EXTRACTING AND LABELLING BOUNDARY SEGMENTS IN NATURAL SCENES (Revised and Updated)[1]

John M. Prager

COINS Technical Report 78-17

September 1978

Abstract

This paper describes a set of algorithms used to perform segmentation of natural scenes through boundary analysis. The techniques include preprocessing, differentiation using a very simple operator, relaxation using case analysis, and postprocessing. The system extracts line segments as connected sets of edges, labels them, and computes features for them such as length and confidence.

[1]This research was supported by the Office of Naval Research under Grant N00014-75-C-0459 and by the National Institute of Health under Grant R01 NS 09755-07 COM.

I. Introduction

We describe in this paper a set of programs that are used to perform a boundary analysis of images of outdoor scenes. Each process can operate in parallel across the whole image, since the value computed at a pixel is a function only of a small neighborhood around it. Due to the modular nature of these programs, they can easily be "unplugged" and replaced by more sophisticated versions, or left out altogether, if desired. The relaxation process in particular (described in Section III.4) was designed to be a minimal functional implementation of the ideas behind the algorithm. A more sophisticated elaboration of these ideas is described in [3]. The techniques described in this paper have been developed for use in the VISIONS scene analysis system [2].

I.1 Overview of the System

The goal of the system is to form a segmentation of an image, that is, to delineate the boundaries of the objects in the scene. This is done via differentiation using edge masks, a process which finds discontinuities in the intensity image. Prior to this stage, however, preprocessing is required to clean up the raw data somewhat. After differentiation, a relaxation process will consolidate the edges formed on the basis of local consistency requirements. Individual edges are then linked together during the binding stage to form extended line segments. Finally, properties of these line segments such as confidence are computed, allowing removal of low-confidence lines.

Conceptually there are four stages to our line-finding process. Each of these is implemented as one or more computational modules.

(1) Preprocessing: This stage cleans up the raw data.

(1a) UNMIX corrects for "mixed pixels" introduced in digitization.

(1b) CONDITIONAL AVERAGE smooths out random noise and fine microtexture.

(2) Generating the Edge Representation: This is the heart of the whole process.

(2a) DIFFERENTIATION finds the apparent edge-strength at each point in the image.

(2b) SUPPRESSION removes "multiple edges" formed by spatial differentiation of boundaries which are composed of a gradient across many pixels.

(2c) RELAXATION drives the probability of an edge at each point to 1 or 0 on the basis of local support or inhibition.

(3) Grouping: This stage joins edges into line segments and finds features of these lines.

(3a) BIND joins contiguous edges together to form line segments, and each line segment is given a unique label.

(3b) FEATURE EXTRACTION produces features such as length, contrast, location for each line segment.

(4) Postprocessing.

(4a) TRIM1 removes selected line segments (e.g., short, low contrast lines).

(4b) CONFIDENCE generates a confidence for each of the remaining segments that actually is a meaningful line segment.

(4c) TRIM2 removes low confidence lines by thresholding on their confidence level.

II. Stage 1 -- Preprocessing

We start with the image digitized into three arrays containing the red, green, and blue components of the scene. These are now averaged to form the black and white intensity image. Prior to differentiation,

this image undergoes two preprocessing stages which provide the black and white image upon which the line-finding process works.

(Step 1a) UNMIX: The first process is designed to eliminate what is known as the "mixed-pixel" problem. This problem occurs whenever images are digitized, and is due to the fact that boundaries in the image will not in general fall in register with the digitization grid. Thus, the intensity recorded at a pixel might overlap two regions, and therefore represent a weighted average of them (see Figure 1).

The procedure must test to see if a two-step intensity gradient occurs at the same place in all of the three colored images. If it does, then a mixed pixel is assumed to have been formed (of course this assumption might not be correct and then errors would be introduced). It is consequently "unmixed" by assigning to it the values of its nearest neighbor along the direction of the gradient. This has the effect of shifting the boundary by a fraction of a pixel (see Figure 1c).

(Step 1b) CONDITIONAL AVERAGE: The second process is an adaptation of a smoothing process due to Rosenfeld [7] which helps eliminate noise in the image. In this process, the intensity value at each point is replaced by the average of itself and its neighbors, except that if the difference between the value of the point and a neighbor is greater than a certain value T, then that neighbor is not included in the average.

For the neighborhood $\{N_i\}$ of the point N_0 in Figure 2, its updated value is given by:

$$N_0' = \frac{1}{n} \sum_{N_i \epsilon S} N_i$$

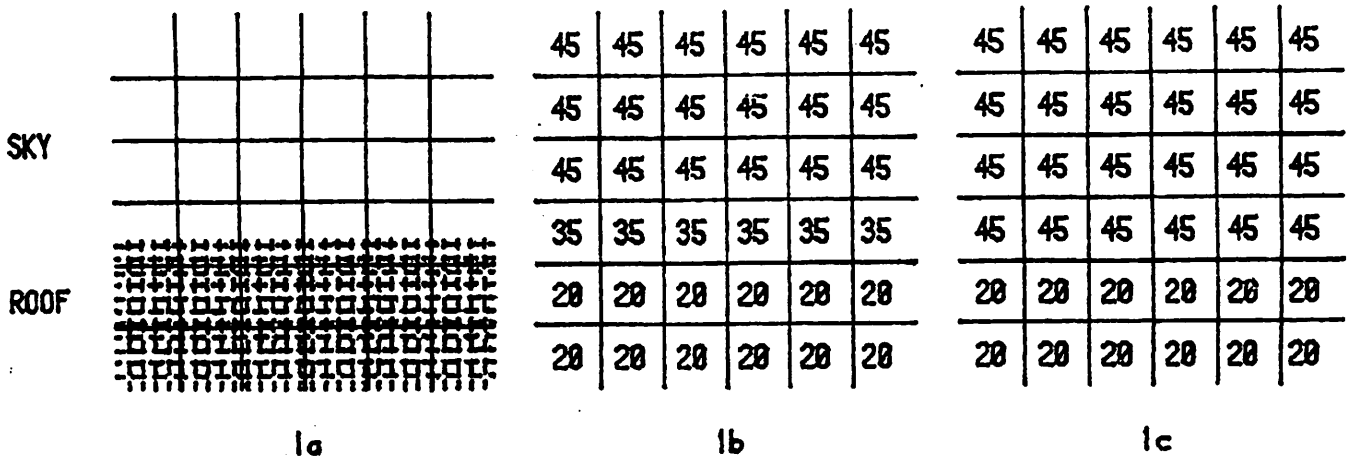


Figure 1. The 'mixed-pixel' problem. 1a: Digitization grid superimposed upon a portion of an image. 1b: Intensity values recorded in this grid. 1c: "UNMIX" correction applied to 1b.

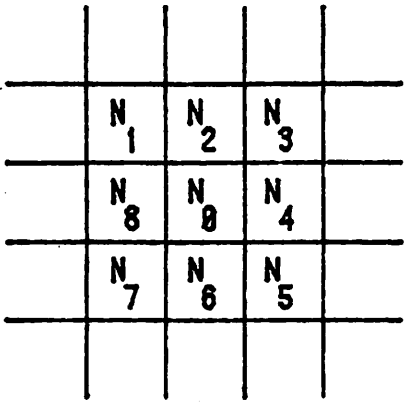


Figure 2. Neighborhood considered in 'CONDITIONAL AVERAGE'..

where $S = \{N_i : |N_i - N_0| < T\}$ and n is the cardinality of S. Note that S always contains N_0. This procedure has the following effects:

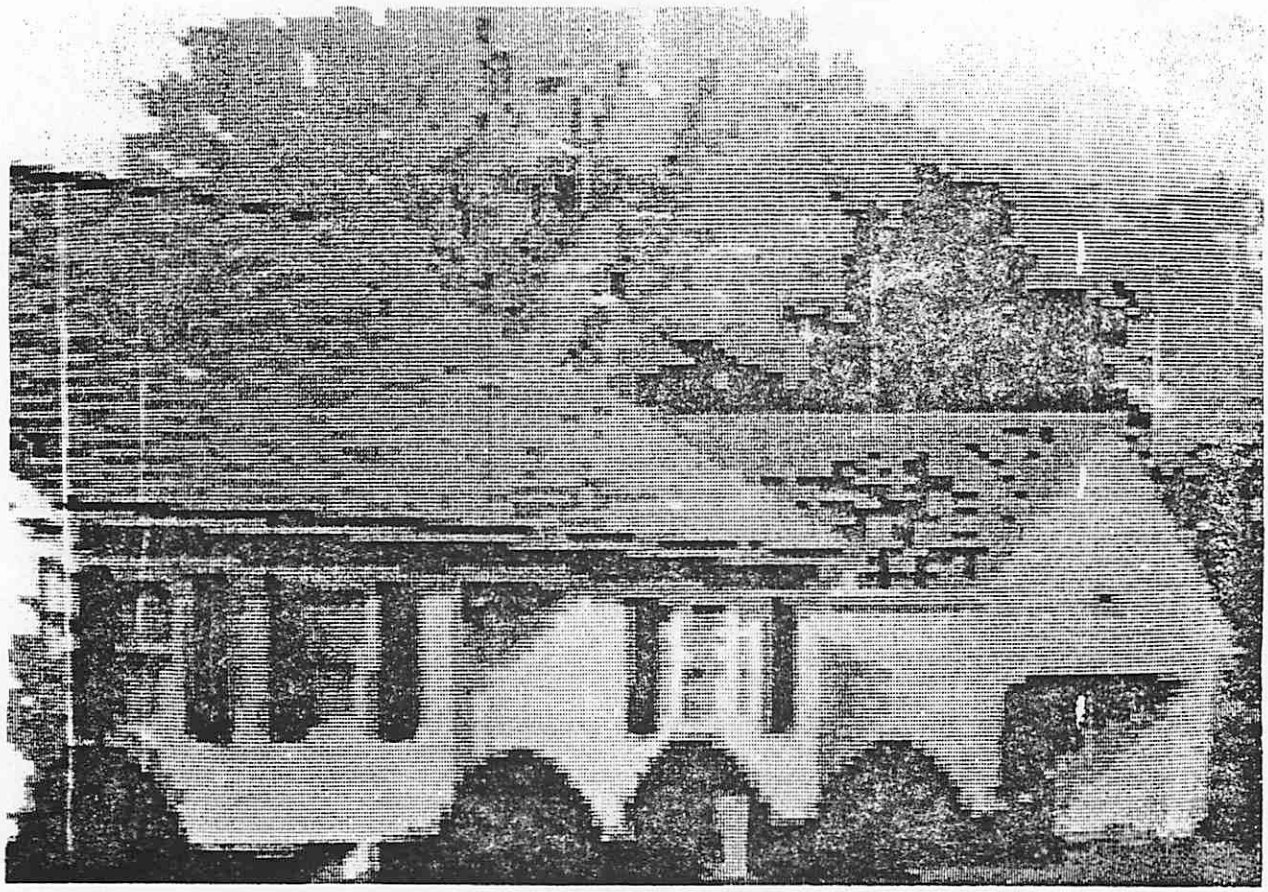
(1) Within a homogeneous region, it smooths out small amounts of noise (small relative to T).

(2) Near a region boundary whose contrast is greater than T it includes no points across the boundary in the average. This allows a smoothing of the points on either side of the boundary without blurring the boundary as a nondiscriminatory averaging process would do.

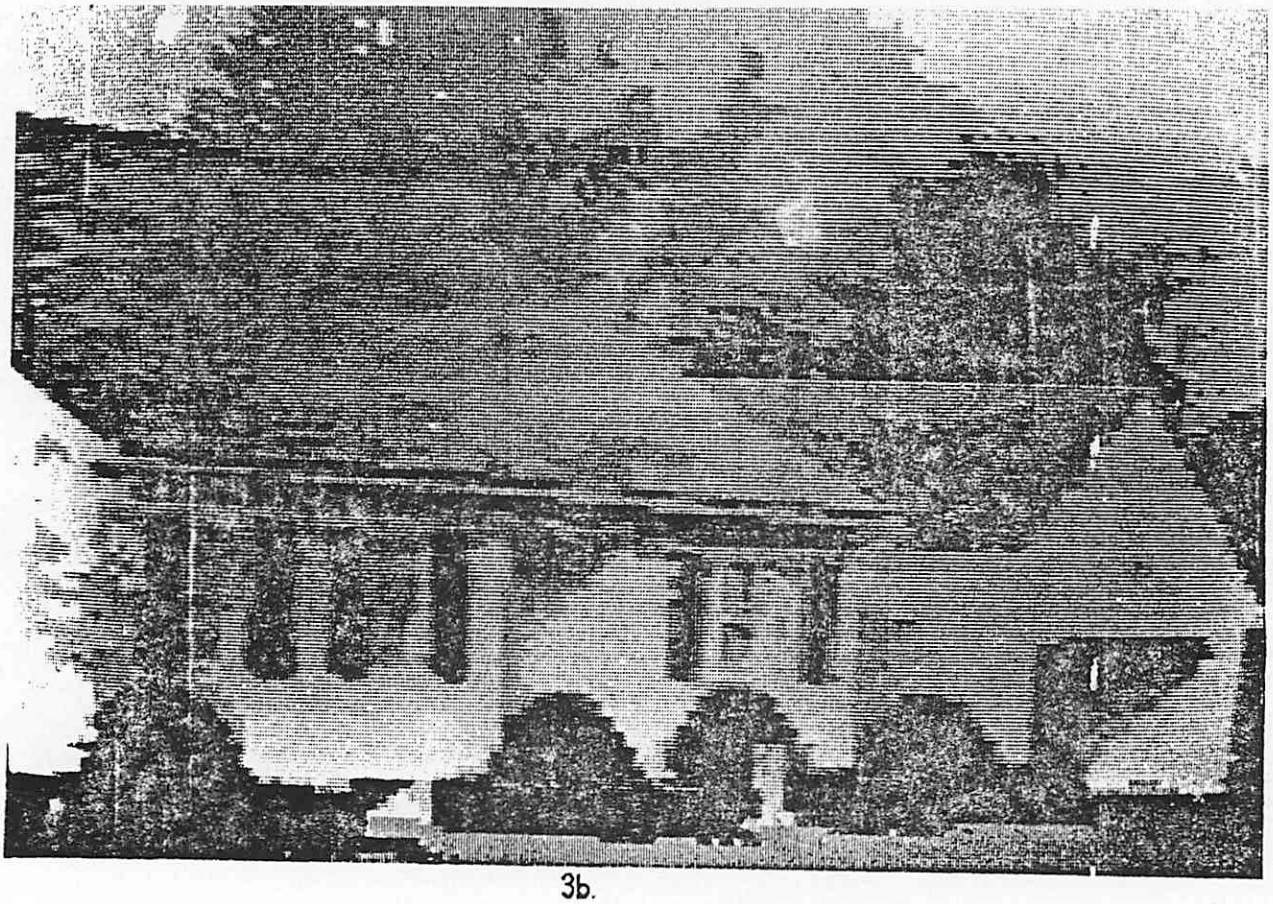
(3) Within an intensity gradient, the process averages a point with roughly as many other points that have smaller intensity as greater. This will smooth noise within the gradient, but it will not destroy the gradient.

(4) In a textured region, if the texture elements differ little in intensity (relative to T), they will be smoothed into a homogeneous region. If the texture elements differ by more than T, then no averaging will occur, except perhaps within the texture elements themselves.

After experimental testing of the differential operator (to be described later) on images that have selectively undergone the UNMIX and/or CONDITIONAL AVERAGE passes, it was subjectively concluded that an application of both processing techniques gives the cleanest results (i.e., no loss of any important lines, or addition of extraneous ones). Examples of the combined use of these preprocessing stages on intensity images are shown in Figure 3.

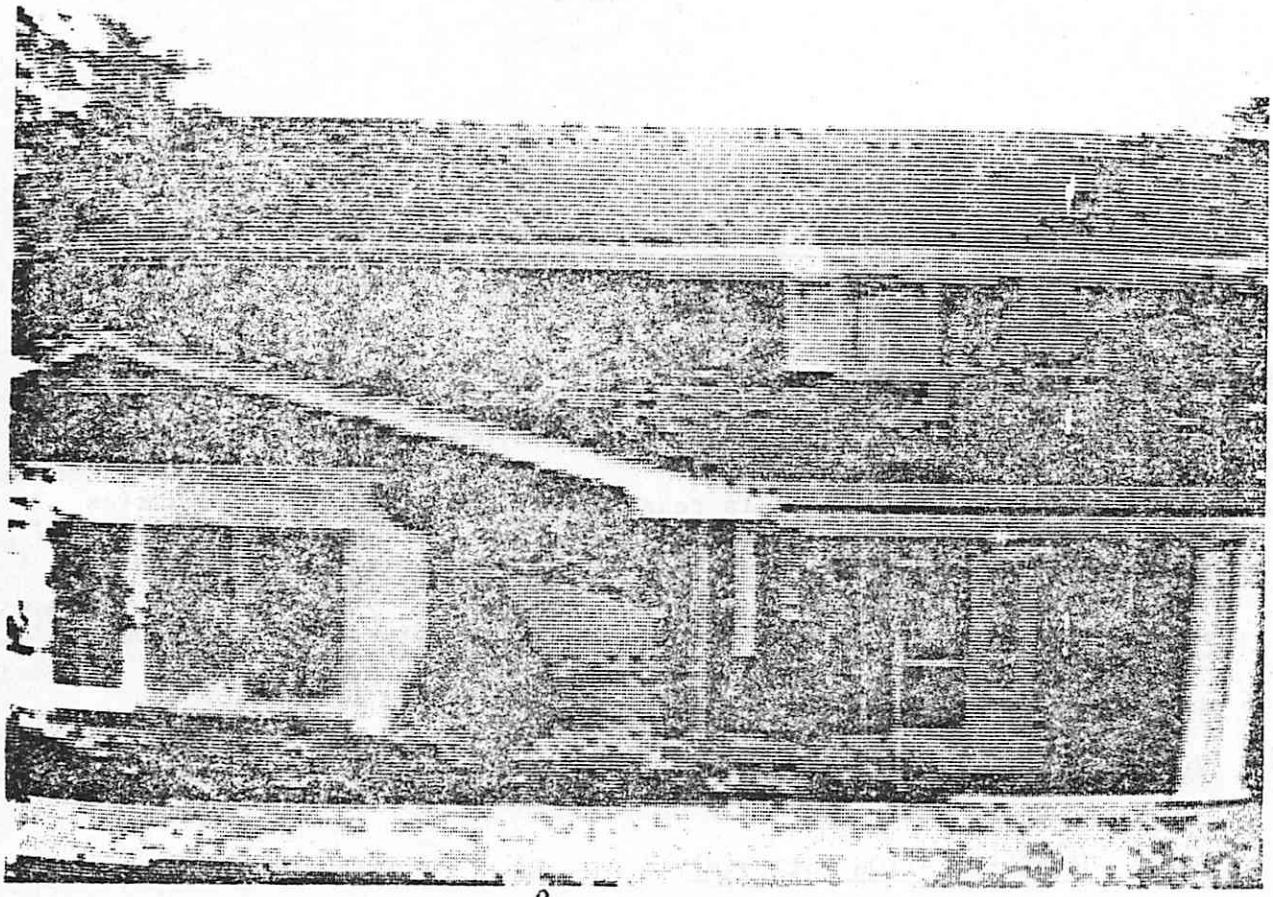


3a

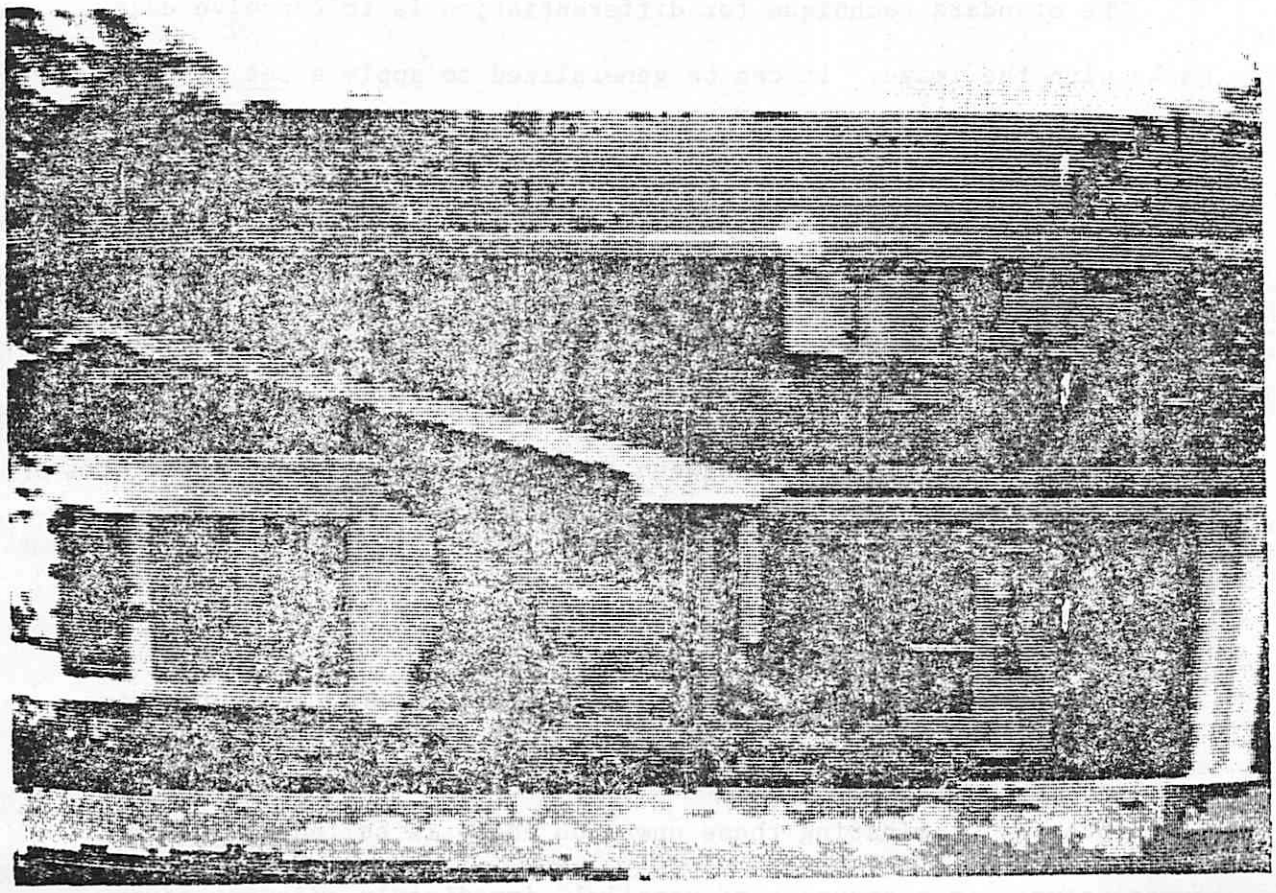


3b.

Figure 3. Preprocessing. Figure 3a shows the original intensity data. Figure 3b shows the same data after the "UNMIX" and "CONDITIONAL AVERAGE" steps.



3c



3d

Figure 3d shows the intensity data of Figure 3c after preprocessing.

III. Stage 2 -- Generating the Edge Representation

III.1 Representation of the Edge Image

The input consists of an array of numbers representing the light intensity of each position in the image. Since each of these pixels is to be in some region, it is reasonable to constrain the boundaries of regions to fall only between pixels. Representing the image on a rectangular grid and constraining edges to lie between pixels imposes a boundary that consists entirely of horizontal and vertical edges (as in [1] and [10]). This greatly facilitates further processing.

III.2 DIFFERENTIATION (Step 2a)

The standard technique for differentiation is to convolve edge masks with the image. It can be generalized to apply a set of masks, and to compute the output as some function of the responses of these masks, often the maximum response.

In [5], experiments were performed with several combinations of different masks, and it was found that on the preprocessed data, the simplest possible mask gave the best overall results. This mask computes the difference in intensity between two adjacent pixels and associates the result with their common boundary.

III.3 SUPPRESSION (Step 2b)

One of the weaknesses of using masks is that wide gradients give rise to multiple parallel indications of the same edge. A simple technique for eliminating these unwanted edges is called non-maxima suppression: in a sequence of parallel, immediately adjacent edges, all but the strongest are eliminated. If, further, the total contrast

of all the edges across the gradient is collected into the remaining edge, then the edge-picture resulting is much more representative of the strengths of the gradients; this is the algorithm used in the system described in this paper. For further discussion of these techniques, see [3,5,6].

III.4 RELAXATION (Step 2c)

III.4.1 Background

The edge strengths produced by the differentiation process depend upon the local contrast in the image. Weak edges may arise from low-contrast boundaries, gradients extending over many pixels, or from texture internal to a region, or indeed from noise. The output of the differentiation process is thus usually far from being clean. If the strengths of edges are viewed as probabilities, or confidences, of the existence of edges, usually few of them would be considered to have probabilities of 0 or 1. An edge probability that is neither 0 nor 1 effectively is an ambiguous interpretation of the entity concerned. However, the local context around each edge contains information for updating the probability so that ultimately the ambiguity will be reduced and interpretations will be locally consistent. The process of updating these confidences in parallel will be called a relaxation process.

In this scheme, a label λ is assigned to each position with an initial probability $P(\lambda)$. The set of labels Λ can be a set of edge-descriptors, such as "horizontal edge", "vertical edge", etc., and would typically include a special label, the "null edge" label, which is an assertion that there is no edge at that point. Λ is chosen so that the labels are mutually exclusive, since it is desired that ultimately one label

will be present with probability one, the others with probability zero at each point. The labels may be regarded as competing at each position in the image during relaxation. In this process, each probability for every label at every position is then updated in parallel according to its compatibility with the labels at neighboring positions (in some predefined neighborhood). Under quite restricted conditions convergence can be guaranteed [11], although not necessarily to any meaningful global interpretation.

A consideration of the relaxation model of Rosenfeld et al. described in detail in [8] (and summarized above) leads one to conclude that the heart of the scheme is in the setting of the (many) compatibility coefficients. Not only are there many of these which need to be set, but due to heavy interdependence of effects there is no direct correlation between the setting of an individual weight and the performance of the system. Thus, tuning can be very difficult, since it requires optimization of many variables simultaneously. Furthermore, there is no guarantee that it is possible to set weights such that all the desired effects can be achieved simultaneously. While it is fairly straightforward to set the weights so that some of the more obvious cases are taken care of, there is rarely enough leeway to adjust them so that the more "awkward" cases, such as when part of an edge's neighborhood supports it and the other part inhibits it (case 0-1 described in III.4.5 below), are managed correctly. Indeed, it is difficult to determine where the system is failing, or how it is achieving its results.

It appears that one source of these difficulties arises when the updating process employs a single formula that is used to take care of

the various very different cases that arise. In the next section, an alternative scheme is proposed which will deal with each of the aforementioned problems separately, in a clearly structured manner.

III.4.2 A Different Representation for Relaxation

In the scheme just described, a set of labels are competing for each position in the image. Thus for a point on a diagonal boundary, both horizontal and vertical labels will be competing. In our representation, we can allow both labels to coexist at a pixel since we are placing edges at interpixel boundaries, not on top of the pixel. Therefore, at each vertical pixel boundary the only labels we need to consider are "vertical edge" and "no edge", and similarly for horizontal edges. In this way, the set of two probabilities at each edge location $\{P_i(\lambda) | \lambda \in \Lambda\}$ can reduce to a single parameter P_i. The probability of an edge at position i is P_i, while the probability of the null label "no edge" at position i is $1 - P_i$. Relaxation is very much simplified as a result of this representation.

We will use the notation of Figure 4 to describe the edge-configurations under consideration.

An open rectangle will represent the edge to be updated.
A dotted line will represent an edge position with no edge present,
a thick solid line the presence of an actual edge,
a thin solid line an edge of undetermined strength.

Now let us describe the algorithm. Associated with each edge-position will be a value indicating the probability or confidence that an edge exists at that position. Every edge-position has two end-points at which that edge could continue, and every end-point has three other edge-positions incident upon it. Each edge end-point will be classified as one of four "vertex-classes" according to the strengths of the incident

edges. The vertex classes of the end-points of the edge-position under examination will then determine how the edge-strength is to be updated.

III.4.3 Cases for Updating Edges

An iterative procedure for updating the probabilities is described below. We will denote the configuration of continuing edges at an end-point by an integer n in the range 0 to 3, representing the number of such edges in the pattern. A configuration of n edges at a vertex of edge e will be considered equivalent no matter which of the three possible edge positions to that side of e that they take. These four equivalence classes of continuing edge patterns are depicted in Figure 5.

Now it will be remembered that few edges are present or absent with any certainty. Therefore the usual case is that each equivalence class has a probability of being true which is a function of the probabilities of the individual edges. The determination of which classes, or vertex types are present, computed as a function of the probabilities of the three edges to either side is now discussed.

III.4.4 Computation of Neighborhood Patterns

We would like to classify the configuration of edges to each side of e as one of the four vertex types of Figure 4a-d. Consider one end-point, say the left one, in Figure 4e. We will assume that the numerical values associated with edges are in the range 0-1, representing probabilities of the presence of an edge.

Since we are treating perpendicular continuation as equivalent to straight-line continuation (i.e., a and c have exactly the same effect on e as does b), we can assume without loss of generality that

$$a \geq b \geq c.$$

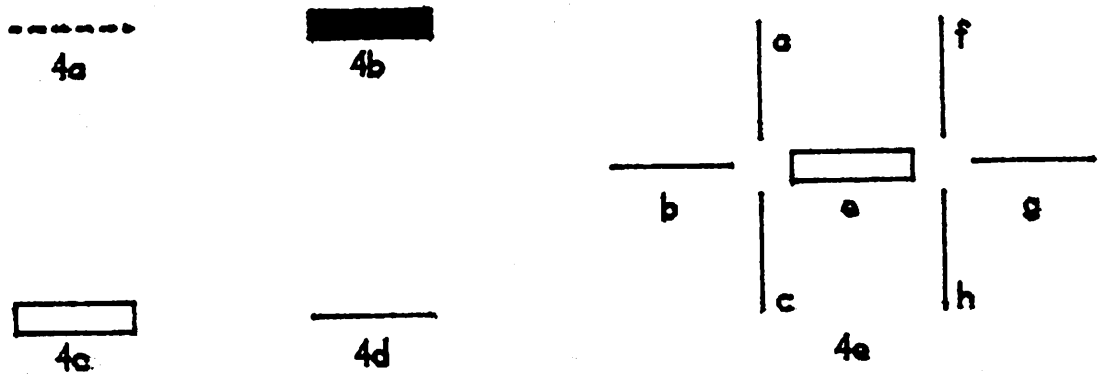


Figure 4. Notation. 4a. Edge position with no edge. 4b. Edge position with edge. 4c. Edge to be updated. 4d. Edge of unknown strength. 4e. Configuration of edges around central edge e.

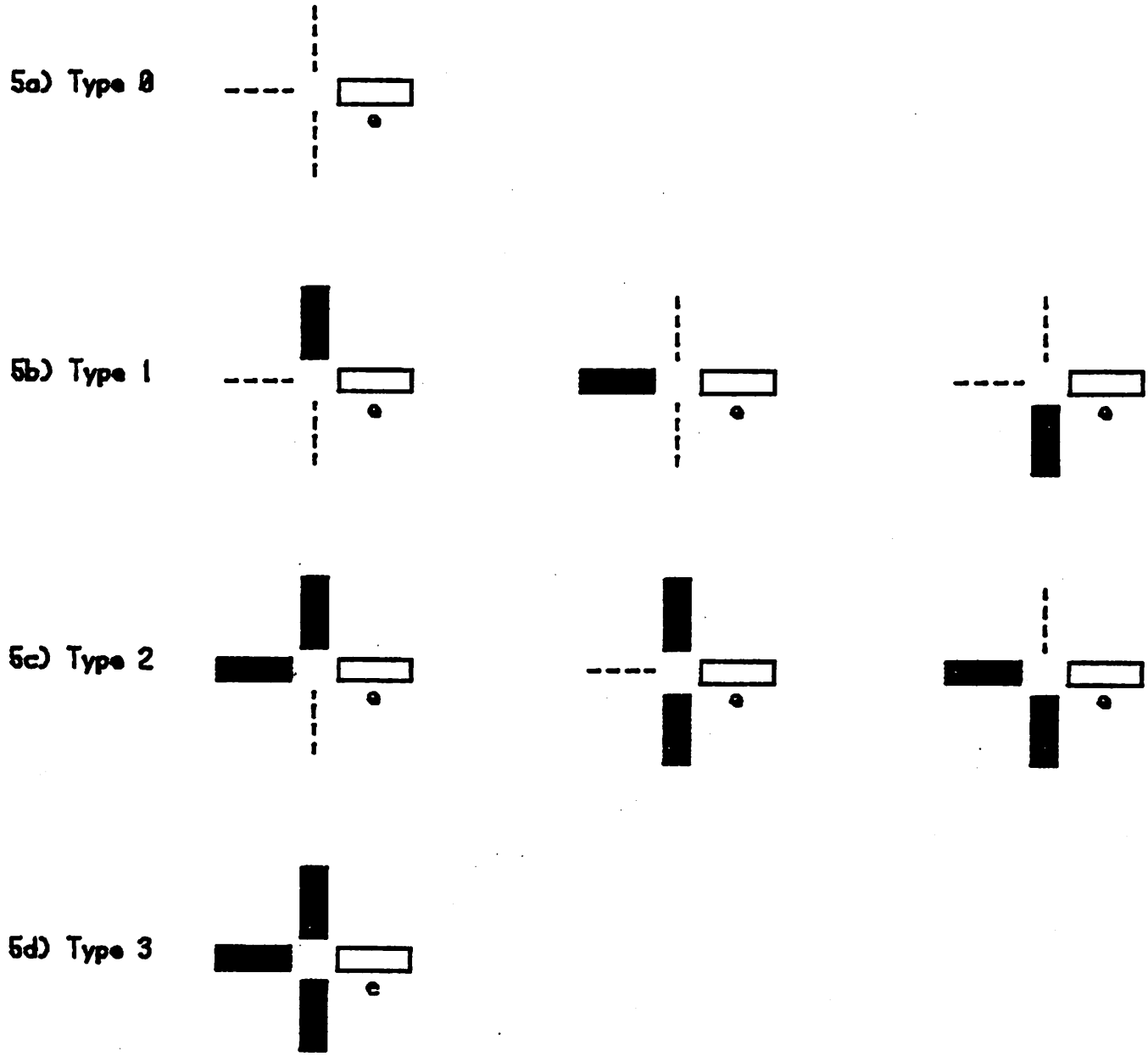


Figure 5. Classification of vertex-type of left-hand endpoint of edge e.

Assuming independence of the edges (unfortunately, often a bad assumption), a simple calculation would give for vertex types 0-3:

$$\Pr(\text{type } 0) = (1-a)(1-b)(1-c)$$
$$\Pr(\text{type } 1) = a(1-b)(1-c)$$
$$\Pr(\text{type } 2) = ab(1-c)$$
$$\Pr(\text{type } 3) = abc.$$

The case with the highest probability is then chosen as being the "state" of the left side of edge e.

However, there are cases where, for example, b and c are very low and a is considerably larger than them, but perhaps not close to 1. In such situations we would like a strong indication of a type 1 vertex (see Figure 6a). The remedy would be as follows: instead of subtracting a, b, and c from 1 to form the no-edge probabilities, we can subtract from m, where $m = \max(a,b,c) = a$ in this case. m thus represents at a very local level the probability of a high-confidence edge. Thus we have

$$\Pr(0) = (m-a)(m-b)(m-c)$$
$$\Pr(1) = a(m-b)(m-c)$$
$$\Pr(2) = ab(m-c)$$
$$\Pr(3) = abc.$$

There is one difficulty with this formulation. It could occur that a is much larger than b or c, but also be very close to zero (see Figure 6b); our formula would calculate a larger value for Pr(1) than Pr(0) when type 0 should actually be selected. This can be easily fixed by anchoring m to some minimum value q. We need a lower bound for m because there is always a chance that a stronger edge could be present. This will guarantee type 0 to be the most probable vertex type when

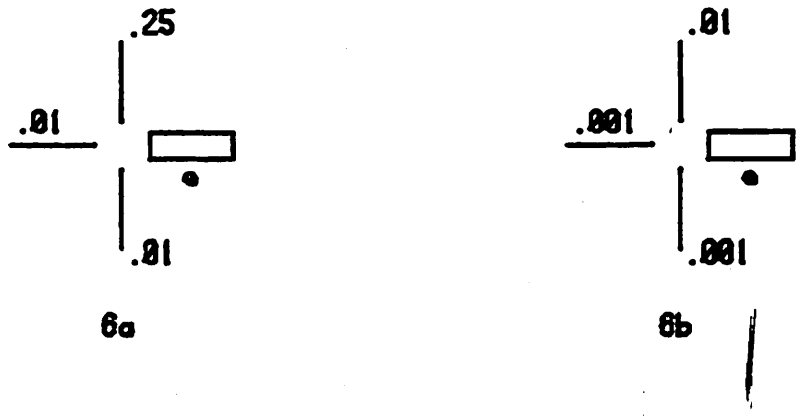


Figure 6. Two configurations depicting low-probability neighbors of e. Vertex-type 1 is indicated in 6a, and vertex-type 2 in 6b.

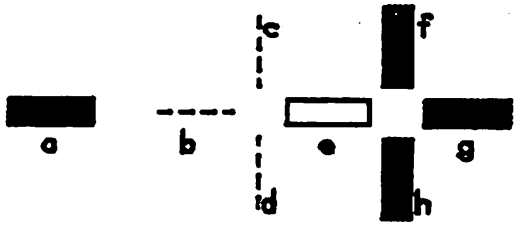


Figure 8. Edge e is classified as being type 2-3. If its strength is high, it is likely that edge a will join up with it. The desirability of this effect is not so clear if e is weak.

all incident edges have very low strengths. Thus the final definition of m is

$$m = \max(a,b,c,q) \text{ and}$$
$$Pr(0) = (m-a)(m-b)(m-c)$$
$$Pr(1) = a(m-b)(m-c)$$
$$Pr(2) = ab(m-c)$$
$$Pr(3) = abc.$$

Since we will select vertex type i, where $Pr(i) = \max_j[Pr(j)]$, we only need to know the relative sizes of the Pr(i), so we do not need to normalize these probabilities. q can be calculated as a function of edges in some neighborhood of e in the image. A suitable such function might be $\mu-\sigma$, where μ is the average edge strength and σ its standard deviation. We found that a constant value for q of about .1 performed very well over several images.

III.4.5 Calculation of Direction of Update

The following notation is used to depict the neighborhood characteristics (or state) of an edge: the symbols i-j denote that configuration i is at one vertex of central edge e, and j is at the other. Obviously, $i-j \equiv j-i$, so we need only consider the ten cases of i-j where $i \leq j$, shown in Figure 7.

Now in states 0-0, 0-2, and 0-3 one can quite confidently say that there is no good support for e, and in 1-1, 1-2, and 1-3 one can quite confidently say that there is. However, if e is in state 0-2, for example, it is conceivable that the situation is really as in Figure 8. In such a case, the current strength of e itself may be a determining factor for the case that b and e should be grown in to complete the line.

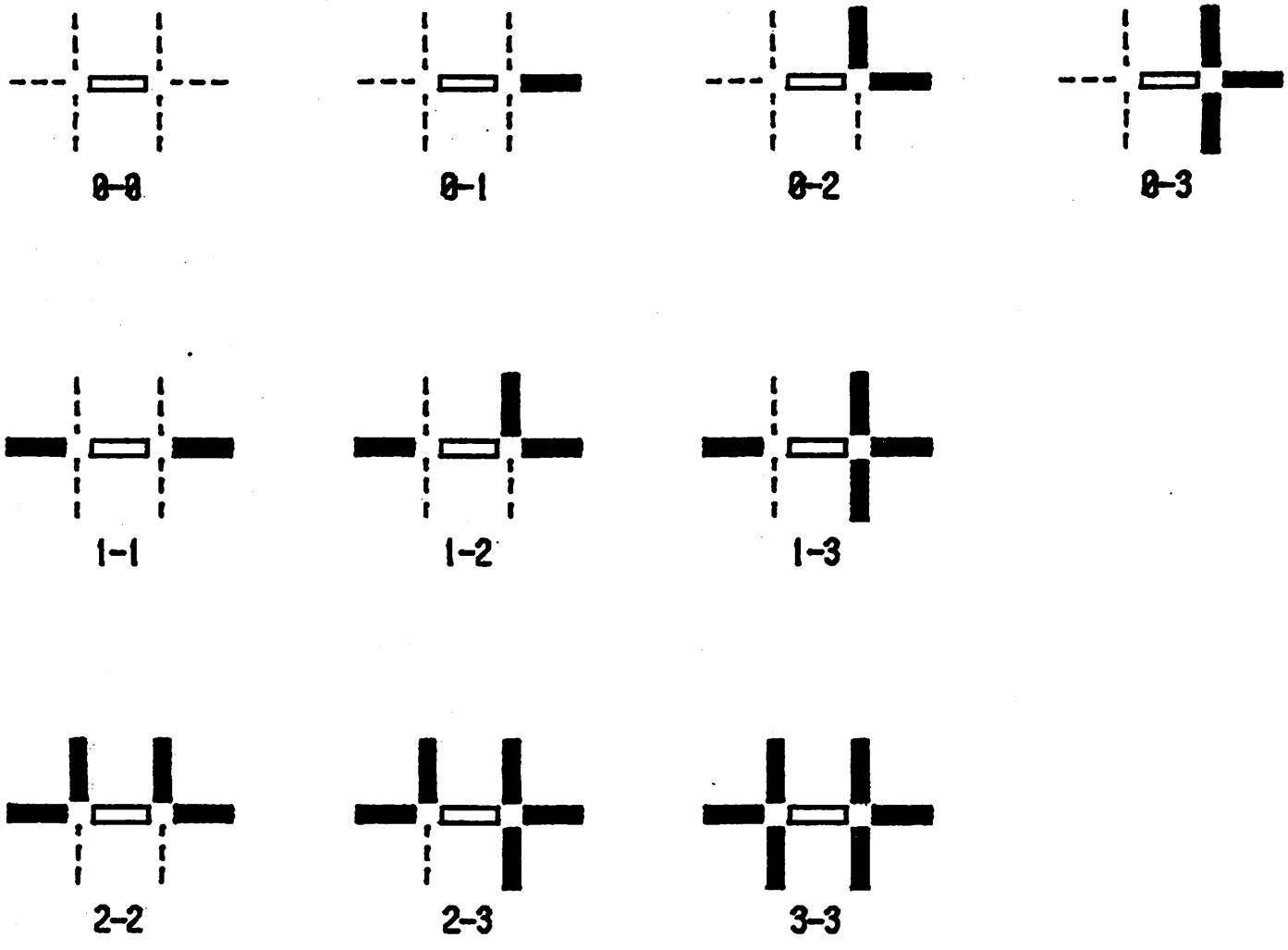


Figure 7. Representative combinations of vertex-types. This figure depicts all possible cases, subject to symmetry and the equivalences noted in the text.

Two points may now be made. First, in some of the above cases it is clear how edge e should be updated. Therefore, the updating process should explicitly increment or decrement the edge. Secondly, as information may need to organize and propagate for some period of time over some distance in the image, updating increments (decrements) should not drive the probabilities to one (zero) too quickly. Rather, the increase (decrease) should be some small amount on each iteration. In this manner the influence of regions which are initially locally consistent will spread into "less confident" regions, much as "islands of reliability" played a part in the Hearsay speech analysis system [4].

So in cases 1-1, 1-2, and 1-3 we will let e increase (see Figure 9a); and in cases 0-0, 0-2, and 0-3 we will let e decrease (see Figure 9b). In all other cases the context is not very clear. Leaving aside case 0-1 for the moment, we see that in none of the cases 2-2, 2-3, 3-3 (see Figure 9c) is the presence or absence of e critical for the continuation of a neighboring edge since they have alternative directions for continuation. It will not introduce or eliminate "cracks" -- edges terminating at an indeterminate point. Whether e should exist or not depends largely upon its contrast strength, as well as continuity properties on either border, and little else, at least until more global views and higher level knowledge is available.

Case 0-1 is really the only problem. The neighborhood on one side strongly supports e, yet the other suggests that e should be absent. As no sensible decision can be made, no action is taken here (or in cases 2-2, 2-3, and 3-3). This is a very important decision: it implies that in the updating process, the 0-1 case remaining constant will prevent a

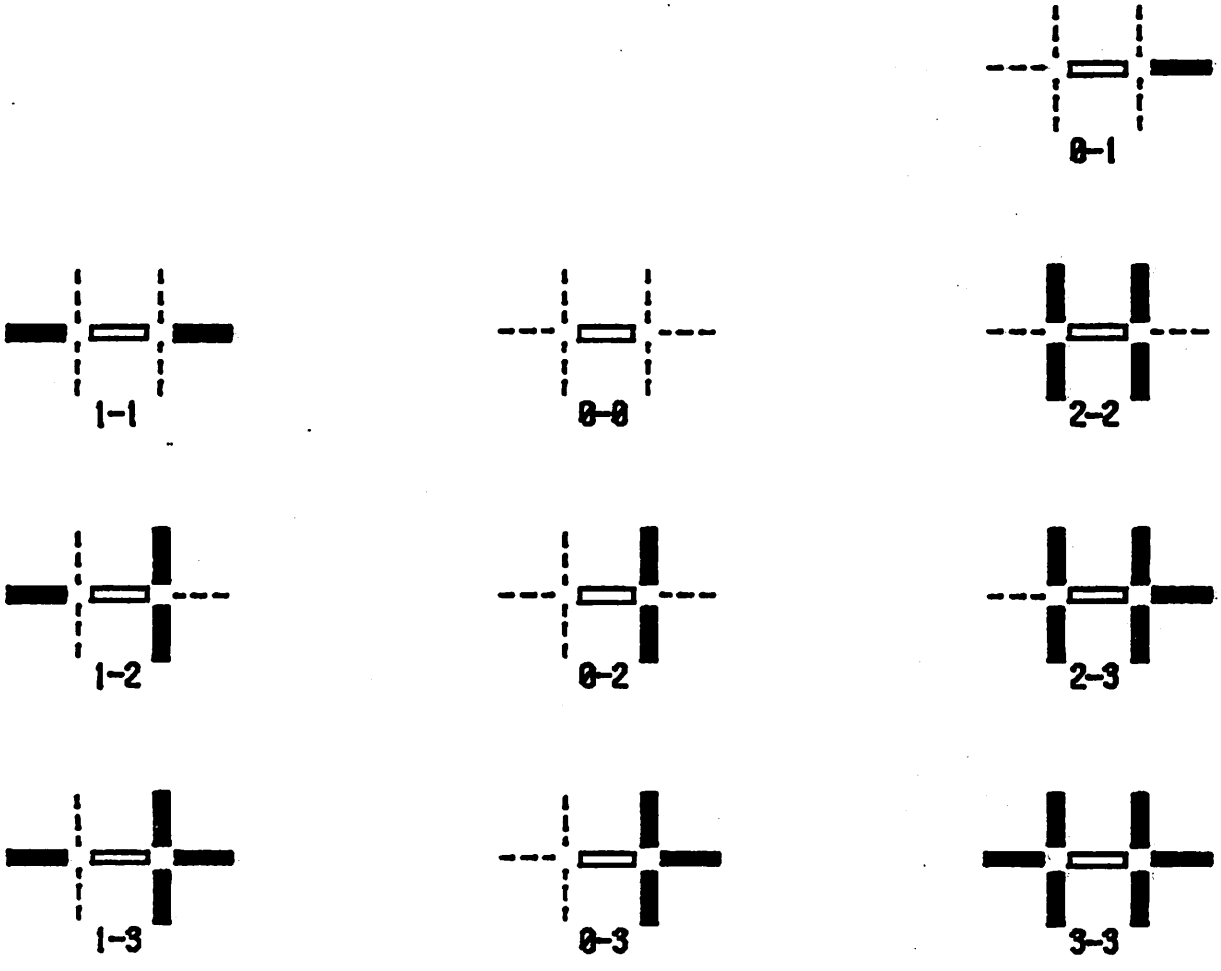


Figure 9. Cases for updating edge. In figure 9a are depicted all those cases where the central edge should be incremented, and in 9b those where the edge should be decremented. Figure 9c shows the uncertain cases.

line from growing into noise or from being eaten away at its terminating point.

III.4.6 Performing the Update

The operation of the system in updating an edge may now be described. Let P_e^t be the probability or confidence (in the range 0-1) of edge e at time t. Then the upating formula depends upon the situation as follows.

Increment case: $P_e^{t+1} = \text{Min}(1, P_e^t + k)$

Decrement case: $P_e^{t+1} = \text{Max}(0, P_e^t - k)$

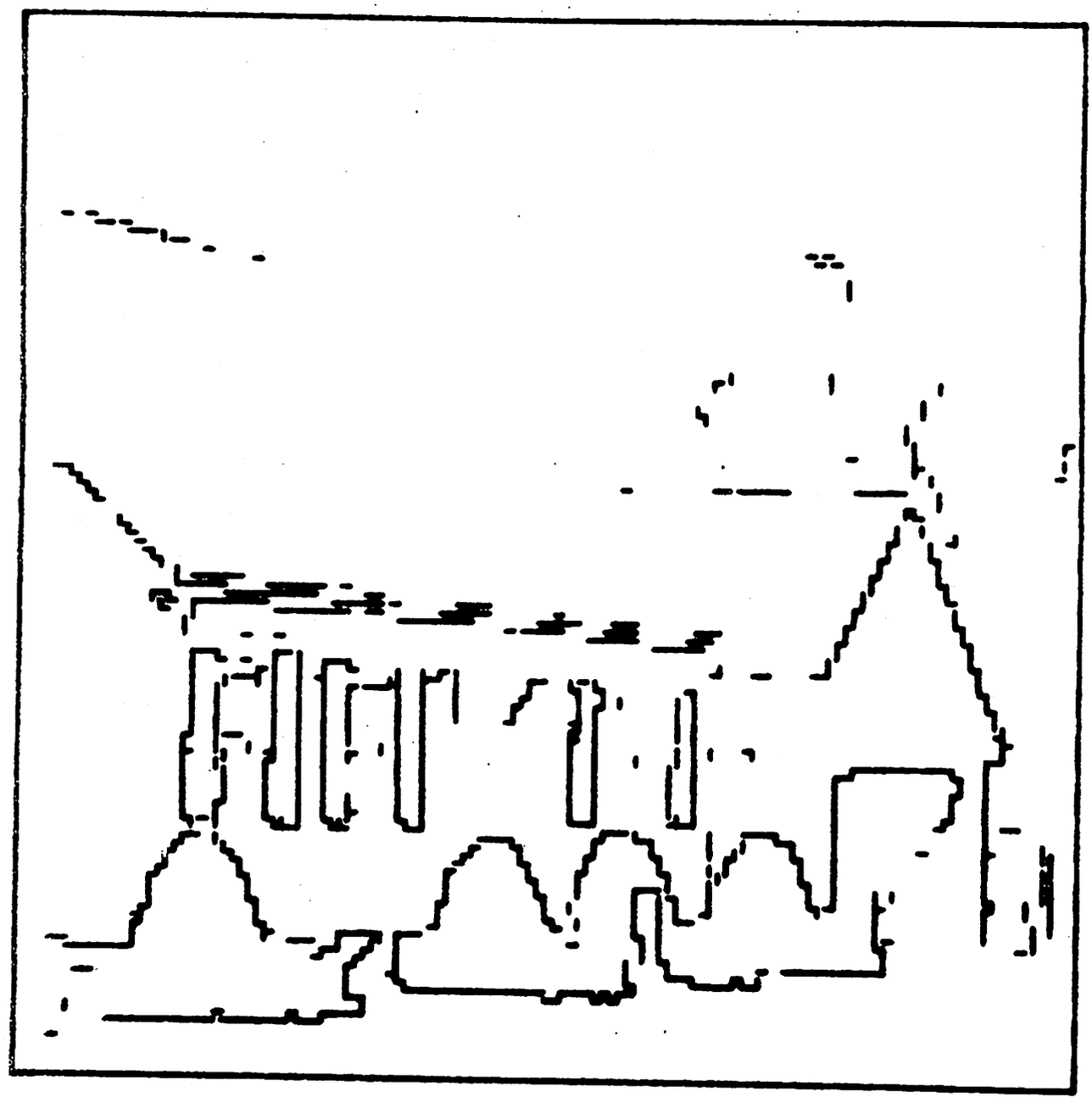
Uncertain case: $P_e^{t+1} = P_e^t$

where k is a constant. A large k gives fast convergence, but does not permit information to propagate very far before the probabilities of edges converge to 0 or 1. For small k the opposite is true. By experimentation on several images, values in the range .15 to .20 were found to be most suitable. Typical results of using this relaxation process are given in Figure 10.

IV. Grouping

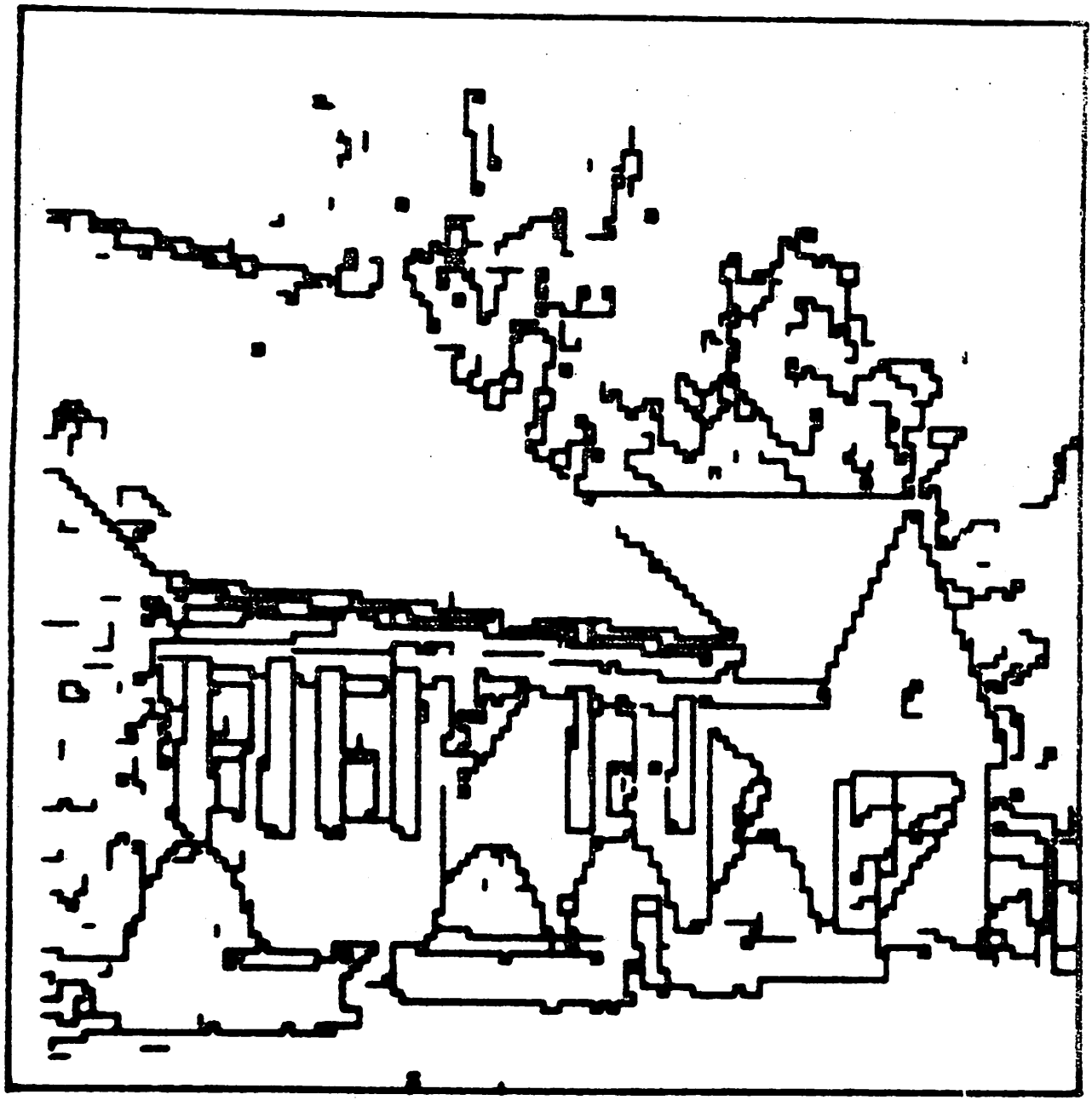
IV.1 BIND (Step 3a)

The next stage is to decide which neighboring edges link up to form extended line segments. It is clear that those points in the current representation which have 1, 3, or 4 edges entering them, i.e., vertices of degree 1, 3, or 4, are natural termination points for these line segments (see Figure 11). Breaking boundaries in these places will tend to form segments which lie between only two regions. This is a highly desirable effect, since there will then be less variation of



10a

Figure 10. Differentiation and relaxation. Figure 10a shows the data in Figure 3a differentiated. Edge strengths have been thresholded at .25 for display purposes only.

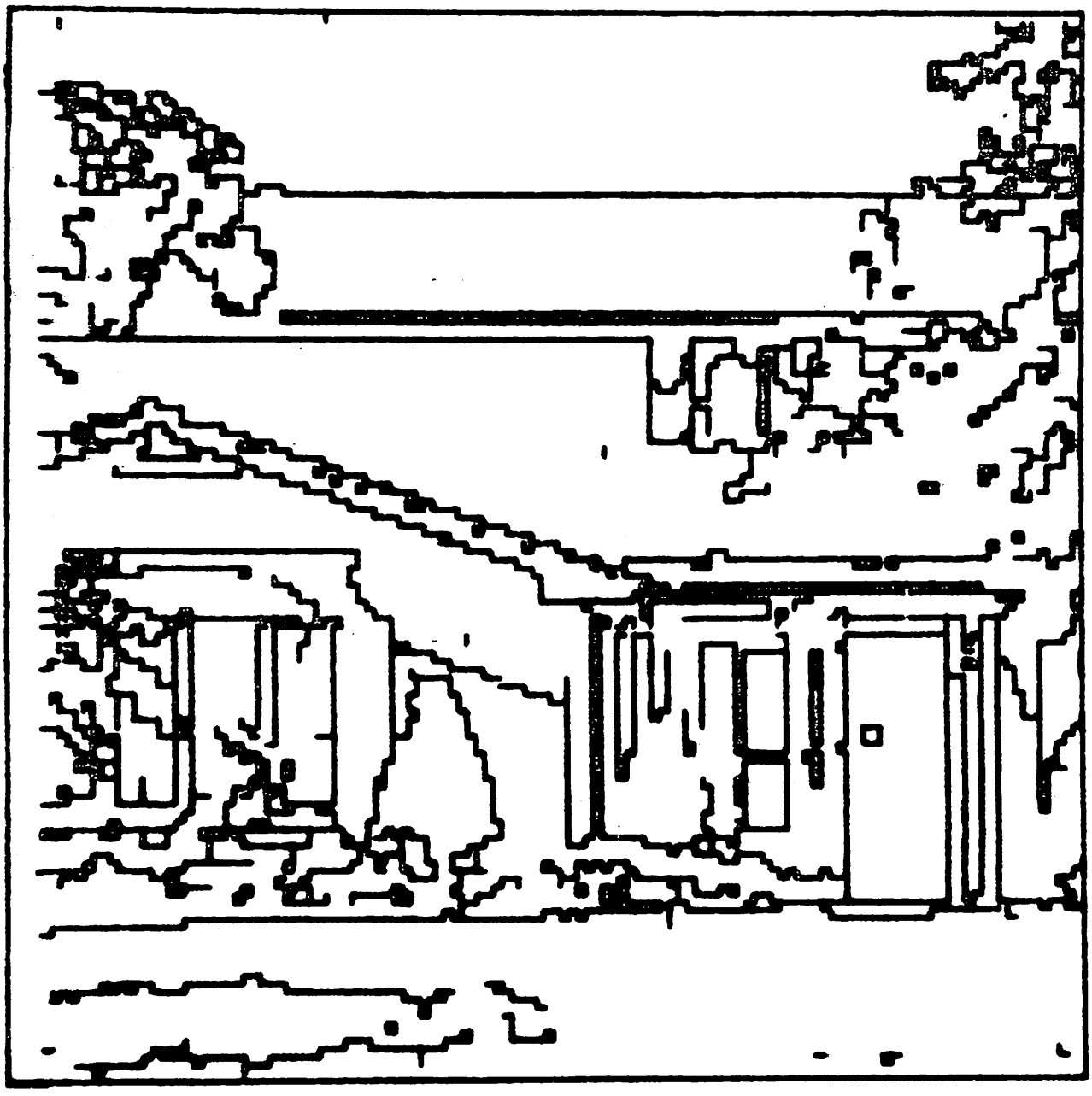


10b

Figure 10b. Results after 5 iterations of relaxation applied to Figure 10a.

10c

Figure 10c. Differentiated version of Figure 3b. Edge strengths have been thresholded at .25 for display purposes only.



10d

Figure 10d. Results after 5 iterations of relaxation applied to Figure 10c.

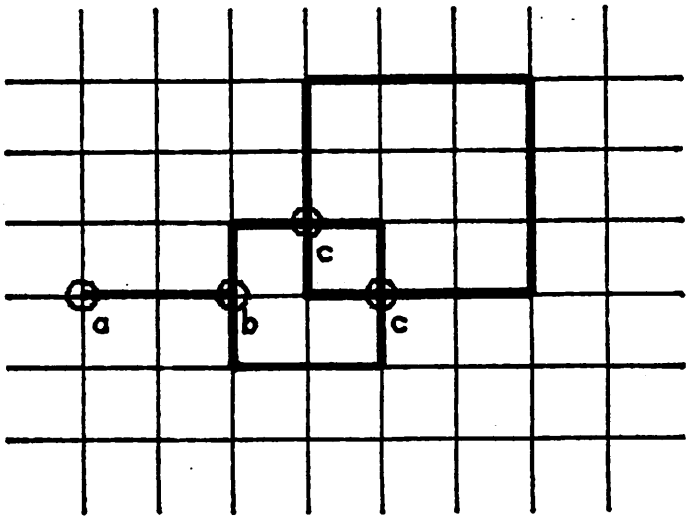


Figure 11. Three kinds of vertices. (a) Order-1. (b) Order-3. (c) Order-4.

properties (such as intensity) on either side of the segments. This was a major design consideration in the RSE representation of low-level output in the VISIONS system [2].

The first stage of the binding process, then, is to mark all vertices which terminate segments as in the configurations in Figure 11. Following this computation, it is straightforward to track all segments between vertices and assign a unique label (line-number) to each boundary segment.

IV.2 FEATURE EXTRACTION (Step 3b)

For each unique line-segment a set of properties can be established, some requiring recourse to the original intensity image, or at least, the intensity image that was differentiated. Typical properties to be associated with the segment label are:

(1) coordinates of end-points;
(2) N-length (defined as the number of edges that comprise the line);
(3) E-length (defined as the Euclidean distance between the end-points);
(4) frequency with which the edges that comprise the line change direction;
(5) mean and variance of contrast across the line, computed along its length;
(6) mean and variance of difference between neighboring points on either side of the boundary computed along its length.

Properties 2 and 5 can be used to give a measure of confidence that the extracted segment represents a meaningful unit of a visible boundary. Property 6 gives an indication of the homogeneity of a thin peripheral strip of the regions that the line bounds. Properties 1, 2, 3, and 4 can be used to compute a measure of the straightness of the line. These properties are important for later use in the interpretation

phases of processing.

V. Postprocessing

V.1 TRIM1

Most unwanted line segments can be eliminated on the basis of low confidence (see V.2), but it turns out that certain kinds of low-confidence edges result as a consequence of the idiosyncracies of the particular relaxation procedure employed. In particular, spurious edges are sometimes formed because multiple edges on a gradient are mistakenly believed to be distinct boundaries, and any noise points remaining despite the earlier preprocessing stages get bounded by "bubbles" (see below); these unwanted edges are best removed by a distinct process. Since they can be detected by their "topological" nature, they can be removed before the confidence generation process in V.2. This improves the latter procedure, as is explained in that section.

TRIM1 detects two kinds of unwanted edges: short edge-segments with at least one order-one vertex -- called "spurs", and some or all of the edges surrounding one-pixel regions -- called "bubbles". For example, all edges marked with a cross in Figure 12 will be removed. Results of applying this process are shown in Figure 13.

V.2 CONFIDENCE GENERATION

The confidence associated with a line segment will be based upon a measure of how dissimilar the points in the regions on either side of the line are to each other. While each line segment has been generated from edges which in turn were formed on the basis of local discontinuities,

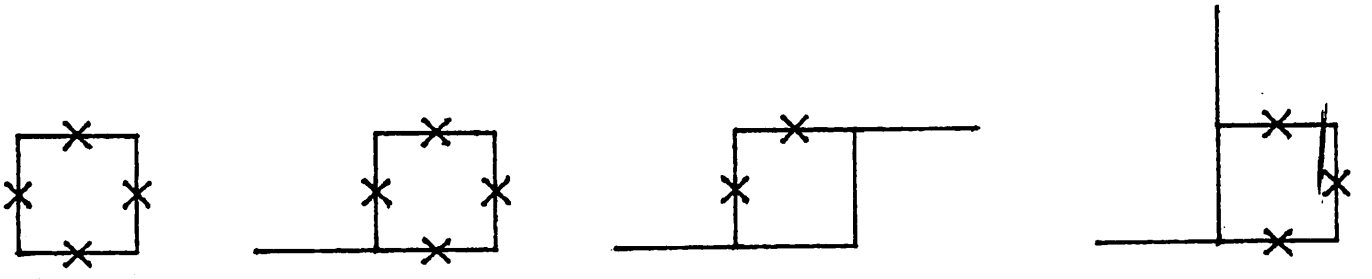


Figure 12. All edges marked with a cross will be removed to eliminate 'bubbles' from the image.

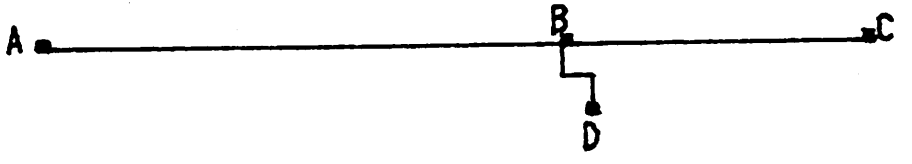
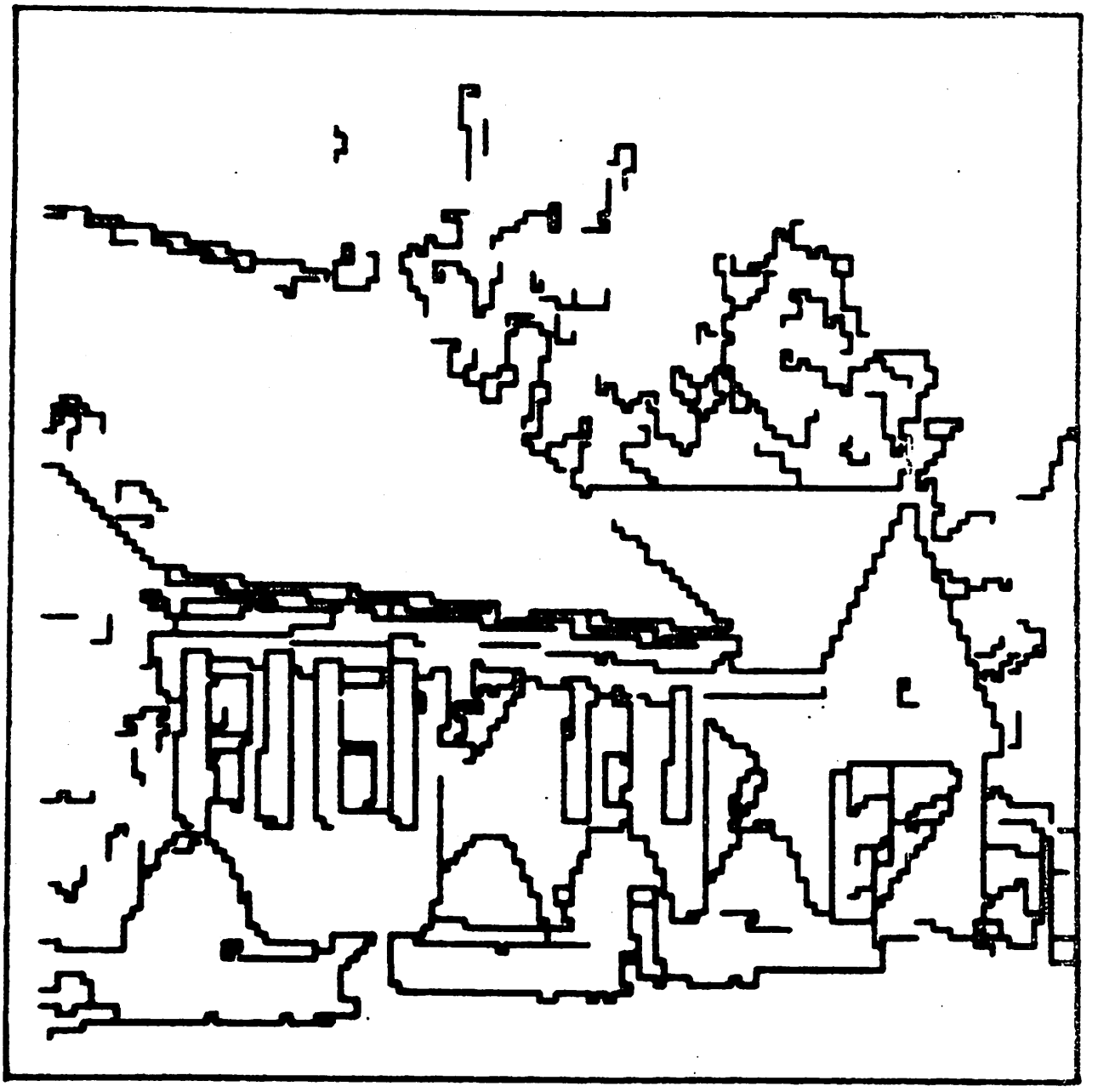
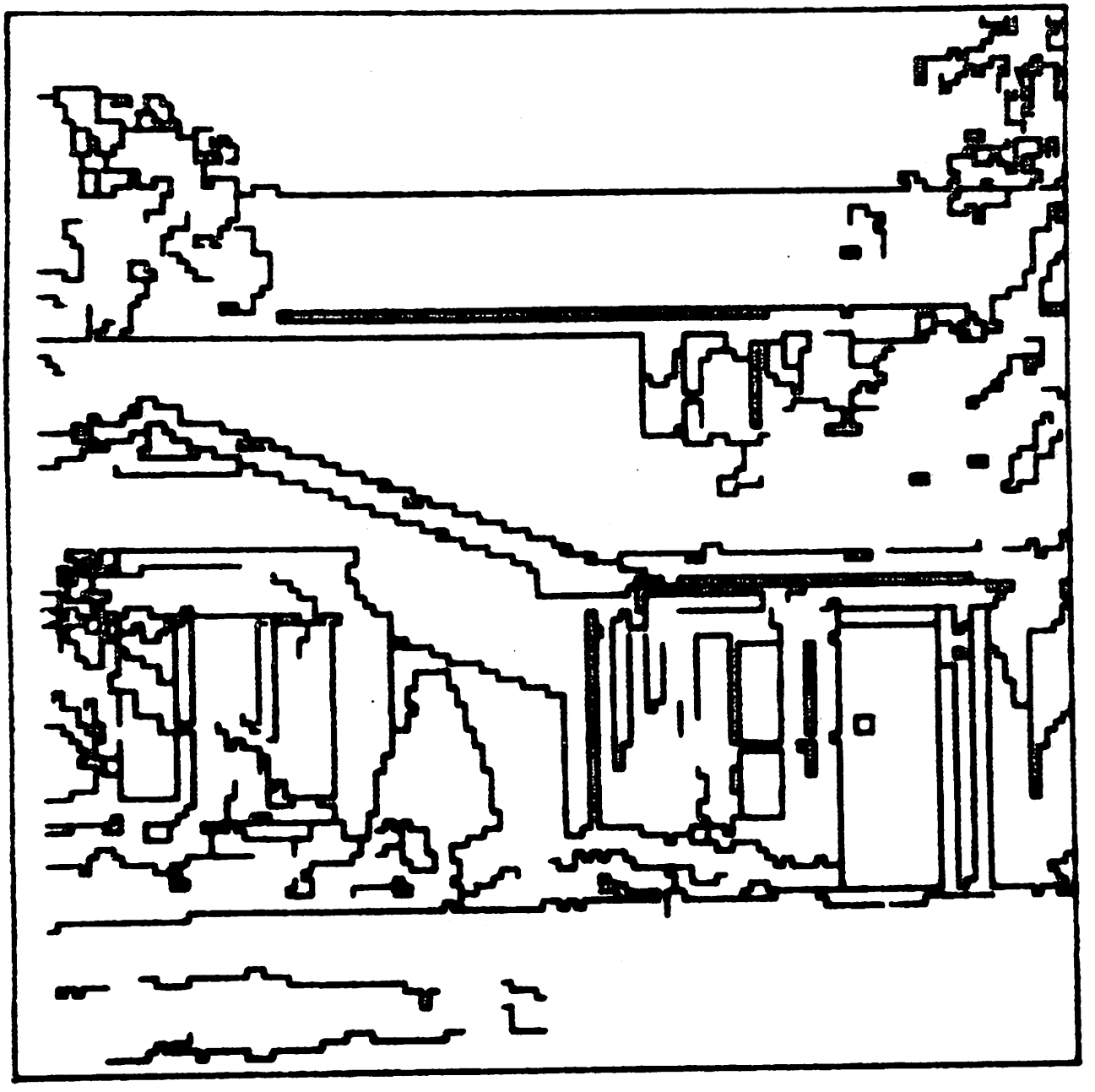


Figure 14. BD should be eliminated as a low-confidence edge. AB and AC should be merged to form AC which, being longer, will be more confident than AB or BC.



13a

Figure 13. Postprocessing. In Figure 13a the short edges and most of the smallest (1-pixel) regions have been removed from the data in Figure 10b.



13b.

Figure 13b. Postprocessing applied to the data in Figure 10d.

the line confidence will reflect the global difference between the regions in the neighborhood of the line.

For any line segment L we consider sets of pixels S_1 and S_2 on either side of L. We now test the hypothesis H_1 that the points in S_1 and S_2 belong to different regions, against H_0 that points in S_1 and S_2 belong to the same region. This statistical test is an extension of Yakimovsky [10], but the manner in which it is employed differs in an important respect. Yakimovsky used the test to form the edges which comprise his boundaries. For each edge calculation, a predetermined neighborhood was used for selecting the points in S_1 and S_2. Each edge was processed independently of each other, so the sets S_1 and S_2 taken each time did not necessarily accurately reflect the region structure formed by the boundary as a whole. In our case, on the other hand, the entire boundary is available for processing, and this allows a better determination of the pixels which are to contribute to the analysis.

Let S_0 be $S_1 \cup S_2$. Under hypothesis H_0, S_0 is one region and will be modelled by the normal distribution $N(\mu_0,\sigma_0)$; under hypothesis H_1, S_1 and S_2 can be modelled by $N(\mu_1,\sigma_1)$ and $N(\mu_2,\sigma_2)$ respectively. A maximum likelihood analysis leads to

$$P(L) = \frac{(\sigma_0^2)^{n_0}}{(\sigma_1^2)^{n_1}(\sigma_2^2)^{n_2}}$$

as a measure of the confidence of L, where $|S_i| = n_i$, $i = 0,1,2$, and $n_0 = n_1 + n_2$ [10].

Let $S_i = \{(X_{ij},Y_{ij},I_{ij}),\ j=1\ldots n_i\}$, $i = 0,1,2$, where I_{ij} is the intensity of pixel (X_{ij},Y_{ij}) in set S_i. The model described above assumes

the I_{ij} are normally distributed about mean μ_i, independent of their location in the image, i.e., it is supposed that the readings I_{ij} are governed by the probability distributions

$$\frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{(I_{ij}-\mu_i)^2}{2\sigma_i^2}}$$

independently of (X_{ij},Y_{ij}). We can improve the test procedure by using a more sophisticated statistical model of the regions on either side of the putative segment L.

Rather than assume that there are no spatial dependencies in the gaussian distribution of the I_{ij} in each region, we will improve the model by assuming that μ_i can vary linearly across the region. This leads us to the concept of a <u>dynamic</u> mean μ^*, which is a function of the spatial coordinates, and represents the expected value of the intensity at a given point in the region. Our model should be better than the simpler one since object brightnesses in real world images are not usually constant but vary across the surfaces concerned. We will suppose that at each point (X_{ij},Y_{ij}) the intensity is normally distributed about the dynamic mean $\mu_i^*(j) = \mu_i + a_i X_{ij} + b_i Y_{ij}$. This assumes that there is a constant gradient of intensity (strength a_i in the X-direction, b_i in the Y-direction) across the region, which will be fairly realistic, at least over small parts of the region. To ensure the accuracy of the analysis, then, we see that the sets S_1 and S_2 should be composed of pixels lying within a short distance of L.

The values of a_i and b_i for $i = 0,1,2$ will be determined by a least squares fit. In those cases where there are no intensity gradients,

a_i and b_i will evaluate to zero, showing that our approach covers all those cases where the simpler analysis would have been sufficient.

For each of the sets $S_i = \{(X_{ij}, Y_{ij}, I_{ij})\}$ we perform the following computations:

1) Determine constants a_i, b_i, μ_i which minimize

$$\sum_{S_i} [I_{ij} - (\mu_i + a_i X_{ij} + b_i Y_{ij})]^2$$

2) Determine the variances $(\sigma_i^*)^2$ by

$$(\sigma_i^*)^2 = \sum_{j \in S_i} \frac{(I_{ij} - \mu_i^*(j))^2}{n_i}$$

The measure of the confidence of L is now given by

$$P(L) = \frac{(\sigma_0^*)^{2n_0}}{(\sigma_1^*)^{2n_1}(\sigma_2^*)^{2n_2}}$$

or, alternatively, but preserving the ordering of the confidences of the various line segments,

$$P(L) = n_0 \log \sigma_0^* - n_1 \log \sigma_1^* - n_2 \log \sigma_2^*.$$

V.3 TRIM2 -- Low Confidence Line Removal

Line segments can be removed on the basis of their relative confidence ratings by removing lines with confidences less than some threshold T. This process should be performed conservatively (with a low threshold) for the following reason. Consider Figure 14 and suppose that AB and BC have average confidences, while that of BD is relatively low. Any reasonable threshold should get rid of BD, but if it is set too high there is a danger that either AB or BC could be removed as well.

This is to be avoided because as soon as BD is gone, ABC might become a single line segment with a much higher confidence than that of either AB or BC alone.

The process of removing weak and uncertain line segments may be performed iteratively. Instead of immediately adjusting threshold T to what is thought to be an optimal level, it can first be set to a lower level T'. Lines of confidence < T' will be removed, stage III grouping reapplied, T' increased somewhat and the whole process repeated until the desired level is reached. In fact, regrouping could occur after each line segment is removed.

VI. A Heuristic Confidence Measure

In order to determine if there was any way of speeding up the confidence analysis presented in V.2 we examined the relative effectiveness of a simple heuristic confidence measure.

Let L be the length of a line segment, and L^* the length of what one would call a "long" line -- say $\frac{1}{4}$ the width of the image. Let C_i be the average contrast across the line and C^* the maximum average contrast one would expect to see in an image -- say 3/4 of the difference between the extremes of the intensity scale. Then define

$$\hat{L}_i = \text{normalized length} = \text{Min}(L_i, L^*)/L^*$$

$$\hat{C}_i = \text{normalized contrast} = \text{Min}(C_i, C^*)/C^*$$

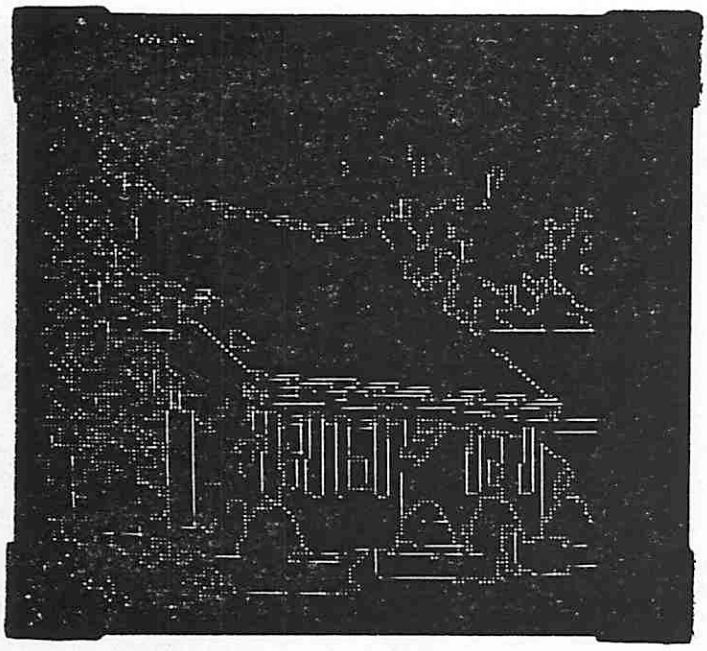
Our heuristic confidence measure for the line is given by

$$f = \hat{L}_i + \hat{C}_i - \hat{L}_i\hat{C}_i.$$

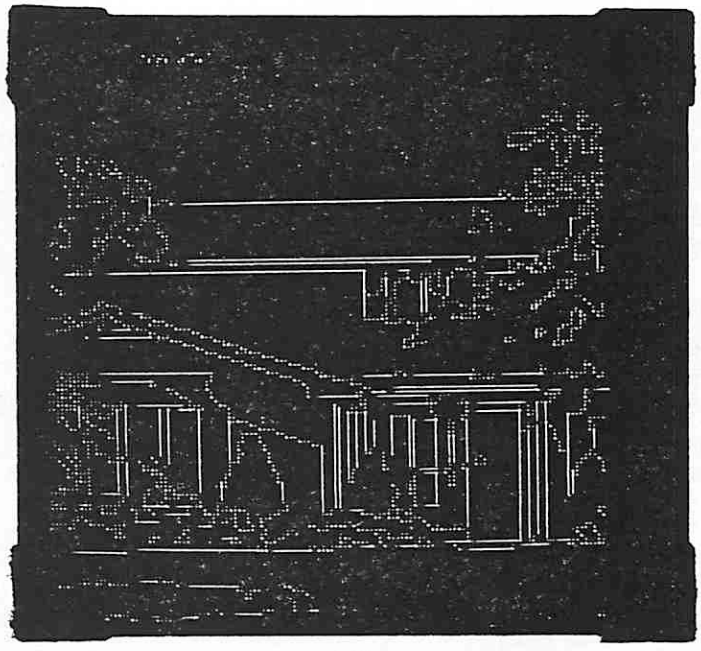
This value is 1 if either $\hat{L}_i$ or $\hat{C}_i$ alone equal 1, zero if both are zero, and is monotone increasing in $\hat{L}_i$ and $\hat{C}_i$ for all other values.

In Figures 15-18 we compare this confidence measure with that described in V.2. The pictures display all those line segments with confidence greater than a suitably scaled threshold value. It will be seen that both methods do equally well with regards assigning high confidence measures to long high-contrast lines, such as the sides of windows or houses, and low-confidence values to many of the boundaries of texture elements in the trees and shrubbery.

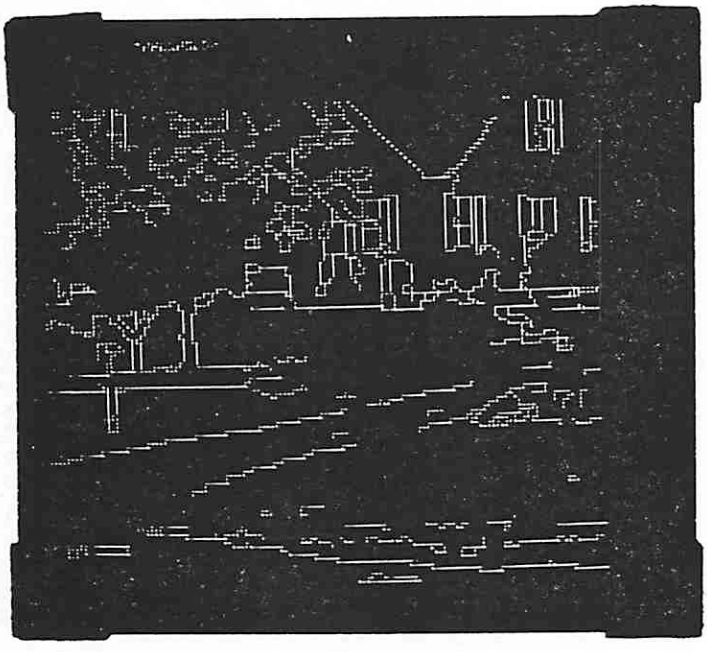
This does not mean that the "sophisticated" analysis of V.2 is no good. On the contrary, it sets a standard with which our heuristic measure may be compared. The heuristic shows itself to produce very acceptable results, and if it proves to be generally reliable, it is recommended over the other method due to computational efficiency.



15a

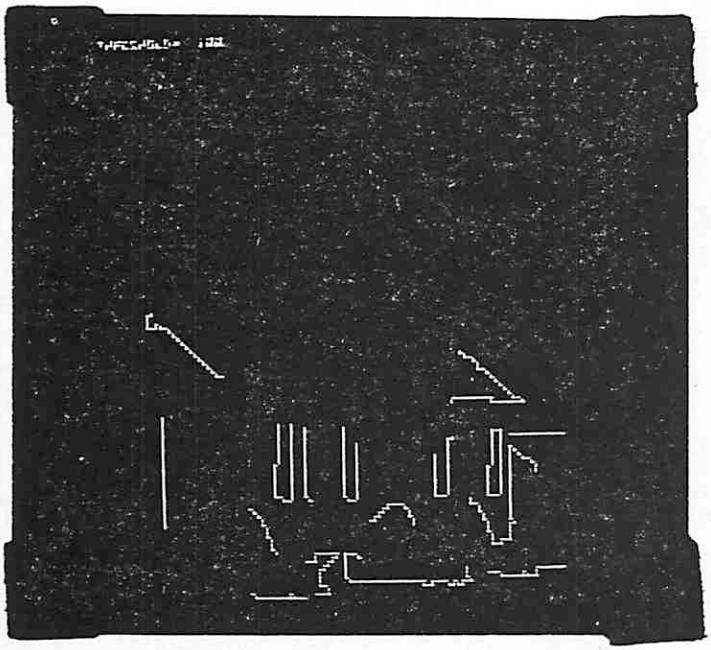


15b

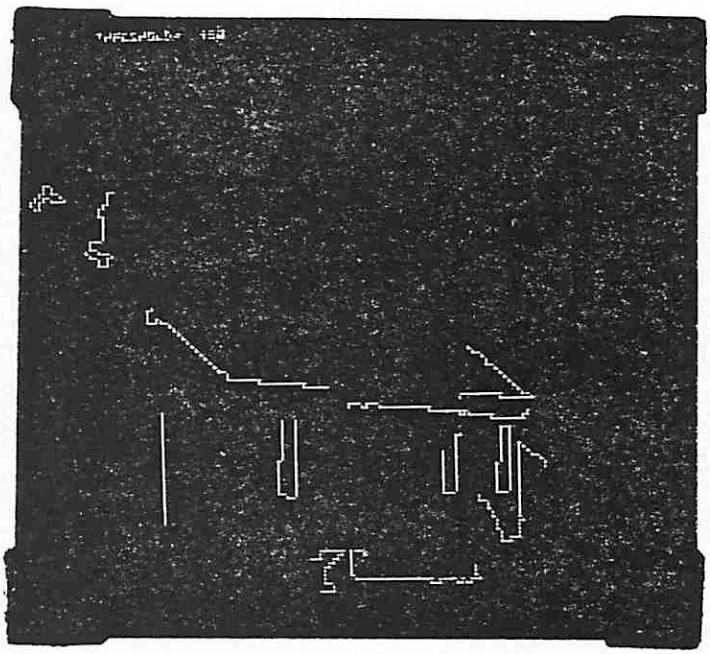


15c

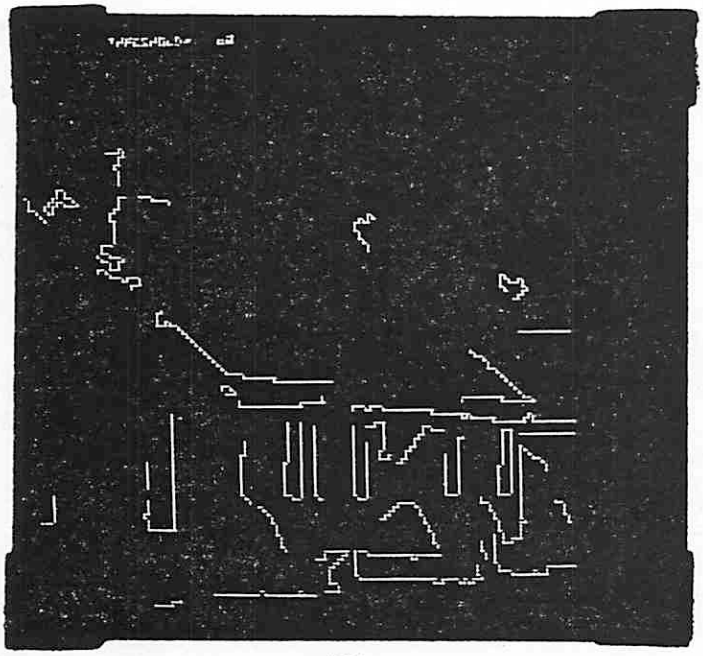
Figure 15. Edge images which are used to compare the two confidence measures described in the text. In Figures 16 - 18 these images are shown thresholded at different levels. Due to the different scales produced by the two measures, an exact comparison is impossible, but sequences of comparable threshold values were chosen for the two algorithms.



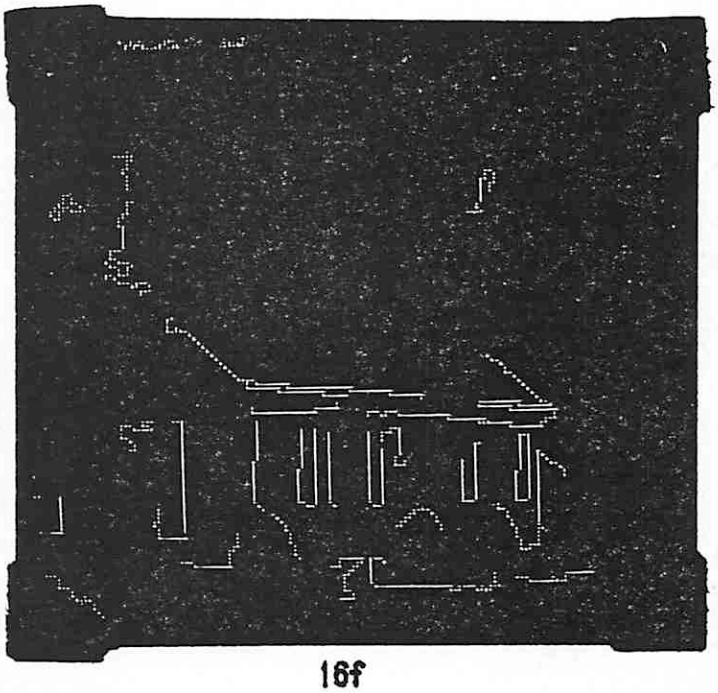
16a



16e

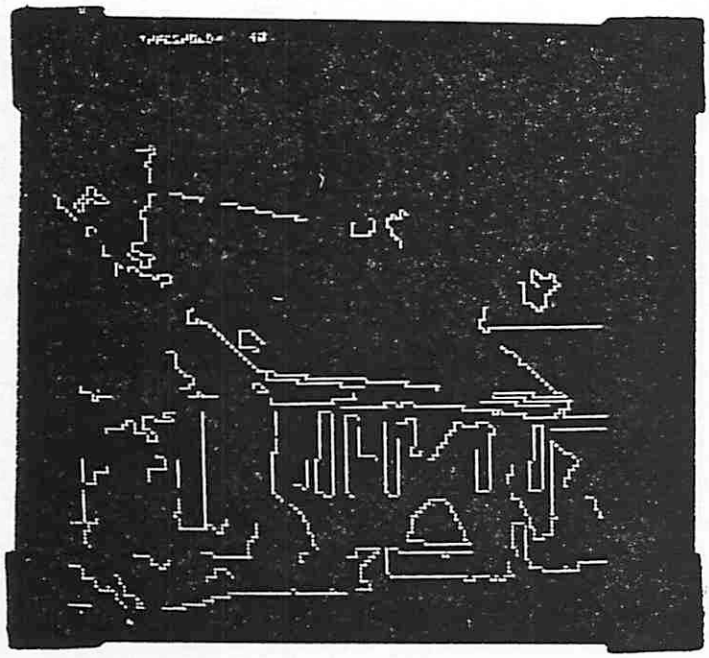


16b

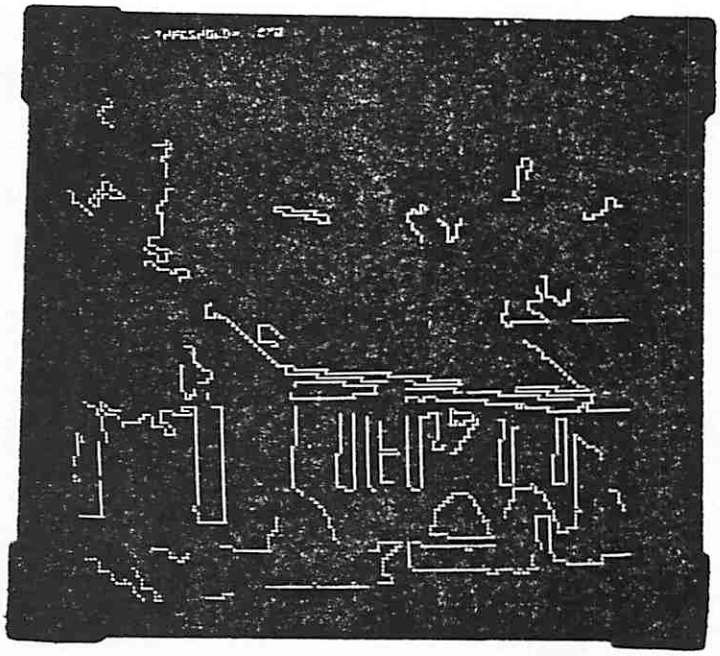


16f

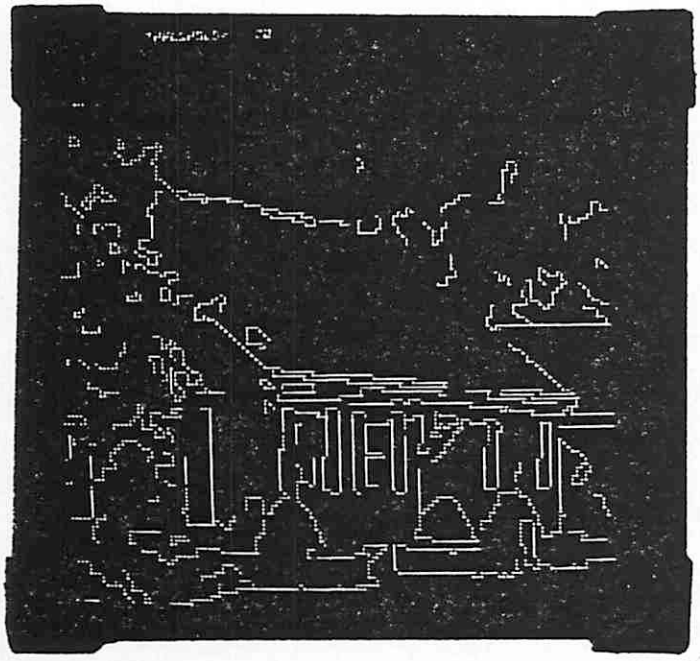
Figure 16. Successive thresholdings of the edge image in Figure 15a. Figures 16a - 16d show line-segments with statistical confidence measures under successively decreasing thresholds. Figures 16e - 16h show the same with heuristic confidence measures.



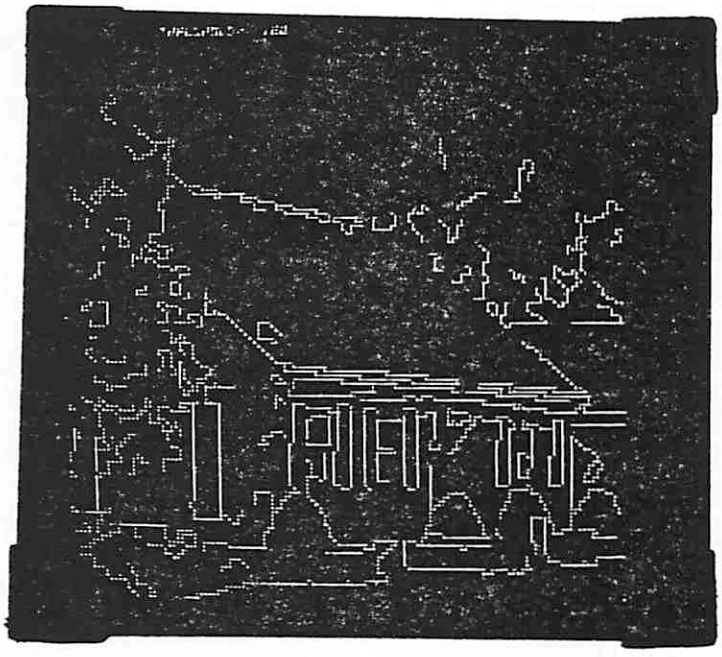
16c



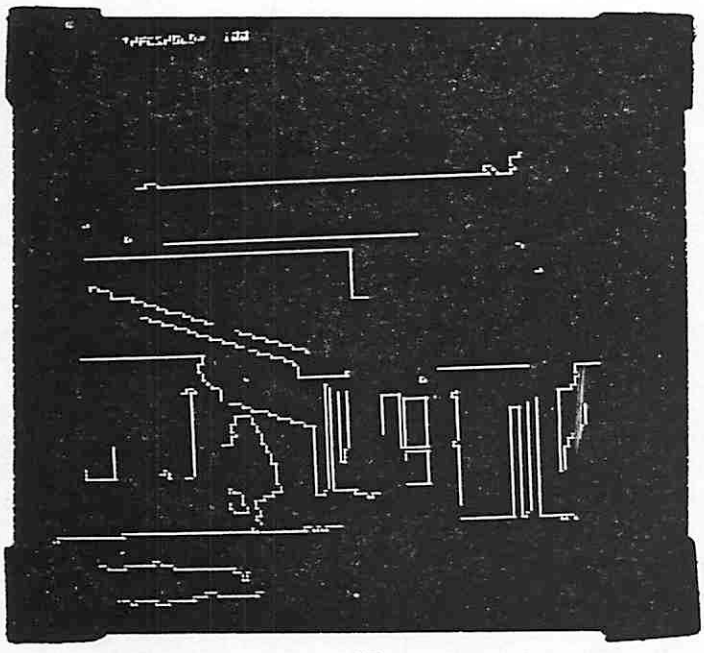
16g



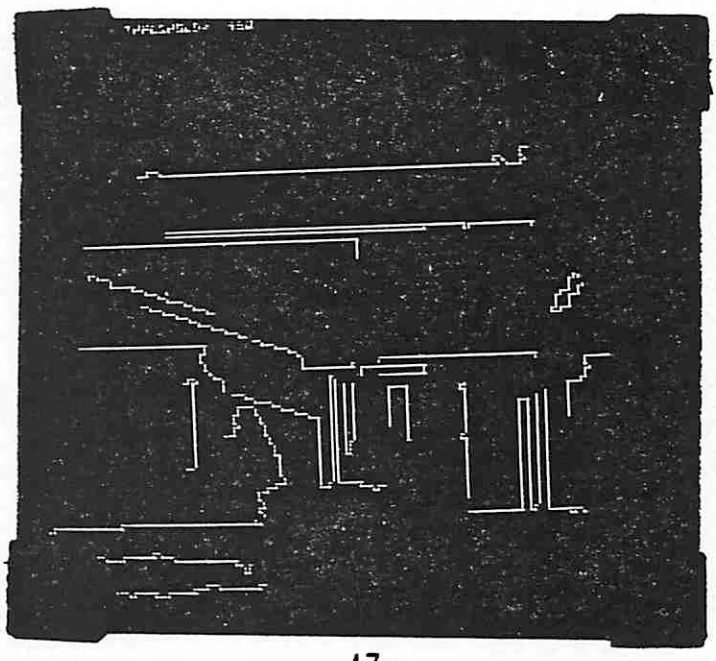
16d



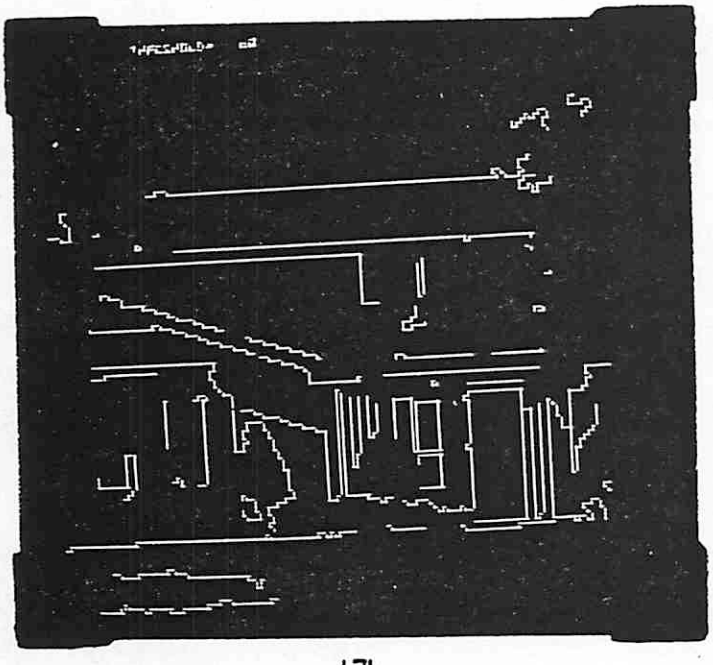
16h



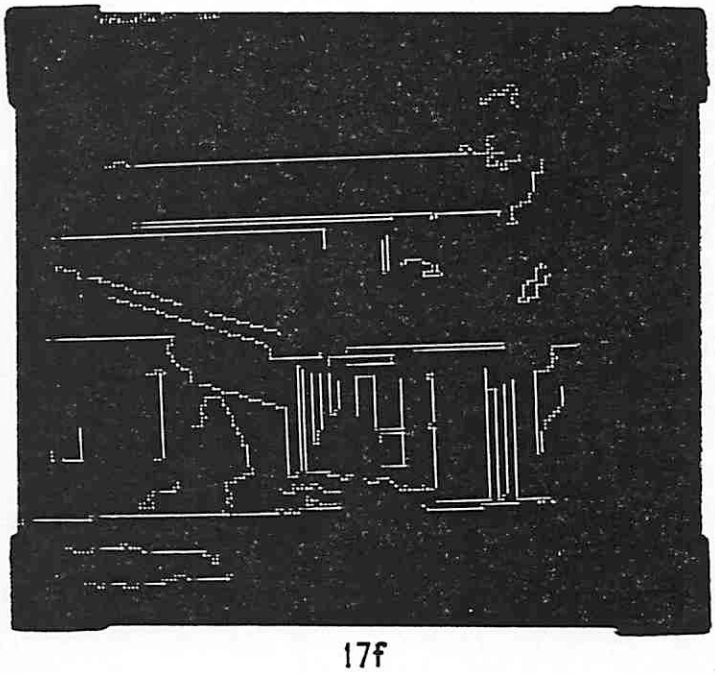
17a



17e

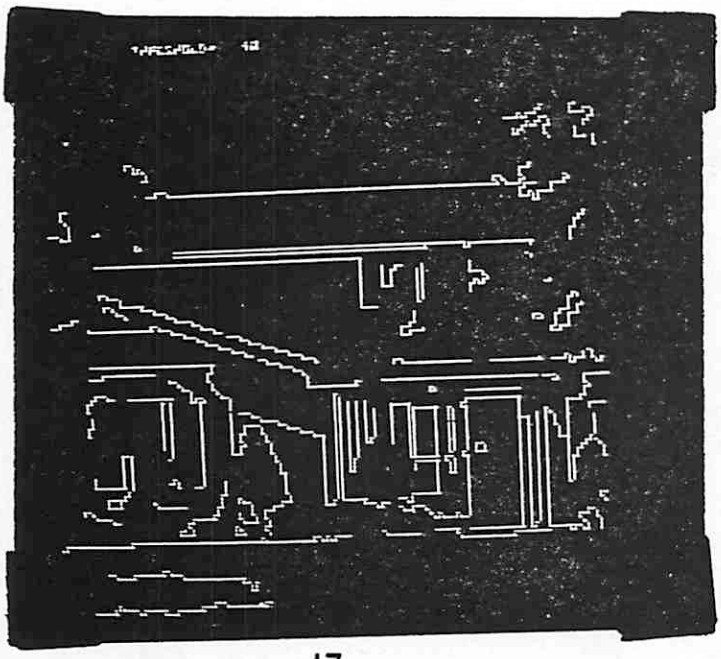


17b

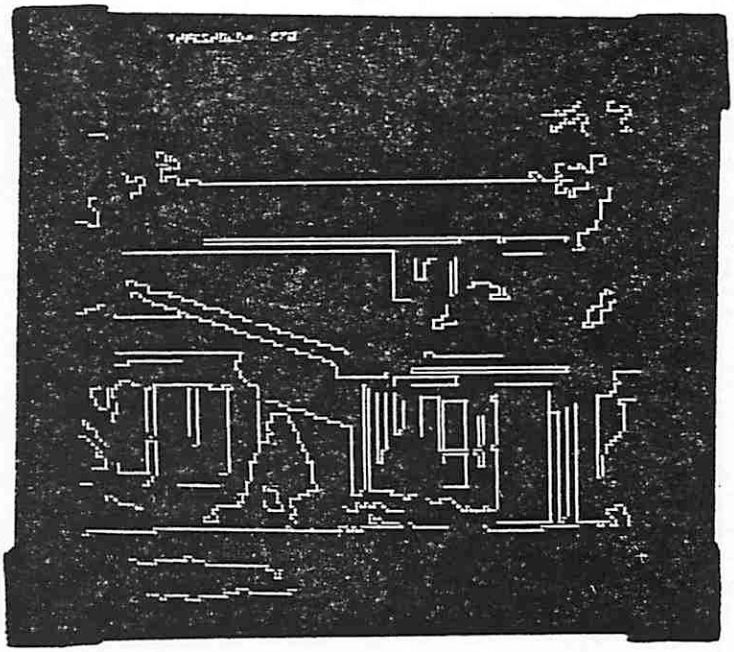


17f

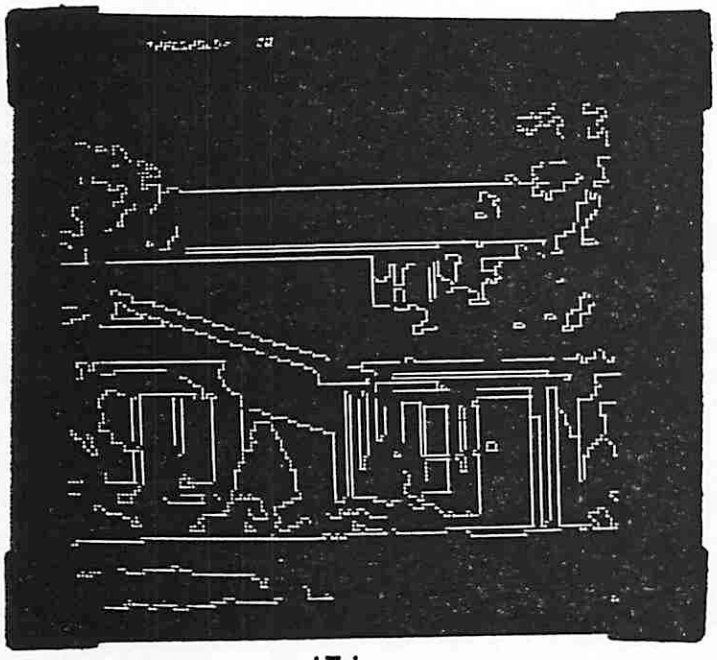
Figure 17. Successive thresholdings of the edge image in Figure 15b. Figures 17a - 17d show line-segments with statistical confidence measures under successively decreasing thresholds. Figures 17e - 17h show the same with heuristic confidence measures.



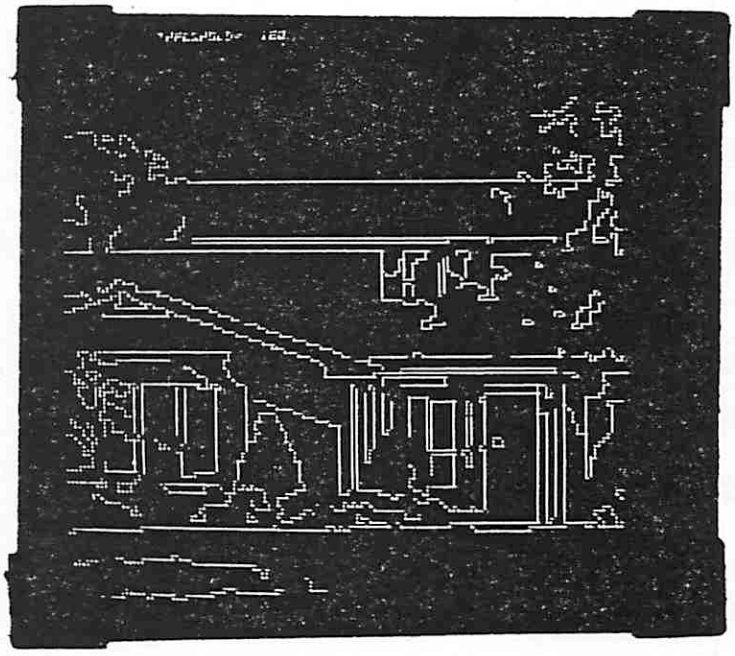
17c



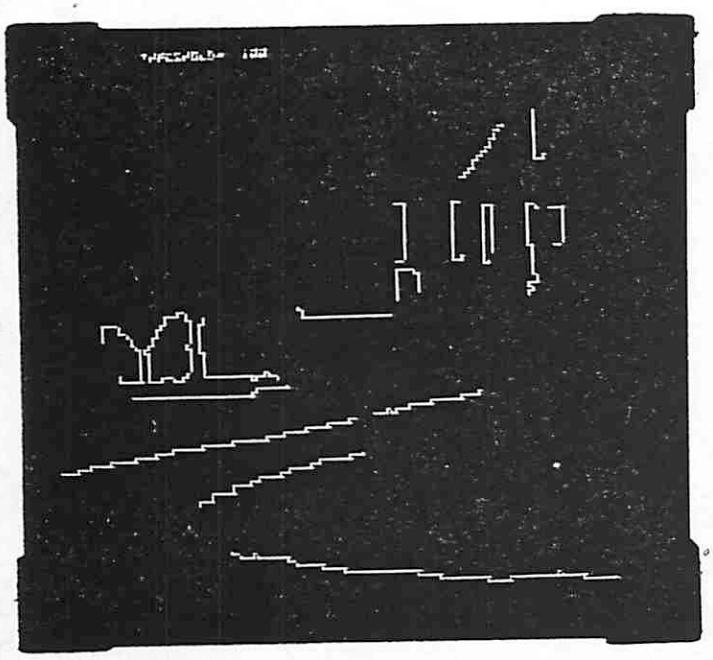
17g



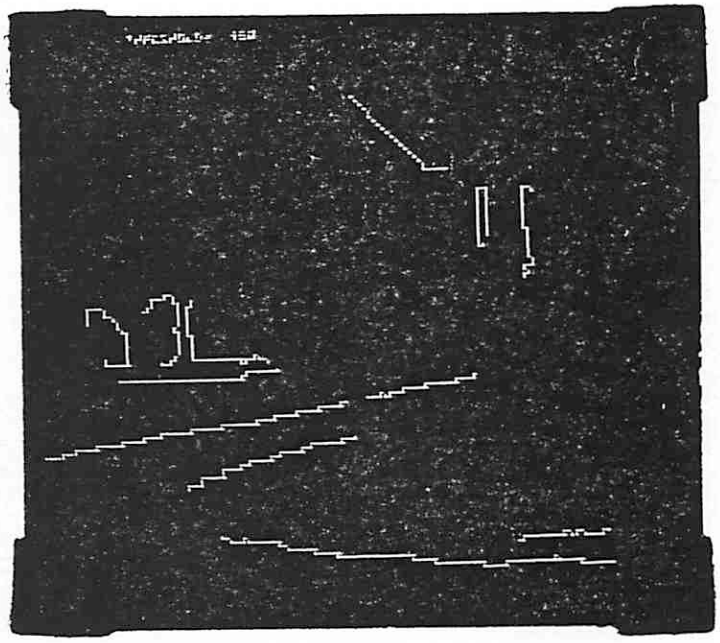
17d



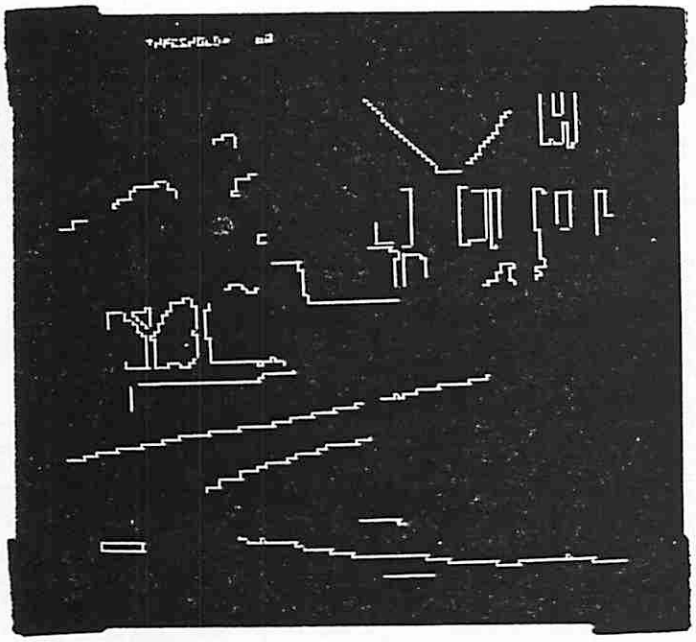
17h



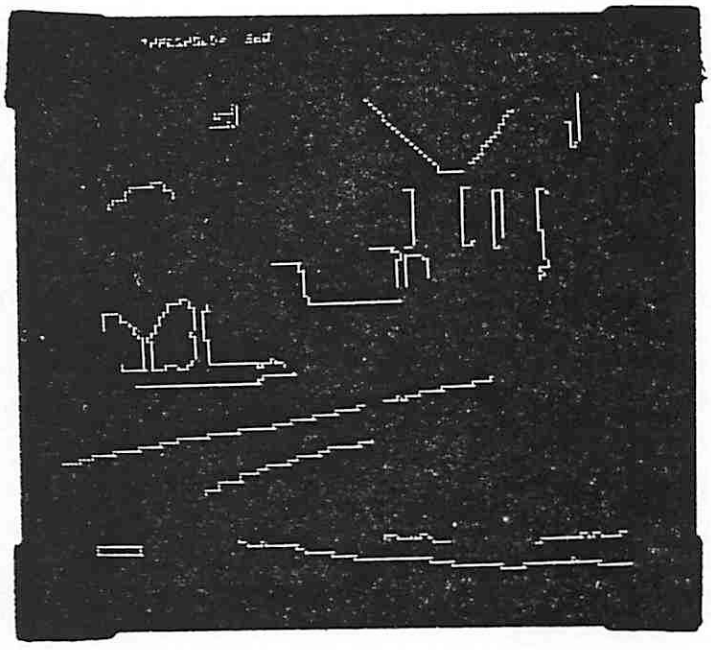
18a



18e

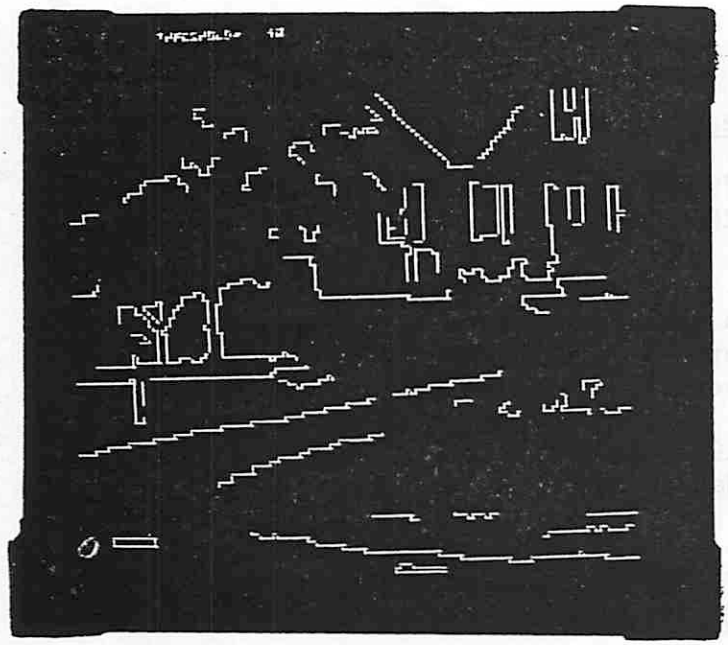


18b

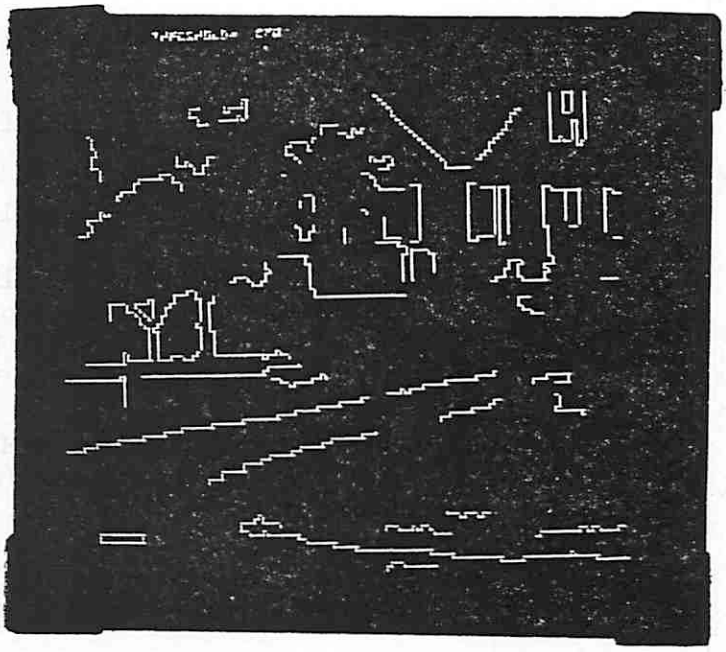


18f

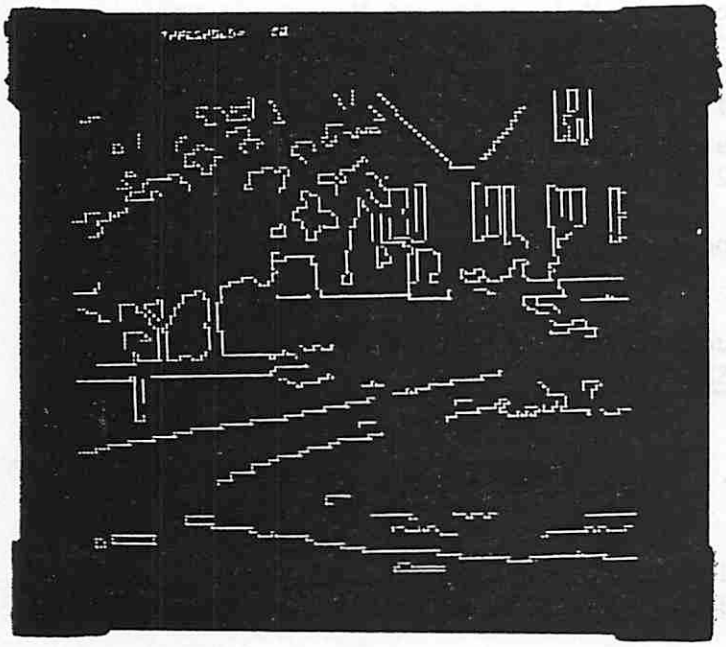
Figure 18. Successive thresholdings of the edge image in Figure 15c. Figures 18a - 18d show line-segments with statistical confidence measures under successively decreasing thresholds. Figures 18e - 18h show the same with heuristic confidence measures.



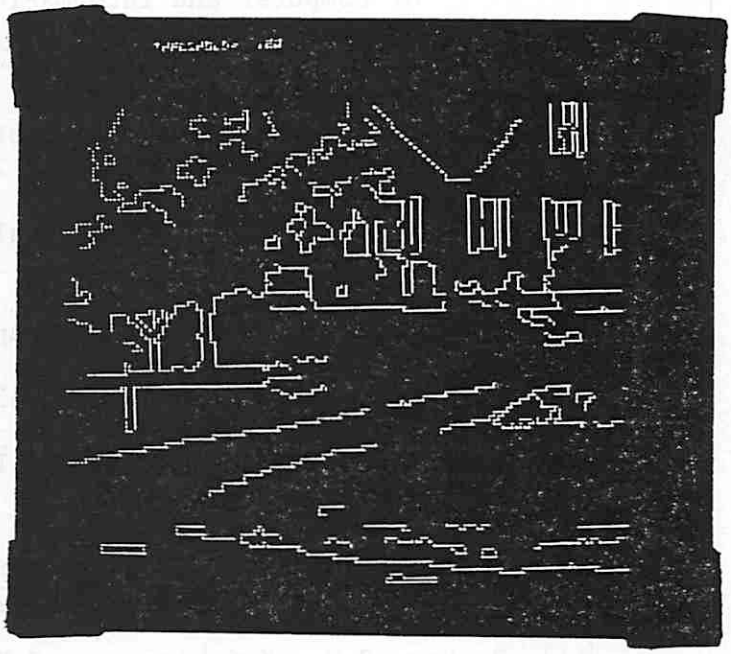
18c



18g



18d



18h

References

[1] Brice, C.R., and Fenema, C.L. 1970. Scene Analysis Using Regions. Artificial Intelligence 1: 205-226.

[2] Hanson, A., and Riseman, E. 1976. A Progress Report on VISIONS: Representation and Control in the Construction of Visual Models. COINS Technical Report 76-9, Department of Computer and Information Science, University of Massachusetts, Amherst.

[3] Hanson, A.R., and Riseman, E.M. 1978. Segmentation of Natural Scenes. Computer Vision Systems (A. Hanson and E. Riseman, Eds.) Academic Press, New York.

[4] Lesser, V.R., and Erman, L.D. 1977. A Retrospective View of the HEARSAY-II Architecture. IJCAI-5, MIT, Cambridge, MA.

[5] Prager, J.M. 1977. Extracting and Labelling Boundary Segments in Natural Scenes. COINS Technical Report 77-7, Department of Computer and Information Science, University of Massachusetts, Amherst.

[6] Riseman, E.M., and Arbib, M.A. 1977. Computational Techniques in the Visual Segmentation of Static Scenes. Computer Graphics and Image Processing 6: 221-276.

[7] Rosenfeld, A. 1977. Technical Report, Computer Science Center, University of Maryland.

[8] Rosenfeld, A., Hummel, R.A., and Zucker, S.W. 1976. Scene Labeling by Relaxation Operations. IEEE Trans. on Systems, Man and Cybernetics SMC-6: 420-433.

[9] Rosenfeld, A., and Kak, A.C. 1976. Digital Picture Processing, Academic Press, New York.

[10] Yakimovsky, Y. 1976. Boundary and Object Detection in Real World Images. Journal of the ACM 23(4): 599-618.

[11] Zucker, S.W., Krishnamurty, E.V., and Hoar, R.L. 1976. Relaxation Processes for Scene Labelling: Convergence, Speed and Stability. Technical Report 477, Computer Science Center, University of Maryland.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: COINS TR 78-17	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): Extracting and Labelling Boundary Segments in Natural Scenes (Revised and Updated)	5. TYPE OF REPORT & PERIOD COVERED: Interim
6. PERFORMING ORG. REPORT NUMBER	
7. AUTHOR(s): John M. Prager	8. CONTRACT OR GRANT NUMBER(s): ONR N00014-75-C-0459
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Arlington, Virginia 22217	12. REPORT DATE: 9/78
	13. NUMBER OF PAGES: 45
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer vision, image processing, segmentation, boundary analysis, relaxation, case analysis, labelling, confidence generation

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

This paper describes a set of algorithms used to perform segmentation of natural scenes through boundary analysis. The techniques include preprocessing, differentiation using a very simple operator, relaxation using case analysis, and postprocessing. The system extracts line segments as connected sets of edges, labels them, and computes features for them such as length and confidence.