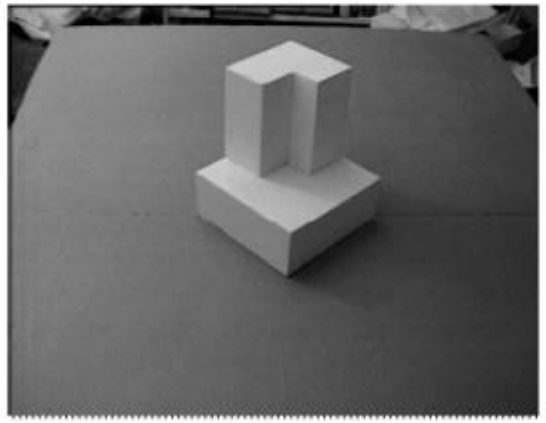
BSB663
Image Processing

Pinar Duygulu

Slides are adapted from
Gonzales & Woods,
Emmanuel Agu
Selim Aksoy

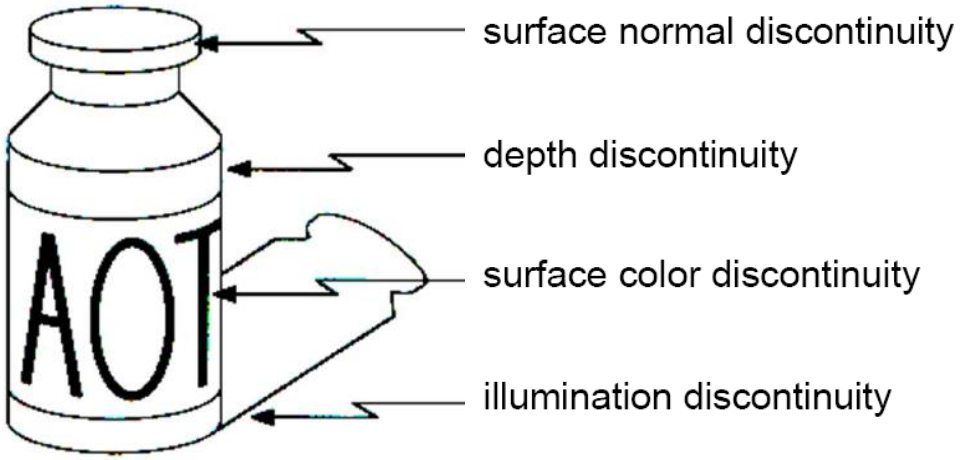
What is an edge?

- Edge? sharp change in brightness (discontinuities)
- Where do edges occur?
  - **Actual edges:** Boundaries between objects
  - Sharp change in brightness can also occur within object
    - Reflectance changes
    - Change in surface orientation
    - Illumination changes. E.g. Cast shadow boundary

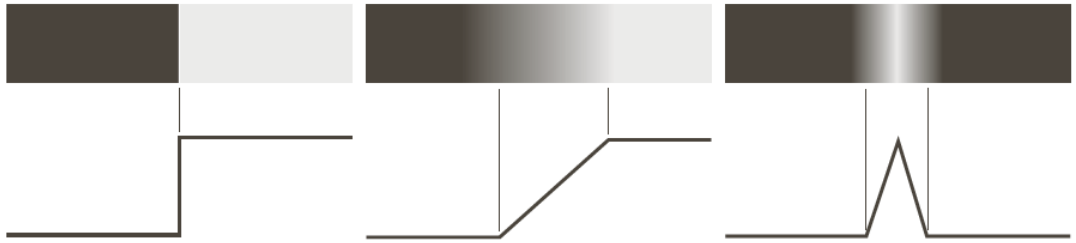


Edge detection

- Sharp changes in the image brightness occur at:
  - Object boundaries
    - A light object may lie on a dark background or a dark object may lie on a light background.
  - Reflectance changes
    - May have quite different characteristics – zebras have stripes, and leopards have spots.
  - Cast shadows
  - Sharp changes in surface orientation



Edge models



a b c

FIGURE 10.8
From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

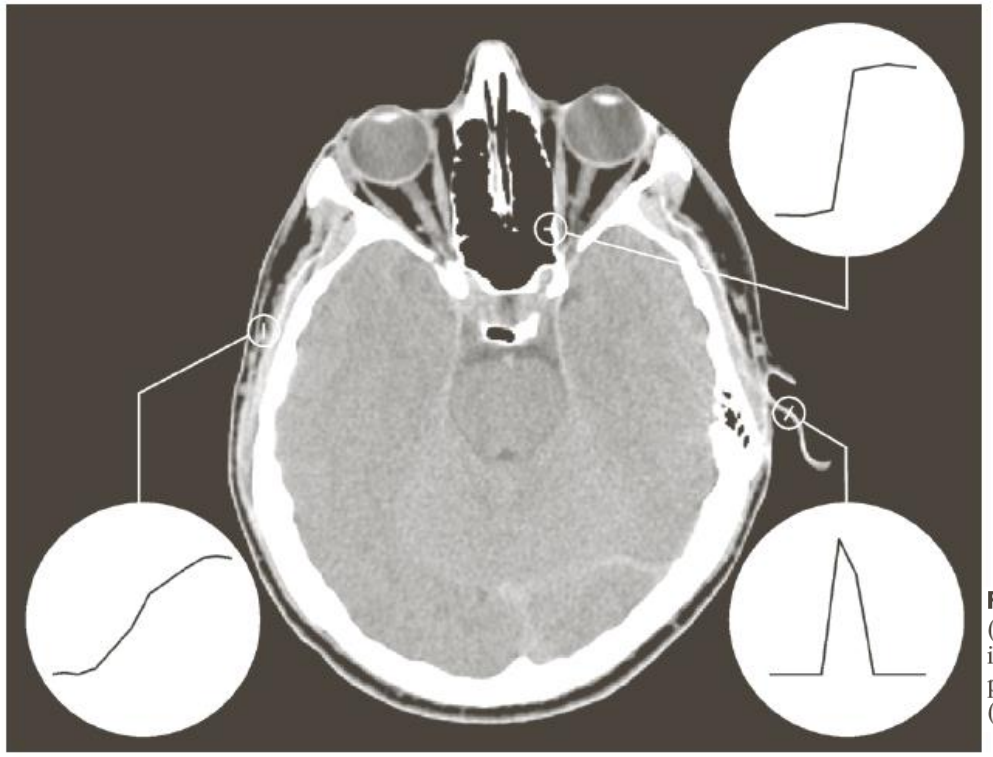
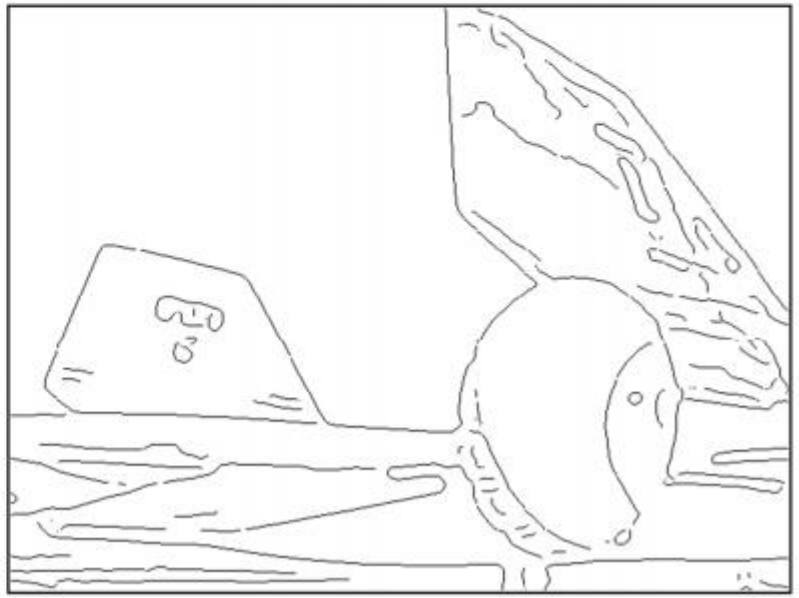


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

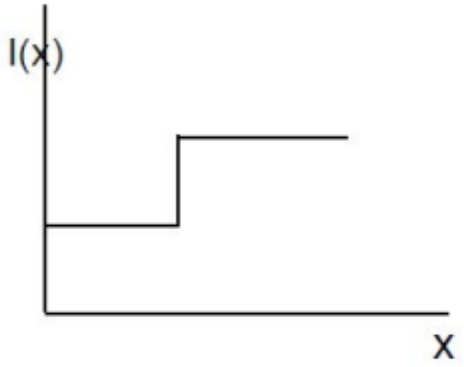
Edge detection

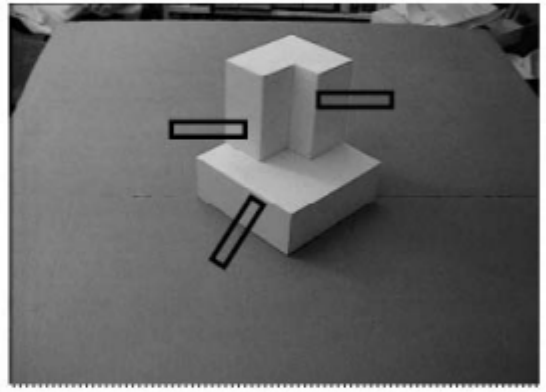
- Image processing task that finds edges and contours in images
- Edges so important that human vision can reconstruct edge lines

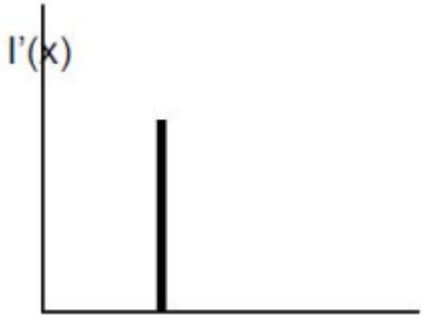


Characteristics of an Edge

- Edge: A sharp change in brightness
- Ideal edge is a step function in some direction



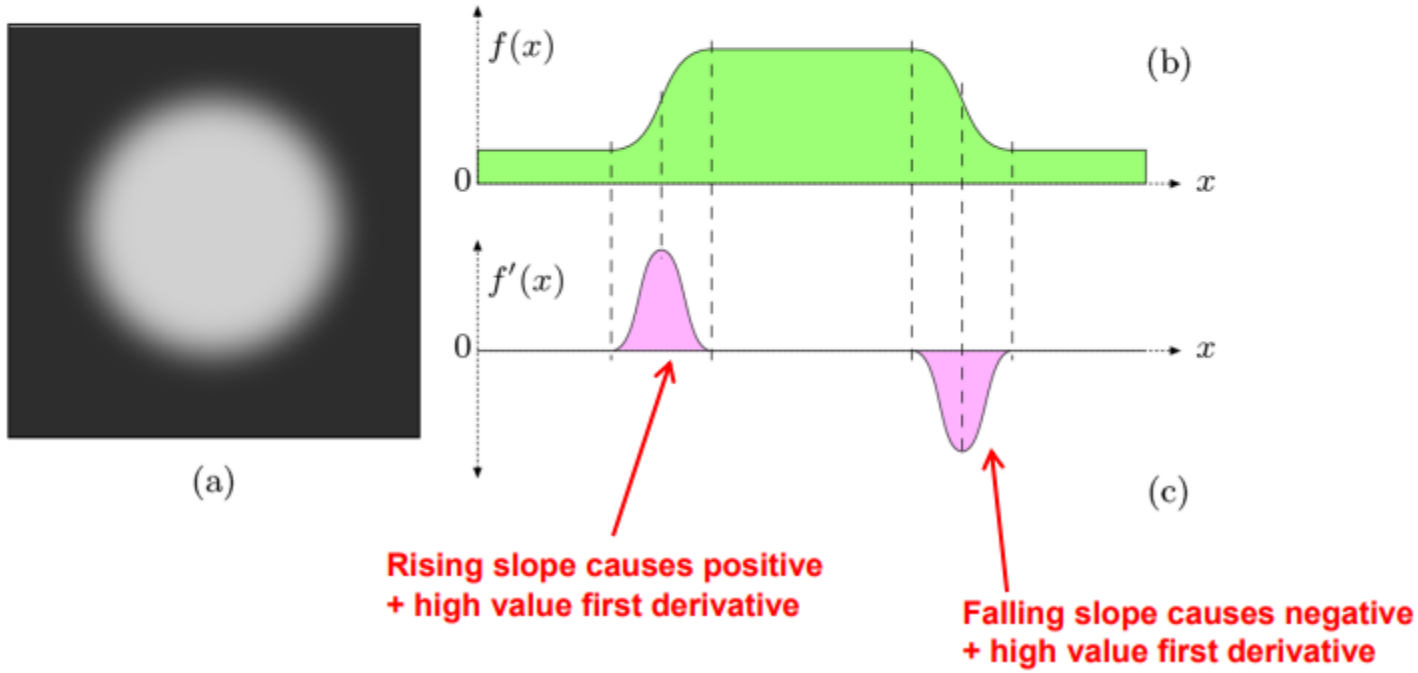




Characteristics of an Edge

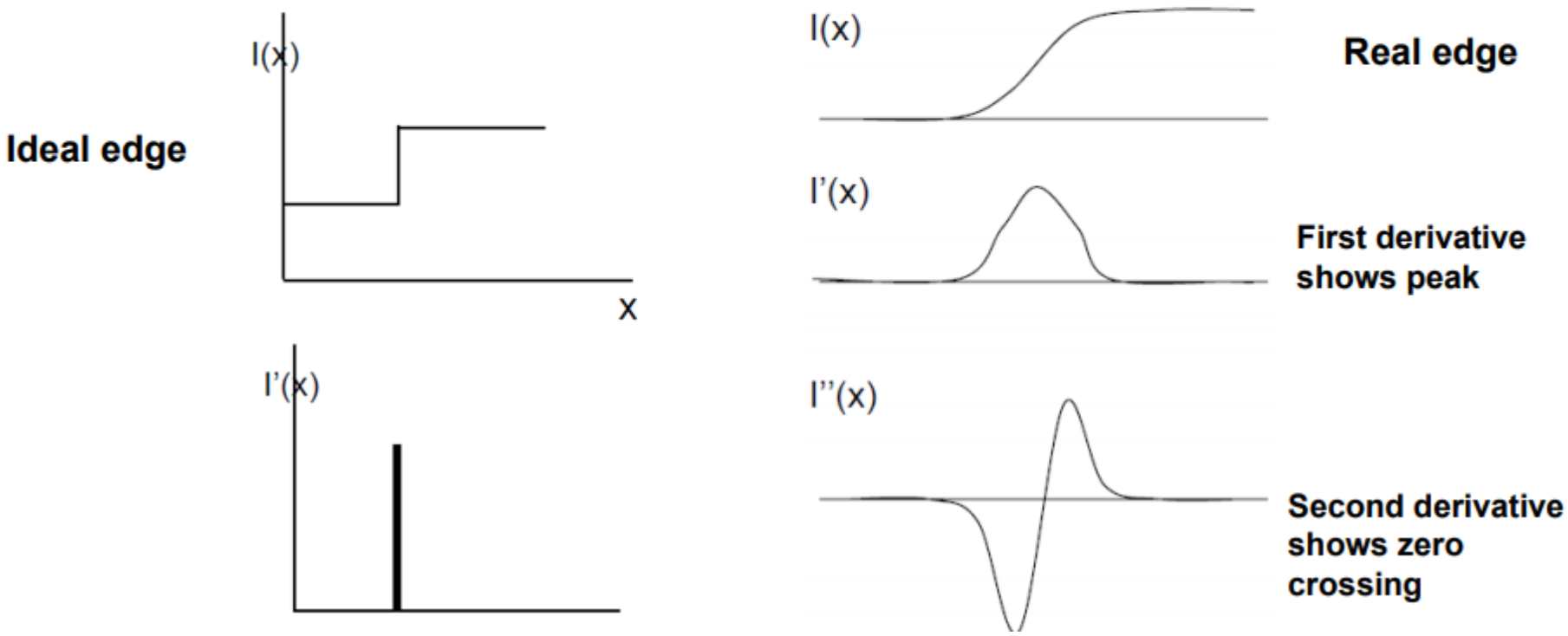
- Real (non-ideal) edge is a slightly blurred step function
- Edges can be characterized by high value first derivative

$$f'(x) = \frac{df}{dx}(x)$$



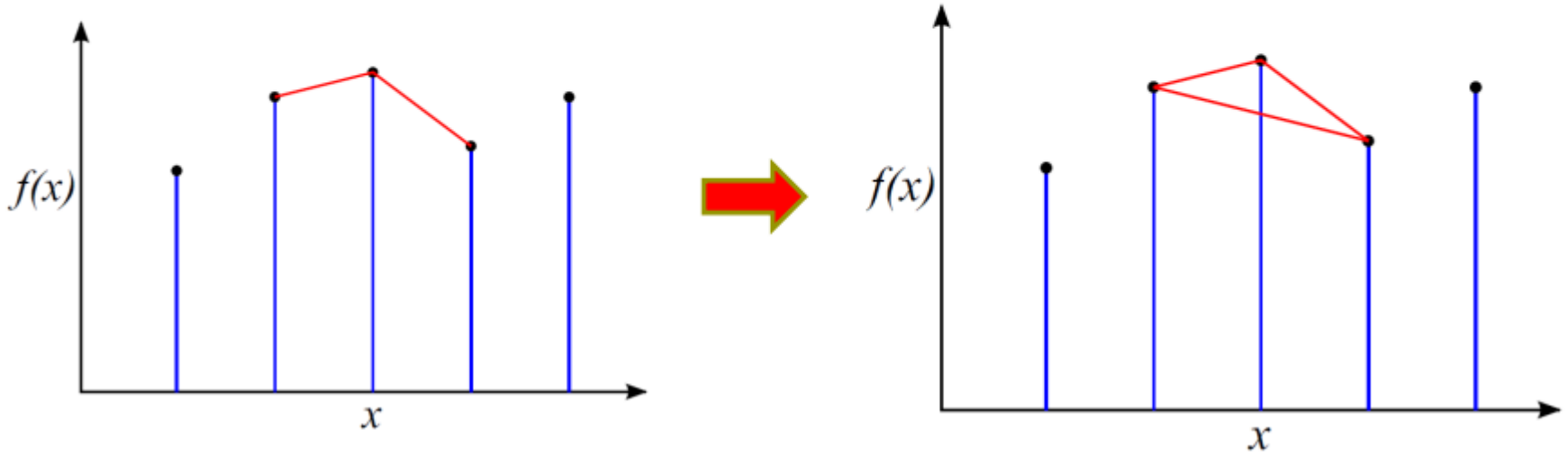
Characteristics of an Edge

- Ideal edge is a step function in certain direction.
- First derivative of I(x) has a **peak** at the edge
- Second derivative of I(x) has a **zero crossing** at edge



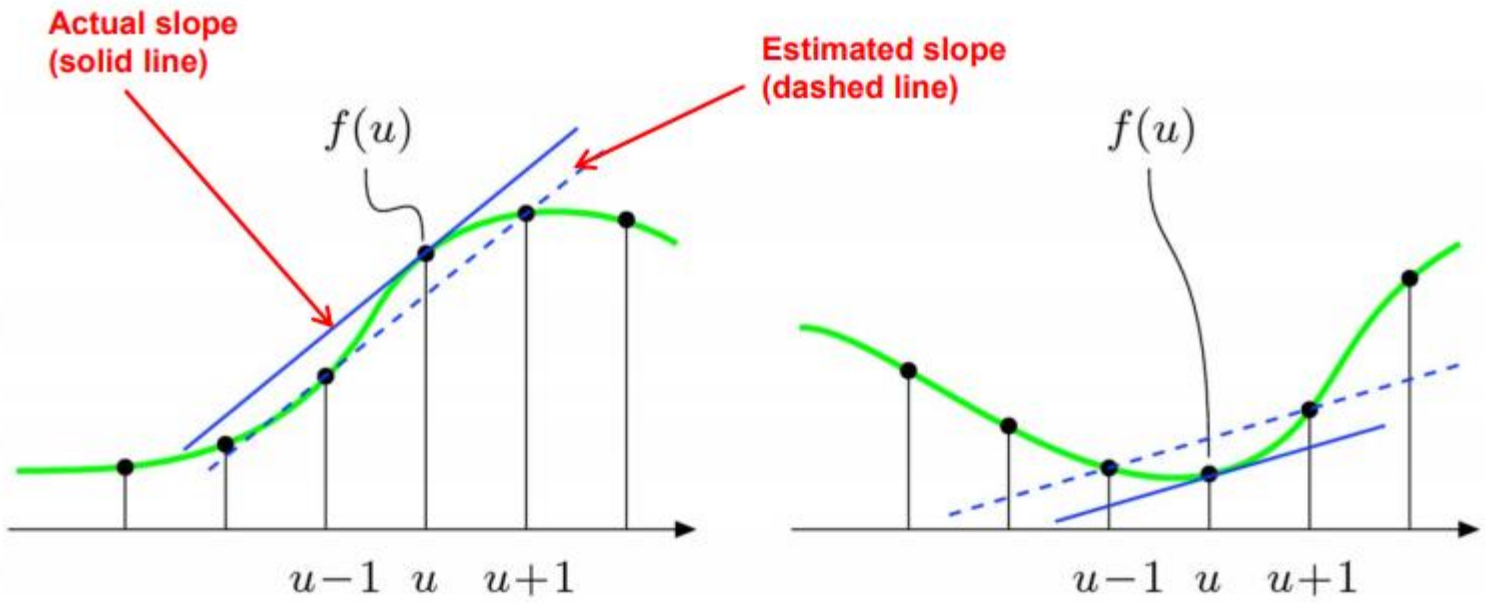
Slopes of discrete functions

- Left and right slope may not be same
- Solution? Take average of left and right slope



Computing Derivative of Discrete Function

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$



Finite differences

- Forward difference (right slope)

$$\Delta_+ f(x) = f(x+1) - f(x)$$

- Backward difference (left slope)

$$\Delta_- f(x) = f(x) - f(x-1)$$

- Central Difference (average slope)

$$\Delta f(x) = \frac{1}{2}\left(f(x+1) - f(x-1)\right)$$

Definition: Function Gradient

- Let $f(x,y)$ be a 2D function
- **Gradient:** Vector whose direction is in direction of maximum rate of change of f and whose magnitude is maximum rate of change of f
- Gradient is perpendicular to edge contour

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]^T$$

- magnitude = $[(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2]^{1/2}$
- direction = $\tan^{-1}(\frac{\partial f / \partial y}{\partial f / \partial x})$

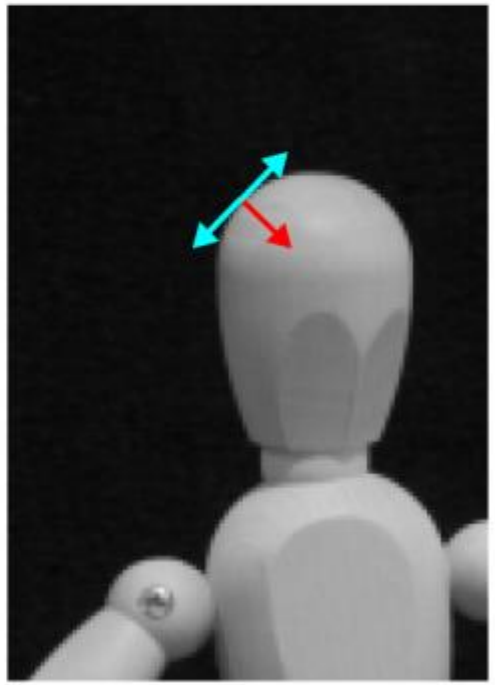


Image Gradient

- Image is 2D discrete function
- Image derivatives in horizontal and vertical directions

$$\frac{\partial I}{\partial u}(u,v) \quad \text{and} \quad \frac{\partial I}{\partial v}(u,v)$$

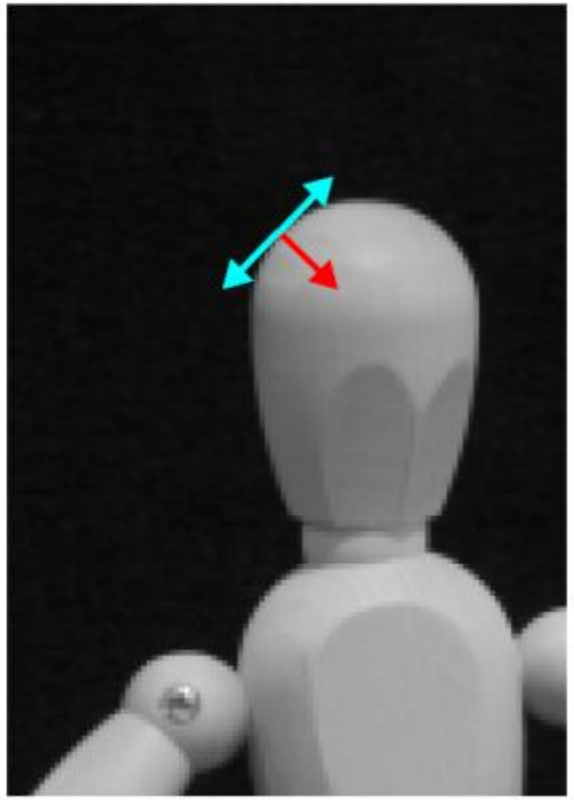
- Image gradient at location (u,v)

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix}$$

- Gradient magnitude

$$|\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2}$$

- Magnitude is invariant under image rotation, used in edge detection



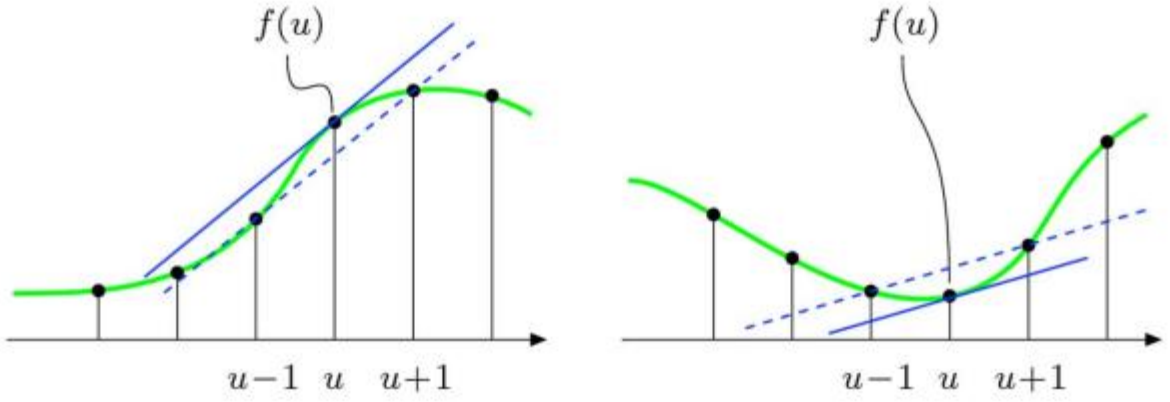
Derivative filters

- Recall that we can compute derivative of discrete function as

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$

- Can we make linear filter that computes central differences

$$H_x^D = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}$$



Finite Differences as Convolutions

- Forward difference

$$\Delta_+ f(x) = f(x+1) - f(x)$$

- Take a convolution kernel $H = [0 \quad -1 \quad 1]$

$$\Delta_+ f = f * H$$

Finite Differences as Convolutions

- Central difference

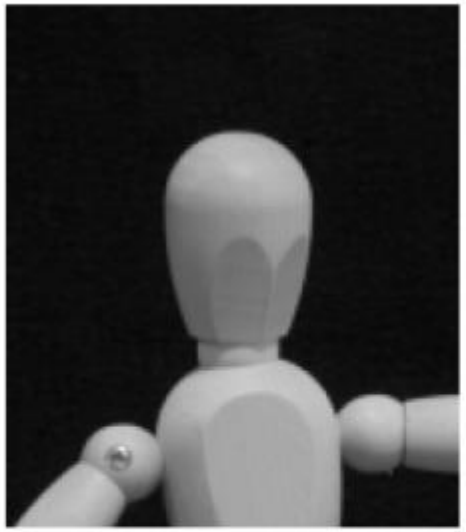
$$\Delta f(x) = \frac{1}{2}\left(f(x+1) - f(x-1)\right)$$

- Convolution kernel is: $H = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix}$

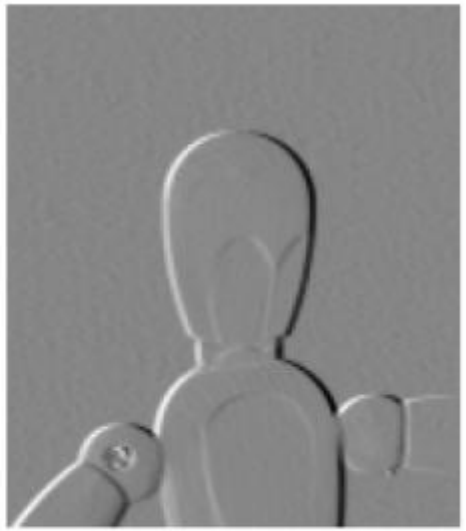
$$\Delta f(x) = f * H$$

- **Notice:** Derivative kernels sum to zero

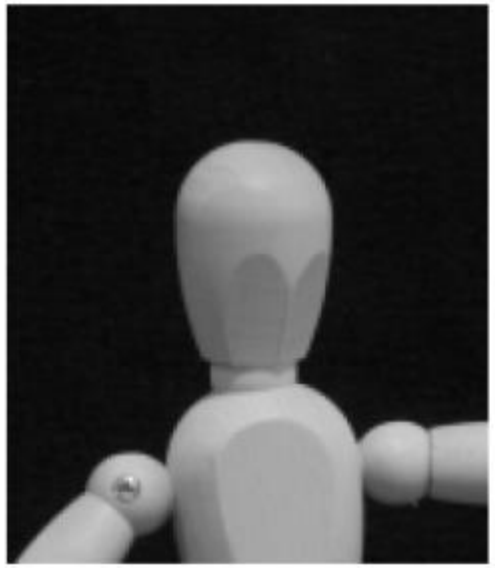
x-Derivative of Image using Central Difference



$$* \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} =$$



y-Derivative of Image using Central Difference



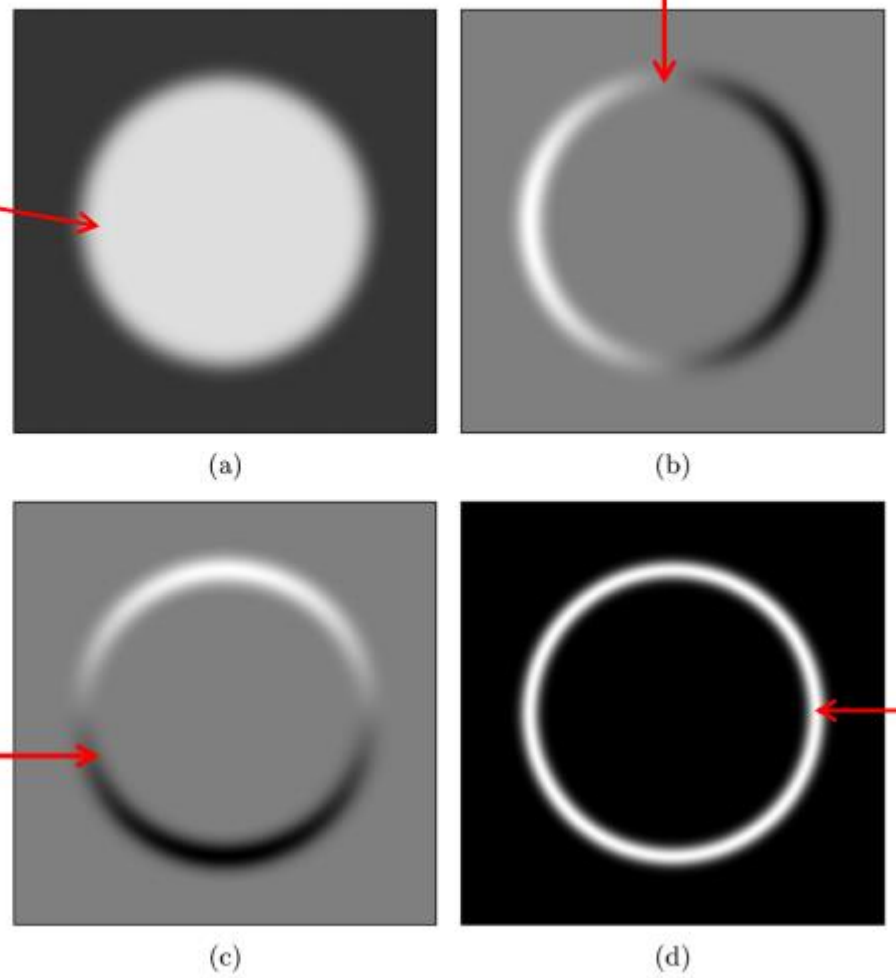
$$* \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} =$$

Derivative Filters

Gradient slope in horizontal direction

$$H_x^D = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}$$

A synthetic image



$$H_y^D = \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

Gradient slope in vertical direction

Magnitude of gradient

Edge Operators

- Approximating local gradients in image is basis of many classical edge-detection operators
- Main differences?
  - Type of filter used to estimate gradient components
  - How gradient components are combined
- We are typically interested in
  - Local edge direction
  - Local edge magnitude

Partial Image Derivatives

- Partial derivatives of images replaced by finite differences

$$\Delta_x f = f(x,y) - f(x-1,y)$$
$$\Delta_y f = f(x,y) - f(x,y-1)$$

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

- Alternatives are:

$$\Delta_{2x} f = f(x+1,y) - f(x-1,y)$$
$$\Delta_{2y} f = f(x,y+1) - f(x,y-1)$$

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

- Robert's gradient

$$\Delta_+ f = f(x+1,y+1) - f(x,y) \rightarrow \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$\Delta_- f = f(x,y+1) - f(x+1,y) \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

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

Prewitt

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

Sobel

Using Averaging with Derivatives

- Finite difference operator is sensitive to noise
- Derivates more robust if derivative computations are averaged in a neighborhood
- Prewitt operator: derivative in *x*, then average in *y*

$$H_x^P = \frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} * [0.5 \quad 0 \quad -0.5] = \frac{1}{6}\begin{bmatrix}1 & 0 & -1\\1 & 0 & -1\\1 & 0 & -1\end{bmatrix}$$

Average in *y* direction

Derivative in *x* direction

Note: Filter kernel is flipped in convolution

- *y*-derivative kernel, H_y^P defined similarly

Sobel operator

- Similar to Prewitt, but averaging kernel is higher in middle

$$H_x^S = \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} * [0.5 \quad 0 \quad -0.5] = \frac{1}{8}\begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix}$$

$$H_y^S = \frac{1}{4}[1 \quad 2 \quad 1] * \begin{bmatrix}0.5\\0\\-0.5\end{bmatrix} = \frac{1}{8}\begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix}$$

Average in *x* direction

Derivative in *y* direction

Note: Filter kernel is flipped in convolution

- Prewitt Operator

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Written in separable form →

$$H_x^P = \begin{bmatrix} 1 \\ \mathbf{1} \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} 1 & \mathbf{1} & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

- Sobel Operator

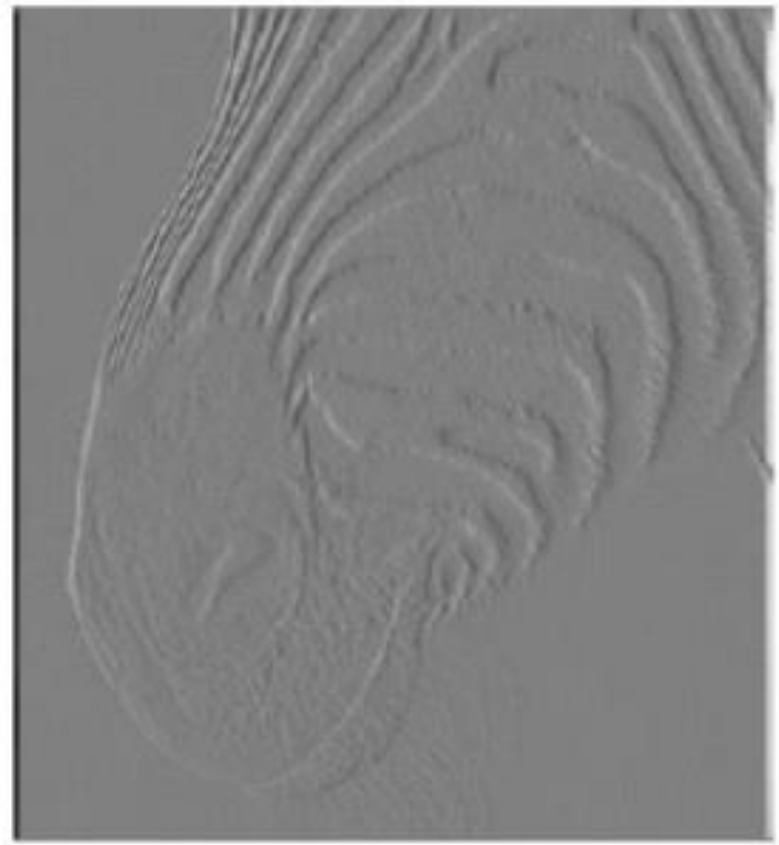
$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Improved Sobel filter

- Original Sobel filter relatively inaccurate
- Improved versions proposed by Jahne

$$H_x^{S'} = \frac{1}{32}\begin{bmatrix} -3 & 0 & 3 \\ -10 & \mathbf{0} & 10 \\ -3 & 0 & 3 \end{bmatrix} \quad \text{and} \quad H_y^{S'} = \frac{1}{32}\begin{bmatrix} -3 & -10 & -3 \\ 0 & \mathbf{0} & 0 \\ 3 & 10 & 3 \end{bmatrix}$$





Scaling Edge Components

- Estimates of local gradient components obtained from filter results by appropriate scaling

Scaling factor for Prewitt operator

$$\nabla I(u,v) \approx \frac{1}{6} \cdot \begin{bmatrix} (I * H_x^P)(u,v) \\ (I * H_y^P)(u,v) \end{bmatrix}$$

Scaling factor for Sobel operator

$$\nabla I(u,v) \approx \frac{1}{8} \cdot \begin{bmatrix} (I * H_x^S)(u,v) \\ (I * H_y^S)(u,v) \end{bmatrix}$$

Gradient-Based Edge Detection

- Compute image derivatives by convolution

$$D_x(u,v) = H_x * I \quad \text{and} \quad D_y(u,v) = H_y * I$$

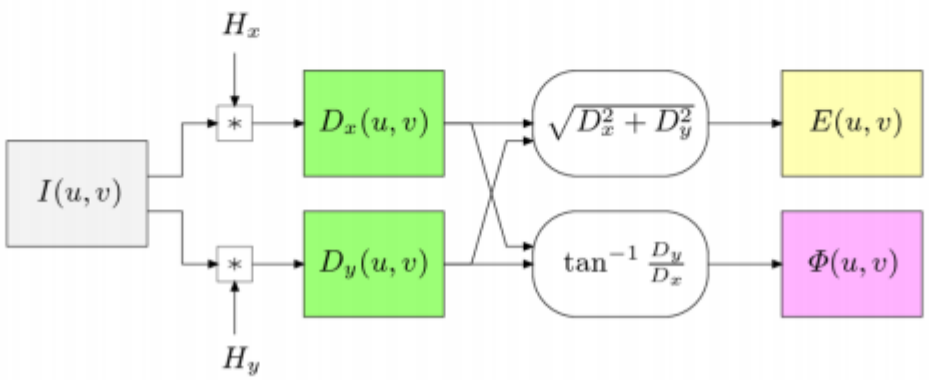
Scaled Filter results

- Compute edge gradient magnitude

$$E(u,v) = \sqrt{\left(D_x(u,v)\right)^2 + \left(D_y(u,v)\right)^2}$$

- Compute edge gradient direction

$$\Phi(u,v) = \tan^{-1}\left(\frac{D_y(u,v)}{D_x(u,v)}\right) = \mathrm{ArcTan}\left(D_x(u,v), D_y(u,v)\right)$$



Typical process of Gradient based edge detection

Difference operators for 2D

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0	0	−1
0	1	1	0

Roberts

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

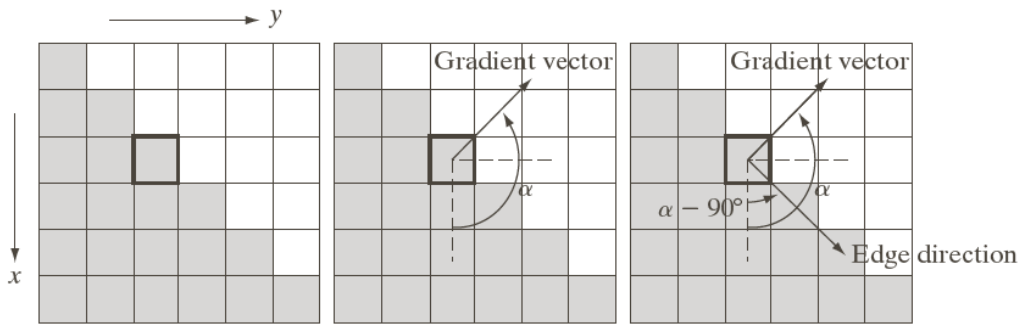
Prewitt

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Sobel

a
b c
d e
f g

FIGURE 10.14
A 3 × 3 region of an image (the z's are intensity values) and various masks used to compute the gradient at the point labeled z_5.



a b c

FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

Adapted from Gonzales and Woods

Gradient-Based Edge Detection

- After computing gradient magnitude and orientation then what?
- Mark points where gradient magnitude is large wrt neighbors



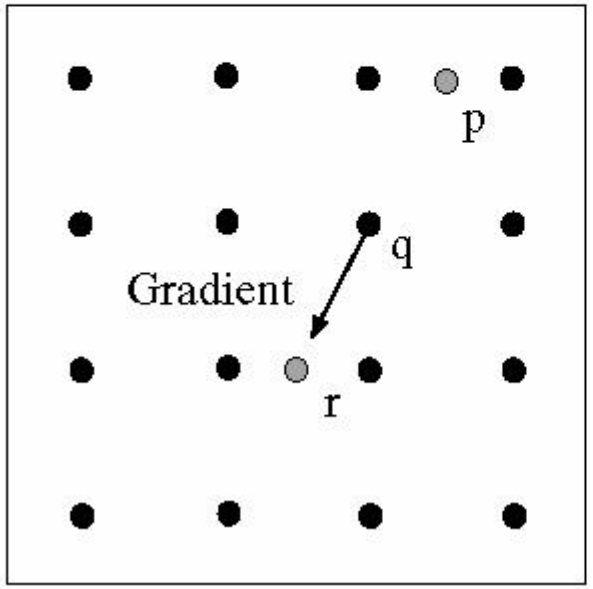
Non-Maxima Suppression

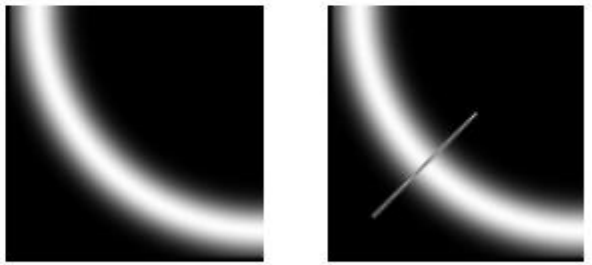
- Retain a point as an edge point if:
  - Its gradient magnitude is higher than a threshold
  - Its gradient magnitude is a local maxima in gradient direction

Simple thresholding will compute thick edges

Non-Maxima Suppression

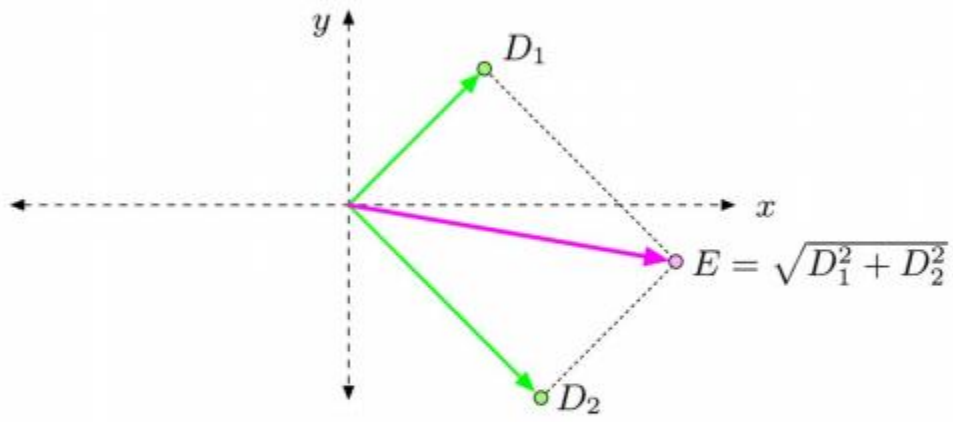
- A maxima occurs at *q*, if its magnitude is larger than those at *p* and *r*





Roberts Edge Operators

- Estimates directional gradient along 2 image diagonals
- Edge strength E(u,v): length of vector obtained by adding 2 orthogonal gradient components $D_1(u,v)$ and $D_2(u,v)$

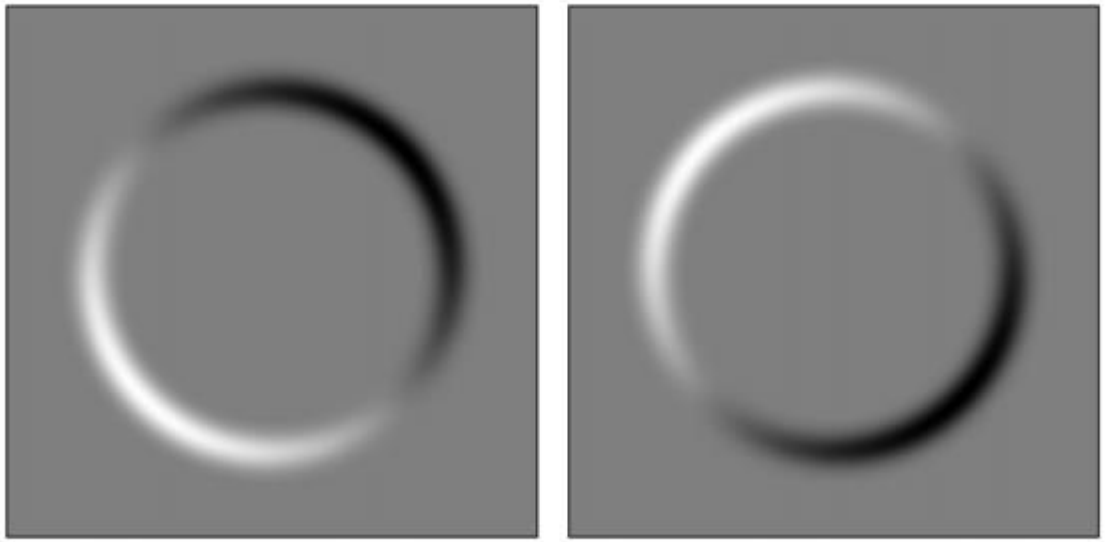


- Filters for edge components

$$H_1^R = \begin{bmatrix} 0 & \mathbf{1} \\ -1 & 0 \end{bmatrix} \quad \text{and} \quad H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & \mathbf{1} \end{bmatrix}$$

Roberts Edge Operators

- Diagonal gradient components produced by 2 Robert filters

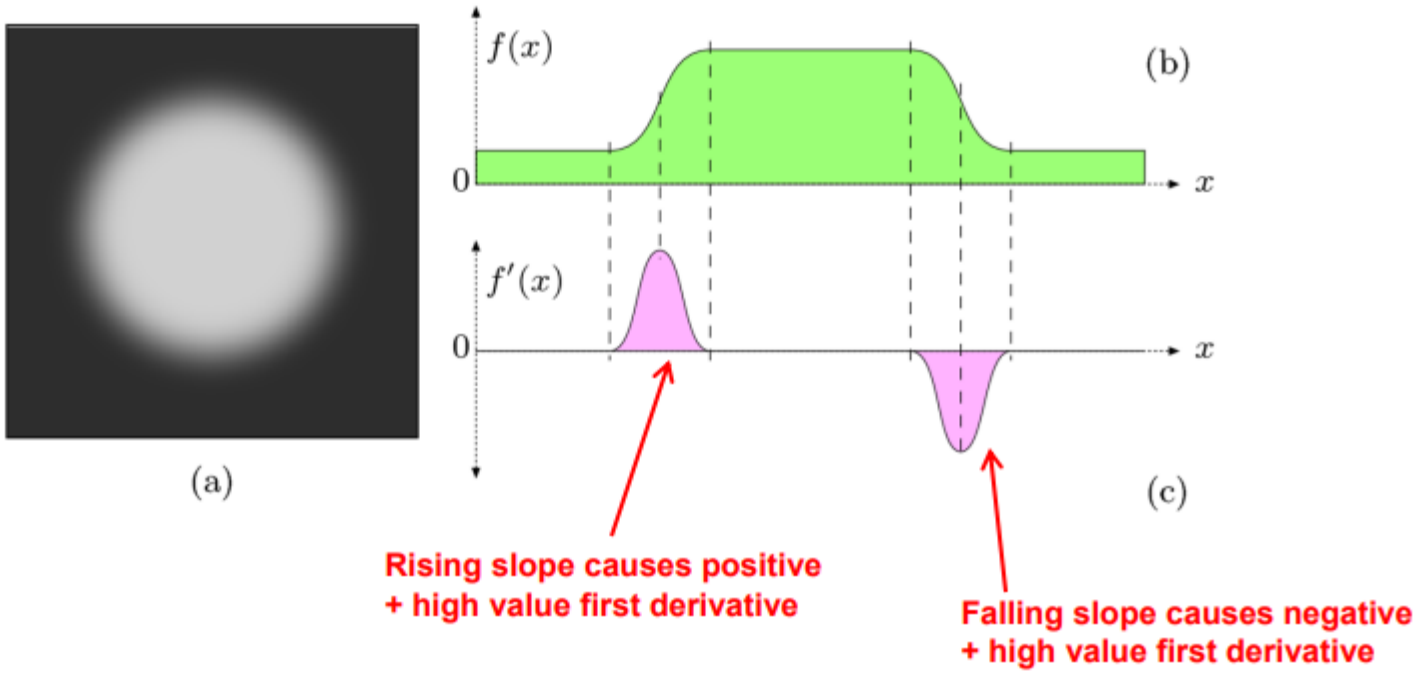


$D_1 = I * H_1^R$ $D_2 = I * H_2^R$

Recall: Characteristics of an Edge

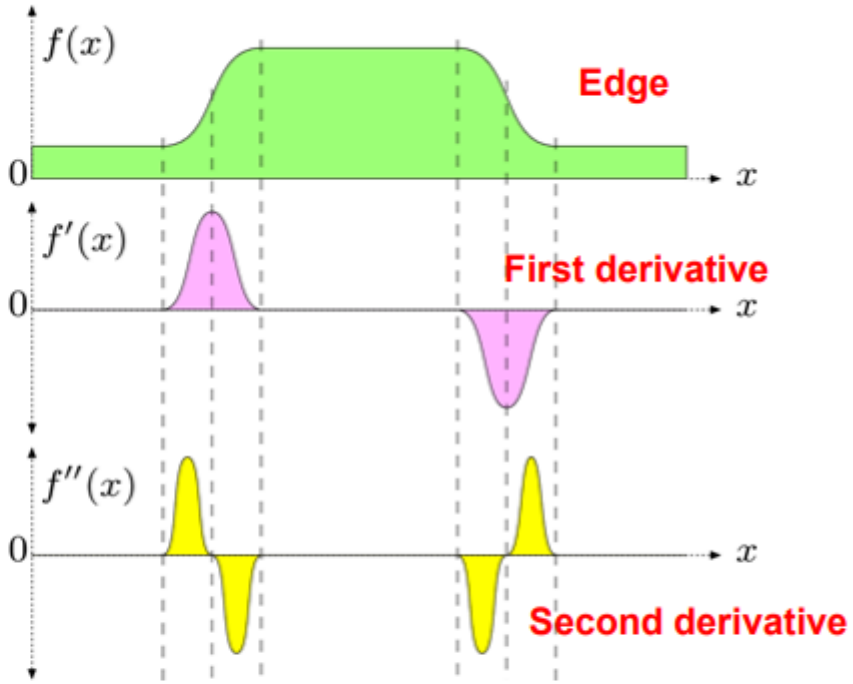
- Real (non-ideal) edge is a slightly blurred step function
- Edges can be characterized by high value first derivative

$$f'(x) = \frac{df}{dx}(x)$$



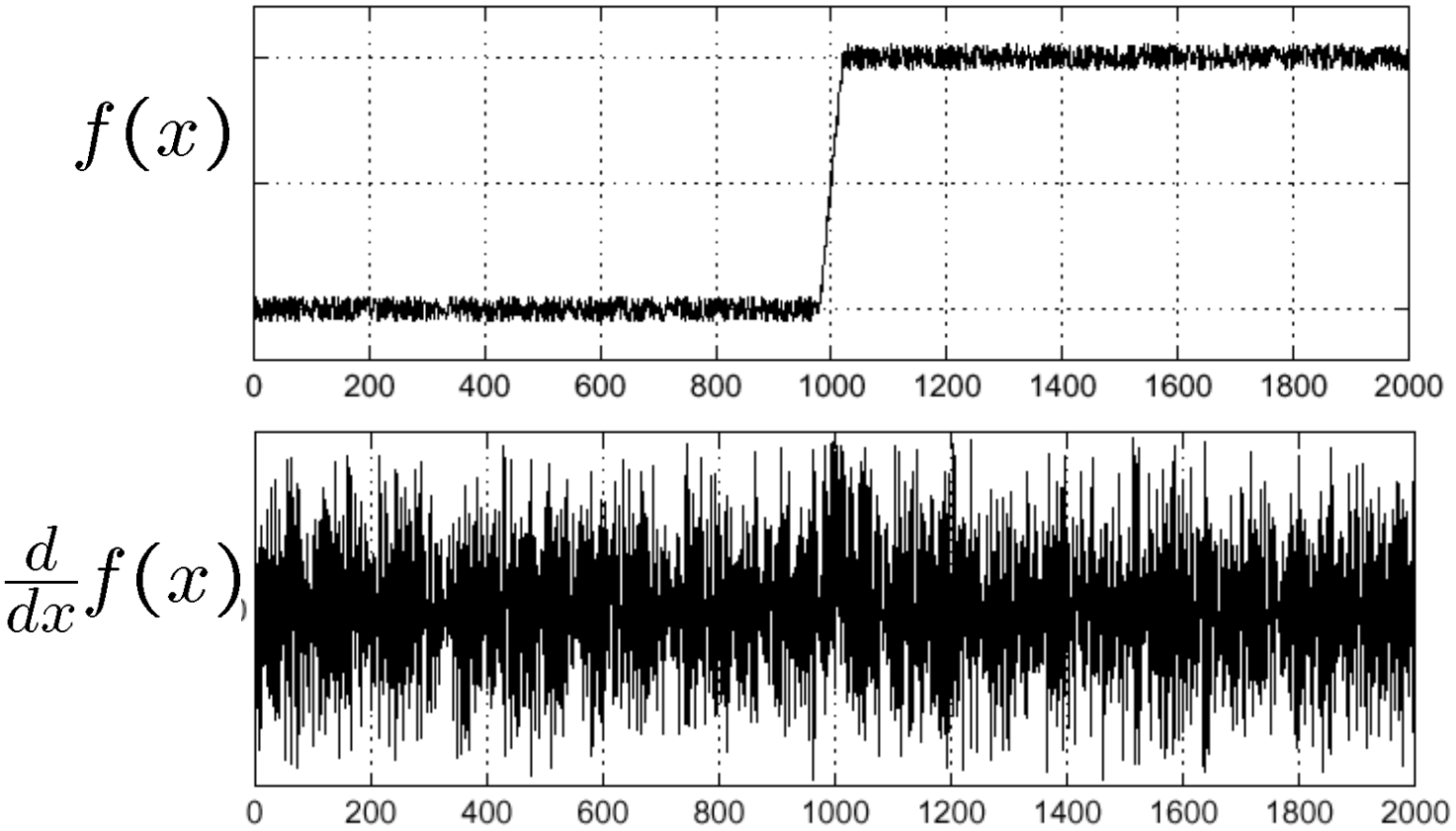
Other Edge Operators

- **Problem with edge operators based on first derivatives:**
  - Edge is proportional to underlying intensity transition
  - Edges may be difficult to localize precisely
- Solution? Use second derivative
- **Recall:** An edge corresponds to a zero crossing of the 2nd derivative
- Since 2nd derivatives amplify image noise, pre-smoothing filters used first



Difference operators under noise

Consider a single row or column of the image.
Where is the edge?



Adapted from Steve Seitz

Difference operators under noise

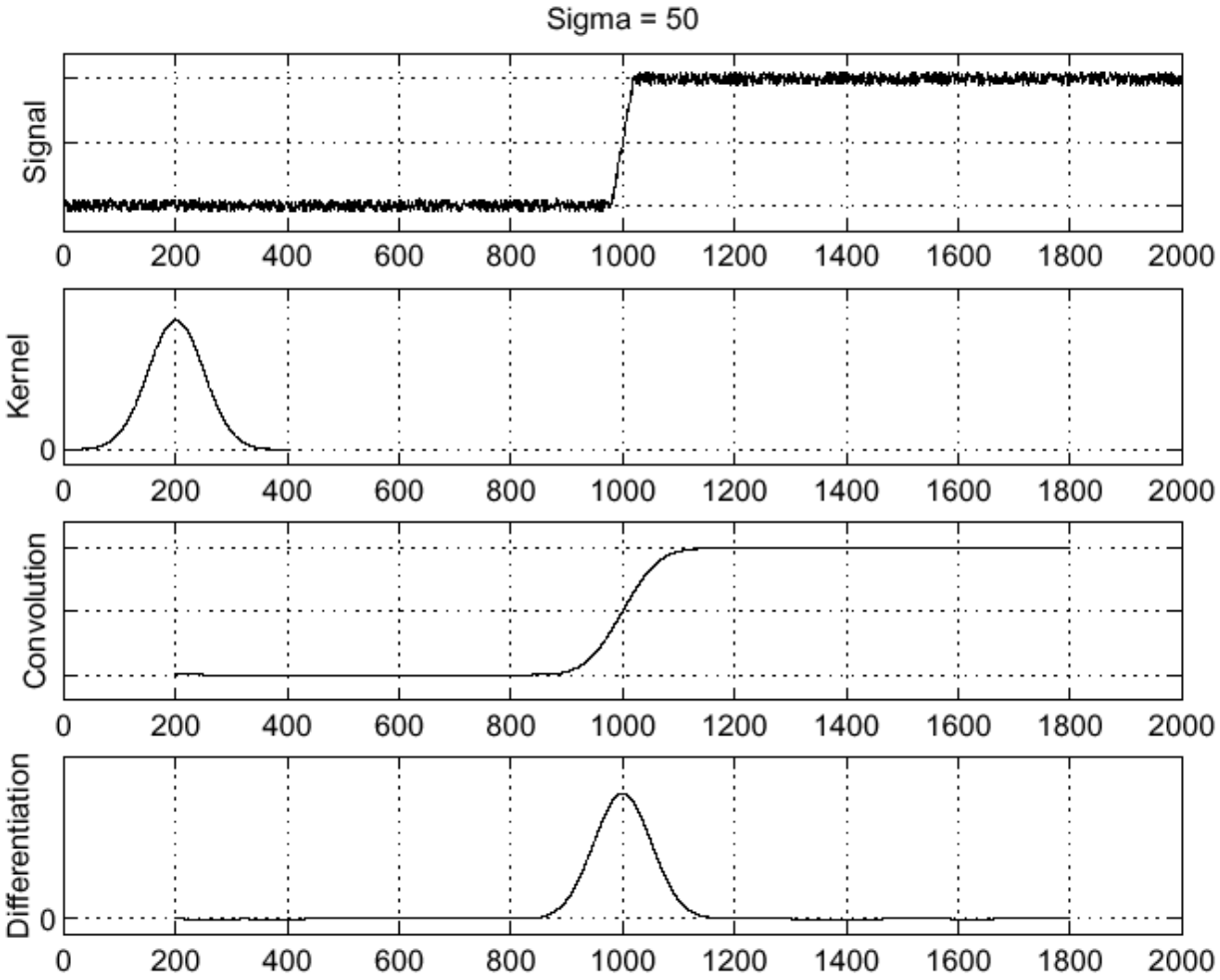
Solution is to smooth first:

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Adapted from Steve Seitz

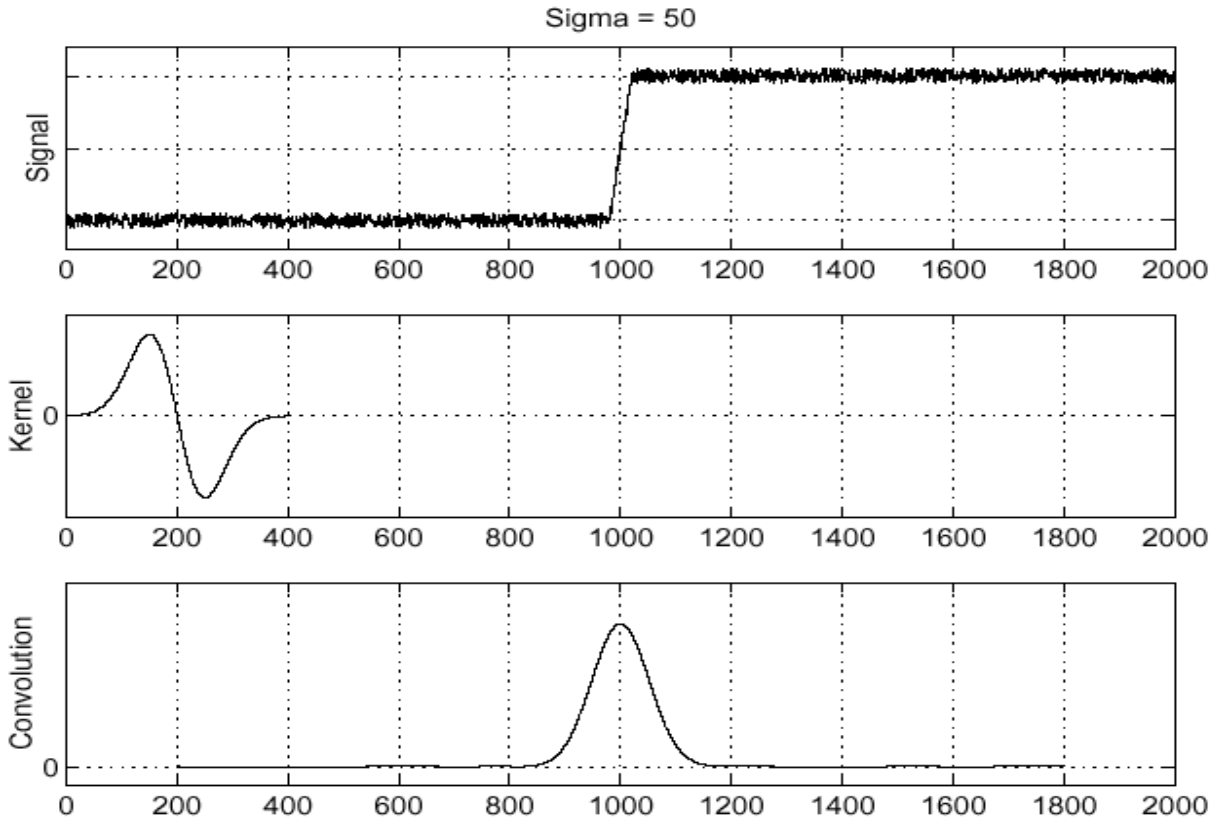
Difference operators under noise

Differentiation property of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



Adapted from Steve Seitz

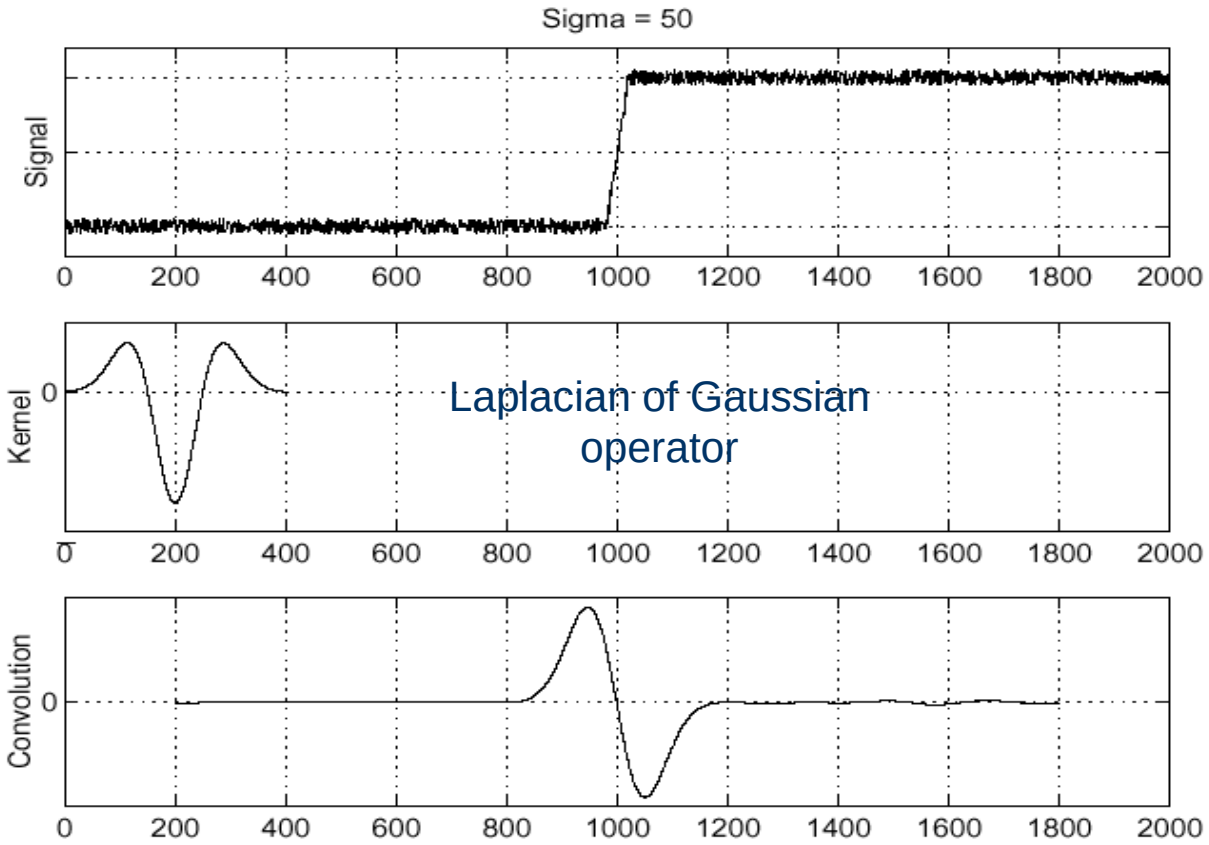
Difference operators under noise

Consider: $\frac{\partial^2}{\partial x^2}(h \star f)$

f

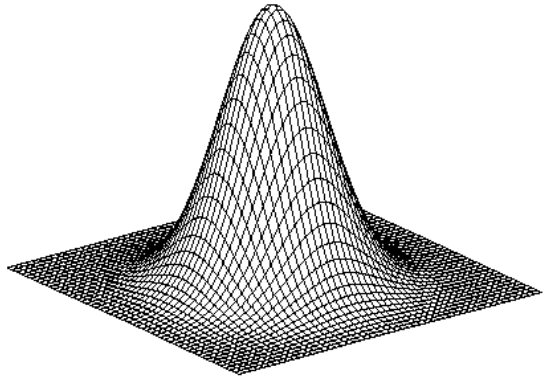
$\frac{\partial^2}{\partial x^2} h$

$(\frac{\partial^2}{\partial x^2} h) \star f$



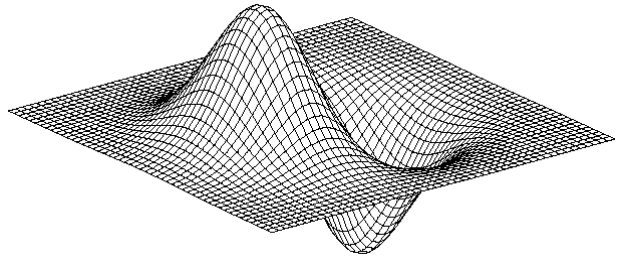
Adapted from Steve Seitz

Edge detection filters for 2D



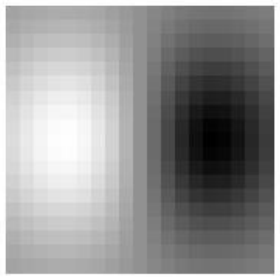
Gaussian

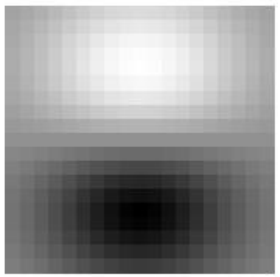
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

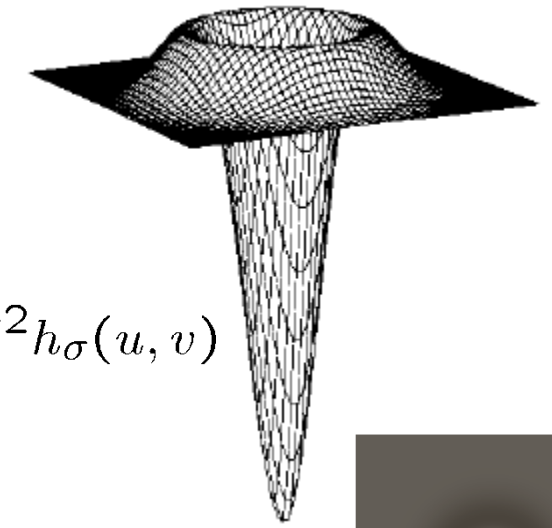
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$





Laplacian of Gaussian

$$\nabla^2 h_\sigma(u,v)$$



Adapted from Steve Seitz, U of Washington

Difference operators for 2D

a b
c d

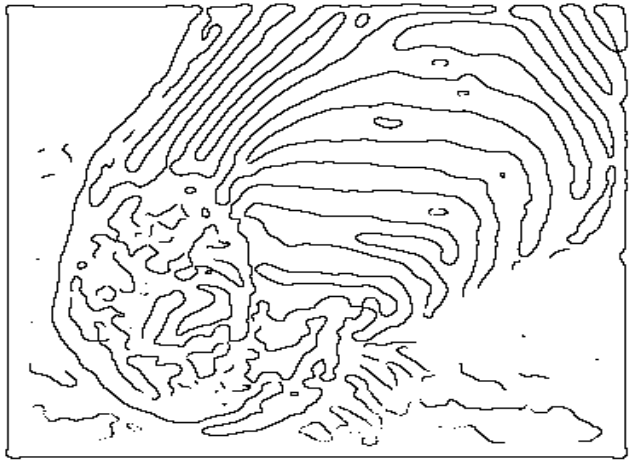
FIGURE 10.16
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) $|g_x|$, the component of the gradient in the x-direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the mask in Fig. 10.14(g). (d) The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

Difference operators for 2D

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 filter prior to edge detection.

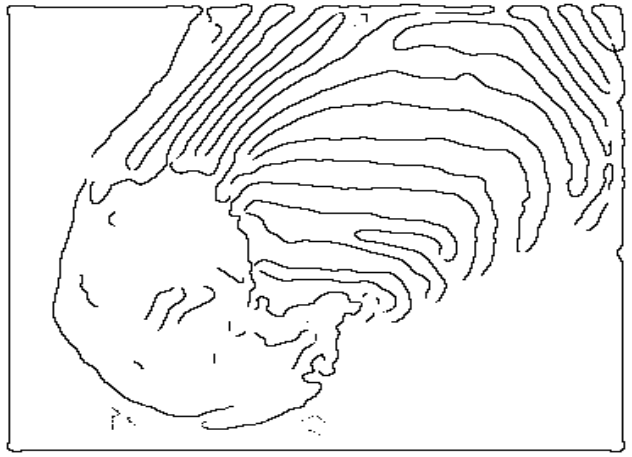
Difference operators for 2D



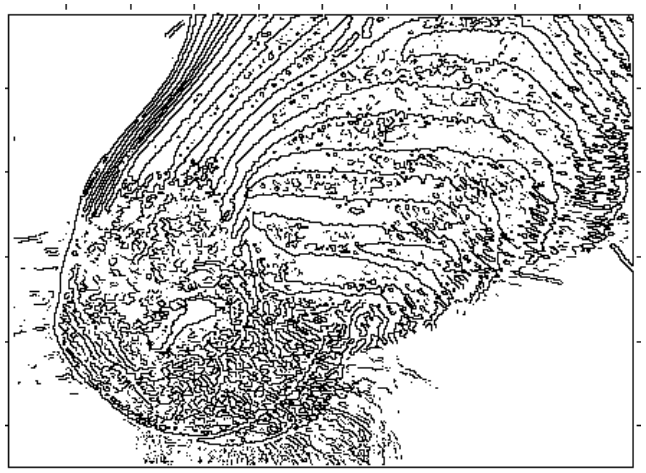
Threshold=1

sigma=4

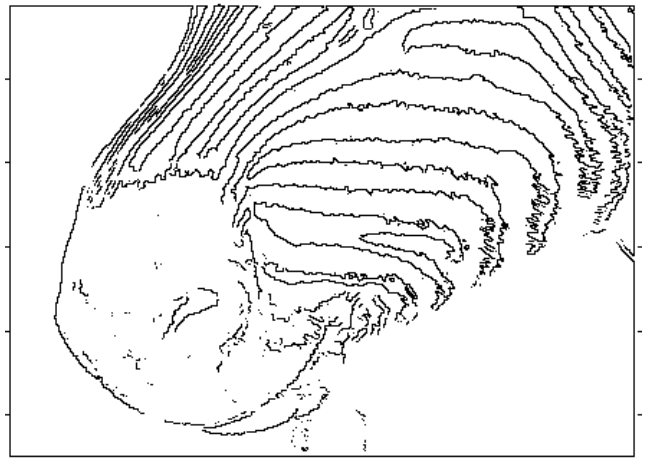
Laplacian of Gaussian
zero crossings



Threshold=4



sigma=2



Adapted from David Forsyth, UC Berkeley

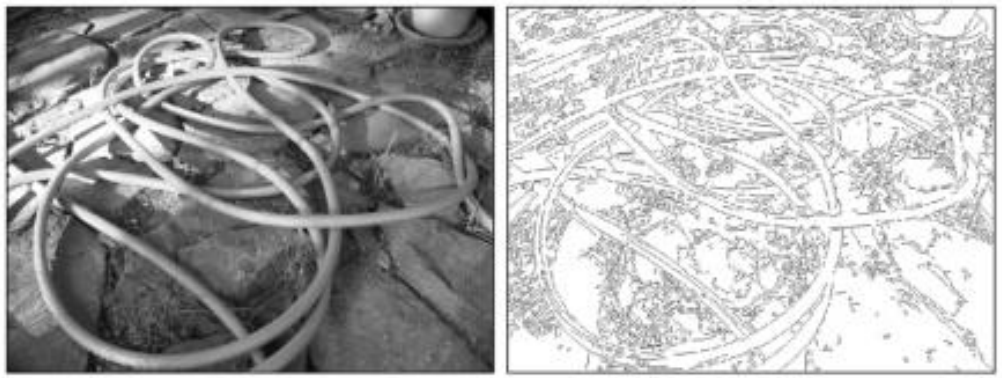
Edges at different scales

- Simple edge operators deviate from human perception in 2 main ways:
  - Edge operators respond to local intensity differences while human visual system **extends** edges across areas of minimal or vanishing contrast
  - Edges exist at multiple scales
- **Hierarchical or pyramid techniques:**
  - For each image position (u,v), apply edge detection filters at multiple scales
  - Use most dominant edge/scale detected (if any)

Canny Edge Detector

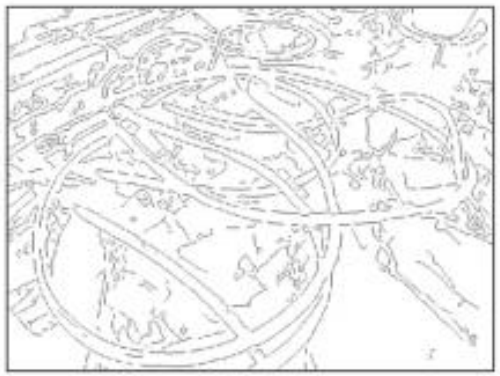
- Popular edge detector that operates at different scales, then combines results into common edge map. Tries to:
  1. Minimize number of false edge points
  2. Achieve good localization of edges
  3. Deliver only a single mark on each edge
- Essentially gradient based using zero crossings of second derivative
- Typically, a single scale implementation (1 image) used with adjustable filter radius (smoothing parameter σ)

Canny Edge Detector

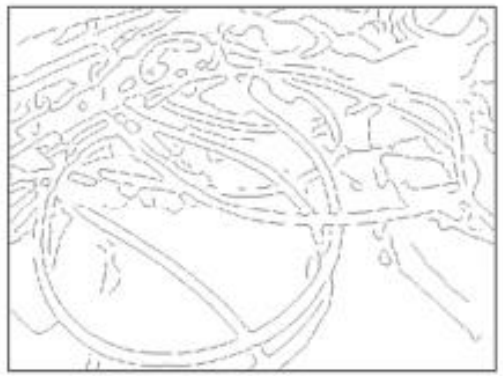


Original

$\sigma = 1.0$

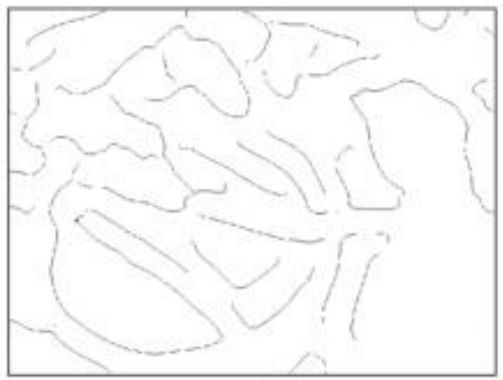


$\sigma = 2.0$



$\sigma = 4.0$

$\sigma = 8.0$



$\sigma = 16.0$

Resulting edge maps for different settings of the smoothing (scale) parameter σ

Original

Roberts

Prewitt

Sobel

Laplacian of Gaussian

Canny ($\sigma = 1.0$)

Edge detection

- Three fundamental steps in edge detection:
  1. Image smoothing: to reduce the effects of noise.
  2. Detection of edge points: to find all image points that are potential candidates to become edge points.
  3. Edge localization: to select from the candidate edge points only the points that are true members of an edge.

Canny edge detector

- Canny defined three objectives for edge detection:
  1. Low error rate: All edges should be found and there should be no spurious responses.
  2. Edge points should be well localized: The edges located must be as close as possible to the true edges.
  3. Single edge point response: The detector should return only one point for each true edge point. That is, the number of local maxima around the true edge should be minimum.

Canny edge detector

1. Smooth the image with a Gaussian filter with spread σ.
2. Compute gradient magnitude and direction at each pixel of the smoothed image.
3. Zero out any pixel response less than or equal to the two neighboring pixels on either side of it, along the direction of the gradient (non-maxima suppression).
4. Track high-magnitude contours using thresholding (hysteresis thresholding).
5. Keep only pixels along these contours, so weak little segments go away.

Canny edge detector

Original image (Lena)

Adapted from Steve Seitz, U of Washington

Canny edge detector

Magnitude of the gradient

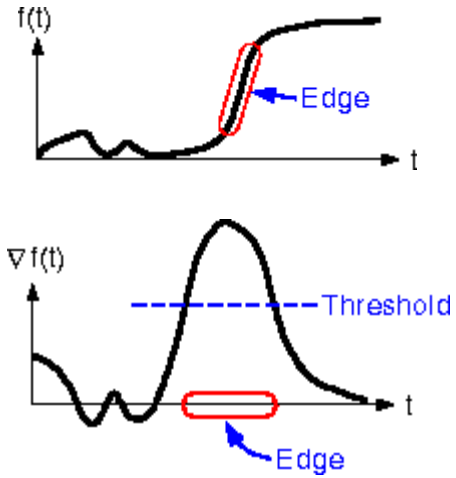
Adapted from Steve Seitz, U of Washington

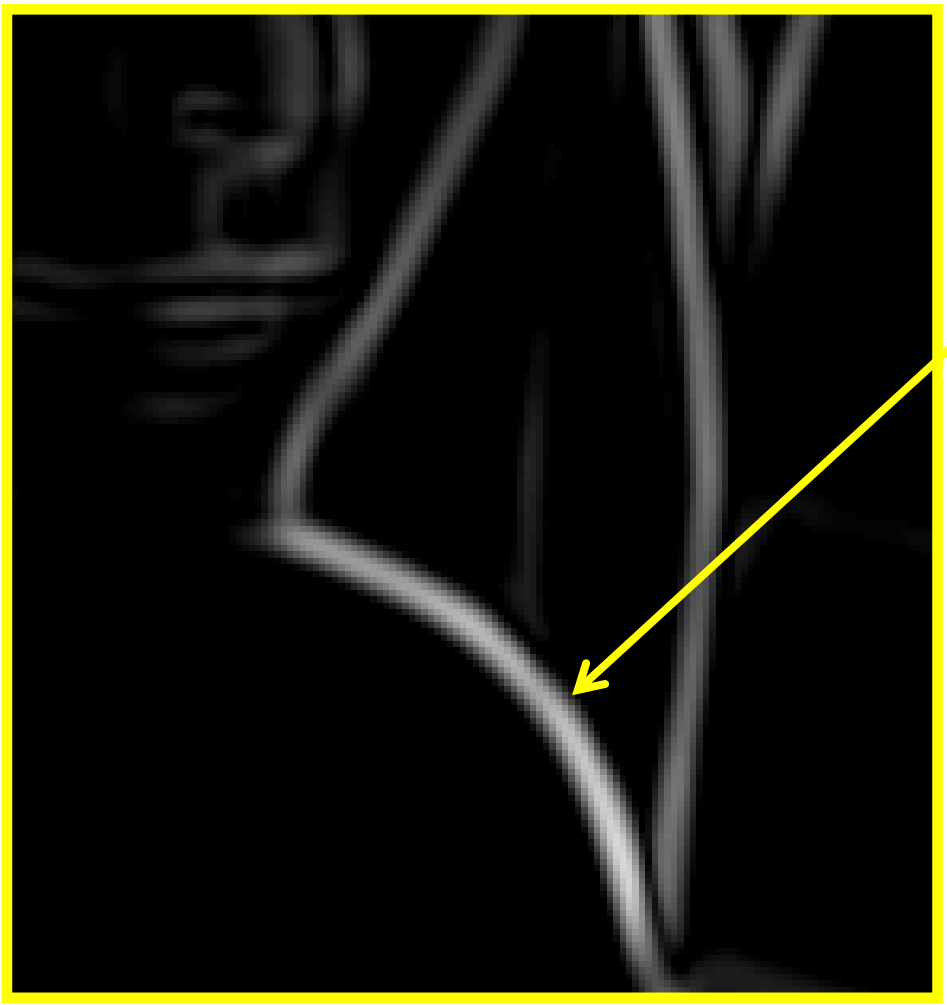
Canny edge detector

Thresholding

Adapted from Steve Seitz, U of Washington

Canny edge detector



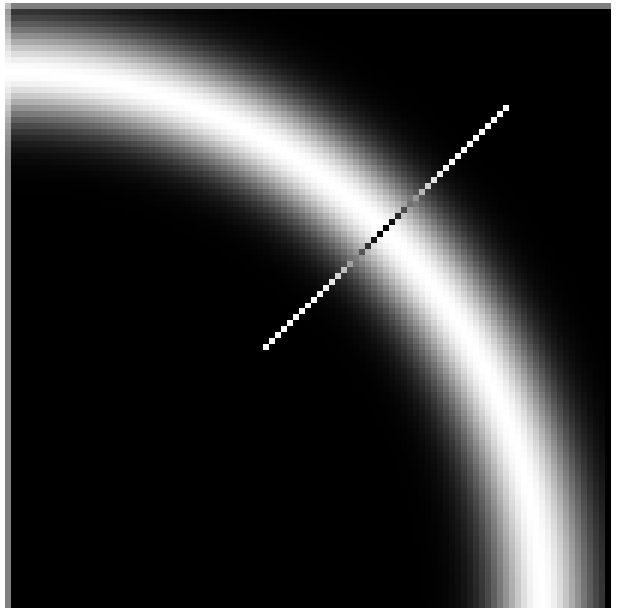


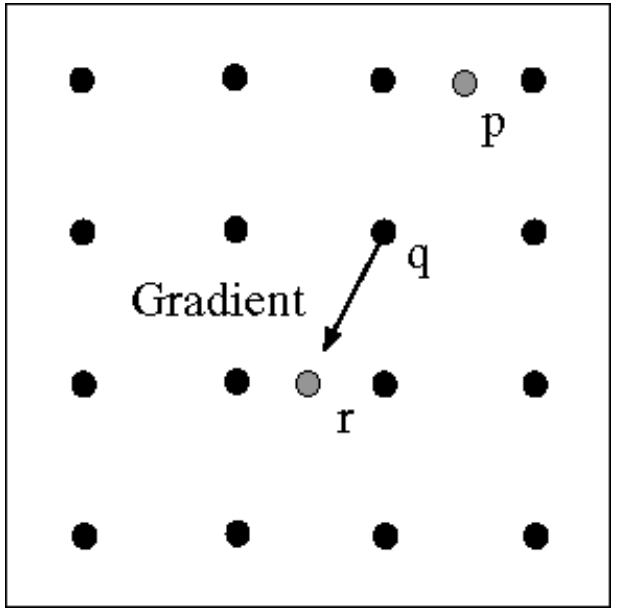
How to turn these thick regions of the gradient into curves?

Adapted from Steve Seitz, U of Washington

Canny edge detector

- Non-maxima suppression:
  - Check if pixel is local maximum along gradient direction.
  - Select single max across width of the edge.
  - Requires checking interpolated pixels p and r.
  - This operation can be used with any edge operator when thin boundaries are wanted.





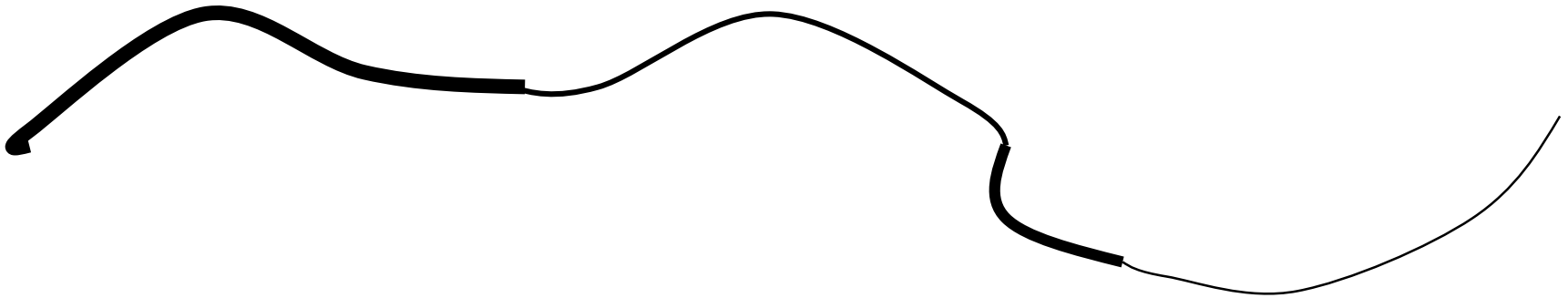
Canny edge detector

Problem: pixels along this edge did not survive the thresholding

Adapted from Steve Seitz, U of Washington

Canny edge detector

- Hysteresis thresholding:
  - Use a high threshold to start edge curves, and a low threshold to continue them.



Canny edge detector

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \quad I_y = G_\sigma^y * I$$

2. Compute magnitude of gradient at every pixel

$$M(x, y) = |\nabla I| = \sqrt{I_x^2 + I_y^2}$$

3. Eliminate those pixels that are not local maxima of the magnitude in the direction of the gradient

4. Hysteresis Thresholding

   - Select the pixels such that $M > T_h$ (high threshold)
   - Collect the pixels such that $M > T_l$ (low threshold) that are neighbors of already collected edge points

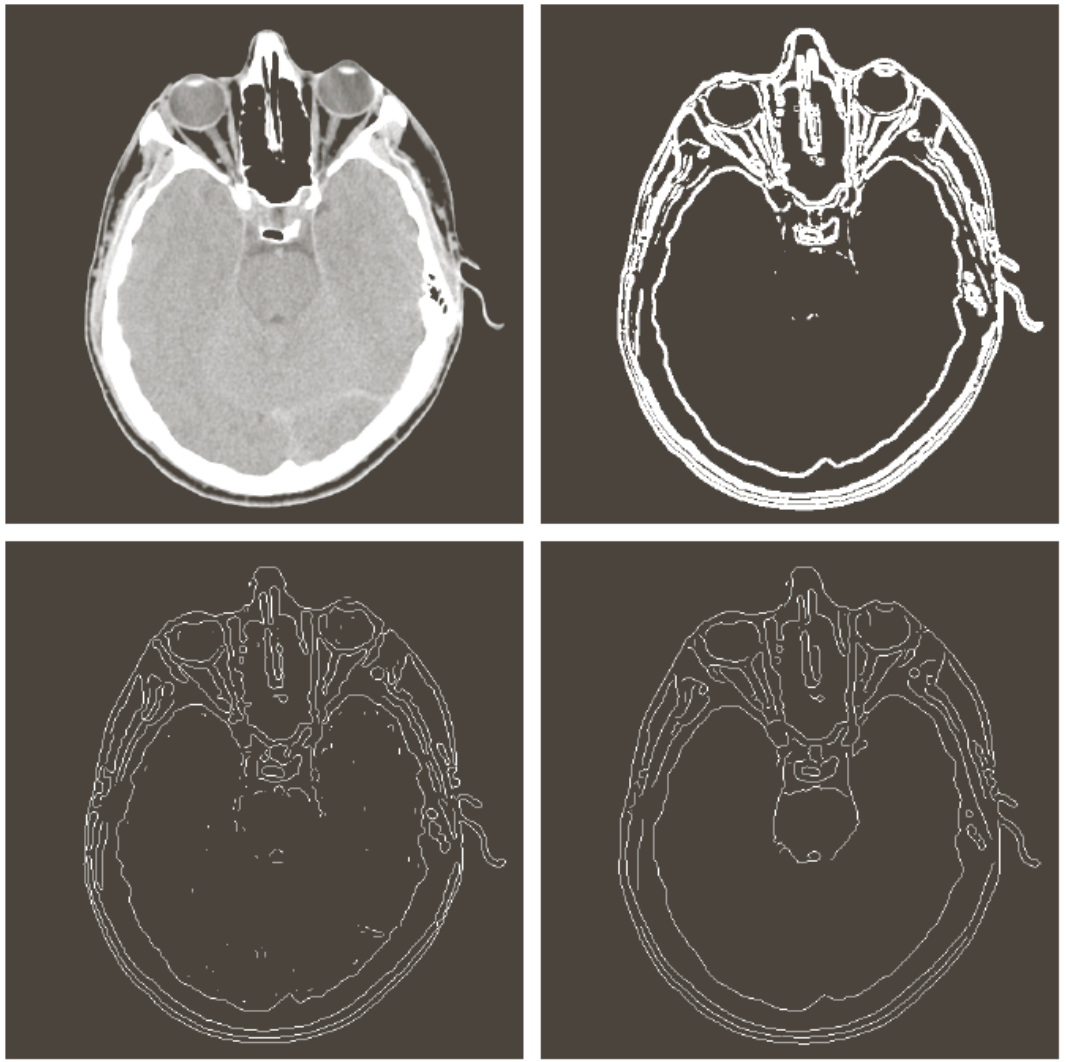
Adapted from Martial Hebert, CMU

Canny edge detector

a b
c d

FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

Canny edge detector



a b
c d

FIGURE 10.26
(a) Original head CT image of size 512 × 512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

Canny edge detector

- The Canny operator gives single-pixel-wide images with good continuation between adjacent pixels.
- It is the most widely used edge operator today; no one has done better since it came out in the late 80s. Many implementations are available.
- It is very sensitive to its parameters, which need to be adjusted for different application domains.

From Edges to Contours

- Edge detection yields at each image position (u,v)
  - Edge strength + orientation
- How can this information be used to detect larger image structures and contours?
- **Contour following:** Starting from image point with high edge strength, follow edge iteratively till the 2 traces meet and a closed contour is formed
- **Several obstacles make this impractical and rarely used:**
  - Edges may end in regions of vanishing intensity gradient
  - Crossing edges lead to ambiguities
  - Contours may branch into several directions

Edge Maps

- Usually after calculating edge strengths, we just make binary decision whether point is valid edge
- **Most common approach:** generate **edge map** by appying threshold (fixed or adaptive) to edge strengths calculated by edge operator
- In practice edge maps seldom contain perfect contours
- Edge map frequently has small, unconnected contour fragments interrupted by positions of insufficient strength