IMAGE SEGMENTATION

Aziz M. Qaroush

Birzeit University

First Semester 2015/2016

Outline

- Element of Image Analysis
- Fundamentals
- Point, Line, and Edge Detection
- Edge Linking and Boundary Detection
- Thresholding
- Region-based Segmentation

Element of Image Analysis

Preprocess

Image acquisition, restoration, and enhancement

↓

Intermediate process

Feature extraction & Image segmentation

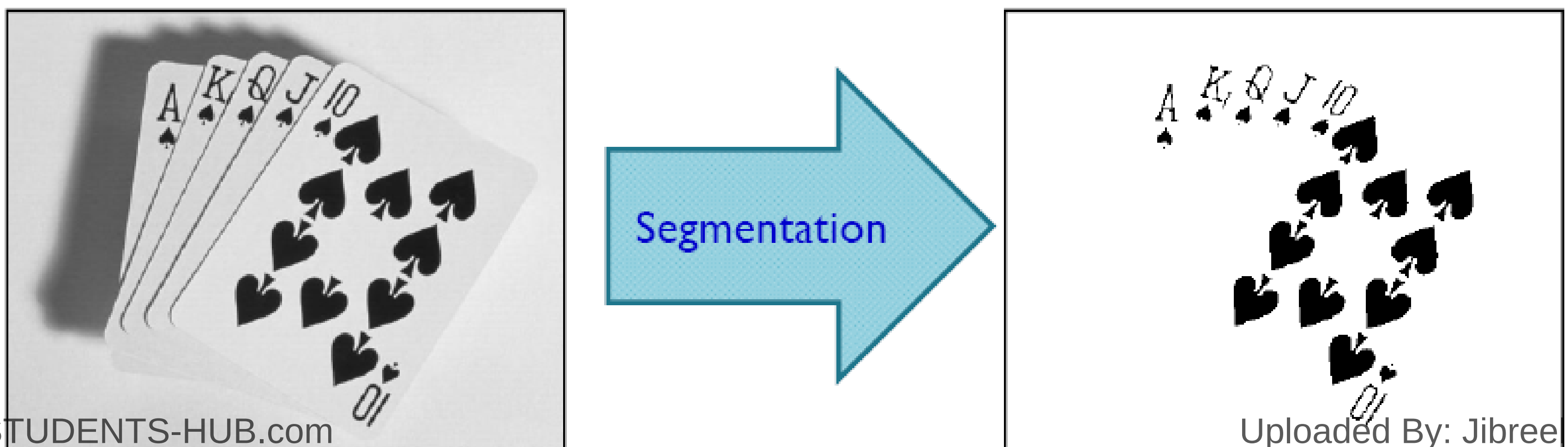
↓

High level process

Image interpretation and recognition

Fundamentals

- **What is image segmentation?**
  - The process of subdividing an image into its individual components/objects
  - A central operation in all machine vision applications
  - The level of subdivision depends on the problem in hand
  - It is nontrivial for complex images
  - The accuracy of the segmentation process determines the correct operation of computerized vision systems



Fundamentals

- Let R represents the entire spatial region occupied by the image. The segmentation processes subdivides R into n subregions $R_1, R_2, \ldots, R_n$ such that

(1) $\bigcup_{i=1}^{n} R_i = R$

(2) R_i is a connected set, i = 1,2,3,...,n

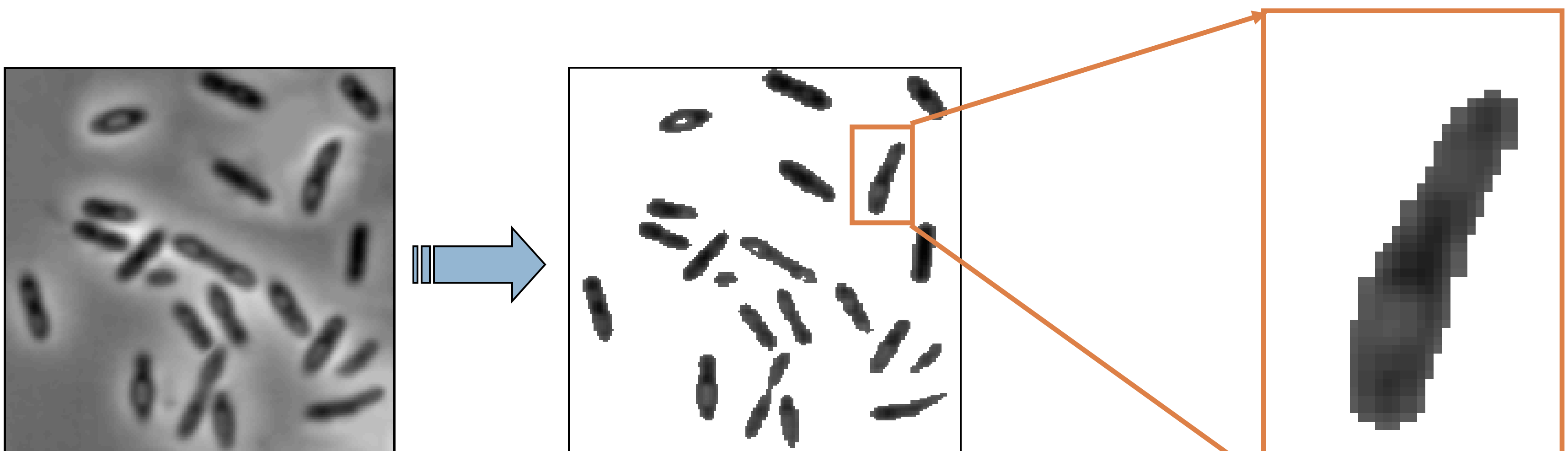
(3) $R_i \cap R_j = \varnothing$, for all i and j, i ≠ j

(4) $Q(R_i) =$ TRUE, for i = 1,2,3,...,n

(5) $Q(R_i \cup R_j) =$ FALSE, for any adjacent regions R_i and R_j

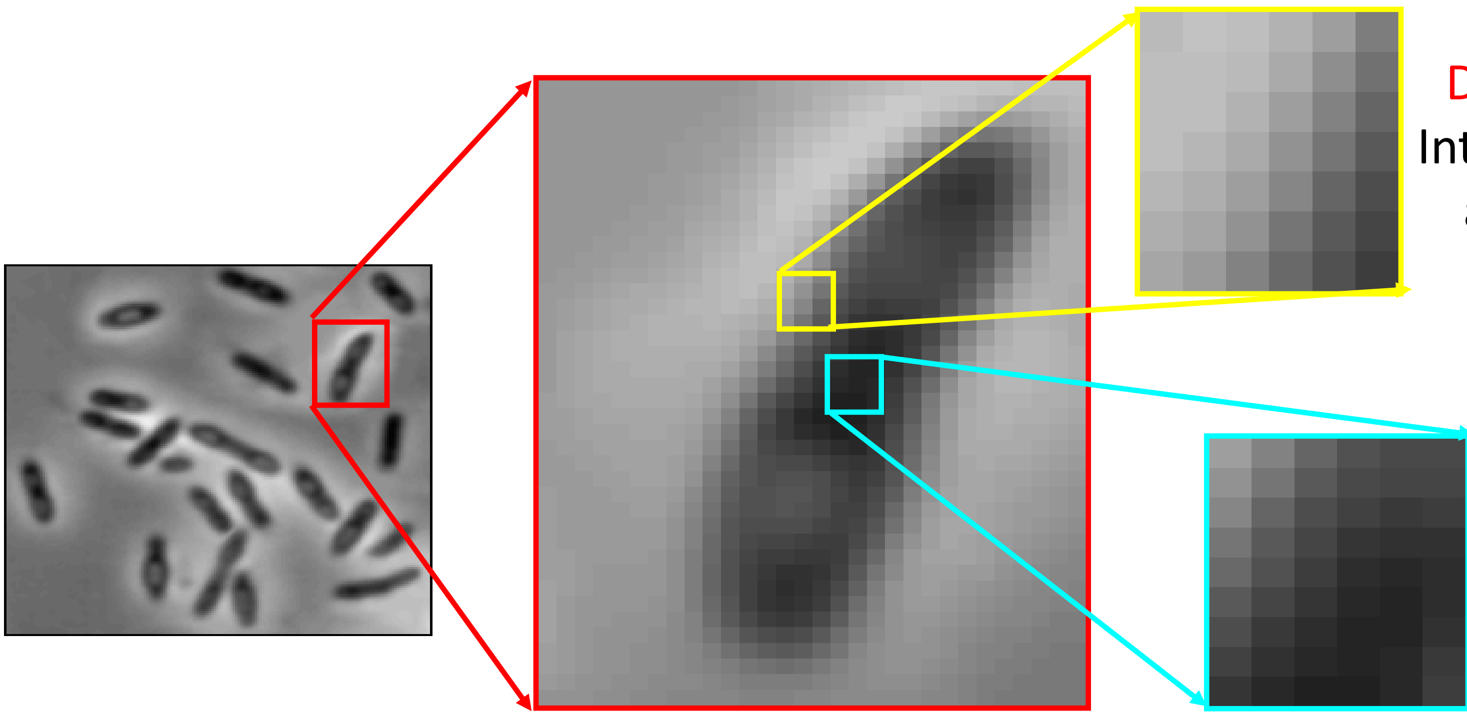
Fundamentals

- Importance of Image Segmentation
  - Image segmentation is used to separate an image into constituent parts based on some image attributes. Image segmentation is an important step in image analysis
    1. Image segmentation reduces huge amount of unnecessary data while retaining only importance data for image analysis
    2. Image segmentation converts bitmap data into better structured data which is easier to be interpreted



Fundamentals

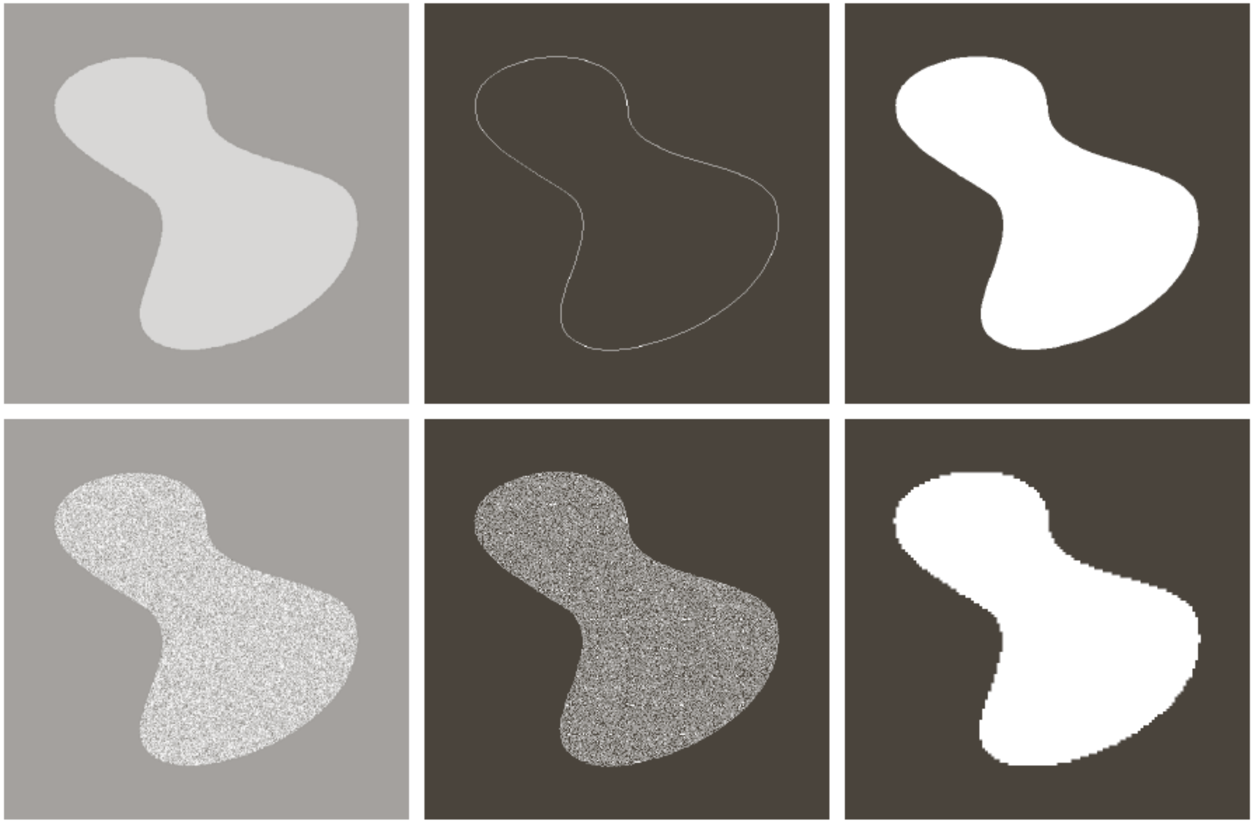
- Image Attributes for Image Segmentation
  1. Similarity properties of pixels inside the object are used to group pixels into the same set.
  2. Discontinuity of pixel properties at the boundary between object and background is used to distinguish between pixels belonging to the object and those of background.



Discontinuity:
Intensity change at boundary

Similarity:
Internal pixels share the same intensity

Fundamentals

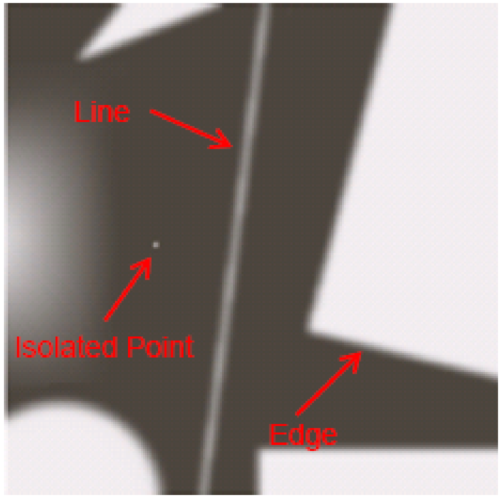


a b c
d e f

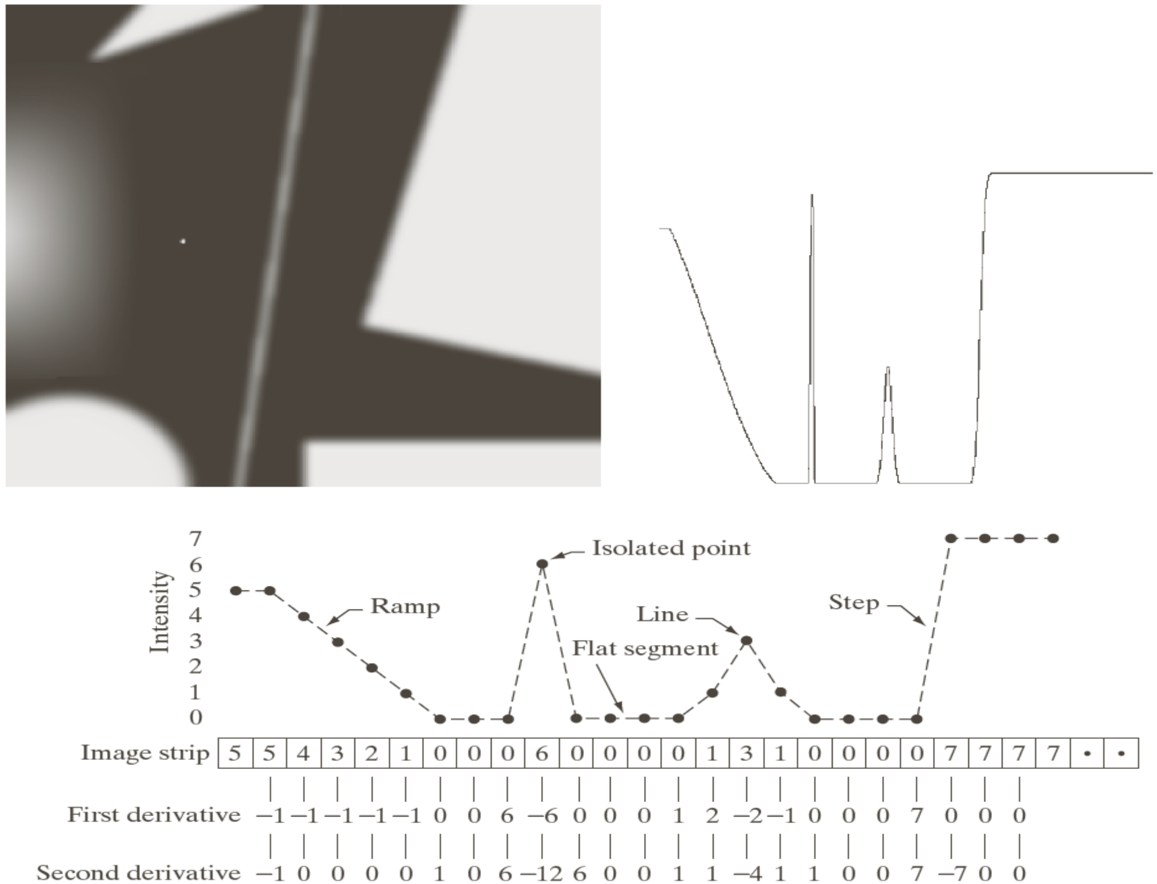
FIGURE 10.1 (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large number 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.

Discontinuity of pixel properties: Point, Edge, Line Detection

- The interest is in detecting sharp local changes in intensity
- Primarily, we are interested in three types of discontinuities: *points, edges, and lines*
- ***Edges***
  - *Edge pixels are pixels at which the intensity of an image function changes abruptly*
  - *Edges are sets of connected edge pixels*
- ***Lines***
  - *Are edge segments in which the intensity of the background on either side of the line is much higher or lower than the intensity of line pixels*
- ***Point***
  - *A line whose length and width is roughly one pixel*



Point, Edge, Line Detection



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).

Point, Edge, Line Detection

- The discontinuities we interested in can be detected using derivatives
- ***First Derivative*** $$f'(x) = f(x+1) - f(x)$$
- ***Second Derivative*** $$f''(x) = f(x+1) + f(x-1) - 2f(x)$$

- ***Notes***
  - First derivative generally produce thicker edges
  - Second order derivative has stronger response to fine detail
  - Second order derivative produce double-edge response at ramp and step transitions
  - The sign of the second derivative determines the type of transition

Point Detection

- We will use second derivative as it has stronger response to discontinuities

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

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

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

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

0	1	0
1	-4	1
0	1	0

or

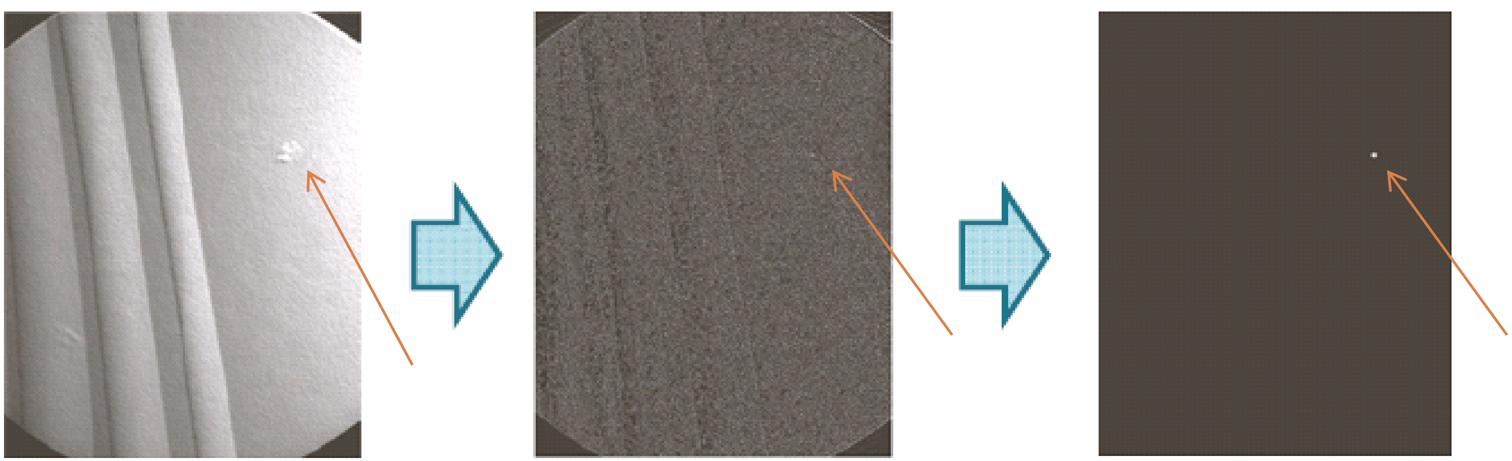
1	1	1
1	-8	1
1	1	1

Point Detection

- We saw earlier that isolated points have very high response for second derivatives
- We can detect the points by simply thresholding the absolute value of the second derivative R(x,y) using

$$g(x,y) = \begin{cases} 1, & |R(x,y)| \geq T \\ 0, & otherwise \end{cases}$$

- The main concern is in specifying the threshold value



Image

Second Derivative Image

Thresholded Second Derivative Image with T = 0.95 max(R)

Line Detection

- Again , we will use second derivative as it has stronger response to discontinuities and produces thinner edges than first derivative

- Keep in min that the second derivative produces double-line effect. Solutions:
  - Take the absolute value of the Laplacian image (produces thicker lines)
  - Take the positive values of the Laplacian image only (in noisy images we have to use a threshold to reduce the effect of noise)

Line Detection

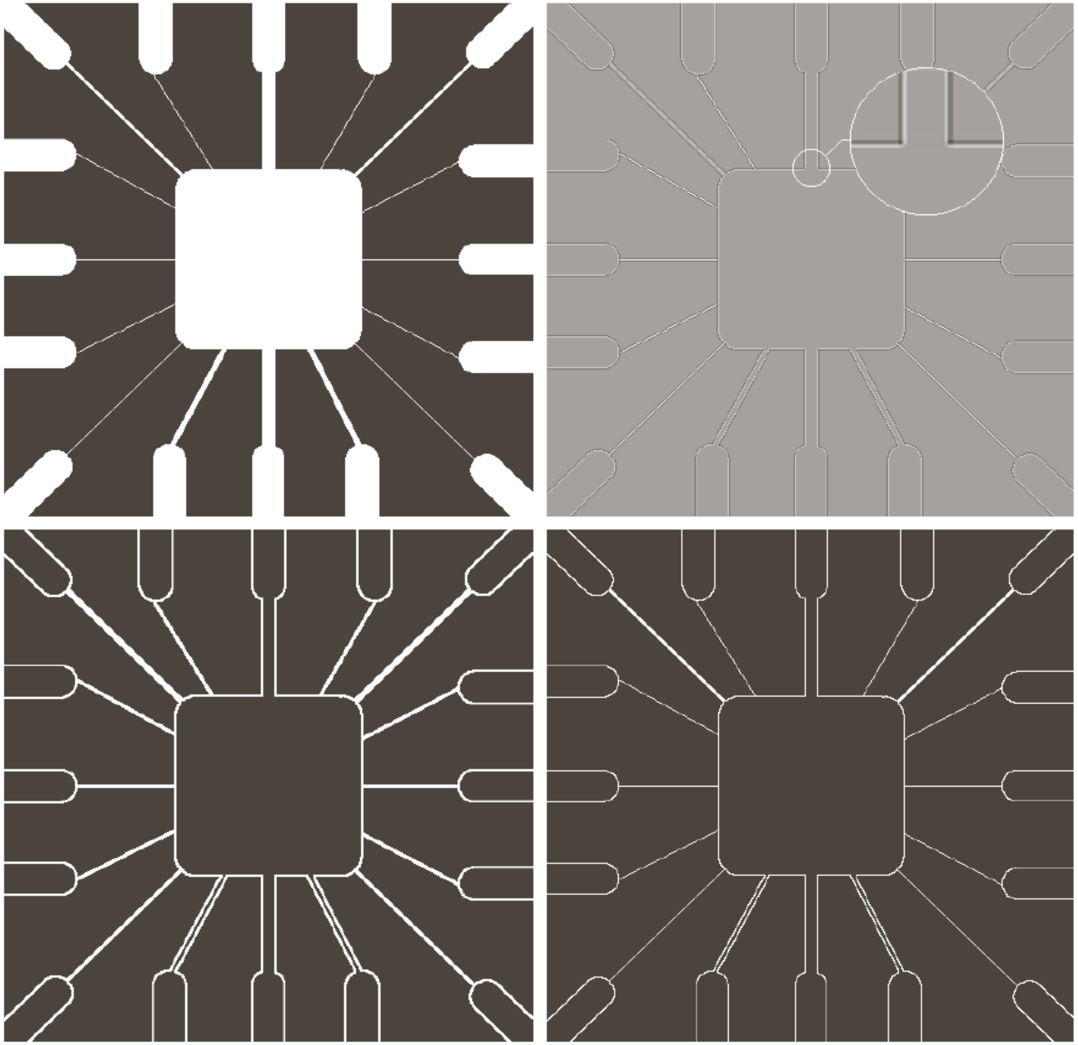


FIGURE 10.5
(a) Original image.
(b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian.
(c) Absolute value of the Laplacian.
(d) Positive values of the Laplacian.

Line Detection

- The mask used in the previous is isotropic, i.e. it is independent of line direction.
- It is desirable sometimes to detect lines in specific directions (0°, +45°, -45°, and 90°). We can use masks whose coefficients along the desired direction have larger weights

−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

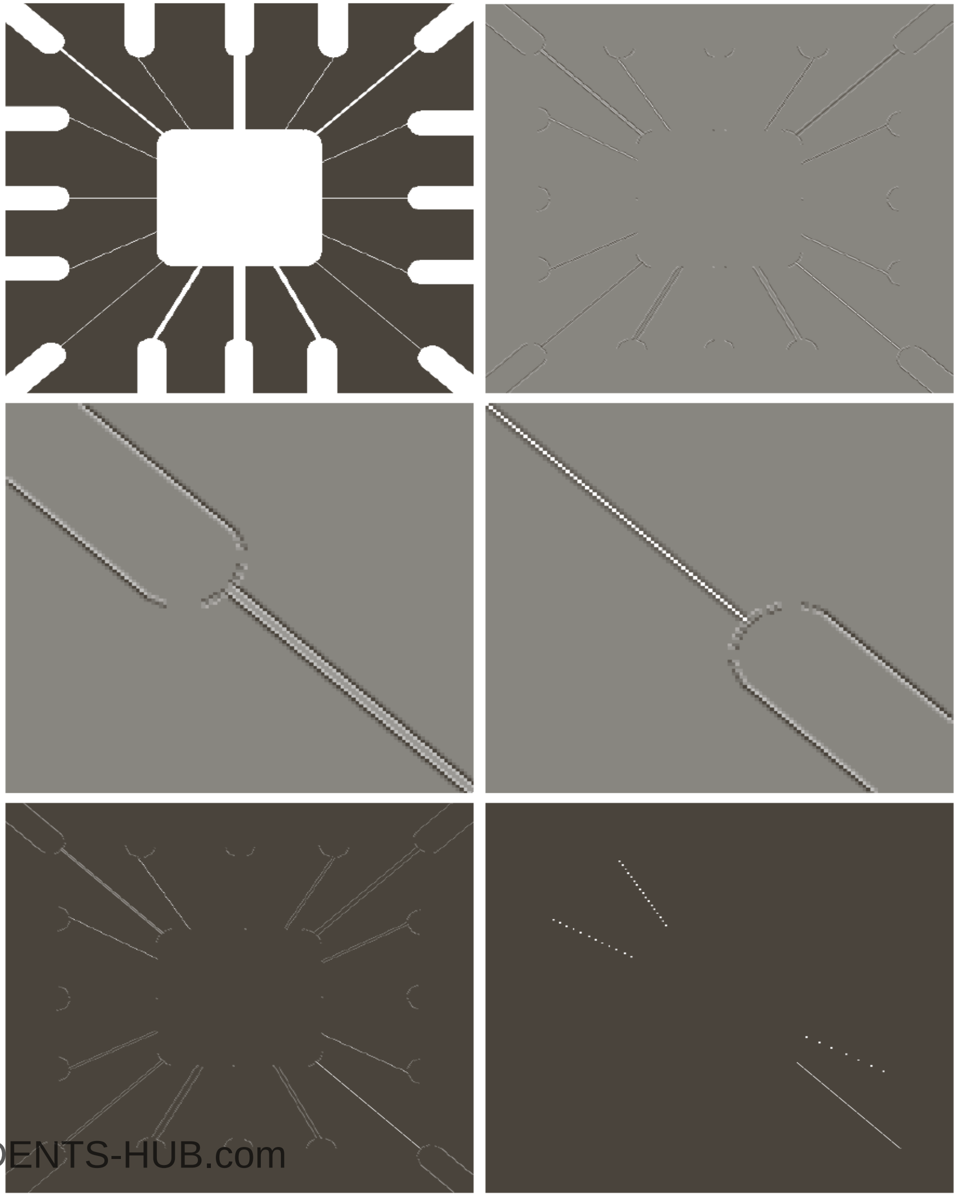
Vertical

−1	−1	2
−1	2	−1
2	−1	−1

−45°

- These masks provide strong response for lines in the specified direction and that are <u>one pixel thick</u>
- In order to detect a line in a specific direction, we use the correct mask then we threshold for <u>the positive result only</u>

Line Detection



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 Detection

- **Edges can be classified according to intensity profiles into**
  - Step Edge (ideal edge)
    - *involves transition between two intensity levels over a distance of one pixel*
    - *Occur mostly in computer generated images*
  - *Ramp Edge*
    - *The transition between two intensity levels occur over a distance that is greater than one pixel*
    - *Appear in real images as a result of noise and focusing limitations of imaging devices*
  - *Roof Edge*
    - *Essentially, they represent blurred lines that pass through a region*

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.

Edge Detection

- It is not unusual to find the three types of edges in one image

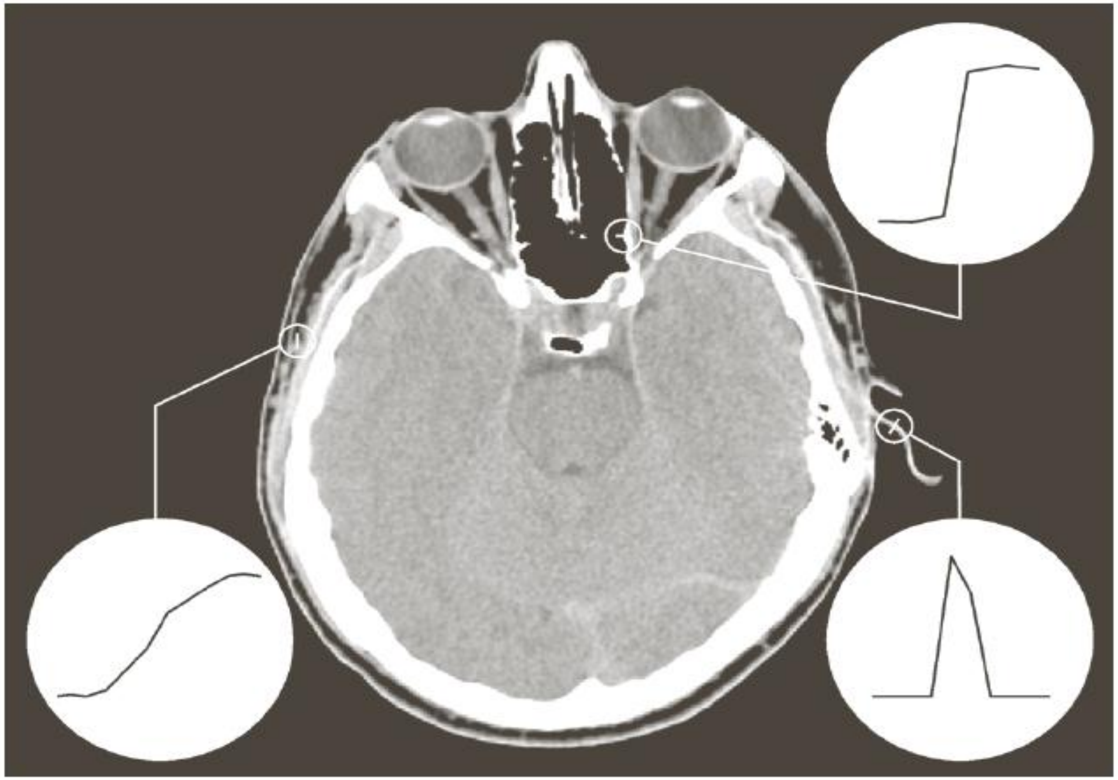
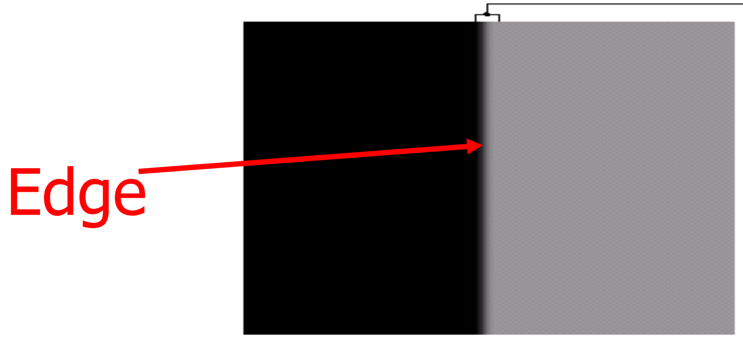


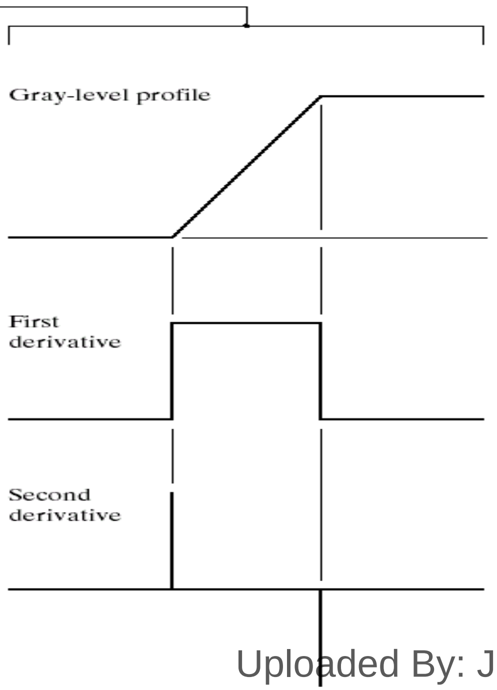
FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively.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

Edge Detection

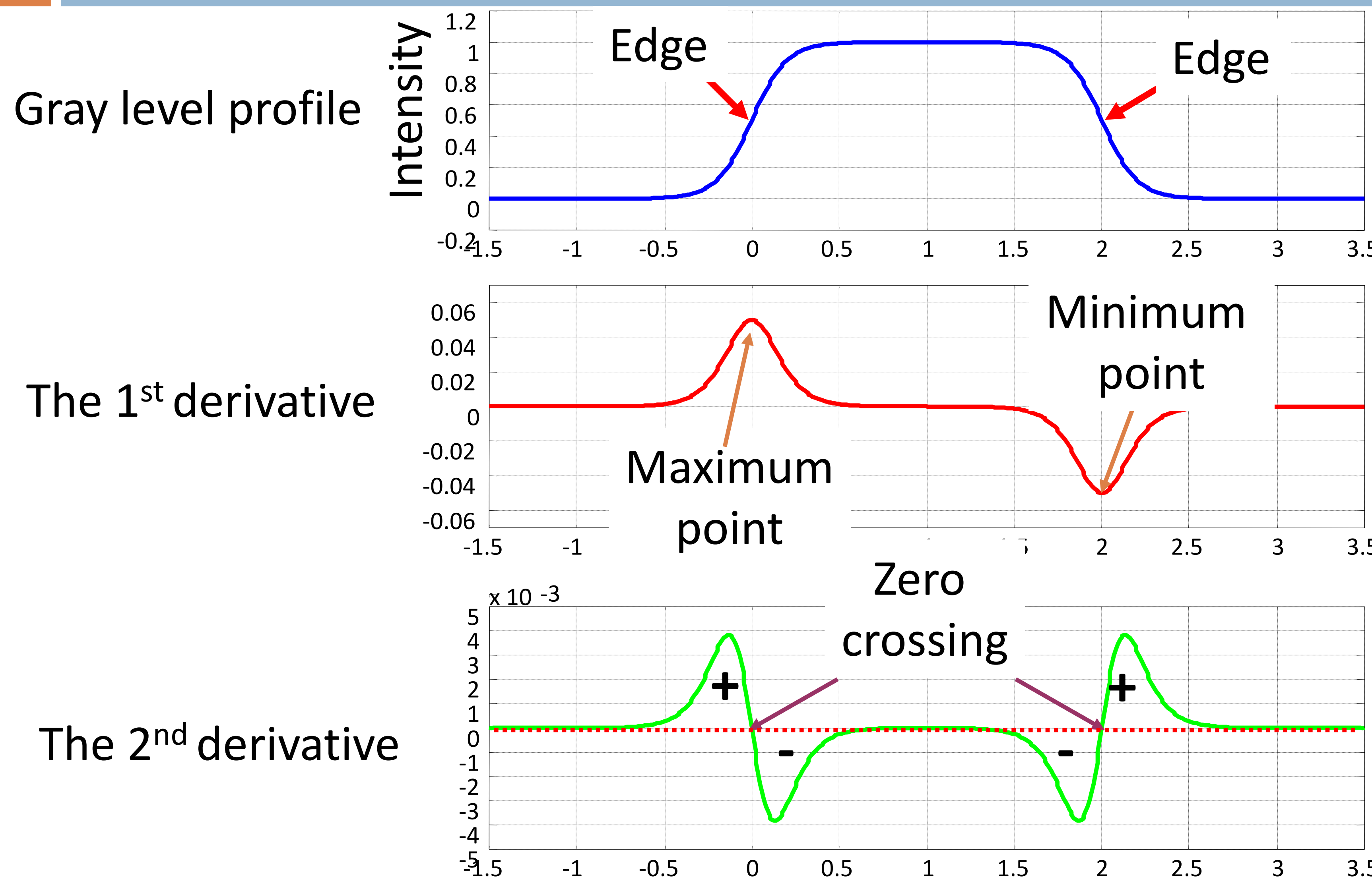
- Detection of edges can be through the use of first-order or second-order derivatives
  - *For the first derivative, the magnitude can be used to detect the presence of an edge*
  - *For the second derivative, the sign of the second derivative is used to detect the presence of the edge*



Original image



Edge Detection



Therefore, for detecting edges, we can apply zero crossing detection to the 2nd derivative image or thresholding the absolute of the 1st derivative image

Edge Detection Using Gradient

- The gradient is a powerful tool in finding the strength and direction of edges. The gradient at pixel (x,y) is defined as

$$\nabla \mathrm{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

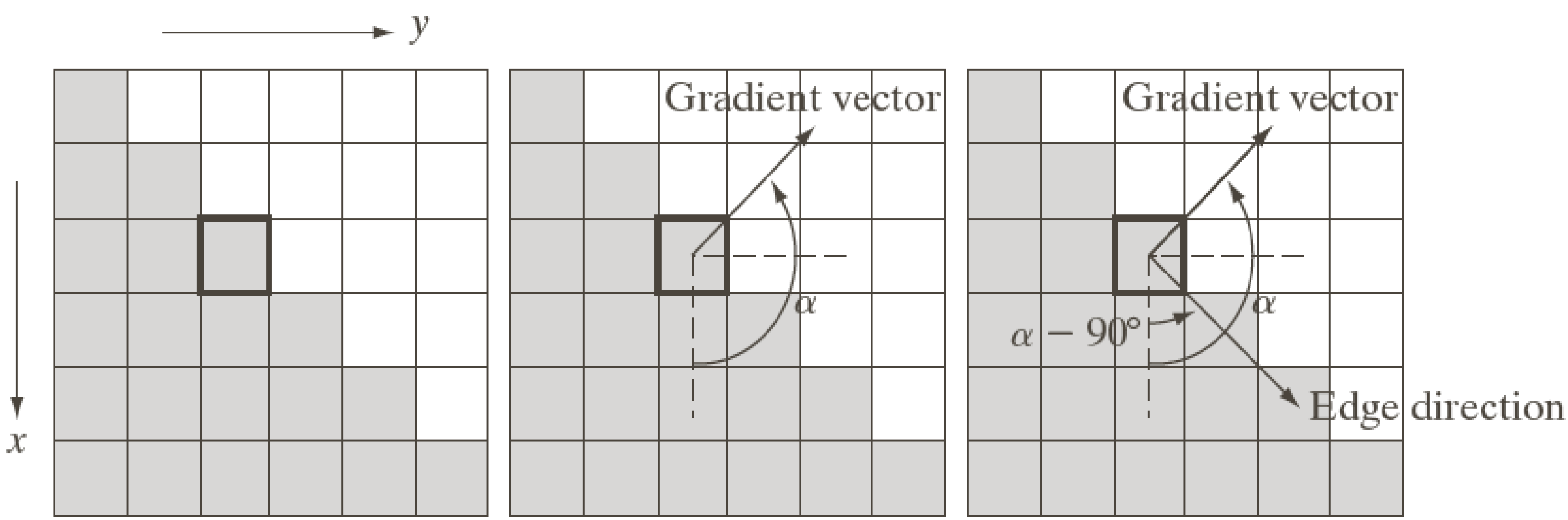
- The magnitude of the gradient measures the strength of the edge (maximum rate of change)

$$M(x,y) = \left[\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2 \right]^{1/2} \qquad |\nabla f| \approx |G_x| + |G_y|$$

- The direction of the gradient is perpendicular to the edge direction

$$\alpha = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Edge Detection Using Gradient



$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

$$\alpha = \tan^{-1}\left(\frac{G_y}{G_x}\right) = 135^o$$

- All edge pixels have the same gradient magnitude and direction

Edge Detection Using Gradient

- **Gradient Masks to compute gradient at Z_5**
  - **Discrete 1st Derivative**

$$G_x = \frac{\partial f}{\partial x} = f(x+1,y) - f(x,y) = Z_8 - Z_5$$

-1
1

$$G_y = \frac{\partial f}{\partial y} = f(x,y+1) - f(x,y) = Z_6 - Z_5$$

-1	1

- Not efficient in detecting diagonal edges

  - **Roberts Cross-gradient Operator**

-1	0
0	1

Horizontal Operator

0	-1
1	0

Vertical Operator

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Pixel z5 and its neighbours

$$G_x(x,y) = z_9 - z_5$$

$$G_y(x,y) = z_8 - z_6$$

  - 2x2 masks are not as good as symmetric masks which capture information from opposite sides around the center point

Edge Detection Using Gradient

- Gradient Masks to compute gradient at Z_5
  - Prewitt Operators

-1	-1	-1
0	0	0
1	1	1

Mask to Compute Gx

-1	0	1
-1	0	1
-1	0	1

Mask to Compute Gy

$$G_x = \frac{\partial f}{\partial x} = (Z_7 + Z_8 + Z_9) - (Z_1 + Z_2 + Z_3)$$

$$G_y = \frac{\partial f}{\partial y} = (Z_3 + Z_6 + Z_9) - (Z_1 + Z_4 + Z_7)$$

  - Sobel Operators

-1	-2	-1
0	0	0
1	2	1

Mask to Compute Gx

-1	0	1
-2	0	2
-1	0	1

Mask to Compute Gy

$$G_x = \frac{\partial f}{\partial x} = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)$$

$$G_y = \frac{\partial f}{\partial y} = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7)$$

  - They have better response than Prewitt masks and have better smoothing which is essential to reduce the effect of noise

Edge Detection Using Gradient

- **Gradient Masks to compute gradient at Z_5**
  - Prewitt and Sobel masks shown before give the strongest response for horizental and vertical edges. We can modify these masks to obtain better response for diagonal edges **Prewitt Operators**

0	1	1
-1	0	1
-1	-1	0

-1	-1	0
-1	0	1
0	1	1

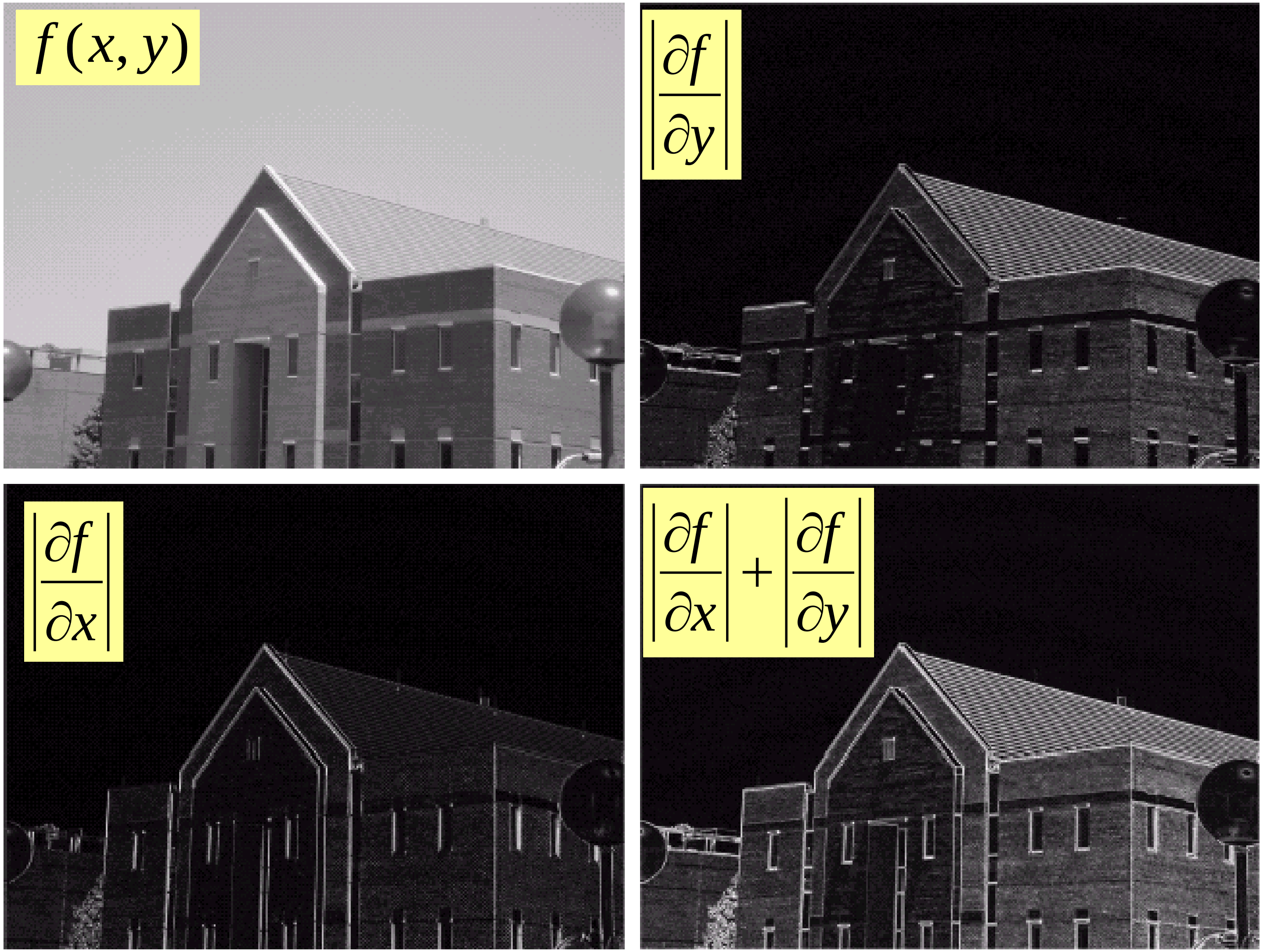
Prewitt Diagonal Masks

0	1	2
-1	0	1
-2	-1	0

-2	-1	0
-1	0	1
0	1	2

Sobel Diagonal Masks

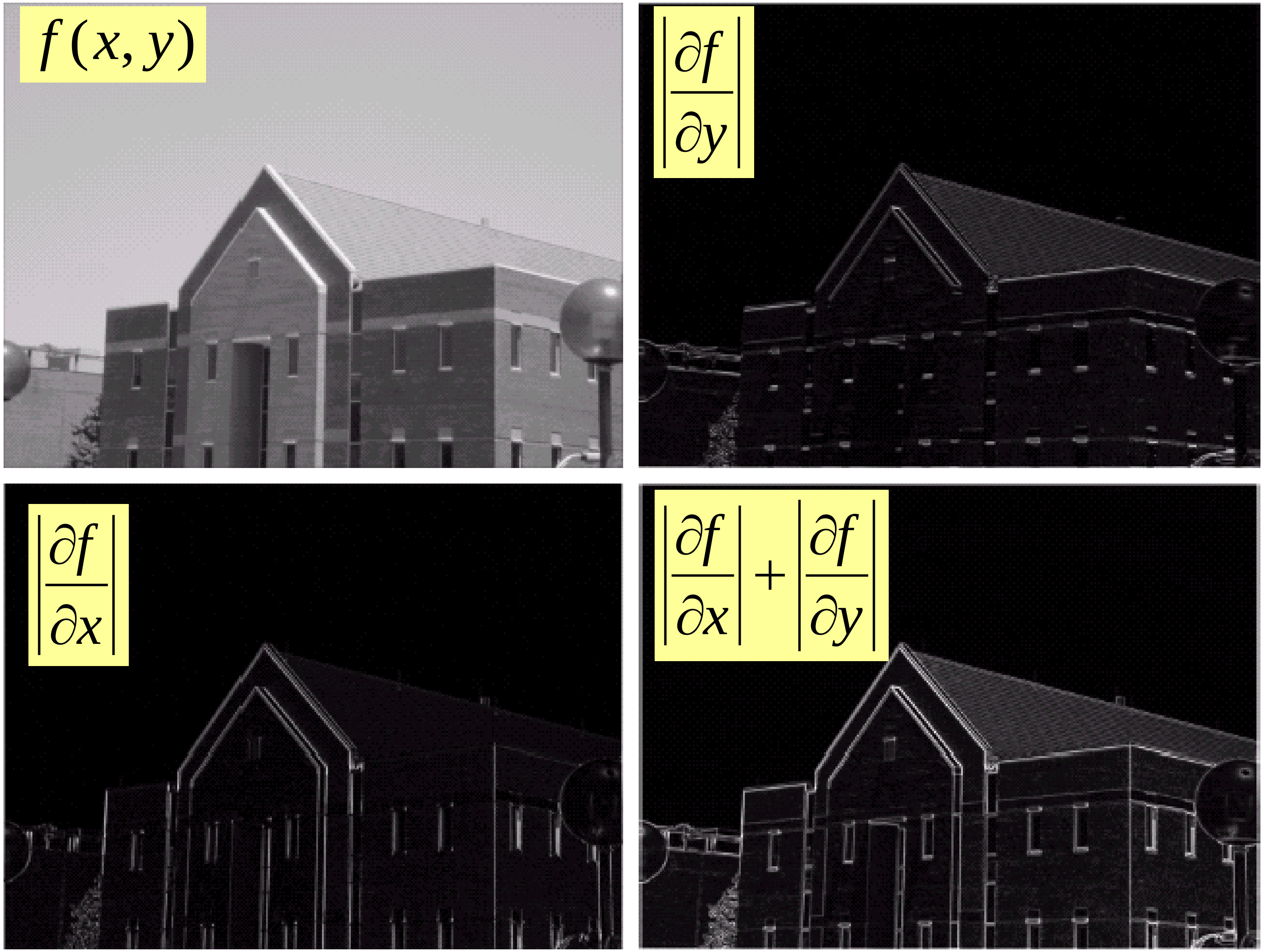
Edge Detection Using Gradient



Edge Detection Using Gradient

- **Example – continued**
  - Note that in the previous slide, the edges of the wall bricks were successfully detected. However, this might not be desirable of the walls if we are interested in the main edges
  - We can eliminate such small edges (which might considered as noise) by
    - Smoothing the image before computing the gradient
    - Thresholding the gradient image
    - Smoothing followed by thresholding

Edge Detection Using Gradient



Note: the original image is smoothed by a 5x5 moving average mask first.

Edge Detection Using Gradient

- **Example – continued**

Gradient Image (|Gx| + |Gy|) without smoothing

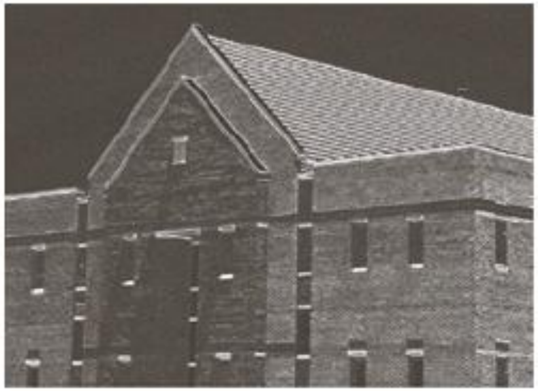
Thresholded Gradient Image

Gradient Image (|Gx| + |Gy|) after the original image was smoothed by 5x5 mask

Thresholded Gradient Image (better connectivity for edges)

Edge Detection Using Gradient

- **Example – continued**
  - The Sobel masks used in the previous slides were those that have stronger response for vertical and horizontal directions. How about diagonal directions?



Gx computed using Sobel Operator (Horizental)

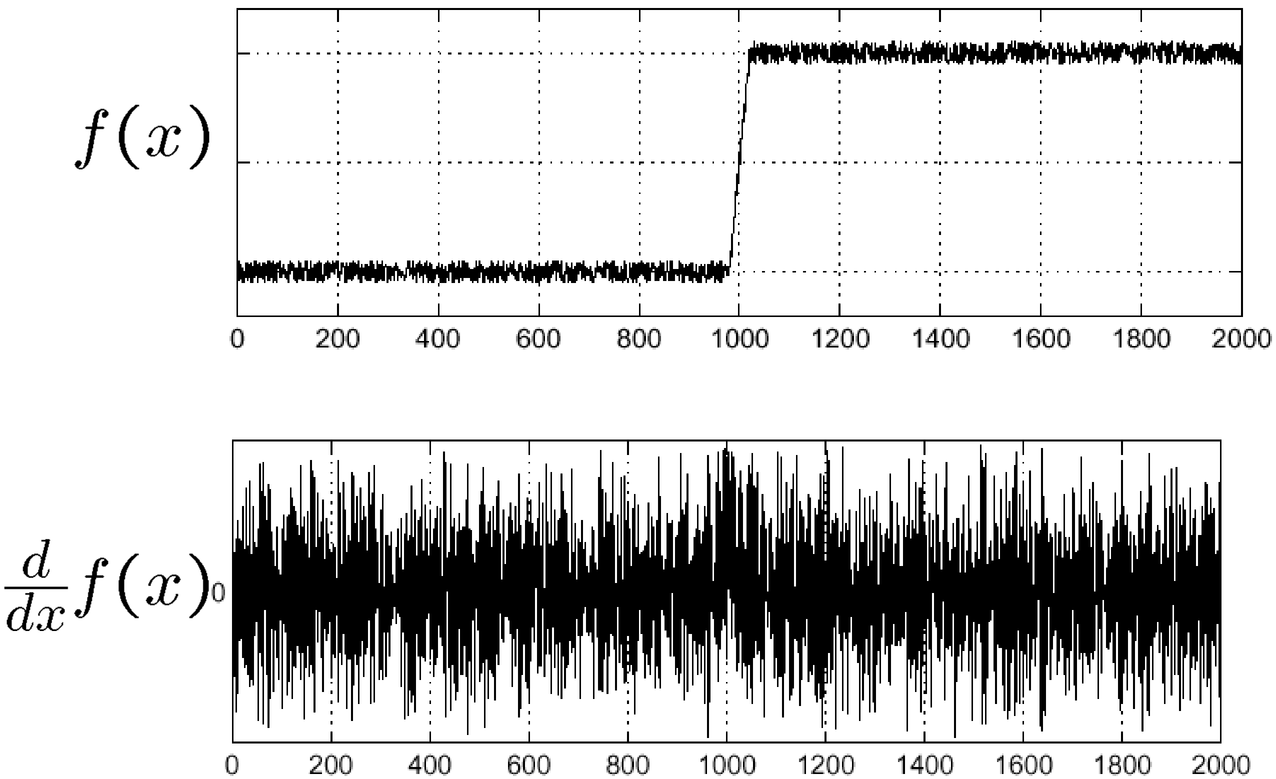
Gy computed using Sobel Operator (vertical)

Gx computed using Diagonal Sobel Operator (+45)

Gx computed using Diagonal Sobel Operator (-45)

Effects of noise

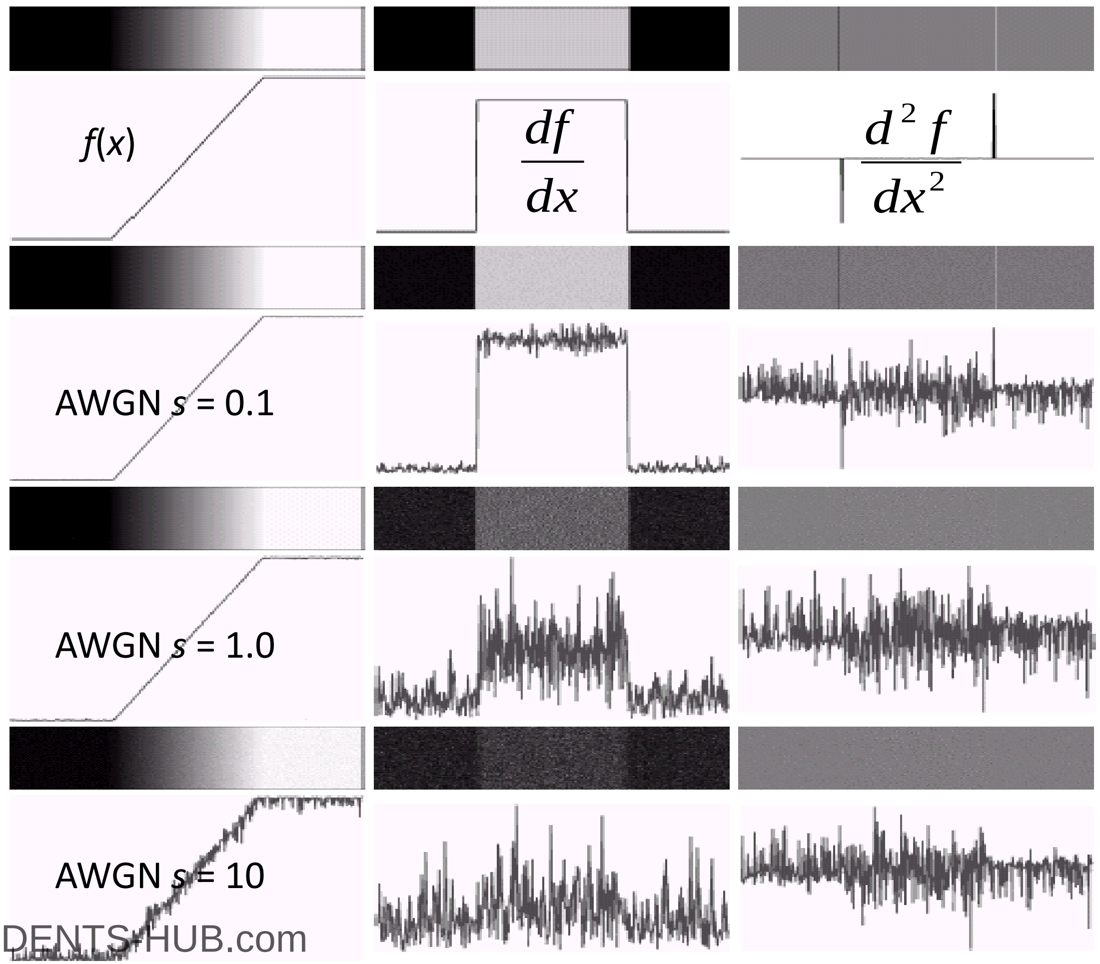
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a *signal*



Where is the edge?

Edge Detection with Noise

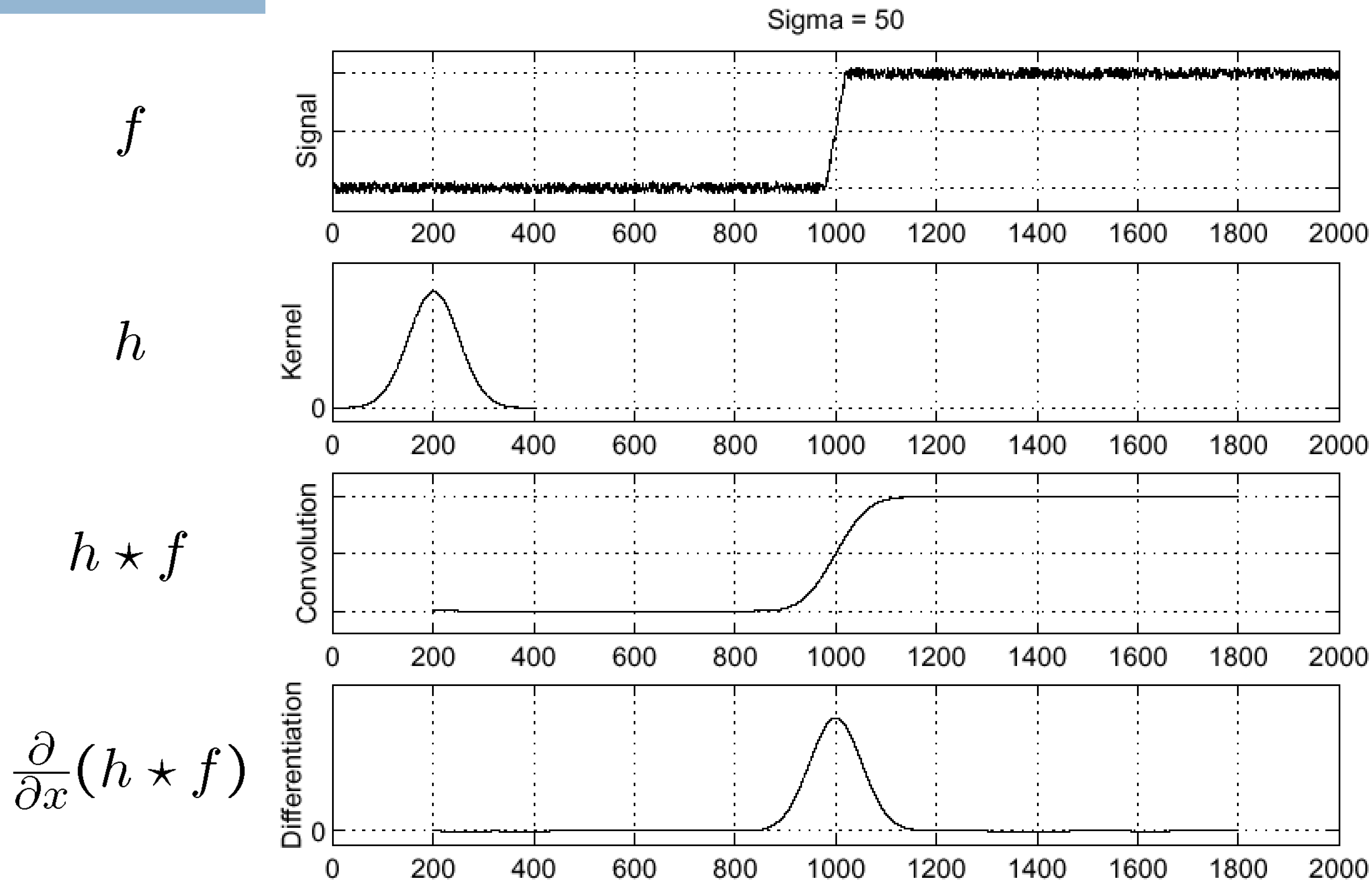
- Although it sounds straight forward to detect edges using firs-order or second-order derivatives, the presence of noise may affect this operation significantly depending on noise level



In general, detecting edges, involves :

1) Smoothing to reduce the sensitivity of derivatives to noise
2) Detection of all possible edge points
3) Edge localization to select from the candidate point those that comprise the edge

Solution: smooth first



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

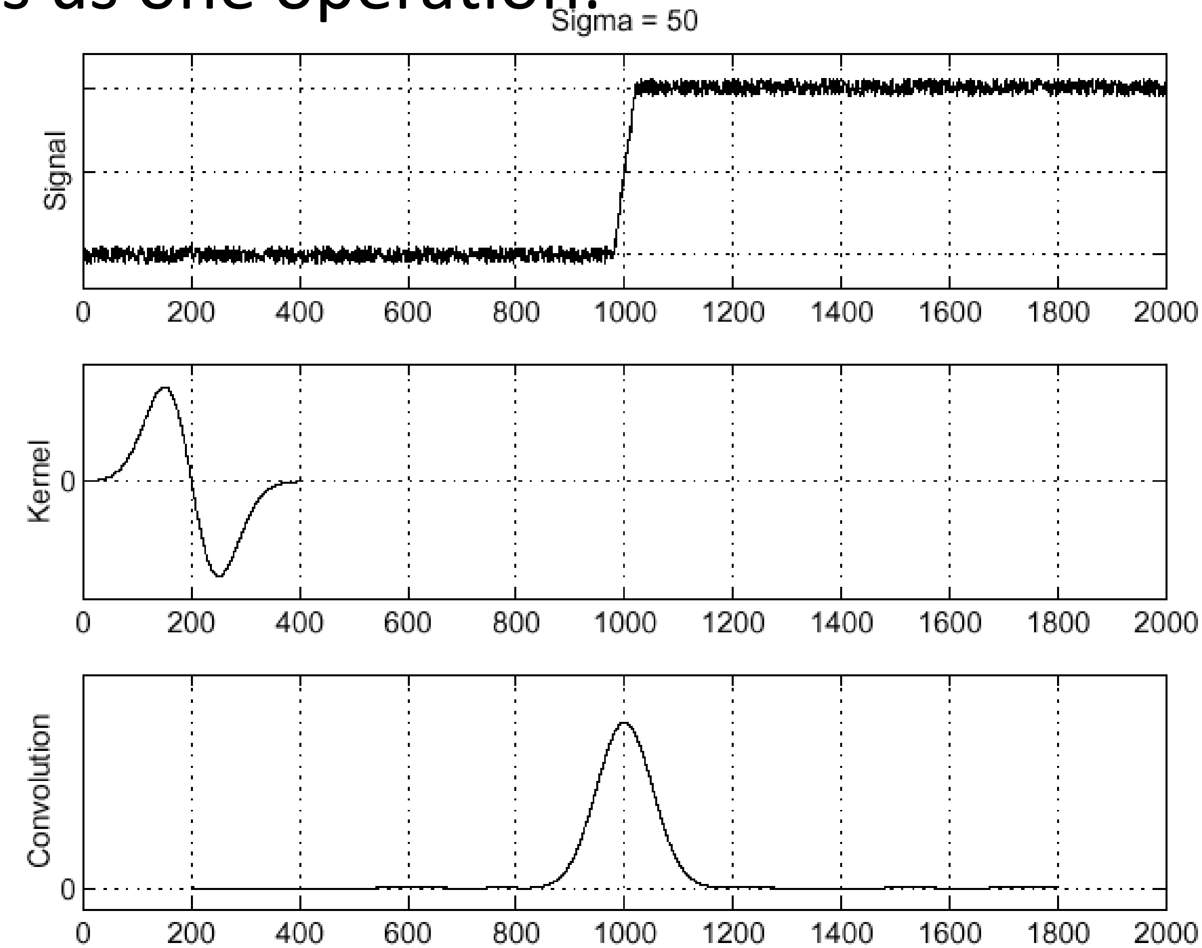
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

- This saves us one operation:

f

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

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



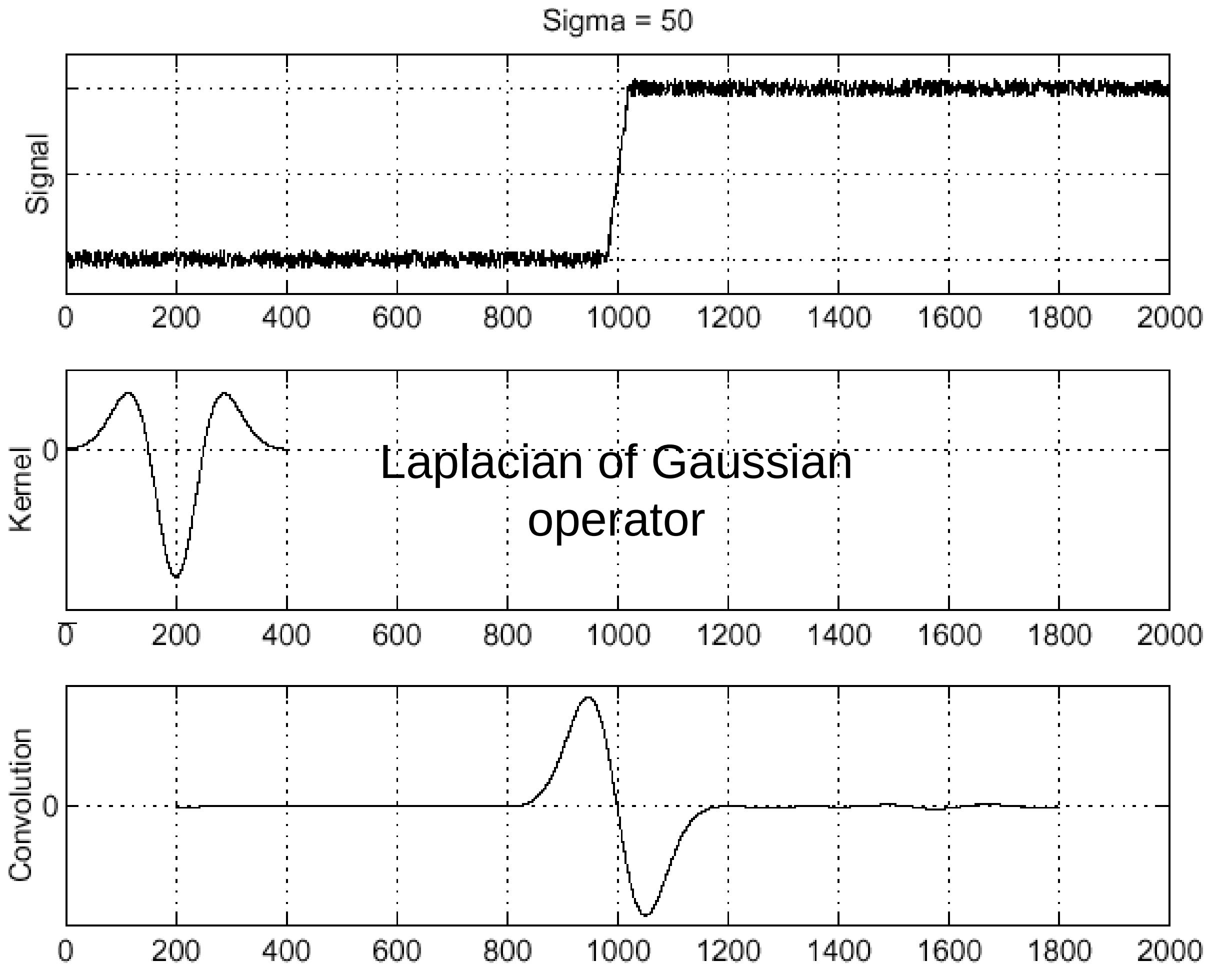
Laplacian of Gaussian

- Look for zero-crossings of $\frac{\partial^2}{\partial x^2}(h \star f)$

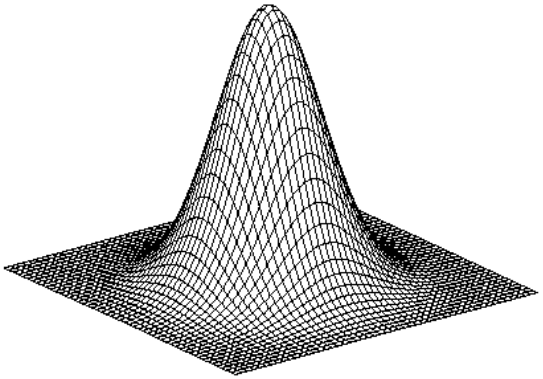
$$f$$

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

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

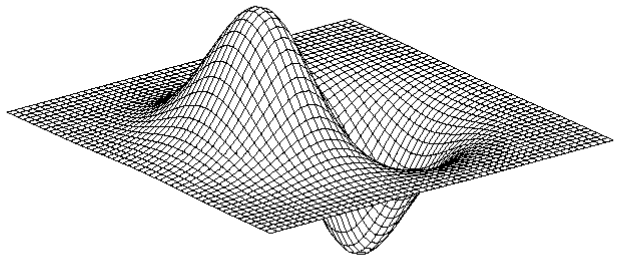


2D edge detection filters



Gaussian

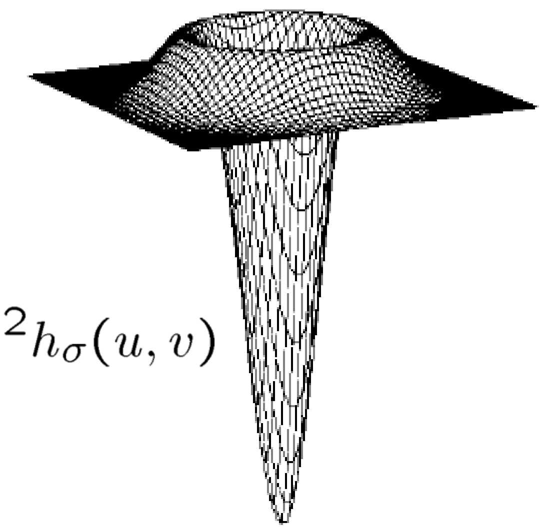
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

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

Laplacian of Gaussian



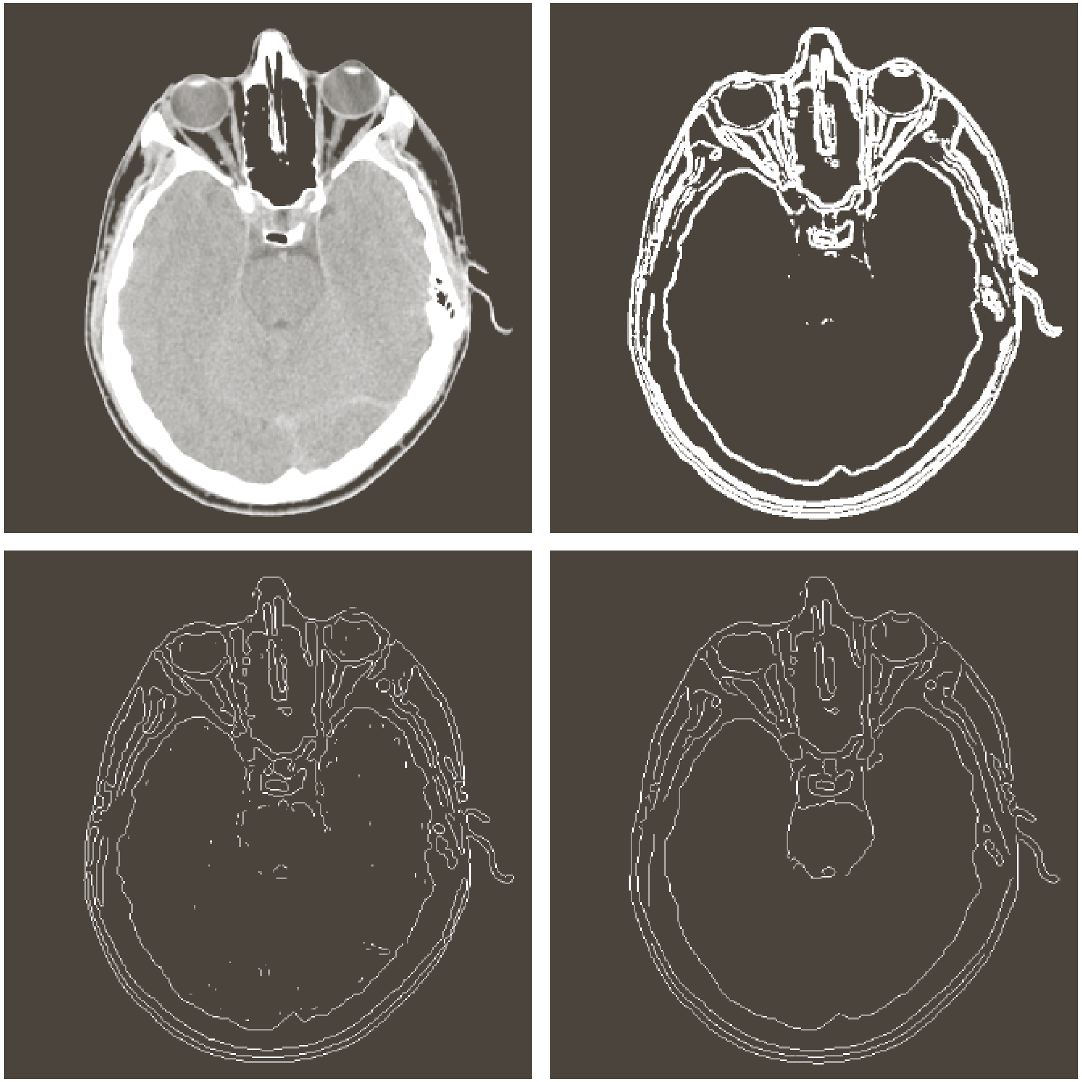
$$\nabla^2 h_\sigma(u,v)$$

∇^2 is the **Laplacian** operator:

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

- **The Canny Edge Detector**
  - Assume:
    - Linear filtering
    - Additive Gaussian noise
  - Edge detector should have:
    - Good Detection. Filter responds to edge, not noise.
    - Good Localization: detected edge near true edge.
    - Single Response: one per edge.

Advanced Techniques for Edge Detection



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.)

Edge Detection Summery

- Since edge detection is the initial step in object recognition, it is important to know the differences between edge detection techniques.
  - Gradient-based algorithms such as the Prewitt filter have a major drawback of being very sensitive to noise.
    - The size of the kernel filter and coefficients are fixed and cannot be adapted to a given image.
    - An adaptive edge-detection algorithm is necessary to provide a robust solution that is adaptable to the varying noise levels of these images to help distinguish valid image contents from visual artifacts introduced by noise.

Edge Detection Summery

- The performance of the Canny algorithm depends heavily on the adjustable parameters, , which is δ and the threshold values, 'T1' and 'T2'.
  - The bigger the value for δ, the larger the size of the Gaussian filter becomes. This implies more blurring, necessary for noisy images, as well as detecting larger edges.
  - However, the larger the scale of the Gaussian, the less accurate is the localization of the edge.
  - The user can tailor the algorithm by adjusting these parameters to adapt to different environments.
  - Canny's edge detection algorithm is computationally more expensive compared to Sobel, Prewitt and Robert's operator. However, the Canny's edge detection algorithm performs better than all these operators under almost all scenarios

Edge Detection Summery

Original (a)

Sobel (b)

(c)

Prewitt

(d)

Roberts

Edge Detection Summery

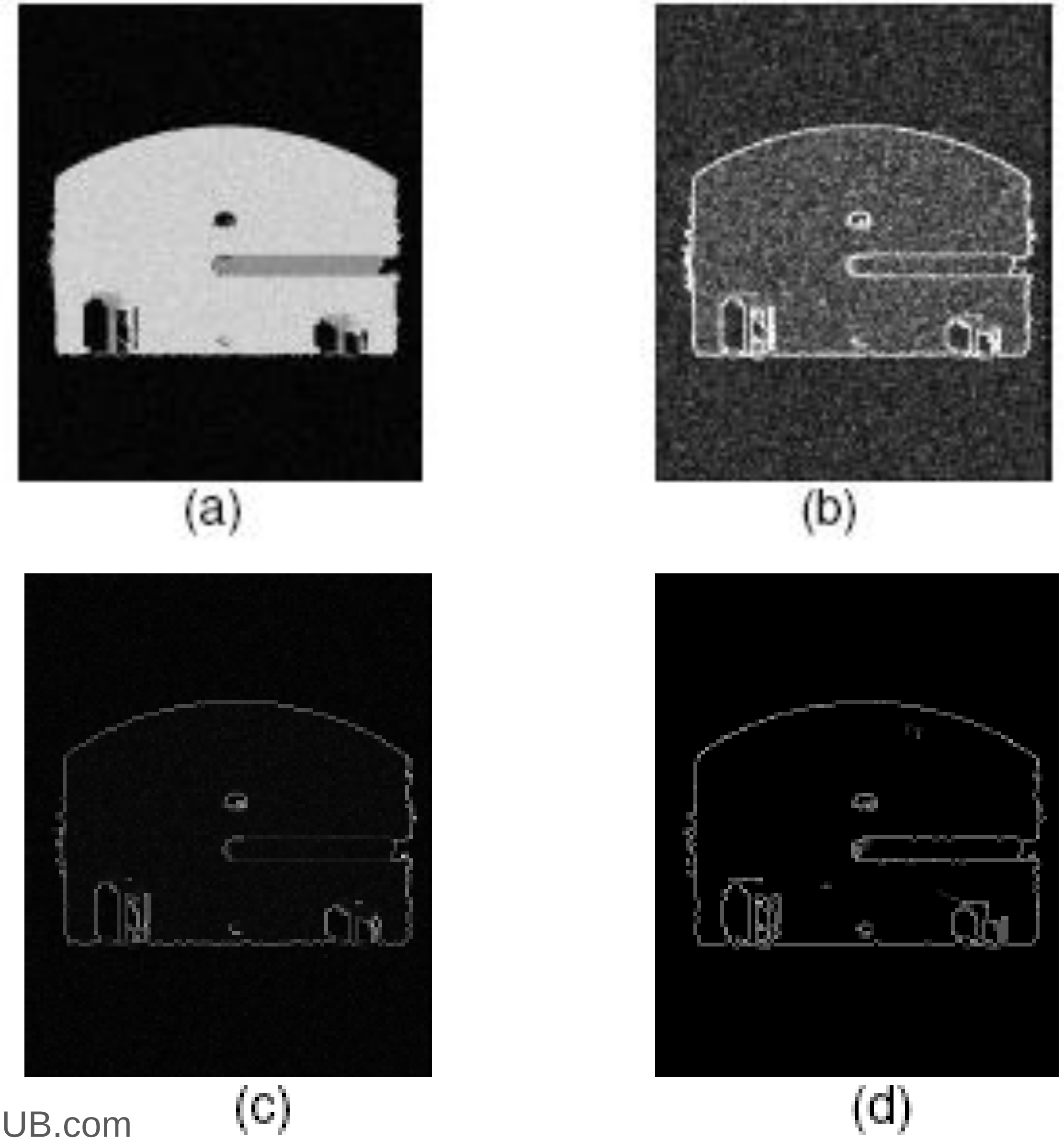
Original (a)

Canny (b)

Roberts (c)

Sobel (e)

Edge Detection Summery



(a) Original Image with Noise
(b) Sobel
(c) Robert
(d) Canny

Edge Detection Summery

- SOFT COMPUTING APPROACHES
  - *Fuzzy based Approach*
  - *Genetic Algorithm Approach*
  - *Neural Network Approach*

- Soft computing approaches, are applied on a real life example image of nature scene

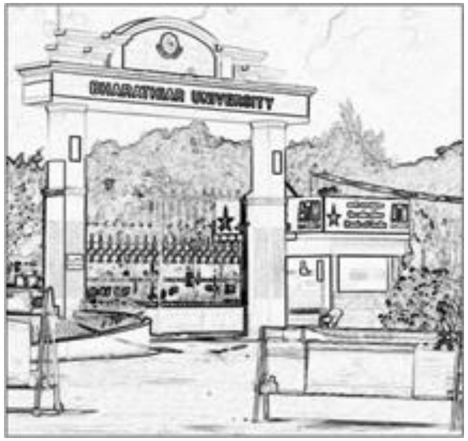
Edge Detection Summery

Original

Roberts

Sobel

Fuzzy



Genetics

Neural Network

Edge Linking and Boundary Detection

- Ideally, edge detection should produce the sets of pixels that form the edges in the image
- Practically, this is not always correct !
  - Some detected pixels correspond to noise
  - Some of the edges pixels are not detected due to breaks in the edges
    - Nonuniform illumination
    - Efficiency of the detection method
- It is a common practice to follow the detection process by linking process to fill in the breaks and to remove non-edge pixels
- Three common approaches
  - Local - Assumes knowledge about edge points in a local region
  - Regional – Requires that the points on the boundary be known
  - Global – Operates on the entire edge image

Local Edge Linking

- It is based on analyzing a small neighborhood about every candidate edge pixel in the edge image. All pixels that are similar according to some criteria are linked to form an edge using these similar pixels
- Two principle properties used to measure similarity of edge pixels are the magnitude and direction of gradient
  - A pixel (s,t) in a small neighborhood S_{xy} around pixel (x,y) is similar to pixel (x,y) by gradient magnitude if

$$|M(s,t) - M(x,y)| \le E$$

  - A pixel (s,t) in a small neighborhood S_{xy} around pixel (x,y) is similar to pixel (x,y) by gradient direction if

$$|\alpha(s,t) - \alpha(x,y)| \le A$$

  - The pixel (s,t) in Sxy is linked to pixel (x,y) if the previous two conditions are satisfied

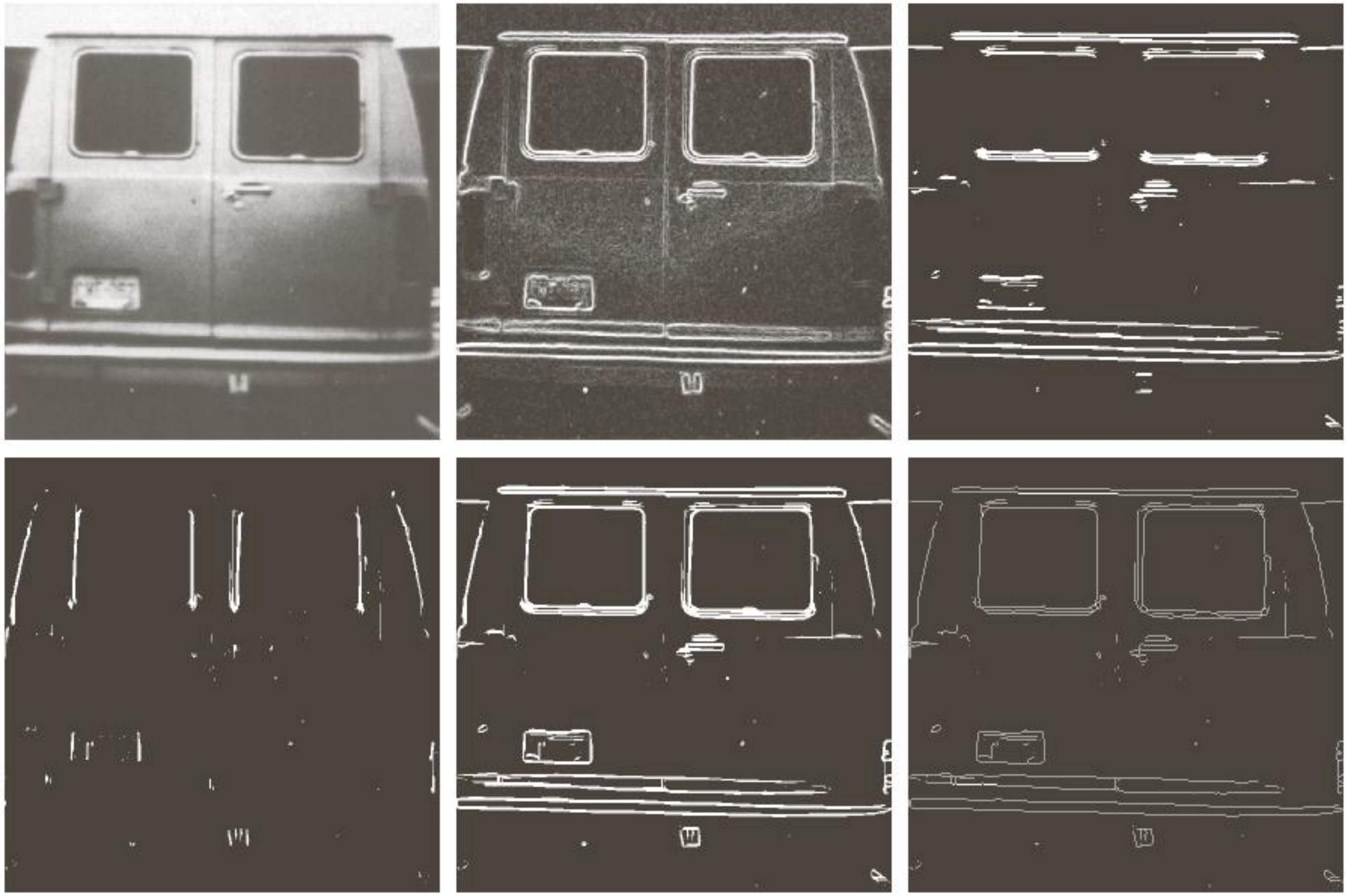
Local Edge Linking

- The previous approach is expensive as it requires the investigation of each edge pixel
- Alternatively, we simplify by following these steps
  1) Compute the gradient magnitude and angle arrays, M(x,y) and α(x,y) of the edge image f(x,y)
  2) Form the binary image g(x,y) by using

$$g(x,y) = \begin{cases} 1\text{ , if } M(x,y) > T_M \text{ and } \alpha(x,y) = A \pm T_A \\ 0\text{ , otherwise} \end{cases}$$

  3) Scan the rows of g(x,y) and fill (set to 1) all gaps (sets of 0s) in each row that don't exceed a specified length K. A gap is bounded by 1s from both ends. Processing is done on each raw individually
  4) To detect gaps in any other direction θ, rotate g(x,y) by θ and repeat the horizontal scanning in step 3. Once, done, rotate the result by -θ

Local Edge Linking



T_M = 30% of maximum gradient value

A = 90^o TA = 45^o

K = 25 pixels

a b c
d e f

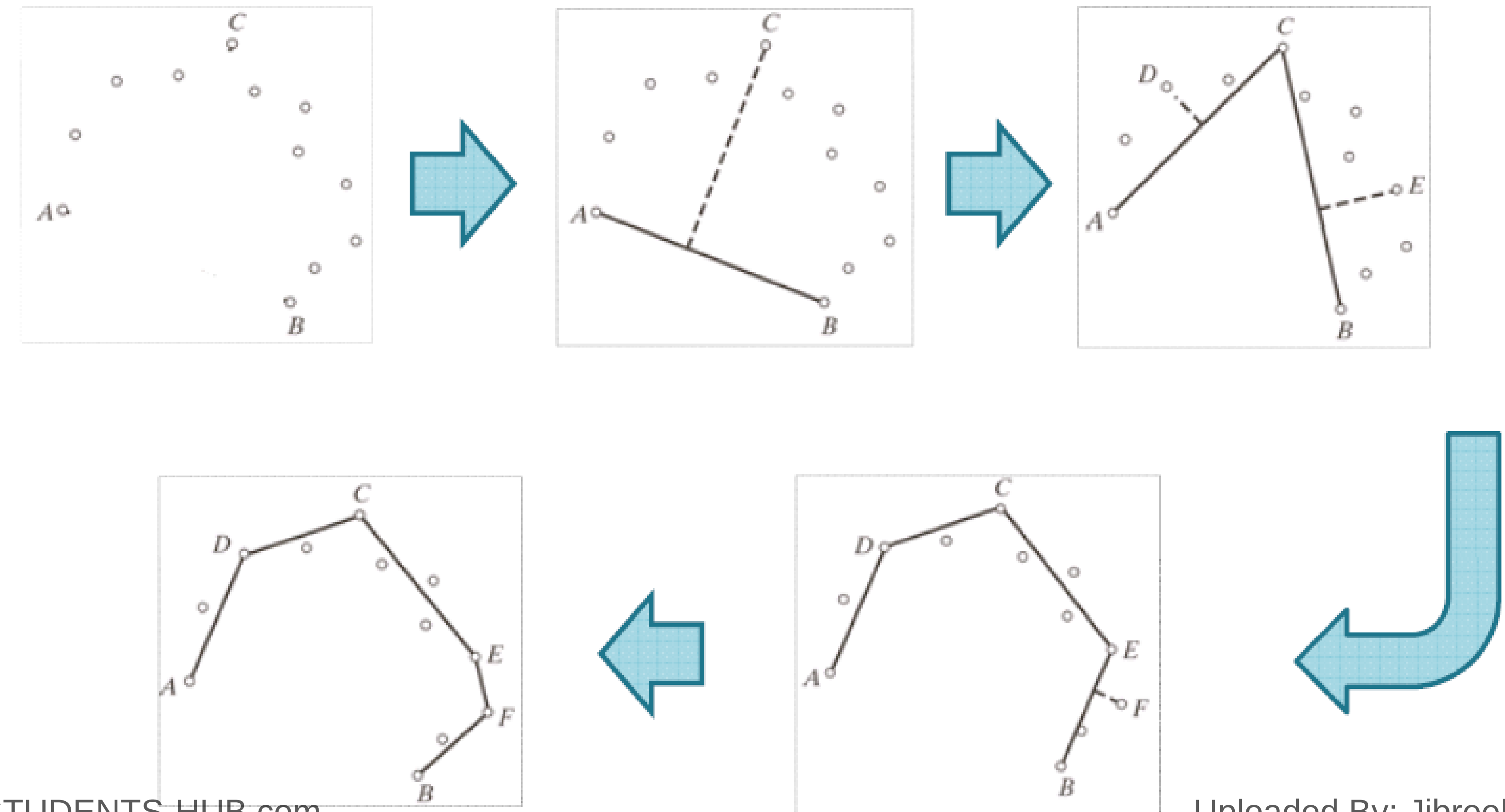
FIGURE 10.27 (a) A 534 × 566 image of the rear of a vehicle. (b) Gradient magnitude image. (c) Horizontally connected edge pixels. (d) Vertically connected edge pixels. (e) The logical OR of the two preceding images. (f) Final result obtained using morphological thinning. (Original image courtesy of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

Regional Edge Linking

- Assumes that the locations of regions of interest in the image are known or can be determined.
- In other words, the regional membership of pixels in the corresponding edge image is known
- A popular approach in this category is functional approximation by curve-fitting the known points (assume that the desired edge/boundary is approximated by some mathematical expression)
- Another approach is polygonal approximations. They capture the essential shape features of a region while keeping the representation of the boundary simple

Regional Edge Linking

- **Polygonal Approximation**
  - Given a set of points that represent an open curve with end points A and B which are assumed to be vertices of a the polygon

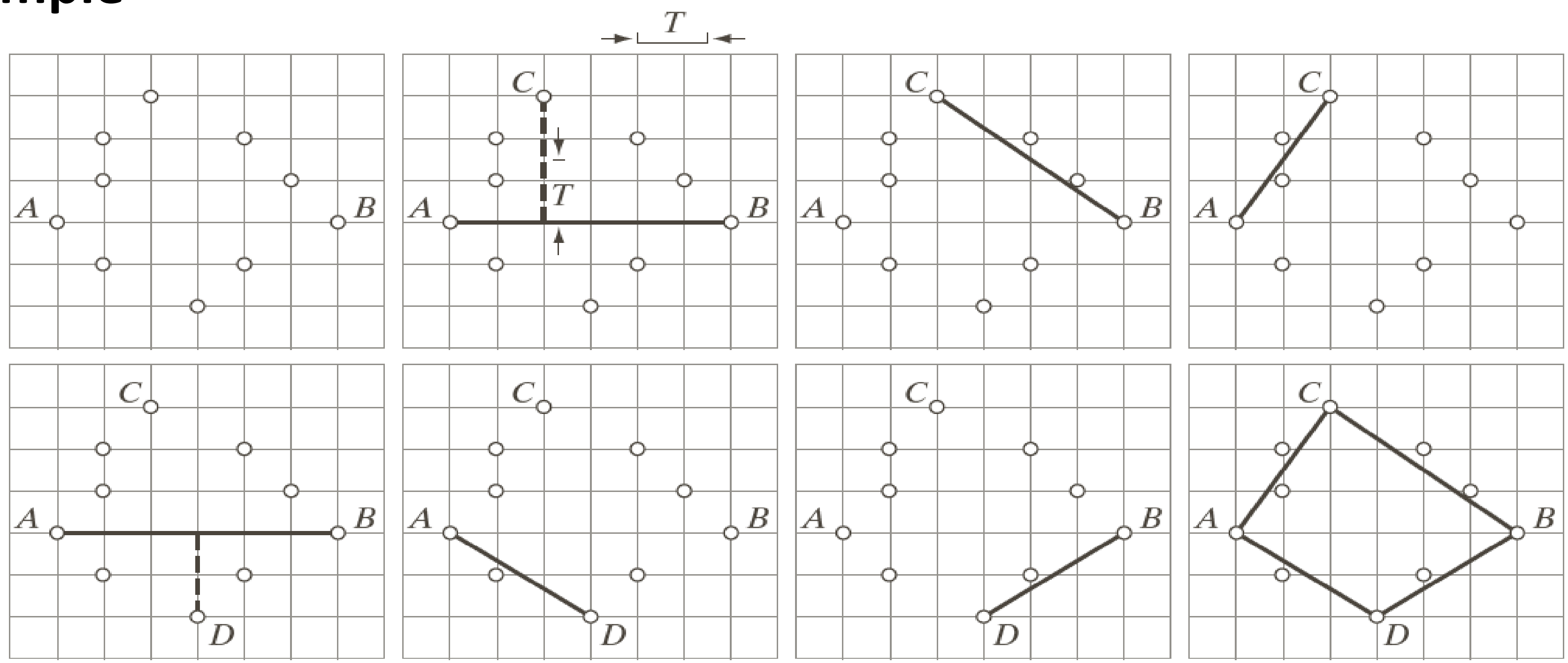


Regional Edge Linking

- **Algorithm: for a set of ordered distinct points P in a binary image, finding a polygonal fit can be done by**
  1) Specify two starting points, A and B
  2) Specify a threshold T and two empty stacks, OPEN and CLOSED
  3) If the points in P correspond to a closed curve, then put A into OPEN and B into OPEN and CLOSED. If the points in P correspond to open curve, put A into OPEN and B into CLOSED
  4) Compute the parameters of the line passing through the last vertex in Closed and the last vertex in OPEN
  5) Compute the distances from the line in Step 4 to points in P whose sequence places them between the vertices in Step 4. Select the point V_{max} that has maximum distance D_{max}.
  6) If $D_{max} > T$, then place V_{max} at the end of the OPEN stack as a new vertex. Go to Step 4
  7) If $D_{max} <= T$, then remove the last vertex from OPEN and insert it as the last vertex in closed
  8) If OPEN is not empty, go to Step 4, else exit. The vertices in CLOSED are the vertices of the polygonal fit to points in P

Regional Edge Linking

Example



CLOSED	OPEN	Curve segment processed	Vertex generated
B	*B*, *A*	—	*A*, *B*
B	*B*, *A*	(*BA*)	*C*
B	*B*, *A*, *C*	(*BC*)	—
B, *C*	*B*, *A*	(*CA*)	—
B, *C*, *A*	*B*	(*AB*)	*D*
B, *C*, *A*	*B*, *D*	(*AD*)	—
B, *C*, *A*, *D*	*B*	(*DB*)	—
B, *C*, *A*, *D*, *B*	Empty	—	—

Regional Edge Linking

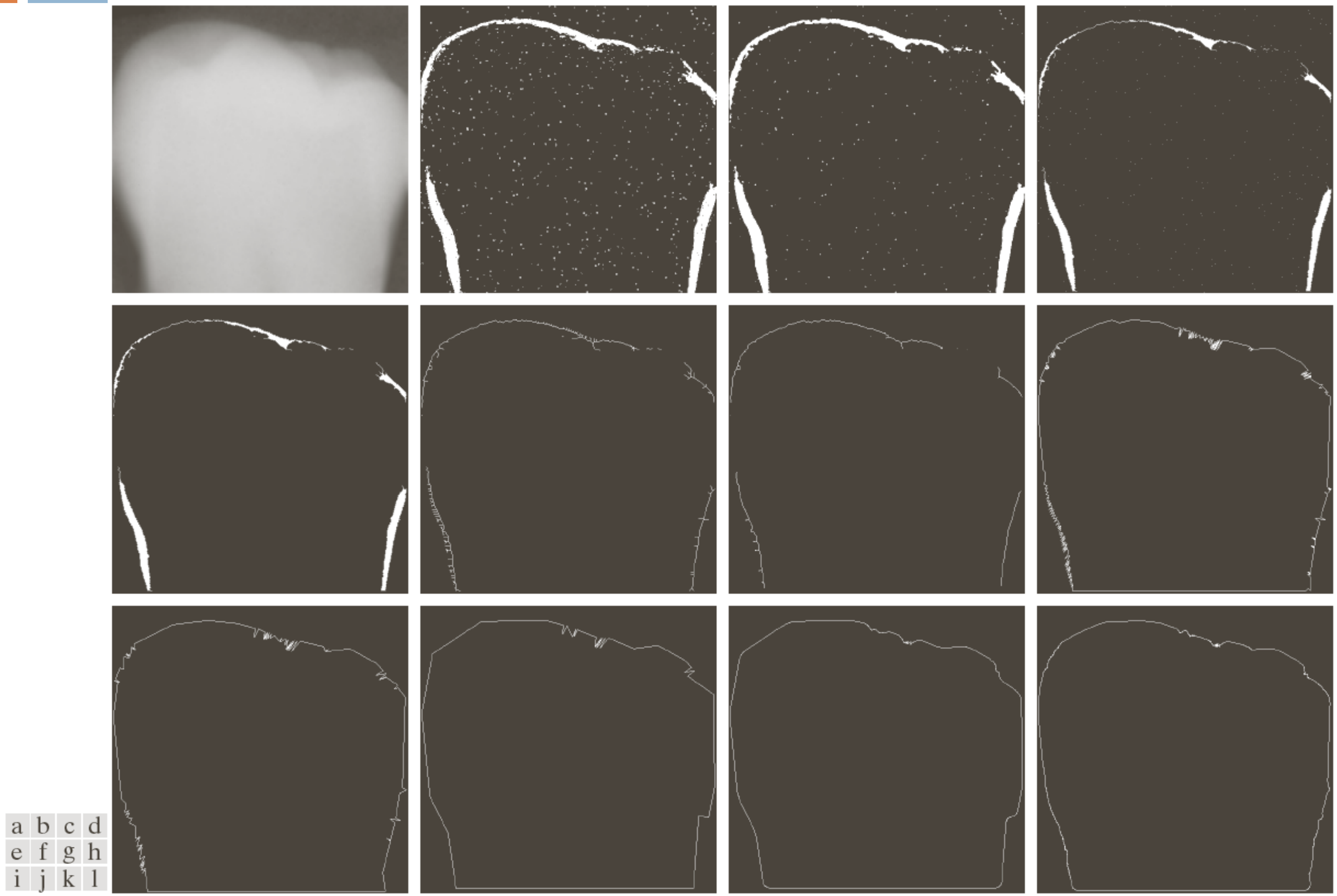


FIGURE 10.30 (a) A 550 × 566 X-ray image of a human tooth. (b) Gradient image. (c) Result of majority filtering. (d) Result of morphological shrinking. (e) Result of morphological cleaning. (f) Skeleton. (g) Spur reduction. (h)–(j) Polygonal fit using thresholds of approximately 0.5%, 1%, and 2% of image width ($T = 3$, 6, and 12). (k) Boundary in (j) smoothed with a 1-D averaging filter of size 1 × 31 (approximately 5% of image width). (l) Boundary in (h) smoothed with the same filter.

Hough Transform

- The input is the edge image with all edge pixels being candidates for linking

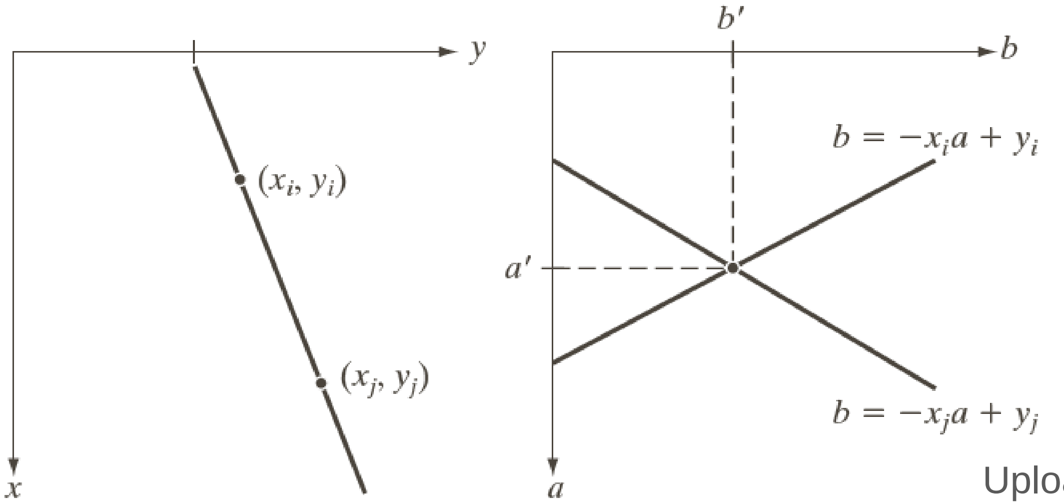
- Acceptance or elimination of pixels is based on some global properties, usually a specified shape such as a line or circle

- Given n edge points, suppose we want to find subsets of these points that lie on straight lines. A simple approach is find all possible lines between every pair of points then pick those lines that are close to particular lines

- However, this is computationally expensive !! It requires finding $n(n-1)/2$ lines and performing $n^2(n-1)/2$ comparisons !

Hough Transform

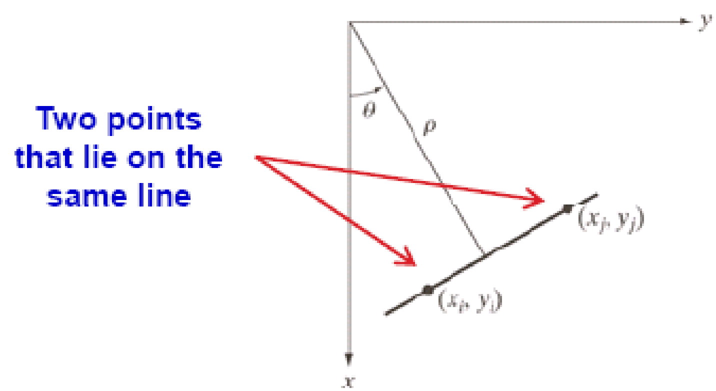
- Consider a point (x_i,y_i) in the xy-plane and the general equation of a straight line $y_i = a x_i + b$
- Infinite number of lines pass through this point with different values of a and b
- If we rewrite the line equation such that $b = - a x_i + y_i$ and consider the ab-plane (the parameter space), then we have the line equation for a fixed pair (x_i,y_i)
- If we consider another point (x_j,y_j) that lie on the same line as (x_i,y_i), then in the ab-space the two lines corresponding to these two point will intersect at (a',b') and so do all the lines for the points on the line in the xy-plane

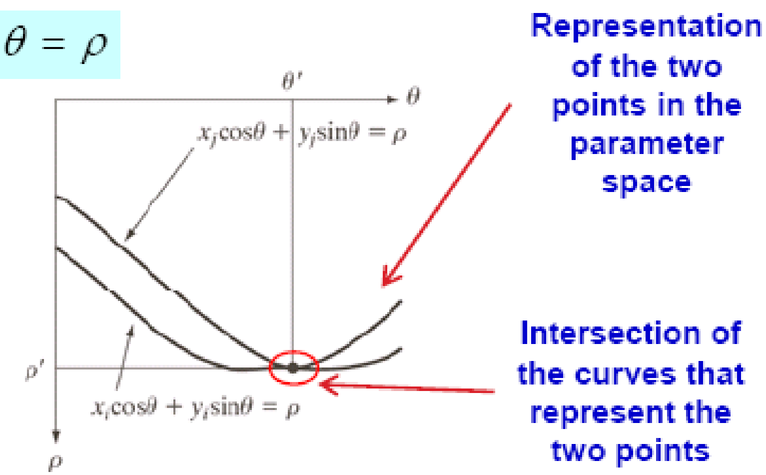


Hough Transform

- In order to find the principle lines in the image, we can compute all the parameter-space lines and then select those points that have large number of intersections
- A major difficulty in this approach is that the slope of the line approaches infinity for vertical and near vertical lines
- One way around is to use the normal representation of the line in the xy-plane

$$x_i \cos\theta + y_i \sin\theta = \rho$$



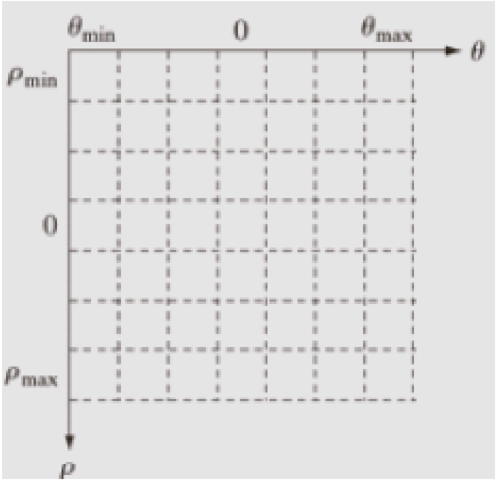


- In this representation, a vertical line is represented in the parameter-space by θ = 90 and p being the y-intercept.
- Points that lie on the same line have their normal representation intersect at (θ',p'). As the number of points that lie on the same line increases, we will have more curve crossing in the parameter space at the point (θ',p')

Hough Transform

- **Procedure for finding principle lines in an image**

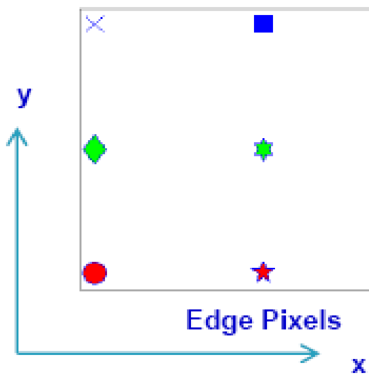
1. Subdivide the pθ parameter space into so-called accumulator cells and initialize them to zero. The range of each of the parameters is specified by
    a. $-90 \le \theta \le 90$
    b. $-D \le p \le D$ with D being the maximum diagonal length in the image
2. For every edge/line candidate pixel (x_i, y_i) in the edge image, we substitute every possible value of θ in $p = x_i \cos\theta + y_i \sin\theta$. Round the values of p to the allowed subdivision values on the p-axis
3. If θ_m results in p_n, then increment the count in cell (m,n) by one, i.e. $A(m,n) = A(m,n)+1$
4. Select accumulator cells with counts greater than certain threshold and find the corresponding θ and for these cells
5. Draw the lines in the xy-plane for each pair of pair of θ and p found in Step 4

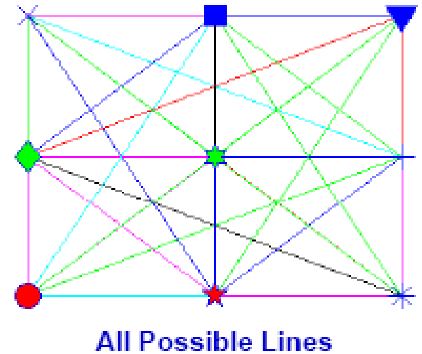


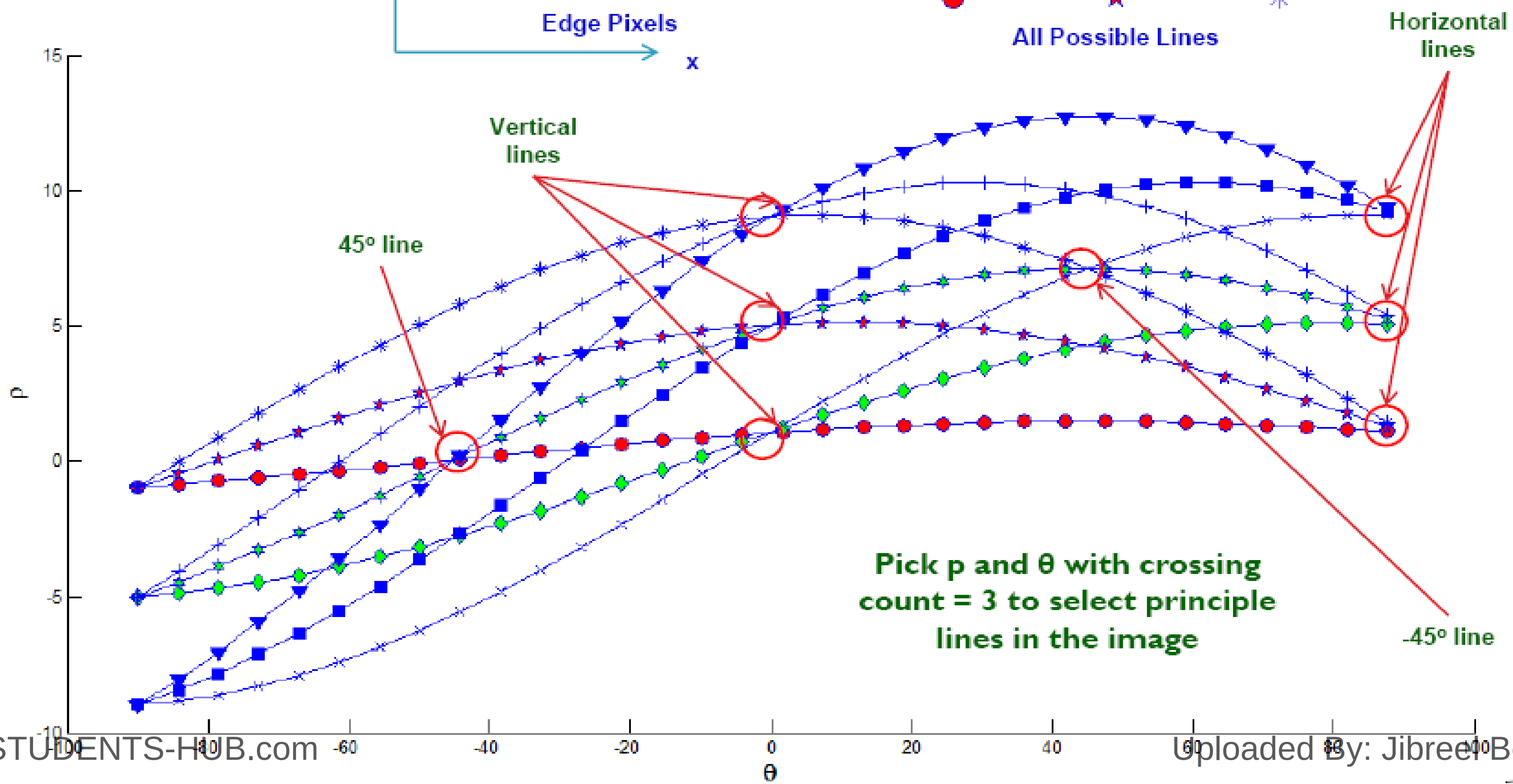
Subdividing the parameter space to generate accumulator cells

Hough Transform

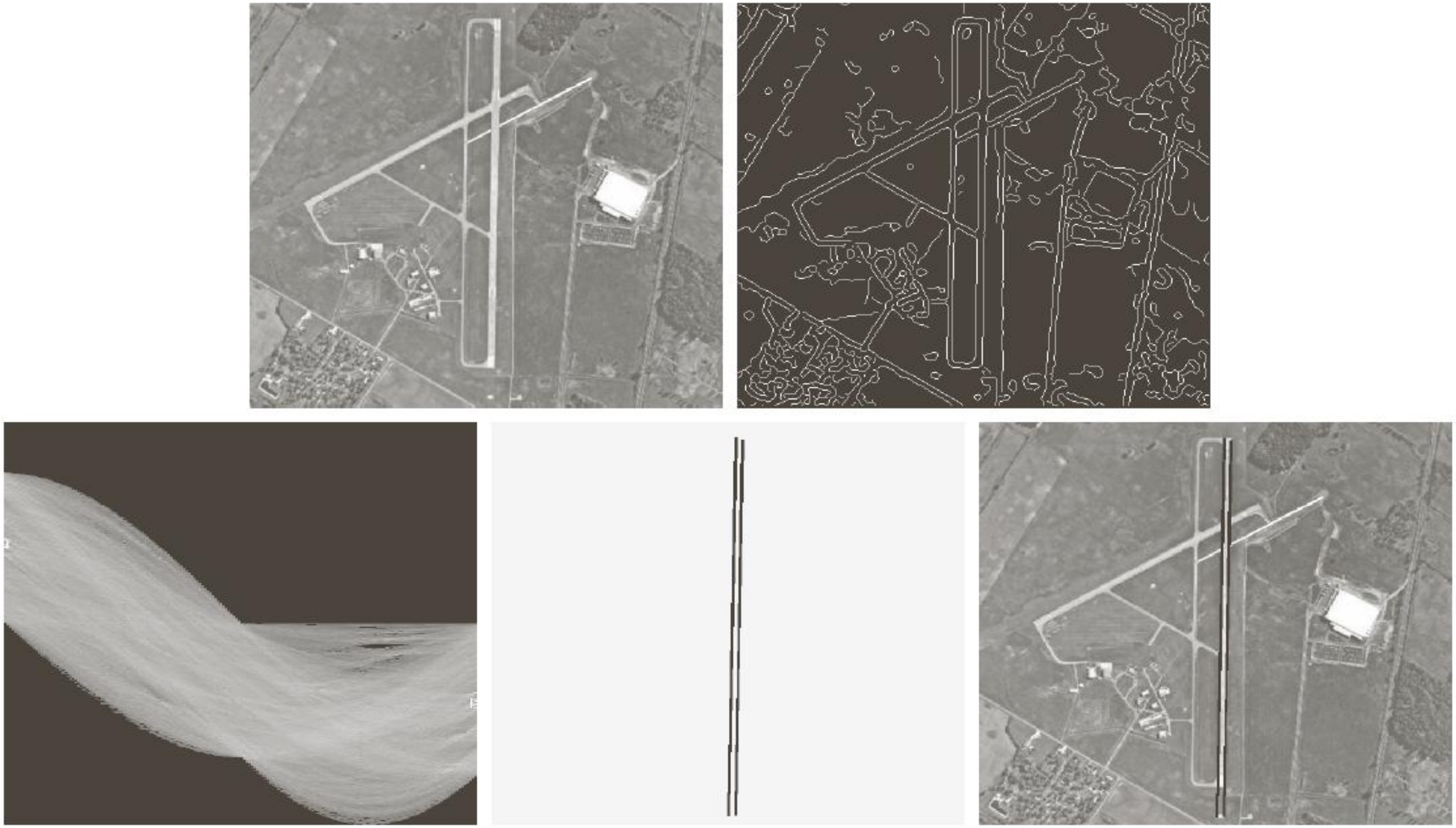
- **Example**







Hough Transform



a b
c d e

FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

Thresholding

- Segmentation can be achieved by
  - Detecting discontinuities (edge-based segmentation)
    - Assumes that the boundaries between the objects themselves and the background are sufficiently different
  - Detecting similarities
    - Assumes that the objects to be segmented are similar according to some criteria
    - Examples
      - Thresholding
      - Region based segmentation
      - Watersheds

Thresholding

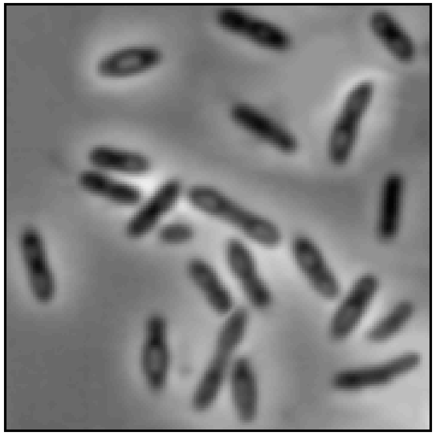
- Thresholding is based on partitioning the image into regions based on intensity values and/or properties of these values

- It is simple and computationally cheap

- In its very simple form, thresholding attempts to find an intensity value that separates the objects of interest from remaining of the image

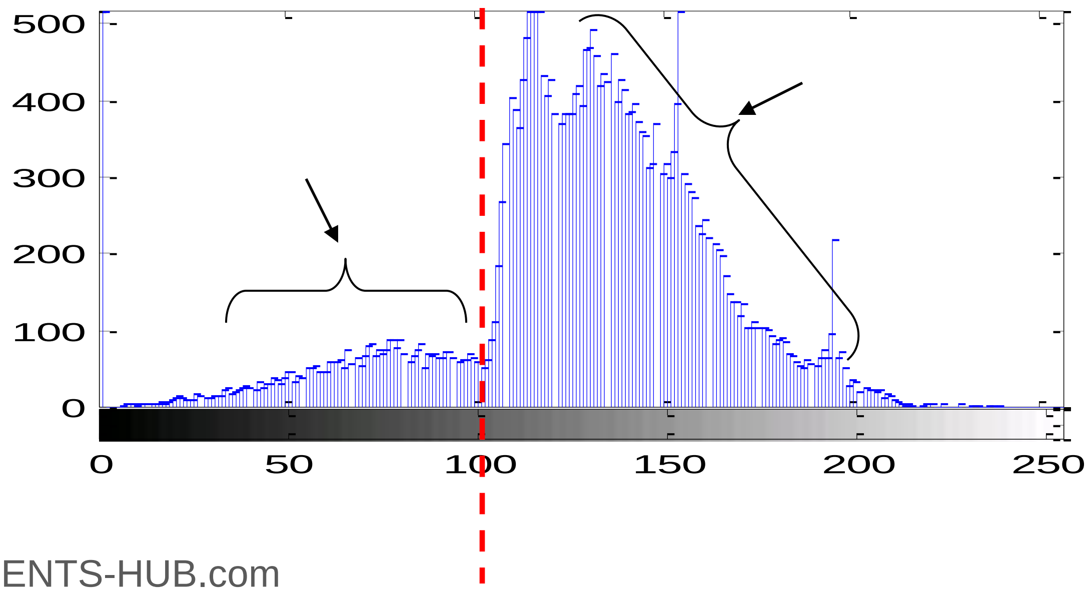
- Usually this is performed by investigating the image histogram which may contain modes that represent the object(s) and the background

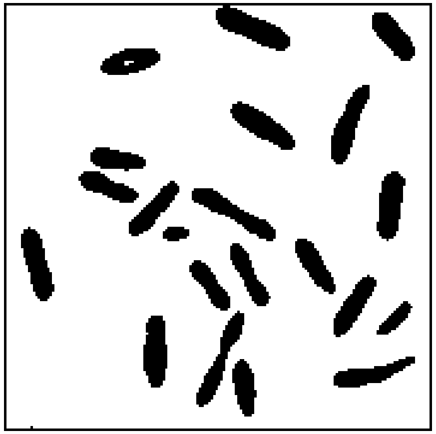
Thresholding



$$g(x,y)=\begin{cases}1 & f(x,y) > T\\ 0 & f(x,y) < T\end{cases}$$

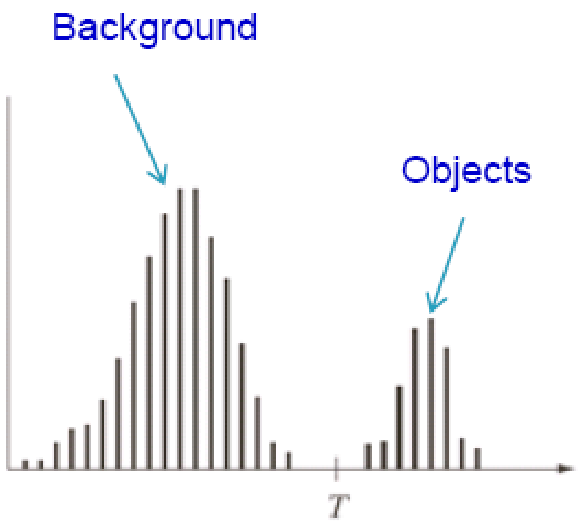
T = 102





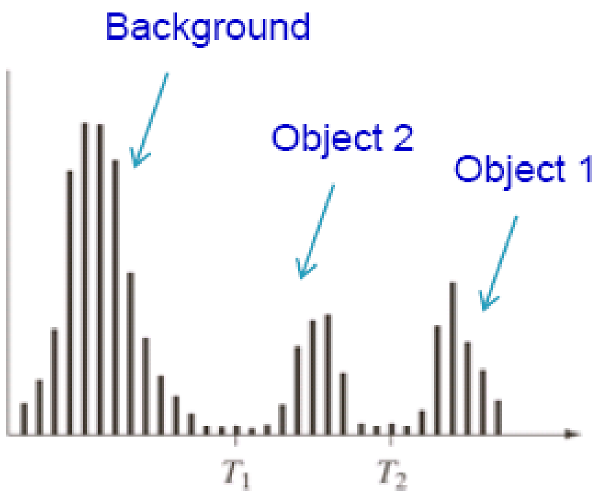
After thresholding

Thresholding



Histogram of an image with light objects and dark background . Segmenting the objects can be achieved by specifying T such that

$$g(x,y)=\begin{cases} 1, & f(x,y) > T \\ 0, & f(x,y) \leq T \end{cases}$$

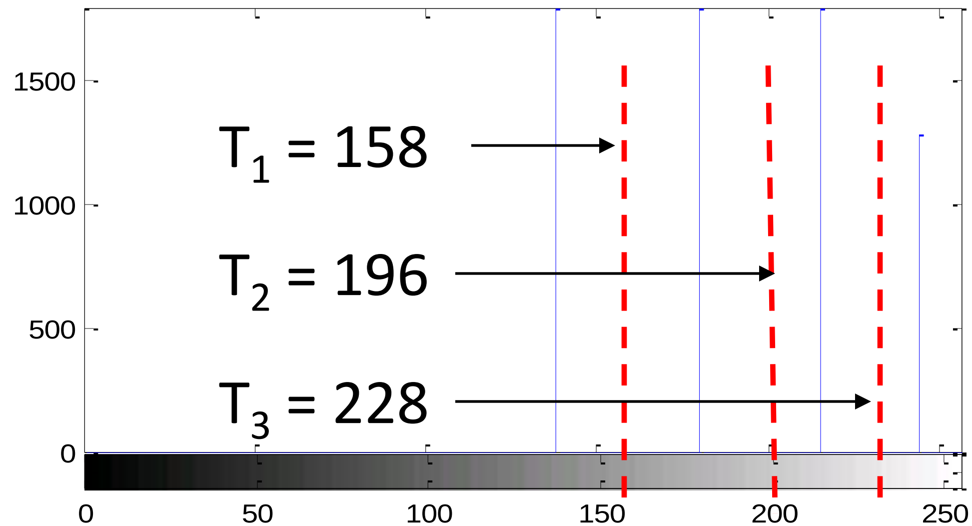


More challenging situation where we need to specify two thresholds in order to segment the object from background (Multiple Thresholding)

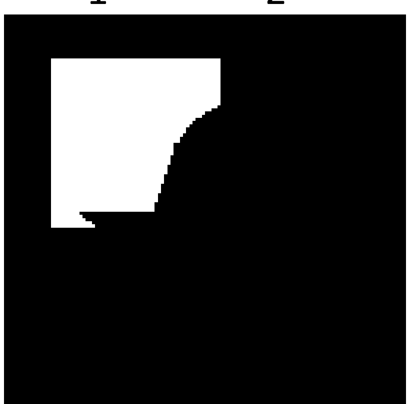
$$g(x,y)=\begin{cases} a, & f(x,y) > T_2 \\ b, & T_1 < f(x,y) \leq T_2 \\ c, & f(x,y) \leq T_1 \end{cases}$$

Thresholding

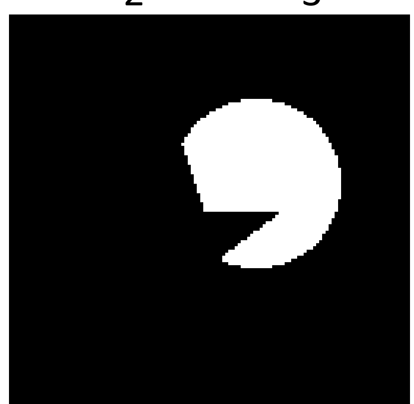
Histogram



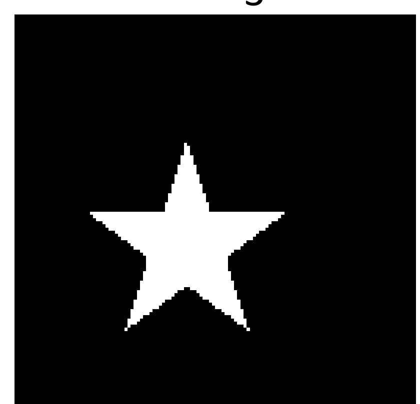
$T_1 < P < T_2$



$T_2 < P < T_3$



$P > T_3$

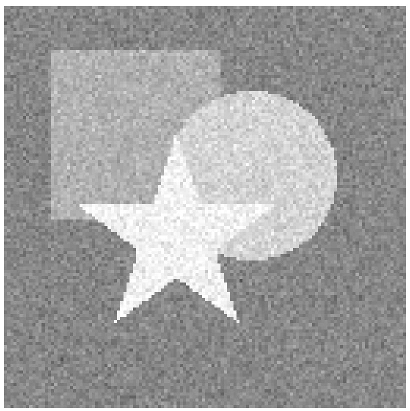


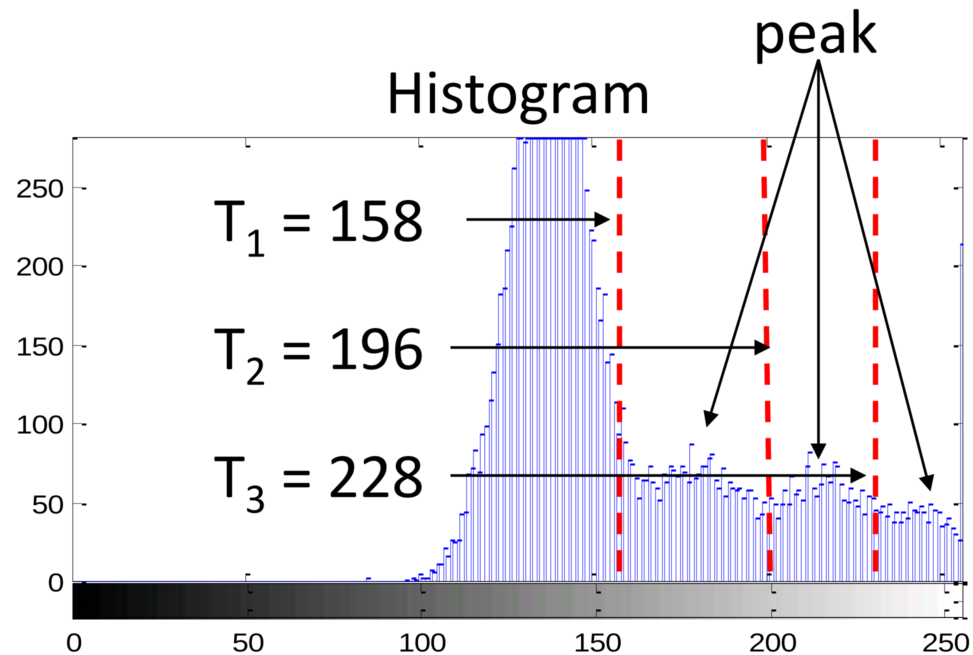
Thresholding

- **Thresholding Approaches**
  - Global – specify one or more fixed threshold values based on histogram shape
  - Local or regional thresholding – threshold values T at any point (x,y) is dependent on the properties of neighborhood around (x,y)
  - Dynamic or adaptive – threshold value depends on neighborhood properties and spatial coordinates as well
- **The success of intensity thresholding depends on**
  - The separation between the peaks in the histogram
  - The noise content in the image (the histogram modes broaden as noise increases)
  - The relative size of objects and background
  - The uniformity of illumination source
  - The uniformity of the reflectance properties of the image

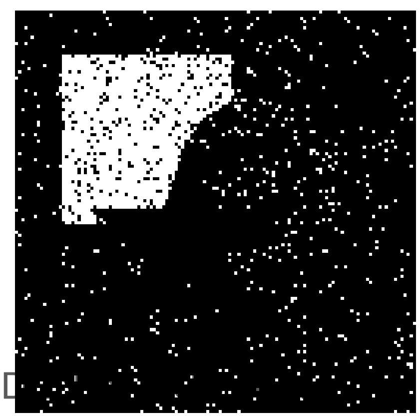
Thresholding - Effect of Noise

Image degraded by Gaussian noise (s =12)

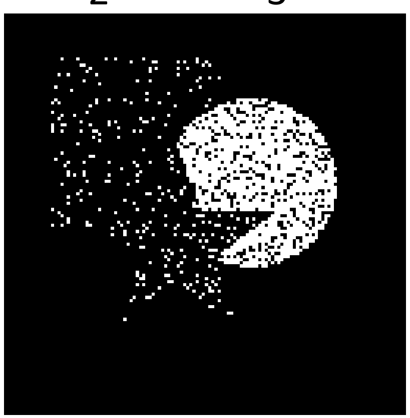




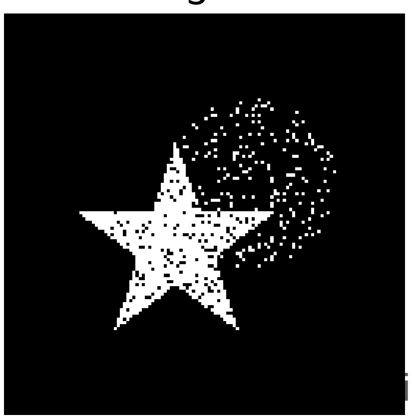
$T_1 < P < T_2$



$T_2 < P < T_3$

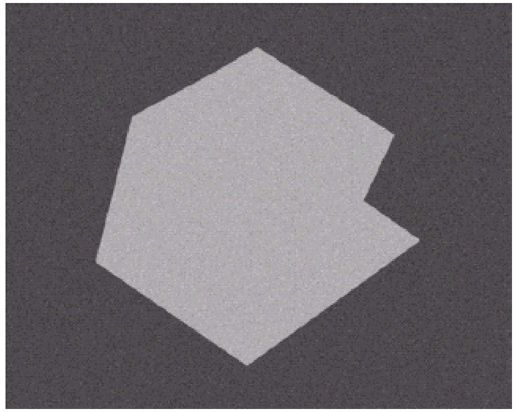


$P > T_3$



Thresholding - Effect of Illumination

Reflectance Function $r(x,y)$



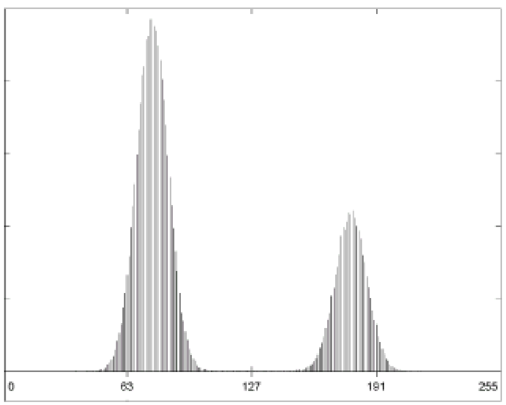
An image can be expressed as

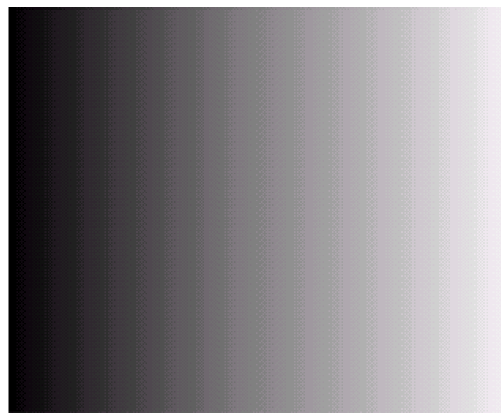
$$f(x,y) = i(x,y)r(x,y)$$

$i(x,y)$ = illumination component

$r(x,y)$ = reflectance component

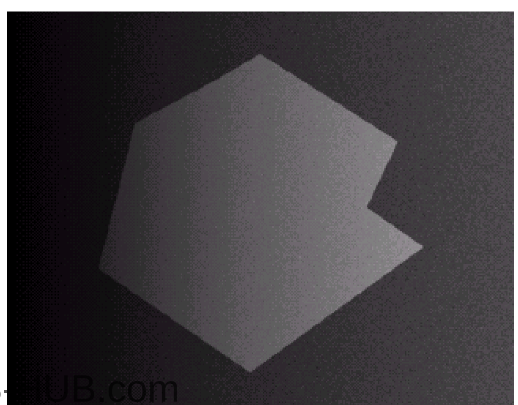
Histogram





Illumination Function $i(x,y)$

$f(x,y)$



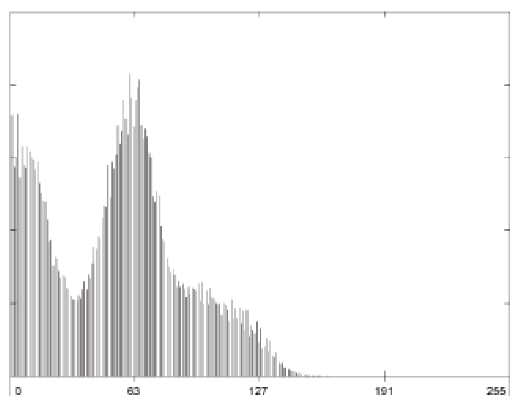
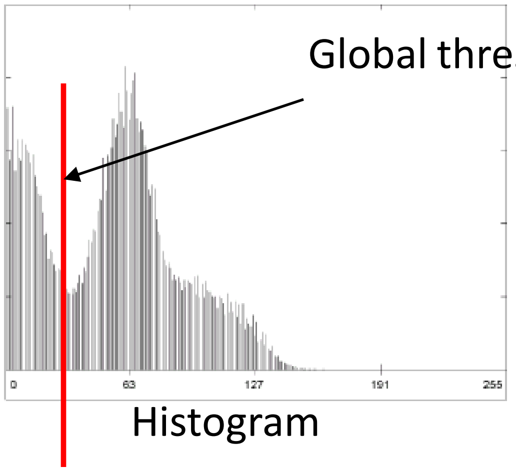


Image histogram

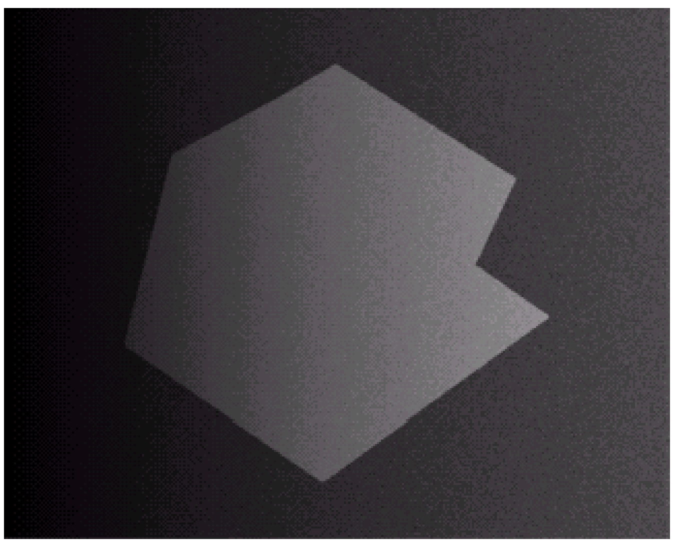
Thresholding - Effect of Illumination

Global threshold level



Histogram

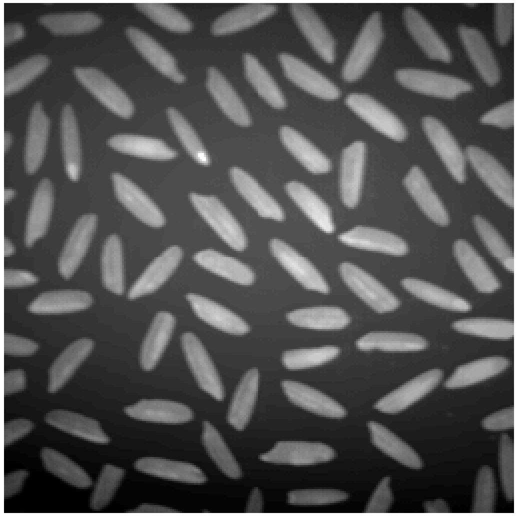
Global thresholding of nonuniform illumination image can cause huge errors!

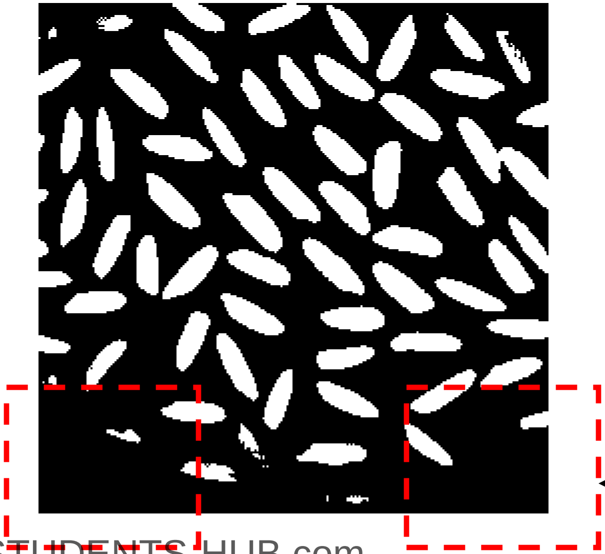


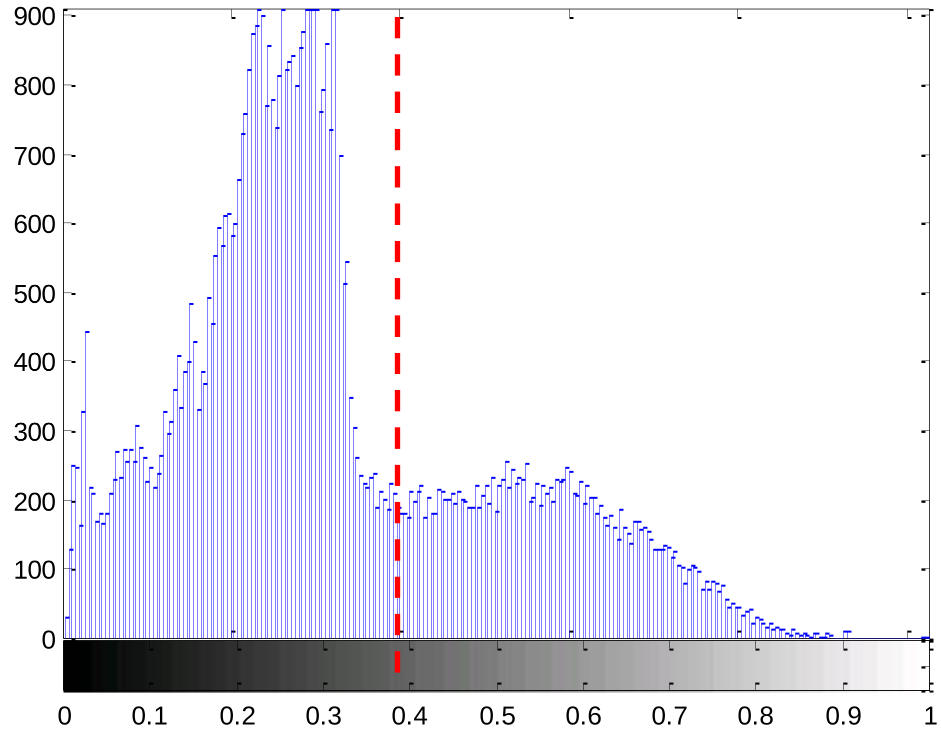
Nonuniform illumination image

Global thresholding result

Thresholding - Effect of Illumination







T=0.4

Error

Basic Global Thresholding

- When the intensity distribution of objects and background is sufficiently enough, it is possible to use a single global threshold value; a very common situation in practice
- To find such global threshold **automatically**

1. Select an initial estimate for the global threshold value, T
2. Segment the image using T to produce two groups of pixels; G1 containing pixels with intensity values > T and G2 with intensity values ≤ T
3. Compute the mean intensity value for each group, m_1 and m_2, and define the new threshold by

$$T = (m_1 + m_2)/2$$

4. Repeat steps 2 and 3 until the change in T is smaller than a specified value ΔT

Basic Global Thresholding

- **Example**

Original

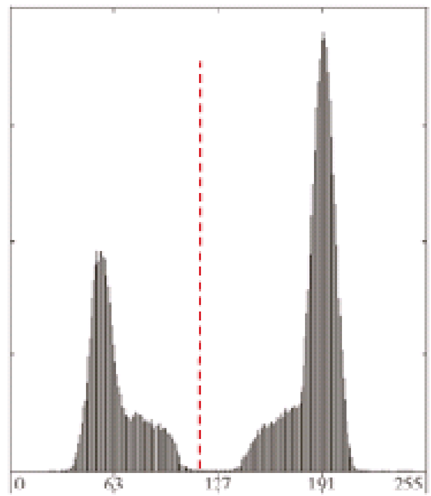
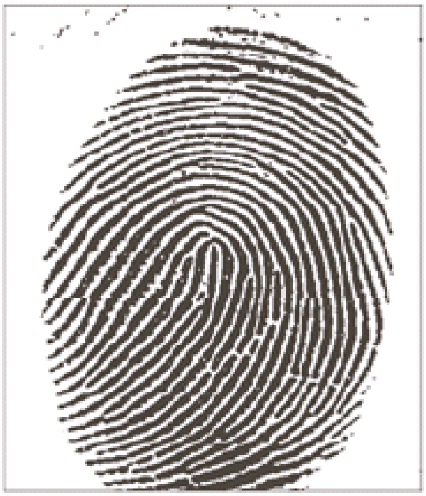


Image Histogram



Segmentation Result

Initial threshold = mean intensity of image

Optimal threshold = 125.4

Three iterations

Optimal Global Thresholding

- Views the thresholding problem as a statistical-decision problem with the objective minimizing the average error of miss-classifying pixels to two or more groups
- The method requires the knowledge of
  - The probability density function of the intensity levels of each class
  - The probability that each class occurs in a given applications
- Such requirements are usually hard to obtain in practice !
- Alternatively, we consider optimal global thresholding using Otsu's method
  - The method is optimal in the sense that maximizes the between-class variance
  - It is based on computations performed on the histogram of an image; an easily obtainable 1-D array

Otsu's Optimal Global Thresholding

- Let {0,1,2,..L-1} denote L distinct levels in MxN image . with normalized histogram whose components are $p_i = n_i/MN$
- Suppose we threshold the image with a threshold $T(k) = k$, $0<k<L-1$, to produce two classes C_1 and C_2, where C_1 is the set of pixels with intensity levels in [0,k] and C_2 is the set of pixels with intensity levels in [k+1,L-1]
- Based on this threshold, the probability that a pixel belongs to C_1 (the probability of class C_1 occurring) is

$$P_1(k) = \sum_{i=0}^{k} p_i$$

and for C_2 is

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$$

Otsu's Optimal Global Thresholding

- The mean intensity value m_1 of pixels assigned to C_1 is

$$m_1(k) = \sum_{i=0}^{k} i \times P(i / C_1) = \sum_{i=0}^{k} i \times P(C_1 / i) P(i) / P(C_1) = \frac{1}{P_1(k)} \sum_{i=0}^{k} i \times p_i$$

- Similarly, the mean intensity m_2 value of pixels assigned to C_2 is

$$m_2(k) = \sum_{i=k+1}^{L-1} i \times P(i / C_2) = \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} i \times p_i$$

- The average intensity of the entire image is given by

$$m_G = \sum_{i=0}^{L-1} i \times p_i$$

- The cumulative mean up to level k is $m(k) = \sum_{i=0}^{k} i \times p_i$

- We can easily verify

$$P_1 m_1 + P_1 m_2 = m_G$$
$$P_1 + P_1 = 1$$

Otsu's Optimal Global Thresholding

- In order to evaluate the goodness of the threshold at level k we use the metric (separability measure)

$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2}$$

where σ_G^2 is the variance of the entire image and σ_B^2 is the between-class variance which is defined as

$$\begin{aligned}\sigma_B^2(k) &= P_1(k)(m_1(k) - m_G)^2 + P_2(k)(m_2(k) - m_G)^2 \\ &= P_1(k)P_2(k)(m_1(k) - m_2(k))^2 \\ &= \frac{(m_G P_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))}\end{aligned}$$

- For optimal segmentation, we need to find the intensity level k that maximizes η as it implies larger seperability between classes

$$\eta(k^*) = \max_{0 \le k \le L-1} \sigma_B^2(k)$$

Otsu's Optimal Global Thresholding

- **Notes**
  - As the difference between m_1 and m_2 increases, σ_B^2 increases to indicate better seperability
  - The definition of the metric η assumes that $\sigma_G^2 > 0$. The variance of an image could be zero if it contains one intensity level only
  - In order to find the optimal threshold k*, we have to compute σ_B^2 for all integer values of k and then pick the value that maximizes the between-class variance σ_B^2
  - Once the optimal threshold is found, we simply threshold the image into two classes using

$$g(x,y) = \begin{cases} 1, & f(x,y) > k^* \\ 0, & f(x,y) \le k^* \end{cases}$$

Optimal Global Thresholding

- **Summary of Otsu's Algorithm**
  - Compute the normalized histogram of the input image and denote its components by p_i , i = 1,2,…,L-1
  - Compute the cumulative sum $P_1(k)$, for k = 0,1,2,…,L-1 using

$$P_1(k) = \sum_{i=0}^{k} p_i$$

  - Compute the cumulative means, m(k), for k = 0,1,2,…,L-1 using

$$m(k) = \sum_{i=0}^{k} i \times p_i$$

  - Compute the global intensity mean m_G using

$$m_G = \sum_{i=0}^{L-1} i \times p_i$$

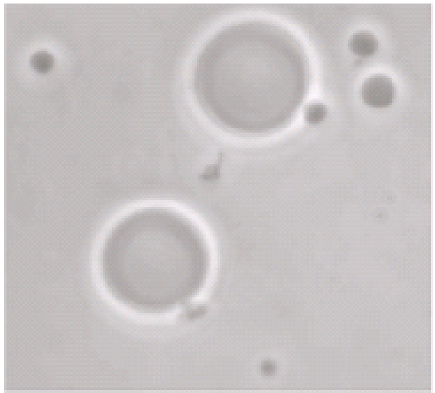
  - Compute the between-class variance $\sigma_B^2(k)$ for k = 0,1,2,…,L-1 using

$$\sigma_B^2(k) = \frac{(m_G P_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))}$$

  - Pick Otsu's threshold k* that gives the maximum value for $\sigma_B^2(k)$
  - Obtain the separability measure at k* using $\eta(k^*) = \frac{\sigma_B^2(k^*)}{\sigma_G^2}$

Optimal Global Thresholding

- **Example:** segmentation of the molecules from background



Polymersome cells image

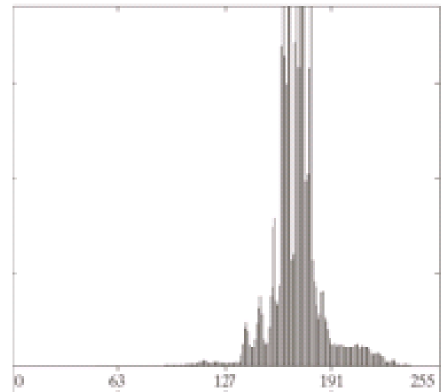


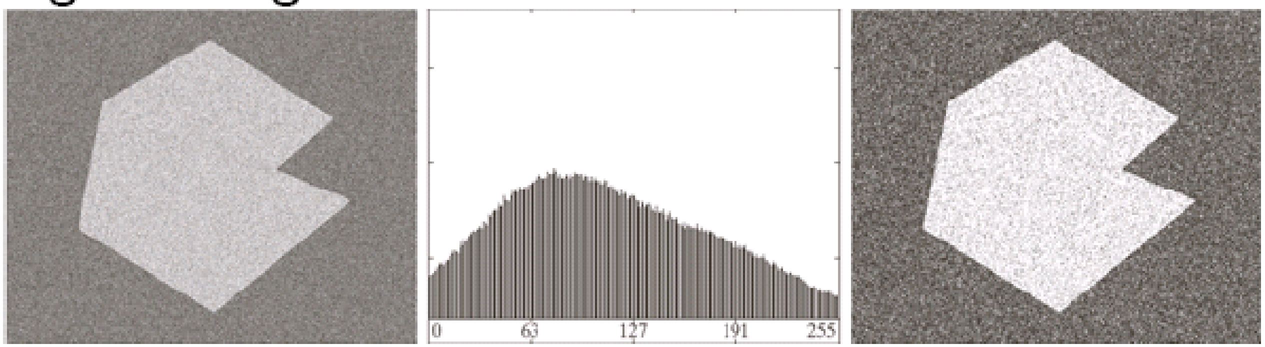
Image Histogram

Segmentation using basic global thresholding T* = 169

Segmentation using Otsu's global thresholding k* = 181

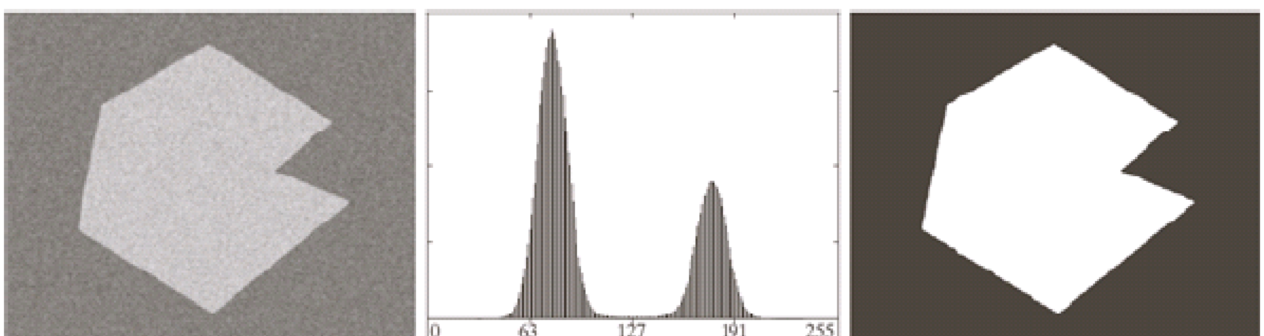
Enhancing Global Thresholding by Smoothing

- It was noted earlier that the presence of noise highly affect the result of thresholding
- One approach to improve thresholding result is to smooth the original image



Segmentation Fails

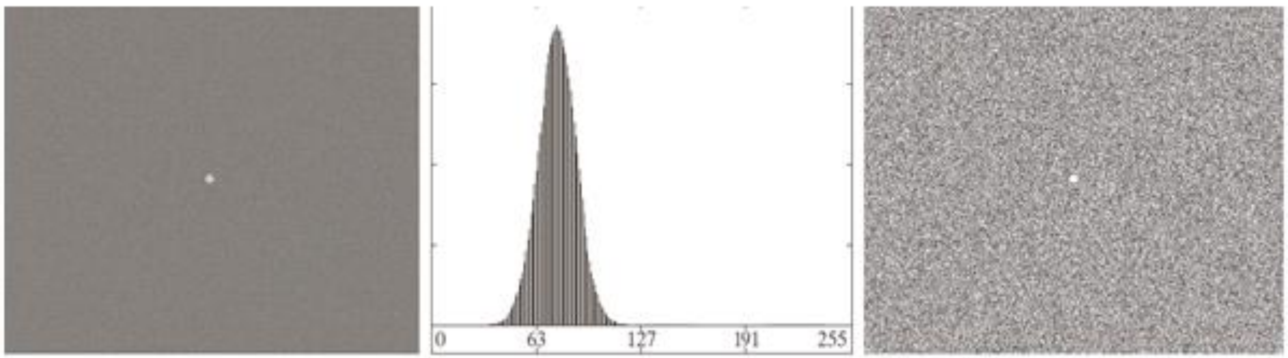
Noisy image, its histogram, and Otsu's segmentation result



Noisy image after smoothing by 5x5mask, its histogram, and Otsu's segmentation result

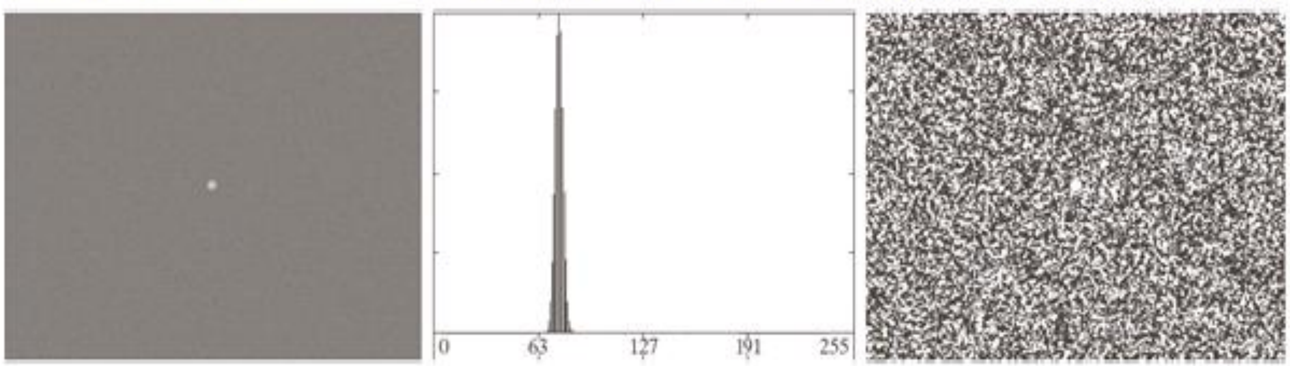
Effect of Relative Object Size on Thresholding

- As the size of the object decreases, its contribution to the image histogram decrease and it becomes harder to segment out



Segmentation Fails

Noisy image, its histogram, and Otsu's segmentation result



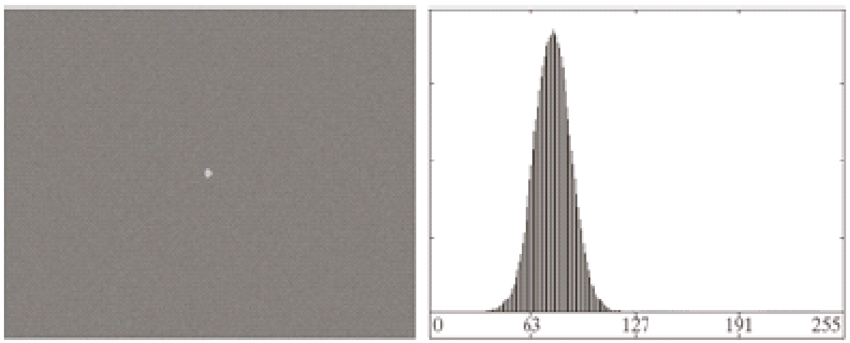
Segmentation Fails

Noisy image after smoothing by 5x5mask, its histogram, and Otsu's segmentation result

Effect of Relative Object Size on Thresholding

- Global thresholding can be improved by incorporating edge information, especially when segmenting unimodal histograms

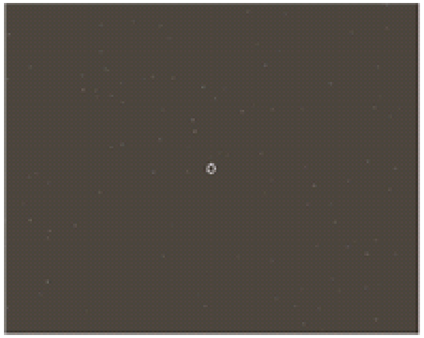
1



Noisy image with small object and its histogram

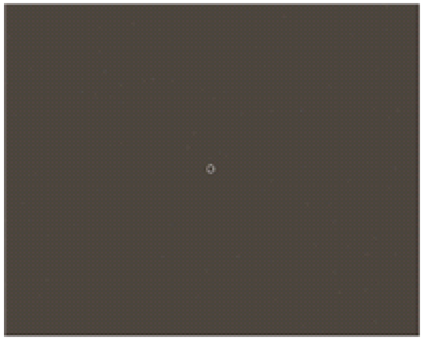
Compute the gradient magnitude image and threshold it by T

2

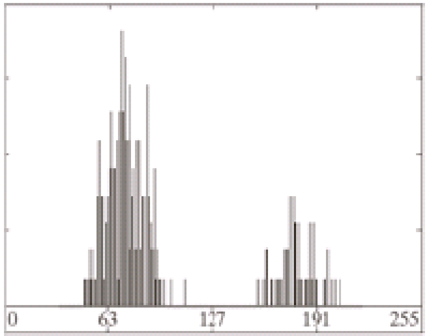


Thresholded Gradient Image
T = 99.7%
(binary image)

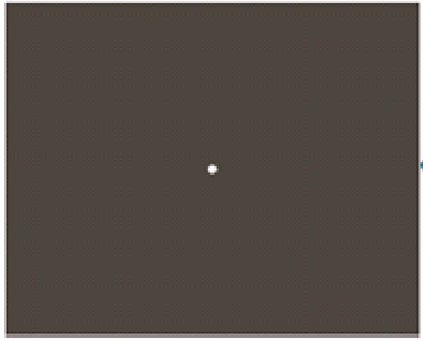
3



Logical AND between 1 and 2

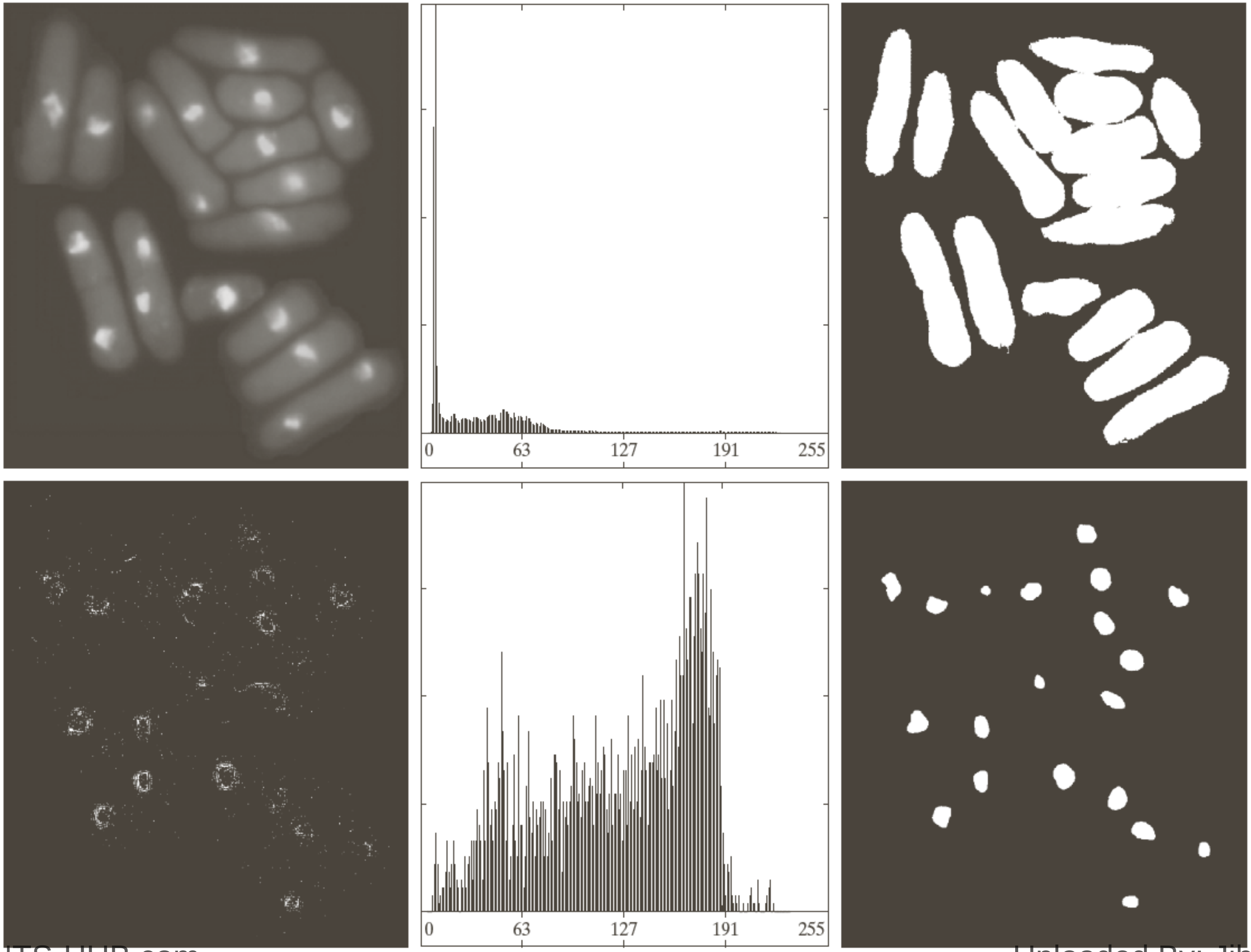


Histogram of 3



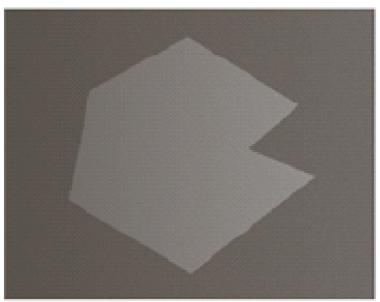
Segmentation result using Otsu's threshold computed from 3
($k^* = 134$)

Effect of Relative Object Size on Thresholding

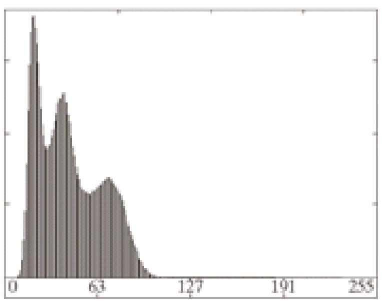


Variable Thresholding

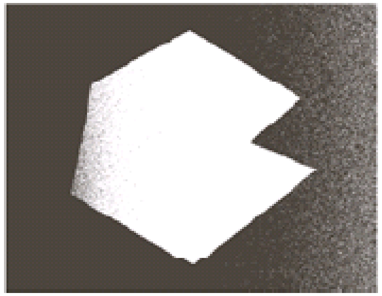
- **Variable Thresholding Based on Image Partitioning**
  - Subdivide the image into nonoverlapping rectangles to compensate in non-uniformities in illumination and/or reflectance. Threshold each subimage separately
  - The method works well when the objects and the background occupy regions of reasonably comparable size



Noisy image with nonuniform illumination



Histogram



Otsu's segmentation result

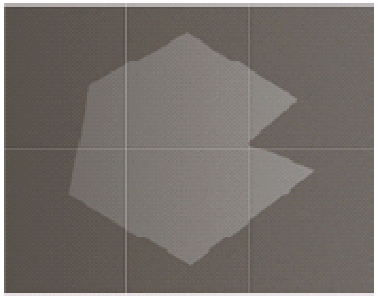
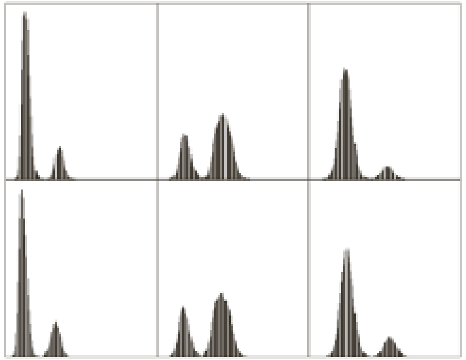


Image partitioning



Histogram of each subimage

Segmentation result after applying Otsu's method to each subimage

Variable Thresholding

- **Variable Thresholding Based on Local Image Properties**
  - Specify different threshold value at each pixel location (x,y) based on some statistical properties of the pixel's neighborhood
  - Some useful statistical measures for a neighborhood S_{xy} are the mean m_{xy} and standard deviation σ_{xy}
  - Some common forms for specifying the variable threshold are

$$T_{xy} = a\,\sigma_{xy} + b\,m_{xy}$$

OR

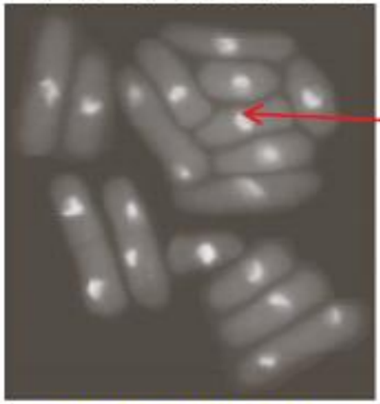
$$T_{xy} = a\,\sigma_{xy} + b\,m_G$$

  - We can also use by using predicate functions based on the neighborhood parameters

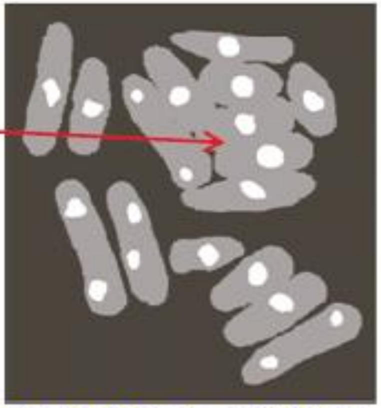
$$g(x,y) = \begin{cases} 1\,, & f(x,y) > a\sigma_{xy} \text{ and } f(x,y) > m_{xy} \\ 0\,, & otherwise \end{cases}$$

Variable Thresholding

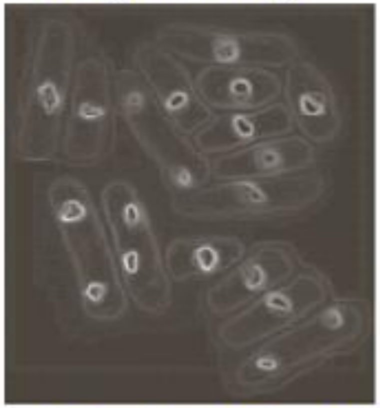
- **Variable Thresholding Based on Local Image Properties - EXAMPLE**



Original Image



Dual Segmentation



Local Standard Deviation Image (3x3 neighborhood)

Variable Thresholding using g1(x,y)

$$g(x,y)=\begin{cases}1\ , & f(x,y) > 30\sigma_{xy} \text{ and } f(x,y) > 1.5m_G\\ 0\ , & otherwise\end{cases}$$

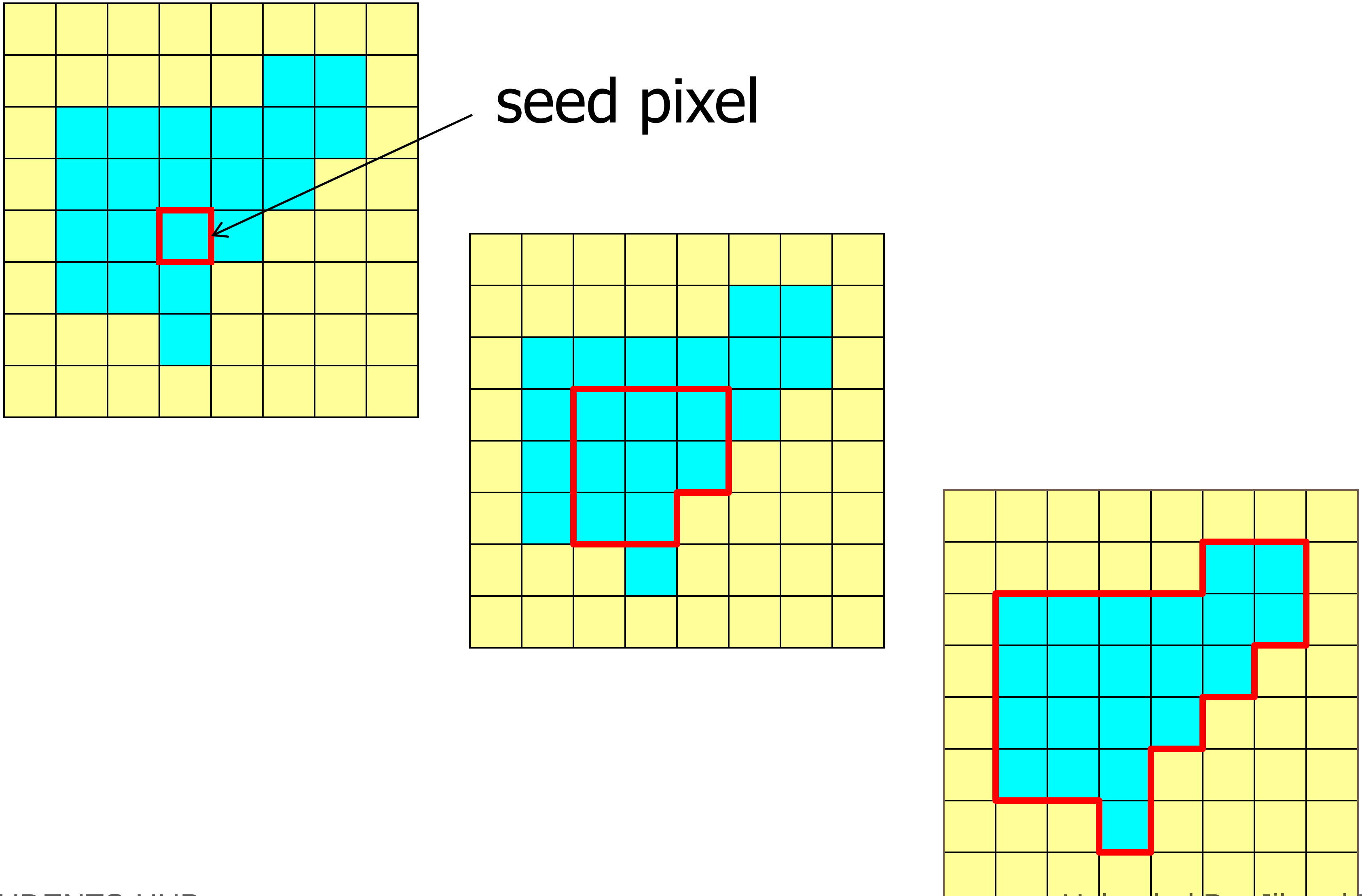
Other Thresholding/Binarization Techniques

- Niblack’s method
- local-variance-based method by Sauvola
- Local adaptive method proposed by Bernsen
- Entropy-based method By Kapur
- learning framework for the optimization of the binarization methods by Cheriet

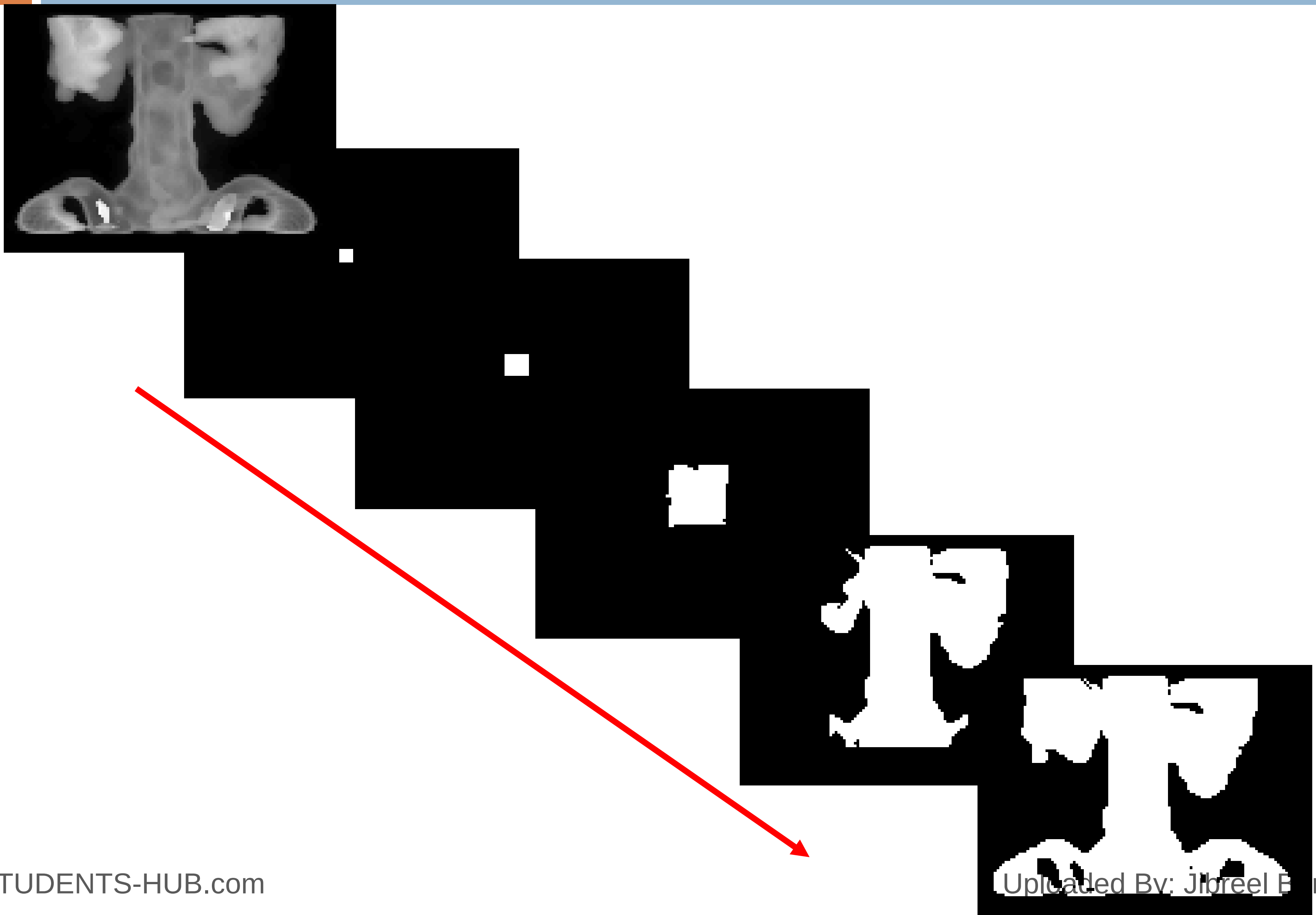
Segmentation by Region Growing

- The approach is based on grouping pixels or subregions into larger regions based on a predefined criteria for growth
- The basic approach is to start from a set of seed pixels and from these grow regions by appending to each seed those neighboring pixels that meet the predefined properties
- How to select the seeds ?
  - Nature of problem
  - Random
  - Interactively
- How to select the similarity properties?
  - Nature of problem and image type (color , monochrome ..)
  - Use statistical measures of local neighborhoods
  - May incorporate pixel location

Segmentation by Region Growing



Segmentation by Region Growing

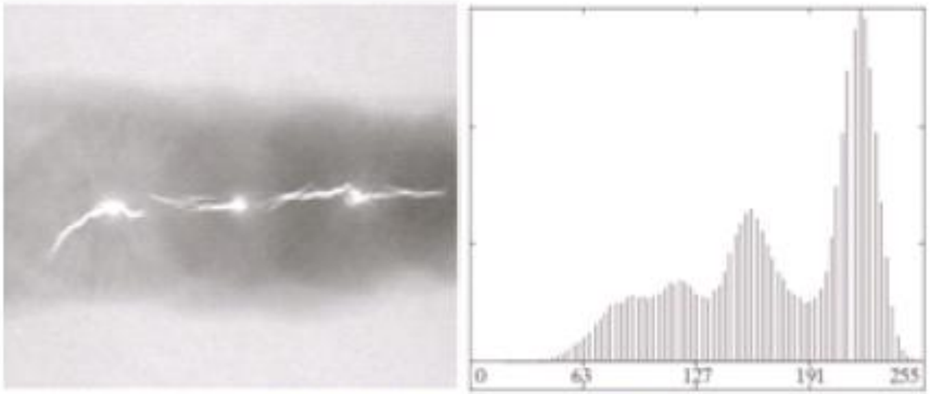


Segmentation by Region Growing

- **A General Region Growing Algorithm**
  1. Define the similarity criteria (difference intensity, variance, …)
  2. Define the stopping criteria (properties of the region, size, shape, …)
  3. Select a set of seed pixels; S
  4. Define an empty array O(x,y) and initialize its elements to 0's except at seed locations. Seed locations are assigned different intensity values
  5. For each seed in S, check its neighbors (4-,8-, or d-neighbors) against the similarity criteria
  6. If any of the neighboring pixels satisfy the criteria, then set the corresponding location in O to the same intensity level as its seed
  7. Check the if the stopping criteria is not met. If not, repeat steps 4 through 7 to the newly added pixels
  8. Repeat steps 4 through 7 until all seed pixels are processed

Segmentation by Region Growing

- **Example: Segment the cracks in the weld**



Original Image and its histogram

Thresholding result to identify regions of high intensity

Seeds specified by random selection from those in the thresholded image

Result of region growing

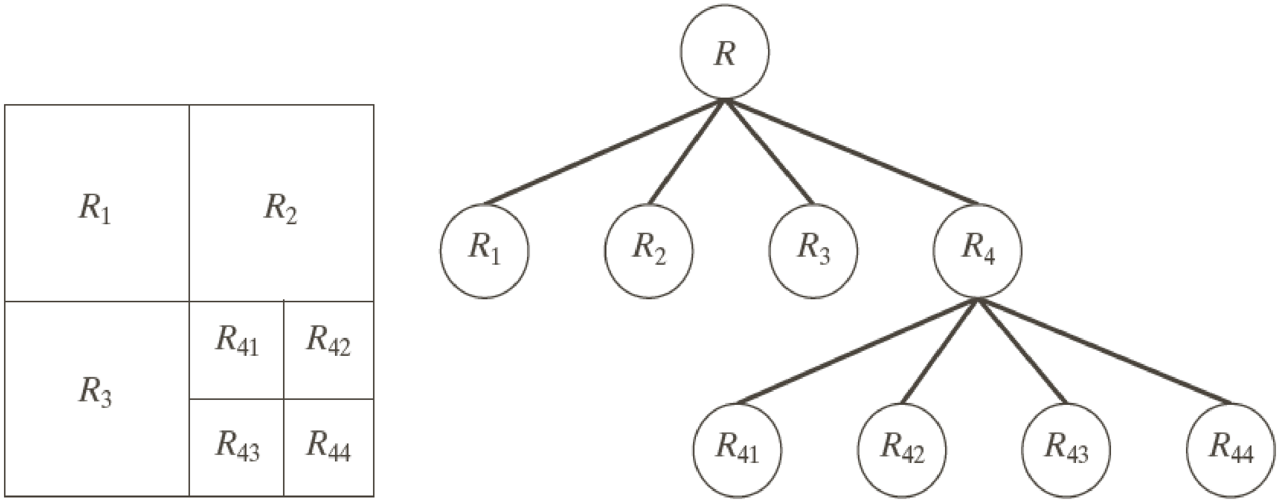
$$g(x,y) = \begin{cases} 1, & |f(x,y) - N(x,y)| > 126 \\ 0, & otherwise \end{cases}$$

Segmentation by Region Splitting and Merging

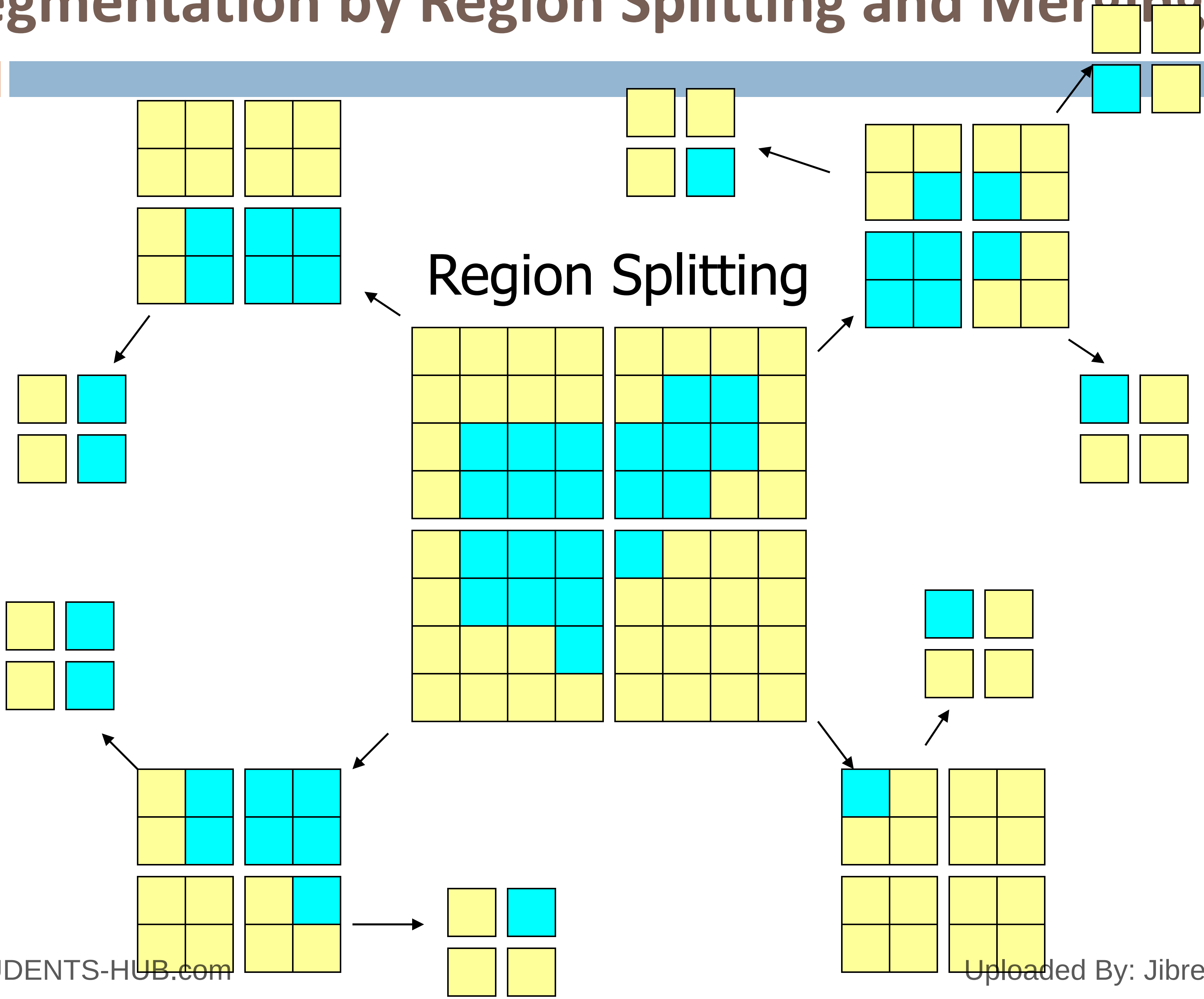
- Unlike region growing, region splitting and merging approaches the segmentation problem in a top-down approach
- In this method, the image region is split into a set of disjoint regions and then merge and/or split the regions to satisfy certain criteria and the segmentation conditions we discussed earlier
- Let R represents the entire image region and Q be a given predicate function. Region splitting attempt to subdivide the image successively into smaller and smaller quadrant regions, R_i, such that $Q(R_i)$ = TRUE. We start from the entire image R. If Q(R) =FALSE, subdivided it into 4 quadrants. Repeat the process for every quadrant and divide it into 4 quadrants if $Q(R_i)$= FALSE
- Once no splitting is possible (according to some criteria), check adjacent quadrants. If the $Q(R_j \cup R_k)$ = TRUE, then merge these two regions

Segmentation by Region Splitting and Merging

- **Algorithm**
  1. Split into four disjoint quadrants any region Ri for which Q(Ri) = FLASE
  2. When no further splitting is possible based on some criteria (size of quadrant), merge any adjacent regions for which $Q(R_j \cup R_k)$ = TRUE
  3. Stop when no further merging is possible

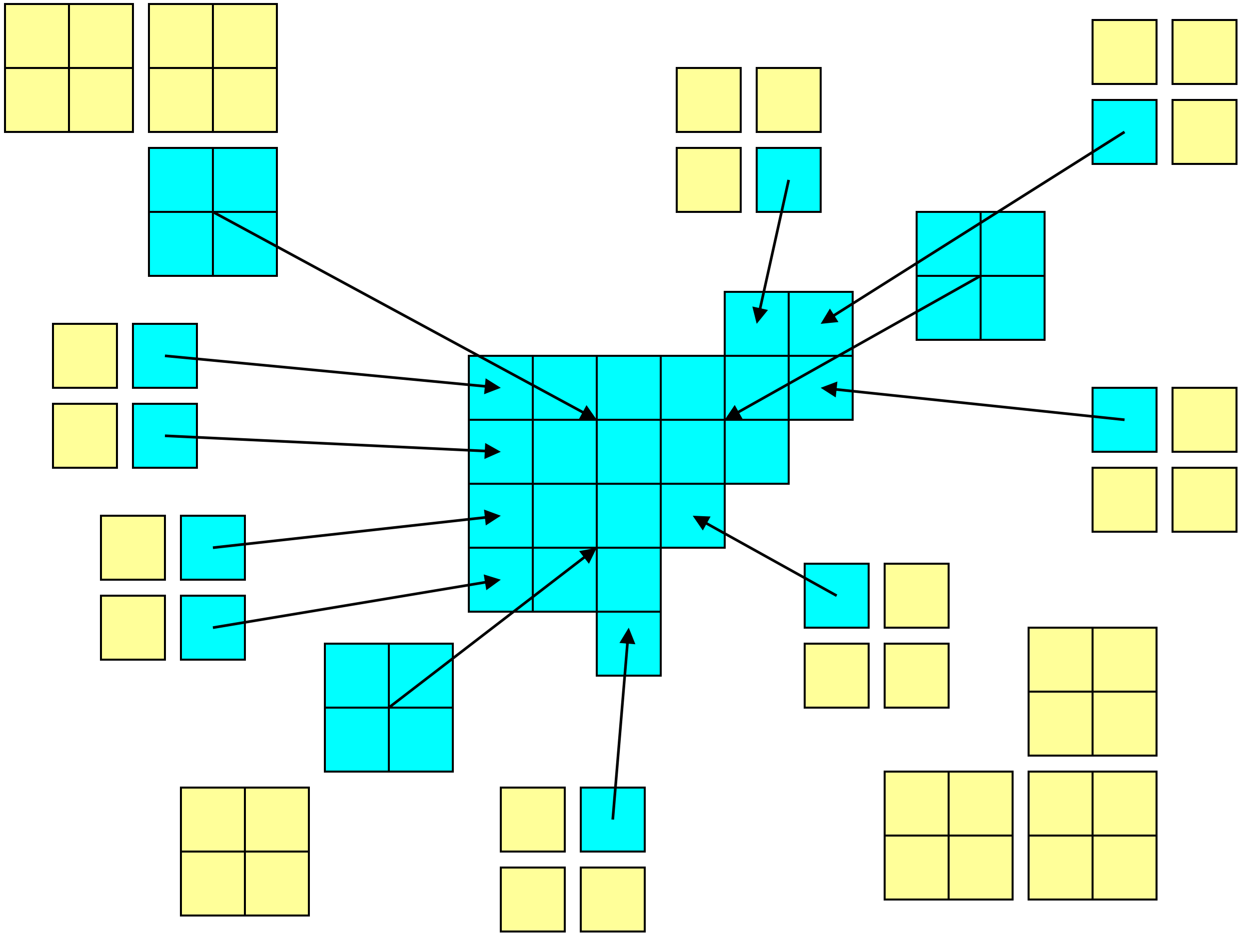


Segmentation by Region Splitting and Merging



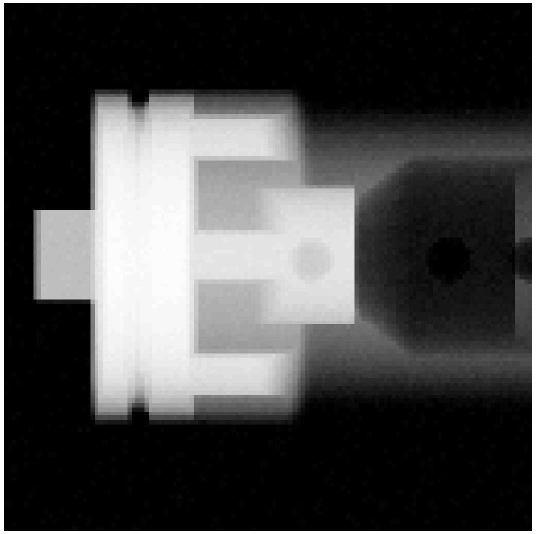
Segmentation by Region Splitting and Merging

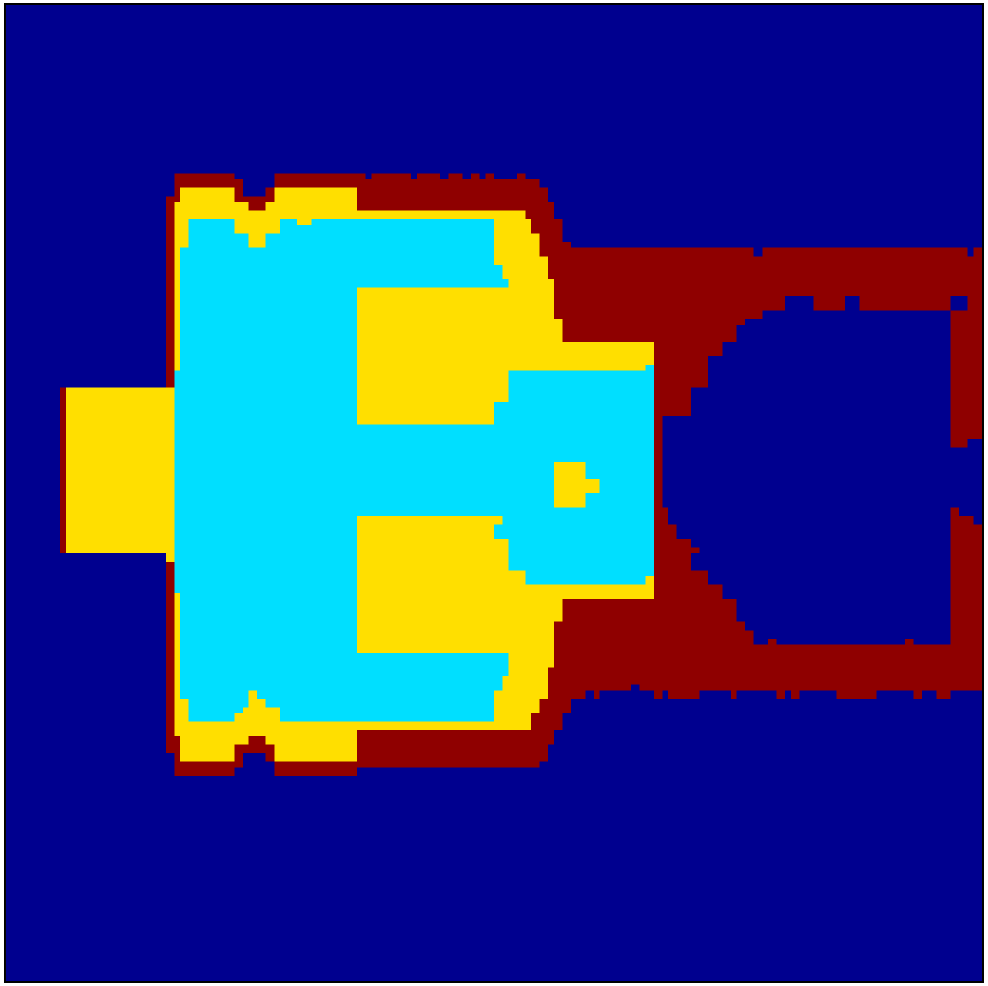
Merging



Segmentation by Region Splitting and Merging

- Merging





Segmentation by Region Splitting and Merging

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

original image

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 1

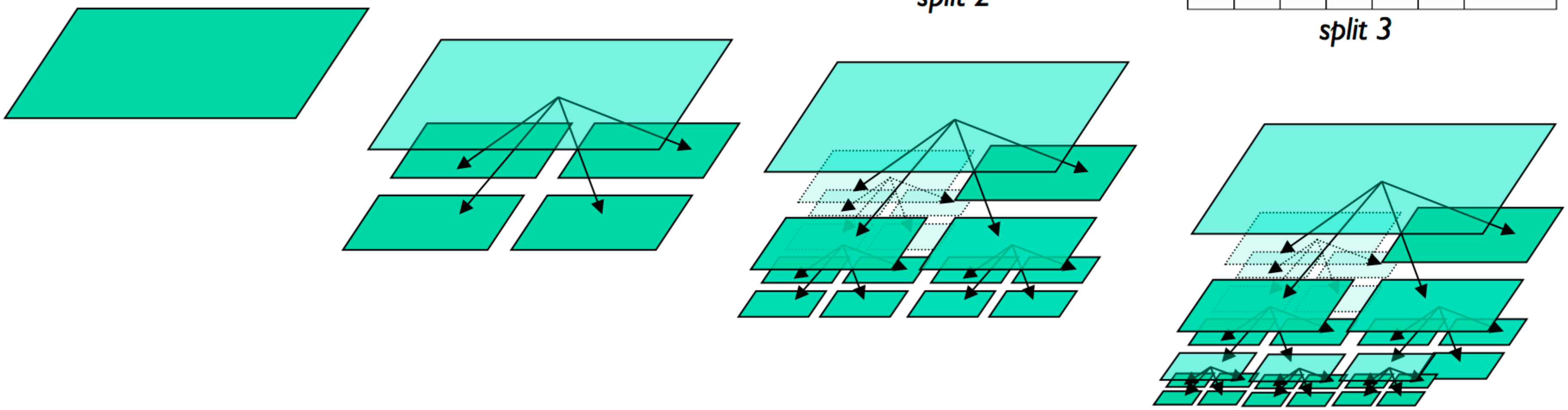
0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 2

0	1	0	0	7	7	7	7
1	0	2	2	7	7	7	7
0	2	2	2	7	7	7	7
4	4	2	2	7	7	7	7
0	0	1	1	3	3	7	7
1	1	2	2	3	7	7	7
2	4	3	0	5	7	7	7
2	3	3	5	5	0	7	7

split 3

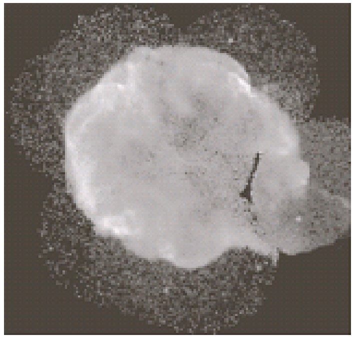
http://



Segmentation by Region Splitting and Merging

- **Example**
  - Segment the less dense matter which is of noisy nature (high standard deviation when compared to background and the dense region) and moderate intensity
  - Use the predicate function

$$g(x,y)=\begin{cases} 1\,, & \sigma > 10 \text{ and } \; 0<m<126 \\ 0\,, & otherwise \end{cases}$$



Original

Minimum quadrant size 32x32

Minimum quadrant size 16x16

Minimum quadrant size 8x8

Other Segmentation Techniques

- Morphological Watersheds
- Clustering Based Segmentation Methods
- Graph-based methods (graph---cut, random walk)
- Shape-based methods (level set, active contours)
- Energy minimization methods (MRF,..)
- Machine Learning based methods