CHARACTER RECOGNITION USING GEOMETRICAL ANALYSIS

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology

in

Electronics and Communication Engineering

under the Supervision of

Prof. Sunil V. Bhooshan

By

Rachit Khanna (081028)

Naveen Kr. Yadav (081070)

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

CERTIFICATE

This is to certify that project report entitled "CHARACTER RECOGNITION USING GEOMETRICAL ANALYSIS", submitted by **Rachit Khanna** and **Naveen Yadav** in partial fulfillment for the award of degree of Bachelor of Technology in Electronics and Communication Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

SBhooshan

Date: 2nd JUNE, 2012

Prof. Sunil V. Bhooshan

(Head of Department)

(Electronics and Communication)

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 express our gratitude and sincere thanks to **Prof. Sunil V. Bhooshan** (Head of Department, Electronics and Communication) who laid the stepping-stone to our quest of knowledge. He had the foresight to the entire project and has been the source of inspiration and motivation.

We also thank **Dr. Vinay Kumar** (Asst. Prof., Electronics and Communication Department) for providing us the opportunity to undertake the project under his able guidance. He helped us develop novel solutions to every problem and emerge with good engineering acumen.

Rachit

Rachit Khanna

(081028)

Naveen Yadav

Naveen Kr. Yadav

(081070)

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF SYMBOLS AND ACRONYMS

OCR – Optical Character Recognition

ICR – Intelligent Character Recognition

OMR – Optical Mark Recognition

ASCII – American Standard Code for Information Interchange

XML – Extensible Markup Language

TE – True Elements

NTE – Non True Elements

ABSTRACT

Optical Character Recognition (OCR) is the mechanical or electronic translation of images of handwritten, typewritten or printed text (usually captured by a scanner) into machine-editable text. Character Recognition is a field of research in pattern recognition, artificial intelligence and machine vision. Though academic research in the field continues, the focus on character recognition has shifted to implementation of proven techniques.

In view of above-mentioned discussion, there exists a need to provide improved method of character recognition by attribute analysis that can be applied to each character image, to achieve high degree of success. In particular, the method discussed relates to recognition of characters in a textual document based on certain attributes such as, strokes and size.

The method discloses a process of recognising characters in a textual document, the textual document being represented as an image matrix. The method includes applying threshold operation on the image matrix to present the intensity thereof with predefined intensity values, converting the image matrix into single bit binary representation, reducing the thickness of the character strokes to a single line by applying thinning operation on the image matrix, performing grid transformation on the image matrix considering the basic pixel element in a virtual triangle and reducing dependency of character recognition on character curvatures, segregating individual character sequentially from the image matrix starting from a predefined location and storing it in a character matrix, calculating horizontal strokes, vertical strokes, length, and true elements of each character, recognising individual character of the character matrix based on such horizontal strokes, vertical strokes, length, and true elements and repeating the above steps until all the characters of the image matrix representing the complete text document are recognized and an editable text file corresponding to the input text document is created. The emphasis has been to eliminate the dependency of the recognition process on the curvatures possessed by the characters so as to make it more reliable, accurate and less complex.

CHAPTER 1

INTRODUCTION

Optical Character Recognition (OCR) is a process that translates images of typewritten scanned text into machine-editable text, or pictures of characters into a standard encoding scheme representing them in ASCII or Unicode. An OCR system enable us to feed a book or a magazine article directly into a electronic computer file, and edit the file using a word processor. Though academic research in the field continues, the focus on OCR has shifted to implementation of proven techniques. Optical character recognition (using optical techniques such as mirrors and lenses) and digital character recognition (using scanners and computer algorithms) were originally considered separate fields. Because very few applications survive that use true optical techniques, the OCR term has now been broadened to include digital image processing as well. Early systems required training (the provision of known samples of each character) to read a specific font. "Intelligent" systems with a high degree of recognition accuracy for most fonts are now common. Some systems are even capable of reproducing formatted output that closely approximates the original scanned page including images, columns and other non-textual components. However, this approach is sensitive to the size of the fonts and the font type. For handwritten input, the task becomes even more formidable. Soft computing has been adopted into the process of character recognition for its ability to create input output mapping with good approximation. The alternative for input/output mapping may be the use of a lookup table that is totally rigid with no room for input variations.

1.1 Existing OCR Systems

OCR concept came into picture for two main reasons: telegraphy and creating reading devices for the blind, about a century ago. There are different type of OCR systems that are under development or have been designed, like the desktop or server OCR softwares that consider sequences of characters rather than whole words or phrases and then based on analysis of sequential lines and curves, it makes 'best guesses' at characters using database look-up tables. There are some other OCR systems that utilize a database of dictionary words for the recoginition of words as a whole. In such OCR systems a word by word approach is followed rather than a character by character approach.

As far as accuracy is concerned for the currently available OCR systems, character-by-character accuracy for commercial OCR software varies from 71% to 98%; total accuracy can be achieved only by human review.

Accuracy rates can be measured in several ways, and how they are measured can greatly affect the reported accuracy rate. For example, if word context (basically a lexicon of words) is not used to correct software finding non-existent words, a character error rate of 1% (99% accuracy) may result in an error rate of 5% (95% accuracy) or worse if the measurement is based on whether each whole word was recognized with no incorrect letters.

Depending upon the techniques that are employed to recognize text, there are some OCR systems that recognize the patterns of the strokes of the characters and are independent of the font size, but they produce problems with recognition of characters having same shape in uppercase and lowercase and in other cases where size of the recognized character also plays important role, while in some of the other systems, recognition also takes character size into account, hence giving extra information

Thus there is still a lot of work that can be done in this fields and keeping in line with this fact, systems under development in contemporary world use different algorithms, thus giving favourable results in one situation but lacking in some other situation while recognising the text, so designing a perfect systems that works in all environment is not possible but designing the system working in a perticular domain and giving good result needs to be taken into consideration.

1.2 Methodology

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 research for an innovative approach to character recognization 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 oparation that is used to remove selected foreground pixel from binary images. Then, statistical analysis of stroke patterns, including both horizontal and vertical 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.

1.3 Benefits

Save data entry costs - automatic recognition by OCR/ICR/OMR/barcode engines ensure lower manpower costs for data entry and validation.

Lower licensing cost - since the product enables distributed capture-licensing costs for OCR/ICR engine is much lower. For instance 5 workstations may be used for scanning and indexing but only one OCR/ICR license may be required.

Export the recognized data in XML or any other standard format for integration with any application or database.

1.4 Applications

Industries and Institutions in which control of large amounts of paper work is critical

- Banking, Credit cards, Insurance industries

Libraries and archives

- For conservation and preservation of vulnerable documents and for the provision of access to source documents

OCR fonts are used for several purposes where automated systems need a standard character shape defined to properly read text without the use of barcodes. Some examples of OCR font implementations include bank checks, passports, serial labels and postal mail.

An OCR has a variety of commercial and practical applications in reading forms, manuscripts and their archival etc. Such a system facilitates a keyboard less user-computer interaction. Also the text, which is either printed or hand-written, can be directly transferred to the machine. The challenge of building an OCR system that can match the human performance also provides a strong motivation for research in this field.

CHAPTER 2

METHODOLOGY

2.1 Conventional Approach

With the need of effective and accurate approaches for extracting and retrieving information from scanned documents and images many information retrieval techniques have been proposed over the years. Our technique is based on feature analysis of each individual character. The efficiency of the algorithm increases when characters are separated in the image. The shape properties and gradient information of the original image are usually subject to font variations such as boldness, font size, font type, etc.

The contents of a textual document can be broken down into individual character alphabets. Each of these individual characters has certain attributes such as strokes, curvatures and size. It is on the basis of these attributes that the character recognition process is carried out.

In a conventional approach the technique of character recognition is applied based on attributes such as horizontal and vertical strokes, slant strokes with positive and negative slope, left end and right end closed curves. However, the limitations in such approach includes if the technique is applied to all alphabets, it is unable to rightly recognise alphabets with similar upper case and lower case such as C, V, O, W, K etc. Further, it also fails to distinguish between character with equal number of all such parameters such as D and P etc.

2.2 Approach Followed

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 character the images are drawn with the help of MS Paint and then their properties are determined using MATLAB.

The text in a document can be broken down into individual characters – the basic building block of a textual document. Each of these individual characters have certain attributes – strokes, curvatures, length, width, etc. It is on the basis of these attributes that the character recognition process is carried out. The approach that we have followed incorporates attribute analysis that can be applied to each character image, to match a set of predefined criteria for character differentiation.

The emphasis has been to eliminate the dependency of the recognition process on the curvatures possessed by the characters so as to make it more reliable, accurate and less complex. Also, to do away with the ambiguity experienced while dealing with similar upper case and lower case characters such as C, V, O, W, etc, the method takes into consideration parameters relating to the size of the character.

Although when it comes to OCR systems total accuracy can only be achieved by human review, the method discussed tries to provide an error free final product to the user that recognizes both upper case and lower case alphabets, special characters, word spacing, line breaks and even paragraph breaks, so that minimal formatting is required.

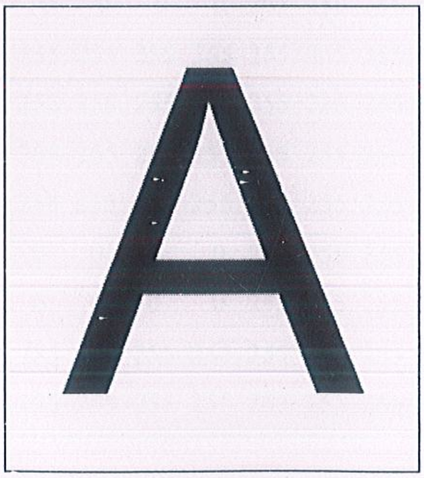


Figure 1: Original image for alphabet ‘A’

We consider that the textual document is scanned and saved as an image. Let this image matrix be called I, which consists of 8-bit per pixel intensity values (ranging from 0-255). Such an image matrix is shown below for the alphabet ‘E’.

Ideally, the intensity values for the pixels forming the character elements should be 0 (black) while the rest of the pixels should correspond to the intensity value 255 (white). However, the intensity values representing individual pixels may at some points take values between 0 and 255 (for the non ideal case when there is edge distortion, shadow projection, unwanted noise and other such factors).

The proposed process to recognize each individual character from a textual document consists of a series of operations. The complete methodology is explained stepwise.

255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	0	0	0	0	0	0	0	0	0	255	255	255	255	255	255	255	255	255
65	0	0	0	0	0	0	0	0	0	0	255	255	255	255	255	255	255	255	255
65	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	0	0	0	0	0	0	0	0	0	255	255	255	255	255	255	255	255	255
65	0	0	0	0	0	0	0	0	0	0	255	255	255	255	255	255	255	255	255
65	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
65	0	0	0	0	0	0	0	0	0	0	255	255	255	255	255	255	255	255	255
65	0	0	0	0	0	0	0	0	0	0	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255

Figure 2: The input image matrix for the alphabet ‘E’

2.3 Threshold Operation

Image matrix I consists of 256 intensity values. We apply thresholding operation to restrict I to only two intensity values – 0 and 255. All values greater than or equal to 75 are changed to 255, while all values below 75 are converted to 0 (this value is chosen as threshold since it is suitable with respect to shadows, external noise and retention of relevant information). This way the image matrix I is now reduced to black and white with no shades of grey in between. The result can be seen below for the alphabet ‘E’. The intensity values (below 75) near the edges of the character and elsewhere have been changed to ‘0’.

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

Figure 3: Image matrix for alphabet ‘E’ after applying Threshold Operation.

2.4 Binarization

The image matrix I, after applying the threshold operation, still consists of 8-bit intensity values – 0 and 255. To convert I into a binary image containing only 1-bit values, Binarization is carried out. 0's (8-bit) are converted to 0's (1-bit) and 255's (8-bit) are converted to 1's (1-bit). The image matrix I now has 0's – which make up the character part (True Elements), and 1's – which make up the non-character part (Non-True Elements). Each pixel value of the image matrix I has a single bit representation in the form of 0 (TE) or 1 (NTE).

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

Figure 4: Image matrix of alphabet 'E' after Binarization.

2.5 Thinning Operation

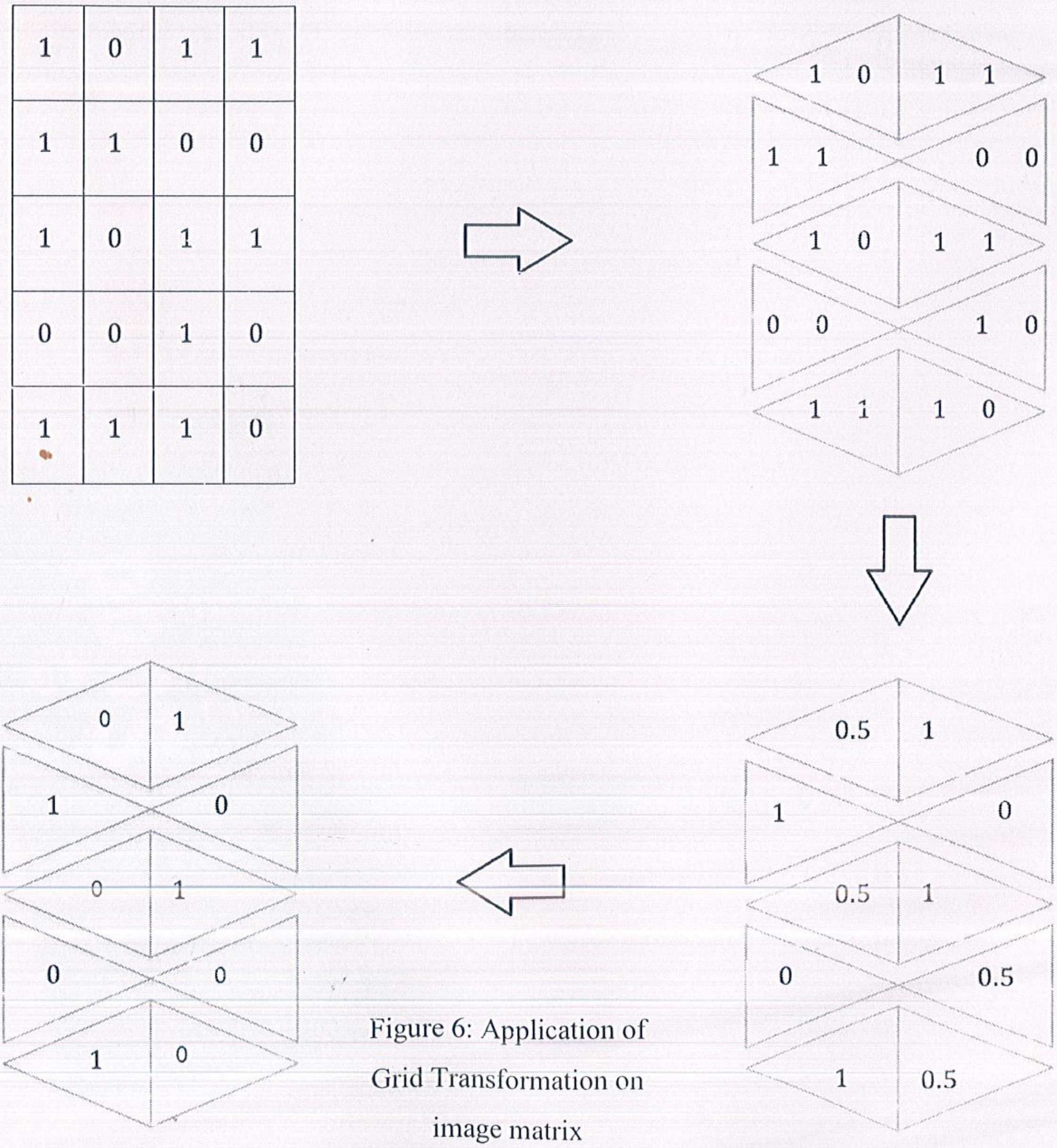
In the character image matrix of Figure 4, it can be seen that the alphabet ‘E’ is made up of line strokes (horizontal strokes and vertical strokes). In the figure, each of the horizontal and vertical strokes are represented by 2 consecutive rows and columns made up of true elements. The thinning operation is applied in order to reduce these strokes to a single line thickness. This is represented in Figure 5 where the strokes contained in alphabet ‘E’ have been narrowed down to a single line. The purpose of this operation is to retain the relevant information required for the recognition process and to do away with any superfluous information. In this case, a thick line stroke provides the same information as a single line thickness line stroke. Therefore, no useful information is lost on application of thinning operation.

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

Figure 5: Image matrix of alphabet ‘E’ after applying Thinning Algorithm.

2.6 Grid Transformation

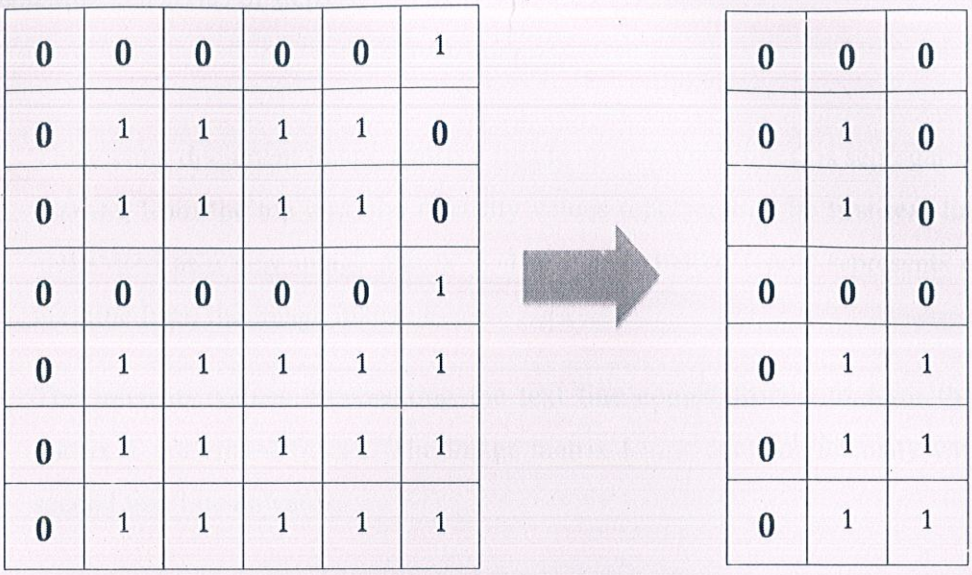
The image matrix I obtained after applying the thinning operation is now made to undergo Grid Transformation (GT). The basic pixel element is assumed to be a virtual triangle rather than a square. This is carried out by clubbing two consecutive horizontal pixel intensity values and replacing it with the average of the two. Thus, a ‘0’ is stored for two consecutive 0’s and a ‘1’ for consecutive 1’s. The values that remain after the application of GT are 0, 0.5 and 1. Since the image matrix needs to be binary, 0.5 is changed to a 0. The pixel grid of I, now having half the number of columns it was having initially, is again reduced to TE’s and NTE’s. The step-by-step process is illustrated in Figure 6.



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

Figure 6: Application of Grid Transformation on image matrix

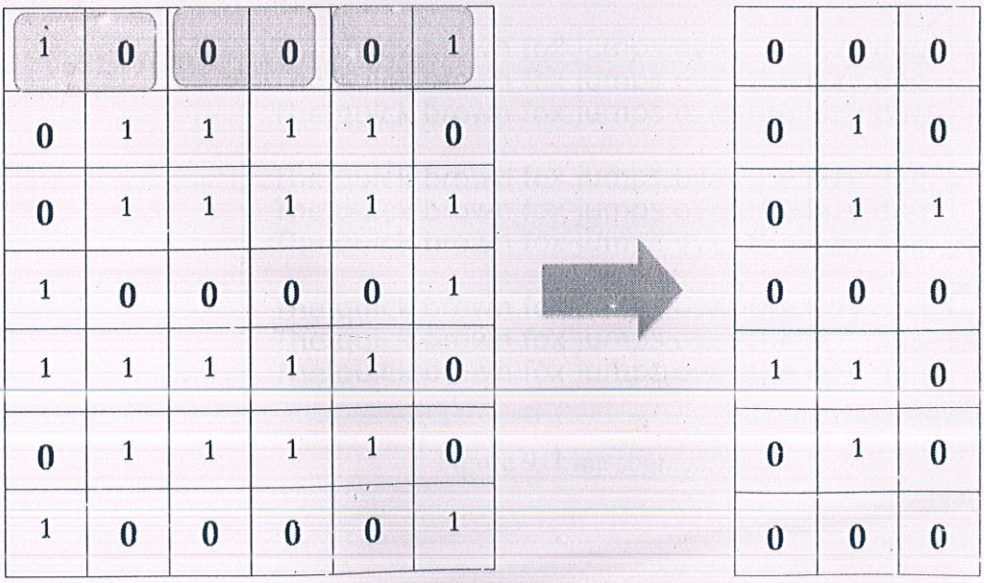
After application of grid transformation by implementing horizontal averaging there is no loss of significant information with regard to our recognition algorithm. Grid transformation is helpful in reducing the dependency of character recognition on character curvatures, thereby making the process much more accurate. This is illustrated in Figure 7 and Figure 8 that serpentine curves have been reduced to horizontal and vertical strokes.



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

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

Figure 7: Application of Grid Transformation on character ‘P’



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

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

Figure 8: Application of Grid Transformation on character ‘S’

2.7 Character Segregation

A textual document might contain numerous paragraphs, lines and words of text. Each character contained in the textual document image matrix I, after the application of above explained operations, needs to be processed individually and correctly recognized. For this, the character is studied separately from the paragraph, statement or word that it might be a part of. This is implemented in a series of steps –

- The textual document image matrix I (made up of several lines) is segregated line by line. Starting from the top end, the intensity values representing the first text line are copied and stored as a new image matrix T. The image matrix T now represents one complete text line from the image matrix I.
- The intensity values representing the text line copied from I to form the new image matrix T are erased from I. The image matrix I now contains intensity values from the second text line onwards.

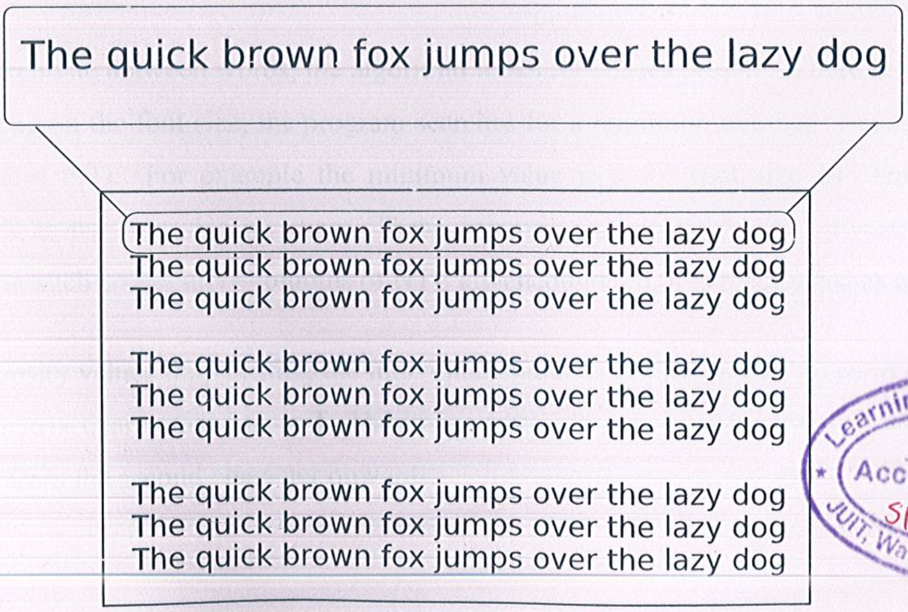

The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog

Figure 9: Line segregation

- The image matrix T, which is made up of several words is now segregated character by character, wherein a space occurring between words is utilized to differentiate between words. Starting from the left, the intensity values representing the first character are copied and stored as a new image matrix C. After finding the location of the first TE from the left end, the algorithm searches for one complete column of NTE. This complete column of NTE represents the gap between the characters. Once the coordinates corresponding to these two pixel values are located, the values in between are copied to form the new image matrix C. The image matrix C represents a single character extracted from T.

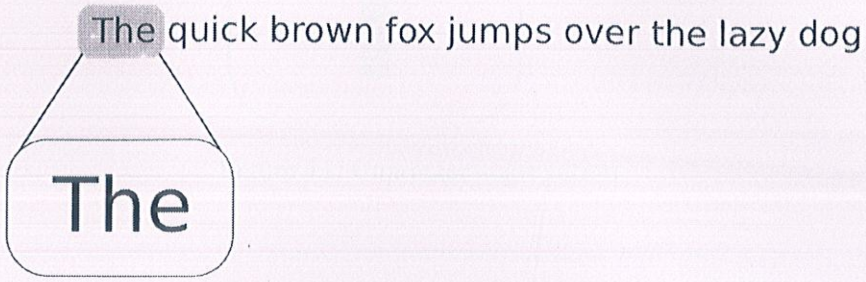


Figure 10: Word segregation

- To differentiate between words, the algorithm looks for spaces present in between words. Depending on the font size, the program searches for a minimum number of consecutive columns of NTE. For example the minimum value is 5 for font size 14. This value increases as the font size increases. These columns represent the gap between words. Wherever such consecutive columns of NTE are encountered, it is recognized as a space.

- The intensity values representing the individual character copied from T to form the new image matrix C are erased from T. The image matrix T now contains intensity values of the line from the second character onwards.

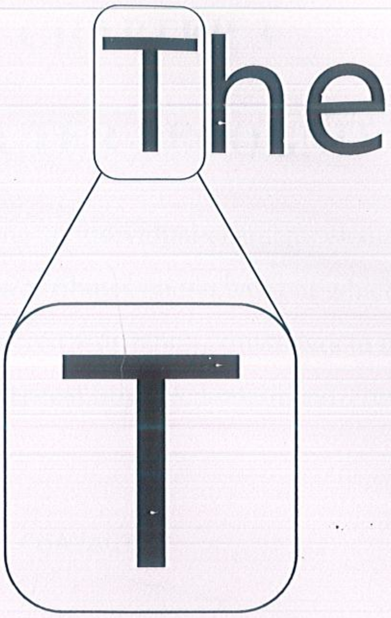


Figure 11: Character segregation

- The above steps are repeated until each character is stored in the matrix C individually. The segregation of the text document into constituent lines, the lines into constituent words and the words into constituent individual characters is carried out for the entire document. This way each character can be processed and recognized individually by the character recognition steps to follow.

CHAPTER 3

CHARACTER RECOGNITION

We take an image matrix C, containing an individual character element. C is processed in order to determine the character parameters/attributes on the basis of which a particular character is correctly recognized. We have considered 4 different parameters to differentiate one character from another – Horizontal Strokes, Vertical Strokes, Length and True Elements. Each of these has been explained below.

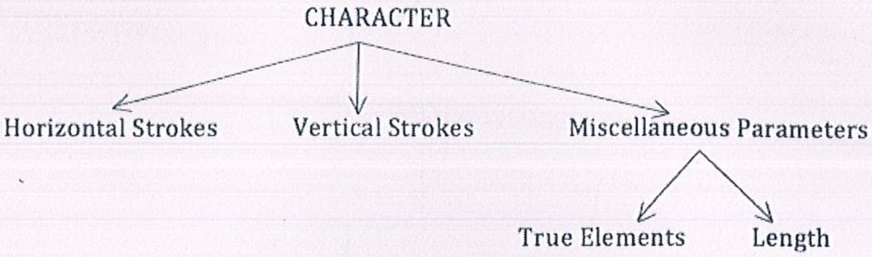
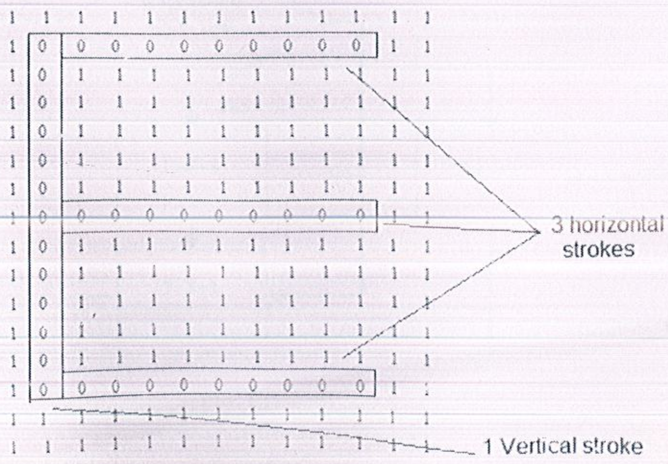


Figure 12: Character parameters

3.1 Line strokes – Horizontal and Vertical Strokes

Each character when represented in its matrix form is made up of true and non-true elements. In order to recognize vertical and horizontal strokes the algorithm starts scanning the image matrix vertically and horizontally respectively. On encountering a line stroke the value of the corresponding line stroke variable (H or V) is incremented by one. The process continues till all the constituent line strokes of the character are identified.



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

Figure 13: Horizontal and Vertical strokes for character 'E'

3.1.1 Horizontal Strokes

On finding the first true element while traversing the image matrix C from left to right and top to bottom, the algorithm determines the existence of a horizontal stroke (H) by checking for adjacent columns for a true value. If the number of consecutive TE are 2 or greater than 2, then it is identified as a horizontal stroke and the value of H (initially 0) is incremented by 1. The algorithm then moves on to the next row, and the process is repeated. The flow chart for the calculation of horizontal strokes has been depicted in Figure 14.

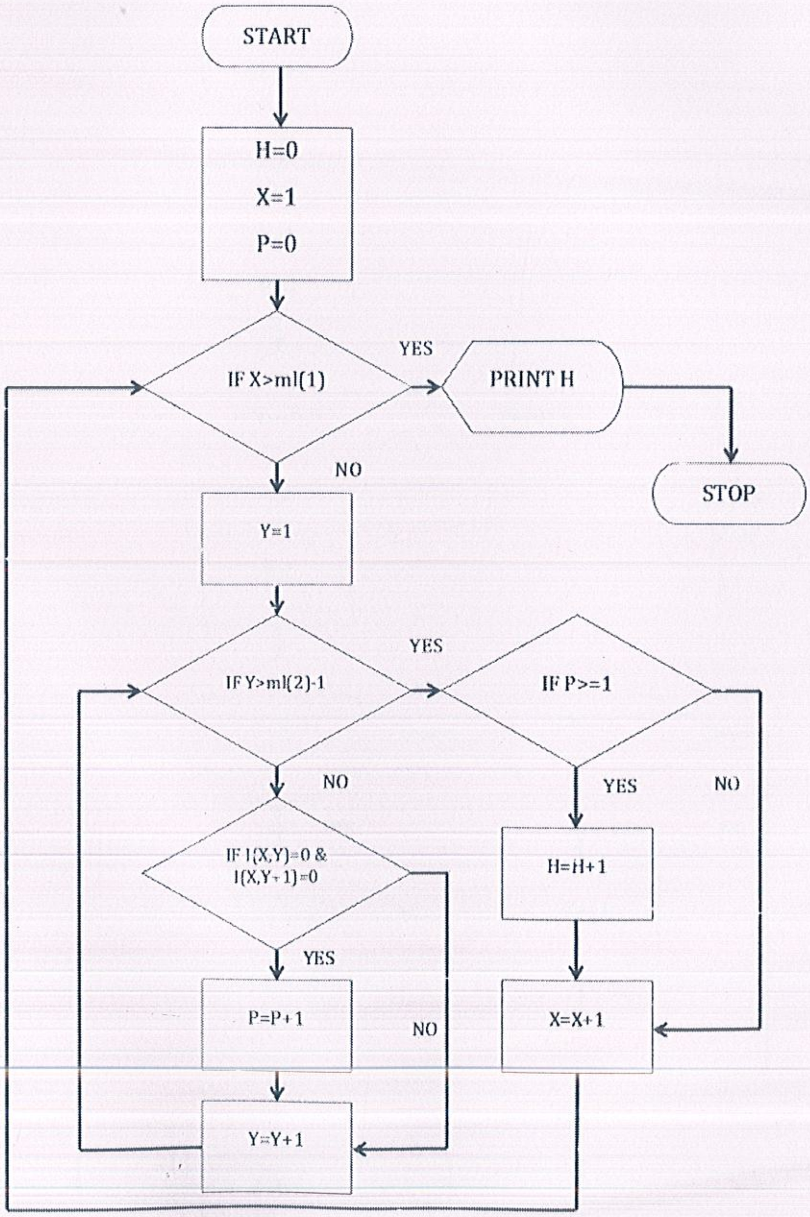


Figure 14: Flow chart for calculation of horizontal strokes.

3.1.2 Vertical Strokes

To calculate the number of vertical strokes (V), the character image matrix C is traversed from top to bottom and left to right. If the number of consecutive true elements in the same column comes out to be 3 or greater than 3 (The value is 2 while working with horizontal strokes since the horizontal extent of a character is less than the vertical extent), it is recognized as a vertical stroke and the value of V(initially 0) is incremented by 1. The control then passes on to the next column and the process is repeated until all the columns are scanned for vertical strokes. The flow chart for the calculation of horizontal strokes has been depicted in Figure 15.

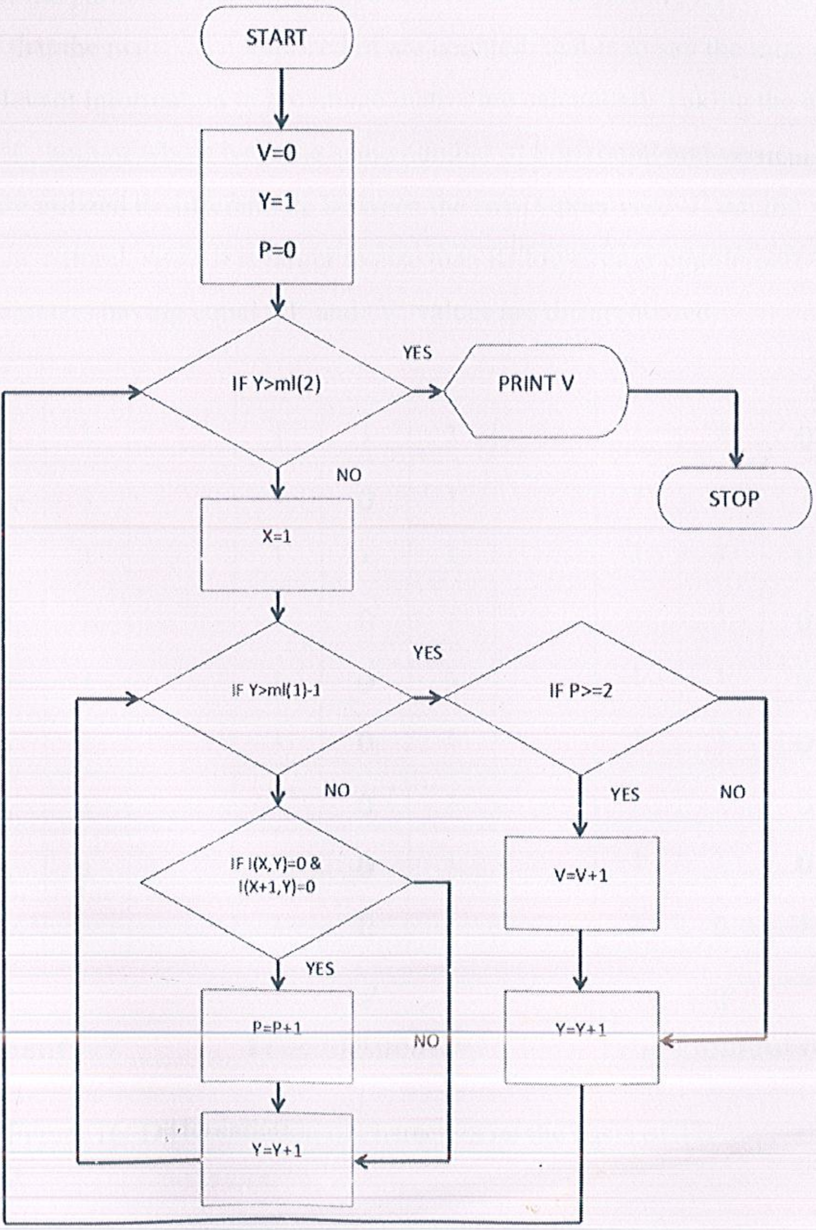


Figure 15: Flow chart for calculation of vertical strokes.

3.2 Miscellaneous Parameters

Not all characters can be differentiated on the basis of line strokes alone. It is a possibility that two or more characters might have the same value for H and V. For e.g. both upper case ‘I’ and lower case ‘i’ have H=0 and V=1. In such cases miscellaneous parameters are utilized to differentiate between the characters.

3.2.1 True Elements (TE)

The image matrix of the particular character is scanned from top to bottom, left to right and the total number of TE that the matrix is comprised of are counted, that is to say the total elements that make up the relevant information in the image matrix are calculated. Taking the above example of ‘I’ and ‘i’, both of which have the same number of horizontal and vertical strokes, the number of TE are utilized to differentiate between the two. Upper case ‘I’ having more number of TE is quite natural, since it is larger in size than its lower case counterpart ‘i’. Thus, on this basis the characters having equal ‘H’ and ‘V’ values are differentiated.

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

‘I’

True Elements=9

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

‘i’

True Elements=7

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

‘j’

True Elements=8

Figure 16: Differentiation of characters on the basis of TE

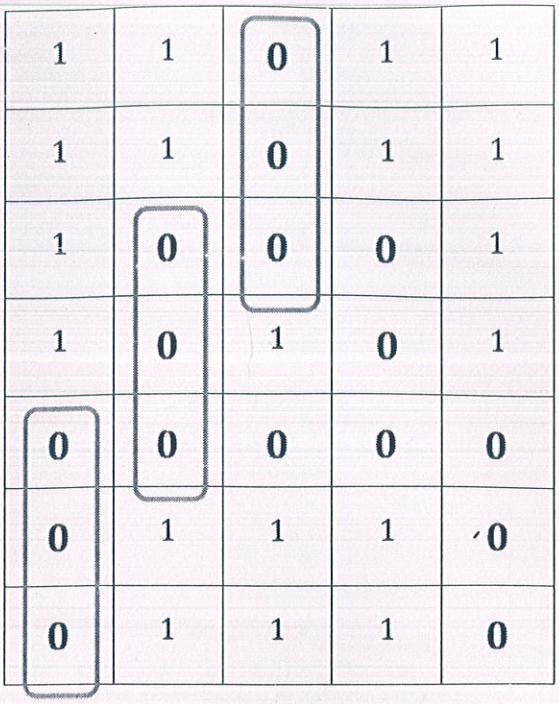
0	1	1
0	1	1
0	1	1
0	0	0
0	1	0
0	1	0
0	0	0

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

Figure 17: The characters 'b' and 'd' are essentially mirror images of each other, hence cannot be differentiated on the basis of H and V.

3.2.2 Length (L)

It might happen that the value of all the parameters discussed up till now comes out to be the same for a particular combination of characters i.e. the values of 'H', 'V', and number of TE all come out to be the same. For e.g. the characters 'T' and 'L' have the same number of horizontal strokes (1), vertical strokes (1) and number of TE (17). In such cases, another miscellaneous parameter 'Length' is taken into account. Two coordinate positions corresponding to the top left and bottom left TE of the character are calculated. The variable L(initially 0) traverses through each column between these two coordinates and on encountering two consecutive vertically placed TE, value of L is incremented. This procedure has been illustrated in Figure 17 for the character 'A'.



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

Figure 18: Calculation of the parameter 'Length' for alphabet 'A'

The four parameters – Horizontal Strokes (H), Vertical Strokes (V), Length (L) and True Elements (TE) are calculated for each individual character of the document image matrix I. One by one each constituent character of the document image matrix I is extracted and stored in the character matrix C. C is overwritten each time a new character is extracted from I, but before that the character contained in C it is recognized by the character recognition algorithm and exported to a text file.

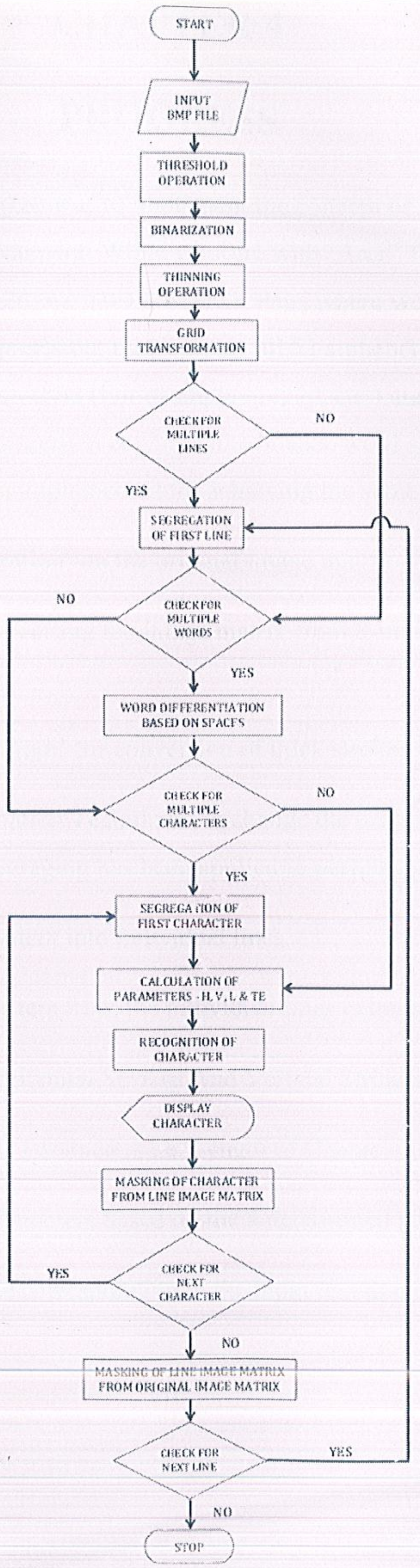

START
INPUT BMP FILE
THRESHOLD OPERATION
BINARIZATION
THINNING OPERATION
GRID TRANSFORMATION
CHECK FOR MULTIPLE LINES
NO
YES
SEGREGATION OF FIRST LINE
NO
CHECK FOR MULTIPLE WORDS
YES
WORD DIFFERENTIATION BASED ON SPACES
CHECK FOR MULTIPLE CHARACTERS
NO
YES
SEGREGATION OF FIRST CHARACTER
CALCULATION OF PARAMETERS - H, V, L & TE
RECOGNITION OF CHARACTER
DISPLAY CHARACTER
MASKING OF CHARACTER FROM LINE IMAGE MATRIX
YES
CHECK FOR NEXT CHARACTER
NO
MASKING OF LINE IMAGE MATRIX FROM ORIGINAL IMAGE MATRIX
CHECK FOR NEXT LINE
YES
NO
STOP

Figure 19: Complete flow chart for the method

CHAPTER 4

PROGRESS

The objective that we had undertaken was to implement the concept of Optical Character Recognition (OCR) to a whole document. While working with 'Arial' font with font sizes ranging from '14', '16', '18' and 20, we have reached a stage where we are able to recognize all individual characters with 100% precision, that is to say all 52 alphabet characters (26 upper case and 26 lower case), 31 special characters (For example ~, !, @, etc.) and 10 digits have been recognized. The algorithm also correctly recognises line breaks, word spacing and paragraph breaks. Following steps have been implemented for achieving the same:

1. Application of 'Threshold Operation' on the original image matrix.

2. 'Binarization Operation' for converting the image matrix from 8-bit intensity values to 1-bit intensity values.

3. Application of 'Thinning Algorithm' for conversion of thick strokes to single pixel thickness.

4. Application of 'Grid Transformation Technique' to change the original rectangular pixel grid to a virtual triangle. Horizontal averaging has been applied to perform grid transformation.

5. Segregation of the textual document into individual lines.

6. Segregation of individual characters from the individual lines extracted.

7. Determination of number of Horizontal Strokes And Vertical Strokes.

8. Determination of Miscellaneous Parameters i.e. Length of Characters and number of TE.

9. Recognition of all individual characters based on the 4 determined parameters.

10. Generating an editable text file corresponding to the input text document.

Character	Horizontal Strokes (H)	Vertical Strokes (V)	Length (L)	True Elements (Zeroes)
A	3	5	10	27
B	3	3	12	32
C	3	2	1	24
D	2	3	12	30
E	3	1	12	24
F	2	1	12	19
G	3	2	0	26
H	1	2	12	29
I	0	1	13	14
J	1	1	12	17
K	3	3	13	30
L	1	1	13	17
M	7	6	12	44
N	4	4	12	33
O	3	2	1	29
P	2	2	12	25
Q	3	2	0	30
R	2	3	12	31
S	4	2	8	25
T	1	1	12	17
U	1	2	12	27
V	2	6	11	26
W	4	8	29	47
X	3	5	2	26
Y	0	5	11	20
Z	2	3	4	20

Table 1: Parameter values for upper case alphabets for font size 14

Character	Horizontal Strokes (H)	Vertical Strokes (V)	Length (L)	True Elements (Zeroes)
a	4	2	1	21
b	3	2	13	26
c	3	1	1	14
d	3	2	11	25
e	4	2	1	21
f	0	1	10	13
g	5	3	1	29
h	1	2	13	24
i	0	1	10	12
j	0	1	13	16
k	6	3	13	25
l	0	1	13	14
m	2	3	9	31
n	1	2	9	20
o	2	3	1	19
p	3	2	13	26
q	3	3	10	26
r	1	1	9	11
s	4	2	1	17
t	0	1	12	13
u	1	2	9	22
v	3	4	8	18
w	4	6	7	29
x	3	3	3	18
y	4	4	3	22
z	2	2	3	13

Table 2: Parameter values for lower case alphabets for font size 14

Character	Horizontal Strokes (H)	Vertical Strokes (V)	Length (L)	True Elements (Zeroes)
~	2	0	2	7
`	0	1	2	3
!	0	1	11	13
@	11	7	4	64
#	2	3	12	29
$	5	4	2	23
%	3	5	7	38
^	2	3	7	12
&	5	4	2	30
*	1	1	0	7
(	0	1	0	16
)	0	2	6	18
-	1	0	0	2
_	1	0	0	5
=	2	0	0	8
+	1	1	9	13
{	0	2	11	17
}	0	1	1	17
[	0	1	16	17
]	0	1	1	17
\	0	2	9	12
|	0	1	17	18
:	0	1	2	4
;	0	1	2	5
‘	0	1	4	5
<	5	3	2	14

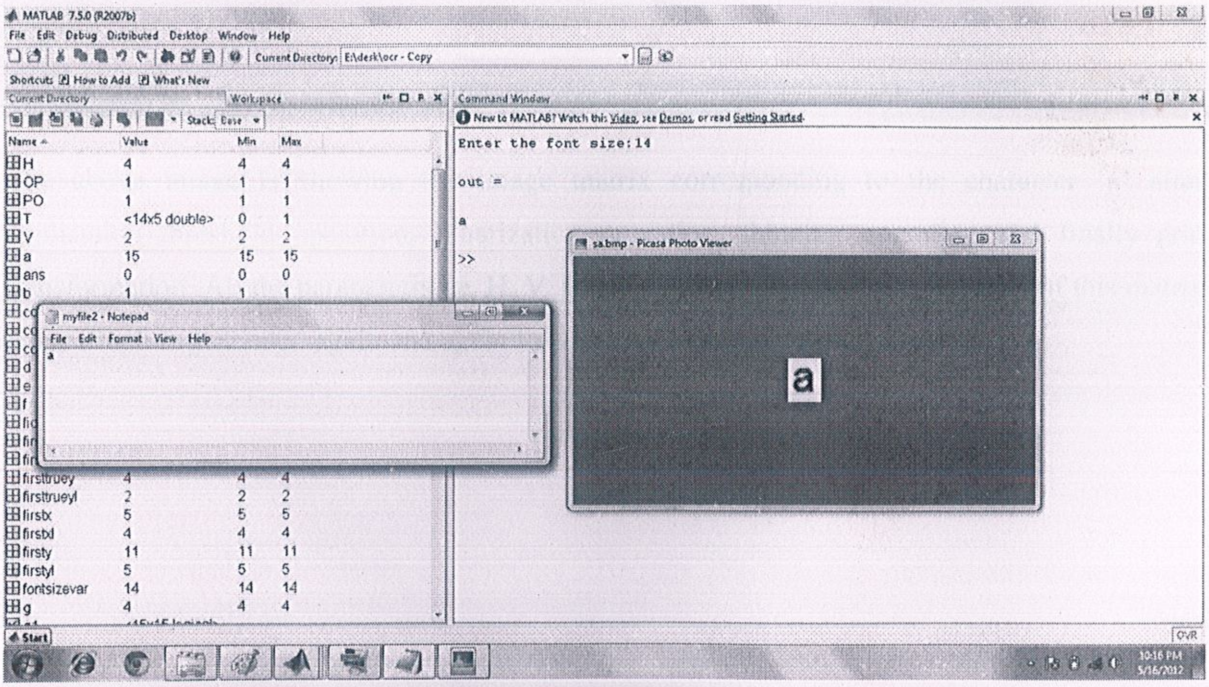
Character	Horizontal Strokes (H)	Vertical Strokes (V)	Length (L)	True Elements (Zeroes)
>	4	2	2	11
?	1	3	5	16
/	0	2	9	12
0	3	2	1	26
1	3	2	12	16
2	2	2	5	20
3	3	3	4	22
4	2	4	8	22
5	3	2	0	21
6	3	3	1	27
7	1	2	6	15
8	4	3	2	27
9	5	5	2	29
.	0	0	0	1
,	0	0	0	2

Table 3: Parameter values for digits and special characters for font size 14

CHAPTER 5

OUTPUT OF ALGORITHM ON MATLAB PLATFORM

Each of the following figures represents an output·of experimentation where images were drawn in MS Paint and saved as BMP files. It is to be noted that all the experiments have been performed on Windows 7 OS and the software version of MATLAB used is 7.5.0.342. All the inputs have been accepted in monochrome bitmap file format.

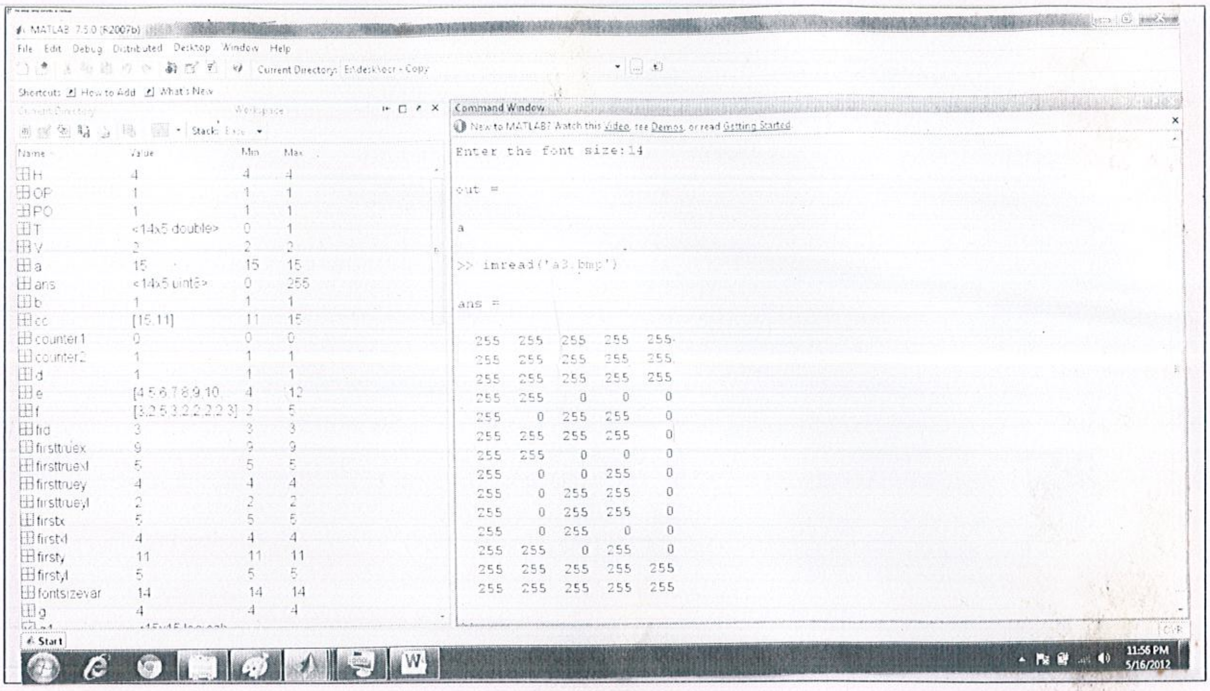

MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: E:\desk\ocr - Copy
Shortcuts How to Add What's New
Current Directory
Workspace
Stack: Base
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
myfile2 - Notepad
File Edit Format View Help
sa.bmp - Picasa Photo Viewer
a
Start
OVR
10:16 PM
5/16/2012

Name ▲	Value	Min	Max
H	4	4	4
OP	1	1	1
PO	1	1	1
T	<14x5 double>	0	1
V	2	2	2
a	15	15	15
ans	0	0	0
b	1	1	1
firsttruey	4	4	4
firsttrueyl	2	2	2
firstx	5	5	5
firstxl	4	4	4
firsty	11	11	11
firstyl	5	5	5
fontsizevar	14	14	14
g	4	4	4

```
Enter the font size:14

out =

a

>>
```

Figure 20: Output of algorithm for input character ‘a’ for font size 14.

For the input image of a single character (‘a’ in the above case), the algorithm does not require any segregation in terms of words or lines. The character is straight away recognized on the basis of the four parameters calculated (H, V, L and TE).

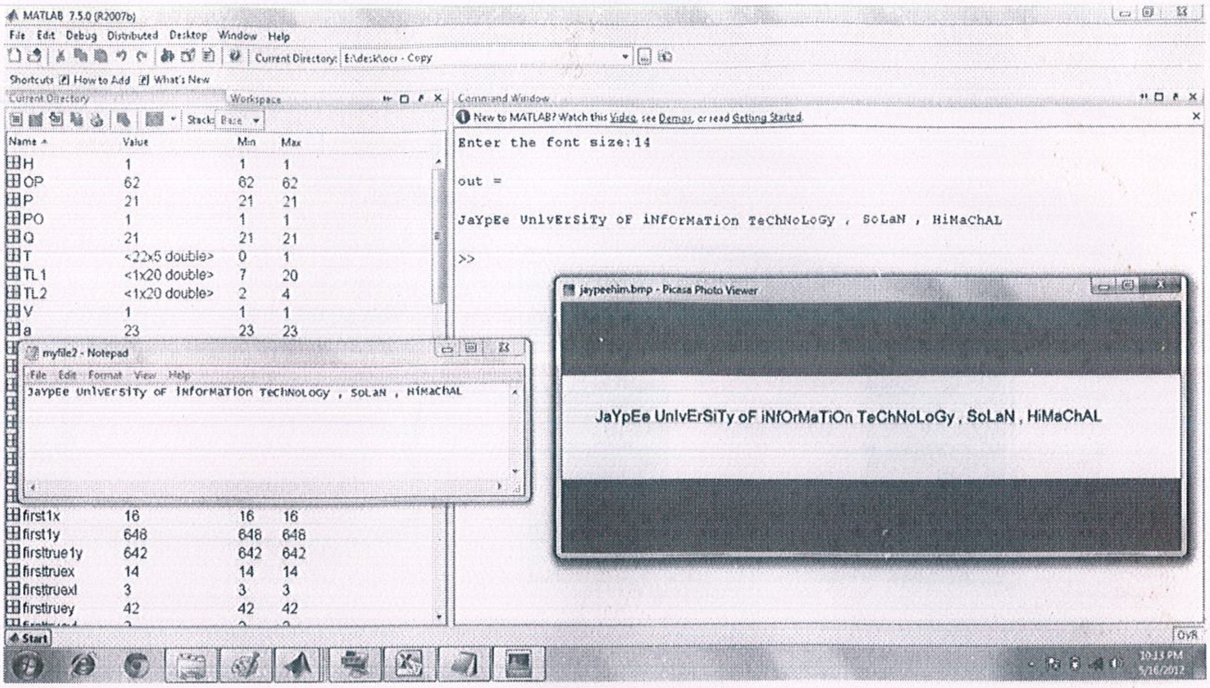


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: E:\desk\ocr - Copy
Shortcuts How to Add What's New
Workspace
Stack: Base
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Start
11:56 PM
5/16/2012

Name	Value	Min	Max
H	4	4	4
OP	1	1	1
PO	1	1	1
T	<14x5 double>	0	1
V	2	2	2
a	15	15	15
ans	<14x5 uint8>	0	255
b	1	1	1
cc	[15,11]	11	15
counter1	0	0	0
counter2	1	1	1
d	1	1	1
e	[4,5,6,7,8,9,10,	4	12
f	[3,2,5,3,2,2,2,2,3]	2	5
fid	3	3	3
firsttruex	9	9	9
firsttruexl	5	5	5
firsttruey	4	4	4
firsttrueyl	2	2	2
firstx	5	5	5
firstxl	4	4	4
firsty	11	11	11
firstyl	5	5	5
fontsizevar	14	14	14
g	4	4	4

```
Enter the font size:14

out =

a

>> imread('a3.bmp')

ans =

  255  255  255  255  255
  255  255  255  255  255
  255  255  255  255  255
  255  255    0    0    0
  255    0  255  255    0
  255  255  255  255    0
  255  255    0    0    0
  255    0    0  255    0
  255    0  255  255    0
  255    0  255  255    0
  255    0  255    0    0
  255  255    0  255    0
  255  255  255  255  255
  255  255  255  255  255
```

The above image is showing the image matrix corresponding to the character 'a' after application threshold operation, binarization operation, thinning operation and finally grid transformation. All the parameters i.e. H, V, L and TE are calculated with the help of this matrix only, which is generated again and again for all the characters encountered.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: E:\desk\ocr - Copy
Shortcuts How to Add What's New
Current Directory
Workspace
Stack: Base
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
myfile2 - Notepad
File Edit Format View Help
jaypeehim.bmp - Picasa Photo Viewer
Start

Name	Value	Min	Max
H	1	1	1
OP	62	62	62
P	21	21	21
PO	1	1	1
Q	21	21	21
T	<22x5 double>	0	1
TL1	<1x20 double>	7	20
TL2	<1x20 double>	2	4
V	1	1	1
a	23	23	23
first1x	16	16	16
first1y	648	648	648
firsttrue1y	642	642	642
firsttruex	14	14	14
firsttruexl	3	3	3
firsttruey	42	42	42

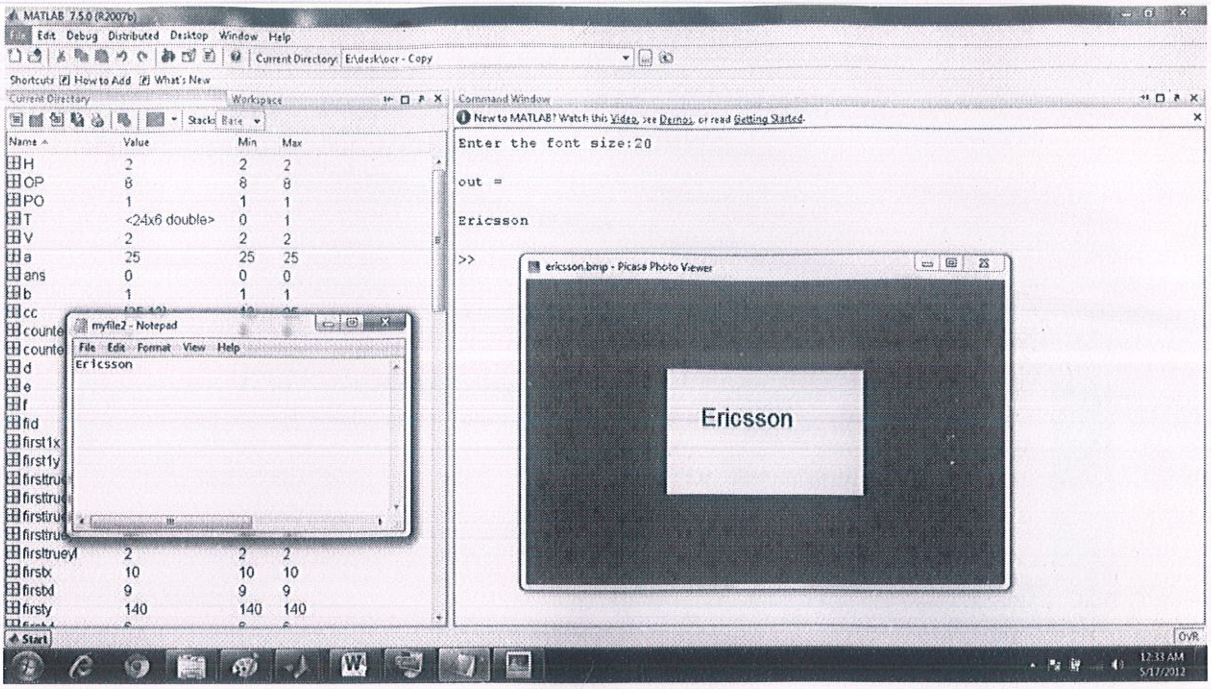
```
Enter the font size:14

out =

JaYpEe UnIvErSiTy oF iNfOrMaTiOn TeChNoLoGy , SoLaN , HiMaChAL

>>
```

```
JaYpEe UnIvErSiTy oF iNfOrMaTiOn TeChNoLoGy , SoLaN , HiMaChAL
```

JaYpEe UnIvErSiTy oF iNfOrMaTiOn TeChNoLoGy , SoLaN , HiMaChAL

Figure 21: Output of algorithm for an input image consisting of a single statement of font size 14.

The recognition of a textual document consisting of a single statement containing both special characters and alphabets is depicted in Figure 21. The algorithm first works to segregate the statement character by character and works to provide spacing wherever occurring in between words. The output is provided in an editable text file.



MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: E:\desk\ocr - Copy
Shortcuts How to Add What's New
Current Directory
Workspace
Stack: Base
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
myfile2 - Notepad
File Edit Format View Help
ericsson.bmp - Picasa Photo Viewer
Ericsson
Start
OVR
12:33 AM
5/17/2012

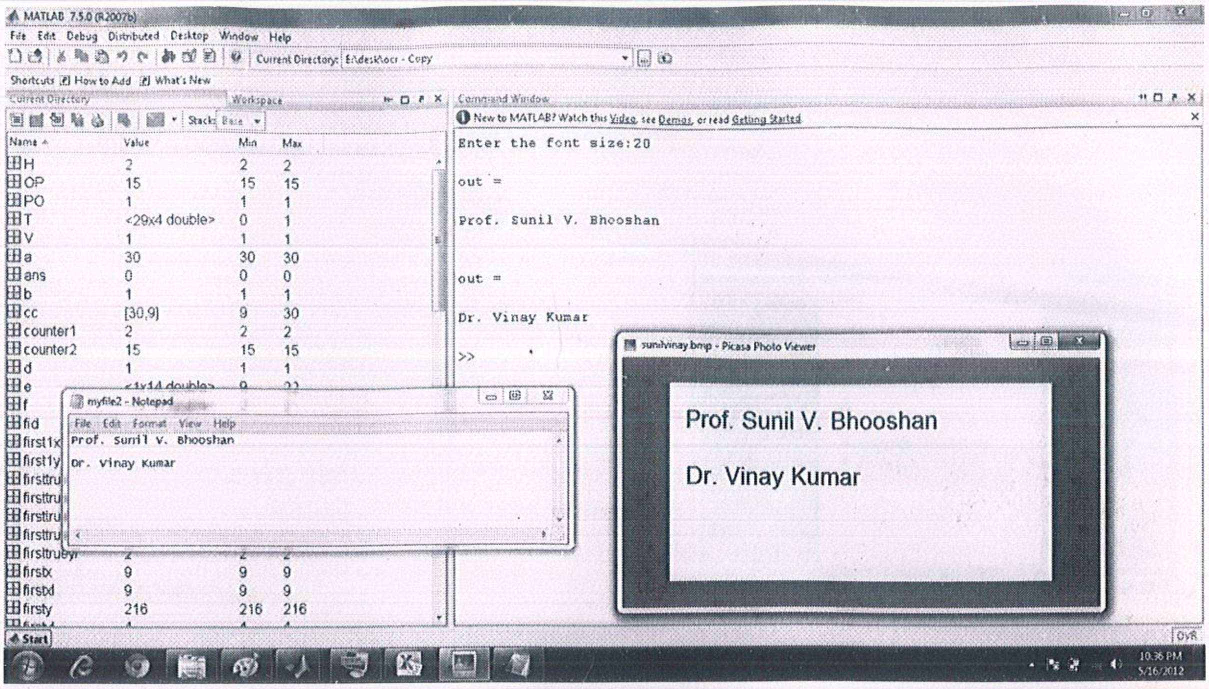
Name	Value	Min	Max
H	2	2	2
OP	8	8	8
PO	1	1	1
T	<24x6 double>	0	1
V	2	2	2
a	25	25	25
ans	0	0	0
b	1	1	1
cc			
counte			
counte			
d			
e			
f			
fid			
first1x			
first1y			
firsttru			
firsttru			
firsttru			
firsttru			
firsttrueyl	2	2	2
firstx	10	10	10
firstxl	9	9	9
firsty	140	140	140

```
Ericsson
```

```
Enter the font size:20

out =

Ericsson

>>
```

Figure 22: Output of algorithm for single word of font size 20.

The above output shows single word input to the program for the font size 20. For this BMP file the algorithm extracts each character on a one to one basis and with the help of character parameters calculated thereafter, the output is exported to the editable text file. There are no spaces present between the characters. This is determined by the program with the help of the number of blank columns (having Non-True Elements) present between the two characters, as discussed in the methodology.



MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: E:\desk\ocr - Copy
Shortcuts How to Add What's New
Current Directory
Workspace
Stack: Base
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
myfile2 - Notepad
File Edit Format View Help
sunilvinay.bmp - Picasa Photo Viewer
Start
OVR
10:36 PM
5/16/2012

Name	Value	Min	Max
H	2	2	2
OP	15	15	15
PO	1	1	1
T	<29x4 double>	0	1
V	1	1	1
a	30	30	30
ans	0	0	0
b	1	1	1
cc	[30,9]	9	30
counter1	2	2	2
counter2	15	15	15
d	1	1	1
e	<1x14 double>	9	22
f			
fid			
first1x			
first1y			
firsttru			
firsttru			
firsttru			
firsttru			
firsttrueyr	2	2	2
firstx	9	9	9
firstxl	9	9	9
firsty	216	216	216

```
Enter the font size:20

out =

Prof. Sunil V. Bhooshan


out =

Dr. Vinay Kumar

>>
```

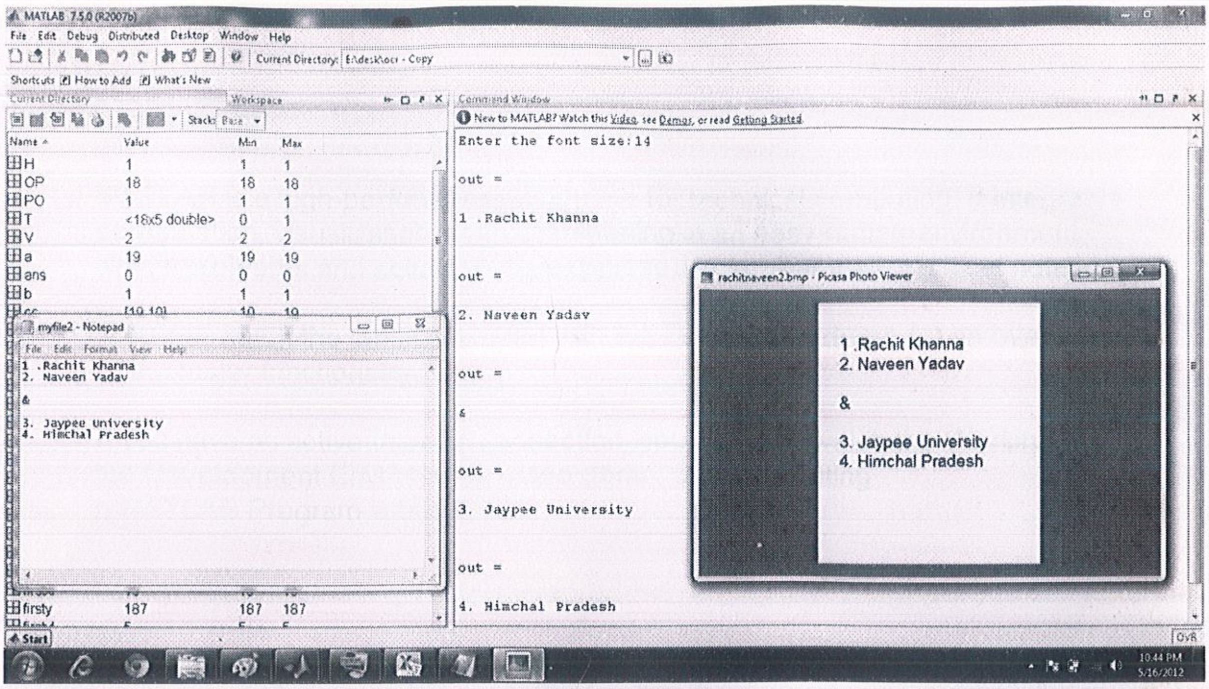
```
Prof. Sunil V. Bhooshan

Dr. Vinay Kumar
```

Prof. Sunil V. Bhooshan

Dr. Vinay Kumar

Figure 23: Output of algorithm for an input of multiple lines

The above image shows the output for multiple lines for font size 20. When multiple lines are encountered then the program extracts each lines individually from the document. When program starts searching for the true element while traversing the program from the top left, it takes into account the vertical coordinate value of that first true element encountered and then it further searches for a complete rows of non true elements while moving vertically downwards, and as that row is encountered it again takes into account the vertical coordinate value of the row and then copies all the values in between and this part is used to make one more matrix which is a simple line matrix and can be processed as discussed in the methodology.


MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: E:\desk\ocr - Copy
Shortcuts How to Add What's New
Current Directory
Workspace
Stack: Base
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
myfile2 - Notepad
File Edit Format View Help
rachitnaveen2.bmp - Picasa Photo Viewer
Start
OVR
10:44 PM
5/16/2012

Name	Value	Min	Max
H	1	1	1
OP	18	18	18
PO	1	1	1
T	<18x5 double>	0	1
V	2	2	2
a	19	19	19
ans	0	0	0
b	1	1	1
firsty	187	187	187

```
Enter the font size:14

out =

1 .Rachit Khanna


out =

2. Naveen Yadav


out =

&


out =

3. Jaypee University


out =

4. Himchal Pradesh
```

```
1 .Rachit Khanna
2. Naveen Yadav

&

3. Jaypee University
4. Himchal Pradesh
```

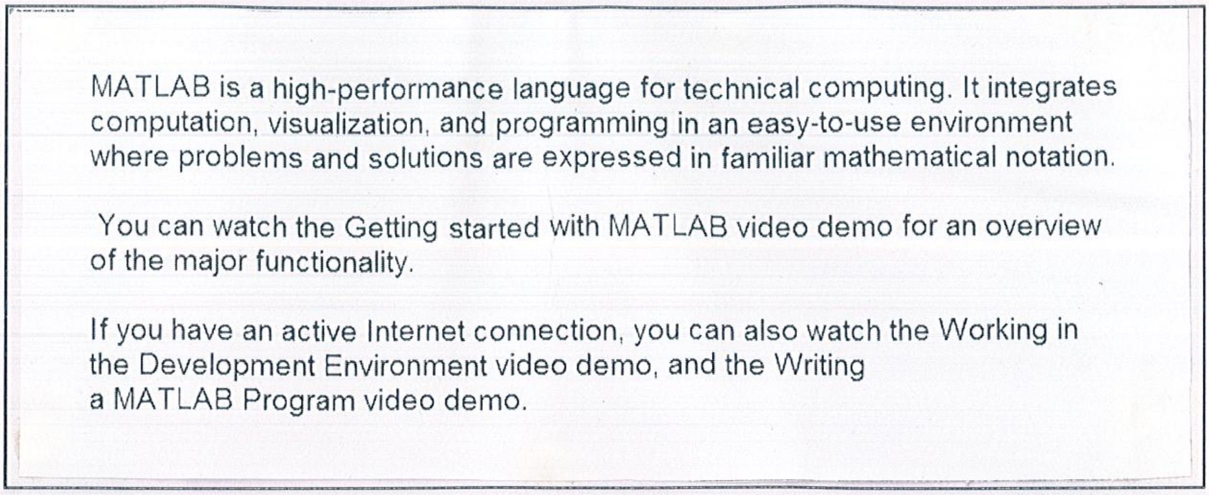
1 .Rachit Khanna
2. Naveen Yadav

&

3. Jaypee University
4. Himchal Pradesh

Figure 24: Output of algorithm for an input of multiple lines with special characters, digits and paragraph break

The above image shows the output generated by the program for input image having special character, digits and paragraph breaks i.e. multiple lines with spacing in between few lines so as to make a new paragraph. When such image input is provided, the program identifies the spacing in between the lines and print the same in the text file so that the user don't have to do the formatting explicitly for paragraphs.

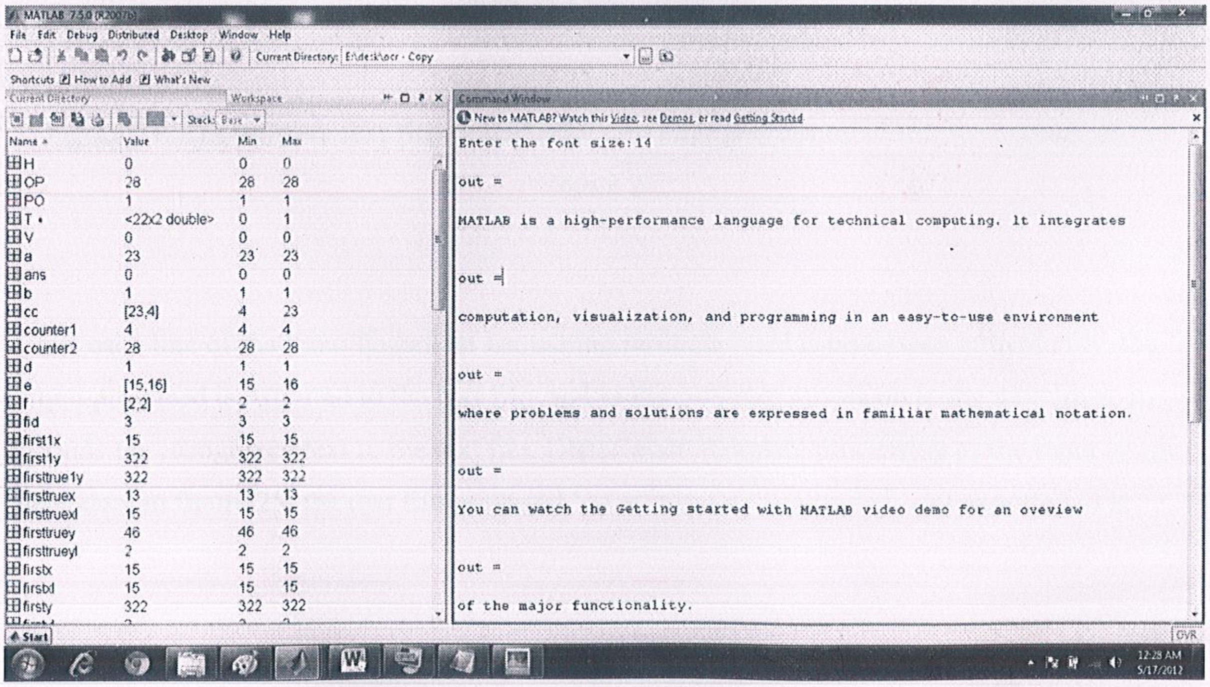


MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

You can watch the Getting started with MATLAB video demo for an overview of the major functionality.

If you have an active Internet connection, you can also watch the Working in the Development Environment video demo, and the Writing a MATLAB Program video demo.

Figure 25: Input image of multiple lines containing large amount of text and special characters.

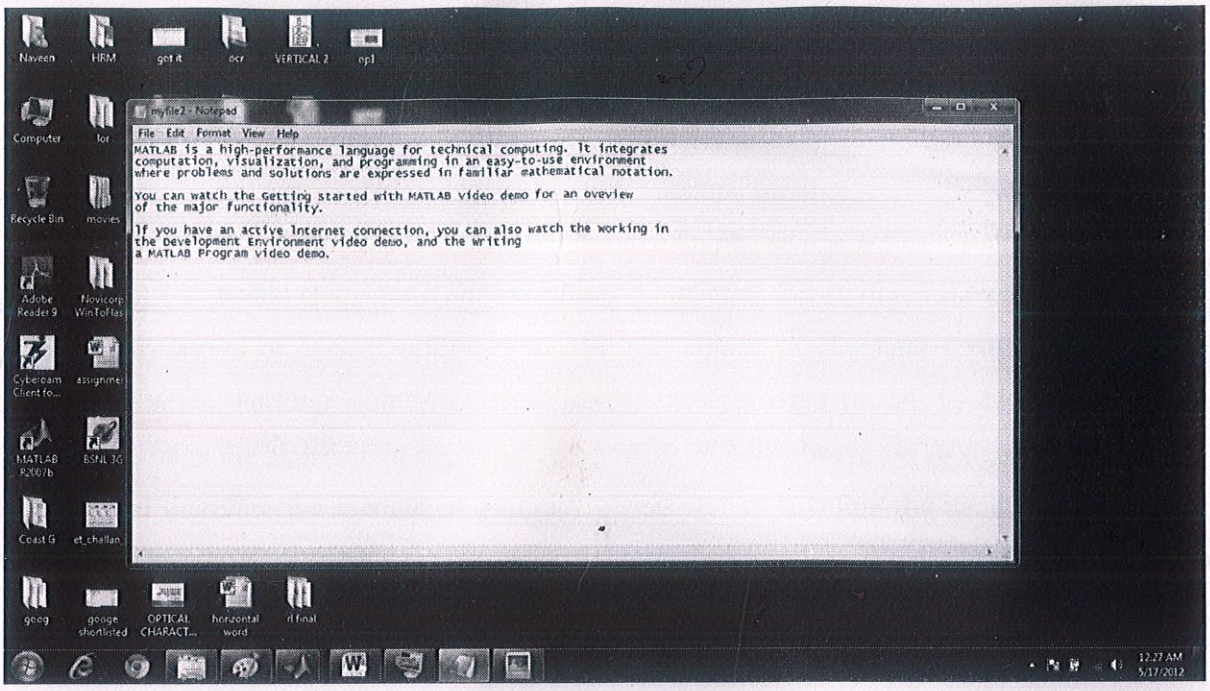

MATLAB 7.5.0 (R2007b)
File Edit Debug Distributed Desktop Window Help
Current Directory: E:\desk\ocr - Copy
Shortcuts How to Add What's New
Current Directory
Workspace
Stack: Base
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started
Start
OVR
12:28 AM
5/17/2012

Name ▲	Value	Min	Max
H	0	0	0
OP	28	28	28
PO	1	1	1
T	<22x2 double>	0	1
V	0	0	0
a	23	23	23
ans	0	0	0
b	1	1	1
cc	[23,4]	4	23
counter1	4	4	4
counter2	28	28	28
d	1	1	1
e	[15,16]	15	16
f	[2,2]	2	2
fid	3	3	3
first1x	15	15	15
first1y	322	322	322
firsttrue1y	322	322	322
firsttruex	13	13	13
firsttruexl	15	15	15
firsttruey	46	46	46
firsttrueyl	2	2	2
firstx	15	15	15
firstxl	15	15	15
firsty	322	322	322

```
Enter the font size:14

out =

MATLAB is a high-performance language for technical computing. lt integrates


out =

computation, visualization, and programming in an easy-to-use environment


out =

where problems and solutions are expressed in familiar mathematical notation.


out =

You can watch the Getting started with MATLAB video demo for an oveview


out =

of the major functionality.
```

Figure 26: Command Window output for the input image shown in figure 25.

As explained in the methodology that each line of the document is extracted and processed individually like it was a single line input, the command window output produced in the MATLAB is shows all lines of the input image as different output.



```
MATLAB is a high-performance language for technical computing. It integrates
computation, visualization, and programming in an easy-to-use environment
where problems and solutions are expressed in familiar mathematical notation.

You can watch the Getting started with MATLAB video demo for an oveview
of the major functionality.

If you have an active Internet connection, you can also watch the working in
the Development Environment video demo, and the writing
a MATLAB Program video demo.
```

Figure 27: The editable text file created after complete processing of the input image of the figure 25.

When each line of the input document containing multiple lines is processed individually, the output produced is exported to the text file, and as the program moves on to the next lines, it appends the recognized text to the text file. Hence after complete processing of the input image file shown in figure 25, the text file produced has all the data (multiple lines) exported to it.

CHAPTER 6

CONCLUSION AND FUTURE WORK

Simulations run to test the proposed method for the 52 alphabet characters (26 upper case and 26 lower case), 31 special characters (For example ~, !, @, etc.) and 10 digits shows positive results without any errors or loss of information. The experiment of character recognition has been performed while working with 'Arial' font having the font sizes 14, 16, 18 and 20. It is to be noted that once an alphabet is correctly recognized, 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.

The success of the project work lies in the high levels of accuracy obtained when the experiments are performed for a complete document containing numerous characters, words, lines or even paragraphs.

The project has been successfully completed within the stipulated time frame. We have achieved high accuracy on all results and have documented the project work in the form of a research paper. The future implications of this novel approach seem to be bright. The concept can be extended for recognition of sans serif fonts other than Arial, serif fonts and hand written text.

REFERENCES

1. 'Character Recognition using Geometrical Features of Alphabet: A Novel Technique' – Sunil Bhooshan, Vinay Kumar, Ateendra K. Singh, Tanupriya Negi, Jyoti Miglani, JUIT.

2. Scalisi, Corrie. MMCR: A Mathematical Character Recognition System for MATLAB, University of California, May 2006.

3. http://en.wikipedia.org/wiki/Optical_character_recognition

4. J. Mantas, "An Overview of Character Recognition Methodologies", Pattern Recognition, vol. 19, no. 6, pp. 425-430, 1986.

5. V. K. Govindan and A.P. Shivaprasad, "Character Recognition - A Review", Pattern Recognition, vol. 23, no. 7, pp. 671-683, 1990.

6. C. Y. Suen, "Character Recognition by Computer and Application", in Handbook of pattern recognition and image processing, pp. 569-586, 1986.

7. S. Mori, "A non-metric model of hand printed characters", Res. Electrotech. Lab. Tokyo, vol. 798, August 1979.