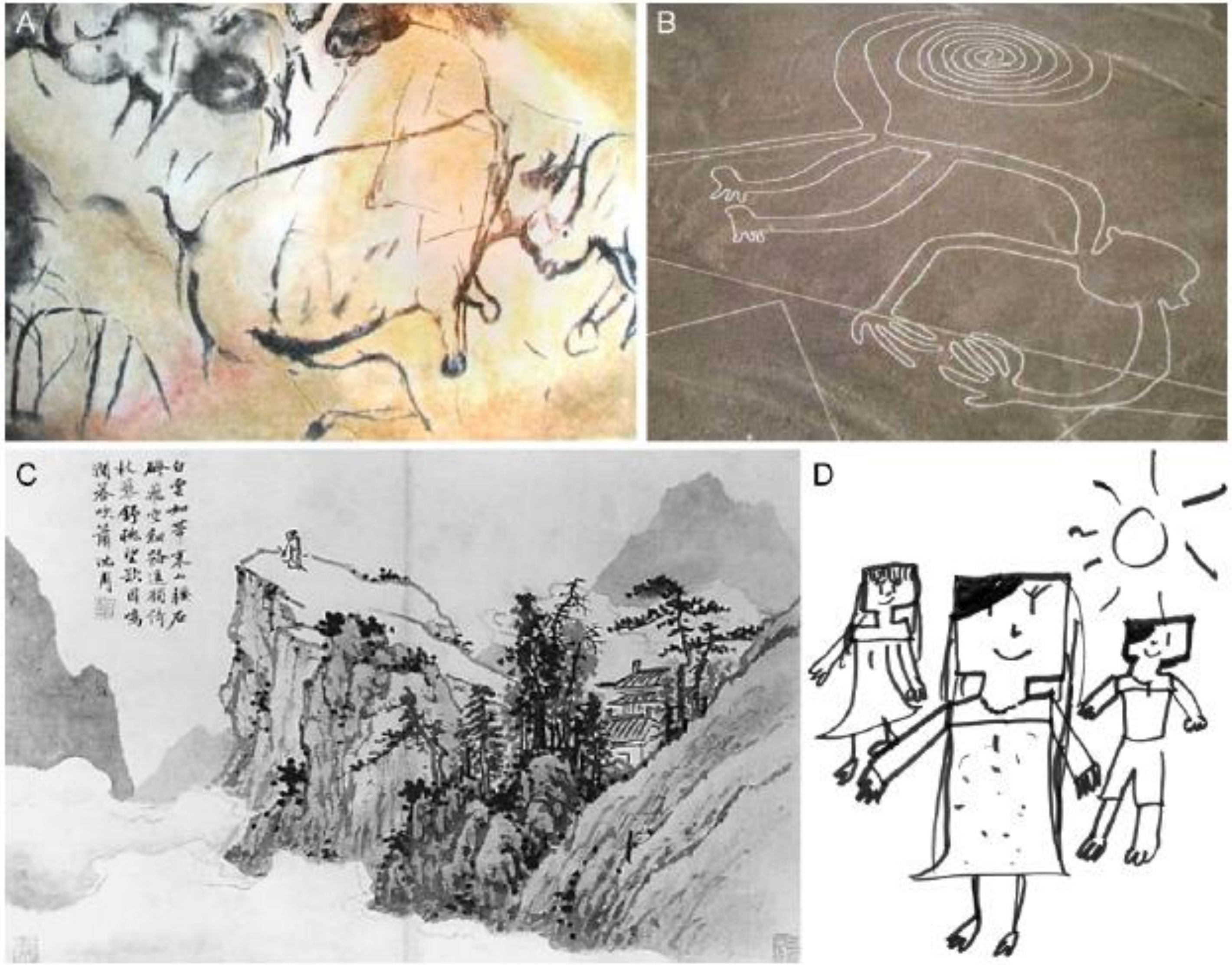
Edge Detection



(A) Cave painting at Chauvet, France, about 30,000 B.C.;
(B) Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;
(C) Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
(D) Line drawing by 7-year old I. Lleras (2010 A.D.).

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges

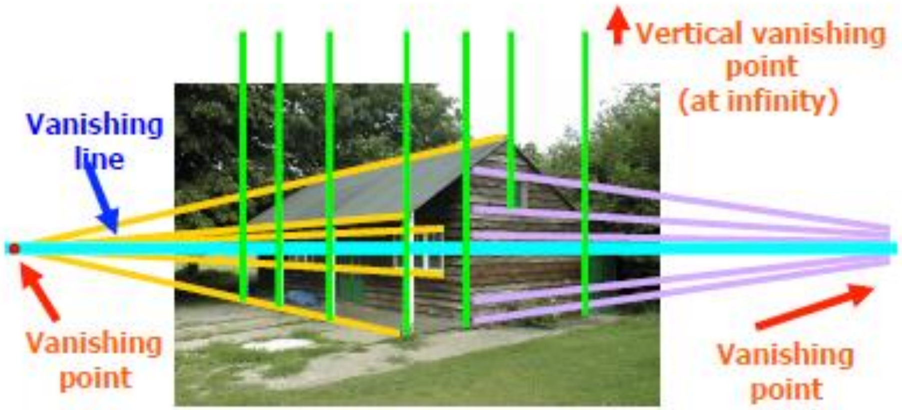
- **Ideal:** artist's line drawing

Source: D. Lowe

Why do we care about edges?

- Extract information, recognize objects

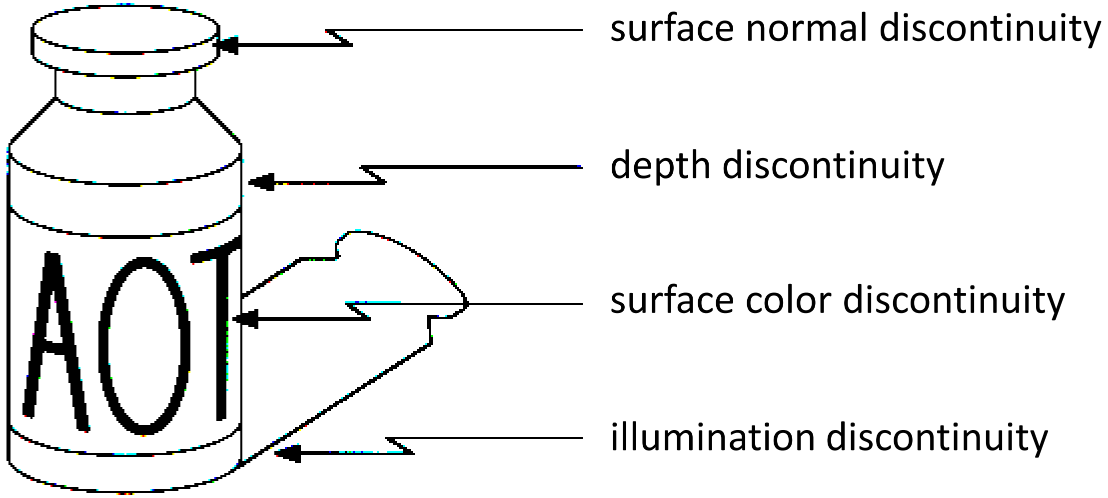
- Recover geometry and viewpoint



Source: J. Hayes

Origin of edges

- Edges are caused by a variety of factors:



Source: Steve Seitz

Origins of edges

surface normal discontinuity

depth discontinuity

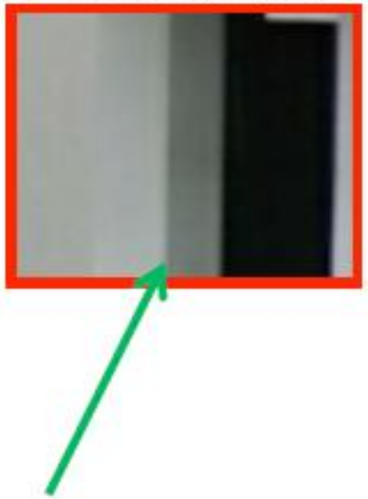
surface color discontinuity

illumination discontinuity

Source: D. Hoiem

Closeup of edges

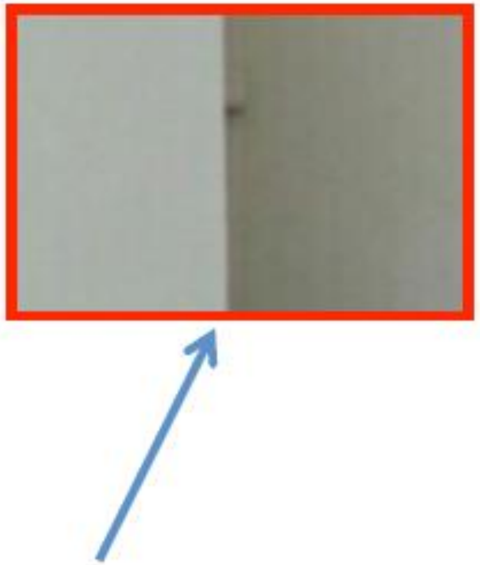
Surface normal discontinuity



Source: D. Hoiem

Closeup of edges

Depth discontinuity



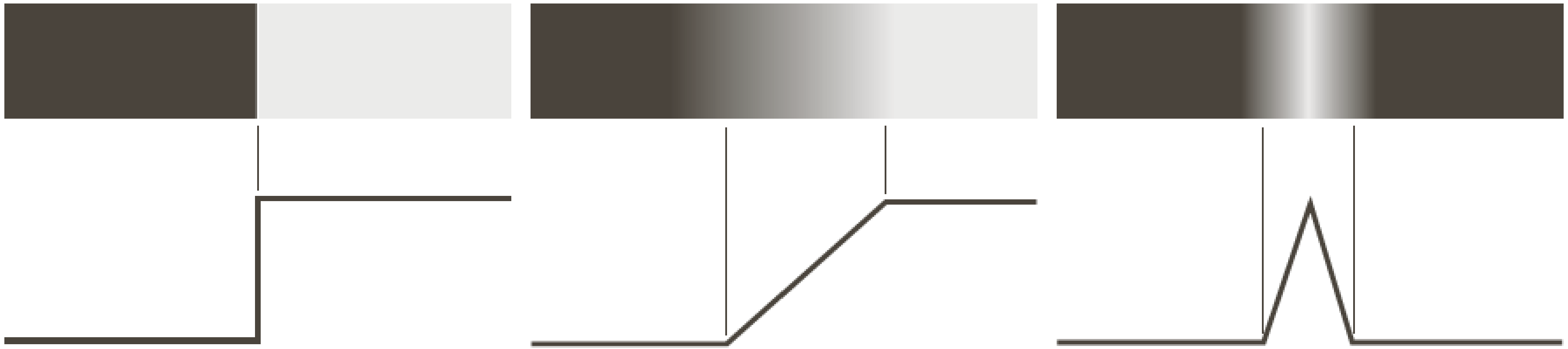
Source: D. Hoiem

Closeup of edges

Surface color discontinuity

Source: D. Hoiem

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.

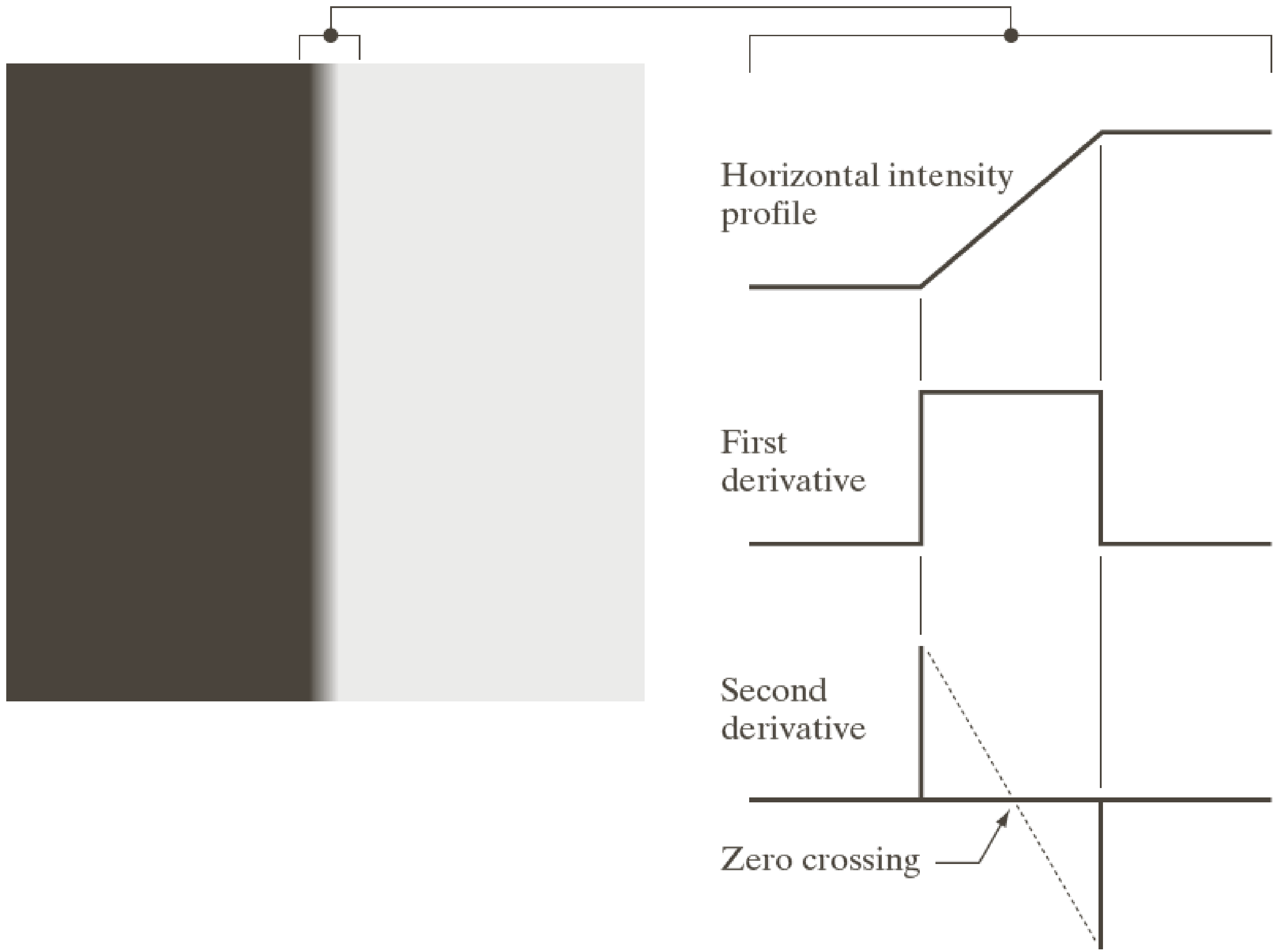
Background

- First-order derivative

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

- Second-order derivative

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



a b

FIGURE 10.10
(a) Two regions of constant intensity separated by an ideal vertical ramp edge.
(b) Detail near the edge, showing a horizontal intensity profile, together with its first and second derivatives.

Characteristics of First and Second Order Derivatives

First-order derivatives generally produce thicker edges in image

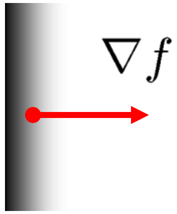
Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise

Second-order derivatives produce a double-edge response at ramp and step transition in intensity

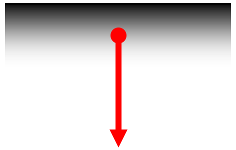
The sign of the second derivative can be used to determine whether a transition into an edge is from light to dark or dark to light

Image gradient

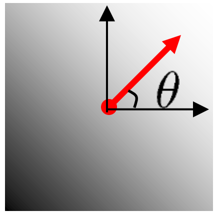
- The gradient of an image: $$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



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

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

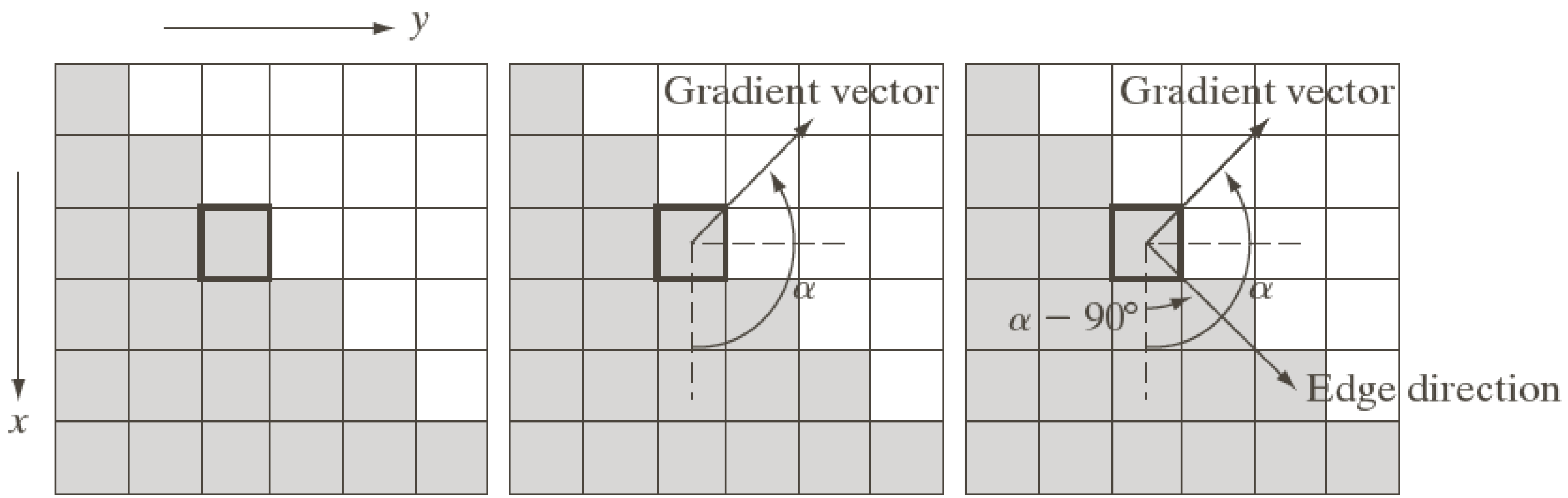
- How does this direction relate to the direction of the edge?

The gradient direction is given by $$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right)$$

The *edge strength* is given by the gradient magnitude

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

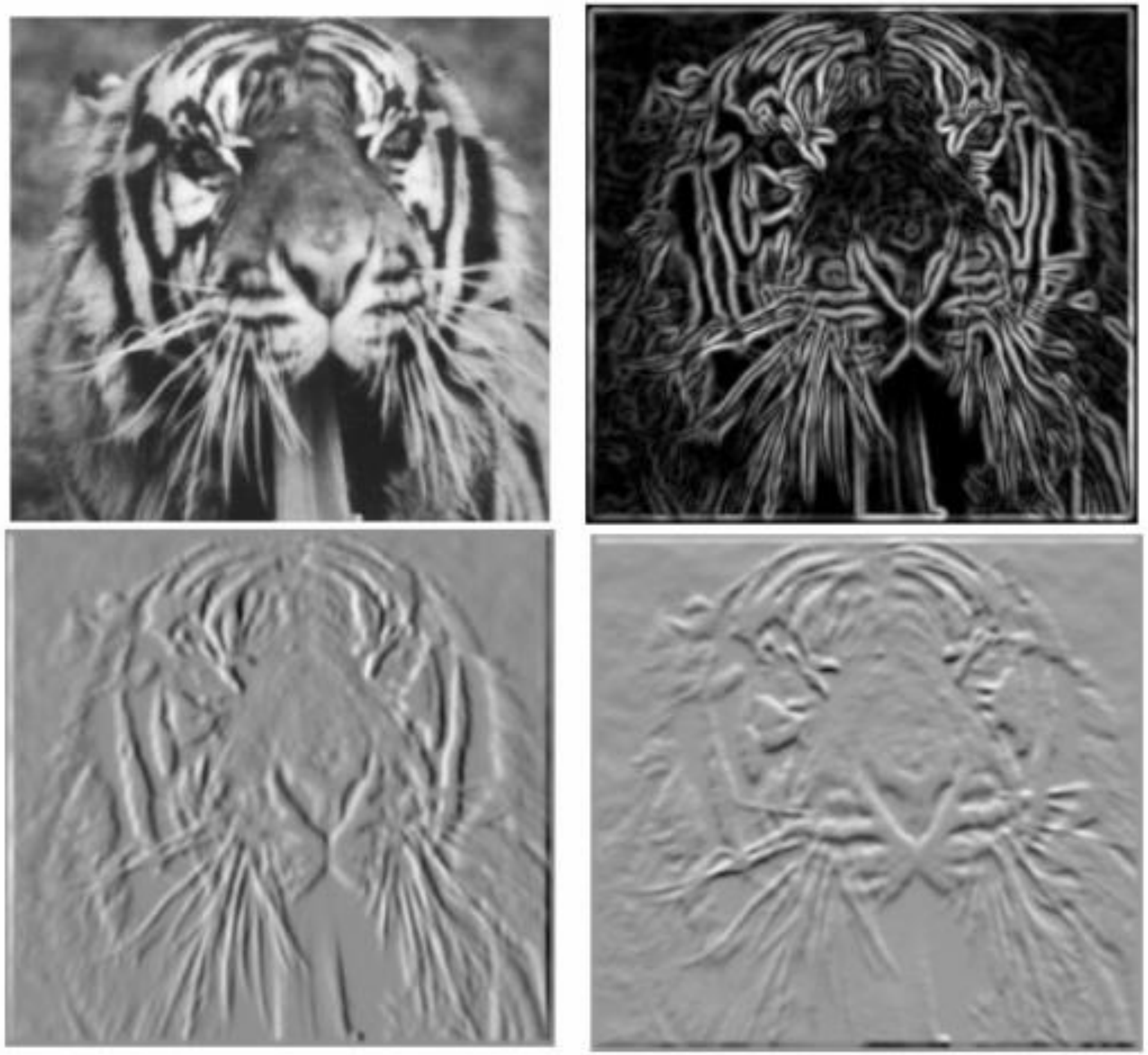
Source: Steve Seitz



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.

Finite differences: example



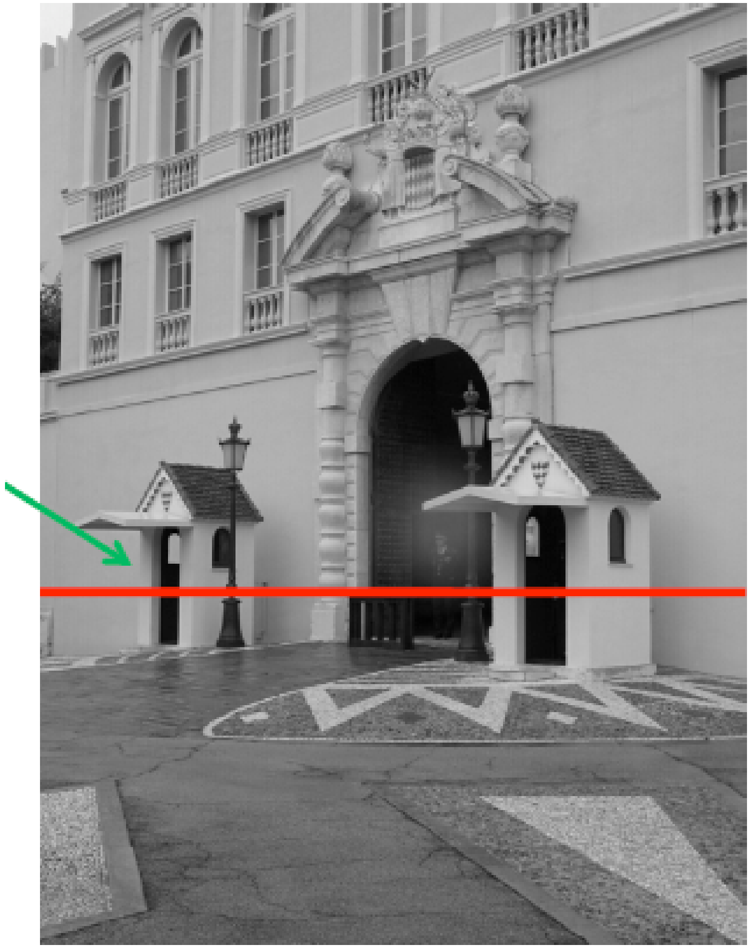
Original Image

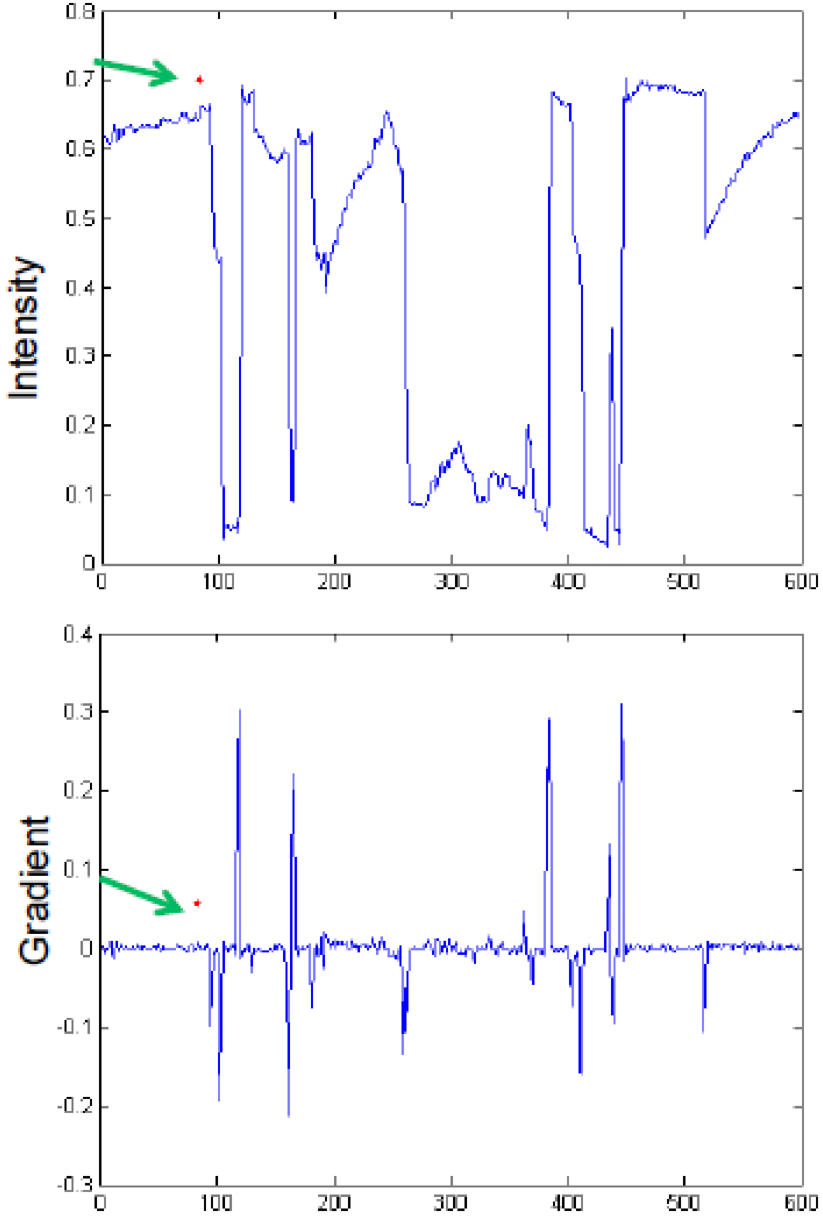
Gradient magnitude

x-direction

y-direction

Intensity profile

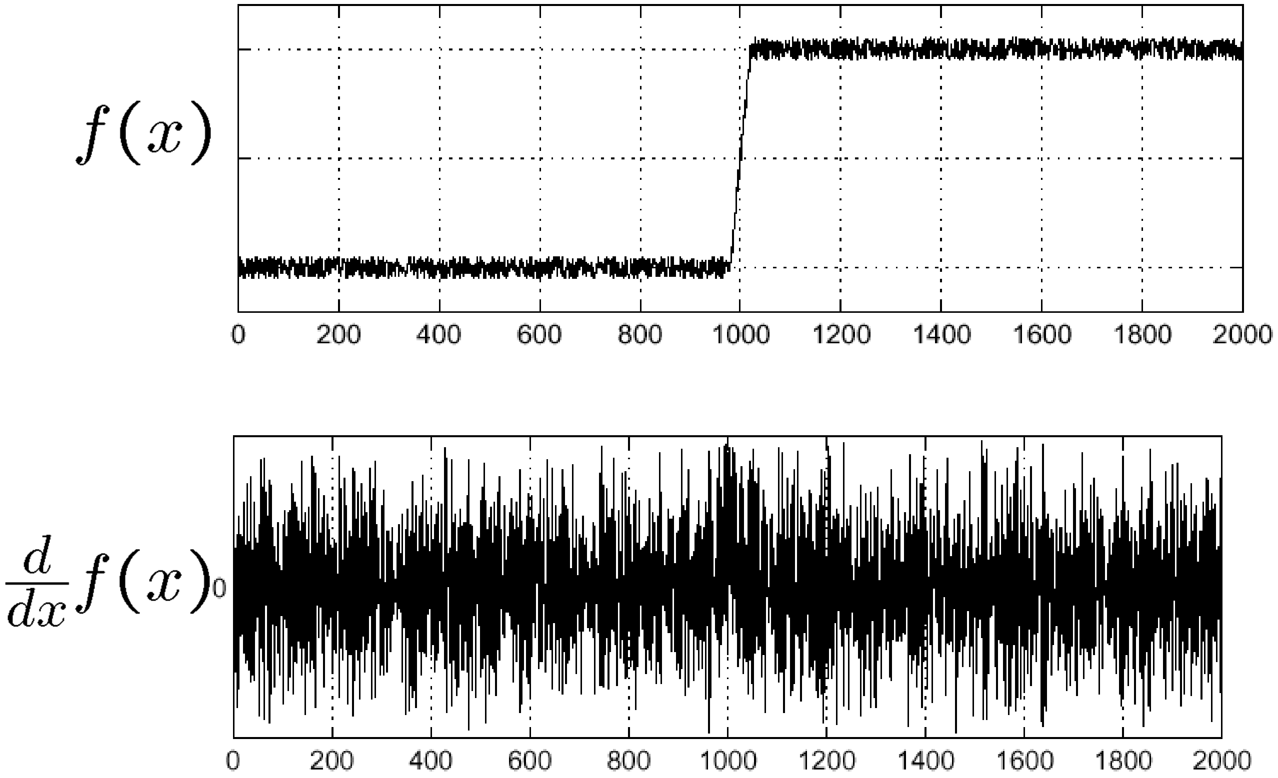




Source: D. Hoiem

Effects of noise

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



Where is the edge?

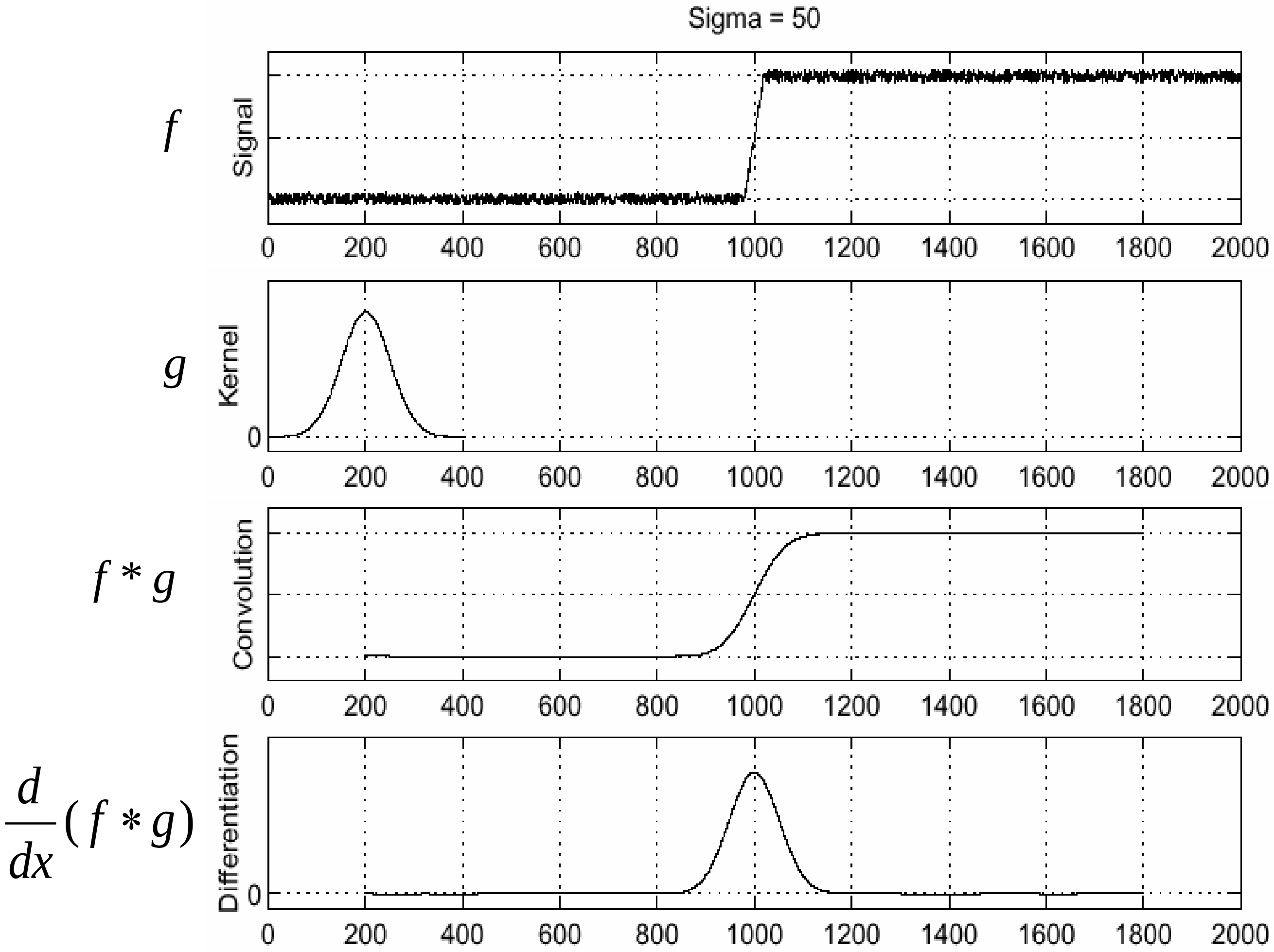
Source: S. Seitz

Effects of noise

- Finite 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 is to be done?

Source: D. Forsyth

Solution: smooth first



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

Source: S. Seitz

Derivative theorem of convolution

- This theorem gives us a very useful property:

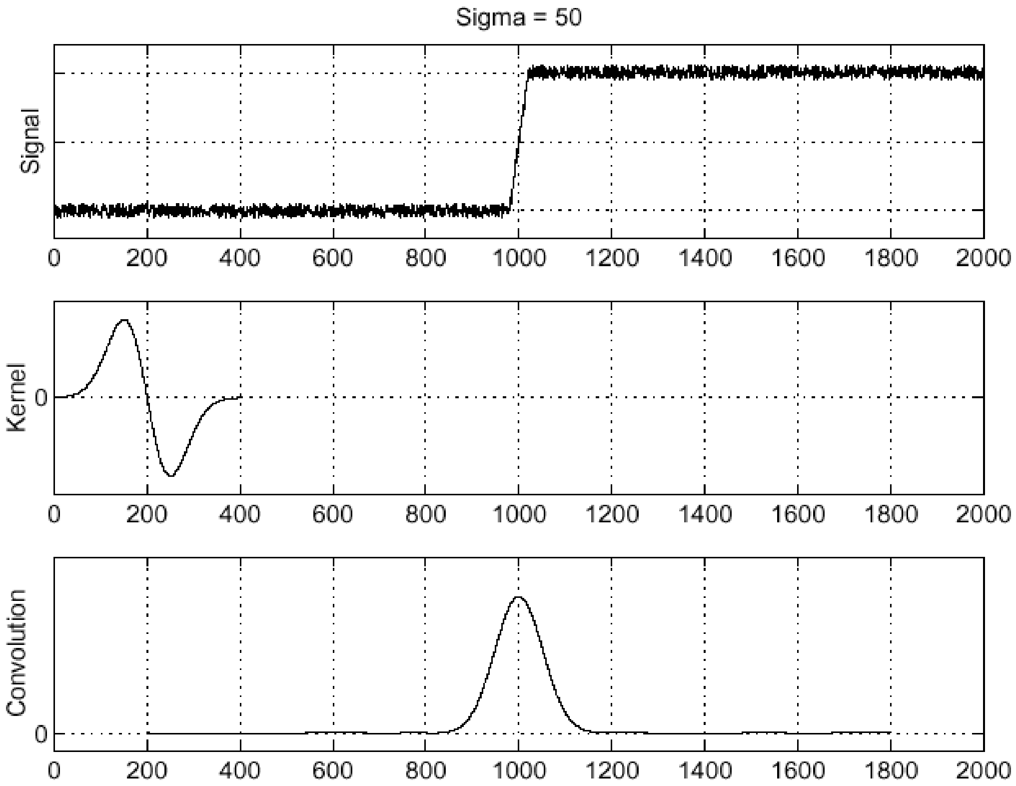
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

f

$\frac{d}{dx} g$

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



Source: S. Seitz

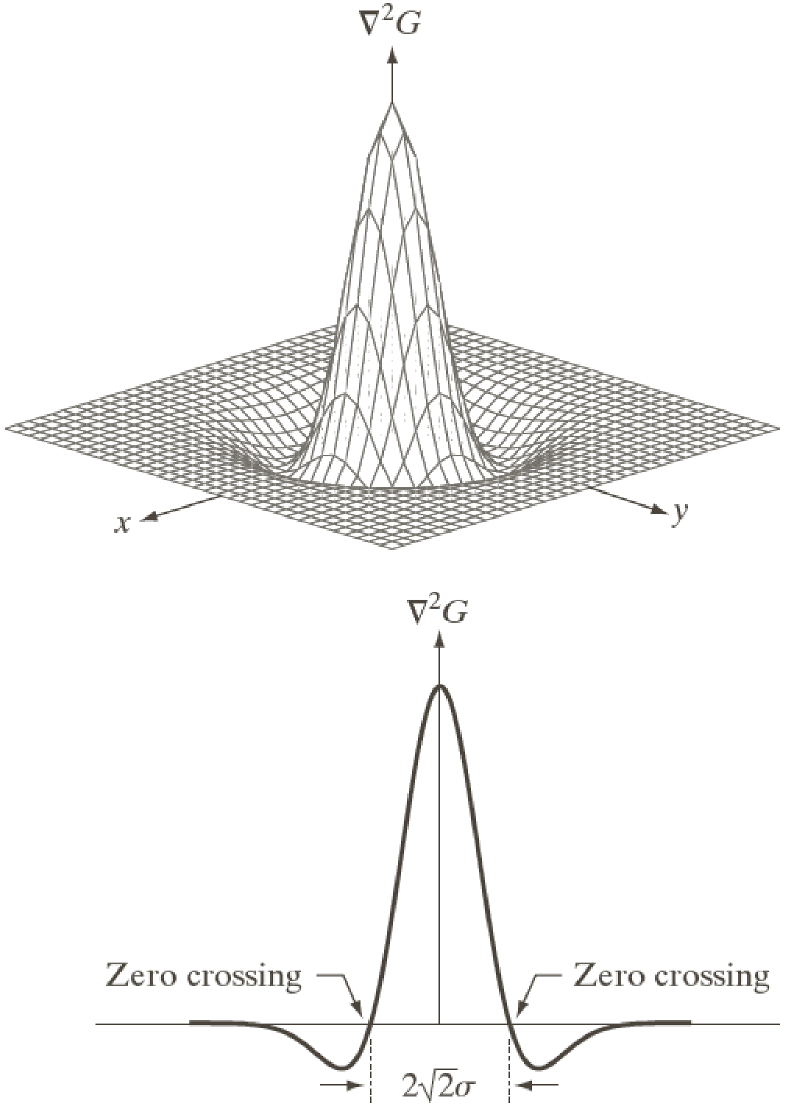
The Marr-Hildreth edge detector

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}, \ \ \sigma : \text{space constant.}$$

Laplacian of Gaussian (LoG)

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

$$= \left[\frac{x^2 + y^2 - \sigma^2}{\sigma^4} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$



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

a	b
c	d

FIGURE 10.21 (a) Three-dimensional 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 mask approximation to the shape in (a). The negative of this mask would be used in practice.

Marr-Hildreth Algorithm

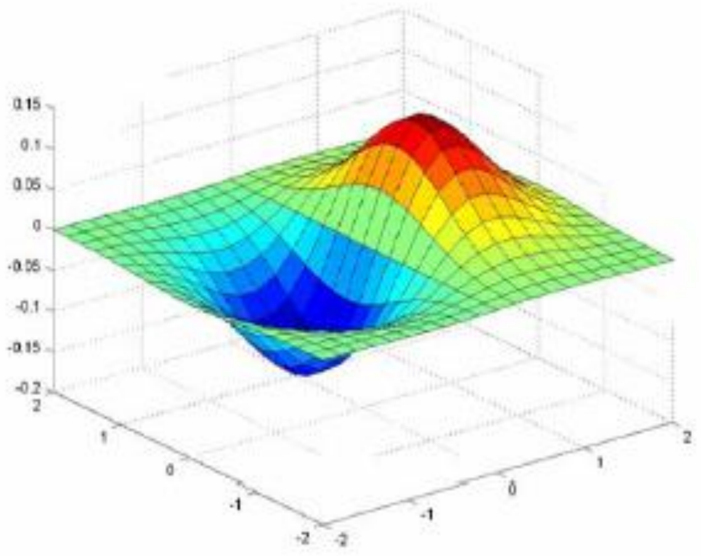
1. Filter the input image with a nxn Gaussian lowpass filter. N is the smallest odd integer greater than or equal to 6

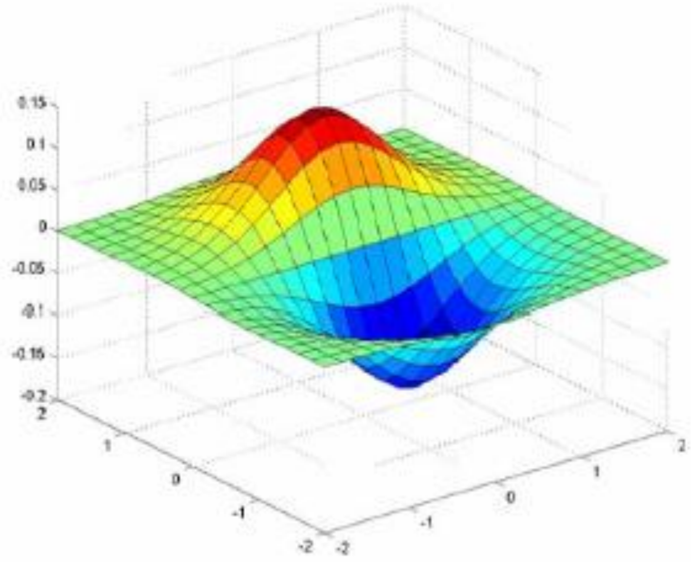
2. Compute the Laplacian of the image resulting from step1

3. Find the zero crossing of the image from step 2

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

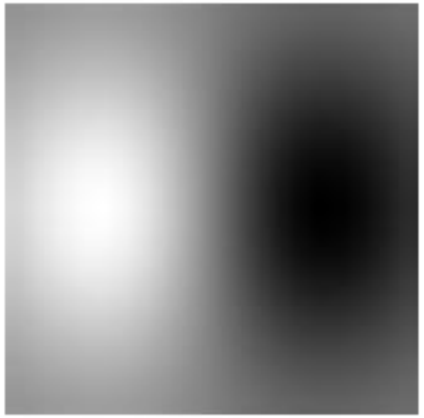
Derivative of Gaussian filter

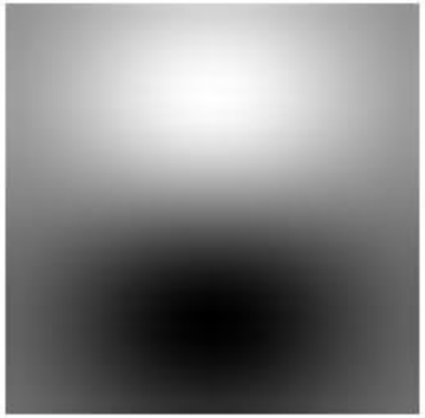




x-direction

y-direction





Detection of Isolated Points

- The Laplacian

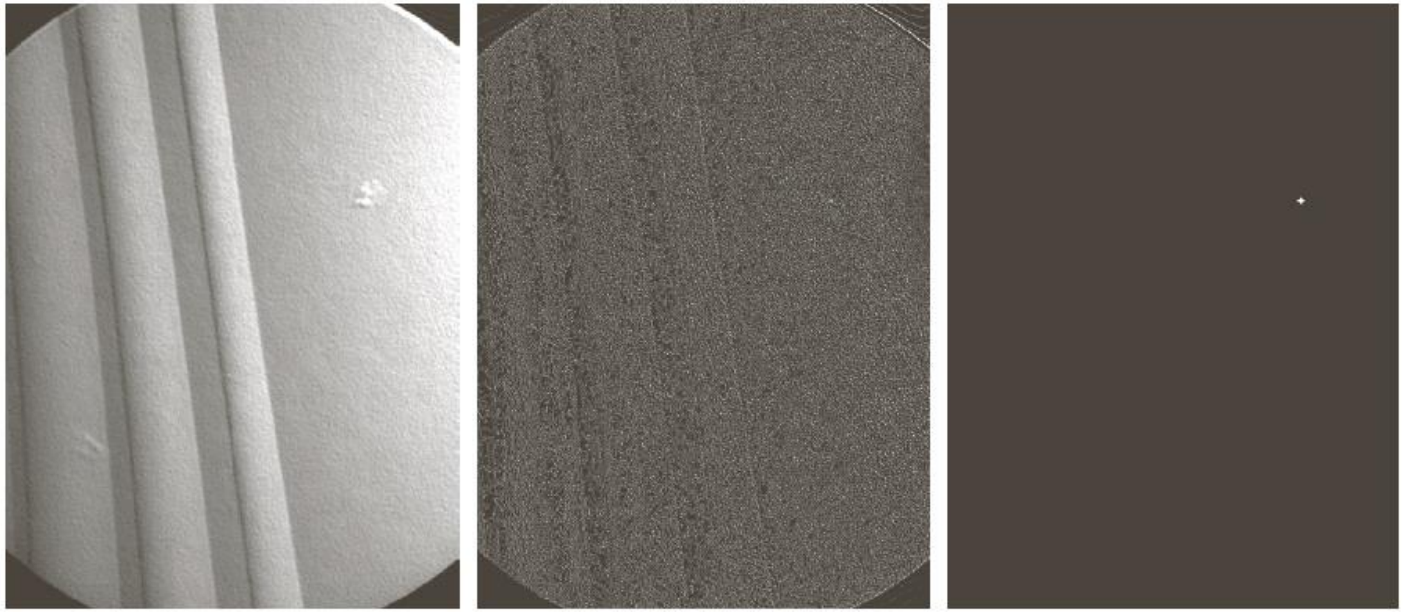
$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

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

$$-4f(x,y)$$

$$g(x,y) = \begin{cases} 1 & \text{if } |R(x,y)| \geq T \\ 0 & \text{otherwise} \end{cases} \qquad R = \sum_{k=1}^{9} w_k z_k$$

1	1	1
1	−8	1
1	1	1

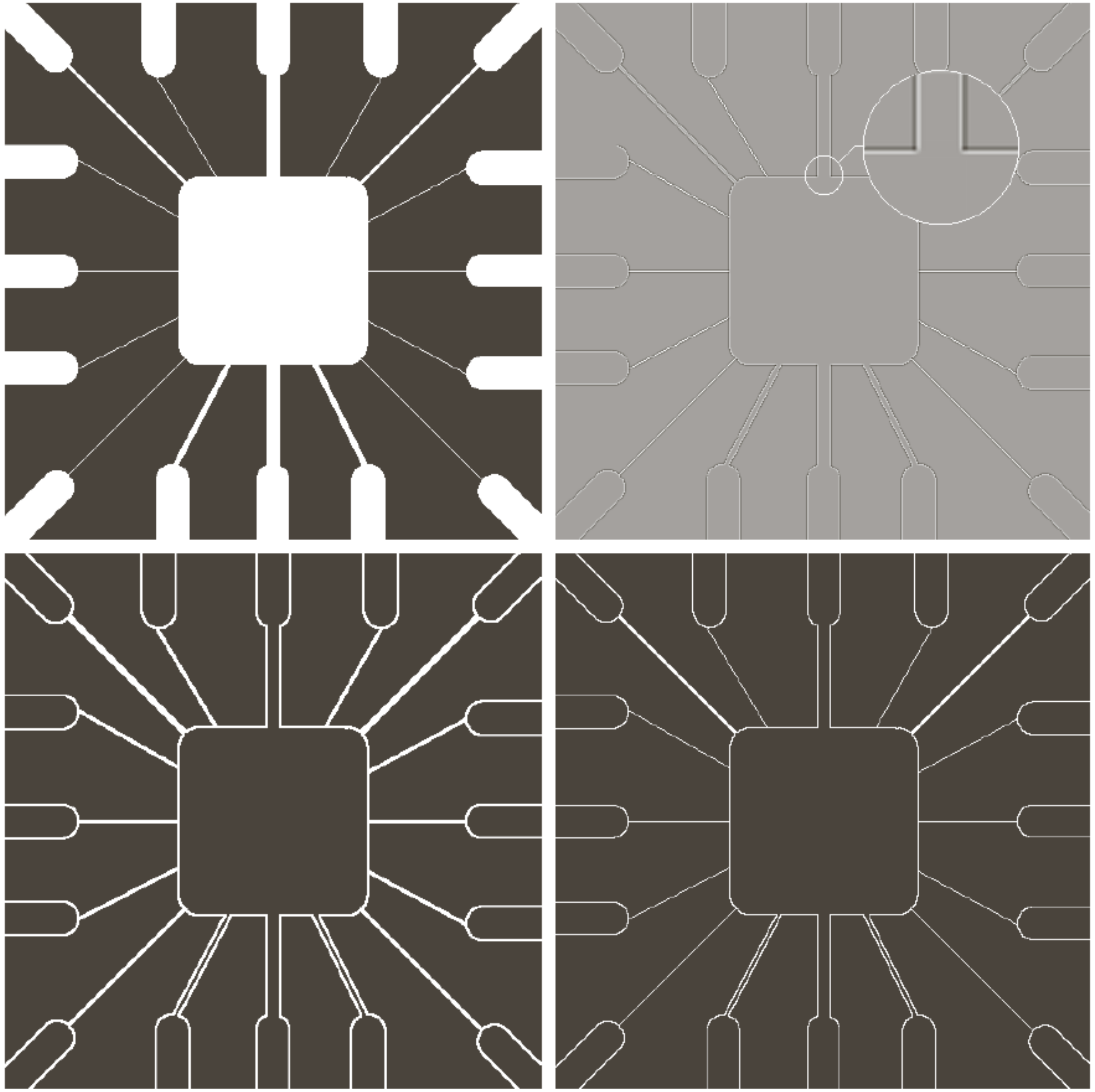


a
b c d

FIGURE 10.4 (a) Point detection (Laplacian) mask. (b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. (c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

Line Detection

- Second derivatives to result in a stronger response and to produce thinner lines than first derivatives
- Double-line effect of the second derivative must be handled properly



a b
c d

FIGURE 10.5
(a) Original image.
(b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian.
(c) Absolute value of the Laplacian.
(d) Positive values of the Laplacian.

Detecting Line in Specified Directions

Horizontal

−1	−1	−1
2	2	2
−1	−1	−1

+45°

2	−1	−1
−1	2	−1
−1	−1	2

Vertical

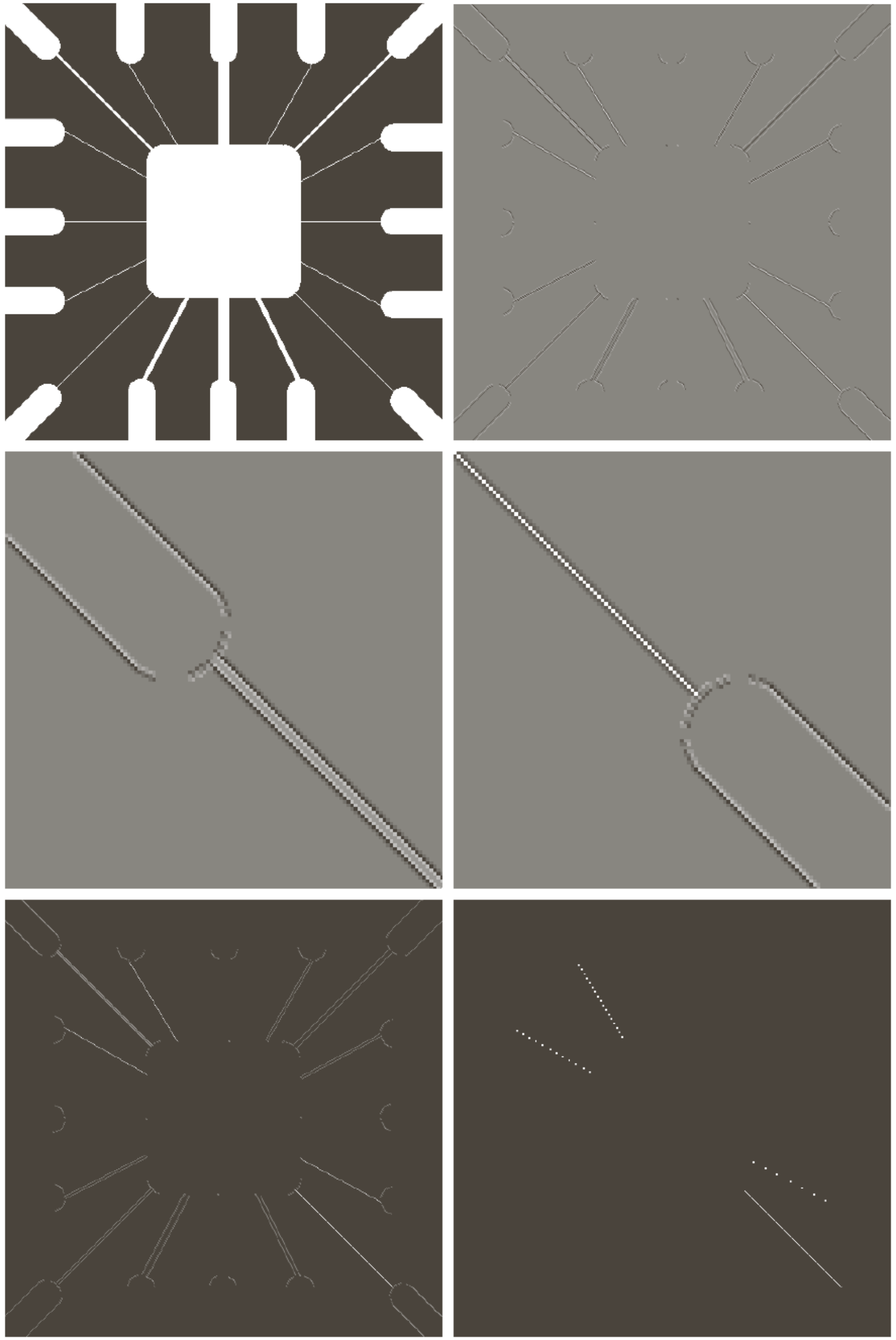
−1	2	−1
−1	2	−1
−1	2	−1

−45°

−1	−1	2
−1	2	−1
2	−1	−1

FIGURE 10.6 Line detection masks. Angles are with respect to the axis system in Fig. 2.18(b).

- Let R_1, R_2, R_3, and R_4 denote the responses of the masks in Fig. 10.6. If, at a given point in the image, $|R_k|>|R_j|$, for all $j \neq k$, that point is said to be more likely associated with a line in the direction of mask k.



a	b
c	d
e	f

FIGURE 10.7 (a) Image of a wire-bond template. (b) Result of processing with the $+45°$ line detector mask in Fig. 10.6. (c) Zoomed view of the top left region of (b). (d) Zoomed view of the bottom right region of (b). (e) The image in (b) with all negative values set to zero. (f) All points (in white) whose values satisfied the condition $g \geq T$, where g is the image in (e). (The points in (f) were enlarged to make them easier to see.)

Implementation issues

- The gradient magnitude is large along a thick "trail" or "ridge," so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth

Canny edge detector

- Criteria for a good edge detector:
  - Good detection: the optimal detector should find all real edges, ignoring noise or other artifacts
  - Good localization
    - the edges detected must be as close as possible to the true edges
    - the detector must return one point only for each true edge point

J. Canny, *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

Examples:

‘Lena’

Alexander Sawchuk @ USC, 1973

The Canny Edge Detector: Algorithm

Let $f(x, y)$ denote the input image and $G(x, y)$ denote the Gaussian function:

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

We form a smoothed image, $f_s(x, y)$ by convolving G and f:

$$f_s(x, y) = G(x, y) \star f(x, y)$$

The Canny Edge Detector: Algorithm

Compute the gradient magnitude and direction (angle):

$$M(x, y) = \sqrt{{g_x}^2 + {g_y}^2}$$

and

$$\alpha(x, y) = \arctan(g_y / g_x)$$

where $g_x = \partial f_s / \partial x$ and $g_y = \partial f_s / \partial y$

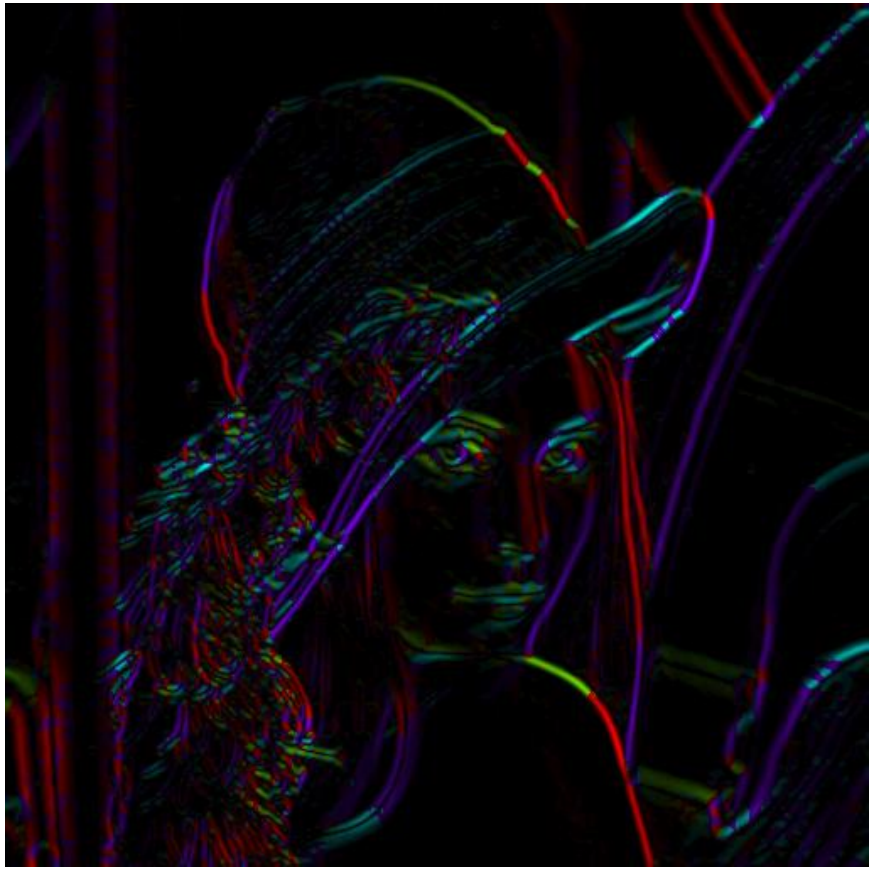
Compute Gradient Magnitude



sqrt(X-Deriv.ofGaussian ^2 + Y-Deriv.ofGaussian ^2) = gradient magnitude

Compute Gradient Orientation

theta = atan2(gy, gx)



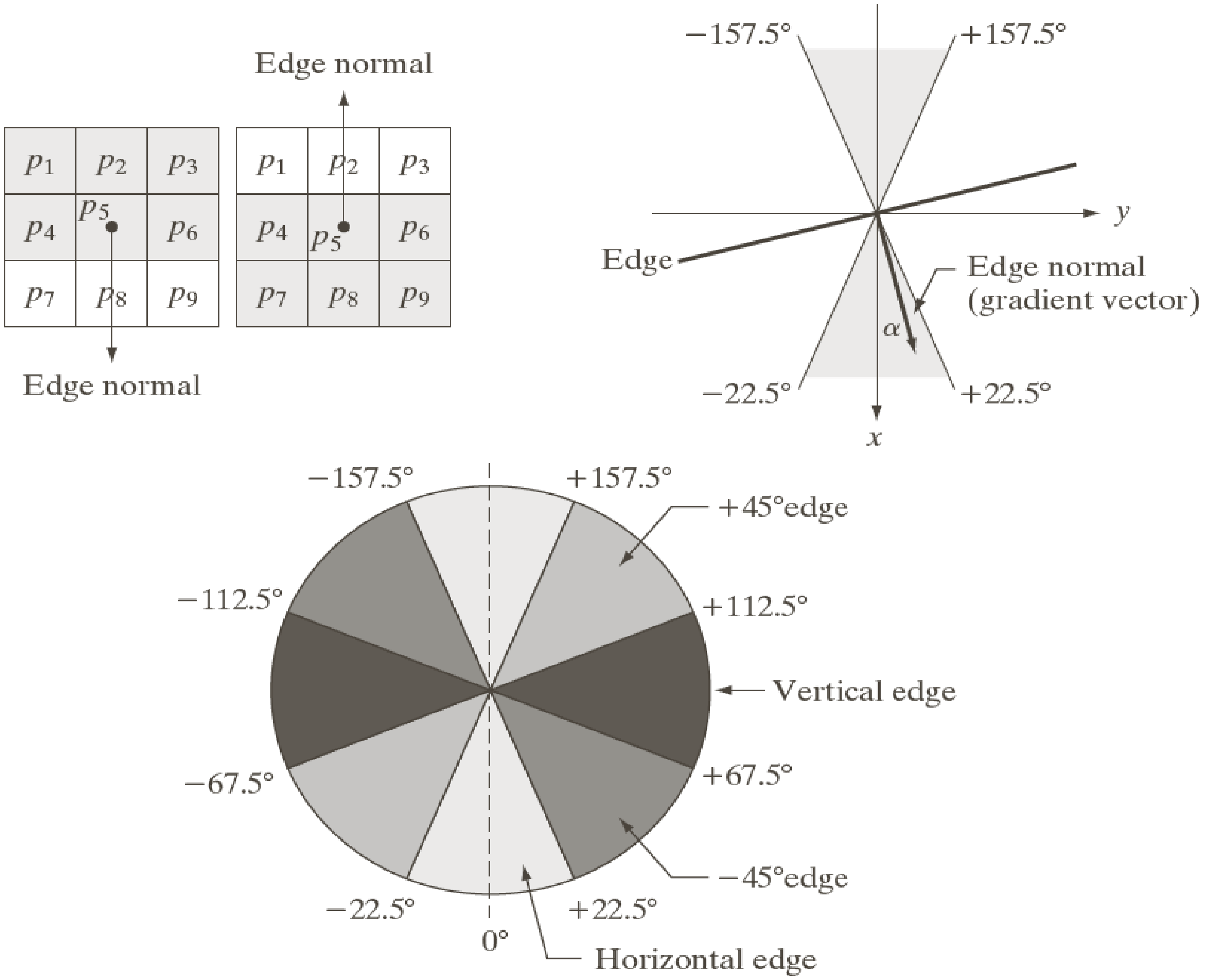
The Canny Edge Detector: Algorithm

The gradient $M(x, y)$ typically contains wide ridge around local maxima. Next step is to thin those ridges.

Nonmaxima suppression:

Let d_1, d_2, d_3, and d_4 denote the four basic edge directions for a 3×3 region: horizontal, -45°, vertical, +45°, respectively.

1. Find the direction d_k that is closest to $\alpha(x, y)$.
2. If the value of $M(x, y)$ is less than at least one of its two neighbors along d_k, let $g_N(x, y) = 0$ (suppression); otherwise, let $g_N(x, y) = M(x, y)$


Edge normal
p_1 p_2 p_3
p_4 p_5 p_6
p_7 p_8 p_9
Edge normal
p_1 p_2 p_3
p_4 p_5 p_6
p_7 p_8 p_9
−157.5°
+157.5°
y
Edge
Edge normal
(gradient vector)
α
−22.5°
+22.5°
x
−157.5°
+157.5°
+45°edge
−112.5°
+112.5°
Vertical edge
−67.5°
+67.5°
−45°edge
−22.5°
+22.5°
0°
Horizontal edge

Before Non-max Suppression

Gradient magnitude

James Hays

After non-max suppression

Gradient magnitude

James Hays

The Canny Edge Detector: Algorithm

The final operation is to threshold $g_N(x, y)$ to reduce false edge points.

Hysteresis thresholding:

$$g_{NH}(x,y) = g_N(x,y) \geq T_H$$
$$g_{NL}(x,y) = g_N(x,y) \geq T_L$$

and

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)$$

The Canny Edge Detector: Algorithm

Depending on the value of T_H, the edges in $g_{NH}(x, y)$ typically have gaps. Longer edges are formed using the following procedure:

(a). Locate the next unvisited edge pixel, p, in $g_{NH}(x, y)$.

(b). Mark as valid edge pixel all the weak pixels in $g_{NL}(x, y)$ that are connected to p using 8-connectivity.

(c). If all nonzero pixel in $g_{NH}(x, y)$ have been visited go to step (d), esle return to (a).

(d). Set to zero all pixels in $g_{NL}(x, y)$ that were not marked as valid edge pixels.

Final Canny Edges