CHARACTER RECOGNITION USING GEOMETRICAL ANALYSIS

Project Report submitted in partial fulfilment of the requirement for the

degree of

Bachelor of Technology

in

Electronics and Communication Engineering

By

PRATEEK CHATURVEDI (061097)
PRATIK GOLCHHA (061099)
RITESH MITRUKA (061112)
VIBHANSHU PRATAP SINGH (061139)

Under the supervision of
Prof. Sunil V. Bhooshan

Jaypee University of Information Technology

Waknaghat, Solan - 173 234, Himachal Pradesh

Dedicated to our Parents

ACKNOWLEDGEMENT

The zeal to accomplish the task of formulating the project report on "**Character Recognition Using Geometrical Analysis**" could not have been realized without the support and cooperation of the members of the faculty.

We take this opportunity with much pleasure to thank all the people who have helped us through the course of our journey towards producing this thesis. We sincerely thank our project supervisor, **Prof. Sunil V. Bhooshan** (Head of Department, Electronic and Communication) for his guidance, help and motivation. Apart from the subject of my research, we learnt a lot from him, which we are sure will be useful in different stages of our life.

We would like to express our gratitude to **Dr. Vinay Kumar**(Asst. Professor, Electronics and Communication Department)whose able guidance and vision to the project helped us add new dimension to our approach .He helped us develop novel solutions to every problem and helped us emerge with good engineering acumen.

[illegible]
Prateek Chaturvedi
(061097)

Pratik Golchha
Pratik Golchha
(061099)

[illegible]
Ritesh Mitruka
(061112)

Vibhanshu
Vibhanshu Pratap Singh
(061139)

CERTIFICATE

This is to certify that the work entitled "Character Recognition Using Geometrical Analysis" submitted by **Prateek Chaturvedi, Pratik Golchha, Ritesh Mitruka, Vibhanshu Pratap Singh** in partial fulfilment for the award of degree of Bachelor of Technology, of Jaypee University of Information Technology, in 2010 has been carried out under my supervision. This work has not been submitted partially or wholly to any other university or institute of award for this or any other degree or diploma.

[illegible]

Dr. Vinay Kumar

(Asstt. Professor)

(Electronics and Communication)

[illegible]

Prof. Sunil V. Bhooshan

(Head of Department)

(Electronics and Communication)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Optical character recognition refers to the process by which document is scanned into a computer and analyzed to determine which characters appear in the document. This project work describes an angelic font recognition method using stroke pattern analysis on decomposed word images. The character images are extracted from scanned text documents containing words objects in various fonts and styles. The work done in the field of font recognition priory, mainly focus on pattern analysis on single character or large text blocks which are subject to font and style variations such as size, serif , boldness etc. We have extended this work for lower case alphabets and words. Our method takes advantage of Thinning Algorithm which is applied on each word and character image and performs statistical analysis on stroke patterns obtained from decomposed sub-images. Experiments conducted on small character images extracted from scanned documents also show an encouraging result. The need of using simple and innovative concepts in order to reduce the complexity of task at hand was also realized.

CHAPTER-1
INTRODUCTION

Optical Character Recognition or OCR is a technique that actually recognizes or reads the textual content on the paper document and transforms it into a machine readable form.

Optical Character Recognition (OCR) is a visual recognition process that turns printed or written text into an electronic character-based file. Scanning a document and converting it to a PDF provides the basis for character recognition methods whereby an electronic character based file is assigned to each individual character that is read from the PDF document.

This technique termed OCR obviates the onerous task of manually typing the document into the computer system. Hence it finds successful application in situations where there is need of entering voluminous documents into computers or other databases for purposes such as archiving or analysis of documents. The platform to OCR technology is provided by the character library where the match for the scanned character is searched and compared.

To aid OCR systems in identifying words with various fonts and styles in the document images, several font recognition methods have been proposed. As can be seen from anterior methods, Optical Character Recognition places a heavy reliance on a character library for identification of data in an input document. This works well for printed fonts, and works in some cases for cursive scripts as well As far as efficiency of this method is concerned, the cases in which poor results are obtained, several processing attempts will be required before the system actually finds the erroneous match. It is therefore necessary to have an effective OCR system that does not rely on character library to identify the data in an input document.

The idea on this year long project was conceived with a vision that there is a need to reduce the complexity involved in the field of character recognition and thus we began the search for an innovative approach to character recognition problems.

To implement this, first, Thinning algorithm is applied to each word image to produce sub-images representing different feature components. Thinning is a structural operation that is used to remove selected foreground pixel from binary images. Then, statistical analysis of stroke patterns, including both vertical and horizontal strokes, is performed on the sub-images to distinguish between different characters. In the process of our recognition, we have assumed the pixel grid to be a virtual triangle rather than a square.

The general idea is that we see words as a complete patterns rather than the sum of letter parts. Some claim that the information used to recognize a word is the pattern of ascending, descending, and neutral characters. Another formulation is to use the envelope created by the outline of the word. We have considered words as the sum of letter parts to obtain a less complex method of recognition. We have also extended our work to small alphabets.

CHAPTER-2

PROJECT PROGRESS

The algorithm has gone through several stages over a period of one year and thus it becomes mandatory to discuss the assiduous efforts of the team and process through which the final algorithm has evolved. The project time line is shown in the Figure (1).

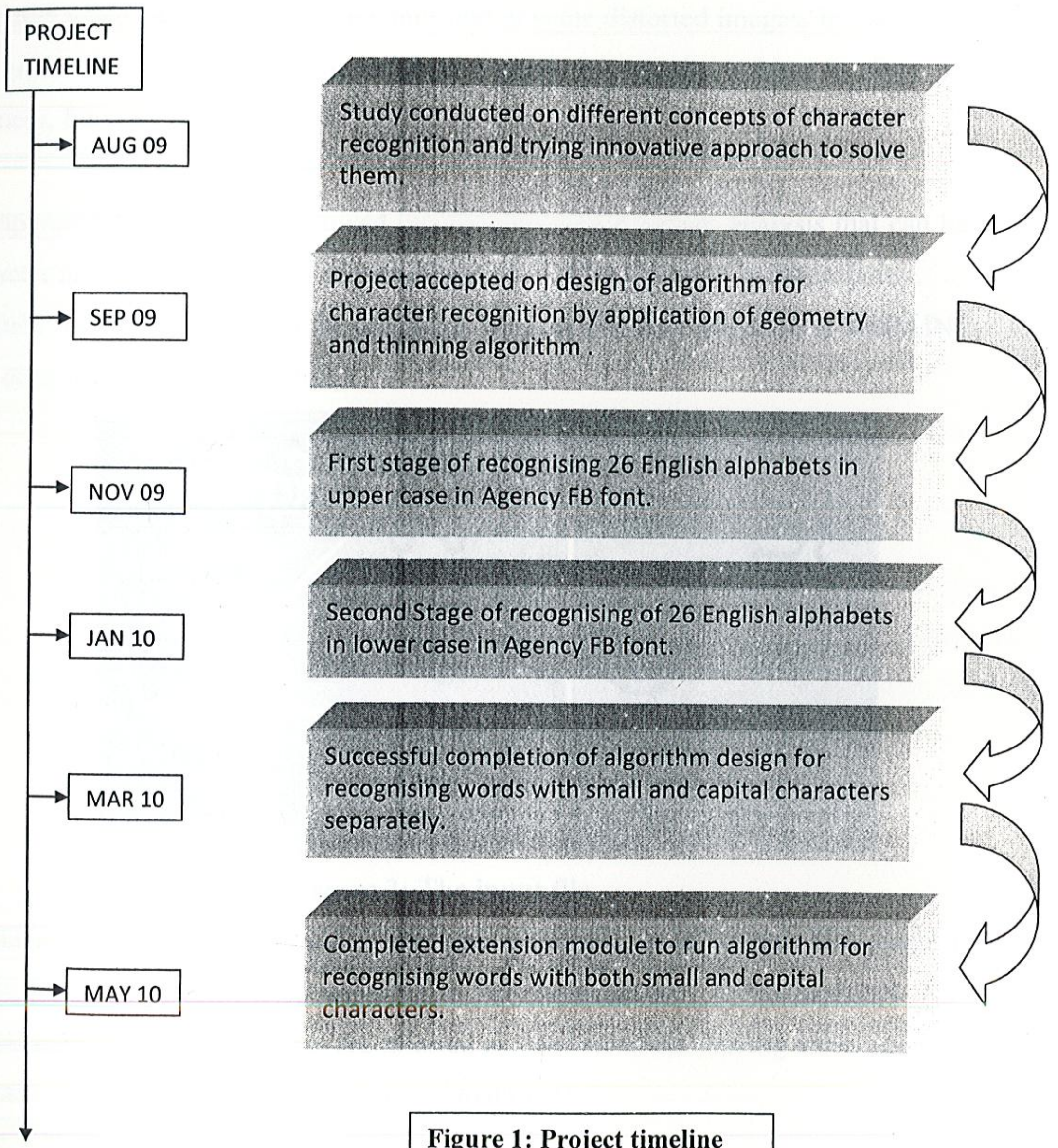

PROJECT TIMELINE
AUG 09
Study conducted on different concepts of character recognition and trying innovative approach to solve them.
SEP 09
Project accepted on design of algorithm for character recognition by application of geometry and thinning algorithm .
NOV 09
First stage of recognising 26 English alphabets in upper case in Agency FB font.
JAN 10
Second Stage of recognising of 26 English alphabets in lower case in Agency FB font.
MAR 10
Successful completion of algorithm design for recognising words with small and capital characters separately.
MAY 10
Completed extension module to run algorithm for recognising words with both small and capital characters.

Figure 1: Project timeline

CHAPTER-3

METHODOLOGY

With the need of effective and accurate approaches for extracting and retrieving information from scanned document images, many information retrieval techniques have been proposed over the years. One common technique is to convert the scanned text images into machine readable format using OCR, which is followed by text retrieval.

Our technique is based on feature analysis of each individual character. The efficiency of algorithm decreases when characters are interconnected in some distorted images. In addition, shape properties and gradient information of the original images are usually subject to font variations such as boldness, font size, font type etc.

The approach that we have followed incorporates stroke pattern analysis that can be applied to each character image, to match a set of predefined criteria for character differentiation.
To study the characteristics of the character, we start withdrawing them in MSPAINT, then determining their properties.

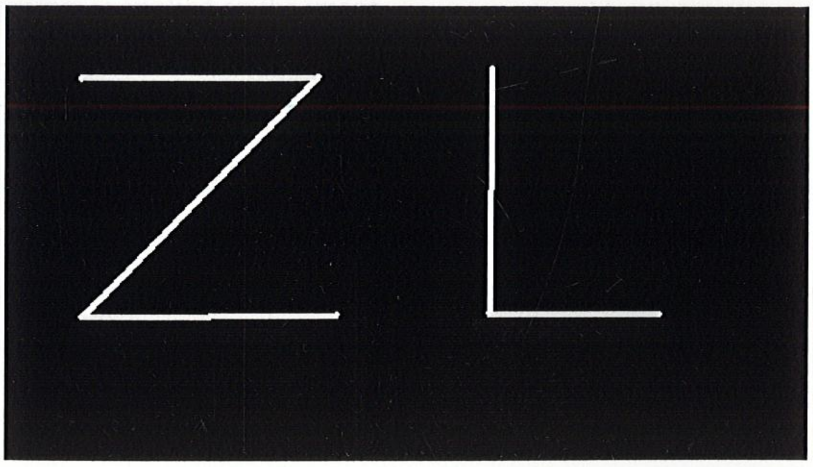


Figure 2: The input file

We have considered the figure 3.1, the input character, as an image. The image is converted into matrix with every element contributing to the characteristics of the alphabet. All the cells of the matrix representing the alphabet are referred as 'true elements' having value equal to '0'and all other cells as 'non true elements' having value equal to '1'.

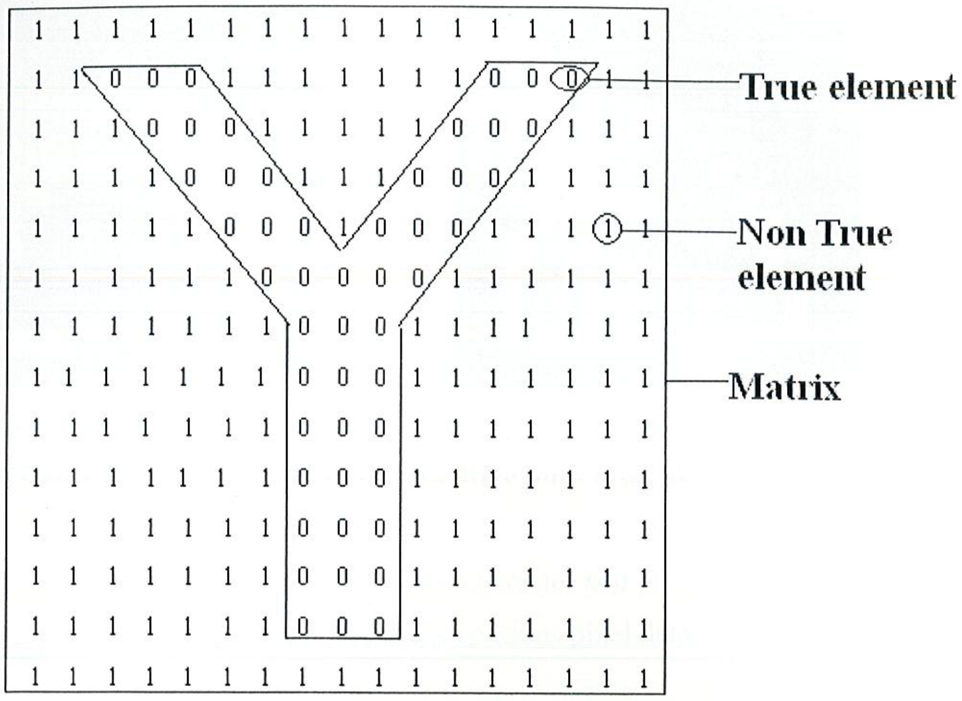


Figure 3: The input file converted to matrix form

The character shown in the matrix form in the above figure contains repetitive lines of the 'true elements' and 'non true elements'. To reduce all the repetitive lines to a single pixel thickness, Thinning algorithm is applied on the image.

3.1 Thinning Algorithm

Thinning is a structural operation that is used to remove selected foreground pixel from binary images. It can be used for various applications, but is particularly useful for skeletonization and in accurate calculation of parameters present in an individual character. Thinning is normally only applied to binary images, and produces another binary image as output.

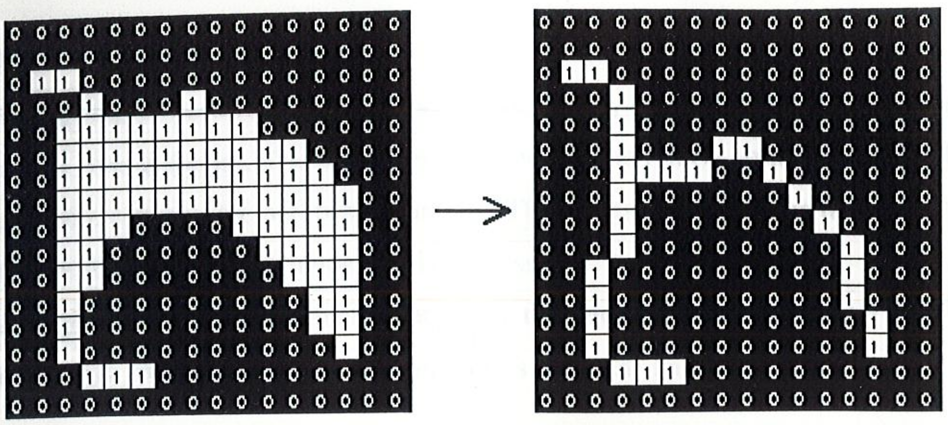


Figure 4: Reduction in the number of repetitive lines after skeletonization

Thinning preserves the unique characteristics of the character and also maintains the connectivity of the original region while throwing away most of the spurious pixel data.

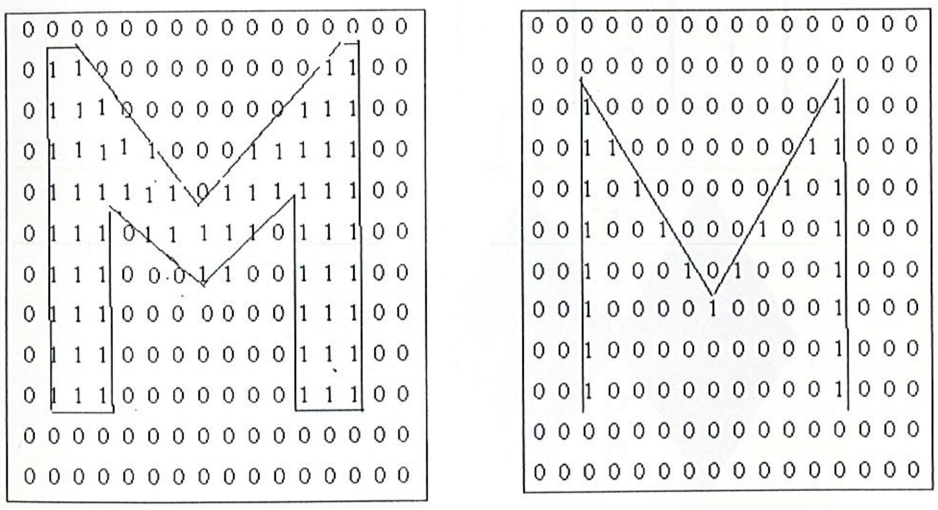


Fig. Before Thining **Fig. After Thining**

Figure 5: Application of thinning algorithm

As can be seen from the figure above, the thickness of character 'M' is reduced from 3 layers to a single layer. The difference between both the figures by visualisation only, one can see that the properties of the character are more prominent ion the 'after thinning' image. This laid the foundation to search for and design the algorithm for character recognition with an easy approach and keeping the point of reducing the complexity in mind. .

Major disadvantage of thinning includes distortion of images. A certain trade off is required to maintain balance between thinning and amount of distortion.

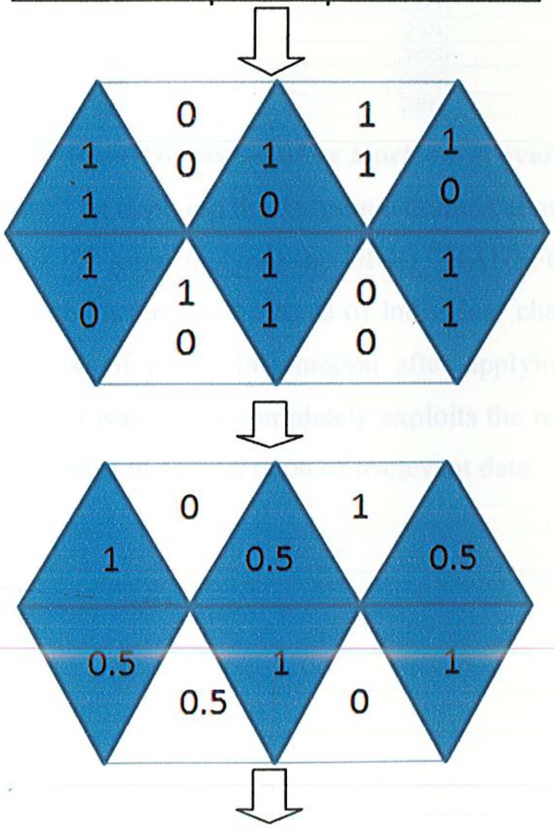
3.2 GRID TRANSFORMATION

Now we are going to perform grid transformation, that is, pixel grid will be considered as a virtual triangle rather than a square. The square grid as seen in the matrix form is converted into triangular by averaging the consecutive pixels horizontally. The representation of the matrix in '1' and '0' form and setting the threshold values helps in this process.

The advantage of using grid transformation is the increment in horizontal and vertical stroke information of the character as curve information is reduced. The properties

The consecutive pixels containing values '0' and '1' are stored in a different matrix and values greater than 0.5 is converted as '1'. Similarly '0' is stored for consecutive zeros and '1' for consecutive ones.

1	0	1	1	1
1	0	0	1	0
1	1	1	0	1
0	0	1	0	1



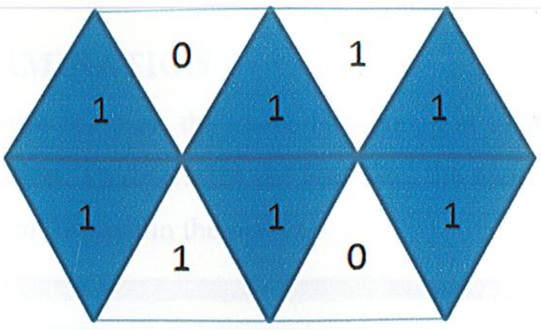
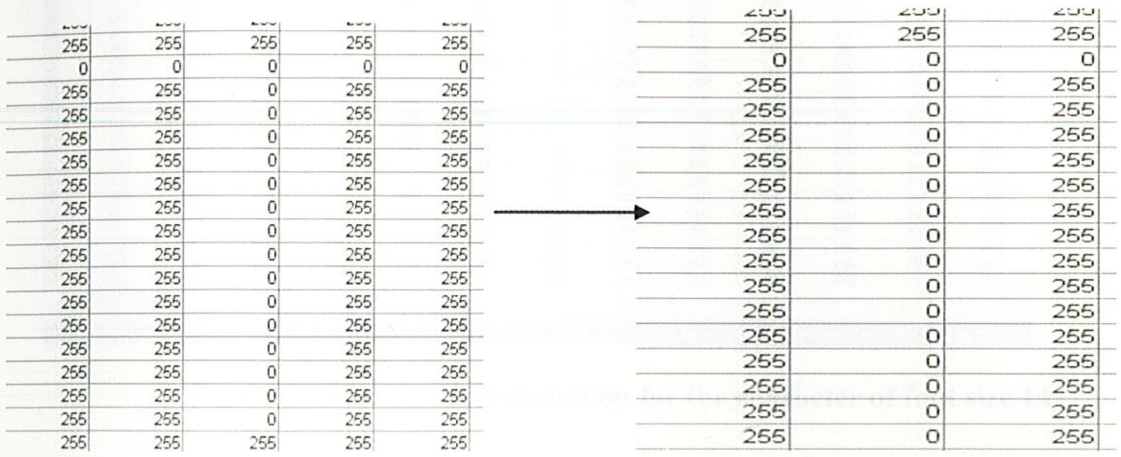


Figure 6: Compression of the matrix by grid transformation

As shown in the above matrix the elements contained in the triangle are averaged and the values are stored as calculated. This represents a virtual triangle. The final values as obtained after threshold application are shown in the triangular grid.



255	255	255	255	255
0	0	0	0	0
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	0	255	255
255	255	255	255	255

255	255	255
0	0	0
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255
255	0	255

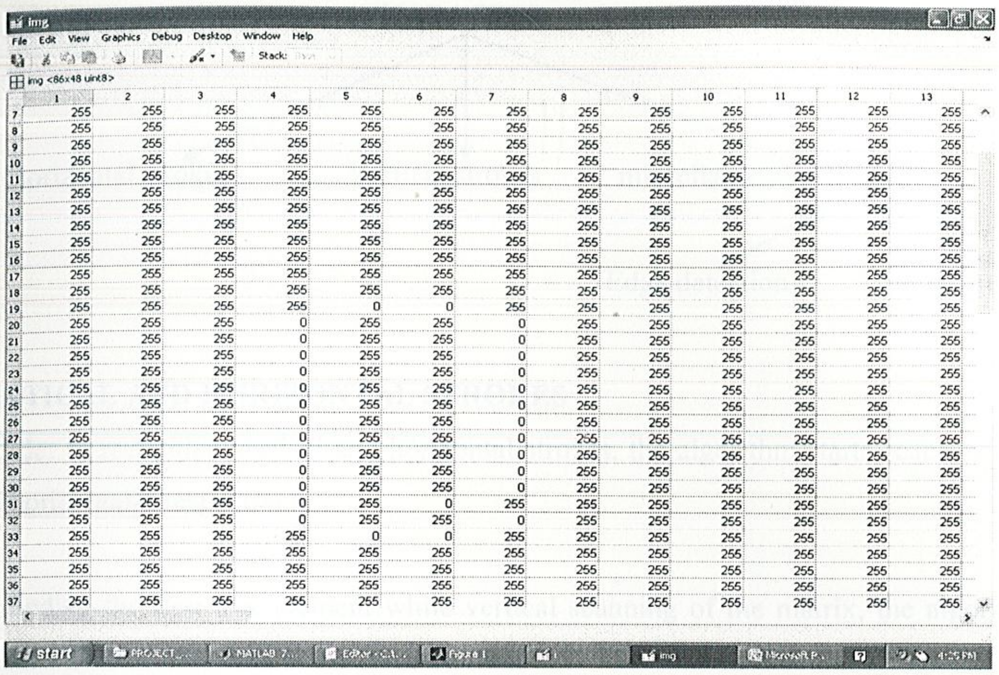
Figure 7: Result obtained after Horizontal averaging

The above figure shows the result on applying the technique defined above on the character ‘T’.

The characteristics of character ‘T’ after applying the GRID TRANSFORMATION remains same, which helps us to differentiate characters on the basis of individual characteristics. As can be seen there is very low loss or no loss of useful information after applying the threshold. Thus grid transformation is applied in such a way that it completely exploits the redundancy of data present in the image. This technology also helps in compression of irrelevant data.

3.3 FONT SIZE DETERMINATION

Taking the character input from the user, the exact determination of the font size of the character becomes a valuable consideration. To determine the font size, the matrix is horizontally scanned for the first and the last true element present in the matrix.



img <86x48 uint8>

	1	2	3	4	5	6	7	8	9	10	11	12	13
7	255	255	255	255	255	255	255	255	255	255	255	255	255
8	255	255	255	255	255	255	255	255	255	255	255	255	255
9	255	255	255	255	255	255	255	255	255	255	255	255	255
10	255	255	255	255	255	255	255	255	255	255	255	255	255
11	255	255	255	255	255	255	255	255	255	255	255	255	255
12	255	255	255	255	255	255	255	255	255	255	255	255	255
13	255	255	255	255	255	255	255	255	255	255	255	255	255
14	255	255	255	255	255	255	255	255	255	255	255	255	255
15	255	255	255	255	255	255	255	255	255	255	255	255	255
16	255	255	255	255	255	255	255	255	255	255	255	255	255
17	255	255	255	255	255	255	255	255	255	255	255	255	255
18	255	255	255	255	255	255	255	255	255	255	255	255	255
19	255	255	255	255	0	0	255	255	255	255	255	255	255
20	255	255	255	0	255	255	0	255	255	255	255	255	255
21	255	255	255	0	255	255	0	255	255	255	255	255	255
22	255	255	255	0	255	255	0	255	255	255	255	255	255
23	255	255	255	0	255	255	0	255	255	255	255	255	255
24	255	255	255	0	255	255	0	255	255	255	255	255	255
25	255	255	255	0	255	255	0	255	255	255	255	255	255
26	255	255	255	0	255	255	0	255	255	255	255	255	255
27	255	255	255	0	255	255	0	255	255	255	255	255	255
28	255	255	255	0	255	255	0	255	255	255	255	255	255
29	255	255	255	0	255	255	0	255	255	255	255	255	255
30	255	255	255	0	255	255	0	255	255	255	255	255	255
31	255	255	255	0	255	0	255	255	255	255	255	255	255
32	255	255	255	0	255	255	0	255	255	255	255	255	255
33	255	255	255	255	0	0	255	255	255	255	255	255	255
34	255	255	255	255	255	255	255	255	255	255	255	255	255
35	255	255	255	255	255	255	255	255	255	255	255	255	255
36	255	255	255	255	255	255	255	255	255	255	255	255	255
37	255	255	255	255	255	255	255	255	255	255	255	255	255

Figure 8: Font size determination for the character of font size 14

The corresponding horizontal indices of the respective true elements are stored. Font size is determined by finding differences between these respective indices. For the figure shown above:

Upper horizontal index=33,

Lower horizontal index=19

Font size=33-19 i.e.'14'.

This technique works well irrespective of the font size and type. We have worked on characters with font sizes 12, 14, 16 and 20. Characters of different font sizes show different characteristics.

CHARACTER PARAMETER RECOGNITION

After obtaining single pixel thickness character by thinning and applying horizontal averaging on them, the matrix thus obtained is taken into account to determine the number of horizontal and vertical strokes based on the pre-defined criteria of the thresholds for the same.

$$\begin{tikzcd}
 & \text{CHARACTER} \arrow[dl] \arrow[d] \arrow[dr] & & \\
\text{Horizontal strokes} & \text{Vertical strokes} & \text{miscellaneous parameters} \arrow[d] \arrow[dr] & \\
 & & \text{Edge detection} & \text{True elements}
\end{tikzcd}$$

VERTICAL AND HORIZONTAL STROKES

In order to recognize vertical and horizontal strokes, the algorithm starts scanning matrix vertically and horizontally respectively.

On finding the first true element while vertical scanning of the matrix, the algorithm searches for consecutive true elements in the same column.

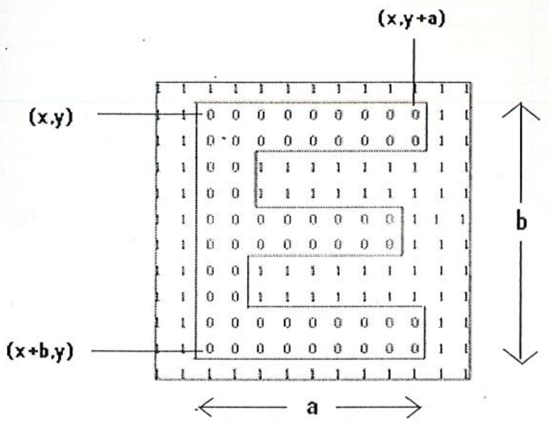


Figure 9: The above character contains '1' vertical and '3' horizontal strokes

If the number of consecutive true elements comes out to be '2' or greater than '2', it is considered as a vertical stroke and algorithm stops scanning in the same column further.

Recognizing a horizontal stroke is a similar process where on finding a true element, the algorithm searches for consecutive true elements in the same row.

CHARACTERISTICS TABLE FOR UPPERCASE ALPHABETS

ALPHABETS	V	H
A	2	7
B	2	3
C	2	2
D	2	2
E	1	3
F	1	2
G	2	2
H	2	1
I	1	0
J	2	1
K	2	7
L	1	1
M	4	6
N	3	5
O	2	2
P	2	2
Q	3	2
R	2	2
S	2	2
T	1	1
U	2	1
V	3	6
W	5	5
X	3	6
Y	3	2
Z	1	2

Table 1: Font size 12

ALPHABETS	V	H
A	3	0
B	2	3
C	2	2
D	2	2
E	1	3
F	1	2
G	2	2
H	2	1
I	1	0
J	2	5
K	3	8
L	1	1
M	4	1
N	4	2
O	2	2
P	2	2
Q	2	2
R	3	6
S	2	2
T	1	1
U	2	1
V	3	3
W	3	5
X	3	4
Y	3	3
Z	3	2

Table 2: Font size 14

ALPHABETS	V	H
A	3	5
B	2	3
C	2	2
D	2	2
E	1	3
F	1	2
G	2	2
H	2	1
I	1	0
J	2	1
K	3	2
L	1	1
M	4	6
N	2	2
O	2	2
P	2	2
Q	2	2
R	2	4
S	2	2
T	1	1
U	1	1
V	3	4
W	5	5
X	3	4
Y	1	1
Z	0	2

Table 3: Font size 16

ALPHABETS	V	H
A	4	1
B	3	3
C	2	2
D	2	2
E	1	3
F	1	2
G	2	2
H	2	1
I	1	0
J	2	1
K	4	3
L	1	1
M	6	3
N	4	6
O	2	2
P	2	2
Q	2	3
R	3	3
S	3	2
T	1	1
U	2	1
V	4	0
W	6	7
X	3	1
Y	4	3
Z	3	2

Table 4: Font size 20

CHARACTERISTIC TABLE FOR LOWERCASE ALPHABETS

ALPHABETS	H	V
a	2	2
b	2	2
c	2	2
d	2	2
e	2	3
f	1	1
g	2	3
h	2	0
i	1	0
j	1	0
k	1	4
l	1	0
m	3	2
n	2	0
o	2	2
p	2	0
q	2	1
r	2	0
s	2	2
t	1	0
u	2	1
v	2	3
w	4	3
x	3	0
y	2	4
z	0	2

Table 5: Font size 16

ALPHABETS	V	H
a	2	2
b	2	2
c	2	2
d	2	2
e	2	3
f	1	1
g	2	3
h	2	1
i	1	0
j	1	0
k	1	3
l	1	0
m	3	2
n	2	0
o	2	2
p	2	1
q	2	2
r	1	1
s	2	2
t	1	2
u	2	1
v	2	4
w	4	3
x	2	4
y	2	4
z	0	2

Table 6:Fontsize14

The characters having unique number of horizontal and vertical strokes are recognized easily by the algorithm, while some of the characters, as seen from the table above, shows similar properties .These characters are further studied on the basis of miscellaneous parameters defined specially for these characters.

CHARACTERS WITH SAME PARAMETERS:

Characters with same parameters	Number of Vertical strokes	Number of Horizontal strokes	Methodology used to differentiate
L,T	1	1	Geometrical Pattern
H,U,J	2	1	Extra elements
C,G,O,P,Q,S,Z	2	2	Edge detection
R,K	3	7	Geometrical Pattern

Table 7: Same parameter for uppercase (Font size 16)

Characters with same parameters	Number of Vertical strokes	Number of Horizontal strokes	Methodology used to differentiate
L,T	1	1	Geometrical Pattern
H,U,J	2	1	Extra elements
C,D,G,O,P,Q,Y,S	2	2	Edge detection
B,N,R	2	3	Geometrical Pattern
A,M	3	5	Geometrical Pattern

Table 8: Same parameter for uppercase (Font size 14)

Characters with same parameters	Number of Vertical strokes	Number of Horizontal strokes	Methodology used to differentiate
L,T	1	1	Geometrical Pattern
Z,F	1	2	Extra elements
H,U,J	2	1	Extra elements
C,D,O,P,R,S,G	2	2	Edge detection
Y,Q	3	2	Geometrical Pattern
V,X	3	6	Geometrical Pattern

Table 9: Same parameter for uppercase (Font size 12)

Characters with same parameters	Number of Vertical strokes	Number of Horizontal strokes	Methodology used to differentiate
L,T	1	1	Geometrical Pattern
H,U,J	2	1	Extra elements
C,D,O,P,G	2	2	Edge Detection
Z,S	3	2	Geometrical Pattern
B,R	3	3	Edge Detection
K,Y	4	3	Number of True elements

Table 10: Same parameter for uppercase (Font size 20)

MISCELLANEOUS PARAMETERS

EDGE DETECTION

This parameter takes into account the variation in number of non true elements in the respective columns of the two character matrices. The edge lengths of particular column are stored in separate matrix, and then the comparative analysis of the same gives the distinguishing parameter.

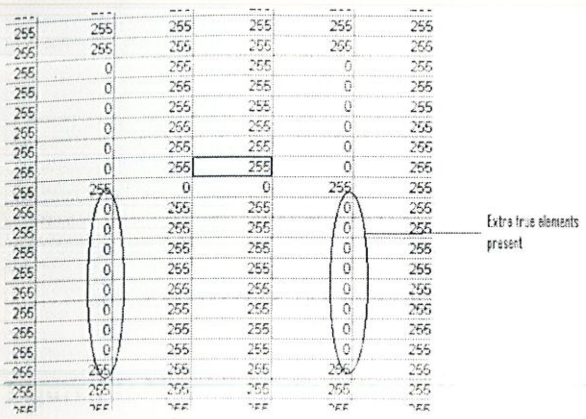


Figure 10: Matrix of character ‘H’

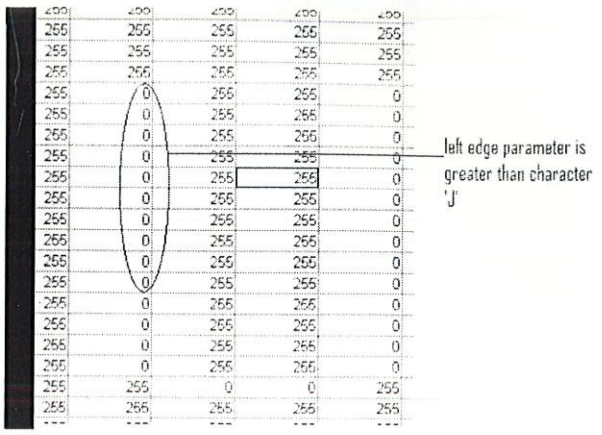


Figure 11: Matrix of character ‘U’

It is crystal clear from the above images that both ‘H’ and ‘U’ consist of one horizontal line. They can be differentiated on the account that “extra true elements” are present in the image of ‘H’, which is absent in the case of ‘U’

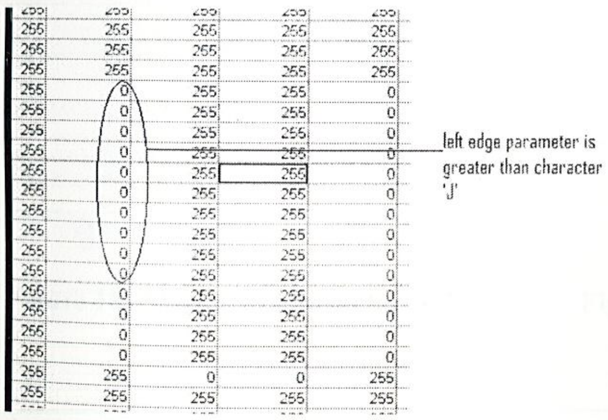


Figure 12: Matrix of character ‘U’

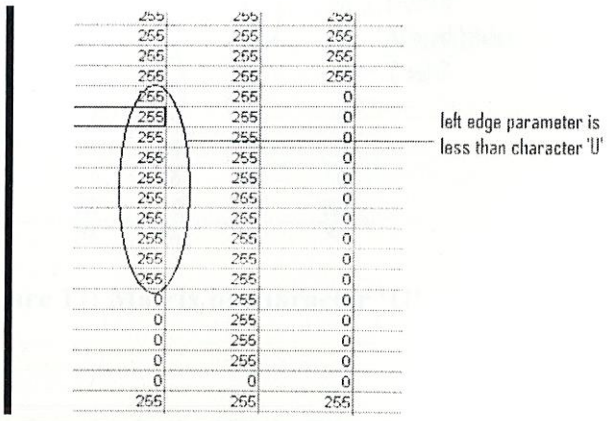
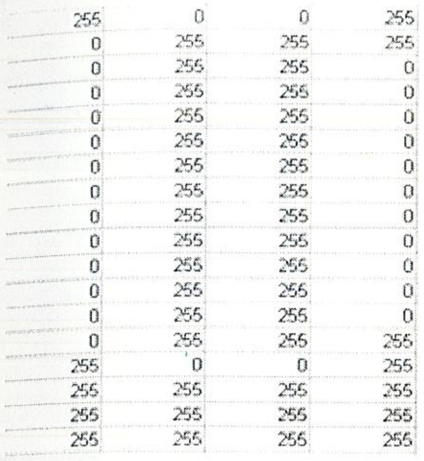


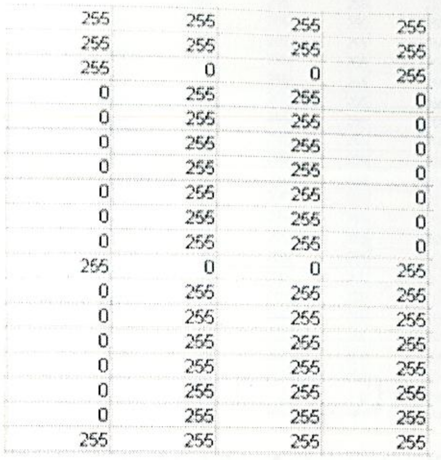
Figure 13: Matrix of character ‘J’

'U' and 'J' can be distinguished on the basis of "Edge Parameter". On observing the above images, it is discovered that "Left Edge Parameter" of 'J' is small than that of 'U', which makes a good distinctive feature.



255	0	0	255
0	255	255	255
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	255
255	0	0	255
255	255	255	255
255	255	255	255
255	255	255	255

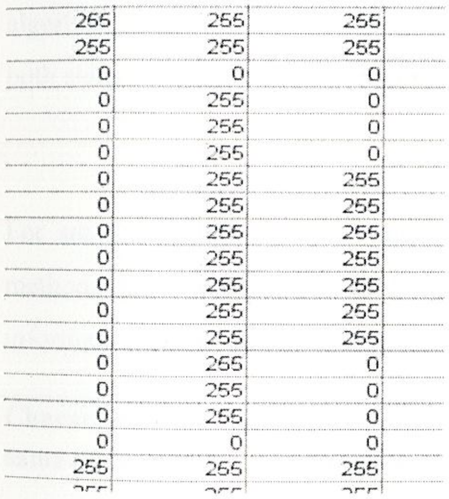
Figure 14: Matrix of character 'D'



255	255	255	255
255	255	255	255
255	0	0	255
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
255	0	0	255
0	255	255	255
0	255	255	255
0	255	255	255
0	255	255	255
0	255	255	255
0	255	255	255
255	255	255	255

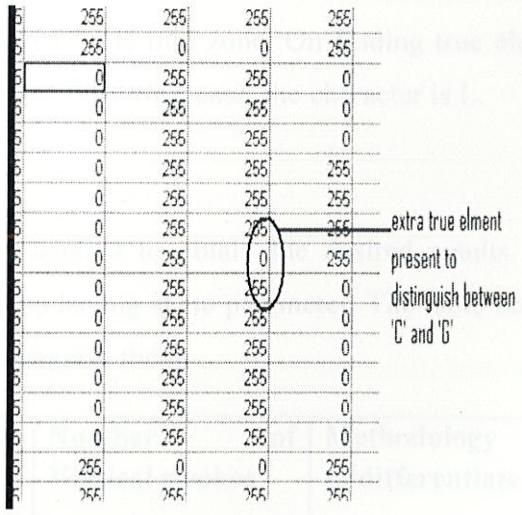
Figure 15: Matrix of character 'P'

The same holds true for the characters 'C' and 'G'.



255	255	255
255	255	255
0	0	0
0	255	0
0	255	0
0	255	0
0	255	255
0	255	255
0	255	255
0	255	255
0	255	255
0	255	255
0	255	255
0	255	0
0	255	0
0	255	0
0	0	0
255	255	255
255	255	255

Figure 16: Matrix of character 'C'



255	255	255	255
255	0	0	255
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	255
0	255	255	255
0	255	255	255
0	255	0	255
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
0	255	255	0
255	0	0	255
255	255	255	255

extra true elment present to distinguish between 'C' and 'G'

Figure 17: Matrix of character 'G'

As shown in the figure above, the number of true elements in the "Right Edge Parameter" serves as distinctive feature.

To distinguish between characters T and L

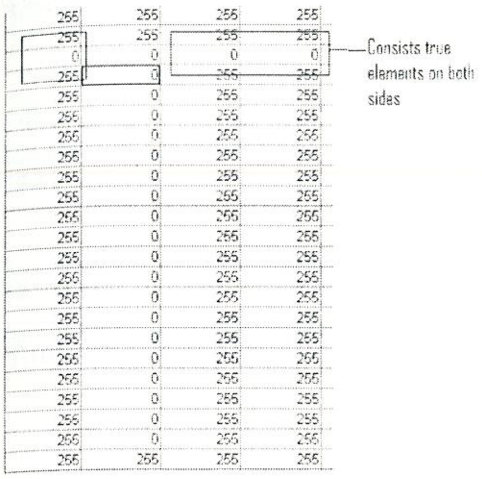


Figure 18: Matrix of character 'L'

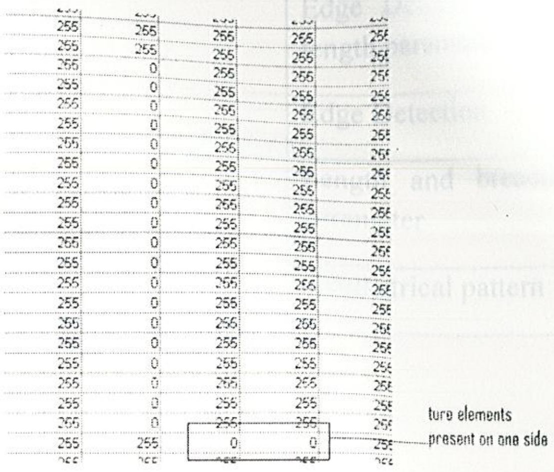


Figure 19: Matrix of character 'T'

To differentiate between 'T' and 'L' the mid zone i.e. crossing of the two strokes is identified and the algorithm searches for true elements on either side of that mid zone. On finding true elements on both sides the character, it is identified as T and in the otherwise case, the character is L.

For small alphabets, similar approach has been applied to obtain the desired results. Different methods have been applied to distinguish characters having same parameter. The table below gives information about different methods applied to distinguish them.

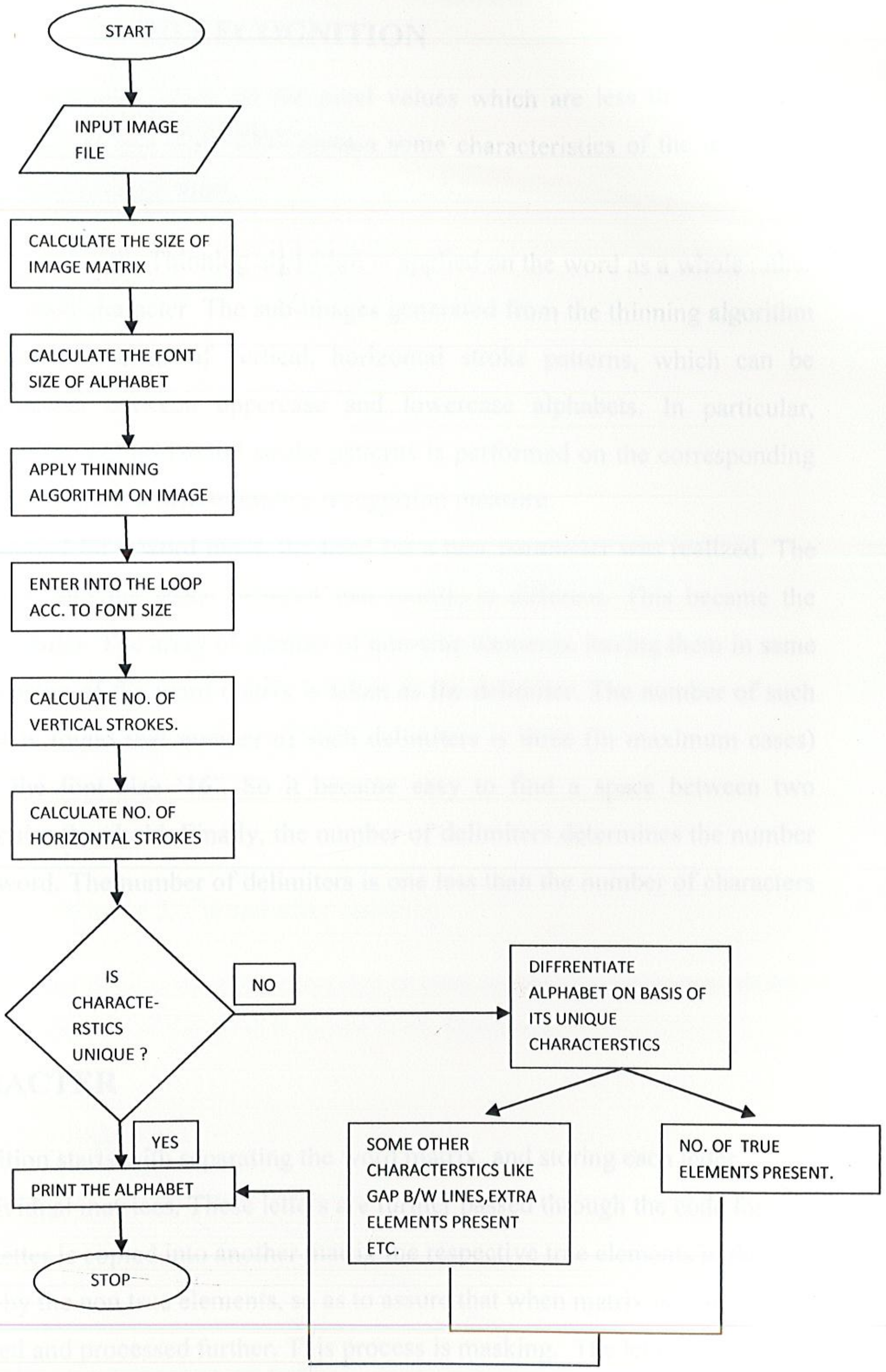
Characters with same parameters	Number of Horizontal Strokes	Number of Vertical strokes	Methodology used to differentiate
a,b,c,d,o,s	2	2	Edge Detection and length parameter
e,g,v	3	2	Edge Detection
h,n,p,r	0	2	Geometrical Pattern
i,j,l,t	0	1	Length and breadth parameter

Table 11: Same parameter for lowercase (Font size 16)

Characters with same parameters	**Number of Horizontal Strokes**	**Number of Vertical strokes**	**Methodology used to differentiate**
a,b,c,d,o,s,q	2	2	Edge Detection and length parameter
e,g	3	2	Edge Detection
i,j,l	0	1	Length and breadth parameter
r,f	1	1	Geometrical pattern

Table 12: Same parameter for lowercase (Font size 14)

FLOWCHART EXPLANATIONS



START
INPUT IMAGE FILE
CALCULATE THE SIZE OF IMAGE MATRIX
CALCULATE THE FONT SIZE OF ALPHABET
APPLY THINNING ALGORITHM ON IMAGE
ENTER INTO THE LOOP ACC. TO FONT SIZE
CALCULATE NO. OF VERTICAL STROKES.
CALCULATE NO. OF HORIZONTAL STROKES
IS CHARACTE-RSTICS UNIQUE ?
NO
DIFFRENTIATE ALPHABET ON BASIS OF ITS UNIQUE CHARACTERSTICS
SOME OTHER CHARACTERSTICS LIKE GAP B/W LINES,EXTRA ELEMENTS PRESENT ETC.
NO. OF TRUE ELEMENTS PRESENT.
YES
PRINT THE ALPHABET
STOP

Figure 20: Flowchart for Character Recognition

CHAPTER-4

WORD RECOGNITION

Extending our work from characters to word, all the pixel values which are less than '255' are represented by '0'. The pixel values less than '255' contain some characteristics of the individual characters, and thus it is important to retain them.

After converting the pixel value into '0', Thinning algorithm is applied on the word as a whole rather than applying individually on each character. The sub-images generated from the thinning algorithm step contain ample information in terms of vertical, horizontal stroke patterns, which can be effectively utilized to distinguish between uppercase and lowercase alphabets. In particular, statistical analysis of both vertical and horizontal stroke patterns is performed on the corresponding sub-images and is combined to produce a discriminative recognition measure.

On studying the patterns obtained for a word input, the need for a new parameter was realized. The space between two characters, and the space between two words, is different. This became the baseline to think of a new delimiter. The array of number of non-true elements, having them in same number, as the vertical dimension of the word matrix is taken as the delimiter. The number of such delimiters is counted, and it is found that number of such delimiters is three (in maximum cases) between two characters of the font size '16'. So it became easy to find a space between two characters by setting a particular threshold. Finally, the number of delimiters determines the number of characters present in the word. The number of delimiters is one less than the number of characters present in the word.

MASKING OF CHARACTER

The process of word recognition starts with separating the word matrix, and storing each letter present in the word into individual matrices. These letters are further passed through the code for their recognition. Once the letter is copied into another matrix the respective true elements in the original matrix are replaced by the non true elements, so as to assure that when matrix is scanned next time, next letter is copied and processed further. This process is masking. The letters within a word are recognized simultaneously, and the letter information is used to recognize the words.

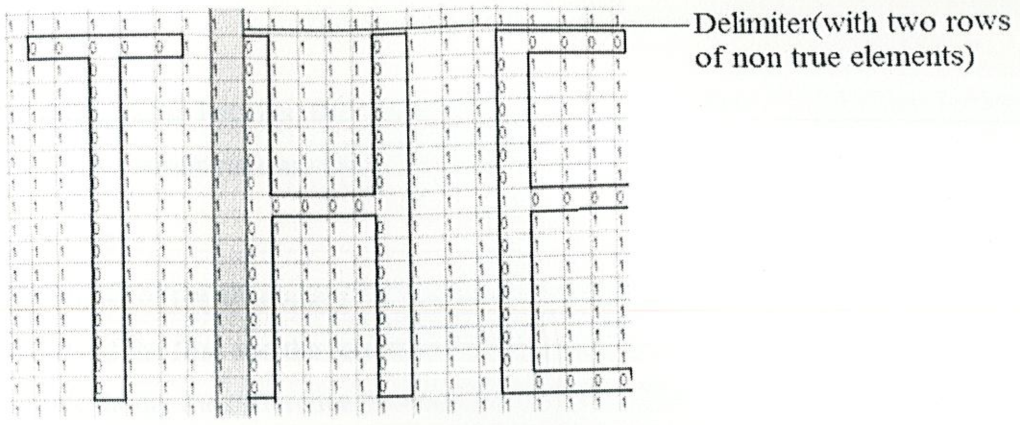


Figure 21: Word before masking

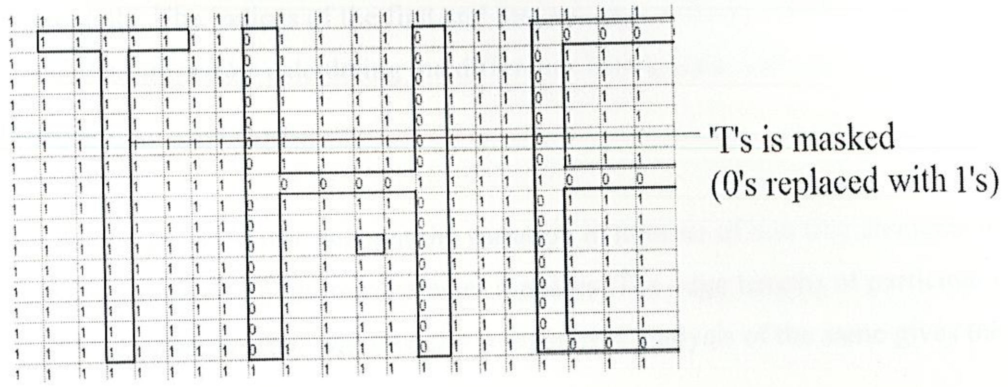


Figure 22: Word after masking

All the characters recognized individually, are now segregated in their recognition order to form the word. The diagrammatic representation of the same is shown in the figure below.

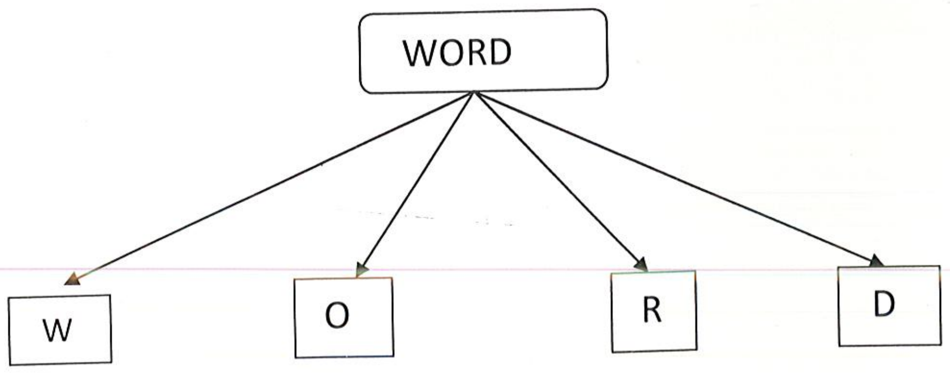


Figure 23: Separation of words into characters

PARAMETERS

In order to differentiate between the characters having same number of vertical and horizontal strokes, following parameters are taken into consideration.

1. **LENGTH**

   The length of the character is determined by scanning the character matrix vertically. The indices of the first and the last 'true element' are thus obtained. The length is then determined by calculating the difference between the indices of first and last 'true element'.
2. **WIDTH**

   Similarly, the width of the character is determined by scanning the character matrix horizontally. The indices of the first and the last 'true element' are thus obtained. The width is then determined by calculating the difference between the indices of first and last 'true element'.
3. **EGDES**

   This parameter takes into account the variation in number of non true elements in the respective columns of the two character matrices. The edge lengths of particular columns are stored in separate matrix, and then the comparative analysis of the same gives the distinguishing parameter.
4. **NUMBER OF ZEROS**

   This parameter takes into account of number of true elements present in a particular character.

CHARACTERISTICS TABLE FOR WORD RECOGNITION

VERTICAL	HORIZONTAL	CHARACTERS
1	0	I , t, i ,j ,e
1	1	T, L, r
1	2	f ,F ,z
1	3	E
2	1	H ,U ,J ,h ,n ,q ,u
2	2	O ,G ,C ,Q ,Z ,P , S ,o ,a ,s ,c ,d ,b
2	3	e ,g ,v ,p
2	4	Y ,y
2	6	K
3	0	X ,x
3	1	M ,m
3	2	A ,D
3	3	B
3	4	N
3	5	V
3	7	K ,R
4	3	W
4	8	W

Table 13: Parameters for 'multiple characters'

IMPLEMENTATION OF PARAMETERS

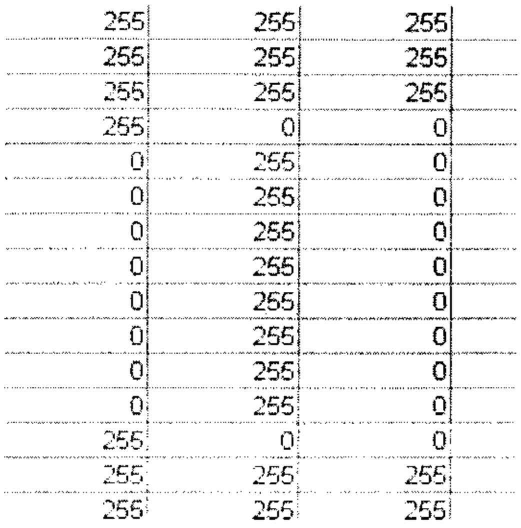
As seen from the above table, there are some characters which show the same number of 'horizontal and vertical' strokes. All these characters have '2' vertical and '2' horizontal strokes respectively.

The uppercase letters showing these characteristics are 'O', 'G', 'C', 'Q', 'Z', 'P', 'S'.

The lowercase letters showing this characteristic are 'o', 'a', 's', 'c', 'd', 'b'.

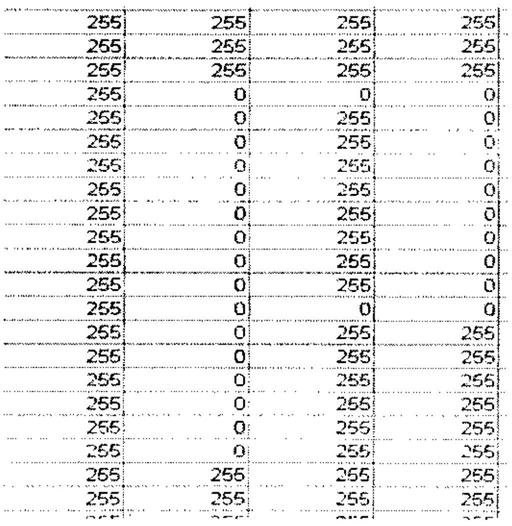
Here we describe the methodology that we have incorporated to distinguish these characters. Again, we have exploited the use of our four parameters (length, width, edges and number of zeros) to do the same, that we have described earlier.

Variation in the length between some of the above characters has been taken into consideration for their recognition. The characters like 'o', 'a', 's', 'c 'has small length as compared to others.



255	255	255
255	255	255
255	255	255
255	0	0
0	255	0
0	255	0
0	255	0
0	255	0
0	255	0
0	255	0
0	255	0
0	255	0
255	0	0
255	255	255
255	255	255

Figure 24: Matrix of character 'o'



255	255	255	255
255	255	255	255
255	255	255	255
255	0	0	0
255	0	255	0
255	0	255	0
255	0	255	0
255	0	255	0
255	0	255	0
255	0	255	0
255	0	255	0
255	0	255	0
255	0	0	0
255	0	255	255
255	0	255	255
255	0	255	255
255	0	255	255
255	0	255	255
255	0	255	255
255	255	255	255
255	255	255	255

Figure 25: Matrix of character 'P'

In the above figure, the length of the character 'o' is less than that of character 'P'. Thus, on calculating length parameter we can differentiate characters.

Then applying the edge parameter, we calculate the edge length of the character which helps us to differentiate between characters which have same parameters (horizontal strokes, vertical strokes, length).

255	255	255
255	255	255
255	255	255
255	0	0
0	255	0
0	255	0
0	255	0
0	255	0
0	255	0
0	255	0
0	255	0
0	255	0
255	0	0
255	255	255
255	255	255

Figure 26: Matrix of character 'O'

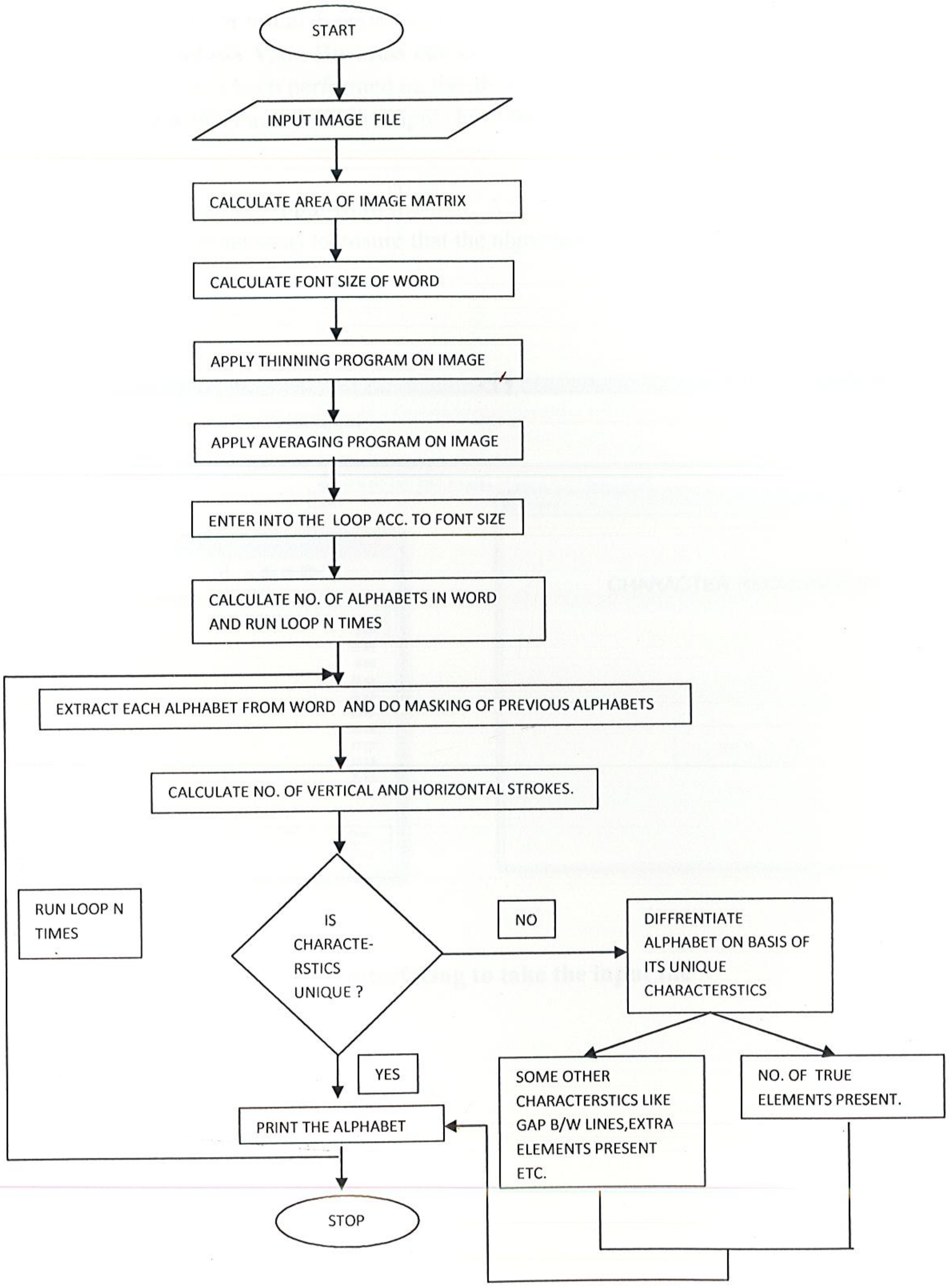
255	255	255
255	0	0
0	255	0
0	255	0
0	255	0
0	255	255
0	255	255
0	255	255
0	255	0
0	255	0
255	0	0
255	255	255
255	255	255

Figure 27: Matrix of character 'C'

After calculation of 'edge parameter' we can distinguish characters on the basis of 'left edge length' and 'right edge length'. For example in character 'c' and 'o' the left edge length is same but the right edge length of 'o' is greater than that of 'c'. Thus by calculating edge length the geometrical pattern of different characters can be recognized, which helps to distinguish between different characters.

If some of the characters that cannot be distinguish on the basis of above parameters then 'zeros parameter' can be used to do the same. Zeros parameters calculates the number of true elements present in a particular character. This parameter helps us to distinguish characters like 'O' and 'Q' which has almost same geometrical representation.

FLOWCHART EXPLANATIONS



START
INPUT IMAGE FILE
CALCULATE AREA OF IMAGE MATRIX
CALCULATE FONT SIZE OF WORD
APPLY THINNING PROGRAM ON IMAGE
APPLY AVERAGING PROGRAM ON IMAGE
ENTER INTO THE LOOP ACC. TO FONT SIZE
CALCULATE NO. OF ALPHABETS IN WORD AND RUN LOOP N TIMES
EXTRACT EACH ALPHABET FROM WORD AND DO MASKING OF PREVIOUS ALPHABETS
CALCULATE NO. OF VERTICAL AND HORIZONTAL STROKES.
RUN LOOP N TIMES
IS CHARACTE-RSTICS UNIQUE ?
NO
DIFFRENTIATE ALPHABET ON BASIS OF ITS UNIQUE CHARACTERSTICS
YES
PRINT THE ALPHABET
SOME OTHER CHARACTERSTICS LIKE GAP B/W LINES,EXTRA ELEMENTS PRESENT ETC.
NO. OF TRUE ELEMENTS PRESENT.
STOP

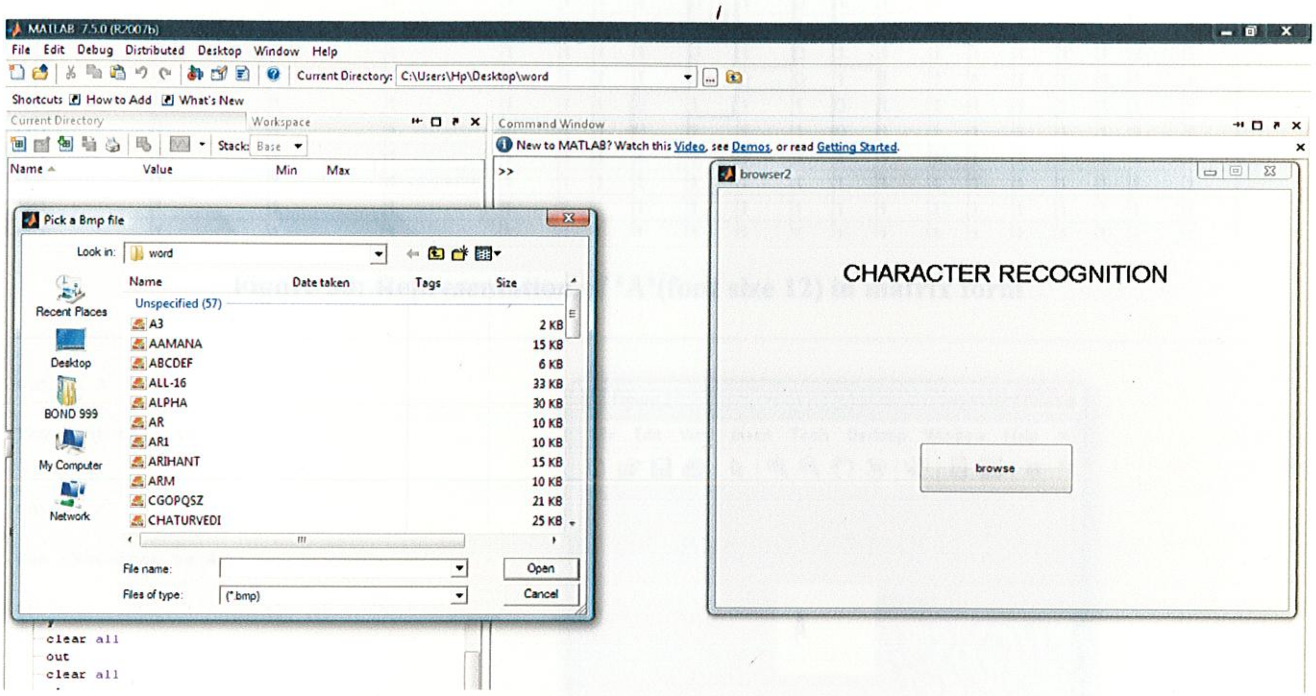
Figure 28: Flowchart for word recognition

CHAPTER-5

OUTPUT OF ALGORITHM ON MATLAB PLATFORM

Figure represents an output of or initial experimentation where symmetric images were drawn in **Mspaint** of **Microsoft Windows Vista Business** edition operating system. It is to be noted that all the experiments thereafter have been performed on the above mentioned OS and the software used is –**MATLAB version 7.1.0.246 Pack 3.**All the inputs have been accepted in monochrome bitmap (.bmp)file format.

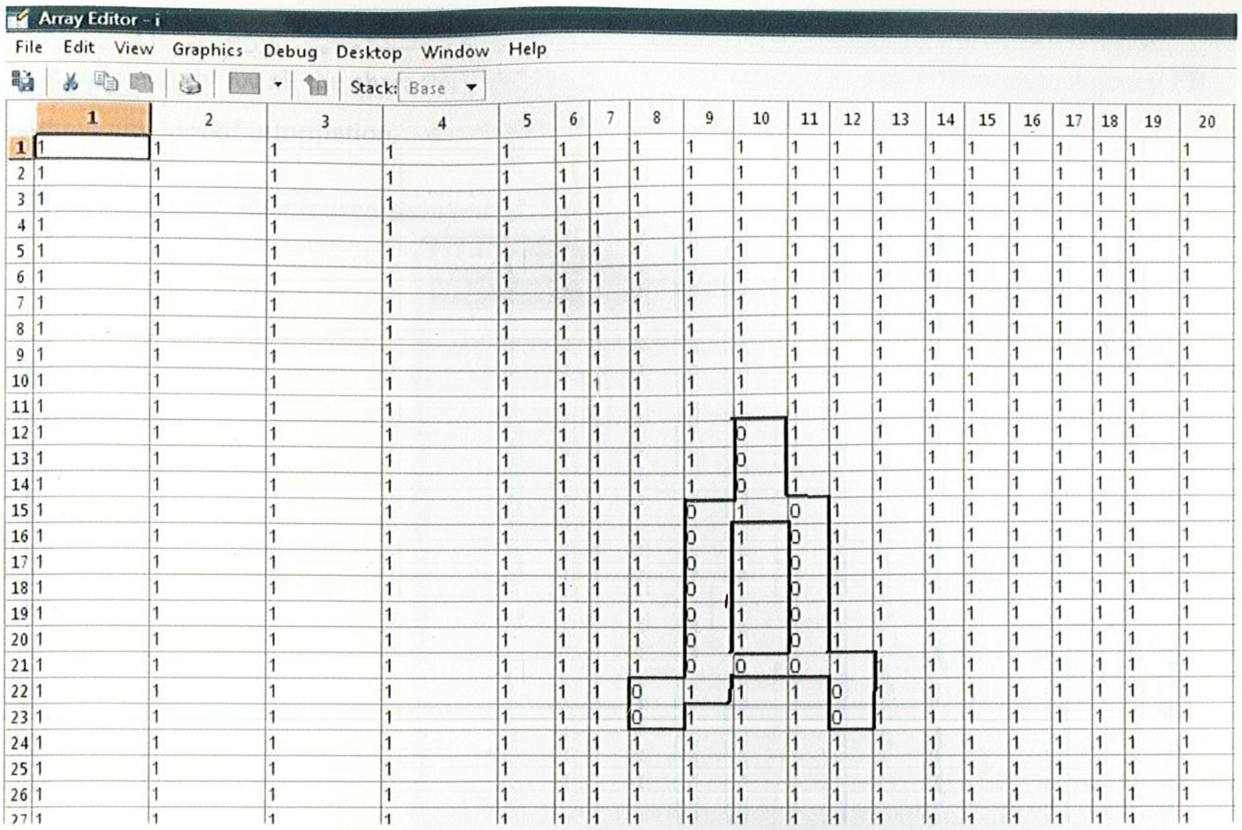
GUI interfacing is done to take the input from the user. Any file stored anywhere in the computer is browsed and connectivity is done so as to ensure that the algorithm takes the browsed file as input.



Name	Date taken	Tags	Size
Unspecified (57)			
A3			2 KB
AAMANA			15 KB
ABCDEF			6 KB
ALL-16			33 KB
ALPHA			30 KB
AR			10 KB
AR1			10 KB
ARIHANT			15 KB
ARM			10 KB
CGOPQSZ			21 KB
CHATURVEDI			25 KB

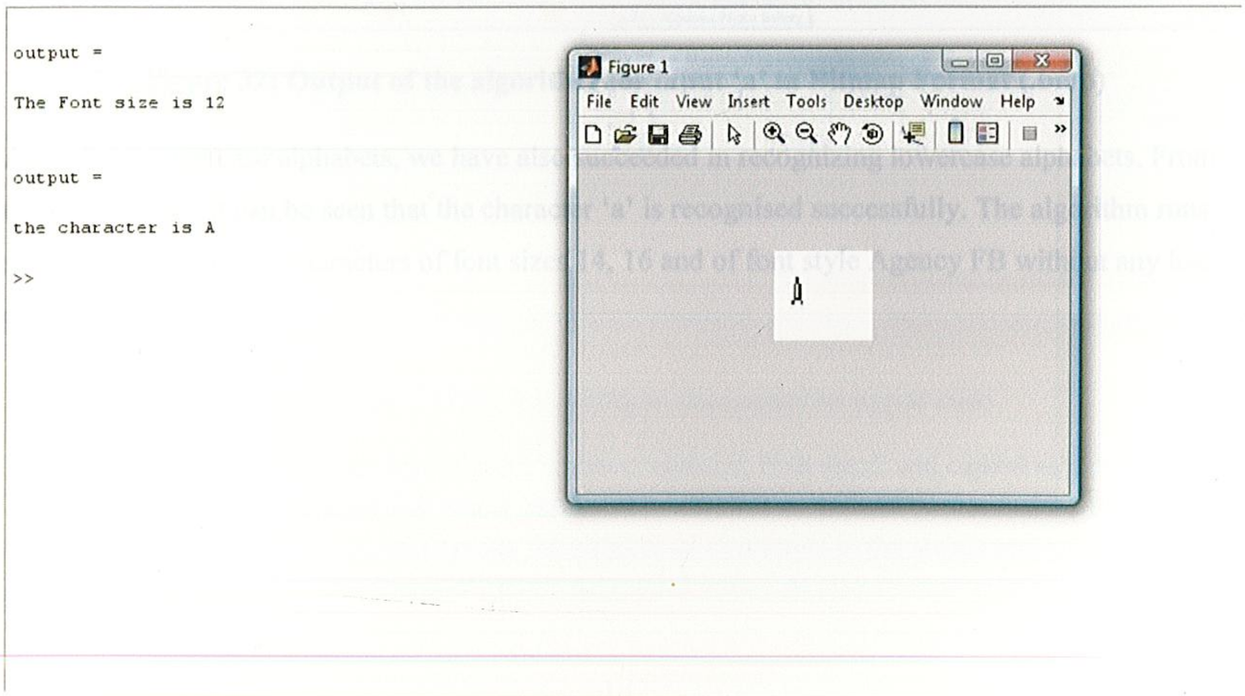
```
clear all
out
clear all
```

Figure 29: GUI interfacing to take the input file



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	1	1	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
12	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1
17	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1
21	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
26	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 30: Representation of 'A'(font size 12) in matrix form



```
output =

The Font size is 12


output =

the character is A

>>
```

Figure 31: Output of the algorithm for input 'A' in Bitmap Format (.bmp)

From the above figure it can be seen that the character 'A' is recognised successfully. The algorithm runs successfully for all the characters of font sizes 12, 14, 16 and 20 and of font style Agency FB without any loss of information.

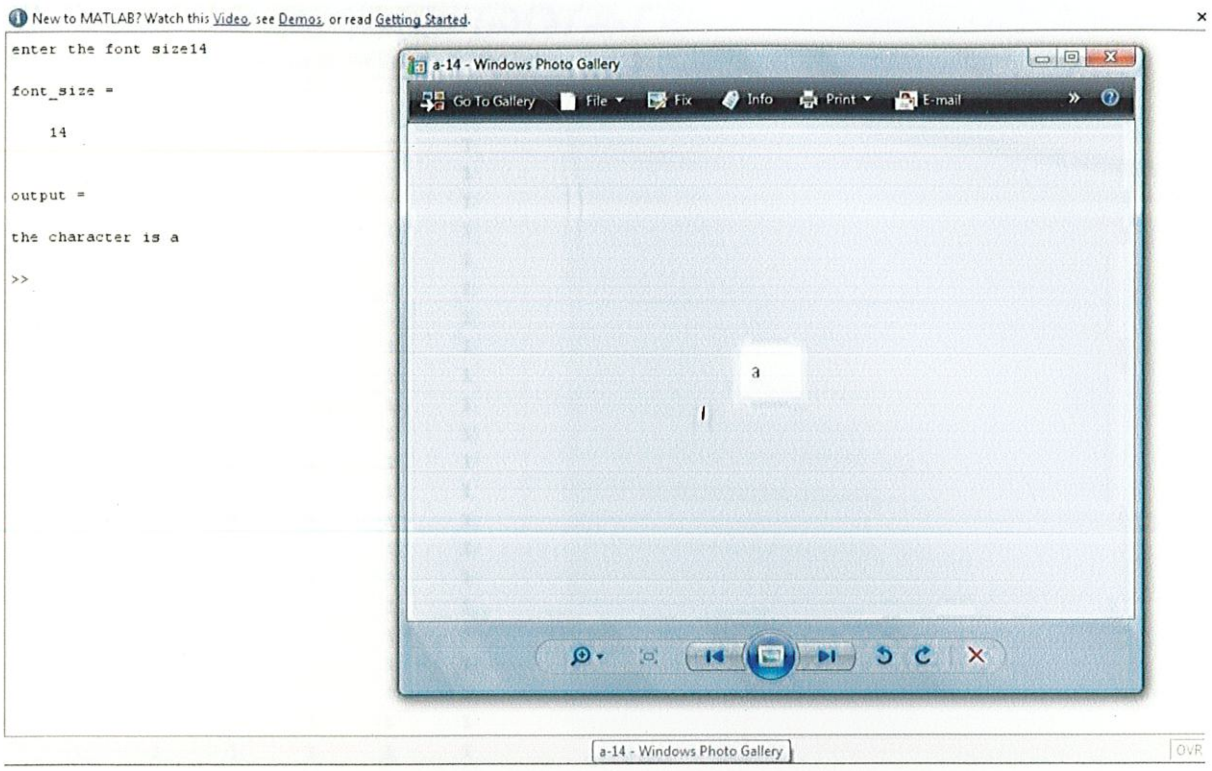
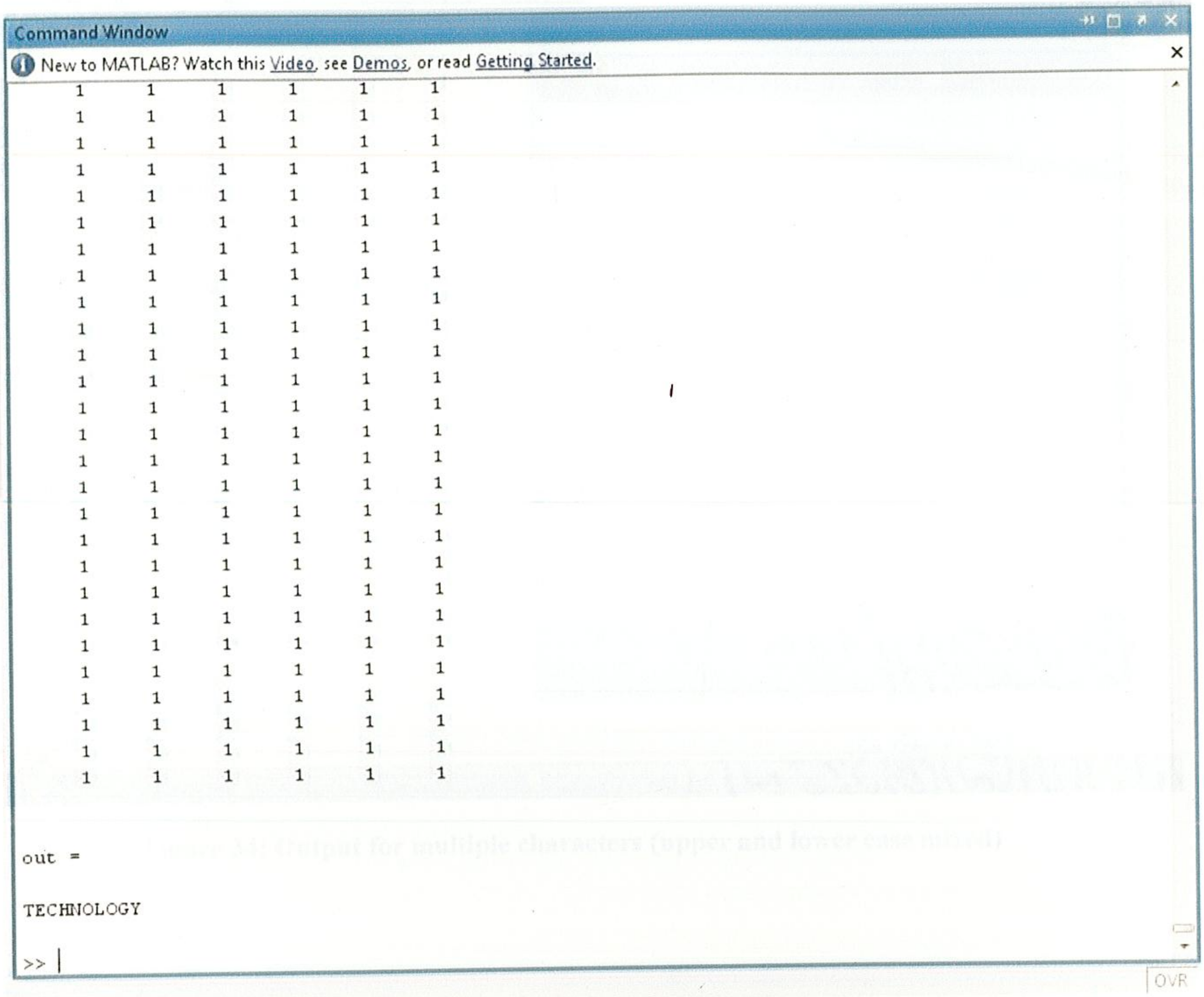

```
enter the font size14

font_size =

    14


output =

the character is a

>>
```
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
a-14 - Windows Photo Gallery
Go To Gallery
File
Fix
Info
Print
E-mail
a
a-14 - Windows Photo Gallery
OVR

Figure 32: Output of the algorithm for input 'a' in Bitmap Format (.bmp)

Apart from uppercase alphabets, we have also succeeded in recognizing lowercase alphabets. From the above figure it can be seen that the character 'a' is recognised successfully. The algorithm runs successfully for all the characters of font sizes 14, 16 and of font style Agency FB without any loss of information.

WORD RECOGNITION

Firstly, the file containing a word written in capital letters is taken as input and the algorithm runs successfully for the same. The output is shown in the figure below.

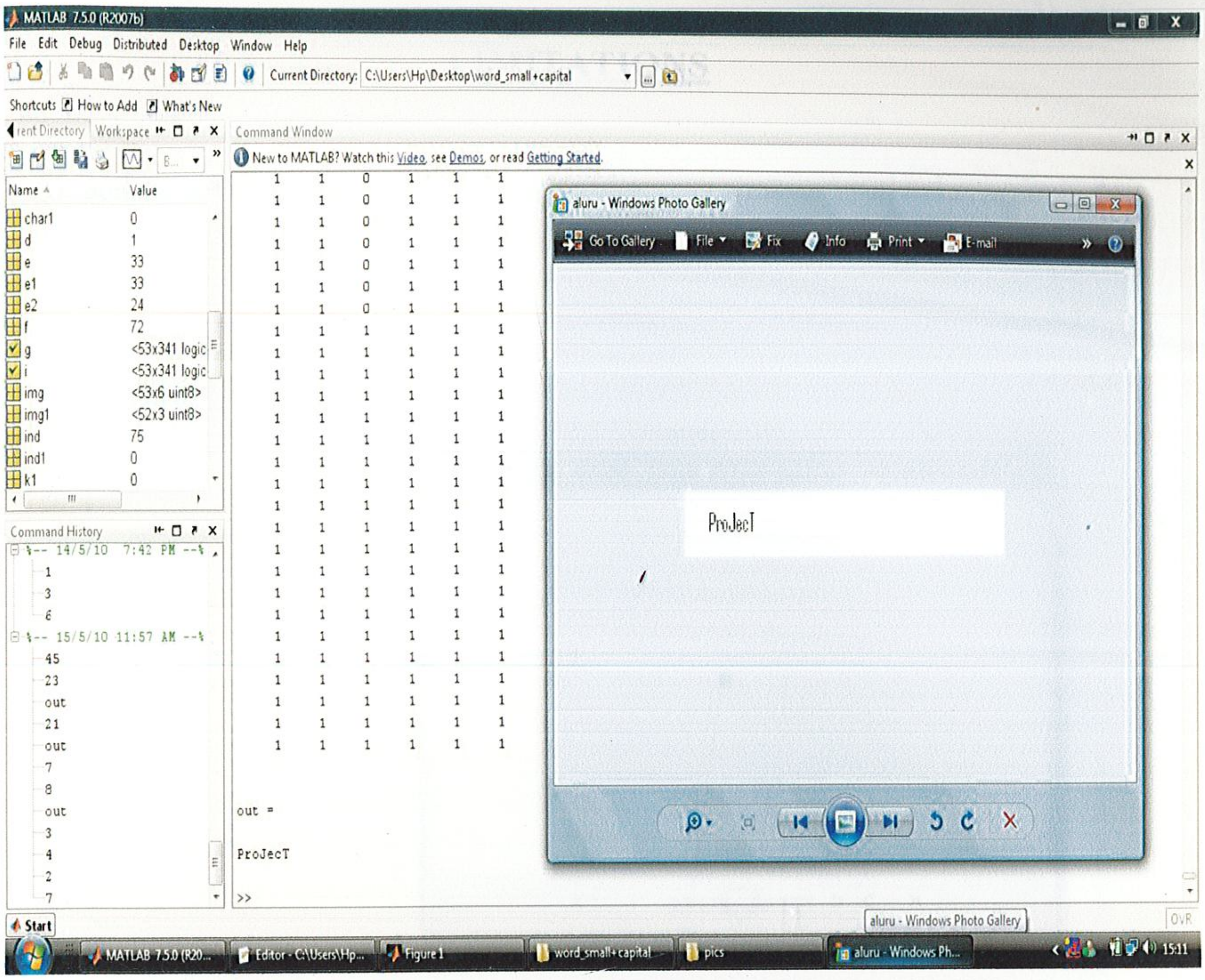


Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
     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
     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
     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
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1


out =

TECHNOLOGY

>>
```

Figure 33: Output for multiple characters in upper case

Further the work has been extended for the words containing both small and capital alphabets. Figure shows the output generated for the 'word' input. The delimiter as defined earlier is the space between two characters. The algorithm first counts the number of characters in the words and then the algorithm runs for the same number of times. Any word with font size 16 is recognised successfully by the algorithm.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: C:\Users\Hp\Desktop\word_small+capital
Shortcuts How to Add What's New
rent Directory Workspace
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Command History
aluru - Windows Photo Gallery
Go To Gallery File Fix Info Print E-mail
ProJecT
Start
MATLAB 7.5.0 (R20... Editor - C:\Users\Hp... Figure 1 word_small+capital pics aluru - Windows Ph...
15:11

Name	Value
char1	0
d	1
e	33
e1	33
e2	24
f	72
g	<53x341 logic
i	<53x341 logic
img	<53x6 uint8>
img1	<52x3 uint8>
ind	75
ind1	0
k1	0

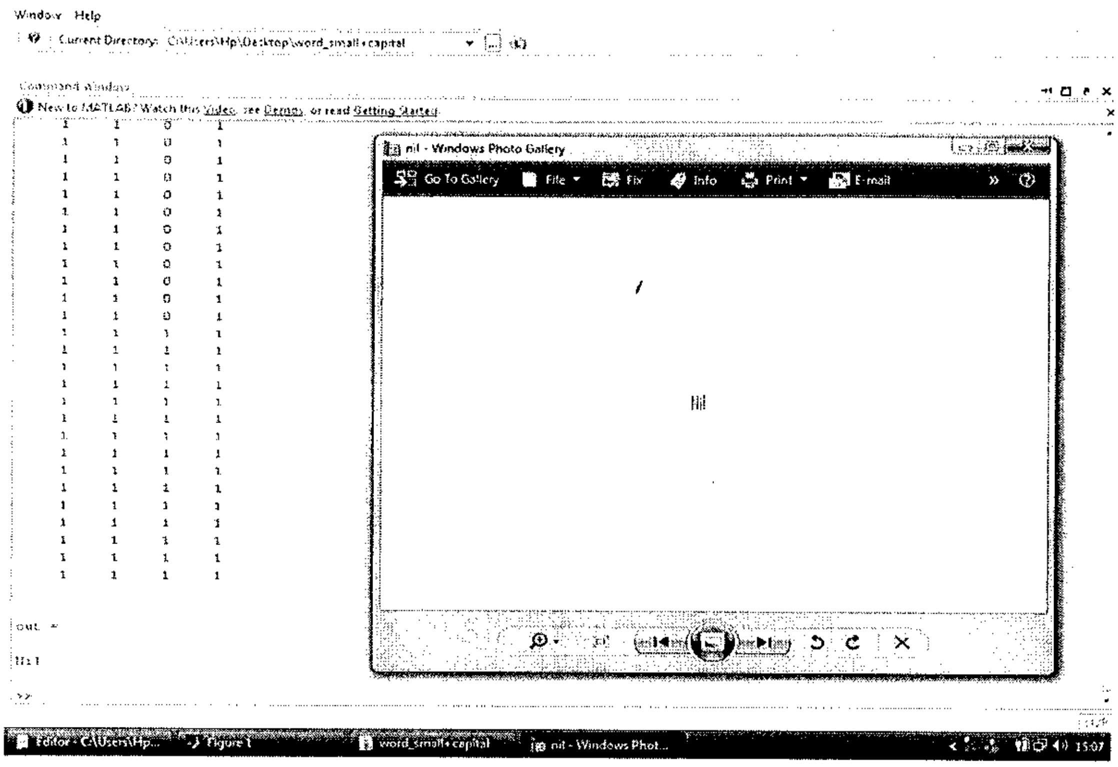
```
%-- 14/5/10  7:42 PM --%
  1
  3
  6
%-- 15/5/10 11:57 AM --%
  45
  23
  out
  21
  out
  7
  8
  out
  3
  4
  2
  7
```

```
     1     1     0     1     1     1
     1     1     0     1     1     1
     1     1     0     1     1     1
     1     1     0     1     1     1
     1     1     0     1     1     1
     1     1     0     1     1     1
     1     1     0     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     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     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


out =

ProJecT

>>
```

Figure 34: Output for multiple characters (upper and lower case mixed)

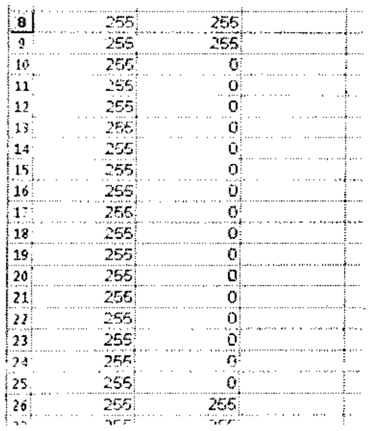
LIMITATIONS

1) Some of the characters like 'I' and 'l' did not show any distinctive feature on any of the four parameters (length, width, edge and no. of true elements).


Window Help
Current Directory: C:\Users\Hp\Desktop\word_small+capital
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
nil - Windows Photo Gallery
Go To Gallery
File
Fix
Info
Print
E-mail
Editor - C:\Users\Hp...
Figure 1
word_small+capital
nil - Windows Phot...
15:07

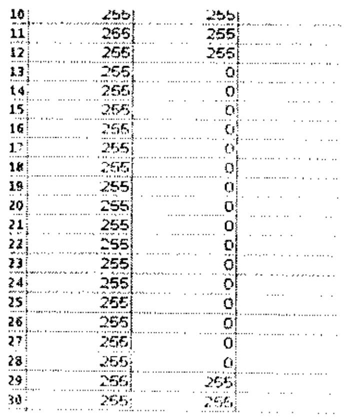
```
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     1
     1     1     0     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     1
     1     1     1     1
     1     1     1     1
     1     1     1     1

out =

lil

>>
```

Figure 35: Output showing 'I' instead of 'l'



8	255	255
9	255	255
10	255	0
11	255	0
12	255	0
13	255	0
14	255	0
15	255	0
16	255	0
17	255	0
18	255	0
19	255	0
20	255	0
21	255	0
22	255	0
23	255	0
24	255	0
25	255	0
26	255	255

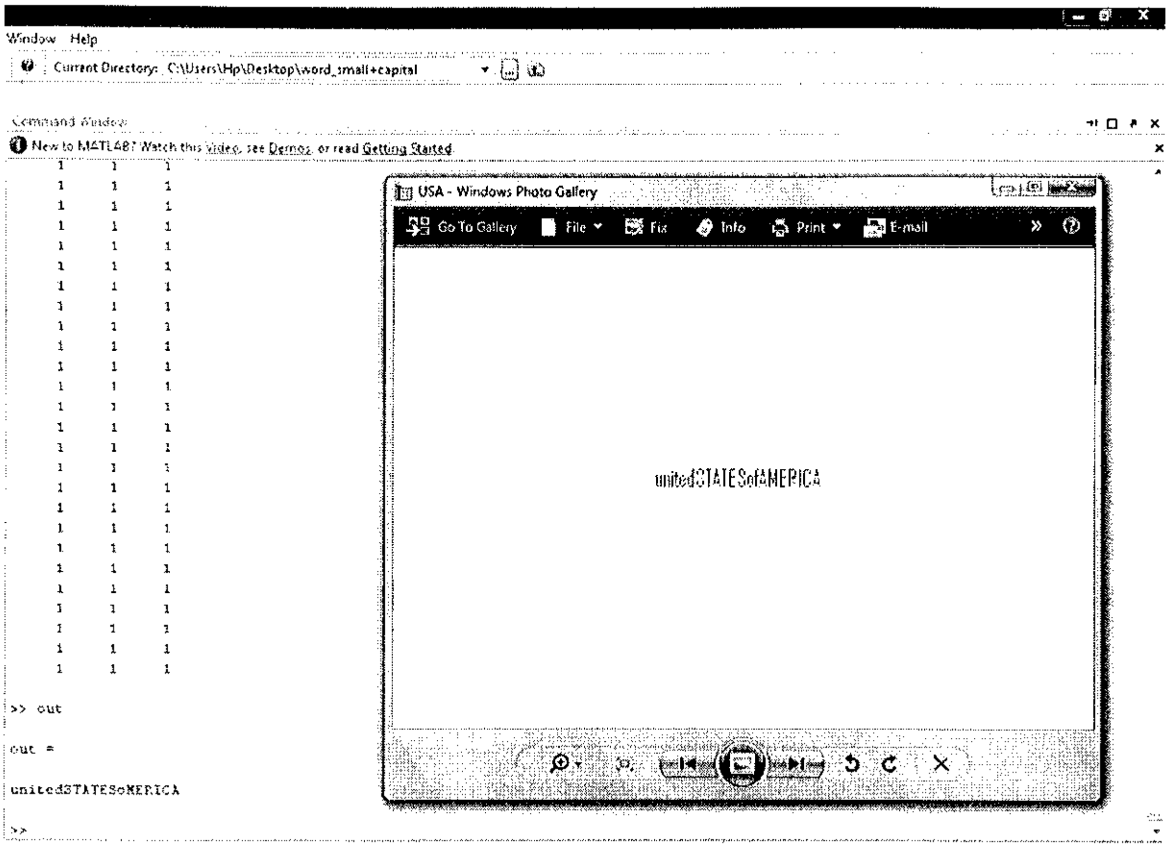
Figure 36: Matrix of character 'I'



10	255	255
11	255	255
12	255	255
13	255	0
14	255	0
15	255	0
16	255	0
17	255	0
18	255	0
19	255	0
20	255	0
21	255	0
22	255	0
23	255	0
24	255	0
25	255	0
26	255	0
27	255	0
28	255	0
29	255	255
30	255	255

Figure 37: Matrix of character 'l'

2) The character 'V' could not be recognized in font size '16' in a word.
3) Recognition of space between two words also remains a limitation of algorithm designed, space has been neglected when output is printed
4) In some of the cases, there were no delimiters obtained between two characters. Consequently, those characters were not recognized by our proposed algorithm.


Window Help
Current Directory: C:\Users\Hp\Desktop\word_small+capital
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
USA - Windows Photo Gallery
Go To Gallery
File
Fix
Info
Print
E-mail
unitedSTATESofAMERICA

```
     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
     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

>> out

out =

unitedSTATESoMERICA

>>
```

Figure 38: Example showing word without delimiter

RESULT

Experiments carried out to test the proposed method with 26 upper case and 26 lower case English alphabets shows positive results without any loss of information. The success of the project work lies in the high levels of accuracy obtained when the experiments are performed for the 'word' recognition. We have successfully recognised words containing lower case and upper case English alphabets separately. Word containing mixed upper and lower case English alphabets is also successfully recognized. It is to be noted that once an alphabet is recognised correctly, it will always be recognized on infinite repeats and therefore the method of accuracy checking, i.e., counting the number of errors per 1000 trials, has not been used.

Font Size	Upper case Alphabets detected	Lower case Alphabets detected	% accuracy	Alphabets wrongly detected
12	26	-	100	None
14	26	26	100	None
16	26	26	98.1	V
20	26	-	100	None

Table 14: Results Table

CONCLUSION

The project has been successfully completed within the stipulated time frame .We have achieved 100% accuracy on all results and have documented the entire concept under a research paper titled "Character recognition using Geometrical Analysis". It is likely to be published soon. The future implications of this innovative approach seem bright. The concept can be extended to recognition of sans serif font other than Agency FB, serif fonts and hand written texts. The final objective, as we realized, is recognition of a full document .This vision is promising and possible.

BIBLIOGRAPHY

BOOKS:

- T. Keith, *calligraphic Tendencies in the Development of Sanserif Types in the* 20th Century University of Reading. Berkshire 2002

WEB PAGES:

- www.homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm
- www.csharpcorner.com/UploadFile/hscream/ImagetoBinary
- www.google.com
- www.wikipedia.com

RESEARCH PAPERS:

- S. Khoubyari and J.J. Hull, Font and Function Word Identi£cation in Document Recognition, *Computer Visionand Image Understanding*, vol. 63, no.1, pp. 66-74, 1996.
- J. Rocha and T. Pavlidis, "A shape analysis model with application to a character recognition system," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 16, pp. 394–404,1994
- S. Mori, "A non-metric model of handprinted characters," *Res. Electrotech. Lab. Tokyo*, vol. 798, August 1979
- C. Y. Suen, "Character recognition by computer and application," *in Handbook of Pattern recognition and Image Proceeding*, pp. 569–586, 1986.
- Y. Zhu, T.N. Tan, Y.H. Wang, Font Recognition Based on Global Texture Analysis, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, 2001.