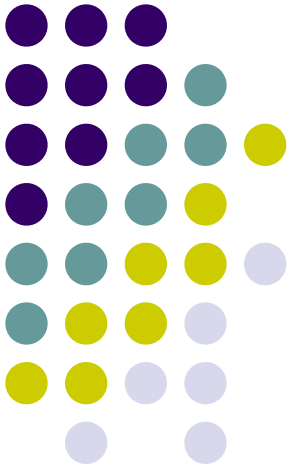
Topics in Image Processing

Point, Line, Edge Detection, and Image Segmentation

Instructor: Yiming Xiao

Email: yiming.xiao@concordia.ca

Department of Computer Science
and Software Engineering
Concordia University



Slides modified from materials provided by Dr. Tien D Bui

Image segmentation

computer vision & image processing

- pattern recognition
- video processing
- image registration
- document analysis
- etc.

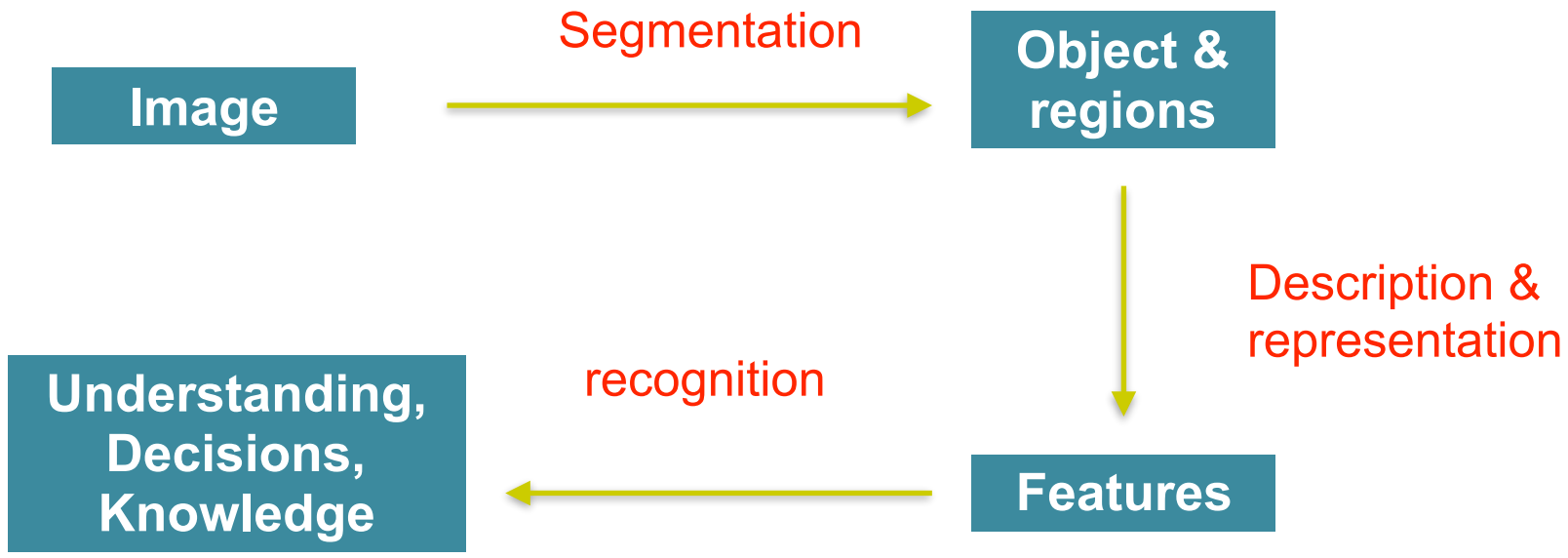


Image Segmentation

Objective: Partition an image Ω into sub-regions Ω_i: $\Omega = \Omega_1 \cup \ldots \cup \Omega_n$ such that:

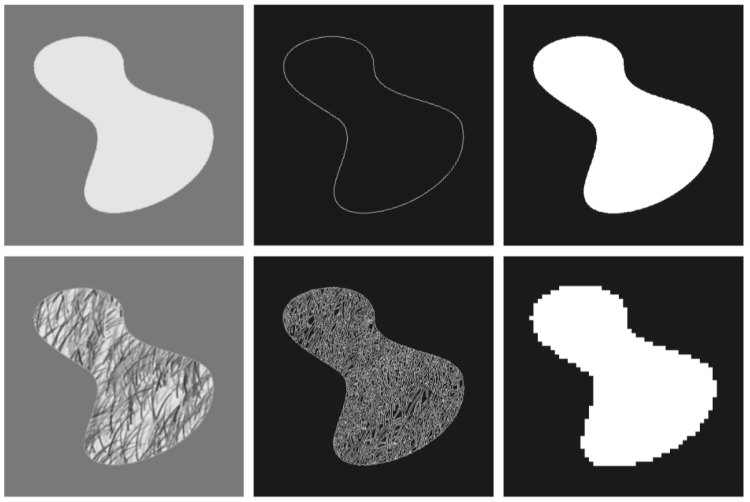
1. the image data $u(x, y)$ vary smoothly and/or slowly within each Ω_i

2. the image $u(x, y)$ varies discontinuously and/or rapidly across the boundary C between different Ω_i

Segmentation and edge detection

Image segmentation

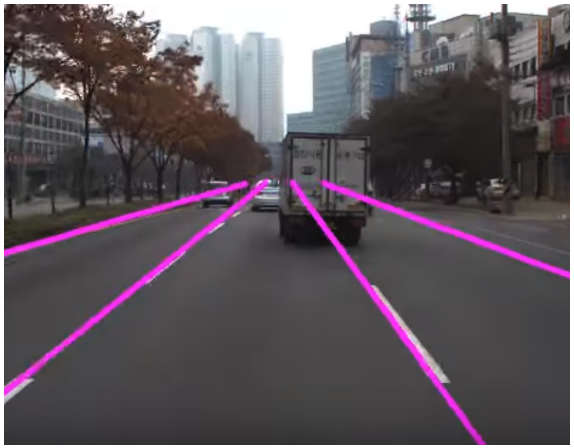
- Partition an image into its constituent parts or objects.
- Two main concepts for image segmentation methods
  - **Homogeneity measure within objects ⇨similarity**
  - **Contrast measure on the border of objects ⇨discontinuity**
- Autonomous segmentation is one of the most difficult tasks in digital image processing.

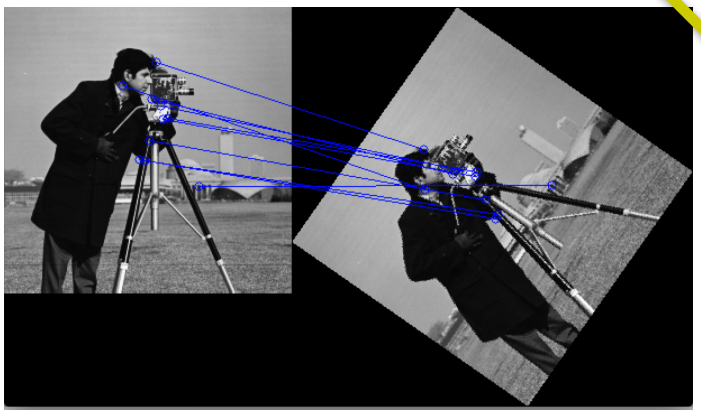


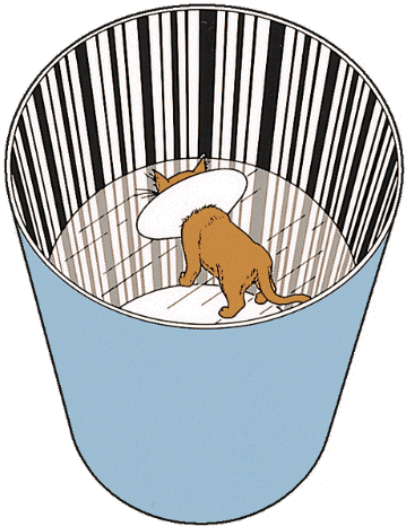
a b c
d e f

FIGURE 10.1
(a) Image of a constant intensity region. (b) Boundary based on intensity discontinuities. (c) Result of segmentation. (d) Image of a texture region. (e) Result of intensity discontinuity computations (note the large number of small edges). (f) Result of segmentation based on region properties.

Points, Lines, and edges detection







Kittens raised without exposure to horizontal lines.

Blakemore and Cooper (1970)

Derivative types

$$\frac{\partial f}{\partial \mathrm{x}} \approx \frac{f(\mathrm{x}+h)-f(\mathrm{x})}{h} \approx f(x+1)-f(x)$$

$$\frac{\partial f}{\partial \mathrm{x}} \approx f(x+1,y)-f(x,y)$$

$$\frac{\partial f}{\partial \mathrm{y}} \approx f(x,y+1)-f(x,y)$$

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

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

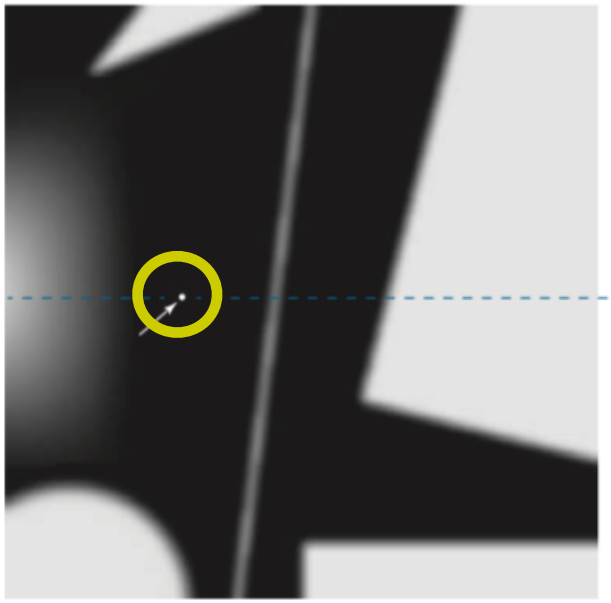
Laplacian

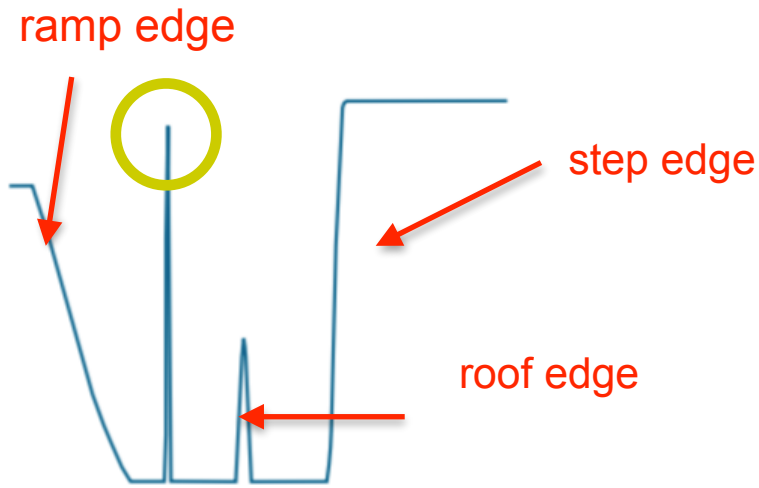
$$\nabla^2 f = \frac{\partial^2 f}{\partial \mathrm{x}^2} + \frac{\partial^2 f}{\partial \mathrm{y}^2} \Rightarrow \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

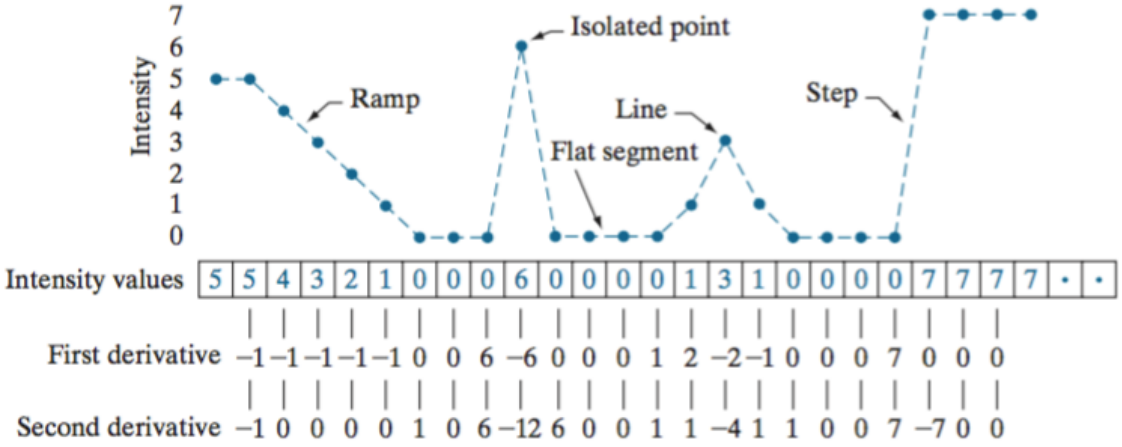
Points, Lines, and Edge detection

a b
c

FIGURE 10.2
(a) Image.
(b) Horizontal intensity profile that includes the isolated point indicated by the arrow.
(c) Subsampled profile; the dashes were added for clarity. The numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10-4) for the first derivative and Eq. (10-7) for the second.







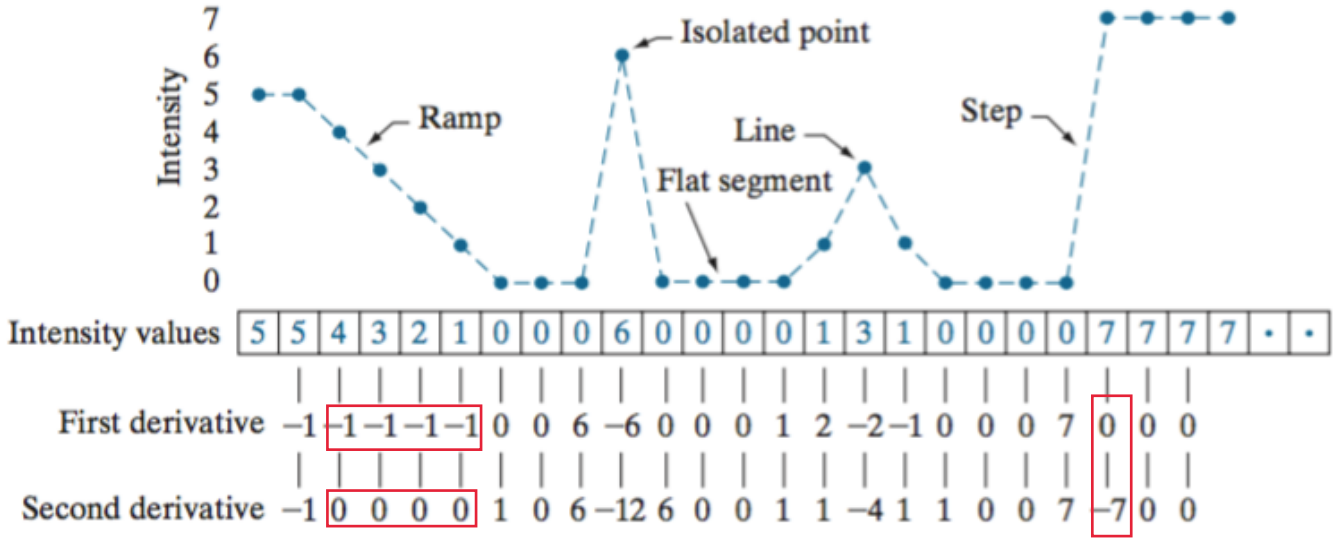
Points, Lines, and Edge detection

1st Derivatives

- Must be zero in areas of constant intensity;
- Must be nonzero at the onset of an intensity step or ramp;
- Must be nonzero at points along an intensity ramp

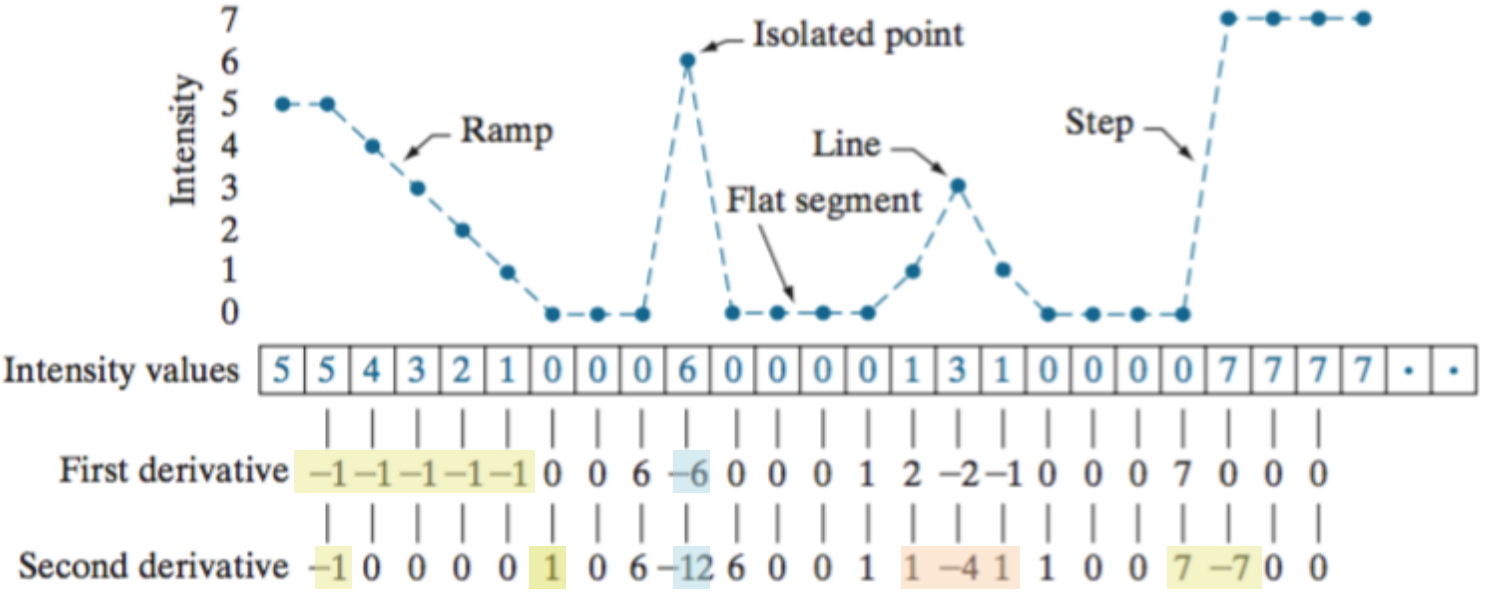
2nd Derivatives

- Must be zero in areas of constant intensity;
- Must be nonzero at the onset and end of an intensity step or ramp;
- Must be zero along an intensity ramp



1st and 2nd derivatives

- First-order derivatives generally produce thicker edges
- Second-order derivatives produce a double-edge response at ramp and step transitions in intensity
- Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise
- The sign of the second derivative can be used to determine whether a transition into an edge is from light to dark or dar to light



Detection of isolated points

Point detection should be based on 2nd derivatives

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

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

a b
c d

FIGURE 3.37
(a) Filter mask used to implement Eq. (3.6-6).
(b) Mask used to implement an extension of this equation that includes the diagonal terms.
(c) and (d) Two other implementations of the Laplacian found frequently in practice.

Detection of isolated points

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

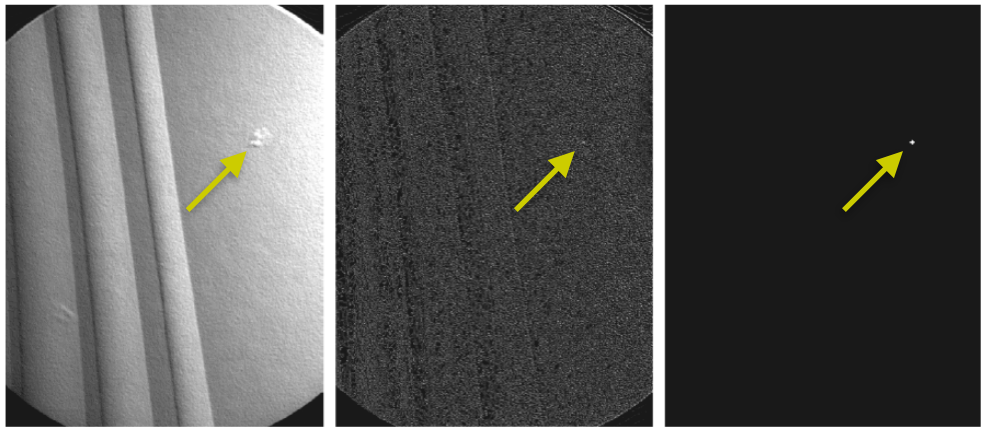
$$R = \sum_{k=1}^{9} w_k z_k$$

z_k = intensity of the corresponding pixel

$$\text{Output (image) } g(x,y) = \begin{cases} 1 \text{ if } |R(x,y)| \geq T \\ 0 \quad \text{otherwise} \end{cases}$$

T = threshold

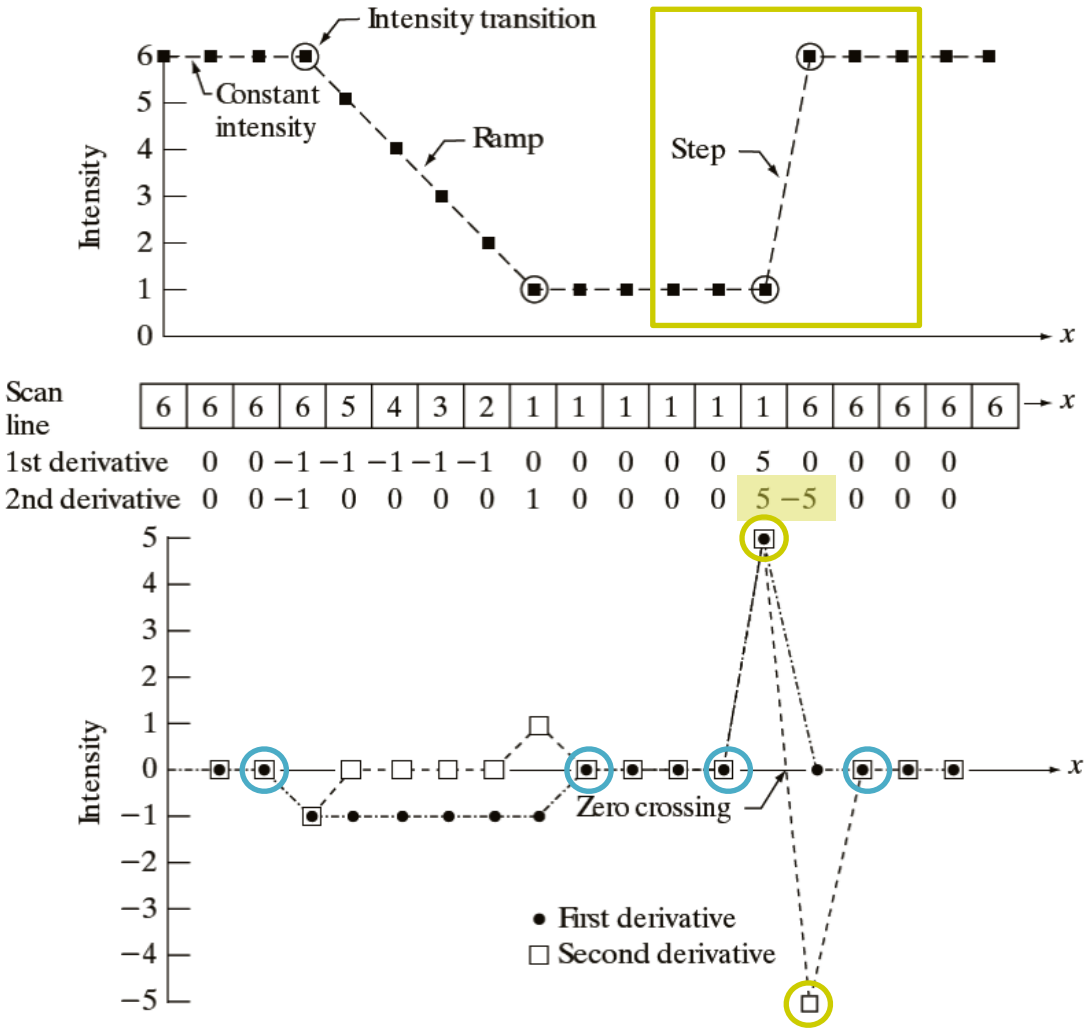
1	1	1
1	−8	1
1	1	1



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.)

Zero crossings



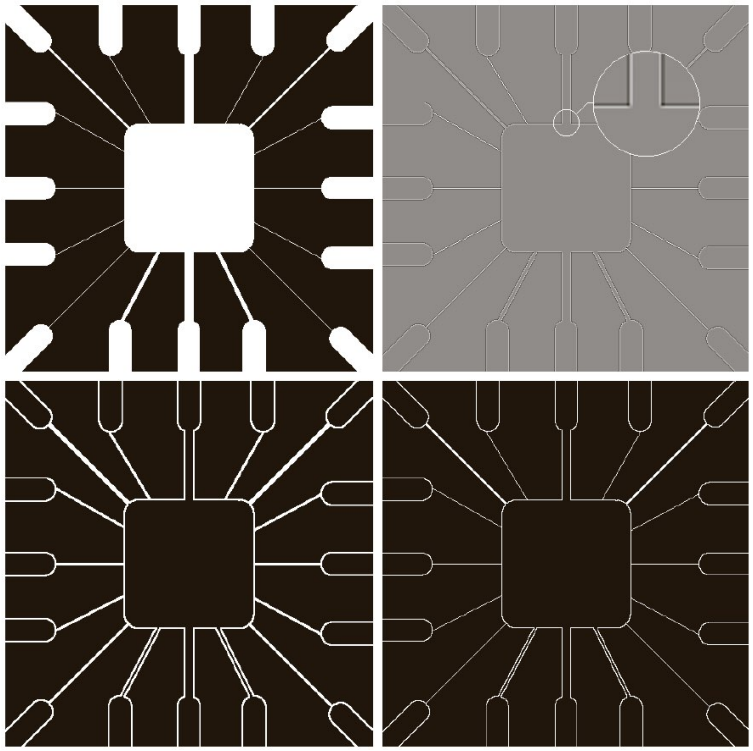
a
b
c

FIGURE 3.36 Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

Detection of lines

Second derivatives give stronger response and thinner lines compared to first derivatives, but show the double line effect.

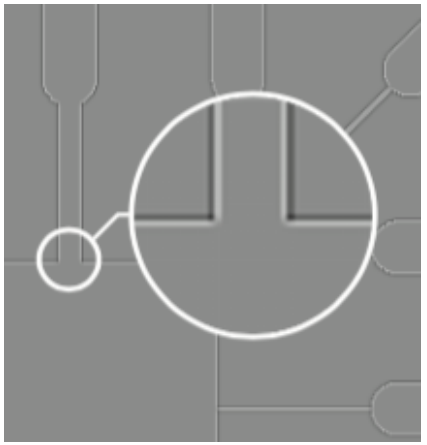
1	1	1
1	−8	1
1	1	1



Absolute values Positive values

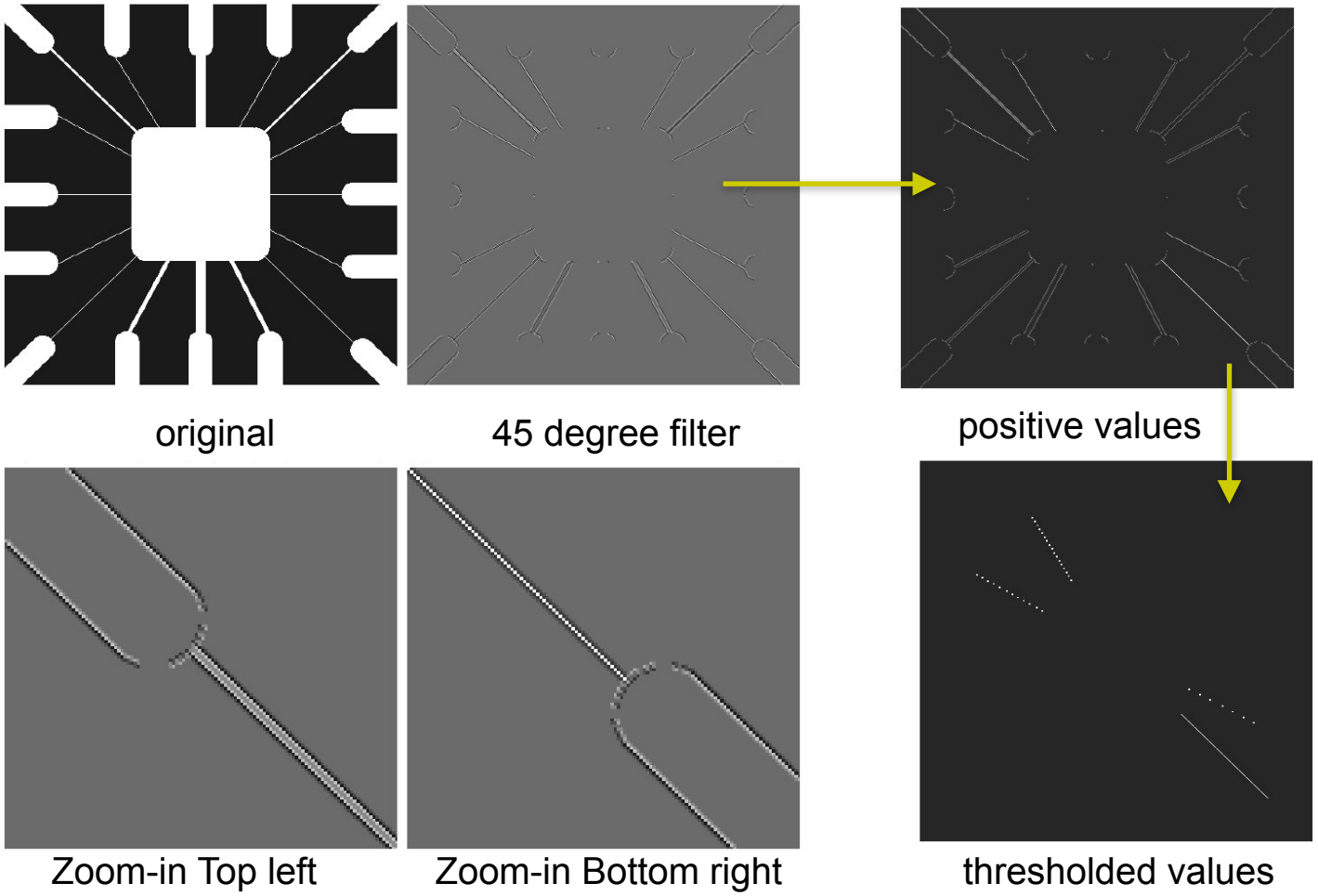
a b
c d

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.



−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
Horizontal			+45°			Vertical			−45°		

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

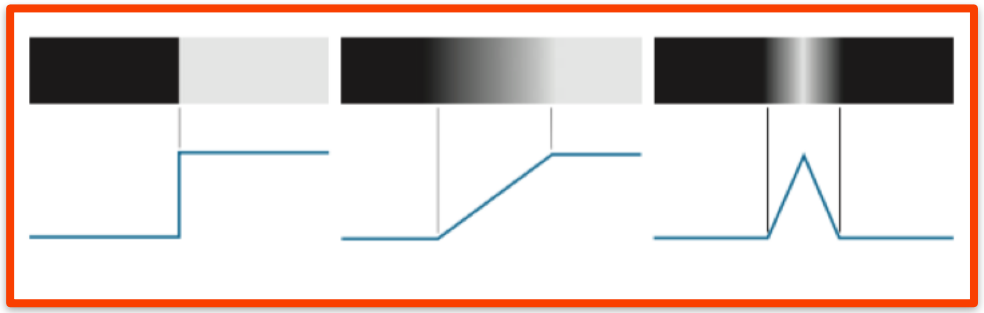


a b
c d
e f

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 and edge detection

Step edges = ideal case, used in CAD or animation.
Ramp, roof edges = more common in natural images



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.

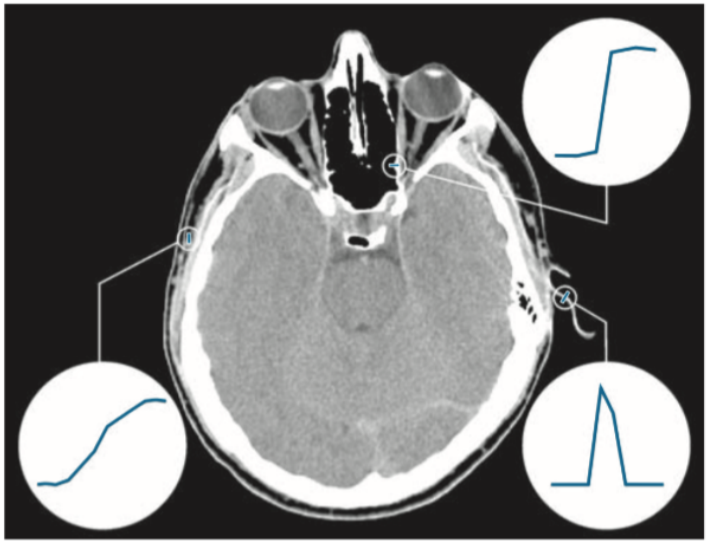
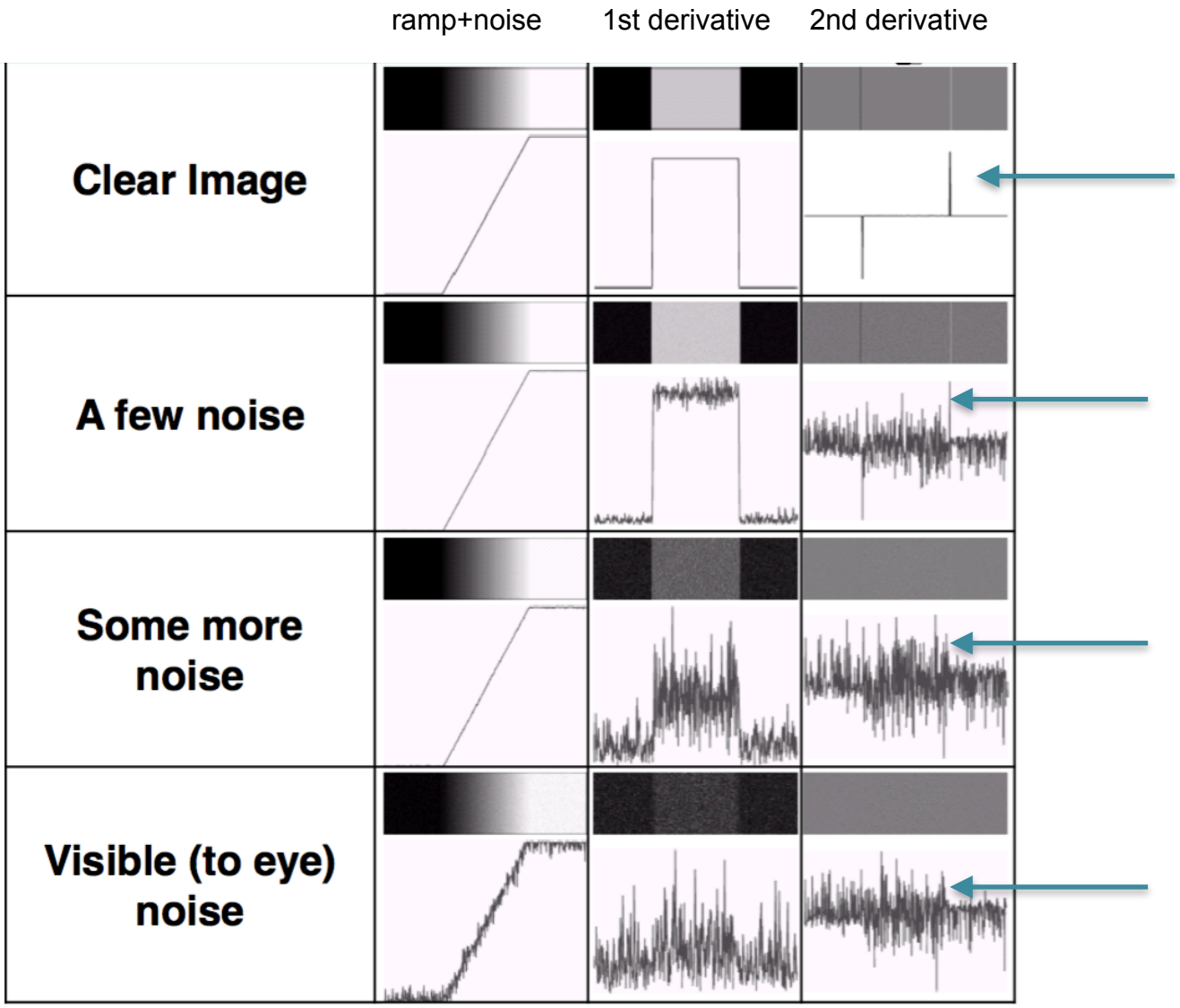


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively.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

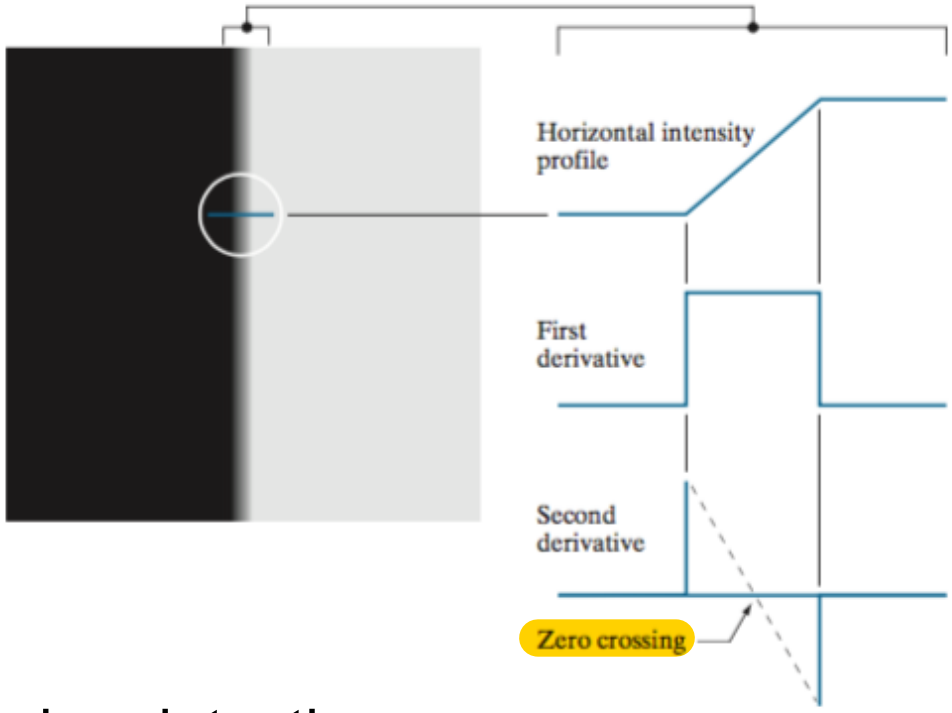
Edge models - sensitivity



Edge models and edge detection

a b

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



Three steps in edge detection:

- Image smoothing for noise reduction
- Detection of edge points: extract all points that are potential candidates for an edge.
- Edge localization: select all the points that are members of an edge.

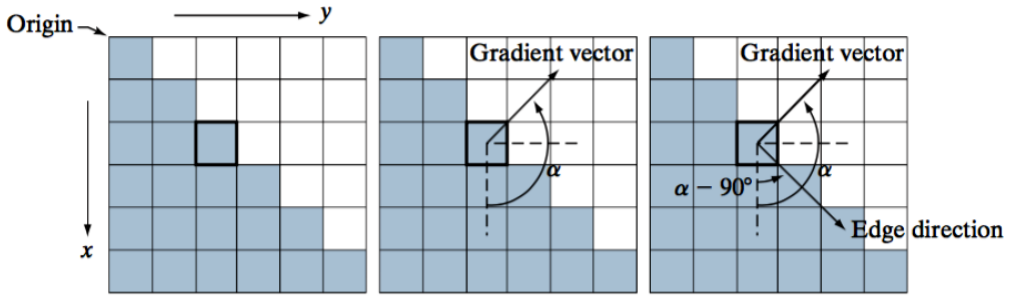
Edge detection

1. Use image gradients (1st derivatives)

$$\nabla f(x,y) = grad(f(x,y)) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\text{Magnitude (length) } M = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

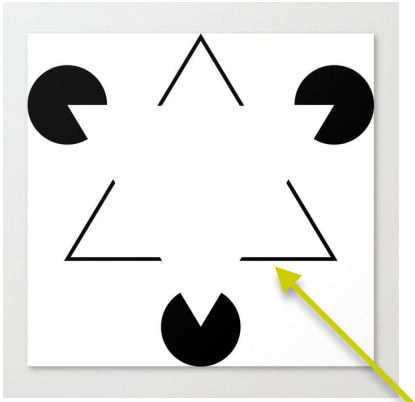
$$\text{Angle } \alpha(x,y) = \tan^{-1}\left(\frac{g_x}{g_y}\right); \; g_x = \frac{\partial f}{\partial x}, \ldots.$$



a b c

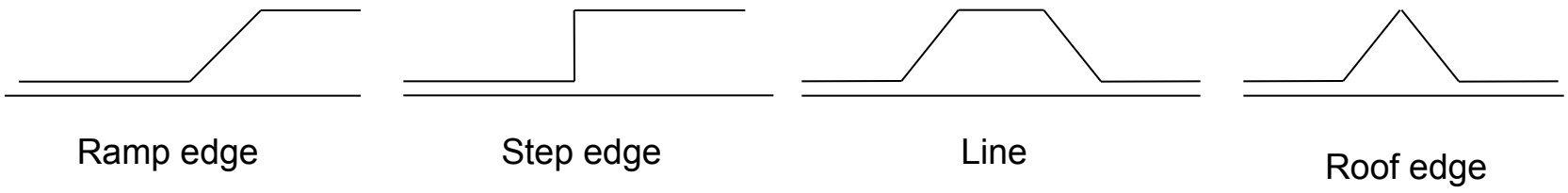
FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge direction is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square represents one pixel. (Recall from Fig. 2.19 that the origin of our coordinate system is at the top, left.)

Edge models



a white triangle?

- Challenges in edge detection
  - **Unclear (or smoothed) edges in real image**
  - **Ambiguity due to shadow, light or pattern**
- Most classical methods are based on differential operations since they measure the rate of change in a function.
- Edge models can be considered:



Edge detection *(1st derivatives)*

−1
1

−1	1

a b

FIGURE 10.13 One-dimensional masks used to implement Eqs. (10.2-12) and (10.2-13).

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

Masks size 2x2 (Roberts) are not as useful for computing edge directions as masks that are asymmetric about the centre point – the smallest size is 3x3.

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.

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

Prewitt

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

Sobel

a b
c d

FIGURE 10.15 Prewitt and Sobel masks for detecting diagonal edges.

Take a break!

Sobel Operator

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

(Non-smoothed image)

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

gradient image

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, $|g_x| + |g_y|$.

$$g_x = \frac{\partial f}{\partial x}$$

Sobel Operator - image smoothing

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

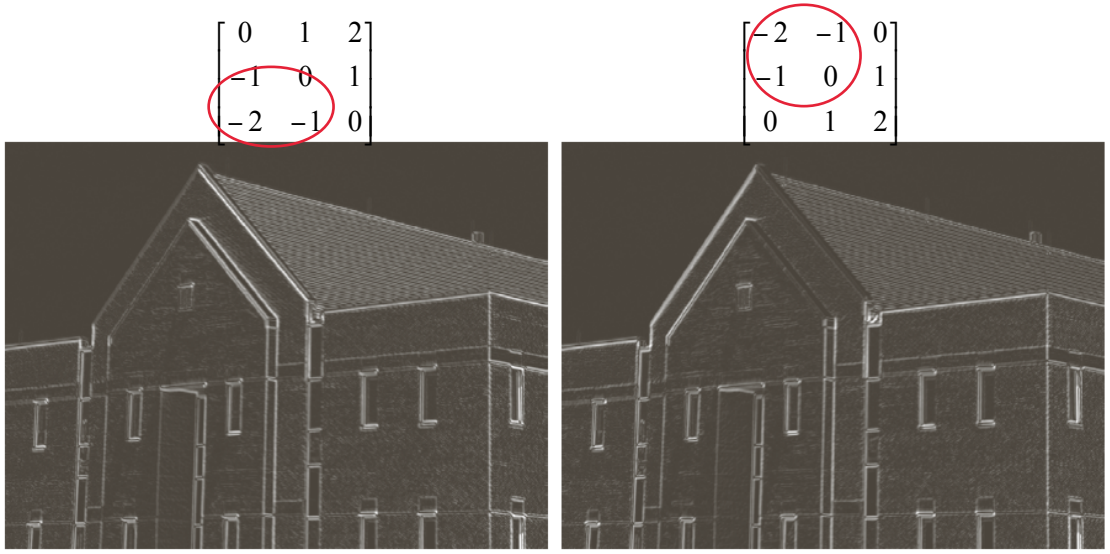
(Smoothed image)

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

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.

Sobel Operator - edge orientation



a b

FIGURE 10.19 Diagonal edge detection. (a) Result of using the mask in Fig. 10.15(c). (b) Result of using the mask in Fig. 10.15(d). The input image in both cases was Fig. 10.18(a).

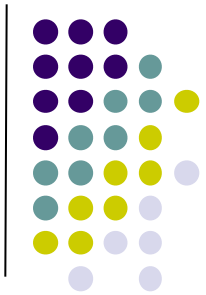
a b

no smoothing

with smoothing

FIGURE 10.20 (a) Thresholded version of the image in Fig. 10.16(d), with the threshold selected as 33% of the highest value in the image; this threshold was just high enough to eliminate most of the brick edges in the gradient image. (b) Thresholded version of the image in Fig. 10.18(d), obtained using a threshold equal to 33% of the highest value in that image.

Roberts Operator (1st derivatives)



- Roberts operator marks edge points only
  - **Any information about edge orientation is not obtained.**
- One of the simplest methods
- Work best with binary images
- Form 1

$$\sqrt{\big(u(x,y)-u(x-1,y-1)\big)^2+\big(u(x-1,y)-u(x,y-1)\big)^2}$$

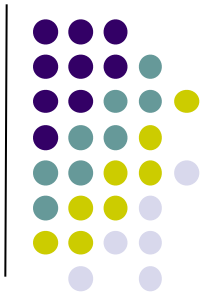
- Form 2

$$\big|u(x,y)-u(x-1,y-1)\big|+\big|u(x-1,y)-u(x,y-1)\big|$$

- Form 2 is often used in practice due to its computational efficiency.

(1965)

Sobel & Prewitt Operators (1st derivatives)



- Sobel edge detection masks look for edges in both the horizontal and vertical directions.
- Then combine both directions into a single metric.
- Masks:

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$ **Row mask** → s_1

s_1 & s_2 are masked pixel values

$$\text{Edge magnitude}: \sqrt{s_1^2 + s_2^2}$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$ → s_2 **Column mask**

$$\text{Edge direction}: \tan^{-1}\left[\frac{s_1}{s_2}\right]$$

- Prewitt Operator
  - Same as Sobel except masks

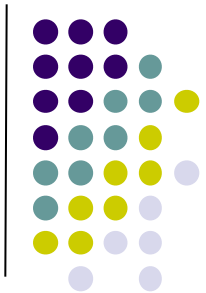
Row mask

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

Column mask

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

Compass Operators (1st derivatives)

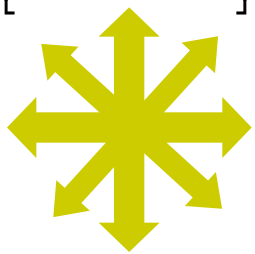


- Measure gradients in a selected number of directions:

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

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

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



shown with **Prewitt**

- Edge magnitude is defined as the maximum value found by all directional masks.

Laplacian Operator (2nd derivatives)

(The Marr-Hildreth edge detector)

The Laplacian of a Gaussian : $G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$

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

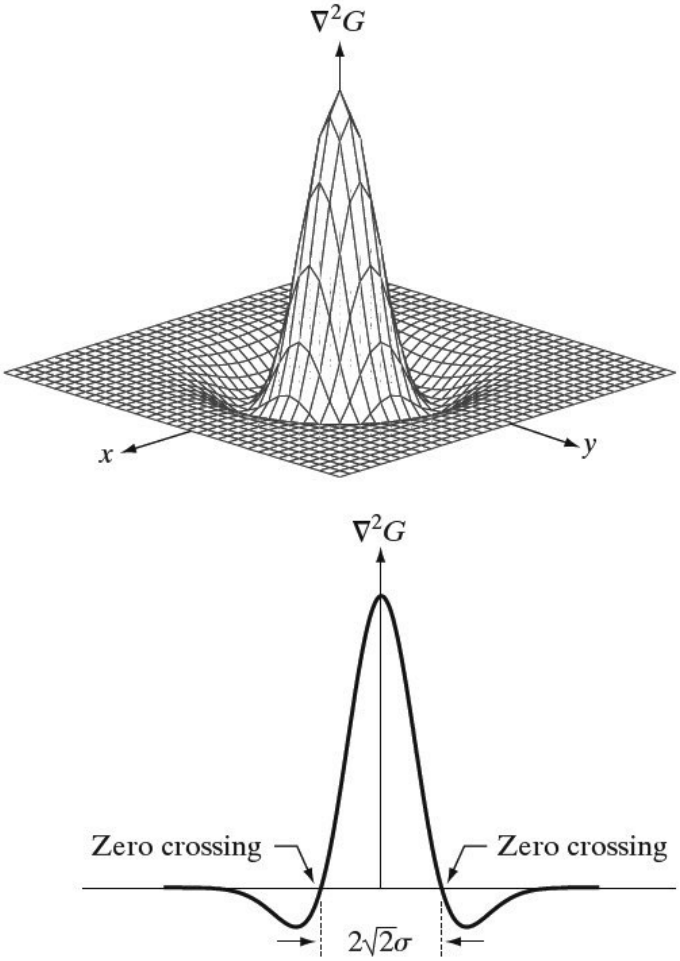
$$= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

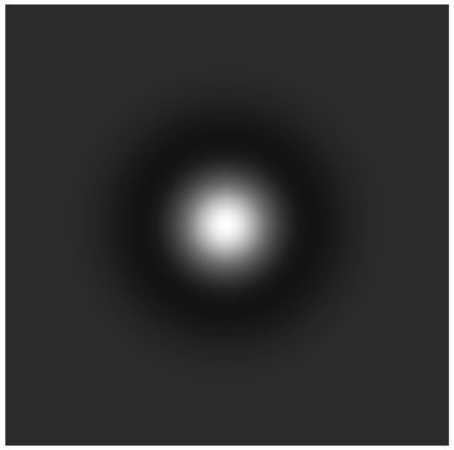
$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

This is called the *Laplacian of a Gaussian* (*LoG*)

Laplacian of a Gaussian



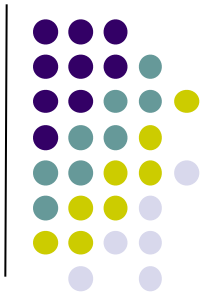


0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

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.

The Marr-Hildreth algorithm:



$$g(x,y) = [\nabla^2 G(x,y)] * f(x,y) \Rightarrow \text{LoG} * image \quad (1)$$

Because these are linear processes:

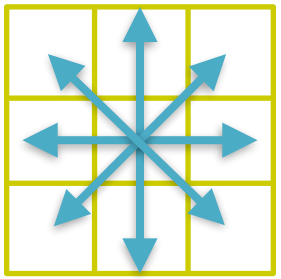
$$g(x,y) = \nabla^2 [G(x,y) * f(x,y)] \quad (2)$$

1. Filter the input image by an *nxn* Gausian lowpass filter.
2. Compute the Laplacian of the resulting image using for example the *3x3* Laplacian mask.
3. Find the zero crossings of the image from step 2).

Note: *You could use either Eq. (1) or Eq. (2), they would give the same results*.

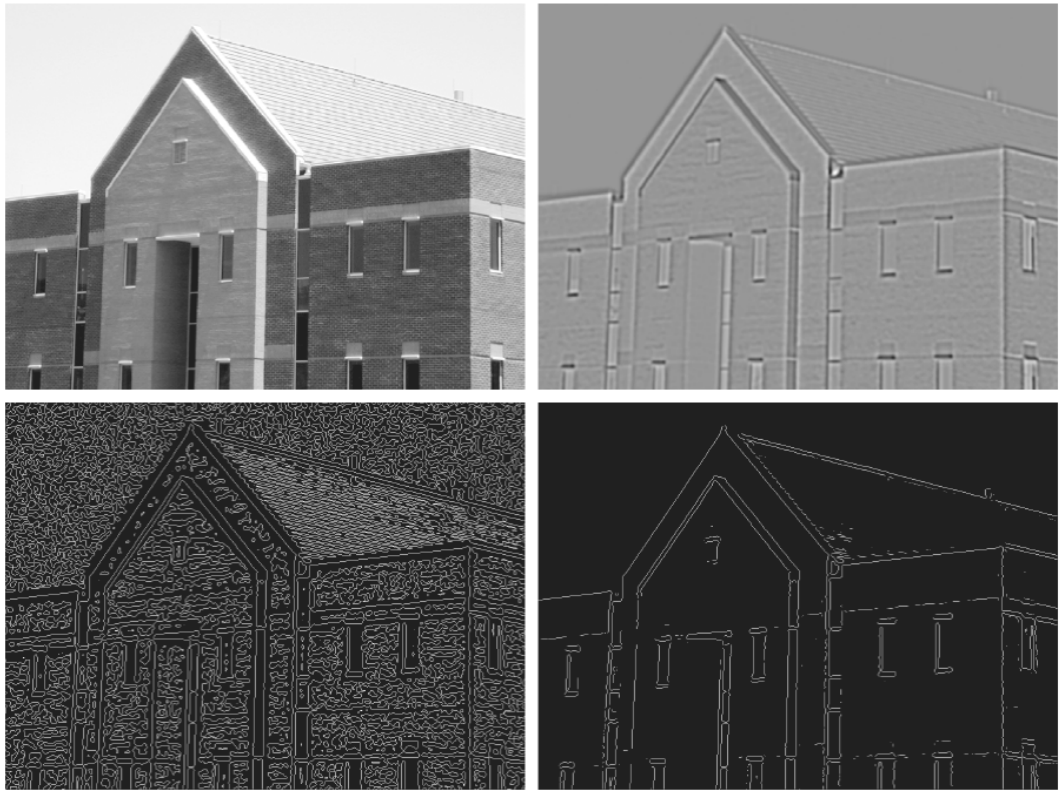
Finding zero crossings

To find the zero crossing at a pixel *p* of the filtered image *g(x,y), one approach can be:*



- Use a 3x3 neighbourhood mask centred at p
- Zero-crossing = the sign of at least 2 opposing neighbours must be different, resulting in four cases to test:
  - ✓left/right, up/down, and the two diagonals
- If a threshold is needed, then not only the signs need to be different, the absolute values of the numerical differences must also exceed the threshold

The Marr-Hildreth algorithm:



threshold = 0

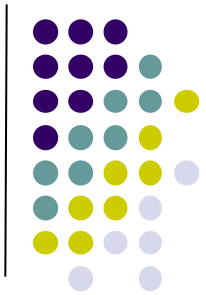
threshold = 4% of max

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.

Note: *Zero crossings should be found from sign changes not from zero values which are unstable.*

Canny edge detector



One of classical techniques for edge detection

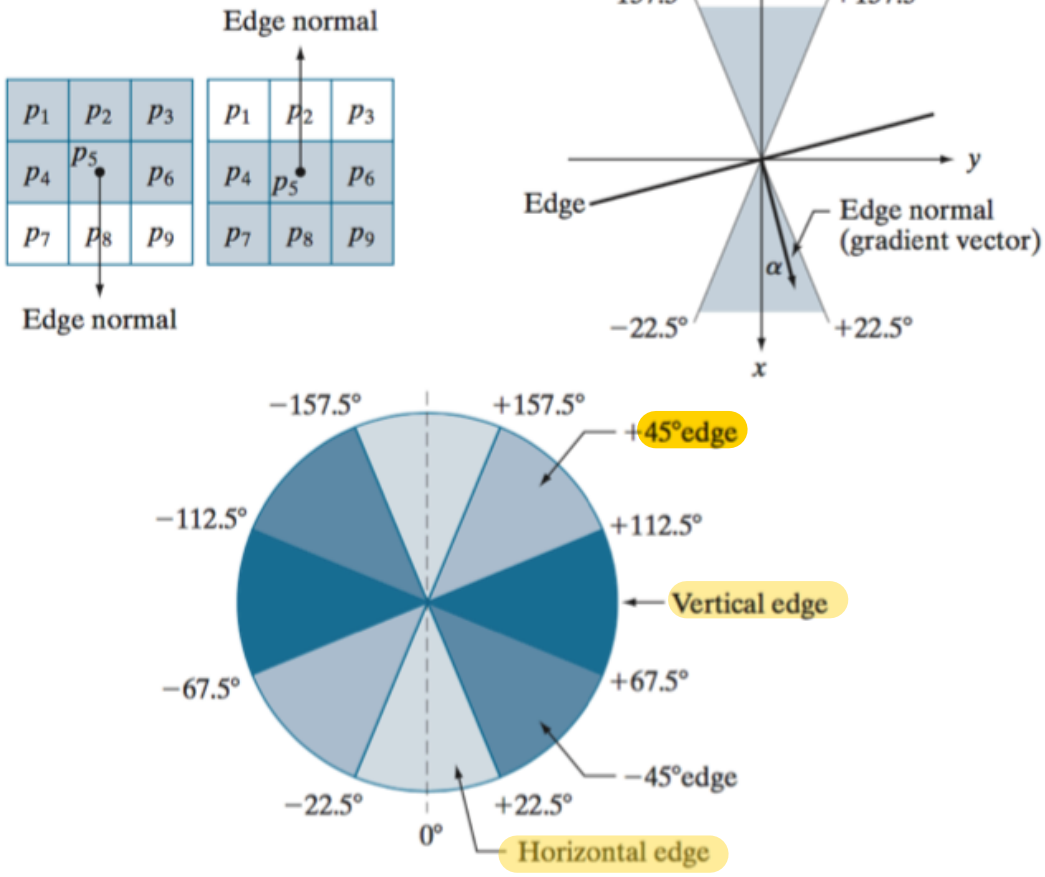
"A computational Approach to Edge Detection,"
J. Canny, IEEE PAMI, no.6, pp 679-698,1986

- More complex but has better performance than all methods discussed so far.
- Based on first order derivatives, and directions.
- Smoothed image: $f_{smooth}(x,y) = G(x,y) * f(x,y)$
- Compute the gradient magnitude and directions:

$$M(x,y) = \sqrt{g_x^2 + g_y^2} \quad \text{and} \quad \alpha(x,y) = \tan^{-1}\left(\frac{g_x}{g_y}\right)$$

- g_x and g_y can be calculated using Sobel or Prewitt operators.
- The directions are quantized as follows:

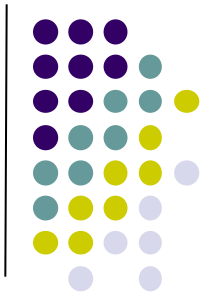
Directional Maps in Canny Detector



a b
c

FIGURE 10.24
(a) Two possible orientations of a horizontal edge (shaded) in a 3×3 neighborhood. (b) Range of values (shaded) of α, the direction angle of the edge normal for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3×3 neighborhood. Each edge direction has two ranges, shown in corresponding shades.

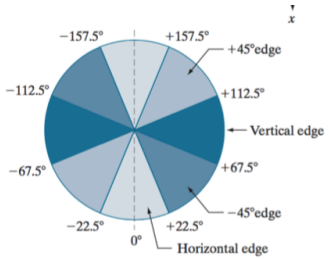
Canny edge detector



- Canny defined a *comprehensive* set of goals for the computation of edge points.
- Three criteria
  - Low error rate (find all edges and no responses to non-edges)
  - Well-localized edge points
    - Distance between detected edge pixels and the actual edge should be as close as possible.
  - Only one response to a single edge.
- Algorithm
  - Step 1: Smoothing image (using Gaussian filter)
  - Step 2: Obtain edge magnitude using Sobel operator
  - Step 3: Obtain edge direction using Sobel operator

Canny edge detector

- Algorithm *(cont.)*
  - Step 4: Relating edge directions obtained from Step 3
    - For example, 8 mask orientations can be used.
    - Need to group them into the discrete bins



  - Step 5: Non-maximum suppression
    - Follow the edge points in the edge direction
    - Suppress any pixel value that is not considered to be an edge, (i.e., not at the maximum).
    - This approach generates a thin line for the edge map.
  - Step 6: Eliminating streaking
    - Edge contour can be broken up by noise and a low threshold.
    - Streaking can be eliminated by using two thresholds

Canny edge detector

- Algorithm *(cont.)*
  - Step 6: Eliminating streaking: detect and link edges
    - Edge contour can be broken up by noise and a low threshold.
    - Streaking can be eliminated by using two thresholds

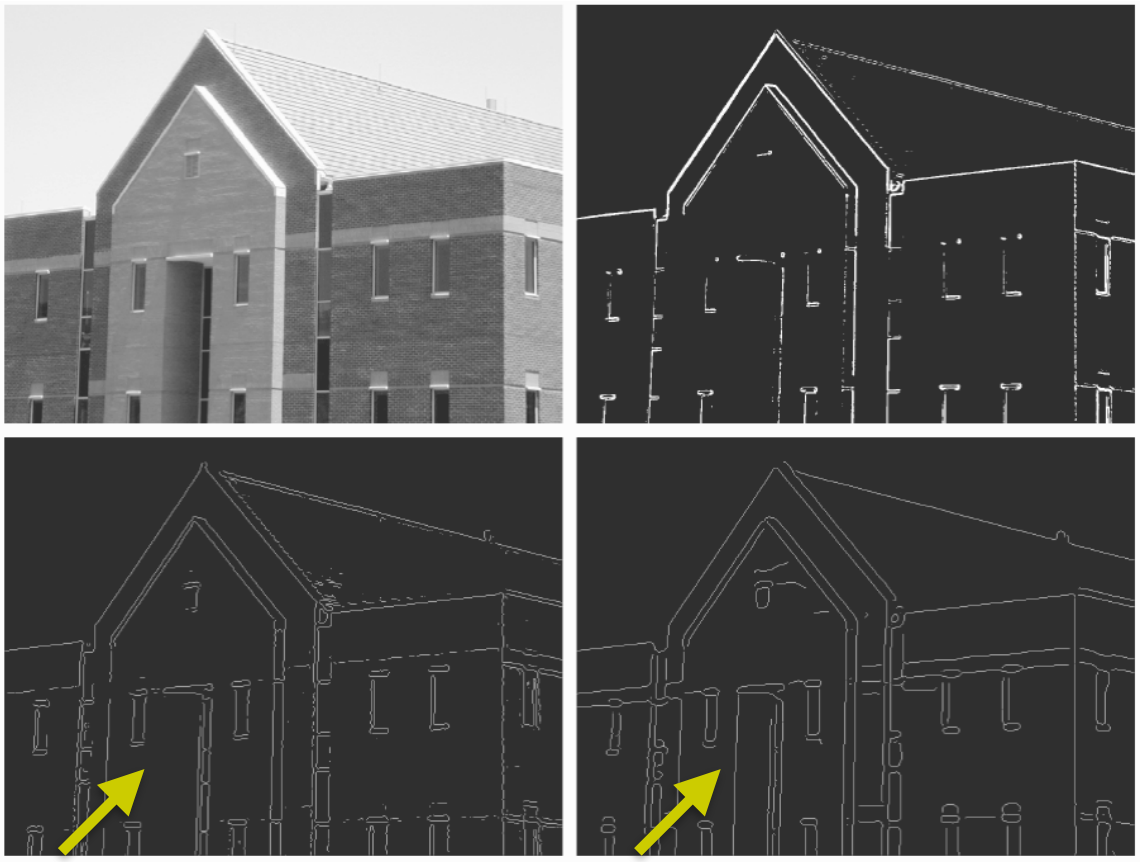
- If $Value \geq T_1$ it is an edge point;
- else if $T_2 \leq Value < T_1$ and a connected pixel value $\geq T_1$, it is an edge point.
- else non - edge.

T1/T2 ——> 2/1 ~ 3/1

You can also take a look at this video that explains canny edge detection

https://www.youtube.com/watch?v=17cOHpSaqi0

Comparison of Marr-Hildreth and Canny edge detectors

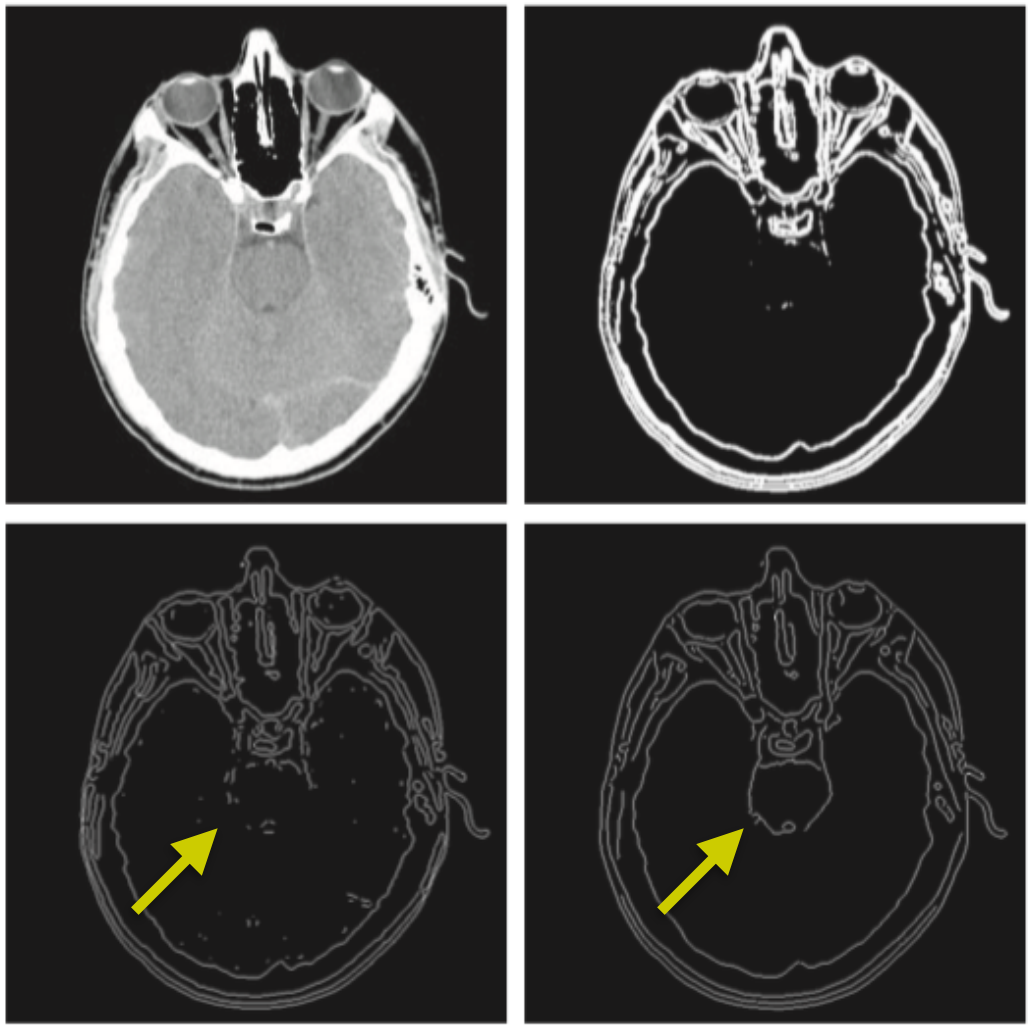


Marr-Hildreth Canny

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.

Comparison of Marr-Hildreth and Canny edge detectors

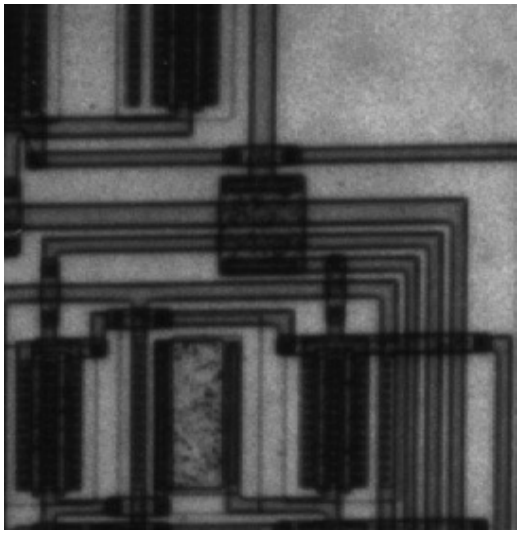


Marr-Hildreth Canny

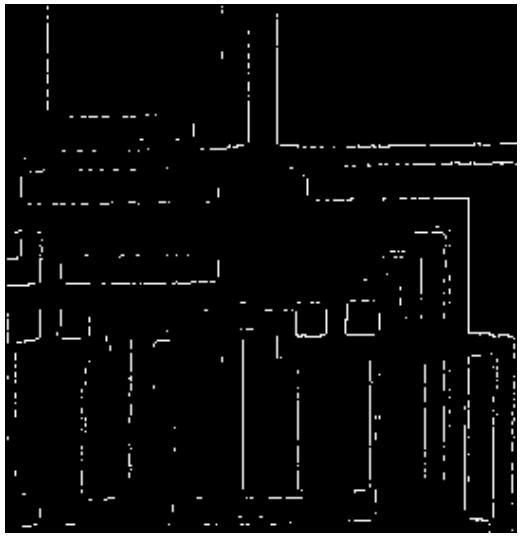
a b
c d

FIGURE 10.26
(a) Head CT image of size 512×512 pixels, with intensity values scaled to the range $[0, 1]$.
(b) Thresholded gradient of the 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.)

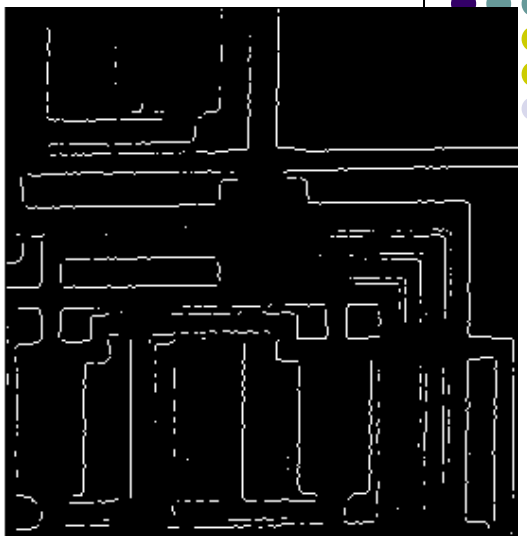
Classical techniques for edge detection



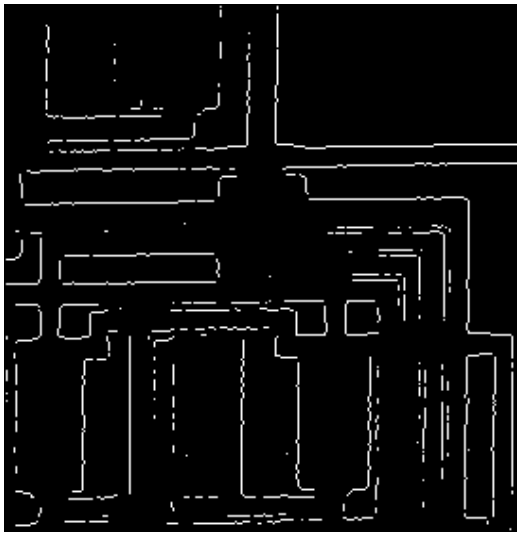
Original image



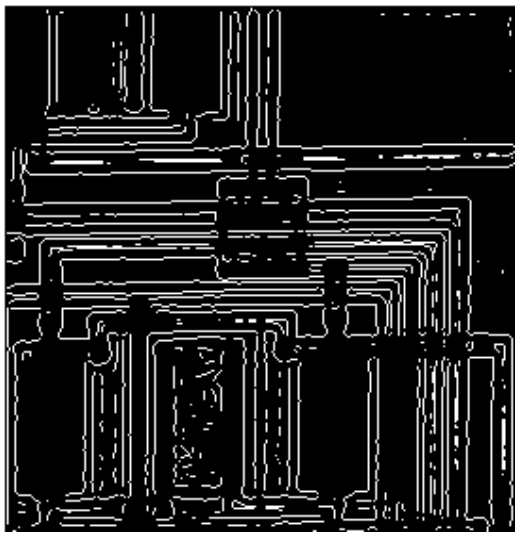
Roberts operator



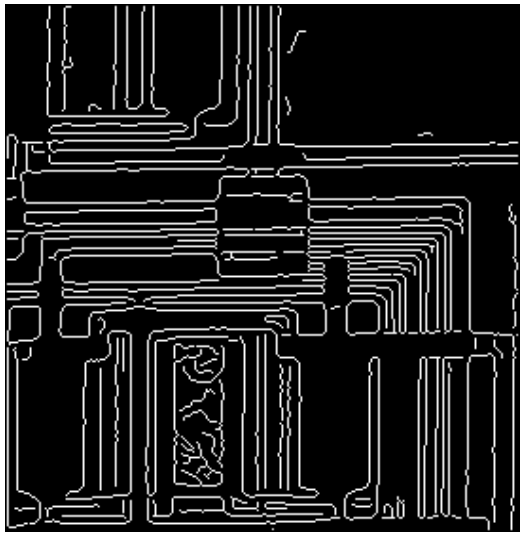
Sobel operator



Prewitt operator



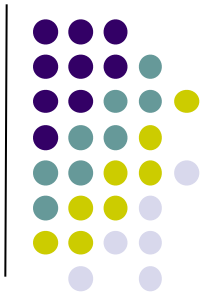
Laplacian operator



Canny

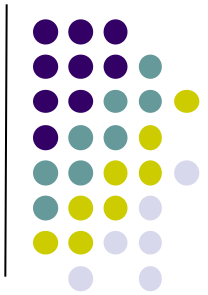
Images generated by MATLAB Image Processing Tools

In summary



- Different edge models
- Point and line detection with 2nd derivative filters
- Characteristics of 1st and 2nd derivative edge detection
- Edge detection (1st and 2nd derivative filters)
- Marr-Hildreth/LoG and **Canny algorithms**

Reading materials



- Chapter 10
- Page 700 -734

Questions

1. Please describe the characteristics of a derivative type of filter (e.g., laplacian filter)?
2. When we need to compute gradients for edges, why do we prefer a minimum size of 3x3 for the filters in practice?
3. Why the impacts of LoG and the sequence of first performing Gaussian filtering and then applying laplacian filter are the same?
4. Describe the algorithm for detecting zero-crossings.
5. How does Canny edge detector link edges?
6. Noise tolerance: 1st order derivative filter vs. 2nd order derivative filter
7. How does a compass operator work for edge detection?