https://ntrs.nasa.gov/search.jsp?R=19710006457 2018-07-20T10:15:19+00:00Z

N71-15932
NASA CR-116131

Technical Report 70-115 June, 1970
NGL-21-002-008
AT-(40-1)-3662

THINNING ALGORITHMS ON RECTANGULAR, HEXAGONAL AND TRIANGULAR ARRAYS

E. S. Deutsch

CASE FILE COPY

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND

Technical Report 70-115 June, 1970
NGL-21-002-008
AT-(40-1)-3662

THINNING ALGORITHMS ON RECTANGULAR, HEXAGONAL AND TRIANGULAR ARRAYS

E. S. Deutsch

ABSTRACT

In this report three thinning algorithms are developed; one each for use with rectangular, hexagonal and triangular arrays. The approach to the development of each algorithm is the same. Pictorial results produced by each of the algorithms are presented and the relative performances of the latter are compared. It is found that the algorithm operating with the triangular array is the most sensitive to image irregularities and noise. Yet it will yield a thinned image with an overall reduced number of points. It is concluded that the algorithm operating in conjunction with the hexagonal array has features which strike a balance between those of the other two.

This research was supported in part by Grant NGL-21-002-008 from the National Aeronautics and Space Administration, and in part by Contract AT-(40-1)-3662 with the Atomic Energy Commission.

Introduction

In much of the literature concerned with thinning or skeletonizing operations on digital images, the most common type of array used is the rectangular one. All of the thinning operations proposed, however diverse, make use of a small rectangular subarea, centered at each point in the picture, within the confines of which the skeletonizing operations are performed. This is not surprising, for conventional scanning techniques, operating in a line-by-line fashion, at once suggest this type of array and hence the subsequent form of processing.

In this paper it is proposed to examine additional types of arrays: not only rectangular, but also hexagonal and triangular arrays. These arrays correspond to the two-dimensional mosaics. The fact that none of these arrays, with the exception of the first, follow a strict row-and-column arrangement is of no real consequence, for given a fixed rectangular array, the others can always be derived therefrom.

In comparing algorithms associated with each of the four types of arrays, the basic approach to each algorithm will be the same. A thinning algorithm developed for rectangular arrays forms the basis of the approach. This thinning algorithm was first proposed by Rutovitz [1] and subsequently modified by the author [2].

The use of a generalized approach, rather than the development of a different type of algorithm for each

specific type of array, is deliberate, as it has the advantage that the relative merits of each type of array can be assessed in an easier manner. Otherwise, factors pertaining to the individuality of each approach would have to be taken into account. Suppose, for example that we found that an adaptive approach to thinning on a rectangular array was the best. Then it would be unfair to compare the results with those obtained using, say, an averaging technique on an hexagonal array. The approach adopted here is based on notions of connectivity.

Of interest will be the relative "efficiency" of each type of array. Clearly, as the structure becomes more involved so might its corresponding thinning algorithm. However, it will be useful to determine what advantages if any a particular array has, e.g., the resulting image reduction and processing time in each case.

Rectangular Arrays

The skeletonizing algorithm described below is partially described in [1] and [2]. It is presented here in full. In dealing with patterns on rectangular arrays it has been pointed out [3] that there are two connectivity situations. The pattern itself may be either four-way or eight-way connected. Correspondingly, the complement, or background, must be either eight-way or four-way connected. Verification of this statement is presented below and follows the argument in [3].

Consider the pattern shown in Figure 2. Assuming four-neighbor connectivity for both the pattern and the complement, the number of vertices V in the pattern is

16, the number of edges E is 16 and the number of faces is 4.

Application of the Euler formula

$$V - E + F$$

to the pattern should give its genus. Thus by the above formula the genus is 16 - 16 + 4 = 4. However the pattern has four components and the background has two, so that the genus (the number of pattern components minus the number of background components + 1) is 4 - 2 + 1 = 3. A similar disagreement in the value of the genus arises when eight-neighbor connectivity is assumed. For then the number of vertices in Figure 2 is 12, the number of edges 16, and the number of faces 4. Thus by Euler's formula the genus is 0, whereas in fact the genus should be 1.

However, if four-neighbor connectivity is assumed for the pattern and eight for the background then the genus, by Euler's formula, is 4. Since the number of background components is now 1 (not 2) the value of the genus obtained by counting the number of components is also 4. Similarly, when we assume the pattern to be eight-neighbor connected and the background to be four-neighbor connected, both methods of calculating the value of the genus, which is 0, agree.

For any element $a_{i,j}$ in row i and column j of the matrix, let $\gamma(1) \ldots \gamma(8)$ be its eight neighbors starting from $a_{i,j+1}$, in counter-clockwise order.

The crossing number, χ, is defined as

$$\chi = \sum_{k=1}^{8} | \gamma(k+1) - \gamma(k) |$$

and indicates the number of distinct continuous groups of black and white (pattern and background) elements around $a_{i,j}$. In order to delete an element from the pattern all of the following conditions must hold:

1. $\chi = 2$ or 4
2. $\sum_{k=1}^{8} \gamma(k) \geq 2$, i.e., the element must have at least two neighbors in the pattern
3. $\gamma(1)\gamma(3)\gamma(5) = 0$
4. $\gamma(1)\gamma(3)\gamma(7) = 0$
5. If $\chi = 4$ then in addition, either condition (a) or (b) must hold:

   a. $\{\gamma(1) \wedge \gamma(7) = 1\}$
   and
   $\{\gamma(2) \vee \gamma(6) = 1\}$
   and
   $\{\gamma(3) \vee \gamma(4) \vee \gamma(5) \vee \gamma(8) = 0\}$

   b. $\{\gamma(1) \wedge \gamma(3) = 1\}$
   and
   $\{\gamma(4) \vee \gamma(8) = 1\}$
   and
   $\{\gamma(2) \wedge \gamma(5) \wedge \gamma(6) \wedge \gamma(7) = 0\}$

The deletion operations continue until no further change occurs.

Briefly, the function of rule 2 is to prevent the already thinned components from vanishing. Rules 3 and 4 preserve the connectivity in the top and right hand positions within the rectangular window. Unless the pattern component is diagonal, erasure can only take

place if there is only one peripheral pattern component ($\chi = 2$, Rule 1). A special case arises when the pattern is a diagonal line, in which $\chi = 4$. Thinning also takes place if the diagonal line is two elements thick (Rule 5).

Rules 1 through 4 apply equally to a four-way connected pattern; rule 5, however, applies only if eight-neighbor connectivity is used. It has been shown that the algorithm does not disconnect [1], [2].

A further rule can be applied to each element in the image after the last pass, provided the elimination of rectangular curves is not considered harmful: if

$$\gamma(k)\gamma(k+2) = 1 \text{ for } k = 2, 4, 6, 8$$

then $a_{i,j}$ can be deleted.

It should be noted that the crossing number can be defined in a simpler way:

$$\chi = \sum_{k=1,3,5,7} | \gamma(k+2) - \gamma(k) |.$$

The value of χ now gives the number of eight-way connected components surrounding $a_{i,j}$. The previous definition of χ yielded the number of such four-way connected components.

The algorithm in its present form is nonisotropic. In order to render it isotropic the following additional rules have to be used:

6. $\gamma(3)\gamma(5)\gamma(7) = 0$
7. $\gamma(5)\gamma(7)\gamma(1) = 0$
8. if $\chi = 4$ then either condition (a) or (b) must apply:

a. $\{\gamma(5) \wedge \gamma(3) = 1\}$
and
$\{\gamma(6) \vee \gamma(2) = 1\}$
and
$\{\gamma(1) \wedge \gamma(4) \wedge \gamma(7) \wedge \gamma(8) = 0\}$

b. $\{\gamma(7) \wedge \gamma(5) = 1\}$
and
$\{\gamma(8) \vee \gamma(4) = 1\}$
and
$\{\gamma(5) \wedge \gamma(6) \wedge \gamma(7) \wedge \gamma(2) = 0\}$

Note that the last set of rules consists of rules 3 through 5 "rotated" through 180°. After the first pass using rules 1 through 5, rules 1, 2 and 6 through 8 are applied on a second pass, thus completing one cycle.

In what follows, similar sets of rules are developed for the other arrays. The approach will be the same in principle, and in each case cognizance will have to be taken of the neighborhood connectivities of the pattern and the background. Once this is established one can proceed with the design of a thinning algorithm.

Hexagonal Arrays

The first thing to note concerning hexagonal arrays is that there is no choice of connectivity for either the pattern or the background; both must be six neighbor connected. This is verified using the Euler formula on the pattern shown in Figure 2(a).

Assuming six neighbor connectivity, the number of vertices V is 24, the number of edges E is 30 and the number of faces F is 6. Using the Euler formula, the

genus is equal to 0. Since the number of components of the pattern is 1 and the number of components of the background is 2, the genus has the value 1 - 2 + 1 = 0. Thus both values of the genus agree.

In developing an algorithm for hexagonal arrays, it is noted that rules 1 and 2 pertaining to the rectangular array apply here too. Specifically, rule 2 is required to prevent already thinned lines from disappearing.

The crossing number can take the values 0, 2, 4 and 6. However, unlike the rectangular case, as soon as its value exceeds 2, the central element cannot be deleted; for in those cases the peripheral pattern (black) elements form two separate components which will be disconnected unless the center element is retained.

We thus have the first two rules

$$\sum_{k=1}^{6} \gamma(u) \geq 2$$

and $\chi = 2$

both of which must apply if the element is to be deleted.

Here too we require a set of rules similar to rules 3 and 4 above. In Figure 2(a) let the neighboring element immediately to the right of the center element be labelled $\gamma(1)$, and the remaining ones, in counterclockwise direction, $\gamma(2)$ through $\gamma(6)$. At first sight one is lead to stipulate that the following three connectivity conditions must be obeyed if the element is to be deleted:

(a) $\gamma(1)\gamma(2)\gamma(3) = 0$

(b) $\gamma(1)\gamma(2)\gamma(6) = 0$

(c) $\gamma(1)\gamma(5)\gamma(6) = 0$

These rules seem reasonable since they guarantee connectivity between any set of three consecutive peripheral elements. For example, if rule (a) is not obeyed, the center element is not deleted, and the connectivity between $\gamma(1)$ and $\gamma(3)$ is maintained via the center element. Any set of three consecutive elements not explicitly written down forms the mirror image of one of the sets in (a), (b), or (c), and is taken care of when a previous, or a later element is considered.

However, consider the simple pattern shown in Figure 3. In each pair of elements marked 1 and 2, one of the elements will have to be retained if the pattern is not to break up. By rule (c) the elements marked 1 are retained, whereas those labelled 2 are removed. Now consider the effect rules (a), (b), and (c) would have on the line pattern shown in Figure 4(a). The initial thinning stages are shown in Figures 4(b) and (c). Evidently rules (a) and (c) are incomparable: they cannot operate together in their present form and one of them must be changed.

Taking the mirror image of rule (c) in order to preserve connectivity of any three consecutive elements, i.e., the rule

$$\gamma(2)\gamma(3)\gamma(4) = 0$$

overcomes the difficulty encountered with the pattern in Figure 4; but if this rule is to be used then the pattern shown in Figure 5 will eventually vanish.

However, the connectivity between $\gamma(1)$ and $\gamma(5)$ (and between $\gamma(2)$ and $\gamma(4)$ when $\gamma(3)$ is the center element)

will be maintained if the following two rules are substituted for rule (c):

$$\gamma(1) = 1$$

$$\chi_{\gamma(1)} \neq 2$$

That is, the crossing number at neighbor $\gamma(1)$ (the latter must also belong to the pattern), must not be 2. For then, if $\gamma(5)$ belongs to the pattern, connectivity between $\gamma(1)$ is maintained via the center element.

The rules constituting the hexagonal thinning algorithm are summarized below.

1. $\sum_{k=1}^{6} \gamma(k) \geq 2$
2. $\chi = 2$
3. $\gamma(1)\gamma(2)\gamma(3) = 0$
4. $\gamma(1)\gamma(2)\gamma(6) = 0$
5. $\gamma(1) = 1$ and $\chi_{\gamma(1)} \neq 2$.

Once again, in order to render the algorithm isotropic, the additional rules given below can be used:

6. $\gamma(4)\gamma(5)\gamma(6) = 0$
7. $\gamma(3)\gamma(4)\gamma(5) = 0$
8. $\gamma(4) = 1$ and $\chi_{\gamma(4)} \neq 2$

Triangular Arrays

The structure of the triangular array is somewhat more complicated than either of those discussed above. The first thing to notice about the array is the fact that the triangular elements have alternating orientations. The arrangement of peripheral elements will thus vary accordingly. (See Figure 6.)

Here too it will be necessary to establish the neighborhood connectivity of both the pattern and its background. Before doing so we first show, in Figures 6(a) and (b), the two different nearest-neighbor arrangements for the triangular array. For ease of reference, the central element in Figure 6(a) will be referred to as Δ, and that of Figure 6(b) as ∇.

It will be observed from the figure that the nearest neighbors of both Δ and ∇ can be divided into three sets such that all the elements within each set are equidistant from the central element, distances being measured from centroids. The first set contains the triangular neighbors whose centroids are at a distance of 2 units away from the central element's centroid (where the median of each triangle is of length 3 units). The second set contains the six elements each entroid of which is at a distance of $2\sqrt{3}$ units away, and the third contains the three elements furthermost away, at a distance of 4 units. Note that included in this count are all the elements which share either an edge or a vertex with either Δ or ∇. This suggests a choice of neighborhood connectivity; 3, 9, or 12. See Figure 6(c), where the elements around Δ have been numbered according to the sets to which they belong.

It will be found that the Euler equation quoted above is satisfied if the pattern is 12-neighbor connected. As a result the background must be 3-neighbor connected. As a verification consider the configuration of elements in Figure 6(d), where the shaded elements constitute the pattern.

The number of vertices V -- if 12 neighbor connectivity is assumed -- is 12. The number of edges E is 23 and the number of faces in the pattern is 11. Accordingly, the genus is given by

$$12 - 23 + 11 = 0$$

The genus is in fact zero because the numbers of components and holes in the configuration are both 1.

For the pattern to be 12-way connected, the background must be 3-neighbor connected. This, because now, the only way the chain of triangles surrounding Δ can be broken, so as to connect the "hole" in the center with the "outside", is for two neighboring elements, with a common edge, to belong to the background too -- i.e., the background must be 3-way connected.

By the same token, if the pattern in Figure 7(d) is 3-way connected, then the number of vertices it contains is 13 (there are in effect two vertices at A), the number of edges is 23, and the number of faces is 11. Thus the genus, by Euler's formula, is

$$13 - 23 + 11 = 1$$

Once again this corresponds to the true value of the genus; there are now only two components, pattern and background, whereas before there were three, one of the pattern and two of the background. With 3-way connectivity of Figure 6(d) there is in fact a gap at A. The same argument would apply had the pattern in Figure 6(d) consisted of all but the center element and any other of the 12 neighbors.

As a further example consider the arrangement shown in Figure 7(a). If 12-neighbor connectivity is assumed then there will be no gap at A. Here the connection is maintained by an element belonging to the third set of neighbors. Similarly, there would not be a gap at B, where connectivity is maintained by an element belonging to the second set of neighbors. This, in principle, is similar to the anomalies arising in rectangular arrays: in Figure 7(c) there would be a gap at A unless 8-neighbor connectivity is assumed.

Let each peripheral element be denoted by $\gamma_h(k)$ where k is the number of the element belonging to the set h = 1, 2, or 3. At the same time it will be convenient to refer to any of the 12 neighboring elements simply as $\gamma(k)$. In the latter representation k denotes the neighbor number, 1 - 12, in general. Thus $\gamma_2(3) \equiv \gamma(4)$; see Figure 8. The arrangement of a computer printout of a triangular array is shown in Figure 9(a) together with the 12 neighbors of Δ and ∇ corresponding to Figure 8. The distances of neighbor $\gamma(1)$ through $\gamma(12)$ from Δ are given by the pairs

(1,1),(1,2),(0,3),(-1,2),(-1,1),(-2,0)
(-2,-1),(-1,-2),(0,-1),(1,-2),(2,-1),(2,0), respectively.

For ∇ the corresponding distances are

(1,-1),(2,0),(2,1),(1,2),(0,1),(-1,2)
(-2,1),(-2,0),(-1,-1),(-1,-2),(0,-3),(1,-2) respectively.

It will be observed that the $\gamma_1(k)$ elements are flanked on either side by two $\gamma_2(k)$ neighbors. In turn, each such pair of $\gamma_2(k)$ elements is separated by a $\gamma_3(k)$ type element.

Similarly, each $\gamma_2(k)$ neighbor has as its immediate neighbor a $\gamma_1(k)$ and a $\gamma_3(k)$ neighbor, while every $\gamma_3(k)$ neighbor has two $\gamma_2(k)$ element as its immediate neighbor.

If the pattern were 3-way connected, then for either Δ or ∇ to be completely surrounded by pattern elements, all the 12 neighbors must belong to the pattern. On the other hand, for a 12-way connected pattern the central element can be completely surrounded by combinations of either $\gamma_1(k)$ or $\gamma_2(k)$ elements or both. The $\gamma_3(k)$ type neighbors can be bypassed. This fact will be used later on.

In view of its rather large size, there arise within the window itself a number of closed subpatterns, the connectivity of which must be preserved. One such case is shown in Figure 10(a) in which

$$\gamma_1(3) = 0 \text{ and } \gamma_2(5) = \gamma_2(6) = \Delta = 1.$$

There are a total of six such cases, in each of which the central element together with the two $\gamma_2(n)$ neighbors flanking its $\gamma_1(n)$ neighbor have the value 1 while the $\gamma_1(n)$ element itself is 0. We thus have the first connectivity rule which must be satisfied before either Δ or ∇ can be deleted from the pattern:

$$\overline{\gamma_1(k)} \wedge \gamma_2(2k-1) \wedge \gamma_2(2k) = 0$$

$$\text{for } k = 1, 2, \text{ or } 3$$

The two-neighbor rule used for the other arrays cannot be applied indiscriminately here. Figure 10(b) shows two patterns, one being a rotated version of the

other, which would vanish if this rule were to be applied. There are a total of twelve such cases, six for each of the two neighbor arrangements. Accordingly, we have a further rule which must be satisfied prior to the erasure of either Δ or ∇:

$$\sum_{k=1}^{12} \gamma(k) > 2$$

or if $\sum \gamma(k) = 2$ then $\gamma_1(k) \wedge [\gamma_2(2k-1) \vee \gamma_2(2k)] = 0$
for $k = 1, 2,$ or 3

It was stated above that in tracing a 12-way connected path around Δ it is immaterial whether or not the $\gamma_3(n)$ elements are included in the path. Thus a further condition must be satisfied if the central element is to be deleted:

$$\bigwedge_{k=1,2,3} \{\gamma_1(k) \vee [\gamma_2(2k-1) \wedge \gamma_2(2k)]\} = 0.$$

This test examines whether the central element is completely surrounded by a path in the pattern. If the above expression is nonzero then the element cannot be erased since it combines with its nearest and next nearest set of neighbors to form one component of the pattern. Some examples are shown in Figure 11.

Two separate thinning algorithms were developed for this array, the difference between them being the method of defining the crossing number. In the first algorithm all the peripheral elements are used just as in the case of the rectangular arrays. In the second algorithm the crossing number was calculated to be the number of connected

components around Δ or ∇.

Thus for the first algorithm we have the same rule as for the preceding cases, namely that $\chi \geq 2$.

From Figure 9(b) it will be seen that the $\gamma_2(n)$-type neighbors form a hexagon around Δ (or ∇). It was therefore thought convenient to apply, in principle, the rules developed for the hexagonal arrays. Accordingly two further sets of rules were added both of which must be satisfied before the central element is removed:

If the central element is Δ, we require

$$\gamma(2)\gamma(4)\gamma(12) = 0$$

$$\gamma(2)\gamma(10)\gamma(12) = 0$$

$$\chi_{\gamma_2(2)} \geq 2$$

If the central element is ∇ we require

$$\gamma(2)\wedge\gamma(4)\wedge\gamma(6) = 0$$

$$\gamma(2)\wedge\gamma(4)\wedge\gamma(12) = 0$$

$$\chi_{\gamma_2(2)} \geq 2$$

It was observed from the results that images are not reduced to single-line thickness. Briefly the reason for this is due to the abnormally large window size and the method whereby the crossing number was defined. This algorithm was not developed any further because the one described below gave better thinning results. As a final point in connection with this algorithm, an improvement may result if the rules containing the associated $\gamma_1(n)$ and $\gamma_3(n)$ elements are also included. Thus for the deletion of Δ we would require the condition that

$$\gamma(2)\gamma(4)\gamma(5) = 0$$
$$\text{or } \gamma(3)\gamma(4)\gamma(5) = 0.$$

Similarly for the deletion of ∇ we would require that

$$\gamma(2)\wedge\gamma(3)\wedge\gamma(12) = 0$$
$$\text{or } \gamma(1)\wedge\gamma(2)\wedge\gamma(4) = 0$$

Let us inspect the triangular array a little more closely. The neighbors of Δ or ∇ together with Δ or ∇ respectively can be thought of as forming three hexagonal lobes (see Figure 12(a)); in each lobe the central element provides the sixth vertex. Considering one such lobe, it differs from the ordinary hexagonal arrangement in that the former has no central element. (This is in fact a method of generating triangular arrays; however, in so doing, the density of points changes. In the experiments described here the density of points remained as high as possible for a given fixed basic rectangular array.) This being the case, and since each vertex of each lobe is connected directly to all the other vertices in that lobe (this also applies to the elements which form part of the neighboring lobes), we can define a new crossing number as the value of χ' where,

$$\chi' = \sum_{k=1,5,9} |\{\gamma(k)\vee[\gamma(k-1)\wedge\gamma(k+1)]\}-\{\gamma(k+4)\vee[\gamma(k+3)\wedge\gamma(k+5)]\}|.$$

The value of χ', just as in the case of the modified crossing number definition for rectangular arrays, gives the number of 12-way connected components surrounding Δ or ∇. Thus in the modified algorithm we have the rule $\chi' \geq 2$ which replaces the old $\chi \geq 2$ rule; that is, unless $\chi' \geq 2$ Δ or ∇ cannot be erased. Note that the rule concerning the number of neighbors remains as before.

It is now necessary to establish new connectivity conditions, for those used in the previous algorithm are inadequate. This will become apparent as the new rules are developed below.

The similarity of the formation of the 12 neighbors to a hexagonal arrangement is still maintained, but with the new definition of the crossing number there are two hexagonal arrangements to consider: Firstly, that formed by the $\gamma_2(n)$ neighbors, as before, and secondly, that portion of a hexagonal array formed by two $\gamma_1(n)$ elements and one $\gamma_2(n)$ element (Figure 13(b) and (c)). The latter arrangement did not have to be considered in the previous algorithm because the crossing number of this combination, if it existed on its own, would not have been 2. The neighbors forming the additional <u>partial</u> hexagonal arrangement around Δ and ∇ respectively are

$$\gamma(1),\ \gamma(5),\ \gamma(12)$$
$$\text{and } \gamma(1),\ \gamma(9),\ \gamma(2)$$

It will have been observed that the 12 neighbor arrangement of ∇ is identical with that of Δ rotated through 180°. This being the case, we will first develop the rules for the neighbor of Δ and then apply them to those of ∇. Reference to the rules developed for the hexagonal arrays above indicates that there is one additional rule, applying to the neighbors of ∇ only, which involves the neighbors $\gamma(1)$, $\gamma(9)$ and $\gamma(2)$.

Consider once again Figure 12(b) and the first connectivity rule for the hexagonal array. It is apparent

that the equivalent of this rule here would be the two rules

$$\gamma(2)\gamma(4)\gamma(12) = 0$$

$$\text{and } \gamma(1)\gamma(5)\gamma(12) = 0,$$

or on combining them

$$[\gamma(1)\vee\gamma(2)]\wedge[\gamma(4)\vee\gamma(5)]\wedge\gamma(12) = 0.....T_1$$

The above rule incorporates the connectivity conditions of parts of the hexagonal arrays formed by the $\gamma_2(n)$ neighbors and the $\gamma_2(n)$ and $\gamma_1(n)$ neighbors.

Similarly the second connectivity rule, "borrowed" from the hexagonal array, requires that

$$\gamma(2)\gamma(10)\gamma(12) = 0.$$

There is, however, a further element combination to the right of Δ which cannot be disturbed, namely that formed by $\gamma(2)$, and the pairs of $\gamma(1)$ and $\gamma(12)$ and $\gamma(10)$ and $\gamma(9)$. Accordingly we have the combined rule for the above two cases

$$[\gamma(1)\vee\gamma(12)]\wedge[\gamma(9)\vee\gamma(10)]\wedge\gamma(2) = 0.....T_2$$

Finally in the same vein we have the rule which will maintain the connectivity between the $\gamma_1(n)$ neighbors only,

$$\gamma(1)\gamma(9) = 0.....T_3$$

The connectivity situations discussed so far involved combinations of either $\gamma_1(n)$ or $\gamma_2(n)$ elements or both. There are, however, further combinations involving all three types of neighbors.

In Figure 13 are shown some examples of a pattern consisting of $\gamma_1(n)$, $\gamma_2(n)$ and $\gamma_3(n)$ type elements. It will be appreciated that such combinations can only be

formed along the three principal lines of the triangular window. Each such line includes one of the sides of either Δ or ∇.

In Figure 13(a), we have the full combination

$$\gamma_1(k)\gamma(k\pm2)\gamma(k\pm3) = 1 \text{ for } k = 1,$$

yet the partial combination of $\gamma_1(n)$, $\gamma_2(n)$ and $\gamma_3(n)$ elements shown in Figure 13(c) must also be considered.

In total there are six such principal combinations, three for Δ and three for ∇. However, from Figure 14(a) it becomes clear that the combination shown is the only one that has to be considered for Δ. The remaining two combinations formed along the remaining two principal lines lie either to the left of or below Δ and need not be examined. We thus have the rule that unless $\gamma(1)$ together with any three of the elements $\gamma(k\pm2)$ and $\gamma(k\pm3)$ satisfy

$$\gamma_1(k)\wedge\gamma(k\pm2)\gamma(k\pm3) = 0 \quad \text{for } k = 1.....T_4$$

Δ cannot be erased.

There will be no direct equivalent of rule T_1 for Δ, in the case of ∇. However, the equivalent of the third rule for hexagonal arrays yields here the rule

$$\gamma(2)\gamma(4)\gamma(6) = 0$$

and by supplementing this rule so that it also includes the $\gamma_2(n)$ and $\gamma_3(n)$ neighbor we get the combined rule

$$[\gamma(2)\vee\gamma(3)]\wedge[\gamma(4)\vee\gamma(5)]\wedge\gamma(6) = 0.....T_5$$

The equivalent of rule T_2 is

$$[\gamma(2)\vee\gamma(1)]\wedge[\gamma(4)\vee\gamma(5)]\wedge\gamma(12) = 0 \ldots\ldots T_6.$$

A further rule, applicable to ∇ only, which is the equivalent of the fifth rule

$$\gamma(2) = 1 \text{ and } \chi'_{\gamma(2)} = 2$$

The equivalent of rule T_3 will be

$$\gamma(1)\gamma(5) = 0 \ldots\ldots T_7.$$

Lastly, we have to consider the connectivity situations involving all three types of elements. Whereas there was only one such principal (along one principal line) case to be considered for Δ there are two principal cases here (Figures 13(b) and (d)). Accordingly, we have the two connectivity rules similar to rule T_4:

$$\gamma_1(k)\gamma(k\pm2)\gamma(k\pm3) = 0 \quad \text{for } k = 1 \text{ and } 2 \ldots\ldots T_8.$$

To summarize, the thinning rules applicable to the triangular array are

1. $\sum_{k=1}^{12} \gamma(k) > 2$, or if $\sum_{k=1}^{12} = 2$ then

   $\gamma_1(k)\wedge[\gamma_2(k-1)\vee\gamma_2(k)] = 0 \quad k = 1, 2, \text{ and } 3$

2. $\bar{\gamma}_1(k)\wedge\gamma_2(2k-1)\wedge\gamma_2(k) = 0 \quad k = 1, 2, \text{ and } 3.$

3. $\chi' = 2$

Δ	∇
4a $\gamma(1)\wedge\gamma(9)=0$	4b $\gamma(1)\wedge\gamma(5)=0$
5b $[\gamma(4)\vee\gamma(5)]\wedge[\gamma(1)\vee\gamma(2)]\wedge\gamma(12)=0$	5b $[\gamma(4)\vee\gamma(5)]\wedge[\gamma(2)\wedge\gamma(3)]\wedge\gamma(6)=0$
6a $[\gamma(1)\vee\gamma(12)]\wedge[\gamma(9)\wedge\gamma(10)\wedge\gamma(2)=0$	6b $[\gamma(2)\vee\gamma(1)]\wedge[\gamma(4)\wedge\gamma(5)]\wedge\gamma(12)=0$
7a $\gamma_1(k)\wedge\gamma(k\pm2)\wedge\gamma(k\pm3)=0 \quad k=1$	7b $\gamma_1(k)\wedge\gamma(k\pm2)\wedge\gamma(k\pm3)=0 \quad k=1,2$
	8 $\gamma(2) = 1$, and $\chi'_{\gamma(2)} \neq 2$

Rules 1 through 3 remain unchanged for the isotropic algorithm; however, rules 4 through 8 are changed by symmetry as follows

9a $\gamma(5)\wedge\gamma(9)=0$

9b $\gamma(5)\wedge\gamma(9)=0$

10a $[\gamma(6)\vee\gamma(7)]\wedge[\gamma(8)\vee\gamma(9)]\wedge\gamma(10)=0$

10b $[\gamma(1)\wedge\gamma(12)]\wedge[\gamma(9)\wedge\gamma(10)]\wedge\gamma(8)=0$

11a $[\gamma(5)\vee\gamma(6)]\wedge[\gamma(8)\vee\gamma(9)]\wedge\gamma(14)=0$

11b $[\gamma(5)\vee\gamma(6)]\wedge[\gamma(8)\vee\gamma(9)]\wedge\gamma(10)=0$

12a $\gamma_1(k)\wedge\gamma(k\pm2)\wedge\gamma(k\pm3)=0,\ k=2,3$

12b $\gamma_1(k)\wedge\gamma(k\pm2)\wedge\gamma(k\pm3)=0,\ k=2$

13 $\gamma(6)=1$ and $\chi'_{\gamma(6)}\neq2$

Results and Discussion

A field in which thinning algorithms have found a wide application is that of character recognition, in which, prior to the encoding of a character's shape, it is necessary to reduce the original image to a line drawing. A good testing ground for the performances of the algorithms developed above would therefore be their application to alphanumeric characters. The following would thus be of interest; given a fixed rectangular array of points can these points, interconnected to form a different array structure, and in conjunction with an appropriate thinning algorithm, yield a more useful processor. Clearly, the ultimate usefulness of any preprocessor will depend upon its performance within an entire image-recognition system of which it only forms a part. However given that the desired result is a simple line drawing of the original image consisting of a minimal number of points it is possible to compare such algorithms.

The algorithms developed are all parallel in operation, and similar operations in each algorithm use similar instructions. A useful way of comparing their "cost" would be to compare the storage and processing time of each. Another factor worth comparing is the comparative data reduction resulting from each algorithm, bearing in mind that the densities of points in the arrays differ.

Figure 14(a) and (b) show some images operated upon by the thinning algorithms, and in Figure 15 the various

performance parameters pertaining to each algorithm are summarized.

It will be seen from Figure 14 that of all the resulting thinned images, those obtained using the triangular array contain the least number of points per image. This result is not unexpected, since on this array, the neighbors span the largest distance. The ratio of the maximum distances of any neighbor on the rectangular, hexagonal and triangular arrays is $1:\sqrt{2}:3/\sqrt{2}$ respectively. However the increased size of the basic window renders the processing on a triangular array - and thus the resulting image - very sensitive to edge irregularities, and still more important, to noise. The former effect is clearly demonstrated in the last image of Figure 14(b). From this point of view the hexagonal array is preferential, since all its neighbors, theoretically at least, are equidistant. The hexagonal array has the additional attraction that the processing time required is considerably less.

Both the triangular and the hexagonal arrays contain the same number of points - half the number of points contained within the original rectangular array. Yet despite the fact that the image on the triangular array contains the minimal number of points there is an additional argument against the use of triangular arrays to be considered. If the thinned image is to be chain encoded then the number of direction vectors, in the case of the triangular array, is twelve. Thus the maximum number of bits required to represent a single direction vector is four; this compares with three bits required for the other two arrays. Thus

from the point of view of storage or transmission of a complete resulting line drawing, the triangular array will only be useful if the number of points of the image thereon is less than three quarters the number of points in the image on the hexagonal array. The experiments show this is the case, yet all this is obtained at the expense of increased processing time.

The hexagonal grid also has advantages over the rectangular grid as far as processing time and data reduction are concerned. See [4] for a further discussion on the relative merits of these two types of arrays.

Conclusion

Thinning algorithms for various types of arrays were developed in this paper and compared experimentally. It is concluded that triangular arrays while being the most expensive in terms of processing time will yield an image with a minimal number of points. The algorithm operating with this array is, however, very sensitive to noise and edge irregularities. The hexagonal array offers a balance between the rectangular and the triangular arrays in that it requires almost the same storage, yields a midway average reduction value and has the shortest processing time.

References

[1] Rutovitz, D., "Pattern Recognition", Journal of the Royal Statistical Society, (A) Vol. 129 IV, pp. 504-530, 1966.

[2] Deutsch, E. S., "Some Comments on a Thinning Algorithm", British Computer Journal, Vol 12, No. 3, November 1969.

[3] Rosenfeld, A., "Connectivity in Digital Pictures", JACM, Vol. 17, No. 1, pp. 146-160, January 1970.

[4] Deutsch, E. S., "On Parallel Operations on Hexagonal Arrays". Proc. IEEE on Computers (Correspondence), 1970. To be published.

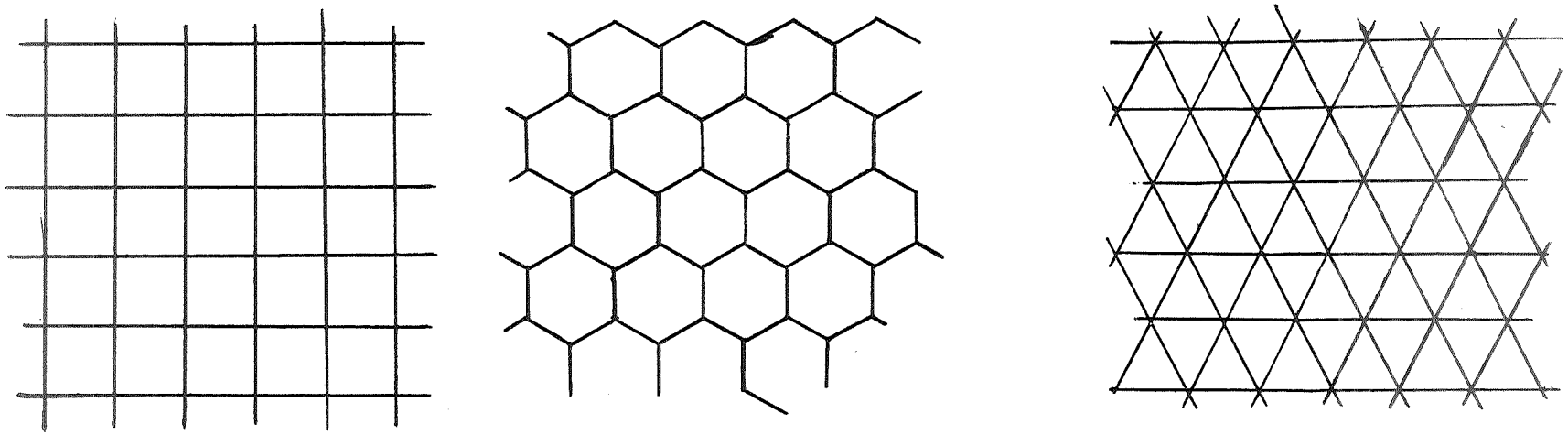


Figure 1

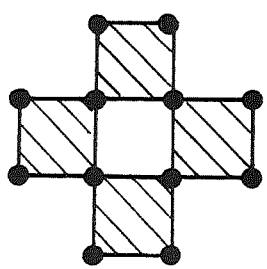


Figure 2

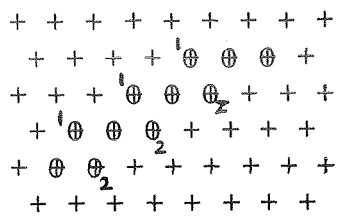


Figure 3

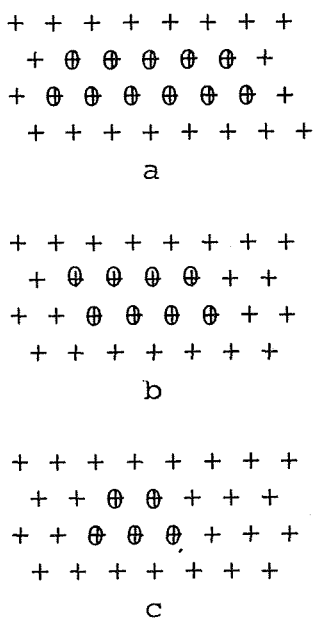


Figure 4

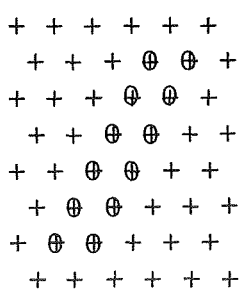


Figure 5

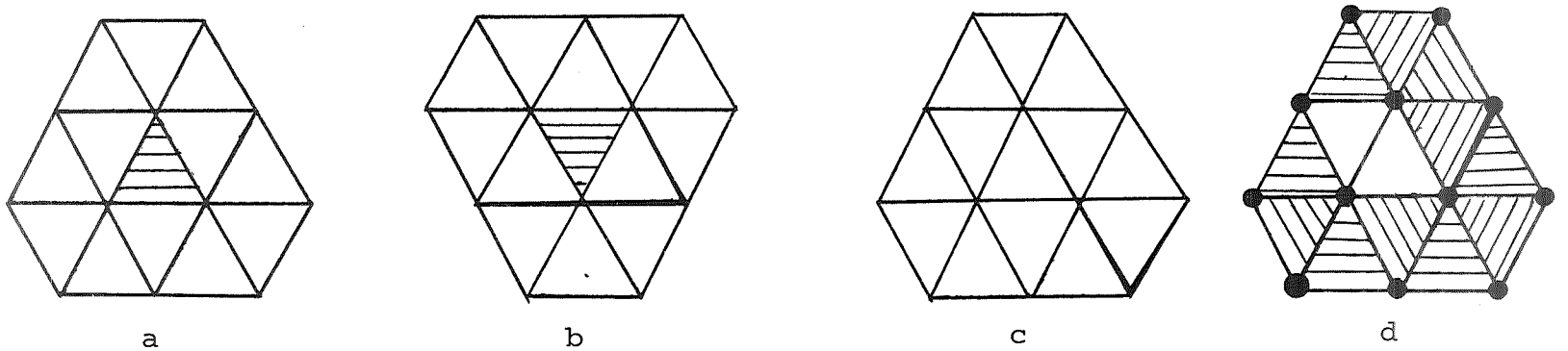


Figure 6

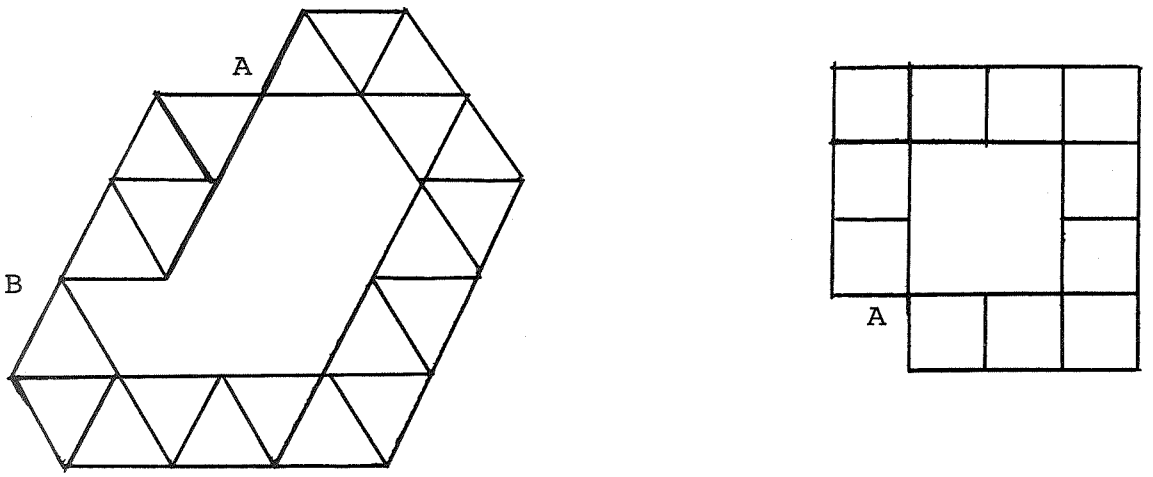
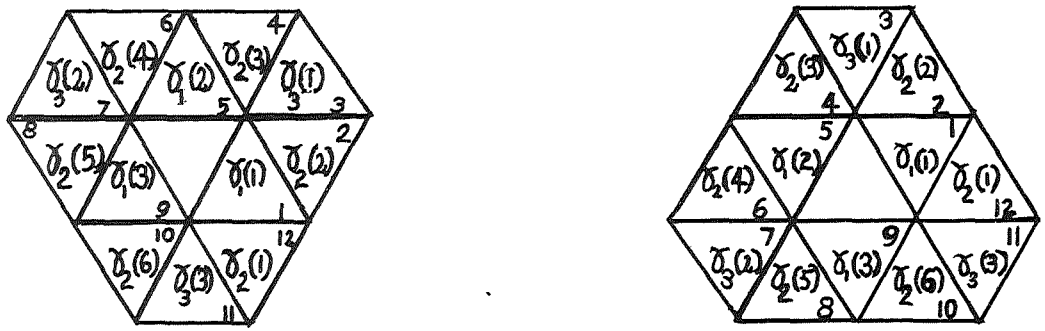
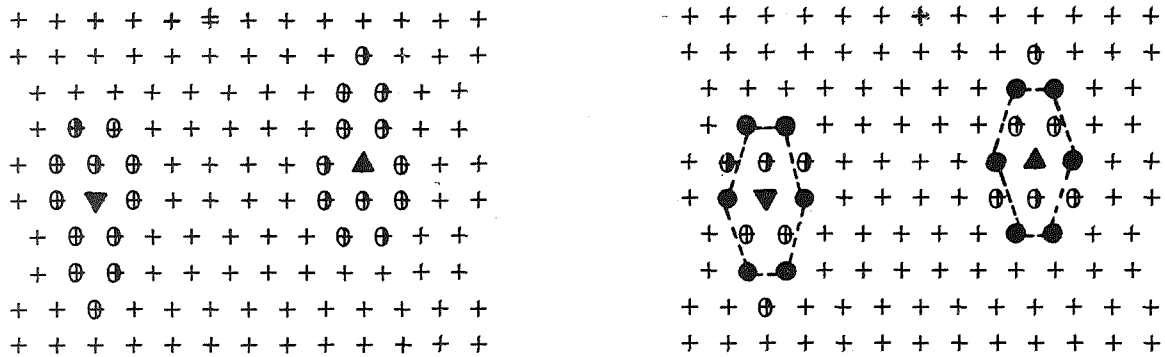


Figure 7



a b

Figure 8



a b

Figure 9

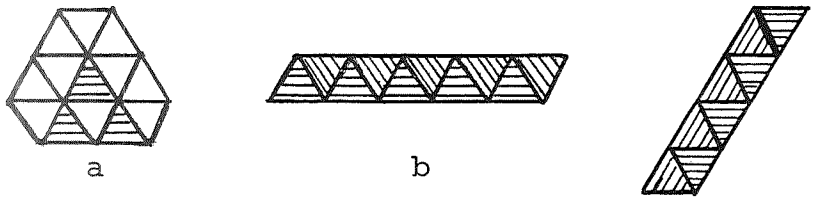


Figure 10

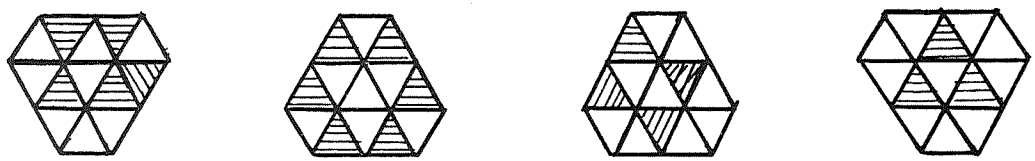


Figure 11

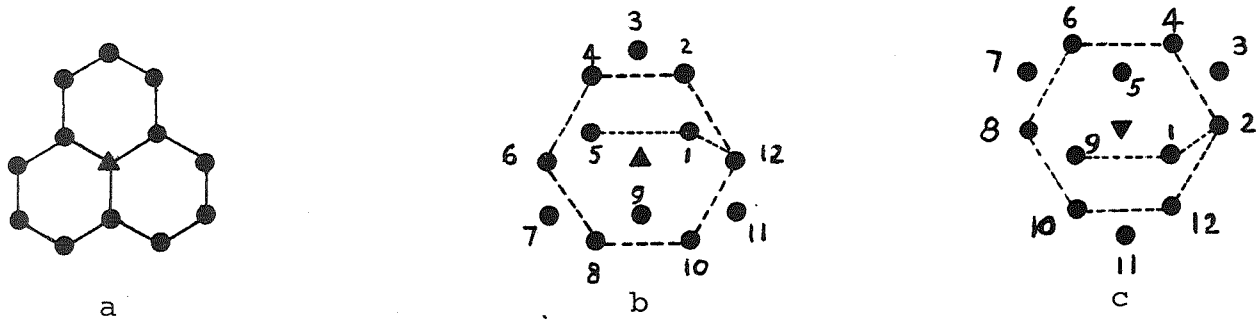


Figure 12

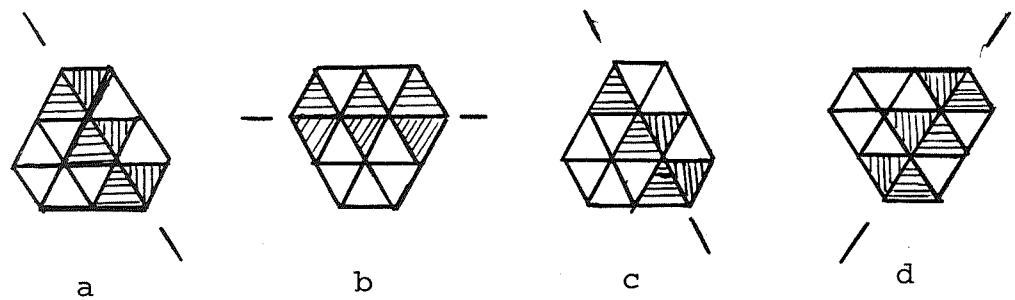
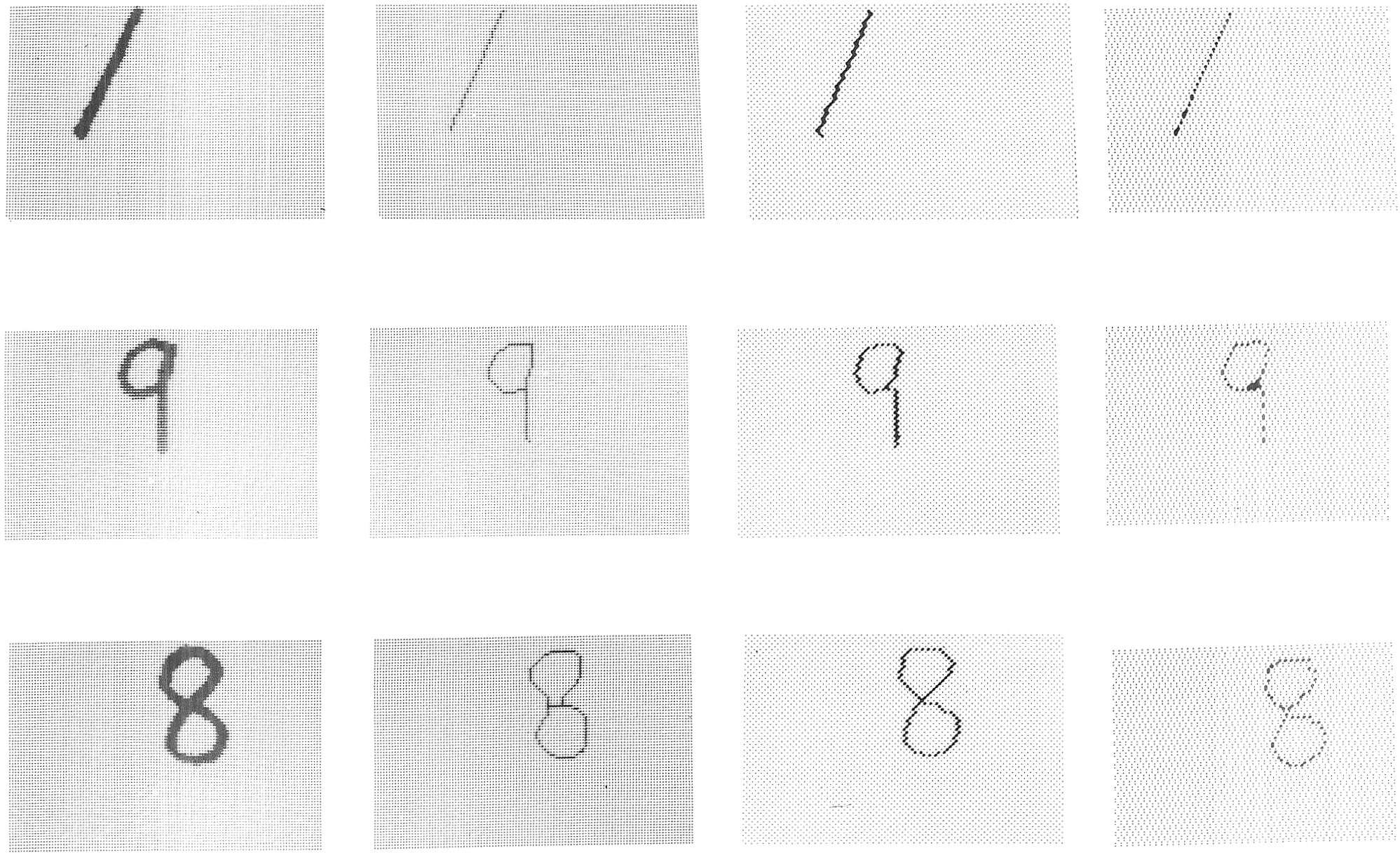
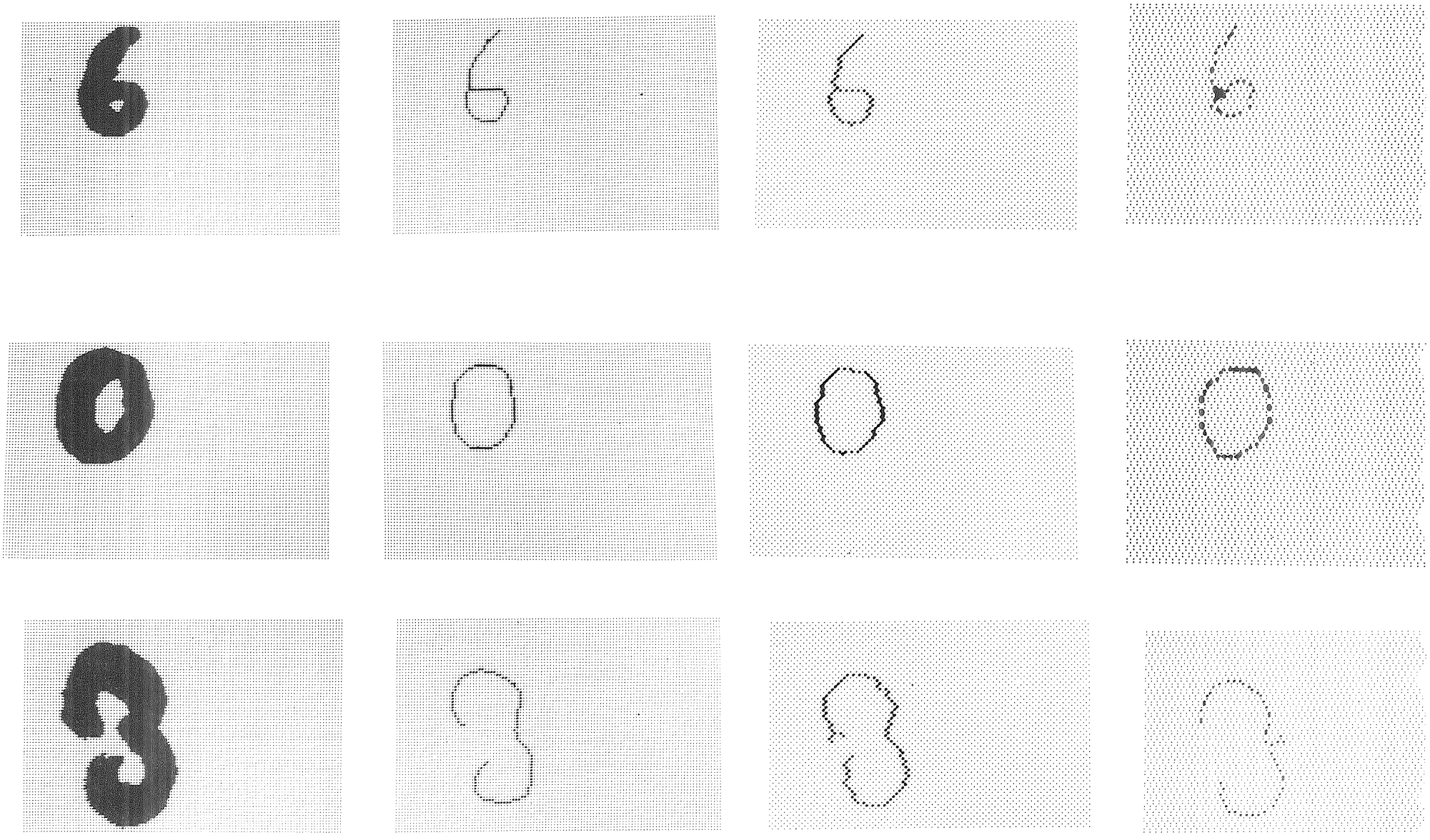


Figure 13



Original | Rectangular array | Hexagonal array | Triangular array

Figure 14(a)



Original | Rectangular array | Hexagonal array | Triangular array

Figure 14(b)

	Rectangular	Hexagonal	Triangular
Image Processing Time	1	0.5	1.8
No. of Core Locations	1	1.0	1.1
Average Image Reduction	1	1.7	0.8

Figure 15