ME 5405 Machine Vision

Binary Machine Vision

KB Lim
Mechanical Engineering Department
National University of Singapore
mpelimkb@nus.edu.sg

1. 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 (“1” in binary image); and
- Low values if they are not (“0” in binary image).

1. Introuction

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

1. 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.

2. Thresholding

2.1 Introduction

- 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.

2.1 Thresholding - Introduction

							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.

2.1 Thresholding Introduction

Determine Threshold Value from Histogram

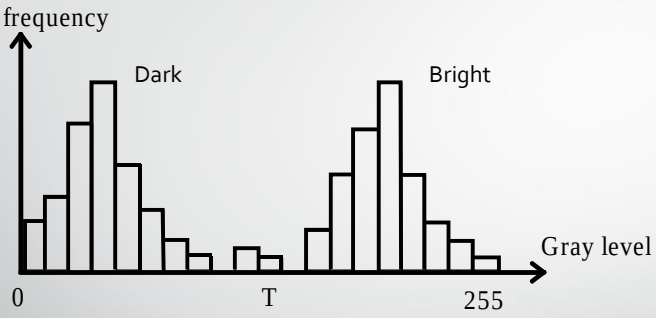


Fig 3. Gray level histogram with single threshold

- 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.

2.1 Thresholding Introduction

Multiple Thresholds

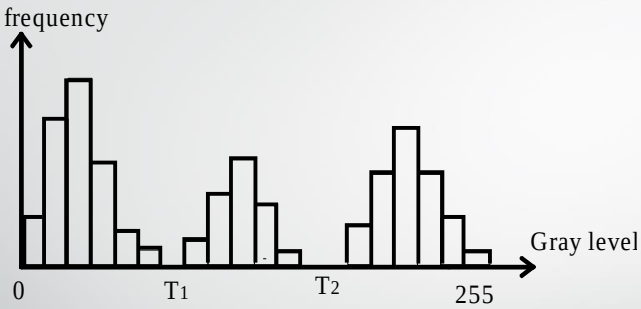
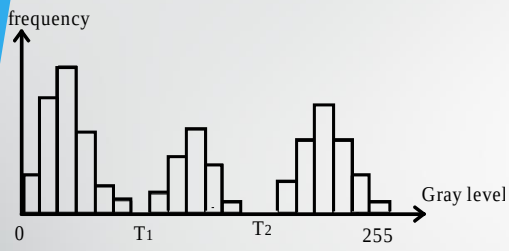


Fig 4. Gray level histogram with multiple thresholds

- 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

2.1 Thresholding Introduction

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.

2.1 Thresholding Introduction

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:

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

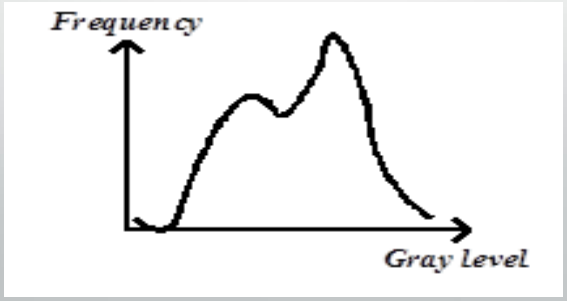
When ***T*** depends only on ***f(x, y)***, the threshold is called ***global***.
If ***T*** depends on both ***f(x, y)*** and ***p(x, y)***, then the threshold is called ***local***.
If ***T*** depends on the spatial coordinates ***x*** and ***y***, the threshold is called ***dynamic***。

2.2 Thresholding - Difficulties

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.



2.2 Thresholding - Difficulties

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

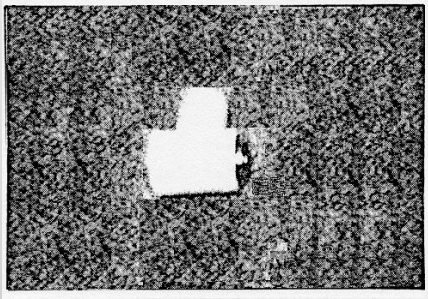
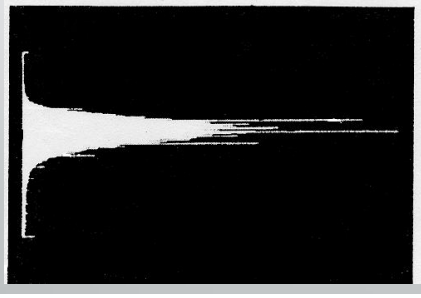


Fig 5

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.

2.2 Thresholding - Difficulties

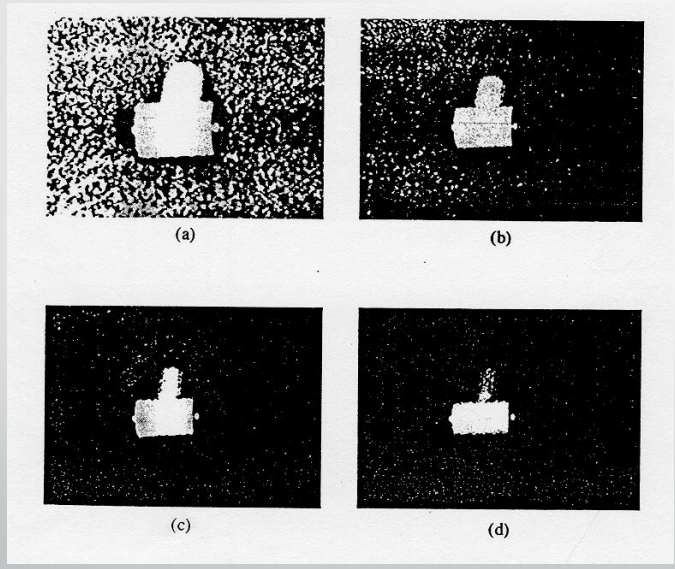


Fig 7, image of the BNC Connector under different threshold values:
(a) 110; (b) 130; (c) 150 and (d) 170.

Threshold too Low -> label more pixel as bright.

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

2.2 Thresholding - Difficulties

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.

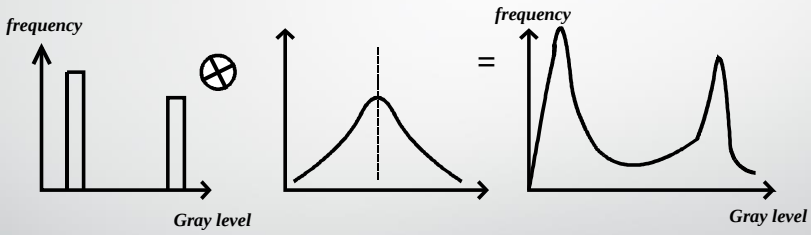


Fig. 8: Result of convolution of an ideal histogram with a noise probability distribution.

2.2 Thresholding - Difficulties

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.

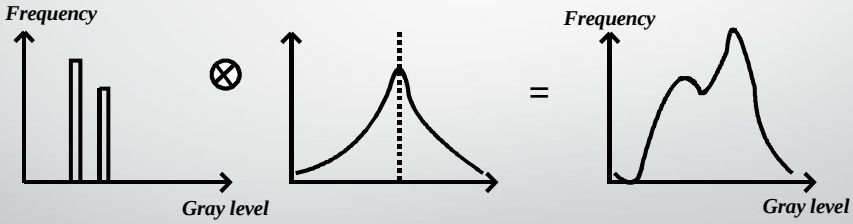


Fig. 9: Result of convolution of an ideal histogram (with two closely positioned peaks) with a noise probability distribution.

2.3 Roles of illumination

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

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

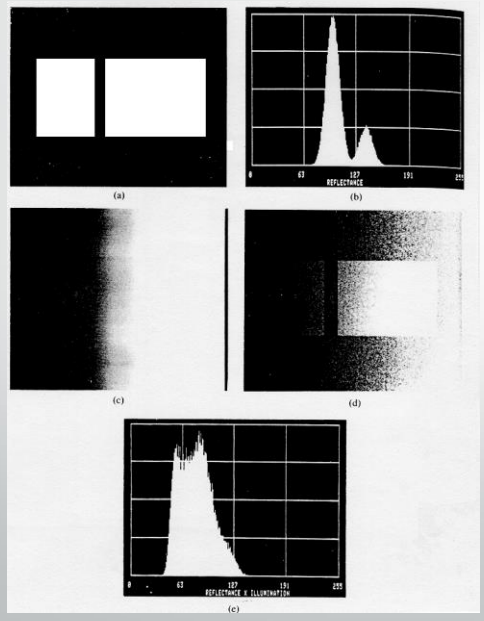
where *i(x,y)* and *r(x,y)* are the illumination and reflectance, respectively.

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).

2.3 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

2.4 Thresholding Techniques

2.4.1. 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.

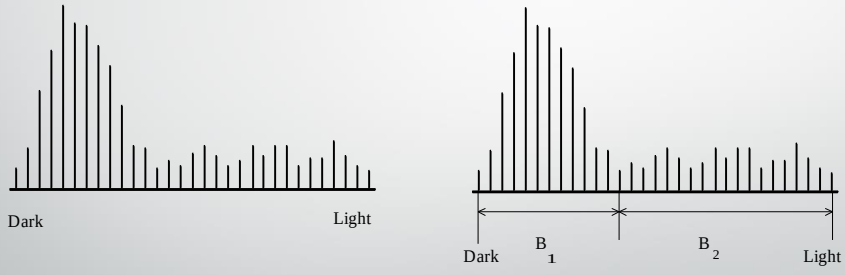


Fig. 11 Histogram Thresholding

2.4.1 Global Thresholding

2.4.1. 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.

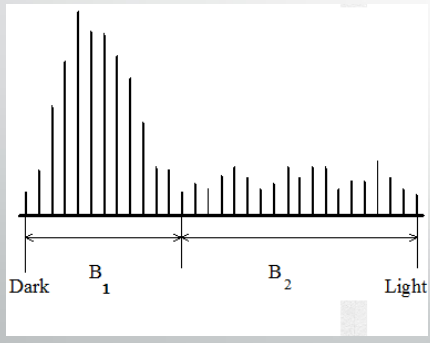


Fig 11(b) is the histogram of an image *f(x,y).*

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.

2.4.1 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, …., N-1*;

Pass 2: for each column in *f(x,y)*, i.e., *y = 0, 1, …..., 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.

2.4.1 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.

2.4.1 Global Thresholding

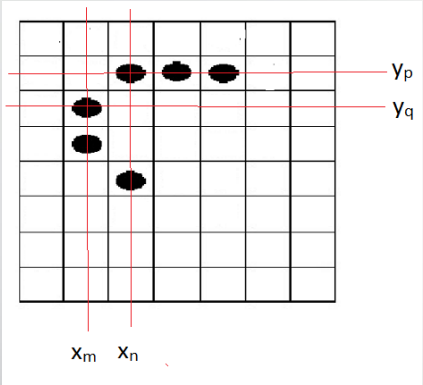
Pass 2

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.

2.4.1 Global Thresholding



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

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

2.4.1 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:

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

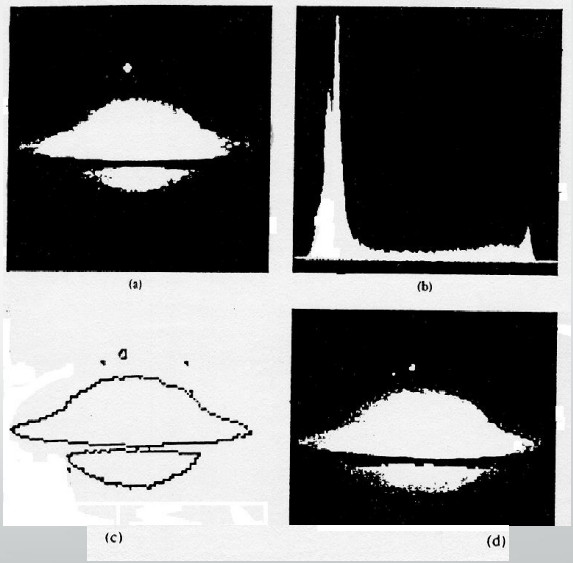
- The above procedure can be generalised to more gray-level bands as the relations in Eq. (4) and (5) are based only on a change in gray-level band from one pixel to the next.
- One possible extension would be to code the edge points with different gray-levels, depending on the band in which the change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

2.4.1 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.



2.4.2 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:

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

2.4.2 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:

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

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.

2.4.2 Optimal Thresholding

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

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

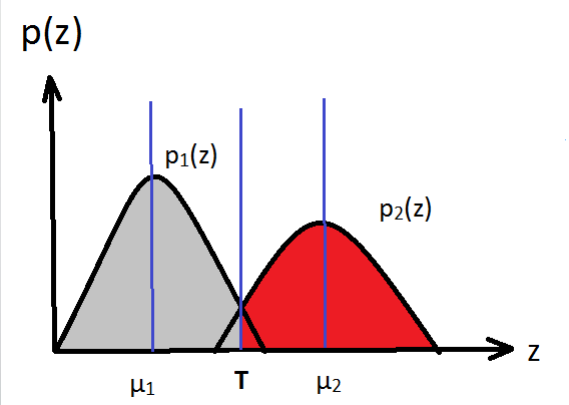
where μ_1 and μ_2 are the mean values of the two brightness regions, and σ_1 and σ_2 are the standard deviations about the respective means.

Note:

á priori 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 *priori probability* of this event.

$$\boldsymbol{P_1 + P_2 = 1} \quad (9)$$

2.4.2 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***

2.4.2 Optimal Thresholding

Now, if

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

Hence $\mu_1 < \mu_2$

A threshold *T* may be defined such that:

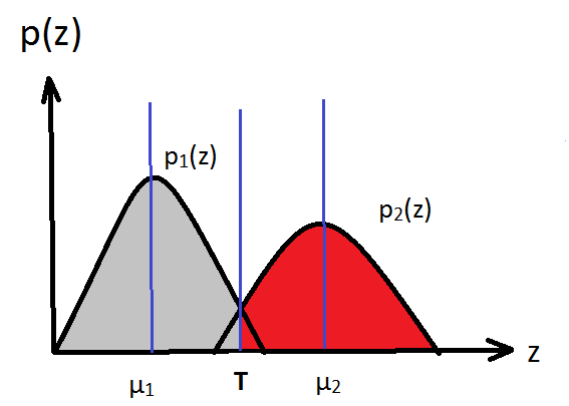
- All pixels where *f(x,y) < T are considered as background points.*
- All pixels where *f(x,y) > T are considered as object points.*

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***

2.4.2 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

2.4.2 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:

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

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

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

The overall probability of error is then given by:

$$E(T) = P_2 E_1(T) + P_1 E_2(T) \qquad (12)$$

2.4.2 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:

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

With

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

2.4.2 Optimal Thresholding

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

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

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.

2.4.2 Optimal Thresholding

If the two variances are equal, i.e

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

then a single threshold is sufficient:

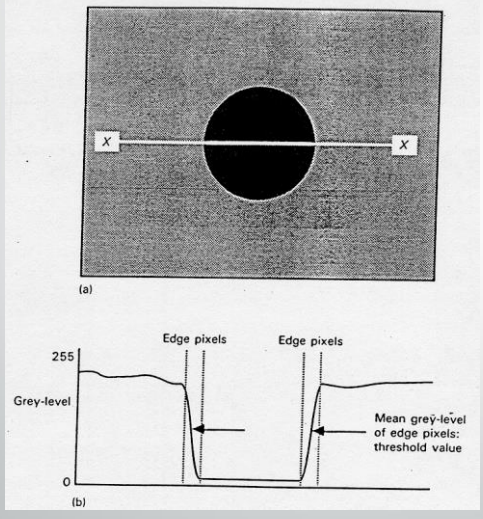
$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln\left(\frac{P_2}{P_1}\right) \quad (16)$$

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.*

2.4.2 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. 13). 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. 13 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.

3. 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.

3. 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.**

3.1. 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.

3.1. 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.

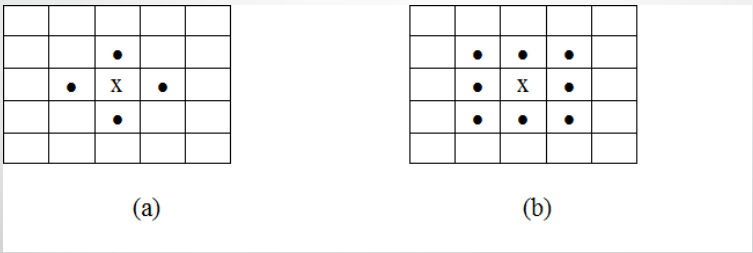


Fig. 14 (a) Pixel that are 4-connected to the centre point
(b) Pixel that are 8-connected to the centre point x.

3.1. 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 14 (a) and 14 (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

3.2. 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

3.2. Connected Components Algorithms

A simple illustration:

Fig. 15 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.

3.2. 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. 15 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.**

3.2.1 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. 16):

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

3.2.1 An Iterative Algorithm

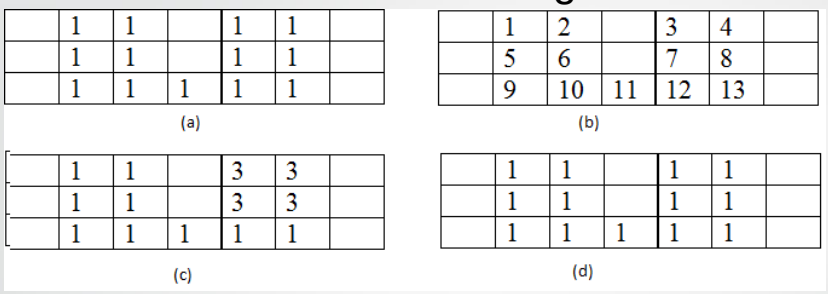


Fig. 16 Iterative algorithm for connected component labeling.
(a) shows the original image;
(b) the results after initialization of each 1-pixel to a unique label;
(c) the results after the first top-down pass, in which the value of each non-zero pixel is replaced by the minimum value of its non-zero 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.

3.2.2 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.

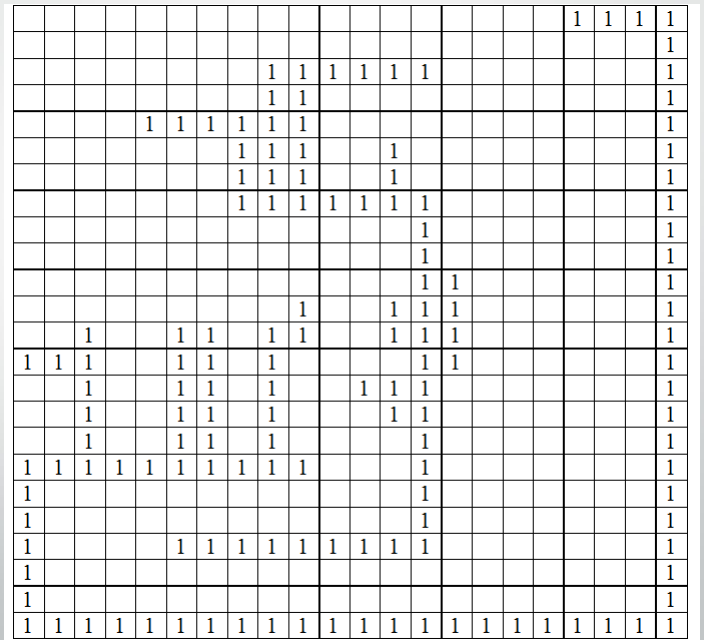
Each entry in the equivalent table consists of an ordered pair, the values of its components being the labels found to be equivalent.

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.

3.2.2 The Classical Algorithm

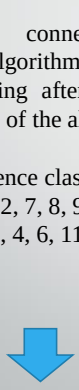
Fig. 17(a) Classical connected components labeling algorithm: The initial binary image.



3.2.2 The Classical Algorithm

Fig. 17(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}.



																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	1

3.2.2 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.

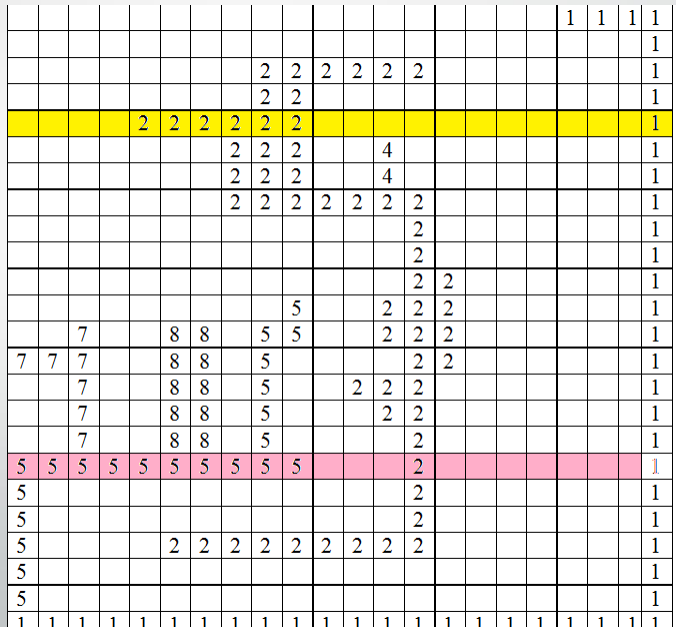
3.2.3 A Space-Efficient Two-Pass Algorithm using Local Equivalence Table

- 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. 17(a) is shown in Fig. 18.

3.2.2 The Space Efficient Algorithm

Fig. 18
Results after the top-down pass of the local table method on the binary image of Fig. 17(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.



3.2.3 A Space-Efficient Two-Pass Algorithm using Local 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. 17(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.

3.2.3 A Space-Efficient Two-Pass Algorithm using Local 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.

3.2.3 A Space-Efficient Two-Pass Algorithm using Local 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.

4. Region Analysis and Properties

This part of the Binary Machine Vision will deal with ***Region Analysis***.

- It consists of computing global properties for each region produced by the connected components labeling algorithm.
- The properties of each region or segment are stored as a measurement vector that is the input to a classifier.
- We shall describe the computation of properties from regions and segments.

A variety of property measurements can be made on each region on the basis of the region's shape and the gray level values for those pixels that participate in the region. The common ones are:

1. Area
2. Perimeter
3. Centroid
4. Moments

4. Region Analysis and Properties

Definitions:

In the discussion that follows, we denote the set of pixels in a region by ***R***. Simple global properties include the region's area ***A*** and centroid ($\bar{r}$, $\bar{c}$). Assuming square pixels:

Area:

$$A = \sum_{(r,c)\in R} 1 \qquad (18)$$

Centroid:

$$\bar{r} = \frac{1}{A}\sum_{(r,c)\in R} r \qquad (19)$$

$$\bar{c} = \frac{1}{A}\sum_{(r,c)\in R} c$$

Note that although ***r*** and ***c*** are integers, but $\bar{r}$, $\bar{c}$ are usually not.

4. Region Analysis and Properties

Definitions:

Perimeter:

A simple definition of the perimeter of a region without holes is the sequence of its interior border pixels. A pixel of a region is a border pixel if it has some neighbouring pixel that is outside the region. To compute the length of the perimeter ***P***, the pixels in the perimeter must be ordered in a sequence

$$P = \{(r_o, c_o), (r_1, c_1), \ldots\ldots, (r_{K-1}, c_{K-1})\} \qquad (20)$$

and each pair of successive pixels in the sequence being neighbours, including the first and last pixels. Then the perimeter is given by:

$$|P| = \#\{k \mid (r_{k+1}, c_{k+1}) \in N_4(r_k, c_k)\}$$
$$+\sqrt{2}\,\#\{k \mid (r_{k+1}, c_{k+1}) \in N_8(r_k, c_k) - N_4(r_{k+1}, c_{k+1})\} \qquad (21)$$

.

where ***k***+1 is the computed modulo ***K***.

4. Region Analysis and Properties

The length of the perimeter squared divided by the area is sometimes used as a measure of a shape's compactness or circularity.

For example, for circle: $$\frac{A}{P^2} = \frac{\pi r^2}{(2\pi r)^2} = \frac{1}{4\pi}$$

for square: $$\frac{A}{P^2} = \frac{a^2}{(4a)^2} = \frac{1}{16}$$

However, for digital shapes, the said quotient assumes its smallest value not for digital circles, as it would for continuous planar shapes.

But for digital octagons or diamonds, depending on whether the perimeter is computed as the number of its 4-neighbouring border pixels or as the length of its border, counting 1 for vertical or horizontal moves and √2 for diagonal moves.

4. Region Analysis and Properties

Alternatively, the ***Area*** and ***Position of the Centroids*** can be defined by:
Area is given by the zeroth moment:

$$A = \iint f(x,y)\, dx\, dy \qquad (22)$$

Position of the Centroid is given by the first moment:

$$\bar{x} = \frac{\iint x\, f(x,y)\, dx\, dy}{\iint f(x,y)\, dx\, dy}$$

(23)

$$\bar{y} = \frac{\iint y\, f(x,y)\, dx\, dy}{\iint f(x,y)\, dx\, dy}$$

4. Region Analysis and Properties

Distance from the Centroid to the shape boundary

The mean μ_R and the standard deviation σ_R^2 of the distance from the centroid to the shape boundary can be defined in terms of the pixels (r_k, c_k), ***k*** = 0, 1, ..., ***K -1*** in the perimeter ***P*** (Note that ***K*** is the number of pixels on the boundary).

$$\mu_R = \frac{1}{K}\sum_{k=0}^{K-1}\left\|(r_k, c_k) - (\bar{r}, \bar{c})\right\| \tag{24}$$

$$\sigma_R^2 = \frac{1}{K}\sum_{k=0}^{K-1}\left[\left\|(r_k, c_k) - (\bar{r}, \bar{c})\right\| - \mu_R\right]^2 \tag{25}$$

The ratio of $\frac{\mu_R}{\sigma_R}$ has the following properties:

1. As the digital shape becomes more circular, the ratio increases monotonically.
2. The value of the ratio for similar digital and continuous shapes are similar.
3. It is orientation and area independent.

4. Region Analysis and Properties

Distance from the Centroid to the shape boundary

Furthermore, the number ***N*** of sides to a regular digital polygon can be estimated from the circularity measure by the relation:

$$N = 1.4111\left(\frac{\mu_R}{\sigma_R}\right)^{0.4724} \tag{26}$$

We can determine for each region ***R***, its gray level mean and its gray level variance. The gray level mean is a first-order property. The gray level variance is a second-order property.

Average Gray level: $$\mu = \frac{1}{A}\sum_{(r,c)\in R} I(r,c) \tag{27}$$

Gray Level Variance: $$\sigma^2 = \frac{1}{A}\sum_{(r,c)\in R}\left[I(r,c) - \mu\right]^2 \tag{28}$$

5 Moments

The moment transform of an image function *f(x,y)* is given by

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y)\, dxdy \tag{29}$$

where $p,q = 0,1,2,....,\infty$

In the case of a spatially discretised M x N image, denoted *by f(i, j)*,

$$m_{pq} = \sum_{i=0}^{M} \sum_{j=0}^{N} i^p j^q f(i,j) \tag{30}$$

When we apply this approach to a region from a segmented binary image R, we have

$$m_{pq} = \sum_{i,j \in R} \sum i^p j^q \tag{31}$$

as $$f(i,j) = \begin{cases} 1 & \textbf{bright} \\ 0 & \textbf{dark} \end{cases}$$

Hence, all points *(i, j)* are either the boundary or interior points of the region R.

5 Moments

Let us consider the following moments.

$$m_{00} = \sum_{i,j \in R} \sum 1$$

$\Rightarrow$ Number of pixels in the region R.

$$m_{10} = \sum_{i \in R} i$$

$$m_{01} = \sum_{j \in R} j \tag{32}$$

Hence, m_{10} & m_{01} are just the sum of the indices in the discretized image.

The centroid is then given by

$$i_c = \frac{m_{10}}{m_{00}}$$

$$j_c = \frac{m_{01}}{m_{00}} \tag{33}$$

We shall now see an example to illustrate the procedures in determining the various properties.

5 Moments

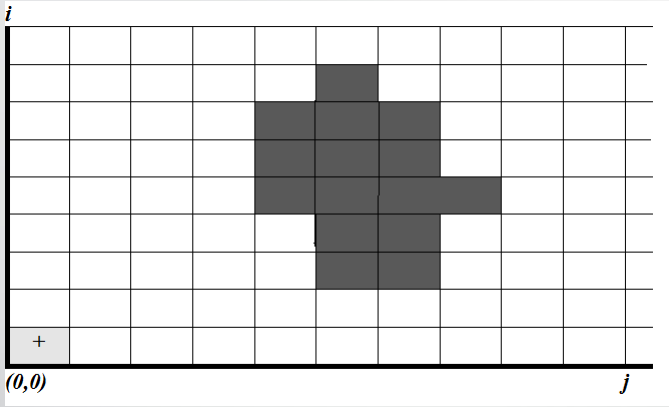


Fig. 22 Example of a digitized square

i	4	5	6	2	3	4	5	6	7
j	4	4	4	5	5	5	5	5	5
i	2	3	4	5	6	4			
j	6	6	6	6	6	7			

5 Moments

i	4	5	6	2	3	4	5	6	7
j	4	4	4	5	5	5	5	5	5
i	2	3	4	5	6	4			
j	6	6	6	6	6	7			

In the example for the square shown in Fig. 26.

$$m_{00} = 15$$

$$m_{10} = 4+5+6+2+3+4+5+6+ 7+2+3+4+5+6+4 = 66$$

$$m_{01} = 4x3+5x6+5x6+7 = 79$$

The centroid is at

$$i_c = \frac{66}{15} = 4.4$$

$$j_c = \frac{79}{15} = 5.3$$

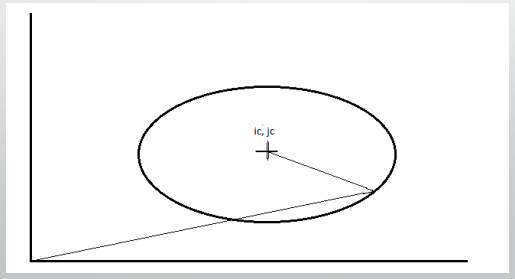
Although the image has been discretized, the value of the coordinates of the centroid are usually non-integers.

. Central Moments

The moments and the associated properties depends on the position of the origin of the image. To avoid this inconvenience, it is better to express the quantities with respect to the centroid of the object of interest.

Continuous case: $$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q \; f(x,y)dx\,dy \quad (34)$$

For a digital image $$\mu_{pq} = \sum_{i,j}\sum_{\in R} (i-i_c)^p \, (j-j_c)^q \; f(i,j) \quad (35)$$



6. Central Moments

The moments and the associated properties depends on the position of the origin of the image. To avoid this inconvenience, it is better to express the quantities with respect to the centroid of the object of interest.

Continuous case: $$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\bar{x})^p (y-\bar{y})^q \; f(x,y)dx\,dy \quad (36)$$

For a digital image $$\mu_{pq} = \sum_{i,j}\sum_{\in R} (i-i_c)^p \, (j-j_c)^q \; f(i,j) \quad (37)$$

For the following cases:

$$\mu_{00} = \sum_{i,j}\sum_{\in R} 1 = m_{00}$$

$$\mu_{10} = \sum_{i,j}\sum_{\in R} (i-i_c)\,(j-j_c)^o \; f(i,j)$$

$$= \sum_{i\in R} i \; f(i.j) - i_c \sum_{i\in R} f(i.j)$$

$$= m_{10} - \frac{m_{10}}{m_{00}} . m_{00}$$

$$= 0$$

Similarly,

6. Central Moments

Similarly

$$\mu_{01} = 0 \tag{38}$$

How about μ_{20} & μ_{02}?

$$\mu_{20} = \sum_{i \in R} (i - i_c)^2 = \sum_{i \in R} i^2 - 2i_c \sum_{i \in R} i + i_c^2 \sum_{i \in R} 1$$

$$\mu_{20} = m_{20} - 2 \cdot \frac{m_{10}}{m_{00}} \cdot m_{10} + m_{00} \cdot \frac{{m_{10}}^2}{{m_{00}}^2} \tag{39}$$

$$\mu_{20} = m_{20} - \frac{{m_{10}}^2}{m_{00}}$$

6. Central Moments

You may wish to show that.

Similarly,

$$\mu_{02} = m_{02} - \frac{m_{01}^2}{m_{00}}$$

$$\mu_{11} = m_{11} - \frac{m_{10} m_{01}}{m_{00}}$$

$$\mu_{30} = m_{30} - 3\, i_c\, m_{20} + 2\, i_c^2\, m_{10} \tag{40}$$

$$\mu_{03} = m_{03} - 3\, j_c\, m_{02} + 2\, j_c^2\, m_{01}$$

μ_{pq} is still sensitive to Rotation and Scale Transformation.

6. Scale Invariance

Scale invariance may be obtained with

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \quad \text{where} \quad \gamma = \frac{p+q}{2} + 1 \tag{41}$$

Let us now return to the example shown in Fig. 26 for a digitised square binary image. We will now calculate the following quantities:

m_{20} = 16 + 25 + 36 + 4 + 9 + 16 + 25 + 36 + 47 + 4 + 9 + 16 + 25 + 36 + 16
= 322

m_{02} = 3 x 16 + 6 x 25 + 5 x 36 + 49 = 427

μ_{20} = m_{20} - i_c m_{10} = 322 - 4.4 x 66 = 31.6

μ_{02} = m_{02} - j_c m_{01} = 427 - 5.3 x 79 = 8.3

:

$$\therefore \eta_{20} = \frac{\mu_{20}}{\mu_{00}^{\gamma}} \qquad \gamma = 2$$

$$= \frac{31.6}{15^2}$$

$$= 0.140$$

$$\eta_{02} = \frac{\mu_{02}}{\mu_{00}^{2}} = \frac{8.3}{15^2} = 0.037$$

$$\phi_1 = \eta_{20} + \eta_{02} = 0.177$$

6. Scale Invariance

Seven derived moment invariants

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

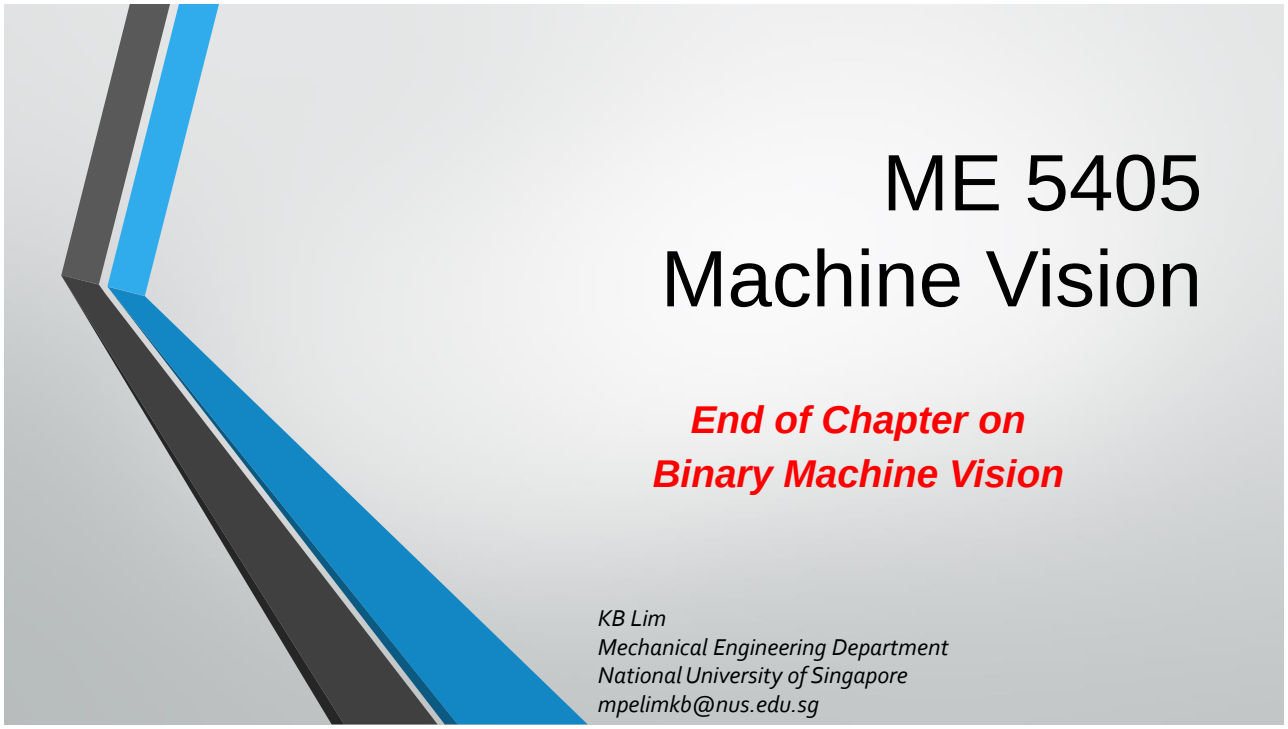
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

$$\phi_6 = (\eta_{20} - \eta_{02})\left[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$

(42)


ME 5405
Machine Vision
End of Chapter on
Binary Machine Vision
KB Lim
Mechanical Engineering Department
National University of Singapore
mpelimkb@nus.edu.sg