

SHARPENING FILTERS

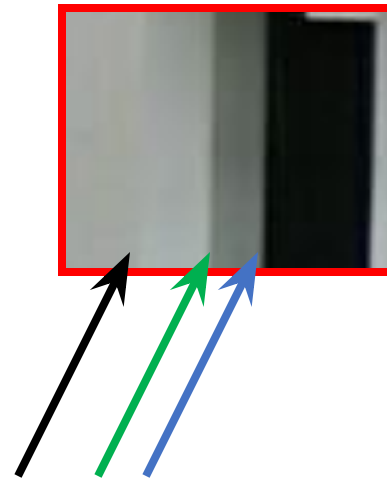
Sharpening Spatial Filters

- 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**

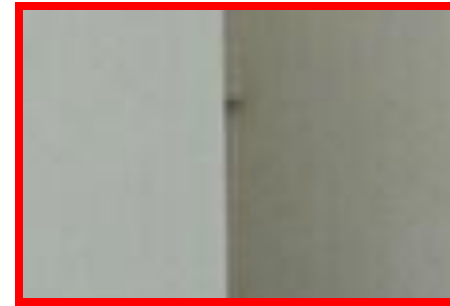
Closeup of edges



Closeup of edges



Closeup of edges

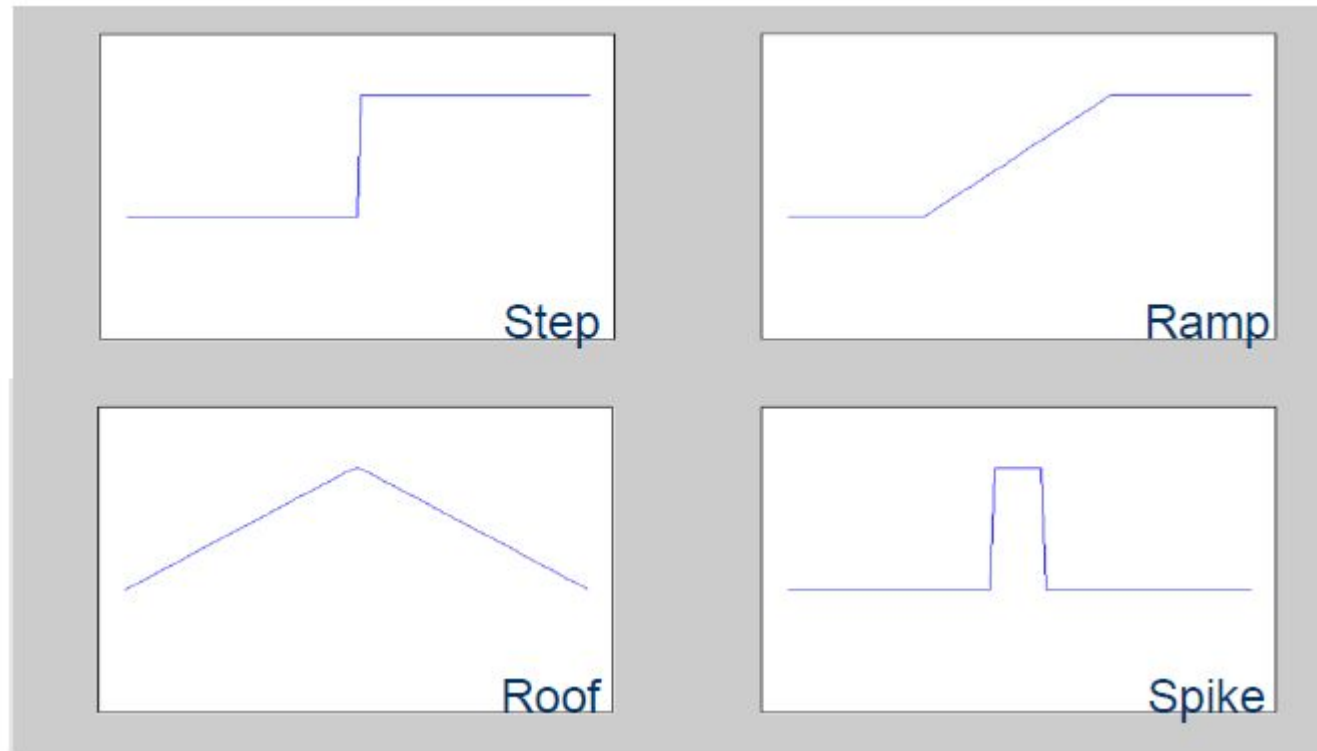


Closeup of edges



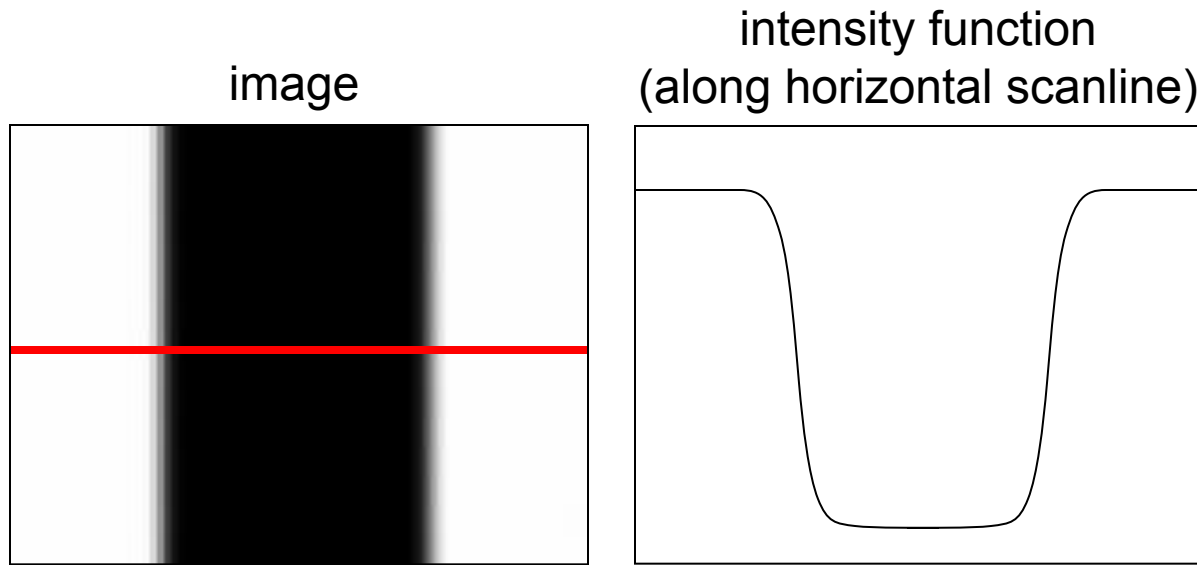
Edge

- Discontinuity of intensities in the image



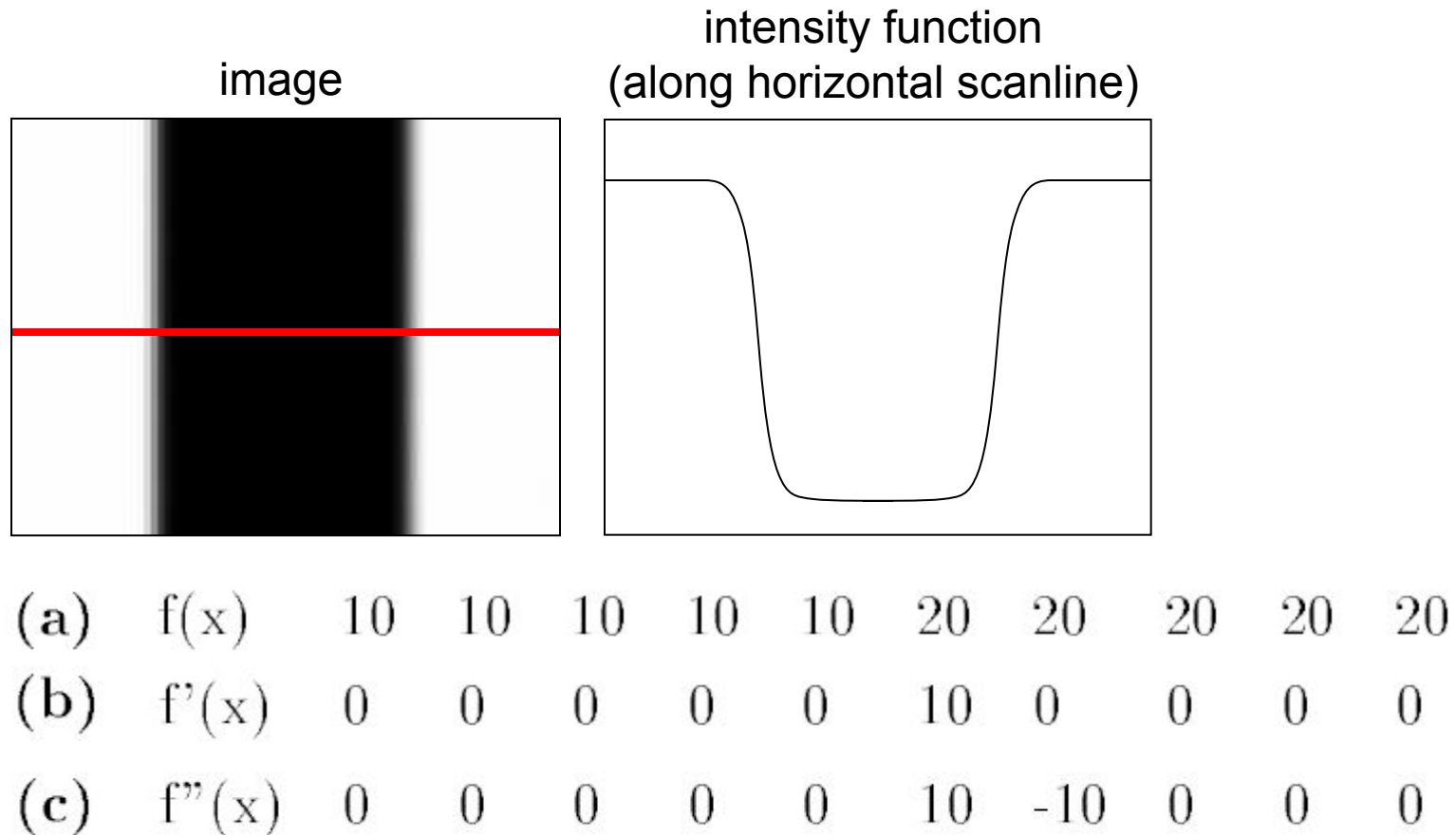
Characterizing edges

- place of rapid change in the image intensity function



Characterizing edges

- place of rapid change in the image intensity function



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

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

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

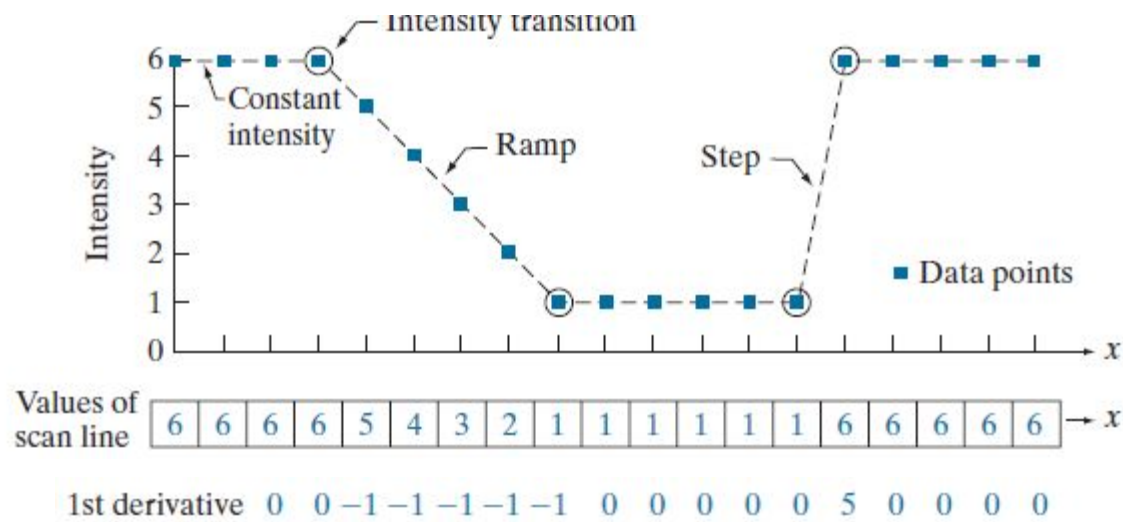
- Derivatives of a digital function are defined in terms of differences
- Various ways to define these differences

Derivative Operator

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

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

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



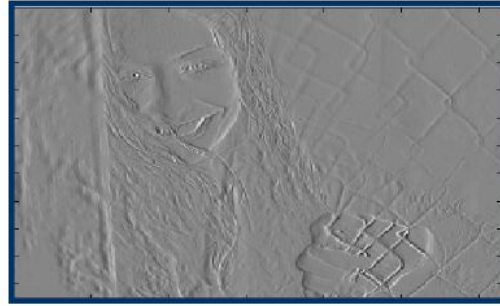
• for a first derivative

- 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

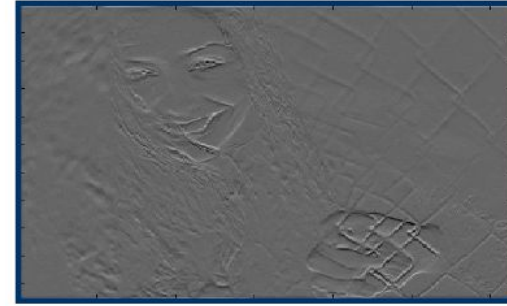
First Order Image Derivatives



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)**

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

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

$$\alpha(x, y) = \tan^{-1}\left(\frac{g_y}{g_x}\right) \quad \text{angle is measured wrt the X-axis}$$

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)

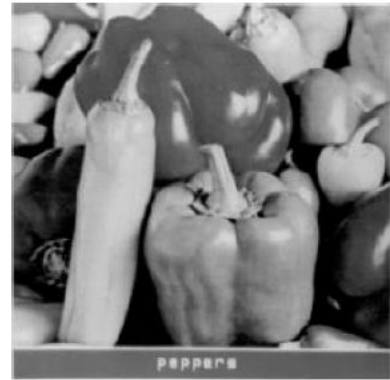
$$M(x, y) = \text{mag}(\nabla f) = [g_x^2 + g_y^2]^{1/2} \\ = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2}$$

- $M(x, y)$ is an image of the same size as the original, created when x and y are allowed to vary over all pixel locations in $f \rightarrow$ “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

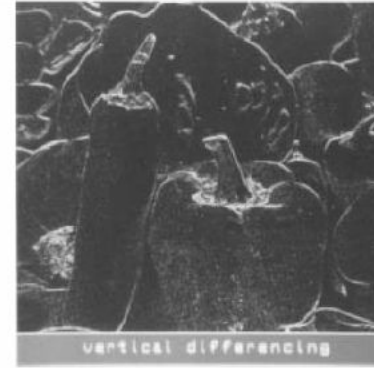
- Motivation: detect changes change in the pixel value \longrightarrow large gradient



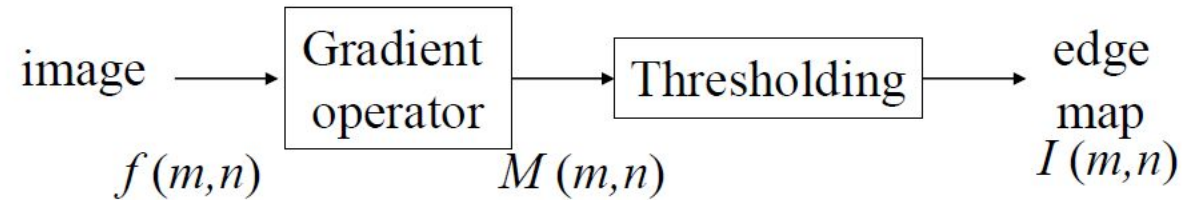
(a) Original



(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

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

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

-1
1

-1	1
----	---

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

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

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

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)

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

- $g_x : (z_9 - z_5), \quad g_y : (z_8 - z_6)$
- Gradient image can be computed as:

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

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

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

- 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	-1	0	1
0	0	0	-1	0	1
1	1	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	-1	0	1
0	0	0	-2	0	2
1	2	1	-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 834×1114 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

FIGURE 10.17

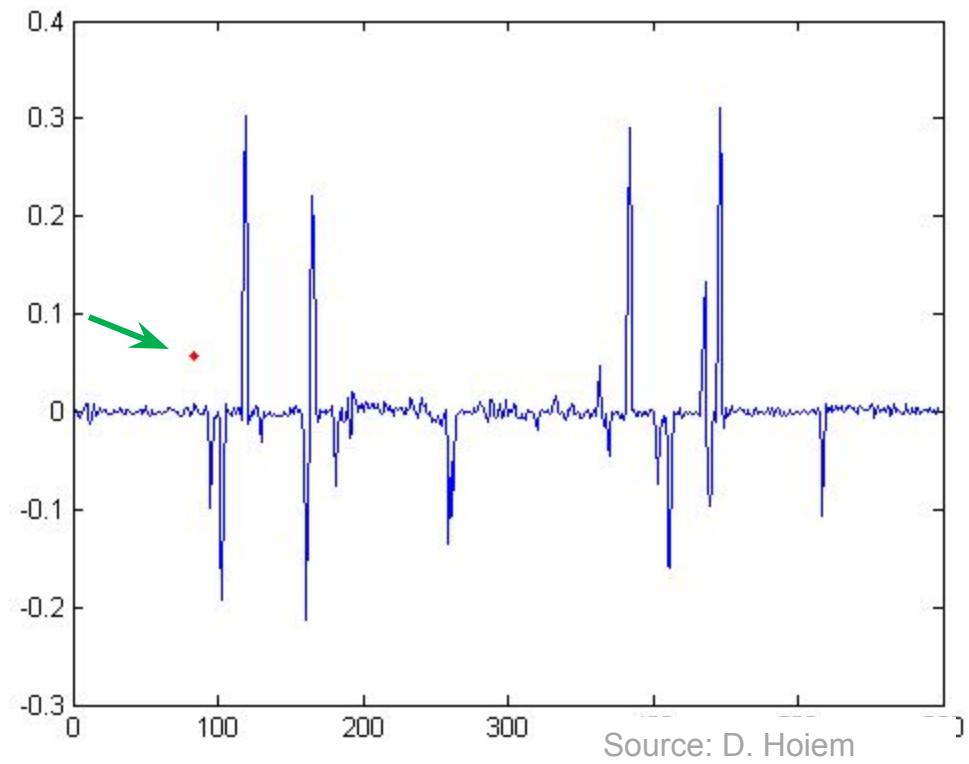
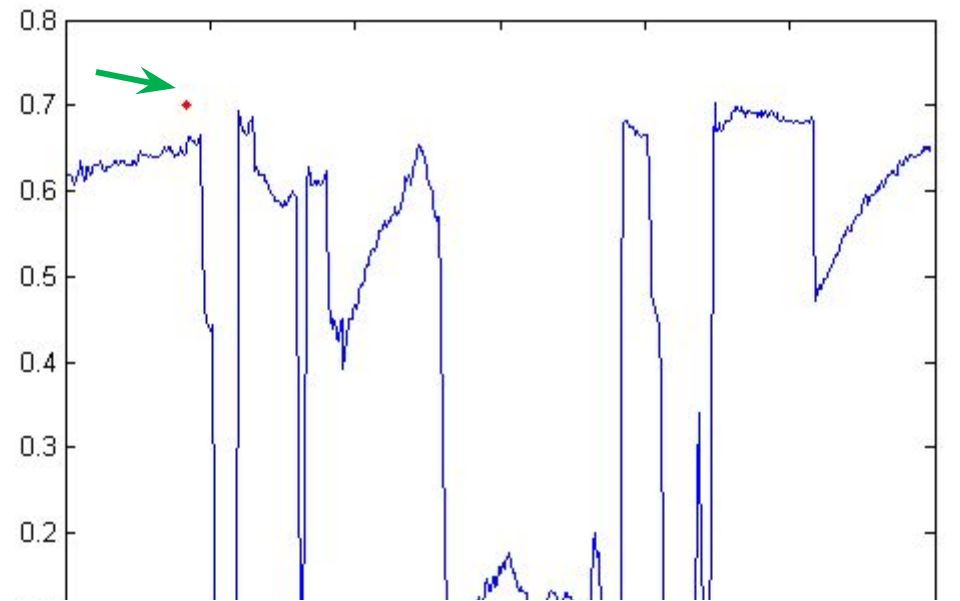
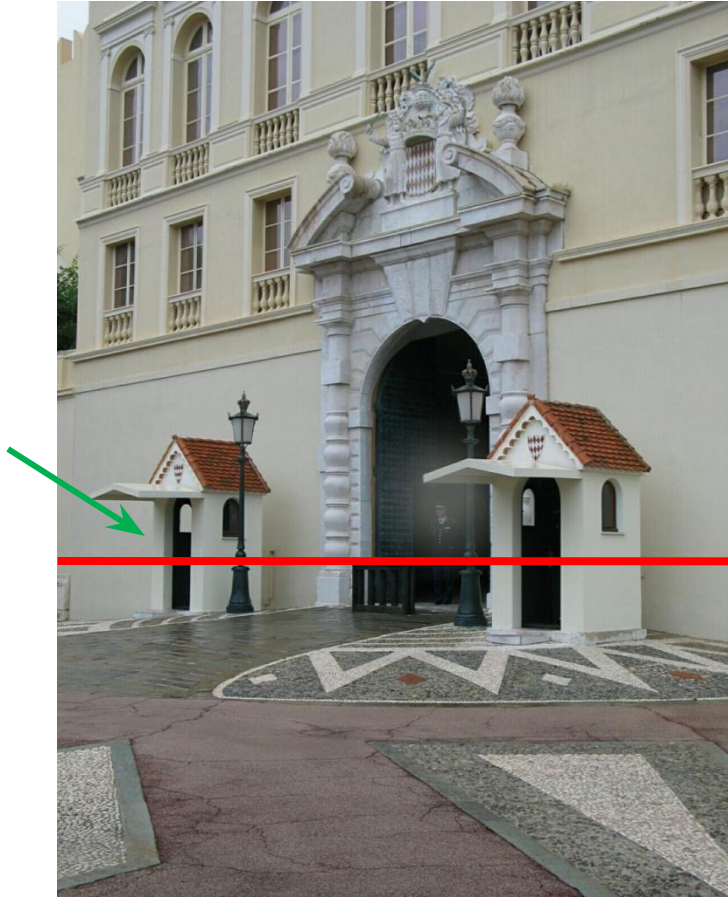
Gradient angle image computed using Eq. (10-18). Areas of constant intensity in this image indicate that the direction of the gradient vector is the same at all the pixel locations in those regions.



Gradient Masks

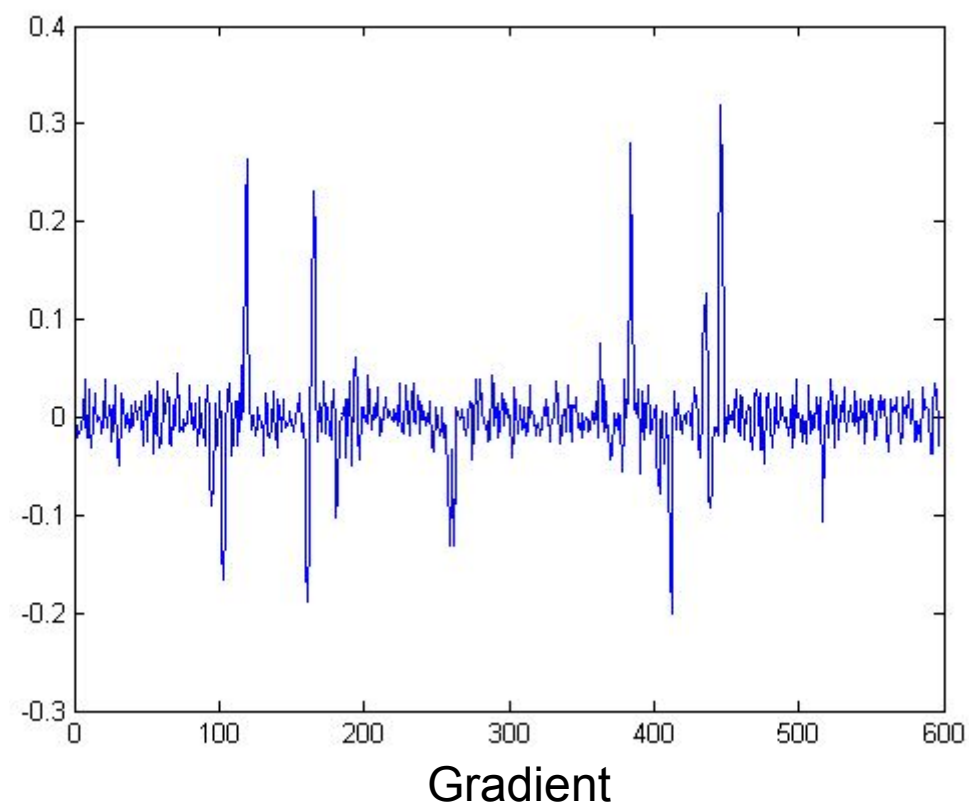
- Prewitt and Sobel operators: mostly used in practice for computing digital gradients.
- Prewitt masks: simpler to implement.
- Sobel masks have slightly superior noise-suppression characteristics
 - an important issue when dealing with derivatives
 - Reason: a weight value of 2 is used to achieve some 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

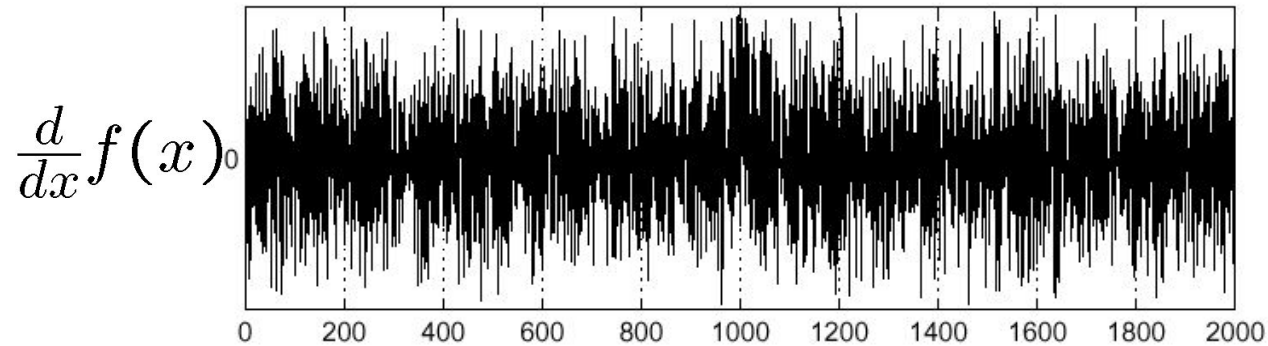
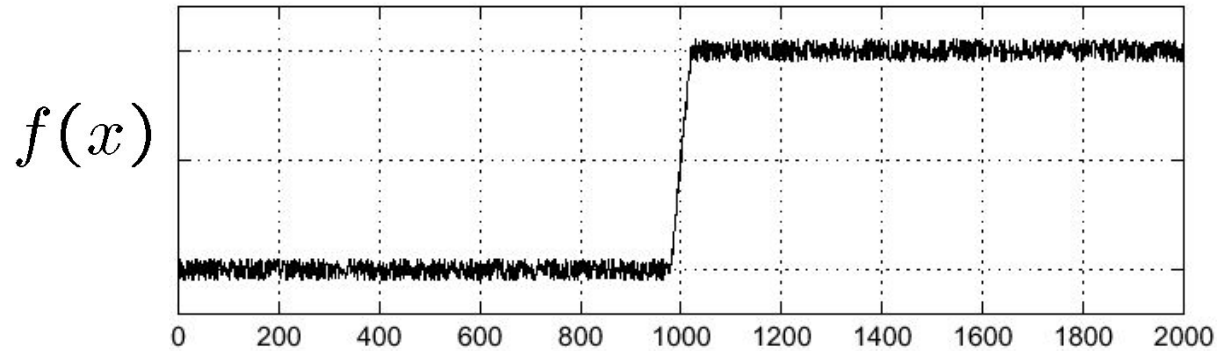


Source: D. Hoiem

Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

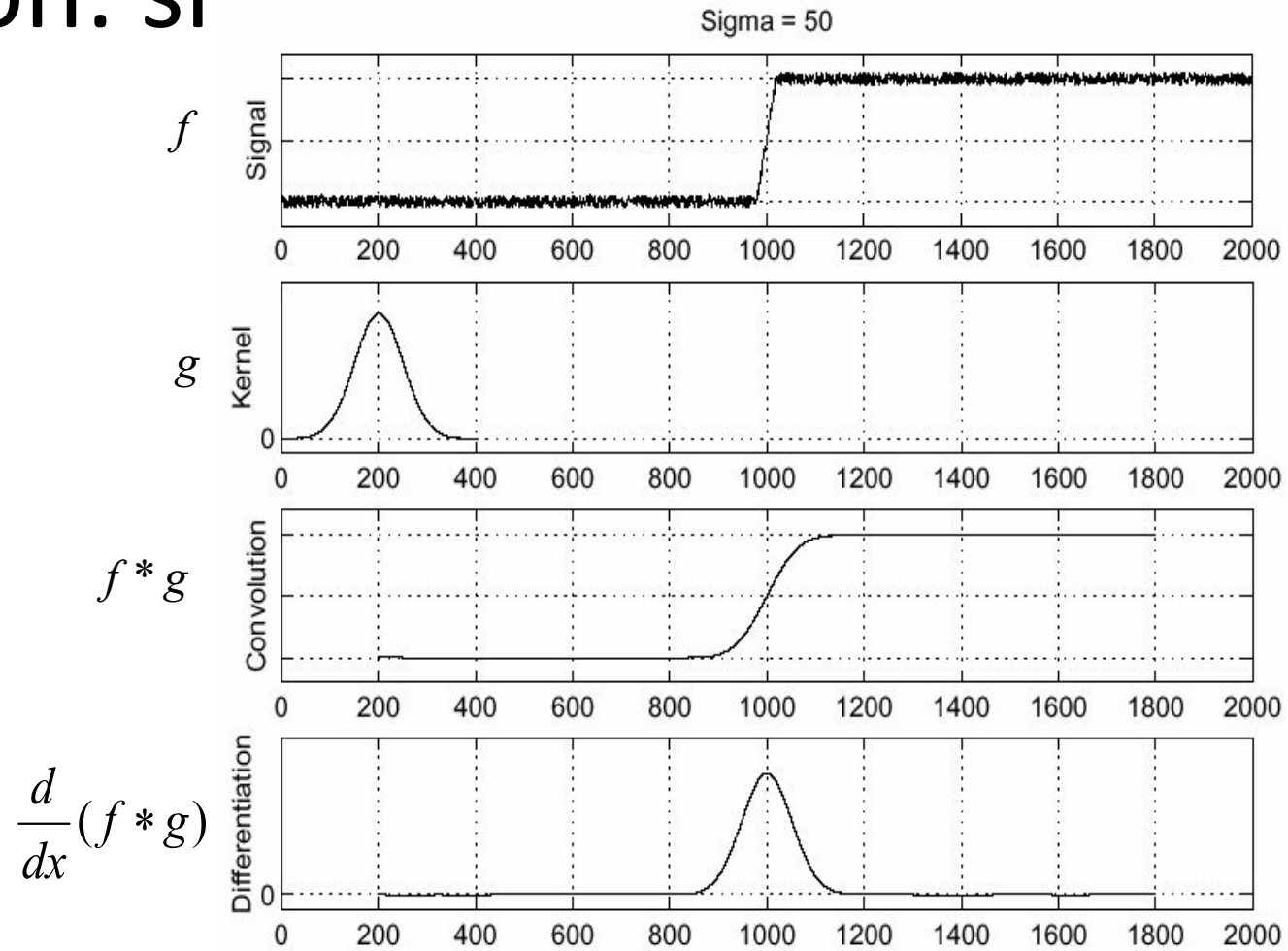


Where is the edge?

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?

Solution: smooth first



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

Sobel on Smoothed image

a	b
c	d

FIGURE 10.18

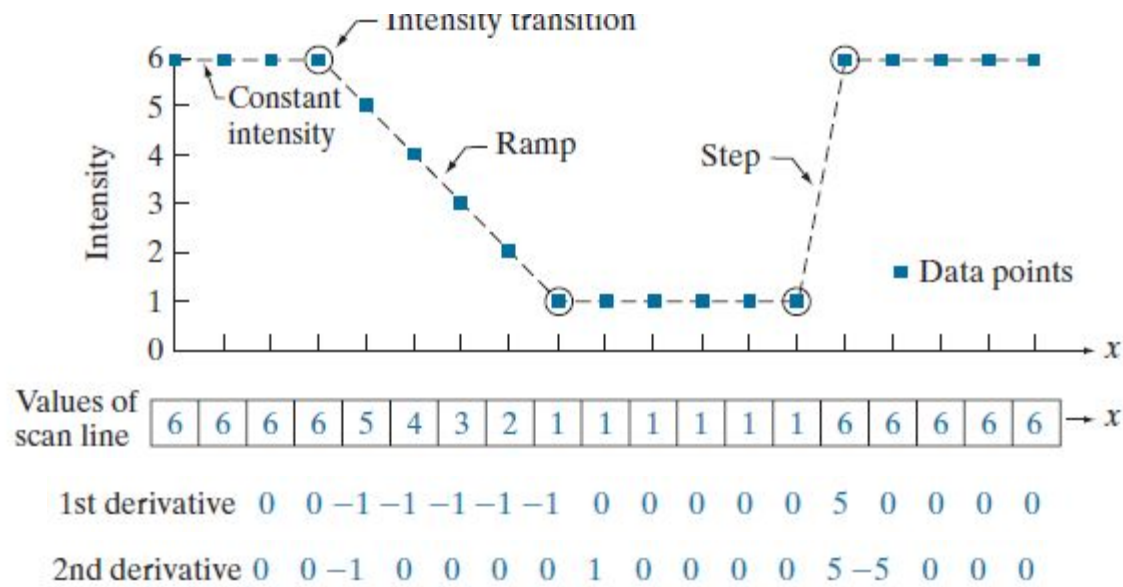
Same sequence as in Fig. 10.16, but with the original image smoothed using a 5×5 averaging kernel prior to edge detection.



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

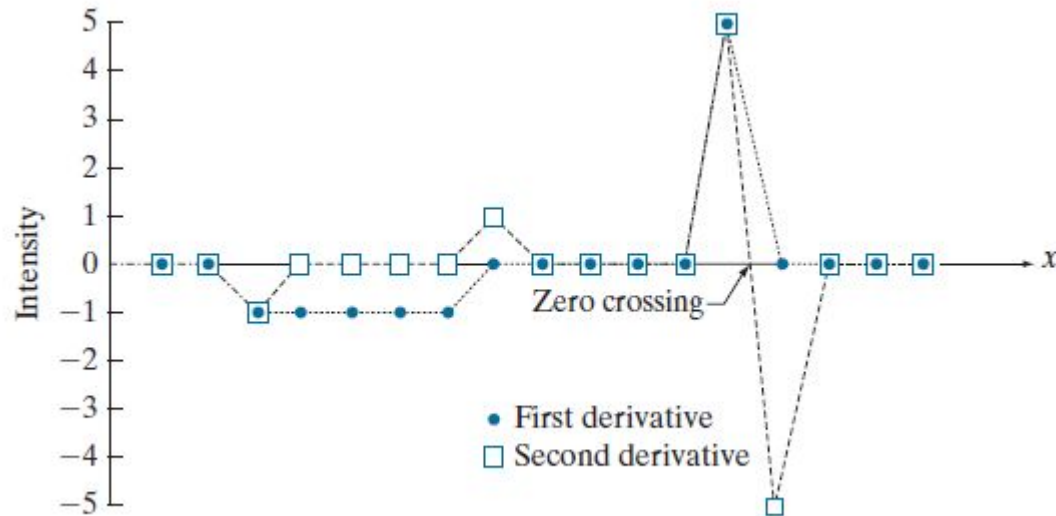
Image Sharpening

Second Order Derivatives



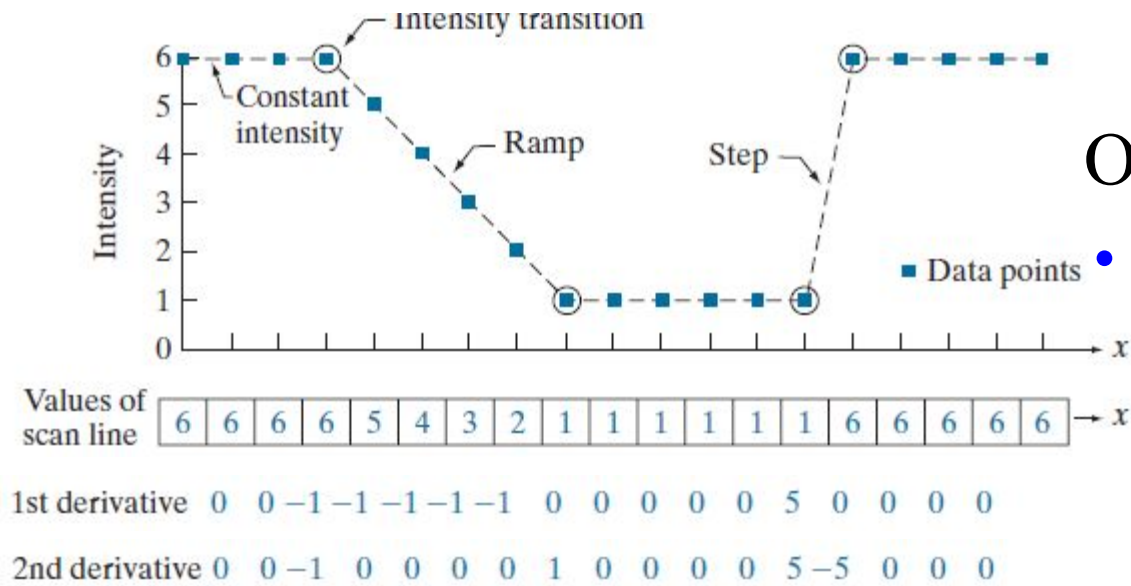
- for a first derivative

- 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



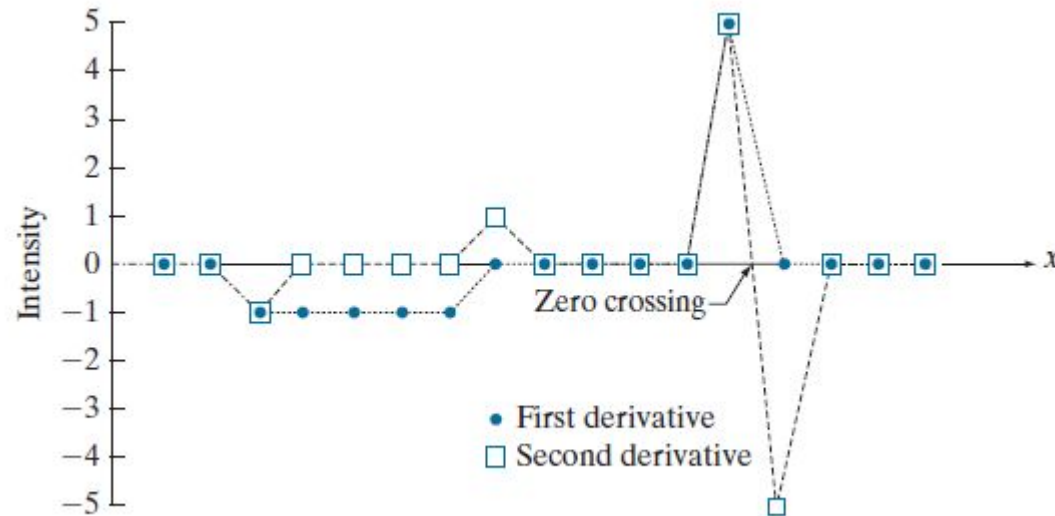
- for a second derivative

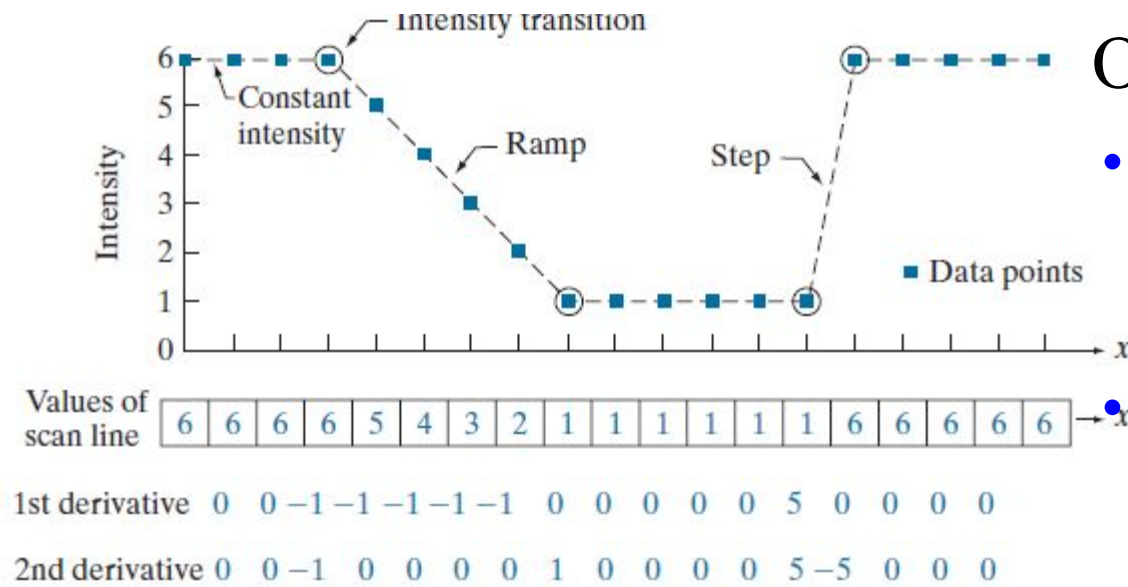
- must be zero in flat areas;
- must be nonzero at the onset & end of a gray-level step or ramp; &
- must be zero along ramps of constant slope



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



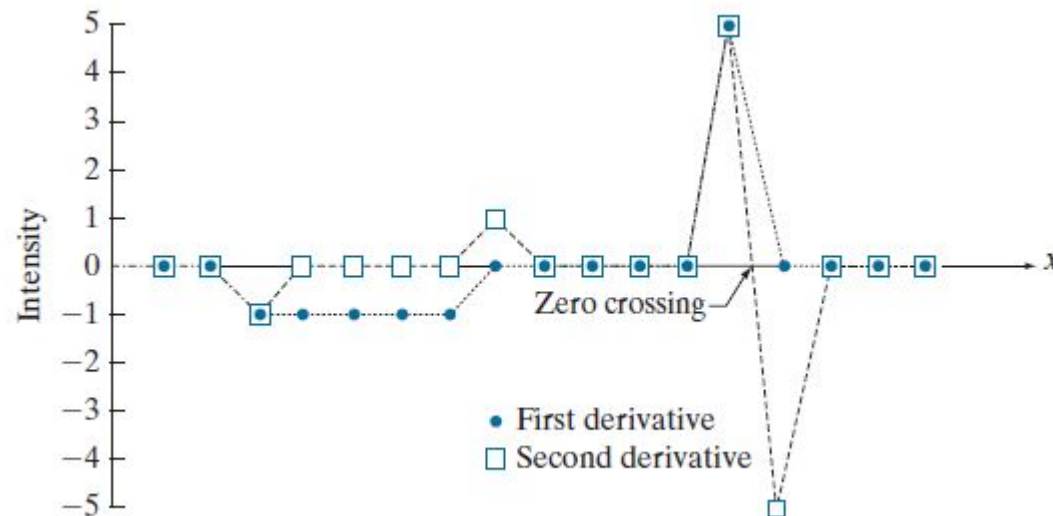


Observations:

- 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

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

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

0	1	0
1	-4	1
0	1	0

This filter mask gives an isotropic result for rotations in increments of 90°

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

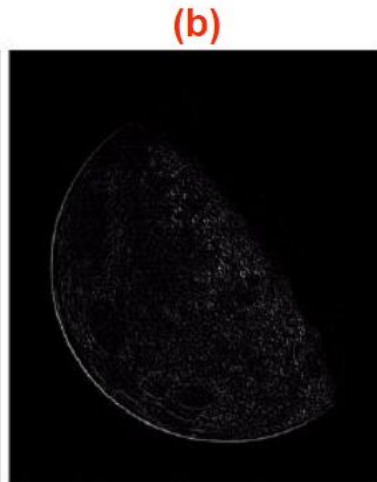
(a)			(b)		
0	1	0	1	1	1
1	-4	1	1	-8	1
0	1	0	1	1	1
(c)			(d)		
0	-1	0	-1	-1	-1
-1	4	-1	-1	8	-1
0	-1	0	-1	-1	-1

(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

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

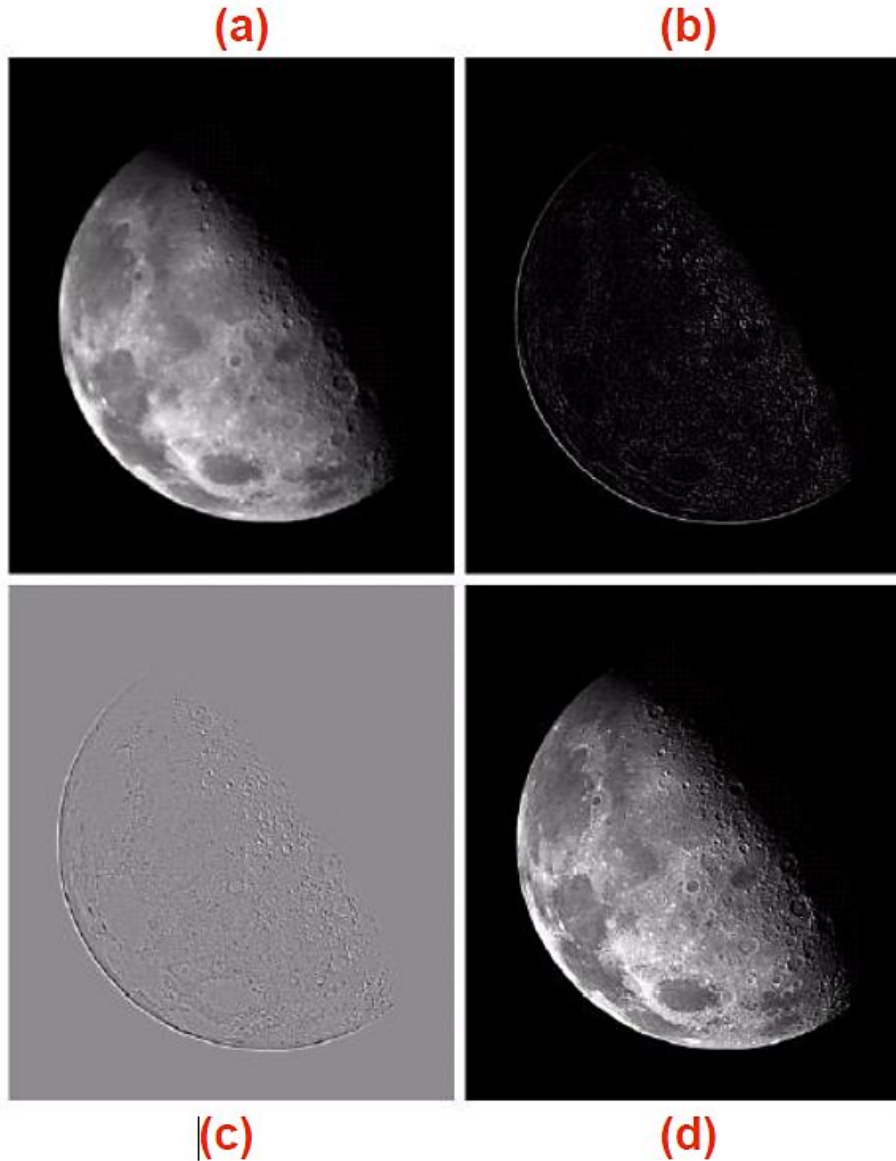
Depending upon the type mask

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

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

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

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
-1	5	-1
0	-1	0

=

0	0	0
0	1	0
0	0	0

+

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

0	-1	0
-1	9	-1
0	-1	0

=

0	0	0
0	1	0
0	0	0

+

0	-1	0
-1	8	-1
0	-1	0

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

=

0	0	0
0	1	0
0	0	0

-

0	1	0
1	-4	1
0	1	0

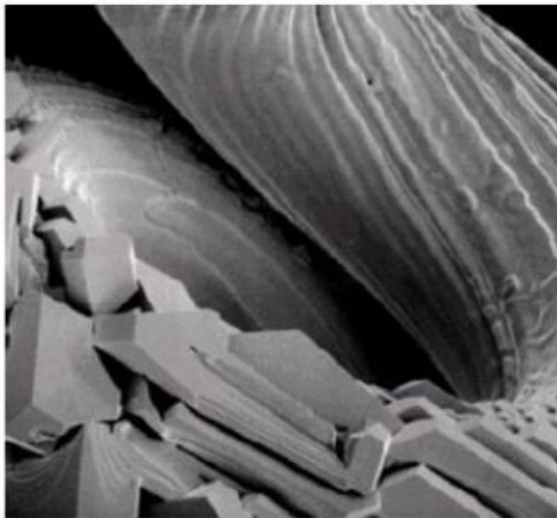
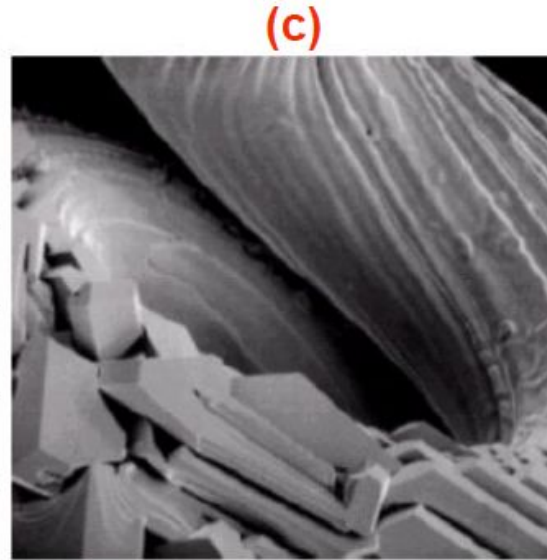
1. Sharpening

(a)

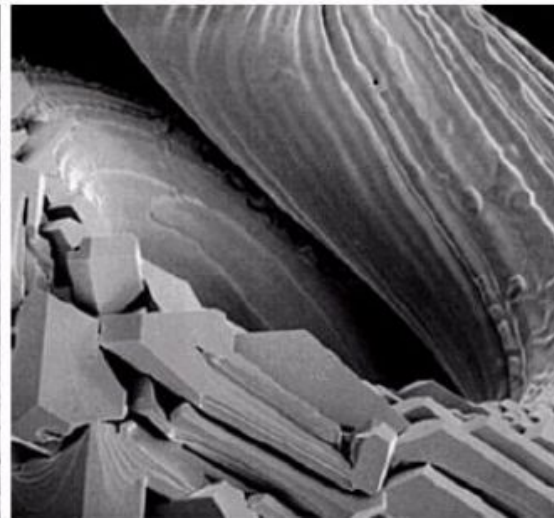
0	-1	0
-1	5	-1
0	-1	0

(b)

-1	-1	-1
-1	9	-1
-1	-1	-1



(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

- 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
- **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

$$\begin{aligned}\nabla^2 G(x, y) &= \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} \\ &= \frac{\partial}{\partial x} \left(\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right) + \frac{\partial}{\partial y} \left(\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right) \\ &= \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}} + \left(\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}\end{aligned}$$

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,

$$g(x, y) = [\nabla^2 G(x, y)] \star f(x, y) \quad (10-30)$$

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

2. Edge Detection

a b
c d

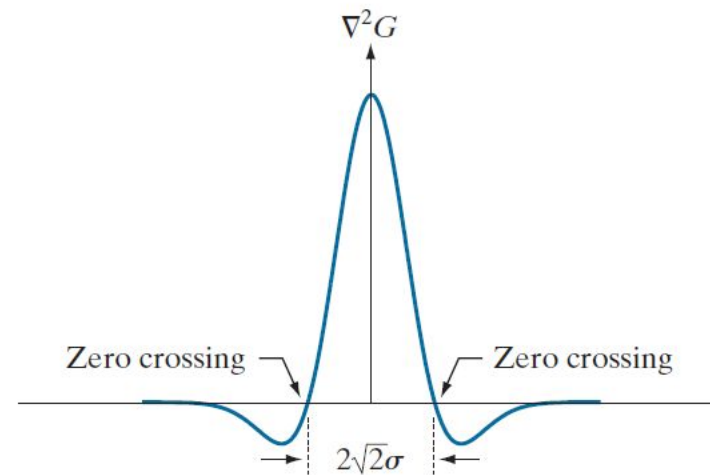
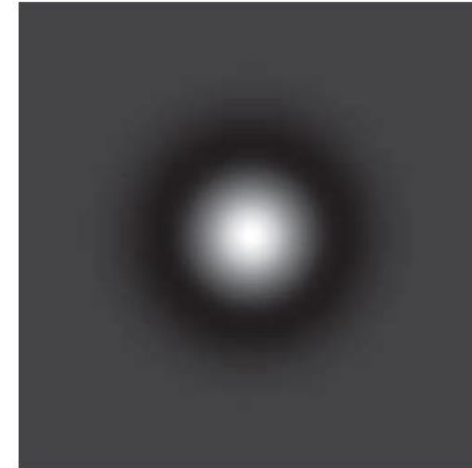
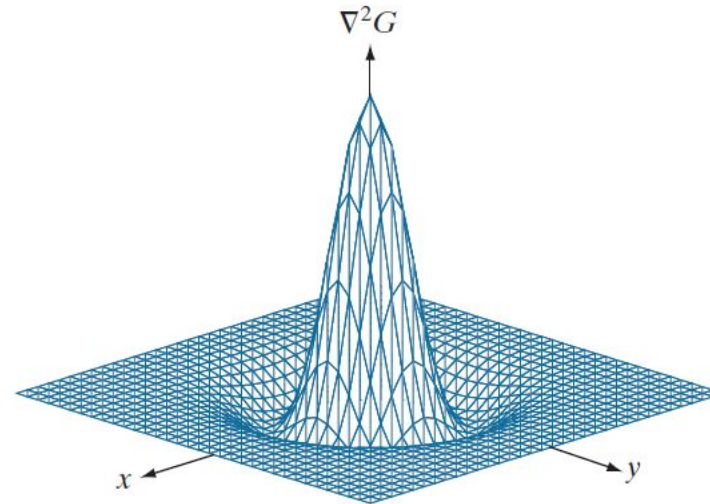
FIGURE 10.21

(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) 5×5 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.

a	b
c	d

FIGURE 10.22

(a) Image of size 834×1114 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.

