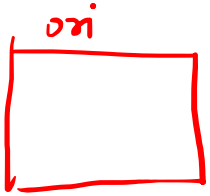
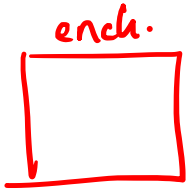
# Spatial Filtering

Last class.

Point to Point Image Transform




f



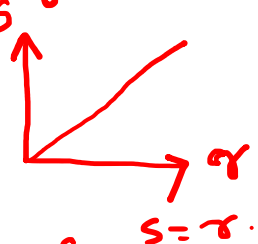

g

$g(x,y) = T(f(x,y))$

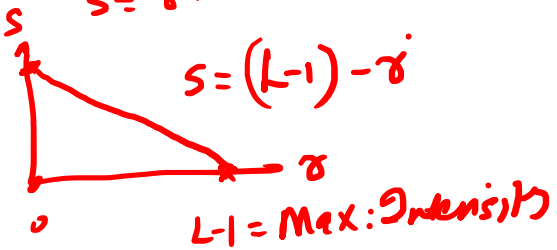
s = o/p image intensity  r = i/p image intensity

$s = T(r)$

(1) Linear
- Identity




- Negative.




(2) Log Tranf

$s = c \log(1+r)$


(3) Power law Tranf.

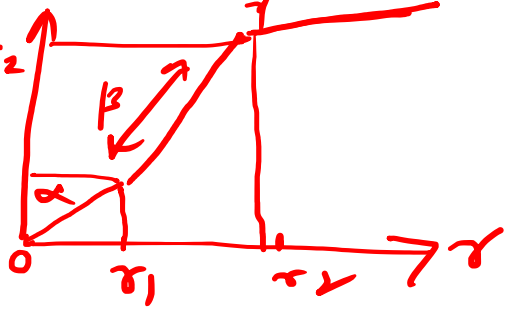
$s = c \cdot r^{\gamma}$

- Piece-wise Contrast-stretching

$s = \alpha r \quad (r < r_1)$

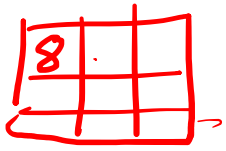
$s = s_1 + \beta(r - r_1)$

$s = s_2 + \gamma(r - r_2)$




(4) Bit plane slicing

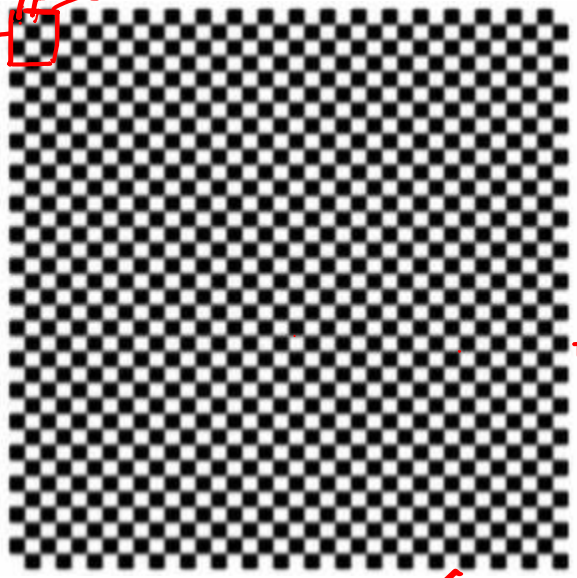
4-bit image



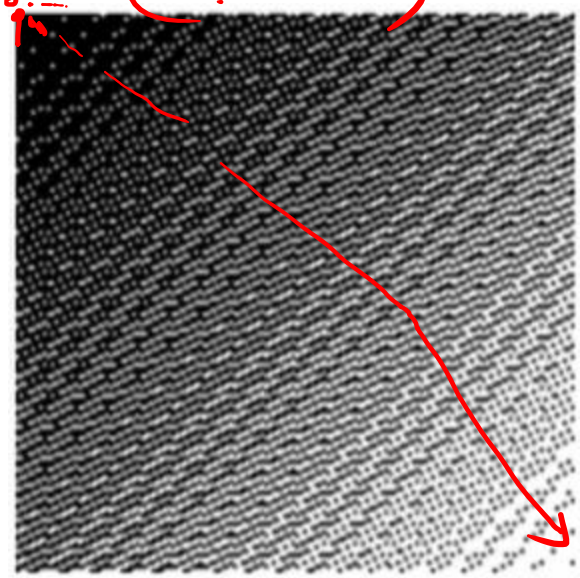

= 0100

# Spatial Frequencies

Radical change in intensity



Slowly varying change in intensity



① High frequency Image

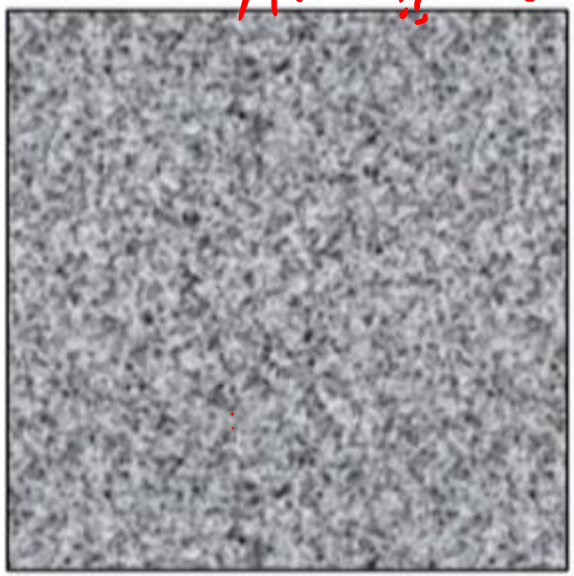
Changes in tone is abrupt over small areas

② Low frequency Image

Little variation in tone over several pixels

# Spatial Frequencies

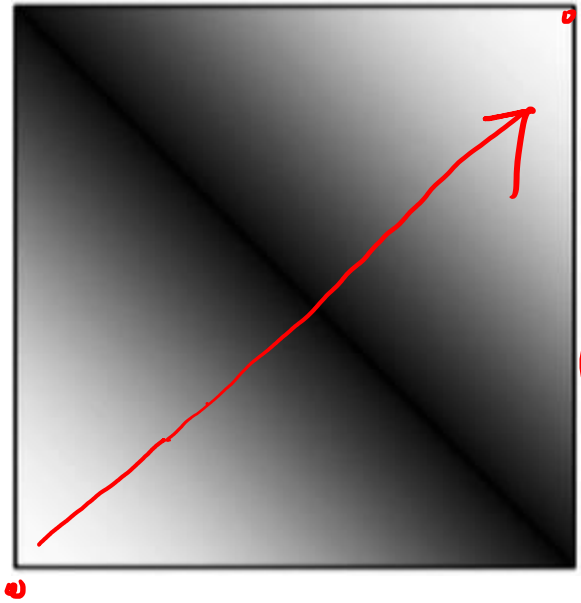
Radical change in intensity



High frequency Image

Changes in tone is abrupt over small areas

Slowly varying change in intensity



Low frequency Image

Little variation in tone over several pixels

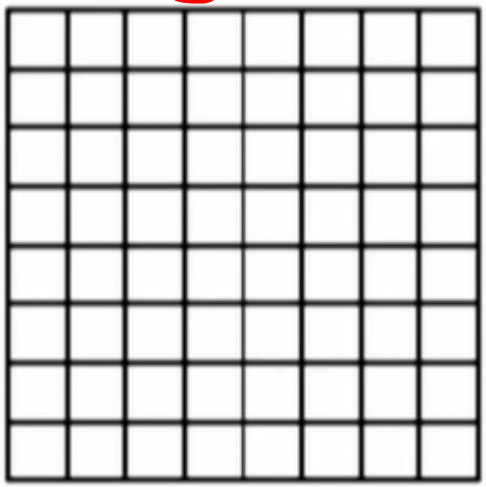
# Spatial Frequency Definitions

- Number of changes in intensity value per unit distance for any particular part of the image.

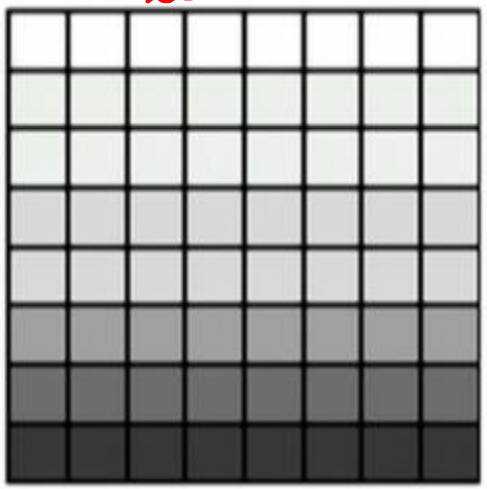
- Image is composed of a
  - Low frequency details- basic details, few changes over a given area
  - High Frequency details – fine details, dramatic changes over a given area

# Spatial Frequency Examples

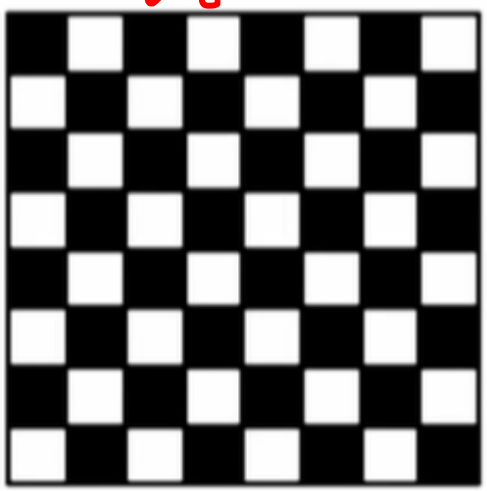
- Zero spatial frequency - flat image, all pixels same value.
- Low spatial frequency - smoothly varying grayscale image.
- High spatial frequency- an image consisting of check board fashion



Zero spatial frequency



Low spatial frequency

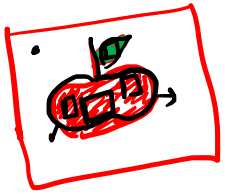


High spatial frequency-

Spatial filtering.

High
Low.

# Spatial filtering



## Smoothening
(blurring / averaging) (avg.)

→ Suppress high freq comp.
//
• enhance low freq comp.

Goal = Retain basic structure of the image

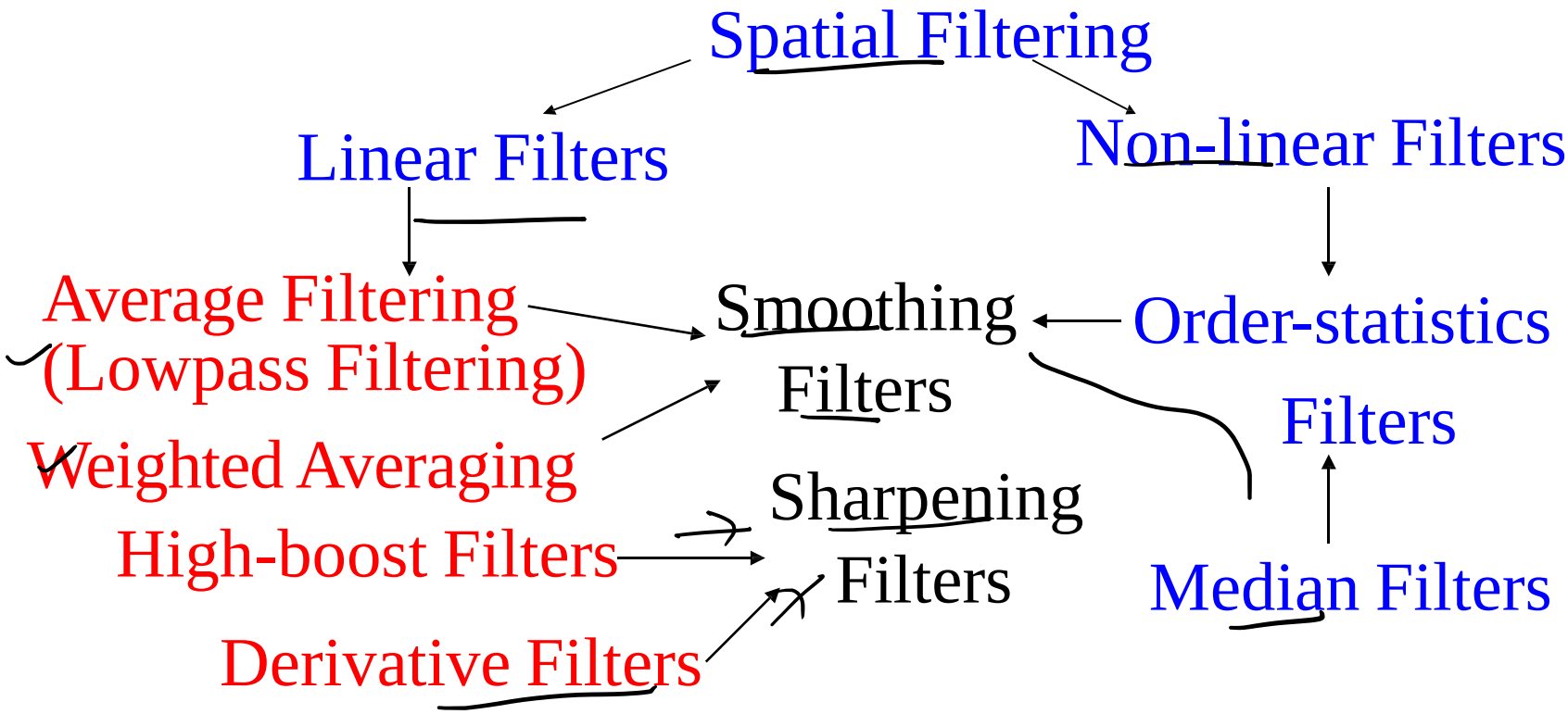
- linear
  - (1) Box filters
  - (2) weighted Avg filters
  - (3) Gaussian filters
- non-linear.
  - (4) Median filters

## Sharpening.

→ enhance H. freq. comp
• suppress the low.
• improve for highlight finer details or edges

- first order derivative
  - (1) Simple FOD
  - (2) Prewitt
  - (3) Sobels
- second order derivative
  - (4) Laplacian

# Spatial Filtering

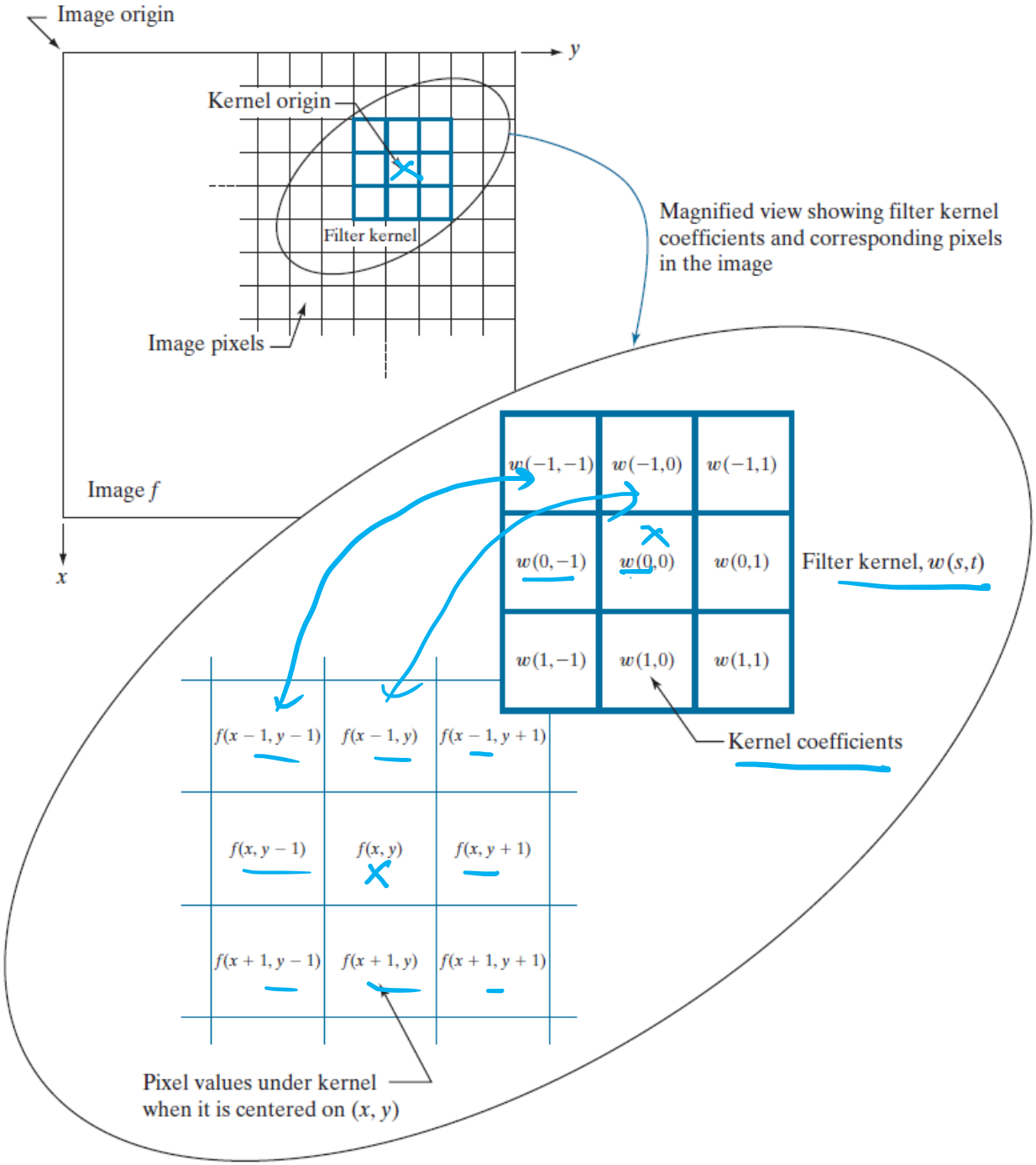



Smoothing filters: used for blurring and for noise reduction

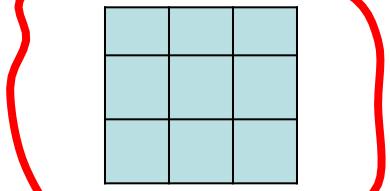
Sharpening filters: used to highlight fine detail in an image or to enhance detail that has been blurred, either in error or as a natural effect of a particular method of image acquisition

(2) local processing. "convolution"

# Linear Spatial Filtering



# Linear Spatial Filtering



Mask

| | | |
|---|---|---|
| $W(-1,-1)$ | $W(-1,0)$ | $W(-1,1)$ |
| $W(0,-1)$ | $W(0,0)$ | $W(0,1)$ |
| $W(1,-1)$ | $W(1,0)$ | $W(1,1)$ |

Mask coefficients, showing coordinate arrangement

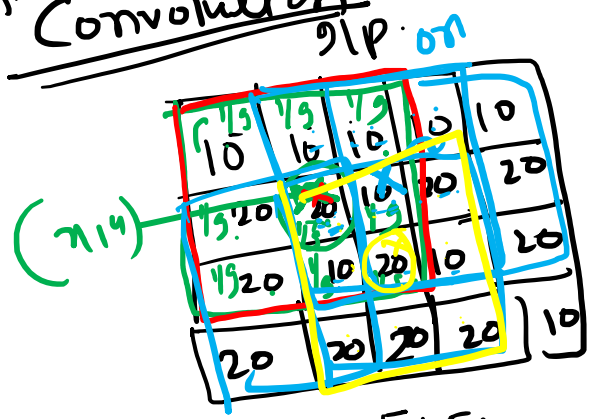
Image $f(x, y)$

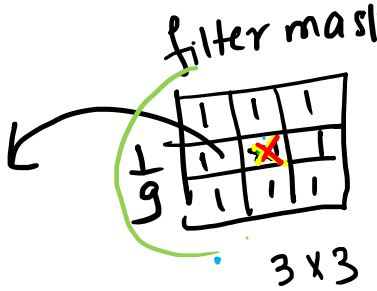
| | | |
|---|---|---|
| $f(x-1,y-1)$ | $f(x-1,y)$ | $f(x-1,y+1)$ |
| $f(x,y-1)$ | $f(x,y)$ | $f(x,y+1)$ |
| $f(x+1,y-1)$ | $f(x+1,y)$ | $f(x+1,y+1)$ |

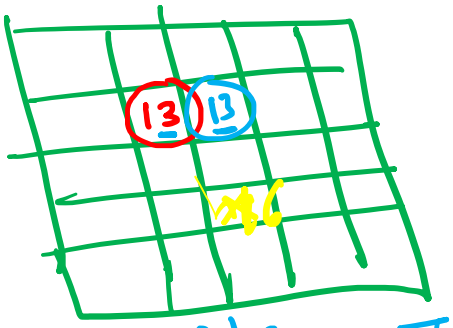
Pixels of image section under mask

Since the operations performed in the convolution are linear, these types of filters are called linear filters

# filtering. Convolution



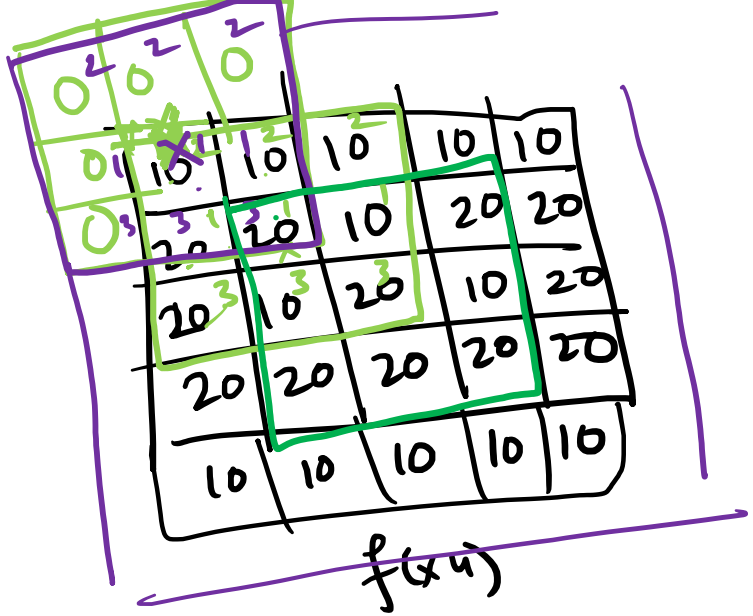

sop

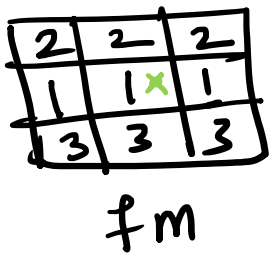
$$g(x,y) = \left(\begin{array}{l} 10\times\frac{1}{9} + 10\times\frac{1}{9} + 10\times\frac{1}{9} \\ + \ 20\times\frac{1}{9} + 20\times\frac{1}{9} + 10\times\frac{1}{9} \\ + \ 20\times\frac{1}{9} + 10\times\frac{1}{9} + 20\times\frac{1}{9} \end{array}\right) = \frac{50}{5} + \frac{80}{9} = \frac{130}{9} \downarrow 13$$

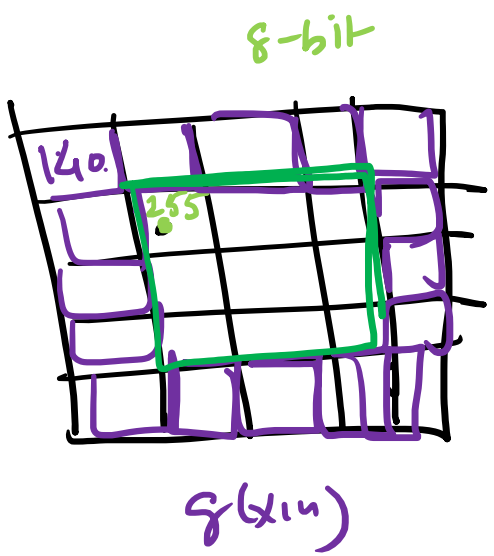
$$g(x',y') = \frac{1}{9}\left[30+50+40\right] = \frac{120}{9} = 13.3 \Rightarrow 13$$

$x' = x+1$, $y' = y+1$

Replacing a pixel value → avg value of its 3x3 neighbor



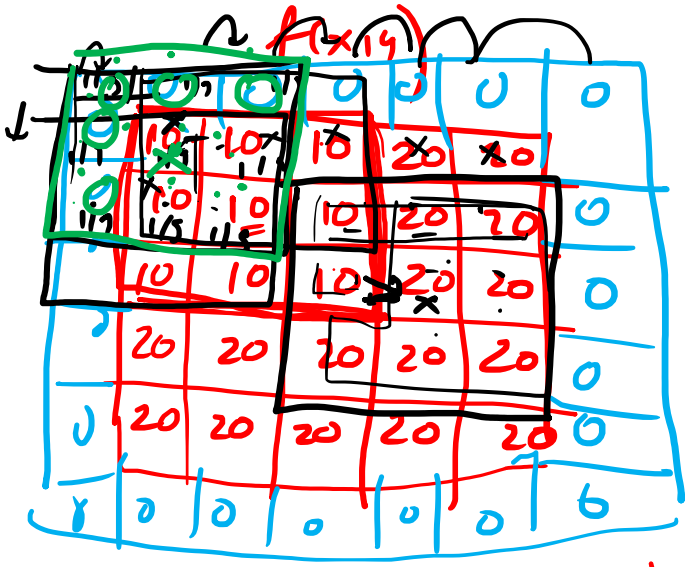




$$g(x,y) = [20 + 10\times2 + 10\times2 + 20\times1 + 20\times1 + 10\times1 + 10\times3 + 10\times3 + 20\times3]$$

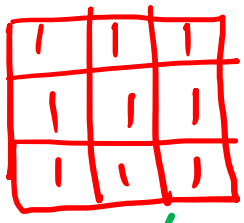
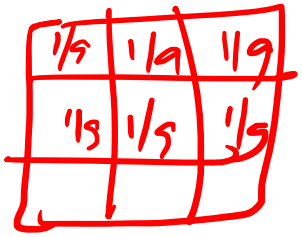
$$= [60 + 60 + 150]$$

$$= (270) \rightarrow 255$$

$$[0 - 255]$$

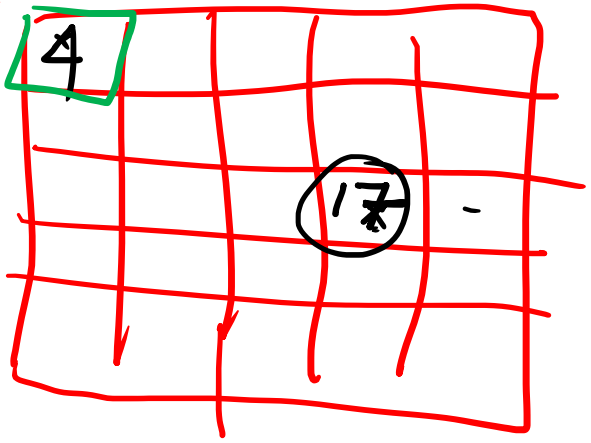



grayscale → [0, 255]

$\frac{1}{9}$  = 

replace a pixel in f(x,y) with the avg. value of pixels in its 3x3 neighborhood

→ g(x,y) =




(1) Convol^n using zero padding

$= 0\times\frac{1}{9} + 0\times\frac{1}{9} + 0\times\frac{1}{9} + 0\times\frac{1}{9} + 10\times\frac{1}{9} + 10\times\frac{1}{9} + 0 + \frac{10}{9} + \frac{10}{9}$

$= 40/9 = \lfloor 4.4 \rfloor = 4$

$= 160/9 = 17.3$

(2) Conv^n without zero padding

local processing?

Convolution with zero padding

$f(x,y)$

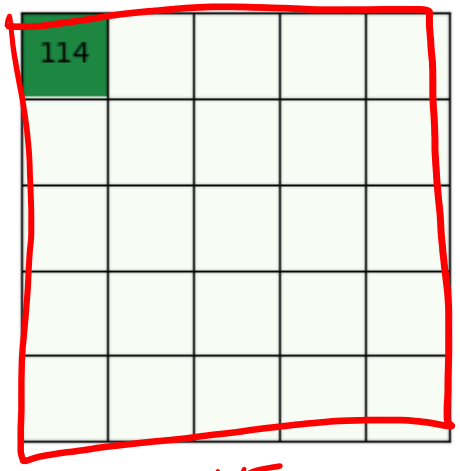
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 60 | 113 | 56 | 139 | 85 | 0 |
| 0 | 73 | 121 | 54 | 84 | 128 | 0 |
| 0 | 131 | 99 | 70 | 129 | 127 | 0 |
| 0 | 80 | 57 | 115 | 69 | 134 | 0 |
| 0 | 104 | 126 | 123 | 95 | 130 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5x5

Kernel

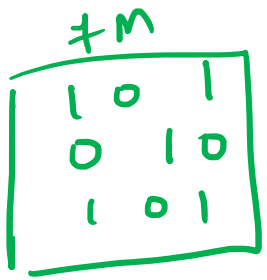
| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

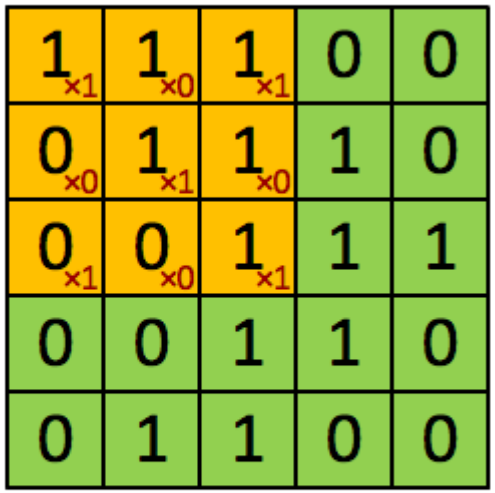
$g(x,y)$



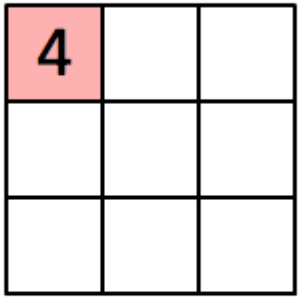

5x5.

Convolution without zero padding




Image




Convolved Feature

# Linear Spatial Filtering

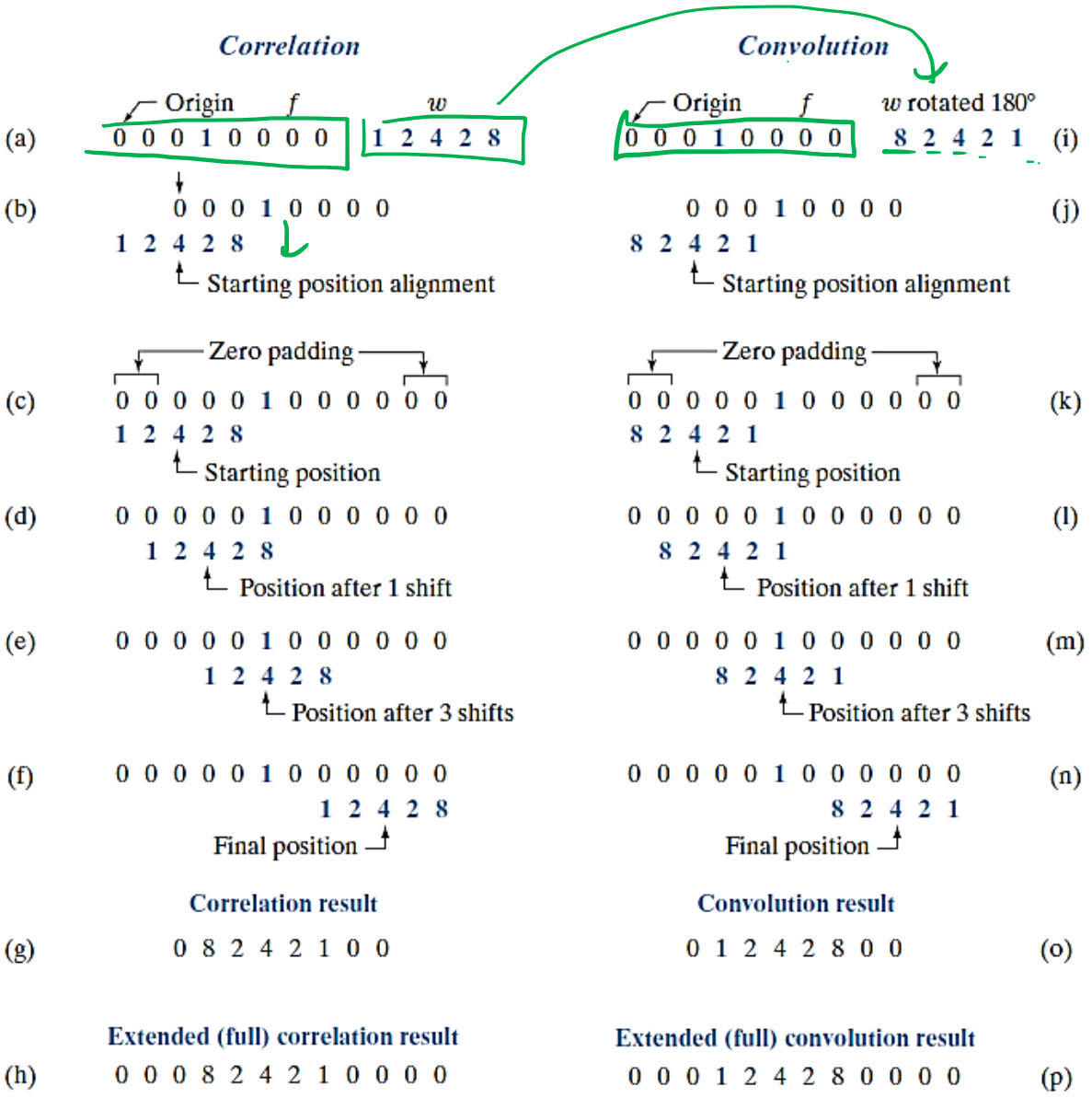
- Border Effects- Special Consideration
  - For a small value of *m* and *n*, it is not that much problem; otherwise we may loose substantial amount of data
  - Common procedures:
    - Border strips are added and set to zero
    - Rows and columns are considered to wrap around
    - Limit the excursion of the center of the filter mask to a distance no less than (*n*-1)/2 pixels from the border of the original image
      - The resulting filtered image will be smaller than the original, but all the pixels in the filtered image will have been processed with the full mask

# Spatial Correlation and Convolution

- Correlation is the process of moving a filter mask over the image and computing the sum of products at each location, as explained

- The mechanics of convolution are the same, except that the filter is first rotated by $180^0$

# Spatial Correlation and Convolution



# Spatial Correlation and Convolution

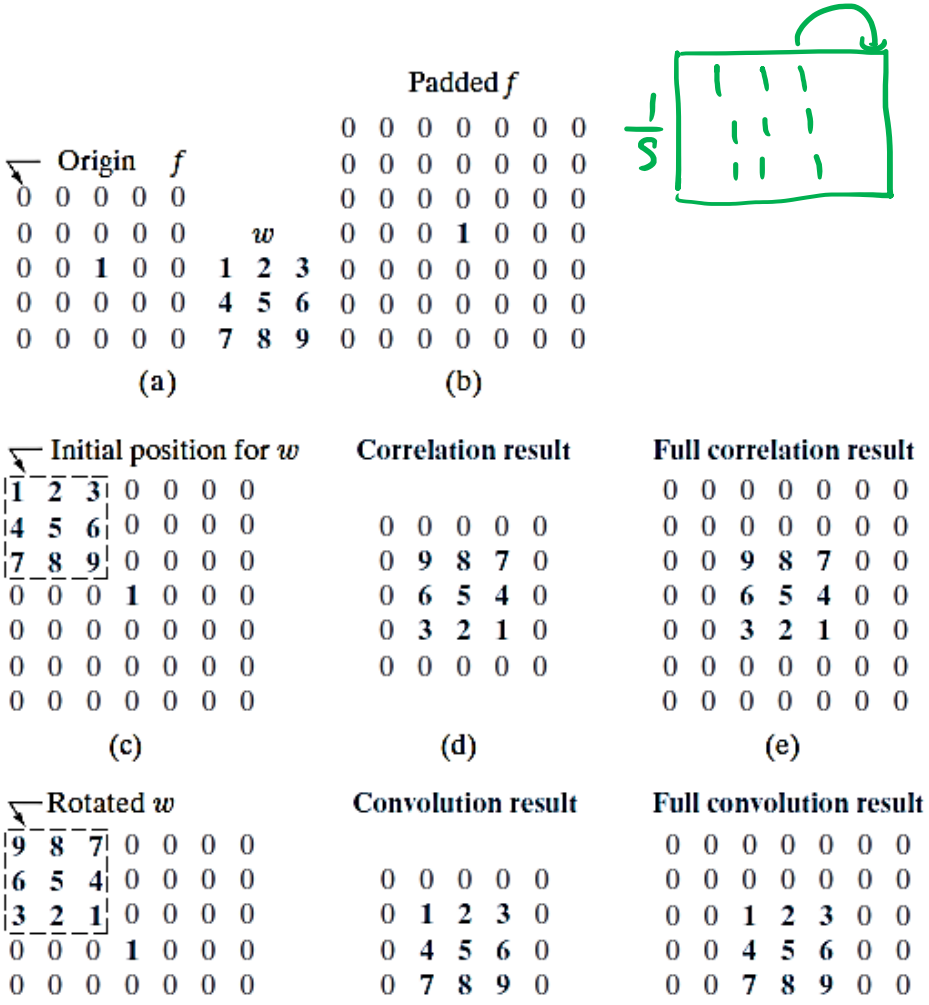



**FIGURE 3.30** Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

- See that convolution of a function with an impulse copies the function at the location of the impulse
- If the filter mask is symmetric, correlation and convolution yield the same result

# Spatial Correlation and Convolution

- Using correlation or convolution to perform spatial filtering is a matter of preference
  - Either can perform the function of the other by a simple rotation of the filter
  - Important is the filter mask used in a given filtering task be specified in a way that corresponds to the intended operation
- The linear spatial filtering discussed here are based on convolution
- Convolution filter, Convolution mask, Convolution kernel → spatial filter (not necessarily used for true convolution)
- Convolving a mask with an image is often used to denote the sliding, sum-of-products process

# Generating Spatial Filter Mask

- Example:

Consider the image: $\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$

Suppose the second row of the image is to be highlighted, what is the mask that is needed?

Correlation (SOP)

→ Convolution

→ Mask are symmetric

FM → SOP

180°

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

FM → SOP

| 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|

↳ FM → SOP.

| 1 | 2 | 3 |
|---|---|---|
| 4 | 5 | 6 |
| 7 | 8 | 9 |

180°

| 1 | 4 | 7 |
|---|---|---|
| 2 | 5 | 8 |
| 3 | 6 | 9 |

90°

| 7 | 8 | 9 |
|---|---|---|
| 4 | 5 | 6 |
| 1 | 2 | 3 |

=

| 3 | 2 | 1 |
|---|---|---|
| 6 | 5 | 4 |
| 9 | 8 | 7 |

| 1 | 2 | 1 |
|---|---|---|
| 2 | 4 | 2 |
| 1 | 2 | 1 |

→

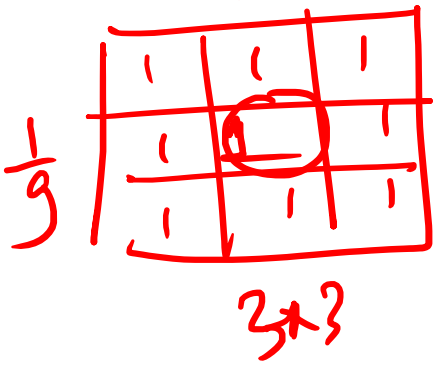
180°

| 1 | 2 | 1 |
|---|---|---|
| 2 | 4 | 2 |
| 1 | 2 | 1 |

symmetric → both are same

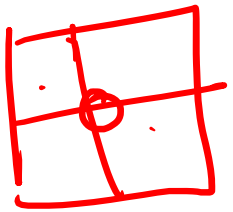
Smoothing

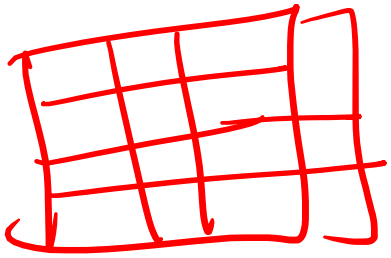
(1) box filter





odd-size mask ⇒ give us a unique centre of the mask.





Smoothening → [blurring / averaging]

averaging.

(1) box filter: ⇒ $\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ ⇒ $\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$

(Simple avg filter)

3×3 5×5

replacing f(x,y) ⇒ avg of a nxn neighbouring pixels

(2) Weighted avg. ⇒ $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ ⇒ $\frac{1}{(\Sigma w)}\begin{bmatrix} 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 2 \\ 4 & 8 & 16 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$

3×3 = W.A.F. 5×5

# Generating Spatial Filter Mask

- One interesting observation:
  - Overall effect can be predicted on the general pattern, if one uses convolution mask
- If the sum of coefficients of the mask is one, average brightness of the image will be retained
- If the sum of coefficients of the mask is zero, average brightness of the image will be lost and will return a dark image
- If coefficients are alternating positive and negative, the mask is a filter that will sharpen an image
- If coefficients are all positive, it is a filter that will blur the image

# Smoothing Spatial Filters

- Used for blurring and for noise reduction
- Blurring is used in preprocessing steps, such as
  - Removal of small details from an image prior to large object extraction, and
  - Bridging of small gaps in lines or curves
- Noise reduction can be accomplished by blurring with a linear filter and also by nonlinear filtering
- The output/response of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask
- Also called Averaging Filters or Lowpass Filters

# Smoothing Linear Filters

- Replacing the value of every pixel in an image by the average of the gray levels in the neighborhood will reduce the "sharp" transitions in gray levels
- Sharp intensity transitions
  - random noise in the image
  - edges of objects in the image
- Thus, smoothing can reduce noises (desirable) and blur edges (undesirable)
- Other uses
  - smoothing of false contours resulting from using an insufficient number of intensity levels
  - reduction of irrelevant detail in an image (here, "irrelevant" means pixel regions that are small wrt the size of filter mask)

# General Form : Smoothing Mask

- Filter of size $m$ x $n$ ($m$ and $n$ odd)

$$g(x, y) = \frac{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\, f(x+s, y+t)}{\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)}$$

summation of all coefficient of the mask (computed once)

The complete filtered image is obtained by applying the operation for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$

# 3x3 Smoothing Linear Filters

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

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

3x3

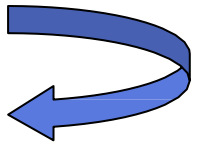
box filter

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

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 2 | 4 | 2 |
| 1 | 2 | 1 |

3x3

weighted average



the center is the most important and other pixels are inversely weighted as a function of their distance from the center of the mask

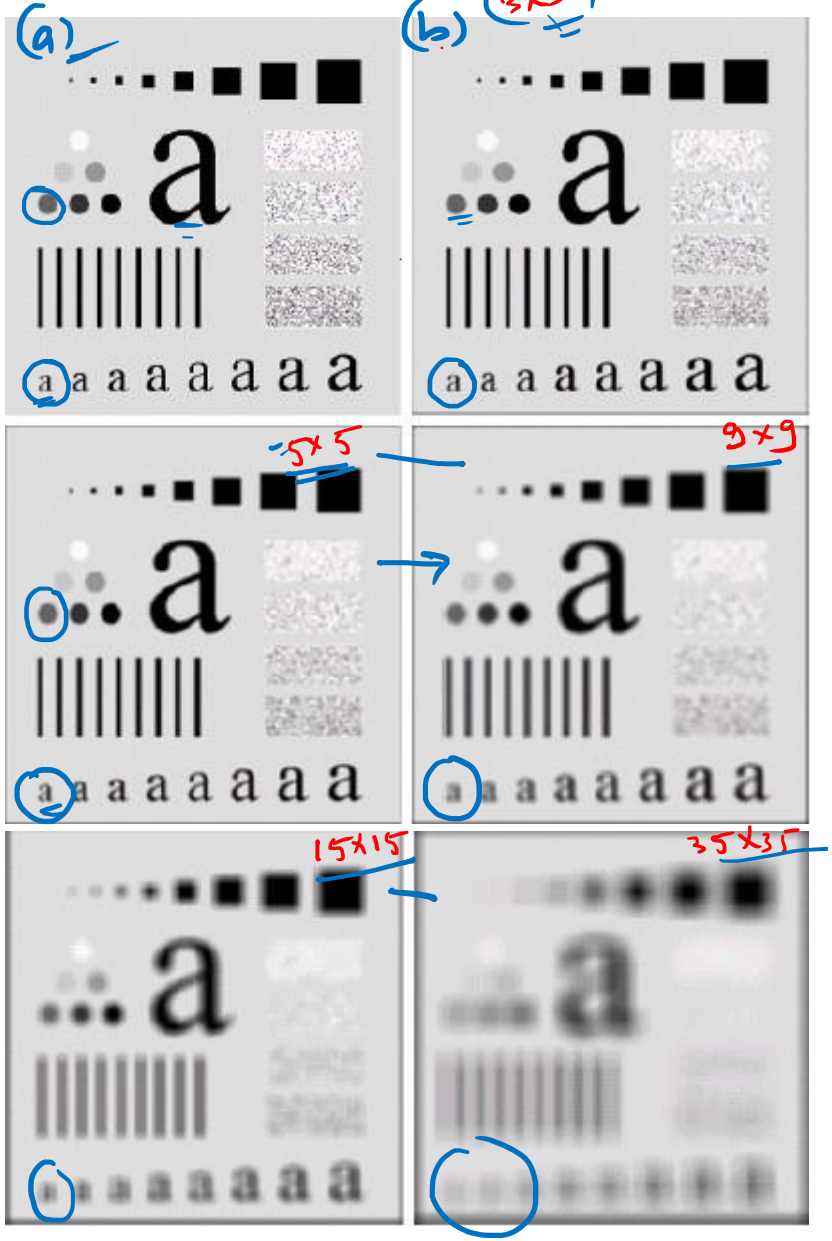
# Weighted Average Filter

- The basic strategy behind weighting the center point the highest and then reducing the value of the coefficients as a function of increasing distance from the origin is simply an attempt to reduce blurring in the smoothing process.

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Weighted Average Filter

# Example 1



| | |
|---|---|
| a | b |
| c | d |
| e | f |

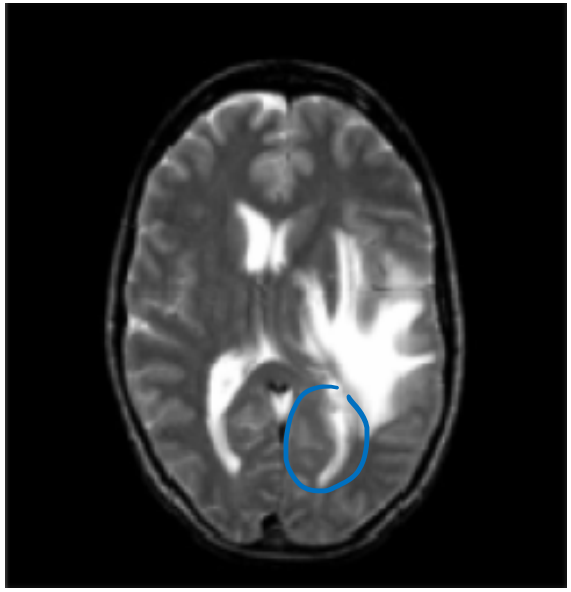
(a) Original image of size 500x500 pixel

(b)-(f) Results of smoothing with square averaging filter masks of sizes $n$ = 3, 5, 9, 15, and 35 respectively

Note:

- details that are of approx. the same size as the filter mask are affected considerably more
- the size of the mask establishes the relative size of the objects that will be blended with the background
- big mask is used to eliminate small objects from an image

# Example 2

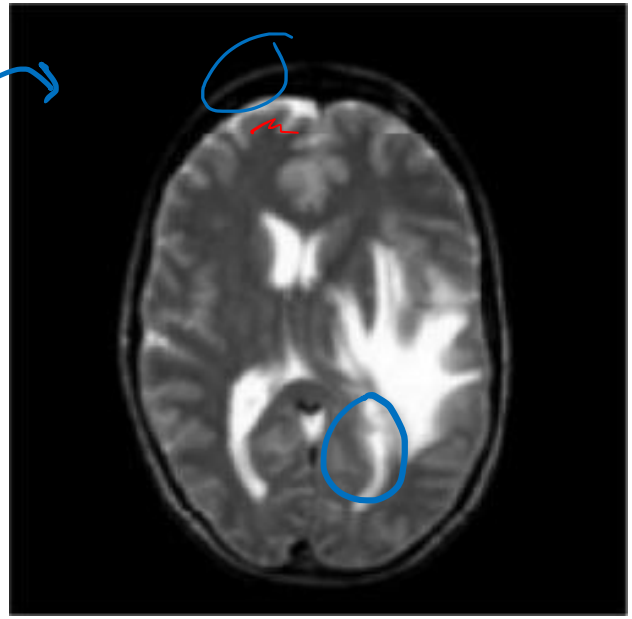


Before smoothing

3 x 3

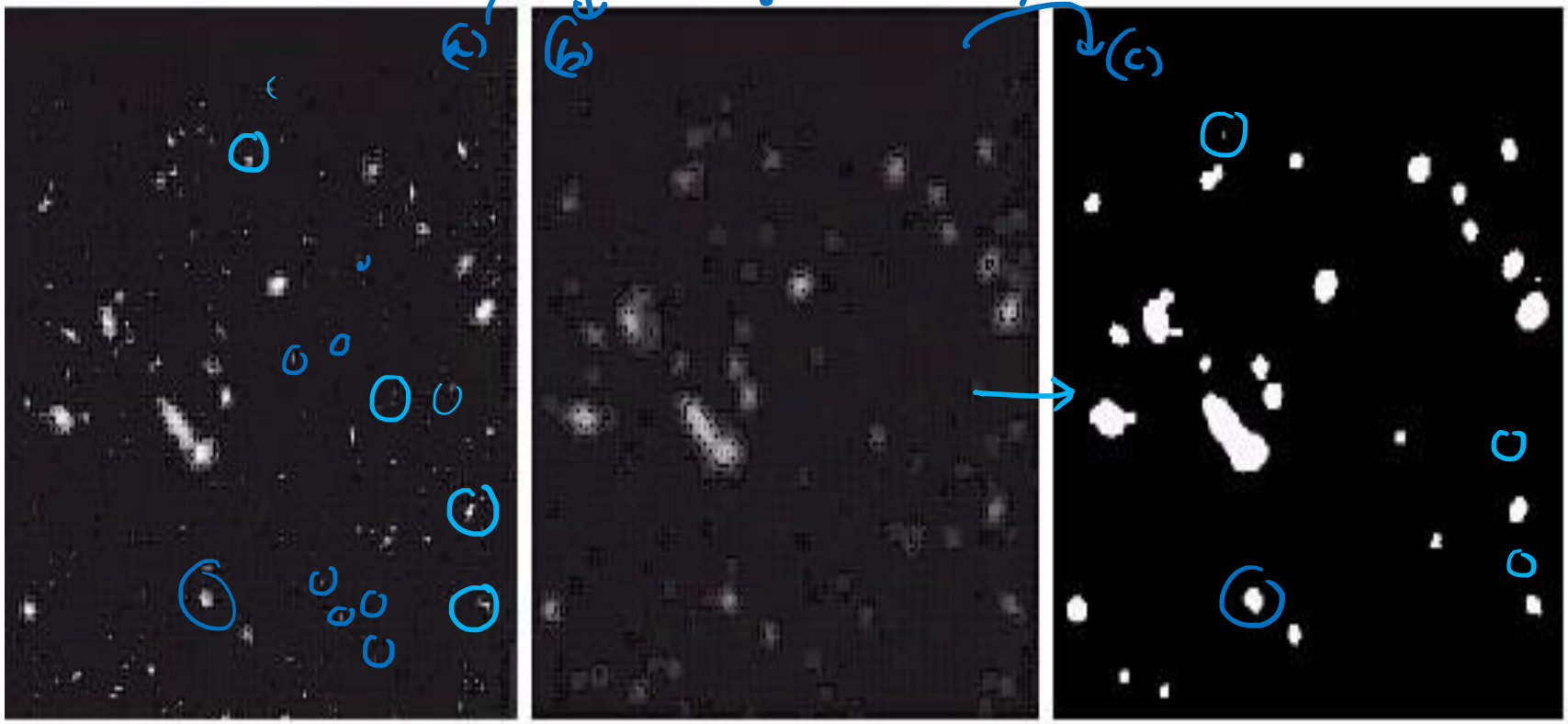
| | | |
|---|---|---|
| 1/16 | 2/16 | 1/16 |
| 2/16 | 4/16 | 2/16 |
| 1/16 | 2/16 | 1/16 |

weighted Avg



After smoothing

# Example 3




original image

result after smoothing with 15x15 averaging mask

result of thresholding

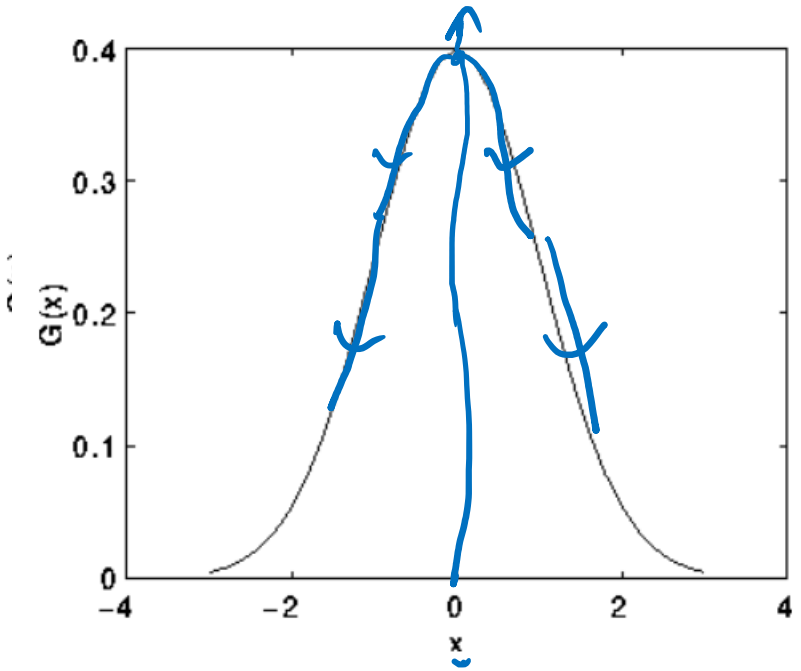
we can see that the result after smoothing and thresholding, the remains are the largest and brightest objects in the image

# Gaussian Filters

The Gaussian distribution in 1-D has the form:

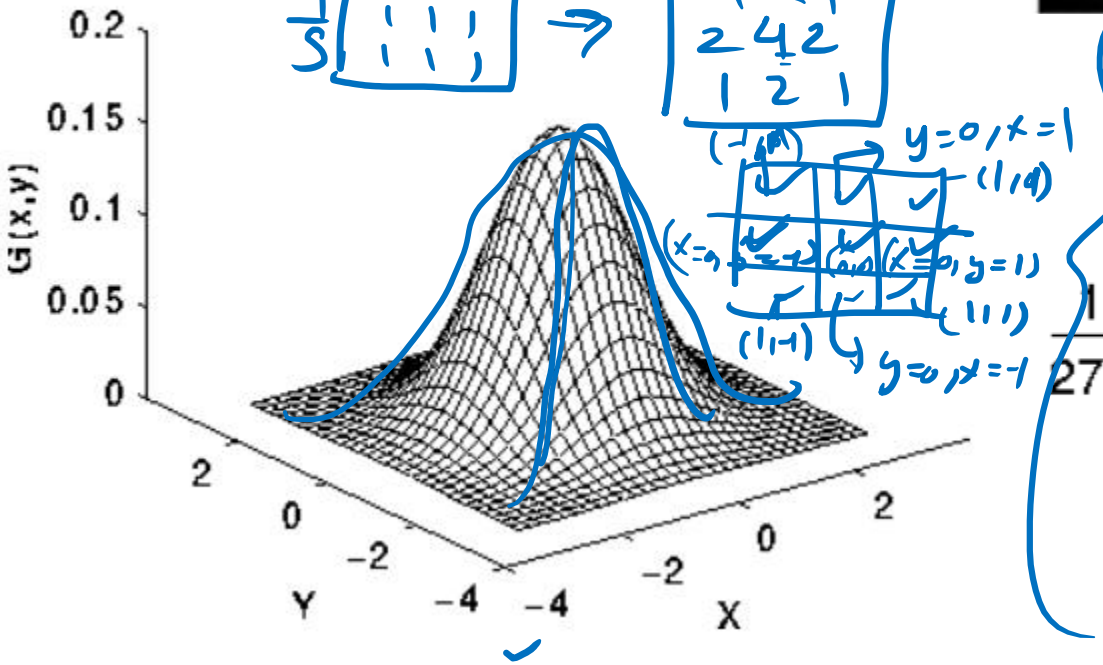
$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

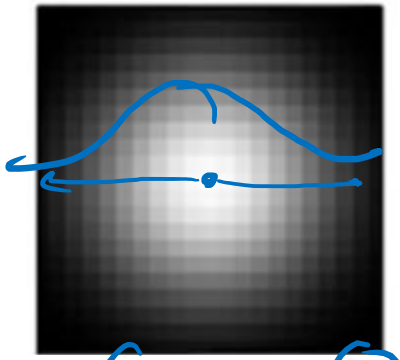
$N(0, \sigma)$



# Gaussian Filters

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



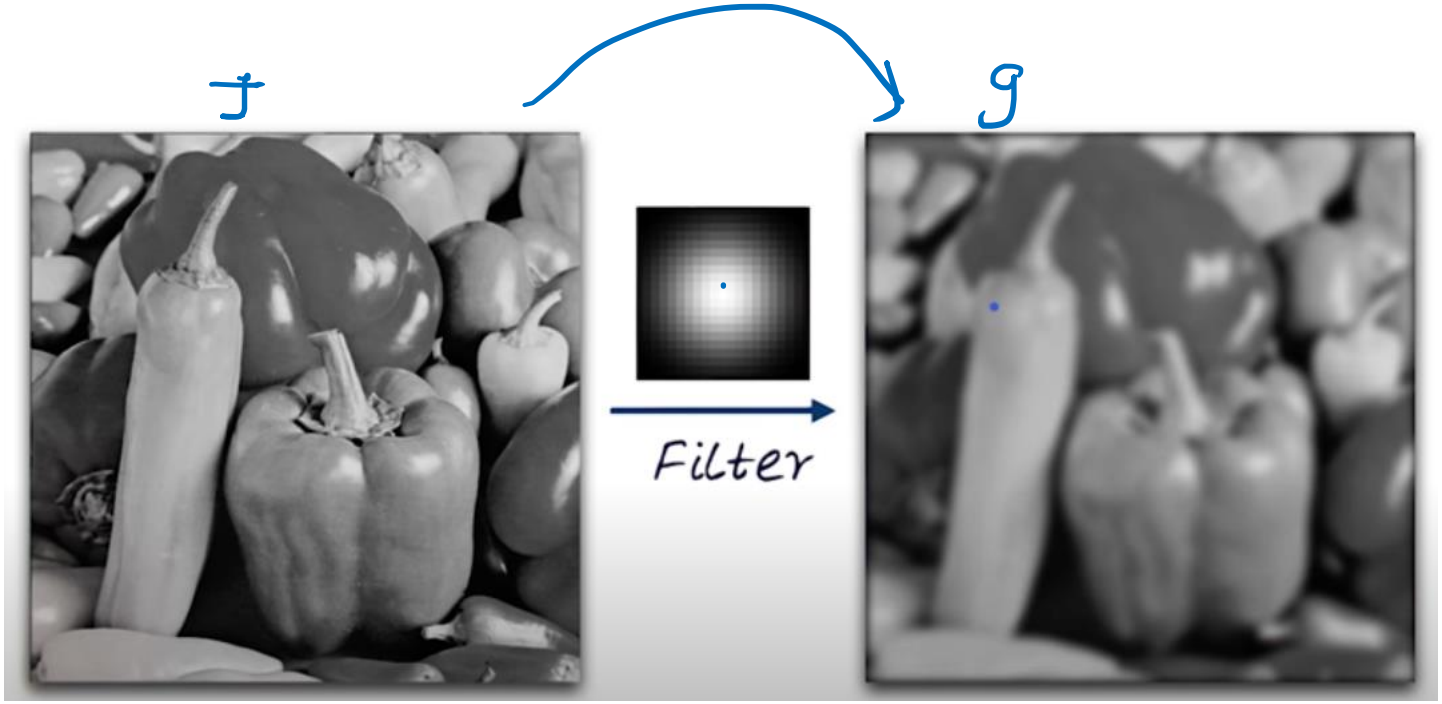



| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 4 | 1 |
| 4 | 16 | 26 | 16 | 4 |
| 7 | 26 | 41 | 26 | 7 |
| 4 | 16 | 26 | 16 | 4 |
| 1 | 4 | 7 | 4 | 1 |

$\frac{1}{273}$

**Figure 3** Discrete approximation to Gaussian function with $\sigma$=1.0

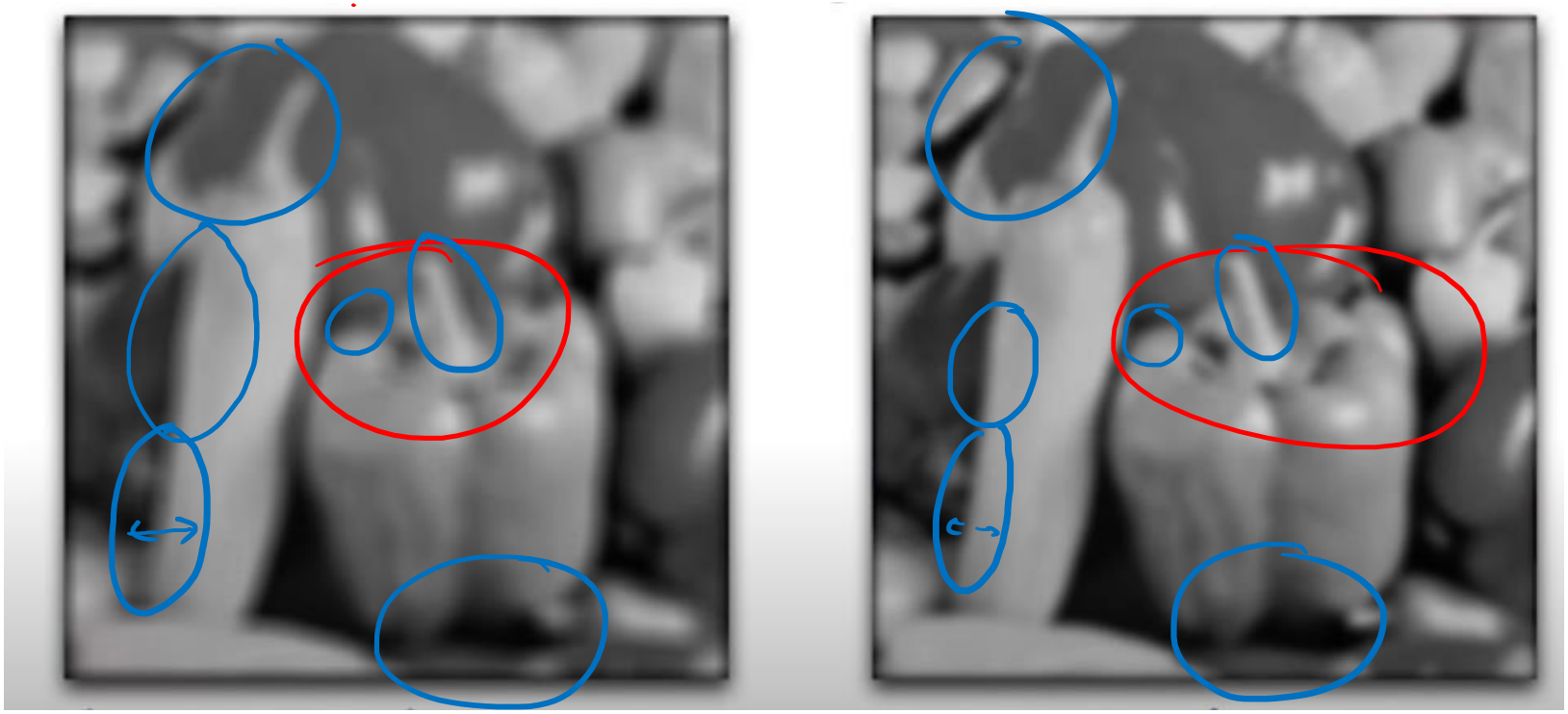
# Gaussian Filters




Smoothed/ blurred version.

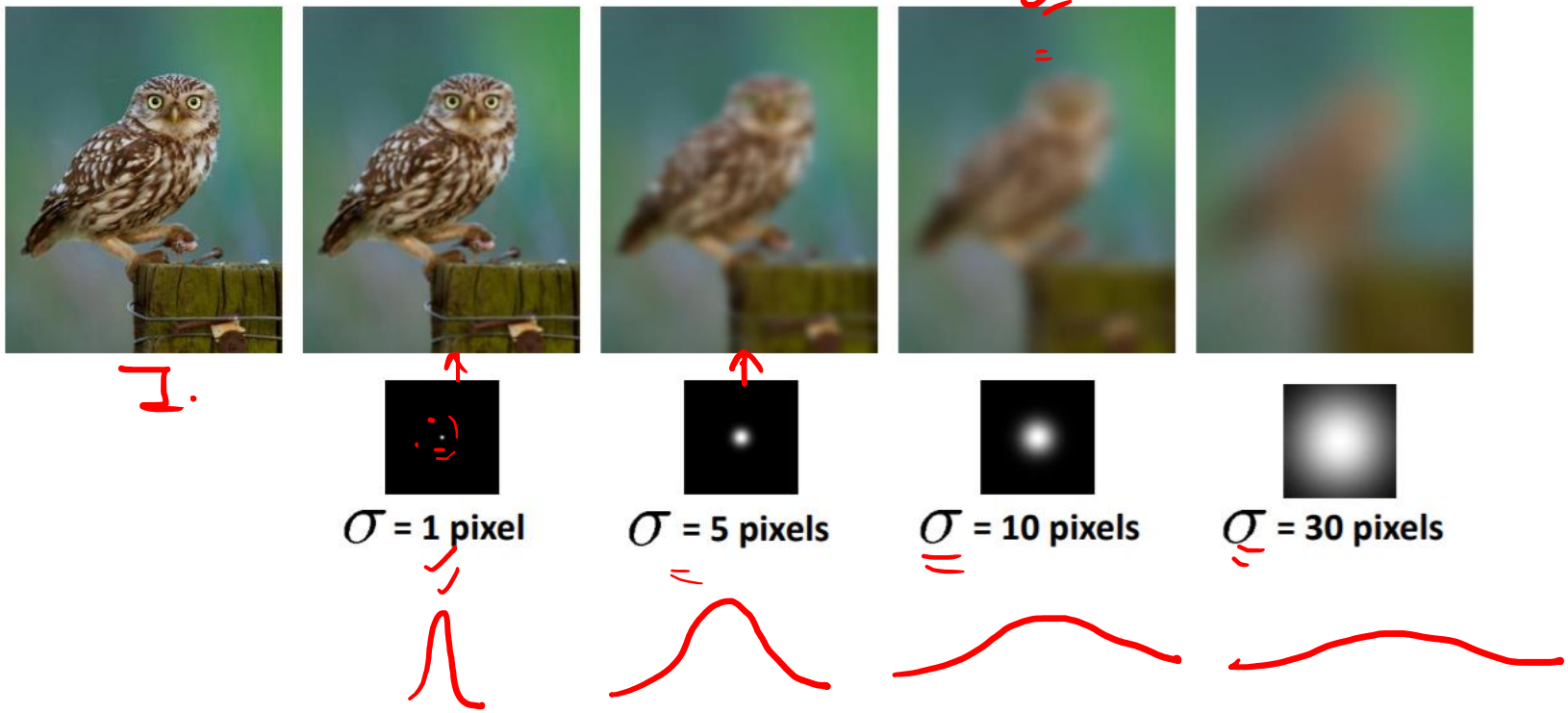
# Gaussian Filters

- Average vs Gaussian



# Gaussian Filters

- Effect of varying $\sigma$, as we increase it blurring effect will increase.

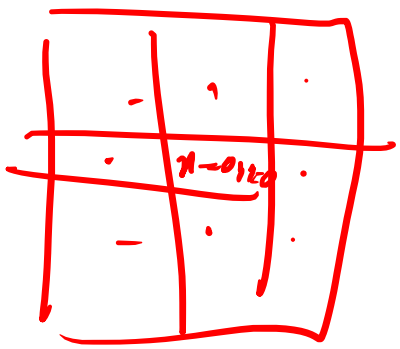


Smoothening (linear operation)

(1) box $= \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ ⇓ sup

(2) Weighted $= \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

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



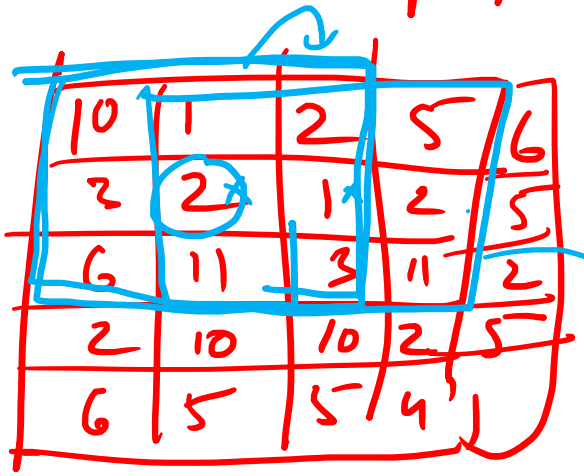
# (Median) filtering.

$A = [2, 12, 6, 1, 5]$

mean $= (2+12+6+1+5)$
$= 26/5 = 5.2$.

$[1, 2, \textcircled{5}, 6, 12]$ ← median $= 5$.

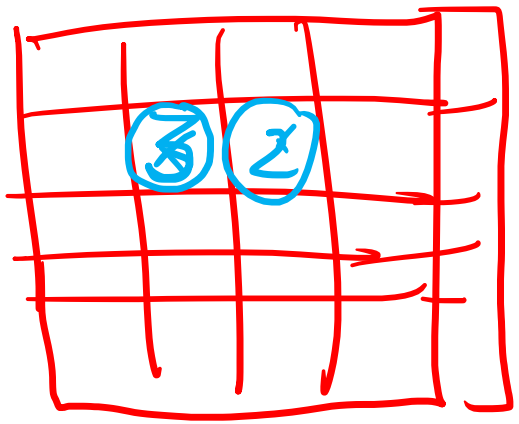
Replaces a pixel at position $f(x,y)$ with the median value specified by nxn neighbourhood




$f(x,y)$

$\begin{bmatrix} 10 & 1 & 2 \\ 3 & 2 & 1 \\ 6 & 11 & 3 \end{bmatrix}$ → $[1, 1, 2, 2, \textcircled{3}, 3, 6, 10, 11]$

$\begin{bmatrix} 1 & 2 & 5 \\ 2 & 1 & 2 \\ 11 & 3 & 11 \end{bmatrix}$ → $[1, 1, 2, 2, \textcircled{2}, 5, 5, 11, 11]$




$g(x,y)$

# Order-Statistics Filters

- Nonlinear spatial filters
- The response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and

  then replacing the value of the center pixel with the value determined by the ranking results
- Example ($n$ x $n$ is the size of the mask)
  1. median filter : $R = \text{median}\{z_k \mid k = 1, 2, \ldots, n \text{ x } n\}$
  2. max filter : $R = \max\{z_k \mid k = 1, 2, \ldots, n \text{ x } n\}$
  3. min filter : $R = \min\{z_k \mid k = 1, 2, \ldots, n \text{ x } n\}$
- Most popular: median filter

# Median Filters

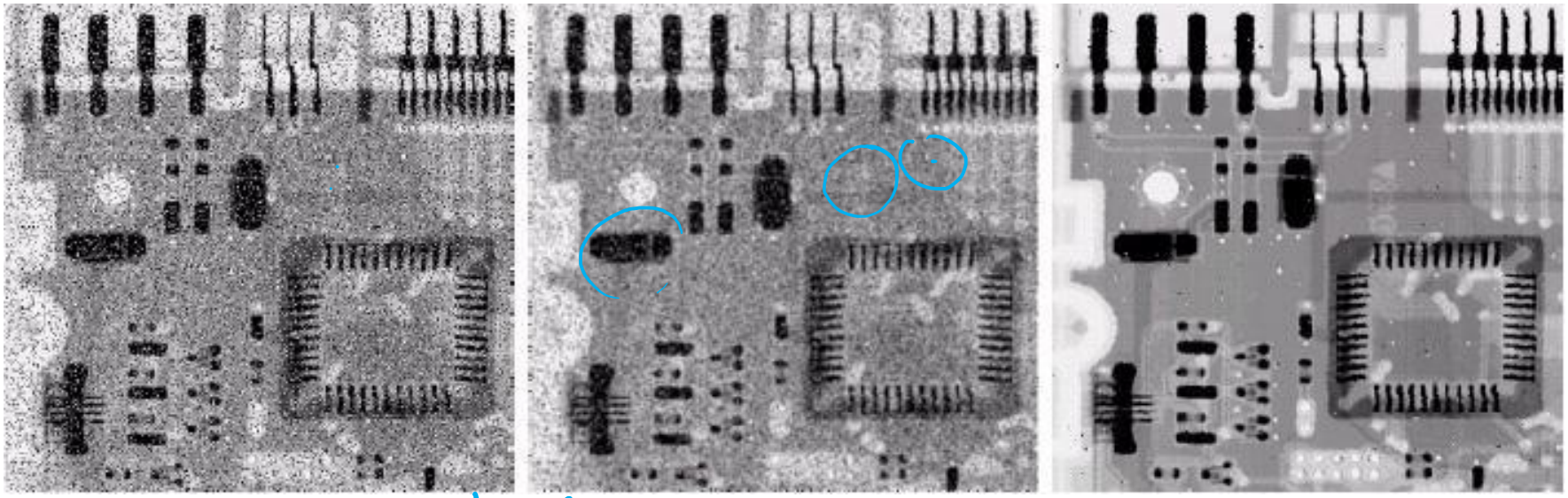
- It replaces the value of the centre pixel by the Median/Min/Max value in the neighborhood pixels
  - Original value of the pixel is included in the computation
  - Median: $50^{th}$ percentile of a ranked set of nos.
    - For ex: in a 3x3 neighborhood the median is the 5th largest value (5x5 neighborhood -> 13th largest value)
  - $100^{th}$ percentile: max filter; $0^{th}$ percentile: min filter
- Quite popular because for certain types of random noise (impulse noise ⇨ salt and pepper noise), they provide excellent noise-reduction capabilities, with considering less blurring than linear smoothing filters of similar size

# Median Filters

- Forces the points with distinct gray levels to be more like their neighbors
- Isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $n^2/2$ (one-half the filter area), are eliminated by an $n$ x $n$ median filter
- Eliminated ≡ forced to have the value equal the median intensity of the neighbors
- Larger clusters are affected considerably less

# Median Filtering: Example 1

Remove salt & pepper kind of noise



(a) (b) avg. (c) median

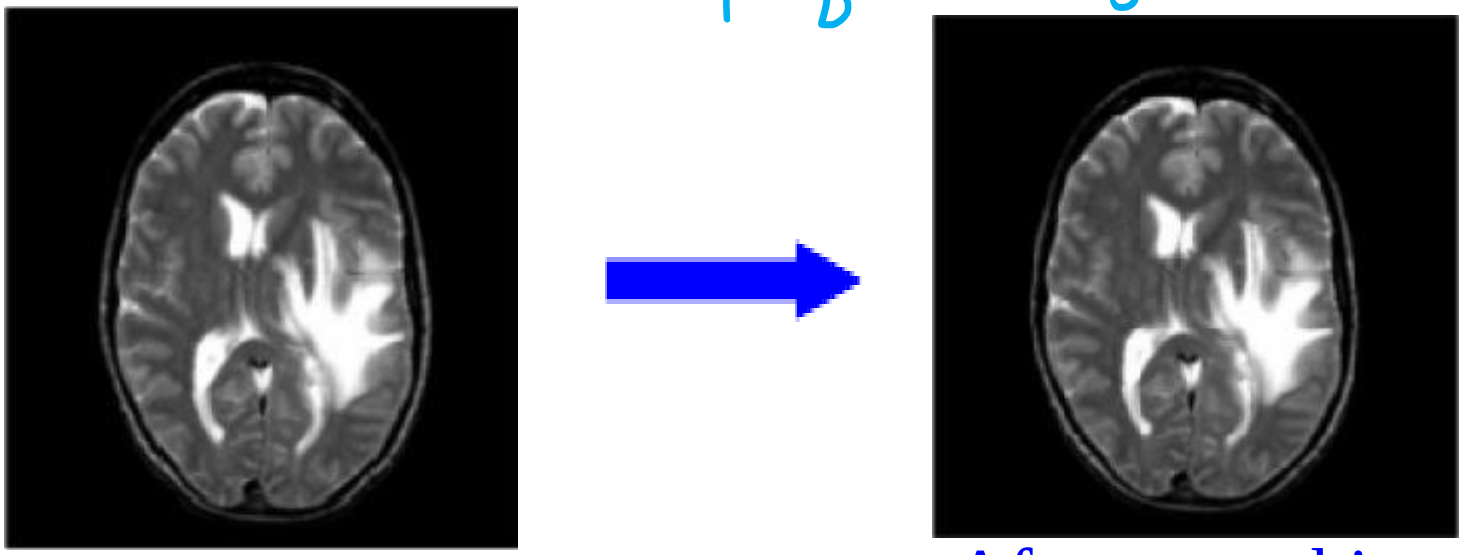
(a) X-ray image of circuit board corrupted by salt-and-pepper noise

(b) Noise reduction with a 3x3 averaging mask
-less noise reduction but more blurring

(c) Noise reduction with a 3x3 median filter

# Median Filtering: Example 2

Don't perform very well for smoothing



Before smoothing

After smoothing

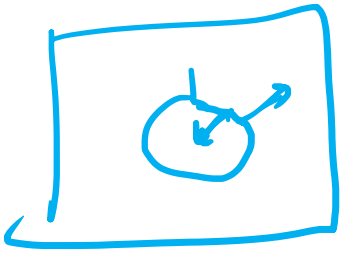
The smoothed MR brain image obtained by using median filtering over a fixed neighborhood of 3x3 pixels

Smoothening → blurring / avg / (reduce the HFC)
∫ integral

Sharpening → derivatives (increase the HFC)
highlight the edges (Edges / noise)

# Image Sharpening using Filters

(Edge Detection)



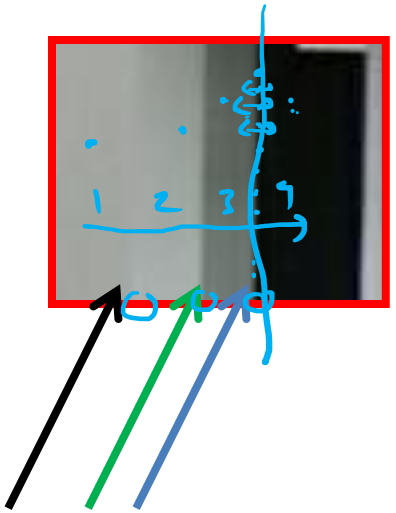
# Sharpening Spatial Filters

- To highlight fine detail in an image
- or to enhance detail that has been blurred, either in error or as a natural effect of a particular method of image acquisition
- Blurring vs. Sharpening
  - as we know that blurring can be done in spatial domain by pixel averaging in a neighbors
  - since averaging is analogous to integration
  - thus, we can guess that the sharpening must be accomplished by spatial differentiation

# Closeup of edges

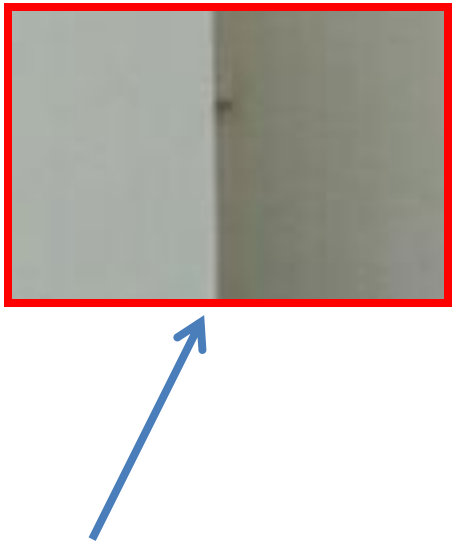
Source: D. Hoiem

# Closeup of edges



Source: D. Hoiem

# Closeup of edges



Source: D. Hoiem

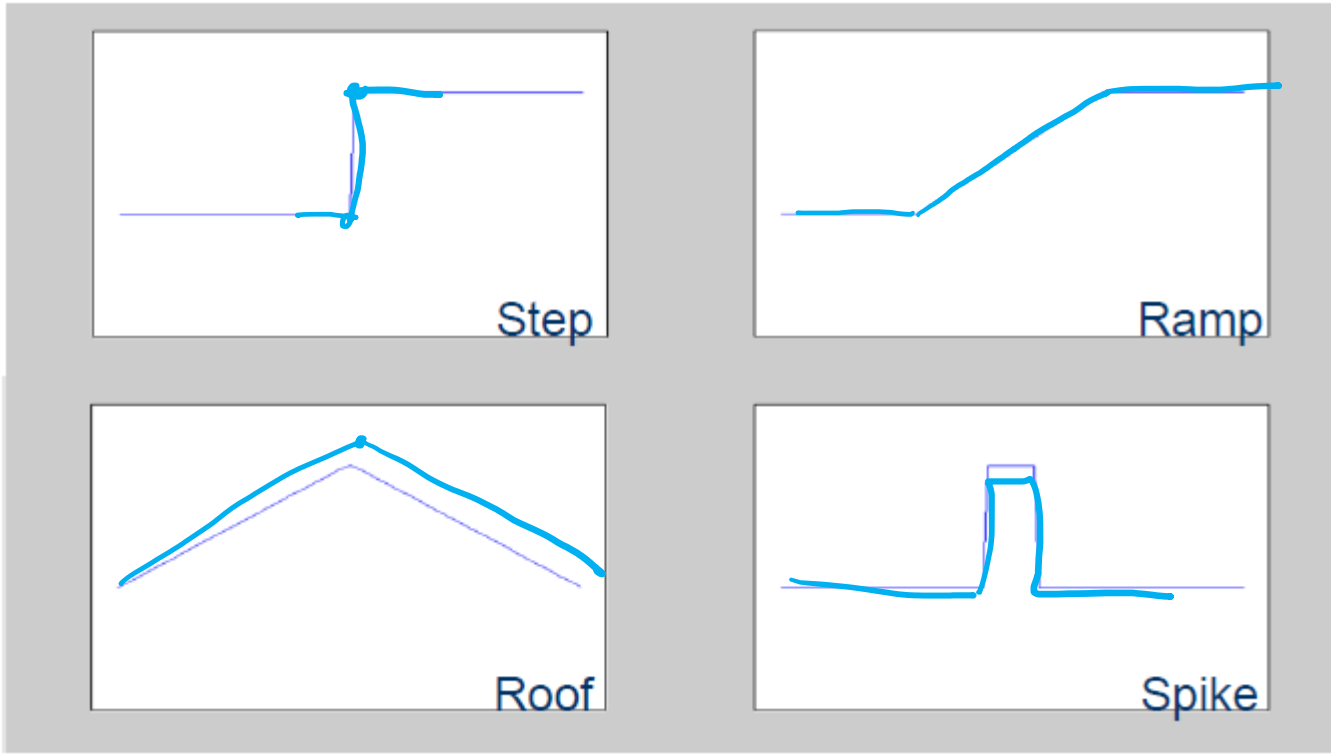
# Closeup of edges

Source: D. Hoiem

# Edge

Change / discontinuity in intensity

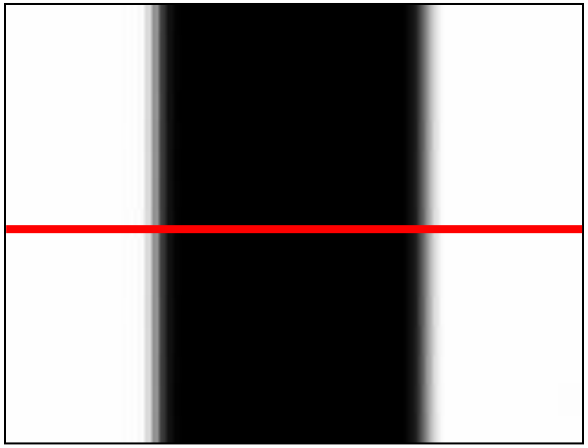
- Discontinuity of intensities in the image



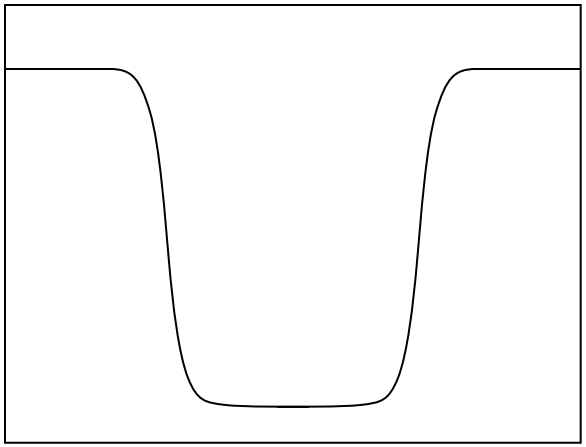
# Characterizing edges

- place of rapid change in the image intensity function

image



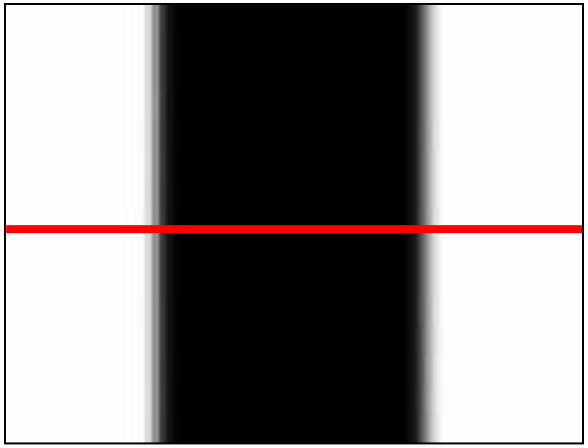
intensity function
(along horizontal scanline)



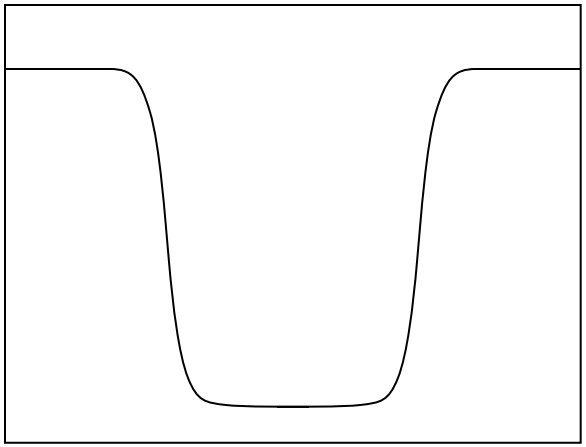
# Characterizing edges

- place of rapid change in the image intensity function

image

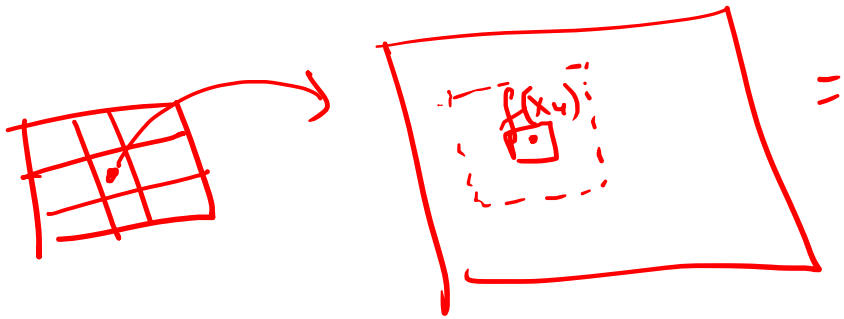


intensity function
(along horizontal scanline)



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| (a) | f(x) | 10 | 10 | 10 | 10 | 10 | 20 | 20 | 20 | 20 | 20 |
| (b) | f'(x) | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 |
| (c) | f"(x) | 0 | 0 | 0 | 0 | 0 | 10 | -10 | 0 | 0 | 0 |

CONVOLUTION →




Smoothing

- box $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$
- weighted avg $\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}$
- Gaussian $\frac{1}{2\pi\sigma}e^{-\frac{(x^2+y^2)}{2\sigma^2}}$
- median filter

Sharpening filters

- first order Derivative

$\frac{\partial f}{\partial x} = I_x =$ [1 | -1]

$\frac{\partial f}{\partial y} = I_y =$ [-1 / 1] = [1 / -1]

R.C = [-1 0 / 0 1] [0 -1 / 1 0]

Prewitt [-1 -1 -1 / 0 0 0 / 1 1 1] [-1 0 1 / -1 0 1 / -1 0 1]

Sobel [-1 -2 -1 / 0 0 0 / 1 2 1] [-1 0 1 / -2 0 2 / -1 0 1]

# Derivative Operator

- The strength of the response of a derivative operator is proportional to the degree of intensity discontinuity of the image at the point at which the operator is applied
- Thus, image differentiation
  - enhances edges and other discontinuities (noise)
  - deemphasizes area with slowly varying gray-level

# Derivative Operator

- The strength of the response of a derivative operator is proportional to the degree of intensity discontinuity of the image at the point at which the operator is applied
- Thus, image differentiation
  - enhances edges and other discontinuities (noise)
  - deemphasizes area with slowly varying gray-level values
- First-order derivative
- Second-order derivative

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

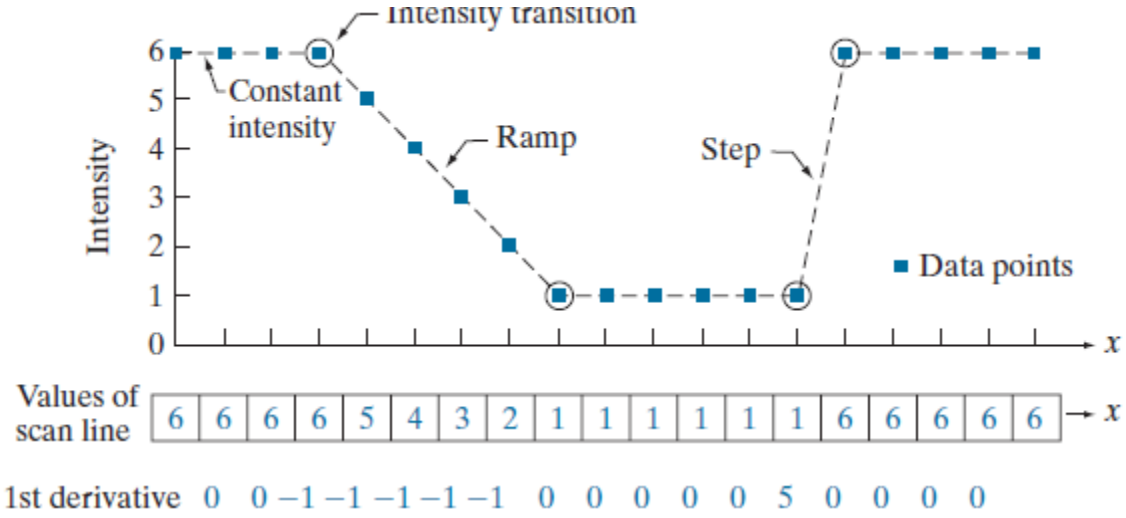
$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

# Derivative Operator

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

- Derivatives of a digital function are defined in terms of differences
- Various ways to define these differences




- for a first derivative
  - must be zero in flat segments (areas of constant gray-level values);
  - must be nonzero at the onset of a gray-level step or ramp; and
  - must be nonzero along ramps

# First Order Image Derivatives

Image $I$

$$I_x = I * \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$I_y = I * \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

# First Order Image Derivatives

- First derivatives are implemented using the magnitude of the gradient.
- For a function $f(x, y)$, the gradient of f at coordinates $(x, y)$ is defined as a 2-d column vector
- Geometrical property of gradient vector:
  it points in the direction of the greatest rate of change of $f$ at location $(x, y)$

$$\nabla \mathrm{f} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Let $\alpha(x, y)$ represent the direction angle of the vector $\nabla f$ at $(x, y)$

$$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

angle is measured wrt the X-axis

The direction of an edge at $(x, y)$ is perpendicular to the direction of the gradient vector at that point

# First Order Image Derivatives

- The magnitude of this vector is the value of the rate of change in the direction of the gradient vector at $(x, y)$

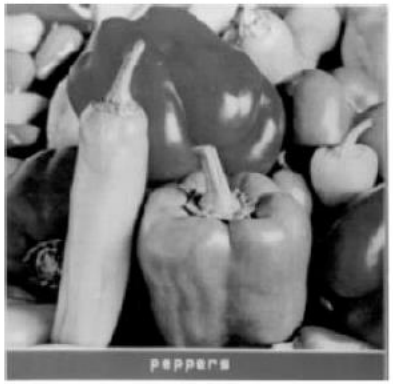
$$M(x,y) = mag(\nabla f) = [g_x^2 + g_y^2]^{1/2} = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{1/2}$$

- $M(x, y)$ is an image of the same size as the original, created when x and y are allowed to vary over all pixel locations in $f$ → "gradient image" or simply "gradient"
- Gradient vector is a linear operator but its magnitude is not
- Computationally it is more suitable to approximate the magnitude by absolute values

$$M(x,y) \approx |g_x| + |g_y|$$

# Edge Detection using Gradient

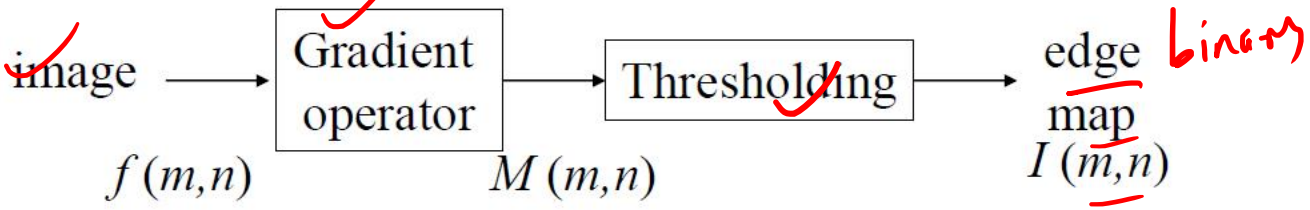
- Motivation: detect changes  change in the pixel value ⟶ large gradie




(a) Original


(b) Horizontal magnitude


(c) Vertical magnitude



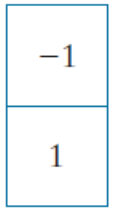

$$I(m,n) = \begin{cases} 1 & |\, g(m,n) \,| > th \\ 0 & otherwise \end{cases}$$

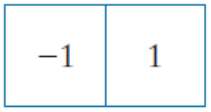
# Gradient Masks

- Computation of the gradient of an image is based on obtaining partial derivatives $g_x$ and $g_y$ at every pixel location
- It is always possible to implement the derivatives in digital form in different ways

$$g_x(x,y) = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y)$$

$$g_y(x,y) = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y)$$



# Gradient Masks

- Computation of the gradient of an image is based on obtaining partial derivatives $g_x$ and $g_y$ at every pixel location
- It is always possible to implement the derivatives in digital form in different ways

- Follow the following notation:

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

**3x3 area representing the gray levels in a neighborhood of an image**

Center point $z_5$ denotes $f(x, y)$ at an arbitrary location $(x, y)$;
$z_1$ denotes $(x - 1, y - 1)$; and so on..

# Gradient Masks

- When diagonal edge is of interest, we need 2-D kernels

  - **Roberts Cross-Gradient Operators:** (Sum of the magnitude of the differences of the diagonal neighbors)
  - $g_x : (z_9 - z_5), \quad g_y : (z_8 - z_6)$
  - Gradient image can be computed as:

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

$$M(x,y) = [(z_9 - z_5)^2 + (z_8 - z_6)^2]^{1/2}$$
$$M(x,y) \approx |z_9 - z_5| + |z_8 - z_6|$$

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

  - These derivatives can be implemented for an entire image by using the mask

| | |
|---|---|
| −1 | 0 |
| 0 | 1 |

| | |
|---|---|
| 0 | −1 |
| 1 | 0 |

Roberts operators

# Gradient Masks

- **Roberts Cross-Gradient Operators:**

| | |
|---|---|
| −1 | 0 |
| 0 | 1 |

| | |
|---|---|
| 0 | −1 |
| 1 | 0 |

Roberts operators

- Masks of size 2 x 2 are awkward to implement because they do not have a clear center.
- It makes the edge point only, NOT the information about the edge orientation.

# Gradient Masks

- Simplest 3x3 approximations
  - **Prewitt Operators:**
  - $g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$
  - $g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

3x3 area representing the gray levels in a neighborhood of an image

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −1 | 0 | 1 |
| −1 | 0 | 1 |

Prewitt operators

# Gradient Masks

- **Sobel Operators:**
- $g_x : (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$
- $g_y : (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

3x3 area representing the gray levels in a neighborhood of an image

| | | |
|---|---|---|
| −1 | −2 | −1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

Sobel operators

- A weight value of 2 is used to achieve some smoothing by giving more importance to the center point.

# Gradient Masks

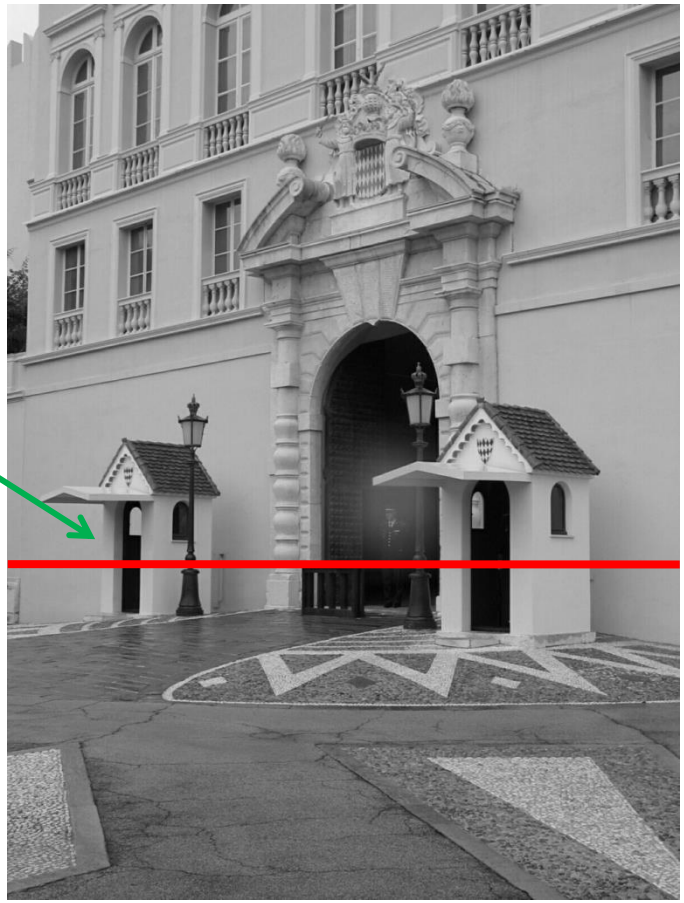
a b
c d

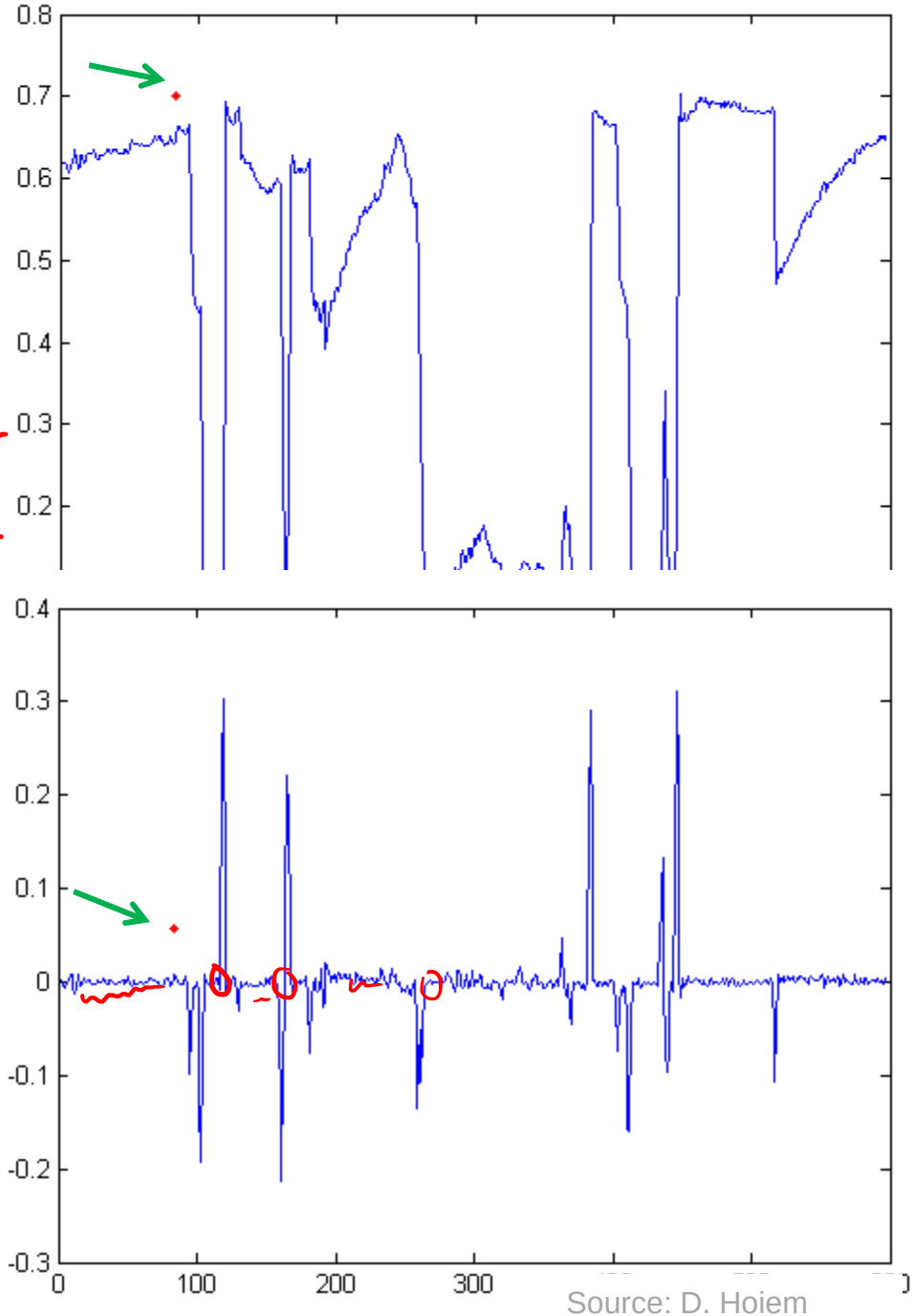
**FIGURE 10.16**
(a) Image of size $834 \times 1114$ pixels, with intensity values scaled to the range [0,1]. (b) $|g_x|$, the component of the gradient in the $x$-direction, obtained using the Sobel kernel in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the kernel in Fig. 10.14(g). (d) The gradient image, $|g_x| + |g_y|$.

# Gradient Masks

- Prewitt and Sobel operators: mostly used in practice for computing digital gradients.
- Prewitt masks: simpler to implement.
- Sobel masks have slightly superior noise-suppression characteristics
  - an important issue when dealing with derivatives
  - Reason: a weight value of 2 is used to achieve some smoothing by giving more importance to the center point
- The sum of coefficients in all gradient masks is 0:
  - They give a response of 0 in areas of constant gray level (as expected from a derivative operator)

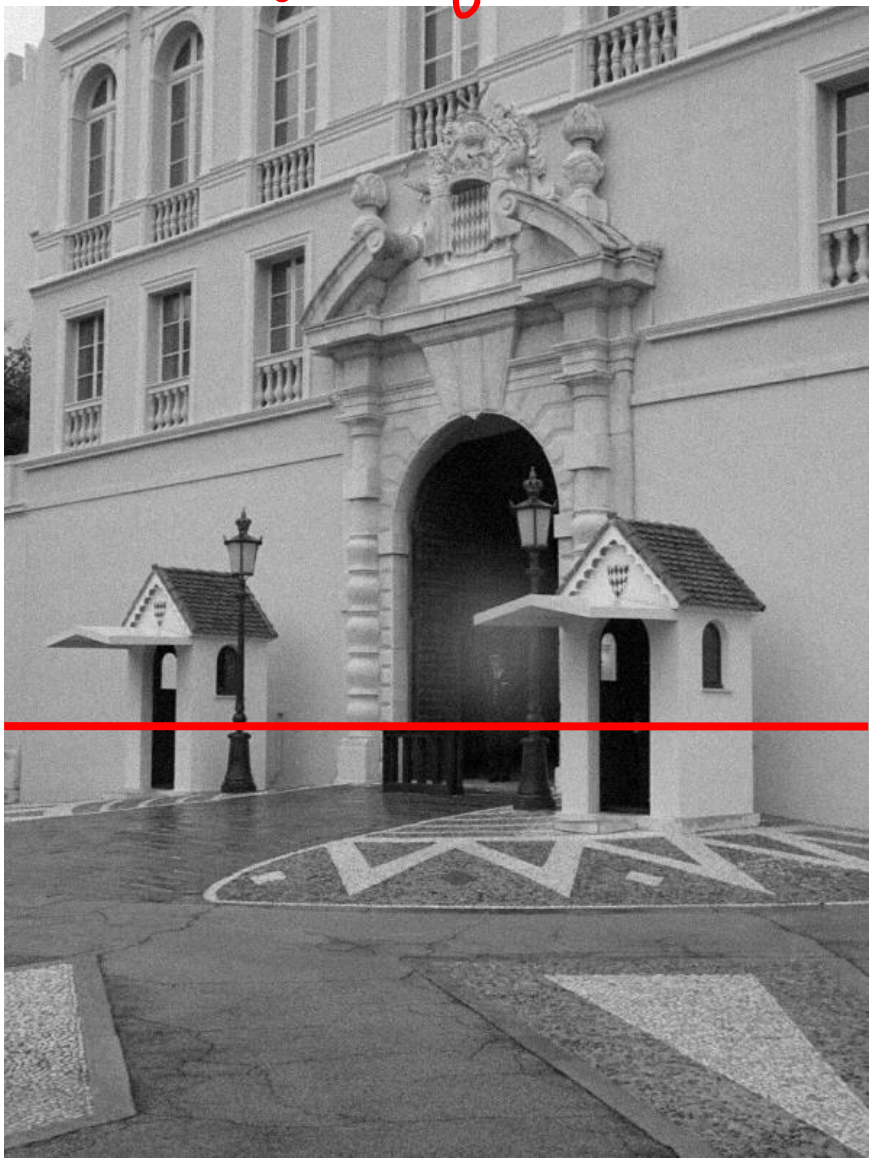
# Intensity profile

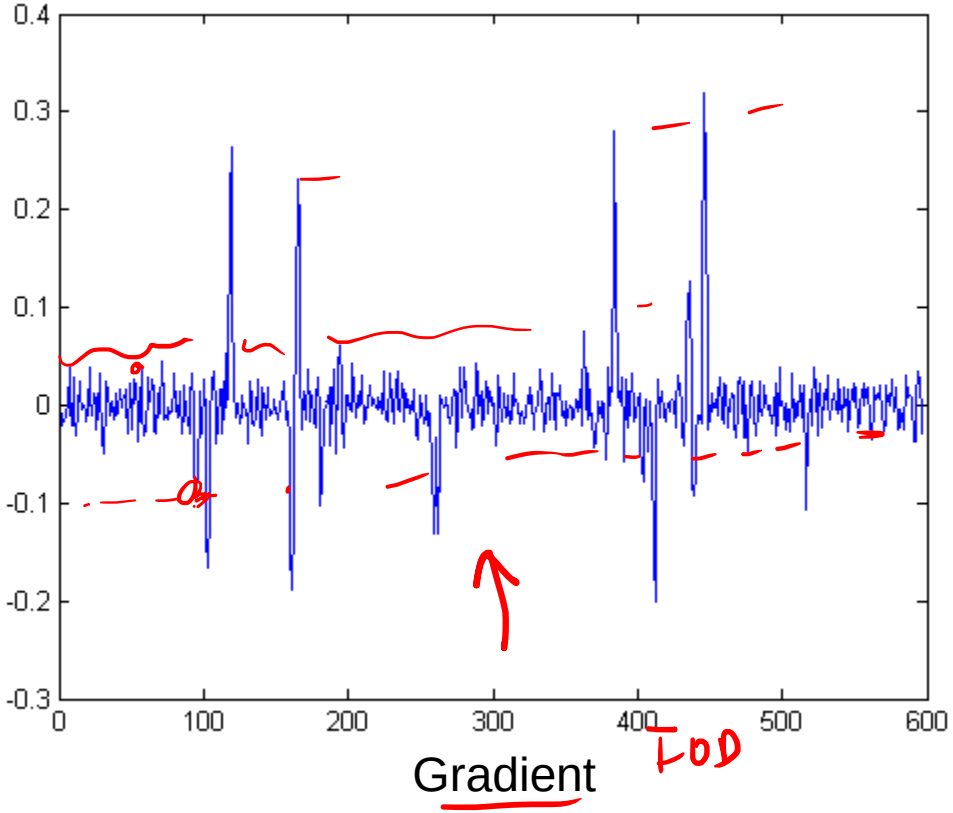





Source: D. Hoiem

# With a little Gaussian noise

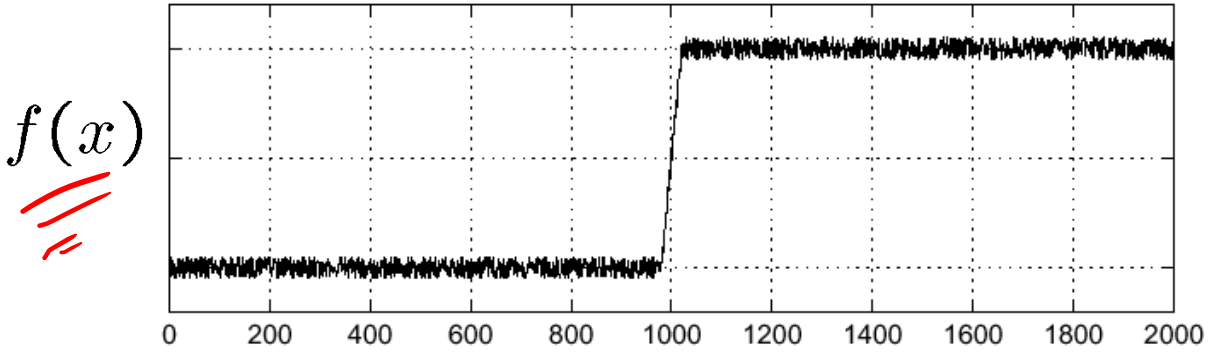


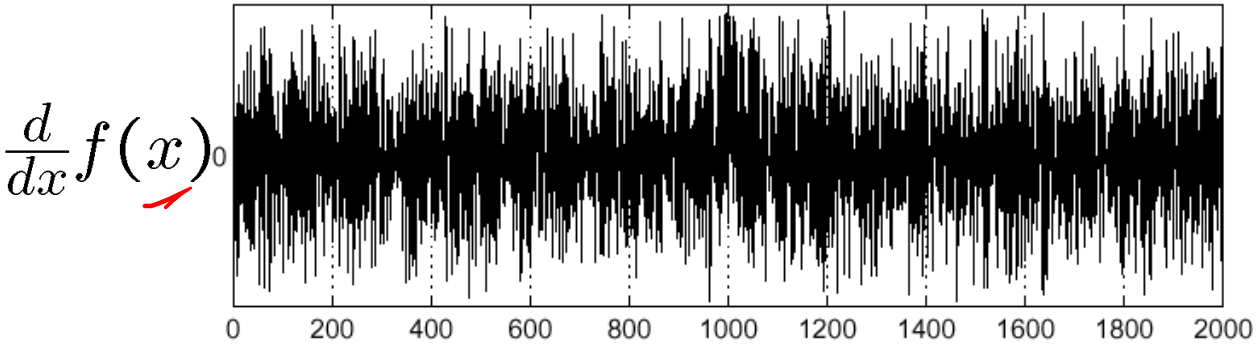



Source: D. Hoiem

# 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?

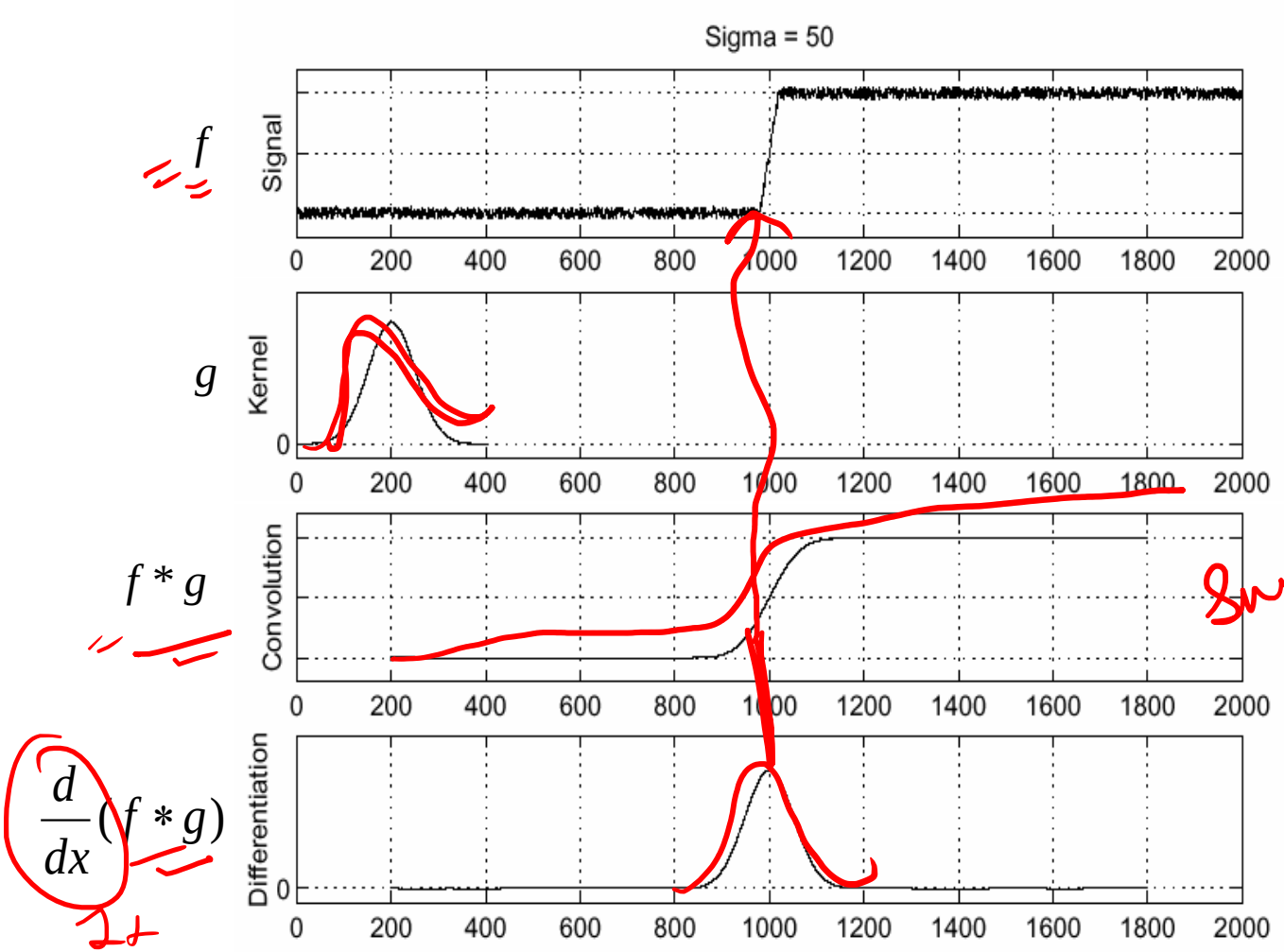
Source: S. Seitz

# Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Source: D. Forsyth

# Solution: smooth first




$f$

$g$

$f * g$

$\frac{d}{dx}(f * g)$

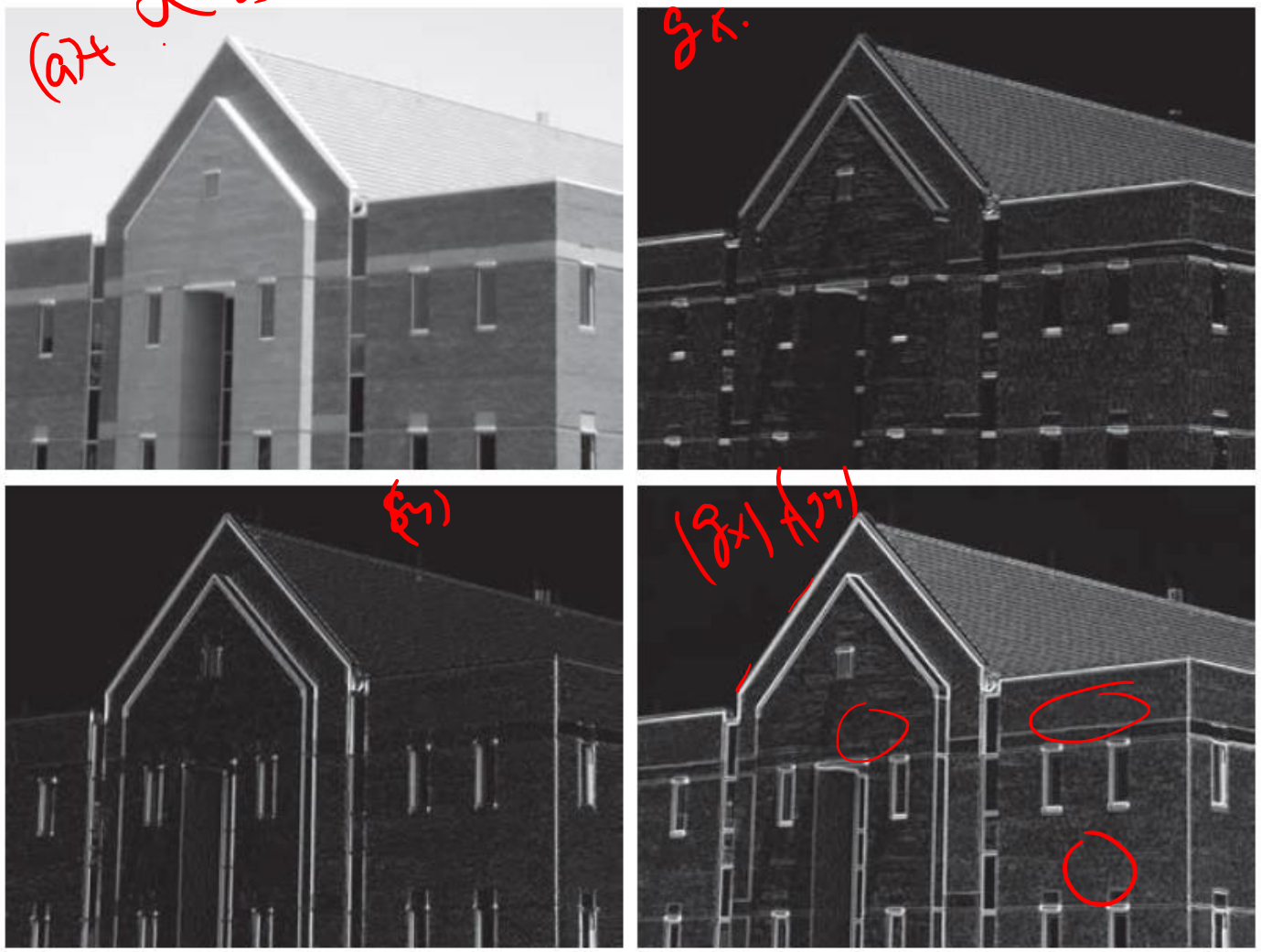
- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

# Sobel on Smoothed image

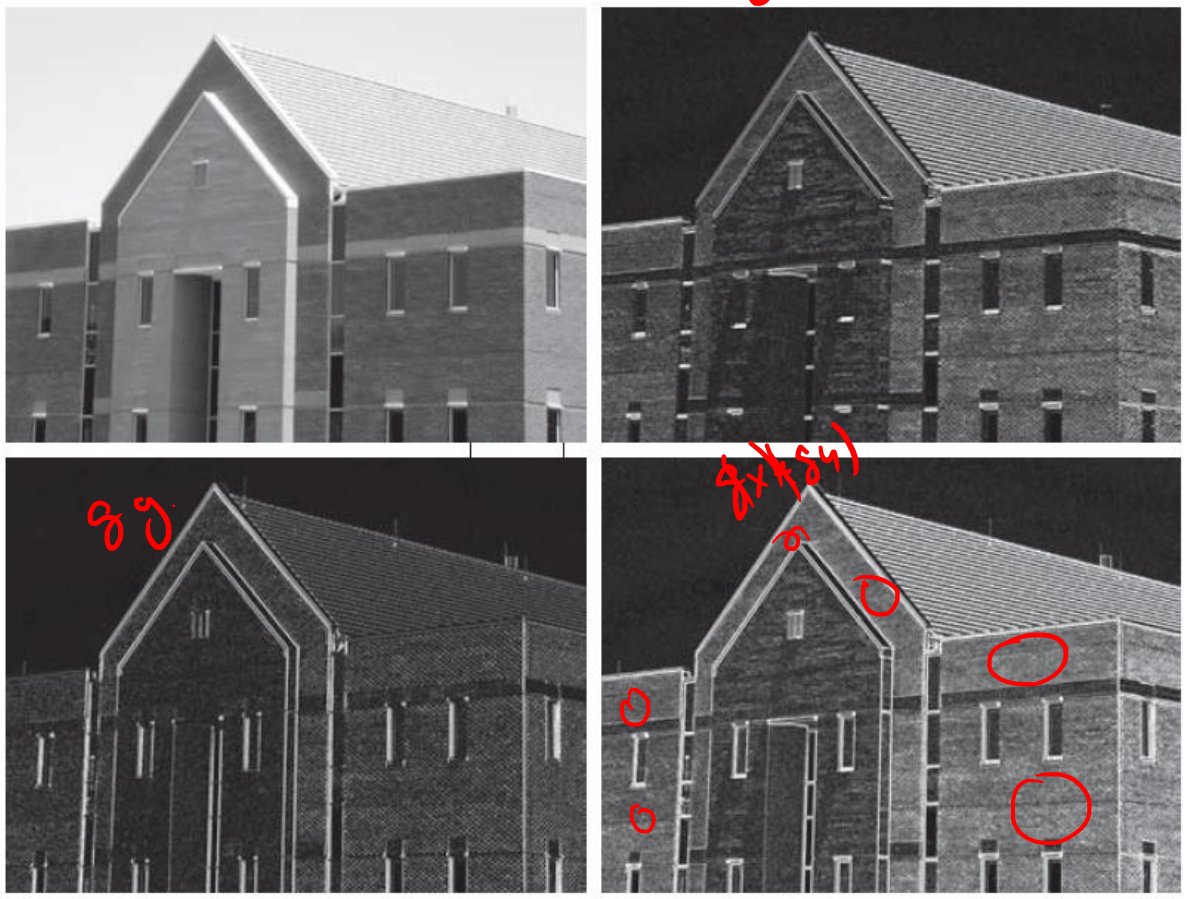
a b
c d

**FIGURE 10.18**
Same sequence as in Fig. 10.16, but with the original image smoothed using a $5 \times 5$ averaging kernel prior to edge detection.



fewer edges in the thresholded image, and that the edges in this image are much sharper

# Sobel on Unsmoothed image



a b
c d

**FIGURE 10.16**
(a) Image of size $834 \times 1114$ pixels, with intensity values scaled to the range [0,1]. (b) $|g_x|$, the component of the gradient in the $x$-direction, obtained using the Sobel kernel in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the kernel in Fig. 10.14(g). (d) The gradient image, $|g_x| + |g_y|$.

- Image Sharpening
- Second Order Derivatives

Last class.

First order derivatives → Edge detection – [discontinuity in intensity]

FOD. $\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y)$ → $\begin{bmatrix} 1 & -1 \end{bmatrix}$ $\begin{bmatrix} -1 & 1 \end{bmatrix}$


$\frac{\partial f}{\partial x} = f(x, y+1) - f(x, y)$ → $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$ or $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$m = \frac{y_2 - y_1}{x_2 - x_1}$

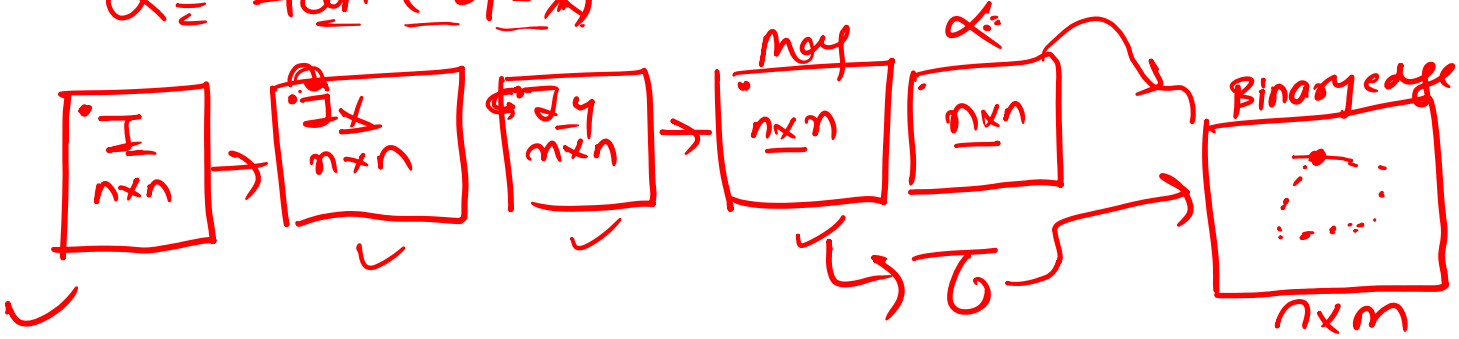
$I_x = I * \partial f / \partial x \qquad I_y = I * \partial f / \partial y$

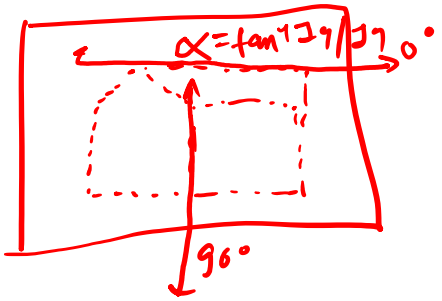
$\tan\theta = m$

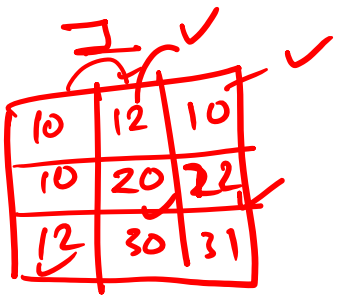
$\theta = \tan^{-1}(m)$

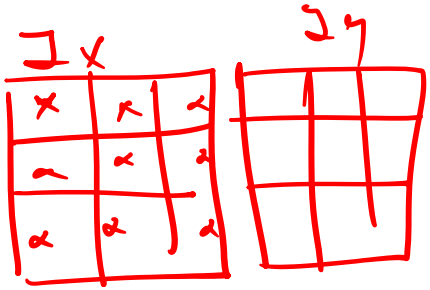
Grad Mag $= \sqrt{I_x^2 + I_y^2}$ or $|I_x| + |I_y|$ { strength of presence of an edge }

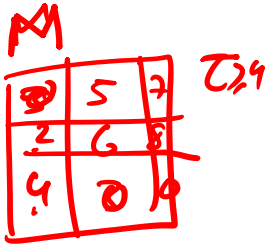
$\alpha = \tan^{-1}(I_y / I_x)$

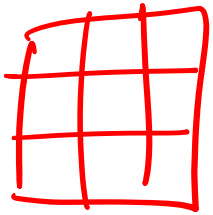


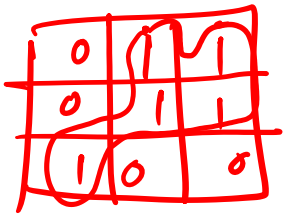


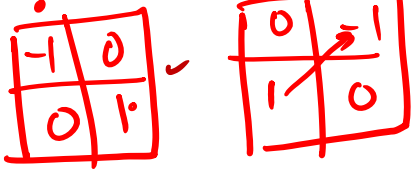





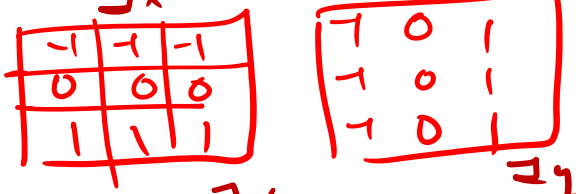

Binary edge map

Various kinds of FOP

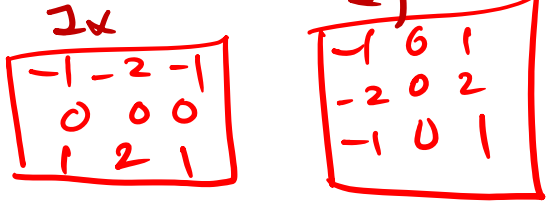
- RC Robert cross operator




(1) Prewitt operator

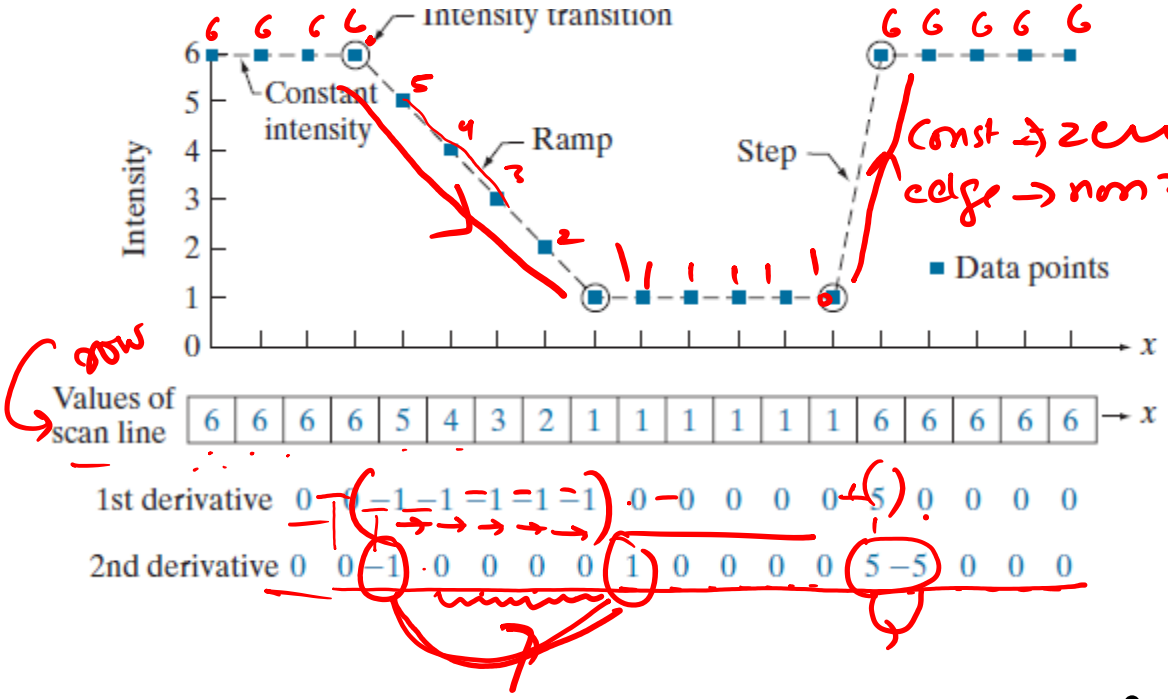


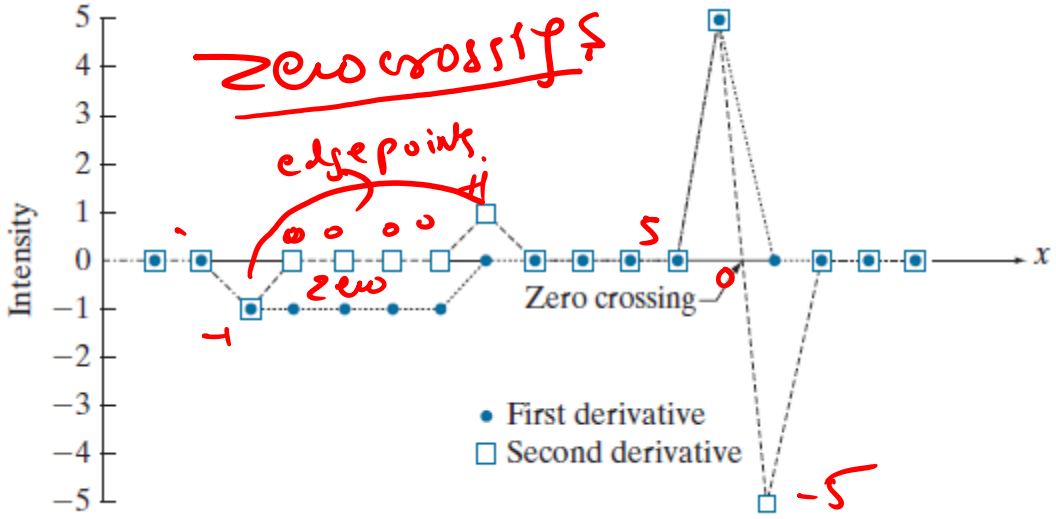

(2) Sobel



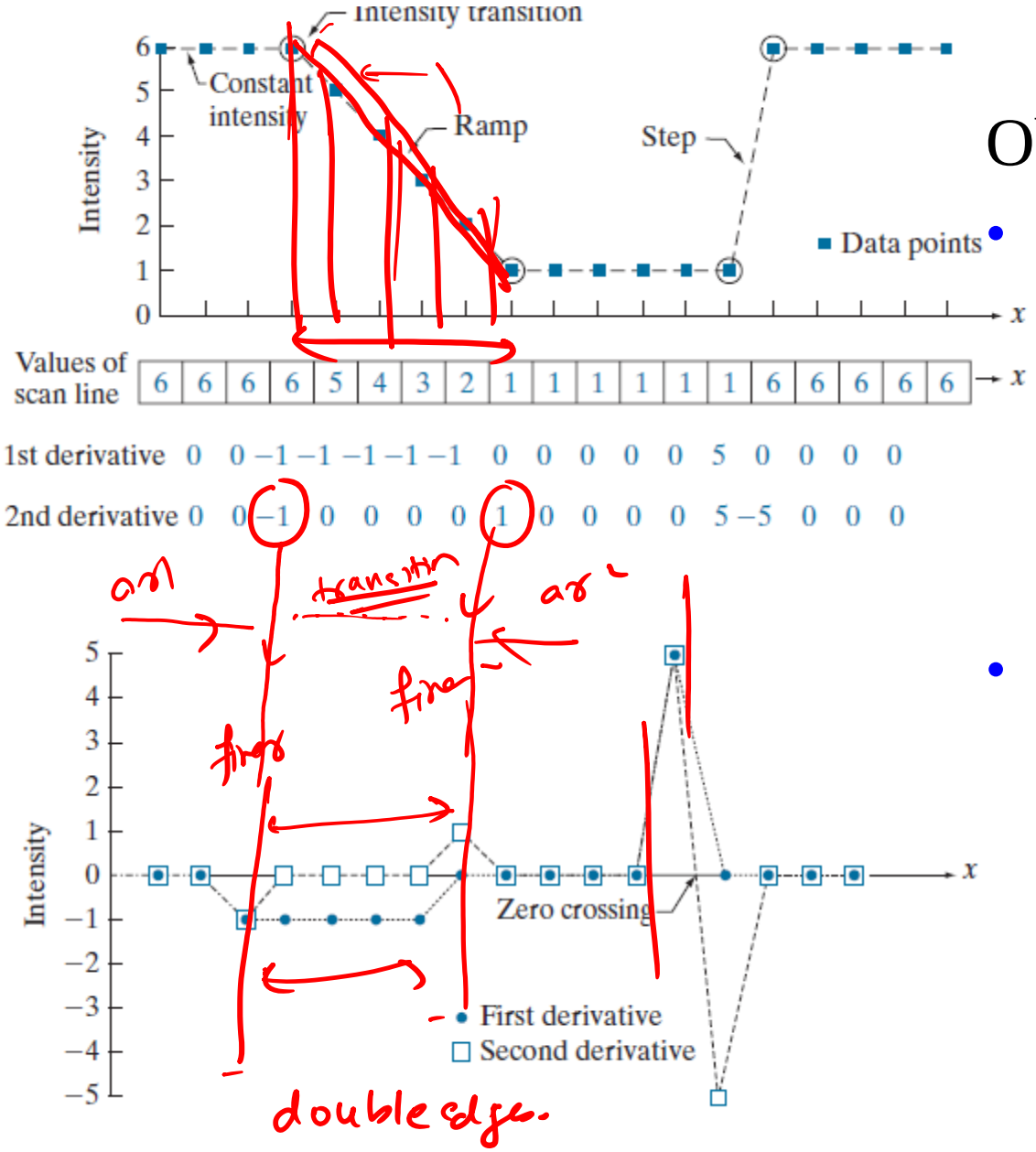

effect of noise on edge detection

- First smooth the image
- Take its FOD. → detect. edge detection.



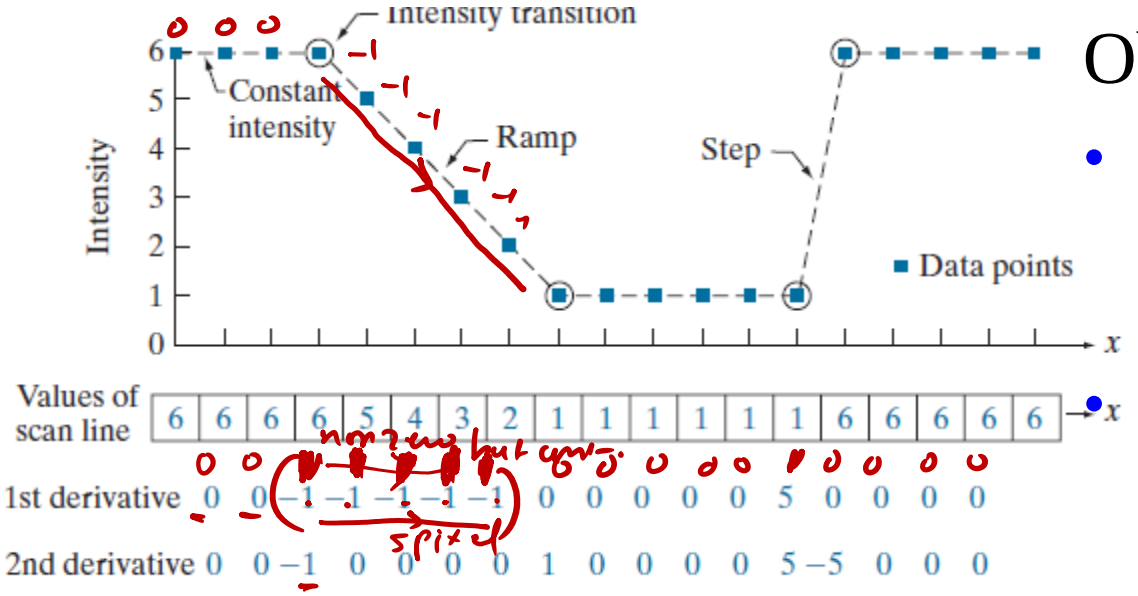



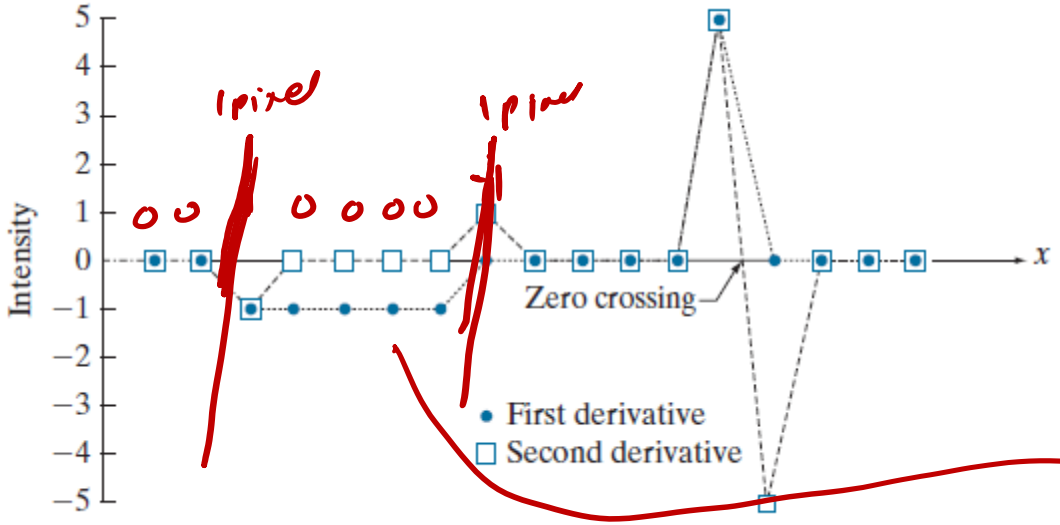

- for a first derivative
  - must be zero in flat segments (areas of constant gray-level values);
  - must be nonzero at the onset of a gray-level step or ramp; and
  - must be nonzero along ramps
- for a second derivative
  - must be zero in flat areas;
  - must be nonzero at the onset & end of a gray-level step or ramp; &
  - must be zero along ramps of constant slope




# Observations:

- First order derivative is nonzero along the entire ramp, while the second order derivative is nonzero only at the onset and end of the ramp
- Because edges in an image resemble this type of transition, we can say that First order derivatives produce "thick" edges and second order derivatives produce finer edges



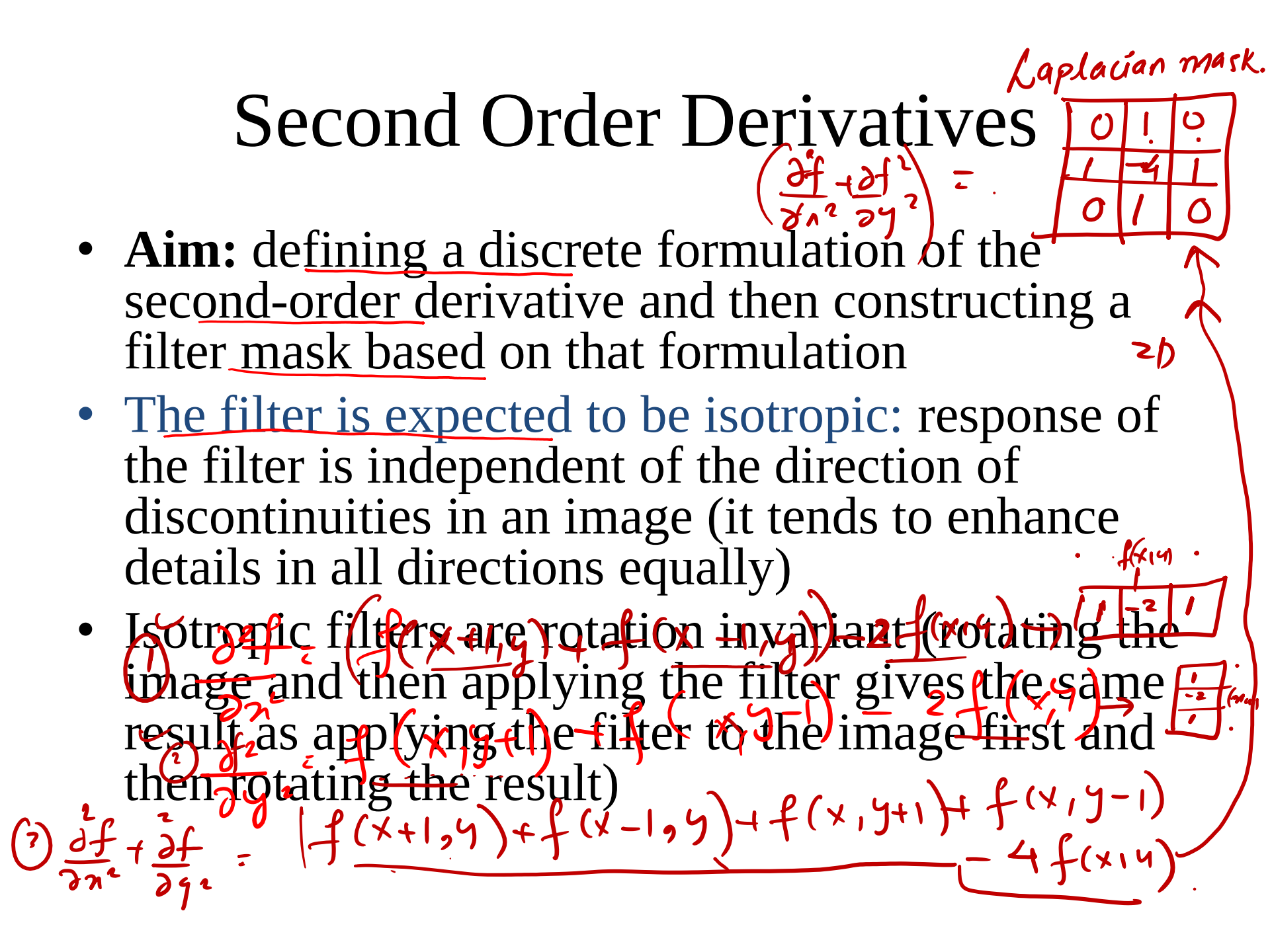




# Observations:

- The response of the two derivatives is the same at the gray-level step
- The sign of the second derivative changes at the onset and end of a step or ramp (zero crossing – useful property for locating edges)
  - The second derivative has a transition from positive back to negative
  - It would produce a double edge one pixel thick, separated by zeros

# Second Order Derivatives

- **Aim:** defining a discrete formulation of the second-order derivative and then constructing a filter mask based on that formulation
- The filter is expected to be isotropic: response of the filter is independent of the direction of discontinuities in an image (it tends to enhance details in all directions equally)
- Isotropic filters are rotation invariant (rotating the image and then applying the filter gives the same result as applying the filter to the image first and then rotating the result)



# Second Order Derivatives

- Laplacian operators

Gradient operator

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

Laplacian operator

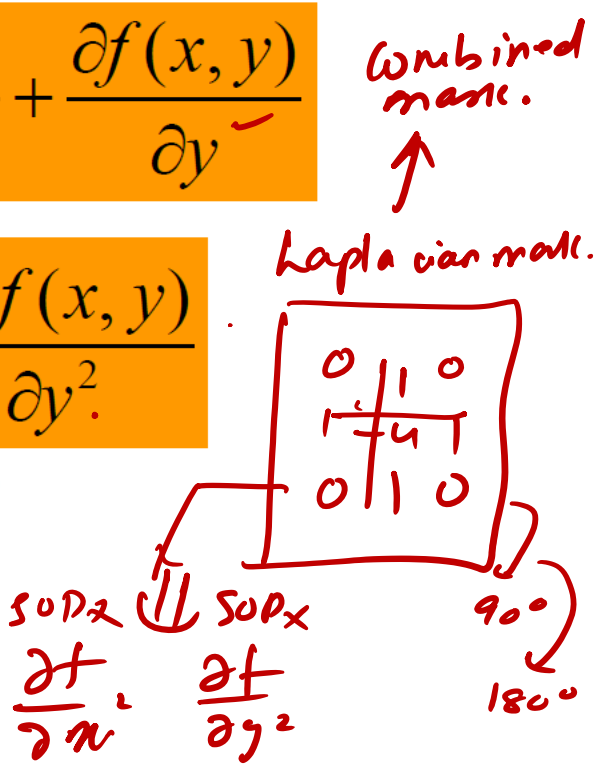
(linear operator)

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

simplest isotropic derivative operator

invariant to rotation

The equation needs to be expressed in discrete form



# Laplacian Masks

from

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

summing the two components yield,

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

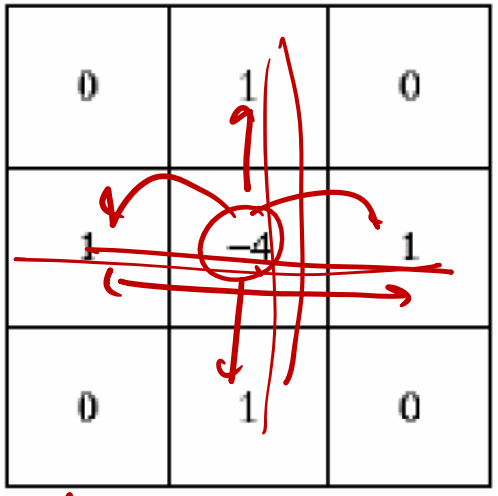
The equation can be implemented as mask

# Laplacian Masks

One mask to detect $SOD_x$ + $SOD_y$. (horiz + vertical edges)

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

Composite




Laplacian mask

This filter ~~mask~~ gives an isotropic result for rotations in increments of $90^0$

Implementation is similar as linear smoothing filters. Only coefficients are different

# Laplacian Masks

(a)

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | −4 | 1 |
| 0 | 1 | 0 |

(b)

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | −8 | 1 |
| 1 | 1 | 1 |

(a) Filter mask used to implement the digital Laplacian as defined in the equation

(b) Mask used to implement an extension of this equation that includes the diagonal neighbors (this mask yields isotropic results for increment of 450)

# Laplacian Masks

h+v

h+v+LD+RD.

(a)

| | | |
|---|---|---|
| 0 | +1 | 0 |
| 1 | −4 | 1 |
| 0 | 1 | 0 |

(b)

| | | |
|---|---|---|
| +1 | +1 | 1 |
| +1 | −8 | 1 |
| 1 | 1 | 1 |

=

=

| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | +4 | −1 |
| 0 | −1 | 0 |

(c)

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | +8 | −1 |
| −1 | −1 | −1 |

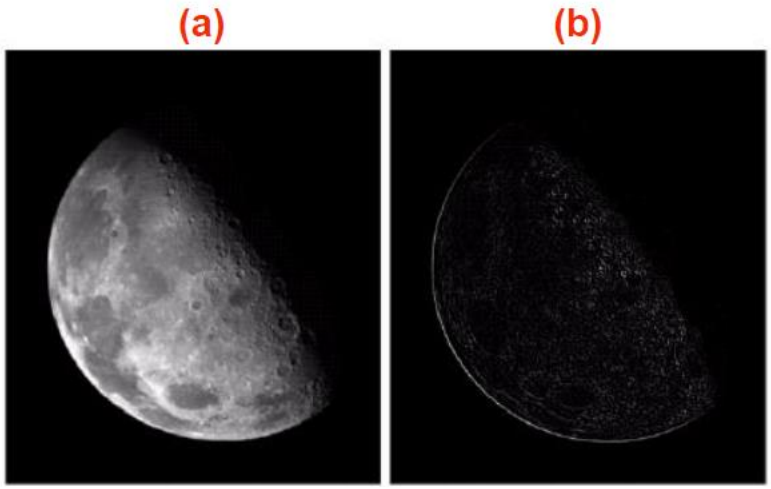
(d)

(a) and (d) two other implementations of the Laplacian

Figure (a) and (c) yield equivalent results, but the difference in sign must be kept in mind when combining (by addition or subtraction) a Laplacian-filtered image with another image

# Effect of Laplacian Operator

- As it is a derivative operator,
  - it highlights gray-level discontinuities in an image
  - it deemphasizes regions with slowly varying gray levels
- Tends to produce images that have
  - grayish edge lines and other discontinuities, all superimposed on a dark, featureless background




(a) Image of the North Pole of the moon
(b) Laplacian filtered image with mask

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

# Use of Laplacian

1. Image Sharpening

2. Edge Detector-
   - LoG

# 1. Sharpening

Adding original image to Laplacian output

$$g(x,y) = f(x,y) + c\nabla^2 f(x,y)$$

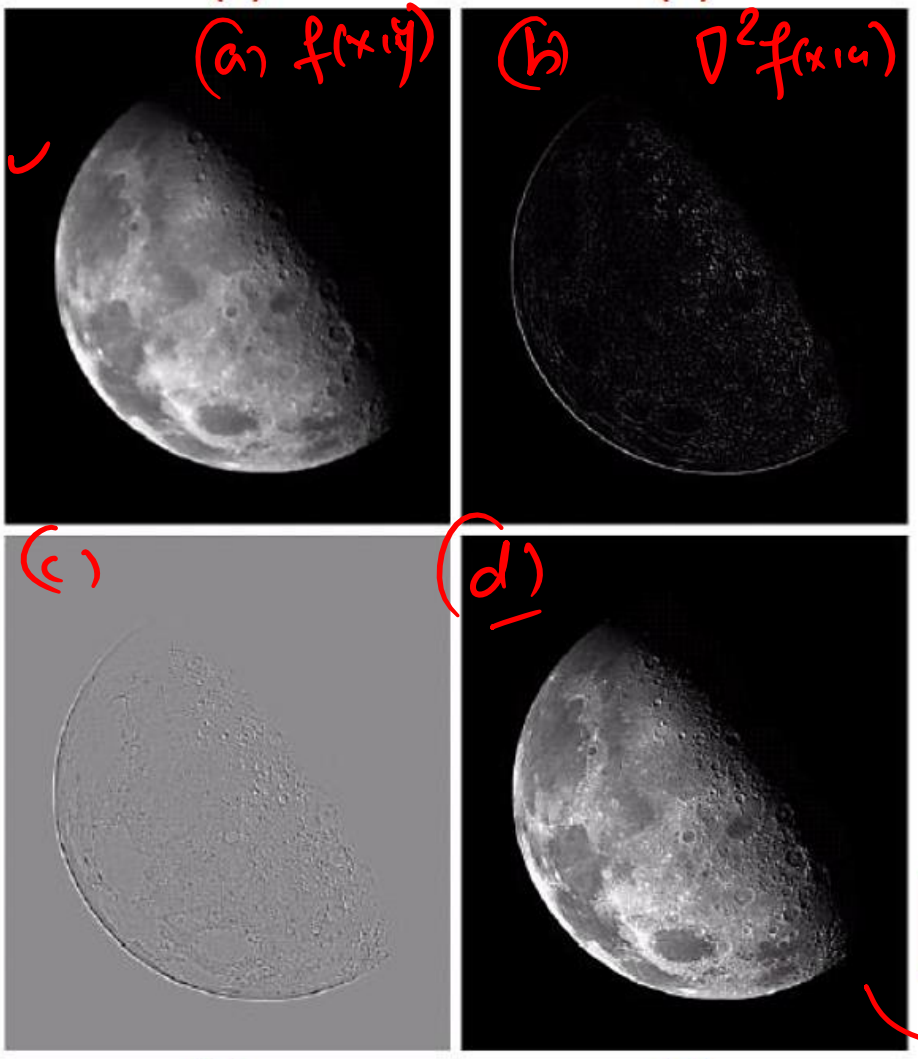
Depending upon the type mask

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) \\ f(x,y) + \nabla^2 f(x,y) \end{cases}$$

if the center coefficient of the Laplacian mask is negative

if the center coefficient of the Laplacian mask is positive

# 1. Sharpening




(a) Image of the North Pole of the moon

(b) Laplacian filtered image with mask

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

(c) Laplacian image scaled for display purposes

(d) Image enhanced by using the equation

$$g(x,y) = f(x,y) - \nabla^2 f(x,y)$$

How to write a composite sharpening mask.

$$g(x,y) = f(x,y) + c \cdot \nabla^2 f(x,y)$$

$$= f(x,y) - \nabla^2 f(x,y)$$

$$= f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)]$$

$$= f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1) + 4f(x,y)$$

$$= 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)$$

$$g(x,y) = [I * (M)]$$ composite mask.

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

(M)

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

$$g(x,y) = f(x,y) + c\nabla^2 \cdot f(x,y)$$

$$= f(x,y) - \nabla^2 f(x,y)$$

$$\nabla^2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$= f(x,y) - \big[ f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) + f(x-1,y-1) + f(x+1,y+1) + f(x+1,y-1) + f(x-1,y+1) - 8f(x,y) \big]$$

$$= 9f(x,y) - [f(x+1,y) \ldots -]$$

$$= \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$\Rightarrow$ composite $\rightarrow$ I * CM.

# 1. Sharpening

- To simplify the computation, we can create a mask which do both operations, Laplacian Filter and Addition the original image

$$
\begin{aligned}
g(x,y) &= f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\quad + f(x,y+1) + f(x,y-1) - 4f(x,y)] \\
&= 5f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\quad + f(x,y+1) + f(x,y-1)]
\end{aligned}
$$

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

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

# 1. Sharpening

- Note

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y) \\ f(x,y)+\nabla^2 f(x,y) \end{cases}$$

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -1 | 0 | = | 0 | 0 | 0 | + | 0 | -1 | 0 | | | | |
| -1 | 5 | -1 | | 0 | 1 | 0 | | -1 | 4 | -1 | | | | |
| 0 | -1 | 0 | | 0 | 0 | 0 | | 0 | -1 | 0 | | | | |
| 0 | -1 | 0 | = | 0 | 0 | 0 | + | 0 | -1 | 0 | | | | |
| -1 | 9 | -1 | | 0 | 1 | 0 | | -1 | 8 | -1 | | | | |
| 0 | -1 | 0 | | 0 | 0 | 0 | | 0 | -1 | 0 | | | | |
| 0 | -1 | 0 | = | 0 | 0 | 0 | - | 0 | 1 | 0 | | | | |
| -1 | 5 | -1 | | 0 | 1 | 0 | | 1 | -4 | 1 | | | | |
| 0 | -1 | 0 | | 0 | 0 | 0 | | 0 | 1 | 0 | | | | |

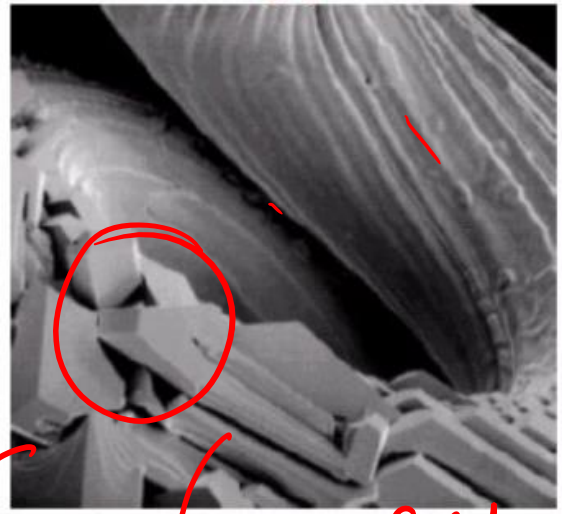
# 1. Sharpening

(a)

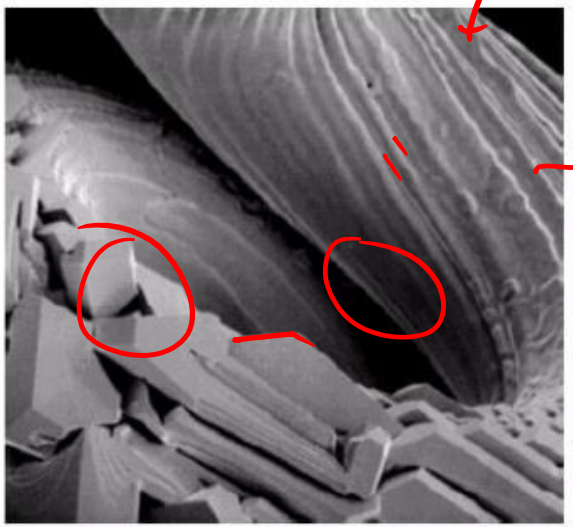
| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | 5 | −1 |
| 0 | −1 | 0 |

(b)

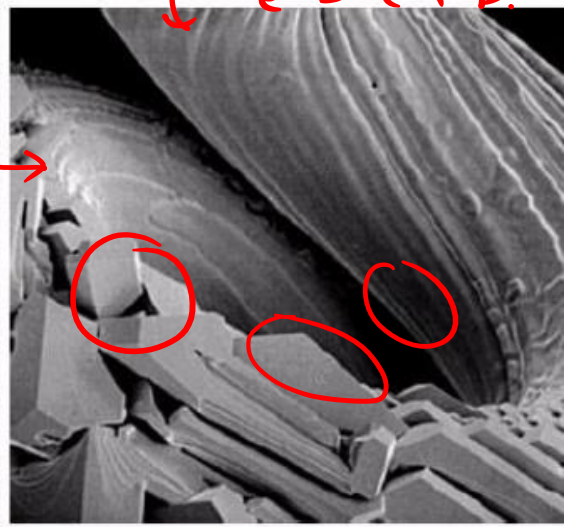
| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 9 | −1 |
| −1 | −1 | −1 |



(c)



(d)



(e)

(a) Composite Laplacian mask
(b)A second composite mask
(c) Scanning electron microscope image
(d)Result of filtering with the mask in (a)
(e) Result of filtering with the mask in (b)
Note how much sharper (e) is than (d)

# 2. Edge Detection

- **The Marr-Hildreth Edge Detector**
- Marr and Hildreth proposed a Gaussian Filter, combined with the Laplacian for edge detection.
- It is called the Laplacian of Gaussian (LoG) $\nabla^2 G$

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

# 2. Edge Detection

Way-1 – 2 step

- The Marr-Hildreth edge-detection algorithm may be summarized as follows:
- **1.** Filter the input image with an $n \times n$ Gaussian lowpass kernel obtained by sampling

smoothed.

- **2.** Compute the Laplacian of the image resulting from Step 1 using.

- **3.** Find the zero crossings of the image from Step 2.

# 2. Edge Detection

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

$$= \frac{\partial}{\partial x}\left(\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right) + \frac{\partial}{\partial y}\left(\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right)$$

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

Collecting terms, we obtain

$$\nabla^2 G(x,y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

This expression is called the *Laplacian of a Gaussian* (LoG).

# 2. Edge Detection

The Marr-Hildreth algorithm consists of convolving the LoG kernel with an input image,

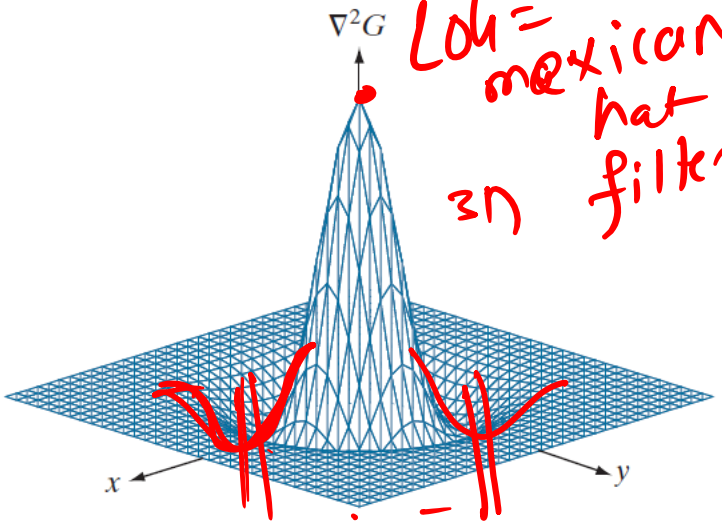
$$g(x, y) = \left[\nabla^2 G(x, y)\right] \star f(x, y) \qquad (10\text{-}30)$$

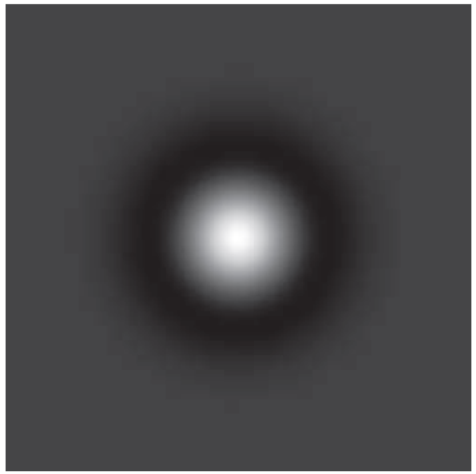
and then finding the zero crossings of $g(x, y)$ to determine the locations of edges in

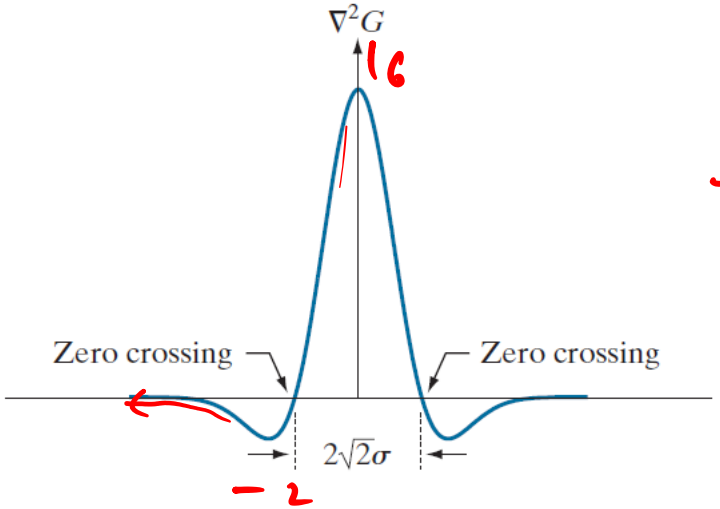
# 2. Edge Detection

a b
c d

**FIGURE 10.21**
(a) 3-D plot of the *negative* of the LoG.
(b) Negative of the LoG displayed as an image.
(c) Cross section of (a) showing zero crossings.
(d) $5 \times 5$ kernel approximation to the shape in (a). The negative of this kernel would be used in practice.






| | | | | |
|---|---|---|---|---|
| 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 |

# Find zero crossings

- One approach for finding the zero crossings at any pixel, p, of the filtered image, g(x ,y ), is to use a 3 3 × neighborhood centered at p.
- A zero crossing at p implies that the signs of at least two of its opposing neighboring pixels must differ. There are four cases to test: left/right, up/down, and the two diagonals.
- If the values of g(x,y) are being compared against a threshold (a common approach), then not only must the signs of opposing neighbors be different, but the absolute value of their numerical difference must also exceed the threshold before we can call p a zero-crossing pixel.

G&W

a b
c d

**FIGURE 10.22**
(a) Image of size $834 \times 1114$ pixels, with intensity values scaled to the range [0, 1]. (b) Result of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.