Sequential Operations in Digital Picture Processing

AZRIEL ROSENFELD AND JOHN L. PFALTZ
University of Maryland, College Park, Maryland*

Abstract. The relative merits of performing local operations on a digitized picture in parallel or sequentially are discussed. Sequential local operations are described which label the connected components of a given subset of the picture and compute a "distance" from every picture element to the subset. In terms of the "distance" function, a "skeleton" subset is defined which, in a certain sense, minimally determines the original subset. Some applications of the connected component and distance functions are also presented.

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.

In this paper it is suggested 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.

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 dis-

* Computer Science Center. The research described in this paper was supported in part by NASA Grant NsG-398 to the University of Maryland, and in part under NASA Contract NAS5-3461 with the Budd Company, McLean, Virginia.

Journal of the Association for Computing Machinery, Vol. 13, No. 4 (October, 1966), pp. 471–494

crete 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 ($1 \leqq i \leqq m$, $1 \leqq j \leqq n$) 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 $v_1, \cdots, v_k$. In practice, m and n are usually not greater than 2^{10}, and k is rarely greater than 2^6. We assume in what follows that the values are nonnegative integers.

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 only operations of this type are considered.[1]

Local operations can also be used to define *local properties* of a picture. If $f: a_{i,j} \to a^*_{i,j}$ is a local operation, one can speak of *the property that* $a^*_{i,j} = v$, where v is a picture element value.

It is sometimes convenient to consider operations which involve more than one picture at a time. Strictly speaking, however, this is no more general than operating on a single picture. In fact, we have:

LEMMA. *Let* (a_{ij}), (b_{ij}) *be* m *by* n *pictures; then there exist an* m *by* n *picture* (c_{ij}) *and two functions* g, h *such that* $g(c_{ij}) = a_{ij}$, $h(c_{ij}) = b_{ij}$ *for all* i, j.

PROOF. Let $c_{ij} = 2^{a_{ij}}3^{b_{ij}}$, so that c_{ij} is a positive integer; let $g = \phi_2$, $h = \phi_3$, where ϕ_2, ϕ_3 are defined for positive integer x by

$$\phi_2(x) = \max \{k \mid k \text{ nonnegative integer}; 2^k \text{ divides } x\},$$

$$\phi_3(x) = \max \{k \mid k \text{ nonnegative integer}; 3^k \text{ divides } x\}.$$

COROLLARY. *Let* f *be any function which takes a* pair *of* m *by* n *pictures into an* m *by* n *picture, say* $f((a_{ij}), (b_{ij})) = (d_{ij})$. *Then there exist a picture* (c_{ij}) *and a function* f^* *such that* $f^*(c_{ij}) = (d_{ij})$.

PROOF. Take (c_{ij}), g, h, as in the proof of the Lemma and define $f^*(c_{ij}) = f(g(c_{ij}), h(c_{ij}))$.

Evidently this argument generalizes immediately to operations on any number of pictures. Note also that if f is local—in other words, if d_{ij} depends only on $a_{i-1,j-1}$,

[1] 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 $(m+1)$-st row and an $(n+1)$-st column, giving these fictitious elements a value which does not occur in the "real" picture array, and defining the function appropriately when some of its arguments have this value. In many of the cases considered in what follows, one can avoid this complication by simply adding to the definition of f the phrase "for whichever of these elements is defined."

$\cdots$, $a_{i+1,j+1}$ and on $b_{i-1,j-1}$, $\cdots$, $b_{i+1,j+1}$—then f^* is also local, since c_{ij} depends only on a_{ij} and b_{ij}.

When processing a picture using local operations, it is often convenient to store the results of intermediate processing steps as auxiliary pictures. By the Lemma and the last remark, local operations involving such auxiliary pictures can be regarded as operations on a single picture. In Section 3 and in the Appendix repeated advantage is taken of this convenience.

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, and so on.

Each of the local operations used in the examples just given is performed independently on every point of the picture. The arguments $(a_{i-1,j-1}, \cdots, a_{i+1,j+1})$ are always the original picture matrix values; the new values $a^*_{ij} = f(a_{i-1,j-1}, \cdots, a_{i+1,j+1})$ 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. In this paper the countersuggestion is made 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 be defined in what follows, 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 any 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*.

The particular processing sequence just described will be called the (forward) *raster* sequence. There will be occasion in the next two sections to use other sequences as well.

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 to guarantee that a sequential local operation is matched by parallel local operations, a long sequence of the latter may be required. In practice, 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, particularly if a sequential computer must be used.

As an illustration, consider the distance transformation defined in Section 4. It can be performed by two sequentially applied local operations, involving a total of $2mn$ individual local operations. On the other hand, it is easily verified that this transformation can also be accomplished by applying the local operation

$$f(a_{i,j}) = \min (a_{i-1,j}, a_{i+1,j}, a_{i,j-1}, a_{i,j+1}) + 1 \quad \text{if} \quad a_{i,j} \neq 0, \quad f(0) = 0,$$

$m+n$ times in parallel to every picture element. If this is done on a sequential computer, it involves $mn(m+n)$ individual local operations—that is, $(m+n)/2$ times as many as required by the method of Section 4. Note, however, that if a parallel computer is available, it need perform only $m+n$ local operations on the picture in parallel, a saving by a factor of $2mn/(m+n)$ over the method of Section 4. 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*

A set of sequentially applied local operations which "labels" the connected components of any given picture subset is described in this section. For simplicity, it is

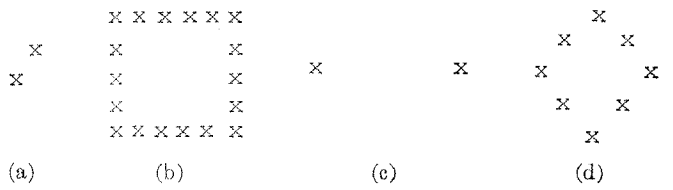

FIG. 1. Examples of connected and nonconnected sets

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 is called *connected* if for any two points P and Q of the subset there exists a sequence of points $P = P_0, P_1, P_2, \cdots, P_{n-1}, P_n = Q$ of the subset such that P_i is a neighbor of P_{i-1}, $1 \leqq i \leqq n$. In Figure 1(a), the set of x's and the set of blank points are both connected; in Figure 1(b), the x's are connected but the blanks are not; in Figure 1(c), the blanks are connected but the x's are not. For these examples, the definition agrees with the intuitive concept of connectivity. On the other hand, both the x's and blanks in Figure 1(d) 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 x's and blanks in Figure 1d 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. This is of course not a mathematical paradox, but it is unsatisfying intuitively; nevertheless, connectivity is still a useful concept. It should be noted that if a digitized picture is defined as an array of hexagonal, rather than square, elements, the paradox disappears; this is because in the hexagonal case an element has an edge in common with every one of its six neighbors.

In general, a subset of a picture (say the subset of "0" points) consists of a number of connected parts or *components*.[2] The problem of distinguishing among these components is now to be considered. Specifically, it is desired to construct a transformed picture in which the 0 points have new values $v_1, \cdots, v_k$ (positive integers each greater than 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

[2] Formally, these components are the *equivalence classes* of picture points defined by the relation "is connected to" (that is: "is a neighbor of a neighbor . . . of a neighbor of"), which is evidently an equivalence relation.

special value is assigned. Further processing is then applied to these special values so as to eliminate redundant values from the picture.

In Section 3.2 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 Section 3.3 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.

3.2 *Sequential operations for determining connected components of a picture*

Let f be the local operation which takes $a_{i,j}$ into $a'_{i,j}$ defined by:

(a) If $a_{i,j} = 1$, then $a'_{i,j} = 1$.

(b) If $a_{i,j} = 0$, and $a'_{i-1,j-1} = a'_{i-1,j} = a'_{i-1,j+1} = a'_{i,j-1} = 1$, then $a'_{i,j} = v_k$, where v_k is one of a set of as yet unused labels. (This indicates that $a_{i,j}$ is possibly the start of a new connected component of 0's. If it is not desired to allow an operation which can draw on a set of labels in this way, one can simply use 2^i3^j as the label; these labels are automatically distinct for all i, j.)

(c) If $a_{i,j} = 0$ and each of $a'_{i-1,j-1}$, $a'_{i-1,j}$, $a'_{i-1,j+1}$ and $a'_{i,j-1}$ is either 1 or some v_k (but not all of them are 1), then $a'_{i,j}$ is the smallest of the v_k's. (If there is only one v_k, this indicates that $a'_{i,j}$ belongs to that same component. If there is more than one, it indicates that two or more components were actually parts of the same component.)

Furthermore, let f also create an auxiliary picture (b_{ij}) in which $b_{i,j} = 1$ when more than one v_k is involved in case (c), and $b_{i,j} = 0$ otherwise. (These 1's thus label points at which two or more components have "merged." It is clear that (b_{ij}) could be combined with (a'_{ij}), if desired, by the method of the Lemma in Section 2.1.) When f is applied to a binary picture sequentially, e.g., in forward raster sequence, it labels the 0's in such a way that 0's which get the same label must belong to the same connected component; note, however, that the 0's in a given component may have several labels.

To complete the task of labeling the connected components, it remains only to eliminate all labels which have merged with other ones. For example, if v_r has "met" v_s, where $v_r < v_s$, one should replace all the v_s's prior to the meeting point with v_r's.

If one were not restricted to performing only local operations on the picture, it would be fairly easy to eliminate the redundant labels by processing a list of the redundant pairs. A method for doing this is described in Section 4. In the remainder of this section, elimination of redundancies using local operations only is described.

After f has been performed, the redundancies are identified by 1's in (b_{ij}) at the points where they were detected. To eliminate a redundancy, say of v_s with v_r, *using only local operations*, it is necessary to convey the information that v_s "equals" v_r to the neighborhood of every point (i, j) of the picture, so that if $a'_{i,j}$ has value v_s, it can be replaced by v_r. This is done for one redundancy at a time by proceeding as follows.

(1) *Pick the first redundancy not yet processed.* This can be done by applying a local operation to (b_{ij}) in *zigzag* sequence $(1, 1), \cdots, (1, n), (2, n), \cdots, (2, 1), (3, 1), \cdots, (3, n), \cdots, (m, 1), \cdots, (m, n)$ if m is odd; $\cdots, (m, n), \cdots, (m, 1)$ if m is even. This operation g takes (b_{ij}) into (b'_{ij}), and also generates an auxiliary pic-

ture (c_{ij}), such that $b'_{1,1} = c_{1,1} = 0$ and the succeeding elements are defined as follows:

For $(i, j) \neq (1, 1)$, let (i', j') denote the predecessor of (i, j) in the zigzag sequence.

If $c_{i',j'} = b_{i,j} = 0$, then $b'_{i,j} = c_{i,j} = 0$.

If $c_{i',j'} = 0$ and $b_{i,j} = 1$, then $b'_{i,j} = 0$ and $c'_{i,j} = 1$.

If $c_{i',j'} = 1$ or 2, then $b'_{i,j} = b_{i,j}$ and $c_{i,j} = 2$.

Thus g "erases" the first 1 in (b_{ij}) ("first" in the sense of the zigzag sequence) and stores a 1 in the corresponding position in (c_{ij}). (Note that the first 1 in (b_{ij}) cannot be in the first row, since components cannot yet have merged; hence setting $b'_{1,1} = c_{1,1} = 0$ is safe.) As soon as it has done this, it stores 2's in (c_{ij}) thereafter; this keeps it from mistaking subsequent 1's in (b_{ij}) for "first." The reason for using the zigzag sequence is to ensure that (i', j') is always a neighbor of (i, j), so that g is local.

(2) *"Transmit" the information about this redundancy to every point.* Let $c_{x,y} = 1$, and let $v_p < v_q \leqq v_r \leqq v_s$ be labels of $a'_{x-1,y-1}$, $a'_{x-1,y}$, $a'_{x-1,y+1}$ or $a'_{x,y-1}$. It is desired to express the fact that the labels v_q, v_r, v_s are to be replaced by v_p. This can be done by using the integer $v = 2^{v_p}3^{v_q}5^{v_r}7^{v_s}$ as a label, since this integer carries the values v_p, v_q, v_r, v_s in retrievable form, as well as indicating which of them is to replace the others. To get this information to every point, one constructs an auxiliary picture every element of which has value v. Specifically, one defines the local operation h_1, taking (c_{ij}) into (d_{ij}) such that $d_{1,1} = 0$ and:

If $d_{i',j'} = 0$ and $c_{i,j} \neq 1$, then $d_{i,j} = 0$.

If $d_{i',j'} = 0$ and $c_{i,j} = 1$, then $d_{i,j} = v$, where v is defined in terms of the elements $a_{i-1,j-1}$, $a_{i-1,j}$, $a_{i-1,j+1}$ and $a_{i,j-1}$ as described just above.

If $d_{i',j'} = v$, then $d_{i,j} = v$.

Thus h_1, applied in zigzag sequence, constructs a picture (d_{ij}) in which every element after the 1 in (c_{ij}) is v. (In particular, $d_{m,n} = v$.) To make the remaining elements v's, one need only apply a simple local operation h_2, in *reverse* zigzag sequence. Let (i'', j'') denote the predecessor of (i, j) in this sequence, and let h_2 take (d_{ij}) into (d^*_{ij}) such that $d^*_{m,n} = d_{m,n} = v$, and $d^*_{i,j} = v$ whenever $d_{i'',j''} = v$.

(3) *Use this information to correct the redundancy.* It remains only to define the function h by

$$h(a'_{i,j}) = v_p \quad \text{if} \quad a'_{i,j} = v_q, v_r \text{ or } v_s,$$

$$h(a'_{i,j}) = a'_{i,j} \qquad \text{otherwise.}$$

This is a local operation—indeed, it operates only on a single (i, j) at a time—since $(d^*_{i,j})$ has brought these v's to every (i, j); it can be applied in any sequence. The resulting transform of (a'_{ij}) has all its v_q's, v_r's and v_s's replaced by v_p's, and is otherwise the same as a'_{ij}.

We now *repeat the process* described in (1)–(3), but starting with the new (a'_{ij}) and with (b'_{ij}). Application of g in zigzag sequence now erases the first 1 in (b'_{ij}), say $b_{z,w}$ (which was the second 1 in (b_{ij})), and leaves the rest of it unchanged; it also constructs a new (c_{ij}) with a 1 in the (z, w) position. Application of h_1 and h_2 then yields a new (d^*_{ij}) with appropriate label v (not the same as the previous v). It is quite possible that the redundancy at (z, w) involved only v_p, v_q, v_r or v_s,

and is now eliminated; in this case the new v reduces to 2^{v_p}, and application of h does nothing to (a'_{ij}). In any event, this second cycle of (1)–(3) yields an (a'_{ij}) in which further possible redundancies have been eliminated.

Let this entire process be repeated as many times as necessary. (One can, of course, stop when there are no more 1's in (b_{ij}), but if only local operations are involved, this can only be detected by performing a counting operation on (b_{ij}) after each iteration. Certainly mn repetitions—in fact, considerably fewer—will suffice to clear up all the redundancies in (a'_{ij}).)

In summary: Given (a_{ij}) in which every element is 0 or 1, one can construct (a'_{ij}) in which every connected component of 0's has a unique label by proceeding as follows:

(a) Apply f to (a_{ij}) in (e.g.) raster sequence (zigzag would do just as well).

(b) Iterate the following sequence mn times: (g in zigzag sequence; h_1 in zigzag sequence; h_2 in reverse zigzag sequence; h in any sequence).

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.[3]

(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, since processing is not constrained to neighborhoods within the table; (3) it is easy to stop the processing when the table is exhausted, rather than blindly repeating it mn times.

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

(1) Apply f as described in Section 3.2; but rather than generating an auxiliary picture (b_{ij}), simply store the redundant labels in the first unused place in a table T. The ith entry in this table thus has the form $(v_{pi}, v_{qi}, v_{ri}, v_{si})$, where $v_{pi} < v_{q_i} \leqq v_{ri} \leqq v_{si}$.

(2) To process the table:

(a) Order the entries lexicographically (in order of increasing first value; for each of these, in order of increasing second value; and so on).

(b) Store $(v_{p1}, v_{q1}, v_{r1}, v_{s1})$ in a second table T'. In the remaining entries, replace every v_{q1}, v_{r1} and v_{s1} by v_{p1}. Reorder each entry (if necessary) to re-establish $v_{pi} < v_{qi} \leqq v_{ri} \leqq v_{si}$; if all terms are equal, erase the entry.

[3] 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.

Repeat steps (a) and (b) until every entry has been erased or stored in T'. When this is finished, T' consists of a set of entries in lexicographic order. The first term of each entry is the smallest representative of an equivalence class of redundant values; the remaining terms of the entries which have first term v are the remaining elements of its equivalence class.

(3) Scan the picture in any sequence, comparing the value of each element with the entries in T'. Whenever a value is found to be in T', but not as a first term, replace it by the corresponding first term. (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 2,000 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 ($0 \leqq t_1 \leqq t_2 < 2^6-1$), 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 three labels in the original processing, since it was detected as a possible "new" component at each of the three elements which are circled in the figure. Component D obviously had many labels originally, while components A and C had unique labels throughout the processing.

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

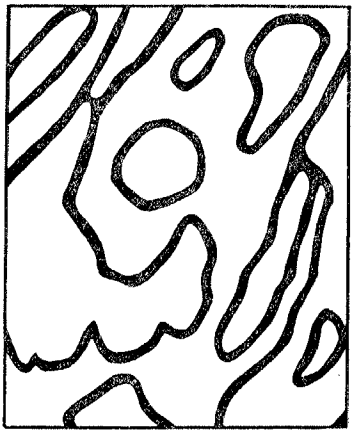

FIG. 2. Picture input to processing programs

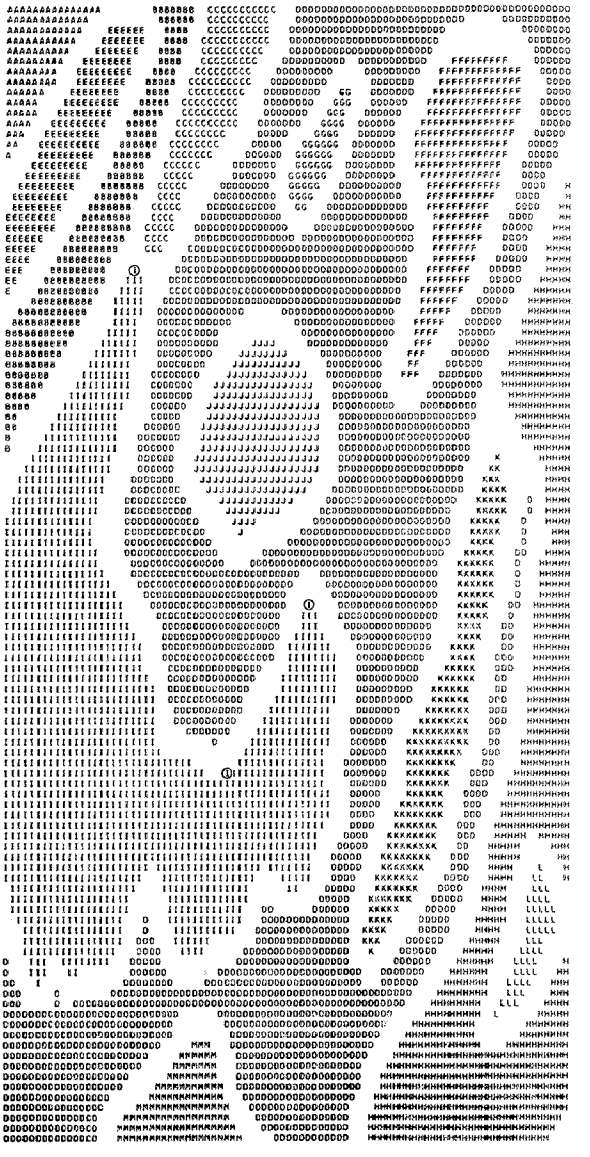

FIG. 3. Connected component transform of Figure 2

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 $P = P_0, P_1, \cdots, P_n = Q$ with P_i a neighbor of P_{i-1}, $1 \leqq i \leqq n$. 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 $d \geqq d^*$. For simplicity, the restricted definition is used in the remainder of this paper.

Evidently, $d(P, Q)$ (and similarly for d^*) has all the properties of a *metric*.[4] 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.[5]

4.2 *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

$$\begin{aligned} f_1(a_{i,j}) &= 0 && \text{if } a_{i,j} = 0, \\ &= \min(a_{i-1,j} + 1, a_{i,j-1} + 1) && \text{if } (i,j) \neq (1,1) \text{ and } a_{i,j} = 1, \\ &= m + n && \text{if } (i,j) = (1,1) \text{ and } a_{1,1} = 1, \\ f_2(a_{i,j}) &= \min(a_{i,j}, a_{i+1,j} + 1, a_{i,j+1} + 1). \end{aligned}$$

Since no two points of the picture can be distance $m+n$ apart, we know that $a_{1,1}$ is at a distance less than $m+n$ from the set of 0's, if this set is nonempty; thus the

[4] 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.

[5] For the metric d^*, the corresponding locus is an upright square of side $2d+1$ centered at P.

final value of $a_{1,1}$ (or of any other element labeled $m+n$ by f_1) will be the value assigned to it by f_2.

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* reverse *raster sequence. Then C is the distance transform of A.*

PROOF. Note first that if $a_{i,j} = 1$ 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 is less than 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 are greater than or equal to 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 $c_{i-1,j} \leqq k - 1$; contradiction.

The distance transforms for a circle, two rectangles[6] 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.

4.3 *Distance skeleton*

Blum has suggested [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 is defined for the distance transform introduced above,[7] and it is 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 $G(A) = (c_{ij})$, 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 less than or equal to* 1.

PROOF. Let $A = (a_{ij})$, $G_1(A) = (b_{ij})$. By definition of g_1,

$$b_{i,j} \geqq b_{i,j-1} - 1 \quad \text{and} \quad b_{i,j} \geqq b_{i-1,j} - 1, \qquad \text{for all } i, j,$$

[6] The two rectangles actually have the same proportions; the difference between their shapes in the figure results from the unequal horizontal and vertical size of a character space. The circle appears distorted for the same reason.

[7] 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.

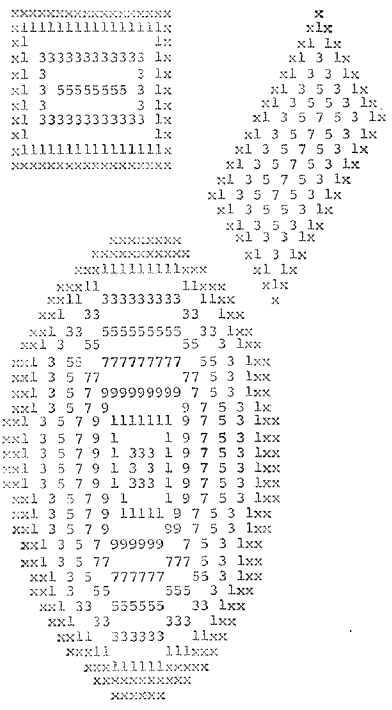

FIG. 4a. Distance transforms of two rectangles and a circle

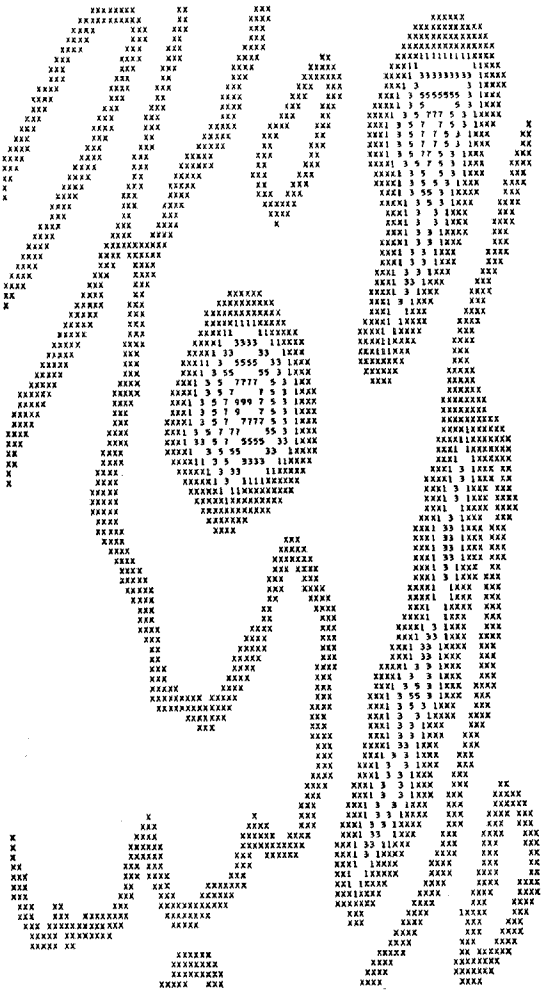

FIG. 4b. Distance transforms of components F, J, K of Figure 3

so that

$$b_{i,j+1} \geqq b_{i,j} - 1 \quad \text{and} \quad b_{i+1,j} \geqq b_{i,j} - 1, \qquad \text{for all } i, j.$$

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

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

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

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

If it can be shown 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 Lemma 1 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 is less than or equal to $c_{i,j} + 1$. 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 $c_{m-1,n} \leqq c_{m,n} + 1$. This completes 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

$$\begin{aligned} g_1(a_{i,j}) &= \max (a_{i,j}, g_1(a_{i,j-1}) - 1, g_1(a_{i-1,j}) - 1) \\ &= \max (a_{i,j}, a_{i,j-1} - 1, a_{i-1,j} - 1) \end{aligned}$$

by induction hypothesis, and this equals $a_{i,j}$ 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 $G_1(G_1(A)) = G_1(A)$; 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 4.2, 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. The proof is as for the Corollary to Lemma 2.

LEMMA 4. *Let $A = (a_{ij})$, $B = (b_{ij})$ be pictures such that $a_{i,j} \leqq b_{i,j}$ 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 $c_{i,j} \leqq d_{i,j}$ for all i, j, so that $c_{h,k} \leqq d_{h,k} = a_{h,k}$. But G never decreases the value of a picture element; hence $a_{h,k} \leqq c_{h,k}$.

If $P = (p_{ij})$ is any picture, $P' = (p'_{ij})$ will be called a *partial picture* of P if $p'_{i,j} = p_{i,j}$ or 0 for all i, j.

COROLLARY. *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.

LEMMA 5. *Let $A = (a_{ij})$ be a picture, $G(A) = (c_{ij})$, and let $a_{h,k} < V$; let $c_{h-1,k}$, $c_{h,k-1}$, $c_{h+1,k}$, $c_{h,k+1}$ all be less than or equal to 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 $b_{h-1,k} \leqq c_{h-1,k} \leqq V$, $b_{h,k-1} \leqq c_{h,k-1} \leqq V$, 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 *skeleton* of T. In order words, T^* is the set of *local maxima* of the distance transform. We assume that T is not the trivial picture every element of which has value $m+n$.

THEOREM. *$G(T^*) = T$, 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 P is any picture with integer-valued elements, let P_k be the set of elements of P 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 $k(= N, N-1, \cdots, 1, 0)$, 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 less than or equal to M (Lemma 1), while its value in T' is $0 < M$, so that by Lemma 5 its value in $G(T')$ is still less than M, contradicting $G(T') = T$. Thus T' must contain every element of T^* of value greater than 0; this completes 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 the two rectangles in Figure 5 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 for the circle in Figure 5.

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].

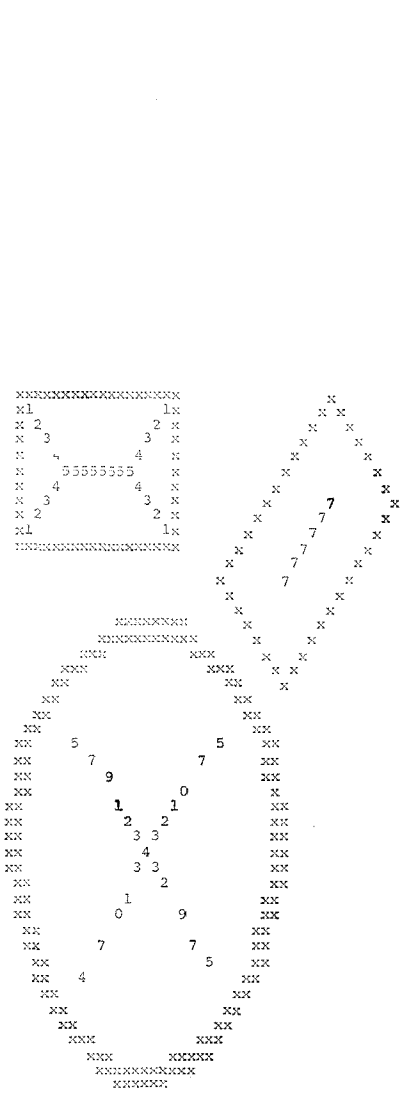

FIG. 5a. Skeletons for Figure 4a

FIG. 5b. Skeletons for Figure 4b

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[8]; 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.

5.1 *Adjacency and order of connectivity*

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

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

connectivity transformation are: (a) constructing the *graph* corresponding to a given dissection of a picture into connected components, and (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 ith and jth nodes are joined by an arc; $a_{ij} = 0$ otherwise.

In Figure 3, the connected components labeled A, $\cdots$, 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 in Table I.

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 2 by 2 subpicture and to enter 1's in the

TABLE I

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

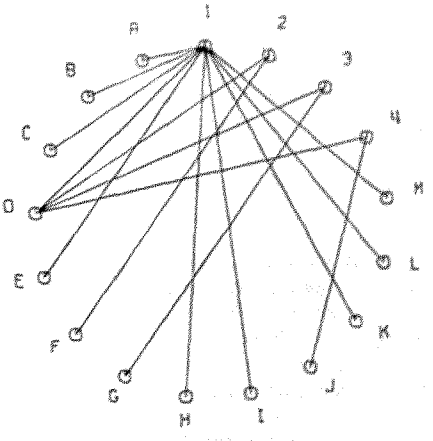

FIG. 6. Graph for Figure 3

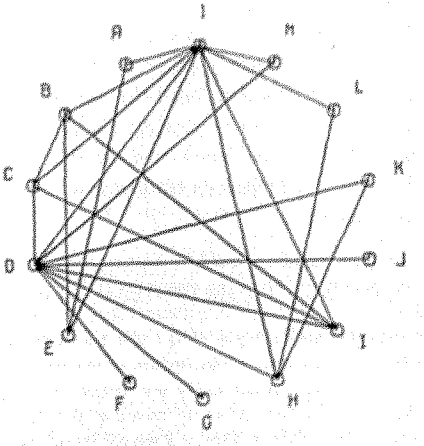

FIG. 7. Proximity graph for Figure 3

TABLE II

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

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 in 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 incidence matrix of the graph after deleting the corresponding row and column.[9] A FORTRAN program which performs this analysis has been written; its output for Figure 3 is shown in Table II.

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 labeled 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 labeled 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.[10] 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.

[9] 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 labeling 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.

[10] 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 of Figure 3 would be adjacent, and K would be adjacent to the picture exterior.

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). 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 labeled 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.[11]

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 in what follows 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 8(a)). On the other hand, if the figure is a very elongated rectangle (Figure 8(b)), 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 Figures 8(c)–(e).

As the solid curves in Figure 8, especially parts (a) and (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.

[11] 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.

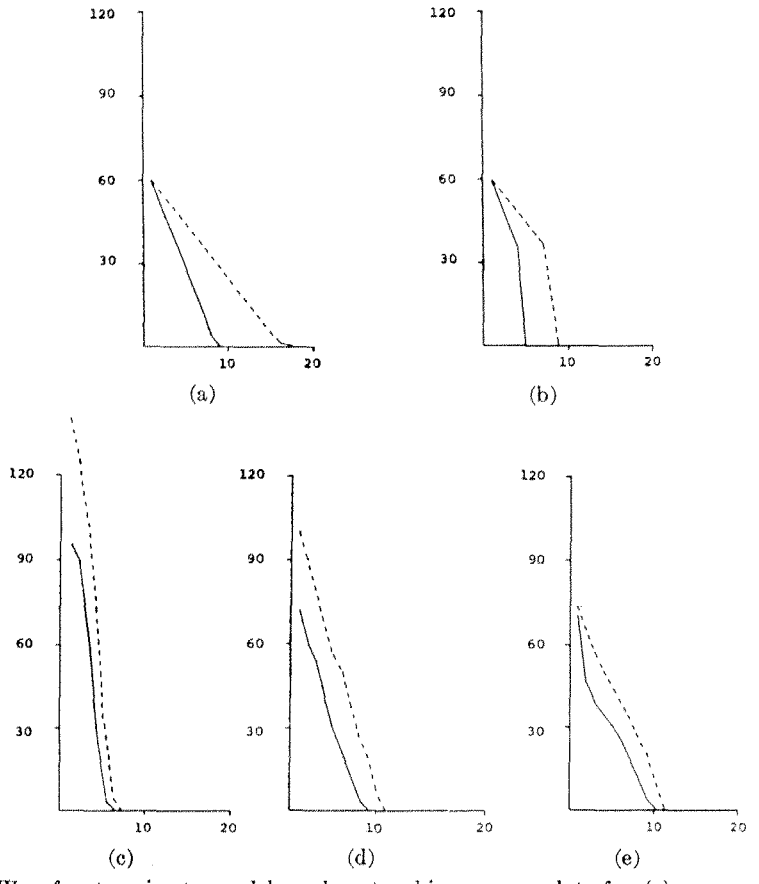

FIG. 8. Wavefront perimeter and boundary-touching square plots for (a) a square; (b) an elongated rectangle; (c) component K of Figure 3 (elongated); (d) component F of Figure 3 (partly elongated); (e) component J of Figure 3 (roughly circular)

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. 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.[12]

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 is defined in Section 6.2, using the distance skeleton concept introduced in Section 4.3.

[12] 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.

FIG. 9a. Lake, river and streams

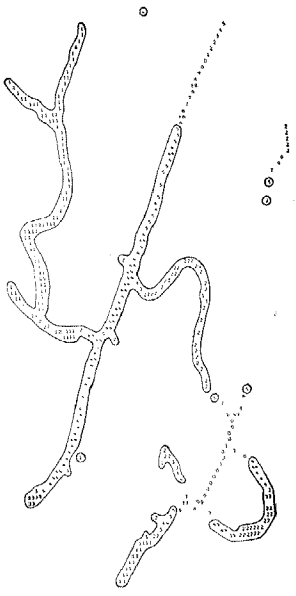

FIG. 9b. Skeleton of Figure 9a

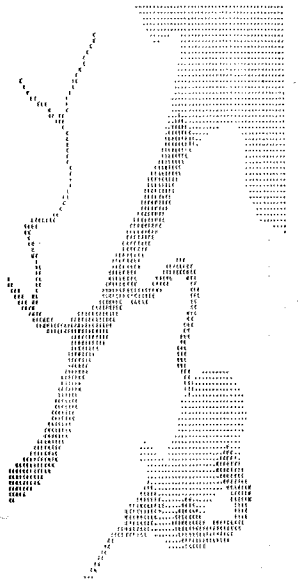

FIG. 9c. "Elongated" parts of Figure 9a

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 in what follows 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.[13] 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.

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

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 $k = 2, 3, \cdots$, up to half the picture diameter if necessary, consider the set S_k of skeleton points which have values less than or equal to 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 $5k$–$10k$,[14] corresponding to a set of proximate skeleton points whose "length" is at least $2\frac{1}{2}$ 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.

Appendix. Proof of the Equivalence of Parallel and Sequential Local Operations

It must be shown that any local operation applied in parallel is equivalent to a series of local operations applied sequentially, and vice versa.

Note first that if the local operation f is applied to each element of the picture A, and the results are stored in another picture B, rather than modifying A, then it makes no difference whether f is applied to A in parallel or sequentially, since the arguments of f are always the original elements of A. Hence if f applied in parallel takes A into A^*, then f applied in any sequence, *using B for storage of the results*, yields $B = A^*$. By the Lemma of Section 2.1, it is known that if necessary, this procedure can be re-expressed in terms of A alone (for example, one could transform A in such a way as to "keep" the original element values as exponents of 2, while storing the transformed values as exponents of 3). It has thus been shown that the result of any parallel local operation can be obtained by applying a local operation sequentially. (Strictly speaking, an additional operation of "erasing" A and "replacing" it by $B = A^*$ should also be performed; this operation does not even involve neighbors, and can be performed either sequentially or in parallel. In the example using exponents given just above, this operation is simply ϕ_3 (see Section 2.1).)

Conversely, let f be a local operation which takes A into A^* when applied sequentially. Let $A = (a_{ij})$, $A^* = (a^*_{ij})$, $1 \leqq i \leqq m$, $1 \leqq j \leqq n$, and define $a_{0,j} = a_{m+1,j} = a_{i,0} = a_{i,n+1} = v$, $1 \leqq i \leqq m$, $1 \leqq j \leqq n$, where v is different from every $a_{i,j}$ and $a^*_{i,j}$, $1 \leqq i \leqq m$, $1 \leqq j \leqq n$. Now define:

$$(1)\ f_1(a_{i,j}) = f(a_{i-1,j-1}, \cdots, a_{i+1,j+1}), \quad \text{if} \quad a_{i-1,j} = a_{i,j-1} = v;$$
$$= a_{i,j} \text{ otherwise}; \qquad 1 \leqq i \leqq m, \quad 1 \leqq j \leqq n.$$

(Since $a_{i-1,j} = a_{i,j-1} = v$ is equivalent to $(i, j) = (1, 1)$, applying this f_1 in parallel to all of A is equivalent to applying f to $a_{1,1}$ only.) At the same time, let f_1 generate an auxiliary picture (b_{ij}) such that

$$\left.\begin{aligned} b_{i,j} &= w \quad \text{if } a_{i-1,j} = a_{i,j-1} = v \\ &\qquad \text{(i.e., if } (i, j) = (1, 1)) \\ &= a_{i,j} \quad \text{otherwise} \end{aligned}\right\}, \quad 1 \leqq i \leqq m, \quad 1 \leqq j \leqq n,$$

[14] This uncertainty can be reduced if the skeleton locus is "thinned" before this step is performed.

where w is different from every $a_{i,j}$ and $a^*_{i,j}$ as well as from v.

$$\left.\begin{aligned} (2)\ f_2(a_{i,j}) &= f(a_{i-1,j-1}, \cdots, a_{i+1,j+1}) \\ &\quad \text{if } a_{i-1,j} = v \text{ and } b_{i,j-1} = w \\ &= a_{i,j} \text{ otherwise} \\ f_2(b_{i,j}) &= w \text{ if } a_{i-1,j} = v \text{ and } b_{i,j-1} = w \\ &= b_{i,j} \text{ otherwise} \end{aligned}\right\}, \quad 1 \leqq i \leqq m, \quad 1 \leqq j \leqq n.$$

(If f_1 has been applied to A, the unique (i, j) satisfying $a_{i-1,j} = v$ and $b_{i,j-1} = w$ is $(1, 2)$. Thus applying f_2 in parallel to $f_1(A)$ is equivalent to applying f to $a_{1,2}$. Application of f_2 also puts a w in the $(1, 2)$ position; hence if it is applied *again*, the unique (i, j) satisfying the conditions is now $(1, 3)$. Hence repetition of f_2 $n-1$ *times* is equivalent to applying f sequentially to $a_{1,2}, \cdots, a_{1,n}$.)

$$\left.\begin{aligned} (3)\ f_3(a_{i,j}) &= f(a_{i-1,j-1}, \cdots, a_{i+1,j+1}) \\ &\quad \text{if } a_{i-1,j} = w \text{ and } a_{i,j-1} = v \\ &= a_{i,j} \text{ otherwise} \\ f_3(b_{i,j}) &= w \text{ if } b_{i-1,j} = w \text{ and } a_{i,j-1} = v \\ &= b_{i,j} \text{ otherwise} \end{aligned}\right\}, \quad 1 \leqq i \leqq m, \quad 1 \leqq j \leqq n.$$

(If f_1 has been applied, these conditions are satisfied by $(i, j) = (2, 1)$ only. Hence applying f_3 after f_1 and $n-1$ f_2's have been applied has the same effect as applying f to $a_{2,1}$ after having applied it to $a_{1,1}, \cdots, a_{1,n}$.)

$$\left.\begin{aligned} (4)\ f_4(a_{i,j}) &= f(a_{i-1,j-1}, \cdots, a_{i+1,j+1}) \\ &\quad \text{if } b_{i-1,j} = b_{i,j-1} = w \\ &= a_{i,j} \text{ otherwise} \\ f_4(b_{i,j}) &= w \text{ if } b_{i-1,j} = b_{i,j-1} = w \\ &= b_{i,j} \text{ otherwise} \end{aligned}\right\}, \quad 1 \leqq i \leqq m, \quad 1 \leqq j \leqq n.$$

(Readily, after f_1, $n-1$ f_2's and f_3 have been applied, this f_4 singles out the $(2, 2)$ element; and repeating it a total of $n-1$ times singles out the $(2, 3), \cdots, (2, n)$ elements, successively. Moreover, after this has been done, application of f_3 again will single out the $(3, 1)$ element; successive applications of f_4 after this will pick the $(3, 2), \cdots, (3, n)$ elements; and so on.)

In summary: Applying f to A sequentially is equivalent to applying the following series of parallel local operations to A:

$$f_1; f_2, \cdots, f_2, \quad (n-1 \text{ times}); \qquad [f_3; f_4, \cdots, f_4, \quad (n-1 \text{ times})],$$

with the bracketed series of operations repeated a total of $m-1$ times, giving a total of mn parallel operations.

Acknowledgments. 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, Virginia. 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.

RECEIVED JANUARY, 1966; REVISED APRIL, 1966

REFERENCES

1. DINNEEN, G. P. Programming pattern recognition. Proc. Western Joint Comput. Conf., Vol. 7, 1955, pp. 94–100.
2. KIRSCH, R. A., ET AL. Experiments in processing pictorial information with a digital computer. Proc. Eastern Joint Comput. Conf., Vol. 12, 1957, pp. 221–229.
3. UNGER, S. H. A computer oriented toward spatial problems. *Proc. IRE 46* (1958), 1744–1750.
4. NARASIMHAN, R. Labeling schemata and syntactic descriptions of pictures. *Inform. Control 7* (1964), 151–179.
5. HARMON, L. D. Line drawing pattern recognizer, *Electronics 33* (Sept. 2, 1960), 39–43; Proc. Western Joint Comput. Conf., Vol. 17, 1960, pp. 351–364.
6. SINGER, J. R. Model for a size invariant pattern recognition system. Bionics Symposium, WADD Tech. Rep. 60-600, Dec. 1960, pp. 239–245.
7. ——. Electronic analog of the human recognition system. *J. Opt. Soc. Amer. 51* (1961), 61–69.
8. ——. A self organizing recognition system. Proc. Western Joint Comput. Conf., Vol. 19, 1961, pp. 545–554.
9. STEVENS, M. E. Abstract shape recognition by machine. Proc. Eastern Joint Comput. Conf., Vol. 20, 1961, pp. 332–351.
10. BLUM, H. An associative machine for dealing with the visual field and some of its biological implications. In Bernard, Eugene E., and Kare, Morley R., Eds., *Biological Prototypes and Synthetic Systems, Vol. 1.* (Proc. Second Annual Bionics Symp., Cornell U., 1961). Plenum Press, New York, 1962, pp. 244–260.
11. ——. A machine for performing visual recognition by use of antenna-propagation concepts. *Proc. IRE Wescon*, Pt. IV, 1962.
12. ——. A transformation for extracting new descriptors of shape. Symp. on Models for the Perception of Speech and Visual Form (Boston, Nov. 1964); in press, M.I.T. Press, Cambridge, Mass.
13. KOTELLY, J. A Mathematical Model of Blum's Theory of Pattern Recognition. AFCRL Res. Rep. 63-164, April, 1963.
14. NUTTALL, T. C. Apparatus for Counting Objects. U. S. Patent 2803406, Aug. 20, 1957.
15. SABBAGH, E. N. Aerial reconnaissance electronic reader. RADC Tech. Rep. 60–132, June, 1960.
16. MINOT, O. N. Counting and outlining of "two-dimensional" patterns by digital computer. U. S. Naval Elect. Lab. TM-414, Aug. 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., pp. 1–13.
19. VON FOERSTER, H. Circuitry of clues to Platonic ideation. In Muses, C. A., Ed., *Aspects of the Theory of Artificial Intelligence* (Proc. First Int. Symp. on Biosimulation, Locarno, Switz., 1960). Plenum Press, New York, 1962, pp. 43–81.
20. BABCOCK, M. L. Some physiology of automata. Proc. Western Joint Comput. Conf., Vol. 19, 1961, pp. 291–298.
21. WESTON, P. Photocell field counts random objects. *Electronics 34* (Sept. 22, 1961), 46–47.
22. TENERY, G. A pattern recognition function of integral geometry. *IEEE Trans. MIL-7* (1963), 196–199.
23. ——. Information flow in a bionics image recognition system. Symp. on Models for the Perception of Speech and Visual Form. (Boston, Nov. 1964); in press, M.I.T. Press, Cambridge, Mass.