SLAC-70
UC-32
TID-4500 (48th Ed.)

TWO-DIMENSIONAL PATTERN DESCRIPTION AND RECOGNITION VIA CURVATUREPOINTS

by

C. T. Zahn

December 1966

Technical Report
Prepared Under
Contract AT(04-3)-515
for the USAEC
San Francisco Operations Office

Printed in USA. Available from CFSTI, National Bureau of Standards,
U. S. Department of Commerce, Springfield, Virginia. 22151
Price: Printed Copy $3.00; Microfiche $0.65.

TABLE OF CONTENTS

LIST OF FIGURES

(All figures follow page 55)

I. INTRODUCTION

Kirsch et al[1] have stated that the aim of pattern recognition is "to reduce the amount of information to the minimum necessary for recognizing one pattern from a group of patterns." It is probably more accurate to say that the aim is to efficiently transform and select information from the input in such a way that the remaining information is both sufficient for the discrimination desired and in a format suitable for the efficient performance of that discrimination by whatever mechanism is being considered.

Narasimhan[2] has recently argued that, "Pattern recognition is only one aspect of the much more fundamental problem of analysis and description of classes of patterns. The aim of any adequate recognition procedure should be not merely to arrive at a 'yes,' 'no,' 'don't know' decision but to produce a structured description of the input picture."

In spite of the apparent contradiction, we are in substantial agreement with both these points of view. In the light of the above it appears desirable to have a method for transforming the input picture into a "structural description" with no loss of information* and with such a format that the selection of information required by a large class of different discrimination and recognition tasks can be efficiently performed by a general purpose digital computer according to algorithms which are natural and straightforward implementations of human intuition. This paper presents such a "structural description" and an efficient method for performing the indicated transformation. A fairly wide range of input pictures has been taken from existing literature as well as other sources to justify the wide range of general usefulness of the method.

*Meaning that the input picture can be recovered unambiguously from the "structural description" by a mechanical algorithm.

II. ASSUMPTIONS

This paper presents a method for obtaining descriptions of that class of two-dimensional pictures whose elements are a finite set of points in the plane each having value zero (white) or one (black) and such that the points are arranged in the form of some uniform lattice or grid-system. Later on we will show how grey-scale* pictures can also be handled by the method. The method as presented assumes that pictures are black and white points arranged on a square lattice of N^2 points. Figure 1 is an example of such a picture on a 32×32 square lattice. We have chosen the square lattice because almost all mechanical or electronic picture scanning devices of which we are aware use just such a lattice and other research publications in this area deal exclusively with these "matrix pictures" as we shall call them. Although our scientific traditions are closely tied to the square grid and we have learned to think in its terms, there are some rather deep and fundamental reasons, argued most eloquently by Buckminster Fuller,[3] for using instead a triangular lattice as shown in Fig. 40. The merits of the triangular lattice in relation to our method are discussed more fully in Section XVII.

* Picture values range over an ordered finite set, such as [0, 1, 2, 3].

III. MAJOR CHARACTERISTICS

Several properties of our method are worthy of comment even before its full explanation. The "structural description" we employ consists of a set of simple polygonal closed curves whose vertices are ordered in either clockwise or counter-clockwise sense. These curves represent the continuous boundaries between connected sets of black points which we call "objects" and connected sets of white points which we call "holes." Figure 2 depicts the structural description of the pattern in Fig. 1. Furthermore, our structure contains information indicating which curves are immediately inside a given curve. This information is a directed tree as shown in Fig. 3. The points of a curve are ordered so that the curve is traced keeping black points to the right and white to the left. As a result, curves which define the outside boundary of black areas proceed clockwise and vice versa for white areas. From this "structural description" the original "matrix picture" can be unambiguously reconstructed so that no information has been lost by the transformation.

Certain familiar numerical properties of objects such as perimeter, area and degree of multiple-connectivity are readily calculated, as are other less well-known but just as useful properties such as oblongness, stringiness and wiggliness, which are all defined in later sections.

The most important advantage of this particular "structural description" lies in the simple fact that the sequential trace of the boundary of an object contains the most useful data for recognizing objects one from another in a great many applications. To cite one example, the curve defining the outside boundary of the letter "capital A" will have certain properties that are invariant over several fonts and would hardly be confused with the corresponding boundary curve for the letter "capital F." It is our opinion that discrimination between objects based on the trace of their bounding edges will encounter difficulty precisely where human discrimination* begins to have trouble. Thus we can safely follow our intuition in constructing algorithms for particular types of recognition.

* unaided by contextual information.

The number of bits required to encode structural descriptions is dependent on the picture itself; in fact, the relative size of two separate picture descriptions frequently approximates a crude intuitive measure of relative complexity. More important yet is the fact that picture descriptions require less than double the original number of bits when the resolution is doubled, as opposed to quadruple in the matrix mode of representation. For example, Figs. 5 and 6 show the results of hand digitizing the simple pattern of Fig. 4 at two resolutions, the first requiring 40 points and the second only 72 although the resolution is double. This linear increase in the amount of data is a simple consequence of the one-dimensionality of edges compared to areas. As a result of the linearity of this increase it may be practical to digitize a picture at enough resolution to capture the finest details even though the degree of detail of a large portion of the picture does not warrant the finer resolution.

As a final point we mention that the creation of the "structural description" can be accomplished efficiently by a combination of hardware and software. The hardware is merely required to digitize a picture and record those points where a boundary edge changes direction. We shall refer to these as "curvaturepoints." A program has been written to accept curvaturepoints as they might be generated by a scanning mechanism and to link them together properly into the closed curves of our "structural description." The simplicity of this program is as surprising as it is crucial.

IV. PREPROCESSING

The reader may already be wondering how we handle the problem of noise. Our approach to this problem is approximately as follows:

An input picture is first digitized into the binary matrix format. Next the matrix picture is preprocessed employing local averaging such as described by Dinneen[4] and gap filling. In a recent paper Narasimhan[5] points out that localized gaps and spurious extra points may be fairly successfully deleted by local preprocessing but that "large gaps which destroy basic connectivity in a picture can effectively be removed only by the use of the implicit syntax of the patterns." If the digitizing hardware is to perform the local preprocessing (as we feel it should), then the design should be flexible enough to satisfy a reasonably wide class of different preprocessing algorithms. The experimental research on preprocessing carried on at the Department of Computer Science of the University of Illinois, particularly the report by Yamada and Fornango,[6] leads us to be confident that an extremely general purpose preprocessing algorithm can be constructed from simple formulas applied to (3 × 3) windows* along with a handful of parameters to provide flexibility. See Figs. 7, 8, 9 and 10.

Only after local preprocessing and gap filling have been accomplished does our method take over; the "structural description" sees, so to speak, exactly what is presented to it and hence any spuriosities must be deleted before the description is constructed. The "large gaps" referred to by Narasimhan must be bridged by the recognition software which can be made to reflect the syntax on a more global level. This separation between local gap filling and not-so-local gap filling has an analog in human visual recognition. Some gaps in printed characters are small enough so that by squinting or backing away we no longer perceive the gap; on the other hand, some spurious gaps in characters remain no matter how hard we squint and their spuriosity can only be identified in relation to an implicit syntax.

In the remainder of this paper we shall assume that input pictures have been suitably preprocessed and the resulting idealization of the original

* any (3 × 3) submatrix of the matrix picture.

unprocessed picture contains what we intuitively consider to be the essential content of the picture. It is our opinion that "Preprocessing" is a large enough problem to warrant separate investigation and that it is natural to separate it from the "Description" and "Recognition" phases of pattern recognition as much as possible.

V. CONTOURING

The closed curves of the "structural description" introduced in Section III are, in fact, contour lines for a continuous function $F(x,y)$ defined over the square of the "matrix picture" and so that $F(i,j) = P_{ij}$, where $1 \leq i \leq N$, $1 \leq j \leq N$ and P_{ij} (= 0 or 1), is the value of the picture point at the (i,j) position in the "matrix picture". We shall define the function F as follows:

Consider the subdivision of the picture area into smaller squares defined by the square matrix of picture points by drawing horizontal and vertical lines through the points of the picture. It should be clear that a triangulation* of the picture area can now be effected by separately triangulating each smaller square, as shown in Fig. 11. We shall describe rules for determining which diagonal will be added to each square to complete the triangulation. These rules, which depend strictly on the four picture values at the corners of the square, may seem strange and arbitrary at this point but will be justified later on.

There are only six essentially different combinations of values possible for the corners of a square. These are shown in Fig. 11 along with the proper diagonal. Where no diagonal is shown it means that either one may be chosen.

Now that the picture area has been triangulated, we define our function F separately for each triangle. Supposing some triangle to have vertices A, B and C, we let P_A, P_B, P_C denote the picture values at points A, B, C which are indeed points of the "matrix picture." There is a unique linear function $F^*(x,y) = ax + by + c$ which takes on values P_A, P_B, P_C at points A, B, C. Each triangle of the triangulation determines uniquely an F^*_{ABC} and F is defined so that $F(x,y) = F^*_{ABC}(x,y)$ whenever (x,y) is inside or on triangle ABC.

The proofs that F is well-defined and continuous are easily derived from the exemplary behavior of linear functions. In fact, the only way discontinuity could possibly occur is by way of F being doubly defined for some point on a common edge $\overline{AB}$ between two triangles Δ_1 and Δ_2. The reason that this cannot occur is because F_1^* and F_2^* (the linear functions for Δ_1 and Δ_2) when restricted to the edge $\overline{AB}$ are both found to vary linearly between A and B and to have the same values at these two points; hence $F_1^* = F_2^*$ on the edge $\overline{AB}$ and F is well-defined.

---

*Subdivision of plane area into disjoint triangles.

Having defined F over the picture area as a continuous piecewise-linear approximation to the original picture, we can simply remark that the "structural description" of the picture is the set of contour lines of F drawn at a height of 1/2. This approach certainly corresponds well with any intuitive notion of edges in a binary picture.

VI. MESH TRIANGULATION

In deciding how to triangulate the square mesh we follow two simple rules. If there is a choice of whether to connect two black points or two white points, choose the black and try to have as few curvaturepoints as possible. The triangulation rule embodied in (d) of Fig. 11 assures that black points have precedence, while rules (b) and (c) prevent a straight edge at 45° from being encoded as a sawtooth contour line. The reader may verify these statements by choosing the other diagonal and then constructing the contour. The actual contour for the top pattern in Fig. 11 is presented at the bottom of Fig. 11 and hatch marks have been appended to show which side of a contour line is the high or black side.

The need to establish rules regarding the triangulation, although not a serious handicap, is nevertheless an annoyance from the point of view of aesthetics and symmetry. The "rhombic" or triangular grid, being already triangulated, needs no arbitrary rules and has the very pleasing property that the local neighborhood of a point has complete symmetry. As it turns out, this symmetry can be translated into ease and efficiency of programming as well as allowing a simpler, more efficient encoding of the contours.

VII. CURVATUREPOINTS

As mentioned earlier, a "curvaturepoint" is a point where one of the contour lines changes direction. For example, in the bottom of Fig. 11 the numbered points 1 - 10 are the curvaturepoints whereas E and F are points on the contour but not at bends. This is because the two sections of contour line which meet at E are both in the same direction.* Referring to Fig. 12, we see that at E there is an incoming edge and an outgoing edge both in direction 5, whereas at point 8 there is an incoming edge in direction 6 and an outgoing edge in direction 5.

It turns out that curvaturepoints can only occur at points midway between two picture points. Furthermore, the question of whether or not a given point is a curvaturepoint can be determined by referring to the values of just 6 picture points arranged in a (2 × 3) or (3 × 2) array. Figure 12 depicts how curvaturepoint 4 of Fig. 11 is determined by the (2 × 3) window shown. If P_{ij} denotes the value of picture point (i, j), then $(P_{33} = 0,\ P_{23} = 1$ and $P_{34} = 1)$ implies that there is an incoming edge in direction 3; furthermore $(P_{33} = 0,\ P_{23} = 1$ and $P_{32} = 1)$ implies an outgoing edge in direction 5. As a result, we record that the fourth curvaturepoint is at i = 2.5, j = 3 with $\theta_1 = 3$ and $\theta_2 = 5$.†

If D1 - D6 denote the picture point values as shown in Fig. 12, then the following hold for the point midway between D2 and D5:

First of all, the point is an edgepoint if and only if D2 and D5 are different in value. If not, it is useless to proceed further with this point. The values EDGEIN and EDGEOUT are determined by the following logic formulas, which are easily implemented by digital hardware:

$$D5 = 1,\ D1 = D2 = D4 = 0 \Rightarrow \text{EDGEIN} = 1$$
$$D2 = D6 = 1,\ D5 = 0 \Rightarrow \text{EDGEIN} = 3$$
$$D2 = D3 = 1,\ D5 = D6 = 0 \Rightarrow \text{EDGEIN} = 4$$
$$D2 = 1,\ D3 = D5 = D6 = 0 \Rightarrow \text{EDGEIN} = 5$$
$$D1 = D5 = 1,\ D2 = 0 \Rightarrow \text{EDGEIN} = 7$$
$$D4 = D5 = 1,\ D1 = D2 = 0 \Rightarrow \text{EDGEIN} = 0$$

*Contours are directed so that the hatch mark is on the right.
†θ_1 is the direction of the incoming edge and θ_2 the outgoing.

$D3 = D5 = 1,\ D2 = 0 \quad \Rightarrow \text{EDGEOUT} = 1$

$D2 = 1,\ D1 = D4 = D5 = 0 \quad \Rightarrow \text{EDGEOUT} = 3$

$D1 = D2 = 1,\ D4 = D5 = 0 \quad \Rightarrow \text{EDGEOUT} = 4$

$D2 = D4 = 1,\ D5 = 0 \quad \Rightarrow \text{EDGEOUT} = 5$

$D5 = 1,\ D2 = D3 = D6 = 0 \quad \Rightarrow \text{EDGEOUT} = 7$

$D5 = D6 = 1,\ D2 = D3 = 0 \quad \Rightarrow \text{EDGEOUT} = 0$

The determination of which edgepoints are curvaturepoints is then simply a matter of whether or not EDGEIN $\neq$ EDGEOUT. It should be fairly clear from the simplicity of the above conditions that the digital logic involved is easily implemented. We have written an ALGOL 60 procedure, CURVATURE-POINTS, to perform the same function via software, and this is reproduced in the Appendix.

VIII. LINKAGE

So far we have described how an input "matrix picture" can be transformed into a set of "curvaturepoints" of its approximate contour lines. It is still not entirely obvious that these separate and sometimes widely scattered points can be linked together in the proper way to reconstruct the polygonal curves of its "structural description." The reader may have noticed that in an earlier section we spoke of the "structural description" as consisting of closed curves, whereas it is perfectly possible for contour lines to end at the edge of a map or in our case, a picture. To eliminate any chance of confusion, we shall assume that the picture points on the boundary of an input picture are all zero. This will force all contours to be closed curves but will not really hamper the validity of our approach. The only reason we make this assumption is to simplify the discussion below; in practice there is no need to border a picture with zeros this way, but the linkage of curvaturepoints is somewhat more complicated otherwise.

We shall assume that all the curvaturepoints of a given input picture are delivered to a general purpose digital computer in the order in which they are encountered during a normal left-to-right and top-to-bottom raster scan. The rows will be designated by increasing values of Y and the column positions within each row by increasing values of X. Also, the incoming and outgoing edge directions θ_1 and θ_2 are considered an integral part of any curvaturepoint. For example, the simple pattern of Fig. 11 would be represented as:

	Y	X	θ_1	θ_2
(1)	1.5	3.0	1	0
(2)	1.5	4.0	0	7
(3)	2.0	4.5	7	6
(4)	2.5	3.0	3	5
(5)	3.0	1.5	3	1
(6)	3.0	2.5	5	7
(7)	3.0	3.5	1	3
(8)	3.0	4.5	6	5
(9)	3.5	3.0	7	1
(10)	4.5	3.0	5	3

A simple inspection of the above table shows the following interesting facts about the two points 1 and 5. First of all, $\theta_1[1] = \theta_2[5] = 1$ and $X[1] + Y[1] = X[5] + Y[5] = 4.5$; furthermore, there is no curvaturepoint P between* these two which satisfies the condition $X[P] + Y[P] = 4.5$. It should be clear that whenever two points P and Q ($P < Q$) have the same value of X + Y, then they lie on the same line drawn in direction 1 - 5 (see Fig. 12). If, in addition, there is no intermediate curvaturepoint and either $\theta_1[P] = \theta_2[Q] = 1$ or $\theta_2[P] = \theta_1[Q] = 5$, *then the two points are adjacent curvaturepoints of the* same contour curve and hence should be linked together. There are similar conditions for the linkage of points with the same (X - Y), X or Y values. Each case is associated with a pair of directions (respectively 3 - 7, 2 - 6 or 0 - 4) and the precise conditions are geometrically self-evident. These conditions are all that are needed to perform the linkage of the curvaturepoints into simple closed polygonal curves. The implementation of this procedure uses very strongly the fact that the points are ordered by increasing X within increasing Y since there is a great deal of geometric information implicit in this sorting order. The ALGOL 60 procedure LINKAGE is reproduced in the Appendix.

* meaning a point with a number between 1 and 5.

IX. SMOOTHING INFLECTIONS

In Fig. 13, we have redrawn the outermost curve of the "structural description" in Fig. 2. Some of the curvaturepoints are not solid dots and these are to be deleted from the description. The criteria which control which points will remain are explained below, along with an intuitive justification for performing this smoothing operation.

An "inflection" will be defined as two adjacent curvaturepoints, one with a bend* of -1, the other +1, and such that the distance between the two points is 1 or $\sqrt{2}$. Referring to Fig. 13, the pair of points (3, 4) are at distance 1 from each other and have bends respectively +1 and -1 as shown in the insert. As a result, the incoming edge at 3 and the outgoing edge at 4 are in the same direction, namely 7. Similarly the pair (6, 8) are at distance $\sqrt{2}$ with bends -1 and +1; hence, we have another "inflection".

When points 3, 4, 6 and 8 have been deleted, the sequence of curvaturepoints becomes 1, 5, 9, ... and point 5 retains a bend value of +1. Since we allow point deletions to occur only in pairs with bends that cancel (+1 and -1), it is clear that the total bend around a curve is not disturbed at all. The bend at point 5 in Fig. 14 is not exactly +1 as can readily be seen, but as an approximation it does not really go far wrong. Furthermore, if more precision is required, it can be calculated from the X and Y coordinate values of the 3 points 1, 5 and 9.

The main advantages of performing this smoothing of inflections are the reduction in the amount of data to be processed in subsequent discrimination algorithms and the elimination of some sequences of edge bends which we may call "spurious wiggles" since the corresponding edge in a typical original continuous pattern was probably straight. Figure 13 shows a "spurious wiggle" of 4 pairs of points. In the example depicted the reduction in data is substantial, amounting to 40%, although the picture pattern contains a great deal of real detail.

*The bend at any curvaturepoint measures the change in direction of the edge at that point in units of 45°. Bends toward the right are negative and toward the left, positive.

As a final point we want to make it perfectly clear that this smoothing procedure is not an absolutely necessary part of the overall method. Depending on the particular application, it may or may not be appropriate. We present it here in place of several other smoothing algorithms which we have experimented with because it has been found to accomplish a substantial reduction in data without eliminating many major details and while requiring only simple decision criteria.

X. MERGING CURVATUREPOINTS

When the sequence of curvaturepoints describing a single closed contour has been constructed and the "inflections" have been deleted as described in the previous section, then a very compact but extremely characterizing signature can be computed by merging curvaturepoints according to the following rules:

Transform the closed curve into a sequence of integers representing amounts of bend by adding together the values of bend for two adjacent curvaturepoints of like sign whenever their distance apart is less than some threshold which is appropriate to a given application.

Figure 15 depicts how the grouping of like points would be made for our sample curve of the last section, assuming a threshold distance of approximately 4.0. The compact signature for this curve is

(-4, +1, -4, +4, -1, -3, +5, -5, +1, -2, +2, -4, +2, +1, -2, +2, -3, +2)

Once again we must emphasize that special applications will generally suggest variations in the merging of bends in addition to guiding the determination of the distance threshold. Some applications may, of course, require the retention of more detailed edge length information in which case this simple scheme would not be appropriate.

XI. FORMAL SYNTAX

In this section we shall demonstrate how formal syntax can be used very naturally to construct languages in which polygonal curves can be described and classified. We must warn the reader that a curve is not quite equivalent to a finite sequence because the latter has a definite beginning and end whereas the polygonal curve is properly a cyclic ordering of curvaturepoints. As it turns out, however, the Backus Normal Form (BNF) can be used in the same way as for strictly linear languages except that any parsing mechanism must realize it is parsing a cycle and not a linear string.

To show a concrete example of the use of formal BNF syntax we have chosen the picture in Fig. 16, which depicts a human chromosome and was borrowed from an article by Ledley and Ruddle[7] in which syntax directed pattern recognition is discussed. A comparison of our methods and those of Ledley and Ruddle is contained in a later section where other related research is also discussed.

The bottom of Fig. 16 depicts the digitized chromosome as it exists after the "inflections" have been deleted and like bends have been grouped together. The first column of Fig. 17 represents an ordered list of the bend groups and edge lengths of the curve. Every line represents either an edge or a bend and they occur alternately; for example, -4,9 represents a bend group totalling -4 (180° clockwise), the edge length around the bend being 9, as can be seen from Fig. 16 at the top of the chromosome.

The second column of Fig. 17 has been constructed by altering the notation of the first column slightly and calculating a measure of the rate of bend for bend groups. For example -4,9 has been changed to -4 [2.25]/. This is done by copying the bend amount -4 just as is, then dividing the length 9 by the absolute value 4 of the bend and enclosing the result 2.25 in square brackets. The slash is added to formally separate the bends and edges.

The third and last column of Fig. 17 represents the final transformation of the curve description which is accomplished by replacing the signs of bends by the words CONVEX and CONCAVE in an obvious way, by classifying the

rates of bend into three categories, ABRUPT, NORMAL and GRADUAL by the rules:

(0 through 1.0)	→	ABRUPT
(1.0 through 3.0)	→	NORMAL
(3.0 and up)	→	GRADUAL,

by categorizing edge lengths by the rules:

(0 through 3.5)	→	SHORT
(3.5 through 6.5)	→	MEDIUM
(6.5 through 9.5)	→	LONG
(9.5 and up)	→	EXTRA-LONG,

and by renaming bend amounts by the rules:

4	→	END
3	→	WEDGE
2	→	RIGHT
1	→	SLIGHT.

The rules used to produce the curve description of column three are heavily parameterized. This provides the flexibility which is required for the method to be generally useful.

Finally, the formal BNF syntax to describe chromosomes similar to the one depicted (and including this one) is contained in Fig. 18. A stylized version of the chromosomes described is drawn at the bottom of the figure along with the pattern elements called ⟨short vee⟩ and ⟨long parallels⟩.

XII. NUMERICAL SIGNATURES

As we said earlier, the "structural description" permits the calculation of the area, perimeter and degree of multiple connectivity of any connected set of like points in the picture, "objects" or "holes". In the cases where an object or hole has multiple connectivity, the area is equal to the area enclosed by the outer curve minus the sum of the areas enclosed by the inner curves; the perimeter is the sum of the perimeters of the outer and all inner curves. The "connexity" (degree of connectivity) is simply the number of boundary curves, counting the outer curve as well as all inner curves.

As a result of investigating the properties of objects which are most useful in our own discrimination, we have found the three properties "oblongness," "stringiness" and "curvaturerate" to be extremely characterizing. The first of these is defined as the largest ratio between length and width for all possible circumscribed rectangles. As such it could hardly be computed from our "description," but a good approximation is given by calculating the ratio length/width for two circumscribed rectangles, one having edges parallel to the axes of the picture matrix and the other parallel to the two lines at 45° to the axes.

The second property, "stringiness," is defined as (2 × AREA)/PERIMETER and can be used to test whether or not the "object" or "hole" is composed of linelike parts. This is because any object made out of lines which are approximately W units wide will satisfy the approximation W ≐ STRINGINESS = (2 × AREA)/PERIMETER. For objects with small AREA, this approximation does not work very well, but otherwise it is extremely accurate.

The third property, "curvaturerate," is simply the number of curvaturepoints* divided by PERIMETER. To be a faithful value, the number of curvaturepoints should be calculated after the "inflections" have been deleted.

Other properties easily derived from the "structural description" and conceivably very useful are "number of concavities," "number of convexities," "average concave bend," "average convex bend," "maximum edge length," etc.

Another property, which would most appropriately be named "wiggliness," could be defined as the number of occurrences of two adjacent curvaturepoints

*weighted by the absolute value of the bend at each point.

of unlike bend (one positive and one negative) after the spurious "inflections" have been deleted. As in the case of "curvaturerate," this number can be normalized by dividing by PERIMETER if it is more useful to do so.

We fully intend to continue experimenting with these "signatures" to gain deeper insight into which are more fundamental in the sense of being useful for a larger range of discrimination and recognition problems.

As an example of the usefulness of some of the above properties we exhibit Fig. 19. In Fig. 20 we have compiled a table of numerical signatures for the seven objects in Fig. 19.

In spite of the fact that these signatures are readily calculated and extremely useful, we feel compelled to point out that the sequential trace of the boundary curves is still the most useful format for the information in most picture patterns.

XIII. AREA VISIT SIGNATURES

In a recent paper Brown[9] reports on a clever method for on-line recognition of hand-printed characters. The key to the method lies in the fact that the time sequence of the production of separate strokes of a printed character is very valuable information. In the Brown method, the square area in which each character is printed is subdivided into seven subareas, as shown in Fig. 21. The strokes of the character being made are encoded into a "trace" with 0 representing the fact that the stylus has been raised from contact with the character area. Hence "A" printed in three strokes as shown in Fig. 21 is encoded with three zeros separating the strokes 641, 643, and 4. The reader may verify that the trace (14643040) is one fairly common variation of printing "A".

We have adapted the above idea to provide a compact signature of any "structural description," one which retains the flavor of the sequential curve trace. First of all, we choose to subdivide a square area into nine subareas as shown in Fig. 22. They will be labeled with the letters A, B, C, D, E, F, G, H, K as shown. To provide an analog to the time sequence which is exploited so nicely by Brown, we have our ordered sequence of curvaturepoints for each closed curve of a black area. By replacing each such point with the appropriate label (one of the 9 letters) and eliminating adjacent occurrences of the same label, we finally arrive at a rather compact but revealing signature. In the example of Fig. 22, the outer curve of the character "A" has been reduced to the simple "area visit" signature (BKEG).

To allow for expected variations in printing, we would accept other signatures as emanating from an "A" also; for example, (CKFEG) or (BFKFDGD) or (AKEDG) should be recognized as "A".

It should be clear that the above idea is very flexible since the square area need not really be square at all, the area (whatever it is) may be positioned relative to the visual field or relative to the character in question, and the subdivisions are also arbitrary.

It may be of some interest to know that formal BNF syntax can also be used in this context. For example, we might use the following to create a

reasonable definition of "A" which would include the variants mentioned above:

⟨left crossbar⟩ ← E | ED | FE | FD

⟨long up-right⟩ ← GA | GB | HB | HC | GDB | HEC

⟨long down-right⟩ ← AH | BH | BK | CK | BFK | AEH |

⟨short down-left⟩ ← DG | EG | EH | FH

⟨short up-left⟩ ← KF | KE | HE | HD

⟨capital A⟩ ← ⟨long down-right⟩ ⟨short up-left⟩ ⟨left crossbar⟩ ⟨short down-left⟩ ⟨long up-right⟩

If we parse (BFKFDGD), we get BFK, KF, FD, DG, GDB, which collapses to ⟨capital A⟩ using the above rules. According to these rules, the trace (BHEHEC) is a valid syntactical ⟨capital A⟩ . The reason we need not worry is that it cannot occur because the curve would not be closed and it would cross itself.

Another syntax for the same character can be based on the labels themselves rather than pairs and triples of adjacent labels. In fact, it is much simpler and may produce more efficient parsing:

⟨top⟩ ← A | B | C

⟨bottom⟩ ← G | H | K

⟨right middle⟩ ← E | F

⟨left middle⟩ ← D | E

⟨middle⟩ ← D | E | F

⟨crossbar⟩ ← ⟨middle⟩ | ⟨right middle⟩ ⟨left middle⟩

⟨capital A⟩ ← ⟨top⟩ ⟨bottom⟩ ⟨crossbar⟩ ⟨bottom⟩

With these rules, such patent nonsense as (AGFEK) is syntactically correct, but there is no problem since any valid simple* closed curve which parses to ⟨capital A⟩ must be reasonably like the outer boundary of "A".

Finally, we mention the fact that by using only selected types of curvature-points (convex ends, for example) in the trace, one obtains a different characterization which might be useful.

---

*intersects itself only once where it closes.

XIV. FIVE-FOUR THEORY

The research reported in this paper grew out of a fairly simple discovery made by the author in 1965. In thinking about simple characterizations of points on or near the edge of an object in a binary matrix pattern, we considered looking at 3×3 windows centered at given points. Since edges are boundaries between black and white areas, it seemed perfectly reasonable to look more closely at those points whose 3×3 window was almost evenly divided between black and white points, specifically 5 black and 4 white or vice versa. It came as no surprise to us that all these so-called "5-4 points" were indeed along edges of the objects on which we tried the idea. What did come as a surprise was the fact that all 5-4 points were at places where the edges were changing direction, whereas points along straight sections of edge were points with a 6-3 split. In Fig 23. we have shown the 5-4 points for the chromosome pattern used in an earlier section; we have also indicated two "6-3 points" to indicate how they occur along straight portions of boundary edges.

Figure 24 shows the polygonal curve drawn through the 5-4 points in Fig. 23; it is clear that this is an approximation of the actual picture edge to an accuracy of one mesh unit.* The elimination of points where the edge does not bend implies a reduction of data which we found very pleasing. The inclusion of pairs of 5-4 points at places where the edges have inflections of the sort described earlier is unavoidable because it is essentially "quantization noise," a necessary concomitant to the process of reducing a continuous picture to a discrete matrix.

Unfortunately, there are several important drawbacks to the use of the 5-4 points. Although most 5-4 points belong to one of the six types shown at the top of Fig. 25, if the pattern contains connected sets of black points which are thin, then irregular 5-4 points can and do occur as shown in the middle section of Fig. 25. What is even more deadly is the fact that some patterns with a great deal of curvature may be missed entirely if they are too thin, such as the thin line at the bottom of Fig. 25.

Another problem is that of determining how to link the 5-4 points together properly to obtain the correct edge polygons for the pattern. We managed to solve

*The distance between two adjacent points of the matrix pattern.

this problem in a fairly adequate way for patterns whose 5-4 points are all of the common variety. By performing the calculations given by the formulas at the top of Fig. 26 for each 3×3 window and then computing σ_1 (DX, DY) by the following table, we are able to determine the direction θ_1 of the incoming edge and the value $\Delta\theta$ of the bend according to the formulas below:

$$\sigma_1(3, -1) = 2 \qquad \sigma_1(3, 0) = 2$$
$$\sigma_1(1, -3) = 3 \qquad \sigma_1(2, -2) = 3$$
$$\sigma_1(-1, -3) = 4 \qquad \sigma_1(0, -3) = 4$$
$$\sigma_1(-3, -1) = 5 \qquad \sigma_1(-2, -2) = 5$$
$$\sigma_1(-3, 1) = 6 \qquad \sigma_1(-3, 0) = 6$$
$$\sigma_1(-1, 3) = 7 \qquad \sigma_1(-2, 2) = 7$$
$$\sigma_1(1, 3) = 0 \qquad \sigma_1(0, 3) = 0$$
$$\sigma_1(3, 1) = 1 \qquad \sigma_1(2, 2) = 1$$

$$|\Delta\theta| = \begin{cases} 1 & \text{if SUM + CENTER} = 5 \\ 2 & \text{otherwise} \end{cases}$$

$$\text{SIGN}(\Delta\theta) = \begin{cases} -1 & \text{if CENTER} = 1 \\ +1 & \text{otherwise} \end{cases}$$

$$\Delta\theta = \text{SIGN}(\Delta\theta) \cdot |\Delta\theta|$$

$$\theta_1 = \begin{cases} \sigma_1 + 1 \pmod 8 & \text{if } \Delta\theta = -1 \text{ or } \Delta\theta = -2 \\ \sigma_1 & \text{if } \Delta\theta = +1 \\ \sigma_1 - 1 \pmod 8 & \text{if } \Delta\theta = +2 \end{cases}$$

A third difficulty of the 5-4 theory is that when the edges bend sharply (135°), the bend occurs at a 6-3 point which does not figure in the theory. This means that even assuming no thin lines in a picture, we still may lose some edge bends by recording only 5-4 points. Of course, we could record the 6-3 points which are sharp bends, but already the simplicity of the 5-4 idea is slipping away. It was because of these dissatisfactions that we looked for a similar but more rigorous contouring scheme.

XV. RELATED RESEARCH

A brief attempt will be made in this section to mention picture processing research which we feel is closely related to the method we have described here. The order in which we introduce the various research efforts has only minimal significance.

The first research effort we shall treat is the work being done on the Illiac III computer at the University of Illinois under the direction of Professor B. H. McCormick. The Illiac III is a processor of visual information and is described in Ref. 10 and Ref. 11. The hardware has the ability to store and operate upon 32×32 bit matrices or "matrix pictures." In Fig. 27 we have reproduced a diagram from Ref. 10 which indicates the stages of pattern articulation as these transformations are carried out in the Illiac III. The derived graph at the bottom should seem a familiar description to the reader in view of the previous sections. So that there should be no mistake, we have drawn in Figs. 28-30 the analogous steps in our method for the same picture pattern. We feel it is important that our technique recognizes solid figures as well as line figures and does not require the idealization to a line figure. As far as we can tell from the literature references to IlliacIII, the pattern articulation depends on the pattern being linearized, and presumably the two uppermost patterns of Fig. 27 would not be recognized as different in the Illiac scheme. Of course, they are easily recognizable in our method because the outer edge of the "hole" is contoured as well as the outer edge of the black "object."

The papers by Yamada and Fornango[6] and Narasimhan and Fornango[12] are concerned with preprocessing to obtain "matrix pictures" suitable for description and recognition. The first of these is similar in spirit to Dinneen,[4] while the latter demonstrates how preprocessing at a local level can be used successfully to reduce a grey-scale photograph of a face to a black-white pattern which retains the more important features. More about this paper will be said in the next section.

The two papers by Narasimhan[2,5] referred to earlier contain the description of a class of "labelling algorithms" which are employed to recognize the important properties of patterns consisting of linelike parts. These algorithms are clearly constructed with a parallel pattern processing capability in mind, but they might be extremely useful as part of the preprocessing hardware we spoke of in Section IV. Finally, we must point out that Ref. 5 is mainly concerned with the possibility of

describing classes of pictures by using formal syntax. The main difference between the Narasimhan approach and our own is that he considers points of the picture as primitives, whereas we feel very strongly that the edges between black areas and white areas could be regarded as primitive for some picture classes.

We know we have not done justice to the Illiac effort in this brief resumé but hope the reader has obtained a general feeling for how that research relates to ours.

Another group, which has been working in the area of picture processing for more than ten years, is the one at the National Bureau of Standards in Washington, D. C., under the direction of R. Kirsch. In their initial paper Kirsch, et al.[1] report on experiments with picture processing in a digital computer. Useful numerical values extracted from two-dimensional patterns were area, compact encoding, number of blobs,* center of gravity and a "first derivative" which amounts to retaining only the boundary of an object and is identical with the first stage of Illiac III pattern articulation as depicted in Fig. 27. We feel our methods allow for the calculation of these quantities (except center-of-gravity) in a very efficient manner. We could calculate the center-of-gravity of the curvaturepoints and this might be as useful as the object center-of-gravity.

As far as we can tell, Ref. 1 contains the first explicit mention of the possibility of using linguistic syntax in the description and discussion of picture patterns, although the idea of syntax-directed pattern recognition has since been further developed by Narasimhan[2] and Ledley and Ruddle.[7] Kirsch[13] has also developed this idea to the extent of defining the formal syntax of simple subsets of English capable of expressing the existence and interrelationships of picture elements. In this same paper it is suggested that the simplest way to choose primitive elements in pictures is to choose the smallest parts which are easily recognizable. It is for just such a reason that we have suggested that portions of edge could be considered as primitive in a formal syntax for picture patterns.

The third research effort we will describe is the human chromosome analysis reported by Ledley and Ruddle.[7] The aim of that work was quite similar to the aim of this paper in that they reduced a "matrix picture" to a cyclic description of edge curvature. In this cyclic description portions of edge are labelled

* Contiguous black areas

with one of 13 possible labels, four of which are as follows:

E – clockwise curve,

O – fairly straight segment,

V – slight counterclockwise curve,

Y – sharp counterclockwise curve.

In Fig. 31 we have copied two sample chromosomes from Ref. 7 to show how an object might be encoded. The similarity between this and our own formal syntax is no accident,since we were aware of this work before our method was entirely worked out.

Once again, the advantage of our procedure is that it obtains the cyclic description much more efficiently than the Ledley and Ruddle programs in which a search through the picture matrix must be made to follow the edge.

Next we discuss briefly the classic paper of Grimsdale, et al.[14] In this paper a system is described to reduce line figures to a description of various parts and the way they fit together to make up the figure. The method uses second differencing techniques to determine edge curvatures after some averaging has been done to smooth edges. The method has the advantage of being successful even when the pattern has spurious gaps of small size.

We feel that our method, coupled with appropriate preprocessing, will turn out to be more generally usable than the Grimsdale, et al. method as well as being more direct and efficient, but as yet we have no experimental data to document our bias. The reader familiar with Ref. 14 will probably realize that our method of getting at the sequential edge description is very direct and easily performed, whereas a moderate amount of struggle is required in the Grimsdale procedures.

The research probably most closely related to our own is that reported by Prather and Uhr.[15] They propose to recognize two-dimensional black-white patterns in two steps; the first step is embodied in a computer program which transforms the input pattern into a sequential description of its outer edge, while the second step involves a computer program to accept "edge descriptions" and learn how to recognize different characters on the basis of accumulated experience. They call the first program the "preprocessor" and the second the "recognizer," but we feel "describer" is more apt for the former. The problem of noise is not dealt with and would presumably be handled by a preprocessor before the description and recognition phases begin.

The pattern description program of Prather and Uhr converts an input pattern 20×20 into a 60×60 matrix by replacing each element of the original matrix by a 3×3 submatrix, all nine of whose values are identical to the original one. Then a two-cell wide border of zeros is added, making the array 64×64. Next the pattern is thickened by making a cell "1" if any of its eight neighbors is a "1." Finally, all cells which are centered on a 3×3 window with all nine cells black are erased. This leaves a black pattern representing the edges of the original pattern. This set of points is followed by the program and reduced to a sequence of vectors, each vector being in one of the cardinal directions N, E, S, W. Then this vector sequence is processed to smooth out irregularities due to the "quantization." The reader is referred to Ref. 15 for the details. Suffice it to say that the programmed edge-following required, and the unnecessary blowup in data volume by a factor of nine, suggest the use of our procedures in place of the Prather-Uhr "preprocessor." Since they have chosen to separate the two tasks of description and learned recognition, we feel that by fairly simple adjustments their learning program could handle our "structural descriptions" thus providing a more efficient system.

As a finale we would like to discuss two closely related efforts, the research of Selfridge[16] and Dinneen[4] at MIT in 1955, and that of Bomba[17] at Bell Labs in 1959.

The research of Selfridge and Dinneen involves experiments performed on black-white digitized alphanumeric characters, in particular the effects of transforming picture patterns via local operations of two kinds: averaging/thresholding and measuring the degree of radial asymmetry at a point. In the paper by Dinneen examples are given to show how the averaging/thresholding has the effect of thickening or thinning, depending on a low or high threshold, respectively. They used a 5×5 window for this operation. It should be pointed out that "thickening" has the effect of gap filling and noise cleaning so necessary to any preprocessing mechanism.

The second operation employing a 7×7 window finds points in the pattern where a high degree of radial asymmetry is in evidence; for sample alphanumeric characters this operation extracts the major bends in the edges.

The paper by Bomba[17] four years later continues the experimentation with local operations to the point of discriminating a set of 36 alphanumeric characters based on features extracted using approximately 20 local operations. This research should prove valuable in guiding the construction of preprocessing algorithms.

In summary, we find in the published research a great deal of evidence to suggest the importance of major bends in the boundary curves of objects within

the picture. Some research* recognizes this explicitly, e.g., Refs. 7, 14 and 15, whereas others do so in a somewhat less obvious way, but it seems to crop up repeatedly in these papers. We regard our efforts embodied in this paper as constituting a direct and fairly uncomplicated procedure for extracting the edge bend information from a digitized pattern, and we have reason to hope that it can be efficiently implemented by a suitable hardware-software system.

---

*Since this paper was written the work of Unger,[26,27] Freeman[23,24,25] and Kuhl[28] has come to our attention. Their research is so closely related to "curvaturepoints" that omission of adequate reference would be unthinkable. In particular, the papers by Freeman dovetail beautifully with this paper because they describe an approximate encoding of an arbitrary continuous curve in the plane by a sequence of vectors, each in one of eight possible directions. Our "structural descriptions" are essentially Freeman encodings of closed curves. References 24 and 25 describe algorithms to calculate important functions of encoded curves such as length, width, height, symmetry about various axes, area, moments, and center of gravity. Other signatures useful for recognition are directionality spectrum, asymmetry spectrum, the difference sequence (edge bends) and a smoothed difference sequence in which high-frequency variations due to noise have been effectively filtered out.

Unger's research relates to pattern recognition using sequences of edge directions programmed on a hypothetical parallel processing "spatial computer."

The paper by Kuhl represents an attempt to wed the edge-sequence idea of Unger and the Freeman encoding. A binary pattern is processed by looking at a sequence of 3×3 windows in such a way as to trace out the outer boundary of a connected object, the trace being encoded in the Freeman way. While this approach is admittedly very similar to ours, there are important differences. Our definition of "curvaturepoint" depends on extremely local context and is hence more compatible with a sequential scanning/digitizing environment. Also the Kuhl approach produces only the outer boundary of an object, whereas ours allows any degree of multiple connectivity.

XVI. GREY-SCALE PICTURES

As promised earlier we shall briefly describe how our methods may be used in the context of grey-scale pictures such as the digitized chromosome of Fig. 32. Our first suggestion is quite trivial and simply involves converting the picture to a binary scale (black and white) using several different thresholds and contouring these pictures in the way described earlier. See Figs. 33, 34 and 35 for examples of contouring Fig. 32 at three levels.

It seems to us reasonable to use the low threshold (.5) to obtain rough indications of where the large blobs are located. Then these areas are further investigated using a medium threshold (2.5). Finally, we notice a "hole" in the blob of Fig. 34 and in order to check if this is really a "hole" we up the threshold to 4.5 and get the contours in Fig. 35. These show that the "hole" isn't really a hole at all but rather a deeper portion of an inlet between two arms of land. We do not feel *this multiple-level contouring tells the whole story about grey-scale pictures*, but it may be useful in contexts where "edgeness" is of paramount importance. The problem of recognizing textural differences in such pictures is admittedly more complicated, although we feel it may turn out to be amenable to a proper mix of local preprocessing and contouring.

A concrete example which has bolstered our confidence in the possible success of a preprocessing/contouring method is the experiment reported by Narasimhan and Fornango[12] in which they took an eight-level digitizing of a photograph of Lincoln and by suitable local preprocessing reduced it to a "black-white" sketch which retains most of the significant features of the original, as shown in Fig. 36.

As a final point we mention that the ideas of triangulation and approximate contouring developed earlier do not depend on the assumption of a discrete set of picture values. The simplicity of the algorithms vanishes, however, when the values are allowed more latitude.

XVII. TRIANGULAR GRID

In Section II we briefly alluded to an article by Buckminster Fuller[3] in which he argues in behalf of space-coordinate systems based on triangles and tetrahedra, claiming that these are the systems of space geometry which nature employs. The evidence he cites covers a broad range from crystalline structure to molecular biology and we have no intention to try a summary here. We feel that this section simply adds another argument to Fuller's already long list in support of triangular geometry.

To begin the discussion we refer the reader to Fig. 37 (borrowed from Ref. 3) which depicts microphotographs of the cornea of the eye in human and cow. The human eye in particular shows very clearly a hexagonal or "honeycomb" pattern like that in Fig. 39. Comparing the close-packed circles of Fig. 38 with the hexagonal packing of Fig. 39, it is easy to see the close relation they bear to one another; in fact, the microphotograph of the human eye seems to be intermediate between the two when examined carefully. It is reasonable to think of the close-packed circles as approaching a hexagonal pattern when the circles are squeezed more closely together under the assumption that the circles are semiflexible. A hexagonal packing is thus seen to be a rather natural result of nature's efficiency embodied in the mathematically optimal close-packing of circles.

If we now consider each hexagon in Fig. 39 as an area related to a light intensity receptor similar to the spot-size of electronic scanners and assign the light intensity value to the center of the hexagon, then we obtain the triangular grid of Fig. 40. The area associated with one of the picture points is shown dotted. The triangular grid is the exact analog of the more familiar square grid used to describe our method in earlier sections. We have thus encountered the first advantage of the triangular grid; namely, it is already triangulated and we can proceed with the contouring of each triangle, whereas in Section VI arbitrary rules were required to establish a reasonable triangulation.

Figure 41 shows part of a triangular grid along with the possible contour segments, three per triangle.* The possible curvaturepoints are, as before, the medians of the edges of the original triangles.

---

*We cannot prevent ourselves from pointing out nature's obvious lack of any anti-Semitic leaning. See "Star of David" in Fig. 41.

To determine curvaturepoints, we need only examine four-point rhombic windows as shown in Fig. 42; we must do so for all three possible angular orientations for such windows, of course.

The logic to determine if C is a curvaturepoint in Fig. 42 goes as follows: C is an edgepoint if, and only if, $P_1 \neq P_2$. In case it is, we have the following table of values for EDGEIN and EDGEOUT:

EDGEIN	EDGEOUT	LOGICAL CONDITION
$\overline{DC}$	$\overline{CE}$	$P_4 \,\&\, \neg P_1 \,\&\, \neg P_2 \,\&\, \neg P_3$
$\overline{AC}$	$\overline{CB}$	$\neg P_1 \,\&\, P_2 \,\&\, P_3 \,\&\, P_4$
$\overline{EC}$	$\overline{CD}$	$\neg P_4 \,\&\, P_1 \,\&\, P_2 \,\&\, P_3$
$\overline{BC}$	$\overline{CA}$	$P_1 \,\&\, \neg P_2 \,\&\, \neg P_3 \,\&\, \neg P_4$

These four possibilities cover all the ways in which C can be a curvaturepoint and the logic is extremely simple. Furthermore, all bends are either $\pm 60^{\circ}$, another simplification of the situation compared to the square grid. Also, all contour segments are of equal length which simplifies matters for later programs.

The reader may notice that once the incoming edge to a known curvaturepoint has been specified, the outgoing edge is uniquely determined since only two such edges are possible, and one results in no bend. This means that a polygon can be unambiguously reconstructed by remembering the initial point and direction (one of six possible) and thereafter just the length of each successive straight edge. In fact, to determine if the bends at the two ends of a given edge are opposite in sign, we need only check the parity of the edge length; an even length implies opposite bends and an odd length the same bend. The extreme compactness of the polygonal descriptions is likely to be valuable when storage considerations are of paramount importance.

Figure 43 shows our familiar chromosome as it might look on a triangular grid, and Fig. 44 shows in detail the contour and sequential description of a very simple pattern. A clearer understanding of some of the claims of the previous paragraph can be obtained by a careful study of Fig. 44.

Another advantage of triangular grid pictures is related to the fact that on most high-speed printers attached to digital computers the space between characters is related to the space between lines in a ratio 3/5. This is very annoying

to those of us who want to see an accurate representation of what we are working with. The ratio 3/5 can be traded for 5/6 by inserting blanks every other character on each line, but a 17% distortion* is still fairly aggravating. If we use the same printer to represent pictures on a triangular grid by the scheme shown in Fig. 45, then the distortion is only 4% since the height of each equilateral triangle should be $0.866 = \sqrt{3}/2$ times its base, whereas $5/6 = 0.833$.

As a final remark we would like to point out that building a triangular grid scanner can be accomplished by modifying presently used rectangular scanners. All that need be changed is the spacing of the lines relative to spot diameter and staggering the sample points on every other line. In fact, the character array in Fig. 45 indicates exactly how the scan would be accomplished.

---

*In representing pictures on a square grid.

XVIII. HALF-ADAPTIVE PERCEPTRONS AND PANDEMONIUM

Roberts[18] found that by structuring a perceptron to the extent that each cell in the first layer was connected to eight localized input cells, the efficiency of the learning was enchanced. The function connecting the eight input cells to the first layer was simple summation. He found the behavior very sensitive to differences in the reward function also, but we should like to dwell on the structured first layer. It is not clear from the published paper what the local eight point arrays looked like, but if they were the eight neighbors of a single point, then the linear sum would be similar to what we were doing in Section XIV. In any case, Roberts makes it clear that the logic connecting the input layer to the first perceptron layer was both local and homogeneous.

We feel this suggests very strongly the replacement of the first layer of a random perceptron by a completely structured, local and homogeneous layer which embodies in parallel the method we have been describing. To make the discussion simpler and because we feel strongly that the triangular grid is best, we shall use that grid.

In Fig. 46 we have shown the parallel digital logic required to compute the existence of edges and bend at C in the rhombic window of Fig. 42. The four edges incident to Point C can each occur in one of two directions, giving us eight possible directed edge segments. If point C is an edgepoint (EP), then eight possibilities are to be accounted for, four of which result in C being a curvature-point (CP). If we desire a binary perceptron then we might let CP be the cell value; otherwise, the eight possible edge pairs could be coded numerically to obtain the cell value. We conjecture that the simple binary model would do rather well on fairly distinct classes of patterns where gross curvature differences are apparent. A model more in keeping with this paper is one in which there are four cells in the first layer for each rhombus in the input layer, and each cell represents the existence or not of one of the four possible edge bends – ACB, DCE, BCA, EDC.

Another adaptive pattern recognition scheme which may benefit from our method is the Pandemonium of Selfridge.[19] Pandemonium is a multiple-level perceptron-like model in which the first level cells called "computational demons" pass information to next level known as "cognitive demons" and the loudest yelling cognitive demon is judged the winner. Each cognitive demon thus

represents some class of patterns to be recognized. The system is built in such a way that the weights in the functions defining "cognitive demon" outputs are variable with experience; a more striking feature is the system's ability to delete "bad" computational demons and construct new computational demons from two "good" ones by something analogous to a binary truth-function.

The reader may verify that so far nothing has been said relative to the two-dimensionality of patterns. Indeed, Pandemonium has been used for recognizing Morse Code. It is clearly intended to be a system in which the low level primitive computational demons are specified for each problem by a human analyst. In fact, the input layer is left unspecified and can vary – for the Morse Code model it was essentially a shift register.

Inasmuch as the computational demons are not specified, we propose that they might be constructed as in Fig. 46 when two-dimensional patterns are the inputs. If this is done we feel the combining of "good" demons should favor local pairs before others.

XIX. CONSTRUCTION OF RECOGNITION ALGORITHMS

We shall discuss in this section the construction via human analysis of a recognition algorithm for a stylized alphabet and then a possible approach to programmed construction of recognition algorithms.

In Figs. 47a-c we exhibit a somewhat artificial and extremely stylized alphabet; next to each letter we have entered the Area-Visit Signature as defined in an earlier section. The third column shows the minimum initial portion of each signature required to recognize that letter from all others. A complete decision tree for this alphabet in terms of the area-visit signatures is depicted in Fig. 48. Using the letter frequencies of English text shown in Fig. 49, we calculated the expected value of the number of tests needed per character and found it to be 4.1 for this decision tree.

The objections which would be raised to the foregoing are that actual alphabets are not so stylized and the problem of describing a correct decision tree, especially for a large set of characters, might be prohibitively difficult. In fact, each "character" of the alphabet* under consideration will be represented by several different area-visit signatures in any realistic environment. We would like a program which could look at samples representing each character and automatically construct a decision tree for recognizing the alphabet with a high success rate; we would hope the efficiency of the decision tree were near-optimal also, as measured by the expected time required to recognize a character. This measure would involve the entropy of characters as well as relative difficulty of the tests. The constraint that the polygonal description is already sequential means that there are not many choices among possible tests and hence the task confronting the decision-tree construction program may be a reasonable task after all.

At this point we would like to make a distinction between two varieties of learning; we do not know if the distinction is valid for human psychology, but we can make it fairly precise in the more abstract sanctuary of programmed models. In one learning procedure the inputs are fed to the learning mechanism one at a time and the mechanism is made aware of mistaken responses by a simple "punish or reward" stimulus following its response. It then "adapts" itself by making

*We shall use this term in a very general sense – a set of characters.

slight changes in its own organization and proceeds to wait for the next input stimulus. This strictly sequential learning seems to us a reasonable model for most human learning of which we are barely conscious.

We can, however, consider another learning mechanism which is more analytical and whose range of sight, so to speak, is much more global. This mechanism gobbles up large amounts of "input-correct response" data, as if in parallel, and organizes this mass of information by grouping input-response pairs with common responses. It then performs a thorough analysis of which properties of the inputs of each group serve to best discriminate the group from other groups. Only after the data has been thus organized and analyzed does the mechanism construct a decision tree to actually perform discrimination on its own. We feel this second learning mechanism has a great deal in common with that very conscious, very structured learning process known as scientific research.

The programmed models of which we are aware all fall into the first class, and it would seem that the organized analytical approach has been slighted. This is not to say that highly analytical recognition programs are non-existent, but rather that little attention has been given to learning programs which construct recognition programs by means of a global analysis of the input environment.

We intend to develop such a system to construct a decision tree for some fairly small alphabet, like the 26-letter Roman alphabet, allowing the character classes to vary widely as they do in practice.

When such a system has produced a decision tree, then it may be appropriate to allow further adaptation to be guided by a sequential learning mechanism similar to EPAM (Elementary Perceiver and Memorizer) developed by Feigenbaum.[20] If a new body of knowledge of substantial size needs to be learned, then control is returned to the analytical learner and the data is poured in, etc.

XX. COMPACT ENCODING

This section will discuss ways of encoding the information content of binary pictures and give an example of how many bits are required to encode using different encoding schemes on the same picture. The pattern we will use is a lower case 'a' digitized on a 32×32 array and reproduced* in Fig. 50. As shown in the lower left corner there are approximately 64 values for the Y coordinate of curvaturepoints, while within each line there are about 32 values of X. The edge lengths can be encoded with only 3 bits because no lengths actually exceed 7; note that the unit of diagonal length is only $\sqrt{2}/2$ while the horizontal and vertical unit length is 1. The pattern can be encoded with a mere 333 bits using the "structural description," whereas 1024 bits are required in the matrix mode. Figure 51 shows that, after smoothing, the pattern can be encoded as a sequence of (X, Y) values using 336 bits. The choice between these two encodings would depend on the particular needs of a given application, but both are an improvement on the matrix storage scheme in usability of format and storage efficiency. Even better encodings can be obtained by going to a Huffman code as described in Ref. 21, but decoding is more time consuming.

*Borrowed from Fischer, et al.[8]

XXI. ADVANTAGES OF THE CURVATUREPOINT METHOD

This section will be devoted to a summary of the salient advantages of the method described in this paper. This material will be organized into a simple sequence of paragraphs,each one roughly representing a single idea.

The first point we would like to make is that our method is general enough to be applicable to any "black/white" patterns whose significant entities are represented by connected sets of similar points. Hence, the method is applicable to any input pictures which can be made to comply with this condition by suitable preprocessing as shown , for example, by Figs. 9 and 10. As explained in Section XVI, the method also works for "grey-scale" pictures in which edges are important. The binary pattern in Fig. 36 can be described by our method also, showing that with a certain degree of subtlety in preprocessing even facial photographs can be transformed into "black/white" patterns. To be more specific, we hope to use the method of curvaturepoints to process the following input types: alphanumeric characters (upper and lower case, special, etc.); bubble chamber photographs; microphotographs of human chromosomes and similar cell structures; fingerprints; topographic maps; medical x-ray movies of internal organs; schematic drawings.

The "structural description" for a binary pattern represents a different format for the information in the original pattern but involves absolutely no loss of information. We mean this in the very precise sense that no two distinct patterns can possibly have identical structural descriptions and furthermore, that a reasonable algorithm exists for recovering the original binary pattern given only the "structural description." The algorithm which recovers the binary pattern is a simple consequence of a certain form of the Jordan curve theorem which says that any continuous curve joining two points "outside" a given closed curve (or both "inside") crosses the closed curve an even number of times, zero being admissable, of course. Details of this algorithm may be found in the Appendix.

In a recent article on the general subject of pattern recognition, Uhr[22] argues that "intuition" and "introspection" are valuable tools for pattern recognition methods as they are in other scientific research. We can presumably discover important properties by guessing how we ourselves seem to recognize. To make his point more concrete, Uhr reproduces part of a hypothetical discourse:

"Objects have edges; angles and loops are important; the interrelations between lengths and slopes are important."

According to Uhr, many subjects might even say these are the "meaningful" characteristics. The reason we mention these statements is that our method lends itself to the construction of recognition algorithms based on the existence and interrelationship of edges, angles, slopes and lengths – precisely the properties which Uhr feels would be suggested by "introspection." Inasmuch as our method lies close to human intuition, we feel that the construction of recognition algorithms based upon "structural descriptions" will be much less mysterious than it has seemed in other methods.

Another important advantage is that the data volume increases linearly with resolution. As a result we can afford to digitize with more resolution than is possible in binary matrix mode. This property will be greatly appreciated in applications where single objects have fine detail in several places but generally have smooth edges.

Although the method is clearly directed toward recognition via edge bend sequence, nevertheless the familiar signatures area, perimeter, oblongness, height, width and others are readily computed from the "structural description." Furthermore, the more closely a signature is associated with the object boundary curve, the more readily it can be computed.

The "structural description" is so organized that recognition algorithms can choose to be "blind" to overall size, absolute position and rotation if this is appropriate; on the other hand, the information is always there if needed. In a paper already cited, Selfridge[16] lists six properties which should not affect recognition of alphanumeric characters. They are:

1. Over-all size, between wide limits
2. Position
3. Orientation, within limits
4. Angular separation of the two uprights
5. Relative length of the two uprights
6. Thickness, and changes in thickness, of the line segments, within limits

The reader is invited to verify to his own satisfaction that the recognition of alphanumeric characters based on the "structural description" can easily be made to comply with these conditions.

Recognition based on the "structural description" lends itself to a decision tree formulation which, as Uhr[22] points out, is generally more efficient than a property vector approach since measurements are made as needed and not unless they are called for. The economy which can be obtained using decision trees is exemplified in the stylized alphabet recognition depicted in Figs. 47 and 48. The characters "A" and "N" can be recognized after a very small portion of their edge sequence has been examined.

Another recommendation which Uhr[22] suggests for pattern recognition systems is that they be compatible with an "elegant language" for describing pattern classes, one in which complex pattern class descriptions can be built from simpler descriptions, etc. down to primitives. Kirsch[13] and Narasimhan[2] have stressed this point also, but along with Uhr they consider the "picture point" to be the basic primitive, whereas we argue the usefulness of considering "curvaturepoint" as primitive. In any case, Sections XVII and XIII demonstrate how formal syntax can be applied rather naturally to describe patterns, and we have reason to expect that a fairly sophisticated picture language can be constructed to speak about "structural descriptions." Our method seems to be the first computer recognition model in which complex relations of "insidedness" among objects are detectable with little added effort.

The utter simplicity of our method, especially as seen in the conditions for "curvaturepoint" on a triangular grid, argues in its favor, not only because of efficiency but also because the scientifically desirable hypothesis is the simpler one and, in fact, simple hypotheses are very often true.*

As a final point, we mention the compatibility of our method with other recognition systems. Our method represents an approach to the first (lowest) level of picture function selection and as such it can be incorporated into "pattern learning programs" which have already been developed. We have discussed its use for three existing systems – Robert's Perceptron, Selfridge's Pandemonium and Prather-Uhr's learning program.

---

*See Uhr (Ref. 22, p. 370)

XXII. THREE PHASES OF PICTURE PROCESSING

In this section we shall attempt to outline our own approach to the general problem of recognizing two-dimensional black/white patterns and in particular describe what we feel to be the most appropriate subdivision of the recognition problem. We choose to divide the problem into three phases:

1. Preprocessing
2. Description
3. Recognition

Preprocessing is the term we associate with whatever sequence of transformations is required to eliminate noise and re-establish the "correct" degree of connectedness in the picture pattern. Experience seems to indicate that "correctness" depends primarily on the characteristics of the original patterns and the particular types of noise or distortion being encountered. The example of bubble chamber photographs demonstrates what we mean more concretely. Referring to Fig. 7, we see that one of the tracks on the right side is composed of a sequence of small blobs strung out in a very straight line. This is indeed a faithful representation of what the camera actually sees, for these blobs are bubbles of gaseous hydrogen caused by a moving charged particle which is itself invisible. If we consider the particle tracks (space curves) as constituting what we would like to see in the picture, then we will want preprocessing which connects together the bubbles as much as possible along the direction of the track (see Fig. 8). The point we wish to emphasize is that the "correct" type of preprocessing is dictated by the source of pictures and noise independent of what particular "physical event" types are to be recognized.

The "description" of a preprocessed pattern consists of a transformation of the pattern from the matrix or grid format to a different format more readily useable by recognition algorithms. Our transformation to "structural description" constitutes one approach to this phase.

Finally, "recognition" consists of assigning a pattern to one of several possible classes as a result of detailed examination of the pattern description.

Returning to the initial phase for a moment, we must point out that in the interests of simplicity and efficiency it is useful to restrict preprocessing

transformations to be:

1. Local – This means that the function defining the new value of a picture point has as inputs only a small number of original values, all of whose picture points are near the one being transformed.
2. Homogeneous – This means that the transformed value of a picture point is computed by the same function for all points of the picture.

We are reasonably confident that the more commonly occurring preprocessing requirements can be effectively met using these restricted transformations and, what is probably more important, these are the transformations which are readily implemented by either parallel hardware or sequential programming.

A simple example of the three phases of pattern recognition is depicted in Figs.52 - 54. A very noisy pattern is preprocessed by the application of two local homogeneous transformations. The first is a gap-filling operation with strict vertical bias; the second simply cleans away isolated points. The "structural description" is constructed and smoothed* as shown in Fig. 53. Then the sequential edge description is condensed by grouping bends and recognizing long and straight portions as seen in Fig. 54. The final step is to scrutinize the remaining sequence from left to right until we have reduced to one possible classification, namely, the letter "N."

*The problem of whether smoothing is part of description or recognition is not so obviously solved, but the argument is not likely to be of any benefit anyway.

REFERENCES

1. R.A. Kirsch, L. Cahn, C. Ray and G.H. Urban, "Experiments in Processing Pictorial Information with a Digital Computer," Proceedings of the Eastern Joint Computer Conference (1957); pp. 221-229.
2. R. Narasimhan, "Syntax-Directed Interpretation of Classes of Pictures," Communications of the Association for Computing Machinery, 9 (No. 3), 166-172 (March 1966).
3. R. Buckminster Fuller, "Conceptuality of Fundamental Structures," in Structure in Art and in Science, ed. Gyorgy Kepes (George Braziller, Inc., New York, 1965); pp. 66-88.
4. G.P. Dinneen, "Programming Pattern Recognition," Proceedings of the Western Joint Computer Conference (1955); pp. 94-101.
5. R. Narasimhan, "Labeling Schemata and Syntactic Descriptions of Pictures," Information and Control, 7, 151-179 (1964).
6. S. Yamada and J.P. Fornango, "Experimental Results for Local Filtering of Digitized Pictures," Report No. 184, Department of Computer Science, University of Illinois (June 1965).
7. R. S. Ledley and F. H. Ruddle, "Chromosome Analysis by Computer," Scientific American (April 1966); pp. 40-46.
8. G.L. Fischer, Jr., D.K. Pollock, B. Raddack and M.E. Stevens, editors, Optical Character Recognition (Spartan Books, Washington, D.C., 1962).
9. R.M. Brown, "On-Line Computer Recognition of Handprinted Characters," IEEE Trans. on Electronic Computers, EC-13, 750-752 (December 1964).
10. B.H. McCormick, Sylvian R. Ray, Kenneth C. Smith and S. Yamada, "Illiac III: A Processor of Visual Information," Report No. 183, Department of Computer Science, University of Illinois (June 1965).
11. B.H. McCormick, "The Illinois Pattern Recognition Computer (Illiac III)," Annual ACM National Conference, Denver, Colorado, August 1963.
12. R. Narasimhan and J.P. Fornango, "Some Further Experiments in the Parallel Processing of Pictures," IEEE Trans. on Electronic Computers, EC-13, 748-750 (December 1964).
13. R.A. Kirsch, "Computer Interpretation of English Text and Picture Patterns," IEEE Trans. on Electronic Computers, EC-13, 363-376 (August 1964).

14. R.L. Grimsdale, F.H. Sumner, C.J. Tunis and T. Kilburn, "A System for the Automatic Recognition of Patterns," Proc. Inst. Elect. Engrs. 106B, 210-221 (1958).
15. R.C. Prather and L.M. Uhr, "Discovery and Learning Techniques for Pattern Recognition," Proceedings of the 19th National Conference of the Association for Computing Machinery (1964); p. D2.2 .
16. O.G. Selfridge, "Pattern Recognition and Modern Computers," Proceedings of the Western Joint Computer Conference (1955); pp. 91-93.
17. J.S. Bomba, "Alpha-Numeric Character Recognition Using Local Operations," Proceedings of the Eastern Joint Computer Conference (1959); pp. 218-224.
18. L.G. Roberts, "Pattern Recognition with an Adaptive Network," in Pattern Recognition, ed. Leonard Uhr (John Wiley & Sons, New York, 1966); pp. 295-300.
19. O.G. Selfridge, "Pandemonium: A Paradigm For Learning," Ibid., pp. 339-348.
20. E.A. Feigenbaum, "The Simulation of Verbal Learning Behavior," in Computers and Thought, ed. E.A. Feigenbaum and J. Feldman (McGraw-Hill Book Company, New York, 1963); pp. 297-309.
21. J.R. Pierce, Symbols, Signals and Noise, (Harper & Row, New York, 1961); pp. 94-96.
22. L. Uhr, "Pattern Recognition," in Pattern Recognition, ed. Leonard Uhr (John Wiley & Sons, New York, 1966); pp. 365-381.
23. H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," IRE Transactions on Electronic Computers, EC-10, No. 2, pp. 260-268 (1961).
24. ____ ____ ____, "Techniques for the Digital Computer Analysis of Chain-encoded Arbitrary Plane Curves," Proc. National Electronics Conference (1961); pp. 421-432.
25. ____ ____ ____, "On the Digital Computer Classification of Geometric Line Patterns," Proc. National Electronics Conference (1962); pp. 312-324.
26. S.H. Unger, "A Computer Oriented Toward Spatial Problems," Proc. IRE, 46, pp. 1744-1750 (October 1958).

27. ____ ____ ____, "Pattern Detection and Recognition," Proc. IRE, 47, pp. 1737-1752 (October 1959).

28. F. Kuhl, "Classification and Recognition of Hand-Printed Characters," IEEE International Convention Record, Part 4 (1963); pp. 75-93.

29. R. Courant and H. Robbins, What is Mathematics? (Oxford University Press, London, 1941); pp. 267-269.

APPENDIX A

ALGOL 60 PROCEDURE TO FIND CURVATUREPOINTS

```
PROCEDURE CURVATUREPOINTS(PICTURE,N,X,Y,EDGEIN,EDGEOUT,NPOINTS);
     COMMENT THIS PROCEDURE SCANS A BINARY PATTERN,PICTURE[I,J],
             FOR 1≤I≤N AND 1≤J≤N AND DETECTS ALL CURVATUREPOINTS.
             THESE POINTS ARE RECORDED AS FOUR ARRAYS,X,Y,EDGEIN
             AND EDGEOUT[POINT].X[P],Y[P] DENOTE THE POSITION OF
             THE CURVATUREPOINT P WHILE EDGEIN[P] AND EDGEOUT[P]
             GIVE THE DIRECTIONS FOR THE INCOMING AND OUTGOING
             EDGES AT POINT P. THE TOTAL NUMBER OF CURVATUREPOINTS
             DETECTED IS RECORDED AS NPOINTS. THE PATTERN IS
             ASSUMED TO BE BORDERED BY ZEROS;
     BOOLEAN ARRAY PICTURE[1,1];
     INTEGER ARRAY X,Y,EDGEIN,EDGEOUT[1];
     INTEGER N,NPOINTS;
     BEGIN
          BOOLEAN D1,B2,D3,D4,D5,D6,B1,B2,B3,B4,B5,B6;
          INTEGER I,J;


          PROCEDURE HORIZONTAL;
               COMMENT THIS PROCEDURE DETERMINES WHETHER OR NOT
                       A CURVATUREPOINT EXISTS AT THE CENTER OF
                       THE HORIZONTAL (2×3) WINDOW WHOSE PICTURE
                       VALUES ARE GIVEN BY D1 THRU D6;
               BEGIN
                    INTEGER IN,OUT;
                    LABEL NOEDGE;
                    IF D2 THEN
                         BEGIN
                              IF NOT D5 THEN
                                   BEGIN
                                        IN←IF D6 THEN 3
                                      ELSE IF D3 THEN 4
                                      ELSE            5;
                                       OUT←IF D4 THEN 5
                                      ELSE IF D1 THEN 4
                                      ELSE            3
                                   END
                                   ELSE GO TO NOEDGE
                         END
                         ELSE
                         BEGIN
                              IF D5 THEN
                                   BEGIN
                                        IN←IF D1 THEN 7
                                      ELSE IF D4 THEN 0
                                      ELSE            1;
                                       OUT←IF D3 THEN 1
                                      ELSE IF D6 THEN 0
                                      ELSE            7
                                   END
```

```
                    ELSE GO TO NOEDGE
               END;
               COMMENT IF THIS PLACE IS REACHED THEN
                       AN EDGEPOINT HAS BEEN DETECTED
                       AND SHOULD BE ADDED TO THE LIST
                       IF IT IS ALSO A CURVATUREPOINT;
               IF IN≠OUT THEN
                    BEGIN
                         NPOINTS←NPOINTS+1;
                         X[NPOINTS]←2×J+1;
                         Y[NPOINTS]←2×I;
                         EDGEIN[NPOINTS]←IN;
                         EDGEOUT[NPOINTS]←OUT
                    END;
NOEDGE:
END HORIZONTAL;


PROCEDURE VERTICAL;
     COMMENT THIS PROCEDURE DETERMINES WHETHER OR NOT
             A CURVATUREPOINT EXISTS AT THE CENTER OF
             THE VERTICAL (3×2) WINDOW WHOSE PICTURE
             VALUES ARE GIVEN BY B1 THRU B6;
     BEGIN
          INTEGER IN,OUT;
          LABEL NOEDGE;
          IF B2 THEN
               BEGIN
                    IF NOT B5 THEN
                         BEGIN
                              IN←IF B6 THEN 1
                                 ELSE IF B3 THEN 2
                                 ELSE            3;
                               OUT←IF B4 THEN 3
                                 ELSE IF B1 THEN 2
                                 ELSE            1
                         END
                         ELSE GO TO NOEDGE
               END
               ELSE
               BEGIN
                    IF B5 THEN
                         BEGIN
                              IN←IF B1 THEN 5
                                 ELSE IF B4 THEN 6
                                 ELSE            7;
                               OUT←IF B3 THEN 7
                                 ELSE IF B6 THEN 6
                                 ELSE            5
                         END
                         ELSE GO TO NOEDGE
               END;
               COMMENT  EDGEPOINT DETECTED;
               IF IN≠OUT THEN
                    BEGIN
```

```
                              NPOINTS←NPOINTS+1;
                              X[NPOINTS]←2×J;
                              Y[NPOINTS]←2×I+1;
                              EDGEIN[NPOINTS]←IN;
                              EDGEOUT[NPOINTS]←OUT
                         END;
          NOEDGE;
          END VERTICAL;


     COMMENT CURVATUREPOINTS REALLY STARTS HERE,
             INITIALIZE NPOINTS AS ZERO;
     NPOINTS←0;

     FOR I←1 STEP 1 UNTIL N-2 DO
          BEGIN
               FOR J←1 STEP 1 UNTIL N-2 DO
                    BEGIN
                         D1←PICTURE[I,J];
                         D2←PICTURE[I,J+1];
                         D3←PICTURE[I,J+2];
                         D4←PICTURE[I+1,J];
                         D5←PICTURE[I+1,J+1];
                         D6←PICTURE[I+1,J+2];
                         HORIZONTAL
                    END;
               FOR J←1 STEP 1 UNTIL N-1 DO
                    BEGIN
                         B1←PICTURE[I,J+1];
                         B2←PICTURE[I+1,J+1];
                         B3←PICTURE[I+2,J+1];
                         B4←PICTURE[I,J];
                         B5←PICTURE[I+1,J];
                         B6←PICTURE[I+2,J];
                         VERTICAL
                    END
          END;

     COMMENT NOW DO LAST HORIZONTAL ROW I=N-1 ;
     I←N-1;
     FOR J←1 STEP 1 UNTIL N-2 DO
          BEGIN
               D1←PICTURE[I,J];
               D2←PICTURE[I,J+1];
               D3←PICTURE[I,J+2];
               D4←PICTURE[I+1,J];
               D5←PICTURE[I+1,J+1];
               D6←PICTURE[I+1,J+2];
               HORIZONTAL
          END
END CURVATUREPOINTS;
```

APPENDIX B

ALGOL 60 PROCEDURE TO LINK CURVATUREPOINTS

```
PROCEDURE LINKAGE(X,Y,EDGEIN,EDGEOUT,NPOINTS,N,NPOLYGONS,TOP,BOTTOM,
          NEXTPOINT,NEXTEDGE,LASTPOINT,LASTEDGE,ANGLE,LENGTH);
    INTEGER NPOINTS,N,NPOLYGONS;
    INTEGER ARRAY X,Y,EDGEIN,EDGEOUT,TOP,BOTTOM,NEXTPOINT,NEXTEDGE,
            LASTPOINT,LASTEDGE,ANGLE,LENGTH[1];
    COMMENT LINKAGE ACCEPTS THE SET OF CURVATUREPOINTS DEFINED BY
        X,Y,EDGEIN,EDGEOUT[CP] FOR 1≤CP≤NPOINTS AND,ASSUMING
        THEM TO BE ORDERED BY INCREASING X WITHIN INCREASING Y,
        IT LINKS THEM PROPERLY INTO DIRECTED CLOSED POLYGONS.
        EDGES ARE CREATED AS NEEDED AND GIVEN ATTRIBUTES ANGLE
        AND LENGTH.THE POLYGONS ARE INDEXED BY INTEGERS IN THE
        RANGE [1:NPOLYGONS] BUT ONLY THOSE INDICES J SUCH THAT
        TOP[J]≠0 ARE OF INTEREST. IN CASE TOP[J]≠0 IT GIVES THE
        INDEX OF THE CURVATUREPOINT AT THE TOP OF POLYGON J.
        BOTTOM[J] GIVES THE LOWEST OR BOTTOM CURVATUREPOINT.
        THE ARRAYS NEXTPOINT,NEXTEDGE,LASTPOINT,LASTEDGE SERVE
        AS POINTERS TO ESTABLISH THE CORRECT LINKAGES BETWEEN
        POINTS AND EDGES OF THE STRUCTURAL DESCRIPTION;
    BEGIN
        BOOLEAN INFLAG,OUTFLAG;
        INTEGER IN,OUT,LAST,NEWEDGE,ANTENODE,POSTNODE;
        INTEGER ARRAY SAMEX[2:2×N-1],
                      SAMEPLUS[5:4×N-5],
                      SAMEMINUS[-2×N+5:2×N-5],
                      HEADER,START,FINISH[1:NPOINTS];


        INTEGER PROCEDURE OLDPOINT(THETA,CP);
            COMMENT FINDS THE MOST RECENTLY ENCOUNTERED
                    CURVATUREPOINT ALONG DIRECTION THETA(=0 TO 3)
                    THROUGH NEW CURVATUREPOINT CP;
            INTEGER THETA,CP;
            BEGIN
                OLDPOINT← IF THETA=0 THEN LAST
                     ELSE IF THETA=1 THEN SAMEPLUS[X[CP]+Y[CP]]
                     ELSE IF THETA=2 THEN SAMEX[X[CP]]
                     ELSE                 SAMEMINUS[X[CP]-Y[CP]]
            END OLDPOINT;


        PROCEDURE STORE(THETA,CP);
            COMMENT RECORDS CURVATUREPOINT CP RELATIVE TO
                    DIRECTION THETA THROUGH(X[CP],Y[CP]) ;
            INTEGER THETA,CP;
            BEGIN
                IF THETA=0 THEN LAST←CP
            ELSE IF THETA=1 THEN SAMEPLUS[X[CP]+Y[CP]]←CP
            ELSE IF THETA=2 THEN SAMEX[X[CP]]←CP
            ELSE                 SAMEMINUS[X[CP]-Y[CP]]←CP
            END STORE;
```

```
PROCEDURE LINKIN(CP,IN);
     COMMENT CREATES A LINKAGE BETWEEN CURVATUREPOINT CP
             AND AN ALREADY ENCOUNTERED POINT WHOSE
             OUTGOING EDGE HAS THE SAME DIRECTION AS THE
             INCOMING EDGE AT CP. LATTER IS CALLED IN.
             AN EDGE IS CREATED AT THIS TIME AND PROPERLY
             LINKED TO THE TWO POINTS WHICH SERVE TO
             DEFINE IT. LENGTH AND ANGLE FOR THE NEW EDGE
             ARE CALCULATED AND PLACED IN MEMORY;
     INTEGER CP,IN;
     BEGIN
          INTEGER NODE;
          NODE←OLDPOINT(IN,CP);
          NEWEDGE←NEWEDGE+1;
          NEXTEDGE[NODE]←NEWEDGE;
          NEXTPOINT[NEWEDGE]←CP;
          LASTEDGE[CP]←NEWEDGE;
          LASTPOINT[NEWEDGE]←NODE;
          ANGLE[NEWEDGE]←IN;
          LENGTH[NEWEDGE]←IF IN=0 OR IN =4
                             THEN ABS(X[NODE]-X[CP])×.5
                          ELSE ABS(Y[NODE]-Y[CP])×
                          (IF ODD[IN] THEN .707 ELSE .5)
     END LINKIN;


PROCEDURE LINKOUT(CP,OUT);
     COMMENT ANALOGOUS TO LINKIN;
     INTEGER CP,OUT;
     BEGIN
          INTEGER NODE;
          NODE←OLDPOINT(OUT,CP);
          NEWEDGE←NEWEDGE+1;
          LASTEDGE[NODE]←NEWEDGE;
          LASTPOINT[NEWEDGE]←CP;
          NEXTEDGE[CP]←NEWEDGE;
          NEXTPOINT[NEWEDGE]←NODE;
          ANGLE[NEWEDGE]←OUT;
          LENGTH[NEWEDGE]← IF OUT=0 OR OUT=4
                              THEN ABS(X[NODE]-X[CP])×.5
                           ELSE ABS(Y[NODE]-Y[CP])×
                           (IF ODD[OUT] THEN .707 ELSE .5)
     END LINKOUT;


PROCEDURE BOTHLINKS;
     COMMENT IN CASE BOTH EDGES AT CP HAVE JUST BEEN
             LINKED,IT IS NECESSARY TO MERGE THE TWO
             CONNECTED SETS INTO A SINGLE POLYGONAL
             CHAIN OR IF CP HAS JUST BEEN LINKED TO THE
             TWO ENDS OF A SINGLE CHAIN THEN CP BECOMES
             THE BOTTOM OF A NEW CLOSED POLYGON;
     BEGIN
```

```
          IF HEADER[ANTENODE]=HEADER[POSTNODE] THEN
               BOTTOM[HEADER[ANTENODE]]←CP
     ELSE IF HEADER[ANTENODE]<HEADER[POSTNODE] THEN
               BEGIN
                    FINISH[HEADER[ANTENODE]]←
                         FINISH[HEADER[POSTNODE]];
                    TOP[HEADER[POSTNODE]]←0
               END
     ELSE
               BEGIN
                    START[HEADER[POSTNODE]]←
                         START[HEADER[ANTENODE]];
                    TOP[HEADER[ANTENODE]]←0
               END
    END BOTHLINKS;


PROCEDURE NOLINKS;
     COMMENT IF NEITHER EDGE AT CP HAS BEEN LINKED,THEN
             A NEW HEADER MUST BE CREATED FOR CP;
     BEGIN
          NPOLYGONS←NPOLYGONS+1;
          HEADER[CP]←NPOLYGONS;
          START[NPOLYGONS]←CP;
          FINISH[NPOLYGONS]←CP;
          TOP[NPOLYGONS]←CP
     END NOLINKS;


COMMENT PROGRAM STARTS HERE. INITIALIZE ;
NEWEDGE←NPOLYGONS←0;
FOR CP←1 STEP 1 UNTIL NPOINTS DO
     BEGIN
          IN←EDGEIN[CP];
          INFLAG← NOT(1≤IN AND IN≤4);
          K←IF IN>3 THEN IN-4 ELSE IN;
          IF INFLAG THEN LINKIN(CP,K)
                    ELSE STORE(K,CP);
          OUT←EDGEOUT[CP];
          OUTFLAG← (1≤OUT AND OUT≤4);
          K←IF OUT>3 THEN OUT-4 ELSE OUT;
          IF OUTFLAG THEN LINKOUT(CP,K)
                     ELSE STORE(K,CP);
          COMMENT NEXT PORTION OF LOOP UPDATES THE
                  HEADER INFORMATION FOR CP DEPENDING
                  ON VALUE OF FLAGS INFLAG,OUTFLAG.
                  PROCEDURE BOTHLINKS OR NOLINKS MAY BE
                  CALLED ON AT THIS POINT;
          IF INFLAG THEN
               BEGIN
                    ANTENODE←LASTPOINT[LASTEDGE[CP]];
                    IF OUTFLAG THEN
                         BEGIN
                              POSTNODE←
                                NEXTPOINT[NEXTEDGE[CP]];
```

```
                              BOTHLINKS
                         END
                    ELSE
                         BEGIN
                              FINISH[HEADER[ANTENODE]]←CP;
                              HEADER[CP]←HEADER[ANTENODE]
                         END
               END
          ELSE
                    IF OUTFLAG THEN
                         BEGIN
                              START[HEADER[POSTNODE]]←CP;
                              HEADER[CP]←HEADER[POSTNODE]
                         END
                    ELSE NOLINKS
     END LOOP ON CP;
END LINKAGE;
```

APPENDIX C

ALGORITHM TO RECOVER ORIGINAL BINARY PATTERN

One of the claims made earlier concerning the "structural description" was that a mechanical algorithm exists which will recover the original digitized binary matrix pattern given only the structural description. We shall give a brief outline of such an algorithm and discuss the theoretical foundation for its validity.

The famous mathematical theorem known as the "Jordan Curve Theorem" states that any simple closed curve in the plane divides the remainder of the plane into two distinct sets called the "inside" and "outside," such that any two points in the same set can always be connected with a continuous curve lying entirely in that same set; any continuous plane curve joining a point "inside" the curve to one "outside" must have at least one point in common with the closed curve. The proof of this theorem is much simpler for polygons than for general curves, and Ref. 29 contains a proof for the polygonal case which depends on the fact that if a point "outside" a closed curve Γ is connected by a straight line L to another point P, then P is also "outside" Γ if and only if L intersects Γ an even number of times. If L intersects Γ an odd number of times, then P is "inside" Γ.

The algorithm we propose is based on the latter fact, which is also proved in Ref. 29. Referring to Fig. 55, we shall show how it is possible to determine that the points in ranges β and δ (for $X = X_0$) are inside Γ while ranges α, γ and ϵ are outside. The proper assignment of values then depends only on the status of Γ ("object" or "hole").

The determination that ranges β and δ are inside Γ proceeds as follows:

Curve Γ is traced from its top all the way around and back to the top again. Whenever the curve Γ crosses the vertical line $X = X_0$, we record the fact by making a mark in all points below the crossing. For example, Γ crosses first at e_1 and all points in ranges β, γ, δ and ϵ are marked + as shown in the table. This means that each of these marked points can be connected by a straight line to the top of the picture area only by a line crossing Γ at e_1. The table is filled in for e_2, e_3, and e_4 in an entirely similar manner. After e_4, there are no more crossings and the curve returns to the starting point. We then determine which points have received an odd number of marks and find that all points

of ranges β and δ are inside Γ.

When this procedure is carried out for all vertical lines X_o, then the "inside" points for Γ will have been determined. In any actual implementation, we assume that all X_o's would be handled simultaneously while Γ was being traced only once. The hierarchy of curves does not invalidate the method but merely requires some care in the order of processing curves.

No claim is made for the efficiency of this procedure; we merely wanted to show with some degree of mathematical rigor that the "structural description" contains all the information of the original digitized pattern and it can be mechanically recovered if necessary.

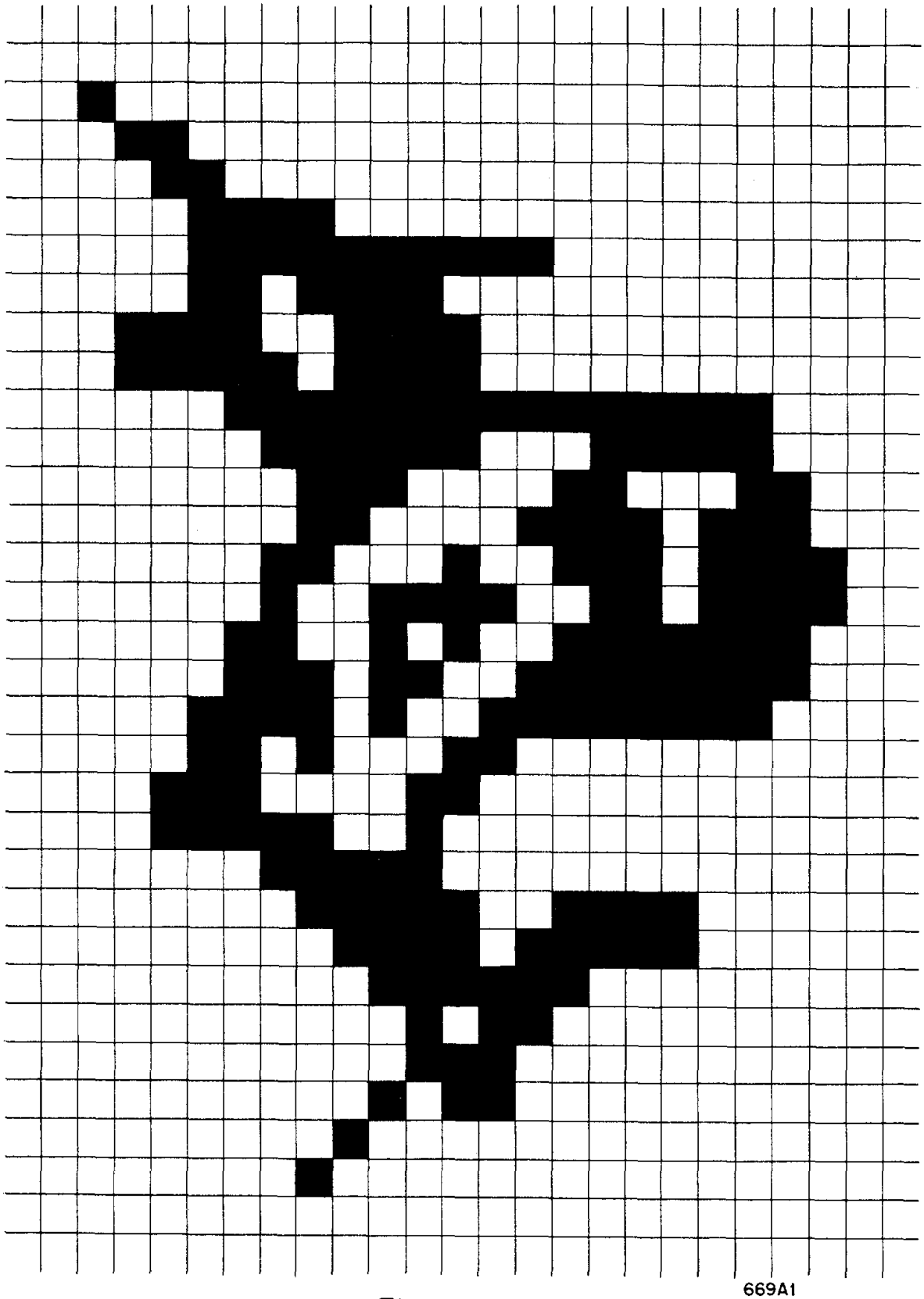


Fig. 1

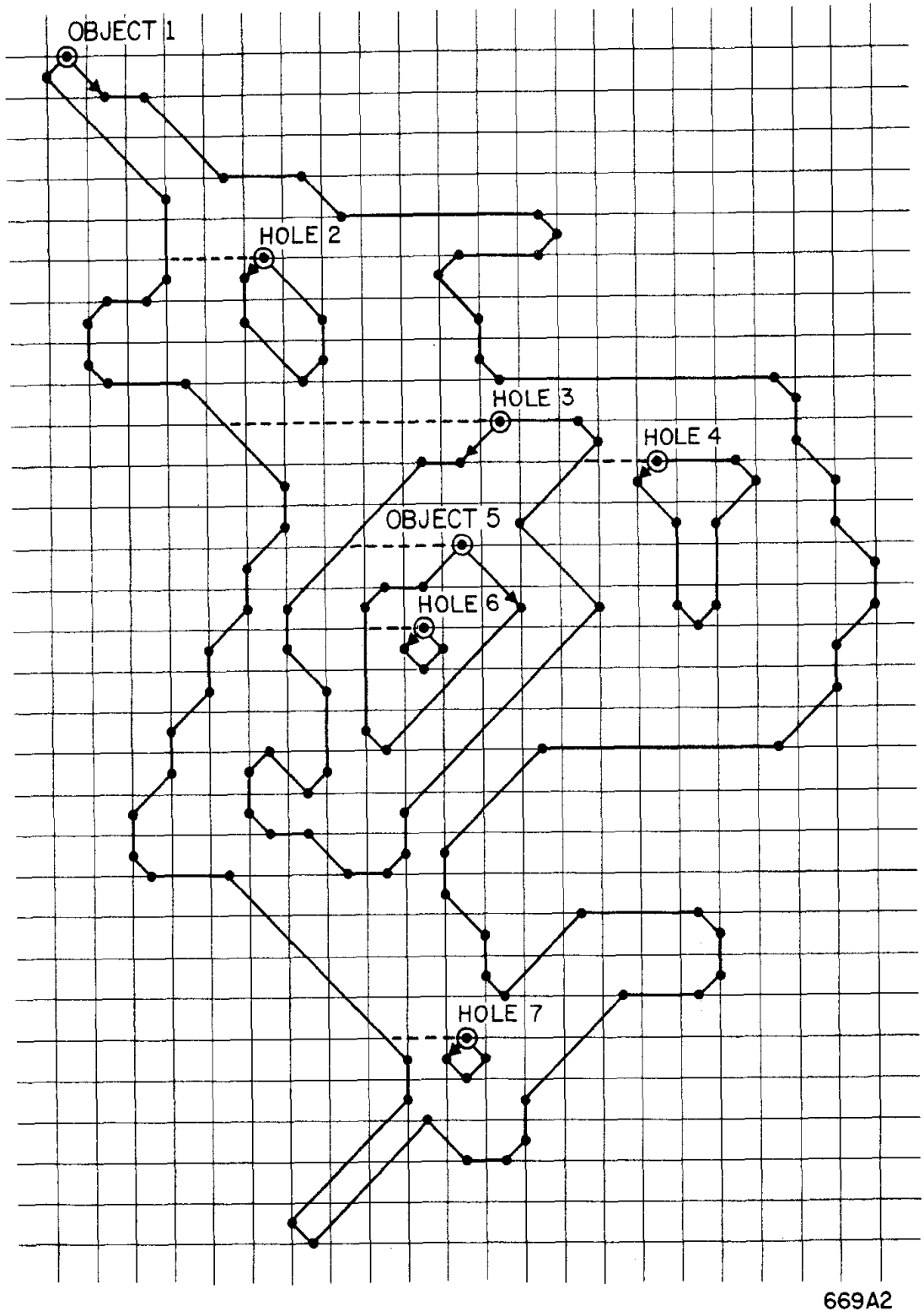


FIGURE 2

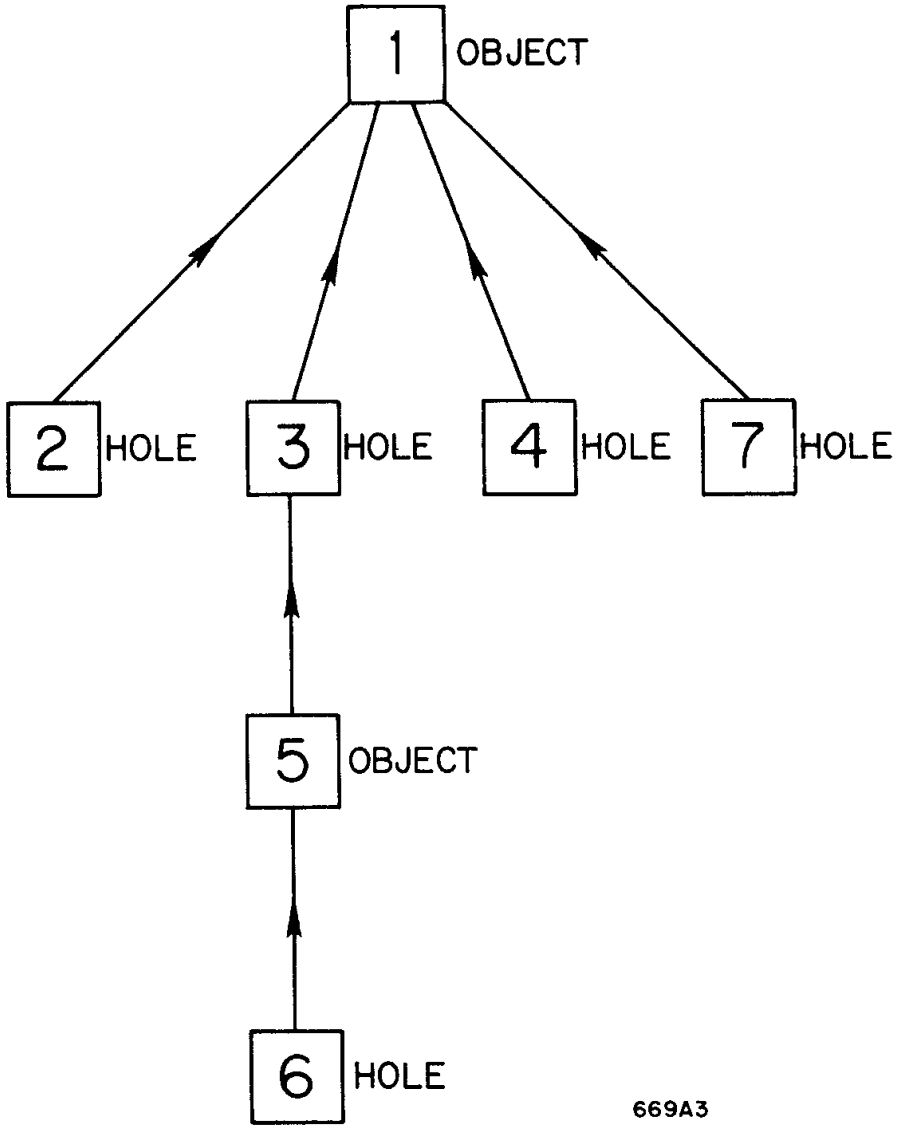
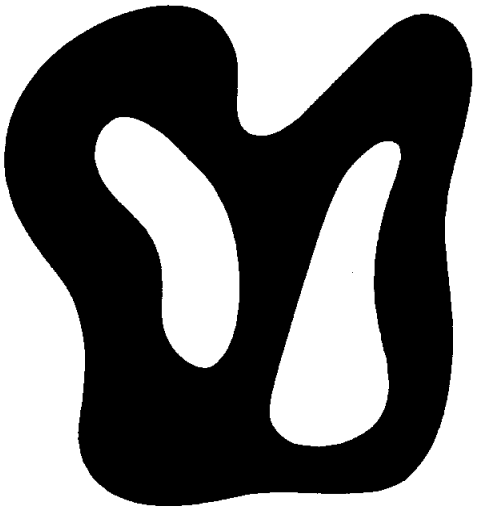


Fig. 3



669A4

FIGURE 4

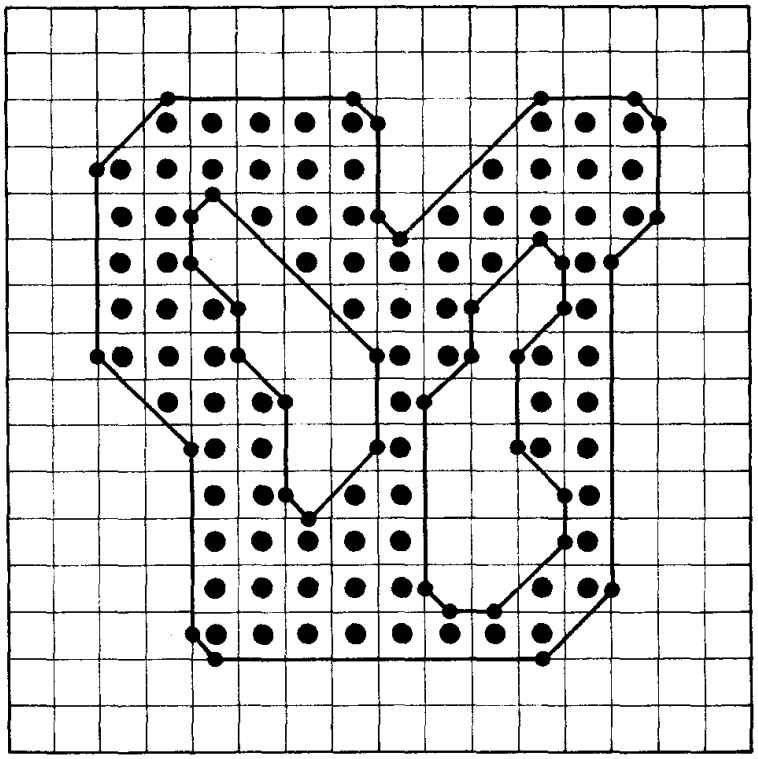


FIG. 5

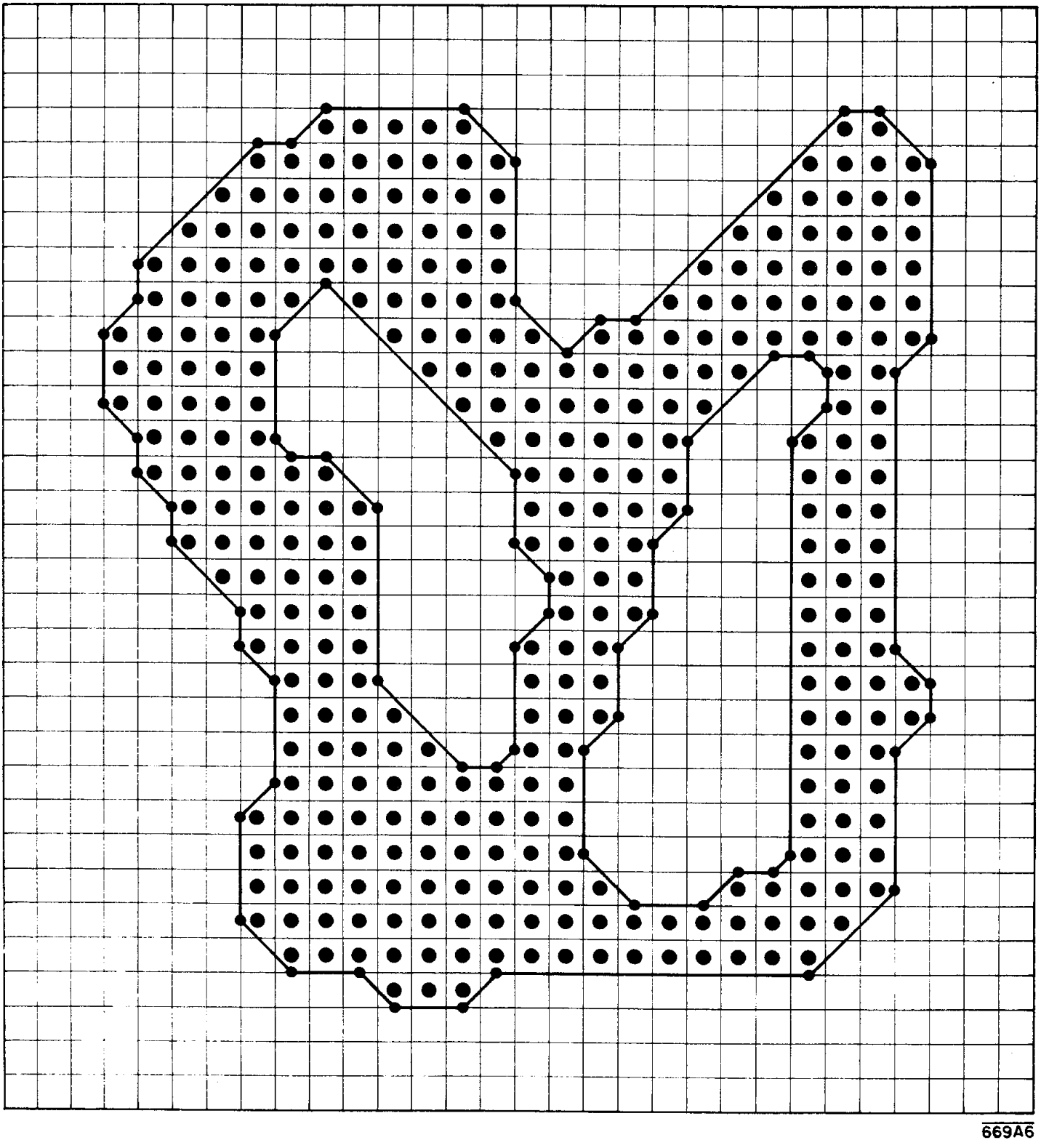


FIG. 6

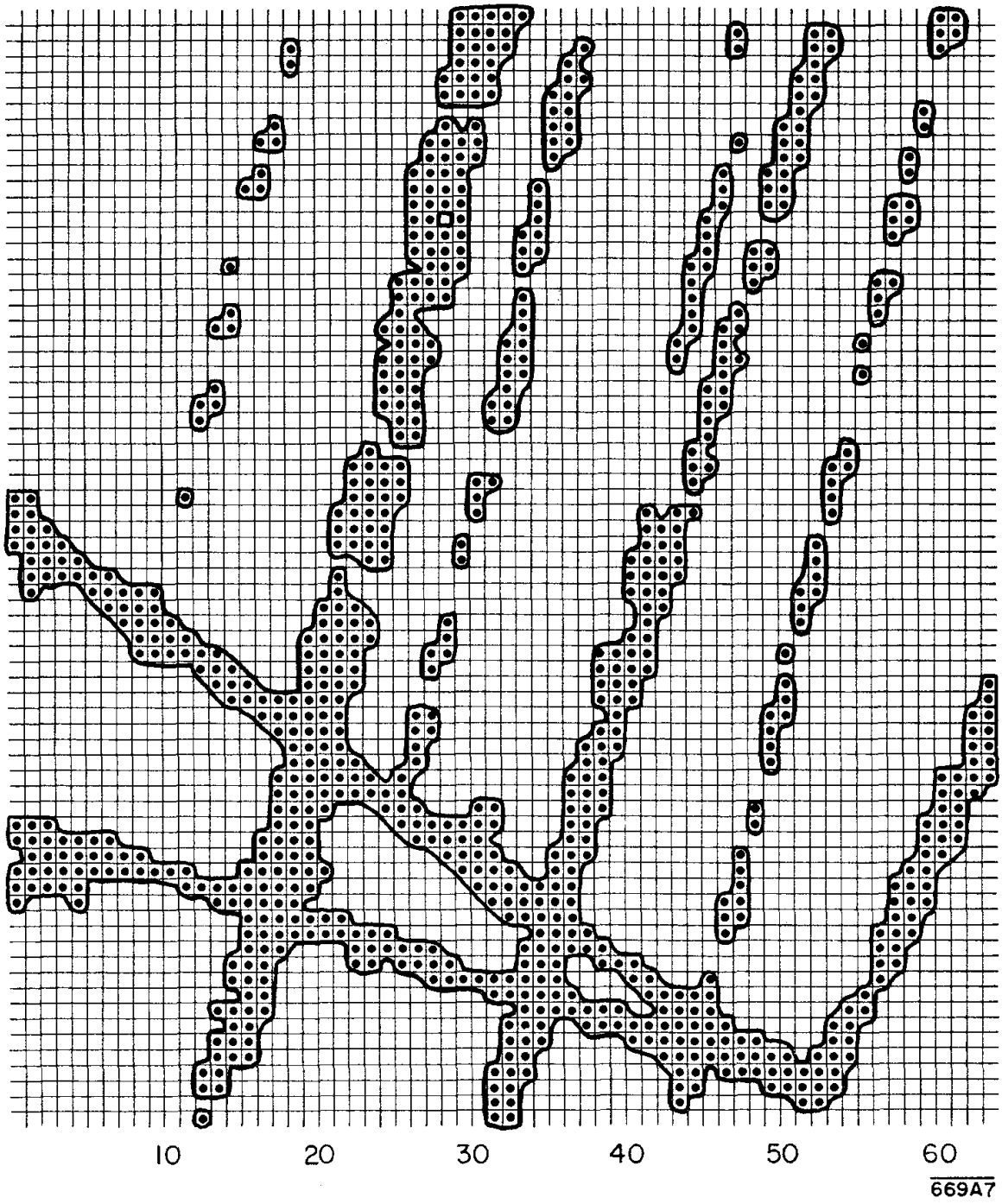


FIG. 7

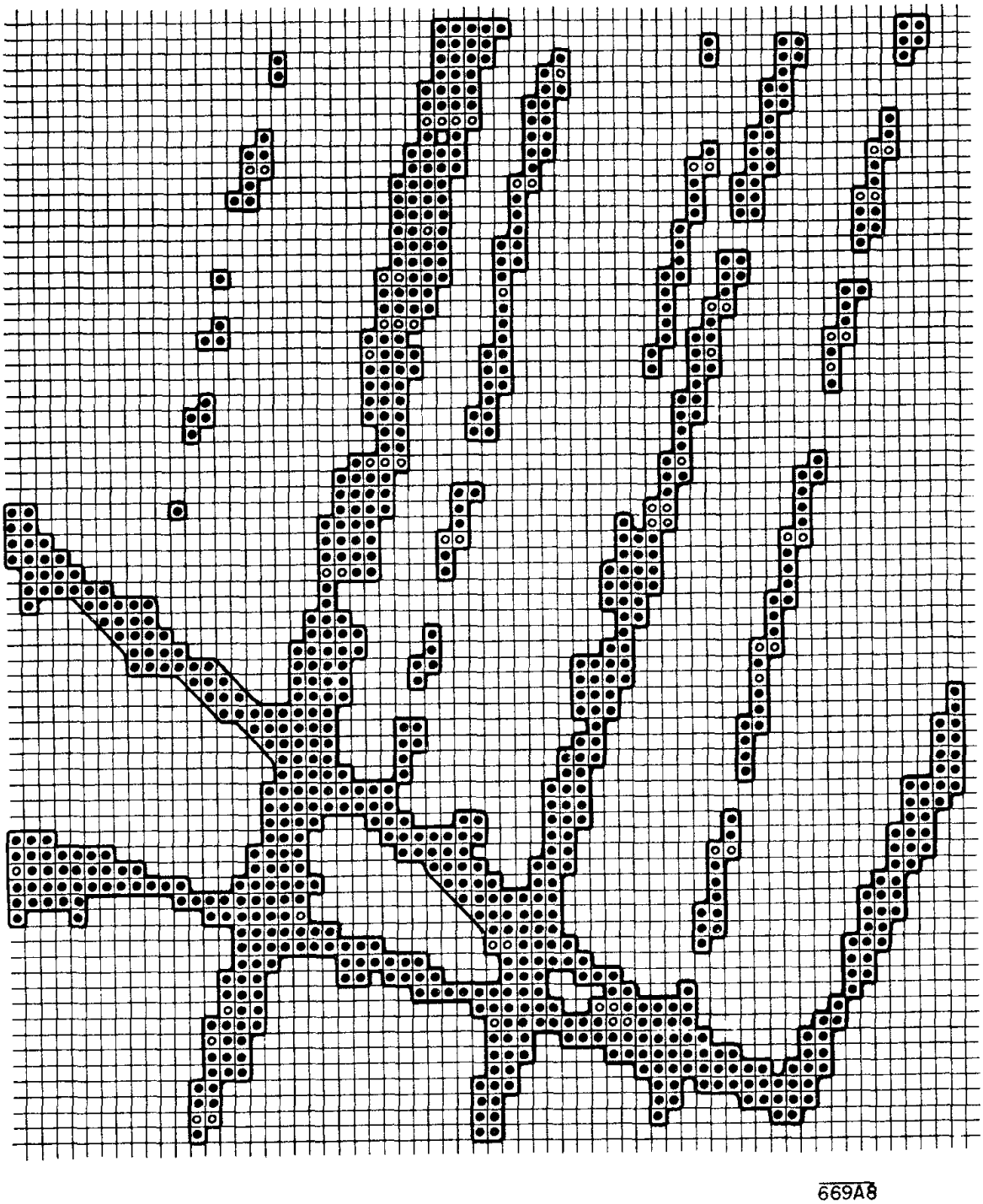
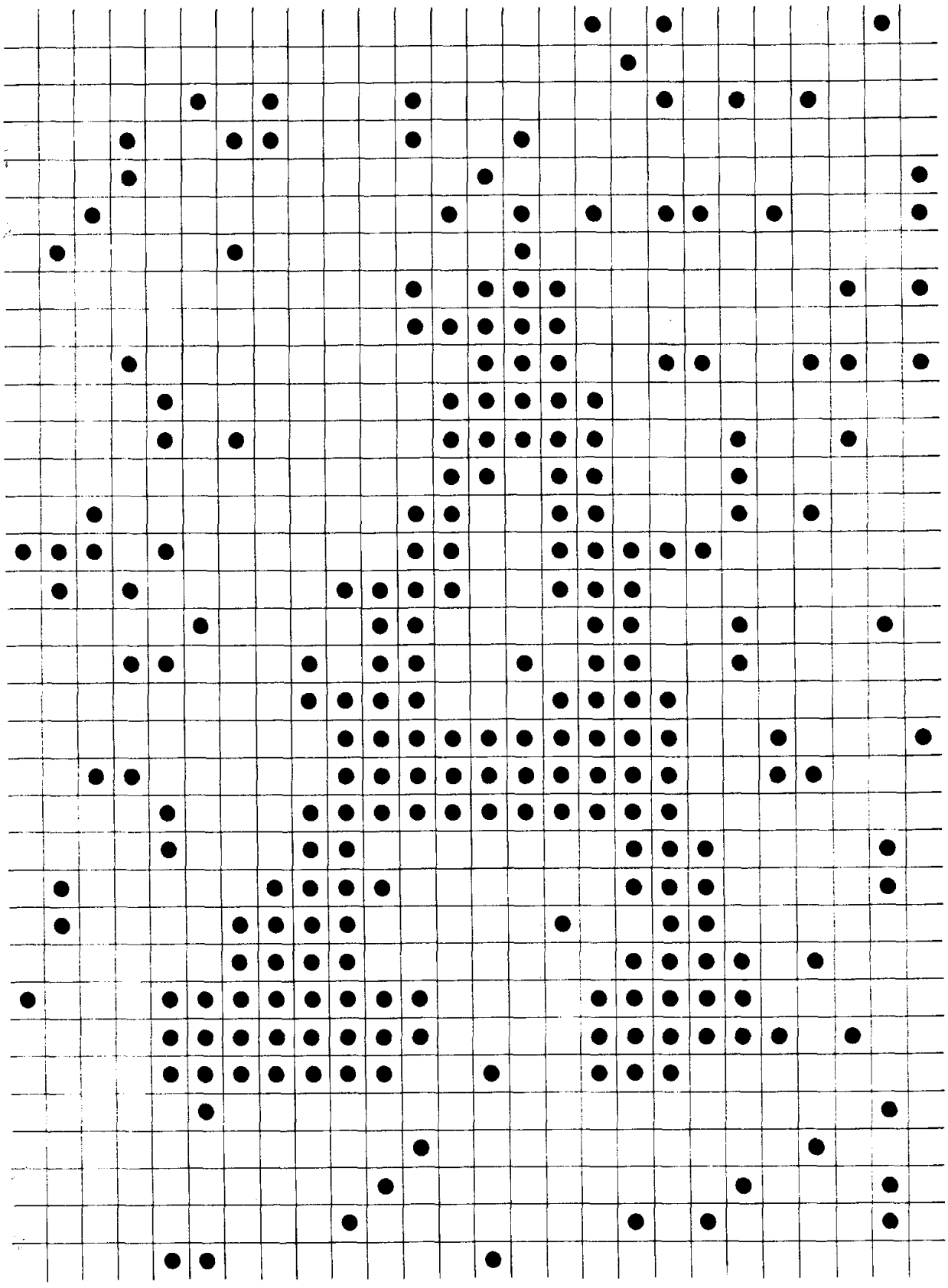
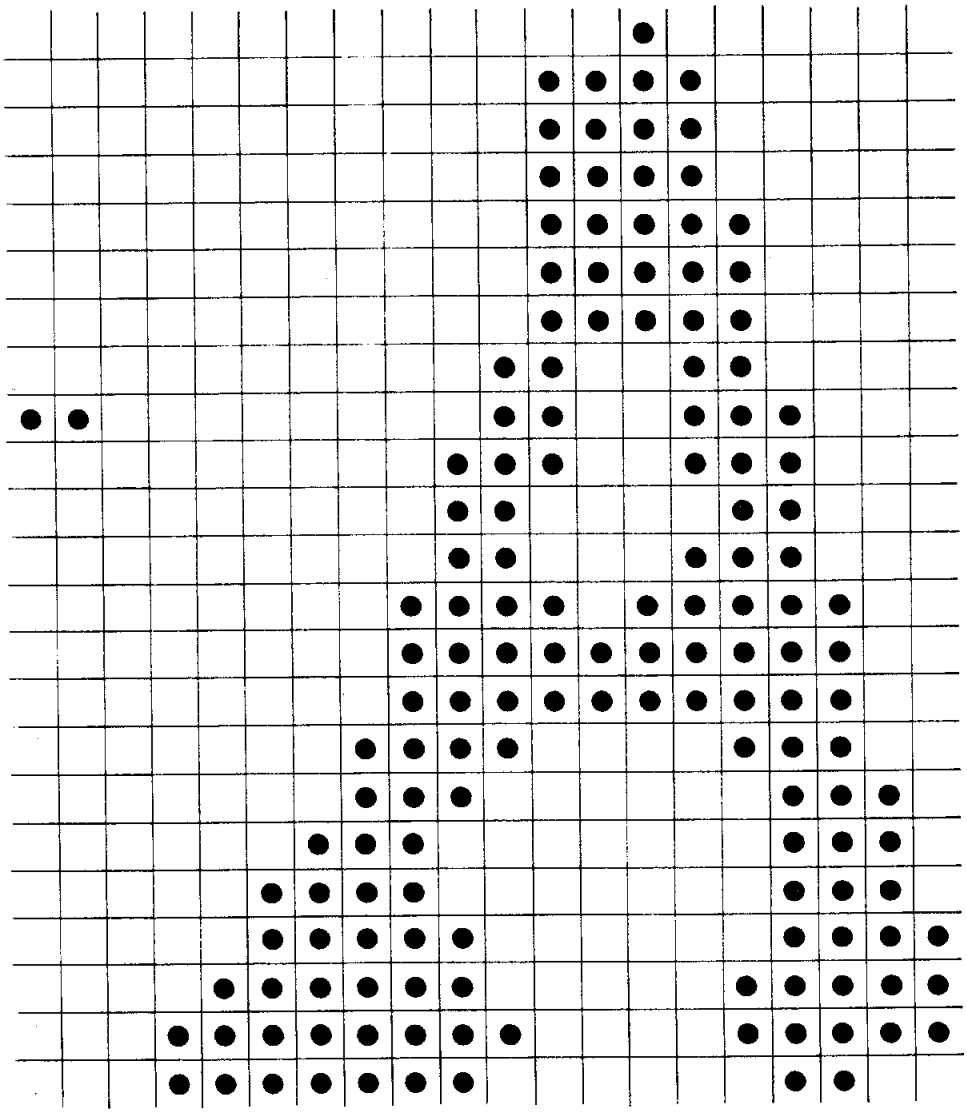


Fig. 8



NOISY LETTER

FIG. 9

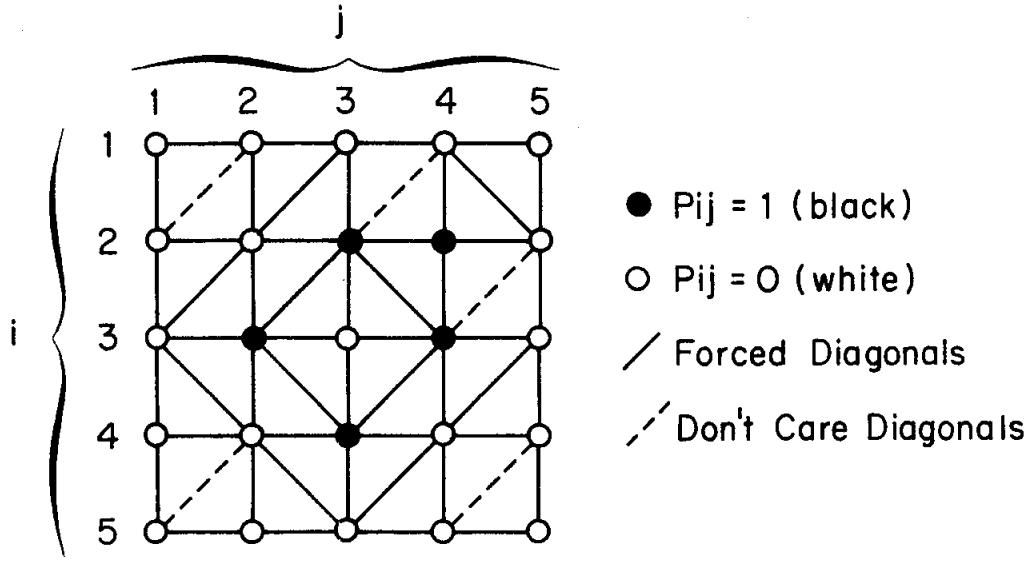


699A10

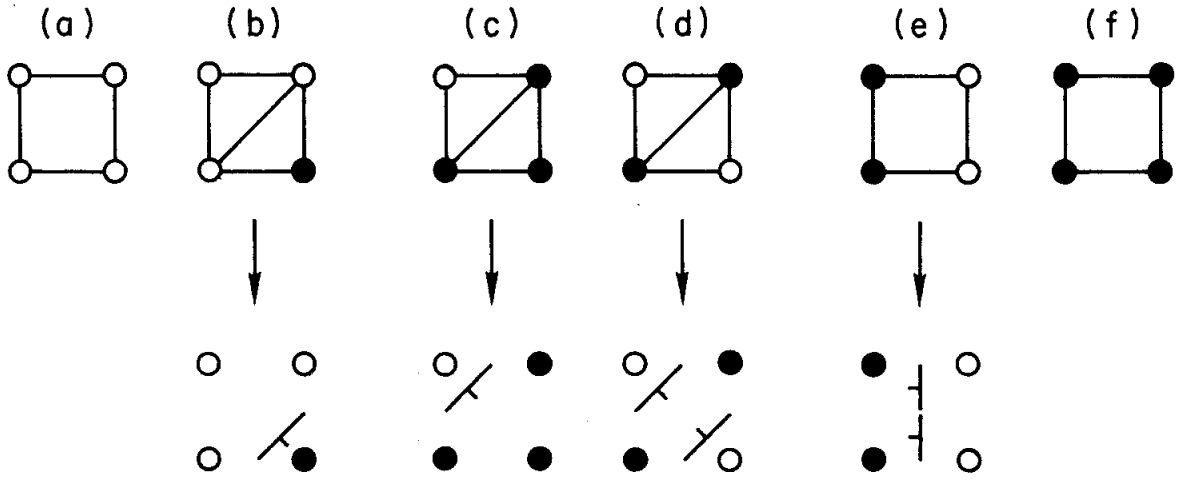
AFTER SIMPLE MAJORITY PRESMOOTHING

FIG. 10

MATRIX PATTERN



TRIANGULATION RULES



CONTOUR OF ABOVE PATTERN

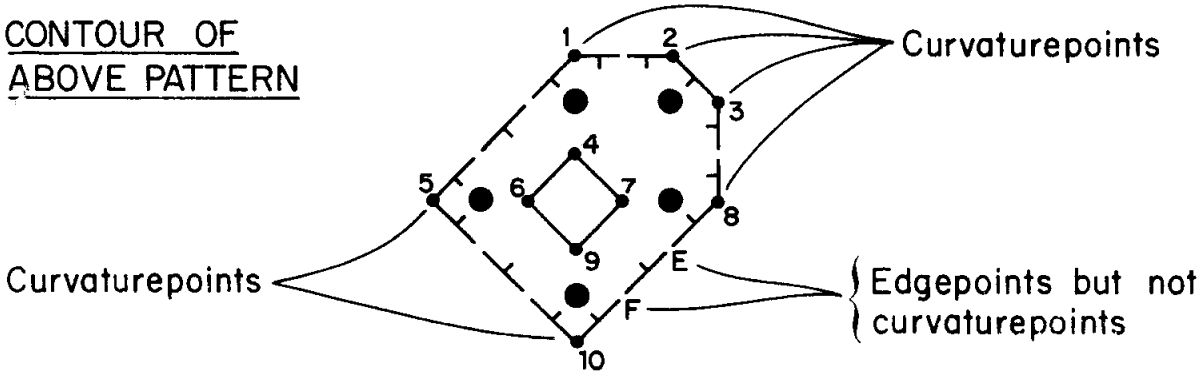
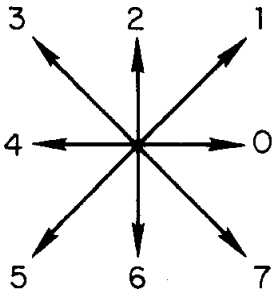


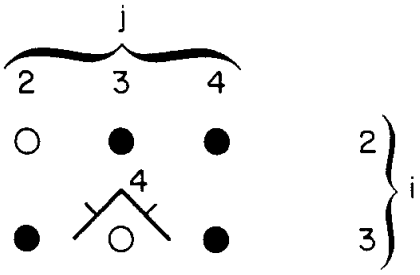
Fig. 11

669A11

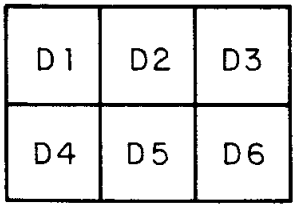
CONTOUR EDGE
DIRECTIONS



DETERMINING
CURVATUREPOINTS
BY (2 x 3) WINDOW

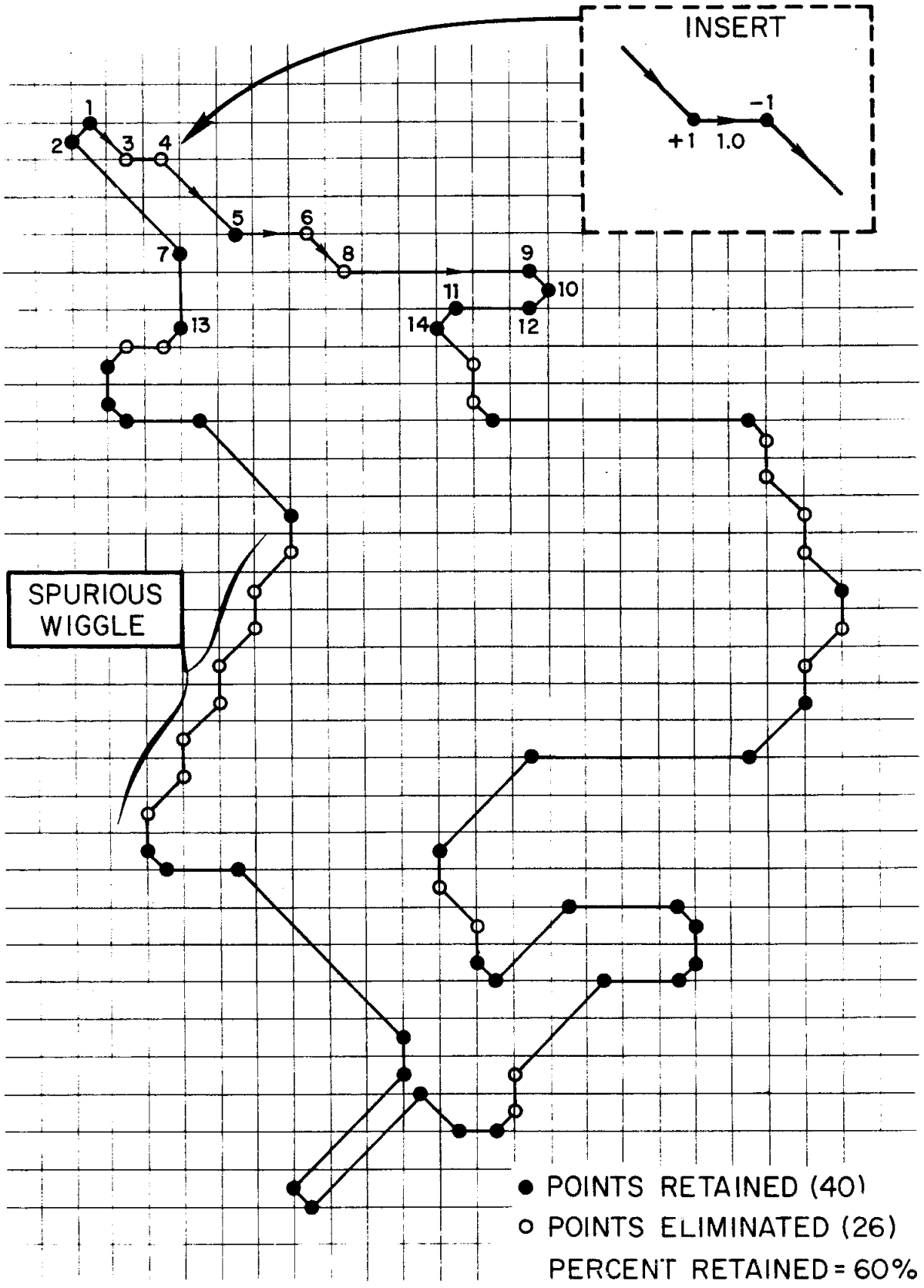


GENERAL
(2 x 3) WINDOW



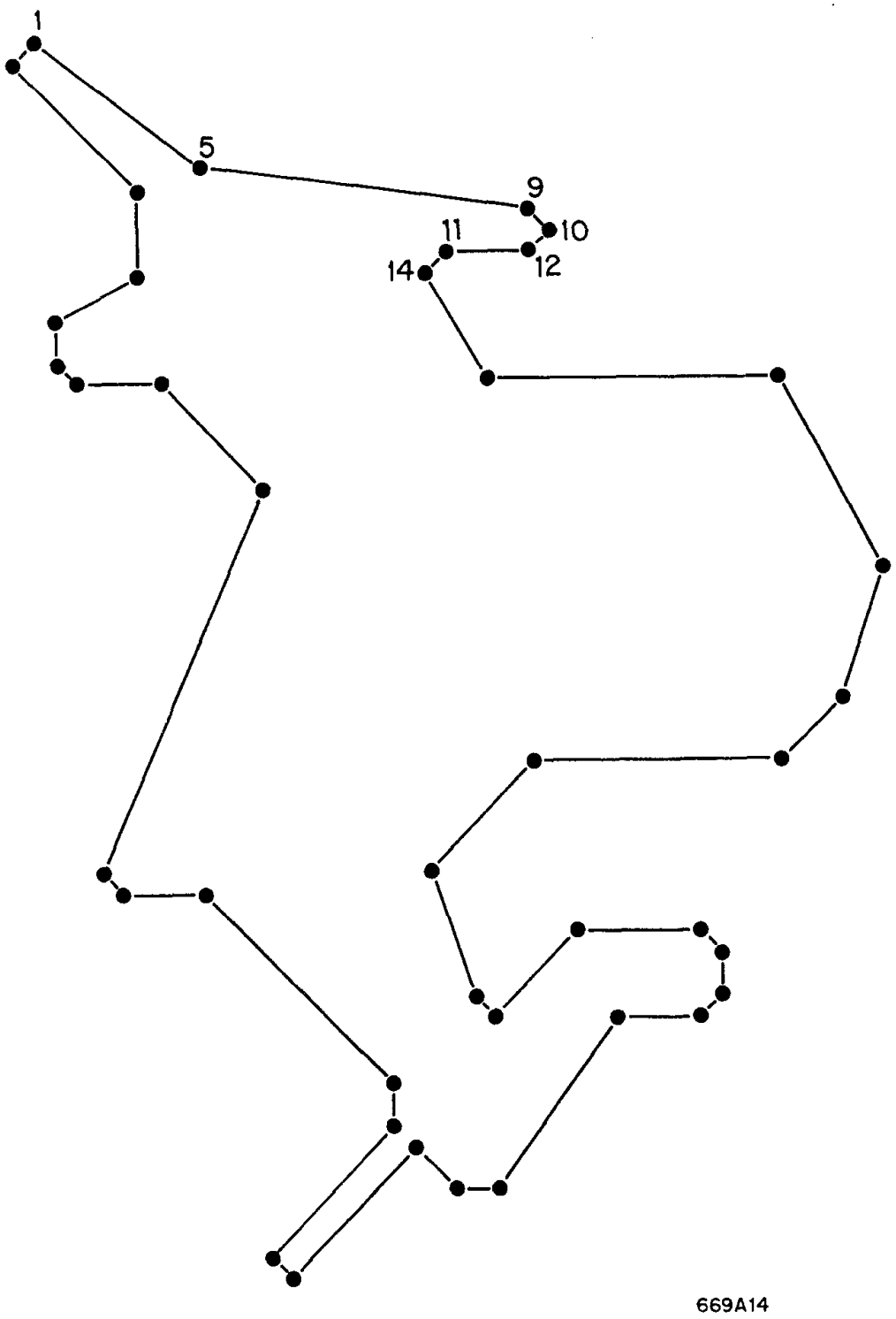
669A12

FIG. 12



669A13

Fig. 13



SMOOTHED EDGE CONTOUR

Fig. 14

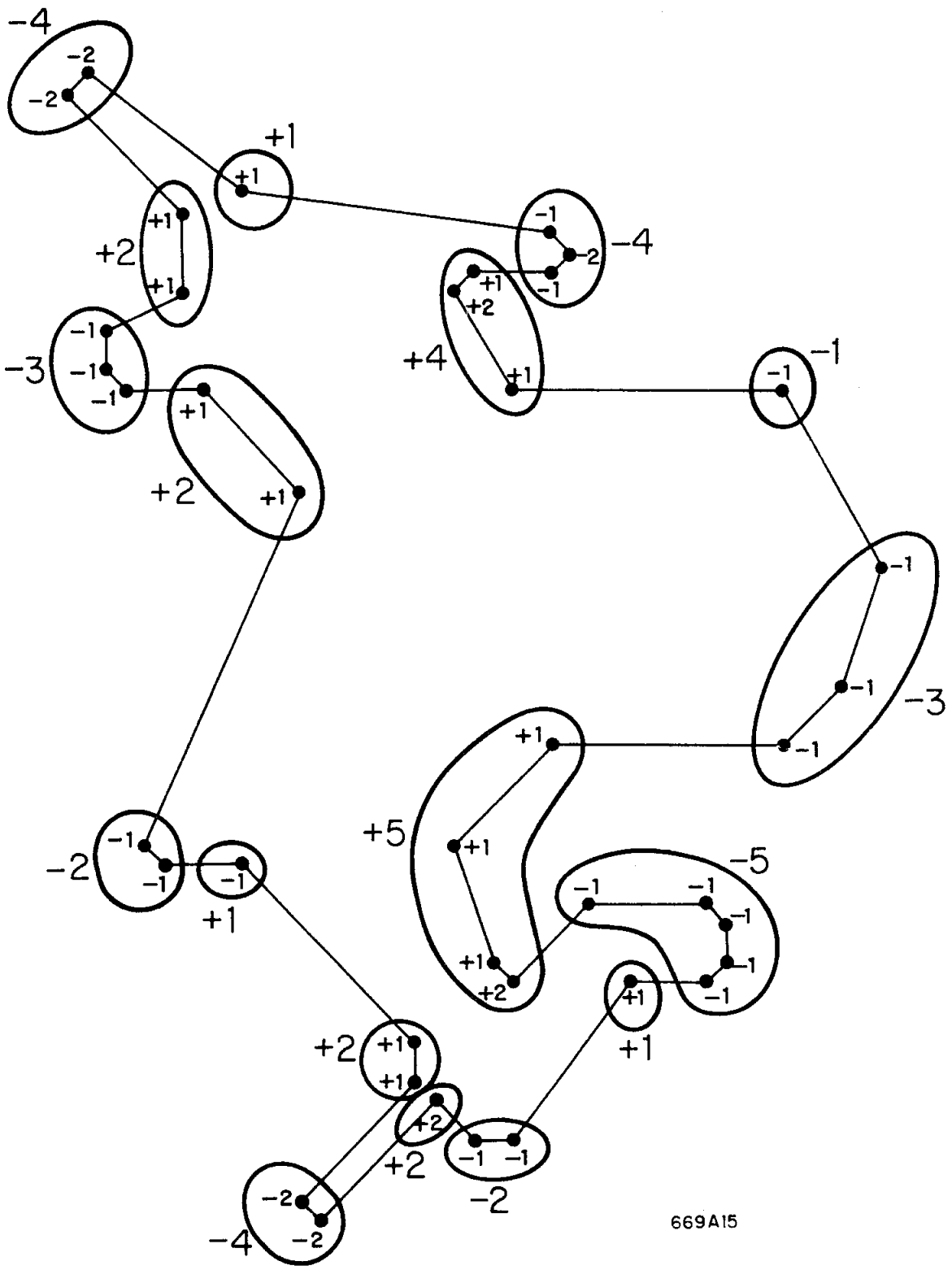


Fig. 15

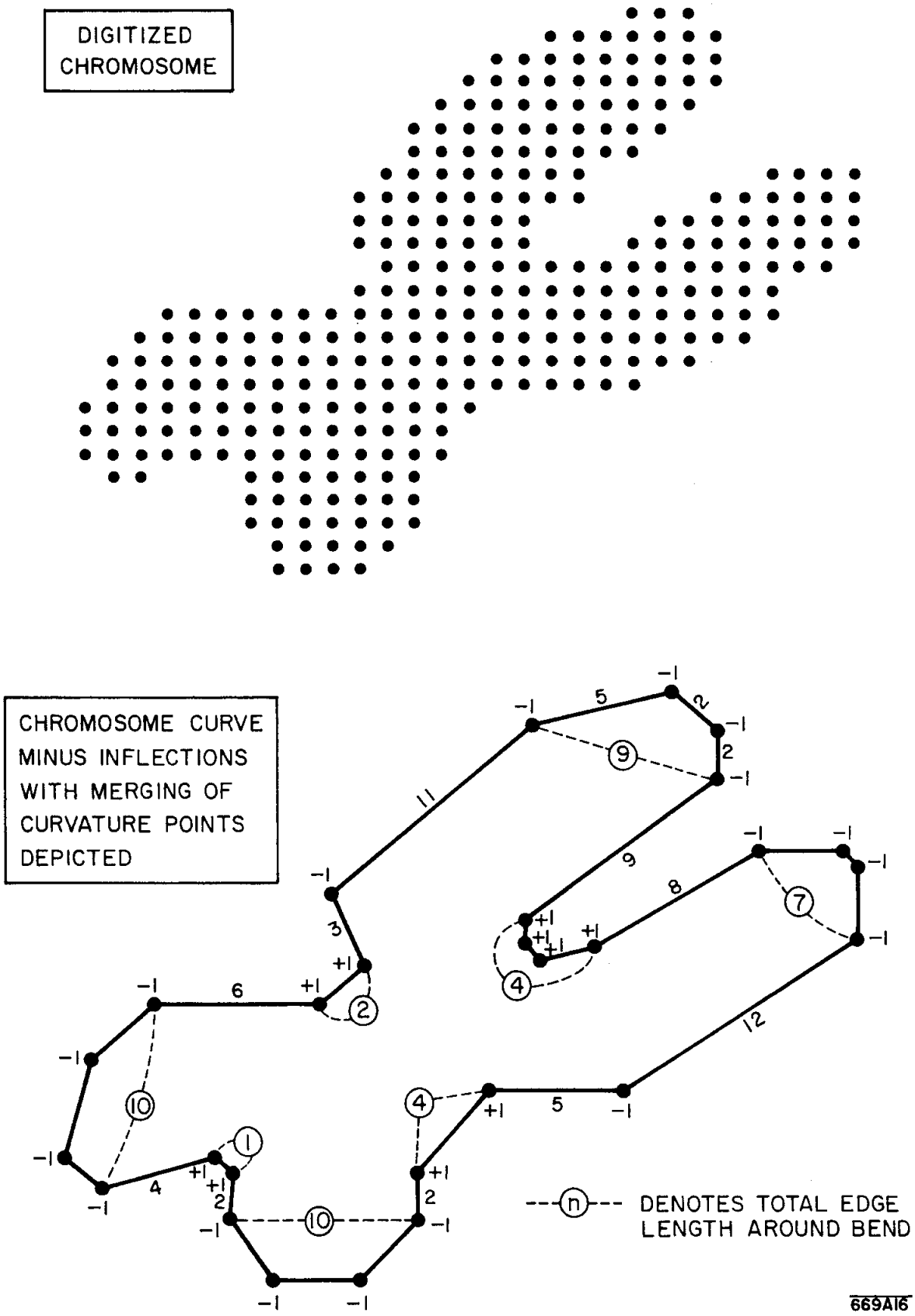

DIGITIZED CHROMOSOME
CHROMOSOME CURVE MINUS INFLECTIONS WITH MERGING OF CURVATURE POINTS DEPICTED
DENOTES TOTAL EDGE LENGTH AROUND BEND
669A16

FIG. 16

-4, 9	-4 [2.25] /	CONVEX END [NORMAL] /
9	9 /	LONG/
+4, 4	+4 [1.0] /	CONCAVE END [ABRUPT] /
8	8 /	LONG /
-4, 7	-4 [1.75]/	CONVEX END [NORMAL] /
12	12 /	EXTRA-LONG/
-1	-1 /	CONVEX SLIGHT/
5	5 /	MEDIUM/
+2, 4	+2 [2.0] /	CONCAVE RIGHT [NORMAL] /
2	2 /	SHORT/
-4, 10	-4 [2.5] /	CONVEX END [NORMAL] /
2	2 /	SHORT/
+2, 1	+2 [.5]/	CONCAVE RIGHT [ABRUPT] /
4	4 /	MEDIUM/
-4, 10	-4 [2.5] /	CONVEX END [NORMAL] /
6	6 /	MEDIUM/
+2, 2	+2 [1.0] /	CONCAVE RIGHT [ABRUPT]/
3	3 /	SHORT/
-1	-1 /	CONVEX SLIGHT/
11	11 /	EXTRA-LONG/

FIG. - 17

< long parallels > ← < long arm > / < concave end > / < long arm >

< short vee > ← < short arm > / < concave right > / < short arm >

< long arm > ← < longe edge > / < convex end > / < long edge >

< short arm > ← < short edge > / < convex end > / < short edge >

< long edge > ← LONG | EXTRA-LONG

< short edge > ← SHORT | MEDIUM

< convex end > ← CONVEX < end phrase >

< concave end > ← CONCAVE < end phrase >

< end phrase > ← END | END [< curve rate >]

< curve rate > ← ABRUPT | NORMAL | GRADUAL

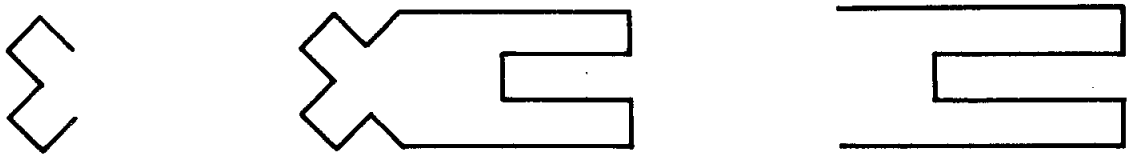
< concave right > ← CONCAVE < right phrase >

< right phrase > ← RIGHT | RIGHT [< curve rate >]

< forward inflection > ← CONVEX SLIGHT / < short edge > / < concave right >

< backward inflection > ← < concave right > / < short edge > / CONVEX SLIGHT

< chromosome > ← (< long parallels > / < forward inflection > / < short vee > / < backward inflection >)



< short vee > < chromosome > < long parallels >

669A17

FIGURE 18

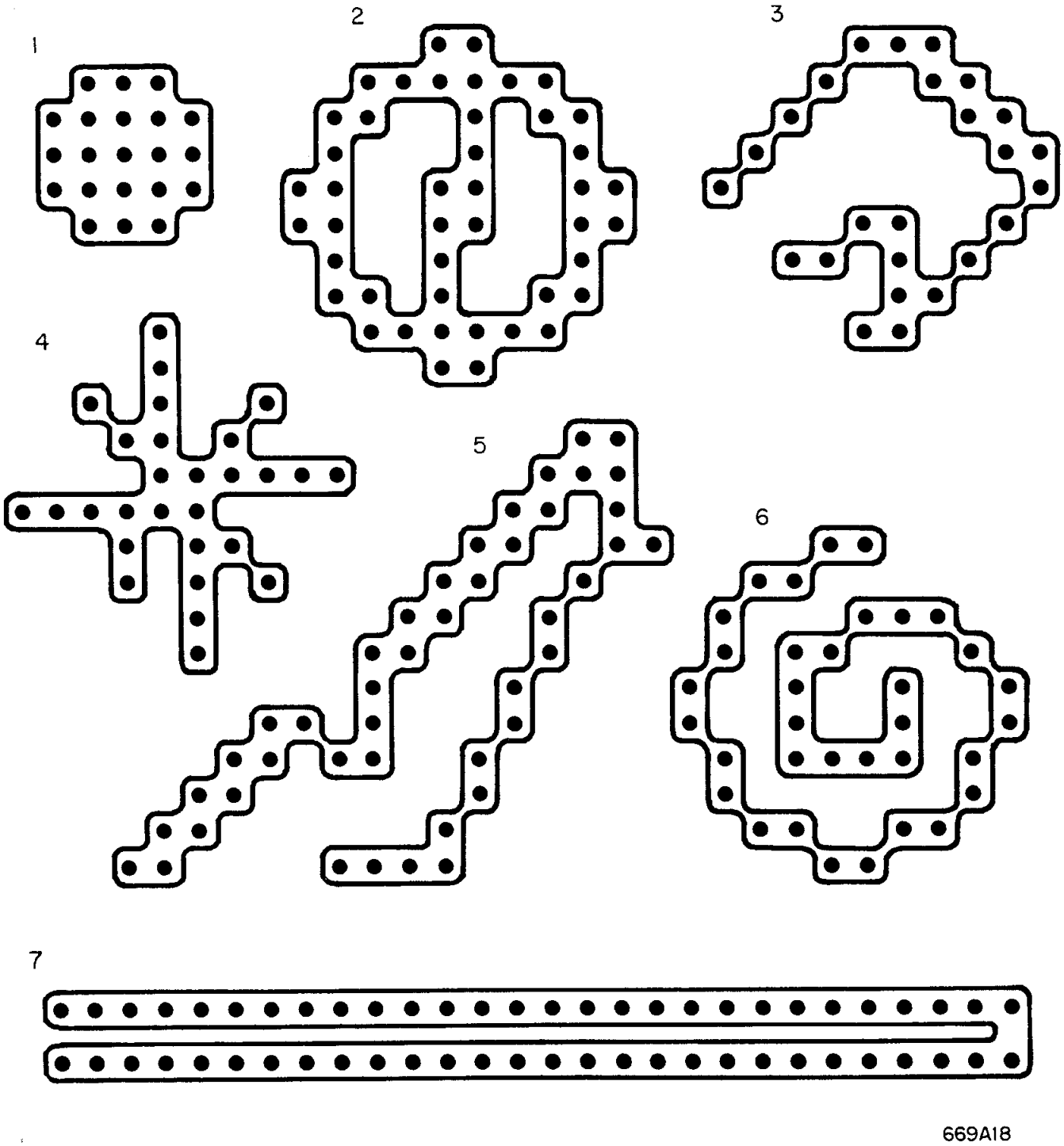
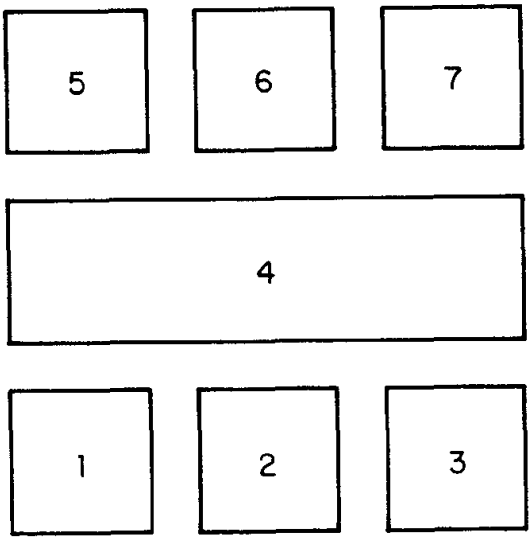


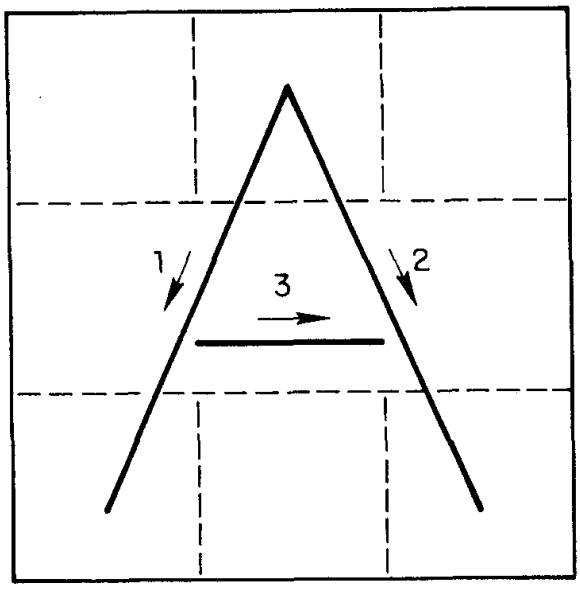
FIG. 19

SIGNATURES	OBJECTS 1	2	3	4	5	6	7
AREA	20.5	44.5	24.5	27.5	43.5	33.5	56.5
PERIMETER	16.5	62	51	51.1	77.2	77.0	114
CONNEXITY	1	3	1	1	1	1	1
STRINGINESS	2.5	1.4	.96	1.08	1.13	.87	.99
OBLONGNESS	1.0	1.0	1.11	1.0	2.7	1.08	9.3
CURVATURE RATE	.5	.4	.4	.8	.33	.42	.14
WIGGLINESS (UNNORMALIZED)	0	0	4	16	6	4	2
WIGGLINESS (NORMALIZED)	0	0	.08	.32	.08	.05	.02
ENDS (CONVEX)	0	0	2	8	2	2	2
ENDS (CONCAVE)	0	2	0	8	1	0	1
MAXIMUM EDGE LENGTH	2.1	5.4	6.4	3.0	13.2	7.2	27
MAX. BEND GROUP (CONVEX)	8	8	11	4	10	20	12
MAX. BEND GROUP (CONCAVE)	0	8	6	3	6	12	4

FIG. 20



SUBDIVISION OF
CHARACTER AREA

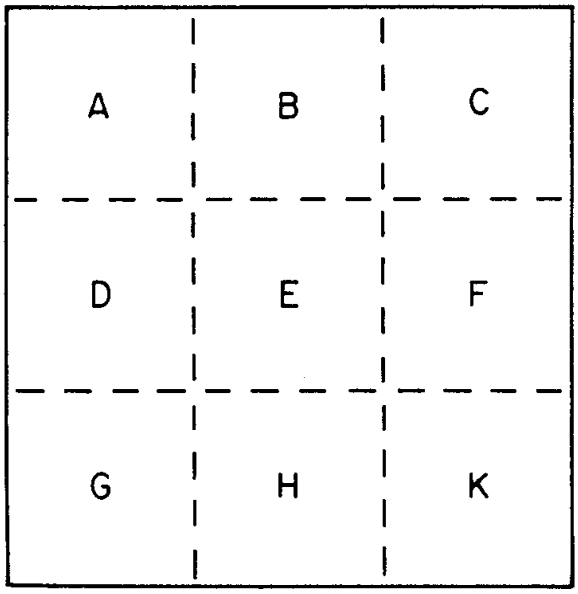


CHARACTER "A"
AS PRINTED IN
THREE STROKES

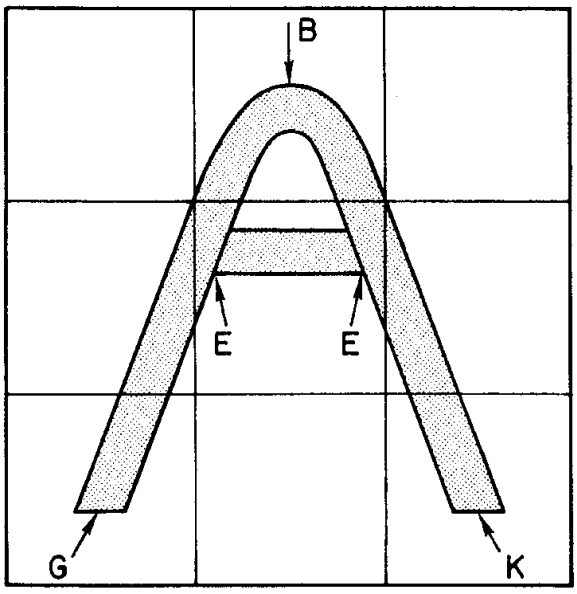
TRACE (A) = [6410643040]

699AI9

Fig. 21



AREA SUBDIVISION
AND LABELS

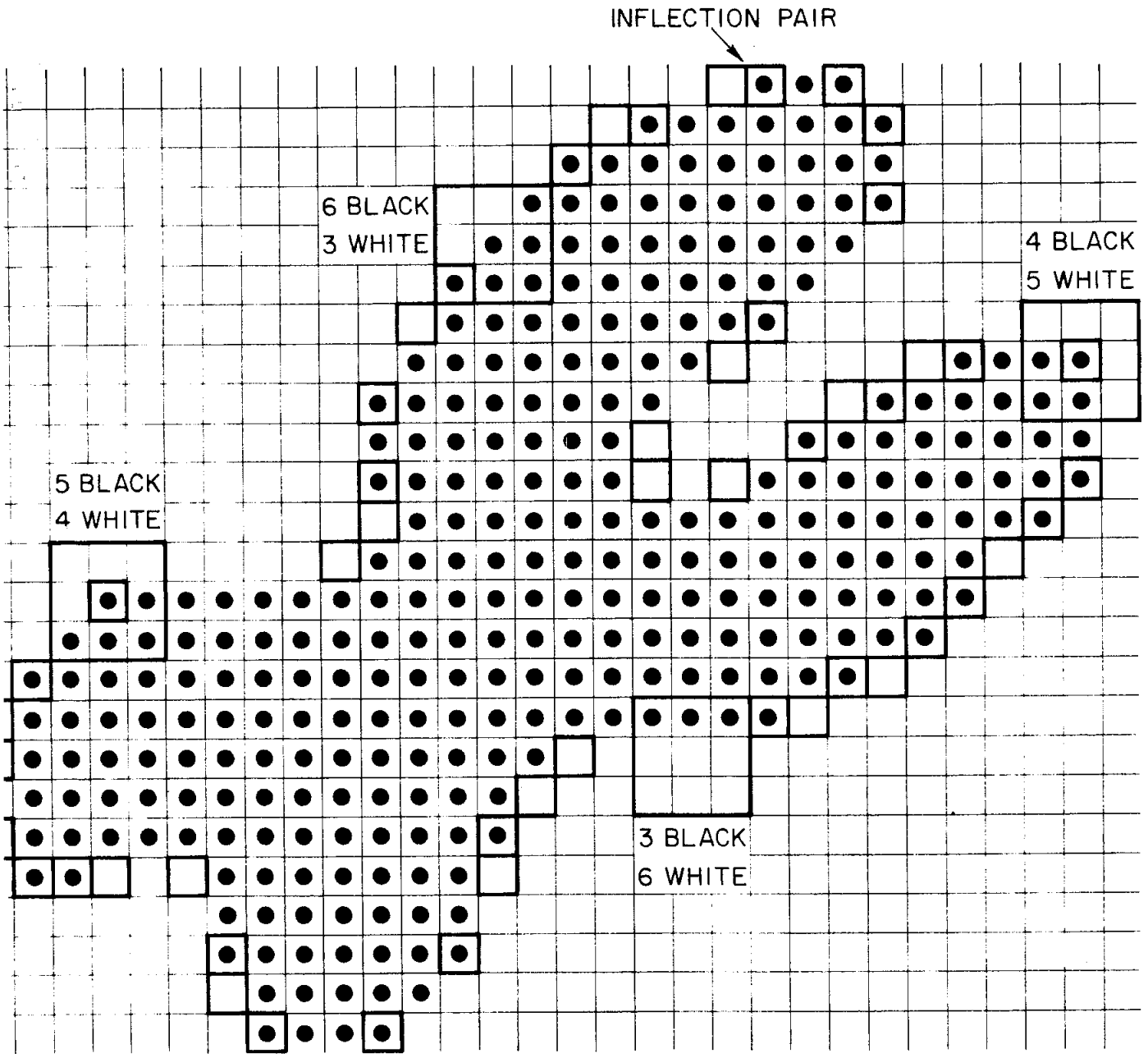


CHARACTER "A"

OUTER CURVE (A) = (BKEG)

669A20

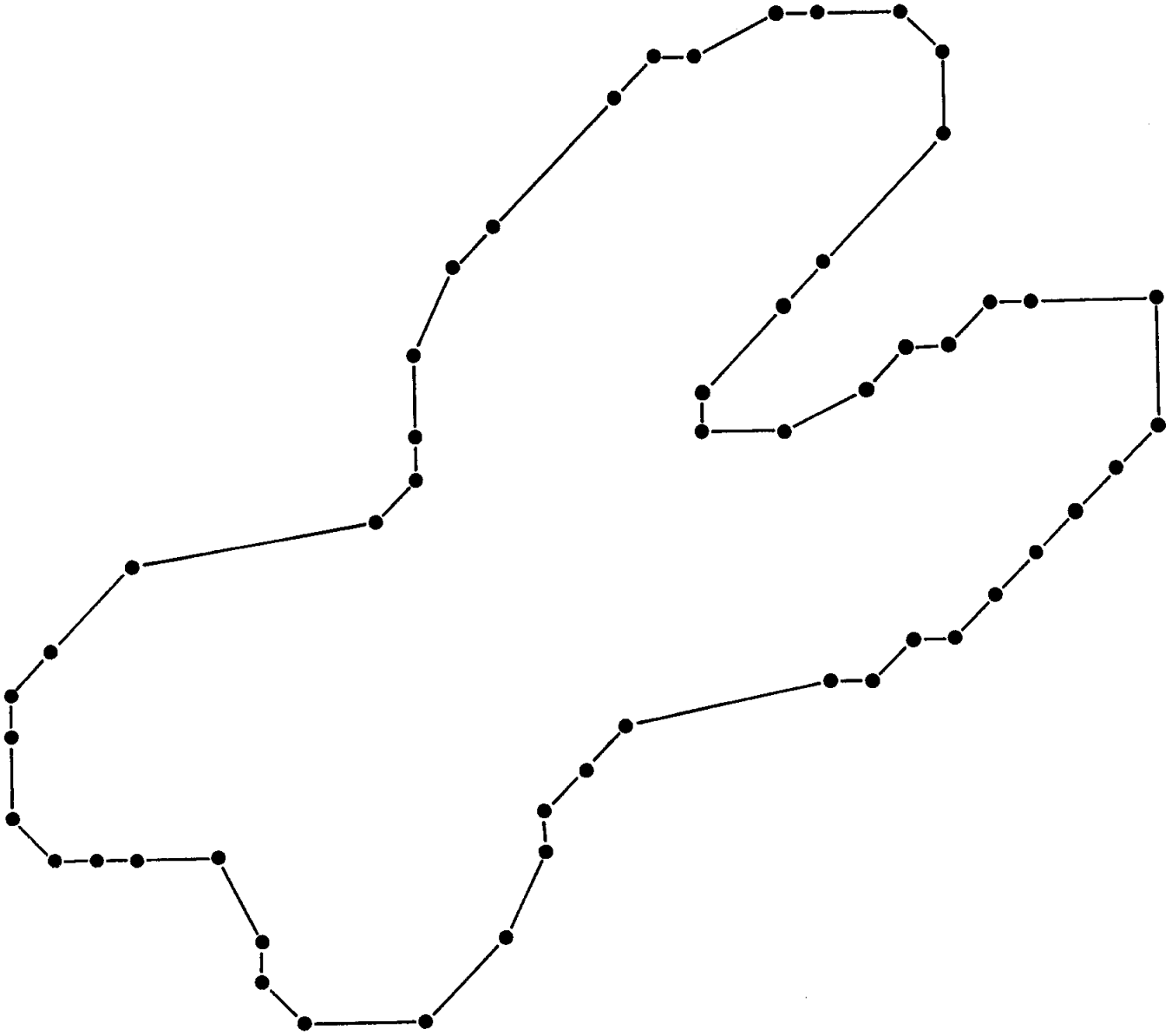
Fig. 22



□ OR ◉ DENOTES 5-4 SPLIT IN 3 X 3 WINDOW OF WHICH GIVEN SQUARE IS CENTER

669A21

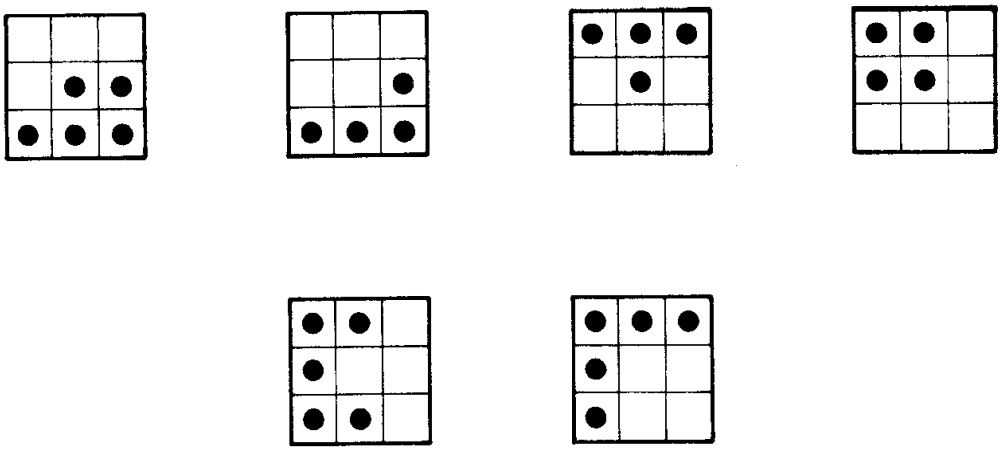
FIG. 23



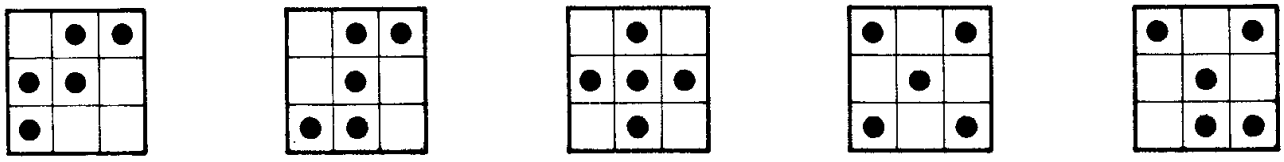
669A22

5-4 POINTS LINKED TOGETHER INTO POLYGON

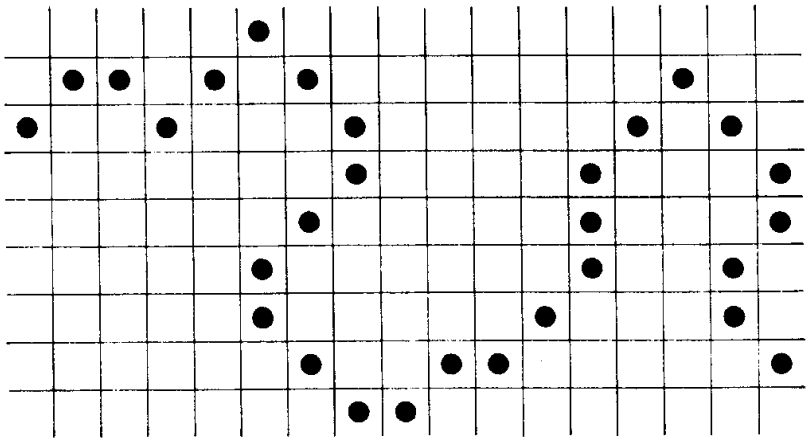
Fig. 24



MOST COMMON TYPES OF 5-4 POINTS



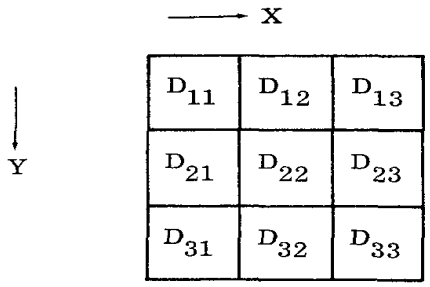
IRREGULAR 5-4 POINTS



669A23

WIGGLY PATTERN WITH NO 5-4 POINTS

FIG. 25



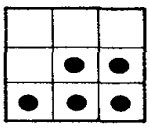
3 × 3 Window

$$SUM = \sum_{i=1}^{3} \sum_{j=1}^{3} D_{ij}$$

$$CENTER = D_{22}$$

$$DX = \sum_{i=1}^{3} D_{i3} - \sum_{i=1}^{3} D_{i1}$$

$$DY = \sum_{j=1}^{3} D_{3j} - \sum_{j=1}^{3} D_{1j}$$



SUM = 5

CENTER = 1

DX = 1 ; DY = 3

$\sigma_1 = \sigma_1(DX, DY) = \sigma_1(1, 3) = 0$

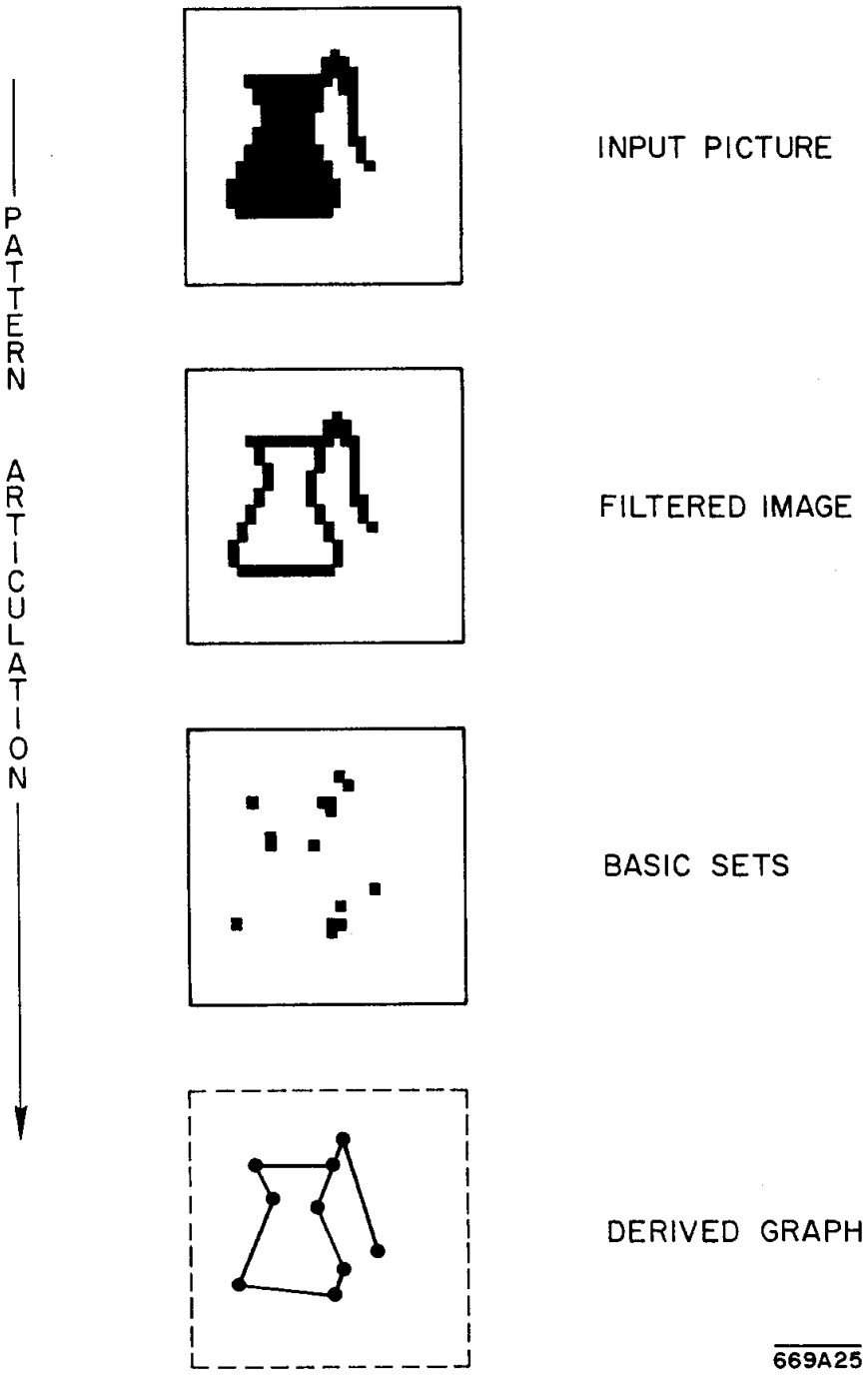
$|\Delta\theta| = 1$ (SINCE SUM + CENTER = $6 \neq 5$)

SIGN $(\Delta\theta) = -1$ (CENTER = 1)

$\theta_1 = \sigma_1 + 1 = 1$ (SINCE $\Delta\theta = -1$)

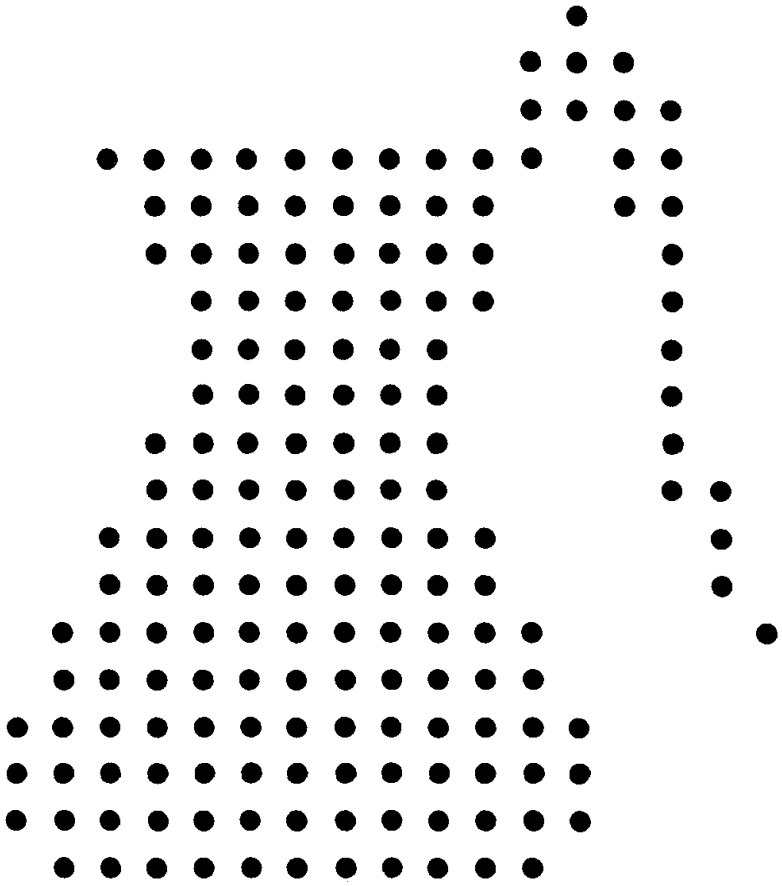
669A24

Figure 26



VISUAL PATTERN ARTICULATION

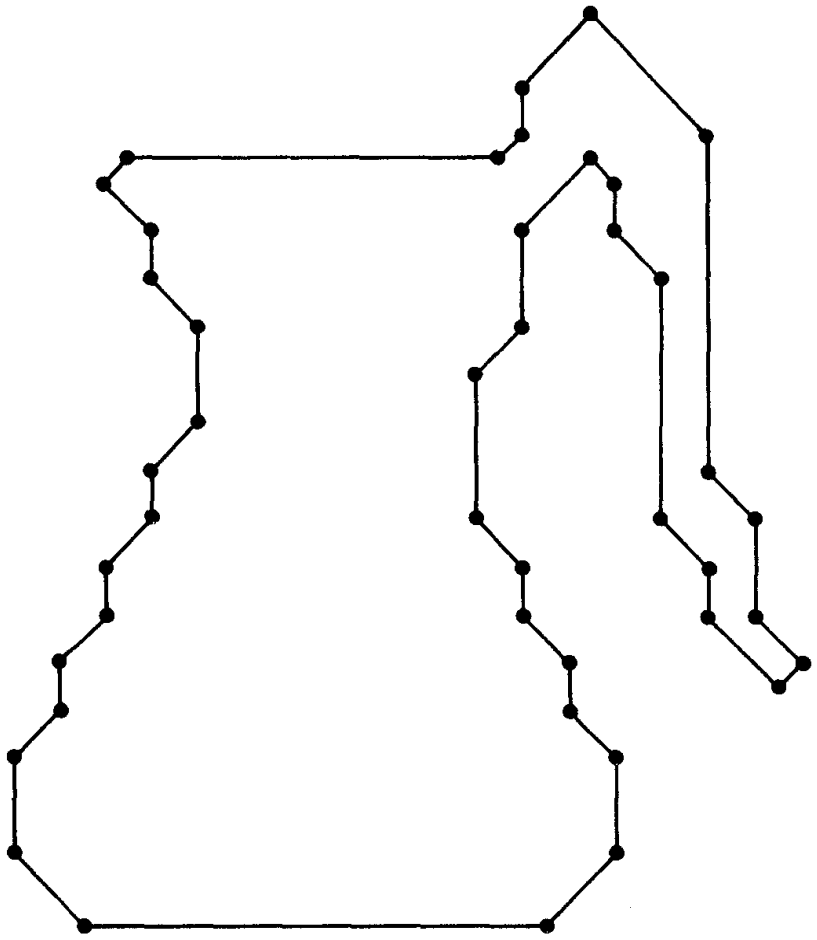
FIG. 27



INPUT PICTURE

669A26

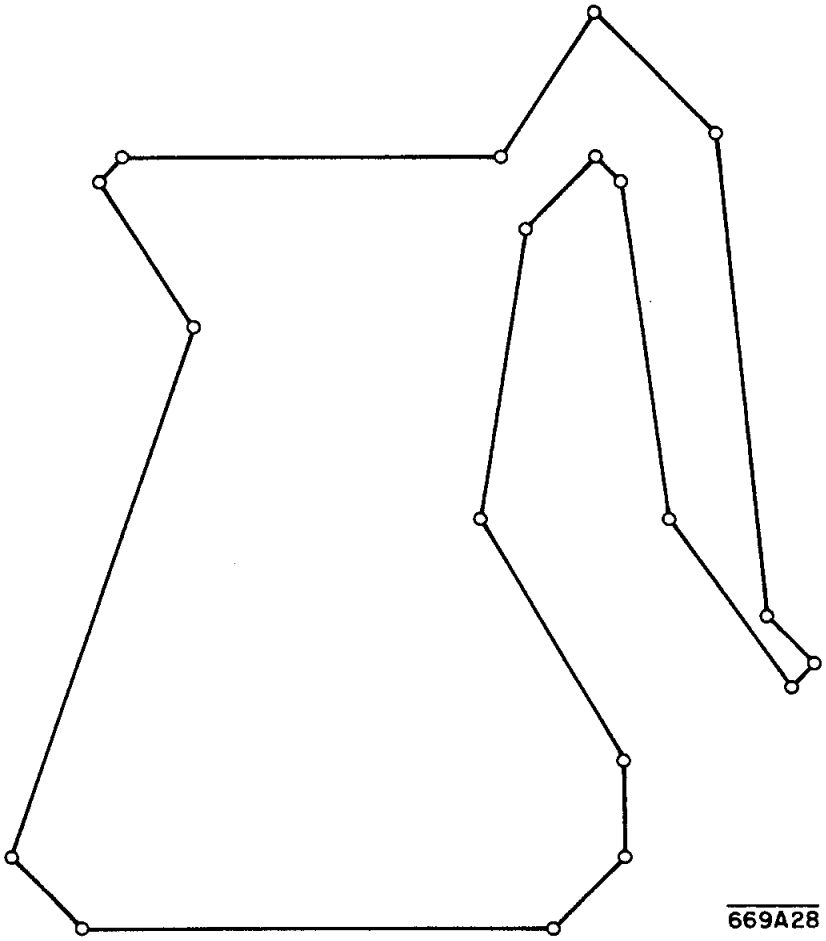
FIG. 28



669A27

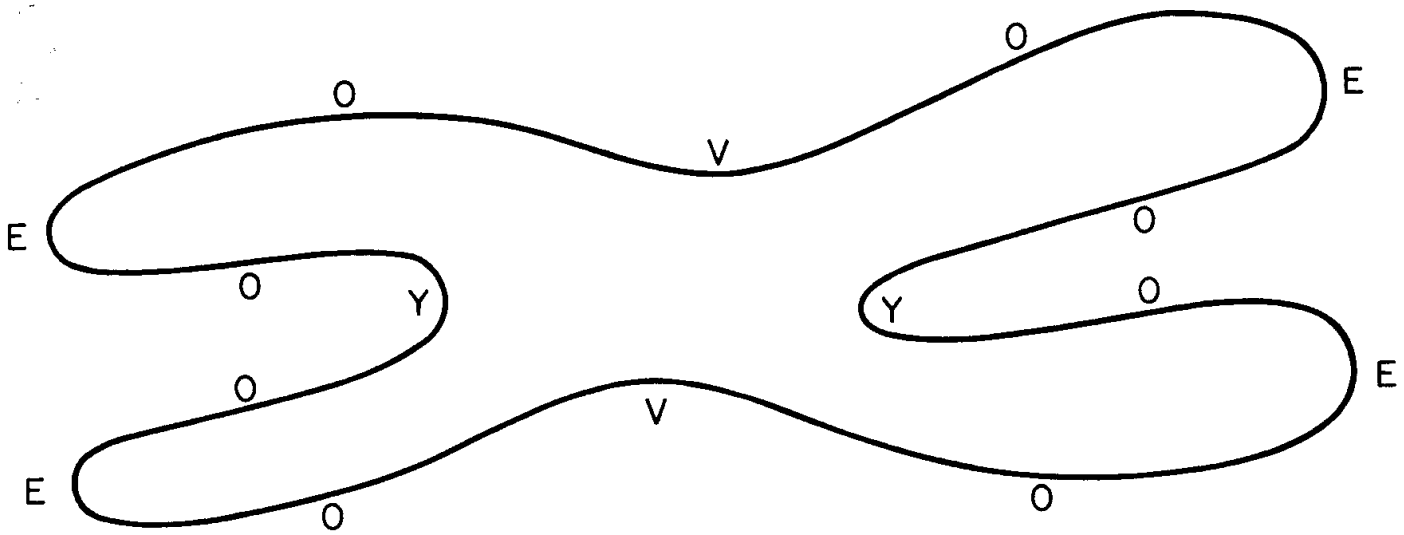
STRUCTURAL DESCRIPTION

FIG. 29

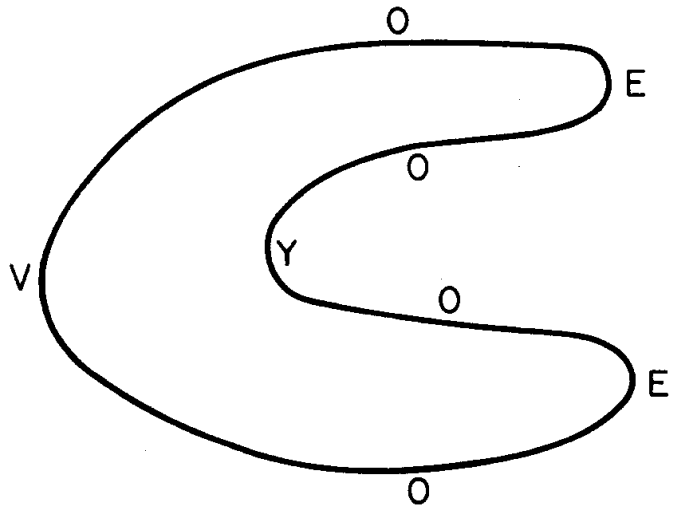


DERIVED GRAPH AFTER DELETION
OF SPURIOUS INFLECTIONS

FIG. 30



[O E O V O E O Y O E O V O E O Y]

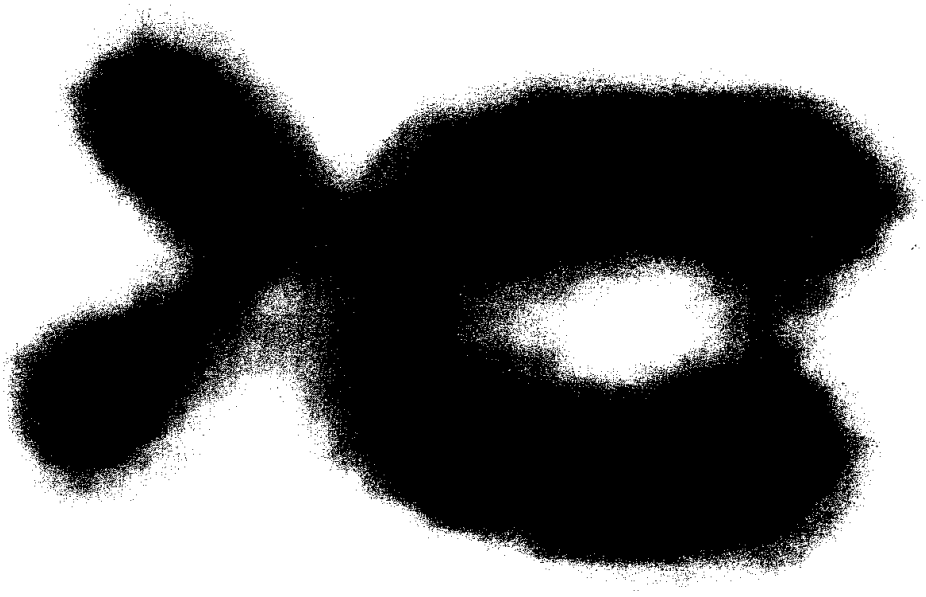


[O E O Y O E O V]

669A29

Fig. 31

ORIGINAL PHOTOGRAPH



669A31

FIG. 32 a

DIGITIZED VERSION

```
                1
          2 3 4 2 2 1
        2 3 4 4 5 4 3 1                       1 3 2 2 3 3 3 3 2 2
        2 4 5 5 5 5 4 2 1     1 1 3 4 4 5 5 5 5 5 5 5 5 4 4 2
      1 4 4 5 5 5 4 3 2 1 2 4 5 5 5 6 6 5 5 5 6 6 5 5 4 1
      1 3 4 5 5 5 4 4 3 2 4 4 5 5 6 6 6 6 6 6 6 6 6 6 5 4 2
        1 3 4 4 5 5 4 3 4 4 5 5 6 6 6 6 6 6 6 6 6 6 6 5 5 3
          1 3 4 4 4 4 4 5 5 5 6 6 6 6 6 6 5 6 5 5 6 6 6 5 4 2
        1 1 3 3 4 4 4 4 5 5 5 5 5 5 5 4 4 4 4 4 5 5 5 5 4 2
      1 1 2 3 4 4 4 3 4 4 5 5 5 4 4 4 4 2 2 2 2 3 4 4 4 2 1
    1 2 4 4 4 4 4 4 4 4 5 5 5 5 4 4 3 3 1 1 1 2 3 4 3 3 2
  2 4 4 4 5 5 4 4 3 4 4 5 5 5 5 4 4 4 3 2 2 2 2 3 4 4 2 1
  3 4 4 5 5 5 4 4 3 3 4 5 5 6 5 5 5 4 4 4 4 4 4 5 5 5 4 2
  3 4 5 5 5 5 4 2 1 2 4 5 5 6 6 6 6 5 5 5 5 5 6 6 6 6 5 4 2
  1 3 4 4 5 4 3 1 1 2 4 5 5 5 6 6 6 6 6 6 5 6 6 6 6 5 4
    2 3 4 3 2 1     1 3 4 5 5 6 6 6 6 6 6 6 6 6 6 5 5 4
        1           3 4 4 5 5 6 5 5 6 6 6 6 5 5 5 4 2
                    1 2 3 4 4 5 5 5 5 5 4 4 4 3 2
                      1 3 2 4 4 3 3 2 1 1
```

669A31

FIG. 32 b

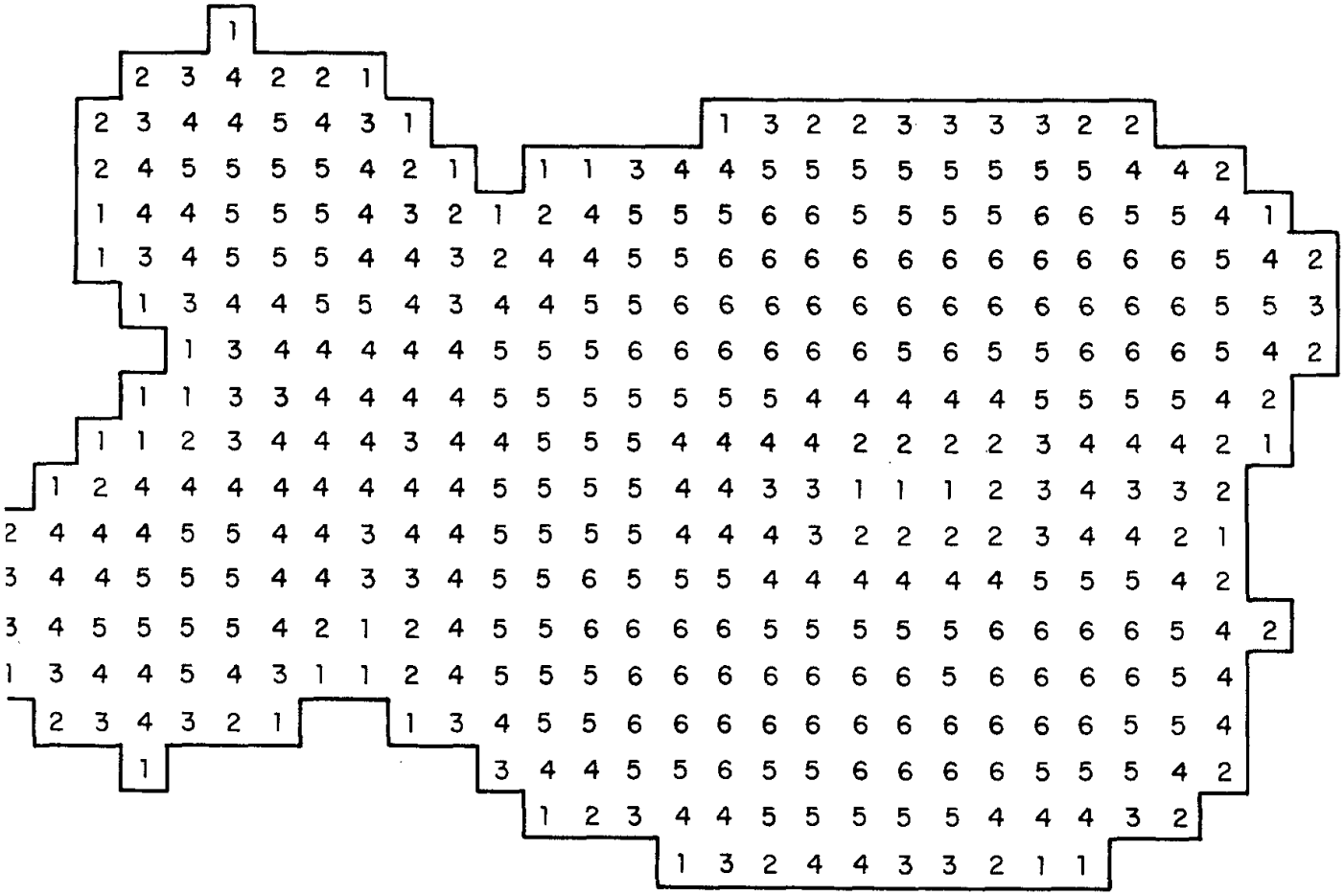


Fig. 33

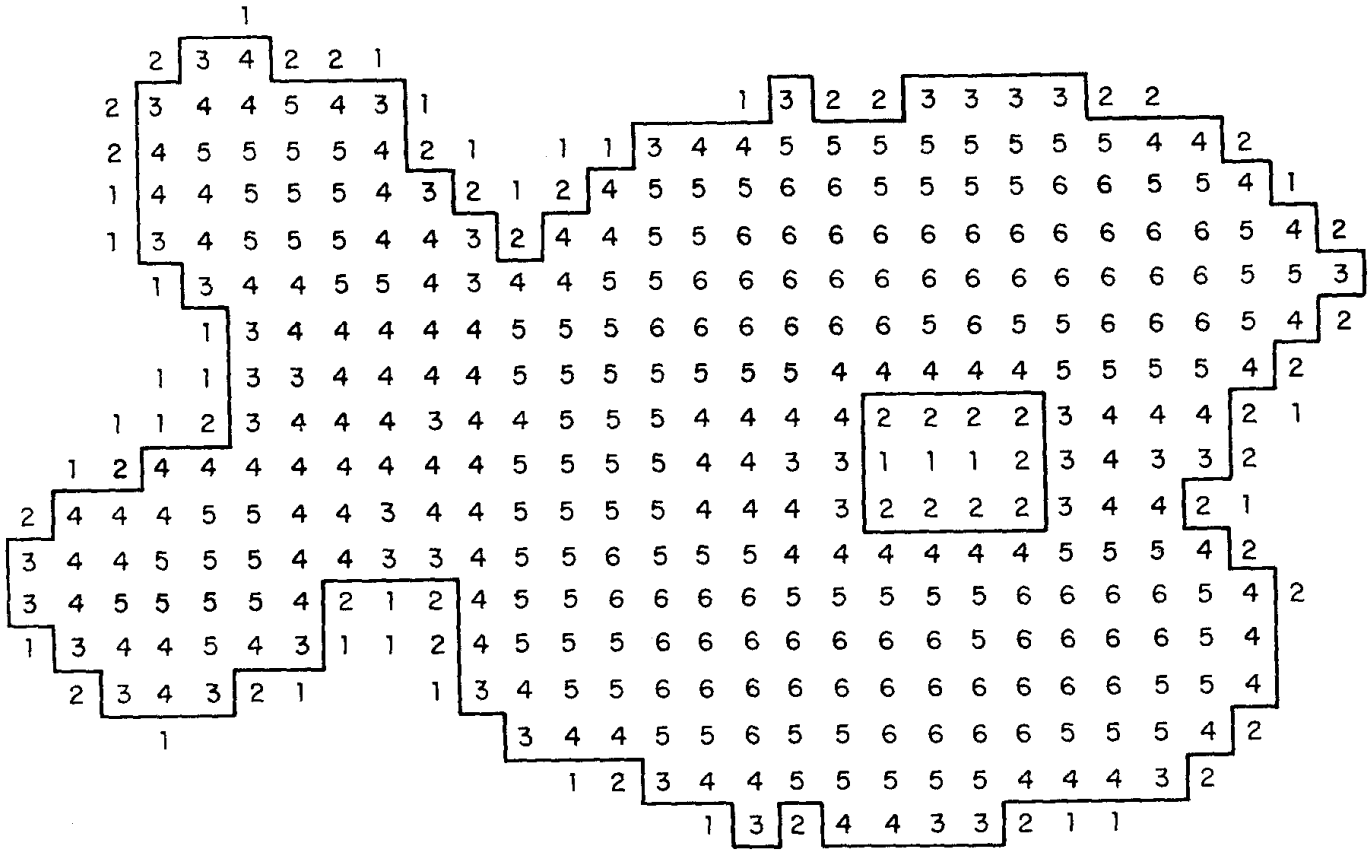


Fig. 34

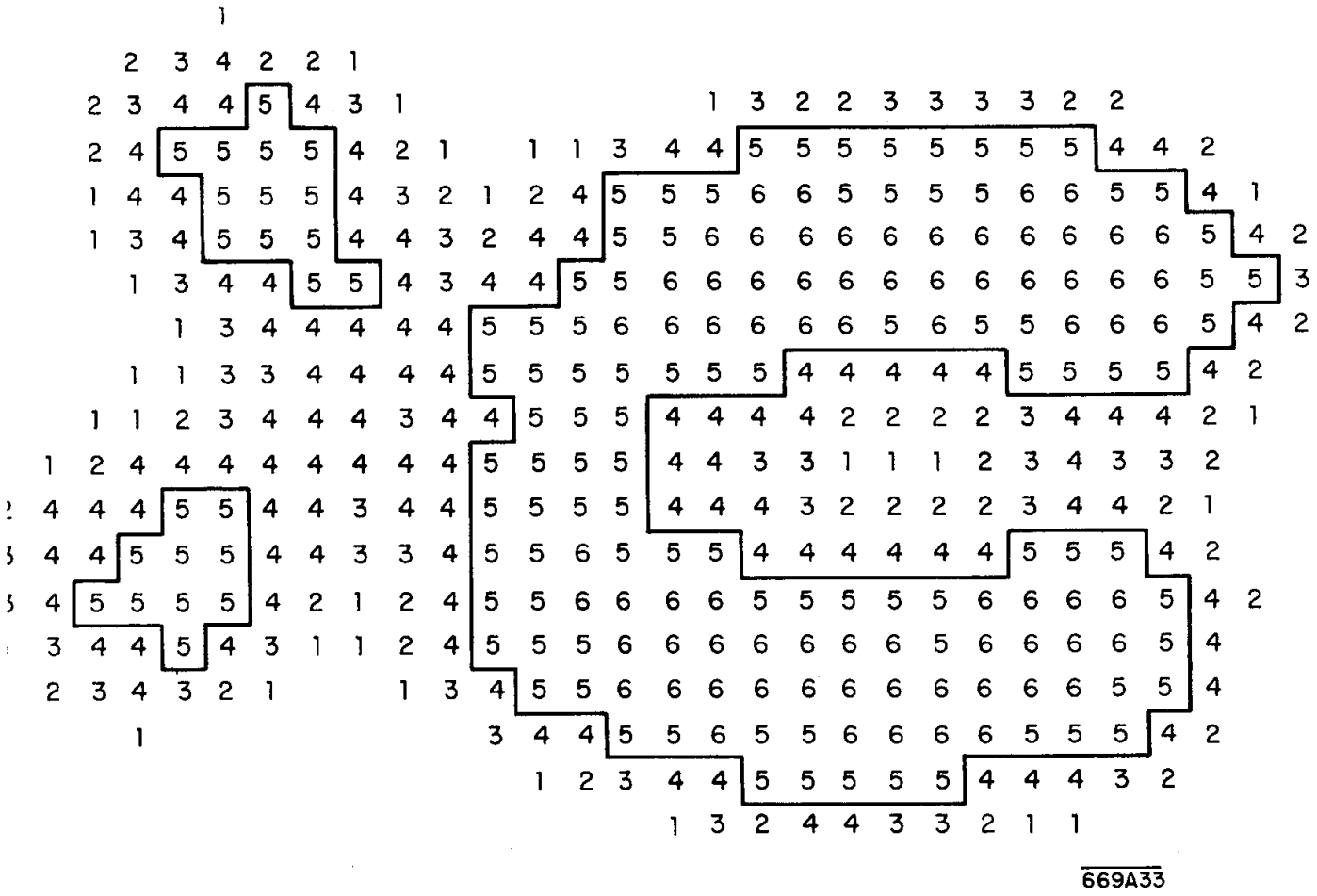


Fig. 35

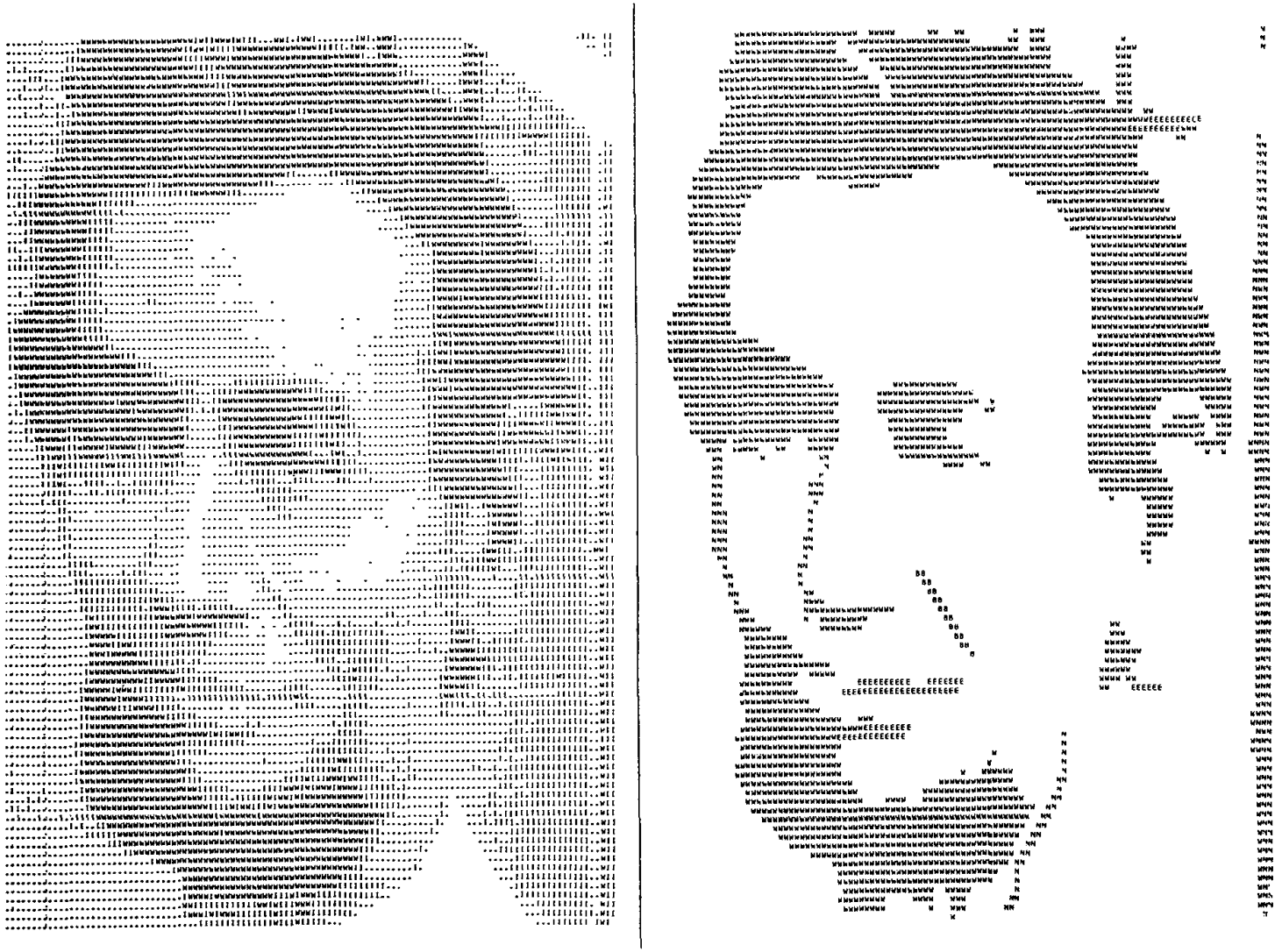
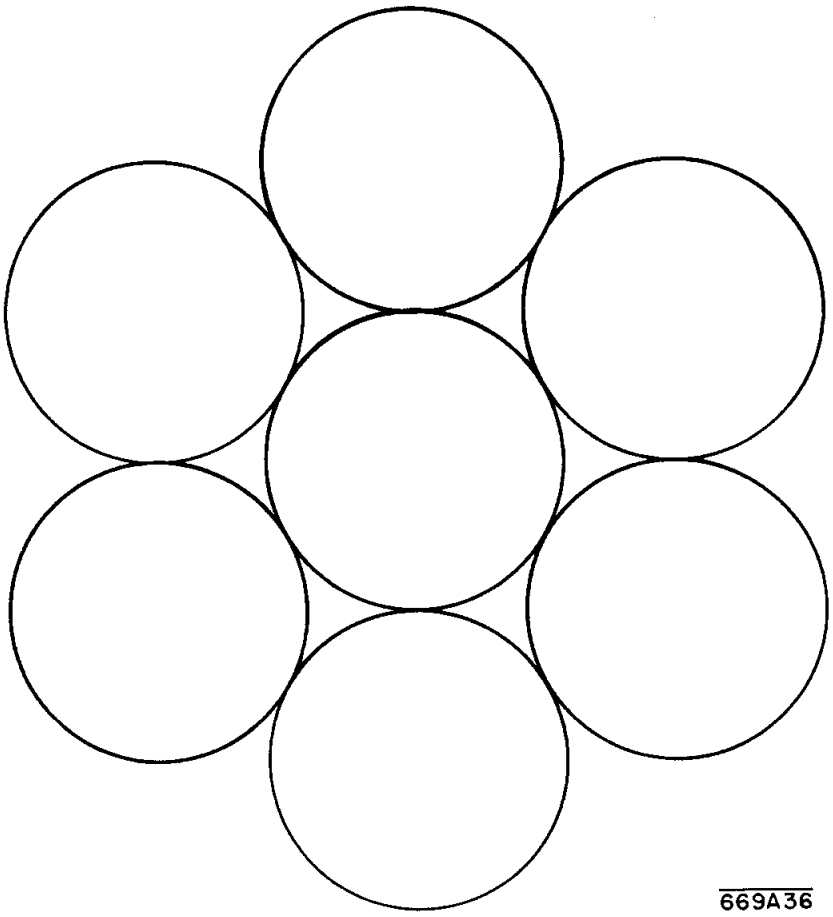


Fig. 36

(a) Dr. Arthur H. C. von Hochstetter, Head of the Department of Anatomy, Faculty of Medicine, University of Western Ontario, London, Province of Ontario, has succeeded in micro-photographing the structure of the cornea of the human eyeball. Dr. von Hochstetter says, "The cornea, the most regular shaped structure of the body, consists of interwoven bands of unyielding collagenic fibres that are arranged in many lamellae. This fibrous architecture includes a dense meshwork of cells. In these specimens the injection of a mixture of India-ink and Gelatin into the Cornea has partly filled the space-system in which the cells lie. The figures formed by the case indicate that the cells are arranged in a three-dimensional geodesic grid of the tetrahedronal type. The various hexagonal or polygonal figures (holes of the grid) are from 10 to 100 micron in diameter."

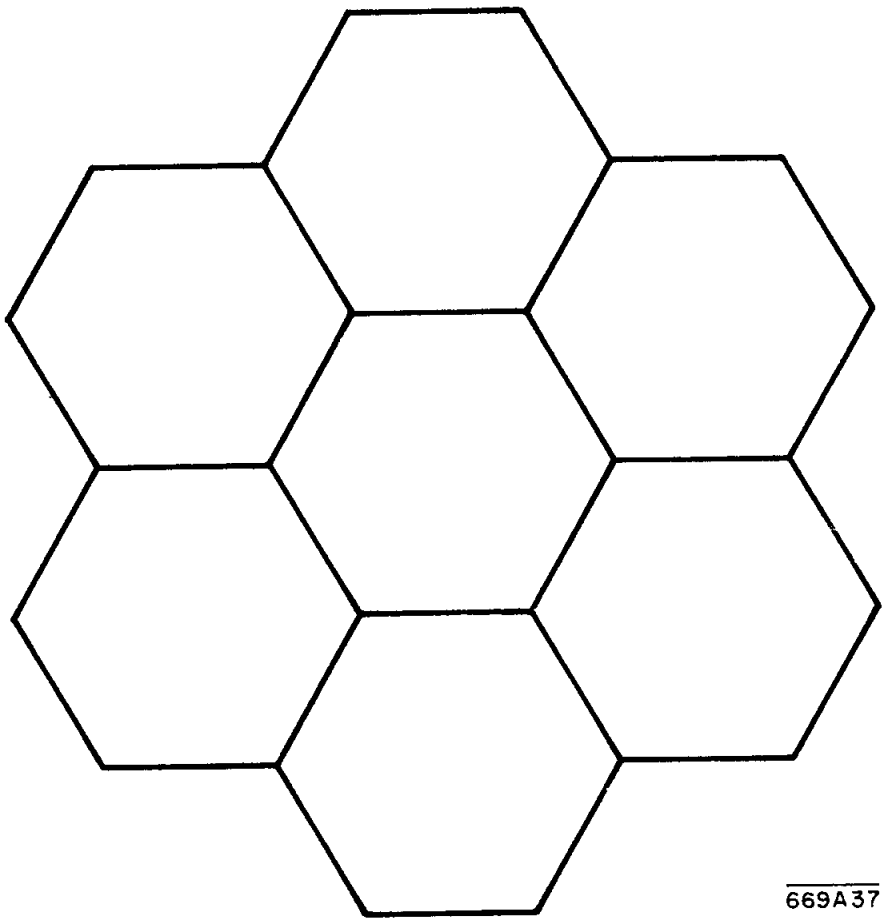
(b) Dr. von Hochstetter also succeeded, by the same gelatin and ink staining method, in micro-photographing the structure of the cornea of the eye of a cow which clearly discloses the three-way multi-frequency geodesic grid. (These two photographs are from Structure in Art and in Science, edited by Gyorgy Kepes, George Braziller, New York, 1965, with permission of the publishers.)

Figure 37



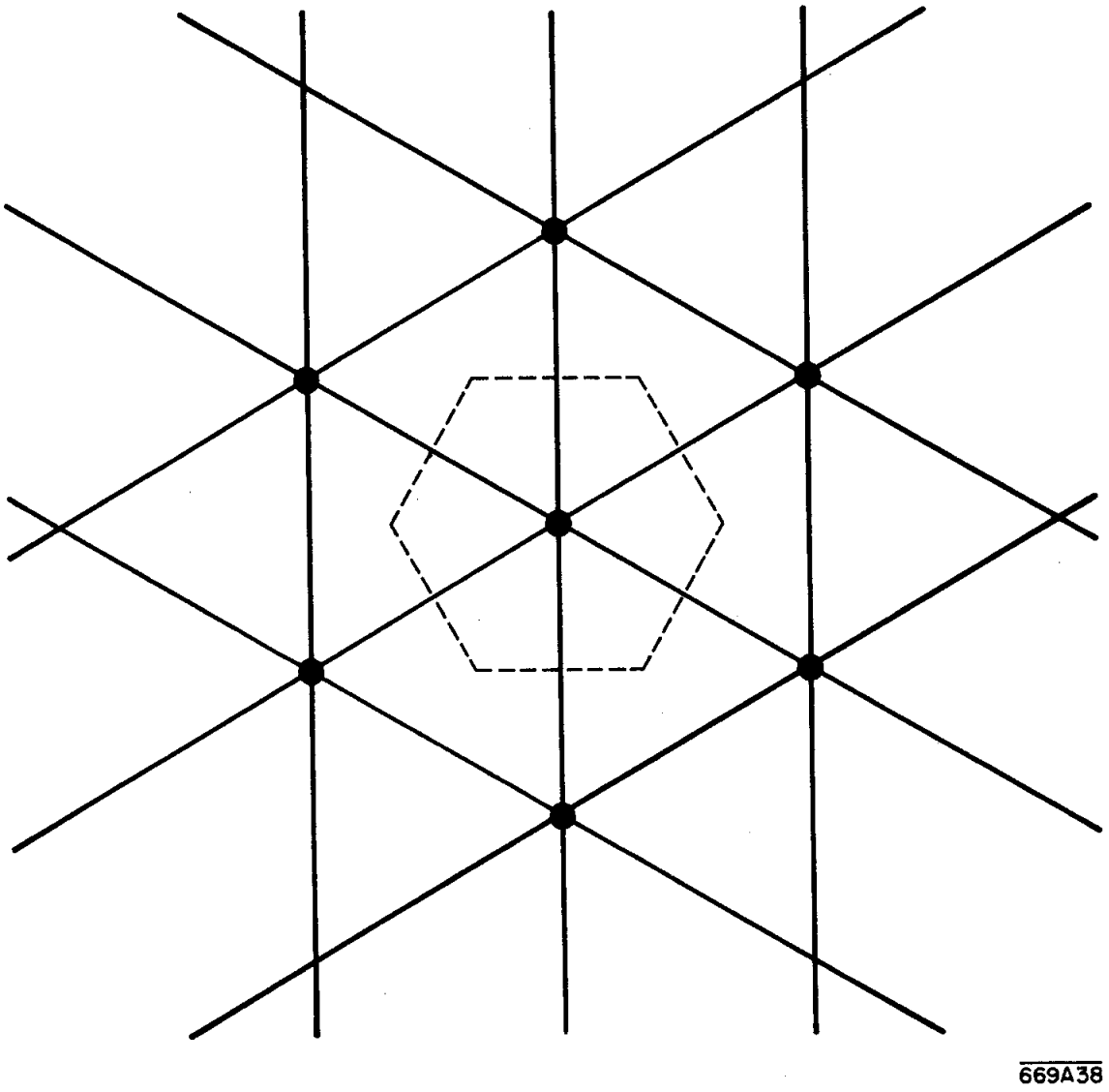
CLOSE-PACKED CIRCLES

FIG. 38



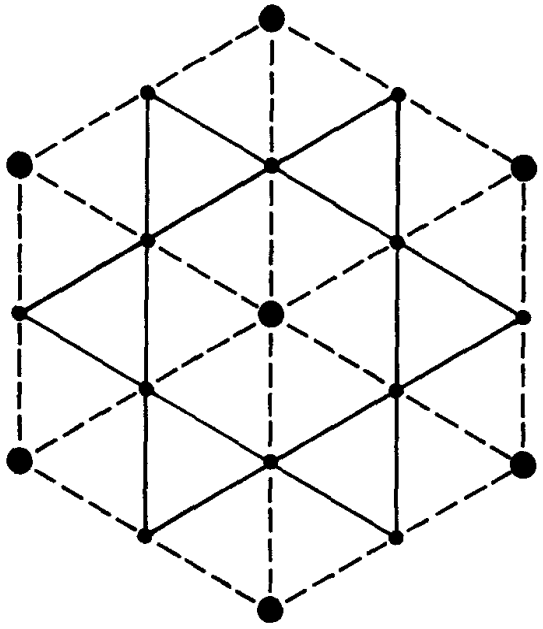
HEXAGONAL PACKING

FIG. 39



TRIANGULAR GRID

FIG. 40



669A39

● PICTURE POINTS

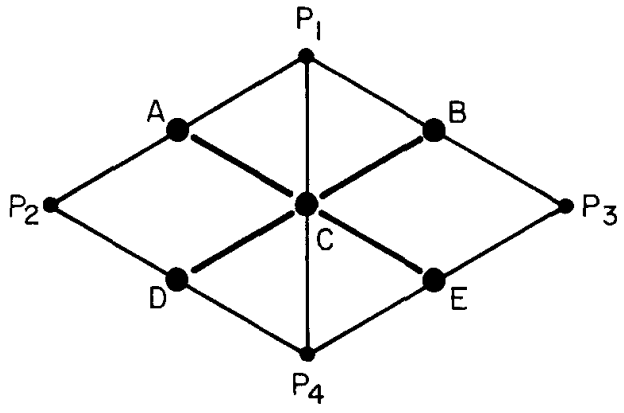
• POSSIBLE CURVATUREPOINTS

---- ORIGINAL TRIANGULAR GRIDLINES

———— POSSIBLE CONTOUR SEGMENTS

CURVATUREPOINT GRID

FIG. 41



669A40

P_i — PICTURE POINTS

A, B, C, D, E — POSSIBLE END-POINTS FOR CONTOUR SEGMENTS

C — CANDIDATE FOR CURVATUREPOINT

FOUR-POINT RHOMBIC WINDOW

FIG. 42

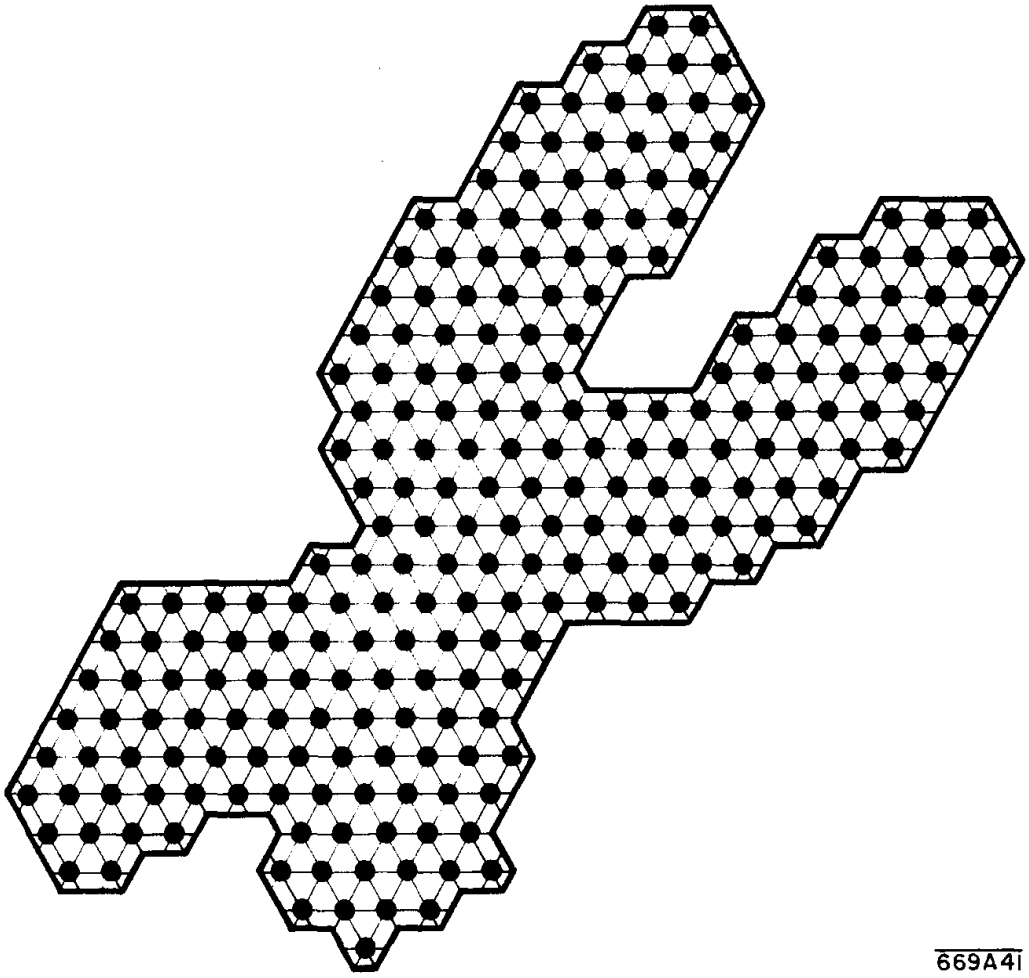
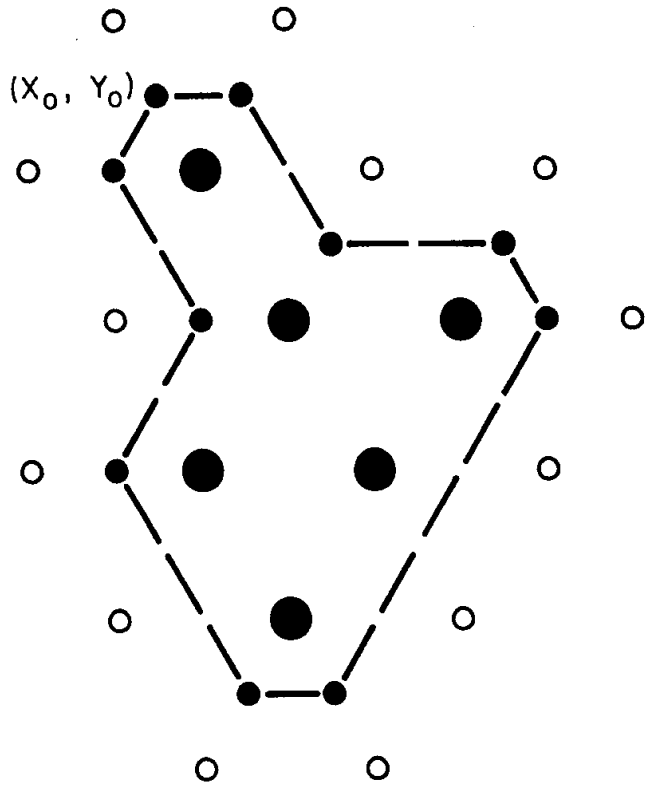


FIG. 43



BLACK POINTS

WHITE POINTS

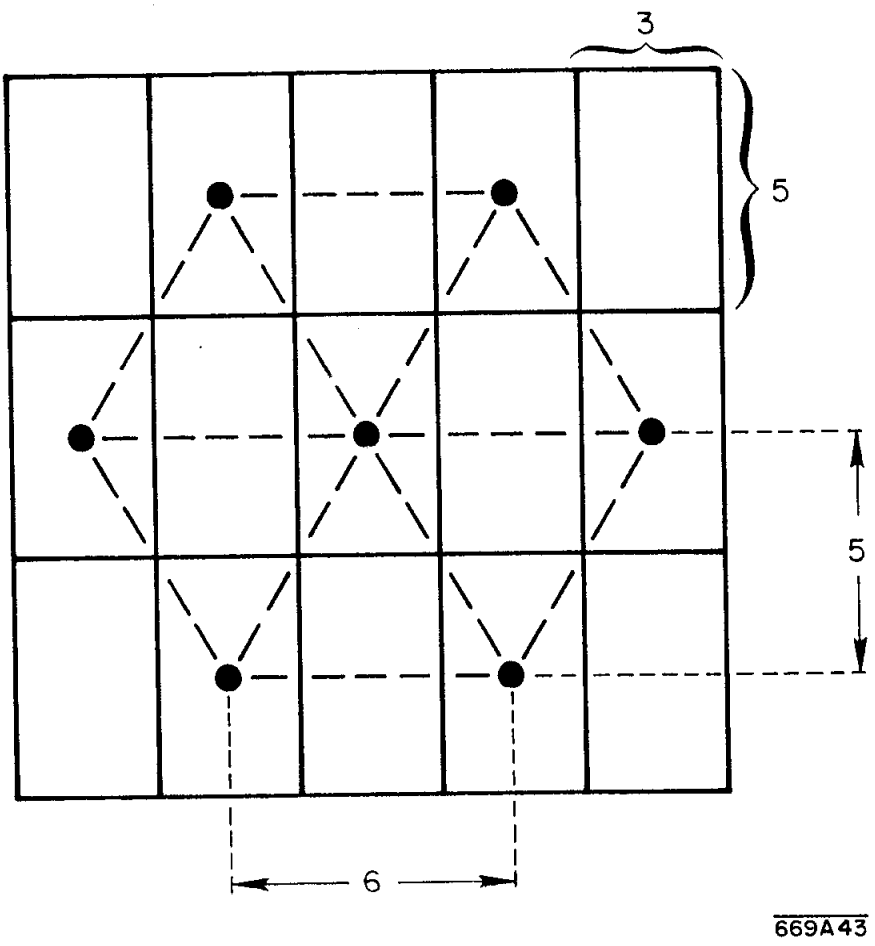
CURVATUREPOINTS

CONTOUR SEGMENTS

SEQUENTIAL DESCRIPTION = $[X_0, Y_0, \longrightarrow ; 1, 2, 2, 1, 5, 1, 3, 2, 2, 1]$

FIG. 44

● PICTURE POINTS



5/6 = .833

$\sqrt{3}/2$ = .866

.833/.866 = 96% LEAVING 4% DISTORTION

FIG. 45

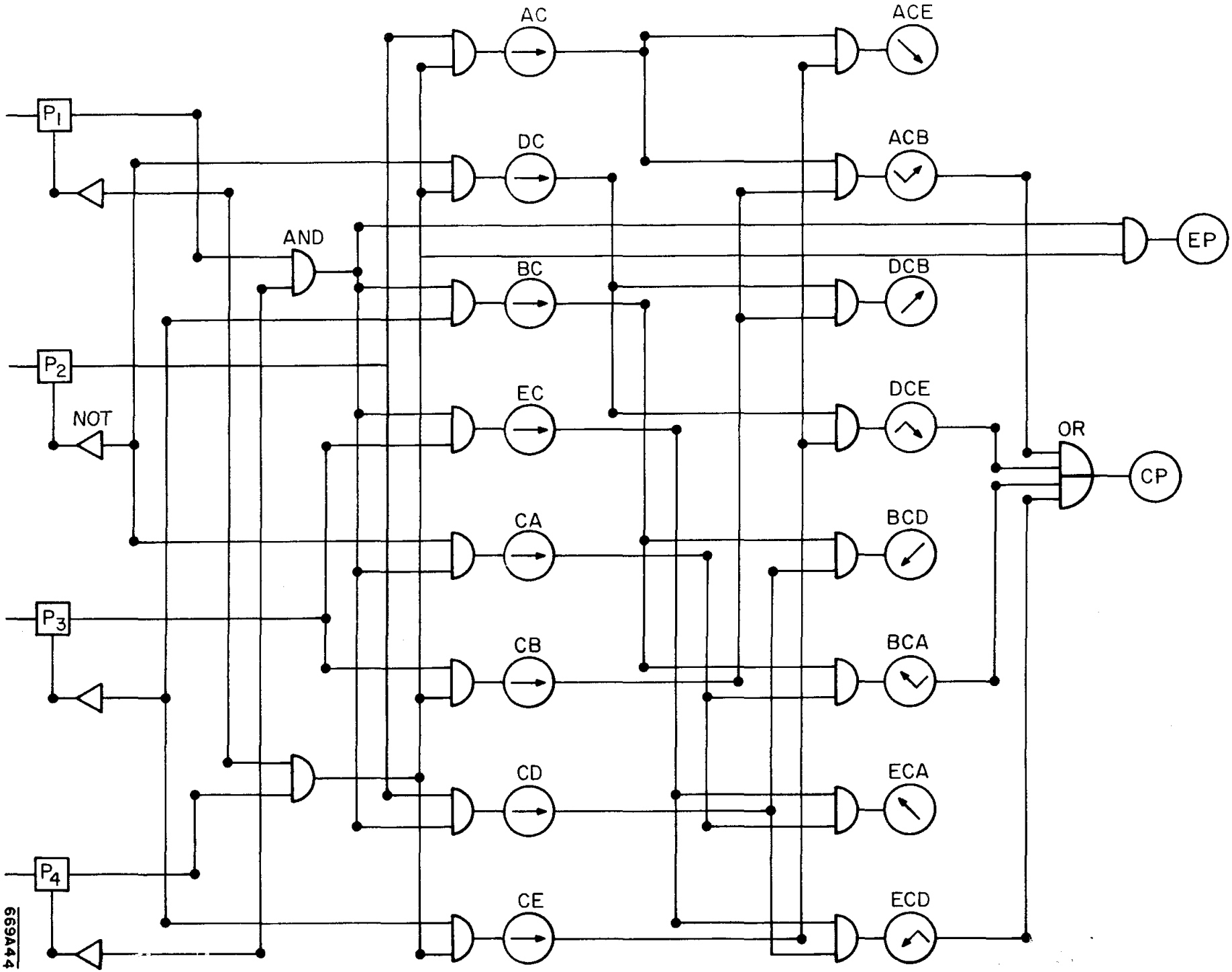

AC
ACE
P1
DC
ACB
AND
EP
BC
DCB
P2
EC
DCE
NOT
OR
CP
CA
BCD
P3
CB
BCA
CD
ECA
P4
CE
ECD
669A44

FIG. 46

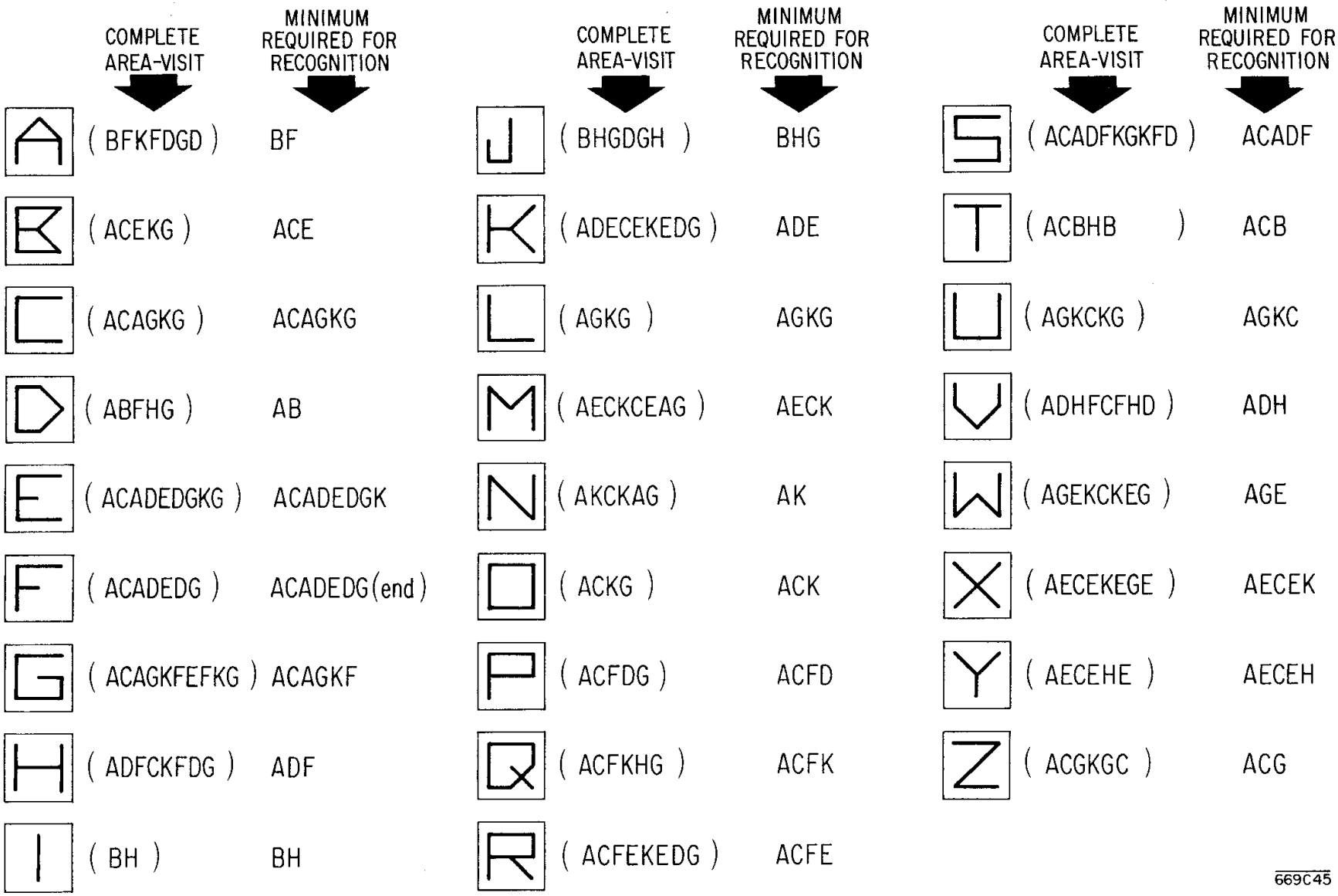

COMPLETE AREA-VISIT
MINIMUM REQUIRED FOR RECOGNITION
A (BFKFDGD) BF
B (ACEKG) ACE
C (ACAGKG) ACAGKG
D (ABFHG) AB
E (ACADEDGKG) ACADEDGK
F (ACADEDG) ACADEDG(end)
G (ACAGKFEFKG) ACAGKF
H (ADFCKFDG) ADF
I (BH) BH
J (BHGDGH) BHG
K (ADECEKEDG) ADE
L (AGKG) AGKG
M (AECKCEAG) AECK
N (AKCKAG) AK
O (ACKG) ACK
P (ACFDG) ACFD
Q (ACFKHG) ACFK
R (ACFEKEDG) ACFE
S (ACADFKGKFD) ACADF
T (ACBHB) ACB
U (AGKCKG) AGKC
V (ADHFCFHD) ADH
W (AGEKCKEG) AGE
X (AECEKEGE) AECEK
Y (AECEHE) AECEH
Z (ACGKGC) ACG
669C45

FIG. 47

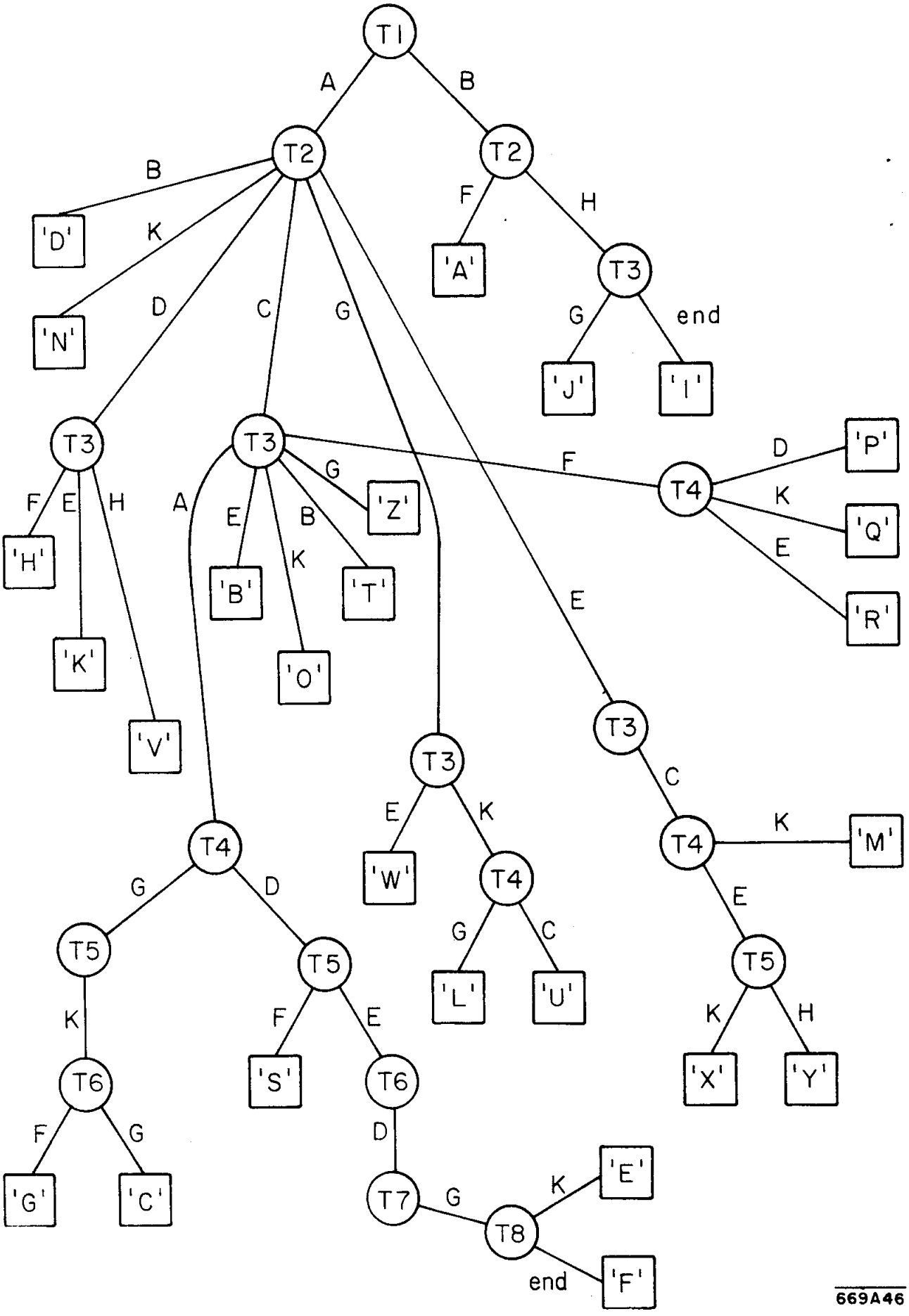

T1
A
B
T2
T2
B
F
H
'D'
K
'A'
T3
'N'
D
C
G
G
end
'J'
'I'
T3
T3
F
D
'P'
T4
F
E
H
A
E
B
G
'Z'
K
'Q'
E
'H'
'B'
'T'
'R'
'K'
'O'
E
'V'
T3
T3
C
E
K
T4
K
'M'
T4
'W'
T4
G
D
E
T5
T5
G
C
'L'
'U'
K
F
E
K
H
T6
'S'
T6
'X'
'Y'
F
G
D
'G'
'C'
T7
G
K
'E'
T8
end
'F'
669A46

FIG. 48

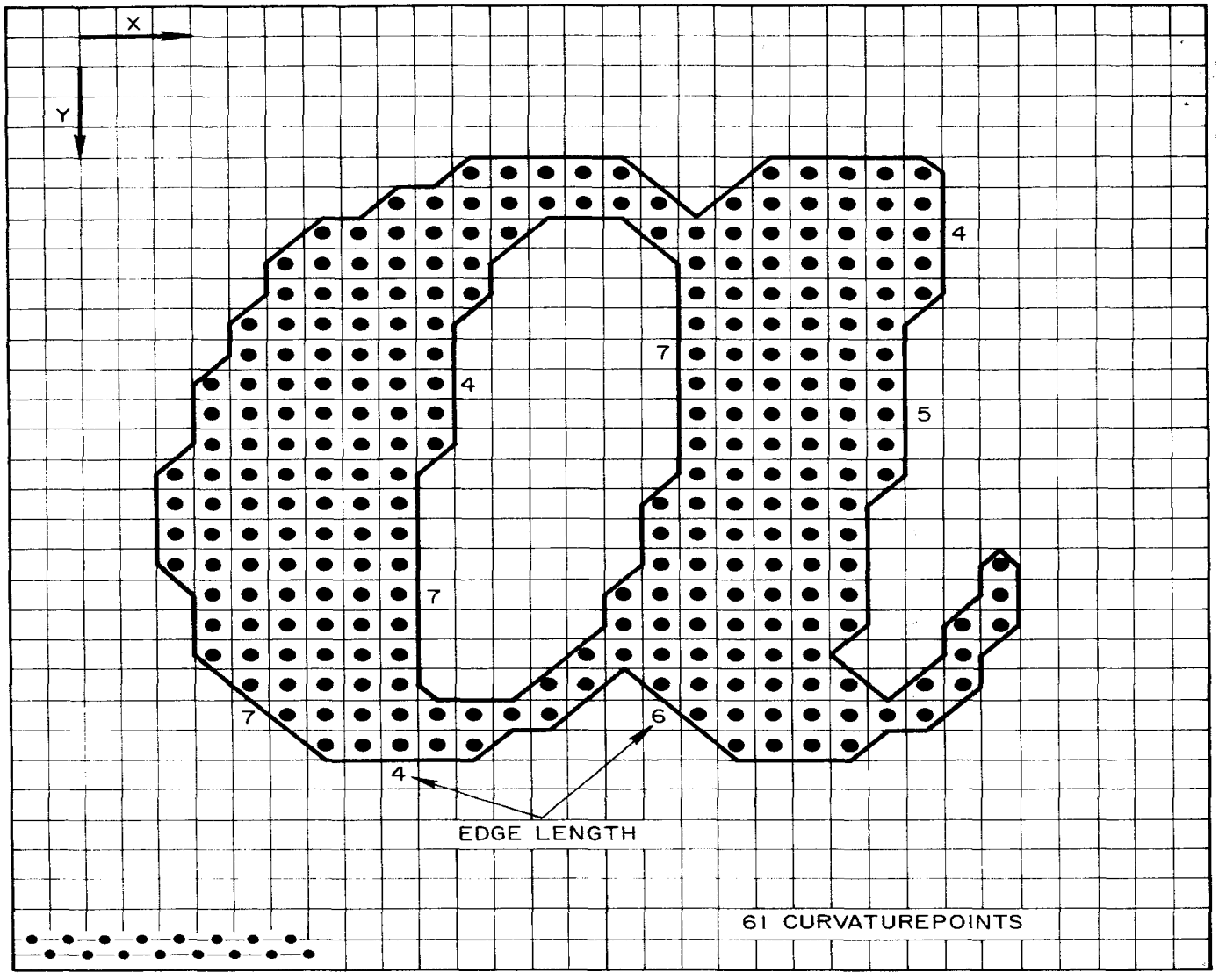
LETTER FREQUENCIES

Rank	Letter	Probability	
1	E	.13105	
2	T	.10468	
3	A	.08151	
4	O	.07995	
5	N	.07098	
6	R	.06882	← 50% division
7	I	.06345	
8	S	.06101	
9	H	.05259	
10	D	.03788	
11	L	.03389	
12	F	.02924	
13	C	.02758	
14	M	.02536	
15	U	.02459	
16	G	.01994	
17	Y	.01982	
18	P	.01982	
19	W	.01539	
20	B	.01440	
21	V	.00919	
22	K	.00420	
23	X	.00166	
24	J	.00132	
25	Q	.00121	
26	Z	.00077	

669A47

Figure 49

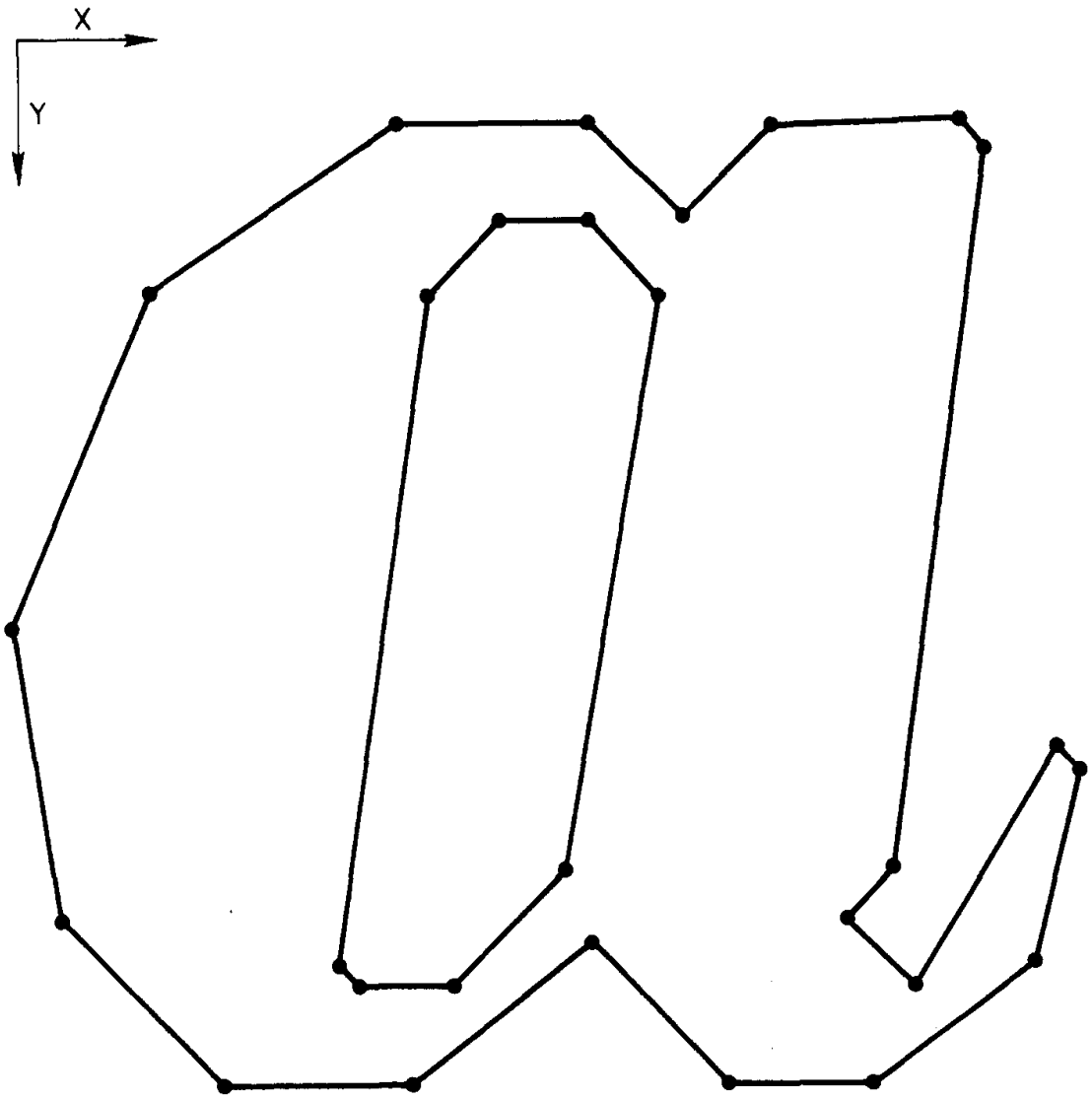
ARRAY (32 x 32) = 1024 BITS IN MATRIX MODE



2 TOPS — 22 BITS 16 BITS FOR Y, 5 BITS FOR X
2 INITIAL DIRECTIONS — 6 BITS 8 POSSIBLE DIRECTIONS
61 EDGE LENGTH — 183 BITS NO ACTUAL LENGTHS >7
61 BENDS — 122 BITS 1 BIT FOR BEND, 1 BIT FOR TERMINATE
TOTAL = 333 BITS

669A48

FIG. 50



669A49

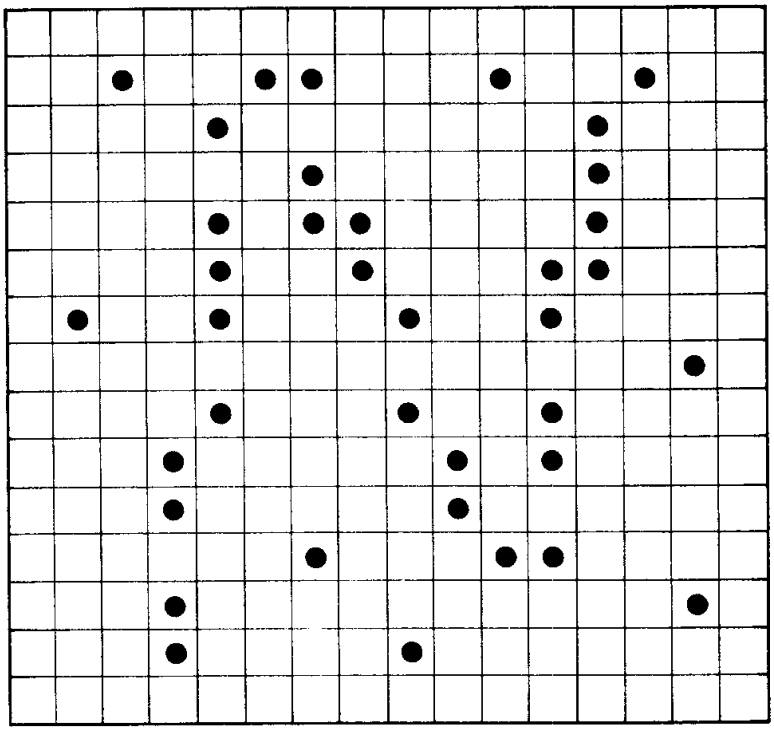
SMOOTHED A

28 CURVATURE POINTS AT 12 BITS EACH { Y – 6 BITS; X – 5 BITS; TERMINATE – 1 BIT }

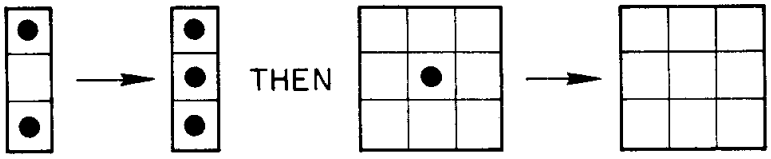
TOTAL OF 336 BITS

FIG. 51

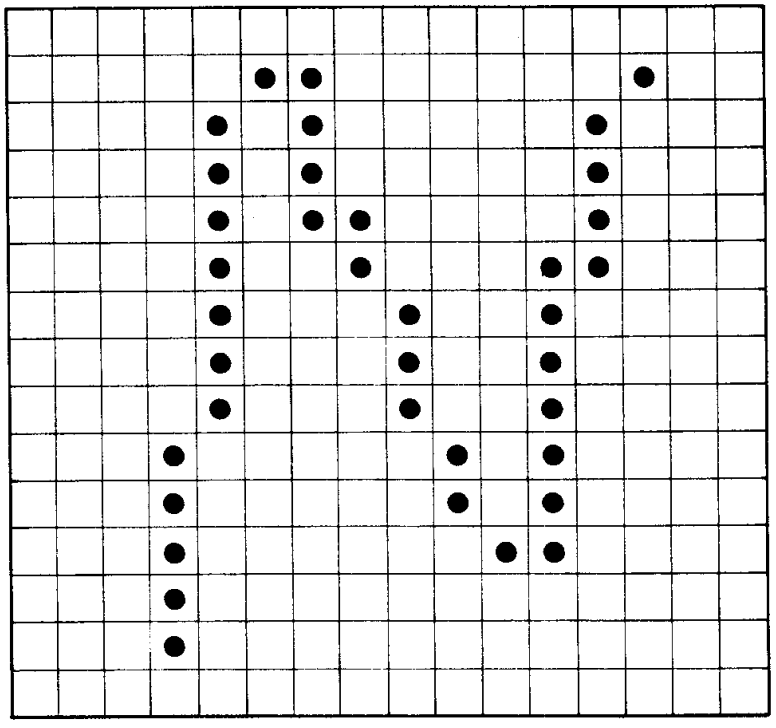
ORIGINAL
BINARY
PATTERN



PREPROCESSING
FORMULAS

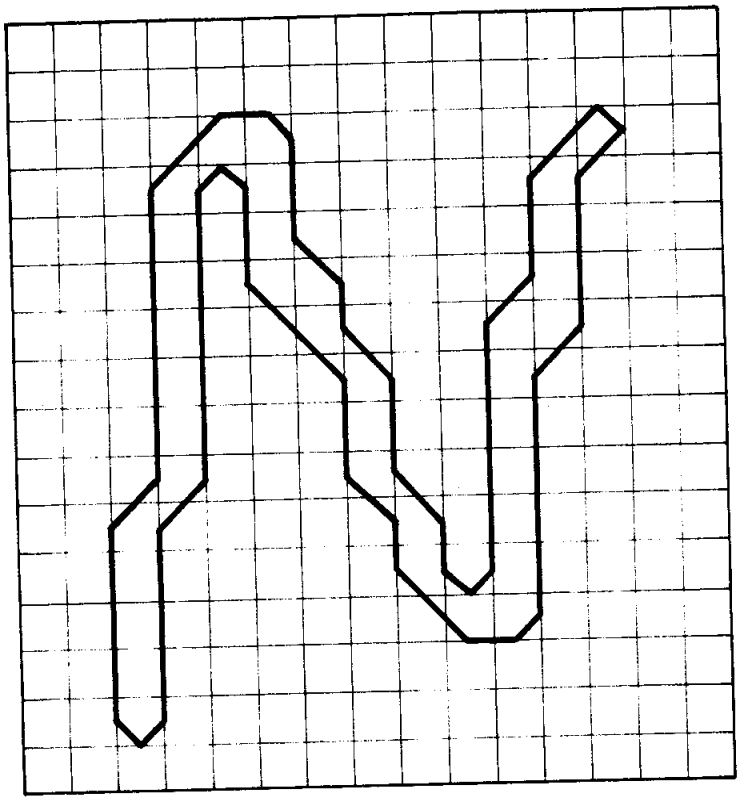


PREPROCESSED
PATTERN

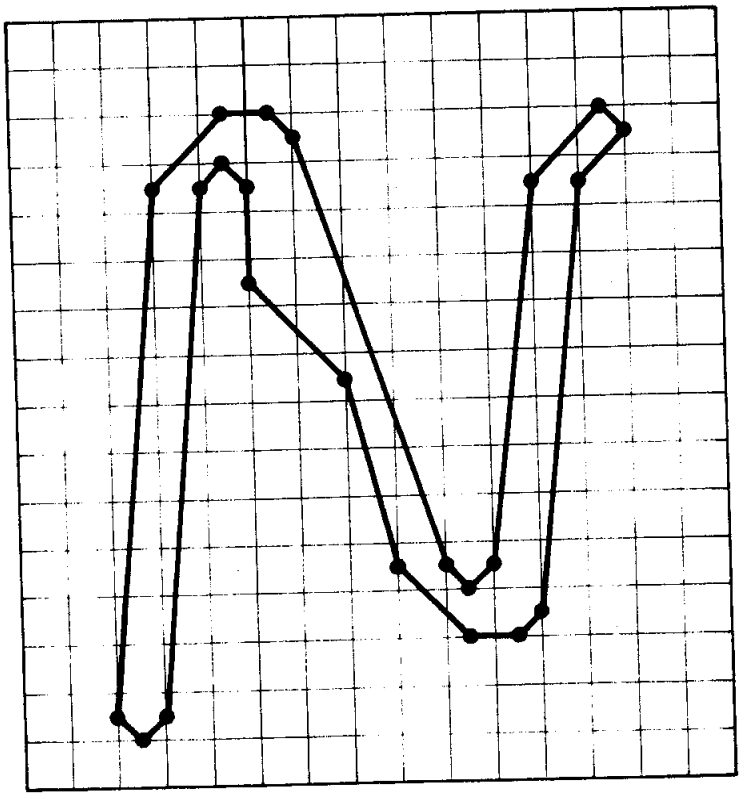


669A50

FIG. 52



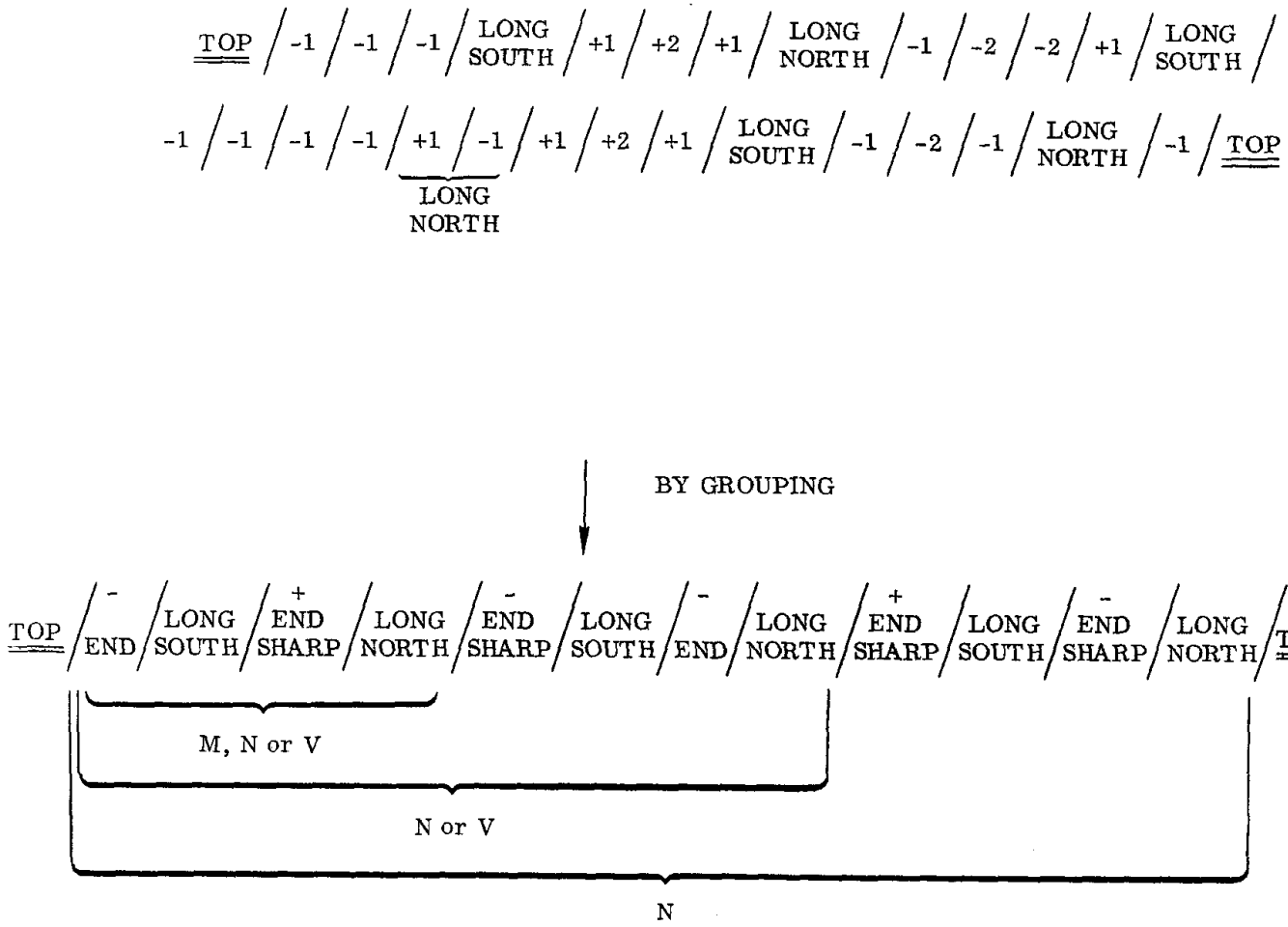
STRUCTURAL
DESCRIPTION
DEPICTED



SMOOTHED
DESCRIPTION
DEPICTED

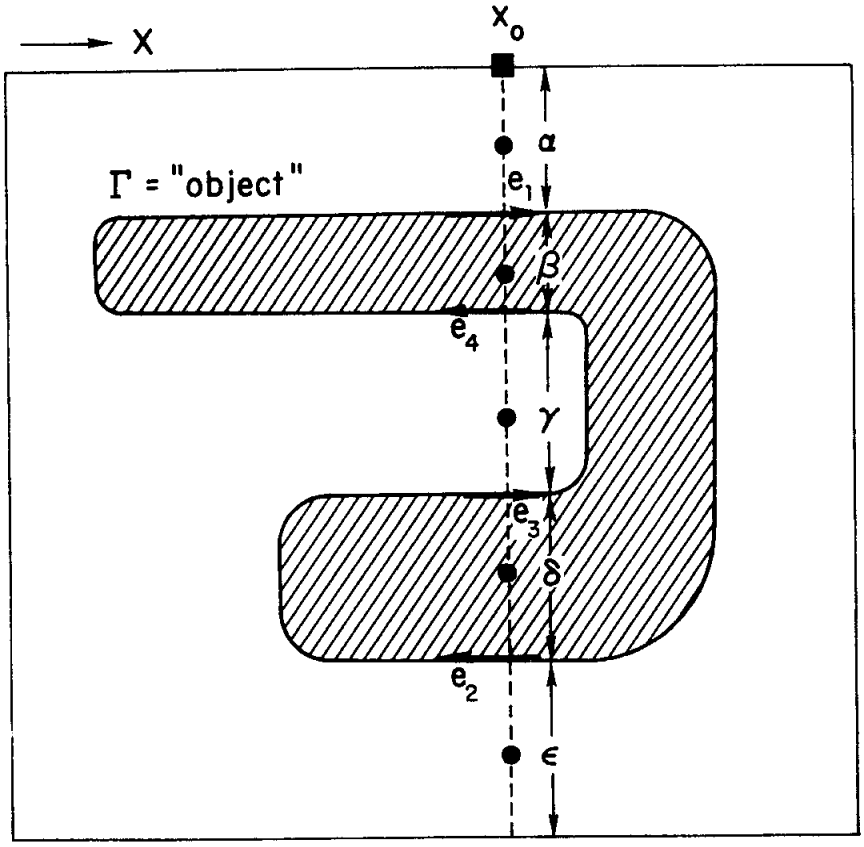
669A51

FIG. 53



RECOGNITION

Figure 54



	α	β	γ	δ	ε
e_1		+	+	+	+
e_2					+
e_3				+	+
e_4			+	+	+
$+\Sigma$	0	1	0	1	0

669A53

Fig. 55