EEE-6512: Image Processing and Computer Vision

October 11, 2017

Lecture #7: Spatial Domain Filtering

Dept. of Electrical and Computer Engineering

dwoodard@ece.ufl.edu

Chapter Outline

- Convolution
- Smoothing by Convolving with a Gaussian
- Computing the First Derivative
- Computing the Second Derivative
- Nonlinear Filters
- Grayscale Morphological Operators

2D Convolution

- **2D Convolution:** used to perform filtering on a 2D image.

$$I'(x,y) = I(x,y) \circledast G(x,y) = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} I(x+\tilde{w}-i, y+\tilde{h}-j) G(i,j)$$

where *w* and *h* are the width and height of the kernel, respectively.

For each pixel in the image

Align the filter over the image pixel and sum over all pixels in the filter

Multiply the filter pixel value * image pixel value

Averaging Filter: An Intuitive Approach

- Replace each pixel by the average of pixels in a square window surrounding this pixel

$$g(m,n) = \frac{1}{9}(f(m-1,n-1) + f(m-1,n) + f(m-1,n+1) + f(m,n-1) + f(m,n) + f(m,n+1) + f(m+1,n-1) + f(m+1,n) + f(m+1,n+1))$$

- Trade-off between noise removal and detail preserving:
  - Larger window -> can remove noise more effectively, but also blur the details/edges

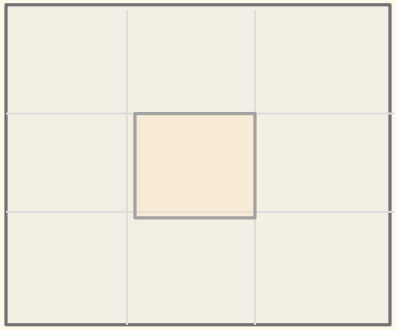
Example: 3x3 average

100	100	100	100	100
100	200	205	203	100
100	195	200	200	100
100	200	205	195	100
100	100	100	100	100

→

100	100	100	100	100
100	144	167	145	100
100	167	200	168	100
100	144	166	144	100
100	100	100	100	100

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$



Filter = kernel = weighting mask
Sum of all weights must = 1

Example Weighting Mask

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

But neighboring pixels likely have more relevance than far away pixels

$$\frac{1}{16} \times$$

1	2	1
2	4	2
1	2	1

Use unequal weighting to favor nearby pixels

Example: Weighted Average

100	100	100	100	100
100	200	205	203	100
100	195	200	200	100
100	200	205	195	100
100	100	100	100	100

$\frac{1}{9}\times$

1	1	1
1	1	1
1	1	1

Noise reduced to 0.11

100	100	100	100	100
100	144	167	145	100
100	167	200	168	100
100	144	166	144	100
100	100	100	100	100

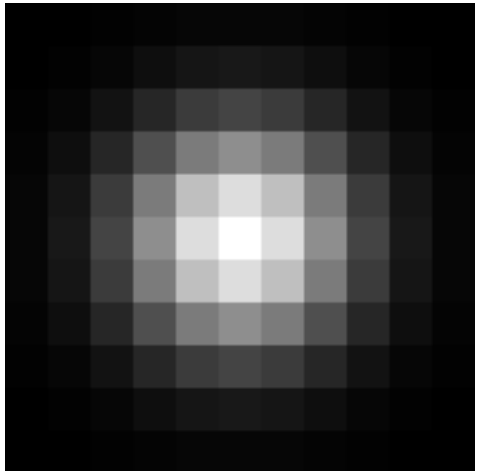
$\frac{1}{16}\times$

1	2	1
2	4	2
1	2	1

Noise reduced to 0.14

100	100	100	100	100
100	156	176	158	100
100	174	201	175	100
100	156	175	156	100
100	100	100	100	100

Smoothing with Gaussians



Constructing Gaussian Kernels

- To construct a 1D Gaussian kernel with an arbitrary standard deviation σ, simply sample the continuous zero-mean Gaussian function:

$$gauss_{\sigma^2}(x) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-x^2}{2\sigma^2}\right)$$

- Normalize by dividing each element by the sum of all the elements

$$\sum_i gauss_{\sigma^2}[i]$$

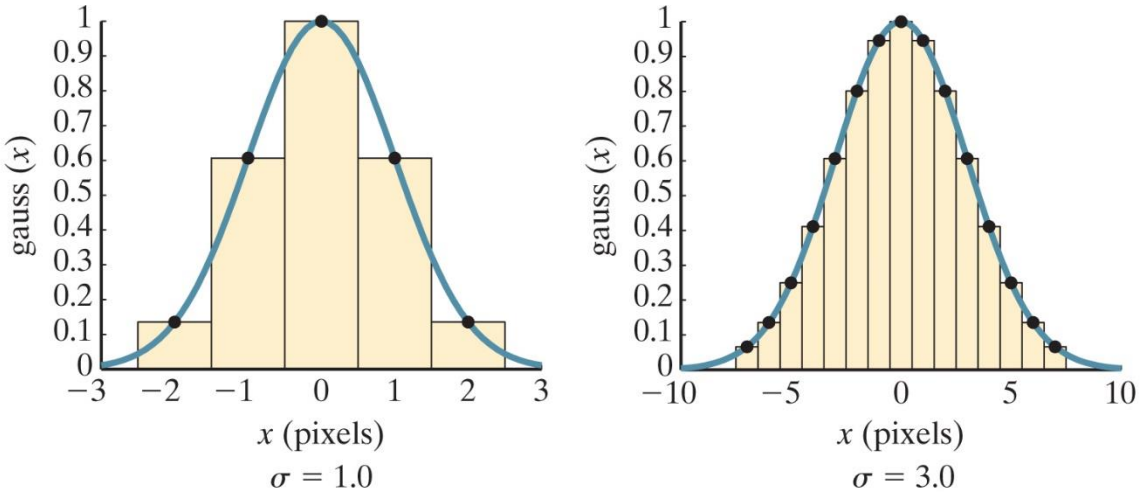
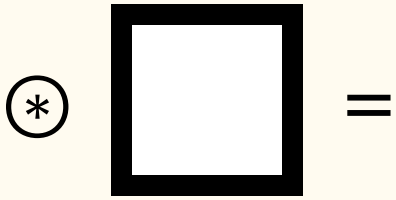


Figure 5.4 A continuous Gaussian (blue curve) and its discrete approximation (black circles) for two different values of σ. The shaded area under the rectangles approximates the area under the continuous curve between $x = -2.5\sigma$ and $x = +2.5\sigma$. The width and half-width of the kernels are $w = 5$, $\tilde{w} = 2$ (for $\sigma = 1.0$), and $w = 15$, $\tilde{w} = 7$ (for $\sigma = 3.0$), according to $\tilde{w} + 0.5 = 2.5\sigma$ and $w = 2\tilde{w} + 1$.

NIH ImageJ examples

Dr. O'Dell's filter plugin

518 x 800 pixels



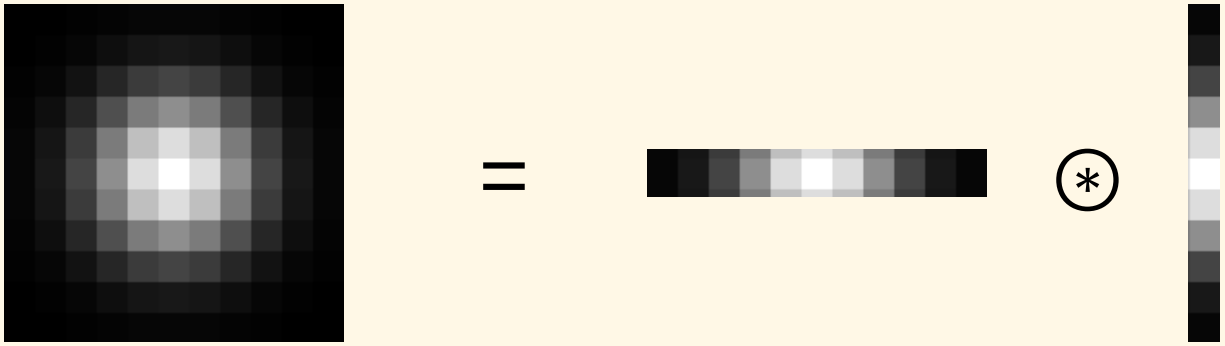
13 x 13 pixels
11x11 ON

Separability

- When a 2D kernel can be decomposed into the convolution of two 1D kernels, the kernel is **separable.**
- When the kernel is separable there is a compact, elegant notation:
- Where $\mathbf{G}_h$ is the convolution matrix constructed from the 1D kernel $\mathbf{g}_h$.

$$\mathbf{I}' = \mathbf{G}_h \mathbf{I} \mathbf{G}_v^\top$$

Separability of 2D Gaussian



$$G_{2D} = G_{hort} \circledast G_{vert}$$

Computational speed advantage:

$$F \circledast G_{2D} = F \circledast (G_h \circledast G_v) = (F \circledast G_h) \circledast G_v$$

2D convolution is $O(w_F^2 w_G^2)$

where w_G = width of Gaussian kernel

two 1D convolutions are $O(w_F^2 2w_G)$

Smoothing with Large Gaussians

- **Central limit theorem**: states that the repeated convolution of any nonnegative kernel with itself always converges to the shape of a Gaussian.

Gaussian Kernels

- Convolving a box filter with itself yields an approximation to a **Gaussian kernel.**

$$gauss_{\sigma^2}(x) \equiv \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

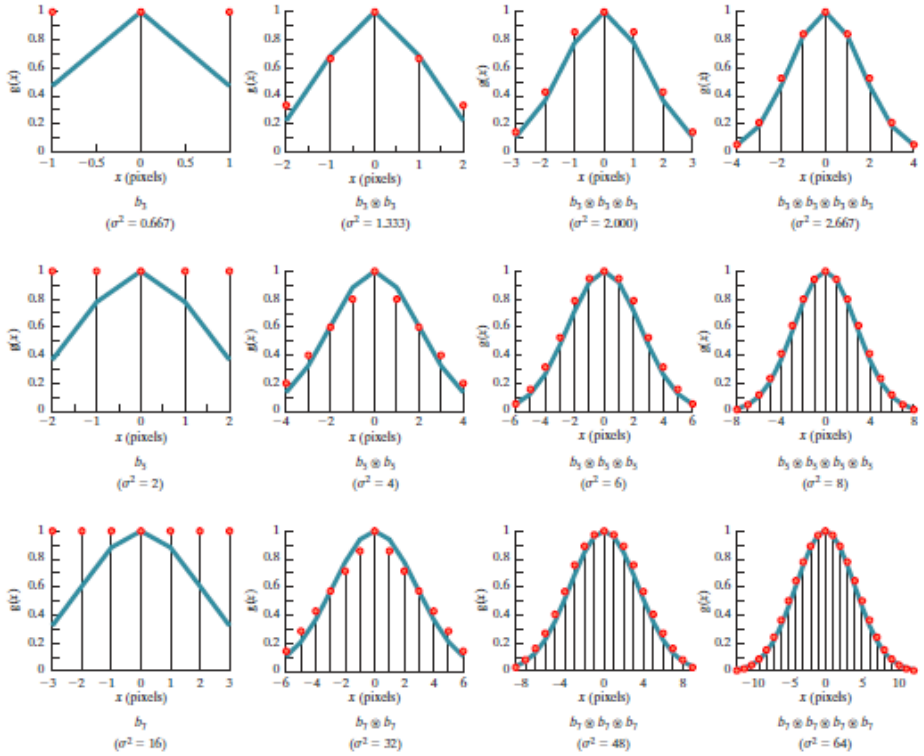
where σ^2 is the variance.

$$\mu = \frac{\sum_i i g(i)}{\sum_i g(i)}$$

$$\sigma^2 = \frac{\sum_i (i-\mu)^2 g(i)}{\sum_i g(i)}$$

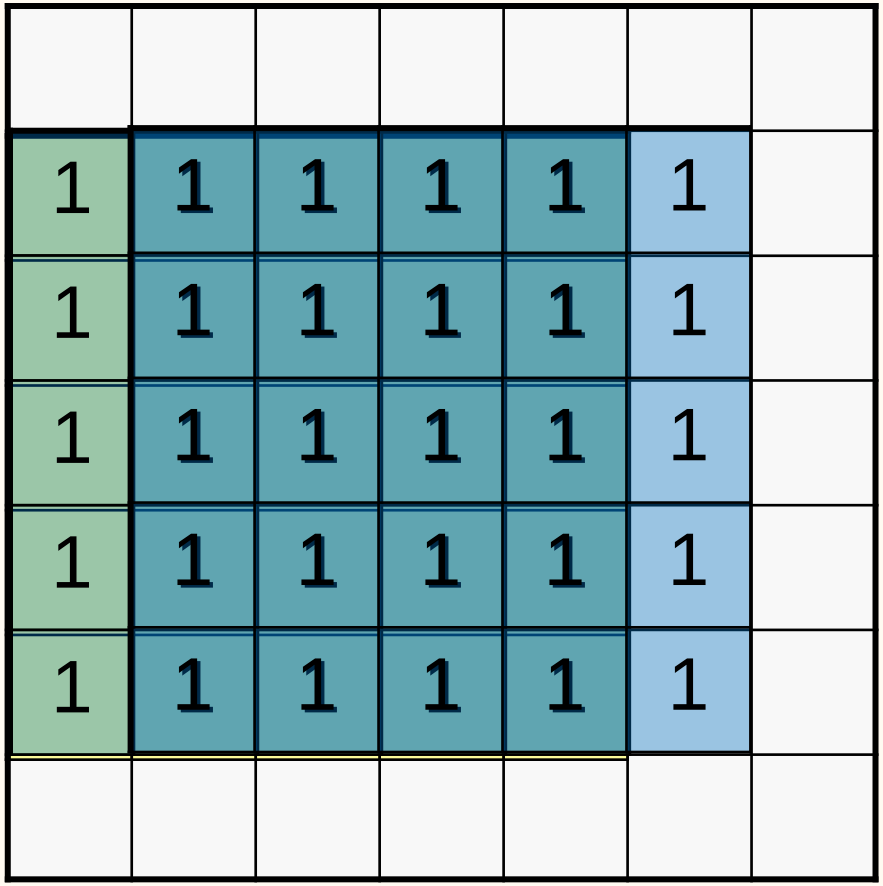
Smoothing with Large Gaussians (cont'd)

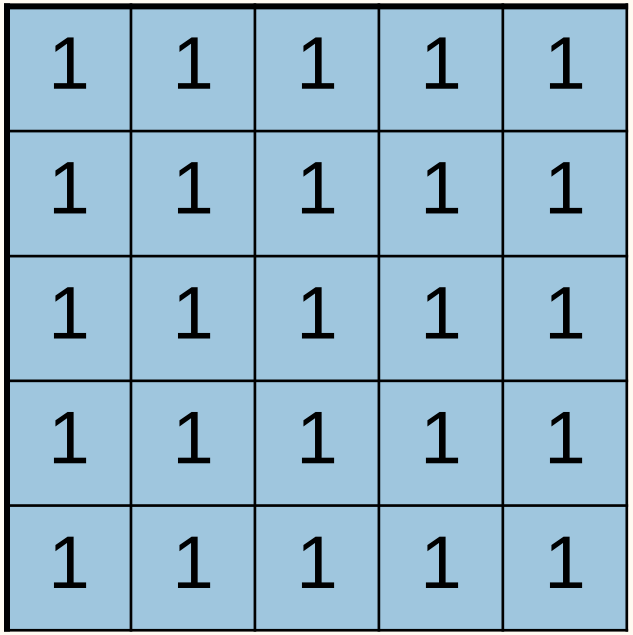
Figure 5.6 Repeated convolutions of a box filter leads to a Gaussian. The vertical bars (in blue) show the samples of discrete kernels obtained by convolving a box filter with itself, for $w = 3$ (top), $w = 5$ (middle), and $w = 7$ (bottom); while the overlaid plots (in red) show the sampled Gaussians with the same variance as the kernels. After just a few convolutions, the approximation is extremely accurate, thus enabling efficient approximation of arbitrarily-sized Gaussians.



Box filter speed-up:

needs only incremental addition rather than pixel-wise multiplication





At next pixel only need to + new column and
– previous column

Integral Image

$$S(x,y) = I(x,y) - S(x-1,y-1) + S(x-1,y) + S(x,y-1)$$

1	2	3	4	5	1	1
2	4	6	8	1	1	1
3	6	9	1	1	1	1
4	8	1	1	1	1	1
5	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

Integral Image

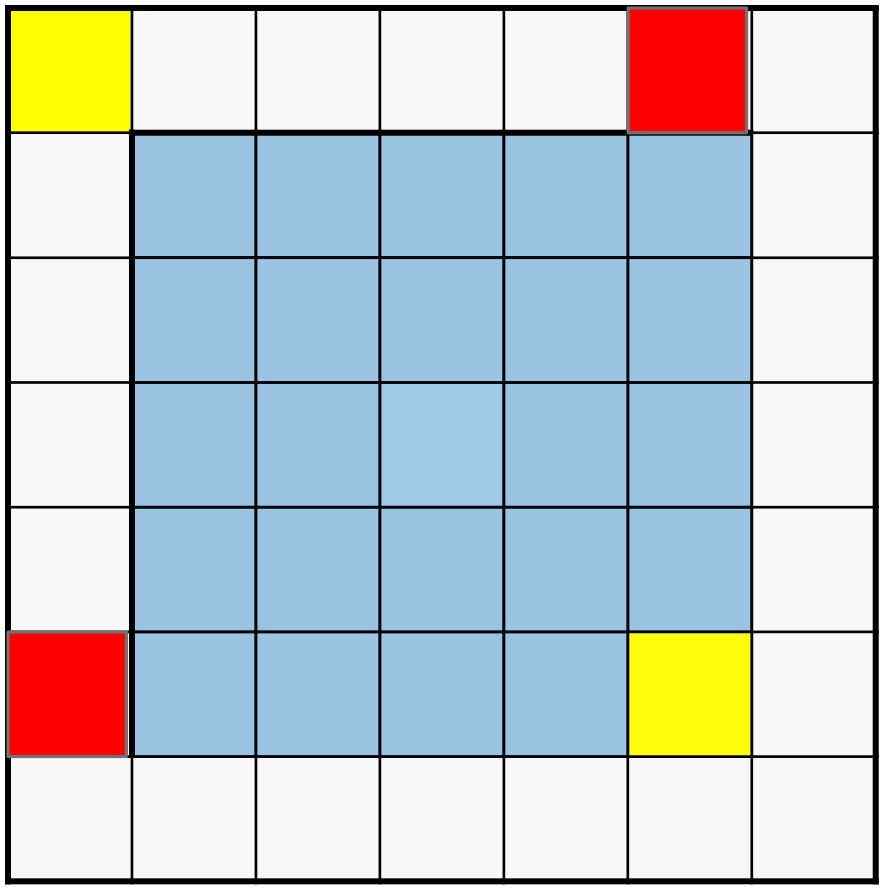
- The running sum is extended to 2D, where it is known as the **integral image.**

$$\sum_{x=x_0}^{x_1} \sum_{y=y_0}^{y_1} I(x,y) = S(x_0-1, y_0-1) + S(x_1, y_1) - S(x_0-1, y_1) - S(x_1, y_0-1)$$

- where (x_0, y_0) and (x_1, y_1) are the top-left and bottom-right coordinates of the rectangle, respectively.

Integral Image for 5x5 Box Filter Sum

$$\sum_{x=x_0}^{x_1} \sum_{y=y_0}^{y_1} I(x,y) = S(x_0-1, y_0-1) + S(x_1, y_1) - S(x_0-1, y_1) - S(x_1, y_0-1)$$



Convolving a 5x5 box filter 3 times is faster than convolving a $\sigma^2=6$ Gaussian once.

Computing First Derivative

Gaussian Derivative Kernels

- 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].

The values are backwards because convolving involves flipping the kernel

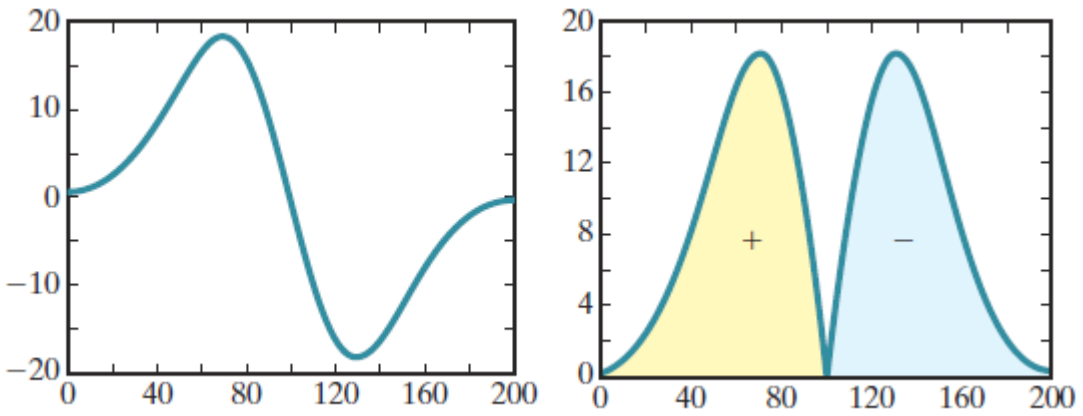
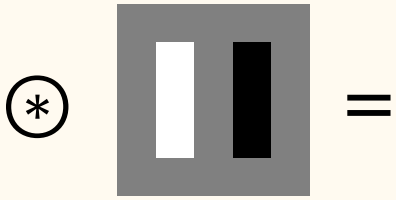


Figure 5.7 The 1D Gaussian derivative (left), along with an equivalent view of the operation (right) in which a weighted sum of the values on one side are subtracted from a weighted sum of the values on the other side.

$$(I \circledast K_G) \circledast K_D = I \circledast (K_G \circledast K_D)$$

NIH ImageJ examples

Dr. O'Dell's filter plugin



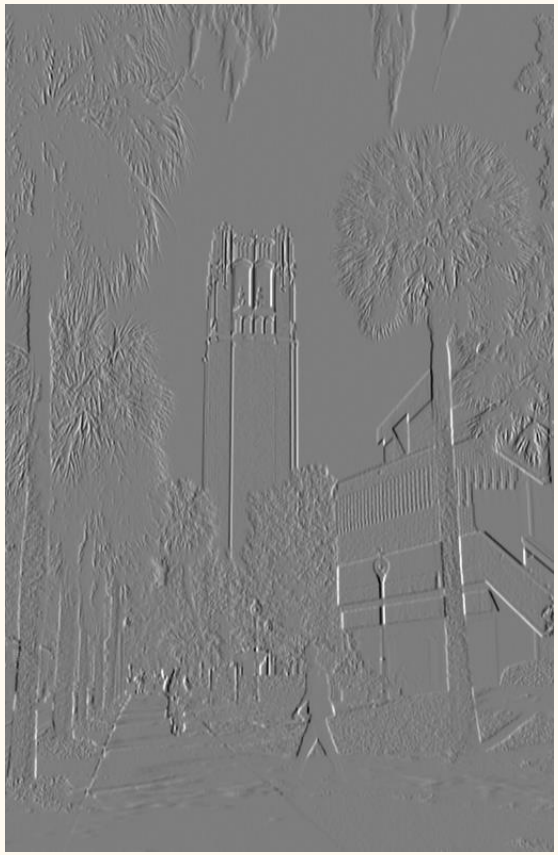


Image Gradient

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

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

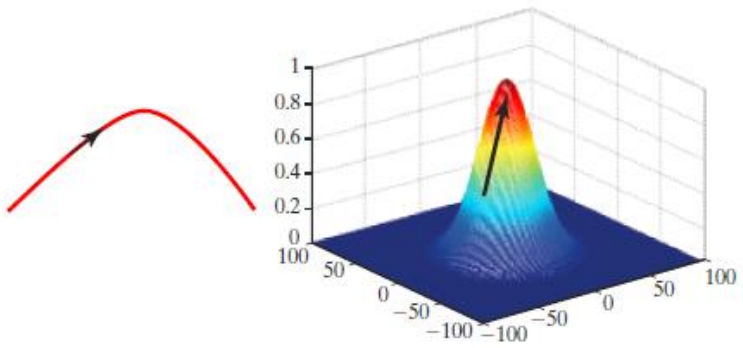


Figure 5.9 The derivative of a 1D function (left), and the gradient of a 2D function (right).

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:

for averaging

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

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

Sobel Operator

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

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

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

Scharr Operator

- The **Scharr operator** is similar to Sobel, but with a smaller variance ($\sigma^2 = 0.375$) in the smoothing kernel:

$$Scharr_x = gauss_{0.375}(y) \circledast \dot{g}auss_{0.5}(x) = \frac{1}{16}\begin{bmatrix} 3 \\ 10 \\ 3 \end{bmatrix} \circledast \frac{1}{2}[1 \quad 0 \quad -1] = \frac{1}{32}\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$$

$$Scharr_y = gauss_{0.375}(x) \circledast \dot{g}auss_{0.5}(y) = \frac{1}{16}[3 \quad 10 \quad 3] \circledast \frac{1}{2}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} = \frac{1}{32}\begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$

Computing Second Derivative

Computing the Second Derivative

- This can be seen in 1D by convolving the function with the noncentralized difference operator, then convolving the result again with the same operator:

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

Computing the Second Derivative (cont'd)

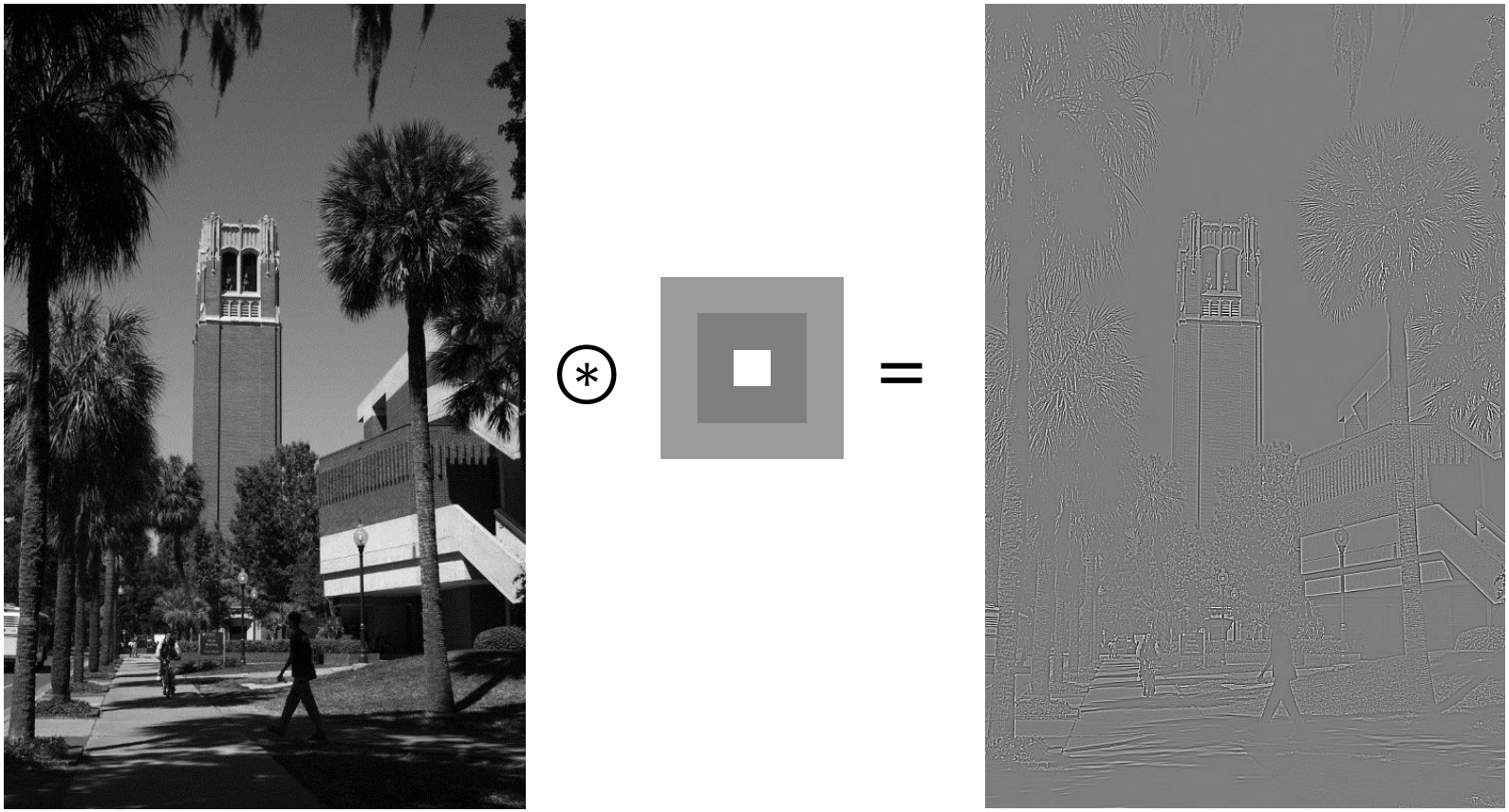
- For a function (or image) of two variables, the second-derivative in the *x* and *y* directions can be obtained by convolving with the appropriately oriented second-derivative kernel:

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

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

NIH ImageJ examples

Eye Kernel = 2D second-derivative

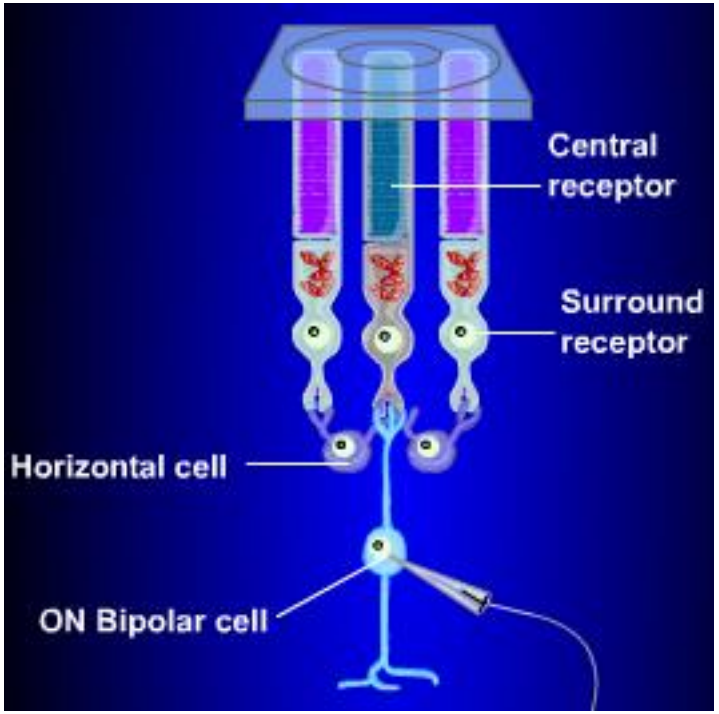


Human Retina Processing

The layer of cells below the photoreceptors create a center-surround activation pattern that acts as a 2D edge detector:

http://neuroscience.uth.tmc.edu/s2/chapter14.html

-1	-1	-1
-1	8	-1
-1	-1	-1



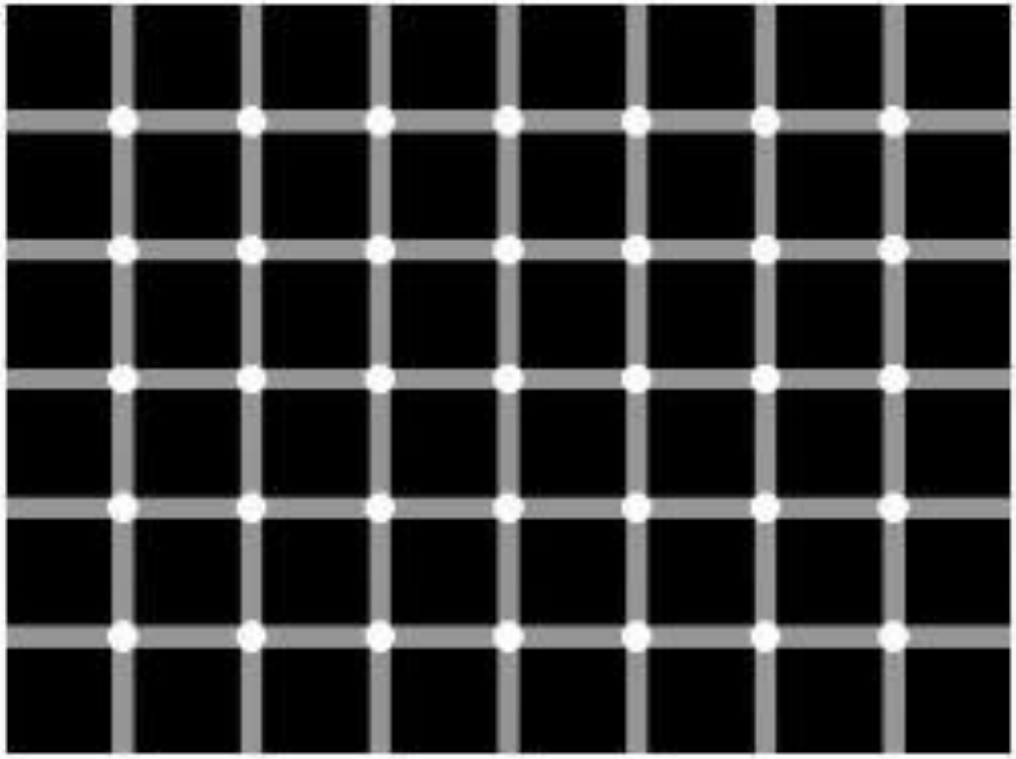
Uniform light → no signal to brain

Edge of light → partial signal

Pinpoint of light → large signal

Human Retina Processing

-1	-1	-1
-1	8	-1
-1	-1	-1



Laplacian of Gaussian (LoG)

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

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

- Because of associativity, computing the Laplacian of a smoothed image is the same as convolving the image with the **Laplacian of Gaussian (LoG):**

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

Difference of Gaussians (DoG)

- The Laplacian of Gaussian (LoG) is closely connected to the **difference of Gaussians (DoG)**.

$$
\begin{aligned}
\frac{d\, gauss_{\sigma^2}(x)}{d\sigma} &= \frac{d}{d\sigma}\left(\frac{1}{\sqrt{2\pi}}\sigma^{-1}e^{-\frac{1}{2}x^2\sigma^{-2}}\right) \\
&= \frac{1}{\sqrt{2\pi}}\left(-\sigma^{-2} + \sigma^{-1}(x^2\sigma^{-3})\right) e^{-\frac{x^2}{2\sigma^2}} \\
&= \frac{1}{\sqrt{2\pi}}\left(-\frac{1}{\sigma^2} + \frac{x^2}{\sigma^4}\right) e^{-\frac{1}{2}x^2\sigma^{-2}} \\
&= -\sigma\left(\frac{1}{\sigma^2} - \frac{x^2}{\sigma^4}\right) gauss_{\sigma^2}(x)
\end{aligned}
$$

Difference of Gaussians (DoG) (cont'd)

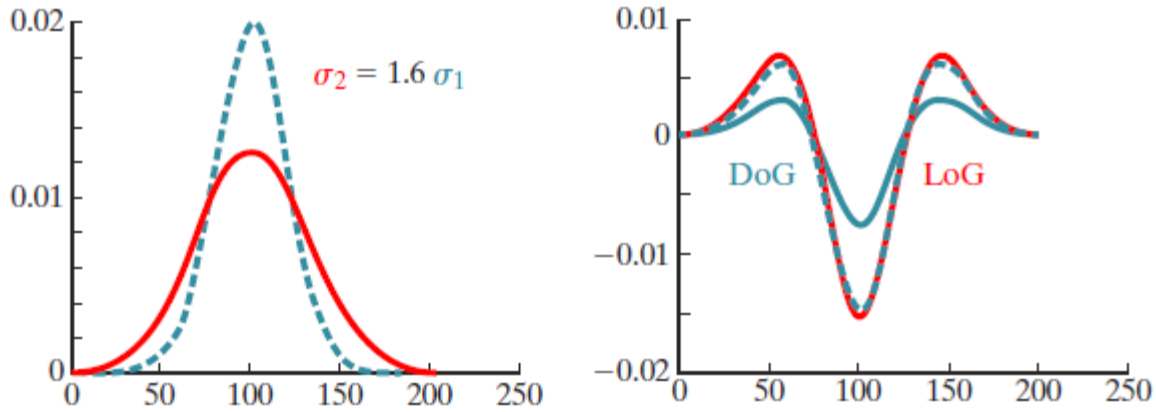


Figure 5.15 LEFT: Two Gaussians whose ratio of standard deviations is 1.6. RIGHT: The difference of Gaussians (solid blue) and 1D Laplacian of Gaussian (solid red). The scaled DoG (dashed blue) approximates the LoG.