ECE5554 – Computer Vision
Lecture 3a – Other Edge Detection Methods

Creed Jones, PhD

Course Updates

- HW assignment 1 is due TONIGHT at 11:59 PM
- HW2 is posted – due next Wednesday, June 20
- Videos of lectures are posted up to one day after the class session

Today's Objectives

Edge detection

- Compass edge operators
- Difference of Gaussians
- Laplacian operators
- Laplacian of Gaussian

"Compass" edge detection

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

$$0^\circ \qquad\qquad 45^\circ \qquad\qquad 90^\circ \qquad\qquad 135^\circ$$

- At each pixel location, compute responses to several templates
- The largest magnitude determines the "winner" -- nonlinear!
- These are sometimes called "generalized Sobel masks"

Another "compass" example: Nevatia-Babu masks

$$\begin{bmatrix} -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \end{bmatrix}$$

0°

$$\begin{bmatrix} -100 & 32 & 100 & 100 & 100 \\ -100 & -78 & 92 & 100 & 100 \\ -100 & -100 & 0 & 100 & 100 \\ -100 & -100 & -92 & 78 & 100 \\ -100 & -100 & -100 & -32 & 100 \end{bmatrix}$$

30°

$$\begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ -32 & 78 & 100 & 100 & 100 \\ -100 & -92 & 0 & 92 & 100 \\ -100 & -100 & -100 & -78 & 32 \\ -100 & -100 & -100 & -100 & 100 \end{bmatrix}$$

60°

$$\begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \\ 0 & 0 & 0 & 0 & 0 \\ -100 & -100 & -100 & -100 & -100 \\ -100 & -100 & -100 & -100 & -100 \end{bmatrix}$$

90°

$$\begin{bmatrix} 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 78 & -32 \\ 100 & 92 & 0 & -92 & -100 \\ 32 & -78 & -100 & -100 & -100 \\ -100 & -100 & -100 & -100 & -100 \end{bmatrix}$$

120°

$$\begin{bmatrix} 100 & 100 & 100 & 32 & -100 \\ 100 & 100 & 92 & -78 & -100 \\ 100 & 100 & 0 & -100 & -100 \\ 100 & 78 & -92 & -100 & -100 \\ 100 & -32 & -100 & -100 & -100 \end{bmatrix}$$

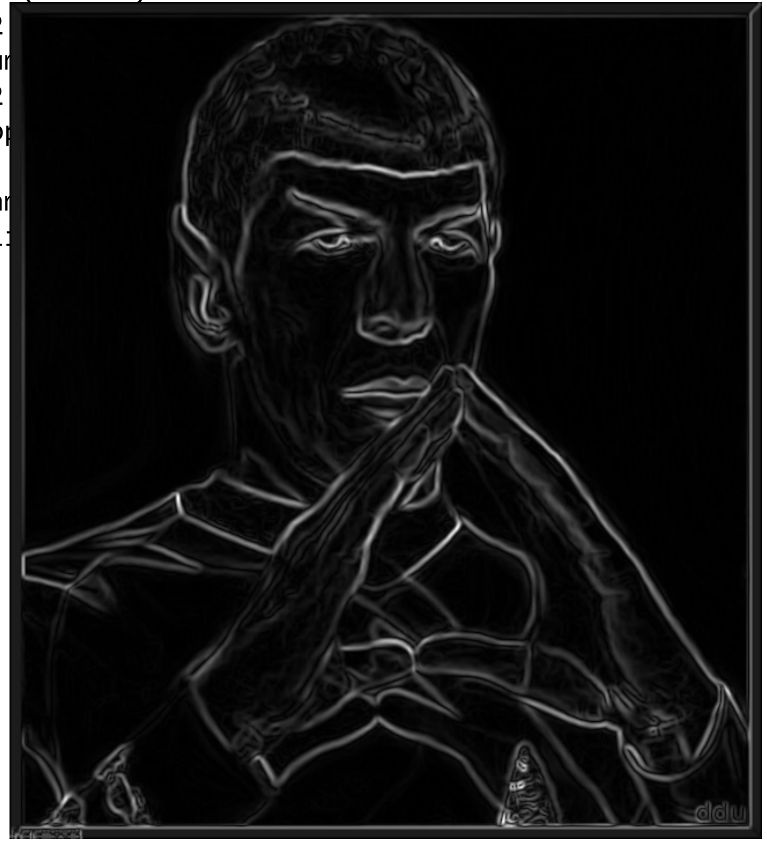
150°

```
def NevatiaBabu():
    nb0 = np.asarray([ [-100, -100, 0, 100, 100 ],
            [-100, -100, 0, 100, 100],
            [-100, -100, 0, 100, 100],
            [-100, -100, 0, 100, 100],
            [-100, -100, 0, 100, 100] ])

    nb30 = np.asarray([ [-100, 32, 100, 100, 100],
             [-100, -78, 92, 100, 100],
             [-100, -100, 0, 100, 100],
             [-100, -100, -92, 78, 100],
             [-100, -100, -100, -32, 100] ])

    nb60 = np.asarray([ [100, 100, 100, 100, 100],
             [-32, 78, 100, 100, 100],
             [-100, -92, 0, 92, 100],
             [-100, -100, -100, -78, 32],
             [-100, -100, -100, -100, -100] ])

    nb90 = np.asarray([ [100, 100, 100, 100, 100],
             [100, 100, 100, 100, 100],
             [0, 0, 0, 0, 0],
             [-100, -100, -100, -100, -100],
             [-100, -100, -100, -100, -100] ])

    nb120 = np.asarray([ [100, 100, 100, 100, 100],
              [100, 100, 100, 78, -32],
              [100, 92, 0, -92, -100],
              [32, -78, -100, -100, -100],
              [-100, -100, -100, -100, -100] ])

    nb150 = np.asarray([ [100, 100, 100, 32, -100],
              [100, 100, 92, -78, -100],
              [100, 100, 0, -100, -100],
              [100, 78, -92, -100, -100],
              [100, -32, -100, -100, -100] ])
```

```
# load an image and form the filter responses
   pathname = 'C:/Data/Images/'
   filename = "spock.png"
   img = cv2.imread(pathname + filename)
   grayimg =cv2.cvtColor(img,
                       cv2.COLOR_BGR2GRAY).astype(np.float32)
   result = np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb0))
   result = np.maximum(result,
           np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb30)))
   result = np.maximum(result,
           np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb60)))
   result = np.maximum(result,
           np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb90)))
   result = np.maximum(result,
           np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb120)))
   result = np.maximum(result,
           np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb150)))
   normed = result.copy()

   cv2.imwrite(pathname + 'NevatiaBabu.png’,
           cv2.normalize(result, normed, 0, 255, cv2.NORM_MINMAX))
```

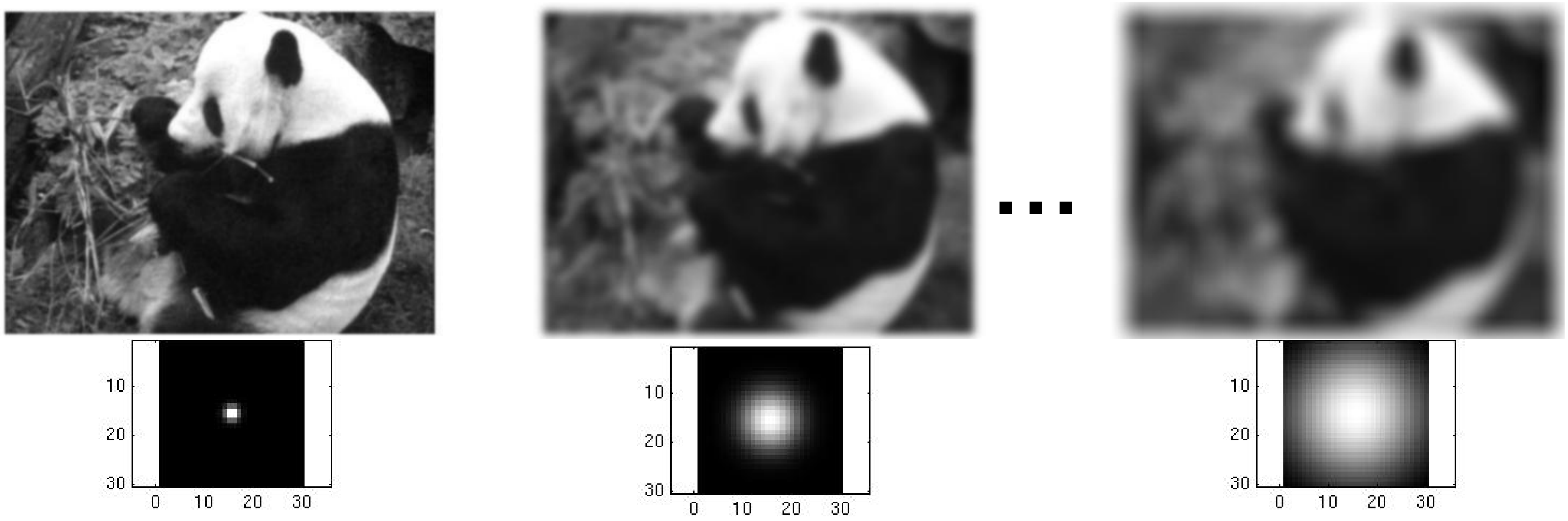
```python
def NevatiaBabu():
    nb0 = np.asarray([ [-100, -100, 0, 100, 100 ],
            [-100, -100, 0, 100, 100],
            [-100, -100, 0, 100, 100],
            [-100, -100, 0, 100, 100],
            [-100, -100, 0, 100, 100] ])

    nb30 = np.asarray([ [-100, 32, 100, 100, 100],
             [-100, -78, 92, 100, 100],
             [-100, -100, 0, 100, 100],
             [-100, -100, -92, 78, 100],
             [-100, -100, -100, -32, 100] ])

    nb60 = np.asarray([ [100, 100, 100, 100, 100],
             [-32, 78, 100, 100, 100],
             [-100, -92, 0, 92, 100],
             [-100, -100, -100, -78, 32],
             [-100, -100, -100, -100, -100] ])

    nb90 = np.asarray([ [100, 100, 100, 100, 100],
             [100, 100, 100, 100, 100],
             [0, 0, 0, 0, 0],
             [-100, -100, -100, -100, -100],
             [-100, -100, -100, -100, -100] ])

    nb120 = np.asarray([ [100, 100, 100, 100, 100],
              [100, 100, 100, 78, -32],
              [100, 92, 0, -92, -100],
              [32, -78, -100, -100, -100],
              [-100, -100, -100, -100, -100] ])

    nb150 = np.asarray([ [100, 100, 100, 32, -100],
              [100, 100, 92, -78, -100],
              [100, 100, 0, -100, -100],
              [100, 78, -92, -100, -100],
              [100, -32, -100, -100, -100] ])
```

```python
# load an image and form the filter responses
   pathname = 'C:/Data/Images/'
   filename = "spock.png"
   img = cv2.imread(pathname + filename)
   grayimg =cv2.cvtColor(img,
                     cv2.COLOR_BGR2GRAY).astype(np.float32)
   result = np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb0))
   result = np.maximum(result,
           np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb30)))
   result = np.maximum(result,
           np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb60)))
   result = np.maximum(result,
           np.abs(cv2.filter2D(grayimg, cv2.CV_32F, nb90)))
   result = np.maximum(result,
           np.abs(cv2
   result = np.maximum
           np.abs(cv2
   normed = result.cop

   cv2.imwrite(pathnam
           cv2.normal
```



Remember the use of Gaussian kernels to smooth an image

Recall: parameter σ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.

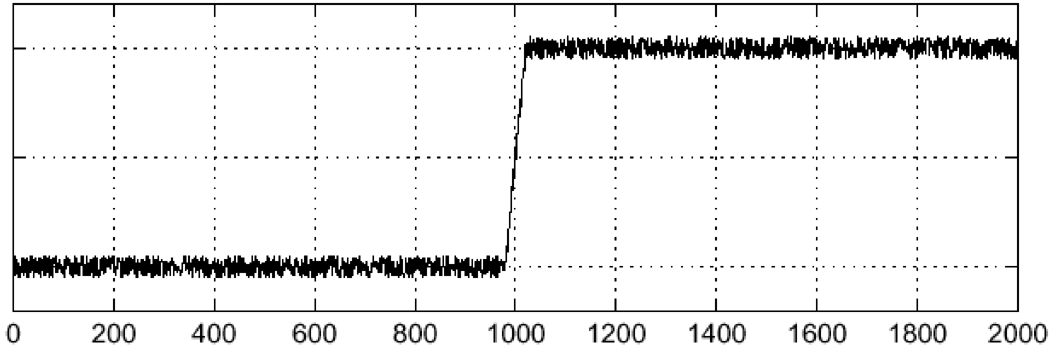


Slide credit: Grauman

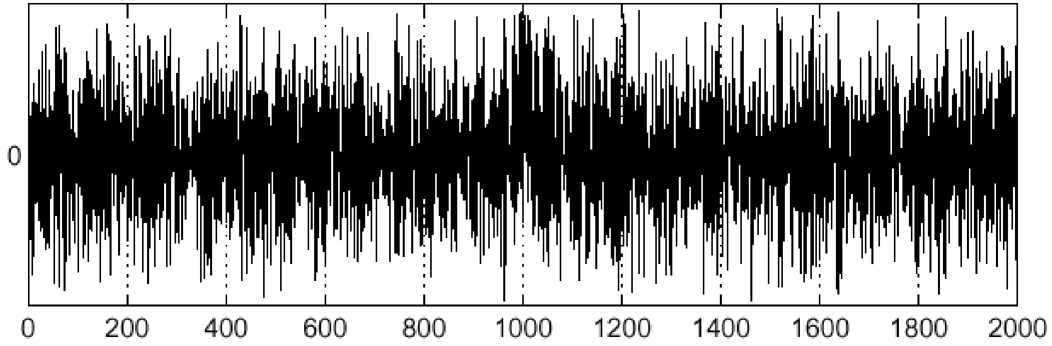
Noise can make it difficult to find the location of an edge (light-dark transitions are obscured)

Consider a single row or column of an image

$f(x)$



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



Where is the edge?

Slide credit: Seitz

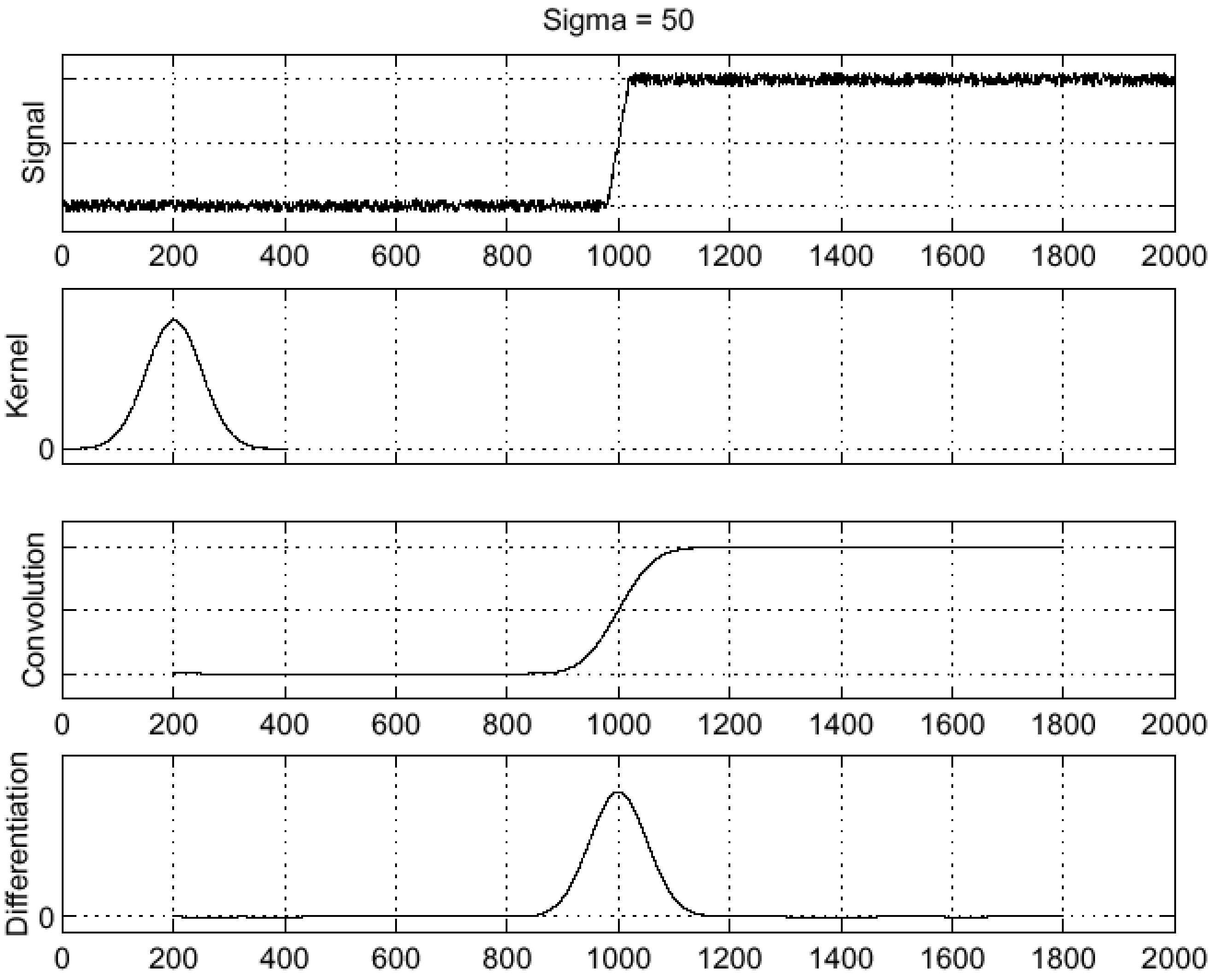
Solution: smooth first

f

G

$G \star f$

$\frac{\partial}{\partial x}(G \star f)$



Where is the edge?

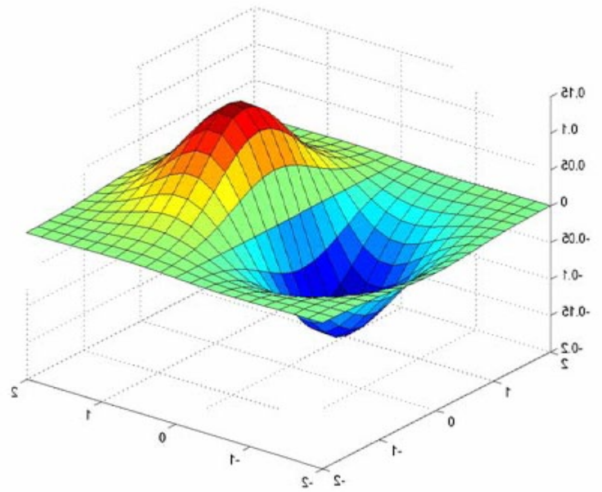
Look for peaks in $\frac{\partial}{\partial x}(G \star f)$

Slide credit: Grauman (adapted)

Derivative-of-Gaussian filters

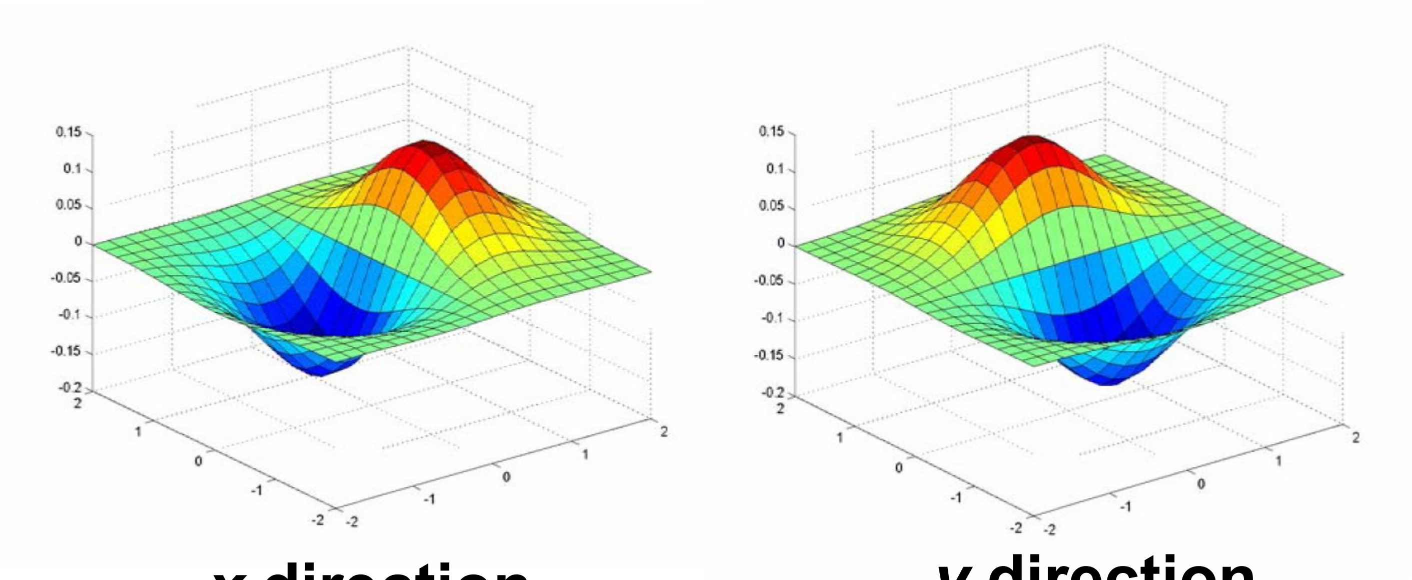
$$(I * g) * h = I * (g * h)$$

$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} * \begin{bmatrix} 1 & -1 \end{bmatrix}$$

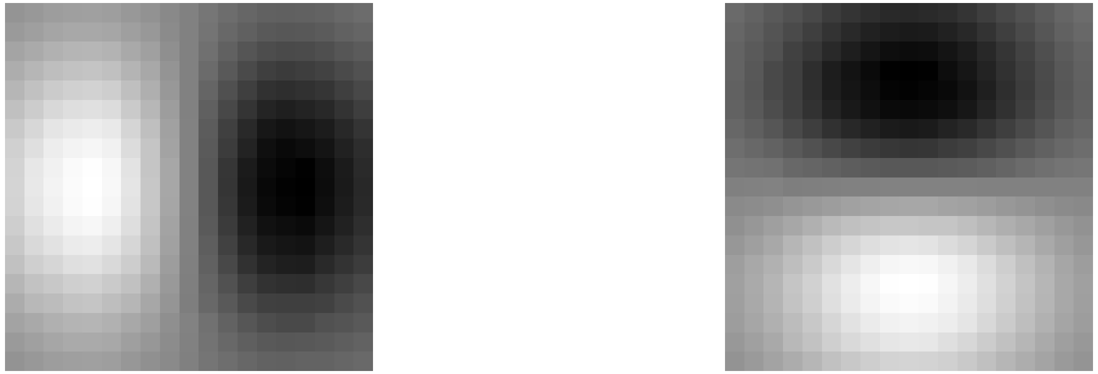


Slide credit: Grauman

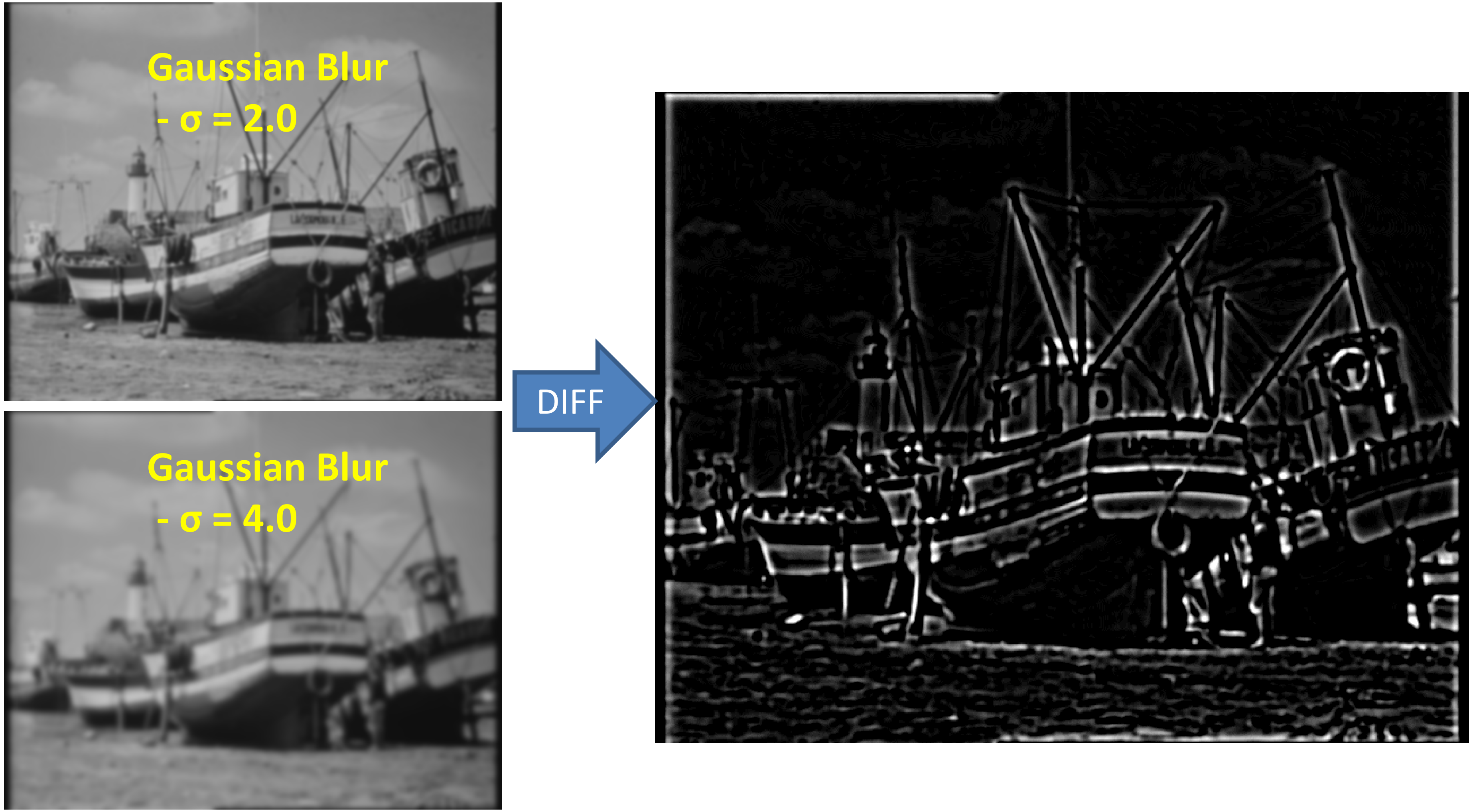
Derivative-of-Gaussian filters



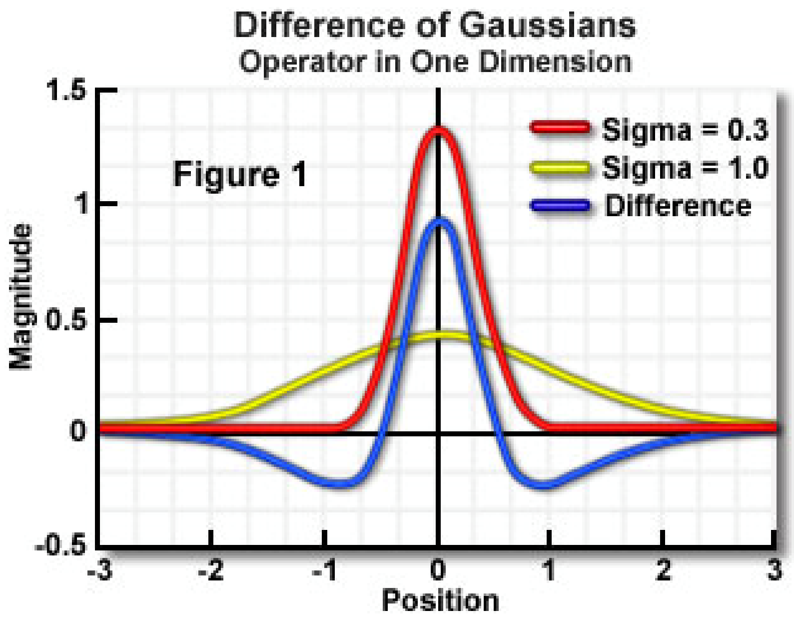
***x* direction** **y direction**

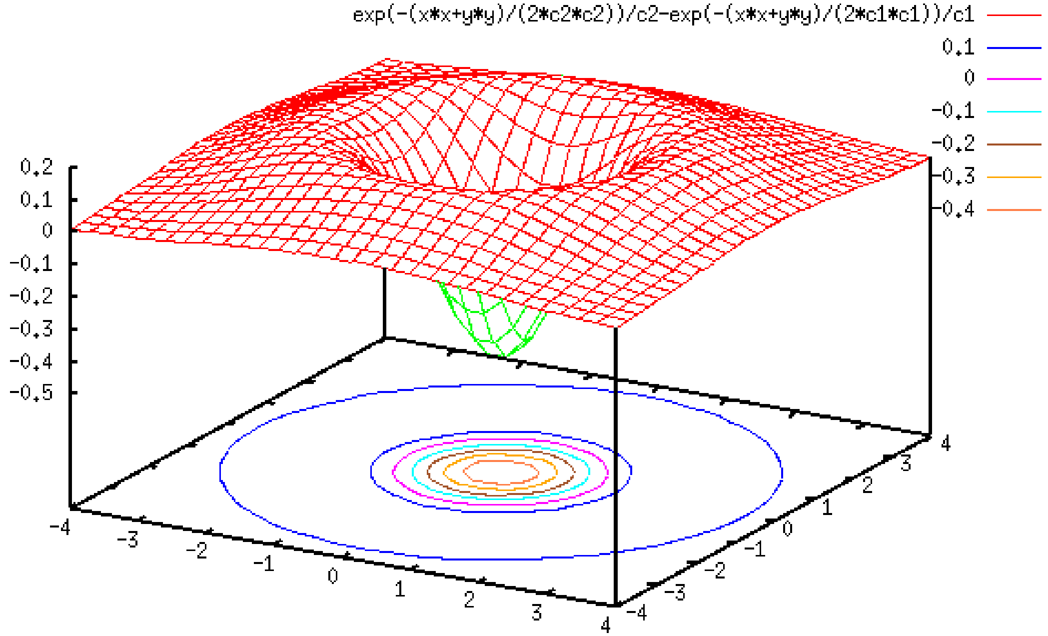


Another common edge detection technique is difference of Gaussians (DoG) – the image is filtered using two Gaussians at two different σ, and the results are subtracted



The difference of Gaussians (DoG) operator can be implemented as a single kernel to make the operation more efficient





We have looked at 1st-derivative operators What about using 2nd derivatives?

- The simplest nondirectional 2nd-derivative operator is the **Laplacian:**

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

- Here are two common approximations to the Laplacian:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

Changes of sign within a kernel indicate derivatives – one change of sign (from side to side) indicates approximation to first spatial derivative; two changes of sign indicates second derivative

- These kernels approximate the *Laplacian* or second spatial derivative

(a)

0	1	0
1	−4	1
0	1	0

(b)

1	1	1
1	−8	1
1	1	1

(c)

0	−1	0
−1	4	−1
0	−1	0

(d)

−1	−1	−1
−1	8	−1
−1	−1	−1

a b
c d

FIGURE 3.39 (a) Filter mask used to implement the digital Laplacian, as defined in Eq. (3.7-4). (b) Mask used to implement an extension of this equation that includes the diagonal neighbors. (c) and (d) Two other implementations of the Laplacian.

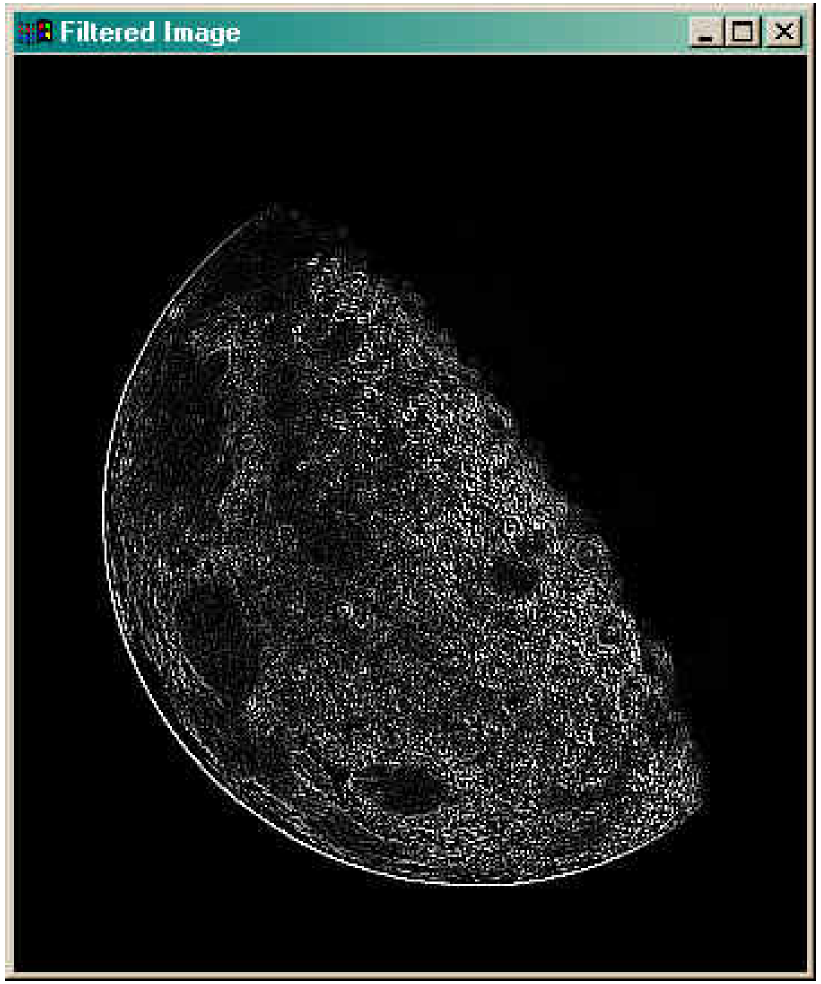
Laplacian operations on images approximate second derivatives

- The *Laplacian* operator is a method of approximating the second spatial derivative of the image brightness

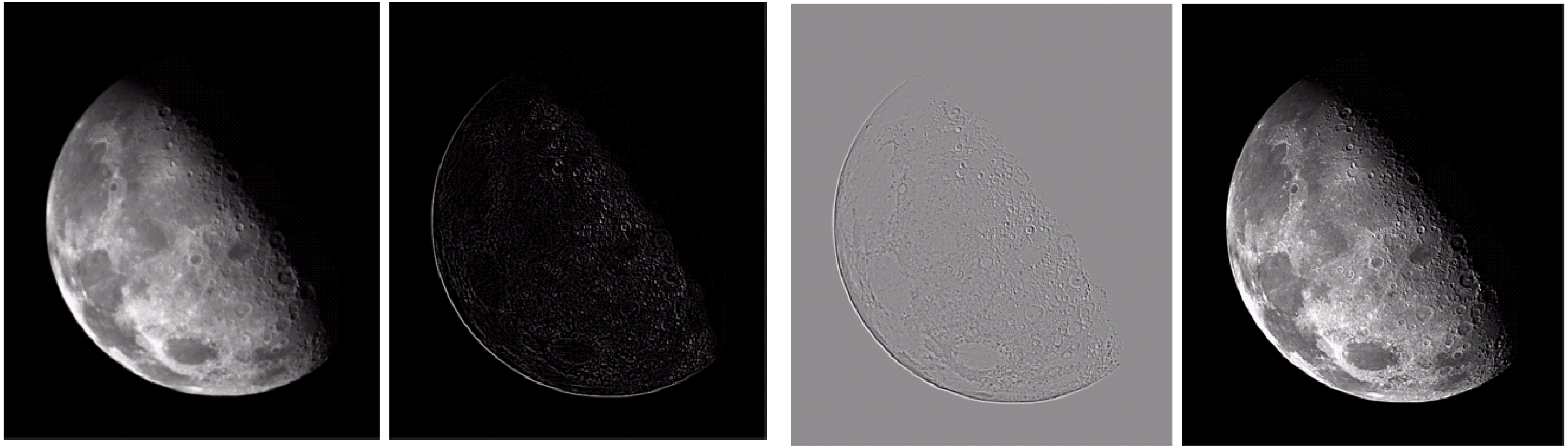
$$L(x,y) = \nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Since the second derivative is the rate of change of the slope, it will be zero when the image is constant or changing at a constant rate, but will be high (bright) when the image is changing

Result of Laplacian



The result of the Laplacian operator can be added back to the original image to enhance the edges in the image



a b
c d

FIGURE 3.40 (a) Image of the North Pole of the moon. (b) Laplacian-filtered image. (c) Laplacian image scaled for display purposes. (d) Image enhanced by using Eq. (3.7-5). (Original image courtesy of NASA.)

In 1979, Marr, Hildreth, and Poggio caused a lot of excitement when they suggested the combination of Laplacian and Gaussian filtering

- Differentiation and convolution are both associative and commutative!

$$I_{new}(x,y) = \nabla^2\big(I(x,y) * G(x,y)\big)$$

- Do this, then detect zero-crossings in the result

- Implement as:

$$I_{new}(x,y) = \nabla^2\big(I(x,y) * G(x,y)\big) = \big(\nabla^2 I(x,y)\big) * G(x,y) = I(x,y) * \big(\nabla^2 G(x,y)\big)$$

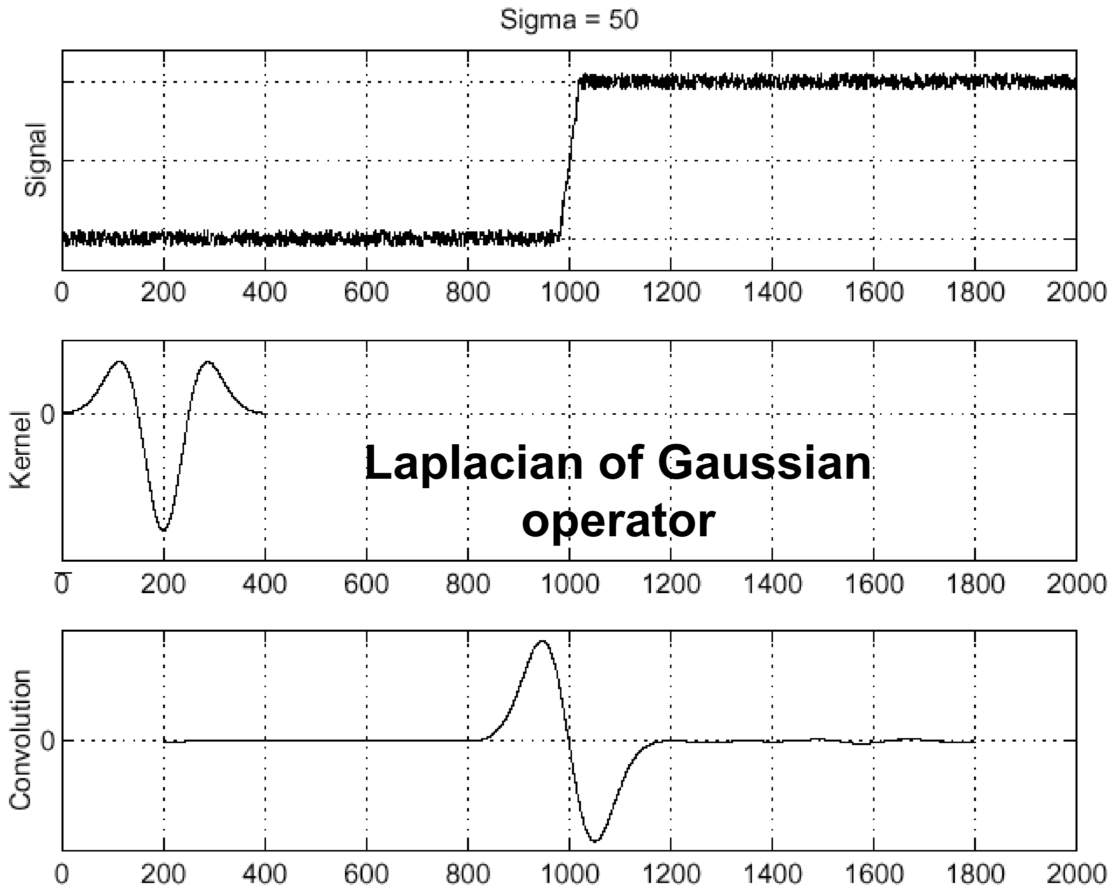
- The Laplacian-of-Gaussian (LoG) operator: $\nabla^2 G(x,y)$

Laplacian of Gaussian - $\frac{\partial^2}{\partial x^2}(G * f)$

Consider f

$\frac{\partial^2}{\partial x^2} G$

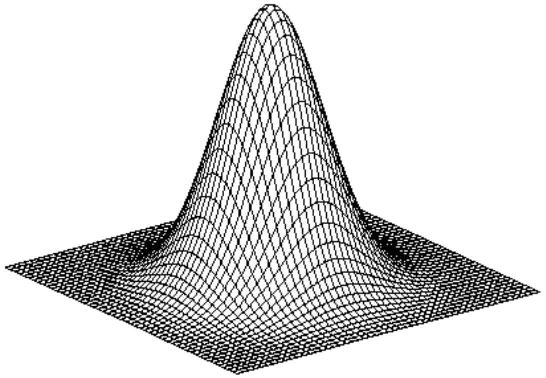
$\left(\frac{\partial^2}{\partial x^2} G\right) \star f$



Where is the edge? Zero-crossings of bottom graph

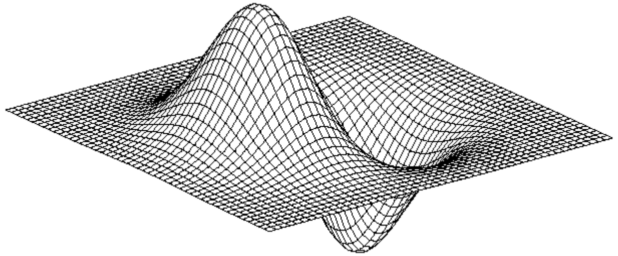
Slide credit: Seitz

2D edge detection filters



Gaussian

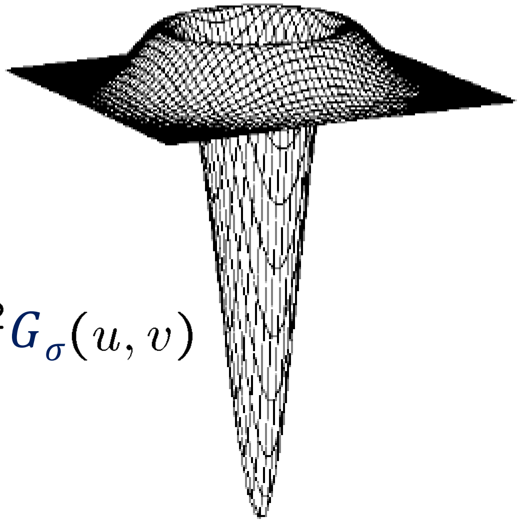
$$G_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} G_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 G_\sigma(u,v)$$

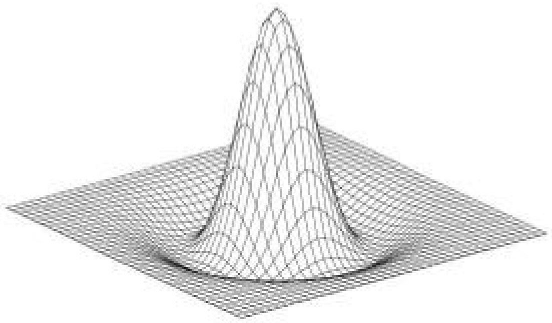
- ∇^2 is the Laplacian operator:

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

$\nabla^2 G$, shown upside down

This operator has several names:

- LoG operator
- Marr-Hildreth operator
- "Mexican hat" operator
- "Sombrero" function



By carefully selecting the width of the Gaussian, we can control the level of detail in the result

Each choice of sigma selects a different "scale"

Effect of σ on derivatives

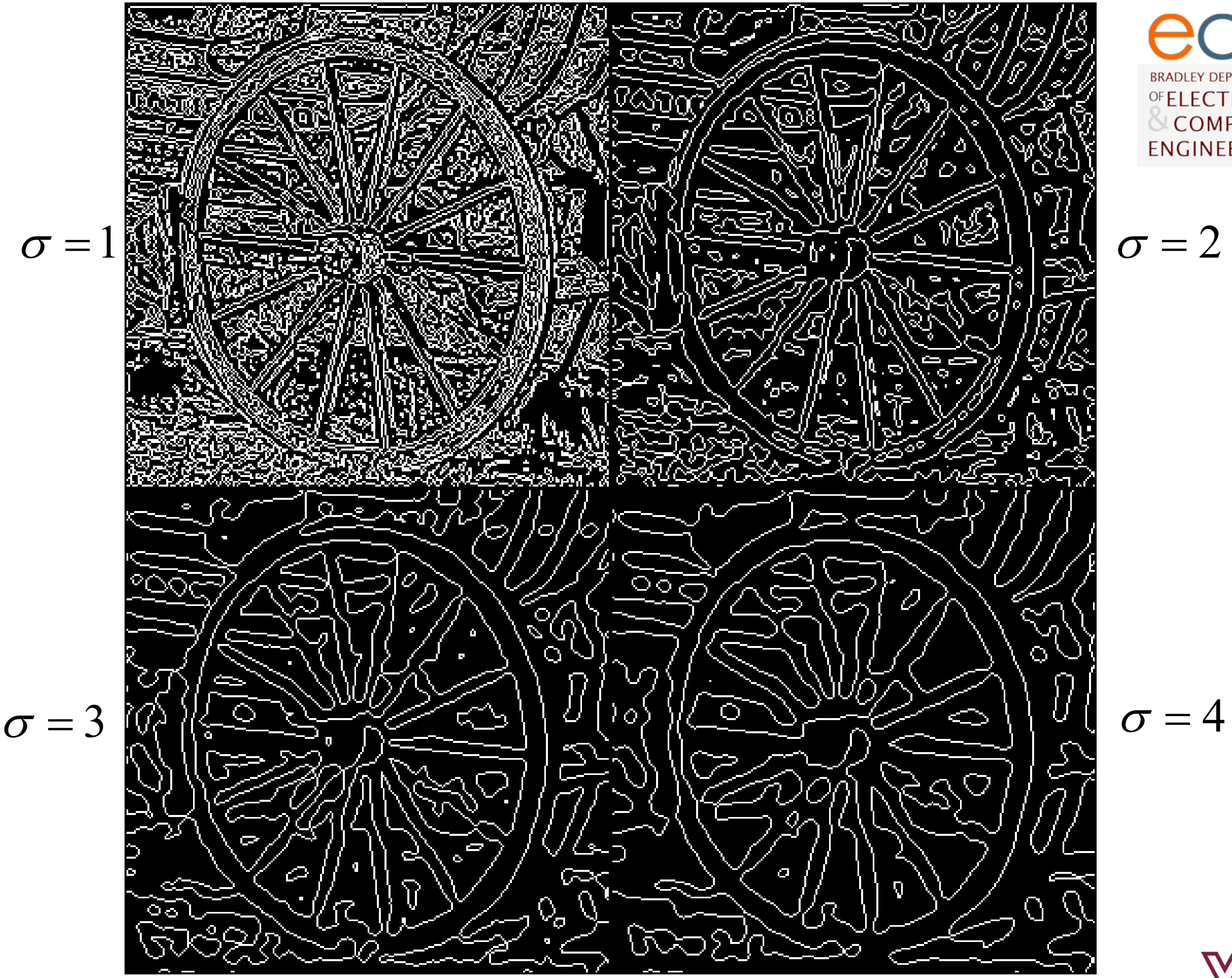
σ = 1 pixel **σ = 3 pixels**

The apparent structures differ, depending on σ

Larger values: coarser-scale features detected
Smaller values: finer-scale features detected

Slide credit: Grauman (adapted)

LoG example



VIRGINIA TECH.

ece BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Proc. R. Soc. Lond. B **207**, 187–217 (1980)
Printed in Great Britain

Theory of edge detection

By D. Marr and E. Hildreth

M.I.T. Psychology Department and Artificial Intelligence Laboratory,
79 *Amherst Street, Cambridge, Massachusetts* 02139, *U.S.A.*

(*Communicated by S. Brenner, F.R.S. – Received* 22 *February* 1979)

A theory of edge detection is presented. The analysis proceeds in two parts. (1) Intensity changes, which occur in a natural image over a wide range of scales, are detected separately at different scales. An appropriate filter for this purpose at a given scale is found to be the second derivative of a Gaussian, and it is shown that, provided some simple conditions are satisfied, these primary filters need not be orientation-dependent. Thus, intensity changes at a given scale are best detected by finding the zero values of $\nabla^2 G(x, y) * I(x, y)$ for image I, where $G(x, y)$ is a two-dimensional Gaussian distribution and ∇^2 is the Laplacian. The intensity changes thus discovered in each of the channels are then represented by oriented primitives called zero-crossing segments, and evidence is given that this representation is complete. (2) Intensity changes in images arise from surface discontinuities or from reflectance or illumination boundaries, and these all have the property that they are spatially localized. Because of this, the zero-crossing segments from the different channels are not independent, and rules are deduced for combining them into a description of the image. This description is called the raw primal sketch. The theory explains several basic psychophysical findings, and the operation of forming oriented zero-crossing segments from the output of centre-surround $\nabla^2 G$ filters acting on the image forms the basis for a physiological model of simple cells (see Marr & Ullman 1979).

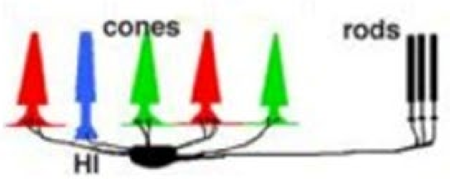
Introduction

The experiments of Hubel & Wiesel (1962) and of Campbell & Robson (1968) introduced two rather distinct notions of the function of early information processing in higher visual systems. Hubel & Wiesel's description of simple cells as linear with bar- or edge-shaped receptive fields led to a view of the cortex as containing a population of feature detectors (Barlow 1969, p. 881) tuned to edges and bars of various widths and orientations. Campbell & Robson's experiments, showing that visual information is processed in parallel by a number

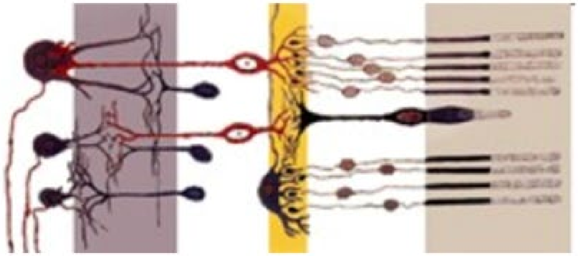
ece BRADLEY DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

VIRGINIA TECH.

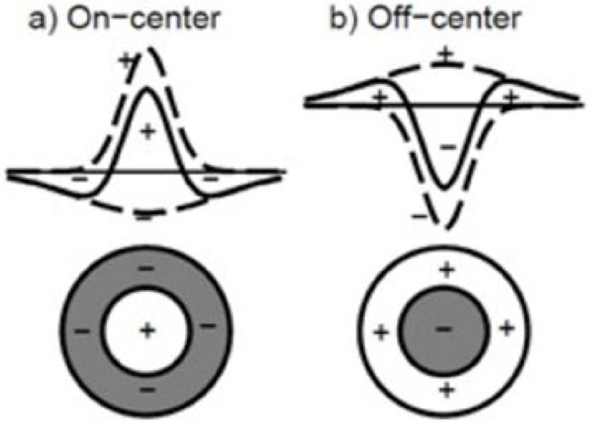
Retina



- The retina is not a passive camera registering images.
- Crucial rule: enhancing contrasts underlining changes in space and time, strengthening edges, uniformly lit areas are less important.
- Photoreceptors in rods and cones,
- 3-layer network, ganglion cells =>LGN.



- Receptive fields: areas, which stimulate a given cell.
- The combination of signals in the retina gives center-surround receptive fields (on-center) and vice versa, detects edges.
- Each individual field of cells can be modeled as a Gaussian model, so these fields are obtained as a difference of Gaussians (DOG).



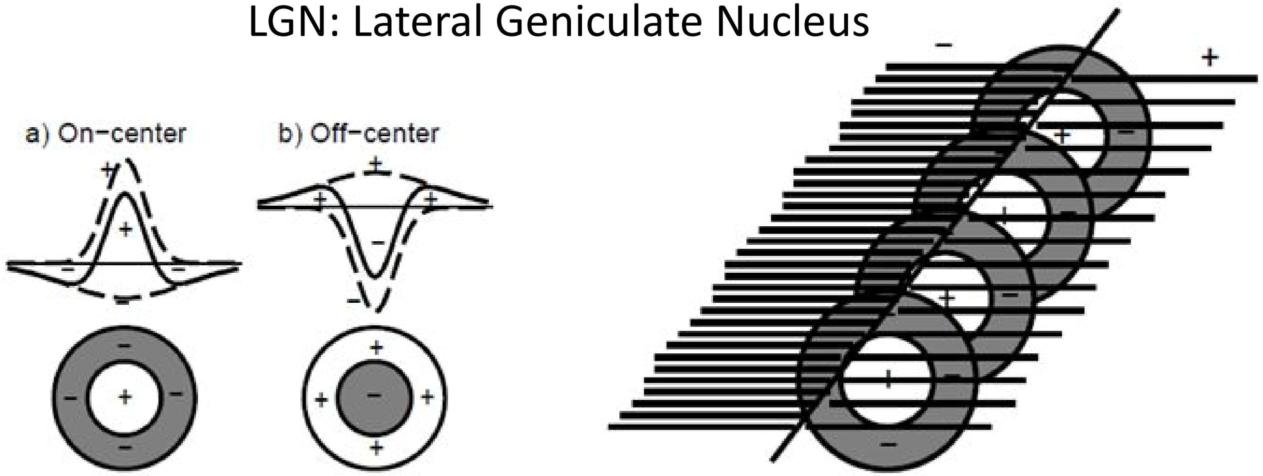
12

Janusz Starzyk, Univ of Colorado

Edge detectors

Contrasting signals points from the LGN are organized by the V1 cortex into edge detectors oriented at a specific angle.

LGN: Lateral Geniculate Nucleus



Simple V1 cells combine into edge detectors, enabling the determination of shapes, other cells react to color and texture.

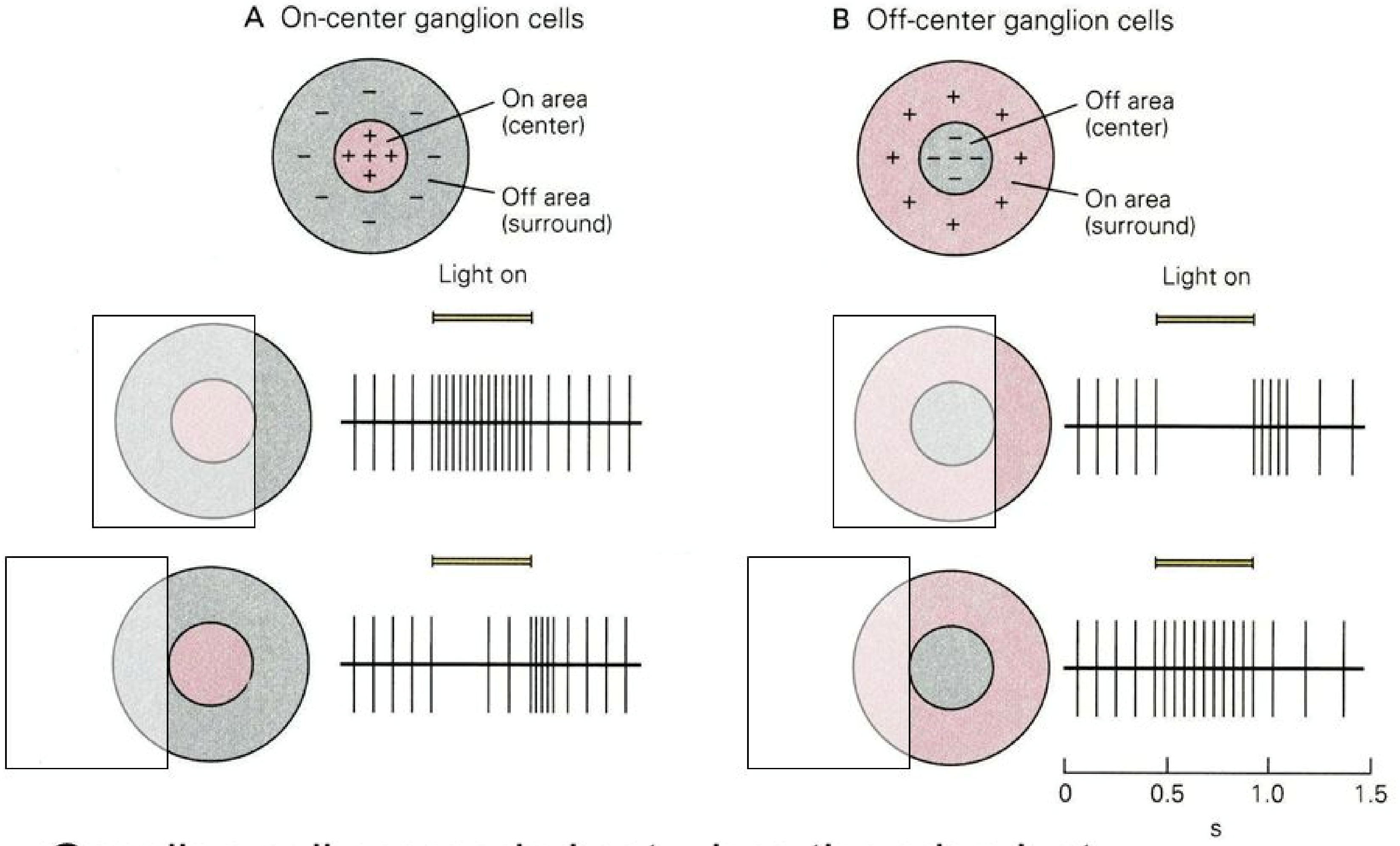
Properties of edge detectors: different orientation;

high frequency = fast changes, narrow bands;

low frequency = gentle changes, wide bands;

polarization = dark-light or vice-versa, dark-light-dark or vice-versa. 26

Edge detection begins in the retina



- Ganglion cell responds best when the edge just touches the central region of the receptive field

(Kandel et al)

Today's Objectives

Edge detection

- Compass edge operators
- Difference of Gaussians
- Laplacian operators
- Laplacian of Gaussian