Local Thresholding Using Convergent Evidence

Amar Mitiche
Larry S. Davis

Computer Sciences Department
University of Texas
Austin, Texas 78712

TR-81-4 June 1981

This research was supported in part by the Air Force Office of Scientific Research under Grant AFOSR 81-0080.

1. INTRODUCTION

In [1] Milgram presents a thresholding procedure which, on an object-by-object basis, computes a threshold which maximizes a measure of contrast at the boundaries of the thresholded components. The power of this procedure can be attributed to its emphasis on locality (since each object is analyzed individually) and convergence of evidence (since both edge and intensity information is considered). The procedure does, however, have relatively high computational cost since each object must be analyzed separately. Furthermore, the unknown, non-uniform spatial distribution of objects in the image would make it difficult to decompose the computation into a large number of smaller, parallel computations.

In [2], Chow and Kaneko develop a thresholding procedure based on partitioning an image into square windows, assigning thresholds to those windows having clearly bimodal histograms, and finally computing an interpolating threshold surface over the image, based on the thresholds computed for the bimodal regions.

This paper describes a thresholding procedure which represents a synthesis of the work described in [1] and [2], by adapting Milgram's algorithm to operate on a regular decomposition of the image.

The strengths of the proposed algorithm are that:

1) The regular decomposition of the image allows for parallel computation on available image processing machines.

2) An initial set of windows for segmentation are selected on the basis of a global analysis of the image (essentially those subregions having the largest number of edge points) and the results of those segmentations are propagated to other windows (where they serve as "advice") which would have been otherwise difficult to segment.

Section 2 describes the algorithm; section 3 contains several examples of the application of the procedure to various images, and section 4 is a brief summary.

2. DESCRIPTION OF THE ALGORITHM

We are given a digitized picture function $f(i,j)$ with $1 \leq i, j \leq n$ and a partition of the picture into nw x nw windows. A threshold level is determined for each window following the steps of the algorithm described below.

(a) Initial Computation of Thresholds

At this initial step, each window is examined and its edges counted. These edges could be the result of applying a simple, computationally efficient contrast sensitive operator. For example, edges in images with highly contrasting background and foreground can be computed using a simple fixed neighborhood gradient operator. For noisy, textured images,

an operator such as e_k[3] (which is used in this report) would be more reliable.

On the basis of the number of edges in the window, a decision is taken as to whether a threshold should be computed for this window. If the edge map for a window is not dense, it is an indication that most of the pixels in the window belong to the background or foreground. Thus a direct computation of a threshold for such a window from the gray levels of pixels it contains will not be reliable. The threshold selection for windows that have too few edges is performed at a later step of the algorithm and will be discussed subsequently. The level at which the number of edges in a window is declared too low to allow a reliable direct computation of a threshold is a parameter that must be determined. In the experiments reported in this report, this level is equal to the average number of edges per window as measured over the entire image. In this way, assuming that the number of edges in a fixed size neighborhood has a symmetric distribution, about half of the windows in the image will be chosen for an initial threshold assignment.

For those windows that have enough edges, a set of thresholds is selected and for each threshold a segmentation is computed. In this report, the thresholds are automatically selected by dividing the segment $[\mu - \delta, \mu + \delta]$ into a given number of intervals. Here, μ is the mean gray level in the

window and δ is the standard deviation. The quality of each segmentation is then measured and a best threshold finally selected. The quality of a segmentation can be related to how well its borders match the edge map for the window [1]. This method suffers of course from the fact that results are at most as reliable as the edge maps that are given for the windows. Instead, a measure of quality based on the results of an edge sensitive operator such as contrast relieves one from the burden of having to compute explicitely a reliable edge map to be used for the purpose of matching with the segmentation borders. Given an image function $f(i,j)$, $1 \leq i, j \leq n$ and a set $D = \{d_i, d_j\}$ of displacements, contrast in a subset S of the image can be defined by the following expression:

$$C = \sum_{(d_i,d_j)\in D} \sum_{(i,j)\in S} \left| f(i,j) - f(i+d_i, j+d_j) \right|$$

Experimental results using contrast computed for each window at the borders of the segmentation are described in this report.

(b) Threshold Modification

Because thresholds in step (a) are obtained by dividing an interval into a fixed number of subintervals, it is possible that better threshold values will not be selected. This step is intended to correct this problem. Let t_i be the threshold

for window W_i and let S_i be the best segmentation obtained for this window by thresholding at t_i. If the set of 4-neighbors of W_i that have a threshold assigned at the previous step of this algorithm is not empty, then the thresholds of these windows are used to compute segmentations $S_1', \ldots, S_k'$ (with $1 \leq k \leq 4$) for window W_i. Let S_ℓ' be the best (always in the sense discussed earlier) segmentation and t_ℓ' the corresponding threshold. If S_ℓ' is better than S_i then t_ℓ' is substituted for t_i.

(c) Window Sliding

Let W be a window for which no threshold has been computed yet and let (iw, jw) be the coordinates of the upper left hand corner of W. The operations described in step (a) of this algorithm are performed on the windows with upper left hand corner coordinates (iw - nw/2, jw), (iw + nw/2, jw), (iw, jw - nw/2) and (iw, jw + nw/2). Figure 1 shows these windows. If the subset of these windows for which a threshold has been computed is not empty, the best of these thresholds is assigned to window W. Such a procedure for threshold assignment is most desirable in cases as the one illustrated in Figure 2. This figure shows a picture composed of a background labeled B and a foreground labeled F. There are 4 windows numbered 1 through 4. A threshold will be properly assigned to windows 1 and 3 by looking at the pixels of the windows delineated by the dashed lines.

(d) Threshold Propagation

The remainder of the algorithm consists in iteratively assigning to those windows without a threshold, the best of the thresholds (if any) of their 4-neighbors. This case is illustrated in Figure 3 which shows a picture with a background B and a foreground F. Windows 1 and 3 will be properly thresholded using the threshold levels of windows 2 and 4.

Appendix A contains a summary of the algorithm described in this section.

3. EXPERIMENTS

The thresholding algorithm described above was run on a set of six textures. Figure 4a shows the pattern of a grating. The thresholded windows (black and white) obtained after the initial threshold assignment/modification, window sliding and threshold propagation steps successively overwrite the gray scale picture in Figure 4b-d. These windows are 16 pixels x 16 pixels. Total contrast at the borders, as defined previously, is the criterion for measuring segmentation quality. As a comparison, Figure 4e shows the result of thresholding the entire picture at the average gray level. Notice that in this case the borders in the upper half of the image are either missing or significantly thinner than the borders in the lower half of the image. This is in contrast with the uniform width borders of figure 4d obtained using the algorithm described in this report. Figures 5a - e, 6a - e, 7a - e, 8a - e, show

similar results for the orchard, brick, concrete and pebble textures. The performance of local thresholding using convergence of evidence is significantly better than the performance of global thresholding for the orchards and bricks. Comparable results are obtained for concrete and pebbles with both methods. Figure 9a is the pattern of concrete of figure 7a to which a gray scale gradient has been added (at a rate of 1 gray scale unit every 3 rows, starting with the top row). The results of local thresholding using convergence of evidence are shown in Figures 9b - d. The windows are 8 pixels x 8 pixels. Global thresholding at the average gray level eliminates most of the borders in the image. This is illustrated in Figure 9e.

Table 1 shows the variation of local thresholds for the orchard pattern.

4. CONCLUSION

Thresholding is an important tool in many image segmentation tasks. The goal of this study was to design a computationally efficient thresholding procedure that combines the advantages of both local computation and the use of convergence of evidence. The regular decomposition of the image means processing simplicity and allows parallel computation on available image processing hardware.

REFERENCES

[1] D. L. Milgram, "Region Extraction Using Convergent Evidence," Computer Graphics and Image Processing, 11, 1979, pp 1-12.

[2] C. K. Chow and T. Kaneko, "Automatic Boundary Detection of the Left Ventricule from Cineangiograms," Comp. Biomed. Res., 5, 1972, pp 388-410.

[3] A. Mitiche and L. S. Davis, "Theoretical Analysis of Edge Detection in Textures," 5th International Conference on Pattern Recognition, Miami Beach, Florida, December 1-4, 1980, pp 540-547.

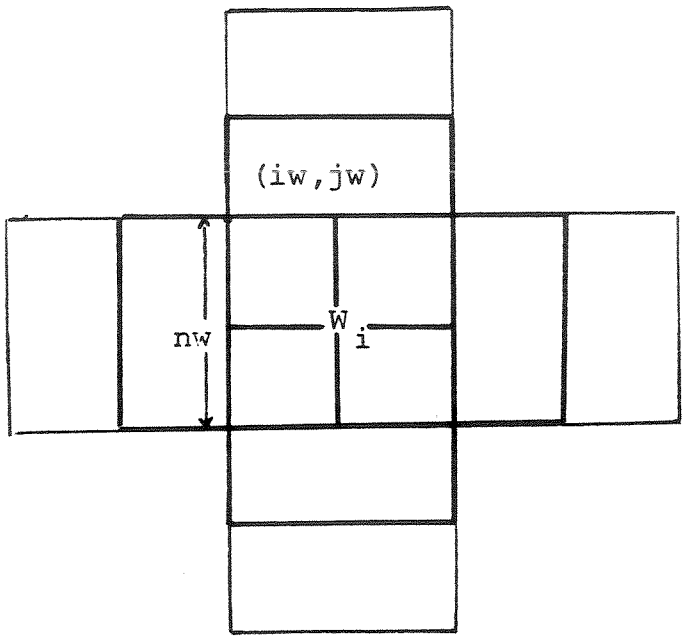


Figure 1.

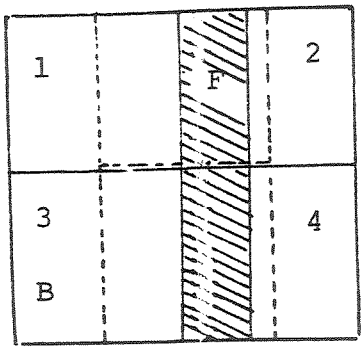


Figure 2.

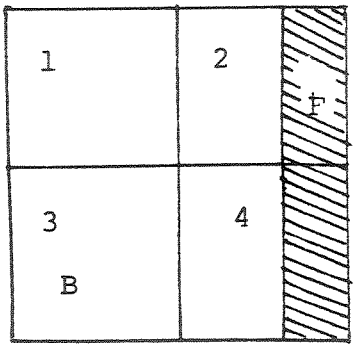
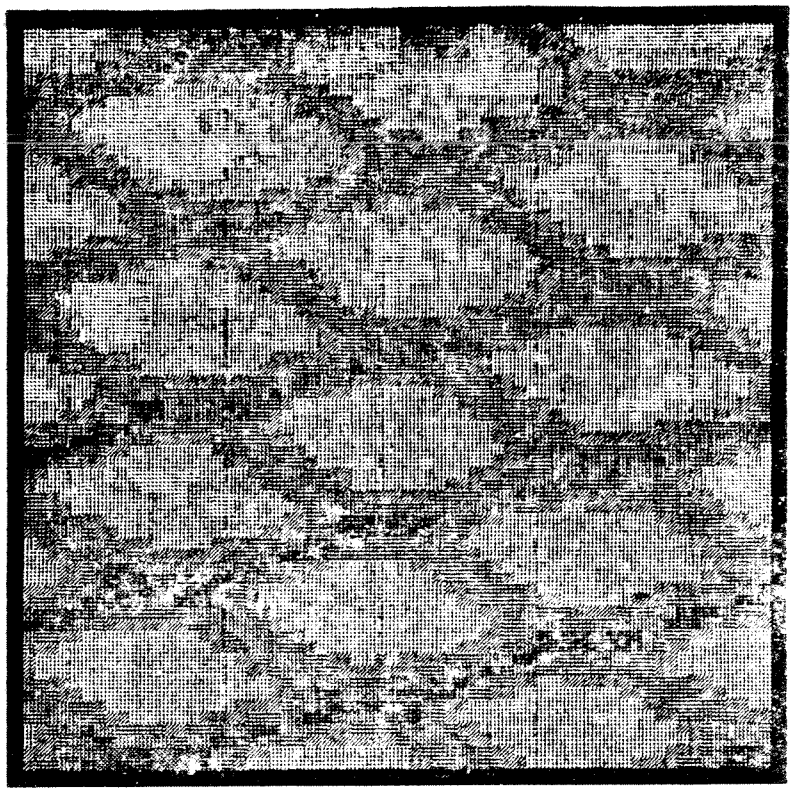


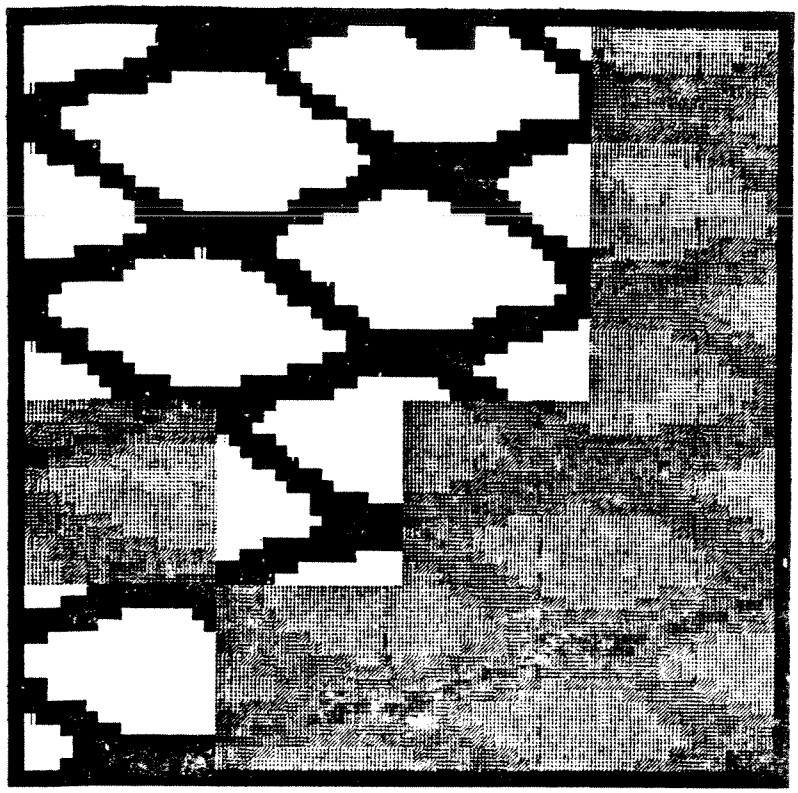
Figure 3.



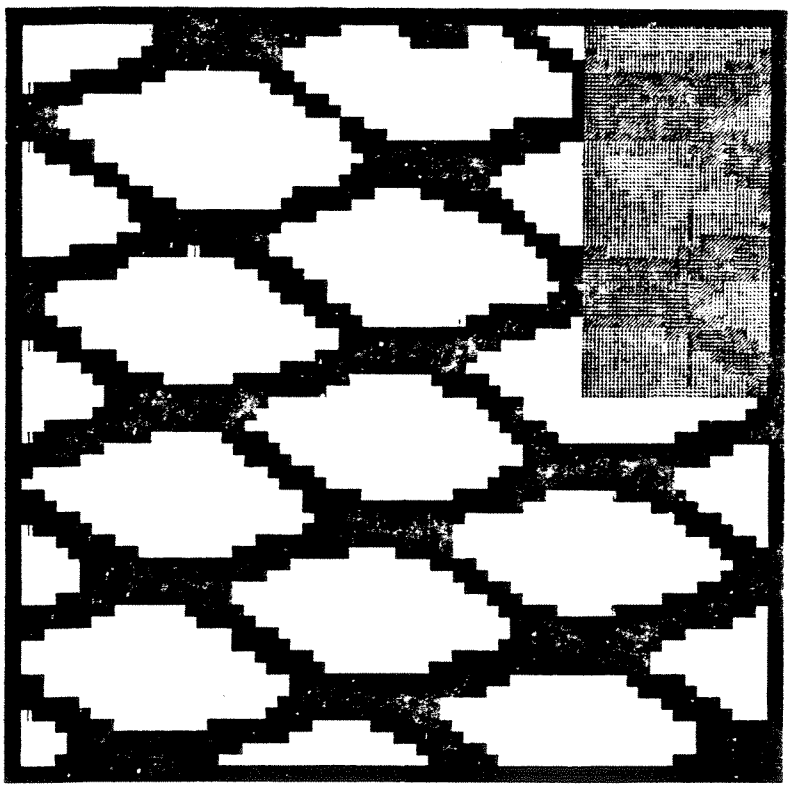
(a)

Figure 4.

(a) Grating
(b) After steps (a) and (b)
(c) After step (c)
(d) After step (d)
(e) Globally thresholded image at the average gray level

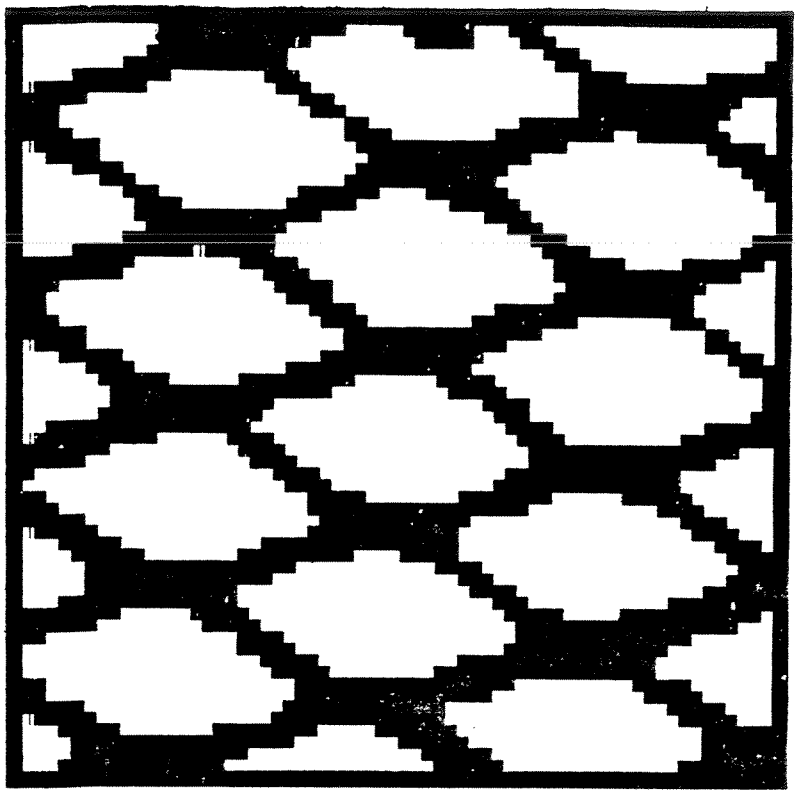


(b)

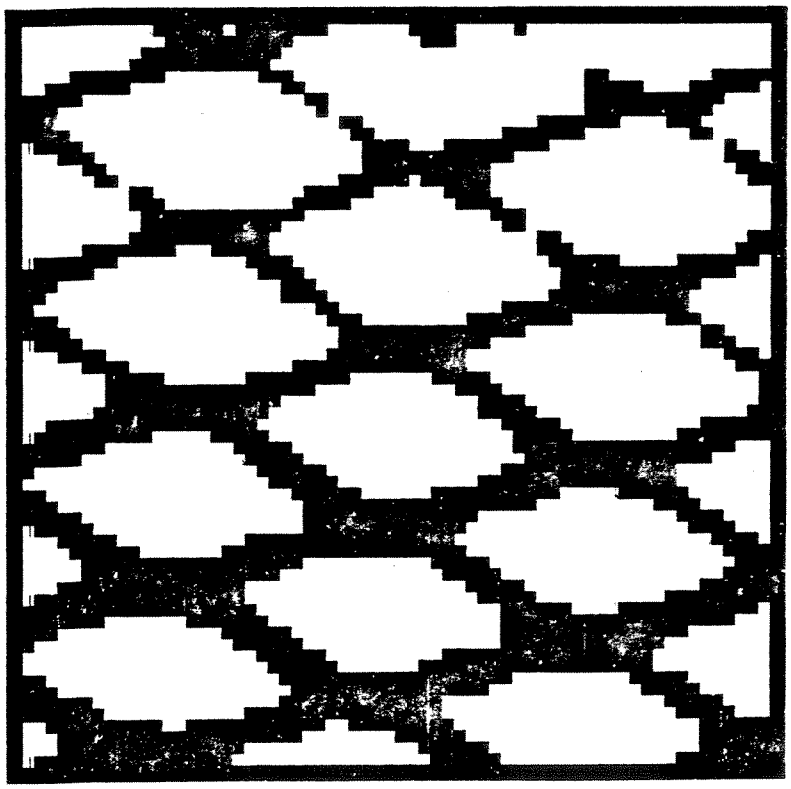


(c)

Figure 4 (continued).

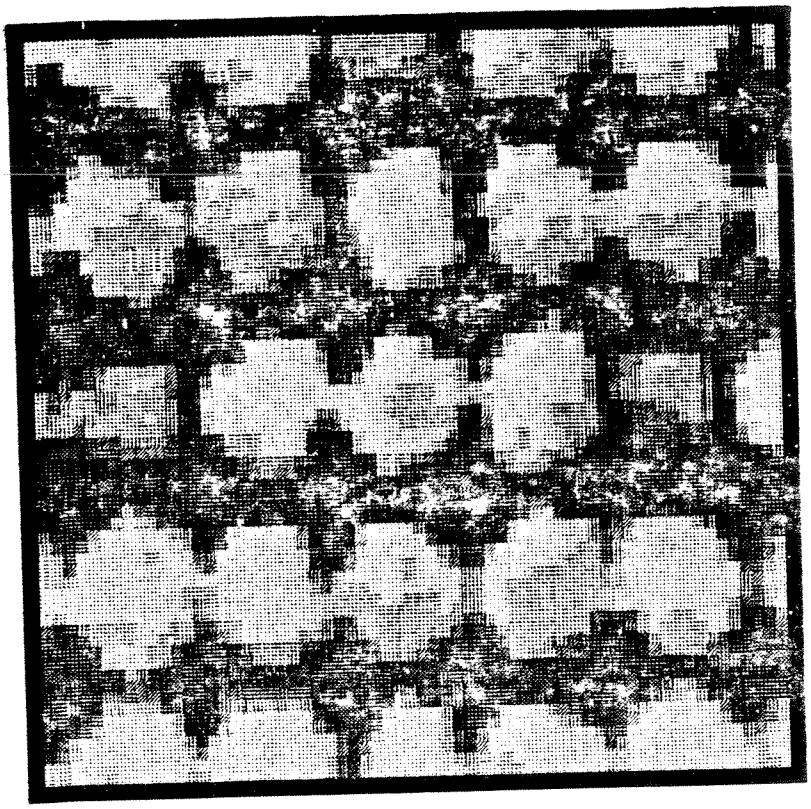


(d)



(e)

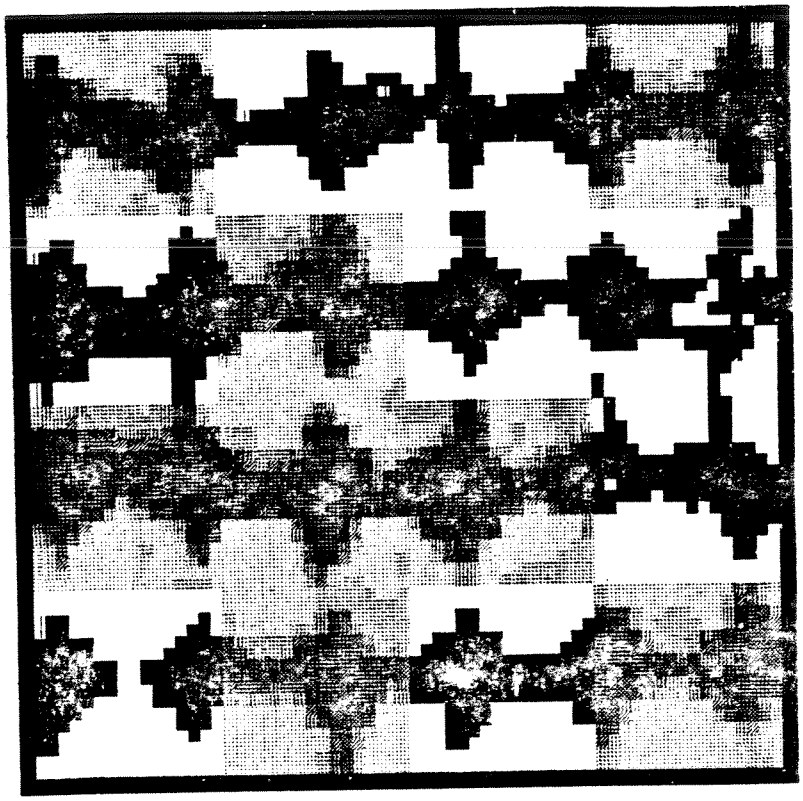
Figure 4 (continued).



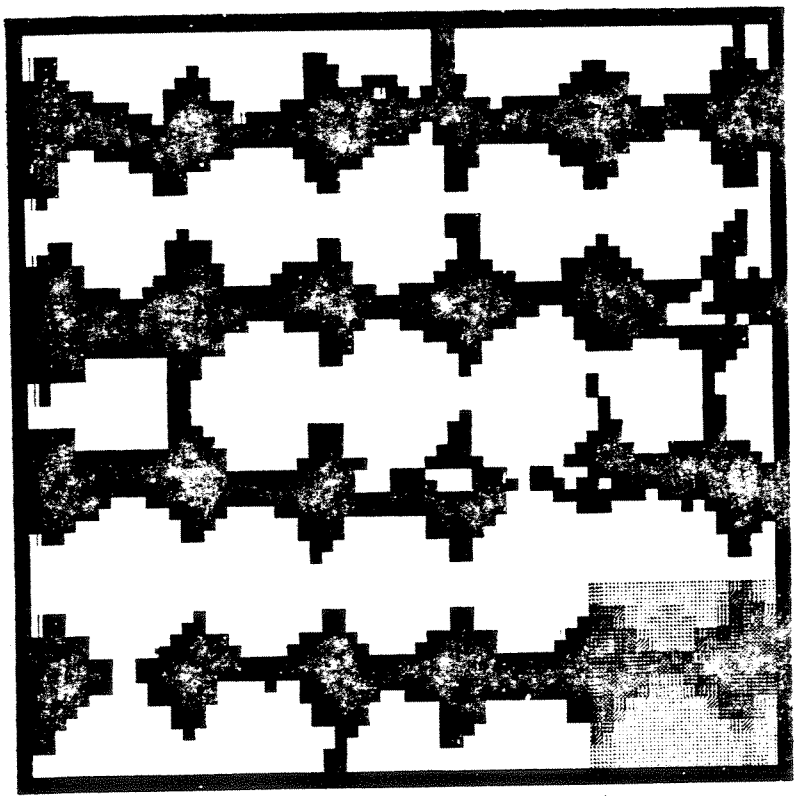
(a)

Figure 5.

(a) Orchards
(b) After steps (a) and (b)
(c) After step (c)
(d) After step (d)
(e) Globally thresholded image at average gray level

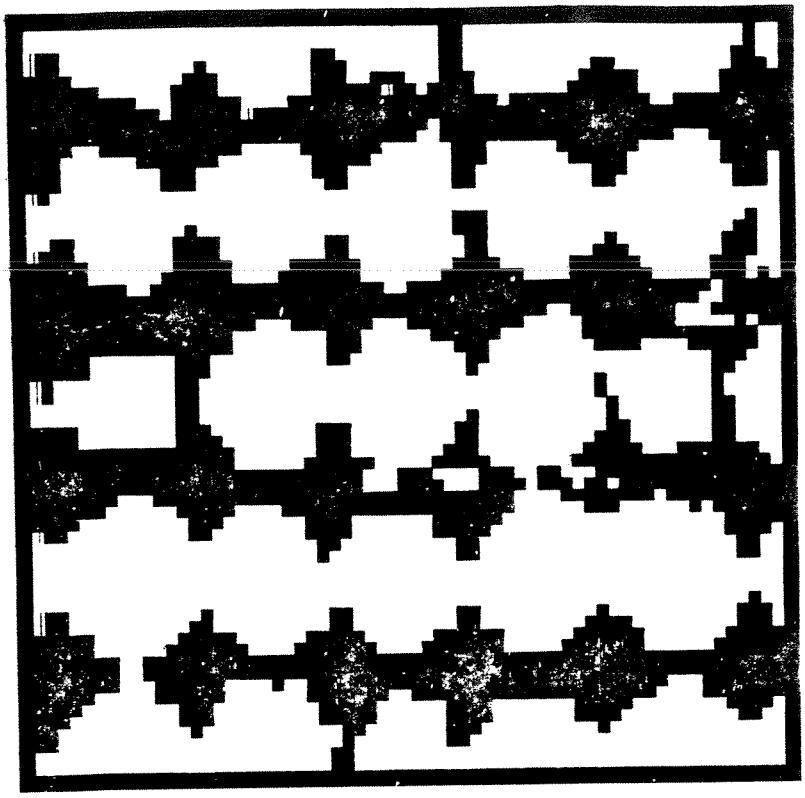


(b)

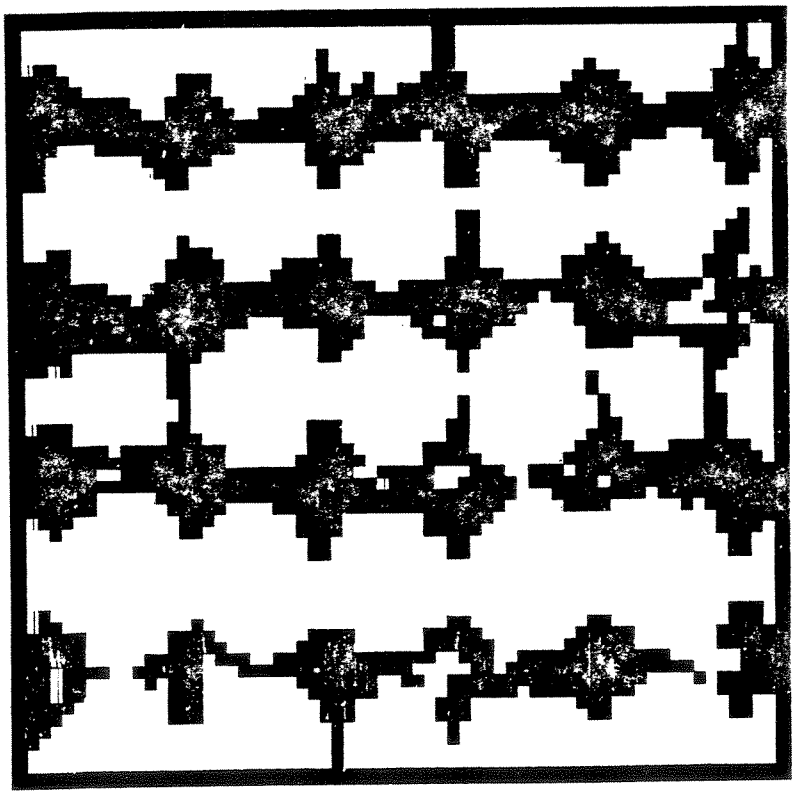


(c)

Figure 5 (continued).

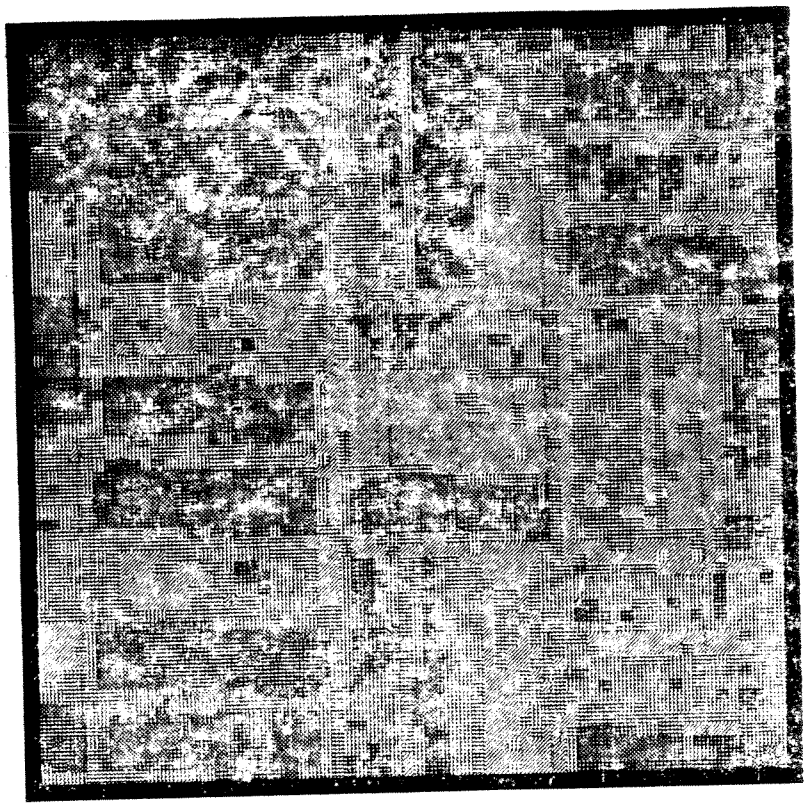


(d)



(e)

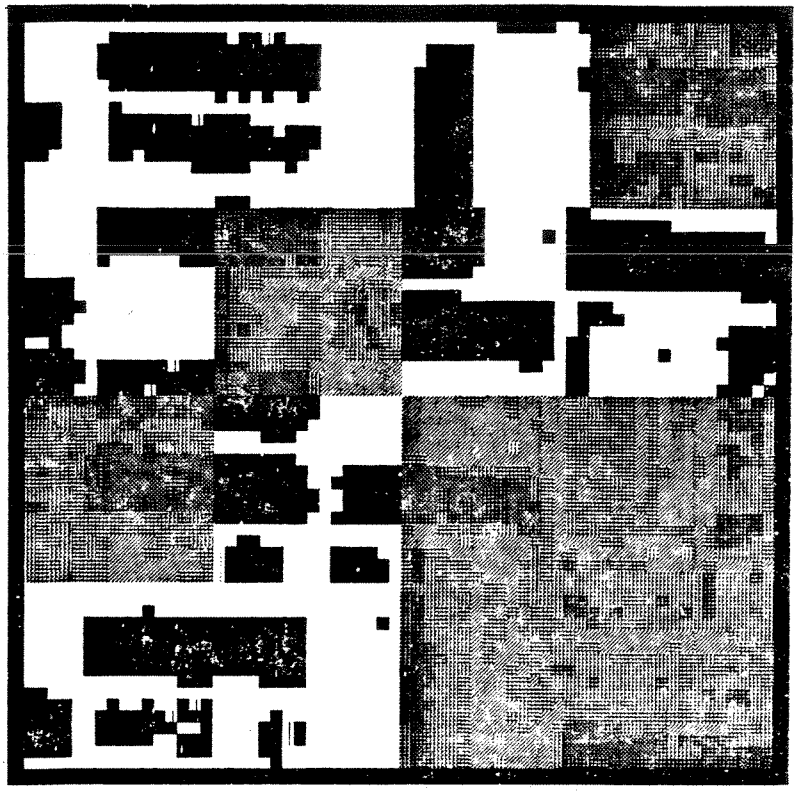
Figure 5 (continued).



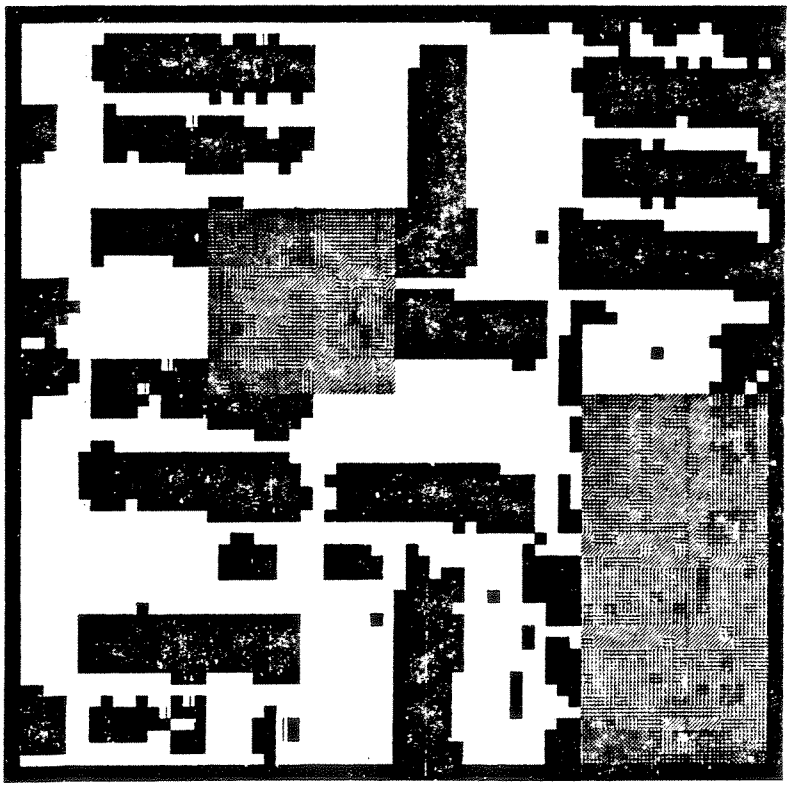
(a)

Figure 6.

(a) Bricks
(b) After steps (a) and (b)
(c) After step (c)
(d) After step (d)
(e) Thresholded image at the average gray level

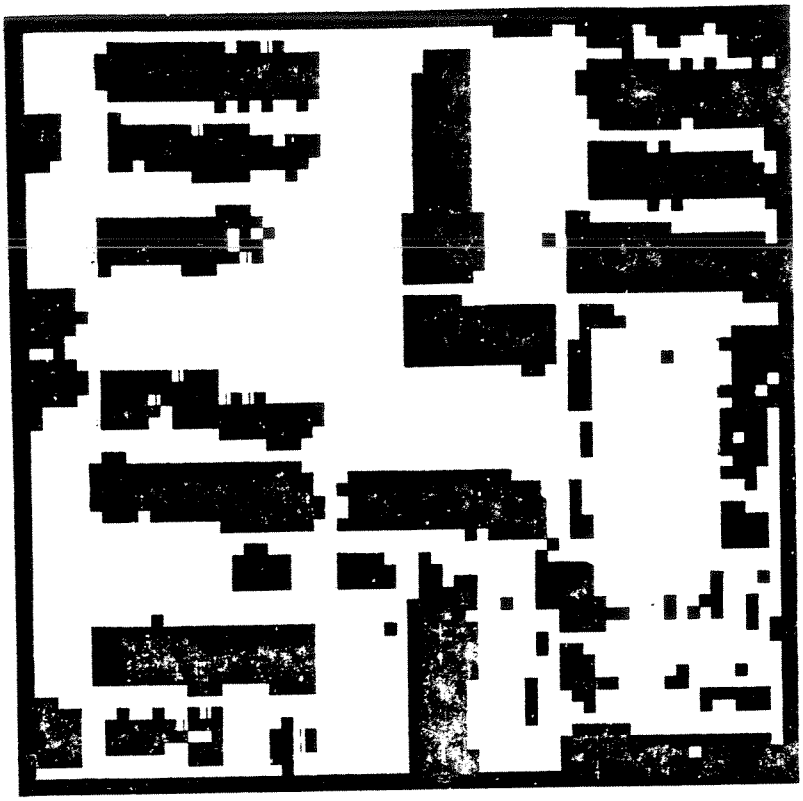


(b)

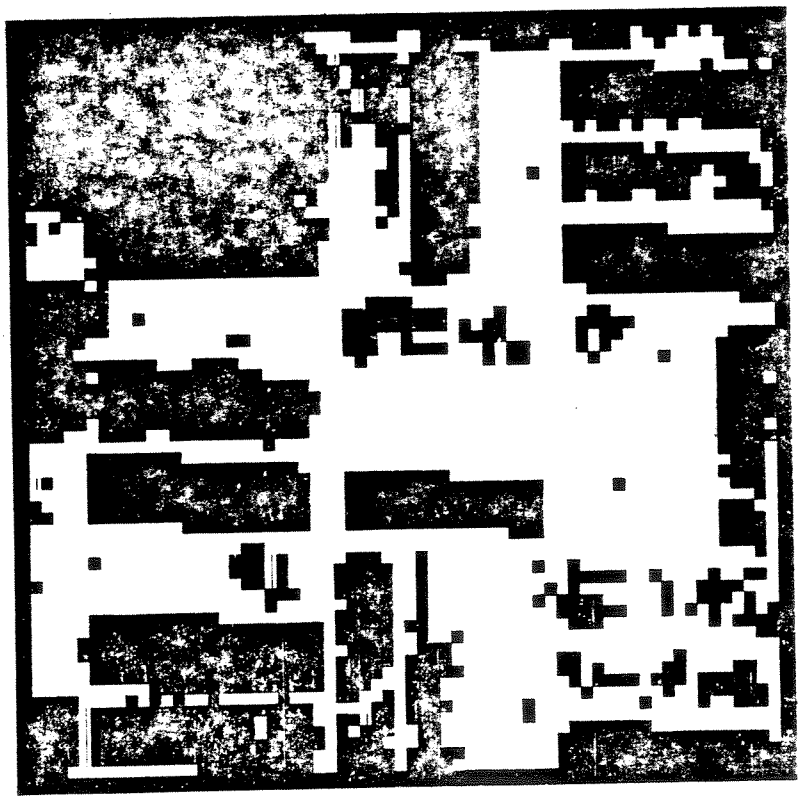


(c)

Figure 6 (continued).

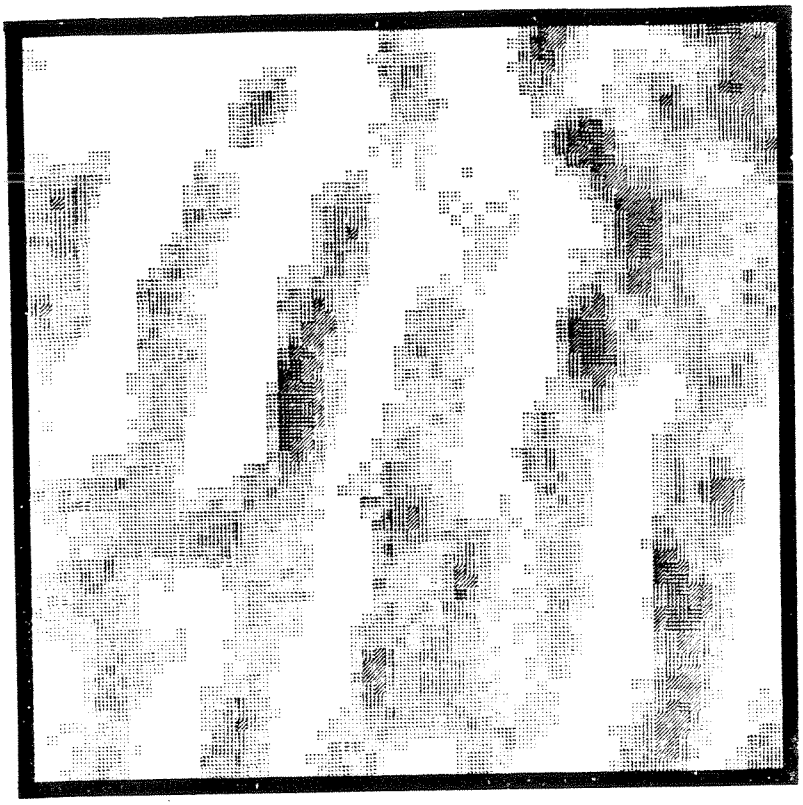


(d)



(e)

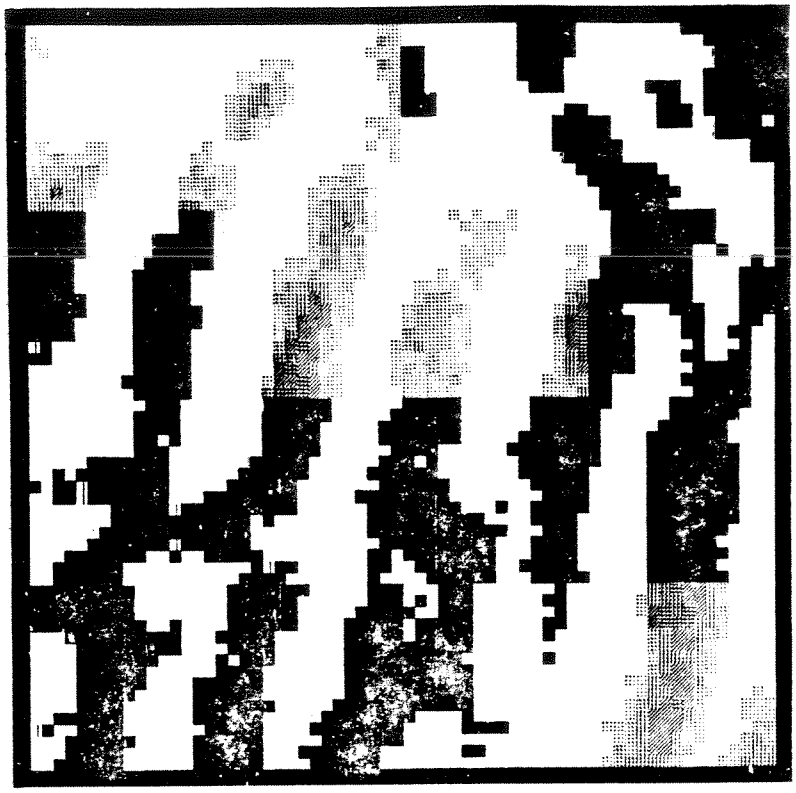
Figure 6 (continued).



(a)

Figure 7.

(a) Concrete
(b) After steps (a) and (b)
(c) After step (c)
(d) After step (d)
(e) Thresholded image at the average gray level



(b)

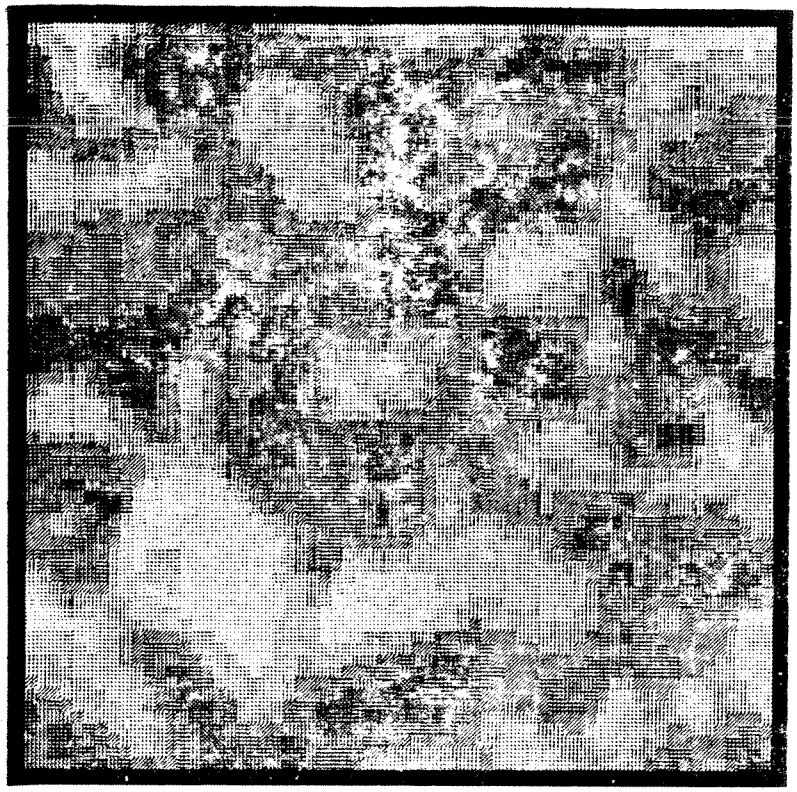
(c)

Figure 7 (continued).

(d)

(e)

Figure 7 (continued).



(a)

Figure 8.

(a) Pebbles
(b) After steps(a) and (b)
(c) After step (c)
(d) After step (d)
(e) Thresholded image at the average gray level

(b)

(c)

Figure 8 (continued).

(d)

(e)

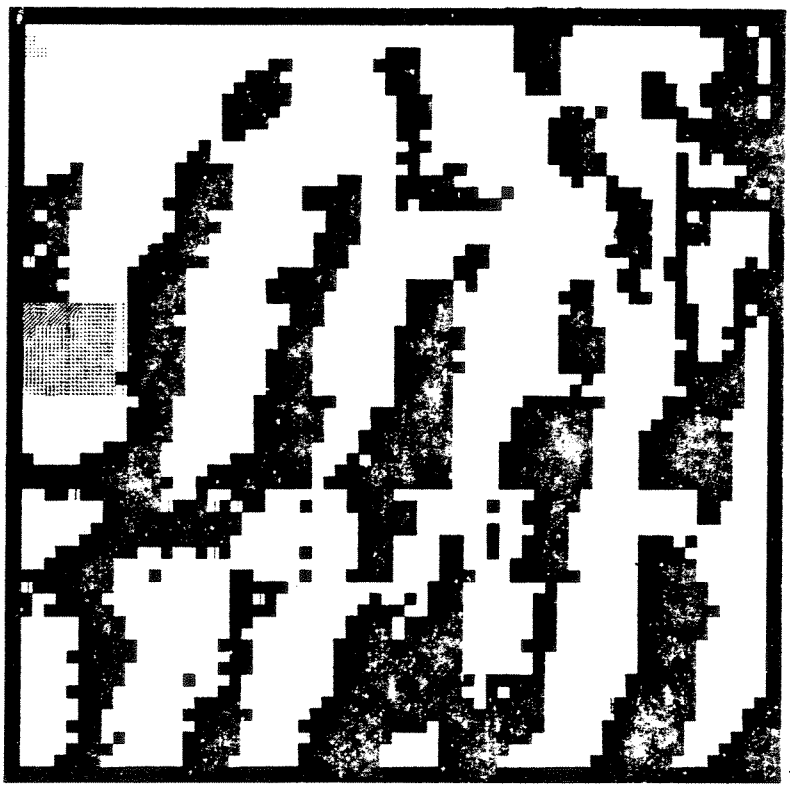
Figure 8 (continued).

(a)

Figure 9.

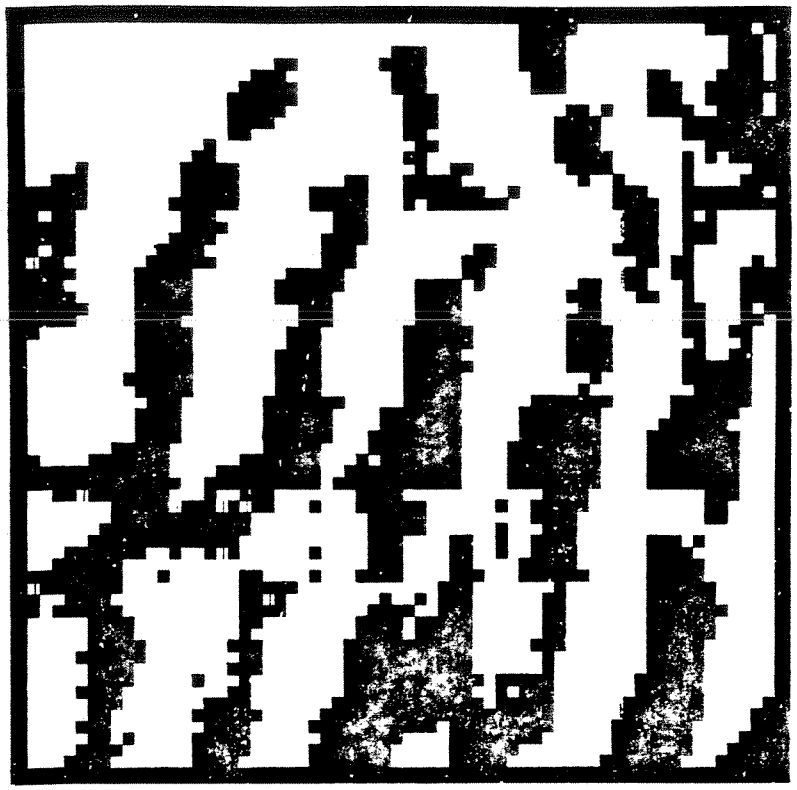
(a) Concrete (modified)
(b) After steps (a) and (b)
(c) After step (c)
(d) After step (d)
(e) Thresholded image at the average gray level

(b)



(c)

Figure 9 (continued).



(d)

(e)

Figure 9 (continued).

22	25	34	27
17	32	22	30
25	32	32	30
20	22	21	21

Table 1.

APPENDIX A

THE ALGORITHM IN PIDGIN PASCAL

The algorithm described above is summarised in pidgin pascal notation below.

```
Begin Algorithm
  Comment:  Initial Threshold Computation.
  For each window do
  If enough edges in window then
  Begin
  Mark window
  Select a set of m thresholds
    For each threshold do
    Begin
    Threshold window.
    Measure quality of resulting segmentation.
    End
  Select best threshold for this window
  Comment:  Threshold Modifiation.
  For each marked window do
  Begin
  Try thresholds (if any) of its 4-neighbors
  Modify threshold if necessary
  End
```

Comment: Window sliding.

For each non-marked window do

Begin

Let (iw, jw) be the coordinates of the window and let nw be the size of the window.

If thresholds can be computed for windows with coordinates (iw-nw/2,jw), (iw+nw/2, jw), (iw,jw-nw/2) and (iw,jw+nw/2)

Then assign best of these thresholds to the window; mark window

End

Comment: Propagate Thresholds.

While there is a non-marked window do

Assign to the non-marked window the best of its neighbors thresholds (if any) and mark window;

end algorithn.