

ECG782: MULTIDIMENSIONAL DIGITAL SIGNAL PROCESSING IMAGE SEGMENTATION

OUTLINE

- Fundamentals
- Point, Line, and Edge Detection
- Thresholding
- Region-Based Segmentation

SEGMENTATION

- Transition toward more high level systems/analysis
 - Now: Input = images \rightarrow output = attributes of regions or objects
 - Image processing : input = image \rightarrow output = image
- Important but difficult task as part of image understanding pipeline
 - Best to control system as much as possible (e.g. lighting in a factory inspection)
 - When observation control is limited need to consider sensing technology (e.g. thermal vs. visual imaging)
 - Practically, may not want to limit to just imaging
- Operate using intensity similarity and discontinuity
 - Regions vs. edges

FUNDAMENTALS

- Divide image into parts that correlate with objects or “world areas”
 - Important step for image analysis and understanding
- Complete segmentation
 - Disjoint regions corresponding to objects
 - $R = \bigcup_{i=1}^n R_i, \quad R_i \cap R_j = \emptyset, \quad i \neq j$
 - Typically requires high level domain knowledge
- Partial segmentation
 - Regions do not correspond directly to objects
 - Divide image based on homogeneity property
 - Brightness, color, texture, etc.
 - $Q(R_i) = \text{TRUE}$ and $Q(R_i \cup R_j) = \text{FALSE}$
 - High-level info can upgrade partial segmentation to complete segmentation
- Main goal is reduction in data volume for higher level processing

FUNDAMENTALS II

- Monochrome segmentation based on either intensity discontinuity or similarity
- Discontinuity
 - Edge-based segmentation
 - Boundaries of regions are distinct
- Similarity
 - Region-based segmentation
 - Image partitions are formed by similar areas (based on some criteria $Q(.)$)

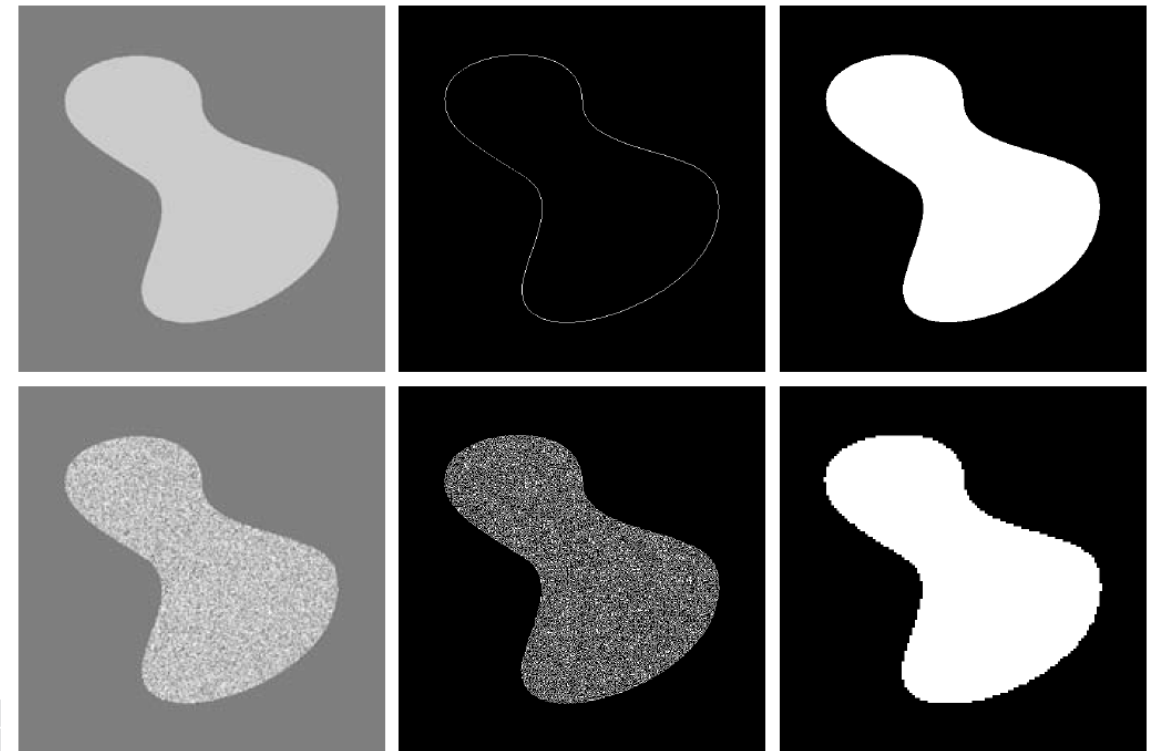


FIGURE 10.1 (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large number of small edges that are connected to the original boundary, making it difficult to find a unique boundary using only edge information. (f) Result of segmentation based on region properties.

POINT, LINE, AND EDGE DETECTION

- Look for sharp “local” changes in intensity
- All require the use of derivatives
 - $\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x)$
 - Thick edges
 - $\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$
 - Fine edges (more aggressive)
 - Double response
 - Sign determines intensity transition
- Edge
 - Edge pixels – pixels at which the intensity function changes abruptly
 - Edges (segments) – are connected edge pixels

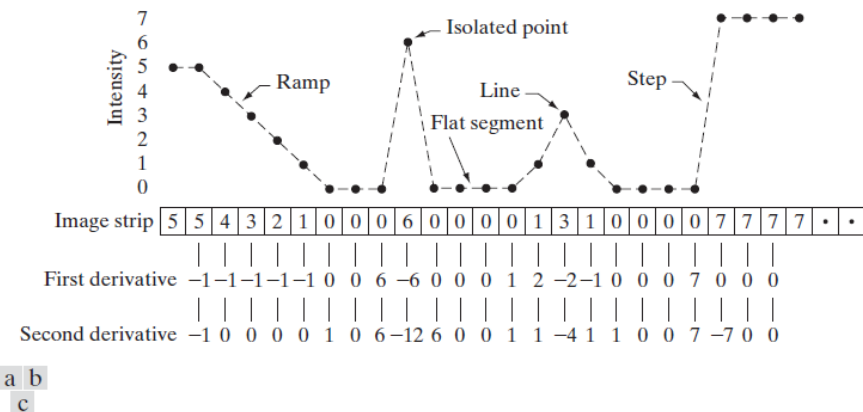
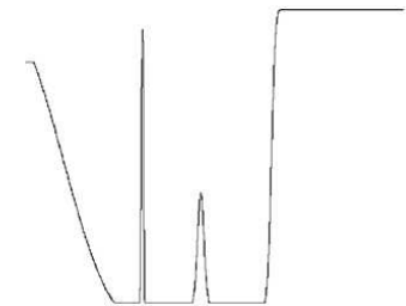


FIGURE 10.2 (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values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

EDGE DETECTION

- Locate changes in image intensity function
 - Edges are abrupt changes
- Very important pre-processing step for many computer vision techniques
 - Object detection, lane tracking, geometry
- Edges are important neurological and psychophysical processes
 - Part of human image perception loop
 - Information reduction but not understanding
- Edgels – edge element with strong magnitude
 - Pixels with large gradient magnitude



INFORMATIVE EDGES

- Edges arise from various physical phenomena during image formation
 - Trick is to determine which edges are most important

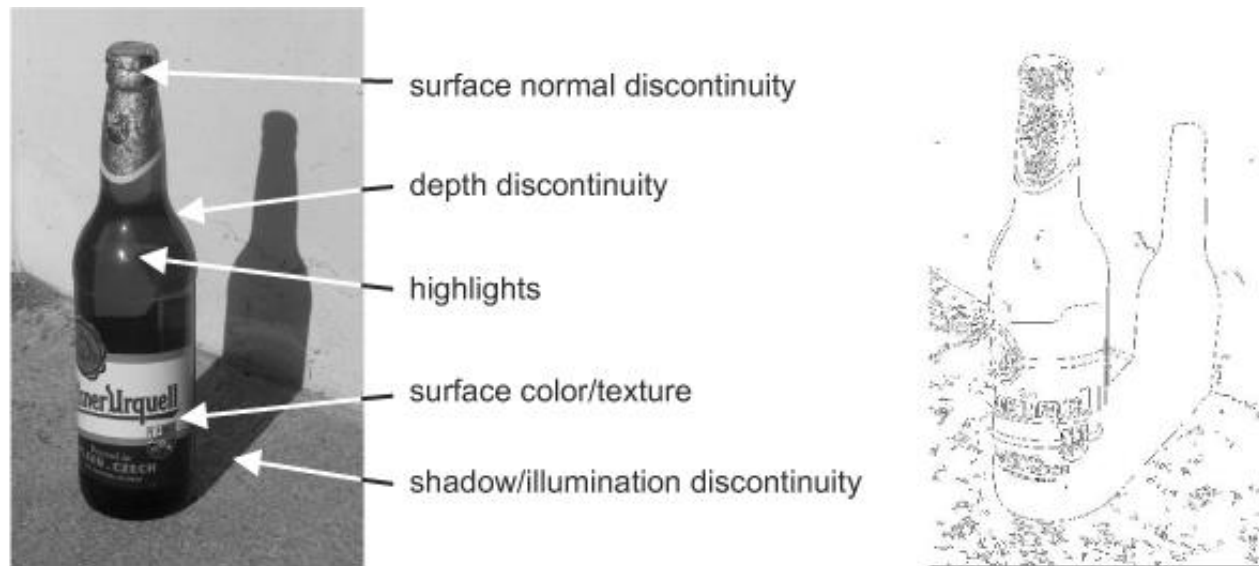


Figure 5.15: Origin of edges, i.e., physical phenomena in image formation process which lead to edges in images. At right, a Canny edge detection (see Section 5.3.5). © Cengage Learning 2015.

ISOLATED POINT DETECTION

- Use of second order derivative
 - More aggressive response to intensity change
- Laplacian
 - $\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
 - $\nabla^2 f(x, y) = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y)$
- Output from thresholding

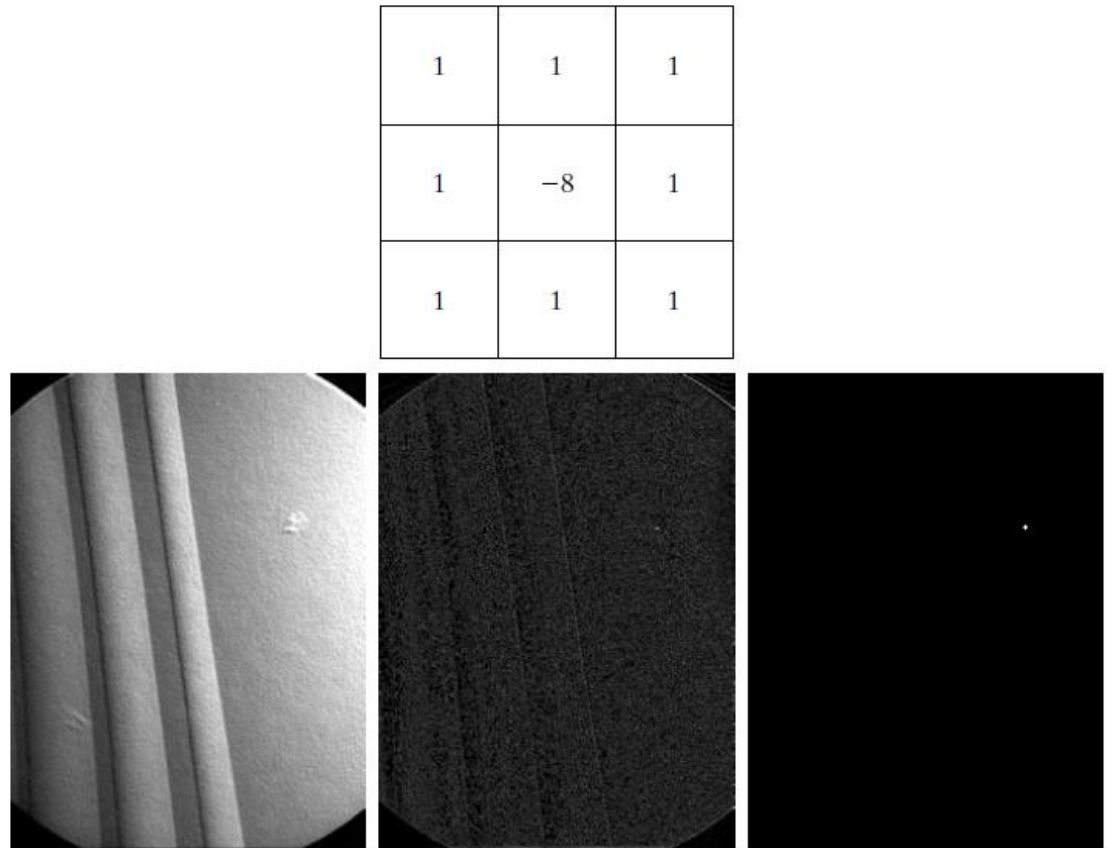


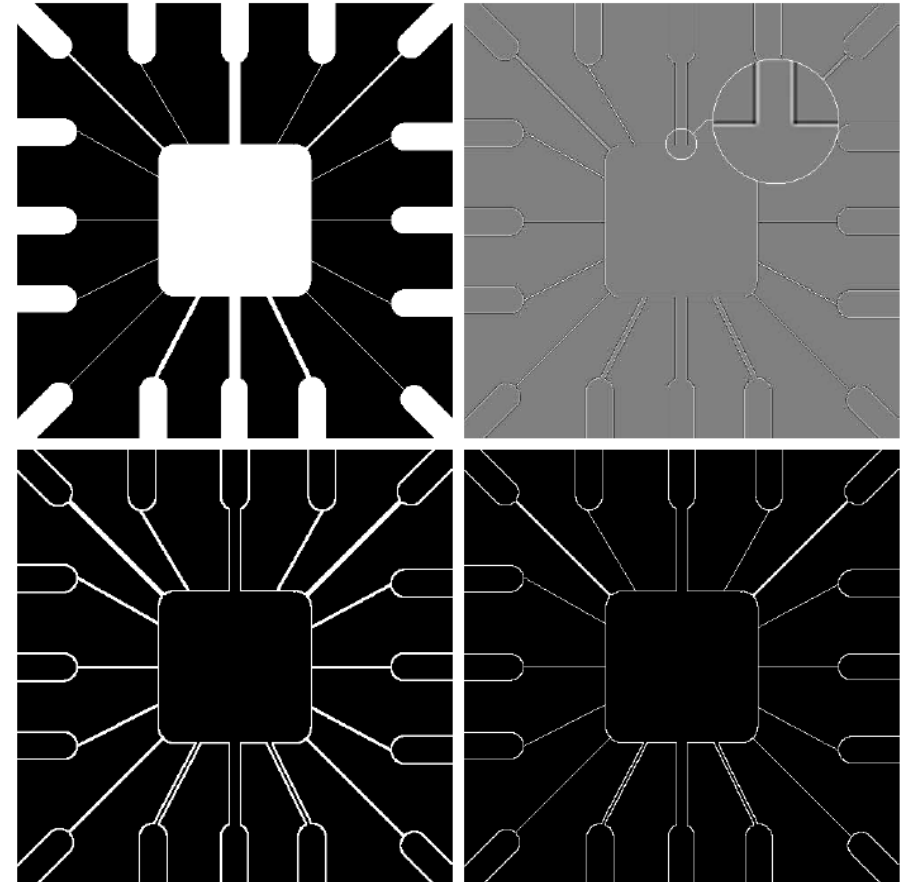
FIGURE 10.4
(a) Point detection (Laplacian) mask. (b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. (c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see).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

LINE DETECTION

- Again use Laplacian
 - Lines are assumed to be thin with respect to the size of the Laplacian kernel
- Be aware that Laplacian produces double response to a line
 - Positive response on one side of line
 - Negative response on the other side
- Typically, thin lines are required
 - Must appropriately select value (e.g. positive response)

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.

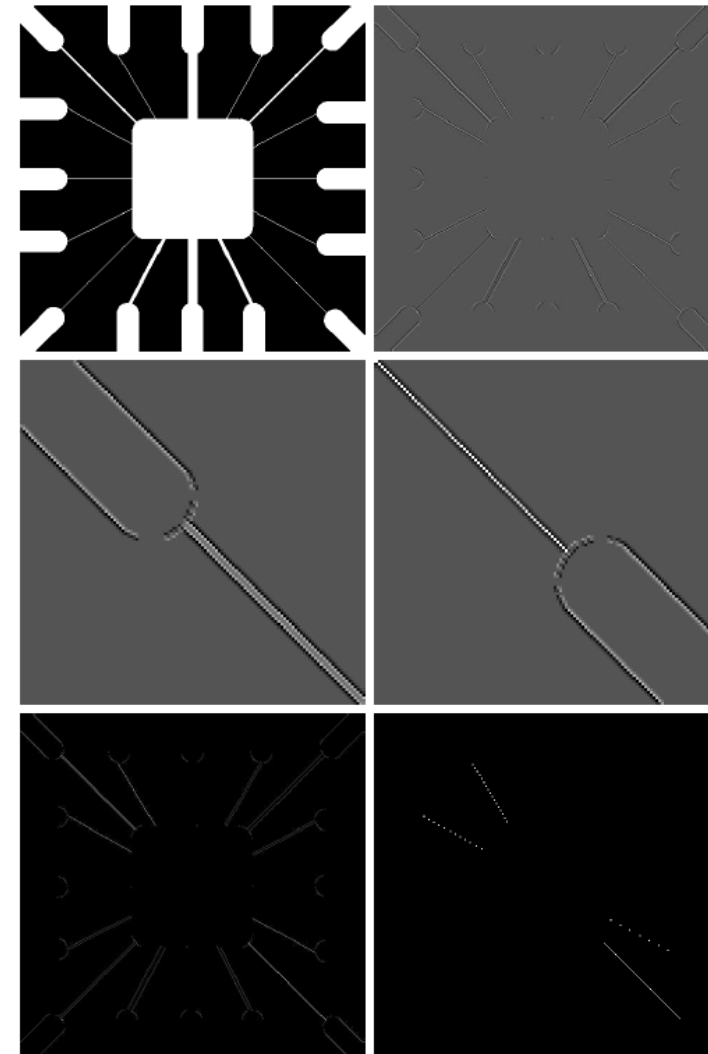


LINE DETECTION II

- Edges at a particular orientation can be detected
 - Adjust kernel to match desired direction

-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

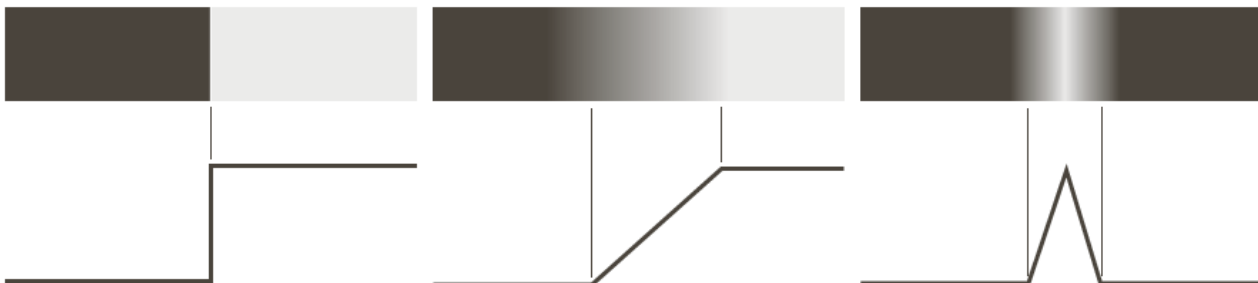
FIGURE 10.7
(a) Image of a wire-bond template.
(b) Result of processing with the +45° line detector mask in Fig. 10.6.
(c) Zoomed view of the top left region of (b).
(d) Zoomed view of the bottom right region of (b).
(e) The image in (b) with all negative values set to zero.
(f) All points (in white)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

- Classified according to intensity profiles
 - Step (ideal) edge – transition between two (large) intensity levels in a small (1 pixel) distance
 - Ramp edge – “real” edge thicker than 1 pixel width due to blurring of ideal edge
 - Roof edge – blurred line to have thickness



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.)



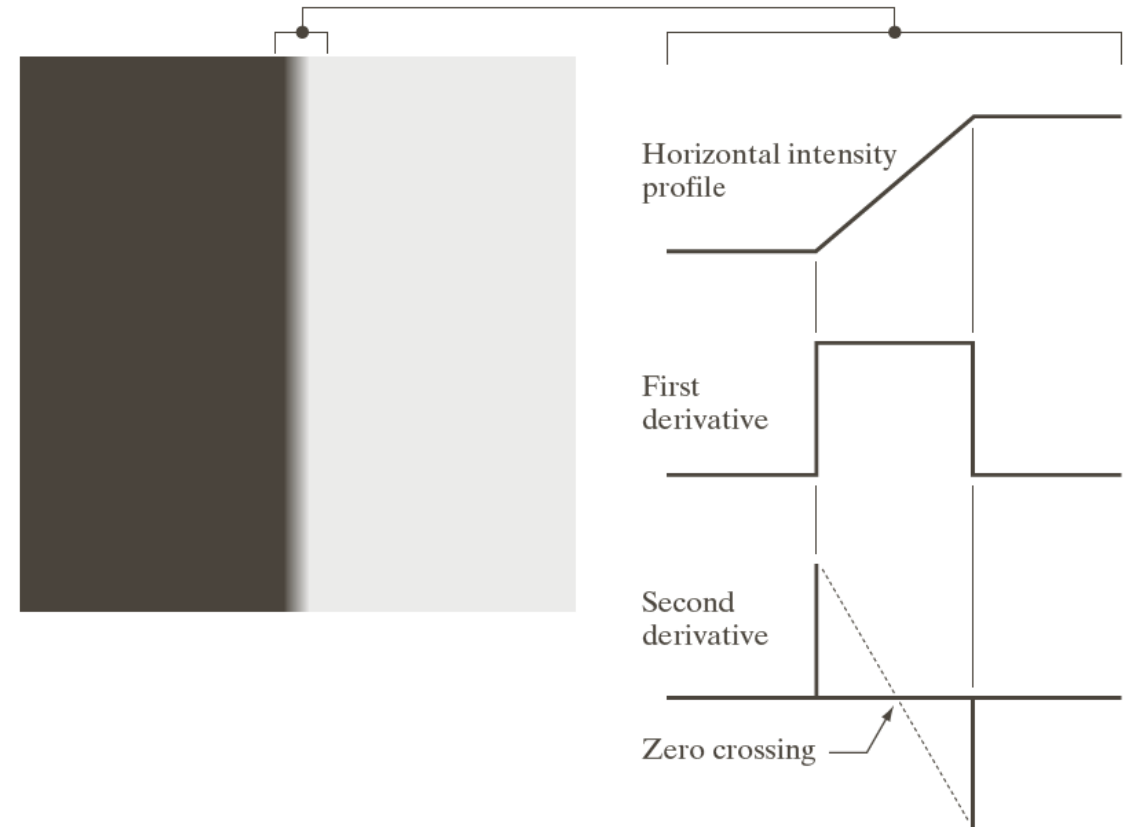
a b c

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

Real Edges

EDGE DERIVATIVES

- First derivative
 - Constant along ramp
 - Magnitude used to detect edge
- Second derivative
 - Dual response to ramp
 - Sign used to determine whether edge pixel is in dark or light side of edge
- Zero-crossing used to detect center of a thick edge



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.

REAL EDGES WITH NOISE

- Real images will have noise that corrupt the derivative operation
 - Remember this is a high pass filter
- Second derivative very sensitive to noise
 - Even small amounts of noise make it impossible to use
- First derivative less sensitive
- Three steps for edge detection
 - Image smoothing for noise reduction
 - Detection of edge points (1st or 2nd derivative)
 - Edge localization to select only true edge pixels

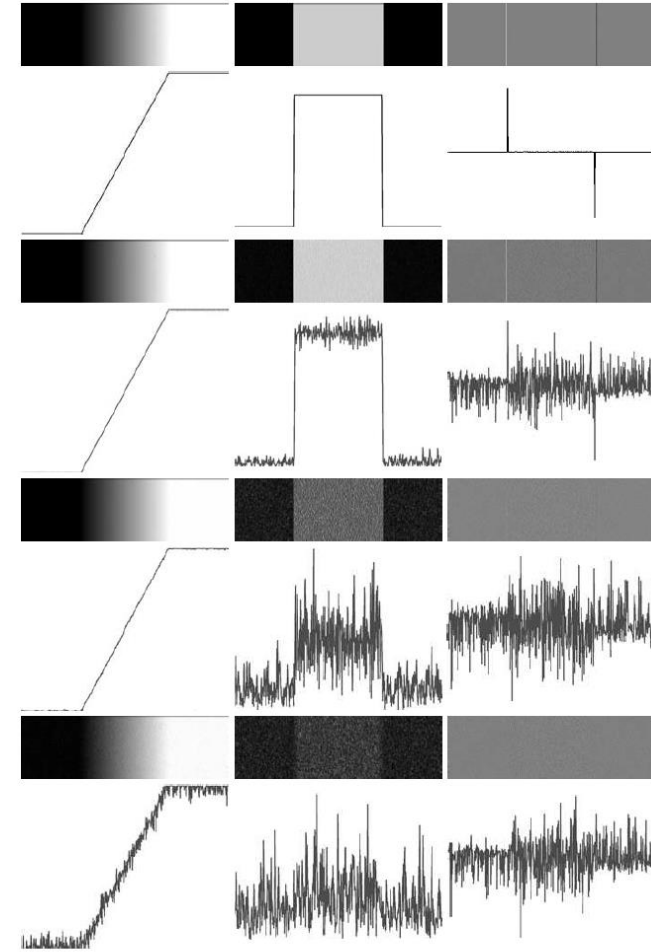


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

BASIC EDGE DETECTION

- Image gradient

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

- g_x - gradient image

- Direction of greatest change in intensity

- Edge is perpendicular to the gradient direction

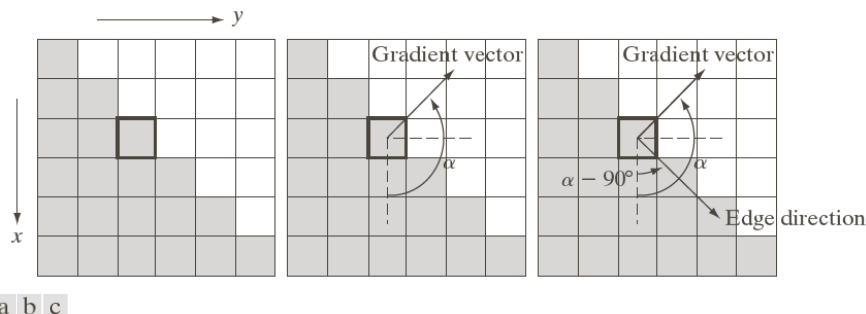


FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

- Magnitude

- $M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y|$

- Rate of change in the direction of gradient vector

- Approx. only valid in horizontal vertical directions

- Direction

- $\alpha(x, y) = \tan^{-1} \frac{g_y}{g_x}$

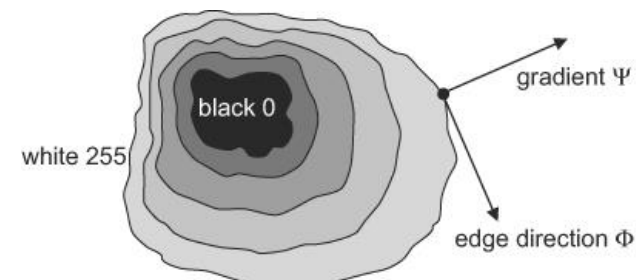


Figure 5.16: Gradient direction and edge direction. © Cengage Learning 2015.

GRADIENT OPERATORS

- Use digital approximations of partial derivatives → first difference

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

-1
1

-1	1
----	---

a b

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

- Can consider diagonal edges → Roberts kernel
- Usually want odd symmetric kernels for computational efficiency
 - Prewitt – centered first difference
 - Sobel – weighed centered first difference (noise suppression)

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

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 .

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

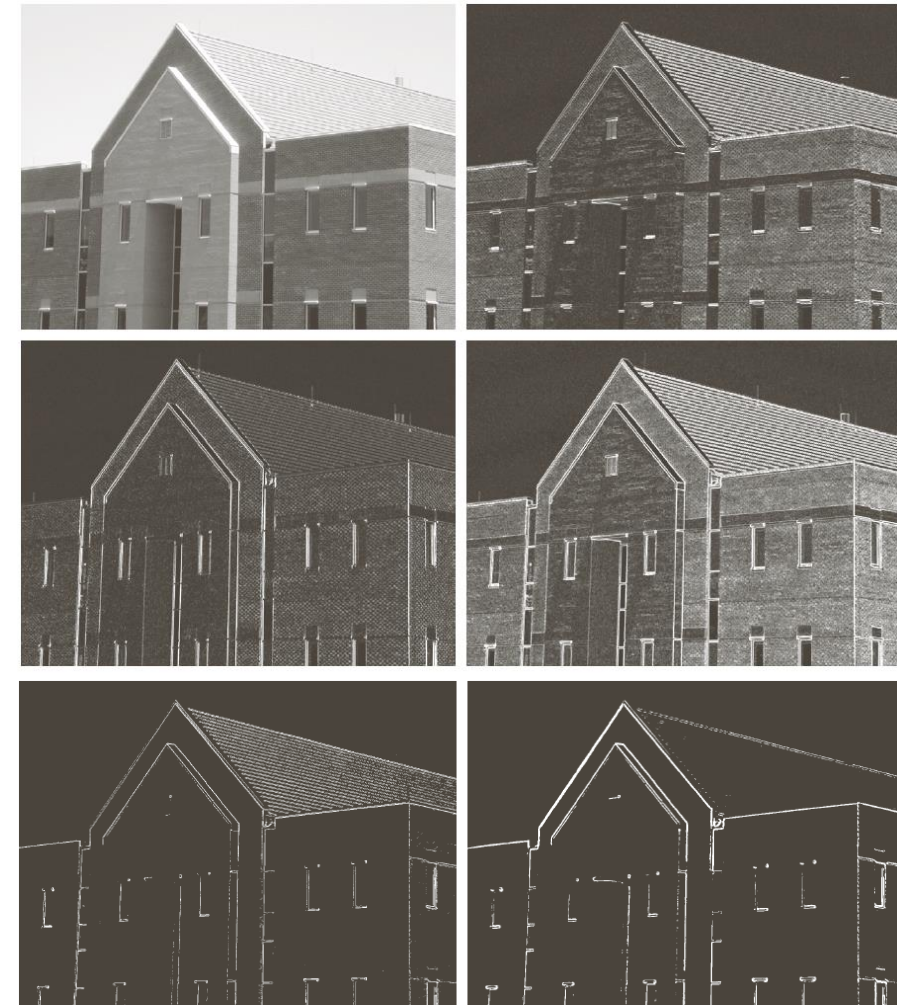
Sobel

EDGE EXAMPLES

- Gradient images show preference to edge direction
- Magnitude gives strength of edge
- Gradient thresholding used to highlight strong edges
 - Use smoothing for cleaner gradient images
 - See Fig. 10.18

a b

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.



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, $|g_x| + |g_y|$.

MORE ADVANCED EDGE DETECTION

- Simple edge detection
 - Filter image with smoothing mask and with gradient kernels
 - Does not account of edge characteristics or noise content
- Advanced detection
 - Seeks to leverage image noise properties and edge classification
 - Marr-Hildreth detector
 - Canny edge detector
 - Hough transform

MAAR-HILDRETH EDGE DETECTOR

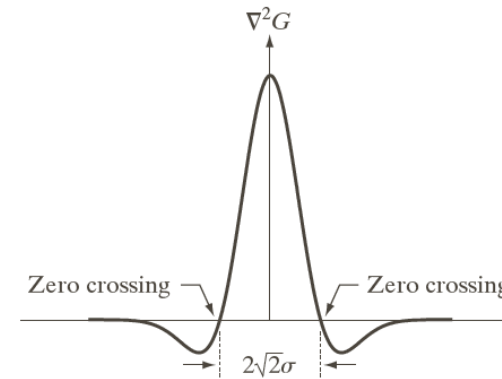
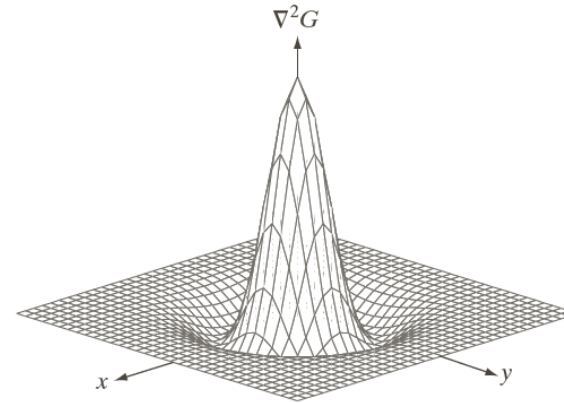
■ Insights

- Edges (image features) depend on scale
- Edge location is from zero-crossing

■ Laplacian of Gaussian (LoG) operator

- $\nabla^2 G(x, y) = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] e^{-\frac{x^2 + y^2}{2\sigma^2}}$
 - σ is the space constant – defines circle radius
- Gaussian blurs the image at scales much smaller than σ
- Second derivative Laplacian responds to edges in all directions

- Also called Mexican hat kernel



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

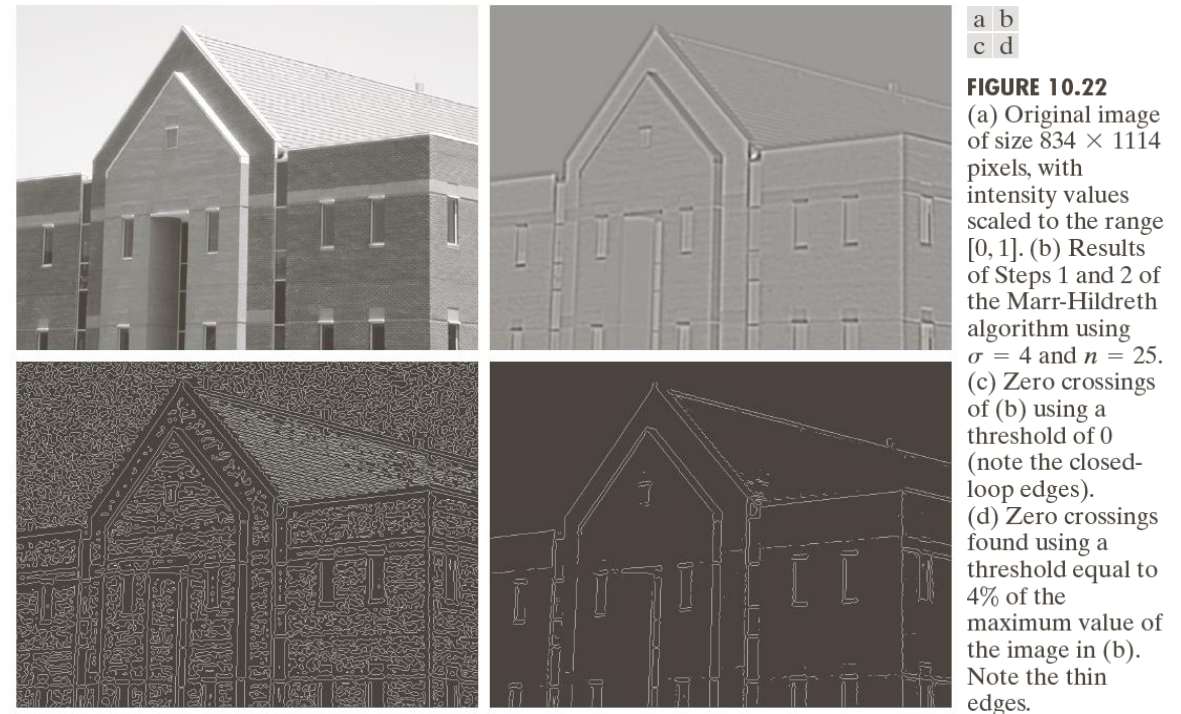
FIGURE 10.21

(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.

MAAR-HILDRETH ALGORITHM

- $g(x, y) = [\nabla^2 G(x, y)] * f(x, y)$
- By linearity
 - $g(x, y) = \nabla^2 [G(x, y) * f(x, y)]$
 - Smooth image first then apply Laplacian
- Follow with zero crossing detection
 - Search a 3×3 neighborhood for changes in sign in opposite pixels
 - May consider magnitude threshold to deal with noise
- Size of LoG filter ($n \times n$) should be greater than or equal to 6σ

- Simplification possible using the difference of Gaussians (DoG)
 - Similar to human visual process

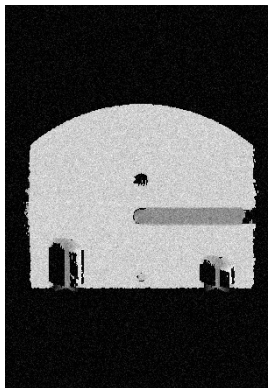


CANNY EDGE DETECTOR

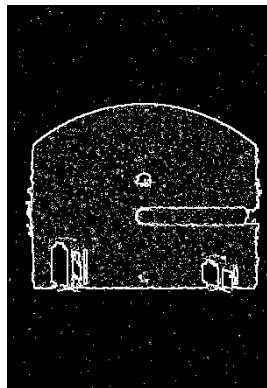
- Three objectives
 - Low error rate: find all edges with minimal false detections
 - Edge points localized: should find center of true edge
 - Single edge response: only single pixel for thick edges
- Key operations
 - Non-maxima suppression of groups of large magnitude 1st derivative response
 - Hysteresis threshold for long connected edges
- Canny algorithm Overview
 1. Smooth image with Gaussian filter
 2. Compute gradient magnitude and angle
 3. Apply nonmaxima suppression of gradient magnitude
 4. Use hysteresis thresholding and connectivity analysis to detect and link edges

CANNY EDGE DETECTION I

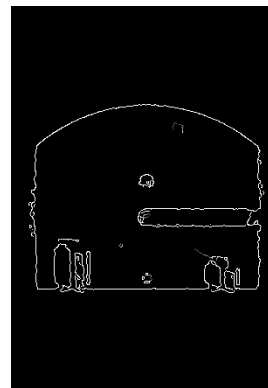
- Popular edge detection algorithm that produces a thin lines
- 1) Smooth with Gaussian kernel
- 2) Compute gradient
 - Determine magnitude and orientation (45 degree 8-connected neighborhood)



object



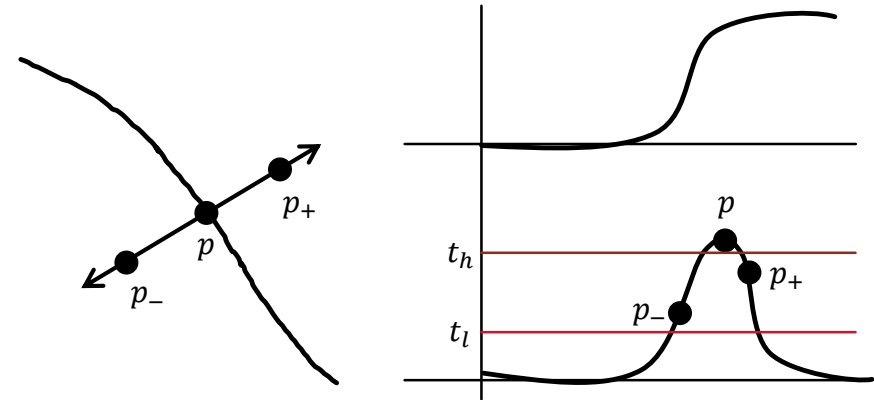
Sobel



Canny

<http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm>

- 3) Use non-maximal suppression to get thin edges
 - Compare edge value to neighbor edgels in gradient direction



- 4) Use hysteresis thresholding to prevent streaking
 - High threshold to detect edge pixel, low threshold to trace the edge

CANNY EDGE DETECTION II

- Optimal edge detection algorithm
 - Returns long thin (1 pixel wide) connected edges
- Non-maximal edge suppression technique to return a single pixel for an edge
 - Examine pixels along gradient direction
 - Only retain pixel if it is larger than neighbors
- Hysteresis threshold to remove spurious responses and maintain long connected edges
 - High threshold used to find definite edges
 - Low threshold to track edges

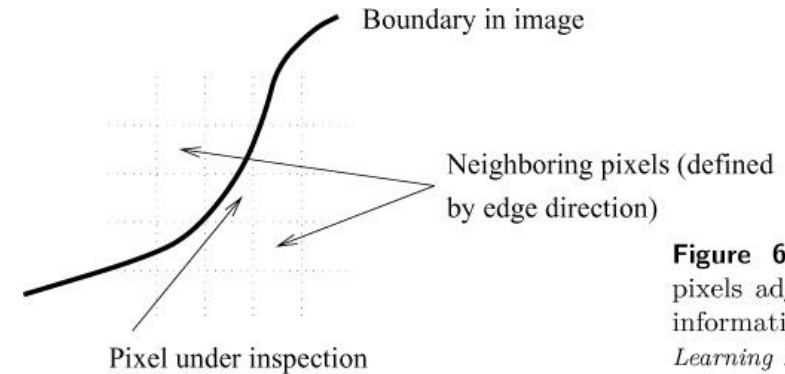
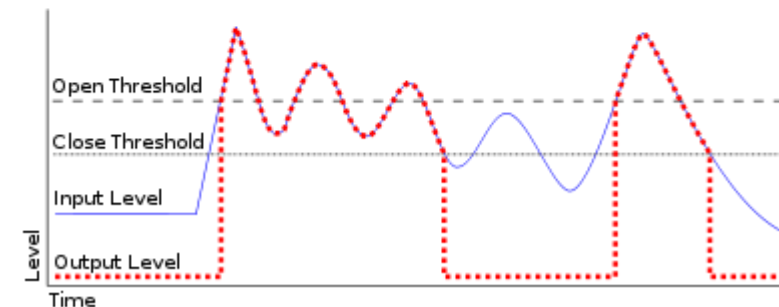
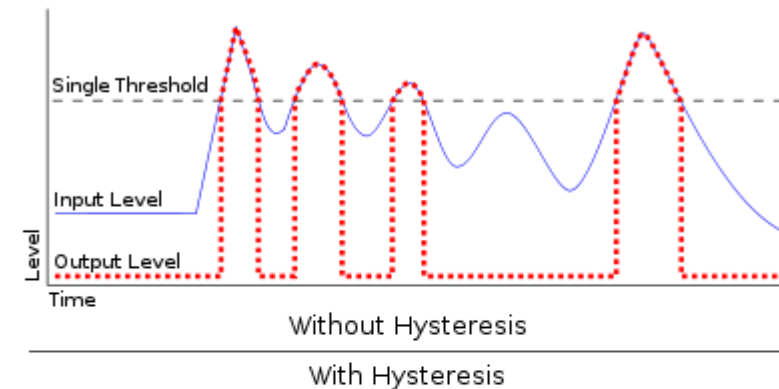


Figure 6.10:
pixels adjacent
information a
Learning 2015.



NONMAXIMA SUPPRESSION

- Gradient produces thick edges (for steps/ramps)
 - Consider 4 orientations in 3x3 neighborhood
 - Horizontal, vertical, and diagonals
1. Quantize gradient angle into 8 directions
 - d_k mapped from $\alpha(x, y)$
 2. Suppress edge pixel if any of it's gradient neighbors has greater magnitude
 - $g_N(p) = 0$ if $M(p) < M(d_k^+)$ or $M(d_k^-)$
 - $g_N(p) = M(p)$ otherwise

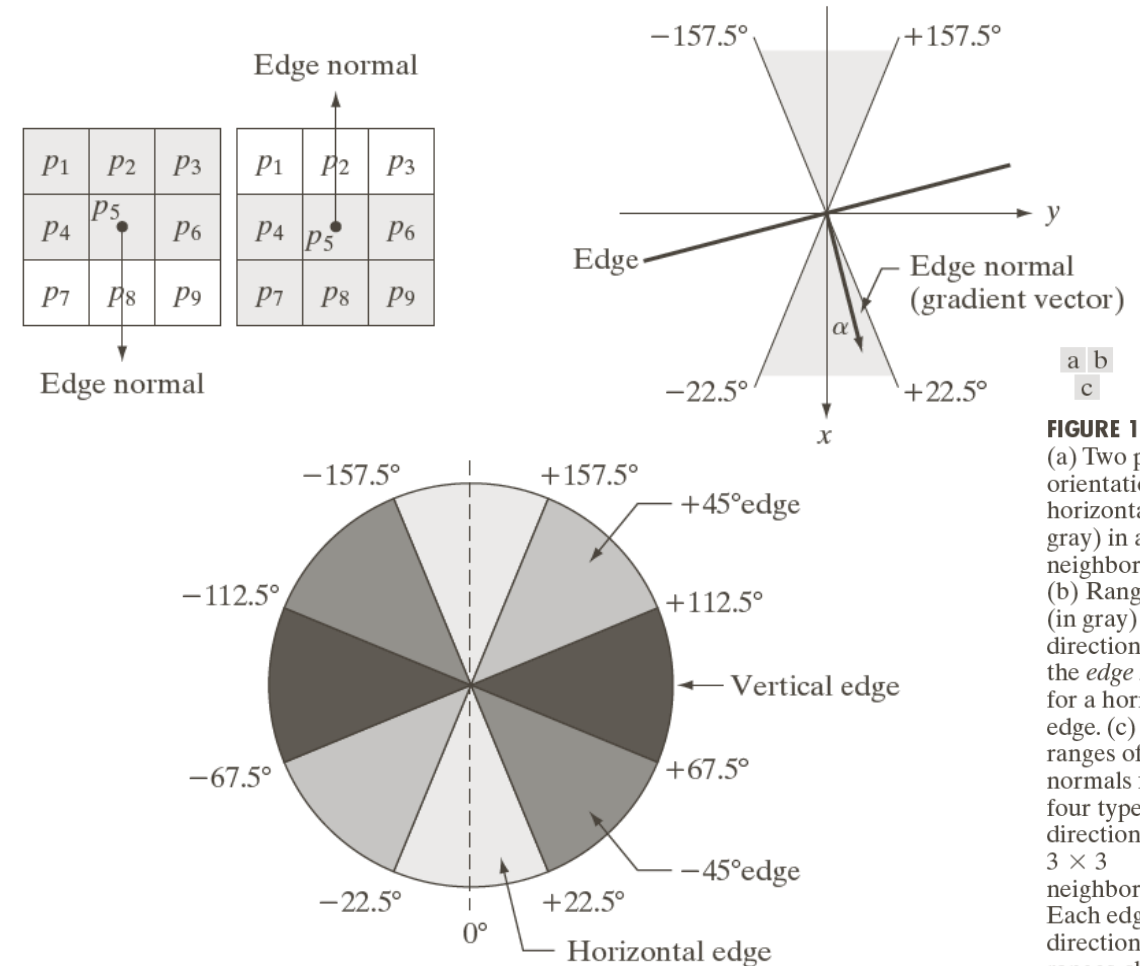


FIGURE 10.24 (a) Two possible orientations of a horizontal edge (in gray) in a 3×3 neighborhood. (b) Range of values (in gray) of α , the direction angle of the edge normal, for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3×3 neighborhood. Each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

CANNY EDGE EXAMPLES

Simple Edge Detection

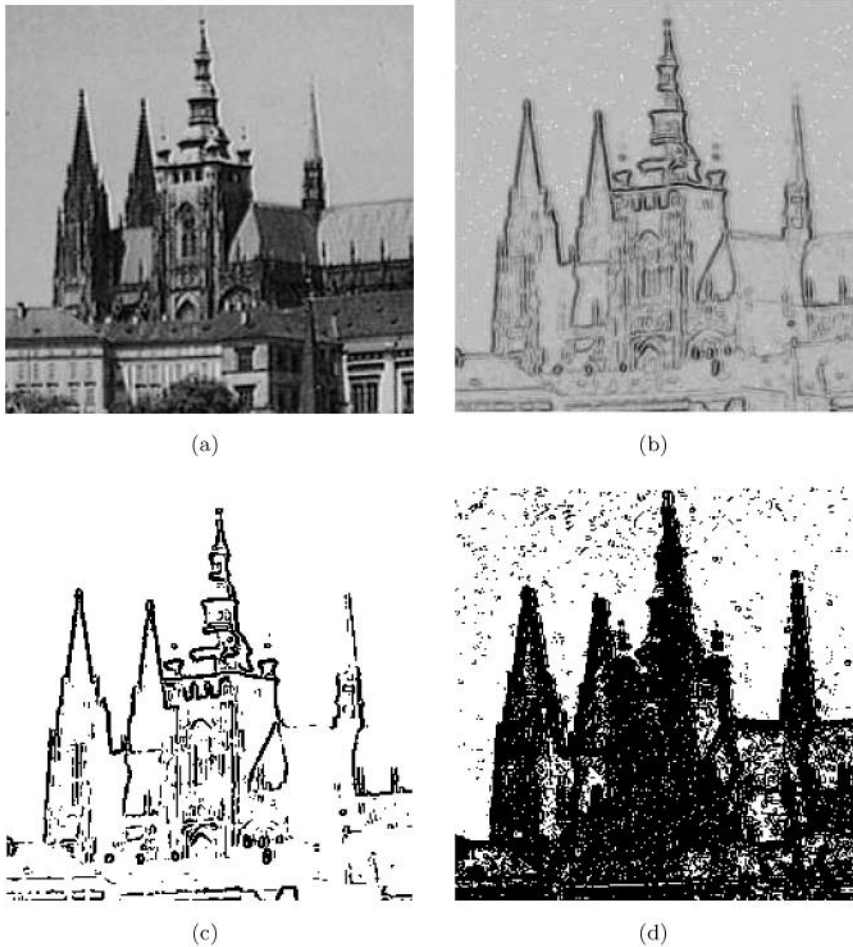


Figure 6.9: Edge image thresholding. (a) Original image. (b) Edge image (low contrast edges enhanced for display). (c) Edge image thresholded at 30. (d) Edge image thresholded at 10. © Cengage Learning 2015.

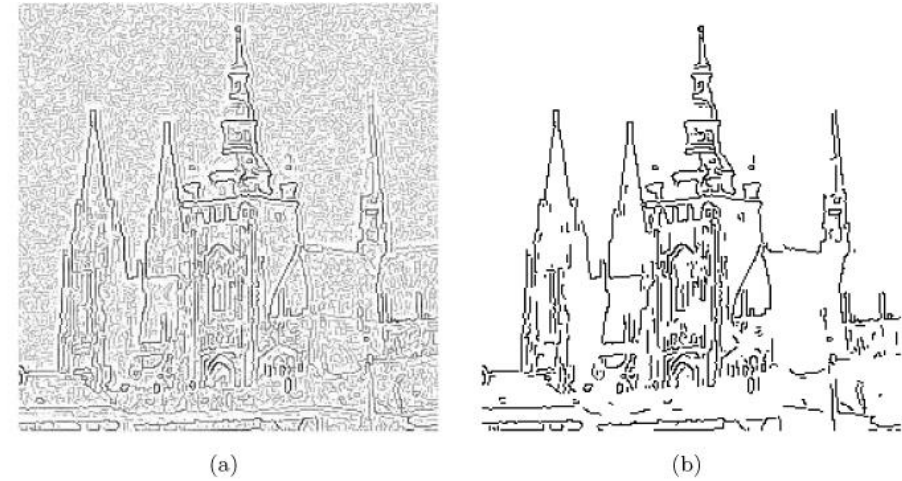


Figure 6.11: (a) Non-maximal suppression of the data in Figure 6.9b. (b) Hysteresis applied to (a); high threshold 70, low threshold 10. © Cengage Learning 2015.

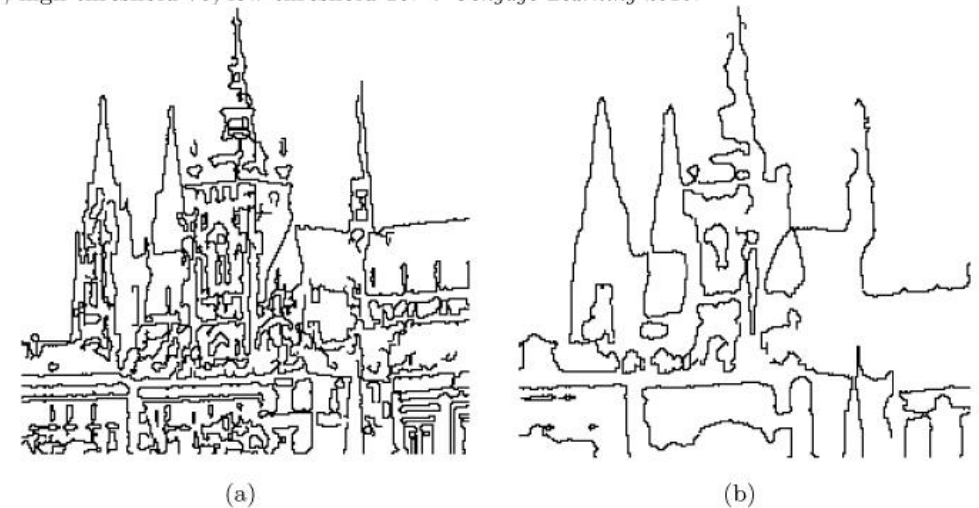


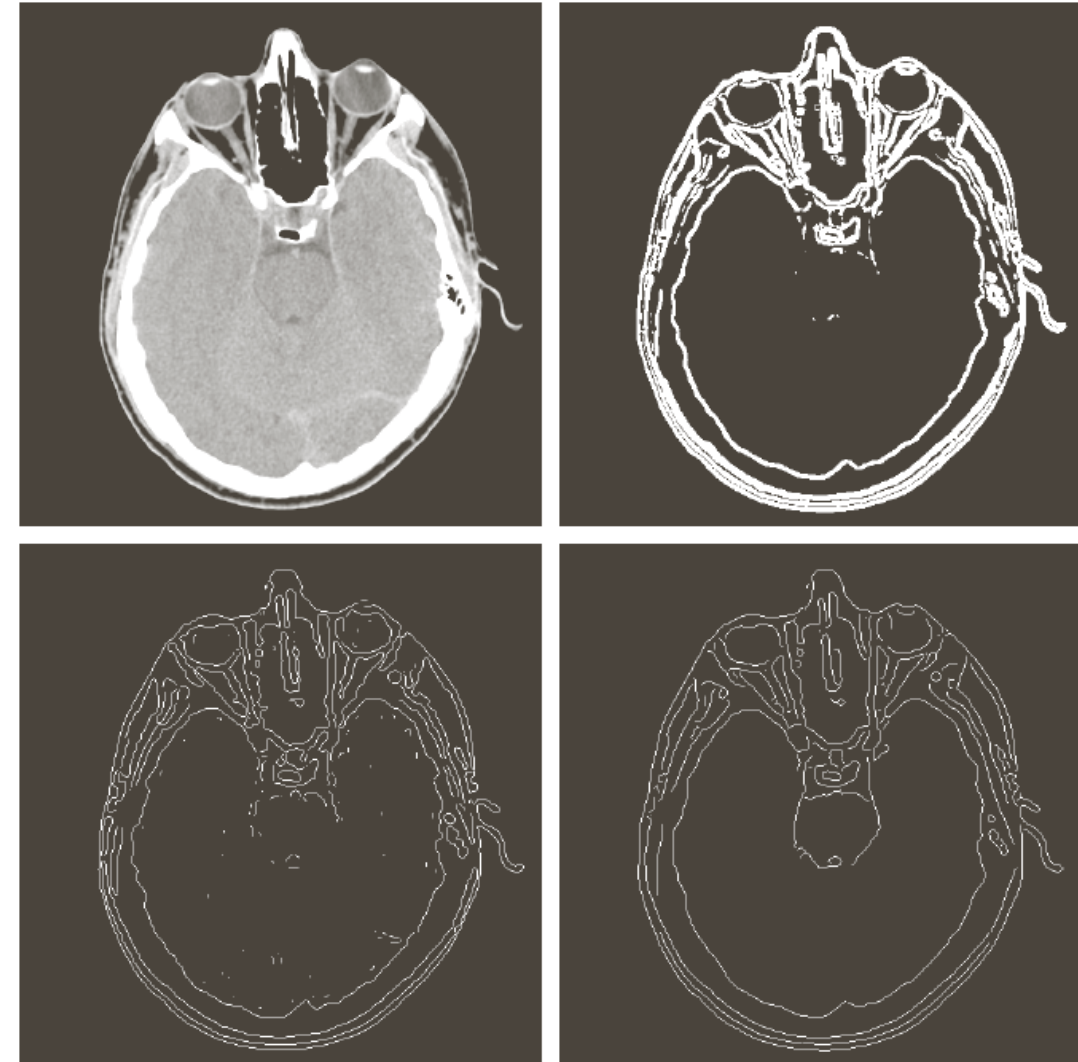
Figure 5.23: Canny edge detection at two different scales. © Cengage Learning 2015.

CANNY EDGE EXAMPLES II



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.)

HOUGH TRANSFORM

- Segmentation viewed as the problem of finding objects
 - Must be of known size and shape
- Typically hard to do because of shape distortions
 - Rotation, zoom, occlusion
- Search for parameterized curves in image plane
 - $c(x, a) = 0$
 - a – n-dimensional vector of curve parameters
 - Each edge pixel “votes” for different parameters and need to find parameter set with most votes

HOUGH TRANSFORM FOR LINES

- Lines are the original motivation for Hough transform
- Lines in the real-world can be broken, collinear, or occluded
 - Combine these collinear line segments into a larger extended line
- Hough transform creates a parameter space for the line
 - Every pixel votes for a family of lines passing through it
 - Potential lines are those bins (accumulator cells) with high count
- Uses global rather than local information
- See `hough.m`, `radon.m` in Matlab

HOUGH TRANSFORM INSIGHT

- Want to search for all points that lie on a line
 - This is a large search (take two points and count the number of edgels)
- Infinite lines pass through a single point (x_i, y_i)
 - $y_i = ax_i + b$
 - Select any a, b
- Reparameterize
 - $b = -x_i a + y_i$
 - ab -space representation has single line defined by point (x_i, y_i)

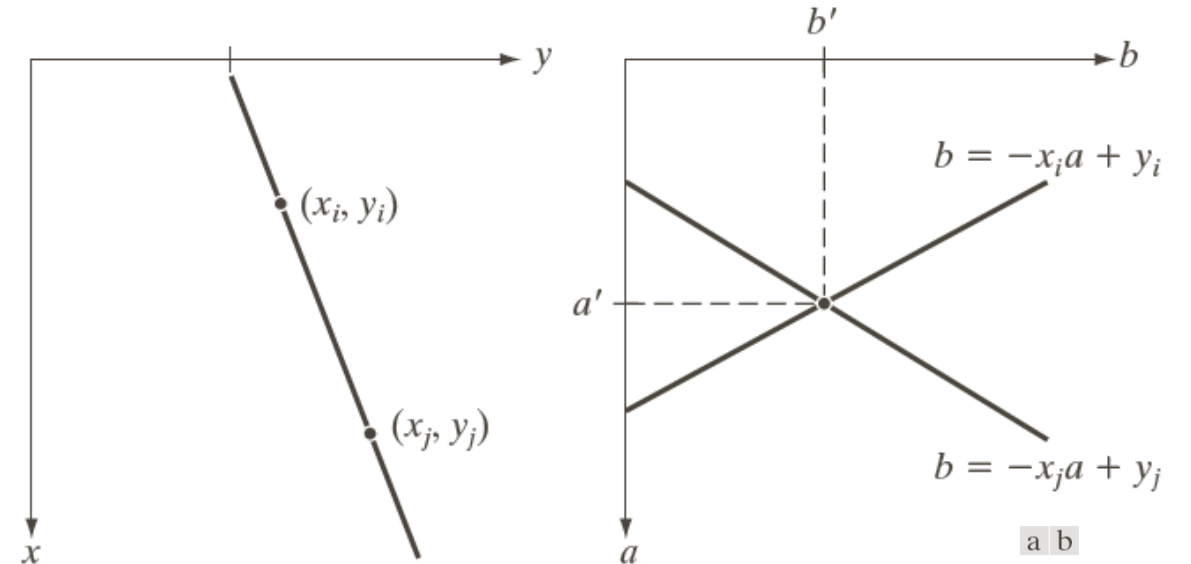


FIGURE 10.31
(a) xy -plane.
(b) Parameter space.

- All points on a line will intersect in parameter space
 - Divide parameter space into cells/bins and accumulate votes across all a and b values for a particular point
 - Cells with high count are indicative of many points voting for the same line parameters (a, b)

HOUGH TRANSFORM IN PRACTICE

- Use a polar parameterization of a line – why?

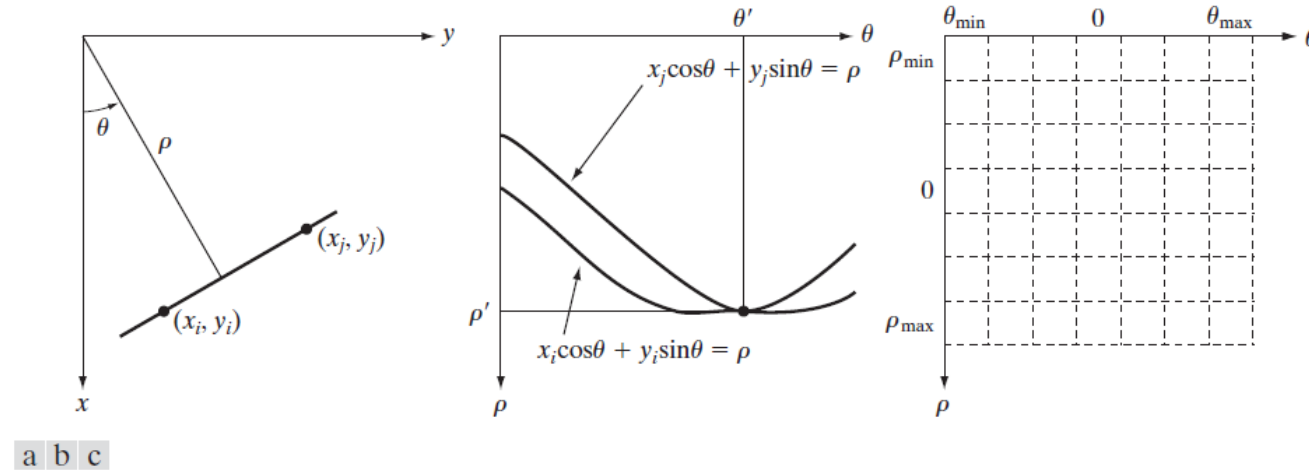


FIGURE 10.32 (a) (ρ, θ) parameterization of line in the xy -plane. (b) Sinusoidal curves in the $\rho\theta$ -plane; the point of intersection (ρ', θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy -plane. (c) Division of the $\rho\theta$ -plane into accumulator cells.

- After finding bins of high count, need to verify edge
 - Find the extent of the edge (edges do not go across the whole image)
- This technique can be extended to other shapes like circles

HOUGH TRANSFORM EXAMPLE I



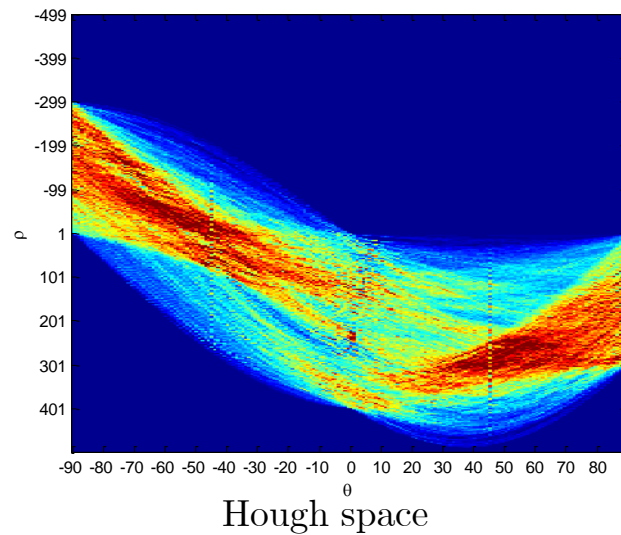
Input image



Grayscale



Canny edge image

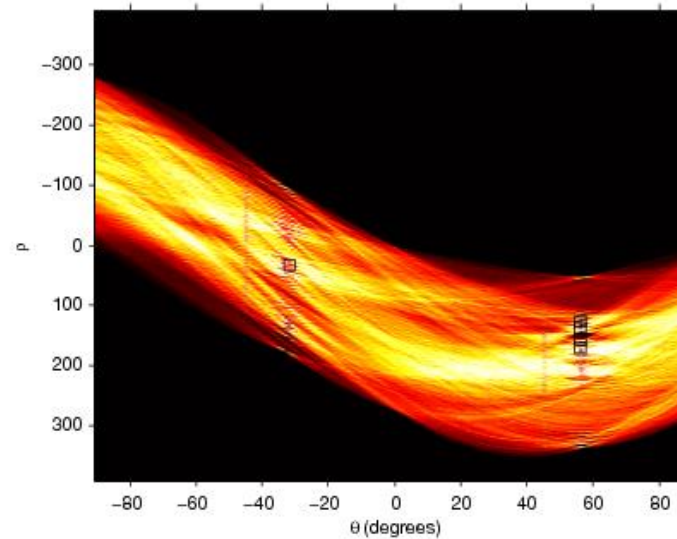
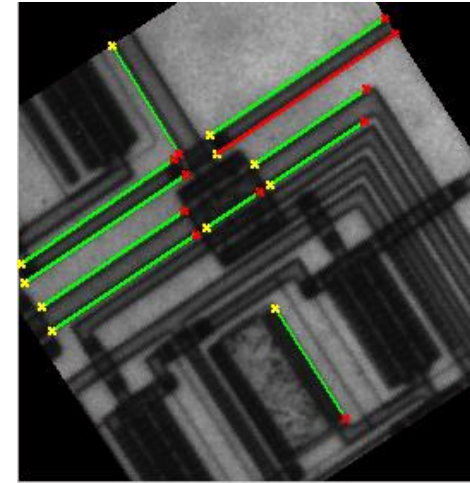
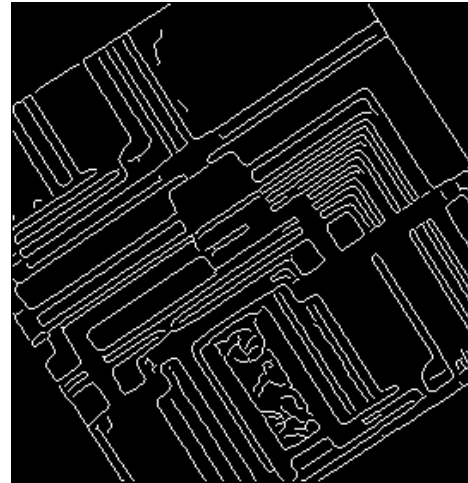
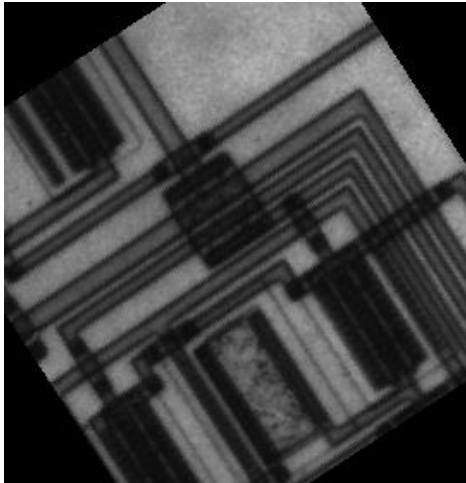


Hough space



Top edges

HOUGH TRANSFORM EXAMPLE II



HOUGH TRANSFORM FOR CIRCLES

- Consider equation of circle
 - $(x_1 - a)^2 + (x_2 - b)^2 = r^2$
 - (a, b) – center of circle; r – radius

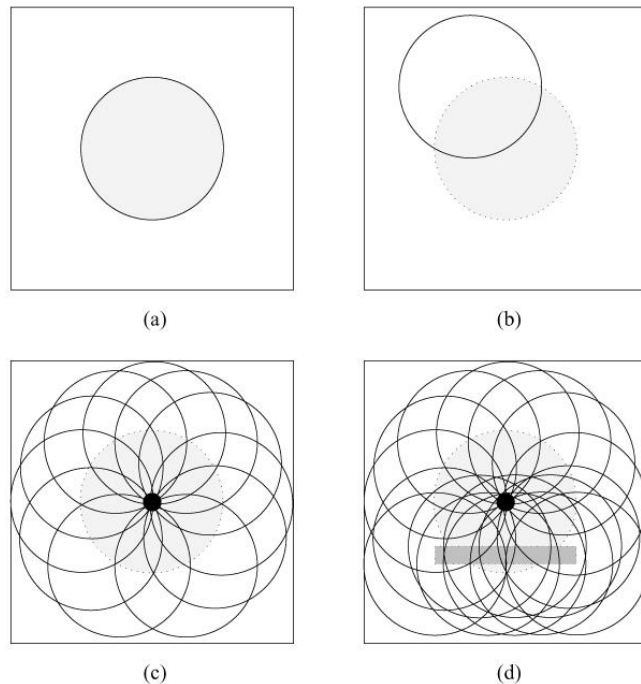


Figure 6.31: Hough transform—example of circle detection. (a) Image of a dark circle, of known radius r , on a bright background. (b) For each dark pixel, a potential circle-center locus is defined by a circle with radius r and center at that pixel. (c) The frequency with which image pixels occur on circle-center loci is determined—the highest-frequency pixel represents the center of the circle (marked by •). (d) The Hough transform correctly detects the circle (marked by •) in the presence of incomplete circle information and overlapping structures. (See Figure 6.32 for a real-life example.) © Cengage Learning 2015.

- Each edgel votes for a circle of radius r at center (a, b)
- Accumulator array is now 3-dimensional
 - Usually for fixed radius circle

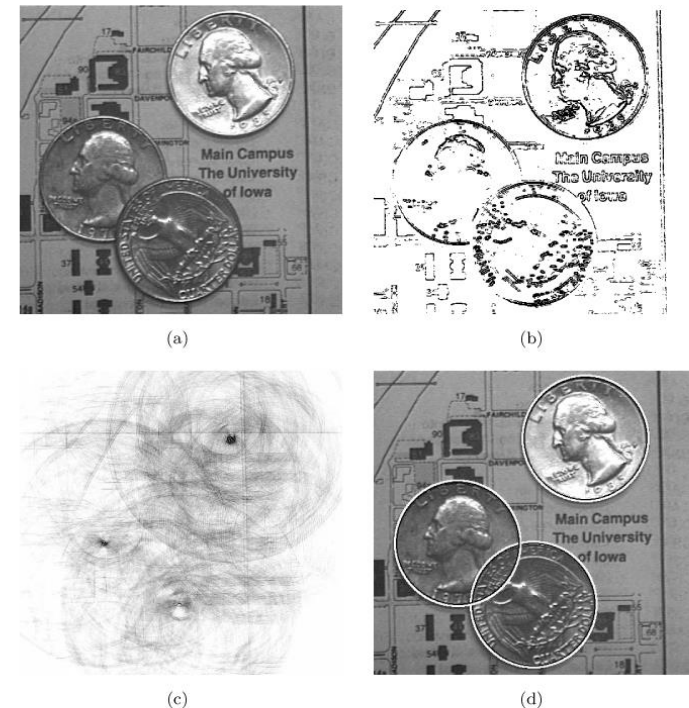


Figure 6.32: Hough transform—circle detection. (a) Original image. (b) Edge image (note that the edge information is far from perfect). (c) Parameter space. (d) Detected circles. © Cengage Learning 2015.

HOUGH TRANSFORM CONSIDERATIONS

- Practical only for up to 3-dimensions
 - Exponential growth of accumulator array
- Use gradient information to simplify process
 - Only accumulate limited number of bins
 - Accounts for local consistency constraints
 - Line pixels should be in edge direction (orthogonal to gradient direction)
- Weight accumulator by edge magnitude
 - Consider only the strongest edges
- “Back project” strongest accumulator cells of each pixel to remove other votes
 - Sharpen accumulator response
- Line tracing
 - Find endpoints of line

MULTISPECTRAL EDGES

- Pixel (i, j) has n -dimensional vector representation
- Trivial edge detection
 - Operate on each spectral band separately
 - Combine all bands to form single edge image
- Multiband (Roberts-like) edge operator
 - $2 \times 2 \times n$ - neighborhood

$$\frac{\sum_{r=1}^n [d(i, j)] [d(i+1, j+1)]}{\sqrt{\sum_{r=1}^n [d(i, j)]^2 \sum_{r=1}^n [d(i+1, j+1)]^2}} \frac{\sum_{r=1}^n [d(i+1, j)] [d(i, j+1)]}{\sqrt{\sum_{r=1}^n [d(i+1, j)]^2 \sum_{r=1}^n [d(i, j+1)]^2}},$$

where $d(k, l) = f_r(k, l) - \bar{f}(k, l)$.

(5.60)

THRESHOLDING

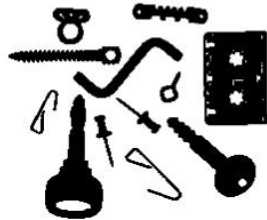
- Segment object from background

- $$g(i,j) = \begin{cases} 1 & f(i,j) > T \\ 0 & f(i,j) \leq T \end{cases}$$

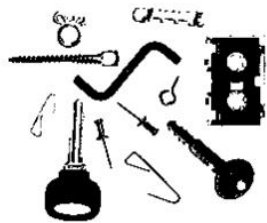
- T – threshold
- 1 object and 0 background



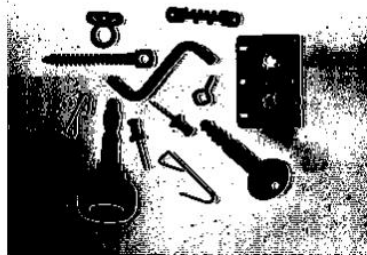
(a)



(b)



(c)



(d)

Figure 6.1: Image thresholding. (a) Original image. (b) Threshold segmentation. (c) Threshold too low. (d) Threshold too high. © Cengage Learning 2015.

- Requires the correct threshold of this to work

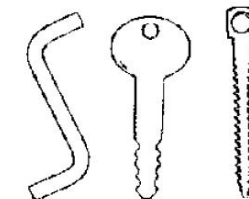
- Difficulty to use a single global threshold
 - $T = \mathcal{T}(f)$
- More often want adaptive threshold
 - $T = \mathcal{T}(f, f_c)$
 - f_c - is smaller image region (e.g. subimage)

- Many simple variants

- Band thresholding - range of values for object
- Multiband – multiple bands to give grayscale result



(a)



(b)

Figure 6.2: Image thresholding modification. (a) Original image. (b) Border detection using band-thresholding. © Cengage Learning 2015.

THRESHOLD DETECTION METHODS

- When objects are similar, the resulting histogram is bimodal
 - Objects one color and background another
 - Good threshold is between “peaks” in less probable intensity regions
 - Intuitively the lowest point between peaks

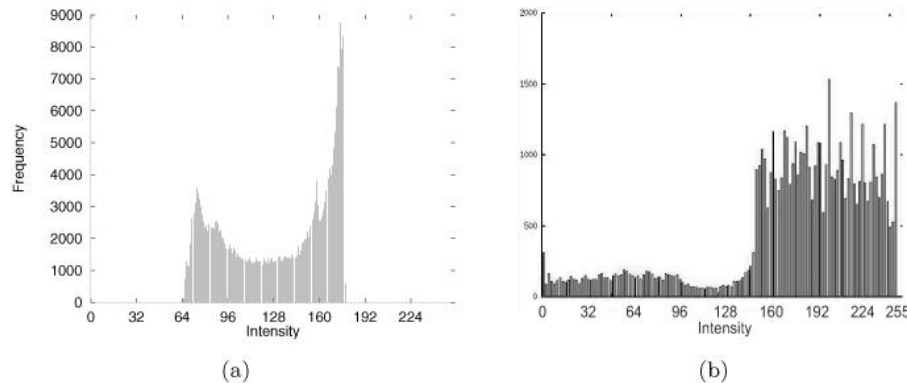


Figure 6.3: Bimodal histograms. (a) In cases with well-separable objects from the background, the shown histogram is clearly bimodal. (b) An example of a more shallow bimodal histogram (see top-left of Figure 6.5 for original image, in which the distinction between foreground and background has been deliberately perturbed). Note a wide, shallow peak whose distribution reaches from 0 to approximately 140, and a higher one more easily visible to the right. The distributions overlap in the gray-levels 100–160. © Cengage Learning 2015.

- In practice is difficult to tell if a distribution is bimodal
- There can be many local maxima
 - How should the correct one be selected?
- Notice also that since the histogram is global, a histogram for salt and pepper noise could be the same as for objects on background
- Should consider some local neighborhood when building the histogram
 - Account for edges

OPTIMAL THRESHOLDING

- Model the histogram as a weighted sum of normal probability densities
- Threshold selected to minimize segmentation error (minimum number of mislabeled pixels)
 - Gray level closest to minimum probability between normal maxima
- Difficulties
 - Normal distribution assumption does not always hold
 - Hard to estimate normal parameters

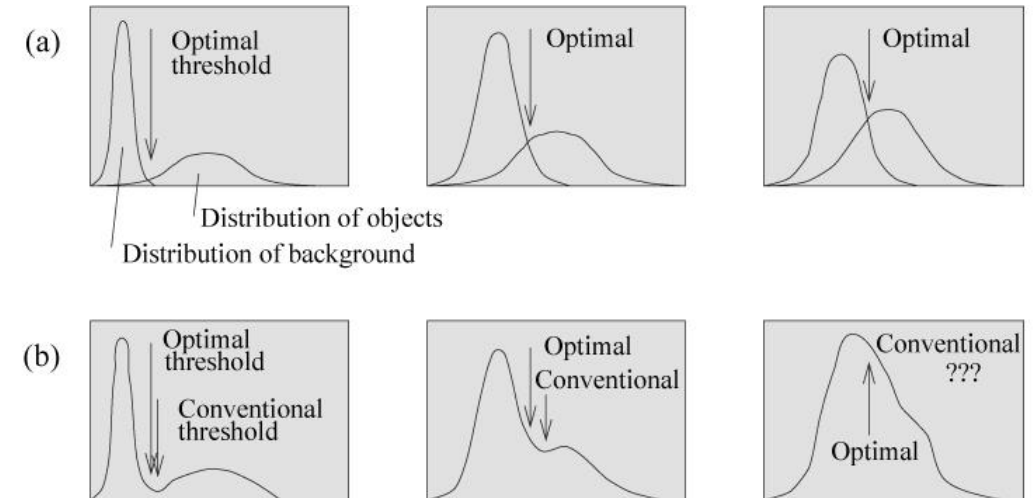


Figure 6.4: Gray-level histograms approximated by two normal distributions—the threshold is set to give minimum probability of segmentation error. (a) Probability distributions of background and objects. (b) Corresponding histograms and optimal threshold. © Cengage Learning 2015.

- Useful tools:
 - Maximum-likelihood classification
 - Expectation maximization
 - Gaussian mixture modeling

OTSU'S ALGORITHM

- Automatic threshold detection
 - Test all possible thresholds and find that which minimizes foreground/background variance
 - “Tightest” distributions
- 1. Compute histogram H of image and normalize to make a probability
- 2. Apply thresholding at each gray-level t
 - Separate histogram into background B and foreground F
- 3. Compute the variance σ_B and σ_F
- 4. Compute probability of pixel being foreground or background
 - $w_B = \sum_{j=0}^t H(j)$
- 5. Select optimal threshold as
 - $\hat{t} = \min_t \sigma(t)$
 - $\sigma(t) = w_B \sigma_B(t) + w_F(t) \sigma_F(t)$



Figure 6.5: Top left, an image with artificially stretched white background—the image has also been showered with random noise. Top right, thresholded with Otsu's method: the histogram is shown in Figure 6.3, and the algorithm delivers $t = 130$. At bottom, the results of $t = 115, 130, 145$ on the trickiest part of the image; segmentation quality degrades very quickly. © Cengage Learning 2015.

MIXTURE MODELING

- Assume Gaussian distribution for each group/object
 - Defined by mean intensity and standard deviation
 - $h_{model}(g) = \sum_{i=1}^n a_i \exp\{-(g - \mu_i)^2 / 2\sigma_i^2\}$
- Determine parameters by minimizing mismatch between model and actual histogram with fit function
 - Match Gaussians to histogram
 - $F = \sum_{g \in G} (h_{model}(g) - h_{region}(g))^2$
- Can use Otsu's as a starting guess
 - Limit search space

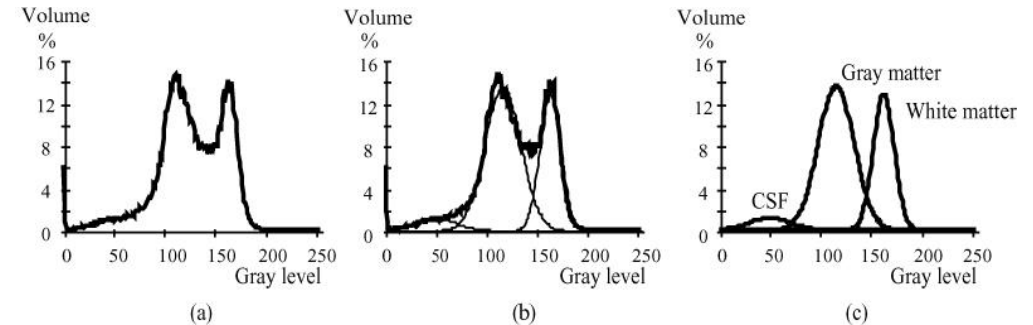


Figure 6.6: Segmentation of 3D T1-weighted MR brain image data using optimal thresholding. (a) Local gray-level histogram. (b) Fitted Gaussian distributions, global 3D image fit. (c) Gaussian distributions corresponding to WM, GM, and CSF. *Courtesy of R. J. Frank, T. J. Grabowski, The University of Iowa.*

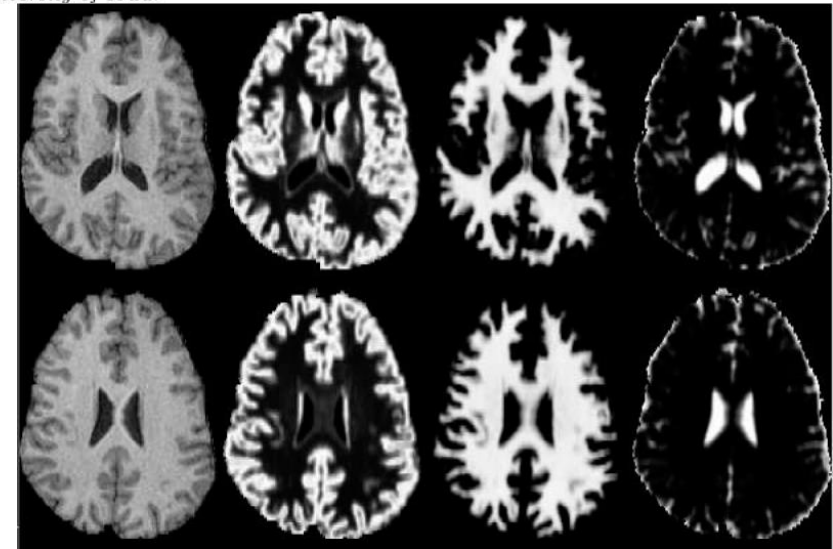


Figure 6.7: Optimal MR brain image segmentation. Left column: original T1-weighted MR images, two of 120 slices of the 3D volume. Middle left: Partial-volume maps of gray matter. The brighter the voxel, the higher is the partial volume percentage of gray matter in the voxel. Middle right: Partial-volume maps of white matter. Right column: Partial-volume maps of cerebro-spinal fluid. *Courtesy of R. J. Frank, T. J. Grabowski, The University of Iowa.*

MULTI-SPECTRAL THRESHOLDING

- Compute thresholds in spectral bands independently and combine in a single image
 - Used for remote sensing (e.g. satellite images), MRI, etc.
- Algorithm 6.3 (Sonka)
 1. Compute histogram and segment between local minima on either side of maximum peak for each band
 2. Combine segmentation regions into multispectral image
 3. Repeat on multispectral regions until each region is unimodal

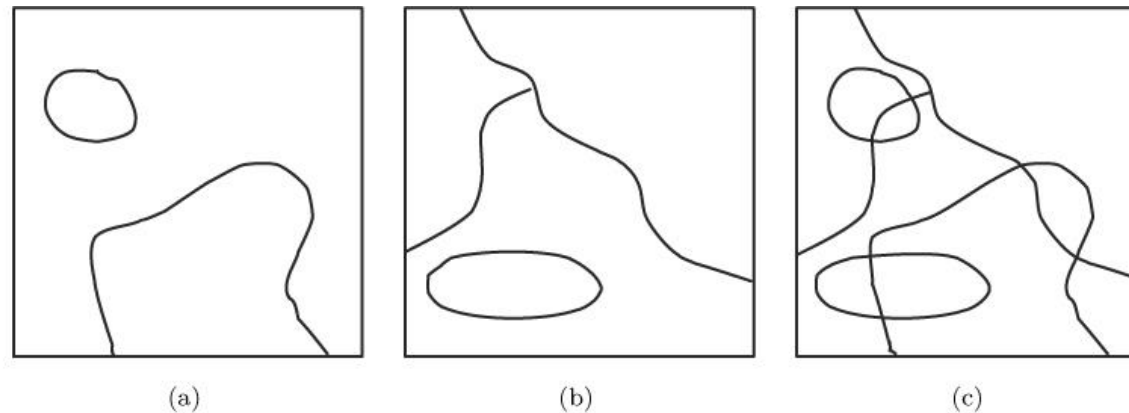


Figure 6.8: Recursive multi-spectral thresholding. (a) Band 1 thresholding. (b) Band 2 thresholding. (c) Multi-spectral segmentation. © Cengage Learning 2015.

REGION-BASED SEGMENTATION

- Regions are areas defined inside of borders
 - Simple to go back and forth between both
 - However, segmentation techniques differ
- Region growing techniques are typically better in noisy images
 - Borders are difficult to detect
- A region is defined by a homogeneity constraint
 - Gray-level, color, texture, shape, model, etc.
 - Each individual region is homogeneous
 - Any two regions together are not homogeneous

REGION MERGING

- Start with each pixel as a region and combine regions with a merge criterion
 - Defined over adjacent regions (neighborhood)
- Be aware the merging results can be different depending on the order of the merging
 - Prior merges change region relationships
- Simplest merge methods compute statistics over small regions (e.g. 2×2 pixels)
 - Gray-level histogram used for matching

REGION MERGING VIA BOUNDARY MELTING

- Utilize crack information (edges between pixels)
- Merge regions if there are weak crack edges between them

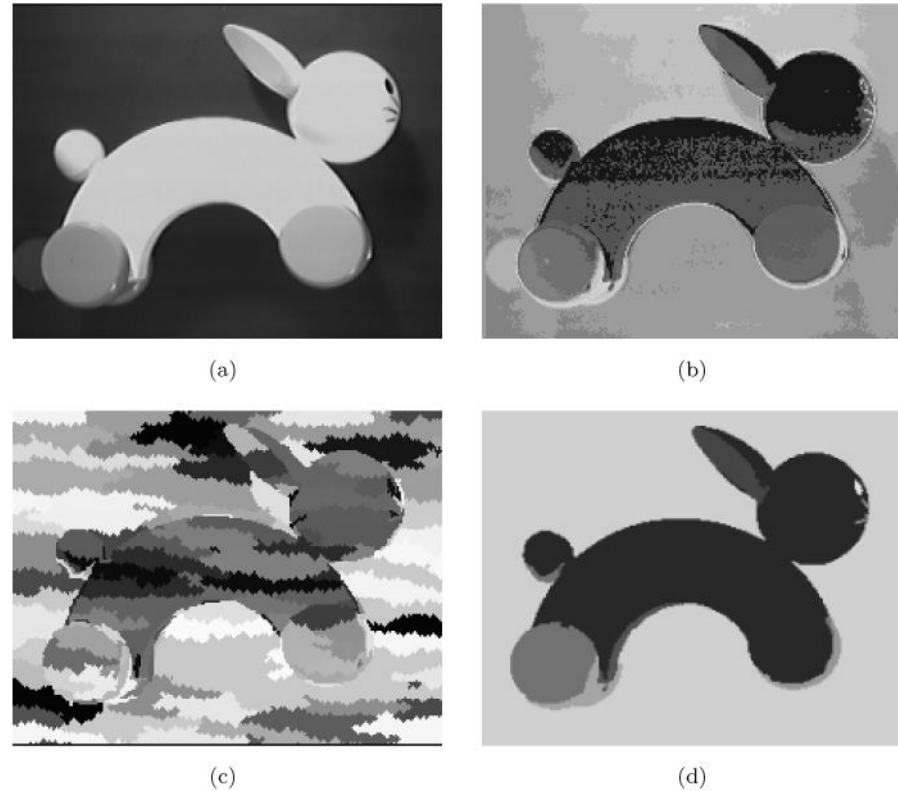


Figure 6.40: Region merging segmentation. (a) Original image. (b) Pseudo-color representation of the original image (in grayscale). (c) Recursive region merging. (d) Region merging via boundary melting. *Courtesy of R. Marik, Czech Technical University.*

REGION SPLITTING

- Opposite of region merging
 - Start with full image as single region and split to satisfy homogeneity criterion
- Merging and splitting do not result in the same regions
 - A homogenous split region may never have been grown from smaller regions
- Use same homogeneity criteria as in region merging

SPLIT AND MERGE

- Try to obtain advantages of both merging and splitting
- Operate on pyramid images
 - Regions are squares that correspond to pyramid level
 - Lowest level are pixels
- Regions in a pyramid level that are not homogeneous are split into four subregions
 - Represent higher resolution a level below
- 4 similar regions are merged into a single region at higher pyramid level
- Segmentation creates a quadtree
 - Each leaf node represents a homogenous region
 - E.g. an element in a pyramid level
 - Number of leaf nodes are number of regions

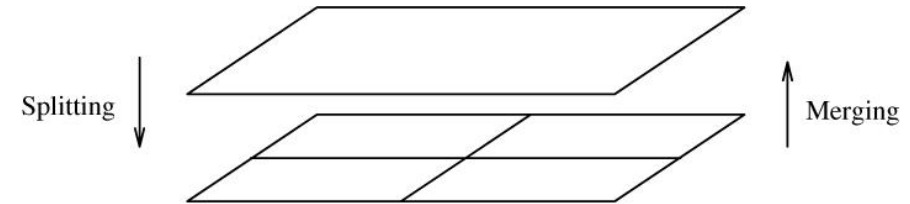


Figure 6.41: Split-and-merge in a hierarchical data structure. © Cengage Learning 2015.

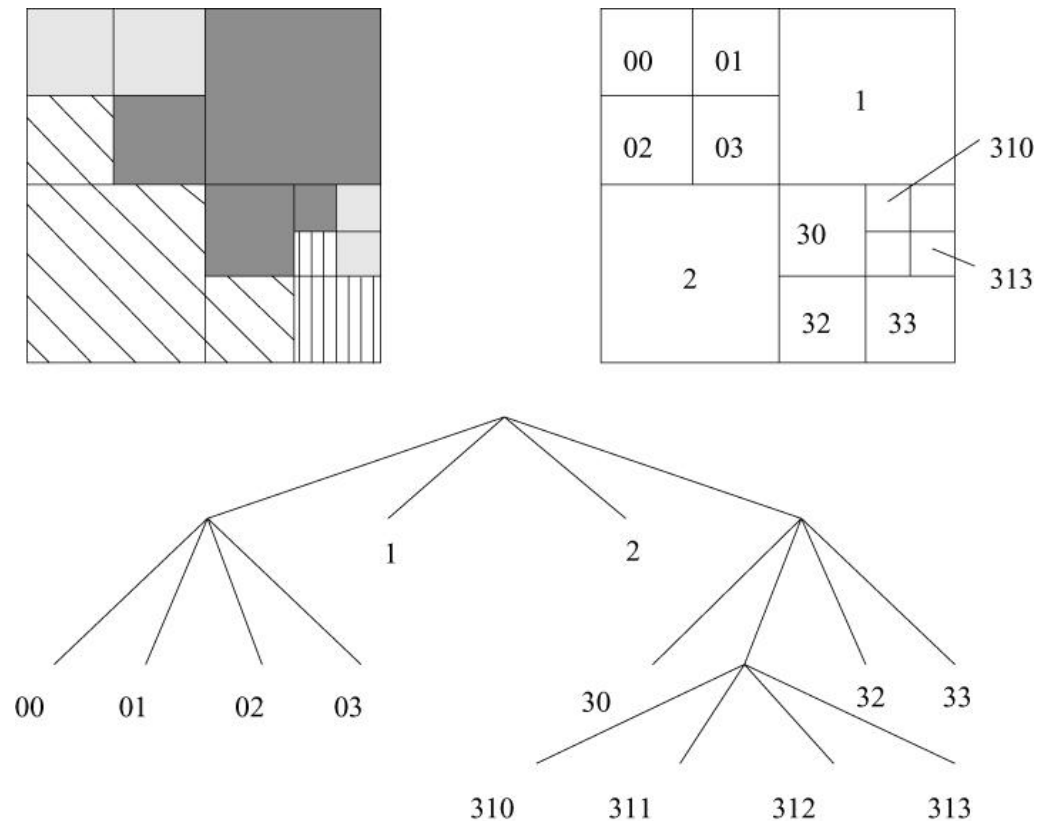


Figure 6.42: Segmentation quadtree. © Cengage Learning 2015.

WATERSHED SEGMENTATION

- Topography concepts
 - Watersheds are lines dividing catchment basins
- Region edges correspond to high watersheds
- Low gradient areas correspond to catchment basins
 - All pixels in a basin are simply connected and homogeneous because they share the same minimum

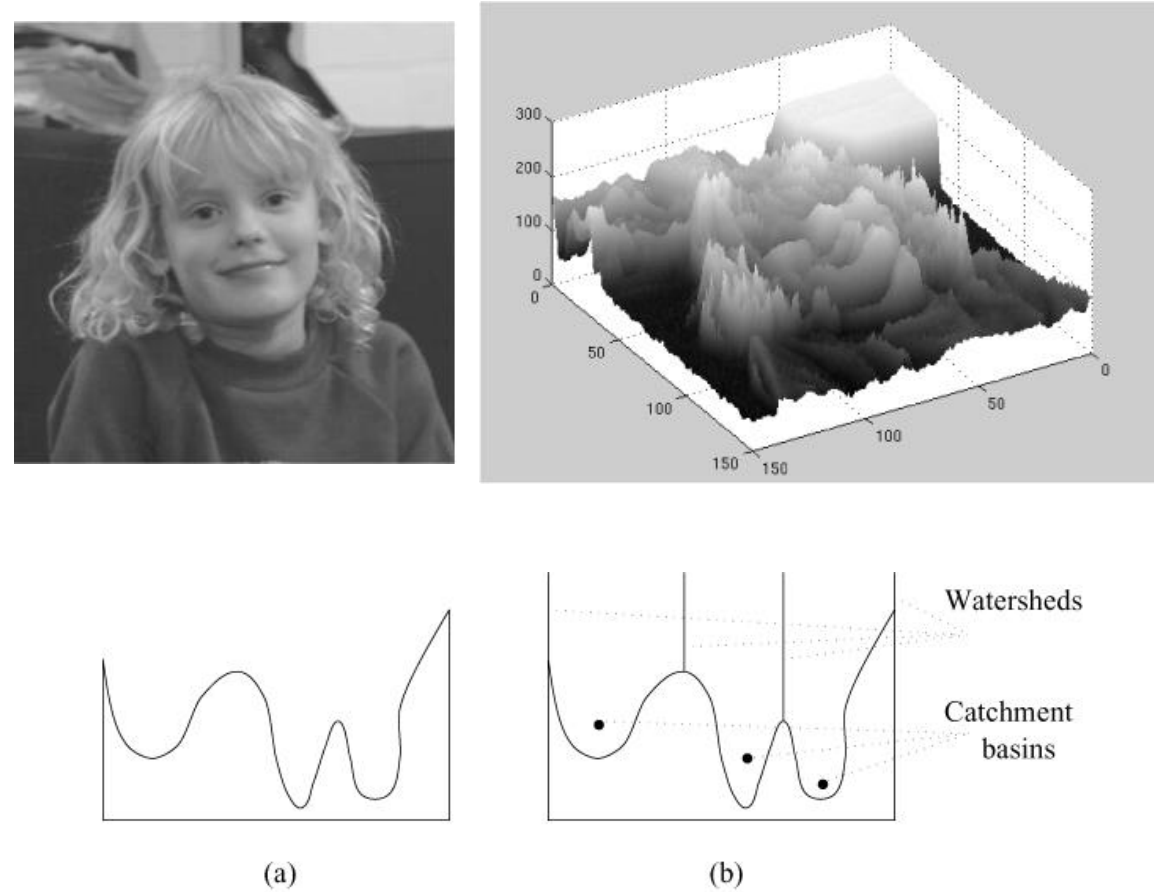


Figure 6.44: One-dimensional example of watershed segmentation. (a) Gray-level profile of image data. (b) Watershed segmentation—local minima of gray-level (altitude) yield catchment basins, local maxima define the watershed lines. © Cengage Learning 2015.

WATERSHED COMPUTATION

- Can build watersheds by examining gray-level values from lowest to highest
- Watersheds form when catchment basins merge
- Raw watershed results in oversegmentation
- Use of region markers can improve performance
 - [Matlab tutorial](#)

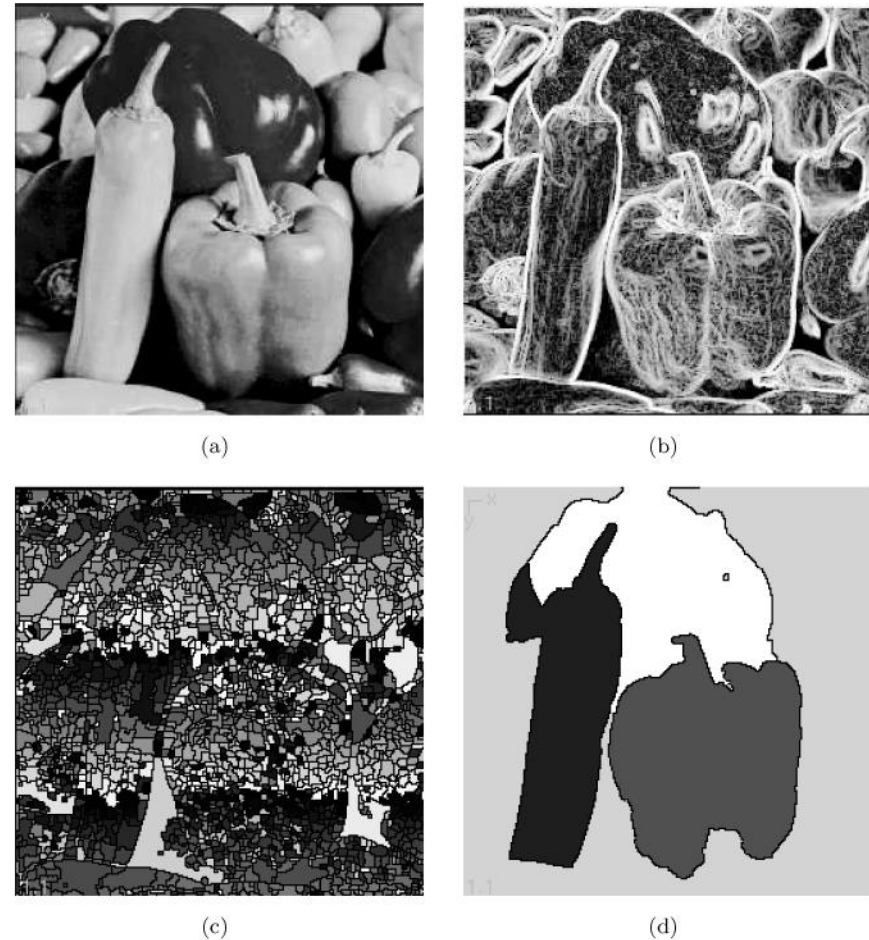


Figure 6.46: Watershed segmentation. (a) Original; (b) Gradient image, 3×3 Sobel edge detection, histogram equalized. (c) Raw watershed segmentation. (d) Watershed segmentation using region markers to control oversegmentation. *Courtesy of W. Higgins, Penn State University.*

MATCHING

- Basic approach to segmentation by locating known objects (search for patterns)
 - Generally have a model for object of interest
- Various examples of matching with different levels of sophistication
 - Optical character recognition (OCR)
 - Template matching when font is known and image carefully aligned
 - Font-independent OCR
 - Match pattern of character
 - Face recognition
 - Match pattern of face to image
 - More variability in appearance
 - Pedestrian behavior matching
 - Explain what a pedestrian is doing

TEMPLATE MATCHING

- Try to find template image in larger test image
- Minimize error between image and shifted template

$$E(\mathbf{x}) = \sum_{i=1}^{r_T} \sum_{j=1}^{c_T} (T_{i,j} - I_{x_a+i, x_b+j})^2 = 0, \quad (6.29)$$

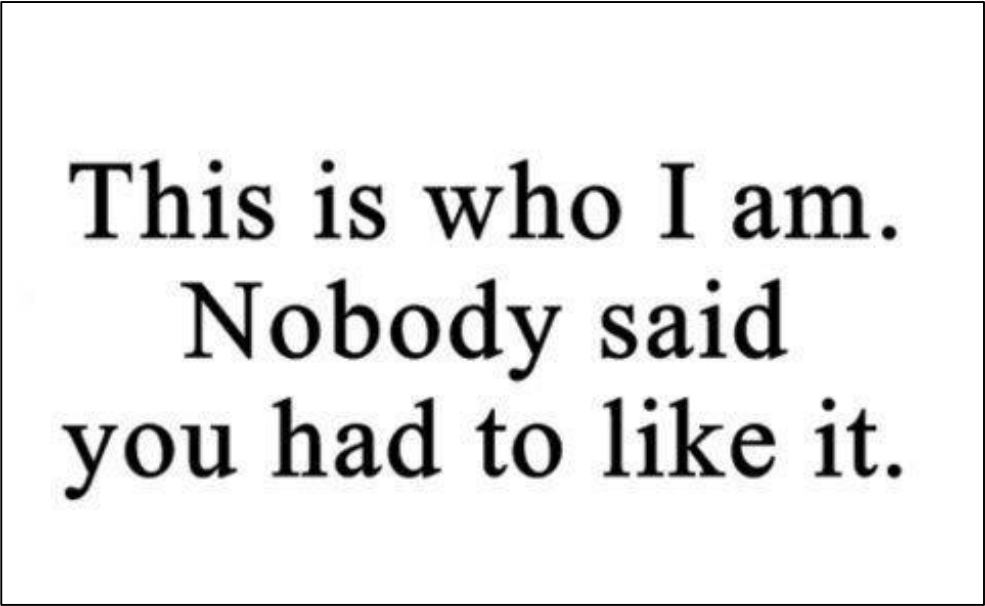
$$\begin{aligned} E(\mathbf{x}) &= \sum_{i=1}^{r_T} \sum_{j=1}^{c_T} (T_{i,j} - I_{x_a+i, x_b+j})^2 \\ &= \sum_{i=1}^{r_T} \sum_{j=1}^{c_T} (T_{i,j})^2 - 2 \sum_{i=1}^{r_T} \sum_{j=1}^{c_T} (T_{i,j} I_{x_a+i, x_b+j}) + \sum_{i=1}^{r_T} \sum_{j=1}^{c_T} (I_{x_a+i, x_b+j})^2, \end{aligned} \quad (6.30)$$

- First term is a constant and the last term changes slowly so only the middle term needs to be maximized

$$Corr_T(\mathbf{x}) = \sum_{i=1}^{r_T} \sum_{j=1}^{c_T} (T_{i,j} I_{x_a+i, x_b+j}). \quad (6.31)$$

BINARY FILTERING AS DETECTION

- Filtering (correlation) can be used as a simple object detector
 - Mask provides a search template
 - “Matched filter” – kernels look like the effects they are intended to find



This is who I am.
Nobody said
you had to like it.

image



y

template

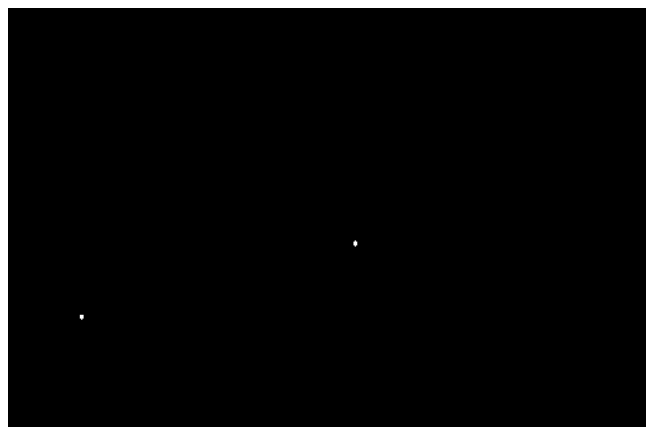
CORRELATION MASKING



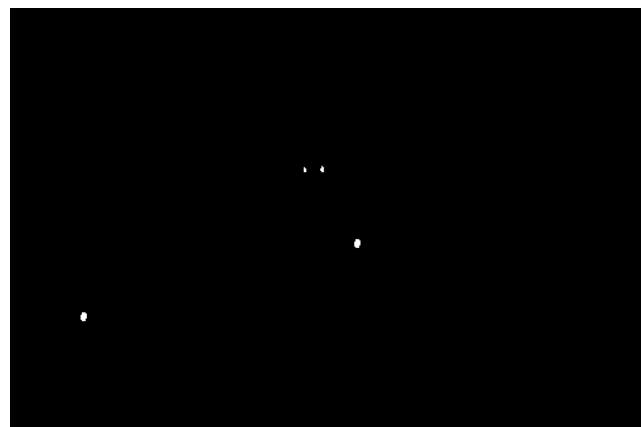
correlation

This is who I am.
Nobody said
you had to like it.

detected letter



0.9 max threshold



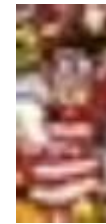
0.5 max threshold

NORMALIZED CROSS-CORRELATION

- Extension to intensity values
 - Handle variation in template and image brightness



scene

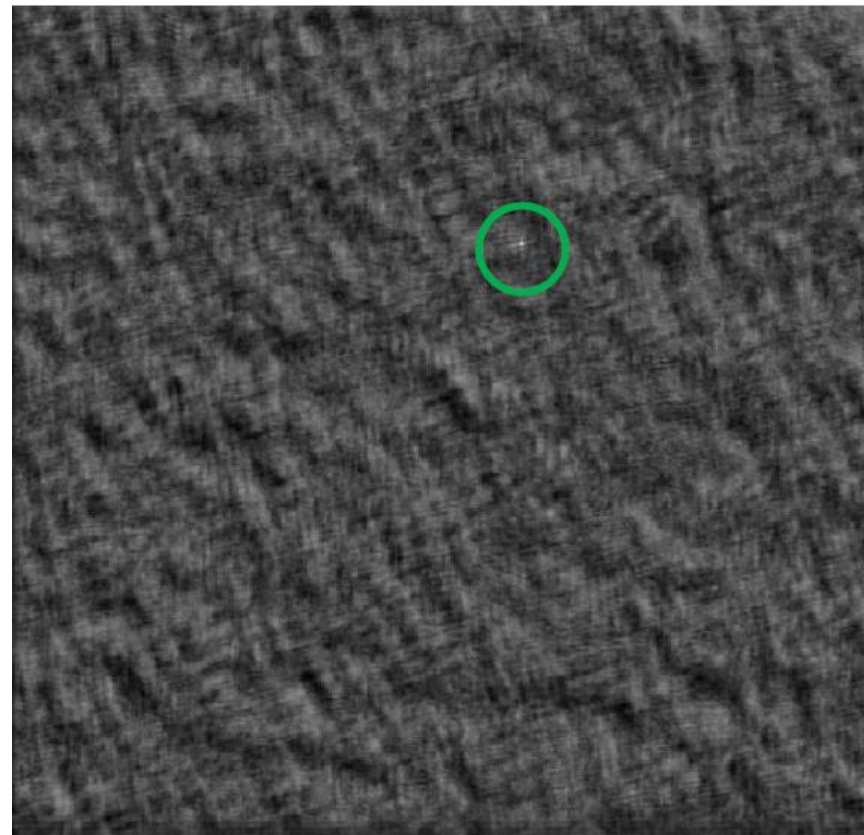


template

WHERE'S WALDO



Detected template



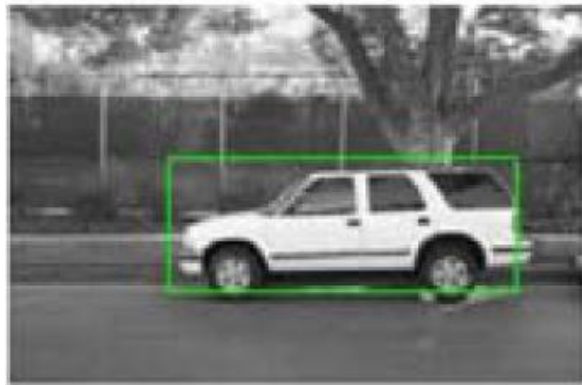
correlation map

DETECTION OF SIMILAR OBJECTS

- Previous examples are detecting exactly what we want to find
 - Give the perfect template
- What happens with similar objects
- What to do with different sized objects, new scenes



Template



Detected template

- Works fine when scale, orientation, and general orientation are matched

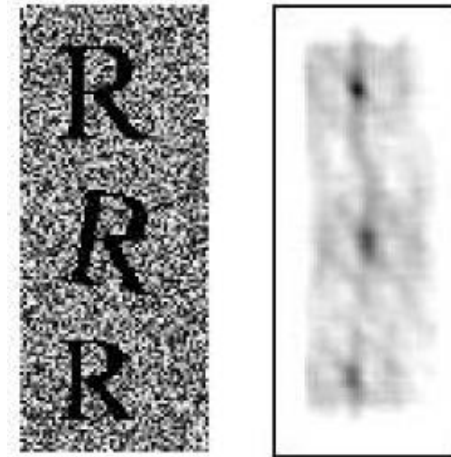


Figure 6.48: Template matching: A template of the letter **R** is sought in an image that has itself, a slightly rotated version, and a smaller version. The correlation response (contrast stretched for display) illustrates the diffuse response seen for even small adjustments to the original. © Cengage Learning 2015.

TEMPLATE MATCHING STRATEGIES

- Detection of parts
 - Full “pixel perfect” match may not exist, but smaller subparts may be matched
 - Connect subparts through elastic links
- Search at scale
 - Pattern matching is highly correlated in space
 - Neighborhoods around match have similar response
 - Search at low resolution first and go to higher resolution for refinement
 - Less comparisons, much faster
- Quit sure mismatches quickly
 - Do not compute full correlation when error is too large
 - Matches are rare so only spend time on heavy computation when required (cascade classifier later)

EVALUATING SEGMENTATIONS

- Need to know what is the “right” segmentation and then measure how close and algorithm matches
- Supervised approaches
 - Use “expert” opinions to specify segmentation
 - Evaluate by:
 - Mutual overlap
 - Border position errors (Hausdorff set distance)
- Unsupervised approaches
 - No direct knowledge of true segmentation
 - Avoid label ambiguity
 - Define criterion to evaluate region similarity and inter-region dissimilarity

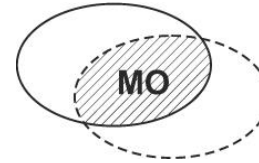


Figure 6.52: Mutual overlap: machine segmented region in solid, ground truth in dashed. © Cengage Learning 2015.

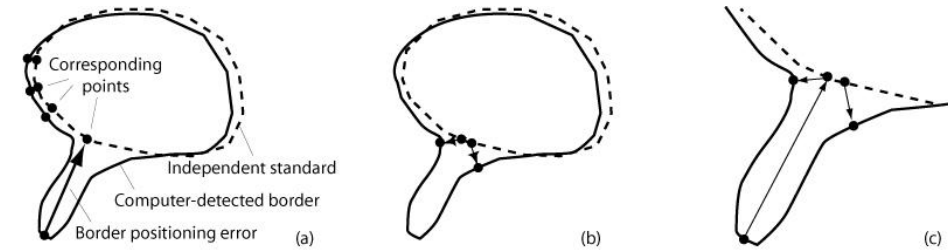


Figure 6.53: Border positioning errors. (a) Border positioning errors are computed as directed distances between the computer-determined and correct borders. (b) If errors are calculated in the opposite direction (from ground truth to the computer-determined border), a substantially different answer may result. (c) Zoomed area showing the difference in calculating directional errors. © Cengage Learning 2015.

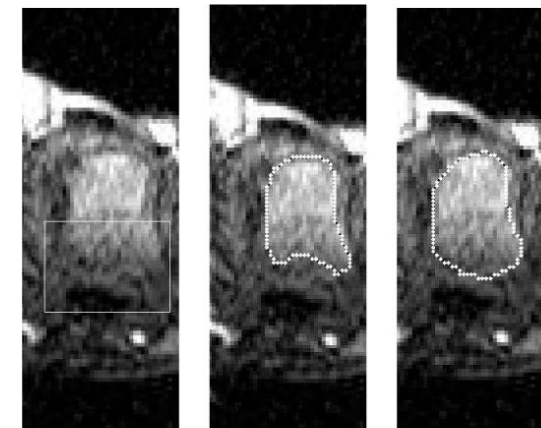


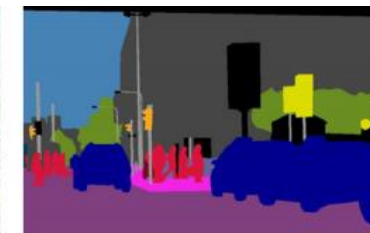
Figure 6.51: A region from a dynamically enhanced MRI study with partly ambiguous boundary. Two different experts have overlaid their judgments. Courtesy of O. Kubassova, S. Tanner, University of Leeds.

DEEP SEGMENTATION

- Beyond finding regions → semantic segmentation
 - Assign each pixel with a predefined category label
- Use convolutional neural networks (CNNs) to encode image features followed by “decoding” to generate segmented image
- Highly dependent on data
 - Lots of effort to generate large, high quality datasets
- Survey papers:
 - [Image Segmentation Using Deep Learning: A Survey \(2020\)](#)
 - [A survey on deep learning techniques for image and video semantic segmentation \(2018\)](#)
 - [A Brief Survey on Semantic Segmentation with Deep Learning \(2020\)](#)



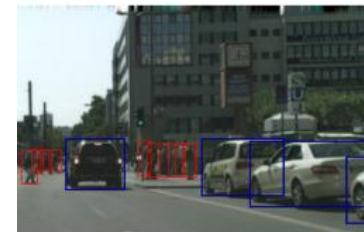
(a) Image



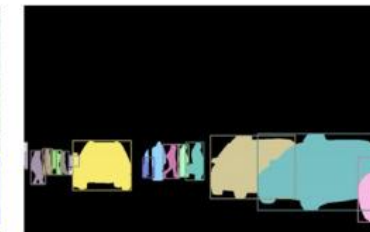
(b) Semantic segmentation



(c) Image classification



(d) Object detection



(e) Instance segmentation



(f) Panoptic segmentation