To the Graduate Council:

I am submitting herewith a thesis written by Fernando Contreras entitled "A Dynamic Threshold Algorithm Using Connectivity Analysis." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

R. C. Gonzalez

R. C. Gonzalez, Major Professor

We have read this thesis
and recommend its acceptance:

M. J. Roberts

Carl G. Wagner

Accepted for the Council:

Carl Minkel

The Graduate School

A DYNAMIC THRESHOLD ALGORITHM USING CONNECTIVITY ANALYSIS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Fernando Contreras

March 1985

DEDICATION

With love, to my wife and children, for their patience, understanding, and continuous encouragement.

ACKNOWLEDGMENTS

The author wishes to acknowledge his thesis advisor, Dr. Ralph C. Gonzalez, for his advice and guidance; this is also extended to committee members, Dr. Carl G. Wagner and Dr. Michael J. Roberts, for review of this thesis; and to the authorities of the Universidad de Santiago de Chile, who gave him the opportunity and support for his Master's studies.

ABSTRACT

In order to recognize an object which is embedded in a complex scene, it first must be located and its contour well defined. Such a preprocess, one of Digital Image Processing techniques, is called segmentation.

In this thesis, some methods to perform binary segmentation, by dynamic thresholding the digitized images, were developed. Results from these methods are presented; two of them were used in a character extraction application, based on contour analysis. The darkness relation between object and background plays an important role in the segmentation process. There is no general algorithm which will work for all images; so, every known characteristic from a given set of pictures should be included in a particular implementation of the appropriate method. Hints in selecting a method are also given.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

In order to identify or recognize objects from a digitized image, the objects are typically extracted from the background. The process of isolating objects in an image is known as segmentation, a process which results in a partitioned image with regions that correspond to objects or other areas of interest. One of the most difficult problems in object recognition is the conversion of nonideal analog images into ideal binary images composed by regions to be analyzed based on their shapes, sizes, relative positions, and other characteristics [3]. There are two basic approaches to image segmentation: by outlining contour, or by specifying all pixels inside a region. In the first approach, discontinuities between some pixel features of neighbor regions are used; in the second approach, regions are built based on the similarities of some features of the picture elements [4,6,9,10]. Then, segmentation can be done via contours or via regions.

Since a character is essentially a distribution of black on white, or of one color in another, for character extraction, it is only desired to differentiate between characters and background; hence, it is a two-class classification or binarization case [13]. The most convenient technique then is thresholding, which consists in classifying a pixel according to whether it lies on the blacker

or whiter side of a threshold that is called the binarizing threshold. Gray level thresholding is a widely and successfully used technique of image segmentation [8]. The thresholded image also provides a compressed image representation with respect to the original storage required. Image segmentation is a critical component of an image recognition system because errors in this step will propagate to feature extraction and classification [4]. There is no complete theory of image segmentation [7].

Thresholding can be formulated as a test function of the form

$$T[x,y,g(x,y),N(x,y)] \tag{1}$$

which maps the grayscale into the set {0,1} or {object,background}; where x and y are the pixel's spatial coordinates, g(x,y) is the gray-level of pixel (x,y) and N(x,y) is a local property of pixel (x,y) [14].

Depending on the function T, thresholding can be called global, local, or dynamic. Thresholding is global if it depends only on f(x,y), local if it depends on g(x,y) and N(x,y), or dynamic if, in addition, it depends on (x,y).

Global thresholds are selected based on gray level histograms. For some images, where a clear distinction between foreground-background exists, such as backlighted objects, a single fixed threshold will be able to correctly classify most of the object pixels. These types of images have a gray-level histogram strongly bimodal, where one peak corresponds to objects, and the other to background. Then, the threshold should be chosen between them, at the histogram valley.

Such a method is called mode method or standard-feature histogram method [9,11].

If the histogram is not markedly bimodal, it can be modified by weighting, selecting, or averaging pixels using a local property, such as the gradient information, as decision variable. A typical approach is based on weighting pixels toward high or low gradient pixels, or by building a histogram of a digital gradient picture rather than the original image histogram, and then applying the mode method [9].

If the approximate area of the object is known, the p-tile method can be used. It states that a threshold can be chosen at the gray level which maps at least the percentage representing such area at the tail of the picture histogram if objects are brighter than the background; otherwise, at the beginning. This method is rarely applicable because the a priori knowledge required about the picture and the object is seldom known, or it varies from one sample image to another.

A local threshold sensitive to contrast for example, was computed by Bartz by averaging contrast over previously scanned objects, that is,

$$T = k\,v + c$$

where v is the contrast average, and k and c are optimizing parameters heuristically determined [1]. The average contrast is the local property used in this method. Four similar thresholds were combined to compute a single final threshold.

Although position (x,y) is implicitly considered by averaging previous contrasts, threshold will not be position dependent, because if the same sequence of gray level values occurs anywhere, the threshold will be the same, so it is neighborhood dependent.

Since position dependence is always implicit when local properties are used, there are no absolute local thresholds. Global, local and dynamic is not a classification; it is merely a denomination.

Dynamic thresholds are local thresholds that also depend on the location of the pixel. In this context, dynamic means variable with respect to some independent variable. A good example of dynamic threshold is the local histogram and threshold interpolation method developed by Chow and Kaneko [2,5,14]. They divided the image in several overlapping regions and applied the mode method to all the regions that showed a bimodal histogram, because these should consist of object and background pixels. For the other regions, thresholds were interpolated from thresholds of their adjacent regions. Then, after every region was assigned a threshold, a second interpolation in a pointwise sense was done to get a smoothly varying thresold for the whole image, which was used in the final thresholding operation.

In the work leading to this thesis, six dynamic thresholding schemes based on gray level features for monochrome pictures, and a general recursive threshold updating algorithm were developed. They were implemented and tested in a character extraction application in complex scenes.

CHAPTER II

DYNAMIC THRESHOLDING

Although global thresholding works well for images whose histograms are strongly bimodal, most scenes of practical interest require the use of dynamic thresholding for segmentation since their histograms are commonly multimodal and span almost the entire gray-scale. The basic idea behind dynamic thresholding is to follow gray-level variations by computing the threshold in a recursive way, i.e., by applying a correction factor to the last value of the threshold to obtain a new threshold:

$$T(n+1) = T(n) + c \tag{1}$$

where c is the correction factor and n is a position index. If a row scan is bring used, Equation (1) becomes:

$$T(x,y) = T(x,y-1) + c \tag{2}$$

Conversely, for a column scan it follows that:

$$T(x,y) = T(x-1,y) + c \tag{3}$$

Here and in the subsequent equations, x and y take values from one to the image size. Most of the following discussion deals with a row-scan format, but the results are equally applicable to a column scan.

Two of the most important factors in updating a dynamic threshold are: (1) the gray level of the pixel being thresholded, and (2) the history of gray level variations which have contributed to the value of the old threshold. Thus, the correction factor, c, should depend on these two factors:

$$c = F\,[g(x,y)\ ,\ T(x,y-1)] \tag{4}$$

The most adequate control quantity to make the threshold follow the gray level variations is their difference, which in order to get closer, must be made smaller than the previous one; thus, the correction function can be written as:

$$c = F\,[g(x,y) - T(x,y-1)] \tag{5}$$

The case of F being the identity function is excluded, since then the threshold will always equate the pixel's gray level g(x,y), as shown below.

$$\begin{aligned} T(x,y) &= T(x,y-1) + F\,[g(x,y) - T(x,y-1)] \\ &= T(x,y-1) + g(x,y) - T(x,y-1) \\ &= g(x,y) \end{aligned}$$

Such a value would not be helpful because all pixels would be classified the same giving a meaningless result; therefore, to avoid this, the threshold should not reach the gray-level values while following them.

When the gray level g(x,y) is greater than the current threshold T(x,y-1), that pixel should be classified as white. The threshold should be incremented to approach to g(x,y) but, in agreement with that classification; that is, preserving the original relation g(x,y) > T(x,y). The condition for the updated threshold is then:

$$T(x,y-1) \leq T(x,y) < g(x,y)$$

The equality is applied only when T(x,y-1) is so close to g(x,y) that the smallest correction would make the threshold reach the gray level that is forbidden, as explained before. The correction factor c is, by definition:

$$c = T(x,y) - T(x,y-1)$$

so,

$$c < g(x,y) - T(x,y-1)$$

For g(x,y) less than the current threshold, the condition is then:

$$T(x,y-1) \geq T(x,y) > g(x,y)$$

so, in this case the correction factor is

$$c > g(x,y) - T(x,y-1)$$

Hence, the correction function value must always lie between zero (no correction) and g(x,y) - T(x,y-1) (maximum theoretical correction). The minimum step magnitude that should be detected as

an edge gives the minimum distance at which the threshold should follow the gray-level function.

A threshold updating sequence is shown in Figure 1. Assume Figure 1 (a) is an ascending gray level ramp with a noisy pixel at $y = 3$; which, if thresholded with the fixed threshold I, the pixels at $y = 1$ and at $y = 3$ will be classified as black. Since index x is constant for a picture line, in the following explanation it has been suppressed for simplicity. Using the dynamic threshold updating algorithm, and starting with the same initial threshold $T(0) = I$, the pixel at $y = 1$ will be classified as black. Then, when making $T(1)$ closer to $g(1)$ than $T(0)$, that pixel must maintain its class; so, it is necessary that the condition

$$T(0) \geq T(1) > g(1)$$

be satisfied, as shown in Figure 1 (b). Then, as $g(2)$ is greater than $T(1)$, it will be classified as white if

$$T(1) \leq T(2) < g(2)$$

For the next pixels, the gray-levels are also greater than the previous thresholds; hence, the similar relation should be observed to correctly classify them. In Figures 1 (c) and 1 (d), the dynamic threshold and the thresholded line are shown, respectively.

During the scanning process, when leaving a very dark object or a very light one, the threshold should react promptly in the proper direction, i.e., whenever a high contrast occurs, a large threshold correction towards the gray-level value is needed; so, the function F

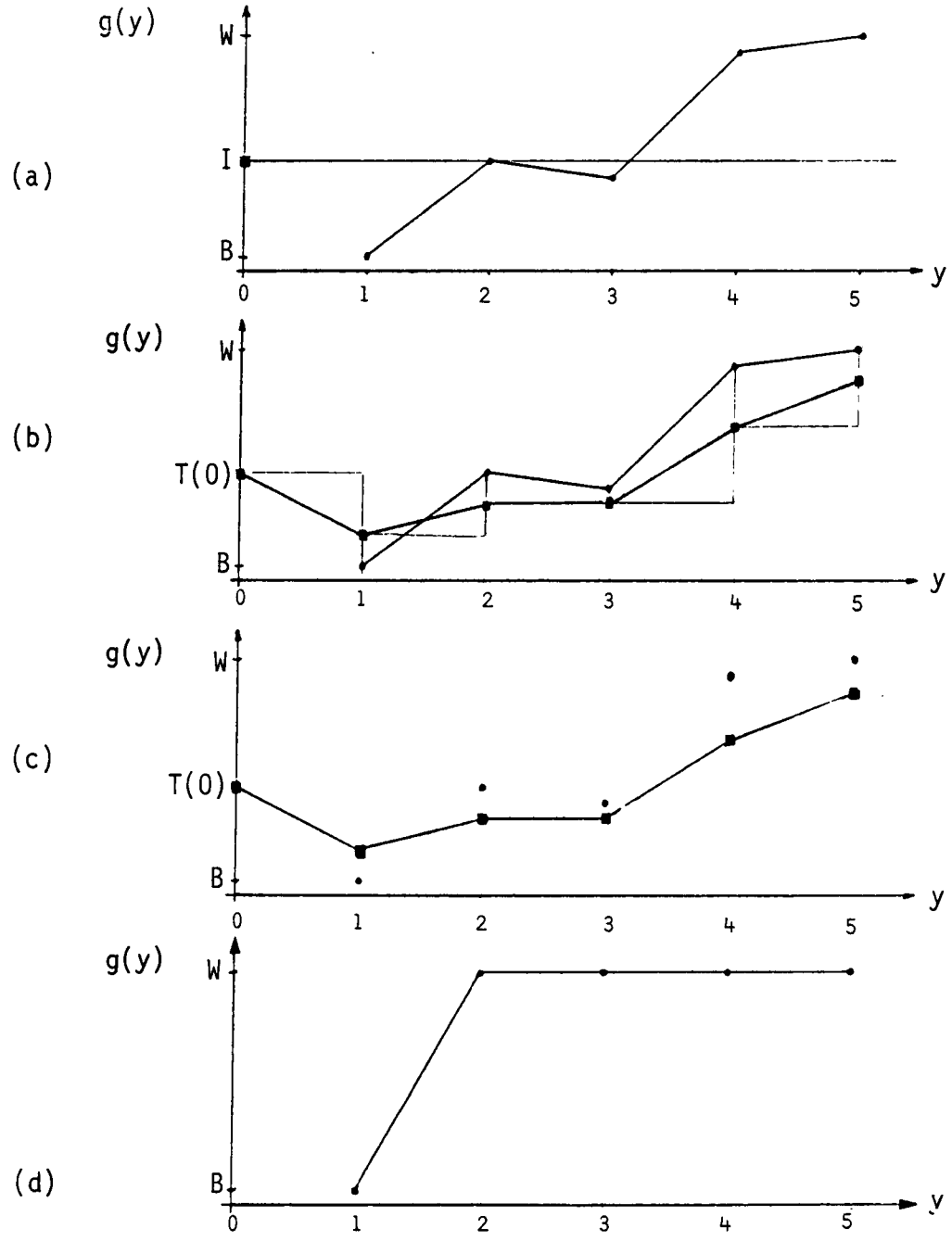


Figure 1. Thresholding Updating Sequence.

(a) Sample gray-level line; (b) Threshold updating; (c) Dynamic threshold; (d) Threshold pixels.

should have odd symmetry. On the other hand, a very large correction is dangerous, because a reaction due to isolated noise pixels can erroneously bias the threshold value. Thus, threshold stability for a uniform segment is expected beyond the first pixel, but no farther than the width of the assumed most narrow object; otherwise, it would be missed.

Threshold approximation to gray level will be done depending on the type of correction function selected. For a linear function,

$$c = k\,[g(x,y) - T(x,y-1)] \tag{6}$$

where k is a positive fraction, the updated threshold will be:

$$T(x,y) = T(x,y-1) + k\,[g(x,y) - T(x,y-1)]$$

and the next gray-level threshold difference will be

$$g(x,y+1) - T(x,y) = g(x,y+1) - T(x,y-1) - k[g(x,y) - T(x,y-1]$$

but, for a uniform segment

$$g(x,y+1) = g(x,y)$$

then, the threshold approximation rate to a uniform gray level segment will be

$$\frac{g(x,y+1) - T(x,y)}{g(x,y) - T(x,y-1)} = 1 - k \tag{7}$$

Thus, the gray level threshold difference will decrease linearly as a function of fraction k. The approximation rate

(Equation 7) should be: (1) positive, which means the threshold is getting closer to gray level without overpassing it, (2) smaller than one to guarantee that the threshold is effectively approaching to gray level value, and (3) nonzero to avoid the threshold reaching the gray level; hence, to fulfill the above three conditions, the quantity k must be restricted to the open interval (0,1).

By using a linear function, approximation to gray levels will be done at constant rate whatever the difference is, but what is needed is a low rate for small variations, and a high correction ratio for large differences as in a high contrast edge. So, a quadratic or higher order function is recommended. Such a function will give a fast adaptive threshold response in cases where a large correction is to be applied.

For a quadratic updating function

$$c = k*[g(x,y)-T(x,y-1)] * sgn[g(x,y)-T(x,y-1)] \qquad (8)$$

where k is a positive scaling fraction that insures to stay within the grayscale range, and sgn(.) is the sign function; then, the threshold will be

$$T(x,y) = T(x,y-1) + k*[g(x,y)-T(x,y-1)]**2 * sgn[g(x,y)-T(x,y-1)]$$

and the next difference

$$g(x,y+1)-T(x,y) = g(x,y+1)-T(x,y-1) - k[g(x,y)-T(x,y-1)]**2 * sgn[g(x,y)-T(x,y-1)]$$

and, as before, if

$$g(x,y+1) = g(x,y)$$

then:

$$\frac{g(x,y+1)-T(x,y)}{g(x,y)-T(x,y-1)} = 1-k[g(x,y)-T(x,y-1)]*sgn[g(x,y)-T(x,y-1)]$$

hence, the approximation rate will be:

$$\frac{g(x,y+1)-T(x,y)}{g(x,y)-T(x,y-1)} = 1-k\ |g(x,y)-T(x,y-1)| \qquad (9)$$

So, in this case, the difference magnitude will decrease by the unit complement of the value $k\ |g(x,y)-T(x,y-1)|$, provided that such value does not reach the unit, as in linear case; that is

$$k\ |g(x,y)-T(x,y-1)| < 1$$

or,

$$k < 1/\ |g(x,y)-T(x,y-1)|$$

The limits for constant k are then:

$$0 < k < 1/ \text{ (maximum difference magnitude)}$$

In general, for an n-power updating function as

$$c = k*[g(x,y)-T(x,y-1)] ** n \qquad (10)$$

the approximation rate will be:

$$\frac{g(x,y+1)-T(x,y)}{g(x,y)-T(x,y-1)} = 1-k\ |g(s,y)-T(x,y-1)|^{**}(n-1) \qquad (11)$$

where k is restricted to the open interval

[0 , 1/(maximum difference magnitude)**(n-1)]

The procedures discussed thus far will make the threshold follow only the horizontal gray level variations in a picture row; however, due to the bidimensional nature of pictures, it is important to obtain vertical thresholds as well. The vertical threshold behavior should be similar to that of the horizontal threshold in order to detect horizontal changes in a column scan process; therefore, all discussion related to horizontal threshold is also valid for a vertical dynamic threshold computation. These two thresholds can then be combined to get a final dynamic threshold. Some alternative methods are:

1. Average thresholds

$$Td(x,y) = .5\ [Th(x,y) + Tv(x,y)] \qquad (12)$$

where Th is the horizontal threshold computed as before, Tv is the vertical threshold computed in a similar way to Th, and Td is the dynamic threshold.

2. Use maximum threshold value

$$Td(x,y) = \max\ [Th(x,y)\ ,\ Tv(x,y)] \qquad (13)$$

3. Use minimum threshold value

$$Td(x,y) = \min\ [Th(x,y)\ ,\ Tv(x,y)] \qquad (14)$$

4. Use horizontal threshold instead of gray level in vertical threshold computation

$$Td(x,y) = Td(x,y-1) + G\,[Th(x,y)\ ,\ Td(x-1,y)] \qquad (15)$$

where G(.) is a function as F discussed above [15]; or use both thresholds separately and then:

5. Logically AND the binary results

$$b(x,y) = bh(x,y)\ \text{AND}\ bv(x,y) \qquad (16)$$

where b(x,y) is a pixel of final binary image, bh(x,y) is the corresponding binary pixel horizontally thresholded, and bv(x,y) is the corresponding binary pixel vertically thresholded, or

6. logically OR the binary results

$$b(x,y) = bh(x,y)\ \text{OR}\ bv(x,y) \qquad (17)$$

Results given by these six methods will be shown and analyzed in the next chapter.

Horizontal and vertical thresholds must be initialized to some value; a thresholding algorithm should have low sensitivity to this value, which implies that it must have a fast response to high contrast variations.

The performance of a dynamic thresholding algorithm can be further enhanced by using connectivity analysis (i.e., region tracking). The use of region tracking for threshold modification is based on the assumption that neighbors of pixels that belong to an object have high probability of being part of the same object since these are always connected [12]. Those eight pixels encircled around a

given pixel P within a radius smaller than two digitizing-grid units are called eight-neighbors of P; that is

$$NB(P) = \{ G \,/\, D_e(P,Q) < 2 \}$$

where NB(P) is the set of eight-neighbors of P, and $D_e(P,Q)$ is the Euclidean distance between pixels P and Q. P and Q are said to be eight-connected if Q is an eight-neighbor of P.

To avoid missing part of the object because of uneven lighting or lower contrast, the acceptance criteria (threshold) is made less stringent for pixels that are eight-neighbors of an already accepted one. This is done by using a multiplicative factor to modify the dynamic threshold value before classifying the pixel as object or background; that is,

$$Tm(x,y) = Td(x,y) * q \qquad (18)$$

where q is a factor to increment or decrement Td(x,y) by some percent, and Tm(x,y) is the modified dynamic threshold.

Such a factor can be used as a parameter to control the segmentation; for example, using a value of one makes the region tracking procedure inoperative, because the dynamic threshold value itself is used in the classification process. Values greater than one will give more populated images but, with high values, objects will become agglomerated as small gaps between them will be filled; on the other hand, a factor less than one will give more clean images. For still lower factor values, lighter objects or part of them will be missed.

Since the threshold modification (Equation 18) is biased during the programming step, the use of connectivity requires knowledge about the objects-background darkness relationship. If this knowledge is not available at that time or it is intended to extract objects both lighter or darker than the background, it should be better to use a non-connectivity based thresholding method.

The following algorithms, presented in top-down form, summarize the procedures already discussed. As used here, run is a constant gray-level sequence of adjacent pixels in a row.

General Algorithm

Step 1. Initialize horizontal and vertical thresholds.

Step 2. Apply dynamic thresholding to first line.

Step 3. Compute runs in first line.

Step 4. Track objects found in previous line.

Step 5. Compute runs in next line.

Step 6. If this is not last line, go to Step 4.

Step 7. Output segmented image file.

Step 8. Stop.

Dynamic Threshold Algorithm

Step 1. Take first pixel.

Step 2. Update horizontal threshold (Equation 2).

Step 3. Update this column vertical threshold (Equation 3).

Step 4. Compute dynamic threshold (Equations 12 to 17).

Step 5. Modify dynamic threshold according to region tracking factor q (Equation 18).

Step 6. Dynamically threshold the pixel.

Step 7. Do Steps 2 to 5 for each pixel in the line segment.

Region Tracking Algorithm

Step 1. Take first run in the line.

Step 2. If it is a background run, dynamically threshold the line segment at threshold values. If it is an object run, include the two neighbors in the extremes of the run; modify the dynamic threshold, and dynamic threshold the expanded run.

Step 3. If this is the last run in the line, stop; otherwise, take next run and go to Step 2.

Run-start Coding Algorithm

Step 1. Store first run gray level.

Step 2. Search for next run in the line.

Step 3. If found, store starting column; go to Step 2. If not, store an end flag value; then stop.

CHAPTER III

EXPERIMENTAL RESULTS AND RECOMMENDATIONS

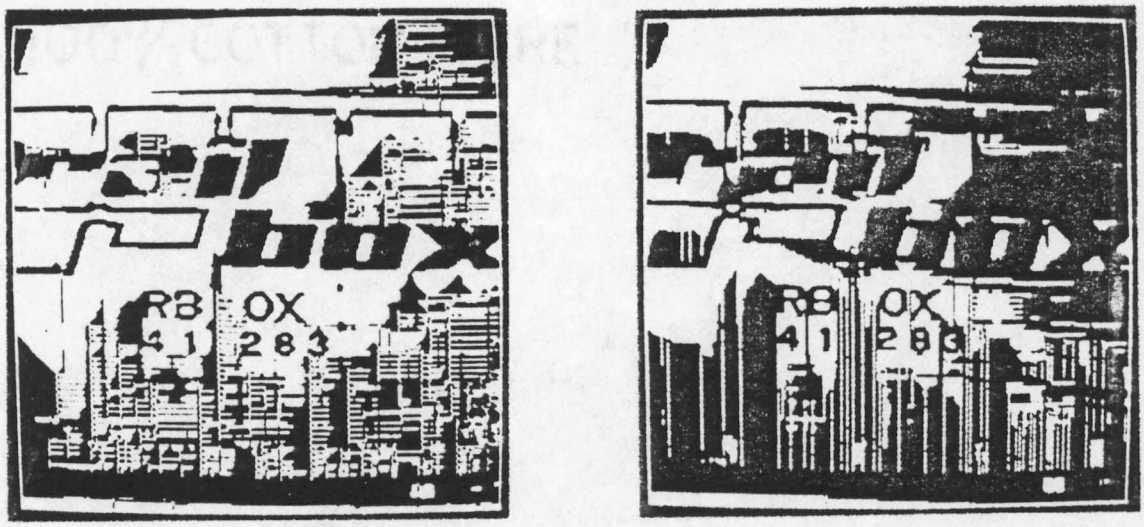
A binarizing algorithm for digitized multilevel images has been developed in the previous chapter, and six alternative methods for specific computations have been suggested. Some results from many tests that have been done so far are to be analyzed here. For each threshold combining method stated in Equations 12 through 17, there are basically three parameters: the initial threshold value, the modifying factor q used in region tracking algorithm (Equation 18), and the correction function for threshold updating (Equation 5). Those different methods will be illustrated by applying them to test pictures shown in Figure 2.

Results using connectivity with modifying factor q = 1.03, the average gray level (145) as initial threshold, and quadratic correction function (Equation 8) are shown in Figure 3. It is observed that inclusion of connectivity analysis tends to create horizontal and vertical strokes in all cases but in the combined threshold method (Equation 15). This is produced by the modification applied over the threshold propagating the error of a misclassified pixel while moving in the scanning directions. As expected, results given by the second method are darker than those from the third method because of its lighter threshold.

(a) (b)

Figure 2. Sample Test Images.

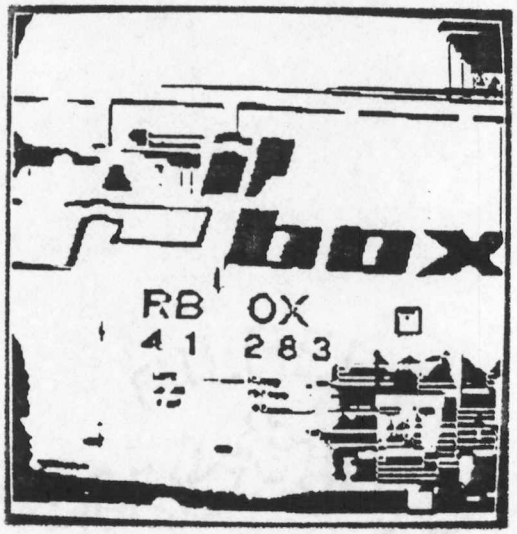
(a) Train car. (b) Commercial box.



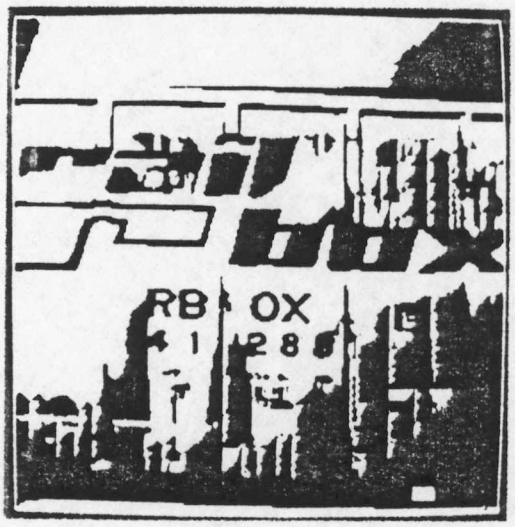
(a) (b)

Figure 3. Train Car Image Dynamically Thresholded with Connectivity Analysis, Quadratic Correction Function, and the Average as Initial Threshold.

(a) Average threshold method. (b) Maximum threshold method. (c) Minimum threshold method. (d) Combined threshold method. (e) AND method. (f) OR method.

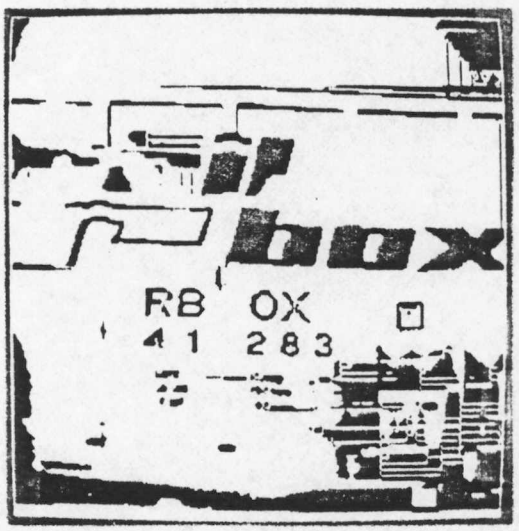


(c)



(d)

(e)



(f)

Figure 3 (continued)

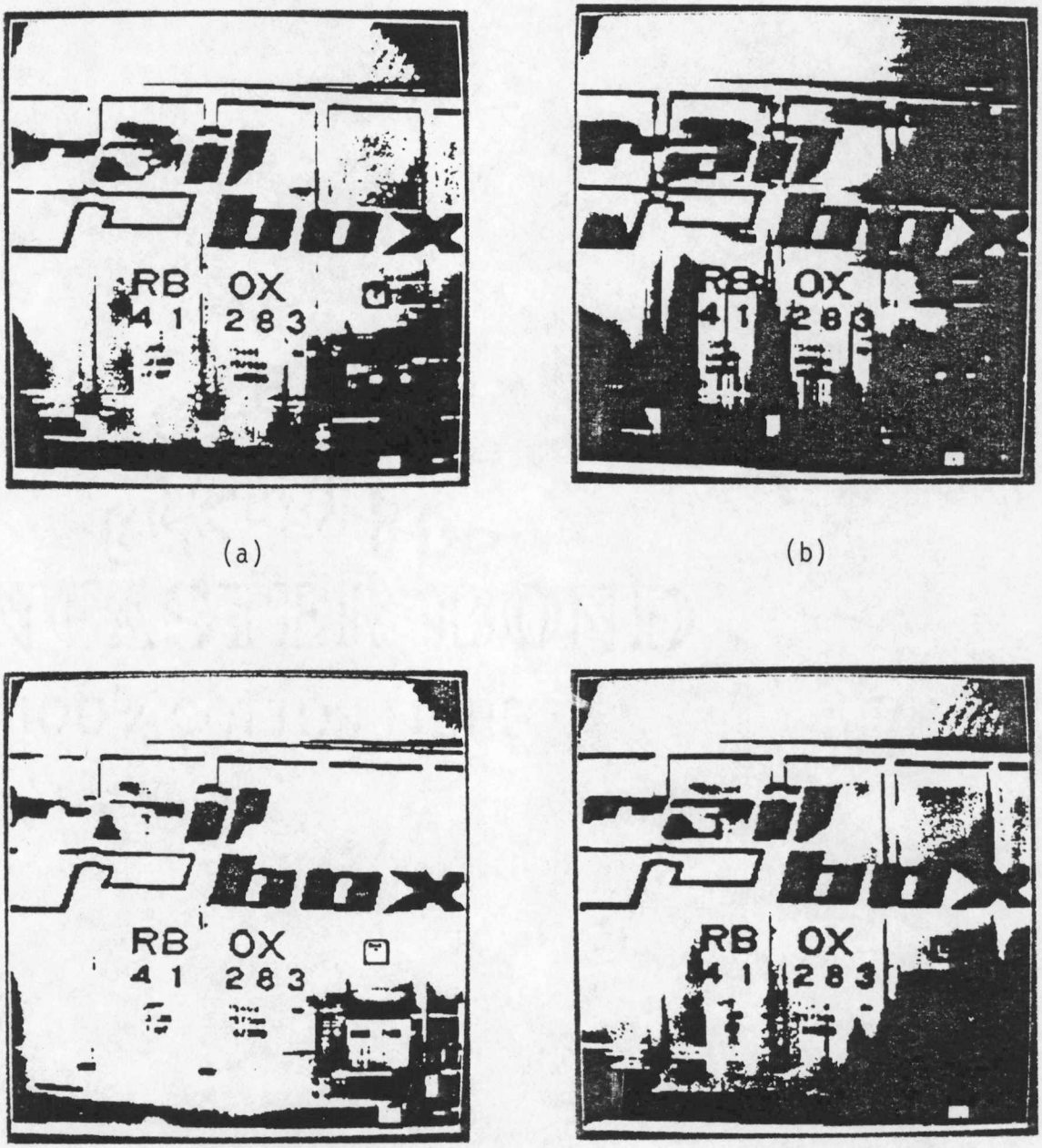
As can be observed in Figure 4, corresponding processes with the same methods and parameters but excluding connectivity are better than those which use it. So, connectivity should be used only with the combined threshold method (Equation 15) which gives better results on the average. Figure 5 shows the best (AND method) and the worst (OR method) results for commercial box image.

Since AND method and OR method can be seen as a combination of maximum and minimum threshold methods, results given by AND method are similar to those from maximum threshold method because a dark classified pixel is dominant for AND operation; and similarly, results from the OR method are very close to those of the minimum threshold method because white classified pixels are dominant for OR operation. So, when light objects are expected, AND method should be preferred, among AND and OR methods; and conversely, OR method for dark objects.

Without connectivity analysis, the best methods for white objects are the second and the fifth, followed very closely by the fourth. For dark objects, the best results are given by the third and the sixth methods, which produce a lighter image preserving the cleanness of objects in a light background.

For a linear correction function, results are better for smaller k values, becoming in some cases the best with $k = 0$, that is, no correction at all or fixed thresholding at the initial threshold. Some examples of test images processed with linear threshold updating functions are presented in Figure 6.

As vertical and horizontal thresholds are carried, the Dynamic Threshold algorithm is essentially a sequential process; so, a high

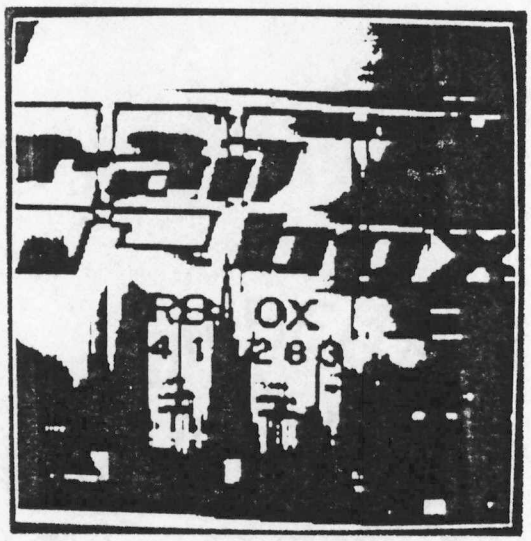


(a) (b)

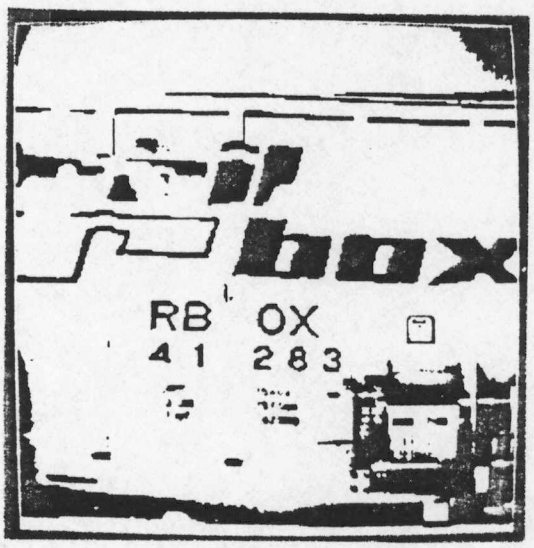
(c) (d)

Figure 4. Train Car Image Dynamically Thresholded Without Connectivity Analysis, Quadratic Correction Function, and the Average as Initial Threshold.

(a) Average threshold method. (b) Maximum threshold method. (c) Minimum threshold method. (d) Combined threshold method. (e) AND method. (f) OR method.

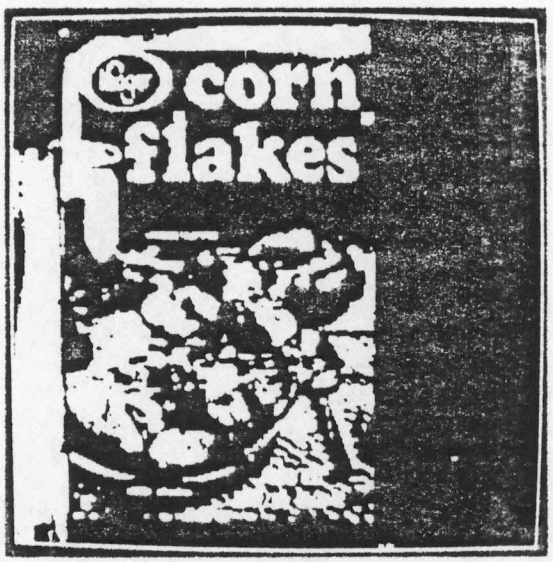


(e)



(f)

Figure 4 (continued)

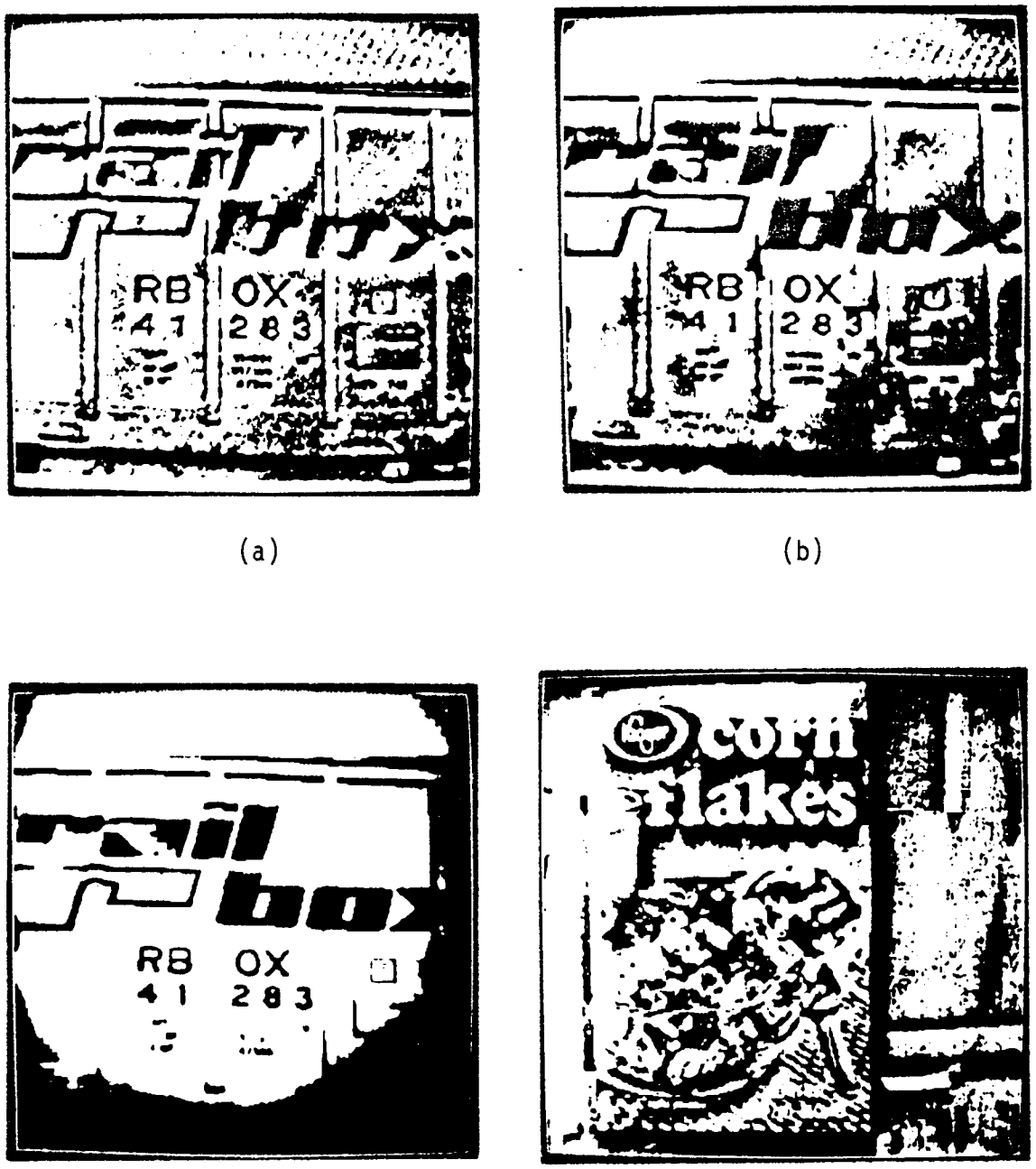


(a)

(b)

Figure 5. Commercial Box Image Thresholded Without Connectivity Analysis, Quadratic Correction Function, and the Average as Initial Threshold.

(a) AND method (best). (b) OR method (worst).



(a) (b)

(c) (d)

Figure 6. Test Images Thresholded Without Connectivity Analysis, Linear Correction Function, and the Average as Initial Threshold.

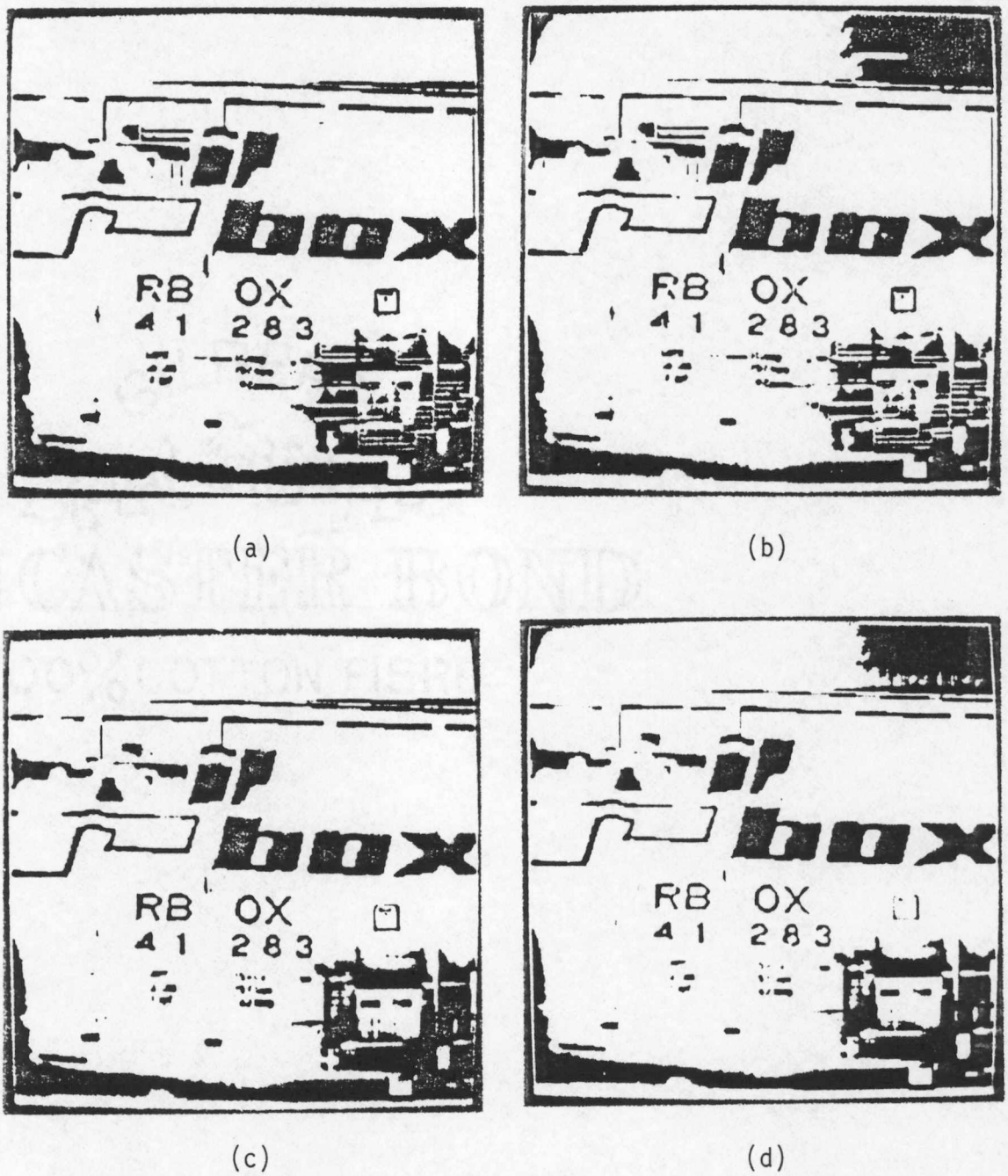
(a) k = 0.5 (b) k = 0.25 (c) k = 0 (d) k = 0.25

speed processor should be used for real time applications; however, program speed can be improved by including precomputed values for correction functions as a look-up table. Functions were included in analytic form as illustrative examples to future users who should then replace them by their own.

If parallel processing is available, processing time will be farther improved by using Equations 16 or 17 to combine the results of two fast independent processes that can be applied simultaneously to all image rows and lines.

Inclusion of region tracking algorithm helps in the recovery of uneven lighted objects; however, it produces a biased response towards one of the scale ends, say black or white. Therefore, if detection of light objects on dark background or dark objects on light background is desired, the use of region tracking is not recommended to get a balanced response.

Sensitivity to initial value can be a problem which must be considered image dependent. If background is lighter than objects, the initial threshold must be chosen at a darker value than in the opposite case in order to start correctly classifying top edge pixels. Anyway, the initial threshold should lie within the image's grayscale range; and heuristically has been determined that the average value is a good starting point for most of the images. All examples shown so far were processed with the average gray level as initial threshold, because it gives acceptable results; and, in addition, it can be easily obtained at TV speed with modern hardware. Figure 7 shows results obtained with initial values 31% above and below the average



(a) (b)

(c) (d)

Figure 7. Train Car Image Dynamically Thresholded by OR Method with Quadratic Correction Function.

(a) and (b) with connectivity analysis, (c) and (d) without connectivity analysis.

(a) Initial threshold = 100 (b) Initial threshold = 190 (c) Initial threshold = 100 (d) Initial threshold = 190.

value. By comparing Figures 7(a) and 7(b) with Figure 3(f) (page 20), and Figures 7(c) and 7(d) with Figure 4(f) (page 23), only a slight change is observed; hence, the algorithm has low sensitivity to initial values around the gray level average.

CHAPTER IV

AN APPLICATION: CHARACTER EXTRACTION

After the picture has been binarized, objects are to be contour analyzed to leave only those that are good candidates to form a character sequence, as in a word or license number, within certain size limits and other restrictions. Such objects are to be given as input data to a pattern recognition system.

Binary image is duplicated to preserve it for later use as required. It is raster scanned searching for an object pixel. Once one is found, counterclockwise border following starts using 8-connectivity criterion. The process goes on until the initial pixel is reached again, which will always happen because all objects are closed boundaries as picture edges are also considered object borders.

Every object in the image is scanned in this way, and then the following rejection criteria are applied:

- Minimum height
- Maximum height
- Maximum width
- Minimum perimeter
- Maximum perimeter
- Incompleteness (bounded by picture edge)

Objects passing these tests are compared with the previously accepted object looking for some sequential relationship. If it

exists, coordinates of the box surrounding it are stored in a file to be used by the recognition program to extract the supposed character sequences.

The system is able to detect sequences of characters of approximately the same height, with given range, that have centroids aligned horizontally, and that are non-connected, as in printed characters, written in one or more lines.

Results given by contour-tracing and sequence-selector program (CTSS) are shown in Figure 8. Train car identification "R B O X 4 1 2 8 3" was obtained from the segmented image in Figure 4(f) (page 23); and the commercial box label "corn flakes" was obtained from the segmented image in Figure 5(a) (page 23).

Contour Analysis Algorithm

Step 1. Make a work copy of binary image.

Step 2. Search for an object pixel.

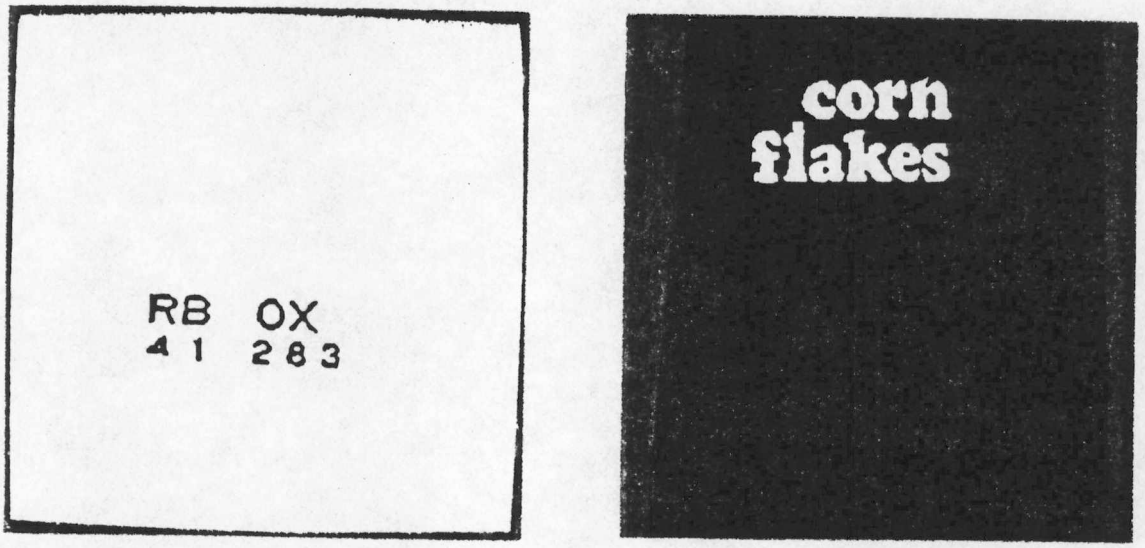
Step 3. If none is found, stop.

Step 4. Follow border of object found.

Step 5. Test object features.

Step 6. If conditions are satisfied, store box coordinates.

Step 7. Delete object from work image; return to Step 2.



(a) (b)

Figure 8. Test Images After Contour Analysis Process.

(a) Train car identification. (b) Commercial box label.

8-Connected Border Follower Algorithm [11]

Let:

IP : Given initial pixel of object S border

Q : Given IP-8-neighbor not in S

N : Next 8-neighbor of IP in counterclockwise sense from Q

Step 1. P ← IP

Step 2. Compute N's location

Step 3. Does N belong to S?

No, Q ← N; continue in Step 4.

Yes, continue in Step 5.

Step 4. Are P neighbors left?

Yes, continue in Step 2.

No, it is an isolated pixel. Stop.

Step 5. P ← N

Step 6. If P is distinct from IP continue in Step 2.

Step 7. Stop.

CHAPTER V

CONCLUSIONS

Dynamic Thresholding Algorithm using connectivity analysis works well with combined threshold updating method (Equation 15). In that case, it can be applied to both types of images: with objects lighter or darker than the background. However, connectivity analysis cannot be adapted during the process; it requires prior knowledge about the object-background darkness relationship.

When it is known that the images to be processed have objects darker than the background, the best results are given by the minimum threshold method (Equation 14) and the OR method (Equation 17); and when objects are lighter than the background, the best results are given by the maximum threshold method (Equation 13) and the AND method (Equation 16).

For all methods, the initial value should be selected around the average gray-level of the image.

LIST OF REFERENCES

LIST OF REFERENCES

1. Bartz, M. R., "The IBM 1975 Optical Page Reader," IBM Journal of Research and Development, Vol. 12, 1968, pp. 354-363.

2. Chow, C. K. and Kaneko, T., "Automatic Boundary Detection of the Left Ventricle from Cineangiograms," Computers and Biomedical Research, Vol. 5, 1972, pp. 388-410.

3. Foith, J. P., Eisenbarth, C., Enderle, E., Geiselmann, H., Ringhauser, H. and Zimmermann, G., "Real-Time Processing of Binary Images for Industrial Applications," in Digital Image Processing Systems, L. Bolc and Z. Kulpa (Eds.), Berlin, Heidelberg, New York: Springer-Verlag, 1981, pp. 61-168.

4. Fu, K. S. and Mui, J. K., "A Survey on Image Segmentation," Pattern Recognition, Vol. 13, 1981, pp. 3-16.

5. Gonzalez, R. C. and Safabakhsh, R., "Computer Vision Techniques for Industrial Applications and Robot Control," Computer, Vol. 15, 1982, pp. 17-32.

6. Gonzalez, R. C. and Wintz, P., Digital Image Processing, Reading, Massachusetts: Addison-Wesley Publishing Company, 1977.

7. Haralick, R. M., "Image Segmentation Survey," in Fundamentals of Computer Vision, O. D. Faugeras (Ed.), Cambridge: Cambridge University Press, 1983, pp. 209-223.

8. Kittler, J., Illingworth, J., Foglein, J. and Paler, K., "An Automatic Thresholding Algorithm and Its Performance," Proceedings of the Seventh International Conference on Pattern Recognition, Montreal, Canada, 1984, Vol. 1, pp. 287-289.

9. Kohler, R., "A Segmentation System Based on Thresholding," Computer Graphics and Image Processing, Vol. 15, 1981, pp. 319-337.

10. Milgram, D. L. and Rosenfeld, A., "Object Detection in Infrared Images," in Digital Image Processing Systems, L. Bolc and Z. Kulpa (Eds.), Berlin, Heidelberg, New York: Springer-Verlag, 1981, pp. 228-353.

11. Rosenfeld, A. and Kak, A. C., Digital Picture Processing Vol. 2 New York: Academic Press, 1982.

12. Rosenfeld, A., "Segmentation: Pixel-based Methods," in Fundamentals of Computer Vision, O. D. Faugeras (Ed.), Cambridge: Cambridge University Press, 1983, pp. 225-237.

13. Ullman, J. R., "Picture Analysis in Character Recognition," in Digital Picture Analysis, A. Rosenfeld (Ed.), Berlin, Heidelberg, New York: Springer-Verlag, 1976, pp. 295-343.

14. Weszka, J. S., "A Survey of Threshold Selection Techniques," Computer Graphics and Image Processing, Vol. 7, 1978, pp. 259-265.

15. White, J. M. and Rohrer, G. D., "Image Thresholding for Optical Character Recognitino and Other Applications Requiring Character Image Extraction," IBM Journal of Research and Development, Vol. 27, 1983, pp. 400-411.

APPENDIXES

APPENDIX A

DTRT PROGRAM IMPLEMENTATION

DTRT PROGRAM IMPLEMENTATION

Dynamic-Threshold Region-Tracking (DTRT) Program accepts black and white pictures of size 256x256 pixels quantized in 256 gray levels, and produces a binary picture of the same size by dynamic thresholding the pixels with regino tracking criterion. It is written in FORTRAN and it is implemented in a VAX 11/780 computer. Interactively, it asks for the image filename, assuming extension PIC if none is given. It creates an output image file with the same name and extension BIN. It uses subroutine THRES to threshold the image, and subroutine TRACK to track objects darker than background. Output image will have a white frame one pixel wide to speed up the contour analysis program, as it will be shown later. Program listing is in Appendix B.

As the region tracking Algorithm requires two thresholds, and the Dynamic Threshold Algorithm provides only one, the more flexible second one will be computed as a percent of the dynamic threshold for that pixel. This less stringent threshold must be shifted toward white to accept slightly lighter pixels as a part of a dark object.

Initial threshold ITHR is set at the beginning of the program; it is used to initialize the vertical threshold array VTHR, and the horizontal threshold HTHR at the beginning of each line.

The first line is dynamically thresholded and its runs computed into RO (Runs in Odd line). Then, every subsequent line is tracked by using runs of the previous line, RO or RE depending on whether it was an odd or an even line.

The first and last line, as first and last pixel on each line, are blanked to create the white frame.

The subroutine THRES realizes the dynamic thresholding process. It updates the horizontal and vertical threshold, computes the dynamic threshold, and then proceeds to classify the pixel as object or background by comparing it with the modified dynamic threshold.

Subroutine RUNS searches for runs on a line, encoding them in a run-start code, i.e., store column numbers where a new run starts. As the first run always starts in column one, instead of this, the first-run value (black or white gray level) is stored. At the end of the line, a flag value of size +2 is stored.

Subroutine TRACK realizes the region tracking process by using the previous line run-start coding generated by subroutine RUNS. When a background run is found, subroutine THRES is called to threshold the corresponding line segment with corresponding dynamic threshold, by using factor q = 1. When it is an object run, subroutine THRES is given a factor q = 1.03 to use the lighter threshold value under it and includes the two 8-neighbors at the extreme positions to accept diagonal growing borders.

APPENDIX B

DTRT PROGRAM LISTING

DTRT PROGRAM LISTING

```
C       PROGRAM : DYNAMIC-THRESHOLD WITH REGION-TRACKING
C       AUTHOR  : FERNANDO CONTRERAS
C       PURPOSE : THRESHOLD A COMPLEX SCENE PICTURE
C       METHOD  : DYNAMIC THRESHOLD ALGORITHM WITH REGION
C                 TRACKING ALGORITHM (ROSENFELD & KAK, 1976)

C       VARIABLES
C               A : CURRENT BLOCK BUFFER (TWO RASTER LINES)
C              AO : ODD LINE IN BUFFER A
C              AE : EVEN LINE IN BUFFER A
C             RUN : ARRAY TO STORE RUNS IN CURRENT BLOCK
C              RO : RUNS IN ODD LINE
C              RE : RUNS IN EVEN LINE
C            HTHR : HORIZONTAL THRESHOLD
C            VTHR : VERTICAL THRESHOLD
C            ITHR : INITIAL THRESHOLD VALUE
C          INIBLK : INITIAL DATA BLOCK
C          LASTBL : LAST DATA BLOCK

        INTEGER A(512),RUN(512),BLOCK,SIZE,HTHR,BLK
        INTEGER AO(256),AE(256),RO(256),RE(256),VTHR(256)
        EQUIVALENCE (A(1),AO(1)),(A(257),AE(1)),
     1              (RUN(1),RO(1)),(RUN(257),RE(1))
        CHARACTER*16 FNAME

C   ASK FOR INPUT PICTURE FILENAME
        TYPE *,'IMAGE FILENAME'
        ACCEPT 1,FNAME
1       FORMAT(A16)
        K=INDEX(FNAME,'.')
        IF(K)10,10,20
10      K=INDEX(FNAME,' ')

C   ASSUME DEFAULT EXTENSION
        FNAME=FNAME(1:K-1)//'.PIC'
20      OPEN(UNIT=1,NAME=FNAME,TYPE='OLD',ACCESS='DIRECT',
     1       READONLY,FORM='UNFORMATTED',RECORDSIZE=128)
```

```
C    OPEN OUTPUT FILE WITH EXTENSION "BIN"
     FNAME=FNAME(1:K)//'BIN'
     OPEN(UNIT=2,NAME=FNAME,TYPE='NEW',ACCESS='DIRECT',
    1      FORM='UNFORMATTED',RECORDSIZE=128)

C    ASK FOR INITIAL THRESHOLD VALUE
     TYPE *,'INITIAL THRESHOLD'
     ACCEPT *,ITHR

C    READ PICTURE HEADER TO GET FILE PARAMETERS
     CALL IPICRD(1,A,1)
     K=3.-(A(4)-76.)/4.
     SIZE=64*2**K
     INIBLK=256*A(6)+A(5)
     NBLOCK=256*A(8)+A(7)
     LASTBL=NBLOCK+INIBLK-1

C    MODIFY HEADER AND WRITE IT ON OUTPUT FILE
     A(5)=2
     CALL IPICWT(2,A,1)

C    READ IN FIRST DATA BLOCK
     CALL IPICRD(1,A,INIBLK)
     BLK=2

C    INITIALIZE THRESHOLD VARIABLES
     HTHR=ITHR
     DO 25 I=1,256
25      VTHR(I)=ITHR

C  FIRST LINE THRESHOLDING
     CALL THRES(AO,HTHR,VTHR,1,SIZE,1.)

C    COMPUTE FIRST LINE RUNS
     CALL RUNS(AO,RO,SIZE)

C    TRACK SECOND LINE
     CALL TRACK(AE,RO,VTHR,SIZE)
     CALL RUNS(AE,RE,SIZE)

C    SET WHITE MARGIN
     DO 30 I=1,257
30      A(I)=255
     A(512)=255
```

```
C    PROCESS NEXT BLOCKS
          DO 40 BLOCK=INIBLK+1,LASTBL
C         OUTPUT PREVIOUS PROCESSED BLOCK
             CALL IPICWT(2,A,BLK)
             BLK=BLK+1
C         READ IN NEXT BLOCK
             CALL IPICRD(1,A,BLOCK)
C         TRACK ODD LINE OF CURRENT BLOCK
             CALL TRACK(AO,RE,VTHR,SIZE)
C         COMPUTE RUNS IN ODD LINE OF CURRENT BLOCK
             CALL RUNS(AO,RO,SIZE)
C         TRACK EVEN LINE OF CURRENT BLOCK
             CALL TRACK(AE,RO,VTHR,SIZE)
             CALL RUNS(AE,RE,SIZE)
C         SET WHITE MARGIN
             A(1)=255
             A(256)=255
             A(257)=255
             A(512)=255
C         GO ON NEXT BLOCK
40        CONTINUE

C    CLOSE WHITE MARGIN IN LAST LINE
          DO 50 I=258,511
50        A(I)=255

C    OUTPUT LAST LINE
          CALL IPICWT(2,A,BLK)

C    CLOSE FILES
          CLOSE(UNIT=1)
          CLOSE(UNIT=2)

          STOP
          END
```

```
C    SUBPROGRAM RUNS

C         PURPOSE : RUN-START ENCODING OF A RASTER LINE.
C                   EVERY TIME A NEW RUN IS FOUND, ITS
C                   STARING POSITION IS STORED.

C         VARIABLES

C                   A : LINE BUFFER
C                 RUN : RUNS STORAGE
C                   L : POINTER TO RUN

          SUBROUTINE RUNS(A,RUN,SIZE)
          INTEGER A(256),RUN(256),SIZE
C    STORE FIRST PIXEL VALUE
          K=A(1)
          RUN(1)=K
C    SET POINTER
          L=2
C    PROCESS ONE LINE
          DO 20 I=2,SIZE
C         SEARCH FOR A CHANGE
             IF(A(I)-K)10,20,10
C         FOUND ONE, POSITION STORED
10           RUN(L)=I
C         UPDATE POINTER TO RUN
             L=L+1
C         REPLACE COMPARATOR VALUE
             K=A(I)
20        CONTINUE

C    END OF LINE MARKER
          RUN(L)=SIZE+2

          RETURN
          END
```

```
C    SUBPROGRAM TRACK

C         PURPOSE :  TRACK A LINE HAVING RUNS FROM
C                    PREVIOUS ONE.

C         VARIABLES

C                 A :  LINE BUFFER
C               RUN :  RUN-START LINE CODING
C              VTHR :  VERTICAL THRESHOLD.
C              SIZE :  PICTURE SIZE

          SUBROUTINE TRACK(A,RUN,VTHR,SIZE)
          INTEGER A(256),RUN(256),VTHR(256),HTHR,SIZE

C    INITIALIZE HORIZONTAL THRESHOLD
          HTHR=VTHR(1)
C    WHAT WAS FIRST RUN VALUE?
          IF(RUN(1))20,20,10
C    IT WAS BLACK.  SET THRESHOLDING LIMIT
10        J1=RUN(2)-1
C    THRESHOLD UNDER FIRST RUN COLUMNS
          CALL THRES(A,HTHR,VTHR,1,J1-1,1.)
C    END OF LINE ?
          IF(J1.GE.SIZE)RETURN
C    SET POINTER TO NEXT RUN
          L=3
          GO TO 30
C    IT WAS WHITE.  SET POINTER AND LIMITS
20        J1=1
          L=2
30           J2=MINO(RUN(L),SIZE)
C    THRESHOLD UNDER ACCEPTED RUN
             CALL THRES(A,HTHR,VTHR,J1,J2,1.03)
C         END OF LINE ?
             IF(J2.EQ.SIZE)RETURN
C         SET LIMIT
             J1=RUN(L+1)-1
C          THRESHOLD UNDER REJECTED RUN
             CALL THRES(A,HTHR,VTHR,J2+1,J1-1,1.)
C         END OF LINE ?
             IF(J1.GE.SIZE)RETURN
C         UPDATE LIMIT
             L=L+2
          GO TO 30

          END
```

```
C    SUBPROGRAM THRES
C        PURPOSE :  THRESHOLD ALL OR PART OF A PICTURE LINE,
C                   UPDATING THE DYNAMIC THRESHOLD.
C        METHOD  :  AVERAGE THRESHOLD METHOD (EQ. 12)

C        VARIABLES
C                 BUF :  BUFFER CONTAINING A PICTURE LINE
C                HTHR :  HORIZONTAL THRESHOLD
C                VTHR :  VERTICAL THRESHOLD
C                  I1 :  FIRST COLUMN TO BE THRESHOLDED
C                  I2 :  LAST COLUMN TO BE THRESHOLDED
C              FACTOR :  MULTIPLICATIVE FACTOR FOR TRACKING
C                        CRITERIA

         SUBROUTINE THRES(BUF,HTHR,VTHR,I1,I2,FACTOR)
         INTEGER BUF(256),VTHR(256),Z,HTHR,GRAYL,F,G

C    PROCESS FROM COLUMN I1 TO COLUMN I2
         IF(I1.GT.I2)RETURN
         DO 10 I=I1,I2
            GRAYL=BUF(I)
C        UPDATE HORIZONTAL THRESHOLD
            HTHR=HTHR+F(GRAYL-HTHR)
            Z=VTHR(I)
C        UPDATE VERTICAL THRESHOLD
            Z=Z+G(GRAYL-Z)
            VTHR(I)=Z
C        COMPUTE FINAL THRESHOLD AND APPLY TRACKING FACTOR
            MM=.5*(HTHR+Z)*FACTOR
C        THRESHOLDS PIXEL I
            BUF(I)=MIN0(GRAYL/MM,1)*255
10       CONTINUE

C    BACK TO CALLING PROGRAM
         RETURN
         END
```

```
C    SUBPROGRAM THRES
C        PURPOSE : THRESHOLD ALL OR PART OF A PICTURE LINE,
C                  UPDATING THE DYNAMIC THRESHOLD.
C        METHOD  : MAXIMUM THRESHOLD METHOD (EQ. 13)

C        VARIABLES
C                 BUF : BUFFER CONTAINING A PICTURE LINE
C                HTHR : HORIZONTAL THRESHOLD
C                VTHR : VERTICAL THRESHOLD
C                  I1 : FIRST COLUMN TO BE THRESHOLDED
C                  I2 : LAST COLUMN TO BE THRESHOLDED
C              FACTOR : MULTIPLICATIVE FACTOR FOR TRACKING
C                       CRITERIA

         SUBROUTINE THRES(BUF,HTHR,VTHR,I1,I2,FACTOR)
         INTEGER BUF(256),VTHR(256),Z,HTHR,GRAYL,F,G

C    PROCESS FROM COLUMN I1 TO COLUMN I2
         IF(I1.GT.I2)RETURN
         DO 10 I=I1,I2
            GRAYL=BUF(I)
C        UPDATE HORIZONTAL THRESHOLD
            HTHR=HTHR+F(GRAYL-HTHR)
            Z=VTHR(I)
C        UPDATE VERTICAL THRESHOLD
            Z=Z+G(GRAYL-Z)
            VTHR(I)=Z
C        COMPUTE FINAL THRESHOLD AND APPLY TRACKING FACTOR
            MM=MAXO(HTHR,Z)*FACTOR
C        THRESHOLDS PIXEL I
            BUF(I)=MINO(GRAYL/MM,1)*255
10       CONTINUE

C    BACK TO CALLING PROGRAM
         RETURN
         END
```

```
C    SUBPROGRAM THRES
C        PURPOSE :  THRESHOLD ALL OR PART OF A PICTURE LINE,
C                   UPDATING THE DYNAMIC THRESHOLD.
C        METHOD  :  MINIMUM THRESHOLD METHOD (EQ. 14)

C        VARIABLES
C                 BUF :  BUFFER CONTAINING A PICTURE LINE
C                HTHR :  HORIZONTAL THRESHOLD
C                VTHR :  VERTICAL THRESHOLD
C                  I1 :  FIRST COLUMN TO BE THRESHOLDED
C                  I2 :  LAST COLUMN TO BE THRESHOLDED
C              FACTOR :  MULTIPLICATIVE FACTOR FOR TRACKING
C                        CRITERIA

         SUBROUTINE THRES(BUF,HTHR,VTHR,I1,I2,FACTOR)
         INTEGER BUF(256),VTHR(256),Z,HTHR,GRAYL,F,G

C    PROCESS FROM COLUMN I1 TO COLUMN I2
         IF(I1.GT.I2)RETURN
         DO 10 I=I1,I2
            GRAYL=BUF(I)
C        UPDATE HORIZONTAL THRESHOLD
            HTHR=HTHR+F(GRAYL-HTHR)
            Z=VTHR(I)
C        UPDATE VERTICAL THRESHOLD
            Z=Z+G(GRAYL-Z)
            VTHR(I)=Z
C        COMPUTE FINAL THRESHOLD AND APPLY TRACKING FACTOR
            MM=MINO(HTHR,Z)*FACTOR
C        THRESHOLDS PIXEL I
            BUF(I)=MINO(GRAYL/MM,1)*255
10       CONTINUE

C    BACK TO CALLING PROGRAM
         RETURN
         END
```

```
C    SUBPROGRAM THRES
C        PURPOSE : THRESHOLD ALL OR PART OF A PICTURE LINE,
C                  UPDATING THE DYNAMIC THRESHOLD.
C        METHOD  : COMBINED THRESHOLD METHOD (EQ. 15)

C        VARIABLES
C                BUF : BUFFER CONTAINING A PICTURE LINE
C               HTHR : HORIZONTAL THRESHOLD
C               VTHR : VERTICAL THRESHOLD
C                 I1 : FIRST COLUMN TO BE THRESHOLDED
C                 I2 : LAST COLUMN TO BE THRESHOLDED
C             FACTOR : MULTIPLICATIVE FACTOR FOR TRACKING
C                      CRITERIA

         SUBROUTINE THRES(BUF,HTHR,VTHR,I1,I2,FACTOR)
         INTEGER BUF(256),VTHR(256),Z,HTHR,GRAYL,F,G

C    PROCESS FROM COLUMN I1 TO COLUMN I2
         IF(I1.GT.I2)RETURN
         DO 10 I=I1,I2
            GRAYL=BUF(I)
C        UPDATE HORIZONTAL THRESHOLD
            HTHR=HTHR+F(GRAYL-HTHR)
            Z=VTHR(I)
C        UPDATE VERTICAL THRESHOLD
            Z=Z+G(HTHR-Z)
            VTHR(I)=Z
C        COMPUTE FINAL THRESHOLD AND APPLY TRACKING FACTOR
            MM=MAX1(Z*FACTOR,1.)
C        THRESHOLDS PIXEL I
            BUF(I)=MINO(GRAYL/MM,1)*255
10       CONTINUE

C    BACK TO CALLING PROGRAM
         RETURN
         END
```

```
C    SUBPROGRAM THRES
C        PURPOSE : THRESHOLD ALL OR PART OF A PICTURE LINE,
C                  UPDATING THE DYNAMIC THRESHOLD.
C        METHOD  : LOGICAL "AND" METHOD (EQ. 16)

C        VARIABLES
C                 BUF : BUFFER CONTAINING A PICTURE LINE
C                HTHR : HORIZONTAL THRESHOLD
C                VTHR : VERTICAL THRESHOLD
C                  I1 : FIRST COLUMN TO BE THRESHOLDED
C                  I2 : LAST COLUMN TO BE THRESHOLDED
C              FACTOR : MULTIPLICATIVE FACTOR FOR TRACKING
C                       CRITERIA

         SUBROUTINE THRES(BUF,HTHR,VTHR,I1,I2,FACTOR)
         INTEGER BUF(256),VTHR(256),Z,HTHR,GRAYL,F,G

C    PROCESS FROM COLUMN I1 TO COLUMN I2
         IF(I1.GT.I2)RETURN
         DO 10 I=I1,I2
            GRAYL=BUF(I)
C        UPDATE HORIZONTAL THRESHOLD
            HTHR=HTHR+F(GRAYL-HTHR)
            Z=VTHR(I)
C        UPDATE VERTICAL THRESHOLD
            Z=Z+G(HTHR-Z)
            VTHR(I)=Z
C        APPLY TRACKING FACTOR
            K=HTHR*FACTOR
            Z=Z*FACTOR
C        THRESHOLDS PIXEL I
            BUF(I)=MIN0((GRAYL/K,1)*(GRAYL/Z),1)*255
10       CONTINUE

C    BACK TO CALLING PROGRAM
         RETURN
         END
```

```
C    SUBPROGRAM THRES
C        PURPOSE : THRESHOLD ALL OR PART OF A PICTURE LINE,
C                  UPDATING THE DYNAMIC THRESHOLD.
C        METHOD  : LOGICAL "OR" METHOD (EQ. 17)

C        VARIABLES
C                 BUF : BUFFER CONTAINING A PICTURE LINE
C                HTHR : HORIZONTAL THRESHOLD
C                VTHR : VERTICAL THRESHOLD
C                  I1 : FIRST COLUMN TO BE THRESHOLDED
C                  I2 : LAST COLUMN TO BE THRESHOLDED
C              FACTOR : MULTIPLICATIVE FACTOR FOR TRACKING
C                       CRITERIA

         SUBROUTINE THRES(BUF,HTHR,VTHR,I1,I2,FACTOR)
         INTEGER BUF(256),VTHR(256),Z,HTHR,GRAYL,F,G

C    PROCESS FROM COLUMN I1 TO COLUMN I2
         IF(I1.GT.I2)RETURN
         DO 10 I=I1,I2
            GRAYL=BUF(I)
C        UPDATE HORIZONTAL THRESHOLD
            HTHR=HTHR+F(GRAYL-HTHR)
            Z=VTHR(I)
C        UPDATE VERTICAL THRESHOLD
            Z=Z+G(HTHR-Z)
            VTHR(I)=Z
C        APPLY TRACKING FACTOR
            K=HTHR*FACTOR
            Z=Z*FACTOR
C        THRESHOLDS PIXEL I
            BUF(I)=MIN0((GRAYL/K,1)+(GRAYL/Z),1)*255
10       CONTINUE

C    BACK TO CALLING PROGRAM
         RETURN
         END
```

```
C    FUNCTIONS F AND G
C        PURPOSE : COMPUTE THRESHOLD CORRECTION
C        METHOD : QUADRATIC CORRECTION
C        VARIABLE
C                INT : GRAY-LEVEL THRESHOLD DIFFERENCE

C        REMARKS
C                THESE FUNCTIONS ARE TO BE IMPLEMENTED
C                AS LOOK-UP TABLES IN SUBPROGRAM THRES.

         INTEGER FUNCTION F(INT)
         F=INT**2/271
         IF(INT)10,20,20
10       F=-F
20       RETURN
         END

         INTEGER FUNCTION G(INT)
         G=INT**2/271
         IF(INT)10,20,20
10       G=-G
20       RETURN
         END


C        SAMPLE LINEAR CASE WITH K=0.25

         INTEGER FUNCTION F(INT)
         F=0.25*INT
         RETURN
         END

         INTEGER FUNCTION G(INT)
         G=0.25*INT
         RETURN
         END
```

```
C    SUBPROGRAM IPICRD
C        PURPOSE : READ ONE PICTURE FILE BLOCK AND
C                  STORE IT IN INTEGER FORM

         SUBROUTINE IPICRD(FILE,BUF,BLOCK)
         INTEGER BUF(512),FILE,BLOCK
         LOGICAL*1 BYTE(512)
         READ(FILE'BLOCK)BYTE
         DO 10 I=1,512
         INT=BYTE(I)
         IF(INT.LT.0)INT=INT+256
10       BUF(I)=INT
         RETURN
         END
```

```
C    SUBPROGRAM IPICWT
C        PURPOSE : RECEIVE ONE DATA BLOCK IN INTEGER
C                  FORM AND WRITES IT IN PICTURE FILE
C                  IN BYTE FORM.

         SUBROUTINE IPICWT(FILE,BUF,BLOCK)
         INTEGER BUF(512),FILE,BLOCK
         LOGICAL*1 BYTE(512)
         DO 10 I=1,512
         INT=BUF(I)
         IF(INT.GT.127)INT=INT-256
10       BYTE(I)=INT
         WRITE(FILE'BLOCK)BYTE
         RETURN
         END
```

APPENDIX C

CTSS PROGRAM IMPLEMENTATION

CTSS PROGRAM IMPLEMENTATION

Program Contour Tracing and Sequence Selector (CTSS) accepts images produced by program DTRT. It is written in FORTRAN and it is implemented in a VAX 11/780 computer. It asks for binary image file-name and assumes BIN as default extension. It produces a sequential file with the same name and extension BOX containing the coordinates of boxes that enclose objects detected as possible characters. The Program Listing is in Appendix D.

The main program searches for an object pixel, and calls sub-routine BORDER when one is found.

Subroutine BORDER follows the object border until it gets back into the initial pixel. The location of the next 8-neighbor belonging to S is computed by function NEXT. The white margin inserted by program DTRT guarantees that the object border pixels will not have neighbors out of the image, making follower control for out of picture indexes unnecessary, speeding up the process; and so does handling the image as a vector (only one index).

When the boundary path has been closed, collected features are analyzed to proceed to a rejection or acceptance of the object. If geometric features (height, width, perimeter) passed the tests, incom-pleteness, i.e., an object that touches a picture edge is considered incomplete, is checked.

Then its height and centroid row are compared with those of previously accepted objects to detect a sequence. Then, its coordinates are recorded permanently in file 2 if it is part of such a sequence; otherwise, temporarily, because it could be the first element of the next sequence or line. If the next object found is not forming a sequence with this one, it will be deleted from the file because isolated characters are not expected.

After an object is analyzed, it is deleted from the work image; then a search for another object is started. When there are no more objects, the files are closed and the program stops.

APPENDIX D

CTSS PROGRAM LISTING

CTSS PROGRAM LISTING

```
C    PROGRAM : CONTOUR TRACING AND SEQUENCE SELECTOR
C    PURPOSE : ANALIZE EACH  OBJECT IN A BINARY PICTURE
C              AND SELECT THOSE THAT CAN BE A PART OF A
C              CHARACTER SEQUENCE.
C    METHOD  : ONCE AN  OBJECT  PIXEL IS FOUND, ITS CONTOUR
C              IS TRACED UNTIL BACK INTO THE INITIAL PIXEL;
C              THEN SEVERAL REJECTION CRITERIA ARE APPLIED.
C              IF  THE  OBJECT  PASSES  ALL  TESTS,  THEIR
C              COORDINATES  ARE STORED IN A FILE. FOR EVERY
C              ACCEPTED   OBJECT,   THE   EXISTENCE   OF  A
C              SEQUENTIAL  RELATIONSHIP  WITH ITS  PRIOR IS
C              CHECKED. FAILING THIS TEST, THE PRIOR OBJECT
C              IS  DELETED  FROM  THE  FILE.

      INTEGER A(512),BLOCK,SIZE,ROW,COL,OBJ
      INTEGER IMAGE(65536)
      COMMON IMAGE
      LOGICAL*1 SW
      CHARACTER*16 FNAME
C     ASK FOR INPUT FILE FILENAME
      TYPE *,'PICTURE FILENAME'
      ACCEPT 1,FNAME
1     FORMAT(A16)
      K=INDEX(FNAME,'.')
      IF(K)10,10,20
10    K=INDEX(FNAME,' ')
C     ASSUME DEFAULT EXTENSION
      FNAME=FNAME(1:K-1)//'.BIN'
20    OPEN(UNIT=1,NAME=FNAME,TYPE='OLD',ACCESS='DIRECT',
     1          READONLY,FORM='UNFORMATTED',RECORDSIZE=128)
C     OPEN OUTPUT FILE WITH EXTENSION "BOX"
      FNAME=FNAME(1:K)//'BOX'
      OPEN(UNIT=2,NAME=FNAME,TYPE='NEW',FORM='UNFORMATTED')
      TYPE *,'OBJECT'
      ACCEPT *,OBJ
```

```
C    READ PICTURE HEADER TO GET FILE PARAMETERS
          CALL IPICRD(1,A,1)
          K=3.-(A(4)-76.)/4.
          SIZE=64*2**K
          INIBLK=256*A(6)+A(5)
          NBLOCK=256*A(8)+A(7)
          LASTBL=NBLOCK+INIBLK-1

C    CREATE A WORKFILE
          DO 30 BLOCK=INIBLK,LASTBL
             CALL IPICRD(1,A,BLOCK)
             I=(BLOCK-INIBLK)*512+1
             L=1
                   DO 30 K=I,I+511
                   IMAGE(K)=A(L)
                   L=L+1
30        CONTINUE

          CLOSE(UNIT=1)

C    INITIALIZE VARIABLES
          LASTH=0
          LCENT=0
          LCOL=SIZE
          LBOT=0
          LTOP=0
          SW=.FALSE.

C    PROCESS FILE
          DO 50 I=258,65279
C         SEARCH FOR AN OBJECT PIXEL
             IF(IMAGE(I)-OBJ)50,40,50
C         FOUND ONE.  GO TO BORDER FOLLOWER ROUTINE
40           CALL BORDER(I,SIZE,LASTH,LCENT,LCOL,LBOT,LTOP,SW,OBJ)
50        CONTINUE

C    WAS LAST ITEM A SEQUENCE PART ?
          IF(SW)BACKSPACE 2
          ENDFILE 2

          CLOSE(UNIT=2)
          STOP
          END
```

```
C    SUBPROGRAM BORDER
C    PURPOSE :  TRACE AN OBJECT CONTOUR STARTING WITH GIVEN
C               INITIAL  PIXEL AND DECIDE IF THE OBJECT CAN
C               BE  PART OF A CHARACTER  SEQUENCE.  ACCEPTED
C               OBJECTS  ARE  LISTED  IN  FILE  2.
C    METHOD  :  CONTOUR   TRACING  IS   DONE   USING  EIGHT
C               CONNECTIVITY  CRITERION.  IT GOES ON  UNTIL
C               BACK  ON  INITIAL  POINT;   THEN,  ABSOLUTE
C               REJECTION CRITERIA ARE APPLIED.  IF PASSED,
C               SEQUENTIAL  CRITERIA  ARE  USED.

      SUBROUTINE BORDER(I,SIZE,LASTH,LCENT,LCOL,LBOT,
     1                                        LTOP,SW,OBJ)
      INTEGER BUF(65536),MINC(256),MAXC(256)
      INTEGER ROW,COL,SIZE,HEIGHT,WIDTH,PREV,CENT,OBJ
      LOGICAL*1 SW
      COMMON BUF

C     INITIALIZE VARIABLES
      K=I
      PREV=K-1
      MINROW=K/SIZE+1
      MAXROW=MINROW
      MINCOL=MOD(K,SIZE)
      MAXCOL=MINCOL
      LENGTH=1

C     INITIALIZE COLUMN MINIMA AND MAXIMA
      DO 10 N=1,256
         MINC(N)=256
         MAXC(N)=1
10    CONTINUE

C     SEARCH FOR AN OBJECT PIXEL IN THE 8-NEIGHBOR SET
20    DO 40 N=1,7
C        COMPUTE NEXT NEIGHBOR LOCATION
         NGHB=NEXT(K,PREV,SIZE)
C        IS IT AN OBJECT PIXEL ?
         IF(BUF(NGHB)-OBJ)30,50,30
C        NO,  IT IS NOT
30       PREV=NGHB
40    CONTINUE

C     NO MORE NEIGHBORS;  IT IS AN ISOLATED PIXEL
      MINC(MINROW)=MINCOL
      MAXC(MINROW)=MAXCOL
      GO TO 80
```

```
C    IT IS AN OBJECT PIXEL
50       K=NGHB

C    COMPUTE COORDINATES AND STORE EXTREME VALUES
         ROW=K/SIZE+1
         MAXROW=MAXO(ROW,MAXROW)
         MINROW=MINO(ROW,MINROW)
         COL=MOD(K,SIZE)
         IF(COL.EQ.0)COL=SIZE
         MINCOL=MINO(COL,MINCOL)
         MINC(ROW)=MINO(COL,MINC(ROW))
         MAXCOL=MAXO(COL,MAXCOL)
         MAXC(ROW)=MAXO(COL,MAXC(ROW))

C    UPDATE LENGTH
         LENGTH=LENGTH+1

C    GET BACK INTO INITIAL PIXEL ?
         IF(I.NE.K) GO TO 20

C    YES. COMPUTE DECISION FUNCTIONS
         WIDTH=MAXCOL-MINCOL
         HEIGHT=MAXROW-MINROW
         CENT=(MINROW+MAXROW)/2

C    APPLY REJECTION CRITERIA
C        TOO TALL
         IF(HEIGHT.GE.33) GO TO 80
C        TOO SMALL
         IF(HEIGHT.LE.8) GO TO 80
C        TOO WIDE
         IF(WIDTH.GE.30) GO TO 80
C        PERIMETER TOO SHORT
         IF(LENGTH.LE.20) GO TO 80
C        PERIMETER TOO LONG
         IF(LENGTH.GE.200) GO TO 80
C        IS IT COMPLETE ?
         IF(MINCOL.EQ.2.OR.MINROW.EQ.2) GO TO 80
         IF(MAXCOL.EQ.255.OR.MAXROW.EQ.255) GO TO 80
C        IS IT SIMILAR TO PREVIOUS ACCEPTED OBJECT ?
         IF(IABS(LASTH-HEIGHT).LE.2)GO TO 52
         IF(IABS(LBOT-MAXROW).LE.2)GO TO 60
         IF(IABS(LTOP-MINROW).LE.2)GO TO 60
         GO TO 58
```

```
52        LCENT=IABS(LCENT-CENT)
          IF(LCENT.LE.2)GO TO 60
          IF(ABS(LCENT-HEIGHT/3.).LE.2.)GO TO 60

C         NO, IT IS NOT
58        IF(SW)BACKSPACE 2
          SW=.TRUE.
          GO TO 70

C    YES, THEY ARE A SEQUENCE
60        SW=.FALSE.

C    STORE COORDINATES IN THE FILE
70        WRITE(2)MINROW,MAXROW,MINCOL,MAXCOL

C    SAVE SEQUENCE CHARACTERISTICS
          LASTH=HEIGHT
          LCENT=CENT
          LCOL=MAXCOL
          LTOP=MINROW
          LBOT=MAXROW

C    DELETE FROM WORKFILE
80        CALL DELETE(MINROW,MAXROW,MINC,MAXC,SIZE,OBJ)

C    GO BACK TO CALLING PROGRAM
          RETURN
          END
```

```
C    SUBPROGRAM NEXT
C    PURPOSE : GIVEN THE POSITIONS OF A PIXEL AND ONE OF
C              ITS NEIGHBORS, FOUND THE NEXT NEIGHBOR
C              POSITION IN COUNTERCLOCKWISE SENSE.
C    METHOD  : USING GIVEN POSITIONS, THE RELATIVE
C              POSITION IS COMPUTED AND THEN NEXT
C              NEIGHBOR LOCATION IS CALCULATED BY
C              ADVANCING COUNTERCLOCKWISE AROUND GIVEN
C              PIXEL.

         FUNCTION NEXT(K,PREV,SIZE)
         INTEGER  PREV,SIZE

C    COMPUTE RELATIVE POSITION OF PIXEL AND ITS NEIGHBOR
         KPDIF=K-PREV
         IF(KPDIF)10,10,40

C    NEIGHBOR IS POSTERIOR
10       IF(IABS(KPDIF+SIZE-1)-2)20,30,30
C    TO THE LEFT OR CENTERED
20       NEXT=PREV+1
         RETURN
C    TO THE RIGHT
30       NEXT=PREV-SIZE
         RETURN

C    NEIGHBOR IS PRIOR
40       IF(IABS(KPDIF-SIZE+1)-2)50,60,60
C    TO THE RIGHT OR CENTERED
50       NEXT=PREV-1
         RETURN
C    TO THE LEFT
60       NEXT=PREV+SIZE
         RETURN
         END
```

```
C    SUBPROGRAM DELETE
C    PURPOSE :  DELETE  FROM  WORKFILE AN ALREADY  CHECKED
C               OBJECT.
C    METHOD  :  THE OBJECT IS BLANKED BY REPLACING IT WITH
C               BACKGROUND  VALUE.

          SUBROUTINE DELETE(MINROW,MAXROW,MINC,MAXC,SIZE,OBJ)
          INTEGER A(65536),MINC(256),MAXC(256)
          INTEGER SIZE,BLOCK,ROW,OBJ,BG
          COMMON A
          BG=255-OBJ

C    PROCESS THROUGH ROW RANGE
          DO 20 ROW=MINROW,MAXROW
          BLOCK=(ROW-1)*SIZE
          KI=BLOCK+MINC(ROW)
          KF=BLOCK+MAXC(ROW)

C    PROCESS ONE ROW
          DO 10 I=KI,KF
10           A(I)=BG
20        CONTINUE

          RETURN
          END
```

VITA

Fernando Contreras was born in Santiago, Chile on October 13, 1948. He was graduated from the Liceo de Aplicación in December 1965. In March 1966, he entered the Universidad Técnica del Estado, with which he has been associated since, in Electrical Engineering at the Santiago campus. In March 1971, he continued his studies toward a second engineering degree at the same university. Meanwhile, he worked as a teaching assistant in the Computer Center during 1972 and 1973; then, he gave their first lectures. From 1974 until 1980, he worked at the Computer Center as head of the Academic Area, and taught several computer related courses. In September 1978 he received the title of Electrical Engineer. Since 1981 he has been teaching in the Informatics Department. In 1982, the Universidad de Santiago de Chile, formerly Universidad Técnica del Estado, gave him a scholarship to come to the U.S.A. to work toward the Master's degree at The University of Tennessee in Knoxville, where he entered in the Fall of 1982. He will resume his work at the Universidad de Santiago de Chile upon completion of his degree.