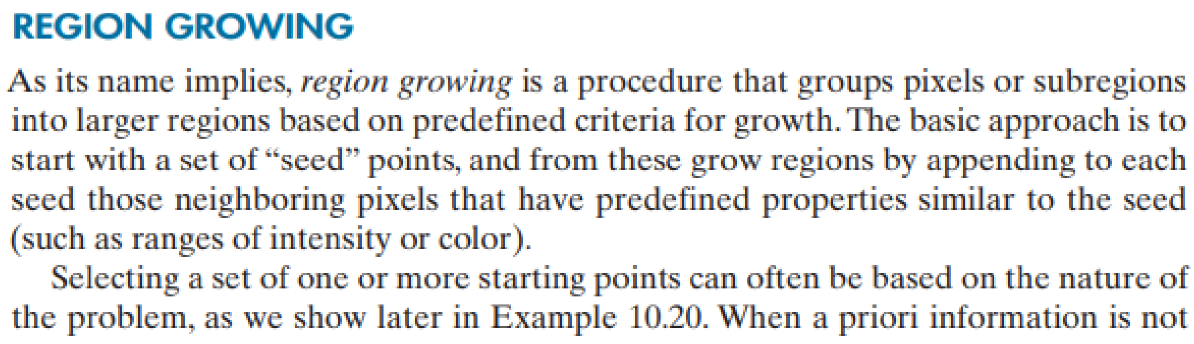
<u>Q1. Explain the region based approaches to image segmentation?</u>

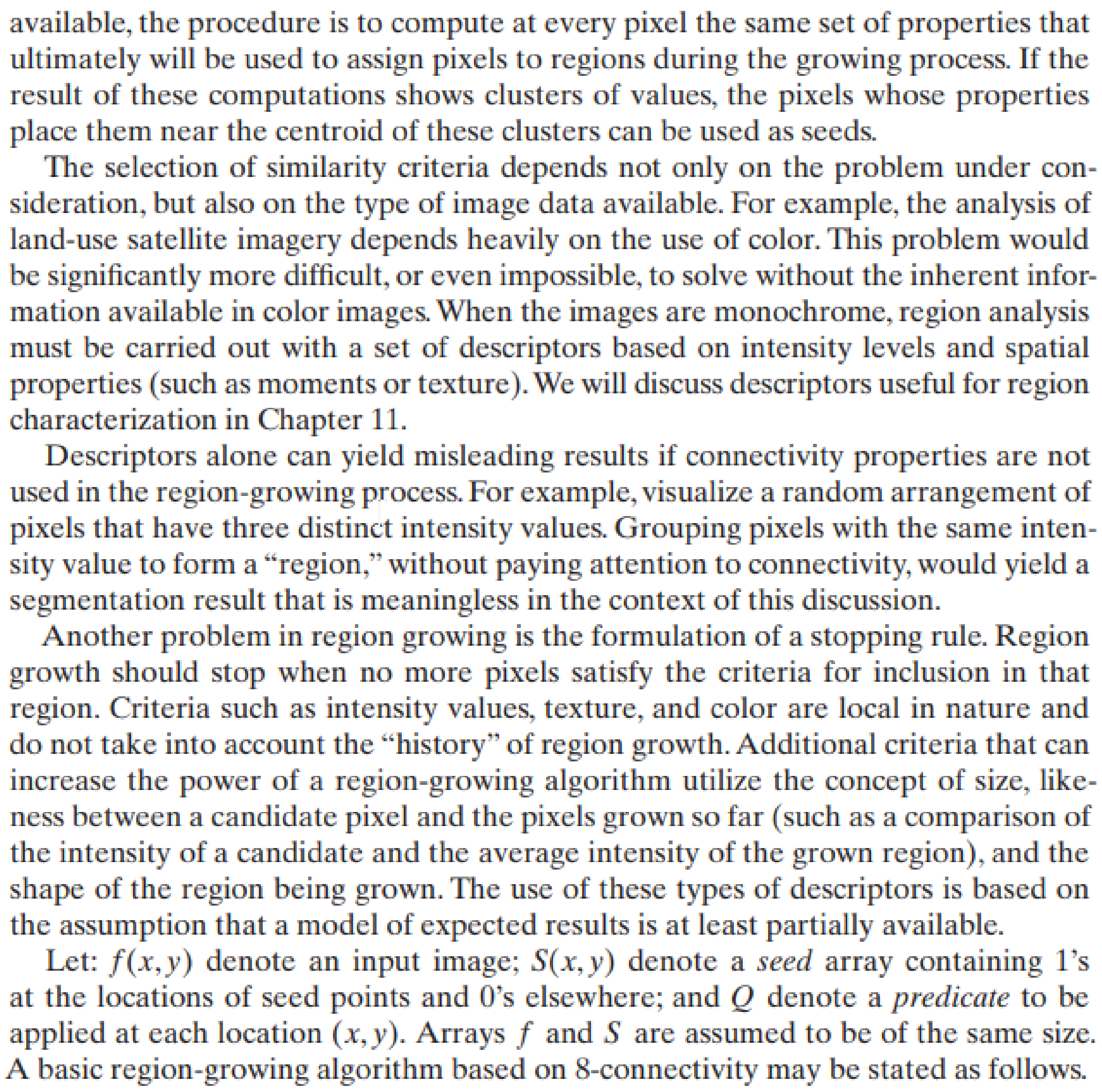
Ans:



REGION GROWING

As its name implies, *region growing* is a procedure that groups pixels or subregions into larger regions based on predefined criteria for growth. The basic approach is to start with a set of "seed" points, and from these grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed (such as ranges of intensity or color).

Selecting a set of one or more starting points can often be based on the nature of the problem, as we show later in Example 10.20. When a priori information is not



available, the procedure is to compute at every pixel the same set of properties that ultimately will be used to assign pixels to regions during the growing process. If the result of these computations shows clusters of values, the pixels whose properties place them near the centroid of these clusters can be used as seeds.

The selection of similarity criteria depends not only on the problem under consideration, but also on the type of image data available. For example, the analysis of land-use satellite imagery depends heavily on the use of color. This problem would be significantly more difficult, or even impossible, to solve without the inherent information available in color images. When the images are monochrome, region analysis must be carried out with a set of descriptors based on intensity levels and spatial properties (such as moments or texture). We will discuss descriptors useful for region characterization in Chapter 11.

Descriptors alone can yield misleading results if connectivity properties are not used in the region-growing process. For example, visualize a random arrangement of pixels that have three distinct intensity values. Grouping pixels with the same intensity value to form a "region," without paying attention to connectivity, would yield a segmentation result that is meaningless in the context of this discussion.

Another problem in region growing is the formulation of a stopping rule. Region growth should stop when no more pixels satisfy the criteria for inclusion in that region. Criteria such as intensity values, texture, and color are local in nature and do not take into account the "history" of region growth. Additional criteria that can increase the power of a region-growing algorithm utilize the concept of size, likeness between a candidate pixel and the pixels grown so far (such as a comparison of the intensity of a candidate and the average intensity of the grown region), and the shape of the region being grown. The use of these types of descriptors is based on the assumption that a model of expected results is at least partially available.

Let: $f(x, y)$ denote an input image; $S(x, y)$ denote a *seed* array containing 1's at the locations of seed points and 0's elsewhere; and Q denote a *predicate* to be applied at each location (x, y). Arrays f and S are assumed to be of the same size. A basic region-growing algorithm based on 8-connectivity may be stated as follows.

1. Find all connected components in $S(x, y)$ and reduce each connected component to one pixel; label all such pixels found as 1. All other pixels in S are labeled 0.
2. Form an image f_Q such that, at each point (x, y), $f_Q(x, y) = 1$ if the input image satisfies a given predicate, Q, at those coordinates, and $f_Q(x, y) = 0$ otherwise.
3. Let g be an image formed by appending to each seed point in S all the 1-valued points in f_Q that are 8-connected to that seed point.
4. Label each connected component in g with a different region label (e.g.,integers or letters). This is the segmented image obtained by region growing.

REGION SPLITTING AND MERGING

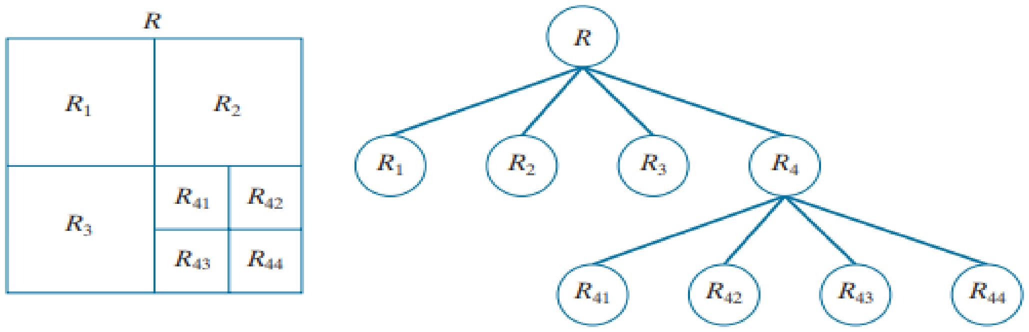
The procedure just discussed grows regions from seed points. An alternative is to subdivide an image initially into a set of disjoint regions and then merge and/or split the regions in an attempt to satisfy the conditions of segmentation stated in Section 10.1. The basics of region splitting and merging are discussed next.

Let R represent the entire image region and select a predicate Q. One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i, $Q(R_i) = \text{TRUE}$. We start with the entire region, R. If $Q(R) = \text{FALSE}$, we divide the image into quadrants. If Q is FALSE for any quadrant, we subdivide that quadrant into sub-quadrants, and so on. This splitting technique has a convenient representation in the form of so-called *quadtrees*; that is, trees in which each node has exactly four descendants, as Fig. 10.47 shows (the images corresponding to the nodes of a quadtree sometimes are called *quadregions* or *quadimages*). Note that the root of the tree corresponds to the entire image, and that each node corresponds to the subdivision of a node into four descendant nodes. In this case, only R_4 was subdivided further.

If only splitting is used, the final partition normally contains adjacent regions with identical properties. This drawback can be remedied by allowing *merging* as well as splitting. Satisfying the constraints of segmentation outlined in Section 10.1 requires merging only adjacent regions whose combined pixels satisfy the predicate Q. That is, two adjacent regions R_j and R_k are merged only if $Q(R_j \cup R_k) = \text{TRUE}$.

The preceding discussion can be summarized by the following procedure in which, at any step, we

1. Split into four disjoint quadrants any region R_i for which $Q(R_i) = \text{FALSE}$.
2. When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cup R_k) = \text{TRUE}$.



3. Stop when no further merging is possible.

Numerous variations of this basic theme are possible. For example, a significant simplification results if in Step 2 we allow merging of any two adjacent regions R_j and R_k if each one satisfies the predicate individually. This results in a much simpler (and faster) algorithm, because testing of the predicate is limited to individual quadregions. As the following example shows, this simplification is still capable of yielding good segmentation results.

Q2. Define Image Enhancement. Explain the following Enhancement operations and draw the graphs if transformation function:

a. Bit plane slicing

b. Grey level slicing

Ans:

Image enhancement is the process of manipulating an image so the result is more suitable than the original for a specific application. The word *specific* is important here, because it establishes at the outset that enhancement techniques are problem oriented. Thus, for example, a method that is quite useful for enhancing X-ray images may not be the best approach for enhancing satellite images taken in the infrared band of the electromagnetic spectrum.

Bit-Plane Slicing

Pixel values are integers composed of bits. For example, values in a 256-level gray-scale image are composed of 8 bits (one byte). Instead of highlighting intensity-level ranges, as 3.3, we could highlight the contribution made to total image appearance by specific bits. As Fig. 3.13 illustrates, an 8-bit image may be considered as being composed of eight one-bit planes, with plane 1 containing the lowest-order bit of all pixels in the image, and plane 8 all the highest-order bits.

Figure 3.14(a) shows an 8-bit grayscale image and Figs. 3.14(b) through (i) are its eight one-bit planes, with Fig. 3.14(b) corresponding to the highest-order bit. Observe that the four higher-order bit planes, especially the first two, contain a significant amount of the visually-significant data. The lower-order planes contribute to more subtle intensity details in the image. The original image has a gray border whose intensity is 194. Notice that the corresponding borders of some of the bit

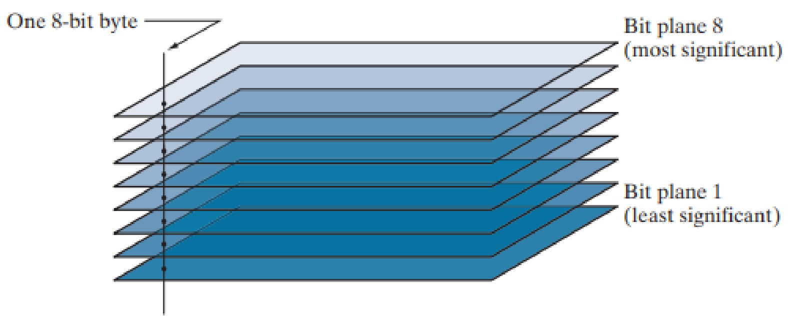


FIGURE 3.13 Bit-planes of an 8-bit image.

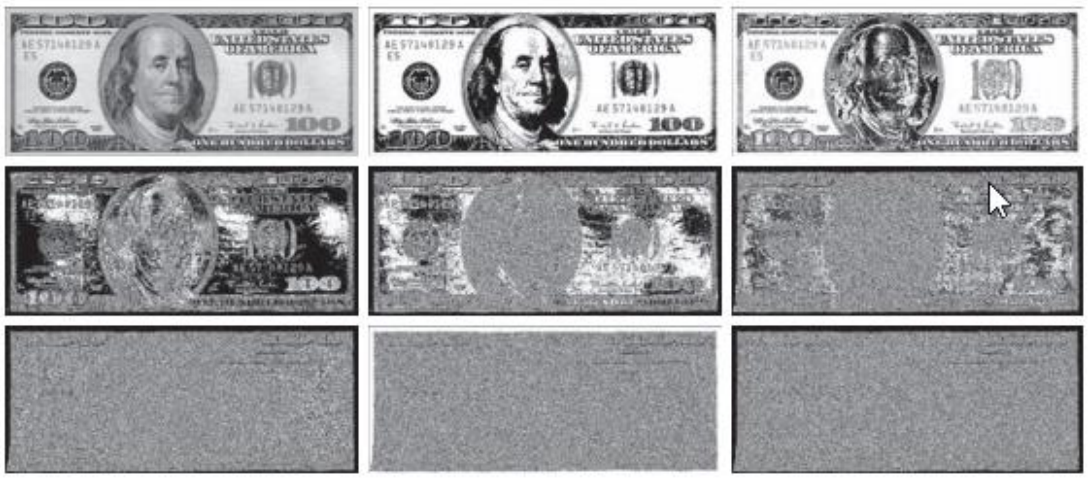
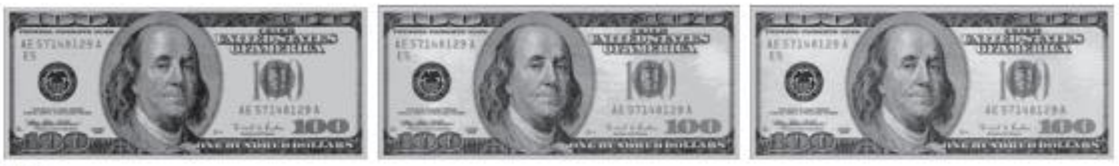

a b c
d e f
g h i

FIGURE 3.14 (a) An 8-bit gray-scale image of size 550 × 1192 pixels. (b) through (i) Bit planes 8 through 1, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image..

planes are black (0), while others are white (1). To see why, consider a pixel in, say, the middle of the lower border of Fig. 3.14(a). The corresponding pixels in the bit planes, starting with the highest-order plane, have values 1 1 0 0 0 0 1 0, which is the binary representation of decimal 194. The value of any pixel in the original image can be similarly reconstructed from its corresponding binary-valued pixels in the bit planes by converting an 8-bit binary sequence to decimal.

The binary image for the 8th bit plane of an 8-bit image can be obtained by thresholding the input image with a transformation function that maps to 0 intensity values between 0 and 127, and maps to 1 values between 128 and 255. The binary image in Fig. 3.14(b) was obtained in this manner. It is left as an exercise (see Problem 3.3) to obtain the transformation functions for generating the other bit planes.

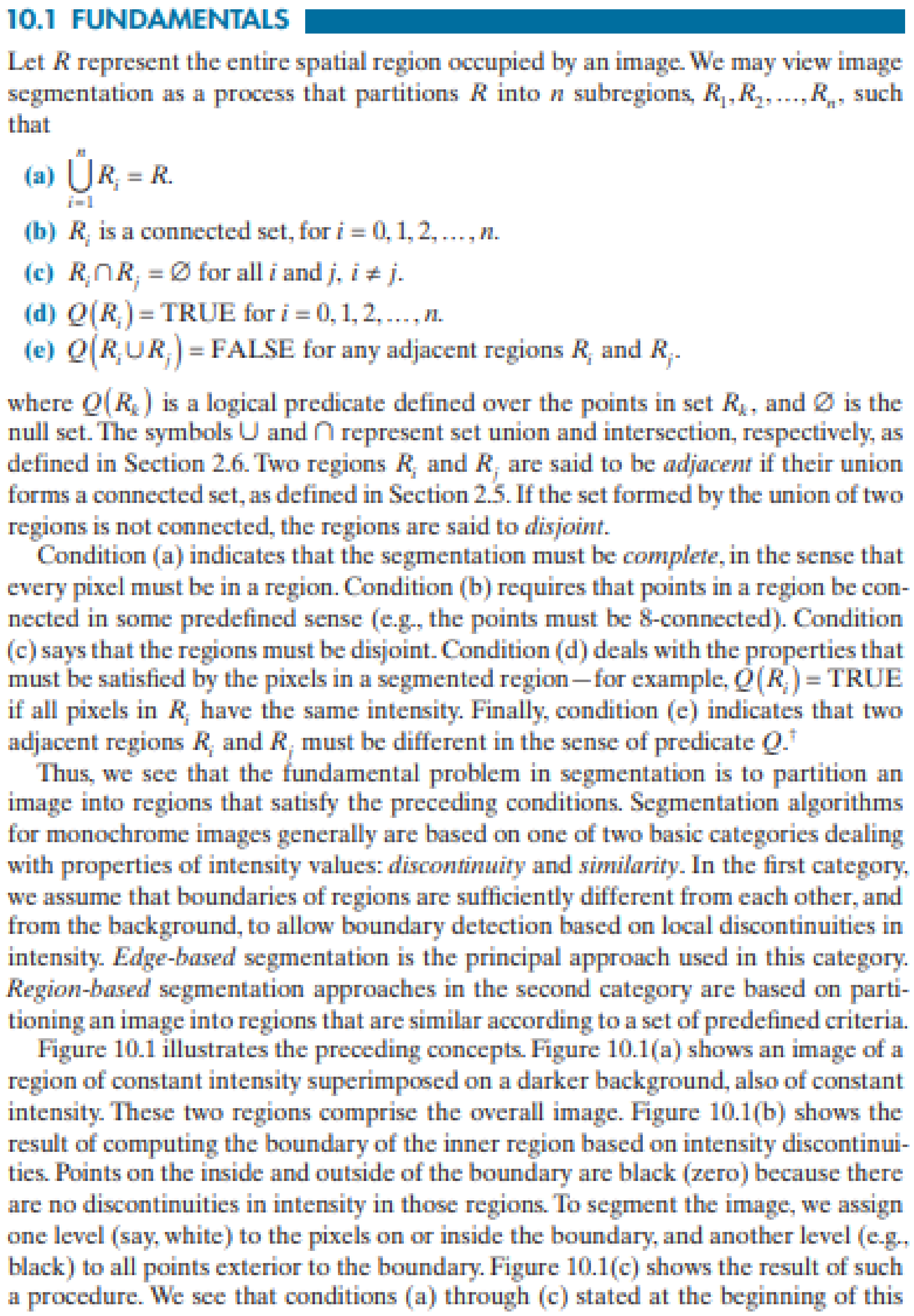
Decomposing an image into its bit planes is useful for analyzing the relative importance of each bit in the image, a process that aids in determining the adequacy of the number of bits used to quantize the image. Also, this type of decomposition is useful for image compression (the topic of Chapter 8), in which fewer than all planes are used in reconstructing an image. For example, Fig. 3.15(a) shows an image reconstructed using bit planes 8 and 7 of the preceding decomposition. The reconstruction is done by multiplying the pixels of the nth plane by the constant 2^{n-1}. This converts the nth significant binary bit to decimal. Each bit plane is multiplied by the corresponding constant, and all resulting planes are added to obtain the grayscale image. Thus, to obtain Fig. 3.15(a), we multiplied bit plane 8 by 128, bit plane 7 by 64, and added the two planes. Although the main features of the original image were restored, the reconstructed image appears flat, especially in the background. This


a b c **FIGURE 3.15** Image reconstructed from bit planes: (a) 8 and 7; (b) 8, 7, and 6; (c) 8, 7, 6, and 5.

is not surprising, because two planes can produce only four distinct intensity levels. Adding plane 6 to the reconstruction helped the situation, as Fig. 3.15(b) shows. Note that the background of this image has perceptible false contouring. This effect is reduced significantly by adding the 5th plane to the reconstruction, as Fig. 3.15(c) illustrates. Using more planes in the reconstruction would not contribute significantly to the appearance of this image. Thus, we conclude that, in this example, storing the four highest-order bit planes would allow us to reconstruct the original image in acceptable detail. Storing these four planes instead of the original image requires 50% less storage.

<u>Q3. What is meant by image segmentations? Give two applications of image segmentation</u>

Ans:



10.1 FUNDAMENTALS

Let R represent the entire spatial region occupied by an image. We may view image segmentation as a process that partitions R into n subregions, $R_1, R_2, \ldots, R_n$, such that

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

(b) R_i is a connected set, for $i = 0, 1, 2, \ldots, n$.

(c) $R_i \cap R_j = \varnothing$ for all i and j, $i \neq j$.

(d) $Q(R_i) = \text{TRUE}$ for $i = 0, 1, 2, \ldots, n$.

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

where $Q(R_k)$ is a logical predicate defined over the points in set R_k, and $\varnothing$ is the null set. The symbols $\cup$ and $\cap$ represent set union and intersection, respectively, as defined in Section 2.6. Two regions R_i and R_j are said to be *adjacent* if their union forms a connected set, as defined in Section 2.5. If the set formed by the union of two regions is not connected, the regions are said to *disjoint*.

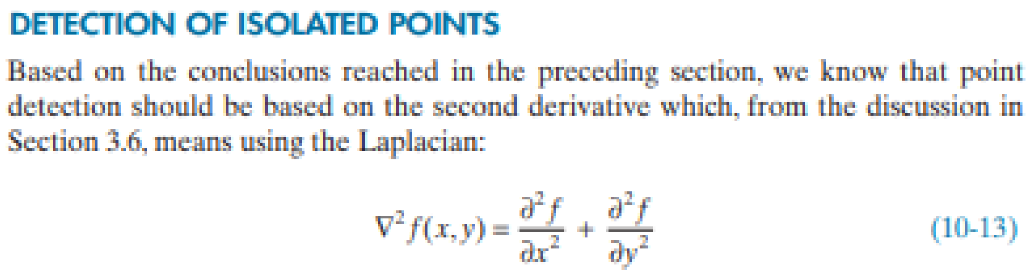
Condition (a) indicates that the segmentation must be *complete*, in the sense that every pixel must be in a region. Condition (b) requires that points in a region be connected in some predefined sense (e.g., the points must be 8-connected). Condition (c) says that the regions must be disjoint. Condition (d) deals with the properties that must be satisfied by the pixels in a segmented region—for example, $Q(R_i) = \text{TRUE}$ if all pixels in R_i have the same intensity. Finally, condition (e) indicates that two adjacent regions R_i and R_j must be different in the sense of predicate Q.[†]

Thus, we see that the fundamental problem in segmentation is to partition an image into regions that satisfy the preceding conditions. Segmentation algorithms for monochrome images generally are based on one of two basic categories dealing with properties of intensity values: *discontinuity* and *similarity*. In the first category, we assume that boundaries of regions are sufficiently different from each other, and from the background, to allow boundary detection based on local discontinuities in intensity. *Edge-based* segmentation is the principal approach used in this category. *Region-based* segmentation approaches in the second category are based on partitioning an image into regions that are similar according to a set of predefined criteria.

Figure 10.1 illustrates the preceding concepts. Figure 10.1(a) shows an image of a region of constant intensity superimposed on a darker background, also of constant intensity. These two regions comprise the overall image. Figure 10.1(b) shows the result of computing the boundary of the inner region based on intensity discontinuities. Points on the inside and outside of the boundary are black (zero) because there are no discontinuities in intensity in those regions. To segment the image, we assign one level (say, white) to the pixels on or inside the boundary, and another level (e.g., black) to all points exterior to the boundary. Figure 10.1(c) shows the result of such a procedure. We see that conditions (a) through (c) stated at the beginning of this

Q4. Explain the detection of isolated points in an image.

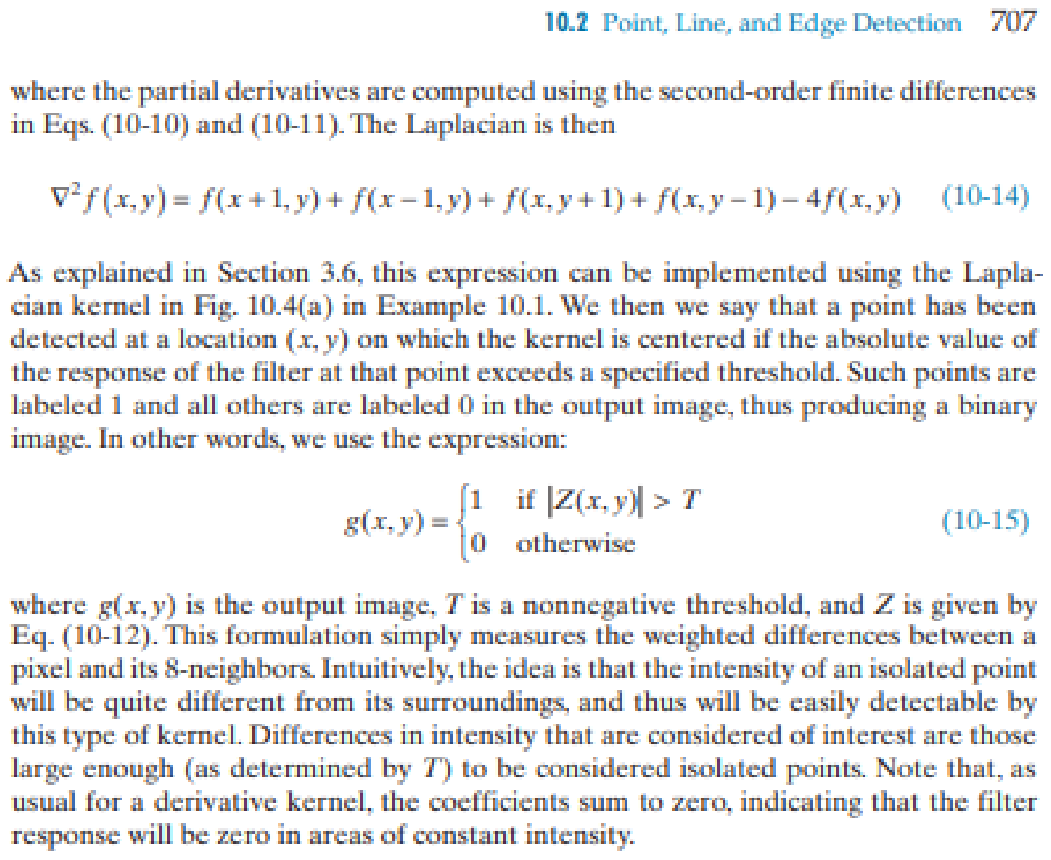
Ans:



DETECTION OF ISOLATED POINTS

Based on the conclusions reached in the preceding section, we know that point detection should be based on the second derivative which, from the discussion in Section 3.6, means using the Laplacian:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (10\text{-}13)$$



where the partial derivatives are computed using the second-order finite differences in Eqs. (10-10) and (10-11). The Laplacian is then

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \qquad (10\text{-}14)$$

As explained in Section 3.6, this expression can be implemented using the Laplacian kernel in Fig. 10.4(a) in Example 10.1. We then we say that a point has been detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point exceeds a specified threshold. Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image. In other words, we use the expression:

$$g(x,y) = \begin{cases} 1 & \text{if } |Z(x,y)| > T \\ 0 & \text{otherwise} \end{cases} \qquad (10\text{-}15)$$

where $g(x, y)$ is the output image, T is a nonnegative threshold, and Z is given by Eq. (10-12). This formulation simply measures the weighted differences between a pixel and its 8-neighbors. Intuitively, the idea is that the intensity of an isolated point will be quite different from its surroundings, and thus will be easily detectable by this type of kernel. Differences in intensity that are considered of interest are those large enough (as determined by T) to be considered isolated points. Note that, as usual for a derivative kernel, the coefficients sum to zero, indicating that the filter response will be zero in areas of constant intensity.

Q5. Explain the basics of intensity thresholding in image segmentation.

Ans:

The Basics of Intensity Thresholding

Suppose that the intensity histogram in Fig. 10.32(a) corresponds to an image, $f(x, y)$, composed of light objects on a dark background, in such a way that object and background pixels have intensity values grouped into two dominant modes. One obvious way to extract the objects from the background is to select a threshold, T, that separates these modes. Then, any point (x, y) in the image at which $f(x, y) > T$ is called an *object point*. Otherwise, the point is called a *background* point. In other words, the segmented image, denoted by $g(x, y)$, is given by

Remember, $f(x,y)$ denotes the intensity of f at coordinates (x,y).

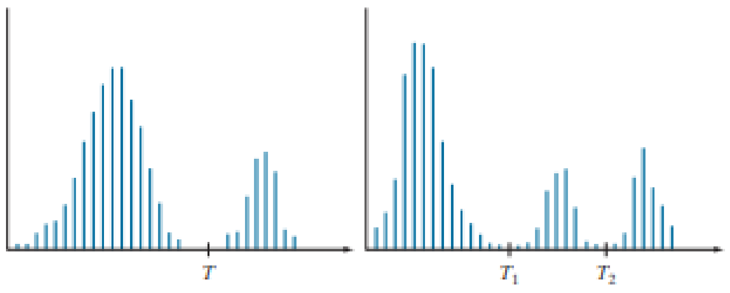
Although we follow convention in using 0 intensity for the background and 1 for object pixels, any two distinct values can be used in Eq. (10-46).

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) \le T \end{cases} \tag{10-46}$$

When T is a constant applicable over an entire image, the process given in this equation is referred to as *global thresholding*. When the value of T changes over an image, we use the term *variable thresholding*. The terms *local* or *regional* thresholding are used sometimes to denote variable thresholding in which the value of T at any point (x, y) in an image depends on properties of a neighborhood of (x, y) (for example, the average intensity of the pixels in the neighborhood). If T depends on the spatial coordinates (x, y) themselves, then variable thresholding is often referred to as *dynamic* or *adaptive* thresholding. Use of these terms is not universal.

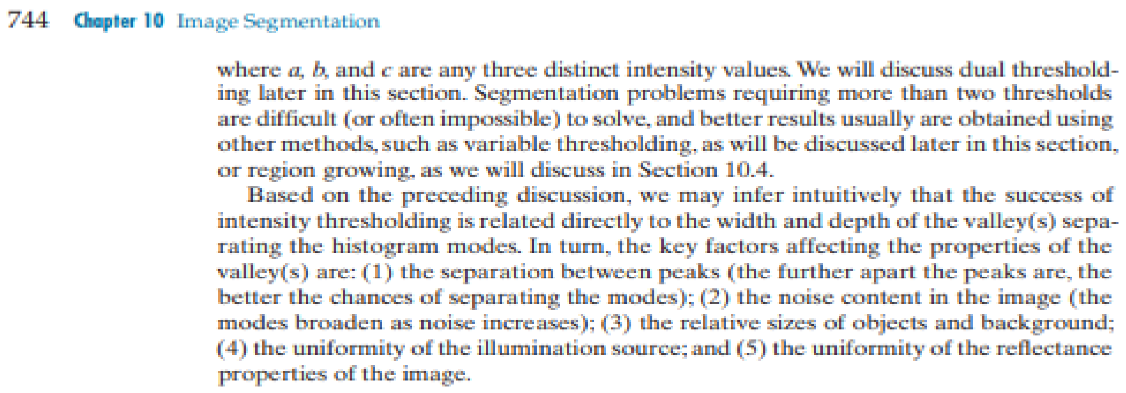
Figure 10.32(b) shows a more difficult thresholding problem involving a histogram with three dominant modes corresponding, for example, to two types of light objects on a dark background. Here, *multiple thresholding* classifies a point (x, y) as belonging to the background if $f(x,y) \le T_1$, to one object class if $T_1 < f(x,y) \le T_2$, and to the other object class if $f(x,y) > T_2$. That is, the segmented image is given by

$$g(x,y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \le T_2 \\ c & \text{if } f(x,y) \le T_1 \end{cases} \tag{10-47}$$



a b

FIGURE 10.32 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.



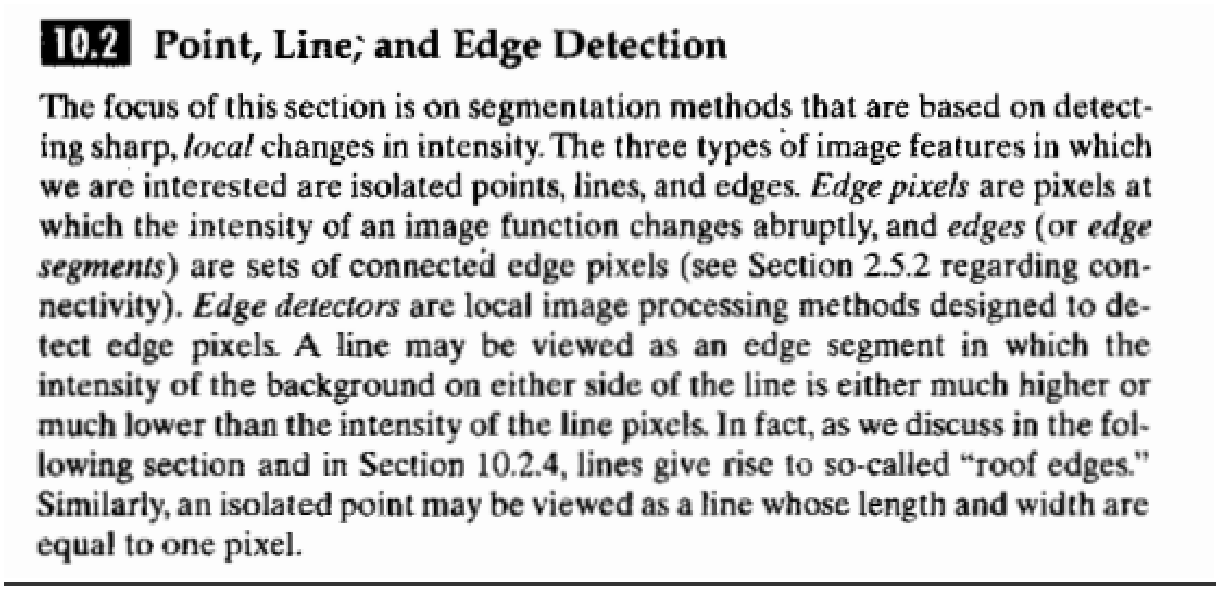
744 Chapter 10 Image Segmentation

where a, b, and c are any three distinct intensity values. We will discuss dual thresholding later in this section. Segmentation problems requiring more than two thresholds are difficult (or often impossible) to solve, and better results usually are obtained using other methods, such as variable thresholding, as will be discussed later in this section, or region growing, as we will discuss in Section 10.4.

Based on the preceding discussion, we may infer intuitively that the success of intensity thresholding is related directly to the width and depth of the valley(s) separating the histogram modes. In turn, the key factors affecting the properties of the valley(s) are: (1) the separation between peaks (the further apart the peaks are, the better the chances of separating the modes); (2) the noise content in the image (the modes broaden as noise increases); (3) the relative sizes of objects and background; (4) the uniformity of the illumination source; and (5) the uniformity of the reflectance properties of the image.

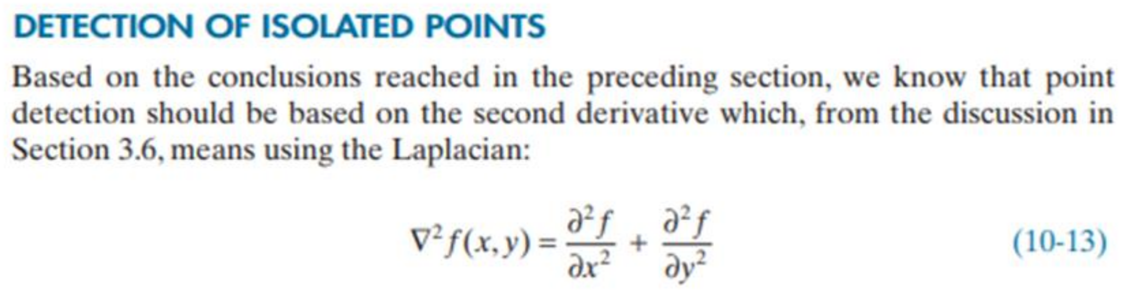
Q6. Explain point detection , edge detection , line detection in image segmentation.

Ans:



10.2 Point, Line; and Edge Detection

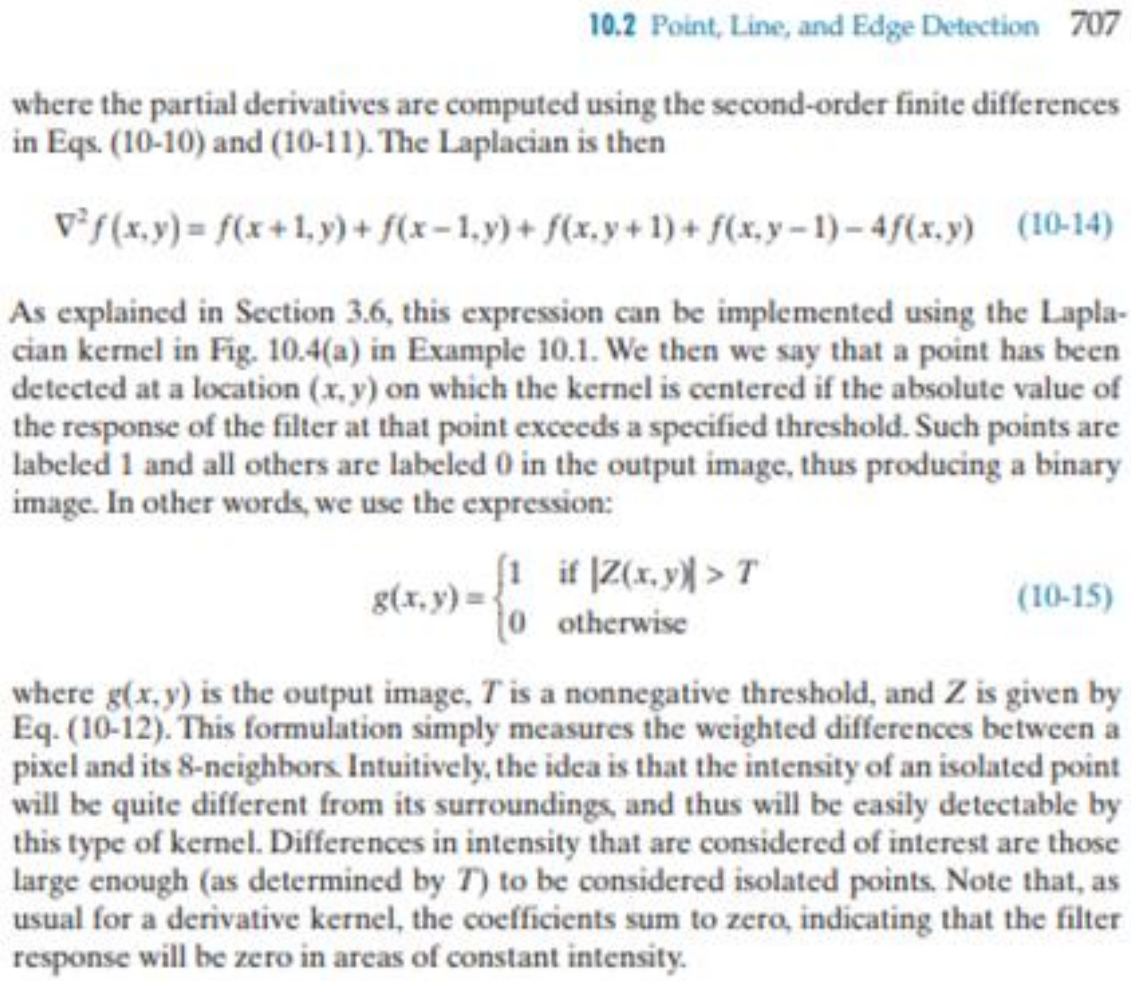
The focus of this section is on segmentation methods that are based on detecting sharp, *local* changes in intensity. The three types of image features in which we are interested are isolated points, lines, and edges. *Edge pixels* are pixels at which the intensity of an image function changes abruptly, and *edges* (or *edge segments*) are sets of connected edge pixels (see Section 2.5.2 regarding connectivity). *Edge detectors* are local image processing methods designed to detect edge pixels. A line may be viewed as an edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels. In fact, as we discuss in the following section and in Section 10.2.4, lines give rise to so-called "roof edges." Similarly, an isolated point may be viewed as a line whose length and width are equal to one pixel.



DETECTION OF ISOLATED POINTS

Based on the conclusions reached in the preceding section, we know that point detection should be based on the second derivative which, from the discussion in Section 3.6, means using the Laplacian:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (10\text{-}13)$$



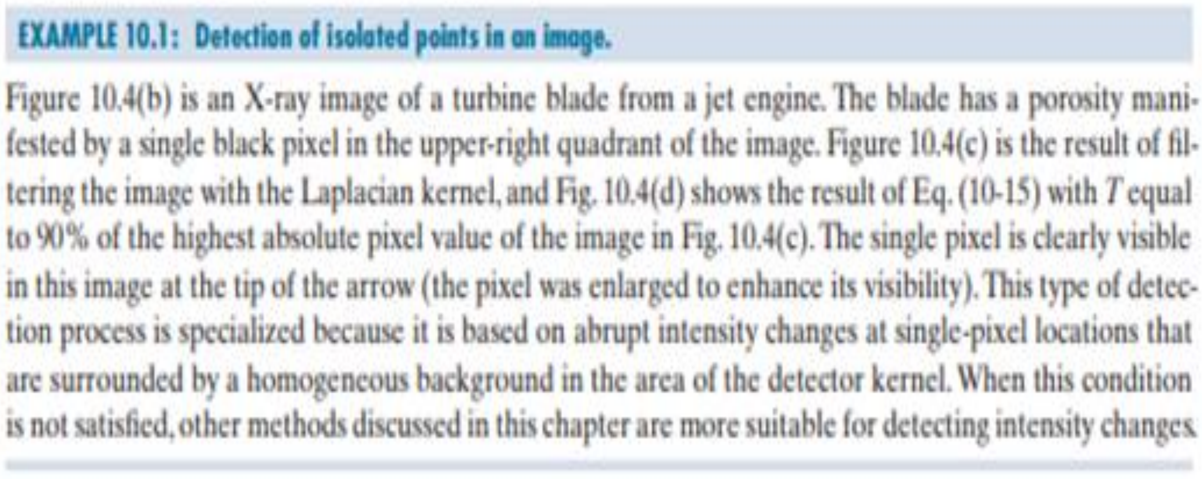
where the partial derivatives are computed using the second-order finite differences in Eqs. (10-10) and (10-11). The Laplacian is then

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \quad (10\text{-}14)$$

As explained in Section 3.6, this expression can be implemented using the Laplacian kernel in Fig. 10.4(a) in Example 10.1. We then we say that a point has been detected at a location (x, y) on which the kernel is centered if the absolute value of the response of the filter at that point exceeds a specified threshold. Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image. In other words, we use the expression:

$$g(x,y) = \begin{cases} 1 & \text{if } |Z(x,y)| > T \\ 0 & \text{otherwise} \end{cases} \quad (10\text{-}15)$$

where $g(x, y)$ is the output image, T is a nonnegative threshold, and Z is given by Eq. (10-12). This formulation simply measures the weighted differences between a pixel and its 8-neighbors. Intuitively, the idea is that the intensity of an isolated point will be quite different from its surroundings, and thus will be easily detectable by this type of kernel. Differences in intensity that are considered of interest are those large enough (as determined by T) to be considered isolated points. Note that, as usual for a derivative kernel, the coefficients sum to zero, indicating that the filter response will be zero in areas of constant intensity.

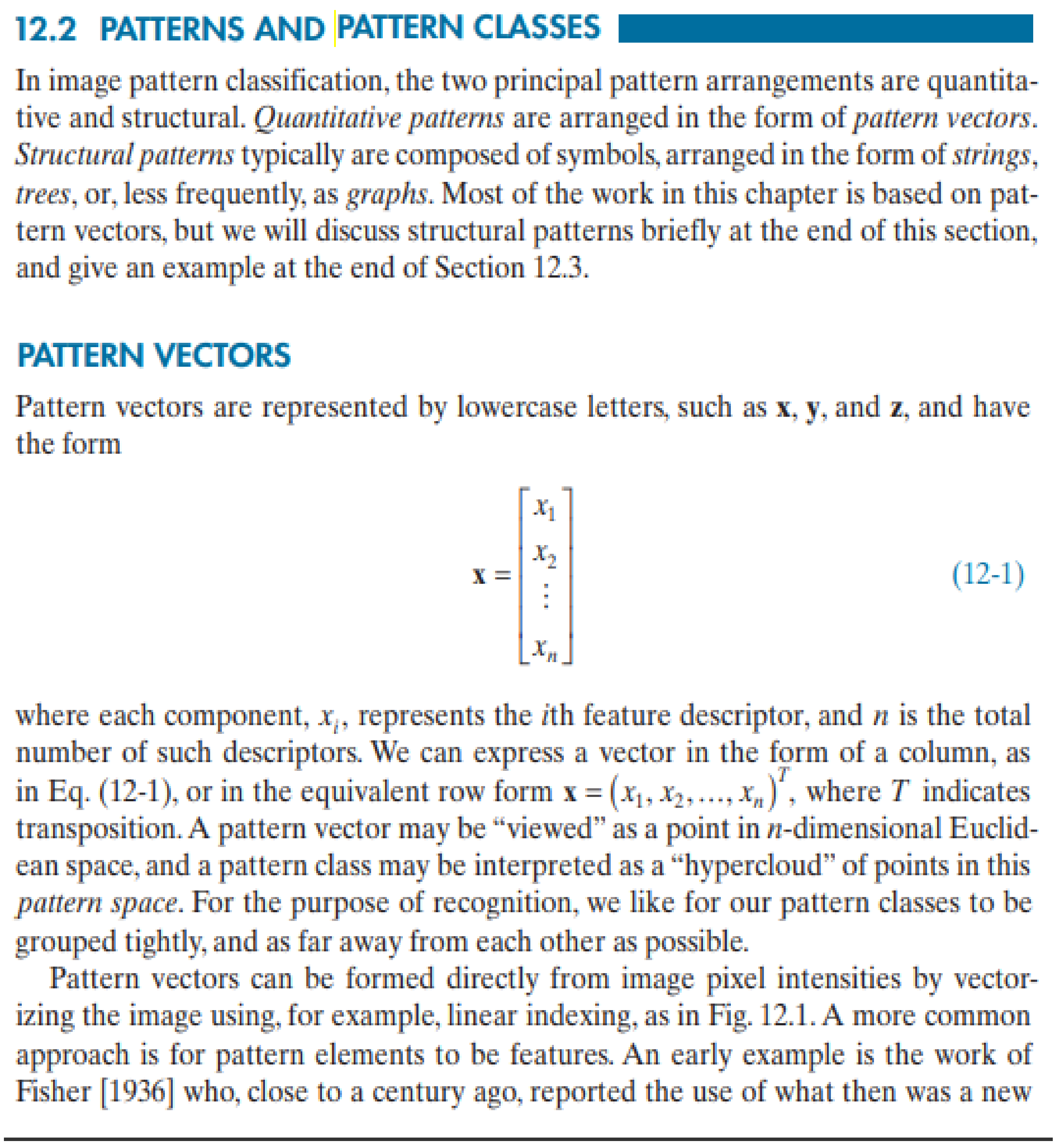


EXAMPLE 10.1: Detection of isolated points in an image.

Figure 10.4(b) is an X-ray image of a turbine blade from a jet engine. The blade has a porosity manifested by a single black pixel in the upper-right quadrant of the image. Figure 10.4(c) is the result of filtering the image with the Laplacian kernel, and Fig. 10.4(d) shows the result of Eq. (10-15) with T equal to 90% of the highest absolute pixel value of the image in Fig. 10.4(c). The single pixel is clearly visible in this image at the tip of the arrow (the pixel was enlarged to enhance its visibility). This type of detection process is specialized because it is based on abrupt intensity changes at single-pixel locations that are surrounded by a homogeneous background in the area of the detector kernel. When this condition is not satisfied, other methods discussed in this chapter are more suitable for detecting intensity changes.

Q7. Explain the pattern and pattern classes in object recognition

Ans:



12.2 PATTERNS AND PATTERN CLASSES

In image pattern classification, the two principal pattern arrangements are quantitative and structural. *Quantitative patterns* are arranged in the form of *pattern vectors*. *Structural patterns* typically are composed of symbols, arranged in the form of *strings*, *trees*, or, less frequently, as *graphs*. Most of the work in this chapter is based on pattern vectors, but we will discuss structural patterns briefly at the end of this section, and give an example at the end of Section 12.3.

PATTERN VECTORS

Pattern vectors are represented by lowercase letters, such as **x**, **y**, and **z**, and have the form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad (12\text{-}1)$$

where each component, x_i, represents the ith feature descriptor, and n is the total number of such descriptors. We can express a vector in the form of a column, as in Eq. (12-1), or in the equivalent row form $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$, where T indicates transposition. A pattern vector may be "viewed" as a point in n-dimensional Euclidean space, and a pattern class may be interpreted as a "hypercloud" of points in this *pattern space*. For the purpose of recognition, we like for our pattern classes to be grouped tightly, and as far away from each other as possible.

Pattern vectors can be formed directly from image pixel intensities by vectorizing the image using, for example, linear indexing, as in Fig. 12.1. A more common approach is for pattern elements to be features. An early example is the work of Fisher [1936] who, close to a century ago, reported the use of what then was a new a b

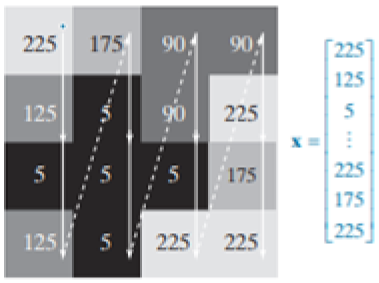


FIGURE 12.1 Using linear indexing to vectorize a grayscale image.

Sepals are the undergrowth beneath the petals.

technique called *discriminant analysis* to recognize three types of iris flowers (*Iris setosa*, *virginica*, and *versicolor*). Fisher described each flower using four features: the length and width of the petals, and similarly for the sepals (see Fig. 12.2). This leads to the 4-D vectors shown in the figure. A set of these vectors, obtained for fifty samples of each flower gender, constitutes the three famous *Fisher iris pattern classes*. Had Fisher been working today, he probably would have added spectral colors and shape features to his measurements, yielding vectors of higher dimensionality. We will be working with the original iris data set later in this chapter.

A higher-level representation of patterns is based on feature descriptors of the types you learned in Chapter 11. For instance, pattern vectors formed from descriptors of boundary shape are well-suited for applications in controlled environments, such as industrial inspection. Figure 12.3 illustrates the concept. Here, we are interested in classifying different types of noisy shapes, a sample of which is shown in the figure. If we represent an object by its signature, we would obtain 1-D signals of the form shown in Fig. 12.3(b). We can express a signature as a vector by sampling its amplitude at increments of θ, then formimg a vector by letting $x_i = r(\theta_i)$, for $i = 0, 1, 2, \ldots, n$. Instead of using "raw" sampled signatures, a more common approach is to compute some function, $x_i = g\left(r(\theta_i)\right)$, of the signature samples and use them to form vectors. You learned in Section 11.3 several approaches to do this, such as statistical moments.

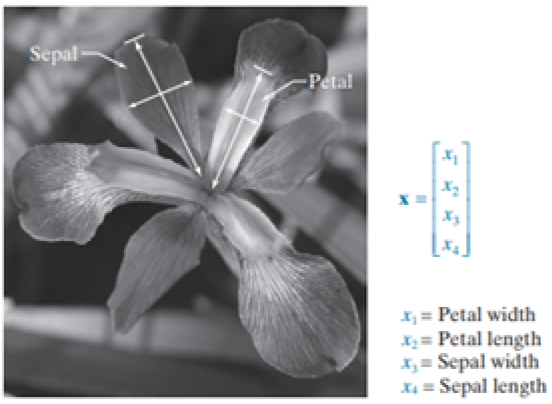
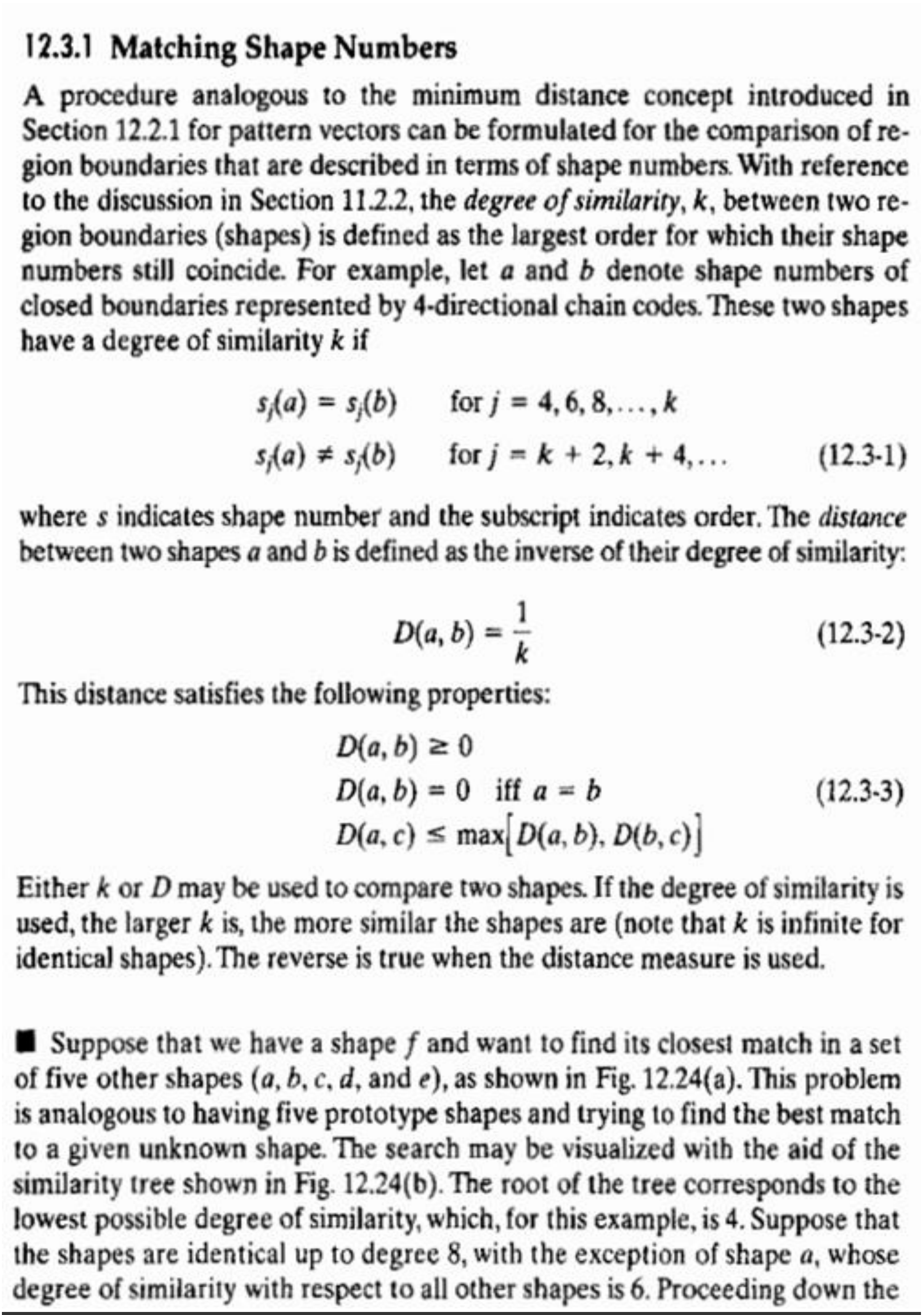


FIGURE 12.2 Petal and sepal width and length measurements (see arrows) performed on iris flowers for the purpose of data classification. The image shown is of the *Iris virginica* gender. (Image courtesy of USDA.)

Q8. Explain Matching Shape Numbers

Ans:



12.3.1 Matching Shape Numbers

A procedure analogous to the minimum distance concept introduced in Section 12.2.1 for pattern vectors can be formulated for the comparison of region boundaries that are described in terms of shape numbers. With reference to the discussion in Section 11.2.2, the *degree of similarity*, k, between two region boundaries (shapes) is defined as the largest order for which their shape numbers still coincide. For example, let a and b denote shape numbers of closed boundaries represented by 4-directional chain codes. These two shapes have a degree of similarity k if

$$\begin{aligned} s_j(a) &= s_j(b) \qquad \text{for } j = 4, 6, 8, \ldots, k \\ s_j(a) &\neq s_j(b) \qquad \text{for } j = k+2, k+4, \ldots \end{aligned} \tag{12.3-1}$$

where s indicates shape number and the subscript indicates order. The *distance* between two shapes a and b is defined as the inverse of their degree of similarity:

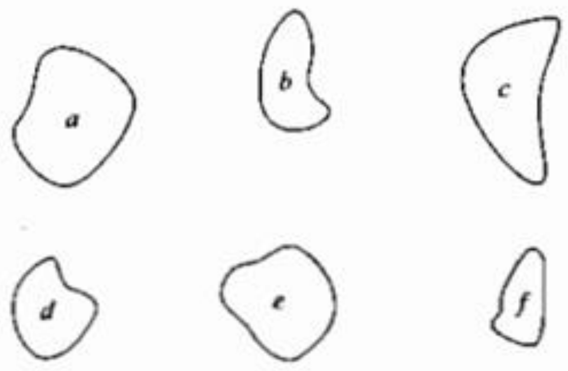
$$D(a, b) = \frac{1}{k} \tag{12.3-2}$$

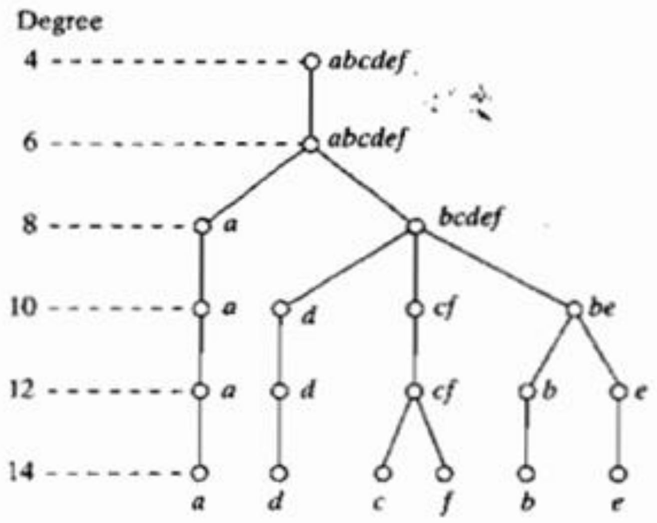
This distance satisfies the following properties:

$$\begin{aligned} &D(a, b) \geq 0 \\ &D(a, b) = 0 \quad \text{iff } a = b \\ &D(a, c) \leq \max\left[D(a, b), D(b, c)\right] \end{aligned} \tag{12.3-3}$$

Either k or D may be used to compare two shapes. If the degree of similarity is used, the larger k is, the more similar the shapes are (note that k is infinite for identical shapes). The reverse is true when the distance measure is used.

■ Suppose that we have a shape f and want to find its closest match in a set of five other shapes ($a, b, c, d,$ and e), as shown in Fig. 12.24(a). This problem is analogous to having five prototype shapes and trying to find the best match to a given unknown shape. The search may be visualized with the aid of the similarity tree shown in Fig. 12.24(b). The root of the tree corresponds to the lowest possible degree of similarity, which, for this example, is 4. Suppose that the shapes are identical up to degree 8, with the exception of shape a, whose degree of similarity with respect to all other shapes is 6. Proceeding down the





	a	*b*	*c*	*d*	*e*	*f*
a	∞	6	6	6	6	6
b		∞	8	8	10	8
c			∞	8	8	12
d				∞	8	8
e					∞	8
f						∞

tree, we find that shape *d* has degree of similarity 8 with respect to all others, and so on. Shapes *f* and *c* match uniquely, having a higher degree of similarity than any other two shapes. At the other extreme, if *a* had been an unknown shape, all we could have said using this method is that *a* was similar to the other five shapes with degree of similarity 6. The same information can be summarized in the form of a *similarity matrix*, as shown in Fig. 12.24(c).