

AD-A157 167

CAR-4
CS-1278

DAAK70-83-K-0018
May 1983

THE LABELED DISCRETE VORONOI DIAGRAM

Tsaiyun Phillips
Takashi Matsuyama

Center for Automation Research
University of Maryland
College Park, MD 20742

CENTER FOR AUTOMATION RESEARCH

DTIC FILE COPY

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

This document has been approved
for public release and sale; its
distribution is unlimited.



85 7 19 028

①

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
<i>per DTIC Form 50 on file</i>	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
<i>A-1</i>	

TR-CAR-4
TR-CS-1278

DAAK70-83-K-0018
May 1983

THE LABELED DISCRETE VORONOI DIAGRAM

Tsaiyun Phillips
Takashi Matsuyama

Center for Automation Research
University of Maryland
College Park, MD 20742



ABSTRACT

Generalized Voronoi diagrams of sets of digital curves are a helpful tool in picture analysis. In this paper, an algorithm for computing labeled Voronoi diagrams for digital straight line segments is given. Special emphasis was given to the use of a labeled Euclidean distance transform. This transform is the key feature of the proposed label propagation process. The proposed parallel algorithm for computing labeled Voronoi diagrams has time complexity $O(\max\{M, N\})$ for input pictures of size $N \times M$ using a mesh-connected array processor. The proposed serial algorithm for computing labeled Voronoi diagrams has time complexity $O(MN)$.

Additional keywords: image analysis; pixels.

↑

The support of the Defense Advanced Research Projects Agency and the U.S. Army Night Vision Laboratory under grant DAAK70-83-K-0018 (DARPA Order 3206) is gratefully acknowledged, as is the help of Janet Salzman in preparing this paper. The author also wishes to thank Reinhard Klette for comments on the final version of the paper.

1. Introduction

At some stage in the process of image analysis, a given image may be represented as a set of points, a set of lines, or a set of curves. Clusters of these sets often correspond to meaningful objects in the scene. In this paper, we introduce a labeled digital Euclidean distance transform for constructing labeled Voronoi polygons of a set of curve segments.

Consider a finite set P of points in the plane. One of the basic geometrical structures which can be used to characterize the shape of such a set P is a partition of the plane into a set of disjoint* polygons known in computational geometry as the (nearest-point) Voronoi diagram or the Dirichlet tessellation [1]. Here each polygon consists of those points of the plane that are closer to some given point of P than to any other point of P . The Voronoi diagram depends upon the particular metric that was used.

Voronoi diagram structures have been used in geography [2] as Thiessen polygons, in physics as Wigner-Seitz cells, and also as a data analysis tool in statistics [3]. Recently they have been used as an alternative way of extracting skeletons of objects [4,5] as well as for clustering and analyzing dot patterns in pattern recognition [6-8].

In the Euclidean plane with the Euclidean metric d , the Voronoi diagram of a set of points $P=\{p_1, p_2, \dots, p_n\}$ may be

*To be more exact, the interior sets of these polygons are disjoint, but different polygons may share an edge. Edges of polygons will be called Voronoi edges.

represented by a set of disjoint convex polygons $\{G_1, G_2, \dots, G_n\}$, where a point x of the real plane is in polygon G_i iff there is no other point p_j in P which is closer (according to metric d) to x than point p_i , for $1 \leq i \leq n$. Formally,

$$G_i = \{x : d(x, p_i) = \min\{d(x, p_j) : 1 \leq j \leq n\}\}, \text{ for } 1 \leq i \leq n.$$

Although Voronoi diagrams originally were designed for point sets in m -dimensional Euclidean space, they can be easily extended to sets whose elements are lines, curves, or other simple shapes. Let S be a planar compact point set, and let p be a point in the plane. Then $d(p, S)$ is defined to be the minimal Euclidean distance from point p to set S , i.e., $d(p, S) = \inf\{d(p, q) : q \in S\}$.

The Voronoi diagram for a set $\{P_1, P_2, \dots, P_n\}$ of lines or curves in Euclidean space can be defined as a set $\{G_1, G_2, \dots, G_n\}$ of closed point sets where

$$G_i = \{x \mid d(x, P_i) = \min\{d(x, P_j) : 1 \leq j \leq n\}\}, \text{ for } 1 \leq i \leq n.$$

Many algorithms have been introduced for computing Voronoi diagrams of point sets in two-dimensional Euclidean space [9-13]. The diagrams can be constructed by computing one polygon at a time (simple but slow), or by constructing and merging polygons using the divide-and-conquer technique which results in algorithms with time complexity $O(n \log n)$. Recently, $O(n \log n)$ algorithms have been proposed for computing the Voronoi diagram of a set of line segments or circles [4,5].

Matsuyama [14] has described an algorithm for extracting the medial axis of a simple polygon from the Voronoi diagram of disjoint boundary segments of the object; i.e., only an incomplete

border of the object is available as input. (For the definition of medial axis see [15], e.g.; roughly speaking, this is the set of all centers of maximal disks contained in the given object.) The algorithm in [14] uses a labeled Voronoi diagram to characterize geometric relations among line segments. As will be described in Section 2, this labeled Voronoi diagram can be used for various picture processing tasks.

The algorithm given in [14] is based on the continuous Euclidean plane. The objective of this paper is to describe an algorithm to compute a labeled Voronoi diagram in the digital plane where a set of digital straight line segments is given as input.

Our algorithm uses a specialized distance transform in the digital plane, as opposed to the usual distance transforms in digital image processing - i.e., city-block [15], chessboard, or octagonal (a combination of city-block and chessboard, which is generated by using them alternately at even and odd time steps) distance transforms.

The main problem with these distance transforms is their lack of invariance under rotation. This is illustrated in Figure 1. Figure 1a illustrates the effect of rotation on the Voronoi edges computed using the city block distance; Figure 1b is a similar illustration using the chessboard distance. The city-block, chessboard, and octagonal distance transforms, as well as the other digital distance transforms, in general also fail to produce connected medial axes.

A Euclidean digital distance transform is needed in computing point distances in the digital plane in order to solve the problem of the invariance under rotation and the problem of the connectivity of medial axes. Montanari [16] proposed a quasi-Euclidean distance transform for obtaining a connected skeleton on the digital space. Recently, Danielsson [17] gave an algorithm for computing a digital Euclidean transform of a given binary image. Figure 2 shows the Voronoi boundary resulting from this digital Euclidean transform, for the same input as used in Figure 1. As we can see, the shape of the resulting Voronoi boundary is invariant under rotation.

We shall define a special labeling scheme for the parts of line segments. We shall then show that if we use a suitably chosen distance transform to define the Voronoi diagram of a set of labeled digital line segments, we will be able to partition the pixels on the edges of the Voronoi polygons based on the labels of the neighbors of the Voronoi edges. The labeling scheme and partitioning algorithm are discussed in Section 2.

In Section 3, a general algorithm for computing the labeled Voronoi diagram of a set of labeled lines or curves in a given digital image is presented. In the algorithm, we first construct the Voronoi diagram for the labeled segments; then each pixel which is on the edge of a Voronoi polygon obtains a special label depending on the labels of the pixels in its neighborhood.

2. Labeling and Partitioning

The structure of Voronoi diagrams has found wide application in analyzing the shapes or the patterns of a given set of objects in either the Euclidean or digital plane. The area of a Voronoi region can be used to characterize the distance of an object from its neighbors, and the shape of the region can be used to determine the number of neighbors.

Lee [5] has used the structure of the Voronoi diagram as a method of computing the medial axis of a given polygon. The bisector defined by two objects is the locus of all points having equal distance from the two objects. Thus for a set of two objects the bisector and the Voronoi edge are identical, and for a set of three or more, all Voronoi edges are segments of bisectors. Lee shows that the geometric properties of bisectors may be used for extracting the medial axis of a polygon from its Voronoi diagram. In fact, the medial axis is a subset of the Voronoi diagram in the case of a simple polygon. Matsuyama [14] also studied the geometric properties of Voronoi edges and suggested a method for partitioning and labeling Voronoi edges.

Suppose we are given a pair of straight line segments in the Euclidean plane. In general, their Voronoi diagram is as in Figure 3, where the space is divided into two half planes (Voronoi regions) by a continuous curve (bisector or Voronoi edge). The Voronoi edge is composed of five segments at most. Using notation similar to that in [18], these segments are $\hat{B}(a,c)$, $\hat{B}((a,b),c)$, $\hat{B}((a,b),(c,d))$, $\hat{B}((a,b),d)$, and $\hat{B}(b,d)$.

Here a , b , c , and d denote the endpoints of the line segments, and (a,b) , (c,d) are their interiors, which are open line segments. $B(p,q)$ denotes the bisector of the two elements p and q . $B(\text{endpoint}, \text{endpoint})$ is a straight line perpendicular to the straight line containing these two endpoints, $B(\text{open-line-segment}, \text{open-line-segment})$ is the bisector of the angle defined by the two line segments, and $B(\text{endpoint}, \text{open-line-segment})$ is a parabola whose focus is the endpoint. By $\hat{B}(p,q)$ we denote the part of the Voronoi edge arising from elements p,q . Lee [5] discusses several geometric properties of the Voronoi diagram of a set of straight line segments.

As we can see from the example in Figure 3, the Voronoi diagram represents various geometric properties of the position and length of the given line segments which may be used for shape analysis. But since each Voronoi edge is composed of several segments representing different parts of the given line segments, it is convenient to regard it not as an individual unit for shape analysis, but as a collection of segments.

In general, Voronoi edges can be partitioned into three different classes of segments by using geometric labels for the segments $B(\text{endpoint}, \text{endpoint})$, $B(\text{endpoint}, \text{open-line-segment})$, and $B(\text{open-line-segment}, \text{open-line-segment})$. As illustrated in Figure 3, a Voronoi edge changes its label when it crosses over one of the straight lines which are perpendicular to the given line segments, and which cross these line segments at their endpoints.

Suppose we are given a pair of digitized straight line segments. In the conventional algorithm for computing the connected

Voronoi edge of these two line segments, first a unique label is assigned to all the pixels of each line segment. Then the labels are propagated using the given distance transform. The Voronoi edges consist of those pixels where different labels meet. (See Figure 4.)

Assume we want to label each Voronoi edge pixel according to its class - i.e., assign it to one of the three classes defined above. This labeling is impossible without further analysis if we do not discriminate the endpoints from the interior pixels in each line segment.

In our approach, first we label the endpoints of a given line segment differently from all other pixels interior to the line segment. The classification of a Voronoi edge pixel then becomes simple and can be done at the stage of Voronoi edge extraction. The details of labeling line segments and classifying Voronoi edges will be given in the following subsections. The general method for labeling (label propagation) is similar to that used in the conventional algorithm described above; the results depend upon the chosen distance transform. See Figure 5.

2.1 Labeling of digitized line segments

As input, a binary image $I[M,N]$ of size $M \times N$ is given, which contains a set of line segments (composed of pixels with value 1) on a background of pixels with value 0. As output, an image $L[M,N]$ of size $M \times N$ has to be computed containing the labeled segments. Our procedure can be described as follows:

a) The procedure for initial labeling:

Step 1: Apply a connected component algorithm [15] to image I , and give the pixels in each component a unique label.

Step 2: For all end-of-line pixels, add a large constant to their labels. The constant must be large enough that labels from the result of adding this constant will not be confused with other labels in the image.

This procedure can be modified to handle the case where the input image I contains digital curves. In this case, sharp corners will be treated as end-of-line pixels.

2.2 Classification of Voronoi edge points

Suppose we are given a set of digital straight line segments which are labeled according to the scheme in Section 2.1. Now, a certain distance transform algorithm (described in Section 3) is applied to these labeled line segments for label propagation. We shall call the image resulting from the application of this transform the computed labeled distance transform image $P[M,N]$. The pixels in a computed labeled distance transform image have certain labels denoting either an "open-line-segment" pixel or an "endpoint" pixel. The procedure for transforming image $L[M,N]$ into the computed labeled distance transform image $P[M,N]$ will be considered in Section 3.

For these labels in the computed labeled distance transform image we use the following definitions. Let LL be the $B(\text{open-line-segment}, \text{open-line-segment})$ pixel label, PL the $B(\text{endpoint}, \text{open-line-segment})$ pixel label, PP the $B(\text{endpoint}, \text{endpoint})$ pixel label, and BB the label for a branching point on a Voronoi edge. Furthermore, let BG be the background pixel label, and OJ the pixel label for the original line segments.

Now, for classifying an arbitrary pixel in the computed labeled distance transform image P , the following definitions are used:

- 1) Two labels ℓ_1, ℓ_2 in a computed labeled distance transform image are said to be equivalent, $\ell_1 \cong \ell_2$, iff ℓ_1 and ℓ_2 belong to the same line segment; i.e., an endpoint and an interior pixel from the same line segment are equivalent.

- 2) A pixel in a computed labeled distance transform image is a Voronoi edge pixel iff there are two or more non-equivalent labels in its neighborhood, where the definition of neighborhood depends upon the chosen distance transform.
- 3) A pixel in a computed labeled distance transform image is a BB pixel iff there are more than two non-equivalent labels in its neighborhood.
- 4) A pixel in a computed labeled distance transform image is an LL pixel iff there are exactly two non-equivalent open-line-segment labels in its neighborhood.
- 5) A pixel in a computed labeled distance transform image is a PP pixel iff there are exactly two non-equivalent endpoint labels in its neighborhood.
- 6) A pixel in a computed labeled distance transform image is a PL pixel iff there are one open-line-segment label ℓ_1 and one endpoint label ℓ_2 , $\ell_1 \neq \ell_2$, in its neighborhood.
- 7) A pixel in a computed labeled distance transform image is a BG pixel iff it is not a Voronoi edge pixel, and if it is not on any of the original line segments.
- 8) A pixel in a computed labeled distance transform image is an OJ pixel iff it is on one of the original line segments.

The classification procedure is as follows:

As input, a computed labeled distance transform image $P[M,N]$ is given, which was obtained by applying the given distance

transform to the image $L[M,N]$. The image $L[M,N]$ was computed according to the procedure in Section 2.1.

As output, a labeled Voronoi diagram $V[M,N]$ will be determined. Our procedure can be described as follows:

For each pixel $P[i,j]$ in P do:

 Check the neighborhood of $P[i,j]$ and assign the proper label to $V[i,j]$ according to the definitions of Voronoi edge classification given above.

In Figure 8, we show the labeled Voronoi edge of the pair of straight line segments that was used in Figure 5. In the next section, we shall describe various possibilities for using distance transforms for computing the image $P[M,N]$ from the image $L[M,N]$.

3. The labeled distance transform image

Informally, the process of generating the Voronoi diagram of a set of objects in the Euclidean plane can be described by a "grassfire procedure" (like that used for computing the medial axis [19]) which operates as follows. Imagine the space is filled with some flammable material and a fire is started at each object (line segment, point, curve, ...). If an object consists of more than a single point, the fire is started at all points on the object at once. We may assume each object has its own color of fire. The fire spreads in all directions with equal speed. The locations where fire fronts from different objects meet together form the Voronoi diagram of the objects.

The computation of a digital Voronoi diagram is closely related to that of the medial axis transform. The main differences are that the medial axis transform is usually defined for a closed boundary of an object, and that the points on the medial axis are labeled with certain distance values given by the transform. The Voronoi diagram, on the other hand, is defined for a set of disjoint objects, and the points on the Voronoi edges are normally not labeled with distance values.

In the two-dimensional digital plane, the Voronoi diagram of a given digital image may be computed as follows:

- 1) Labeling : Give each object a unique label
- 2) Label propagation: Propagate all labeled pixels on the object boundaries in predetermined directions until different labels meet.

- 3) Edge extraction: Extract the Voronoi edges at the places where different labels meet together.

Normally, in this procedure either the city-block (4-direction) or the chessboard (8-direction) distance transform is used in label propagation (step 2). The well-known problem when we use these distance transforms is non-invariance under rotation, as we saw in Figure 1. In Figures 6a, 7a, and 8a, the Voronoi label expansions of a pair of straight line segments are shown after four iterations using three different distance transforms. The final expansions are shown in Figures 6b, 7b, and 8b.

As we can see from these Figures, the labels expanded from the objects form either box-like or diamond-like shapes. These shapes are quite different from the circular shape that would be obtained by using Euclidean distance. This problem is less significant if octagonal distance is used in the computation of label propagation, but the octagonal expansion is still significantly different from the Euclidean expansion.

In the grassfire procedure as we have informally defined it for the Voronoi diagram, the fire should spread from the object in all directions with equal speed. Thus even in a digital image, we want to define a propagation process in which the pixels on objects expand their labels with equal speed, i.e., expand one unit distance in every direction at each step. Next, we will describe a parallel algorithm that expands labels in this way.

3.1 The parallel algorithm

The labeled digital Euclidean distance transform algorithm (LED) takes a set of labeled line segments as input image; these line segments are given in image L . The line segments in L are labeled according to our procedure in Section 2.1. At the beginning of the computation, LED translates the input image L into a vector image, V , as follows.

a) Initial step:

A pixel $L(i,j)$ in L will be represented as a vector of three components in V , $V(i,j)=(x,y,l)$, where x , y , and l are integers. The Euclidean distance, $d=\sqrt{x^2 + y^2}$, is the distance from pixel $L(i,j)$ to its nearest (according to the Euclidean distance d) object, say $L(h,k)$, such that $|i-h|$ is equal to x , $|k-j|$ is equal to y , and $L(h,k)$ has label l . Initially, $V(i,j) = (0,0,L(i,j))$ iff $L(i,j)$ is not zero (not a background pixel) and $V(i,j) = (\$, \$, 0)$ if $L(i,j)$ is zero, where $\$$ stands for a very large number.

In the algorithm we use an expansion counter W_count to control the label expansion speed. W_count is set to zero at the beginning. Without this expansion counter, the distances of the label propagations would not be correct and the resulting Voronoi edges would have the same shapes as if we used the chess-board distance transform. The label expansion resulting from not using W_count is illustrated in Figures 9a and 9b.

For each parallel step of the label expansion procedure, W_count will be incremented by 1, and each $V(\$, \$, 0)$ will be modified as follows:

b) Label expansion step (while any $V(\$,\$,0)$ remains in V):

Let V_0 be the pixel $V(i,j)$ with vector $V(x_0,y_0,l_0)$;

let $V_1, \dots, V_8$ with $V(x_1,y_1,l_1), \dots, V(x_8,y_8,l_8)$ be its 3×3 neighbors:

V_5	V_1	V_6		(x_5,y_5,l_5)	(x_1,y_1,l_1)	(x_6,y_6,l_6)
V_3	V_0	V_4	$\equiv$	(x_3,y_3,l_3)	(x_0,y_0,l_0)	(x_4,y_4,l_4)
V_7	V_2	V_8		(x_7,y_7,l_7)	(x_2,y_2,l_2)	(x_8,y_8,l_8)

In the algorithm we compute the values, $val(x,y) = (x*x) + (y*y)$, for all of the following eight pairs (x,y) :

(x_5+1,y_5+1) (x_1+1,y_1) (x_6+1,y_6+1)

(x_3, y_3+1) (x_4, y_4+1)

(x_7+1,y_7+1) (x_2+1,y_2) (x_8+1,y_8+1)

and let k be such that $val(x,y)$ is minimal for the corresponding pair (x,y) in comparison to all the other 7 computed values. (If k is not uniquely defined in this way then we select k according to larger x -component, e.g.) If (x,y) is the pair giving rise to k , and if $\text{SQRT}(val(x,y))$ is less or equal to $W_count + \text{SQRT}(2) - 1$, then the center pixel, $V(i,j)$, will obtain the value (x, y, l_k) ; otherwise there will no change for $V(i,j)$. For example, if x is x_5+1 and y is y_5+1 , then $V(i,j)$ will be $V(x_5+1,y_5+1,l_5)$. If x is x_1+1 and y is y_1 , then $V(i,j)$ will be $V(x_1+1,y_1,l_1)$.

This parallel label propagation stops when there is no value $V(\$,\$,0)$ remaining in the image V , i.e., it stops after $\max\{M,N\}$ steps at most.

c) Edge extraction step:

The Voronoi edges can be marked by checking the label changes in the third component of each $V(i,j)$; see Section 2.2.

The labeled distance transform image resulting from this algorithm can be used for many purposes. The Voronoi diagram of input image I can be obtained by outputting all pixels marked as the Voronoi edge pixels. Furthermore, the Voronoi diagram now contains information about the distances to the given objects which may be useful for reconstruction or shape analysis. The main purpose of this algorithm is to make it possible to classify the Voronoi edges into the three classes described in Section 2.

The vector representation for this algorithm in image V is a modification of the two component vectors which were used in [17] by Danielsson for the computation of the "Euclidean distance mapping." For sequential computation, Danielsson used a vector of two components for computing distance transforms in binary digital images in two passes, with two picture scans in each pass (first pass from top to bottom, second pass from bottom to top). Our algorithm can also be implemented in two passes. This will be described in Section 3.2.

In Figures 8a and 8b, an example of our algorithm is shown (for four iterations and for the final expansion). Note that when our algorithm is performed in parallel, it has time complexity $O(\max\{M,N\})$ on a mesh-connected array processor.

3.2 The sequential algorithm

The steps in the sequential algorithm are described in the same order as we used for the parallel algorithm in Section 3.1. An important feature of the sequential algorithm in comparison to the parallel algorithm is that we do not use a step counter.

a) Initial step:

The sequential algorithm translates the input image L into the vector image V and initializes each vector $V(i,j)$ just as the parallel algorithm does.

b) First picture pass:

Let V_0 be the pixel $V(i,j)$ with vector $V(x_0,y_0,l_0)$;
let $V_1, \dots, V_5$ with $V(x_1,y_1,l_1), \dots, V(x_5,y_5,l_5)$ be its 2×3 neighbors:

$$\begin{array}{ccc} V_4 & V_1 & V_5 \\ V_2 & V_0 & V_3 \end{array} \equiv \begin{array}{ccc} (x_4,y_4,l_4) & (x_1,y_1,l_1) & (x_5,y_5,l_5) \\ (x_2,y_2,l_2) & (x_0,y_0,l_0) & (x_3,y_3,l_3) \end{array}$$

1) Scan from left to right:

We compute the values $val(x,y) = (x*x) + (y*y)$ for all of the following five pairs (x,y) :

$(x_4+1, y_4+1) \quad (x_1+1, y_1) \quad (x_5+1, y_5+1)$

$(x_2, y_2+1) \quad (x_0, y_0)$

Let k be such that $val(x,y)$ is minimal. Then pixel $V(i,j)$ gets the value (x,y,l_k) .

2) Scan from right to left:

This is analogous to the left-to-right scan for the following five pairs (x,y) :

$(x_4+1, y_4+1) \quad (x_1+1, y_1) \quad (x_5+1, y_5+1)$

$(x_0, y_0) \quad (x_3, y_3+1)$

and with the same replacement rule for $V(i,j)$.

c) Second picture pass:

Let V_0 be pixel $V(i,j)$ with vector $V(x_0,y_0,l_0)$, and let $V_1, \dots, V_5$ with $V(x_1,y_1,l_1), \dots, V(x_5,y_5,l_5)$ be its 2×3 neighbors in the following positions:

$$\begin{array}{ccccc} V_2 & V_0 & V_3 & \equiv & (x_2,y_2,l_2) \ (x_0,y_0,l_0) \ (x_3,y_3,l_3) \\ V_4 & V_1 & V_5 & & (x_4,y_4,l_4) \ (x_1,y_1,l_1) \ (x_5,y_5,l_5) \end{array}$$

1) Scan from right to left, using the pairs

$$\begin{array}{ccc} (x_0, & y_0) & (x_3, & y_3+1) \\ (x_4+1, & y_4+1) & (x_1+1, & y_1) & (x_5+1, & y_5+1) \end{array}$$

2) Scan from left to right, using the pairs

$$\begin{array}{ccc} (x_2, & y_2+1) & (x_0, & y_0) \\ (x_4+1, & y_4+1) & (x_1+1, & y_1) & (x_5+1, & y_5+1) \end{array}$$

d) Edge extraction step:

The Voronoi edges can now be marked by checking the label in the third component of each $V(i,j)$.

The serial algorithm for computing the labeled Voronoi diagram has time complexity $O(M \times N)$, altogether. In Figure 10, the application of the algorithm is illustrated for an input image containing nine digital straight line segments.

4. Summary

In this paper, a special labeling scheme for a set of digital line segments was explained. This labeling scheme can be extended to an arbitrary set of digital curves. The ends and any sharp corners of a given digital curve can be marked analogously to the ends of digital straight lines. For example, Fischler [20] introduced a labeling scheme for marking the "critical" points for a set of closed object boundaries. (His labeling scheme was used to obtain skeletons from noisy object boundaries.) In the next step, by propagation of the labels on the digital curves we can partition the generalized Voronoi diagram of the set of curves according to their geometric properties. The partitioned Voronoi edges can then be used for further shape analysis.

For the label propagation process, we gave a parallel algorithm for realizing a Euclidean "grassfire" label expansion in a digital image. The serial implementation of this parallel algorithm was also explained. The resulting sequential algorithm is closely related to the "Euclidean distance mapping" in [17]. The errors in comparison to the ideal Euclidean distance are very small and can be ignored (for the exact error see [17]). The labeled Euclidean Voronoi diagrams as computed in this paper will be used for characterizing the shapes of digital curves - e.g., for completion or reconstruction of broken boundaries.

References

1. Shamos, M. I., "Computational Geometry," Ph.D. thesis, Yale University, May 1978.
2. Rhynsburger, D., "Analytic delineation of Thiessen polygons," Geographical Analysis, Vol. 5, April 1973, pp. 133-144.
3. Sibson, T., "The Dirichlet tessellation as an aid in data analysis," Scandinavian Journal of Statistics, Vol. 7, 1980, pp. 14-20.
4. Kirkpatrick, K. G., "Efficient computation of continuous skeletons," Proc. 20th Annual Symposium on the Foundations of Computer Science, 1979, pp. 18-27.
5. Lee, D. T., "Medial axis transformation of a planar shape," IEEE Trans., Vol. PAMI-4, No. 4, 1982, pp. 363-369.
6. Ahuja, N., "Dot pattern processing using Voronoi neighborhoods," IEEE Trans., Vol. PAMI-4, No. 3, 1982, pp. 336-343.
7. Tuceryan, M. and N. Ahuja, "Segmentation of dot patterns containing homogeneous clusters," Proc. 6th ICPR, 1982, pp. 392-394.
8. Fairfield, J., "Segmenting dot patterns by Voronoi diagram concavity," IEEE Trans., Vol. PAMI-5, No. 1, 1983, pp. 104-110.
9. Green, P. J. and R. Sibson, "Computing Dirichlet tessellations in the plane," The Computer Journal, Vol. 21, 1978, pp. 168-173.
10. Horspool, R. N., "Constructing the Voronoi diagram in the plane," Technical Report No. SOCS 79.12, School of Computer Science, McGill University, July 1979.
11. Shamos, M. I., "Geometric complexity," Proc. 7th ACM Symposium on the Theory of Computing, May 1975, pp. 224-233.
12. Brassel, K. E. and D. Reif, "A procedure to generate Thiessen polygons," Geographical Analysis, Vol. 11, July 1979, pp. 289-303.
13. Shamos, M. I. and D. Hoey, "Closest-point problems," Proc. 16th Annual Symposium on the Foundations of Computer Science, October 1975, pp. 151-162.
14. Matsuyama, T. and T. Phillips, "Extracting medial axes from Voronoi diagrams of boundary segments: An alternative method for closed boundary detection," Technical Report, Center for Automation Research, University of Maryland, College Park, MD, 1983, in preparation.

15. Rosenfeld, A. and A. Kak, Digital Picture Processing, Academic Press, NY, 1976.
16. Montanari, U., "A method for obtaining skeletons using a quasi-Euclidean distance," J. ACM, vol. 15, No. 4, 1968, pp. 602-624.
17. Danielsson, P. E., "Euclidean distance mapping," Computer Graphics Image Processing, Vol. 14, 1980, pp. 227-248.
18. Lee, D. T. and R. L. Drysdale, "Generalization of Voronoi diagrams in the plane," SIAM J. Computing, Vol. 10, No. 1, 1981, pp. 73-87.
19. Blum, H., A transformation for extracting new descriptors of shape, in Models for the Perception of Speech and Visual Form (W. Wathen-Dunn, Ed.), pp. 362-380, MIT Press, Cambridge, MA, 1967.
20. Fischler, M. A. and P. Barrett, "An iconic transform for sketch completion and shape abstraction," Computer Graphics Image Processing, Vol. 13, 1980, pp. 334-360.

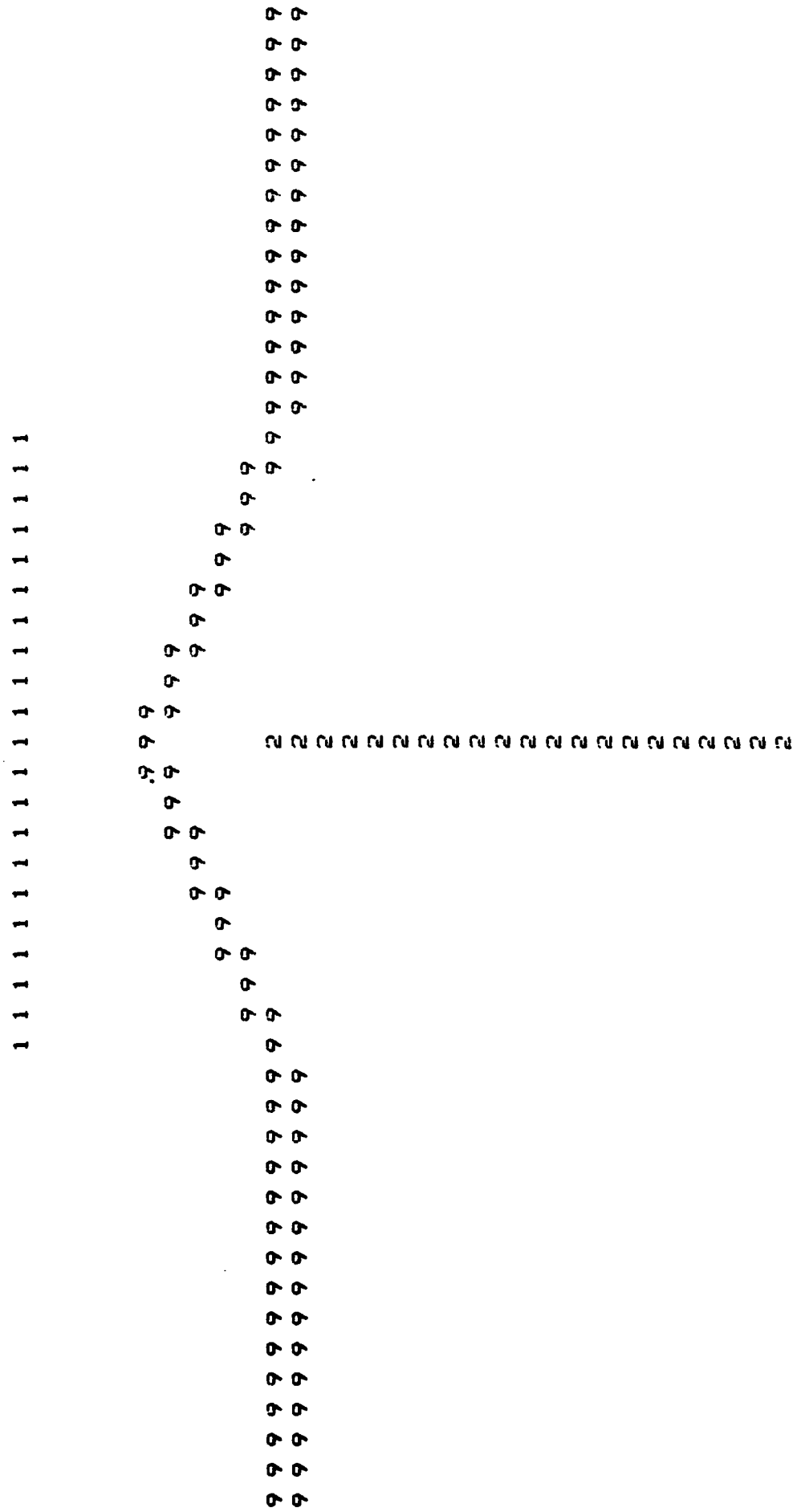


Fig. 1a. Digital Voronoi edge (labelled "9") for a pair of digital straight lines using the cityblock distance transform.

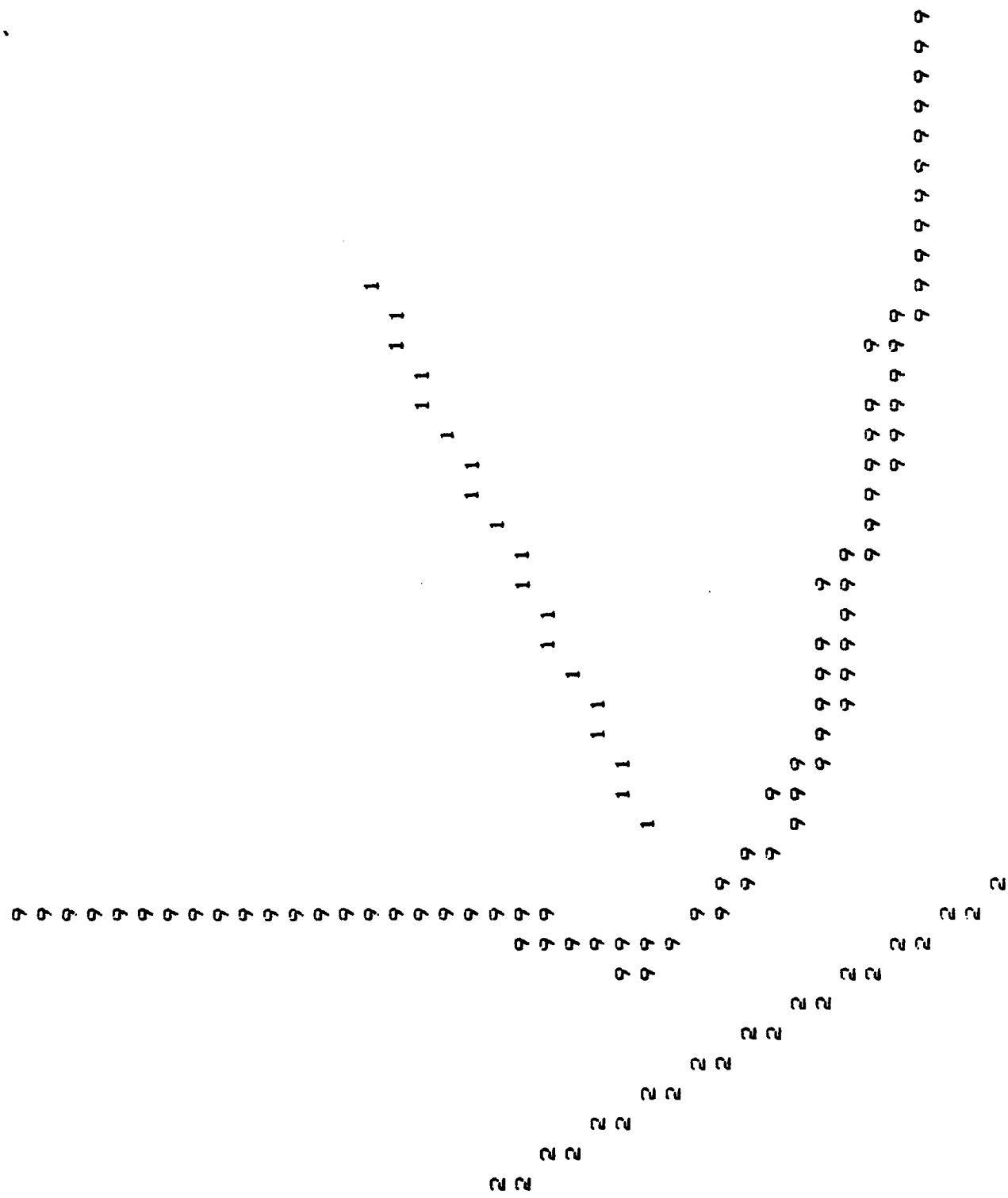


Figure 1a, cont'd.

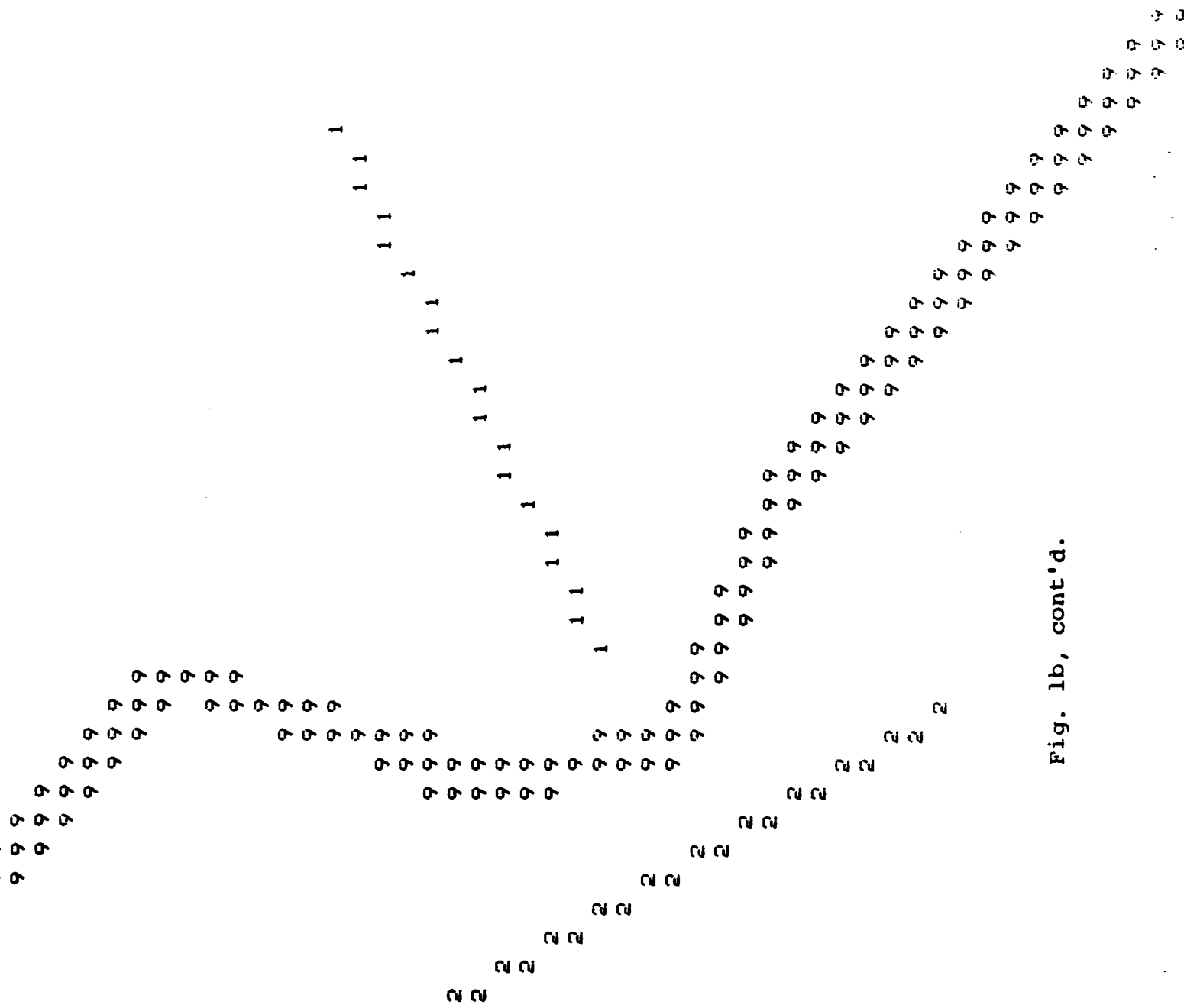


Fig. lb, cont'd.

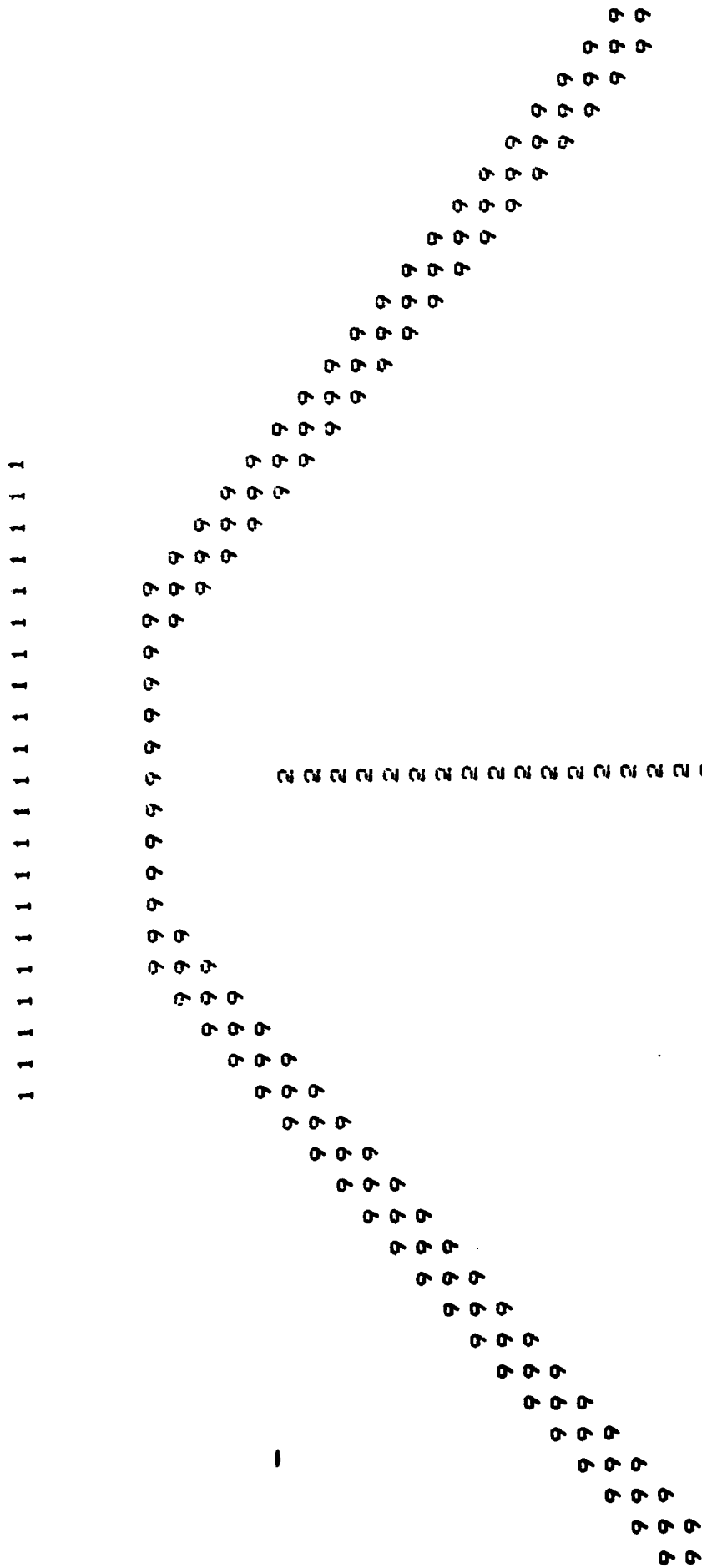


Fig. lb. Analogous to Fig. la using the chessboard distance transform.

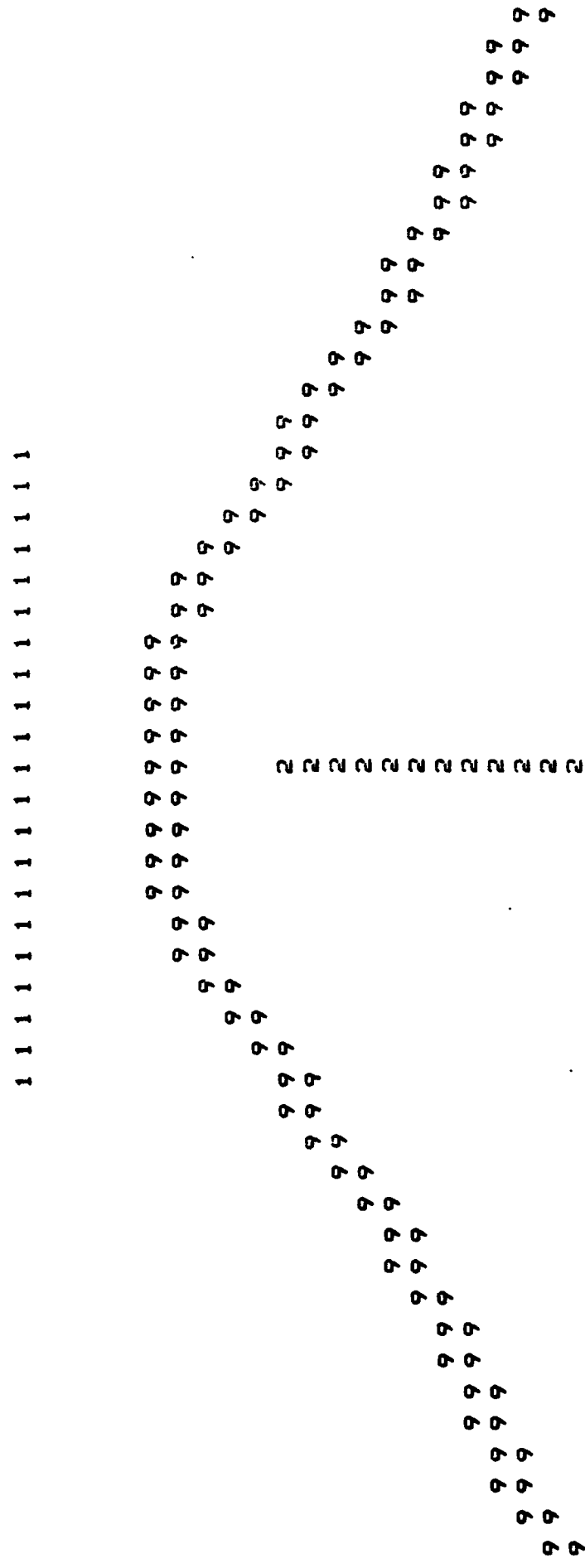


Fig. 2. Analogous to Figs. 1a-b using the Euclidean distance transform.

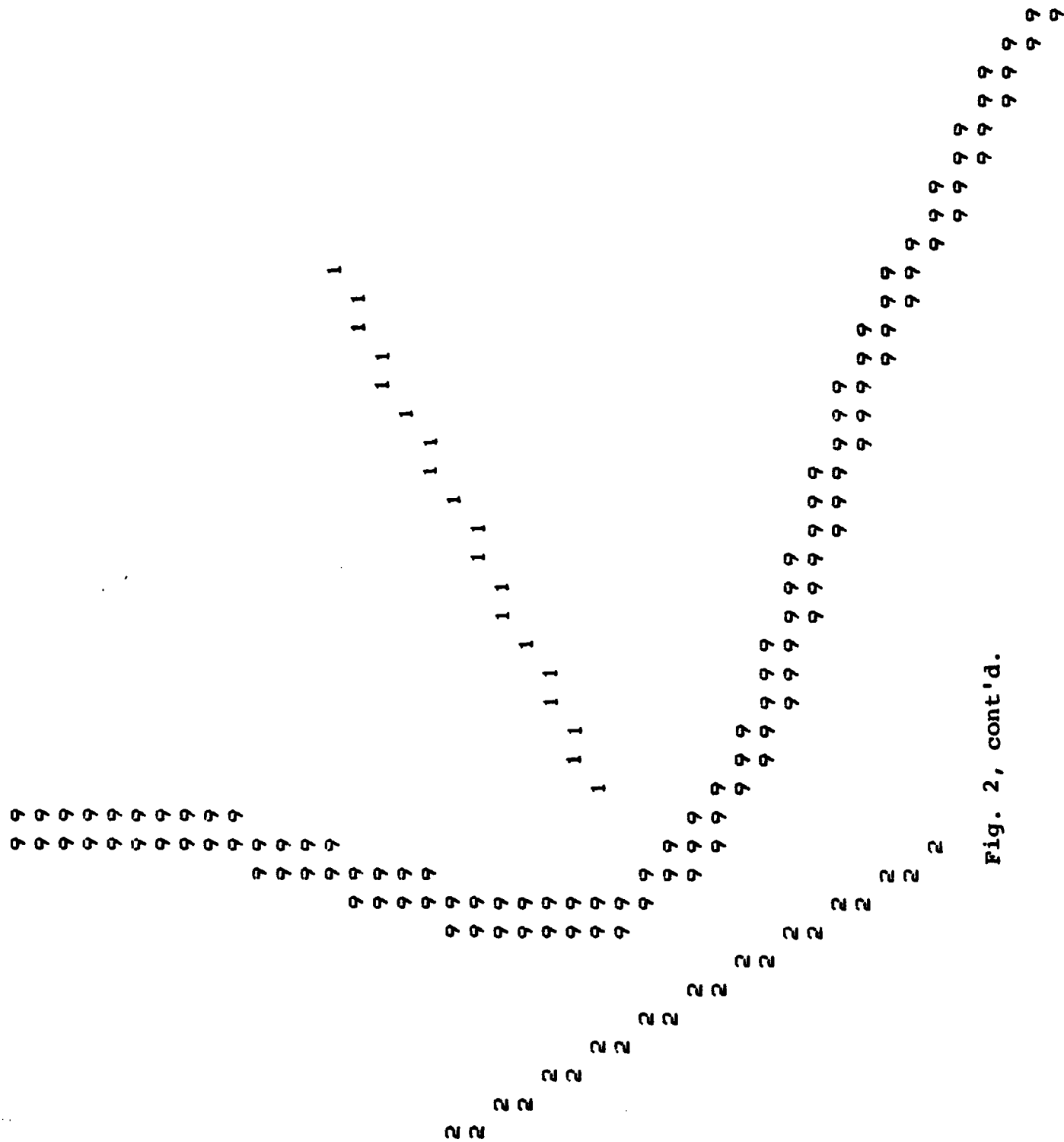


Fig. 2, cont'd.

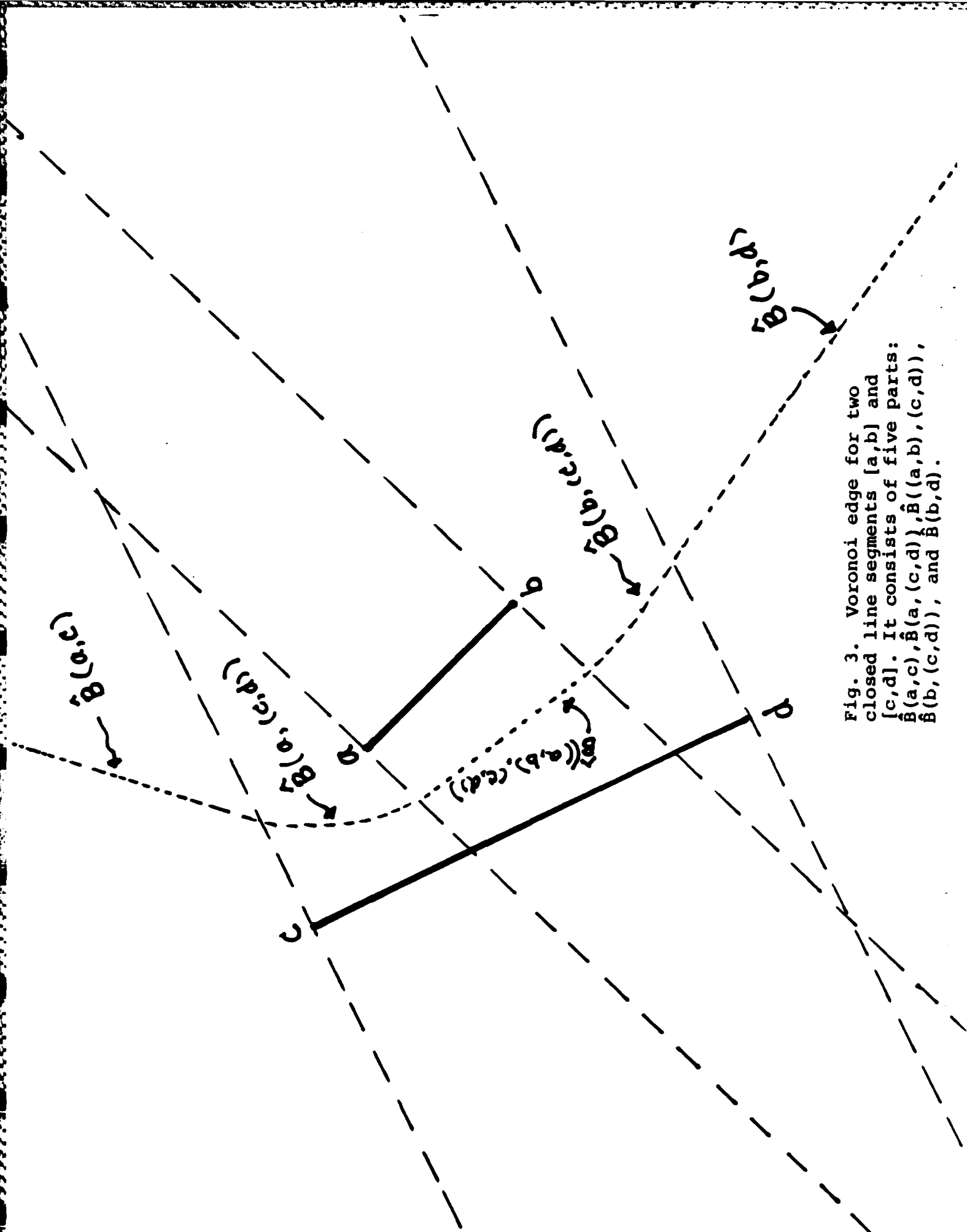


Fig. 3. Voronoi edge for two closed line segments $[a,b]$ and $[c,d]$. It consists of five parts: $\hat{B}(a,c)$, $\hat{B}(a,(c,d))$, $\hat{B}((a,b),(c,d))$, $\hat{B}(b,(c,d))$, and $\hat{B}(b,d)$.

[illegible]

Fig. 4a. Illustration of the conventional algorithm for computing the Voronoi edge by uniform labelling of the given line segments, using the city block distance transform.

Fig. 4b. Analogous to Fig. 4a, using the chessboard distance transform.

Fig. 4b. Analogous to Fig. 4a, using the chessboard distance transform.

[illegible]

Fig. 5a. Illustration of the modified algorithm proposed in this paper using labelling of the endpoints and interiors of the line segments, for the city block distance transform.

Fig. 5b. Analogous to Fig. 5a, using the chessboard distance transform.

Fig. 5c. Analogous to Figs. 5a and 5b, using the Euclidean distance transform. Note that here, unlike Figs. 5a and 5b, the ends of the Voronoi edge correspond to P-P interactions.

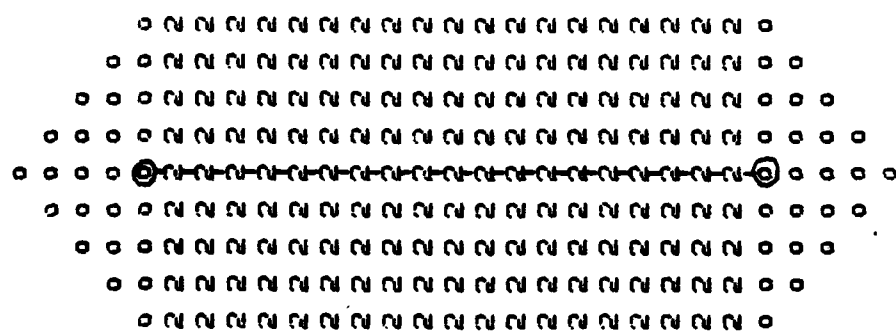
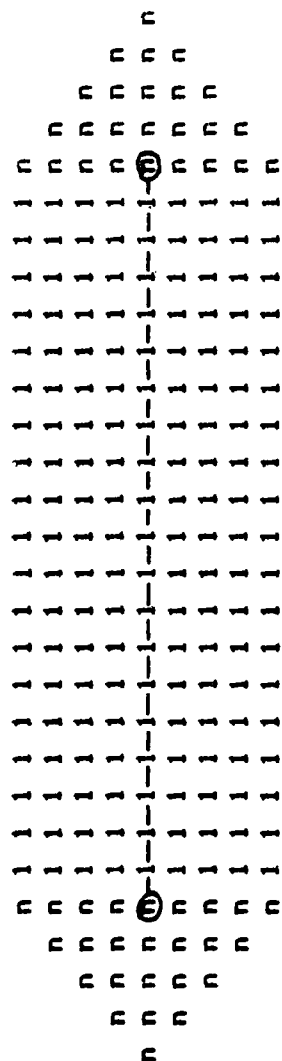
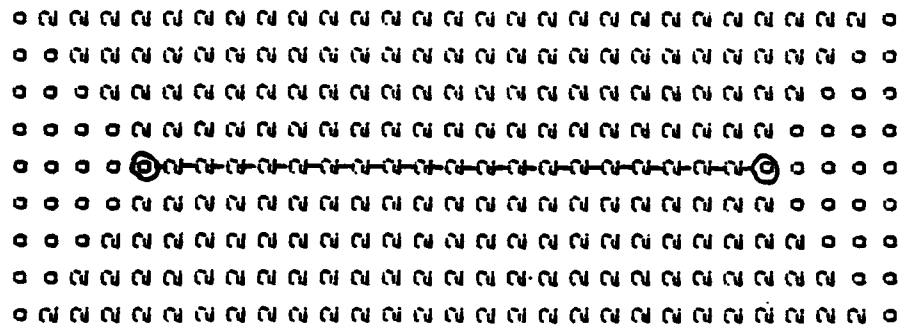
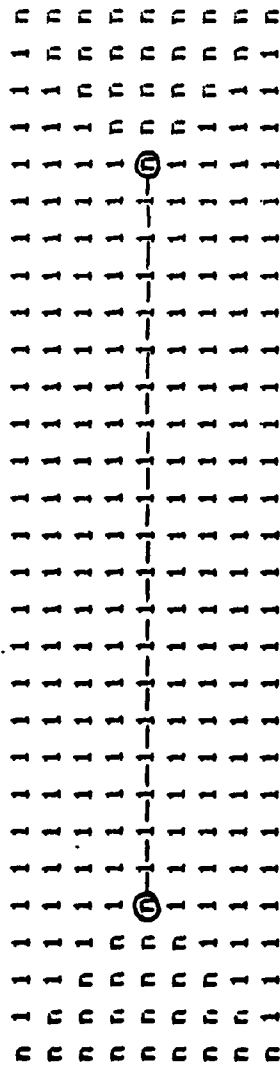


Fig. 6a. Label expansion after 4 iterations using the city block distance trans-form.



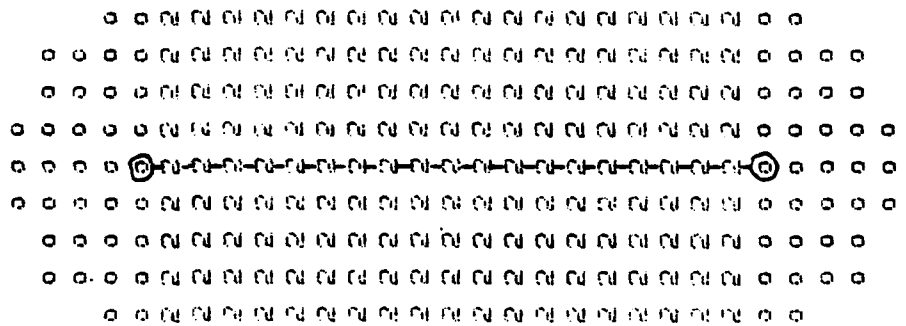
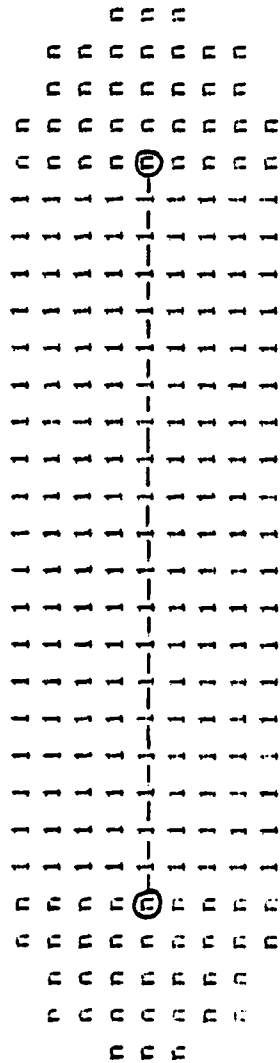


Fig. 8a. Analogous to Figs. 6a and 7a using the labelled Euclidean distance transform.

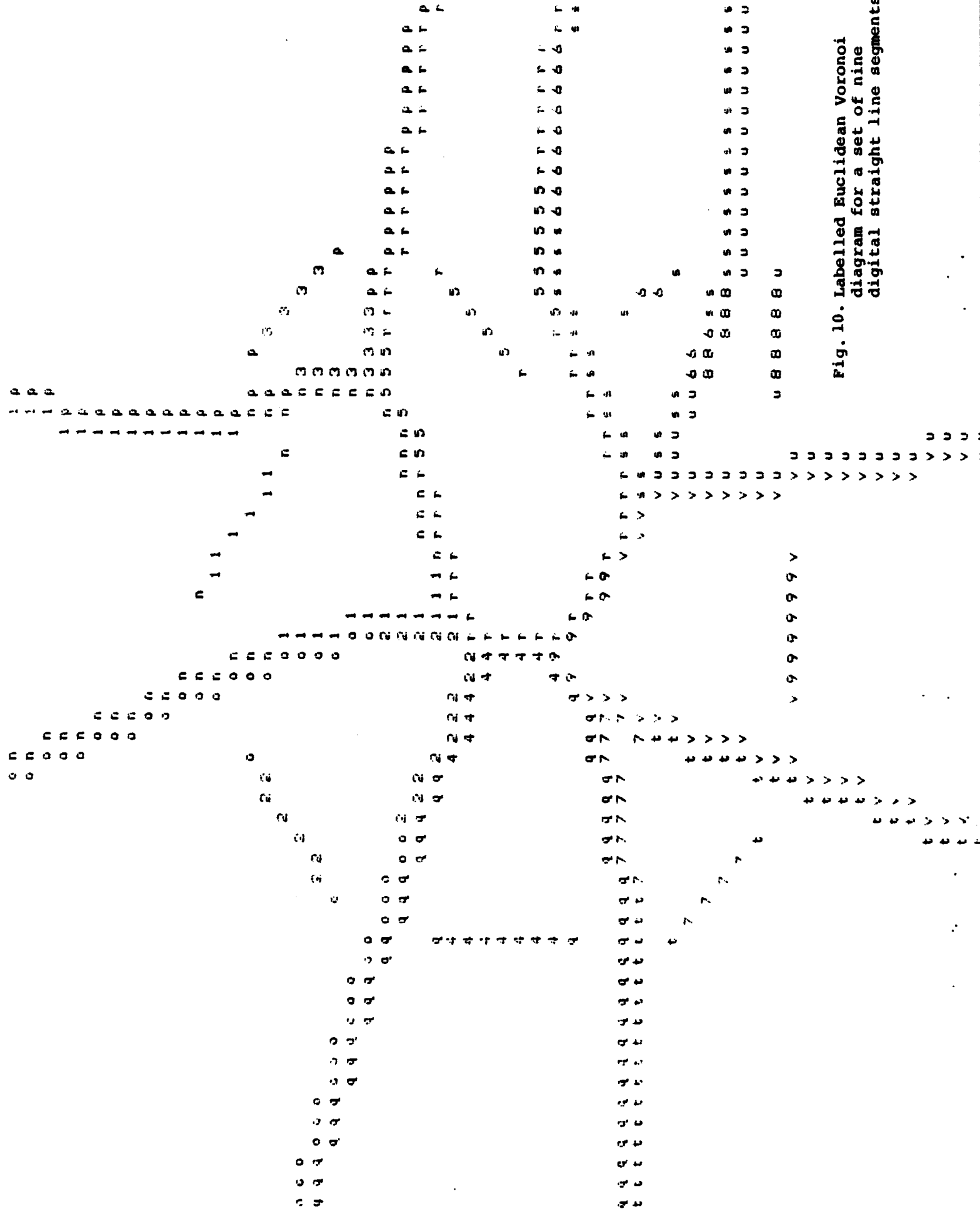


Fig. 10. Labelled Euclidean Voronoi diagram for a set of nine digital straight line segments.