Point, line, edge detection

Tools

Partial derivatives: $\partial_1 f(x,y)$, $\partial_2 f(x,y)$.

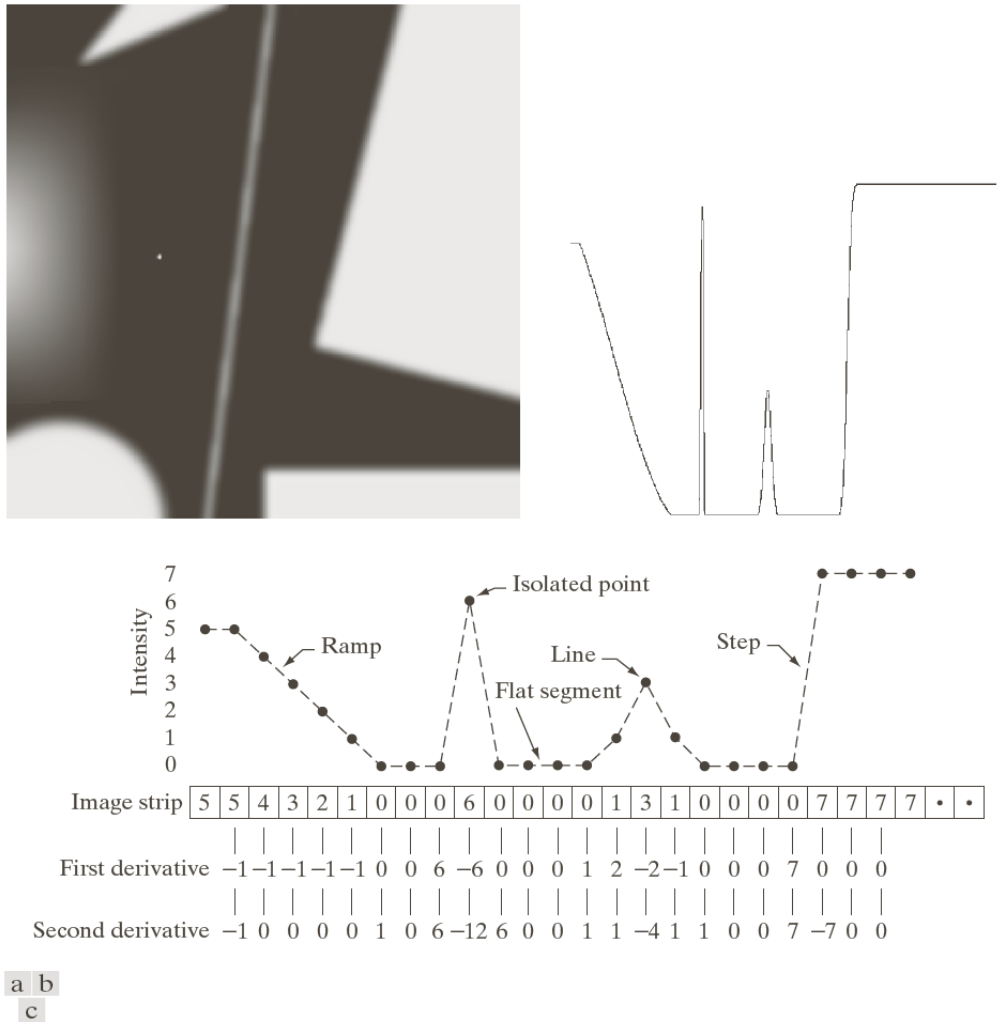
Discrete version:
$f(i+1,j) - f(i,j)$, $f(i,j+1) - f(i,j)$.

Second order partial derivatives: $\partial_{11} f(x,y)$, $\partial_{22} f(x,y)$.

Discrete version:
$f(i+1,j) - 2f(i,j) + f(i-1,j)$, $f(i,j+1) - 2f(i,j) + f(i,j-1)$.

Edge models

- Ramp edge: intensity changes gradually, linear change.
- Step edge: intensity changes suddenly, jump.
- Roof edge: intensity change up and down, linear change.



a b
c

FIGURE 10.2 (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10.2-1) and (10.2-2).

Detection of isolated point

Laplace operator: $\partial_{11} f(x, y) + \partial_{22} f(x, y)$.

Discrete version:

$$\nabla^2 f(i, j) = f(i + 1, j) + f(i - 1, j) + f(i, j + 1) + f(i, j - 1) - 4f(i, j)\,.$$

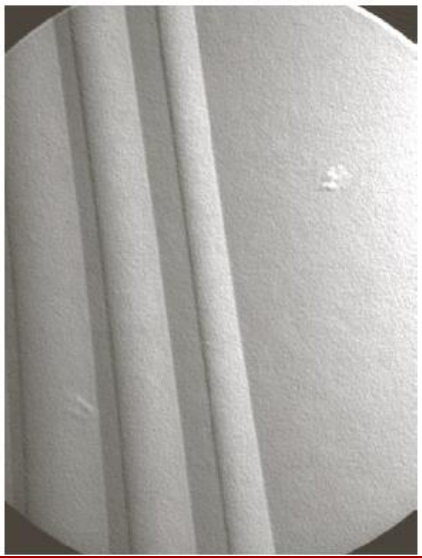
Reminder

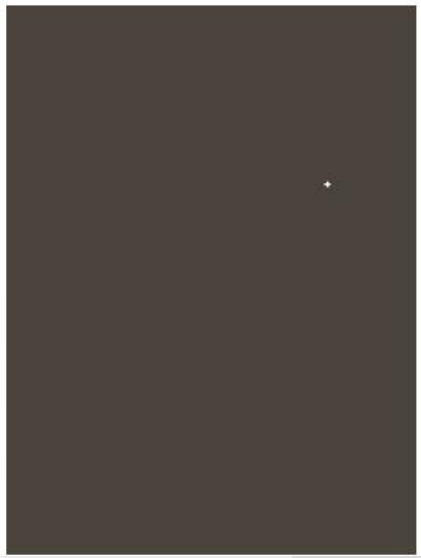
0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1

0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

Laplace operator and thresholding.

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

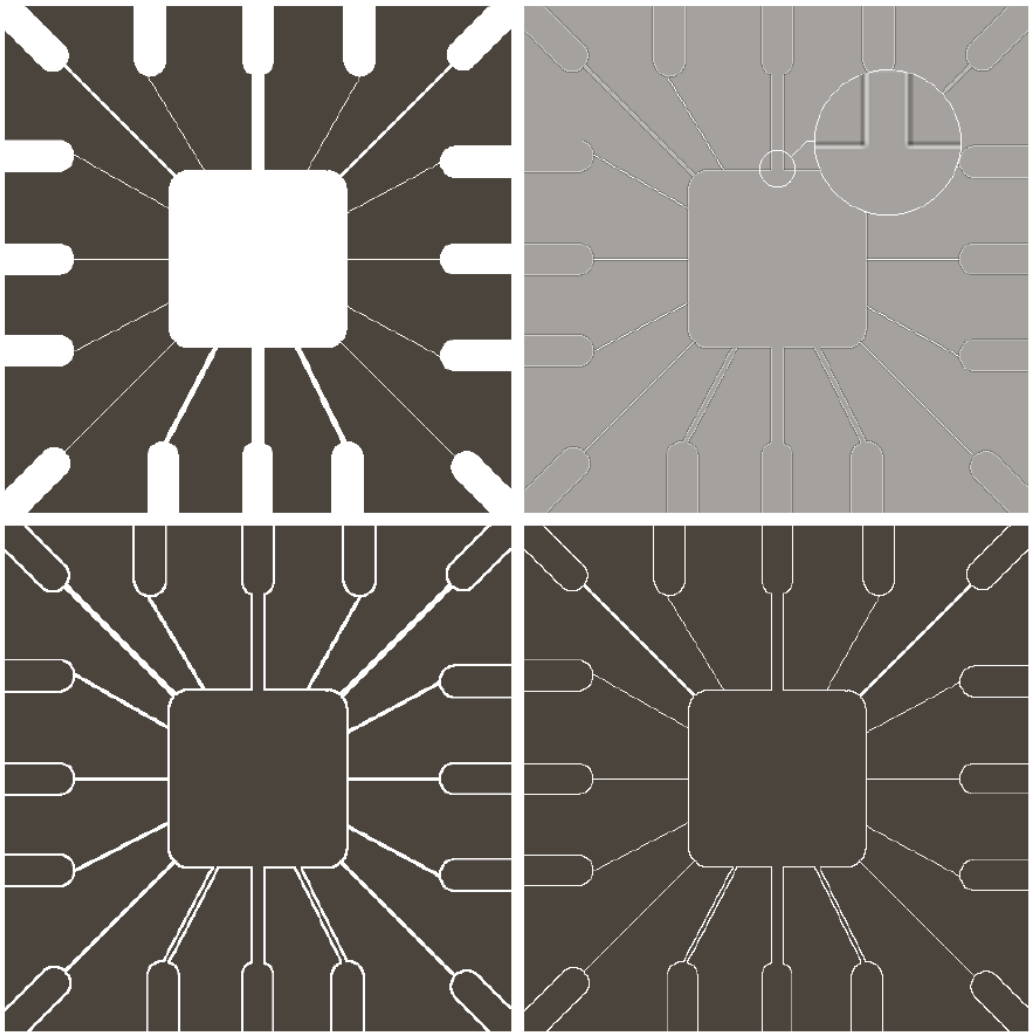
Laplace operator:

- Isotropic.
- Result: positive, negative values.
- Displaying

  Absolute values: doubled line thickness.

  Positive values only: thinner lines.
- Thick lines: 0 values ("valley") between the edges.

Line detection by Laplace operator (thresholding)



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.

Line detection with direction

- Optimized for one pixel thick lines.
- Need for thresholding.
- The surviving isolated point can be filtered out.

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

2	−1	−1
−1	2	−1
−1	−1	2

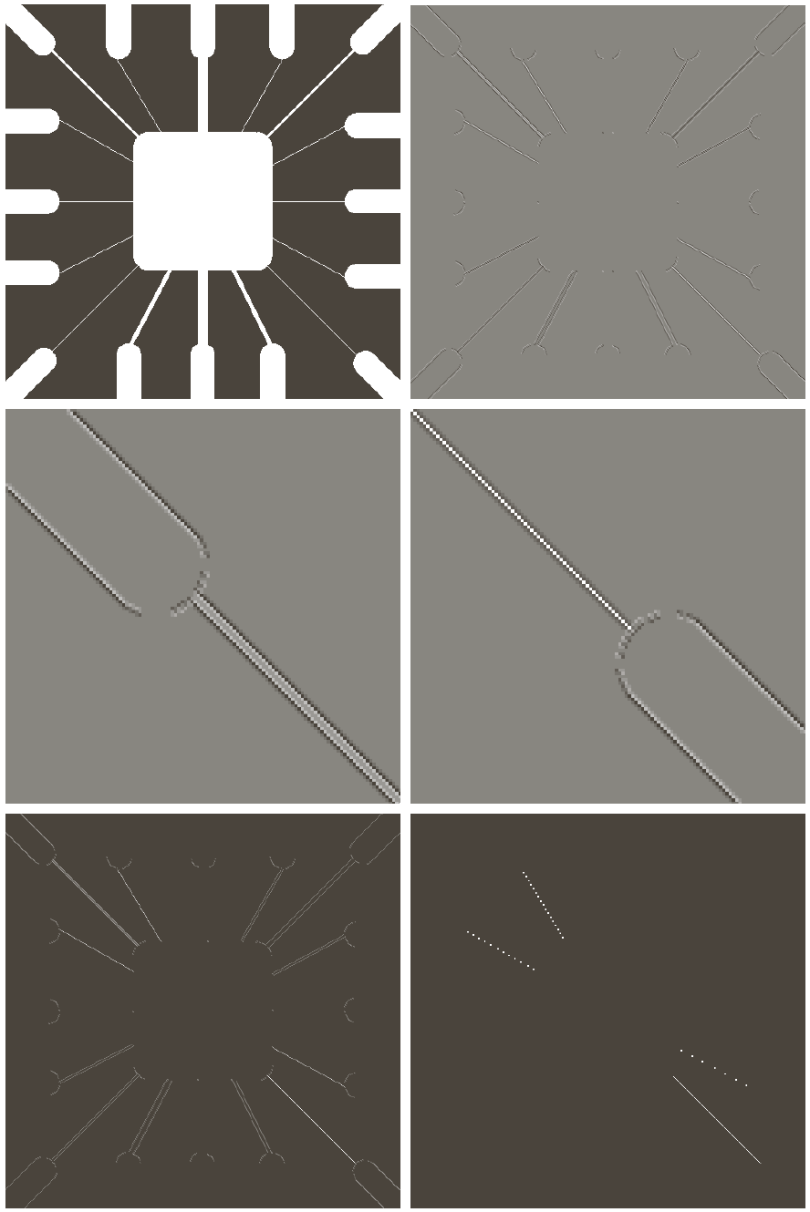
+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

−1	−1	2
−1	2	−1
2	−1	−1

−45°



a b
c d
e f

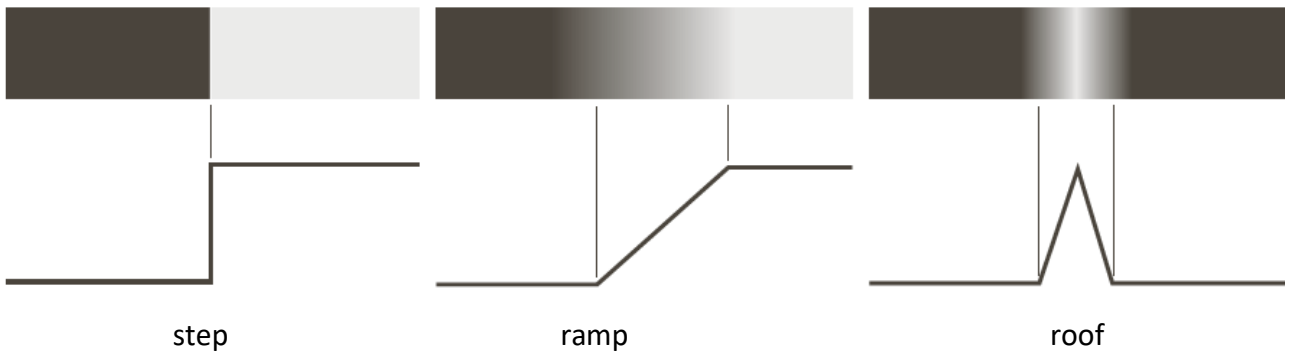
FIGURE 10.7
(a) Image of a wire-bond template.
(b) Result of processing with the +45° line detector mask in Fig. 10.6.
(c) Zoomed view of the top left region of (b).
(d) Zoomed view of the bottom right region of (b). (e) The image in (b) with all negative values set to zero. (f) All points (in white) 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.)

Edge models

Step edge: in the ideal case from one pixel to the next one.
No noise, no blur, no smoothing filters.

Ramp edge: continuous intensity change.
More realistic. Technical limitation of imaging devices, noise. The slope of change depends on the degree of blurs.

Roof edge: line.
Pipelines, roads on images.



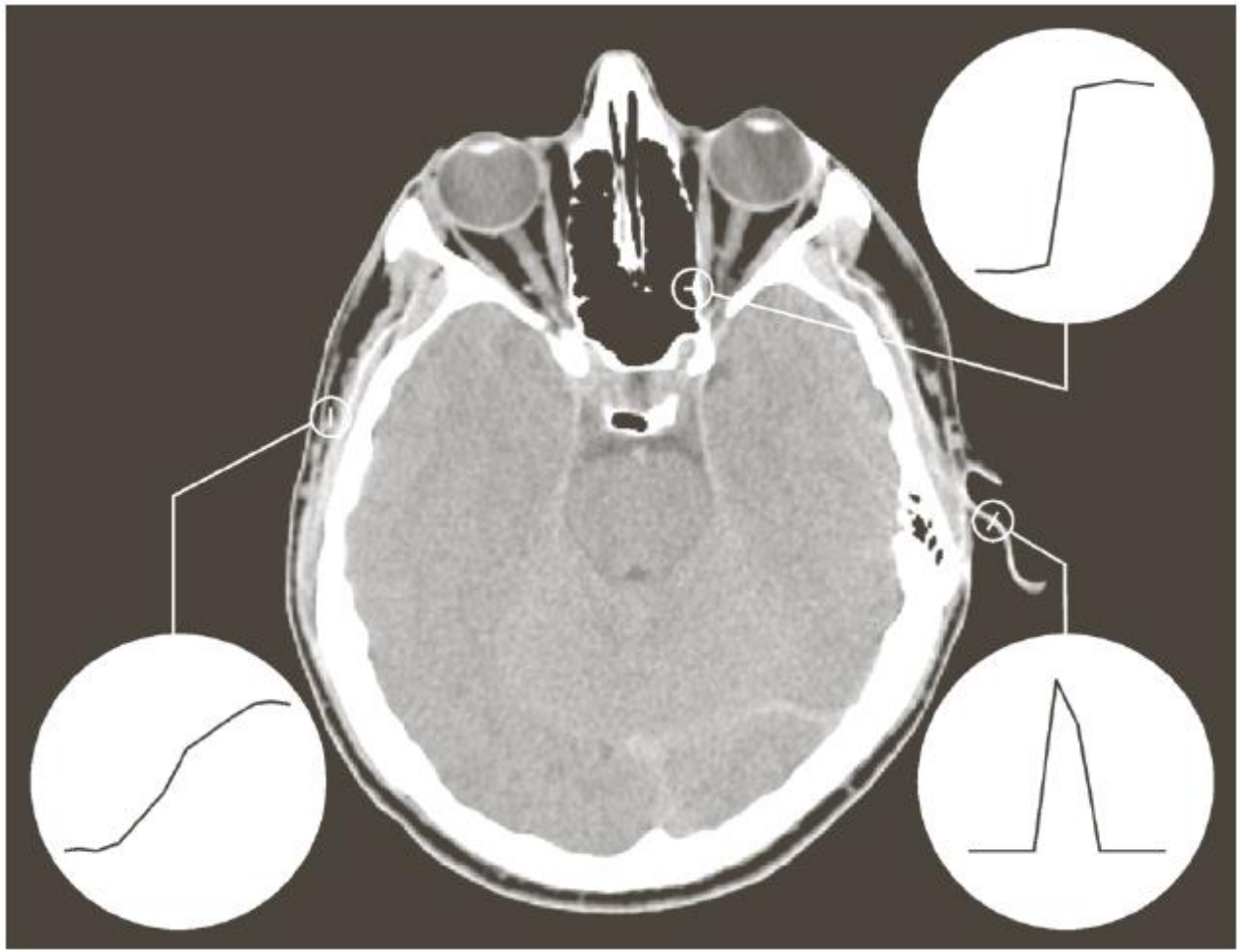
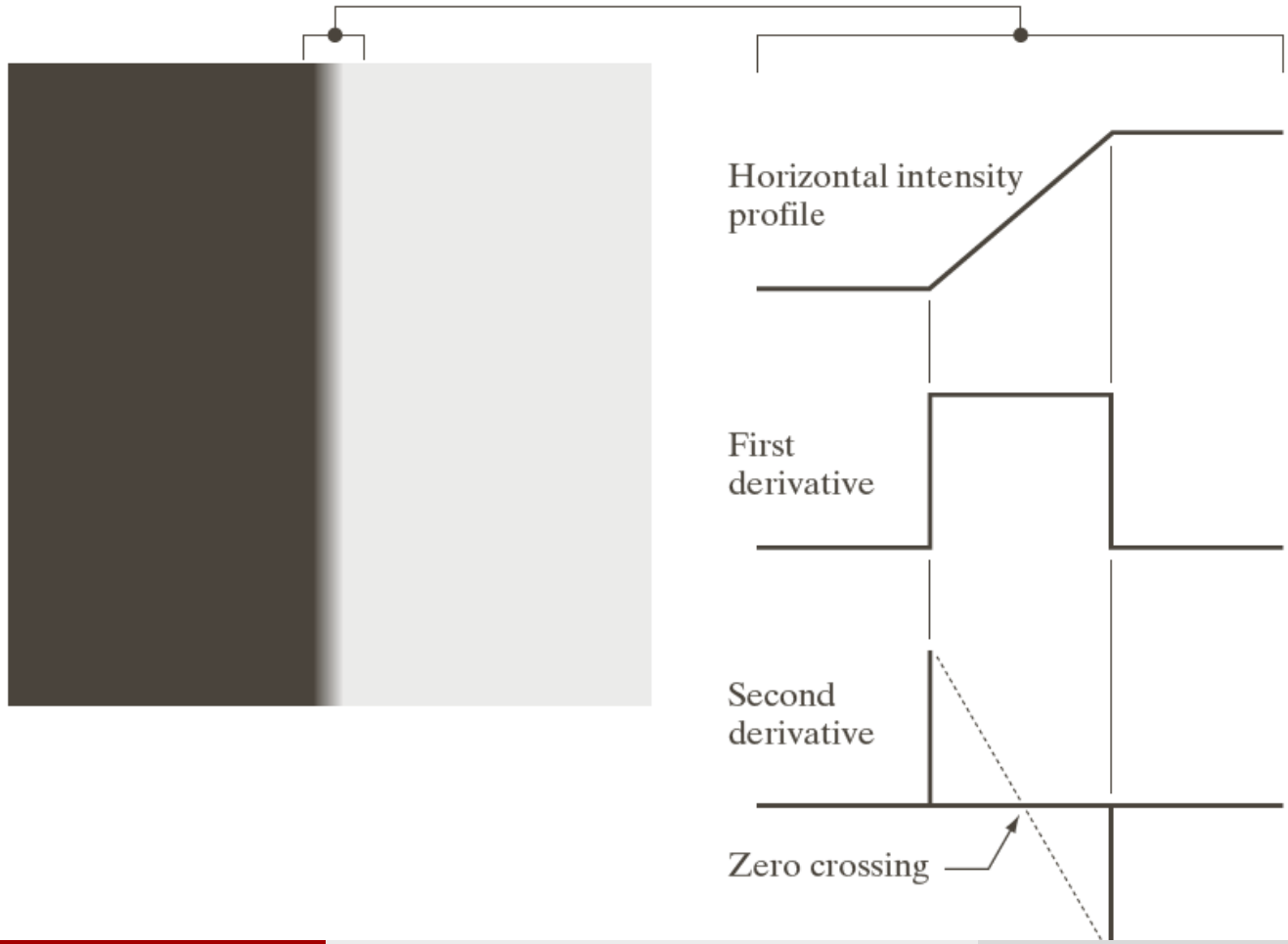


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively.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

Derivatives and edges

Second derivative: "zero crossing".



Images including noise

First image: noiseless.

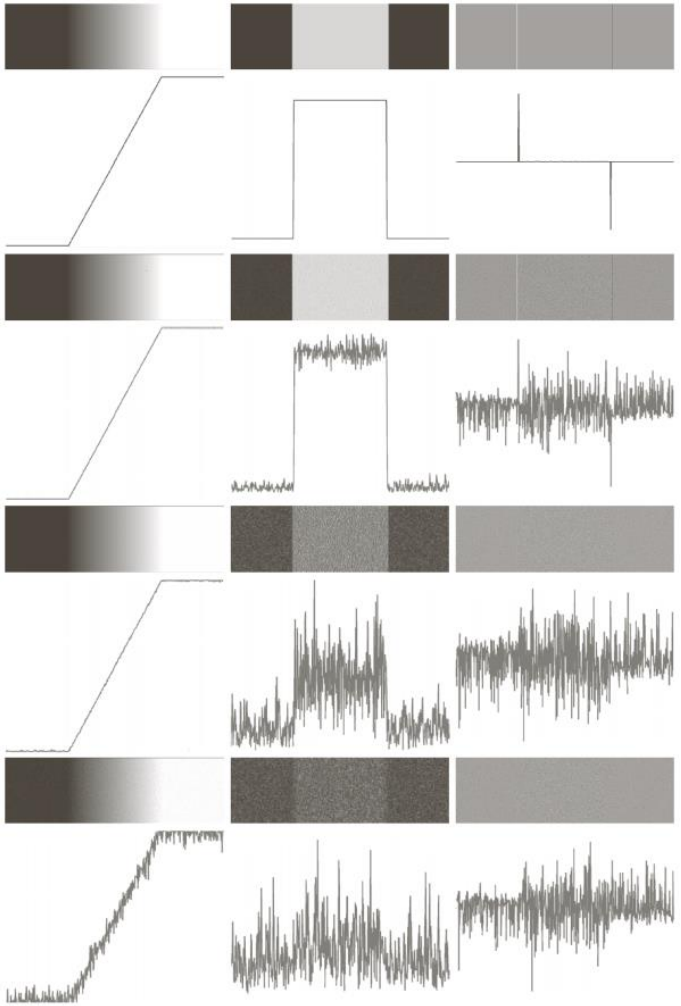
Further 3 images: Gauss-noise added.

Expectation: 0.

Deviation: 0.1, 1, 10.

Second derivative: positive, negative values

Rescaling: 0 gray.



Edge detection on real (non ideal) images

Steps

- Noise filtering: smoothing the image.
- Detection of edge points.

  Local operator. Finding the potential edge points.
- Determining the edge points.

  Selecting the real edge point from the potential ones.

Application of gradient

Gradient

- Gradient: $\nabla f(x,y) = (\partial_1 f(x,y), \partial_2 f(x,y))$

  Shows fast changes.

- The absolute value of the gradient:

  $|\nabla f(x,y)| = \sqrt{\partial_1^2 f(x,y) + \partial_2^2 f(x,y)}$

  Show how fast the change is. Very useful, for instance in thresholding.

  Approximation: $|\partial_1 f(x,y)| + |\partial_2 f(x,y)|$.

- The direction of the gradient: $\arctan \frac{\partial_2 f(x,y)}{\partial_1 f(x,y)}$.

  Point to the direction of fastest change. Perpendicular to the edge.

Usually: $\partial_1 f(j,k) = f(i+1,j) - f(j,k)$, $\partial_2 f(j,k) = f(i,j+1) - f(j,k)$.

Gradient operators

Variants

By definition:
1×2, 2×1 masks.

Roberts: 2×2 variant.

3×3 variants.
More information about the direction.
They are better in present of noise.

Noise: Sobel is better than Prewitt.

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

0	1	1
−1	0	1
−1	−1	0

−1	−1	0
−1	0	1
0	1	1

Prewitt

0	1	2
−1	0	1
−2	−1	0

−2	−1	0
−1	0	1
0	1	2

Sobel

a b
c d

FIGURE 10.15
Prewitt and Sobel masks for detecting diagonal edges.

1. Original image.

2.-4. images: Sobel $\partial_1 f$, $\partial_2 f$, gradient $|\partial_1 f| + |\partial_2 f|$.

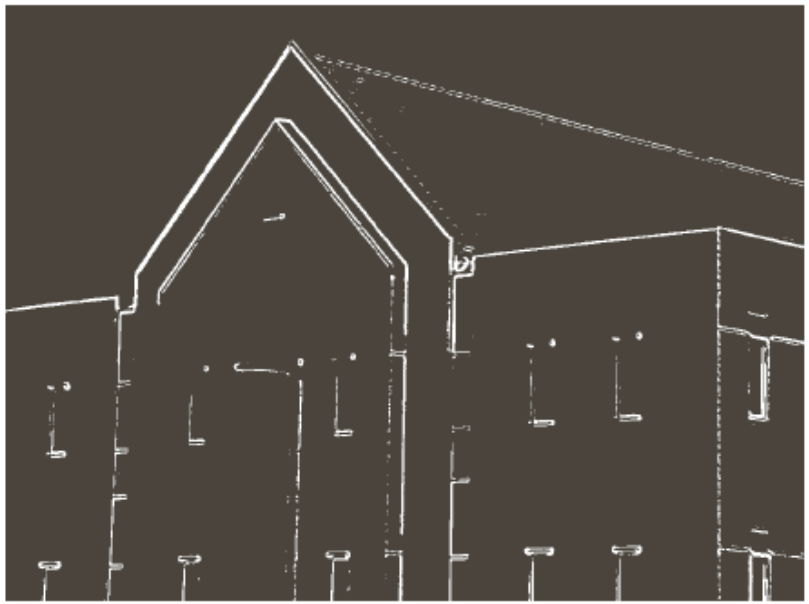
Same as before, but 5×5 averaging filter is used before edge detection.

Using diagonal Sobel edge detector for the previous case

Thresholding: for example keeping the 33% upper values.

Result: highlighting the main edges, less but sharper line, broken lines

The combination of smoothing, filtering and thresholding is advised.



Marr-Hildreth edge detector

Previous methods: local operators with fixed dimension.

Need for:

adaptive operators, variable dimension, adjust to the resolution.
For instance: higher dimension for blurred images, edges.

Remark: The derivative is still important.

Idea

- Use smoothing first by Gauss operator: $G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$.
  Remark: scalable.
- Then use Laplace operator for edge detection.
  Remark: since it is isotropic it works for any direction. No need to use different operators for different directions.

Result

The LoG ("Laplacian of a Gaussian") opertor: $\triangle(G * f)$.

$$\begin{aligned}\triangle(G * f)(x,y) =& \partial_{11} \int_0^1 \int_0^1 G(x-u, y-v) \cdot f(u,v)\, du\, dv \\ &+ \partial_{22} \int_0^1 \int_0^1 G(x-u, y-v) \cdot f(u,v)\, du\, dv\end{aligned}$$

parametric integral

$$\begin{aligned}&= \int_0^1 \int_0^1 \partial_{11} G(x-u, y-v) \cdot f(u,v)\, du\, dv \\ &+ \int_0^1 \int_0^1 \partial_{22} G(x-u, y-v) \cdot f(u,v)\, du\, dv \\ &= \big((\triangle G) * f\big)(x,y).\end{aligned}$$

In two steps: $\triangle(G * f)$.

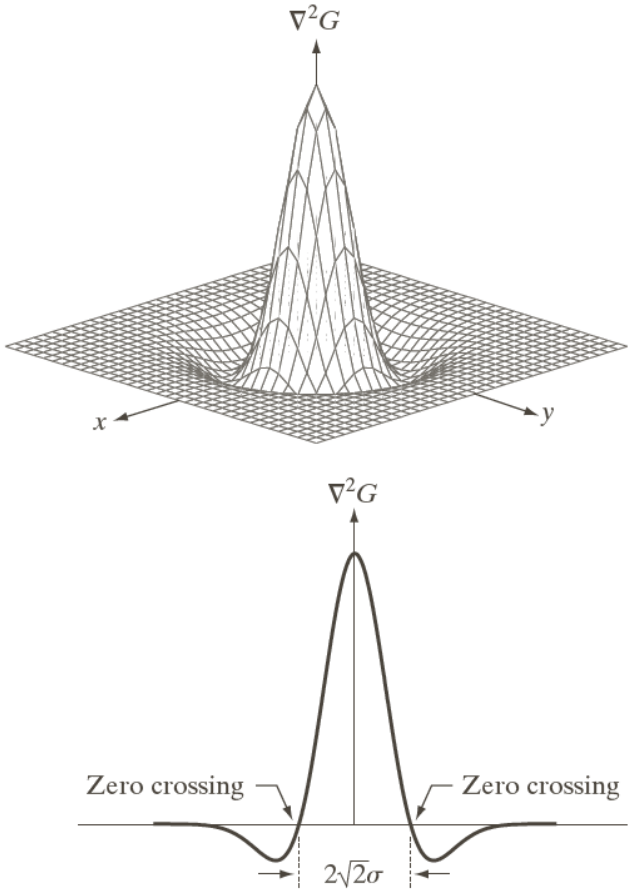
In one step: $(\triangle G) * f$.

$$(\triangle G)(x,y) = \partial_1\Big(-\frac{x}{\sigma^2}G(x,y)\Big) + \partial_2\Big(-\frac{y}{\sigma^2}G(x,y)\Big)$$
$$= \Big(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\Big)e^{-\frac{x^2+y^2}{2\sigma^2}} + \Big(\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\Big)e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

LoG:

$$\triangle G(x,y) = \Big(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\Big)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Discrete version with dimension 5×5.



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.

Simple rule for choosing proper size: $n \times n$.

n : the smallest odd number for which $n \geq 6\sigma$.

Reasoning

$$\oint_{x^2+y^2 \leq 9\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\, dx\, dy \geq 0.997 \int_{\mathbb{R}^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\, dx\, dy$$

Implementation

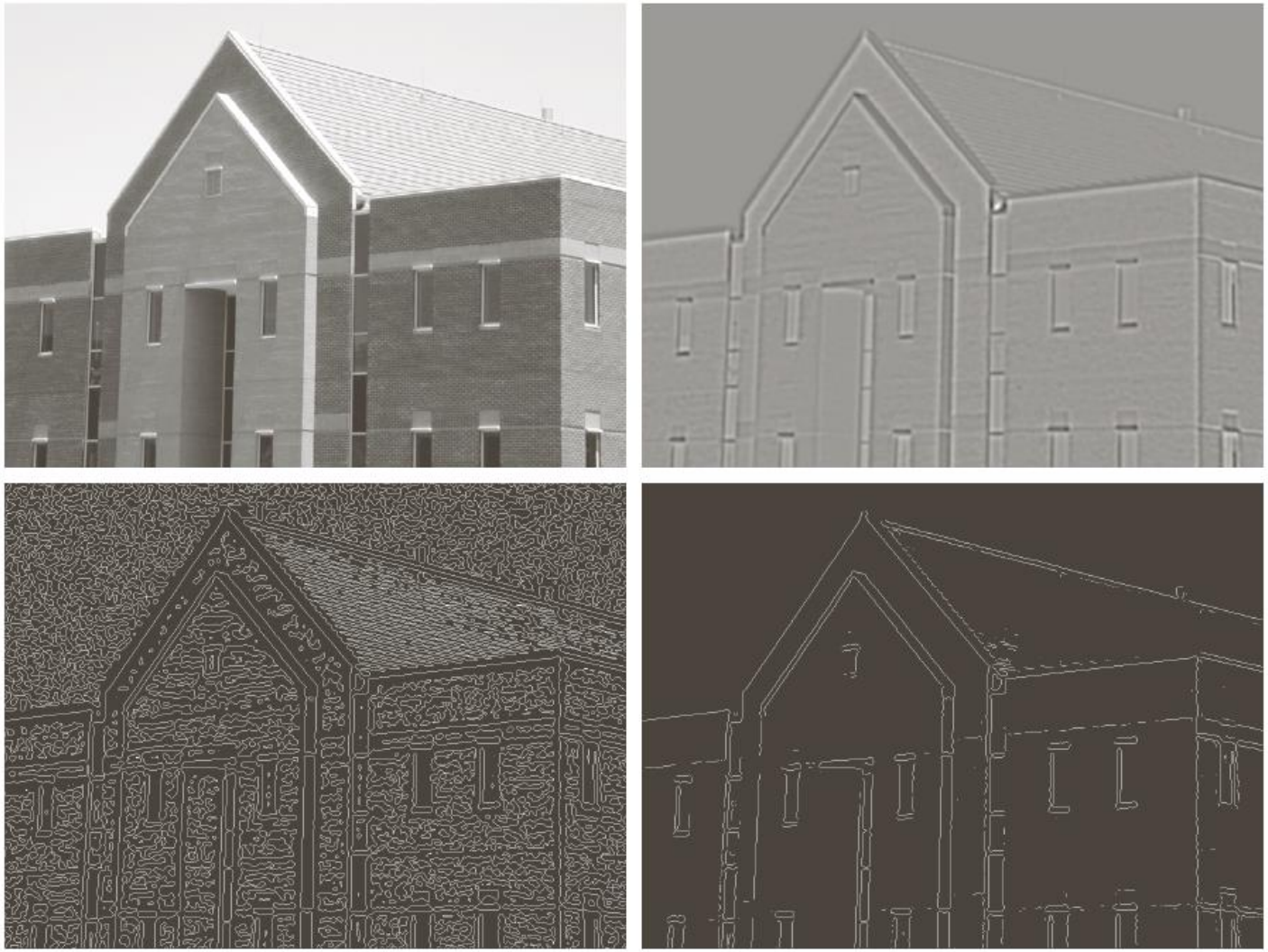
- In one or in two steps.
- Filtering by Gauss filter of size $n\times$.
- Use of Laplace operator. For instance with size 3×3.

Detection of potential edge points: 0 crossing.

Finding 0 crossings:

- 3×3 block around every point: check the sign changes in horizontal, vertical, diagonal directions.
- The value of jump should exceed a predetermined threshold. For instance 4% of the maximal value of the LoG transform.
- The result: one pixel thin edges.

Remark: Gauss filters with different σ values, 0 crossing maps, keeping the common points.



a b
c d

FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results 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.

Fast implementation

LoG:

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

i.e.

$$\begin{aligned}\big(\triangle G * f\big)(x,y) = &\int_0^1 \left(\int_0^1 \left(\frac{(x-u)^2-\sigma^2}{\sigma^4} e^{-\frac{(x-u)^2}{2\sigma^2}}\right) f(u,v)\, du \right) e^{-\frac{(y-v)^2}{2\sigma^2}}\, dv \\ &+ \int_0^1 \left(\int_0^1 \left(\frac{(y-v)^2-2\sigma^2}{\sigma^4}\right) e^{-\frac{(y-v)^2}{2\sigma^2}} f(u,v)\, dv\, e^{-\frac{(x-u)^2}{2\sigma^2}} \right) du\end{aligned}$$

Consequence

Instead of a two dimensional convolution the LoG transform can be performed by 4 one dimensional convolutions.

Image size: $M \times N$. Mask size: $n \times n$.

Number of operations using the one dimensional decomposition: $4MNn$.

Number of operations using two dimensional convolution: MNn^2.

Example: In case of $n = 25$ it is an acceleration by 6.

Remark

The LoG operator can be approximated by the difference of two Gauss operators: DoG transform.

$$\mathrm{DoG}(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}} \qquad (\sigma_1 > \sigma_2)$$

Generally: $\sigma_1/\sigma_2 = 1.75$, or 1.6.

Log-Dog comparison: if

$$\sigma = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln \frac{\sigma_1^2}{\sigma_2^2}$$

then same 0 crossings.

Demands

- Accuracy: Find every edge but no fake edges.
- Find the true positions of the edge points. The distance between the detected edge point and the real edge point should be minimal.
- Only one detected point for each real edge point.

Construction by experiments.

The aim of Canny

Satisfying the conditions above in a mathematical way.

Optimal realization for each of the three conditions: No closed formula can be expected.

One dimensional result

Numerical optimization: one dimensional case, step edge, Gauss noise.

The optimum can be approximated by the derivative of the Gauss function (difference within 20%):

$$G'(x) = \frac{-x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}.$$

Generalization for two dimension

Application of the one dimensional method in the direction that is perpendicular to the the edge.

Problem: the direction of the edges are not known.

Let us check every direction: Gauss filter, gradient at every point.
Direction of gradient $\rightarrow$ direction of edge.
Absolute value of gradient $\rightarrow$ the "strongness" of edge.

Two dimensional case

- Filtering by Gauss function: $f_s = G \circ f$.
- Calculation of gradient:

  The "absolute value" of gradient (estimation):
  $M(x, y) = |\partial_1 f_s| + |\partial_2 f_s|$.

  The direction of gradient: $\alpha(x, y) = \arctan \frac{\partial_2 f_s}{\partial_1 f_s}$.

Data processing

Problem: Many points with big $M(x, y)$ around the edges.

Goal: thinning.

Determining the direction of the gradient

- 4 main direction in case of 3×3 masks:
  horizontal, vertical, diagonal.
- Take the one which is closest to $\alpha(x, y)$.
  For example: horizontal if $-\pi/8 \leq \alpha(x, y) \leq \pi/8$.

Selecting the possible edge points: construction of the corresponding image g_N

Method: nonmaxima supression.

- 4 main directions.
- At a given point check the absolute values of the gradients for the two neighboring points.
- If the gradient at any neighbors is bigger than at the given point then set $M(x, y) = 0$: $g_N(x, y) = 0$.
- If the gradient at the given point is bigger than at the neighbors then keep the original value $M(x, y)$: $g_N(x, y) = M(x, y)$.

Filtering the fake edge points

Method: thresholding.

Poblem:

Low threshold $\rightarrow$ fake edge points survive.

High threshold $\rightarrow$ loosing real edge points.

Double thresholding

High threshold T_H :

$$g_{NH} = \begin{cases} g_N(x,y), & g_N(x,y) \geq T_H; \\ 0, & g_N(x,y) < T_H. \end{cases}$$

Low threshold T_L :

$$g_{NL}(x,y) = \begin{cases} g_N(x,y), & g_N(x,y) \geq T_L; \\ 0, & g_N(x,y) < T_L. \end{cases}$$

Double thresholding (continued)

The ratio of the thresholds: T_H/T_L is usually 2 or 3.

Modifying g_{NL}: $g^*_{NL} = g_{NL} - g_{NH}$.

Strong edge points: $g_{NH}(x, y) \neq 0$.

Weak edge points: $g^*_{NL}(x, y) \neq 0$.

After thresholding

- Strong edge point are considered as real edge points.
- Result: real but broken, discontinuous edges.
- Reason: real edge point are dropped by using high threshold.
- There are real edge point among the weak edge points.

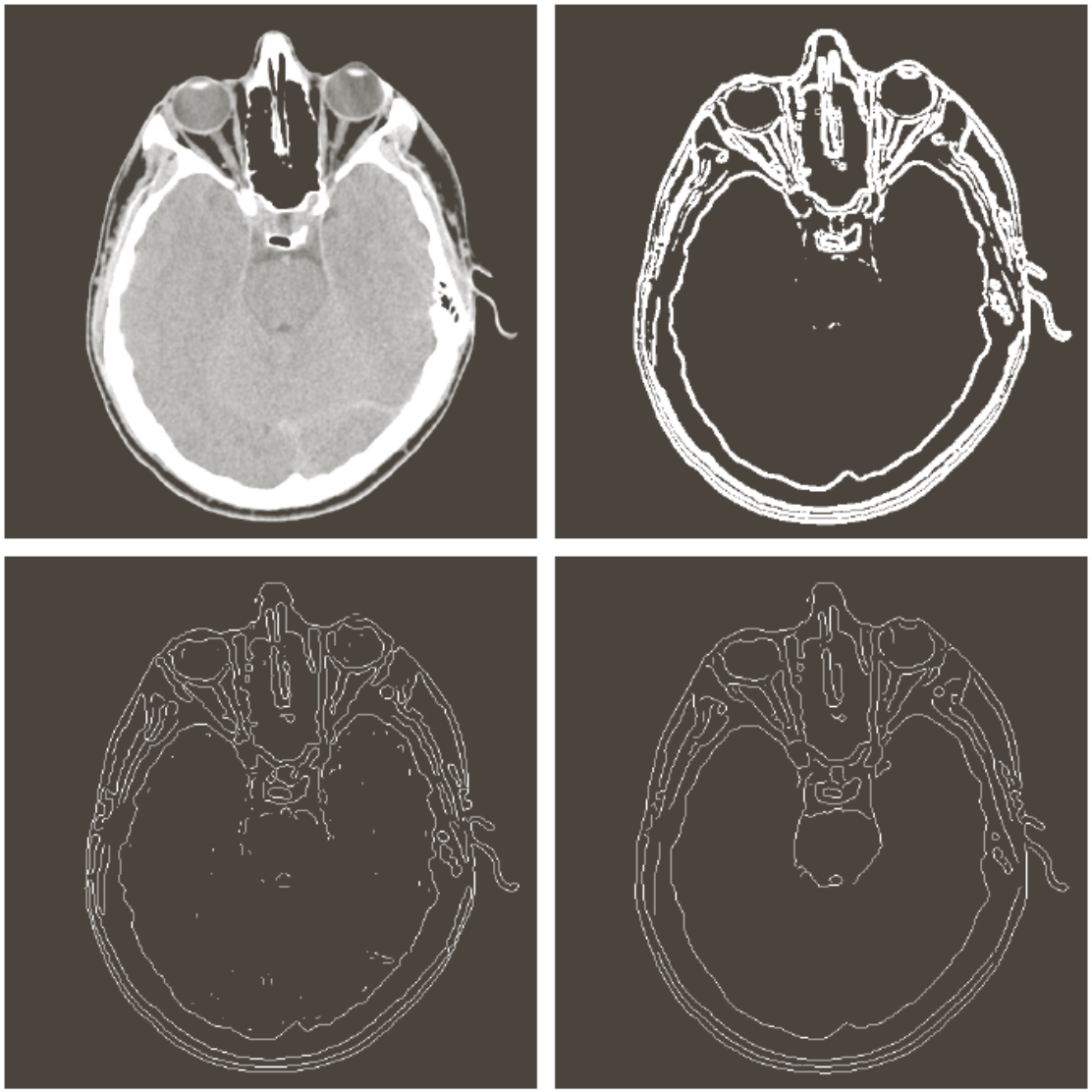
Adding edge point to the strong edge points

- Goal: finding the real edge points.
- Every strong edge point is real edge point.
- Take (for instance) the 3×3 neighborhood of every strong edge point. Weak edge points within this neighborhood will be taken as real edge points.
  For such (x, y) points set $g_{NL}^{**}(x, y) = g_{NL}^{*}(x, y)$.
- After exhausting the set of strong edge points drop those weak edge points that were not selected.
  For these (x, y) points set $g_{NL}^{**}(x, y) = 0$.

  Finally: $g(x, y) = g_{NH}(x, y) + g_{NL}^{**}(x, y)$.

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.



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.)

Set of detected edge points

The edge points that are identified do not usually form an edge.

Possible reasons:

- noise,
- illumination varies,
- other fake intensity changes on the image.

Solution: algorithm that connects the edge points.

Local method

- Fix a window size.
- Select thresholds E, A.
- Take a window S_{jk} around every edge point (j, k).

Local method (continued)

- Take an edge point (m, n) within S_{jk}.
- If

$$|M(j,k) - M(m,n)| \leq E \quad \text{és} \quad |\alpha(j,k) - \alpha(m,n)| \leq A$$

(M the absolute value of the gradient, α the angle of the gradient), then connect the two pixels.

- Go through every edge pixel and do the connections.

Complicated, computationally demanding.

Simplification: edges in fix direction.

Fixed dierction

Let β denote the fixed direction.

Take the binary image:

$$g(j,k) = \begin{cases} 1, & \text{if } M(j,k) \geq E_\beta \text{ and } |\alpha(j,k) - \beta| \leq A_\beta; \\ 0, & \text{otherwise.} \end{cases}$$

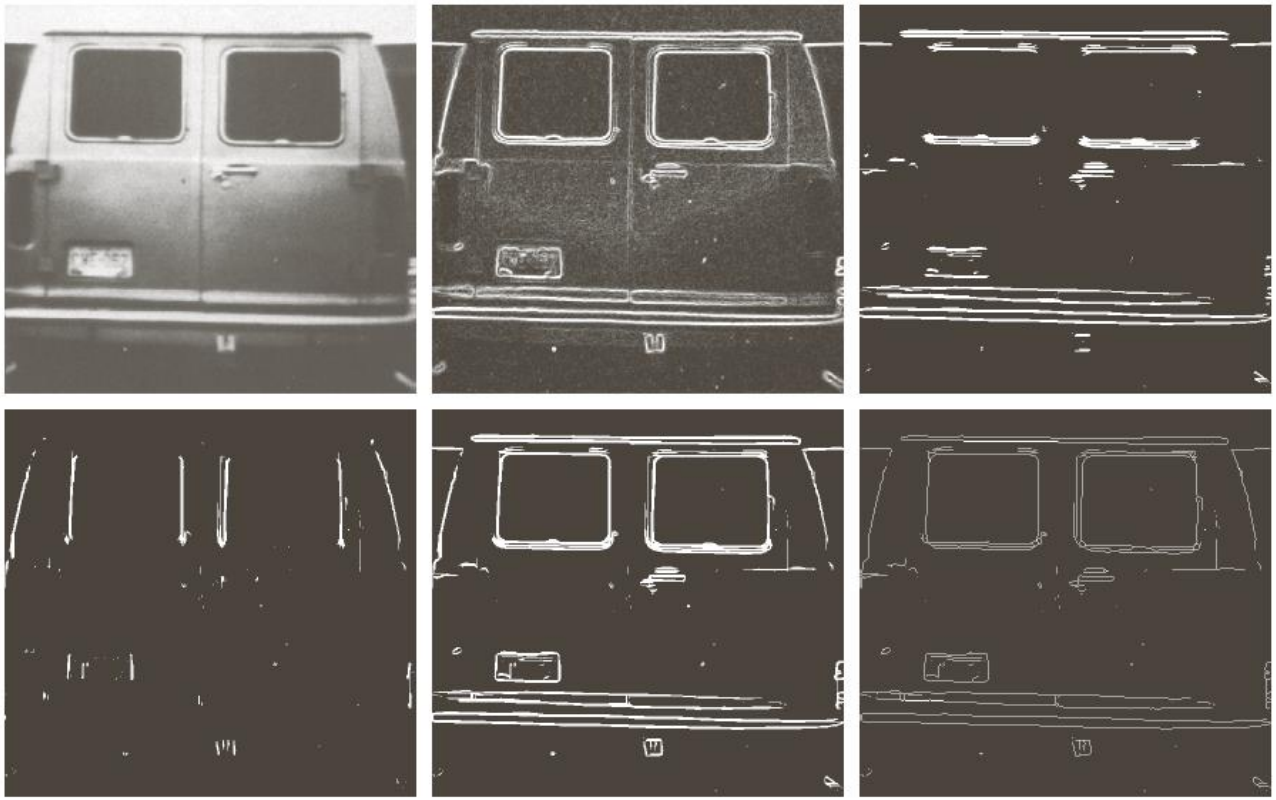
Horizontal edge

- Go through the rows of g.
- Fill the gaps if the size of them do not exceed a predetermined value K :

  if there are at most K pieces of 0 between two 1, then change them to 1.

In case of other direction (except for vertical) rotate the image to make it horizontal or vertical.

E is 30% of the greatest value of the gradient, $\beta = 0, \pi/2$, $A_\beta = \pi/4$, $K = 25$.



a b c
d e f

FIGURE 10.27 (a) A 534 × 566 image of the rear of a vehicle. (b) Gradient magnitude image. (c) Horizontally connected edge pixels. (d) Vertically connected edge pixels. (e) The logical OR of the two preceding images. (f) Final result obtained using morphological thinning. (Original image courtesy of Perceptics Corporation.)