitt_y = \frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} \circledast \frac{1}{2}\begin{bmatrix}1\\0\\-1\end{bmatrix} = \frac{1}{6}\begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\-1 & -1 & -1\end{bmatrix}$$

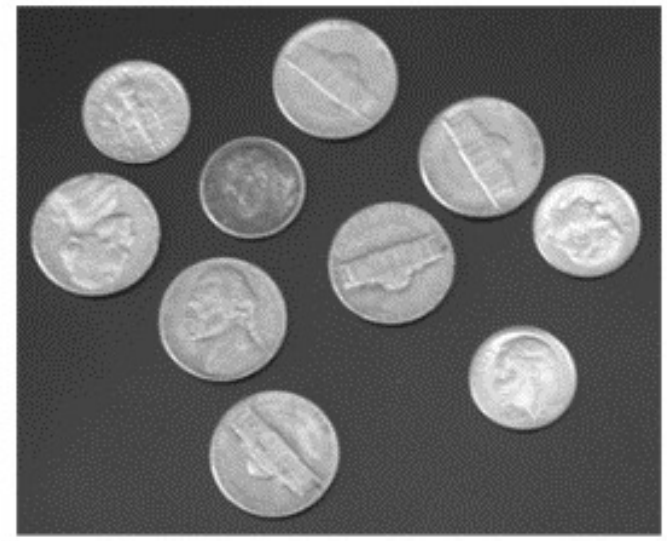
Sobel operator

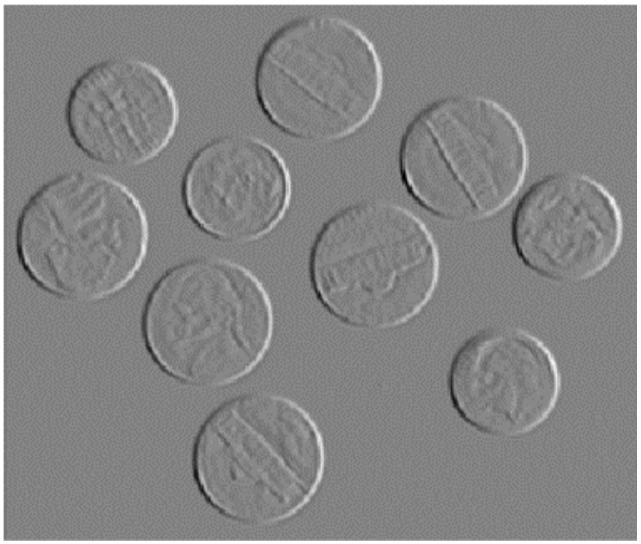
- The **Sobel operator** is more robust, as it uses the Gaussian ($\sigma^2 = 0.5$) for the smoothing kernel:

$$Sobel_x = gauss_{0.5}(y) \circledast \dot{g}auss_{0.5}(x) = \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} \circledast \frac{1}{2}[1 \quad 0 \quad -1] = \frac{1}{8}\begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix}$$

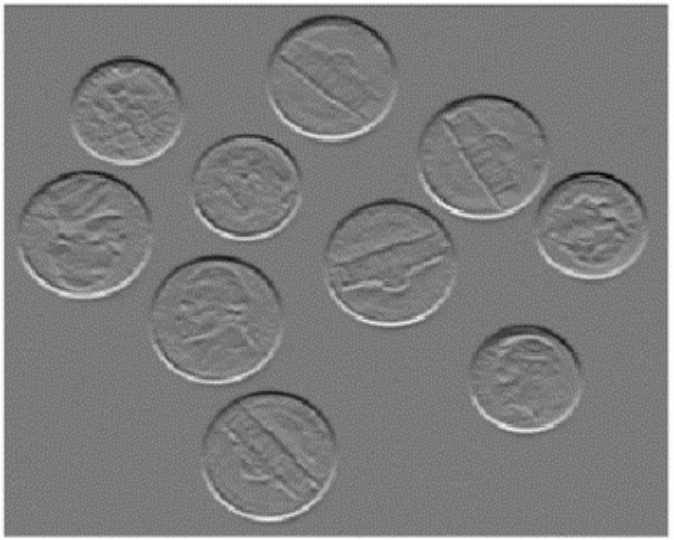
$$Sobel_y = gauss_{0.5}(x) \circledast \dot{g}auss_{0.5}(y) = \frac{1}{4}[1 \quad 2 \quad 1] \circledast \frac{1}{2}\begin{bmatrix}1\\0\\-1\end{bmatrix} = \frac{1}{8}\begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix}$$

Sobel operator





Δfx(i,j) Smooths over columns, derivative over rows.



Δfy(i,j) Smooths over rows, derivative over columns.

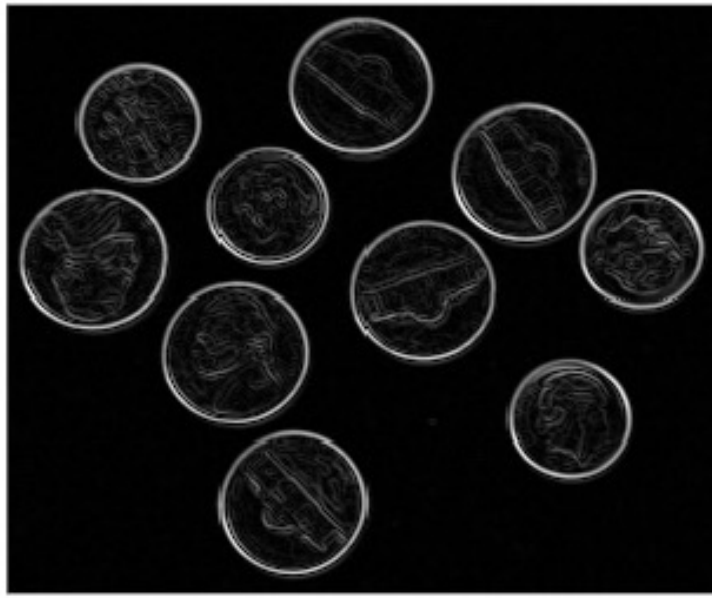
$$G(i,j) = \sqrt{\Delta f_x(i,j)^2 + \Delta f_y(i,j)^2}$$

Gradient image

$$\theta(i,j) = \operatorname{atan} \frac{\Delta f_x(i,j)}{\Delta f_y(i,j)}$$

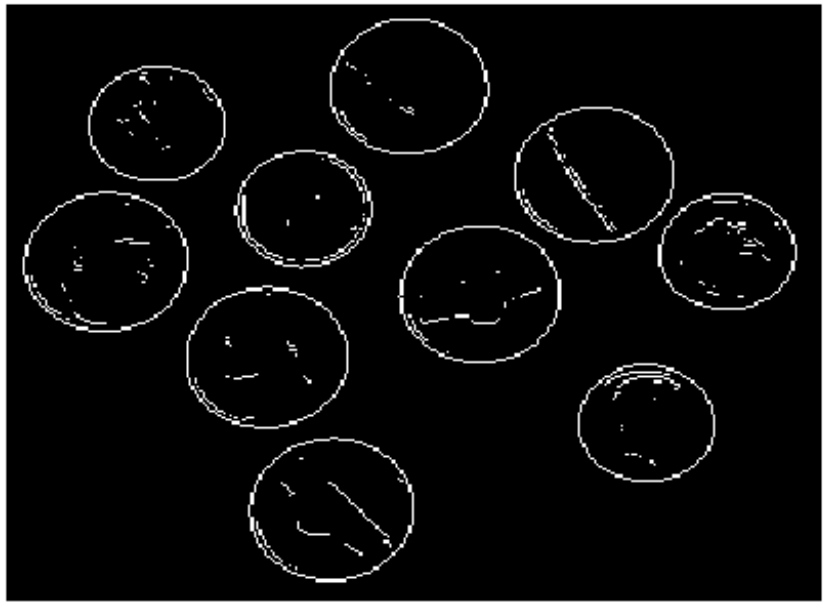
Angle image

Sobel – gradient image and edge map



$$G(i,j) = \sqrt{\Delta f_x(i,j)^2 + \Delta f_y(i,j)^2}$$

Magnitude of the gradient
(Euclidean def.)



Combining thresholded gradient image with angle information can give binary image/edge map.

Example: used in Canny – edge detection

1. Image is smoothed by Gaussian filter to reduce noise
2. Local gradient and edge direction are found (by sobel / prewitt)
3. The ridges of the gradient image is tracked, set to zero all pixels not on the ridge top -> thin line Thereafter, hysteresis threshold

   Ridge pixel > T2 -> strong edge pixel

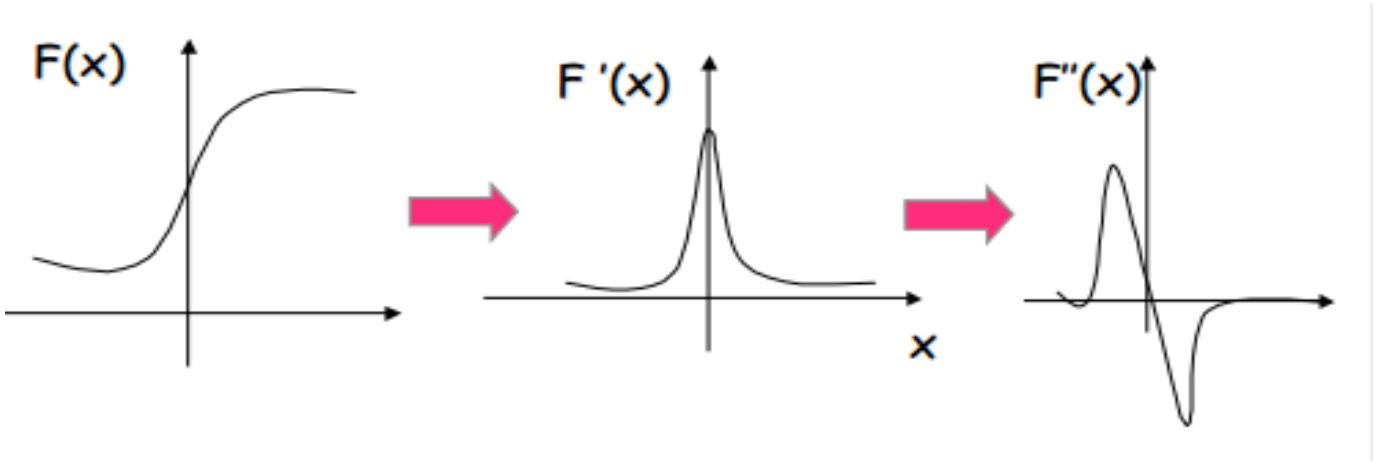
   T1<Ridge pixel<T2 -> weak edge pixel

4. Edge linking -> include weak edge pixels that are 8- connected to strong edge pixels.

Second derivative

- Recall Sharp changes in gray level of the input image correspond to “peaks or valleys” of the first-derivative of the input signal.

- Peaks or valleys of the first-derivative of the input signal, correspond to “zero-crossings” of the second-derivative of the input signal.



Second derivative

- Finding the second derivative can be seen in 1D by convolving the function with the noncentralized difference operator twice:

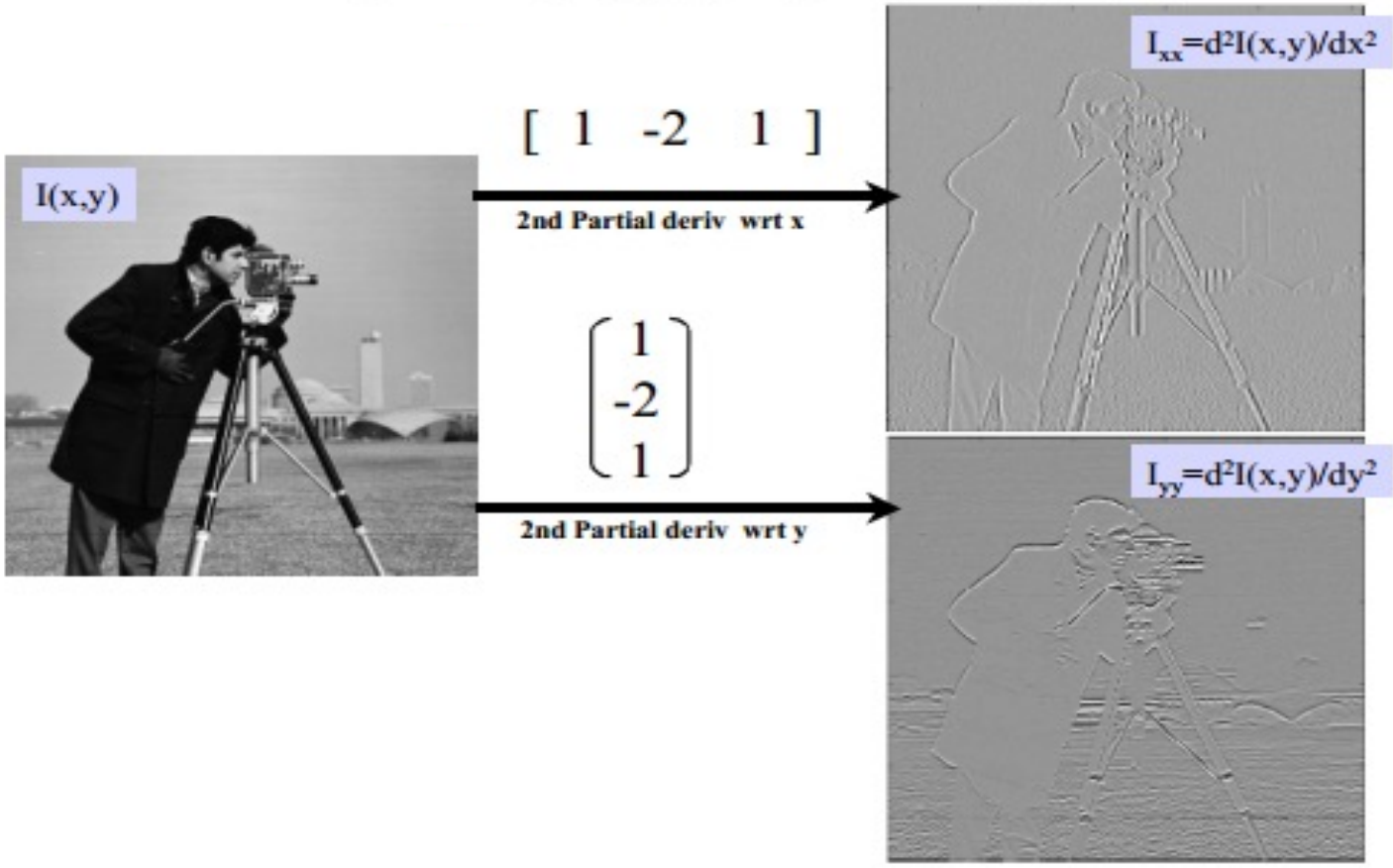
$$(f(x) \circledast [1 \quad -1]) \circledast [1 \quad -1] = f(x) \circledast ([1 \quad -1] \circledast [1 \quad -1]) = f(x) \circledast [1 \quad -2 \quad 1]$$

- In 2D, the second-derivative in the *x* and *y* directions can be obtained by convolving with the second-derivative kernel:

$$\frac{\partial^2 I(x,y)}{\partial x^2} = I(x,y) \circledast [1 \quad -2 \quad 1]$$

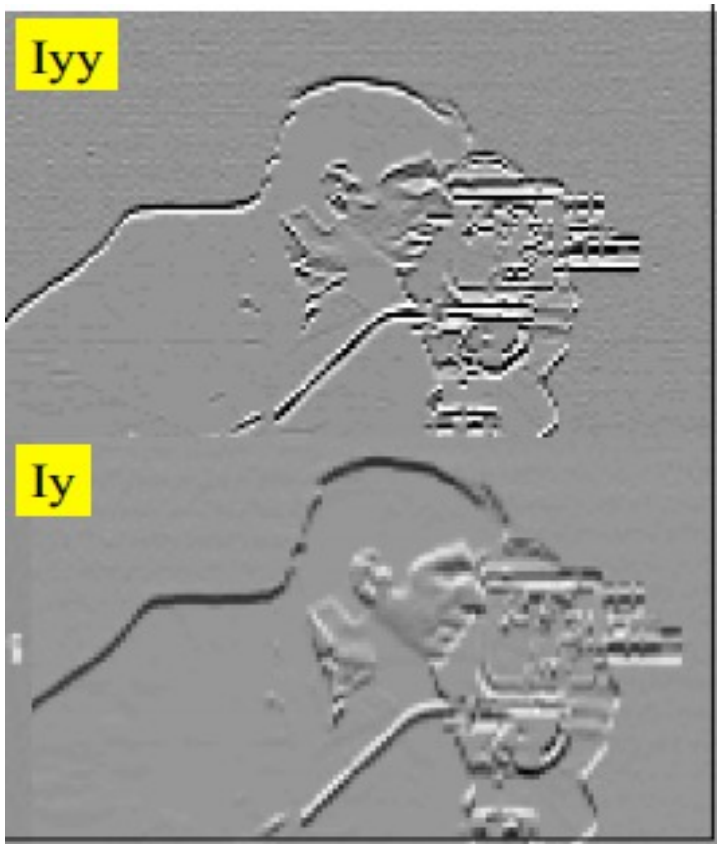
$$\frac{\partial^2 I(x,y)}{\partial y^2} = I(x,y) \circledast \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

Second derivatives and convolution



Second derivatives and convolution

- Better localized edges
- But more sensitive to noise



Laplacian filter

- Taking the second derivative of a function (image) gives a zero-crossing at and edge.

- The Laplacian of an image f can be find as:

$$\Delta f = \nabla \cdot \nabla f = \nabla^2 f$$

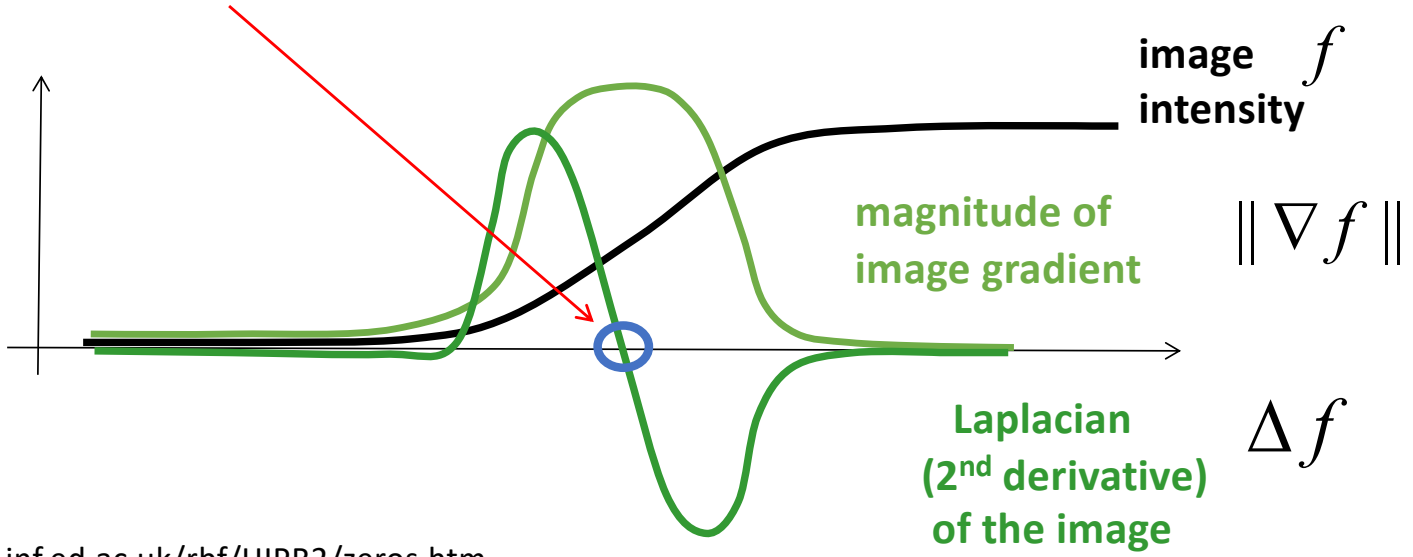
$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Second Image Derivatives

- Laplacian Zero Crossing

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

- Used for edge detection
  (alternative to computing Gradient extrema)



http://homepages.inf.ed.ac.uk/rbf/HIPR2/zeros.htm

Laplacian filtering

- Laplace operator – approximation filter mask

rotationally invariant
second derivative for 2D functions:

0	0	0
-1	2	-1
0	0	0

Finite Difference Second Order Derivative in x

+

0	-1	0
0	2	0
0	-1	0

Finite Difference Second Order Derivative in y

=

0	-1	0
-1	4	-1
0	-1	0

But problems with noisy images.

Laplacian filtering

[1 -2 1] -> Ixx

[1 -2 1]´ -> Iyy

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} * I$$

-Zero on uniform regions
-Positive on one side of an edge
-Negative on the other side
-Zero at some point in between
on the edge itself
➔ band-pass filter (Suppresses both high and low frequencies)

http://www.cse.psu.edu/~rcollins/CSE486/lecture11_6pp.pdf

Laplacian – some remarks

- Can be found using a single mask (1.st derivative needs two)
- Orientation is lost
- Taking derivatives increase noise
  - Second derivative is very noise sensitive!
- Should be combined with a smoothing ….

  if we first do gaussion smoothing -> LoG filter

Laplacian of Gaussian (LoG)

- **Laplacian operator ∇^2**: defined as the divergence of the gradient of a function.

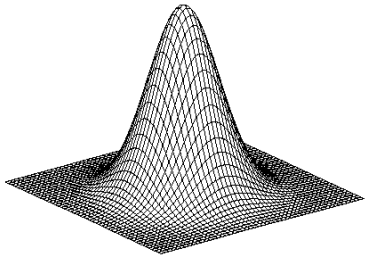
$$\nabla^2 I = \nabla \cdot \nabla I = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \end{bmatrix}^{\mathsf{T}} = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

- Computing the Laplacian of a smoothed image is the same as convolving the image with the **Laplacian of Gaussian (LoG):**

$$\frac{\partial^2 (I \circledast Gauss(x,y))}{\partial x^2} + \frac{\partial^2 (I \circledast Gauss(x,y))}{\partial y^2} = I \circledast \left(\frac{\partial^2 Gauss(x,y)}{\partial x^2} + \frac{\partial^2 Gauss(x,y)}{\partial y^2} \right)$$

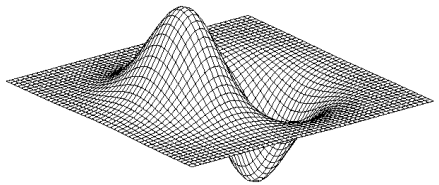
- LoG also called Mexican hat
- 2D Gaussian second derivative is separable

Laplacian of Gaussian (LoG)



Gaussian

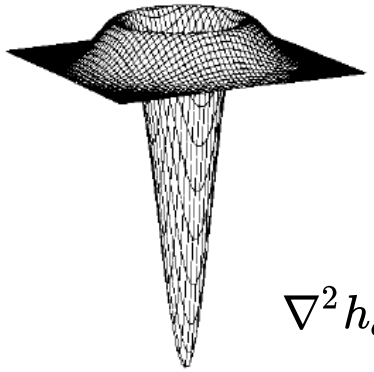
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



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:

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

Laplacian of Gaussian (LoG)

- Gaussian function: $$G_0(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$
- Disregard scaling: $$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$
- Laplacian of function f(x,y): $$\Delta f(x,y) = \nabla^2 f(x,y) = \frac{d^2 f(x,y)}{dx^2} + \frac{d^2 f(x,y)}{dy^2}$$
- LoG: $$\nabla^2 G(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Laplacian of Gaussian (LoG) 3x3 kernels

The only possible for 3x3 Gaussian 1. order differenting kernel is ½ [1 0 -1]. The corresponding only 2.order gauss diff. 3x3 kernel is [1 -2 1]

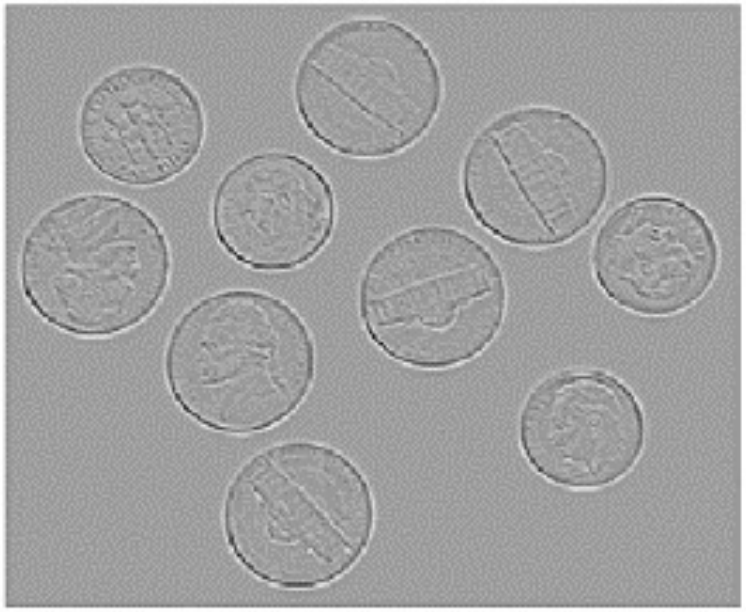
However, different smoothing functions (Gaussian with different varians) in the LoG gives different LoG kernels:

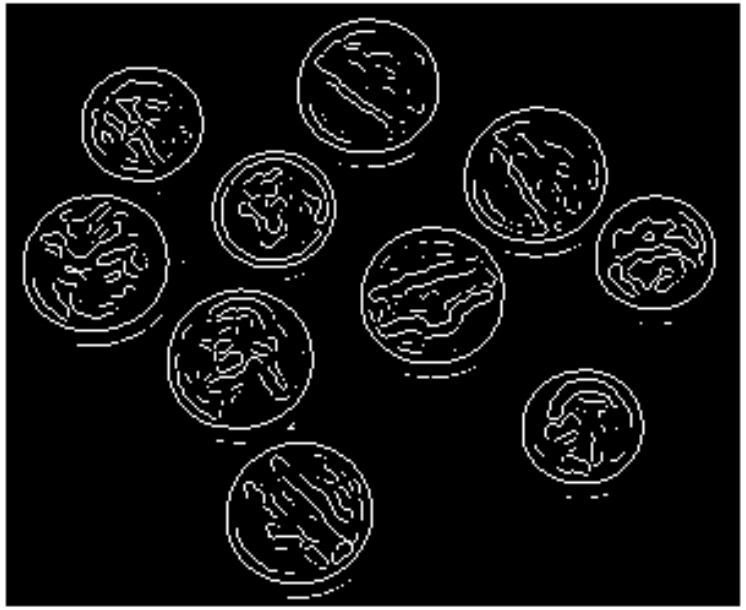
$\sigma^2 = 0.0$	$\sigma^2 = 0.167$	$\sigma^2 = 0.20$	$\sigma^2 = 0.25$	$\sigma^2 = 0.33$	$\sigma^2 = 0.5$
$[0 \quad 1 \quad 0]$	$\frac{1}{12}[1 \quad 10 \quad 1]$	$\frac{1}{10}[1 \quad 8 \quad 1]$	$\frac{1}{8}[1 \quad 6 \quad 1]$	$\frac{1}{6}[1 \quad 4 \quad 1]$	$\frac{1}{4}[1 \quad 2 \quad 1]$
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$	$\frac{1}{5}\begin{bmatrix} 1 & 3 & 1 \\ 3 & -16 & 3 \\ 1 & 3 & 1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix}$

TABLE 5.5 Various discrete 3 × 3 LoG kernels. For each choice of variance, the middle row shows the 1D smoothing kernel, while the last row shows the resulting LoG kernel.

LoG as edge detector

LoG gives doble edge image. Can use zero-crossings to find edges.

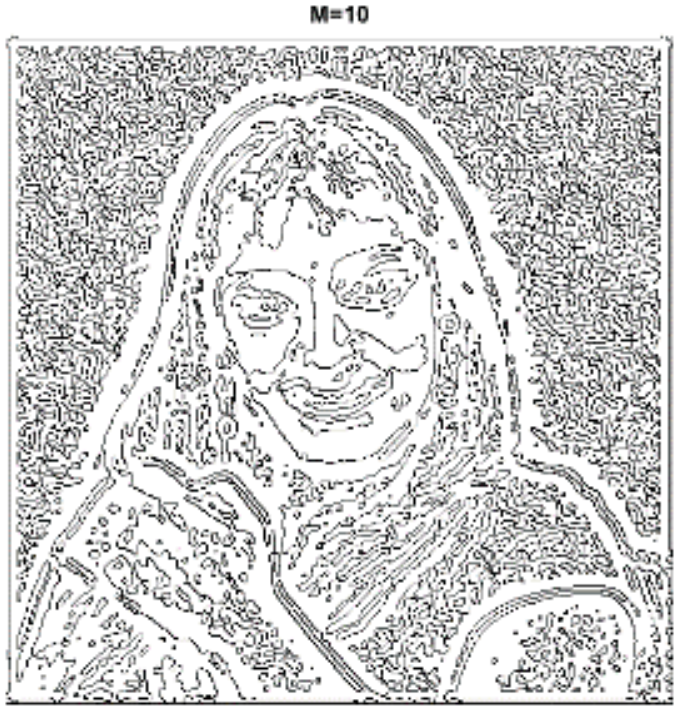




Matlab: edge(Im,'log',T,sigma)

T – threshold on edges. sigma – σ of Gaussian filter

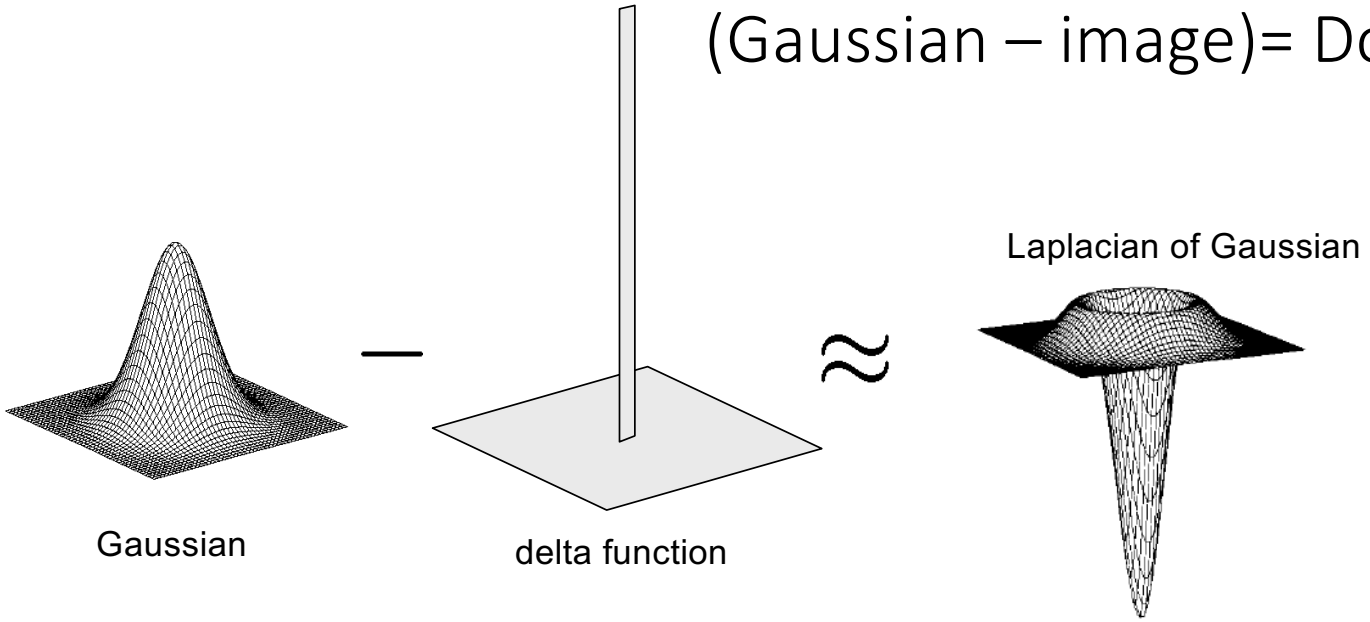
LoG – closed contours



If edge map defined from zero-crossings at LoG output, we get closed contours.

Difference of Gaussian (DoG)

(Gaussian – image)= DoG $\approx$ LoG of image



Edge detection by subtraction (DoG)

original

smoothed (5x5 Gaussian)

DoG=(smoothed-original)+128

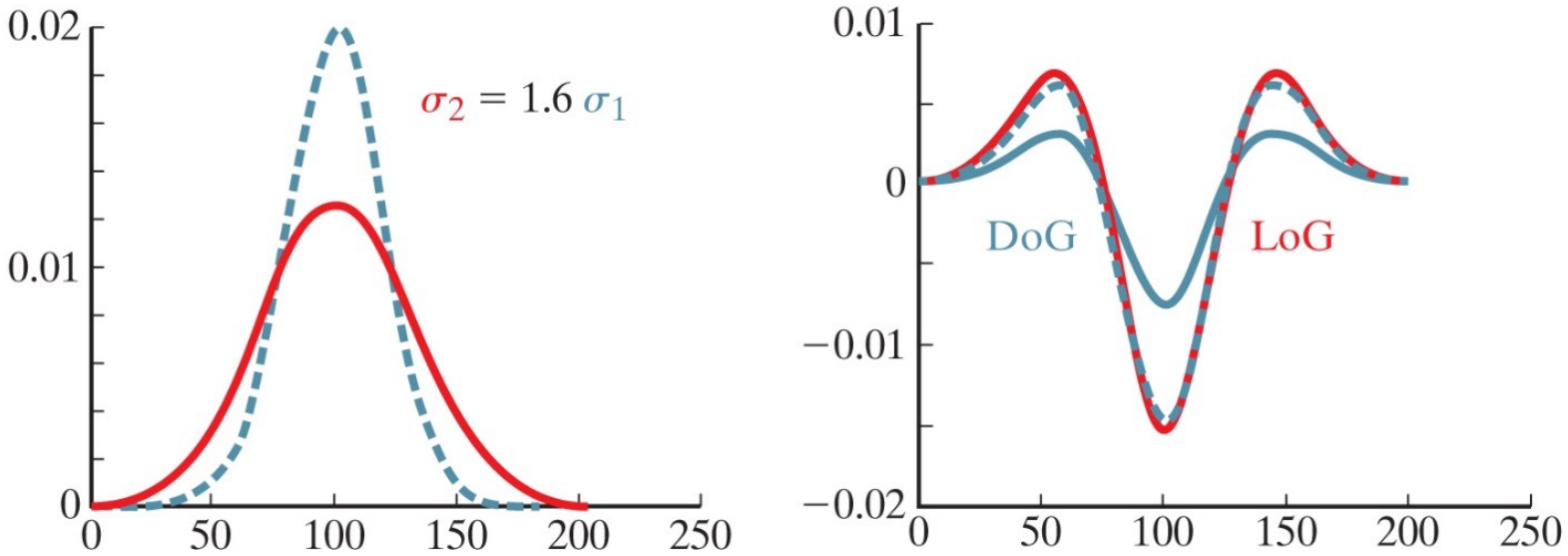


Figure 5.15 LEFT: Two Gaussians whose ratio of standard deviations is 1.6. RIGHT: The difference of Gaussians (solid blue) and 1D Laplacian of Gaussian (solid red). The scaled DoG (dashed blue) approximates the LoG.

LoG can be approximated by a difference between two Gaussians at different scales

Best approx. when $\sigma_1 = \dfrac{\sigma}{\sqrt{2}},\ \sigma_2 = \sqrt{2}\sigma$

Spatial domain filtering – linear filters – derivative kernels (5.3)

Three points from the topic:

1. How is the derivatives (1st. and 2.nd order) connected to important information in images?
   - ✓ High frequencies – texture and edges – will produce high values on derivatives

2. How (and why) can we smooth before taking derivatives?
   - ✓ Why: because noise can cause a lot of unwanted effects in derivatives. How: smooth the derivative filtermask first, producing a new filter: (sobel, laplacian of Gaussian)

3. How can we do edge detection?
   - ✓ Threshold gradient images peaks from 1. derivatives (sobel), zerocrossing from 2. derivatives (laplacian) .
   - ✓ Maybe add some edge linking (Canny) or morphological operations (opening and closing) to connect edges and remove smaller things.