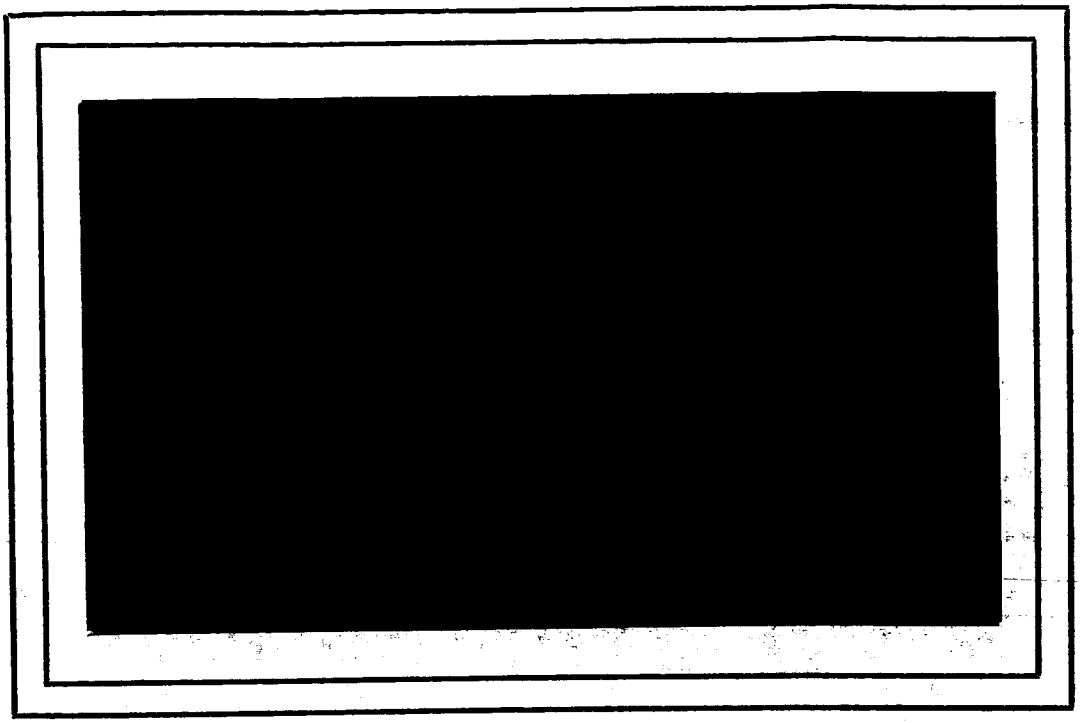


UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND

Technical Report TR-65-18 May 1965

NsG-398

Pattern Recognition: IV. Sequential Operations in Digital Picture Processing*

by

Azriel Rosenfeld

John L. Pfaltz

*This work was supported by the National Aeronautics and Space Administration Grant NsG-398 to the Computer Science Center, University of Maryland, College Park, Maryland

1. Introduction

Computer programs for processing digitized pictorial information have received increasing attention in recent years. Much of the work done in this field has involved performing "local" operations on picture "neighborhoods." As several investigators have shown, a wide variety of picture processing transformations can be accomplished by applying such operations independently, or "in parallel", to each element of the given picture.

This paper suggests that local operations performed on picture elements taken in a definite sequence, using at each step the results obtained by operating on the preceding elements in the sequence, may be preferable to the "parallel" approach in some cases. It is shown that the parallel and sequential approaches are mathematically equivalent, and that the latter should be competitive in processing time required if a sequential computer is used.

"Sequential" local operations for performing two basic picture transformations are next described. The first of these labels the connected components of any given picture subset, while the second determines the "distance" (in a certain sense) from every picture element to the nearest element of a given subset. In connection with the latter transformation, a "skeleton" subset is defined which can be used in place of the given subset to generate the same transformed picture by applying the "reverse" local operations sequentially. Examples of the outputs of sequential computer programs which perform these transformations are given.

In the concluding sections of this paper, various applications of these basic picture transformations to the analysis of picture subsets are indicated. Programs

are described which construct the graphs corresponding to dissections of a picture into regions, and which determine the orders of connectivity of the multiply connected regions. Two approaches to the discrimination of elongated from non-elongated regions or parts of regions using the "distance" transformation are presented, one of them involving components of the skeleton subset.

The research described in this paper was supported in part by National Aeronautics and Space Administration Grant NsG-398 to the University of Maryland. The assistance of Mr. David Wilson, who made several significant contributions to the development of the "distance" transformation program, is gratefully acknowledged. Early versions of both this and the connected component program were developed by the senior author and his colleagues at the Budd Information Sciences Center, McLean, Va., with the partial support of NASA under Contract NAS5-3461 with The Budd Company. This initial work was accomplished through the efforts of Mr. James N. Orton of Budd, with the assistance of Messrs. Ernest W. Smith and Bernard Altschuler.

2. Sequential and parallel neighborhood operations

2.1. Operations on digitized pictures

A digitized picture, for the purposes of this discussion, is a finite rectangular array of "points" or "elements", each of which has associated with it one of a discrete finite set of "values." If the array has m rows and n columns, any "point" in it is specified by a pair of numbers i, j ($1 \leq i \leq m$, $1 \leq j \leq n$) denoting its row and column. The picture can thus be represented as an m by n matrix in which each entry a_{ij} has one of the "values," say $v_1, \ldots, v_k$. In practice, m and n may be of the order of 2^3 to 2^{15} and k may range from 2^1 to 2^{10}.

By an operation on a digitized picture is meant a function which transforms a given picture matrix into another one. A general function of this type has mn numerical arguments (one for each position in the matrix), and is correspondingly difficult to handle computationally. For practical purposes, it is desirable to work with operations on digitized pictures which can be defined in terms of functions having considerably fewer arguments.

By a local operation or neighborhood operation on a picture is meant a function which defines a value for each element in the transformed picture in terms of the values of the corresponding element and a small set of its neighbors in the given picture. For example, such operations can be defined using a neighborhood which consists of the given element and its eight immediate neighbors; an operation of this type has only nine arguments and is of the form

$$a^*_{i,j} = f(a_{i-1,j-1}, a_{i-1,j}, a_{i-1,j+1}, a_{i,j-1}, a_{i,j},$$
$$a_{i,j+1}, a_{i+1,j-1}, a_{i+1,j}, a_{i+1,j+1})$$

In what follows we shall consider only operations of this type.*

Local operations can be used not only to generate transforms of a given picture, but also to define distinguished subsets of the picture. For example, one can define a picture as having a certain local property at a given point if the values of the point and its neighbors satisfy certain conditions (all the same, half the same and the rest different, etc.). Given any such local property, one can always define a local operation which assigns (say) the value 1 to points which have the property, and the value 0 to points which do not have it. This local operation is in effect the characteristic function of the set of points which have the property. The transformed picture which it produces can be used to select the points of the original picture which have the property by taking the logical "and" of the original and the transform.

As the last remark suggests, one can also consider local operations which involve more than one picture at a time. However, from a formal standpoint no gain

---

*The function f is not defined for "border" picture elements which do not have all eight neighbors. To avoid such exceptions, one can "augment" the picture matrix by adding a zeroth row and column, an m+1st row and an n+1st column, giving these fictitious elements a value which do not occur in the "real" picture array, and defining the function appropriately when some of its arguments have this value. Alternatively, one can add to the definition of f the phrase "for whichever of these elements is defined."

in generality results from this. In fact, suppose that one has two pictures (a_{ij}), (b_{ij}) of the same dimensions (m by n), with element values $v_1,\ldots v_r$ and $u_1,\ldots u_s$, respectively; one can then define a new m by n picture (c_{ij}) of the form

$$c_{ij}=f(a_{i-1,j-1},\ldots,a_{i+1,j+1};b_{i-1,j-1},\ldots,b_{i+1,j+1})$$

However, any such (c_{ij}) can always be defined using local operations on a single picture. For example, consider the m by n picture (A_{ij}), where $A_{ij}=2^{a_{ij}}3^{b_{ij}}$; let $\varphi_2(n)$, $\varphi_3(n)$ be the highest powers of 2 and 3 which divide n, respectively, where n is any integer. Then

$$c_{ij}=f(\log_2(\varphi_2(A_{i-1,j-1})),\ldots,\log_2(\varphi_2(A_{i+1,j+1}));$$
$$\log_3(\varphi_3(A_{i-1,j-1})),\ldots,\log_3(\varphi_3(A_{i+1,j+1}))$$

is the result of applying a local operation to (A_{ij}).

2.2. Parallel operations

Many useful transformations of a given picture can be achieved by performing a single local operation, or at most a few of them in succession, on each point of the picture. For example, a "noisy" picture can often be effectively "smoothed," or an "unsharp" picture "enhanced," by a single neighborhood operation which takes a local average or computes a finite - difference Laplacian. Similarly, a picture which contains thick "roads" ("lines" or "curves" of points having given values) can be "thinned" by iterating a "border element deletion" operation, perhaps alternated with a smoothing operation, where the number of iterations required is relatively small since the roads are narrow compared to the picture size. Transformations of these types have been demonstrated by Dinneen [1], Kirsch [2], Unger [3], Narasimhan [4] and others. These operations can also be used to define picture subsets consisting of points which have "smooth" or "broken" neighborhoods, lie on "edges" or "roads," etc.

Each of the local operations used in the examples just given is performed independently on every point of the picture. The arguments $(a_{i-1,j-1},\ldots,a_{i+1,j+1})$ are always the original picture matrix values; the new values $a^*_{ij}=f(a_{i-1,j-1},\ldots,a_{i+1,j+1})$ are stored, but are not used until the operation has been performed for every (i,j), when they then become arguments for the next operation (if any). Since the sequence in which the points are processed is thus entirely irrelevant, the operation can be thought of as being performed "in parallel," simultaneously for every picture point. Extensive consideration has in fact been given to the design of computers

which actually do perform identical operations simultaneously on each of a large number of stored quantities. Even when processing digitized pictures on conventional sequential computers, many investigators have used programs which simulate the operation of such "parallel" machines.

The wide variety of picture transformations which can be performed using local operations applied in parallel has given rise to the widespread belief that this approach is optimum for local picture processing in general. This paper makes the countersuggestion that an alternative type of processing, in which sequential application of local operations plays a crucial role, is equally general in scope, and may even have significant advantages, particularly when processing is being done on a sequential computer.

The concept of a sequentially applied neighborhood operation will now be defined, and the relative merits of the parallel and sequential approaches considered.

2.3. Sequential operations

Suppose that a local operation is applied to the points of a digitized picture in some definite sequence. For simplicity, suppose that the points are processed row by row beginning at the upper left - that is, in the sequence $a_{11}, a_{12}, \cdots a_{1n}, a_{21}, \cdots, a_{2n}, \cdots, a_{m1}, \cdots, a_{mn}$. Unlike the cases described in Section 2.2, however, suppose that as soon as a point is processed, its new value rather than the original value is used in processing and succeeding points which have it as neighbor. If this is done, the general form of the operation becomes

$$a^*_{i,j} = f(a^*_{i-1,j-1}, a^*_{i-1,j}, a^*_{i-1,j+1}, a^*_{i,j-1}, a_{i,j},$$
$$a_{i,j+1}, a_{i+1,j-1}, a_{i+1,j}, a_{i+1,j+1}),$$

since points (i-1,j-1), (i-1,j), (i-1,j+1) and (i,j-1) have already been processed, while the remaining points have not yet been processed. Such an operation will be called sequential.

At first glance, this type of operation seems more complex, and hence presumably less basic, than the "parallel" type, which uniformly uses "old" values until the entire picture has been processed. However, it is easily shown that the two types of operation are entirely equivalent in the sense of the following

Theorem. Any picture transformation that can be accomplished by a series of parallel local operations can also be accomplished by a series of sequential local operations, and conversely.

A proof of this Theorem is given in Appendix A.

In the proof, it is shown that any parallel local operation is equivalent to just two sequential local

operations; but a general sequential local operation is matched only by a sequence of 2mn(m+n-1) parallel local operations. It is possible, of course, that this figure is not a minimum even in the general case. In fact, one can often obtain the result of a sequential operation using relatively few parallel operations which produce the result without following the stepwise progress of the sequential operation. However, it at least appears plausible that there exist picture transformations which are more efficiently performed using sequentially applied operations.

To make this comparison more explicit, suppose that it takes time T to perform a single local operation. A sequential computer can then perform such an operation in the neighborhood of every element of an m by n picture in time mnT; there should be essentially no difference between the cases in which the operation is performed "in parallel" (that is, using old element values at every step) or "sequentially" (using new values insofar as avaliable). Let F be a picture transformation which can be performed by a single sequentially applied local operation, but which requires 2mn(m+n-1) local operations in parallel. Then the sequential computer can perform F "sequentially" in time mnT, but requires time $2m^2n^2(m+n-1)T$ to perform it "in parallel." Furthermore, suppose given a parallel computer which can perform a local operation in time T on every picture neighborhood simultaneously. Even this computer requires time 2mn(m+n-1)T to perform F "in parallel," so that the sequential computer can still perform it "sequentially" in less time.

The figures just given should by no means be interpreted as implying that sequential computers are generally faster than parallel computers at performing picture pro-

cessing tasks. These figures apply only to an operation which cannot be performed except by processing the picture element by element in sequence. Many operations, even if they have very simple definitions in terms of such sequential processing, can also be accomplished in other ways. For example, the distance transformation defined in Section 4 requires just two sequentially applied local operations, involving a total of 2mn individual operations on picture element neighborhoods, or a processing time of 2mnT. On the other hand, it is easily shown that this transformation can also be accomplished by applying the local operation

$$f(a_{i,j}) = \min(a_{i-1,j}, a_{i+1,j}, a_{i,j-1}, a_{i,j+1}) + 1$$

m+n times to every picture element. If this non-sequential procedure is carried out on a sequential computer, mn(m+n) individual local operations must be performed; this is (m+n)/2 times as many as needed for the method of Section 4, with a processing time of mn(m+n)T. However, if a parallel computer is available, the required processing time is only (m+n)T, a saving by a factor of 2mn/(m+n). Thus if m=n, parallel processing on a parallel computer is n times faster than sequential processing on a sequential computer; but this in turn is n times faster than "parallel" processing on a sequential computer even when this efficient parallel method of performing the transformation is used.

3. Sequential operations for connected component discrimination

This section describes a set of sequentially applied local operations which "labels" the connected components of any given picture subset. For simplicity, it is assumed that the given subset consists of picture elements which have value 0, while every other element has value 1. The results can be immediately extended to subsets consisting of elements which have any given property; it suffices to first transform the picture using the characteristic function of the complementary property.

3.1. Connectivity

A subset of a digitized picture will be called connected if for any two points P and Q of the subset there exists a sequence of points $P=P_0,P_1,P_2,\ldots,P_{n-1},P_n=Q$ such that P_i is a neighbor of P_{i-1}, $1 \leq i \leq n$. In Figure 1a, the set of shaded points and the set of unshaded points are both connected; in Figure 1b, the shaded points are connected but the unshaded points are not; in Figure 1c, the unshaded points are connected but the shaded points are not. For these examples, the definition agrees with the intuitive concept of connectivity. On the other hand, both the shaded and unshaded points in Figure 1d are connected, which runs counter to intuition. This results from the fact that a point is connected to any of its eight neighbors, including the diagonal ones. If connectivity were redefined to require that P_i be one of the four horizontal and vertical neighbors of P_{i-1}, both the shaded and unshaded points in Figure 1 would become disconnected, which is still not consistent with intuition. The "paradox" of Figure 1d can be rephrased as follows: If the "curve" of shaded points is connected ("gapless"), it does not disconnect its interior from its exterior; if it is totally disconnected, it does disconnect them. In spite of the paradoxes associated with this neighborhood definition of connectivity, it is still a useful concept; paradoxes arise only when the connectivity of isolated strings of diagonally adjacent elements is considered.

In general, a subset of a picture (say the subset of "0" points) consists of a number of connected parts or components.* The problem of distinguishing among these

*Formally, these components are the equivalence classes of picture points defined by the relation "is connected

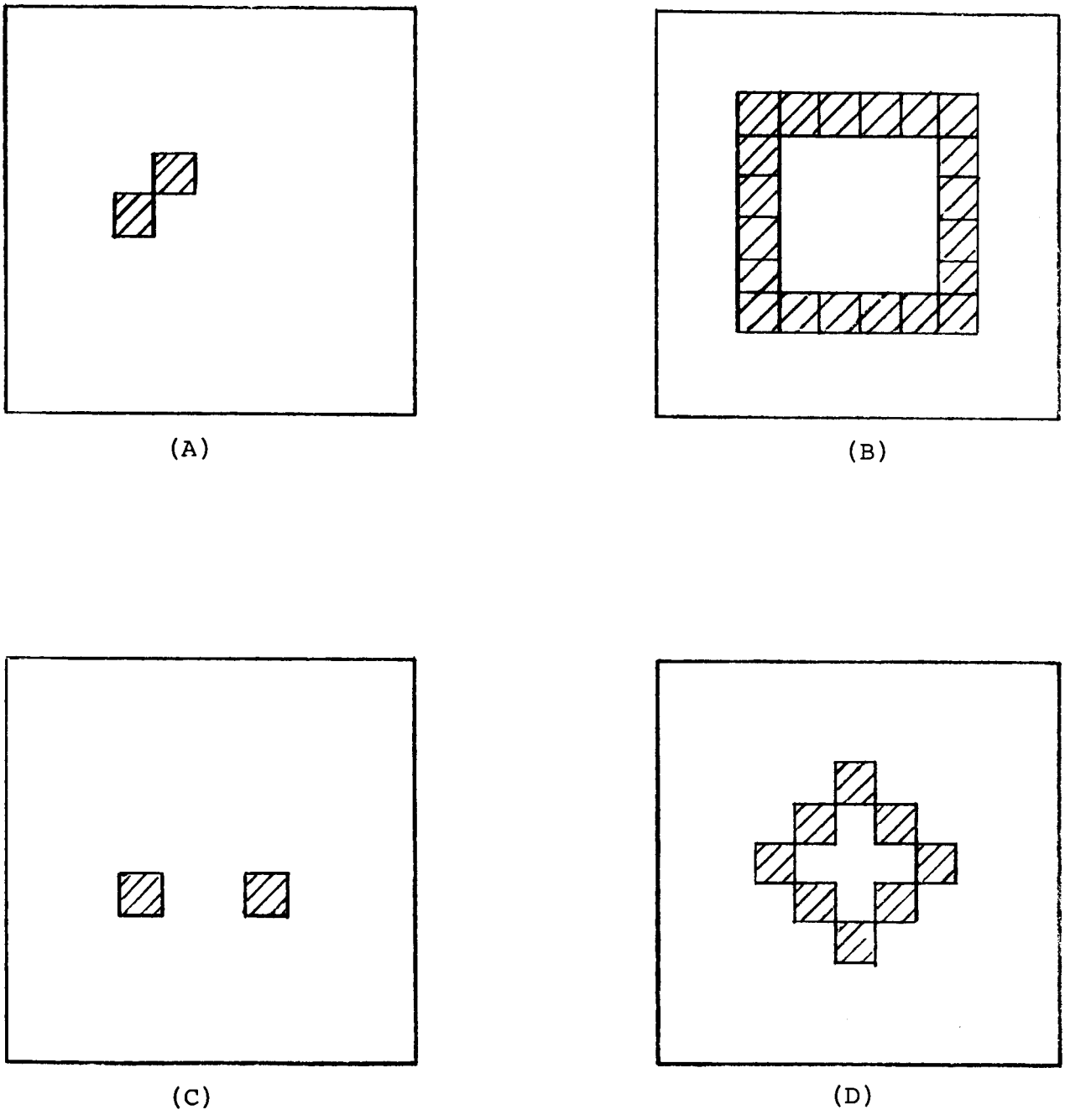


Figure 1. Examples of Connected and Non-connected Sets

components will now be considered. Specifically, it is desired to construct a transformed picture in which the "0" points have new values $v_1,\ldots,v_k$ (positive integers each > 1), two points having the same value if and only if they belong to the same connected component of "0" points on the original picture.

Neighborhood operations lend themselves naturally to the study of connectivity, since it is defined in terms of neighbors. If the operations are applied in parallel, it is not easy to distinguish among connected components, since the parallel operations treat every point of the given set identically. Using sequentially applied operations, however, one can "track" each connected region, assigning a value to each point of it as the tracking proceeds. If two of the tracked regions merge, a special value is assigned. Further processing is then applied to these special values so as to eliminate redundant values from the picture.

In the next subsection a set of sequential local operations is defined which assigns to every point of each connected region on the picture a value which "labels" the region. In the following subsection a computer program is described which generally follows this sequence of operations but which does not adhere strictly to the requirement that only local operations are permitted, and which in consequence is considerably more efficient.

---

to" (that is: "is a neighbor of a neighbor...of a neighbor of"), which is evidently an equivalence relation.

3.2. Sequential operations for determining connected components of a picture

In the discussion below, it is convenient to apply neighborhood operations to the picture using more than one element sequence. The following four sequences are particularly useful:

(1) The forward raster sequence: (1,1),...,(1,n), (2,1),...,(2,n),...,(m,1),...,(m,n) as in Section 2.3

(2) The backward raster sequence: (m,n),...,(m,1), (m-1,n),...(m-1,1),...,(1,n),...(1,1), the exact reverse of (1)

(3) The forward zigzag sequence: (1,1),...,(1,n), (2,n),...,(2,1),(3,1),...,(3,n),...,(m,1),..., (m,n) if m is odd;...,(m,n),...,(m,1) if m is even

(4) The backward zigzag sequence: The exact reverse of (3)

We begin by defining a local operation f which, when it is applied to the picture $(a_{i,j})$ in forward raster (or zigzag) sequence,

(a) Leaves the 1's in the picture unaltered

(b) Tags each "new" 0 (that is, each 0 which has only 1's as predecessor neighbors*) with a unique label $2^i 3^j$ (these values are clearly all distinct for all $1 \leq i \leq m$, $1 \leq j \leq n$). Such a picture element is potentially part of a new connected component.

(c) Tags each 0 whose predecessor neighbors all have

---

*For the forward raster sequence, the "predecessor neighbors" of $a_{i,j}$ are $a_{i-1,j-1}$, $a_{i-1,j}$, $a_{i-1,j+1}$ and $a_{i,j-1}$, or whichever of them exist.

the same label with that label, indicating that it too is part of that connected component

(d) Tags each 0 whose predecessor neighbors have two different labels with the special label $2^a3^b5^c7^d$, indicating that the two labels 2^a3^b and 2^c3^d are redundant since the two "connected components" which they represent have merged. (It will be shown below that the predecessor neighbors cannot have more than two different labels, so that cases (a-d) exhaust all possibilities.)

Specifically, for all $a_{i,j}$ in the picture, $f(a_{i,j})$ is an integer of the form $2^w3^x5^y7^z$, where

a) If $a_{i,j}=1$, then $w=x=y=z=0$, so that $f(a_{i,j})=1$

b) If $a_{i,j}=0$, and $f(a_{i-1,j-1})=f(a_{i-1,j})=f(a_{i-1,j+1})=f(a_{i,j-1})=1$*, then $w=i$, $x=j$, $y=z=0$, so that $f(a_{i,j})=2^i3^j$

c) If $a_{i,j}=0$ and case (b) does not apply, but there exists a pair of positive integers (a,b) such that each of these f's is either 1 or $2^a3^b5^h7^k$ or $2^p3^q5^a7^b$ for some h,k or p,q, then $w=a$, $x=b$, $y=z=0$, so that $f(a_{i,j})=2^a3^b$

d) If $a_{i,j}=0$, and cases (b-c) do not apply, but there exist two pairs of positive integers (a,b) and (c,d) such that each of these f's is either 1, $2^a3^b5^h7^k$, $2^c3^d5^p7^q$, $2^s3^t5^a7^b$, or $2^u3^v5^c7^d$, then $w=a$, $x=b$, $y=c$, $z=d$, so that $f(a_{i,j})=2^a3^b5^c7^d$

*Strictly speaking, in defining f we should speak of these four elements, rather than their f's, as being 1. For the sake of clarity, however, the definitions given in this section describe the results of applying f in sequence, so that when f is being applied to $a_{i,j}$, it has already been applied to these four elements.

Lemma. When f is applied to a picture in forward raster sequence, cases (a-d) are the only possibilities.

Proof: We must show that the highest powers of 2 and 3, 5 and 7 dividing the four predecessor neighbor f's can never be different to the point where even case (d) does not apply. Let these f's be $2^{w_1}3^{x_1}5^{y_1}7^{z_1},\ldots,2^{w_4}3^{x_4}5^{y_4}7^{z_4}$, and suppose that the Lemma is true for all elements preceding $a_{i,j}$. Since $a_{i-1,j-1}$ is a predecessor neighbor of $a_{i-1,j}$, one of the following possibilities must hold:

a) $f(a_{i-1,j})=1$

b) $f(a_{i-1,j-1})=1$

c)-d) w_2,x_2 or $y_2,z_2 = w_1,x_1$ or y_1,z_1

Similarly, since $a_{i-1,j}$ is a predecessor neighbor of $a_{i-1,j+1}$, if neither of their f's is 1 we must have w_2,x_2 or $y_2, z_2=w_3, x_3$ or y_3,z_3; and since $a_{i-1,j}$ is a predecessor neighbor of $a_{i,j-1}$, if neither f is 1 we must have w_2,x_2 or $y_2,z_2=w_4,x_4$ or y_4,z_4. Hence if $f(a_{i-1,j})\neq 1$ we can take $a,b,c,d=w_2,x_2,y_2,z_2$ (ignore the c,d if $y_2=z_2=0$). Furthermore, since $a_{i-1,j-1}$ is a predecessor neighbor of $a_{i,j-1}$, if neither of their f's is 1 we have w_1,x_1 or $y_1,z_1=w_4,x_4$ or y_4,z_4. Let a,b be the common pair, c,d=either of w_3,x_3 or y_3,z_3 to cover the case where $f(a_{i-1,j})=1$. Since the Lemma is trivially true for $a_{1,1}$ (indeed, for any $a_{1,j}$), this completes the proof.

The Lemma shows that the forward raster sequence application of the function f to the picture $(a_{i,j})$ is well defined. The resulting transformed picture now has one or more unique labels of the form 2^a3^b designating the points of each connected component, and special labels

of the form $2^a3^b5^c7^d$ designating points where two of these "subcomponents" merge, indicating that one of their labels is redundant.

It now remains to further process the picture so as to eliminate all the redundant labels. This will be done as follows: For each label of the form $2^a3^b5^c7^d$ in the picture, all 2^c3^d's and 5^c7^d's are systematically replaced by 2^a3^b's. Each time this is done, the number of labels 2^c3^d which are redundant with other labels is reduced by one. A sufficient number of repetitions of this process thus eventually eliminates all redundancies, leaving just one unique label for each connected component.

We proceed by defining two sequences of functions f_r, g_r (r=1,2,...). These functions will operate as follows on the picture:

a) Nothing is done until the first element which shows a redundancy (i.e., which is divisible by 5) and which has not been processed by the preceding f's and g's is reached; let this element be $2^a3^b5^c7^de$, where 2,3,5,7 do not divide e

b) The value of this element is divided by 5^c7^d, eliminating the redundant label (c,d).

c) This and all subsequent elements (in the sense of the forward zigzag sequence) are tagged by multiplying their values by π_r, a product of four primes not previously used as tags, raised to the exponents a,b,c,d, respectively. This procedure "carries" the values a,b,c,d to all subsequent points of the picture so that they can be used for processing these points as necessary. Whenever an element is found whose original label involves (c,d), it is replaced by (a,b). This

completes the application of f_r to the picture.

d) All preceding elements (=subsequent elements in the sense of the backward zigzag sequence) are then processed as in step (c). This constitutes the application of g_r.

Specifically, f_r, defined as follows, is applied to the picture in forward zigzag sequence:

a) If $a_{i,j}$ is not divisible by 5, and p_{4r+1} (the 4r+1st prime) does not divide any of the f_r's of its predecessor neighbors (in the sense of the forward zigzag sequence), then $f_r(a_{i,j})=a_{i,j}$

b) If $a_{i,j}$ is divisible by 5, and p_{4r+1} does not divide any of these f_r's, let $a_{i,j}=2^a3^b5^c7^de$, where 2,3,5,7 do not divide e; then $f(a_{i,j})=2^a3^bp_{4r+1}^ap_{4r+2}^bp_{4r+3}^cp_{4r+4}^d$ ($=2^a3^b\pi_r$ for short)

c) If p_{4r+1} does divide one of these f_r's, let $a_{i,j}=2^w3^x5^y7^ze$, where 2,3,5,7 do not divide e; then

(1) If neither (w,x) nor (y,z)=(c,d), $f_r(a_{i,j})=a_{i,j}\pi_r$

(2) If (w,x)=(c,d) and (y,z)$\neq$(a,b), $f_r(a_{i,j})=2^{a-c}3^{b-d}a_{i,j}\pi_r$

(3) If (y,z)=(c,d) and (w,x)$\neq$(a,b), $f_r(a_{i,j})=5^{a-c}7^{b-d}a_{i,j}\pi_r$

(4) If (w,x)=(a,b) and (y,z)=(c,d), $f_r(a_{i,j})=5^{-c}7^{-d}a_{i,j}\pi_r$

(5) If (y,z)=(a,b) and (w,x)=(c,d), $f_r(a_{i,j})=2^{a-c}3^{b-d}5^{-a}7^{-b}a_{i,j}\pi_r$

After f_r has been applied to the picture, g_r, defined as follows, is then applied in backward zigzag sequence:

a) If p_{4r+1} divides $a_{i,j}$, then $g_r(a_{i,j})=a_{i,j}$

b) If p_{4r+1} does not divide $a_{i,j}$, but does divide one of the g_r's of its predecessor neighbors in the sense of the backward zigzag sequence, then proceed exactly as in part (c) of the definition of f_r.

Applying f_1, g_1, f_2, g_2,..., f_{mn}, g_{mn} to the picture insures that every redundancy has been eliminated, since there cannot be more redundancies than picture elements (in fact, there must be considerably fewer), and each (f_r,g_r) cycle eliminates at least one redundancy. We have thus shown that the desired task of tagging each connected component of the picture with a unique label is accomplished by the following sequence of local operations:

f in forward raster* sequence
f_1 in forward zigzag sequence
g_1 in backward zigzag sequence
:
:
f_{mn} in forward zigzag sequence
g_{mn} in backward zigzag sequence

Note that in practice the task will be completed long before the mn^{th} cycle, and the remaining cycles will not change the picture at all. However, there is no easy way of detecting the completion if only sequential local operations are permitted.

*or zigzag

3.3. Programming considerations

A practical computer program for determining connected components along the lines described above need not be as elaborate, since it need not restrict itself to local operations alone. Some immediate shortcuts, once other types of operations are permitted, include the following:

a. The "redundancies" can be stored as a table "outside" the picture; this greatly reduces storage requirements.*

b. Elimination of redundancies can be performed by processing this table. When this has been done, the redundant picture element labels can be "translated" into irredundant ones as the very last step, by making one scan of the picture, "looking up" each value in the processed table and substituting the equivalent irredundant value if different from the given value.

c. The table can be processed in many fewer steps than are required to process redundancies within the picture, since (1) the table is in general much smaller; (2) when a redundancy is being "reduced," it need not be "carried" through the table in the form of a set of high prime powers, since processing is not constrained to neighborhoods within the table; (3) it is easy to stop

*Since auxiliary tables which contain stored information about the picture are used in this and the following steps, the operations performed are no longer local; the table makes available information about picture elements which are not neighbors of the element being processed.

the processing when the table is exhausted, rather than blindly repeating it mn times.

d. The labels used can simply be consecutive integers, if a count is kept outside the picture so that no label is used twice; there is no need to use cumbersome prime power products indexed to the picture element coordinates to insure uniqueness.

In the light of these simplifications, a program for connected component determination can proceed as follows:

(1) Apply the function f to the picture in forward raster sequence, where

a) If $a_{i,j}=1$, $f(a_{i,j})=1$

b) If $a_{i,j}=0$, and all of $f(a_{i-1,j-1})$, $f(a_{i-1,j})$, $f(a_{i-1,j+1})$ and $f(a_{i,j-1})=1$, then $f(a_{i,j})=$ the first label value not yet used

c) If $a_{i,j}=0$, and all of these f's which $\neq 1$ are the same, $f(a_{i,j})=$that same label value

d) If $a_{i,j}=0$, and two of these f's $\neq 1$ and are different, $f(a_{i,j})=$the smaller of the two label values. In this case the pair of label values is stored in the first unused space in an auxiliary table, with the smaller value first.

(2) Process the table as follows:

a) Order the pairs lexicographically (in order of increasing first value, and for each of these, in order of increasing second value)

b) Let the first pair of values be A,B. Scan the table and replace every B by an A. Store the pair (A,B) in a second auxiliary table (which will be used to relabel the picture components in step (3)) and erase the first pair

c) Reorder the remaining pairs lexicographically and with smaller term first, erasing all pairs whose terms are equal

d) Repeat steps (b-c) until every pair has been processed and erased. The second table now contains a set of pairs of the form

$$(A_1,B_{11}),(A_1,B_{12}),\ldots,(A_1,B_{1,n_1}),(A_2,B_{21}),\ldots,$$
$$(A_2,B_{2,n_2}),\ldots,(A_h,B_{h_1}),\ldots,(a_h,B_{h,n_h})$$

which are ordered lexicographically. The A's in this table consist of one representative (the smallest) from each equivalence class of redundant values, and the B_{ij}'s (all distinct) are the remaining elements of the class represented by A_i.

(3) Scan the picture in any sequence, comparing each element's label value with the B's in the second table. Replace each B in the picture by the corresponding A. (If desired, gaps in the sequence of labels can be "closed up" before output.)

Note that this program consists basically of a single sequentially applied local operation (step (1), except that the redundancies are stored outside the picture to simplify the remaining steps), followed by sequential

processing of the table and sequential relabeling of the picture. The approach is still essentially sequential, even though for simplicity the restriction to "pure" neighborhood operations has been relaxed.

An IBM 7090/94 program along these lines has been written and tested. The program, originally written in the FAP symbolic assembly language, has been adapted so that it can be called as a FORTRAN subroutine. It accepts as input a digital picture on magnetic tape. Each record on the tape contains information about one row of the picture. A picture element can have any value from 0 to 2^6-1, and each row can consist of up to 2000 elements. The number of rows is limited only by the capacity of the tape.

The program selects any prespecified rectangular subpicture (rows r through r+u, columns s through s+v, for example). It "slices" the picture element values between any two prespecified levels t_1, t_2 ($0 \leq t_1 \leq t_2 < 2^6-1$), treating all values between t_1 and t_2 as "0" and all values outside the range as "1". The program then proceeds to label the connected components of the set of "0"s essentially as described above. The labels used for processing are simply the integers. For printout purposes, only the 46 distinct labels

ABCDEFGHIJKLMNOPQRSTUVWXYZ123456789+-/='.)$*,(

are used, in that order. If there are more than 46 connected components, these symbols are used over and over again, as many times as necessary. The output is a matrix of alphanumerics in which the symbol printed at each "0"

point is the label of the connected component which contains the point; "1" points are left blank. A labeled version of the picture is also written on tape for input to succeeding processing routines.

An example of a simple picture input to the program is shown as Figure 2 ("1"=black, "0"=white). The corresponding output for this picture is shown as Figure 3. In this picture, the component labeled I, for example, had two labels in the original processing. The redundancy of these labels was detected when the scan reached the "I" element which is circled on the Figure. Component D obviously had many labels originally, while components A and C had unique labels throughout the processing.

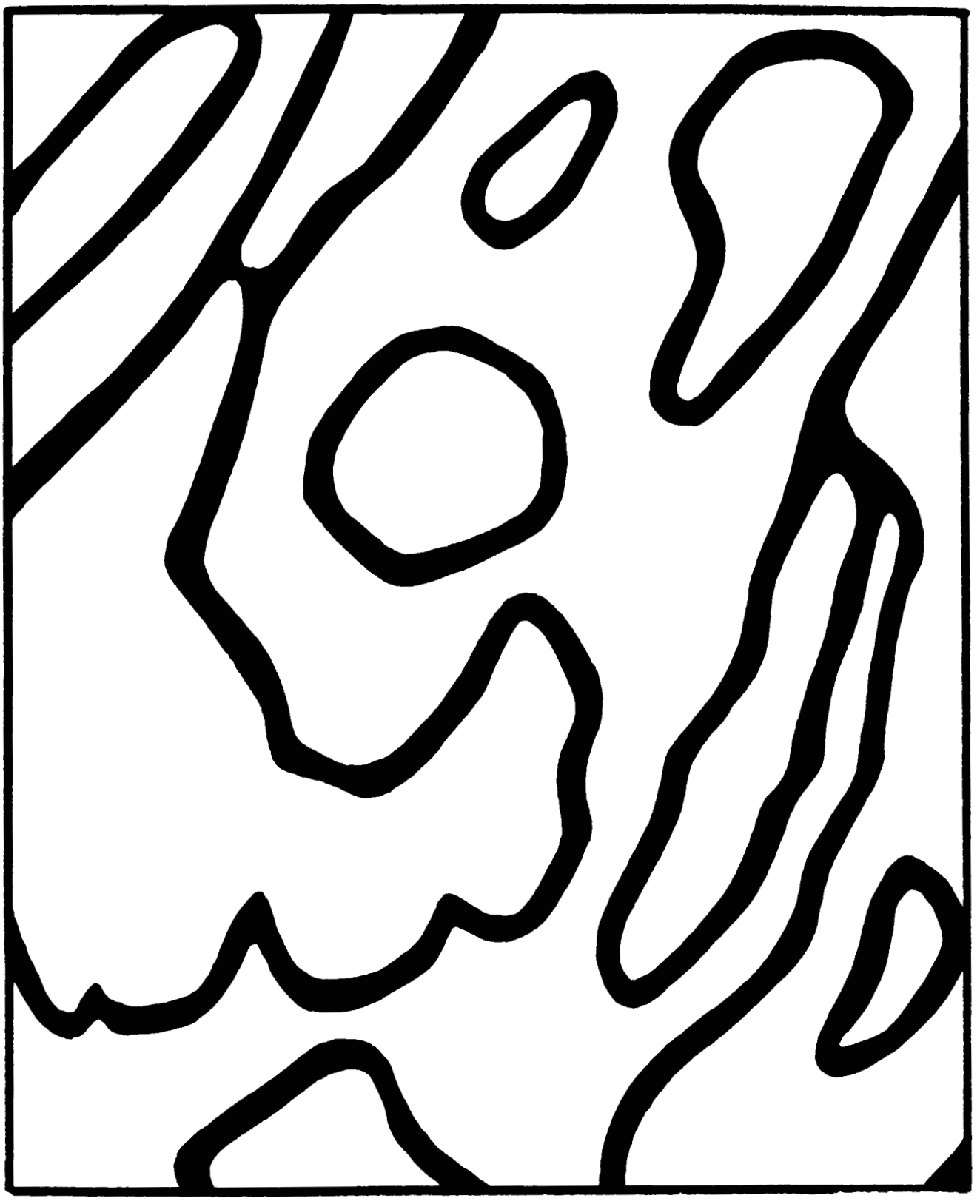


Figure 2. Picture Input to Processing Programs

Figure 3. Connected Component Transform of Figure 2

4. Sequential operations for "distance" determination

Several investigators over the past decade have considered picture transformations in which a given subset is "propagated" over the picture, or dually in which the subset is examined by an expanding array of sensors. (For the latter approach see Harmon [5] and Singer [6-8]; compare also Stevens [9].). In early work on digital picture processing (Kirsch [2]), a transformation of this type was performed by a sequence of local operations performed in parallel; this approach requires two operations for each incremental propagation step. More recent discussions of this type of transformation and its implications for shape description may be found in several papers by Blum ([10-12]; see also Kotelly [13]), who also considers the possible role of such transformations in visual form perception.

"Propagating" a subset over a picture is tantamount to finding the "distance", in the sense of the propagation process, between the subset and each point of the picture. In this section a simple "distance" concept, appropriate to digitized pictures, is introduced, and a transformation is defined which determines the distance from every picture element to a given subset.

4.1. Distance

Let P and Q be any two distinct points in a digitized picture, and let $d^*(P,Q)$ be the smallest positive integer such that there exists a sequence of distinct points $P=P_0, P_1, \ldots, P_n=Q$ with P_i a neighbor of P_{i-1}, $1 \leq i \leq n$. This d^* is called the distance from P to Q; if P=Q, the distance between them is defined as zero. The distance from P to a given subset S of the picture is defined as the smallest of the distances from P to the points in S.

Like connectivity, the distance concept is defined by iterating the property of being a neighbor. Here, however, the minimum number of iterations required to "reach" Q from P is of interest, whereas in the case of connectivity, the question considered was whether Q could be "reached" at all from P using only points in a given subset as intermediate points. As was pointed out for connectivity in Section 3.1, a "distance" can also be defined using only horizontal and vertical neighbors as "steps". If d(P,Q) is the distance from P to Q using this more restricted definition, it is clear that $d \geq d^*$. For simplicity, the restricted definition will be used in the remainder of this paper.

Evidently, d(P,Q) (and similarly for d^*) has all the properties of a metric*. It should be emphasized, however, that d is not even approximately the Euclidean distance. In fact, the locus of points at a given "distance" $d>0$ from a given point P is a diagonally oriented square of side d+1 centered at P, rather than a circle**.

*It is positive definite by definition, and is clearly symmetric (the reversal of a sequence from P to Q is a sequence from Q to P and vice versa). Moreover, since any two sequences from P to Q and Q to R, respectively, can be put end to end to give a sequence from P to R, it evidently satisfies the triangle inequality.

** For the metric d^*, the corresponding locus is a square of side 2d+1, centered at P and oriented horizontally and vertically.

4.2 The distance transformation

Given a digitized picture whose elements have only the values 0 and 1, it is desired to construct a "distance transform" of the picture in which each element has an integer value equal to its distance from the set of 0's. (It is assumed that the set of 0's is nonempty.) Thus in particular, the 0's remain unchanged, since they are at zero distance from themselves; the 1's which are horizontal or vertical neighbors of 0's also remain unchanged; the 1's which are horizontal or vertical neighbors of such 1's become 2's; and so on.

This transform can be performed using just two sequentially applied local operations as follows: Let

$$f_1(a_{i,j}) = \begin{cases} 0 \text{ if } a_{i,j} = 0 \\ \min(a_{i-1,j}+1,\ a_{i,j-1}+1) \text{ if } (i,j)\neq(1,1) \text{ and } a_{i,j}=1 \\ m+n \text{ if } (i,j)=(1,1) \text{ and } a_{1,1}=1 \end{cases}$$

$$f_2(a_{i,j}) = \min(a_{i,j},\ a_{i+1,j}+1,\ a_{i,j+1}+1)$$

Theorem. Let $C = (c_{i,j})$ be the picture which results when f_1 is applied to the picture $A = (a_{i,j})$ in forward raster sequence, followed by f_2 in backward raster sequence. Then C is the distance transform of A.

Proof: Note first that if $a_{i,j}\neq 1$ and a horizontal or vertical neighbor of $a_{i,j}$ is zero, evidently $c_{i,j}=1$, and conversely. Suppose now that $c_{i,j}$ is equal to the distance from the (i,j) element to the closest zero element in A for all (i,j) such that this distance $< k$. Let $B=(b_{i,j})$ be the picture which results from applying f_1 in forward raster sequence to A. If $c_{i,j}=k$, by the induction hypothesis the distance from the (i,j) element to the nearest zero must be at least k.

If it is greater than k, by definition of distance it must be at least k for each of the (i,j) element's horizontal and vertical neighbors. In particular, $c_{i+1,j}$ and $c_{i,j+1}$ each $\geq k$, so that $c_{i,j}=k$ implies $b_{i,j}=k$ by definition of f_2. But then $b_{i-1,j}$ or $b_{i,j-1}$, say the former, must be k-1 by definition of f_1, so that $c_{i-1,j} \leq k-1$, contradiction.

The distance transforms for a circle, two rectangles*, and regions F, J. and K of Figure 3 are shown as Figure 4. These transforms illustrate the output of an IBM 7090/94 program, written in FORTRAN, which accepts input digital picture data as described in Section 3.3. For simplicity, only the odd distance values are printed out modulo 10, while the even values are left blank; the points with value zero are printed as X's.

*Each of the rectangles is 1.1 by 1.8 inches, quantized 10 elements to the inch; the difference between their shapes in the Figure results from the unequal horizontal and vertical size of a character space on the printer. The circle appears distorted for the same reason.

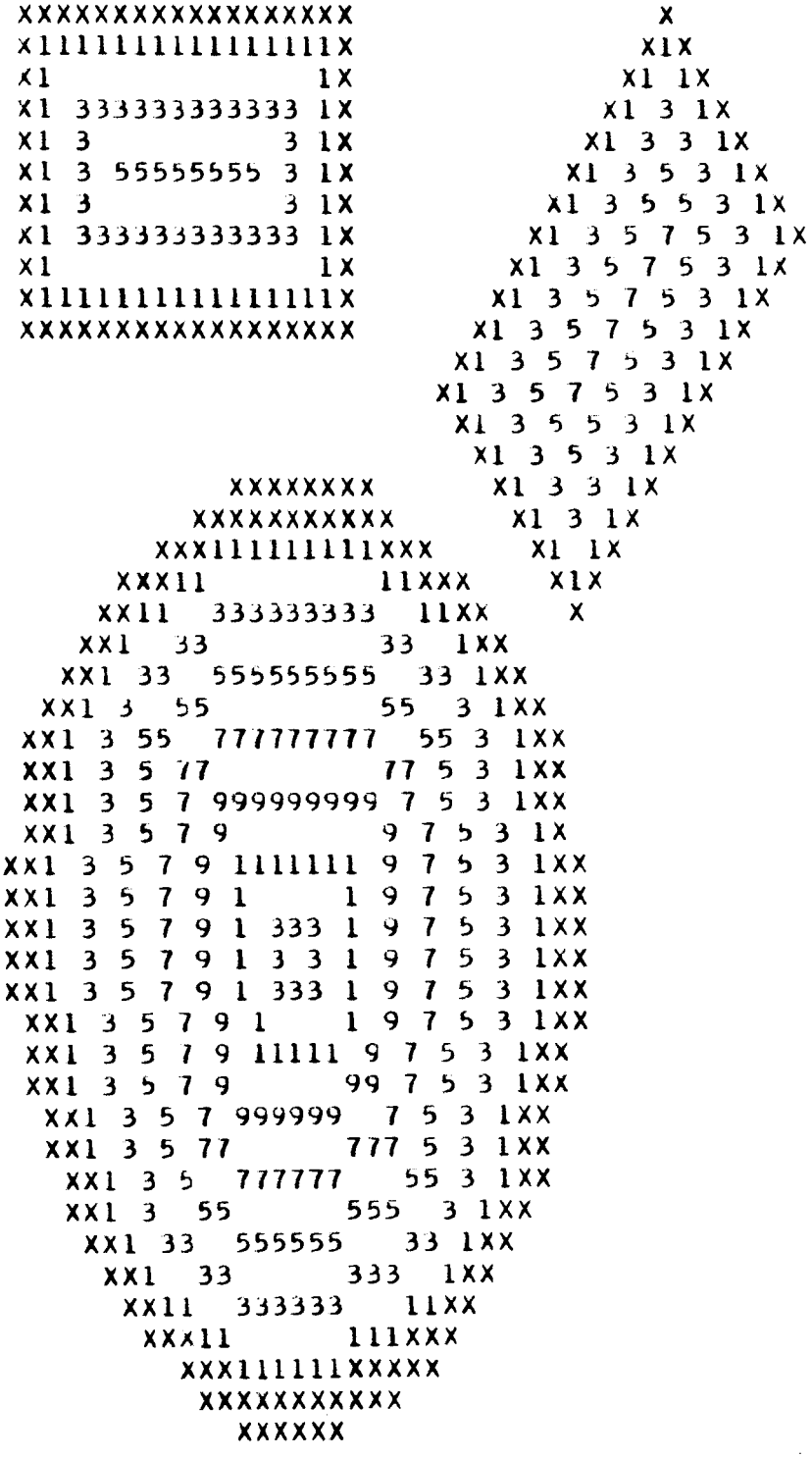


Figure 4. Sheet 1 - Distance Transforms of Two Rectangles and a Circle

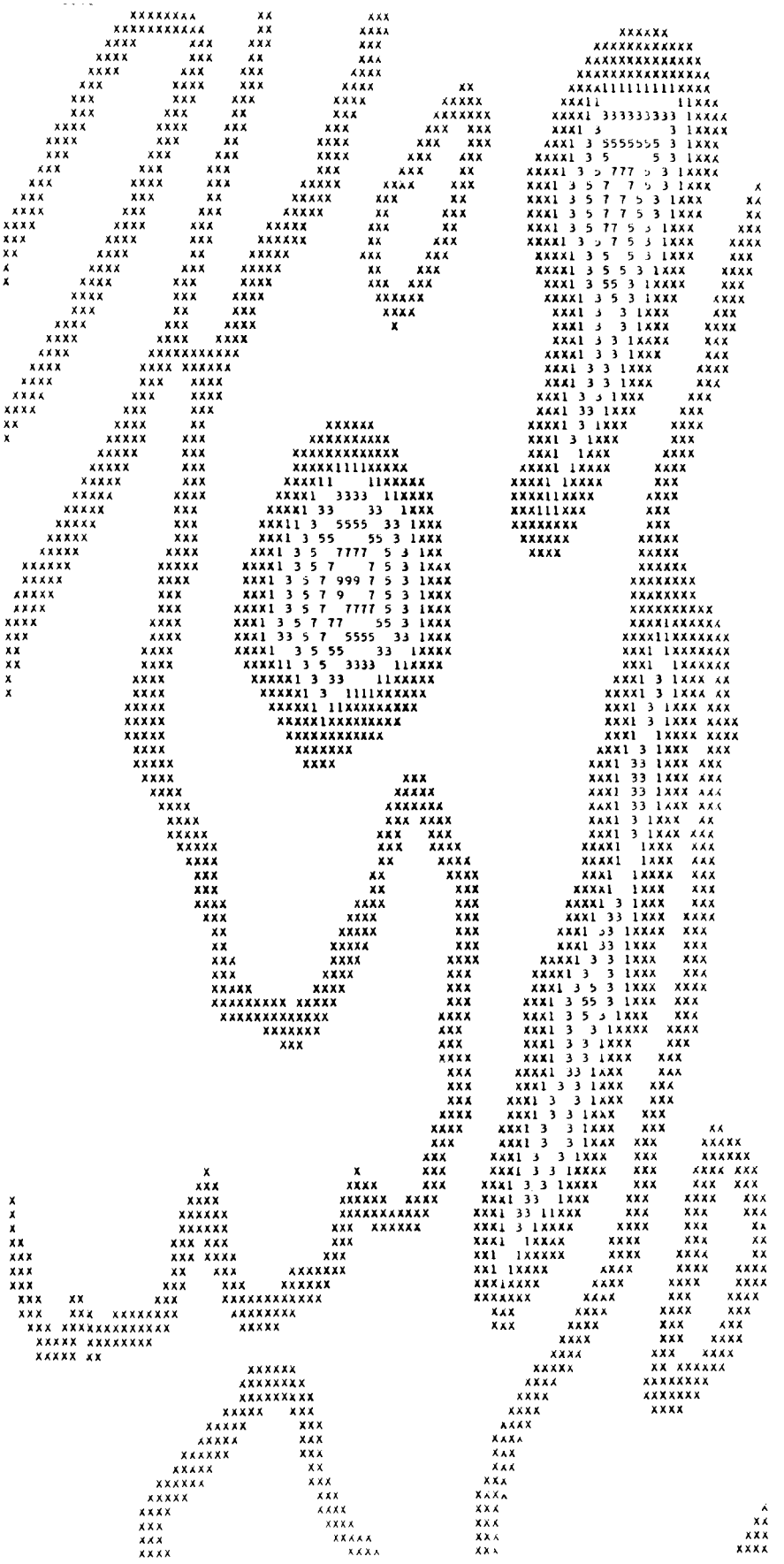


Figure 4. Sheet 2 - Distance Transforms of Components F,J,K of Figure 3

4.2. The distance "skeleton"

Blum has suggested in [12] that the locus of points at which the propagation wave front "intersects itself" may be perceptually important. This locus defines a sort of "skeleton" (Blum: "medial axis") for the original picture. In this subsection, a "skeleton" subset will be defined for the distance transform introduced above*, and it will be shown that this skeleton is the smallest subset of the transform picture from which the entire transform picture can be reconstructed by "reversing" the distance-measuring process.

Define the local operations g_1 and g_2 by

$$g_1(a_{i,j})=\max(a_{i,j},\ a_{i,j-1}-1,\ a_{i-1,j}-1)$$

$$g_2(a_{i,j})=\max(a_{i,j},\ a_{i,j+1}-1,\ a_{i+1,j}-1)$$

Let $G_1(P)$ be the picture which results when g_1 is applied to P in forward raster sequence; $G_2(P)$, the result of applying g_2 to P in backward raster sequence; and $G(P)=G_2(G_1(P))$.

Lemma 1. If A is any picture and $G(A)=(c_{i,j})$, then all of $|c_{i,j}-c_{i+1,j}|$, $|c_{i,j}-c_{i,j+1}|$, $|c_{i,j}-c_{i-1,j}|$ and $|c_{i,j}-c_{i,j-1}|$ which are defined are ≤ 1.

Proof: Let $A=(a_{ij})$, $G_1(A)=(b_{ij})$. By definition of g_1 we have

$b_{i,j}\geq b_{i,j-1}-1$ and $b_{i,j}\geq b_{i-1,j}-1$ for all i,j

so that $b_{i,j+1}\geq b_{i,j}-1$ and $b_{i+1,j}\geq b_{i,j}-1$ for all i,j

*It should be emphasized that since the distance considered here is non-Euclidean, as already pointed out, the resulting "skeleton" is not likely to have any special significance for visual form perception; however, it is still a useful picture processing tool.

Similarly, by definition of g_2 we have for any $B=(b_{ij})$ and for all i,j

$$c_{i,j} \geqq c_{i,j+1}-1 \qquad c_{i,j} \geqq c_{i+1,j}-1$$

$$c_{i,j-1} \geqq c_{i,j}-1 \qquad c_{i,j},j \geqq c_{1,j}-1$$

where $G_2(B)=(c_{ij})$.

If we can show that the above relations on the b's remain true when the b's are replaced by c's (that is, when g_2 is applied), the assertion made in the Lemma will follow immediately, since (e.g.) $c_{i,j-1} \geqq c_{i,j}-1$ and $c_{i,j} \geqq c_{i,j-1}-1$ are equivalent to $|c_{i,j}-c_{i,j-1}| \leqq 1$.

Suppose that these relations hold for all the c's through the $(i,j)^{th}$ in the sense of the backward raster sequence. Since g_2 can never decrease the value of a picture element, we have $b_{i,j-1} \leqq b_{i,j}+1 \leqq c_{i,j}+1$. By the induction hypothesis, $c_{i+1,j-1}-1 \leqq c_{i+1,j}$, and by the relations on the c's, this $\leqq c_{i,j}+1$. Hence

$$c_{i,j-1}=\max(b_{i,j-1},\ c_{i,j}-1,\ c_{i+1,j-1}-1) \leqq c_{i,j}+1,$$

proving the induction step. Finally,

$$c_{m,n-1}=\max(b_{m,n-1},\ c_{m,n}-1) \leqq \max(b_{m,n}+1,\ c_{m,n}+1)=c_{m,n}+1$$

and similarly $c_{m-1,n} \leqq c_{m,n}+1$, completing the proof.

Lemma 2. Let A be a picture such that $a_{i,j} \geqq a_{i,j-1}-1$ and $a_{i,j} \geqq a_{i-1,j}-1$ for all i,j; then $G_1(A)=A$. Similarly, if A is such that $a_{i,j} \geqq a_{i,j+1}-1$ and $a_{i,j} \geqq a_{i+1,j}-1$ for all i,j, then $G_2(A)=A$.

Proof: Clearly $g_1(a_{11})=a_{11}$. If $G_1(A)=A$ for all elements up to the i,jth (in the sense of the forward raster sequence), then $g_1(a_{i,j})=\max(a_{i,j}, g_1(a_{i,j-1})-1, g_1(a_{i-1,j})-1)=\max(a_{i,j}, a_{i,j-1}-1, a_{i-1,j}-1)$ by induction hypothesis, and this$=a_{i,j}$ by the original assumption about A. The proof of the second part is exactly analogous.

Corollary. $G_1(G_1(A))=G_1(A)$, $G_2(G_2(A))=G_2(A)$, and $G(G(A))=G(A)$ for all A

Proof: By the proof of Lemma 1, $G_1(A)$ has the properties of the first part of Lemma 2, so that $G_1(G_1(A))=G_1(A)$, and similarly for $G_2(A)$. By Lemma 1, G(A) has the properties of both parts of Lemma 2; hence by Lemma 2 $G(G(A))=G_2(G_1(G(A))=G_2(G(A))=G(A)$.

Lemma 3. The distance transform of any picture has the property of Lemma 1.

Proof: By the Theorem of Section 4.1, in such a picture each element value is equal to the distance from the element to a zero-valued element, and clearly these distances for an element and any of its neighbors can differ by at most 1.

Corollary. If T is any distance transform picture, G(T)=T.

Proof: As for the Corollary to Lemma 2.

Lemma 4. Let $A=(a_{ij})$, $B=(b_{ij})$ be pictures such that $a_{i,j}\leq b_{i,j}$ for all i,j. Let $G(A)=(c_{ij})$, $G(B)=(d_{ij})$, and let $a_{h,k}=b_{h,k}=d_{h,k}$ for some h,k. Then $a_{h,k}=c_{h,k}$.

Proof: Evidently we must have $c_{i,j}\leq d_{i,j}$ for all i,j, so that $c_{h,k}\leq d_{h,k}=a_{h,k}$. But G never decreases the value of a picture element; hence $a_{h,k}\leq c_{h,k}$.

If $P=(p_{ij})$ is any picture, $P'=(p'_{ij})$ will be called a <u>partial picture</u> of P if $t'_{i,j}=t_{i,j}$ or 0 for all i,j.

<u>Corollary</u>. Let T be any distance transform picture, T' any partial picture of T. Then all the elements of T' which are equal to the corresponding elements of T are invariant under G.

Proof: Take A=T', B=T in Lemma 4. By Lemma 3, $b_{i,j}=d_{i,j}$ for all i,j; hence for all $a_{h,k}$ such that $a_{h,k}=b_{h,k}$, we have $a_{h,k}=c_{h,k}$, as required.

<u>Lemma 5</u>. Let $A=(a_{ij})$ be a picture, $G(A)=(c_{ij})$, and let $a_{h,k}<V$; $c_{h-1,k}$, $c_{h,k-1}$, $c_{h+1,k}$, $c_{h,k+1}$ all $\leq V$. Then $c_{h,k}<V$.

Proof: Let $G_1(A)=(b_{ij})$. Since G_2 never decreases the value of an element, we have $b_{h-1,k}\leq c_{h-1,k}\leq V$, $b_{h,k-1}\leq c_{h,k-1}\leq V$, so that $b_{h,k}=\max(a_{h,k}, b_{h-1,k}-1, b_{h,k-1}-1)<V$, and $c_{h,k}=\max(b_{h,k}, b_{h+1,k}-1, b_{h,k+1}-1)<V$.

If $T=(t_{ij})$ is a distance transform picture, the partial picture $T^*=(t^*_{ij})$ defined by $t^*_{i,j}=t_{i,j}$, if none of $t_{i-1,j}$, $t_{i+1,j}$, $t_{i,j-1}$, $t_{i,j+1}$ is $t_{i,j}+1$; 0, otherwise

will be called the <u>skeleton</u> of T. We assume that T is not the trivial picture every element of which has value m+n.

<u>Theorem</u>. $G(T^*)=T$, and if T' is any partial picture of T such that G(T')=T, then T^* is a partial picture of T'. In other words, T^* is the partial picture of T with fewest nonzero elements such that $G(T^*)=T$.

Proof: If T is any picture with integer-valued elements, let T_k be the set of elements of T which have value k. Let N be the highest value of any element of T; then by definition, $T^*_N=T_N$, so that by the Corollary to Lemma 4, $G(T^*)_N=T_N$. Suppose that $G(T^*)_{M+1}=T_{M+1}$. By definition, T* contains every element of value M which has no element of value M+1 as a neighbor in T (or equivalently, in G(T*)), and by the Corollary to Lemma 4, G(T*) still contains these elements. On the other hand, if $t_{h,k}=M$ and has a neighbor in G(T*) with value M+1, then by definition of G, the (h,k) element in G(G(T*)) has value at least M. But G(G(T*))=G(T*) (Corollary to Lemma 2), and by the proof of Lemma 4, the (h,k) element in G(T*) can have value at most that of the (h,k) element in G(T)=T (Corollary to Lemma 3); hence every such element has value M in G(T*), proving that $G(T^*)_M=T_M$. This induction argument proves $G(T^*)_k=T_k$ for all k(=N,N-1,...,1,0), so that G(T*)=T. Conversely, let T' be any partial picture of T such that G(T')=T, and let $t_{h,k}$ be an element of T of value M>0 which has no neighbor in T of value M+1 and which fails to be in T'. Then the values of its neighbors in G(T')=T are $\leq$M (Lemma 1), while its value in T' is 0 <M, so that by Lemma 5 its value in G(T') is still <M, contradicting G(T')=T. Thus T' must contain every element of T* of value >0, completing the proof.

Skeletons for the pictures of Figure 4 are shown as Figure 5. In this Figure, the nonzero skeleton point values are printed out modulo 10. As Figure 5 b-c show, the skeleton is not invariant under rotation; note also that the Euclidean "skeleton" for a circle would evidently be just the point at its center, unlike the skeleton shown in Figure 5a.

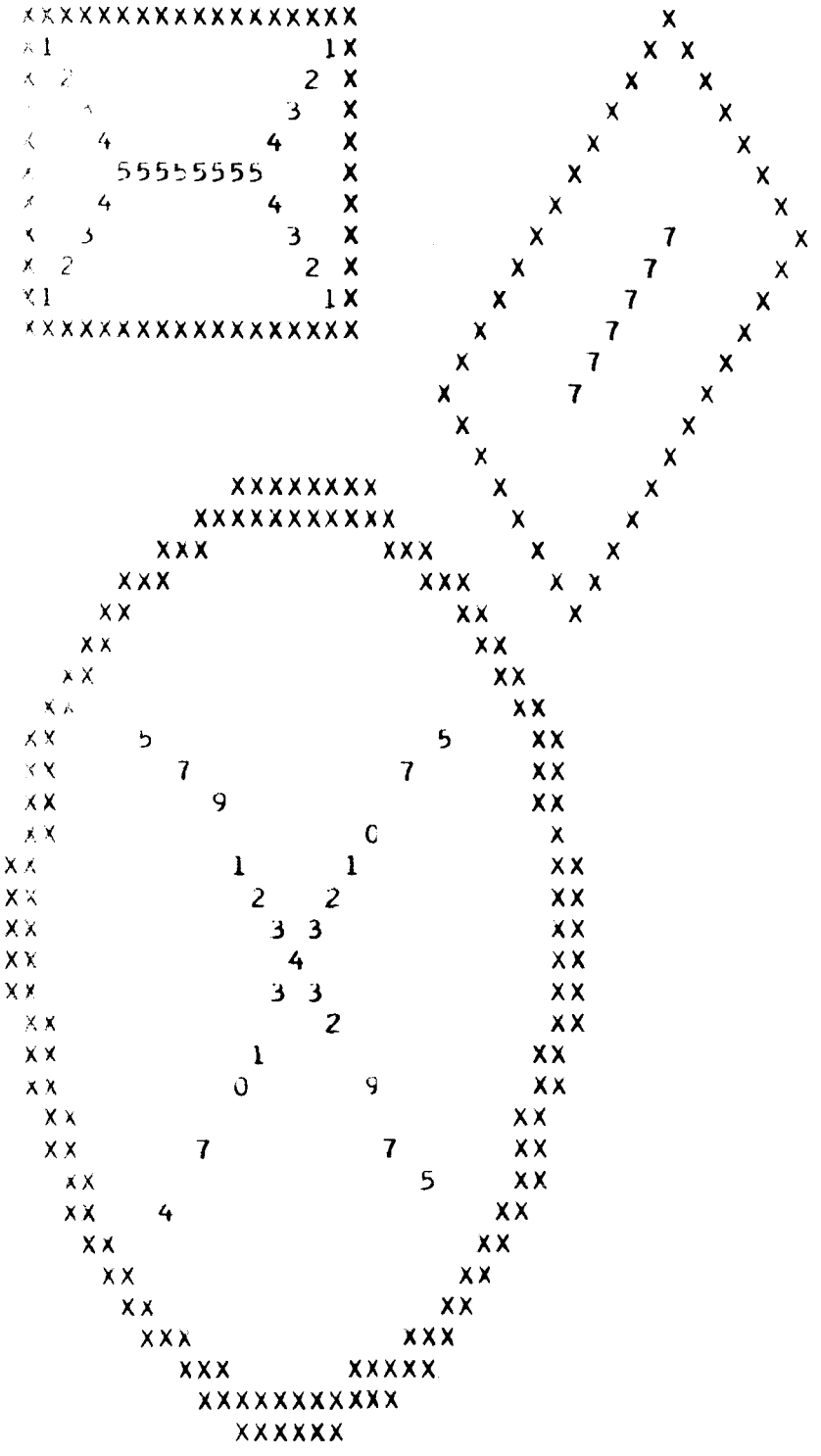


Figure 5. Sheet 1
Skeletons for Figure 4

Figure 5. Sheet 2 - Skeletons for Figure 4

5. Applications: Connectivity and proximity

The connectivity transformation described in Section 3 has several immediate applications. Once a label has been assigned to each connected component of a picture, it is trivial to count the number of such components by simply counting the number of labels which were used. (A special-purpose version of the connected component program can be written which only counts the components but does not label each element of each component; see Nuttall [14] and Sabbagh [15] pp 43-48. It is also possible to perform this "blob counting" operation by a process of successively deleting from each component border elements which do not disconnect the component until only one element per component remains;* for this approach, which is easily implemented using parallel local operations, see Kirsch [2], Minot [16] and Izzo [17-18]. A closely related approach, using accretion rather than deletion, has been implemented by von Foerster and his colleagues [19-21]. One can also measure the area of any given component by counting the number of times its label occurs.

*Provided that the components are simply connected; if they have "holes" in them, a more complicated procedure is necessary.

5.1. Adjacency and order of connectivity

Two somewhat less trivial problems which can be solved with the aid of the basic connectivity transformation are

(a) Constructing the graph corresponding to a given dissection of a picture into connected components.

(b) Determining the order of connectivity of a given connected component.

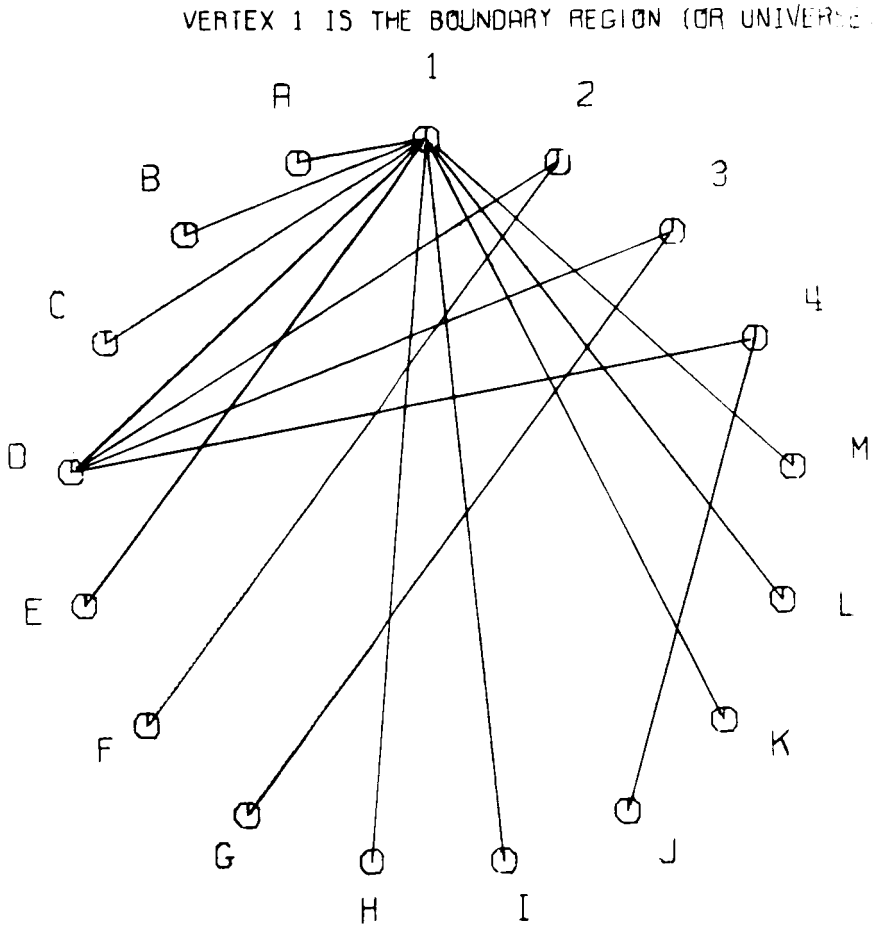
In the graph of any dissection of the picture, each connected component is represented by a single node or vertex; for simplicity, these may be taken to be the vertices of a regular polygon. Two nodes are joined by an arc (a side or diagonal of the polygon) if and only if the corresponding components are adjacent. For many purposes it is convenient to represent the exterior of the picture by a single additional node, which is considered to be adjacent to every component having elements on the boundary of the picture. A graph with k nodes is completely specified by its incidence matrix; this is a k-1 by k-1 triangular matrix (a_{ij}), $i<j$, in which $a_{ij}=1$ if the i^{th} and j^{th} nodes are joined by an arc; =0 otherwise.

In Figure 3, the connected components labeled A,...,M, together with the connected components of the set of blank points, constitute a dissection of the picture. Since the border of Figure 2 was black, the additional "exterior" component is connected to all of these blank points except those in the belts immediately surrounding regions F, G, and J. The set of blank points thus has four components, namely the one of which the picture border is a part and the ones surrounding F, G, J. The nonzero entries in the incidence matrix of the graph of Figure 3 are shown below:

Region	Border of F	Border of G	Border of J	A	B	C	D	E	F	G	H	I	J	K	L	M
Picture border	0	0	0	1	1	1	1	1	0	0	1	1	0	1	1	1
Border of F		0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
Border of G			0	0	0	0	1	0	0	1	0	0	0	0	0	0
Border of J				0	0	0	1	0	0	0	0	0	1	0	0	0

A computer program for constructing the graph of a given dissection is easily written, once the connected components of the dissection have been labeled. It suffices, for example, to examine every two by two subpicture and to enter 1's in the appropriate positions in an incidence matrix whenever such a subpicture contains points which have two or more different labels. The graph of Figure 3, constructed in this way by a FORTRAN program and output by a tape-controlled plotter, is shown as the first part of Figure 6.

Once the graph has been constructed, it is easy to solve the second problem posed earlier. It is easily seen that the order of connectivity of a connected component is equal to the number of nonadjacent "pieces" into which the picture is divided if the given component is deleted. Evidently, this is just the number of connected components into which the graph is divided if the corresponding node and the arcs emanating from it are deleted. This number can be determined by examining the



REGION			ORDER OF CONNECTIVITY
1	BOUNDARY OF PICTURE		10
2	BOUNDARY OF REGION	F	2
3	BOUNDARY OF REGION	G	2
4	BOUNDARY OF REGION	J	2
D			4

Figure 6. Graph and Multiple Connectivity Table for Figure 3.

incidence matrix of the graph after deleting the corresponding row and column*. A FORTRAN program which performs this analysis has been written; its output for Figure 3 is shown in the second part of Figure 6.

*This number could be determined directly from the picture as follows: Temporarily label the points of the given component 1, the points of its complement 0, whether these points were 1 or 0 in the original picture. Apply the connected component labelling program to this new set of 0's and simply count the number of its components. However, it is much simpler and faster to determine the orders of connectivity from the graph, once this has been constructed.

5.2. Proximity

Closely related to the two problems just discussed is the question of defining a "graph" for a picture such as Figure 3 ignoring the unlabeled "borders", and considering two of the labelled regions as being "adjacent" if they are separated only by a border. Intuitively, region I on Figure 3 is adjacent in this sense to regions B, C and D, but region B is not adjacent to region D. The graph for the labelled regions and "exterior region" of Figure 3 corresponding to this notion of adjacency is shown as Figure 7.

The concept of adjacency in the intuitive definition just given requires careful consideration. If the borders between regions in a picture all have approximately the same thickness, the adjacency of two regions in this sense essentially reflects their degree of proximity; regions are adjacent provided they approach one another within a distance just greater than the border thickness.* As in the case of true adjacency, this can be determined by examining all possible subpictures of the appropriate size and entering 1's in the incidence matrix whenever points with two or more labels are contained within such a subpicture. The graph in Figure 7 was drawn by a FORTRAN program which analyzed 4 by 4 subpictures of Figure 3 in this manner.

A more difficult problem is presented if the borders between regions are of varying thickness. Even in this case, however, one might proceed by determining the degree and direction of the elongation of any segment of border (see Section 6.2.).

*One should certainly not define two regions as being "adjacent" if it is merely possible to go from one to the other by moving through border elements only; by this criterion, regions A and B, B and D on Figure 3 would be "adjacent", and K would be "adjacent" to the picture exterior.

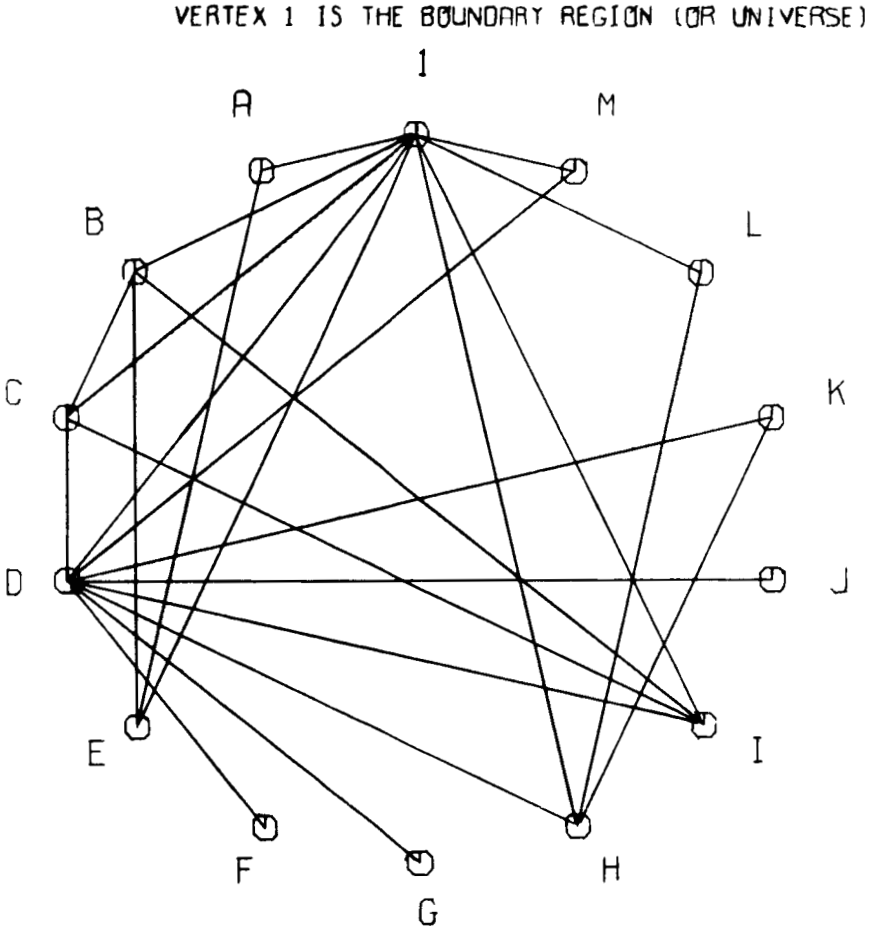


Figure 7. Proximity Graph for Figure 3

"Adjacent" could then be defined as "separated by a segment of border which is tangentially elongated". This definition would be consistent with the intuitive concept of adjacency for the labelled regions of Figure 3.

It should be pointed out that the intuitive definitions of adjacency suggested in the last three paragraphs are of more limited usefulness than the mathematical definition. For example, one cannot in general determine the order of connectivity of a region from a graph based on such a definition. Examination of the graph in Figure 7 does indeed show, in agreement with intuition, that all the regions except D are simply connected, while D has order of connectivity 4. However, suppose that in Figure 3 the six D's to the right of the top row of F's are replaced by blanks, making a "cut" in region D. This does not change the graph of the Figure (region D is still connected, and region F still not adjacent, in the intuitive sense, to the picture exterior); but region D now has order of connectivity only 3*.

*Note that making this "cut" does change the graph in Figure 6, since it combines the "border of F" region with the "picture border" region. The modified graph does in fact reflect the orders of connectivity which result from making the cut.

6. Applications: Shape (Elongation)

The distance transform can be used to obtain a variety of information about the shapes of regions on a picture. In this section, two applications of this transform to the definition of elongation are discussed. The "elongation" considered here is an intrinsic shape property; a snake is considered to be elongated even when it is coiled. It is difficult to define this property in conventional geometrical terms, in spite of its evident intuitive significance.

Since the connected component transformation provides the ability to single out any component of a picture for analysis of its shape, it will be assumed below that the given picture contains only a single connected region ("figure") consisting of 1's, and that the remainder of the picture consists of 0's.

6.1. Elongation as the proportion of a figure which lies close to its boundary

In his original papers [10-11] on the propagation concept, Blum suggested that the successive wavefronts--that is, the sets of points which are at a given distance from the original figure boundary--could provide useful information about the shape of the figure. For example, if the figure is a square, the wavefronts are concentric squares, and the numbers of points in them decrease linearly to zero. (See the solid curve in Figure 8a). On the other hand, if the figure is a very elongated rectangle (Figure 8b), the number of points in a wavefront decreases linearly until the center line of the rectangle is reached, when it drops abruptly to zero. Analogous plots of number of points vs. number of steps for three irregular figures of approximately equal area (regions F, J and K of Figure 3) are shown in Figure 8c-e.

As the solid curves in Figure 8, especially parts (a-b), indicate, the manner in which wavefront "perimeter" decreases provides a measure of the elongation of a given figure. This measure represents the degree to which the interior of the figure lies close to its boundary, which is intuitively related to the intrinsic elongation of the figure.

A shape descriptor closely allied to wavefront perimeter is the number of different squares of a given size which are contained within a given figure and touch its boundary. For simplicity, only squares whose sides are parallel to the sides of the picture will be considered. These numbers can be computed by systematically erecting all possible squares on the cross-sections of the figure by the rows of the picture.

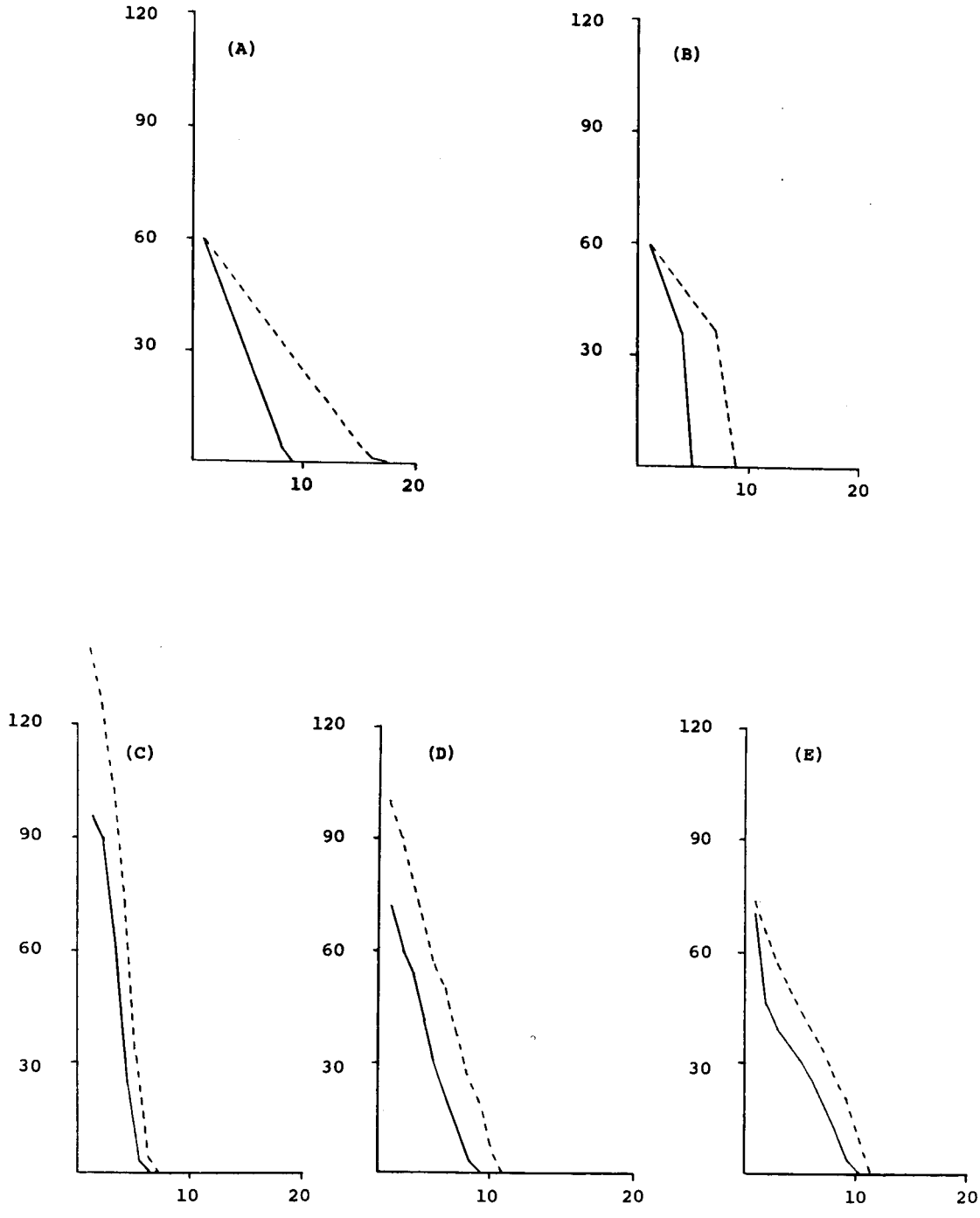


Figure 8. Wavefront Perimeter and Boundary-touching Square Plots for (A) a square; (B) an elongated rectangle; (C) component K of Figure 3 (elongated); (D) component F of Figure 3 (partly elongated); (E) component J of Figure 3 (roughly circular)

An IBM 7090/94 program for doing this has been written in FAP. In Figure 8, the numbers of boundary-touching squares are plotted as dashed curves for comparison with the wavefront perimeters.*

The measure of elongation provided by these shape descriptors is relatively crude, since they are computed over the entire figure. A much more sensitive measure of elongation will now be defined, using the distance skeleton concept introduced in Section 4.3.

*The wavefront and enclosed squares descriptors are conceptually related, but not equivalent, to the "Buffon needle" shape descriptor proposed by Tenery [22-23], which involves the probability that a line segment of given length randomly dropped on a figure with one end inside the figure also has the other end inside.

6.2 Elongated parts of a figure

The skeleton subset introduced in Section 4.3 can be used to define a variety of useful shape properties. In this section it is applied, in combination with proximity analysis (see Section 5.2.), to the problem of determining elongated parts of a given figure. The approach described below is based on the intuitively appealing idea that any elongated figure part should give rise to a skeleton subset similar to that of an elongated rectangle. Such a skeleton should contain a large number of adjacent or proximate points located a relatively short distance away from the figure border. If the "reverse" distance transformation (Section 4.3.) is applied to this set of points, the elongated part of the original figure should be regenerated.

Consider as an example the fictitious "hydrography" of Figure 9(a), in which the X's are water and the blanks land. Intuitively, the river, tributaries and creeks are elongated, but the lake and bay are not. Figure 9(b) shows the corresponding skeleton locus modulo 10, where the land points have been treated as 0's, the water points as 1's. In this figure the components of the points with values 5 or less, defined by a proximity criterion using a 3 by 3 subpicture, have been circled*. The large components (25 elements or more) evidently correspond to elongated portions of the hydrography, the one at the lower right to the elongated loop of lake around the island. In Figure 9(c) these elongated pieces have been regenerated by applying the reverse distance transform starting with these large components only. The regenerated parts are represented by E's, the remaining original water points by dots.

*The number 5 is an arbitrary threshold, not necessarily optimum. Skeleton points contained in the same 3 x 3 subpicture were considered to belong to the same component only if their values differed by two or less.

Figure 9A. Lake, River and Streams

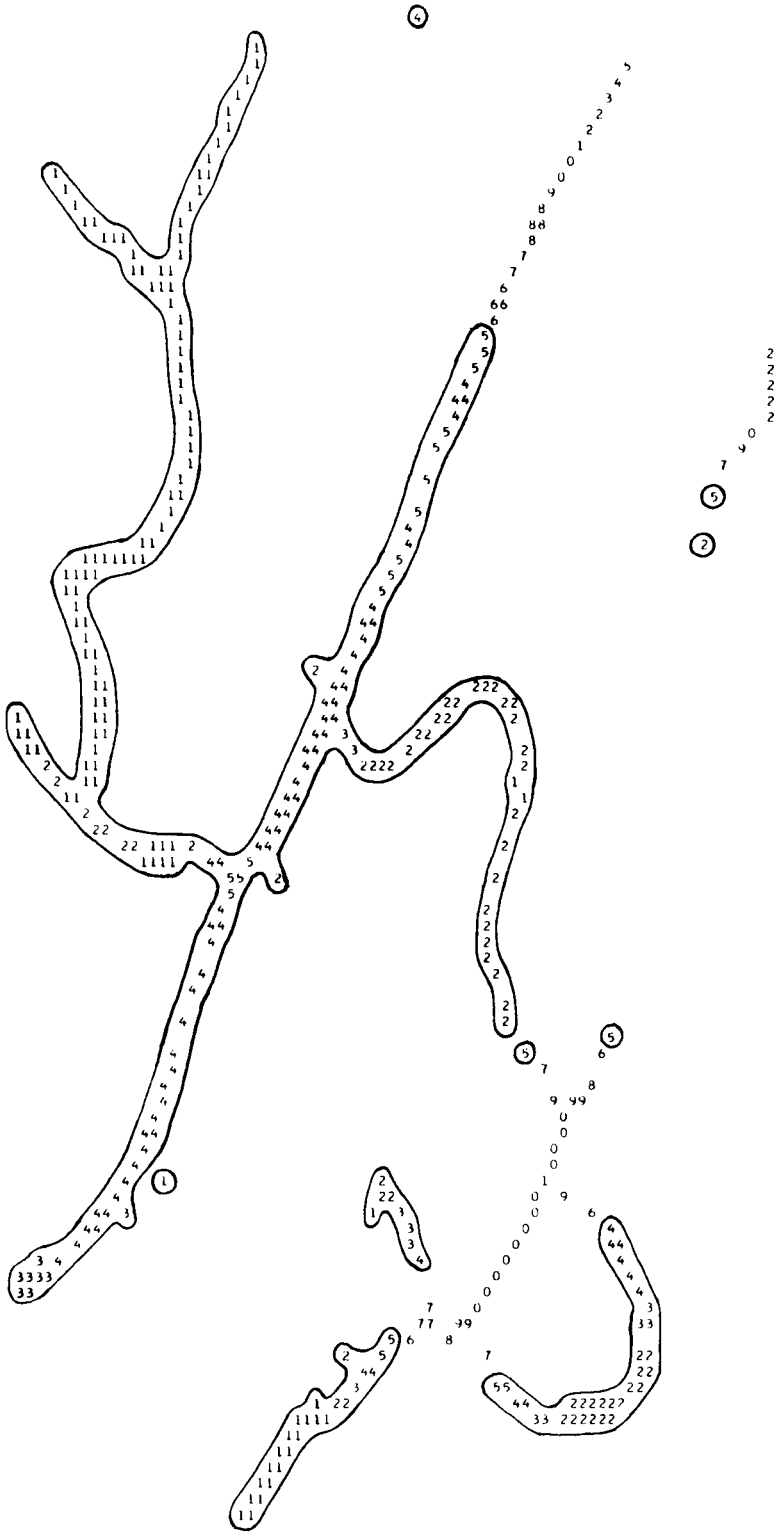


Figure 9B. Skeleton of Figure 9A

Figure 9C. "Elongated Parts" of Figure 9A

This example suggests the following general procedure for determining elongated parts of a figure:

(1) Apply the distance transform to the figure and determine the skeleton locus.

(2) For each k=2,3,..., up to half the picture diameter if necessary, consider the set S_k of skeleton points which have values $\leq k$.

(3) Determine "proximity components" of each S_k, and count the number of points in each component.

(4) Select those components, if any, which have more than t_k points. A reasonable value for t_k is in the range 5k-10k*, corresponding to a set of proximate skeleton points whose "length" is at least 2½ times the "width" of the piece of figure which gave rise to it.

(5) Apply the reverse distance transform to these components to reconstruct the elongated parts of the original figure.

*This uncertainty can be reduced if the skeleton locus is "thinned" before performing this step.

References

1. Dinneen, G. P. Programming pattern recognition, Proceedings, 1955 Western Joint Computer Conference, 94-100.

2. Kirsch, R. A. et al. Experiments in processing pictorial information with a digital computer, Proceedings, 1957 Eastern Joint Computer Conference, 221-229.

3. Unger, S. H. A computer oriented toward spatial problems, Proc. I.R.E. 46, (1958), 1744-1750.

4. Narasimhan, R. Labeling schemata and syntactic descriptions of pictures, Info. Control. 7, (1964), 151-179.

5. Harmon, L. D. Line drawing pattern recognizer, Electronics 33, (September 2, 1960), 39-43; Proceedings, 1960 Western Joint Computer Conference, 351-364.

6. Singer, J. R. Model for a size invariant pattern recognition system, Bionics Symposium, WADD Technical Report 60-600, (December, 1960), 239-245.

7. ---------- . Electronic analog of the human recognition system, J. Opt. Soc. Amer. 51, (January, 1961), 61-69.

8. ---------- . A self organizing recognition system, Proceedings, 1961 Western Joint Computer Conference, 545-554.

9. Stevens, M. E. Abstract shape recognition by machine, Proceedings, 1961 Eastern Joint Computer Conference, 332-351.

10. Blum, H. An associative machine for dealing with the visual field and some of its biological implications, Biological Prototypes and Synthetic Systems, Plenum, New York, 1962, 244-260.

11. -------. A machine for performing visual recognition by use of antenna-propagation concepts, Proc. I.R.E. Wescon Pt. IV, 1962.

12. -------. A transformation for extracting new descriptors of shape, Symp. on Models for the Perception of Speech and Visual Form, (November, 1964).

13. Kotelly, J. A mathematical model of Blum's theory of pattern recognition, (April, 1963), AD 412437.

14. Nuttall, T. C. Apparatus for counting objects, U. S. Patent 2803406, (August 20, 1957).

15. Sabbagh, E. N. Aerial Reconnaissance Electronic Reader, (June, 1960), AD 248200.

16. Minot, O. N. Counting and outlining of "two-dimensional" patterns by digital computer, U. S. Naval Elect. Lab. TM-414, (August, 1960).

17. Izzo, N. F. and Coles, W. Blood-cell scanner identifies rare cells, Electronics 35, (April 27, 1962), 52-57.

18. Izzo, N. F. Electro-optical enhancement of microscopic images, Proceedings, S.P.I.E. Image Enhancement Seminar, (March, 1963), Society of Photographic Instrumentation Engineers, Redondo Beach, Calif., 1-13.

19. von Foerster, H. Circuity of clues to platonic ideation, Aspects of the Theory of Artificial Intelligence, Plenum, New York, 1962, 43-81.

20. Babcock, M. L. Some physiology of automata, Proceedings, 1961 Western Joint Computer Conference, 291-298.

21. Weston, P. Photocell field counts random objects, Electronics 34, (September 22, 1961), 46-47.

22. Tenery, G. A pattern recognition function of integral geometry, I.E.E.E. Trans. on Mil. Elect. MIL-7, (April-July, 1963), 196-199.

23. ---------. Information flow in a bionics image recognition system, Symp. on Models for the Perception of Speech and Visual Form, (November, 1964).

Appendix

Proof of the equivalence of parallel and sequential local operations

We must show that any local operation applied in parallel is equivalent to a series of local operations applied sequentially, and vice versa. But indeed, let f be any local operation which when applied in parallel takes $a_{i,j}$ into $a^*_{i,j}$, and let the values of the $a_{i,j}$'s and $a^*_{i,j}$'s range between 0 and 2^r-1. Now define the local operations g and h as follows: g takes a_{ij} into

$$a'_{i,j}=2^r a_{i,j}+f([2^{-r}a'_{i-1,j-1}],[2^{-r}a'_{i-1,j}],[2^{-r}a'_{i-1,j+1}],$$
$$[2^{-r}a'_{i,j-1}],a_{i,j},a_{i,j+1},a_{i+1,j-1},a_{i+1,j},a_{i+1,j+1})$$

where the bracket denotes the integer part; and h takes $a'_{i,j}$ into $a''_{i,j}=a'_{i,j}-2^r[2^{-r}a'_{i,j}]$. Evidently $[2^{-r}a'_{i,j}]$ is just $a_{i,j}$, so that $a'_{i,j}=2^r a_{i,j} + a^*_{i,j}$; and then $a''_{i,j}=a'_{i,j}-2^r a_{i,j}= a^*_{i,j}$. Hence if g is applied to the picture sequentially, and then h is applied sequentially,* the result is exactly the same as if f is applied in parallel.

Conversely, let f be any local operation which when applied sequentially takes $a_{i,j}$ into $a^*_{i,j}$. To construct a sequence of local operations which produce the same result as

---

*h of course involves only the (i,j) point and not any of its neighbors, so that applying it to the picture "sequentially" and "in parallel" are no different. This proof and that of the converse use the device of "shifting" the element values, by multiplying them by 2^r, so as to keep them "separate" from other values which are to be stored at the same picture element locations. Taking $[2^{-r}a_{i,j}]$ eliminates the new value and "shifts" the original value back, while taking $a_{i,j}-2^r[2^{-r}a_{i,j}]$ eliminates the shifted original value. It would also have been possible to formulate proofs which used powers of different primes to keep the values separate, as in Section 2.1.

f when they are successively applied in parallel, we proceed as follows:

Define $a_{o,j}=a_{m+1,j}=a_{i,o}=a_{i,n+1}=v, 1\leq i\leq m, 1\leq j\leq n$, where v is a value not taken on by any $a_{i,j}$ or $a^*_{i,j}, 1\leq i\leq m$, $1\leq j\leq n$. Let all $a_{i,j}, a^*_{i,j}$ and v lie in the range 0 to 2^r-1.

Define functions $\psi_o, \psi_R, \psi_L, \psi_U, \psi_D$ as follows:

$$\psi_o(a_{i,j})=2^r a_{i,j}+a_{i,j}$$

$$\psi_R(a_{i,j})=2^r[2^{-r}a_{i,j}]+a_{i,j-1}-2^r[2^{-r}a_{i,j-1}]$$

$$\psi_L(a_{i,j})=2^r[2^{-r}a_{i,j}]+a_{i,j+1}-2^r[2^{-r}a_{i,j+1}]$$

$$\psi_U(a_{i,j})=2^r[2^{-r}a_{i,j}]+a_{i+1,j}-2^r[2^{-r}a_{i+1,j}]$$

$$\psi_D(a_{i,j})=2^r[2^{-r}a_{i,j}]+a_{i-1,j}-2^r[2^{-r}a_{i+1,j}]$$

Then $\psi_R(\psi_o(a_{i,j}))=2^r a_{i,j}+a_{i,j-1}$, so that ψ_R applied after ψ_o preserves the original element values $a_{i,j}$ multiplied by 2^r, but shifts the unmultiplied element values one step to the right. Similarly, ψ_L, ψ_U and ψ_D, applied after ψ_o, shift the unmultiplied element values one step to the left, upward and downward, respectively.

Let g take $a_{i,j}$ into $[2^{-r}a_{i,j}]$ if $a_{i,j}-2^r[2^{-r}a_{i,j}]=v$, and into $f([2^{-r}a_{i-1,j-1}],\ldots,[2^{-r}a_{i+1,j+1}])$, otherwise. Then it is easily verified that the following sequence of local operations, each applied in parallel, produces the same result as applying f sequentially:

$$\psi_o, \psi_R^{n-1}, \psi_L^{n-1}, \psi_D^{m-1}, \psi_U^{m-1}, g,$$

$$\psi_o, \psi_R^{n-2}, \psi_L^{n-1}, \psi_R, \psi_D^{m-1}, \psi_U^{m-1}, g,$$

....

$$\psi_o, \psi_R, \psi_L^{n-1}, \psi_R^{n-2}, \psi_D^{m-1}, \psi_U^{m-1}, g,$$

$\psi_o, \psi_L^{n-1}, \psi_R^{n-1}, \psi_D^{m-1}, \psi_U^{m-1}, g;$

$\psi_o, \psi_R^{n-1}, \psi_L^{n-1}, \psi_D^{m-2}, \psi_U^{m-1}, \psi_D, g,$

$\psi_o, \psi_R^{n-2}, \psi_L^{n-1}, \psi_R, \psi_D^{m-2}, \psi_U^{m-1}, \psi_D, g,$

....

$\psi_o, \psi_L^{n-1}, \psi_R^{n-1}, \psi_D^{m-2}, \psi_U^{m-1}, \psi_D, g;$

::::

$\psi_o, \psi_R^{n-1}, \psi_L^{n-1}, \psi_D, \psi_U^{m-1}, \psi_D^{m-2}, g,$

$\psi_o, \psi_R^{n-2}, \psi_L^{n-1}, \psi_R, \psi_D, \psi_U^{m-1}, \psi_D^{m-2}, g,$

....

$\psi_o, \psi_L^{n-1}, \psi_R^{n-1}, \psi_D, \psi_U^{m-1}, \psi_D^{m-2}, g;$

$\psi_o, \psi_R^{n-1}, \psi_L^{n-1}, \psi_U^{m-1}, \psi_D^{m-1}, g,$

$\psi_o, \psi_R^{n-2}, \psi_L^{n-1}, \psi_R, \psi_U^{m-1}, \psi_D^{m-1}, g,$

....

$\psi_o, \psi_L^{n-1}, \psi_R^{n-1}, \psi_U^{m-1}, \psi_D^{m-1}, g$

(mn subsequences ending in g; the general subsequence is

$\psi_o, \psi_R^{n-j}, \psi_L^{n-1}, \psi_R^{j-1}, \psi_D^{m-i}, \psi_R^{m-1}, \psi_D^{i-1}, g$).

In fact, the general subsequence successively converts every element $a_{s,t}$ of the matrix to the right, left, above and below the (i,j) element to the value $2^r a_{s,t}+v$, while the (i,j) element retains the value $2^r a_{i,j}+a_{i,j}$; g then operates like f on the (i,j) element, while taking every other element back into $a_{s,t}$. Thus applying the subsequences in succession ((i,j)=(1,1),(1,2),...,(1,n),(2,1),...,(2,n),...,(m,1),..., (m,n)) is the same as applying f sequentially.