January 1987 UILU-ENG-87-2206

COORDINATED SCIENCE LABORATORY
College of Engineering

EXTRACTING PERCEPTUAL STRUCTURE IN DOT PATTERNS: AN INTEGRATED APPROACH

Mihran Tuceryan
Narendra Ahuja

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS: None
2a. SECURITY CLASSIFICATION AUTHORITY: N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): UILU-ENG-87-2206	5. MONITORING ORGANIZATION REPORT NUMBER(S): N/A
6a. NAME OF PERFORMING ORGANIZATION: Coordinated Science Lab University of Illinois — 6b. OFFICE SYMBOL (If applicable): N/A	7a. NAME OF MONITORING ORGANIZATION: Air Force Office of Scientific Research
6c. ADDRESS (City, State and ZIP Code): 1101 W. Springfield Avenue Urbana, Illinois 61801	7b. ADDRESS (City, State and ZIP Code): Bolling AFB Washington, D.C. 20332
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Air Force Office of Scientific Research — 8b. OFFICE SYMBOL (If applicable): N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR 86-0009
8c. ADDRESS (City, State and ZIP Code): Bolling AFB Washington, D.C. 20332	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
N/A	N/A	N/A	N/A

11. TITLE (Include Security Classification) "Extracting Perceptual Structure in Dot Patterns:An Integrated Approach"

12. PERSONAL AUTHOR(S) Tuceryan, Mihran Ahuja, Narendra

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Technical	FROM ______ TO ______	January 1987	140

16. SUPPLEMENTARY NOTATION

N/A

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

perceptual grouping, dot patterns, perceptual structure

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper presents a computational approach to perceptual grouping in dot patterns. Detection of perceptual organization is done in two steps. The first step, called the lowest level grouping, extracts the perceptual segments of dots that group together because of their relative locations. The grouping is accomplished by interpreting dots as belonging to interior or border of a perceptual segment, or belonging to a perceived curve, or being isolated. To perform the lowest level grouping, first the geometric structure of the dot pattern is represented in terms of certain geometric properties of the Voronoi neighborhoods of the dots. The grouping is accomplished through independent modules that possess narrow expertise for recognition of typical interior dots, border dots, curve dots and isolated dots, from the properties of the Voronoi neighborhoods. The results of the modules are allowed to influence and change each other so as to result in perceptual components that satisfy global, Gestalt criteria such as border or curve smoothness and component compactness. Thus an integration is performed of multiple constraints active at different perceptual levels and having different scopes in the dot pattern, to infer the lowest level perceptual structure. The result of the lowest level grouping phase is the partitioning of a dot pattern into different perceptual segments or tokens.

The second step further groups the lowest level tokens to identify any hierarchical structure present. The grouping among tokens is performed using a variety of constraints including their proximity, orientations, sizes, and terminations, integrated according to their apparent perceptual roles. The hierarchical grouping process repeats until no new groupings are formed. The final result of the implementation described here is extraction of perceptual structure in a dot pattern at a range of resolutions.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED [✓ X] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL: NONE

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Extracting Perceptual Structure in Dot Patterns: An Integrated Approach

Mihran Tuceryan
Narendra Ahuja

Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
1101 W. Springfield Ave.
Urbana, Illinois 61801

ABSTRACT

This paper presents a computational approach to perceptual grouping in dot patterns. Detection of perceptual organization is done in two steps. The first step, called the lowest level grouping, extracts the perceptual segments of dots that group together because of their relative locations. The grouping is accomplished by interpreting dots as belonging to interior or border of a perceptual segment, or belonging to a perceived curve, or being isolated. To perform the lowest level grouping, first the geometric structure of the dot pattern is represented in terms of certain geometric properties of the Voronoi neighborhoods of the dots. The grouping is accomplished through independent modules that possess narrow expertise for recognition of typical interior dots, border dots, curve dots and isolated dots, from the properties of the Voronoi neighborhoods. The results of the modules are allowed to influence and change each other so as to result in perceptual components that satisfy global, Gestalt criteria such as border or curve smoothness and component compactness. Thus an integration is performed of multiple constraints active at different perceptual levels and having different scopes in the dot pattern, to infer the lowest level perceptual structure. The result of the lowest level grouping phase is the partitioning of a dot pattern into different perceptual segments or tokens.

The second step further groups the lowest level tokens to identify any hierarchical structure present. The grouping among tokens is performed using a variety of constraints including their proximity, orientations, sizes, and terminations, integrated according to their apparent perceptual roles. The hierarchical grouping process repeats until no new groupings are formed. The final result of the implementation described here is extraction of perceptual structure in a dot pattern at a range of resolutions.

The support of the Air Force Office of Scientific Research under grant AFOSR 86-0009 is gratefully acknowledged.

CONTENTS

1. INTRODUCTION

The projection of the three-dimensional world onto two-dimensional images results in the loss of information such as depth. The true three-dimensional structure has to be recovered by the visual system from images that could have arisen from an infinite number of possible scenes (Figure 1). The recovery of the lost three-dimensional information cannot be done uniquely based on a geometrical theory alone, and additional assumptions are necessary. These assumptions could be very specific in some well specified domains. For example, the assumption might be that a database of all the possible objects that would ever be encountered, and their three-dimensional models, is given. The three-dimensional structure could then be deduced by using two-dimensional images as indices into the database. Although useful for applications in constrained environments, such assumptions are very restrictive and impractical for general vision.

Assumptions that do not restrict the domain of functionality must exploit properties of scenes and imaging that are general. The following questions arise in this regard. What are the general properties of scenes that might be used to perform domain independent interpretation of images? Must all properties be expressed explicitly in terms of the entities in the scene, and constraints imposed by the projective geometry? Or, could the properties of real world lead to their image plane counterparts defined strictly in terms of image plane entities with no direct reference to three-dimensional geometry? If so, what are the image entities to which these properties apply? How must the constraints be expressed if not in terms of the projective geometry? At what stage does the explicit three-dimensional nature of the scene enter the interpretation process?

To illustrate these questions, consider an image that contains two parallel lines. The parallelism could be the result of actual parallelism of two lines in three-dimensional space. Or, the parallel lines could be the projections of two parallel, planar curves viewed so that the planes containing the curves project as straight lines. This latter case is unlikely since it assumes an unstable viewpoint. In general, it appears safe to make the assumption that if two lines in the image plane are parallel, then they are also parallel in space, without requiring further data to verify that the image does not really result from an unstable viewpoint. Making this assumption eliminates the need of first obtaining a three-dimensional description of the lines, and then of testing whether they are parallel in three-dimensional space. This illustrates the use of image plane entities, and constraints on their image plane structure, to make inference about three-dimensional scene structure directly, without involving any three-dimensional structural

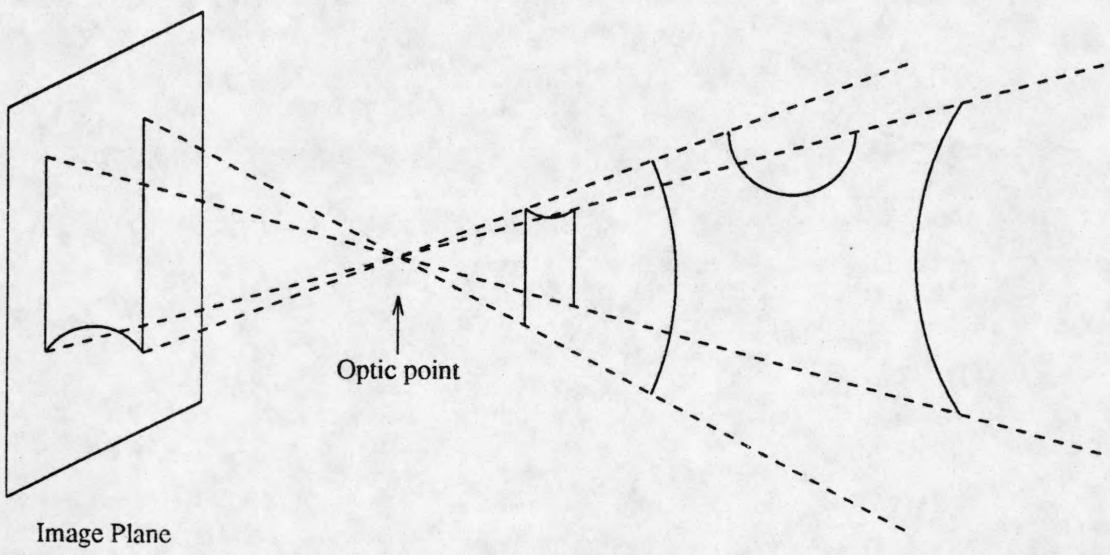


Figure 1. The image could be the projection of any of an infinite number of three-dimensional objects [19].

primitives, or using specific previous knowledge about the contents of the scene [19]. Significance of image plane structure is determined by image plane entities. Structurally related entities are said to be grouped.

Grouping thus means "putting items seen in the visual field together," or "organizing" image data. The organization may be at different scales. The rules to detect the organization may be completely stated in terms of intrinsic properties of tokens being grouped, and their image plane relationships. Since the detected image organization ultimately captures the organization of the scene, application of these rules should be a useful step towards image interpretation. Thus, grouping is a form of early inference about the structure of objects in the scene being viewed without the explicit use of three-dimensional or domain specific knowledge.

The image plane entities, or tokens, that may be grouped include blobs, edge segments, and geometrical features of image regions. This paper is concerned with grouping of the simplest of the image plane entities – dots in a dot pattern. The goal is to develop a set of rules, as well as a computational process that makes use of the rules for identifying the groupings of the dots perceived by the humans. The following section describes in more detail the phenomenon of perceptual grouping, puts in perspective the nature of the problem addressed in this paper, and explains the relevance and significance of the results.

2. PERCEPTUAL GROUPING

This section examines the process of grouping, presents a motivation for studying grouping in dot patterns, and ends with an overview of the work reported in this paper.

2.1. Gestalt Laws of Perceptual Grouping

In Figures 2 and 3, we can see striking examples of human visual processing which could be attributed to the above properties. Figure 2 shows an example of subjective contours given by Kanizsa [16] in which the humans can see a smooth contour being filled in parts of the image where it is not supported by the local intensity data. Figure 3 shows some shapes parts of which are occluded. The visual system seems to complete the partly visible figures using only certain shapes; out of an infinity of possibilities, the human visual system fills the visible parts of the boundaries with contours that smoothly extend the visible parts of the boundary at the occlusion points.

Gestalt psychologists undertook the first detailed study of the grouping phenomenon in human vision in the first part of this century [17,35]. They experimented with dot patterns such as the ones shown in Figure 4. For example, in Figure 4(a) we organize the set of dots in the manner shown in Figure 4(b). Of course, there are other equally valid organizations one of which is shown in Figure 4(c). The chosen grouping reveals our preference for one interpretation. The Gestalt psychologists at the time identified certain rules or principles to explain the particular way the human perceptual system groups tokens together. They suggested that grouping among tokens takes place based on the following criteria [35].

(1) Proximity

(2) Similarity

(3) Good continuity

(4) Closure

(5) Symmetry

(6) The factor of common fate

For any given stimulus, one or more of these rules might be at work in determining the perceived grouping. If more than one of the rules are at work, then they might be cooperating or competing; an example of each is shown in Figures 10 and 11 [35]. One question that must be answered then is how to resolve the conflicts that may arise among the results of applying these different rules. The Gestalt psychologists raised such questions. However, they were not able to propose any mechanisms that explained the working details of the grouping process, nor did they

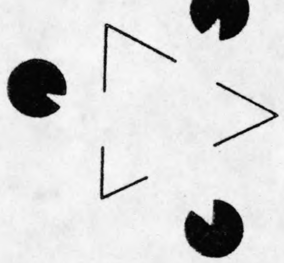
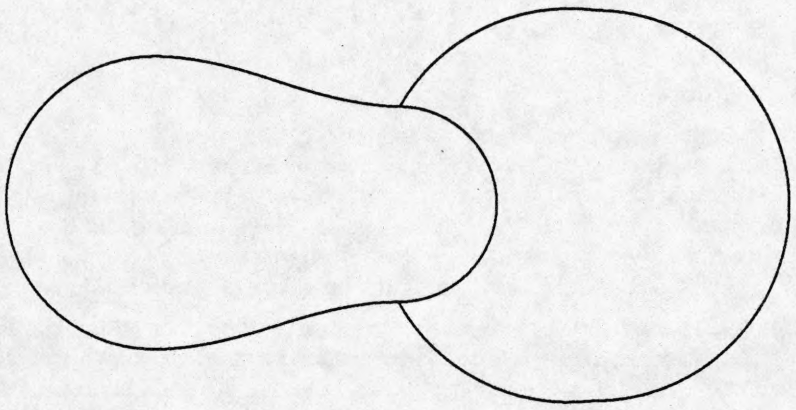


Figure 2. A subjective contour, demonstrating that the human visual system perceives structure beyond that which is explicitly present in the stimulus [20].

(a)



(b)

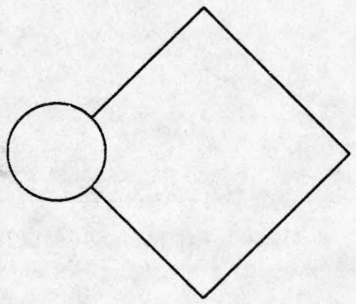


Figure 3. Occluded figures demonstrating the ability of the human visual system to fill in information which is not explicitly present in the stimulus.

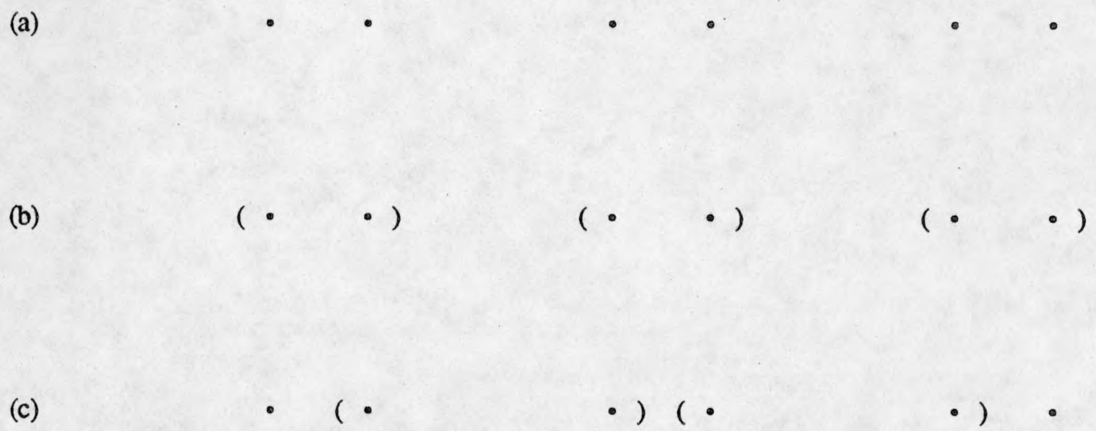


Figure 4. Humans group the dot pattern in (a), as indicated by the parentheses in (b). An example of another grouping, syntactically as feasible as (b), but unacceptable to human visual system, is shown in (c).

give any reasons for the existence of such a process to begin with.

2.2. A Computational View of Grouping

Our environment causes the images to have the structure they do. Therefore, the criteria for grouping are intimately tied to properties of the environment. Some examples of properties of the environment that may have a significant impact on the nature of image structure are discussed by Marr [20]: the visible world consists of smooth surfaces that have certain reflectances; a surface's reflectance function often is generated by processes operating at different scales; the tokens generated at a given scale by the same process tend to be similar to each other; markings on a surface which are generated by the same process often have certain coherent spatial arrangements; the loci of discontinuities in depth or surface orientation are often smooth almost everywhere. See Figure 5 for examples of these properties.

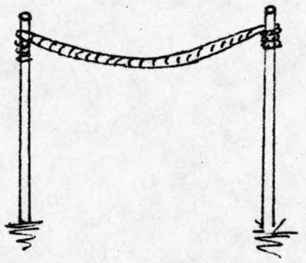
There has been an interest in computational research in precisely formulating the implications of the environmental wellbehavedness on image structure [19, 29, 37]. Following are two examples of significant relationships between the structure of images and the environment structures, which can be formulated precisely and used for computational inference of environment structure from image structure.

(1) Projection onto a plane preserves most of the smoothness properties of the environment. Therefore, it is reasonable to expect that contours in the images are smooth.

(2) Possible interpretations of an image that depend upon an accidental viewpoint are generally ignored by the human visual system and interpretations that are viewpoint-stable are preferred. Further, the larger the number of simultaneous accidental alignments required to support a given interpretation, the less likely that particular interpretation becomes. For example, in Figure 6 we see two intersecting, smooth curves instead of the two V-like objects touching at their tips. The first interpretation is in accordance with the expectation of smoothness. On the other hand, the second interpretation is unlikely because: (a) the likelihood that the V-like figures assume the given positions and orientations from among an infinite number of possibilities (so as to result in two smooth curves) is low, and (b) a small perturbation in either of the two conditions given above will alter the perceived structure resulting in an unstable percept. The first interpretation is not sensitive to the viewpoint, since there is nothing unique about the point where the two curves intersect. Small perturbations in the configuration of the curves do not change the structure drastically: if the curve a-d is moved up by a small amount, we would still interpret it as two curves intersecting, only now intersecting at a different point.

Some of the grouping criteria that the Gestalt psychologists identified result from assumptions which eliminate unlikely interpretations. Thus, assuming that contours in the physical world are smooth eliminates the interpretation that the two contours in Figure 7 are accidentally aligned. If multiple rules are involved then again we see that the interpretation preferred is the one that does not rely on unlikely events. For example, in Figure 7

(a)



(b)

Figure 5. The physical world is full of smooth curves, boundaries and surfaces. (a) A hanging rope results in a smooth curve in the image. (b) The silhouette of an animal has a fairly smooth boundary and its body is a fairly smooth surface (Figure 5(b) taken from [20]).

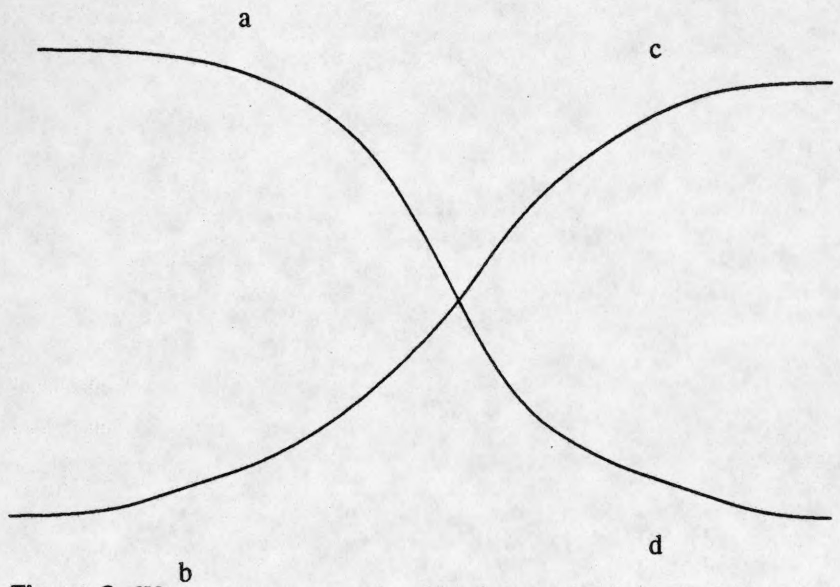
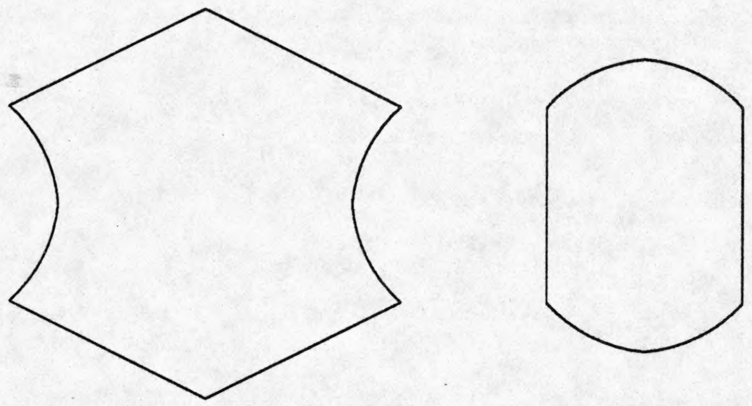


Figure 6. We see two curves a-d and b-c intersecting instead of a-b and c-d touching at a single point.

(a)



(b)

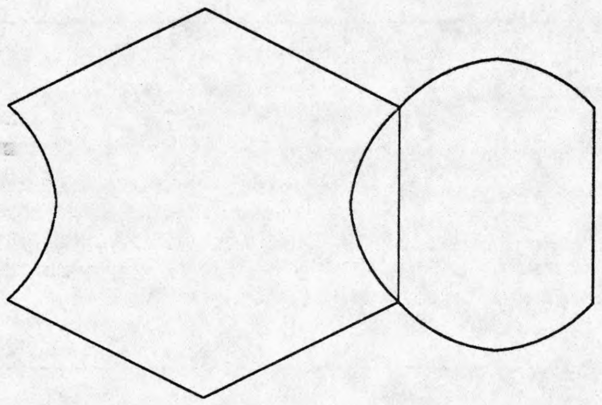


Figure 7. When symmetry and continuity are put into conflict, perception based on continuity overrides that based on symmetry.

symmetry is overriden by smoothness because the likelihood is low that the two symmetric figures in question have come together to meet at exactly the right two points, and the directions of the curves are exactly aligned on the two sides.

Following is the list of the Gestalt rules and explanations of why they work:

Proximity

When two tokens in the visual field are close together and there is no indication from their other properties (e.g. similarity), then most likely the two tokens are the result of either the same process or related processes. Hence it is reasonable to assume that they belong together.

Similarity

When two tokens are similar to each other, then the likelihood of their being generated by the same physical process is high. Therefore, in this case one makes the assumption that the two tokens are related to each other.

Closure

When a region in the visual field is surrounded by a closed border, it is highly likely that the region belongs to a single physical object since usually the closed border around an object in the scene is projected also as a closed border in the image. Sometimes, however, this closed border may be broken for reasons such as bad lighting, occlusions by another object, etc. In such situations, the resulting pieces of border segments need to be grouped together in order to recover the original border. Thus, if the tokens or line segments are aligned such that they could form a closed boundary if connected, the likelihood that they are the boundary of one physical object is greater than the likelihood that a number of independent segments have aligned just the right way. Hence, the interpretation of a single object is accepted and those tokens are grouped together.

Continuity

As discussed above, if two oriented tokens have the same orientation at a point, it is likely that they are part of the same physical object. This is also the result of the properties of the physical objects being smooth almost everywhere and the unlikelihood of two oriented tokens having precisely the given locations and orientations.

Symmetry

Two tokens that form a symmetric figure are favored for grouping together because the likelihood of two independent figures forming a symmetric figure is very low. Also, the physical world is abundant with objects whose surfaces are surfaces of revolution. The images resulting from such objects are symmetric. Therefore, if two tokens form a symmetric figure, it is very likely that they are images from a single physical object, and hence, they must be grouped together.

These rules of Gestalt psychologists do not hold all the time. Making generalized assumptions of these kinds necessarily entails cases in which the images are interpreted wrongly. An example of this was seen in Figure 7.

More recently, Lowe has discussed the roles of perceptual grouping in object recognition [19]. These are: (a) segmentation of the scene into groups of interrelated features (this means segmenting the image into objects each consisting of a number of interrelated features such as line segments, and treating each group as a unit instead of treating the features independently); (b) three-dimensional inference from certain relations among these features; and (c) using the mutual constraints on the image parts provided by these relations for reducing the combinatorics of the search space for recognition. Grouping is also an integral part of the approach to early visual processing proposed by Marr [20] in which the full primal sketch is obtained by identifying the perceptual groupings among tokens produced by the raw primal sketch. Witkin and Tenenbaum [37] view perceptual grouping as a bridge connecting the low levels of the vision to more abstract levels. The low levels are those where primitive features are detected with no previous knowledge about the contents of the scene; the more abstract levels are the ones in which

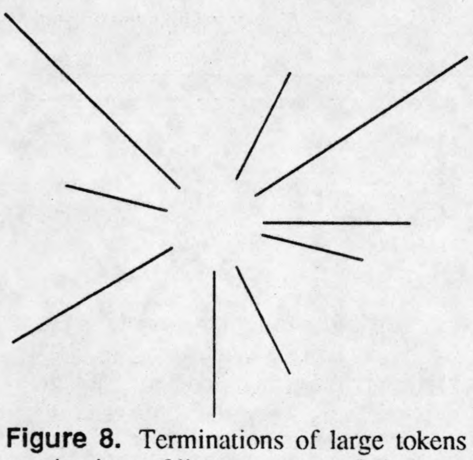


Figure 8. Terminations of large tokens appear to act as entities that can be grouped independently. Here the terminations of linear segments fall along a circle, and this grouping is perceived strongly.

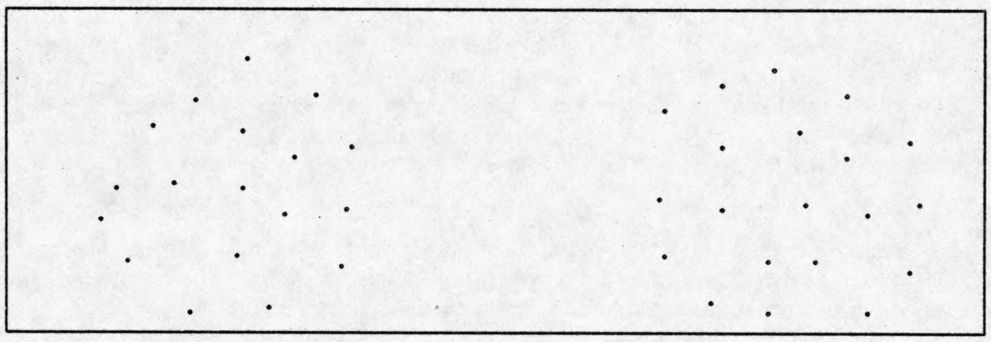


Figure 9. We see two blobs as a result of grouping of the dots which are close together.

the physical world is modeled. The features and tokens detected at the lower levels must be interrelated without the help of any semantic information. Witkin and Tenenbaum argue that the structure imposed on the stimulus in this manner corresponds most of the time to meaningful structure when semantics are introduced, and therefore, perceptual grouping is an early form of inference. They suggest that perceptual grouping is performed to identify relations among low level tokens, and to start building wholes from parts which are useful later on when interpretation is made; the importance of perceptual grouping is that it can be performed using a relatively small set of domain independent rules.

2.3. Grouping in Dot Patterns

In images, the tokens for grouping such as blobs, edges and feature points must first be extracted. These tokens have properties such as position, shape, size, orientation, color, brightness, and the termination points (if the tokens are elongated or curvilinear as in Figure 8). The roles of some of the properties in grouping may be complex, as illustrated in Figures 10 and 11.

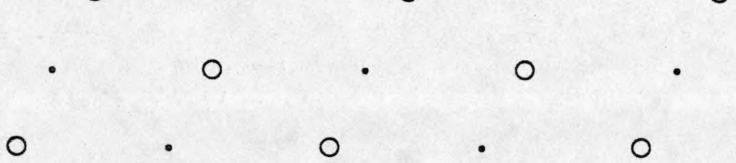


Figure 10. An example illustrating competition between the rules for proximity and similarity grouping. The proximity grouping results in black-white-black or white-black-white triples, each containing tokens that are close together. The similarity grouping results in white-white-white or black-black-black triples, each containing tokens that are similar. The resulting two groupings compete with each other and our perception alternates between the two.

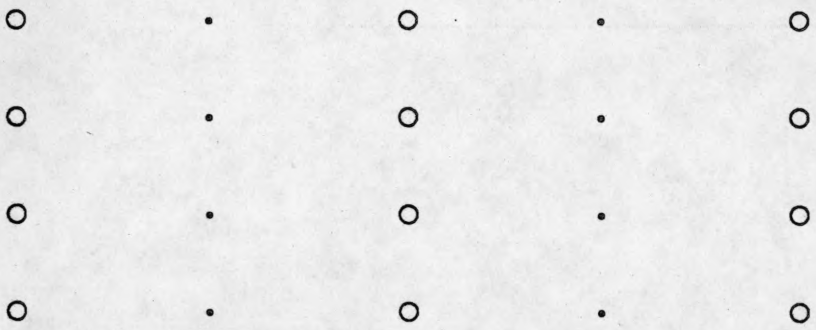


Figure 11. An example illustrating cooperation between the rules for proximity and similarity grouping. Application of each rule leads to the same columnar structure and this agreement seems to reinforce the perception of the columnar grouping.

In view of this complexity, a first step toward understanding the grouping phenomenon may be to study the roles of some relatively simple properties. One way of accomplishing this is to eliminate all but one property at a time and examine the effects of that property on grouping. Dot patterns provide a means for studying the effect of token positions on their grouping. With dots as tokens, the role of nonpositional properties is minimized since dots are without size, orientation, color, and shape.

We call the initial grouping of dots based only on their positions as the lowest level grouping (Figure 9). The simplicity of tokens holds only for this level of grouping. The perceptual segments or tokens defined by the lowest level grouping have spatial extent, and hence, properties such as orientation, shape and size. For example, pairs of dots grouped together at the lowest level form virtual lines whose effect on perception appears to be similar to actual oriented line segments [20]. Groups consisting of more than two dots can form curves and regions of various shapes. These groups act as tokens and may further group hierarchically to yield groupings at higher levels. Figure 12 shows an example of hierarchical grouping. The higher levels represent organizations at coarser scales. Therefore, by working with dot patterns and considering the hierarchy of groupings possible, we can study not only the roles of the positional properties of tokens on grouping, but also the effect of other properties. With the above notion of "level," i.e. the number of recursive grouping steps from the the initial level containing dots, the sizes of tokens at the same level but in different perceptual segments may differ significantly. Therefore, a token at a given level in one segment may further group with a token at a different level occurring in a nearby segment (Figure 12).

Associated with the hierarchical nature of the groupings is the phenomenon of the viewer's "focus of attention." When we view a pattern with multiple levels of groupings in it, we do not see all of the different levels of a given hierarchy at the same time. Nor do we perceive groupings exclusively among entities at a fixed level. We tend to concentrate our attention to one level of the hierarchy in one part of the pattern, and to a cutset of levels across grouping hierarchies defining perceptual segments across the dot pattern, such that all components of the perceived structure correspond to geometrical phenomena at the same scale of resolution. When we want to see more detail, we focus our attention to a lower level of detail by shifting the cutset of focus down, and when we want to see coarser structure, we move the cutset of focus up, to a higher level.

Sometimes dots in a region can be grouped in different ways leading to different hierarchies of groupings. The different groupings may be equally strong individually. These hierarchies may share dots at the lowest level or even groups of dots at intermediate levels. However, the structural roles fulfilled by the shared subpatterns in different, existing parallel hierarchies are different. The perceptions of the different roles played by any shared subpattern may be mutually exclusive. Thus, the dot pattern may be perceived to contain different overall groupings at different times, determined by the mixture of roles that the various shared components are perceived to play at any given time. If the mixture changes, the overall percept changes. Each percept may be relatively stable as is the case for the Necker cube phenomenon, although it may periodically shift to a different relatively stable percept. Sometimes, when all groupings are comparably strong, the perception may be of groupings constantly drifting from one state to another. An illustration of such rivalry among groupings appears in Figure 13 [21].

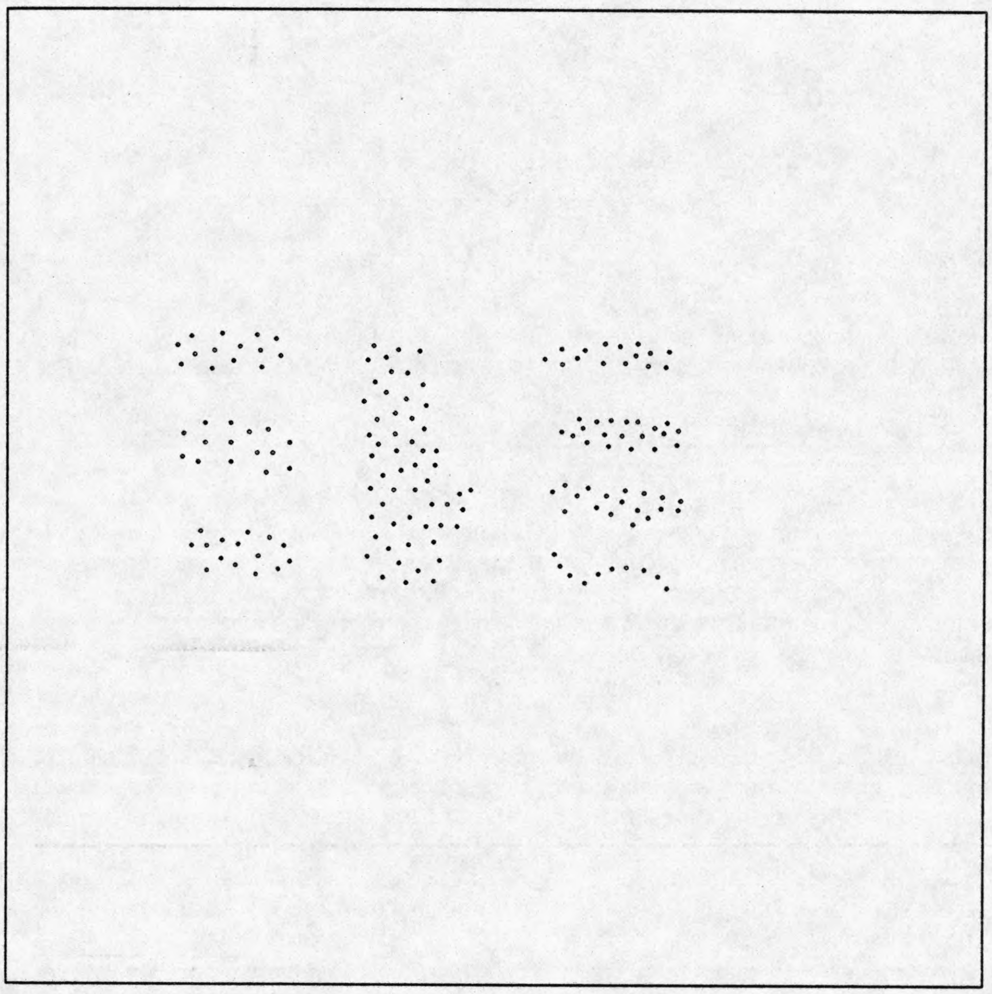


Figure 12. An example pattern showing hierarchical grouping of tokens.

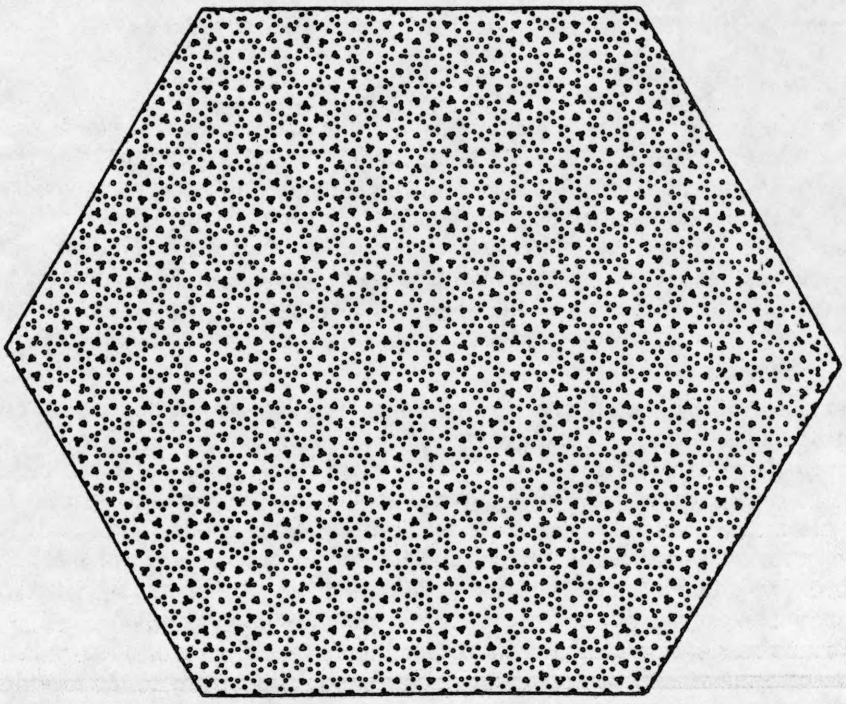


Figure 13. An example pattern with multiple, rival groupings [21]. Our perception of structure here seems to drift from that of one grouping to another.

2.4. Characterizing Perceptual Structure

To define a computational approach to perceptual grouping in dot patterns, it is necessary to specify what precisely is the output of a perceptual grouping process given the input dot pattern. Consider, first, the lowest level grouping. Since the grouping is among dots, it should be possible to define the perceptual structure also with dots as elements of description. In such a description, each dot should have associated with it a component of the perceptual structure, or a perceptual role. The goal of the lowest level perceptual grouping process, then, is to assign to each dot its perceptual role. To see what perceptual roles a dot can play in the lowest level grouping, let us closely examine the process of the grouping.

The single variable that determines the grouping of dots is the relative locations, or proximity, of dots. If the proximity is more pronounced in one or two directions, the result is a curvilinear structure of dots. If a dot has all its proximal dots occurring in a contiguous sector of its surround, and not just along one or two directions, then the dot lies along the border of a perceptual segment. This means that one side of the dot is empty relative to the other side. When a dot is completely surrounded by other dots, this indicates that the given dot lies in the interior region of a perceptual cluster. The surrounding dot density in such a case may be uniform or variable. One last possibility is the case in which a dot has no other dot in its proximity. Such an isolated dot gives rise to a single-dot cluster. Thus, the possible perceived structures are: (a) a cluster with nonempty interior, (b) a cluster with no interior (e.g., a bar), (c) curvilinear structures, and (d) a single-dot cluster. In the first case a dot may lie either along the border of a cluster or in the interior region. The goal of the computation of the lowest level groupings may be achieved by assigning to each dot one of the above roles, or, one of the following labels: *INTERIOR*, *BORDER*, *CURVE* and *ISOLATED*.

At the higher levels, the primitives of grouping become more complex in shape as do the grouping criteria. However, the perceived structures remain the same as at the lowest level, namely, the various types of clusters and curves, although they comprise of different tokens. The goal of the computation of perceptual grouping again is to partition the primitives into sets based on the various criteria such as proximity, orientation, size, termination points and curvilinearity. The results provided by different criteria may differ in which case appropriate synthesis of the perceived grouping is necessary from the groupings provided by the individual criteria.

2.5. Overview of the Work Presented

The work described in this paper aims at defining and implementing a computational model of perceptual grouping in dot patterns. Central to the definition of a computational model of grouping is the issue of representation. How should the geometric structure of a dot pattern be represented? How is the perceptual structure related to the geometric structure? How are local and global perceptual organizations related? How should inference of perceptual structure be done? How should the inferences be checked for their validity? How should the nature of disagreement between the model's predictions and the human judgement be used to improve the computational model? How is the focus of attention accommodated? How is rivalry among groupings incorporated? We provide one set of answers to some of these questions and incorporate the answers in the definition of a computational approach.

Detection of perceptual organization is done in two steps. The first step, called the lowest level grouping, extracts the perceptual segments of dots that group together because of their relative locations. The grouping is accomplished by interpreting dots as belonging to interior or border of a perceptual segment, or being along a perceived curve, or being isolated. To perform the lowest level grouping, first the geometric structure of the dot pattern is represented in terms of certain geometric properties of the Voronoi neighborhoods of the dots (Section 4). These properties are then related to the primitives of the perceptual structure, e.g. interiors of blobs, borders of blobs, curves and isolated dots. The grouping is seeded by assigning to dots their locally evident perceptual roles and iteratively modifying the initial estimates to enforce global Gestalt constraints (Section 5). This is done through independent modules that possess narrow expertise for recognition of typical interior dots, border dots, curve dots and isolated dots, from the properties of the Voronoi neighborhoods. The results of the modules are allowed to influence and change each other so as to result in perceptual components that satisfy global, Gestalt criteria such as border or curve smoothness and component compactness. Such lateral communication among the modules makes feasible a perceptual interpretation of the local structure in a manner that best meets the global expectations. Thus, an *integration* is performed of multiple constraints, active at different perceptual levels and having different scopes in the dot pattern, to infer the lowest level perceptual structure. The local interpretations as well as lateral corrections are performed through constraint propagation using a probabilistic relaxation process. The result of the lowest level grouping phase is the partitioning of a dot pattern into different perceptual segments or tokens. Unlike dots, these segments possess size and shape properties in addition to locations.

The second step further groups the lowest level tokens to identify any hierarchical structure present (Section 6). The grouping among tokens is again done based on a variety of constraints including their proximity, orientations, sizes, and terminations, *integrated* so as to mimic the perceptual roles of these criteria. The result of the grouping of lowest level tokens is even larger tokens. The hierarchical grouping process repeats until no new groupings are formed. The final result of the implementation described here is a hierarchical representation of the perceptual structure in a dot pattern. Often the representation is a tree or a forest of trees whose coarsest level represents one or more spatially disjoint perceptual components of structure; the lower levels for each component represent finer structures nested within the parent component. When there exist multiple spatially overlapping structures in the dot pattern, the representation is a graph, in particular, an interleaving of trees. Such a representation of perceptual structure can model "focus of attention" through the multiplicity of levels, and "rivalry" of groupings at a given level through the probabilistic interpretation of groupings.

Thus, the central theme of the work to be presented in this paper is integration of diverse cues and constraints to perform perceptual interpretation of a dot pattern. The cues that are integrated include local structure of dots, smoothness of borders, and compactness of components; the corresponding constraints incorporate information having increasingly global spatial scope in the dot pattern. In the implementation of the approach, we have attempted to minimize the use of ad hoc thresholds for decision making. Whenever possible, we have tried to increase the amount of information used to reduce any ambiguity in interpretation. The thresholds used are listed in Appendix A.

Many of these thresholds are adaptive. That is, they are specified as bounds on the allowed statistics of local, structural characteristics of the dot pattern, rather than as absolute numbers.

Before we go on to describe our approach, the next section briefly reviews related previous work.

3. A SUMMARY OF PREVIOUS WORK

The Gestalt psychologists in the 1920's and 1930's appear to have been the first to study grouping extensively as part of the general process of perception [17,35]. Some details of the Gestalt theory have already been presented in the previous chapter. Although Gestaltists derived a set of rules that accounted for the observed perceptual phenomena, they did not explain why and how these rules worked. They also tried to explain some of these phenomena in terms of certain neurological theories that were known at the time and drew parallels to such physical phenomena as electromagnetic fields. This level of explanation, however, is not an appropriate one because it does not deal with the reasons for the grouping at a functional level; it remains too much at the mechanism level.

3.1. Recent Work in Perceptual Organization

Recently, grouping has started to receive fresh attention. In the psychology community we see Rock [25] arguing the inferential nature of perception. He argues that most of the phenomena in perception such as organization, figure-ground perception, form perception are the result of a process of inference in which the final solution picked in human perception is the one that is not explained by accidental alignments or accidental viewpoints. We see the same ideas discussed by Witkin and Tenenbaum in a recent paper [37] They regard the process of extracting perceptual structure as one providing the initial primitives upon which the semantic interpretation is going to be based. They state that the perceptual structure detected by the human visual system captures the underlying causal relationships among the tokens in the image without the benefit of the semantic information, and the structure detected turns out to be meaningful for interpretation.

We see yet another version of the same idea in Lowe's work [19] He also argues that the significance of a perceptual structure detected by the human visual system depends on the degree of non-accidentalness of the relationship between the parts. He makes the result of the perceptual groupings directly accessible to the recognition process, which facilitates construction of 3-D models from the primal sketch through 3-D inferences. This is helpful in the interpretation of objects under degraded lighting conditions or of line drawings where surface information is not necessarily available. Marr proposes the use of grouping to obtain the full primal sketch from the raw primal sketch, using grouping criteria such as collinearity, clustering, etc. among the tokens in the raw primal sketch [20].

Zucker has examined the extraction of orientation information by the human visual system [40,41]. He suggests that there are two fundamentally different processes at work in the extraction of orientation selection: type I and type II processes. Type I processes are very sensitive to small variations in positional information, whereas type II processes are much more tolerant of positional errors. He suggests that the two types of processes are for fundamentally two different tasks. Type I processes are for detecting boundaries on surfaces which are well defined and specific. Type II processes are for obtaining information about within surface variations such as textures, and they are two-dimensional in nature.

Zucker and Hummel [39] describe an approach to the perceptual segmentation of a dot pattern by identifying the different roles that a dot can play in a segment, namely whether it lies on the border or in the interior. Such perceptual roles of dots are determined by both local and global structure of the dot pattern, that must be appropriately taken into account in the process of inferring the roles. A perceptually significant definition of local structure of a dot pattern is captured in the definition of Voronoi neighborhood of a dot described by Ahuja [1] and used as the starting point in the work described in this paper.

3.2. Work in Psychology

Researchers in experimental psychology have investigated the perception of structure in both static and moving dot patterns including detection of dotted lines in noisy background, perception of bilateral symmetry and the perception of flow patterns. Most work has been done with dot patterns to provide control of the experimental conditions. Uttal et al. studied the detection of dotted lines in a noisy background [31] and found that the detection suffered as the dot spacing along the line increased. Recently, the same phenomenon of detecting a dotted straight line and a dotted curve in a noisy background was studied by Vistnes [33] who obtained similar results. His experiments also revealed that when the dot spacing in the lines and curves was comparable to or more than the surrounding dot spacing the detection suffered. In addition he found that as the jaggedness of the line or curve segment increased it became harder to detect these structures. The detection of bilateral symmetry in random dot patterns has been studied by Jenkins [15] and Barlow and Reeves [3]. Both studies found that only a fraction of the statistical information available in the stimulus is used in the detection of symmetry. Jenkins found that the symmetry information utilized by the human visual system fell within a strip about one degree wide around the central axis of symmetry.

Barlow and Reeves have found that only about 25% of the available statistical information is used in symmetry detection and also that the orientation of the symmetry axis is not important in this task. While these experiments provide data about different aspects of the overall behavior of human visual processes, they do not attempt to explain their functional characteristics and relationships that would explain and unify the various observed data.

Glass [10] has studied the perception of Moire patterns. Moire patterns are obtained by superimposing a transformed (e.g. dilated, rotated, etc.) version of a random dot pattern onto the original pattern. Glass and Perez [11] have observed that if only a small region is seen in those patterns, then the correlation of dots disappears. Glass and Switkes [12] have found that there are limits to the amount of transformation a pattern being superimposed on itself could undergo before the perception of the Moire effect disappears. Borjesson and Hofsten [5,6] have studied moving two and three dot patterns and have identified the properties of the motion which gives rise to perception of depth.

3.3. Work in Clustering

The interest in perceptual grouping for computational vision is recent [19,20,29,37,40]. The bulk of the previous research in grouping has been in the field of clustering. We now briefly review this work.

Given a set of points, **P**, clustering is partitioning of **P** into "natural" subsets or classes that maximizes both the similarity among members of the same subset, and dissimilarity across classes [8]. These points act as tokens to be grouped and the resulting clusters contain groupings of tokens. The tokens usually have vector attributes and are viewed as points in a multidimensional feature space. The issue dealt with here is not perceptual organization directly. The success of clustering is determined by the power of the measures that define the homogeneity over a cluster. In contrast perceptual organization concerns the partitioning or clustering of dots in the original (planar) space of the dots or sometimes in a three- or four-dimensional space, and the measures for partitioning to detect perceptually significant clusters. Despite this fact, however, the work done in clustering is very relevant to certain problems in vision and is the closest body of work to the grouping processes of interest in this paper.

To define specific approaches to clustering, several issues must be addressed. First, a measure of "similarity" among the members of a single cluster must be defined in order for the clustering algorithms to work. This usually depends on the particular application and what is considered a natural partition in that particular domain. Second, since the similarity measure can only be based upon the positional information attached to the points and the relative locations of these points, the concepts of "neighbors" and "neighborhood" of a point become crucial in the definition of the similarity measure. Third, the algorithm which uses this information in order to actually perform clustering is also very important.

The concept of the "neighbors" of a dot has been defined in many ways in the past. Going from simple to complex, the different definitions include the dots that fall into a circular neighborhood [24], k-nearest neighbors [32,39], O'Callaghan's definition, which, in addition to distances of points, also incorporates relative orientations of points and information on whether a dot is hidden from another dot [23], the minimum spanning tree used by Zahn [38] in which the two dots are neighbors if they are connected by an edge in the minimum spanning tree of the set of points, the relative neighborhood graph and the Gabriel graph used by Urquhart [30] and Toussaint [28], and finally the Voronoi tessellation and its dual, Delaunay graph, recently described by Ahuja [1] and also used in this paper. While the first two definitions have been used as *ad hoc* definitions of neighbors for clustering, the remaining graph based definitions are motivated by perceptual considerations, especially the last definition in terms of the Voronoi tessellation. A sound approach to extracting global, perceptual organization must have a sound definition of local structure, namely, a perceptually significant notion of neighbors. A detailed discussion of the advantages and disadvantages of the various notions of "neighbor" mentioned above can be found in [1].

Given a definition of neighbor, the clustering algorithms perform partitioning using two criteria: a) a measure of similarity indicating if given tokens belong to a single cluster, and b) a criterion to decide when a given clustering is a good fit to the given data.

The different measures of similarity used in these algorithms may be based on the distance between dots, or they may be defined as the inner products of feature vectors associated with dots, depending on what is appropriate for that particular domain, and what constitutes a natural grouping for the given data. The different criterion functions for deciding when a particular partition is a good fit to data include sum of squared errors, minimum variance criteria, different scattering matrices (within cluster, between cluster scatter matrices) and various scalar measures computed from them. A clustering algorithm typically performs some sort of iterative optimization on the set of data using the above mentioned criteria. A review of such clustering criteria and techniques can be found in [7,8].

Other clustering techniques do not use the standard optimization procedures. Two major classes of such algorithms consist of the hierarchical clustering algorithms and graph-theoretical clustering algorithms. The hierarchical algorithms are usually implemented in one of two ways: a) agglomerative algorithms which start with the individual samples as singleton sets and combine the clusters recursively to get larger sets, which, if repeated, eventually results in one cluster; b) divisive algorithms which start with the entire sample set as one cluster, and successively divide each cluster into smaller clusters which, if repeated, eventually results in each sample point being put in a separate cluster. Of course, the recursive splitting or merging may stop at any stage when a ''stable'' clustering has been achieved.

Graph theoretical algorithms start with a certain graph structure defined on the data set, and using criteria based on the properties of that graph eliminate certain of the edges thus splitting the set of points into subsets. In this sense the graph theoretic clustering algorithms are similar to the divisive hierarchical clustering algorithms. Examples of the applications of these can be seen in [30, 38].

Perception of shapes of contours in dot patterns and perception of subparts of figures resulting from such contours has been studied by Fairfield [9]. He proposes the use of the Blum transform [4] and a fuzzy measure (in the range [0,1]) based on the angle ranges between the extreme points of a segment of the Blum transform. This measure when thresholded at different levels results in the generation of various perceptual contours for the dot pattern which are closely related to the human perception of subparts in such a figure.

4. REPRESENTING GEOMETRIC STRUCTURE

In Section 2 we saw that dot patterns comprise of shapeless tokens whose positions are of utmost importance. In reality the dots are not abstract points; they have finite sizes and shapes. However, if the shapes of all the dots in a pattern are the same and the separation between individual dots is sufficiently large compared to the dot diameters then the phenomenal effect of each dot is that of a point with no shape. The resulting percepts from such stimuli, therefore, are based effectively on the relative positions of the dots.

As we mentioned in Section 2, the dots interact with each other locally, i.e., direct relationships among the relative positions is important only for nearby dots. This interaction and perceived structure change when a sufficient number of intervening dots is introduced (Figure 14). Therefore, it appears to be sufficient to capture only the local geometric environment, or the *geometric structure*, of a dot in a basic representation. In our approach, this geometric structure then serves as input to a procedure that infers perceived structural components such as blobs, borders, and curves, or the *perceptual structure* of a dot pattern. The geometric structure represents an abstraction of the dot pattern that captures all details of the dot pattern relevant to the inference of perceptual characteristics. It helps divide the problem of extracting perceptual description into two relatively loosely coupled components. The perceptual component has access to the information in the dot pattern mainly through the geometric structure. It is thus crucial to have a sound notion and representation of the geometric structure – a representation that captures all raw information of perceptual relevance.

In this paper, we use the Voronoi tessellation of a dot pattern to associate with each dot a planar region, or *neighborhood*, that represents the dot's geometric environment. The geometric properties of the Voronoi neighborhoods represent the geometric structure [1]. As one part of the geometric environment, the Voronoi neighborhoods also specify the *neighbors* of a dot. Before we discuss the representation of geometric structure further, we first review the definition of the Voronoi tessellation of a dot pattern.

4.1. Voronoi Tessellation

Suppose that we are given a set S of three or more points in the Euclidean plane. Assume that these points are not all collinear, and that no four points are cocircular. Consider an arbitrary pair of points P and Q. The bisector of the line joining P and Q is the locus of points equidistant from both P and Q and divides the plane into two halves. The half plane $H_Q^P(H_P^Q)$ is the locus of points closer to P (Q) than to Q (P). For any given point P a set of such half planes is obtained for various choices of Q. The intersection $\bigcap_{Q \in S, Q \neq P} H_Q^P$ defines a polygonal region consisting of points closer to P than any other point. Such a region is called the Voronoi [34] polygon associated with the point. The set of complete polygons is called the *Voronoi diagram* of S [27]. The Voronoi diagram together with the incomplete polygons in the convex hull define a *Voronoi tessellation* of the entire plane. The Voronoi tessellation for the example pattern in Figure 15 is shown in Figure 16. Two points are said to be *Voronoi neighbors* if the Voronoi polygons enclosing them share a common edge. The dual representation of the Voronoi tessellation is the *Delaunay graph* which is obtained by connecting all the pairs of points which are Voronoi neighbors as defined above.

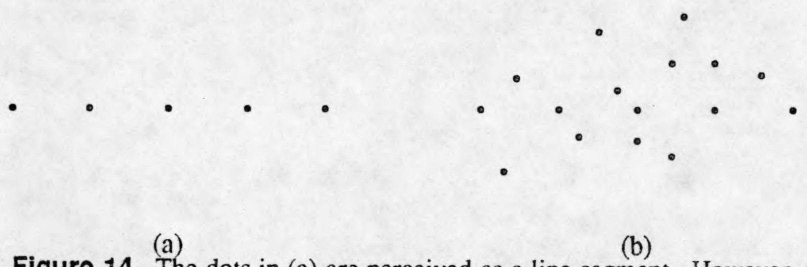


Figure 14. The dots in (a) are perceived as a line segment. However, when included among many other dots that locally interact with the collinear dots, the perception of collinearity is lost (b) [19].

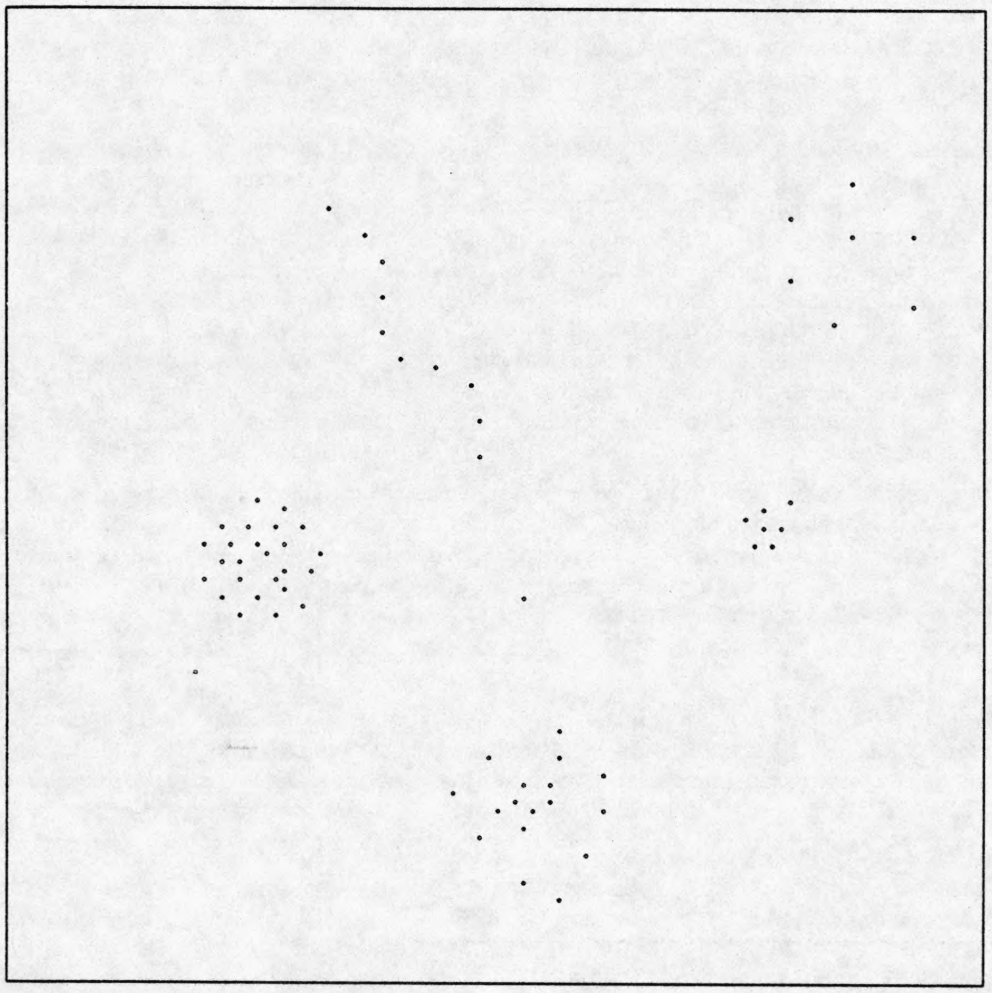


Figure 15. An example dot pattern [30].

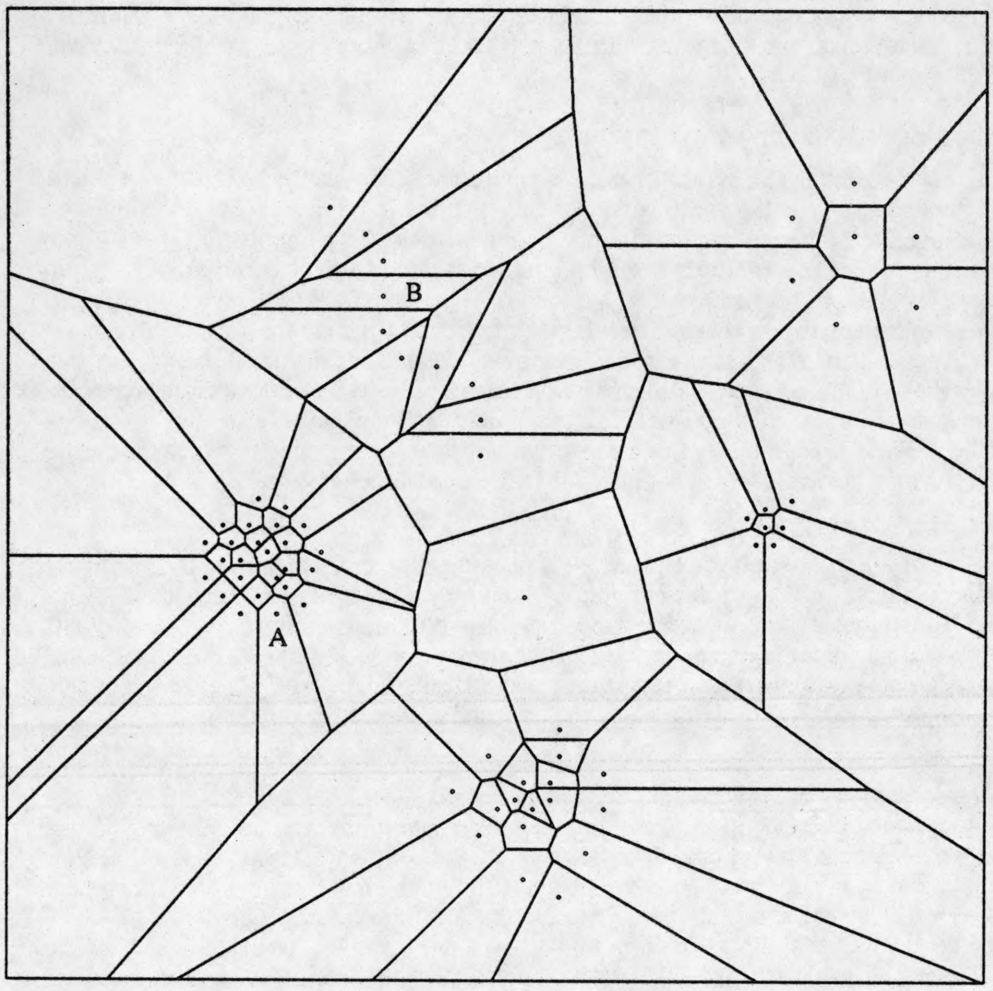


Figure 16. The Voronoi tessellation of the dot pattern shown in Figure 15.

4.2. Neighborhood of a Dot

The notion of the "neighborhood" of a dot allows access to the properties of a two-dimensional region around the dot in question such as area and shape properties. In contrast, when only neighbors of a dot are given, only properties such as distances between dots may be used which capture a hybrid of one- and two-dimensional information. The Voronoi tessellation and the polygonal region assigned to each dot in this representation lead to a natural and intuitive definition of neighborhood of a dot. Because the Voronoi tessellation is adaptive to variations in dot distribution, such properties of the Voronoi polygons as the area, shape, etc. also vary with changes of scale, density variations, etc., thus reflecting the differences in spatial distributions of dots.

We consider as the neighborhood of a point P (the region enclosed by) the Voronoi polygon containing P. Considering the way a Voronoi polygon is constructed, this is an intuitively appealing approach. The local environment of a point in a given pattern is reflected in the geometrical characteristics of its Voronoi polygon. This presents a convenient way to compare the local environments of different points. Since the perceived structure in a dot pattern results from the relative spatial arrangement of points, the geometric properties of Voronoi polygons may be useful for describing and detecting structure in dot patterns. Such an approach lends a fully two-dimensional character to the problem in that the dot pattern is converted into a planar image or a mosaic.

The advantages of the Voronoi neighborhood compared to the various definitions reviewed earlier are that the Voronoi neighborhood is (i) intuitive, (ii) adaptive, and (iii) two-dimensional in character. It is adaptive in the sense that a) the assignment of neighbors does not depend on the scale of the dot pattern and neighbors are assigned to dots that reflect the local density variations, and b) the number of neighbors of a dot is not fixed and may vary depending on the structure in the vicinity of the dot.

4.3. From Voronoi Neighborhoods to Geometric Structure

Many of the perceptually significant characteristics of a dot's environment are manifested in the geometric properties of the Voronoi neighborhoods. It is the specification of such geometric properties that constitutes our representation of the geometric structure of the dot pattern. In this section we describe the properties that we have used for this purpose. The selection of the properties was based on intuition. Along with the description of the properties we also give below their perceptual significance, which led to their selection. The knowledge of the perceptual significance will later help us to formulate procedures that infer the perceptual structure from the measurements of these properties in a given dot pattern. The terms *polygon* and *neighborhood*, the latter being the two-dimensional area surrounded by the Voronoi polygon, will be used interchangeably. All of the measures described except the areas of the polygons are within the range 0 to 1. The eccentricity and elongation also have directional information attached to them. In the case of eccentricity this information indicates the direction in which the density increases. The elongation carries with it the angle information for the major axis of the polygon.

Area

The first property is the area of a Voronoi polygon of a dot. This is a measure that depends on the magnitude of the dot density in the local vicinity of a dot. In the interiors of homogeneous clusters, the density does not change. Recalling the way the Voronoi tessellation is constructed, we observe that this uniformity of density will result in the Voronoi neighborhoods having equal areas in the interior regions of such clusters. In clusters with varying density, the areas of the Voronoi neighborhoods will change systematically reflecting the change in dot density. The areas of the Voronoi neighborhoods will decrease along the direction of increasing dot density.

Eccentricity

The second property is the eccentricity of the Voronoi polygons. Consider a dot pattern whose density varies in a given direction. If the density increases in the given direction, then the extent of the Voronoi polygons in that direction will tend to decrease by an amount proportional to the density change. Further, the positions of the dots will also be off the center of gravity of their respective Voronoi polygons, shifted in the direction of increasing density. The eccentricity measure is a scaled vector indicating how much a dot is off the center of gravity of its Voronoi polygon. The situation and the computation of the eccentricity is shown in Figure 17. The significance of the eccentricity measure is that it is related to the change of density of dots. The interiors of uniform clusters are expected to have cells with very low eccentricities because of the lack of variation in the density. The interiors of varying density clusters will have cells with high eccentricities due to the density variation; the directions of the eccentricities will be pointing towards the increasing density direction. Thus, the eccentricity vectors of the cells in the interiors of varying density clusters will be aligned most of the time. At the borders of clusters, the eccentricity

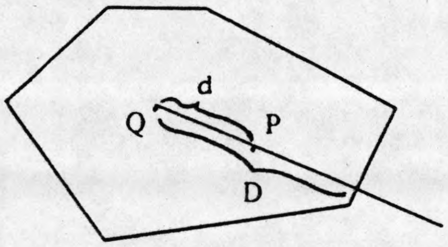


Figure 17. Eccentricity of a cell belonging to point P is defined as d/D. Eccentricity direction is in the direction of QP. Here Q is the centroid of the cell.

directions in most cases will be expected to point towards the interiors of the clusters because of the sudden increase in the dot density; i.e., from the very low density in the intercluster space to the comparatively high density in the interior of the cluster. This observation also holds on the borders of bars which are clusters without interior points.

Gabriel Measure

The third property is Gabriel measure, whose computation is shown in Figure 18. It measures the "neighborliness" of two Voronoi neighbors. If the line joining two Voronoi neighbors i and j intersects the edge shared by the corresponding Voronoi polygons then i and j are perfect neighbors. That is, there is no third point k, such that the point i is hidden from the point j by the point k, and vice versa. If, on the other hand, there is such a third point k, and the line (i,j) crosses the Voronoi cell for point k, then i is hidden from j by k and the Gabriel measure indicates the amount by which this is true. The deeper the line (i,j) penetrates cell k, the worse neighbors (i,j) are and the lower the Gabriel measure is. This is important on the borders of clusters, where if the two points are not perfect neighbors and have a low Gabriel measure, then the border follows through the intervening point instead of the two points. For example, in Figure 18 if Gabriel measure is sufficiently low, the border passes through points (i,j,k) instead of going through (i,j) directly.

Isotropicity

The fourth property is the isotropicity of the Voronoi cells. In the interiors of clusters, even though the density of points may vary, the points are surrounded uniformly by other points all around. This results in the equal angles subtended on a dot by its neighboring dots which are themselves neighbors (Figure 19). Cells within such clusters are "compact" or "isotropic." Now consider the cells in the region where borders of two clusters approach each other. The outside of a cluster border has a nonuniform distribution of dots. The interior side of the cluster border, however, has a uniform distribution of dots. This results in the dot distribution around a point lying on such a border segment to be uneven which, in turn, is reflected in the shape of the Voronoi polygon of the dot. An example of such a case is the point A and its Voronoi neighborhood in Figure 16. Such a polygon has a wedge-like

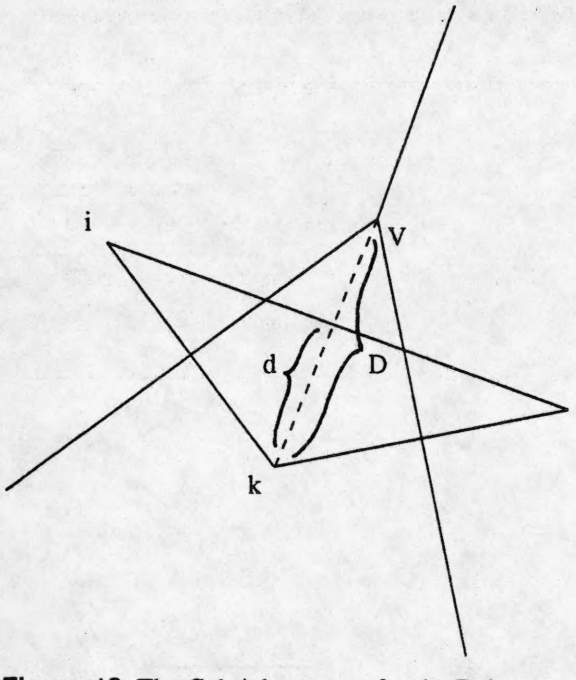


Figure 18. The Gabriel measure for the Delaunay edge (i,j): $gabr_{ij}=d/D$ if the line (i,j) intersects the line (k,V), and $gabr_{ij}=1$ otherwise.

shape, and is "non-compact" or "non-isotropic." The computation of the isotropicity measure is shown in Figure 19.

Elongation

Another property of the Voronoi cells is their elongation and the direction of their major axis. This measure is important in clusters which have spatially homogeneous distribution of dots but the density of dots in different directions across the pattern is different. In such cases, the Voronoi polygons tend to be more squeezed along the direction in which density is higher than along the lower density direction, thus, resulting in more elongated cells. The major axis and minor axis directions of such cells indicate the directions along which the dot densities are the smallest and the greatest, respectively. The cells along a curve will also tend to be elongated (as illustrated by point *B* in Figure 16) due to the fact that on the two sides of the curve the dot density is very low compared to the dot density along the curve. Thus, the presence of such elongated cells is one of the indications of the existence of curvilinear structures.

There are many ways the elongation of a polygon can be computed. One possibility is the ratio of the area of the polygon to its perimeter squared. Another, which we have used, is based on the moments of area. The elongation measure computed in this way is rotation invariant.

So far all of the properties we have discussed are two-dimensional properties of the Voronoi polygons. Most of these properties make sense when used for clusters that have interiors. For curves or single-point clusters the two-dimensional properties are no longer well behaved. For example, the area differences are not meaningful when used in regions with curvilinear structures, or on borders of clusters. In such regions one dimensional measures such as distances, etc., are more useful. In the following paragraphs, we will describe several such properties.

Distance Measure

The first such property involves differences in the length of a Delaunay edge, and the average distance of each of its endpoints to its Voronoi neighbors. The computation of this property is shown in Figure 20. The Delaunay edges that pass through intercluster space have endpoints on two different clusters each of which may be a dense cluster, a single-point cluster, or a curve. If one of the endpoints is on the border of a dense cluster then the distances on the interior of that cluster will have relatively small values compared to the length of the Delaunay edge under consideration. Thus at least one side of a Delaunay edge having an average distance which is small compared

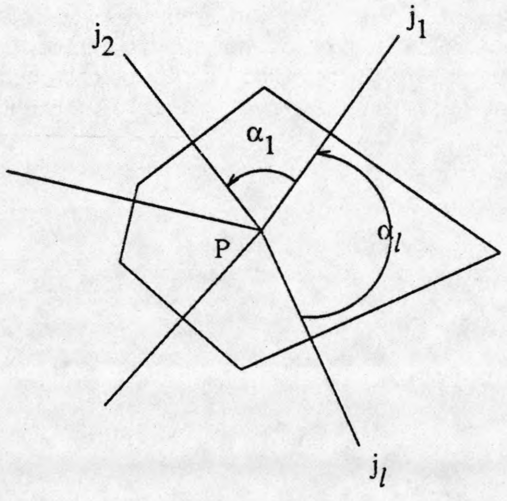


Figure 19. Isotropicity measure is computed in terms of the angles α_k. Let $\alpha_{av} = \frac{(\sum_{k=1}^{l} \alpha_k) - \alpha_{\max}}{l-1}$. Then the isotropicity of the cell for point i, $w_i = \frac{\alpha_{\max} - \alpha_{av}}{\pi}$.

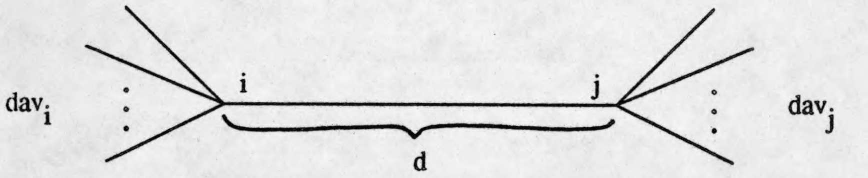


Figure 20. The distance measure for the Delaunay edge (i,j), $dist_{ij}$, is defined in terms of the length of the Delaunay edge (i,j), d, and the average Delaunay edge lengths on its two endpoints, dav_i and dav_j. Let $D = min(dav_i, dav_j)$. Then $dist_{ij} = 1 - D/d$ if $d > D$, and $dist_{ij} = 0$ otherwise.

to the length of that Delaunay edge is an indication that the edge is likely to be in the intercluster space. In this case the distance measure will have a high value indicating this high likelihood.

Squeezedness

This measure, is very closely related to elongation, but it is a property of the Delaunay edges rather than Voronoi polygons, and it is based upon relative lengths of the surrounding Delaunay edges. Figure 21 shows the computation. This measure is also mostly useful for curvilinear clusters. Delaunay edges that lie on a curvilinear cluster are likely to be shorter compared to the Delaunay edges extending laterally on the two sides of the curve. Thus, if the squeezedness measure of a Delaunay edge is high, it indicates that the Delaunay edge is likely to be on a curve.

Some of the measures presented above may seem to be overlapping and redundant. In fact, some of them are very closely related, for example, elongation and squeezedness. However, there are cases in which one measure gives a different response than another very closely related measure. Some such examples illustrating the differences between the different measures are given in Figure 22.

4.4. Computation of Geometric Properties from Moments

The various moments of area of the Voronoi neighborhoods serve as a useful tool to compute several of the geometric properties just described. We actually did compute several of these properties in our experiments from moments. These measures are the area, centroid, elongation, and the principle axes of the Voronoi neighborhoods. Below, we will briefly review the definition of the moments of area [14] of a figure and then describe computation of these measures that we computed from moments.

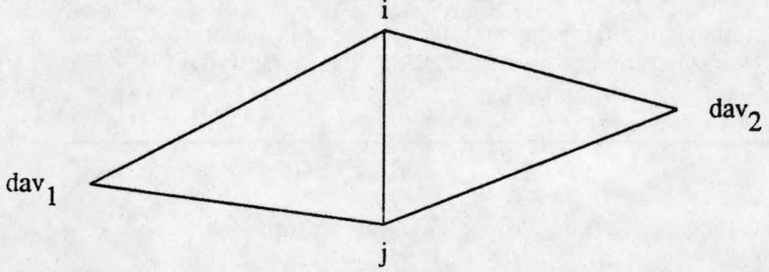


Figure 21. Squeezedness measure, sq_{ij}, is defined in terms of the length of the Delaunay edge (i,j), d, and the average Delaunay edge lengths on its two sides, dav_1 and dav_2. Let $D = min(dav_1, dav_2)$. Then $sq_{ij} = 1 - d/D$ if $d < D$, and $sq_{ij} = 0$ otherwise.

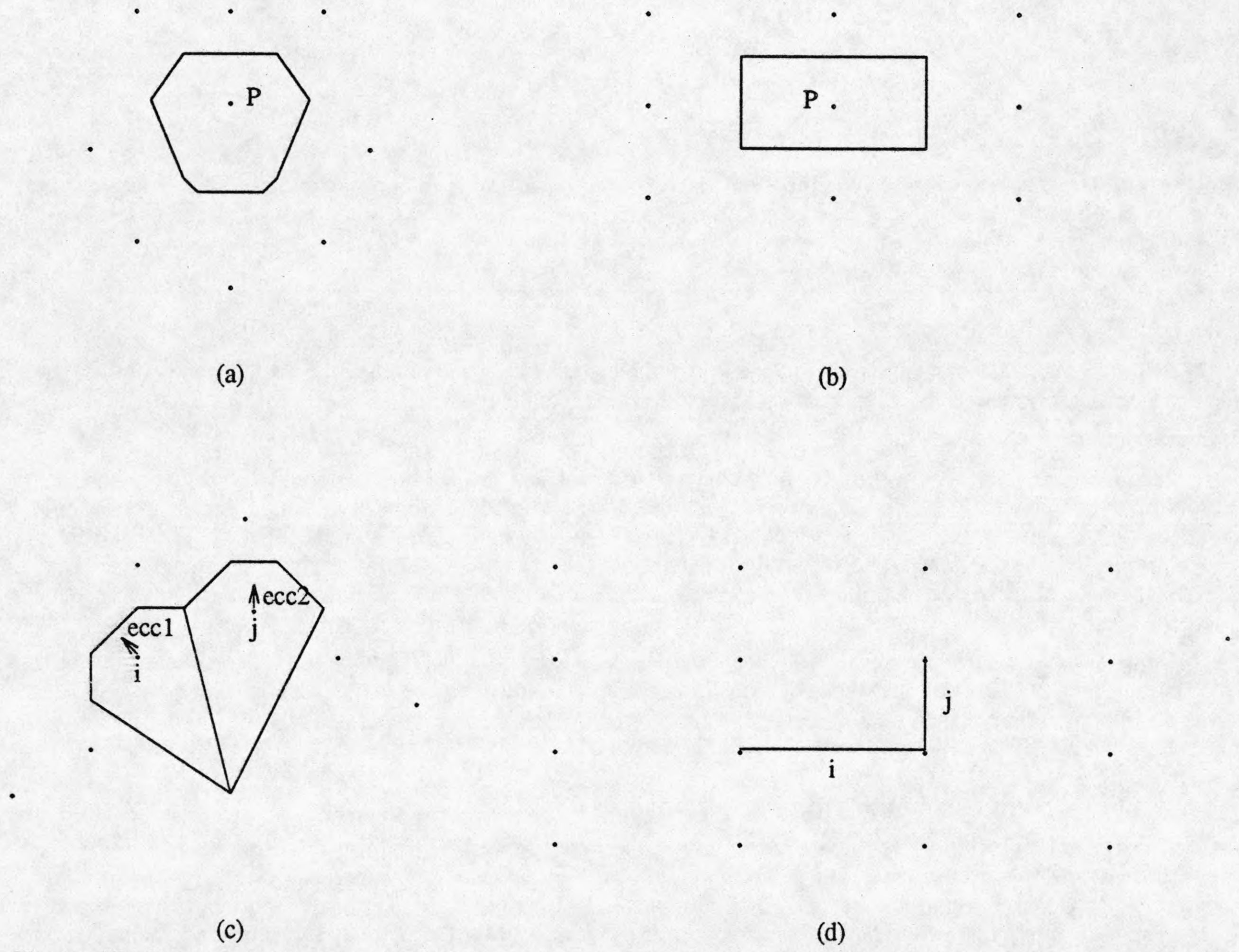


Figure 22. Examples illustrating the differences among various geometric measures. (a) The cell belonging to dot P is eccentric but isotropic. (b) The cell belonging to P is elongated but isotropic and noneccentric. (c) The two eccentricities ecc1 and ecc2 point in the same direction even though the two dots i and j are on the borders of different clusters. The fact that the two dots belong to two different clusters is reflected in the nonisotropicity of the cells. (d) Edge j connects dots that group together along a curve, but not edge i. The distances from endpoints to neighbors used to compute the distance measure do not capture this perceptual difference. On the other hand, the distance to lateral neighbors used to compute the squeezedness measure distinguishes between the edges appropriately.

The $p+q$th order moments of the area of a closed region R are defined by the following formulae:

$$m_{pq} = \iint_R x^p y^q \, dA$$

where $p+q = 0,1,2,\ldots$. The set of moments computed in this way depend on the scale and location of the figure. To obtain scale and position invariance we normalize the above computed moments with respect to scale and location to obtain the corresponding invariant moments, μ_{pq}. The location invariant moments (central moments) can be written as

$$m'_{pq} = \iint_R (x-\overline{x})^p (y-\overline{y})^q \, dA$$

in which $(\overline{x},\overline{y})$ is the centroid of the region R. Using these central moments we compute the scale invariant moments, μ_{pq}, by:

$$\mu_{pq} = \frac{m'_{pq}}{m_{00}{}^{\gamma}} \qquad \gamma = \frac{p+q}{2}+1$$

The details of the efficient computation of these moments for polygonal regions can be found in [36].

Computation of this set of moments immediately gives us the means to compute the measures mentioned above. These measures are computed as follows:

$$area = m_{00}$$

$$(\overline{x},\overline{y}) = (m_{10}/m_{00}, m_{01}/m_{00})$$

Finally, the elongation of a polygon and its major axis direction, α, can be computed using the following formulae:

$$elong = \left[\frac{\left[(\mu_{20}-\mu_{02})^2 + 4\mu_{11}{}^2 \right]^{\frac{1}{2}}}{\left[(\mu_{20}-\mu_{02})^2 + 4\mu_{11}{}^2 \right]^{\frac{1}{2}} + \mu_{20} + \mu_{02}} \right]^{\frac{1}{2}}$$

$$\alpha = \frac{1}{2} \tan^{-1} \left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}} \right)$$

5. LOWEST LEVEL GROUPING

Perception of structure in dot patterns amounts to an assignment of structural roles to dots. In Section 2 we argued that the different roles that a dot could play are *INTERIOR*, *BORDER*, *CURVE* and *ISOLATED*. Thus the goal of a computational approach to perceptual grouping may be accomplished by assigning one of these roles to each dot.

5.1. The Need for Integration

What determines the perceptual role of a dot? In Section 4 we showed that the different perceptual roles of a dot cooccur with characteristic local geometric structures. However, this association is only *typical*. The presence of a certain local geometric structure is neither necessary nor sufficient for a certain perceptual interpretation of *a given* dot – the local geometric structures of other dots, near and far, and their perceptual interpretations have a profound and usually domineering influence on how the given dot is perceived. "Faults" in the expected local geometric structures of a limited number of dots are tolerated in favor of Gestalt properties such as smoothness of borders or curves, or compactness of components, thus resulting in global interpretations that may assign such perceptual roles to dots which conflict with their local geometric structures. See Figure 23 for an example.

This brings out the multiplicity and diversity of constraints governing the final perceptual role of a dot and the need for an integrated treatment of these constraints: While the interpretation process for a dot must be seeded by its local geometric structure in a bottom-up manner, no firm interpretation may be made until similar tentative bottom-up interpretations have been carried out elsewhere, and it is determined which interpretations are coherent, i.e., they satisfy expectations about global structure.

5.2. An Integrated Approach to Lowest Level Grouping

We now describe an approach to the lowest level grouping that makes integrated use of multiple constraints. The lowest level grouping phase consists of three steps. The modules and control structure of this phase are shown in Figure 24. The first step (box A in Figure 24) consists of three independent modules (boxes II, BI, and CI) running in parallel. Each of these modules responds to a certain aspect of the stimulus and thus serves as a detector of a structural primitive. The first one (II) identifies interior dots, the second one (BI) identifies border dots, and the third one (CI) identifies curves. Each of these modules possesses fairly limited expertise about how to recognize only its own perceptual primitive. The expertise is, of course, in terms of the expected geometric structure which is expressed in terms of the local geometric properties given in the previous chapter. Because of the narrowness and spatial locality of this expertise, each module makes errors in detection. However, where one module may not find enough evidence for a decision, another may be very confident in its decision. Thus, modules may complement each other in the strengths and weaknesses of their performance. The second step (B) compares the results of the modules II and BI to correct possible errors that might exist in their results individually, thus combining the strengths of the two modules. The correction is done by perturbing the output of each module such that smoothness of the borders as well as agreement among the modules are maximized. This idea of mutual cooperation and strengthening is carried further into the third step where the already detected borders are used to identify potential components, or regions, of dots. Interpretations of dots and edges are modified so that they result in compact components having smooth borders. Thus the third step (C) combines the results of the border correction (BC) and interior correction modules (IC) based upon more global criteria. To illustrate, the outputs of all these modules obtained for the sample pattern in Figure 15 are shown in Figures 25-31. We have used the relatively sparse pattern of Figure 15 to illustrate the role of each module, since it contains several different perceptual components including curve, an isolated dot, and dense clusters. The final results of the lowest level grouping are given later for other relatively more complex patterns any single one of which may not contain all of the above components.

A powerful feature of this approach is the use of distributed expertise among modules. The limited expertise of the modules makes them computationally efficient, even though they lack in correctness of results. The latter problem is alleviated by strong lateral communication among modules. The lateral communication enforces the Gestalt expectations on the interpretations of dots provided by the modules, to synthesize the correct, combined interpretation. Modules of successive stages use Gestalt properties of higher complexity, e.g., starting with curve properties at the first stage, and moving to component properties at the next.

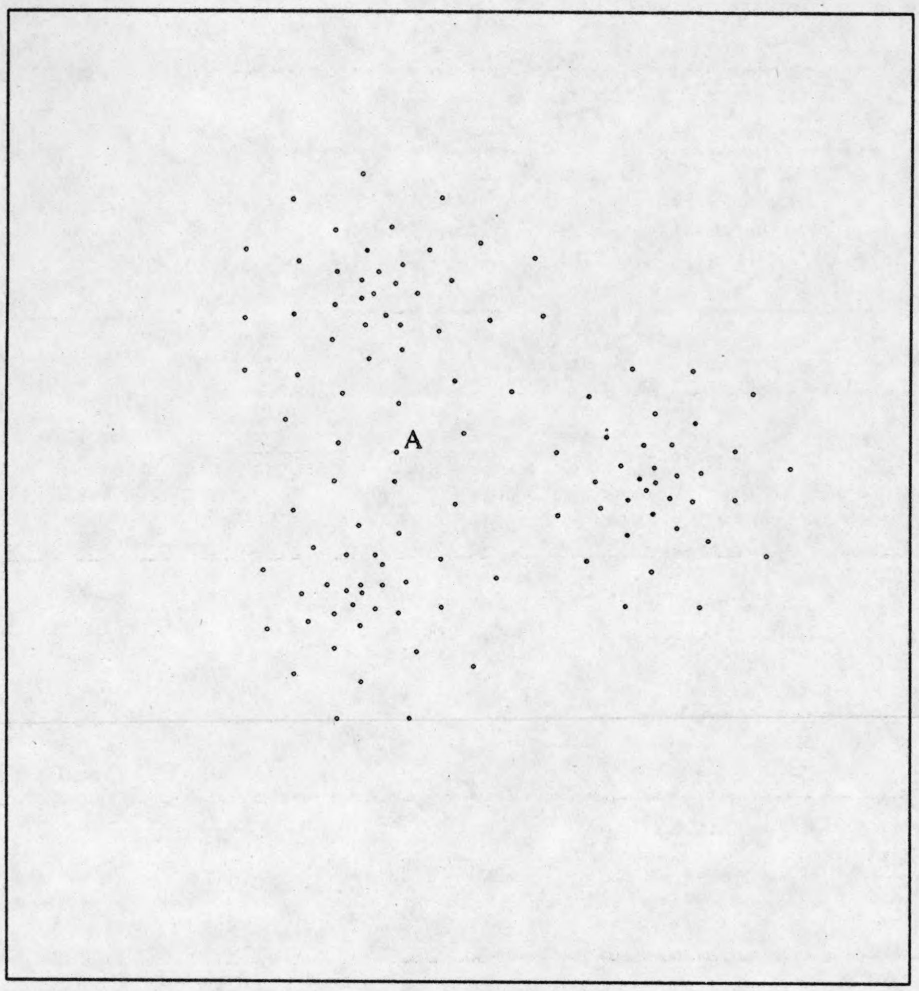


Figure 23. The dot labeled A has local geometric structure characteristic of interior and yet it is perceived as a border dot because this alternate role results in a smoother border.

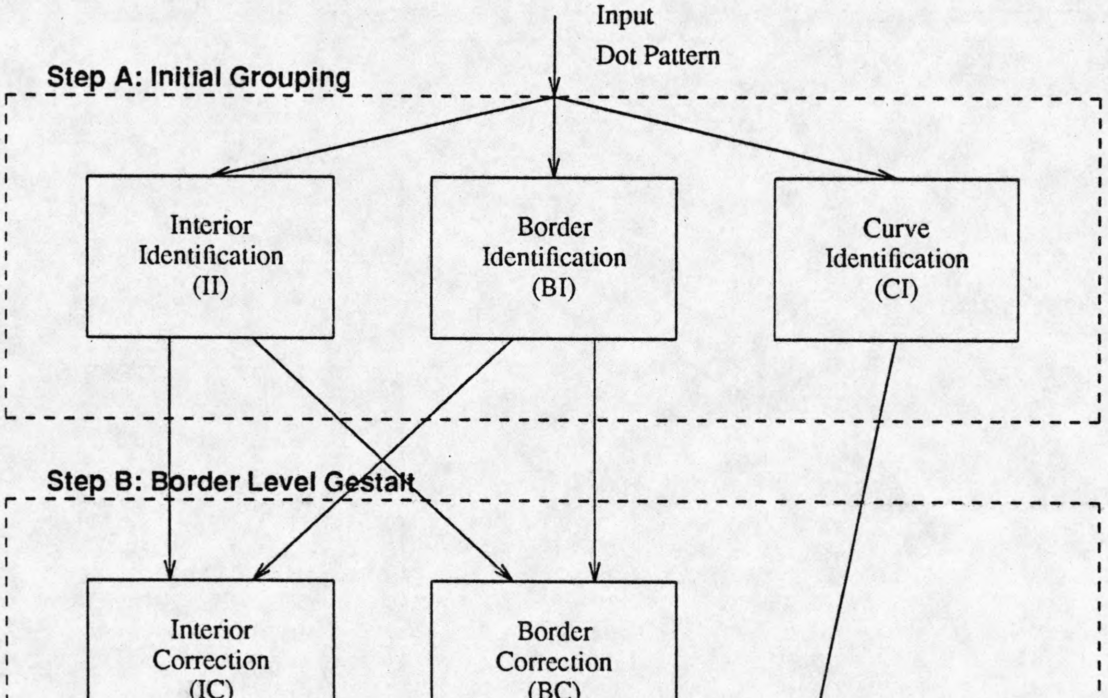

Input
Dot Pattern
Step A: Initial Grouping
Interior Identification (II)
Border Identification (BI)
Curve Identification (CI)
Step B: Border Level Gestalt
Interior Correction (IC)
Border Correction (BC)
Step C: Component Level Gestalt
Interior Border Combination (IBC)
Curve Correction (CC)
Lowest Level
Perceptual Grouping

Figure 24. The various modules and control flow for the lowest level grouping.

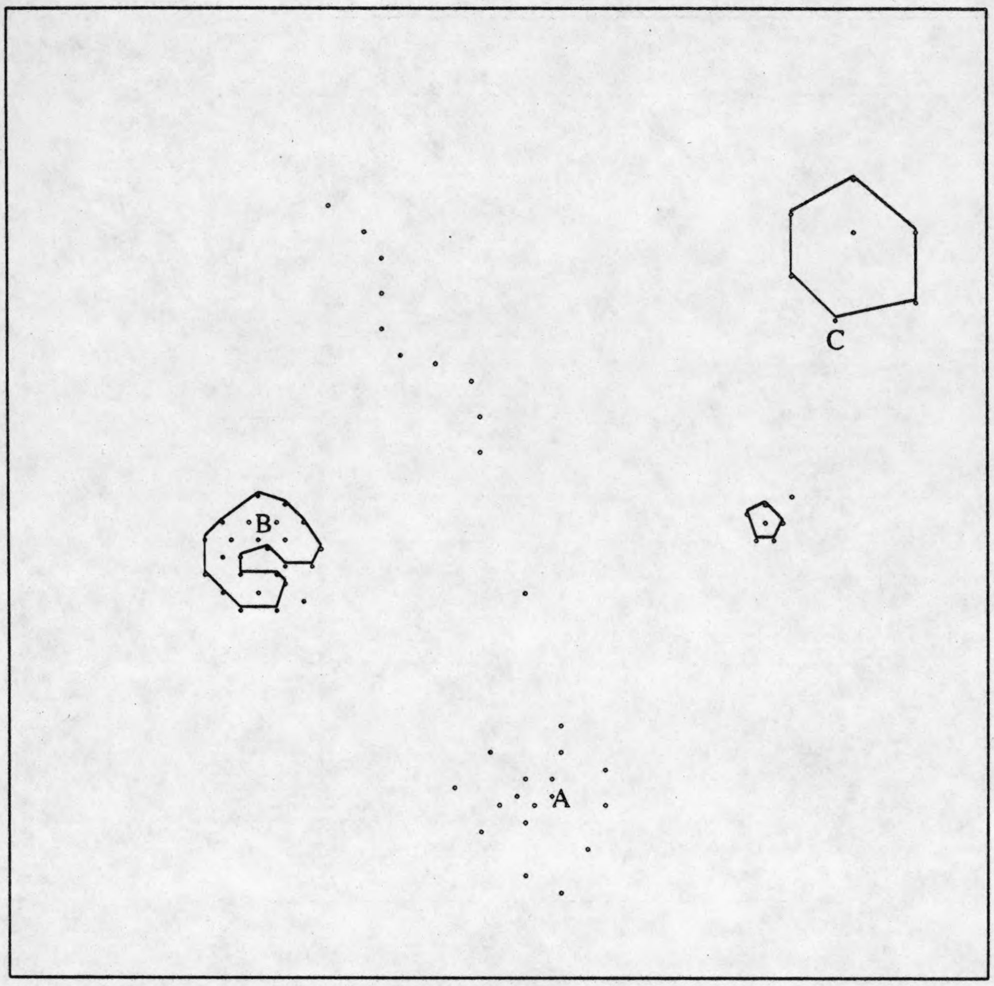


Figure 25. The output of the interior identification (II) module for the pattern in Figure 15. The dots that lie inside the closed borders are labeled as *INTERIOR*. All the dots that lie either on the borders or outside the borders are labeled *NONINTERIOR*.

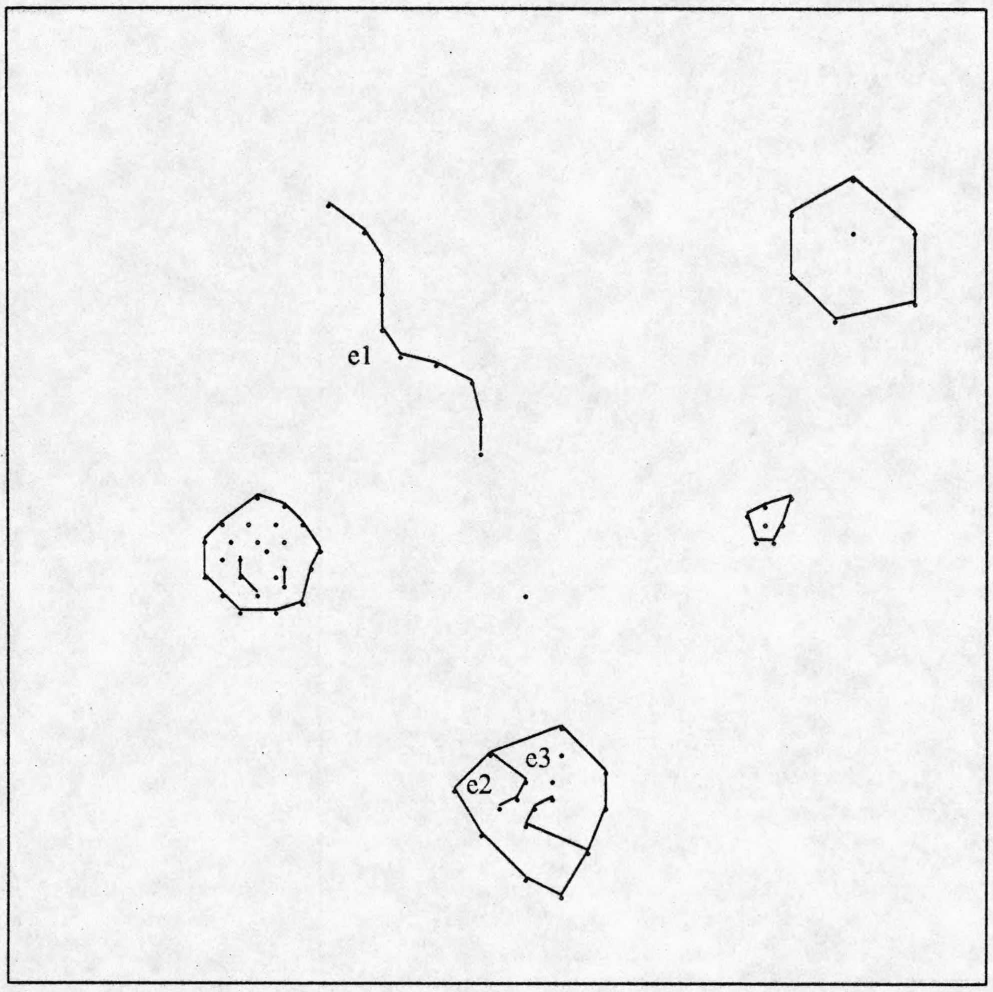


Figure 26. The output of the border identification (BI) module for the pattern in Figure 15.

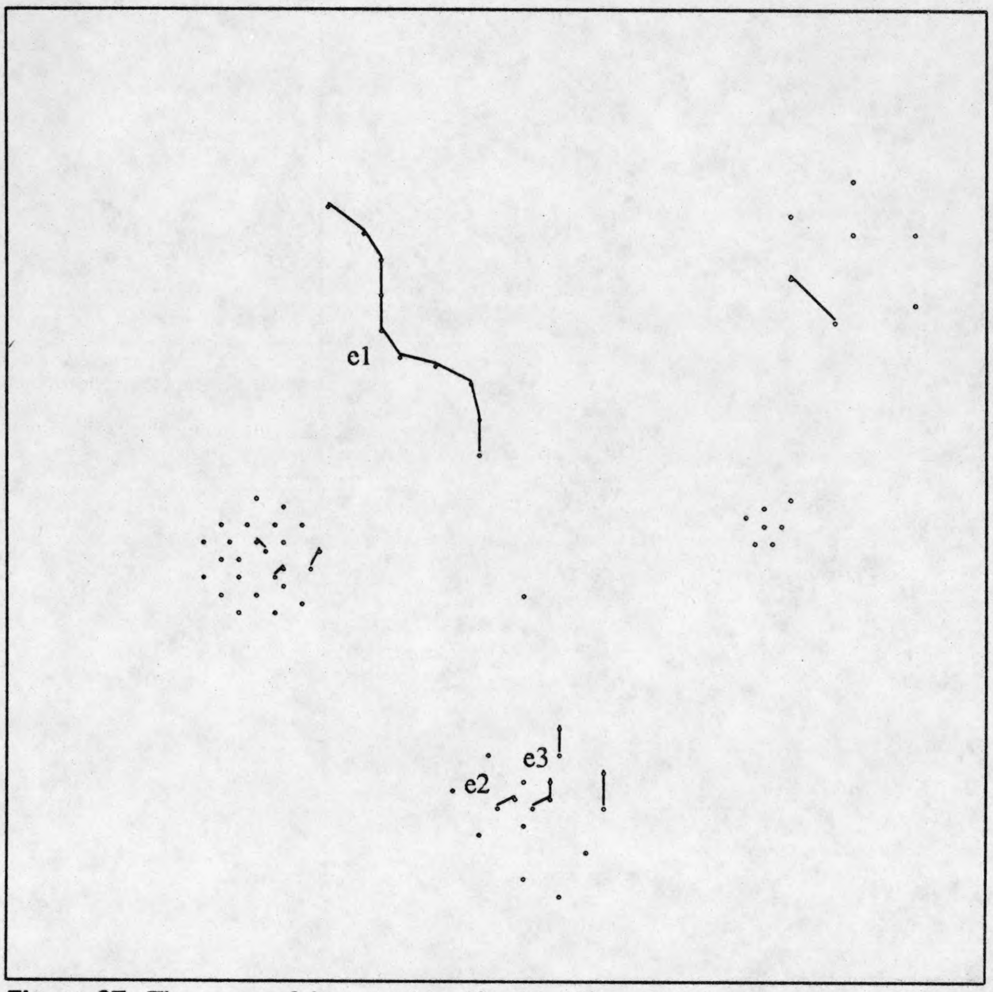


Figure 27. The output of the curve identification (CI) module for the pattern in Figure 15.

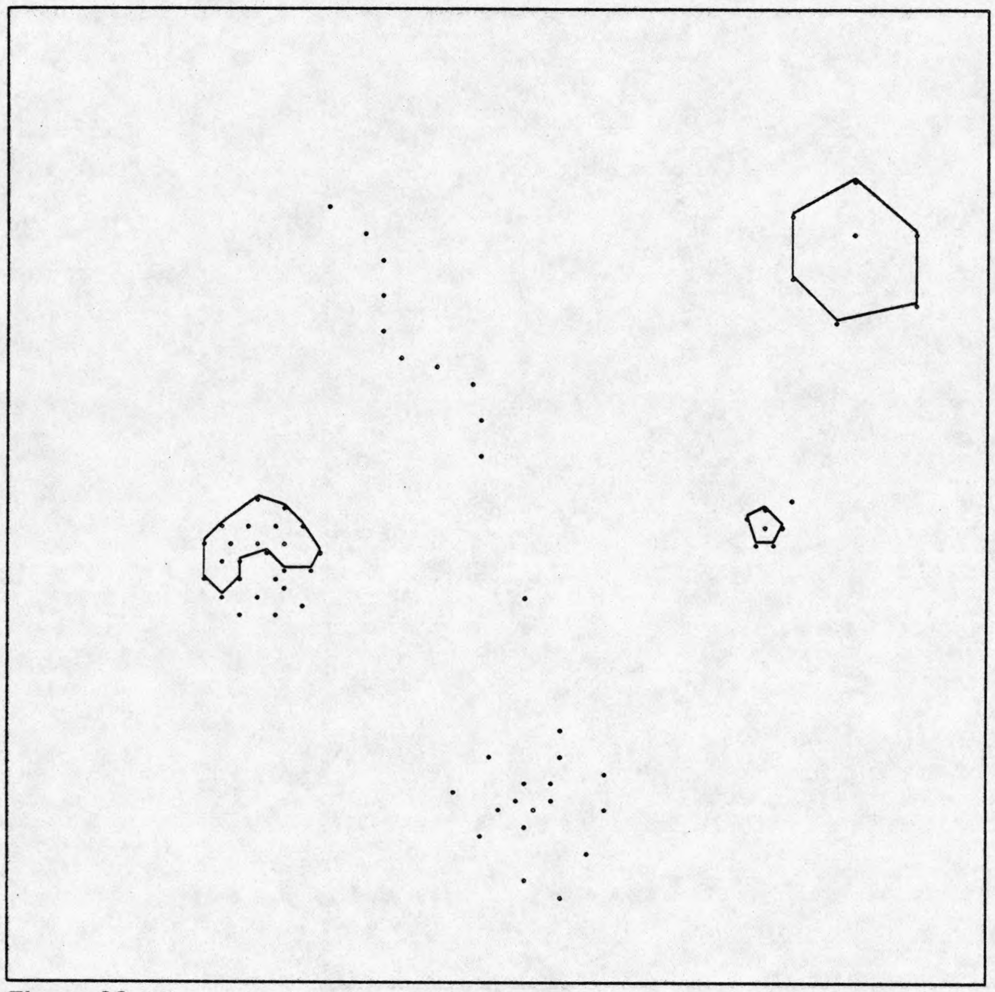


Figure 28. The result of running the interior correction (IC) module on the outputs of II and BI in Figures 25 and 26.

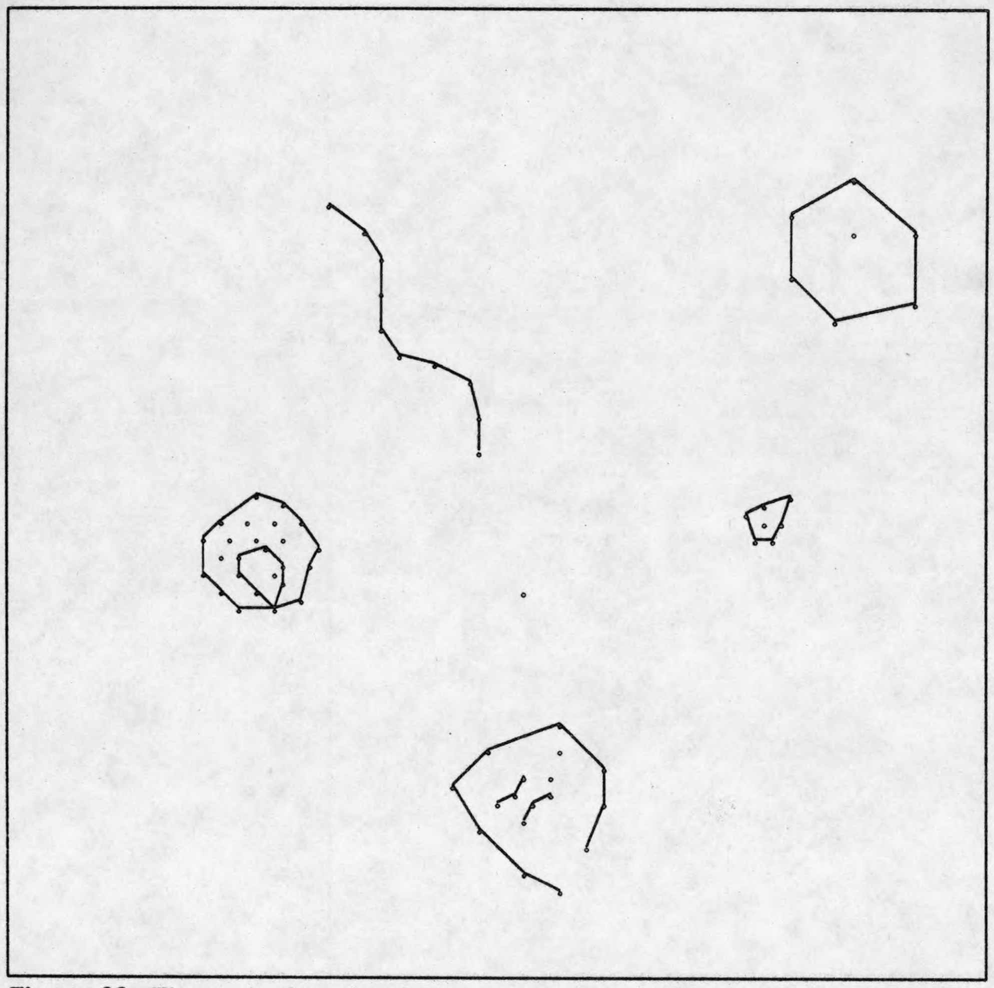


Figure 29. The result of running the border correction (BC) module on the outputs of II and BI in Figures 25 and 26.

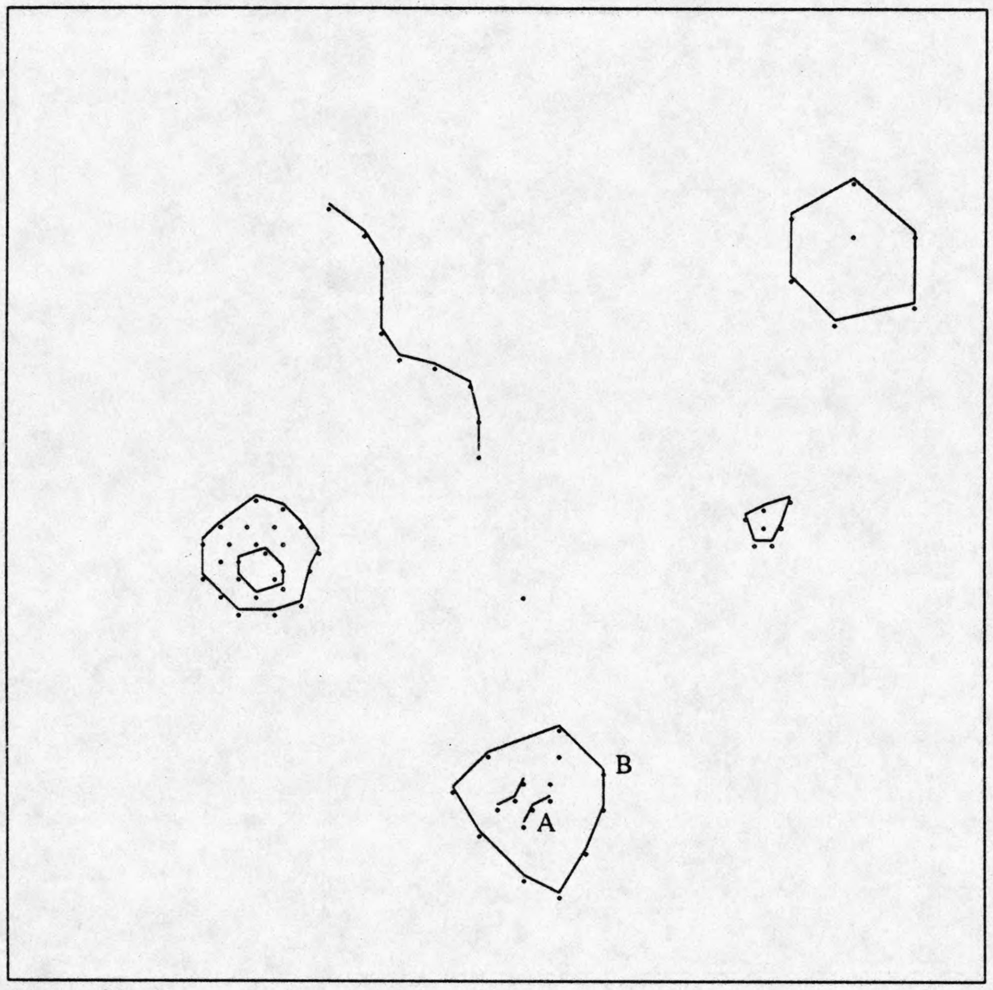


Figure 30. The result of running the combination (IBC) module on the outputs of IC and BC in Figures 28 and 29.

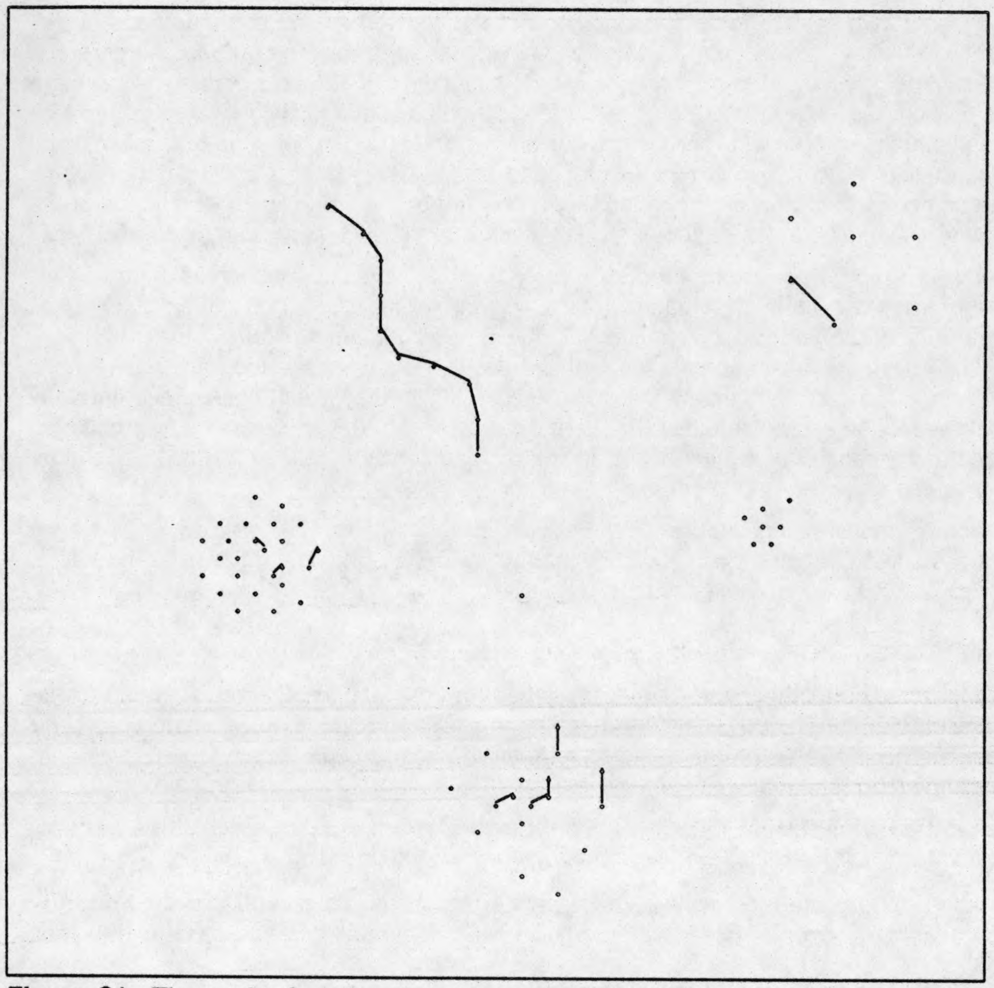


Figure 31. The result of running the curve correction (CC) module on the outputs of BC, CI and IBC in Figures 29, 27, and 30.

5.3. An Implementation

To define a computational approach to grouping, we need to answer the following questions. How do we relate the locally perceived role of a dot to its geometric structure? What are the Gestalt constraints on the joint interpretations of different dots? How does a formal inference procedure combine the information in geometric structure and the Gestalt constraints, to identify the perceptual structure?

The first question concerns relating a dot's role to its geometric structure. The measures we discussed in Section 4 represent different attributes of a dot relate to its perceptual role. These attributes must be appropriately combined to obtain evidence in support of a given perceptual role of the dot. It needs to be determined exactly how the attribute values must be mapped onto the different perceptual roles. Further, only a fluid interpretation is desired at this stage since the interpretation will be subject to further revision when global constraints are taken into account. Thus, we should define functions of attributes such that the function values reflect only the *degrees* to which a dot acts as different perceptual entities instead of assigning a single role to the dot. We have chosen these functions to yield numerical values between 0 and 1 indicating the strength of a perceptual role that a dot performs. The function may be viewed as the probability that the dot is the given perceptual entity, since the values of the function are required to sum up to 1 over all perceptual roles. The net result of the local geometric analysis is then a probability vector assigned to each dot whose elements are the likelihoods of the dot fulfilling various perceptual roles based on local evidence.

The second question is about how the Gestalt constraints restrict the allowed spatial configurations of different perceptual elements such that they have desirable global characteristics. For example, the local interpretations (probabilities) of a set of dots as being along a curve will be strengthened if the dots actually lie along a smooth curve. Similarly, the probabilities that a set of dots are border dots will be strengthened if the dots lie along a smooth curve and surround a compact component. The Gestalt constraints thus relate to both one-dimensional and two-dimensional aggregations of dots. Clearly, to enforce such constraints not only curves and borders must be extracted, but even connected components of dots must be identified. This implies multiple levels of analysis of dot pattern which must be performed on tentative interpretations, the latter subject to change as a result of the analysis.

The third question concerns the specification of a process of inference that would utilize the global constraints discussed in the second question to refine the interpretations obtained in answering the first question. Such a process must combine the probabilities of the various perceptual roles of a dot, each having a value between 0 and 1, with similar probabilities for the remaining dots, to obtain the probabilities of globally supported roles, each having a value between 0 and 1. The mutual interaction among the dots is according to the Gestalt constraints. In many cases, the result of the global interaction can be simulated by iterative local interaction. Thus, a dot's probability vector may be modified based on the probability vectors of the dots in the immediate vicinity, with successive iterations of the process achieving spatial propagation of a dot's influence.

We will now describe our current implementation of the lowest level grouping. We will do so by giving details of the implementation of each module in Figure 24. Each module uses probabilistic relaxation for inference, therefore, we will first briefly review the relaxation labeling process.

5.3.1. Relaxation Labeling

Relaxation labeling is a technique of parallel constraint propagation for obtaining locally consistent interpretations (labels) of a class of objects [26]. When the context allows multiple interpretations of a single object, the (probabilistic) relaxation process orders them according to their likelihoods. The local consistency of interpretations of objects is based on *a priori* knowledge about the way objects interact.

Formally, we have a collection of objects a_i, $i=1,...,n$, and a set of labels λ_i, $i=1,...,m$. Each object has a set of probabilities $p_i(\lambda_j)$, assigned to it that represent the likelihood of object a_i having label λ_j, where $\sum_j p_i(\lambda_j)=1$. Each object is assigned a set of initial probabilities, $p_i^{(0)}(\lambda_j)$ as described later in this chapter. Then, the probabilities are updated iteratively, obtaining a new set of probabilities after each iteration. The most commonly used updating formula is the following [26]:

$$p_i^{(k+1)}(\lambda)=\frac{p_i^{(k)}(\lambda)[1+q_i^{(k)}(\lambda)]}{\sum_\lambda p_i^{(k)}(\lambda)[1+q_i^{(k)}(\lambda)]}$$

where

$$q_i^{(k)}(\lambda)=\sum_j d_{ij}\,[\sum_{\lambda'} r_{ij}(\lambda,\lambda')p_j^{(k)}(\lambda')]$$

The term $r_{ij}(\lambda,\lambda')$ in the last expression represents the compatibility of labels λ and λ' on objects a_i and a_j, respectively. The d_{ij}'s are weights associated with the interaction of the pair of objects a_i and a_j, where $\sum_j d_{ij}=1$, in order to keep the probabilistic properties of p_i. Thus $q_i^{(k)}(\lambda)$ represents the support given to the label λ at the object a_i by all the neighbors, a_j, of a_i.

5.3.2. Interior Identification

The interior identification is formulated in a probabilistic relaxation scheme with the dots being labeled as either *INTERIOR* or *NONINTERIOR*. This formulation is based upon the local geometrical properties of the Voronoi neighborhoods. This task, of course, is only meaningful if the clusters of dots have nonempty interiors.

The major step is to formulate the local compatibilities between pairs of dots in terms of the geometric properties of their polygons. Specifically, in the interiors of homogeneous clusters, the areas of Voronoi polygons are approximately the same and the eccentricities of the cells are low. In the interiors of nonhomogeneous clusters the eccentricities are high but they are pointing in the same direction, namely, in the increasing density direction. These facts, used conservatively, will result in the most obvious interior dots being identified. Those local regions where there might be noise effects on the positions of the dots even though they are in the interior of a segment, will either be classified erroneously or the labeling will not be confident.

The relaxation compatibilities are defined for the four possible label combinations at two dots, i and j, namely 1) $INTERIOR_i$-$INTERIOR_j$, 2) $INTERIOR_i$-$NONINTERIOR_j$, 3) $NONINTERIOR_i$-$INTERIOR_j$, and 4) $NONINTERIOR_i$-$NONINTERIOR_j$. In order to define these compatibilities we have to consider all the possible contexts in which a given combination of labels can occur. For example, all possible contexts in which the combination *INTERIOR* and *NONINTERIOR* can occur are shown in Figure 32. For each of these contexts an expression is written which summarizes the context through the use of relevant geometric properties that characterize the given context. For example, the case shown in Figure 32(a), represents the context in which two dots can occur in the interior of a homogeneous cluster. Thus, whenever this context is observed, it counts as a support for the labels *INTERIOR-INTERIOR*. Equivalently, the support, or compatibility, of the labels *INTERIOR-INTERIOR* is obtained by considering all contexts including the one in Figure 32(a) that support the label pair *INTERIOR-INTERIOR*. An example of an expression for the contribution of the context shown in Figure 32(a), to the compatibility of *INTERIOR-INTERIOR* is

$$min(1-ecc_i, 1-ecc_j, 1-\Delta_{ij}) \quad (1)$$

In this expression ecc_i and ecc_j are the eccentricity magnitudes for the Voronoi polygons of the cells of dots i and

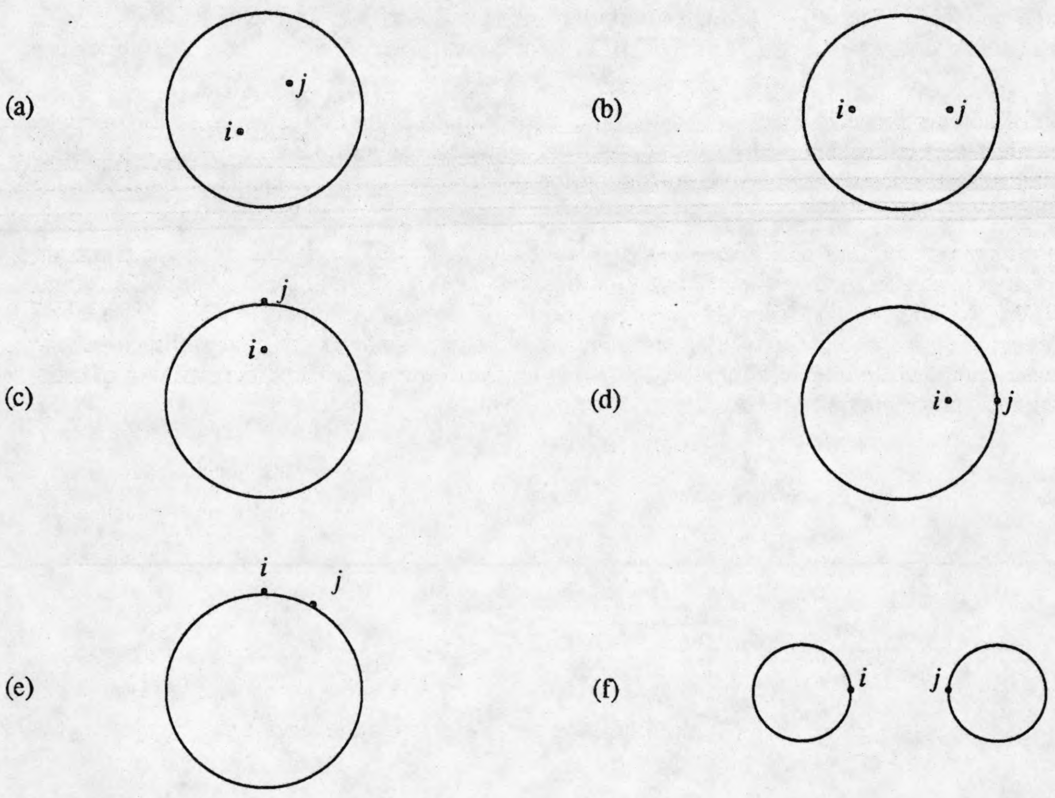


Figure 32. The possible contexts for which *INTERIOR* and *NONINTERIOR* compatibilities are computed. (a) *INTERIOR-INTERIOR* in homogeneous clusters. (b) *INTERIOR-INTERIOR* in non-homogeneous clusters. (c) *INTERIOR-NONINTERIOR* in homogeneous clusters. (d) *INTERIOR-NONINTERIOR* in non-homogeneous clusters. (e) *NONINTERIOR-NONINTERIOR* on the border of the same cluster. (f) *NONINTERIOR-NONINTERIOR* on the borders of different clusters.

j, respectively. Δ_{ij} is the area difference of the two polygons normalized to the range [0,1] and is defined as $\Delta_{ij} = abs(A_i - A_j)/max(A_i, A_j)$, where A_i and A_j are the areas of the polygons for the dots i and j, respectively. The intuitive meaning of the expression is that in the interior of a homogeneous cluster the eccentricity magnitudes and the area differences are expected to be small. If that is the case with the two given dots, then expression (1) will have a high value and will have a positive contribution to the *INTERIOR-INTERIOR* compatibility value. After the expressions for all cases in which labels *INTERIOR-INTERIOR* occur have been derived, the expressions are combined by a fuzzy *OR* operation. In particular, for two dots i and j to be compatible with labels *INTERIOR-INTERIOR*, they must have the context shown in case (a), or case (b), of Figure 32. Similarly, expressions are obtained for other combinations of labels for the two dots and then combined to obtain label compatibilities. A complete list of the compatibility expressions is given in Appendix B.

Once the compatibilities have been defined, relaxation labeling is performed to assign to each dot probabilities of being *INTERIOR* or *NONINTERIOR* (step A, module II in Figure 24). Most of these probabilities converge to either very high or very low values resulting in unambiguous labelings (even though they may be the wrong labels). In case of the few dots that have ambiguous probabilities, we assign them the label with the stronger probability. If this turns out to be the wrong label, the later phases will correct this, taking into consideration a larger context (step B in Figure 24), information coming from other independent modules and Gestalt assumptions such as border smoothness (steps B and C), and closure (step C). Table 1 shows the label probabilities of some representative points in the sample dot pattern in Figure 25 as the probabilities change through the iterations.

5.3.3. Border Identification

The borders are the segments of curves that surround interior regions of dots. The surrounded interior region might be nonempty and contain dots labeled as *INTERIOR* or it might be empty in which case the dot cluster is a bar-like structure (Figure 33).

The identification of borders proceeds very similarly to the interior identification. However, in this module, the objects to be labeled are the Delaunay edges instead of dots. This enables the algorithm to detect not only which dots lie on the border, but also to identify successive dots along the border. The possible labels of the Delaunay edges are *BORDER* and *NONBORDER*.

The computation of the compatibility coefficients in this module also proceeds very similarly to the computation of the compatibility coefficients in the interior identification module. We first look at the four possible combinations of the two labels on two neighboring Delaunay edges and identify the possible contexts in which these labels could arise. A representative list of these contexts is given in Figure 34. Once again, all these possibilities and the values of the measures are combined to obtain the final expressions for the compatibility coefficients. An exhaustive list of these expressions is given in Appendix B.

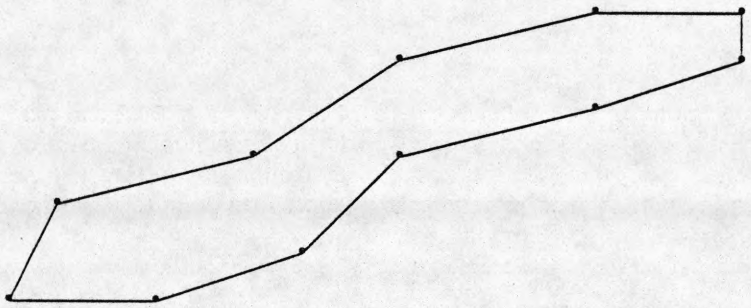


Figure 33. A barlike structure is one which contains no interior points within its borders.

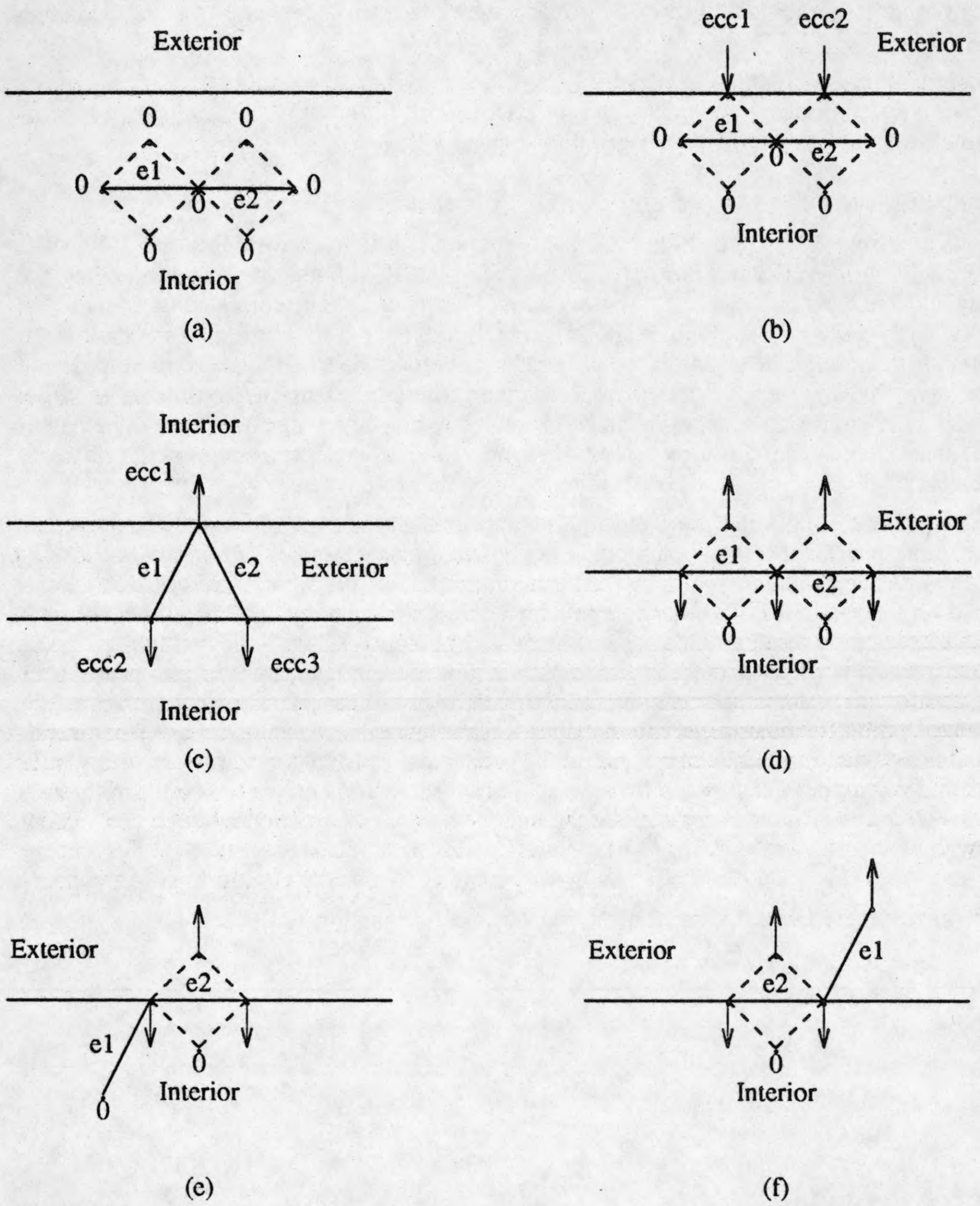


Figure 34. Schematic illustration of some example contexts for the labels *BORDER* and *NONBORDER* and the expected values of the eccentricities. (a) Both Delaunay edges e1 and e2 and their lateral neighbors are inside a homogeneous cluster. (b) The lateral neighbors on the same side of e1 and e2 are on the border of a homogeneous cluster. In this case the eccentricities ecc1 and ecc2 of the dots on the border are expected to point towards the interior. (c) The two edges e1 and e2 are between two clusters. Here the eccentricities on the two borders are expected to point away from each other. (d), (e), and (f) show the expected values of eccentricities in some other possible contexts.

5.3.4. Curve Identification

Curves in dot patterns are borderlike structures with adequate separation on both sides from other structures. In the case of cluster borders, one of the sides has a high density of dots whereas the other side usually has a large amount of empty space. Curves are well separated from other dots on both sides.

Curve identification is similar to border identification. The Delaunay edges are the objects to be labeled, with labels *CURVE* or *NONCURVE*.

The computation of compatibility coefficients is also similar to the case of border identification. The contexts for the curvilinear structures are different than the borders, and the compatibility expressions are computed accordingly. The computation of the compatibility coefficients is given in Appendix B.

5.3.5. Label Corrections to Enforce Agreement and Border Smoothness

As a result of the previous step (step A in Figure 24), the dots and the Delaunay edges are labeled as *INTERIOR-NONINTERIOR*, *BORDER-NONBORDER*, or *CURVE-NONCURVE*. Some of these labels may not be correct due to lack of local evidence, ambiguities, etc. These incorrect labels need to be corrected by using information from a larger context. The results of the modules (II, BI, and CI) and their comparison provide part of the necessary information from a larger context. In addition, the criterion that borders be smooth is also used to decide whether a labeling of a dot or a Delaunay edge needs to be corrected. The context that is considered is larger because, for example, a border segment which is expected and constrained to be smooth extends beyond the neighborhood of the dot or Delaunay edge being considered for correction. Similarly, if the results of the different modules are forced to agree, this involves combination of information from different contexts.

Before going into the details of the algorithm for correcting the labels at this step, there is a minor representational point to be clarified. In the interior identification module the dots are labeled as *INTERIOR* or *NONINTERIOR*, but there is no explicit border information, which is needed in order to make the appropriate cross comparisons between the outputs of II and BI modules. In order to get comparable representations for the outputs of both modules, the interior regions identified by the II module are surrounded by borders. This is done by the rules given in Figure 35. Except for clusters with empty interiors, this transformation makes it meaningful to speak about agreement between the two results, for example, by computing the fractions of border segments that are shared by the outputs of the two modules. Similarly, the transformation makes it feasible to enforce smoothness of borders in the interior identification module. The clusters with empty interiors, i.e. bars and isolated dots, do not have any interior dots and hence no border edges are provided by the II module. Therefore, there is no way to compare the two sets of output in these cases and they are handled separately in the interior-border combination module (IBC) in the next step.

The label correction step consists of two modules (IC and BC in Figure 24). Each one concurrently and independently corrects one set of labels from the previous step using the information from the previous step as

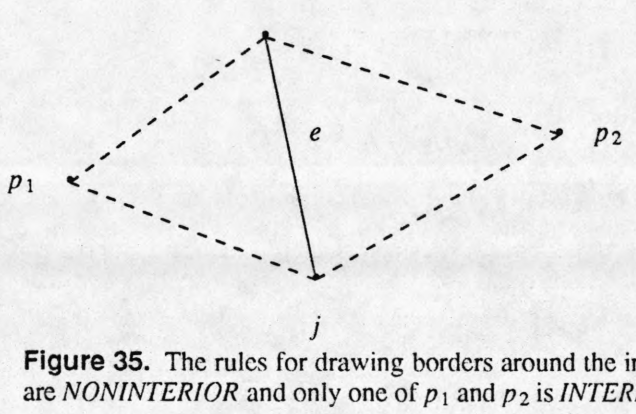


Figure 35. The rules for drawing borders around the interior regions. The edge e is converted to border if i and j are *NONINTERIOR* and only one of p_1 and p_2 is *INTERIOR*. Otherwise, the edge is marked as nonborder.

shown in Figure 24. Not all of the dots or the Delaunay edges are considered for correction. The most confident ones (i.e. the dots or Delaunay edges whose identifications from the two independent modules II and BI are in agreement) are omitted. Only the objects whose identifications from the two independent modules conflict are considered for correction. This increases the efficiency of the correction process. A module changes the labels of its input if doing so improves the measure of border smoothness and increases agreement with the results of other modules. Some representative contexts for each set of inputs coming from the previous stage and the effect of changing the label are shown in Figure 36. The correction process is also formulated in a probabilistic relaxation scheme with the labels { *CHANGE*, *NOCHANGE* } on the objects. An object which has the label *CHANGE* at the end of the correction process has its label from step A changed to the corresponding complementary label. If its label is *NOCHANGE* the original label is retained. The objects are dots (for the correction of dot labels), and Delaunay edges (for the correction of border identifications). The computation of the compatibilities proceeds by examining the border segments around a particular object, before and after a label change. An example of computing compatibilities r_{ij} for objects i and j is as follows:

$$r_{ij}=\frac{((1-curv_i)+(1-curv_j)+0.5(agr_i+agr_j))}{3}$$

In this expression, *curv* stands for the value of the curvature of a border segment at the objects i and j normalized to the range 0 to 1. The expression *agr* is a measure of the agreement of the results of the two independent modules normalized to the range 0 to 1. Therefore, this computation reflects the expectation that the curvatures of borders be minimized (i.e. the border smoothness be maximized) and the agreements of the results between different modules be high.

Once the correction of the interior and border identifications is completed, then the necessary changes are made and the correction process described above is iterated on the new set of identifications. This iteration is

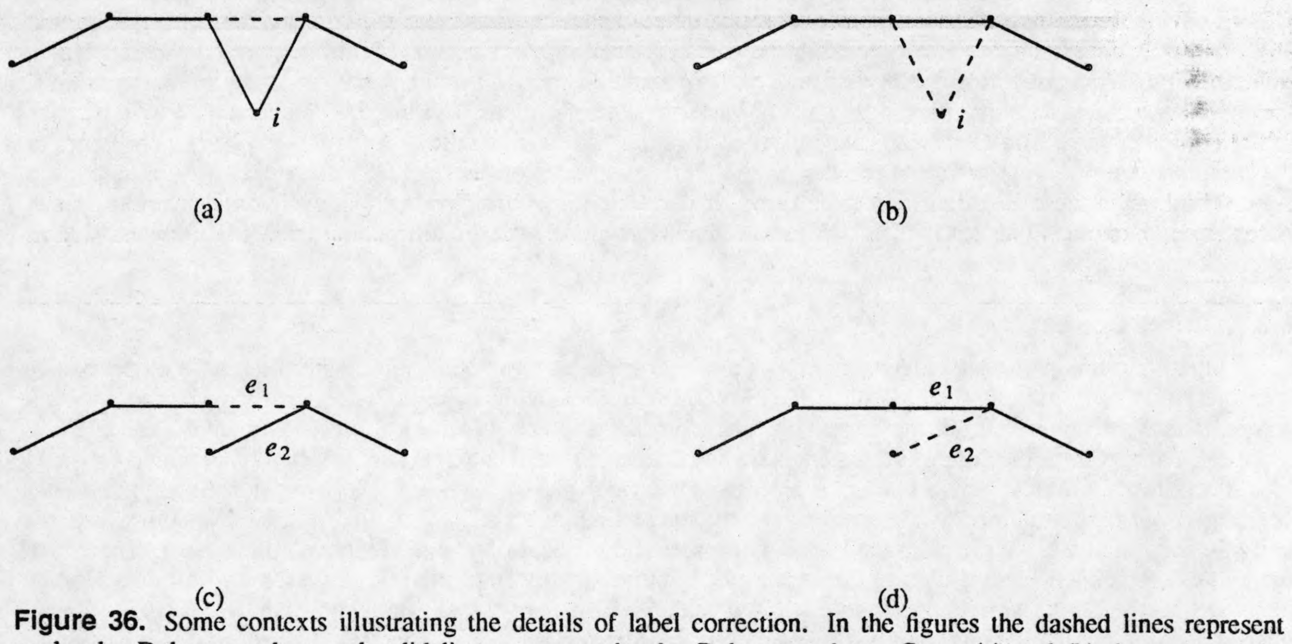


Figure 36. Some contexts illustrating the details of label correction. In the figures the dashed lines represent nonborder Delaunay edges and solid lines represent border Delaunay edges. Cases (a) and (b) demonstrate the effect of changing the label of a point by the interior correction (IC) module. In (a) when the label for point i is changed from *NONINTERIOR* to *INTERIOR*, the change is reflected in the border around it as shown in (b). Cases (c) and (d) demonstrate the effect of changing labels on the Delaunay edges by the border correction (BC) module. In (c), when the labels for Delaunay edges e_1 and e_2 are changed, the new labels in (d) are obtained.

necessary in order to propagate the effect of the newly changed labels. This iteration proceeds until there are no more label changes. These corrected results are then combined to get a final segmentation in the next step.

5.3.6. Combining the Results to Enforce Component Compactness

In this step (step C in Figure 24), the corrected results from step B are combined with the aid of assumptions about more global properties such as closure of borders. It consists of two modules: a) interior border combination module (IBC in Figure 24), and b) curve correction module (CC in Figure 24). In the interior border combination module, a connected component analysis is performed as described below.

First, the borders around the dots labeled as interior by the module IC are identified as described in the previous step. This results in border segments that surround the interior regions. Then, the intersection of these results and the results of module BC is taken. This results in those Delaunay edges being identified as border that have confirmation from two independent processes. The result is a set of border segments and a set of interior dots next to them. Each of these border segments is given a label (e.g., they are numbered). The interior dots then are assigned the labels of all the border segments that surround them. That is, a dot P is assigned the number of a border segment B, if there exists a path $p_1 p_2 \cdots p_k$ through neighboring points such that $p_1 = P$, p_k is on the border segment B, and all the dots $p_1 p_2 \cdots p_{k-1}$ are labeled interior. The result is that all the interior dots are assigned labels of one or more border segments. The goal is to have all these border segments to form a closed contour since they surround the same component of dots. Note that the number of final border segment labels associated with a single interior region may be more than one since a component may have holes in it.

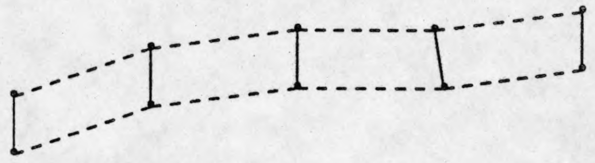
The combination process proceeds with the interior dots that have only one component label assigned to them as described above. If the border is closed no further processing is done on that region. If the border is not closed, then the process attempts to extend it with the eventual goal of closing it and, at the same time, ensuring that the border segment is extended smoothly. If there still remain border segments around a region that are not closed, they are extended as smoothly as possible so that some of these segments will either merge with each other to form longer border segments to be further processed, or they will become closed, thus ending the processing of that region. While combining the results of the correction step (step B in Figure 24), one must be careful in handling the regions which are bar-like (i.e. two segments of parallel borders with no interior region between them) (see Figure 33). These are important because they might be part of a neck in a cluster and if they are not considered at this stage then problems arise in trying to close the borders being extended. Problems will arise due to the fact that if these border segments are not merged with the border segments of regions with interior dots then there will be gaps in the border of the entire cluster and its closure will be missed. To avoid this difficulty the border segments of these regions with no interiors are merged with the border segments of the regions with interiors if possible. The contexts in which there is a transition from a region with interior to a region without an interior in a cluster can occur are limited (see Figure 51). This contextual knowledge along with the criterion of border smoothness is used when merging the border segments.

5.3.7. Curve Correction

Finally, in this part of the algorithm curves are refined so as to be compatible with the results of the border and interior identification as well as some other criteria described below. Some dot patterns give rise to conflicts among the results obtained from the border identification (BI) and curve identification (CI) modules. Specifically, as seen in the results of Figure 37, when there is a series of pairs of dots along a smooth path, the border identification module detects the border of a segment with no interiors, whereas the curve identification module detects a set of curves defined by successive pairs of dots along the path. This conflict must be resolved by considering the distribution of distances of the identified borders and the context in which these conflicts exist. Therefore, the curves are refined by considering the result of the curve identification module (CI) along with those borders identified by the border correction module (BC) which are not part of the clusters with interiors. This is accomplished by looking at the configurations in which the conflict arises, namely those in Figure 37(a). In the given configuration the aspect ratios of the edges are computed and a threshold is applied so that if this aspect ratio is sufficiently large, then the longer edges are assumed to be erroneously labeled and, hence, deleted.

Besides the conflicting results between the two modules BI and CI, the curve identification module suffers from the same problems that the modules II and BI suffer, namely from the narrowness and locality of their expertise. One such problem in the results of the curve identification module is the erroneous labeling of long Delaunay edges aligned with but across curves as curves while they should really be labeled as gaps (see Figure 37(b)). This error occurs because the lateral context of the wrongly labeled edge supports the curve label and so does the alignment of the edge with its succeeding and preceding Delaunay edges which are also labeled as curve. No constraint

(a)



(b)

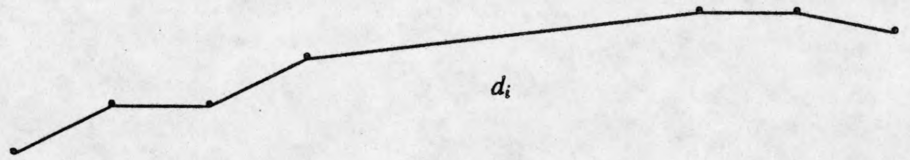


Figure 37. a) Possible configurations where conflicts may arise between curve and border labels. Dashed lines indicate the border edges and solid lines indicate the curve edges. b) A gap edge d_i that is labeled as a curve edge and needs to be corrected.

has been used to enforce the expected similarity of lengths of successive curve edges. This is now done by enforcing the condition that an edge cannot be labeled as *CURVE* if its length differs by more than a threshold from the preceding and succeeding curve segments. Specifically, considering Figure 37(b), if $d_i > \mu_d + n\sigma_d$ then the Delaunay edge i is deleted as part of the curve. Here d_i is the length of the edge i, μ_d is the mean Delaunay edge length of a chain of Delaunay edges on the two sides of edge i, and $\sigma_d{}^2$ is the variance of the lengths of edges in those chains.

5.3.8. Initial Probability Assignments

The initial probability vectors for the dots are computed based on the local compatibilities. For a given label at a dot, all the neighbors of the dot are scanned and for each neighbor the label with the maximum support for the original label is found. An average value of maximum supports from all neighbors is assigned as the net support of the original label at the given dot. The net support received by different labels are then normalized to the [0,1] range to obtain the estimates of initial probabilities of different labels. Thus, the net support $s_i(\lambda)$ of label λ at object i is given by:

$$s_i(\lambda) = \underset{j}{avg}\left\{\max_{\lambda'}\left\{\frac{1+r_{ij}(\lambda,\lambda')}{2}\right\}\right\}$$

which is then normalized over λ to obtain initial probabilities $p_i(\lambda)$,

$$p_i(\lambda) = \frac{s_i(\lambda)}{\sum_{\lambda'} s_i(\lambda')}$$

so that $\sum_{\lambda} p_i(\lambda) = 1$.

5.4. Results

The lowest level grouping algorithm described in this chapter was tested on a number of dot patterns. Some example patterns and the final results of the algorithm on them are shown in Figures 38-46.

The relaxation labeling is obtained by running it for 100 iterations which results in the final probabilities of each label on the dots to be either very high or very low. At the end of 100 iterations, the probabilities for most of the dots have converged to their final values (usually > 0.9 for the final label). Tables 1-3 show the probabilities of some sample points in Figure 15 as they change during processing for each module. As we can see in these tables,

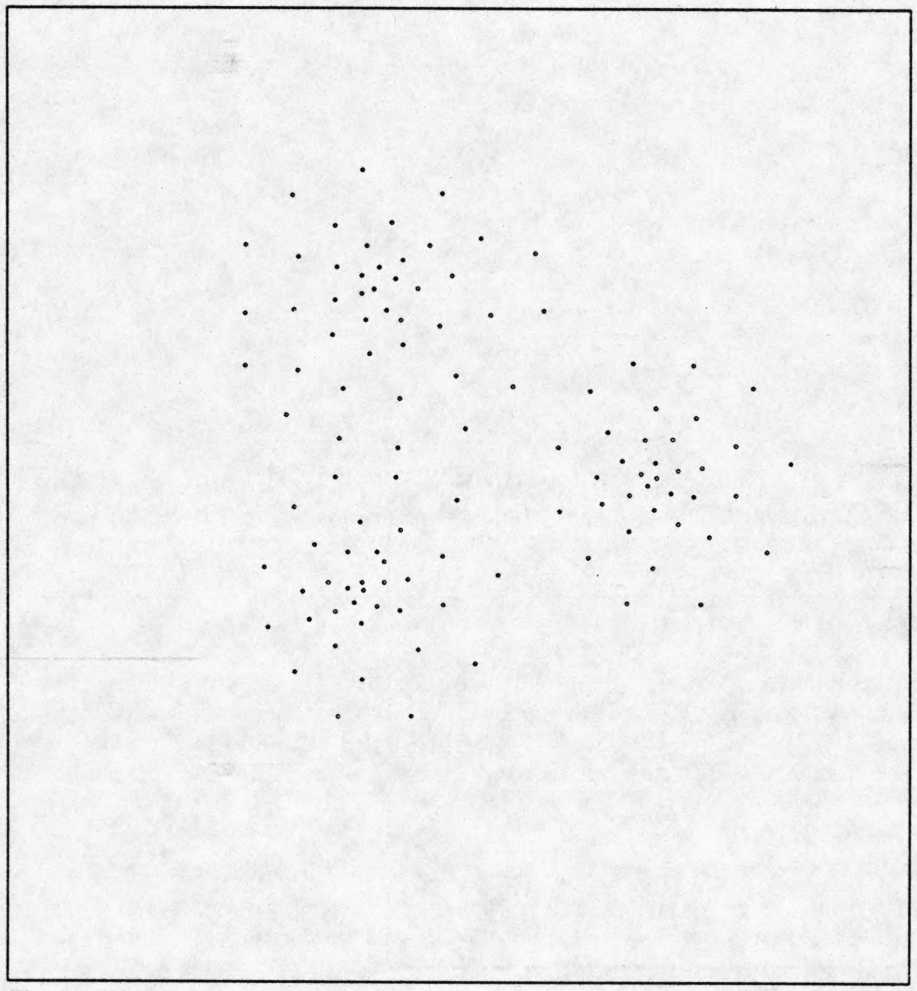


Figure 38(a). An example dot pattern.

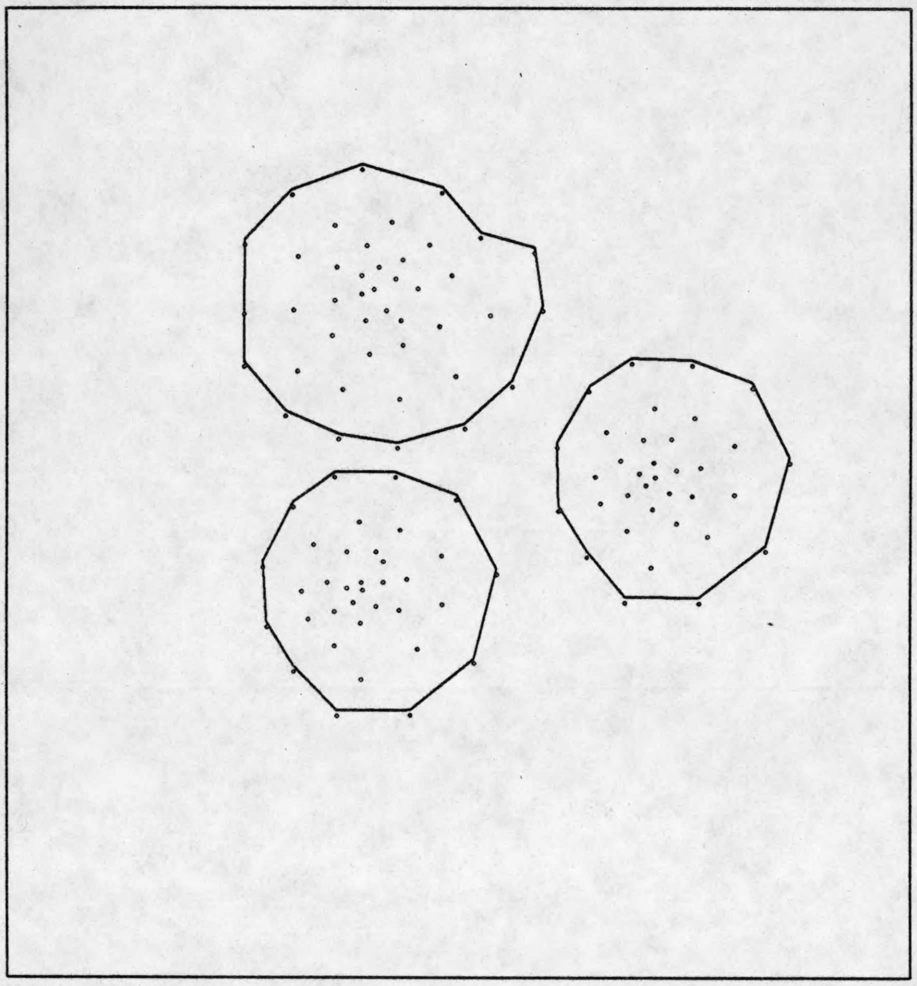


Figure 38(b). The resulting identification.

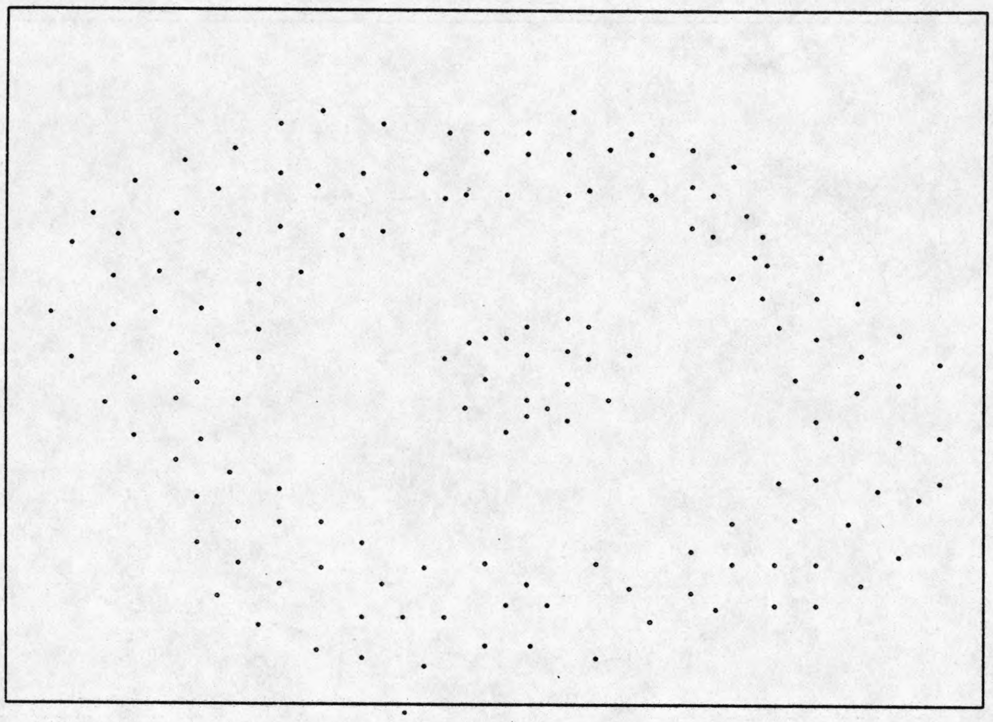


Figure 39(a). An example dot pattern (Figure taken from [38]).

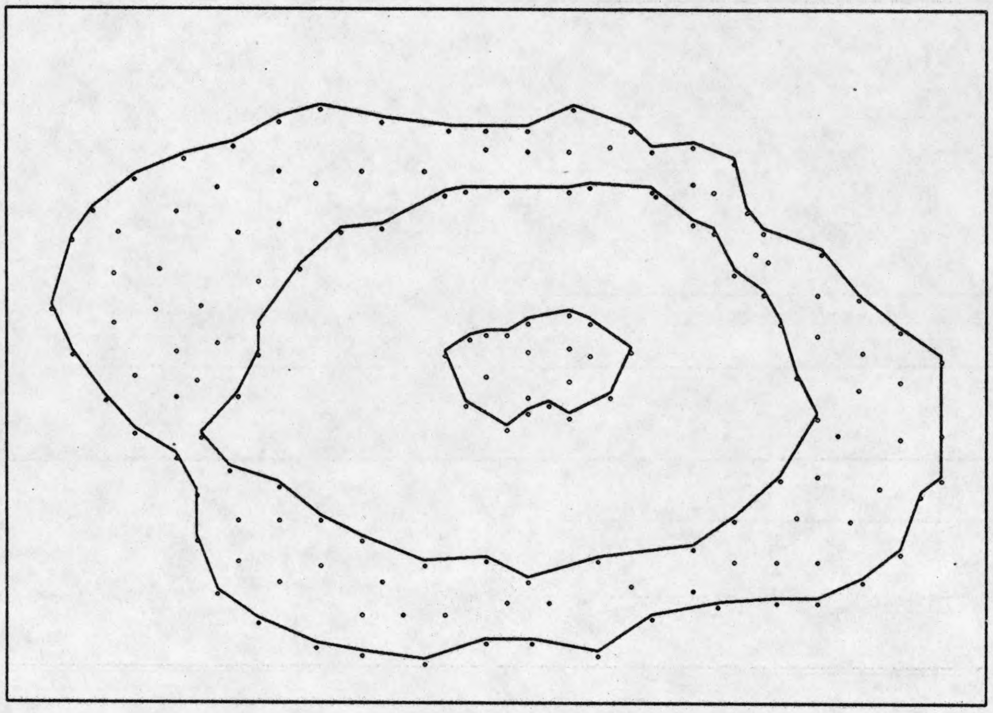


Figure 39(b). The resulting identification.

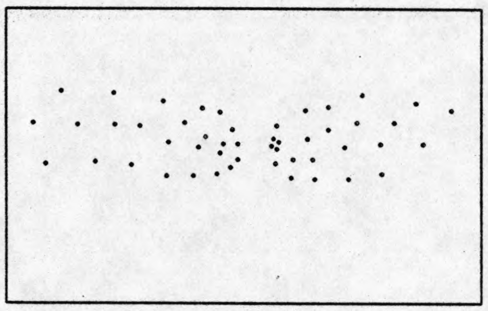


Figure 40(a). An example dot pattern.

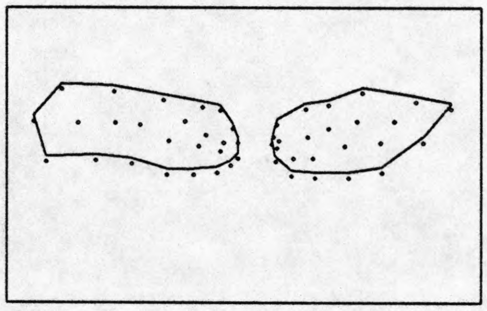


Figure 40(b). The resulting identification.

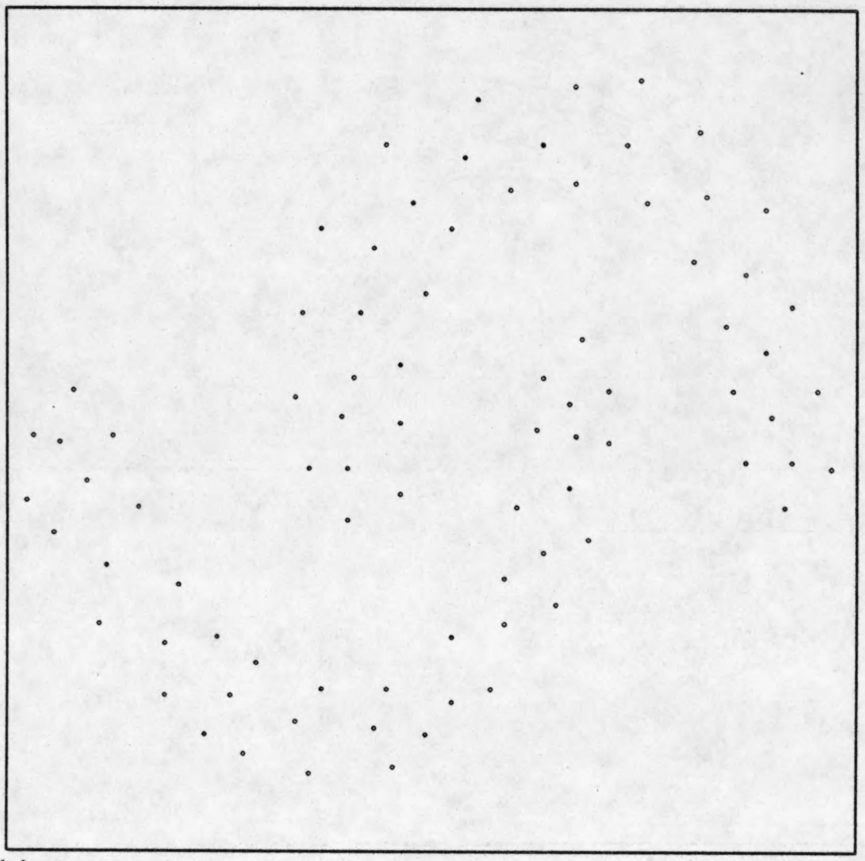


Figure 41(a). An example dot pattern [39].

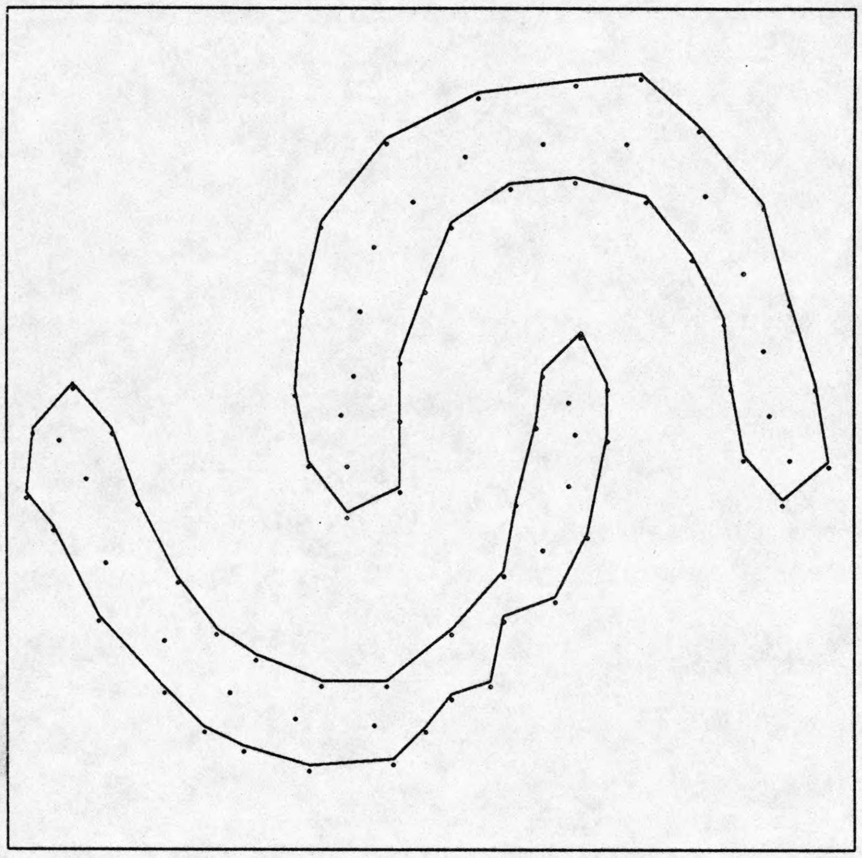


Figure 41(b). The resulting identification.

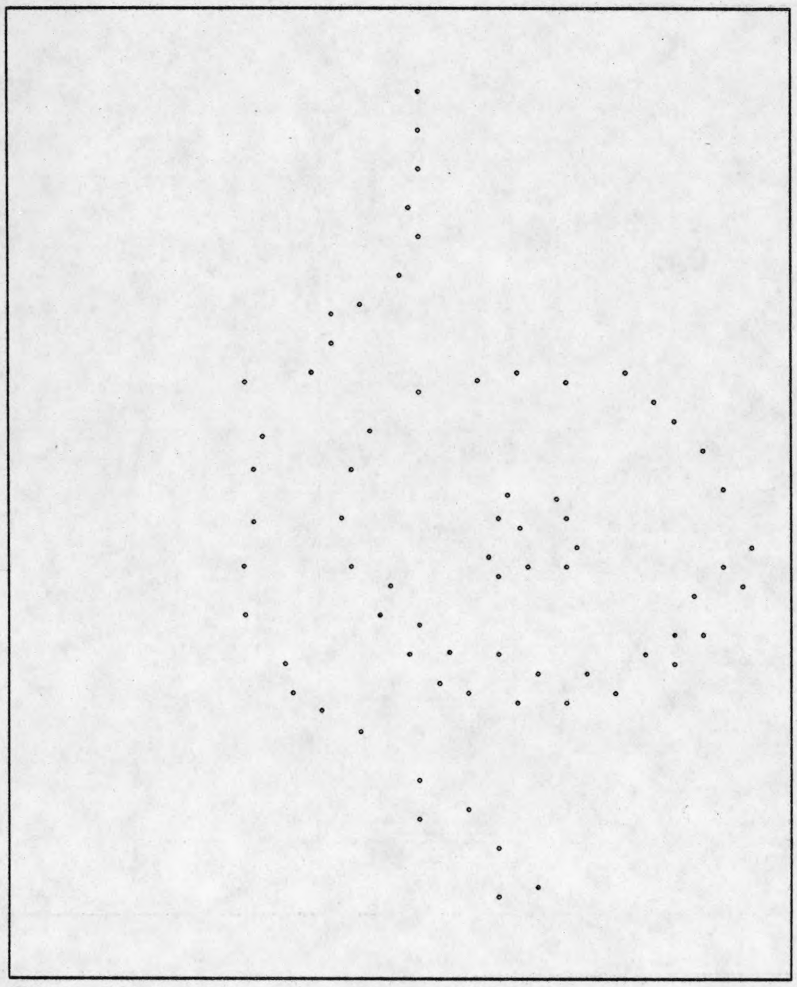


Figure 42(a). An example dot pattern [30].

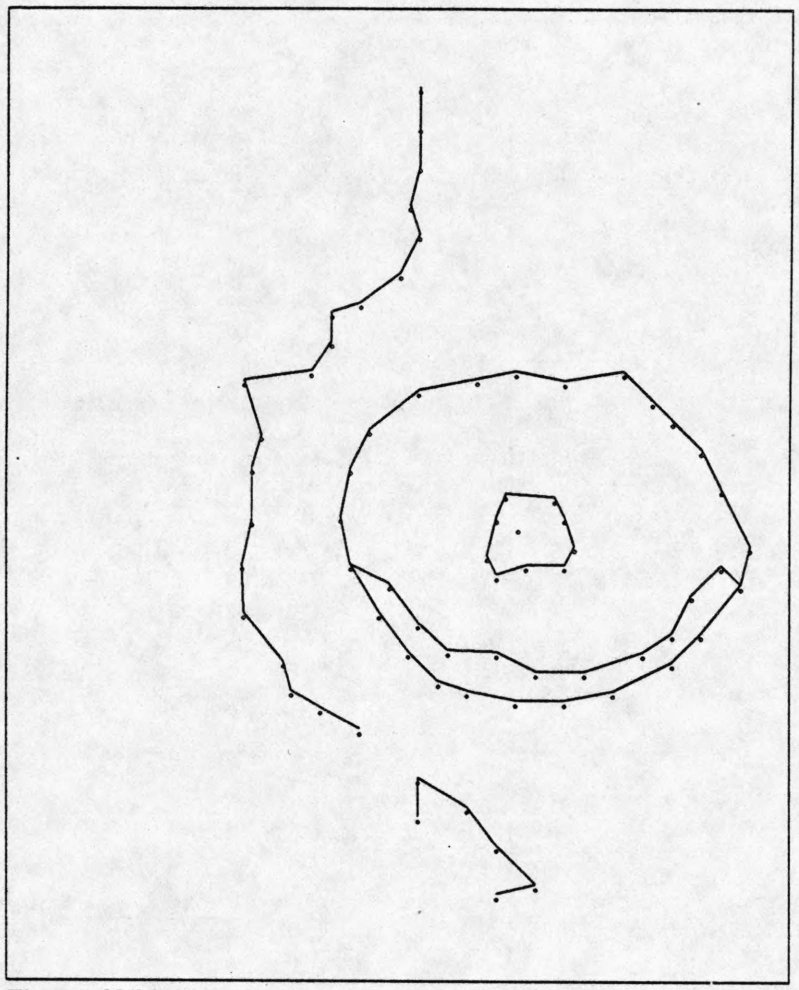


Figure 42(b). The resulting identification.

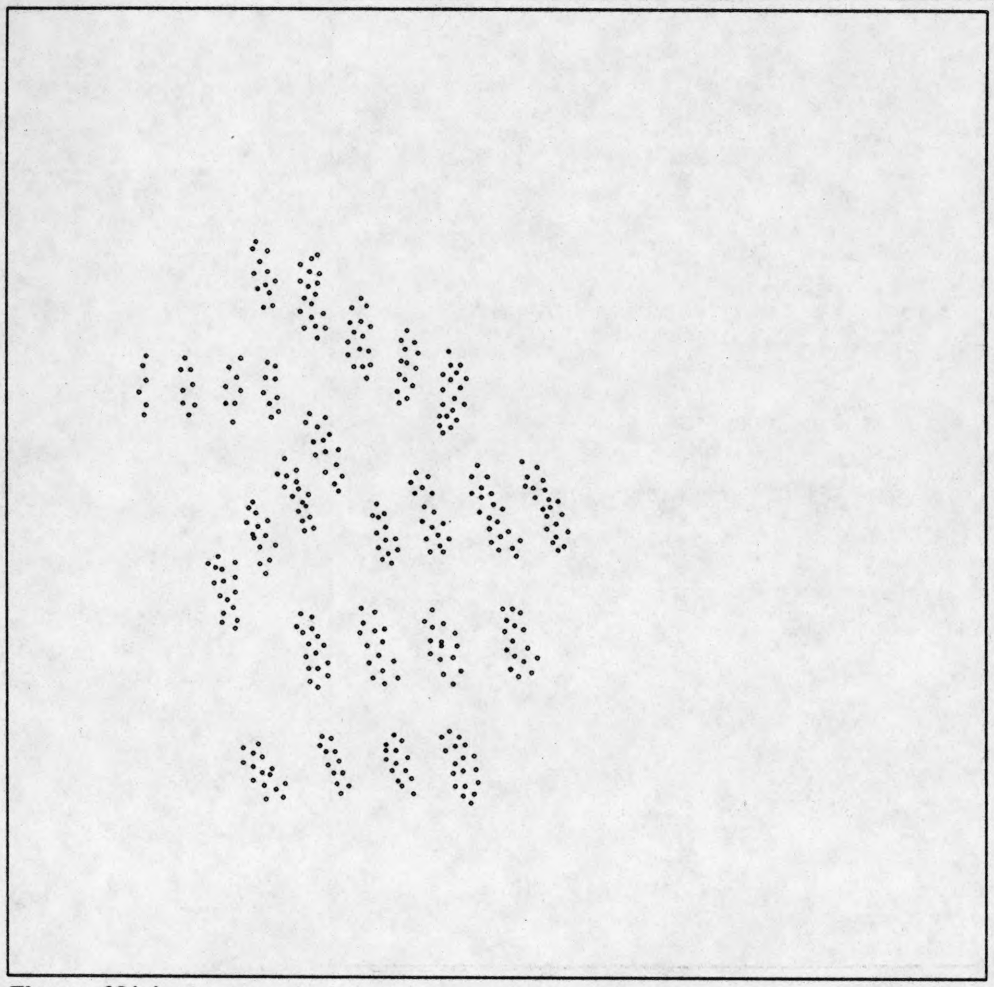


Figure 43(a). An example dot pattern.

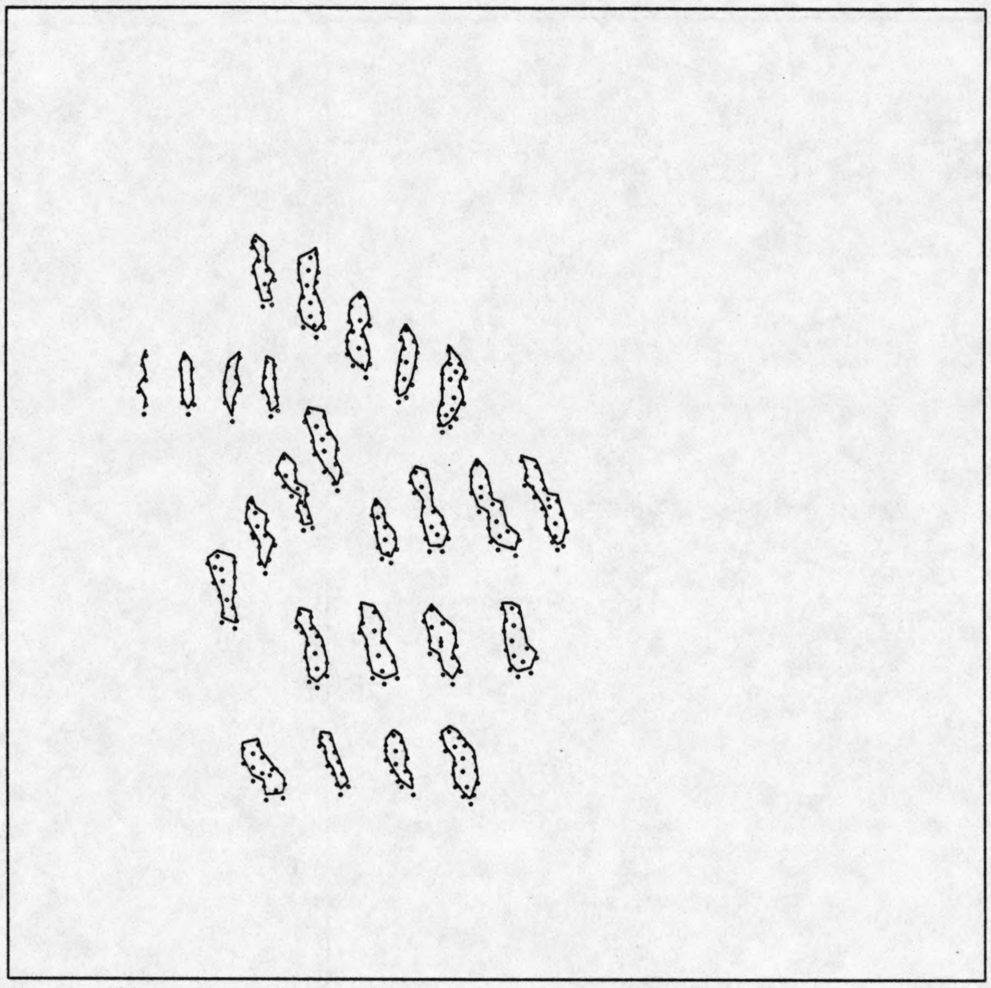


Figure 43(b). The resulting identification.

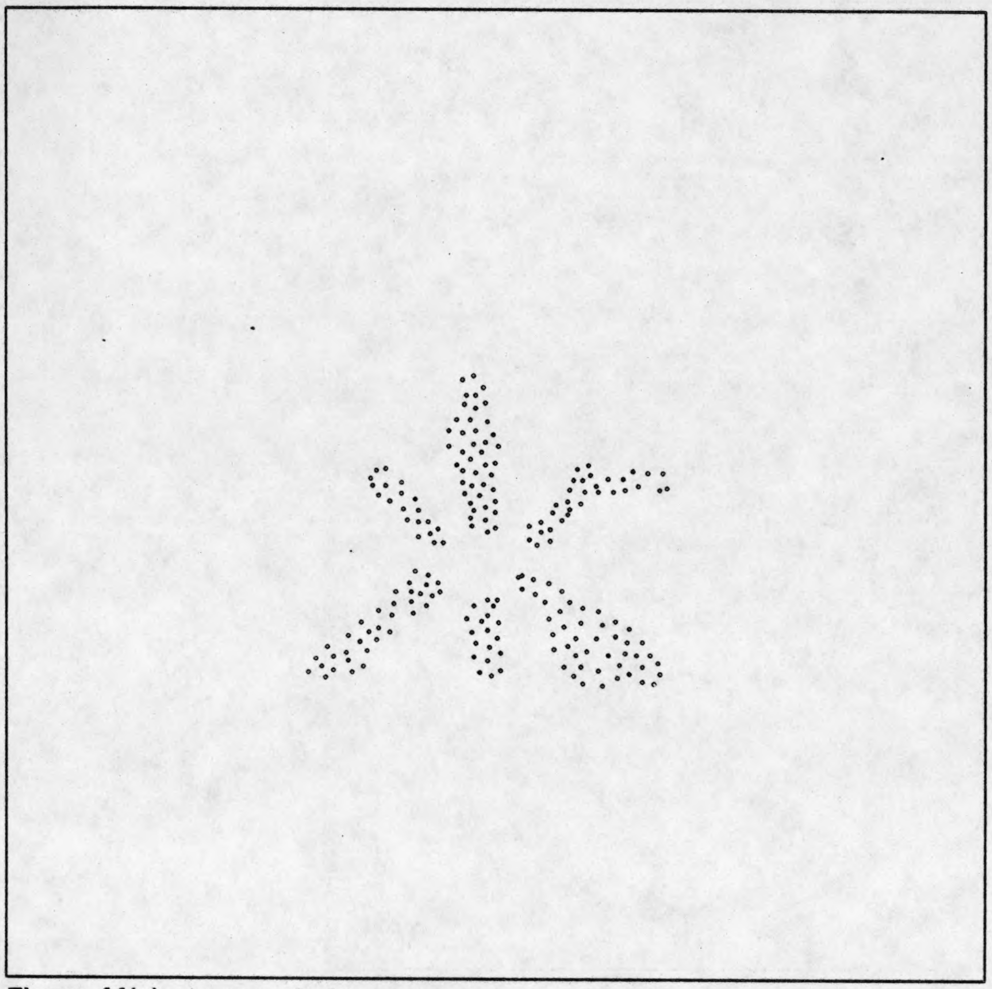


Figure 44(a). An example dot pattern.

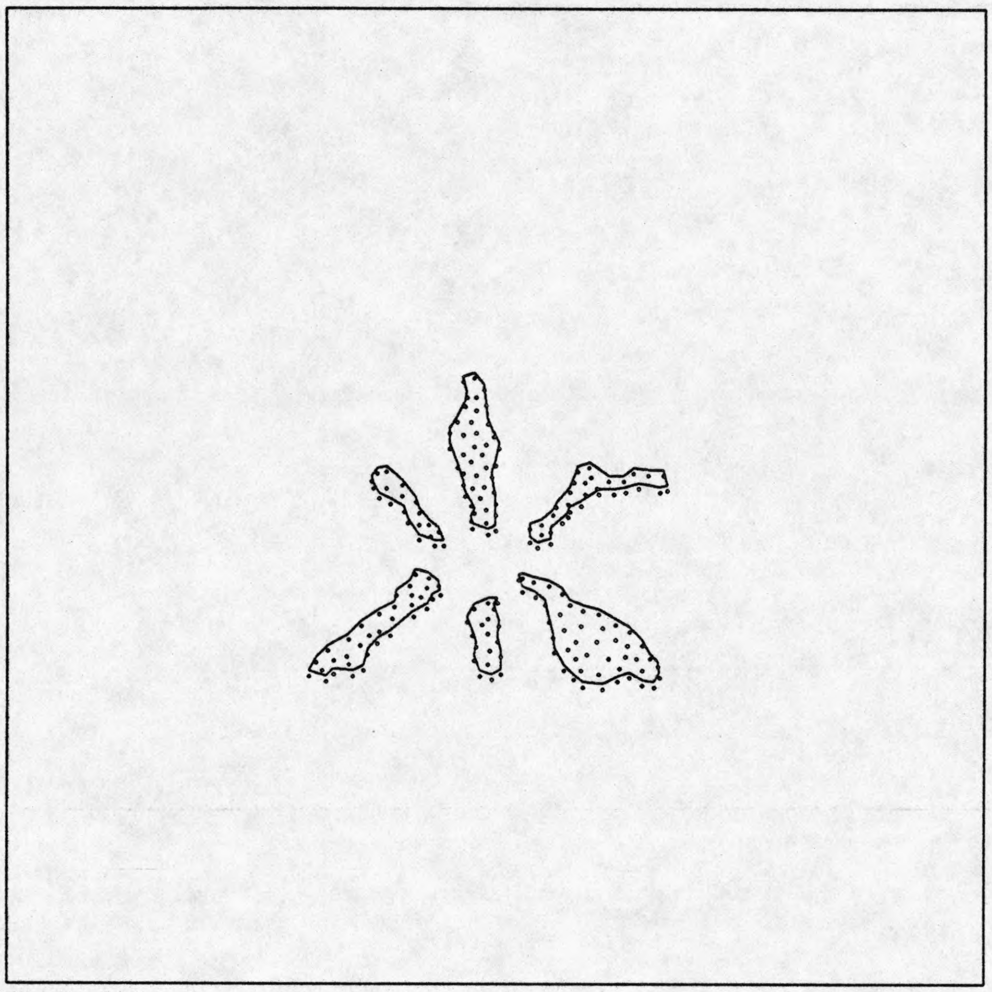


Figure 44(b). The resulting identification.

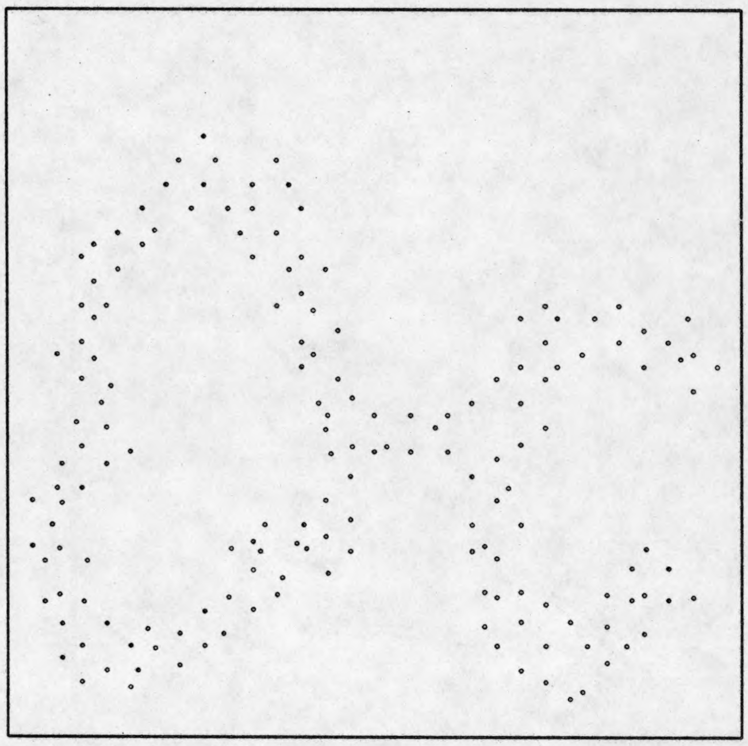


Figure 45(a). An example dot pattern [22].

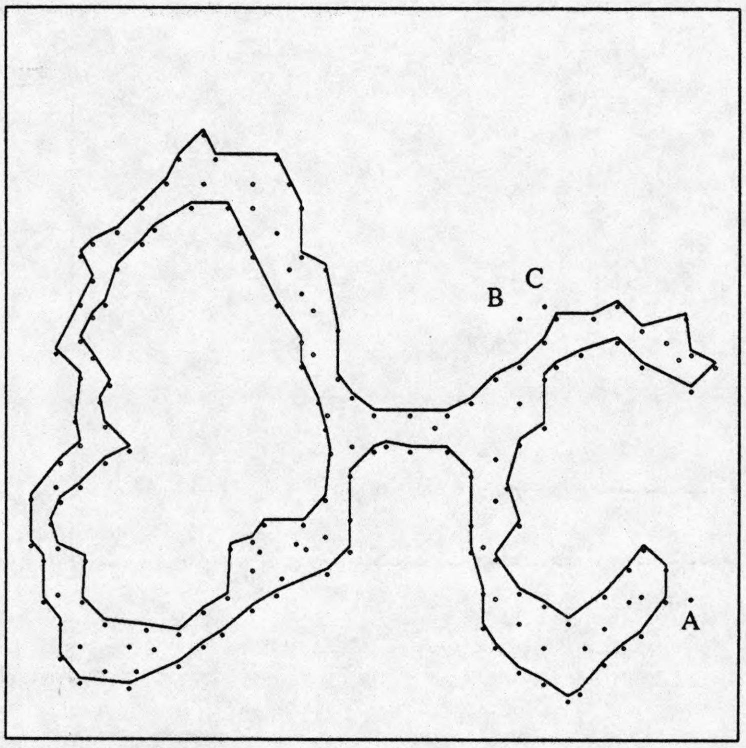


Figure 45(b). The resulting identification.

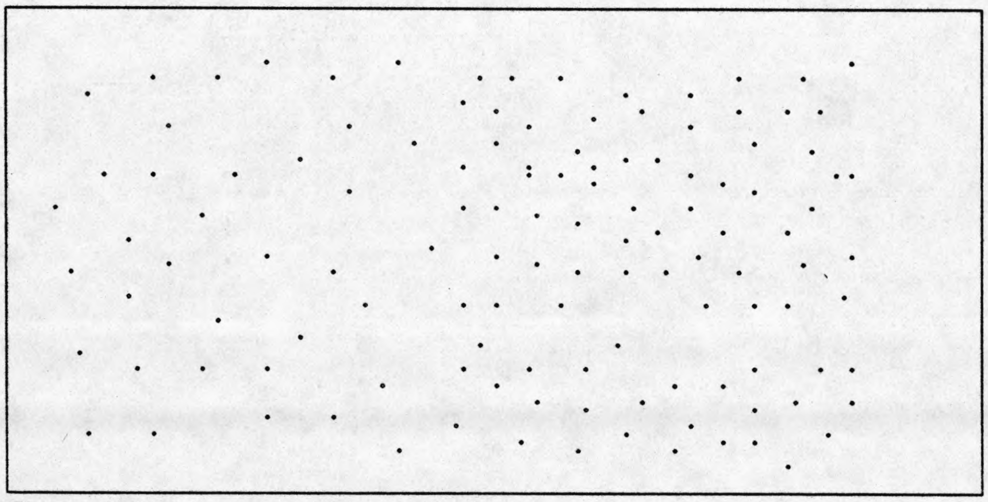


Figure 46(a). An example dot pattern.

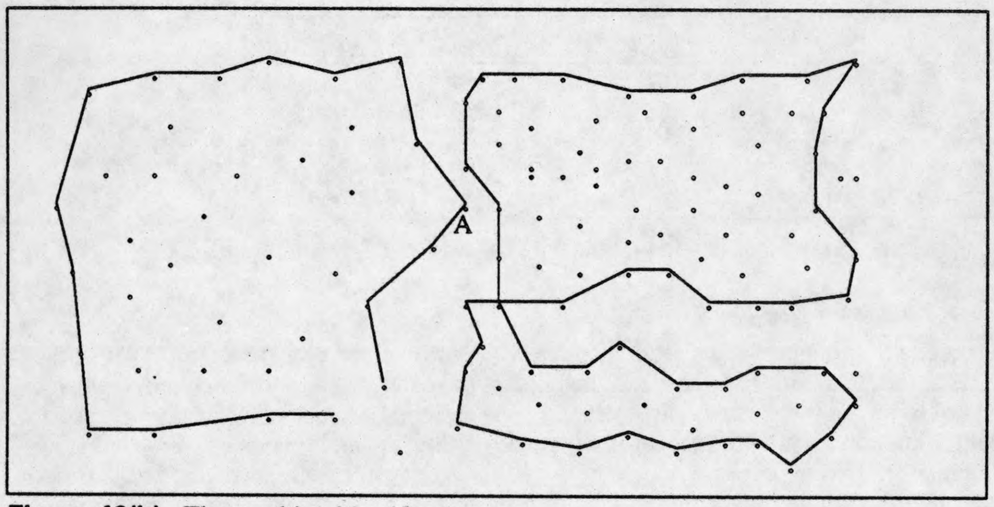


Figure 46(b). The resulting identification.

some of the probabilities converge to their final values very fast, others converge more slowly. The points whose probabilities take a long time to converge are in ambiguous places. In order to converge to the correct values these points must wait for the neighboring dots whose locations are less ambiguous to converge to the correct values.

The dot patterns used were taken mainly from two sources: (a) from previous papers published by other researchers, so that we could compare the results of our algorithm with the results given in those papers, (b) from a hand generated set of dot patterns, to test the performance in detecting specific perceptual structures such as curves,

Iteration	point A		point B		point C	
	INTERIOR	*NON-INTERIOR*	*INTERIOR*	*NON-INTERIOR*	*INTERIOR*	*NON-INTERIOR*
0	0.4288	0.5712	0.4128	0.5872	0.4168	0.5832
10	0.3347	0.6653	0.6733	0.3267	0.2452	0.7548
20	0.2180	0.7820	1.0000	0.0000	0.1241	0.8759
30	0.1313	0.8687	1.0000	0.0000	0.0000	1.0000
40	0.0000	1.0000	1.0000	0.0000	0.0000	1.0000

Table 1. The probability values for the interior identification module are shown for the dots in Figure 25.

Iteration	edge e1		edge e2		edge e3	
	BORDER	*NON-BORDER*	*BORDER*	*NON-BORDER*	*BORDER*	*NON-BORDER*
0	0.8693	0.1307	0.5157	0.4843	0.5140	0.4860
10	1.0000	0.0000	0.6448	0.3552	0.5942	0.4058
20	1.0000	0.0000	0.7659	0.2341	0.7103	0.2897
30	1.0000	0.0000	0.8568	0.1432	0.8083	0.1917
40	1.0000	0.0000	1.0000	0.0000	0.8798	0.1202
50	1.0000	0.0000	1.0000	0.0000	1.0000	0.0000

Table 2. The probability values for the border identification module are shown for the dots in Figure 26.

Iteration	edge e1		edge e2		edge e3	
	CURVE	*NONCURVE*	*CURVE*	*NONCURVE*	*CURVE*	*NONCURVE*
0	0.6421	0.3579	0.5146	0.4854	0.4564	0.5436
10	1.0000	0.0000	0.6453	0.3547	0.1453	0.8547
20	1.0000	0.0000	0.7655	0.2345	0.0000	1.0000
30	1.0000	0.0000	0.8542	0.1458	0.0000	1.0000
40	1.0000	0.0000	1.0000	0.0000	0.0000	1.0000

Table 3. The probability values for the curve identification module are shown for the dots in Figure 27.

borders resulting from smoothness constraints, etc.

Figures 38-44 show the results of grouping which seem to agree with our own perceptual system. That is, the identified interior regions are perceived as interior regions, and the borders have the structure perceived by humans. The results shown in Figures 45-46, on the other hand, have certain labels assigned to dots or Delaunay edges that are either completely incorrect (point A in Figure 45, point A in Figure 46) or that are ambiguous when looked at by humans (points B and C in Figure 45). In most cases in which the dots or Delaunay edges are misclassified, it is not hard to see the structures found by the grouping algorithm in the dot pattern, but it is obviously not the first choice selected by the human visual system. One of the reasons for this is that some of the global considerations taken into account by the human visual system are not adequately addressed by the algorithm. For example, the pattern in the inset in Figure 47, when seen in isolation, has the structure identified by the algorithm which is shown in Figure 48. But when the same structure is put in the context of a highly correlated field of orientations as shown in Figure 47, there are global (texture-like) effects in the human visual system that dominate the processing and the identified structure shown here is pushed into secondary importance.

In Figure 30 we see the curvilinear structure A identified. When we look at the original dot pattern, however, the first impression is a cluster with interior points (structure A) even though with close scrutiny it is not unreasonable to see the curvilinear structure produced by the algorithm. The reason for the way the algorithm performs is the local distribution of the dots in this particular instance. In human perception, there are global effects at work that are not considered by the lowest level grouping algorithm. One such case which is demonstrated in this example is to produce a hierarchical grouping which results in structure A being seen as a cluster being inside the border B and this information in turn is propagated downward and thus the decisions made about the lowest level labelings is effected. The process of information propagation downwards from higher levels is currently lacking in our algorithm. Similar arguments can also be made for incorrect labelings of patterns in Figures 45 and 46.

5.5. Implementation Details of the Algorithm

Our algorithm was implemented in C. The architecture of the program closely resembles the structure given in Figure 24. It is actually implemented as three separate programs corresponding to the steps A, B and C of Figure 24. The details of each step are discussed below. But first, we describe a program which is common to all the programs in our system and it computes the initial representation, i.e. the Voronoi tessellation.

Tile Program

This is a subroutine which computes the Voronoi tessellation of a dot pattern, given the locations of the dots as a sequence of (x,y) coordinates. It uses Green and Sibson's algorithm [13] which works in $O(n^{3/2})$ average time complexity. It is implemented as a subroutine in standard FORTRAN. Its output is a list of neighbors in counter clockwise order for each dot. Once this contiguity list is known for each dot, it is possible to compute all of the statistics and measures discussed in the previous chapter. Of course, since the dot pattern is static we need to compute these measures only once at the beginning and store them.

Label Identifications (step A)

The identification program consists of three major parts for each module in Figure 24: (a) compatibility coefficient computation, (b) initial probability assignment, and (c) relaxation labeling process. In all other relaxation labeling algorithms used in this paper this basic structure holds. Only the compatibility computation and the objects to be labeled change for each case.

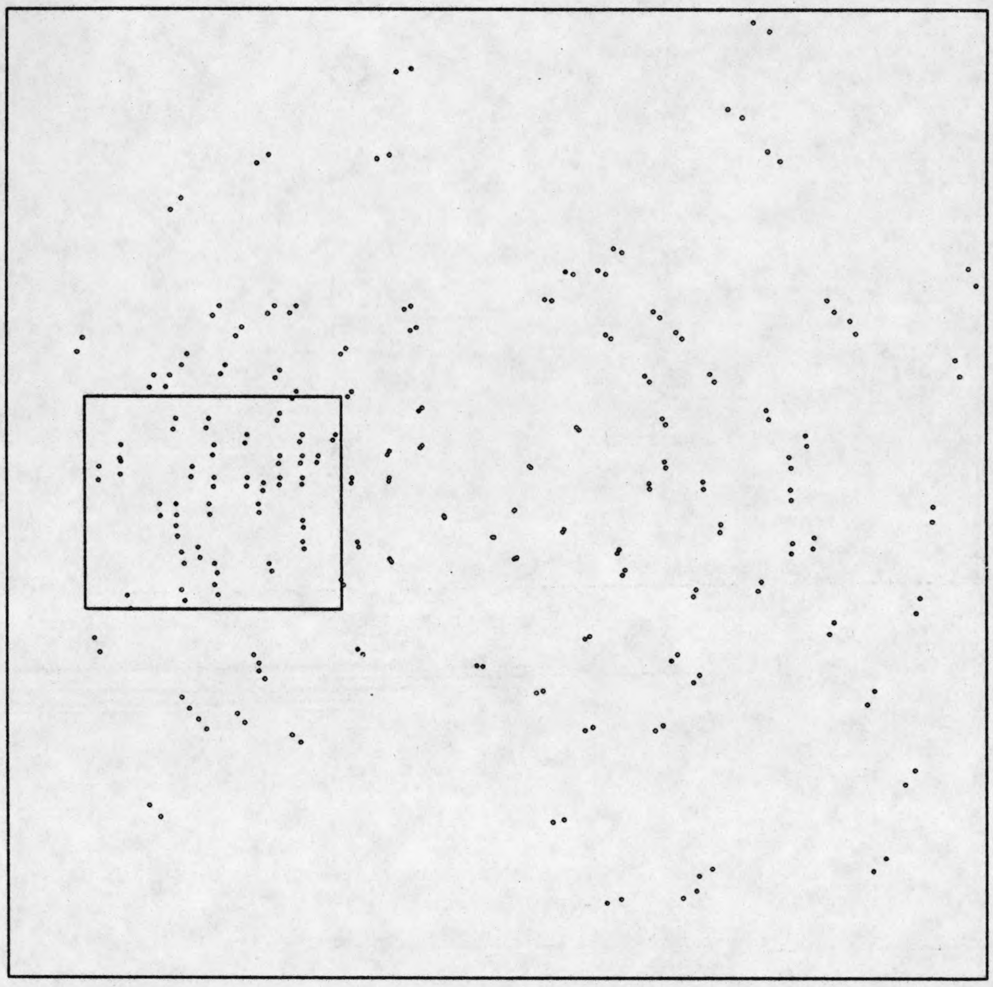


Figure 47. The overall perception of the dot pattern shown is dominated by the global pattern of the orientations of dot pairs. Such global, texture-like analysis is not within the purview of our current algorithm. Local structure, e.g. in the inset rectangle, when isolated from the overall pattern, appears differently. This perceived local structure is extracted by the algorithm as shown in Figure 48.

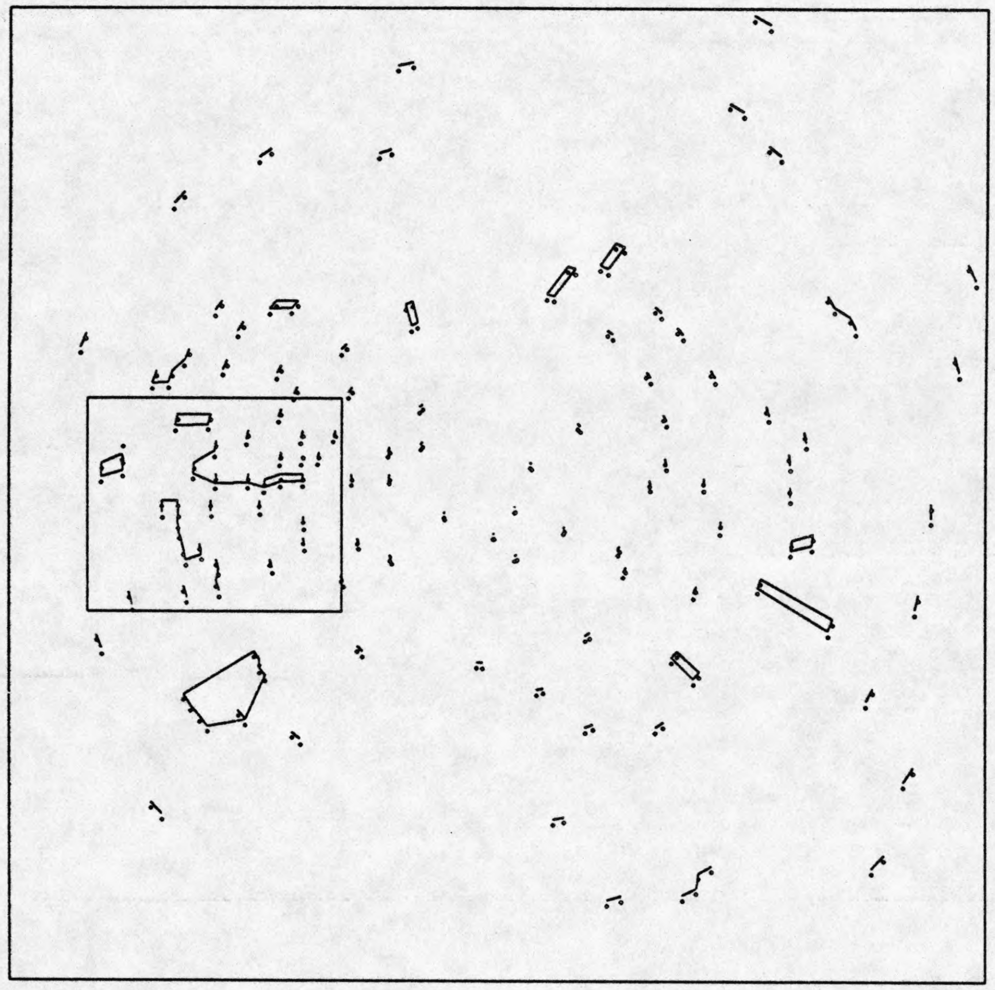


Figure 48. The lowest level groupings found for the pattern shown in Figure 47.

Label Corrections (step B)

The correction process consists also of a relaxation labeling part both for correcting the point labelings and for correcting border labelings. It has, in addition, a preprocessing module. The block diagram and flow control for this step is shown in Figure 49.

Notice that, as we mentioned in Section 5.3.5, the entire preprocessing and correction by relaxation labeling processes are repeated until there are no more labels changed. The module that performs the preprocessing is described below.

Preprocessing

This module performs the very obvious cleaning up of the labels of points and Delaunay edges. The more difficult cases in which it is not obvious whether a label change is needed are left for the interior correction (IC) and border correction (BC) modules which perform a relaxation labeling by looking at a larger context as discussed in Section 5.3.5. This process performs the following *cleanup* operations: (i) A *NONINTERIOR* point which is surrounded by all *INTERIOR* points is converted to interior. (ii) An isolated Delaunay edge which is labeled as *BORDER* is converted to *NONBORDER* if its two endpoints are labeled as interior. This causes the very obvious isolated border edges in interior regions to be corrected. (iii) If there is a dangling single Delaunay edge labeled as

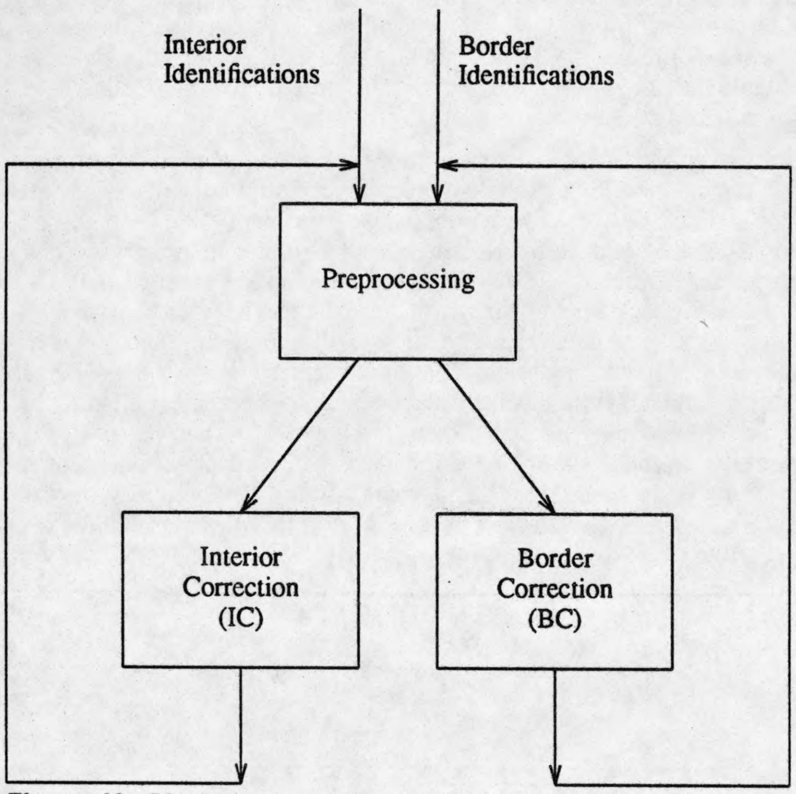


Figure 49. Block diagram and control flow of the label correction process (step B in Figure 24).

border at a junction of degree equal to three in which the other two Delaunay edges belong to border chains consisting of more than one Delaunay edges, then the original single Delaunay edge is deleted.

Combining Interior and Border Labels (step C)

The border and interior correction in step C starts with computation of intersection of the borders resulting from the corrected labels in interior and border identification. This results in the most confident borders to be identified. Then the following steps are performed:
(a) Starting with these border segments, do a connected component analysis and close the gaps between segments.
(b) Look at the configurations in which there is a transition from a region with interior dots to a bar structured border configuration. In such cases continue the borders through the bar borders thus joining the parts with interiors with the parts without interiors.
(c) After all the borders around interior regions are closed, try to merge components that are next to each other.
(d) Finally, a hysteresis smoothing of the borders is done after all the corrections described above. This has the effect of expanding the border outward or grabbing points from the interior if there are sharp corners in the original border that need smoothing. Following are details of the above steps (a)-(d).

(a) Connected component analysis

After computing the intersection of the borders identified by II and BI and corrected by IC and BC, we have a pattern with interior dots which are surrounded by partially identified and incomplete borders. The subroutine performing the connected component analysis starts with these partial borders and interior regions and tries to close the gaps between these border segments.

The subroutine first numbers all such border segments. Then to each interior dot it assigns the labels of all the border segments that are reachable from the interior dot through a sequence of neighbors also labeled as interior (see Section 5.3.6). After each interior point is assigned the labels of all the borders around its region, the subroutine tries to close the gaps between these borders so that at the end of this process all the border segments around an interior region will form a closed curve. The interior region may have more than one closed curves as its border because there may be holes. The labeling of the interior region by the surrounding borders is done so that only the gaps between border segments surrounding a single interior region are attempted for closure. The subroutine does not try to join border segments belonging to two different clusters.

The gap closing process proceeds by examining one of the border segments around an interior region, and considering the border segments surrounding the same region for a possible merge with the original border segment. A pair of borders, B_1 and B_2, are considered for joining 1) if there exists at least one path joining the endpoints of B_1 and B_2 in which all the points are noninterior, 2) both B_1 and B_2 are associated with the same interior region, and 3) the interior region is on the same side of both border segments. If all these three conditions are satisfied and such a pair B_1 and B_2 is found, then comes the task of identifying the path which must close the gap between the two segments. Notice that we said B_1 and B_2 must have at least one path joining them with all noninterior points. There may in fact, be more than one such path. In such a case, all such paths are identified, thus resulting in a list of paths $\{ P_1, P_2, P_3, \ldots \}$. The path selected to join the two segments from this list is the one that connects the B_1 and B_2 in the smoothest possible way (see Figure 50). This smoothness is computed by examining the last two points in B_1, all the points in the path P_i and the first two points in B_2 thus making the final curvature dependent not only on the shape of the path P_i but also on the smoothness of transition from B_1 to P_i and from P_i to B_2. The definition of curvature *curv* of a path through points i=1 to n that we have used here as well as elsewhere in this paper is as follows:

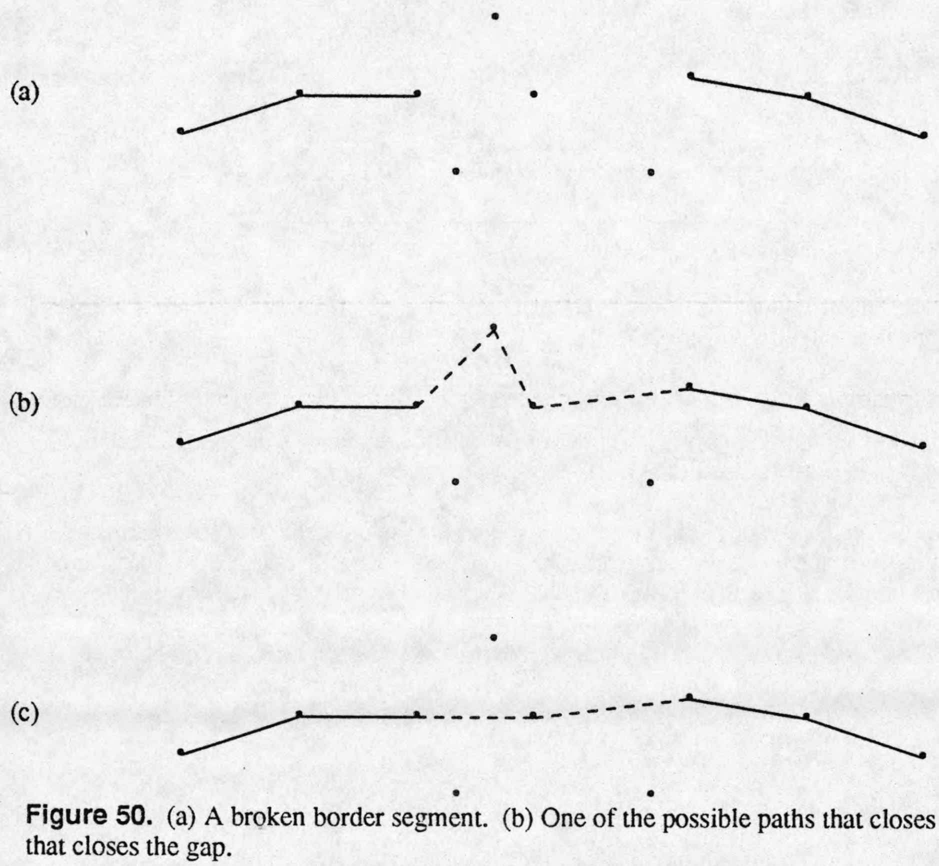


Figure 50. (a) A broken border segment. (b) One of the possible paths that closes the gap. (c) The smoothest path that closes the gap.

$$curv = \frac{1}{n}\sum_{i=1}^{n}(1 - |\cos\alpha_i|)$$

where α_i is the angle between the border Delaunay edges at point i. The smoothest path is selected by picking the path with minimum curvature.

(b) Transition from nonempty to empty interior

This subroutine looks for the configurations shown in Figure 51, and if possible merges the two regions and their borders. A degenerate case of this is the triangular configuration shown in Figure 52. Thus this algorithm handles the range of configurations starting with the simple one of a triangular configuration, going to the more complicated one where the cluster has an extension with no interior (e.g., one of the two clusters with interior in Figure 51 is missing), to the most complex one in which the region with no interior is a transition region connecting two regions with interior.

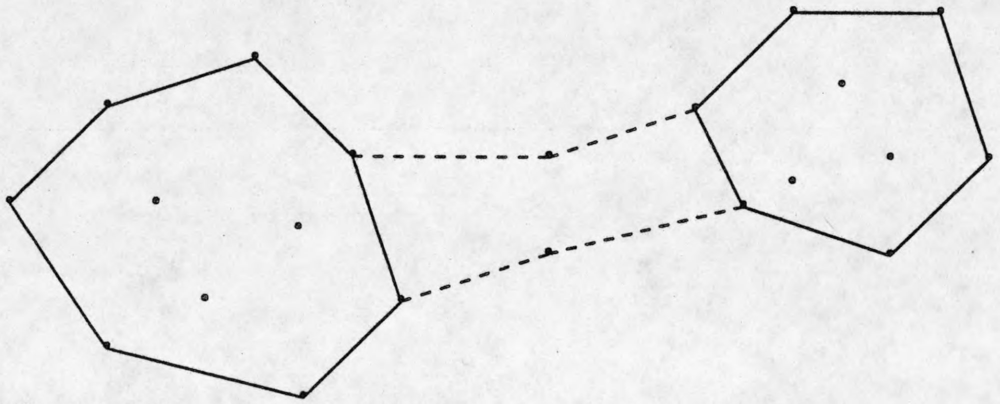


Figure 51. Transition from regions with interior to barlike regions.

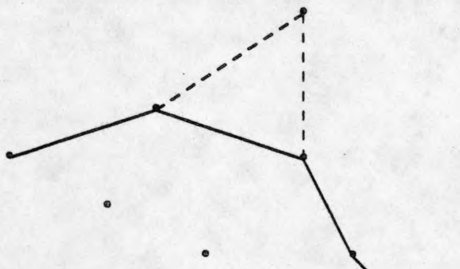
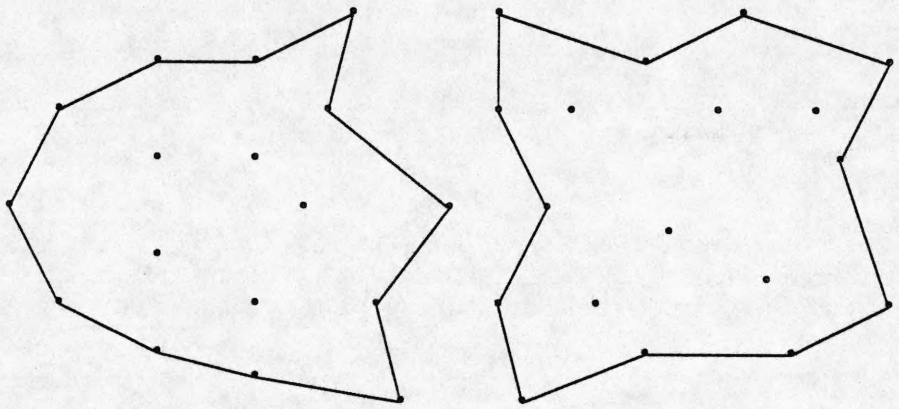


Figure 52. Triangular extension of the border that must be extended.

(c) Component merge analysis

After all the above processing is completed, the dot pattern might still have unnecessary splits in the groupings due to lack of sufficiently global information. When such an erroneous crack is put in a group of dots, it splits the perceptual border of the group such that sharp corners result in the new border and the resulting components are not compact (see Figure 53). Therefore, this algorithm scans the borders of already identified clusters for sharp corners. When a sharp corner is found then the testing and merging process is activated. The first task is to find the cluster with which the possible merge of the current cluster will occur. When looking at a sharp corner there are two possibilities for merge, namely either of the two borders forming the sharp corner may be the crack border which will disappear when the two clusters are merged. In addition, the resulting border after the merger of two clusters must be smooth. Thus, the adjacent components which are candidates for possible merging are found by

(a)



(b)

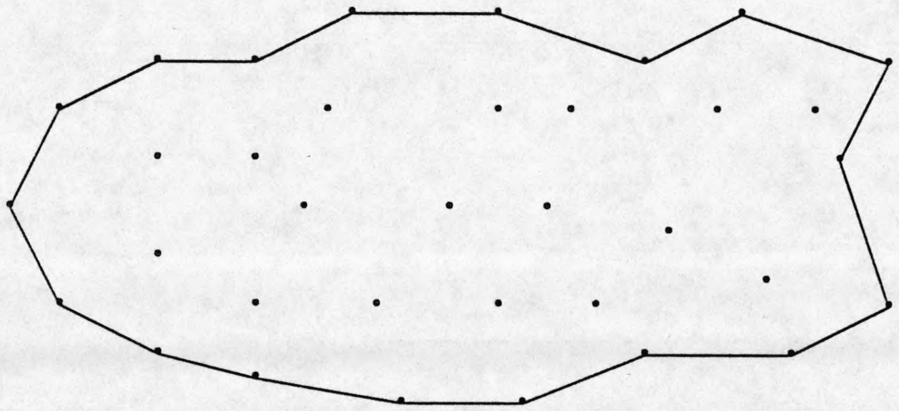


Figure 53. (a) Conditions for possible component merging: dot densities in the two components are comparable; the gap size is comparable to average interdot distance in the components; the border of the merged component is smoother. (b) The result of merging the two components.

scanning the Voronoi neighbors of the corner point within a certain range of angles (so the border will be smooth) for each side of the border. If the neighboring point falls on the border of another cluster and it is a corner point, then this cluster is considered as a candidate for merge. The decision for the merge is based upon four factors, (i) the u_1 and standard deviation σ_1 of the interdot distance within the first cluster, (ii) the mean u_2 and standard deviation σ_2 of the inter dot distance within the second cluster, (iii) the mean interdot distance d_c across the two clusters, and (iv) the smoothness of the new border after merging the two clusters. Specifically, the following criteria are tested:

$$u_2-\sigma_2<u_1<u_2+\sigma_2$$

$$u_1-\sigma_1<u_2<u_1+\sigma_1$$

and

$$u_2-\sigma_2<d_c<u_2+\sigma_2$$

$$u_1-\sigma_1<d_c<u_1+\sigma_1$$

If these two conditions are met, then the borders that form the cracks are converted to interior points and the two clusters are merged.

(d) Border smoothing

There are two cases in which the border can be changed to become smooth. Both cases and the changes applied are illustrated in Figure 54. As seen in these figures, the subroutine looks for sharp corners and tries to see whether the border can go through the gap more smoothly and without altering within component statistics (as

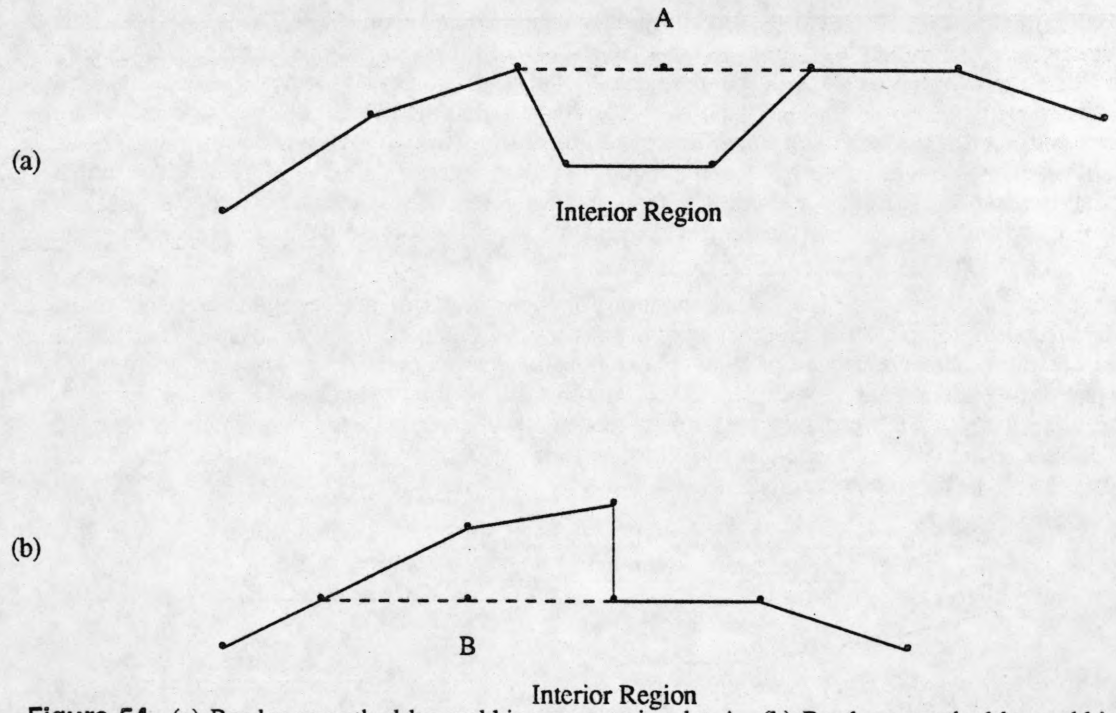


Figure 54. (a) Border smoothed by grabbing an exterior dot A. (b) Border smoothed by grabbing an interior dot B.

would be the case if, for example, the gap were too large). Thus, if the new border improves the border smoothness without much change in the component statistics, the change is made. In all of these cases at most one point is checked for possible smoothing of the border such as the ones shown in Figure 54.

Finally, we summarize the time complexity of our lowest level grouping algorithm.

Time Complexity Analysis

To analyze the time complexity, we can separate the two parts of our algorithm. (i) the computation of the Voronoi tessellation given the dot pattern, and (ii) the computation of the grouping. As we mentioned above the algorithm we have used to obtain the Voronoi tessellation works in $O(n^{3/2})$ average case time complexity where n is the number of dots in the pattern. However, there exist faster algorithms for accomplishing the same task including the optimal one working in $O(n \log n)$ worst case time complexity [18, 27]. Therefore, we assume this first part can be done in $O(n \log n)$ time even though our implementation is not the optimal one.

We can analyze the time complexity of the second part as follows. Most of the time in our algorithm is consumed by the relaxation labeling algorithm. The time complexity of the relaxation labeling algorithm can be analyzed by dividing it into two parts. First, the compatibility coefficients between pairs of dots must be computed. Second, the initial probabilities must be assigned to each dot and the probabilities updated. Since the compatibility of only the neighboring pairs of dots are considered in the Voronoi tessellation, the computation of the compatibility coefficients between a single dot and its neighbors takes a fixed amount of time on average. This is because the number of neighbors of a dot is limited to a small constant [2] in a dot pattern with Poisson distribution (e.g. the average number of neighbors of a dot is 6). This computation of compatibilities has to be carried out for all the dots in the pattern resulting in an $O(n)$ average time complexity for the computation of the compatibility coefficients. Similarly, for the probability updating and initial probability assignments, the interactions between pairs of dots are local, thus making the computations for single dots take a constant amount of time on the average. Once more each of these operations must be carried out for each dot resulting in the average case time complexity of $O(n)$. Therefore, the overall average time complexity of the relaxation labeling process is $O(n)$ a for constant number of iterations.

The preprocessing module of step B and the interior-border combination modules also work in $O(n)$ time because the cleaning and component merging processes are local and, therefore, the resulting processing takes constant time. The gap closing process has a combinatorial part, namely the generation of all the possible paths that can connect two given border segments. But in all the patterns that we have tested the numbers of paths generated are too small to be of consequence for the execution time. There are two reasons for why the number of paths is low: (i) the gaps are usually small (< 4 points in our test patterns), and (ii) the noninterior points through which a smooth path could go are small in number. Therefore, the combinatorial explosion that is possible does not occur. Besides this analysis, however, it is difficult to quantify the number of paths theoretically because the exact properties of the gaps are not known.

There is another part of the program that needs attention for the analysis of time complexity. This is the number of iterations needed to complete the preprocessing correction cycle in step B (Figures 24, 49). Table 4 shows the number of iterations that were needed in order to accomplish the correction. As can be seen, the number for most patterns is one or two, and it is at most five for other patterns. This is the most costly part of the algorithm because for each of these iterations the program performs a relaxation labeling both for interior correction and border correction. However, this label correction is performed not for the entire pattern but only for the selected parts of the pattern labeled by the identification process.

Pattern in figure no.	number of iterations
15	3
38(a)	4
39(a)	3
45(a)	6
46(a)	6

Table 4. The table showing the number of iterations performed by the label correction process (step B in Figure 24) for the indicated patterns.

6. HIERARCHICAL GROUPING

The physical world is mostly organized in a hierarchical manner. This organization is reflected in its spatial structure. For example, a surface's reflectance function is often generated by processes operating at different scales [20]. Thus, a cat's fur is seen with markings at one level and individual hairs at another level. The goal of the hierarchical or multilevel grouping is to extract the hierarchy of spatial structure present in images. The lowest level grouping process described thus far concerns initial perceptual aggregation of the dots. We will now be concerned with the hierarchical grouping of dot segments, or the tokens provided by the lowest level grouping.

As in the lowest level grouping, hierarchical grouping must also capture properties that are unlikely to arise by accident. A salient feature of such grouping of segments is that it is based on the sizes, shapes, orientations and termination points of segments, in addition to their positions. Thus, alignment and parallelism are important relations among different tokens. Similarly, the occurrence of termination points of the tokens along a smooth curve is perceptually important. See Figure 55 for an example of a dot pattern with hierarchical structure.

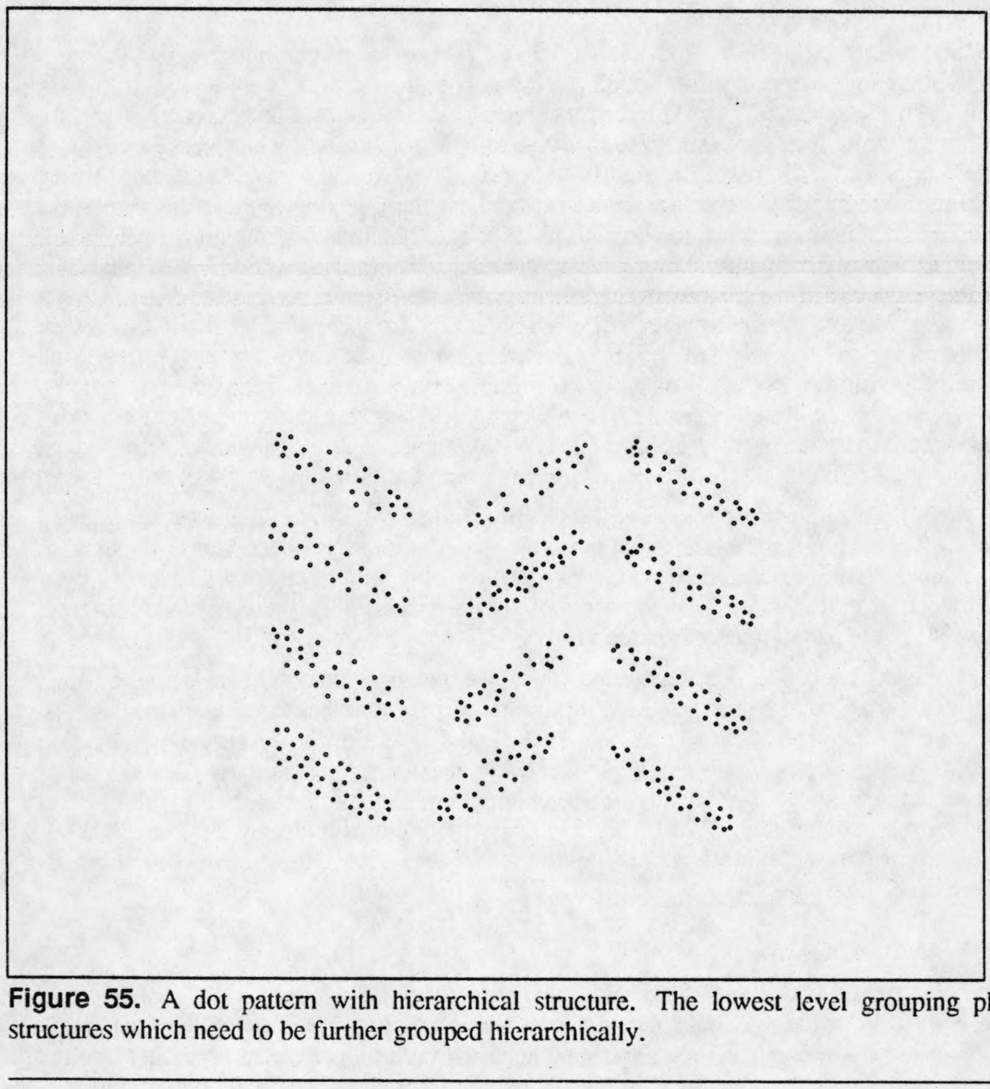


Figure 55. A dot pattern with hierarchical structure. The lowest level grouping phase extracts the bar-like structures which need to be further grouped hierarchically.

To define an approach to hierarchical grouping, we must answer the following questions. What constitutes the geometric structure of these more complex tokens defined by the lowest level segments? What are the properties of the tokens pertinent to grouping? How do these properties determine the grouping? How is the hierarchical organization represented?

Consider the question about geometric structure, which was central to the task of the lowest level grouping. Because the segments may have arbitrary shapes, the neighborhood of a segment, unlike a dot, is neither apparent nor seemingly useful. The distance between adjacent segments is the only aspect of the geometric structure that appears to matter beyond the intrinsic geometric properties of the segments. Whether two segments group together or not depends on their relative positions and similarity of their intrinsic properties. To define an approach to grouping, we must not only specify the role of each property in grouping, but also how they interact and jointly determine the overall perceived structure. We saw an example of this in Figures 10-11 of Section 2 which demonstrates cooperation and competition between the properties of proximity and similarity.

Let us first consider the role of positions of tokens on their grouping when tokens are close together. As in the lowest level grouping, the Gestalt rules dealing with positions of the tokens are still valid, i.e., those tokens that are close together group, and tokens whose locations define a smooth curve still have significance. Thus, grouping may occur because the segments fill a blob-like region and they have similar orientations, or because they lie along a curve with mutually aligned or parallel orientations, or because their dense packing forms a perceptual cluster, or because their terminators form a perceptual cluster, etc.

When the adjacent segments start receding apart from each other, the roles of their relationships based on properties such as terminations and orientations start to diminish. Grouping becomes increasingly a function of merely the positions of the tokens. This is to be expected since when two segments are far apart relative to their sizes, the distinction between the locations of the terminators and the centroid of one segment blurs when viewed from the other segment. The termination points usually lose their significance completely in such cases, and they can be replaced by only the positional information, or the location of the centroid. If there were any grouping among the terminators (e.g., if they formed a smooth curve), presumably, it would also be evident among the positional tokens. We see a similar deterioration in sensitivity to orientation-based relations among the tokens. As the distance between tokens increases, the importance of parallelism between tokens decreases. This is partly due to the fact that when the sizes of tokens are small relative to the intertoken distances, then all the lines connecting any two points of the tokens are roughly parallel. However, the effect of orientation relationship does not disappear entirely with increasing distance; orientation information loses significance for receding tokens but not as fast as the terminators. (If there is a large number of tokens, either aligned or parallel, then we can still perceive grouping of these tokens. Such cases, however, are related more to texture perception than the type of detailed, less global analysis that we are concerned with in this paper. These cases require a texture-like treatment which is beyond the scope of this work.)

The significant role that the intertoken distance plays suggests a partitioning of tokens into groups, each group consisting of tokens that are nearby. Figure 59 shows two such proximity based groupings of segments. Criteria for further grouping of proximal tokens include orientation, curvilinearity, termination points, and finally, size: if a cluster can be split into subclusters each of which consists of segments of similar sizes, and if these sizes are different for different subclusters then a split in the parent cluster is perceived.

These various processes capture a variety of structure defined by the segments at a range of scales. The actual perceived organization of segments is an amalgamation of the different organizations based on different criteria. For example, consider the set of segments shown in Figure 56 which would be grouped together by orientation based grouping. This grouping, however, contains subgroupings. Each subgrouping in this case consists of a set of segments that are parallel and lie along a curve. Specifically, the segments may lie along a curve and their orientations may be aligned along the curve. Alternately, a set of segments may define a grouping if they have similar sizes (Figure 57), or their terminations define a smooth curve (Figure 58). The subgroupings contained in a grouping correspond to microstructure in the grouping.

6.1. A Representation for Hierarchical Grouping

Starting with the dots, the number of available candidate segments that may group monotonically increases as more and more groupings are defined from the previous ones. In any given attempt at grouping only one among a set of hierarchically related (geometrically embedded) segments may be considered, thus restricting to only one the number of segments grouped from any given location in the pattern. In other words, no spatially overlapping segments are considered for grouping. This enforces the condition that a given set of dots in the pattern participates in the overall grouping only in one capacity at any given time, i.e., it fulfills only one perceptual role at one time.

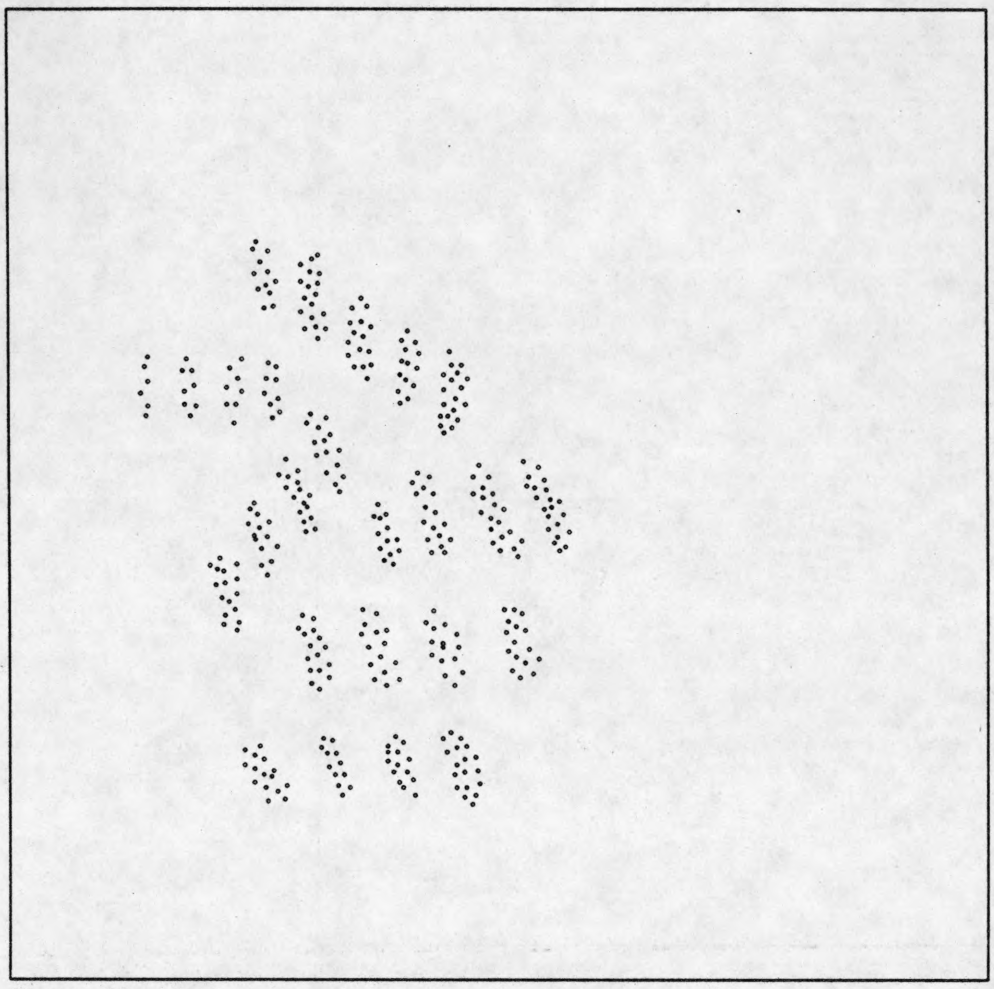


Figure 56. All tokens have similar orientations but we see curvilinear structures as subgroupings.

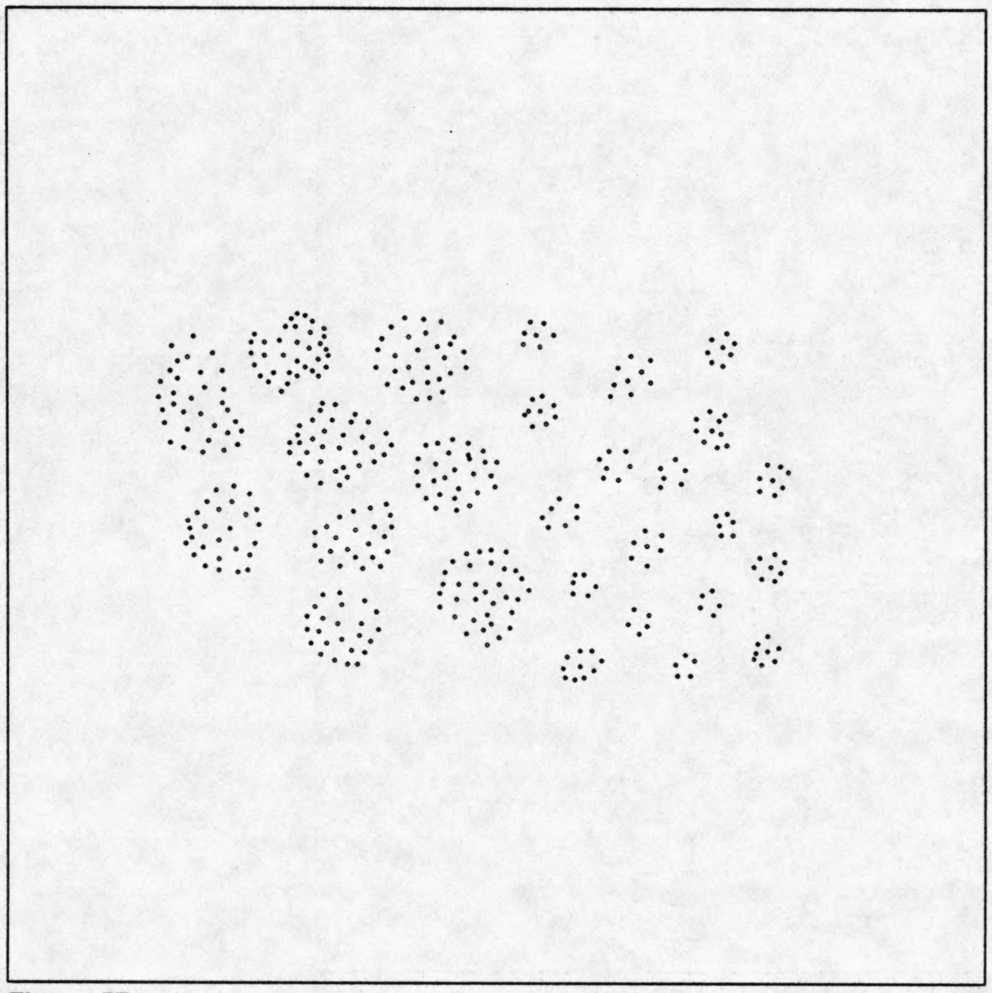


Figure 57. All tokens are close to each other but we see two subgroupings based upon the sizes of tokens in the two halves.

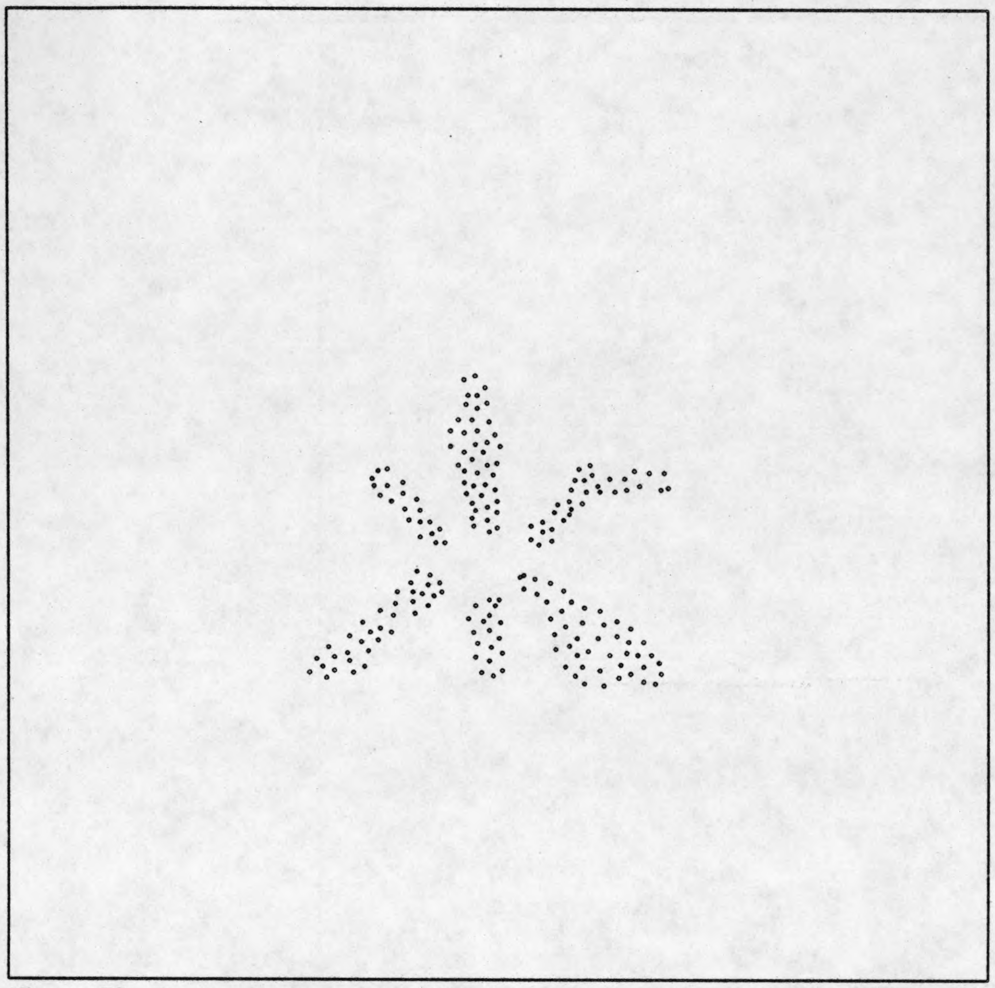


Figure 58. The termination points of the elongated tokens form a closed curve.

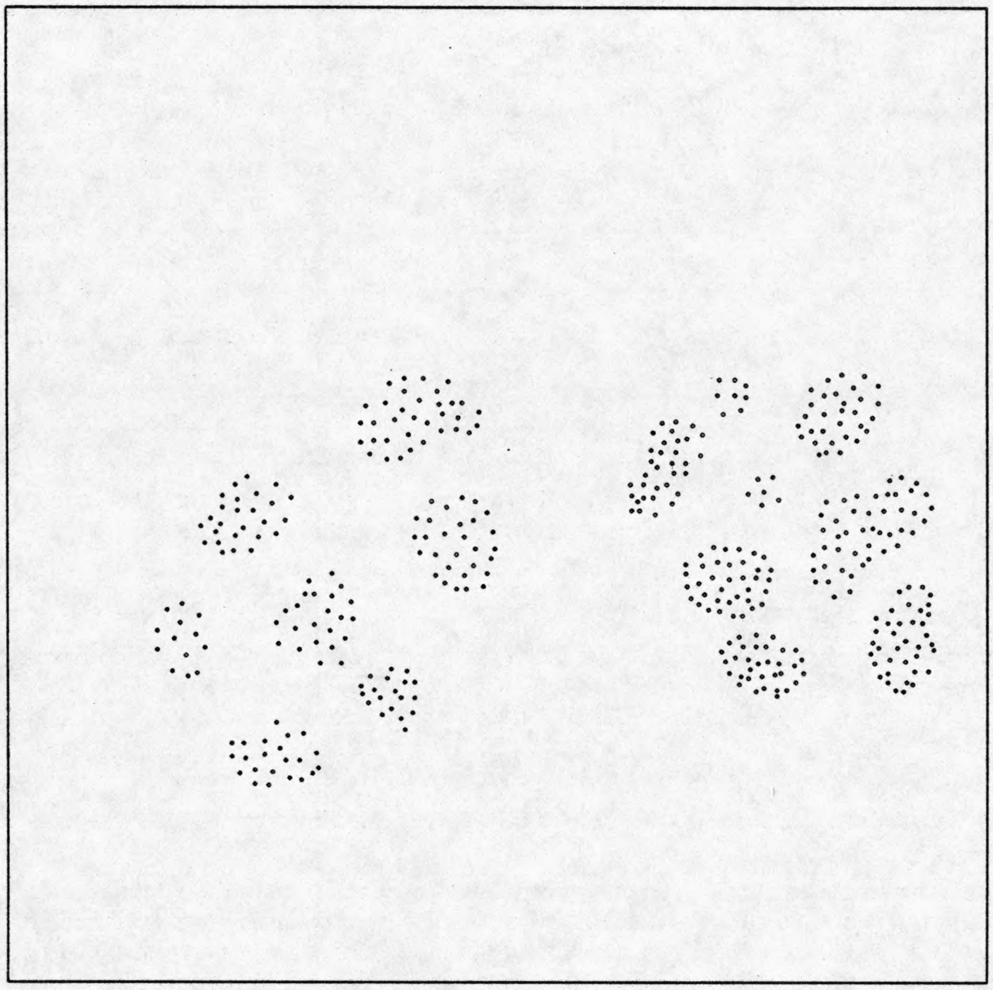


Figure 59. We see two clusters of tokens within each of which the tokens are close together.

We may represent the recursive grouping of tokens into larger tokens by a tree-like structure. Thus, if each dot is represented by a leaf node, then all the dots belonging to a single grouping have a common parent node that corresponds to the parent grouping. The different lowest level groupings correspond to different parent nodes at the level immediately above the leaf nodes. Of course, the numbers of dots in these groupings may be widely different. As groupings are recursively created, the tree becomes taller. The newly generated nodes serve as additional candidates for grouping with *any* other set of existing nodes irrespective of their levels, except for the constraint that only one node along a tree path may be considered for grouping at one time. If a segment can be part of multiple groupings, then the corresponding node has multiple parents, thus making the representation not strictly a tree, but a graph. If a segment does not group with any other, it enters the list of top level groupings, and its hierarchical substructure is given by its corresponding tree or graph. The overall perceptual organization is given by a forest of trees or graphs pointed to by the entries in the top level list.

On the set of segments generated by the lowest level grouping process, hierarchical grouping may be performed by the various grouping processes mentioned earlier working concurrently. Each process examines the available set of distinct segments (i.e., excluding all but one among each set of segments related by ancestor-descendent relationship), and identifies all groupings. For each grouping identified, it adds to the pool of the segments a new segment and records the corresponding ancestor-descendent relationship in the representation graph. When none of the processes can produce further groupings, the graph represents the derived perceptual organization.

The physical size of the segments increases while going up the graph, i.e., from children to parents in the graph. Thus, the nodes higher in the graph correspond to groupings perceived at lower observer resolution. Moving down from parent nodes to children corresponds to an increased observer resolution in a given region, or focusing of attention. Moving up and down the representation graph corresponds to changing the focus of attention.

We will now present details of our implementation of hierarchical grouping in dot patterns. Figure 60 shows a schematic of the various modules and control flow. All the tokens are subjected to proximity, orientation, and size based grouping. The curve based and terminator based grouping, however, are only applied to the nearby tokens, i.e. to the tokens which are identified as belonging to the same cluster by the proximity based grouping module. All the grouping processes are concurrent; they are constantly replacing groups of tokens by new tokens larger in size. The newly produced tokens join the collection of available tokens for grouping and are considered for further grouping. Of course, no two tokens are considered for grouping if one token is contained in the other.

Before we describe the individual grouping processes, in the next two sections we discuss a representation of tokens that helps assign centroid, terminators, size and orientation to a token, and defines the neighbor relationship between tokens.

6.2. Token Representation

One way to represent the shape of a token is by fitting an ellipse that best covers the token. The various desired properties of a token are measured through the corresponding properties of the ellipse:

Size – The lengths of the major and minor axes determine how large a token is.

Position – The center of the ellipse represents the position of the token.

Orientation – If the major and minor axes of the ellipse are not of the same length, then the ellipse has a preferred orientation in the direction of the major axis. The degree of elongation can be computed from the relative lengths of the major and minor axes, and it serves as a measure of significance to be attached to the orientation information. Thus, the direction of orientation of a very round ellipse (approaching a circle) is given less significance than a very elongated ellipse (approaching a line segment).

Terminations – For an elongated ellipse, the extremities along the direction of the major axis give the positions of terminations.

Some examples of dot clusters and their representations by ellipses are shown in Figure 61. While a single ellipse may adequately represent the shape of a compact cluster, the best fitting ellipse to a nonconvex cluster may have a shape greatly different than the cluster (Figure 62(a)). Therefore, the representation must be modified to handle such cases.

Our solution to this problem is to have a hierarchy of ellipses. Whenever the best fitting ellipse is not a tight fit to the cluster, we use the minor axis of the ellipse to divide the cluster into two halves, and compute the ellipse representations for each half of the cluster separately. This subdivision of the ellipses yields smaller ellipses that provide better fits to subsets of tokens. The subdivision continues until all ellipses represent tight fits (Figure 62(b)-(c)). These final ellipses have locations, orientations and dimensions as required to capture the shape details, such as orientation changes in a curve. The various properties such as orientation, terminators, centroids, etc. are computed from these ellipses.

Since the ellipses are desired to approximate the shape of a token, we eliminate all the interior subtokens and fit the ellipse to the border subtokens. The fitting is done using the first and the second order moments of the mass distribution of the border subtokens [14] where each subtoken is assumed to have a unit mass. For a lowest level token containing n dot subtokens at border locations (x_i, y_i), the moment of order (p, q) is given by:

$$m_{pq} = \sum_{i=1}^{n} x_i^p y_i^q$$

The centroid $(\overline{x}, \overline{y})$ of the token is computed from the formulas $\overline{x} = m_{10}/m_{00}$ and $\overline{y} = m_{01}/m_{00}$.

The best fitting ellipse of a segment is computed such that its second order moments of area are equal to those of the segment border. The appropriate parameters a, b, and θ of the ellipse (Figure 63) are obtained by equating the measured second order moment invariants μ_{pq} of the segment with the corresponding expressions for an ellipse (see Section 4 for moment invariants). The following formulas give the parameters of the ellipse [14].

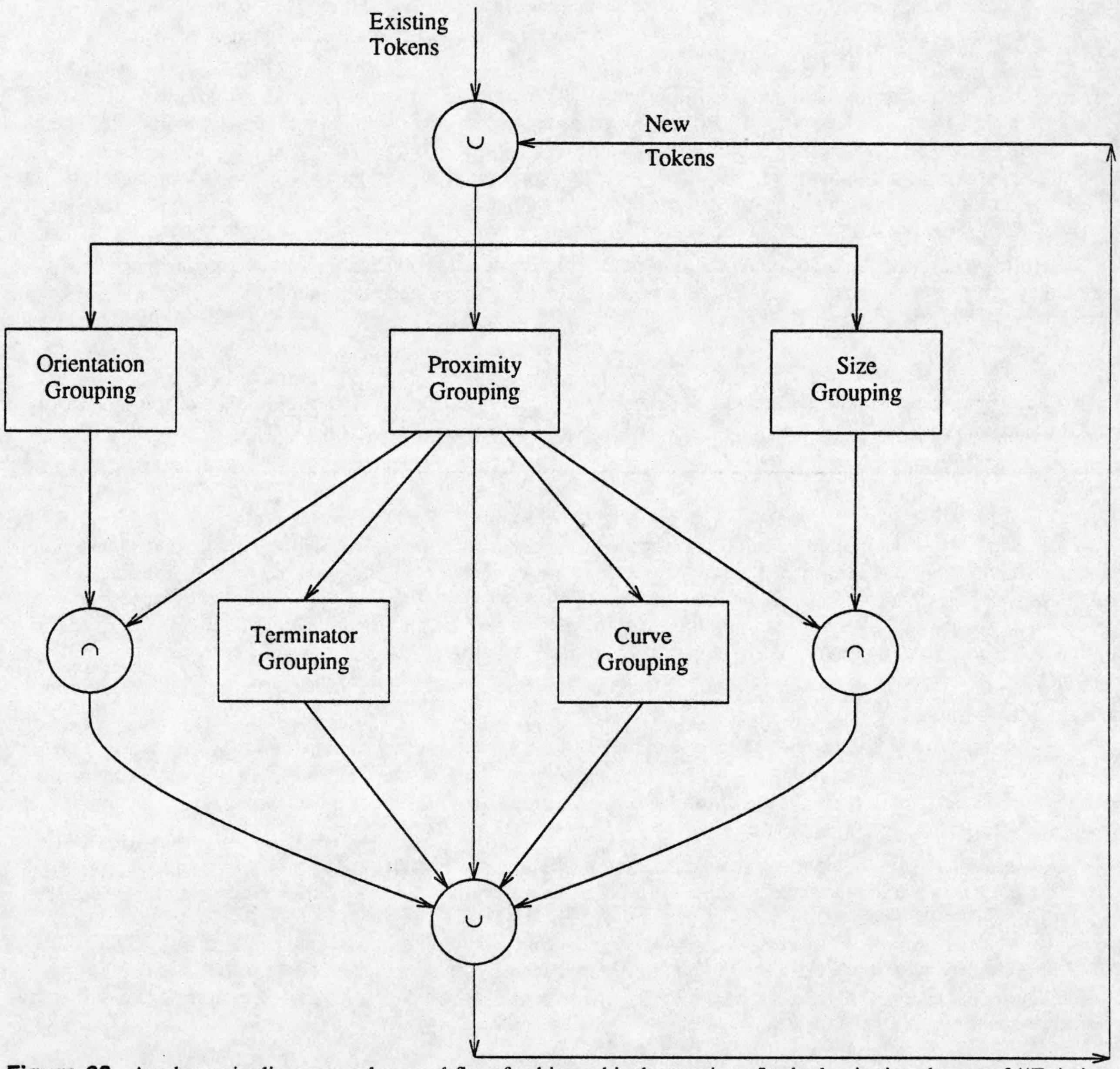


Figure 60. A schematic diagram and control flow for hierarchical grouping. In the beginning the set of "Existing tokens" contains the tokens provided by the lowest level grouping process. Subsequently, this set grows by adding new, larger tokens generated by grouping among existing tokens.

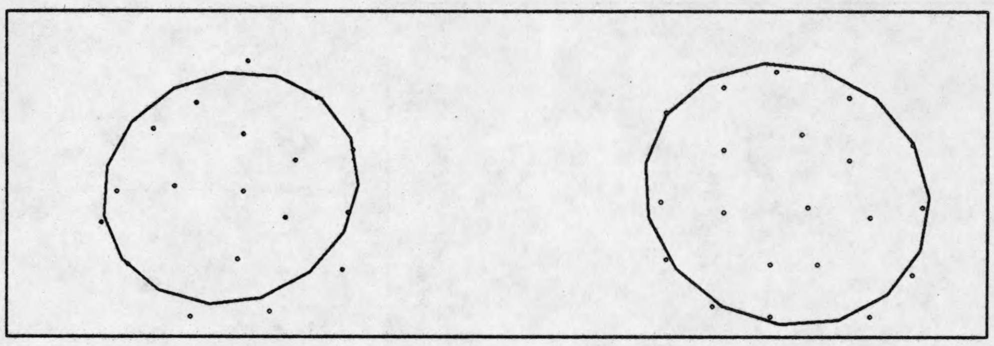


Figure 61. Representation of lowest level groupings by best-fitting ellipses. In this example the ellipses result in a tight fit to the clusters.

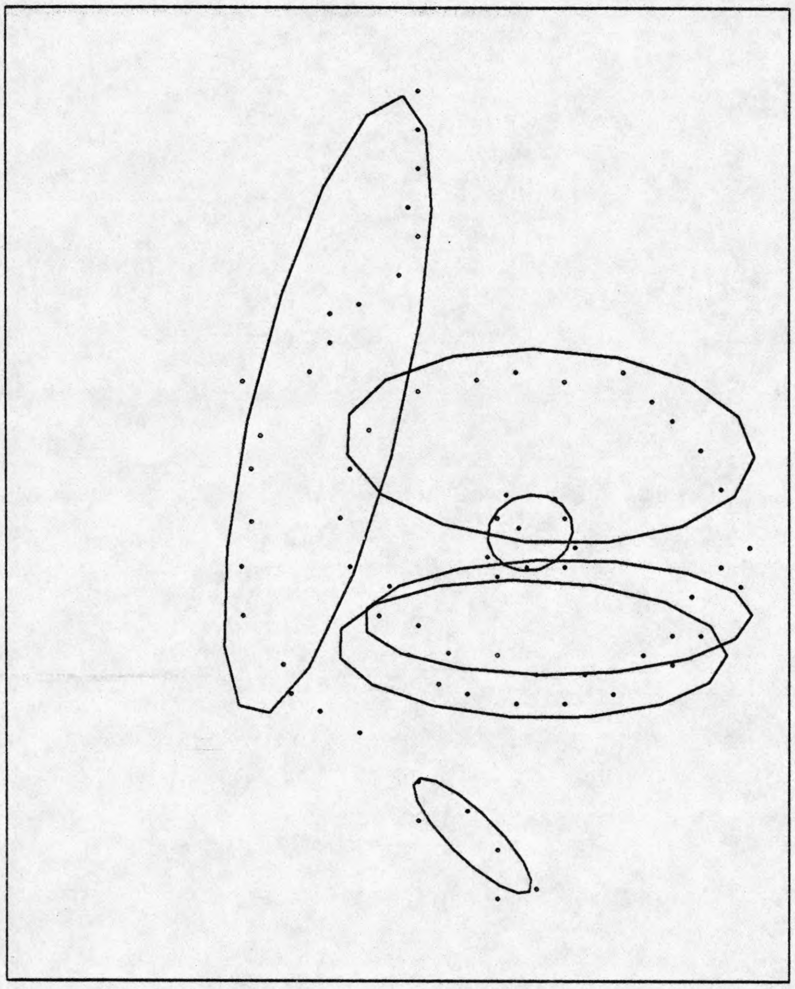


Figure 62(a). Best-fitting ellipses here do not provide a tight fit to the lowest level groupings. Therefore, these ellipses do not constitute a good representation of the lowest level tokens.

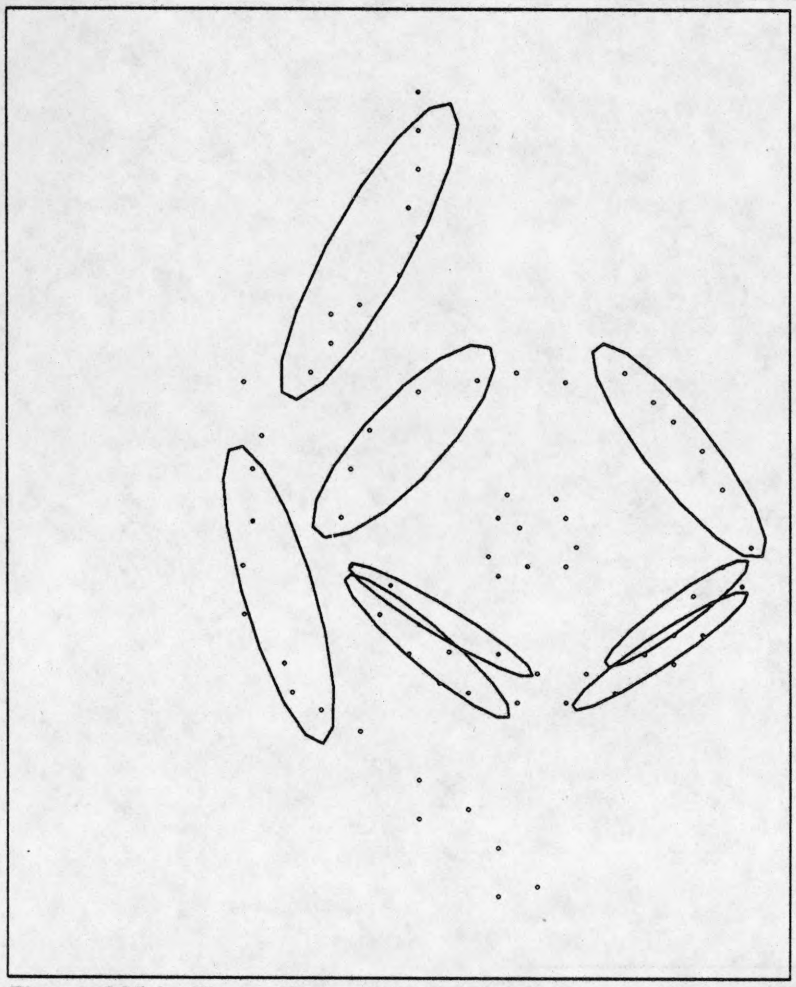


Figure 62(b). Those ellipses in Figure 62(a) that are poor fits to dot clusters are decomposed to yield smaller ellipses. The ellipses resulting after decomposition are better fits than the ones in Figure 62(a), although some ellipses still are inadequate and need to be further decomposed.

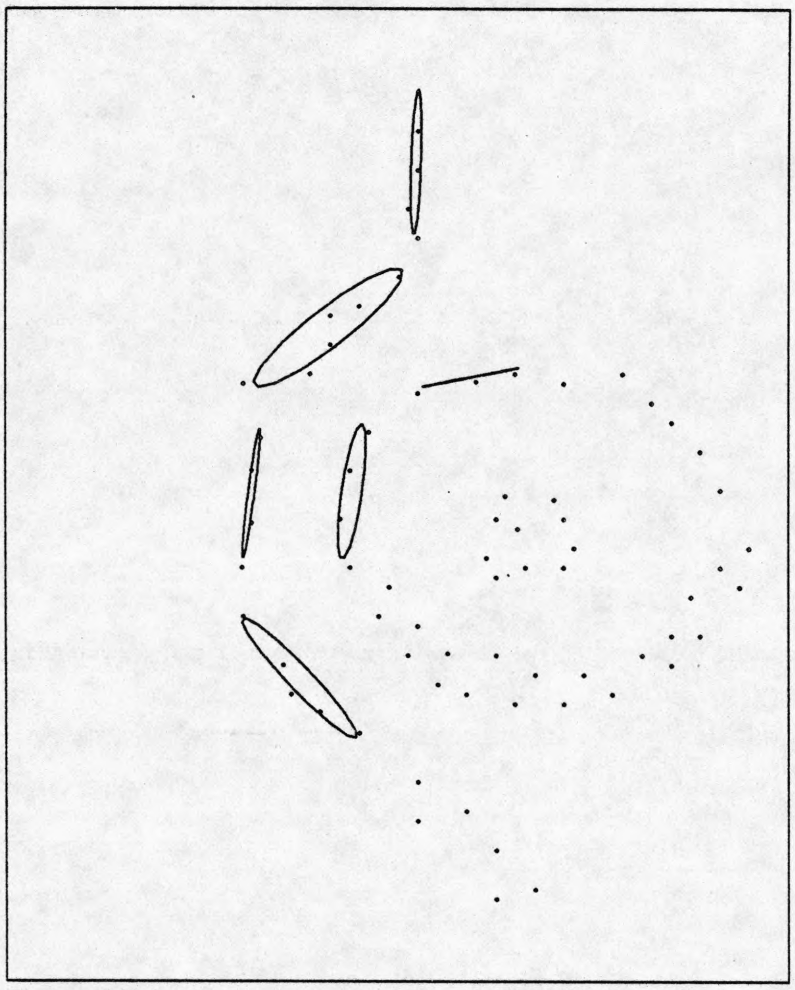


Figure 62(c). One more step of decomposition yields ellipses that provide acceptable fits. The tokens defined by the acceptable ellipses in 62(a), 62(b) and 62(c) together capture the structure of the lowest level groupings in a form suitable for detecting hierarchical structure.

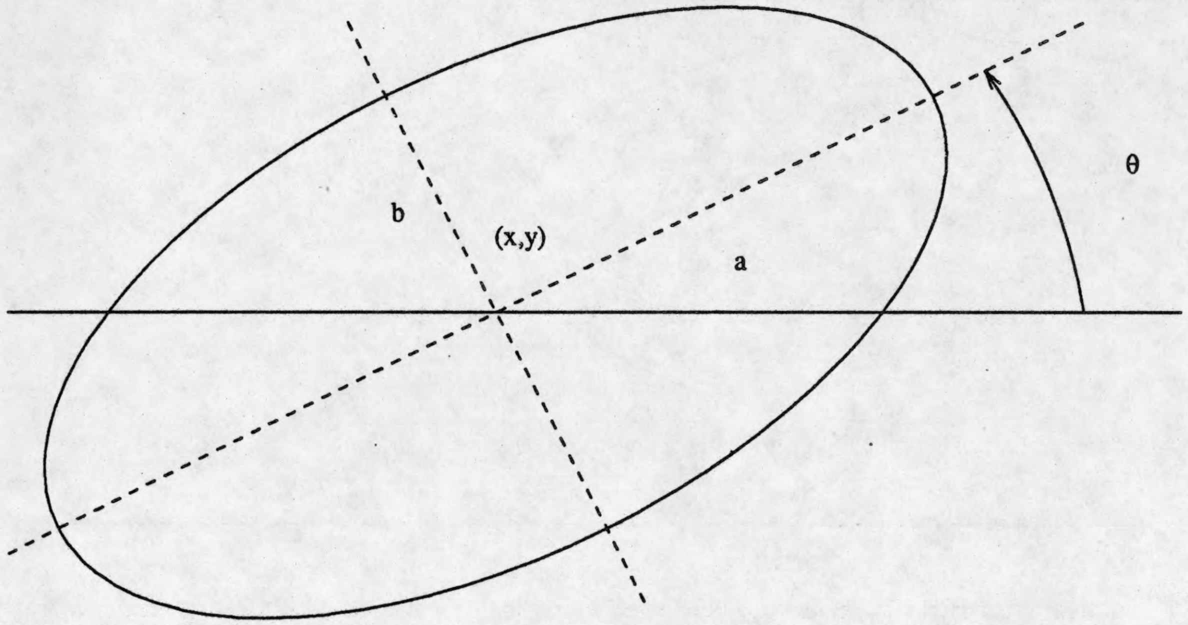


Figure 63. The parameters of an ellipse are the center (x,y), the major and minor axes a and b, and the orientation θ.

$$a=\left[\frac{\mu_{20}+\mu_{02}+\left[(\mu_{20}-\mu_{02})^2+4\mu_{11}^2\right]^{\frac{1}{2}}}{\mu_{00}/2}\right]^{\frac{1}{2}}$$

$$b=\left[\frac{\mu_{20}+\mu_{02}-\left[(\mu_{20}-\mu_{02})^2+4\mu_{11}^2\right]^{\frac{1}{2}}}{\mu_{00}/2}\right]^{\frac{1}{2}}$$

$$\theta=\frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right)$$

A normalization of the ellipse parameters is performed so that the length of the ellipse along its major axis is the same as the length of the segment.

A measure of goodness of fit of the ellipse is computed from the total distance between the borders of the token and the ellipse as shown in Figure 64. Thus,

$$fit=|u|+\sigma$$

In this equation u is the average and σ is the standard deviation of the distances of border subtokens from the ellipse (Figure 64). If this measure of the fit is below a certain threshold, T_{fit}, then the cluster is divided into two subsets, as mentioned above, along the direction θ, and a smaller ellipse is computed for each of the two subsets.

6.3. Definition of Adjacency Between Tokens

The tokens that result after grouping of dots possess spatial extents. Therefore, the adjacency relationship between such tokens is more complicated than for dots. We define an adjacency graph as a graph whose nodes are the tokens, and whose edges connect those tokens identified as adjacent. To obtain the adjacency graph, we start with the Delaunay graph defined by the original set of dots. There exists an edge of the adjacency graph between

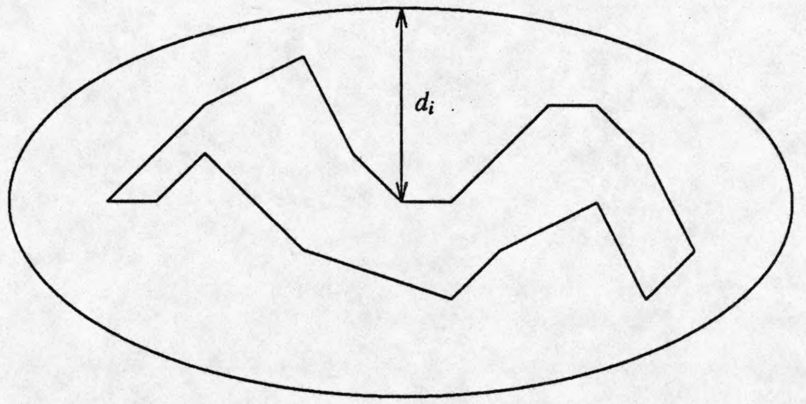


Figure 64. Goodness of fit of an ellipse to a token is based on the distances, d_i, of the ellipse to the subtokens along cluster border.

two tokens T_1 and T_2 if there exist two points $p_1 \in T_1$ and $p_2 \in T_2$ such that p_1 and p_2 have an edge between them in the original Delaunay graph, and T_1 and T_2 are not related with an ancestor-descendant relation. This definition results in an adequate representation for performing the hierarchical grouping. The hierarchical grouping processes to be described below work by labeling the edges of the adjacency graph.

6.4. Proximity Based Grouping

The proximity based grouping of tokens is based on their relative locations. However, unlike dots, the definition of the distance between two adjacent tokens is not obvious. For example, we cannot use the separation of the center points of the tokens because the tokens' spatial extents are not necessarily the same in all directions and for all sizes. We compute the distance $dist_{12}$ between two tokens T_1 and T_2 as follows:

$$dist_{12} = \frac{tot\_dist}{(a_1 + a_2)}$$

where the *tot_dist* is the distance between the centroids of the two tokens T_1 and T_2, and a_1 and a_2 are the major-axis lengths of their corresponding ellipses.

The proximity based grouping module uses a probabilistic relaxation labeling process which labels the edges in the adjacency graph as either *EXIST* (or, equivalently, *NOT−BREAK*) or *NOT−EXIST* (or, equivalently, *BREAK*). An edge between two tokens is labeled as *EXIST* if the two tokens are close to each other relative to their distances to other adjacent tokens. Otherwise, the edge between the two tokens is labeled as *NON−EXIST*. The complete expressions for the computation of the compatibility coefficients are given in Appendix C. The proximity based grouping for an example dot pattern is shown in Figure 65. In Figure 65 and the figures to follow, the hierarchical grouping of tokens is shown by drawing that subset of the original Delaunay edges around each set of grouped tokens, that surrounds all the grouped dots. This is the Delaunay border of the grouped dots, and it tightly encloses the grouped tokens.

6.5. Orientation Based Grouping

In orientation based grouping the two properties which are important are (i) parallelism and (ii) alignment of elongated tokens. The orientation based grouping module uses a probabilistic relaxation labeling process which labels the edges in the adjacency graph as either *EXIST* or *NOT−EXIST*. An edge between two tokens is labeled as *EXIST* if the two tokens are aligned or parallel. If there is a large difference between the orientations, the edge between the two tokens is broken indicating that they do not group together. The compatibility coefficients are given in Appendix C. The result on an example is shown in Figure 66.

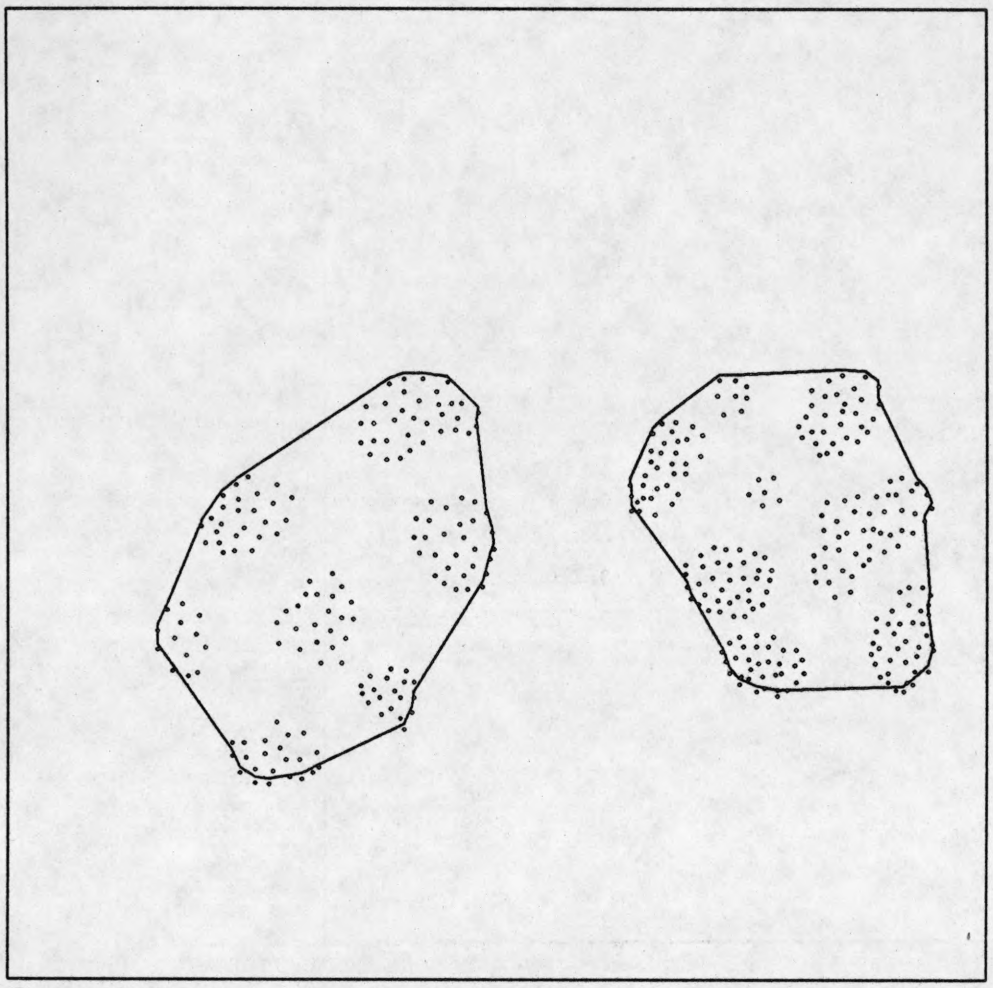


Figure 65. The result of proximity grouping of the lowest level tokens for the pattern in Figure 59. The groupings are shown by drawing the outermost Delaunay edges in each grouping that form a closed curve around the grouped dots.

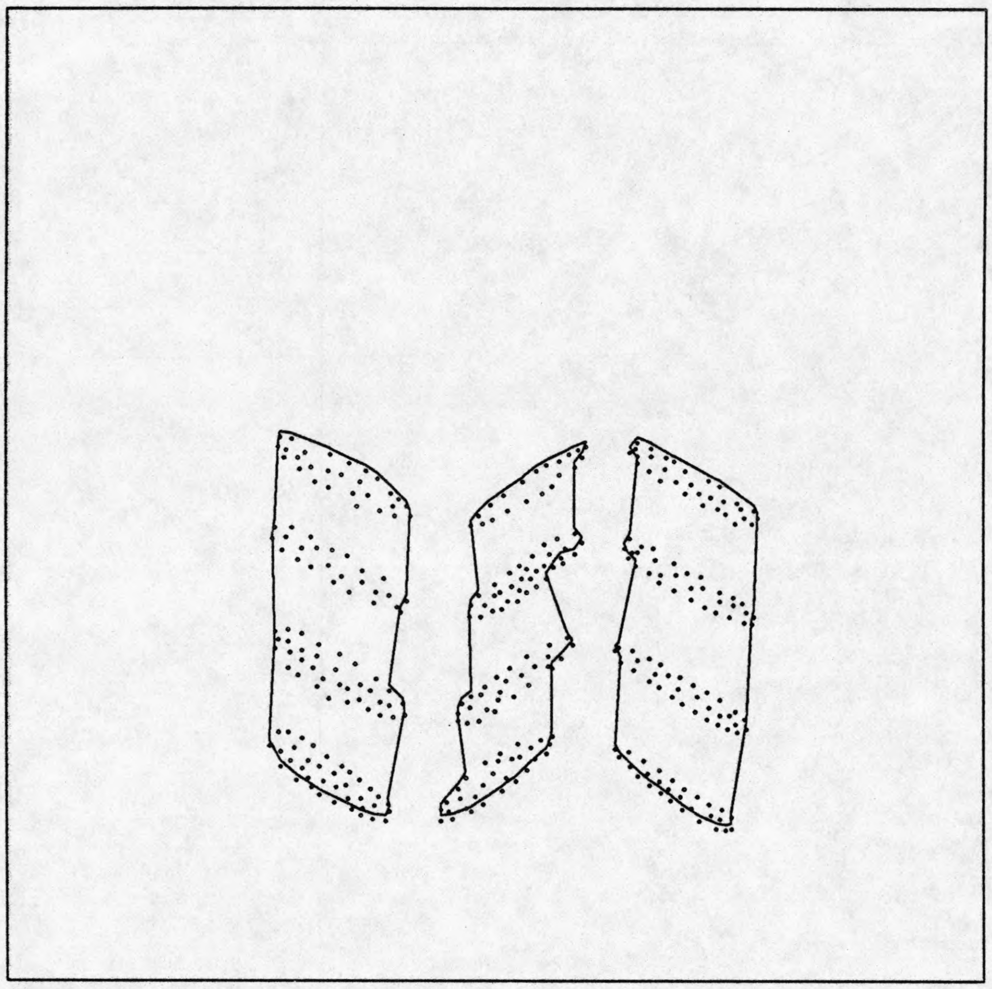


Figure 66. The orientation based grouping of the lowest level tokens obtained for the pattern shown in Figure 55. There are three clusters each consisting of identically oriented tokens with tokens in different clusters having different orientations.

6.6. Size Based Grouping

The size of a token is simply the area of the token. The size based grouping module uses a probabilistic relaxation labeling process which labels an edge between two tokens in the adjacency graph as either *EXIST* or *NOT−EXIST* according to whether the tokens have similar or dissimilar sizes. The computation of the compatibility coefficients is given in Appendix C. Finally, the result of this module on an example pattern is shown in Figure 67.

6.7. Curve Based Grouping

The curve based grouping module also uses a probabilistic relaxation labeling process which labels an edge between two tokens in the adjacency graph as either *EXIST* or *NOT−EXIST*, according to whether the tokens lie on a curve or not. The compatibility coefficients are computed in a way similar to the proximity based grouping, using the same definition of the distance measure. The compatibility coefficients are computed so as to enforce curve smoothness. The computation of the compatibility coefficients is given in Appendix C. Finally, the result of this module for the example pattern of Figure 56 is shown in Figure 68.

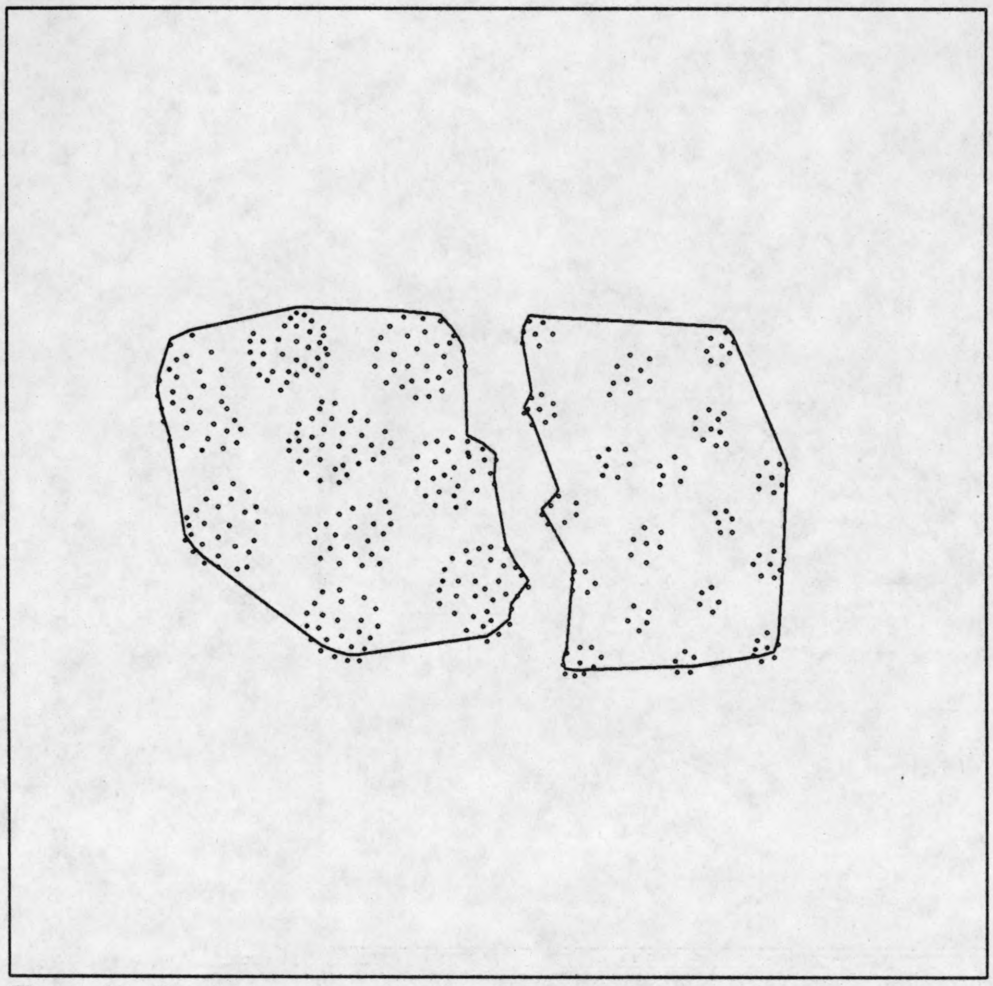


Figure 67. The size based grouping of the lowest level tokens for the pattern shown in Figure 57.

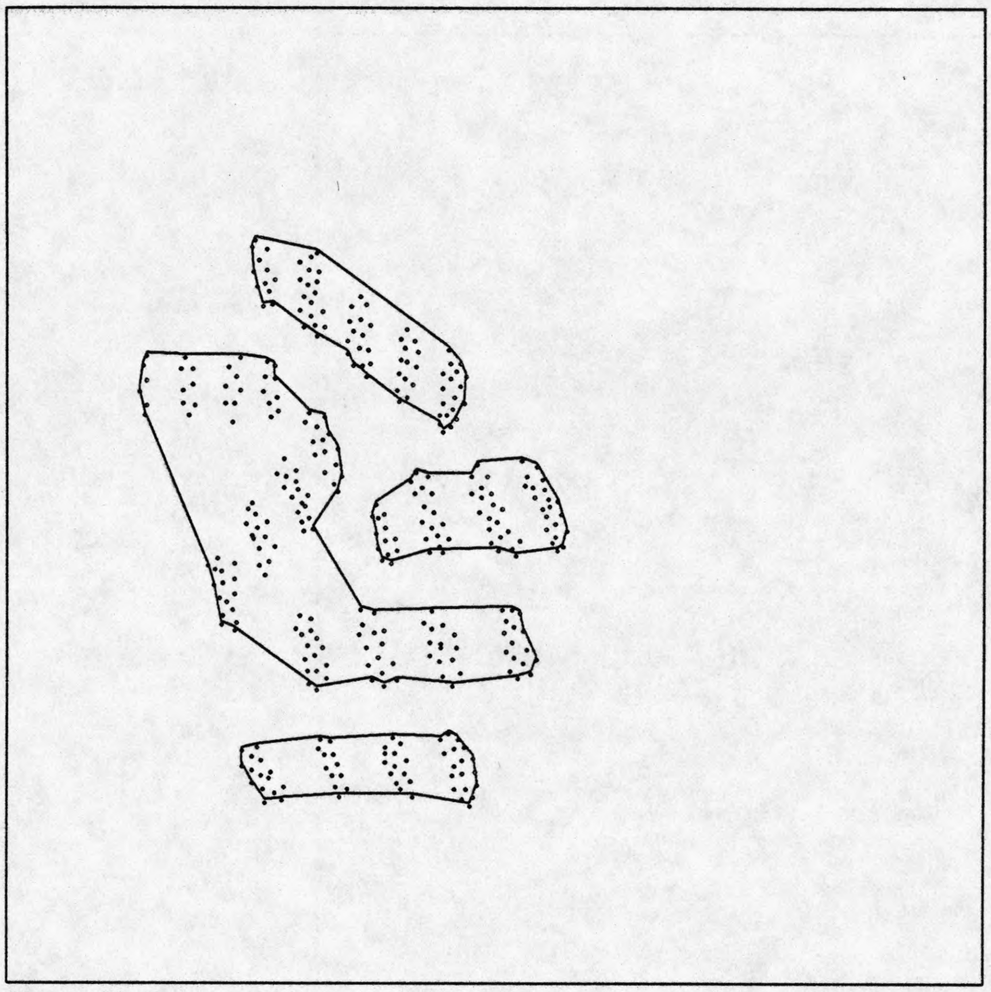


Figure 68. The curve based grouping of the lowest level tokens for the pattern shown in Figure 56.

6.8. Terminator Based Grouping

The terminators are identified for each token obtained during hierarchical grouping. This results in a new dot pattern in which perceptual structure needs to be identified. This structure may contain curves, or bar-like patterns which may not have any interior dots. To identify this structure, the dot pattern of the terminators is fed into the curve detector module of the lowest level grouping process. The result of this module for the example pattern of Figure 58 is shown in Figure 69.

6.9. Final Results of Grouping

Recall that after each module obtains its own grouping, their results are combined to obtain new tokens as shown in Figure 60. These new tokens are added to the existing pool of tokens, the adjacency graph is modified, and the hierarchical grouping process is repeated to obtain the next level of grouping as shown by the cycle in Figure 60. This process continues as long as new groupings are found and new tokens are generated during the execution of the cycle shown in Figure 60. When no new groupings are found, the cumulative set of tokens, along with their spatial interrelationships, such as containment and adjacency, define a multilevel description of perceptual structure in the original dot pattern. Figures 71-76 show the results of the various stages in the hierarchical grouping process from the start to the end for the example pattern shown in Figure 59; Figure 77 shows *all* groupings and

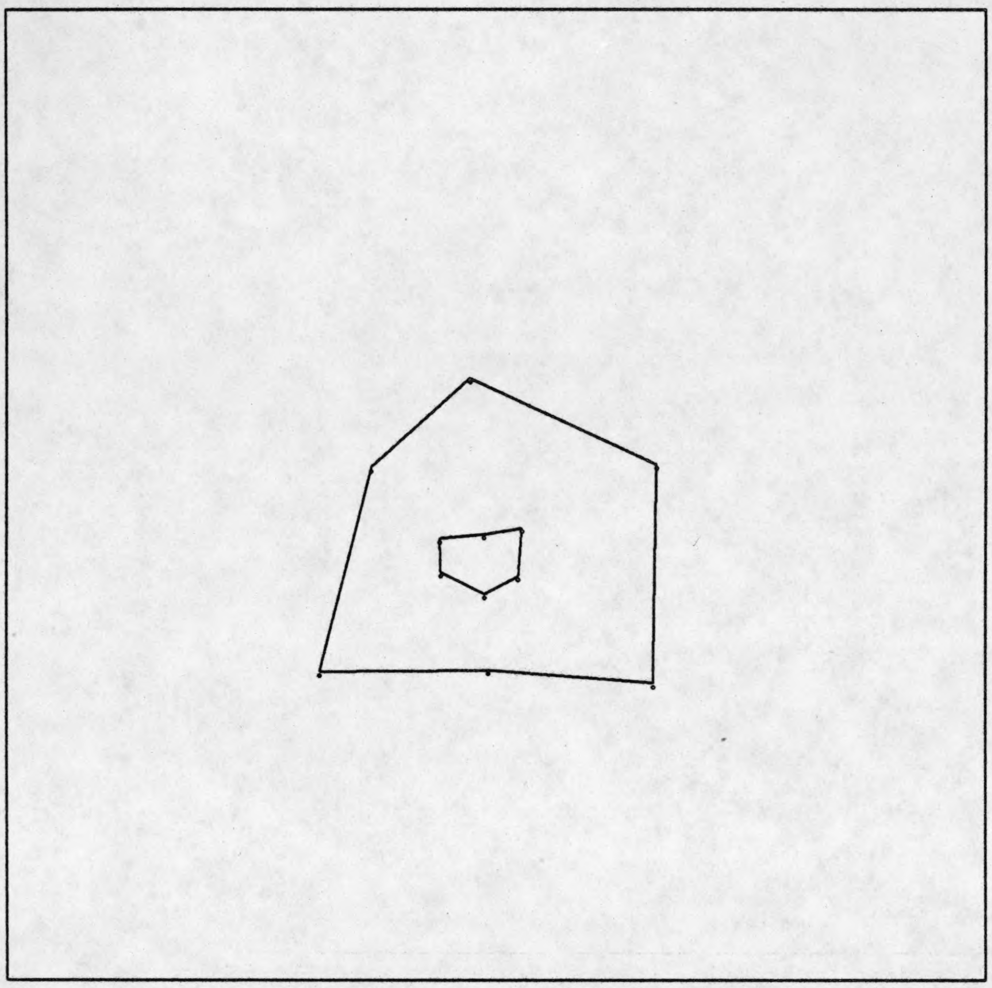


Figure 69. The terminator based grouping of the lowest level tokens for the pattern shown in Figure 58.

subgroupings found by the hierarchical grouping process, starting with the lowest level tokens until the algorithm stops. These various groupings together define the hierarchical structure extracted. This structure is shown in the figure by making an overlay of all groupings shown in Figures 71-76. Thus, on the original dot pattern are superposed all the borders that define the various groupings found. These borders tessellate the dot pattern into cells. The tessellation is recursive, i.e., cells merge together to form larger cells. The smallest cells represent elements of the smallest (micro) structure resolved. The property that groups the tokens present in a given cell is known from the specific grouping processes the intersection of whose results gives rise to the cell. Cells contained within other cells represent a structural detail within the parent structure. The spatial nesting of cells corresponds to the tree representation that we have alluded to earlier.

The process of overlaying (intersection) of the results obtained by different grouping processes gives rise to some empty cells. For viewing purposes such empty cells are distracting. Figure 78 shows the cleaned up version of Figure 77 where the edges surrounding empty cells have been deleted. The resulting partition of the dot pattern portrays the hierarchical structure detected by the algorithm. Figures 79-92 show the final results of hierarchical grouping on some other patterns; for each case we have shown the original pattern, the tessellation representing all the groupings, and the cleaned-up tessellation.

If the hierarchical structure is not unique, e.g. Figure 13, the tessellation will not be unique. In particular, in such cases, there may be multiple comparable ways of merging cells together to form larger cells, thus giving rise to multiple macro groupings that may be mutually exclusive. Sharing of a cell by multiple macro cells is equivalent to the interleaving of multiple tree hierarchies discussed earlier with regard to the representation of the perceptual structure.

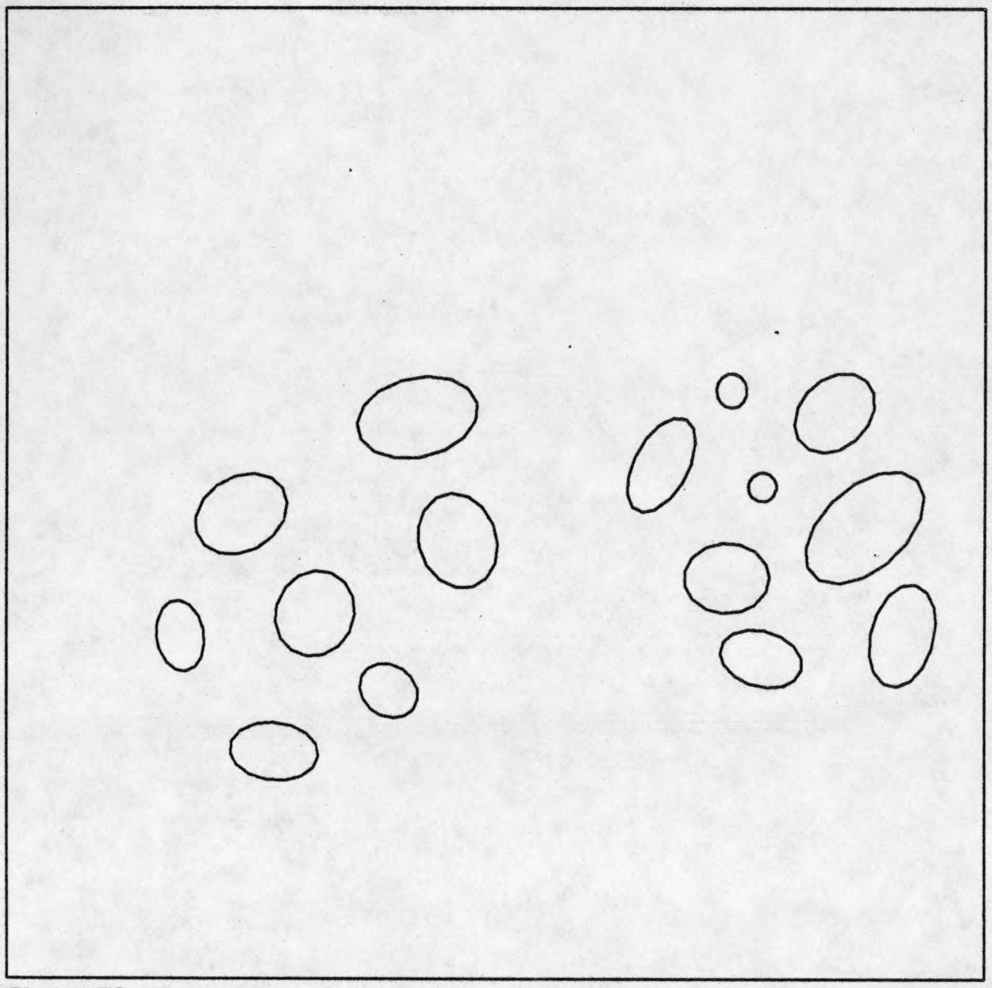


Figure 70. The initial set of ellipses representing the lowest level groupings for the pattern in Figure 59.

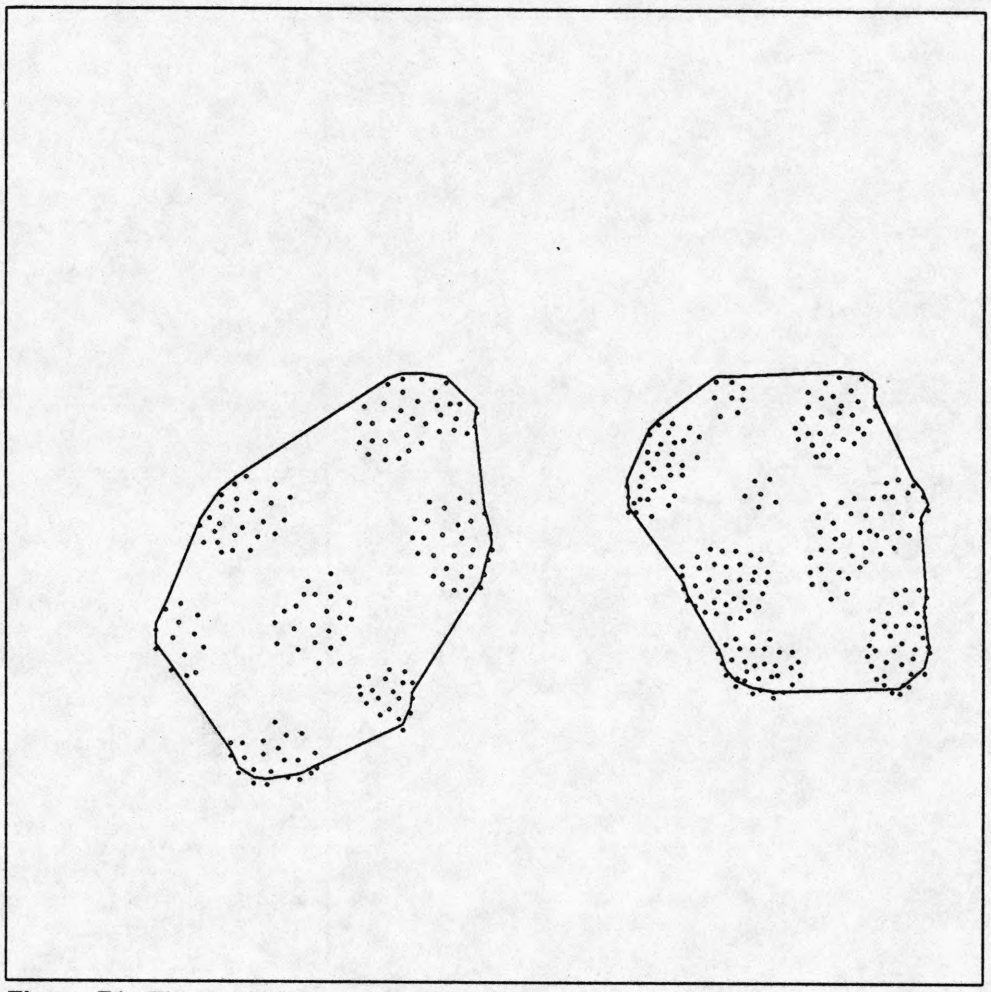


Figure 71. The proximity grouping of the tokens in Figure 70.

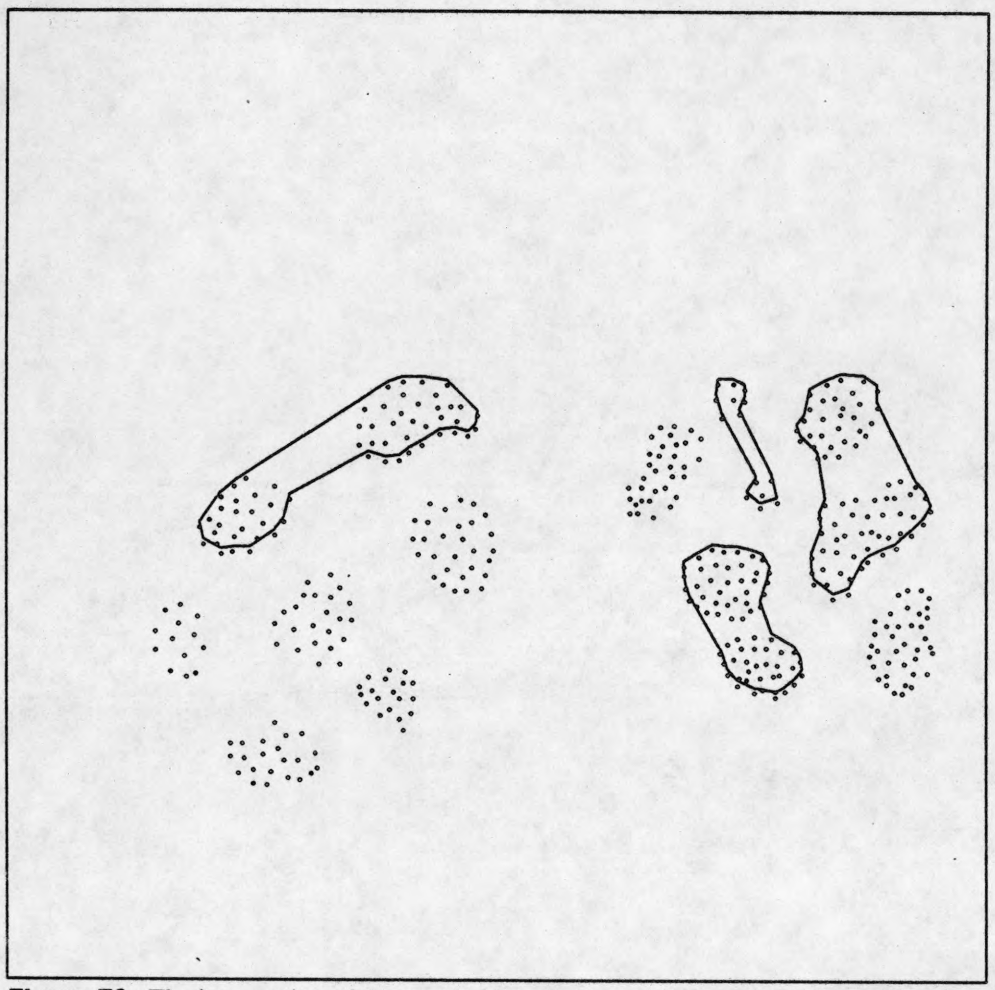


Figure 72. The intersection of the proximity and orientation grouping (Figure 60) for the tokens in Figure 70.

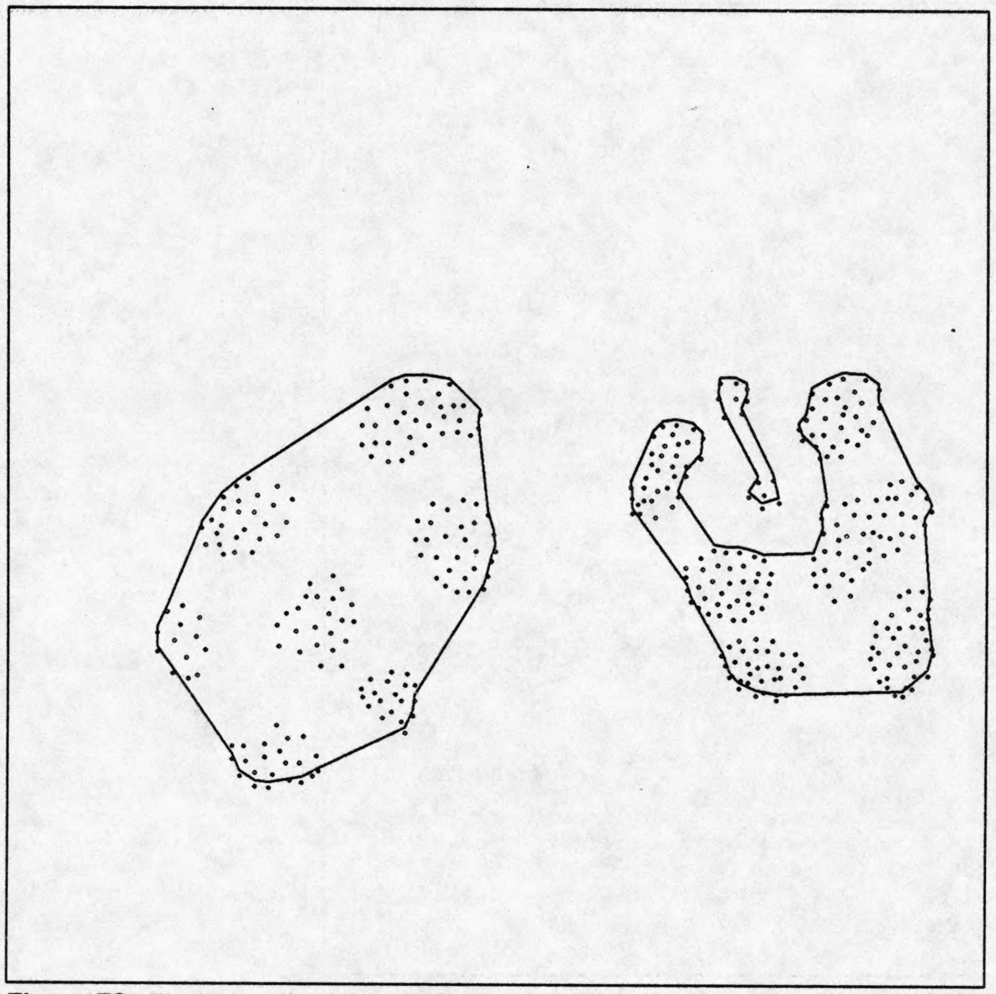


Figure 73. The intersection of the proximity and size grouping (Figure 60) for the tokens in Figure 70.

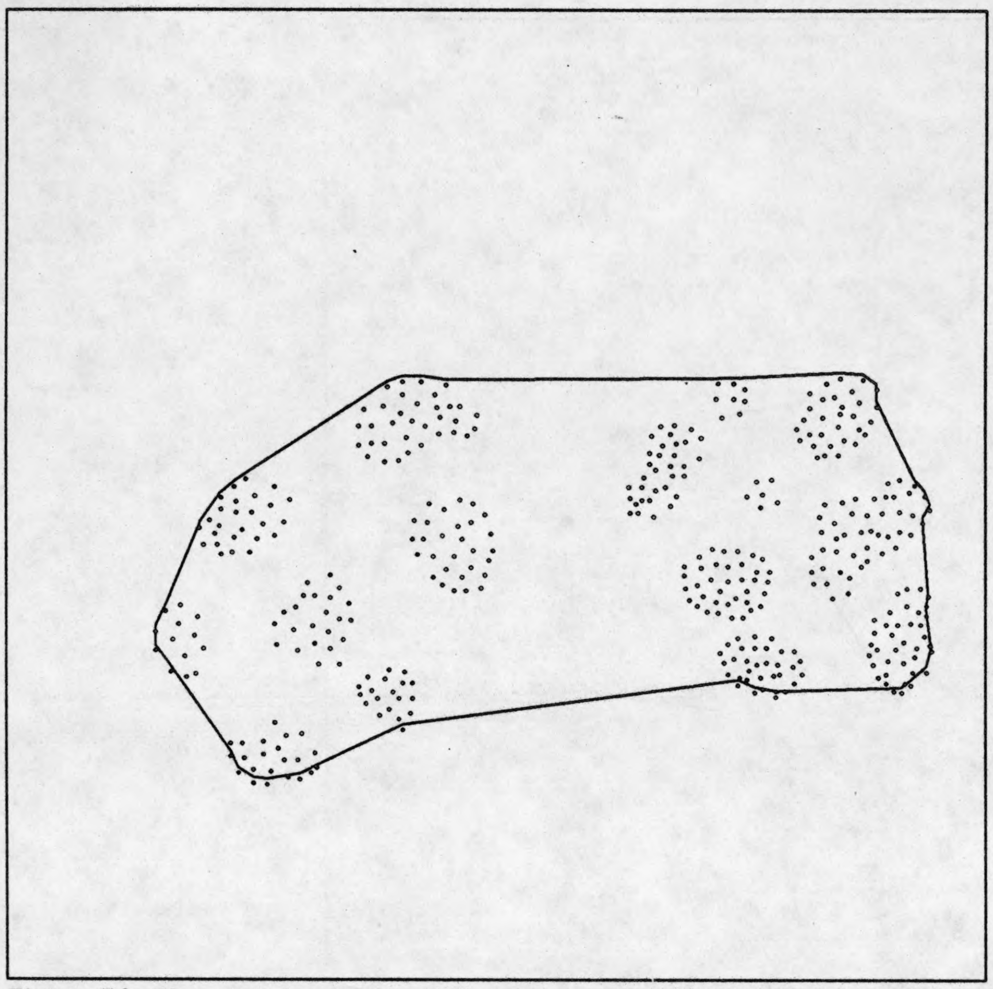


Figure 74. The proximity grouping during the second execution of the hierarchical grouping cycle of Figure 60. The tokens available for grouping during this cycle are the initial lowest level tokens (Figure 70) as well as those generated during the first cycle (Figures 71, 72, 73).

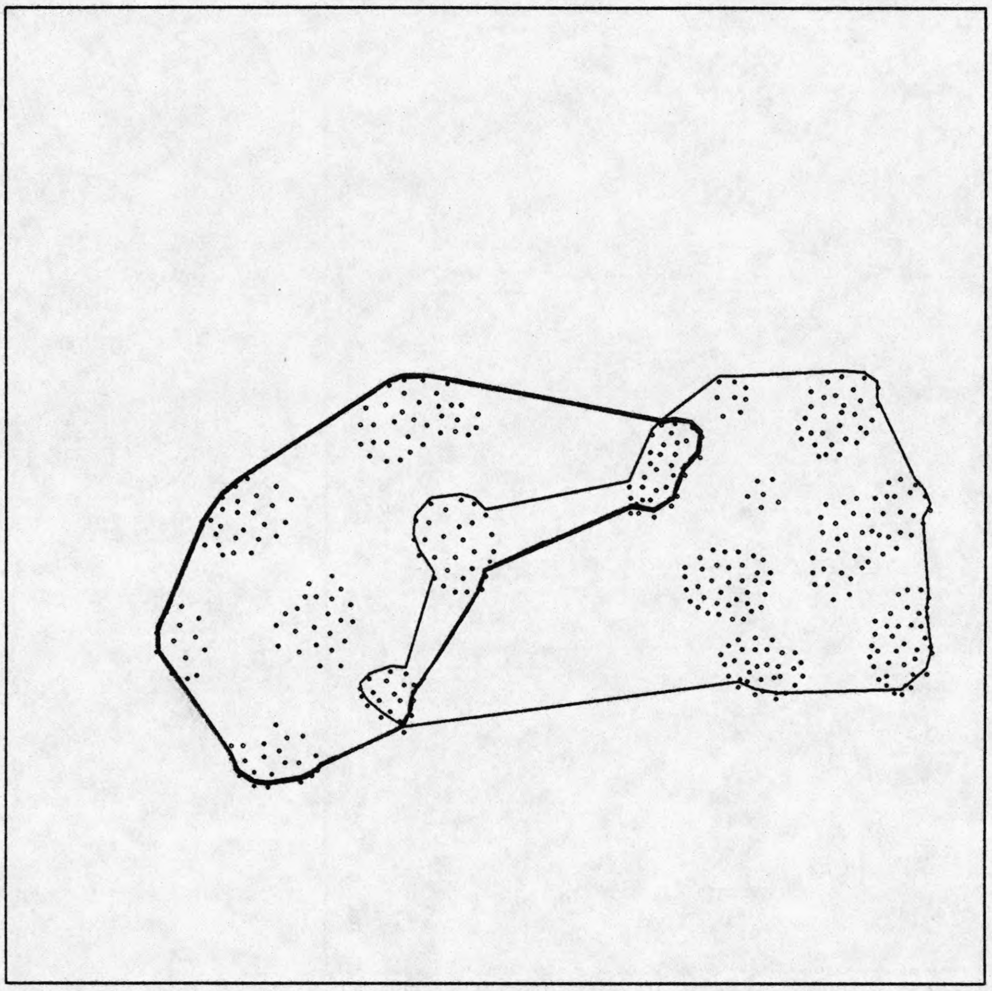


Figure 75. The intersection of proximity and orientation grouping during the second cycle. Here the border segments of one of the groupings have been made thicker to distinguish it from the other. The orientation based grouping is clustering together tokens having compatible orientations, irrespective of the sizes and levels of tokens.

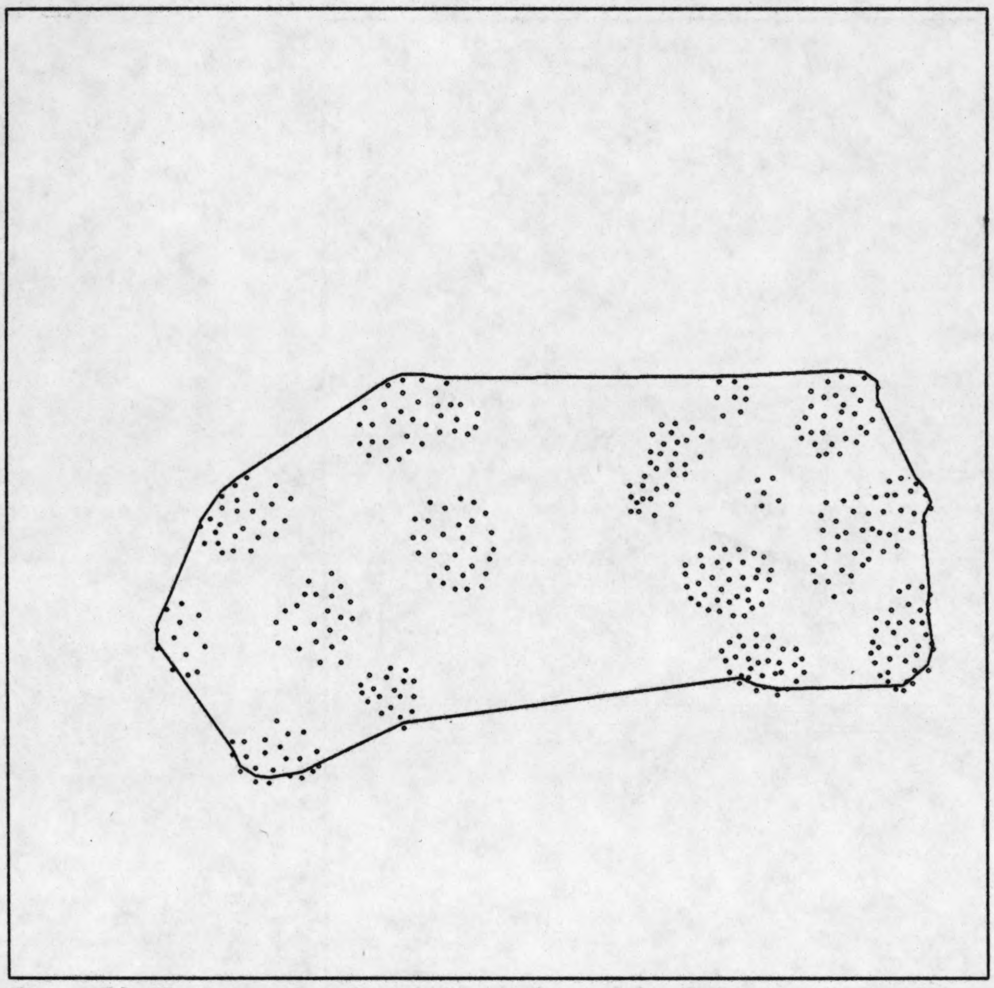


Figure 76. The intersection of proximity and size grouping during the second cycle.

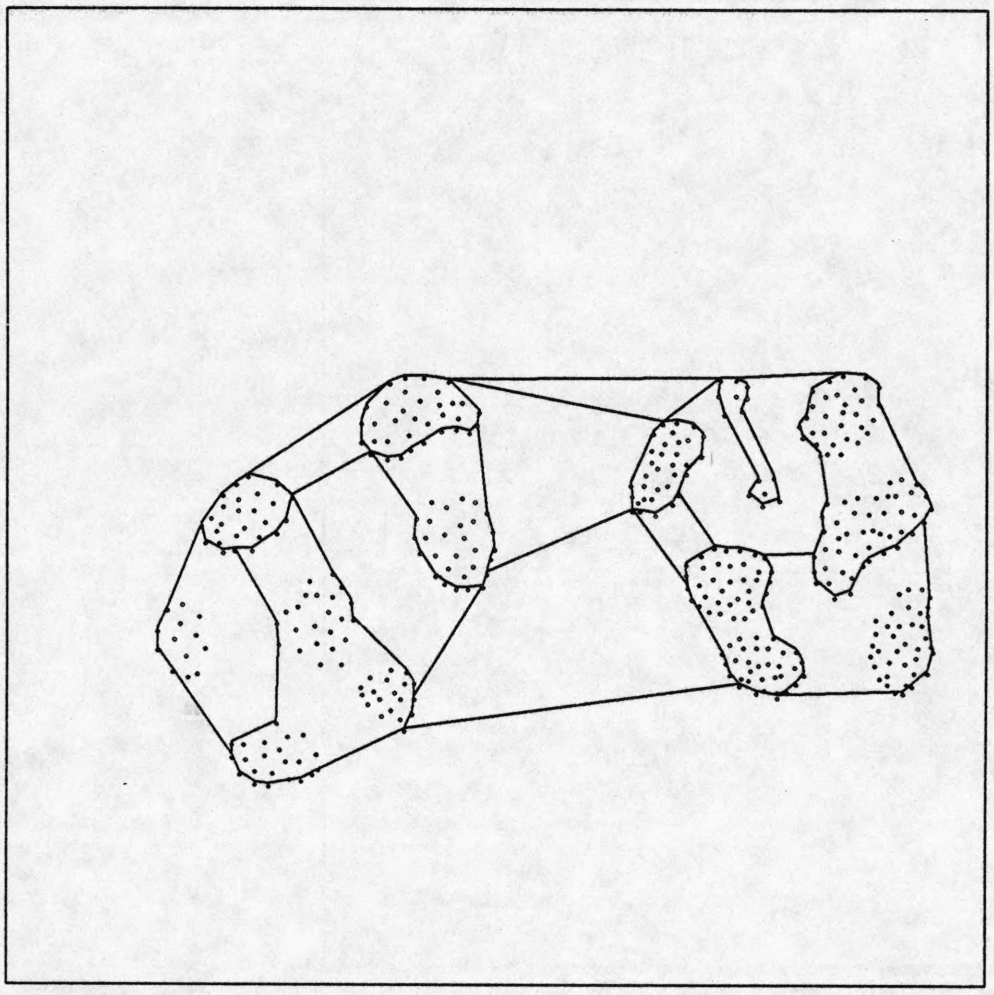


Figure 77. All groupings detected before the hierarchical grouping algorithm stops. This figure is an overlay of the results in Figures 71-76. The border edges partition the dot pattern into structural components. The empty cells are spurious, and are deleted in the next figure to help understanding of the detected perceptual structure.

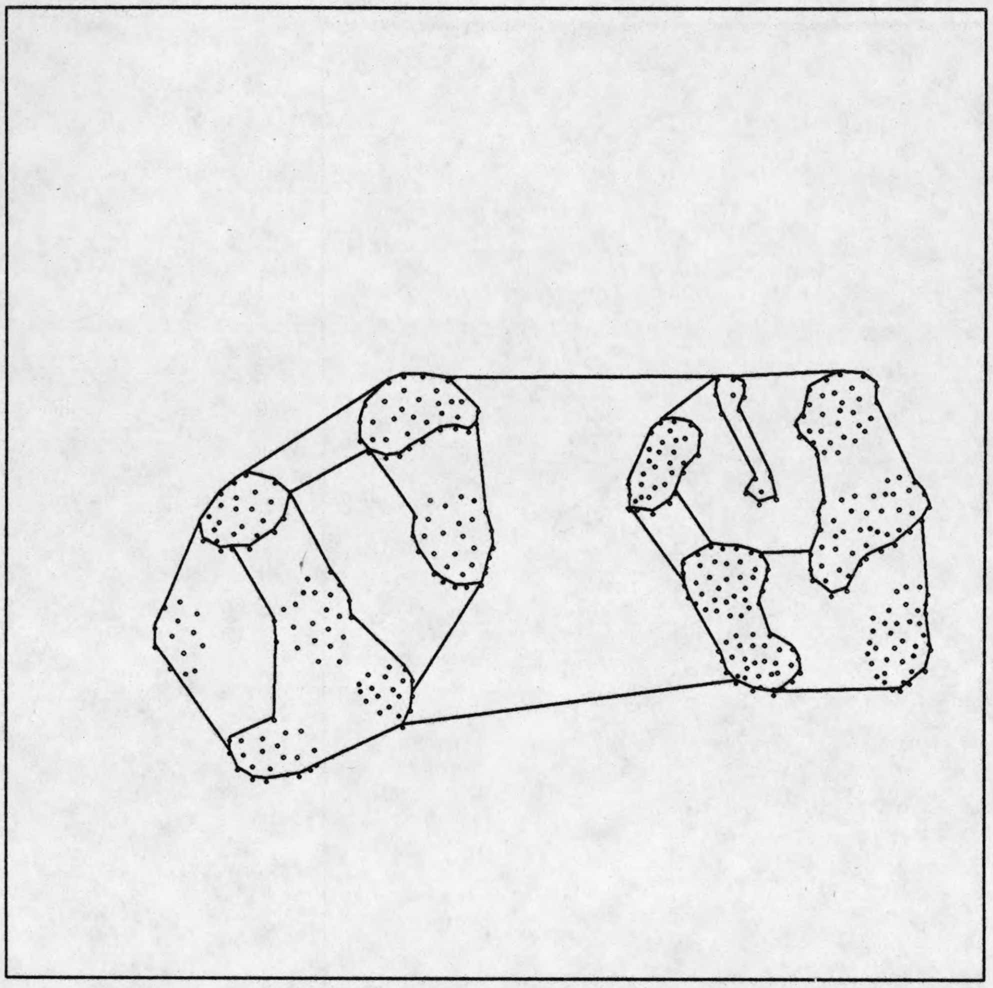


Figure 78. Cleaned up version of Figure 77 for aid in viewing. The structure detected is shown by a recursive tessellation. The outermost border represents the final grouping. Within this border are subcells each of which has its own subcells, and so on. Different subcells may arise from different grouping criteria. The overall embedding of cells corresponds to the hierarchical structure perceived.

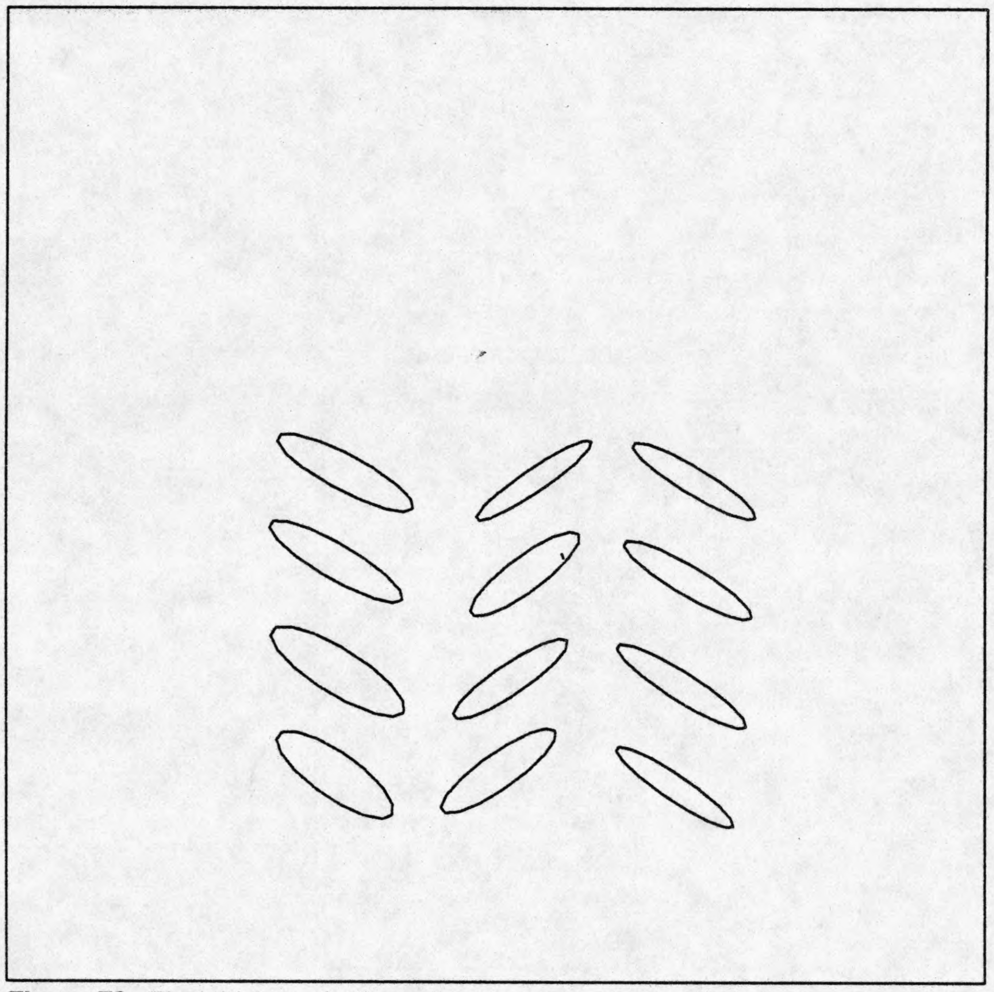


Figure 79. The initial set of ellipses fit to the tokens obtained from the lowest level grouping for the pattern in Figure 55.

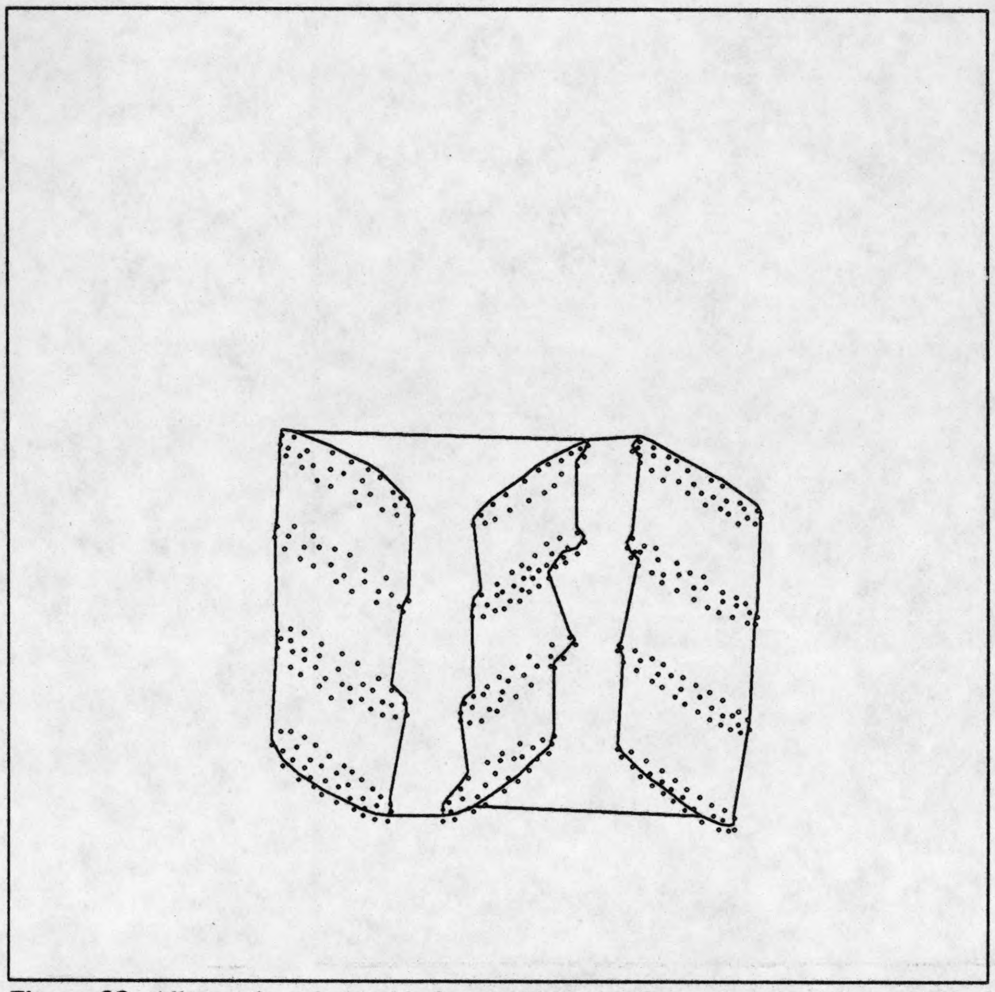


Figure 80. All groupings detected before the hierarchical grouping algorithm stops.

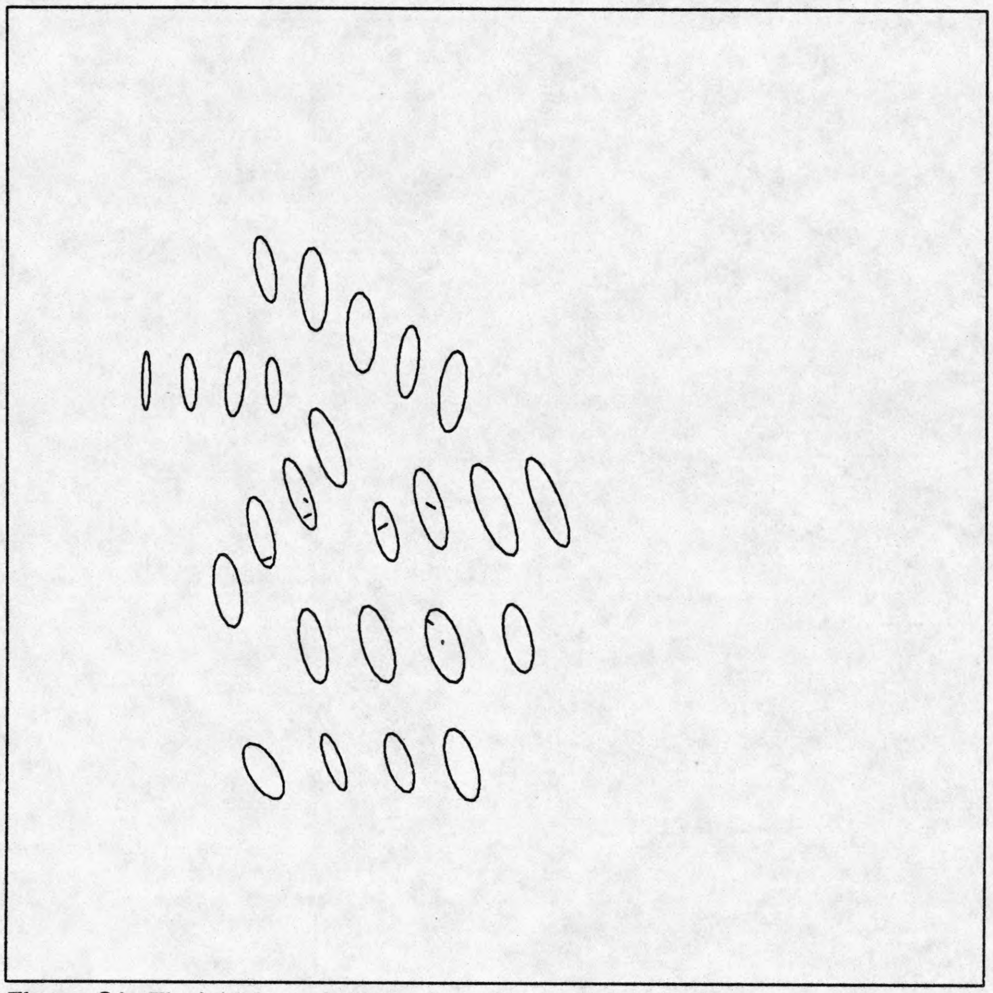


Figure 81. The initial set of ellipses fit to the tokens obtained from the lowest level grouping for the pattern in Figure 56.

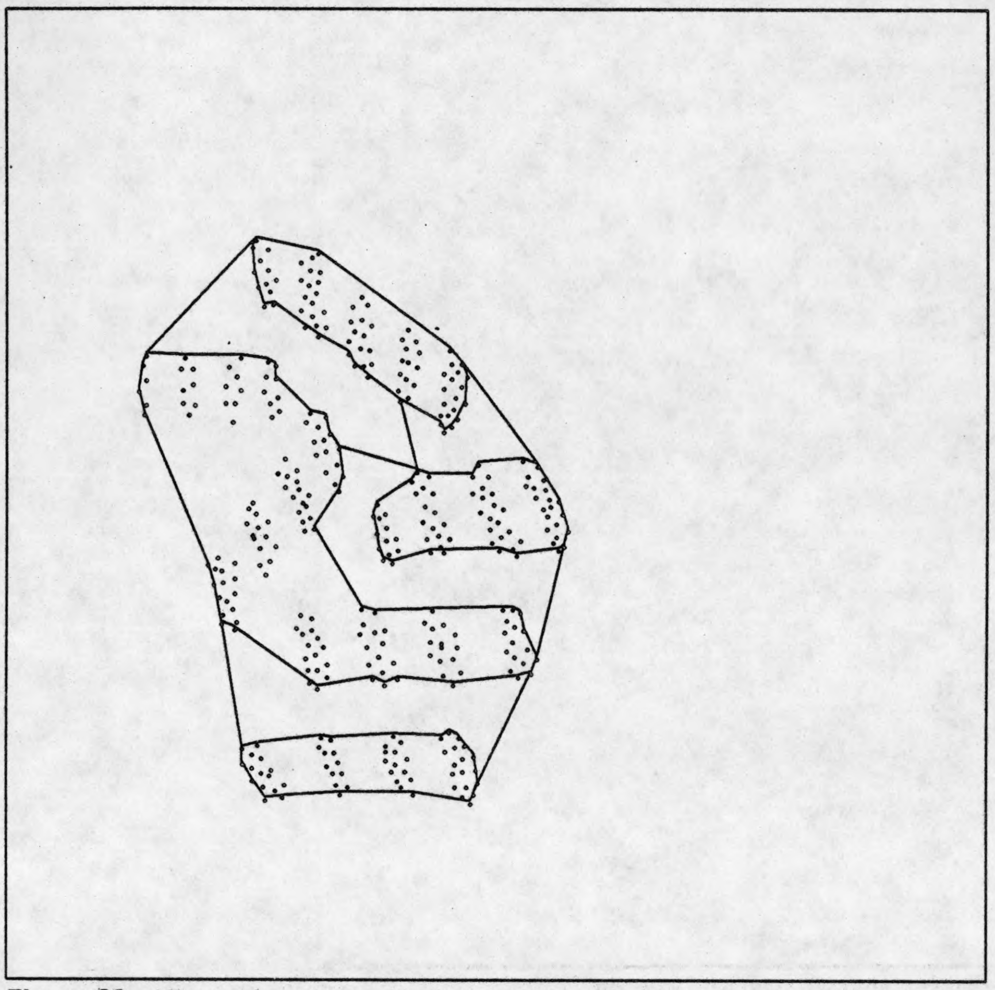


Figure 82. All groupings detected before the hierarchical grouping algorithm stops.

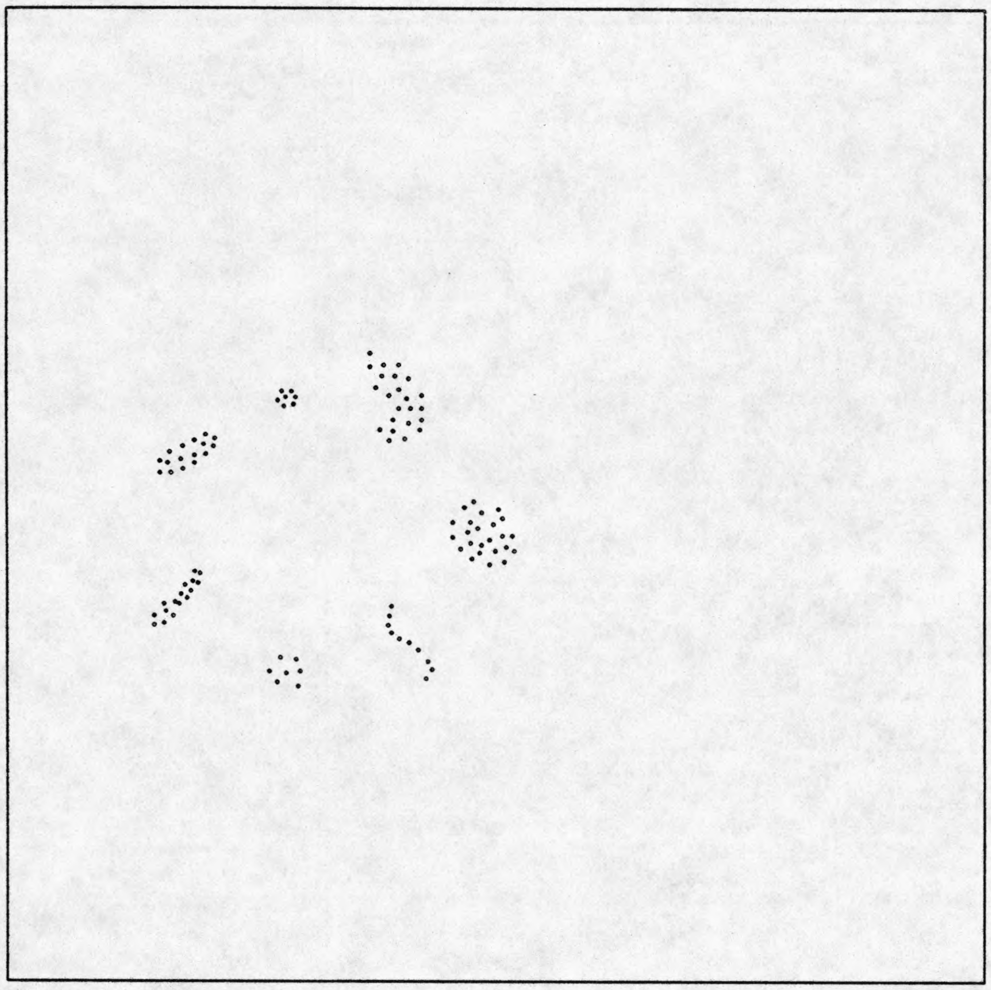


Figure 83. An example dot pattern with a hierarchical structure.

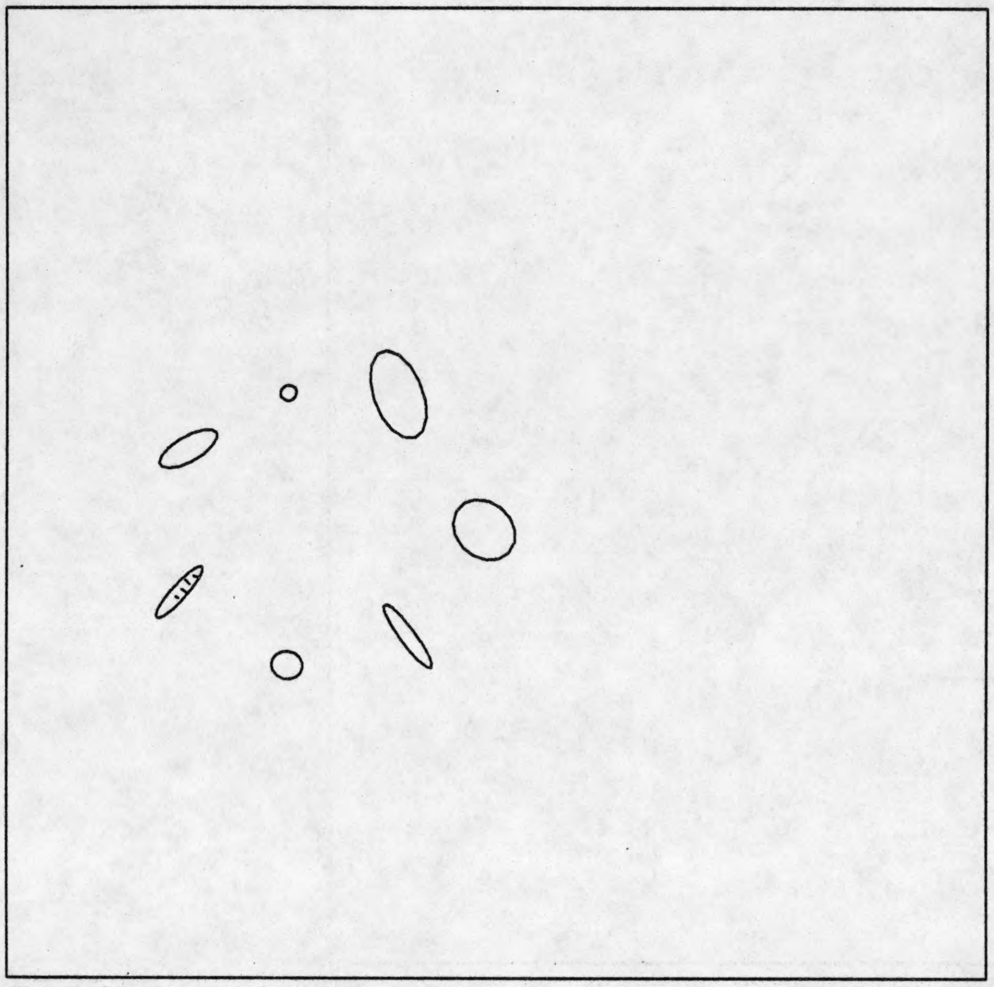


Figure 84. The initial set of ellipses fit to the tokens obtained from the lowest level grouping for the pattern in Figure 83.

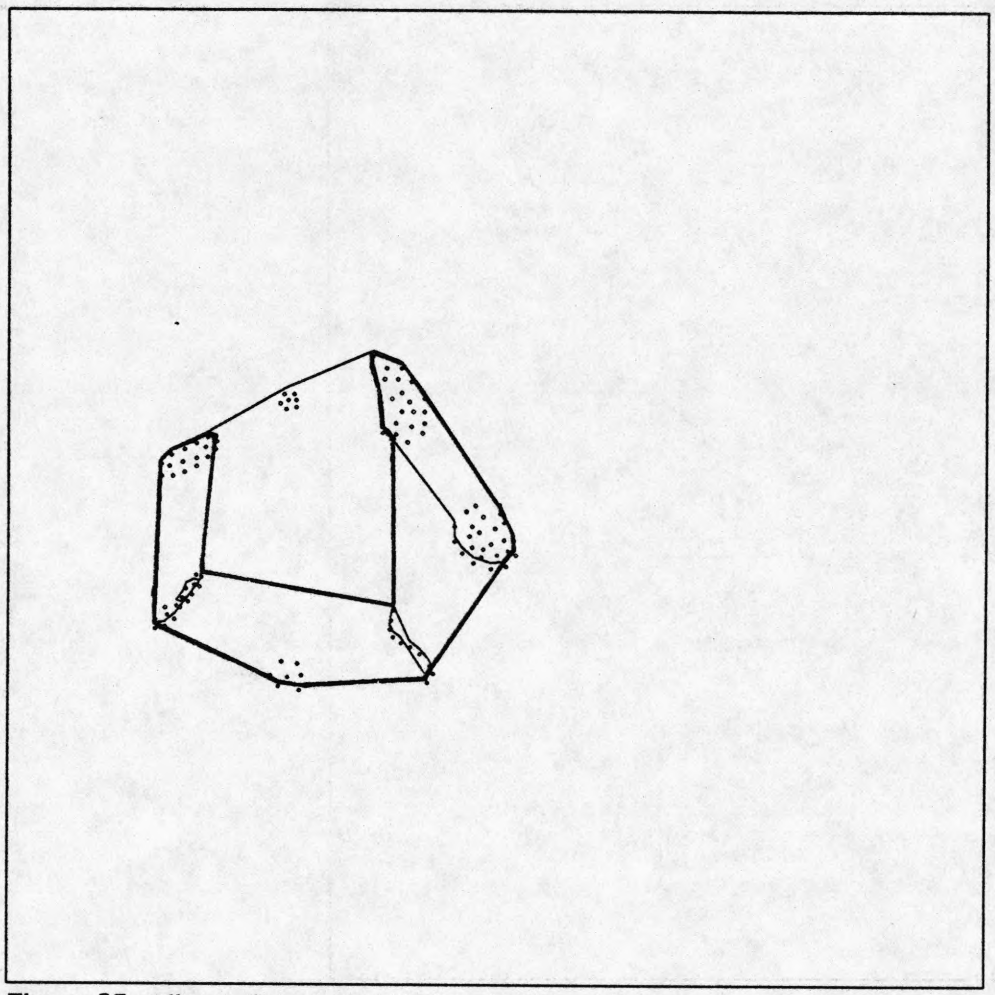


Figure 85. All groupings detected before the hierarchical grouping algorithm stops.

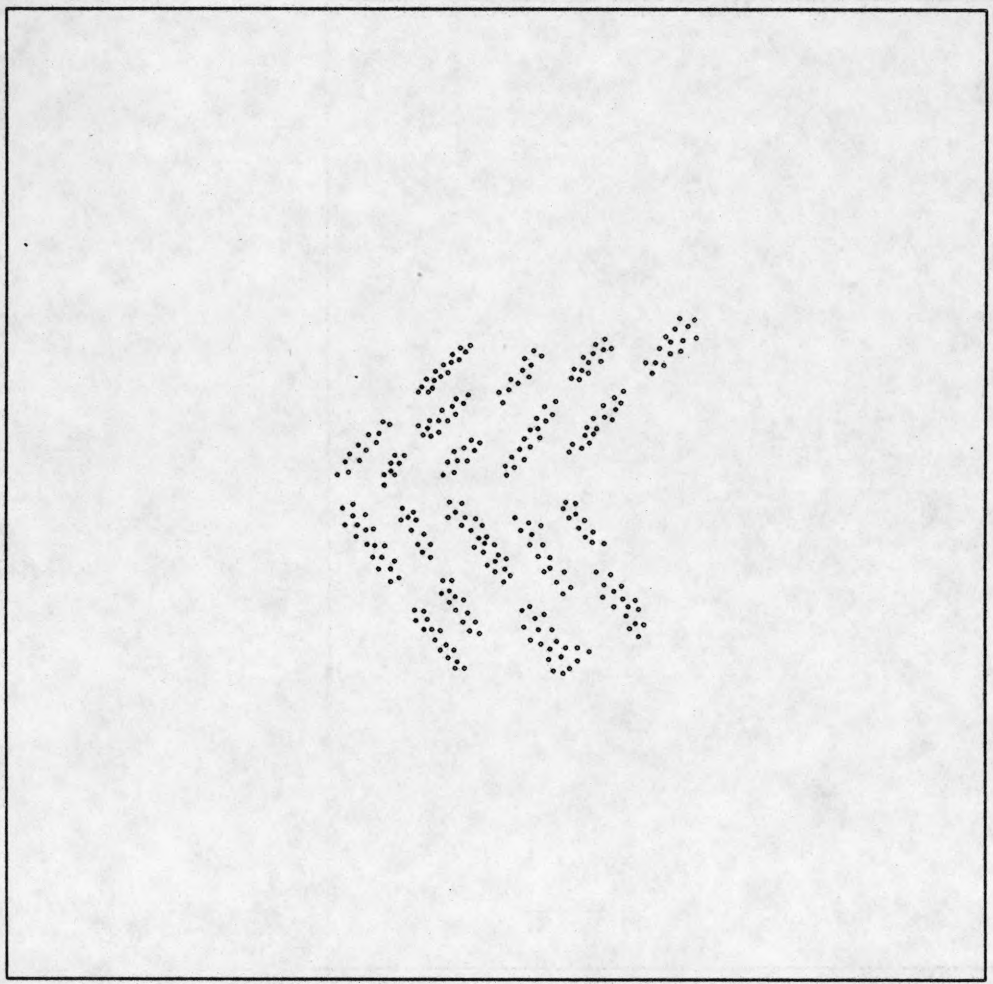


Figure 86. An example dot pattern with hierarchical structure.

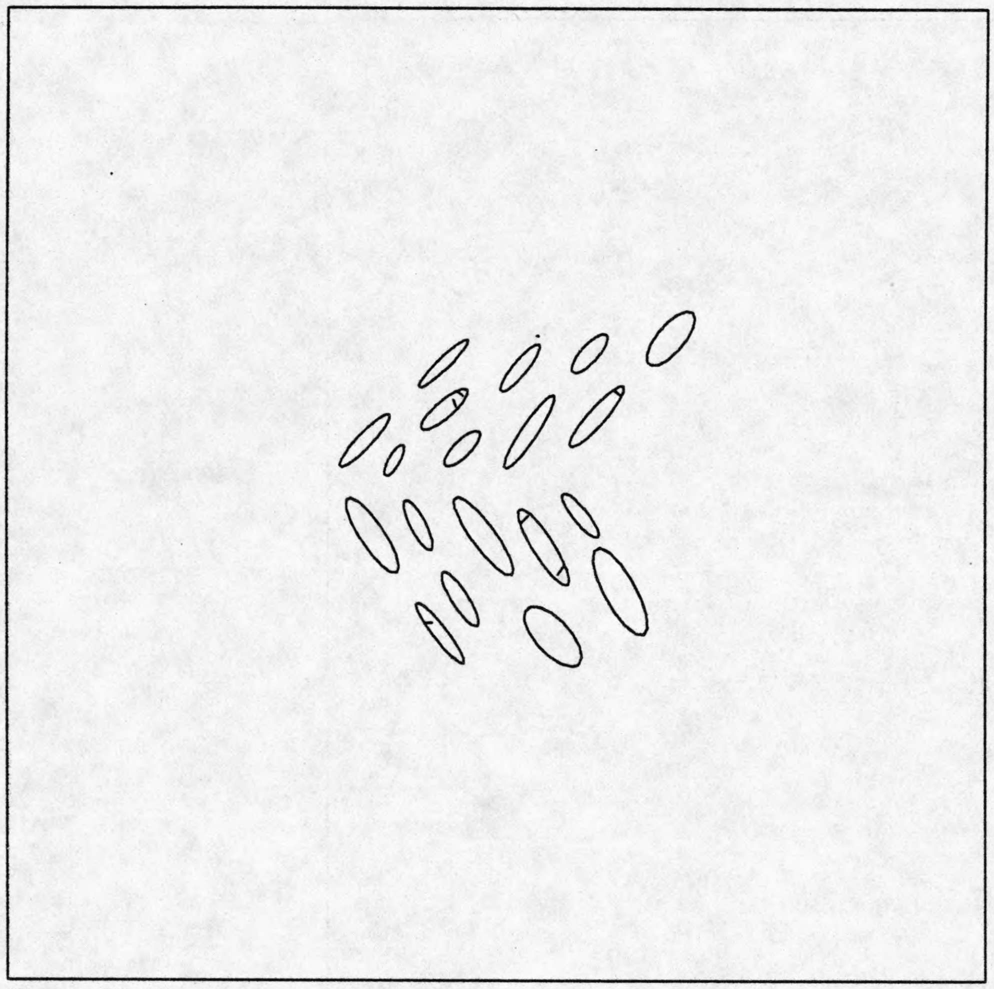


Figure 87. The initial set of ellipses fit to the tokens obtained from the lowest level grouping for the pattern in Figure 86.

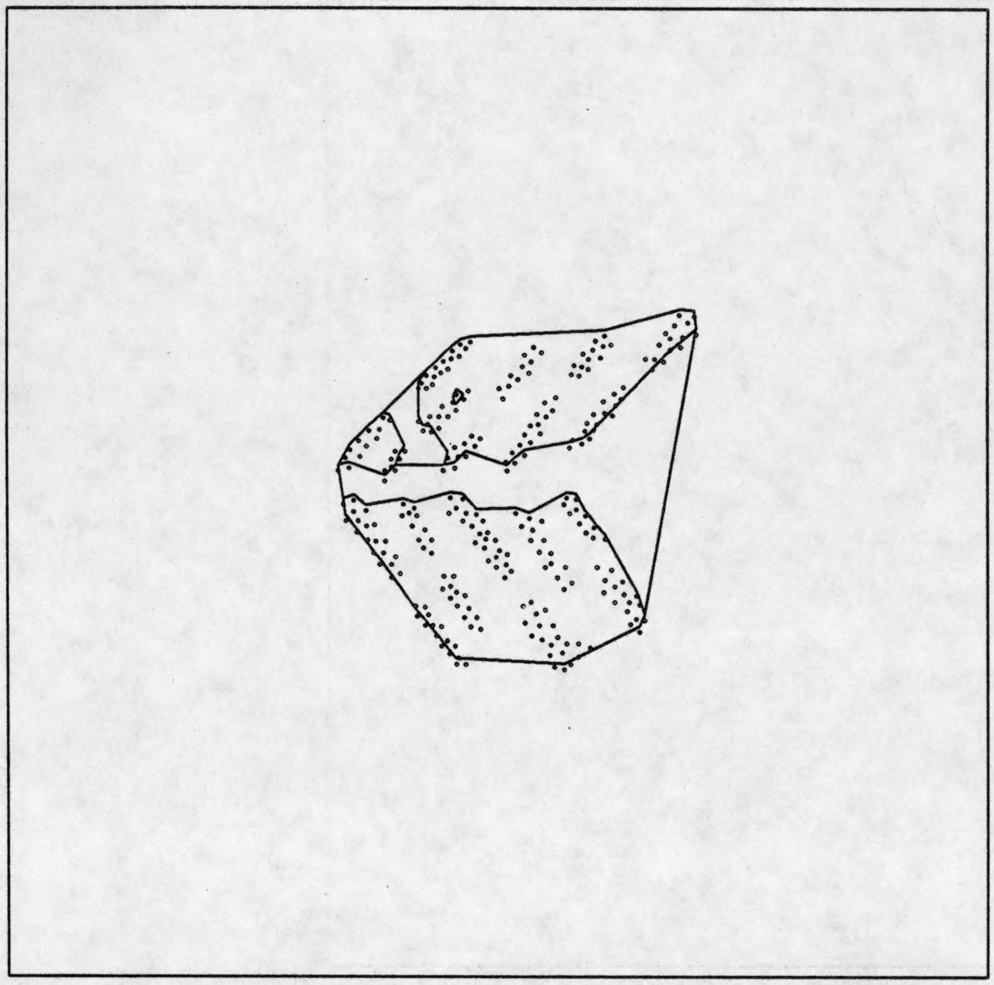


Figure 88. All groupings detected before the hierarchical grouping algorithm stops.

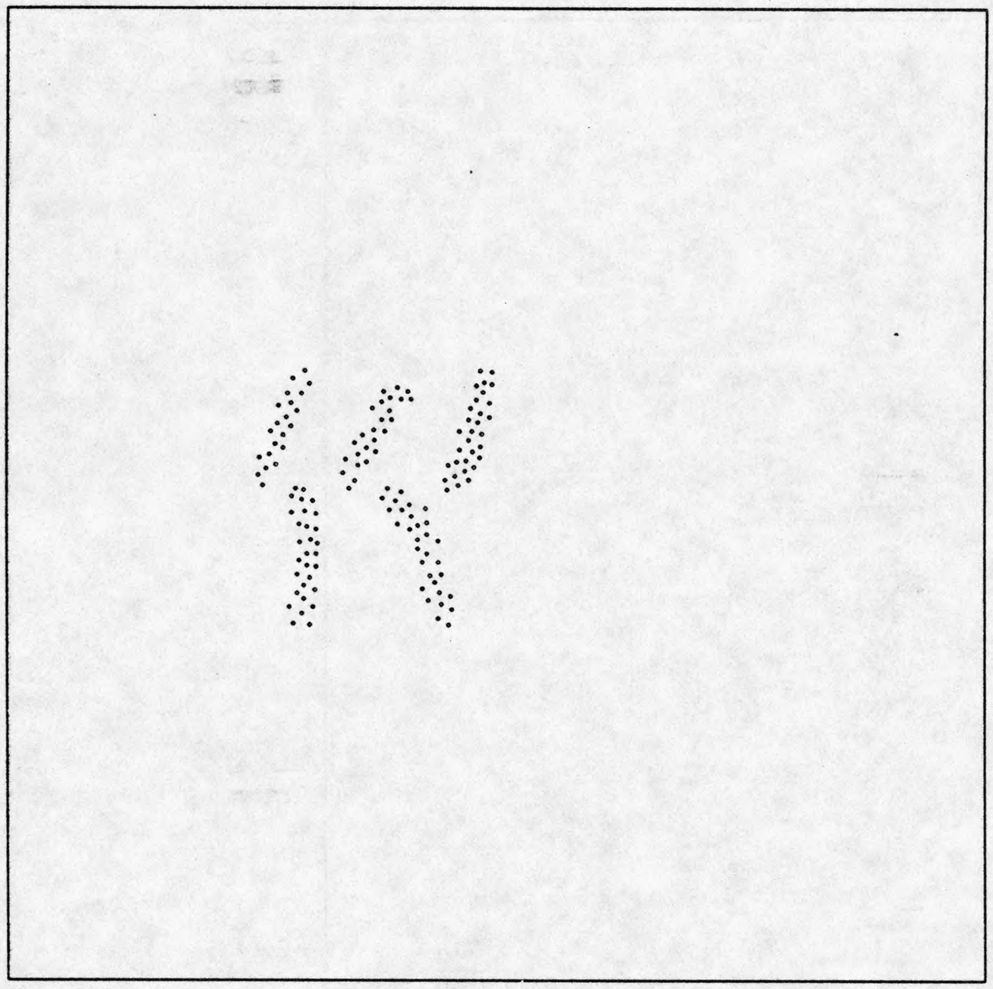


Figure 89. An example dot pattern in which the termination points of the lowest level tokens lie along a curve.

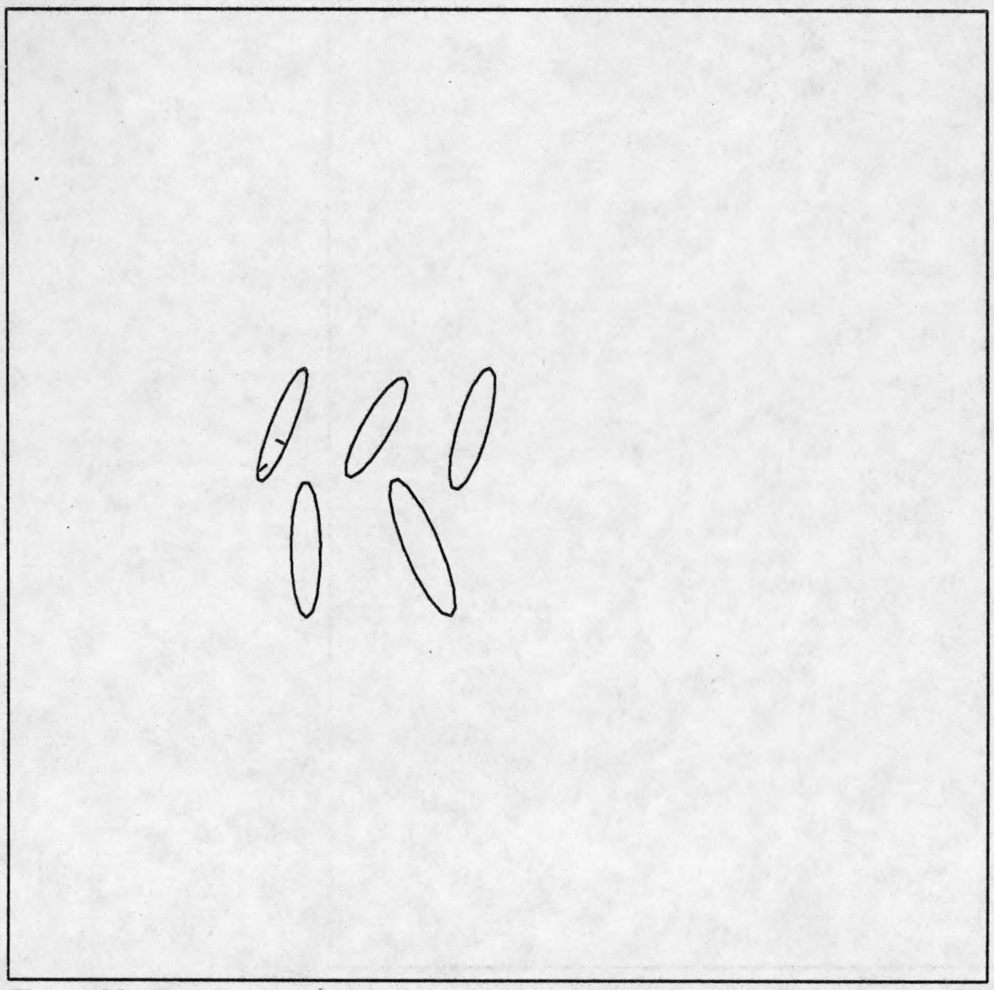


Figure 90. The ellipse representations obtained for the lowest level tokens for the pattern in Figure 89.

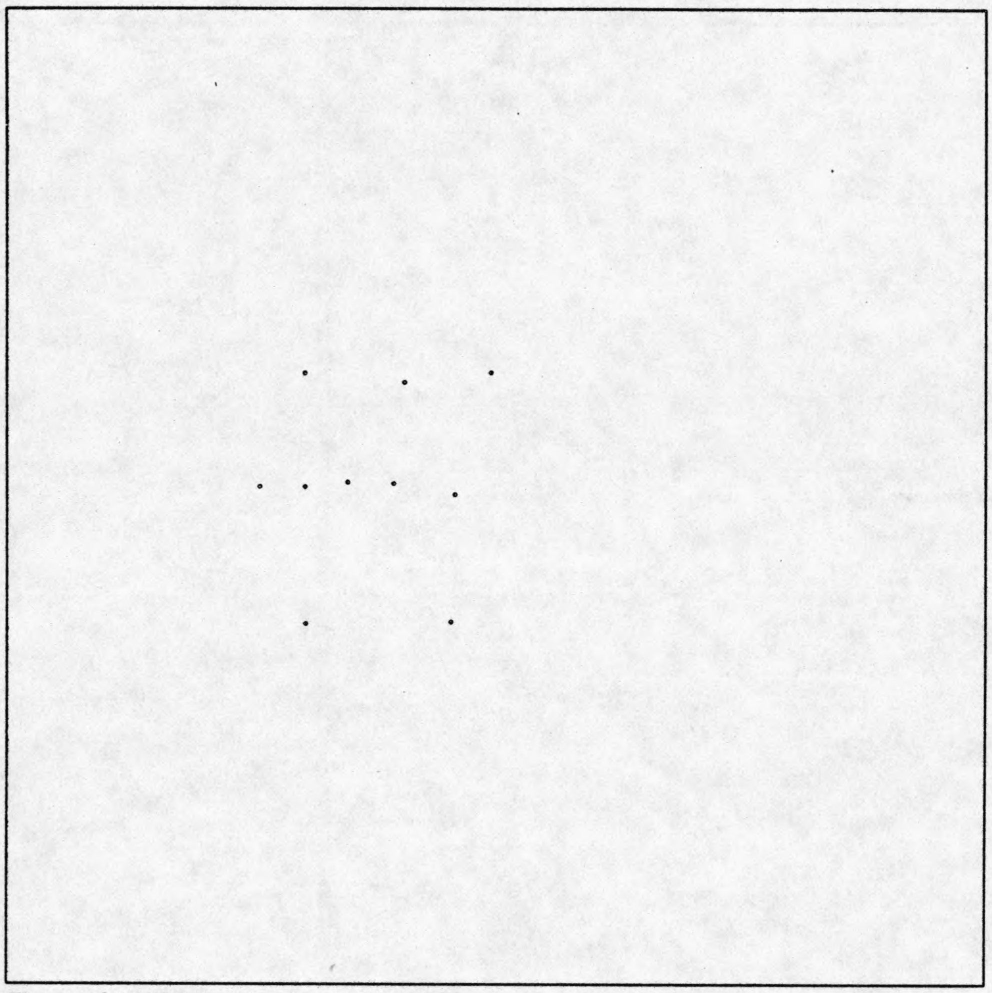


Figure 91. The termination points of the tokens obtained in Figure 90.

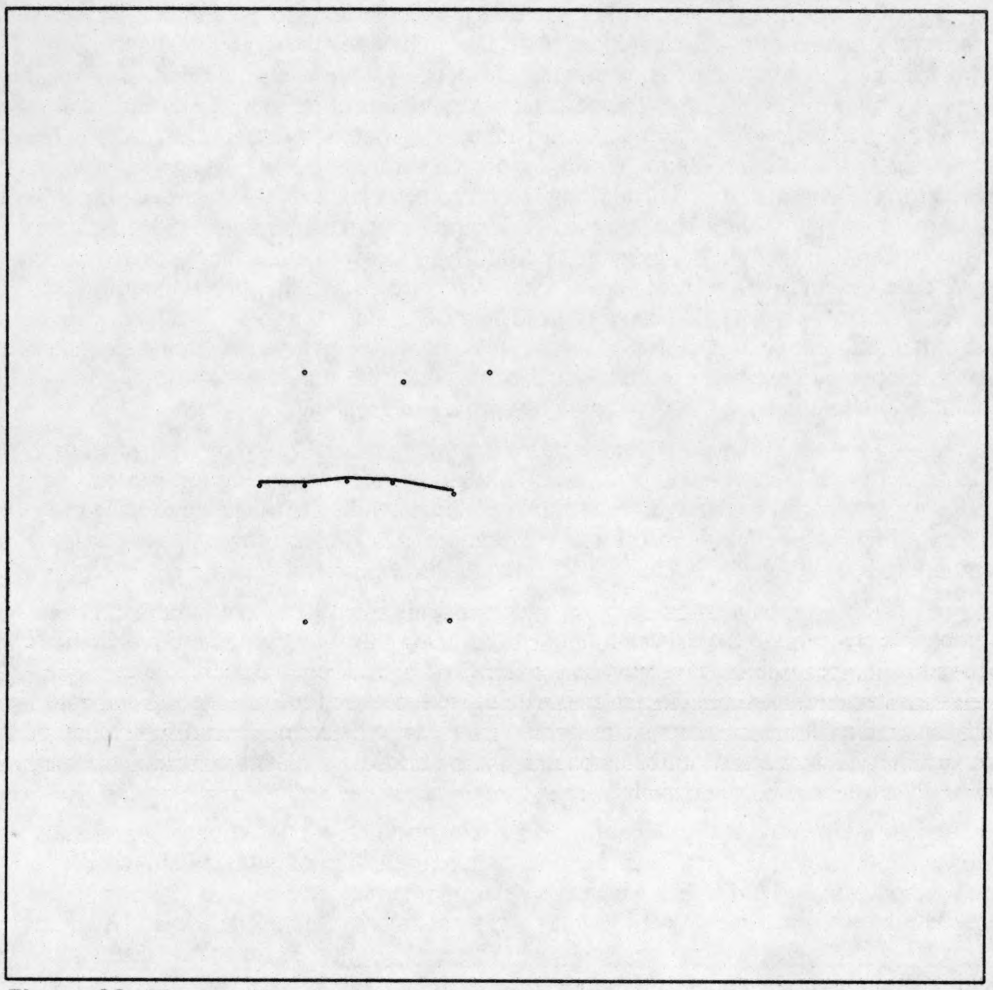


Figure 92. The curvilinear structure detected in the dot pattern formed by the terminators in Figure 91.

7. SUMMARY

We have described an integration approach to extraction of perceptual structure in a dot pattern that may exist at a range of levels. The lowest level structure of relative dot locations is represented via the Voronoi neighborhoods of the dots which are two-dimensional polygonal regions containing the dots. The geometric properties of these neighborhoods capture the local environments of the dots. The perceptual structure is inferred by first grouping dots into the lowest level segments, or the lowest level tokens, and then recursively grouping these tokens further. The lowest level grouping is accomplished by labeling each dot as an interior dot, border dot, curve dot or isolated dot. For each label, the typical geometric environment of a dot fulfilling the given perceptual role is expressed in terms of the properties of the dot's Voronoi neighborhood. The observed geometric properties of the neighborhoods thus provide an initial estimate of the perceptual role of a dot as evidenced locally. However, since the perceived structure depends not only on local criteria, but also on more global Gestalt criteria such as border smoothness, the locally assigned labels of dots are modified to meet non-local criteria. Thus, the evidence from the local structure of a dot may be overruled to assign it a different perceptual role if this makes a border or a curve to become smoother, or at a higher level, if it makes a component become more compact. This need for the simultaneous use of multiple constraints to perform perceptual interpretation suggests an integrated approach to grouping.

The lowest level tokens are then further grouped by a hierarchical grouping process, based on their proximity, orientations, sizes and terminations. The different groupings are accomplished by different concurrent processes all of which start with the lowest level tokens, and constantly introduce tokens corresponding to the new groupings they find. When no more groupings are found, the grouping process stops. The final result of the lowest level grouping followed by the hierarchical grouping is a multilevel structural description of the original dot pattern.

All constraints are enforced using a probabilistic relaxation process. This often helps us avoid using rigid thresholds. The relaxation compatibilities capture the expected interactions among the objects and the probabilities of interpretations to which the relaxation procedure converges are usually close to 1. We have attempted to minimize the use of ad hoc thresholds for decision making. Whenever possible, we have tried to increase the amount of information used to reduce any ambiguity in interpretation. The thresholds used are listed in Appendix A. Many of the thresholds used are adaptive. That is, they are specified as bounds on the allowed statistics of local, structural characteristics of the dot pattern, rather than as absolute numbers.

The current implementation of the hierarchical grouping process does not incorporate all necessary details. For example, it does not enforce the condition that termination points belonging to the same token should not be grouped if they are separated by the token body. Further, if the tokens are not sufficiently elongated, we may not want to extract orientation based groupings since such groupings may fragment other groupings determined by more dominant properties. Examples of such fragmentation can be seen in Figures 72 and 75. In these figures most tokens grouped by the orientation based grouping are not very elongated.

Currently, the relative perceptual salience of the various groupings is not taken into account. For example, the proximity based grouping dominates the orientation based grouping: the proximity based subgroupings of an orientation based grouping are easily perceived, whereas it requires "strong" orientation based subgroupings to be perceived within a proximity based grouping. Similarly, the size based grouping appears to dominate the orientation based grouping. A measure of the relative salience of groupings needs to be incorporated in our implementation. Further, as stated in Section 6, we have not addressed the task of hierarchically grouping a large number of tokens in cases where global texture perception plays a significant role.

The ellipse fitting algorithm for obtaining tokens needs improvement. The current version computes the moments of mass of dots along the cluster border. The computed moments are sensitive to variations in density along the border. To eliminate this undesirable dependence, the moment of area can be computed for the entire region enclosed by the cluster border. Another area of improvement is the computation of the curvature of border segments. Currently, the curvature is computed by examining the angles between the Delaunay edges connecting successive border dots. This can be improved by fitting a smooth curve to the set of border dots and using the curvature of the curve segment. We plan to carry out some of these enhancements outlined here in future work.

APPENDIX A

SUMMARY OF THRESHOLDS

The number of thresholds used by the algorithm is small. Many of the thresholds are adaptive, i.e. they are specified as bounds on acceptable, local statistical properties, rather than as absolute limits. Some of the thresholds are used a number of times in different contexts. This appendix gives a list of all thresholds used in our algorithm.

A.1. Cut-off on Probabilities

The relaxation labeling algorithm updates the probabilities of labels for each object. This process can be speeded up by having a certainty threshold on the probabilities of the labels such that a label probability above this threshold is considered to be 1.0. In our current version of the algorithm this certainty threshold is set to 0.9. This means that once the probability of a label on any object reaches a value of 0.9, the label is considered to be correct, and the label probability is set to 1.0.

A.2. Neighborhood for Distance Measure Computation

The computation of the distance measure shown in Figure 93 is based on the lengths of only a subset of the neighbors of dots i and j. The neighbors considered are those whose angles, α_k, with the line joining dots i and j are less than a threshold, T_α. In our algorithm $T_\alpha = \frac{\pi}{2}$.

A.3. Uniformity of Borders and Curves

The constraint of uniformity in the lengths of Delaunay edges along borders and curves is used at various points in the algorithm to narrow down the search for possible extensions of incomplete borders or curves. This occurs when scanning for possible paths for completing broken borders during connected component analysis, transition from nonempty to empty interior, merging deifferent components, and smoothing borders.

Given a border segment of n points (connected by n-1 Delaunay edges), the border uniformity measure is computed as follows. First we compute the average distance measure *average_distance* of the Delaunay edges in the border segment.

$$average\_distance = \frac{(\sum d_i) - d_{\max}}{(n-1)}$$

where d_i's are the lengths of the Delaunay edges in the border segment, and $d_{\max}$ is the length of the longest edge. Now, using this measure we compute the uniformity measure, *unif* , as follows.

$$unif = \frac{average\_distance}{d_{\max}}$$

The uniformity measure will be very close to 1.0 when the border segment consists of very regularly spaced dots. It will be very close to 0.0 if there is a very long edge in the border segment.

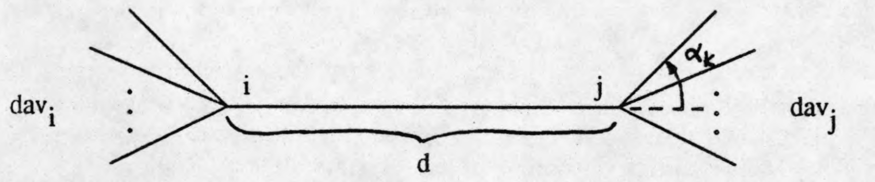


Figure 93. The neighbors considered for the dots i and j are only those with $\alpha_k < T_\alpha$. $T_\alpha = \frac{\pi}{2}$.

In the search for a Delaunay edge to extend the border segment, the neighbors of the endpoint under consideration are scanned (Figure 94). Using a neighbor to extend the border segment may change the uniformity measure of the extended segment. A neighbor is not considered for extension, if the updated uniformity measure would fall below a certain threshold, T_u. In our algorithm this threshold is set to 0.3.

A.4. Curvature of Borders and Curves

The search for Delaunay edges for extending a border or curve segment, as well as for merging components is also narrowed down by limiting the range of the angles in which the neighbors are searched (Figure 94). The thresholds used are as follows:

(i) In the border extension process, the neighbors of the endpoints are considered if $\cos(\alpha)>0.3$.

(ii) In the component merge process the possible merge points are searched from the corner points of the components (Figure 95). The average μ and variance σ^2 of the normalized angles α/π along a border segment are computed. If the neighbor j of the dot i has angle α_j, then the neighbor j is considered if $\mu-2\sigma<\alpha_j/\pi<\mu+2\sigma$. Otherwise, it is rejected.

A.5. Gabriel Threshold

In the processes of border smoothing (Figure 54) and component merge (Figure 95) just described, a new edge is not considered if its Gabriel measure is below a certain threshold, T_G. In our algorithm, T_G=0.9.

A.6. Corner Grabbing

In the corner grabbing process (Figure 52) a measure of distance uniformity and curvature is considered. Given a border segment and a dot as a possible corner point to be grabbed, we consider the four border segments shown in Figure 96.

The following measures are computed for these border segments: d11 – distance uniformity measure for border (a), c11 – average curvature measure for border (a). Similarly, d12, and c12 for border (b), d21 and c21 for border (c), and d22 and c22 for border (d) are computed. Using these we compute

$$\Delta_{11}=\frac{(1-c\,11)-(1-c\,12)}{\max((1-c\,11),(1-c\,12))}$$

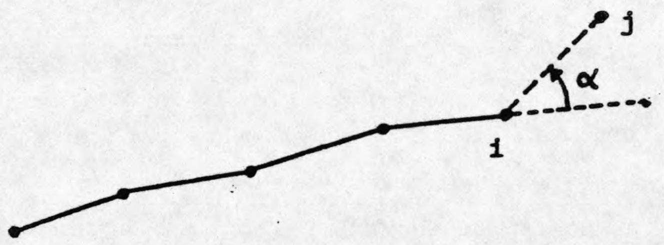


Figure 94. The neighbors (e.g., j) of the endpoint i are searched for possible extension of the border segment terminating at i. If the length of edge ij is much larger than the average edge length along the segment, then the point j is not considered for possible extension. There is also an upper limit to the the angle α for the neighbor j to be considered.

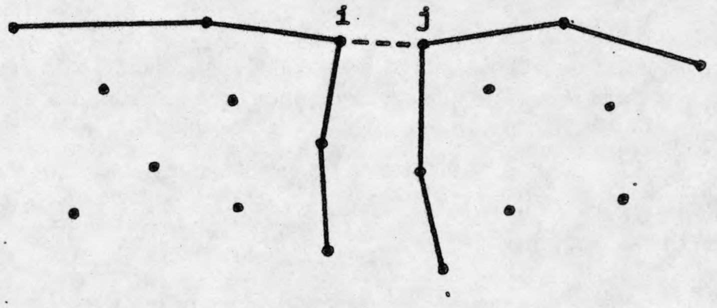


Figure 95. The search for the Delaunay edges that merge components. The neighbors (j) of the high curvature point i are searched for possible merge of the components to which i and j belong.

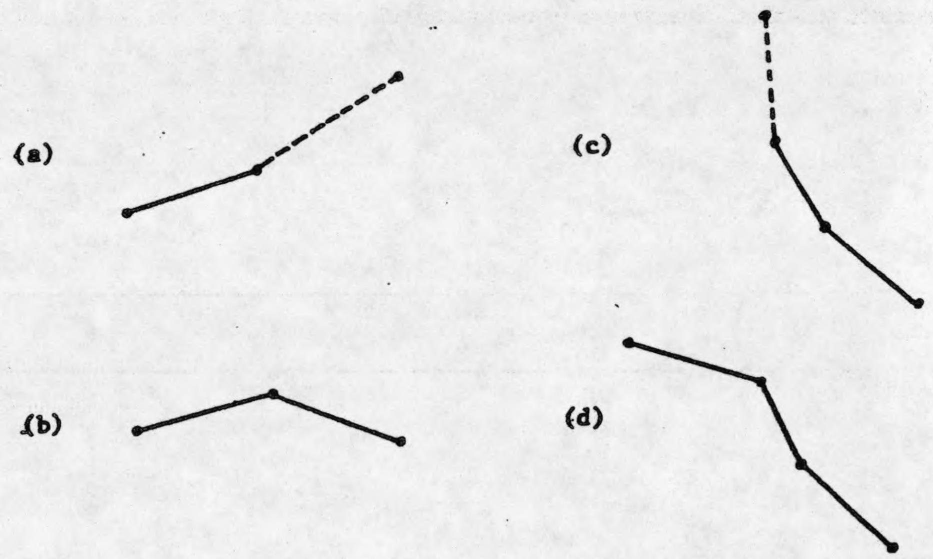


Figure 96. The four border segments considered for the corner grabbing context given in Figure 52.

$$\Delta_{13}=\frac{d11-d12}{\max(d11,d12)}$$

$$\Delta_{21}=\frac{(1-c21)-(1-c22)}{\max((1-c21),(1-c22))}$$

$$\Delta_{23}=\frac{d21-d22}{\max(d21,d22)}$$

Now, a total score, *total_score*, is computed as

$$total\_score = \Delta_{11}+\Delta_{13}+\Delta_{21}+\Delta_{23}$$

The corner grabbing takes place if $total\_score > 0$. That is, the overall distance uniformity and curvature measures for the new border (c11, d11, c21, d21) are better than the overall distance uniformity and curvature measures for the old border (c12, d12, c22, d22).

A.7. High Curvature Points

The component merge part of the algorithm searches for possible merge points with neighboring components only at high curvature points. Whether a point is a high curvature point or not is determined by comparing the angle at the given point with the angles of the points along the borders on its two sides (Figure 97).

The means, μ_1,μ_2, and variances, σ_1^2,σ_2^2, of the angles on the two sides of the dot are computed. Dot i is considered a high curvature point if $\alpha>\mu_1+3\sigma_1$ and $\alpha>\mu_2+3\sigma_2$.

A.8. Ellipse Fitting

The tokens corresponding to groups of dots are represented as a set of ellipses as described in Section 5.2. The measure *fit* represents the error of fit of an ellipse to a set of dots. If a given ellipse does not provide an acceptable fit to a set of dots, the set is divided into two and a new pair of ellipses is fit to the two groups. An ellipse accepted as a token if the value of fit is lower than a threshold, T_{fit}. In our algorithm, T_{fit}=4.0.

A.9. Implicit Thresholds

There are also implicit thresholds used in the preprocessing portion of the correction process (Step B in Figure 24). For example any single dot with label *NONINTERIOR* surrounded by all *INTERIOR* labeled dots, is converted to an *INTERIOR* dot. This can be considered an implicit threshold. Similarly, single dangling borders that are on long border segments are cleaned up. This is also an implicit threshold.

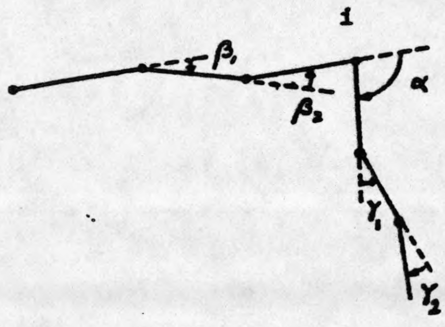


Figure 97. The dot i is a high curvature point if α is large compared to the average of β_i's and also the average of γ_i's.

APPENDIX B

COMPUTATION OF COMPATIBILITIES FOR LOWEST LEVEL GROUPING

In this appendix we present details of the computation of compatibility coefficients for the interior identification, border identification, and curve identification modules. The expressions for the compatibilities are taken from the actual C program code that implements the lowest level grouping. In the syntax used, the variable r[k][labeli][labelj] represents the compatibility coefficient for the label labeli on object i and the label labelj on object j. (The third index k is used by the program for efficient data access, and is of no relevance in the current discussion. Therefore, k may be ignored and r may be treated as a two-dimensional array with indexes labeli and labelj). The compatibilities are actually logical expressions that express the logical conditions that must hold in order for the compatibilities to be high. These various conditions involve measures computed from the Voronoi representation. The fuzzy logical operations *AND*, *OR*, and *NOT* are implemented as $AND(x,y)=min(x,y)$, $OR(x,y)=max(x,y)$, and $NOT(x)=1-x$. The computation is thus divided into two parts: (i) computation of various properties and measures necessary to define compatibilities, and (ii) the actual computation of the compatibility coefficients using these measures. Since the computations listed here directly follow the program, certain dummy are variables defined first and then used in the expressions for compatibility coefficients.

B.1. Interior Identification Compatibilities

The interior identification compatibilities are computed as given below. The compatibility coefficients for the point pair i and j are computed as r[k][labeli][labelj]. The labels for this module are INT for interior and NONINT for noninterior.

```
/* ----------------------------------------------------------------*/
/*
 * We first compute the various measures such as the area difference
 * which are later used for the computation of the compatibility coefficients.
 */
/* area difference measure */
dela = abs(area[i]-area[j])/max(area[i],area[j]);
/* elongations of cells i and j */
elongi = elong[i];
elongj = elong[j];
/* eccentricities */
ecci = ecc[i];
eccj = ecc[j];
/* eccentricity directions */
ecxi = vecx[i];
ecyi = vecy[i];
ecxj = vecx[j];
ecyj = vecy[j];
/* (vijx,vijy) = unit vector from point i to point j */
uvec(pt,i,j,vijx,vijy);
/*
 * dc = 1-cos(angle) where angle is between eccentricities i and j
 * normalized to the range [0,1]. It indicates the amount of
 * direction change.
 */
dc = 0.5*(1.-ecxi*ecxj-ecyi*ecyj);
/*
 * dpijj is the dot product of vector vij with eccentricity of
 * cell j (eccj). It indicates the degree of alignment of eccj
 * with the line joining dots i and j.
 */
dpijj = 0.5*(1.+vijx*ecxj+vijy*ecyj);
```

```
/* dpjii = the degree of alignment of ecci with the line joining j to i */
dpjii = 0.5*(1.-vijx*ecxi-vijy*ecyi);
/* xor is high when one eccentricity is low and the other is high */
xor = max(min(1.-ecci,eccj),min(ecci,1.-eccj));
/* this is the edge number of the Delaunay edge connnecting dots i and j */
ieno = edgeno[k];
/*
 * hbrdij is the measure of Delaunay edge being broken. This is true when
 * the gabriel measure is low OR distance measure is high.
 */
hbrdij = max(1-gabr[ieno],dist[ieno]);
/* w[i] is the non-isotropicity measure for cell i */
/*
 * ----------------------------------------------------------------*
 */
/*
 * Now we describe the computation of the compatibility coefficients.
 * The dummy variables temp1-temp4 are used temporarily to make the
 * expressions simpler.
 */
/* temp1 is high when both eccentricities are low */
temp1=min(1-ecci,1-eccj);
/*
 * temp2 is high when both eccentricities are high and their directions
 * are the same, (i.e. the direction change is small, dc is low)
 */
temp2=min(ecci,eccj,1-dc);
/*
 * temp3 is high when both eccentricities are low and area difference
 * is small.
 */
temp3=min(temp1,1-dela);
/* temp4 is high when both eccentricities are high and their directions
 * are similar (temp2) OR both eccentricities are low and the area
 * difference is low (temp3)
 */
temp4=max(temp2,temp3);
/*
 * Compatibility for the label INTERIOR for object i and the label
 * INTERIOR for object j being computed.
 */
/*
 * This will be high when temp4 is high AND
 * the Delaunay edge ij does not have a high breaking measure (1-hbrdij)
 * AND the cells i and j are not elongated (1-elongi, 1-elongj) AND
 * the cells i and j are isotropic (1-w[i], 1-w[j])
 */
r[k][INT][INT] = 2*min(temp4,1-hbrdij,1-elongi,1.-elongj,1.-w[i],1.-w[j])-1;
/*
 *---------------------------------------------------------------*
 */
/* the compatibility coefficient for i INTERIOR and j NONINTERIOR */
/* temp1 is high when: ecci is low, eccj is high, eccj is aligned with
 * the line ij, the Delaunay edge is not broken, and cell i is compact
 */
temp1=2*min(1.-ecci,eccj,1.-dpijj,1.-w[i],1-hbrdij)-1;
/* INT-NONINT compatibility is high when (one eccetricity is high and
 * the other low AND temp1 is high) OR (both eccentricities are either
 * low or high).
 */
r[k][INT][NONINT] = max(-xor,min(temp1,xor));
/*
 *--------------------------------------------------------------*
 */
/* i NONINTERIOR and j INTERIOR */
/* similar to above with i and j interchanged */
temp1=2*min(1.-eccj,ecci,1.-dpjii,1.-w[j],1-hbrdij)-1;
r[k][NONINT][INT] = max(-xor,min(temp1,xor));
/*
 *--------------------------------------------------------------*
 */
/* both i and j NONINTERIOR */
/*
 * temp1 is low when both eccentricities are low OR both are high
 * and the eccentricities are in the same direction.
 */
temp1 = 1-max(min(1-ecci,1-eccj),min(ecci,eccj,1-dc));
/*
 * temp2 is high when both eccentricities are high and there is a
 * direction change OR eccentricity i is low and eccentircity j is high
 * and eccentricity j points away from the point i OR same with i and
 * j swapped.
 */
temp2 = 2*max(min(ecci,eccj,dc),min(1-ecci,eccj,dpijj),
          min(ecci,1-eccj,dpjii))-1;
/*
 * temp3 is high when either temp1 condition is high and temp2 condition
 * is high OR temp1 condition is low.
 */
temp3 = max(-temp1,min(temp2,temp1));
/* temp4 is high when (cell i is either elongated OR nonisotropic)
 * AND (cell j is elongated OR nonisotropic).
 */
temp4 = 2*min(max(w[i],elongi),max(w[j],elongj))-1;
/* temp5 is high when the Delaunay edge is weak */
temp5 = 2*hbrdij-1;
/*
 * NONINTERIOR - NONINTERIOR compatibility is high when the conditions
 * temp3 OR temp4 OR temp5 are high.
 */
r[k][NONINT][NONINT] = max(temp3,temp4,temp5);
```

B.2. Border Identification Compatibilities

This section describes the computation of the border compatibility coefficients for the labels BORDER and NONBORDER, on objects that are Delaunay edges. Let i and j denote Delaunay edges connecting points (p1, p2) and (p3,p2) thus sharing the point p2. Now orient the figure such that p1,p2, and p3 appear in that order in a left-to-right scan. Then points m1, m2, denote the lateral neighbor points of edges i, j at the top, and points m3, m4 denote the lateral neighbors of edges at the bottom. In the following, first the computation of the various measures is given; these measures are later used in both the computation of border identification compatibilities and the curve identification comptibilities. The compatibilities are again computed in the array entries r[k][labeli][labelj].

```
/*
 * This section computes the various measures of the Delaunay edges and
 * the Voronoi cells that are later on used in the computation of the
 * compatibility coefficients.
 */
/*
 * The measures here are computed for one Delaunay edge i. The edge i
 * has the two endpoints p1 and p2. It also has two lateral neighbors,
 * m1 and m2.
 */
/* area1 and area2 are the cell areas of the two endpoints p1 and p2 */
area1=area[p1];
area2=area[p2];
/*
 * dar.d12 is the area difference for the cells of p1 and p2 defined
 * the same way as in the interior identification module.
 */
dar.d12=abs(area1-area2)/max(area1,area2);
/* compute hbrd measure same as in interior identification module. */
hbrd.hi=max(dist[i],1-gabr[i]);
/* compute the vectors */
uvec(loc,p1,p2,v12x,v12y);
e1=ecc[p1]; e2=ecc[p2];
/*
 * compute the various dot products. These are measures of alignment
 * between various vectors computed as the cosine of the
 * angle between them. For example, dp.e1e2 is the dot product of the
 * two eccentricity vectors e1 and e2 which belong to the cells
 * of p1 and p2. idp.e1e2 is the negative of the dot product of the
 * two vectors. These are meaningful only if the eccentricity magnitudes
 * are high.
 */
temp=0.5*(1+vecx[p1]*vecx[p2]+vecy[p1]*vecy[p2])
dp.e1e2=min(e1,e2,temp); idp.e1e2=min(e1,e2,1-temp)
temp=0.5*(1+vecx[p1]*v12x+vecy[p1]*v12y)
dp.e1v12=min(e1,temp); idp.e1v12=min(e1,1-temp)
temp=0.5*(1+vecx[p2]*v12x+vecy[p2]*v12y)
dp.e2v12=min(e2,temp); idp.e2v12=min(e2,1-temp)
temp=abs(minorx[p1]*v12x+minory[p1]*v12y)
dp.min112=min(elong[p1],temp); idp.min112=min(elong[p1],1-temp)
temp=abs(minorx[p2]*v12x+minory[p2]*v12y)
dp.min212=min(elong[p2],temp); idp.min212=min(elong[p2],1-temp)
/* ---------------------------------------------------------------- */
/* Measures for the top half of the edge <p1,p2,m1> */
/*
 * These measures compute the area differences between p1, p2, m1.
 */
aream1=area[m1];
dar.d1m1=abs(area1-aream1)/max(area1,aream1);
dar.d2m1=abs(area2-aream1)/max(area2,aream1);
/*
 * Compute hbrd measures between p1, p2, m1.
 */
hbrd.h1=max(dist[d1],1-gabr[d1]);
hbrd.h2=max(dist[d2],1-gabr[d2]);
/*
 * compute the vectors p1 to m1, p2 to m1.
 */
uvec(loc,p1,m1,v1m1x,v1m1y);
uvec(loc,p2,m1,v2m1x,v2m1y);
em1=ecc[m1];
/*
 * Once again, the various alignment measures among the eccentricities
 * and lines joining points p1, p2, m1, and m2 in terms of dot products.
 */
temp=0.5*(1+vecx[p1]*v1m1x+vecy[p1]*v1m1y);
dp.e1v1m1=min(e1,temp); idp.e1v1m1=min(e1,1-temp);
temp=0.5*(1+vecx[p2]*v2m1x+vecy[p2]*v2m1y);
dp.e2v2m1=min(e2,temp); idp.e2v2m1=min(e2,1-temp);
temp=0.5*(1+vecx[m1]*v1m1x+vecy[m1]*v1m1y);
dp.em1v1m1=min(em1,temp); idp.em1v1m1=min(em1,1-temp);
temp=0.5*(1+vecx[m1]*v2m1x+vecy[m1]*v2m1y);
dp.em1v2m1=min(em1,temp); idp.em1v2m1=min(em1,1-temp);
temp=0.5*(1+vecx[p1]*vecx[m1]+vecy[p1]*vecy[m1]);
dp.e1em1=min(e1,em1,temp); idp.e1em1=min(e1,em1,1-temp);
temp=0.5*(1+vecx[p2]*vecx[m1]+vecy[p2]*vecy[m1]);
dp.e2em1=min(e2,em1,temp); idp.e2em1=min(e2,em1,1-temp);
temp=abs(minorx[m1]*v1m1x+minory[m1]*v1m1y);
dp.minm11m1=min(elong[m1],temp); idp.minm11m1=min(elong[m1],1-temp);
temp=abs(minorx[m1]*v2m1x+minory[m1]*v2m1y);
dp.minm12m1=min(elong[m1],temp); idp.minm12m1=min(elong[m1],1-temp);
/*
 * (a.top) This measure computes how distorted the triangle p1,p2,m1 is.
 * If highly distorted, then probably the two edges p1,p2,m1 or m1,p1,p2
 * are on a border.
 */
ddp1 = v12x*v1m1x+v12y*v1m1y;
ddp2 = v12x*v2m1x+v12y*v2m1y;
a.top=max(0.,-ddp1,ddp2);
/*-----------------------------------------------------------------*/
/* Measures for the bottom half. <p1,p2,m2> */
/*
 * We go through the same computations for the bottom of the two edges
 * which includes now the point m2.
 */
aream2=area[m2];
dar.d1m2=abs(area1-aream2)/max(area1,aream2);
dar.d2m2=abs(area2-aream2)/max(area2,aream2);
/*
 * compute hbrd measures
 */
hbrd.h3=max(dist[d3],1-gabr[d3]);
hbrd.h4=max(dist[d4],1-gabr[d4]);
/*
 * compute the vectors from p1 to m2 and p2 to m2.
 */
uvec(loc,p1,m2,v1m2x,v1m2y);
uvec(loc,p2,m2,v2m2x,v2m2y);
em2=ecc[m2];
/*
 * Compute the various alignment measures for the bottom half of the edge.
 */
temp=0.5*(1+vecx[p1]*v1m2x+vecy[p1]*v1m2y);
dp.e1v1m2=min(e1,temp); idp.e1v1m2=min(e1,1-temp);
```

```
temp=0.5*(1+vecx[p2]*v2m2x+vecy[p2]*v2m2y);
dp.e2v2m2=min(e2,temp); idp.e2v2m2=min(e2,1-temp);
temp=0.5*(1+vecx[m2]*v1m2x+vecy[m2]*v1m2y);
dp.em2v1m2=min(em2,temp); idp.em2v1m2=min(em2,1-temp);
temp=0.5*(1+vecx[m2]*v2m2x+vecy[m2]*v2m2y);
dp.em2v2m2=min(em2,temp); idp.em2v2m2=min(em2,1-temp);
temp=0.5*(1+vecx[p1]*vecx[m2]+vecy[p1]*vecy[m2]);
dp.e1em2=min(e1,em2,temp); idp.e1em2=min(e1,em2,1-temp);
temp=0.5*(1+vecx[p2]*vecx[m2]+vecy[p2]*vecy[m2]);
dp.e2em2=min(e2,em2,temp); idp.e2em2=min(e2,em2,1-temp);
temp=abs(minorx[m2]*v1m2x+minory[m2]*v1m2y);
dp.minm21m2=min(elong[m2],temp); idp.minm21m2=min(elong[m2],1-temp);
temp=abs(minorx[m2]*v2m2x+minory[m2]*v2m2y);
dp.minm22m2=min(elong[m2],temp); idp.minm22m2=min(elong[m2],1-temp);
/*
 * (a.bot) compute the distortion of the bottom triangle
 */
ddp3 = v12x*v1m2x+v12y*v1m2y;
ddp4 = v12x*v2m2x+v12y*v2m2y;
a.bot=max(0.,-ddp3,ddp4);
/*--------------------------------------------------------------*/
/*
 * Compute the various measures for the cases in which the Delaunay
 * edge i can be in a NONBORDER context.
 */
/*
 * inside a uniform cluster
 */
tt1=min(1-ecc[p1],1-ecc[p2],1-dar.d12);
tt2=min(1-dar.d1m2,1-dar.d2m2);
tt3=min(1-dar.d1m1,1-dar.d2m1);
/*
 * m1 on the border.
 */
tt4=min(tt1,tt2,1-ecc[m2],max(idp.em1v1m1,idp.em1v2m1));
/*
 * m2 on the border
 */
tt5=min(tt1,tt3,1-ecc[m1],max(idp.em2v1m2,idp.em2v2m2));
/*
 * everything inside.
 */
tt6=min(tt1,tt2,tt3,1-ecc[m1],1-ecc[m2]);
t1=max(tt4,tt5,tt6);
/*
 * eccentricities opposing and the edge is not squeezed
 */
tt1=min(idp.e1e2,1-sqms(i));
/*
 * Either tt1 is true or hbrd measure is high.
 */
t2=max(tt1,hbrd.hi);
/*
 * between two clusters, eccentricities pointing away from the edge.
 */
tt1=min(1-ecc[p1],dp.e2v12);
tt2=min(1-ecc[p2],idp.e1v12);   /* symmetric case for the above */
```

```
/*
 * Either above conditions are true or hbrd measure is high
 */
t3=max(tt1,tt2,hbrd.hi);
/*
 * Inside uniform cluster one of the endpoints on the border.
 */
t4=min(1-ecc[p2],dp.e1v12);
t8=min(1-ecc[p1],idp.e2v12); /* symmetric case for the above */
/*
 * inside nonuniform cluster.
 */
/* inside completely */
t12=min(1-dar.d12,1-dar.d1m1,1-dar.d1m2,1-dar.d2m1,1-dar.d2m2,
     dp.e1e2,dp.e1em1,dp.e1em2,dp.e2em1,dp.e2em2);
t13=min(ecc[m1],1-dar.d1m2,1-dar.d2m2,
       dp.e1e2,dp.e1em2,dp.e2em2);          /* m1 on the border */
t14=min(ecc[m2],1-dar.d1m1,1-dar.d2m1,
       dp.e1e2,dp.e1em1,dp.e2em1);          /* m2 on the border */
/*
 * p1 on the border p2 inside
 */
t15=min(ecc[p1],1-dar.d2m1,1-dar.d2m2,
       dp.e2em1,dp.e2em2);        /* m1 and m2 inside */
t16=min(ecc[p1],ecc[m1],1-dar.d2m2,
       dp.e2em2);          /* m1 on the border, m2 inside */
t17=min(ecc[p1],ecc[m2],1-dar.d2m1,
       dp.e2em1);          /* m2 on the border, m1 inside */
/*
 * p2 on the border p1 inside (symmetric of previous ones)
 */
t18=min(ecc[p2],1-dar.d1m1,1-dar.d1m2,
       dp.e1em1,dp.e1em2);        /* m1 and m2 inside */
t19=min(ecc[p2],ecc[m1],1-dar.d1m2,
       dp.e1em2);          /* m1 on the border, m2 inside */
t20=min(ecc[p2],ecc[m2],1-dar.d1m1,
       dp.e1em1);          /* m2 on the border, m1 inside */
t21=1-max(hbrd.h1,hbrd.h2,hbrd.h3,hbrd.h4,a.top,a.bot);

/*
 * nonbord - The overall measure of the edge i being nonborder.
 * temp1 is a temporary dummy variable.
 */
term1=min(max(t1,t4,t8,t12,t13,t14,t15,t16,t17,t18,t19,t20),
         1-hbrd.hi,t21);
nonbord=max(t2,t3,term1);
/*-----------------------------------------------------------------*/
/*
 * The measure indicating the likelihood that the edge i is
 * on a border and its top half (Delaunay edges <p1,m1> and <p2,m1>)
 * is in the intercluster crack.
 */
/*
 * em1 is pointing away from <p1,m1> and edge <p1,m1> has a high
 * likelihood of being broken.
 */
t1=max(dp.em1v1m1,hbrd.h1);
```

```
t2=max(dp.em1v2m1,hbrd.h2); /* same for <p2,m1> */
brk1=max(min(t1,t2),a.top); /* top isin the crack */
t1=max(idp.e1em1,hbrd.h1);  /* em1 pointing toward p1 */
t2=max(idp.e2em1,hbrd.h2);  /* em1 pointing toward p2 */
brk2=max(min(t1,t2),a.top); /* another way top may be broken. */
/* third way top may be broken. */
brk3=max(brk2,min(1-ecc[p1],1-ecc[p2],brk1));
t1 = min(1-ecc[m1],1-ecc[m2],dp.e1v1m2,dp.e2v2m2);
/*
 * overall measure that i is border and top is in the intercluster crack.
 */
brdtop=min(max(t1,brk3),1-hbrd.hi);

/*-----------------------------------------------------------------*/
/*
 * The measure indicating the likelihood that the edge i is
 * on a border and its bottom half (Delaunay edges <p1,m2> and <p2,m2>)
 * is in the intercluster crack.
 */
/* measures similar to the computation for the tops */
t1=max(dp.em2v1m2,hbrd.h3);
t2=max(dp.em2v2m2,hbrd.h4);
brk1=max(min(t1,t2),a.bot);
t1=max(idp.e1em2,hbrd.h3);
t2=max(idp.e2em2,hbrd.h4);
brk2=max(min(t1,t2),a.bot);
brk3=max(brk2,min(1-ecc[p1],1-ecc[p2],brk1));
t1 = min(1-ecc[m1],1-ecc[m2],dp.e1v1m1,dp.e2v2m1);
/*
 * Overall measure that edge i is border and its bottom half
 * is in the intercluster crack.
 */
brdbot=min(max(t1,brk3),1-hbrd.hi);

/*-----------------------------------------------------------------*/
/*
 * Now using the above computed measures, we compute the compatibilities.
 */
/* compute the measure of edge i being nonbord */
brdmsi.nonbord=brdms[i].nonbord;
/* edge i has top lateral neighbors broken */
brdmsi.top=brdms[i].top;
/* edge i has bottom lateral neighbors broken */
brdmsi.bottom=brdms[i].bottom;
/* the same computations for edge j */
brdmsj.nonbord=brdms[j].nonbord;
brdmsj.top=brdms[j].top;
brdmsj.bottom=brdms[j].bottom;
/*-----------------------------------------------------------------*/
/*
 * Compatibility coefficient for the label pair NONBORDER-NONBORDER
 * for object pair i-j.
 */
/* both nonborder measures for edges i and j must be high. */
temp=min(brdmsi.nonbord,brdmsj.nonbord);
r[k][NBORD][NBORD]=2*temp-1;
/*-----------------------------------------------------------------*/
```

```
/*
 * NONBORDER-BORDER compatibility coefficient.
 * Edge i has high nonborder measure, AND
 * Edge j has high measure of being border by either having
 * its top lie on a crack OR its bottom.
 */
temp=min(brdmsi.nonbord,max(brdmsj.top,brdmsj.bottom));
r[k][NBORD][BORD]=2*temp-1;
/*-----------------------------------------------------------------*/
/*
 * BORDER-NONBORDER, similar to NONBORDER-BORDER with i and j interchanged.
 */
temp=min(max(brdmsi.top,brdmsi.bottom),brdmsj.nonbord);
r[k][BORD][NBORD]=2*temp-1;
/*-----------------------------------------------------------------*/
/*
 * Both edges are BORDER.
 * This will be true only if the top halves of both edges lie on
 * a crack OR the bottoms of both edges lie on a crack.
 */
temp=max(min(brdmsi.top,brdmsj.top),min(brdmsi.bottom,brdmsj.bottom));
r[k][BORD][BORD]=2*temp-1;
```

B.3. Curve Identification Compatibilities

The curve identification compatibilities are computed for labels CURV and NCURV, on the objects that are Delaunay edges. The compatibilities are again assigned to the variables r[k][labeli][labelj]. The measures and definitions used here are the same as used in the border compatibilities. Therefore, their computation is not described below.

```
/*-----------------------------------------------------------------*/
/*
 * Compute the curve identification compatibility coefficients.
 */
/*
 * crv and noncrv measures are computed similar to the nonborder,
 * top, and bottom measures given in the border identification
 * compatibilities.
 */
/*
 * compute the measure that edge i is noncurv
 */
/* various measures of vector alignment are computed. */
t1=min(idp.min112,idp.min212);
t2=max(idp.e1v12,dp.e2v12);
/* sqms[i] is the squeezedness measure for edge i */
t3=max(sqms[d1],sqms[d2],sqms[d3],sqms[d4]);
tt1=max(a.top,min(hbrd.h1,hbrd.h2));
tt2=max(a.bot,min(hbrd.h3,hbrd.h4));
tt3=min(tt1,tt2);
t4=1-max(tt3,sqms[i]);
noncrv=max(hbrd.hi,t1,t2,t3,t4);
/*
 * compute the measure that an edge is crv
 */
/*
```

```
 * measures for the top half.
 */
tt1=min(hbrd.h1,hbrd.h2);
tt2=min(1-sqms[d1],1-sqms[d2]);
tt3=max(dp.em1v1m1,dp.em1v2m1);
tt4=max(idp.minm11m1,idp.minm12m1);
top=max(tt1,tt2,tt3,tt4,a.top);
/*
 * measures for the bottom half
 */
tt1=min(hbrd.h3,hbrd.h4);
tt2=min(1-sqms[d3],1-sqms[d4]);
tt3=max(dp.em2v1m2,dp.em2v2m2);
tt4=max(idp.minm21m2,idp.minm22m2);
bot=max(tt1,tt2,tt3,tt4,a.bot);
/* Look at the three cases */
/* p1 termination of a curv */
t1=min(top,bot,dp.e1v12,dp.min212,1-hbrd.hi);
/* p1 and p2 in the middle of a curve */
t2=min(top,bot,dp.min112,dp.min212,1-hbrd.hi);
/* p2 termination of a curve */
t3=min(top,bot,idp.e2v12,dp.min112,1-hbrd.hi);
/* p1, p2 a pair of dots */
t4=min(top,bot,dp.e1v12,idp.e2v12,1-hbrd.hi);
/*
 * The final curve measure for the edge i */
 */
crv=max(t1,t2,t3,t4,sqms[i]);

brdmsi.noncrv=brdms[i].noncrv;
brdmsi.crv=brdms[i].crv;

brdmsj.noncrv=brdms[j].noncrv;
brdmsj.crv=brdms[j].crv;

/*--------------------------------------------------------------*/
/*
 * Compatibilities for the labels NONCURVE-NONCURVE.
 * Noncurve measure must be high for both edges.
 */
temp=min(brdmsi.noncrv,brdmsj.noncrv);
r[k][NCURV][NCURV]=2*temp-1;
/*--------------------------------------------------------------*/
/*
 * edge i is noncurve and edge j is curve.
 */
temp=min(brdmsi.noncrv,brdmsj.crv);
r[k][NCURV][CURV]=2*temp-1;
/*--------------------------------------------------------------*/
/*
 * edge i is curve and edge j is noncurve.
 */
temp=min(brdmsi.crv,brdmsj.noncrv);
r[k][CURV][NCURV]=2*temp-1;
/*--------------------------------------------------------------*/
/* both edges have high curve measures */
temp=min(brdmsi.crv,brdmsj.crv);
```

```
r[k][CURV][CURV]=2*temp-1;
```

APPENDIX C

COMPUTATION OF COMPATIBILITIES FOR HIERARCHICAL GROUPING

The syntax used below is similar to that used in Appensix A. The following sections describe the computation of compatibilities for each of the three types of hierarchical grouping. In each case the objects used are the Delaunay edges connecting pairs of tokens. The labels are BRK and NTBREAK. Compatibilities are computed for pairs of Delaunay edges, such that they share a token, similar to the cases of border and curve identification in the lowest level grouping.

C.1. Proximity Grouping Compatibilities

The computation of proximity grouping is given below. The two measures dempty[i] and meas[i] are computed first. The measure dempty[i] is the distance between the two tokens at the endpoints of the edge i which is normalized with respect to the token sizes.

```
/*---------------------------------------------------------------*/
/* First compute the various measures that are used in the
 * computation of the compatibility coefficients.
 */
/*
 * computation of dempty = separation of two tokens, tp1 and tp2,
 * at the endpoints of the edge i.
 * a1 and b1 are the major and minor axis lengths for the
 * ellipse belonging to token tp1.
 * a2 and b2 are the major and minor axis lengths for the ellipse
 * belonging to token tp2.
 * th1 and th2 are the orientations of the ellipses of tp1 and tp2.
 */
tp1 = segendpts[i].pt1;
tp2 = segendpts[i].pt2;
x01 = segments[tp1].xbar;
y01 = segments[tp1].ybar;
x02 = segments[tp2].xbar;
y02 = segments[tp2].ybar;
tdist = hypot((x02 - x01), (y02 - y01));
a1 = segments[tp1].a;
b1 = segments[tp1].b;
th1 = segments[tp1].theta;
a2 = segments[tp2].a;
b2 = segments[tp2].b;
th2 = segments[tp2].theta;
/* distance between (x01,y01) and (x02,y02) */
dempty[i] = tdist;
dempty[i] = dempty[i]/(a1+a2);
/*---------------------------------------------------------------*/
/* two token ellipses tp1 and tp2 at the endpoints of edge i */
tp1 = segendpts[i].pt1;
tp2 = segendpts[i].pt2;
dempty1 = dempty[i];                    /* dempty computed below */
/*
 * compute distance measure compared with the average distances,
 * meas[i], around the edge i.
 */
dm1 = (meas[i] - dempty1) / max(dempty1, meas[i]);
/* tp3 and tp4 are the endpoints of edge j */
```

```
tp3 = segendpts[j].pt1;
tp4 = segendpts[j].pt2;
dempty2 = dempty[j];
/* compute the distance measure for edge j. */
dm2 = (meas[j] - dempty2) / max(dempty2, meas[j]);
dmax = max(dempty1, dempty2);
/*
 * compare the lengths of the two distance measures for edges i and j.
 * m12 is high when edge j is longer than edge i. m21 is high when
 * edge i is longer than edge j. deldm is the absolute value of the
 * difference of the lengths normalized to the range [0,1].
 */
m12 = max(0.0, (dempty2 - dempty1) / dmax);
m21 = max(0.0, (dempty1 - dempty2) / dmax);
deldm = fabs(dempty1 - dempty2) / dmax;

/*---------------------------------------------------------------*/
/*
 * Now compute the compatibilities.
 */
/*---------------------------------------------------------------*/
/* both edges are short compared to their neighboring edges. */
temp = 0.5*(0.5 * (dm1 + dm2)+1);
r[k][NTBRK][NTBRK] = 2*temp*temp-1;
/*---------------------------------------------------------------*/
/*
 * NOTBREAK-BREAK labels. The edge i is short and edge j is long
 * compared to its surroundings OR edge i is shorter than edge j.
 */
temp = 0.5*(1+max(0.5 * (dm1 - dm2), (m12 - m21)));
r[k][NTBRK][BRK] = 2*temp*temp-1;
/*---------------------------------------------------------------*/
/*
 * BREAK-NOTBREAK labels. Similar to NOTBREAK-BREAK with edges i and
 * j interchanged.
 */
temp = 0.5*(1+max(0.5 * (-dm1 + dm2), (m21 - m12)));
r[k][BRK][NTBRK] = 2*temp*temp-1;
/*---------------------------------------------------------------*/
/*
 * BREAK-BREAK. Both edges are long compared to their surrounding edges.
 */
temp = 0.5*(1+0.5 * (-dm1 - dm2));
r[k][BRK][BRK] = 2*temp*temp-1;
```

C.2. Orientation Grouping Compatibilities

```
/*---------------------------------------------------------------*/
/* endpoint tokens tp1 and tp2 for edge i */
tp1 = segendpts[i].pt1;
tp2 = segendpts[i].pt2;
/* the orientation angles and elongations for tp1 and tp2 */
theta1 = segments[tp1].theta;
theta2 = segments[tp2].theta;
elong1 = segments[tp1].elong;
```

```
elong2 = segments[tp2].elong;
/*
 * contr1 = contribution from edge i is high either (when both tokens
 * are elongeted AND their orientations are aligned) OR (when both
 * tokens are round).
 */
temp = min(elong1,elong2);
contr1 = max(min(temp,(1 - fabs(sin(theta1 - theta2)))),
         min(1-elong1,1-elong2));
/* endpoints tp3 and tp4 for edge j */
tp3 = segendpts[j].pt1;
tp4 = segendpts[j].pt2;
/* orientation angles and elongations for tp3 and tp4 */
theta1 = segments[tp3].theta;
theta2 = segments[tp4].theta;
elong1 = segments[tp3].elong;
elong2 = segments[tp4].elong;
/*
 * contr2 = contribution from edge j is high either (when both tokens
 * are elongeted AND their orientations are aligned) OR (when both
 * tokens are round).
 */
temp = min(elong1,elong2);
contr2 = max(min(temp,(1 - fabs(sin(theta1 - theta2)))),
         min(1-elong1,1-elong2));
/*---------------------------------------------------------------*/
/*
 * combine the contributions to obtain the compatibility coefficients
 */
/*---------------------------------------------------------------*/
/* all three tokens must be aligned or all three must be round. */
temp = min(contr1, contr2);
r[k][NTBRK][NTBRK] = 2.0 * temp - 1.0;
/*---------------------------------------------------------------*/
/* (first two tokens are aligned or round) AND (the second and third are
 * not).
 */
r[k][NTBRK][BRK] = 2.0 * min(contr1, 1.0 - contr2) - 1.0;
/*---------------------------------------------------------------*/
/* same as NOTBREAK-BREAK with edges i and j interchanged. */
r[k][BRK][NTBRK] = 2.0 * min(1.0 - contr1, contr2) - 1.0;
/*---------------------------------------------------------------*/
/* None of the three are aligned and not all three are round. */
r[k][BRK][BRK] = 2.0 * min(1.0 - contr1, 1.0 - contr2) - 1.0;
```

C.3. Size Grouping Compatibilities

```
/*---------------------------------------------------------------*/
/* First compute the sizes for the various tokens. */
/* endpoints tp1 and tp2 for edge i */
tp1 = segendpts[i].pt1;
tp2 = segendpts[i].pt2;
/* major and minor axes for the ellipses */
a = segments[tp1].a;
b = segments[tp1].b;
```

```
area1 = PI * a * b;                        /* area for ellipse tp1 */
a = segments[tp2].a;
b = segments[tp2].b;
area2 = PI * a * b;                        /* area for ellipse tp2 */
if (area1 == 0.0 && area2 == 0.0) {
/* If areas are 0 then take the length of the major axis as size.
 * This means for curves consider the length as the size.
 */
   area1 = 2.0 * segments[tp1].a;
   area2 = 2.0 * segments[tp2].a;
};
/* Compute the size difference measures between tokens tp1 and tp2 */
if (area1 > 0.0 || area2 > 0.0) {
   darea12 = fabs(area1 - area2) / max(area1, area2);
}
else {
   darea12 = 0.0;
};
/*
 * same computations as above for edge j with endpoint
 * tokens tp3 and tp4
 */
tp3 = segendpts[j].pt1;
tp4 = segendpts[j].pt2;
/* major and minor axes */
a = segments[tp3].a;
b = segments[tp3].b;
/* compute areas of tokens */
area3 = PI * a * b;
a = segments[tp4].a;
b = segments[tp4].b;
area4 = PI * a * b;
if (area3 == 0.0 && area4 == 0.0) {
/* if curves, then the sizes are lengths. */
   area3 = 2.0 * segments[tp3].a;
   area4 = 2.0 * segments[tp4].a;
};
if (area3 > 0.0 || area4 > 0.0) {
/* if tokens are dots then sizes are 0 */
   darea34 = fabs(area3 - area4) / max(area3, area4);
}
else {
   darea34 = 0.0;
};
/*---------------------------------------------------------------*/
/* Compute the compatibility coefficients */
/*---------------------------------------------------------------*/
/* For the three tokens to be grouped (i.e. the two edges i and j
 * not to be broken) the size differences among all three tokens must
 * be small.
 */
r[k][NTBRK][NTBRK] = 2.0 * min(1.0 - darea12, 1.0 - darea34) - 1.0;
/*---------------------------------------------------------------*/
/*
 * Edge j is broken if size difference for the two endpoints of edge i
 * is small but the size difference for the two endpoints of the
 * edge j is large.
 */
r[k][NTBRK][BRK] = 2.0 * min(1.0 - darea12, darea34) - 1.0;
/*-----------------------------------------------------------------*/
/*
 * Same as NOTBREAK-BREAK with edges i and j interchanged.
 */
r[k][BRK][NTBRK] = 2.0 * min(darea12, 1.0 - darea34) - 1.0;
/*-----------------------------------------------------------------*/
/*
 * If size differences are large for both edges, then both edges
 * are broken and none of the tokens are grouped.
 */
r[k][BRK][BRK] = 2.0 * min(darea12, darea34) - 1.0;
```

REFERENCES

1. Ahuja, N., "Dot Pattern Processing Using Voronoi Neighborhoods," *IEEE Trans. on Pattern Analysis and Machine Intelligence* **PAMI-4**(3) pp. 336-343 (May 1982).
2. Ahuja, N. and Schachter, B., *Pattern Models*, John Wiley and Sons., Inc. (1983).
3. Barlow, H. B. and Reeves, B. C., "The versatility and absolute efficiency of detecting mirror symmetry in random dot displays," *Vision Research* **19** pp. 783-793 (1979).
4. Blum, H., "Biological Shape and Visual Science (Part I)," *Journal of Theoretical Biology* **38** pp. 205-287 (1973).
5. Borjesson, E. and Hofsten, C. von, "Spatial determinants of depth perception in two-dot motion patterns," *Perception and Psychophysics* **11**(4) pp. 263-268 (1972).
6. Borjesson, E. and Hofsten, C. von, "Visual perception of motion in depth: Application of a vector model to three-dot motion patterns," *Perception and Psychophysics* **13**(2) pp. 169-179 (1973).
7. Dubes, R. and Jain, A. K., "Clustering Techniques: The User's Dilemma," *Pattern Recognition* **8** pp. 247-260 (1976).
8. Duda, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis*, John Wiley and Sons, Inc., New York (1973).
9. Fairfield, J., "Contoured Shape Generation: Forms that People See in Dot Patterns," *Proc. IEEE Conf. on Systems, Man, and Cybernetics*, pp. 60-64 (Oct 8-10, 1979).
10. Glass, L., "Moire effect from random dots," *Nature* **223** pp. 578-580 (1969).
11. Glass, L. and Perez, R., "Perception of random dot interference patterns," *Nature* **246** pp. 360-362 (1973).
12. Glass, L. and Switkes, E., "Pattern recognition in humans: correlations which cannot be perceived," *Perception* **5** pp. 67-72 (1976).
13. Green, P. J. and Sibson, R., "Computing Dirichlet Tessellations in the Plane," *The Computer Journal* **21** pp. 168-173 (1978).
14. Hu, M. K., "Visual Pattern Recognition by Moment Invariants," *IRE Trans. on Information Theory* **IT-8** pp. 179-187 (1962).
15. Jenkins, W., "Redundancy in the perception of bilateral symmetry in dot textures," *Perception and Psychophysics* **32**(2) pp. 171-177 (1982).
16. Kanizsa, G., "Subjective Contours," *Scientific American* **234** pp. 48-52 (April 1976).
17. Koffka, K., *Principles of Gestalt Psychology*, Hartcourt, Brace, New York (1935).
18. Lee, D. T., "Proximity and Reachability in the Plane," Ph.D. dissertation, CSL Report ACT-12, University of Illinois at Urbana-Champaign (1978).
19. Lowe, D. G., "Perceptual Organization and Visual Recognition," Ph.D. Thesis, Stanford University (Sep. 1984).
20. Marr, D., *Vision*, W. H. Freeman and Company, San Francisco, CA (1982).
21. Marroquin, J. L., "Human Visual Perception of Structure," Master's Thesis, Massachusetts Institute of Technology (1976).
22. O'Callaghan, J. F., "Computing the Perceptual Boundaries of Dot Patterns," *Computer Graphics and Image Processing* **3** pp. 141-162 (1974).
23. O'Callaghan, J. F., "An Alternative Definition for 'Neighborhood of a Point'," *IEEE Trans. on Computers* **C-24** pp. 1121-1125 (1975).
24. Patrick, E. A. and Shen, L., "Interactive use of problem knowledge for clustering and decision making," *IEEE Trans. on Computers* **C-20** pp. 216-222 (Febuary 1971).
25. Rock, I., *The Logic of Perception*, MIT Press, Cambridge, MA (1983).
26. Rosenfeld, A., Hummel, R., and Zucker, S., "Scene Labeling by Relaxation Operations," *IEEE Trans. on Systems, Man, and Cybernetics* **SMC-6** pp. 420-433 (1976).

27. Shamos, M. I. and Hoey, D., "Closest-point Problems," *Proc. 16th Annual Symp. on Foundations of Computer Science*, pp. 131-162 (October 1975).

28. Toussaint, G. T., "The Relative Neighborhood Graph of a Finite Planar Set," *Pattern Recognition* **12** pp. 261-268 (1980).

29. Tuceryan, M. and Ahuja, N., "Extracting Perceptual Structure in Dot Patterns," *Proc. IEEE Workshop on Languages for Automation: Cognitive Aspects in Information Processing*, (June 28-29, 1985).

30. Urquhart, R. B., "Graph Theoretical Clustering Based on Limited Neighborhood Sets," *Pattern Recognition* **15**(3) pp. 173-187 (1982).

31. Uttal, W. R., Bunnell, L. M., and Corwin, S., "On the detectability of straight lines in the visual noise: An extension of French's paradigm into the millisecond domain," *Perception and Psychophysics* **8**(6) pp. 385-388 (1970).

32. Velasco, F. R. D., "A Method for the Analysis of Gaussian-like Clusters," *Pattern Recognition* **12** pp. 381-393 (1980).

33. Vistnes, R., "Detecting Structure in Random-Dot Patterns," *Proc. of Workshop on Image Understanding, DARPA*, pp. 350-362 (Dec. 1985).

34. Voronoi, G., "Nouvelles applications des parametres continus a la theorie des formes quadratiques. Deuxieme memoire: Recherches sur les parallelloedres primitifs.," *J. Reine Angew. Math.* **134** pp. 198-287 (1908).

35. Wertheimer, M., "Laws of Organization in Perceptual Forms," in *A Source Book of Gestalt Psychology*, W. D. Ellis (ed.), Hartcourt, Brace, New York, pp. 71-88 (1938).

36. Wilson, H. B., Jr. and Farrior, D. S., "Computation of Geometrical and Inertial Properties for General Areas and Volumes of Revolution," *Computer Aided Design* **8**(4) pp. 257-263 (1976).

37. Witkin, A. P. and Tenenbaum, J. M., "On the Role of Structure in Vision," in *Human and Machine Vision*, Jacob Beck, Barbara Hope and Azriel Rosenfeld (ed.), Academic Press, New York, pp. 481-543 (1983).

38. Zahn, C. T., "Graph Theoretical methods for detecting and describing Gestalt clusters," *IEEE Trans. on Computers* **C-20** pp. 68-86 (1971).

39. Zucker, S. W. and Hummel, R. A., "Toward a Low-level Description of Dot Clusters: Labeling Edge, Interior and Noise Points," *Computer Graphics and Image Processing* **9** pp. 213-233 (1979).

40. Zucker, S. W., "Early Orientation Selection and Grouping: Type I and Type II Processes," McGill University Technical Report 82-6, Montreal, Quebec (1982).

41. Zucker, S. W., "Early Orientation Selection: Tangent Fields and the Dimensionality of Their Support," *Computer Vision, Graphics, and Image Processing* **32** pp. 74-103 (1985).