

## ***CHAPTER 3***

---

# **AREA PROCESSES**

Area processes use the input pixel as well as the pixels around it to generate a new output pixel.

# Convolution

- Mathematical Formulation

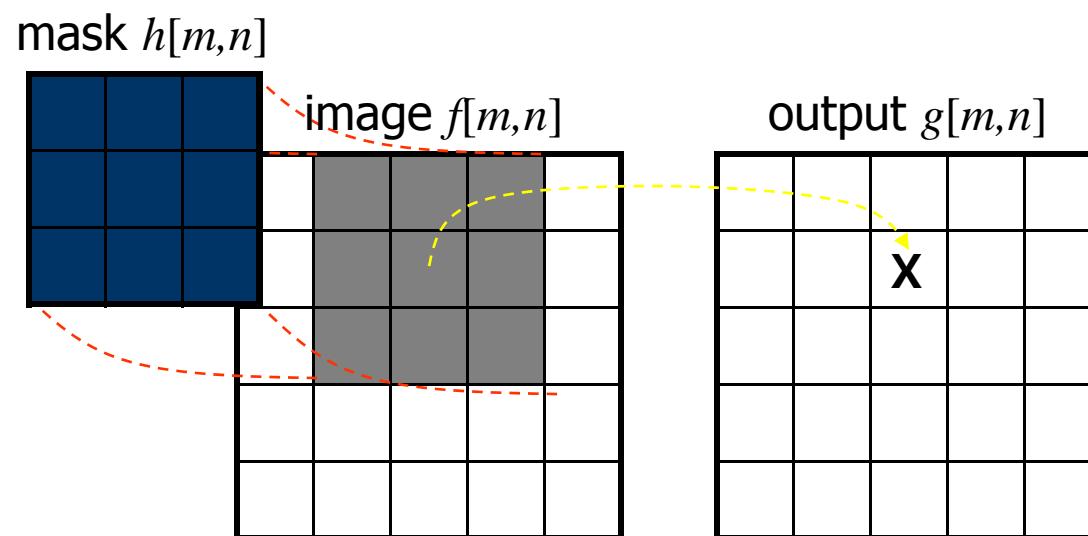
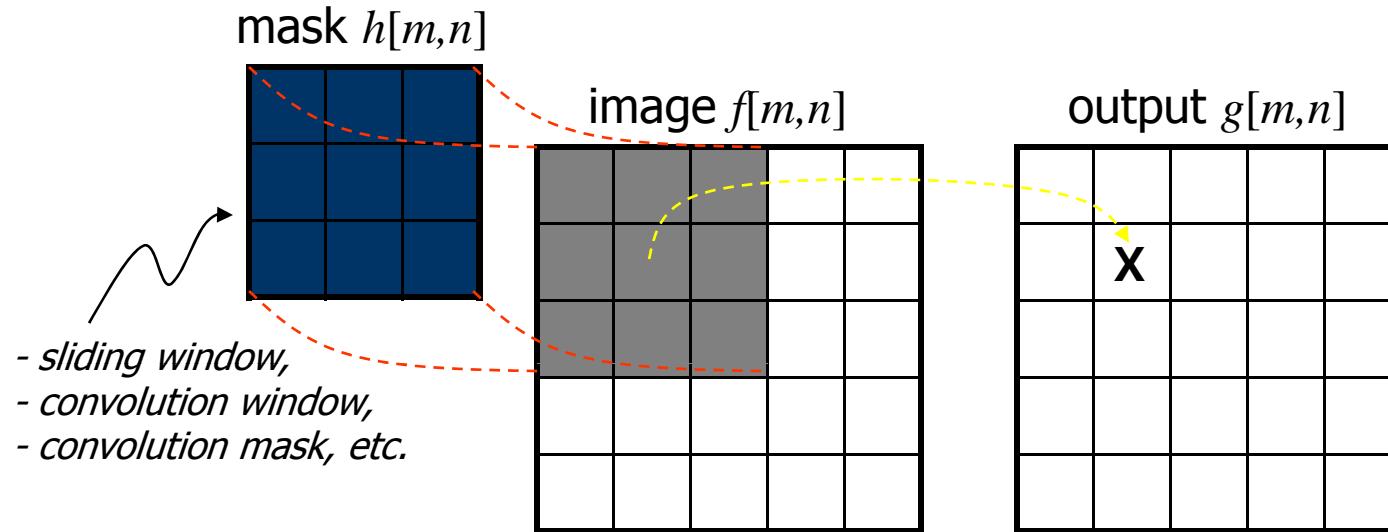
$$g(x, y) = h(x, y) * f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x-x', y-y') f(x', y') dx' dy'$$

$$g[m, n] = h[m, n] * f[m, n] = \sum_{m'=-\infty}^{\infty} \sum_{n'=-\infty}^{\infty} h[m-m', n-n'] f[m', n']$$

- Convolution in image processing consists of the following processes:
  - Overlay the mask  $h[m, n]$  on the image  $f[m, n]$
  - Multiply the coincident terms
  - Sum all the results
  - Move to the next pixel, across the entire image

# ***Convolution***

---



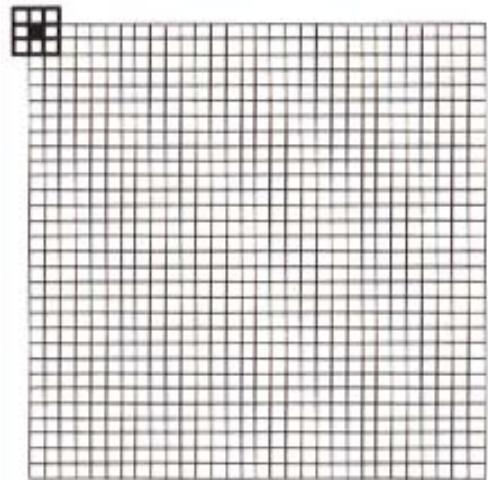
# ***Convolution***

---

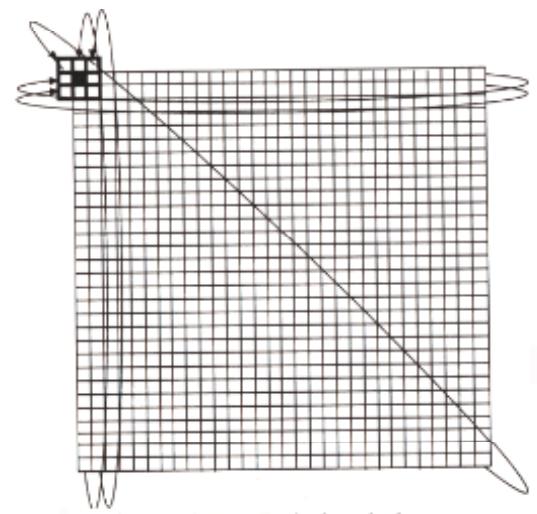
- How to treat the image borders
  - Zero padding: Treat the empty cells in the convolution window as zeros.
  - Skipping border pixels: Start convolving with the pixel at (1,1) instead of the pixel at (0,0).
  - Enlarging image before convolving: Duplicate border pixels.
  - Wrapping around: Assume the image is periodic in both horizontal and vertical axes.

# *Convolution*

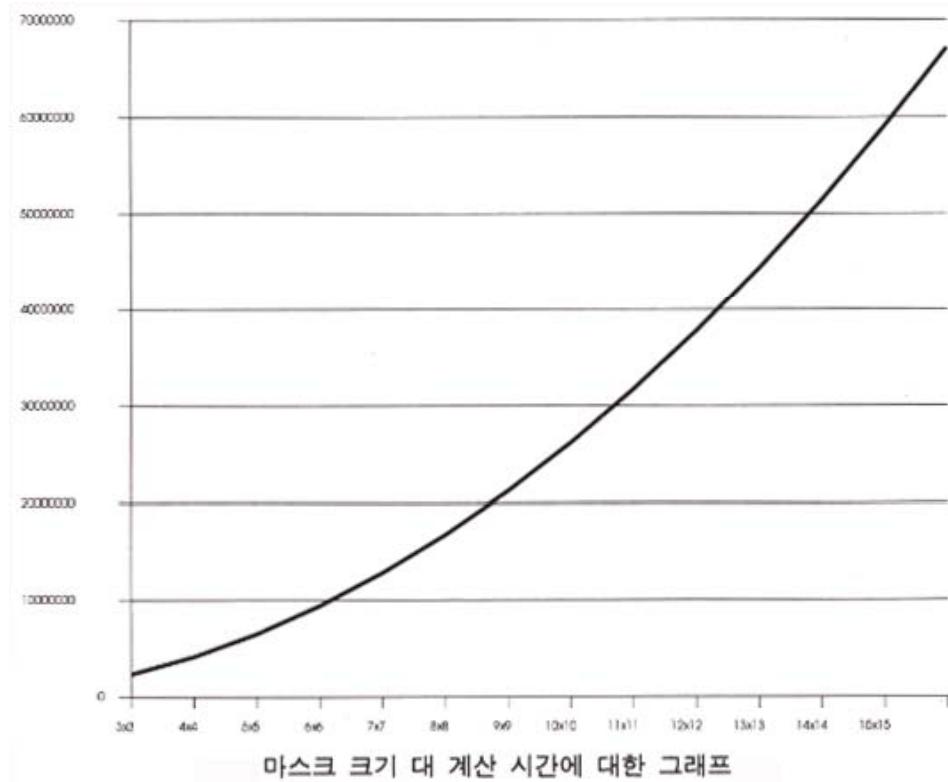
---



영상에 중첩한 희선 마스크



에지 주위를 둘러싼 희선 마스크



# ***Convolution***

---

- Separable Masks
  - Separable function:  $f(x,y) = g(x)h(y)$
  - The convolution can be performed by executing two convolutions with 1-D masks.
  - Separable functions reduce the number of computations required when using large masks.

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}}_{\text{6 multiplications}} \underbrace{(1 \quad 2 \quad 1)}_{\text{9 multiplications}}$$

- Savings increase as mask sizes grow.

# ***Convolution***

---

- Embossing
  - The coefficients of the mask have a center weight of 0 and sum to 0.
  - The key feature is the canceling coefficients about the center coefficients.

-1	0	0
0	0	0
0	0	1



# Blurring

- Blurring (Lowpass Spatial Filtering)
  - Removes the finer details of an image.
  - Reduces high-frequency noise.

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \frac{1}{25} \begin{array}{|c|c|c|c|c|} \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline \end{array}$$

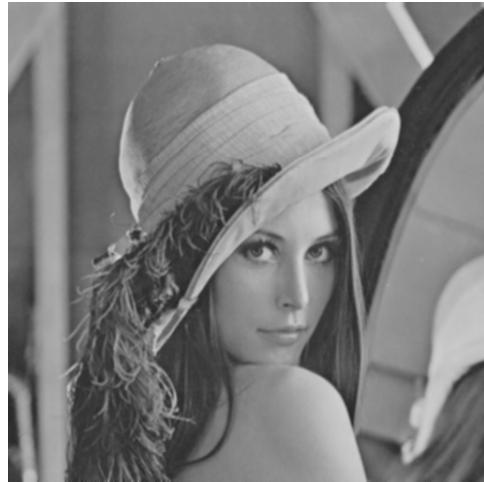
- Blurring is simply neighborhood averaging.
- Averaging tends to remove extreme values from a group.

# ***Blurring***

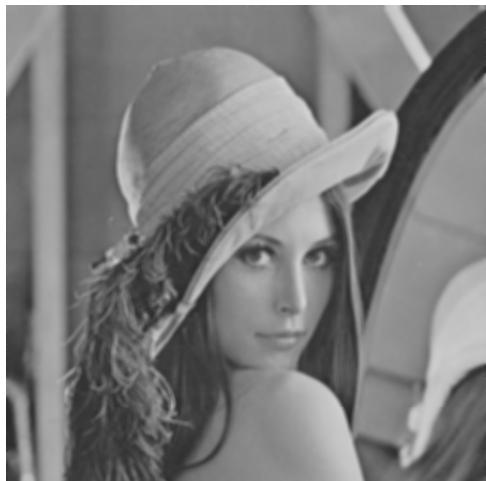
---



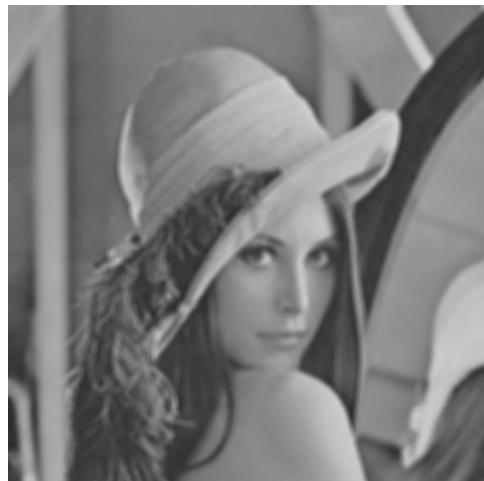
original image



3X3 blurring



5X5 blurring



7X7 blurring

# *Blurring*

---



original image with noise



3X3 blurring



5X5 blurring

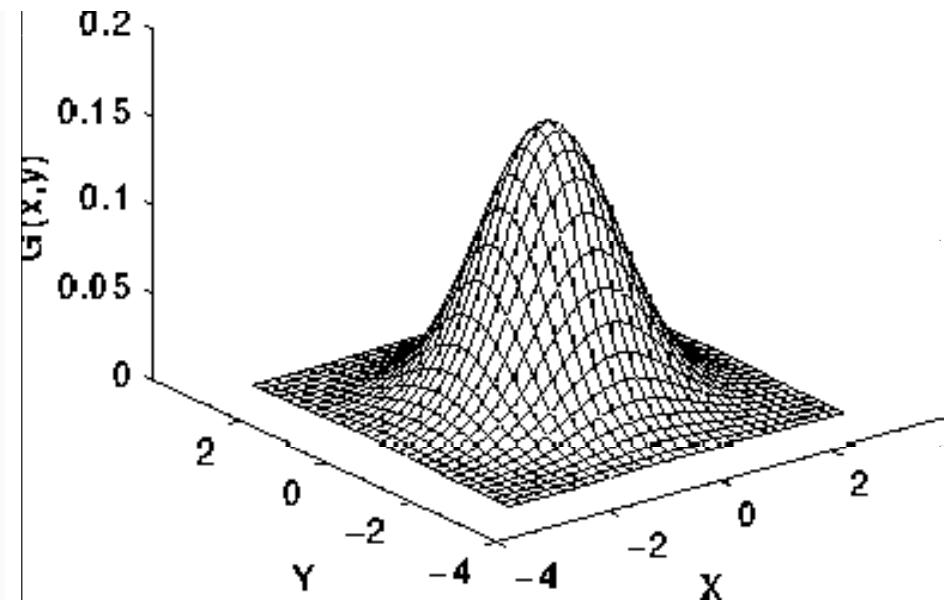
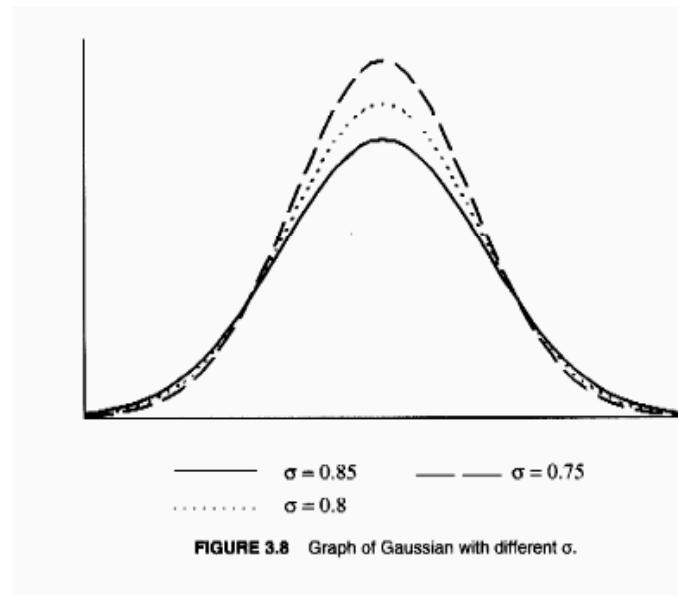


7X7 blurring

# Gaussian Smoothing

- Gaussian Smoothing Filters

$$G(x, y) = G(x)G(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \times \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{y^2}{2\sigma^2}} = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



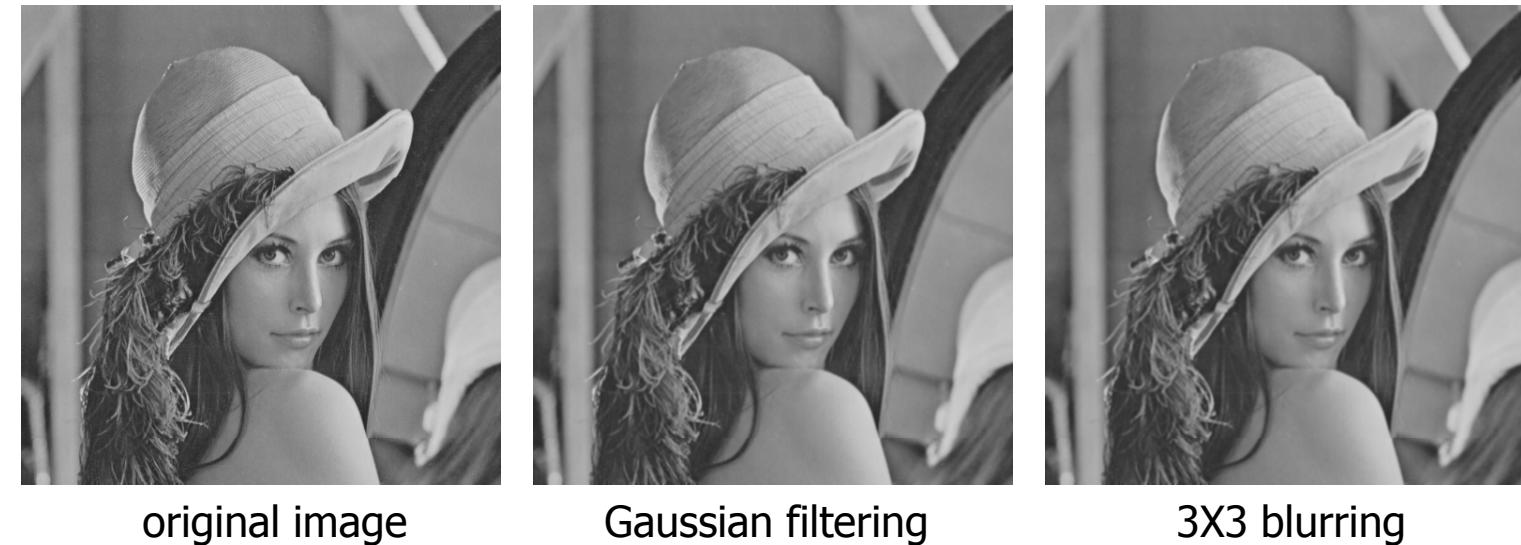
# Gaussian Smoothing

- Mask Approximation for Gaussian Smoothing

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

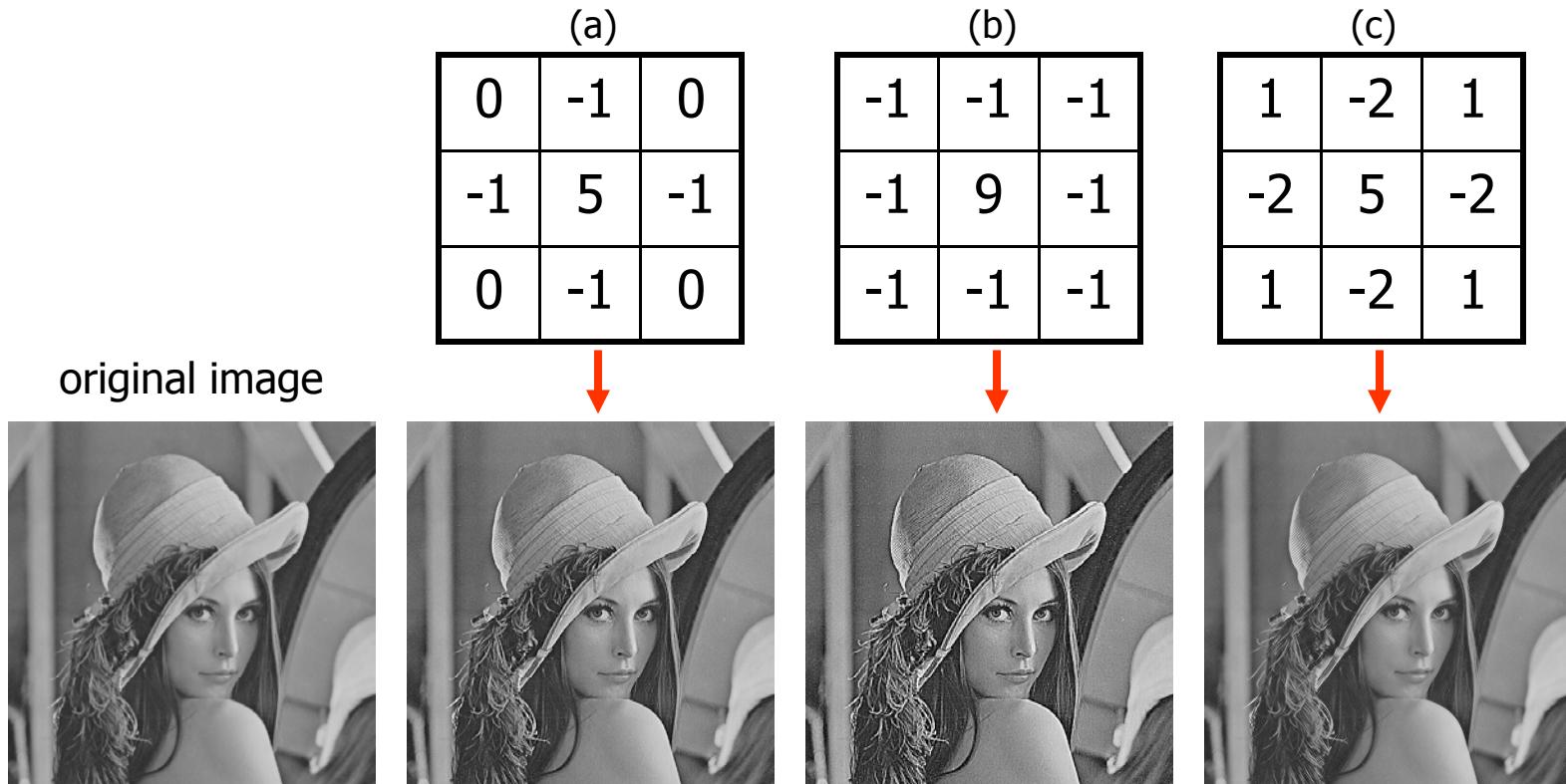
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1



# Sharpening

- Sharpening has the opposite affect of blurring.
- Emphasizes the details in an image.
- The convolution mask has a positive coefficient in its center and mostly negative coefficients around the outer edge.



# Sharpening



original image



sharpened by (a)



sharpened by (b)



sharpened by (c)

# Sharpening

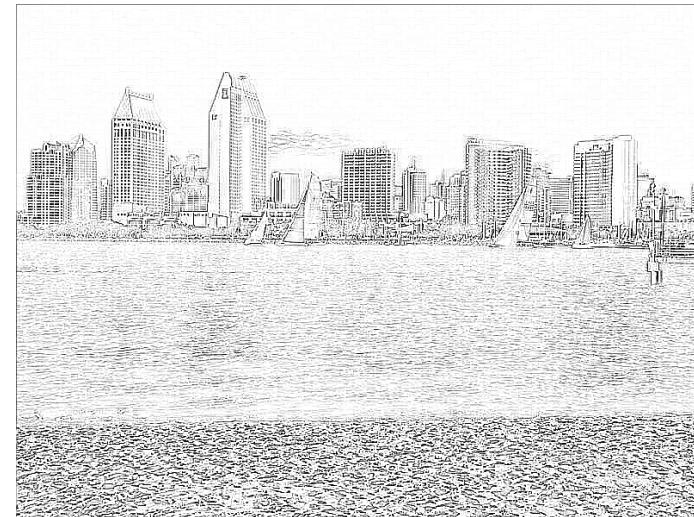
- Sharpening is based on the highpass filter.
- A common highpass filter mask:

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Note: coefficients sum to 0



original image



highpass-filtered image

# Sharpening

- Unsharp Masking (a method of highpass filtering)
  - $[\text{highpass}] = [\text{original}] - [\text{lowpass}]$
- High-Boost Filtering
  - $[\text{High Boost}] = \alpha [\text{original}] - [\text{lowpass}]$

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & \alpha & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad -\frac{1}{9} \quad \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad = \frac{1}{9} \quad \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & w & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

$$\alpha - \frac{1}{9} = \frac{w}{9} \quad \text{or} \quad w = 9\alpha - 1$$

- $\alpha=1$ : highpass filter
- $\alpha>1$ : a fraction of original is added back to highpass result

# Sharpening



## Edge Detection

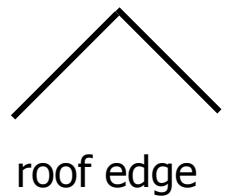
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels



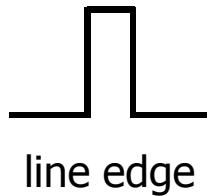
- How can you tell that a pixel is on an edge?

## Edge Detection

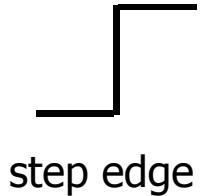
- Intensity changes rapidly around edges.
- Different edge profiles:



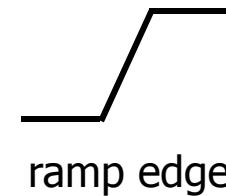
roof edge



line edge



step edge



ramp edge

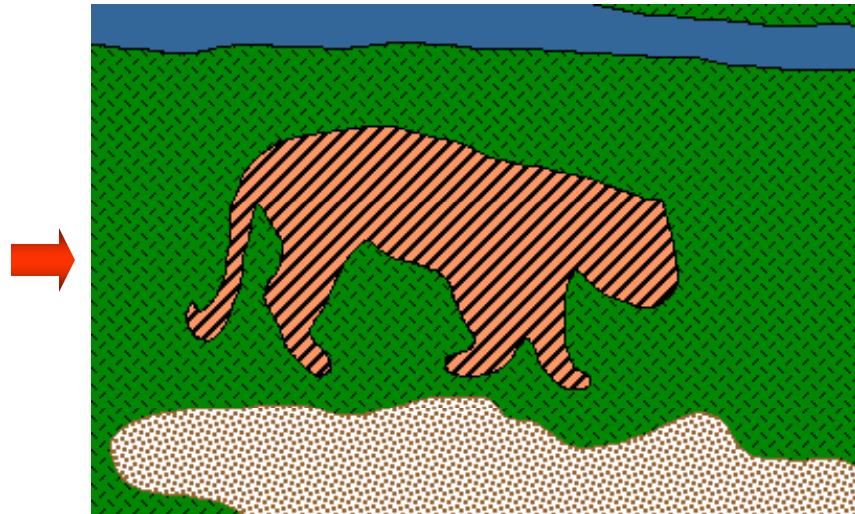
- The edges tell:
  - Where objects are
  - Their shape and size
  - Something about their texture

## Edge Detection

- The applications include image segmentation and registration:
  - Image segmentation: groups pixels into regions to determine an image's composition.



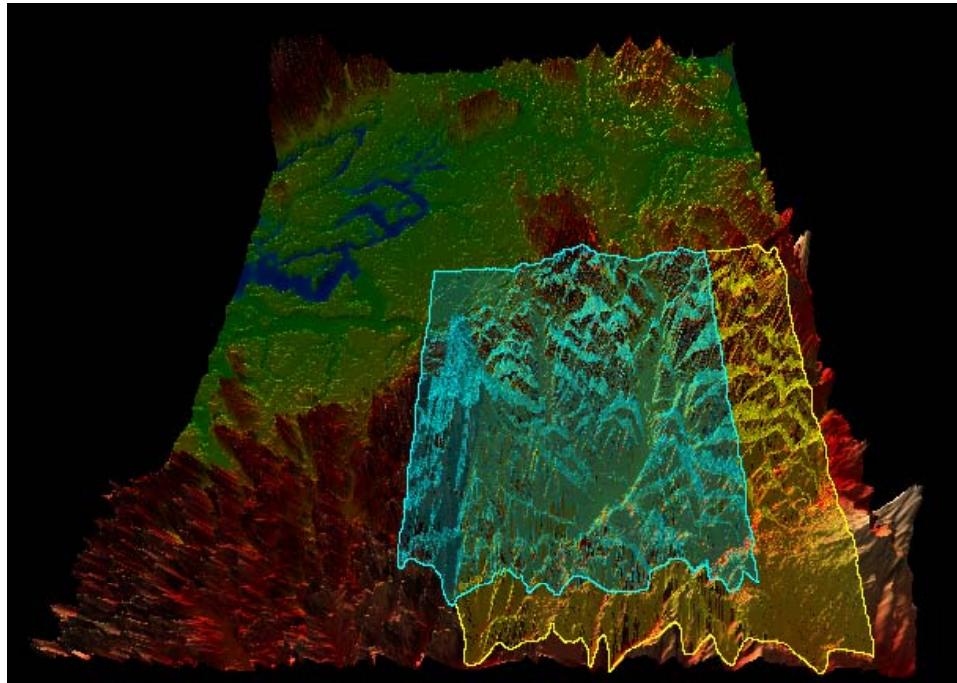
original image



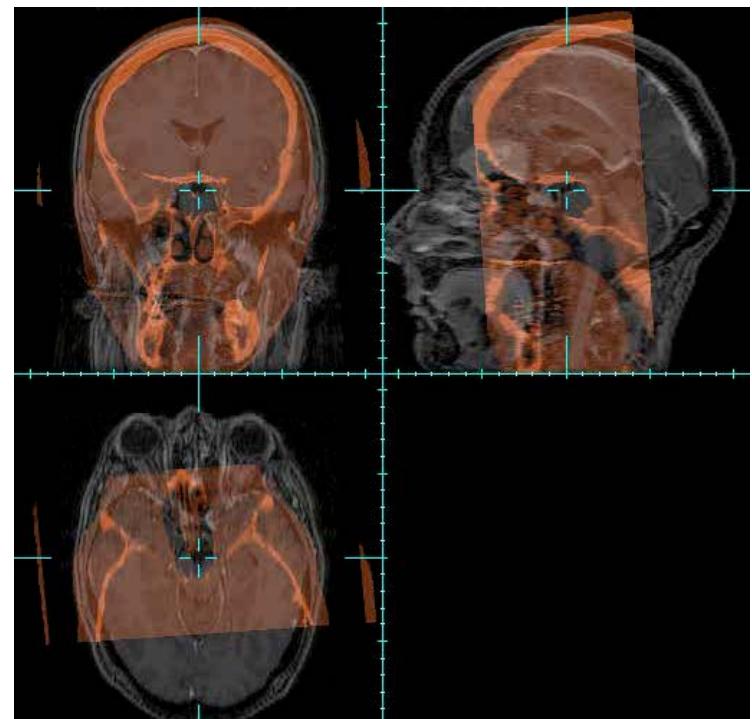
segmented image

## Edge Detection

- Image registration: aligns two images that may have been acquired at separate times or from different sensors.



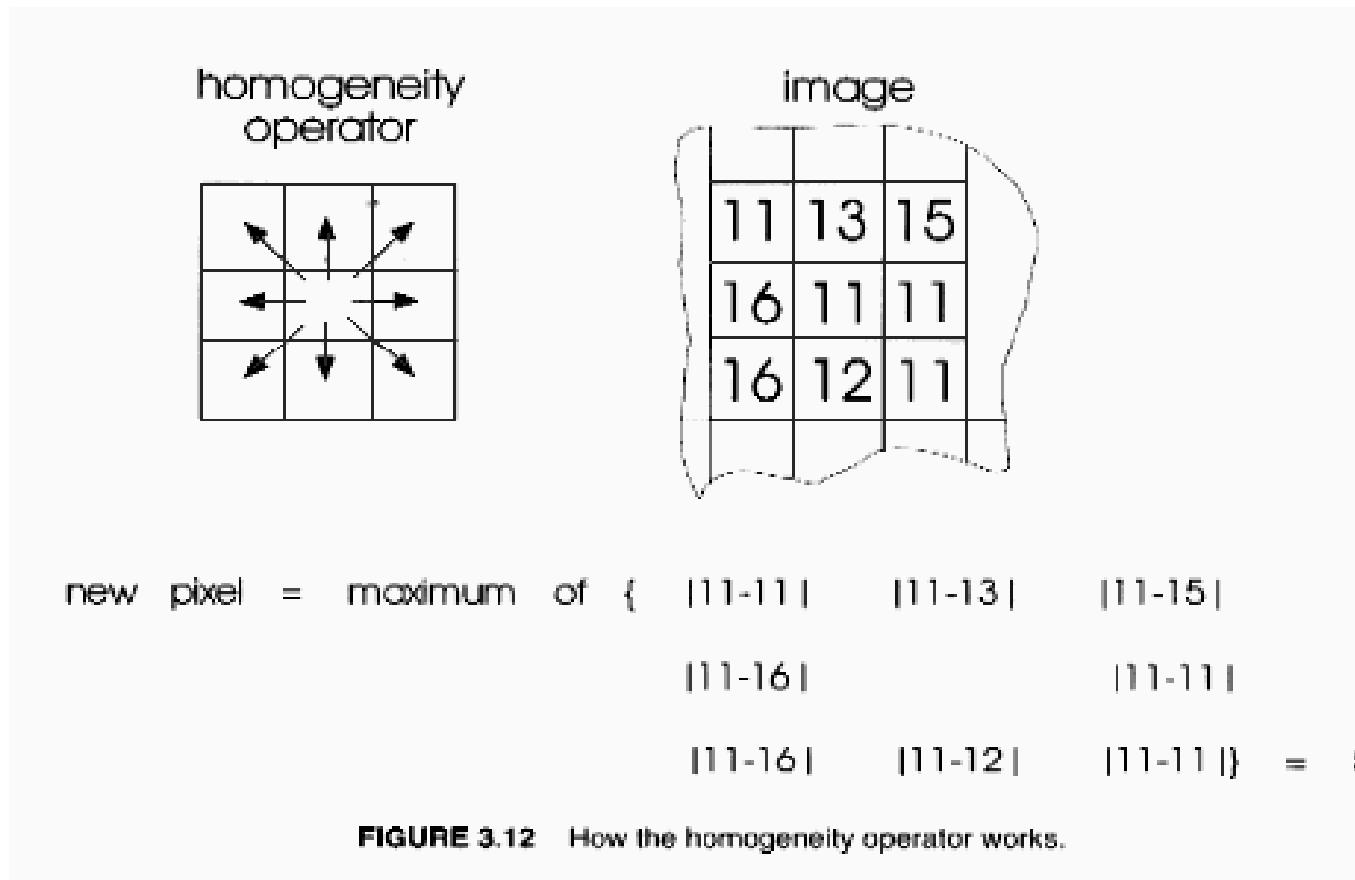
range image registration



medical image registration

## Simple Edge Detectors

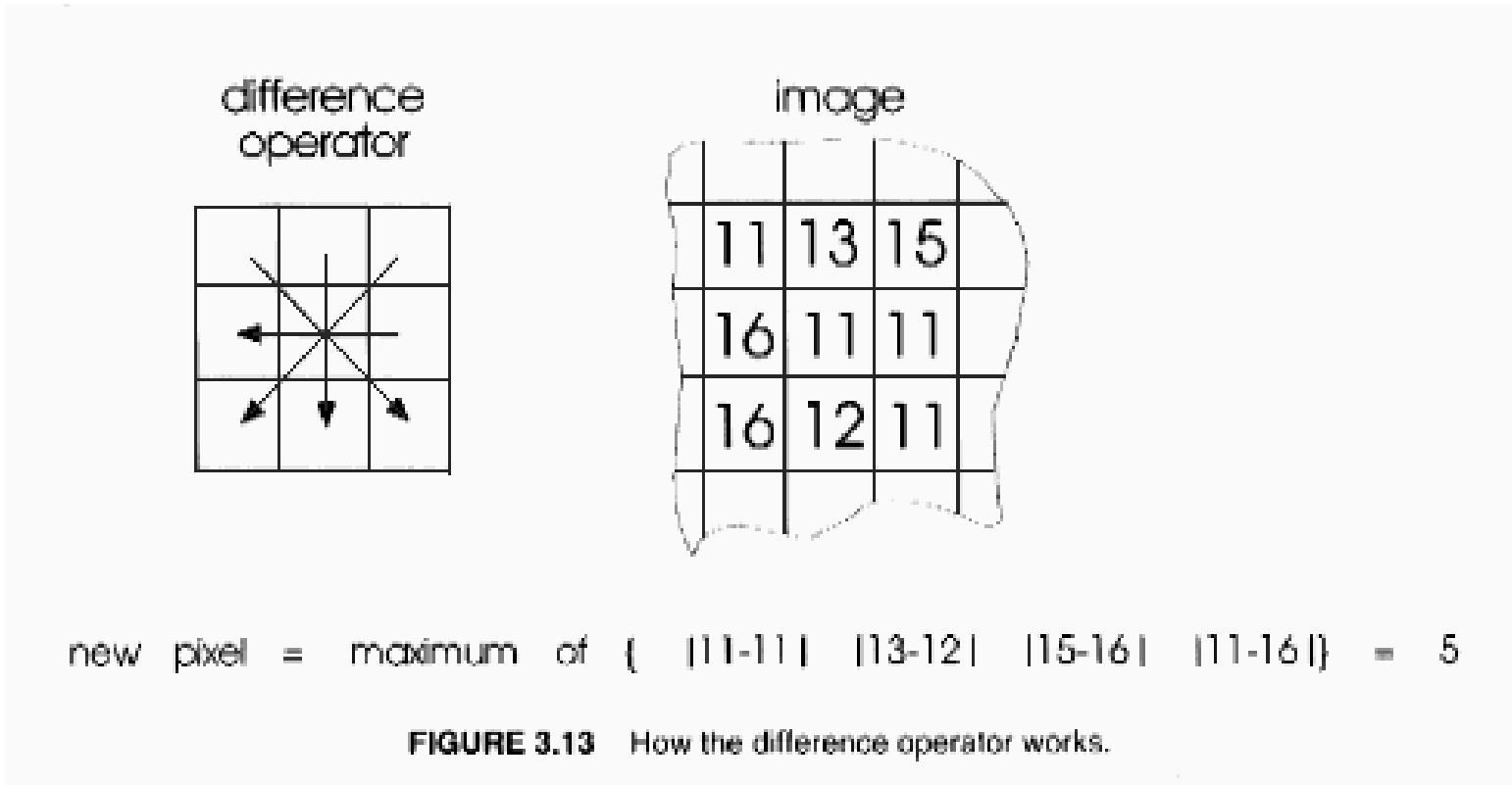
- Homogeneity Operator
  - Subtract each 8 surrounding pixels from the center pixel of a 3x3 window and choose the maximum of the absolute value.



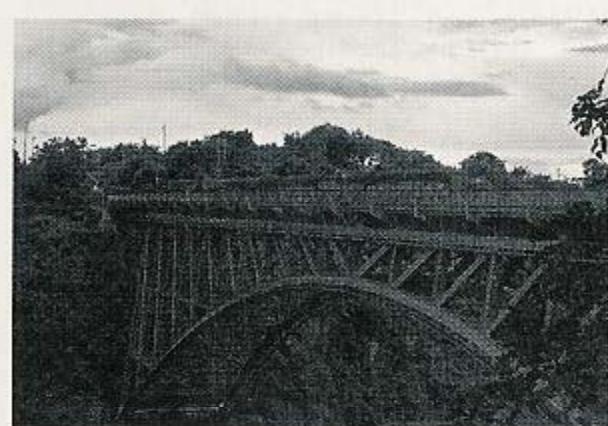
**FIGURE 3.12** How the homogeneity operator works.

## Simple Edge Detectors

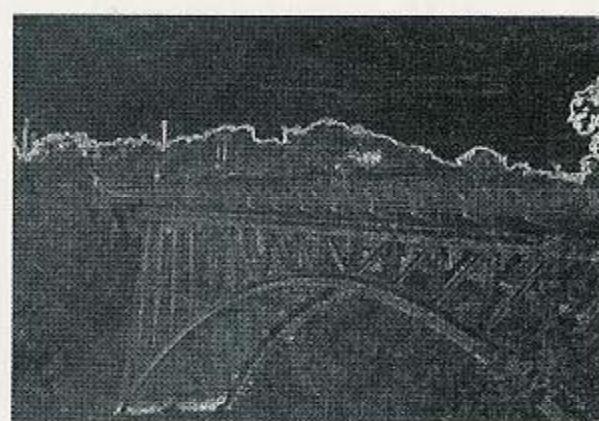
- Difference Edge Detector
  - Subtracts four pixels:



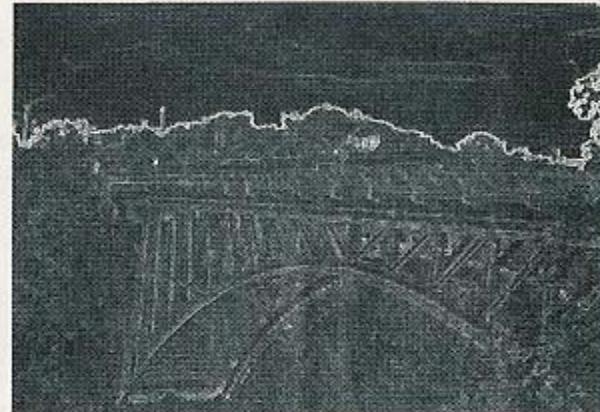
## Simple Edge Detectors



(a)



(b)



(c)

(a) Original image, (b) edge map of homogeneity operator,

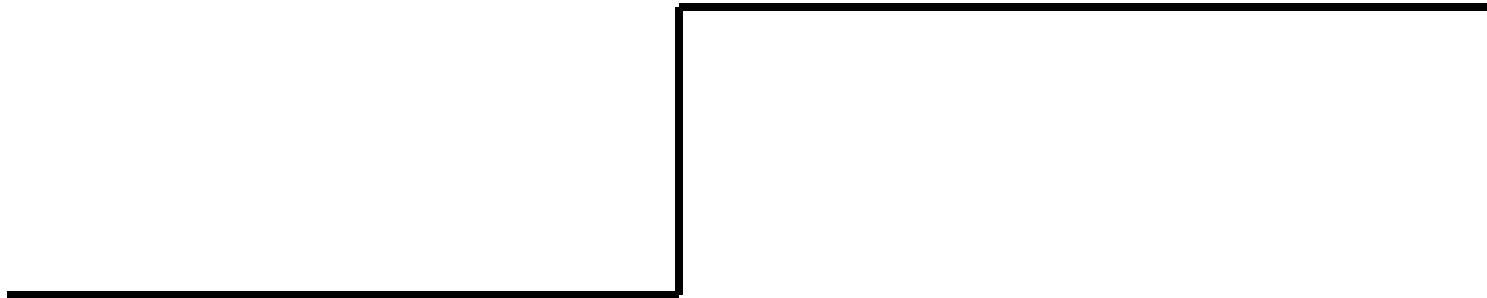
(c) Edge map of difference operator.

## Simple Edge Detectors

- Thresholding an edge map  
(Emphasize the strong edges and deemphasize the weak edges)
  - single threshold
    - $g(x) = 255$  if  $x \geq \text{threshold}$
    - 0 otherwise
  - double threshold
    - $g(x) = 255$  if  $x \geq \text{high\_threshold}$
    - 0 if  $x \leq \text{low\_threshold}$
    - $x$  otherwise

## Edge Detection

- Edge is Where Change Occurs
  - Change is measured by derivative in 1D
  - Biggest change, derivative has maximum magnitude
  - Or 2<sup>nd</sup> derivative is zero.



# Image Gradient

- The gradient of an image:  
edge is where change occurs

$$\nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right]$$
$$\nabla f = \left[ 0, \frac{\partial f}{\partial y} \right]$$
$$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$
$$\theta = \tan^{-1} \left( \frac{\partial f / \partial y}{\partial f / \partial x} \right)$$

The gradient points in the direction of most rapid change in intensity

- The edge strength is given by the gradient magnitude

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

# The Discrete Gradient

- How can we differentiate a digital image  $f(x,y)$ ?
  - Continuous gradient:

$$\frac{\partial f(x, y)}{\partial x} = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x, y) - f(x, y)}{\Delta x}$$

$$\frac{\partial f(x, y)}{\partial y} = \lim_{\Delta y \rightarrow 0} \frac{f(x, y + \Delta y) - f(x, y)}{\Delta y}$$

- Discrete gradient:

$$\frac{\partial f(x, y)}{\partial x} \approx f[x+1, y] - f[x, y]$$

$$\frac{\partial f(x, y)}{\partial y} \approx f[x, y+1] - f[x, y]$$

## Common Gradient Operators (Masks)

- Roberts

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

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

Sensitive to noise

- Prewitt

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

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

Sensitive to vertical/horizontal edges

- Sobel

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

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

Sensitive to diagonal edges

- Frei-Chen

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

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

# **Sobel**

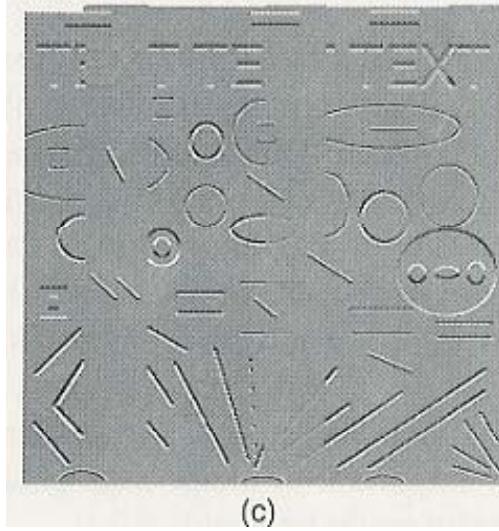
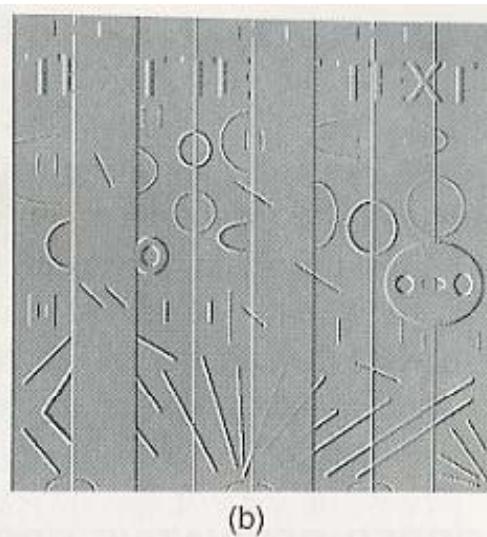
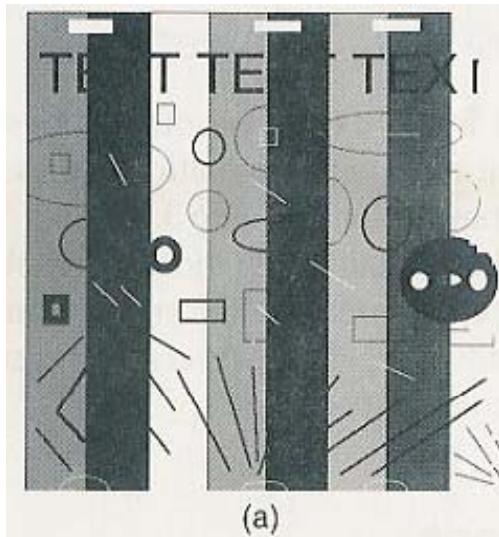
---

- Sobel Operation

```
for (i=1; i<nrow-1; i++) {  
    for (j=1; j<ncol-1;j++) {  
        hr = f[i-1][j-1] + f[i][j-1]*2 + f[i+1][j-1] - f[i-1][j+1] - f[i][j+1]*2 - f[i+1][j+1];  
        hc = -f[i-1][j-1] - f[i-1][j]*2 - f[i-1][j+1] + f[i+1][j-1] + f[i+1][j]*2 + f[i+1][j+1];  
        g[i][j] = sqrt(hr*hr + hc*hc);  
    }  
}
```

# Prewitt

---



(a) Original image

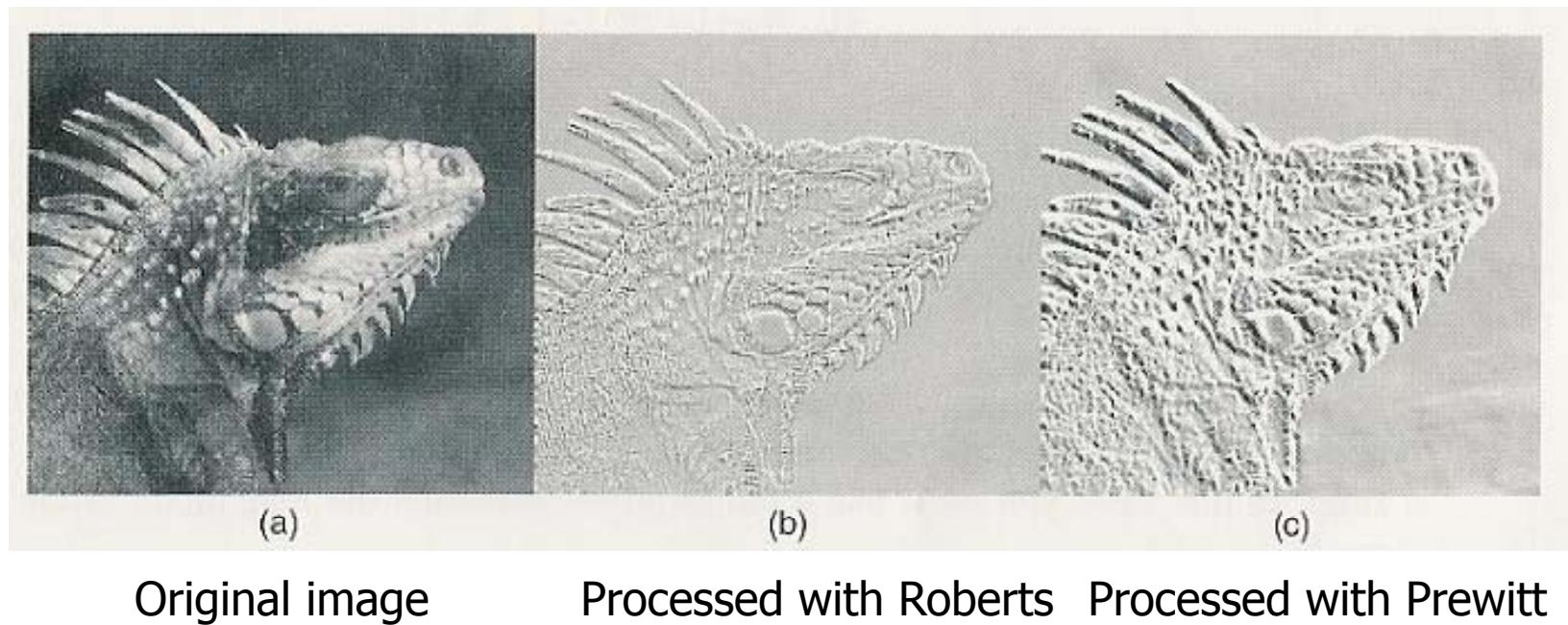
(b)  $\frac{\partial f}{\partial x}$

(c)  $\frac{\partial f}{\partial y}$

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

## *Roberts vs. Prewitt*

---



## Compass Gradient Operator

- Compass Gradient Operator
  - Find edges in eight different orientations.
  - Convolve an image with eight different masks.
  - Prewitt, Kirsch, Robinson 3-level, Robinson 5-level
- The Effects of Convolution
  - Smaller masks are sensitive to noise
  - Larger masks cannot resolve fine detail, and also requires more computations.

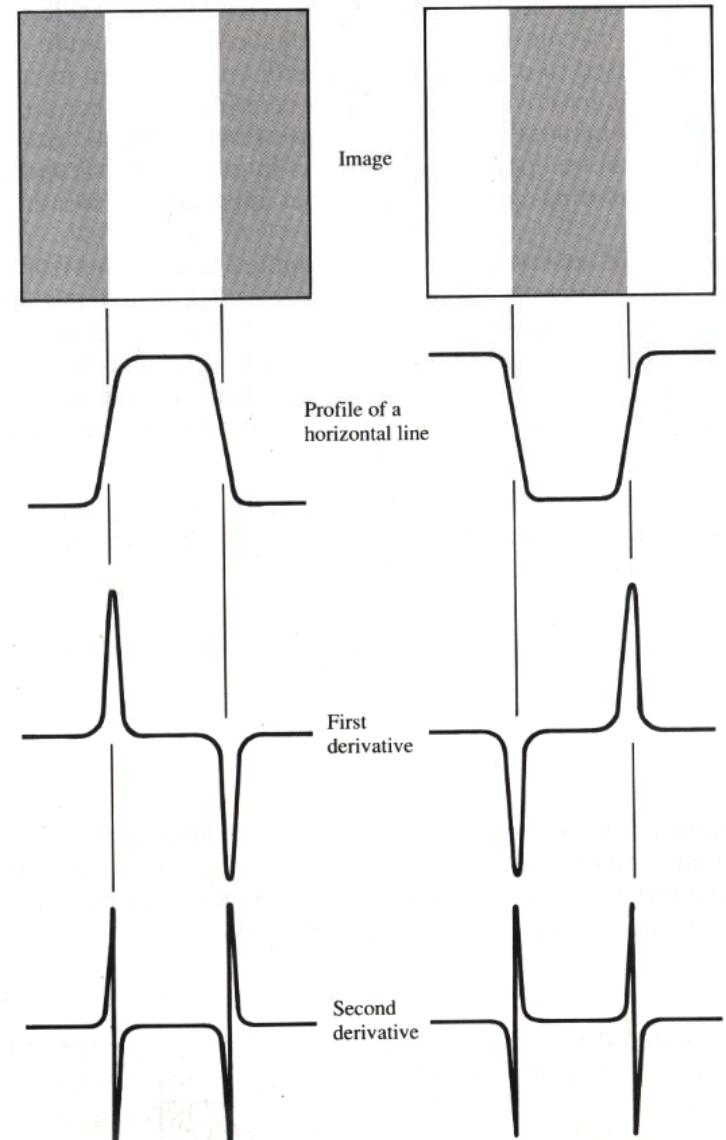
# Compass Gradient Operator

Prewitt	Kirsch	Robinson 3-level	Robinson 5-level
$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix}$	$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
$\begin{bmatrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix}$	$\begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$
$\begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & -1 \end{bmatrix}$
$\begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix}$	$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$
$\begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 1 \\ -3 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & -1 & -2 \end{bmatrix}$

## Second Order Derivative Operators

- Second Derivative Operators
  - Better edge localization
  - Laplacian operator
  - The resulting image exhibits a sign change at image edges.  
(zero crossing)

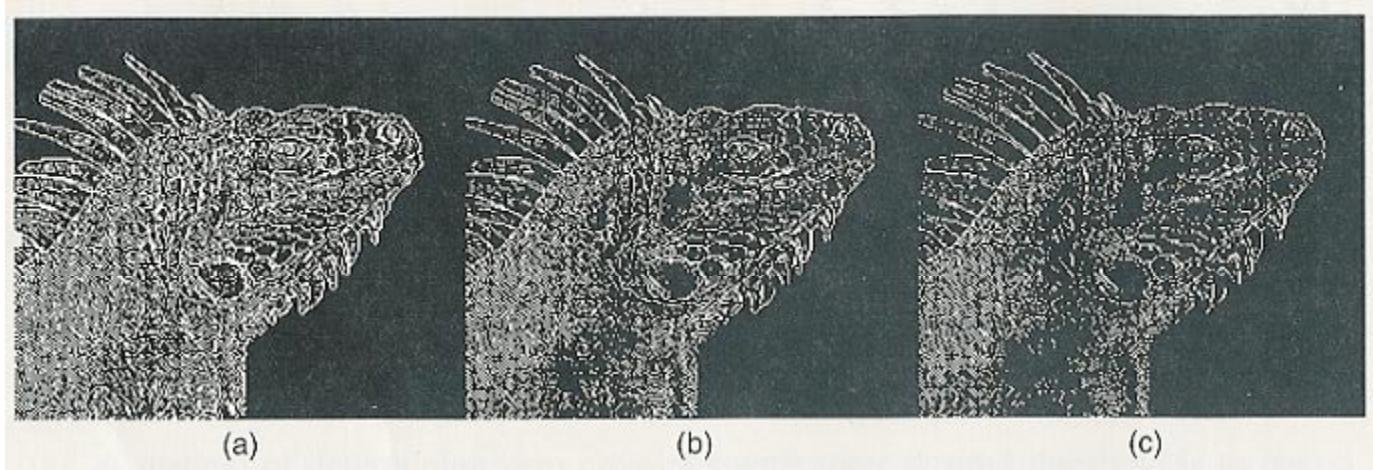
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \text{ or } \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



- Disadvantage of Laplacian
  - Sensitive to noise
  - Produce double edges
  - Unable to detect edge direction

## Zero Crossing

- Simple zero crossing may indicate more edges than desired.
- Use a threshold to determine the magnitude of the edge to mark.
  - Pass a 3x3 window determining the maximum and minimum values within that window.

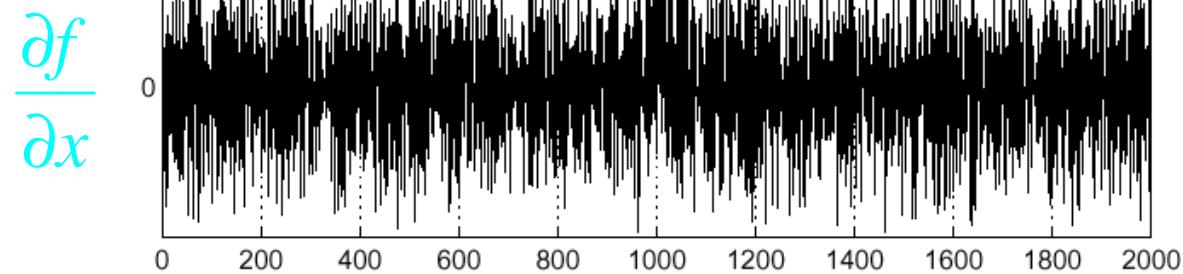
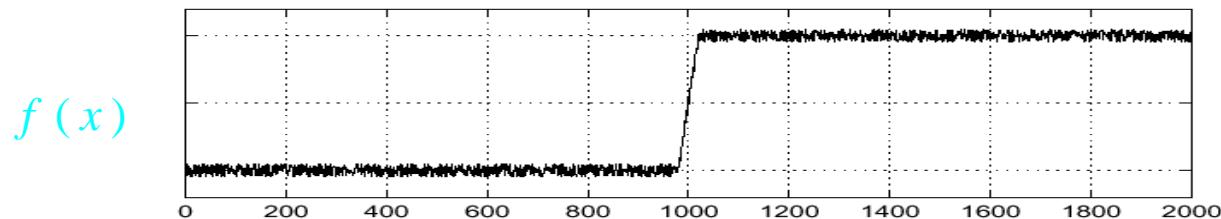


Different thresholds: (a) 20; (b) 48; (c) 72.

## ***Effects of Noise***

---

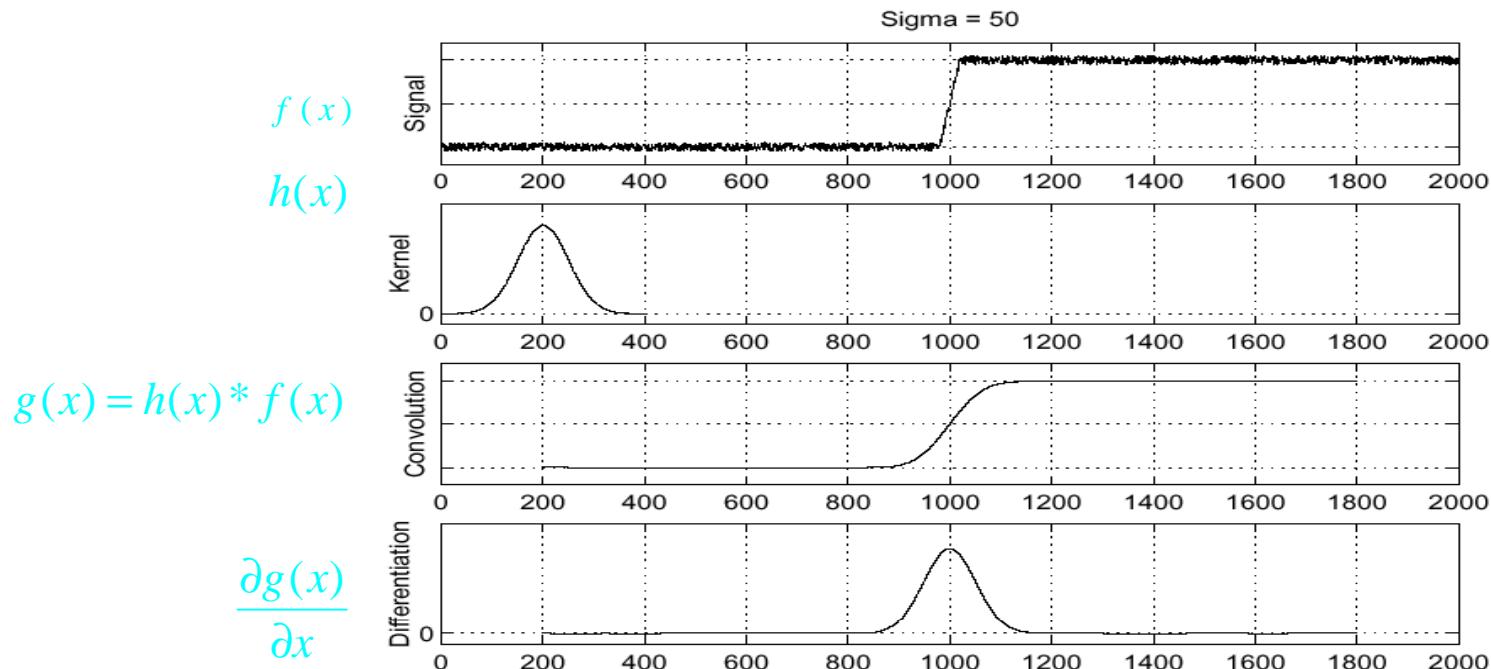
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



Where is the edge?

# Effects of Noise

- Solution: smooth first

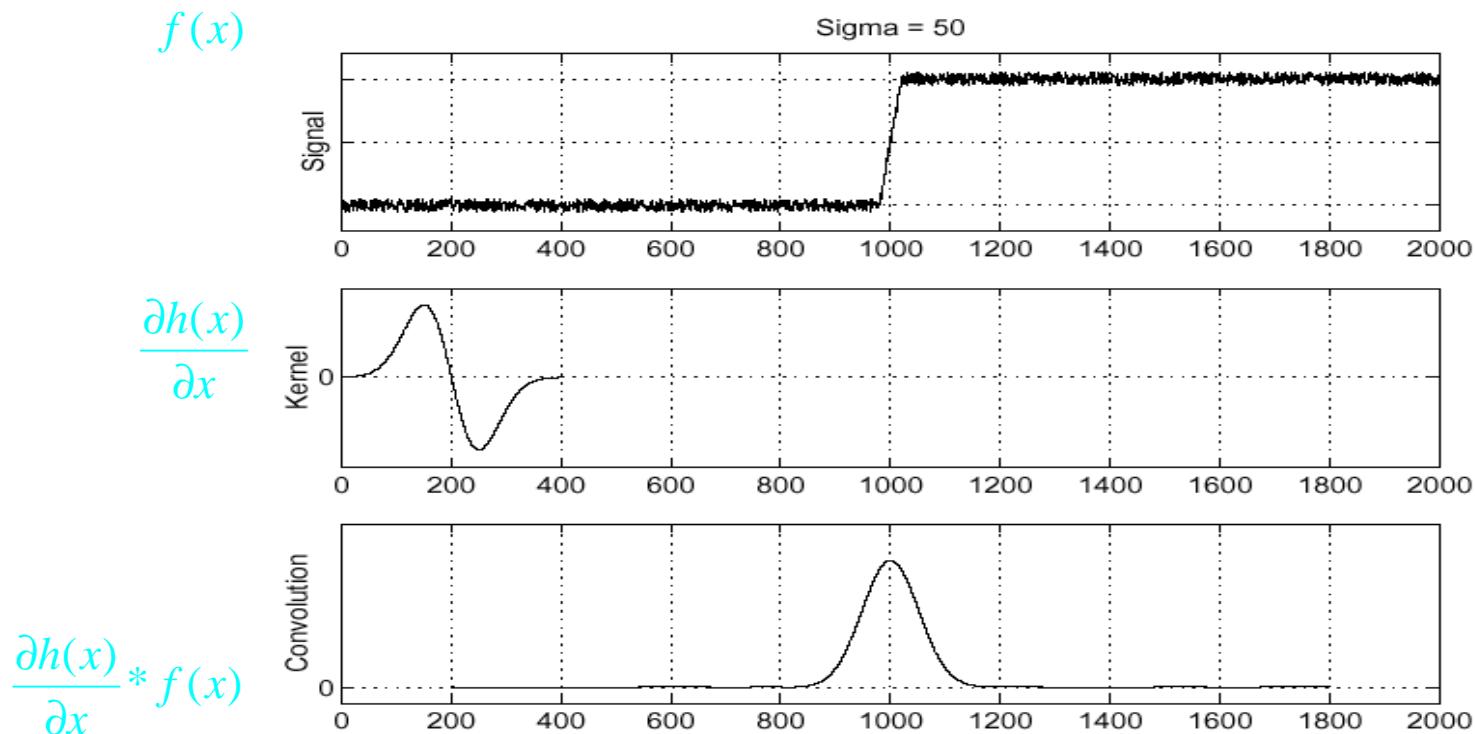


Where is the edge? Look for peaks in

# **Effects of Noise**

---

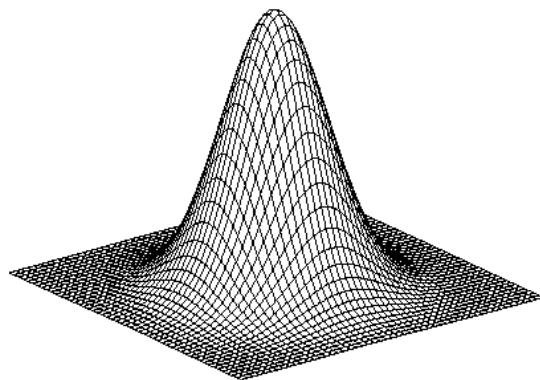
- Derivative of Convolution:  $\frac{\partial h(x) * f(x)}{\partial x} = \frac{\partial h(x)}{\partial x} * f(x)$



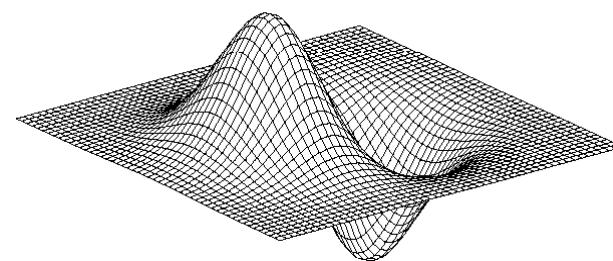
## Laplacian of Gaussian (LoG)

- Gaussian smoothing before applying Laplacian

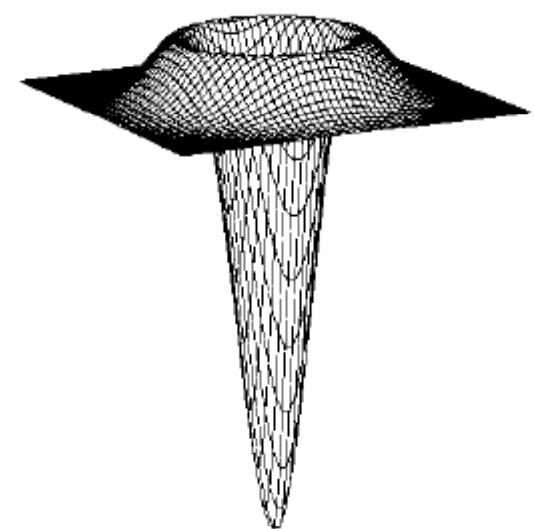
$$LoG(x, y) = \frac{1}{\pi\sigma^4} \left[ 1 - \frac{(x^2 + y^2)}{2\sigma^2} \right] e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



Gaussian



Derivative of Gaussian



Laplacian of Gaussian

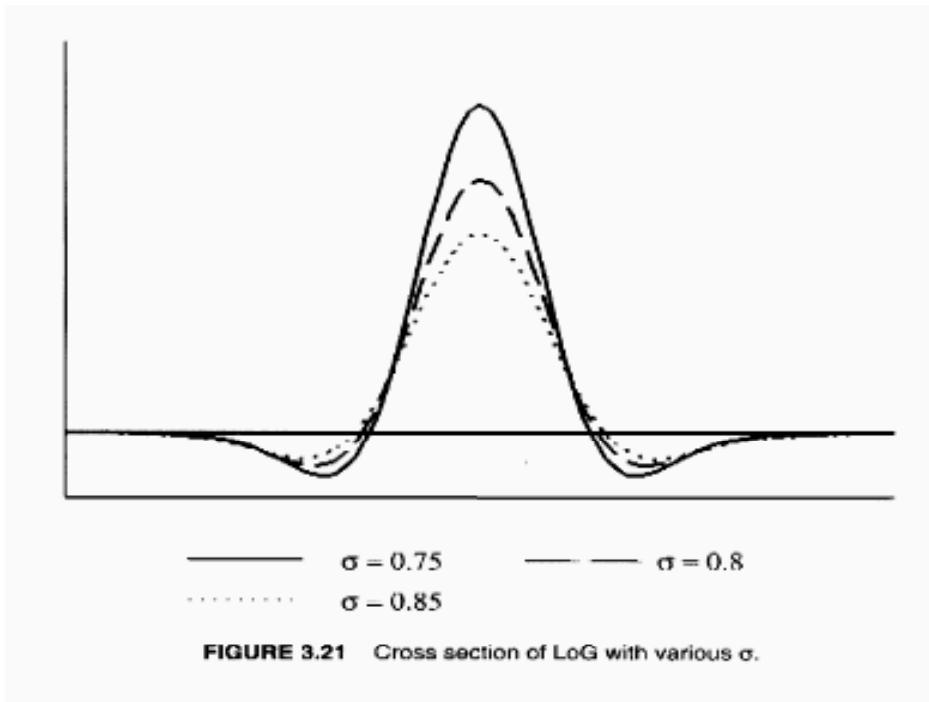
$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{\partial h(x, y)}{\partial x}$$

$$\nabla^2 h(x, y)$$

## Laplacian of Gaussian (LoG)

- The wider the LoG function, the wider the edge that will be detected.
- The greater the value of  $\sigma$ , the wider the convolution mask necessary.



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

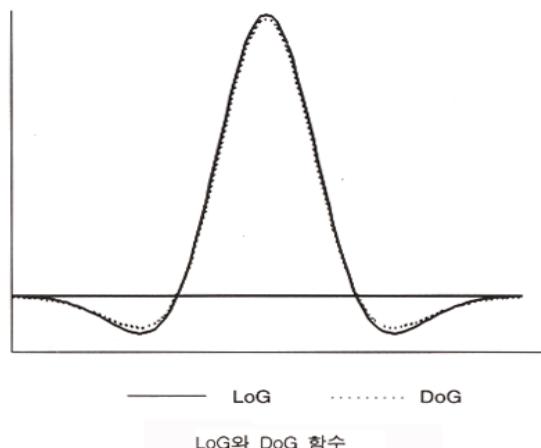
LoG Mask ( $\sigma=1.4$ )

## Difference of Gaussians (DoG)

- The LoG operator requires large computation.
- Approximate LoG using two Gaussians:

$$DoG(x, y) = \frac{1}{2\pi\sigma_1^2} e^{\frac{-(x^2+y^2)}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{\frac{-(x^2+y^2)}{2\sigma_2^2}}, \text{ where } \sigma_1 > \sigma_2$$

- Can specify the width of edges to detect by varying the values of  $\sigma_1$  and  $\sigma_2$ .
- Most similar to the LoG when  $\sigma_1=1.6\sigma_2$ .



7×7 마스크

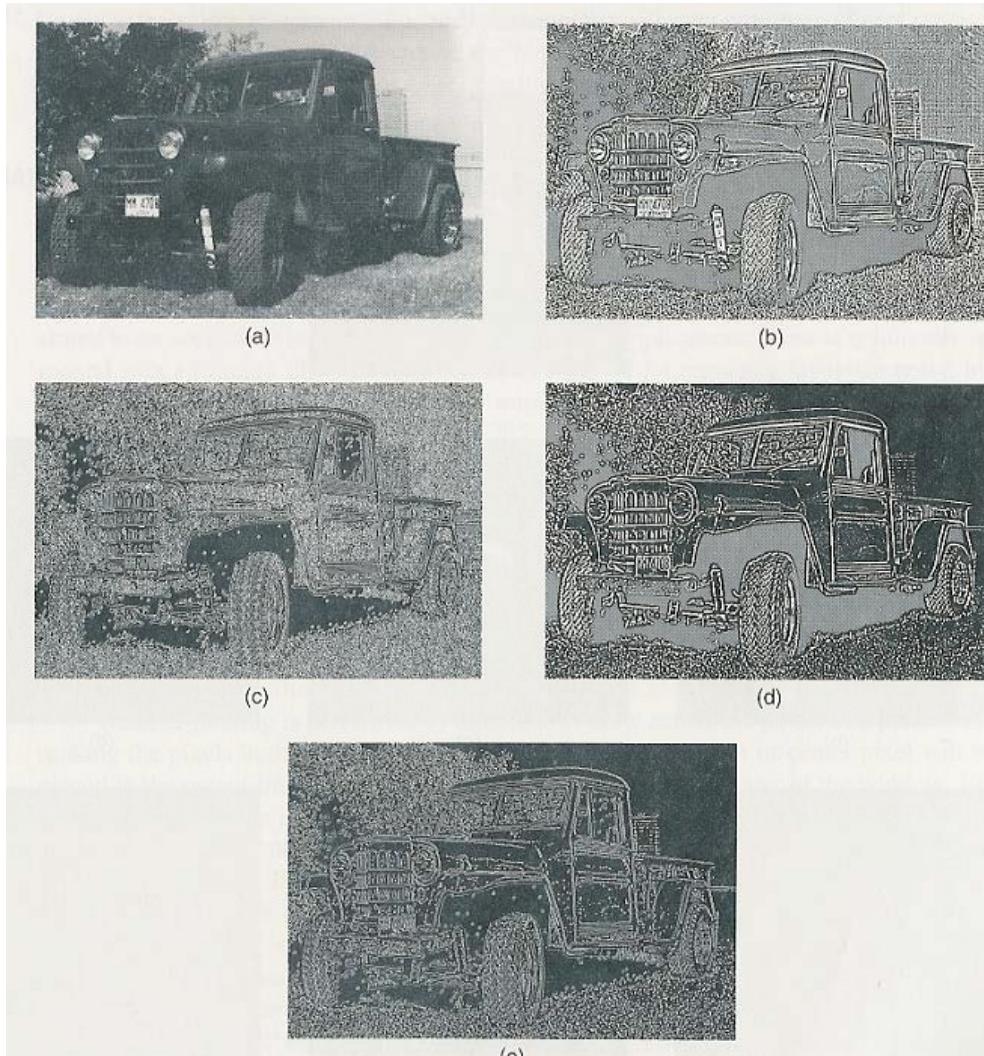
0	0	-1	-1	-1	0	0
0	-2	-3	-3	-3	-2	0
-1	-3	5	5	5	-3	-1
-1	-3	5	16	5	-3	-1
-1	-3	5	5	5	-3	-1
0	-2	-3	-3	-3	-2	0
0	0	-1	-1	-1	0	0

9×9 마스크

0	0	0	-1	-1	-1	0	0	0
0	-2	-3	-3	-3	-3	-3	-2	0
0	-3	-2	-1	-1	-1	-2	-3	0
-1	-3	-1	9	9	9	-1	-3	-1
-1	-3	-1	9	19	9	-1	-3	-1
-1	-3	-1	9	9	9	-1	-3	-1
0	-3	-2	-1	-1	-1	-2	-3	0
0	-2	-3	-3	-3	-3	-3	-2	0
0	0	0	-1	-1	-1	0	0	0

## *Difference of Gaussians (DoG)*

---



(a) Original (b) 7x7 DoG (c) zero-crossings detected  
(d) 9x9 DoG (e) zero-crossings detected.

- The image processed with the 9x9 mask shows wider edges.
- Notice that the edges in the edge maps are shown as closed curves.  
(A characteristic of edge detectors based on the Laplacian of Gaussian smoothing function.)

## Color Edge Detection

- Type I
  - Detect the discontinuity in an image's luminance.
  - Edge detection is performed on the intensity channel of a color image in HSI space.
- Type II
  - Detect the discontinuity in RGB channels.
  - Edge detection is performed on each color component.
  - Color components can also be summed to create the gray scale edge map:

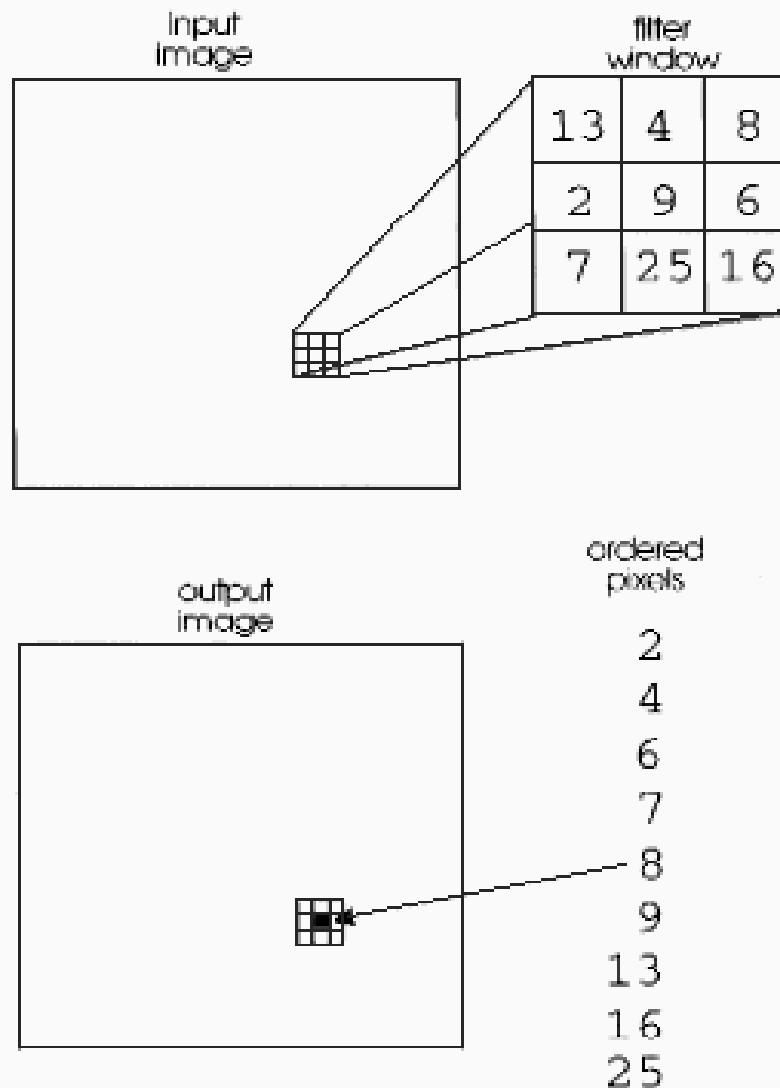
$$G(x, y) = \sqrt{G_{red}^2 + G_{green}^2 + G_{blue}^2}$$

## ***Median Filtering***

---

- **Advantage:**
  - Suited for removing impulse noise while preserving sharp edges and details.
  - Note: Lowpass filtering is suited for removing Gaussian noise, but not impulse noise.
- **Three steps for median filtering:**
  - Sort gray values of the pixels within a window
  - Find the median value
  - Replace the center pixel value with the median

# ***Median Filtering***



**FIGURE 3.26** How a median filter works.

## Median Filtering

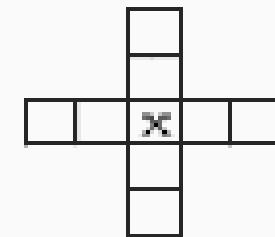
- Median filter windows:



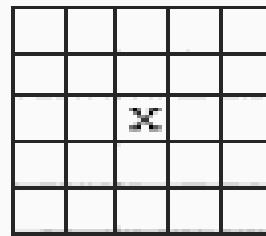
horizontal



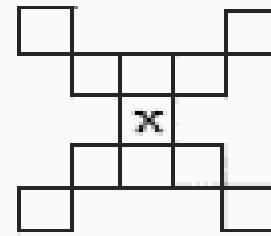
vertical



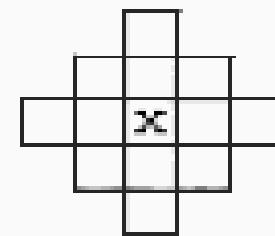
cross



block



x



diamond

**FIGURE 3.27** Median filter windows.

# *Median Filtering*

---



Impulse noise



Gaussian filtered



Median filtered



Gaussian noise



Gaussian filtered



Median filtered

## ***Median Filtering***

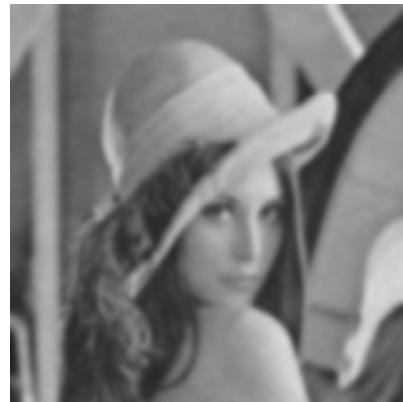
---



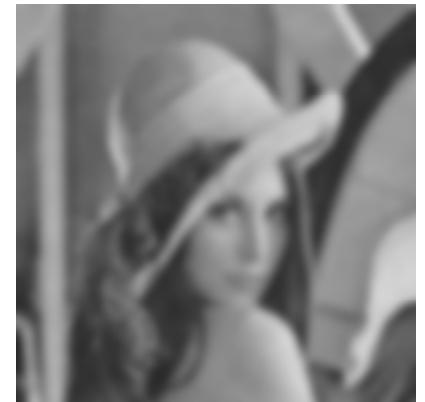
Gaussian noise



3x3 Gaussian filtered



5x5 Gaussian filtered



7x7 Gaussian filtered



3x3 Median filtered



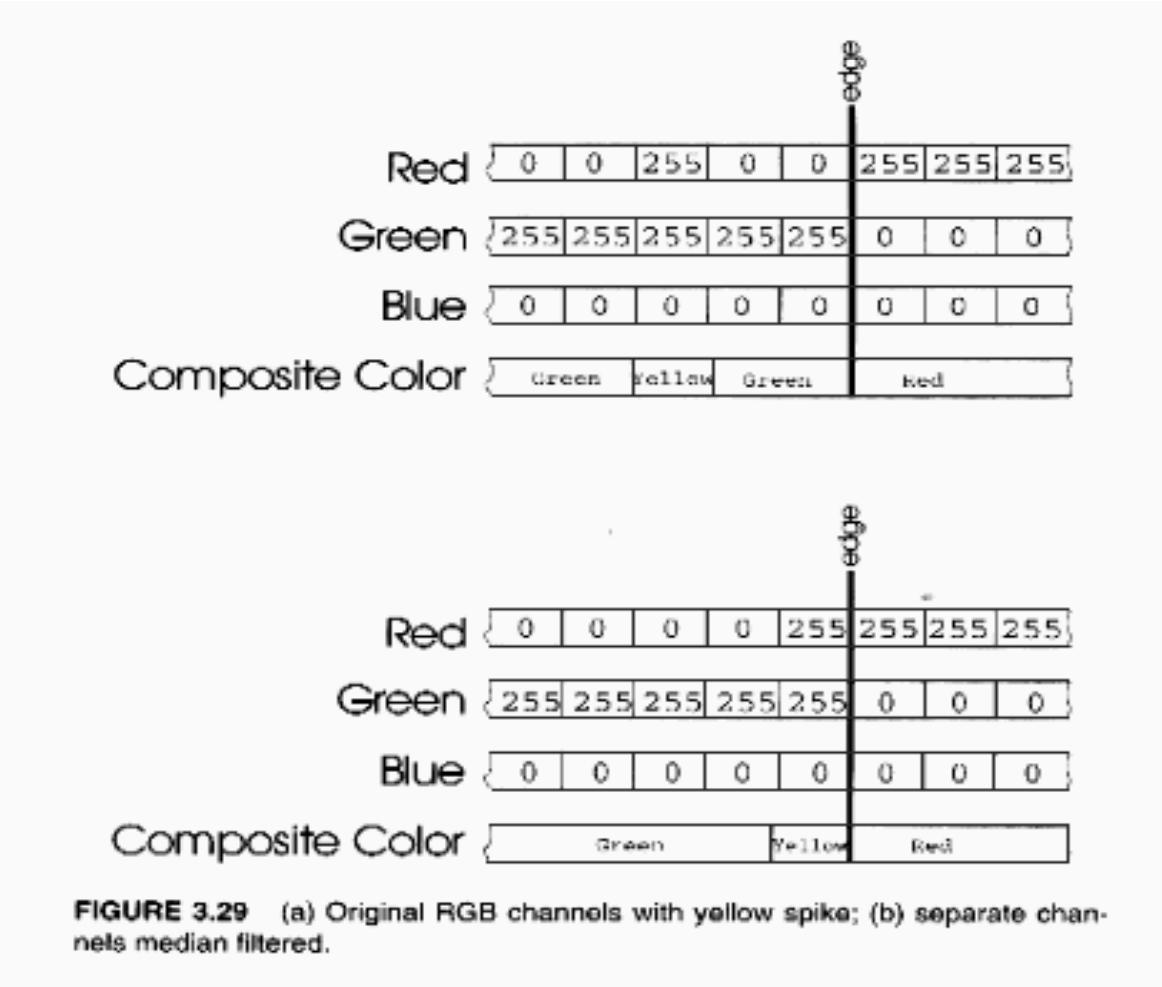
5x5 Median filtered



7x7 Median filtered

## Color Median Filtering

- The problem with median filtering each color component separately:



## Color Median Filtering

- A unique property of median filtering:

$$\sum_{i=1}^N |x_{\text{med}} - x_i| \leq \sum_{i=1}^N |y - x_i|,$$

where  $N$  is the number of elements in the set,  $y$  an arbitrary value in the set, and  $x_{\text{med}}$  the median value.

- Break up  $x$  and  $y$  in the unique property into red, green, blue components ( $x_i = \{red_i, green_i, blue_i\}$ ) :

$$Dist_i = \sum_{i=1}^N |red_i - red_j| + \sum_{i=1}^N |green_i - green_j| + \sum_{i=1}^N |blue_i - blue_j|,$$

where  $i$  is the pixel being processed and  $j$  represents the other pixel samples.

- The pixel  $x_i$  with smallest  $Dist_i$  is the filter output.

## ***Minimum/Maximum Filtering***

---

- Replace center pixel with min or max window value.
- Minimum filter removes white spikes.
- Maximum filter removes dark spikes.
- Cascade of max and min filters remove white & dark spikes (“salt and pepper” noise)
  - Closing filter: max+min filter - removes dark spike without altering original.
  - Opening filter: min+max filter – removes white spike without altering original.