M. Tech (Computer science) Dissertation Series

AN ALGORITHM FOR THINNING OF GRAY LEVEL BENGALI SCRIPT

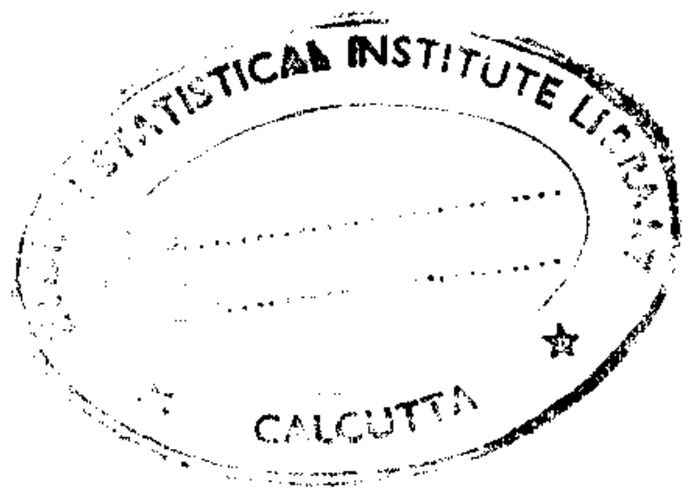
A dissertation submitted in partial fulfillment of the requirements for the M. Tech (Computer Science) degree of the Indian Statistical Institute

By

MANISH SARKAR

under the supervision of

Prof. ASHOK KUMAR DUTTA



INDIAN STATISTICAL INSTITUTE

203, Barrackpore Trunk Road

Calcutta - 700035

C - O - N - T - E - N - T - S

ACKNOWLEDGEMENT

I wish to express my deepest sense of gratitude to my guide Prof. Ashoke Kumar Dutta without whose valuable guidance and encouragement this paper could not have been completed . I will also like to thank Mr. Sur who helped me a lot in getting the input data and to start the work .

I also take this opportunity to thank the people in charge of the KBCS DEPT. of Indian Statistical Institute , Calcutta , for extending their helpful cooperating hands during works in the centre , where all the algorithms have been implemented. I shall also thank the people working in MUSIC AND SPEECH RECOGNITION LAB. for their helpful advice regarding my project.

INTRODUCTION

The machine replication of human capability has been the subject of research for many decades . The machine , having the same reading capability of human is one of such a cherished goal . Still there is a great gap between human reading capability and machine reading capability, and further effort is required to narrow down the gap .

In recent years our very complex and technologically oriented society has created a situation in which more people and organization have become concerned with handling information and fewer with handling materials. One of the MAJOR PROBLEMS IN THE DESIGN of modern information system is automatic pattern recognition,which forms the theme of the work.

CHARACTER RECOGNITION : A BRIEF REVIEW

Ultimate goal of a character recognition research is to develop a machine which can read any text with the same recognition capabilities as human beings .This is the motivation behind all such works. The expectation is that , if the features that people use to recognize characters are properly described and used in a character recognition algorithm then the algorithm should perform as well as human. This descriptive approach is now popular in character recognition approach . A descriptive approach can be provided easily with the flexibility needed to take care of the infinite variation of the character shapes in a category description . A description represents a higher level of intelligence . The description of character involves features (structural details) and the rules under which they compose a character . To achieve an efficient description , the feature used should be independent (i.e. presence of new features or absence of old features should not affect the description of remaining features) .

APPLICATION OF THE SCRIPT RECOGNITION IN OTHER FIELDS

A few of the various applications are as follows :

- Aid for blind people . Such an aid uses photosensors which converts picture information into Electrical Digital Signal.This signal is later converted to sound and tactile simulator [R4,R5].
- Also used for readingnormal text and reproduction of Braille character[R6] .
- Used in postal department for postal address reading and as a reader for handwritten and printed postal codes [R7,R8] .
- Used in Motor Vehicle Bureau as automatic number plate reader and also used as recorder for road traffic control [R9] .
- Used for automatic processing of documents and automatic data entry from sheets . Used in publishing industry as a multipurpose document reader for large scale data processing(as credit card reader[R10]) .
- Used for examination assessment and as a marksheet reader .
- Used in information units and libraries.

IMAGE ACQUISITION

The gray level image of a Bengali alphabet is captured through CCD camera and IMAGE GRABBER ,and stored in the hard disc of a PC/AT. The gray value of the foreground is of low value and the gray value of background is of high value . The range of the gray value is from 0 to 255.

The conversion of input gray level image into two-tone image is done by THRESHOLDING OPERATION .

DEFINITION OF THRESHOLDING :

It is a mapping T from a gray valued image of M x N size to a binary image of M x N size such that

$$T = F\ (f(x,y), p(x,y), (x,y))$$

when f(x,y) = gray value at (x,y)

p(x,y) = local property at (x,y)

If threshold value depends only p(x,y) ---> LOCAL THRESHOLDING.

If threshold value depends only p(x,y) & f(x,y) ---> GLOBAL THRESHOLDING.

If threshold value depends on all parameters ---> DYNAMIC THRESHOLDING.

We create a thresholded image g(x,y) by defining

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) <= T \end{cases}$$

T = threshold value.

RESULT OF CHOOSING INCORRECT THRESHOLD

For the following discussion we assume the gray value of background pixel is greater than gray value of foreground pixel .

If incorrect threshold value > actual threshold value then thresholded image will be connected at arbitrary points. It will cause false additional connection and deformation of shape. This thing has happened in case of fig. 2c.

If incorrect threshold value <= actual threshold value then thresholded image will have unwanted or false disconnection as well as deformation. This thing has happened in case of fig. 2d.

Fig. 2e shows the thresholded image at gray value 123. Fig. 2f shows the thresholded image at gray value 128.

Fig.2a

Fig.2c

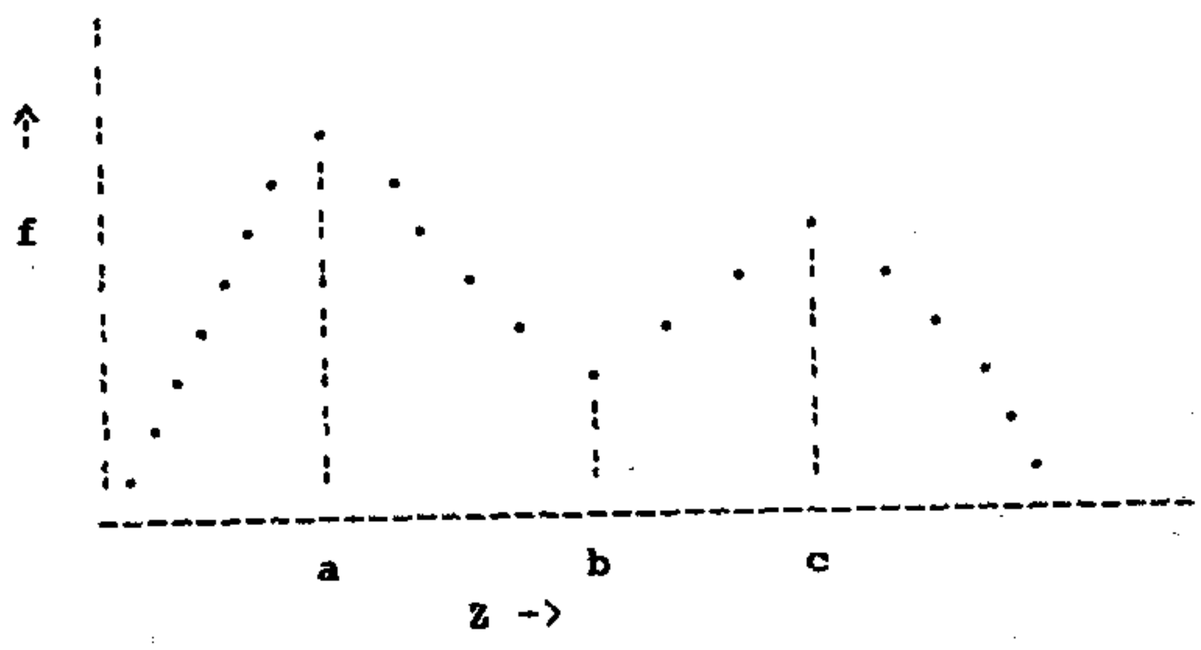
Fig.2d

Fig.2e

Fig.2f

DIFFERENT METHODS FOR THRESHOLDING

1. THRESHOLDING BY HISTOGRAM TECHNIQUE



If (p(b)/min(p(a),p(c))) <= finite predetermine value then gray value at b is the proper threshold .

But if min(p(a),p(c)) is very small then (p(b)/min(p(a),p(c))) is always greater than predetermined value. Moreover there will be a prominent valley in the histogram if the background and object has different statistics.Otherwide it will be very difficult to find a valley.

2. THRESHOLDING BY BOUNDARY DETECTION METHOD

Here to outline the boundary between objects and background, we divide the histogram into two bands separated by threshold T. The goal is to select T so that band B1 contains,as closely as possible,levels associated with the background while B2 contains the level of the objects. As the image is scanned a change in gray

level from one band to other denotes the presence of a boundary. Once B1 and B2 have been selected, the following procedure will be followed :

It consists of following two passes :

f(x,y) is the input picture of N X N size.

Pass 1: For each row in f(x,y)(i.e x=0,1,2.3 ..N-1), create a corresponding row in an intermediate image g1(x,y) using the following relation for y=1,2..N-1:

$$g1(x,y) = \begin{cases} Le, & \text{if the levels of } f(x,y) \text{ and } f(x,y-1) \\ & \text{are in different bands of gray scale} \\ Lb, & \text{otherwise} \end{cases}$$

where Le and Lb are specified edge and background levels,respectively .

Pass 2: For each column in f(x,y)(i.e x=0,1,2.3 ..N-1), create a corresponding row in an intermediate image g2(x,y) using the following relation for y=1,2..N-1:

$$g2(x,y) = \begin{cases} Le, & \text{if the levels of } f(x,y) \text{ and } f(x-1,y) \\ & \text{are in different bands of gray scale} \\ Lb, & \text{otherwise} \end{cases}$$

where Le and Lb are specified edge and background levels,respectively .

The desired image consisting of the points on the boundary of objects different (as defined by T) from the background, is obtained by using the follow-

ing relation for x,y =0,1..N-1:

$$g(x,y)=\begin{cases} \text{Le, if either } g1(x,y) \text{ or } g2(x,y) \text{ is equal to Le.} \\ \text{Lb, otherwise.} \end{cases}$$

3. COOCCURANCE MATRIX :

Here a 256 x 256 matrix A is taken. A[i,j] represents total no. of pixels present in the whole picture of gray value i followed by (horizontally) a pixel of gray value j .

Another 256 x 256 matrix B is taken.

B[i,j] represents total no. of pixels present in the whole picture of gray value i followed by (vertically) a pixel of gray value j .

Now we form a 256 x 256 matrix C by

C[i,j] = A[i,j] + B[i,j]

Take any seed value s as threshold when 0 <= s < 256.

Let,

a = summation of all C[i,j] such that 0 <= i < s and 0 <= j < s.

b = summation of all C[i,j] such that s <= i < 255 and 0 <= j < s.

c = summation of all C[i,j] such that 0 <= i < s and s <= j < 255.

d = summation of all C[i,j] such that s <= i < 255 and s <= j < 255.

Now vary the value of s such that

$$p1 = \frac{a+d}{a+b+c+d} \quad \text{and} \quad p2 = \frac{1}{2}\left(\frac{a}{a+c} + \frac{d}{b+d}\right)$$

are minimum for a certain value of s. This value of s will give the required threshold.

ANOTHER APPROACH

USING LOCAL MAXIMA METHOD

Most of the time, we get bimodel histograms from gray value of the image . Threshold value is in between peaks. The histogram is not very smooth . So it is difficult to find the threshold value for a good image background segmentation . Since it is not very smooth , the steepest rise does not give a good threshold value. Moreover a threshold value may vary from one point to another point . So a threshold value is not a global issue but a local issue .

So in this approach the gray level difference value is found out and this information is used by the automaton to find boundary of object.

It consists of following steps :

1. The difference of gray value (d(gray)/dx) along x axis is found out and stored into ARRAYX.

2. The difference of gray value (d(gray)/dy) along Y axis is found out and stored into ARRAYY .

3. ARRAYX and ARRAYY are superimposed on each other to get complete image boundary .

4. But here the boundary may be more than one element thick . so locally maximum gradient method is used .

it consists of following steps :

4.1. Find (d(gray)/dx) & store in ARRAYX.

4.2. Find (d(gray)/dy) & store in ARRAYY.

4.3. For each element of ARRAYX (e.g. ARRAYX[i][j]) transform it to white if ARRAYX[i][j] > theta1 & ARRAYX[i][j-1] < theta1 & ARRAYX[i][j+1] < theta1 . Otherwise transform it to black.

4.4. Similarly transform each element of arrayy to black and white .

4.5. Superimpose ARRAYX[i][j] on ARRAYY[i][j] if ARRAYX[i-1][j] < theta1 and ARRAYX[i+1][j] < theta1 .

"theta1" is some predetermined value .

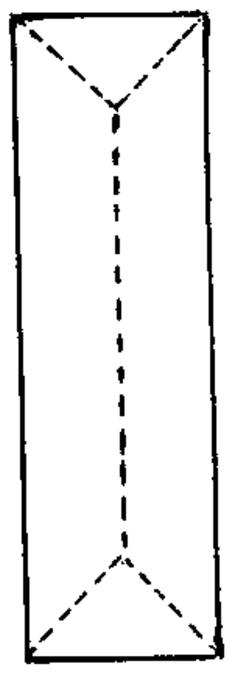
Here main advantage is that theta1 can be changed widely without affecting boundary . So theta1 is much more softer than threshold value because if threshold value is slightly changed,it affects the image heavily. Softness of theta1 comes from smoothness obtained by taking the difference of gray value .

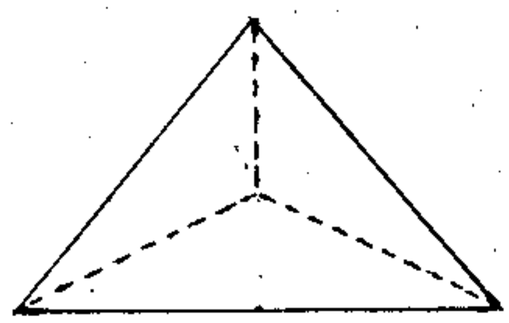
This algorithm is applied on different gray valued pictures. Theta1 is kept as 30. If we change theta1 to 20 or 40 there will not be any appreciable change in the picture.This kind of softness is the main advantage of this algorithm(see Fig 2)

THE DEFINITION OF THINNING

The skeleton of a region may be defined via the medial axis transform (MAT) proposed by BLUM[1967]. The MAT of a region R with border B is defined as follows.

For each point p in R,we find its closest neighbour in B. If p has more than one such neighbour,then it is said to belong to medial axis(skeleton) of R.





Medial axis of simple regions.

Thinning operation should satisfy following criteria :

1. It does not remove end points.
2. It does not break continuity.
3. It does not cause excessive erosion.

PROPERTIES OF A THINNING OPERATOR

A thinning operator is a mapping function associated with each digital set T(s) such that for all digital set "S"

1. T(s) is connected if S is connected.
2. S and T(s) will have same no. of 8-components and same no. of 4-component in T(s) & complement(S). Also (complement(S) U S = image
3. for any finite digital set there exists a no. n such that

$$T^{n}(s) = T^{n+1}(s).$$

$T^{n}(s)$ = n_th iteration.

4. if a point P is in [S-T(s)] then this point is simple (i.e. this point can be removed) .

DIFFERENT EXISTING THINNING ALGORITHMS

1. ZHANG AND SUEN's algorithm.

It is applicable only to binary images.

It consists of several iteration:

Each iteration consists of two passes:

First pass:

1.1> Here a point will be deleted if it satisfies all the following conditions :

1. $2 <= B(p) <=6$

2. $A(p) =1$

3. And at least any one of the following the condition is true

(1) $p_2 =0$

(2) $p_8 =0$

(3) $p_4 = 0$ and $p_6 = 0$

Scan the window one position and apply step 1.1 .

When all the pixel of the given image have been processed and atleast one pixel has become '0' then go to pass II else exit .

Second pass:

2.2> Here a point will be deleted if it satisfies all the following conditions

1. $2 <= B(p) <=6$

2. $A(p)=1$

And at least any one of the following the following condition is true

(1) $p_4 =0$

(2) $P_6 = 0$

(3) $P_2 = 0$ and $P_8 = 0$

B(p) = no. of nonzero neighbours of p

A(p) = no. of '01' or '10' occurrence in the 3 X 3 window .

When all the pixels of the given images have been processed and atleast one pixel has become '0' then go to pass I else exit .

HOLT et. al. thinning algorithm

Holt's algorithm is the Boolean representation of the modified Zhang and Suen's algorithm.

In parallel picture processing the new value given to a point at the nth iteration depends on its own value as well as those of its eight neighbours at the (n-1) th iteration so that all the pictures points can be processed simultaneously . It is assumed that 4 X 4 windows are used and each element is 8-connected .The window is scanning the image from left to right and top to bottom way .

If one of the four conditions, as shown below,is satisfied then the pixel will not be deleted :

1. The candidate pixel will not be on the edge .
2. The value of the west and south neighbours will be '1'and the east neighbour will be on edge .

3. The value of west and east neighbours will be '1' and the south neighbour will be on edge .

4. The east,south-east and south neighbours will be on edge. Repeat the procedure until no '1' to '0' transform takes place .

The overall expression for survival is given bellow :

$$p_1 \text{ and } (\text{not } (\text{EDGE } p_1) \text{ or } (\text{EDGE } p_4 \text{ and } p_2 \text{ and } p_6) \text{ or } (\text{EDGE } p_6 \text{ and } p_8 \text{ and } p_4) \text{ or } (\text{EDGE } p_4 \text{ and EDGE } p_5 \text{ and EDGE } p_6))$$

An EDGE is defined as a pixel P that can be represented in a 3 X 3 window satisfying all the following conditions :

1 . $p_1 = 1$

2 . $2 <= B(p_1) <= 6$

3 . $A(p_1) = 1$

where $A(p_1)$ and $B(p_1)$ as defined in previous sections .

p1	p2	p3
p4	p	p6
p7	p8	p9

A 3 X 3 window with respect to point p.

REGION GROW AUTOMATON TECHNIQUE

The basic idea of skeletonisation from gray image is the selection of the window so that the window lies maximally in the figure with two diametrically oppo site sides on the boundary of the figure . This is done here through the use of an automaton . From top of the input file go on searching for a gray value jump which is greater than thetal . when this kind of jump occurs then we are sure that automaton has entered into the object .The automaton after entering the image area grows itself and adjust its own position , till it is large enough to reach at least two boundaries on the opposite sides . The central position of the automaton is considered to be the thinned image for that one . The background(environment) and the history of own motion is used for properly guiding the automaton .The usual constraints of continuity , connectivity and simplicity are kept into consideration .

This program consists of following modules :

1. MAIN
2. WINDOW
3. MOVE
4. RECURSION
5. PUSH
6. ERASE
7. CONNECT
8. CHECK
9. FINDPATH
10. SEARCH

Important data structure:

```
struct      point {  int  x;
                     int  y;

                  } STACK[50],LSTACK[50];

int         ARRAY[60][60] , DUMMY[60][60],OUTARRAY[60][60];

int         U,L,D,R
```

****
* MAIN *
****

The input image is stored in ARRAY and in DUMMY. DUMMY is used to note that which part of the image is already traversed i.e. it keeps the record of the erased portion of the image .OUTARRAY is used to store the thinned image .

From this module MOVE is called.

Then thinned image is stored in the output file .

****
* MODULE WINDOW *
****

Input parameters :

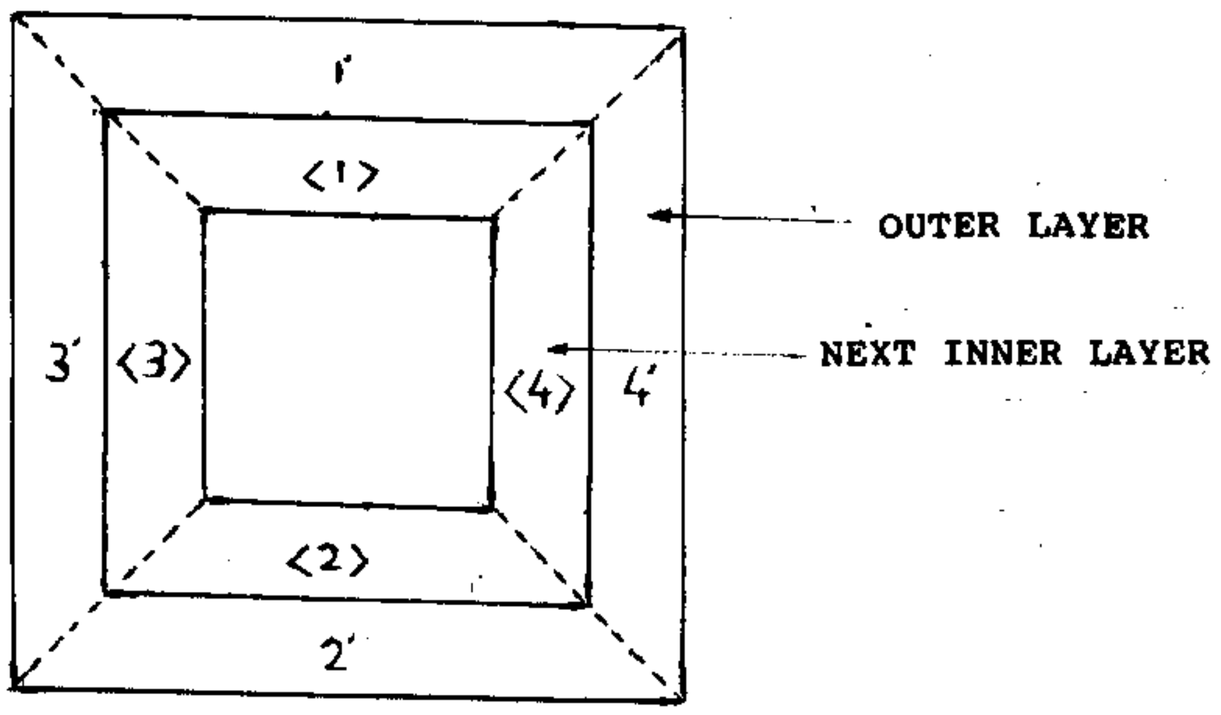
(x,y) : coordinate of the pixel about which the square window has to be formed .

Length : length of the square window .

Output parameters :

U,D,L,R.

One window of size length X length is formed about point(i,j).



An N X N window

Take a point from region <1> and subtract its gray value from a point which is just above it and lies in the region 1' . If the absolute value of the result is greater than thetal then U=1. For all points in the region <1> this measure is continued until (U=1) .

Take a point from region <2> and subtract its

gray value from a point which is just below it and lies in the region 2' . If the absolute value of the result is greater than thetal then D =1.

For all points in the region <2> this measure is continued until (D=1) .

Take a point from region <3> and subtract its gray value from a point which is just left of it and lies in the region 3' . If the absolute value of the result is greater than thetal then L=1. For all points in the region <3> this measure is continued until (L=1) .

Take a point from region <4> and subtract its gray value from a point which is just it and lies in the region 4' region. If the absolute value of the result is greater than thetal then R=1. For all points in the region <4> this measure is continued until (R=1) .

Let us consider a window of 5 X 5 size .

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

5 X 5 window

if absolute(gray value of 7 - gray value of 2) > thetal
19

```
then  U=1;
      else  if absolute(gray value of  8-gray value of 3) > theta1
then  U=1;
      else  if absolute(gray value of  9-gray value of 4) > theta1
then  U=1;
      else  U=0;
```

Similarly (7 and 6) , (12 and 11) ,(17 and 16) are compared to decide whether L is equal to zero or not .

Similarly (17 and 22) , (18 and 23) ,(19 and 24) are compared to decide whether D is equal to zero or not .

Similarly (9 and 10) , (14 and 15) ,(19 and 20) are compared to decide whether R is equal to zero or not.

If the input image touches the boundary of the given image then the window of those boundary points will touch the boundary. Then error signal will come.

```
********************
*  MODULE  MOVE  *
********************
```

Here the input file is sequentially searched from top to find any appreciable jump of gray value (>= theta1) from low gray value to high gray value . If a jump of a gray value is found then we are sure that we have entered into the object from the background . Here

it is assumed no part of the object is on the boundary of the given image(then we will not be able to notice the jump of gray value from low to high.) After entering into the object the PUSH module will be used to expand the growing region within the boundary. During this PUSH operation some 'junction point' may be noticed which will be stored in "STACK" . So after using PUSH "STACK" has to be checked for junction points .If junction points are found then PUSH

module has to be used for all those junction points . Then whole file is searched to check whether there is a portion of the object which is not covered by the growing region . If such a region is found then one point will be taken from this region and module PUSH will be used for this point.

```
**************************
*  MODULE  RECURSION  *
**************************

Input parameters  :

        i,j     :  these   parameters indicate the initial
point  . This   point  is  always  a  junction  point  .

        i1,j1   :  these   parameters indicate the final
point  .
```

has to use either PUSH(i,j-1) or PUSH(i,j+1).

```
.............
 ...*******...
 ...***p***...
 ...***1***...
 ...*******...
 .............
```

Coordinate of p is (i,j) and coordinate of 1 is (i+1,j).

This will produce a zigzag path which is not wanted.To avoid this difficulty one 3 X 3 window with respect to p is opened. Since U=D=L=R=0,it is expanded to 5 X 5.Now U=1; D=L=R=0. Now the window is expanded to next higher level i.e. 7 X 7. For this new window D=1. So, from this observation we are sure that if we had used PUSH(i+1,j),it would cause a zigzag motion .So point p is selected as a point of thinned image and automaton is moved in leftward or rightward direction(if possible).

For different value of U,D,L,R we have to consider the following cases :

CASE	R	L	D	U
1.	0	0	0	1
2.	0	0	1	0
3.	0	1	0	0
4.	1	0	0	0
5.	0	0	1	1
6.	0	1	0	1
7.	0	1	1	0
8.	1	0	0	1
9.	1	0	1	0
10.	1	1	0	0
11.	0	1	1	1
12.	1	0	1	1
13.	1	1	0	1
14.	1	1	1	0
15.	1	1	1	1

CASE 1:

U=1,D=0,L=0,R=0 : it means window can not be incremented in the upward direction (it has already touched the upper boundary).If the current window size = k X k then we should increase the window size around the point ARRAY(i,j) to size (k+2) X (k+2) .If D=1 or L=1 or R=1 k = k+2.

If 'D' of new window is equal to zero and ARRAY[i+1][j] is not already traversed then it is certain that region can expand in the downward direction. If it is not already traveresed we erase upper portion of the old window by calling ERASE(U,k-2,i,j). Now procedure RECURSION(i,j,i+1,j) is used.

If (D # 0) or ARRAY(i+1,j) is already traversed then we will look for a path in the rightward direction. So if (L=0) and ARRAY(i,j-1) is not already traversed then automaton can expand in the leftward direction. So we erase left portion of the old window by calling ERASE(L,k,i,j). Now procedure RECURSION(i,j,i,j-1) is used.

If (D#0 or ARRAY(i+1,j) is already traversed) and (L#0 or ARRAY(i,j-1) is already travesed) then we will look for a path in the rightward direction . So if R=0

and ARRAY(i,j+1) is not traversed then automaton can expand in the leftward direction. We erase the right portion of the old window by calling ERASE(R,k-2,i,j). Procedure RECURSION(i,j,i,j+1) is used.

If all the above three conditions are not true then apparently there is no path from this point. Erase the region of size (k-2) X (k-2) around ARRAY(i,j). So FINDPATH(k,i,j) is called to find any path from it (if exists).

CASE 2:

U=0,D=1,L=0,R=0 : it means window can not be incremented in the downward direction (it has already touched the lower boundary). If the current window size = k X k then we should increase the window size around the point ARRAY(i,j) to size (k+2) X (k+2) .If U=1 or L=1 or R=1 k = k+2;

If 'U' of new window is equal to zero and ARRAY(i-1,j) then it is certain that region can expand in the upward direction . If it is not traveresed we erase upper portion of the old window by calling ERASE(D,k-2,i,j).Now procedure RECURSION(i,j,i+1,j) is used.

If U # 0 or ARRAY(i-1,j) is already traversed then we will look for a path in the rightward direction. So if L=0 and ARRAY(i,j-1) is not already traversed then automaton can expand in the leftward direction. We erase left portion of the old window by calling ERASE(L,k-2,i,j). Procedure RECURSION(i,j,i,j-1) is used.

If (U # 0 or ARRAY(i-1,j) is already traversed) and (L#0 or ARRAY(i,j-1) is already travesed) then we will look for a path in the rightward direction . So if R=0 and ARRAY(i,j+1) is not traversed then automaton can expand in the rightward direction. We erase the right portion of the old window by calling ERASE(R,k-2,i,j). Procedure RECURSION(i,j,i.j+1) is used.

If all the above three conditions are not true apparently there is no path from this point. Erase the region of size (k-2) X (k-2) around ARRAY(i,j). FINDPATH(k,i,j) is called to find any path from it (if exists).

Similar approach is used for case 3 and case 4.

CASE 5:

U=1,D=1,L=0,R=0 : it means window can not be incremented in the upward direction and downward direction (it has already touched the upper boundary and lower boundary).If the current window size = k X k then we should increase the window size around the point (i,j) to size (k+2) X (k+2). If L=1 or R=1 k = k+2.

If 'L' of new window is equal to zero and ARRAY(i,j-1) is not already trversed then it is certain that region can expand in the leftward direction. So we erase left portion of the old window by using (k,i,j). Procedure RECURSION(i,j,i,j-1) is used.

If L # 0 or ARRAY[i,j-1] is already traversed then we will look for a path in the rightward direction. So if R=0 and ARRAY[i,j+1] is not already traversed then automaton can expand in the rightward direction. So we erase left portion of the old window by calling ERASE(1,i,j).Now procedure RECURSION(i,j,i,j+1) is used.

If both the above conditions are not true apparently there is no path from this point.Erase the region of size k X k around ARRAY(i,j). So FINDPATH(k,i,j) is called to find any path from it (if exists).

Similar approach is used for cases 6,7,8,9,10.

CASE 11:

U=1,D=1,L=1,R=0 : it means window can not be incremented in the upward, downward, leftward direction (it has already touched the upper, lower and left boundary).

At first this point is recorded in OUTARRAY by using CHECK and then this point is erased.If DUMMY[i][j-1] > 0 (i.e it is not traversed already) then recursively

PUSH(i,j-1) is called. If DUMMY(i,j-1) < 0 window is expanded to next higher size (k+2) X (k+2). If R of this new window is equal to 1 then (k+2) X (k+2) window around ARRAY(i,j) is erased by using ERASE else k X k window around ARRAY(i,j) is erased. FINDPATH(k,i,j) is called.

CASE 15:

U=1,D=1,L=1,R=1 : It means window can not be incremented in the upward, downward, leftward and rightward direction (as it has already touched the upper, lower, left and rightward boundary). We record this point in OUTARRAY

ERASE. Apparently there is no path from this point. So FINDPATH(k+2,i,j) is called to find any path from it (if exists).

```
**********************
*  MODULE  ERASE  *
**********************
```

Input parameters :

choice : it may be any one of 'U','D', 'L','R'.

length : the length of the window .

i,j : the point with respect to which a

specific part (upper,lower,left,right) of the window has to be erased.

If choice = U erase upper portion of the window .

If choice = D erase lower portion of the window .

If choice = L erase left portion of the window .

If choice = R erase right portion of the window .

To erase a portion of window corresponding points in "DUMMY" is manipulated . Each pixel value of corresponding point is replaced by its previous gray value with negative sign .Initially all DUMMY[i,j] >0 . So no ambiguity exists.

EXAMPLE :

If the length=3, choice ='1' , and point(i,j) is represented by point p the following figure shows the erased portion.

(Erased portion is shown by '-' sign . Non erased portion of the window is shown by dot '.')

```
-----------------                -----------------
|  .  |  .  |  .  |              |  -  |  -  |  .  |
-----------------                -----------------
|  .  |  p  |  .  |     -->      |  -  | -p  |  .  |
-----------------                -----------------
|  .  |  .  |  .  |              |  -  |  -  |  .  |
-----------------                -----------------
```

****
* MODULE CONNECT *
****

Input parameters :

(initi,initj),(fini,finj) are co_ordinates of the two points which have to be connected .

This module is used to connect two given points (initi,initj) & (fini,finj). Connecting line is drawn

to follow minimum distance between points .

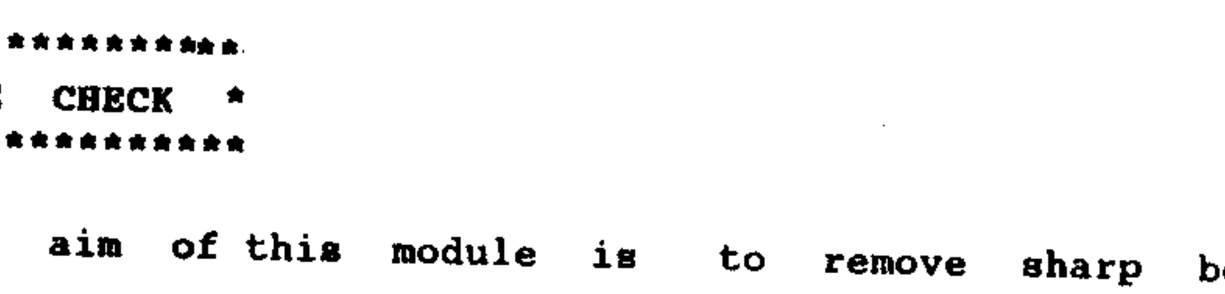
Connecting line between two points. "." represents the points which are to be connected. "*" represents the connecting line.

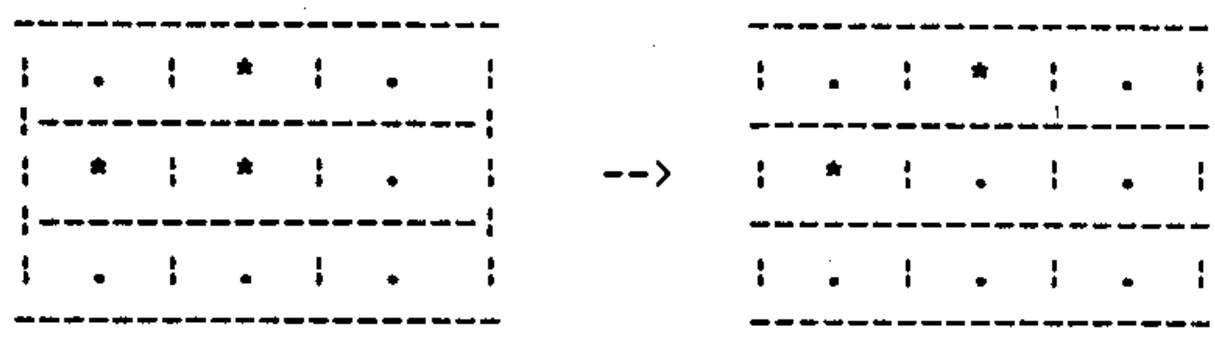
During connection CHECK is called to avoid any sharp bend.

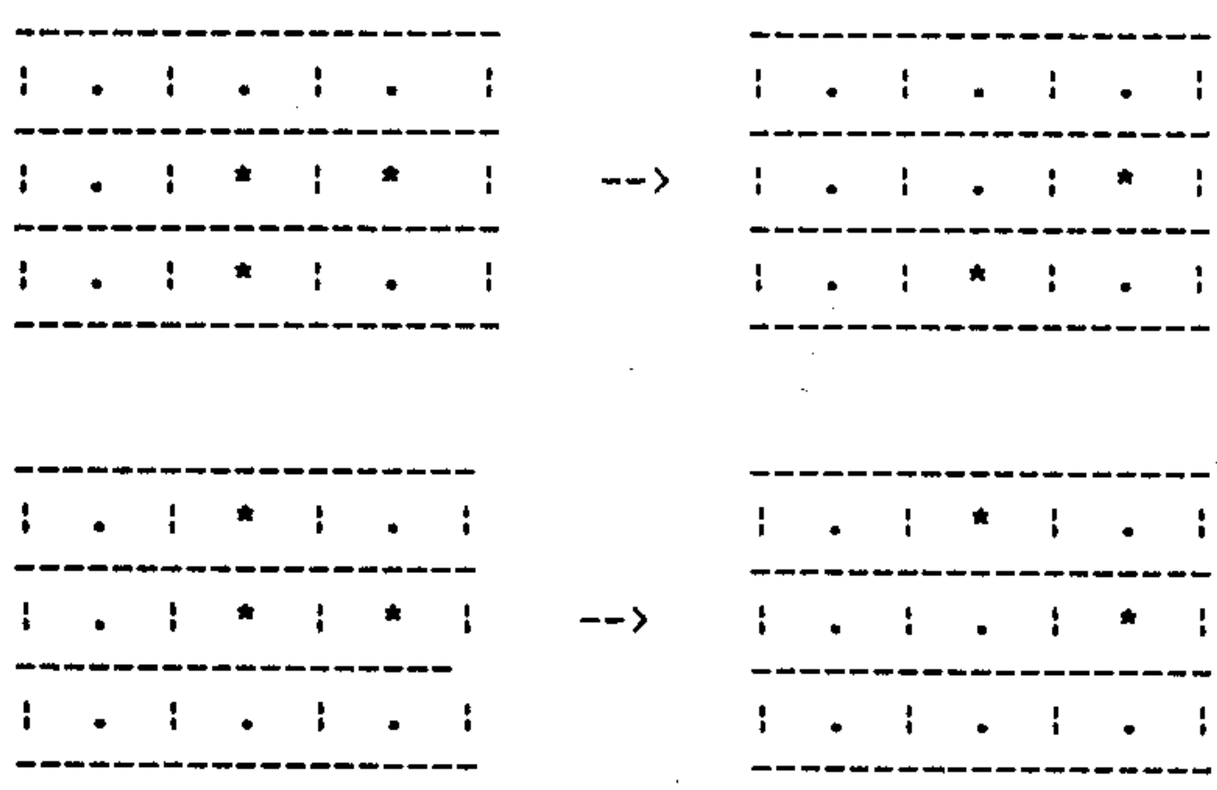
```
**********************
*  MODULE  CHECK  *
**********************
```

The aim of this module is to remove sharp bends of the thinned picture . So it smooths the image by using following rules:

For the following figures "." represents background and "*" represents object point .

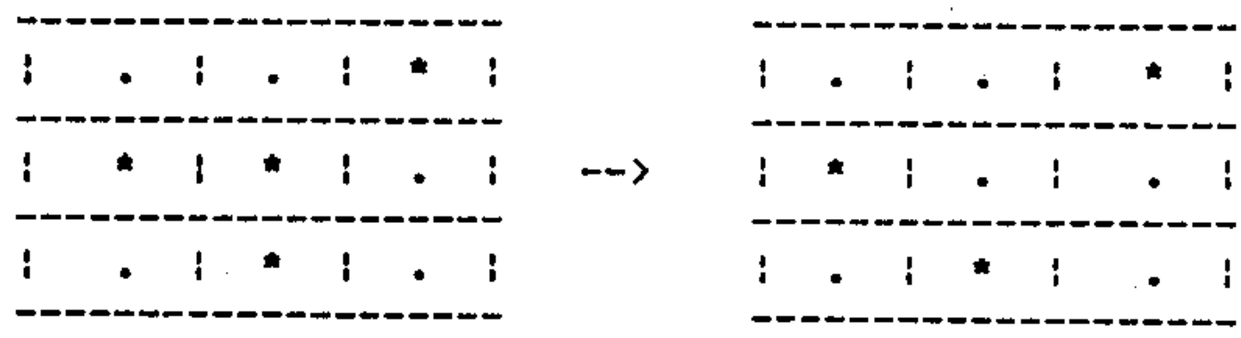




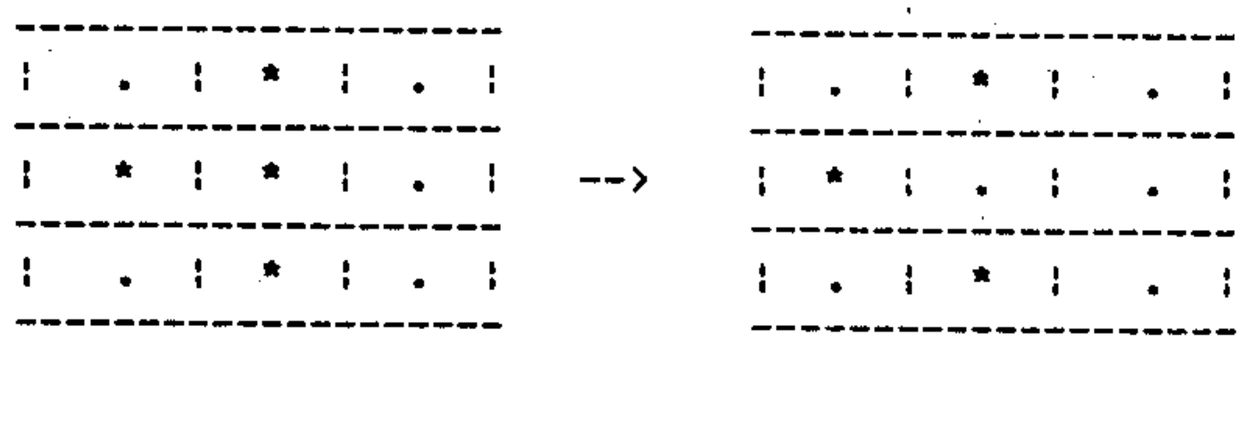


During smoothing connectivity should remain intact.

So following transform is not valid .



Also following transform is not valid .



```
************************
*  MODULE  FINDPATH  *
************************
```

Input parameters :

k : window length.

(i,j) : the coordinate of the point from which a path is to be found out .

In some cases it may happen that window can not expand further; when the growing region can expand . Consider the following binary picture .

```
....................
............*.......
............**......
..........******....
........*p*.........
........***.........
....................
....................
....................
....................
```

Here a window with respect to point p can be expanded upto 5 x 5 . Now it will give U =1; D =1; L =1; R =1. So region can not grow further. But actually from the picture we see that region can grow along the uppermost and rightmost corner . So this module is needed .

It will create a window with respect to p. The window size is K X K . Now at each boundary point it will try to find a path by using the procedure "SEARCH(K,i,j)" . Now "SEARCH" has to return possible points through which the

region can expand further . " SEARCH "will push these points into "LSTACK" .Then FINDPATH will pop these possible points from "LSTACK" { say (x,y) } and it will use PUSH(x,y) . If push succeeds to grow the region then this (x,y) point will be connected to the point 'p'.Otherwise the point (x,y) will not be connected to the point 'p'.

```
**********************
*  MODULE SEARCH  *
**********************
```

Input parameters :

length : the length of the window of max. size around point (i,j).

(i,j) : the point with respect to which search is to be carried out .

At first the window of size =length is to be constructed with respect to the point (i,j) .Now the border points are taken and they are pushed into "LSTACK". Central border points are pushed at last so that they can be popped at first.

p1	p2	p3
p4	p	p6
p7	p8	p9

p2,p4,p6,p8 are called as the central border points for this 3 x 3 window .

CONCLUSION:

The scope of this work is limited by the following constraints.

1. The characters of a particular font are only considered in this work . But it is assumed that any other font with little variation will also be recognized by the same method.

2. The compound characters are left out of the scope of the work as it is found to be much more complicated to be recognized by the same method.

3. The algorithm is not a very simple one.

REFERENCES :

1. A.K.Datsun : An experimental procedure for handwrit ten character recognition :IEEE transactions on computers , viol 23, number 5 ,up 536-545 ,may 5,1974 .

2. A. Rosenfield and A. C. Kak : "Digital Picture Processing ",Academic press,1987.

3. R.C.Gongalez : "Digital Image Processing, "Addision-Wesley,1987 .

4. A.k.Dutta : "A generalized normal approach for

description and analysis of major Indian scripts",journal of I.E.T.E,vol 30 , no ,6 , pp 155-161 ,1984 .

5. Zhang and Suen "A fast parallel algorithm for thinning Digital patterns " .CASM , Vol 27 , no. 3,pp236 -239 ,March 1984 .

6. Holt ,Stevert ,Clint and Perrot "An improved parallel Thinning algorithm " .CASM, vol 30, no 2 ,Feb ,1987 ,pp 156 - 160 .

FURTHER READING

R1. K.S.Fu Syntactic pattern recognition and applications : Prentice Hall ,Englewood Cliffs ,New Jersey (1982).

R2. J.Mantas, An overview of character recognition methodologies , Pattern Recognition 18,425-430(1986).

R3. Govian, A.P.Sivaprasad,Character recognition - A review, Pattern Recognition 23 ,671-683,1990.

R4. J.C.Bliss , A relatively high resolution reading aid for the blind ,IEEE T.man Mach. System 10,1-9 (1969) .

R5. R.D.Badoux,DELTA text reader for blind ,computerized braille production,proc . 5th. Int. workshop, Winterthur, Switzerland,pp. 21-25 (30 October - 1 November 1985).

R6. C.J.V Sprocen and F.Bruggman ,Raised type reading, mini and microcomputers and their applications,Proc.

ISMM Int. Symp. Sant Feliu de Guixols , Spain ,pp. 274-277(25-28 june 1985).

R7. C. W. Swonger, An evaluation of character normalization, feature extraction and classification techniques for postal mail reading, Proc. Automatic Pattern Recognition,Washington ,D.C.,U.S.A,pp 67-87 (6th May 1969).

R8. K .Notbhom and W. Hanisch,Automatic digit recognition in a mail sorting machine , Nachrichtentech, Electron (Germany) 36,472-476 (1986).(In German).

R9. A.Gyafas ,Experiments concerning the inspection and control of car and truck in France, Koezlekdes Tud. Sz. 24, 8 -91 (1974).

10. N.M.Herbst and C.N.liu ,Card based personal identification system ,IBM Technical Disclosure Bull .(U.S.A.) 22 ,4291-4293(February 1980).

It connects (i,j) and (i1,j1) by using procedure CONNECT .Now point (i,j) is pushed into "STACK" for further consideration (because from this junction point more than one path may be found . Now we are considering one path only .In future we may find other paths .). Then Procedure PUSH is used on point (i,j) to expand the growing region.

****
* MODULE PUSH *
****

Input parameters :

i,j : these parameters indicate the point about which the region will try to grow up .

At first one 3 x 3 window is to be set up around (i,j) .

<u>CASE 1:</u>

U=0,D=0,L=0,R=0 :

If we find U,D,L,R all zero then window will be further expanded to 5 x 5 else we will stop the increment. This increment will continue until it touches atleast one edge.

This approach will create a problem :

With respect to point p in the following figure a 3 X 3 window is created.

Now U=D=L=R=0. So the window is expanded to 5 X 5.

Now U=1;D=L=R=0. So as a next step PUSH(i+1,j) will be used. Point p is erased. A 5 X 5 window with respect to 1 is created. D=1;U=L=R=0. Since p is already erased it

THE ROUTE FOLLOWED BY THE AUTOMATION TO THIN THE

FIG 1 .

k = max window length about a specific point (x,y).

x	y	k	U	D	L	R
21	5	3	1	0	0	0
21	4	3	1	0	0	0
21	3	3	1	1	1	0
21	4	3	1	0	0	0
21	5	3	1	0	0	0
21	7	3	1	0	0	0
22	7	5	1	0	0	0
22	8	5	1	0	0	0
22	9	5	1	0	0	0
22	10	5	1	0	0	0
22	11	5	1	0	0	0
22	12	5	1	0	0	0
22	13	5	1	0	0	0
22	14	5	1	0	0	0
22	15	5	1	0	0	0
22	16	5	1	0	0	0
22	17	5	1	0	0	0
22	18	5	1	0	0	0
22	19	5	1	0	0	0
22	20	5	1	0	0	0
22	21	5	1	0	0	0
22	22	5	1	0	0	0
22	23	5	1	0	0	0
23	23	7	1	0	0	0
24	23	9	1	0	0	0
25	23	3	0	0	1	0
25	24	5	0	0	1	0
25	25	7	0	0	1	1
26	25	7	0	0	1	0
26	26	9	0	0	1	0
26	27	11	0	0	1	1
27	27	15	1	0	1	0
28	27	15	0	1	0	0
28	28	7	1	0	0	0
29	28	9	1	0	0	0
30	28	7	0	0	0	1
30	27	9	0	0	0	1
30	26	11	0	1	0	1
30	25	11	0	1	0	0
30	24	11	0	1	0	0
30	23	11	0	1	1	0
30	24	11	0	1	0	0

```
30  25          11          0 1 0 0
30  26          11          0 1 0 1
30  27          9           0 0 0 1
FINDPATH    ( 8  32)        { it means findpath
                              procedure is called
                              about point (28,32) .}
28  32          5         1 1 0 0
28  33          5         1 1 0 0
28  34          5         1 0 0 0
28  35          3         1 0 0 0
29  35          5         1 1 0 0
29  36          5         1 0 0 1
30  36          5         0 1 0 0
30  37          5         1 0 0 1
31  37          5         0 0 1 0
32  37          3         0 0 1 0
32  38          5         0 0 1 1
33  38          5         0 0 1 0
34  38          5         0 0 1 0
35  38          5         0 0 1 0
35  39          7         0 0 1 1
36  39          7         0 0 0 1
37  39          7         0 0 0 1
37  38          9         0 0 1 1
38  38          11        0 1 0 1
38  37          11        0 1 0 0
38  36          7         1 0 0 0
39  36          9         1 0 0 0
39  35          7         1 0 0 0
40  35          9         1 1 0 0
40  34          7         1 0 0 0
41  34          9         1 1 0 0
41  33          9         1 1 0 0
41  32          9         1 1 0 0
41  31          7         1 0 0 0
41  30          7         1 0 0 0
41  29          7         1 0 0 0
41  28          7         1 0 0 0
41  27          7         1 0 0 0
41  26          7         1 0 0 0
41  25          7         1 1 0 0
41  24          7         1 1 0 0
41  23          7         0 1 0 0
40  22          7         1 1 0 0
40  21          7         0 1 0 0
39  20          7         1 1 0 0
39  19          7         0 1 0 0
38  18          7         1 1 0 0
38  17          7         0 1 1 0
37  17          7         1 0 0 0
37  16          7         0 1 1 0
```

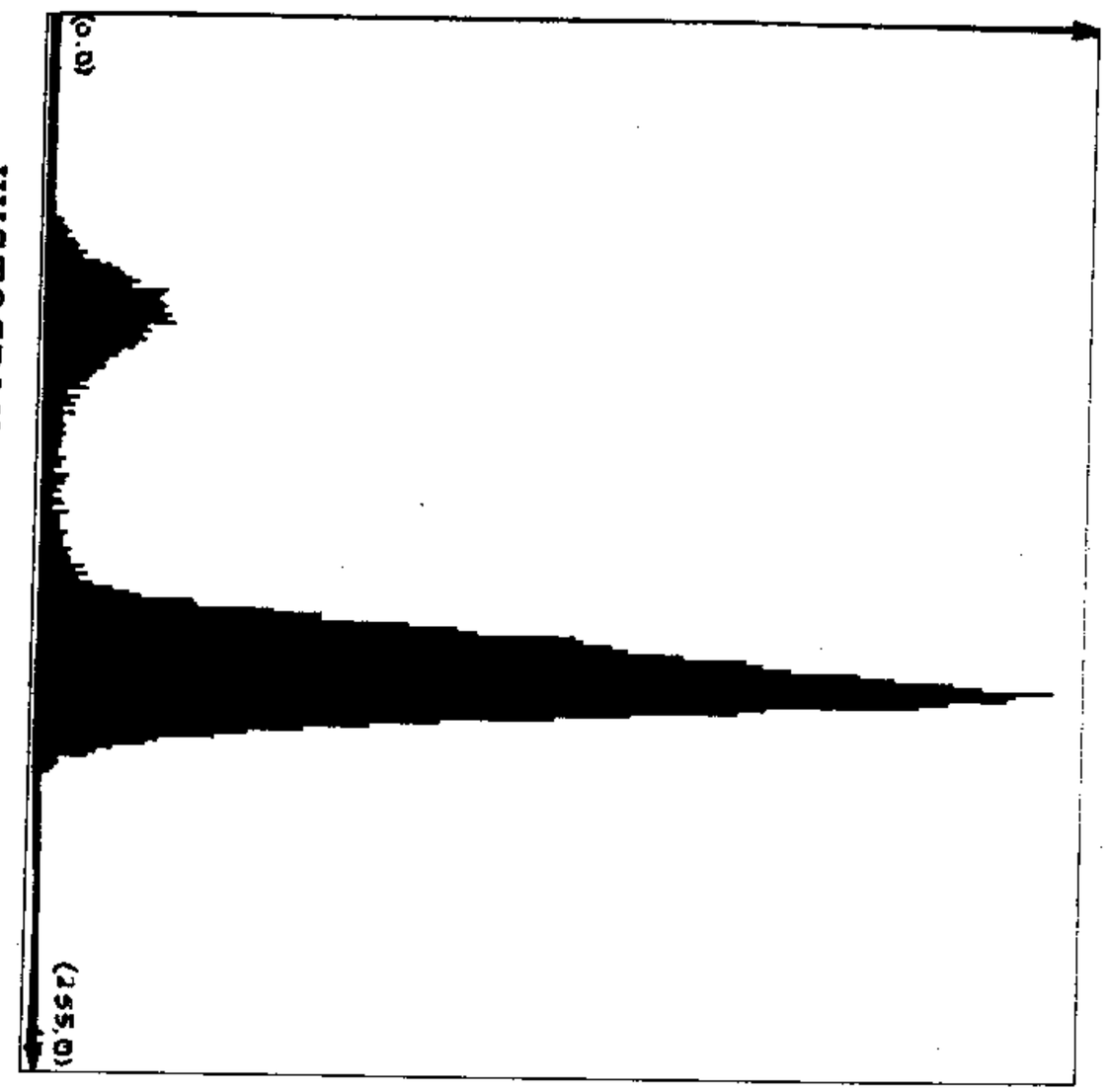
```
36  16          7          1 0 0 0
36  15          7          0 1 1 0
35  15          7          1 0 0 1
35  14          7          0 1 1 0
34  14          7          1 0 0 1
34  13          7          0 1 1 0
33  13          7          1 0 1 1
FINDPATH  ( 29 8 )
29  8           3          0 1 1 0
28  8           3          0 0 1 0
28  9           5          0 0 1 0
27  9           5          1 0 0 1
28  9           5          0 0 1 0
28  8           3          0 0 1 0
29  8           3          0 1 1 0
34  13          7          0 1 1 0
34  14          7          1 0 0 1
35  14          7          0 1 1 0
35  15          7          1 0 0 1
36  15          7          0 1 1 0
36  16          7          1 0 0 0
37  16          7          0 1 1 0
37  17          7          1 0 0 0
38  17          7          0 1 1 0
38  18          7          1 1 0 0
39  19          7          0 1 0 0
39  20          7          1 1 0 0
40  21          7          0 1 0 0
40  22          7          1 1 0 0
41  23          7          0 1 0 0
41  24          7          1 1 0 0
41  25          7          1 1 0 0
41  26          7          1 0 0 0
41  27          7          1 0 0 0
41  28          7          1 0 0 0
41  29          7          1 0 0 0
41  30          7          1 0 0 0
41  31          7          1 0 0 0
41  32          9          1 1 0 0
41  33          9          1 1 0 0
41  34          9          1 1 0 0
40  34          7          1 0 0 0
40  35          9          1 1 0 0
39  35          7          1 0 0 0
39  36          9          1 0 0 0
38  36          7          1 0 0 0
38  37          11         0 1 0 0
FINDPATH  ( 39 43 )
39  43          3          1 0 0 1
40  43          5          1 0 0 1
41  43          5          0 1 1 0
```

41	44	5	1 0 0 0
41	45	3	1 0 0 0
42	45	5	1 0 0 0
42	46	9	0 1 1 0
42	47	9	0 0 0 1
41	47	9	0 0 0 1
40	47	5	0 0 1 0
40	48	7	0 0 1 1
39	48	7	0 0 1 1
38	48	7	0 0 1 1
37	48	7	0 0 1 1
36	48	7	0 0 1 1
35	48	7	0 0 1 1
34	48	7	0 0 0 1
33	48	7	0 0 1 1
32	48	7	0 0 1 1
31	48	7	0 0 1 1
30	48	7	0 0 0 1
29	48	7	0 0 0 1
28	48	7	0 0 0 1
27	48	7	0 0 0 1
26	48	7	0 0 0 1
25	48	7	0 0 0 1
24	48	9	1 0 0 0
24	47	9	1 0 0 0
24	46	9	1 0 0 0
24	45	9	1 0 0 0
24	44	3	0 1 0 0
23	44	5	0 1 0 0
22	43	5	1 0 0 0
22	42	5	1 0 0 0
22	41	5	1 0 0 0
22	40	5	1 0 0 0
22	39	5	1 0 0 0
22	38	5	1 0 0 0
22	37	5	1 0 0 0
22	36	5	1 0 0 0
22	35	5	1 0 0 0
22	36	5	1 0 0 0
22	37	5	1 0 0 0
22	38	5	1 0 0 0
22	39	5	1 0 0 0
22	40	5	1 0 0 0
22	41	5	1 0 0 0
22	42	5	1 0 0 0
22	43	5	1 0 0 0
23	44	5	0 1 0 0
24	44	3	0 1 0 0
24	45	9	1 0 0 0
24	46	9	1 0 0 0
24	47	9	1 0 0 0

```
24  48          9          1 0 0 0
FINDPATH    ( 22 5 )
22  52          3          1 0 0 0
23  52          5          1 1 0 0
23  53          5          1 1 0 1
23  52          5          1 1 0 0
22  52          3          1 0 0 0
25  48          7          0 0 0 1
26  48          7          0 0 0 1
27  48          7          0 0 0 1
28  48          7          0 0 0 1
29  48          7          0 0 0 1
30  48          7          0 0 0 1
31  48          7          0 0 1 1
32  48          7          0 0 1 1
33  48          7          0 0 1 1
34  48          7          0 0 0 1
35  48          7          0 0 1 1
36  48          7          0 0 1 1
37  48          7          0 0 1 1
38  48          7          0 0 1 1
39  48          7          0 0 1 1
40  48          7          0 0 1 1
40  47          5          0 0 1 0
41  47          9          0 0 0 1
42  47          9          0 0 0 1
FINDPATH    ( 47 48 )
47  48          3          0 0 1 0
47  49          5          0 0 1 1
48  49          5          0 1 1 1
47  49          5          0 0 1 1
47  48          3          0 0 1 0
42  46          9          0 1 1 0
42  45          5          1 0 0 0
41  45          3          1 0 0 0
41  44          5          1 0 0 0
41  43          5          0 1 1 0
40  43          5          1 0 0 1
39  43          3          1 0 0 1
38  38          11         0 1 0 1
37  38          9          0 0 1 1
37  39          7          0 0 0 1
36  39          7          0 0 0 1
35  39          7          0 0 1 1
35  38          5          0 0 1 0
34  38          5          0 0 1 0
33  38          5          0 0 1 0
32  38          5          0 0 1 1
32  37          3          0 0 1 0
31  37          5          0 0 1 0
30  37          5          1 0 0 1
```

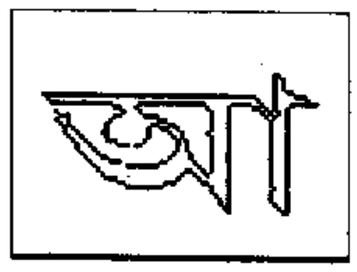
30	36	5	0 1 0 0
29	36	5	1 0 0 1
29	35	5	1 1 0 0
28	35	3	1 0 0 0
28	34	5	1 0 0 0
28	33	5	1 1 0 0
28	32	5	1 1 0 0
30	28	7	0 0 0 1
29	28	9	1 0 0 0
28	28	7	1 0 0 0
28	27	15	0 1 0 0
27	27	15	1 0 1 0
26	27	11	0 0 1 1
26	26	9	0 0 1 0
26	25	7	0 0 1 0
25	25	7	0 0 1 1
25	24	5	0 0 1 0
25	23	3	0 0 1 0
24	23	9	1 0 0 0
23	23	7	1 0 0 0
22	23	5	1 0 0 0
22	22	5	1 0 0 0
22	21	5	1 0 0 0
22	20	5	1 0 0 0
22	19	5	1 0 0 0
22	18	5	1 0 0 0
22	17	5	1 0 0 0
22	16	5	1 0 0 0
22	15	5	1 0 0 0
22	14	5	1 0 0 0
22	13	5	1 0 0 0
22	12	5	1 0 0 0
22	11	5	1 0 0 0
22	10	5	1 0 0 0
22	9	5	1 0 0 0
22	8	5	1 0 0 0
22	7	5	1 0 0 0
21	7	3	1 0 0 0

FIG. 1.1 GRAY LEVEL INPUT IMAGE



HYSTOGRAM OF FIG. 1.1

AA



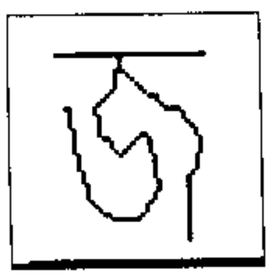
AAA

E

EEE

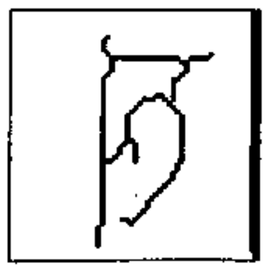
Fig.2

THRES JA

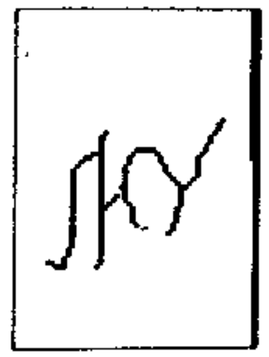


THIN JA

FIG 1.



AA



E