CSI 4133 Computer Methods in Picture Processing and Analysis

Fall 2024

Pengcheng Xi, Ph.D.

Outline

- Sharpening spatial filters
  - First and second derivatives
- Unsharp masking
- Laplacian operator (2^{nd} order)
- Gradient edge operators (1^{st} order)
- Edge detectors
  - Zero-crossing and Canny edge detectors

Sharpening spatial filters

- The main objective of **sharpening** is to highlight fine details in an image or to enhance details that have been blurred
  - *Applications*: electronic printing, medical imaging, industrial inspection, and autonomous guidance in military systems
- Averaging (blurring) $\longleftrightarrow$ integration
- Sharpening $\longleftrightarrow$ spatial differentiation

Sharpening spatial filters

- Operators for sharpening by **digital differentiation**
  - Response of a **derivative operator** is proportional to the degree of discontinuity of the image at the point where the operator is applied
  - The image differentiation *enhances edges and other discontinuities* (e.g., noise) and *deemphasizes areas with slowly varying gray-level values*

Sharpening spatial filters

- An *ordinary* differential equation (**ODE**)
  - Only contains functions of one independent variable, and derivatives in that variable
- A *partial* differential equation (**PDE**)
  - Contains functions of multiple independent variables and their partial derivatives
  - A gray-level image is a function *f* of two variables x and y, say *f*(x,y)
    - Partial derivatives along the two spatial axes

First and second derivatives

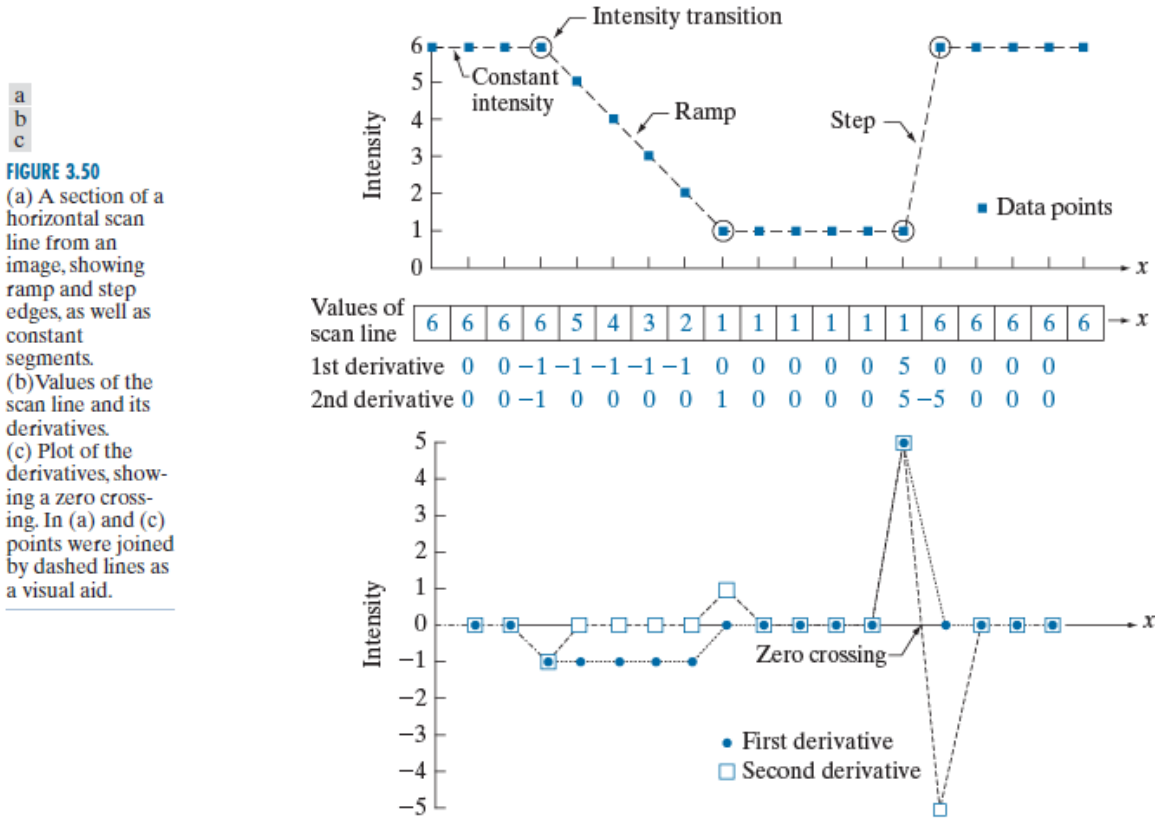
- **First-order** derivative of a one-dimensional function *f*(x):

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

- **Second-order** derivative of *f*(x):

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

1st and 2nd derivative



a
b
c

FIGURE 3.50
(a) A section of a horizontal scan line from an image, showing ramp and step edges, as well as constant segments. (b)Values of the scan line and its derivatives. (c) Plot of the derivatives, showing a zero crossing. In (a) and (c) points were joined by dashed lines as a visual aid.

www.imageprocessingbook.com

The *zero crossing* is useful for locating edges

1st and 2nd derivative

- **Nature** of detection:
  - First-order derivatives: detect gradients or *intensity changes* in an image where there is a rapid change in density, which corresponds to the boundaries of objects or edges in an image
    - Examples: *Roberts, Sobel, Prewitt operators*
    - Typically detect the edges themselves and do not highlight the exact center of the edge, because the response is spread over the region where the intensity change occurs
  - Second-order derivatives: detect areas where the *rate of change of intensity* is the greatest, by detecting the rate at which the gradient itself is changing
    - Examples: *Laplacian operator*
    - Emphasize peaks and zero-crossings, typically highlighting the central point of an edge, and more sensitive to fine details and noise

1st and 2nd derivative

- **Mathematical operation**:
  - First-order derivatives compute the gradient of the image intensity

$$G_x = \frac{\partial I}{\partial x}, \quad G_y = \frac{\partial I}{\partial y}$$

  - Second-order derivatives compute the second derivative of the image intensity
    - Calculates the curvature of the intensity function, highlighting regions where the gradient changes the most (sharp peaks, ridges, etc.)

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

1st and 2nd derivative

- **Edge response**:
  - First-order derivatives detect the edge by finding regions where intensity changes from one value to another.
    - The result is a strong response at the start and end of an edge, but the center of the edge may not be as clearly highlighted
  - Second-order derivatives detect edges by finding zero-crossings in the second derivative of intensity.
    - Creates strong responses at the center of the edge, effective in emphasizing fine details and thin lines, but is more sensitive to noise

1^{st} and 2^{nd} derivative

- **Sensitivity to noise**:
  - First-order derivatives are less sensitive to noise because they focus on broad intensity transitions
    - Tend to smooth out minor fluctuations in the image intensity
  - Second-order derivatives are more sensitive to noise since noise in an image often results in rapid intensity changes
    - Second derivative *amplifies high-frequency changes*
    - Requires more preprocessing like *Gaussian blurring* to reduce the effect

1st and 2nd derivative

- **Applications**:
  - First-order derivatives suitable for *basic edge detection* and *image sharpening in a more robust manner*
  - Second-order derivatives used for *detecting fine edges* and *enhancing intricate details*, particularly *thin lines and textures*
    - Suitable for applications where *detecting sharp features or small details* is crucial (e.g., medical imaging)
    - Often *combined with Gaussian blur to reduce noise* before applying the second derivative (e.g., the *Laplacian of Gaussian*, or LoG filter)

Sharpening spatial filters

- First-order derivatives
  - Generally produce thicker edges in an image
  - Generally have a stronger response to a gray-level step
- Second-order derivatives
  - Have a stronger response to fine detail, such as thin lines and isolated points
  - Has response stronger to a line than a step, and to a point than to a line

Sharpening spatial filters

- In most applications, the *second derivative is better suited* than the first derivative for image enhancement because of the *ability to enhance fine detail*
- The principal use of first derivatives is for *edge extraction*. They can be used in conjunction with the second derivative to obtain some impressive enhancement results

1st and 2nd derivative

- Summary of differences:

Aspect	First-order Derivative (Gradient-based)	Second-order Derivative (Laplacian-based)
Focus	Detects intensity changes (gradients)	Detects changes in the gradient (curvature)
Edge detection	Captures the beginning and end of edges	Captures the center of edges (zero-crossings)
Sensitivity to noise	Less sensitive to noise	More sensitive to noise
Visual outcome	Highlights broader edges	Highlights thin edges and fine details
Common operators	Roberts, Prewitt, Sobel	Laplacian
Application	General edge detection and contour finding	Fine detail detection, often combined with smoothing

Second derivatives – the Laplacian

- **Isotropic** filters
  - "*iso*" for same, and "*tropic*" from tropism, relating to direction
  - Whose response is independent of the direction of the discontinuities in the image to which the filter is applied
  - 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 derivatives – the Laplacian

- **Laplacian**
  - The simplest isotropic derivative operator
  - For a function (image) *f*(x,y) of two variables, defined as

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

  - Gives isotropic result for rotations *in increments of 90 degrees*

0	1	0
1	−4	1
0	1	0

Second derivatives – the Laplacian

- Laplacian

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

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

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

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

Implementations of the Laplacian

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

a b c d

FIGURE 3.51 (a) Laplacian kernel used to implement Eq. (3-62). (b) Kernel used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other Laplacian kernels.

www.imageprocessingbook.com

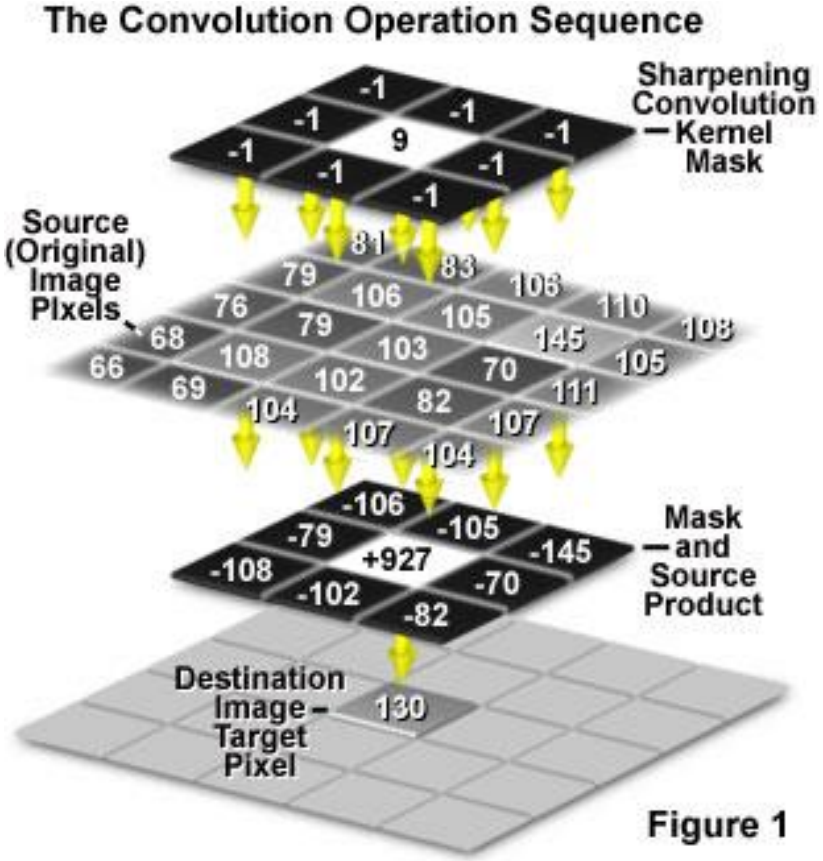
Laplacian for sharpening

- Laplacian *highlights sharp intensity transitions* in an image and *de-emphasizes regions of slowly varying intensities*
- By adding the Laplacian image to the original, background features can be "recovered" while still preserving the sharpening effect of the Laplacian:

$$g(x,y) = f(x,y) + c\left[\nabla^2 f(x,y)\right]$$

  - Where *f*(x,y) and *g*(x,y) are the input and sharpened images. *c*=-1 if the Laplacian kernels in Fig. 3.51 (a) or (b) is used, *c*=1 if either of the other two kernels is used.

The enhanced contours

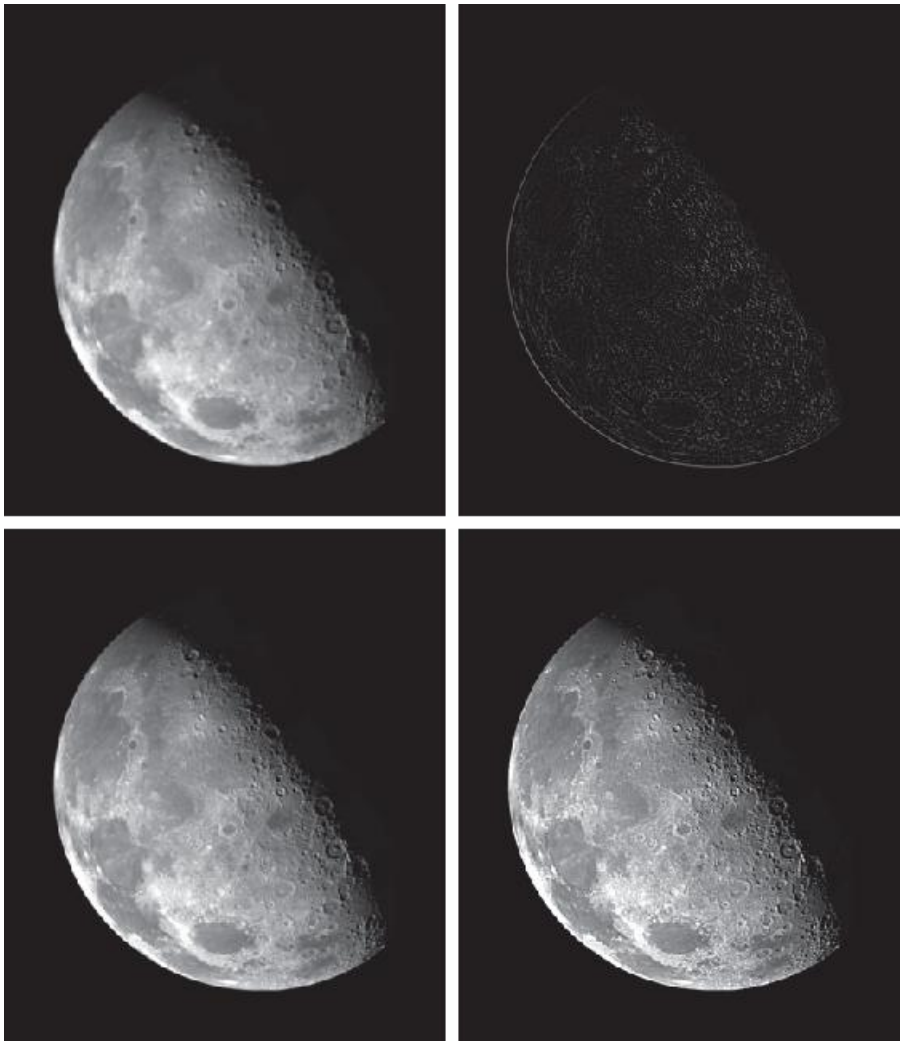


http://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/kernelmaskoperation/

Laplacian filtered image

a b
c d

FIGURE 3.52
(a) Blurred image of the North Pole of the moon.
(b) Laplacian image obtained using the kernel in Fig. 3.51(a).
(c) Image sharpened using Eq. (3-63) with $c = -1$.
(d) Image sharpened using the same procedure, but with the kernel in Fig. 3.51(b).
(Original image courtesy of NASA.)



www.imageprocessingbook.com

Scaled Laplacian image

FIGURE 3.53
The Laplacian image from Fig. 3.52(b), scaled to the full [0, 255] range of intensity values. Black pixels correspond to the most negative value in the unscaled Laplacian image, grays are intermediate values, and white pixels corresponds to the highest positive value.

www.imageprocessingbook.com

Unsharp masking and high-boost filtering

- Subtracting an unsharp (smoothed) version of an image from the original image
- **Unsharp masking** steps:
  - Blur the original image
  - Subtract the blurred image from the original (the resulting difference is called the mask)
  - Add the mask to the original

Letting $\bar{f}(x, y)$ denote the blurred image, the mask in equation form is given by:

$$g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y) \qquad (3\text{-}55)$$

Then we add a weighted portion of the mask back to the original image:

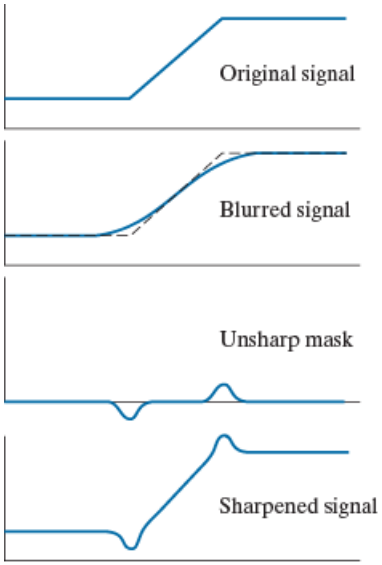
$$g(x, y) = f(x, y) + k\, g_{\text{mask}}(x, y) \qquad (3\text{-}56)$$

When k>1, the process is referred to as *high-boost unsharp masking*

1-D unsharp masking

a
b
c
d

FIGURE 3.54
1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).



www.imageprocessingbook.com

2-D unsharp masking

a b c
d e f

FIGURE 3.55 (a) Unretouched "soft-tone" digital image of size 469×600 pixels. (b) Image blurred using a 31×31 Gaussian lowpass filter with $\sigma = 5$. (c) Mask. (d) Result of unsharp masking using Eq. (3-65) with $k = 1$. (e) and (f) Results of highboost filtering with $k = 2$ and $k = 3$, respectively.

www.imageprocessingbook.com

First derivative – the gradient

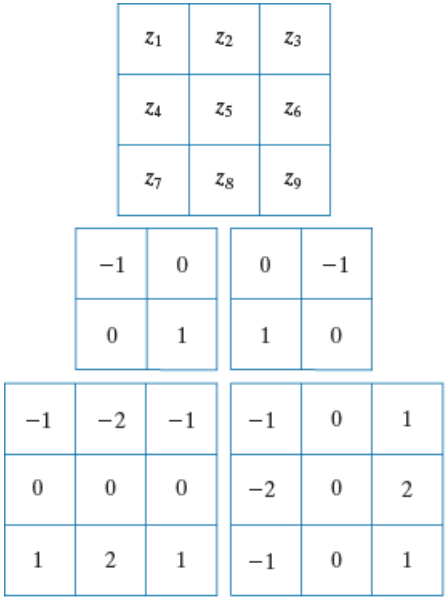
- First derivatives in image processing are implemented using the magnitude of the gradient
- The gradient of an image *f* at coordinates (x,y) is defined as the two dimensional column vector

$$\nabla f \equiv \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Discrete gradient operators

a
b c
d e

FIGURE 3.56
(a) A 3×3 region of an image, where the zs are intensity values. (b)–(c) Roberts cross-gradient operators. (d)–(e) Sobel operators. All the kernel coefficients sum to zero, as expected of a derivative operator.



www.imageprocessingbook.com

Gradient edge operators

- **Robert Cross operator**:
  - One of the earliest edge detection techniques using 2x2 convolution kernels
  - It computes the approximate gradient magnitude in both the horizontal and vertical directions. The edge strength is obtained by combining the results of these convolutions
  - Respond maximally to edges running at 45 degrees
  - One mask for each of the two perpendicular orientations
  - It is simple but sensitive to noise due to its small kernel size. Works best on images with well-defined edges

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

The magnitude of the gradient is calculated as: $|G| = \sqrt{G_x^2 + G_y^2}$

Gradient edge operators

- **Prewitt operator**:
  - A simple edge detection method that uses 3x3 convolution kernels to approximate the gradient in both horizontal and vertical directions
  - It applies convolution kernel G_x and G_y to detect horizontal and vertical edges

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

  - The magnitude of the gradient is calculated as: $|G| = \sqrt{G_x^2 + G_y^2}$
  - An improvement over the Roberts operator, as it uses a larger convolution mask, making it less sensitive to noise

Gradient edge operators

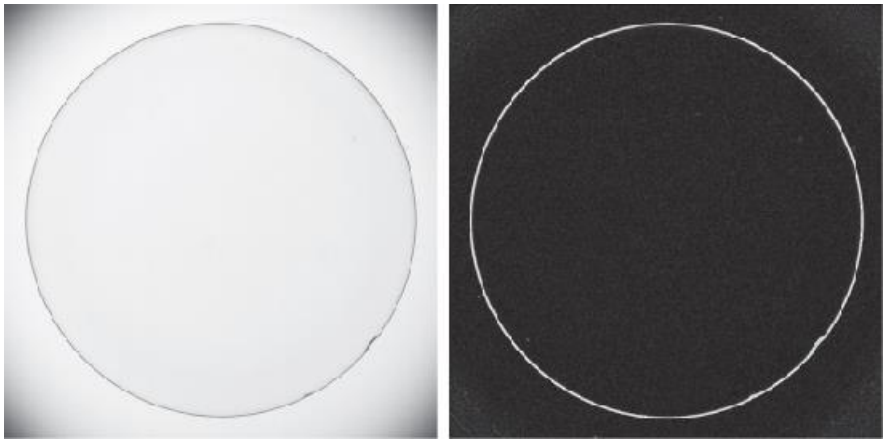
- **Sobel operator**:
  - An enhancement of the Prewitt operator, with an additional weighting to *emphasize the central pixel*
  - It emphasizes pixels that are closer to the center of the convolution kernel, making *the detection of edges smoother and more accurate*
  - Widely used due to its balance between edge detection and noise suppression.

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

Sobel gradient

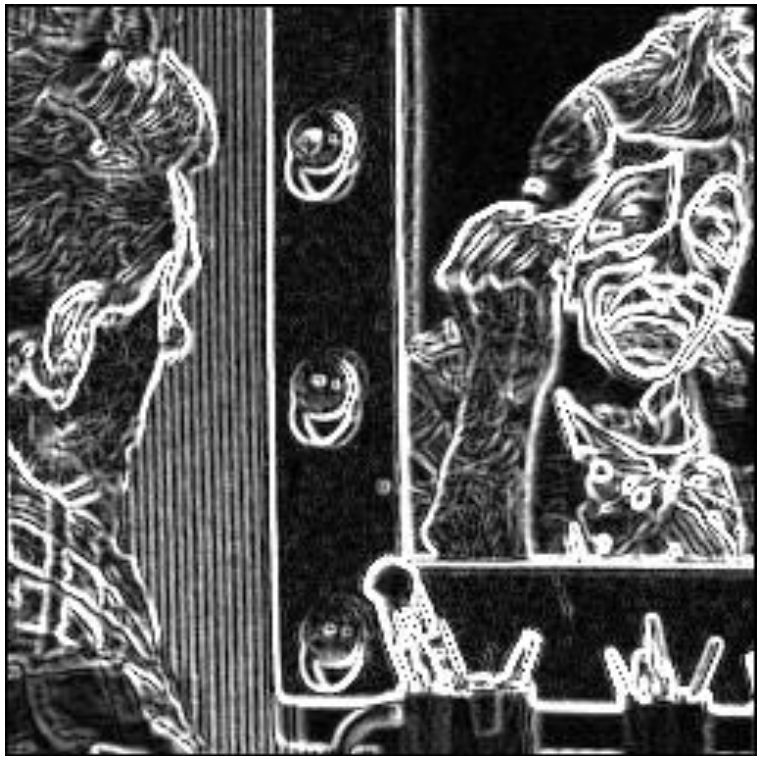
a b

FIGURE 3.57
(a) Image of a contact lens (note defects on the boundary at 4 and 5 o'clock).
(b) Sobel gradient.
(Original image courtesy of Perceptics Corporation.)



www.imageprocessingbook.com

Sobel (magnitude)



Gradient edge operators

- **Kirsch operator** (a directional operator):
  - Uses 3x3 masks to detect edges in eight possible directions. It computes *the maximum gradient magnitude* among all the directions
  - **Kernels**: there are *eight convolution kernels*, each corresponding to one of the eight compass directions (North, South, East, West, and the diagonals).
    - The kernel for detecting edges in the north direction is

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

    - The other kernels can be derived by rotating this kernel
    - The image is convolved with each of the eight kernels, and the *maximum response* from all directions is used as the *edge response at each pixel*

Comparison summary

Operator	Kernel Size	Noise Sensitivity	Directional Sensitivity	Complexity
Roberts	2x2	High	Diagonal	Low
Prewitt	3x3	Moderate	Vertical/Horizontal	Moderate
Sobel	3x3	Low	Vertical/Horizontal	Moderate
Kirsch	3x3 (8 directions)	Low	Multi-directional	High

Edge detection

- Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties of the image.

Criteria for edge detection

- Detection of edge with **low error rate**, which means that the detection should accurately catch as many edges shown in the image as possible.
- The edge point detected from the operator should accurately **localize on the center** of the edge.
- A given edge in the image should only be **marked once**, and where possible, image noise should not create false edges.

Edge detection

- Detect edges using the Sobel method (middle)
- Detect edges using the Laplacian method (right)
  - The edges of an image are located at the zero-crossings of the Laplacian
  - Advantages: no thresholds, subpixel localization
  - Problem: very sensitive to noise
    - Solution: to smooth the image before

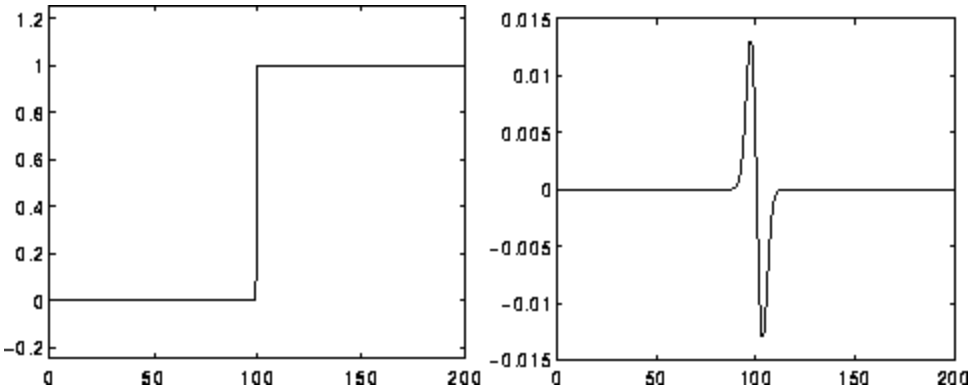
Input image

Zero crossing detector

- **Concept**: based on the behavior of second-order derivatives, typically using the Laplacian operator or the *Laplacian of Gaussian* (**LoG**).
  - It detects edges by identifying locations where the second derivative of the image intensity crosses zero, indicating a rapid change in the gradient.
- *How it works*:
  - Optional smoothing: To reduce noise, the image is often first smoothed using a Gaussian filter before applying the Laplacian. This combination is called LoG or Marr-Hildreth operator.
  - Laplacian filtering: First, a second derivative (like the Laplacian) is applied to the image to detect regions of rapid intensity changes.
  - Zero-crossing detection: The algorithm then identifies where these second derivative values cross from positive to negative or vice versa—these are potential edge locations.

Zero crossing detector

- Laplacian of Gaussian filter
  - 1-D LoG filter to a step edge
    - An 1-D image, 200 pixels long, containing a step edge
    - vs. the response of a 1-D LoG filter with Gaussian standard deviation 3 pixels



Zero crossing detector

- Laplacian of Gaussian filter
  - The behavior is largely governed by the *standard deviation of the Gaussian* used in the LoG filter
    - The higher the value is set, the more smaller features will be smoothed out, and hence fewer zero crossings will be produced

Zero crossing detector

- All edges detected by the zero crossing detector are in the form of closed curves in the same way that contour lines on a map are always closed
- The only exception is where the curve goes off the edge of the image

Zero crossing detector

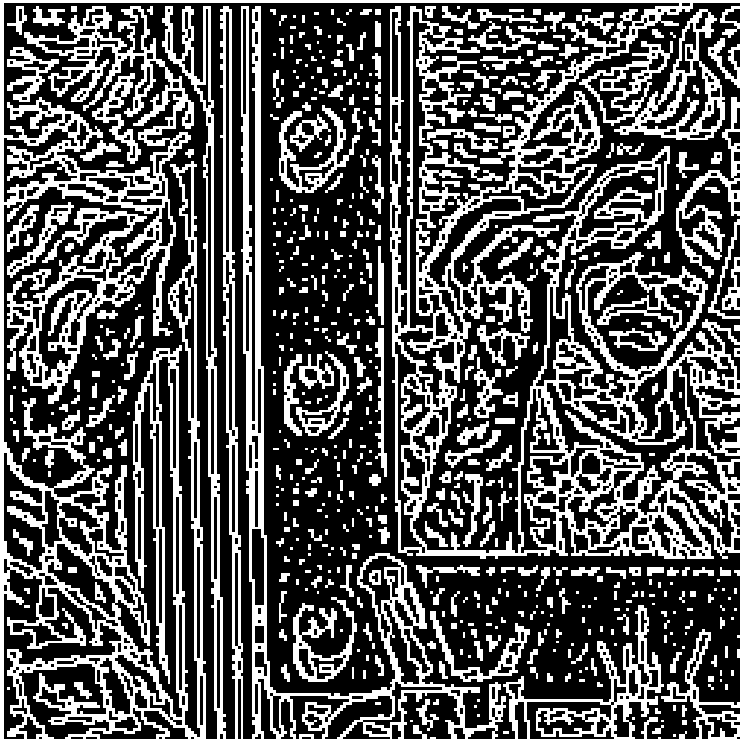
- An LoG filter with Gaussian standard deviation 1.0

http://www.cee.hw.ac.uk/hipr/html/zeros.html

Zero crossing detector

- The zero crossings from this image
  - Note the large number of minor features detected, which are mostly due to noise or very faint details. This smoothing corresponds to a fine ‘scale’.

http://www.cee.hw.ac.uk/hipr/html/zeros.html



Zero crossing detector

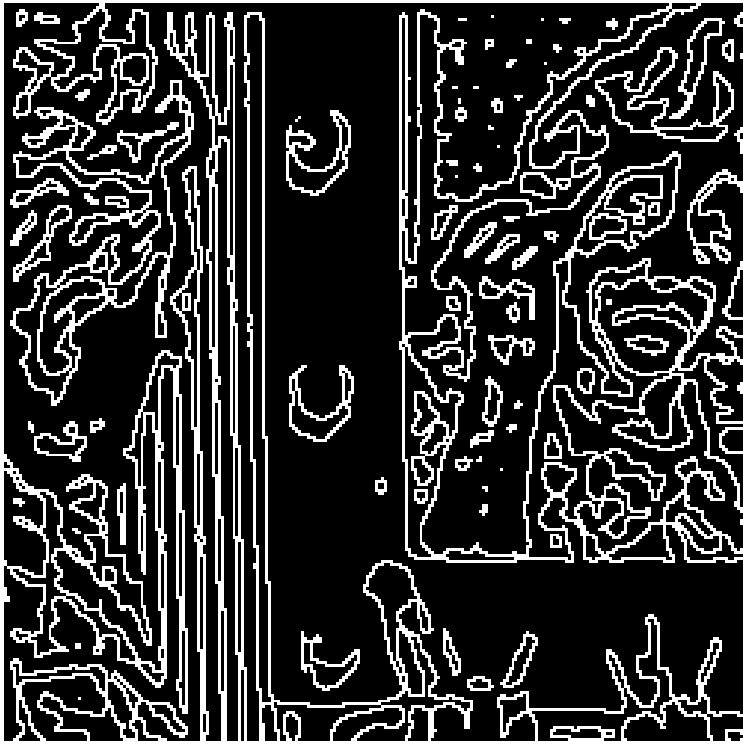
- An LoG filter with Gaussian standard deviation 2.0

http://www.cee.hw.ac.uk/hipr/html/zeros.html

Zero crossing detector

- The zero crossings
  - Note that there are far fewer detected crossings, and that those that remain are largely due to recognizable edges in the image. The thin vertical stripes on the wall are clearly visible.

http://www.cee.hw.ac.uk/hipr/html/zeros.html



Zero crossing detector

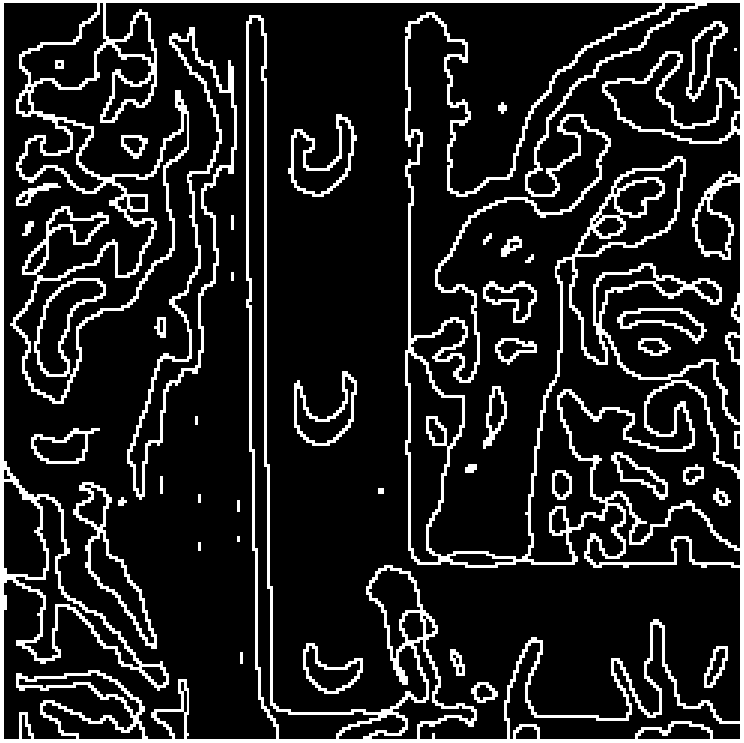
- An LoG filter with Gaussian standard deviation 3.0

http://www.cee.hw.ac.uk/hipr/html/zeros.html

Zero crossing detector

- The zero crossings in this image
  - Note how only the strongest contours remain due to the heavy smoothing. In particular, note how the thin vertical stripes on the wall no longer give rise to many zero crossings

http://www.cee.hw.ac.uk/hipr/html/zeros.html



Canny edge detector

- One of the most widely used and powerful edge detection algorithms
- Aims to achieve optimal edge detection by addressing three key criteria
  - **Good detection:** all real edges should be detected with minimal noise
  - **Good localization:** the detected edges should be as close as possible to the actual edges in the image
  - **Single response:** the algorithm should minimize multiple responses to a single edge

Canny edge detector

- How it works
  - Gaussian filtering
    - The image is first smoothed with a Gaussian filter to reduce noise
  - Gradient computation
    - The image's gradient (using first-order derivatives) is calculated, typically using the Sobel operator, to detect the intensity changes in both the x and y directions
  - Non-maximum suppression
    - Thin edges are identified by suppressing any pixel that is not a local maximum in the direction of the gradient
  - Double thresholding
    - Two thresholds are applied to determine strong and weak edges
  - Edge tracking by hysteresis
    - Weak edges are only retained if they are connected to strong edges, ensuring that noise does not generate spurious edge responses

Non-maximum suppression

- An edge thinning technique
  - Edges give rise to ridges in the gradient magnitude image
  - Applied to "thin" the edge. After applying gradient calculation, the edge extracted from the gradient value is still quite blurred.
  - Non-maximum suppression can help to suppress all the gradient values to 0 except the local maximal, which indicates location with the sharpest change of intensity value

(a) Thresholded gradient magnitude map with threshold at 10% of peak edge magnitude.
(b) Gradient map after non-maximal suppression.

Double thresholding

- High threshold (T_high): Pixels with gradient magnitudes greater than this value are considered strong edges.
- Low threshold (T_low): Pixels with gradient magnitudes below this value are considered non-edges.
- Pixels with gradient magnitudes between the two thresholds are considered weak edges.

Edge tracking by hysteresis

- The **weak edges** are then either kept or suppressed based on their connectivity to strong edges:
  - **Strong edges** are always retained in the final edge map.
  - **Weak edges** are kept **only if they are connected to a strong edge** (in a 8-connected neighborhood).
  - **Non-edges** are suppressed (set to 0).

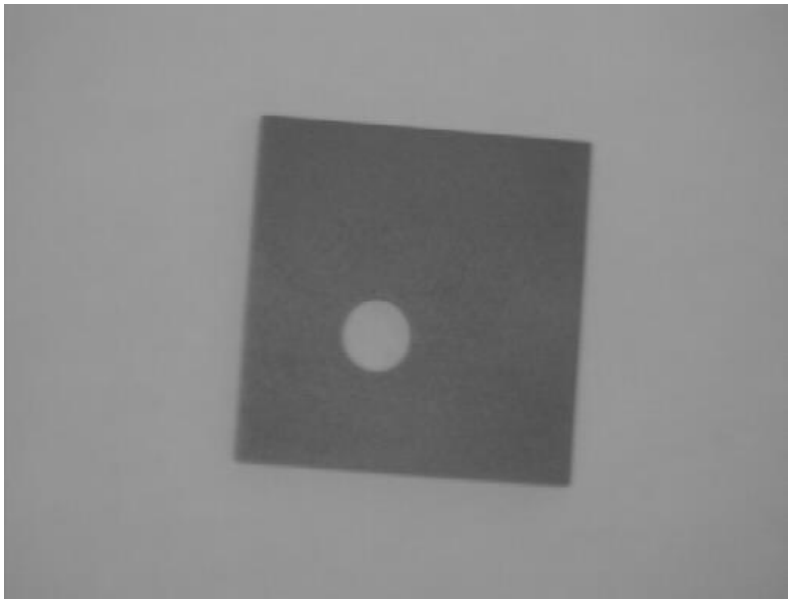
Canny edge detector

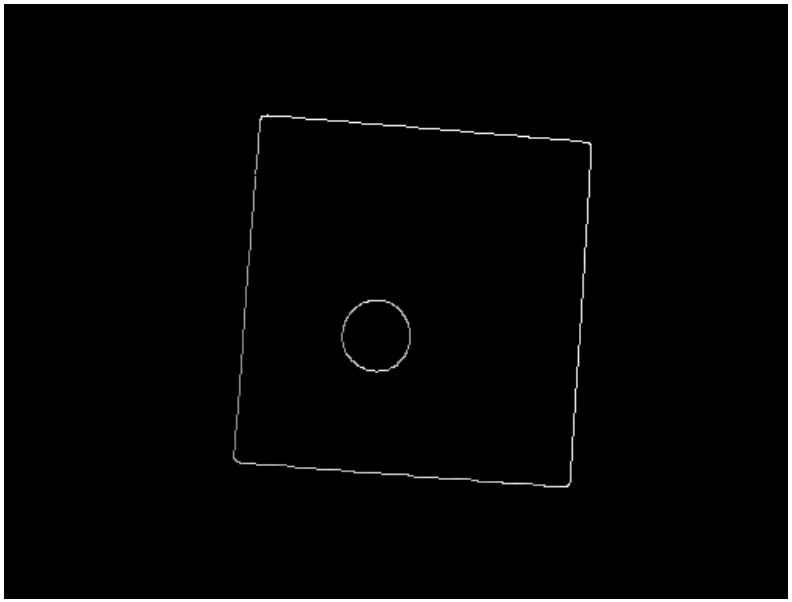
- Mathematics
  - First, gradients are computed $G_x = \frac{\partial I}{\partial x}, \quad G_y = \frac{\partial I}{\partial y}$
  - The gradient magnitude and direction are calculated

$$G = \sqrt{G_x^2 + G_y^2}, \quad \theta = \arctan\left(\frac{G_y}{G_x}\right)$$

  - Non-maximum suppression and double thresholding are then applied to refine and track edges

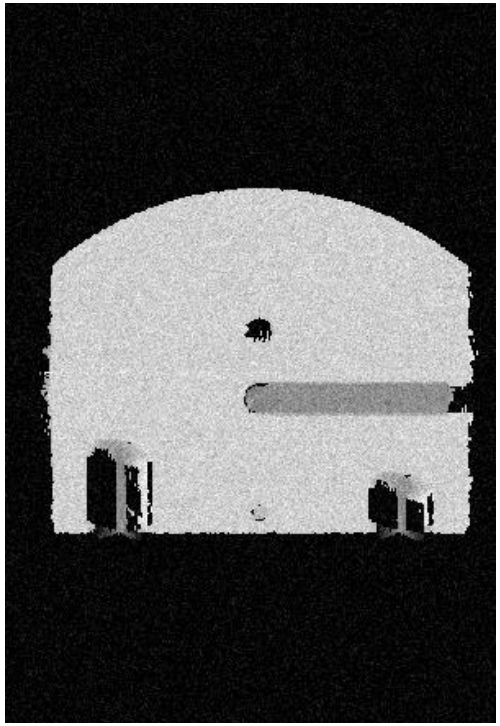
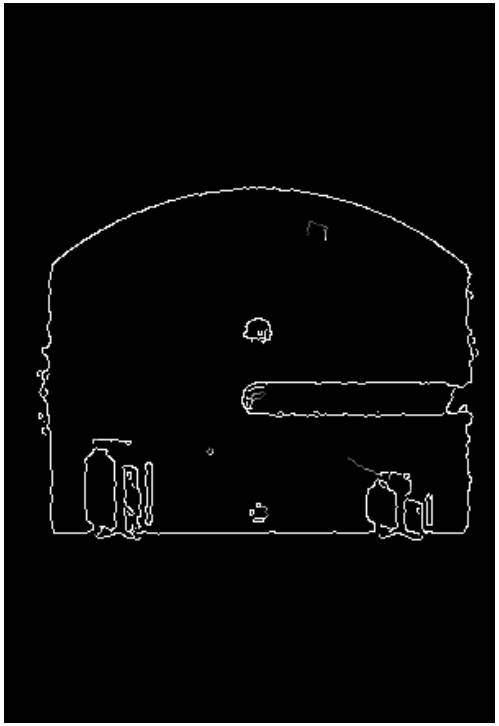
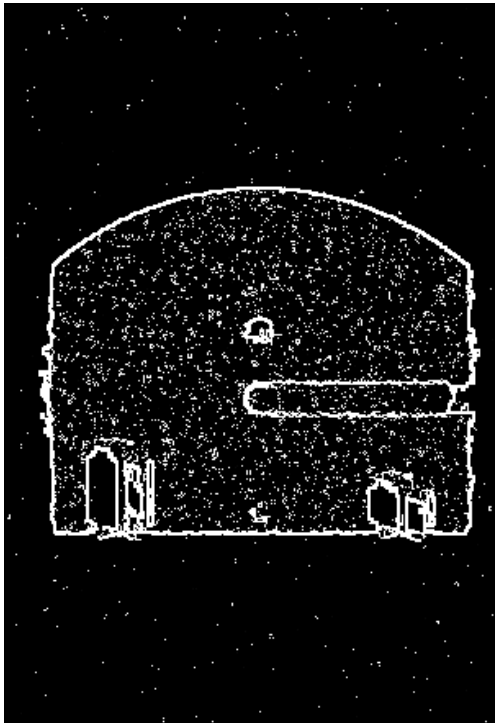
Canny edge detector





Canny edge detector

- Original vs. Canny vs. Sobel

Canny edge detector

- Three parameters
  - *Sigma* of the Gaussian mask
    - High sigma reduces the detector's sensitivity to noise, at the expense of losing some of the finer details in the image
  - The upper and lower *thresholds*
    - For tracking

Canny edge detector

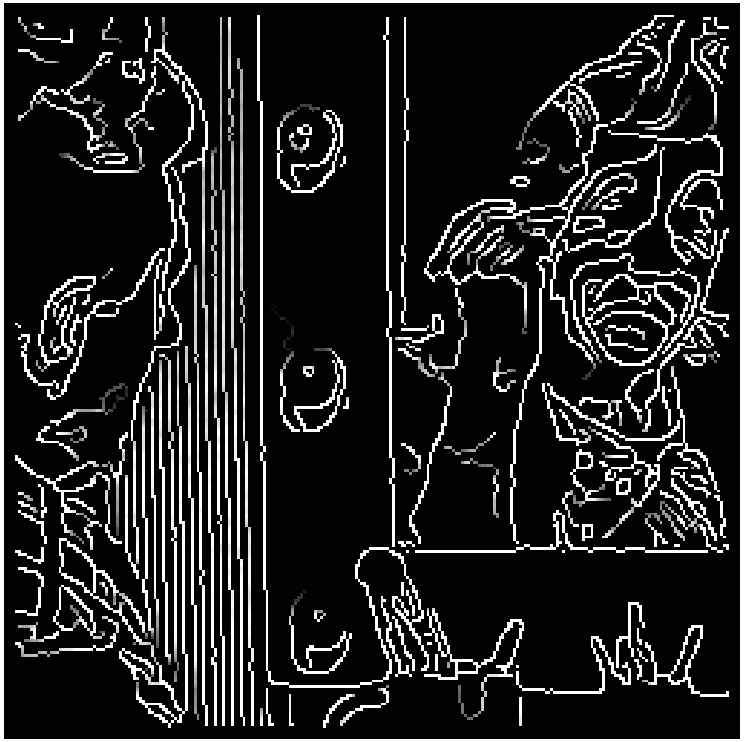
- The *Gaussian smoothing* in the Canny edge detector fulfills 2 purposes:
  - To control the amount of detail which appears in the edge image
  - To suppress noise
- The upper and lower thresholds
  - Setting the lower threshold too high will cause noisy edges to break up
  - Setting the upper threshold too low increases the number of spurious and undesirable edge fragments appearing in the output

Canny edge detector

- Original image

Canny edge detector

- Sigma = 1.0
- T_{high}=255 T_{low}=1
  - Most of the major edges are detected, lots of detail in has been picked out well
    - Note that this may be too much detail for subsequent processing



http://www.cee.hw.ac.uk/hipr/html/canny.html

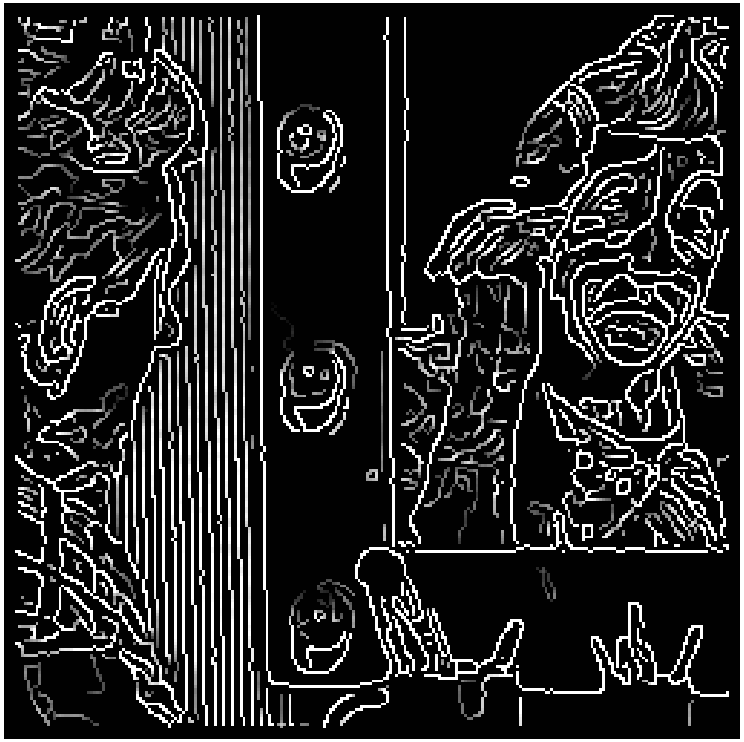
Canny edge detector

- Sigma = 1.0
- T_{high}=255 T_{low}=220
  - The edges have become more broken up than in the previous image, which is likely to be bad for subsequent processing.
  - Also the vertical edges on the wall have not been detected along their full length

http://www.cee.hw.ac.uk/hipr/html/canny.html

Canny edge detector

- Sigma = 1.0
- T_{high}=128 T_{low}=1
  - Many more faint edges are detected along with some short ‘noisy’ fragments.
  - Note that the detail in the clown’s hair is now picked out.



http://www.cee.hw.ac.uk/hipr/html/canny.html

Canny edge detector

- Sigma = 2.0
- T_{high}=128 T_{low}=1
  - Much of the detail on the wall is no longer detected, but most of the strong edges remain.
  - The edges also tend to be smoother and less noisy

http://www.cee.hw.ac.uk/hipr/html/canny.html

Canny edge detector

- Advantages:
  - Noise reduction: By applying Gaussian smoothing, it minimizes noise in the image.
  - Edge localization: Provides accurate edge detection with well-defined boundaries.
  - Reduced false positives: The double threshold and hysteresis tracking steps ensure that only significant edges are detected and noise-related responses are minimized.
- Disadvantages:
  - Computationally intensive: The multi-step process requires more computation than simpler edge detectors like Sobel or Prewitt.
  - Requires parameter tuning: The effectiveness of the Canny operator depends on choosing appropriate high and low thresholds for double thresholding.

Aspect	Zero Crossing Detector	Canny Edge Detector
Main idea	Detects edges by finding zero crossings in the second derivative (e.g., Laplacian)	Detects edges using first-order derivatives and multi-step processing
Noise handling	Sensitive to noise unless combined with Gaussian smoothing	Handles noise well with Gaussian filtering as the first step
Edge localization	Detects thin edges but may not localize well without noise suppression	Very good edge localization through non-maximum suppression
Multiple responses	Can produce multiple responses for a single edge	Minimizes multiple responses through edge tracking
Complexity	Moderate (with Laplacian of Gaussian)	High (due to multiple steps: filtering, gradient calculation, suppression, etc.)
Directionality	Does not provide direction information	Detects edge direction using gradient orientation
Typical use cases	Medical imaging, fine detail detection	General-purpose edge detection, object recognition, computer vision

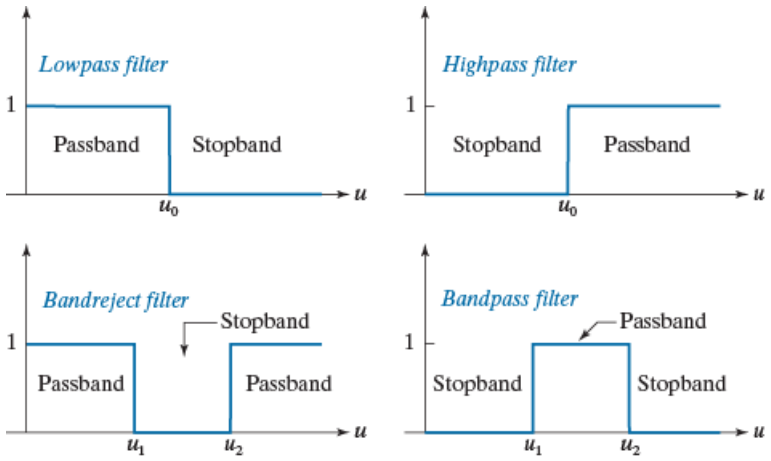
Combining spatial enhancement methods

- Example in image processing
  - **Goal**: to sharpen it by bringing out skeletal details (for detecting diseases such as bone infections or tumors)
  - **Challenge**: narrow dynamic range of the intensity levels and high noise content
  - **Strategy**: to use the Laplacian to highlight fine detail, and the gradient to enhance prominent edges
    - Use a smoothed version of the gradient image to mask the Laplacian image
    - To increase the dynamic range of the intensity levels by using an intensity transformation

Ideal 1-D filters in frequency space

a b
c d

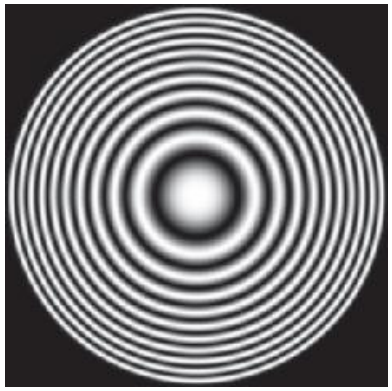
FIGURE 3.58
Transfer functions of ideal 1-D filters in the frequency domain (u denotes frequency).
(a) Lowpass filter.
(b) Highpass filter.
(c) Bandreject filter.
(d) Bandpass filter.
(As before, we show only positive frequencies for simplicity.)



www.imageprocessingbook.com

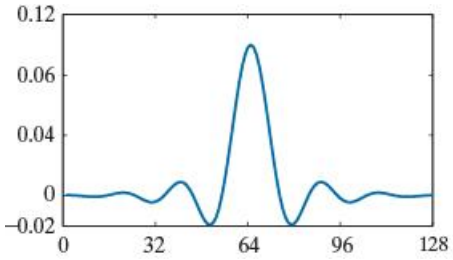
Zone plate and low-pass filter

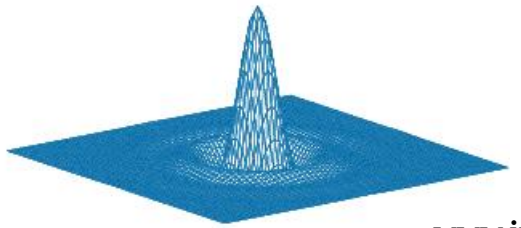
FIGURE 3.59
A zone plate image of size 597 × 597 pixels.



a b

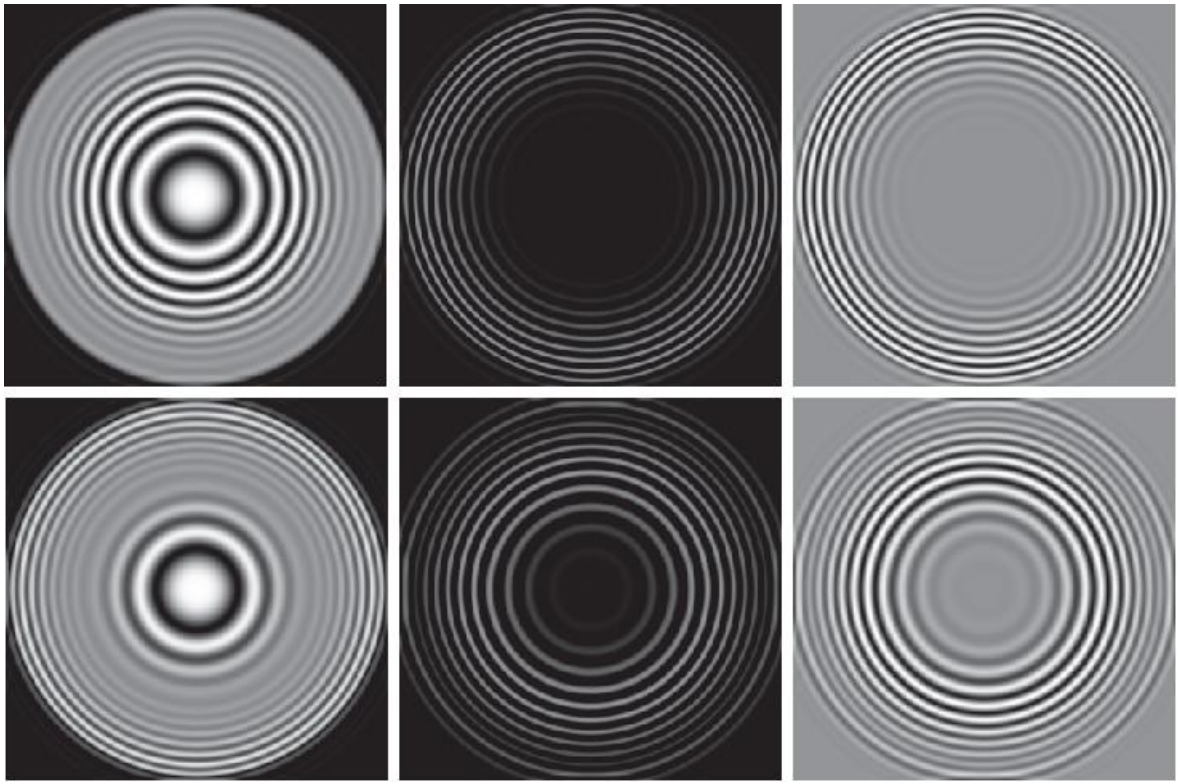
FIGURE 3.60
(a) A 1-D spatial lowpass filter function. (b) 2-D kernel obtained by rotating the 1-D profile about its center.





www.imageprocessingbook.com

Filtering results



www.imageprocessingbook.com

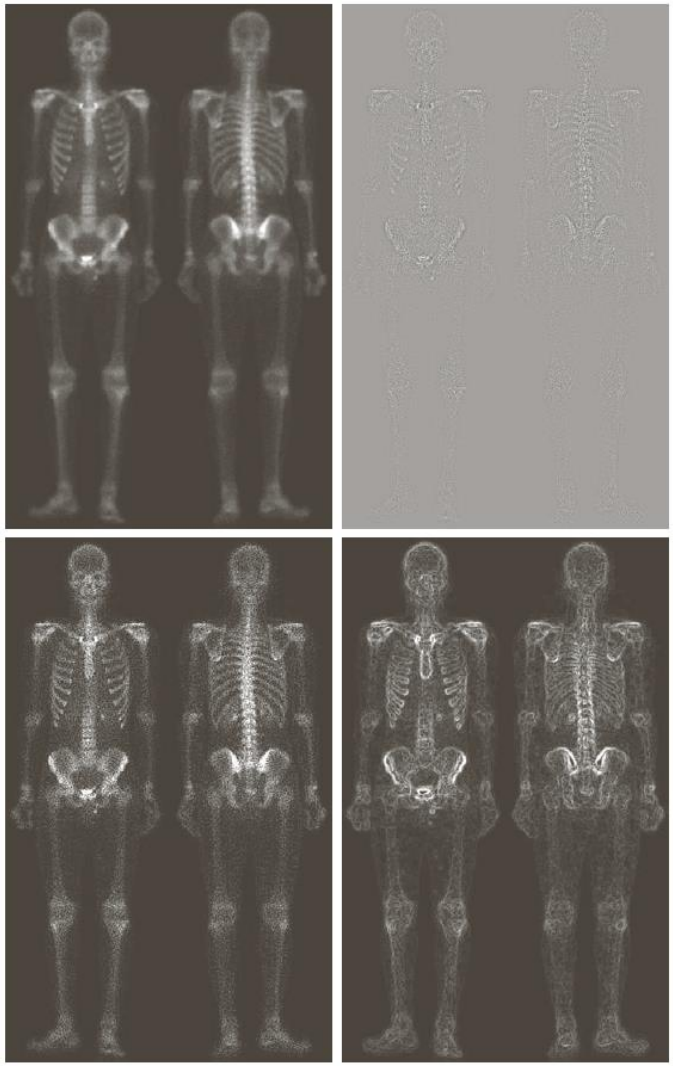
a b c
d e f

FIGURE 3.62
Spatial filtering of the zone plate image. (a) Lowpass result; this is the same as Fig. 3.61(b). (b) Highpass result. (c) Image (b) with intensities scaled. (d) Bandreject result. (e) Bandpass result. (f) Image (e) with intensities scaled.

Another example

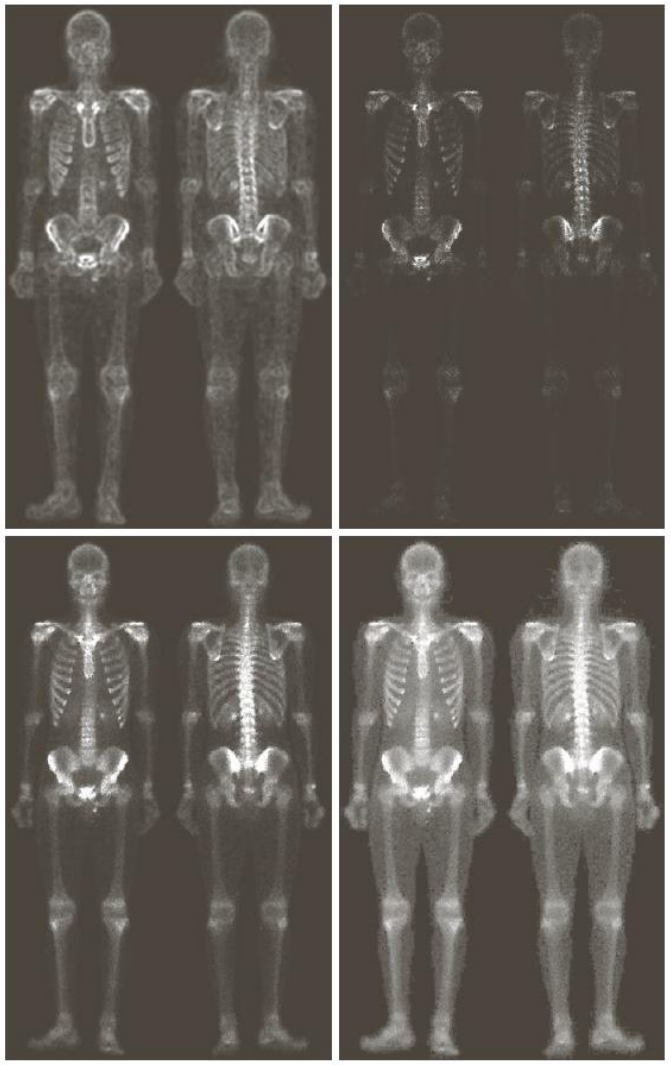
a b
c d

FIGURE 3.63
(a) Image of whole body bone scan.
(b) Laplacian of (a).
(c) Sharpened image obtained by adding (a) and (b).
(d) Sobel gradient of image (a). (Original image courtesy of G.E. Medical Systems.)



e f
g h

FIGURE 3.63
(Continued)
(e) Sobel image smoothed with a 5×5 box filter.
(f) Mask image formed by the product of (b) and (e).
(g) Sharpened image obtained by the adding images (a) and (f).
(h) Final result obtained by applying a power-law transformation to (g). Compare images (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)



Combining spatial enhancement methods

- Steps in the previous example
  - (a) original image (input)
  - (b) Laplacian image
  - (c) sharpened image = (a) + (b)
  - (d) Sobel image of (a)
  - (e) Sobel image smoothed with a 5x5 averaging filter
  - (f) mask image = (b) * (e)
  - (g) sharpened image = (a) + (f)
  - (h) final image (output) = power-law transformation to (g)

Spatial filter

- [question] Using the Sobel operator, compute the gradient magnitude of the 2nd, 3rd and 4th pixels of the fourth line in the following image.

0	8	5	5	5	7	8
4	7	8	3	2	2	8
1	1	0	5	6	6	6
1	**1**	**1**	**5**	7	8	6
1	1	0	4	6	8	6
8	1	1	1	6	6	6
8	1	1	1	1	6	6

Step-by-step solution

- Step 1: locate the pixel and extract the 3x3 neighborhood
- Step 2: apply the Sobel operator in the x-Direction (G_x)
- Step 3: apply the Sobel operator in the y-Direction (G_y)
- Step 4: compute the gradient magnitude

Computation for the 2nd pixel

- Neighborhood = $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$
- G_x·Neighborhood = $\begin{bmatrix} (-1)\cdot 1 & 0\cdot 1 & 1\cdot 0 \\ (-2)\cdot 1 & 0\cdot 1 & 2\cdot 1 \\ (-1)\cdot 1 & 0\cdot 1 & 1\cdot 0 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ -2 & 0 & 2 \\ -1 & 0 & 0 \end{bmatrix}$
- Sum all elements

$$G_x = -1 + 0 + 0 + (-2) + 0 + 2 + (-1) + 0 + 0 = -2$$

Computation for the 2nd pixel

- Neighborhood = $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$
- G_y·Neighborhood =

$$\begin{bmatrix} 1 \cdot 1 & 2 \cdot 1 & 1 \cdot 0 \\ 0 \cdot 1 & 0 \cdot 1 & 0 \cdot 1 \\ (-1) \cdot 1 & (-2) \cdot 1 & (-1) \cdot 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & 0 & 0 \\ -1 & -2 & 0 \end{bmatrix}$$

- Sum all elements

$$G_y = 1 + 2 + 0 + 0 + 0 + 0 + (-1) + (-2) + 0 = 0$$

Computation for the 2nd pixel

- The gradient magnitude is given by:

$$\text{Gradient Magnitude} = \sqrt{G_x^2 + G_y^2} = \sqrt{(-2)^2 + 0^2} = \sqrt{4} = 2$$