CHAPTER

9

Image Segmentation

Not everything that can be counted counts, and not everything that counts can be counted.

–Albert Einstein

LEARNING OBJECTIVES

Segmentation is the process of partitioning a digital image into multiple regions and extracting meaningful regions known as regions of interest (ROI) for further image analysis. Segmentation is an important phase in the image analysis process. After studying this chapter, the reader will become familiar with the following:

- Types of segmentation algorithms
- Edge detection algorithms
- Thresholding techniques
- Region-oriented segmentation algorithms
- Active contour models
- Evaluation of segmentation algorithms

INTRODUCTION

Image segmentation has emerged as an important phase in image-based applications. Segmentation is the process of partitioning a digital image into multiple regions and extracting a meaningful region known as the region of interest (ROI). Regions of interest vary with applications. For example, if the goal of a doctor is to analyse the tumour in a computer tomography (CT) image, then the tumour in the image is the ROI. Similarly, if the image application aims to recognize the iris in an eye image, then the iris in the eye image is the required ROI. Segmentation of ROI in real-world images is the first major hurdle for effective implementation of image processing applications as the segmentation process is often difficult. Hence, the success or failure of the extraction of ROI ultimately influences the success of image processing applications. No single universal segmentation algorithm exists for segmenting the ROI in all images. Therefore, the user has to try many segmentation algorithms and pick an algorithm that performs the best for the given requirement.

Image segmentation algorithms are based on either discontinuity principle or similarity principle. The idea behind discontinuity principle is to extract regions that differ in properties such as intensity, colour, texture, or any other image statistics. Mostly, abrupt changes in intensity among the regions result in extraction of edges. The idea behind similarity principle is to group pixels based on a common property, to extract a coherent region.

9.1.1 Formal Definition of Image Segmentation

An image can be partitioned into many regions $R_1, R_2, R_3, \ldots, R_n$. For example, the image R in Fig. 9.1(a) is divided into three subregions R_1, R_2, and R_3 as shown in Figs 9.1(b) and 9.1(c). A subregion or sub-image is a portion of the whole region R. The identified subregions should exhibit characteristics such as uniformity and homogeneity with respect to colour, texture, intensity, or any other statistical property. In addition, the boundaries that separate the regions should be simple and clear.

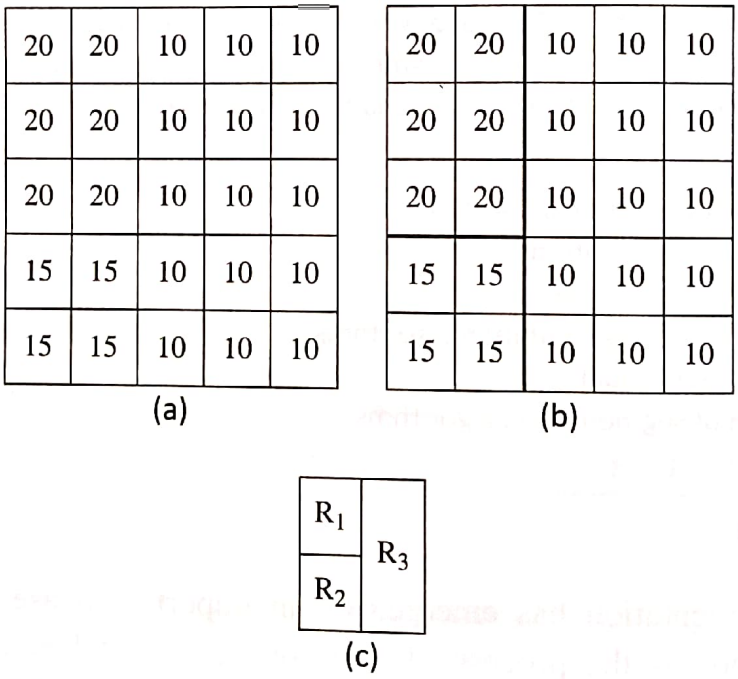


Fig. 9.1 Image segmentation (a) Original image (b) Pixels that form a region (c) Image with three regions

The characteristics of the segmentation process are the following:

1. If the subregions are combined, the original region can be obtained. Mathematically, it can be stated that $\bigcup R_i = R$ for $i = 1, 2, \ldots, n$. For example, if the three regions of Fig. 9.1(c) R_1, R_2, and R_3 are combined, the whole region R is obtained.
2. The subregions R_i should be connected. In other words, the region cannot be open-ended during the tracing process.
3. The regions $R_1, R_2, \ldots, R_n$ do not share any common property. Mathematically, it can be stated as $R_i \bigcap R_j = \varphi$ for all i and j where $i \neq j$. Otherwise, there is no justification for the region to exist separately.

4. Each region satisfies a predicate or a set of predicates such as intensity or other image statistics, that is, the predicate (P) can be colour, grey scale value, texture, or any other image statistic. Mathematically, this is stated as $P(R_i) = \text{True}$.

9.2 CLASSIFICATION OF IMAGE SEGMENTATION ALGORITHMS

There are different ways of classifying the segmentation algorithms. Figure 9.2 illustrates these ways. One way is to classify the algorithms based on user interaction required for extracting the ROI. Another way is to classify them based on the pixel relationships.

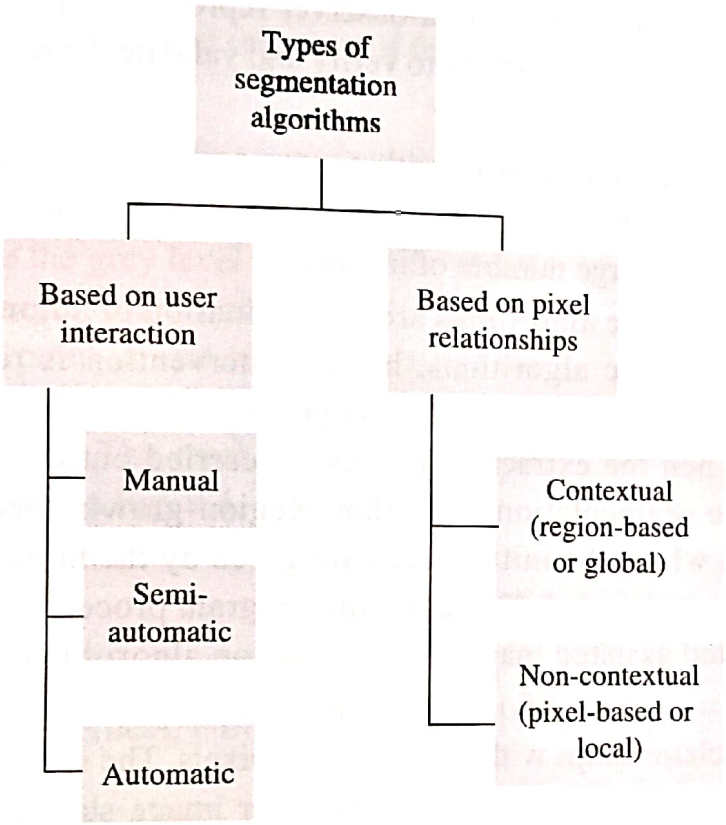


Fig. 9.2 Classification of segmentation algorithms

Based on user interaction, the segmentation algorithms can be classified into the following three categories:

1. Manual
2. Semi-automatic
3. Automatic

The words 'algorithm' and 'method' can be used interchangeably. In the manual method, the object of interest is observed by an expert who traces its ROI boundaries as well, with the help of software. Hence, the decisions related to segmentation are made by the human observers. Many software systems assist experts in tracing the boundaries and extracting them. By using the software systems, the experts outline the object.

The outline can be either an open or closed contour. Some software systems provide additional help by connecting the open tracings automatically to give a closed region. These closed outlines are then converted into a series of control points. These control points are then connected by spline. The advantage of the control points is that even if there is a displacement, the software system ensures that they are always connected. Finally, the software provides help to the user in extracting the closed regions.

Boundary tracing is a subjective process and hence variations exist among opinions of different experts in the field, leading to problems in reproducing the same results. In addition, a manual method of extraction is time consuming, highly subjective, prone to human error, and has poor intra-observer reproducibility. However, manual methods are still used commonly by experts to verify and validate the results of automatic segmentation algorithms.

Automatic segmentation algorithms are a preferred choice as they segment the structures of the objects without any human intervention. They are preferred if the tasks need to be carried out for a large number of images.

Semi-automatic algorithms are a combination of automatic and manual algorithms. In semi-automatic algorithms, human intervention is required in the initial stages. Normally, the human observer is supposed to provide the initial seed points indicating the ROI. Then the extraction process is carried out automatically as dictated by the logic of the segmentation algorithm. Region-growing techniques are semi-automatic algorithms where the initial seeds are given by the human observer in the region that needs to be segmented. However, the program process is automatic. These algorithms can be called assisted manual segmentation algorithms.

Another way of classifying the segmentation algorithms is to use the criterion of the pixel similarity relationships with neighbouring pixels. The similarity relationships can be based on colour, texture, brightness, or any other image statistic. On this basis, segmentation algorithms can be classified as follows:

1. Contextual (region-based or global) algorithms
2. Non-contextual (pixel-based or local) algorithms

Contextual algorithms group pixels together based on common properties by exploiting the relationships that exist among the pixels. These are also known as region-based or global algorithms. In region-based algorithms, the pixels are grouped based on some sort of similarity that exists between them. Non-contextual algorithms are also known as pixel-based or local algorithms. These algorithms ignore the relationship that exists between the pixels or features. Instead, the idea is to identify the discontinuities that are present in the image such as isolated lines and edges. These are then simply grouped into a region based on some global-level property. Intensity-based thresholding is a good example of this method.

9.3 DETECTION OF DISCONTINUITIES

The three basic types of grey level discontinuities in a digital image are the following:

1. Points
2. Lines
3. Edges

9.3.1 Point Detection

An isolated point is a point whose grey level is significantly different from its background in a homogeneous area. A generic 3×3 spatial mask is shown in Fig. 9.3.

The mask is superimposed onto an image and the convolution process is applied. The response of the mask is given as

$$R = \sum_{k=1}^{9} z_k f_k$$

Fig. 9.3 Generic 3×3 spatial mask

where the f_k values are the grey level values of the pixels associated with the image. A threshold value T is used to identify the points. A point is said to be detected at the location on which the mask is centred if $|R| \geq T$, where T is a non-negative integer. The mask values of a point detection mask are shown in Fig. 9.4.

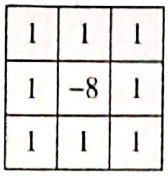


Fig. 9.4 Point detection mask

9.3.2 Line Detection

In line detection, four types of masks are used to get the responses, that is, R_1, R_2, R_3, and R_4 for the directions vertical, horizontal, +45°, and –45°, respectively. The masks are shown in Fig. 9.5(a).

These masks are applied to the image. The response of the mask is given as $R_k = \sum_{k=1}^{4} z_k f_k$.

R_1 is the response for moving the mask from the left to the right of the image. R_2 is the response for moving the mask from the top to the bottom of the image. R_3 is the response of the mask along the +45° line and R_4 is the response of the mask with respect to a line of –45°. Suppose at a certain line on the image, $|R_i| > |R_j|\ \forall\ j \neq i$, then that line is more likely to be associated with the orientation of the mask. The final maximum response is defined by $\max_{i=1}^{4}\{R_i\}$ and the line is associated with that mask. A sample image and the results of the line-detection algorithm are shown in Fig. 9.5(b).

$$M_1 = \begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix}, M_2 = \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}, M_3 = \begin{pmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{pmatrix}, M_4 = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

(a)

Fig. 9.5 Line detection (a) Mask for line detection

9.3 DETECTION OF DISCONTINUITIES

The three basic types of grey level discontinuities in a digital image are the following:

1. Points
2. Lines
3. Edges

9.3.1 Point Detection

An isolated point is a point whose grey level is significantly different from its background in a homogeneous area. A generic 3×3 spatial mask is shown in Fig. 9.3.

The mask is superimposed onto an image and the convolution process is applied. The response of the mask is given as

$$R = \sum_{k=1}^{9} z_k f_k$$

where the f_k values are the grey level values of the pixels associated with the image. A threshold value T is used to identify the points. A point is said to be detected at the location on which the mask is centred if $|R| \geq T$, where T is a non-negative integer. The mask values of a point detection mask are shown in Fig. 9.4.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Fig. 9.3 Generic 3×3 spatial mask

1	1	1
1	−8	1
1	1	1

Fig. 9.4 Point detection mask

9.3.2 Line Detection

In line detection, four types of masks are used to get the responses, that is, R_1, R_2, R_3, and R_4 for the directions vertical, horizontal, +45°, and −45°, respectively. The masks are shown in Fig. 9.5(a).

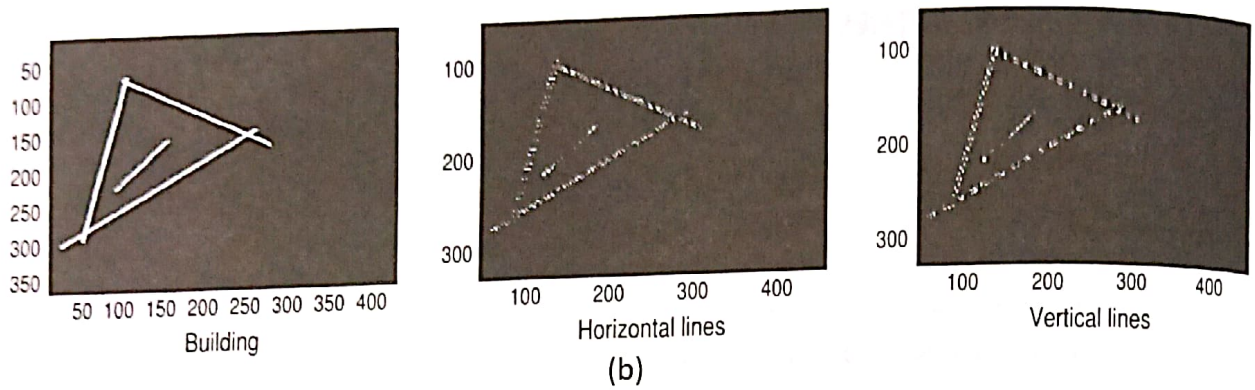
These masks are applied to the image. The response of the mask is given as $R_k = \sum_{k=1}^{4} z_k f_k$.

R_1 is the response for moving the mask from the left to the right of the image. R_2 is the response for moving the mask from the top to the bottom of the image. R_3 is the response of the mask along the +45° line and R_4 is the response of the mask with respect to a line of −45°. Suppose at a certain line on the image, $|R_i| > |R_j|\ \forall\ j \neq i$, then that line is more likely to be associated with the orientation of the mask. The final maximum response is defined by $\max_{i=1}^{4}\{R_i\}$ and the line is associated with that mask. A sample image and the results of the line-detection algorithm are shown in Fig. 9.5(b).

$$M_1 = \begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix}, M_2 = \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix}, M_3 = \begin{pmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{pmatrix}, M_4 = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

(a)

Fig. 9.5 Line detection (a) Mask for line detection



(b)

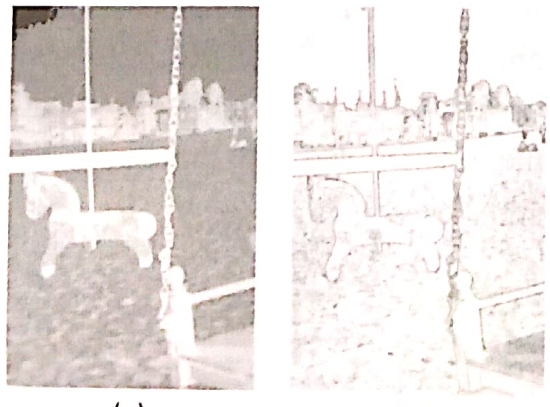
Fig. 9.5 (b) Original image and detected lines

9.4 EDGE DETECTION

Edges play a very important role in many image processing applications. They provide an outline of the object. In the physical plane, edges correspond to the discontinuities in depth, surface orientation, change in material properties, and light variations. These variations are present in the image as grey scale discontinuities. An edge is a set of connected pixels that lies on the boundary between two regions that differ in grey value. The pixels on an edge are called edge points. A reasonable definition of an edge requires the ability to measure grey level transitions in a meaningful manner. Most edges are unique in space, that is, their position and orientation remain the same in space when viewed from different points. When an edge is detected, the unnecessary details are removed, while only the important structural information is retained. In short, an edge is a local concept which represents only significant intensity transitions. An original image and its edges are shown in Figs 9.6(a) and 9.6(b), respectively.

An edge is typically extracted by computing the derivative of the image function. This consists of two parts—magnitude of the derivative, which is an indication of the strength/contrast of the edge, and the direction of the derivative vector, which is a measure of edge orientation. Some of the edges that are normally encountered in image processing are as follows:

1. Step edge
2. Ramp edge
3. Spike edge
4. Roof edge



(a) (b)

Fig. 9.6 Edge detection (a) Original image (b) Extracted edges

These are shown in Fig. 9.7.

Step edge is an abrupt intensity change. Ramp edge, on the other hand, represents a gradual change in intensity. Spike edge represents a quick change and immediately returns to the original intensity level. Roof edge is not instantaneous over a short distance.

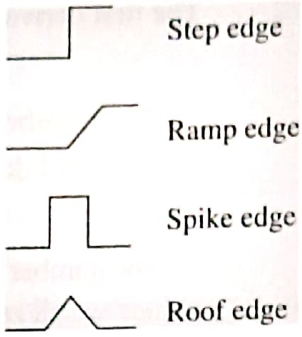


Fig. 9.7 Types of edges

9.4.1 Stages in Edge Detection

The idea is to detect the sharp changes in image brightness, which can capture the important events and properties. This is done in three stages. The edge detection process is shown in Fig. 9.8.

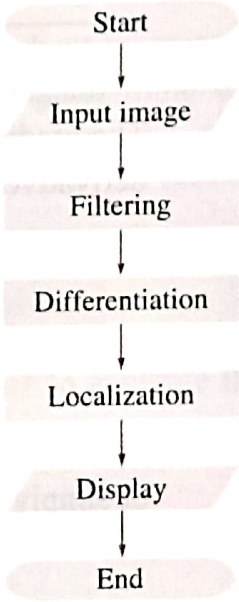


Fig. 9.8 Edge detection process

9.4.1.1 Filtering

It is better to filter the input image to get maximum performance for the edge detectors. This stage may be performed either explicitly or implicitly. It involves smoothing, where the noise is suppressed without affecting the true edges. In addition, this phase uses a filter to enhance the quality of the edges in the image. Normally, Gaussian filters are used as they are proven to be very effective for real-time images.

9.4.1.2 Differentiation

This phase distinguishes the edge pixels from other pixels. The idea of edge detection is to find the difference between two neighbourhood pixels. If the pixels have the same value, the difference is 0. This means that there is no transition between the pixels. The non-zero difference indicates the presence of an edge point. A point is defined as an edge point (or edgel) if its first derivative is greater than the user-specified threshold and encounters a sign change (zero crossing) in the second derivative. The first derivative is $\frac{\partial x}{\partial y} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x}$. Images are discrete. Hence, Δx should be discrete and should be at least 1. Therefore, the gradient vector ∇x (called grad of x) should be equal to $f(x) - f(x - 1)$. If the intensities are same, the derivative is 0. The non-zero element indicates the presence of images. In the case of the second derivatives, the zero-crossings indicate the presence of edges.

Example 9.1 Consider a one-dimensional image $f(x) = 60\ 60\ 60\ 100\ 100\ 100$. What are the first and second derivatives?

Solution The given one-dimensional image is

60 60 60 100 100 100

The first derivative is $f(x+1) - f(x)$. Therefore, the first derivative for the function is given as

0 0 40 0 0

The number 40 is due to the difference (100 – 60 = 40). Remaining values are all zeros. The second derivative is the difference in values of the first derivative. This is now given as

0 40 –40 0

The number 40 is due to the difference (40 – 0) and –40 is due to the difference (0 – 40). Remaining values are all zeros.

The highest magnitude 40 shows the presence of an edge in the first derivative. It can be observed that the sign change in the second derivative represents the edge. This sign change is important and is called zero-crossing.

Images are two-dimensional. Hence, the gradient vector of $f(x,y)$ is also two-dimensional. The gradient of an image $f(x,y)$ at location (x,y) is a vector that consists of the partial derivatives of $f(x,y)$ as follows:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

or simply

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

where $$g_x = \left[\frac{\partial f(x,y)}{\partial x}\right] \text{ and } g_y = \left[\frac{\partial f(x,y)}{\partial y}\right]$$

The magnitude of this vector, generally referred to as the gradient ∇f, is

$$\nabla f(x,y) = \text{mag}(\nabla f(x,y)) = \left[(g_x)^2 + (g_y)^2\right]^{1/2}$$

Edge strength is indicated by the edge magnitude. The direction of the gradient vector is useful in detecting a sudden change in image intensity. The common practice is to approximate the gradient with absolute values that are simpler to implement, as follows:

$$\nabla f(x,y) \cong |g_x| + |g_y|$$

or

$$\nabla f(x,y) \cong \max(g_x, g_y)$$

The gradient direction can be given as

$$\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

9.4.1.3 Localization

In this stage the detected edges are localized. The localization process involves determining the exact location of the edge. In addition, this stage involves edge thinning and edge linking steps to ensure that the edge is sharp and connected. The sharp and connected edges are then displayed.

The prerequisite for the localization stage is normalization of the gradient magnitude. The calculated gradient can be scaled to a specific range say, 0–K by performing this operation. For example, the value of constant K may be an integer, say, 100. $N(x, y)$ is called the normalized edge image and is given as

$$N(x,y) = \frac{G(x,y)}{\max_{i=1,\ldots,n, j=1,\ldots,n} G(i,j)} \times K$$

The normalized magnitude can be compared with a threshold value T to generate the edge map.

The edge map is given as

$$E(x,y) = \begin{cases} 1 \text{ if } N(x,y) > 1 \\ 0 \text{ otherwise} \end{cases}$$

The edge map is then displayed or stored for further image processing operations.

9.4.2 Types of Edge Detectors

The edge detection process is implemented in all kinds of edge detectors. In image processing, four types of edge detection operators are available. As shown in Fig. 9.9, they are gradient (or derivative) filters, template matching filters, Gaussian derivatives, and pattern fit approach.

Derivative filters use the differentiation technique to detect the edges. *Template matching filters* use templates that resemble the target shapes and match with the image. Gradient operations are isotropic in nature as they detect edges in all directions. Hence template matching filters are used to perform directional smoothing as they are very sensitive to directions. If there is a match between the target shape or directions and the masks, then a maximum gradient value is produced. By rotating the template in all eight directions, masks that are sensitive in all directions, called compass masks, are produced. Point detection and

line detection masks are good examples of template matching filters. *Gaussian derivatives* are very effective for real-time images and are used along with the derivative filters. *Pattern fit* is another approach, where a surface is considered as a topographic surface, with the pixel value representing altitude. The aim is to fit a pattern over a neighbourhood of a pixel where the edge strength is calculated. The properties of the edge points are calculated based on the parameters.

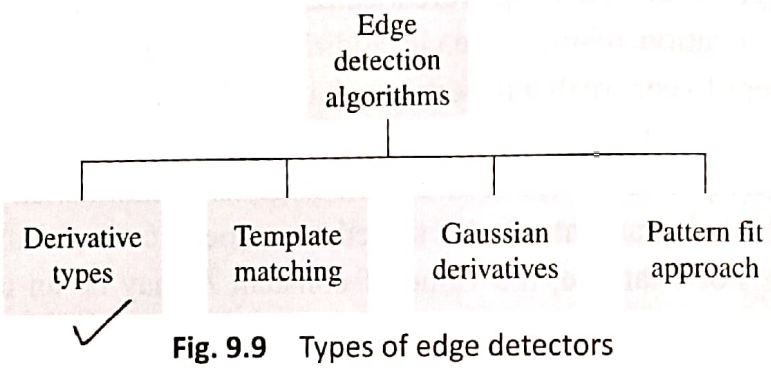


Fig. 9.9 Types of edge detectors

9.4.3 First-order Edge Detection Operators

Local transitions among different image intensities constitute an edge. Therefore, the aim is to measure the intensity gradients. Hence, edge detectors can be viewed as gradient calculators. Based on differential geometry and vector calculus, the gradient operator is represented as

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$

Applying this to the image f, one gets

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The differences between the pixels are quantified by the gradient magnitude. The direction of the greatest change is given by the gradient vector. This gives the directions of the edge. Since the gradient functions are continuous functions, the discrete versions of continuous functions can be used. This can be done by finding the differences. The approaches in 1D are as follows. Here Δx and Δy are the movements in x and y directions, respectively.

$$\text{Backward difference} = \frac{f(x) - f(x - \Delta x)}{\Delta x}$$

$$\text{Forward difference} = \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

$$\text{Central difference} = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}$$

These differences can be obtained by applying the following masks, assuming $\Delta x = 1$:

$$\text{Backward difference} = f(x) - f(x - 1) = [1 \; -1]$$

$$\text{Forward difference} = f(x + 1) - f(x) = [-1 \; +1]$$

$$\text{Central difference} = 1/2 \times [1 \; 0 \; -1]$$

These differences can be extended to 2D as $g_x = \frac{\partial f}{\partial x}$ and $g_y = \frac{\partial f}{\partial y}$. Then the magnitude is given by

$$\nabla f(x, y) = \text{mag}(\nabla f(x, y)) = \left[(g_x)^2 + (g_y)^2 \right]^{1/2}$$

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$.

9.4.3.1 Roberts operator

Let $f(x,y)$ and $f(x + 1, y)$ be neighbouring pixels. The difference between the adjacent pixels is obtained by applying the mask [1 −1] directly to the image to get the difference between the pixels. This is defined mathematically as

$$\frac{\partial f}{\partial x} = f(x + 1, y) - f(x, y)$$

Roberts kernels are derivatives with respect to the diagonal elements. Hence they are called cross-gradient operators. They are based on the cross diagonal differences. The approximation of Roberts operator can be mathematically given as

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

Roberts masks of the for the given cross difference is

$$g_x = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \text{ and } g_y = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

As discussed earlier the magnitude of this vector can be calculated as

$$\nabla f(x,y) = \text{mag}(\nabla f(x,y)) = \left[(g_x)^2 + (g_y)^2\right]^{1/2}$$

and the edge orientation is given by

$$\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

Since the magnitude calculation involves square root operation, the common practice is to approximate the gradient with absolute values that are simpler to implement, as

$$\nabla f(x,y) \cong |g_x| + |g_y|$$

The generic gradient-based algorithm can be given as

1. Read the image and smooth it.
2. Convolve the image f with g_x. Let $\hat{f}(x) = f * g_x$.
3. Convolve the image with g_y. Let $\hat{f}(y) = f * g_y$.
4. Compute the edge magnitude and edge orientation.
5. Compare the edge magnitude with a threshold value. If the edge magnitude is higher, assign it as a possible edge point.

This generic algorithm can be applied to other masks also.

9.4.3.2 Prewitt operator

The Prewitt method takes the central difference of the neighbouring pixels; this difference can be represented mathematically as

$$\frac{\partial f}{\partial x} = f(x+1) - f(x-1)/2$$

For two dimensions, this is

$$f(x+1, y) - f(x-1, y)/2$$

The central difference can be obtained using the mask [−1 0 +1]. This method is very sensitive to noise. Hence to avoid noise, the Prewitt method does some averaging. The Prewitt approximation using a 3×3 mask is as follows:

$$\nabla f \cong \left|(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)\right| + \left|(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)\right|$$

This approximation is known as the Prewitt operator. Its masks are as follows:

$$M_x = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \text{ and } M_y = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

9.4.3.3 Sobel operator

The Sobel operator also relies on central differences. This can be viewed as an approximation of the first Gaussian derivative. This is equivalent to the first derivative of the Gaussian blurring image obtained by applying a 3×3 mask to the image. Convolution is both commutative and associative, and is given as

$$\frac{\partial}{\partial x}(f * G) = f * \frac{\partial}{\partial x} G$$

A 3×3 digital approximation of the Sobel operator is given as

$$\nabla f \cong \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| + \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

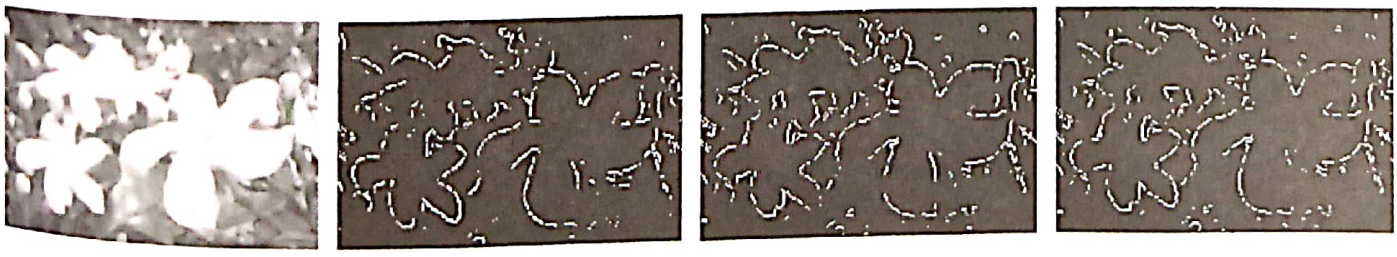
The masks are as follows:

$$M_x = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \text{ and } M_y = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

An additional mask can be used to detect the edges in the diagonal direction.

$$M_x = \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix} \text{ and } M_y = \begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

The edge mask can be extended to 5×5, 7×7, etc. An extended mask always gives a better performance. An original image and the result of applying the Roberts, Sobel, and Prewitt masks are shown in Figs 9.10(a)–9.10(d). It can be observed that the results of the masks vary.



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

Fig. 9.10 Edge detection using first-order operators (a) Original image (b) Roberts edge detection (b) Prewitts edge detection (c) Sobel edge detection

9.4.3.4 Template matching masks

Gradient masks are isotropic and insensitive to directions. Sometimes it is necessary to design direction sensitive filters. Such filters are called *template matching filters*. A few kinds of template matching masks are discussed in this section.

Kirsch masks Kirsch masks are called compass masks because they are obtained by taking one mask and rotating it to the eight major directions: north, north west, west, south west, south, south-east, east, and north-east. The respective masks are given as

$$K_0=\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & -3 & 5\end{bmatrix} K_1=\begin{bmatrix}-3 & 5 & 5\\ -3 & 0 & 5\\ -3 & -3 & -3\end{bmatrix} K_2=\begin{bmatrix}5 & 5 & 5\\ -3 & 0 & -3\\ -3 & -3 & -3\end{bmatrix} K_3=\begin{bmatrix}5 & 5 & -3\\ 5 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}$$

$$K_4=\begin{bmatrix}5 & -3 & -3\\ 5 & 0 & -3\\ 5 & -3 & -3\end{bmatrix} K_5=\begin{bmatrix}-3 & -3 & -3\\ 5 & 0 & -3\\ 5 & 5 & -3\end{bmatrix} K_6=\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & -3\\ 5 & 5 & 5\end{bmatrix} K_7=\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & 5\\ -3 & 5 & 5\end{bmatrix}$$

Each mask is applied to the image and the convolution process is carried out. The magnitude of the final edge is the maximum value of all the eight masks. The edge direction is the direction associated with the mask that produces maximum magnitude.

Robinson compass mask The spatial masks for the Robinson edge operator for all the directions are as follows:

$$R_0=\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix} R_1=\begin{pmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{pmatrix} R_2=\begin{pmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{pmatrix},$$

$$R_3=\begin{pmatrix}2 & 1 & 0\\ 1 & 0 & -1\\ 0 & -1 & -2\end{pmatrix} R_4=\begin{pmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{pmatrix} R_5=\begin{pmatrix}0 & -1 & -2\\ 1 & 0 & -1\\ 2 & 1 & 0\end{pmatrix},$$

$$R_6=\begin{pmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{pmatrix} R_7=\begin{pmatrix}-2 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 2\end{pmatrix}$$

Similar to Kirsch masks, the mask that produces the maximum value defines the direction of the edge. It is sufficient for edge detection. The results of the remaining masks are the negation of the first four masks. Thus the computational effort can be reduced.

Frei-Chen masks Any image can be considered as the weighted sum of the nine Frei-Chen masks. The weights are obtained by a process called projecting process by overlaying a $3\times$

9.4.3.4 Template matching masks

Gradient masks are isotropic and insensitive to directions. Sometimes it is necessary to design direction sensitive filters. Such filters are called *template matching filters*. A few kinds of template matching masks are discussed in this section.

Kirsch masks Kirsch masks are called compass masks because they are obtained by taking one mask and rotating it to the eight major directions: north, north west, west, south west, south, south-east, east, and north-east. The respective masks are given as

$$K_0=\begin{bmatrix}-3 & -3 & 5\\ -3 & 0 & 5\\ -3 & -3 & 5\end{bmatrix} K_1=\begin{bmatrix}-3 & 5 & 5\\ -3 & 0 & 5\\ -3 & -3 & -3\end{bmatrix} K_2=\begin{bmatrix}5 & 5 & 5\\ -3 & 0 & -3\\ -3 & -3 & -3\end{bmatrix} K_3=\begin{bmatrix}5 & 5 & -3\\ 5 & 0 & -3\\ -3 & -3 & -3\end{bmatrix}$$

$$K_4=\begin{bmatrix}5 & -3 & -3\\ 5 & 0 & -3\\ 5 & -3 & -3\end{bmatrix} K_5=\begin{bmatrix}-3 & -3 & -3\\ 5 & 0 & -3\\ 5 & 5 & -3\end{bmatrix} K_6=\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & -3\\ 5 & 5 & 5\end{bmatrix} K_7=\begin{bmatrix}-3 & -3 & -3\\ -3 & 0 & 5\\ -3 & 5 & 5\end{bmatrix}$$

Each mask is applied to the image and the convolution process is carried out. The magnitude of the final edge is the maximum value of all the eight masks. The edge direction is the direction associated with the mask that produces maximum magnitude.

Robinson compass mask The spatial masks for the Robinson edge operator for all the directions are as follows:

$$R_0=\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix} R_1=\begin{pmatrix}0 & 1 & 2\\ -1 & 0 & 1\\ -2 & -1 & 0\end{pmatrix} R_2=\begin{pmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{pmatrix},$$

$$R_3=\begin{pmatrix}2 & 1 & 0\\ 1 & 0 & -1\\ 0 & -1 & -2\end{pmatrix} R_4=\begin{pmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{pmatrix} R_5=\begin{pmatrix}0 & -1 & -2\\ 1 & 0 & -1\\ 2 & 1 & 0\end{pmatrix},$$

$$R_6=\begin{pmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{pmatrix} R_7=\begin{pmatrix}-2 & -1 & 0\\ -1 & 0 & 1\\ 0 & 1 & 2\end{pmatrix}$$

Similar to Kirsch masks, the mask that produces the maximum value defines the direction of the edge. It is sufficient for edge detection. The results of the remaining masks are the negation of the first four masks. Thus the computational effort can be reduced.

Frei-Chen masks Any image can be considered as the weighted sum of the nine Frei-Chen masks. The weights are obtained by a process called projecting process by overlaying a $3\times$

3 image onto each mask and by summing the multiplication of coincident terms. The first four masks represent the edge space, the next four represent the line subspace, and the last one represents the average subspace. The Frei-Chen masks are given as

$$F_1 = \frac{1}{2\sqrt{2}}\begin{pmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{pmatrix} \quad F_2 = \frac{1}{2\sqrt{2}}\begin{pmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{pmatrix}$$

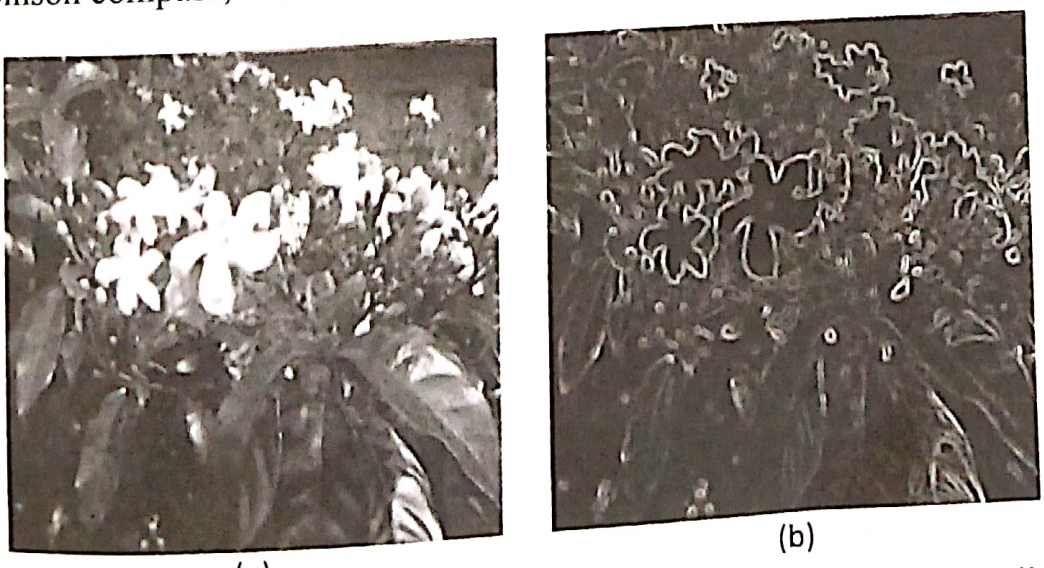
$$F_3 = \frac{1}{2\sqrt{2}}\begin{pmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ \sqrt{2} & 1 & 0 \end{pmatrix} \quad F_4 = \frac{1}{2\sqrt{2}}\begin{pmatrix} \sqrt{2} & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & \sqrt{2} \end{pmatrix}$$

$$F_5 = \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \quad F_6 = \frac{1}{2}\begin{pmatrix} -1 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{pmatrix}$$

$$F_7 = \frac{1}{6}\begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix} \quad F_8 = \frac{1}{6}\begin{pmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{pmatrix}$$

$$F_9 = \frac{1}{3}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Figures 9.11(a)–9.11(d) show an original image and the images obtained by using the Kirsch, Robinson compass, and Frei-Chen masks, respectively.



(a) (b)

Fig. 9.11 Template matching masks (a) Original image (b) Image obtained using Krisch mask

(c) (d)

Fig. 9.11 (c) Image obtained using Robinson compass mask
(d) Image obtained using Frei-Chen mask

9.4.4 Second-order Derivative Filters

Edges are considered to be present in the first derivative when the edge magnitude is la compared to the threshold value. In the case of the second derivative, the edge pixe present at a location where the second derivative is zero. This is equivalent to saying t $f''(x)$ has a zero-crossing which can be observed as a sign change in pixel differences. T Laplacian algorithm is one such zero-crossing algorithm.

However, the problems of the zero-crossing algorithms are many. The problem with Laplac masks is that they are sensitive to noise as there is no magnitude checking—even a small rip causes the method to generate an edge point. Therefore, it is necessary to filter the image bef the edge detection process is applied. This method produces two-pixel thick edges, althou generally, one-pixel thick edges are preferred. However, the advantage is that there is no need the edge thinning process as the zero-crossings themselves specify the location of the edge poi The main advantage is that these operators are rotationally invariant.

The second-order derivative is

$$\nabla \times \nabla = \begin{bmatrix} \partial/\partial x \\ \partial/\partial y \end{bmatrix} \begin{bmatrix} \partial/\partial x \\ \partial/\partial y \end{bmatrix}$$

$$= \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

This ∇^2 operator is called Laplacian operator. The Laplacian of the 2D function $f(x,$ is also defined as

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Since the gradient is a vector, two orthogonal filters are required. However, since the Laplacian operator is scalar, a single mask is sufficient for the edge detection process. The Laplacian estimate is given as

$$\frac{\partial^2 f(x,y)}{\partial y^2}$$
$$= \frac{\delta}{\delta x} f(x+1,y) - \frac{\delta}{\delta x} f(x,y)$$
$$= f(x+1,y) - f(x,y) - [f(x,y) - f(x-1,y)]$$
$$= f(x+1,y) - f(x,y) - f(x,y) + f(x-1,y)$$

Similarly, $\frac{\partial^2 f(x,y)}{\partial y^2} = f(x, y+1) - 2f(x, y) + f(x, y-1)$

Therefore, the Laplacian operator has the form

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} = f(x+1, y) + f(x, y+1) - 4f(x, y) + f(x, y-1) + f(x-1, y)$$

The algorithm is as follows:

1. Generate the mask.
2. Apply the mask.
3. Detect the zero-crossing. Zero-crossing is a situation where pixels in a neighbourhood differ from each other pixel in sign, that is, $|\nabla^2 f(p)| \leq |\nabla^2 f(q)|$, where p and q are two pixels.

Laplacian masks are shown in Figs 9.12(a)–9.12(d). The mask shown in Fig. 9.12(a) is sensitive to horizontal and vertical edges. It can be observed that the sum of the elements amounts to zero. To recognize the diagonal edges, the mask shown in Fig. 9.12(b) is used. This mask is obtained by rotating the mask of Fig. 9.12(a) by 45°. The addition of these two kernels results in a variant of the Laplacian mask shown in Fig. 9.12(c). Two times of the mask shown in Fig. 9.12(a), when subtracted from the mask shown in Fig. 9.12(b), yields another variant mask as shown in Fig. 9.12(d).

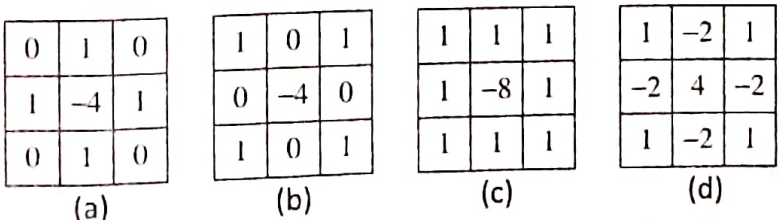


Fig. 9.12 Different Laplacian masks (a) Laplacian filter (b) 45° rotated mask (c) Variant 1 (d) Variant 2

The Laplacian operations are seldom used in practice because they produce double edges and are extremely sensitive to noise. However, the idea of zero-crossing is useful if it is combined with a smoothing signal to minimize sensitivity to noise. The result of applying the Laplacian method on Fig. 9.13(a) is shown in Fig. 9.13(b).

(a) (b)

Fig. 9.13 Laplacian method (a) Original image (b) Result of applying Laplacian mask

9.4.4.1 Laplacian of Gaussian (Marr–Hildrith) operator

To minimize the noise susceptibility of the Laplacian operator, the Laplacian of Gaussian (LoG) operator is often preferred. As a first step, the given image is blurred using the Gaussian operator and then the Laplacian operator is used. The Gaussian function reduces the noise and hence the Laplacian minimizes the detection of false edges.

For 1D, $\nabla^2(f * g) = f * \nabla^2 g = f * \text{LoG}$. Let the 2D Gaussian function be given as $G_\sigma(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{x^2+y^2}{2\sigma^2})$. To suppress the noise, the image is convolved with the Gaussian smoothing function before using the Laplacian for edge detection.

$$\nabla(G_\sigma(x,y) * f(x,y)] = [\nabla G_\sigma(x,y)] * f(x,y) = \text{LoG} * f(x,y)$$

The LoG function can be derived as

$$\frac{\partial}{\partial x} G_\sigma(x,y) = \frac{\partial}{\partial x} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} = -\frac{x}{\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$$

Similarly $\frac{\partial^2}{\partial x^2} G_\sigma(x,y) = \frac{x^2}{\sigma^4} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} - \frac{1}{\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} = \frac{x^2-\sigma^2}{\sigma^4} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$

Similarly, by ignoring the normalization constant $\frac{1}{\sqrt{2\pi\sigma^2}}$, one can get

$$\frac{\partial^2}{\partial y^2} G_\sigma(x,y) = \frac{y^2}{\sigma^4} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} - \frac{1}{\sigma^2} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)} = \frac{y^2-\sigma^2}{\sigma^4} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$$

The LoG kernel can now be described as

$$LoG \triangleq \frac{\partial^2}{\partial x^2} G_\sigma(x,y) + \frac{\partial^2}{\partial y^2} G_\sigma(x,y) = \frac{x^2+y^2-2\sigma^2}{\sigma^4} e^{\left(-\frac{x^2+y^2}{2\sigma^2}\right)}$$

As σ increases, wider convolution masks are required for better performance of the edge operator. A sample of an image and its result after application of the LoG operator are shown in Figs 9.14(a) and 9.14(b), respectively.

The LoG algorithm can now be stated as follows:

1. Generate the mask and apply LoG to the image.
2. Detect the zero-crossing.

(a) (b)

Fig. 9.14 Laplacian of Gaussian operator (a) Original image (b) Result of applying LoG operator

9.4.4.2 Combined detection

The information obtained from different orthogonal operators can be combined to get better results. One method is to combine the first- and second-order derivatives. This can be achieved by implementing the idea of scale space. Just like how the information of an object can be captured at different levels using different apertures of a camera, different types of Gaussian kernels of various sigma values can be used to capture information of the image at different levels and these information can be combined to get the edge map.

Another method called *hysteresis thresholding* uses two thresholds for thresholding the image. This is used by the canny edge detector method.

9.4.4.3 Difference of Gaussian filter

The LoG filter can be approximated by taking two differently sized Gaussians. The Difference of Gaussian (DoG) filter is given as

$$G_{\sigma_1}(x,y) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{x^2+y^2}{2\sigma_1^2}\right) \text{ with Gaussian of width } \sigma_1$$

The width of the Gaussian is changed and a new kernel is obtained as

$$G_{\sigma_2}(x,y) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{x^2+y^2}{2\sigma_2^2}\right) \text{ with Gaussian of width } \sigma_2$$

The DoG is expressed as the difference between these two Gaussian kernels:

$$\begin{aligned} G_{\sigma_1}(x,y) - G_{\sigma_2}(x,y) &= (G_{\sigma_1} - G_{\sigma_2}) * f(x,y) \\ &= \text{DoG} * f(x,y) \end{aligned}$$

The DoG as a kernel can now be stated as

$$\text{DoG} = G_{\sigma_1} - G_{\sigma_2} = \frac{1}{\sqrt{2\pi}}\left[\frac{1}{\sigma_1} e^{-\frac{(x^2+y^2)}{2\sigma_1^2}} - \frac{1}{\sigma_2} e^{-\frac{(x^2+y^2)}{2\sigma_2^2}}\right]$$

So the given image has to be convolved with a mask that is obtained by subtracting two Gaussian masks with two different σ values. If the $\frac{\sigma_1}{\sigma_2}$ value is between 1 and 2, the edge detection operator yields a good performance.

The DoG algorithm can now be stated as follows:

1. Generate the mask and apply DoG to the image.
2. Detect the zero-crossing and apply the threshold to suppress the weak zero-crossings.
3. Display and exit.

9.4.4.4 Canny edge detection

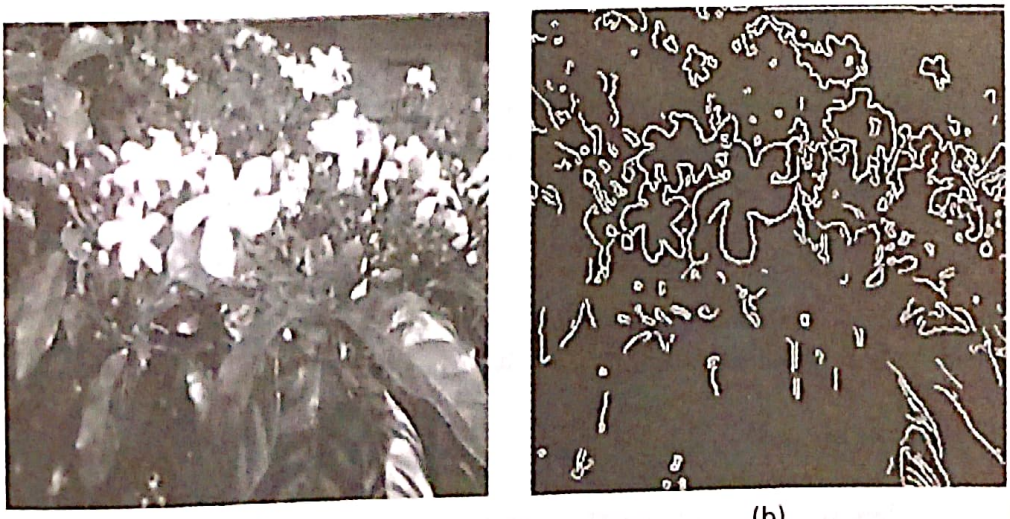
The Canny approach is based on optimizing the trade-off between two performance criteria and can be described as follows:

1. Good edge detection—The algorithm should detect only the real edge points and discard all false edge points.
2. Good edge localization—The algorithm should have the ability to produce edge points that are closer to the real edges.
3. Only one response to each edge—The algorithm should not produce any false, double, or spurious edges.

The Canny edge detection algorithm is given as follows:

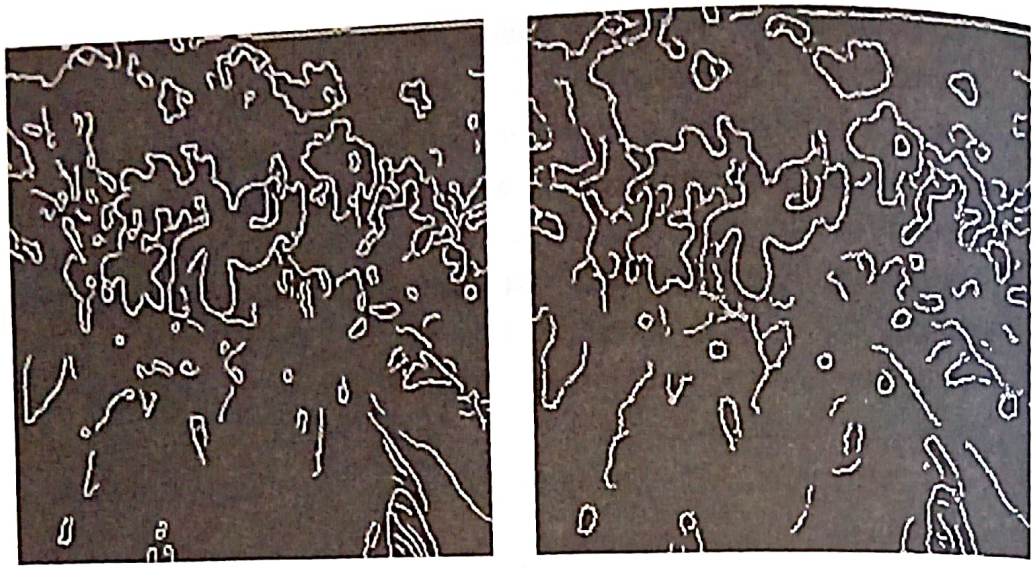
1. First convolve the image with the Gaussian filter. Compute the gradient of the resultant smooth image. Store the edge magnitude and edge orientation separately in two arrays, $M(x, y)$ and $\alpha(x, y)$, respectively.

2. The next step is to thin the edges. This is done using a process called *non-maxima suppression*. Examining every edge point orientation is a computationally intensive task. To avoid such intense computations, the gradient direction is reduced to just four sectors. How? The range of 0–360° is divided into eight equal portions. Two equal portions are designated as one sector. Therefore there will be four sectors. The gradient direction of the edge point is first approximated to one of these sectors. After the sector is finalized, let us assume a point of $M(x, y)$. The edge magnitudes $M(x_1, y_1)$ and $M(x_2, y_2)$, of two neighbouring pixels that fall on the same gradient direction, are considered. If the magnitude of the point $M(x, y)$ is less than the magnitude of the points (x_1, y_1) or (x_2, y_2), then the value is suppressed, that is, the value is set to zero; otherwise the value is retained.
3. Apply hysteresis thresholding. The idea behind hysteresis thresholding is that only a large amount of change in the gradient magnitude matters in edge detection and small changes do not affect the quality of edge detection. This method uses two thresholds, t_0 and t_1. If the gradient magnitude is greater than the value t_1, it is considered as a definite edge point and is accepted. Similarly, if the gradient magnitude is less than t_0, it is considered as a weak edge point and removed. However, if the edge gradient is between t_0 and t_1, it is considered as either weak or strong based on the context. This is implemented by creating two images using two thresholds t_0 and t_1. Low threshold creates a situation where noisier edge points are accepted. On the other hand, a high value of the threshold removes many potential edge points. So this process first thresholds the image with low and high thresholds to create two separate images. The image containing the high threshold image will contain edges, but gaps will be present. So the image created using the low threshold is consulted and its 8-neighbours are examined. So the gaps of the high threshold image are bridged using the edge points of the low threshold image. This process thus ensures that the edges are linked properly to generate a perfect contour of the image. The results of the Canny detector are shown in Figs 9.15(a)–9.15(d).



(a) (b)

Fig. 9.15 Canny edge detection (a) Original image (b) Canny edge detection at $\sigma = 1$



(c) (d)

Fig. 9.15 (c) Canny edge detection at $\sigma = 1.5$ (d) Canny edge detection at $\sigma = 2$

9.4.4.5 Pattern fit algorithm

Digital images are sampled, quantized versions of a continuous function. Instead of computing the desired properties in the discrete domain, it is better if the properties are measured in the continuous domain. Thus the approach of pattern fit algorithms towards edge detection is to reconstruct the continuous domain from the discrete domain. This ensures the detection of edges with more precision at the pixel level. However, construction of continuous domain from the discrete domain is a difficult process and so the reconstruction is attempted in a smaller neighborhood. The logic is to model the image as analytical functions such as biquadratic or bicubic equations. The individual functions are called facets.

Pattern fit algorithms model the image as a topographic surface where the pixel values represent the altitude. The aim is to fit a pattern over a neighbourhood of pixels where the edge strength is to be calculated. The properties of the edge points are calculated based on the parameters of the analytical functions.

A very simple technique is given by Hueckel to model a step edge. A step edge model models two surfaces with intensities α and β.

The step edge model is described as

$$h(x,y) = \begin{cases} \alpha \text{ if } x\sin\theta \geq y\cos\theta \\ \beta \text{ otherwise} \end{cases}$$

θ is the edge orientation. This model fits the step edge over the neighbours of the image. A 2×2 neighbourhood $s(x, y)$ is $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$. Pixels can be assumed to converge at the centre P. The idea is to minimize the error between the model $h(x, y)$ and the neighborhood

$s(x, y)$. The Hueckel approach is to expand $h(x, y)$ and $s(x, y)$ as a set of basis functions and use the sum of the squared differences between the corresponding coefficients as an error measure. The minimization of error helps in finding the parameters of the model.

This logic is extended to include the models that use the best plane fit method to approximate the surfaces. A small neighbourhood of 2×2 is considered. Let g_0, g_1, g_2, and g_3 be the grey values of the pixels in this neighbourhood. A simple pattern fit for the neighborhood can be a polynomial and is given as

$$v = \alpha_1 x + \alpha_2 y + \alpha_3$$

Let the error be a squared difference between the original and the approximated and can be calculated using a least square fit. The error is given as

$$\varepsilon = [a_1 x + a_2 y + a_3 - g_0]^2 + [a_1(x-1) + a_2 y + a_3 - g_1]^2$$
$$+ [a_1(x-1) + a_2(y-1) + a_3 - g_2]^2 + [a_1 x + a_2(y-1) + a_3 - g_3]^2$$

The objective of best plane fit is to minimize the error. This is carried out by finding the partial derivative and setting it to zero. Minimization and simplification of the error leads to

$$\alpha_1 = \frac{g_0 + g_3}{2} - \frac{g_1 + g_2}{2}$$

$$\alpha_2 = \frac{g_0 + g_1}{2} - \frac{g_2 + g_3}{2}$$

This is equivalent to the Roberts filter mask. Here, the edge gradient is given as

$$g'(x, y) = \sqrt{\alpha_1^2 + \alpha_2^2}$$

This approximates the Roberts gradient in the root mean square (RMS) way.

If the plane is estimated with grey scale values $g_i (i = 1, 2, \ldots, 8)$, then

$$\alpha_1 = \frac{1}{2}\left(\frac{g_4 + g_5 + g_6}{3} - \frac{g_1 + g_2 + g_3}{3}\right)$$

$$\alpha_2 = \frac{1}{2}\left(\frac{g_6 + g_7 + g_8}{3} - \frac{g_1 + g_2 + g_3}{3}\right)$$

This is equivalent to the Prewitt mask. This approximates the Prewitt mask by a factor of $\frac{1}{\sqrt{2}}$ in the RMS sense.

If the grey scale values g_1, g_3, g_5, and g_7 are weighted by 2 and the remaining coefficients by 1, we get

$$\alpha_1 = \frac{1}{2}\left(\frac{g_4 + 2g_5 + g_6}{6} - \frac{g_2 + 2g_1 + g_8}{6}\right)$$

$$\alpha_2 = \frac{1}{2}\left(\frac{g_6 + g_7 + g_8}{6} - \frac{g_2 + 2g_3 + g_4}{6}\right)$$

This is proved to be similar to the Sobel operator. This approximates the Sobel mask by a factor of $\frac{2\sqrt{2}}{3}$ in the RMS sense.

The best fit plane logic can be extended for complex images. One such generalized form of pattern fit using bicubic polynomial is called the Haralick facet model, which is useful for modelling complex images.

9.4.5 Edge Operator Performance

The performance of the edge detector algorithms should be evaluated as there is no guarantee that the edge detectors can detect all edges. Some of the problems of the edge detectors are as follows:

1. Missing valid edge points
2. Classifying the noise points as valid edge points
3. Smearing edges

Therefore, performance evaluation of edge detectors is very crucial. The performance evaluation techniques are of two types—subjective evaluation and objective evaluation.

Subjective evaluation techniques use the human visual system as the ultimate judge in evaluating the quality of the edge detector performance. Objective evaluation procedures, on the other hand, involve the quantification of the success and failure of the edge detectors' operational performance.

The Pratt figure of merit rating factor is one such objective evaluation procedure and is given as

$$R = \frac{1}{A}\sum_{i=1}^{D}\frac{1}{1+\alpha d^2}$$

D is the number of edge points detected by the edge operator. A is the maximum of ideal edge points present in the image and amongst the detected edge points (D). The distance between the original and the detected edge is characterized by the variable d. α is a parameter that should be tuned as a penalty for detecting false edges. The value of R is 1 for a perfect edge. The major problem with the objective metric is the quantization of the missing edges. Hence, in image processing applications, the objective metrics have limited practical use.

9.4.6 Edge Linking Algorithms

Edge detectors often do not produce continuous edges. Often, the detected edges are not sharp and continuous due to the presence of noise and intensity variations. Therefore, the idea of edge linking is to use the magnitude of the gradient operator to detect the presence of edges and to connect it to a neighbour to avoid breaks. Continuity is ensured by techniques

such as hysteresis thresholding and edge relaxation. The adjacent pixels (x, y) and (x', y') are connected if they have properties such as the following:

1. Similar gradient magnitude

$$\left\|\nabla f(x,y)\right\| - \left\|\nabla f(x',y')\right\| \leq T$$

2. Similar gradient orientation

$$\left|\phi(\nabla f(x,y) - \phi(\nabla f(x',y')\right| \leq A$$

where A is the angular threshold. Edge linking is a post-processing technique that is used to link edges.

The idea of using edge detection algorithms to extract the edges and using an appropriate threshold for combining them is known as *edge elements extraction by thresholding*. The threshold selection can be static, dynamic, or adaptive. These techniques are discussed in Section 9.7. Usage of edge detection, thresholding, and edge linking requires these algorithms to work interactively to ensure the continuity of the edges.

9.4.6.1 Edge relaxation

Edge relaxation is a process of re-evaluation of pixel classification using its context. The procedure takes a group of cracked edges and verifies the context of the edge in question. Cracks (or crack edges) are the differences between the pixels. The crack edges for the horizontal edges are $|f(x, y) - f(x + 1, y)|$ and for the vertical cracks are $|I(x, y) - I(x, y - 1)|$. For example, consider the image in Fig. 9.16(a) where the centre pixel is highlighted and underlined. Some of the cracks are shown in Figs 9.16(b); they result in the 3–3 crack shown in 9.16(c).

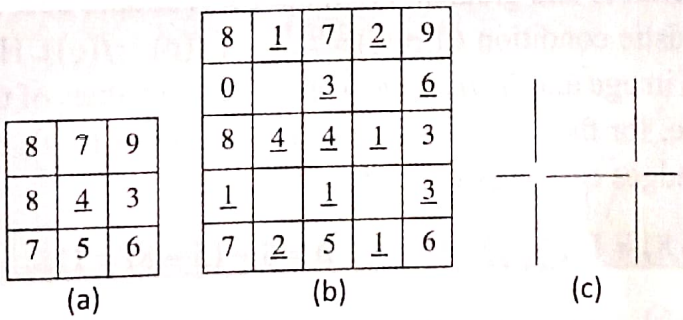


Fig. 9.16 Edge relaxation (a) Original image (b) Cracks with respect to the centre (c) The 3–3 crack edge

The cracks of the edges can be specified as a number pair. It can be observed that the centre is connected to three hanging edges on the left as well as the right. It can be observed that the pattern for the above crack is 3–3. It has been observed that certain cracks such as 1–1, 1–2, and 1–3 exercize a positive influence, and cracks such as 0–0, 0–2, and 0–3 exercize a negative influence. The influence of cracks such as 0–1, 2–2, and 3–3 is uncertain.

If a, b, and c are the values of the hanging cracks at one end, they can be sorted such that $a > b > c$. If N is the number of grey levels of the image, then m can be calculated as $m = N/10$. The algorithm for edge relaxation can be described as follows:

1. Assume an initial confidence level for the edge.
2. Classify the neighbouring edge crack types. It is called based on the value of m, a, b, and c. The edge type is type 0 if $(m - a)(m - b)(m - c)$, type 1 if $a(m - b)(m - c)$, type 2 if $ab(m - c)$, and type 4 if abc.
3. Update the confidence of the edge. Increment if the cracks are of the type (1, 1), (1, 2), (2, 1), (1, 3), or (3, 1), decrement if the cracks are of the type (0, 0), (0, 1), (0, 3), (2, 0), or (3, 0) and do nothing for other cracks.
4. Repeat steps 2–3 until the edge confidence becomes 0 or 1.

9.4.6.2 Graph theoretic algorithms

The graph theoretic approach is quite popular in machine vision as the problem is stated as the optimization of some criteria function. The idea is to construct a graph of an image where the nodes represent the pixel corners of the graph and the edges represent the pixel cracks. Each edge is assigned a cost based on the gradient operator. The cost of the edge is high for improbable edges and very low for potential edges such as cracks. The problem of finding an optimal boundary is considered as a problem of finding minimum cost paths. The components of this algorithm are as follows:

1. Forming the graph
2. Assigning cost functions
3. Identifying the start and end points
4. Finding the minimum cost path

Let us assume that the boundary is assumed to run from top to bottom. Then the root of the graph represents the top of the image. The successors are the corresponding vertical cracks, left horizontal, and right horizontal cracks. Cost functions are assigned based on the gradient operators and gradient orientations. A simple cost allocation mechanism is based on this heuristic condition $C(p, q) = f_{max} - (f(p) - f(q))$. Here f_{max} is the maximum pixel value of the image and $f(p)$ and $f(q)$ are the pixel values of the pixels p and q, respectively. For example, for the sample image shown in Fig. 9.17(a), the value of f_{max} is 8. Then the cost of the edges can be calculated as

$a = 8 - (2 - 8) = 14;$ $b = 8 - (5 - 8) = 11;$ $b' = 8 - (8 - 5) = 5;$

$c = 8 - (2 - 6) = 12;$ $d = 8 - (3 - 5) = 10;$ $e = 8 - (3 - 5) = 10;$

$e' = 8 - (5 - 3) = 6;$ $f = 8 - (5 - 4) = 7;$ $g = 8 - (4 - 3) = 7;$ $h = 8 - (3 - 2) = 7$

The starting point of this graph is the top of the image. In general, the start and end points can be given by the user or they can be special points called landmark points. The possible edge contour now becomes the problem of finding the minimal cost path. One way to find the path is to verify all the paths of the graph. This method is known as exhaustive

search. It is computationally highly intensive as the nodes that are already considered in the graph may be repeatedly revisited. Hence intelligent search algorithms such as the A* algorithms are used to solve this problem. A sample image and the graph constructed from it are shown in Figs 9.17(a) and 9.17(b).

The initial level graph is shown in Fig. 9.17(b). A* is an intelligent search algorithm. What is meant by an intelligent search algorithm? Exhaustive search algorithms may visit the same node repeatedly. Intelligent search algorithms are artificial intelligence techniques that use heuristic functions to guide the search. The process involves selecting a node and expanding it by using a searching function. To keep track of the nodes having been visited, two lists are created. One list is called the open list, which indicates the nodes that are sorted according to minimum cost with their status. The other, called the closed list consists of nodes that will be not be expanded again. The search process is controlled by a heuristic function. A heuristic strategy is one that normally works but also occasionally fails. If it fails, one needs to change the heuristics and the process will be repeated. The heuristic function indicates the work that must be done to reach the goal. Typically a search function is given as

$$f(n) = g(n) + h(n)$$

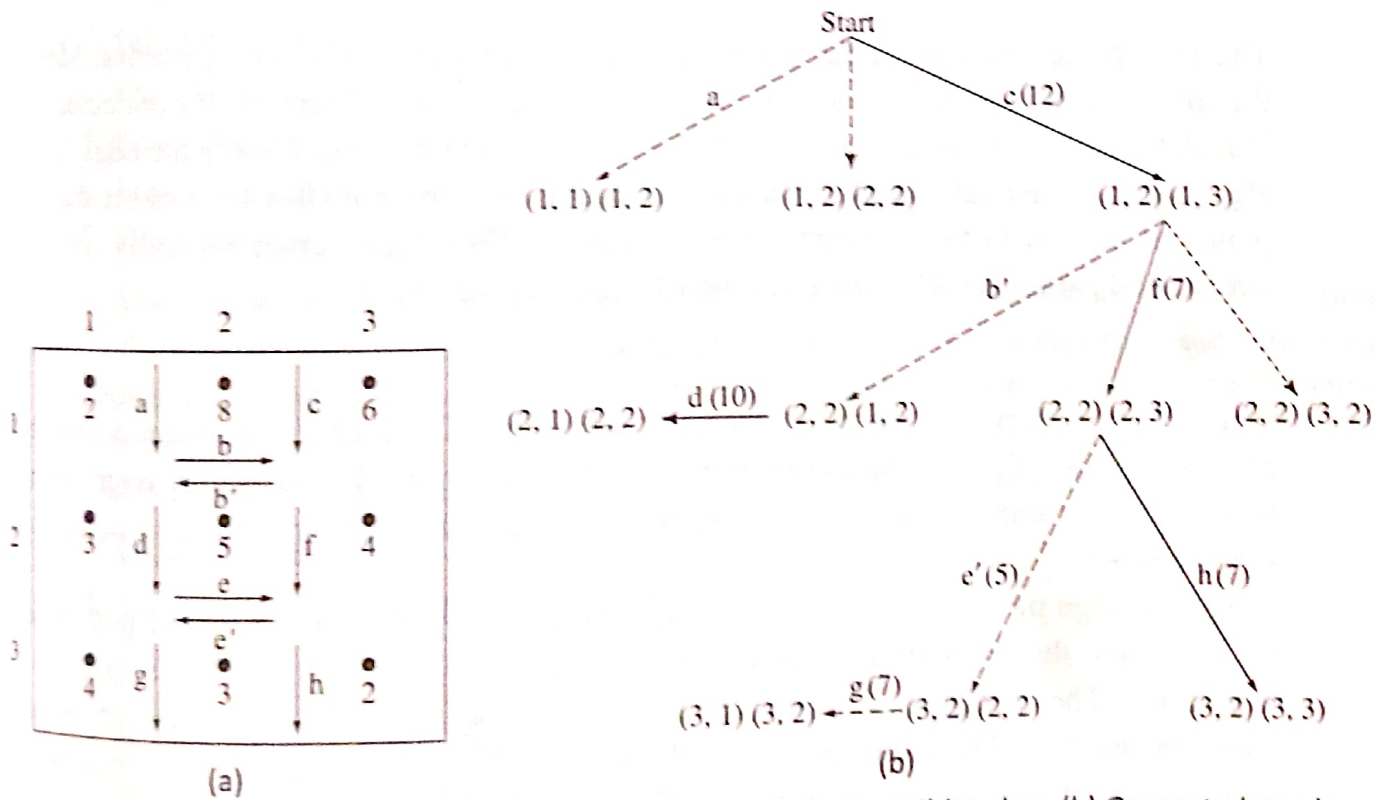


Fig. 9.17 Graph theoretic algorithms (a) Sample image and possible edges (b) Generated graph

$g(n)$ is the cost function that has been designed and $h(n)$ is the heuristic function, which also indicates the path length. $f(n)$ indicates the total effort to reach the goal.

The algorithm is given as follows:

1. Start from the start node. For this situation, the edge is assumed to flow from top to bottom. So edges a and c are the only possibilities. Place the nodes in the open list.
2. Let n be the first node of the open list.
3. If the open list is empty then exit, otherwise extract the node n.
4. If node n is the goal, then exit.
5. Otherwise,
   (a) Expand n
   (b) Evaluate $f(n)$ for all the children
   (c) Sort them in the order of minimum cost
   (d) Place them in the open list and preserve the order
6. Repeat steps 2–5 till the open list becomes empty.

The minimum cost or the shortest path is obtained and it is assumed as the edge. The edge is shown as a thick line in the graph in Fig. 9.17. The advantage of this approach is that it is immune to noise and is very robust. However, construction of the graph for bigger images is a computationally intensive task and is not very practical.

9.5 HOUGH TRANSFORMS AND SHAPE DETECTION

The Hough transform takes the images created by the edge detection operators. Most of the time, the edge map generated by the edge detection algorithms is disconnected. The Hough transform can be used to connect the disjointed edge points. Unlike the edge linking algorithms, the Hough transform does not require any prerequisites to connect the edge points. It is used to fit the points as plane curves. The plane curves typically are lines, circles, and parabolas. The line equation of a line is given as

$$y = mx + c$$

where m is the slope and c is the y-intercept of the line. However, the problem is that there are infinite lines that can be drawn connecting these points. Therefore, an edge point in an x–y plane is transformed to a c–m plane. The trick here is to write the line equation as $c = (-x)m + b$.

All the edge points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ in the x–y plane need to be fitted. Hence, the x–y plane should be transformed into a different c–m plane. All points are lines in the c–m plane. The objective is to find the intersection point. A common intersection point indicates that the edge points are part of the same line. If A and B are points connected by a line in the spatial domain, then they will be intersecting lines in the Hough space. This is illustrated in Fig. 9.18. To check whether they are intersecting lines, the c–m plane is partitioned as accumulator lines. To find this, it can be assumed that the c–m plane can be partitioned as an accumulator array. For every edge point (x, y), the corresponding accumulator element is incremented in the accumulator array. At the end of this process,

the accumulator values are checked. The largest value of the accumulator is called (m', c'). The significance is that this point gives us the line equation of the (x, y) space:

$$y = m'x + c'$$

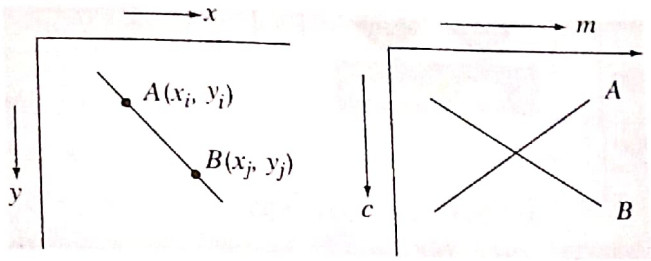


Fig. 9.18 Line in Hough transform

The Hough transform can be stated as follows:

1. Load the image.
2. Find the edges of the image using any edge detector.
3. Quantize the parameter space P.
4. Repeat the following for all the pixels of the image:
   If the pixel is an edge pixel, then
   (a) $c = (-x)m + y$
   (b) $P(c.m) = P(c.m) + 1$
5. Show the Hough space.
6. Find the local maxima in the parameter space.
7. Draw the line using the local maxima.

The major problem with this algorithm is that it does not work for vertical lines, as they have a slope of infinity. Therefore, it is better to convert this line into polar coordinates. The line in polar form can be represented as $\rho = x\cos\theta + y\sin\theta$, where θ is the angle between the line and the x-axis, and ρ is the diameter.
The modified Hough transform for the polar coordinate line is as follows:

1. Load the image.
2. Find the edges of the image using any edge detector.
3. Quantize the parameter space P.
4. Repeat the following for all the pixels of the image:
   If the pixel is an edge pixel, then for all θ
   (a) Calculate ρ for the pixel (x, y) and θ.
   (b) Increment the position (ρ, θ) in the accumulator array P.
5. Show the Hough space.
6. Find the local maxima in the parameter space.
7. Draw the line using the local maxima.

The result of the Hough transform line detection is shown in Figs 9.19(a)–9.19(d). The detected lines have been highlighted.

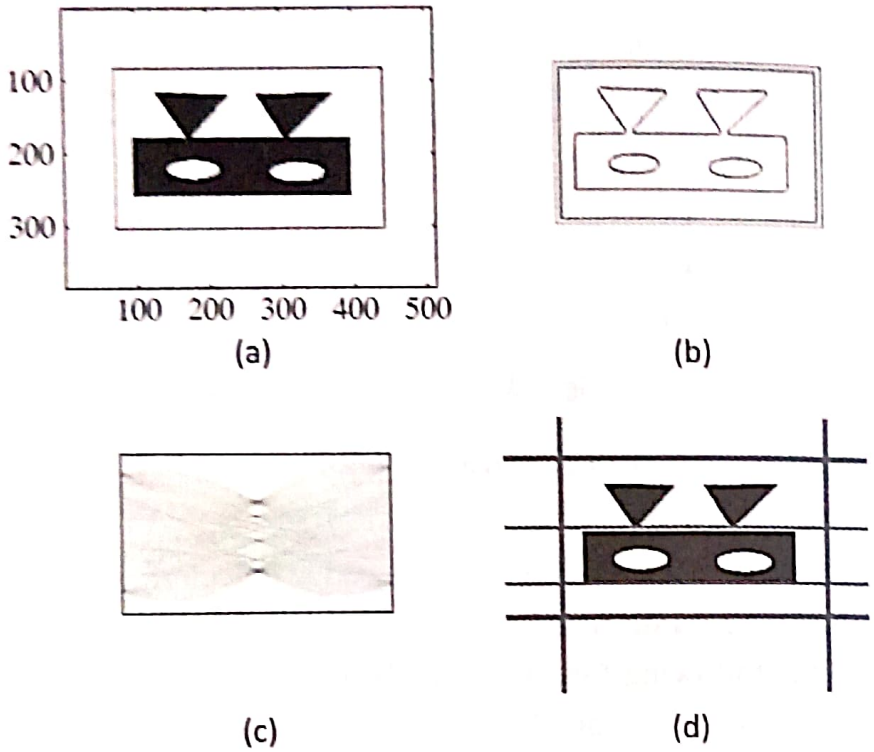


Fig. 9.19 Hough transform line detection results (a) Input image (b) Output edges (c) Accumulator image (d) Detected lines

Similarly, the Hough transform for other shapes can also be found. The Hough transform for a circle detection can be given as follows:

$$(x - x_0)^2 + (y - y_0)^2 - r^2 = 0$$

The parameter space is three dimensional as the unknowns are x_0, y_0, and r.

Similarly, the polar form circle detection algorithm can be written as follows:

1. Load the image.
2. Find the edges of the image using any edge detector.
3. Quantize the parameter space P.
4. Repeat the following for all the pixels of the image:
   If the pixel is an edge pixel, then for all values of r, calculate
   (a) $x_0 = x - r\cos\theta$
   (b) $y_0 = y - r\sin\theta$
   (c) $P(x_0, y_0, r) = P(x_0, y_0, r) + 1$
5. Show the Hough space.
6. Find the local maxima in the parameter space.
7. Draw the circle using the local maxima.

In this manner, a circle can be fit to cover the edge points. The Hough transform is a powerful algorithm and can be modified to fit any generalized shape having disjointed edge points.

6 CORNER DETECTION

A corner is an important feature of an image, which indicates some important points of the image known as landmark points. It is formed at the boundary when two bright regions meet, where the boundary curvature is very high. A corner point exhibits the characteristic of having large variations in all directions, unlike edges, where the variation is in only one direction.

Corner points are also called interest points because the human visual system quickly identifies landmark points, such as corners, high-contrast regions, and edges. Corners are very useful in many imaging applications such as tracking, measurement systems, aligning, and stereo imaging. Figure 9.20 shows the corners indicated by arrow marks.

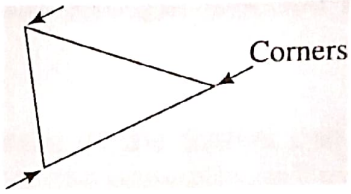


Fig. 9.20 Corner points

One of the earliest corner detection algorithms is called Moravec corner detector which calculates the edge strength in four directions. The minimum of the four directional responses is chosen and the threshold process is applied. Then non-maximal suppression is applied to extract a set of corner points. However, Moravec corner detector is not rotationally invariant and is sensitive to small variations along the edge.

6.1 Harris Corner Detector

A popular corner detector is the Harris corner detector. This detector improves Moravec corner detector using the autocorrelation function to find out the differential of the corner score unit with respect to direction directly, and using it for corner detection. This is done without using shifted patches. The advantages of Harris corner detector include corner detection irrespective of different sampling quantization, small changes in scales, and affine transformations.

The basis of this algorithm is the autocorrelation function where a small square window is placed around each pixel and the window is shifted in all eight directions. Then, autocorrelation is performed. If there is no variation, then the region is a patch where all the pixels have same intensity. If the autocorrelation function produces a large variation, then the pixel is marked as an interest point. Thus, autocorrelation function is the key for detection of parches, edges and corners.

If the original point is (x, y) and the shift of the window is Δx and Δy, in x and y directions, the autocorrelation function or corner score is defined as

$$c(x, y) = \sum_{W} [I(x_i, y_i) - I(x_i + \Delta x, y_i + \Delta y)]^2$$

Here, $I(x_i, y_i)$ is the image function and $I(x_i + \Delta x, y_i + \Delta y)$ is the shifted image. The shifted image can be approximated by Taylor expansion as

$$I(x_i + \Delta x, y_i + \Delta y) \cong I(x_i, y_i) + [I_x(x_i, y_i), I_y(x_i, y_i)]\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

Here, $$I_x(x, y) = \frac{\partial I}{\partial x}(x, y), \quad I_y(x, y) = \frac{\partial I}{\partial y}(x, y)$$

This expression when substituted in the autocorrelation function yields

$$c(x, y) = \sum_W I(x_i, y_i) - (I(x_i, y_i) + I(x_i + \Delta x, y_i + \Delta y)^2$$

$$c(x, y) = \sum_W I(x_i, y_i) - [(I(x_i, y_i) + I_x(x_i, y_i), I_y(x_i, y_i)]\left(\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}\right)^2$$

$$c(x, y) = \sum_W (I(x_i, y_i) - I(x_i, y_i) - [I_x(x_i, y_i) I_y(x_i, y_i)]\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}^2$$

$$c(x, y) = \sum_W -\left([I_x(x_i, y_i) I_y(x_i, y_i)]\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}\right)^2$$

$$c(x, y) = \sum_W +\left([I_x(x_i, y_i) I_y(x_i, y_i)]\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}\right)^2$$

$$c(x, y) = \sum_W (\Delta x I_x(x_i, y_i) + \Delta y I_y(x_i, y_i))^2$$

$$c(x, y) = \sum_W I_x^2(x_i, y_i)(\Delta x)^2 + 2 I_x(x_i, y_i) I_y(x_i, y_i) \Delta x \Delta y + I_y^2(x_i, y_i)(\Delta y)^2$$

Any quadratic equation of the form $F(x, y) = Ax^2 + 2Bxy + Cy^2$ can be expressed in matrix form as $\boldsymbol{F = vMv^T}$, where $\boldsymbol{v} = [x\ y]$ and $\boldsymbol{M}$ is a matrix written as $\begin{pmatrix} A & B \\ B & C \end{pmatrix}$. Accordingly, the autocorrelation function $\boldsymbol{c(x, y)}$ is given as

$$c(x, y) = [\Delta x \Delta y]\begin{pmatrix} A & B \\ B & C \end{pmatrix}\begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

where

$$A(x, y) = \sum_W I_x{}^2(x_i, y_i), \quad B(x, y) = \sum_W I_x(x, y) I_y(x, y), \text{ and } C(x, y) = \sum_W I_y^2(x, y)$$

The autocorrelation function c captures the intensity structure of the neighborhood. In addition, it is smoothed by the linear Gaussian filter. The Gaussian-smoothed matrix $\boldsymbol{R}$ is represented as

$$\boldsymbol{R} = \begin{pmatrix} A & B \\ B & C \end{pmatrix}$$

The Eigen values of this matrix $\boldsymbol{R}$ characterize edge strength, and the Eigen vectors represent the edge orientation. Assume that λ_1 and λ_2 be the Eigen values, then

1. If the values of λ_1 and λ_2 are too small, it implies that the window region is flat.
2. If λ_1 is high and λ_2 is small or vice versa, it is an edge.
3. If both λ_1 and λ_2 are high, then it indicates the presence of a corner.

In general, the corner strength can be characterized as a corner strength function and represented as

$$Q(u, v) = \det(\boldsymbol{R}) - k \times (\text{trace } (\boldsymbol{R})^2)$$

The computation of $Q(u,v)$ is easy as the determinant and trace of the matrix can be expressed in the form of the Eigen values as

$$\det(\boldsymbol{R}) = \prod_{i=1}^{n} \lambda_i \text{ and } \text{trace}(\boldsymbol{R}) = \sum_{i=1}^{n} \lambda_i$$

The algorithm for Harris corner detection is as follows:

1. Select an image location (x, y).
2. Identify it as a corner point when corner strength $Q(u, v)$ > threshold value. Define a window and apply autocorrelation function.
3. Insert into a list of corner points sorted by the corner strength.
4. Eliminate the false corners when a weaker corner point lies near a stronger corner.
5. Display all the corner points.
6. Exit.

9.7 PRINCIPLE OF THRESHOLDING

Thresholding is a very important technique for image segmentation. It produces uniform regions based on the threshold criterion T. The thresholding operation can be thought of as an operation, such as $T = T\{x, y, A(x, y), f(x, y)\}$, where $f(x, y)$ is the grey level of the pixel at (x, y) and $A(x, y)$ is the local property of the image. If the thresholding operation depends only on the grey scale value, it is called global thresholding. If the neighbourhood property is also taken into account, this method is called local thresholding. If T depends on pixel coordinates also, T is called dynamic thresholding.

9.7.1 Histogram and Threshold

The quality of the thresholding algorithm depends on the selection of a suitable threshold. Figures 9.21(a)–9.21(d) show the effect of the threshold value on an image. The selection of an appropriate threshold is a difficult process.

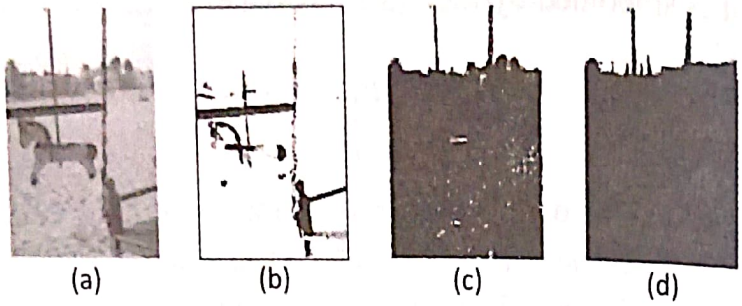


Fig. 9.21 Effect of threshold on images (a) Original image (b) Image at low threshold value (c) Image at optimal threshold value (c) Image at high threshold value

The tool that helps to find the threshold is histogram. Histograms are of two types:

1. Unimodal histogram
2. Multimodal histogram

If a histogram has one central peak, it is called unimodal histogram. On the other hand, a multimodal histogram has multiple peaks. A bimodal histogram is a special kind of histogram that has two peaks separated by a valley. Using the valley, a suitable threshold can be selected to segment the image. Thus the peak (also known as local maxima) and the valley between the peaks are indicators for selecting the threshold. This process is difficult for unimodal algorithms and more difficult if the foreground and background pixels overlap as shown in Fig. 9.22. Some of the techniques that can be followed for selecting the threshold values are as follows:

1. Random selection of the threshold value
2. If the ROI is brighter than the background object, then find the cumulative histogram. Histogram is a probability distribution $p(g) = n_g/n$ where n_g is the number of pixels having the grey scale value as g and n is the total number of pixels. The cumulative histogram is

$$c(g) = \sum_{i=0}^{g} p(g).$$

Set the threshold T such that $c(T) = 1/p$. If we are looking for dark objects in white background, the threshold would be $1 - (1/p)$. Some of these effects are shown in Figs 9.22(a)–9.22(c).

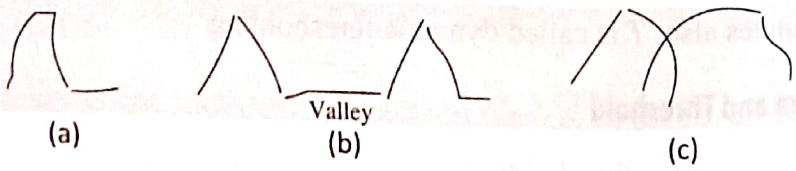


Fig. 9.22 Some examples of histograms (a) Unimodal histogram (b) Bimodal histogram (c) Histogram showing overlapping regions between foreground and background objects

Selecting the threshold for bimodal images is an easy task since the valley can indicate the threshold value. The process is difficult for unimodal histograms and too difficult if the background and foreground pixels overlap.

Noise can affect histograms. Noise can create a spike and the presence of data with two different distributions may cause problems. The presence of data belonging to different distributions may produce a histogram with no distinct nodes, which may complicate the process of finding thresholds.

Global thresholding is difficult when the image contrast is very poor. The background of the image can cause a huge problem too. In addition, textures and colours tend to affect the thresholding process. Similarly, if the lighting conditions are not good, thresholding may yield poor results.

9.7.1.1 Effect of noise over threshold process and peakiness test

Noise can create artificial peaks in the histogram. The peakiness test can be used to check the genuine nature of the peak. A true peak is characterized by its sharpness and depth. Sharpness means that the area is small. The depth is the relative height of a peak. Figure 9.23 illustrates a histogram with a peak.

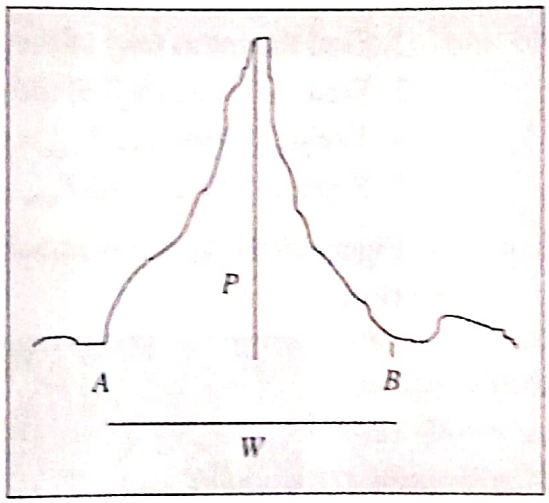


Fig. 9.23 A sample histogram peak

Let W be the width of the peak ranging from valley to valley. Let P be the height of the peak and let the valley points be A and B on both sides of the peak. The sharpness of the peak can then be defined as $\frac{N}{(W \times P)}$. Here N is the number of pixels in the image that is covered by the height P. If the ratio is small, it is better. The peakiness then can be defined in terms of peak sharpness as

$$\text{Peakiness} = \left(1 - \frac{A+B}{2P}\right) \times (1 - \text{Peak sharpness})$$

If the peakiness is above a certain threshold T, the peak can be used for segmentation purposes, otherwise the peak of the histogram is rejected. A modified segmentation algorithm can be specified by incorporating the peakiness test, as follows:

1. Compute the histogram.
2. Smooth the histogram by averaging. This step will remove the small peaks.
3. Identify the candidate peaks and valleys in the histogram. Detect the good peaks by applying the peakiness test. Remove all false peaks created by noise. Then segment the image using the thresholds at the valleys.

9.7.2 Global Thresholding Algorithms

Bimodal images are those where the histograms have two distinct peaks separated by a valley between them. The valley point is chosen as a threshold T. Then the pixels of the given image ($f(x, y)$) are compared with the threshold. If the pixel values are greater than or equal to the threshold value, the pixel is assigned a value of 1. Otherwise, it is assigned a value of 0, giving the output threshold image $g(x, y)$.

The threshold process is given as

$$g(x, y) = \begin{cases} 1 \text{ if } f(x, y) \geq T \\ 0 \text{ otherwise} \end{cases}$$

One of the biggest problems is choosing an appropriate threshold value. Figure 9.21 illustrated the effect of threshold values.

1. Choose an initial threshold value T randomly, say, 128.
2. Find the mean (m_1) of the pixels that lie below T in the histogram.
3. Find the mean (m_2) of the pixels that lie above T in the histogram.
4. The new threshold $T_{new} = (m_1 + m_2)/2$.
5. Repeat steps 2–4 till T_{new} no longer changes.

Figures 9.24(a) and 9.24(b) show an original image and the global thresholding result, respectively.

(a) (b)

Fig. 9.24 Global thresholding algorithm (a) Original image (b) Threshold image

9.7.3 Multiple Thresholding

If there are more than two classes of objects in the image, multiple thresholding should be used. This method is an extension of the simple thresholding technique. Here f_i is the value of the input image pixel and $t_1, t_2, \ldots, t_n$ are multiple threshold values. The values of the output image are given as $g_1, g_2, \ldots, g_n$.

$$\text{Output image} = \begin{cases} g_1 \text{ if } 0 \le f_i \le t_1 \\ g_2 \text{ if } t_1 < f_i \le t_2 \\ \quad \cdots \\ g_n \text{ if } t_{n-1} < f_i \le 255 \end{cases}$$

9.7.4 Adaptive Thresholding Algorithm

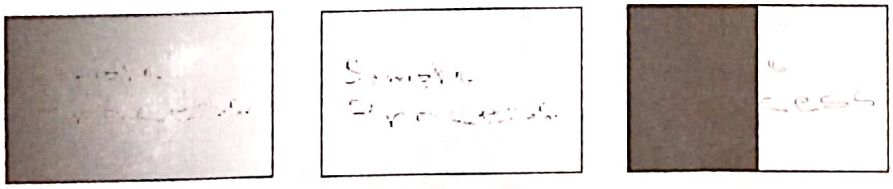
Adaptive algorithm is also known as dynamic thresholding algorithm. Ideally in dynamic thresholding, the image is divided into many overlapping sub-images. The histograms of all the sub-images are constructed and the local thresholds are obtained. Then the threshold value is obtained by interpolating the results of the sub-images. However, the problem with this approach is that it is computationally very intensive and takes a lot of time. Hence, this approach is not suitable for real applications.

Another way of applying adaptive thresholding is to split the image into many subregions. The following image statistics can then be calculated; the idea is to locally apply these for the subregions. Some useful image statistics are as follows:

$$1.\ \text{Mean} + c \qquad 2.\ \text{Median} + c \qquad 3.\ \frac{\text{Max} + \text{Min}}{2}$$

Max and Min correspond to the maximum and minimum of pixel values, respectively, and c is a constant.

If the subregions are large enough to cover the foreground and background areas then this approach works well. In addition, adaptive thresholding is good in situations where the image is affected by non-uniform illumination problems. Figure 9.25(a) shows an original image with non-uniform illumination. The result of the adaptive algorithm is shown in Fig. 9.25(b). It can be observed that this result is far better than the result provided by the global thresholding algorithm for Fig. 9.25(a). The reason is that global thresholding algorithms do not work well if the image has illumination problems.



(a) (b) (c)

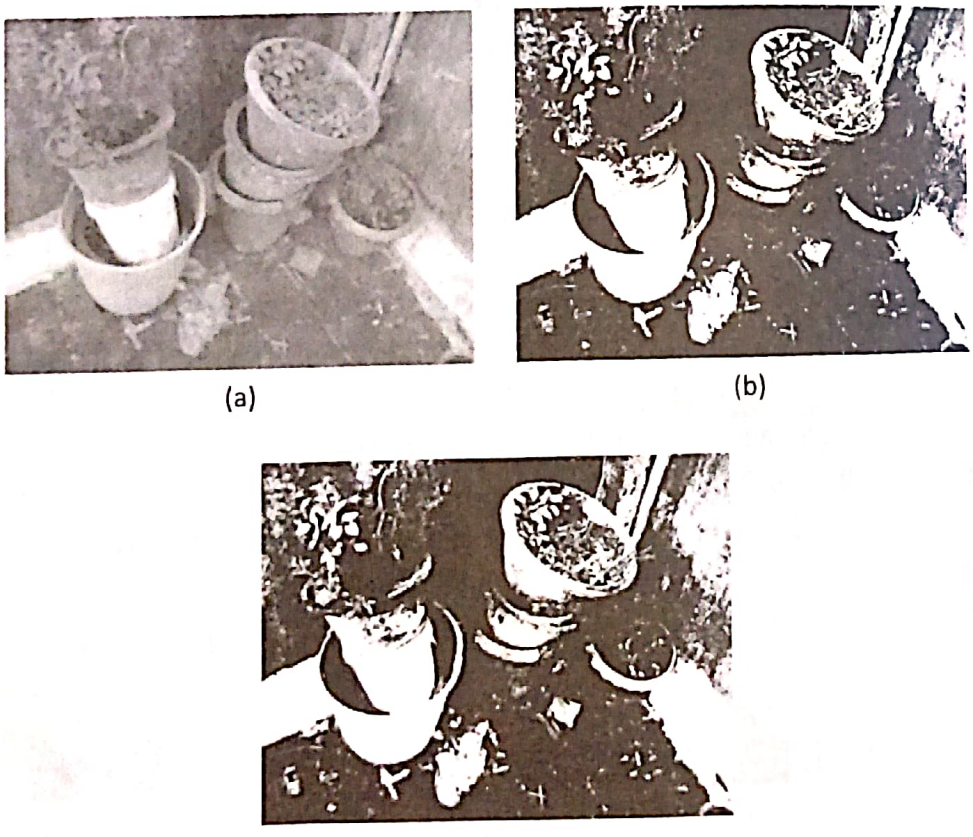
Fig. 9.25 Comparison of thresholding algorithms (a) Original image (b) Result of adaptive algorithm (c) Result of global thresholding algorithm

9.7.5 Optimal Otsu Thresholding Algorithms

There are many situations where the boundary between the foreground object and the background object is not clear. If there is a considerable overlapping of the histogram, then optimal thresholding is used. Optimal thresholding uses a merit function of either maximization or minimization to determine the optimal threshold.

Otsu method uses the goodness criteria for threshold selection. These methods use within-class variance and between-class variance to select an optimal threshold. The Otsu method selects the threshold value based on the criteria $\frac{\sigma_B^2}{\sigma_T^2}$, where σ_T^2 is the total variance.

The optimal thresholding algorithm selects the threshold value optimally based on the characteristics of the image data. Assume that two group of pixels overlap. If only one combined histogram is available, then finding the optimal threshold value becomes a difficult task. It is assumed that the foreground and background belong to two classes or clusters. The aim is to pick a threshold such that the pixels on either side of the threshold are close in value to the mean of pixels of that side. In addition, the mean of one side should be different from the mean of the other side. This algorithm proceeds automatically and is said to be unsupervised. This algorithm is called Otsu algorithm and the results of the algorithm are shown in Fig. 9.26.



(a) (b) (c)

Fig. 9.26 Otsu thresholding algorithm (a) Original image (b) Global threshold result (c) Otsu algorithm result

Assume that there are two classes—class 1 with weight $w_1(t)$ and variance $\sigma_1^2(t)$ and class 2 with weight $w_2(t)$ and variance $\sigma_2^2(t)$. Then the weighted sum of the variance is

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t)$$

It can be observed that the threshold with the maximum *between-class variance* also has the minimum *within-class variance*. So the equation can be rewritten as

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t)$$

Using the fact that $\mu = W_b\mu_b + W_f\mu_f$, the relation can be reduced to

$$w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2$$

This leads to faster computation.

The algorithm can be stated as follows:

1. Load the image.
2. Form the histogram.
3. Calculate the probability of every level.
4. Initialize $w_i(0)$ and $\mu_i(0)$.
5. For every threshold value ranging from one to the maximum intensity, update the w_i and μ_i values and compute $\sigma_b^2(t)$.
6. Find the maximum $\sigma_b^2(t)$. The corresponding threshold value is the optimal threshold value.

9.8 PRINCIPLE OF REGION-GROWING

One of the major disadvantages of the thresholding algorithms is that they still produce isolated regions. So it is necessary to process the segmented image to produce a coherent region. The rationale behind region-growing algorithms is the principle of similarity. The principle of similarity states that a region is coherent if all the pixels of that region are homogeneous with respect to some characteristics such as colour, intensity, texture, or other statistical properties. Thus the idea is to pick a pixel inside a region of interest as a starting point (also known as seed point) and allowing it to grow. By this it is meant that the seed point is compared with its neighbours, and if the properties match, they are merged together. This process is repeated till the regions converge to an extent that no further merging is possible. This results in the segmentation of the region of interest.

9.8.1 Region-growing Algorithm

Region-growing is a process of grouping the pixels or subregions to get a bigger region present in an image. The important issues of the region-growing algorithm are the selection of the initial seed, seed growing criteria, and termination of the segmentation process.

Selection of the initial seed This is a very important issue. The initial seed that represents the ROI should be given typically by the user. The initial seed can also be chosen automatically. The seeds can be either single or multiple.

Seed growing criteria The initial seed grows by the similarity criterion if the seed is similar to its neighbourhood pixels. Similarity criterion denotes the minimum difference in the grey levels or the average of the set of pixels and can apply to grey level, texture, or colour. Thus, the initial seed 'grows' by adding the neighbours if they share the same properties as the initial seed.

Termination of the segmentation process The rule for stopping the growing process is also very important as the region-growing algorithms are computationally very intensive operations and may not terminate easily.

In addition, the other important issues are usage of multiple threshold values and the order of the merging process.

Example 9.2 Consider the image shown in Fig. 9.27. Show the results of the region-growing algorithm.

Solution Assume that the seed points are indicated at the coordinates (2, 4) and (4, 2). The seed pixels are 9 and 1 (underlined in Fig. 9.27). Let them be $s1$ and $s2$. Let the seed pixels 1 and 9 represent the regions C and D, respectively.

Subtract the pixel value from the seed value that represents the region. If the difference between the pixel value and seed point is 4 (i.e., $T = 4$), merge the pixel with that region. Otherwise, merge the pixel with the other region. The resultant image is shown in Fig. 9.28.

1	0	7	8	7
0	1	8	9	8
0	0	7	9	8
0	1	8	8	9
1	2	8	8	9

Fig. 9.27 Region-growing algorithm—original image

C	C	D	D	D
C	C	D	D	D
C	C	D	D	D
C	C	D	D	D
C	C	D	D	D

Fig. 9.28 Region-growing algorithm—resultant image for Example 9.2

9.8.2 Split-and-merge Algorithm

The region-growing algorithm having a single pixel as the seed point is very slow. So the idea of a seed point can be extended to a seed region. Regional Adjacency Graph (RAG), discussed in Box 9.1, can be used to model the adjacency relations of an image. RAG is a graph data structure which consists of a set of nodes $N = \{N_1, N_2, \ldots, N_m\}$ where nodes represent the regions and the edges represent the adjacency relations among the regions.

An example of an original image and its corresponding RAG is shown in Figs 9.30(a) and 9.30(b), respectively.

Region adjacency graph

Region adjacency graph (RAG) is a data structure that is used to represent regions. It is a variant of the graph data structure $G = (V, E)$, which is useful for representing images. Here V is a set of nodes, that is, $V = \{v_1, v_2, \ldots, v_n\}$ and E is a set of edges, that is, $E = \{e_1, e_2, \ldots, e_n\}$.

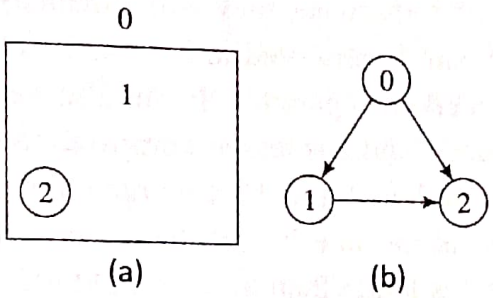


Fig. 9.29 Region adjacency graph (a) Sample image (b) RAG for the sample image

A typical data structure, as shown in Fig. 9.29, is a RAG where the nodes correspond to the regions. The relationship between the regions is represented by the arc or edge. A region is a set of points that share similar properties such as brightness, texture, and colour. Here, the label 0 denotes pixels of the entire image.

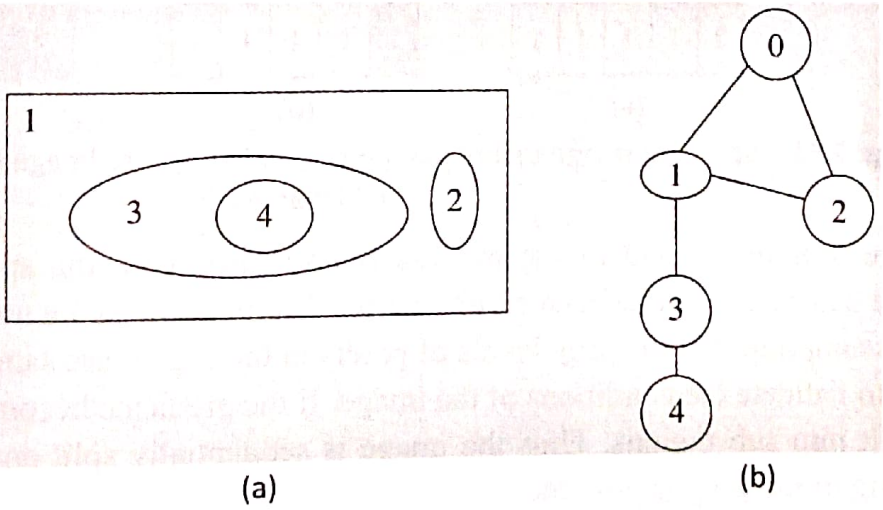


Fig. 9.30 Regional adjacency graph (a) Original image (b) RAG of the original image

Instead of a single pixel, a node of the RAG, that is, a region itself is now considered as a starting point. The split process can be stated as follows:

1. Segment the image into regions $R_1, R_2, \ldots, R_n$ using a set of thresholds.
2. Create a RAG. Use a similarity measure and formulate a homogeneity test.

3. For the regions R_i and R_j of the RAG, apply the homogeneity test. The homogeneity test is designed based on the similarity criteria such as intensity or any image statistics. If the similarity measure (S_{ij}) is greater than the threshold T, merge R_i and R_j. The threshold value can be determined using any known technique.
4. Repeat step 3 until no further region exists that requires merging.

Consider the image shown in Fig. 9.31(a). It has three grey scale values—2, 8, and 1. Assume a hypothetical threshold value of 3, chosen randomly. As per the algorithm, if the difference between the pixels of two regions is less than the assumed threshold value, the pixels can be merged. Otherwise, they will remain as separate regions. Consider the image given in 9.31(a); it can be observed in Fig. 9.31(b) that there are three regions with pixel values 1, 2, and 8 marked separately. It can also be observed that the difference between regions having values 1 and 2 is less as compared to the threshold value of 3. Thus the two regions having pixel values 1 and 2 are merged as a single region as shown in Fig. 9.31(c). The isolated region is the one having the pixel value 8. This is because the difference between 8 and 1 or 2 is larger than the threshold value.

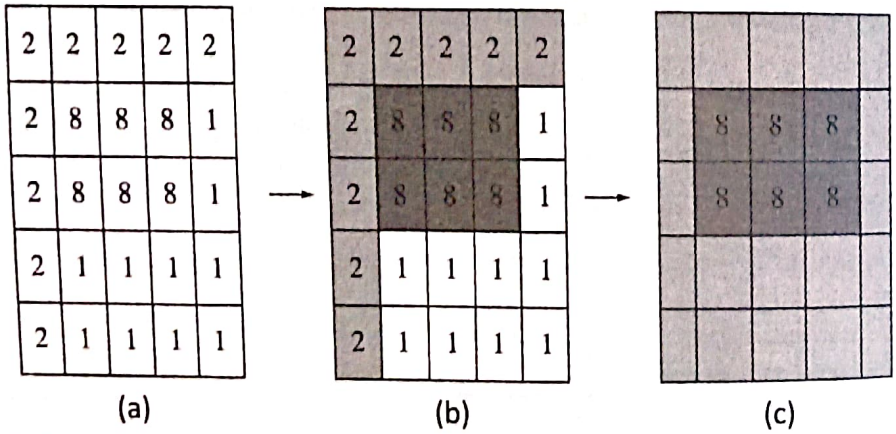


Fig. 9.31 Split-and-merge technique (a) Original image (b) Image showing separate regions (c) Final result

The aforementioned merge process is combined with the split process. Initially, an image can be considered as a whole region. A simple predicate may be designed based on the assumption that the grey levels of pixels in the region are same. This predicate can be used to validate the conditions of the image. If the predicate becomes false, then the region is split into sub-regions. Thus the image is sequentially split and merged till no further spliting or merging is possible.

9.8.3 Split-and-merge Algorithm using Pyramid Quadtree

Initially, the entire image is assumed as a single region. Then the homogeneity test is applied. If the conditions of homogeneity are not met, then the regions are split into four quadrants. This process is repeated for each quadrant until all the regions meet the required homogeneity criteria. If the regions are too small, then the division process is

stopped. Similarly the regions are merged if they share some common properties. Many other techniques are also available. In seed competition technique, regions compete for pixels. So a pixel can be reassigned to another region if it closer to that region. Similarly, the boundary melting technique also can be applied. Using the edge strength, the regions can be merged if the edges are weak. This process of splitting and merging is repeated till there is no observable change. This method is dependent on the problem and the segmentation results are very sensitive to the initial choice.

Quadtree, discussed in Box 9.2, is a useful data structure for this algorithm. The root of the tree corresponds to the entire image. The leaves correspond to the individual regions. The root generates four children at every level. If all the four children have sample property, then it is a finished tree and is pruned. If the children differ in property, the parent adopts the average value of all the four children. This is a non-terminal node. It means that the node is further generated and the process is repeated to the lowest level.

Quadtrees

A quadtree is a tree, except the leaves, has four children. It is helpful in representing the regions of images, where an image is divided into four quadrants at each hierarchical level. An image and its corresponding quadtree are shown in Figs 9.32(a) and 9.32(b). Quadrant A has four children, which are indicated in the quadtree. All other quadrants B, C, and D have the same colour property and therefore can be shown collectively. Hence, there is no need to represent the children.

The advantage of quadtrees is that they help in creating simpler algorithms. However, the disadvantage is that they are dependent on the position, orientation, and relative size of the objects.

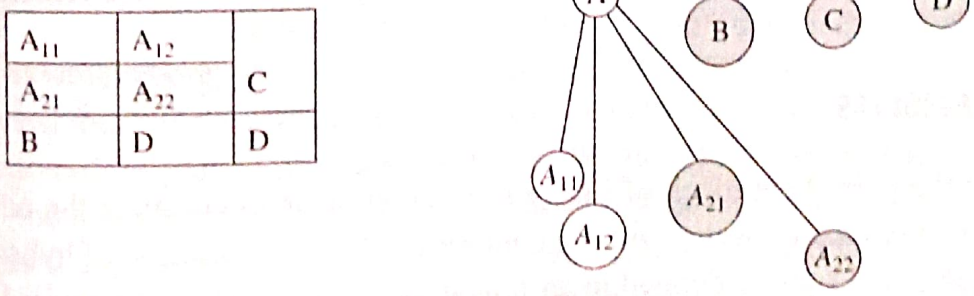


Fig. 9.32 (a) Sample image (b) Quadtree

The algorithm is stated in two phases as follows:

Phase 1: Split and continue the subdivision process until some stopping criteria is fulfilled. The stopping criteria often occur at a stage where no further splitting is possible.

Phase 2: Merge adjacent regions if the regions share any common criteria. Stop the process when no further merging is possible.

Assume the goal is to segment the shaded region in the graph shown in Fig. 9.33(a). Then the algorithm starts assuming the entire image as a whole region. The quadtree representation is shown in Fig. 9.33(b).

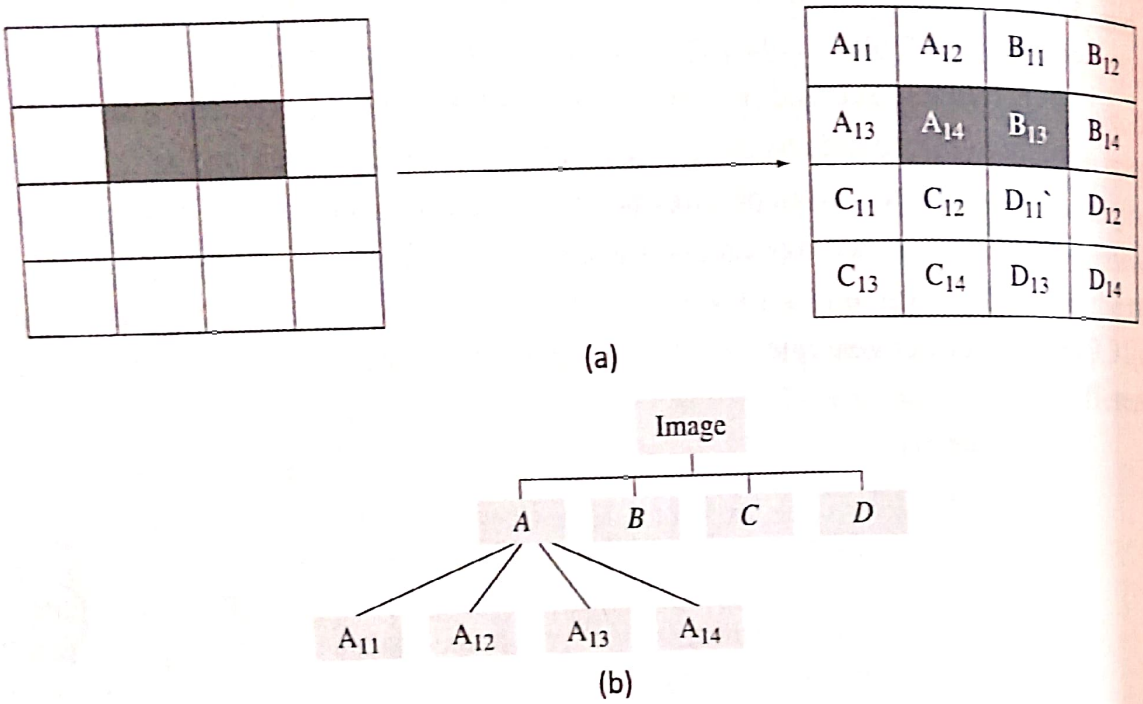


Fig. 9.33 Split-and-merge algorithm using pyramid quadtree (a) Original image with the shaded region (b) Quadtree representation

The quad tree is generated with *A*, *B*, *C*, and *D* as children. All quadrants generate children. C and D have children that are same and hence they are discarded. Only *A* and *B* have the shaded region as children. A has A_{14} and B has B_{13} shaded regions. Since they share the same properties, they are merged into a single region. All the remaining regions become other non-shaded regions.

9.9 ACTIVE CONTOUR MODELS

The idea of the hybrid approach of image segmentation is to combine the advantages of both edge and region approaches. Active contour model is an example of hybrid approach.

Active contour is a curve defined in an image that is allowed to change its location and shape to satisfy the preconditions so that the region of interest can be segmented. This type of algorithm is called "physics-based vision algorithms" as they are posed as a physics problem of energy functional. The basis of the active contour models is the concept of minimizing energy of splines.

Active contour models are also known as deformable models, dynamic contours, and snakes. The performance of active contour models is effective in real images. The

advantage of this approach is that it is immune to noise, boundaries, and gaps present in the image. Snakes are also easy to use and are reasonably faster in segmenting 3D and dynamic data. The basis of these models is the concept of computation of minimal energy paths. For this, the active contour models should be initialized and its energy functional should be minimized to segment the region of interest. The procedure for implementing active contour paths is listed here:

Initialization Initially, a contour is placed like a seed pixel inside the region of interest. The contour consists of small curves which, like an elastic band, grows or shrinks to fit the region of interest. The points of the snake are represented as a spline curve with a set of control points. Specifying all the points makes the algorithm computationally intense. Hence the snake is represented as a spline with few control points and is interpolated through the control points.

The points of the snake C are represented in parametric form as $C(s) = (X(s), Y(s))$, where s ranges between 0 and 1. When $s = 0$, $C(0)$ gives the coordinate pair $(X(0), Y(0))$ of the starting point. The end point is given as $(X(1), Y(1))$. The rest of the points are given by s in the range 0 to 1.

Energy functional The contour is moved by forces or energy functional to the boundaries of the object. The term *force* or *energy functional* is derived from physical systems. These forces or energy functions are derived from the image itself. By controlling the energy function, the evolving snakes can be controlled.

The energy of snakes is of two types:

1. Internal energy
2. External energy

Internal energy depends on intrinsic properties such as the boundary length or curvature. It is imposed as a constraint for image smoothness. The minimization of these forces leads to shrinkage of the contour. To counteract the shrinkage of snakes, external energy is used. The external energy is derived from the image structures and it determines the relationship between the snake and the image. This can be the sum of the grey levels or any complicated function of the grey scale gradients. If it is the sum of the grey levels, then the snake latches onto the brightness. *External energy* depends on factors such as image structure and user-supplied constraints. These constraints also control the external force. The total energy is the sum of internal, external, and constraint energies and can be specified as

$$E_{\text{total}} = \alpha E_{\text{length}} + \beta E_{\text{curvature}} - \gamma E_{\text{image}}$$

E_{length} is the total length of the snake, $E_{\text{curvature}}$ is its curvature, and E_{image} is the counteracting or opposing force. The length and curvature can be calculated in terms of derivatives.

$$E_{\text{length}} = \sum_{i=1}^{n} \left(\frac{\partial x_i}{\partial s} \right)^2 + \left(\frac{\partial y_i}{\partial s} \right)^2$$

$$E_{\text{curvature}} = \sum_{i=1}^{n}\left(\frac{\partial^2 x_i}{\partial s^2}\right)+\left(\frac{\partial^2 y_i}{\partial s^2}\right)$$

The other parameters – α, β, and γ – are used to control the energy functions of the snake. The values of these parameters are estimated empirically and can be adjusted.

Energy minimization The concept of snake is the minimization of energy. The problem now is to minimize the energy functional of internal, external and user constraints which is an iterative process. This is solved numerically in its Euler-Lagrange form. The points of the snake are evaluated iteratively. The parameters are adjusted so that the energy is reduced. This iteration continues till it settles to the minimum of E_{total}. By that time the contour fits tightly around the region of interest. If the chosen initial contour is good, the snake converges very quickly. The main advantage of snakes is that, very often, they provide closed coherent areas.

An active contour model is illustrated in Figs 9.34(a)–(c).

Fig. 9.34 Active contour model (a) Original image—contour is shown in yellow (b) Final contour shown in red (c) Segmented image (Refer to the OUPI website for colour images)

.10 VALIDATION OF SEGMENTATION ALGORITHMS

Validation is carried out to evaluate the accuracy of the segmentation method. This validation process requires a truth model, with which the segmented results can be compared. A common truth model is manual segmentation by an experienced operator. This is an effective technique and the most straightforward approach. However, human errors should also be taken into account. The participation of more human experts increases the objectivity of the validation process. The contingency table for the sensitivity and specificity analyses of a validation algorithm is given in Table 9.1.

Table 9.1 Contingency table

Human expert	Detected by the algorithm (computer system)	Missed by the algorithm (computer system)
ROI present	True positive (TP)	False negative (FN)
ROI absent	False positive (FP)	True negative (TN)

The true positives (TP) and true negatives (TN) are correct classifications. A false positive (FP) is when the outcome is an incorrectly predicted image feature when the feature is in fact absent. A false negative (FN) is when the outcome of the algorithm is incorrectly predicted as absence of a feature, when in reality it is actually present in the image. FP is also known as false alarm. The sensitivity of the segmentation algorithm specifies the ability of the algorithm in detecting the ROI.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{TN}} \times 100\%$$

Specificity is a measure of the ability of a test to give a negative result in extracting the ROI.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \times 100\%$$

The efficiency of the segmentation is given as

$$\text{Efficiency} = \frac{\text{TN} + \text{TP}}{\text{TN} + \text{TP} + \text{FN} + \text{FP}}$$

The efficiency or success rate is the ability of the algorithm to extract the ROI.

SUMMARY

- Segmentation is the process of partitioning the digital image into multiple regions. The main aim of segmentation is to extract the ROI for image analysis.
- Segmentation algorithms can be classified into three categories—manual segmentation, semi-automatic segmentation, and automatic segmentation algorithms. They can also be classified based on the technique, as contextual and non-contextual.
- An edge is a set of connected pixels that lie on the boundary between two regions. The pixels on an edge are called edge points.
- There are three stages of edge detection—filtering, differentiation, and localization.
- The objective of the sharpening filter is to highlight the intensity transitions.
- Edge detectors often do not produce continuous edges. Hence, edge linking algorithms are required to make them continuous.
- The Hough transform is used to fit points as plane curves.
- Thresholding produces uniform regions based on the threshold criterion T.
- Region oriented algorithms use the principle of similarity to produce coherent regions.
- A snake is a contour which, like an elastic band, grows or shrinks to fit the object of interest. Snakes can be controlled by the energy function.

KEY TERMS

Between-class variance It is the variation that exists among members across classes.

Bimodal histogram It is a special kind of histogram, which has two peaks separated by a valley.

Corner point It is formed when two bright regions meet and is characterized by a strong gradient in all directions.

Cracks These (also known as crack edges) are

formed when two aligned pixels meet and represent pixel value differences.

Derivative filters These are filters that use the gradient magnitude and orientation for finding edges.

Edge It is a set of connected pixels that lie on the boundary between two distinct regions that are local, significant, and represent sharp intensity variations.

Edge linking It is the process of connecting the disjoint edges.

Edge map It is also known as edge image where pixels represent the edge magnitude.

Edge point It is a point or pixel where the intensity changes occur rapidly.

Edge relaxation It is a process of evaluation of pixels based on local concept.

Hysteresis thresholding It is a threshold process that uses two thresholds for selecting the edge pixels.

Image Segmentation It is the process of extracting the ROI.

Multimodal histogram It is a histogram that has multiple peaks.

Pattern fit It is the process of approximating an image with a pattern, that is, an analytical function.

Peakiness test It is a test for checking whether the peak of the histogram is genuine or caused by noise.

Region adjacency graph It is a data structure used to model the adjacency relations of regions of an image.

Seed pixel It is the starting pixel of a region-growing process.

Snake It is a deformable model that fits a model for segmenting ROI.

Template matching filters These are filters that are designed using masks that are sensitive to a particular orientation.

Thresholding It is the process of producing uniform regions based on the threshold value.

Thresholding It is the process of using a threshold to extract the ROI.

Unimodal histogram It is a histogram that has one central peak.

Within-class (intra-class) variation It is the variation that exists among members of a single class.

Zero-crossing It is the position of sign change of the first derivative among neighbouring points.

REVIEW QUESTIONS

1. What is the formal definition of segmentation?
2. What are the different ways in which segmentation algorithms can be classified?
3. List the edge operators.
4. What is meant by first- and second-order derivatives of an image?
5. What is the problem of second-order derivatives?
6. What is meant by edge linking?
7. What is meant by a crack edge?
8. What is the objective of edge relaxation?
9. What is a Hough transform?
10. Why and how are edges combined?
11. What is meant by a contour?
12. How can segmentation algorithms be evaluated?
13. How are isolated points and lines recognized?
14. Explain in detail the stages of edge detection algorithms. How are they present in edge operators?
15. Explain in detail the edge linking algorithms.
16. Explain in detail the active contour models.
17. Explain watershed segmentation.
18. Explain how the edge segmentation algorithms are evaluated.

NUMERICAL PROBLEMS

1. Consider a one-dimensional image $f(x) = [10\ 10\ 10\ 10\ 40\ 40\ 40\ 40\ 20\ 20]$ What are the first and second derivatives? Locate the position of edge.
2. Consider an image

$$F = \begin{pmatrix} 1 & 2 & 5 \\ 5 & 5 & 5 \\ 5 & 3 & 2 \end{pmatrix}$$

Show the output of any edge detection algorithm.

3. A black object in a white background is known to occupy 20%, 40%, and 80% of the histogram. What is the threshold value? If the colours of the object and the background are reversed, what is the threshold value?
4. Consider the following image:

8	4	5
8	4	4
7	5	1

Show the crack edges.

5. For the given image, apply graph theoretic algorithm and show the flow of the edges.

9	7	9
2	4	3
7	5	1

6. Explain in detail the graph theoretic algorithm, with respect to the following image:

4	5	6
11	9	38
3	11	13

7. Consider the following image:

1	1	9	8	7
0	1	8	8	8
0	0	7	9	8
0	1	8	8	9
1	2	8	8	9

What is the result if the threshold is 3? If the threshold condition is changed as (pixel–seed) < 0.1(maximum – minimum) of 8-neighbourhood, what is the resultant image?

8. For the given image,

$$F = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 9 & 9 & 9 & 9 & 2 & 2 & 2 & 2 \\ 9 & 9 & 9 & 9 & 2 & 2 & 2 & 2 \\ 9 & 9 & 9 & 9 & 2 & 2 & 2 & 2 \\ 9 & 9 & 9 & 9 & 2 & 2 & 2 & 2 \\ 9 & 9 & 9 & 9 & 2 & 2 & 2 & 2 \\ 9 & 9 & 9 & 9 & 2 & 2 & 2 & 2 \\ 9 & 9 & 9 & 9 & 2 & 2 & 2 & 2 \end{pmatrix}$$

show the result of the split-and-merge algorithm.

9. The contingency table of a segmentation algorithm is given as follows:

Expert human	Present	Absent
ROI present	8	1
Absent	1	1

What is the efficiency of the segmentation algorithm?