# Recognition of well-formed Handwritten Devnagari Characters

*by*

Ashutosh Kumar Jha

Under the supervision of

Prof. A. K. Dutta

# ACKNOWLEDGEMENT

I wish to express my deepest sense of gratitude to my guide Prof. A. K. Datta without whose valuable guidance and encouragement this paper could not have been completed. I will also like to thank Mr. S. Majumdar who helped me a lot in getting input data and Mr. S. K. Tripathy and Mr. Tarun Dan for helping me in starting the work. It is my pleasure to acknowledge the assistance of my friends Suresh, Kamble, Jaydeep, Krishnendu, and Rajan who have spent their time in interesting discussions and have given me enough encouragement throughout the dissertation work.

I also take this oppurtunity to thank the people in charge of computer and statistical service centre of Indian Statistical Institute, Calcutta, for their cooperation where all the algorithms have been implemented.

# CONTENTS

CHAPTER

# I. INTRODUCTION

The machine replication of human capabilities has been the subject of research for many decades. The machine having the same reading capabilities of human is one of such a cherished goal. Still there is a great gap between human reading and machine reading capabilities, and further effort is required to narrow down this gap, if not bridge it. The character recognition problem has attracted many research workers in the field of pattern recognition, also because of the possible commercial applications of character recognition techniques and the ease of availability of data for testing and implementing the various propositions in the general frame work of pattern recognition. Few of the various applications are as follows

-Aid for blind people:Such an aid uses photosensors and tactile simulator which converts picture information into sound[12,13]. Also used for reading and reproduction of Braille original[14].

-Use in postal department: For postal address reading and as a reader for handwritten and printed postal codes[15,16].

-Use in motor vehicle Bureau as automatic number plate reader and recorder for road traffic control[17].

-For automatic processing of documents. For automatic data entry from sheets. Use in publishing industry[45] as a multiprpose document reader for large scale data processing. As credit card reader[19].

-Use for examination assesment and as a marksheet reader.

-In information units and libraries[21].

## II. SCOPE OF DISSERTATION

The main objective is to design and implement an algorithm for machine recognition of well formed hand written Devnagari characters. A set of simple primitives is used, and all Devnagari characters are looked upon as a concatenation of these primitives. Most of the decisions are taken on the basis of the presence/absence or positional relationship or movement relationship of these primitives; and the decision process is a multistage process, where each stage of decision making narrows down the choice regarding the possible class membership of the input token.

## III. CHARACTER RECOGNITION : A BREIF REVIEW

Various technique have been proposed for character recognition, each having its own merits and demerits . The character recognition methodologies can be broadly classified on the basis of

1. approach.
2. the nature of application.
3. feature for recognition.

We give breif overview of each of these techniques.

*Scheme based on approch*

We have two main approaches to pattern recognition. They are Statistical/Decision-Theoretic and Syntactic/Grammatical approach.

The structural information about interconnection in complex patterns cannot be handled very well by Statistical pattern recognition technique. On the other hand the use of formal language theoretic models to represent patterns is the main drawback of the Syntactic approach. Patterns are natural entities which cannot strictly obey the mathematical constraints imposed by formal language theory. Imposing a strict rule on pattern structure is not particularly applicable to character recognition, where intra class variations are infinite.

To quote K.S.Fu[1] "*...The dichotomy of syntactic and decision-theoretic approches appears to be convenient only from the viewpoint of theoretical study . In other words , the* *division of two approches is sometimes not clearcut particularly in terms of practical applications " .*

So, a hybrid model is an interesting solution to practical character recognition problem [4,83].

The ultimate goal of 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, the algorithm should perform as well as human. This is the motivation for so called descriptive approach [4], which is now popular in character recognition approach. A descriptive approach can be provided easily with the flexiblities needed to take care of the infinite variations 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 acheive an efficient description, the feature used should be independent (i.e. Presence of new feature(s) or absence of old feature(s) should not affect the description of remaining featuers ). In our work we have incorporated this idea.

*Scheme based on nature of application*

On the basis of nature of application we can group the character recognition into two main schemes [2], namely on-line and off-line character recognition. In off-line the recognition is not done at the time of preparing the documents, whereas in on-line character recognition, the recognition is done as and when characters are hand drawn and hence timing information of each strokes are also available along with character image. On the basis of capablities and complexity we can further classify the off-line scheme as

(a)Fixed font character recognition which deals with recognition of a specific type writing font.

(b)Multifont character recognition which recognizes more than one font.

(c)Omni font for recognition of any font.

(d)Handwritten character recognition which deals with the recognition of unconnected normal handwritten characters.

(e)Script recognition which deals with recognition of unconstrained handwritten characters which may be cursive or connected .

*Scheme based on feature for recognition*

In terms of feature used, the character recognition can be broadly classified as

(a)Template matching and correlation technique .

(b)Feature analysis and matching technique .

(a)*Template matching and correlation technique*

This directly compares an input character to a standard set of prototypes used. The prototype that matches most closely provides recognition. The comparision technique can be as simple as one to one comparision or as complex as decision tree analysis in which only selected pixels are tested. This type of technique suffers from sensitivity to noise and is not adaptive to differences in writing style.

(b)*Feature analysis and matching technique*

These techniques are based on matching on feature planes or spaces which are distributed on 2-D plane. These are most frequently used techniques for character recognition. In these methods significant features are extracted from a character and compared to the feature description of the stored primitives. The closest matching provides recognition. Most of the previous methods are usually suitable for application to constrained domains [3]. Based on feature extraction techniques, the feature analysis techniques are grouped as

(I) Global transformation and series expansion .

(II) Feature derived from statistical distribution of points.

(III)Geometrical and topological feature .

*Global transformation and series expansion technique* helps to reduce dimensionality of the feature vector and provides feature invariant to some global distortion like translation and rotation. The extraction and mask making processes are easy for these features. However such feature extraction techniques demand high computational complexity. In this area researchers have used Fourier[23-26], Walsh[27,28], Hadamard[29] series expansion etc. and Hough transform[10,11,30], Projection transform[26], Chain-code transform[9] etc.

*Features derived from statistical distribution of points* includes moments[31], n-tupule[32], characteristic loci[33], crossing and distances[34-37]. These features are tolerant to distortion and take care of style variations to some extent. They provide high speed and low complexity for implementation. In general, mask making is difficult for these type of features.

*Geometrical and topological feature* analysis method is the most popular technique investigated by the researchers. The features may represent global and local properties of characters. These includes strokes and bays in various directions, end points, intersection of line segments, loops[4,38] and stroke relations, angular properties, sharp protrusions[39,40]. These features have high tolerances to distortion and style variations and also tolerate a certain degree of translation and rotation. They help to process character at high speed. However the extraction processes are in general very complex and it is difficult to generate masks for these type of features.

# IV. PROBLEM DEFINITION

## IV.1 *Characteristics of Devnagari script*

The Devnagari alphabet consists of 13 vowels and 36 consonants, making a total of 49 characters in all. In Devnagari, some of the characters are essentially extensions of some other characters with certain minor additions (For example character labelled 8 and 9,shown in appendix I).

In the Devnagari script, a vowel following a consonant is not written in the original form but is represented symbolicaly by adding certain details called "ligatures" to the consonant. For example the consonant `क' followed by vowel `इ' will be written as कि . Also there are graphems which correspond to a cluster of consonants and hence the number of patterns encountered in Devnagari script may be very large.

## IV.2 *Problems encountered in recognition*

There are some inherent problems in recognition of handwritten character. They are caused by the following well known factors[3]

a.) Error in reading hand written characters are caused by infinite variations of shapes resulting from the writing habit, style, education, region of origin, social environment, mood, health, as well on the writing instrument, writing surface, scanning methods etc.

b.)noise - which causes disconnected line segments, bumps, and gaps in lines, filled loops etc.

c.)style variation - that is, the use of different shapes to represent the same character as well as serif and other flourishes, slant etc.

d.)translation - movement of whole character or its components.

e.)rotation - change in orientation.

To overcome these problems we might need Statistical survey of handwritings and preprocessing of input pattern.

In off-line character recognition we have lack of stroke order information. This gives rise to the problem of getting starting point of segment to start the segment extraction process.

One of the problems is the possiblity of very large number of possible input patterns in Devnagari script. So, one possible solution to tackle it might be stepwise refinement of possible class membership of input patterns.

The possibility of infinite intra class variation of input pattern suggests that features selected to describe pattern and decision process should be immune to this.

With the above philosophy in mind we have developed a scheme for recognition of *well formed handwritten Devnagari characters*. Based on the structural peculiarities of Devnagari characters, the proposed recognition scheme is a multistage decision process, where each stage of decision narrows down the choice regarding the possible class membership of the input patterns.

A set of simple primitives is used and all the Devnagari characters are looked upon as a concatenation of these primitives. Most of the decisions are taken on the basis of the presence/absence or positional relationship or stroke relationship of these primitives.

In this work, we will limit our attention to the Devnagari alphabets only. We will not consider the recognition of those characters which is obtained by adding minor details to some of the other characters but this recourse is taken under the assumption that the minor additional details can easily be detected and recognized.

## V. PROBLEM FORMULATION

With the above philosophies in mind we, have developed a scheme for recognition of well formed hand-written characters. Due to limitation of the scope of the project we have restricted our attention to recognition of alphabets only. But it has been considered that work can be extended to other aspects of Devnagari character recognition also.

**Co-ordinate system used-:**Rectangular Cartesian co-ordinate system parallel to scan axis has been used as shown in the figure. Clockwise angle measured from x-axis is assumed to be positive. Angles are measured from $0^0$ to $360^0$ (as shown in fig. 1)

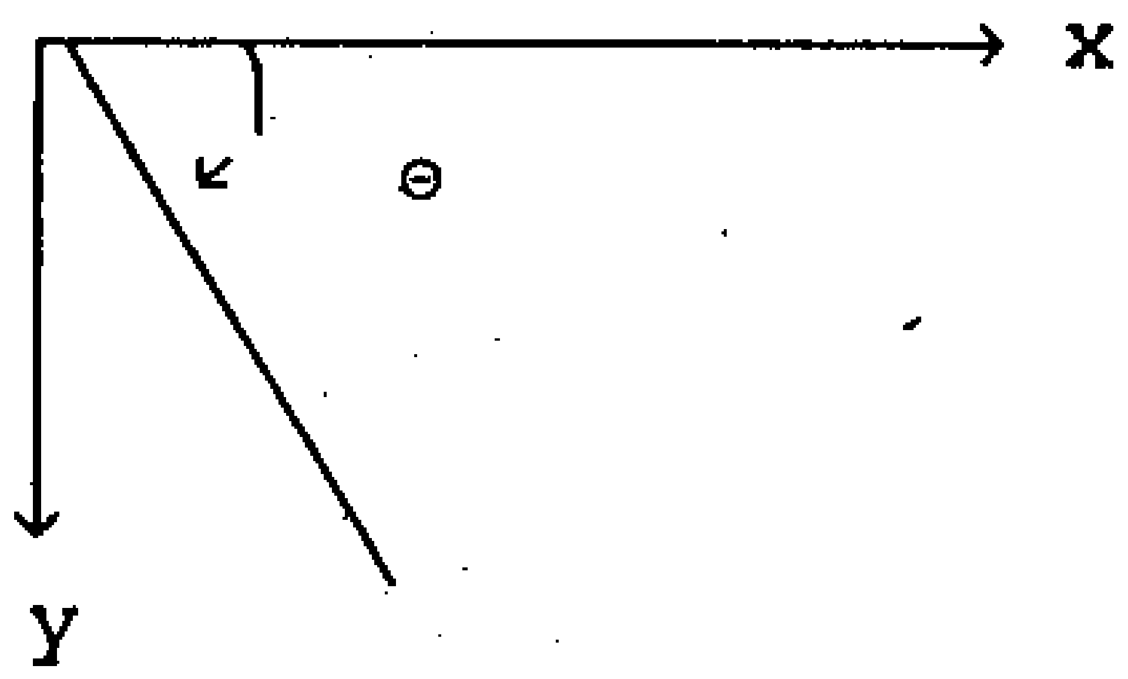


Fig.1

**Def.1**-The direction of a line segment is the average of angles that each of the consecutive points of theline segment make with the co-ordinate axis fixed at the origin.

**Eg.** The direction θ associated with the line segment L as shown in fig.2, with respect to origin (x,y) is given by

$$\theta = \frac{1}{n} \cdot \sum_{i=1}^{n} \arctan(y_i - y)/(x_i - x)$$

This averaging has a smoothing effect[7].

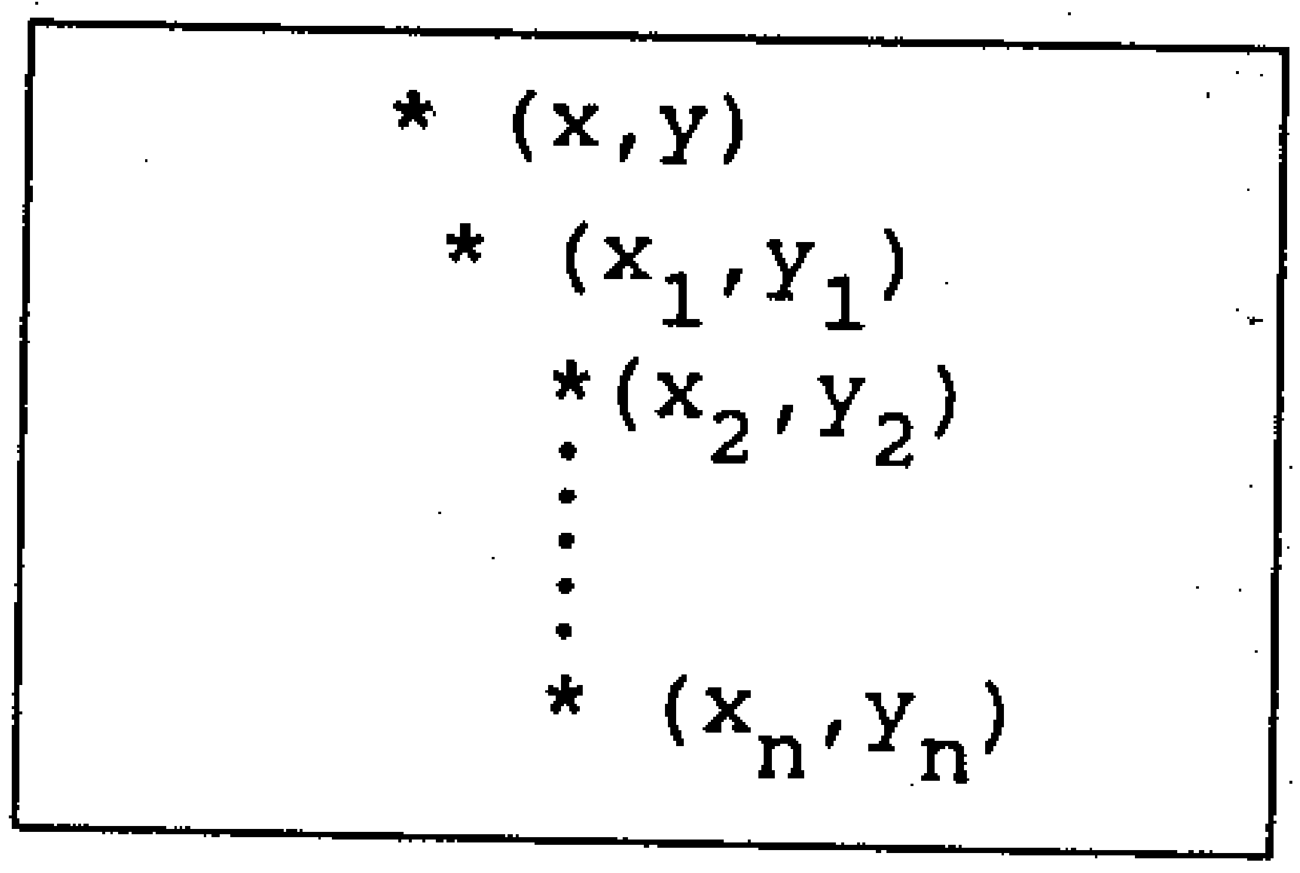


Fig.2 A line segment L.

We take the average of next five pixels in the line segment or the number of pixels available whichever is less.

**Def.2-** The directions associated with a pixel is the direction of all the line segments which is 8-connected to that and fixing the origin at that pixel.

**Eg,** The directions associated with pixel (x,y) is the direction associated with line segment $L_1$, $L_2$ and $L_3$ as shown in fig.3.

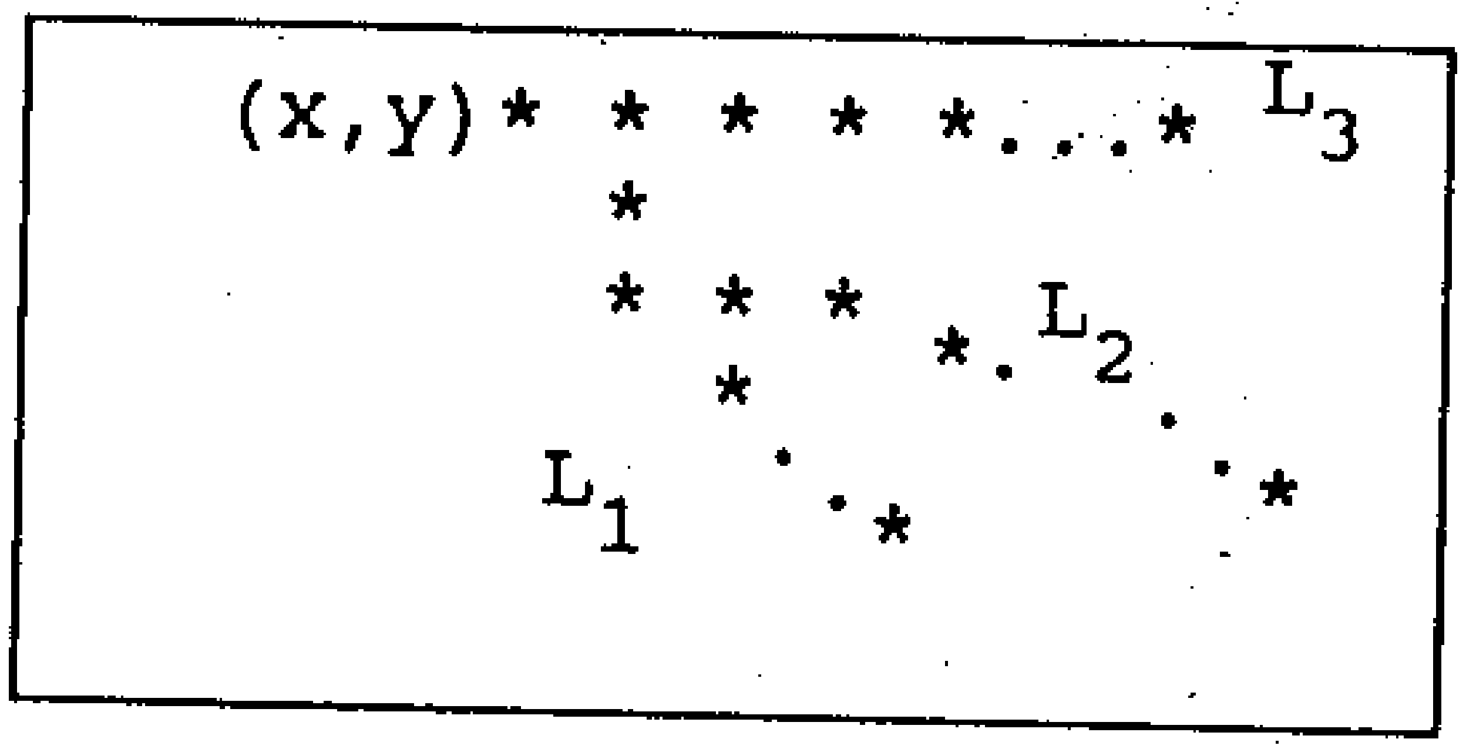


**Fig.3 Three directions associated with pixel (x,y).**

Input character is assumed to be a two tone image in the form of a binary matrix.

A curve segment $\overline{X_1X_2}$ is a directed line with a starting point $X_1$ and ending point $X_2$. A curve segment has curvature function f(l) along the directed line with $0<l<L$ , where L is the total length of the curve segment .

Curvature function $f(l)$ along the curve is defined as follows .

$$f(l)=\lim_{\partial l \to 0} \frac{1}{\partial l} * \left[\text{angle between the tangent lines to the curve at } l-\partial l \text{ and } l+\partial l\right]$$

$$\left.\begin{array}{l} \bar{C} = \overrightarrow{X_1X_2} \;\; ; \; L = \int_0^L \partial l \;\; ; \; A= \int_0^L f(l)\partial l \;\; ; \\ \qquad S = \int_0^L \left( \int_0^S f(l)-A/2\right)\partial s \;\; ; \end{array}\right] \rightarrow \text{EQN. 1}$$

A curve segment is said to be clockwise if $f(l) > 0$ and anticlockwise if $f(l) < 0$ $\quad 0<l<L$.

It has been pointed out[7] that the four attributes $\bar{C}$, L, A, S (as defined above ) are sufficient for shape recognition.

The parameters are modified to make it insensitive to size, to some extent . S gives information about the declination of a curve, which we know at the time of segmentation. Other parameters evaluated are as follows :

(1) Arc length to chord length ratio (ATCR)= $L/|\bar{C}|$ ;

(2) Maximum perpendicular distance of curvature points from $\bar{C}$ to chord length ratio($R_m$)= $p_m/|\bar{C}|$ ;

$P_m$ is the maximum perpendicular distance of curvature points from $\bar{C}$.

(3) A (Total angle of bending, as described in eqn. 1 ).

Depending upon the values of ATCR ,$R_m$, and A and the threshold for the segments the segment is classified in one of the five classes . For example if ATCR $\approx$ 1, $R_m \approx 0$, A $\approx$ 0 means that the segment is a straight line.

The given set of Devnagari characters can be thought of as composed from a set of basic building elements or primitives. Based on the structural peculiarities of Devnagari characters, six primitives are selected as described below. These are (a) Global vertical line (b) Loop (c) Half loop (d) Deep arc (e) Arc (f) Straight line. (Note: If the end points marker is same which is given at the time when segments are taken out then that segment is also called a loop.)

Based upon the observation of writing style and the segmentation process the possible variations allowed in the shape of primitives building the character , is described in Appendix II.

The first problem is to locate the region in which the character lies. This is done by first locating an empty row in the input matrix with minimum y-value and then increasing the y-value of row until we get a non empty row (1ST in fig.4). This way we get the upper end of the character. After this, again y-value of row is increased until we get an empty row. This way we get the lower end of the character (2ND in fig.4).

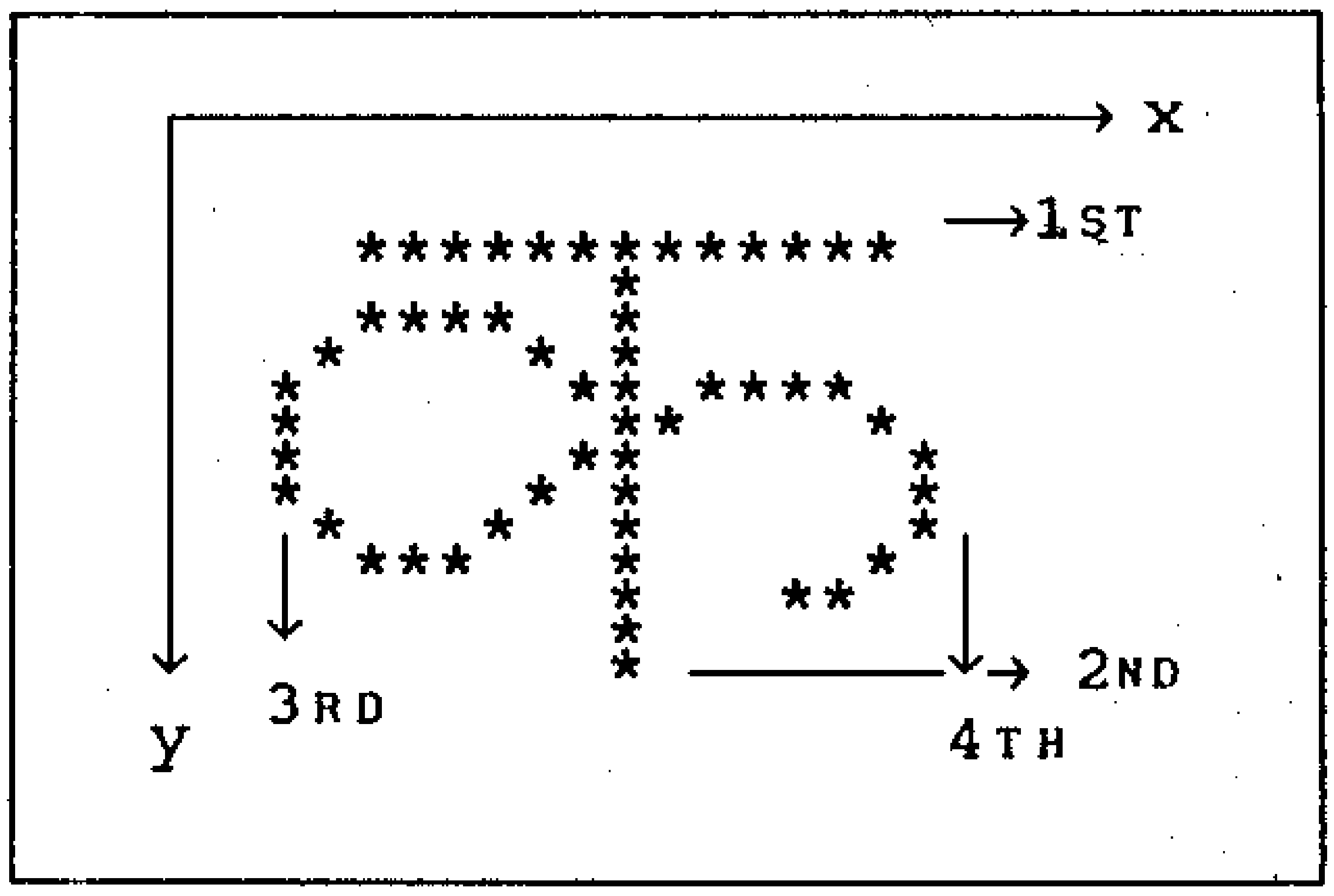


Fig.4 Pictorial view of finding minimum orthogonal rectangle containing a character.

To fix the left end and right end of a character, an empty column with minimum x-value within the fixed upper and lower limit is found. Then the x-value of column to be scanned is increased until we get a non empty column (3rd in fig.4). This gives the left end of the character. Again the x-value of the column to be scanned is increased until we get an empty column. This way we get the right end of the character (4th in fig.4). So, we have the minimum right rectangle containing the character which is called a frame.

Input data has been taken from a digitizer tablet. We want a line segment to be single pixel thick. In general the data received is good but in some cases the line segments are found to be two pixels thick. Based upon observation of input data we remove such pixels. For instance, if the input pixel configuration is as in fig 5(a) and the neighbouring pixel configuration is as in fig 5(b), then the output configuration would be as in fig 5(c). For other two neighbouring configurations (d) and (f), the output pixel configuration will be given by (e) and (g) respectively.

Eg. CORE PIXEL , CONFIGURATION NEIGHBOURING PIXEL CONFIGURATION OUTPUT PIXEL , CONFIGURATION

| | | | |
|---|---|---|---|
| | | | |
| | 0 | 1 | |
| | 1 | 1 | |
| | | | |

(a)

| | | | |
|---|---|---|---|
| | 0 | 0 | 0 |
| | 0 | 1 | 0 |
| | 1 | 1 | |
| | | | |

(b)

| | | | |
|---|---|---|---|
| | | | |
| | 0 | 0 | |
| | 1 | 1 | |
| | | | |

(c)

| | | | |
|---|---|---|---|
| | | | |
| | 0 | 1 | |
| | 1 | 1 | 0 |
| | | 0 | 0 |

(d)

| | | | |
|---|---|---|---|
| | | | |
| | 0 | 1 | |
| | 1 | 0 | |
| | | | |

(e)

| | | | |
|---|---|---|---|
| | | | |
| 0 | 0 | 1 | |
| 0 | 1 | 1 | |
| 0 | 0 | | |

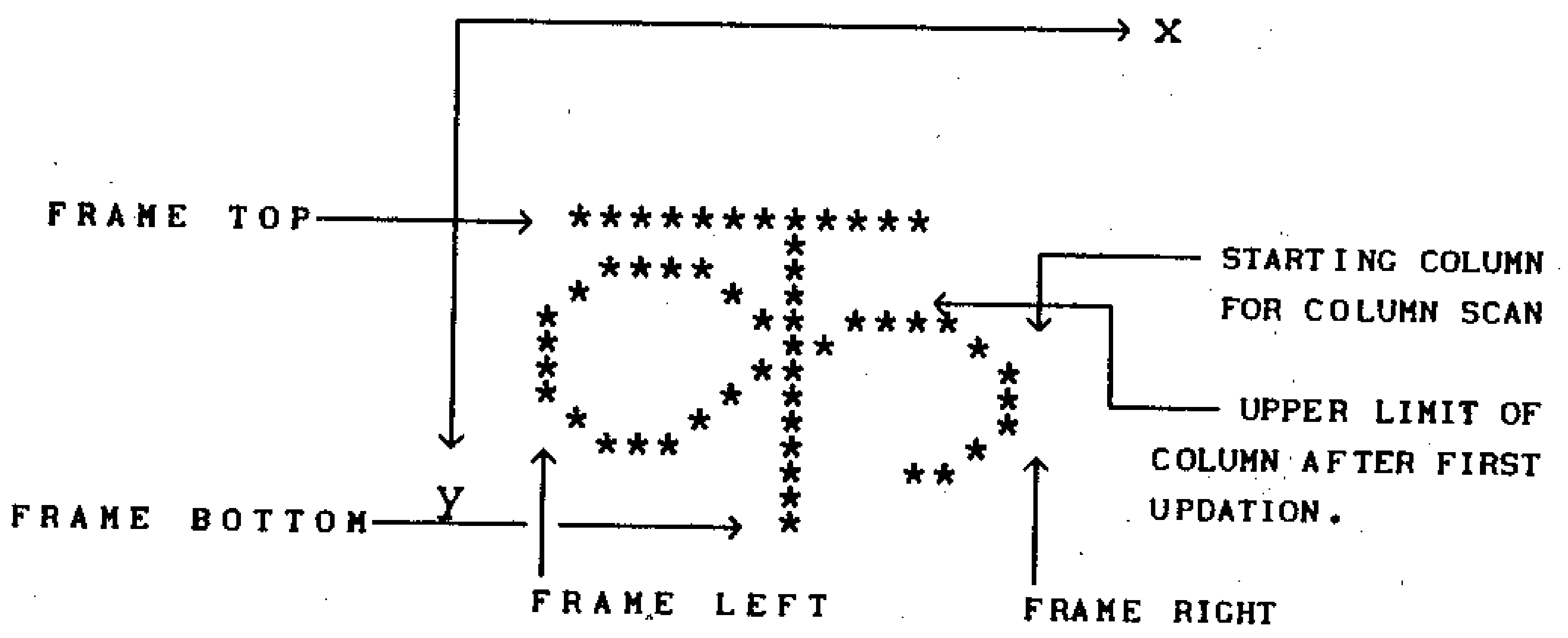
(f)

| | | | |
|---|---|---|---|
| | | | |
| | 0 | 1 | |
| | 0 | 1 | |
| | | | |

(g)

Fig.5 Example of relationship between input and preprocessor output.

After preprocessing, the data is ready for segmentation. Before starting the segmentation process, we have the problem of getting the starting pixel. That is, from where the segmentation process will start. To overcome this problem we observed that we can get the right end pixel of the canopy in the upper half of the frame. For this we start scanning the column in the upper half of the frame from rightmost end and decreasing the x-value of the column to be scanned until we get an image pixel (fig.6). The column is scanned from minimum y-value. As soon as we get the image pixel we find the direction associated with that pixel. If one of the directions of the pixel is within the threshold of



X
FRAME TOP
FRAME BOTTOM
Y
FRAME LEFT
FRAME RIGHT
STARTING COLUMN FOR COLUMN SCAN
UPPER LIMIT OF COLUMN AFTER FIRST UPDATION.

Fig.6 Finding rightmost end of matra.

the direction of the canopy,then that image pixel is considered to be the rightmost starting pixel of the canopy and the direction is said to be the starting direction of canopy. After this we follow the line segment emanating from that image pixel as long as the line segment moves upward. That is the ordinate value of pixel of line segment gets reduced or we hit a junction. We update the upper limit of column to be scanned to that ordinate value. This in turn reduces the height of column to be scanned for finding the rightmost end of the canopy. These operations of finding image pixel, getting the direction associated with that pixel, checking the possibility of its being the rightmost end of the canopy and updation of bound continues until the upper bound of column goes beyond the frame boundary. In general this process is repeated twice.

After getting the rightmost end of canopy and starting direction associated with this we remove the pixels of canopy and all 3*3 neighbours are inserted in the queue. The rightmost pixel we get is called current pixel. For removing the pixels of the canopy we remove the current pixel and find the directions associated with 1*3 neighbour (i.e $p_1$, $p_2$,$p_3$ as shown in fig.7) where p is current pixel.

|  |  |  |
|---|---|---|
| $p_1$ | $p_8$ | $p_7$ |
| $p_2$ | $p$ | $p_6$ |
| $p_3$ | $p_4$ | $p_5$ |

**Fig.7 eight neighbourhood of pixel p.**

The pixel which has got the best matching direction within the threshold of the starting direction is taken as the successor of the current pixel and the starting direction is updated to include the effect of the selected direction of the successor pixel. Others 1*3 neighbours are inserted in queue. If $p_1$is selected as successor then $p_9$ and $p_{10}$ (as shown in fig.8) is checked and if $p_3$ is selected as successor then $p_{11}$ and $p_{12}$is checked. If they are not inserted in queue then after giving a marker corresponding to the segment they are inserted in queue. The marker indicates that which segment has inserted it.

|  |  |  |
|---|---|---|
| $p_9$ | $p_{10}$ |  |
| $p_1$ | $p_8$ | $p_7$ |
| $p_2$ | $p$ | $p_6$ |
| $p_3$ | $p_4$ | $p_5$ |
| $p_{11}$ | $p_{12}$ |  |

Fig.8 3*4 neighbourhood.

Removal of pixels of canopy continues until there is no successor of current pixel.

We observed that canopy is not a discriminatory feature and so we simply removed the pixels of the canopy. After the removal of canopy, the end pixel of segments connected to the removed canopy which is in queue, gives the starting pixel of segment.

Now we start segmentation process. We remove the pixels from queue and check that pixel is still present in image or not until we get a pixel which is still present in image. This pixel is called current pixel. Then, the direction associated with current pixel, which is closest to one of the four major axes is selected as segment direction of current segment. The current pixel is removed from pattern and is stored as a pixel of current segment. 3*3 neighbours of current pixel is checked and the pixel which has one of the best matching direction within the threshold to segment direction is selected as successor of current pixel. Segment direction is updated to include effect of new direction also. In finding direction associated with neighbour pixel the current pixel serves the purpose of origin. If there is no direction associated with any of its neighbour within the threshold of segment direction then the successor is selected as follows.

If possible, the pixel which has maximum angle measured in clockwise direction from segment direction but less than $180^0$ is selected as successor of current pixel and the segment direction is updated to be that direction.

Otherwise the pixel which has minimum angle measured in anticlockwise direction from segment direction and is less than $180^0$is selected as successor pixel and segment direction is updated to be that direction. When we select the successor pixel based on movement (clockwise/anticlockwise), we note down the pixel which has been selected as successor, the count of number of times such a successor is selected, the segment direction at that time, the new segment direction after updation.

Once we have selected the successor based on movement then the pixel which has one of direction within the threshold of segment direction or which has similar movement direction, is a possible member for successor of current pixel.

Other 3*3 members are inserted in queue after giving a marker which indicates that which segment has inserted it. When we are inserting the pixel in queue we also insert the following informations :

-number of 4-connected pixels outputed so far.

-number of 8-connected pixels outputed so far.

-segment direction at that time and total angle of bending.

-movement count at that time.

This process of selecting the successor of the current pixel and other operations following it are carried out until there is no pixel which can be a successor of the current pixel. The pixels removed so far constitute a segment.

If at the end of segmentation process it is found that the current pixel just removed has the same marker as is given for the current segment, it indicates that a loop has been encountered. To seperate the loop part from the current segment the queue is checked for recently removed current pixel and depending upon the other informations available with that pixel, the pixels constituting the loop part is seperated, which forms a new segment.

When we are extracting the segment the following informations are extracted simultaneously :

(1) No. of 8-connected pixels in segment .

(2) No. of 4-connected pixels in segment.

(3) Connectivity to other segments.

(4) Pixel at which it is connected to other segment.

(5) Movement type (Clockwise/Anticlockwise/no movement ).

(6) Curvatures at the pixels where successor is selected based on the direction of movement.

(7) Pixel, which has been selected as the successor of current pixel based on the direction of movement.

(8) Starting direction of segment.

(9) Direction at which it has been inserted in queue.

(10)Whether segment is a loop ?

(11)Total angle of bending.

Example of segmentation is shown in Fig.9, where

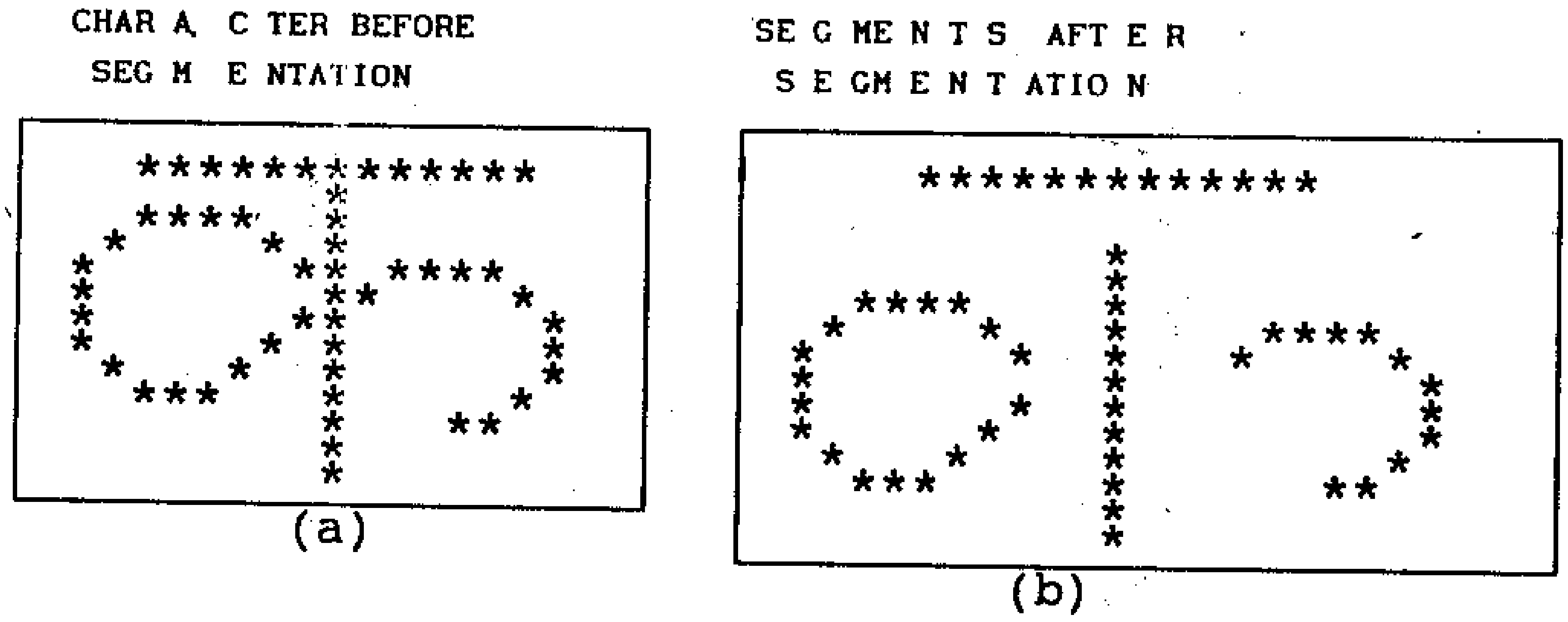


Fig.9 Segmentation of character (a) an input character, (b) different segments of the character.

(a) corresponds to input character and (b) corresponds to the four different segments (global vertical line, loop, half loop and canopy) of the character.

As mentioned in eqn.1, ATCR, $R_m$, and A are calculated and the segment is classified into one of five classes.

All these operations are carried out until queue is empty. After getting the classification of all possible segments; decision process is started to classify the input pattern in one of the possible classes.

The decision process is nothing but a process in which at every step of decision the possible class membership of input pattern gets reduced and finally the character is recognized . The following are the tests used by the decision process.

a : Global vertical line is present or not.

b : Loop connected to canopy.

c : Loop connected to global vertical line.

d : Number of segment connected to canopy.

e : Segment connected to global vertical line having anticlockwise angle difference less than 180°.

f : Number of loops present in character.

g : Stroke matching.

h : Number of segments connected to global vertical line.

i : Segment type and starting direction.

j : Based on connectivity between segments and their types.

k : Movement type present.

l : Movement and segment type;

m : Movement connected to segment with starting direction.

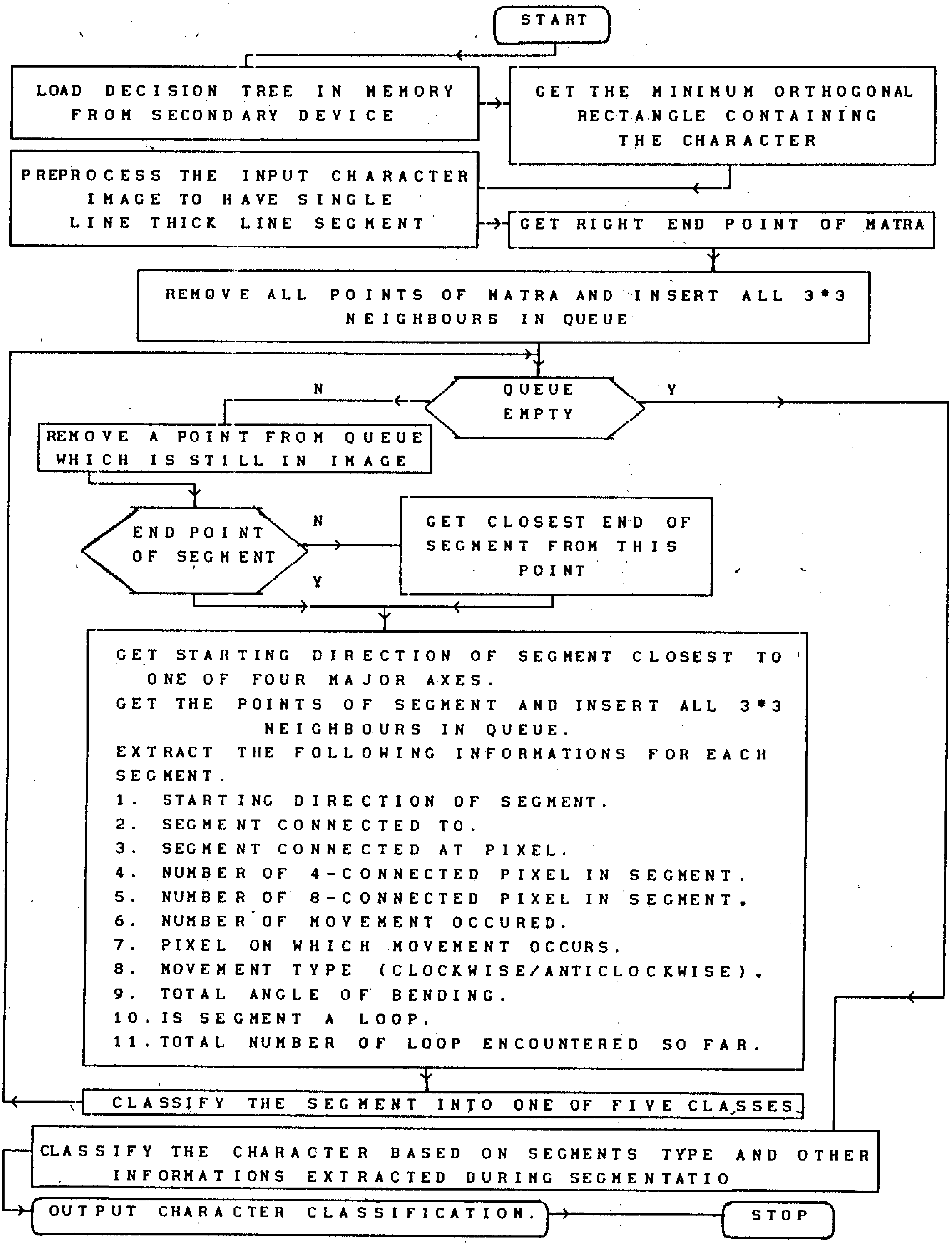
n : Number of segment present in character.

o : Segment type present.

Decision process uses a decision tree (created earlier) which is a binary tree in which each node has a key which specifies the test to be conducted. Character is classified at leaf of decision tree (Decision tree is given in appendix VI). The decision tree has 83 nodes, a maximum depth of 16 and maximum number of nodes at any level is 13.

Input characters before preprocessing and after preprocessing are given in appendix III and IV. Examples of segmentation are given in appendix V. Decision tree is given in appendix VI.

# CONTROL FLOW DIAGRAM OF CHARACTER RECOGNITION PROCESS


START
LOAD DECISION TREE IN MEMORY FROM SECONDARY DEVICE
GET THE MINIMUM ORTHOGONAL RECTANGLE CONTAINING THE CHARACTER
PREPROCESS THE INPUT CHARACTER IMAGE TO HAVE SINGLE LINE THICK LINE SEGMENT
GET RIGHT END POINT OF MATRA
REMOVE ALL POINTS OF MATRA AND INSERT ALL 3*3 NEIGHBOURS IN QUEUE
QUEUE EMPTY
N
Y
REMOVE A POINT FROM QUEUE WHICH IS STILL IN IMAGE
END POINT OF SEGMENT
N
Y
GET CLOSEST END OF SEGMENT FROM THIS POINT
GET STARTING DIRECTION OF SEGMENT CLOSEST TO ONE OF FOUR MAJOR AXES.
GET THE POINTS OF SEGMENT AND INSERT ALL 3*3 NEIGHBOURS IN QUEUE.
EXTRACT THE FOLLOWING INFORMATIONS FOR EACH SEGMENT.
1. STARTING DIRECTION OF SEGMENT.
2. SEGMENT CONNECTED TO.
3. SEGMENT CONNECTED AT PIXEL.
4. NUMBER OF 4-CONNECTED PIXEL IN SEGMENT.
5. NUMBER OF 8-CONNECTED PIXEL IN SEGMENT.
6. NUMBER OF MOVEMENT OCCURED.
7. PIXEL ON WHICH MOVEMENT OCCURS.
8. MOVEMENT TYPE (CLOCKWISE/ANTICLOCKWISE).
9. TOTAL ANGLE OF BENDING.
10. IS SEGMENT A LOOP.
11. TOTAL NUMBER OF LOOP ENCOUNTERED SO FAR.
CLASSIFY THE SEGMENT INTO ONE OF FIVE CLASSES
CLASSIFY THE CHARACTER BASED ON SEGMENTS TYPE AND OTHER INFORMATIONS EXTRACTED DURING SEGMENTATIO
OUTPUT CHARACTER CLASSIFICATION.
STOP

# VI. ALGORITHMS

## Get–the–minimum–orthogonal–rectangle–containing–the–character

```
Input: A character in matrix.
output:Minimum orthogonal rectangle containing the character.
Initializaton : y-min = ordinate of an empty row with minimum
                        ordinate.
Step1: Increse y-min until a non empty row is found.
       Set the top-of-frame to y-min.
Step2: Increse y-min until an empty row is found.
       Set bottom-of-frame to y-min.
Step3: x-min = Abscissa of an empty column between top of frame
               and bottom  of frame with minimum abscissa.
       Increse x-min until a non empty column is found.
       Set the left-of-frame to x-min.
Step4: Increse x-min until an empty column is found.
       Set right-of-frame to x-min.
Step5: Stop.
```

## preprocess–the–input–character

```
Input  :minimum orthogonal rectangle containing the character
        called image_matrix.
Output:preprocessed input character.
        start scanning the rectangle from left to right and from
        top to bottom until all the pixels of input rectangle is
        visited.
```

Let us assume that we are examiniing row i and i+1 and column j and j+1. If input pixel configuration in 2*2 window is as in column I and neighbouring pixel configuration in 4*4 window is as in column II then column III gives the corresponding output configuration.

column I

**CORE PIXEL CONFIGURATION**

|   | j | j+1 |   |   |
|---|---|---|---|---|
|   |   |   |   |   |
|   | 0 | 1 |   | i |
|   | 1 | 1 |   | i +1 |
|   |   |   |   |   |

column II

**NEIGHBOURING PIXEL CONFIGURATION**

|   | j | j+1 |   |   |
|---|---|---|---|---|
|   | 0 | 0 | 0 |   |
|   | 0 | 1 | 0 | i |
|   | 1 | 1 |   | i +1 |
|   |   |   |   |   |

|   |   |   |   |
|---|---|---|---|
|   |   |   |   |
|   | 0 | 1 |   |
|   | 1 | 1 | 0 |
|   |   | 0 | 0 |

|   |   |   |   |
|---|---|---|---|
|   |   |   |   |
| 0 | 0 | 1 |   |
| 0 | 1 | 1 |   |
| 0 | 0 |   |   |

column III

**OUTPUT PIXEL CONFIGURATION**

|   | j | j+1 |   |   |
|---|---|---|---|---|
|   |   |   |   |   |
|   | 0 | 0 |   | i |
|   | 1 | 1 |   | i +1 |
|   |   |   |   |   |

|   |   |   |   |
|---|---|---|---|
|   |   |   |   |
|   | 0 | 1 |   |
|   | 1 | 0 |   |
|   |   |   |   |

|   |   |   |   |
|---|---|---|---|
|   |   |   |   |
|   | 0 | 1 |   |
|   | 0 | 1 |   |
|   |   |   |   |

continue->

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 0 | |
| | 1 | 1 | |
| | | | |

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | |
| 0 | 1 | 0 | |
| | 1 | 1 | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | 0 | 0 | |
| | 1 | 1 | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| 0 | 1 | 0 | |
| 0 | 1 | 1 | |
| 0 | 0 | | |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 0 | |
| | 0 | 1 | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 0 | 0 |
| | 1 | 1 | 0 |
| | | 0 | 0 |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 0 | |
| | 1 | 0 | |
| | | | |

continue->

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 1 | |
| | 0 | 1 | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 1 | |
| | 0 | 1 | 0 |
| | 0 | 0 | 0 |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 1 | |
| | 0 | 0 | |
| | | | |

| | | | |
|---|---|---|---|
| | | 0 | 0 |
| | 1 | 1 | 0 |
| | 0 | 1 | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 0 | |
| | 0 | 1 | |
| | | | |

| | | | |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | 1 | |
| 0 | 0 | 1 | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | 0 | 1 | |
| | 0 | 1 | |
| | | | |

continue->

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 1 | |
| | 1 | 0 | |
| | | | |

| | | | |
|---|---|---|---|
| 0 | 0 | | |
| 0 | 1 | 1 | |
| | 1 | 0 | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | 0 | 1 | |
| | 1 | 0 | |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 1 | |
| 0 | 1 | 0 | |
| 0 | 0 | | |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 1 | |
| | 0 | 0 | |
| | | | |

| | | | |
|---|---|---|---|
| | | 0 | 0 |
| | 1 | 1 | 0 |
| | 1 | 0 | 0 |
| | | | |

| | | | |
|---|---|---|---|
| | | | |
| | 1 | 0 | |
| | 1 | 0 | |
| | | | |

```
If all the pixels in 2*2 window is one then
 k = i+1;
 while(image_matrix[k+1][j]=1 and image_matrix[k+1][j+1]=1)
        do k=k+1 end_while;
 deletable_column = j;
 if(image_matrix[k+1][j+1]=0 or image_matrix[i-1][j+1]=0)
             deletable_column =j+1 end_if;
 for m=i+1 to k-1 do
    if(deletable_column=j+1)
     if(image_matrix[m][j+2]=0) then image_matrix[m][j+1]=0;
      else
       if(image_matrix[m][j-1]=0) then image_matrix[m][j]=0;
     end_if;
    else
```

```
     if(image_matrix[m][j-1]=0) then image_matrix[m][j]=0;
      else
       if(image_matrix[m][j+2]=0) then image_matrix[m][j+1]=0;
     end_if;
    end_if;
    if(k>j+1)
      if(image_matrix[k][j-1]=0 and image_matrix[k+1][j]=0 and
         image_matrix[k+1][j-1]=0) then image_matrix[k][j]=0;
      else
      if(image_matrix[k][j+2]=0 and image_matrix[k+1][j+1]=0
       and image_matrix[k+1][j+2]=0)then image_matrix[k][j+1]=0;
    end-if;
k = j+1;
while(image_matrix[i][k+1]=1 and image_matrix[i+1][k+1]=1)
      do k=k+1 end_while;
deletable_row = i;
if(image_matrix[i+1][k+1]=0 or image_matrix[i+1][j-1]=0)
            deletable_row =i+1 end_if;
for m=i+1 to k-1 do
   if(deletable_row=i+1)
    if(image_matrix[i+2][m]=0) then image_matrix[i+1][m]=0;
     else
      if(image_matrix[i-1][m]=0) then image_matrix[i][m]=0;
     end_if;
   else
```

```
       if(image_matrix[i-1][m]=0) then image_matrix[i][m]=0;
        else
         if(image_matrix[i+2][m]=0) then image_matrix[i+1][m]=0;
       end_if;
     end_if;
     if(k>j+1)
      if(image_matrix[i][k+1]=0 and image_matrix[i-1][k+1]=0 and
         image_matrix[i-1][k]=0) then image_matrix[i][k]=0;
       else
        if(image_matrix[i+2][k]=0 and image_matrix[i+2][k+1]=0
        and image_matrix[i+1][k+1]=0)then image_matrix[i+1][k]=0;
      end_if
     if(image_matrix[i-1][j-1]=0 and image_matrix[i-1][j]=0 and
        image_matrix[i][j-1]=0) then image_matrix[i][j]=0;
      else
       if(image_matrix[i-1][j+1]=0 and image_matrix[i-1][j+2]=0
         and image_matrix[i][j+2]=0) then image_matrix[i][j+1]=0;
     end_if
     if(image_matrix[i+2][j-1]=0 and image_matrix[i+2][j]=0 and
        image_matrix[i+1][j-1]=0) then image_matrix[i+1][j]=0;
      else
       if(image_matrix[i+2][j+1]=0 and image_matrix[i+2][j+2]=0
       and image_matrix[i+1][j+2]=0) then image_matrix[i][j+1]=0;
     end_if
stop.
```

**get-right-end-pixel-of-canopy**

**Input** :Co-ordinate of minimum rectangle containing the character.

**Output**:Rightmost end point and direction of canopy.

**Initialization:**

lower-limit = top-of-frame.

upper-limit =(top-of-of-frame+bottom-of-frame)/2

**Step1:** Start scanning the column from lower-limit to upper-limit from rightmost end of frame towards left. If an image pixel is in column then find the direction associated with that pixel. If one of direction is within the threshold of direction of canopy and there is no 8-connected pixel to the right of it then set rightmost-end-pixel-of-canopy to the pixel and starting-direction-of-canopy to the direction.

**Step2:** (Updation of upper-limit) Move along the line segment emanating from that image pixel upward until a junction is found or there is no such pixel that upward movement is possible. Set the upper-limit to ordinate of current pixel.

**Step3:** Repeat step1 and step2 until upper-limit is less than lower limit.

**Step4:**output

rightmost-end-pixel-of-canopy and starting-direction-of-canopy.

**stop.**

**Remove-pixels-of-canopy**

**Input** :Rightmost end pixel and starting direction of canopy.

**Output**:Pixels of canopy.

**Initialization:**

```
  current-pixel     = rightmost-end-pixel-of-canopy.
  canopy-direction = starting-direction-of-canopy
```

**Step1:** Remove current-pixel.

**Step2:** (Select successor of current pixel)

Find 1*3 neighbours ($p_1$,$p_2$,$p_3$) (as shown in fig.8,p is current-pixel). Select the successor of current-pixel to be that neighbour which has best matching direction (def.2) within the threshold of canopy-direction. Update the canopy direction to include effect of direction of successor pixel.

**Step3:** Insert other 1*3 neighbours in queue after giving a marker which indicates that which segment has inserted the pixel in queue. If $p_1$ is selected as successor then $p_9$ and $p_{10}$ are checked and if $p_3$ is selected as successor then $p_{11}$and $p_{12}$ are checked. If these pixels are not already in queue then these pixels are inserted in queue after giving marker.

**Step4:** Repeat step1,step2 and step3 until there is no successor of canopy.

**Step5:** Stop.

**get–segments**

**Input** :Starting pixel of a segment,and starting direction.

**Output**:Points of segment,total angle of bending, total no of 4-connected pixels, total no of 8-connected pixels, curvatures at the pixels of bending, connected to.

**Initialize:**

current-pixel = starting pixel of segment.

segment-direction = starting direction of segment.

movement-count =0; total-angle-of-bending=0;

no-of-4-connected-pixel=0;

no-of-8-connected-pixel=0;

**Step1**:If the current-pixel has some marker which has been given by some other segment then note connecteivity to that segment. Output the current-pixel.

**Step2**:(find successor of current pixel)

Find directions (def.2) of all 3*3 neighbours of current pixel. Select the successor of the current-pixel to be that pixel which has a best matching direction within the threshold of the segment-direction. segment-direction is updated to include the effect of direction of successor. If there is no such neighbour pixel then select successor of current-pixel as follows:

If possible, select successor of current-pixel to be that pixel which has maximum clockwise direction w.r.t. segment-direction but less than $180^0$. The segment-direction is updated to be that direction. Otherwise, the pixel which has minimum anticlockwise direction w.r.t. segment-direction but less than $180^0$ is selected to be successor and segment-direction is updated to be that direction. Once the successor of current-pixel is selected based on maximum clockwise movement direction or minimum anticlockwise movement direction, then the pixel which has one of direction within the threshold of segment-direction or which has similar movement direction w.r.t. segment-direction is possible member for successor. If successor is selected based on movement then

```
    note down the curvature at that pixel;
    add present angle of bending to total-angle-of-bending;
    increase movement-count by 1;
    note down the pixel selected as successor;
```

If more than one pixel is within the threshold of segment-direction then give precedence to that pixel which has matching direction for more pixels.Except the successor-pixel all other 3*3 neighbours are inserted in queue with following informations

```
        -segment-direction at that time before updation.
        -movement-count.
        -no-of-4-connected-pixel and no-of-8-connected-pixel.
        -total-angle-of-bending.
 If successor is 4-connected pixel to the current-pixel then
      increase the no-of-4-connected-pixel by one.
Increase the no-of-8-connected-pixel by one;
```

**Step3:**Repeat step1 and step2 until there is no successor of current-pixel.

**Step4:**If current pixel just outputed has same marker as given for current segment then call seperate-loop.

**Step5:**Stop.

Threshold of deviation from segment-direction is taken $30^{0}$.

**classify-segment-in-primitive-class**

**Input :** starting pixel of segment, ending pixel of segment, no of 4-connected pixel, no of 8-connected pixel, pixels of segment, pixels where curvature has been calculated,
total angle of bending.

**Output:**primitive class of segment.

chord-length =end to end distance between the starting pixel and ending pixel of segment.

arc-length=(no-of-8-connected-pixel - no-of-4-connected-pixel)* 1.1414 + no-of-4-connected-pixel.

$\overline{C}$ =vector from starting pixel of segment to end pixel of segment p is perpendicular distance of segment pixel from $\overline{C}$. If two segment pixels are in perpendicularly opposite side of $\overline{C}$ then p is sum of perpendicular distance of those pixels from $\overline{C}$.

$p_m$= maximum(p);

We calculate

ATCR = arc-length/chord-length ;

$R_m = p_m / |\overline{C}|$ ;

A is total-angle-of-bending ;

```
if (ATCR > Thhl1 and $R_m$ > Thhl2 and and A >  Thhl3  )
     then  output segment is half loop
 else
    if (ATCR > Thdr1 and $R_m$ > Thdr2 and and A > Thdr3 ) then
 output segment is deep arc
     else
       if (ATCR > Tha1 and $R_m$ > Tha2) then
          output segment is arc
         else if( chord-length >=Threl ) then
                    output segment is global vertical line
                 else output segment is straight line;
end.
```

We have taken Thhl1 = 1.6 ,Thhl2 = 0.5 , Thhl3 = 160 ;
Thdr1 = 1.2 ,Thdr2 = 0.25, Thdr3 = 110;
Tha1 = 1.05 ,Tha2 = 0.13;
Threl = 80 % of height of frame ( minimum orthogonal rectangle containing the character).

For finding $p_m$ we check the pixels which has been selected as successor of current pixel based on movement and the pixel near of it.

**Seperate–loop**

**Input:** Point, which at end of segment has been found to have same marker as is given for that segment called current-pixel, total angle of bending called total-angle-of-bending. count of 8-connected pixel in segment called no-of-8-connected-pixel, count of 4-connected pixel in segment called no-of-4-connected-pixel.

**Output:** pixels constituting the loop part.

**Step1:** find the current-pixel in queue.

q-8-connected-count = 8-connected-count in queue with the current-pixel.

q-4-connected-count = 4-connected-count in queue with the curren-pixel.

q-angle-of-bending = total angle of bending in queue with the current-pixel.

**Step2:** output all the pixels from segment having 8-connected-pixel-count.>= q-8-connected-count.

**Step3:**restore the no-of-8-connected-pixel to q-8-connected, restore the no-of-4-connected-pixel to q-4-connected, restore the total-angle-of-bending to q-angle-of-bending. Set following for the loop : 8-connected-pixel-count to

no-of-8-connected-pixel-count - q-8-connected-count+1;

total-angle-of-bending to

total-angle-of-bending - q-angle-of-bending.

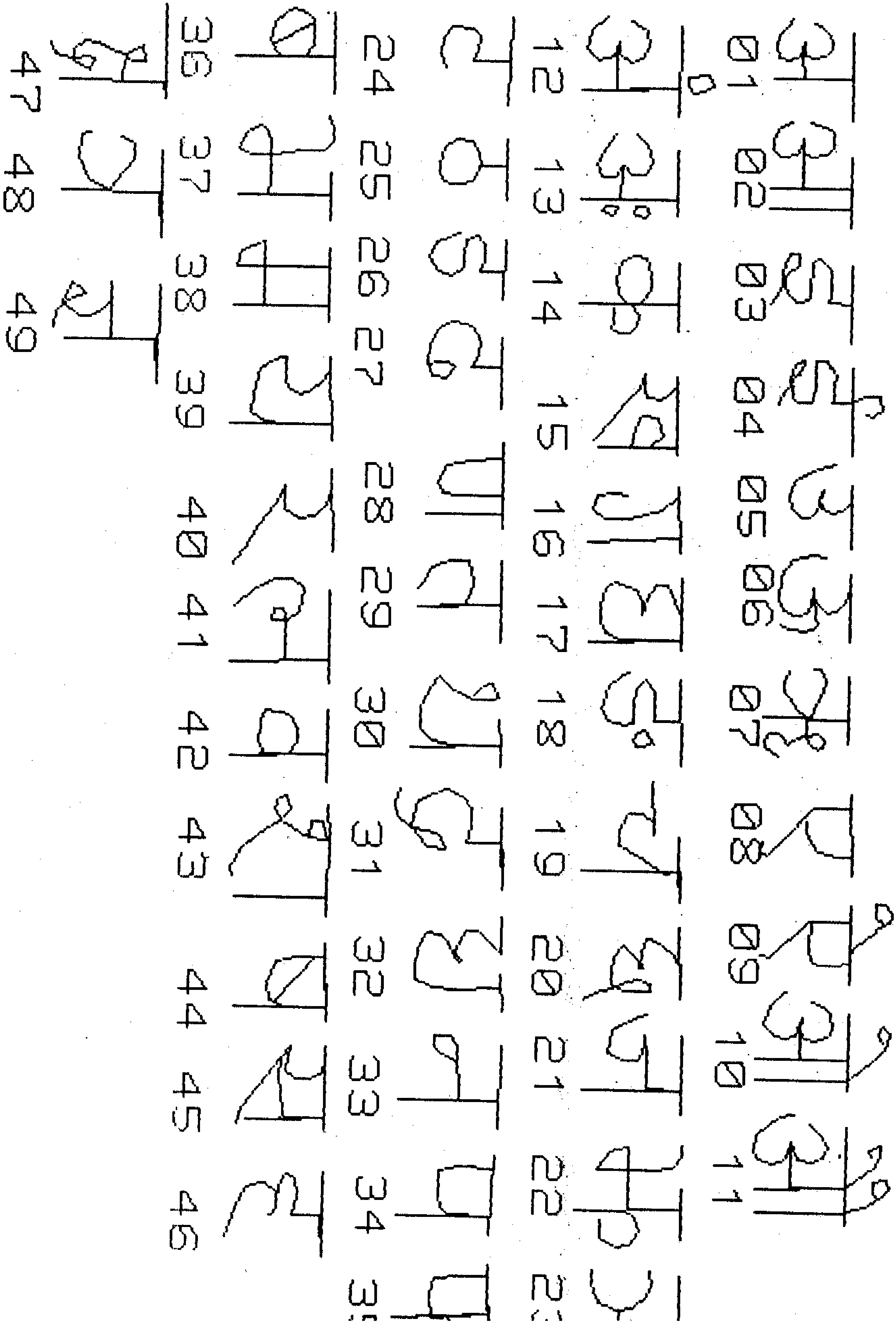
4-connected-pixel-count to

no-of-4-connected-pixel-count - q-4-connected-count+1;

**Stop.**

# APPENDIX

I. Devnagari characters with corresponding assigned label



Let S be the set of selected primitives.

S = {1,2,3,4,5,6}

Where 1 => Global vertical line.

2 => Loop.

3 => Half loop.

4 => Deep arc.

5 => Arc.

6 => Straight line.

Each of these primitives has following attribute.

(i) Connected to which primitive $\in$ S.

(ii) At which pixel it is connected to.

(iii) Movement type(Clockwise/Anticlockwise/No movement).

(iv) Direction of the primitive with the pixel of connection as the origin of co-ordinate axis.

Then we can describe all Devnagari character in terms of these primitives uniquely.

Based on our segmentation process we are presenting a possible description of Devnagari character.

Let a,b $\in$ S. If a is 3 or 4 or 5 then

$a^1$ => primitive "a" has clockwise movement.

$a^2$ => primitive "a" has anticlockwise movement.

a.b => both primitive a and b will be present.

a/b => primitive a or b will be present.

Character label from Table 1 is used to indicate the character.

" " in description is used show no primitive.

| Character label | Character description |
|---|---|
| 1 | 1.(6/$5^1$/$5^2$).((($3^1$/$4^1$/$5^1$).($5^2$/" ").($3^2$/$4^2$/$5^2$).($5^1$/" ")) /(($3^1$/$4^1$/$5^1$).($5^1$/6).($3^2$/$4^2$))) |
| 2 | 1.(6/$5^1$/$5^2$).((($3^1$/$4^1$/$5^1$).($5^2$/" ").($3^2$/$4^2$/$5^2$).($5^1$/" ")) /(($3^1$/$4^1$/$5^1$).($5^1$/6).($3^2$/$4^2$))).1 |
| 3 | (2/" ").((($3^1$/$4^1$/$5^1$).($3^2$/$4^2$/ $5^2$))/((6/$5^1$/$5^2$).($3^2$/$4^2$))) .($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6) |
| 4 | (2/" ").((($3^1$/$4^1$/$5^1$).($3^2$/$4^2$/ $5^2$))/((6/$5^1$/$5^2$).($3^2$/$4^2$))) .($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6).($3^1$/$4^1$/$5^1$) |
| 5 | ($3^1$/$4^1$/$5^1$).($3^1$/$4^1$) |
| 6 | ($3^1$/$4^1$/$5^1$).($3^1$/$4^1$).($3^1$/$4^1$/$5^1$) |
| 7 | 1.(2 /" ").(6 /$3^1$/$4^1$/$5^1$).(6 /$3^2$/$4^2$/$5^2$).($3^1$/$4^1$/$5^1$). ($4^2$/$5^2$/" ").(6 /$5^1$/$5^2$/" ") |
| 8 | (6/$3^1$/$4^1$/$5^1$).(6 /$5^2$).(6 /$5^1$/$5^2$) |
| 9 | (6/$3^1$/$4^1$/$5^1$).(6 /$5^2$).(6 /$5^1$/$5^2$).(2/" ").(6 /$5^2$) |
| 10 | 1.(6/$5^1$/$5^2$).((($3^1$/$4^1$/$5^1$).($5^2$/" ").($3^2$/$4^2$/$5^2$).($5^1$/" ")) /(($3^1$/$4^1$/$5^1$).($5^1$/6).($3^2$/$4^2$))).1.(2/" ").(6 /$5^1$) |
| 11 | 1.(6/$5^1$/$5^2$).((($3^1$/$4^1$/$5^1$).($5^2$/" ").($3^2$/$4^2$/$5^2$).($5^1$/" ")) /(($3^1$/$4^1$/$5^1$).($5^1$/6).($3^2$/$4^2$))).1.(2/" ").(6 /$5^1$). (2/" ").(6 /$5^1$) |
| 12 | 1.(6/$5^1$/$5^2$).((($3^1$/$4^1$/$5^1$).($5^2$/" ").($3^2$/$4^2$/$5^2$).($5^1$/" ")) /(($3^1$/$4^1$/$5^1$).($5^1$/6).($3^2$/$4^2$))).2 |
| 13 | 1.(6/$5^1$/$5^2$).((($3^1$/$4^1$/$5^1$).($5^2$/" ").($3^2$/$4^2$/$5^2$).($5^1$/" ")) /(($3^1$/$4^1$/$5^1$).($5^1$/6).($3^2$/$4^2$))).2.2 |

14 1.(2 /$3^2$).($3^1$/$4^1$/$5^1$)

15 1.(2 /$3^2$).($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6 )

16 1.($3^1$/$4^1$/$5^1$/6 )

17 1.($3^2$/$4^2$/$5^2$).($3^1$/$4^1$/$5^1$).(2/" ")

18 2.((($3^1$/$4^1$/$5^1$).($3^2$/$4^2$/$5^2$))/((6 /$5^1$/$5^2$).($3^2$/$4^2$))).
($3^1$/$4^1$/$5^1$).2

19 1.($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6 )

20 (2.(2/" ").($3^2$/$4^2$/$5^2$).($3^2$/$4^2$/$5^2$).($5^1$/$5^2$/6 ))/
((6/5 /5 ).2.(2/" ").($3^2$/$4^2$/$5^2$).($3^2$/$4^2$/$5^2$).($5^1$/$5^2$/6 ))/
((2/" ").($3^2$/$4^2$/$5^2$).($3^2$/$4^2$/$5^2$).($5^1$/$5^2$/6 ).($3^1$/$4^1$/$5^1$))

21 1.($5^1$/$5^2$/6 ).($3^1$/$4^1$)

22 1.($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6 ).2.($5^2$/(6/6)).(2/" ")

23 1.($5^1$/$5^2$/6 ).($5^1$/6 ).($5^2$/6)

24 (($3^1$/$4^1$/$5^1$).($3^2$/$4^2$))/(($5^1$/$5^2$/6 ).($3^2$/$4^2$))

25 2.(6 /$5^1$/$5^2$/$4^1$)

26 2.((($3^1$/$4^1$/$5^1$).($3^2$/$4^2$/$5^2$))/((6 /$5^1$/$5^2$).($3^2$/$4^2$))).
($3^1$/$4^1$/$5^1$)

27 2.((($3^1$/$4^1$/$5^1$).($3^2$/$4^2$))/($5^1$/$5^2$/6 ).($3^2$/$4^2$)))

28 1.2

29 1.($3^2$/$4^2$/$5^2$)

30 1.2.($3^1$/$4^1$/$5^1$).($3^2$/$4^2$/$5^2$/6 /" ")

31 2.($5^1$/$5^2$/6 ).(($3^1$/$4^1$/$5^1$).($3^2$/$4^2$))/(($5^1$/$5^2$/6 ).($3^2$/$4^2$))

32 1.($3^1$/$4^1$).(2 /$5^1$/$5^2$/6 ).($3^1$/$4^1$/$5^1$).(2/" ")

33 1.2.($5^1$/$5^2$/6 )

34 1.($3^2$/$4^2$/$5^2$)

35 1.($3^2$/$4^2$/$5^2$).($3^1$/$4^1$/$5^1$)

36 1.2.($5^1$/$5^2$/6 )

37 1.2.($5^1$/$5^2$/6 ).($3^2$/$4^2$/$5^2$/(6.(6/" "))).(2/" ")

38 1.2.($5^1$/$5^2$/6 ).($5^1$/6 )

39 1.($3^1$/$4^1$/$5^1$).($3^1$/$4^1$/$5^1$)

40 ($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6)

41 (1.($3^2$/$4^2$).(2 /$5^1$/$5^2$/6).($5^1$/$5^2$/6 /" "))/
(($3^1$/$4^1$/$5^1$).(2 /$5^1$/$5^2$/6 ).($3^1$/$4^1$/$5^1$).($3^2$/$4^2$/$5^2$))

42 1.(2 /$3^2$/$4^2$)

43 1.2.(2/" ").($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6)

44 1.($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6)

45 1.($3^1$/$4^1$/$5^1$).($5^1$/$5^2$/6).($5^1$/$5^2$/6)

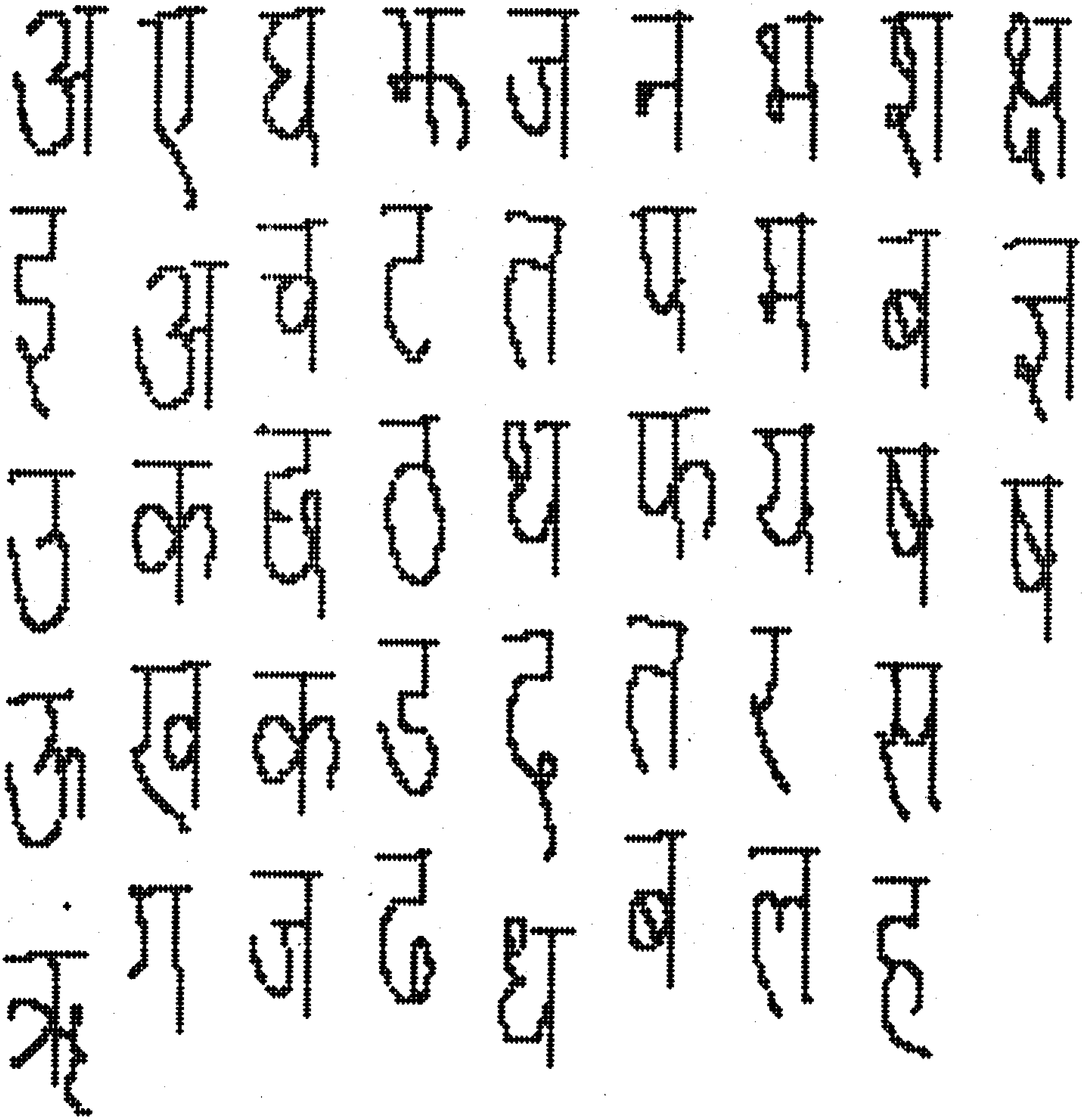
46 ($3^1$/$4^1$/$5^1$/$5^2$/6).($3^2$/$4^2$/$5^2$).($3^2$/$4^2$/$5^2$).($5^1$/$5^2$/6 /" ")

47 1.2.(2/" ").($5^1$/$5^2$/6 ).($5^1$/$5^2$/6).($3^2$/$4^2$/$5^2$)

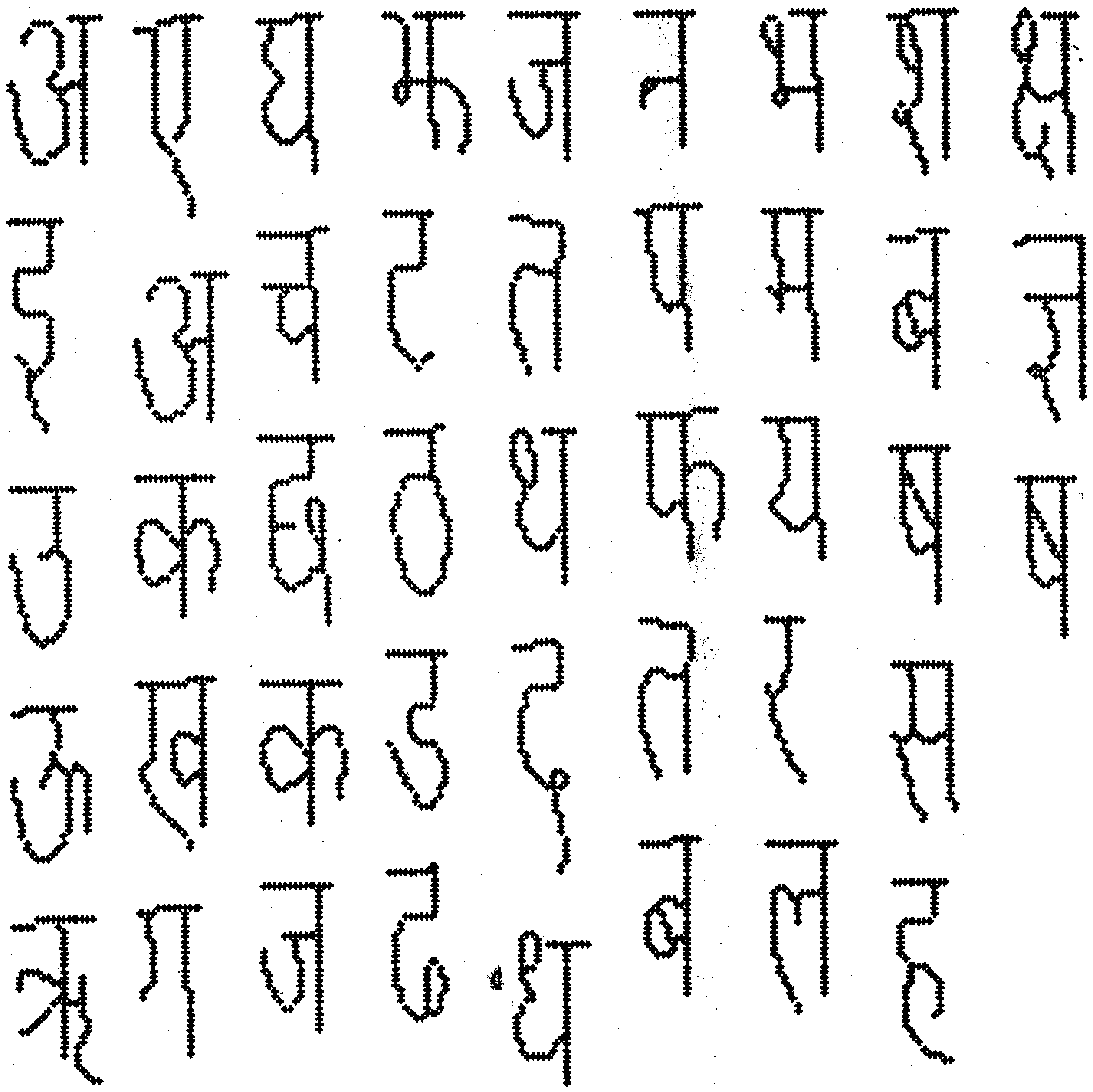
48 1.($5^1$/$4^1$/6 ).($5^2$/$4^2$/6 )

49 1.2.($5^1$/$5^2$/6 ).($3^1$/$4^1$/$5^1$).(($5^1$/$5^2$/6 )/($3^2$/$4^2$))

## III. Characters before preprocessing



## IV. Characters after preprocessing



```
CHARACTER BEFORE SEGMENTATION    SEGMENTS AFTER SEGMENTATION

            1111111                 ****
     1111      1                   *    *                *
    1    1     1                  *      *               *
   1      1    1                  *      *           (a) *
   1      1    1                         *               *
          1    1                   (c)   *               *
          1    1                         *               *
          1    1                         *      (b)      *
          1    1                         *       ***     *
          1    1                        *       *        *
         1     1                       *       *         *
        1      1                       *                 *
1       1   1111                                         *
 1       111   1                   *                     *
 1        1    1                    *                    *
 1        1    1                    *        *           *
 1         1   1                    *   (d)   *          *
 1         1   1                    *         *          *
 1         1   1                    *         *          *
  1        1   1                     *        *          *
  1        1   1                     *        *          *
  1        1   1                     *        *          *
   1       1   1                      *       *          *
   1       1   1                      *       *          *
   1      1    1                      *      *           *
    1   11     1                       *   **            *
     111       1                        ***              *
```

(a) Global vertical line.

(b) straight line.

(c) & (d) are half loops.

```
CHARACTER BEFORE SEGMENTATION

   1111111111
111     1
        1
         1
         1
         1
         1
         1
        1
        1
        11   1
       1  1 1 1
       1   1   1
1     1    1   1
1     1    1   1
1          1   1
1          1   1
 1         1   1
 1         1   1
 1         1   1
 1         1   1
  1        1   1
  1        1   1
  1       1
   1      1
   1     1
    1   1
     111
```

```
SEGMENTS AFTER SEGMENTATION

    *
    *            *
     *          * *
(a)  *             *
     *             *
     *         (c) *
     *             *
    *              *
    *              *
    *              *
   *               *
   *               *
  *     *          *
  *      *         *
          *
          *
          *
          *
          *
*         *
*         *
*         *
*         *
 *        *
 *        *
 *       *
  *      *
  *     *
   *   *
    ***
```

(a) straight line.
(b) half loop.
(c) arc.

```
CHARACTER BEFORE SEGMENTATION   SEGMENTS AFTER SEGMENTATION .

1111111111111111
          1                                    *
         1                                    *
         1                                    *
         1                                    *
         1                                    *
         1                     ***            *
         1                    *   *           *
  111    1  111               *    *          *         ***
 1   1   1 1   1             *      *         *        *   *
 1    1  11    1             *       *        *       *    *
1      1 1     1             *                *            *
1       11      1            *                *             *
1        1      1             *               *             *
1        1      1             *      *        *             *
 1       1      1             *     *         *             *
 1      11      1              *  **          *             *
 1     1 1     1                **            *            *
  1  11  1     1                              *            *
   11    1     1                              *            *
         1     1                              *            *
         1                                    *
         1                                    *
         1                                    *
         1                                    *
         1                                    *
```

(a) Global vertical line.

(b) Loop.

(c) Deep arc.

CHARACTER BEFORE SEGMENTATION

```
          111111
1111111111   1
   1         1
   1         1
   1         1
   1         1
   1         1
   1         1
   1         1
   1         1
   1    111  1
   1   1   111
   1   1     1
   1   1     1
   1   1     1
  1    1     1
 1     1     1
 1      1  111
 1      1 1  1
  1      1   1
  1          1
   1         1
   1         1
    1        1
     1       1
      1      1
       1     1
        1
         1
          1
          1
```

SEGMENTS AFTER SEGMENTATION

```
                           *
  *                        *
  *                        *
  *                        *
  *                        *
  *                        *
  *                        *
  *  (c)                   *
  *                    (a) *
  *        ***             *
  *       *   **           *
  *       *                *
  *       *                *
  *       * (b)            *
 *        *                *
*         *                *
           *  **           *
  *        * *             *
  *         *              *
   *                       *
   *  (d)                  *
    *                      *
    *                      *
     *                     *
      *                    *
       *                   *
        *
         *
          *
           *
           *
```

(a) Global vertical line.

(b) Loop.

(c) and (d) Straight line.

```
CHARACTER BEFORE SEGMENTATION   SEGMENTS AFTER SEGMENTATION

       111111
1111111   1                                               *
  1       1                              *                *
  1       1                              *                *
  1       1                              *                *
 1        1                             *                 *
 1        1                             * (b)             *
 1        1                             *                 *
 1        1                             *                 *
  1       1                              *                *
   1      1                               *               *
    1     1                                *              *
    1     1                                          (a)  *
   1      1                           *                   *
  1       1                          *                    *
  1       1                         *                     *
 1        1                         *                     *
 1        1                        *  (c)                 *
 1        1                        *                      *
 1        1                        *                      *
  1      11                        *                      *
  1     1 1                         *      *              *
   1   1  1                         *     *               *
    111    1                         *   *
           1                          ***
           1
           1
           1
           1
           1
```

a) Global vertical line.

b) Arc.

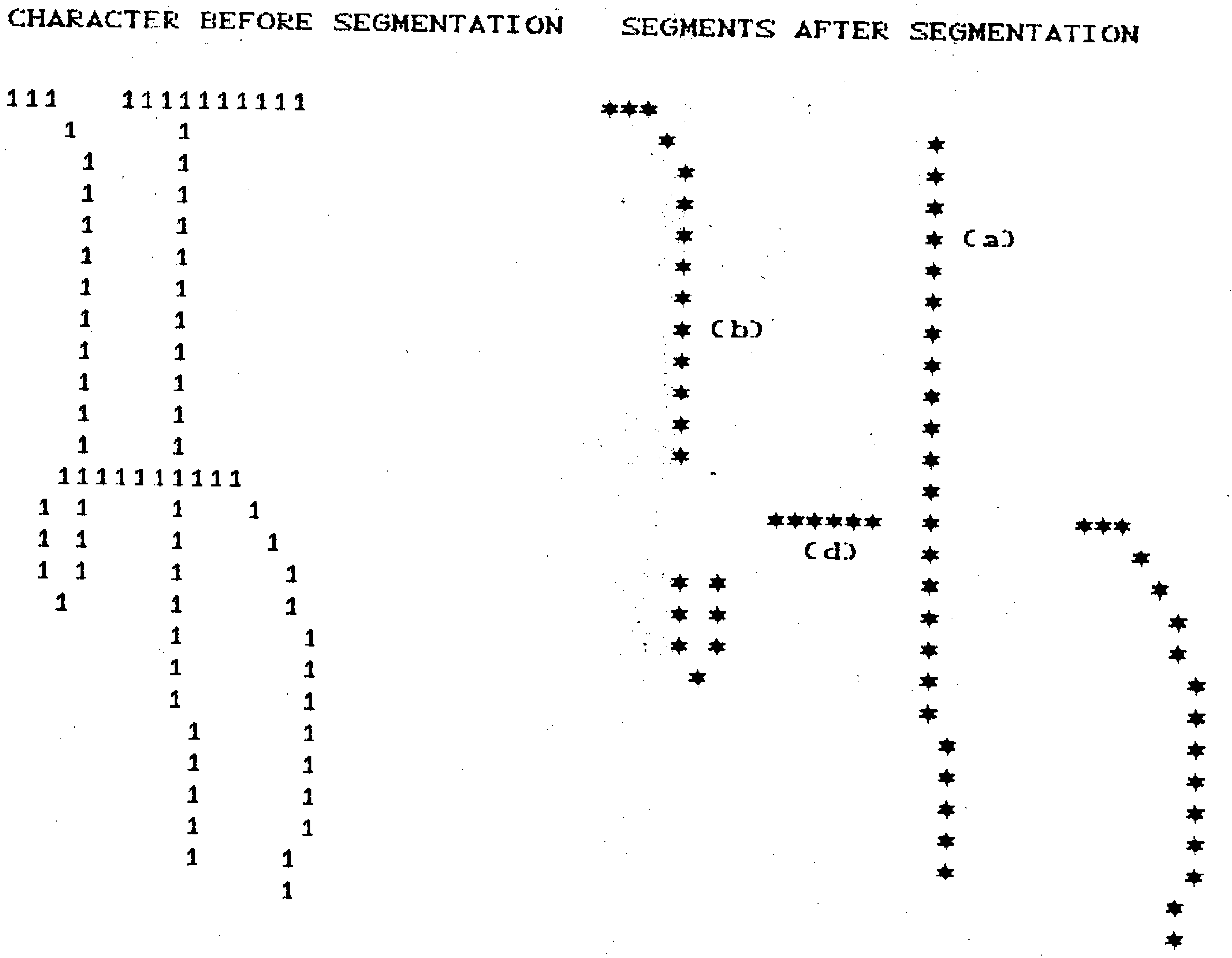
c) Half loop.

CHARACTER BEFORE SEGMENTATION   SEGMENTS AFTER SEGMENTATION

```
111111111111111
          1
          1
          1
          1
          1
          1
      1111
   111
  1
  1
  1       11
  1      1  1
  1      1  1
  1      1  1
  1      1  1
  11111   1 1
  1       1 1
  1        11
  1         1
  1         1
  1         1
  1         1
  1         1
  1         1
   1      111
   1     1   1
    1  11    1
     11      1
             1
             1
             1
             1
             1
             1
```

```
                *
                *
           (a)  *
                *
                *
                *
            ****
         ***
*                 **
*                *  *
*                *  *
*                *  *
* (b)        (c) *  *
*                 * *
*                 * *
*                  *
*         *
*         *        *
*         *         *
*         *         *
*         *         *
*         *         * (d)
 *      **          *
 *     *            *
  *  **             *
   **               *
                    *
   **** (e)
```

(a) Arc.
(b) Half loop.
(c) Loop.
(d) and (e) Straight line.


CHARACTER BEFORE SEGMENTATION

SEGMENTS AFTER SEGMENTATION

(a) Global vertical line.

(b) and (c) arc.

(d) Straight line.

(e) Loop.

Based upon the different types of test we can group the nodes in following way (type of test is as described in decision tree description):

| Type of test conducted. | Node(s) |
|---|---|
| (a) | 1. |
| (b) | 3,6. |
| (c) | 5,7. |
| (d) | 9,12,13,37,249. |
| (e) | 4,8,15,38. |
| (f) | 2,11,61,62. |
| (g) | 19,24,27,28,29,41,42,46,47,52,54,55,64,<br>65,66,67,70,73,75,79,80. |
| (h) | 16,48. |
| (i) | 17,33,40,49,56,68. |
| (j) | 20,22,31,32,53,57,58,59,77. |
| (k) | 18,30,34,39,44,51,63. |
| (l) | 21,35,82,83,84. |
| (m) | 23,25. |
| (n) | 26,43,45,69,71,72,76,250. |
| (o) | 50,60,252. |

# REFERENCES

1.K.s.Fu, Syntactic pattern recognition and applications. Pentice Hall, Englewood Cliffs, New Jersey (1982).

2.J.Mantas, An overview of character recognition methodologies, Pattern Recognition 19,425-430 (1986).

3.C. Y. Suen, M. Berthod and S. Mori, Automatic recognition of handprinted characters-the state of the art, Proc. IEEE 68, 469-485 (1980).

4.Harjinder Singh, Description aided recognition of handprinted characters, Ph.D. Thesis, Indian Institute of Science, Bangalore, India (1985).

5.V. K. Govindan, A. P. Shivprasad, Character recognition - A review, Pattern recognition 23, 671-683, 1990.

6.A. K. Datta, A generalized formal approach for description and analysis for major Indian Scripts, J. Inst. Elec. Telecom. Engng. 30, 155-161 (1984).

7.K. C. You, K. S. Fu Syntactic approach to shape recognition using attributed grammar, IEEE T. Syst. Man Cyb. 9, 334-345 (1979).

8.I. K. Seth and B. Chatterjee, machine recognition of constrained handprinted Devnagari, Pattern Recognition 9, 69-75 (1977).

9.Y. S. Cheng and C. H. Leung, Chain code transform for chinese character recognition, IEEE 1985, Proc. Int. Conf. Cyb. Soc., Tuscon, AZ, U.S.A. pp. 42-45 (1985).

10. M. Kushir, K. Abe and K. Matsumuto, An application of Hough transform to the recognition of Hebrew character, Pattern Recognition 16, 183-191 (1983).

11.A. Oulamara and J. Duvernoy, An application of Hough transform to automatic recognition of Berber characters, Signal process. (Netherlands) 14, 79-90 (1988).

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

13.R. D. Badoux, DELTA [text reader for blind], Computerised braile production, proc. 5th Int. workshop, Winterthur, Switzerland, pp. 21-25 (30 october-1 november 1985).

14. 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).

15.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 (6 may 1969).

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

17.A. Gyarfas, Experiments concerning the inspection and control of car and truck in France, Koezlekedes Tud. Sz. 24, 85-91 (1974) (In Hungarian).

18.G. L. Skalski, OCR in the publishing industry, Data processing XII, Proc. 1967 Int. Data Process. Conf. and Business Exposition, Boston, MA, U.S.A., pp. 255-260 (20-23 june 1967).

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

20.E. G. Nassimbene, Digital compare circuitry, IBM Technical Disclosure Bull. 14, 3421-3422 (1972).

21.J. W. T. Smith and Z. Merali, optical character recognition: the technology and its application in information units and libraries, Report 33, British Library, Boston Spa, Wetherby, West Yorks, England (1985).

22.L. N. Kanal and B. Chandrasekaran, On linguistic, statistical and mixed models of character recognition, Frontiers of Pattern Recognition, S. Watanable, Ed., pp. 163-192. Academic Press, New York (1972).

23.M. Sridhar and A. Badreldin, High accuracy character recognition algorithm using Fourier and topological descriptors, Pattern Recognition 17, 515-523 (1984).

24.M. Sridhar and A. Badreldin, High accuracy syntactic recognition algorithm for handwritten numerals, IEEE T. Syst. Man Cyb. 15, (1985).

25. E. Persoon and K. S. Fu, Shape determination using Fourier descriptors, IEEE T. Syst. Man Cyb. 7, 170-179 (1977).

26.H. F. Lai and S. C. Cheng, Projection profile and Fourier Transform for Chinese character recognition, Int. J. Elec. (U.K.) 54, 299-300 (1983).

27.J. S. Huang and M. Lung, Seperating similar complex Chinese character by Walsh transform, Pattern Recognition 20, 425-428 (1987).

28.G. P. R. Sarvarayudu and I. K. Sethi, Walsh descriptors for polygonal curves, Pattern Recognition 16, 327-336 (1983).

29.S. Wendling, G. Stamon, Hadamard and Haar trransforms and their power spectrum in character recognition, 1976 Joint Workshop on Pattern Recognition and Artificial Intelligence, Hyannis, Mass, U.S.A., 1-3 june 1976, 103-112 (1976).

30.M. Kushnir and K. Matsumuto, Recognition of handprinted Hebrew characters using features selected in Hough transform space, Pattern Recognition 18, 103-114 (1985).

31.N. D. Tucker and F. C. Evans, A two-step strategy for character recognition using geometrical moments, Proc. 2nd Int. J. Conf. Pattern Recognition, pp. 223-225 (1974).

32.J. R. Ullmann, Experiments with n-tuple method for pattern recognition, IEEE T. Comput. 18, 1135-1137 (1969).

33.A. L. Knoll, Experiments with "characteristic loci" for recognition of handprinted characters, IEEE T. Comput. 18, 366-372 (1969).

34.A. W. Holt, Algorithm for a low-cost handprint reader, Comput. Design, 85-89 (February 1974).

35.P. M. Lewis, The characteristic selection problem in recognition system, IEEE T. Inform. Theory 8, 171-178 (1962).

36.C. H. Chen, Computer searching criterion for best feature set in character recognition, Proc. IEEE 53, 2128-2129 (1965).

37.K. S. Fu and G. P. Cardillo, A note on optimum feature selection, IEEE T. Automat. Contr. 12, 588-591 (1967).

38.W. C. Naylor, Some studies in the interactive design of character recognition system, IEEE T. Comput. 20, 1075-1086 (1971).

39.F. Ali and T. Pavlidis, Syntactic recognition of handwritten numerals, IEEE T. Syst. Man Cyb. 7, 537-541 (1977).

40.K. Yamamoto and S. Mori, Recognition of handprinted characters by outermost point method, Proc. 4th Int. J. Conf. Pattern Recognition, Kyoto, Japan pp. 794-796 (7-10 november 1978).