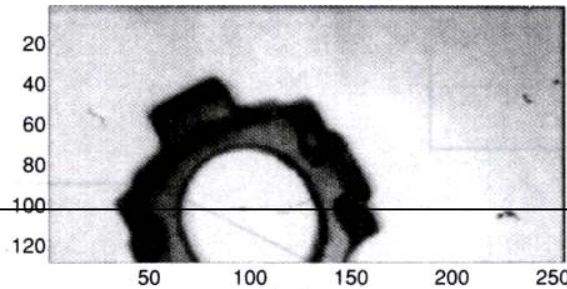


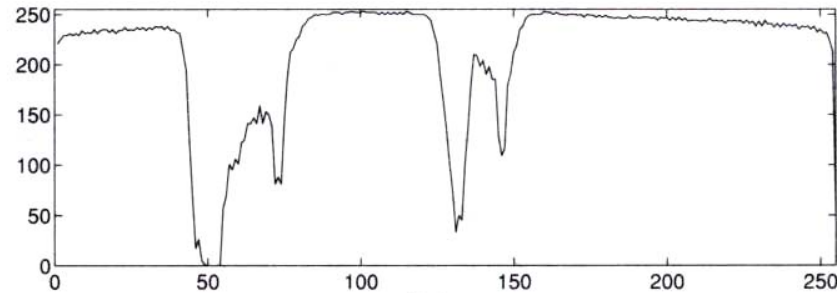
# Edge Detection

- **Edges**: local changes in the image intensity
- Edges typically occur on the boundary between two regions
- Important features for image analysis
- The changes due to noise are not edges
  - real images are very noisy
  - its very difficult to find edges in noisy images
- The changes due to shading are not edges

line of  
profile



(a)



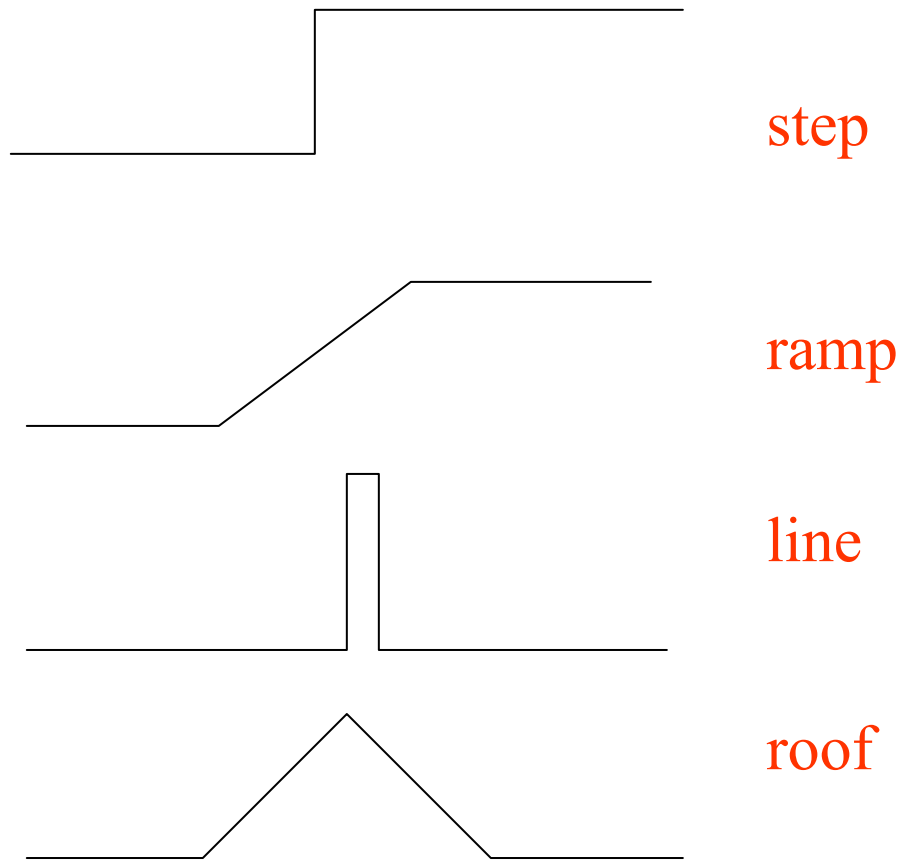
(b)

I. Original noisy image

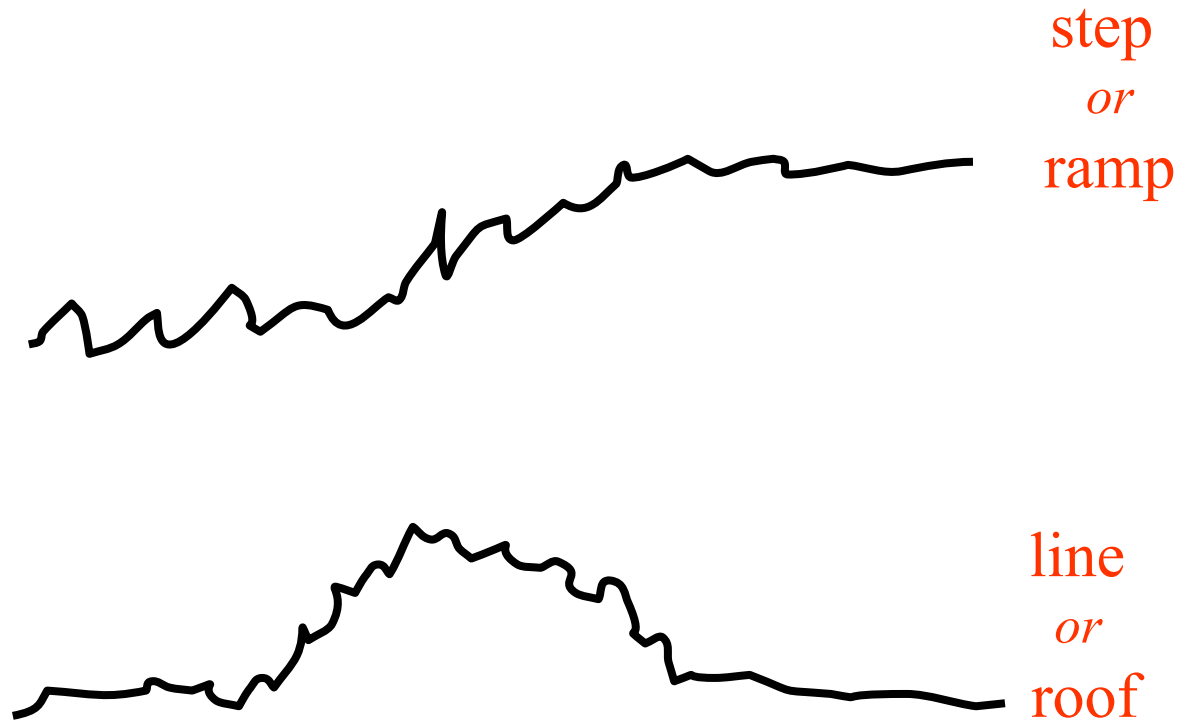
II. Edge profile

- The edges are not perfectly sharp

# Ideal Edges



# Noisy Edges



# Edge Detection

- An **edge point** is a point at the location of a local intensity change
- An **edge detector** is an algorithm that computes the edges in an image
- A **contour** is a list of edges or the curve that matches the list of edges
- **Edge following** is the process of searching the image to determine contours

# Edge Results

- The produced set of edges contain
  - **correct edges** corresponding to real - useful information e.g., object boundaries
  - **false edges** do not correspond to real edges e.g., noise
- There are also **missing edges**
  - **edge linking** may try to recover missing edges
  - edge following will form contours from existing and recovered edges

# First Derivative Operator

- **Gradient:** vector denoting the max rate of change of intensity  $f(x,y)$ :

$$\vec{\nabla}f = \frac{\mathcal{I}f}{\mathcal{I}x} \vec{i} + \frac{\mathcal{I}f}{\mathcal{I}y} \vec{j}$$

– direction

$$\theta = \tan^{-1} \left\{ \frac{\frac{\partial f}{\partial x}}{\frac{\partial f}{\partial y}} \right\}$$

– magnitude

$$\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}, \max \left\{ \left(\frac{\partial f}{\partial x}\right), \left(\frac{\partial f}{\partial y}\right) \right\} \dots$$

# Gradient

- For *any* vertical directions  $(x,y)$ ,  $(x',y')$  and image  $f(x,y)$ :

$$\vec{\nabla}f(x, y) = \vec{\nabla}f(x', y')$$

$$\theta = \tan^{-1} \left\{ \frac{\frac{\partial f}{\partial x}}{\frac{\partial f}{\partial y}} \right\} = \theta' = \tan^{-1} \left\{ \frac{\frac{\partial f}{\partial x'}}{\frac{\partial f}{\partial y'}} \right\}$$

$$\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} = \sqrt{\left(\frac{\partial f}{\partial x'}\right)^2 + \left(\frac{\partial f}{\partial y'}\right)^2}$$

# Digital Approximation of Gradient

- Compute

$$G_x \approx \Delta_x f(i, j) = f(i, j + 1) - f(i, j)$$

$$G_y \approx \Delta_y f(i, j) = f(i, j) - f(i + 1, j)$$

- Equivalently convolve with

$$G_x = \begin{bmatrix} -1 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

# Gradient Convolution Masks

- $G_x = \begin{bmatrix} -1 & 1 \end{bmatrix}$  computes edge point at  $(i+1/2, j+1/2)$
- $G_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$  computes edge point at  $(i+1/2, j)$
- 2x2 masks: edge point at  $(i+1/2, j+1/2)$

$$G_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

- edge points at center pixel  $(i, j)$  use 3x3 masks

$$G_x = \frac{1}{4} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

# Roberts operator

- Computes the edge at the interpolated point  $(i+1/2, j+1/2)$ 
  - magnitude:  $|G_x| + |G_y|$

$$G_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

# Sobel Operator

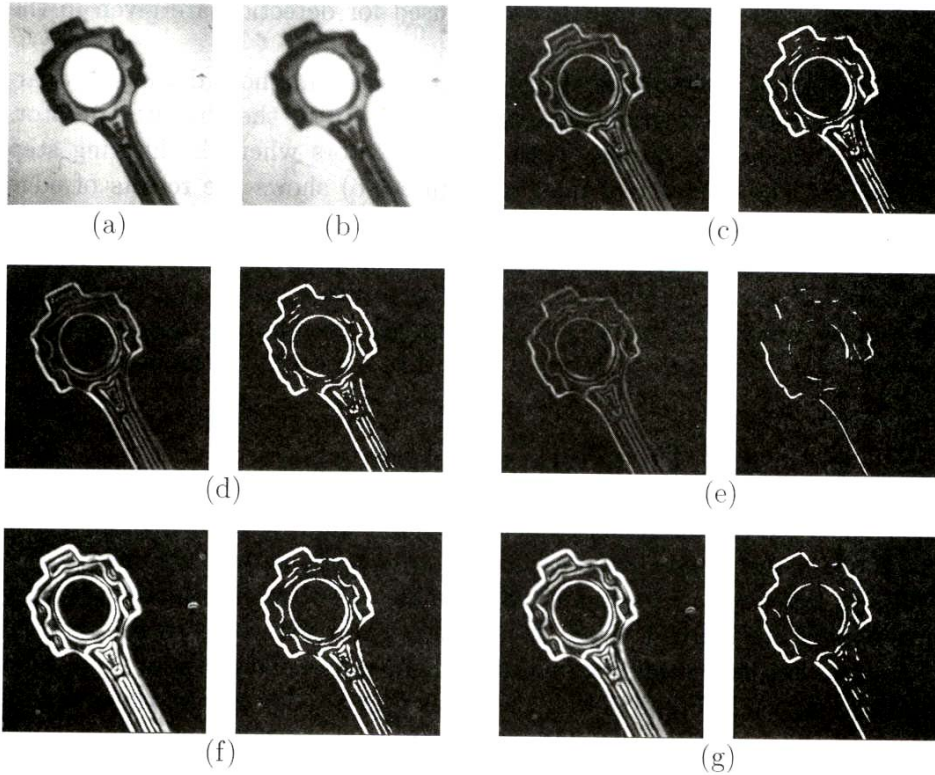
- One the most commonly used edge operators
  - computes the edge at (i,j)
  - Smoothing at the same time
  - places emphasis on pixels closest to (i,j)

$$G_x = \frac{1}{4} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

# Prewit Operator

- Similar to Sobel but does not place any emphasis on pixels

$$G_x = \frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

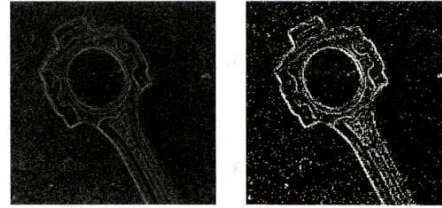


Edge operators  
with smoothing  
first

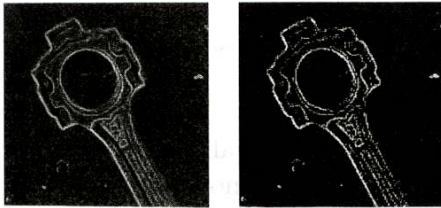
- a. Original image
- b. Smoothed image
- c. Gradient with 1x2,2x1 masks,  $T=32$
- d. Gradient with 2x2 masks,  $T=64$
- e. Roberts operator,  $T=64$
- f. Sobel operator,  $T=225$
- g. Prewit operator,  $T=225$



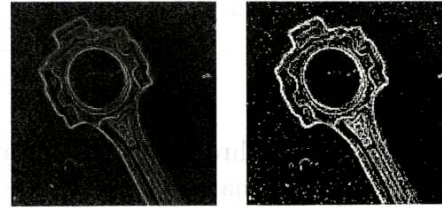
(a)



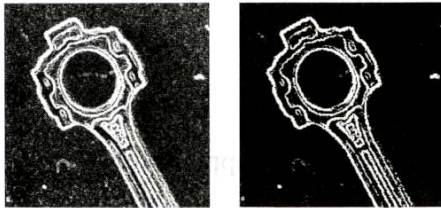
(b)



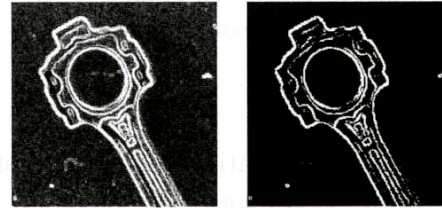
(c)



(d)



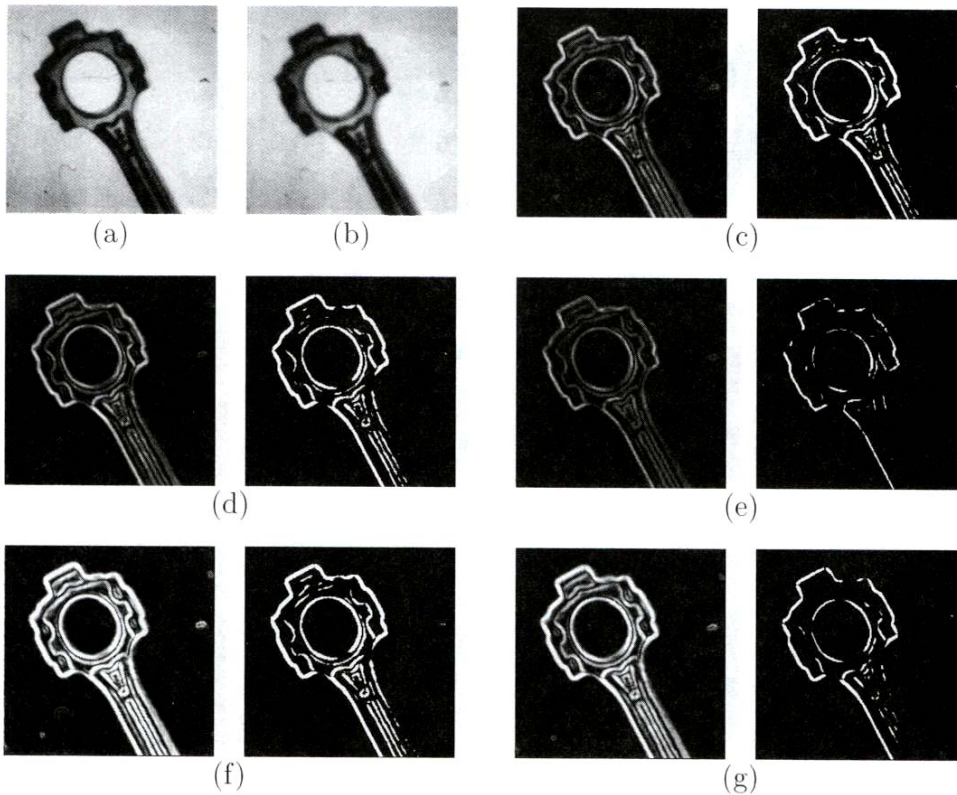
(e)



(f)

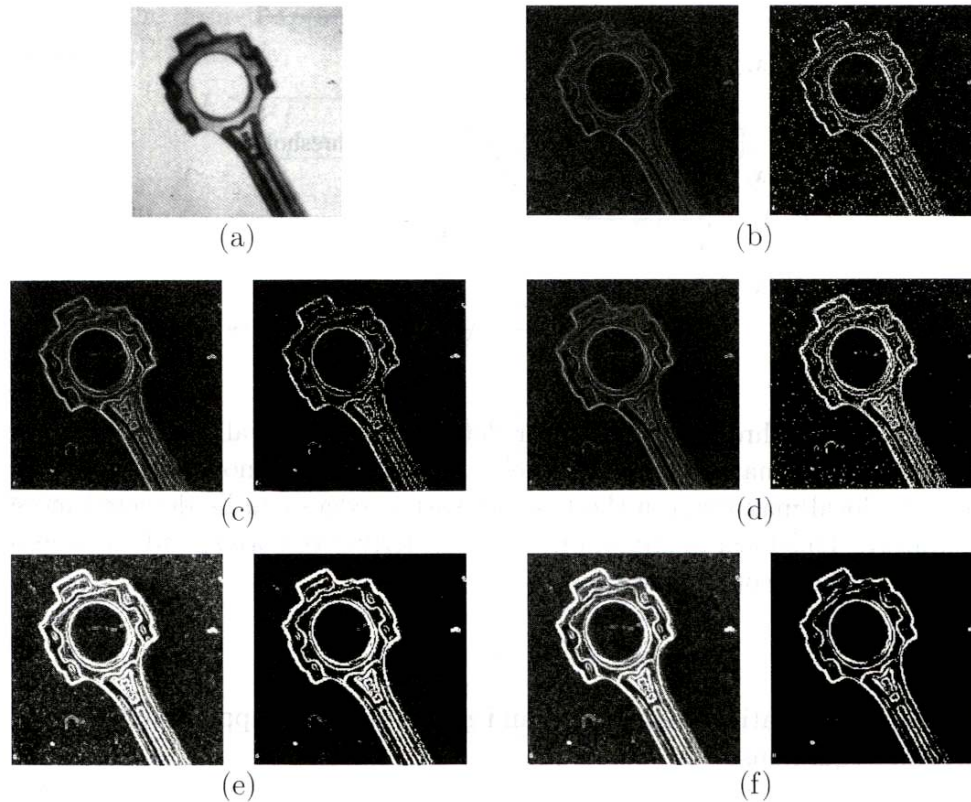
Edge operators  
without  
smoothing

- a. Original image
- b. Simple Gradient with 1x2,2x1 masks,  $T=64$
- c. Gradient with 2x2 masks,  $T=64$
- d. Roberts operator,  $T=64$
- e. Sobel operator,  $T=225$
- f. Prewitt operator,  $T=225$



Edge operators  
with smoothing  
on noisy image

- a. Noisy image
  - b. Filtered image
  - c. Simple Gradient with 1x2,2x1 masks, T=32
  - d. Gradient with 2x2 masks, T=64
  - e. Roberts operator, T=64
  - f. Sobel operator, t=225
  - g. Prewitt operator, T=225
- Edge Detection



Edge detection  
on noisy image  
without smoothing

- a. Noisy image
- b. Simple Gradient using  $1 \times 2, 2 \times 1$  masks,  $T=64$
- c. Gradient using  $2 \times 2$  masks,  $T=128$
- d. Roberts operator,  $T=64$
- e. Sobel operator,  $T=225$
- f. Prewitt operator,  $T=225$

# Edge Detection with Edge Patterns

- Comparison (filtering) with 4 or 8 edge masks
  - each one represents an edge orientation
  - combine their results
  - the edge pattern with the max magnitude represents the magnitude and the orientation of the edge
  - thresholding is still necessary!
- **Optimization:** apply 4 of the 8, the other 4 are symmetric
  - give opposite orientations and negative values of magnitude
  - take the absolute value of negative edge strengths

# Prewitt Edge Patterns

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

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

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

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

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

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

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

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

arrow denotes  
orientation of edge  
detection

# Kirsch edge patterns

$$\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}$$

$$\begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}$$

$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}$$

$$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

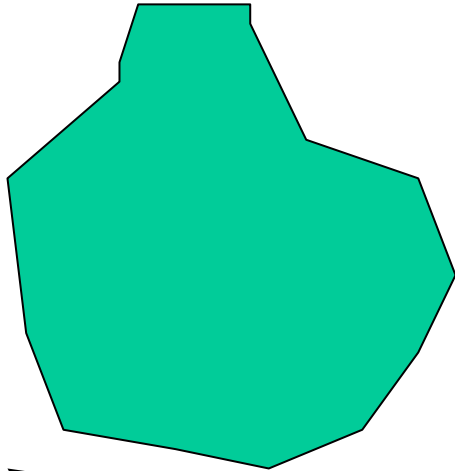
$$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}$$

$$\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$$

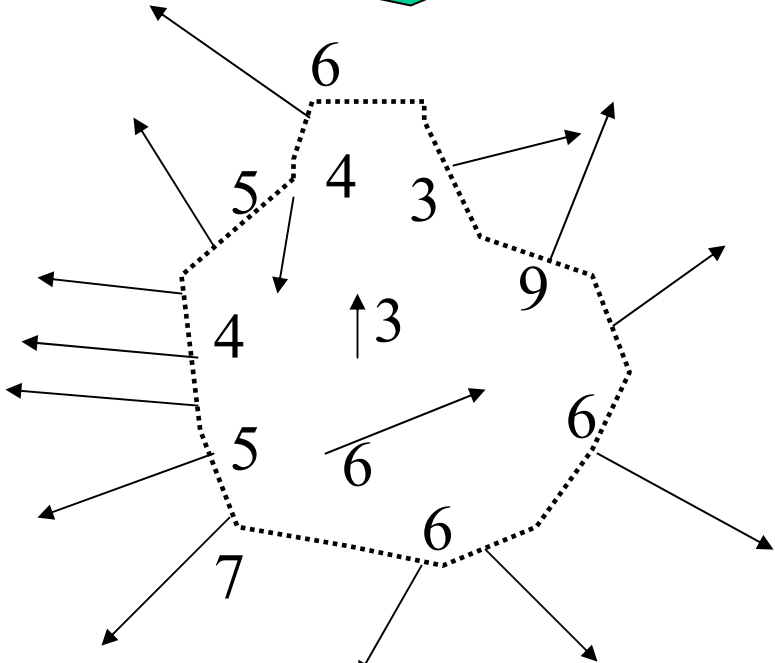
arrow denotes  
orientation of edge  
detection

# Steps in Edge Detection

1. **Filtering** removes noise and improves the performance of edge detection
  - there is a trade off between edge strength and noise reduction
  - filtering smoothes the edge too!!
2. **Edge enhancement** apply a Gradient operator
3. **Detection** keeps only edge points and eliminates false edge points
  - keep points with strong edge content
  - above a threshold **T**
4. **Localization** computes location and orientation of edge



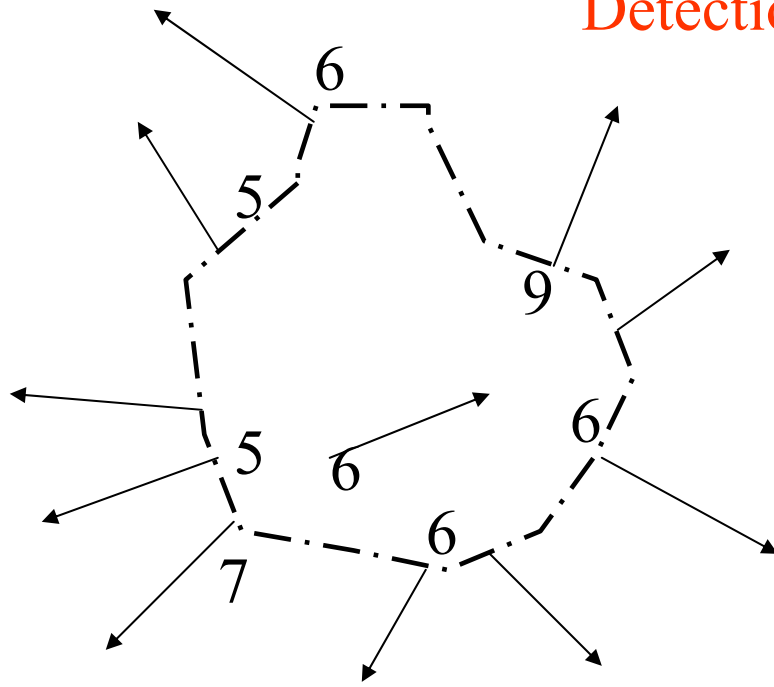
original image



gradient

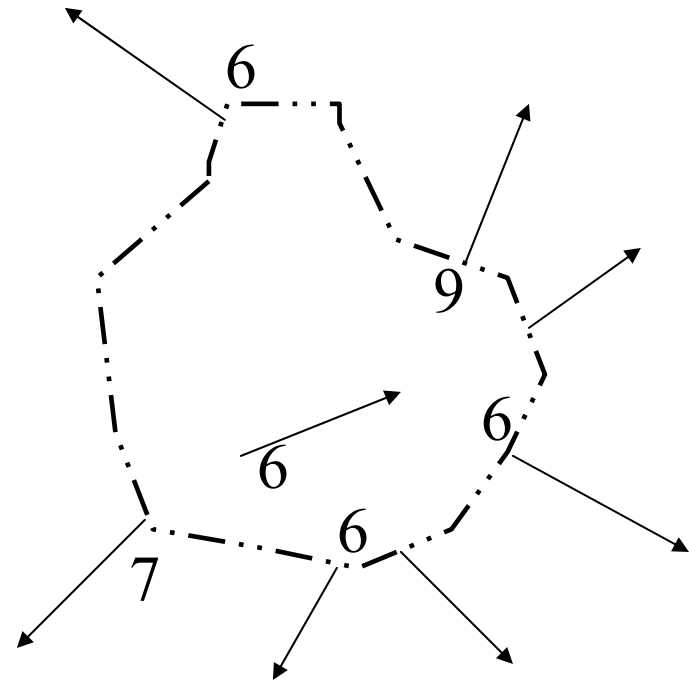
Edge Detection

## Detection and Localization



threshold  $T=4$   
keep edges points  
with strength  $> 4 \rightarrow$   
noise is retained

threshold  $T=5$   
keep edges points  
with strength  $> 5 \rightarrow$   
removes noise but significant  
parts of the contour are lost



# Smoothing & Thresholding

- **Smoothing:**
  - high degree of smoothing (large mask, repeated smoothing) removes noise but blurs the edges
  - low degree of smoothing retains noise
- **Threshold selection:**
  - large thresholds remove noise along with edge points
  - low thresholds retain noise

# Second Derivative Operators

- **Edge points**: peaks of the first derivative
  - use of thresholds
  - detects too many edge points
- Equivalently, zero's of the second derivative
- Combine first and second derivatives
  - locate **peak** in the first derivative and **zero crossing** in the second derivative

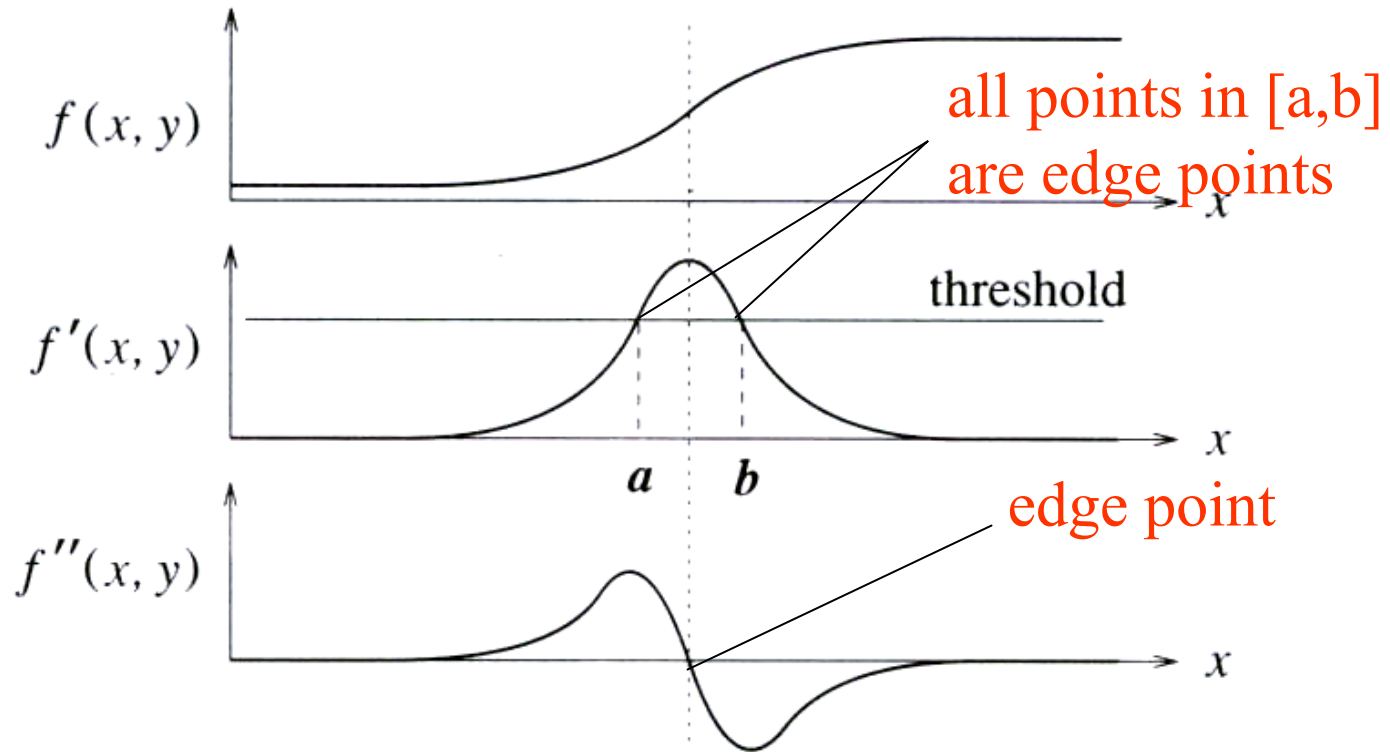
# Combination of Derivatives

- Edge detection
  - smoothing by Gaussian
  - presence of zero crossing in second derivative

$$\vec{\nabla}^2 f[i, j] \approx 0$$

- large peak in the first derivative

$$|\vec{\nabla} f[i, j]| > \textit{Threshold}$$



# Laplacian Operator

- Two dimensional equivalent of the second derivative

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

- Digital approximation

$$\nabla^2 f(i, j) \approx f(i+1, j) + f(i-1, j) + f(i, j+1) + f(i, j-1) - 4f(i, j)$$

# Proof

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{\partial G_x}{\partial x} = \frac{\partial(f[i, j+1] - f[i, j])}{\partial x} = \\ &= \frac{\partial f[i, j+1]}{\partial x} - \frac{\partial f[i, j]}{\partial x} = \\ &= (f[i, j+2] - f[i, j+1]) - (f[i, j+1] - f[i, j]) = \\ &= f[i, j+2] - 2f[i, j+1] + f[i, j]\end{aligned}$$

## Proof (2)

- Second derivative on X is centered around  $[i, j+1]$ ,

– replace  $j$  with  $j-1$  to center edge around  $[i, j]$

$$\frac{\partial^2 f}{\partial x^2} = f[i, j+1] - 2f[i, j] + f[i, j-1]$$

- Similarly for Y

$$\frac{\partial^2 f}{\partial y^2} = f[i+1, j] - 2f[i, j] + f[i-1, j]$$

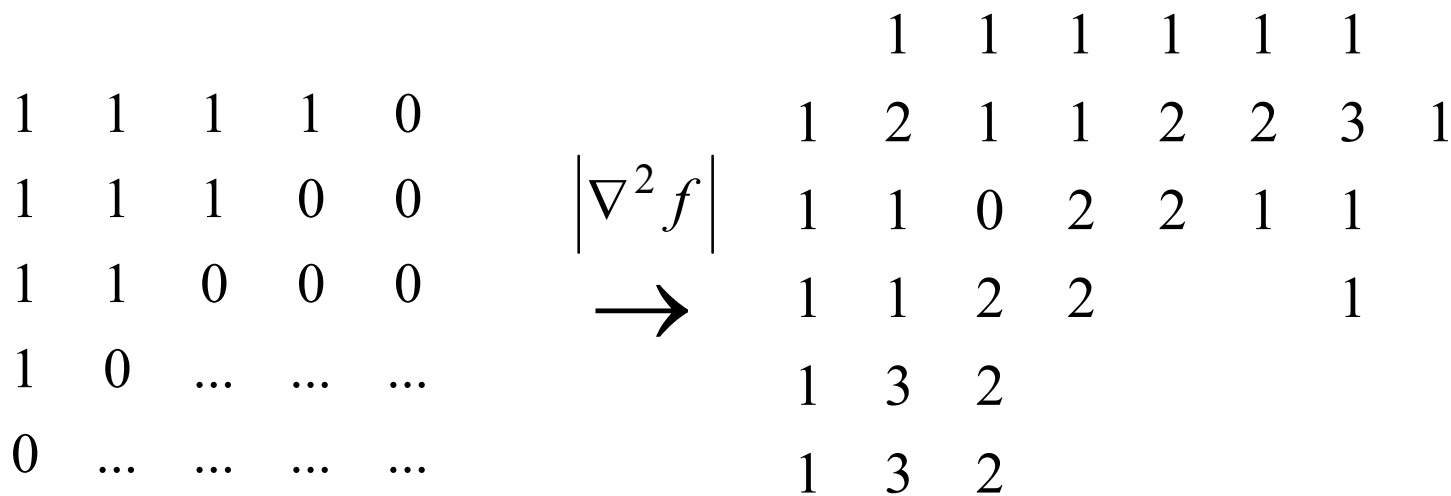
# Laplacian Mask

- The following mask approximates the Laplacian (both x,y)

$$\nabla^2 f[i, j] \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

# Response of Laplacian

- Zero's signal the presence of an edge
  - trivial zeros (uniform regions) are ignored
  - strong response to edges, lines, points, **noise!**



# Response to Step Edges

edge lies between 6 and -6



2	2	2	2	8	8	8	8
2	2	2	2	8	8	8	8
2	2	2	2	8	8	8	8
2	2	2	2	8	8	8	8
2	2	2	2	8	8	8	8
2	2	2	2	8	8	8	8

$$|\nabla^2 f|$$

0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0
0	0	0	6	-6	0	0	0

- The edge point (zero crossing location) may need to be approximated to sub-pixel resolution

# Response to Ramp Edges

2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8
2	2	2	2	5	8	8	8	8

$|\nabla^2 f|$   
→

0	0	0	3	0	-3	0	0	0
0	0	0	3	0	-3	0	0	0
0	0	0	3	0	-3	0	0	0
0	0	0	3	0	-3	0	0	0
0	0	0	3	0	-3	0	0	0
0	0	0	3	0	-3	0	0	0

↑  
edge

# Laplacian of Gaussian

- The second derivative responds strongly to noise
- It is desired to filter the noise before edge detection
- The Laplacian of a Gaussian (**LoG**) combines Gaussian filtering with Laplacian for edge detection

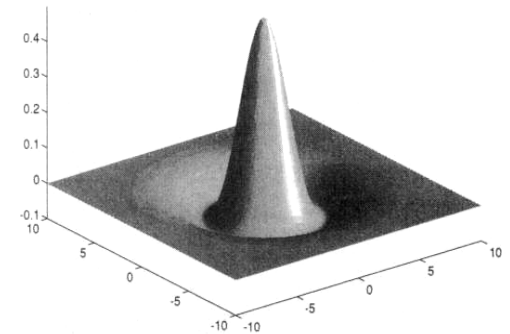
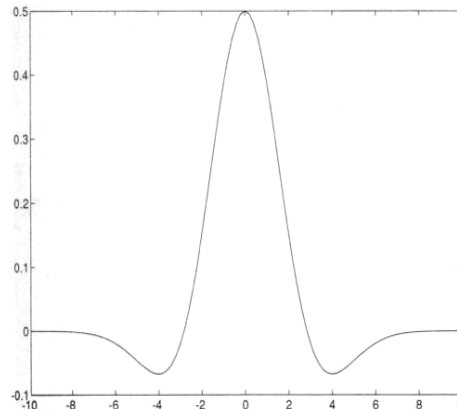
# One-Step LoG

- Notice that  $\nabla^2(G * f) = (\nabla^2 G) * f$

- In practice **LoG** is applied in one step by filtering with

$$\nabla^2 G(r) = -\frac{1}{\pi\sigma^4} \left(1 - \frac{r^2}{2\sigma^2}\right) \exp\left\{-\frac{r^2}{2\sigma^2}\right\}$$

- Known also as **Mexican hat operator**



# Discrete Approximation of LoG

5 × 5 Laplacian of Gaussian mask

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

17 × 17 Laplacian of Gaussian mask

0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0
0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-3	-2	-1	-1	-1	0	0
0	0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-3	-2	-1	-1	0	0
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1	0
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1	0
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1	-1
-1	-1	-3	-3	-3	4	12	21	24	21	12	4	-3	-3	-3	-1	-1
-1	-1	-3	-3	-2	2	10	18	21	18	10	2	-2	-3	-3	-1	-1
-1	-1	-3	-3	-3	0	4	10	12	10	4	0	-3	-3	-3	-1	-1
0	-1	-2	-3	-3	-3	0	2	4	2	0	-3	-3	-3	-2	-1	0
0	-1	-1	-2	-3	-3	-3	-2	-3	-2	-3	-3	-3	-2	-1	-1	0
0	0	-1	-1	-2	-3	-3	-3	-3	-3	-3	-2	-1	-1	-1	0	0
0	0	-1	-1	-1	-2	-3	-3	-3	-3	-3	-2	-1	-1	-1	0	0
0	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0
0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0

# LoG Smoothing

- $\sigma$  controls the **degree of smoothing**
  - large  $\sigma$  results in better noise filtering but also
  - blurs and displaces edges
  - small  $\sigma$  results in too many noise points and false edges
  - edge points are better localized using small  $\sigma$
- The best size of the filter is unknown
- **Scale space filtering**: combine results from multiple filters with varying
  - resembles filtering on the retina

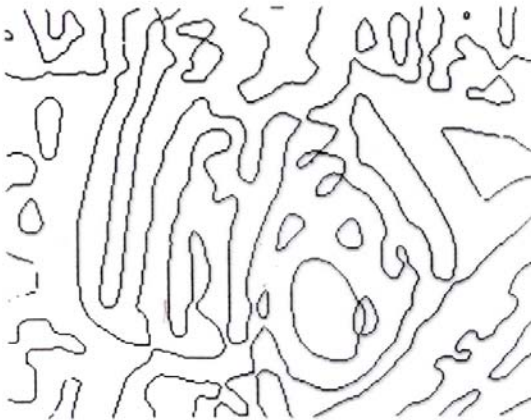
# LoG filtering with varying $\sigma$



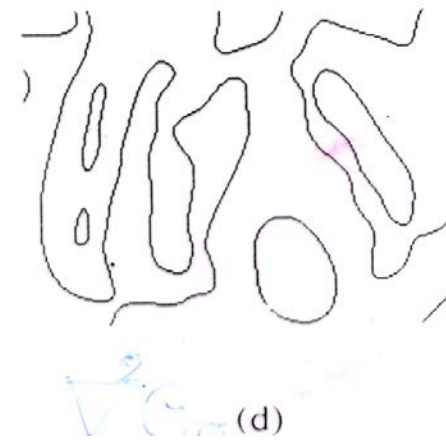
(a)



(b)



(c)

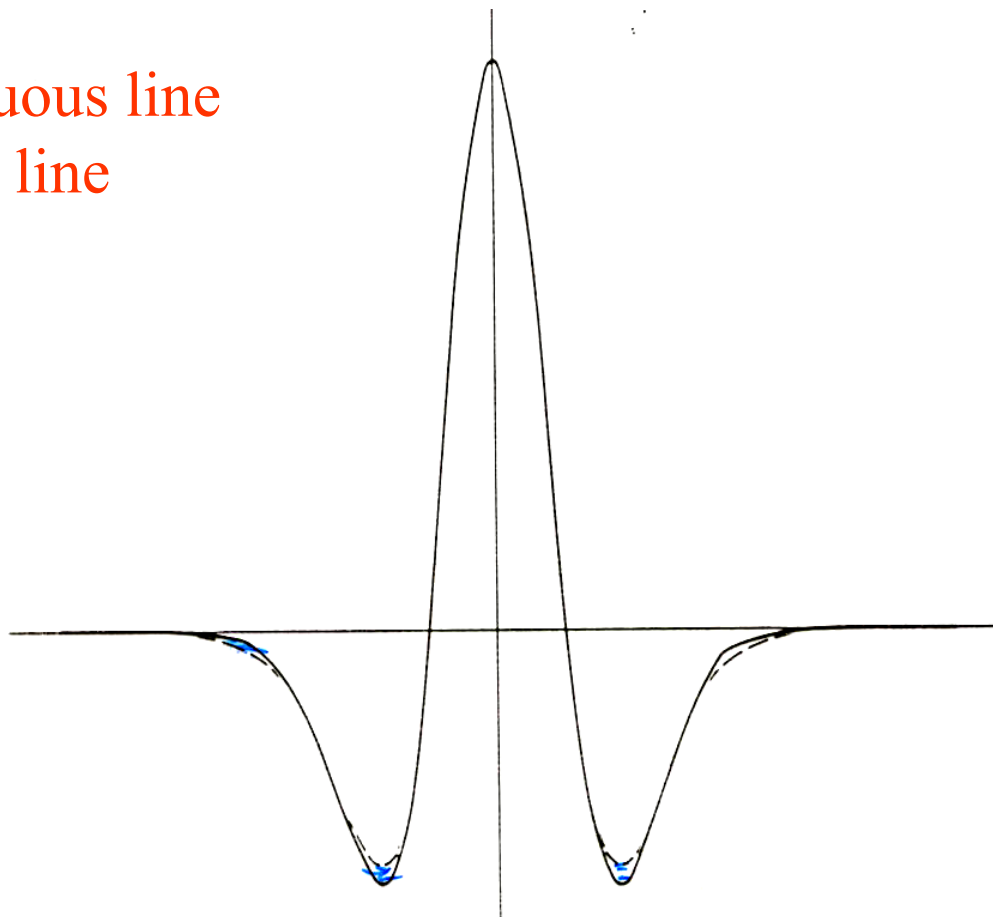


(d)

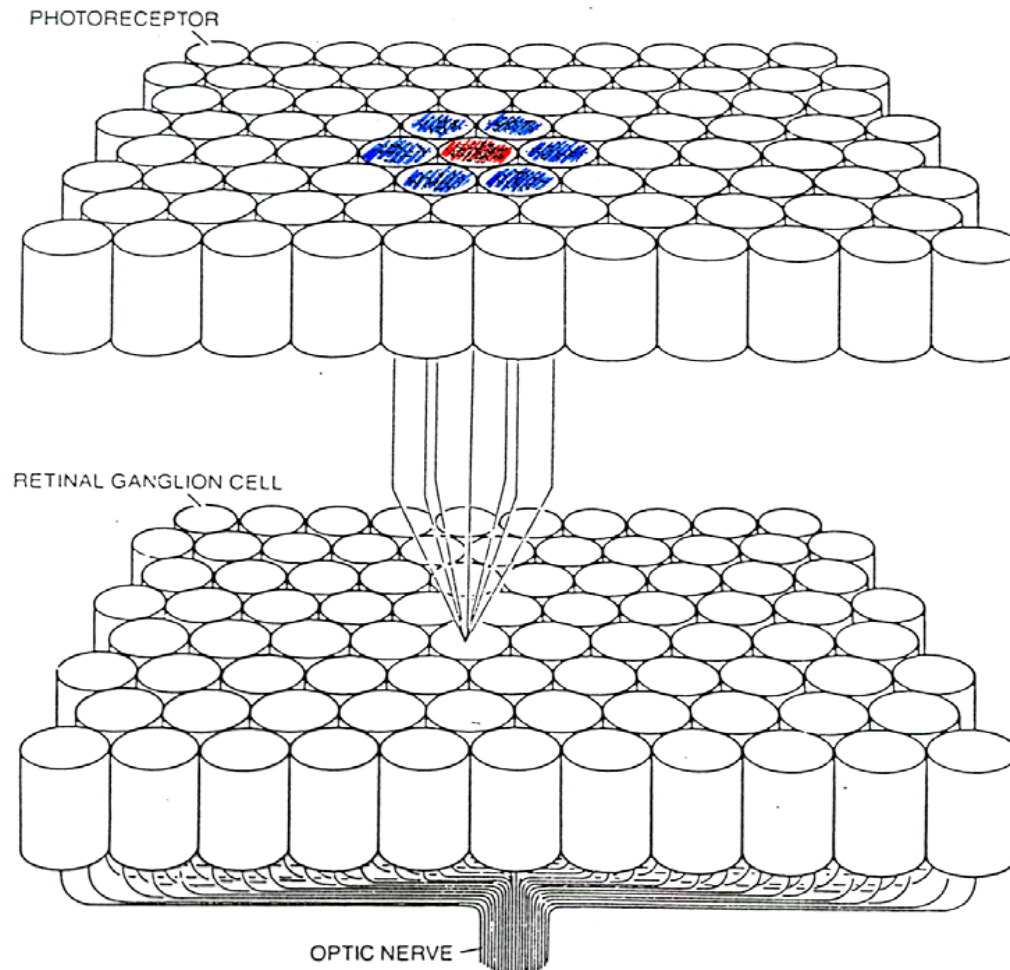
# DoG Approximation of LoG

- The best engineering approximation of LoG is obtained by taking the **Difference of two Gaussians** (DoG) with
  - amplitude ratio 1:1.16
- The same mechanism operates in the retina
  - many filters with varying  $\sigma$

DoG: continuous line  
LoG: dashed line



# DoG Filtering on the Retina



# “Center-Surround” Filtering

- Filtering by cells in retina resemble the effects of DoG
  - the photoreceptors are connected to retinal ganglion cells which send visual data to the brain
  - “Red” photoreceptors excite “*ON center*” ganglion cells (generate neural signals)
  - “Blue” photoreceptors in its surrounding inhibit the ganglion cell “*OFF surround*”