Image Segmentation

Delivered by

Dr. Subarna Chatterjee

subarna.cs.et@msruas.ac.in

Image Segmentation

At the end of this session, student will be able to

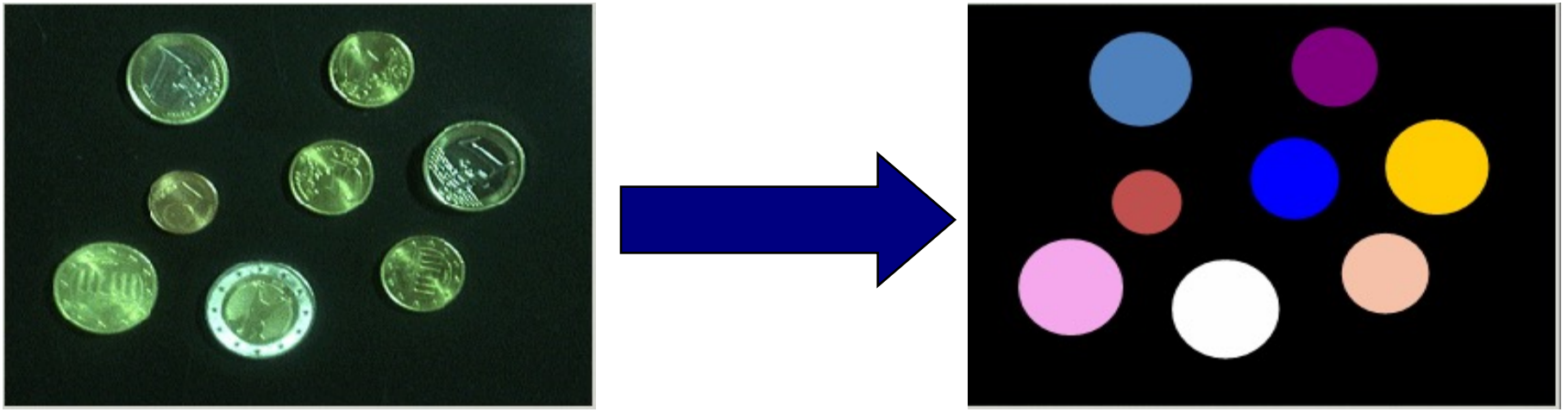
- Describe the importance, applications and significance of image segmentation in the field of Digital Image Processing
- Identify concepts and types of thresholding to segment
- Explain concepts and implement Region growing, region merging and splitting for image segmentation

Session Topics

- Thresholding and binarization
- Threshold level adjustment
- Multi-thresholding
- Region growing
- Detection of regions of interest

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



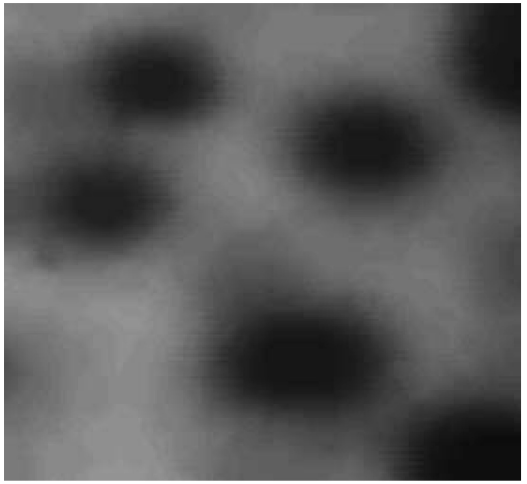
Segmentation

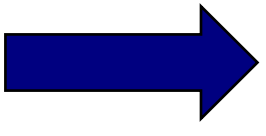
- Segmentation is the fundamental first step towards analyzing and understanding images
- Image segmentation involves breaking an image down into its basic components or regions
- Segmentation may also be defined as decomposing the image into its constituent parts, extracting the location and outline of objects of interest in an image. (However the definition of an "interesting object" is dependent both upon the application and the image modality)
- In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and "the rest" (background)

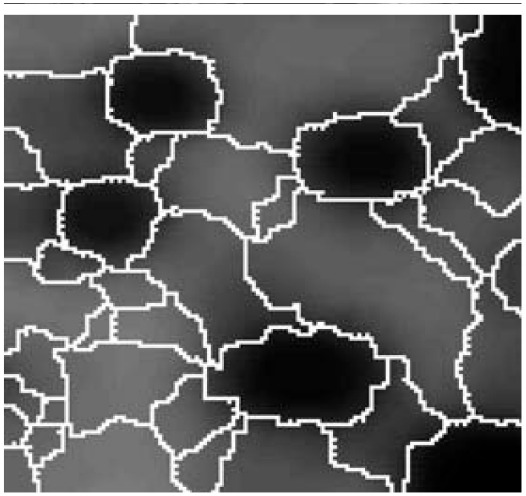
Segmentation Examples

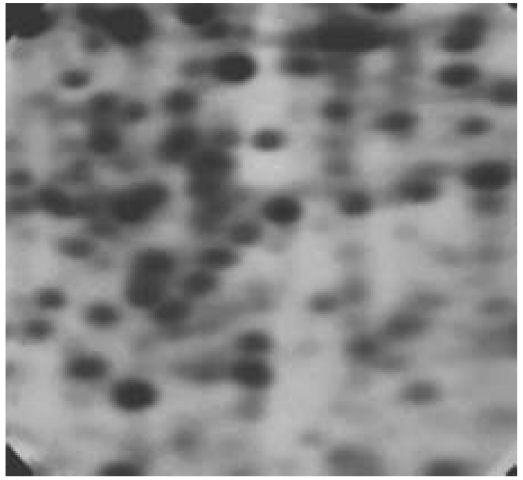
Input Images

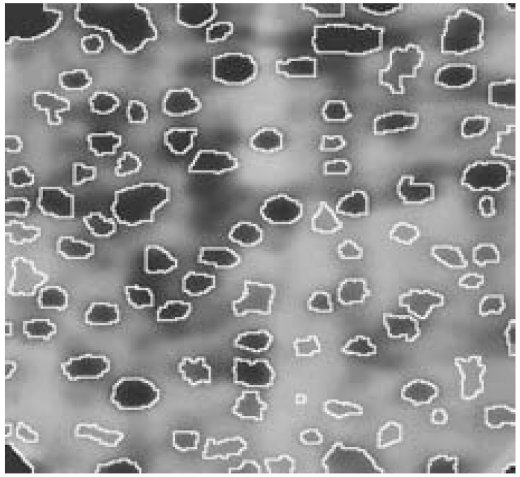
Segmented Images











Segmentation

- It is important to understand that:
  - there is no universally applicable segmentation technique that will work for all images
  - no segmentation technique is perfect
- Biomedical data is characterized by high geometric variation both within and between patients or samples
- Organs deform between observations due to physiological changes or simple displacement, cells grow and split and often we are searching for pathological instances which by their very nature defy classification
- In many cases even human experts differ in their judgment of a good segmentation
- Despite the bleak outlook, it is these difficulties which are being overcome at the forefront of medical image segmentation

Region/Segment

- Region – an aggregation of pixels
- Properties like gray level, color, texture, shape help to identify regions and similarity of such properties, is used to build groups of regions having a particular meaning
- In mathematical sense the segmentation of the image *I*, which is a set of pixels, is the partition of *I* into *n* disjoint sets $R_1, R_2, \ldots, R_n$, called segments or regions such that their union of all regions equals I

$$I = R_1 \cup R_2 \cup \ldots\ldots \cup R_n$$

Image Segmentation Applications

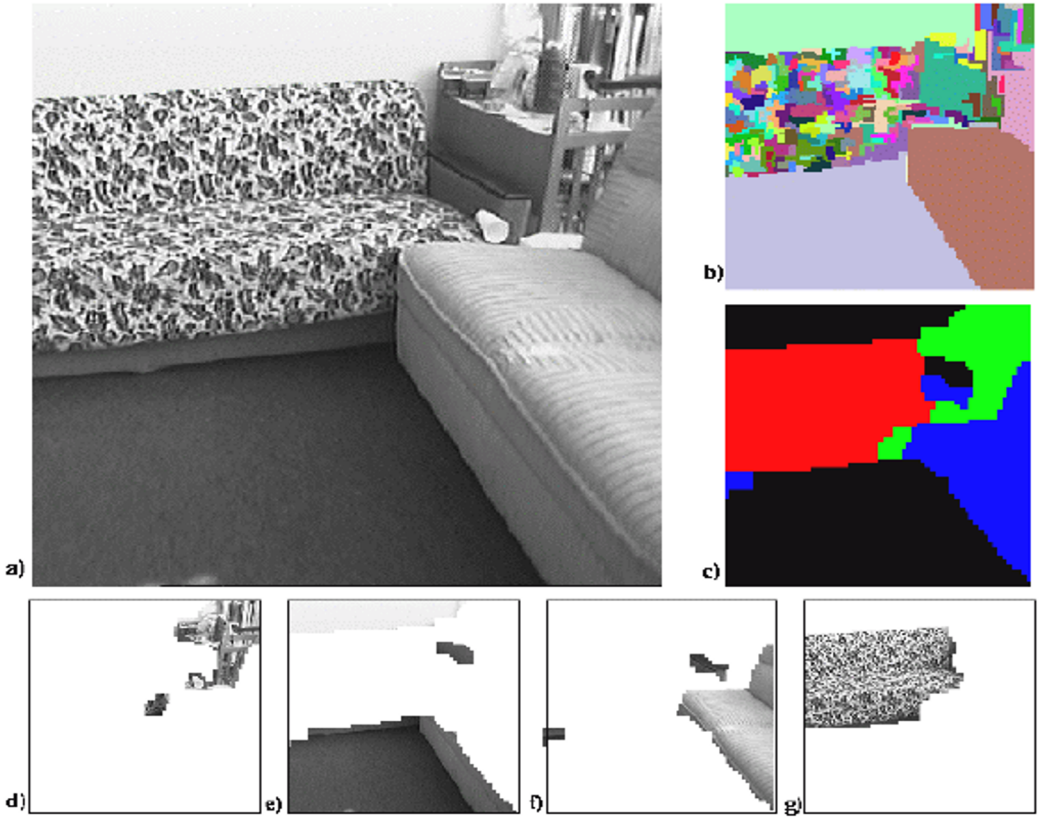
- Usually, image segmentation is an initial and vital step in a series of processes aimed at overall image understanding
- Applications of image segmentation include
  - Identifying objects in a scene for object-based measurements such as size and shape
  - Identifying objects in a moving scene for *object-based video compression (MPEG4)*
  - Identifying objects which are at different distances from a sensor using depth measurements from a laser range finder enabling path planning for a mobile robots

Image Segmentation Examples

- Segmentation based on greyscale
- Very simple ‘model’ of greyscale leads to inaccuracies in object labelling
- Segmentation based on greyscale
- Very simple ‘model’ of greyscale leads to inaccuracies in object labelling

Image segmentation examples

- Segmentation based on texture
- Enables object surfaces with varying patterns of grey to be segmented



Types of Segmentation

- Image segmentation methods will look for objects that either have some measure of *homogeneity* within themselves or have some measure of *contrast* with the objects on their border.
- The homogeneity and contrast measures can include features such as gray level, color and texture
- Image segmentation techniques can be divided into four main categories:
  - Thresholding
  - Region growing and shrinking
  - Boundary detection
  - Clustering methods

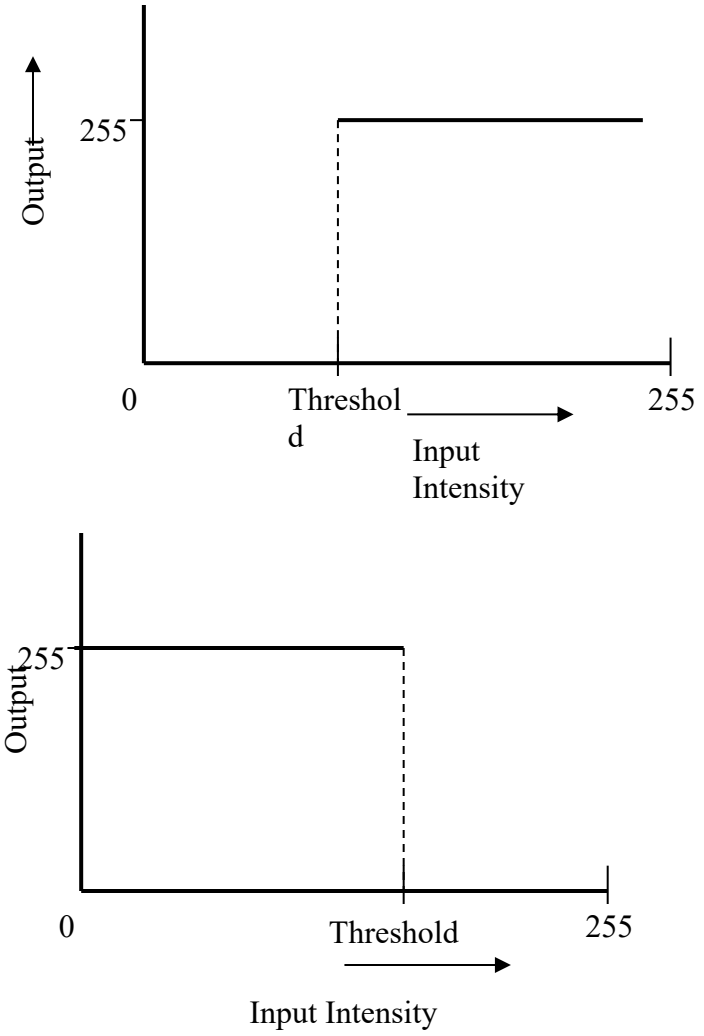
Thresholding

- Thresholding is based upon a simple concept
- Simplicity of implementation
- A parameter called the brightness threshold is chosen and applied to the image a[m,n] as follows:

If a[m,n] > θ then a[m,n] = object = 1
Else a[m,n] = background = 0

0→0(black), 1→255(white)

- This version of the algorithm assumes that we are interested in light objects on a dark background. For dark objects on a light background we would use:
  If a[m,n] < θ a[m,n] = object = 0
  Else a[m,n] = background = 1



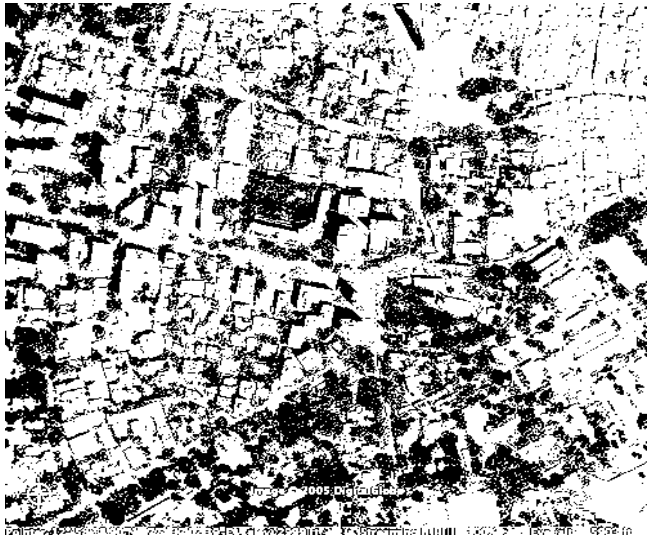
Concept of Binary Thresholding

Binary Thresholding

Original

Threshold=100

Threshold=75



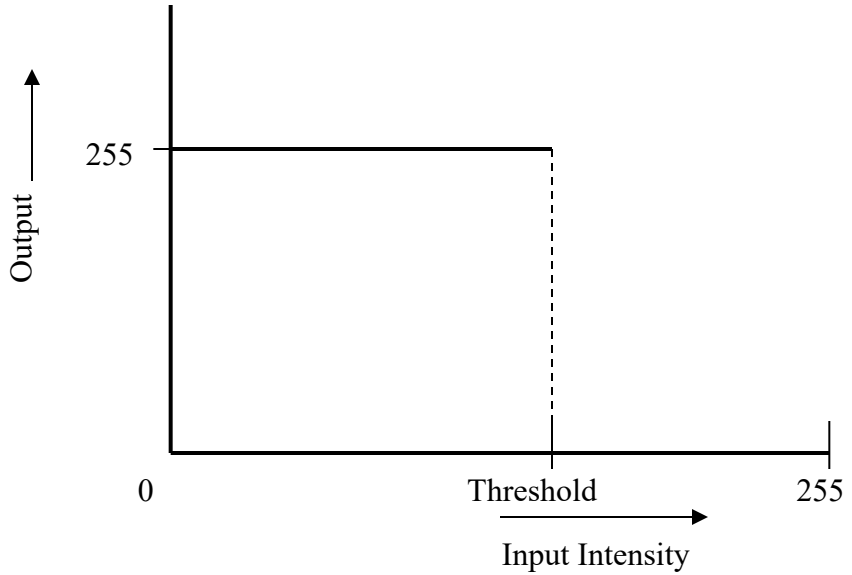
Threshold=150

Binary Inverted Thresholding

For dark objects on a light background we would use:

If a[m,n] < θ a[m,n] = object = 0

else a[m,n] = background = 1

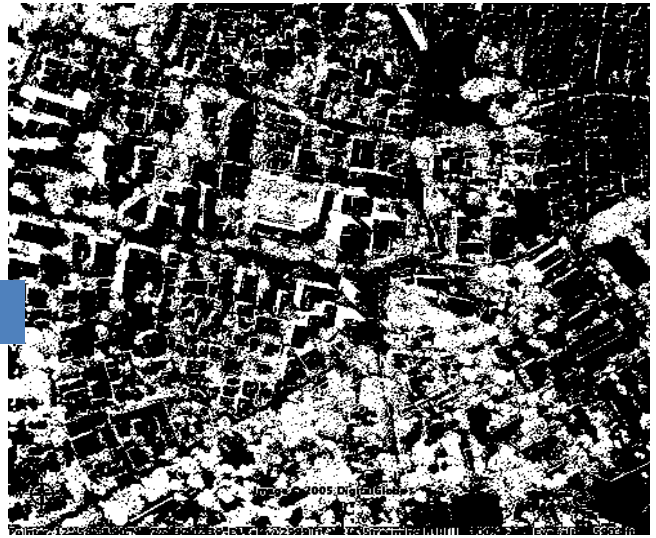


Binary Inverted Thresholding

Original

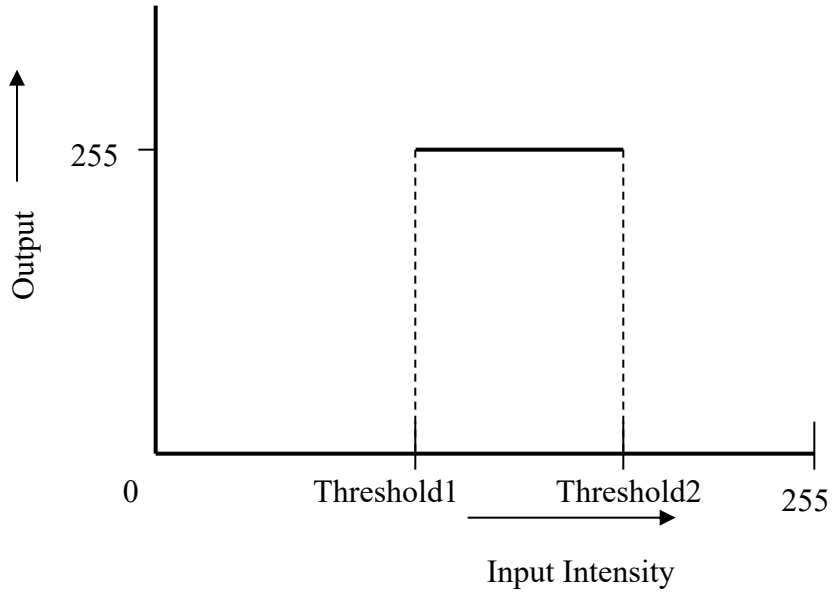
Threshold=100

Threshold=75



Threshold=150

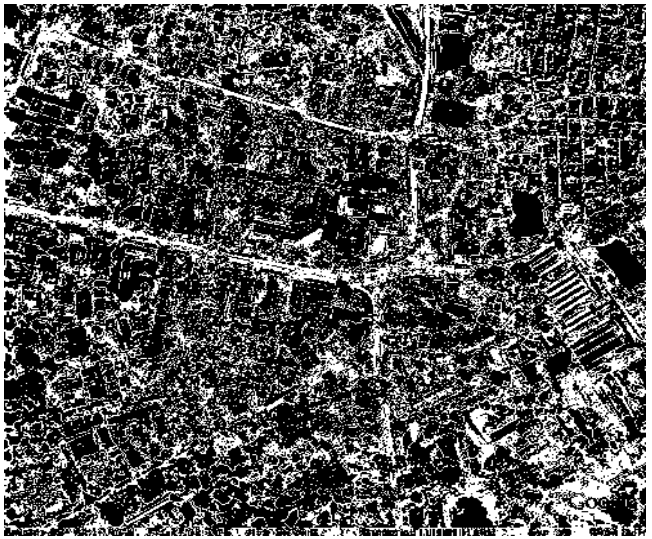
Binary Interval Thresholding



All input gray intensities values between Threshold1 and Threshold2 both inclusive are converted to white and others are assigned black

Binary Interval Thresholding

Original

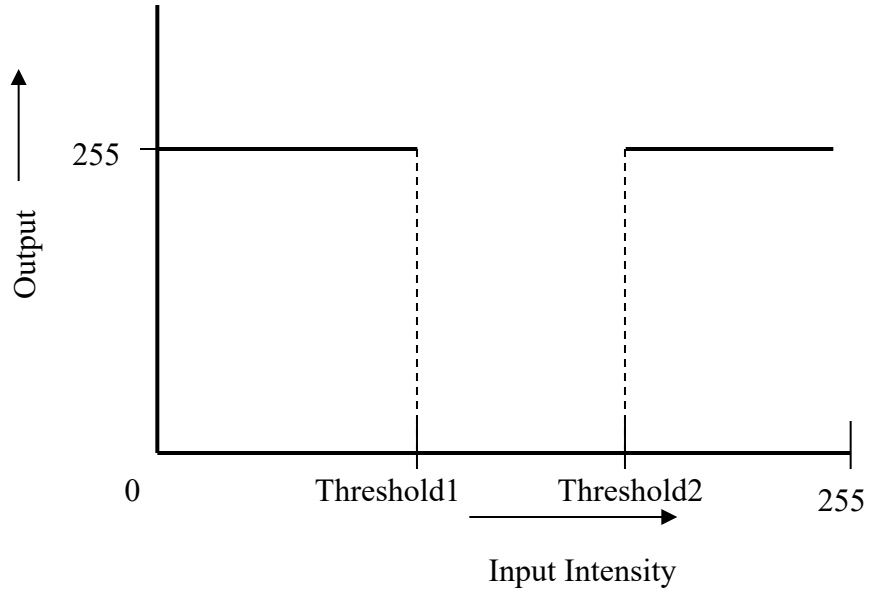


Threshold = 75 to 150

Threshold = 125 to 175

Threshold = 150 to 200

Inverted Binary Interval Thresholding



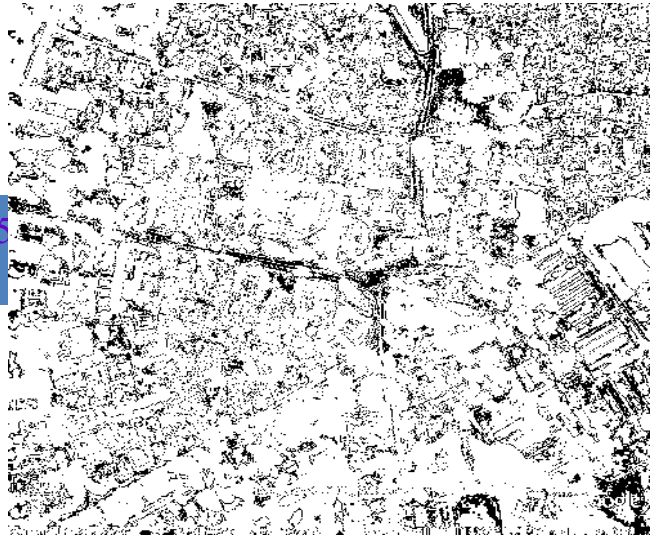
All input gray intensities values between Threshold1 and Threshold2 both inclusive are converted to black and others are assigned white

Inverted Binary Interval Thresholding

Original

Threshold = 75 to 150

Threshold = 125 to 175



Threshold = 150 to 200

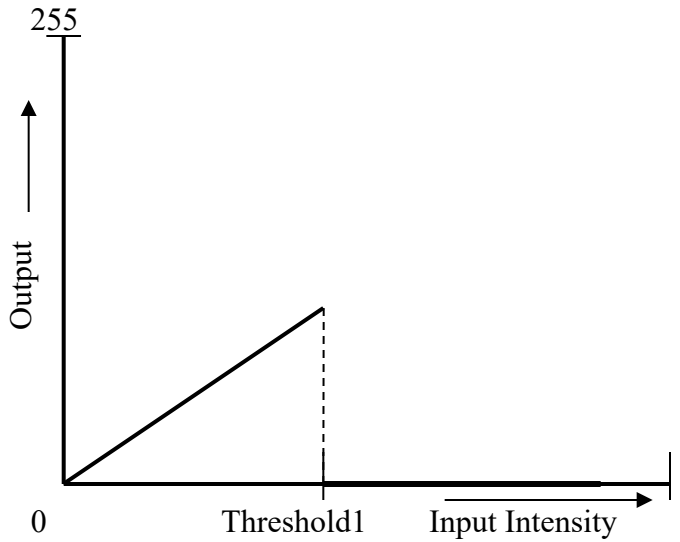
Grayscale Thresholding – Single Intensity Background

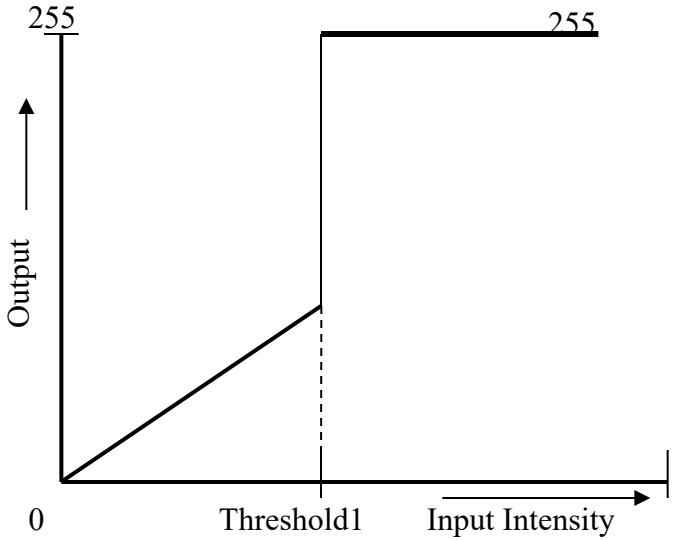
- A threshold is chosen and applied to the image a[m,n] as follows:

  If a[m,n] < θ a[m,n] = object = a[m,n]

  else a[m,n] = Single intensity

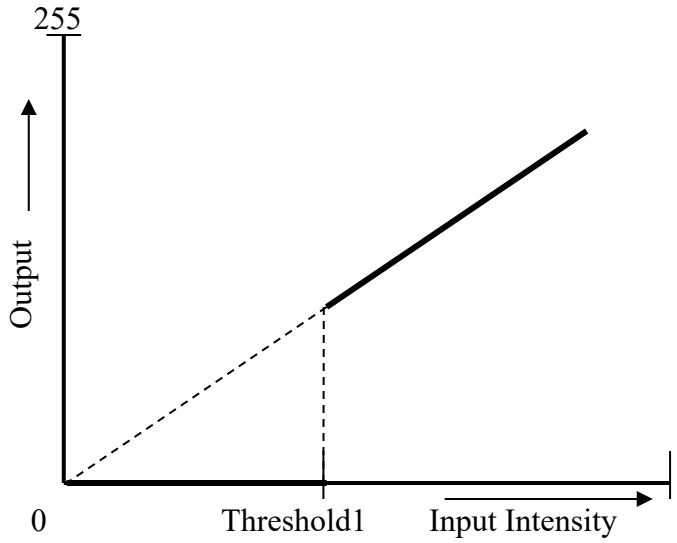
- This version of the algorithm assumes that we are interested in grayscale objects and apply any one intensity to the background

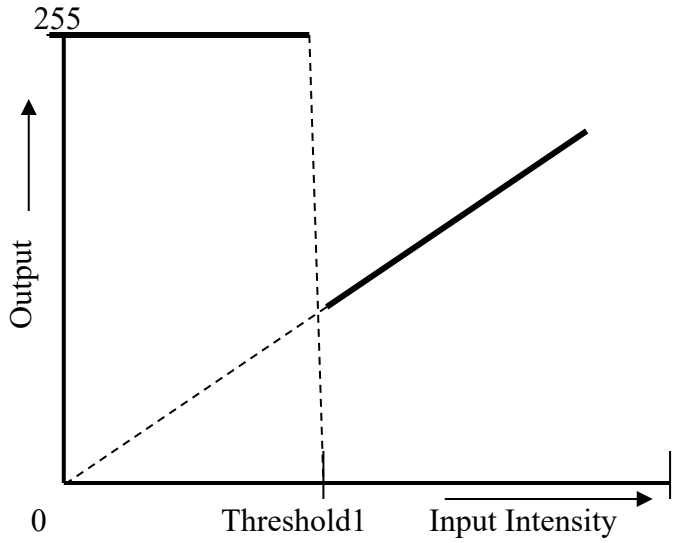




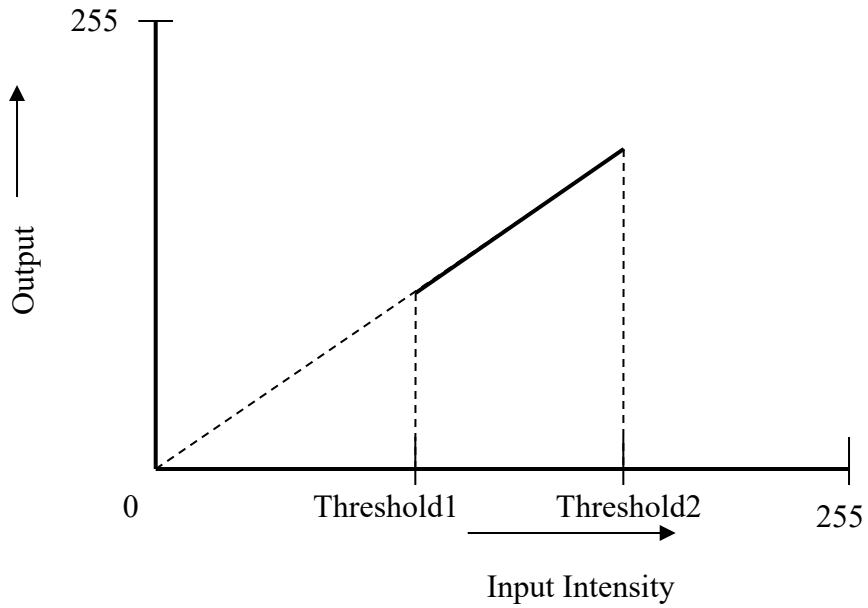
Grayscale Thresholding – Single Intensity Background

If a[m,n] >= θ a[m,n] = object = a[m,n]
else a[m,n] = background = Single intensity





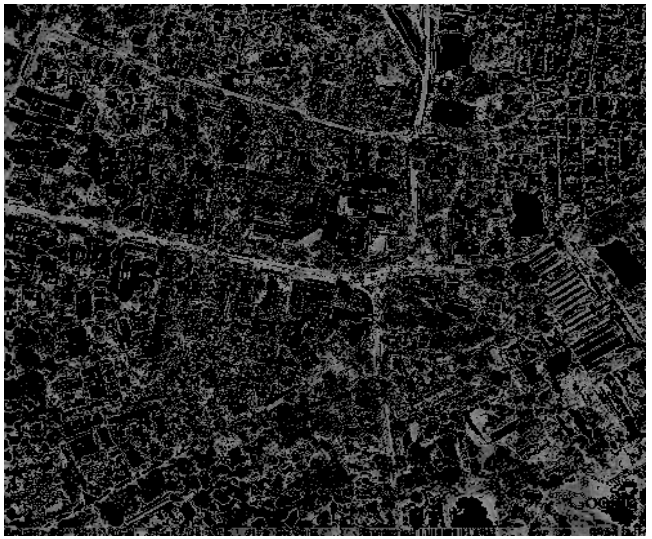
Grayscale Thresholding



All input gray intensities values between Threshold1 and Threshold2 both inclusive are retained and others are assigned black

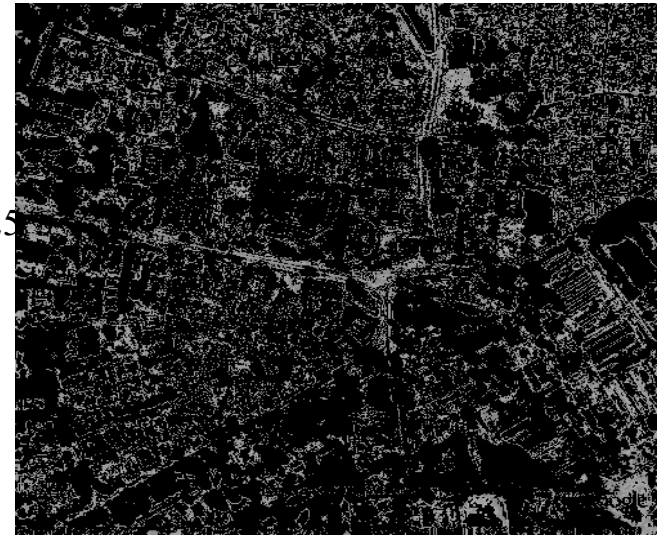
Grayscale Thresholding

Original



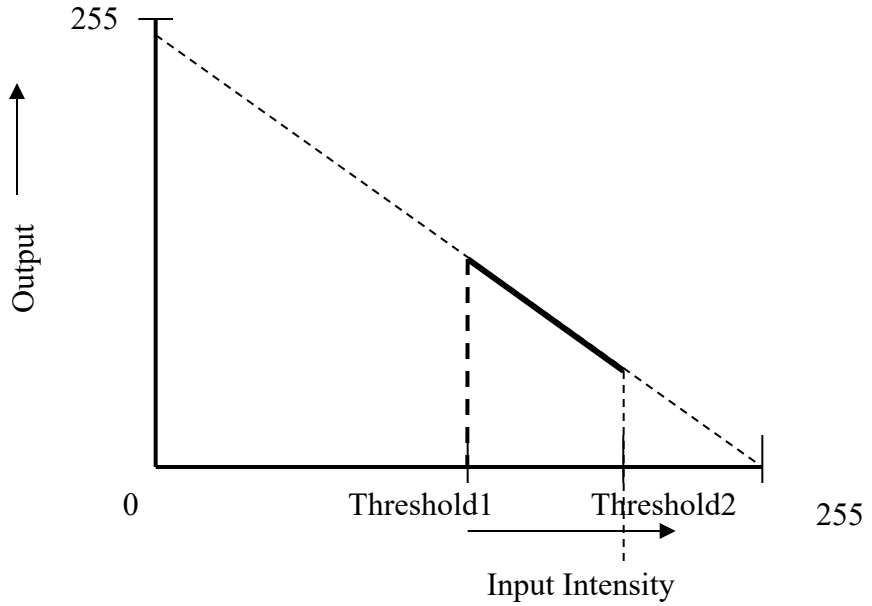
Threshold = 75 to 150

Threshold = 125 to 175



Threshold = 150 to 200

Inverted Grayscale Thresholding



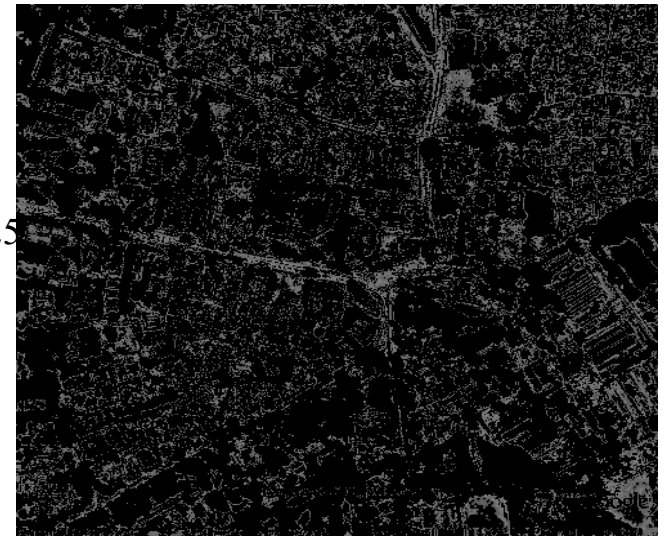
All input gray intensities values between Threshold1 and Threshold2 both inclusive are gray scale inverted and others are assigned black

Inverted Grayscale Thresholding

Original

Threshold = 75 to 150

Threshold = 125 to 175



Threshold = 150 to 200

Thresholding

- There is no universal procedure for threshold selection that is guaranteed to work on all images, but there are a variety of alternatives
- **Fixed threshold** - Use a threshold that is chosen independently of the image data
- If it is known that one is dealing with very high-contrast images where the objects are very dark and the background is homogeneous and very light, then a constant threshold of 128 on a scale of 0 to 255 might be sufficiently accurate
- By accuracy we mean that the number of falsely-classified pixels should be kept to a minimum
- **Histogram-derived thresholds** - In most cases the threshold is chosen from the brightness histogram of the region or image that we wish to segment

Thresholding Example

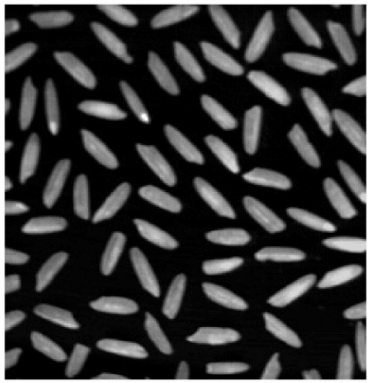


Image of rice with black background

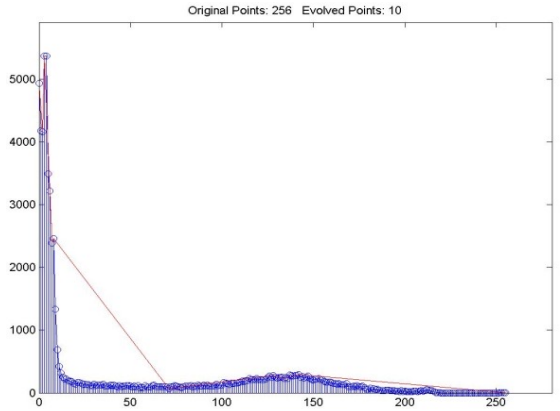


Image histogram of rice

Image after segmentation

Basic Global Thresholding

- Simplest of all thresholding techniques is to partition the image histogram by using a single global threshold T based on the histogram of an image

- Segmentation is achieved by scanning the image pixel by pixel and labelling the pixel as object or background, depending on whether the gray level of that pixel is greater or less than the value of T

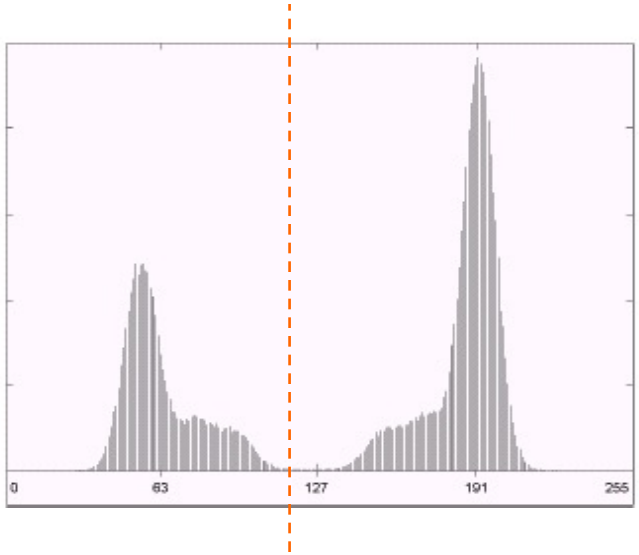
- The success of this technique very strongly depends on how well the histogram can be partitioned

Basic Global Thresholding

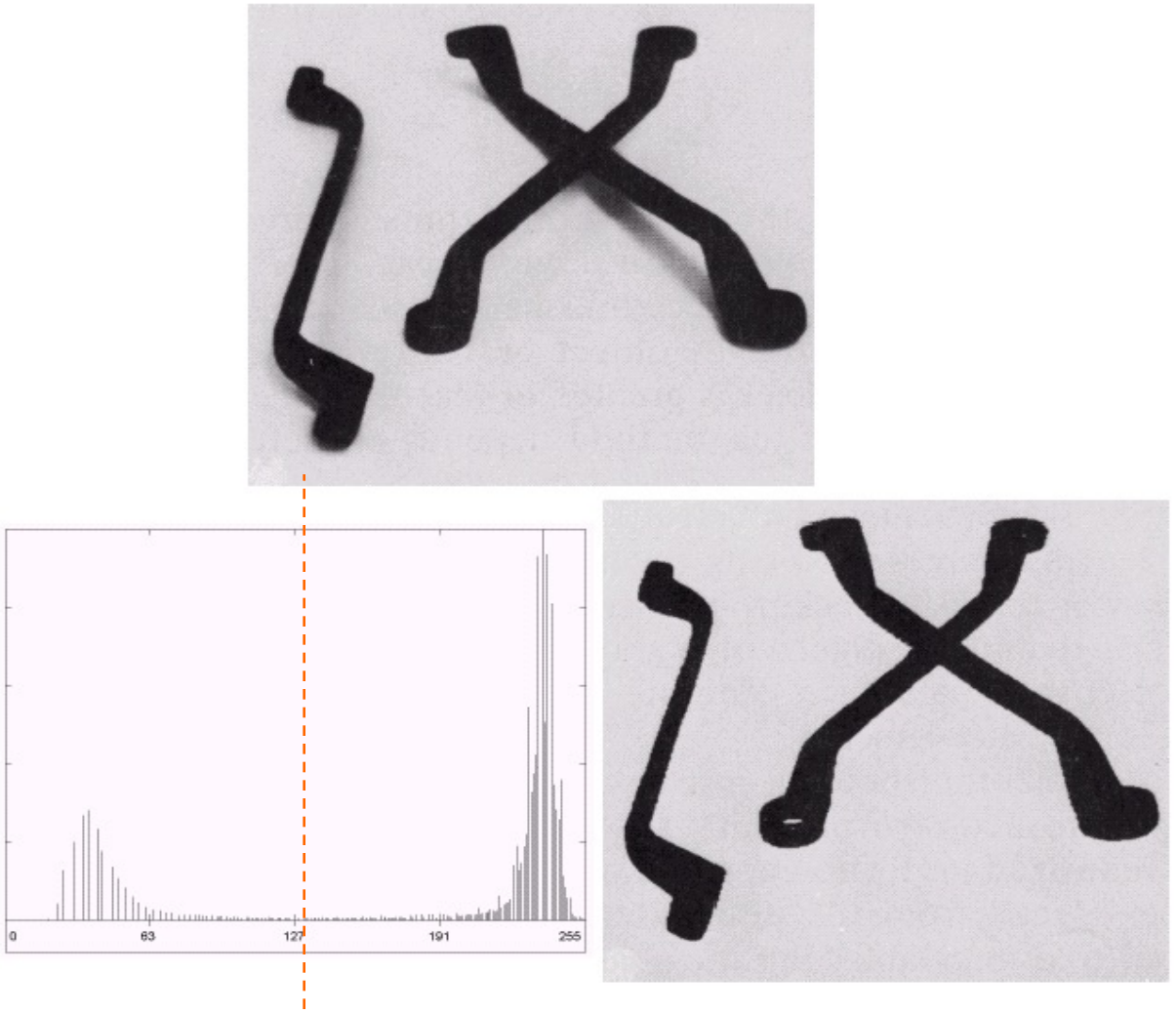
- A simple algorithm:

1. Initial estimate of T
2. Segmentation using T:
   - G_1, pixels brighter than T;
   - G_2, pixels darker than (or equal to) T.
3. Computation of the average intensities m_1 and m_2 of G_1 and G_2.
4. New threshold value: $T_{new} = (m_1 + m_2)/2$
5. If $|T - T_{new}| > \Delta T$, back to step 2, otherwise stop

Global Thresholding Example



Basic Global Thresholding Example



Basic Global Thresholding Algorithm

- A good initial estimate of T is the average gray level value of the pixels
- More appropriate is a value midway between maximum and minimum gray levels
- This algorithm works very well for finding thresholds for bimodal histogram

Ridler and Calvard Iterative Technique

- 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 μ_1 and G_2 to give μ_2
  4. Compute a new threshold value:

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

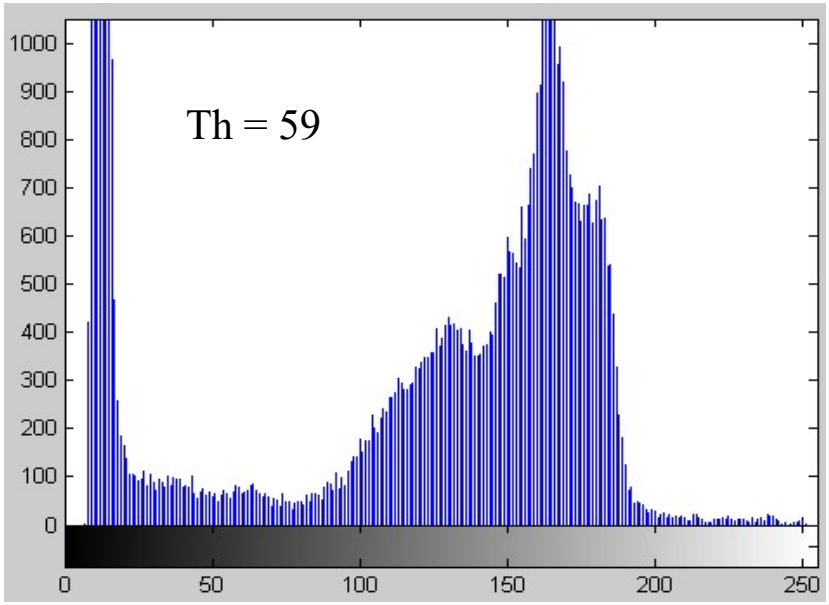
  5. Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit T_0

Results

Segmentation based on threshold estimation using preceding algorithm

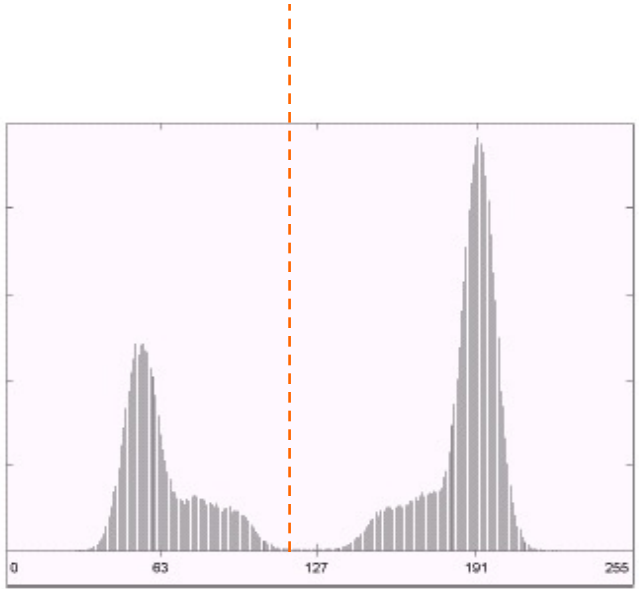
original image

thresholded image

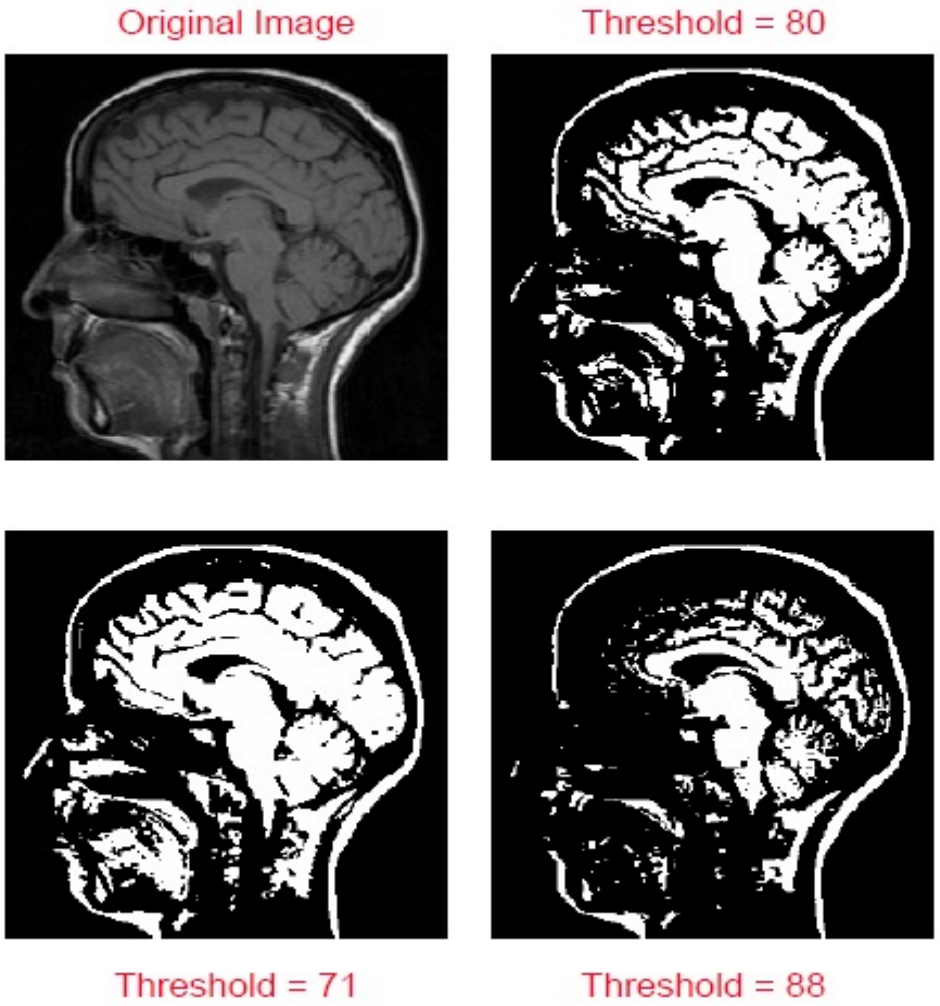


Results

- Segmentation based on threshold estimation using preceding algorithm



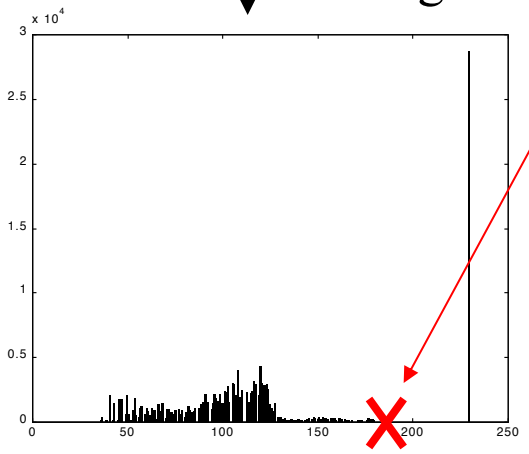
MRI Example



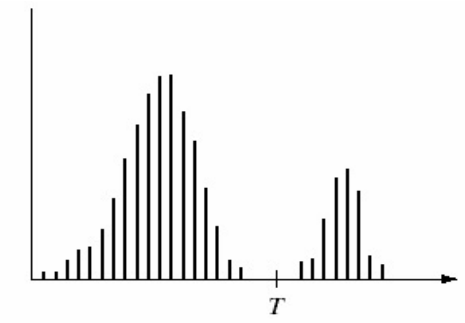
Thresholding

thresholding

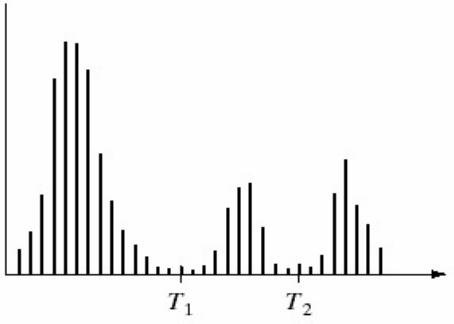
histogram



single threshold

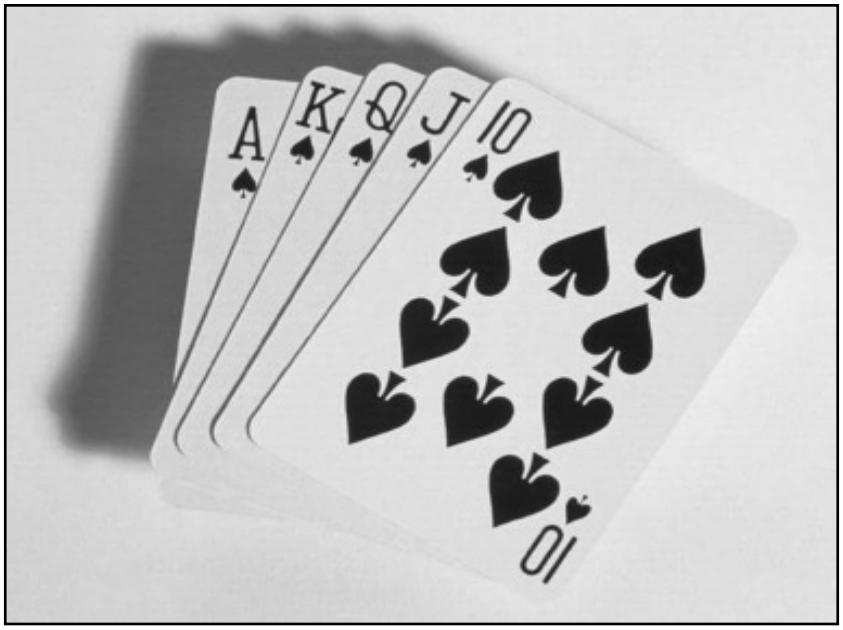


multiple thresholds

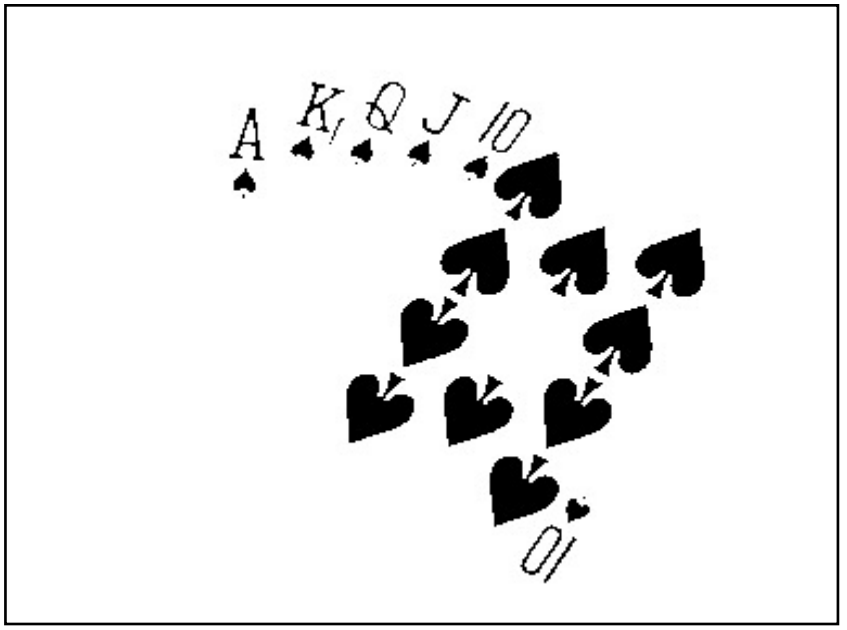


Thresholding Example

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



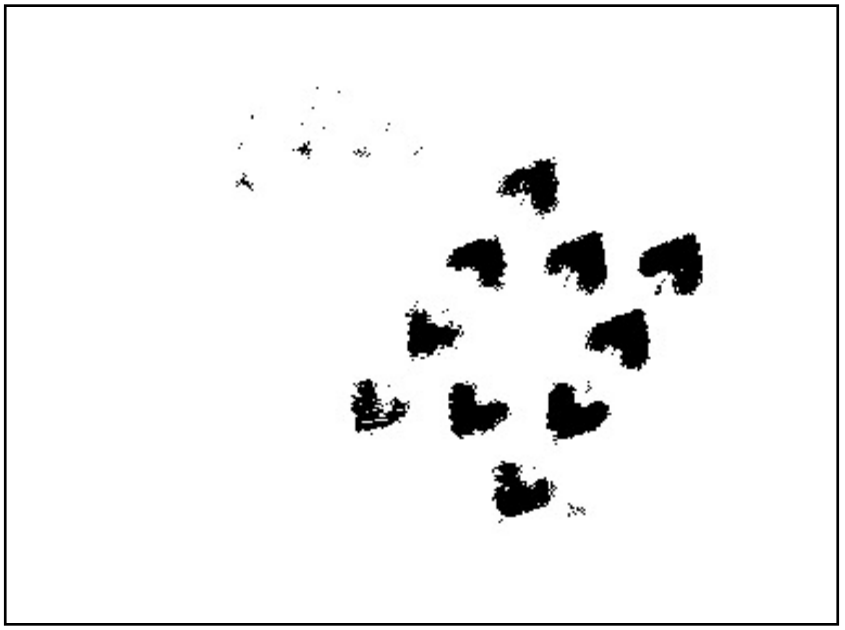
Original Image



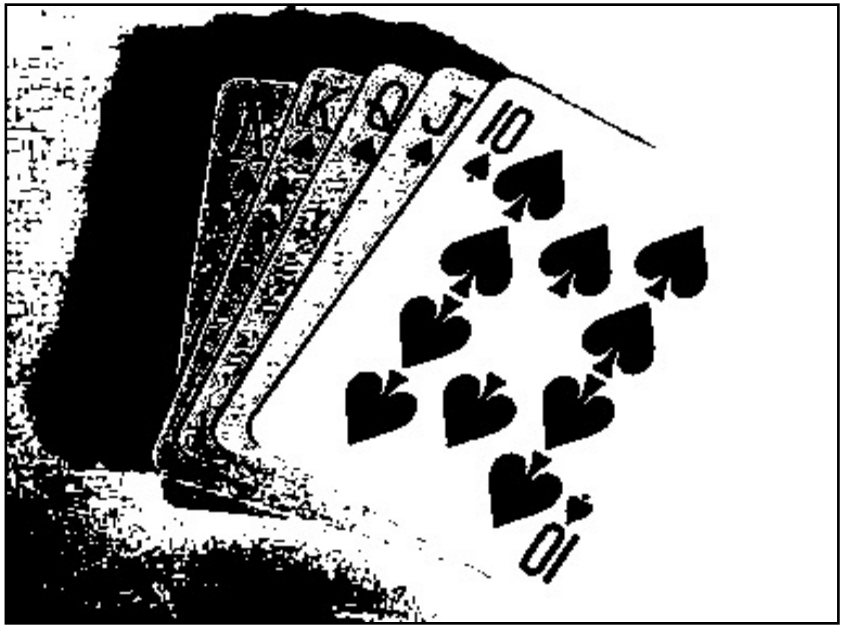
Thresholded Image

But Be Careful

- If you get the threshold wrong the results can be disastrous



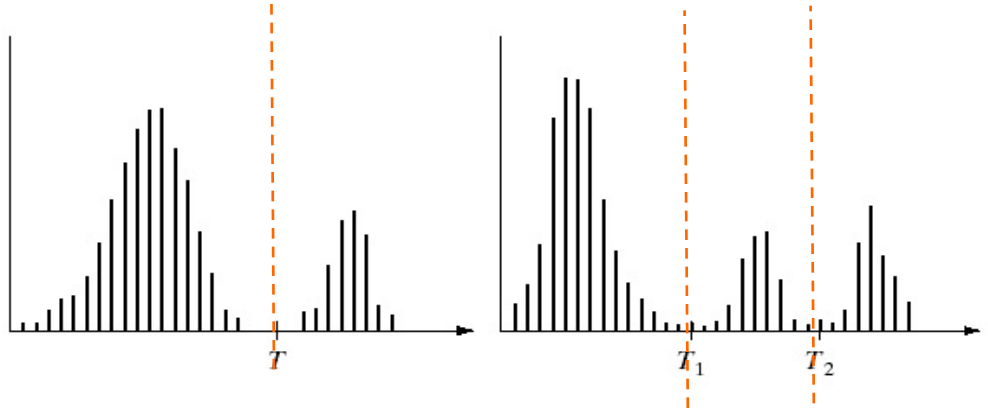
Threshold Too Low



Threshold Too High

Problems With Single Value Thresholding

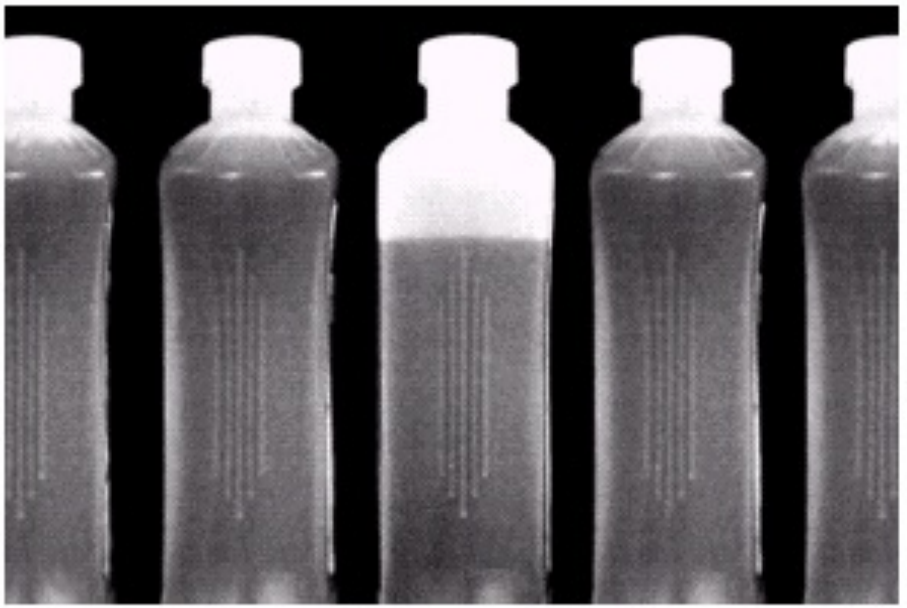
- Single value thresholding only works for bimodal histograms.
- Images with other kinds of histograms need more than a single threshold – multilevel thresholding



- Region growing is the preferred method for multimodal histograms

Problems With Single Value Thresholding

- Let's say we want to isolate the contents of the bottles
- Think about what the histogram for this image would look like
- What would happen if we used a single threshold value?

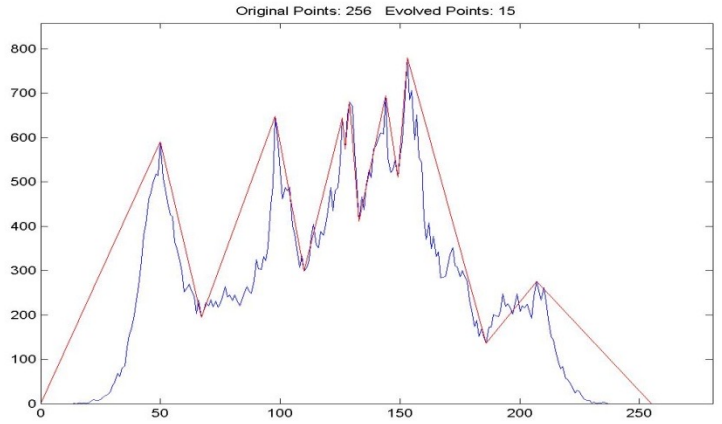


$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \quad \text{(object point)} \\ 0 & \text{if } f(x,y) \le T \quad \text{(background point)} \end{cases}$$

T : global thresholding

Multilevel Thresholding

Original Image of Lena



Histogram of Lena

Multiple thresholding

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

Image after segmentation – we get a outline of her face, hat, shadow, etc

Optimum Thresholding

- In optimal thresholding, a criterion function is devised that yields some measure of separation between regions.
  - A criterion function is calculated for each intensity and that maximizes this function is chosen as the threshold.
- Otsu's thresholding chooses the threshold to **minimize the intraclass variance** of the thresholded black and white pixels by **maximizing inter-class variance**.
  - Formulated as discriminant analysis: a particular criterion function is used as a measure of statistical separation.

Otsu Thresholding Method

- Based on a very simple idea: Find the threshold that minimizes the weighted within-class variance.
- This turns out to be the same as maximizing the between-class variance.
- Operates directly on the gray level histogram [e.g. 256 numbers, P(i)], so it's fast (once the histogram is computed).
- used it with considerable success in "murky" situations.

Otsu Assumptions

- Histogram (and the image) are bimodal.
- [I=imread('1.png'); figure, imhist(I);]
- No use of spatial coherence, nor any other notion of object structure.
- Assumes stationary statistics, but can be modified to be locally adaptive. (exercises)
- Assumes uniform illumination (implicitly), so the bimodal brightness behaviour arises from object appearance differences only.

Optimum Global Thresholding Using Otsu's Method

- Principle: maximizing the between-class variance

Let $\{0,\ 1,\ 2,\ ...,\ L-1\}$ denote the L distinct intensity levels in a digital image of size $M \times N$ pixels, and let n_i denote the number of pixels with intensity i.

$$p_i = n_i / MN \quad \text{and} \quad \sum_{i=0}^{L-1} p_i = 1$$

k is a threshold value, $C_1 \rightarrow [0, k],\ C_2 \rightarrow [k+1, L-1]$

$$P_1(k) = \sum_{i=0}^{k} p_i \quad \text{and} \quad P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$$

Optimum Global Thresholding Using Otsu's Method

The mean intensity value of the pixels assigned to class C_1 is

$$m_1(k) = \sum_{i=0}^{k} iP(i / C_1) = \frac{1}{P_1(k)} \sum_{i=0}^{k} ip_i$$

The mean intensity value of the pixels assigned to class C_2 is

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

$$P_1 m_1 + P_2 m_2 = m_G \quad \text{(Global mean value)}$$

Optimum Global Thresholding Using Otsu's Method

Between-class variance, σ_B^2 is defined as

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2$$

Optimum Global Thresholding Using Otsu's Method

The optimum threshold is the value, k*, that maximizes

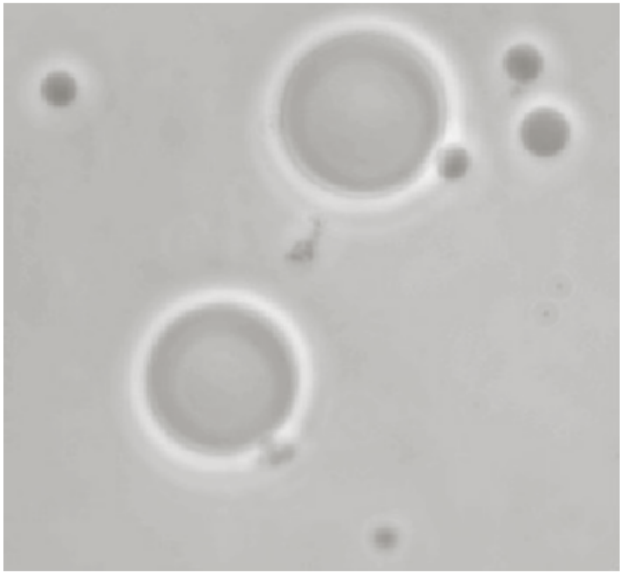
$$\sigma_B^2(k^*), \qquad \sigma_B^2(k^*) = \max_{0 \le k \le L-1} \sigma_B^2(k)$$

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

$$\text{Separability measure} \quad \eta = \frac{\sigma_B^2}{\sigma_G^2}$$

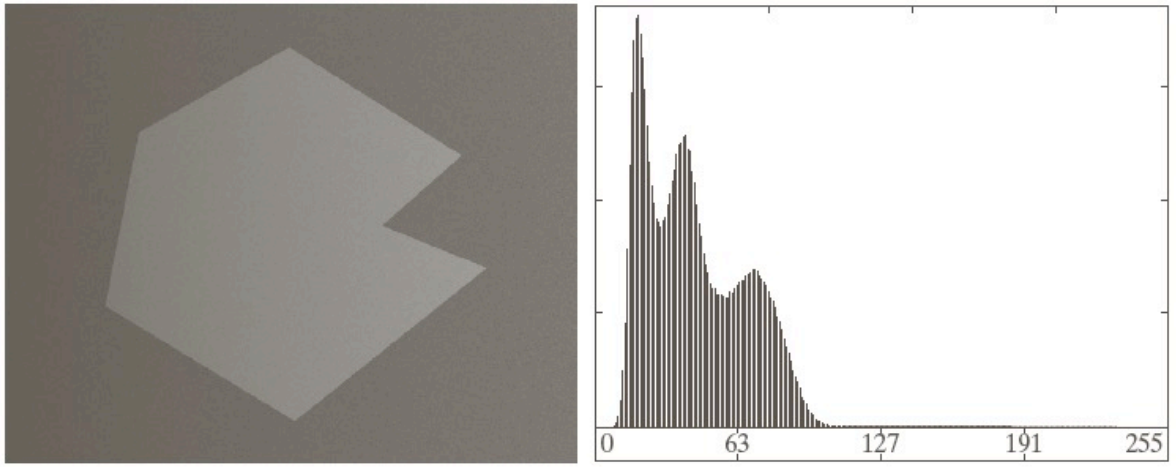
Otsu's Algorithm: Summary

1. Compute the normalized histogram of the input image. Denote the components of the histogram by p_i, i=0, 1, …, L-1.
2. Compute the cumulative sums, $P_1(k)$, for k = 0, 1, …, L-1.
3. Compute the cumulative means, m(k), for k = 0, 1, …, L-1.
4. Compute the global intensity mean, m_G.
5. Compute the between-class variance, for k = 0, 1, …, L-1.
6. Obtain the Otsu's threshold, k*.
7. Obtain the separability measure.



a b
c d

FIGURE 10.39
(a) Original image.
(b) Histogram (high peaks were clipped to highlight details in the lower values).
(c) Segmentation result using the basic global algorithm from Section 10.3.2.
(d) Result obtained using Otsu's method.
(Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)

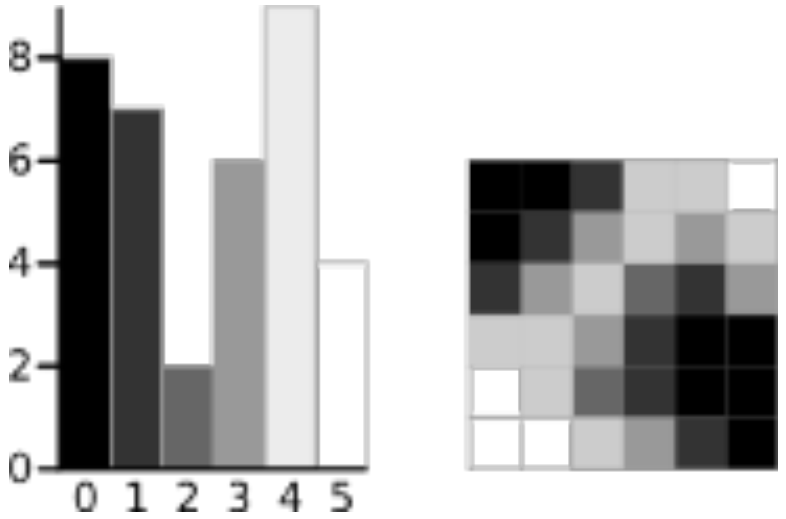


a b c
d e f

FIGURE 10.46 (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (e) Image subdivided into six subimages. (f) Result of applying Otsu's method to each subimage individually.

Cont..

- Otsu's thresholding method involves iterating through all the possible threshold values and calculate a measure of spread for the pixel levels each side of the threshold, i.e. the pixels that either fall in foreground or background.
- The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

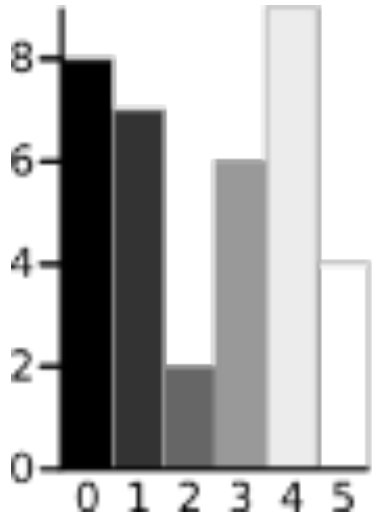


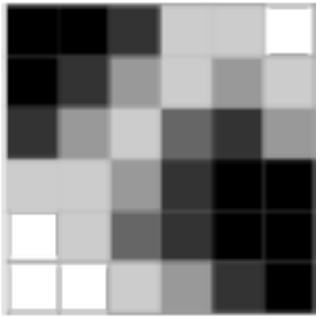
A 6-level greyscale image and its histogram

- The algorithm will be demonstrated using the simple 6x6 image shown below. The histogram for the image is shown next to it. To simplify the explanation, only 6 greyscale levels are used.

Cont..

- It works well with images that have a bi-modal histogram (those with two distinct regions).
- The calculations for finding the foreground and background variances (the measure of spread) for a single threshold are now shown. In this case the **threshold value is 3**.



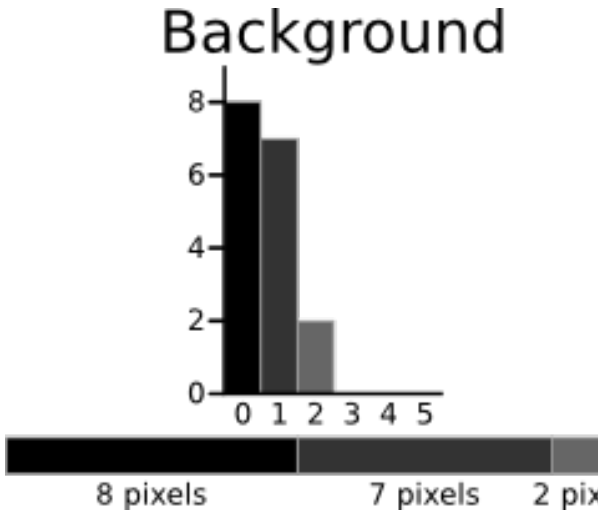


r_0	n
0	8
1	7
2	2
3	6
4	9
5	4
	36

Cont..

- The calculations for finding the foreground and background variances (the measure of spread) for a single threshold are now shown. In this case the **threshold value is 3**.

r_0	n
0	8
1	7
2	2
3	6
4	9
5	4
	36



=(8+7+2)/36 = 17/36 = 0.4722;

$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

= 11/17=0.6471;

$$\text{Mean} \quad \mu_b = \frac{(0\times 8)+(1\times 7)+(2\times 2)}{17} = 0.6471$$

$$\text{Variance} \quad \sigma_b^2 = \frac{((0-0.6471)^2\times 8)+((1-0.6471)^2\times 7)+((2-0.6471)^2\times 2)}{17}$$

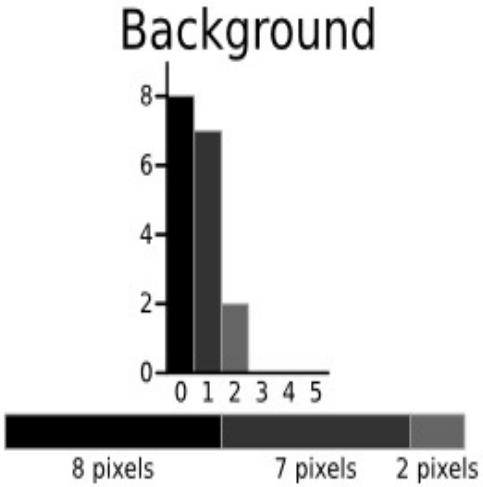
$$= \frac{(0.4187\times 8)+(0.1246\times 7)+(1.8304\times 2)}{17}$$

$$= 0.4637$$

=(0.4187x8)+(0.1245x7)+(1.8303x2)/17
=(3.3496+0.8715+3.6607)/17
=7.8818/17
=0.4636

Cont..

- T= threshold value is 3.



$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

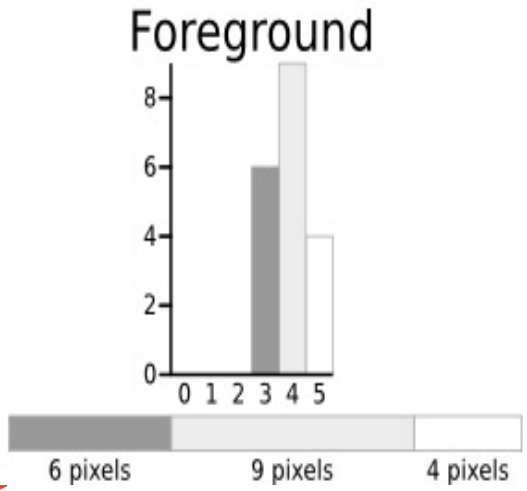
$$\text{Mean} \quad \mu_b = \frac{(0\times 8)+(1\times 7)+(2\times 2)}{17} = 0.6471$$

$$\text{Variance} \quad \sigma_b^2 = \frac{((0-0.6471)^2\times 8)+((1-0.6471)^2\times 7)+((2-0.6471)^2\times 2)}{17}$$

$$= \frac{(0.4187\times 8)+(0.1246\times 7)+(1.8304\times 2)}{17}$$

$$= 0.4637$$

r_0	n
0	8
1	7
2	2
3	6
4	9
5	4
	36



$$\text{Weight} \quad W_f = \frac{6+9+4}{36} = 0.5278$$

19/36=

$$\text{Mean} \quad \mu_f = \frac{(3\times 6)+(4\times 9)+(5\times 4)}{19} = 3.8947$$

(18+36+20)/19=74/19

$$\text{Variance} \quad \sigma_f^2 = \frac{((3-3.8947)^2\times 6)+((4-3.8947)^2\times 9)+((5-3.8947)^2\times 4)}{19}$$

$$= \frac{(4.8033\times 6)+(0.0997\times 9)+(4.8864\times 4)}{19}$$

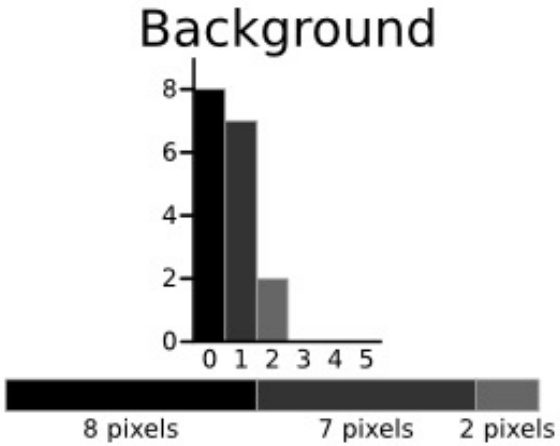
$$= 0.5152$$

=[(0.8005x6)+(0.0111x9)+(1.2217x4)]/19

= (4.803+ 0. 0999 + 4.8868)/19

=9.7897/19= 0.5152

Cont..



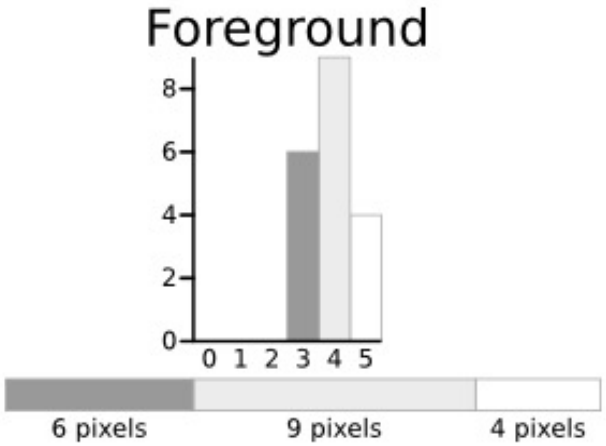
$$\text{Weight} \quad W_b = \frac{8+7+2}{36} = 0.4722$$

$$\text{Mean} \quad \mu_b = \frac{(0\times 8)+(1\times 7)+(2\times 2)}{17} = 0.6471$$

$$\text{Variance} \quad \sigma_b^2 = \frac{((0-0.6471)^2\times 8)+((1-0.6471)^2\times 7)+((2-0.6471)^2\times 2)}{17}$$

$$= \frac{(0.4187\times 8)+(0.1246\times 7)+(1.8304\times 2)}{17}$$

$$= 0.4637$$



$$\text{Weight} \quad W_f = \frac{6+9+4}{36} = 0.5278$$

$$\text{Mean} \quad \mu_f = \frac{(3\times 6)+(4\times 9)+(5\times 4)}{19} = 3.8947$$

$$\text{Variance} \quad \sigma_f^2 = \frac{((3-3.8947)^2\times 6)+((4-3.8947)^2\times 9)+((5-3.8947)^2\times 4)}{19}$$

$$= \frac{(4.8033\times 6)+(0.0997\times 9)+(4.8864\times 4)}{19}$$

$$= 0.5152$$

The next step is to calculate the 'Within-Class Variance'. This is simply the sum of the two variances multiplied by their associated weights.

$$\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 = 0.4722 * 0.4637 + 0.5278 * 0.5152$$

$$= 0.4909$$

= 0.2190+ 0.2719

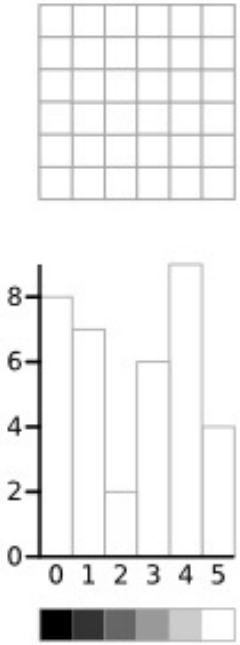
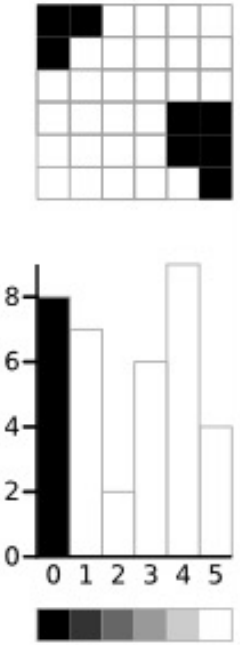
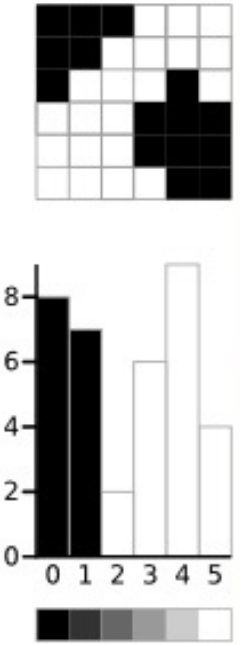
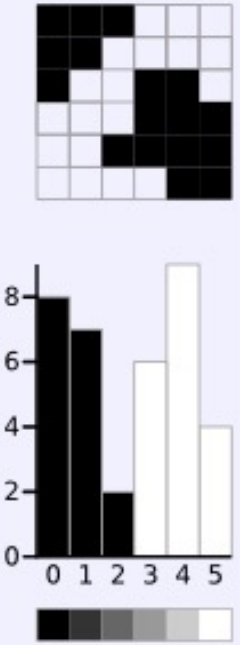
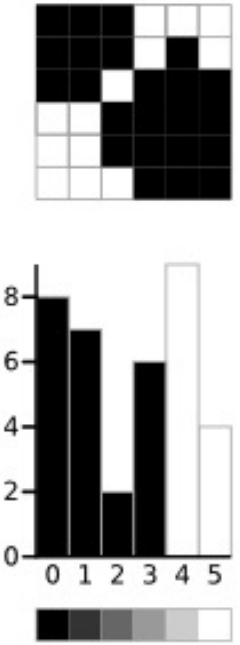
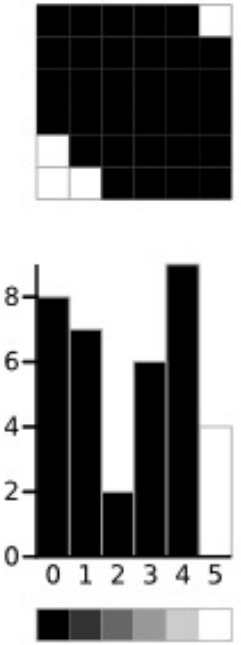
= 0.4909

Cont..

- This final value is the 'sum of weighted variances' for the threshold value 3.
- This same calculation needs to be performed for all the possible threshold values(T) 0 to 5.
- The table next slide shows the results for these calculations.
- The highlighted column shows the values for the threshold calculated above.

Cont..

r_0	n
0	8
1	7
2	2
3	6
4	9
5	4
	36

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
Weight, Background	$W_b = 0$	$W_b = 0.222$	$W_b = 0.4167$	$W_b = 0.4722$	$W_b = 0.6389$	$W_b = 0.8889$
Mean, Background	$M_b = 0$	$M_b = 0$	$M_b = 0.4667$	$M_b = 0.6471$	$M_b = 1.2609$	$M_b = 2.0313$
Variance, Background	$\sigma^2_b = 0$	$\sigma^2_b = 0$	$\sigma^2_b = 0.2489$	$\sigma^2_b = 0.4637$	$\sigma^2_b = 1.4102$	$\sigma^2_b = 2.5303$
Weight, Foreground	$W_f = 1$	$W_f = 0.7778$	$W_f = 0.5833$	$W_f = 0.5278$	$W_f = 0.3611$	$W_f = 0.1111$
Mean, Foreground	$M_f = 2.3611$	$M_f = 3.0357$	$M_f = 3.7143$	$M_f = 3.8947$	$M_f = 4.3077$	$M_f = 5.000$
Variance, Foreground	$\sigma^2_f = 3.1196$	$\sigma^2_f = 1.9639$	$\sigma^2_f = 0.7755$	$\sigma^2_f = 0.5152$	$\sigma^2_f = 0.2130$	$\sigma^2_f = 0$
Within Class Variance	$\sigma^2_W = 3.1196$	$\sigma^2_W = 1.5268$	$\sigma^2_W = 0.5561$	$\sigma^2_W = 0.4909$	$\sigma^2_W = 0.9779$	$\sigma^2_W = 2.2491$

Cont..

- It can be seen that for the T=3, has the lowest sum of weighted variances. ∴ this is the final selected threshold. All pixels with a level less than 3 are background, all those with a level equal to or greater than 3 are foreground. As the images in the table show, this threshold works well.
- This approach for calculating Otsu's threshold is useful for explaining the theory, but it is computationally intensive, especially if you have a full 8-bit greyscale. The next section shows a faster method of performing the calculations which is much more appropriate for implementations.

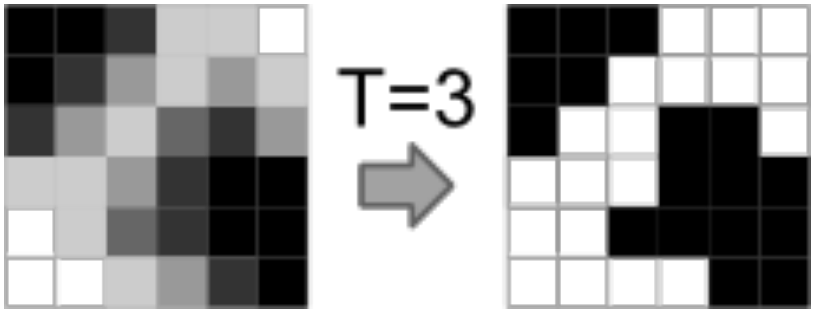


Fig. Result of Otsu's Method

A Faster Approach

- By a bit of manipulation, you can calculate what is called the between class variance, which is far quicker to calculate.
- Luckily, the threshold with the maximum between class variance also has the minimum within class variance.
- So it can also be used for finding the best threshold and therefore due to being simpler is a much better approach to use.

Cont..

- By a bit of manipulation, you can calculate what is called the between class variance, which is far quicker to calculate.
- Luckily, the threshold with the maximum between class variance also has the minimum within class variance.
- So it can also be used for finding the best threshold and therefore due to being simpler is a much better approach to use.
- Simplification of Otsu's threshold calculation

$$\begin{aligned}
&\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 \quad \text{(as seen above)} \\
&\text{Between Class Variance} \quad \sigma_B^2 = \sigma^2 - \sigma_W^2 \\
&\qquad = W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2 \quad (\text{where } \mu = W_b\,\mu_b + W_f\,\mu_f) \\
&\qquad = W_b\,W_f\,(\mu_b - \mu_f)^2
\end{aligned}$$

Cont..

- Simplification of Otsu's threshold calculation

$$
\begin{aligned}
\text{Within Class Variance} \quad \sigma_W^2 &= W_b\,\sigma_b^2 + W_f\,\sigma_f^2 \quad \text{(as seen above)} \\
\text{Between Class Variance} \quad \sigma_B^2 &= \sigma^2 - \sigma_W^2 \\
&= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2 \quad (\text{where } \mu = W_b\,\mu_b + W_f\,\mu_f) \\
&= W_b\,W_f\,(\mu_b - \mu_f)^2
\end{aligned}
$$

Cont..

- Simplification of Otsu's threshold calculation

$$\text{Within Class Variance} \quad \sigma_W^2 = W_b\,\sigma_b^2 + W_f\,\sigma_f^2 \quad \text{(as seen above)}$$

$$\begin{aligned}\text{Between Class Variance} \quad \sigma_B^2 &= \sigma^2 - \sigma_W^2 \\ &= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2 \quad (\text{where } \mu = W_b\,\mu_b + W_f\,\mu_f) \\ &= W_b\,W_f\,(\mu_b - \mu_f)^2\end{aligned}$$

$$\begin{aligned}\text{Between Class Variance} \quad \sigma_B^2 &= \sigma^2 - \sigma_W^2 \\ &= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2 \quad (\text{where } \mu = W_b\,\mu_b + W_f\,\mu_f) \\ &= W_b\,W_f\,(\mu_b - \mu_f)^2 \\ &= 0.4722 \times 0.5278\,(\,0.6471 - 3.8947\,)^2 \\ &= 0.2492 \times 10.5469 \\ &= 2.6287\end{aligned}$$

Cont..

- Simplification of Otsu's threshold calculation

$$\begin{aligned}
\text{Within Class Variance} \quad \sigma_W^2 &= W_b\,\sigma_b^2 + W_f\,\sigma_f^2 \quad \text{(as seen above)} \\
\text{Between Class Variance} \quad \sigma_B^2 &= \sigma^2 - \sigma_W^2 \\
&= W_b(\mu_b - \mu)^2 + W_f(\mu_f - \mu)^2 \quad (\text{where } \mu = W_b\,\mu_b + W_f\,\mu_f) \\
&= W_b\,W_f\,(\mu_b - \mu_f)^2
\end{aligned}$$

- The table below shows the different variances for each threshold value.

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
Within Class Variance	$\sigma^2_W = 3.1196$	$\sigma^2_W = 1.5268$	$\sigma^2_W = 0.5561$	$\sigma^2_W = 0.4909$	$\sigma^2_W = 0.9779$	$\sigma^2_W = 2.2491$
Between Class Variance	$\sigma^2_B = 0$	$\sigma^2_B = 1.5928$	$\sigma^2_B = 2.5635$	$\sigma^2_B = 2.6287$	$\sigma^2_B = 2.1417$	$\sigma^2_B = 0.8705$

Java Implementation

- ```
  // Calculate histogram
  int ptr = 0;
  while (ptr < srcData.length) {
     int h = 0xFF & srcData[ptr];
     histData[h] ++;
     ptr ++;
  }
  ```

- ```
  // Total number of pixels
  int total = srcData.length;

  float sum = 0;
  for (int t=0 ; t<256 ; t++) sum += t * histData[t];

  float sumB = 0;
  int wB = 0;
  int wF = 0;

  float varMax = 0;
  threshold = 0;
  ```

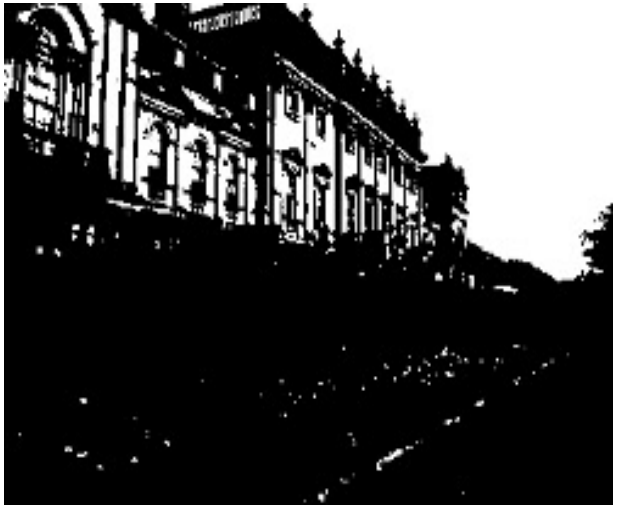
Java Implementation

```
for (int t=0 ; t<256 ; t++) {
   wB += histData[t];                // Weight Background
   if (wB == 0) continue;

   wF = total - wB;                  // Weight Foreground
   if (wF == 0) break;

   sumB += (float) (t * histData[t]);

   float mB = sumB / wB;             // Mean Background
   float mF = (sum - sumB) / wF;     // Mean Foreground

   // Calculate Between Class Variance
   float varBetween = (float)wB * (float)wF * (mB - mF) * (mB - mF);

   // Check if new maximum found
   if (varBetween > varMax) {
      varMax = varBetween;
      threshold = t;
   }
}
```

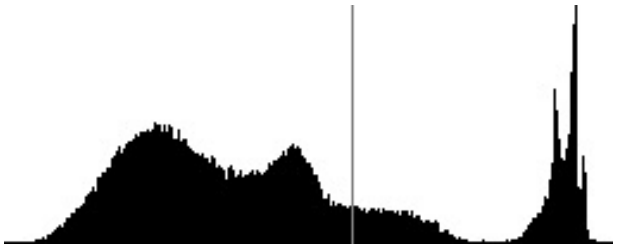
Cont..

Greyscale Image

Binary Image



Histogram

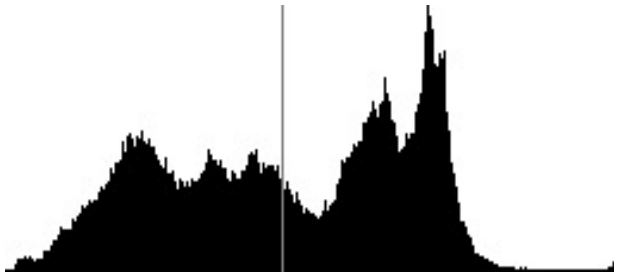


Cont..

Greyscale Image

Binary Image

Histogram

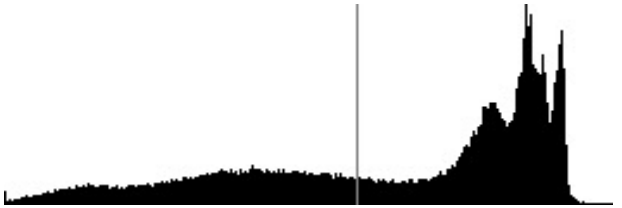


Cont..

Greyscale Image

Binary Image

Histogram



Variable Thresholding Based on Local Image Properties

Let σ_{xy} and m_{xy} denote the standard deviation and mean value of the set of pixels contained in a neighborhood S_{xy}, centered at coordinates (x, y) in an image. The local thresholds,

$$T_{xy} = a\sigma_{xy} + bm_{xy}$$

If the background is nearly constant,

$$T_{xy} = a\sigma_{xy} + bm$$

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T_{xy} \\ 0 & \text{if } f(x,y) \le T_{xy} \end{cases}$$

Variable Thresholding Based on Local Image Properties

A modified thresholding

$$g(x,y) = \begin{cases} 1 & \text{if } Q(\text{local parameters}) \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

e.g.,

$$Q(\sigma_{xy}, m_{xy}) = \begin{cases} \text{true} & \text{if } f(x,y) > a\sigma_{xy} \text{ AND } f(x,y) > bm_{xy} \\ \text{false} & \text{otherwise} \end{cases}$$

a b
c d

FIGURE 10.48
(a) Image from Fig. 10.43.
(b) Image segmented using the dual thresholding approach discussed in Section 10.3.6.
(c) Image of local standard deviations.
(d) Result obtained using local thresholding.

a=30
b=1.5
$m_{xy} = m_G$

Variable Thresholding Using Moving Averages

- Thresholding based on moving averages works well when the objects are small with respect to the image size
- Quite useful in document processing
- The scanning (moving) typically is carried out line by line in zigzag pattern to reduce illumination bias

Variable Thresholding Using Moving Averages

Let z_{k+1} denote the intensity of the point encountered in the scanning sequence at step $k+1$. The moving average (mean intensity) at this new point is given by

$$m(k+1) = \frac{1}{n}\sum_{i=k+2-n}^{k+1} z_i = m(k) + \frac{1}{n}(z_{k+1} - z_k)$$

where n denotes the number of points used in computing the average and $m(1) = z_1 / n$, the border of the image were padded with n-1 zeros.

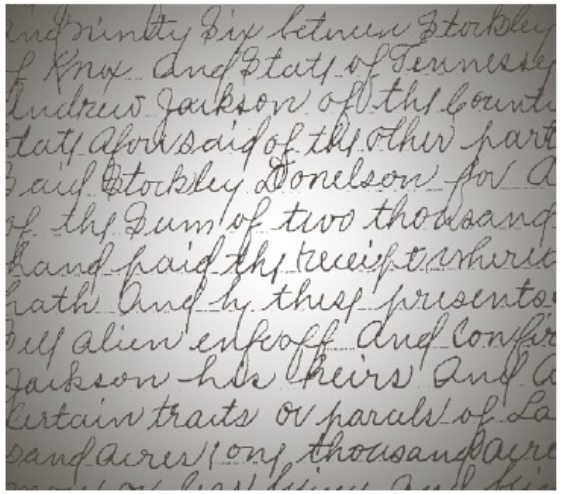
Variable Thresholding Using Moving Averages

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T_{xy} \\ 0 & \text{if } f(x,y) \le T_{xy} \end{cases}$$

$$T_{xy} = bm_{xy}$$

Cont..

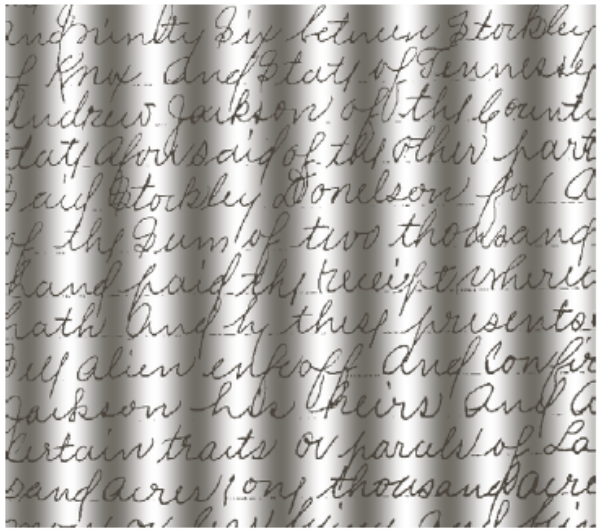
N = 20
b=0.5



a b c

FIGURE 10.49 (a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

Cont..



a b c

FIGURE 10.50 (a) Text image corrupted by sinusoidal shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

Region based Segmentation

Here $P(R_i)$ is logical predicate defined over all points in R_i

(a) Every pixel must be in a region

(b) All the points in a region must be "connected"

(c) Regions must be disjoint

(d) For example $P(R_i) =$ TRUE if all the pixels in R_i have the same gray level

(e) Regions R_i and R_j are different in some sense

Region based segmentation can be carried in four different ways

Region growing Region splitting Region merging Split and merge

Labelling Algorithm

- The connected components labelling operator scans the image by moving along a row until it comes to a point p (where p denotes the pixel to be labelled at any stage in the scanning process) for which V={1}. When this is true, it examines the four neighbours of p which have already been encountered in the scan (in the case of 8-connectivity, the neighbours (i) to the left of p, (ii) above it, and (iii and iv) the two upper diagonal terms). Based on this information, the labelling of p occurs as follows:
  - If all four neighbours are 0, assign a new label to p, else
  - if only one neighbour has V={1}, assign its label to p, else
  - if more than one of the neighbours have V={1}, assign one of the labels to p and make a note of the equivalences.
- After completing the scan, the equivalent label pairs are sorted into equivalence classes and a unique label is assigned to each class.
- As a final step, a second scan is made through the image, during which each label is replaced by the label assigned to its equivalence classes. For display, the labels might be different grey-levels or colours.

Cont..

- Example (8-connectivity)

1	1		1	1	
	1	1			
		1	1		
1				1	
1					
1	1	1			

Original Binary Image

1	1		2	2	
	1	1			
		1	1		
3				1	
3					
3	3	3			

After first scan

Equivalent labels

{1, 2}

Cont..

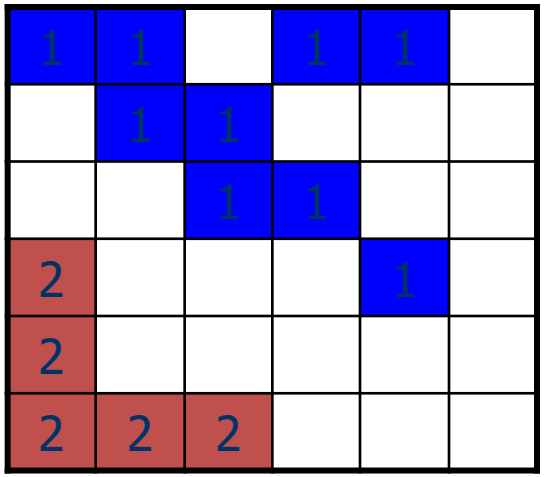
- Example (8-connectivity)

1	1		2	2	
	1	1			
		1	1		
3				1	
3					
3	3	3			

Equivalent labels

{1, 2}

After 1st scan



Final label

1: {1, 2}
2: {3}

After 2nd scan
(two individual objects/regions have been identified)

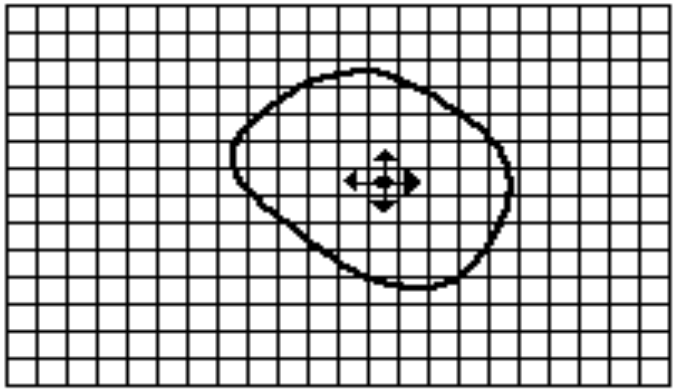
Region Growing

- Start from a set of seed points and from these points grow the regions by appending to each seed those neighboring pixels that have similar properties
- The selection of the seed points depends on the problem
- The selection of similarity criteria depends on the problem under consideration and the type of image data that is available
- Formulation of a “stopping rule”: Growing a region should stop when no more pixels satisfy the criteria for inclusion in that region

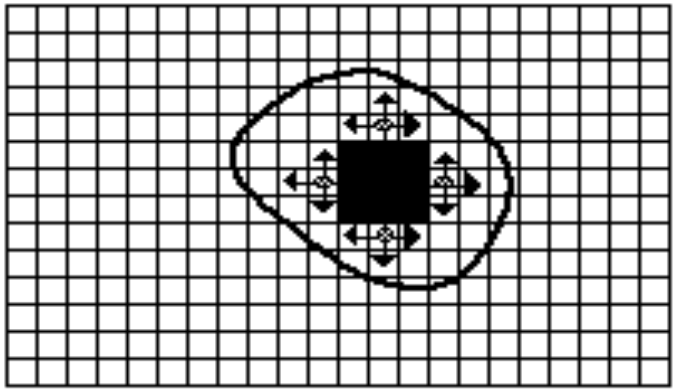
Region Growing Algorithm

1. Select a seed point (x_1,y_1) (an arbitrary pixel from where we begin) and grow regions from this seed pixel
2. We now examine the nearest neighbors (4-connectivity or 8-connectivity) of (x_1,y_1) one by one
3. The neighboring pixel is accepted in the same region as (x_1,y_1) if they together satisfy the homogeneity property of a region
4. Once a new pixel is (x_2,y_2) is accepted as a member of the current region, the neighbors of this new pixel are examined for connectivity
5. This procedure goes on recursively until no new pixel is accepted
6. All the pixels of the current region are given a unique label
7. A new seed pixel is chosen and the same procedure is repeated
8. We continue doing this until all the pixels are assigned to some region or the other

Cont..



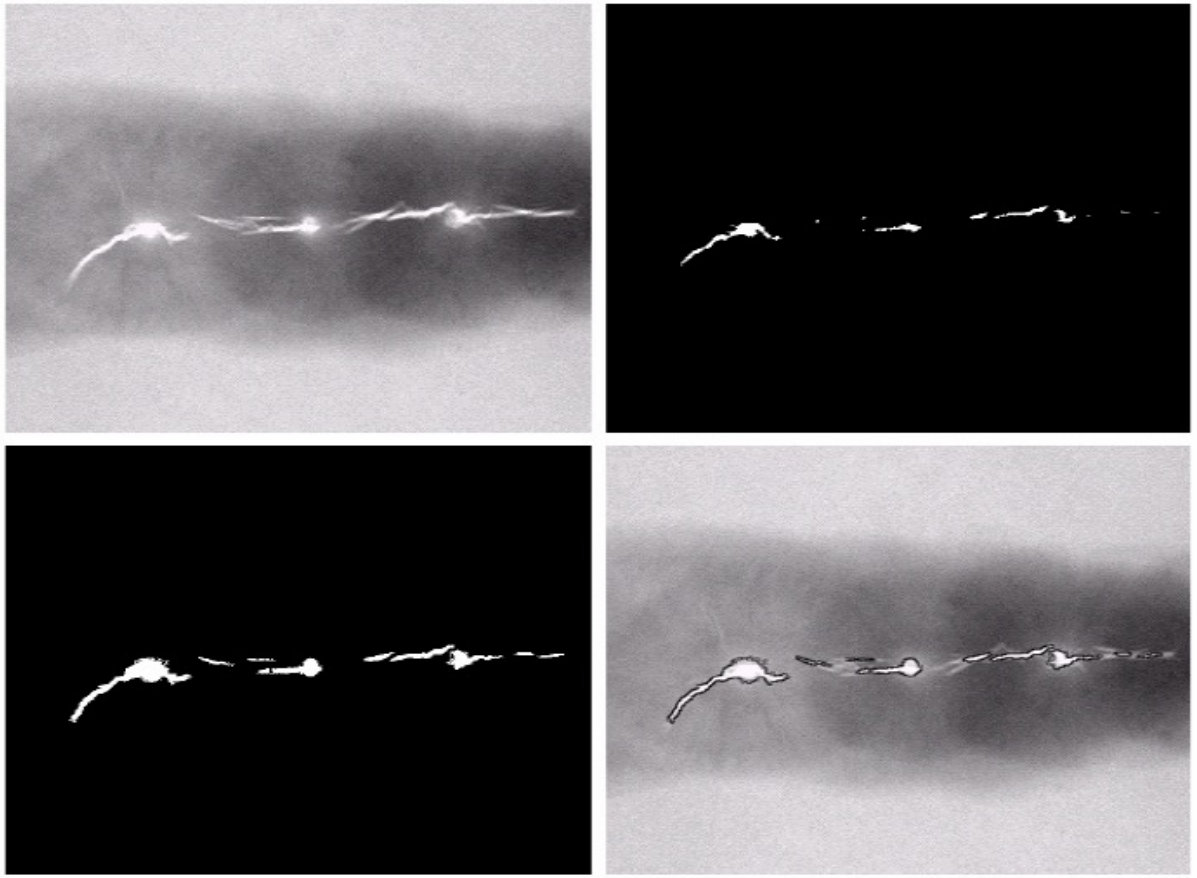
(a) Start of Growing a Region



(b) Growing Process After a Few Iterations

Region Growing - Application

Application of region growing to weld inspection:

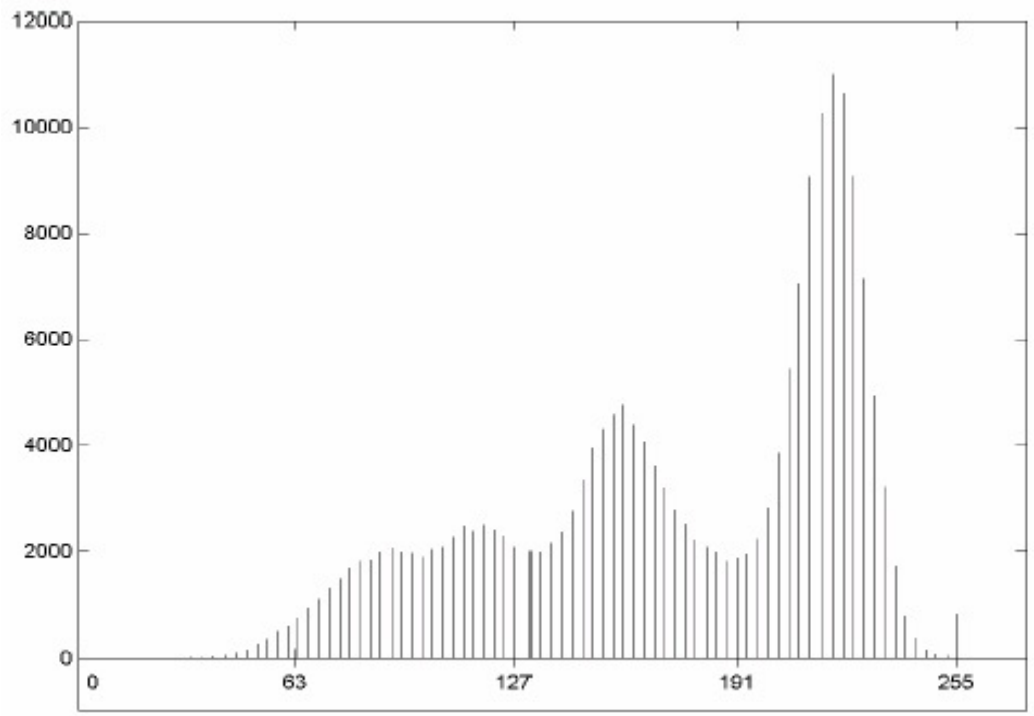


a b
c d

a. Image showing defective welds
b. Seed points
c. Result of region growing
d. Boundaries of segmented defective welds

Region Growing - Application

- As mentioned multimodal histograms are best segmented using region growing
- Use of connectivity was fundamental in solving this problem



Region Growing

Example threshold =3

5	6	6	6	7	7	6	6
5	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
1	3	2	3	3	2	4	7
0	0	1	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

8 x 8 Image Original Image

5	6	6	6	7	7	6	6
5	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
1	3	2	3	3	2	4	7
0	0	1	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

Segmented Image

Change of Seed Pixel

Result of change in the seed pixel which gives a completely different segmented regions

5	6	6	6	7	7	6	6
5	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
1	3	2	3	3	2	4	7
0	0	1	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

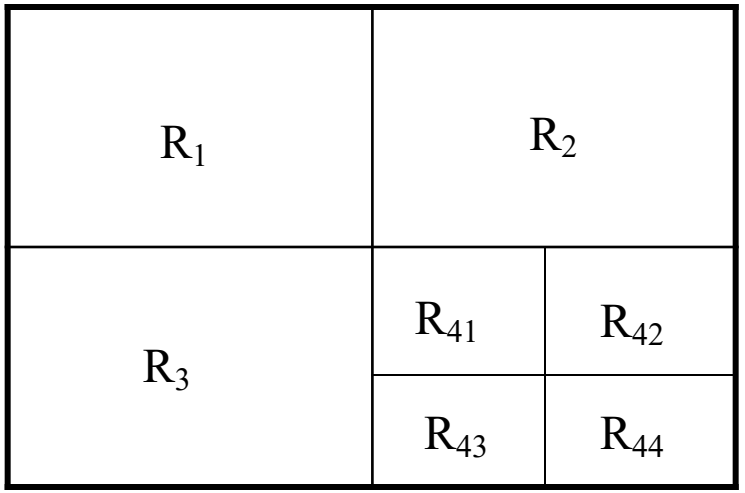
Limitations of Region Growing

- Selection of seed pixel
- Regions growing is based on selection of seed pixel
- Slowest method of segmentation as at a time only one pixel is assigned as a member of the current region
- Solution: Region Splitting

Region Splitting

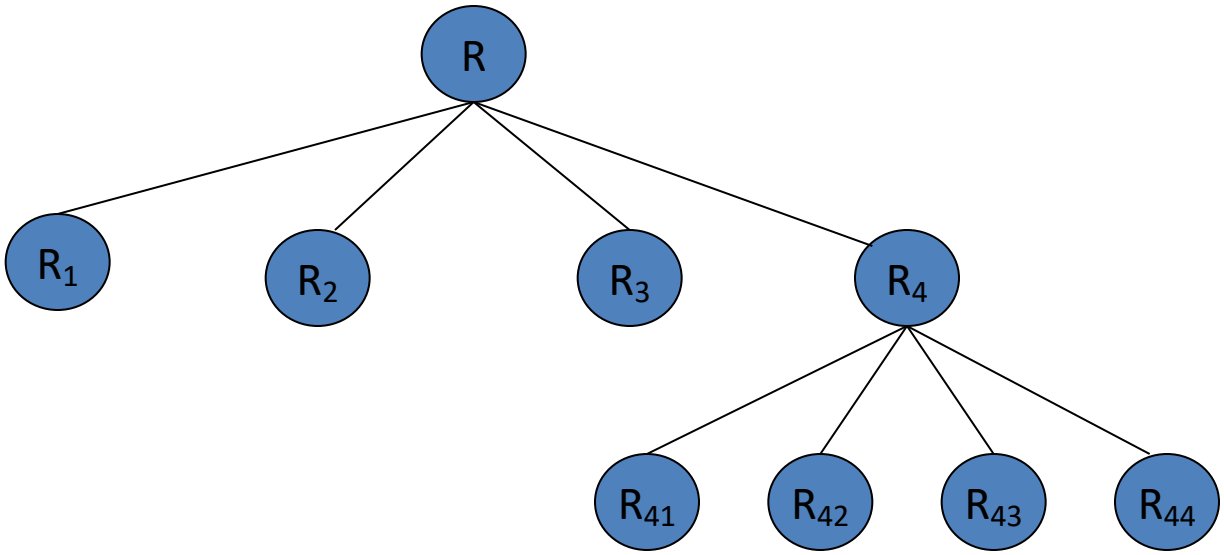
Examine homogeneity property of the regions instead of pixels.

- If the gray levels present in the image do not satisfy the property split the region into four equal quadrants.
- If the property is satisfied leave the regions as it is
- This is done recursively until all the regions satisfy the property
- This method is applicable to images whose size is some integer power of 2
- To explain this in terms of graph theory, we call each region a node



Region Splitting

- This particular splitting technique has a convenient representation in the form of a quad tree
- Here image represents the entire image and hence R is the parent node
- This parent node is split into four leaf nodes R_1, R_2, R_3 and R_4.
- Of these leaf nodes only R_4 does not contain pixels which satisfy some common property and hence is split again into R_{41}, R_{42}, R_{43} and R_{44}.
- The Quad tree for this division as shown:

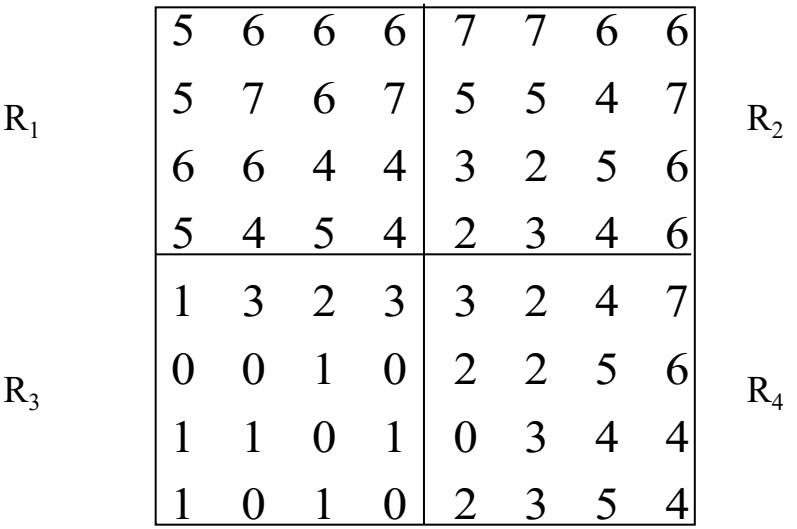


Region Splitting

- Consider the same 8*8 image that we worked with region growing
- Let the threshold =3
- Since the entire image does not satisfy the condition of threshold <=3, we split the image into four regions

5	6	6	6	7	7	6	6
5	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
1	3	2	3	3	2	4	7
0	0	1	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

8 x 8 Image Original Image



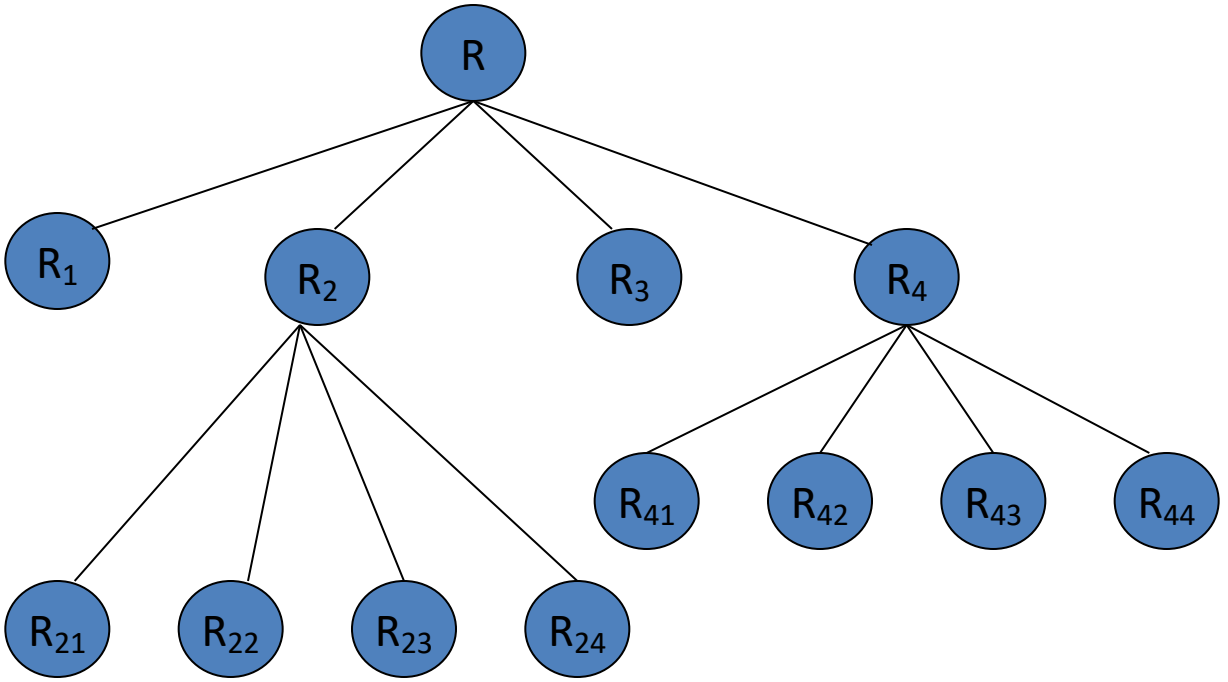
Split 1

Region Splitting

- In split 1, R_1 and R_3 satisfy the homogeneity condition, while R_2 and R_4 do not satisfy the homogeneity condition
- Therefore we split R_2 and R_4 into 4 regions each
- We now see that each region satisfies the homogeneity condition.
- The quad tree is also shown:

5	6	6	6	7	7	6	6
5	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
1	3	2	3	3	2	4	7
0	0	1	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

Split 2



Region Merging

- This method is exactly opposite to that of region splitting
- This method is also applicable to image sizes of integer power of 2
- We start from the pixel level and consider each of them as a homogenous region
- At any level of merging, we check if the four adjacent homogenous regions arranged in a 2*2 fashion together satisfy the homogeneity property
- If yes they are merged to form a bigger region, otherwise the regions are left as they are

5	6	6	6	7	7	6	6
5	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
1	3	2	3	3	2	4	7
0	0	1	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

5	6	6	6	7	7	6	6
5	7	6	7	5	5	4	7
6	6	4	4	3	2	5	6
5	4	5	4	2	3	4	6
1	3	2	3	3	2	4	7
0	0	1	0	2	2	5	6
1	1	0	1	0	3	4	4
1	0	1	0	2	3	5	4

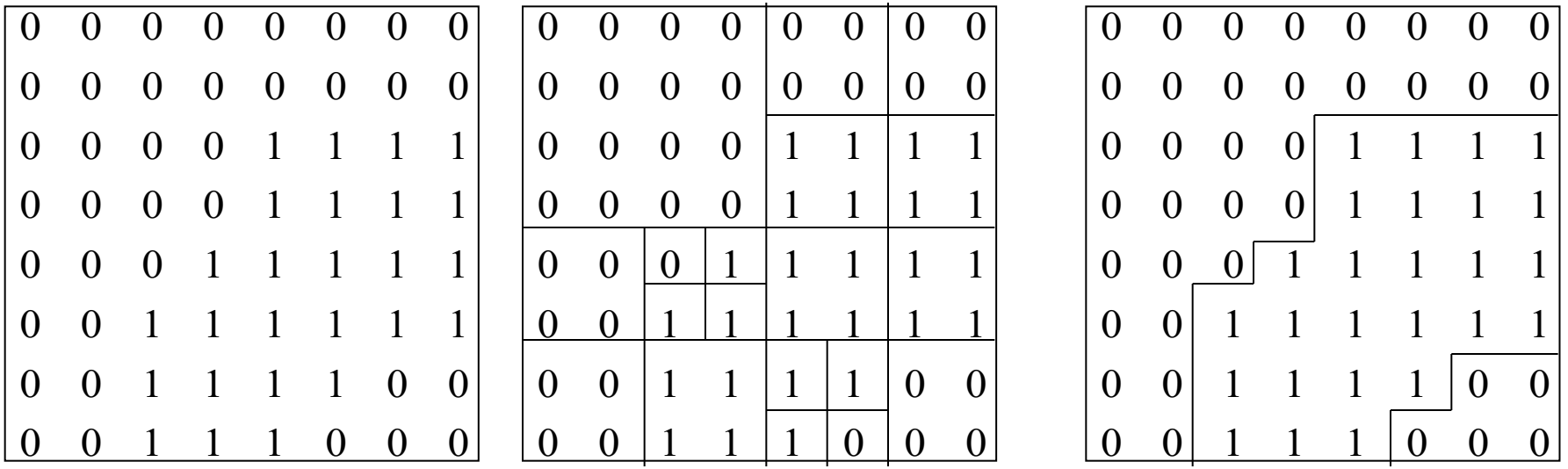
Split and Merge

- If homogenous regions are small, the region merging technique is superior to the region growing technique
- If most of the regions in an image are homogenous, the region splitting technique is preferred to the region merging method
- In most applications a combination of split and merge techniques are used together called split and merge technique
- Here we start with a rectangular regions of size a*b pixels

Algorithm

- Split the given region into four quadrants if the homogeneity is not satisfied
- Continue this subdivision for all new sub-images until the stopping criteria is reached (usually single pixel)
- Merge any adjacent regions for which the uniformity condition is true
- Stop when no further merging is possible

Split and Merge



Summary

- Segmentation attempts to partition the pixels of an image into groups that strongly correlate with the objects in an image
- Usually image segmentation is an initial and vital step in a series of processes aimed at overall image understanding
- Applications of image segmentation include: identifying objects in a scene for object-based measurements such as size and shape, identifying objects in a moving scene, identifying objects which are at different distances from a sensor, computer graphics, medical imaging etc.
- Applications of image segmentation include: identifying objects in a scene for object-based measurements such as size and shape, identifying objects in a moving scene, identifying objects which are at different distances from a sensor, computer graphics, medical imaging etc.
- Region growing starts from a set of seed points and from these points grow the regions by appending to each seed those neighboring pixels that have similar properties
- Region splitting and merging subdivides an image initially into a set of arbitrary, disjoint regions and then merge and/or split the regions in an attempt to satisfy the necessary conditions