KE5108: Developing Intelligent Systems for Performing Business Analytics
Lecture 1: Machine Vision I

Dr. Matthew **CHUA**, PhD
Institute of System Science, NUS

About Me…

- PhD, Medical Engineering from NUS Faculty of Engineering
- Research Interests: Medical Device, Robotics and Artificial Intelligence.
- Heading Research in ISS with over $1 million grant

About the Course

- Class information, lectures and materials can be found on IVLE.
- Do not email me questions, instead post it on the IVLE module forum so that we can all learn.

- Mid course CA presentation (in groups) – 5%

- Final Team CA assignment (in groups) - 25%

- Final Examination – 20%

Class Format

- 3 hour lecture (with breaks) followed by a 1-hour break, and then tutorial
- Questions by other students and forum postings (Participation marks)
- Mid course Team CA presentation
- Final Team CA project paper and presentation
- Final Examinations

Mid-course Team CA Presentation (5%)

- Same Grouping as the term project
- Present your detailed literature review and proposed approach to the Term Project - To receive feedback and criticism on your method for refinement
- Half the teams will present on Week 2 and the remaining in Week 3
- 25 minutes presentation and followed by Q&A (5 mins)
- Exceeding of time will be penalized

Team CA Assignment (25%)

- Choose ONLY ONE theme from either Theme 1 or Theme 2 shown in following two slides.
- You need to submit
  - 1 journal paper, *.doc or *.docx format, including all text, figures, tables, references.
  - 1 softcopy of your sourcecode written in any programming language.
  - Zip all your files into a single zipped file. Please submit only one ZIP file from each team.
- To be presented in powerpoint format during 5th Week, final journal paper to be submitted on 6th Week for grading.
  - 25 minutes presentation and followed by Q&A (5 mins)
  - Exceeding of time will be penalized
- Refer to IVLE for sample "Gold Standard" Papers from previous years

Theme 1: An better image processing project

- Choose any one of image processing projects done by Stanford EE368 students, https://web.stanford.edu/class/ee368/Project_Winter_1718/index.html. Most projects have shared their reports and source code in this website.
- Improve your chosen approach and develop a better solution with justifications in experiments.

Theme 2: Medical Image Analysis

- Choose any one of the medical image analysis challenges hosted on:

  https://grand-challenge.org/All_Challenges/

- You may look for existing codes (baseline) and improve on them to yield better results. Comparison must be side to side.

- Your team is also encourage to submit to the competition if you are confident.

Journal Paper Format

- Introduction
  - Open with background information of the problem case
  - Present literature review of other works and their limitations
  - Summary of how your work will tackle the limitations of others/solve the problem
- Methods and Materials / Modelling
  - Present the derivation of your model and explanations
- Computational Simulation
  - Present the data source and programme/simulation to be conducteed
- Results
  - Present the results of your model simulation
- Discussion and Conclusion
  - Compare your results with other models and highlight its effectiveness
  - Discuss implications, limitations and future work
- Refer to IVLE for “golden standard” paper by previous EBAC team

Course Outline

- Week 1: Machine Vision I
- Week 2: Machine Vision II
- Week 3: Machine Vision III
- Week 4: Applications of Machine Vision
- Week 5: CA Presentations and Discussion

Main References

- Sonka, M., Hlavac, V., & Boyle, R. (2014). *Image processing, analysis, and machine vision*. Cengage Learning.
- Rafael C.. Gonzalez, Richard E.. Woods, & Steven L.. Eddins. (2010). *Digital Image Processing Using MATLAB®*. McGraw Hill Education.
- Smith, S. W. (1997). The scientist and engineer's guide to digital signal processing.
- Orfanidis, S. J. (1995). *Introduction to signal processing*. Prentice-Hall, Inc.
- Nagrath, I. J. (2006). *Control systems engineering*. New Age International.
- Ogata, K., & Yang, Y. (1970). Modern control engineering.
- ME5405 Machine Vision. NUS

Contents of this lecture

1. Introduction
2. Digital Imaging Fundamentals
3. Basic Image Processing
4. Binary Machine Vision

Introduction

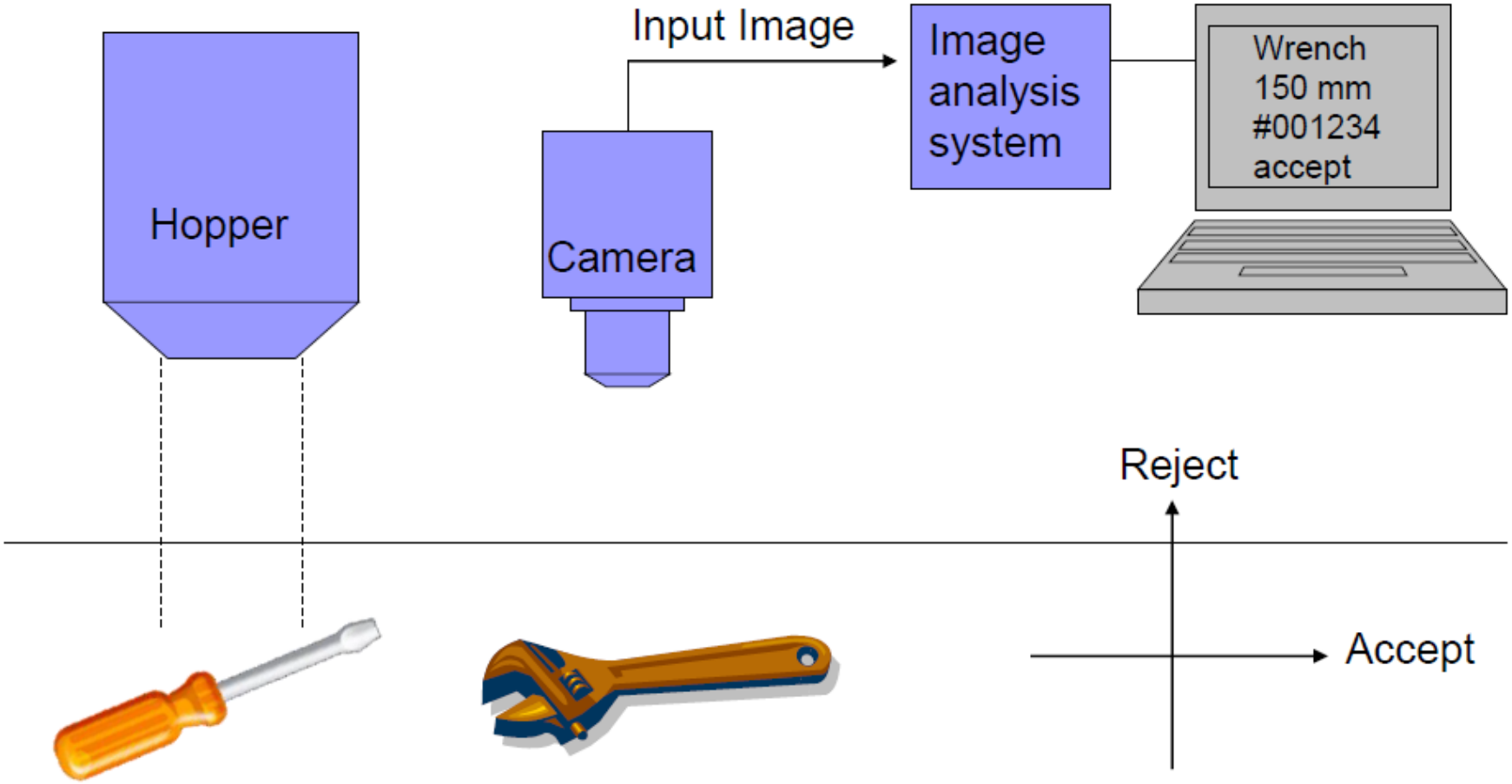
Human Vision, Computer Vision and Machine Vision

- Human Vision: allows humans to perceive and understand the world around him
- Computer Vision: aims to duplicate the effect of human vision by electronically perceiving and understanding an image
- Machine Vision: is the application of computer vision to industries including media and healthcare, and manufacturing.
  - A subfield of engineering that encompasses mechanical engineering, optics, computer science and automation.

Examples of Machine Vision

- Industrial vision – inspection of manufactured goods such as semiconductor chips, automobile components, tools, food and pharmaceuticals
  - Uses digital cameras, smart cameras and image processing software
  - Smart camera is an integrated machine vision system that comprises of image capture circuitry and a microprocessor to extract and process information from images. It normally has an interface device to make results available to other devices.

Examples of Machine Vision

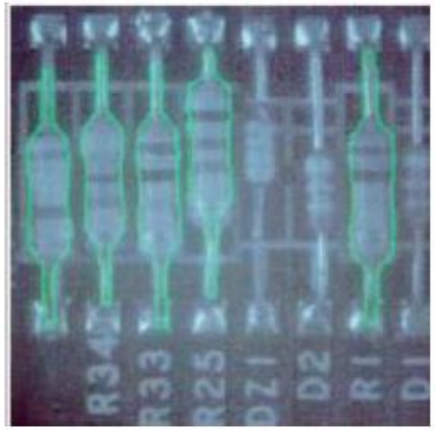


Examples of Machine Vision

- X-ray for health screening
- Satellite maps for battle planning
- Photoshop to prepare photographs
- Face detection
- Capture an image from moon, and then transmit the image to Earth

Machine Vision for Quality Control

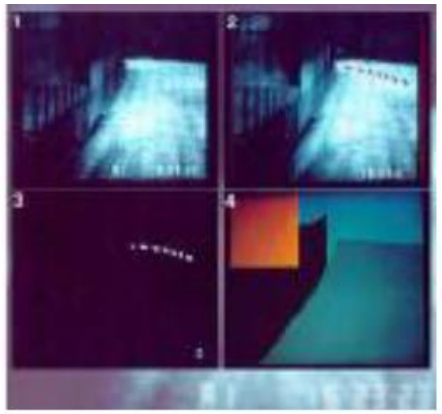
- Machine Vision allows manufacturing industry to
  - Detect defects
  - Calibrate and control the manufacturing process
  - Optimize the use of resources
- Leading to
  - Result repeatability
  - Product reliability
  - 100% high speed inspection
  - Consumer confidence and satisfaction



Source: British Machine Vision Association and Society for Pattern Recognition

Machine Vision for Security and Surveillance

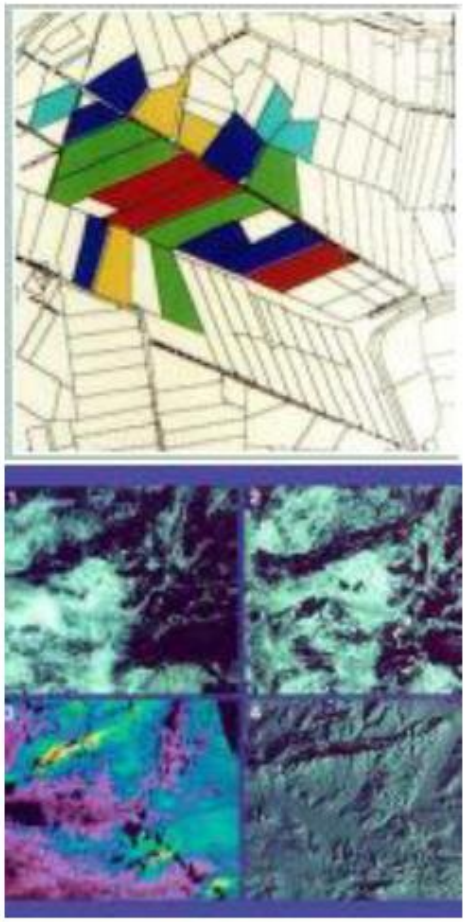
- Machine vision provides the abilities to
  - Track objects and people in 3D
  - Recognize and register specific and generic objects
  - Model and identify gestures, actions and behavior
  - Perform biometric measurements
- Leading to
  - Safer environment
  - Efficient non-obstructive monitoring
  - Reduce crime rates



Source: British Machine Vision Association and Society for Pattern Recognition

Machine Vision for Environment

- Machine vision can be used to
  - Monitor pollution from refuse sites
  - Map and monitor the condition of gas pipelines and railways
  - Police the pollution of the seas
  - Monitor the spread of disease in crops
- Leading to
  - Cleaner environment
  - Greater safety
  - Better planning and use of resources

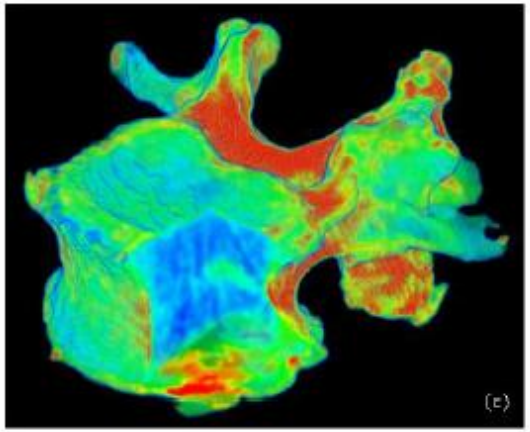


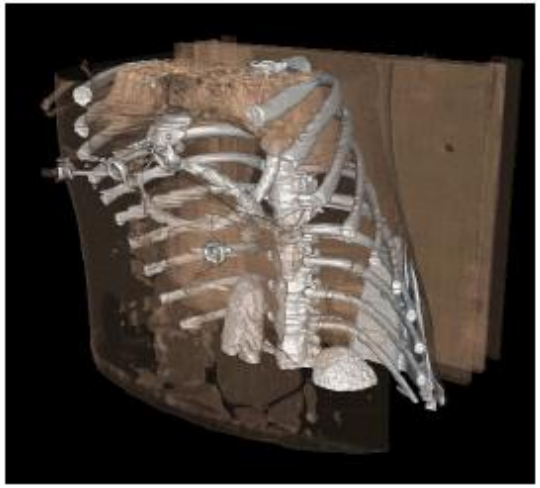
Machine Vision for Entertainment

- Machine vision allows
  - Searching of image databases and video libraries by content
  - Efficient image compression
  - Multiview scenes creation
  - Realistic models of objects generation
- Leading to
  - Greater realism
  - Lower cost
  - Wider accessibility

Machine Vision for Medicine

- Machine vision empowers the clinician with
  - 3D/4D visualization
  - 3D texture analysis
  - 3D and 2D image registration
  - Virtual object manipulation
- Leading to
  - Objectivity in measurement and estimation
  - Result repeatability
  - Decision consistency





Why is Computer and Machine Vision Difficult

- Loss of information in 3D to 2D
  - Pinhole model and single available view - the projective transformation sees a small object close to the camera in the same way as a big object remote from the camera.

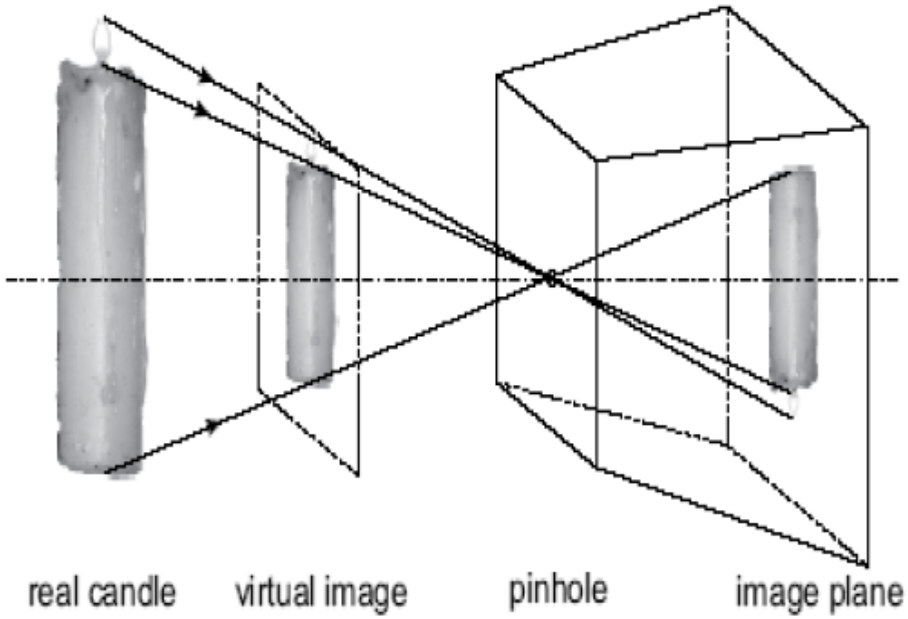


Figure 1.4: The pinhole model of imaging geometry does not distinguish size of objects.

Why is Computer and Machine Vision Difficult

- Interpretation
  - A human uses previous knowledge and experience to understand an image currently under observation.
- Noise
  - Inherently present in every measurement in the real world.

Why is Computer and Machine Vision Difficult

- Too much data
  - An A4 sheet of paper scanned at 300 dots per inch (dpi) at 8 bits per pixel (grayscale) = 8.5 MB
- Brightness measured
  - The brightness or radiance is dependent on the irradiance (light source type, position and intensity), the observer's position, the surface local geometry, and the surface reflectance properties.

Digital Image Fundamentals

Image Fundamentals

- **Image function** ***f(x, y)***

  f(x,y) must be digitized both spatially and in amplitude.

  Image Sampling:Digitization of the spatial coordinates ***(x, y)***.

  Gray-level quantisation: Digitization of amplitude.

  A continuous image ***f(x, y)***is approximated by equally spaced samples in the form of an *N* x *M*array as shown in eq. (1), where each element is a discrete quantity:

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots\cdots & f(0,M-1) \\ f(1,0) & f(1,1) & \cdots\cdots & f(1,M-1) \\ \vdots & & & \\ f(N-1,0) & f(N-1,1) & \cdots\cdots & f(N-1,M-1) \end{bmatrix} \qquad (1)$$

- The array represented by ***f(x,y)*** is commonly called a *digital image.*

Image Fundamentals

f(x, y) **,** ***x*** and ***y*** are integers.

The value of ***f(x, y)*** is the gray level of the image at the coordinates ***x*** and ***y***.
The coordinates are expressed in *units of pixels*.

N x *M* is the size of the image (eg. 512 by 512 = 262,144 pixels)

Gray levels is given by 2^G , where G is the number of bits in a byte which is used to represent the brightness of a pixel.

For example, for an 8-bit byte, the number of gray level available is $2^8 = 256$, or gray levels run from 0 to 255.

Image Fundamentals

2. Image Coordinates

The image will be described by an N x M matrix of pixel values (the element $p(i,j)$ are positive scalars), that indicate the light intensity of the flux on the picture element at (x, y) represented by the pixel (Fig. 1).

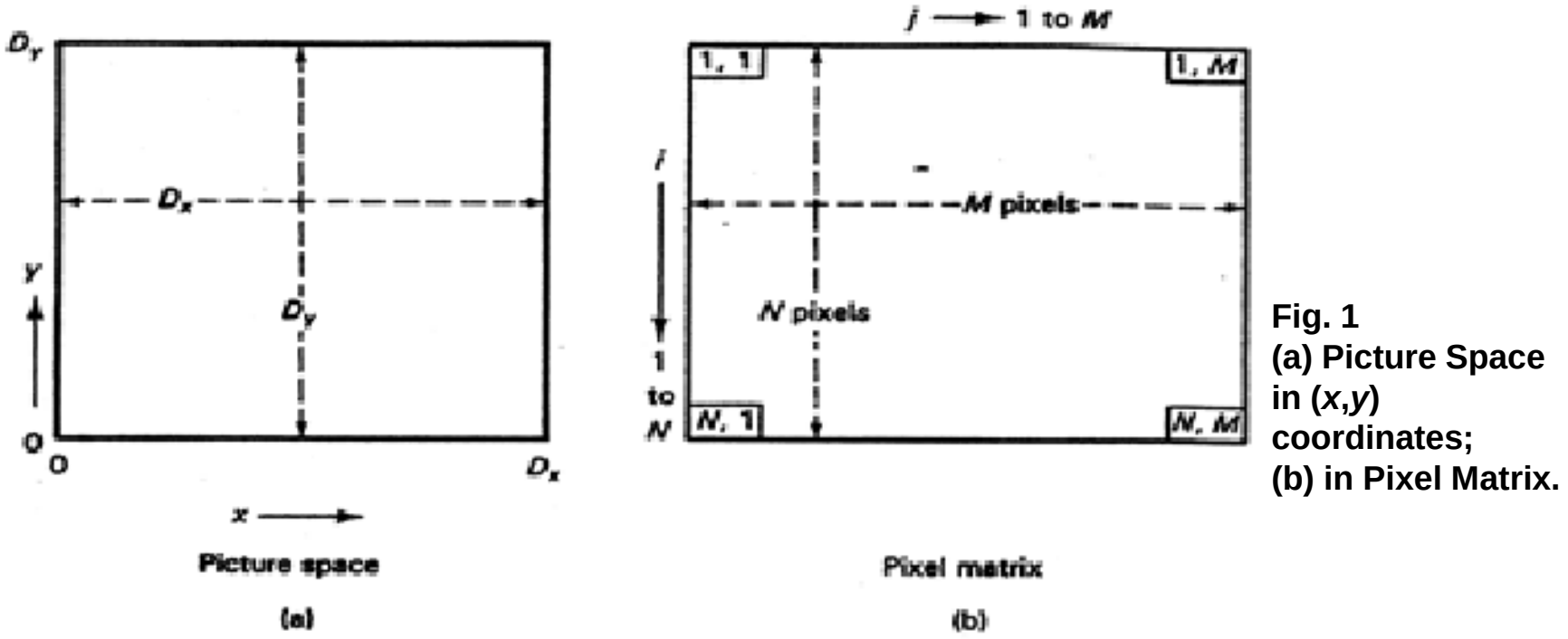


Fig. 1
(a) Picture Space in (x,y) coordinates;
(b) in Pixel Matrix.

Image Fundamentals

2. Image Coordinates

The origin in the picture and the matrix can be different; the *x* and *y* coordinates in the picture at the lower left corner, whereas the numbering of pixels starts at upper left corner of the matrix.

Note that this is not a standard convention, some people may use a different convention; e.g., first location is given as (0,0) instead of (1,1) (see Fig. 2).

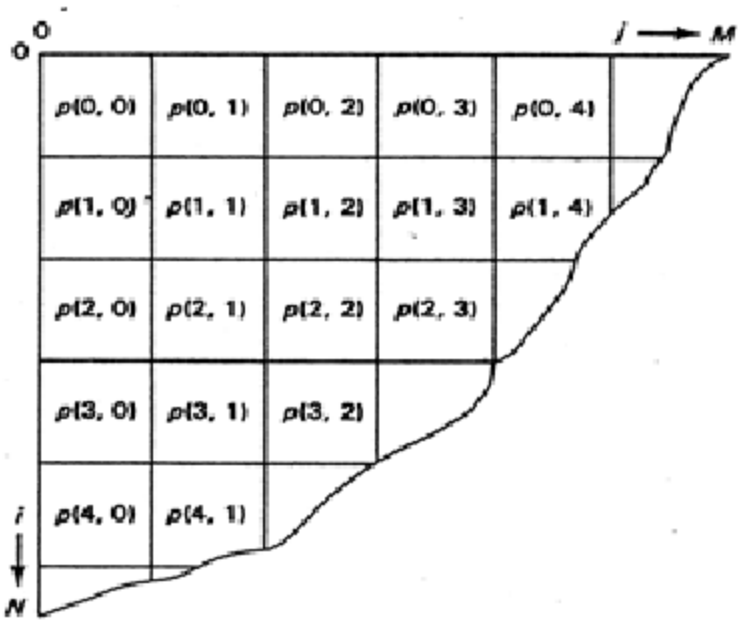


Fig. 2. Pixel Indexing in an image matrix

Image Fundamentals

2. Image Coordinates

Pixel Location

A pixel at location (n,m) has a numerical value which is the illumination (represented by gray level value on it.

In Fig.3, the image matrix with 16 gray levels.

Pixel at (0,0) = 0 (no light)
Pixel at (4,3) = 15 (full illumination)

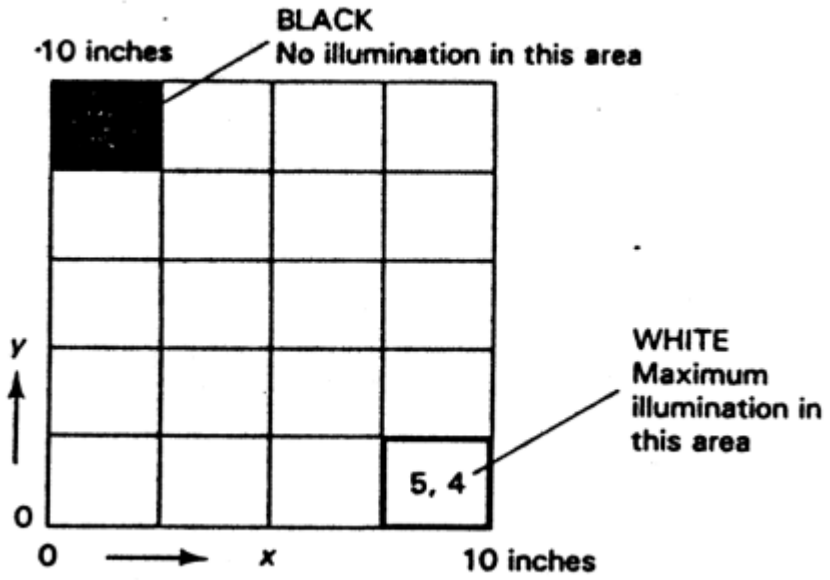


Fig. 3. Pixel location in an image matrix

Image Fundamentals

2. Image Coordinates

Window

If a certain sub-region of an image is of interest to us, it will be designated buy the four corners and the corresponding pixel values. (Fig. 4)

The sub-region is commonly known as *Window* or *Region of Interest (ROI)*.

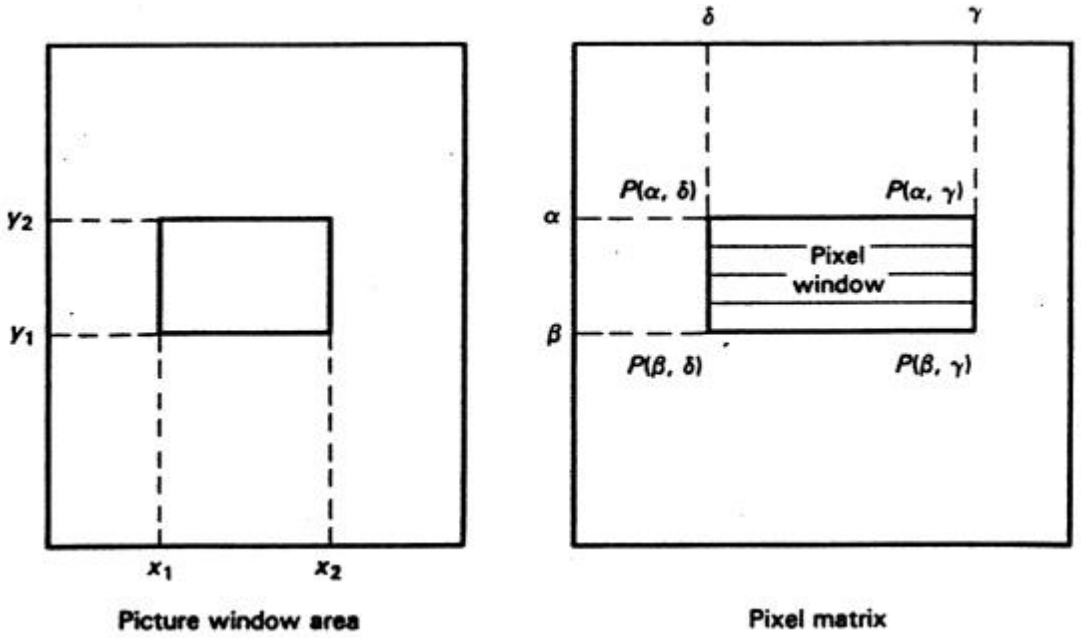


Fig. 4. Windows and the corresponding pixel values.

Image Fundamentals

3. What is a good image

Effect of spatial resolution

Fig. 5(a) shows a 1024x1024, 256 level digital image of a rose. Fig. 5(a) to 5(f) shows the results of reducing the spatial resolution from ***N*** = 1024, 512, 256, 128, 64, and 32, respectively. The gray level allowed is 256 for all cases.

With the display area remains the same at 1024 x 1024, pixels in the lowest resolution images were duplicated in order to fill the entire display. This pixel *replication* produces a checker-board effect, which is especially visible in the images of lowest resolution (Fig. 5(e) & 5(f)).

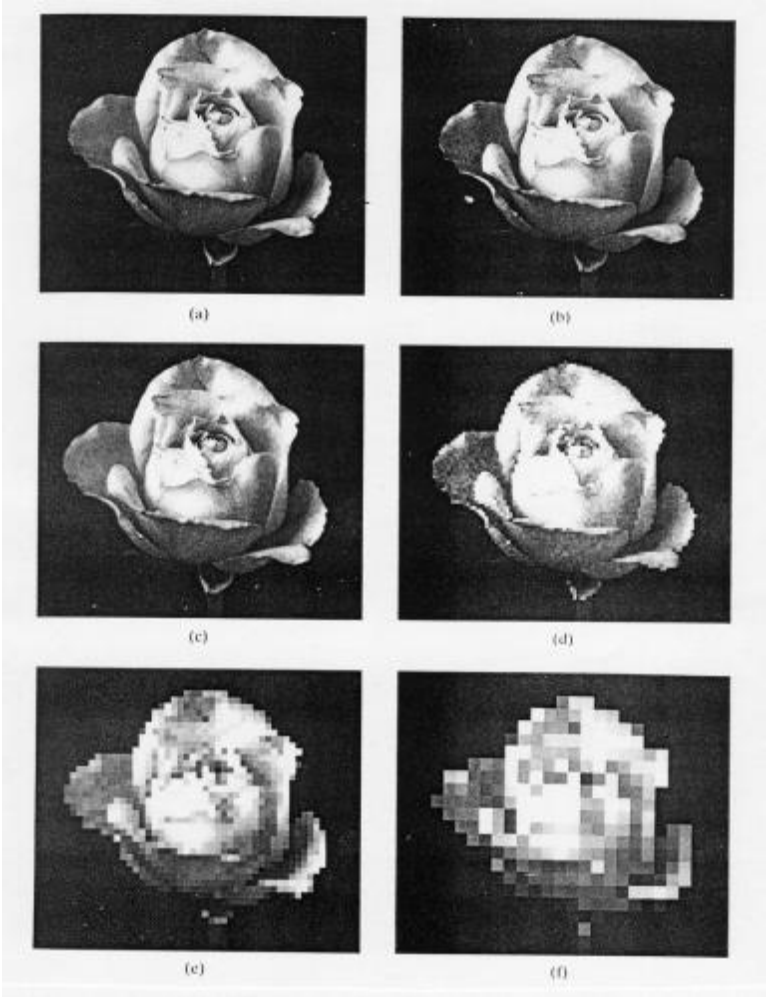


Fig. 5 Effects of reduced spatial resolution

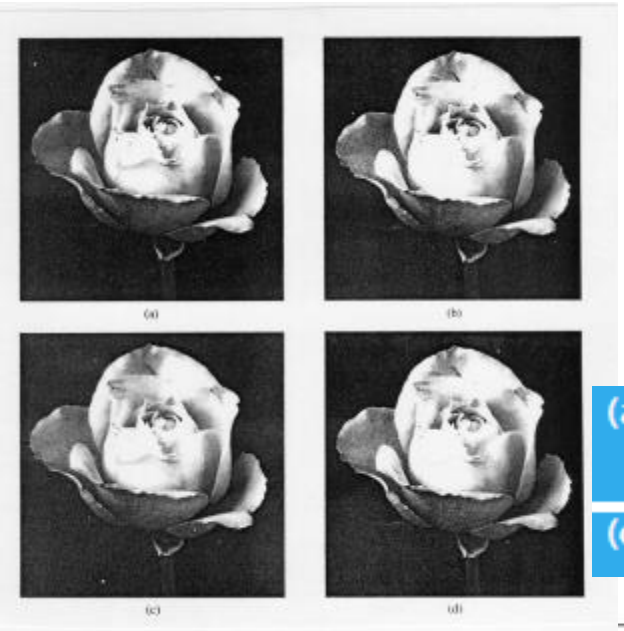
Image Fundamentals

3. What is a good image

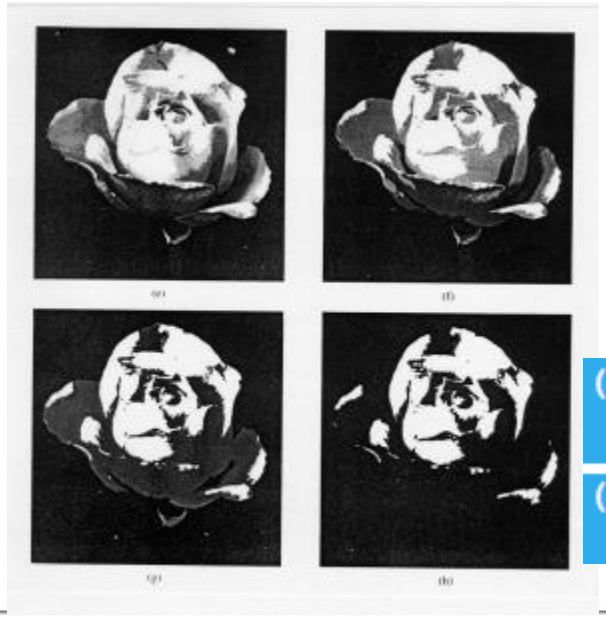
Effect of number of gray levels

Fig. 6(a) shows the 1024x1024, 8-bit image. Fig. 6(b) to 6(h) were obtained by reducing the number of bits from m = 7 to 1, with the samespatial resolution constant at 1024x1024.

The 256, 128 and 64-gray level images are visually identical for all practical purposes. Fig 6(h) with m = 1 is basically a binary image.



(a) (b)
(c) (d)



(e) (f)
(g) (h)

Fig. 6 A 1024x1024 image displayed in 256, 128, 64, 32, 16, 8, 4 and 2 gray levels respectively (From Fig. 6(a) to 6(h).

Image Fundamentals

4. Gray Levels

To provide intervening values of illumination, it is necessary to increase the number of bits representing the pixel value.

Bit number	Gray Scale	Gray level value
1	2^1 = 2 values	0 to 1
3	2^3 = 8 values	0 to 7
4	2^4 = 16 values	0 to 15
8	2^8 = 256 values	0 to 255

The lowest value is assigned for black, and the maximum value is assigned for maximum illumination. The values assigned to pixel are always integers

Basic Image Processing

Basic Image Processing

The processing of data in vision system can be categorised into:

1. Point by point (*monadic*) alteration of data on a global scale, and

2. Multiple point (*dyadic*) determination of elements of a new array of the image.

The generation of the new pixel image matrix will be a function of either individual pixel location values or values of pixels in the neighborhood of the individual cell, as shown in Fig. 7.

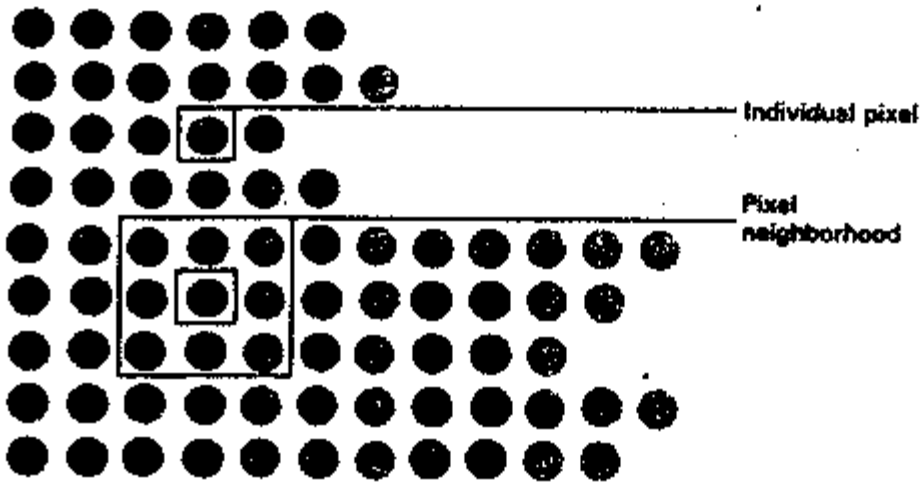


Fig. 7. Point and neighborhood functions

Basic Image Processing

1. *Monadic Operations*

Monadic operations involve the generation of a new array by modifying the pixel value at a single location based on a global rule applied to every location in the original array. The process involves

1. obtaining the pixel value of a given location in the array,
2. modifying it by a simple linear or non-linear operation, and,
3. placing the new pixel value in the specific corresponding location of a new array.

The process is repeated on the pixel value of the next location and continued over the entire array.

$$g(i, j) = f[p(i, j)] \qquad (2)$$

Basic Image Processing

a. Identity Operator

This operation results in the creation of an output image which is identical to the input image (Fig. 8). The input image is *p* and the output image is *q*. The function *f* is a straight line starting at the origin and extending to the maximum pixel value of the image system.

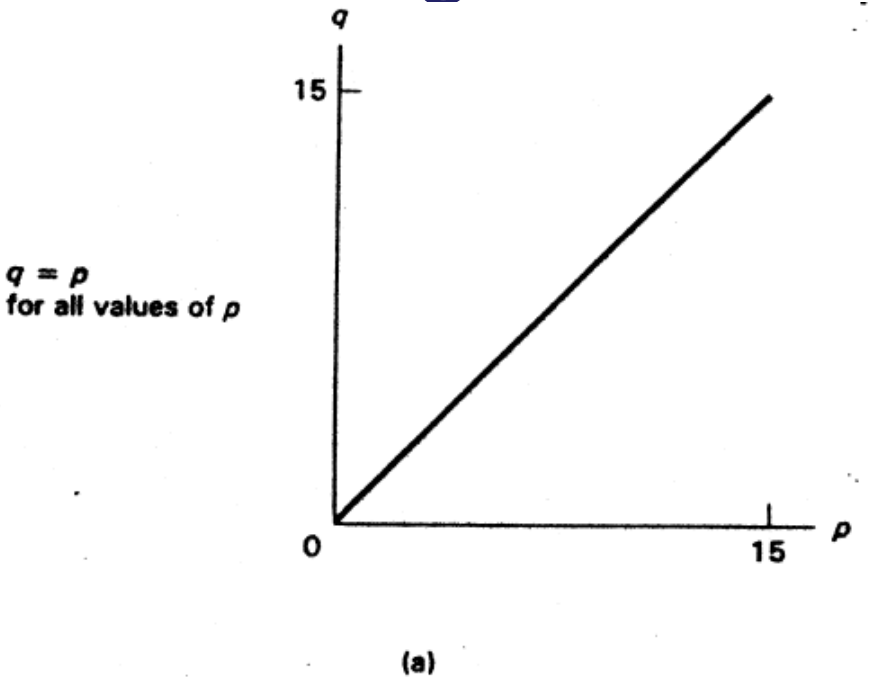
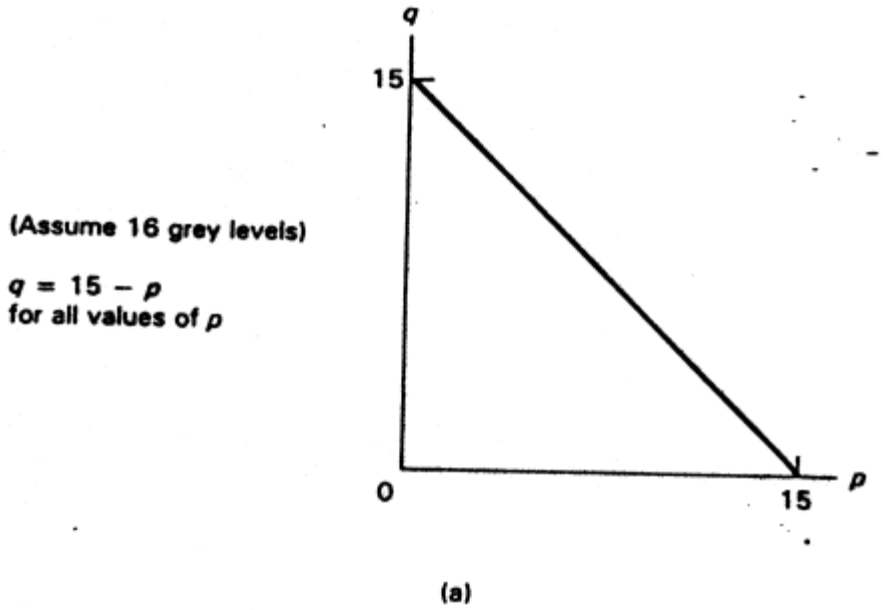


Fig. 8 Identity operator: (a) function, (b) input and (c) output.

Basic Image Processing

b. *Inverse Operator*

The operator results in the creation of an output image which is the inverse of the input image (Fig. 9). The function *f* is a straight line having a value of maximum gray level input value and equal to zero at the maximum gray input value.



$$\begin{bmatrix} 15 & 15 & 0 & 0 & 2 \\ 13 & 13 & 15 & 0 & 0 \\ 0 & 0 & 7 & 14 & 14 \\ 0 & 1 & 2 & 3 & 4 \\ 15 & 14 & 13 & 12 & 11 \end{bmatrix}$$

p matrix

(b)

$$\begin{bmatrix} 0 & 0 & 15 & 15 & 13 \\ 2 & 2 & 0 & 15 & 15 \\ 15 & 15 & 8 & 1 & 1 \\ 15 & 14 & 13 & 12 & 11 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

q matrix

(c)

Fig. 9 Inverse operator: (a) function, (b) input and (c) output

Basic Image Processing

c. *Threshold Operator*

This class operator results in a binary output image from a gray scale input image where the level of transition is given by the input parameter *p1* as shown in Fig. 10, and is known as the threshold. All pixels below *p1* are converted to zero and pixels equal to or greater than *p1* are converted to an 1.

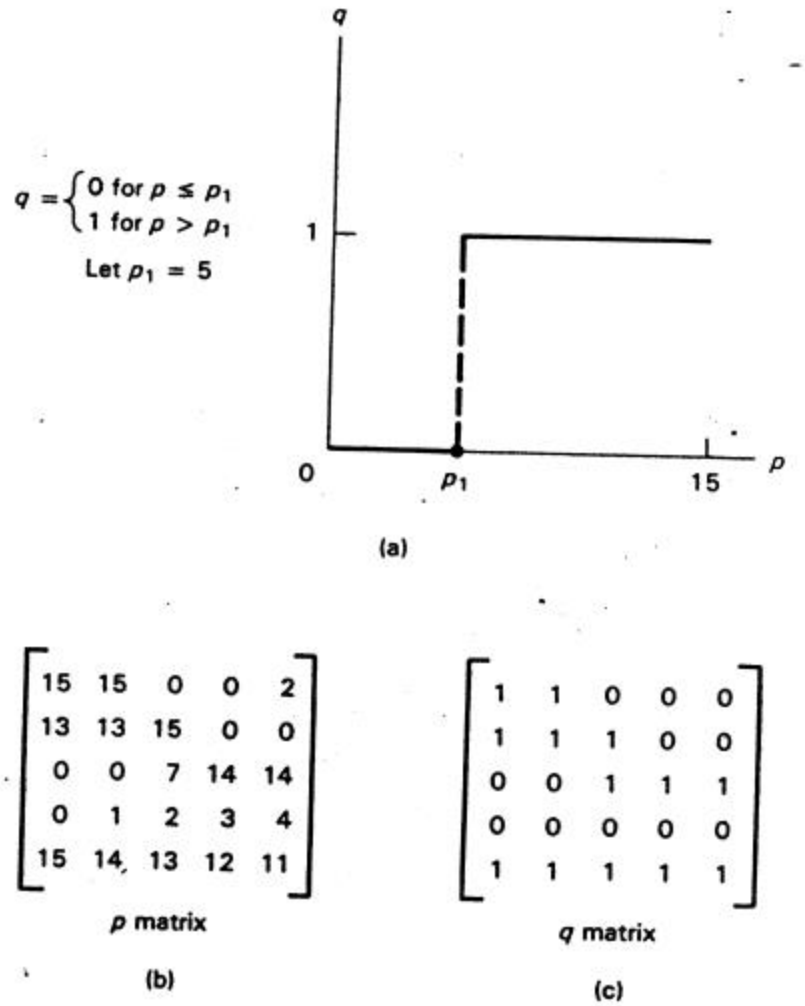
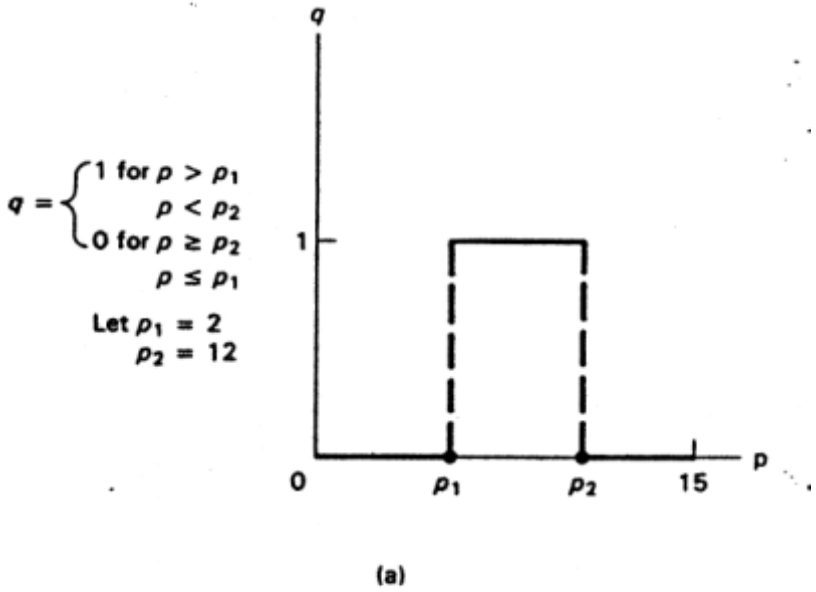


Fig. 10 Threshold operator: (a) function, (b) input and (c) output

Basic Image Processing

d. Binary Threshold interval Operator

This operator can be used to convert a multi-level gray scale to a binary image, or it can be applied to a binary image for inversion purpose. The binary image output, or image produced by the binary threshold interval operator, can be inverted by the inverted binary threshold function as shown in Fig. 11.



(a)

$$\begin{bmatrix} 15 & 15 & 0 & 0 & 2 \\ 13 & 13 & 15 & 0 & 0 \\ 0 & 0 & 7 & 14 & 14 \\ 0 & 1 & 2 & 3 & 4 \\ 15 & 14 & 13 & 12 & 11 \end{bmatrix}$$

p matrix

(b)

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

q matrix

(c)

Fig. 11 Binary Threshold Inverted operator: (a) function, (b) input and (c) output

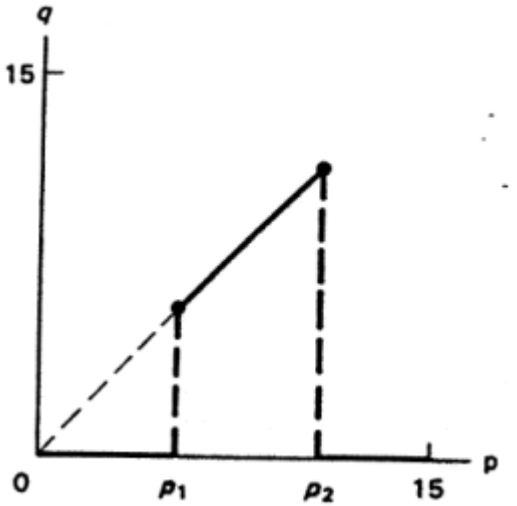
Basic Image Processing

e. *Gray Scale Threshold Operator*

The class of operator shown in Fig. 12 results in a gray scale output image value between *p1* and *p2* and makes all input values outside *p1* to *p2* zero. This operator can be used to identify image features having a specific value for pseudo-colour application processing techniques.

$$q = \begin{cases} p \text{ for } p_1 \le p \le p_2 \\ 0 \text{ for } p < p_1 \\ \quad p > p_2 \end{cases}$$

Let $p_1 = 2$
$p_2 = 12$



(a)

$$\begin{bmatrix} 15 & 15 & 0 & 0 & 2 \\ 13 & 13 & 15 & 0 & 0 \\ 0 & 0 & 7 & 14 & 14 \\ 0 & 1 & 2 & 3 & 4 \\ 15 & 14 & 13 & 12 & 11 \end{bmatrix}$$

p matrix

(b)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 2 & 3 & 4 \\ 0 & 0 & 0 & 12 & 11 \end{bmatrix}$$

q matrix

(c)

Fig. 12 Gray scale threshold operator: (a) function, (b) input and (c) output

Basic Image Processing

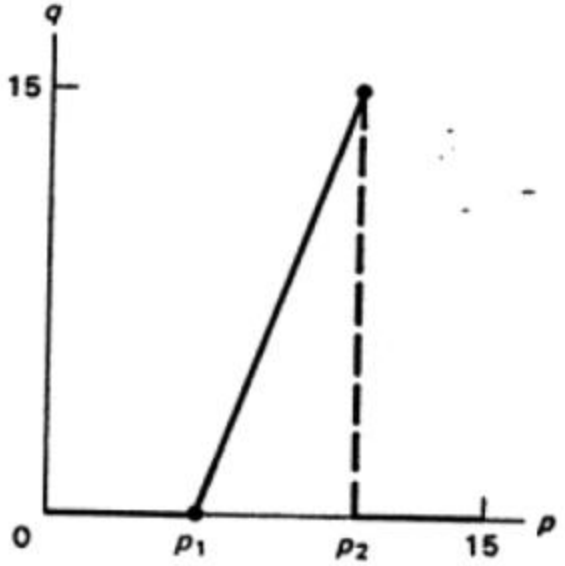
f. Stretch Operator

This class of operators results in a full gray scale output image corresponding to the input interval *p1* and *p2* and suppresses all values outside this range (Fig. 13).

$$q = \begin{cases} (p - p_1)\,\dfrac{15}{(p_2 - p_1)} & \text{for } p_1 < p < p_2 \\ 0 \text{ for } p \le p_1 \\ \quad p \ge p_2 \end{cases}$$

Let $p_1 = 3$

$p_2 = 7$



(a)

$$\begin{bmatrix} 15 & 15 & 0 & 0 & 2 \\ 13 & 13 & 15 & 0 & 0 \\ 0 & 0 & 7 & 14 & 14 \\ 0 & 1 & 2 & 3 & 4 \\ 15 & 14 & 13 & 12 & 11 \end{bmatrix}$$

p matrix

(b)

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

q matrix

(c)

Fig. 13 Stretch operator: (a) function, (b) input and (c) output

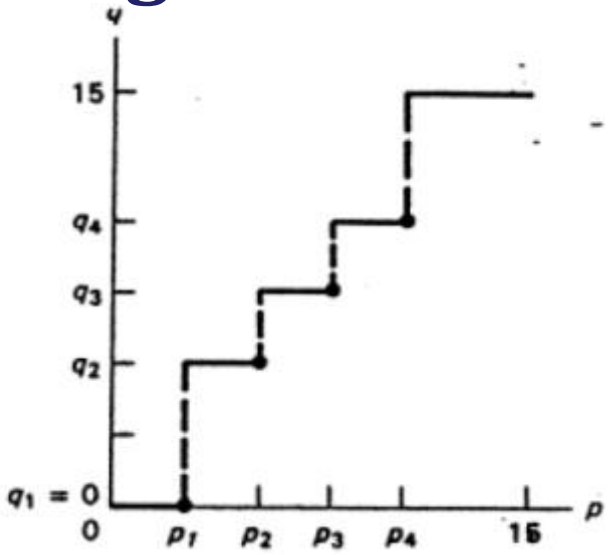
Basic Image Processing

g Gray Level Reduction Operator

This class of operators results in an output image which has a smaller number of gray levels than the number of gray levels of the input image, as shown in Fig. 14, where an input image with 15 levels is converted to an image having five levels: *q1*, *q2*, *q3* , *q4* and 15.

$= 0$ for $p \le p_1$
q_2 for $p_1 \le p \le p_2$
q_3 for $p_2 < p \le p_3$
q_4 for $p_3 < p \le p_4$
15 for $p_4 < p \le 15$

Let $p_1 = 2$ $q_2 = 3$
$p_2 = 4$
$p_3 = 6$ $q_3 = 6$
$p_4 = 8$ $q_4 = 9$



(a)

$$\begin{bmatrix} 15 & 15 & 0 & 0 & 2 \\ 13 & 13 & 15 & 0 & 0 \\ 0 & 0 & 7 & 14 & 14 \\ 0 & 1 & 2 & 3 & 4 \\ 15 & 14 & 13 & 12 & 11 \end{bmatrix}$$

p matrix

(b)

$$\begin{bmatrix} 15 & 15 & 0 & 0 & 0 \\ 15 & 15 & 15 & 0 & 0 \\ 0 & 0 & 9 & 15 & 15 \\ 0 & 0 & 0 & 3 & 3 \\ 15 & 15 & 15 & 15 & 15 \end{bmatrix}$$

q matrix

(c)

Fig. 14 Gray level reduction operator: (a) function, (b) input and (c) output

Basic Image Processing

2. *Dyadic Two Point Transformations*

The dyadic point-by-point operator uses the information contained at the same location in two images. In Fig. 15, pixels from A and B are used to create a new image C. The size of the image does not change and the dyadic transformation function can be either linear or nonlinear.

$$c_{ij} = f_D(a_{ij}, b_{ij}) \qquad (3)$$

Transformation involves two variables associated with the pairs of corresponding pixels,

$$R(i,j) = f[p(i,j), q(i,j)] \qquad (4)$$

where p and q are input matrices, f is the functional operator, and R is the resultant output matrix.

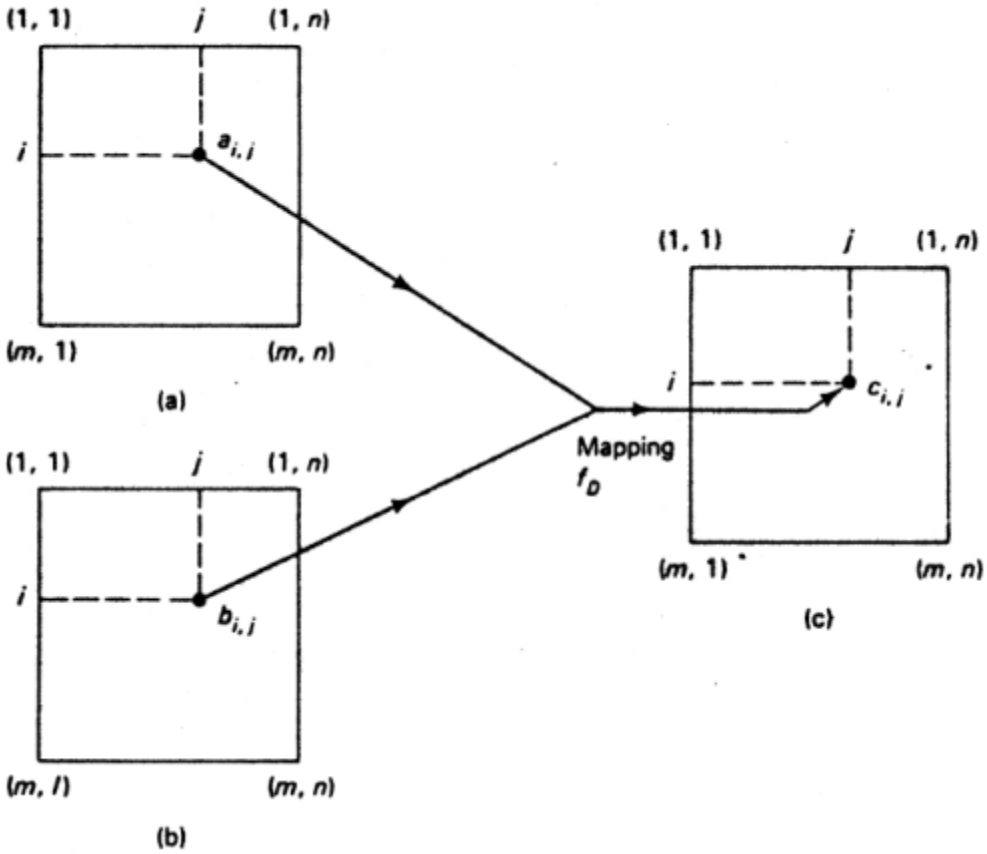


Fig. 15 Dyadic multi point-to-point operator: (a) input image A, and (b) input image B are mapped into a new image C (c).

Basic Image Processing

a. Image Addition

Image addition can be used to reduce the effect of noise in the data, as shown in Fig. 16. The value of the output c_{ij} is given by

$$c_{ij} = \frac{(a_{ij} + b_{ij})}{k} \qquad (5)$$

over the range of values where k equals to the number of samples.

Special rules, such as rounding up, must be devised and applied to the transformation function to produce meaningful results.

$$\begin{bmatrix} 0 & 12 & 142 & 255 \\ 1 & 6 & 40 & 254 \\ 24 & 0 & 20 & 255 \\ 30 & 2 & 10 & 240 \end{bmatrix} \quad \begin{bmatrix} 14 & 11 & 9 & 253 \\ 3 & 5 & 39 & 254 \\ 11 & 1 & 19 & 255 \\ 18 & 2 & 11 & 256 \end{bmatrix} \quad \begin{bmatrix} 7 & 12 & 76 & 254 \\ 2 & 6 & 40 & 254 \\ 18 & 1 & 20 & 255 \\ 23 & 2 & 11 & 248 \end{bmatrix}$$

a_{ij} input 1 . b_{ij} input 2 . c_{ij} output .

Fig. 16 Image addition for i=j=4 and k=2

Basic Image Processing

b. Image Subtraction

Image subtraction can be used to detect changes that have occurred during the time interval between two images are taken if the two images are of the same scene.
In Fig. 17, the relationship for (a) and (b) is given by:

$$c_{ij} = k(a_{ij} - b_{ij}) \qquad (6)$$

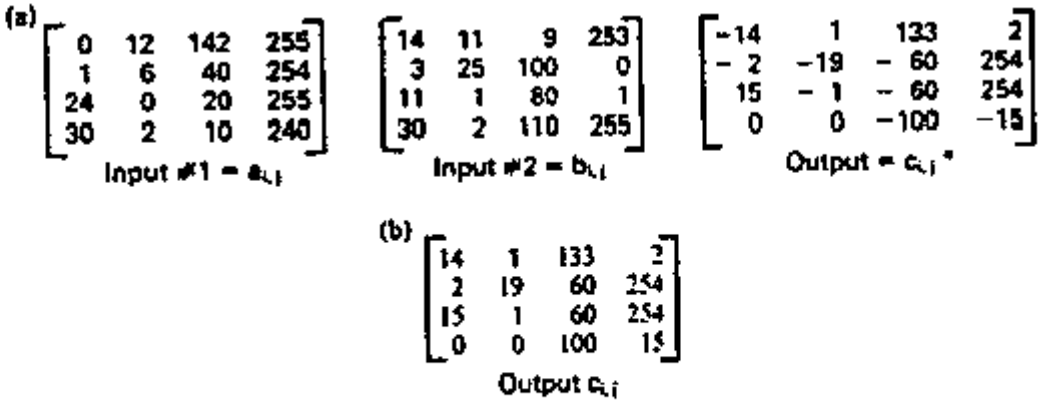


Fig. 17 Image subtraction

Basic Image Processing

b. *Image Subtraction*

$$c_{ij} = k(a_{ij} - b_{ij}) \quad (6)$$

Since image processing uses positive number, it is necessary to define the output in some manner that makes all the values positive. This could involve a rescaling where the largest negative number is set equal to zero and the largest number is set to the maximum gray scale value (255, in part (a) of Fig. 17). Another approach is to define the difference as an unsigned difference or the absolute value of the difference as shown in matrix in Fig. 17 (b).

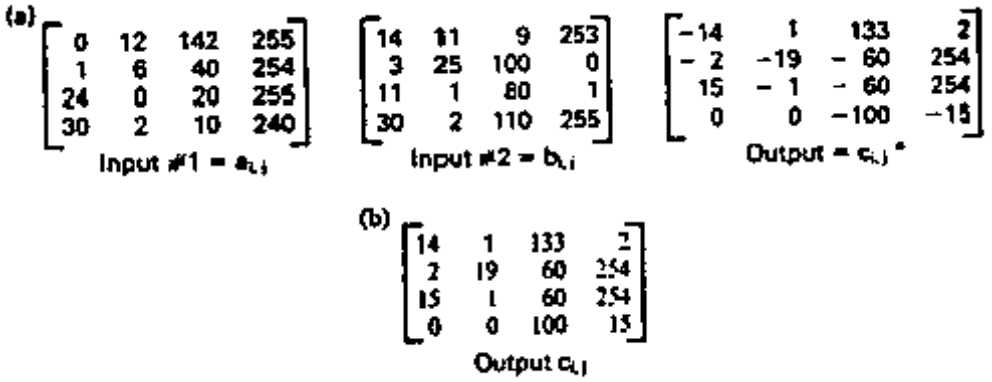


Fig. 17 Image subtraction

Basic Image Processing

b. *Image Subtraction*

There are many different ways that the basic dyadic two point transformations can be used to enhance the data in an image matrix. We must decide on the information required for the application and the data processing techniques that will enhance the information features in the image data.

The monadic operator ***f*** will be applied to c_{ij} matrix in Fig. 15 (b). R_{ij} values are rounded up. The equation

$$R_{ij} = (c_{ij} + 100) \times \frac{255}{354} \quad (7)$$

is used to convert the values. R_{ij} will be 0 when c_{ij} is -100 (minimum value) and 255 when c_{ij} is 254 (maximum value).

(a)

$$\begin{bmatrix} 0 & 12 & 142 & 255 \\ 1 & 6 & 40 & 254 \\ 24 & 0 & 20 & 255 \\ 30 & 2 & 10 & 240 \end{bmatrix}$$

Input #1 = a_{ij}

$$\begin{bmatrix} 14 & 11 & 9 & 253 \\ 3 & 25 & 100 & 0 \\ 11 & 1 & 80 & 1 \\ 30 & 2 & 110 & 255 \end{bmatrix}$$

Input #2 = b_{ij}

$$\begin{bmatrix} -14 & 1 & 133 & 2 \\ -2 & -19 & -60 & 254 \\ 15 & -1 & -60 & 254 \\ 0 & 0 & -100 & -15 \end{bmatrix}$$

Output = c_{ij}

(b)

$$\begin{bmatrix} 14 & 1 & 133 & 2 \\ 2 & 19 & 60 & 254 \\ 15 & 1 & 60 & 254 \\ 0 & 0 & 100 & 15 \end{bmatrix}$$

Output c_{ij}

Basic Image Processing

b. *Image Subtraction*

$$R_{ij} = (c_{ij} + 100) \times \frac{255}{354}$$

R_{ij} will be 0 when c_{ij} is -100 (minimum value) and 255 when c_{ij} is 254 (maximum value). Equation (7) is shown

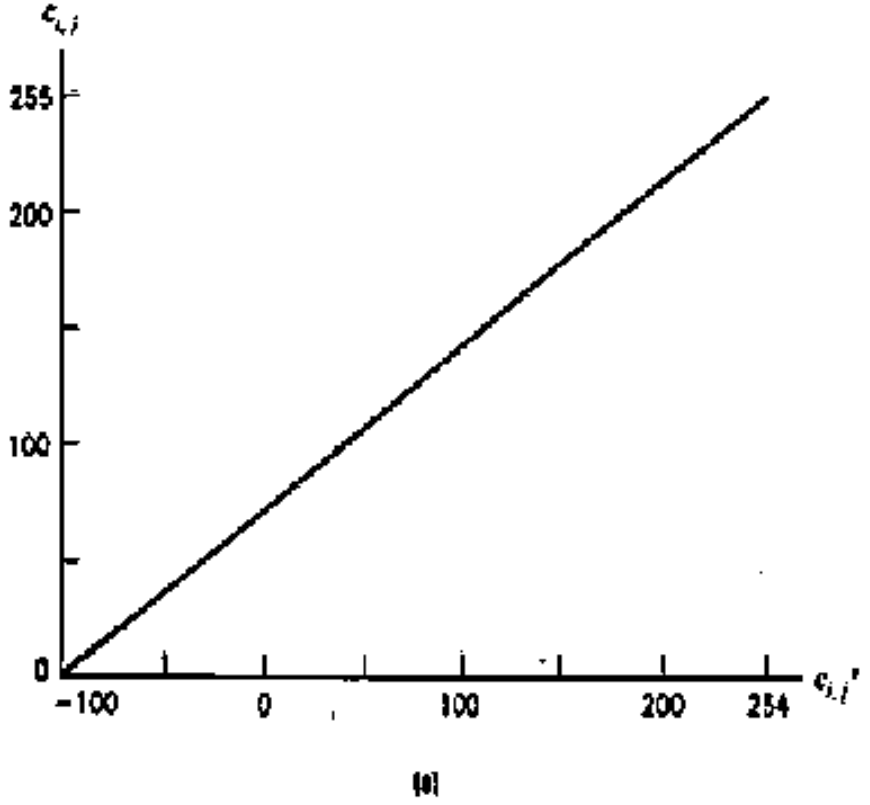


Fig. 16 Image subtraction with function shown in (a) which generates resultant output shown in (b) having all positive values

Basic Image Processing

c. *Convolution: Spatial Transformation*

A new image can be generated where the pixel value assigned to each location is a function of the pixel values of the adjacent locations as shown by the 3x3 convolution in Fig. 19.

In a convolution, the pixel value of the centre location is computed based on the values of the nine locations. The location is then shifted by 1 and the process is repeated until the entire image matrix is generated.

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \times \begin{bmatrix} + & + & - \\ - & + & + \\ + & + & - \end{bmatrix} = \begin{bmatrix} & & \\ & e' & \\ & & \end{bmatrix}$$

Where:

$$e' = +a + b - c - d + e + f + g + h - i$$

Fig. 19 Convolution Mask

Basic Image Processing

c. *Convolution: Spatial Transformation – an example*

Given a partial image shown in Fig. 20(a). It is to be operated on (or filtered) by the masks (or filter) given by Fig. 20(b(i) and (ii)) separately on the central pixel '10'..

$$\begin{bmatrix} x & x & x & x & x \\ x & 254 & 251 & 255 & x \\ x & 250 & 10 & 249 & x \\ x & 255 & 252 & 248 & x \\ x & x & x & x & x \end{bmatrix}$$

Fig 20(a)

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 & 2 \\ 0 & 0 & 0 \\ -2 & -1 & -2 \end{bmatrix}$$

Fig 20(b(i), b(ii))

The central pixel =
(254+251+255+250+10+249+255+252
+248)
= 224.89 = 225

$$\begin{bmatrix} x & x & x & x & x \\ x & 254 & 251 & 255 & x \\ x & 250 & 225 & 249 & x \\ x & 255 & 252 & 248 & x \\ x & x & x & x & x \end{bmatrix}$$

$$\begin{bmatrix} x & x & x & x & x \\ x & 254 & 251 & 255 & x \\ x & 250 & 11 & 249 & x \\ x & 255 & 252 & 248 & x \\ x & x & x & x & x \end{bmatrix}$$

The central pixel =
(2×254+1×251+2×255+0×250+0×10+0×249
+(-2)×255+(-1) ×252+(-2) ×248)
= 11

Basic Image Processing

3. *Neighbours and Connectivity*

Neighbours

A pixel ***p*** at coordinates ***(x, y)*** has four *horizontal* and *vertical* neighbours:

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

This set of pixels, called the 4-***neighbours*** of ***p***, is denoted by $N_4(p)$. Each of these pixels is a unit distance from ***(x, y)*** and some of these neighbours of ***p*** will be outside the digital image if ***(x, y)*** is on the border of the image.

The four *diagonal* neighbours of ***p*** have coordinates:

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

and will be denoted by $N_D(p)$.

The pixels defined above by $N_4(p)$ and $N_D(p)$, are called the *8-neighbours* of ***p***, denoted by $N_8(p)$.

Basic Image Processing

3. *Neighbours and Connectivity*

Connectivity

Connectivity between pixels is an important concept used in establishing boundaries of objects and components of regions in an image.

To establish whether connectivity exists between two pixels, we must set certain criteria:

- whether they are adjacent in some sense (e.g., if they are 4-neighbours); and
- whether their gray levels satisfy a specified criterion of similarity (e.g., if they are equal).

Two pixels in an image might be N_4 neighbours, but they are connected only when they have the same gray level value.

Let V be the set of gray-level values to determine connectivity; for example, if only connectivity of pixels with intensities of 59, 60, and 61 are important, then $V = \{59, 60, 61\}$.

Basic Image Processing

3. *Neighbours and Connectivity*

Connectivity

If V is a set of neighbouring pixels which have some common characteristics, there are two main types of connectivity:

1. *4-connectivity* — Two pixels ***p*** and ***q*** with values from ***V*** are 4-connected if ***q*** is in the set $N_4(p)$.

2. *8-connectivity* — Two pixels ***p*** and ***q*** with values from ***V*** are 8-connected if ***q*** is in the set $N_8(p)$.

Binary Machine Vision

Introduction

In object recognition and detection by a computer vision system, the input image is simplified by generating an output whose pixels:

- Tend to have high values if they are part of an objects of interest; and
- Low values if they are not.

Introduction

To identify an object, the easiest way is to perform a ***Thresholding-Labelling*** operation to produce a binary image.

- Determine a threshold value;
- Those pixels having gray value higher than the Threshold value are given the value 1.
- Those pixels having gray value lower than the Threshold value are given the value 0.

The generation and analysis of such image are called

Binary Machine Vision

Introduction

The conversion of gray-level images to binary representation is important for a number of reasons:

- To identify the extent of objects, represented as a region in the image. For example, we may wish to separate a visual target from its background.

- To concentrate on shape (or morphological) analysis, in which case the intensities of pixels are less significant than the shape of a region.

- To convert an edge-enhanced image to a line drawing of the captured scene. It is necessary to distinguish strong edges that correspond to object outlines (and features) from weak edges due to illumination changes, shadows, etc.

Thresholding

- Thresholding is a labelling operation as described above.
- Main difficulty – what is a good threshold value?
  - It would be known if we know the distributions of the bright and dark pixels. – but this is usually not known.
  - The only clue can be obtained from image histogram – but not always possible.

Thresholding

							5	8	9			
								9	8	9		
		5							7	9	8	
		6								6	9	9
		5			7							
	2				6							
	8					5			1	1		
3						6			1	1	1	
						7						
	4	3						3	2	6	2	
	2	4						3	8	4	3	
	5	9						7	2	3	9	

Fig 1 Before Thresholding

							1	1	1			
								1	1	1		
		1							1	1	1	
		1								1	1	1
		1			1							
	0				1							
	1					1			0	0		
1						1			0	0	0	
						1						
	1	1						1	0	1	0	
	0	1						1	1	1	1	
	1	1						1	0	1	1	

Fig 2 After Thresholding

Image shown in Fig. 1 has been thresholded with a threshold value of 2 shown in Fig. 2.

Thresholding

Determine Threshold Value from Histogram

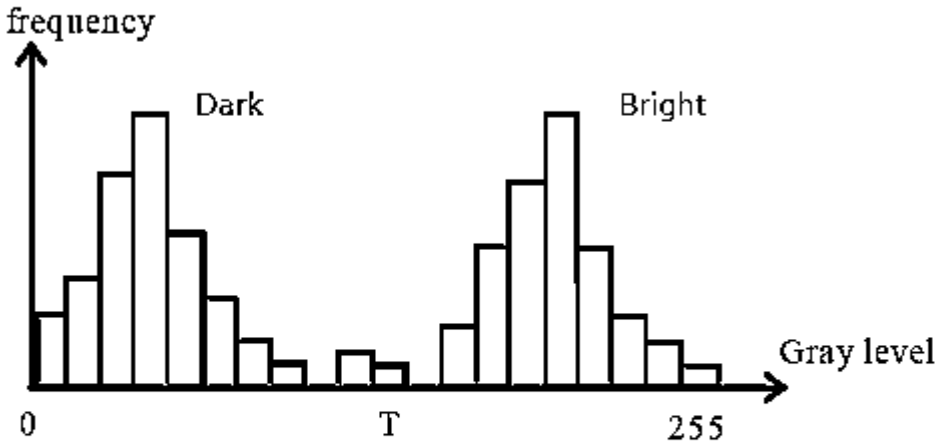


Fig 3. Gray level histogram with single threshold

- Bi-modal Histogram shown in Fig. 3 with little distribution overlaps between the dark and bright pixels. There are two dominant peaks. (If you are lucky!)

- The Threshold value will be at the valley between the two peaks, ***T.***
- Any point (x, y) for which $f(x,y) > T$ is called the object point; otherwise, it is called a background point.

Thresholding

Multiple Thresholds

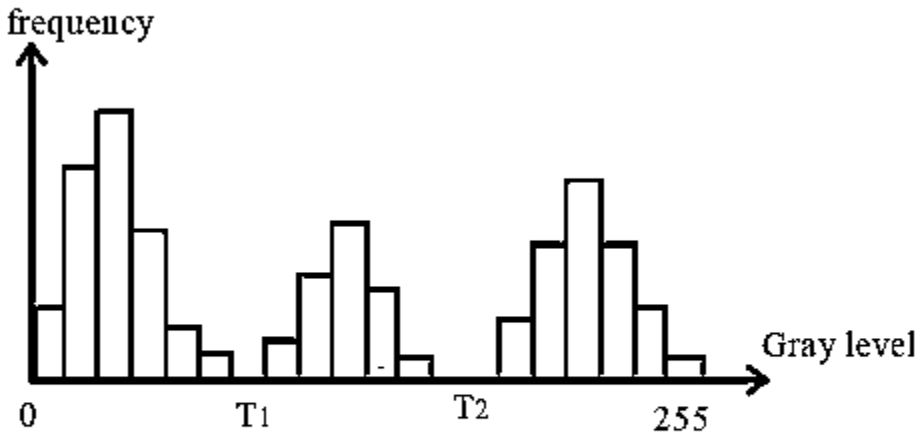
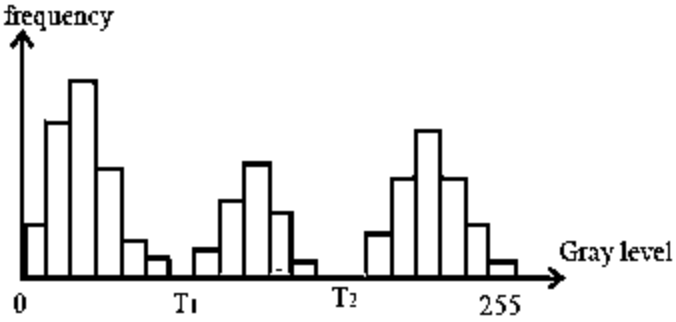


Fig 4. Gray level histogram with multiple thresholds

- Fig. 4 shows a slightly more general case of the approach. There are three dominant modes.
- There two Threshold values T_1 and T_2

Thresholding

Multiple Thresholds



The same classification approach as in the previous case classifies a point (x,y) as

belong to the object class if $T_1 < f(x,y) \le T_2$

belong to the other object class if $f(x,y) > T_2$

to the background is $f(x,y) \le T_1$

This type of multi-level thresholding is less reliable than its single-threshold counterpart.

It is difficult to establish the multiple threshold especially when the number of corresponding histogram modes is large.

Thresholding

Mathematical Definition of Thresholding

In formal mathematical terms, thresholding may be viewed as an operation that involves tests against a function ***T*** of the form:

$$T = T\left[x, y, p(x, y), f(x, y)\right] \qquad (1)$$

where ***f(x, y)*** is the gray level of point ***(x, y)*** and ***p(x, y)*** denotes some local property of the image (for example, the average gray level of a neighbourhood centered on ***(x, y)***.

An image after having gone through thresholding is defined as:

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) \le T \end{cases} \qquad (2)$$

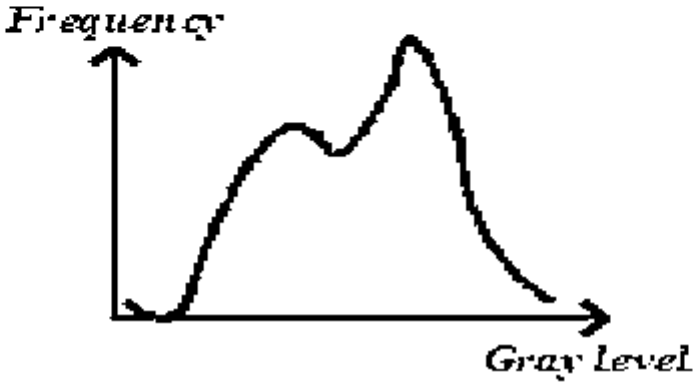
When ***T*** depends only on ***f(x, y)***, the threshold is called *global*.
If ***T*** depends on both ***f(x, y)*** and ***p(x, y)***, then the threshold is called *local*.
If ***T*** depends on the spatial coordinates ***x*** and ***y***, the threshold is called *dynamic*

Difficulties in Thresholding

The determination of a threshold value becomes difficult:

1. When the distributions for the bright and dark pixels become more and more overlapped, the valley between the two histogram modes becomes more difficult, because the valley begins to disappear as the two distributions begin to merge together.

2. Furthermore, when there is substantial overlap, the choice of threshold as the valley point is less optimal in the sense of minimising classification error.



Difficulties in Thresholding

Fig. 5 shows a BNC Connector in a dark texture background

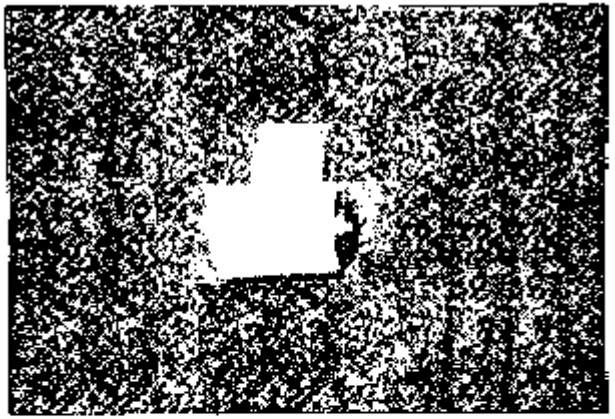


Fig 5

One would expect that the histogram will be a bi-model with a clearly defined valley, and hence a threshold value.

Fig. 6 Histogram of BNC Connector

The histogram of the BNC Connector (Fig. 5) shows that it is not bi-modal, and the placement of the threshold value is very difficult.

Difficulties in Thresholding

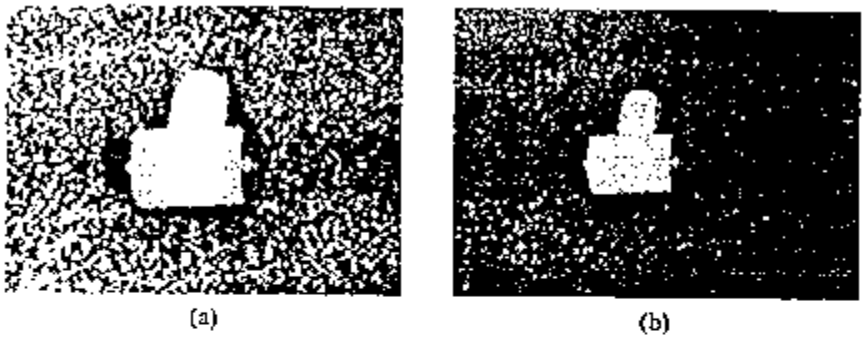


Fig 7, image of the BNC Connector under different threshold values:
(a) 110; (b) 130; (c) 150 and (d) 170.

Threshold too Low -> label more pixel as bright.

Threshold too high -> label more pixel as dark. The object of interest becomes smaller!!

Difficulties in Thresholding

Influence of Noise

If we have a black object on a white background, the histogram will be bi-modal. With the influence of measurement noise, the histogram will be affected as shown in Fig. 8.

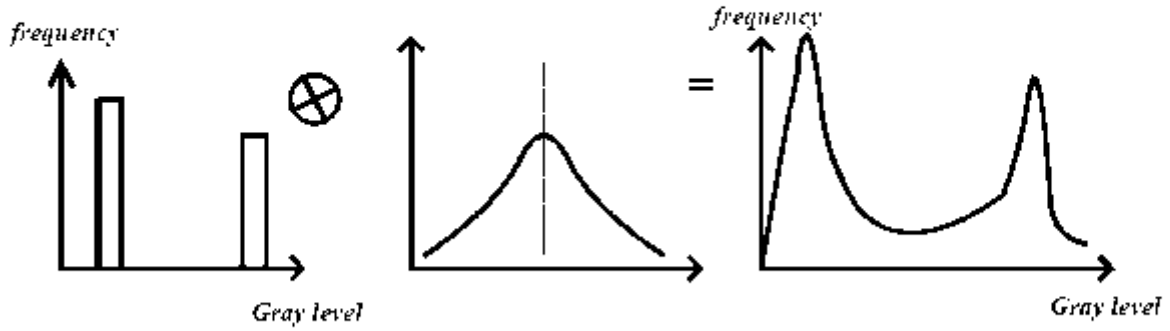


Fig. 8: Result of convolution of an ideal histogram with a noise probability distribution.

Difficulties in Thresholding

Influence of Noise

If the gray levels of the object and the background are fairly close, the influence of noise may result in the object only appearing as a shoulder in the histogram (Fig. 9). The threshold is also difficult to identify. This problem could be due to poor lighting conditions.

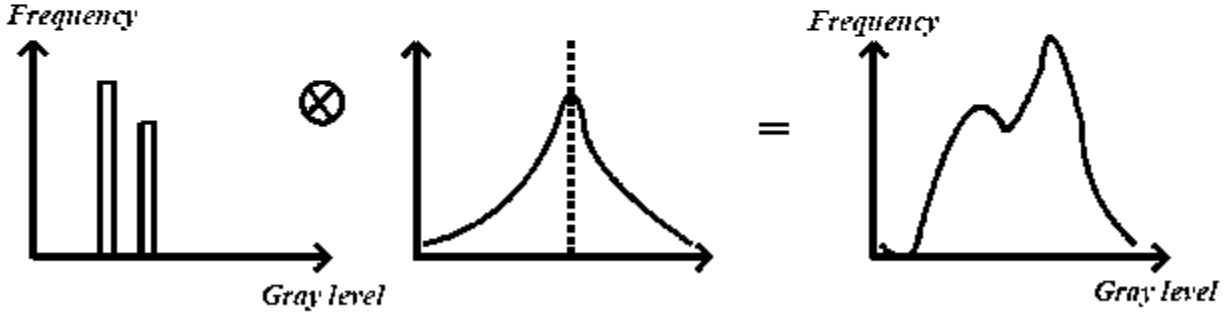


Fig. 9: Result of convolution of an ideal histogram (with two closely positioned peaks) with a noise probability distribution.

Roles of illumination

The formation of an image is the product are the illumination and the reflectance components:

$$f(x,y) = i(x,y) \bullet r(x,y) \qquad (3)$$

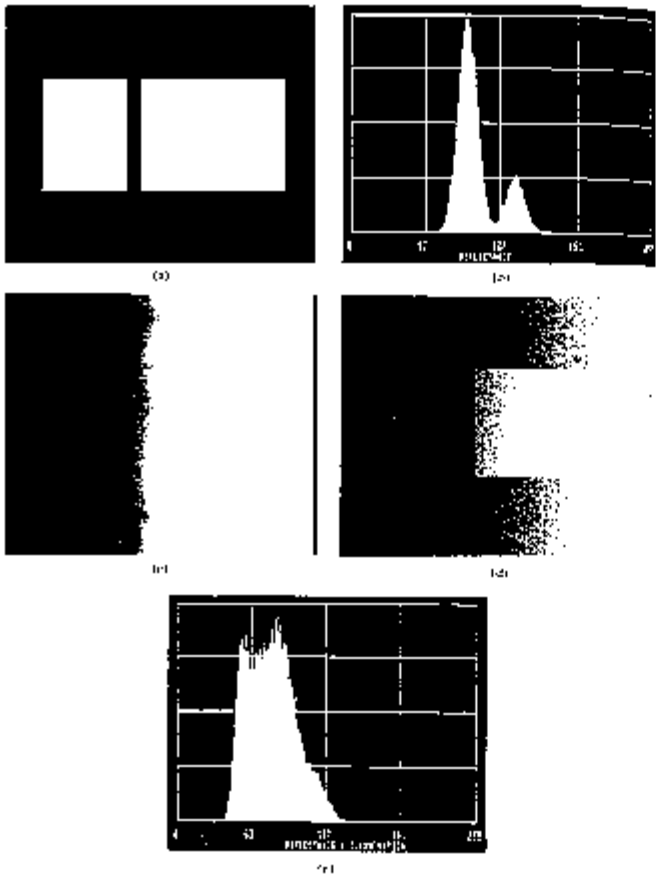
where *i(x,y)* and *r(x,y)* are the illumination and reflectance, respectively.

Fig. 10 shows the effect of illumination.

FIg.10(a) and Fig 10(c) are the computer generated reflectance and illumination functions, respectively. Fig. 10(b) is the well defined bi-modal histogram of Fig. 10(a).

Fig 10(d) is obtained through Eq. (3), and the resulting image is shown in Fig. 10(d) with the histogram shown in Fig. 10(e).

Roles of illumination



- The histogram of the image shown in Fig. 10(e) does not have a clearly defined valley.
- This will make the segmentation with a single threshold value very difficult. This simple illustration shows that the reflective nature of objects and background could be such that they are separable.
- However, an image resulting from poor illumination could be difficult to separate

Fig. 10

Thresholding Techniques

Global Thresholding

This is a simple technique and is useful to segment an image by dividing the gray levels into bands and use thresholds to determine regions or to obtain object boundaries.

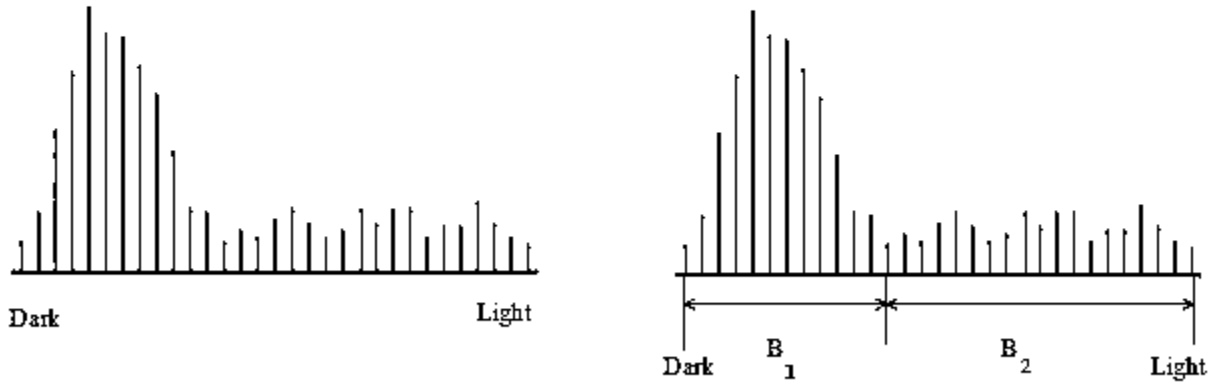


Fig. 11 Histogram Thresholding

Global Thresholding

Global Thresholding

This is a simple technique and is useful to segment an image by dividing the gray levels into bands and use thresholds to determine regions or to obtain object boundaries.

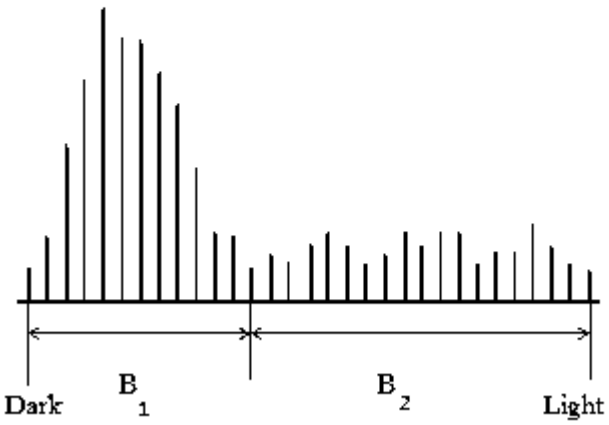


Fig 11(b) is the histogram of an image *f(x,y)*.

We would like to outline the boundary between the objects and the background, by dividing the two bands separated by a threshold ***T***.

We would like to select a ***T*** such that:

Band B_1 : Contains as closely as possible, levels associated with the background.

Band B_2: Contains as closely as possible, levels associated with the objects.

Global Thresholding

Basis

The method proceeds by scanning the given image $f(x,y)$ of size N by N. A ***change*** in gray level from one band to the other denotes the presence of a boundaries.

Scanning is done is two passes:

Pass 1: for each row in $f(x,y)$, i.e. $x = 0, 1, \ldots, N-1$;

Pass 2: for each column in $f(x,y)$, i.e., $y = 0, 1, \ldots, N-1$.

Pass 1 and 2 are to detect changes in the y and x directions, respectively. The combination of the results of the two passes will yield the boundary of the objects in the image.

Global Thresholding

Pass 1

For each row in ***f(x, y)***, i.e., ***x*** = 0, 1, ..., N-1, create a corresponding row in an intermediate image ***g1 (x, y)*** using the following relation for ***y*** = 0, 1,., N-1:

$$g1(x,y) = \begin{cases} LE & \text{if the level of } f(x,y) \text{ and } f(x,y-1) \text{ are in different bands of the gray scale,} \\ LB & \text{otherwise.} \end{cases} \quad (4a)$$

where ***LE*** and ***LB*** are specified edge and background levels, respectively.

Global Thresholding

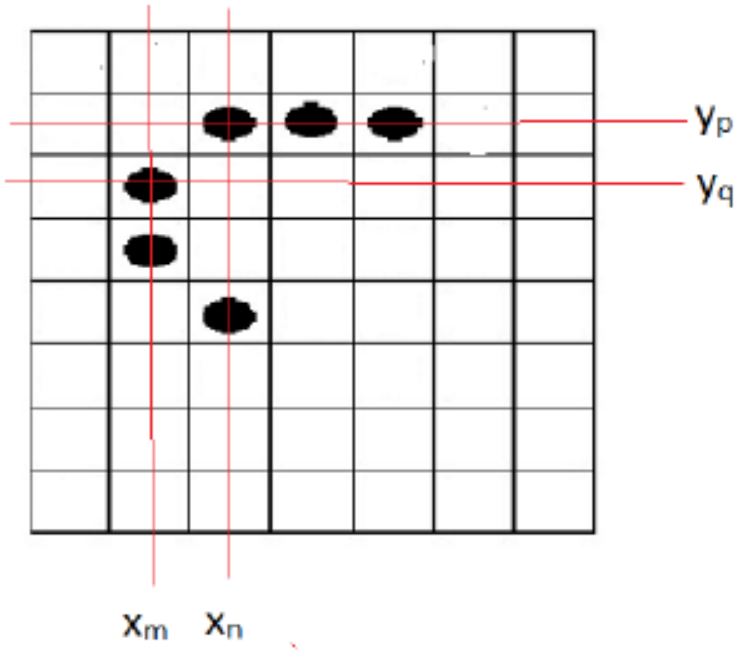
Pass 2

For each row in ***f(x, y)***, i.e., ***y*** = 0, 1, ..., N-1, create a corresponding row in an intermediate image ***g2 (x, y)*** using the following relation for ***x*** = 0, 1,., N-1:

$$g2(x,y) = \begin{cases} LE & \text{if the level of } f(x,y) \text{ and } f(x-1,y) \text{ are in different bands of the gray scale,} \\ LB & \text{otherwise.} \end{cases} \quad (4b)$$

where ***LE*** and ***LB*** are specified edge and background levels, respectively.

Global Thresholding



$$f(x_n, y_p) \text{ and } f(x_{n+1}, y_p) \Rightarrow g_2(x_n, y_p) = L_B$$
$$f(x_n, y_p) \text{ and } f(x_n, y_{p-1}) \Rightarrow g_1(x_n, y_p) = L_E$$

$$f(x_m, y_q) \text{ and } f(x_{m-1}, y_q) \Rightarrow g_2(x_m, y_q) = L_E$$
$$f(x_m, y_q) \text{ and } f(x_m, y_{q\div 1}) \Rightarrow g_1(x_m, y_q) = L_B$$

Global Thresholding

Putting them together

The desired image, consisting of the points on the boundary of objects different (as defined by ***T***) from the background, is obtained by using the following relation for ***x***, ***y*** = 0, 1, ..., N-1:

$$g(x, y) = \begin{cases} \textbf{LE} \text{ if } g1(x, y) \text{ or } g2(x, y) \text{ is equal to } \textbf{LE}, \\ \textbf{LB} \text{ otherwise.} \end{cases} \quad (5)$$

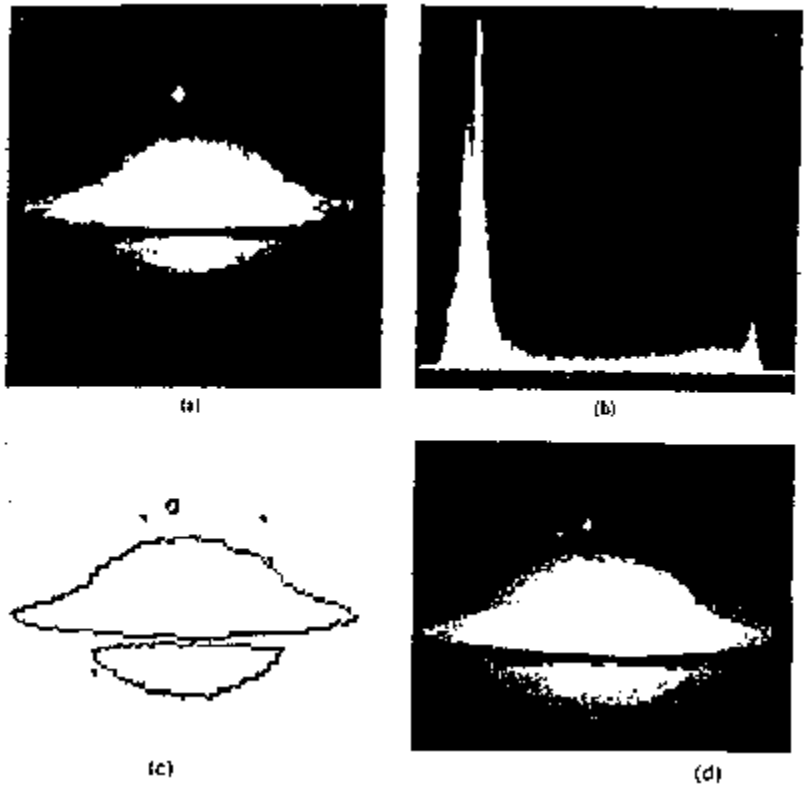
- The above procedure can be generalised to more gray-level bands as the relations in Eq. (4) and (5) are based only on a change in gray-level band from one pixel to the next.
- One possible extension would be to code the edge points with different gray-levels, depending on the band in which the change took place.
- The problem in this multi-bands situation is where to place the thresholds.

Global Thresholding

Results:

Fig. 12 shows an example of edge detection using thresholding.

(a) Picture of a Somberero Nebula.
(b) Histogram
(c) Edge image obtained with *T=128*.
(d) Edge superimposed on original.



Optimal Thresholding

Suppose that an image contains only two principal brightness regions. The histogram of such an image may be considered an estimate of the brightness probability density function ***p(z)***.

This overall density function is the sum or mixture of two uni-modal densities, one for the bright and one for the dark regions in the image.

In addition, the mixture parameters are proportional to the areas of the picture of each brightness.

If the form of the densities is known or assumed, we can determine an optimal threshold (in terms of minimum error) for segmenting the image into two brightness regions.

Note:
The probability for an event ***X*** to occur is given by its probability density function as follows:

$$X = \int_{-\infty}^{X} p(z)dz \qquad (6)$$

Optimal Thresholding

Suppose that an image can be represented by a probability density function $p(z)$, which in turn is the linear combination of the products of the *á priori probability* and the associated Gaussian noise of the two uni-modal brightness regions.

The probability density function can be written as:

$$p(z) = P_1 p_1(z) + P_2 p_2(z) \tag{7}$$

where
$p_1(z)$ and $p_2(z)$ are the Gaussian noise distributions of the dark and bright regions, respectively.

P_1 and P_2 are the *á priori probabilities* of the dark and bright regions, respectively.

Note that $P_1 + P_2 = 1$, and they represent the proportions that the two brightness regions occupy in the image. If $P_1 = 0.40$, $P_2 = 0.60$, then the image has 40% of dark pixels and 60% of bright pixels.

Optimal Thresholding

If the two noise distributions are Gaussian, then Eq (7) becomes:

$$p(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1}\exp\left[-\frac{(z-\mu_1)^2}{2\sigma_1^2}\right] + \frac{P_2}{\sqrt{2\pi}\sigma_2}\exp\left[-\frac{(z-\mu_2)^2}{2\sigma_2^2}\right] \quad (8)$$

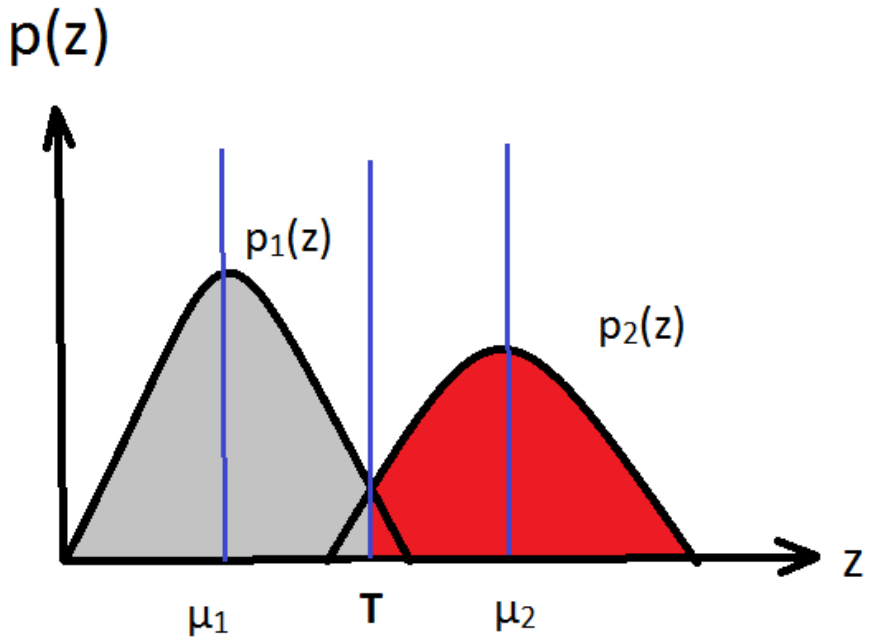
where μ_1 and μ_2 are the mean values of the two brightness regions, and σ_1 and σ_2 are the standard deviations about the respective means.

Note:

á priorí probability can best be explained by the card drawing example in the study of probability. The probability of getting an Ace when drawing a card from a 52-card deck is 4/52 or 1/13. This number is known as the *priorí probability* of this event.

$$P_1 + P_2 = 1 \quad (9)$$

Optimal Thresholding



P_1 and P_2

are the *á priori probabilities* of the dark and bright regions, respectively.

Now, if

Dark Regions --> *Background;*

and

Bright Regions –> Objects.

Hence $\mu_1 < \mu_2$

T is the threshold

$-\infty \le z < T$ ***Background***;

and

$T \le z \le \infty$ ***Objects***

Optimal Thresholding

Now, if

Dark Regions --> *Background;* and Bright Regions --> Objects.

Hence $\mu_1 < \mu_2$

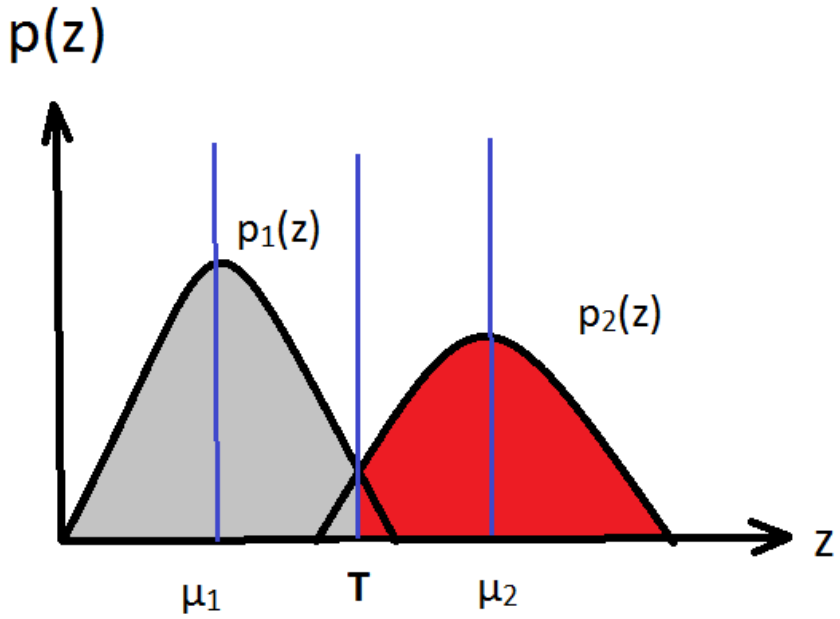
A threshold *T* may be defined such that:

- All pixels where *f(x,y) < T are considered as background points.*
- All pixels where *f(x,y) > T are considered as object points.*

In our probability density function (Eq. (6)), the limit of integration would be::

$-\infty \le z < T$	***Background***; and
$T \le z \le \infty$	***Objects***

Optimal Thresholding



Overall Error Probability:

$$E(T) = P_2 E_1 + P_1 E_2$$

Correct Classification:

Object point $C_1(T) = \int_T^{\infty} p_2(z)\ dz$

Background point $C_2(T) = \int_{-\infty}^{T} p_1(z)\ dz$

Mis- Classification:

Object point $E_1(T) = \int_{-\infty}^{T} p_2(z)\ dz$

Background point $E_2(T) = \int_T^{\infty} p_1(z)\ dz$

Note the difference in the limits of integration

Optimal Thresholding

The basis of Optimal Thresholding is based on the **minimizing the errors in wrongly classifying** an object point as a background point, or vice-versa.

The probability of erroneously classifying an object point as a background point is given by:

$$E_1(T) = \int_{-\infty}^{T} p_2(z)dz \tag{10}$$

The probability of erroneously classifying a background point as an object point is similarly given by:

$$E_2(T) = \int_{T}^{\infty} p_1(z)dz \tag{11}$$

The overall probability of error is then given by:

$$E(T) = P_2 E_1(T) + P_1 E_2(T) \tag{12}$$

Optimal Thresholding

To determine the threshold for which the error in classification is the minimum (Optimal Threshold) we differentiate *E(T) in* Eq. (12) with respect to *T* (using *Liebnitz's rule*), and equate the result to zero:

$$P_1 p_1(T) = P_2 p_2(T) \tag{13}$$

With

$$p_1(T) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left[-\frac{(T-\mu_1)^2}{2\sigma_1^2}\right] \quad p_2(T) = \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left[-\frac{(T-\mu_2)^2}{2\sigma_2^2}\right]$$

Optimal Thresholding

From Eq. (13), we will obtain the following quadratic equation:

$$AT^2 + BT + C = 0 \tag{14}$$

where

$$A = \sigma_1^2 - \sigma_2^2$$

$$B = 2\left(\mu_1\sigma_2^2 - \mu_2\sigma_1^2\right) \tag{15}$$

$$C = \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2\sigma_2^2 \ln\left(\frac{\sigma_2 P_1}{\sigma_1 P_2}\right)$$

There are two possible solutions indicating that there are two optimal threshold values.

Optimal Thresholding

If the two variances are equal, i.e

$$\sigma^2 = \sigma_1^2 = \sigma_2^2$$

then a single threshold is sufficient:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln\left(\frac{P_2}{P_1}\right) \quad (16)$$

If the two *á priori probabilities are equal,* then the optimal threshold would just be the average of the means. The same is true when the standard deviation is zero.

The above analysis for finding the optimal threshold maybe similarly accomplished for other uni-modal densities of known form, such as the Rayleigh and long-Normal densities.

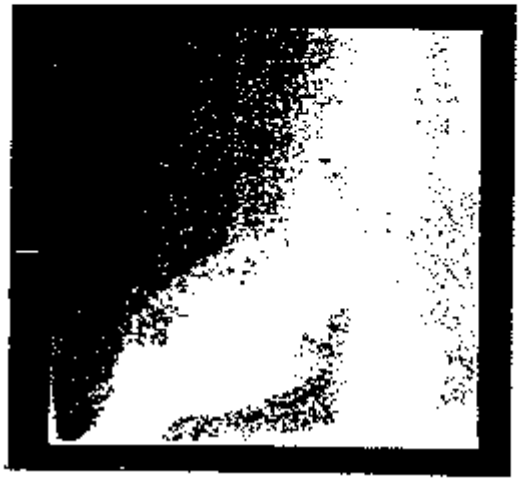
Note:Leibnitz's rule for differentiating an integral is given as: $\frac{d}{dt}\left(\int_c^t f(x)dx\right) = f(t)$ (17)
where *c is a constant.*

Example

Chow and Kaneko has developed an approach for outlining boundaries of the left ventricle in cardio-angiograms (i.e. x-ray pictures of a heart that has been injected with a dye) by using optimal thresholding selection.

Note:

Chow, C.K. and Kaneko, T., "Automatic Boundary Detection of the Left Ventricle from Cardioangiograms,: *Comp. and Biomed. Res.*, vol. 5, 1972, pp. 388-410.

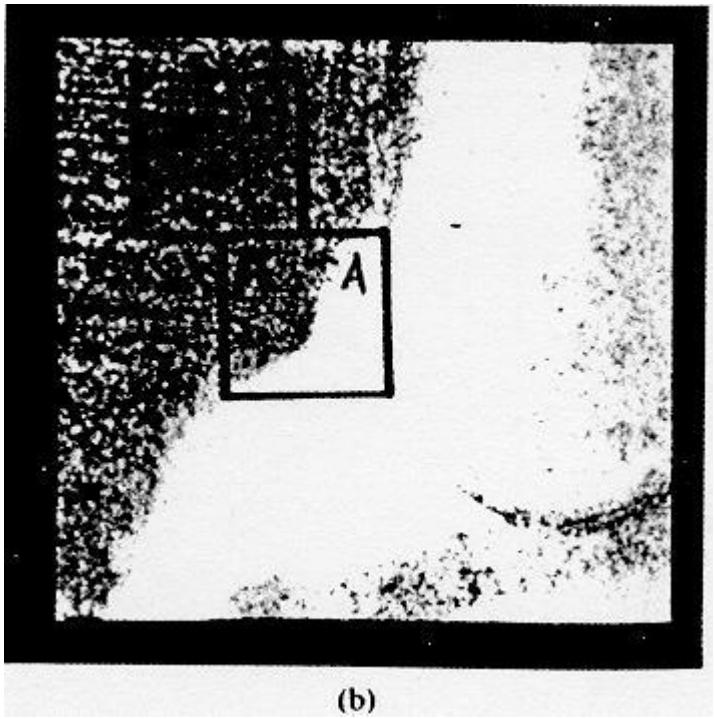


(a)

Fig. 13 A cardioangiogram (a) before processing.

The original x-ray was pre-processed to remove noise and effects caused by radioactive adsorption. (fig. 13(a)

Example



(b)

Fig. 13 A cardioangiogram (b) after processing.

A subtraction was done on two images that were obtained before and after the dye agent was applied in order to remove the un-wanted details (Fig. 13(b)).

Each processed image was sub-divided into 49 regions by placing a 7x7 grid with 50% overlap over each image. Each of the 49 regions contained 64x64 pixels. Fig. 14(a) and (b) are the histograms of the regions marked ***A*** and ***B*** in Fig. 13(b).

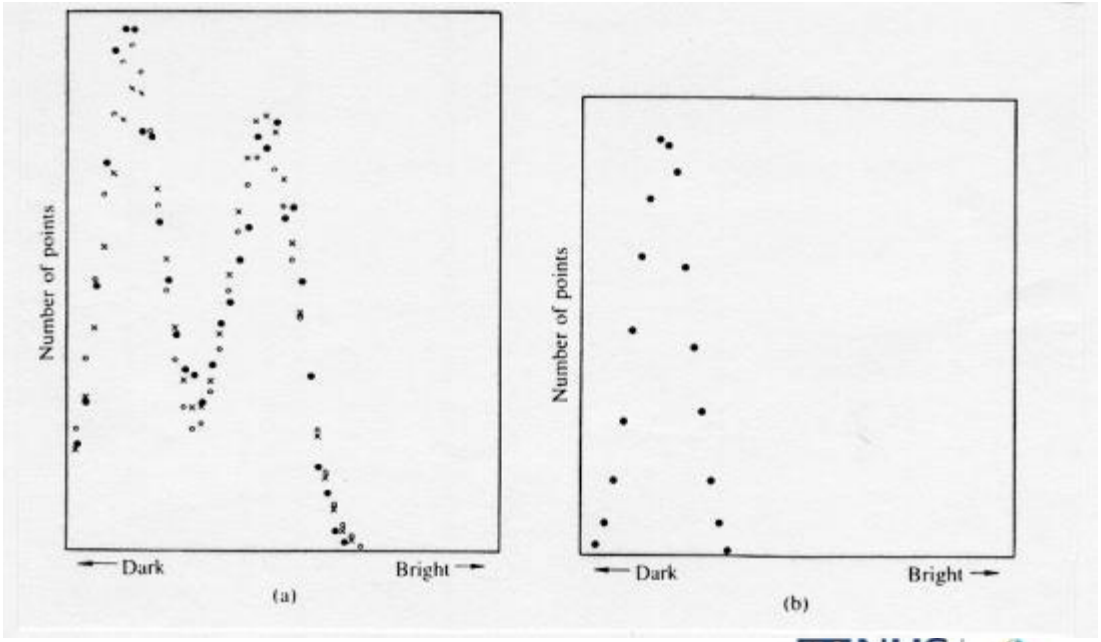
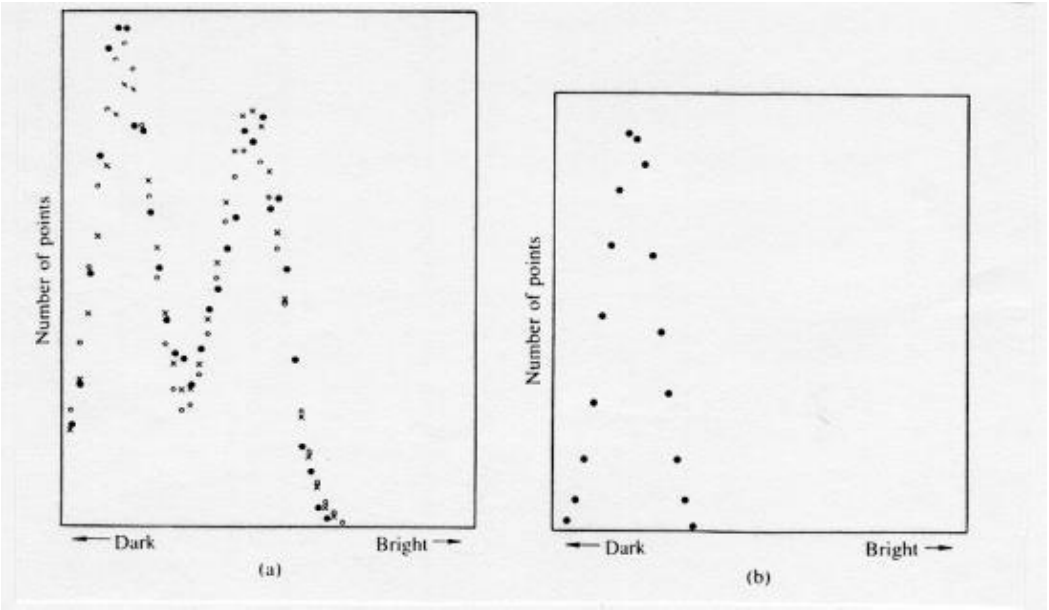


Fig. 14 Histograms (black dots) of regions A and B in Fig. 13(b).

Example

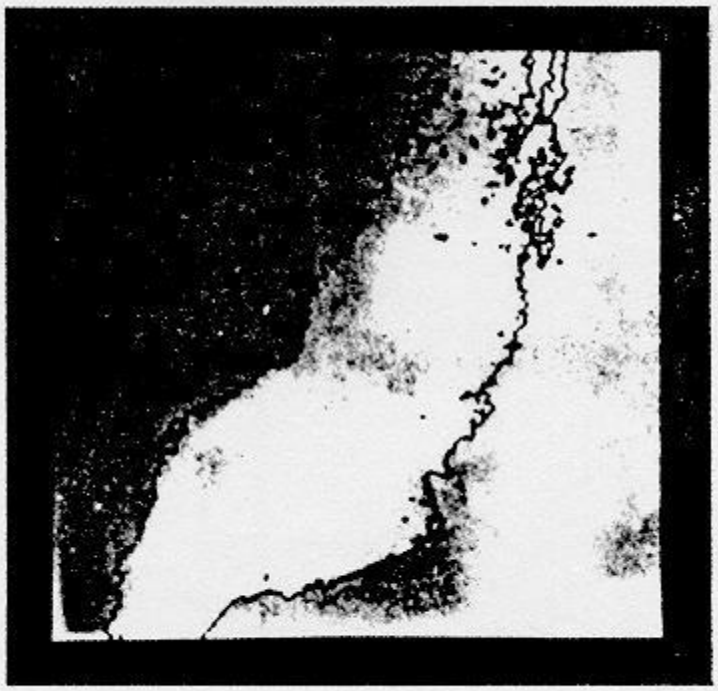


The histogram for region ***A*** is clearly bi-modal, indicating the presence of a boundary. The histogram of region ***B***, is however, uni-modal, indicating the absence of two markedly distinct regions.

After all the 49 histograms were computed, a test of bi-modality was done to reject the uni-modal histograms.

The remaining histograms were then fitted by bi-modal Gaussian density curve (eq. (8)) using an error minimising technique (please refer to their paper for details). The crosses (x) and circles in Fig. 14(a) are two fits to the histogram shown in black dots. The optimal thresholds were then obtained by using Eq. (15) and (16).

Example



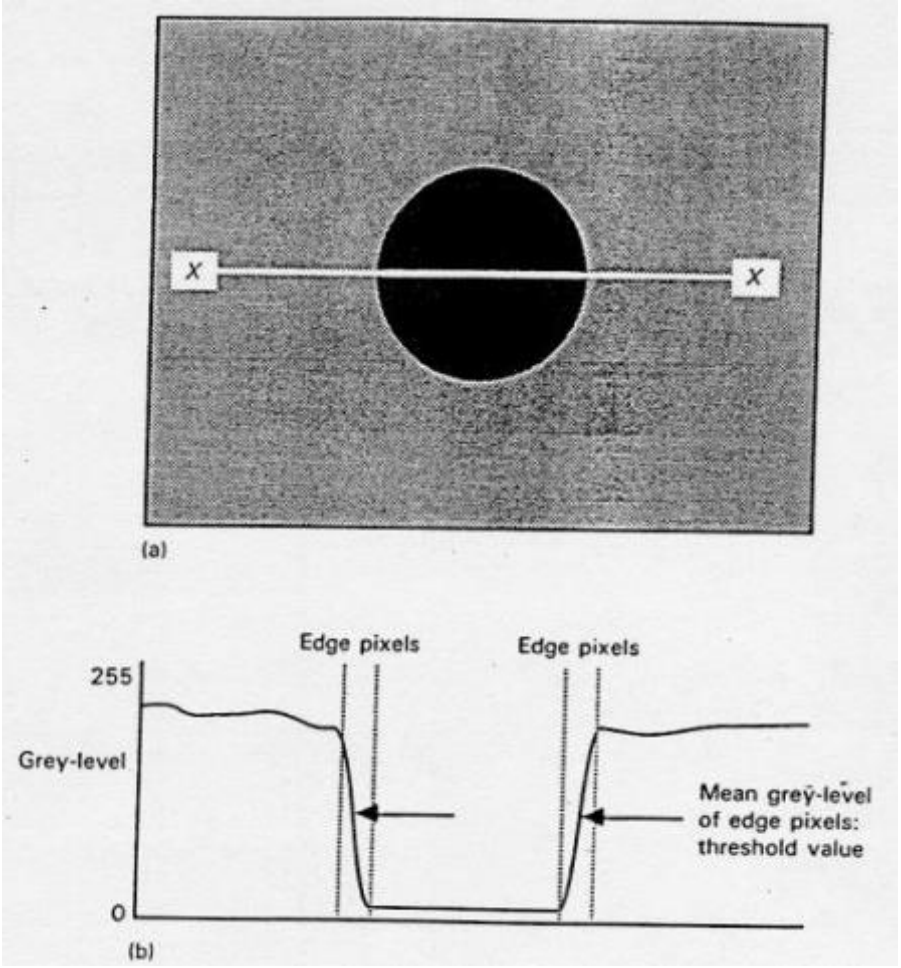
At this stage, the regions with bi-modal histograms were assigned thresholds. The thresholds for the remaining regions were obtained by interpolating these thresholds. Then a second interpolation was carried out point by point by using neighbour threshold values. so that at the end of the procedure, every point in the image had been assigned a threshold.

Finally, a binary decision was carried out for each pixel using the following rule:

$$f(x,y)=\begin{cases}1 & \text{if } f(x,y) \geq T_{xy} \\ 0 & \text{otherwise}\end{cases} \qquad (18)$$

where ***Txy*** was the threshold assigned to location ***(x,y)*** in the image. Boundaries were then obtained by taking the gradient of the binary image. Fig. 15 shows the boundaries superimposed on the original image.

Threshold Selection based on Boundary Characteristics



There are usually abrupt changes in gray level values at the boundaries between objects and background. One way to select a threshold is to consider only those pixels that lie on or near the boundary between objects and the background. (Fig. 16). Finally, using pixels that satisfy some simple measures based on gradient and Laplacian operators has a tendency to deepen the valley between histogram peaks. The main difficult in this case, lies in deciding which pixels are on the boundary. We must look for a good edge detector.

Fig. 16 Using edge pixels to select a threshold:
(a) image of dark, round object on a light background with section X-X shown; (b)Profile of image intensity along section X-X.

Component Labeling

- Connected components analysis of a binary image consists of the **connected components labeling** of the binary 1 pixels followed by property measurement of the component regions and decision making.

- **The connected components labeling** operation performs the unit change from pixel to region of *segment.*

- All pixels that have value binary 1 are given identifying **labels** depends on some conditions.

- The label is a unique name or index of the **region to which the pixels belong**. The label is the identifier for a potential object region.

Component Labeling

- Connected components labeling is a grouping operation that can make a unit change from pixel to region, which is a more complex unit.

- A region has a rich set of properties. It has shape and position properties as well as statistical properties of the gray levels of the pixels in the region.

- In this section, we will examine connected component labeling algorithms, which in essence **group together all pixels belonging to the same region and give them the same label.**

Connected Components Operators

- Once a binary image is produced, a connected components labeling operator is employed to group the binary-1 pixels into maximal connected regions. These regions are called the ***connected components*** of the binary image, and the associated operator is called the ***connected components operator***.

- Its input is a binary image and its output is a symbolic image in which the label assigned to each pixel is an integer uniquely identifying the connected component to which that pixel belongs.

Connected Components Operators

Connectivity

For the connectivity, it is imperative to first define the type of connectivity adopted. In most cases, they are the 4-connectivity and 8 connectivity.

(a) (b)

Fig. 17 (a) Pixel that are 4-connected to the centre point

(b) Pixel that are 8-connected to the centre point x.

Connected Components Operators

Definition

Two binary-1 pixels, ***p*** and ***q*** belong to the same component ***C*** if there is a sequence of 1 pixels (***po , p1 , . . . , pn)*** of ***C*** where ***po*** = ***p***, ***pn*** = ***q***, and ***pi*** is a neighbour of ***pi-1*** for ***i*** = ***1, 2, . . , n.***

- The definition of a connected component depends on the definition of neighbour. Fig 21 (a) and 21 (b) shows a 4-connected region and an 8-connected regions, respectively.
- Whichever definition is used, the neighbours of a pixel are said to be *adjacent* to that pixel. The ***border*** of a connected component of 1-pixels is the subset of pixels belonging to the component that are also adjacent to 0-pixels, and vice-versa

Connected Components Algorithms

General Principles:

- All the algorithms process a row of the image at a time.
- Modifications to process a sub-image rectangular window at a time are straightforward.
- All the algorithms assign **new labels** to the **first pixel of each component** and attempt to propagate the label of a pixel to its neighbours

Connected Components Algorithms

A simple illustration:

Fig. 22 and assume 4-connectivity; left-right top-down scan

1. In the first row two 1-pixels separated by three 0-pixels are encountered. The first is assigned label 1; the second, label 2.
2. In row 2, the first 1-pixel is assigned label 1 because it is a 4-neighbour of the already labeled pixel above it. The second 1-pixel of row 2 is also assigned label 1 because it is a 4-neighbour of the already labeled pixel on its left.
3. The process continues until the pixel marked A in row 4 is encountered. Pixel A has a pixel labeled 2 above it, and it connects regions 1 and 2. Thus all the pixel labeled 1 and all the pixels labeled 2 really belong to the same component; in other words, label 1 and 2 are equivalent.

Connected Components Algorithms

0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	0	1	1	1	0	1
0	0	1	1	1	1	1

(a)

0	0	1	0	0	0	2
0	0	1	1	0	0	2
0	0	1	1	1	0	2
0	0	1	1	1	1	A

(b)

Fig. 18 Propagation Process: Part (a) shows the original binary image, and (b) the partially processed image.

Propagation process:

Label 1 has been propagated from the left to reach pixel A. Label 2 has been propagated down to reach pixel A. The connected components algorithm must assign a label A and make labels 1 and 2 **equivalent.**

An Iterative Algorithm

This iterative algorithm uses no auxiliary storage to produce the labeled image from the binary image. It would be useful in environments whose storage is severely limited or transputer hardware.

The algorithm essentially consists of the following steps (Fig. 23):

1. An initialisation step where each pixel is given a unique label.

2. A top-down pass, and on each row, a left-right pass in which the value of each non zero pixel is replaced by the minimum value of its non-zero neighbours in a recursive manner.

3. A bottom-up pass, and a right-left passes with the same recursive procedure as in step 2.

This algorithm selects the minimum label of its neighbours to assign to pixel A. It does not directly keep track of equivalences but instead uses a number of passes through the image to complete the labeling

An Iterative Algorithm

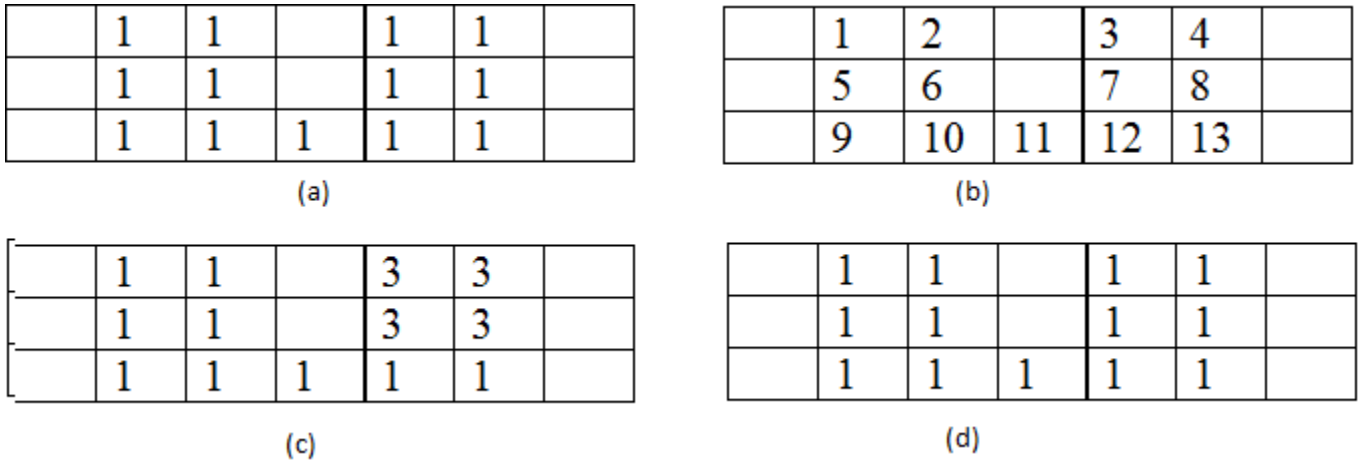


Fig. 19 Iterative algorithm for connected component labeling.
(a) shows the original image;
(b) the results after initialization of each 1-pixel to a unique label;
(c) the results after the first top-down pass, in which the value of each non-zero pixel is replaced by the minimum value of its non-zero neighbours in a recursive manner going from left to right and top to bottom; and
(d) the results after the first bottom-up pass.

This algorithm alternates top-down, left-to-right passes with bottom, right-to-left passes so that labels near the bottom or right margins of the image will propagate sooner than if all passes were top-down, left-to-right.

The classical algorithm

This algorithm makes only two passes through the image, but requires a large global table for recording equivalences. The steps can be summarized below:

1. The first pass performs label propagation, just like the procedure described above. Whenever a situation arises in which two different labels can propagate to the same pixel, the smaller label propagates and each such equivalence found is entered in an equivalence table.

Each entry in the equivalent table consists of an ordered pair, the values of its components being the labels found to be equivalent.

2. After the first pass, the equivalent classes are found by taking the transitive closure of the set of equivalences recorded in the equivalent table. Each equivalent class is assigned a unique label, usually by the minimum label in the class.

3. A second pass through the image performs a translation, assigning to each pixel the label of the equivalence class of its pass-1 label.

The classical algorithm

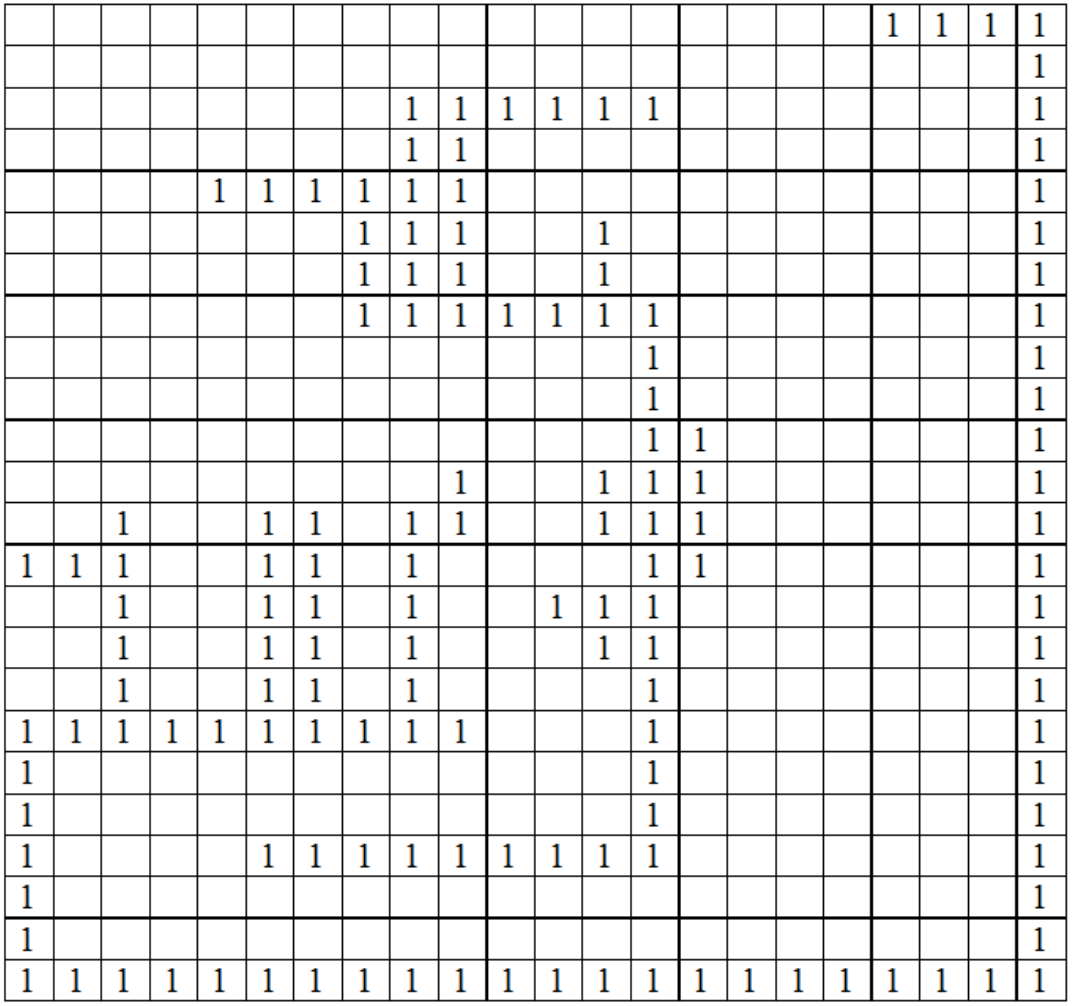


Fig. 20(a) Classical connected components labeling algorithm: The initial binary image.

The classical algorithm

																		1	1	1	1
																					1
								2	2	2	2	2	2								1
								2	2												1
				3	3	3	3	2	2												1
							3	2	2			4									1
							3	2	2			4									1
							3	2	2	2	2	2	2								1
													2								1
													2								1
													2	2							1
									5			6	2	2							1
		7			8	8		9	5			6	2	2							1
10	10	7			8	8		9					2	2							1
		7			8	8		9			11	11	2								1
		7			8	8		9				11	2								1
		7			8	8		9					2								1
12	12	7	7	7	7	7	7	7	7				2								1
12													2								1
12													2								1
12					13	13	13	13	13	13	13	13	2								1
12																					1
12																					1
12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	1

Fig. 20(b):
Classical connected components labeling algorithm:
The labeling after the first up-down pass of the algorithm.

The equivalence classes found are:
1: {1, 12, 7, 8, 9, 10, 5},
and 2: {2, 3, 4, 6, 11, 13}.

The classical algorithm

Essentially, the initial equivalence tables pairs are:

{2,3}, {2,4}, {2,6}, {5,9}, {7,10}, {2,11}, {7,12}, {7,8}, {7,9}, {2, 13}, {1,12}

After using the graph search method to resolve the table entries, we have:

{1,12}, {7,12}, {7,8}, {7,9}, {7,10}, {5,9} and

{2,3}, {2,4}, {2,6}, {2,11}, {2,13}, from which we can derive the equivalent classes as:

1: {1, 12, 7, 8, 9, 10, 5}, and

2: {2, 3, 4, 6, 11, 13}.

The main problem with the classical algorithm is the global equivalence table. For large image with many regions, the equivalence table can become very large. On some machines, the memory may not be able to hold the table.

A Space Efficient Two-Pass Algorithm using Equivalence Table

- A solution to the classical algorithm is to use a smaller local equivalence table that only stores the equivalence detected from the current line of the image and the line that precedes it.
- Thus, the maximum number of equivalences is the number of pixels per line. These equivalences are then used in the propagation step to the next line.
- In this case, not all the equivalencing is done by the end of the first top-down pass, and a second pass is required for both the remainder of the equivalence finding and for assigning the final labels.
- Note that the second pass is in bottom-up direction.
- Results after the top down pass of this method on the binary image shown in Fig. 20(a) is shown in Fig. 21.

The Space Efficient Algorithm

																		1	1	1	1
																					1
								2	2	2	2	2	2								1
								2	2												1
				2	2	2	2	2	2												1
							2	2	2			4									1
							2	2	2			4									1
							2	2	2	2	2	2	2								1
													2								1
													2								1
													2	2							1
									5			2	2	2							1
		7			8	8		5	5			2	2	2							1
7	7	7			8	8		5					2	2							1
		7			8	8		5			2	2	2								1
		7			8	8		5				2	2								1
		7			8	8		5					2								1
5	5	5	5	5	5	5	5	5	5				2								1
5													2								1
5													2								1
5					2	2	2	2	2	2	2	2	2								1
5																					1
5																					1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 21
Results after the top-down pass of the local table method on the binary image of Fig. 24(a).

Note that on the lines where equivalences were detected, the pixels have different labels from those they had after pass 1 of the classical algorithm.

A Space Efficient Two-Pass Algorithm using Equivalence Table

During the top-bottom pass, the local equivalence table only stores the current line and the line precedes it. The two following examples are worth noting.

On line 5:

After scanning line 5, four 1-pixels would have been labeled '3', and the two following as '2' after checking with the local equivalence table (Refer back to Fig. 20(b)). The algorithm will pick the minimum of the equivalence pairs, and label all the corresponding pixels to this minimum. In this case, the four pixels labeled originally with '3' will be labeled as '2', as '2' is the minimum in the equivalence table entries.

A Space Efficient Two-Pass Algorithm using Equivalence Table

On line 18:

After the scanning, we would have the equivalence table entries as:
{9,7}, {9,8}, {9,5}. The algorithm will pick the minimum amongst the entries, which is 5 in this case, and thence converts all the label in this line to 5. The labels '7' and '8' will eventually be converted to 1 during the bottom-up pass. This is because all the '5' will be converted to '1', and on reaching line 18, the equivalence table at line 17 will have {1,7}, {1,8} and {1,5}. The algorithm will convert all the pixels in each of these branches to label '1'.

Finally, the bottom-up pass will propagate the label 1 and 2 to all pixels of the single connected components.

A Space Efficient Two-Pass Algorithm using Equivalence Table

In comparison with the classical algorithm, this local table method is much faster and uses small memory space. The larger the image, the better the performance is when compared with the classical algorithm.

There are several run-time implementation of the local table method. One such method is advocated by Ronse and Devijver. Please refer to their paper on the implementation details "Connected Components in Binary Images: The detection problem", Research Studies, Letchworth, Herts, England, 1984.

End of Lecture 1