Reprinted from:

REAL-TIME PARALLEL COMPUTING IMAGE ANALYSIS (1981)
Edited by Morio Onoe, Kendall Preston, JR.
and Azriel Rosenfeld
Book available from: Plenum Publishing Corporation
233 Spring Street, New York, N.Y. 10013

SOME NEIGHBORHOOD OPERATORS

R. M. Haralick

Virginia Polytechnic Institute and State University

Blacksburg, VA 24061 USA

1. INTRODUCTION

In this paper we review a variety of neighborhood operators in a way which emphasizes their common form. These operators are useful for image segmentation tasks as well as for the construction of primitives involved in structural image analysis. The common form of the operators suggests the possibility of a large scale integration hardware implementation in the VLSI device technology.

Neighborhood operators can be classified according to type of domain, type of neighborhood, and whether or not they are recursive. The two types of domains consist of numeric or symbolic data. Operators having a numeric domain are usually defined in terms of arithmetic operations such as addition, subtraction, computing minimums or maximums, etc. Operators having a symbolic domain are defined in terms of Boolean operations such as AND, OR, NOT, or table look-up operations.

There are two basic neighborhoods a simple operator may use: a 4-connected neighborhood and an 8-connected neighborhood. As illustrated in Figure 1, the 4-connected neighborhood about a pixel consists of the pixel and its north, south, east, and west neighbors. The 8-connected neighborhood about a pixel consists of all the pixels in a 3x3 window whose center is the given pixel.

Recursive neighborhood operators are those which use the same image memory for their input and output. In this way an output from a previously used nearby neighborhood influences the output from its current neighborhood. Non-recursive neighborhood operators are those which use independent image memory for input and

4-connected neighborhood 8-connected neighborhood

Fig. 1. Illustration of the indexing of the pixels in 4-connected and 8-connected neighborhood of X_0.

output. Previous outputs cannot influence current outputs. Rosenfeld and Pfaltz (1966) call recursive operators sequential and non-recursive operators parallel.

2. REGION GROWING OPERATOR

The region growing operator just described is non-recursive and has a symbolic domain. It changes all pixels whose label is the background label to the non-background label of its neighboring pixels. It is based on a two-argument primitive function h which is a projection operator whose output is either its first argument or its second argument depending on their values. If the first argument is the special symbol "g" for background, then the output of the function is its second argument. Otherwise, the output is the first argument. Hence:

$$h(c,d) = \begin{cases} d & \text{if } c = g \\ c & \text{if } c \neq g \end{cases} \qquad (1)$$

The region growing operator uses the primitive function h in the following way. For the operator in the 4-connected mode, let $a_0 = x_0$. Define $a_n = h(a_{n-1},x_n)$, $n = 1,\ldots,4$. Then the output symbol y is defined by $y = a_4$. For the operator in the 8-connected mode, let $a_0 = x_0$. Define $a_n = h(a_{n-1},x_n)$, $n = 1,\ldots,8$. Then the output symbol y is defined by $y = a_8$.

A more sophisticated region growing operator grows background border pixels to the region label a majority of its neighbors have. In the 8-connected mode such an operator sets $a_n = h(x_0,x_n)$, $n = 1,\ldots,8$ and defines the output symbol y by $y = c$ where $\#\{n|a_n = c\} > \#\{n|a_n = c'\}$ for all c'.

3. NEAREST NEIGHBOR SETS

Given a symbolic image with background pixels labeled "g" and each connected set of non-background pixels labeled with a unique label, it is possible to label each background pixel with the label of its closest non-background neighboring pixel. Just iteratively grow the non-background labels into the background labels using the 8-neighborhood if the max distance is desired, using the 4-neighborhood if city block is desired, using the 4-neighborhood and 8-neighborhood alternately in the ratio of $\sqrt{2}$ for Euclidean distances.

4. REGION SHRINKING OPERATORS

The region shrinking operator is non-recursive and has a symbolic data domain. It changes the label on all border pixels to the background label. The region shrinking operator defined here can change the connectivity of a region and can even entirely delete a region upon repeated application. It is based on a two-argument primitive function h which can recognize whether or not its arguments are identical. If the arguments are the same, h outputs the value of the argument. If the arguments differ, h outputs the special symbol "g" for background. Hence:

$$h(c,d) = \begin{cases} c & \text{if } c = d \\ g & \text{if } c \neq d \end{cases} \tag{2}$$

The region shrinking operator uses the primitive function h in the following way. For the operator in the 4-connected mode, let $a_0 = x_0$. Define $a_n = h(a_{n-1}, x_n)$, $n = 1,\ldots,4$. Then the output symbol y is defined by $y = a_4$. For the operator in the 8-connected mode, let $a_0 = x_0$. Define $a_n = h(a_{n-1}, x_n)$, $n = 1,\ldots,8$. Then the output symbol y is defined by $y = a_8$.

A more sophisticated region shrinking operator shrinks border pixels only if they are connected to enough pixels of unlike regions. In the 8-connected mode it sets $a_n = h(x_0, x_n)$, $n = 1,\ldots,8$ and defines the output symbol y by:

$$y = \begin{cases} g & \text{if } \#\{n \mid a_n = g\} > k \\ x_0 & \text{otherwise} \end{cases} \tag{3}$$

As mentioned in the section on nearest neighbor sets, to obtain a region shrinking (region growing) which is close to a Euclidean distance region shrinking (growing), the 4-neighborhood and the 8-neighborhood must be used alternately approximating as closely as possible the ratio $\sqrt{2}/1$ (Rosenfeld and Pfaltz, 1968). A ratio of

4/3 can be obtained by the sequence 4:3 = <4,8,4,8,4,8,4> and a ratio of 3/2 can be obtained by the sequence 3:2 = <4,8,4,8,4>. Alternating these two sequences will give a ratio of 7/5 just smaller than 2. Using one 4:3 sequence followed by two 3:2 sequences gives a ratio of 10/7, just over $\sqrt{2}$. Alternating between <4:3,3:2,3:2> and <4:3,3:2> gives a ratio of 17/12 which differs from $\sqrt{2}$ by less than 2.5×10^{-3}, an approximation which should be good enough for most purposes.

The choice of 4-neighborhood or 8-neighborhood for the current iteration which best approximates the Euclidean distance can be determined dynamically. Let N4 be the number of uses of the 4-neighborhood so far and N8 be the number of the 8-neighborhood so far. If $|N4-2(N8+1)| < |N4+1-2N8|$, then use the 8-neighborhood for the current iteration; else use the 4-neighborhood.

5. MARK INTERIOR BORDER PIXELS

The mark interior/border pixels operator is non-recursive and has a symbolic data domain. It marks all interior pixels with the label "i," standing for interior, and all border pixels with the label "b," standing for border. It is based on two primitive functions. One is a two-argument primitive function h very similar to that used in the region shrinking operator. The other one is a one-argument primitive function f. The two argument primitive function h can recognize whether or not its arguments are identical. For identical arguments it outputs the argument. For non-identical arguments it outputs the special symbol "b" standing for border. The one-argument primitive function f can recognize whether or not its argument is the special symbol "b." If it is it outputs b. If not it outputs the special symbol "i" standing for interior. Hence:

$$h(c,d) = \begin{cases} c & \text{if } c = d \\ b & \text{if } c \neq d \end{cases} \tag{4}$$

$$f(c) = \begin{cases} b & \text{if } c = b \\ i & \text{if } c \neq d \end{cases} \tag{5}$$

The mark interior/border pixel operator uses the primitive function h in the following way. For the operator in the 4-connected mode, let $a_0 = x_0$. Define $a_n = h(a_{n-1},x_n)$, $n = 1,\ldots,4$. Then the output symbol y is defined by $y = f(a_4)$. For the operator in the 8-connected mode, let $a_0 = x_0$. Define $a_n = h(a_{n-1},x_n)$, $n = 1,\ldots,8$. Then the output symbol y is defined by $y = f(a_8)$.

6. CONNECTIVITY NUMBER OPERATOR

The connectivity number operator is a non-recursive operator which has a symbolic data domain. Its purpose is to classify the way a pixel is connected to its like neighbors. As shown in Figure 2, there are six values of connectivity, 5 values for border pixels and 1 value for interior pixels. The border pixels consist of isolated pixels, edge pixels, connected pixels, branching pixels, crossing pixels, and interior pixels. The connectivity number operator associates with each pixel a symbol called the connectivity number of the pixel. The symbol, although a number, has no arithmetic number properties. The number designates which of the six values of connectivity a pixel has with its like neighbors.

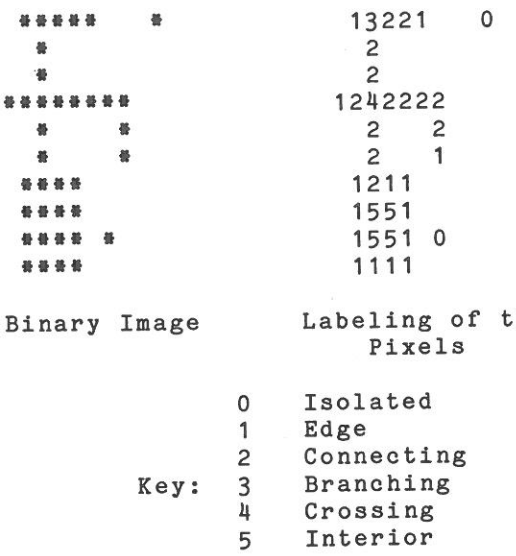


Fig. 2. Illustration of a connectivity number labeling of a binary image.

6.1 Yokoi Connectivity Number

The definition we give here of connectivity number is based on a slight generalization of the definitions suggested by Yokoi, Toriwaki, and Fukumura (1975). This is not the only definition of connectivity number. Another definition given by Rutovitz (1966) is based on the number of transitions from one symbol to another as one travels around the 8-neighborhood of a pixel. The operator, as

defined here, uses an 8-connected neighborhood and can be defined for either 4-connectivity or 8-connectivity.

For 4-connectivity, a pixel is an interior pixel if its value and that of each of its 4-connected neighbors is the same. In this case its 4-connectivity takes the index value 5. Otherwise, the 4-connectivity of a pixel is given by the number of times a 4-connected neighbor has the same value but the corresponding 3 pixel corner neighborhood does not. These corner neighbors are illustrated in Figure 3.

For 8-connectivity, a pixel is an interior pixel if its value and that of each of its 8-connected neighbors is the same. Otherwise the 8-connectivity of a pixel is given by the number of times a 4-connected neighbor has a different value and at least one pixel in the corresponding 3 pixel neighborhood corner has the same value.

The connectivity operator requires two primitive functions: a function h which can determine whether a 3-pixel corner neighborhood is connected in a particular way and a function f which basically counts the number of arguments which have a particular value.

For 4-connectivity, the function h of four arguments is defined by:

$$h(b,c,d,e) = \begin{cases} q \text{ if } b = c \text{ and } (d \neq b \text{ or } e \neq b) \\ r \text{ if } b = c \text{ and } (d = b \text{ and } e = b) \\ s \text{ f } b \neq c \end{cases} \tag{6}$$

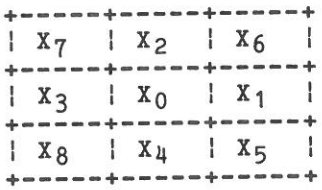
the function f of four arguments is defined by:

$$f(a_1,a_2,a_3,a_4) = \begin{cases} 5 \text{ if } a_1 = a_2 = a_3 = a_4 = r \\ n \text{ where } n = \#\{a_k | a_k = q\}, \text{ otherwise} \end{cases} \tag{7}$$

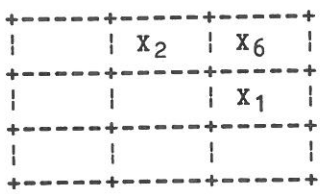
The connectivity operator using 4-connectivity is then defined in the following way. Let

$$a_1 = h(X_0,X_1,X_6,X_2)$$
$$a_2 = h(X_0,X_2,X_7,X_3)$$
$$a_3 = h(X_0,X_3,X_8,X_4)$$
$$a_4 = h(X_0,X_4,X_5,X_1)$$

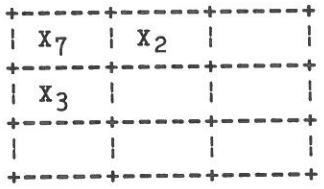
Define the connectivity number y by $y = f(a_1,a_2,a_3,a_4)$.



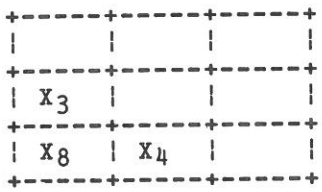
(a) Indexing pixels in a 3 x 3 neighborhood



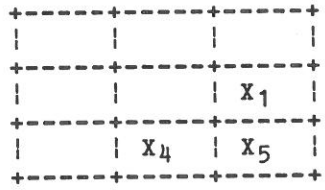
(b) Corner of X_1



(c) Corner of X_2



(d) Corner of X_3



(e) Corner of X_4

Fig. 3. Illustration of the corner neighborhood corresponding to each of the East, North, West, and South neighbors of the center pixel.

For 8-connectivity, the function h is slightly different. It is defined by

$$h(b,c,d,e) = \begin{cases} q \text{ if } b \neq c \text{ and } (d = b \text{ or } e = b) \\ r \text{ if } b = c \text{ and } (d = b \text{ and } e = b) \\ s \text{ if } b \neq c \text{ and } (d \neq b \text{ and } e \neq b) \end{cases} \tag{8}$$

Then, as before, the connectivity number y is defined by $y = f(a_1,a_2,a_3,a_4)$.

6.2 Rutovitz Connectivity Number

The definition we give here of the Rutovitz connectivity number, sometimes called a crossing number, is based on a slight generalization of the definitions suggested by Rutovitz (1966). The Rutovitz connectivity number simply counts the number of transitions from symbols which are different than that of the center pixel to symbols which are the same as that of the center pixel as one travels around the 8-neighborhood of a pixel.

The Rutovitz connectivity number requires a three argument primitive function h defined by

$$h(a,b,c) = \begin{cases} 1 \text{ if } (a = b \text{ and } a \neq c) \text{ or } (a \neq b \text{ and } a = c) \\ 0 \text{ otherwise} \end{cases} \tag{9}$$

Then set

$$\begin{aligned} a_1 &= h(X_0,X_1,X_6) \\ a_2 &= h(X_0,X_6,X_2) \\ a_3 &= h(X_0,X_2,X_7) \\ a_4 &= h(X_0,X_7,X_3) \\ a_5 &= h(X_0,X_3,X_8) \\ a_6 &= h(X_0,X_8,X_4) \\ a_7 &= h(X_0,X_4,X_5) \\ a_8 &= h(X_0,X_5,X_1) \end{aligned}$$

The output value y is then given by

$$y = \sum_{n=1}^{8} a_n \tag{10}$$

7. CONNECTED SHRINK OPERATOR

The connected shrink operator is a recursive operator having a symbolic data domain. It is similar in certain respects to the connectivity number operator and the region shrinking operator. Instead of labeling all border pixels with background symbol "g," the connected shrink operator only labels those border pixels which can be deleted from a connected region without disconnecting the region. Since it is applied recursively, pixels which are interior during one position of the scan may appear as border pixels at another

position of the scan and eventually may be deleted by this operator. After one complete scan of the image, the set of pixels which get labeled as background is a strong function of the way in which the image is scanned with the operator. For example, as illustrated in Figure 4, a top-down left-right scan will delete all edge pixels which are not right boundary edge pixels.

The theoretical basis of the connected shrink operator was explored by Rosenfeld and Pfaltz (1966), Rosenfeld (1970), and Stefanelli and Rosenfeld (1971). Basically, a pixel's label is changed to "g," for background, if upon deleting it from the region it belongs to, the region remains connected. The operator definition given here is due to Yokoi, Toriwaki, and Fukumura (1975). The operator uses an 8-connected neighborhood and can be defined for deleting either 4-deletable or 8-deletable pixels. It requires two primitive functions: a function h which can determine whether the 3-pixel corner of a neighborhood is connected and a function g which basically counts the number of arguments having certain values.

In the 4-connectivity mode, the 4-argument primitive function h is defined:

$$h(b,c,d,e) = \begin{cases} 1 \text{ if } b = c \text{ and } (d \neq b \text{ or } e \neq b) \\ 0 \text{ otherwise} \end{cases} \tag{11}$$

In the 8-connectivity mode, the 4- argument primitive function h is defined by:

$$h(b,c,d,e) = \begin{cases} 1 \text{ if } c \neq b \text{ and } (d = b \text{ or } e = b) \\ 0 \text{ otherwise} \end{cases} \tag{12}$$

The 5-argument primitive function f is defined by:

$$f(a_1,a_2,a_3,a_4,x) = \begin{cases} g \text{ if exactly one of } a_1,a_2,a_3,a_4 = 1 \\ x \text{ otherwise} \end{cases} \tag{13}$$

Using the indexing convention of Figure 3, the connected shrink operator is defined by letting:

$$a_1 = h(X_0,X_1,X_6,X_2)$$
$$a_2 = h(X_0,X_2,X_7,X_3)$$
$$a_3 = h(X_0,X_3,X_8,X_4)$$
$$a_4 = h(X_0,X_4,X_5,X_1)$$

The output symbol y is defined by $y = f(a_1,a_2,a_3,a_4,x_0)$.

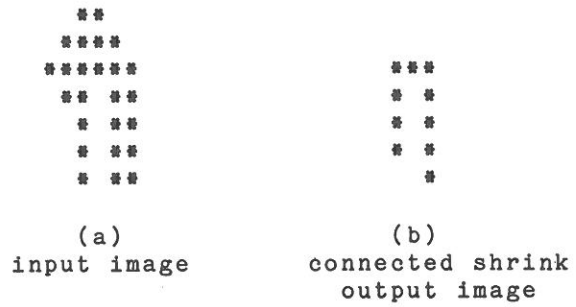


Fig. 4. Illustration of the connected shrink operator applied in a top-down left-right scan using 4-connectivity.

The earliest discussion of connectivity in digital pictures can be found in Rosenfeld (1971). Rutovitz (1966) preceded Rosenfeld in the use of crossing numbers but did not use connectivity in his development. Related algorithms and discussion of connectivity can be found in Levialdi (1972) who introduced a parallel or non-recursive shrinking algorithm for the purpose of counting the number of components in a binary image. This iterative algorithm does not employ the 1-deletability of the Yokoi et al. method; it uses a 2x2 window, rather than a 3x3 window in the shrinking process, but requires the detection of an isolated element during the iterative process so that it may be counted before it is made to disappear by the process. A three-dimensional extension to this non-recursive algorithm can be found in Arcelli and Levialdi (1972). Lobregt, Verbeek, and Groen (1980) discuss a recursive operator for three-dimensional shrinking.

The first discussions of thinning appeared in Hilditch (1969), and Deutsch (1969). These initial insights were later expanded by Fraser (1970), Stefanelli and Rosenfeld (1971), Deutsch (1972), Rosenfeld (1975), and Rosenfeld and Davis (1976). A brief comparison of thinning techniques can be found in Tamura (1978) who suggests that a smoother 8-connected thinning results if 8-deletable pixels are removed from thinning 4-connected curves. Tamura also notes that the thinning of Rosenfeld and Davis (1976) is very sensitive to contour noise when used in the 4-connected mode.

8. PAIR RELATIONSHIP OPERATOR

The pair relationship operator is a non-recursive and has a symbolic data domain. It is a general operator which labels a pixel on the basis of whether that pixel stands in the specified relationship with some neighborhood pixel. An example of a pair relationship operator is one which relabels with a specified label all border pixels which are next to an interior pixel and either can relabel all other pixels with another specified label or leave their labels alone. Formally, a pair relationship operator marks a pixel with the specified label "p" if the pixel has a specified label "1" and neighbors enough pixels having specified label "m." All other pixels it either marks with another specified label or leaves their original labels unmodified.

The pair relationship operator employs two primitive functions. The two-argument function h is able to recognize if its first argument has the value of its second argument. It is defined by:

$$h(a,m) = \begin{cases} 1 \text{ if } a = m \\ 0 \text{ otherwise} \end{cases} \tag{14}$$

For the 4-connected mode, the output value y is defined by:

$$y = \begin{cases} q \text{ if } \sum_{n=1}^{4} h(X_n,m) < \theta \text{ or } X_0 \neq 1 \\ p \text{ if } \sum_{n=1}^{4} h(X_n,m) \geq \theta \text{ and } X_0 = 1, \end{cases} \tag{15}$$

where q can either be a specified output label or the label X_8.

For the 8-connected mode, the output y is defined by:

$$y = \begin{cases} q \text{ if } \sum_{n=1}^{8} h(X_n,m) < \theta \text{ or } X_0 \neq 1 \\ p \text{ if } \sum_{n=1}^{8} h(X_n,m) \geq \theta \text{ and } X_0 = 1 \end{cases} \tag{16}$$

where q can either be a specified output or the label X_8.

9. THINNING OPERATOR

The thinning operator suggested here is defined as a composition of three operators: the mark interior/border operator, the pair relationship operator, and the marked pixel connected shrink operator. It works by marking all border pixels which are next to interior pixels and then deleting (or shrinking) any marked pixel which is deletable. The result of successively applying the thinning operator on a symbolic image is that all regions are symmetrically shrunk down until there are no interior pixels left. What remains is their centerlines as shown in Figure 5. This operator has the nice property that the centerline is connected in exactly the same geometric and topologic way the original figure is connected. For other similar operators which thin without changing geometry or topology, see Davis and Rosenfeld (1975) or Stefanelli and Rosenfeld (1971) or Arcelli and Sanniti di Baja (1978). To implement the operator as the composition of three operators, the mark interior/border operator examines the original symbolic image to produce an interior/border image. The interior border image is examined by the pair relationship operator which produces an image whose pixels are marked if on the original image they were border pixels and were next to interior pixels. The marked pixel image and the original symbolic image constitute the input to the marked pixel connected shrink operator which is exactly like the connected shrink operator except it only shrinks pixels which are deletable and which are marked.

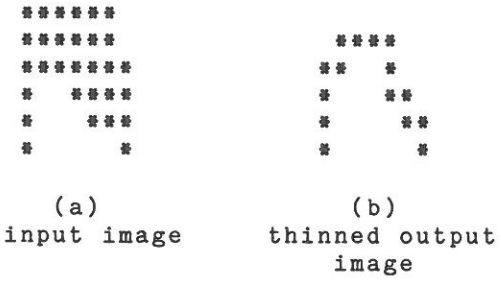


Fig. 5. Illustration of the result of one application of the thinning operator using the 4-connectivity deletability condition.

10. DISTANCE TRANSFORMATION OPERATOR

The distance transformation operator has an implementation as a recursive or as a non-recursive operator. It requires a binary image whose border pixels are labeled with 0 and whose interior pixels are labeled with "i." The purpose of the distance transformation operator is to produce a numeric image whose pixels are labeled with the distance each of them is to its closest border pixel. Distance between two pixels can be defined by the length of the shortest 4-connected path or 8-connected path between them.

As a non-recursive operator, the distance transform can be achieved by successive application of the pair relationship operator. In the first application the pair relationship labels all pixels whose label is "i" and which are next to a pixel whose label is "0" with the label "1." All other pixels keep their labels. In the n^{th} application, the pair relationship operator labels all pixels whose label is "i" and which are next to a pixel whose label is "n-1" with the label "n." When no pixel has the label "i," an application of the pair relationship operator changes no pixel values and the resulting image is the distance transform image. This implementation is related to the one given by Rosenfeld (1968).

Another way of implementing this operator non-recursively is by the use of the primitive function defined by:

$$h(a_0,\ldots,a_N) = \begin{cases} i & \text{if } a_n = i,\ n = 0,\ldots,N \\ \min\{b \mid \text{for some } a_n \le N,\ a_n \neq i,\ b = a_n + 1\} & \\ & \text{if } a_N = i \text{ and there exists } n \text{ such that } a_n \neq i \\ a_N & \text{if } a_N \neq i \end{cases} \tag{17}$$

In the 8-connected mode the output y is defined by $y = h(x_0,x_1,\ldots,x_8)$. In the 4-connected mode, the output symbol y is defined by $y = h(x_0,x_1,x_2,x_3,x_4)$. See Rosenfeld and Pfaltz (1966).

Another way (Rosenfeld and Pfaltz, 1966) of implementing the distance transform involves the application of two recursive operators, the first operator being applied in a left-right top-bottom scan and the second operator being applied in a right-left bottom-top scan. Both operators are based on similar primitive functions. For the first operator the primitive function h is defined by:

$$h(a_1,\ldots,a_N) = \begin{cases} 0 \text{ if } a_N = 0 \\ \min\{a_1,\ldots,a_N\} + 1 \text{ otherwise} \end{cases} \quad (18)$$

In the 8-connected mode, the output symbol y of the first operator is defined by:

$$y = h(x_2,x_7,x_3,x_0).$$

In the 4-connected mode, the output symbol y is defined by:

$$y = h(x_2,x_3,x_0)$$

For the second operator, the primitive function is simply the minimum function. In the 8-connected mode, the output symbol y of the second operator is defined by:

$$y = \min\{x_0,x_1,x_4\}$$

In the second 4-connected mode, the output symbol y is defined by:

$$y = \min\{x_0,x_1,x_4\}$$

11. CONTACT DISTANCES

Contact distances are related to the radius of fusion defined by Rosenfeld and Pfaltz (1968). The radius of fusion for a pixel is the smallest integer n such that after region growing n iterations and region shrinking n + 1 iterations, the pixels retain a non-background label. The radius of fusion concept has the difficulty that using 4-neighborhoods; for example, it is possible for a pair of pixels or a triangle of pixels never to fuse. Its radius of fusion is, therefore, not defined. Defining it by some large number in these cases is artificial.

Contact distance gives a measure of the distance to a pixel's nearest labeled neighbor and it is always defined. To label every pixel with the distance its associated nearest labeled pixel has with its own closest nearest labeled pixel, begin with the original image having some isolated pixels with unique labels and the remainder pixels having the background label. Exactly as done to determine the nearest neighbor sets, perform a region growing to label every pixel with the label of its nearest labeled neighbor. One iteration of a shrink operation on this image can label all border pixels (pixels which have a neighbor with a different label) with

the label 0. Use this image as the input to the distance transformation operator which labels every pixel with its distance to the nearest border. Then mask this distance transformation image with the original image, labeling all pixels with the background label except those pixels having a non-background label on the original image. Pixels having a non-background label on the original image get labeled with the distances associated with their spatial position on the distance transformation image. Finally, region grow the masked image until there are no more pixels with a background label. The resulting image has each pixel labeled with the distance its associated nearest labeled neighbor pixel has with its own nearest labeled pixel.

12. NON-MINIMA-MAXIMA OPERATOR

The non-minima-maxima operator is a non-recursive operator that takes a numeric input image and produces a symbolic output image in which each pixel is labeled with an index 0, 1, 2, or 3 indicating whether the pixel is a non-maximum, non-minimum, interior to a connected set of equal-valued pixels, or part of a transition region (a region having some neighboring pixels greater than and others less than its own value). A pixel whose value is the minimum of its neighborhood and having one neighboring pixel with a value greater than itself may be a minimum or transition pixel but it is certainly a non-maximum pixel. Figure 6 illustrates how a pixel can be its neighborhood maximum, yet not be part of any relative maximum.

The non-minima-maxima operator is based on the primitive function min and max. For the 4-connected case, let $a_0 = b_0 = X_0$ and define

$$\begin{aligned} a_n &= \min \{a_{n-1}, X_0\} \quad n = 1,2,3,4 \\ b_n &= \max \{b_{n-1}, X_0\} \quad n = 1,2,3,4 \end{aligned} \tag{19}$$

The output index 1 is defined by

$$1 = \begin{cases} 0 \text{ (flat)} & \text{if } a_4 = X_0 = b_4 \\ 1 \text{ (non-maximum)} & \text{if } a_4 = X_0 < b_4 \\ 2 \text{ (non-minimum)} & \text{if } a_4 < X_0 = b_4 \\ 3 \text{ (transition)} & \text{if } a_4 < X_0 < b_4 \end{cases} \tag{20}$$

1	1	1	1	4	1	
1	1	1	1	4	1	
1	1	3	3	3	1	
1	1	3	3	3	1	
1	1	3	3	3	1	
1	1	1	1	1	1	
1	1	1	1	1	1	

Fig. 6. Illustration of how a pixel can be its neighborhood maximum yet not be a part of any relative maximum. In its 8-neighborhood, the central 3 is a maximum, yet the flat of 3's it belongs to is a transition region.

For the 8-connected case, let $a_0 = b_0 = X_0$ and define

$$a_n = \min \{a_{n-1}, X_0\} \quad n = 1,2,\ldots 8$$
$$b_n = \max \{b_{n-1}, X_0\} \quad n = 1,2,\ldots 8 \tag{21}$$

The output index l is defined by

$$l = \begin{cases} 0 \text{ (flat)} & \text{if } a_8 = X_0 = b_8 \\ 1 \text{ (non-maximum)} & \text{if } a_8 = X_0 < b_8 \\ 2 \text{ (non-minimum)} & \text{if } a_8 < X_0 = b_8 \\ 3 \text{ (transition)} & \text{if } a_8 < X_0 < b_8 \end{cases} \tag{22}$$

13. RELATIVE EXTREMA OPERATOR

The extrema operators consist of the relative maximum operator and relative minimum operator. They are recursive operators which have a numeric data domain. They require an input image which needs to be accessed but is not changed and an output image which is successively modified. Initially, the output image is a copy of the input image. The operator must be successively applied in a top-

down left-right scan and then a bottom-up right-left scan until no changes are made. Each pixel on the output image contains the value of the highest extrema which can reach it by a monotonic path. Pixels on the output image which have the same value as those on the input image are the relative extrema pixels.

The way the relative maxima operator works is as follows. Values are gathered from those pixels on the output image which corresponds to pixels on the input image which neighbor the given pixel and which have input values greater than or equal to the input values of the given pixel. The maximum of these gathered values are propagated to the given output pixel. The relative minima operator works in an analogous fashion. Figure 7 illustrates the pixel designations for the normal and reverse scans.

The relative maxima operator uses two primitive functions h and max. The four argument function h selects the maximum of its last two arguments if its second argument is greater than or equal to its first argument. Otherwise, it selects the third argument. Hence,

$$h(b,c,d,e) = \begin{cases} \max\{d,e\} \text{ if } c \geq b \\ d \text{ if } c < b \end{cases} \tag{23}$$

The primitive function max selects the maximum of its arguments. In the 8-connected mode, the relative maxima operator lets

$$a_0 = 1_0 \text{ and } a_n = h\,(X_0, X_n, a_{n-1}, 1_n), \; = 1,2,3, \text{ and } 4$$

The output value 1 is defined by $1 = a_4$.

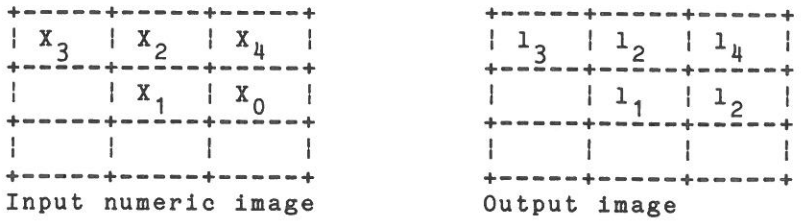
In the 4-connected mode, the operator is

$$a_0 = 1_0 \text{ and } a_n = h(X_0, X_n, a_{n-1}, 1_n), \; n = 1,2,3, \text{ and } 4.$$

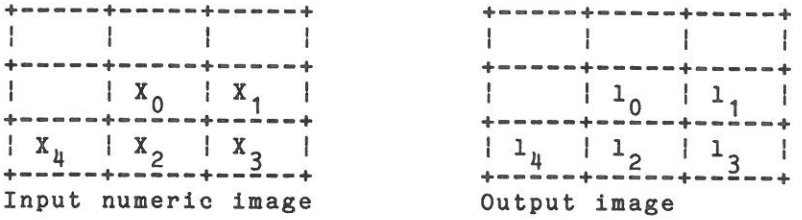
The output value 1 is defined by $1 = a_2$.

The relative minima operator is defined similar to the relative maxima operator with the max function replaced by the min function and all inequalities changed. Hence, for the relative minima operator, h is defined by:

$$h(b,c,d,e) = \begin{cases} \min\{d,e\} \text{ if } c \leq b \\ d \text{ if } c > b \end{cases} \tag{24}$$



(a) Left-right top bottom scans



(b) Right-left bottom top scans

Fig. 7. Illustration of the pixel designations for the recursive operators which require forward and reverse scan and a numeric image, and which recursively produce an output image.

In the 8-connected mode, the relative minima operator lets $a_0 = l_0$ and $a_n = h(X_0, X_n, a_{n-1}, l_n)$, n = 1,2,3, and 4.

The output value l is defined by $l = a_4$.

In the 4-connected mode, the operator lets $a_0 = l_0$ and

$$a_n = h(X_2, X_4, a_{n-1}, l_n), \quad n = 1 \text{ and } 2.$$

An alternative kind of relative extrema operator can be defined using the symbolic image created by the non minima-maxima operator in combination with the original numeric image. Such an operator is a recursive operator and is based on the fact that by appropriate label propagation, all flat regions can be relabeled as transition, minima, or maximum regions and that the pixels originally labeled as non-minima or non-maxima can be relabeled as transition regions or true relative minima or maxima.

The initial output image is taken to be the image created by the minima-maxima operator. Recursive propagation of the labels from one pixel to its neighbor on the output image is performed only if the two labels are not the same and the corresponding two gray tones on the original numeric image are equal.

Let any two neighboring pixels be x and y having respective labels L_x and L_y on the output image. As shown in Figure 8, we have three cases to examine when $L_x \neq L_y$ and $x = y$:

(a) Either L_x or L_y is a flat (0) and the other one is not (1,2,3). In this case, we propagate the non-zero label into the zero label, thereby eliminating pixels marked as flats.

(b) Either L_x is a minimum (1) and L_y is a maximum (2) or vice versa. In this case, since a region constant in tone cannot simultaneously be a minimum and a maximum, mark both pixels as transitions (3).

(c) Either L_x or L_y is a transition (3). In this case, since a region which is constant in tone and has one pixel marked as transition must be a transition region, mark the non-transition region pixel as a transition region, thereby propagating the transition label.

This propagation rule requires one 4-argument primitive function h defined by:

$$h(x,y,a,b) = \begin{cases} a \text{ if } x \neq y \\ 3 \text{ (transition) if } x = y \text{ and} \\ \qquad (a = 3) \text{ or } (b = 3) \text{ or} \\ \qquad (a = 1 \text{ and } b = 0) \text{ or} \\ \qquad (a = 0 \text{ and } b = 1) \\ 2 \text{ (flat) if } x = y \text{ and } (a = 2 \text{ and } b = 2) \\ 1 \text{ (non-minima) if } x = y \text{ and} \\ \qquad (a = 1 \text{ and } b = 2) \text{ or} \\ \qquad (a = 2 \text{ and } b = 1) \\ \qquad \text{or } (a = 1 \text{ and } b = 1) \\ 0 \text{ (non-maxima) if } x = y \text{ and} \\ \qquad (a = 0 \text{ and } b = 0) \text{ or} \\ \qquad (a = 0 \text{ and } b = 2) \\ \qquad \text{or } (a = 2 \text{ and } b = 0) \end{cases} \tag{25}$$

Values of pixels in the original numeric input image are denoted by X_n. Values of pixels in the non-minima-maxima labeled image are denoted by l_n. For the operator using 4-connectedness and the standard 3x3 neighborhood designations, let

$$a_0 = l_0$$

and define

$$a_n = h(X_0, X_n, a_{n-1}, l_n), \quad n = 1,2,3, \text{ and } 4.$$

Numeric Neighbor | x | y |

Symbolic Neighbor | a | b | — Propagation when x = y → | c | c | Propagation Result

b labels / a labels	non-mixima	non-minima	flat	transition
non-maxima	non-maxima	transition	non-maxima	transition
non-minima	transition	non-minima	non-minima	transition
flat	non-maxima	non-minima	flat	transition
transition	transition	transition	transition	transition

Fig. 8. Illustration of the propagation table for the recursive relative extrema operator. The table gives the propagation label C for any pair, a,b of labels of neighboring pixels.

The output l is a_4.

For the operator using 8-connectedness, let

$$a_0 = l_0$$

and define

$$a_n = h(X_0, X_n, a_{n-1}, l_n),\ n = 1,\ldots,8.$$

The output l is a_8.

Propagation can also be achieved by using the forward and reverse scan technique. In this case, the left-right top-bottom scan is alternated with the right-left bottom-top scan. Using the pixel designations in Figure 3 for the operator using 4 -connectedness, let

$$a_0 = 1_0$$

and define

$$a_n = h(X_0, X_n, a_{n-1}, 1_n), \quad n = 1,2,3, \text{ and } 4.$$

The output 1 is a_4.

14. CONNECTED COMPONENTS OPERATOR

The connected components operator is recursive and has a symbolic data domain. Its purpose is to assign with a unique label all pixels belonging to the same maximally connected component. There are a variety of ways of determining connected components including some two-pass algorithms which may require memory for large internal tables. Discussion of such connected component operators can be found in Rosenfeld and Pfaltz (1966). The connected components operator described here requires only a small amount of memory and must repeatedly scan the image until it makes no changes. It also differs from other neighborhood operators in that it requires an internal state which must be remembered and which can change or stay the same after each application of the operator. It also is an operator that can be applied by alternating between top-down, left-right scans (forward scans) and bottom-up, right-left scans (reverse scans). Figure 7 illustrates the positioning of the pixels for the 2x3 windows in each of these types of scans.

We will assume that the symbolic input image has each pixel labeled with a label from a set L. Corresponding to each label $m \in L$ is a set S_m of possible labels for the individual connected components of all regions whose pixels are labeled "1." For notational convenience we will write $S_m(k)$ to denote the k^{th} label from the set S_m. For each label m, the connected components operator must assign a unique label to all maximally connected sets whose pixels originally had the label m.

We assume that for each label $m \in L$ $S_m = \phi$, that $S_m \cap S_n = \phi$ when $m \neq n$, and that each of the S_m sets is linearly ordered. The memory required by the operator is a function f which for each $m \in L$ specifies a label from the set S_m. The specified label from S_m is the next label which can be used to label a connected region

of type "m," that is, a connected region whose pixels have the input label "m."

The connected components operator works by propagating labels. Let d be the label in a pixel and let c be the label in one of its neighboring pixels. Under certain conditions the label c replaces the label d. There are five cases governing this process. The case and action are listed in Figure 9. Notice that for propagation condition (2), the next not yet used label from S_d must be generated. This requires that the operator have access to a function f(m) which is an index to the next not yet used label from S_m.

The operator is based on a 2-argument primitive function h which propagates labels in one of two ways as indicated by the case analysis. If a pixel and its neighbor have a label from the same set S_m, then h propagates the minimum of the labels. If a pixel has not been labeled, then h propagates a label if appropriate, from a neighboring pixel 1. If the neighboring pixel also has not been labeled it starts a new label. The function h is defined by:

$$h(c,d) = \begin{cases} c & \text{if } d \in L,\ c \in S_d \\ \text{next unused label for region type } d & \text{if } d \in L,\ c \notin S_d \\ d & \text{if } d \in S_m \text{ for some } m \in L \text{ and } c \in S_m \\ \min\{c,d\} & \text{if } d \in_m S \text{ and } c \in_m S \text{ for some } m \in L \end{cases} \tag{26}$$

The operator works the following way. If for some m, $X_0 \in S_m$, then set $a_0 = X_0$ and leave f(m) unchanged. If $X_0 \in L$, set $a_0 = S_x(m(X_0))$ and $m(X_0)$, set $m(X_0) + 1$.

For 4-connectedness, define $a_n = h(X_{n-1}, a_{n-1})$, n = 1 and 2. The output label is given by a_2.

For 8-connectedness define $a_n = h(X_{n-1}, a_{n-1})$, = 1,2,3, and 4. The output label is given by a_4.

Two Neighboring Pixels

Condition	Propagation
$d \in L$, $c \in S_d$	d <--c
$d \in L$, $c \notin S_d$	d <-- next unused label for region type d
$d \in S_m$ for some $m \in L$, $c \in S_m$	d <-- d
$d \in S_m$ and $c \in S_m$ for some $m \in L$	d <-- min{c,d}

Fig. 9. The care analysis of what label gets propagated into pixel d by the connected components operator.

15. REACHABILITY OPERATOR

The reachability operators consist of the descending reachability operator and the ascending reachability operator. The operators are recursive and require a numeric input and a symbolic image used for both input and output. Initially the symbolic image has all relative extrema pixels marked with unique labels (relative maxima for the descending reachability case and relative minima for the ascending reachability case). The unique labeling of extrema can be obtained by the connected components operator operating on the relative extrema image. Pixels which are not relative extrema must be labeled with the background symbol "g." The reachability operator, like the connected component operator, must be iteratively and alternately applied in a top-down, left-right scan followed by a bottom-up, right-left scan until no change is produced. The resulting output image has each pixel labeled with the unique label of the relative extrema region that can reach it by a monotonic path if it can only be reached by one extrema. If more than one extrema can reach it by a monotonic path, then the pixel is labeled "c" for common region.

The operator works by successively propagating labels from all neighboring pixels which can reach the given pixel by monotonic paths. In case of conflicts, the label "c" is propagated. Figure 7 illustrates the pixel designations for the reachability operator. To do this, the descending reachability operator employs the four-argument primitive function h. Its first two arguments are labels

from the output image and its last two arguments are pixel values from the input image. It is defined by:

$$h(a,b,x,y) = \begin{cases} a \text{ if } (b = g \text{ or } a = b) \text{ and } x < y \\ b \text{ if } a = g \text{ and } x < y \\ c \text{ if } a \neq g \text{ and } b \neq g \text{ and } a \neq b \text{ and } x < y \\ a \text{ if } x > y \end{cases} \tag{27}$$

The operator uses the primitive function h in the 8-connected mode by letting $a_0 = l_0$ and defining $a_n = h(a_{n-1}, l_n, X_0, X_n)$. n = 1,2,3, and 4. The output label l is defined by $l = a_4$.

The 4-connected mode sets $a_0 = l_0$ and defines $a_n = h(a_{n-1}, l_n, X_0, X_n)$, n = 1 and 2. The output label l is defined by $l = a_2$.

The ascending reachability operator is defined just as the descending reachability operator except that the inequalities are changed. Hence, for the ascending reachability operator, the primitive function h is defined by:

$$h(a,b,x,y) = \begin{cases} a \text{ if } (b = g \text{ or } a = b) \text{ and } x > y \\ b \text{ if } a = g \text{ and } x > y \\ c \text{ if } a \neq g \text{ and } b \neq g \text{ and } a \neq b \text{ and } x > y \\ a \text{ if } x < y \end{cases} \tag{28}$$

16. CONCLUSION

We have reviewed a variety of neighborhood operators from the point of view of the basic primitive functions whose composition generates the required operator. Many of the primitive functions can be implemented as table-lookup functions. The remainder can be implemented with only a small amount of sequential calculation using the standard logical, comparison, or arithmetic functions on a VLSI processor. This suggests the timely appropriateness of considering VLSI implementations of neighborhood operators for image processing.

17. REFERENCES

Arcelli, C., and Levialdi, S., "Parallel Shrinking by Three Dimensions," Computer Graphics and Image Processing 1:21-30 (1972).

Arcelli, C., and Sanniti di Baja, G., "On the Sequential Approach to Medial Line Transformation," IEEE Trans. Systems, Man and Cybernetics SMC-8(2):139-144 (1978).

Deutsch, E., "Comments on a Line Thinning Scheme," Computer Journal 12:412 (Nov. 1969).

Deutsch, E., "Thinning Algorithms on Rectangular, Hexagonal and Triangular Arrays," Communication of Association for Computing Machinery 15(9):827-837 (Sept. 1972).

Fraser, J., "Further Comments on a Line Thinning Scheme," Computer Journal 13:221-222 (May 1970).

Hilditch, C., "Linear Skeletons from Square Cupboards," Machine Intelligence (Meltzer and Michie, eds.), University Press, Edinburgh (1969), pp. 403-420.

Levialdi, S., "On Shrinking of Binary Picture Patterns," Communication of Association for Computing Machinery 15:7-10 (1972).

Lobregt, S., Verbeck, P.W., and Groen, F.C.A., "Three Dimensional Skeletonizations: Principle and Algorithm," IEEE Trans. Pattern Analysis and Machine Intelligence PAMI-2(1):75-77 (1980).

Rosenfeld, A., "A Characterization of Parallel Thinning Algorithms," Information and Control 29:286-291 (Nov. 1975).

Rosenfeld, A., "Connectivity by Digital Pictures," Journal of the Association for Computing Machinery 17(1):146-160 (Jan. 1970).

Rosenfeld, A., and Pfaltz, J., "Distance Function on Digital Pictures," Pattern Recognition 1(1):33-61 (July 1968).

Rosenfeld, A., and Davis, L., "A Note on Thinning," IEEE Trans. Systems, Man and Cybernetics SMC-6(3):226-228 (March 1976).

Rosenfeld, A., and Pfaltz, J., "Sequential Operation in Digital Picture Processing," Journal of the Association for Computing Machinery 12(4):471-494 (Oct. 1966).

Rutovitz, D., "Pattern Recognition," J. Royal Statistics Soc. 129(A): 504-530 (1966).

Stefanelli, R., and Rosenfeld, A., "Some Parallel Thinning Algorithms for Digital Pictures," Journal of the Association for Computing Machinery 18(2):255-264

Tamura, H., "A Comparison of Line Thinning Algorithms from Digital Geometry Viewpoint," 4th IJCPR, Tokyo, Japan, 1978.

Yokoi, S., Toriwaki, J., and Fukumura, T., "An Analysis of Topological Properties of Digitized Binary Pictures Using Local Features," Computer Graphics and Image Processing 4:63-73 (1975).