# Image Processing and Pattern Recognition (IPPR)

## Chapter 7:Image Segmentation


Basanta Joshi, PhD

Asst. Prof., Depart of Electronics and Computer Engineering

Member, Laboratory for ICT Research and Development (LICT)

Institute of Engineering

basanta@ioe.edu.np

http://www.basantajoshi.com.np

https://scholar.google.com/citations?user=iocLiGcAAAAJ
https://www.researchgate.net/profile/Basanta_Joshi2


# The Segmentation Problem

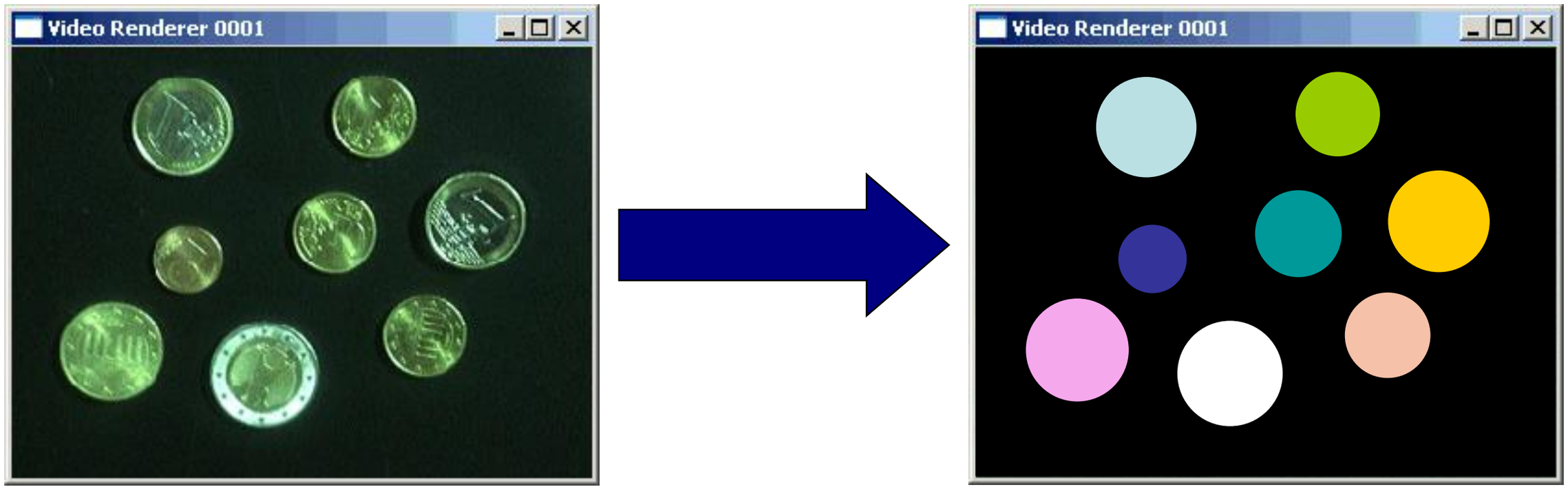
- Segmentation divides an image into its constituent parts or objects
- Level of subdivision depends on the problem being solved
- Segmentation stops when objects of interest in an application have been isolated

- Example:
  - For an air-to-ground target acquisition system interest may lie in identifying vehicles on a road
    - Segment the road from the image
    - Segment contents of the road down to objects of a range of sizes that correspond to potential vehicles
    - No need to go below this level, or segment outside the road boundary

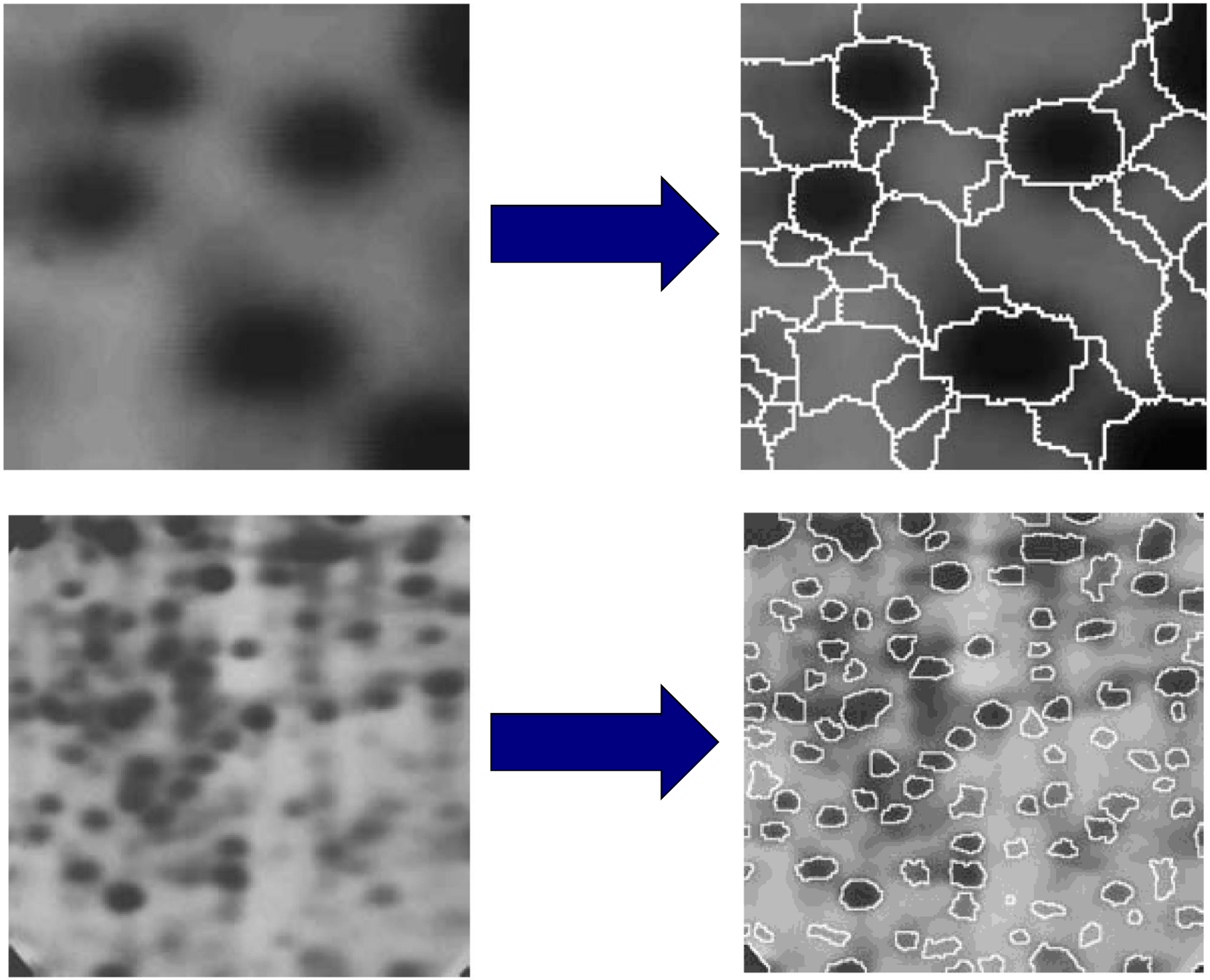
# The Segmentation Problem

Segmentation attempts to partition the pixels of an image into groups that strongly correlate with the objects in an image

Typically the first step in any automated computer vision application



# Segmentation Examples



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Segmentation

- Autonomous segmentation is one of the most difficult tasks in image processing - largely determines the eventual failure or success of the process
- Segmentation algorithms for monochrome images are based on one of two basic properties of gray-level values
  - Discontinuity
  - Similarity
- For discontinuity, the approach is to partition an image based on abrupt changes in gray level
- The principal areas of interest are:
  - detection of isolated points
  - detection of lines and edges in an image

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Image Segmentation

- For similarity, the principal approaches are based on
  - thresholding
  - region growing
  - region splitting
  - merging
- Using discontinuity and similarity of gray-level pixel values is applicable to both static and dynamic (time varying) images
- For dynamic images, the concept of motion can be exploited in the segmentation process

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Detection Of Discontinuities

There are three basic types of grey level discontinuities that we tend to look for in digital images:

- Points
- Lines
- Edges

We typically find discontinuities using masks and correlation

# Detection Of Discontinuities

- Detecting discontinuities (points, lines and edges) is generally accomplished by mask processing (much as in the spatial domain filter examples)
- Use the response equation

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9$$

$$= \sum_{i=1}^{9} w_i z_i$$

- A mask used for detecting isolated points (different from a constant background would be

| | | |
|---|---|---|
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |

# Detection Of Discontinuities

- Detection of isolated points is accomplished by using the previous mask
- An isolated point is detected if the response of the mask is greater that a predetermined threshold

$$|R| > T$$

- This measures the weighted difference between a center point and its neighbors
- The mask is the same as the high frequency filtering mask
- The emphasis here is on the detection of points
  - Only differences that are large enough to be considered isolated points in an image are of interest

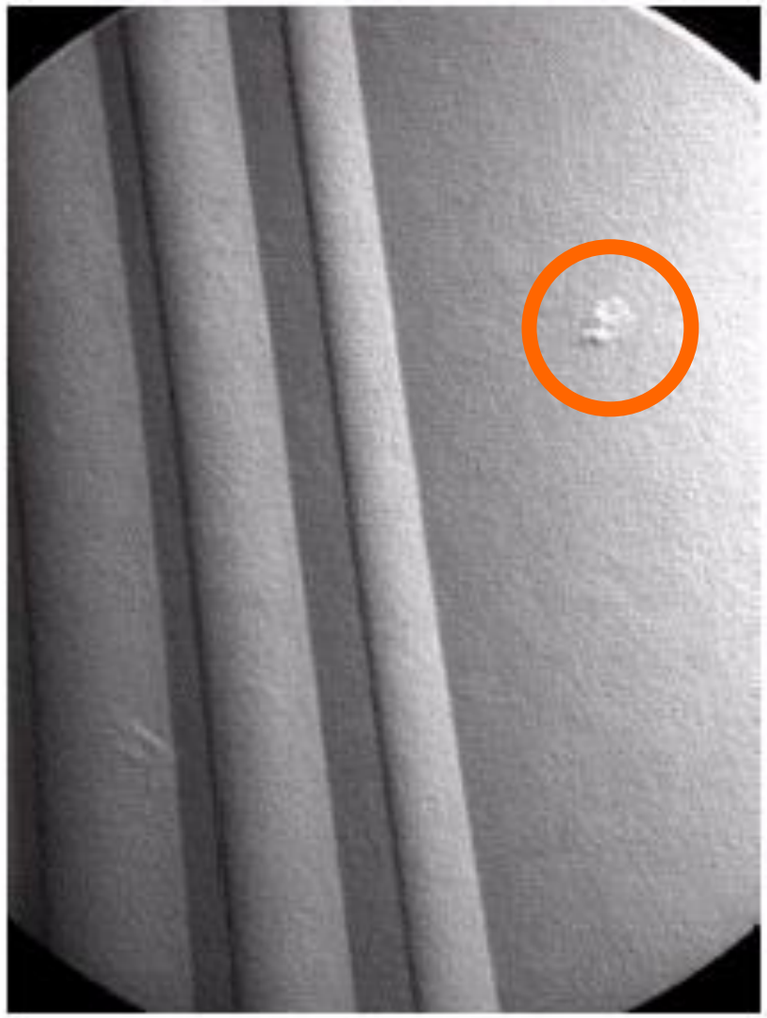
# Point Detection

Point detection can be achieved simply using the mask below:

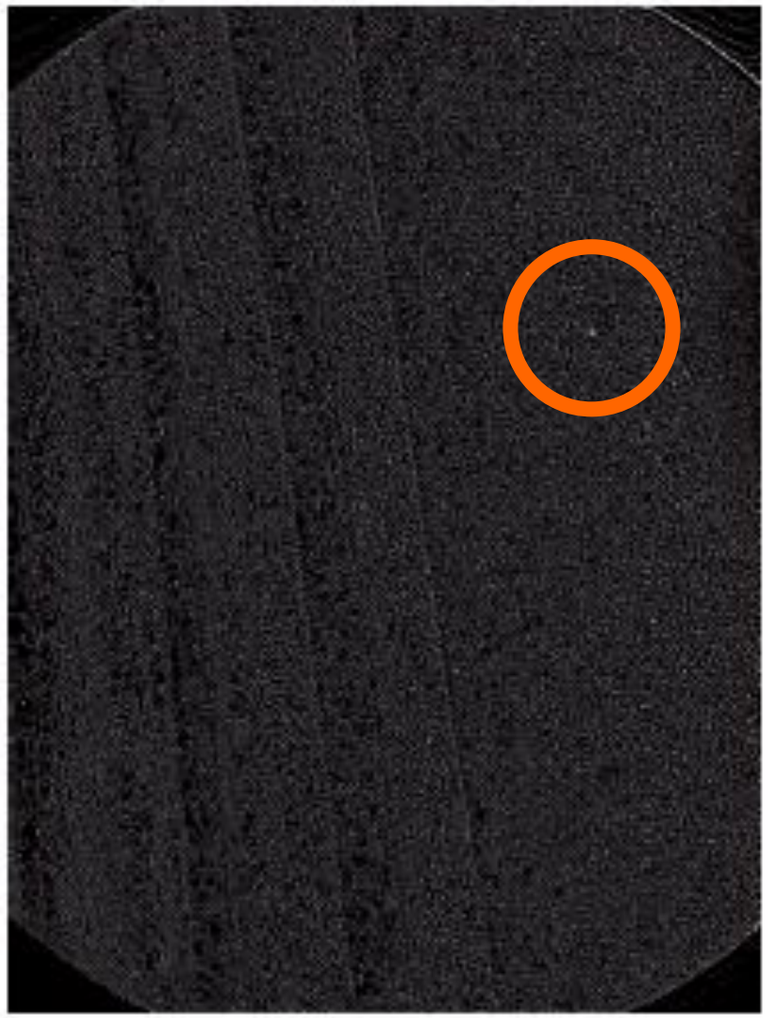
| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

Points are detected at those pixels in the subsequent filtered image that are above a set threshold

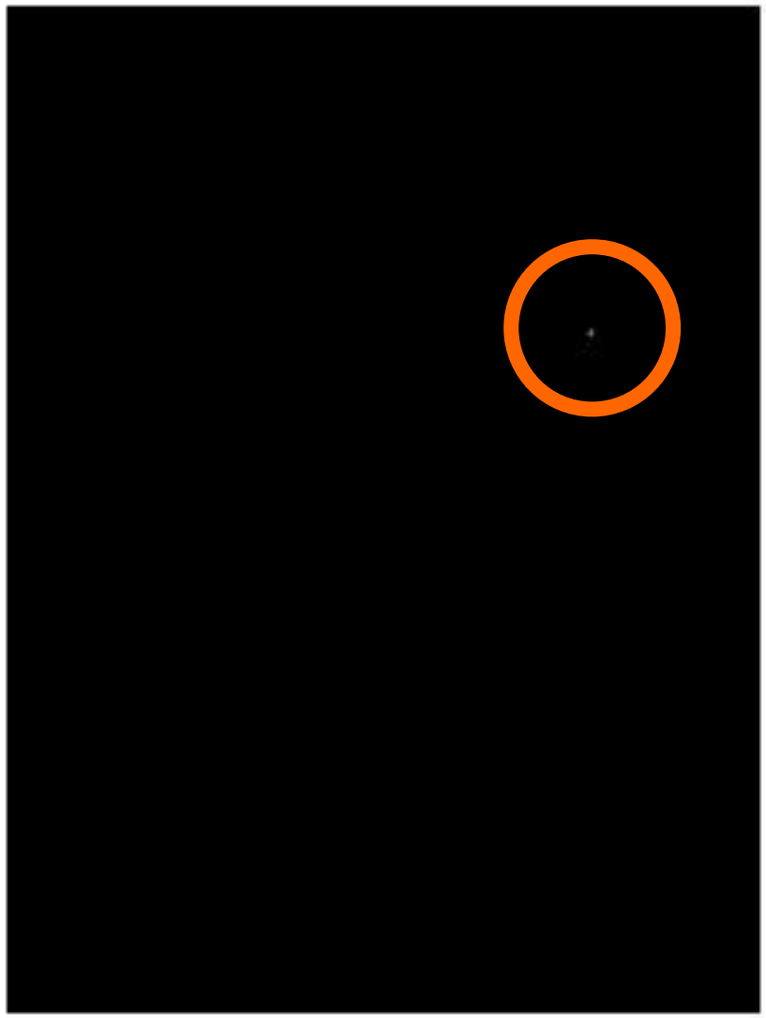
# Point Detection (cont…)



X-ray image of a turbine blade



Result of point detection



Result of thresholding

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Line Detection

The next level of complexity is to try to detect lines

The masks below will extract lines that are one pixel thick and running in a particular direction

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 2 | 2 | 2 |
| −1 | −1 | −1 |

Horizontal

| | | |
|---|---|---|
| −1 | −1 | 2 |
| −1 | 2 | −1 |
| 2 | −1 | −1 |

+45°

| | | |
|---|---|---|
| −1 | 2 | −1 |
| −1 | 2 | −1 |
| −1 | 2 | −1 |

Vertical

| | | |
|---|---|---|
| 2 | −1 | −1 |
| −1 | 2 | −1 |
| −1 | −1 | 2 |

−45°

# Line Detection

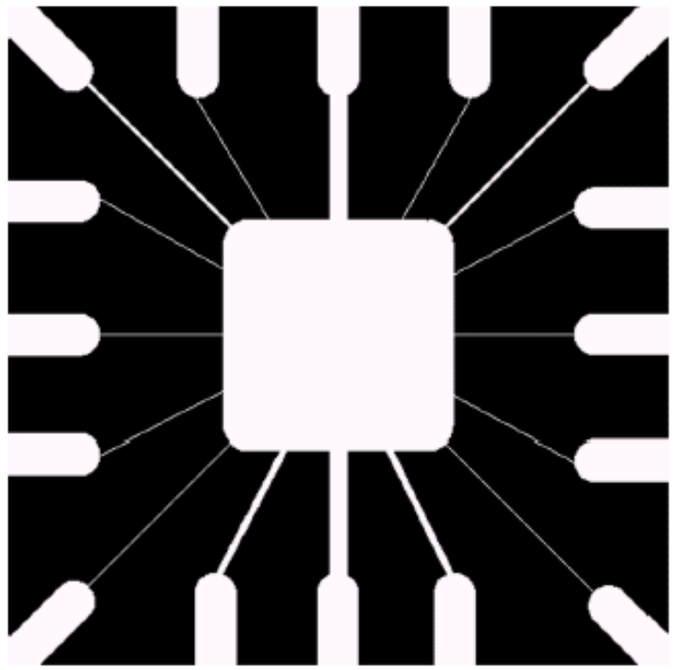
- With a constant background, the maximum response occurs when the line is "lined up" with the center of the mask
- Note that the preferred direction of each mask is weighted with a larger coefficient than other possible directions
- Let $R_1$, $R_2$, $R_3$ and $R_4$ denote the responses of the masks
- If, at a certain point in the image,

$$|R_i| > |R_j| \text{ for all } j \neq i$$

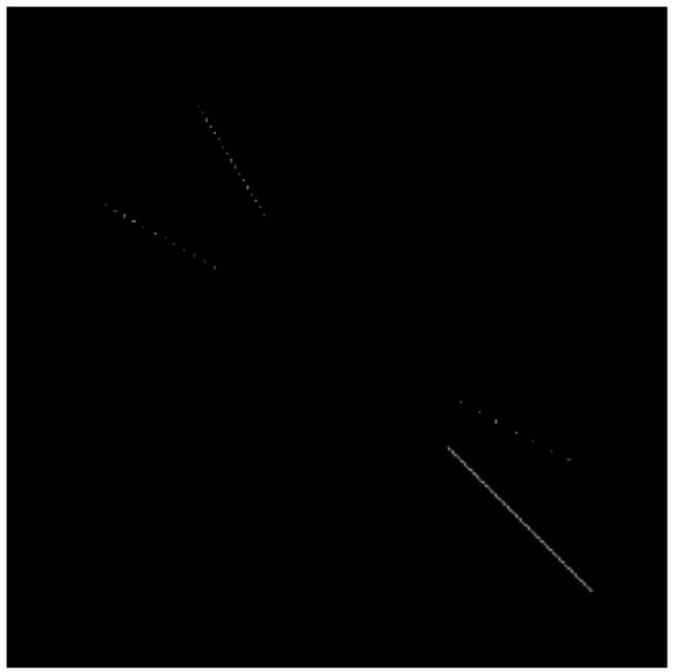
- that point is said to be more likely associated with a line in the direction of mask *i*

# Line Detection (cont…)

Binary image of a wire bond mask



After processing with -45° line detector



Result of thresholding filtering result

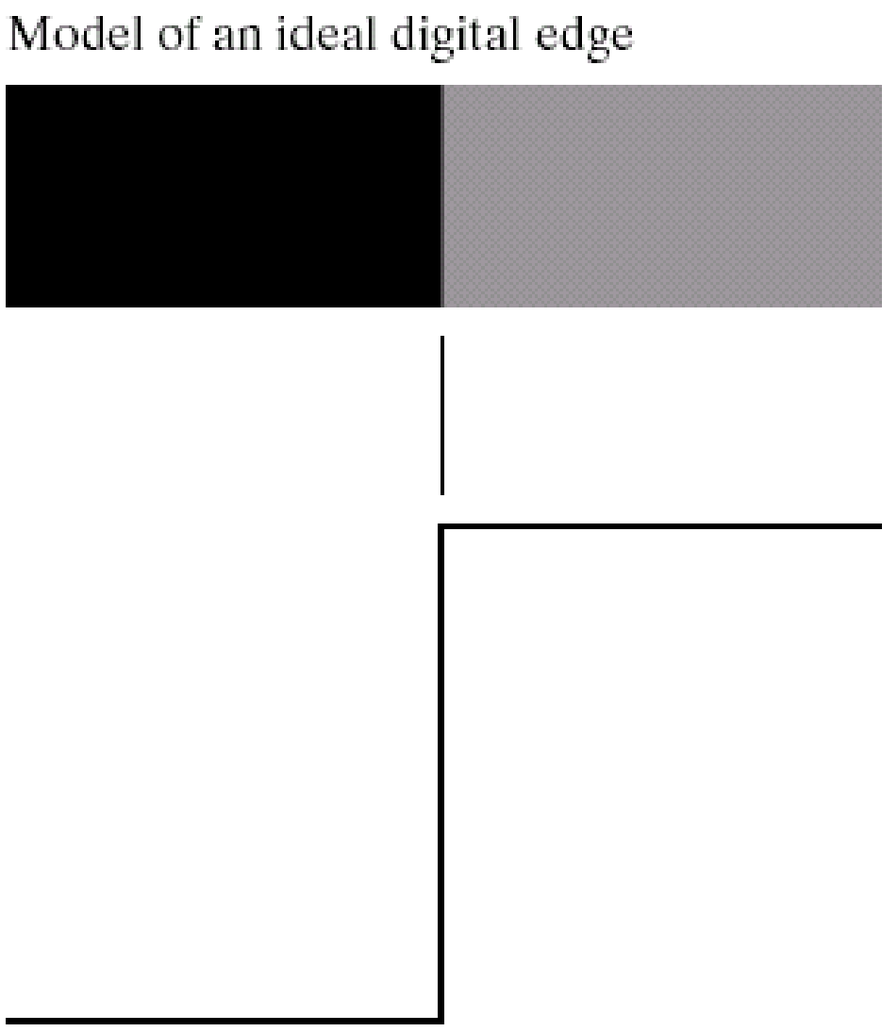
# Edge Detection

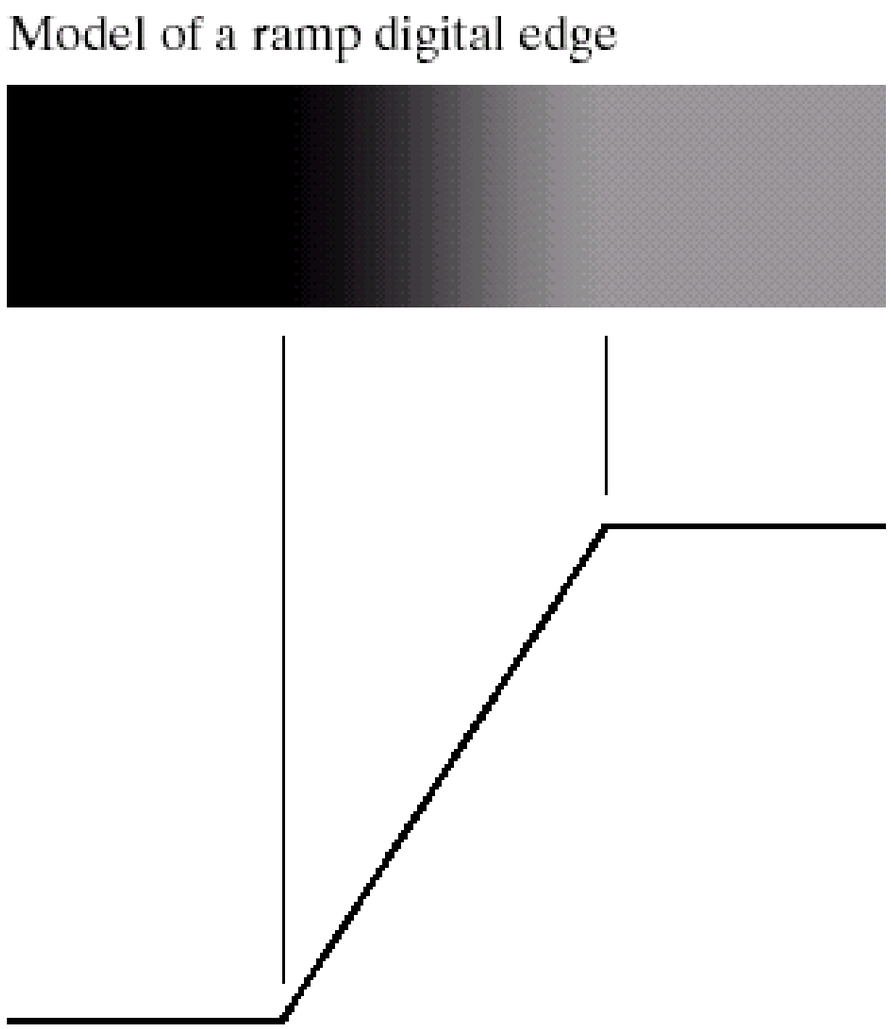
- Edge detection is by far the most common approach for detecting discontinuities in gray levels
  - Isolated points and 1-pixel thin lines are not common in most practical applications
- Basic formulation and initial assumptions
  - An edge is a boundary between two regions with relatively distinct gray-level properties
  - Regions are sufficiently homogeneous so that the transition between the regions can be determined on the basis of gray-level discontinuities alone
  - If this is not valid, some other techniques will be used
- The basic idea behind most edge detection techniques is the computation of a local derivative operator

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Edge Detection

An edge is a set of connected pixels that lie on the boundary between two regions

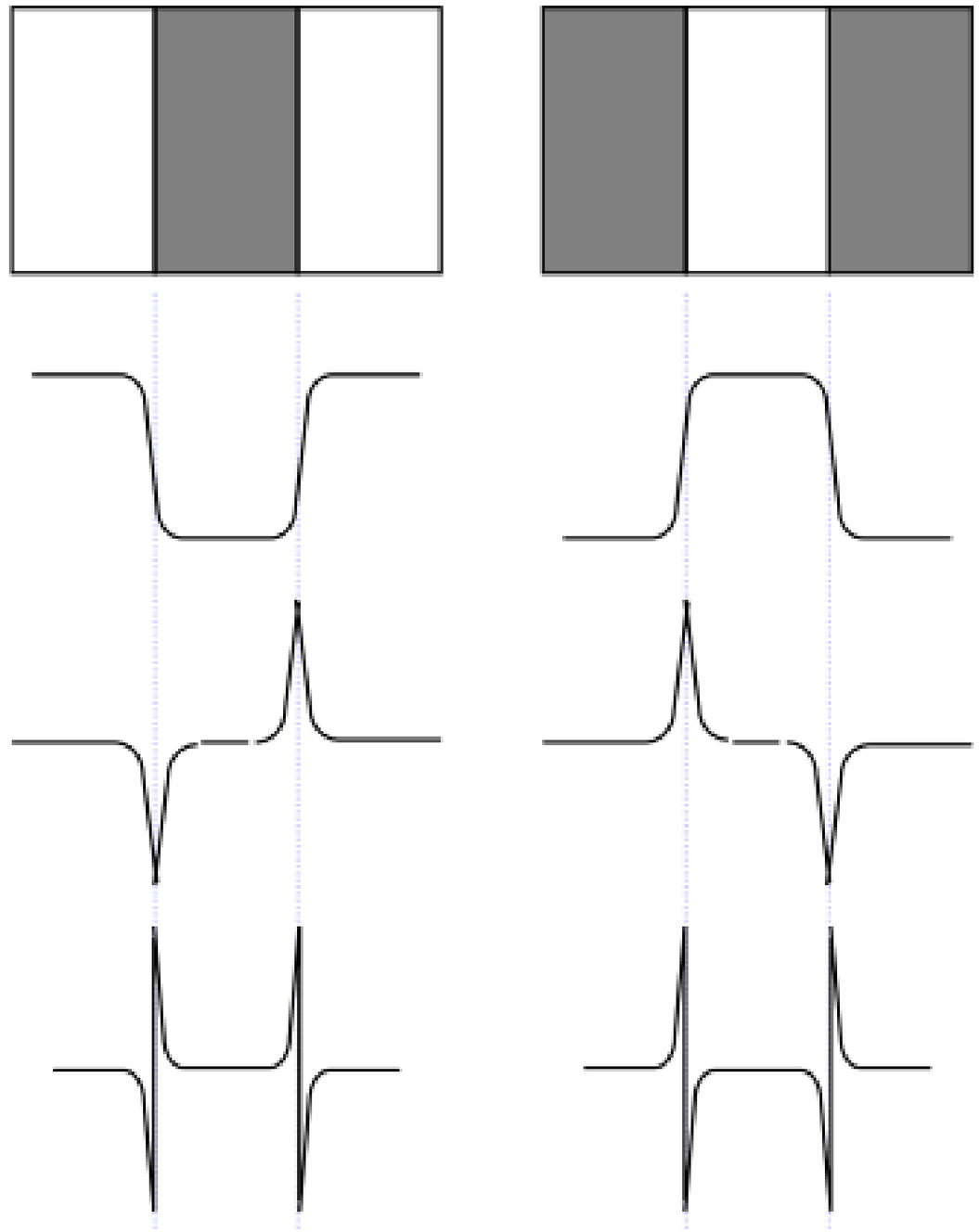



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Derivative Operators

- An image of a dark stripe on a light background (and visa versa)
- A profile of the lines in the image (modeled as a gradual rather than sharp transition)
  - Edges in images tend to be slightly blurred as a result of sampling
- The first derivative: the magnitude detects the presence of an edge
- The second derivative: the sign tells the type of transition (light-to-dark or dark-to-light) Note also the presence of a zero-crossing at each edge



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Derivative Operators

- The first derivative at any point in an image is computed using the magnitude of the gradient

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

- Where the *magnitude* is

$$\nabla f = \mathrm{mag}(\nabla \mathrm{f}) = \left[ G_x^2 + G_y^2 \right]^{1/2}$$

$$\approx |G_x| + |G_y|$$

- The *direction* of the gradient vector is the angle $\alpha(x,y)$ given by

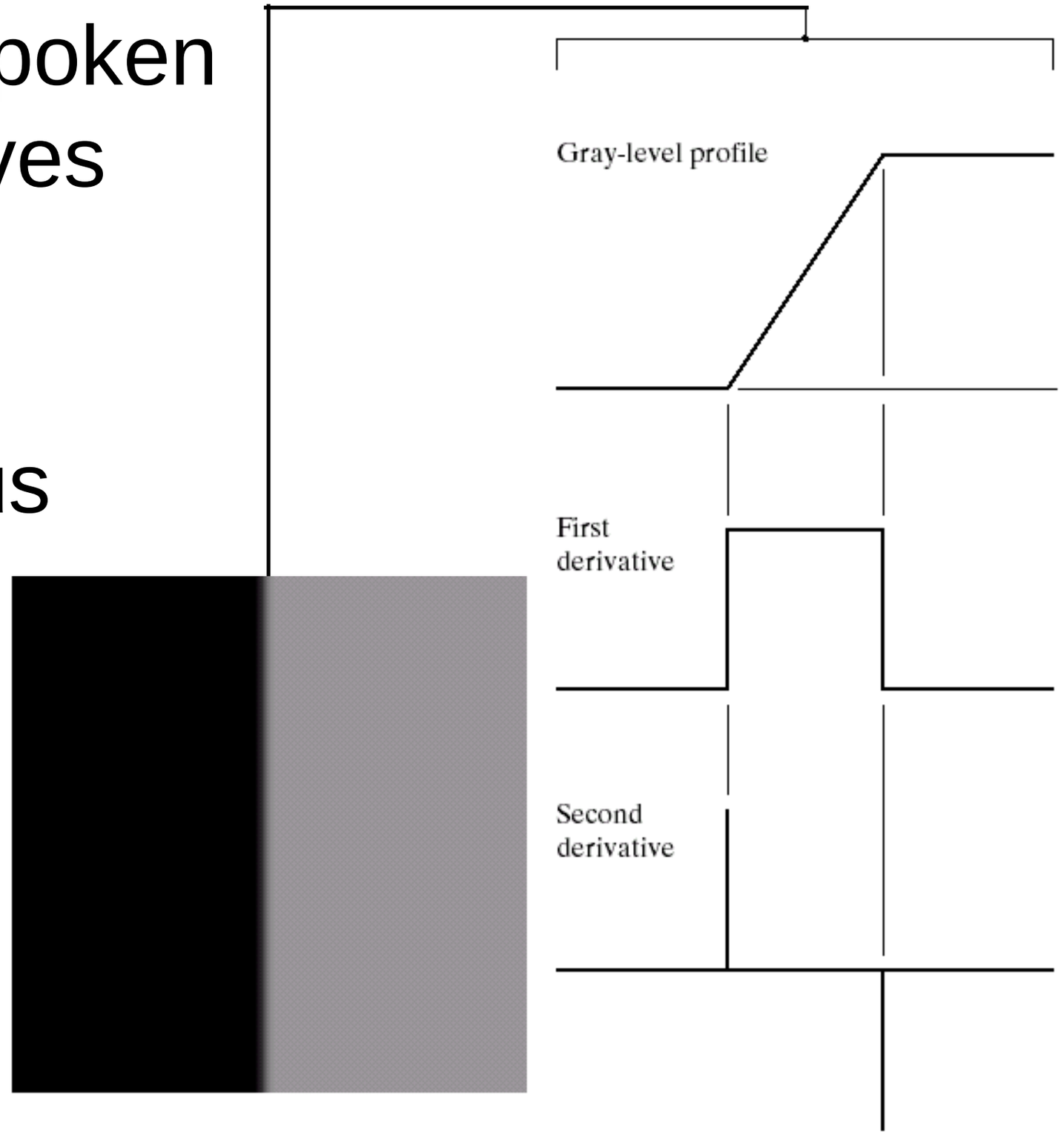
$$\alpha(x, y) = \tan^{-1}\left( \frac{G_y}{G_x} \right)$$

# Edges & Derivatives

We have already spoken about how derivatives are used to find discontinuities

1st derivative tells us where an edge is

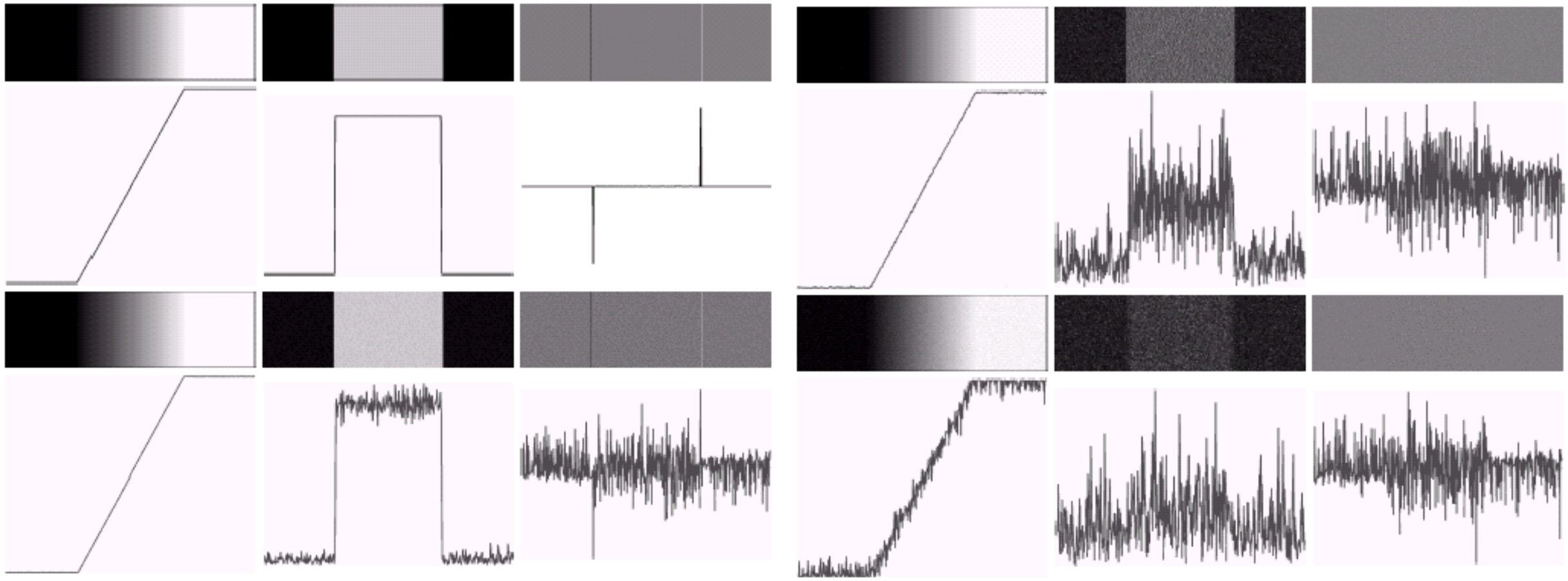
2nd derivative can be used to show edge direction



# Derivatives & Noise

Derivative based edge detectors are extremely sensitive to noise

We need to keep this in mind



# Common Edge Detectors

Given a 3*3 region of an image the following edge detection filters can be used

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −1 | 0 | 1 |
| −1 | 0 | 1 |

Prewitt

| | |
|---|---|
| −1 | 0 |
| 0 | 1 |

| | |
|---|---|
| 0 | −1 |
| 1 | 0 |

Roberts

| | | |
|---|---|---|
| −1 | −2 | −1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

| | | |
|---|---|---|
| −1 | 0 | 1 |
| −2 | 0 | 2 |
| −1 | 0 | 1 |

Sobel

# Edge Detection Example

Original Image

Horizontal Gradient Component

Vertical Gradient Component

Combined Edge Image

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Edge Detection Example

# Edge Detection Example

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Edge Detection Example

# Edge Detection Example

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Edge Detection Problems

Often, problems arise in edge detection in that there are is too much detail

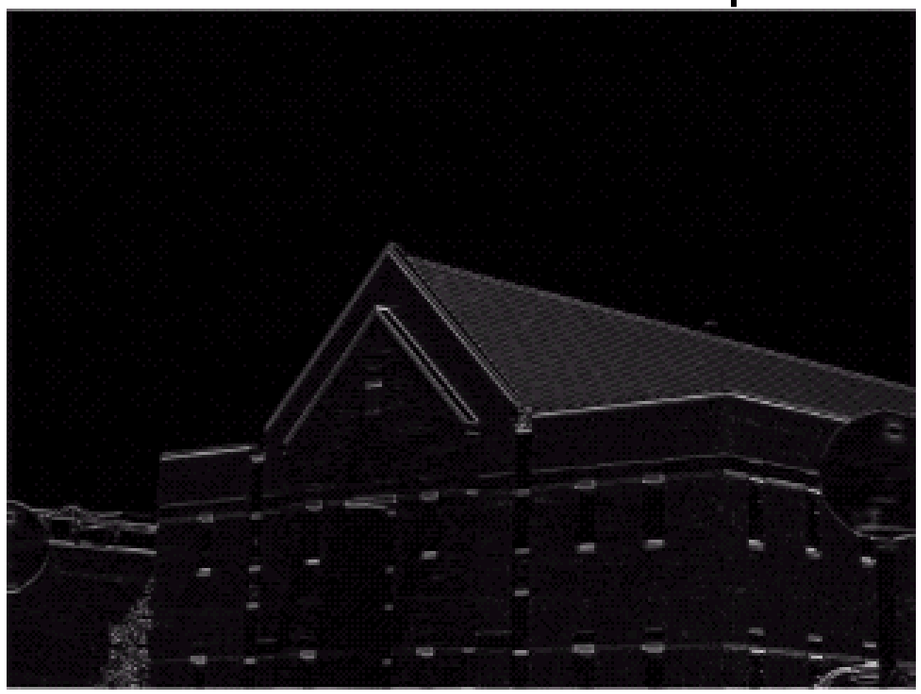
For example, the brickwork in the previous example

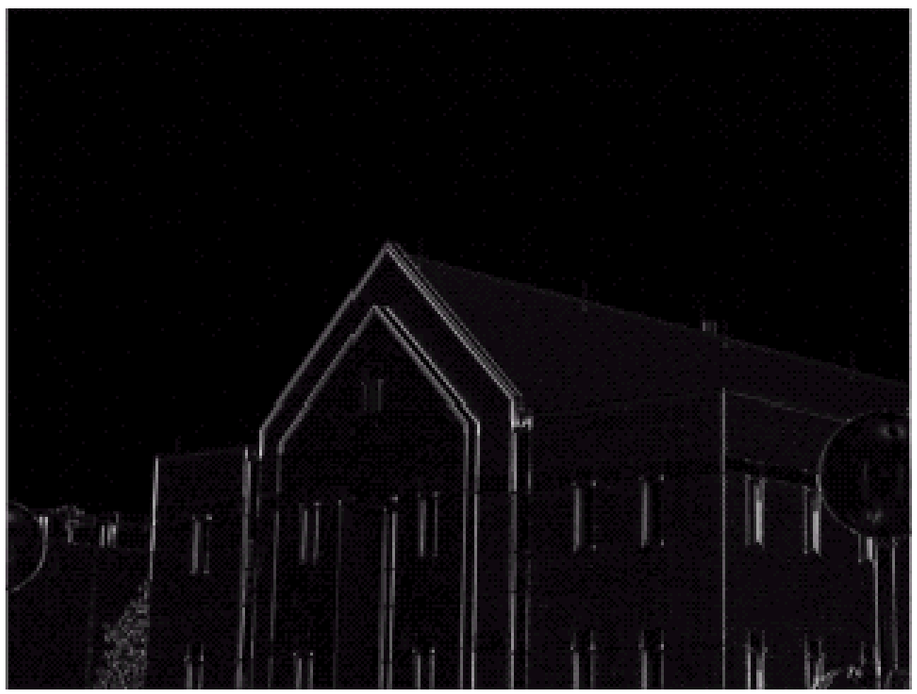
One way to overcome this is to smooth images prior to edge detection

# Edge Detection Example With Smoothing

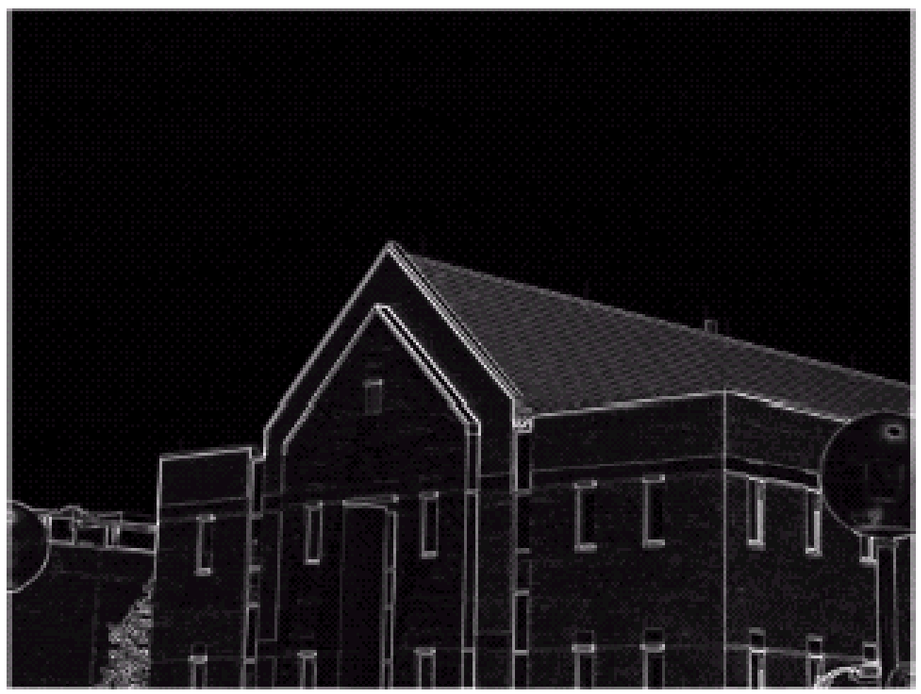
Original Image

Horizontal Gradient Component





Vertical Gradient Component



Combined Edge Image

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Laplacian Edge Detection

We encountered the 2nd-order derivative based Laplacian filter already

| | | |
|---|---|---|
| 0 | −1 | 0 |
| −1 | 4 | −1 |
| 0 | −1 | 0 |

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

The Laplacian is typically not used by itself as it is too sensitive to noise

Usually when used for edge detection the Laplacian is combined with a smoothing Gaussian filter

# Laplacian Edge Detection

- Although the Laplacian responds to changes in intensity, it is seldom used in edge detection for several reasons
  - As a second derivative operator it is typically unacceptably sensitive to noise
  - The Laplacian produces double edges
  - Unable to detect direction
- As such, the Laplacian is used in the secondary role of detector for establishing whether a pixel is on the light or dark side of an edge

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

# Laplacian Edge Detection

- A more general use of the Laplacian is to find the location of edges using the zero-crossings property
- Basic idea is to convolve an image with the Laplacian of a 2-D Gaussian function of the form

$$h(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

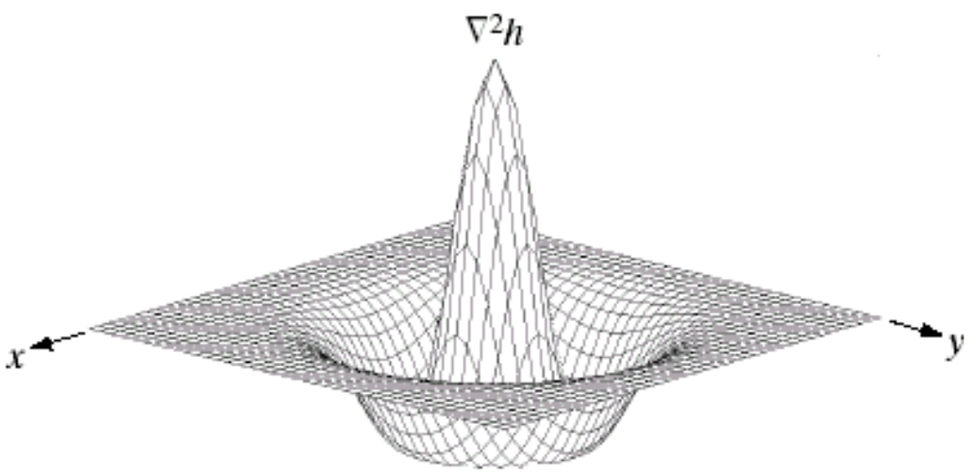
- $\sigma$ = standard deviation.
- If $r^2=x^2+y^2$, then the Laplacian is then

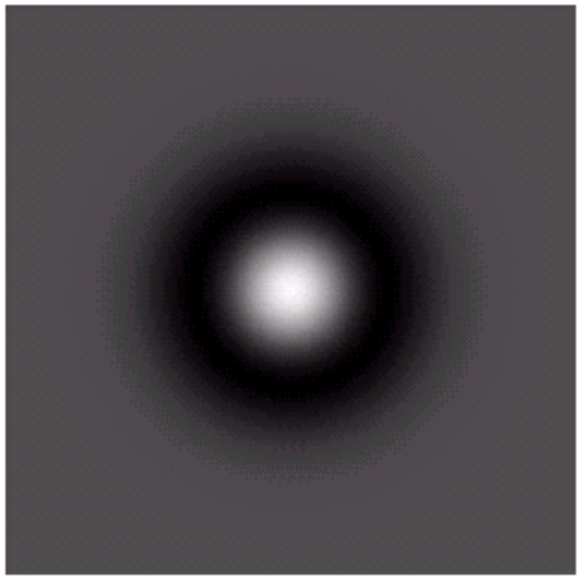
$$\nabla^2 h = \left(\frac{r^2-\sigma^2}{\sigma^4}\right)\exp\left(-\frac{r^2}{2\sigma^2}\right)$$

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

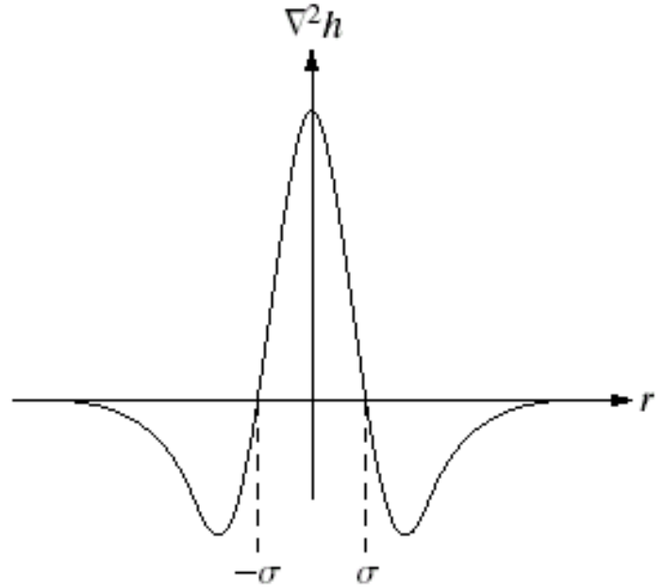
# Laplacian Of Gaussian

The Laplacian of Gaussian (or Mexican hat) filter uses the Gaussian for noise removal and the Laplacian for edge detection



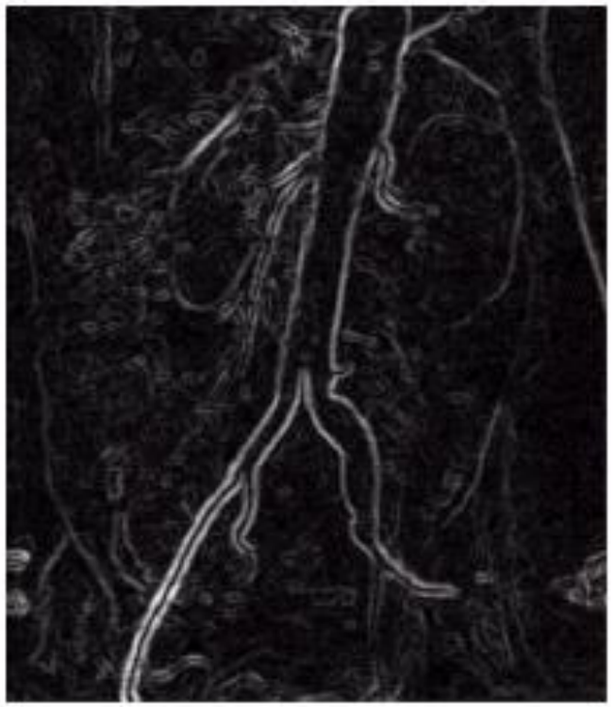



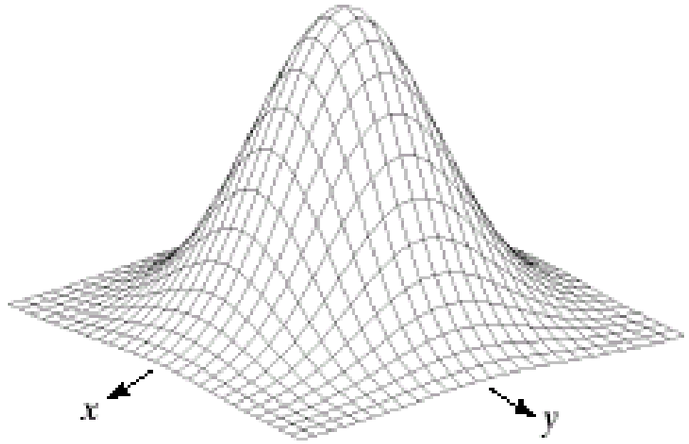
| | | | | |
|---|---|---|---|---|
| 0 | 0 | −1 | 0 | 0 |
| 0 | −1 | −2 | −1 | 0 |
| −1 | −2 | 16 | −2 | −1 |
| 0 | −1 | −2 | −1 | 0 |
| 0 | 0 | −1 | 0 | 0 |



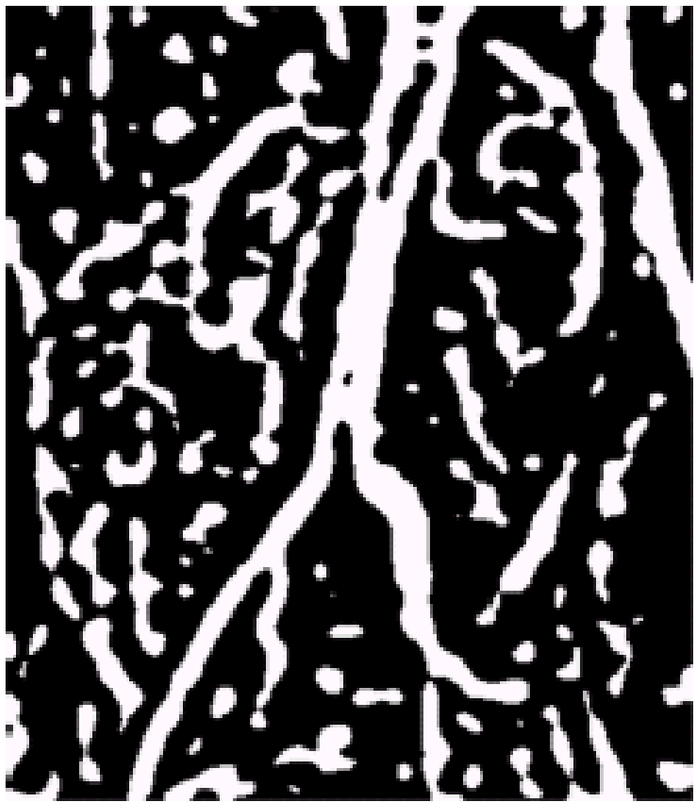

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

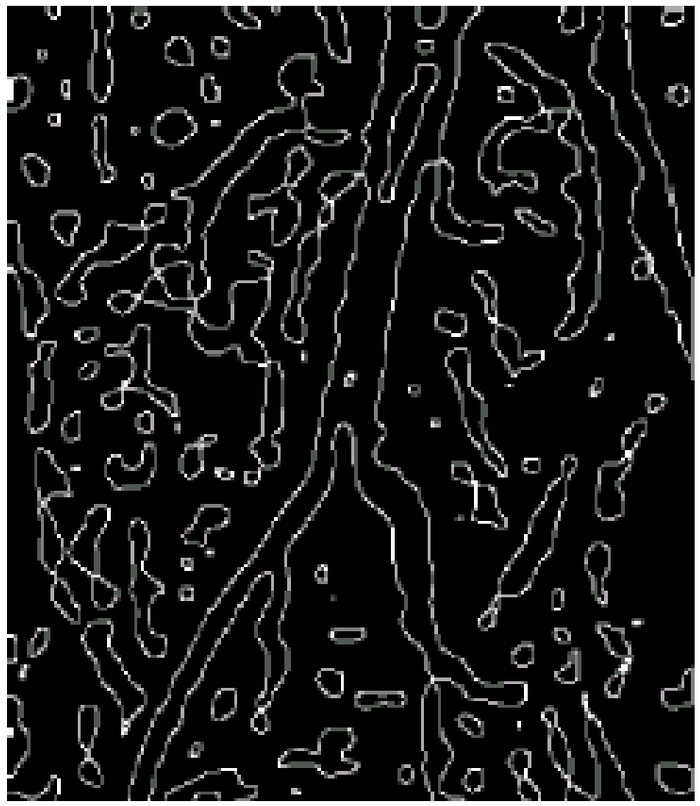
# Laplacian Of Gaussian Example



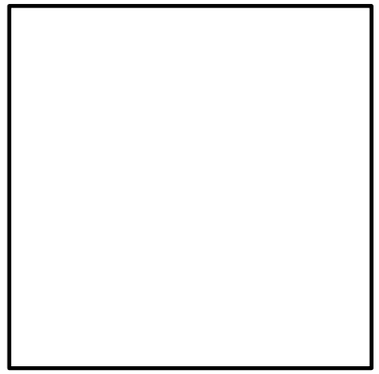



| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |





# Edge Linking and boundary detection



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

*magnitude* is

$$\nabla f = \mathrm{mag}(\nabla \mathrm{f}) = \left[ G_x^2 + G_y^2 \right]^{1/2}$$

$$\approx |G_x| + |G_y|$$

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

# Edge Linking and boundary detection

- Ideally, edge detection techniques yield pixels lying only on the boundaries between regions
- In practice, this pixel set seldom characterizes a boundary completely because of
  - noise
  - breaks in the boundary due to non-uniform illumination
  - other effects that introduce spurious discontinuities
- Thus, edge detection algorithms are usually followed by linking and other boundary detection procedures designed to assemble edge pixels into meaningful boundaries

# Edge Linking : local processing

- Basic idea:
  - Analyze the characteristics of pixels in a small neighborhood (3x3, 5x5, etc) for every point (x,y) that has undergone edge detection
  - All points that are “similar” are linked, forming a boundary of pixels that share some common property
- Two principal properties for establishing similarity
  - The strength of the response of the gradient operator used to produce the edge pixels
  - The direction of the gradient

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

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

*magnitude* is

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}$$

$$\approx |G_x| + |G_y|$$

# Edge Linking : local processing

- An edge pixel at (x',y') in the neighborhood centered at (x,y) is similar in magnitude to the pixel at (x,y) if

$$\left|\nabla f(x,y) - \nabla f(x',y')\right| \leq T$$

- where T is a predetermined threshold

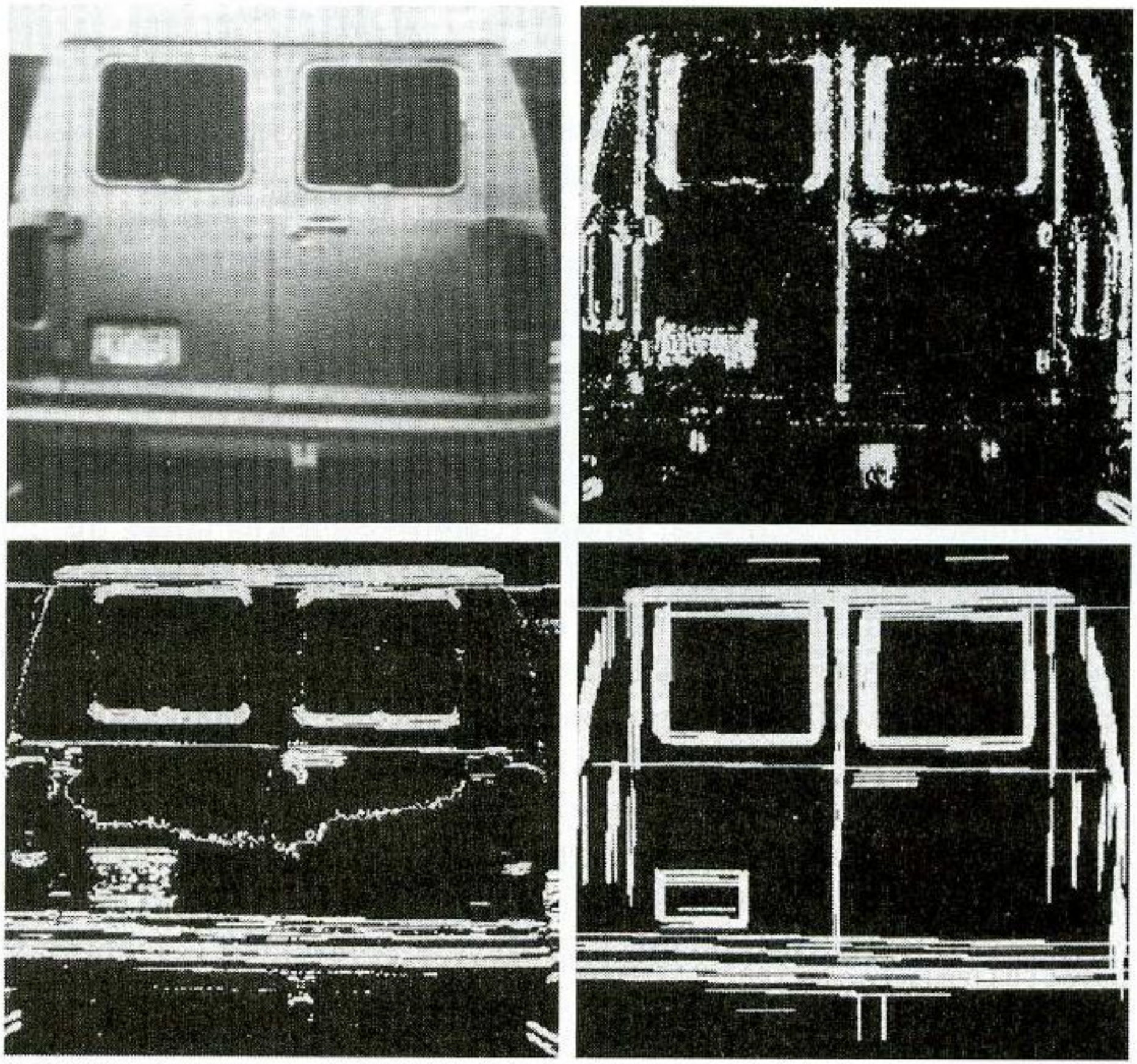
- An edge pixel at (x',y') in the neighborhood centered at (x,y) is similar in angle to the pixel at (x,y) if

$$\left|\alpha(x,y) - \alpha(x',y')\right| \leq A$$

- where A is a predetermined angle threshold
- A point in the neighborhood of (x,y) is linked to (x,y) if both magnitude and angle criteria are satisfied

a b
c d

**FIGURE 10.16**
(a) Input image.
(b) $G_y$ component of the gradient.
(c) $G_x$ component of the gradient.
(d) Result of edge linking. (Courtesy of Perceptics Corporation.)



$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

*magnitude* is

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}$$
$$\approx |G_x| + |G_y|$$

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

$$\left|\nabla f(x, y) - \nabla f(x', y')\right| \leq T$$

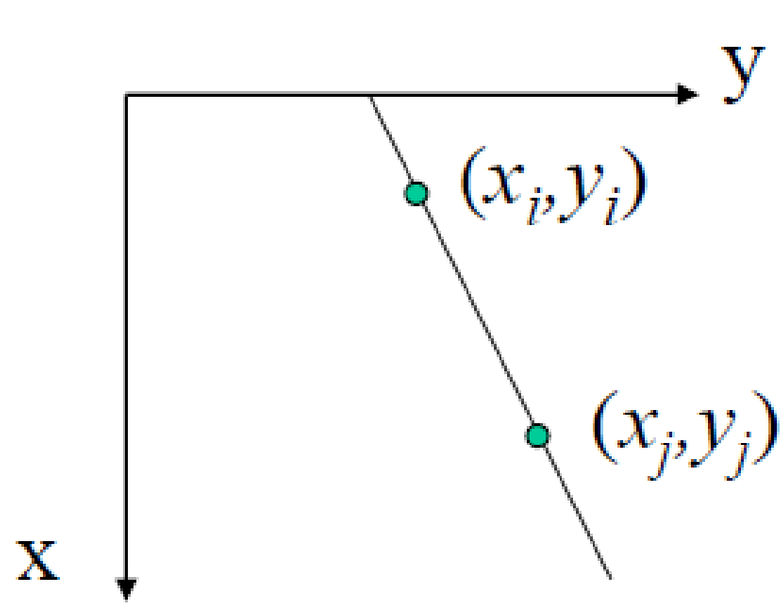
$$\left|\alpha(x, y) - \alpha(x', y')\right| \leq A$$

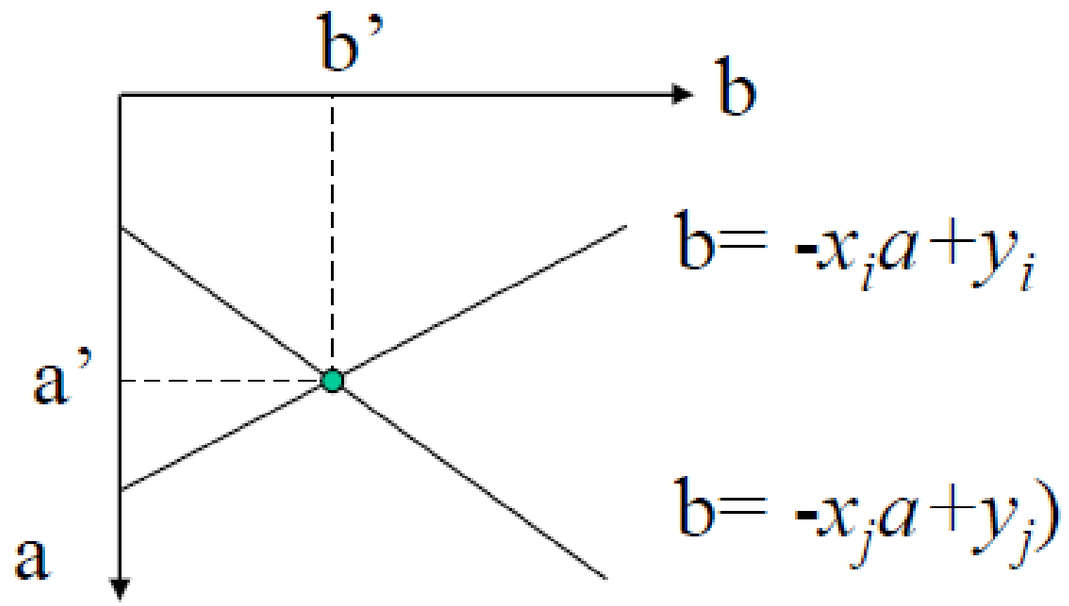
# Global processing : Hough transform

- Now consider global relationships between pixels
- Suppose that, for *n* points in an image, we want to find all subsets of these points that lie on straight lines
- Consider a point $(x_i, y_i)$ and the equation for a straight line $y_i = a x_i + b$
- Infinitely many lines pass through the point $(x_i, y_i)$, all satisfying the equation for varying values of *a* and *b*
- However, writing the equation as $b = -x_i a + y_i$ and considering the *ab* plane (also called the parameter space) yields the equation of a ***single*** line for a fixed point $(x_i, y_i)$

# Global processing : Hough transform

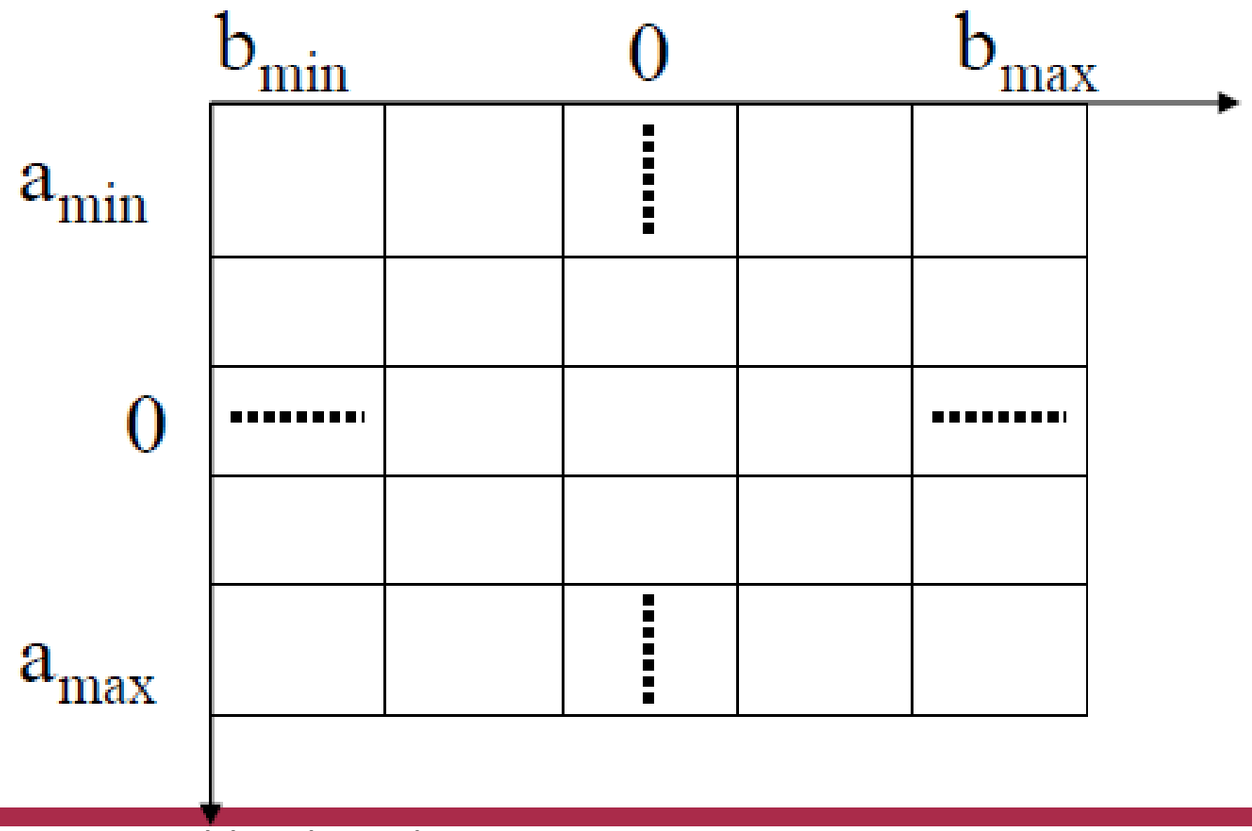
- A second point $(x_j,y_j)$ also has a line in the parameter space associated with it
- This line intersects the line associated with $(x_i,y_i)$ at $(a',b')$
  - $a'$ is the slope and $b'$ is the intercept of the line containing both $(x_i,y_i)$ and $(x_j,y_j)$ in the $xy$ plane
- In fact, all points that lie on this line have corresponding lines in the parameter space that intersect at $(a',b')$



# Hough transform : accumulator cells

- The computational attractiveness of the Hough transform arises from the subdivision of the parameter space into *accumulator cells*
- $(a_{min}, a_{max})$ and $(b_{min}, b_{max})$ are expected ranges of slope and intercept values



# Hough transform : accumulator cells

| | | | | | |
|---|---|---|---|---|---|
| **2** | 0 | 0 | 0 | 0 | 1 |
| **1** | 0 | 0 | 0 | 1 | 1 |
| **0** | 0 | 1 | 20 | 9 | 0 |
| **-1** | 1 | 2 | 0 | 10 | 0 |
| **-2** | 2 | 0 | 0 | 0 | 1 |
| **b/a** | **-90** | **-45** | **0** | **45** | **90** |

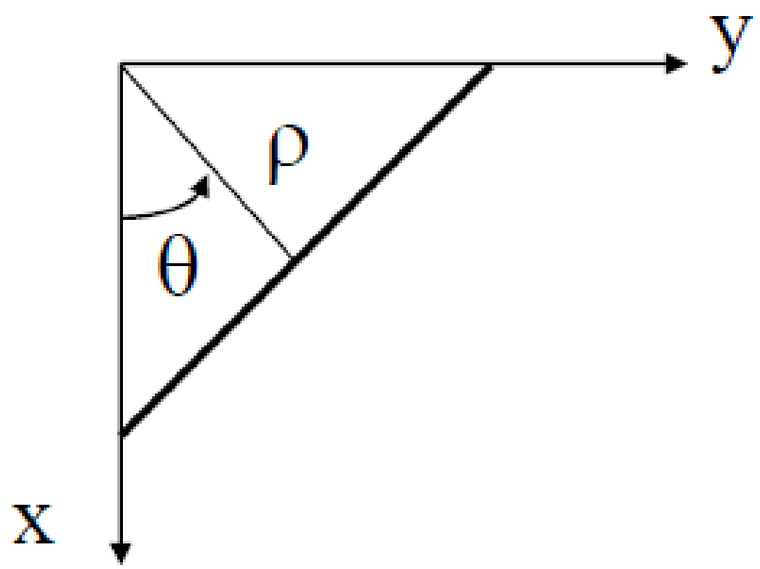
# Hough transform : accumulator cells

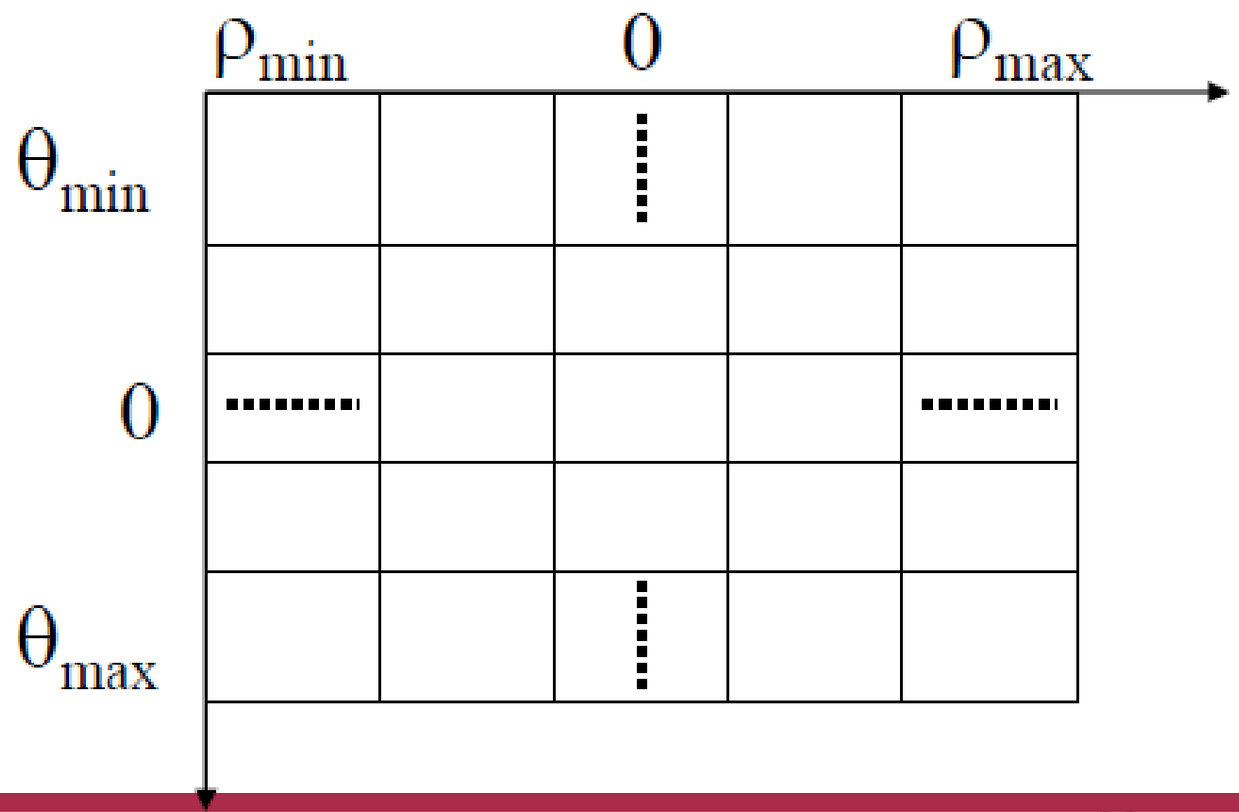
- The cell at coordinates (i,j), with cell value A(i,j), corresponds to the square associated with parameter space coordinates ($a_i$,$b_j$)
- The collection of accumulator cells is commonly called the Hough matrix (or Hough array) and are computed as
  1. Set all cells to zero.
  2. For every point ($x_k$,$y_k$) in the image plane, let *a* equal each of the allowed subdivision values on the *a* axis and solve for *b* using $b=-x_k a+y_k$
  3. The resulting b's are rounded off to the nearest allowed value in the b axis
  4. If a choice of $a_p$ results in solution $b_q$, we let A(p,q)=A(p,q)+1
  5. At the end of the procedure, a value of M in A(i,j) corresponds to M points in the xy plane lying on the line $y=a_i x+b_j$
- The accuracy of the collinearity of these points is determined by the number of subdivisions in the *ab* plane

# Hough transform

- A problem with this representation is that the slope and intercept approach infinity as the line approaches the vertical
- Solution: use the normal representation of a line given by

$$x\cos\theta + y\sin\theta = \rho$$



# Hough transform

- Instead of straight lines in the *ab* plane, we now have sinusoidal curves in the $\rho\theta$ plane
- M collinear points lying on the line

$$x\cos\theta_j + y\sin\theta_j = \rho_i$$

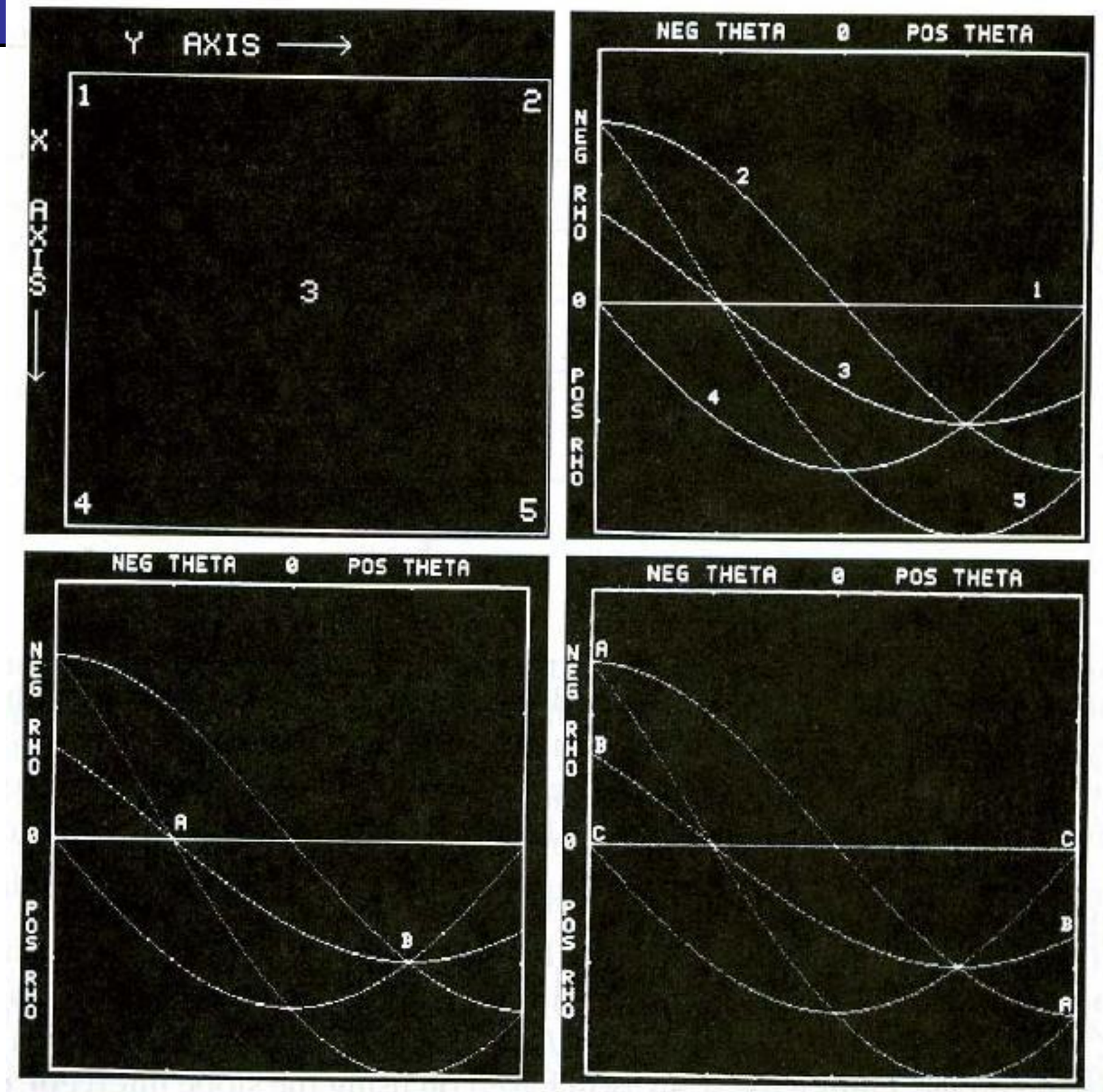
- yields M sinusoidal curves that intersect at $(\rho_i,\theta_j)$ in the parameter space
- The range of $\theta$ is ±90°, measured with respect to the x axis
  - A horizontal line has $\theta$=0°, with $\rho$ equal to the positive x intercept
  - A vertical line has $\theta$=+90°, with $\rho$ equal to the positive y intercept or $\theta$=-90°, with $\rho$ equal to the negative y intercept
- The range of $\rho$ is $\pm(2)^{1/2}D$. Where D is the distance between corners in the image

# Hough transform

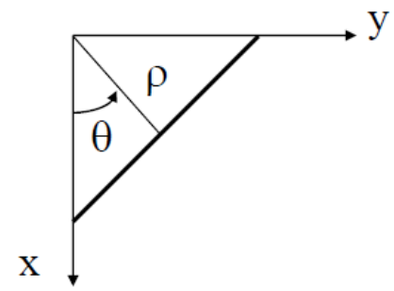
```
function res=hough(im,RHO_MAX,THETA_MAX)
%
%  Name: hough(im,RHO_MAX,THETA_MAX
%
%  Arguments
%  im: is the input,binary, image. If the
%  image is not binary pixels having
%  non-zero values are considered.
%  RHO_MAX: is an integer number specifying
%  the rho quantization.
%  THETA_MAX: is an integer number
%  specifying the theta quantization
%
% Example:  v=hough(im,256,256)
% input is the image im, and the
% quantization is d_rho=X/256 and d_theta=pi/256
% if the size of the image is 256 by 256
% d_rho=1.
%
% v is the number of votes in the
% parameter space. v(i,j) is the number
% of votes for the strip having distance from
% the center of the image equal to
% (i-RHO_MAX/2)*d_rho (d_rho=X/RHO_MAX, the
% image is X by X pixels),and its normal has
% angle j*d_theta,(d_theta=pi/THETA_MAX)
```

```
[X,Y]=size(im);
d_rho=X/RHO_MAX;
d_theta=pi/THETA_MAX;
theta=0:d_theta:pi-d_theta;
smat=sin(theta);
cmat=cos(theta);
fprintf('Finding feature points.\n');
[x,y]=find(im);
% translation by a pixel so that low left pixel has
% (0,0) coordinates
x=x-1;
y=y-1;
fprintf('Translating so the origin is in the middle of
    the image.\n');
fprintf('Doing the Hough Transform.\n');
h1=((y-Y/2) * smat + (x-X/2) * cmat )/d_rho;
h2=h1+RHO_MAX/2;
fprintf('Rounding.\n');
h3=round(h2);
fprintf('Summing the votes.\n');
res=zeros(RHO_MAX,THETA_MAX);
for j=0:RHO_MAX-1
  temp=(h3==j);
  res(j+1,:)=sum(temp);
end
```

# Hough transform

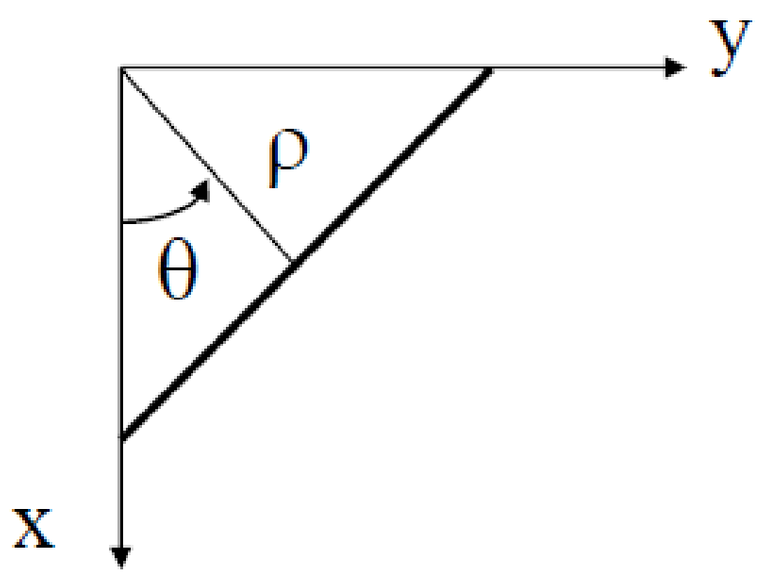


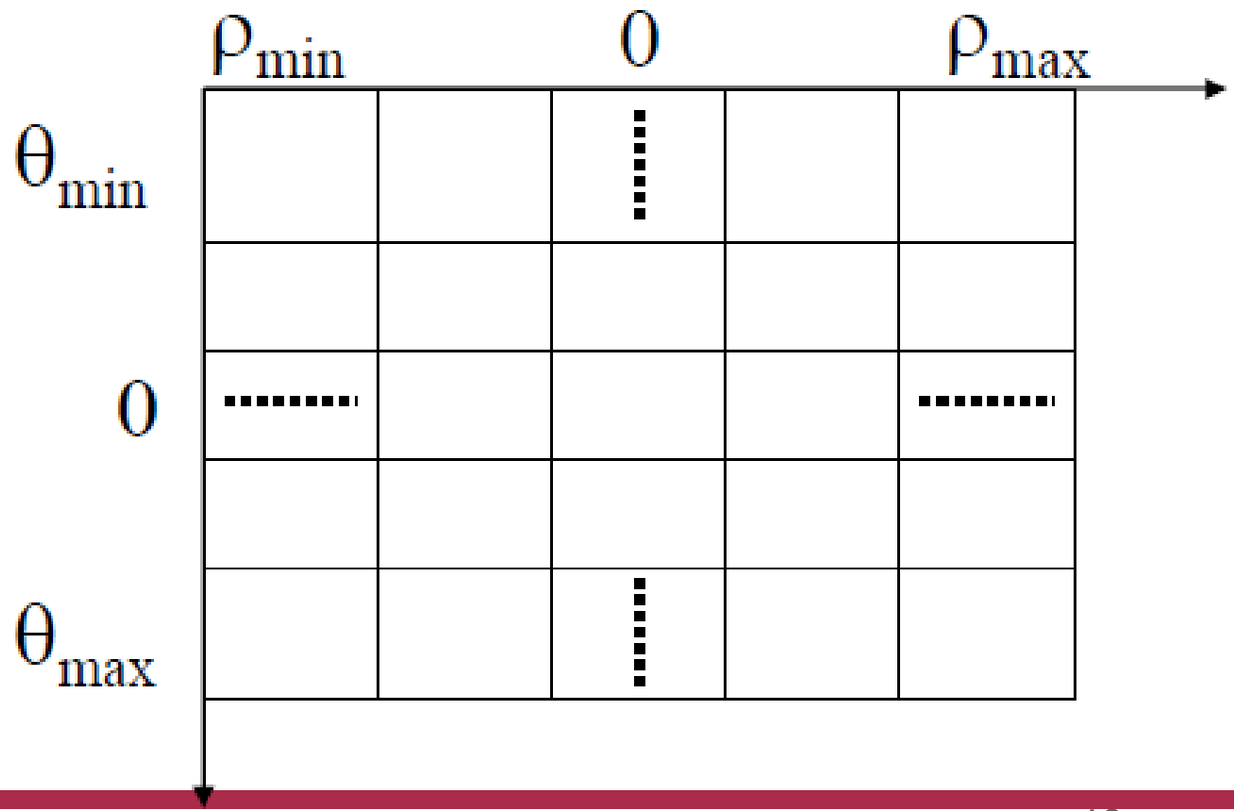

$$x\cos\theta + y\sin\theta = \rho$$



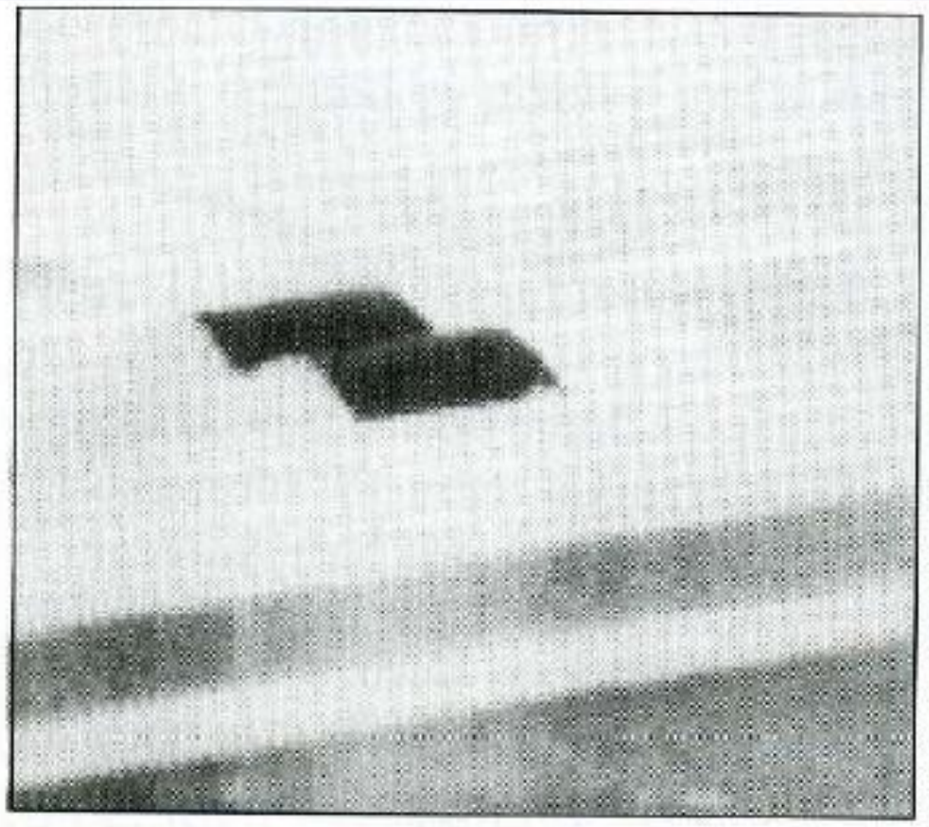
# Hough transform

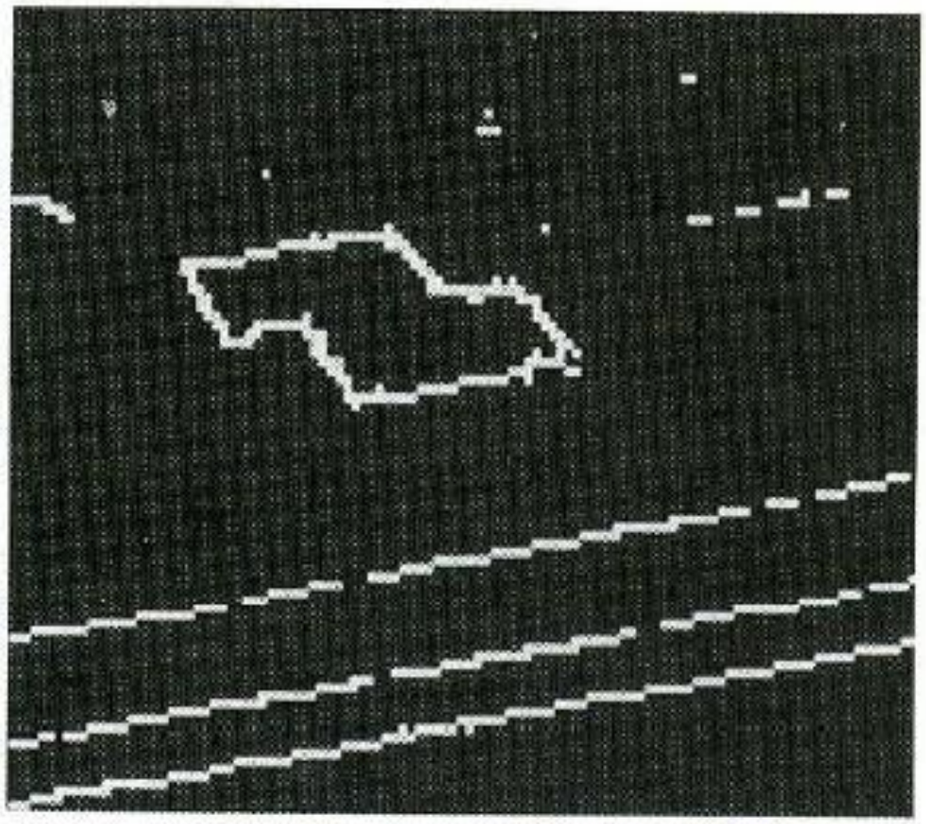
1. Compute the gradient of an image and threshold it to obtain a binary image.
2. Specify subdivisions in the $\rho\theta$-plane.
3. Examine the counts of the accumulator cells for high pixel concentrations.
4. Examine the relationship (principally for continuity) between pixels in a chosen cell.

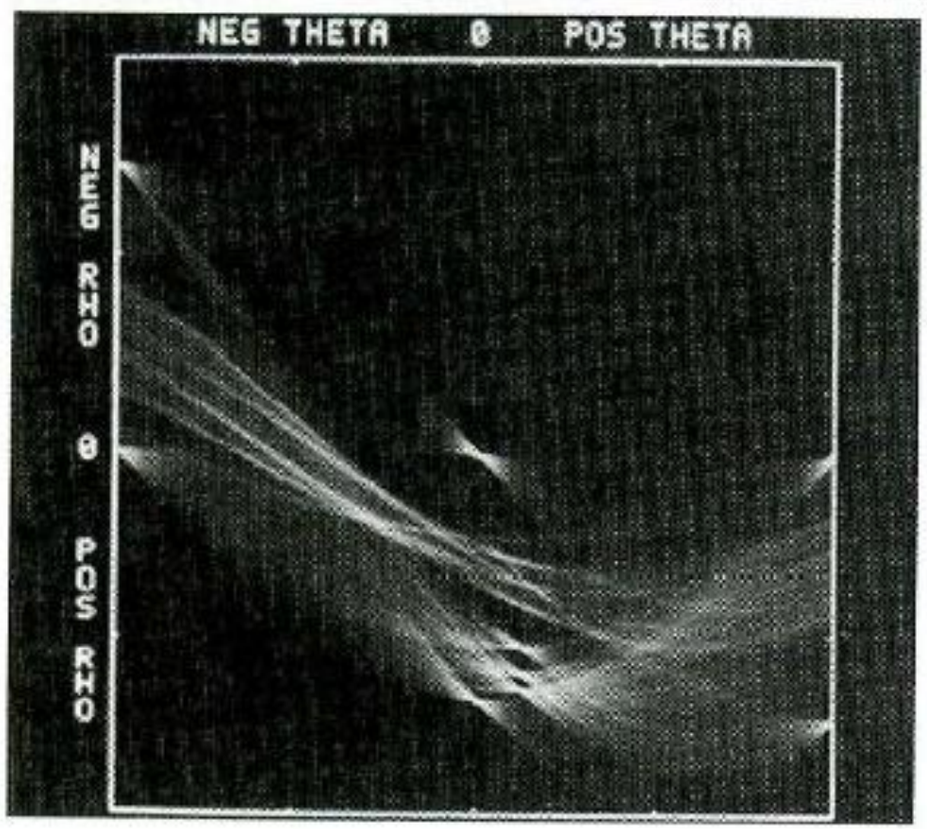


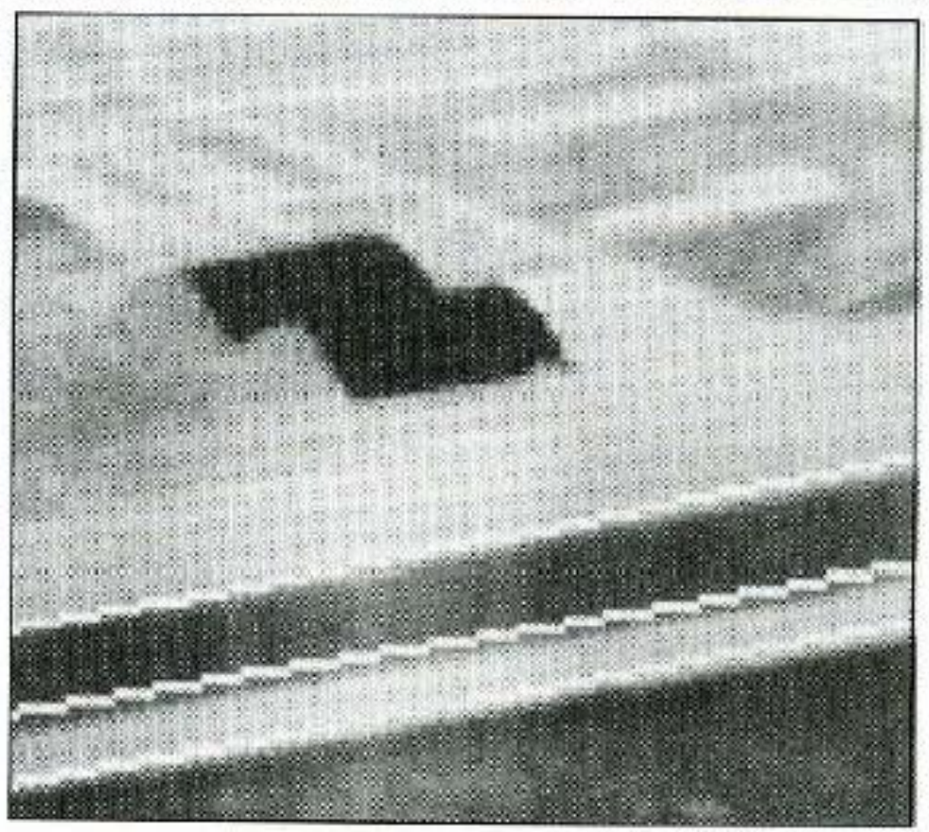



# Hough transform







a b
c d

**FIGURE 10.21**
(a) Infrared image.
(b) Thresholded gradient image.
(c) Hough transform.
(d) Linked pixels.
(Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

Ima

# Summary

In this lecture we have begun looking at segmentation, and in particular edge detection

Edge detection is massively important as it is in many cases the first step to object recognition

# Image Processing and Pattern Recognition (IPPR)

## Chapter 7:Image Segmentation
## Similarity Based Approaches


Basanta Joshi, PhD

Asst. Prof., Depart of Electronics and Computer Engineering

Member, Laboratory for ICT Research and Development (LICT)

Institute of Engineering

basanta@ioe.edu.np

http://www.basantajoshi.com.np

https://scholar.google.com/citations?user=iocLiGcAAAAJ
https://www.researchgate.net/profile/Basanta_Joshi2


# Contents

Today we will continue to look at the problem of segmentation, this time though in terms of thresholding

In particular we will look at:

– What is thresholding?
– Simple thresholding
– Adaptive thresholding
– Region-Based Segmentation

# Thresholding

Thresholding is usually the first step in any segmentation approach

We have talked about simple single value thresholding already

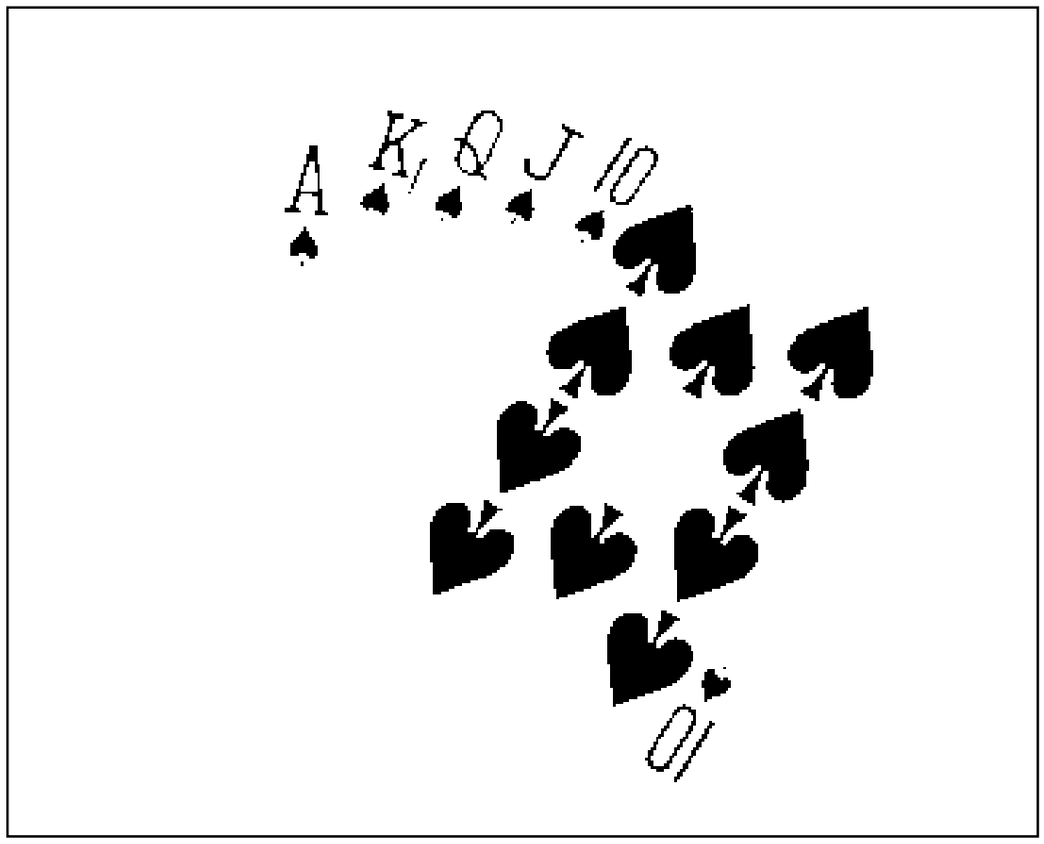
Single value thresholding can be given mathematically as follows:

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

# Thresholding Example

Imagine a poker playing robot that needs to visually interpret the cards in its hand

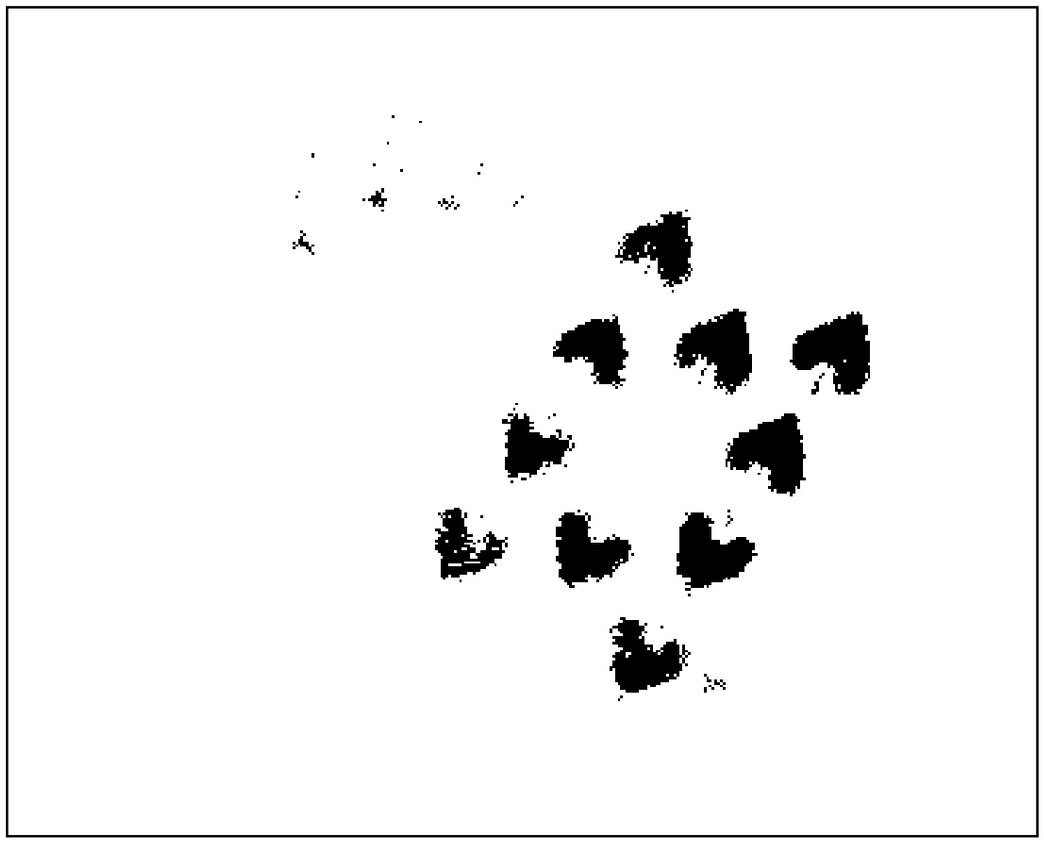
Original Image



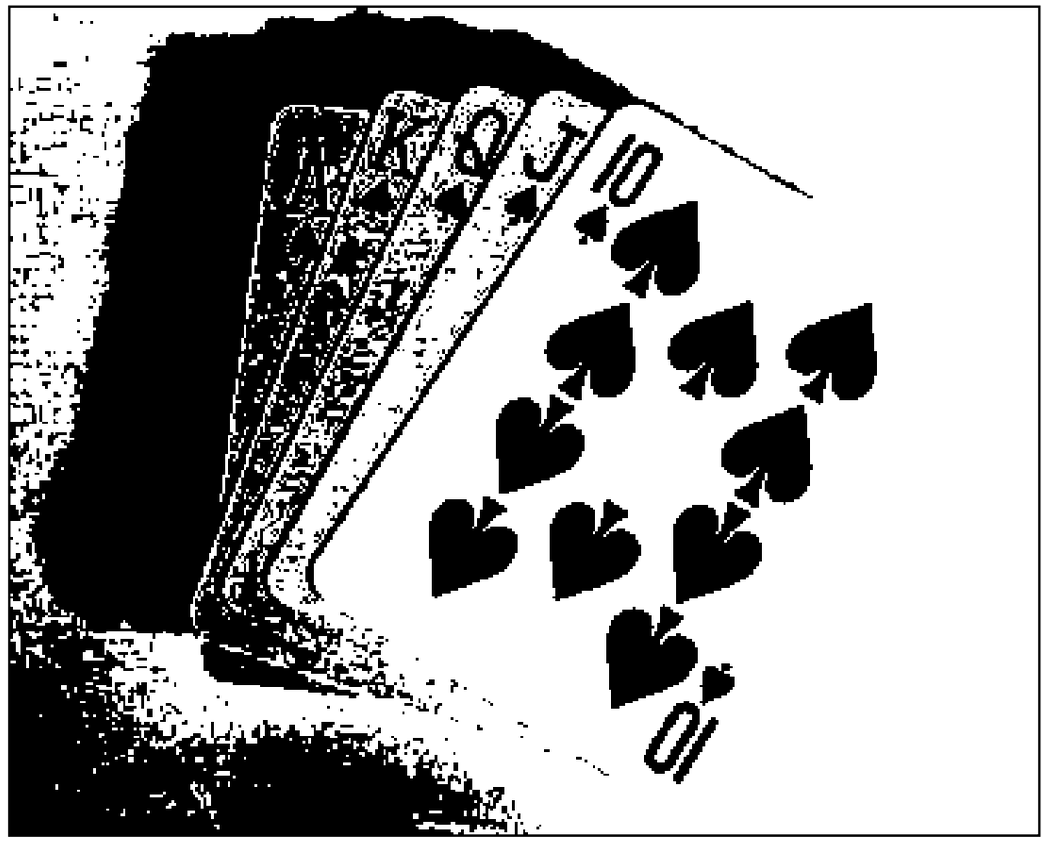
Thresholded Image

# But Be Careful

If you get the threshold wrong the results can be disastrous



Threshold Too Low



Threshold Too High

# Basic Global Thresholding

Based on the histogram of an image

Partition the image histogram using a single global threshold

The success of this technique very strongly depends on how well the histogram can be partitioned

# Basic Global Thresholding Algorithm

The basic global threshold, T, is calculated as follows:

1. Select an initial estimate for T (typically the average grey level in the image)
2. Segment the image using T to produce two groups of pixels: $G_1$ consisting of pixels with grey levels >T and $G_2$ consisting pixels with grey levels ≤ T
3. Compute the average grey levels of pixels in $G_1$ to give $\mu_1$ and $G_2$ to give $\mu_2$

# Basic Global Thresholding Algorithm

4. Compute a new threshold value:

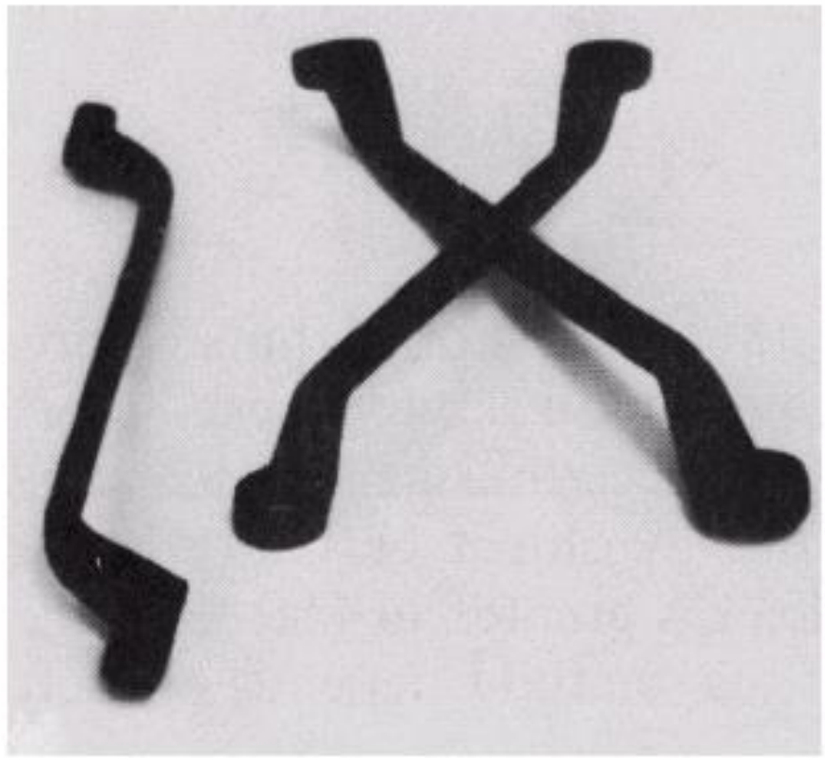
$$T = \frac{\mu_1 + \mu_2}{2}$$

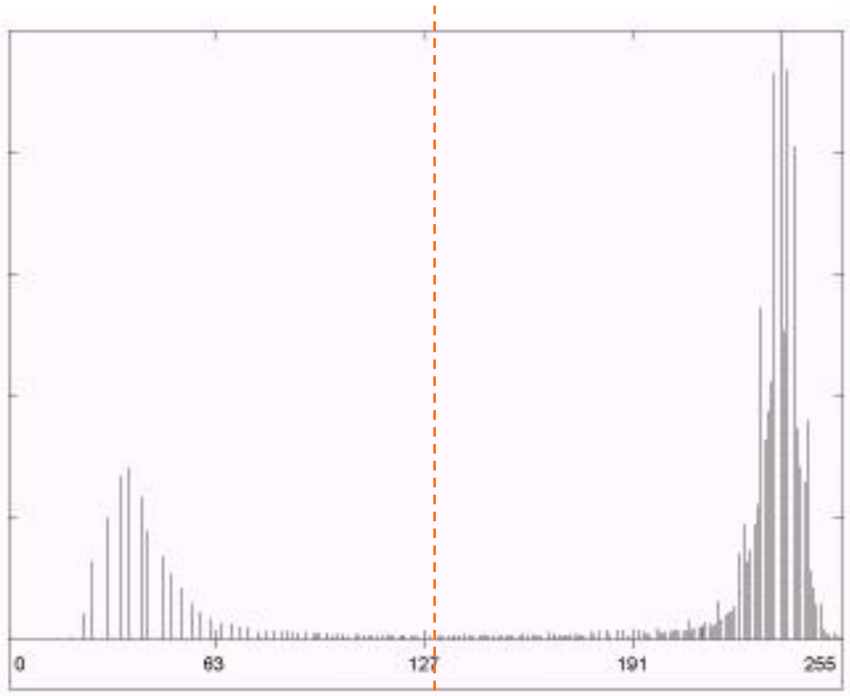
5. Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit $T_\infty$

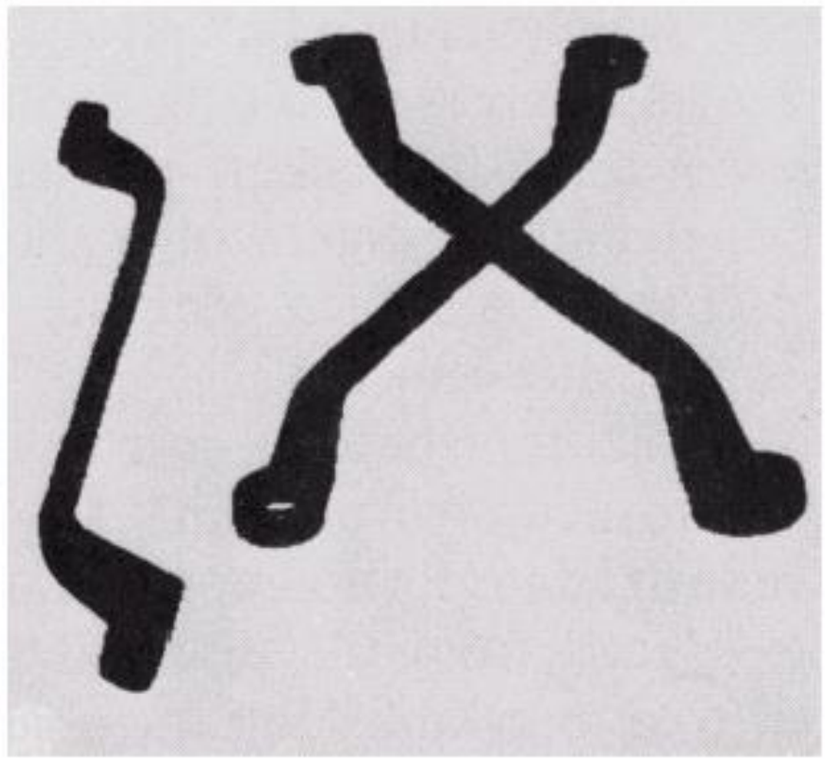
This algorithm works very well for finding thresholds when the histogram is suitable

# Thresholding Example 1

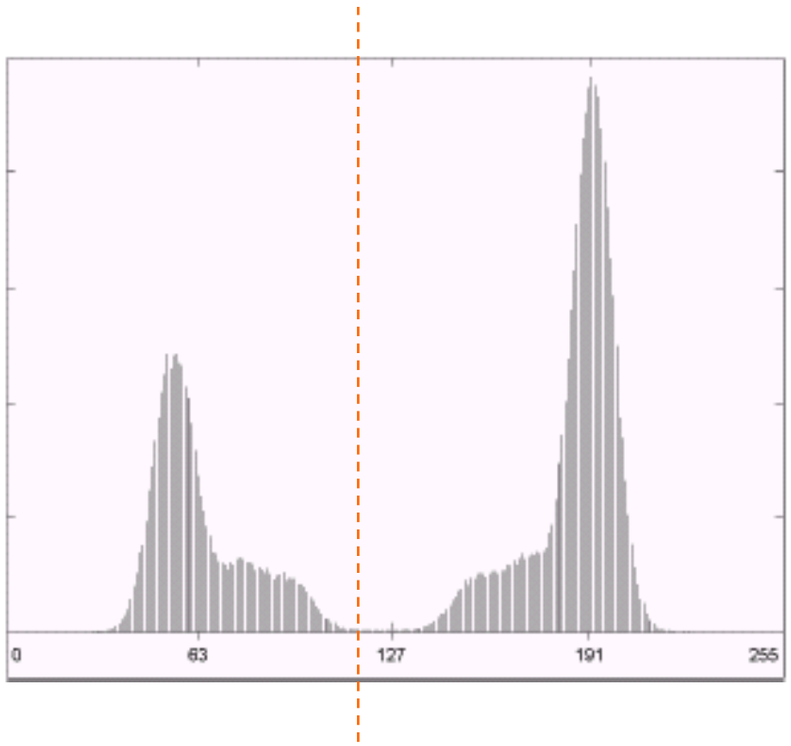


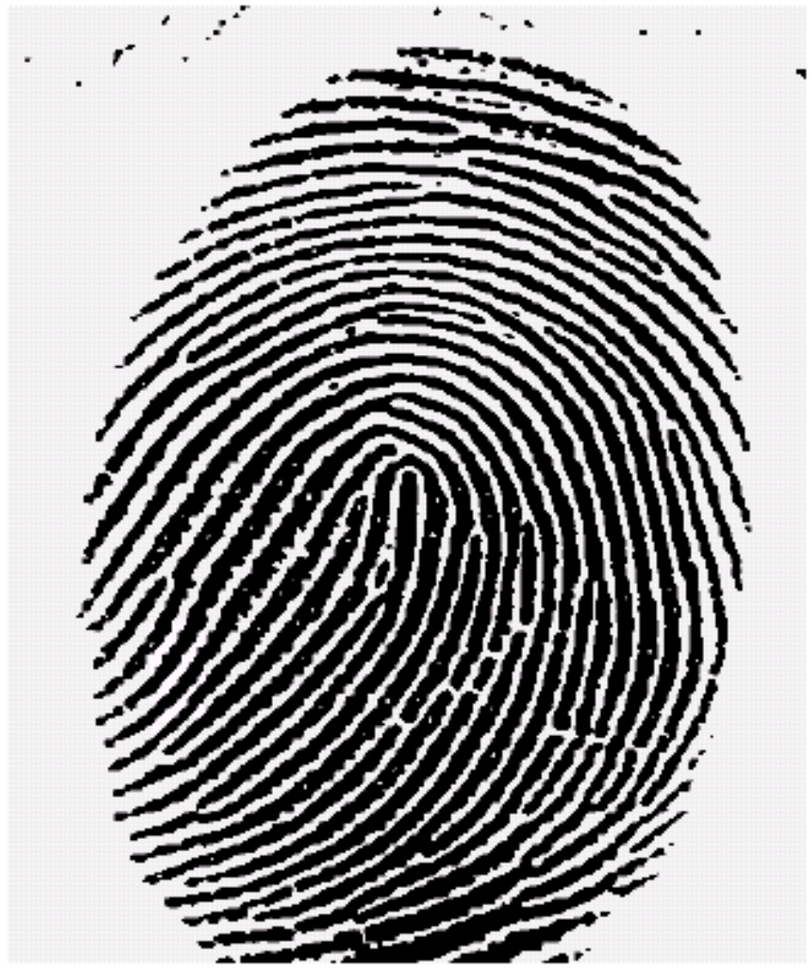




# Thresholding Example 2

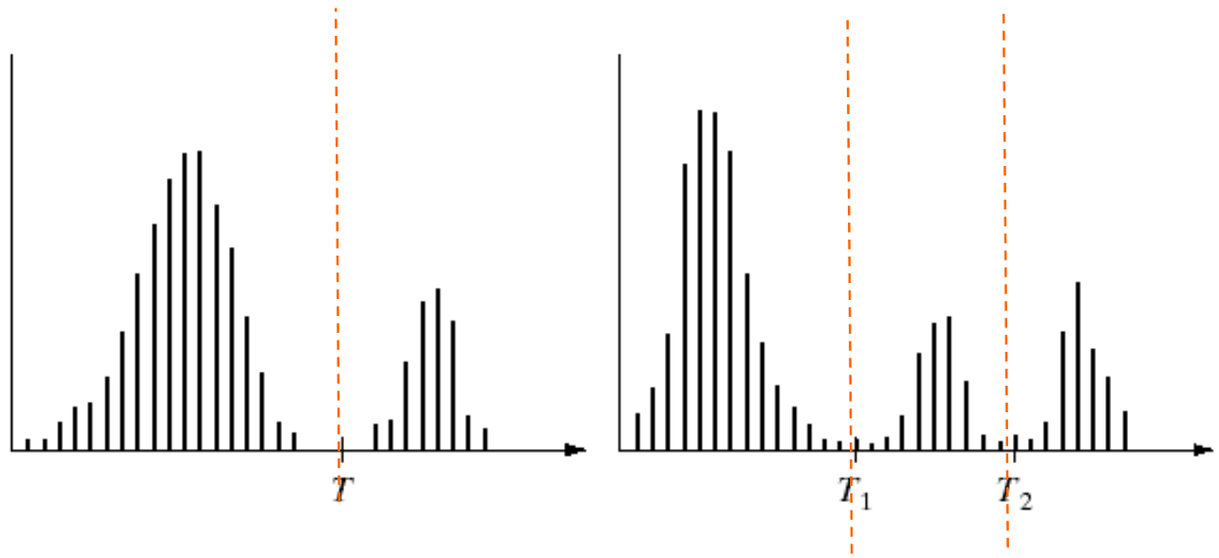


# Problems With Single Value Thresholding

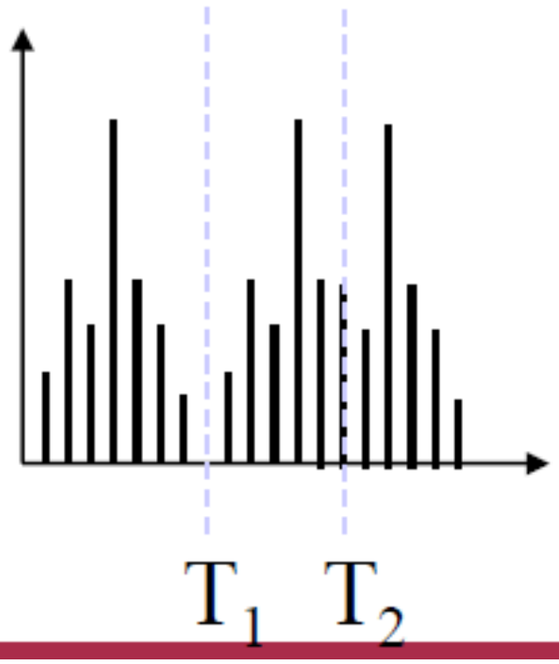
Single value thresholding only works for bimodal histograms

Images with other kinds of histograms need more than a single threshold



# Thresholding (cont…)

- Suppose several objects with differing gray levels (with a dark background) comprise the image
- An object may be classified as belonging to one object class if $T_1 < f(x,y) \leq T_2$, to a second class if $f(x,y) > T_2$ or to the background if $f(x,y) \leq T_1$
- This, however, is generally less reliable than single level thresholding



# Thresholding (cont…)

- Thresholding may be viewed as an operation that tests against a given function of the form

$$T = T[x, y, p(x, y), f(x, y)]$$

- where $f(x,y)$ is the gray level of point $(x,y)$ and $p(x,y)$ is some local property of the point -- the average gray level of a neighborhood around $(x,y)$
- The thresholded image is given by

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

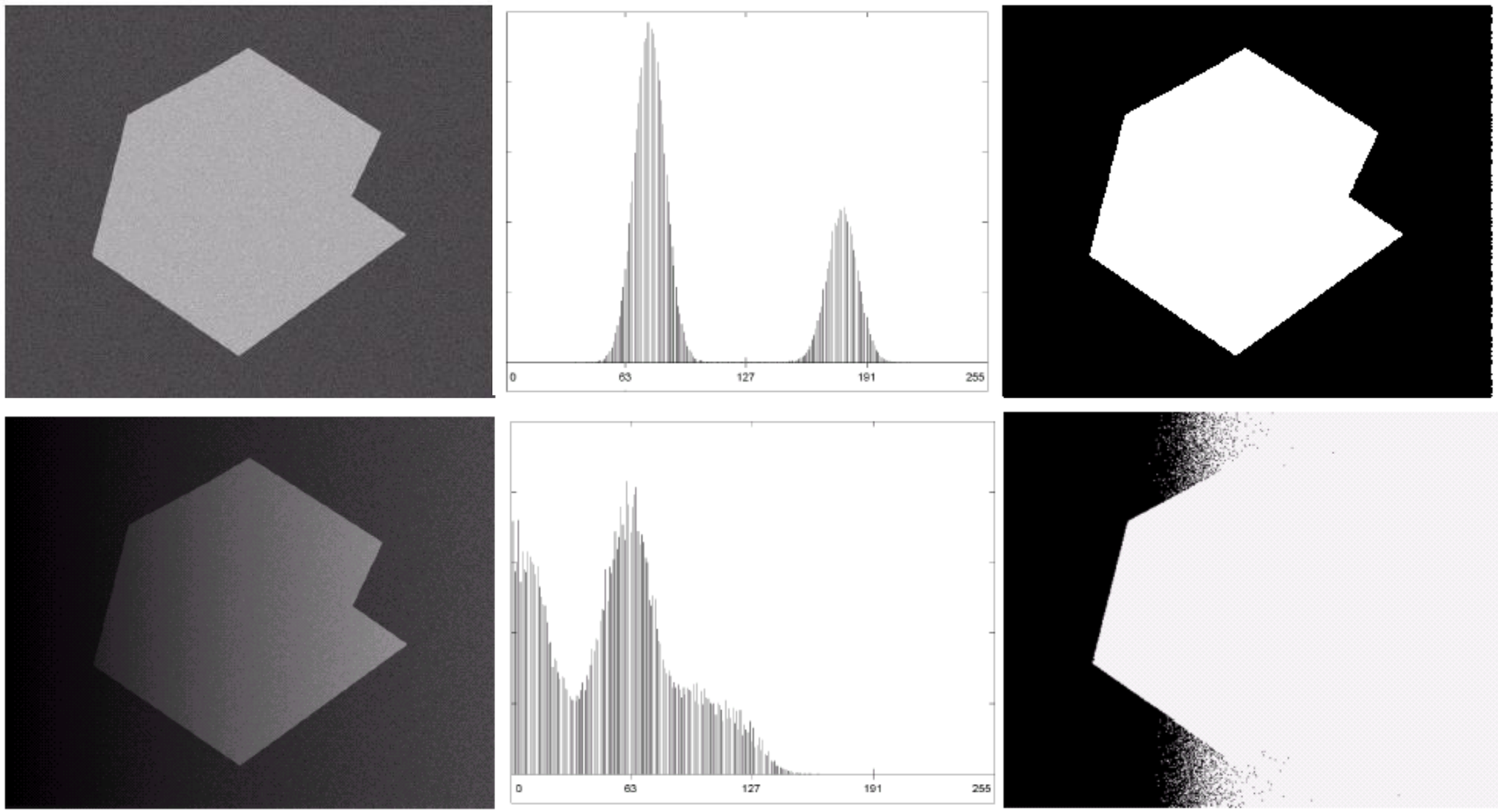
- Pixels labeled 1 (or any other convenient gray level value) correspond to objects

# Thresholding (cont…)

- When *T* depends only on *f(x,y)* the threshold is called *global*
- If *T* depends on *f*(*x,y*) and *p*(*x,y*) the threshold is *local*
- If, in addition, *T* depends on the spatial coordinates (*x,y*), the threshold is called *dynamic*

- For example, a local threshold may be used if certain information about the nature of the objects in the image is known a priori
- A dynamic threshold may be used in the case where object illumination is non-uniform

# Single Value Thresholding and Illumination




Uneven illumination can really upset a single valued thresholding scheme

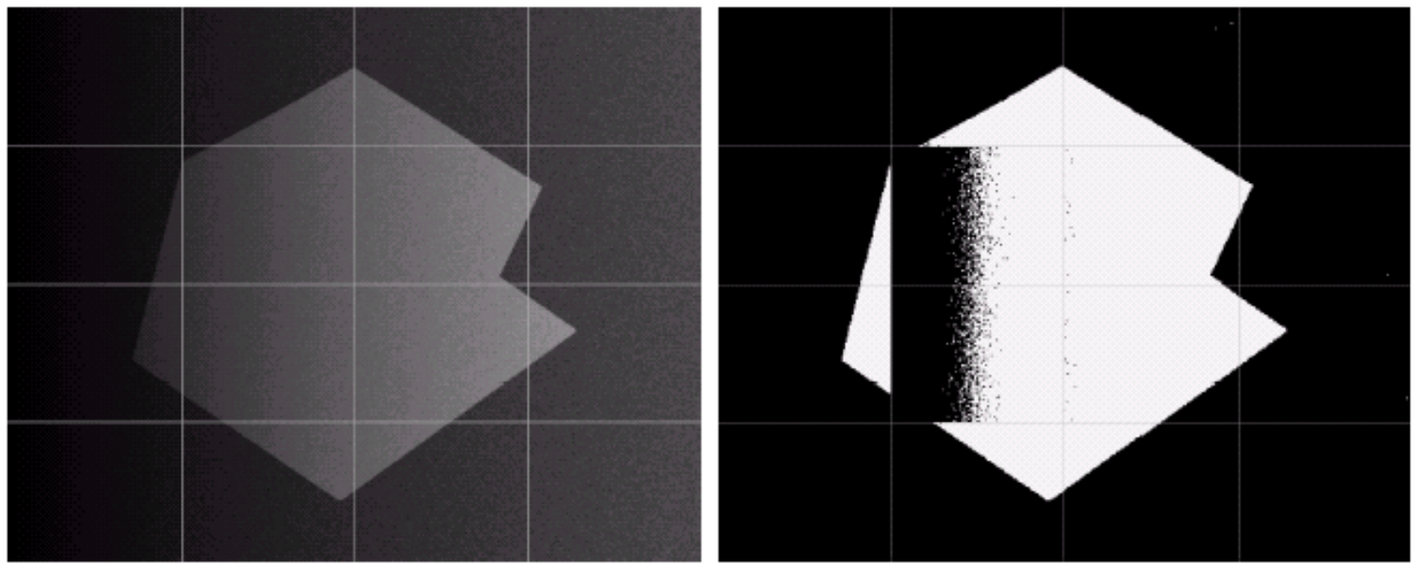
# Basic Adaptive Thresholding

An approach to handling situations in which single value thresholding will not work is to divide an image into sub images and threshold these individually

Since the threshold for each pixel depends on its location within an image this technique is said to *adaptive*

# Basic Adaptive Thresholding Example

The image below shows an example of using adaptive thresholding with the image shown previously



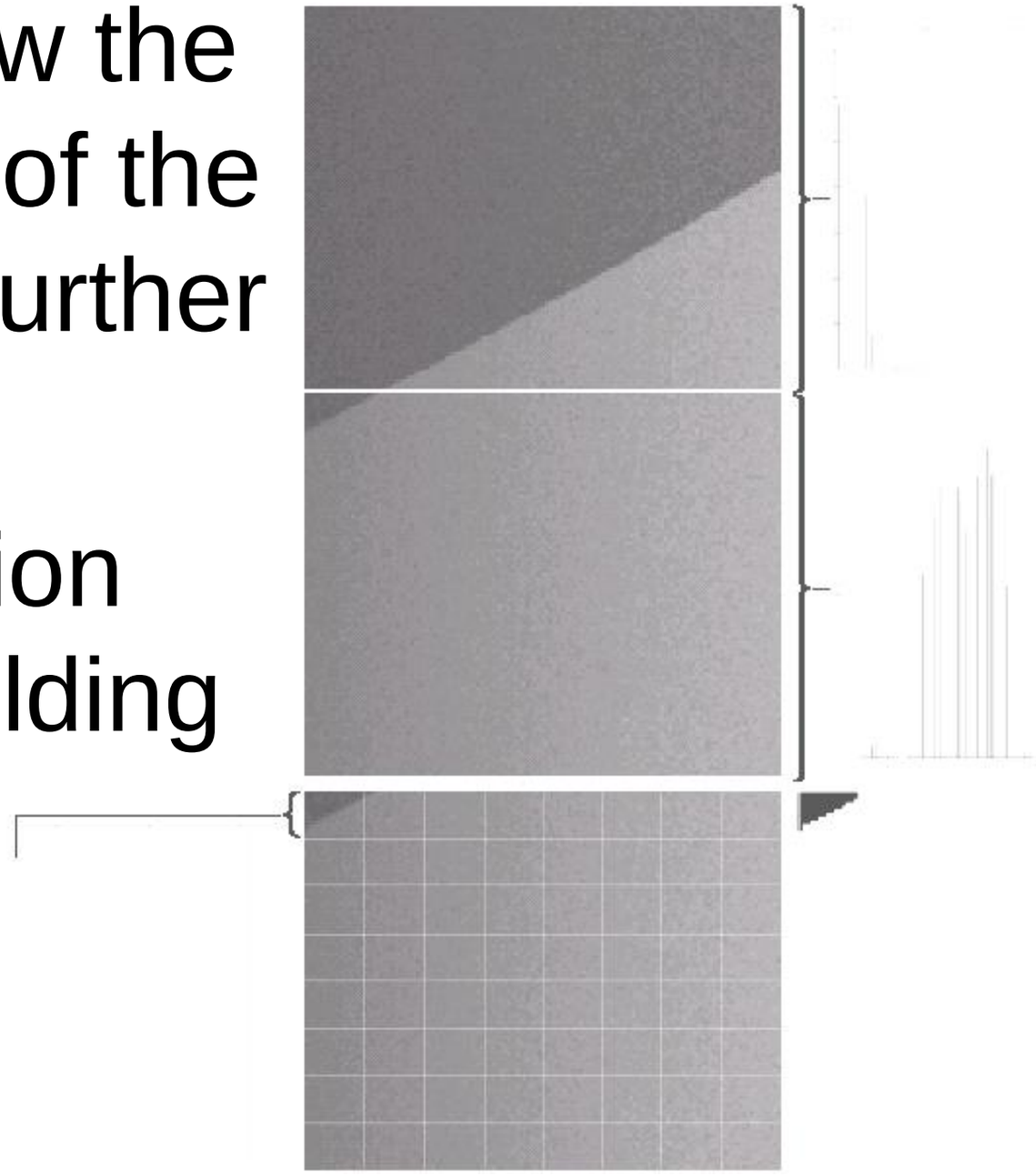
As can be seen success is mixed

But, we can further subdivide the troublesome sub images for more success

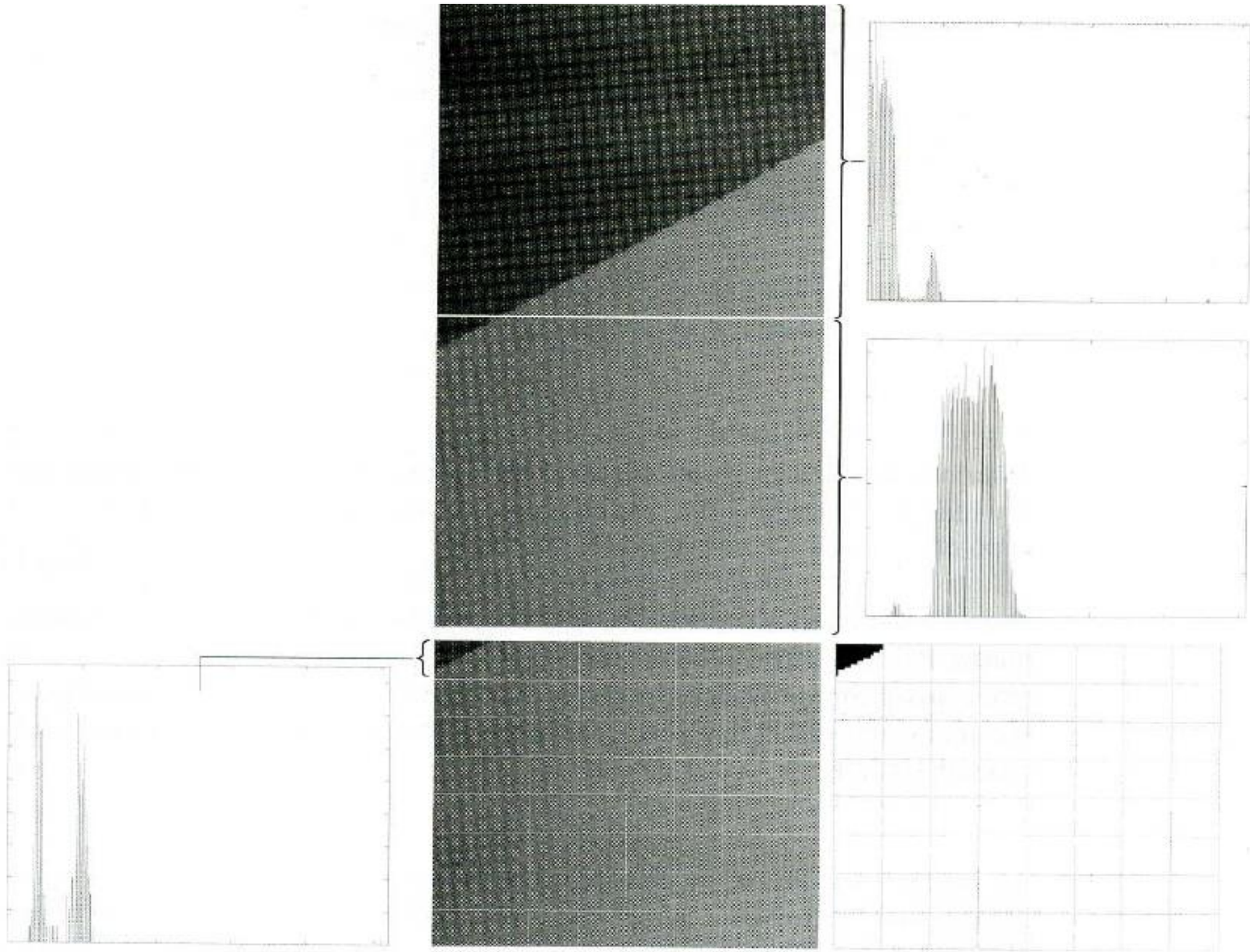
# Basic Adaptive Thresholding Example (cont…)

These images show the troublesome parts of the previous problem further subdivided

After this sub division successful thresholding can be achieved



# Basic Adaptive Thresholding Example (cont…)



# Thresholding based on boundaries

- Important aspect of threshold selection: the ability to reliably identify mode peaks in a given histogram
- The chances of selecting a “good” threshold are enhanced if mode peaks are
  - tall
  - narrow
  - symmetric
  - and separated by deep valleys
- One approach for improving the histogram shape is to consider only those pixels that lie on or near a boundary between objects and the background

# Thresholding based on boundaries

- An obvious advantage is that the histogram becomes less dependent on the size of objects in the image
- By choosing pixels on or near object boundaries ( assuming an equal probability of choosing a pixel on the object or boundary) the histogram peaks tend to be made more symmetric
- Using pixels that satisfy some simple measures based on the gradient and Laplacian operators tends to deepen the valleys between histogram peaks

# Thresholding based on boundaries

- Determining if a pixel lies on a boundary: compute the gradient
- Determining what side, background (dark) or object (light), a pixel lies on: compute the Laplacian
- Using the gradient and Laplacian, a three-level image may be formed according to

$$s(x,y) = \begin{cases} 0 \text{ if } \nabla f < T \\ + \text{ if } \nabla f \geq T \text{ and } \nabla^2 f \geq 0 \\ - \text{ if } \nabla f \geq T \text{ and } \nabla^2 f < 0 \end{cases}$$

- where 0, + and - are three distinct gray levels

# Thresholding based on boundaries

- For a dark object on a light background, s(x,y) is produced where
  - all pixels not on an edge are labeled 0
  - all pixels on the dark side of an edge are labeled +
  - all pixels on the light side of an edge are labeled -
- For a light object on a dark background, s(x,y) is produced where
  - all pixels not on an edge are labeled 0
  - all pixels on the dark side of an edge are labeled -
  - all pixels on the light side of an edge are labeled +

# Thresholding based on boundaries

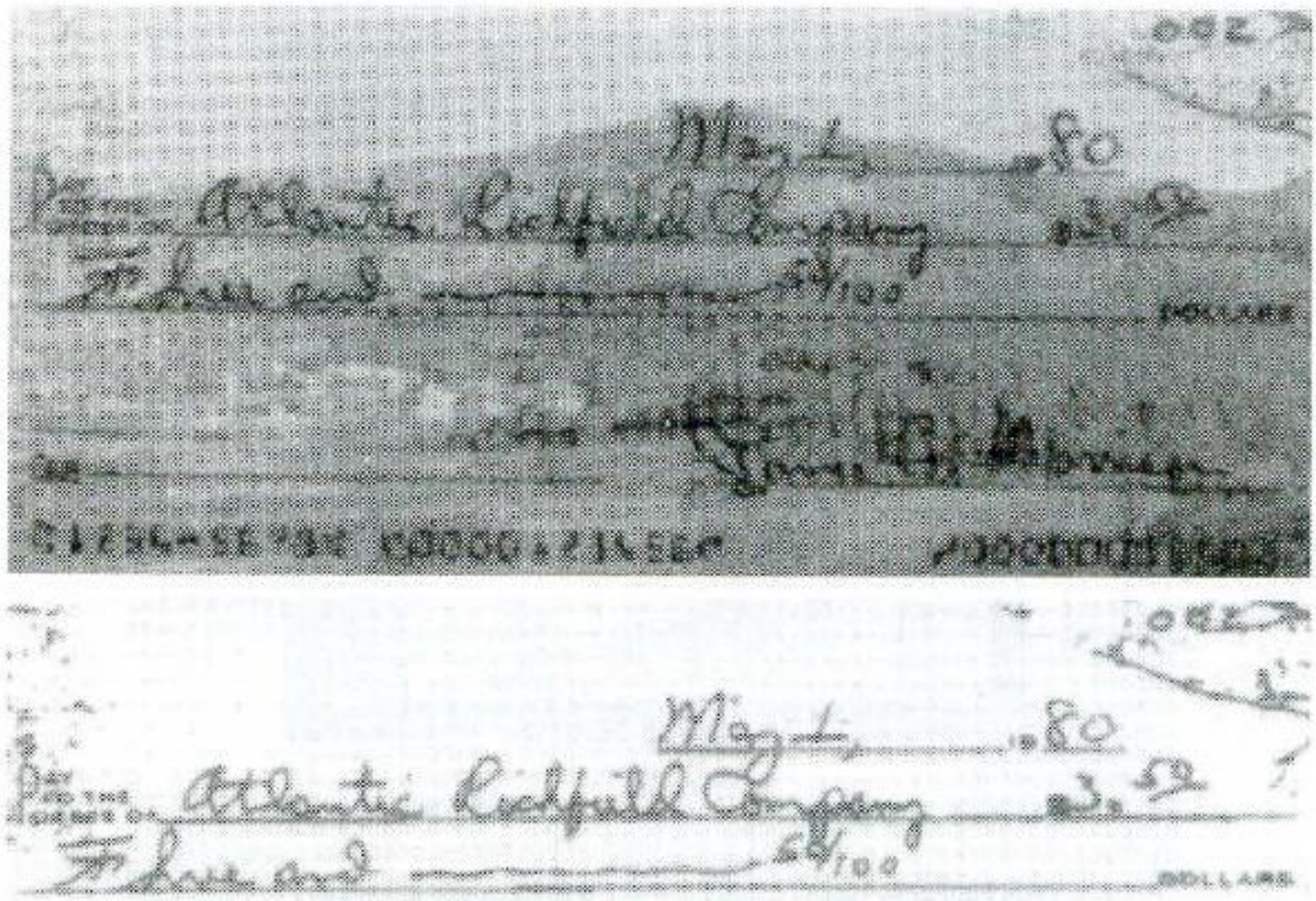
The information obtained with this procedure can be used to generate a segmented, binary image in which 1's correspond to objects of interest and 0's correspond to the background. The transition (along a horizontal or vertical scan line) from a light background to a dark object must be characterized by the occurrence of a − followed by a + in $s(x, y)$. The interior of the object is composed of pixels that are labeled either 0 or +. Finally, the transition from the object back to the background is characterized by the occurrence of a + followed by a −. Thus a horizontal or vertical scan line containing a section of an object has the following structure:

$$(\cdots)(-, +)(0 \text{ or } +)(+, -)(\cdots)$$

# Thresholding based on boundaries

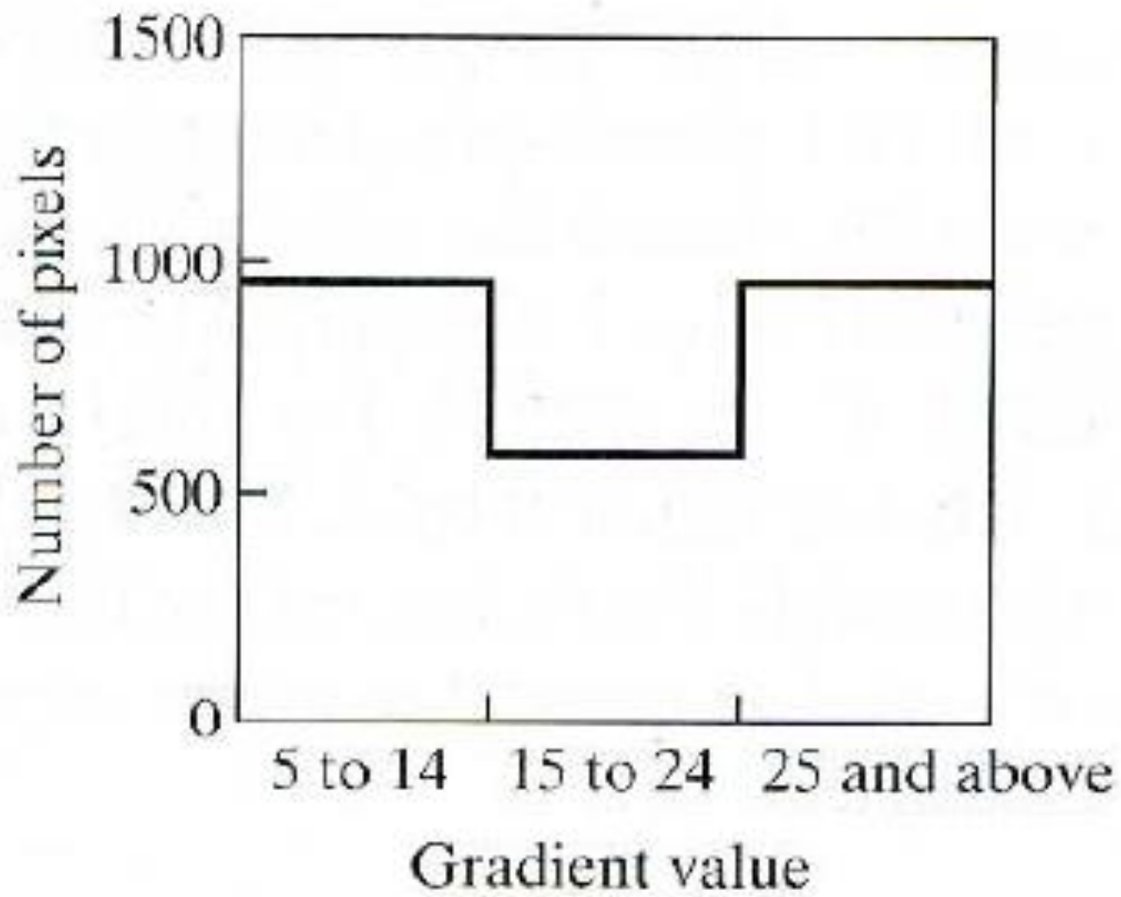
a
b

**FIGURE 10.37**
(a) Original image. (b) Image segmented by local thresholding. (Courtesy of IBM Corporation.)



# Thresholding based on boundaries

**FIGURE 10.38** Histogram of pixels with gradients greater than 5. (Courtesy of IBM Corporation.)



# Region-Based Segmentation

Edges and thresholds sometimes do not give good results for segmentation.

Region-based segmentation is based on the connectivity of similar pixels in a region.

- Each region must be uniform.
- Connectivity of the pixels within the region is very important.

There are two main approaches to region-based segmentation: region growing and region splitting.

# Region-Based Segmentation Basic Formulation

Let $R$ represent the entire image region.

Segmentation is a process that partitions $R$ into subregions, $R_1, R_2, \ldots, R_n$, such that

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

$$\text{(b)}\ R_i \text{ is a connected region, } i = 1,2,\ldots,n$$

$$\text{(c)}\ R_i \cap R_j = \phi \text{ for all } i \text{ and } j, i \neq j$$

$$\text{(d)}\ P(R_i) = \text{TRUE for } i = 1,2,\ldots,n$$

$$\text{(e)}\ P(R_i \cup R_j) = \text{FALSE for any adjacent regions } R_i \text{ and } R_j$$

where $P(R_k)$: a logical predicate defined over the points in set $R_k$

For example: $P(R_k)$=TRUE if all pixels in $R_k$ have the same gray level.

# Region-Based Segmentation Basic Formulation

❖Condition (a) indicates that the segmentation must be complete, i.e., every pixel must be in a region.

❖ Condition (b) requires that points in a region must be connected.

❖ Condition (c) indicates that the regions must be disjoint.

❖ Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region – for example $P(R_i)$ = TRUE if all pixels in $R_i$ have the same intensity.

❖ Condition (e) indicates that regions $R_i$ and $R_j$ are different in the sense of predicate P.

# Region-Based Segmentation
# Region Growing

- Groups pixels or subregions into larger regions based on predefined criteria (gray tone or texture).
- Basic method:

  Start with a set of "seed" points and from these grow regions by appending to each seed those neighboring pixels that have properties similar to the seed, such as specific ranges of gray level.

- Problems in region growing:

(a) Selection of the seeds

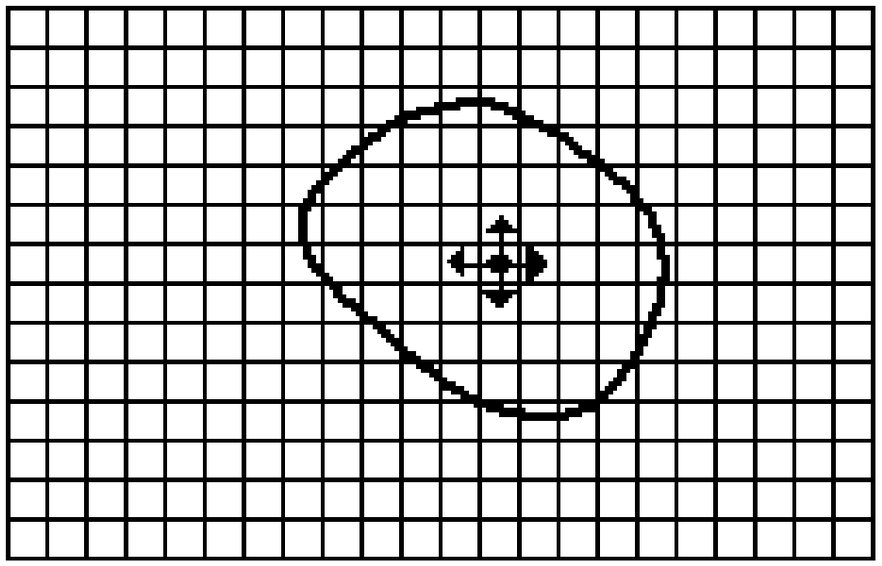
(b) Criteria of similarity

Gray level's similarity / connectivity / texture / moments

(c) Formulation of a stopping rule

Growing a region should stop when no more pixels satisfy the criteria for inclusion in that region

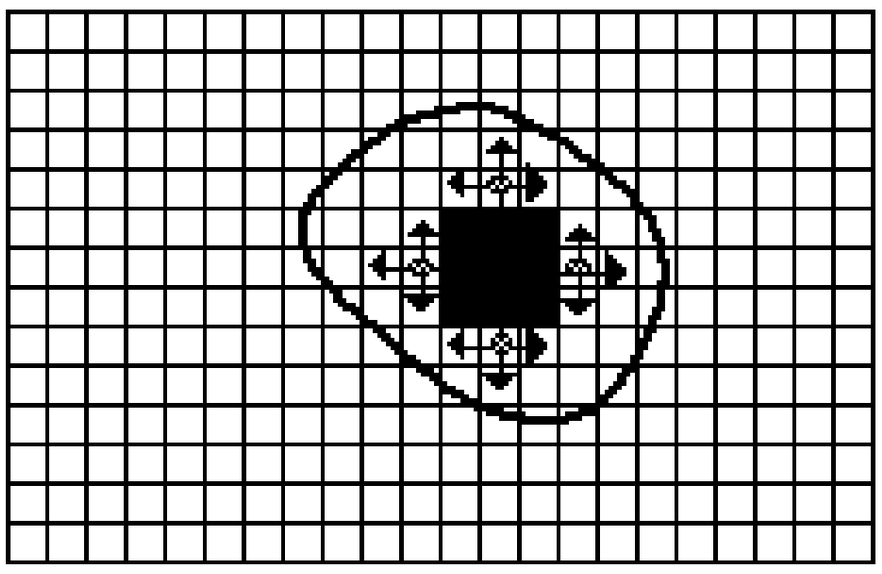
# Region-Based Segmentation
# Region Growing



Seed Pixel

Direction of Growth

(a) Start of Growing a Region

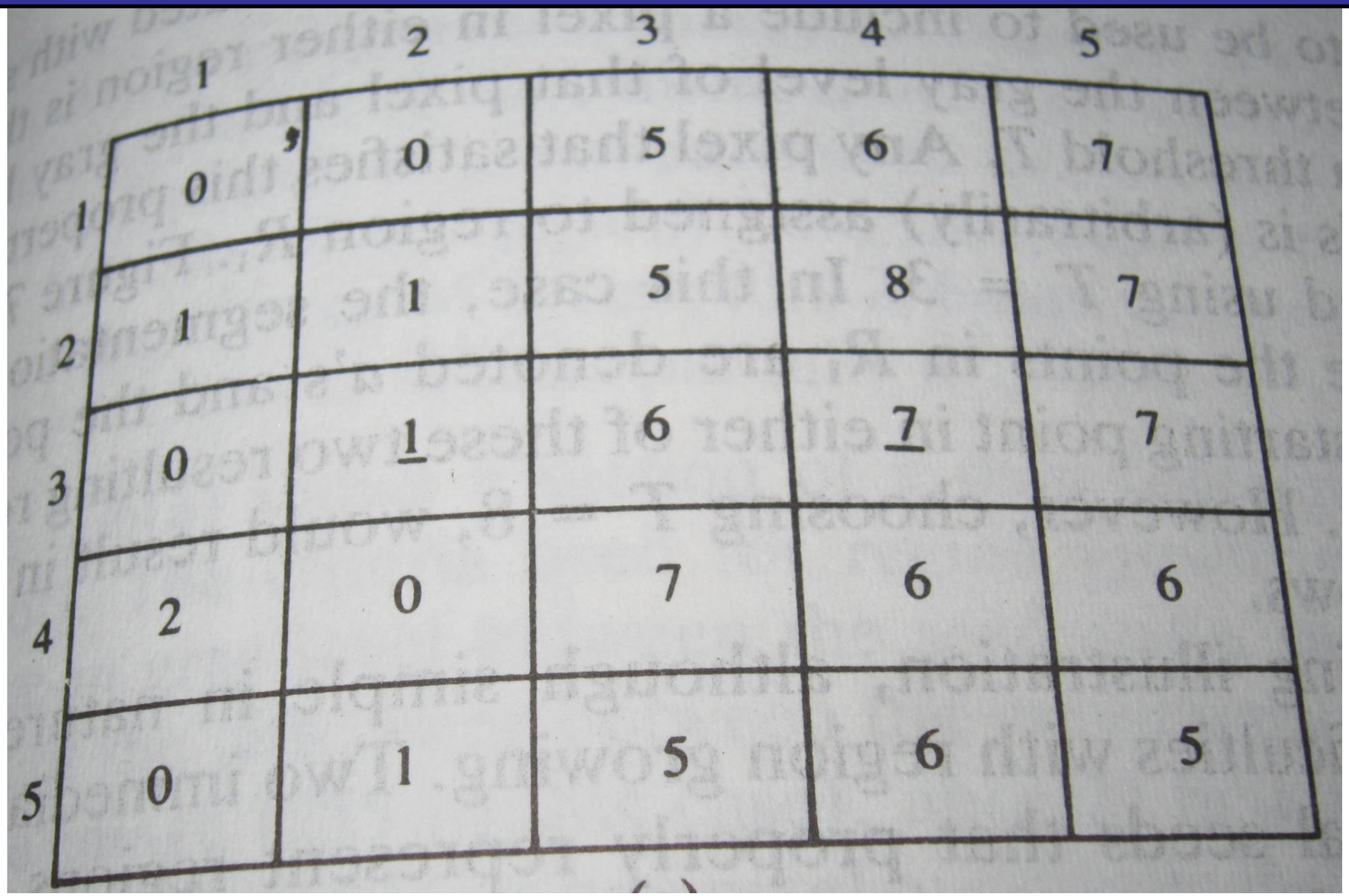


Grown Pixels

Pixels Being Considered

(b) Growing Process After a Few Iterations

# Region-Based Segmentation
# Basic Formulation




## Criteria

Property P to be used to include a pixel in either region is that the absolute difference between the gray level of that pixel and the gray level of the seed be less that a threshold T.

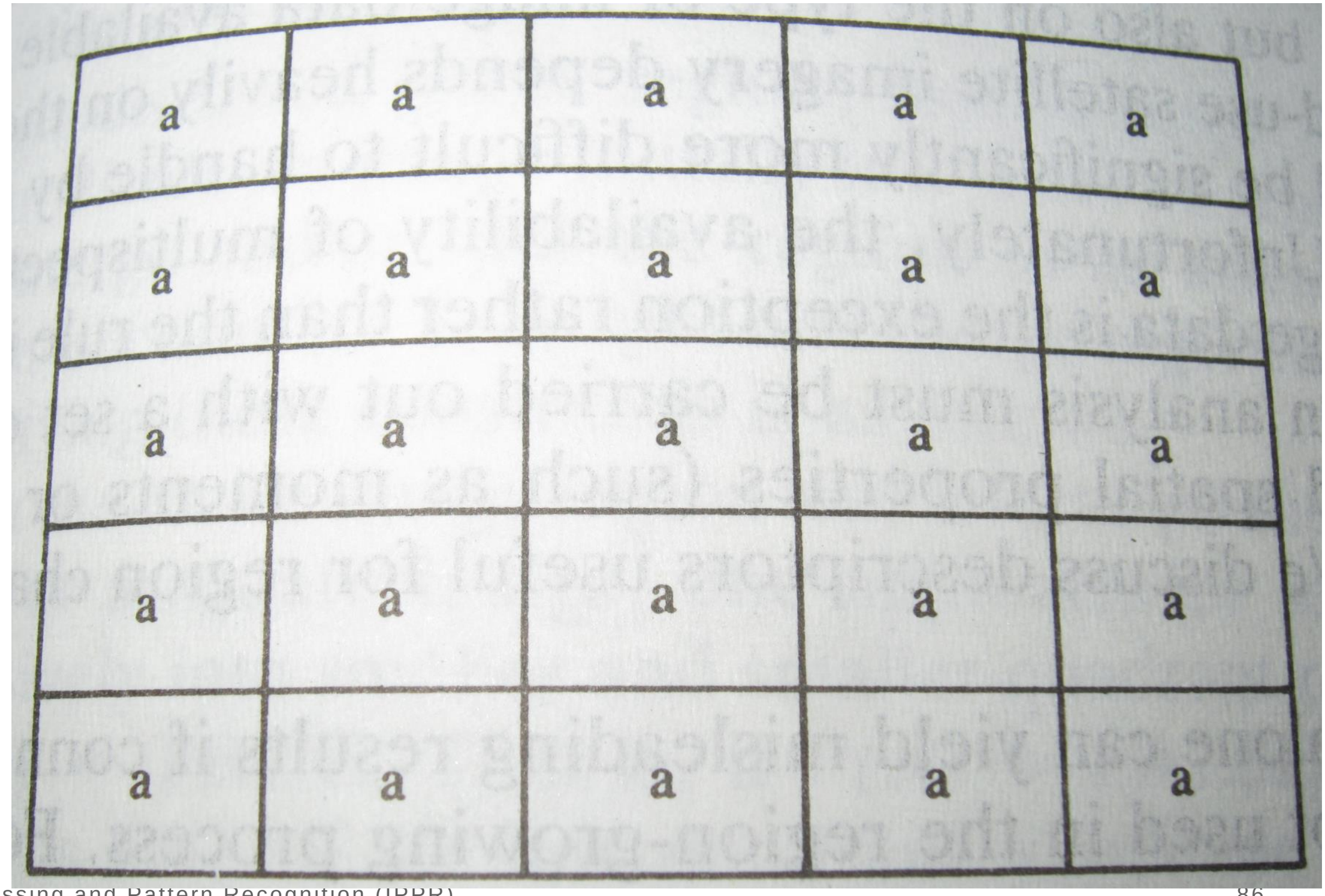
# Region-Based Segmentation
# Basic Formulation

When T = 3

| | | | | |
|---|---|---|---|---|
| a | a | b | b | b |
| a | a | b | b | b |
| a | a | b | b | b |
| a | a | b | b | b |
| a | a | b | b | b |

# Region-Based Segmentation
# Basic Formulation

When T = 8



# Region-Based Segmentation Basic Formulation

❖ **Seed Pixels**

Pixels of defective welds tend to have the maximum allowable digital value (255). All pixels having values of 255 are all selected as seed pixels.

❖ **Criteria for region growing**

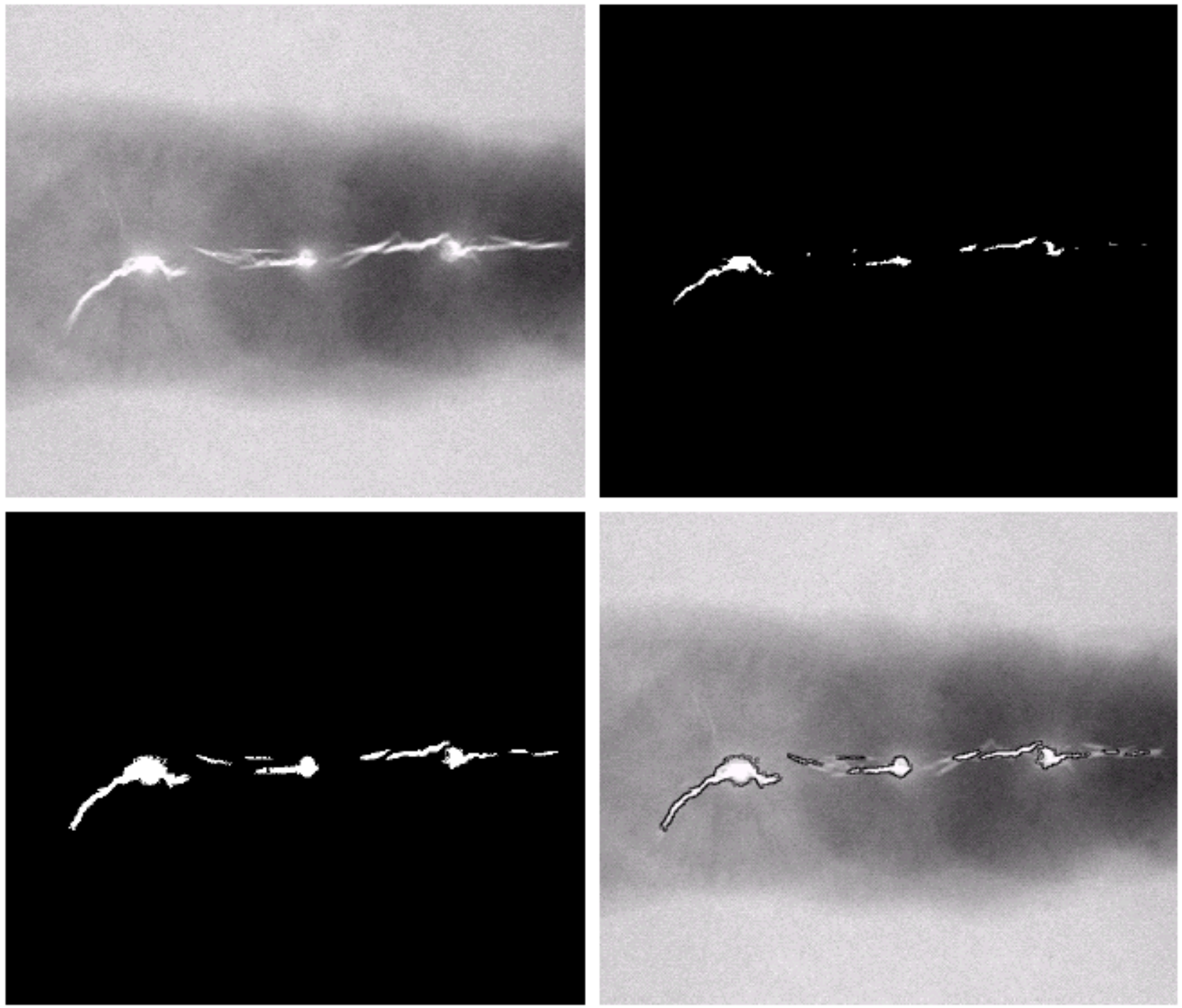
1)The absolute gray-level difference between any pixel and the seed had to be less than 65. This is based on histogram (Difference between 255 and the location of the first major valley to the left.

2)To be included in one of the regions, the pixel had to be 8-connected to at least one pixel in that region. If a pixel was found to be connected to more than one region, the regions were merged.

# Region-Based Segmentation
# Region Growing

a b
c d

**FIGURE 10.40**
(a) Image showing defective welds. (b) Seed points. (c) Result of region growing. (d) Boundaries of segmented defective welds (in black). (Original image courtesy of X-TEK Systems, Ltd.).



# Region-Based Segmentation
# Region Growing

Fig. 10.41 shows the histogram of Fig. 10.40 (a). It is difficult to segment the defects by thresholding methods. (Applying region growing methods are better in this case.)

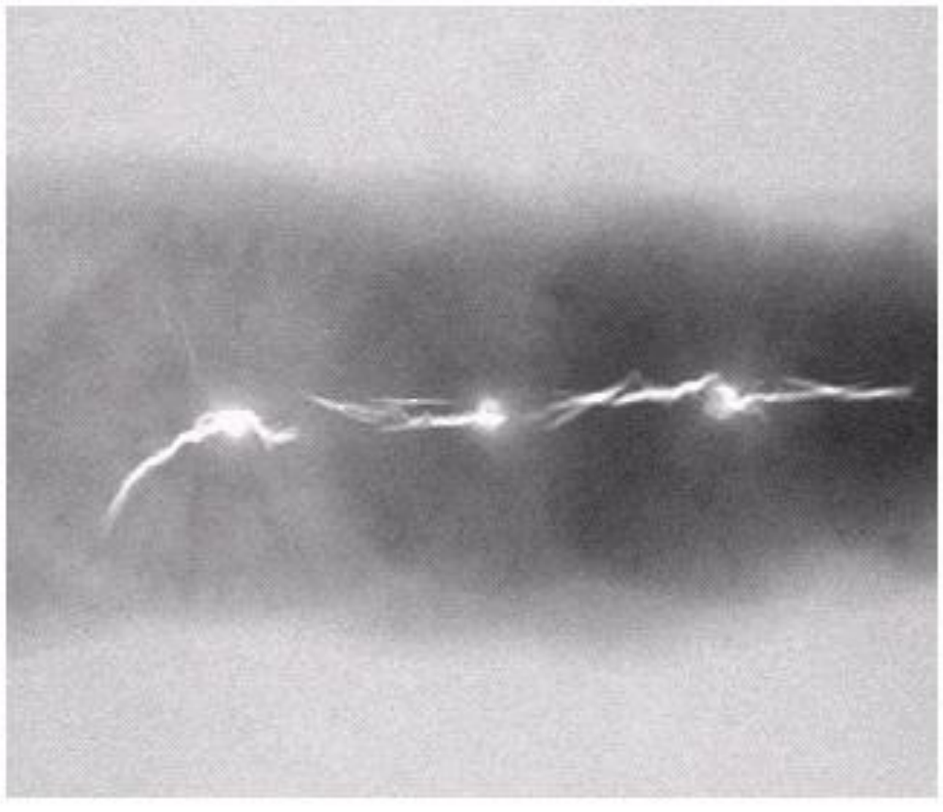


Figure 10.40(a)

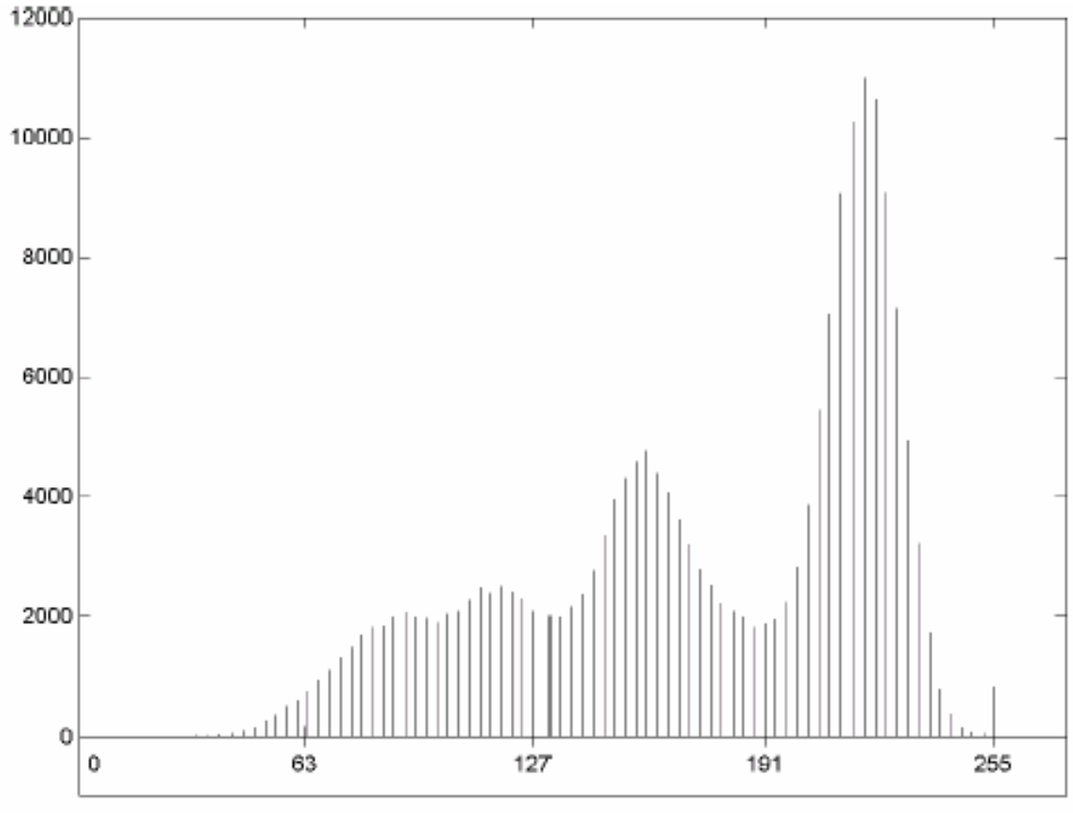


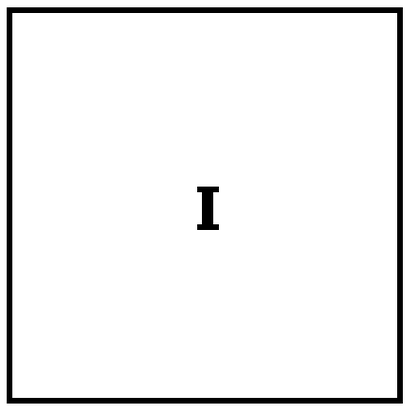

Figure 10.41

# Region-Based Segmentation
# Region Splitting and Merging

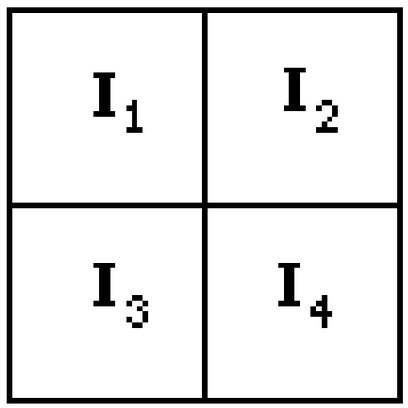
Region splitting is the opposite of region growing.

- First there is a large region (possible the entire image).
- Then a predicate (measurement) is used to determine if the region is uniform.
- If not, then the method requires that the region be split into two regions.
- Then each of these two regions is independently tested by the predicate (measurement).
- This procedure continues until all resulting regions are uniform.

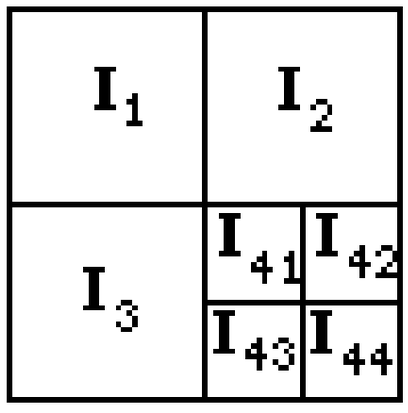
# Region-Based Segmentation
# Region Splitting and Merging



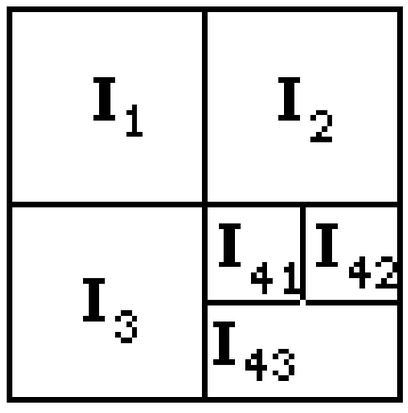

(a) Whole Image




(b) First Split




(c) Second Split




(d) Merge

# Region-Based Segmentation
# Region Splitting

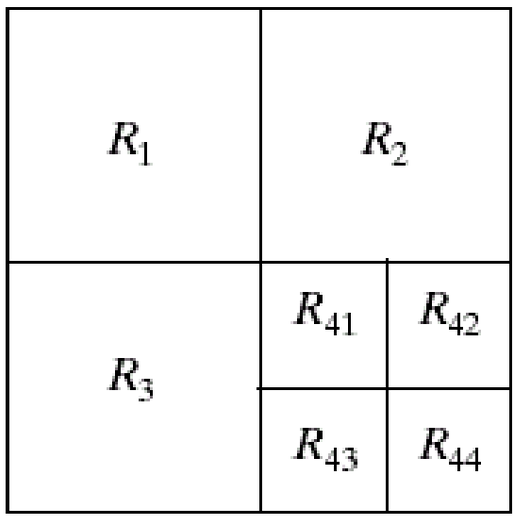
The main problem with region splitting is determining where to split a region.

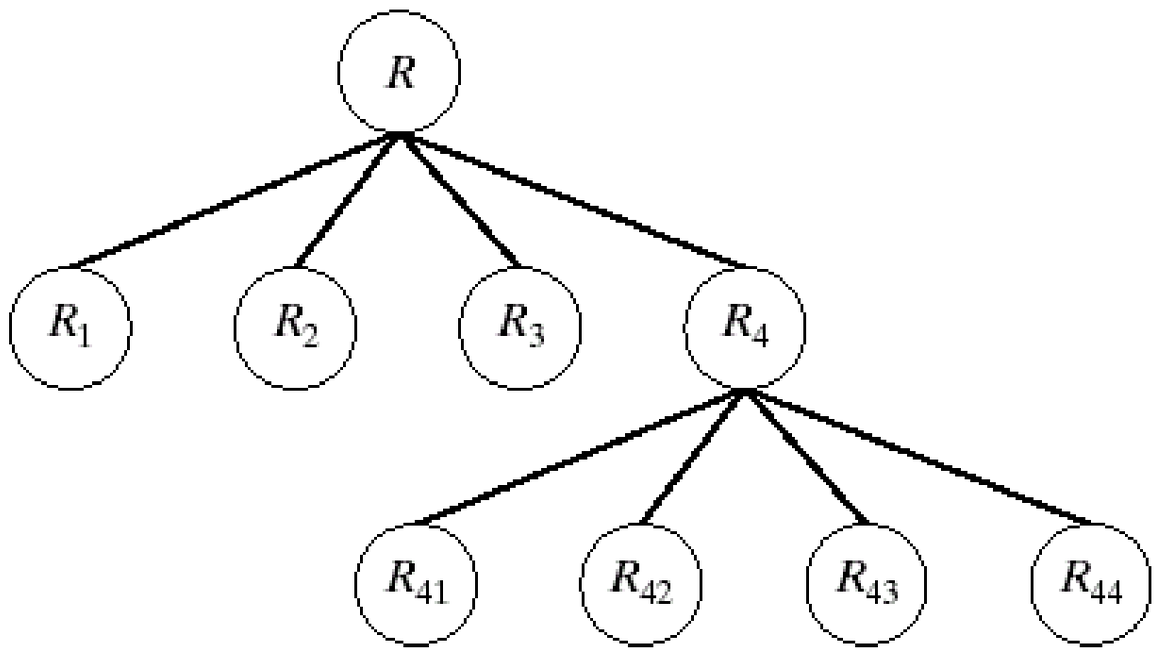
One method to divide a region is to use a quadtree structure.

Quadtree: a tree in which nodes have exactly four descendants.

a b

**FIGURE 10.42**
(a) Partitioned image.
(b) Corresponding quadtree.

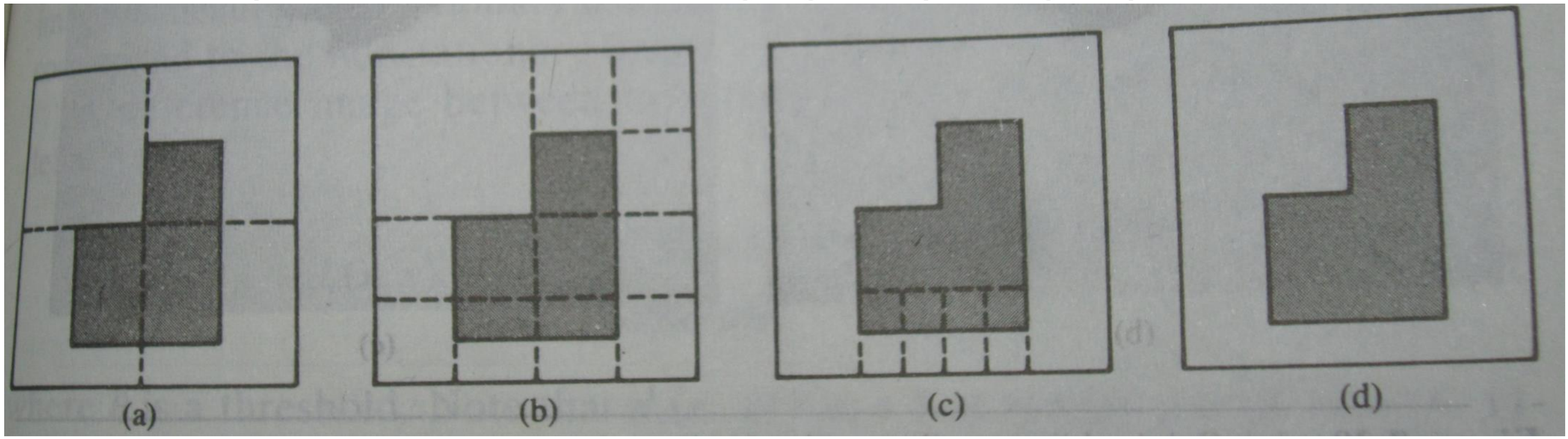


# Region-Based Segmentation
# Region Splitting

The split and merge procedure:

- Split into four disjoint quadrants any region $R_i$ for which $P(R_i)$ = FALSE.
- Merge any adjacent regions $R_j$ and $R_k$ for which $P(R_j \cup R_k)$ = TRUE. (the quadtree structure may not be preserved)
- Stop when no further merging or splitting is possible.

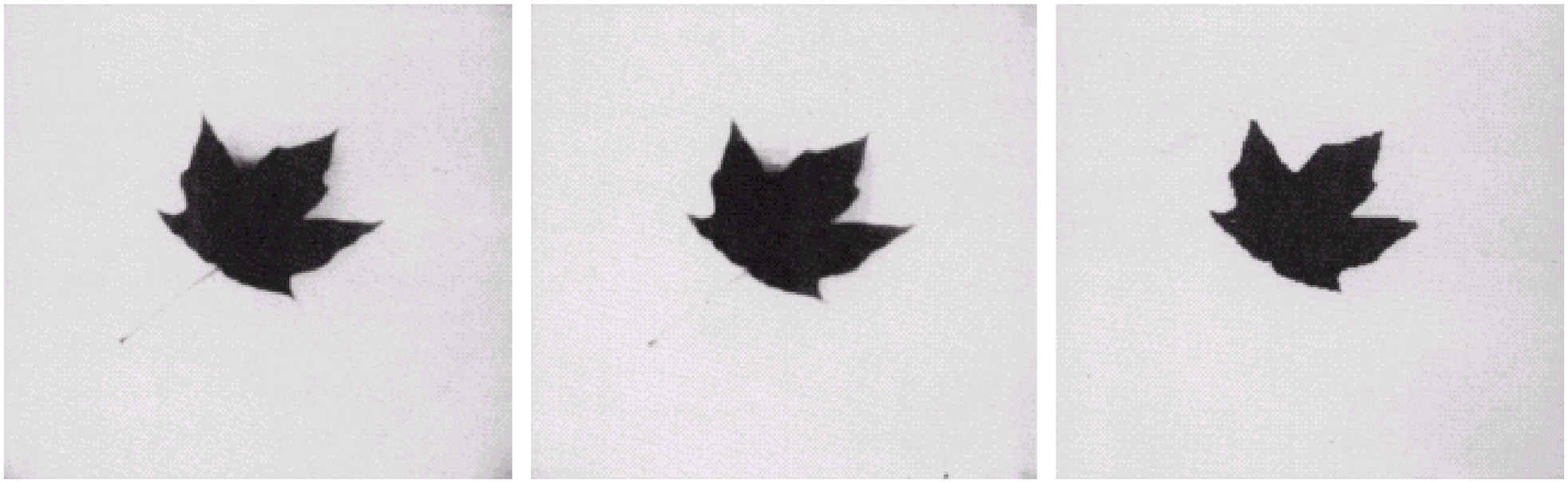



Both the object and background have constant gray levels and $P(R_i)$ = TRUE if all pixels in $R_i$ have the same intensity.

# Region-Based Segmentation
# Region Splitting and Merging

a b c

**FIGURE 10.43**
(a) Original image. (b) Result of split and merge procedure. (c) Result of thresholding (a).



- $P(R_i)$=TRUE if at least 80% of the pixels in $R_i$ have the property

$$\left|z_j - m_i\right| \le 2\sigma_i$$

Pixel intensity value

Mean of the region

Std deviation

- Can specify P(R) based on texture content for texture segmentation

# Results – Region grow

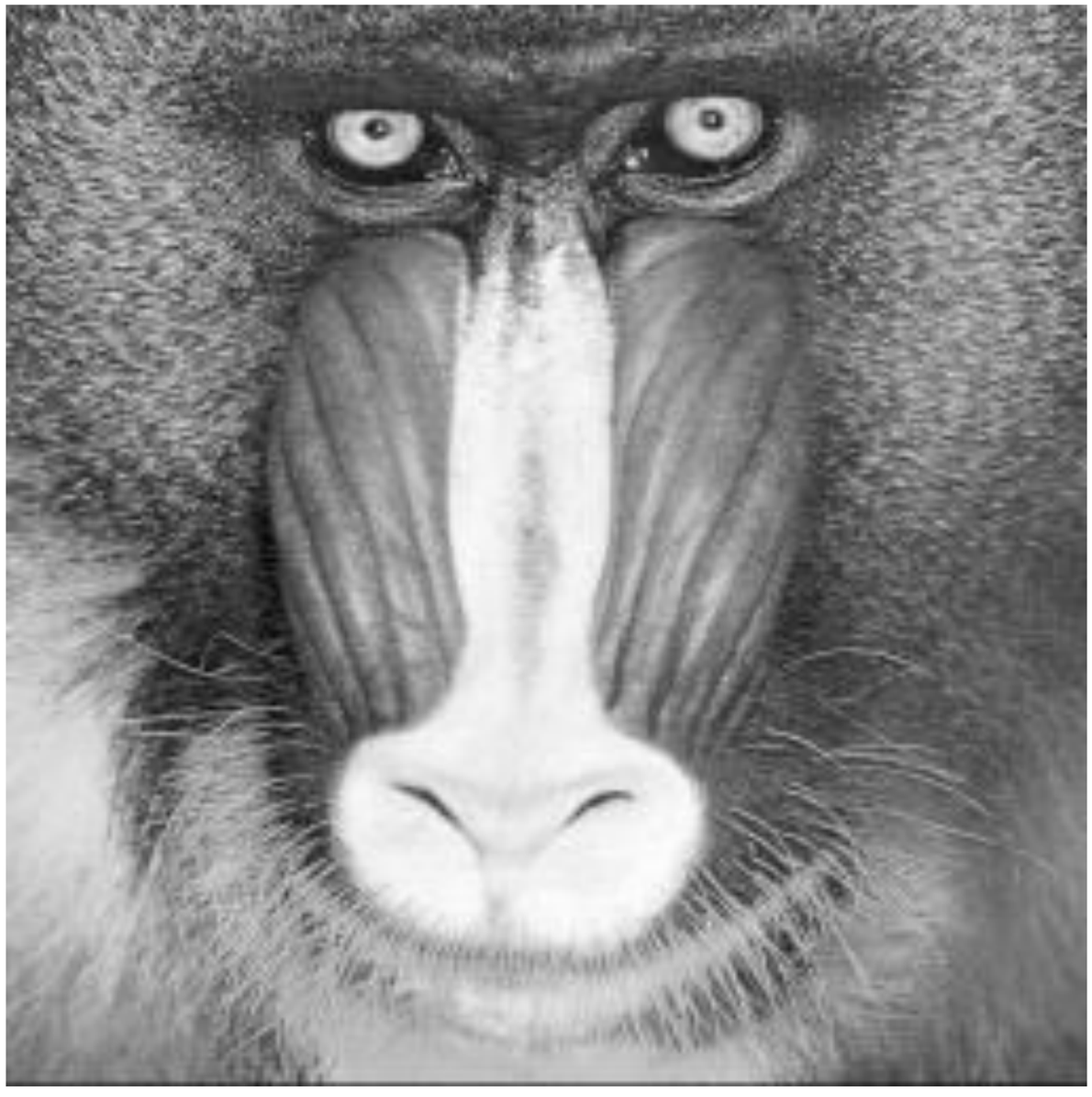
# Results – Region Split

http://astro.temple.edu/~siddu

# Results – Region Split and Merge

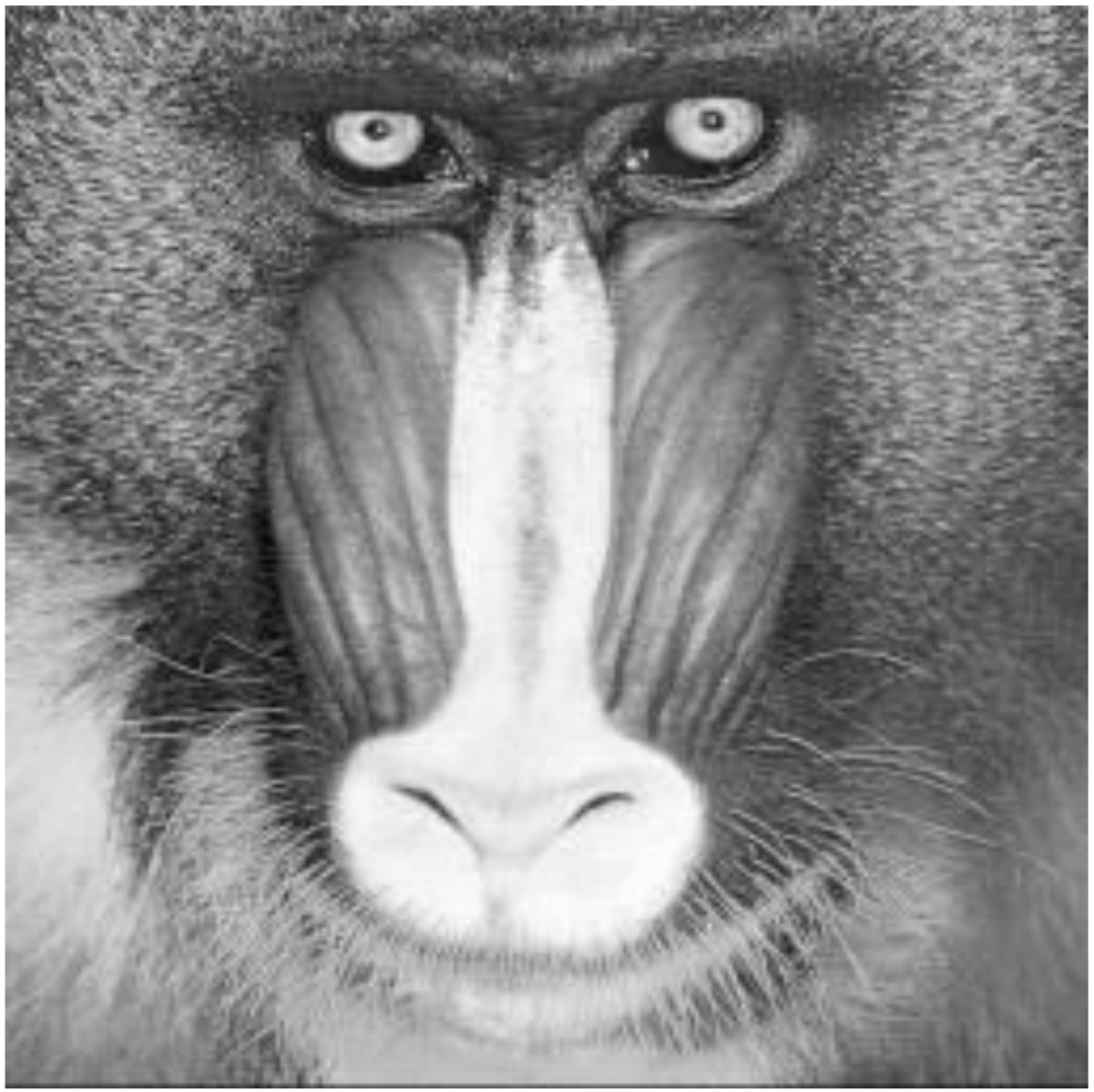
http://astro.temple.edu/~siddu

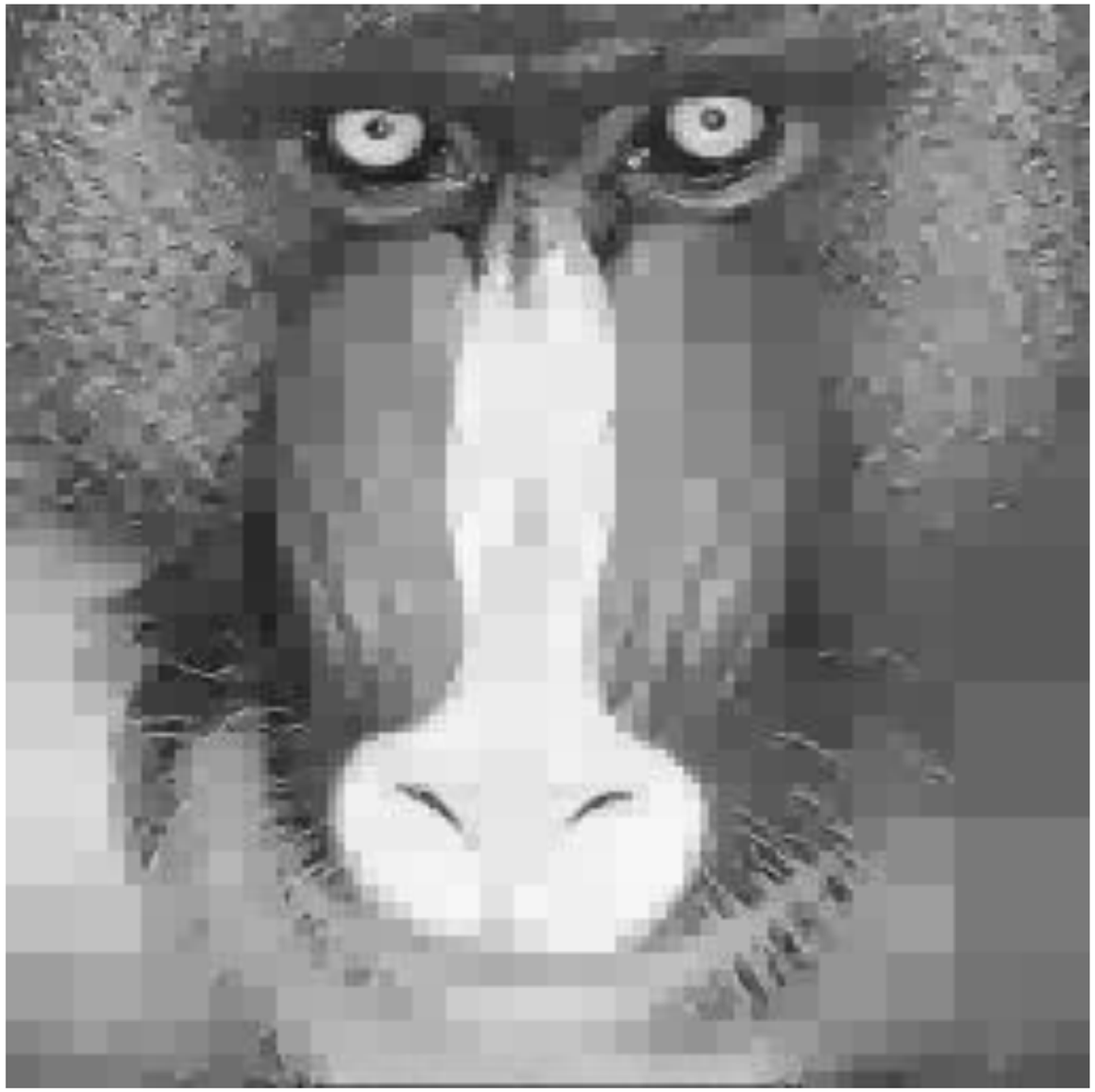
# Results – Region growing



http://astro.temple.edu/~siddu

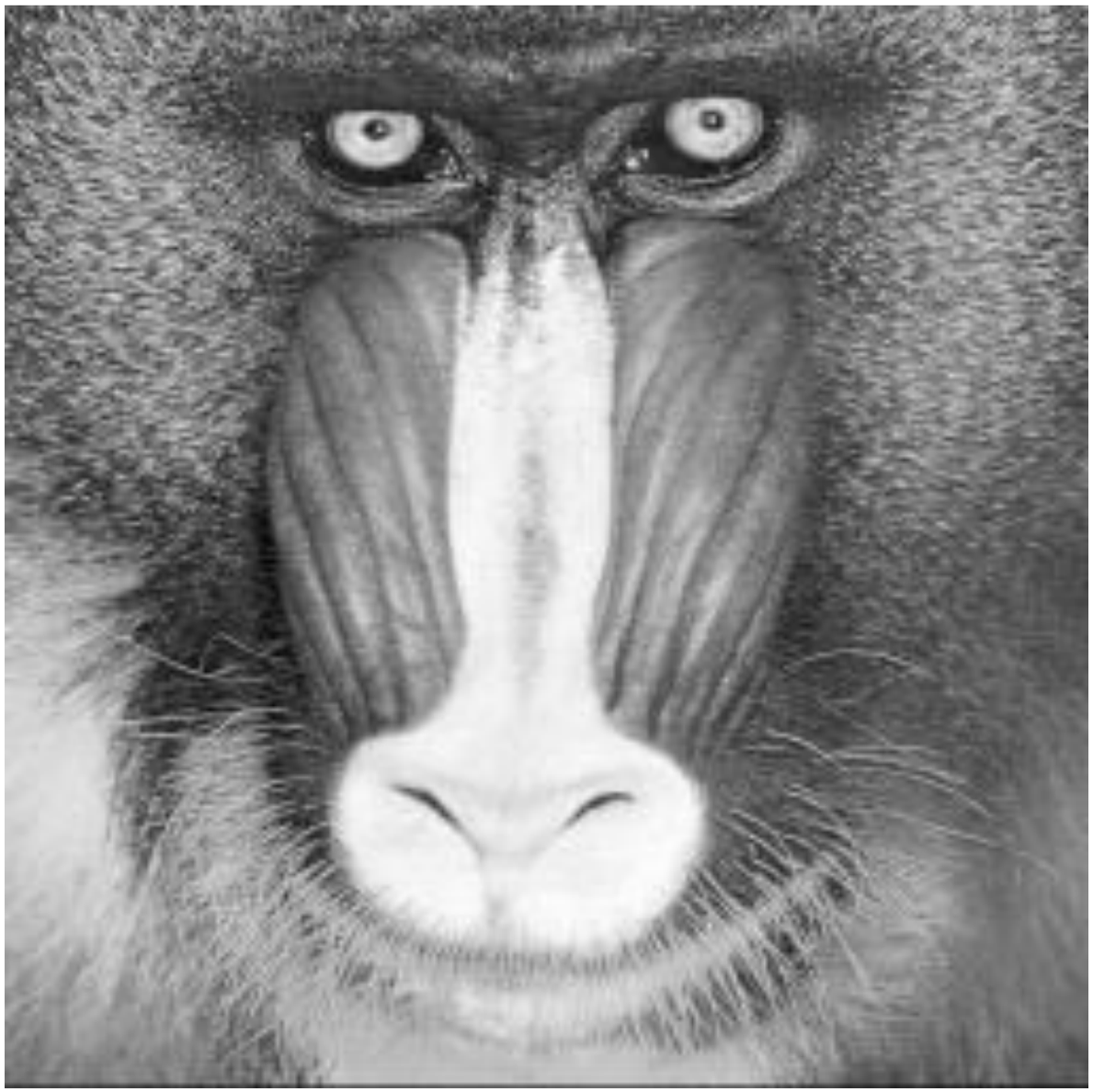
# Results – Region Split

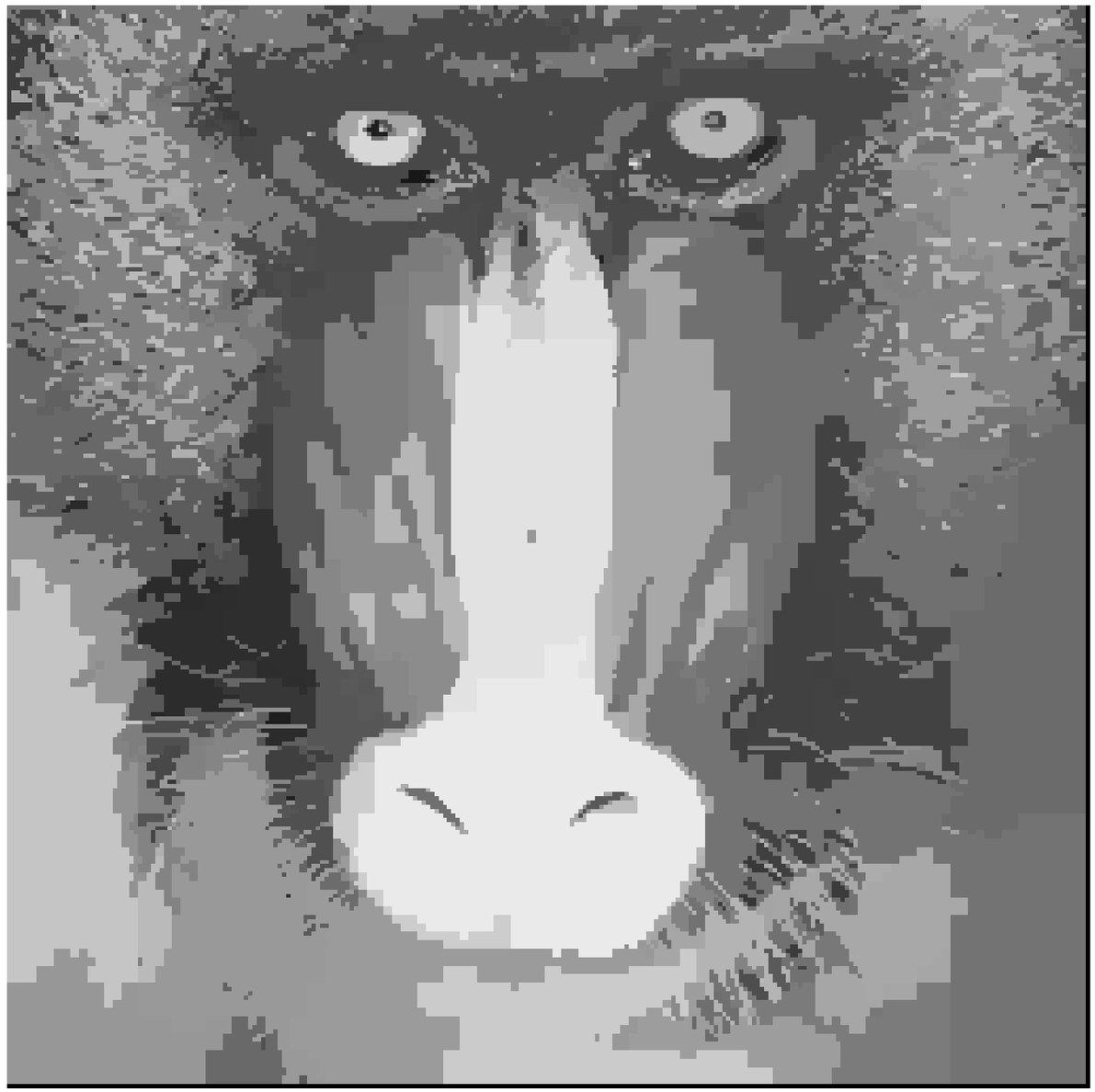




http://astro.temple.edu/~siddu

# Results – Region Split and Merge





http://astro.temple.edu/~siddu

# References

– “Digital Image Processing”, Rafael C. Gonzalez & Richard E. Woods, Addison-Wesley, 2002

–“Fundamentals of Digital Image Processing” Anil K. Jain, 1989

–“Digital Image Processing” – Algorithms and Application , A multimedia approach. Prof. Ioannis Pitas

–Computer Vision and Image Processing: A Practical Approach

–Image Processing and Pattern Recognition Slides of Dr. Sanjeeb Prasad Panday

Thank you !!!