applying

Answer

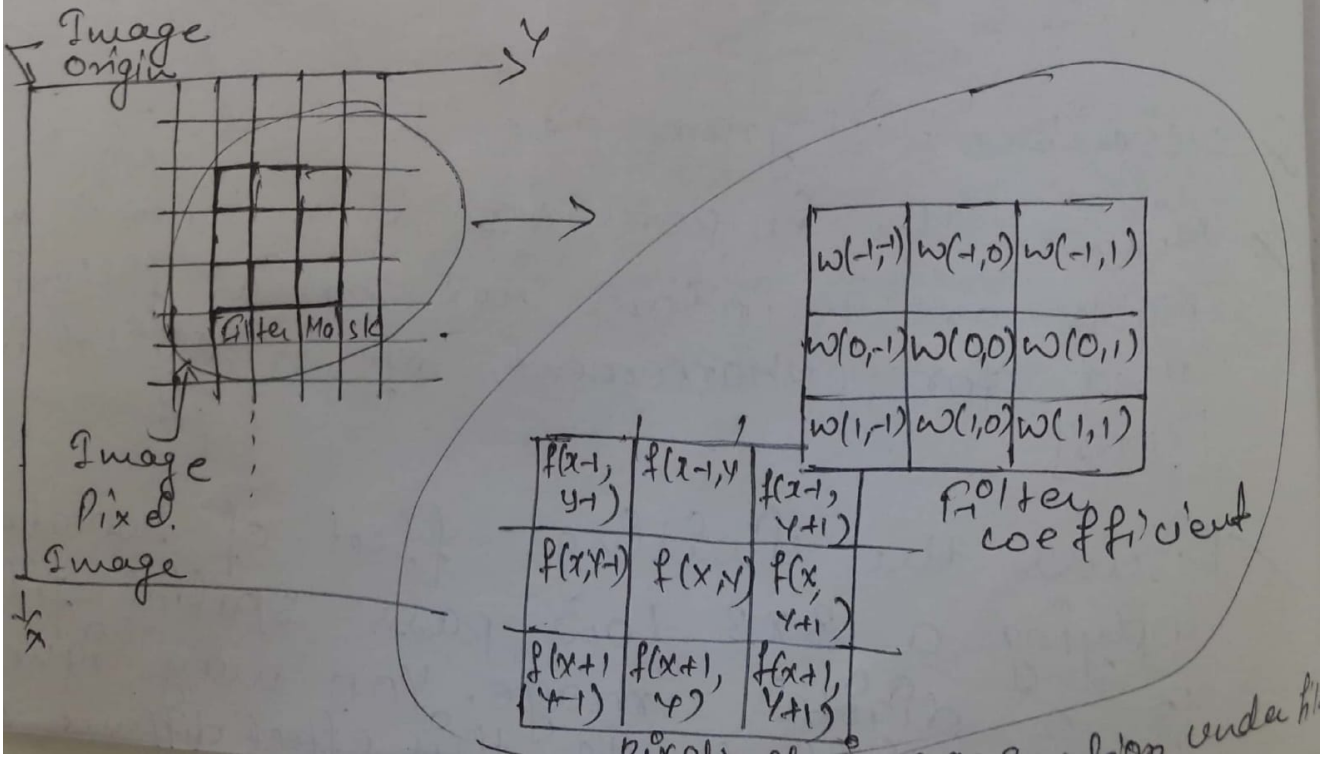
2. The basic concepts of spatial filtering in image enhancement are as follows:

- The mechanics of Spatial Filtering

→ Spatial filtering consists of a neighborhood and a predefined operation that is performed on the image pixels encompassed by the neighborhood.

Filtering creates a new pixel with coordinates equal to the coordinates of the center of the neighborhood, and whose value is the result of the filtering operations.

If the operation performed on the image pixel is linear, then the filter is called a linear spatial filter. Otherwise the filter is non-linear.



- Figure illustrates the mechanics of linear spatial filtering using a 3X3 ~~me~~ neighborhood. At any point (x,y) in the image, the response, g(x,y), of the filter is the sum of products (SOP) of the filter coefficients and the image pixels encompassed by the filter:

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \cdots + w(0,0)f(x,y) + \cdots + w(1,1)f(x+1,y+1)$$

- Linear spatial filtering of an image of size MXN with a filter of size mxn is given by the expression:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

where, x & y are varied so that each pixel in w visits every pixel in f.

Spatial Correlation and convolution

- Correlation is the process of moving a filter mask over the image and computing the sum of products at each location
- Convolution: The mechanics of convolution are the same, except that the filter is first rotated by 180°.

correla

c

f w

a) 00010000 1 2 3 2 8 .

A

b) 0 0 0 1 0 0 0 0

1 2 3 2 8

↳ starting position alignment

c) Zero-padding

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

1 2 3 2 8

d) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

1 2 3 2 8

↳ position after 1 shift

e) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

1 2 3 2 8

↑ after 4 shifts

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

1 2 3 2 8

↑ final position

0 0 0 8 2 3 2 1 0 0 0 0 [Full correlation result]

0 8 2 3 2 1 0 0 [cropped correlation

convolution

i) 0 0 0 1 0 0 0 0 rotated to 180°
8 2 3 2 1

j) 0 0 0 1 0 0 0 0
8 2 3 2 1

k) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1

l) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1

m) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
8 2 3 2 1

n) 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
~~8 2 3 2 1~~ 8 2 3 2 1

o) Full convolution result

0 0 0 1 2 3 2 8 0 0 0 0

p) cropped convolution result

0 1 2 3 2 8 0 0.

* correlation is a function of displacement of filter.

. Using correlation and convolution to perform SP filtering is a matter of preference.

Q. Describe Histog

- Histogram equalization automatically determines a transformation function that produces an uniform histogram.

In Histogram matching this method is used to generate a processed image that has a specified histogram and this process is called Histogram matching or Histogram specification.

$$s = T(r)$$

$$= (L-1)\int_0^r P_r(w)\,dw$$

where, w is dummy variable for integration
s is the continuous version of Histogram equalization.

Let r be continuous random variables consider, z to be continuous random variable with property

$$G(z) = (L-1)\int_0^t P_z(t)\,dt$$

where,

$P_z(t)$ is probability density function
t is dummy variable for integration

so, $G(z) = s$

or, $G(z) = T(r)$

or, $z = G^{-1}[T(r)]$

$= G^{-1}[s]$

Consider 2-bit image of size 5 × 5

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

$L = 4$

intensity level $[0, 3]$

~~Histogram~~ ~~25~~

Total pixels = 25

Histogram component →

$P(r_0) = \frac{6}{25} = 0.24$

$P(r_1) = \frac{7}{25} = 0.28$

$P(r_2) = 0.28$

$P(r_3) = 0.20$

$$\bar{m} = \sum_{i=0}^{3} p_i(r_i)\, r_i$$

Avg. value of intensities in the image.

$= 0(0.24)$
$+ 1(0.28)$
$+ 2(0.28)$
$+ 3(0.20)$
$= 1.44$

→ In addition to gamma correction power law transformations are useful for general purpose.

Histogram:-

1	1	1	5	0	7
1	1	1	4	8	8
2	1	1	3	8	8
3	2	2	2	2	6
4	3	2	2	2	6
2	5	3	3	7	8
2	1	4	3	7	1

Intensity	no. of pixels	Probab.	$p(r_k)$ normh(r_k)
0	1	$\frac{1}{42}$	0.023
1	10	$\frac{10}{42}$	0.238
2	10	$\frac{10}{42}$	0.238
3	6	$\frac{6}{42}$	0.142
4	3	$\frac{3}{42}$	0.0714
5	2	2/42	0.0476
6	2	2/42	0.0476
7	3	3/42	0.0714
8	5	5/42	0.119

histogram of a digital image with intensity levels in range $[0, L-1]$ is a discrete function $h(r_k) = n_k$ where r_k is the k^{th} intensity level and n_k is the no. of pixels in the image with intensity r_k

Otsu's algorithm may be summarized as follows:

1. Compute the normalized histogram of the input image. Denote the components of the histogram by p_i, $i = 0, 1, 2, \ldots, L - 1$.
2. Compute the cumulative sums, $P_1(k)$, for $k = 0, 1, 2, \ldots, L - 1$, using Eq. (10.3-4).
3. Compute the cumulative means, $m(k)$, for $k = 0, 1, 2, \ldots, L - 1$, using Eq. (10.3-8).
4. Compute the global intensity mean, m_G, using (10.3-9).
5. Compute the between-class variance, $\sigma_B^2(k)$, for $k = 0, 1, 2, \ldots, L - 1$, using Eq. (10.3-17).
6. Obtain the Otsu threshold, k^*, as the value of k for which $\sigma_B^2(k)$ is maximum. If the maximum is not unique, obtain k^* by averaging the values of k corresponding to the various maxima detected.
7. Obtain the separability measure, η^*, by evaluating Eq. (10.3-16) at $k = k^*$.

The following example illustrates the preceding concepts.

EXAMP
Optimur
threshol

Figure 10.39(a) shows an optical microscope image of polymersome cells, and [illegible] objective of this example is to seg-

...which may be viewed as a multiclass, multivariable thresholding problem.

10.4 Region-Based Segmentation

...s discussed in Section 10.1, the objective of segmentation is to partition an ...age into regions. In Section 10.2, we approached this problem by attempting to ...nd boundaries between regions based on discontinuities in intensity levels, ...hereas in Section 10.3, segmentation was accomplished via thresholds based on ...e distribution of pixel properties, such as intensity values or color. In this sec...on, we discuss segmentation techniques that are based on finding the regions ...rectly.

10.4.1 Region Growing

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

Selecting a set of one or more starting points often can be based on the ...ature of the problem, as shown later in Example 10.23. When a priori infor...mation is not available, the procedure is to compute at every pixel the same set ...f properties that ultimately will be used to assign pixels to regions during the ...rowing process. If the result of these computations shows clusters of values, ...ose pixels whose properties place them near the centroid of these clusters can ...be used as seeds.

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

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

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

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

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

We illustrate the

10.4.2 Region Splitting and Merging

The procedure discussed in the last section grows regions from a set of se
points. An alternative is to subdivide an image initially into a set of arbitra
disjoint regions and then merge and/or split the regions in an attempt to sa
fy the conditions of segmentation stated in Section 10.1. The basics of splitt
and merging are discussed next.

...proach ... successively into smaller and ...aller quadrant regions so that, for any region R_i, $Q(R_i)$ = TRUE. We start ...ith the entire region. If $Q(R)$ = FALSE, we divide the image into quadrants. ...Q is FALSE for any quadrant, we subdivide that quadrant into subquad- ...nts, and so on. This particular splitting technique has a convenient represen- ...tion in the form of so-called *quadtrees*, that is, trees in which each node has ...actly four descendants, as Fig. 10.52 shows (the images corresponding to the ...odes of a quadtree sometimes are called *quadregions* or *quadimages*). Note ...at the root of the tree corresponds to the entire image and that each node ...rresponds to the subdivision of a node into four descendant nodes. In this ...se, only R_4 was subdivided further.

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

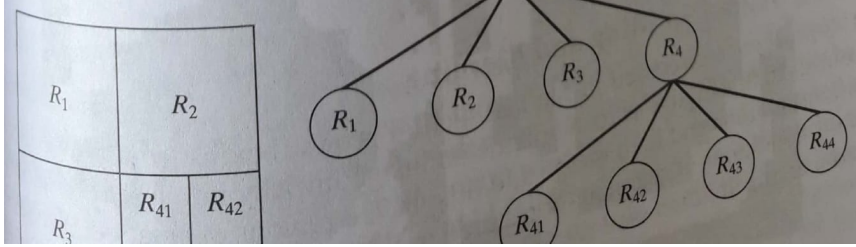
See Section 2.5.2 regarding region adjacency.

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

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

...is customary to specify a minimum quadregion size beyond which no further ...litting is carried out.

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



a b

FIGURE 10.52 (a) Partitione... image. (b) Correspondi... quadtree. *R* represents t... entire image region.

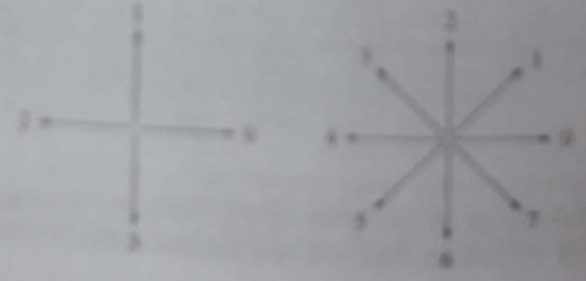
11.1.2 Chain Codes

Chain codes are used to represent a boundary by a connected sequence of straight-line segments of specified length and direction. Typically, this representation is based on 4- or 8-connectivity of the segments. The direction of each segment is coded by using a numbering scheme, as in Fig. 11.3. A boundary code formed as a sequence of such directional numbers is referred to as a *Freeman chain code*.

Digital images usually are acquired and processed in a grid format with equal spacing in the x- and y-directions, so a chain code can be generated by following a boundary in, say, a clockwise direction and assigning a direction to the segments connecting every pair of pixels. This method generally is unacceptable for two principal reasons: (1) The resulting chain tends to be quite long and (2) any small disturbances along the boundary due to noise or imperfect segmentation cause changes in the code that may not be related to the principal shape features of the boundary.

An approach frequently used to circumvent these problems is to resample the boundary by selecting a larger grid spacing, as Fig. 11.4(a) shows. Then, as the boundary is traversed, a boundary point is assigned to each node of the large grid, depending on the proximity of the original boundary to that node, as in Fig. 11.4(b). The resampled boundary obtained in this way then can be represented by a 4- or 8-code. Figure 11.4(c) shows the coarser boundary points represented by an 8-directional chain code. It is a simple matter to convert from an 8-code to a 4-code, and vice versa (see Problems 2.12 and 2.13). The starting point in Fig. 11.4(c) is (arbitrarily) at the topmost, leftmost point of the boundary, which gives the chain code 0766 . . . 12. As might be expected, the accuracy of the resulting code representation depends on the spacing of the sampling grid.

The chain code of a boundary depends on the starting point. However, the code can be *normalized* with respect to the starting point by a straightforward procedure: We simply treat the chain code as a circular sequence of direction numbers and redefine the starting point so that the resulting sequence of numbers forms an integer of minimum magnitude. We can normalize also for rotation (in angles that are integer multiples of the directions in Fig. 11.3) by using the *first difference* of the chain code instead of the code



ure 11.3
ection
bers for
-directional
code, and
-directional
code.