IMAGE REPRESENTATION
&
FEATURE EXTRACTION

Motivation

- One of the major concern of image processing is image (object) recognition
  - Objects are represented as a collection of pixels in an image
- Our Task: To describe the region based on the chosen representation

Image acquisition => digital image

↓

Preprocessing => better image

↓

Segmentation => basic features

↓

Representation and description => advanced features

↓

Object recognition

Representation

- Representation means that we make the object information **more accessible** *for computer-interpretation*
- Two types of representation
  - Using boundary (External characteristics)
  - Using pixels of region (Internal characteristics)

Description

- Description means that ***we quantify*** our representation of the object
- **Boundary Descriptors**
  - Geometrical descriptors : Diameter, perimeter, eccentricity, curvature
  - Shape Numbers
  - Fourier Descriptors
  - Statistical Moments
- **Regional Descriptors**
  - Geometrical descriptors: Area, compactness, Euler number
  - Texture
  - Moments of 2D Functions

Desirable properties of descriptors

- They should define a complete set
  - Two objects must have the same descriptors if and only if *they have the same shape .*
- They should be *invariant to* Rotation, Scaling and Translation (RST)
- They Should be a compact set
  - A descriptor should only contain information about what makes an object **unique**, or different from the other objects.
  - The quantity of information used to describe this characterization should be ***less than*** the information necessary to have a complete description of the object itself.
- They should be robust
  - *Work well against Noise and Distortion*
- They should have low computational complexity

Introduction

- The common **goal** of feature extraction and representation techniques is to *convert the segmented objects into representations that better describe their main features and attributes*.
- The type and complexity of the resulting representation depend on many factors, such as
  - the type of image (e.g., binary, gray-scale, or color),
  - the level of granularity (entire image or individual regions) desired
  - the context of the application that uses the results (e.g., a two-class pattern classifier that tells circular objects from noncircular ones or an image retrieval system that retrieves images judged to be similar to an example image).

Introduction

- "**Feature extraction** is the process by which certain features of interest within an image are detected and represented for further processing."
- It is a critical step in most computer vision and image processing solutions because it marks the transition from pictorial to non-pictorial (alphanumerical, usually quantitative) data representation.
- The resulting representation can be subsequently used as an input to a number of pattern recognition and classification techniques, which will then *label*, *classify*, or *recognize* the semantic contents of the image or its objects.

FEATURE VECTORS & VECTOR SPACES

- A ***feature vector*** is a $n \times 1$ array that encodes the n features (or measurements) of an image or object.
- The array **contents** may be
  - symbolic (e.g., a string containing the name of the predominant color in the image),
  - numerical (e.g., an integer expressing the area of an object, in pixels),
  - or both.
- Mathematically, a numerical feature vector **x** is given by

$$x = (x_1, x_2, \ldots, x_n)^T$$

  - where ***n*** is the total number of features and
  - ***T*** indicates the *transpose* operation.

FEATURE VECTORS & VECTOR SPACES

- The feature vector is a *compact representation of an image* (or object within the image), which can be associated with
  - the notion of a *feature space*,
  - an *n*-dimensional *hyperspace* that allows the visualization (for $n < 4$) and
  - interpretation of the feature vectors' contents, their relative distances, and so on.

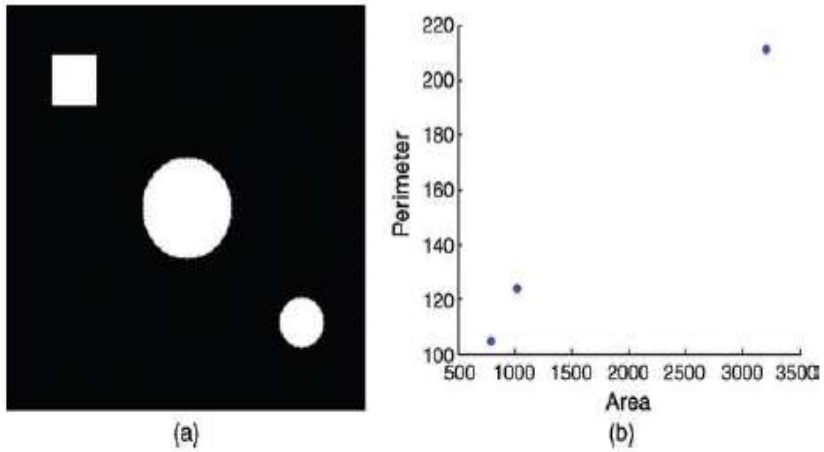


FIGURE 18.1 Test image (a) and resulting 2D feature vectors (b).

The resulting feature vectors will be as follows:

$\mathbf{Sq} = (1024, 124)^T$

$\mathbf{LC} = (3209, 211)^T$

$\mathbf{SC} = (797, 105)^T$

Figure 18.1b shows the three feature vectors plotted in a 2D graph whose axes are the selected features, namely, *area* and *perimeter*.

Invariance & Robustness

- A common requirement for feature extraction and representation techniques is that the features used to represent an image be invariant to rotation, scaling, and translation, collectively known as *RST*.
- RST invariance ensures that a machine vision system will still be able to recognize objects even when they appear at different size, position within the image, and angle (relative to a horizontal reference).

Binary Object Features

- A binary object is a connected region within a binary image $f(x, y)$, which will be denoted as O_i, $i > 0$.
- Mathematically, we can define a function $O_i(x, y)$ as follows:

$$O_i(x, y) = \begin{cases} 1 & if (f(x, y) \in O_i) \\ 0 & otherwise \end{cases}$$

- Area

BOUNDARY DESCRIPTORS

- In this section, we will look at ***contour-based*** representation and description techniques.
- These techniques assume that the contour (or *boundary*) of an object can be represented in a convenient coordinate system (Cartesian—the most common, polar, or tangential) and rely exclusively on boundary pixels to describe the region or object.
- Object boundaries can be represented by different techniques, ranging from simple polygonal approximation methods to more elaborated techniques involving piecewise polynomial interpolations such as B-spline curves.

BOUNDARY DESCRIPTORS

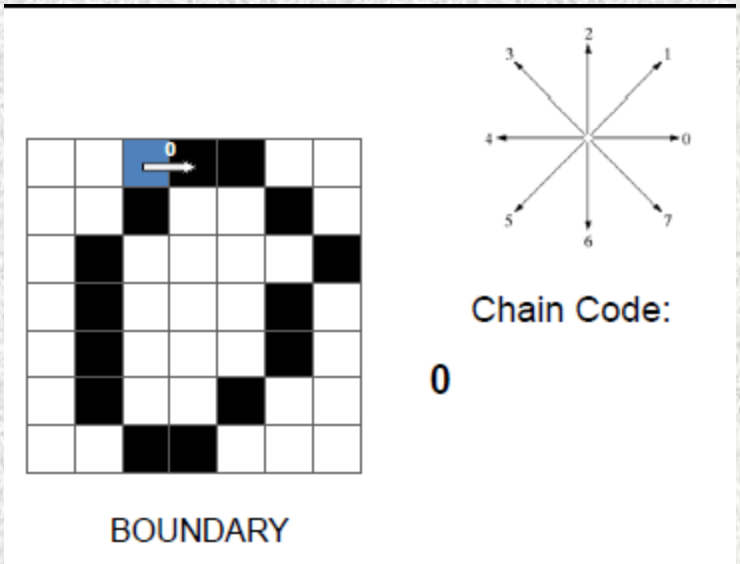
- The techniques described in this section **assume** that the pixels belonging to the boundary of the object (or region) can be ***traced***, starting from any background pixel, using an algorithm known as *bug tracing* that works as follows:
  - As soon as the conceptual bug crosses into a boundary pixel, it makes a left turn and moves to the next pixel; if that pixel is a boundary pixel, the bug makes another left turn, otherwise it turns right; the process is repeated until the bug is back to the starting point.
  - As the conceptual bug follows the contour, it *builds a list of coordinates* of the boundary pixels being visited.

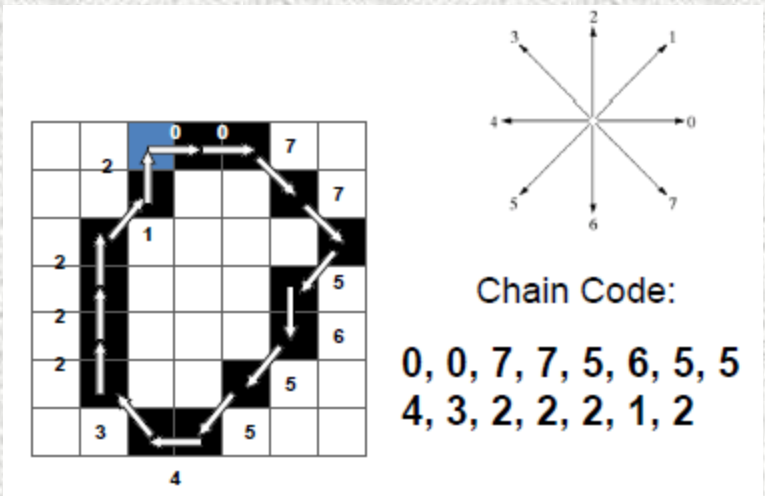
Chain Code, Freeman Code, & Shape Number

- Chain codes are alternative methods for *tracing* and *describing a* ***contour***.
- A chain code is a boundary representation technique by which "***A contour is represented as a sequence of straight line segments of specified length (usually 1) and direction***".
- The simplest chain code mechanism, also known as *crack code*, consists of assigning a number to the direction followed by a **bug tracking algorithm** as follows: right (0), down (1), left (2), and up (3).
- By allocating numbers based on directions, the boundary of an object is reduced to a sequence of numbers .

Chain Code

- Steps for construction chain codes
  - Select some starting point of the boundary and represent it by its absolute coordinates in the image
  - Represent every consecutive point by a chain code showing transition needed to go from current point to next point on the boundary
  - Stop if the next point is the *initial point* or the *end of the boundary*





Chain Code, Freeman Code, & Shape Number

- Assuming that the total number of boundary points is p (the perimeter of the contour), the array C (of size p), where $C(p) = 0, 1, 2, 3$, contains the chain code of the boundary.
- A modified version of the basic chain code, known as the ***Freeman code***, uses eight directions instead of four.
- Figure shows an example of a contour, its chain code, and its Freeman code.

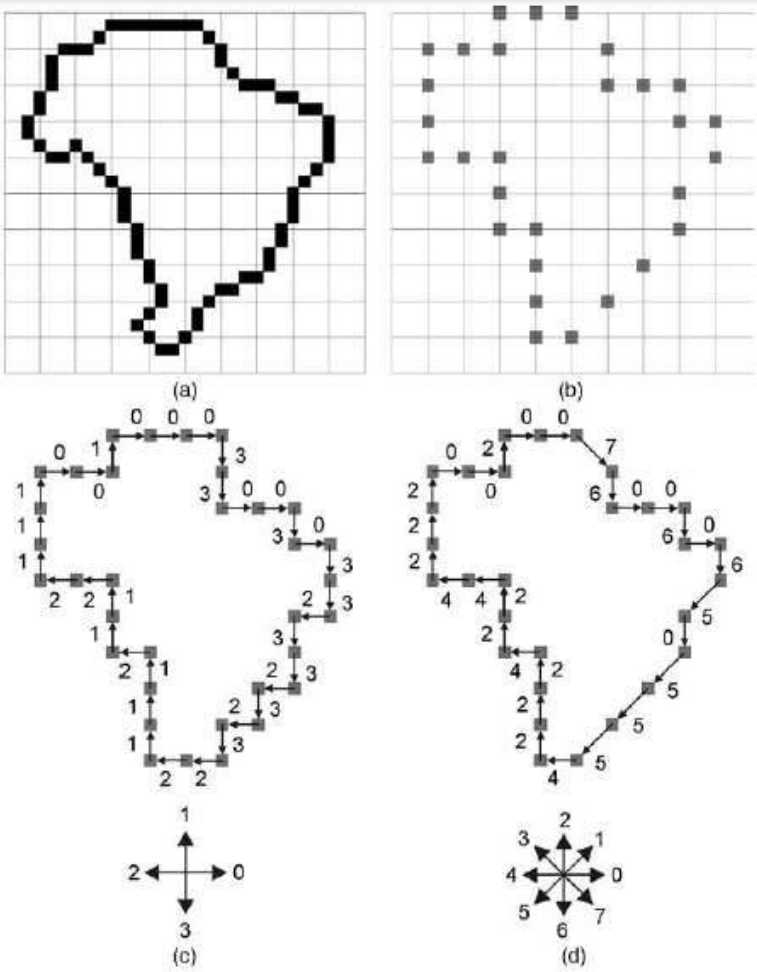


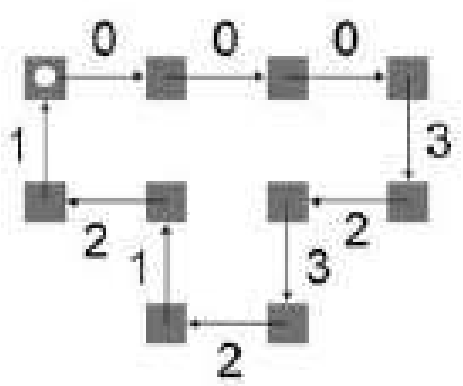
FIGURE 18.10 Chain code and Freeman code for a contour: (a) original contour; (b) subsampled version of the contour; (c) chain code representation; (d) Freeman code representation.

Chain Code, Freeman Code, & Shape Number

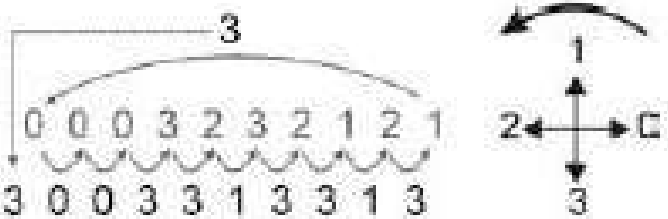
- Once the chain code for a boundary has been computed, it is possible to *convert* the resulting array into a Rotation-Invariant Equivalent, known as the ***first difference***.
- It is obtained by encoding the number of direction changes, expressed in multiples of 90◦ (according to a predefined convention, for example, counter clockwise), between two consecutive elements of the Freeman code.
- The ***first difference of Smallest magnitude*** is obtained by treating the resulting array as a circular array and rotating it cyclically until the resulting numerical pattern results in the smallest possible number is known as the ***Shape number*** of the contour.

Chain Code, Freeman Code, & Shape Number

- The shape number is Rotation invariant and Insensitive to the starting point used to compute the original sequence.
- Figure shows an example of a contour, its chain code, first differences, and shape number.



Chain code: 0 0 0 3 2 3 2 1 2 1



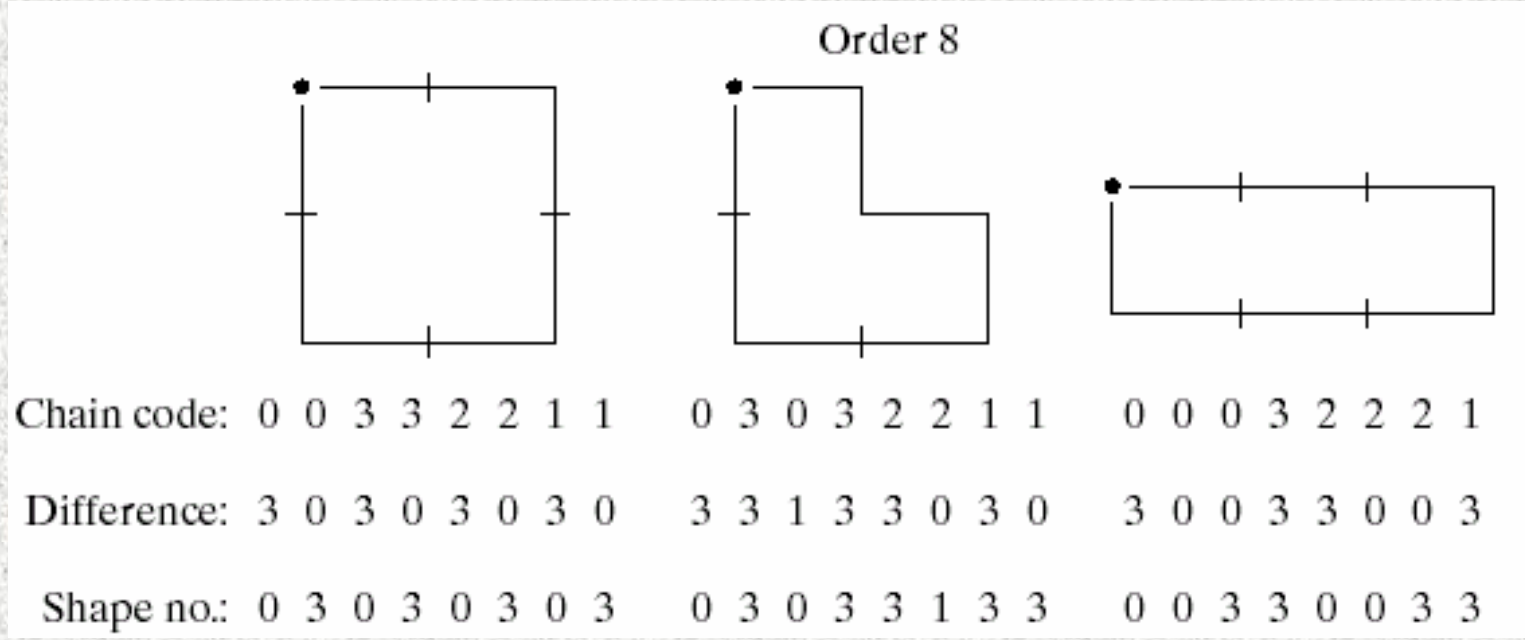
First difference: 3 0 0 3 3 1 3 3 1 3

Shape number: 0 0 3 3 1 3 3 1 3 3

FIGURE 18.11 Chain code, first differences, and shape number.

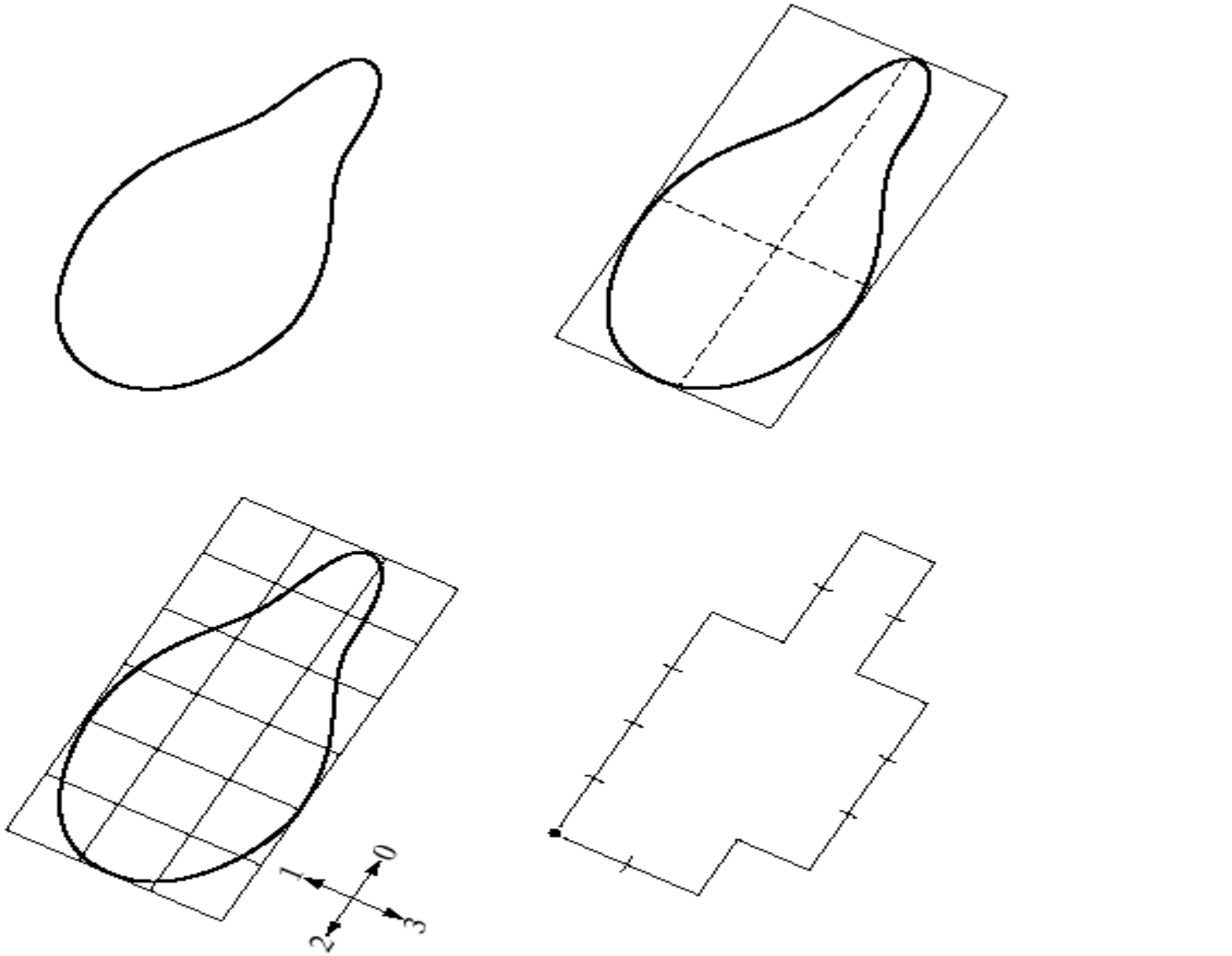
Shape Number

- The shape number of a boundary is defined as the first difference of smallest magnitude
- The order *n* of a shape number is defined as the number of digits in its representation



Algorithm for making a shape number

- Goal: To represent a given boundary by a shape number of order *n*
  - Step-1: Obtain the major axis of the shape and consider it as one of the coordinate axis
  - Step-2: Find the basic (smallest) rectangle that has sides parallel to major axis and just covers the shape
  - Step-3: From possible rectangles of order n, find one which best approximates rectangle of step-2
  - Step-4: Orient the rectangle, so that its major axis coincides with that of the shape
  - Step-5: Obtain the first difference chain code of minimum magnitude after circular shift

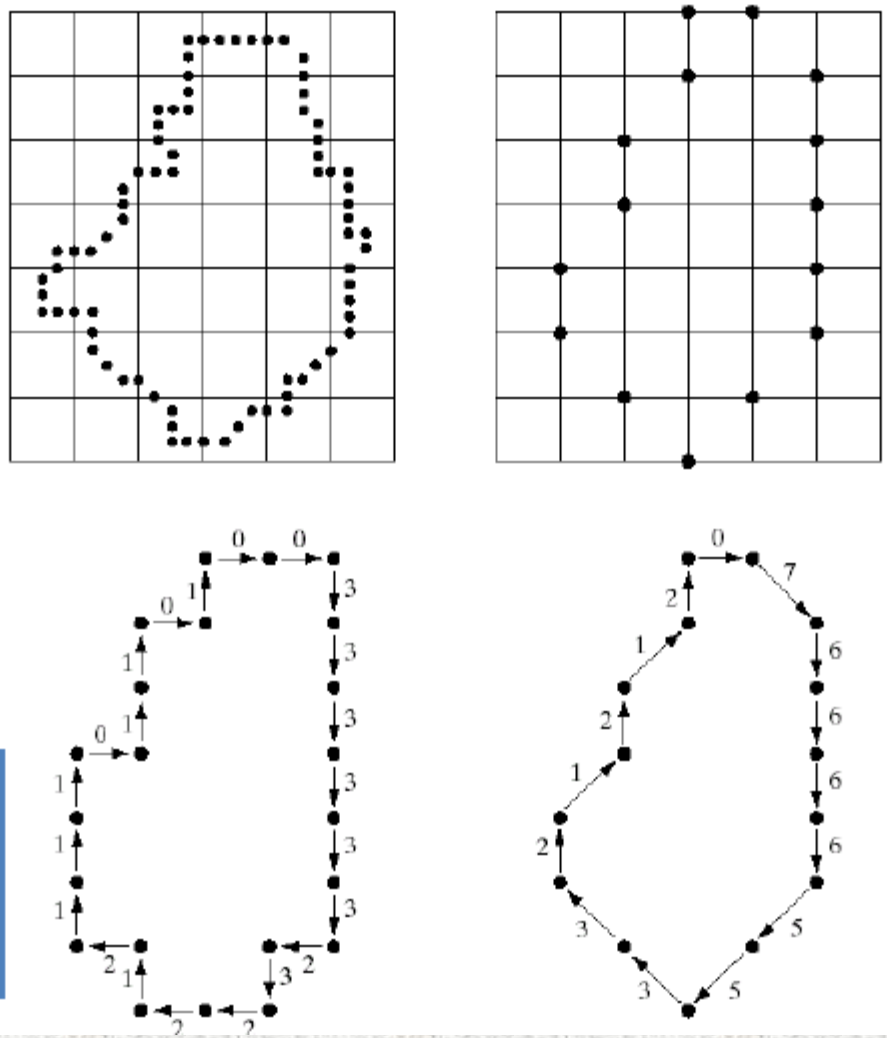


Chain code:	0	0	0	0	3	0	0	3	2	2	3	2	2	2	1	2	1	1
Difference:	3	0	0	0	3	1	0	3	3	0	1	3	0	0	3	1	3	0
Shape no.:	0	0	0	3	1	0	3	3	0	1	3	0	0	3	1	3	0	3

Chain Code

- **Advantages**
  - Preserves the information of interest
  - Provides good compression of boundary description
  - They are translation invariant
- **Problems**
  - Long chains of codes
  - No invariance to Rotation and Scale
  - Sensitive to Noise
- **Solution**
  - *Re-sample* the image to a lower resolution before calculating the code

Chain Codes



a b
c d

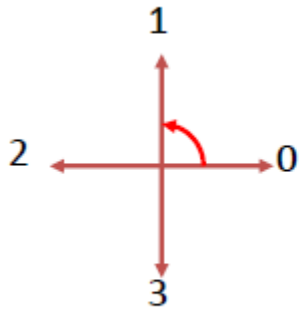
FIGURE 11.2
(a) Digital boundary with resampling grid superimposed. (b) Result of resampling. (c) 4-directional chain code. (d) 8-directional chain code.

Chain Code

- **Problem**
  - A chain code sequence *depends on a starting point.*
- **Solution**
  - Treat a chain code as a circular sequence and redefine the starting point so that the resulting sequence of numbers forms an integer of minimum magnitude after circular shift

2 2 3 0 -> 0 2 2 3

- The first difference of a chain code is counting the number of direction change (in counter clockwise) between 2 adjacent elements of the code



Chain code : The first difference

0 → 1	1
0 → 2	2
0 → 3	3
1 → 0	3
2 → 1	3
3 → 2	3

Example:

- a chain code: 10103322
- The first difference = 3133030
- Treating a chain code as a circular sequence, we get the first difference = 33133030

HISTOGRAM-BASED (STATISTICAL) FEATURES

Histogram-based features are also referred to as amplitude features

- Histograms provide a concise and useful representation of the intensity levels in a gray-scale image.
- The simplest histogram-based descriptor is the *mean* gray value of an image, representing its average intensity m and given by

$$m = \sum_{j=0}^{L-1} r_j p(r_j)$$

- where r_j is the jth gray level (out of a total of L possible values), whose probability of occurrence is $p(r_j)$.

HISTOGRAM-BASED (STATISTICAL) FEATURES

Histogram-based features are also referred to as amplitude features

1. The mean gray value can also be computed directly from the pixel values from the original image $f(x, y)$ of size $M \times N$ as follows:

$$m = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$$

- The *mean is a very compact descriptor* (one floating-point value per image or object) that provides a measure of the overall brightness of the corresponding image or object.
- *It is also RST invariant*.
- On the negative side, it has very limited Expressiveness and Discriminative power.

HISTOGRAM-BASED (STATISTICAL) FEATURES

Histogram-based features are also refered to as amplitude features

2. The *standard deviation* (as descriptor) σ of an image is given by

$$\sigma = \sqrt{\sum_{j=0}^{L-1} (r_j - m)^2 p(r_j)}$$

- where m is mean which define in previous slides.
- The square of the standard deviation is the *variance*, which is also known as the *normalized second-order moment* of the image.
- The standard deviation provides a concise representation of the overall contrast.
- Similar to the mean, it is compact and RST invariant, but has limited expressiveness and discriminative power.

HISTOGRAM-BASED (STATISTICAL) FEATURES

Histogram-based features are also referred to as amplitude features

3. The *skew* of a histogram is a measure of its asymmetry about the mean level. It is defined as

$$skew = \frac{1}{\sigma^3} \sum_{j=0}^{L-1} (r_j - m)^3 p(r_j)$$

- where σ is the standard deviation.
- The sign of the skew indicates whether the *histogram's tail spreads* to the right (positive skew) or to the left (negative skew).
- The skew is also known as the *normalized third-order moment* of the image.

HISTOGRAM-BASED (STATISTICAL) FEATURES

Histogram-based features are also referred to as amplitude features

- If the image's mean value (m), standard deviation (σ), and *mode* (defined as the histogram's highest peak) are known, the skew can be calculated as follows:

$$skew = \frac{m - mode}{\sigma}$$

4. The *energy* descriptor provides another measure of *how the pixel values are distributed along the gray-level range*: images with a single constant value have maximum energy (i.e., energy = 1); images with few gray levels will have higher energy than the ones with many gray levels. The energy descriptor can be calculated as

$$energy = \sum_{j=0}^{L-1} [p(r_j)]^2$$

HISTOGRAM-BASED (STATISTICAL) FEATURES

Histogram-based features are also referred to as amplitude features

5. Histograms also provide information about the complexity of the image, in the form of *entropy* descriptor.

- " The higher the entropy, the more complex the image".
- Entropy and energy tend to vary inversely with one another. The mathematical formulation for entropy is

$$entropy = -\sum_{j=0}^{L-1} p(r_j) \log_2[p(r_j)]$$

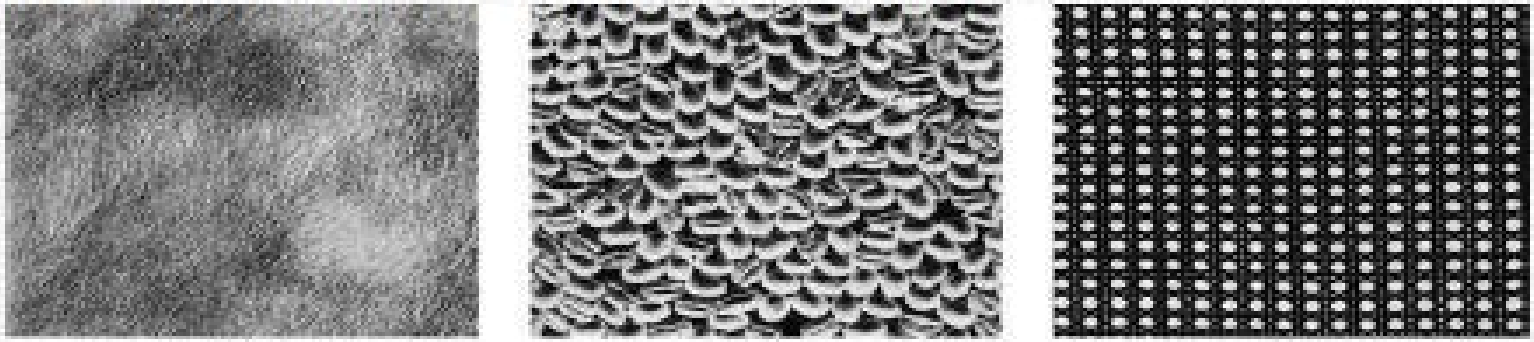
- Histogram-based features and their variants are usually employed as texture descriptors, as we shall see in next slide.

Texture Features

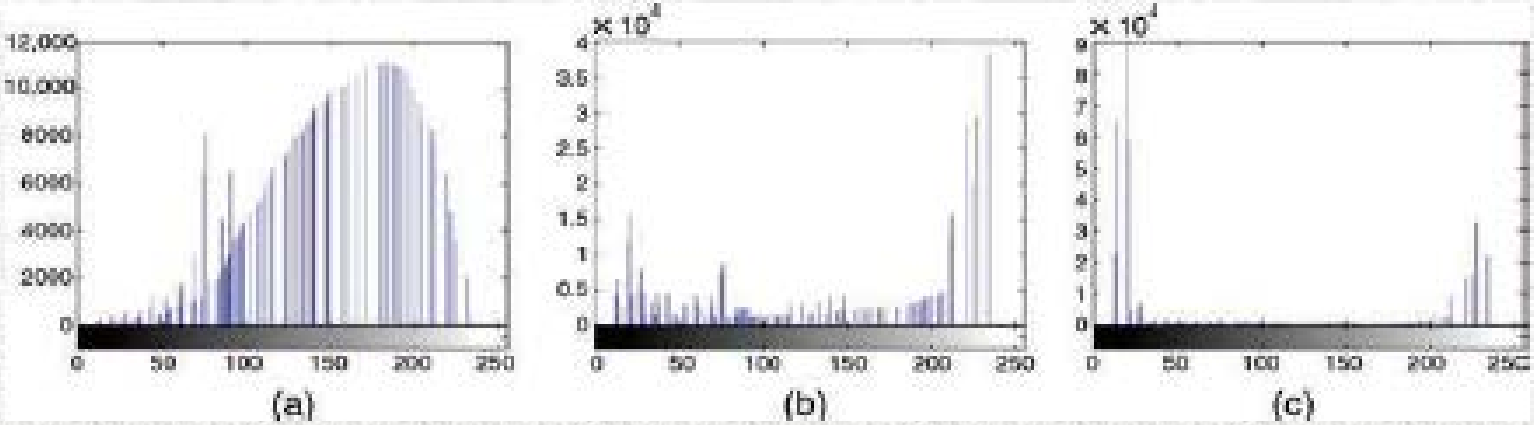
- Texture can be a powerful descriptor of an image (or one of its regions).
- Image processing techniques usually associate the notion of texture with image (or region) properties such as *Smoothness* (or its opposite, *roughness*), *Coarseness*, and *Regularity*.
- Figure 18.16 shows one example of each and Figure 18.17 shows their histograms.
- There are three main approaches to describe texture properties in image processing: Structural, Spectral, and Statistical.
- Most application focus on the statistical approaches, due to their popularity, usefulness and ease of computing.

Texture Features

- FIGURE 18.16 Example of images with smooth (a), coarse (b), and regular (c) texture. Images from the Brodatz textures data set.



- FIGURE 18.17 Histograms of images in Figure 18.16.



Texture Features

- **Highest uniformity has lowest entropy**

Texture	Mean	Standard deviation	Roughness R	Skew	Uniformity	Entropy
Smooth	147.1459	47.9172	0.0341	−0.4999	0.0190	5.9223
Coarse	138.8249	81.1479	0.0920	−1.9095	0.0306	5.8405
Regular	79.9275	89.7844	0.1103	10.0278	0.1100	4.1181

- Histogram-based texture descriptors are limited by the fact that the histogram does not carry any information about the spatial relationships among pixels. One way to circumvent this limitation consists in using an alternative representation for the pixel values that encodes their relative position with respect to one another.
- One such representation is the *gray-level co-occurrence matrix G, defined as a matrix whose* element *g(i, j) represents the number of times that pixel pairs with intensities zi and zj occur in image f (x, y) in the position specified by an operator d.*
- *The vector d is* known as *displacement vector:*

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

(a)

i \ j	0	1	2	3	4	5	6	7
0	0	2	0	0	0	0	0	1
1	0	0	0	0	0	1	0	1
2	1	0	0	0	0	0	1	0
3	1	1	1	0	0	2	2	0
4	0	0	0	0	0	0	0	1
5	0	1	1	1	1	2	0	0
6	0	0	0	4	0	0	0	0
7	1	0	0	0	0	2	1	1

(b)

Gray level co-occurrence Matrix

- The gray-level co-occurrence matrix can be normalized as follows:

$$N_g(i,j) = \frac{g(i,j)}{\sum_i \sum_j g(i,j)}$$

- where *Ng(i, j) is the normalized gray-level co-occurrence matrix. Since all values of Ng(i, j) lie between 0 and 1, they can be thought of as the probability that a pair of* points satisfying **d will have values** (*zi, zj*).
- Co-occurrence matrices can be used to represent the texture properties of an image.
- Instead of using the entire matrix, more compact descriptors are preferred.
- These are the most popular texture-based features that can be computed from a normalized gray-level co-occurrence matrix *Ng(i, j)*:

$$\text{Maximum probability} = \max_{i,j} N_g(i, j)$$

$$\text{Energy} = \sum_i \sum_j N_g^2(i, j)$$

$$\text{Entropy} = -\sum_i \sum_j N_g(i, j) \log_2 N_g(i, j)$$

$$\text{Contrast} = \sum_i \sum_j (i - j)^2 N_g(i, j)$$

$$\text{Homogeneity} = \sum_i \sum_j \frac{N_g(i, j)}{1 + |i - j|}$$

$$\text{Correlation} = \frac{\sum_i \sum_j (i-\mu_i)(j-\mu_j) N_g(i, j)}{\sigma_i \sigma_j}$$

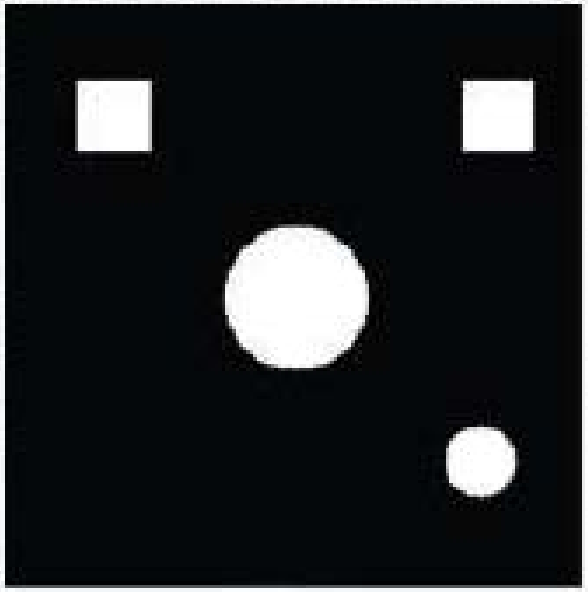
where μ_i, μ_j are the means and σ_i, σ_j are the standard deviations of the row and column sums $N_g(i)$ and $N_g(j)$, defined as

$$N_g(i) = \sum_i N_g(i, j) \tag{18.35}$$

$$N_g(j) = \sum_i N_g(i, j) \tag{18.36}$$

Example

- For the given binary image compute the descriptor values and fill the it in the given table.



(a)

Table for feature extraction results

Object	Area	Centroid (row, col)	Orientation (degrees)	Euler number	Eccentricity	Aspect ratio	Perimeter	Thiness ratio
Top left square								
Big circle								
Small circle								
Top right square								

Question 2 *Do the results obtained for the extracted features correspond to your* expectations? Explain.

Question 3 *Which of the extracted features have the best discriminative power to* help tell squares from circles? Explain.

Question 4 *Which of the extracted features have the worst discriminative power* to help tell squares from circles? Explain.

Question 5 *Which of the extracted features are ST invariant, that is, robust to* changes in size and translation? Explain.

Question 6 *If you had to use only one feature to distinguish squares from circles,* in a ST-invariant way, which feature would you use? Why?

Texture Features

- One of the simplest set of statistical features for texture description consists of the following histogram-based descriptors of the image (or region): **mean**, **variance** (or its square root, the standard deviation), **skew**, **energy** (used as a measure of *uniformity*), and **entropy**, all of which were introduced in Section 18.5.
- The variance is sometimes used as a normalized descriptor of roughness (R), defined as

$$R = 1 - \frac{1}{1 + \sigma^2}$$

- Where, σ^2 is the normalized (to a [0, 1] interval) variance.
- $R = 0$ for areas of constant intensity, that is, smooth texture.