# OFFLINE SEGMENTATION OF MACHINE PRINTED GURMUKHI SCRIPT WITH EMPHASIS ON TOUCHING CHARACTERS

*A Dissertation*

*submitted in partial fulfilment*

*of the requirement for the award of degree of*

*Master of Engineering in Computer Science*


**Submitted by:**

*Anil Kr. Verma*

**Guided By:**

*Mr. G.S Lehal (Asstt. Prof.)*

*Department of Computer Science & Engg.*

*Thapar Institute of Engineering & Technology-Patiala*




*Department of Computer Science and Engineering*

*Thapar Institute of Engineering and Technology*

(Deemed University)

Patiala



MAY-2001

# CERTIFICATE

This is to certify that the dissertation entitled - "Offline Segmentation of Machine Printed Gurmukhi Script with emphasis on Touching Characters" submitted by Anil Kumar Verma {ME(P)-15/97(I)} in partial fulfillment of the requirement for the award of the degree of Master of Engineering in Computer Science in the department of Computer Science and Engineering, Thapar Institute of Engineering and Technology (Deemed University), Patiala is a record of candidate's own work carried out by him under my supervision and guidance. The matter embodied here has not been submitted in part or full to any other University or Institution for the award of any degree.

Gurpreet Singh 14/5/2001

*(G. S. Lehal)*
***Asstt. Prof.***
Deptt. of Computer Science & Engg.
Thapar Institute of Engineering and
Technology, PATIALA. 147 004

RK Sharma 18/5/01

*(Dr. R. K. Sharma)*
*Head*
Deptt. of Computer Science & Engg.
Thapar Instt. of Engg. & Tech.
PATIALA. 147 004

18.5.2001

*(Dr. D. S. Bawa)*
*Dean (Academic Affairs)*
Thapar Instt. of Engg. & Tech.
PATIALA. 147 004

# CONTENTS

# ACKNOWLEDGEMENT

The written words has a tendency to degenerate genuine gratitude into stated formality, but this is the only way I can express my feelings.

It gives me immense pleasure to acknowledge my deep sense of gratitude to my guide- ***Mr. G. S. Lehal*** (Asst. Prof., DCSE, TIET, Patiala) for his constant inspiration and editorial guidance which helped me to present this work in its present form. His superb guidance, constructive criticism helped me to give a systematic and rational approach to this report. I shall always be indebted to his learned and thoughtful guidance and supervision. I would fail in my duty if I don't avail this opportunity to thank ***Er. Ajay Goyal*** *(Dy. Director, Harayana State Electricity Regulation Commission)*, who first talked me into this field and guided me through the initial part of this work.

I wish to express my profound thanks to ***Dr. D S Bawa***, *(Head, Computer Centre, TIET, Patiala)* for helping me in acquiring valuable resources, facilities and providing a stimulating atmosphere and wonderful environment.

Last but not the least, I express my gratitude to my Parents, my wife Sunita and son Sankalp for bearing with me while I spent hours in the library or on computer, at times ignoring their demands.

Once again, my sincere thanks to all who aided me with their cooperation during the completion of this work.

*Thank You!*

Anil Kr. Verma

*(Anil Kumar Verma)*
Systems-Analyst-cum-Prog'r.
Computer Centre
TIET, Patiala

# ABSTRACT

Character segmentation is an important process of optical character recognition system. Success of character recognition depends very much on the success rate of segmentation. Touching characters are a major factor of error in segmentation. A lot of work has been done on segmentation for scripts like Roman (for English), Kanji (for Chinese) and Kana (for Japanese). But none of these is fully applicable to Gurmukhi script.

The OCR system development for Gurmukhi script is difficult because the characters in a word are topologically connected, two or more characters in a word may have intersecting minimum bounding rectangles, presence of multi-component characters and further the presence of touching characters make it even more harder.

In the proposed work, the document image captured by a flat-bed scanner is subjected to thinning (skeletonization), line segmentation, zone detection, word segmentation & character segmentation. An attempt is made to segment the touching characters in Gurmukhi script.



**Keywords:** OCR, Gurmukhi Script, Thinning (Skeletonization), Segmentation , Touching Characters.

# *Chapter 1*

## Introduction To OCR

### *1.1 Introduction*

In the quest of machines which can imitate the human behaviour, scientists, engineers and researchers make use of a wonderful machine known as computer (special thanks to Charles Babbage). These machines have made inroads in almost every sphere of human life such as- universities(academics), industry, research, engineering, scientific, medical etc.

One of the major abilities of the human brain is to recognize and manipulate the patterns, which is also a pre-requisite for a number of different applications in a human-computer interaction. For instance, communication applications rely on a pattern of bits to identify the sender, user authentication, user authorization on a computer network. There are other applications, which use patterns as a means of providing input and output (speech recognition). Further, the problem of replication of human functions by machines (computers) also involves the recognition of both machine printed and hand printed/cursive written characters. So, the character recognition techniques associate a symbolic identity with the image of a character.

In this direction, an effort has been made to study the new emerging pattern recognition techniques- better known as Optical Character Recognition (OCR). As we know that pattern recognition is fundamental to OCR[1]. First of all let us pay some attention to **Pattern Recognition**[1]:

*- A digital computer receives information about a plain black-and-white pattern, and is to decide on the basis of this information whether the pattern is "similar", in some sense to be specified below, to a given prototype. This process of correlating or identifying the given pattern with set patterns available and identifying the given pattern with respect to available pattern is called pattern recognition.*

The recognition of alphabets/numerals is of importance (this literature is understood by the reader if he can recognize the different patters) but there are other fields also in which pattern recognition is helpful:

- Finger print pattern classification
- ECG/MCG pattern detection
- Photo decoders
- Facial descriptors
- Touch screen terminals
- Character Recognition

### *1.2 Character Recognition*

Character recognition is an area of pattern recognition that has been subjected to a lot of research since the early attempt in 1958 by Grimsdale et al[2] and has migrated from research laboratories to industrial world and the personal computing environment.

*Character recognition or optical character recognition (OCR) is the process of converting optically scanned images of machine printed or hand written text-numerals, letters and symbols into a computer processable format (such as ASCII)*

Primitive OCR machines were mechanical devices with a very high failure rates, (1940-Jacob Rabinow)[3]. Modern OCR technology is said to have been born in 1951 with M Sheppard's GISMO- A Reader Writer Robot. In 1954, Jacob Rabinow[3] was able to design a prototype machine that was able to read uppercase typewritten letters at a fantastic speed of one character per minute. Later on by 1967, the OCRs were being marketed by several companies including IBM, Ferrington,Control Data, Optical Scanning Corporation to name a few. The preliminary research in the field of OCR led to the development of many standards, which are helpful even today. These standards included:

- Character Set for OCR (OCR-A & OCR-B)
- Paper used in OCR
- Inks used in OCR
- Character Position in OCR

As on date, the present OCR systems are less expensive , more reliable and more faster. So, it is not uncommon to find a PC based OCR system, which can recognize several hundreds characters per minute of different languages and that too with different font sizes and types. (ABBYY Co. has just released Fine Reader OCR ver4.0, which can recognize images containing many world languages and are omni-fontable)[4].

## *1.3 Components of OCR System:*

The complete OCR system can be divided into six different stages[5]:

- Digitization
- Pre-Processing
- Segmentation
- Recognition
- Post-Processing
- Output encoding

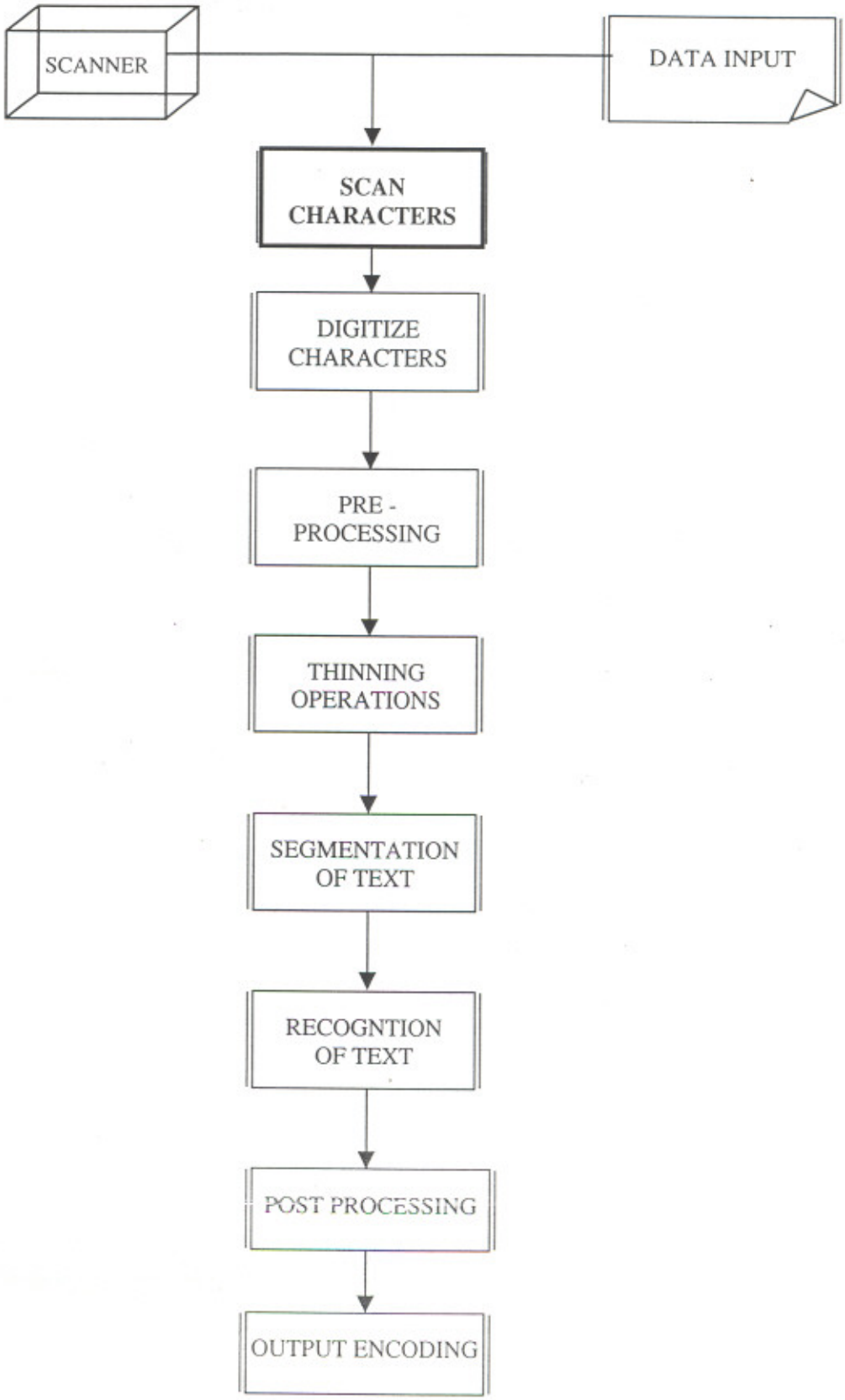



*Fig. 1.1 Stages in Optical Character Recognition (OCR)*

*i) Digitization:* It refers to the process of converting a paper- or film-based document into electronic form. The electronic conversion is accomplished through imaging a process whereby a document is scanned and an electronic representation of the original, in the form of bitmap image, is produced. The imaging process involves recording changes in light intensity reflected from the document as a matrix of dots (also called 'rasters'). The light/color value(s) of each dot is stored in binary digits. One bit would be required for each dot in a binay (B&W) scan, whereas upto 32 bits could be required per dot for a color scan.

*ii) Preprocessing:* There are some problems like slope and size correction, ambiguous characters or words, digital reasoning that do not match human perception etc. which are being taken care by pre-processing and post-processing before and after OCR is performed. Typical pre-processing includes:

- Noise removal
- Smoothing
- Skew correction and
- Skeletonization or Thinning

a) The major objective of *noise removal* is to remove any unwanted bit-pattern which do not have any significance in the output. Therefore, we are given a sharp output & our OCR system is also not overloaded.

b) The objective of *contour smoothing* is to smooth contours of broken and/or noisy input characters.

c) *Skewness* refers to the tilt in the bit-mapped image of the scanned paper for OCR. It is usually caused if the paper is not fed straight into the scanner. Most of the OCR algorithms are sensitive to the orientation (or skew) of the input document image making it necessary to develop algorithms- which can detect and correct the skew automatically. There are basically, three different techniques:

- Projection Profile
- Component based nearest neighbour clustering
- Hough transform

d) *Skeletonization or Thinning* refers to the process of reducing the width of a line like any object from many pixels wide to just a few pixels. This process can remove irregularities in letters and in turn make the recognition algorithm simpler because they only have to operate on a character stroke, which is only a pixel wide. It also reduces the memory space required for storing the information about the input characters and no doubt, this process reduces the processing time too. The common methods are:

- Methods based on iterative deletion of pixels (sequential & parallel thinning algorithms)
- Nonpixel-based methods (Run length encoding for thinning)

*iii) Segmentation:* It is an operation that seeks to decompose an image of sequence of characters into sub-images of individual symbols. Character segmentation is a key requirement that determines the utility of conventional OCR systems. Different methods used can be classified based on the type of text and strategy being followed like:

- Straight segmentation method
- Recognition - based segmentation
- Cut classification method

*iv) Recognition:* Once the characters are segmented, the next step is to recognise them. The recognition process, gives the actual character of the script, appearing in the text.

*v) Post Processing:* OCR results usually contain errors because of character classification and segmentation problems. For the correction of recognition errors, OCR systems apply contextual post-processing techniques. Contextual processing attempts to overcome the shortcoming of decisions made on the basis of local properties and to extend the perception on relationships between characters into word. The two most common post-processing techniques are dictionary lookup and statistical approach. The advantage of statistical approach over dictionary

based methods is computational time and memory utilisation. On the other hand, lexical knowledge about entire words is more accurate when using a dictionary.

*vi) Output encoding:* Mapping the internal representation of the analysis into character codes such as ASCII.

## *1.4 Advantages of OCR*

In general, dealing with big graphical text input images is a serious problem. For example, image size of a US letter size (8.5 in X 11in) with 300 dpi (dots per inch) is 8.4 Mbytes. If this image is to be transferred as a graphics file, the total time spent would be much more than if it is transferred as an ASCII file. Therefore we need to introduce an image representation which is capable of reducing the amount of data to be processed without any loss of information in a short time.

A document image can be displayed onscreen (with clearly legible text) or printed. To the computer, however, the image is not "readable"- it cannot be "understood" as anything other than dots. Let us consider a case of dealing with OCR which will make the text machine-readable and should therefore be considered if:

- the text is to be reused, edited or reformatted.
- the text should be available for full-text information retrieval (e.g., by Internet search engines)
- the text is to be coded in HTML
- the text should be available to adaptive equipment for the visually impaired.
- file size is of concern (in terms of storage or bandwidth to transmit)
- resources are available to perform OCR and correct the output.

| Characteristic | Bitmapped | OCR text |
|---|---|---|
| **Size of file** | Large | Fraction (e.g. <5%) of size of bitmapped image |
| **Original formating** | Retained | Lost if ASCII; retained in part or entirely with some other output formats |
| **Editing and reformatting** | Not possible | Possible |
| **Information retrieval (Indexing)** | Not possible | Possible |
| **Reproduction accuracy** | Provides a facsimile of the original | Not a true representation of original |
| **Effect of processing the image** | while it can be manipulated (e.g. sharpened), hence provides a true representation of the appearance of the original. | Processing can affect both text and format accuracy |

*Fig 1.2 Comparison of different characteristics of Bitmapped image Vs OCR text*

## *1.5 Applications of OCR*

Commercial OCR systems can be grouped into two categories:

*i) Task -Specific Readers*

*ii) General Purpose Page Readers*

Hundreds of OCR systems have been developed since the 1950s and many are commercially available today. Commercial OCR systems can largely be grouped into two categories: *task-specific readers* and *general purpose page readers*. A task specific reader handles only specific documents types. Some of the most common task-specific readers read bank cheque, letter mail, or credit card slips. These readers usually utilise custom-made image lift hardware that captures only a few predefined document regions. For example, a bank cheque reader may just scan the courtesy amount field and a postal OCR system may just scan the address block on a mail piece. Such systems emphasize high throughput rates and low error rates. Applications such as letter mail reading have throughput rates of 12 letters per second with error rates less than 2%. The character recognizer in many of task-specific readers is able recognize both handwritten and machine printed text.

### 1.5.1 *Task-specific readers*

Task-specific readers are used primarily for high-volume applications, which require high system throughput. Since high throughput rates are desired, handling only the fields of interest helps reduce time constraints. Since similar documents possess similar size and layout structure, it is straightforward for the image scanner to focus on those fields where the desired information lies. This approach can considerably reduce the image processing and text recognition time. Some application areas to which task-specific readers have been applied include:

- Assigning ZIP Codes to letter mail.
- Reading data entered in forms e.g., tax forms
- Automatic accounting procedures used in processing utility bills
- Verification of account numbers of courtesy amounts on bank checks
- Automatic accounting of airline passenger tickets
- Automatic validation of passports

#### *1.5.1a) Address Readers*

The address reader in a postal mail sorter locates the destination address block on a mail piece and reads the ZIP Code in this address block. If additional fields in the address block are read with high confidence the system may generate a 9 digit ZIP Code for the piece. The resulting ZIP Code is used to generate a bar code, which is prayed on the envelope.

The *Multilane Optical Character Reader* (MLOCR) used by the United States Postal Services(USPS) locates the address the address block on a mail piece, reads the whole address, identifies the ZIP+4 code, generates 9-digit bar code and sorts the mail to the correct stacker. The character classifier that recognizes up to 400 fonts and that the system can process up to 45,000 mail pieces per hour.

#### *1.5.1b) Form Readers*

A form reading system needs to discriminate between pre-printed form instructions and filled- in data. The system is first trained with a blank form. The system registers those areas on the form where the data should be printed. During

the form recognition phase, the system uses the spatial information obtained from training to scan the regions that should be filled with data. Some readers read hand-printed data as well as various machine written text. They can read data on a form without being confused with the form instructions. Some systems can process forms at a rate of 5,8000 forms per hour.

*1.5.1c) Cheque Readers*

A cheque reader captures cheque images and recognizes courtesy amounts and account information on the cheques . Some readers also recognize the legal amount on cheques and use the information in both fields to crosscheck the recognition results. An operator can correct misclassified characters by cross-validating the recognition results with the cheque image that appears on a system console.

*1.5.1d) Bill Processing Systems*

In general a bill processing system is used to read payment slips, utility bills and inventory documents. The system focuses on certain regions on a documents where the expected information are located, example-account number and payment value.

1.5.1e) Airline Ticket Readers

In order to claim revenue from airline passenger ticket, an airline needs to have three records matched: reservation record, the travel agent record and the passenger ticket. However, it is impossible to match all three records for every tickets sold. Current methods, which use manual random sampling of tickets, are far from accurate in claiming the maximal amount of revenue.

Several airlines are using a passenger revenue accounting system to account accurately for passenger revenues. The system reads the ticket number on passenger ticket and matches it with he one in the airline reservation database. It scans tickets up to 260,000 tickets per day and achieve a sorting rate of 17 tickets per second.

*1.5.1f) Passport Readers*

An automated passport reader is used to speed the returning passengers through custom inspections. The Passport Reader reads a traveller's name, date of birth, and passport number on the passport and checks these against the database records that contain information on fugitive felons and smugglers.

*1.5.2 General-purpose page readers*

General-purpose page readers are designed to handle a broader range of documents such as business letters, technical writings and newspapers. These systems capture an image of a document page and separate the page into text regions and non-text regions. Non-text regions such as graphics and line drawings are often saved separately from the text and associated recognition results. Text regions are segmented into lines, words, and characters and the characters are passed to the recognizer. Recognition results are output in a format that can be post-processed by application software. Most of these page readers can read machine written text but only a few can read hand-printed alphanumerals.

There are two general categories of page readers:

*1.5.2a) High-end page readers, and*

*1.5.2b) Low -end page readers*

High-end page readers are more advanced in recognition capability and higher data throughput than the low-end page readers. Low-end page readers are mostly used in an office environment with desk top workstations, which are less demanding in system throughput. Since they are designed to handle a broader range of documents, a sacrifice of recognition accuracy has to be made. Some commercial OCR software allow users to adapt the recognition engine to customer data for improving recognition accuracy. Some high performance readers can detect typefaces (such as bold face and italic) and output the formatted ASCII text in the corresponding style.

# *Chapter 2*

## Literature Survey

### *2.1 Preprocessing*

There are some problems like slope and size correction, ambiguous characters or words, digital reasoning that do not match human perception etc. which are being taken care by pre-processing and post-processing before and after OCR is performed. In this chapter a brief overview of the work done in pre-processing & segmentation is given.

### *2.2 Thinning*

Thinning is the process of reducing the width of a pattern from many pixels to just a single pixel. This process is sometimes also called as *skeletonization* and the thinned pattern is called the *skeleton*. Skeletons are useful for topological analysis and classification of the shape of pattern containing elongated or line-like parts (for example, printed or handwritten characters, chromosomes etc.) Other terms used to describe the process of thinning are "medial axis transformation" and "symmetric axis transformation".

Character recognition systems require thinning for extracting the basic features of the characters. Other advantages of thinning or skeletonization are :

- to reduce the memory space required to store the essential structural information of the pattern.
- to simplify the data structure required in processing the pattern
- to some extent, the thinning process provides a unification of character shapes by reducing the effects of varying types of fonts.

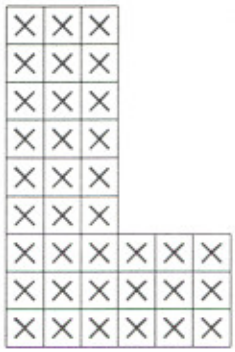


*Fig. 2.1 Scanned input bitmap*

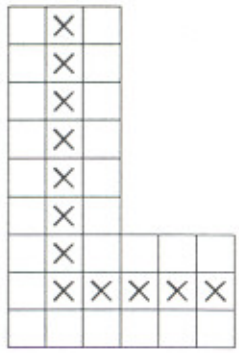


*Fig. 2.2 Thinned image bitmap*

To illustrate the concept, consider Fig. 2.1, which is a typical binary image of letter "L" where an analogue raster scan image has been converted to areas of either "black" or "white". the process of converting an analogue or multilevel (different grey levels) image to black and white is known as thresholding. Starting from this image we wish to obtain figure 2.2, which is the skeleton of Fig. 2.1 where the image is still in the form of bit-map but only the skeletal pixels remains.

### *2.2.1 Methods for Thinning*

Many thinning algorithms have been proposed in the past two decades. Different thinning algorithms produce different kinds of skeletons and distortion. Suen et al.[5] surveyed and discussed a wide range of thinning methods and classified them into two main group

- ***Methods based on iterative deletion of pixels***
- ***Nonpixel-based methods (Run Length Encoding)***

#### *2.2.1a) Methods based on Iterative deletion of pixels*

These algorithms delete successive layer of pixels on the boundary of the pattern until only a skeleton remains. The deletion or retention of (black) pixel p would depend on the configuration of pixels in a local neighbourhood containing p. According to the way they examine pixels, these algorithms can be classified as sequential or parallel. In a sequential algorithm, the pixels are examined for deletion in a fixed sequence in each iteration, and the deletion of p in the nth iteration depends on all the operations that have been performed so far, i.e. on the result of the (n-1)th iteration as well as on the pixels already processed in the nth

iteration would depend only on the result that remain after the (n-1)th, therefore, all pixels can be examined independenlty in a parallel manner in each iteration.

Many methods based on iterative of pixels employ a window operator for thining. These methods place a 3 x 3, 5x 5 or a larger nxn window onto the image and then use a look up table to determine whether to retain or delete the central black pixel.

Some of the thinning algorithms based on Iterative approach are:

*i) Sequential Thinning:* Using sequential algorithms, contour points are examined for deletion in a predetermined order, which can be accomplished by either raster scanning or by the contour following. The Contour following algorithms can visit every border pixel of a simply connected object, and of a multiple connected picture. These algorithms require the examination of only the contour pixels instead of all the pixels in P, in every iteration.

Some of the contour tracing algorithms can be seen in [7],[8],[9] and the contours are traced using the freeman chain codes in [10].

*ii) Parallel Thinning Algorithms:* In parallel thinning, pixels are examined for deletion based on the result of only the previous iteration. For this reason, these algorithms are suitable for implementation on parallel processors where the pixels satisfying a set of conditions can be removed simultaneously. Unfortunately, fully parallel algorithms can have difficulty preserving the connectedness of an image if only 3x3 supports are considered for eg: a horizontal rectangle two pixels in width may completely vanish in such a thinning process. Therefore, the usual practice is to use a 3x3 neighbourhoods and divide each iteration into sub-iterations or subcycles in which only a subset of contour pixels are considered for removal. At the end of each sub-iteration, the remaining image is updated for next iteration. Four subcycles can be used in which each type of contour point (north, east, south and west) is removed in each sub-cycle [10]. These have also been combined into two sub-iterations [11], [12], [13] with one sub iteration deleting the north and east contour and the other deleting the rest.

Iterative methods cannot be simply ignored because of their slow speed, nowadays the computers are less expensive with large RAM and a processing speed greater than 500MHz as compared the old computers with a 640K RAM and a speed of 16 MHz. Moreover, use of a higher cache memory can further enhance the performance.

*2.2.1b) Nonpixel-based methods (Run length encoding for thinning)*

These algorithms [14], [15] produce a certain median or centre line on the pattern directly in one pass without examining all the individual pixels. One type of non-pixel based method is to employ run length encoding for thinning. It has been argued that this is the way human beings would perform thinning and that is possible to retain global features and maintain connectedness in the process.

This method divides a character into a number of segments. These segments are their connections, which are used to determine how the segments will be thinned. These methods work well for approximately constant width of all the lines in a character. Performance of these methods is good only for object in the image with line or curves of similar width. Heuristic rules are used to find the intersections of the skeletons. So these methods are good for Chinese character, because Chinese characters are composed of vertical lines, horizontal lines and slanted lines. But the presence of noise, connected strokes, touching strokes and variations in width along the length of a stroke create segment combinations of many different shapes and hence greatly increase the difficulty of using run length encoding to perform thinning.

## *2.3 Character Segmentation*

For the Convenience of recognition, the OCR system should automatically detect individual text lines and as well as segment the words and then the characters accurately.

***Character Segmentation is an operation that seeks to decompose an image of a sequence of characters into sub images of individual symbols. It is one of the decision processes in a system for OCR .***

Its decision, that a pattern isolated from the image is that of a character (or some other identifiable unit), can be right or wrong. If it is wrong, that means that it makes a major contribution to the error rate of the system. Character segmentation is fundamental to character recognition approach, which rely on isolated characters. It is a critical step because an incorrectly segmented character will not be recognised correctly.

### *2.3.1 Importance of Segmentation*

As shown in the following fig. 2.3, segmentation is the initial step in a three-step procedure in the "classical" OCR approach.

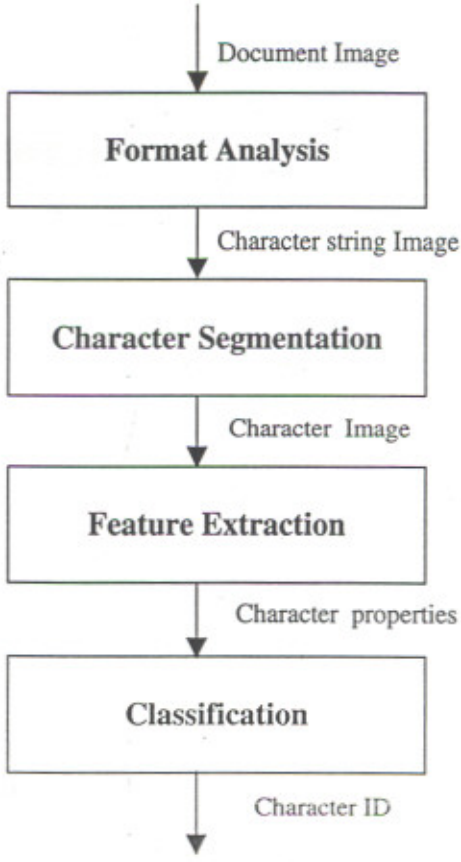



*Fig 2.3 : Classsical Optical Character Recognition (OCR)*

Given a starting point in a document image:

i) Find the next character image

ii) Extract distinguishing attributes of the character image.

iii) Find the member of a given symbol set whose attributes best match those of the input, and output its identity.

This sequence is repeated until no additional character images are found. The process of character recognition consists of a series of stages, with each stage passing its result on to the next in a pipeline fashion. There is no feedback loop that permit an earlier stage to make use of knowledge gained at a later point in the process.

The segmentation decision is not a local decision, independent of previous and subsequent decisions. Producing a good match to a library symbol is necessary, but not sufficient, for reliable recognition. Which means that a poor match on a later pattern can cast doubt on the correctness of the current segmentation/recognition result. Even a series of satisfactory pattern matches can be judged incorrectly if the contextual requirements on the system output are not satisfied. For instance- the letter "cl" closely resemble the alphabet "d", but usually such a choice will not constitute a contextually valid result. Thus, it is seen that the segmentation decision is interdependent with local decisions regarding shape similarity, and with global decisions regarding contextual acceptability.

### *2.3.2 Segmentation Methods:*

When we consider the textual processing we need to consider two different flavours for the segmentation viz., segmentation of machine printed words & segmentation of handwritten words. But, the main difference lies in the method(s) used for classification.

Tappert et al.[16], deals with *"external segmentation"* & *"internal segmentation"*, depending on whether recognition is required in the process. Further, Dunn and Wang [18], use *"straight segmentation"* and *"segmentation-recognition"* for a similar dichotomization. This discussion according to use or non-use of recognition in the segmentation process fails to make clear the fundamental

distinctions among the present day approaches. For instance, it is common to use a spell-checker/corrector as a post-processor in a text based recognition.

According to the survey done by Casey and Lecolinet [17], there are three "pure" strategies for segmentation alongwith numerous "hybrid" approaches, which are nothing but are weighted combination of these three strategies. The distinction between these methods is based on how the segmentation & classification interact in overall process. These strategies are:

- Dissection strategies
- Holistic Methods (Global approach)
- Recognition-based Segmentation (Segmentation based approaches)
- Classical Approach (Hybrid approach)

*2.3.2a) Dissection strategies* - The meaning of the term *dissection* is to decompose the image into a sequence of sub-images using general features like approximate character size, pitch, white space etc.

*2.3.2b) Global Approaches - HOLISTIC (also known as cut-classification method)*
These approaches recognize a word as a whole entity. The classification features are extracted from the entire word image, such as the number of vertical strokes, ascenders, descenders and loops, etc. based on these a decision is made according to the word classifiers.

*2.3.2c) Recognition based Approaches*

A word image is first partitioned into a sequence of segments, the segments are recognised by character classifiers, and then the recognized segments are matched with the possible word.

*2.3.2d) Hybrid Approaches*

These approaches combine both of the above methods (Global Approaches and Segmentation based Approaches) into the segmentation-recognition process. As shown in fig 2.4, these three fundamental strategies occupy orthogonal axes Hybrid method can be represented as weighted combinations of the points lying in

the intervening space. There is a continuous space of segmentation strategies rather than a discrete set of classes with well defined boundaries. Of-course, such a space exists only to assign precise weights to the elements of a particular combination.

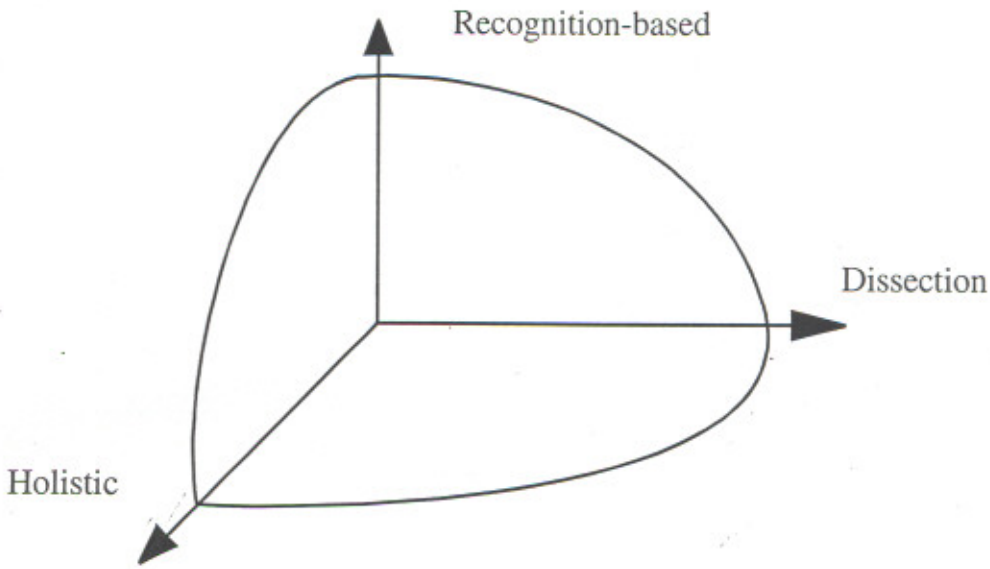



*Fig. 2.4 The space of segmentation strategies (Casey et al[17])*

Recognition based on the entire word is difficult and complicated. It is also time and memory intensive since all the representations of the words in the vocabulary under investigation need to be kept and matched with the input word image. Another drawback is that an incorrectly spelled word cannot be recognised because they do not belong to the vocabulary.

In most existing OCR systems, the word recognition algorithm fall in the last two categories in which character segmentation plays a critical role.

Fig. 2.5, below, depicts some fonts in machine printed text[19].

a) Uniformly spaced characters

b) Well seperated, unbroken characters

c) Broken characters

d) Touching Characters

e) Broken and Touching Characters

f) Broken and Italic Characters

g) Touching and Italic Characters

h) Script Characters

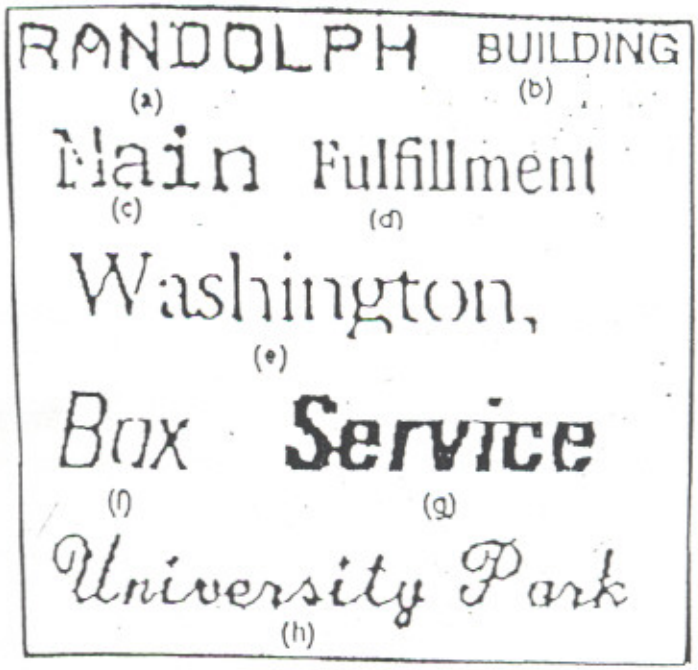



*Fig. 2.5 Machine printed text (Kahan et al[13])*

## 2.4 DISSECTION TECHNIQUES FOR SEGMENTATION

The meaning of the term *dissection* is to decompose the image into a sequence of sub-images using general features like approximate character size, pitch, white space etc.

### *2.4.1 Dissection Directly Into Characters*

In the 1950s and early 1960s, the earliest attempts to automate character recognition, research was focused on the identification of isolated images. Workers mainly sought methods to characterize and classify character shapes. In

some cases, individual characters were mapped onto grids and pixel co-ordinates entered on punched cards, in order to conduct controlled development and testing. As CRT scanners and magnetic storage become available, well separated characters were scanned, segmented by dissection based on white space measurement, and stored on tape. When experimental devices were built to read actual documents, they dealt with constrained printing or writing that facilitated segmentation. For example-bank cheque fonts were designed with strong leading edge features that indicated when a character was properly registered in the rectangular array, from which it was recognized. Hand-printed characters were printed in boxes that were invisible to the scanner, or else the writer was constrained in ways that aided both segmentation and recognition. Following are some methods based primarily on *dissection*:

*2.4.1a) White space and Pitch*

In machine printing, vertical white space often serves to separate successive characters. This property can be extended to handprint by providing separated boxes in which to print individual symbols. In applications such as billing, where document layout is specially designed for OCR, additional spacing is built into the fonts used. The notion of detecting the vertical white space between successive characters has naturally been an important concept in dissection images of machine print or handprint.

In many machine print applications involving limited font sets, each character occupies a block of fixed width. The pitch, or number of characters per unit of horizontal distance, provides a basis for estimating segmentation points. The sequence of segmentation points obtained for a given line of print should be approximately equally spaced at the distance corresponding to the pitch. This provides a global basis for segmentation, since separation points are not independent.

Applying this rule permits correct segmentation in cases when several characters along the line are merged or broken. If most segmentations can be obtained by finding columns of white, the regular grid of intercharacter boundaries can be estimated. Segmentation points not lying near these boundaries can be rejected as probably due to broken characters. Segmentation points missed due to merged

characters can be estimated as well, and a local analysis conducted in order to decide where best to split the composite pattern.

*2.4.1b) Projection Analysis*

The vertical projection (also called the vertical histogram) of a print line (Fig. 2.6)

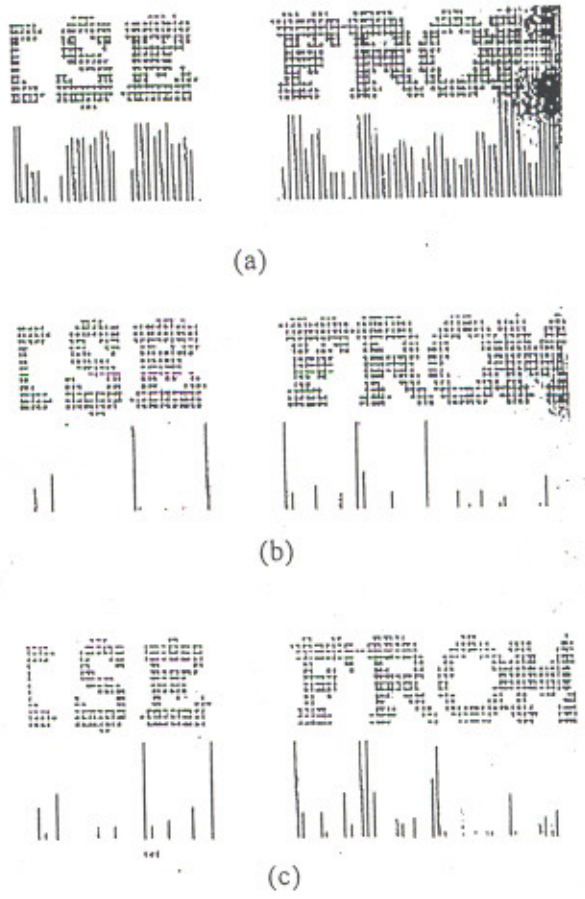


Fig 2.6 Vertical Histogram *(Casey et al[17])*

Fig. 2.6 (a) consists of a simple running count of black pixels in each column. It can serve for detection of white space between successive letters. Moreover, it can indicate locations of vertical strokes in machine printed, or regions of multiple lines in handprint. But it fails to clear the O-M seperation. Then in fig. 2.6(b) a function is used which gives second difference of the projections. This gives a clear peak at most seperation columns, but may still fail for the O-M case. In fig.

2.6(c) image is transformed by an AND of adjacent columns, with the differencing function of fig.2.6(b) applied to the transformed image. The AND operation tends to increase seperation, leading to a better defined peak at the O-M boundary.

*2.4.1c) Connected Component Processing*

Projection methods are primarily useful for good quality machine printing, where adjacent characters can ordinarily be seperated at columns. A one dimensional analysis is feasible in such a case.

However, the methods described above are not generally adequate for segmentation of proportional fonts or hand-printed characters. Thus, pitch based methods cannot be applied when the width of characters is variable. Likewise, projection analysis has limited success when characters are slanted, or when inter-character connections and inter character strokes have similar thickness.

Segmentation of machine printed or kerned machine printing calls for a two dimensional analysis, for even non-touching characters may not be seperable along a straight line. A common approach (see fig. 2.7, below) is based on determining connected black regions *(connected components or blobs).*

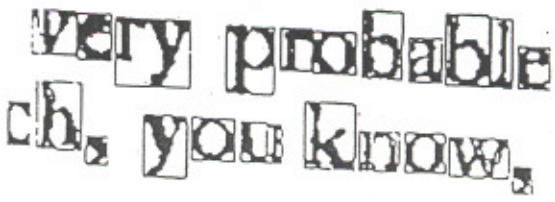



*Fig. 2.7 Connected Components (Casey et al[17])*

Further processing may then be necessary to combine or split these components into character images. There are two types of follow-up processings. One is based

on the *bounding box*, i.e. the location and dimensions of each connected component. The other is based on detailed analysis of the images of the connected components.

*i)* *Bounding Box Analysis:*

The distribution of bounding boxes tells us a great deal about the proper segmentation of an image consisting of non-cursive characters. By testing their adjacency relationships to perform merging, or their size and aspect ratios to trigger the splitting mechanism. In this approach, much of the segmentation task can be accurately performed at a low cost in computation.

Further, this approach has been successfully applied in segmenting handwritten post codes using knowledge of the number of symbols in the code. Connected components were joined or split according to rules based on height and width of their bounding boxes. This approach was able to achieve a result of 93%, with incorrect segmentation of 2.7% and rejection of 4.3%.

*ii) Splitting of connected components:*

Analysis of projections or bounding boxes offers an efficient way to segment non-touching characters in hand or machine printed data. However, more detailed processing is necessary in order to separate the joined characters reliably.

- The intersection of two images can give rise to special image features. Consequently, dissection methods have been developed to detect these features and to use them in splitting a character string image into sub-images.
- Another concern is that segmentation along a straight line path can be inaccurate when the writing is slanted or when characters are overlapping.

Accurate segmentation calls for an analysis of the shape of the pattern to be split, together with the determination of an appropriate segmentation path. For example-dissection based on search and deflect is as shown in fig. 2.8. The initial path of cut trajectory is along a column corresponding to a minimum in the upper profile of the image. When a black pixel is encountered the path is modified recursively, seeking better positions at which to enforce a cut. In this way, multiple cuts can be made at positions which are in the shadow region of the image.

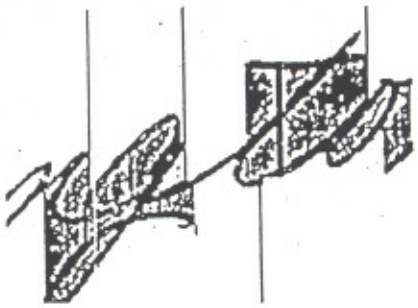


Fig. 2.8 Dissection based on Search and Deflect *(Casey et al[17])*

Depending upon the application, certain a priori knowledge may be used in the splitting process. For example-an algorithm may assume that some but not all input characters can be connected. in an application, such as form data, the characters may be known to digits or capital letters, thus placing a constraint on dimensional variations.

### *2.4.2 Dissection with contextual Processing: Graphemes*

The segmentation obtained by dissection can later be subjected to evaluation based on linguistic contexts, or alternatively, the input image can be divided into sub images that are not necessarily individual characters. The dissection is performed at stable image features that may occur within or between characters, as for example-a sharp downward indentation can occur in the centre of an **M** or at the connection of two touching characters. The preliminary shapes, called *graphemes* or *pseudo-characters* as shown in fig. 2.9, are intended to fall into readily identifiable classes. A contextual mapping function from grapheme classes to symbols can then complete the recognition process.

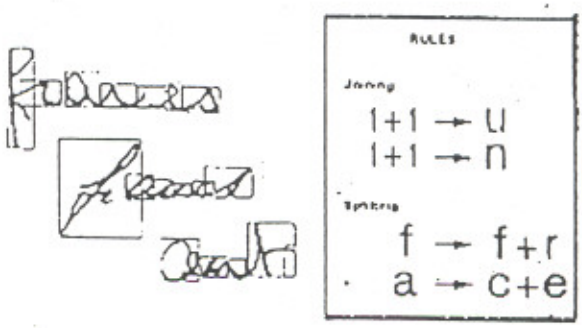



Fig. 2.9 Bounding Boxes *(Casey et al[17])*

For example-as shown in fig. 2.9, the bounding boxes indicate sub images isolated by a cutting algorithm. The algorithm typically cuts **u** or **n** into two pieces as shown, but this is anticipated by the contextual recombination rules at the right. Other rules are shown for cases where the cutting algorithm failed to split two characters. These rules are applied in many combinations seeking to transform the grapheme classes obtained during recognition into a valid character string.

The line segmentation that forms connections between characters in cursive script are known as *ligatures.* Thus, some dissection techniques for script seek *lower ligatures*, connections near the base line that links most lower-case characters. A simple way to locate *ligatures* is to detect the minima of the upper outline of words. Unfortunately, this method several problems unresolved.

- Letters such as **o, b, v** and **w** are usually followed by *upper* ligatures.
- Letters **u** and **w** contain *intraletter* ligatures, i.e. a subpart of these letters can not be differentiated from a ligatures in the absence of context.
- Artifacts some times cause erroneous segmentation.

In typical systems, these problems are treated at a later contextual stage that jointly treats both the segmentation and the recognition errors.

## 2.5 Recognition Based Segmentation

In this technique, no feature based dissection algorithm is employed. Rather, the image is divided systematically into many overlapping pieces without regard to content. These are classified as part of an attempt to find a coherent segmentation recognition result. Systems using such principle perform *recognition-based* segmentation. Letter segmentation is a byproduct of letter recognition, which may itself by contextual analysis. The main interest of this category of methods is that they bypass the segmentation problem.

For example-let us consider the results of such an algorithm that performs recursive segmentation based on recognition (see fig. 2.10). This example shows the results of applying windows of decreasing width to the left side of an input image. When the sub image in the window is recognized (in this case by matching a prototype character stored in the system's memory), then the procedure is recursively applied to the residue image. Recognition (& segmentation) is accomplished if a complete series of matching windows is found.

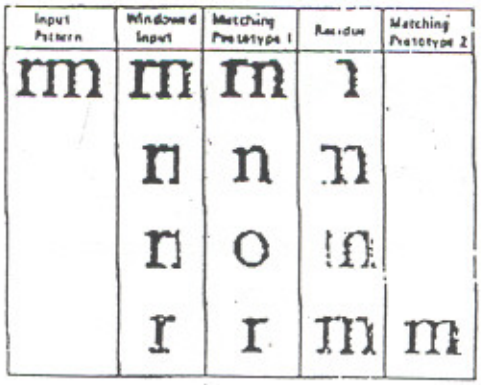



*Fig. 2.10 Recursive Segmentation*

In the top three rows, no match is obtained for the residue image, but successful segmentation is finally obtained as shown at the bottom.

The method of *selective attention* uses neural networks in the handling of segmentation problems. In this approach, a neural net seeks recognizable patterns in an image input, but is inhibited automatically after recognition in order to ignore the region of the recognized character and search for new character images in neighbouring regions

## 2.6 Mixed Strategies: Over-Segmenting

Two radically different segmentation techniques have been considered to this point. One (section 2.3) attempts to choose the correct segmentation points (at least for generation of graphemes) by a general analysis of image features. The other strategy (section 2.4) is at the opposite extreme. No dissection is carried out. Classification algorithms simply do a form of model-matching against image contents.

In this section, intermediate approaches, essentially hybrids of the first two are discussed. This family of methods also uses *pre segmenting*, with requirements that are not as strong as in the grapheme approach. A dissection algorithm is applied to the image, but the intent is to *over-segment*, i.e., to cut the image in sufficiently many places that the correct segmentation boundaries are included among the cuts made, as in Fig. 2.11. below.

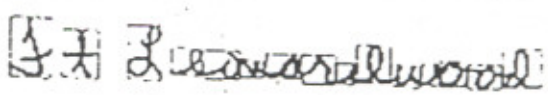


*Fig. 2.11 Letters containing Valley Dissected into multiple parts (Casey et at[17])*

## 2.7 Holistic Strategies

A holistic process recognizes an entire word as a unit. A major drawback of this class of methods is that their use is usually restricted to a predefined lexicon. Since they do not deal directly with letters but only with words, recognition is necessarily constrained to a specific lexicon of words. This point is especially critical when training on word samples is required. A training stage is thus mandatory to expand or modify the lexicon of possible words. This property makes this kind of method more suitable for applications where the lexicon is statically defined (and not likely to change), like cheque recognition.
These methods, usually follow a two step process.

- The first step performs feature extraction.
- The second step performs global recognition by comparing the representation of the unknown word with those of the references stored in the lexicon.

For example-as shown in Fig. 2.12, recognition consists of comparing a lexicon of word descriptions against a sequence of features obtained from an un-segmented word image. The detected features, shown below the images, include loops, oriented strokes, ascenders and descenders.

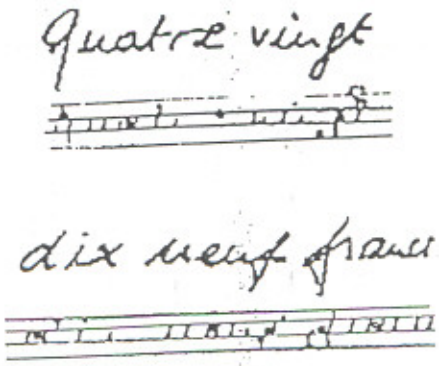


*Fig. 2.12 (Casey et al[17])*

Casey et al[17] have made a conclusion from their survey of methods and strategies of character segmentation that,

1 Missegmentation rates for unconstrained material increase progressively from machineprint ot handprint to cursive writing.

2 Simple techniques based on white separations, between characters suffice for clean fixed pitch typewriting, because of (1) above.

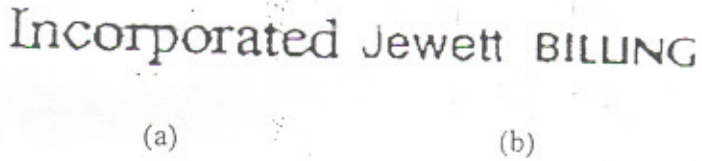
## 2.8 Touching Characters Segmentation Techniques

Merged characters, sometimes referred to as composite characters, are components of connected multiple characters. Segmentation of touching characters has been the most difficult problem in character segmentation. There are two key issues involved in this problem[19]:

- Determine which segments contain multiple characters, and
- find break locations

### *2.8.1 Discriminating Multiple Character Components:*

The most commonly used feature for detecting multiple character segment is the width and the aspect ration of the character. Lu[19], suggested that character width can be dynamically estimated during the segmentation process. Each candidate segment is then examined by comparing its width with the estimated character width or by measuring the aspect ration of the segment. It is well understood that most characters have widths smaller than their height and a single character segment should have a width of less then twice of the estimated character width. The combination of these two measurements works well on characters in uniform spacing and most characters in proportional fonts, but fail in special cases. Fig. 2.13 shows examples of touching character segments, "In", "rp", "ed", that are wider than single characters in the image, and so we can identify them by using either the segment width or the aspect ratio.




(a) (b)

*Fig. 2.13 Touching characters (Dunn et al[18])*

However in Fig. 2.13b the touching character segment "tt" is narrower than "J" and "w", and the touching character segment "LI" is no wider than "N" and "G" in the same string. Furthermore, the aspect ratios of "tt" and "LI" are as normal as a single character segment.

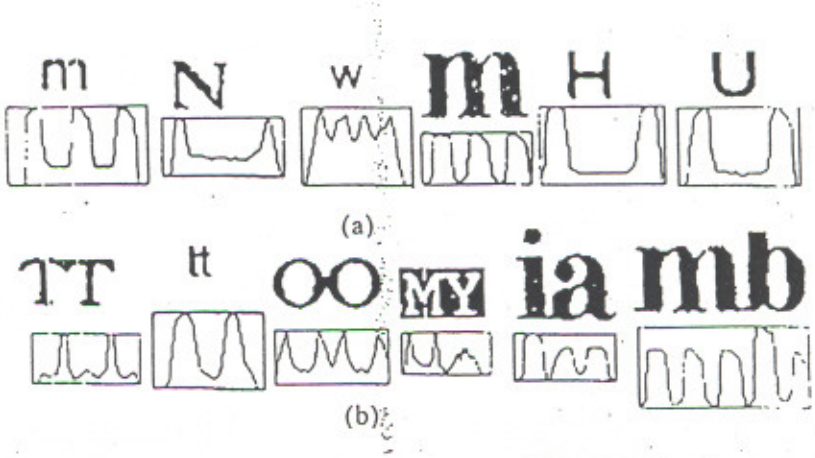



*Fig. 2.14 Multi character Segmentation from single character (Dunn et al[18])*

Lu further proposed generating single and multiple character profile models for discriminating multi character segments from single character segments. The profiles of characters are constructed from smooth vertical projection contour. Fig.2.14 shows the profiles of some single and touching character. A projection contour of multiple characters as at least two significant peaks in their projection contour H, n, N, u, and U also have two significant peaks in their projection contour, and characters m, w, M, and N have two or more significant peaks, depending on fonts. Since there are only a handful of such characters and the projection contours of such single characters are different from multiple characters, Lu suggested constructing profile methods for the characters with two or more significant peaks in their projection contour. The features used in the model construction are number of peaks, height and spread of peaks, width, depth and smoothness of the valleys, and the symmetry of the smoothed vertical projection curves.

Discrimination of touching character pairs from single characters can also done using neural networks[20], where the network is trained on a large database of touching characters (created artificially)

### 2.8.2 *Finding Break Locations in Touching Characters:*

After discrimination, the key issue in the splitting process is to determine where to split the multi character regions. This is done by using techniques like projection analysis (discussed section 2.4) and shortest path segmentation[20].

## 2.9 Character Segmentation of Gurmukhi Script

There are some papers dealing with segmentation of Gurmukhi Script[21-25]. Lehal & Parminder[21] have performed segmentation of Gurmukhi script by connected component analysis of a word assuming the headline not being a part of the word.

Another work on segmentation of Gurmukhi Script is by Madan[22], the author segments the characters after performing the skew-correction. The segmentation has been carried by the shortest-cut method.

Goyal et al[23] have suggested a dissection based Gurmukhi character segmentation method, which segments the character in the different zones of a word by examining the vertical white-space. A preliminary work was done by Goyal et al[24] using neural network for the recognition of Gurmukhi text.

Later on, Lehal and Singh[25], proposed a feature extraction and hybrid classification scheme, using binary decision tree and nearest neighbour, for machine recognition of Gurmukhi script.

A notable contribution has been by Lehal and Singh[26], in which, the segmentation of touching characters has been done for the lower-zone using the linked list representation of the given text, but the segmentation of touching characters in upper-zone & mid-zone were not dealt with.

### 2.9.1 *Segmentation in related Scripts (Bangla & Devnagri):*

One of the notable works published in the Bangla script is by Pal & Chaudhary[27], later on Pal & Chaudhary[28] worked on the segmentatation of Devnagari Script. But, no attention is given to touching characters.

Although, Veena[29] has dealt with touching characters in Devanagri Script. She has referred to Conjuct images by the image (Conj_left, Conj_right, Conj_top, Conj_bottom), where these are the Left, Right, Top and Bottom coordinates of the minimum sized upright bounding box of the image. The problem of touching characters is dealt with the help of following parameters:- vertical projection, horizontal projection, continuity of the collapsed horizontal projection, vertical bar & its location, pen-width. The characters are segmented using the hybrid-approach.

### 2.9.2 *Touching Characters Problem In Gurmukhi Script:*

In Gurmukhi also, like other scripts, touching characters are a major source of error. Some work on touching character problem for Gurmukhi script has been done by Madan[22] and Lehal and Singh[26]. It has been observed that, mostly, the touching character of Gurmukhi are of the form (Fig. 2.15)

*Fig. 2.15 Touching characters in Gurmukhi script*

# *Chapter 3*

## Methodology

The work done, can be divided into three phases, namely :

- Thinning
- Segmentation
- Segmentation of touching characters

### *3.1 Thinning*

A simple algorithm proposed by Zhang and Suen[30], consists of two sub-iterations: one for removing the south-east boundary points and the north-west corner points while the other one for deleting the north-west boundary points and the south-east corner points. In each sub-iteration, a boundary point is removed if it satisfies all the following conditions:

1. There is exactly one white-to dark transition is its 8-neighbours.
2. The number of neighbouring points (which are object points) should be between 2 and 6.
3. The 4-neighbours satisfy two predefined Boolean expressions for selecting the sides.

Lu and Wang[31] improved this algorithm by modifying condition 2 so that number of neighbouring object points is between 3 and 6. Pavlidis[8] proposed another thinning algorithm which can be used to determine skeletal points by local operators and original image can be reconstructed from its skeleton and certain information.

The Safe Point Thinning Algorithm (SPTA) uses rules which are optimised into a set of Boolean expressions which can be evaluated using the neighbours of each point. A set of timings is given for 648 patterns showing that the SPTA is twice as

fast as most similar methods. However, the skeleton suffers from a bending of straight lines at "T" junctions forming more of a "Y" shape.

From Zhang and Suen algorithm (parallel) and some idea from Suen algorithm (sequential) an effort has been made to understand the sequential and parallel algorithm for thinning is as follows.

Region points are assumed to have value 1 and background points to have value 0.

*STEP 1*: The first iteration uses a 4x4 window to delete single pixel (may be noise or part of the text) and four pixels as square in the centre of 4x4 and information about these pixels is saved, which will be used at the time of recognition of characters.

| | | | |
|---|---|---|---|
| P9 | P2 | P3 | P10 |
| P8 | P1 | P4 | P11 |
| P7 | P6 | P5 | P12 |
| P16 | P15 | P14 | P13 |

*Fig. 3.1 16-Neighbours of 4x4 windows*

The iteration is called for pixel P1. Let N(P1) is the number of nonzero neighbours of P1; i.e.,

**N(P1)=P2+P3+P4+ .....+P15+P16**

*Pixel is deleted from image and its information is saved if*

(a) N(P1)=0

**OR**

(b) P4=P5=P6=1 &

P2+P3+P7+P8+P9+P10+P11+P12+P13+P14+P15+P16=0

*STEP 2:* It consists of two subiterations: Pass one for removing the south-east boundary points while Pass two for deleting the north-west boundary points in such a way that no corner points are deleted. 8-Neighbours of P1 are shown in Figure. 3.2

| | | |
|---|---|---|
| P9 | P2 | P3 |
| P8 | P1 | P4 |
| P7 | P6 | P5 |

*Fig. 3.2 8-Neighbours of 3x3 windows*

In Pass one, a boundary point P1 is flagged for deletion if it satisfies all the following condition.

(a) 2<N(P1)<7

(b) S(P1)=1

(c) P2*P4*P6=0

(d) P4*P6*P8=0

(e) P2+P3+P8+P9 !=0

P8+P7+P6+P5 !=0

P2+P3+P4+P9 !=0

P4+P5+P6+P7 !=0

Where N(P1) is the number of nonzero neighbours of p1; i.e.,

***N(P1)=P2+P3+P3+P4+P5+P6+P7+P8***

and S(P1) is the number of 0-1 transitions in ordered sequence P2, P3, P4,....., P8, P9, P2. For example

N(P1)=4 and S(P1)=3 in Figure. 3.3

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | P1 | 0 |
| 1 | 0 | 1 |

*Fig. 3.3*

In Pass two, a boundary point P1 is flagged for deletion if conditions (a), (b) and (e) remain the same, but the conditions (c) and (d) are changed to

(c) P2*P4*P8=0

(d) P2*P6*P8=0

**Thus one iteration of the thinning algorithm consist of**

(1) Applying Pass one to flag borders points for deletion

(2) Deleting the flagged points.

(3) Applying Pass two to flag border points for deletions.

(4) Deleting the flagged points.

The basic procedure is applied iteratively until no further points are deleted.

*STEP 3 :* A final cleanup iteration is performed as in STEP 2 but without condition (e), but with added condition that if any one of the conditions is satisfied then pixel P1 is flagged for deletion.

P7+P8+P9+P2+P3+P4=0 or

P8+P9+P2+P3+P4+P5=0 or

P6+P7+P8+P9+P2+P3=0 or

P5+P6+P7+P8+P9+P2=0 or

P3+P4+P5+P6+P7+P8=0 or

P4+P5+P6+P7+P8+P9=0 or

P2+P3+P4+P5+P6+P7=0 or

P9+P2+P3+P4+P5+P6=0

***Result of Thinning Algorithm***

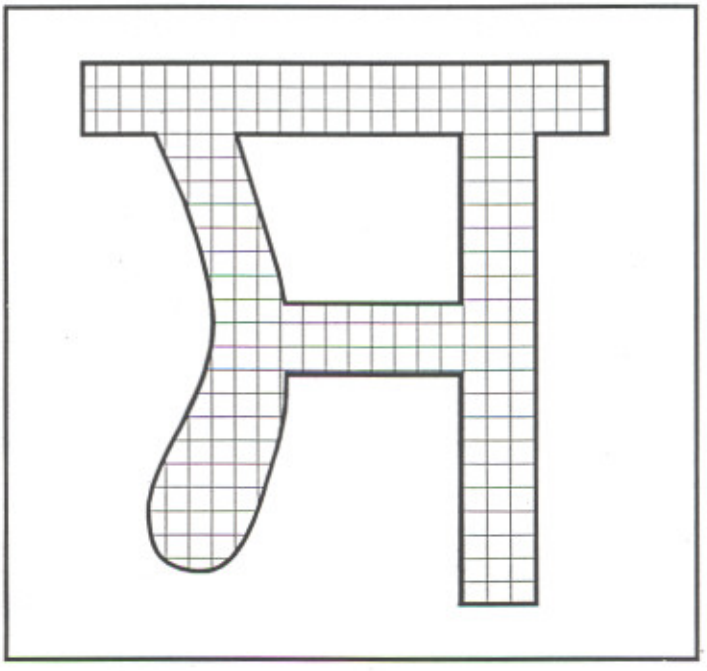
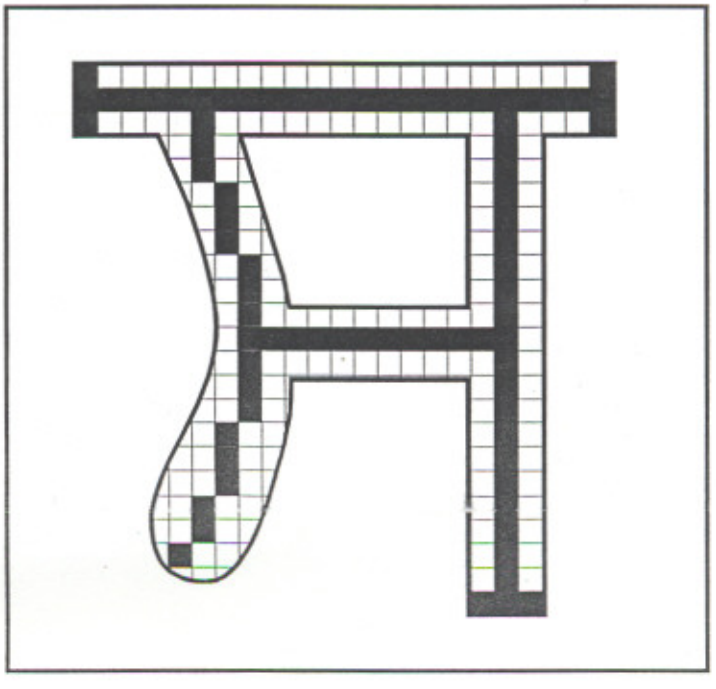


*Fig. 3.4 Input Character*



■ Indicates pixel exists

□ Indicates pixel marked for deletion

*Fig. 3.5 Result of STEP 2, of thinning algorithm during 1st iteration*

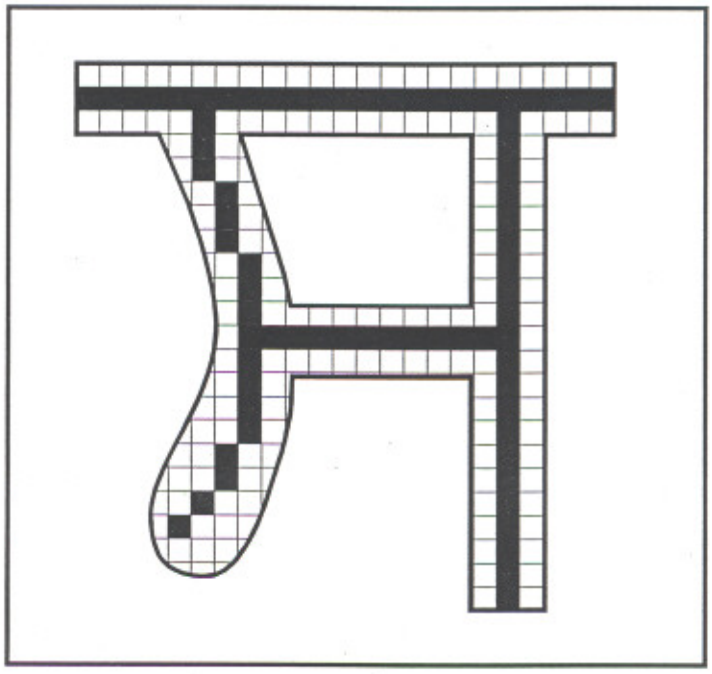


*Fig. 3.6 Result after step 3 (the final Cleanup iteration)*

### *3.2 SEGMENTATION*

Once thinning of the given text image is done, the main task of Segmentation of the given text image commences.

Segmentation problem of Gurmukhi script is entirely different from the segmentation problem of Roman script as well as from other scripts as Chinese and Japanese due to many reasons. A brief on Gurmukhi Script is given as Appendix-A to this report for ready reference.

The concept of upper/lower-case characters is absent in Gurmukhi. In general, a word of few characters of line of Gurmukhi script may be partitioned into three horizontal strips, the middle strip being the most busy one. These strips are shown in Fig. 4.4 (in Appendix A). The upper and lower strips may contain parts of vowel modifiers and diacritical markers.

In Gurmukhi script, most of the characters as shown in Appendix-A, Contain a horizontal line at the top of the middle strip. This line is called the headline. The upper strip can contain symbols like f † ˜ ¨ ¯ ¨ . The lower strip can

contain symbols _ = ,. In Roman script, in each row not more than one character is present in the same column. Thus the windows enclosing each character composing a word in machine printed Roman script do not share same pixel values in x-direction. But in Gurmukhi script. Because of these differences in the physical structure of Gurmukhi characters from the physical structure of Roman, Chinese, Japanese and Arabic scripts, the existing algorithms for character segmentation and recognition of these scripts may not work efficiently for Gurmukhi script. Hence, there is a need for designing a new technique for the Gurmukhi script.

Before commencing the segmentation implementation study let us have an idea about construction of histograms which will be used for finding the headline as well as to find the white spaces between lines and words.

### *3.2.1 Proposed Segmentation Algorithm*

Normally script segmentation is performed in three successive stages:

- line segmentation
- word segmentation and
- character segmentation.

In the proposed method, all the above mentioned three stages up to character segmentation are completed in parallel, in just one pass through the text image. The script characters are segmented by horizontal projection of printed text. Finding the headline is a crucial step in the proposed technique. The headline row of the text-line has the highest density of foreground pixels. The headline marks the end of the upper zone and the beginning of the middle zone. From here onwards, the number of foreground pixels in headline by a proportion of 10 (based on experimental work) mark the end of the middle zone and beginning of the lower zone. In this way, the three zones of a text line are found.

To start segmentation, the text image is scanned from top to bottom and in each scan line from left to right. A horizontal-bit-counter is taken to count the number of 'on' pixels in each horizontal scan-line and to find the headline as it will be having the maximum number of 'on' pixels (i.e., with value 1). An array of {equal

to the number of bytes in one scan-line of text image x 8} bits is taken to count the number of vertical bits (with value 1) in each column. When the horizontal scan line is scanned, both the horizontal-bit-counter for all the columns is processed.

### *3.2.2 Segmentation of Touching Characters*

**Construction of a vertical histogram or vertical projection & horizontal histogram or horizontal projection for line segmentation**

The lines of a text block are segmented by finding the valleys of the histogram computed by a row-wise sum of gray values. The position of the headline for each text line is already known from previous stage. The position between two consecutive head lines where the histogram height is least denotes one boundary line. A text line can be found between two consecutive boundary lines. Consider the Fig. 3.7 below, where the histogram for line segmentation is shown.

After script line segmentation, it is scanned vertically, if one vertical scan two or less black pixels are encountered than the scan is denoted by 0, else the scan is denoted by the number of black pixels. In this way a vertical scanning histogram is constructed as shown in Figure 3.7 below:

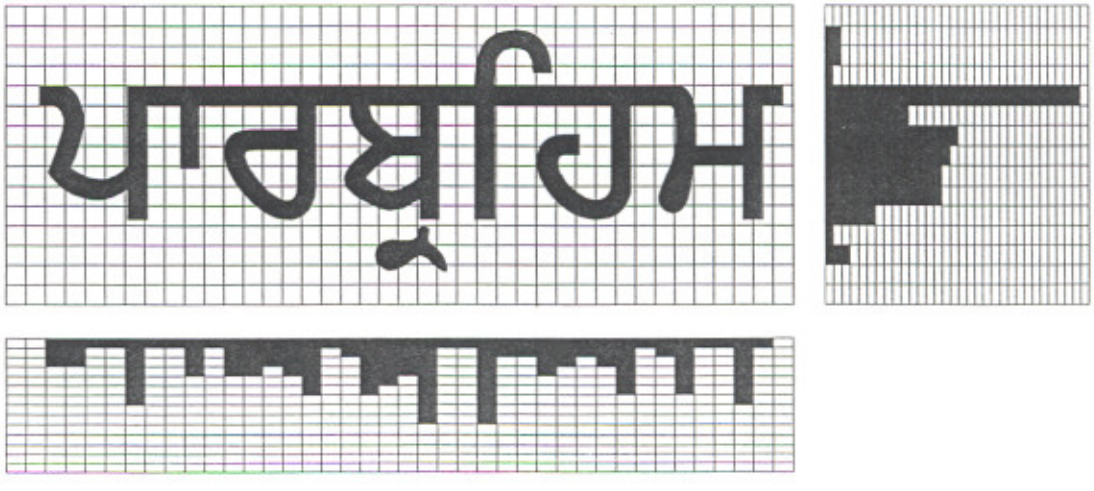


*Fig. 3.7 Histogram for line segmentation*

For segmenting the touching character, we use an emperical formula suggested by Kahan et al[13], this approach has been successfully employed, to segment the touching character in Gurmukhi script. The formula used is:

$$F(x)=V(x-1)-2*V(x)+V(x+1)/V(x)$$

The example considered in the following Fig. 3.8, shows us how this can be implemented for segmenting the touching Gurmukhi character.

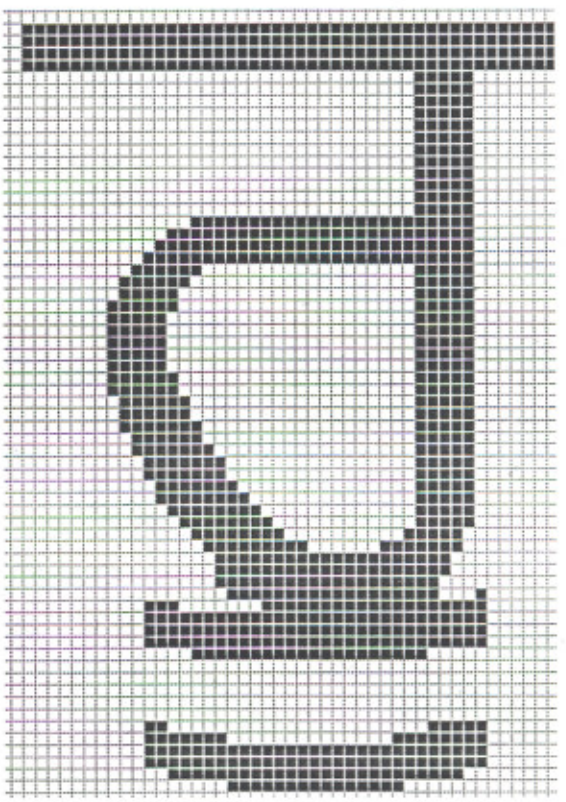


*Fig 3.8 Segmentation of touching character*

In the above Fig. 3.8, we consider the vertical count of the character and, using the formula, we generate the different values of F(x) for each row.

For example-In row 1

$$F(x) = 0-2*45+45/45$$
$$= -45/45$$
$$= -1$$

Similarly, we calculate all the F(x) values of each row.

Now, we take the maximum value of F(x), and that row is considered for cutting, also care is taken that the row length, taken for cutting is approximately 0.8 of the entire row length.

There are 63 rows and we have the maximum value in the $48^{th}$ row, and also the $50^{th}$ row is approximately 0.80 of $63^{rd}$ row. Hence, we should perform a cut in the $50^{th}$ row.

The Fig. 3.9, shows the final segmented text after this operation.

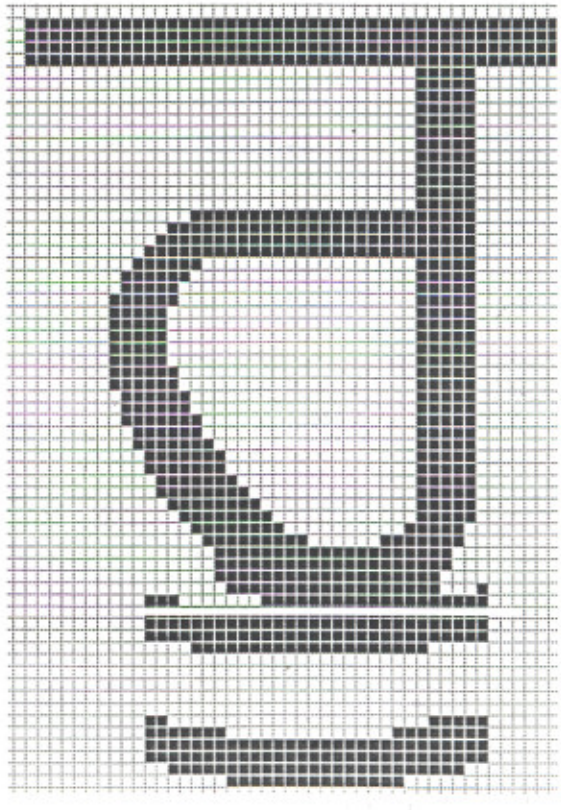


*Fig. 3.9 Cutting of touching character*

# Chapter 4

# Implementation

## 4.1 Input

The text document was scanned using the flatbed scanner UMAX-Powerlook-III. The PCX file format was used to store the scanned binary text image.

## 4.2 Algorithm

*Step 1: Scanning of the machine printed text & storing it as a PCX file*

The machine printed document is scanned & stored as a PCX file, while scanning care is taken that the paper is not mutilated, the printing is even on the paper & the paper is placed exactly parallel with one of the rulers bordering the scanner's object glass. Further, care must also be taken not to cover the orientation holes of the scanner.

*Step 2: Perform thinning operation*

The thinning operation is performed as already discussed in section 3.1

*Step 3: Segmentation*

Segmentation is performed as discussed in section 3.2.

*Step 4: Output*

Final segmented image, as processed by the program is shown in the following section (section 4.4).

## 4.3 Coding Environment

The whole system was developed using Tubo C++. Object oriented approach was used. Many other things were kept in mind during coding for efficient programming such as:

- Variables', constants' and function' names were chose so as to reflect their purpose.
- No GOTO statement was used anywhere to avoid spaghetti coding.
- Proper documentation was used wherever needed to make the code more readable and easy to modify.

- Global variables were used with caution, so that they may not be changed accidentally.
- Top down approach was used.

## 4.4 RESULT

*Step 1*

The input file containing the Gurmukhi script is first scanned by the scanner in the PCX format. For eg. consider the following text.


*Fig. 4.3 Original Scanned Image*

*Step 2*

The input document is scanned in PCX format. This file is read by the program, and the information regarding it is as shown below:

```
Processing File : d:\aku\aku.pcx
Manufacturer              :   a
Version                   :   5
Encoding                  :   1
Bits per pixel per plane  :    1
Picture Dimension         :  1  0  1  0 fe  3 a6  6
Horizontal resolution     :  60   0

Vertical resolution       :  60   0
Color Map :
  0  0  0 ff ff ff ff ff  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
Reserved                  :   0

No. of Color Planes       :   1
No. of bytes/Scan line    :  80   0
No. of bytes/Scan line    :   128
Horizontal screen size    :   0   0
Vertical screen size      :   0   0
```

*Fig.4.4 PCX file Information*

*Step 3*

The program gives the information about the PCX file, as shown below.

Thining algorithm is applied on the file and the thinned image is obtained as shown below:

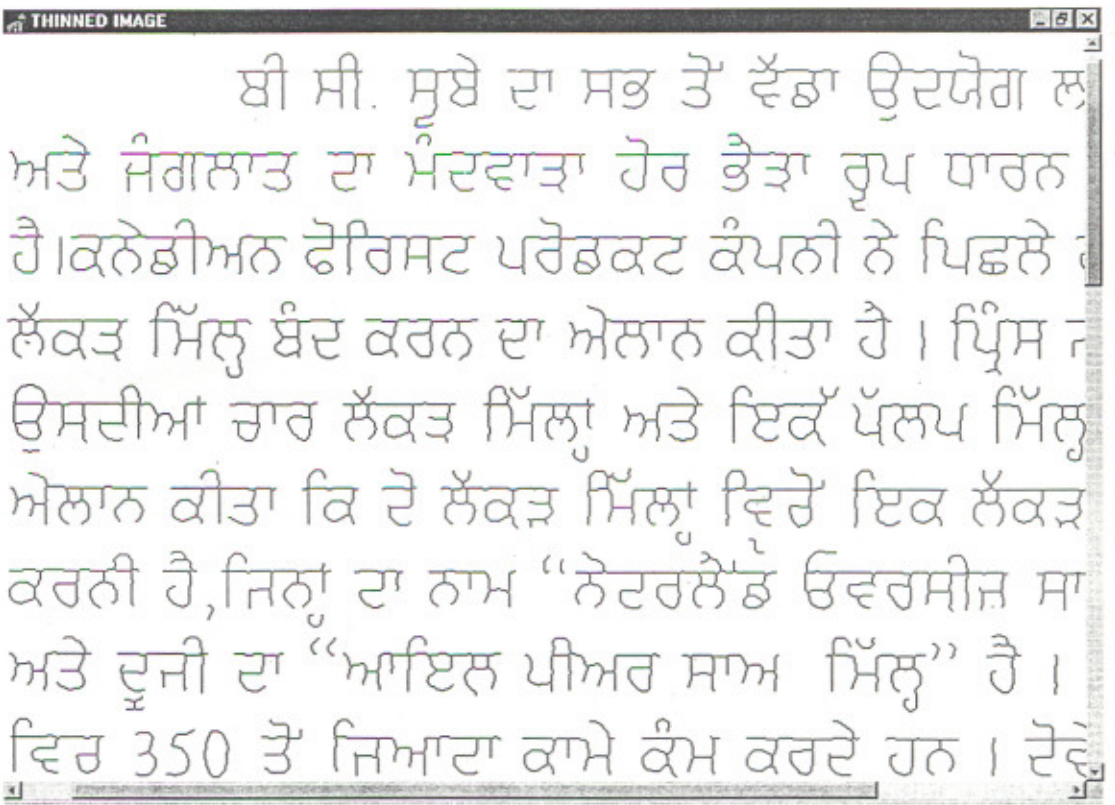



*Fig.4.5 Thinned Image of input text*

*Step 4*

The thinned image is then segmented in upper-zone, middle-zone and lower-zone, considering the touching characters as already discussed.

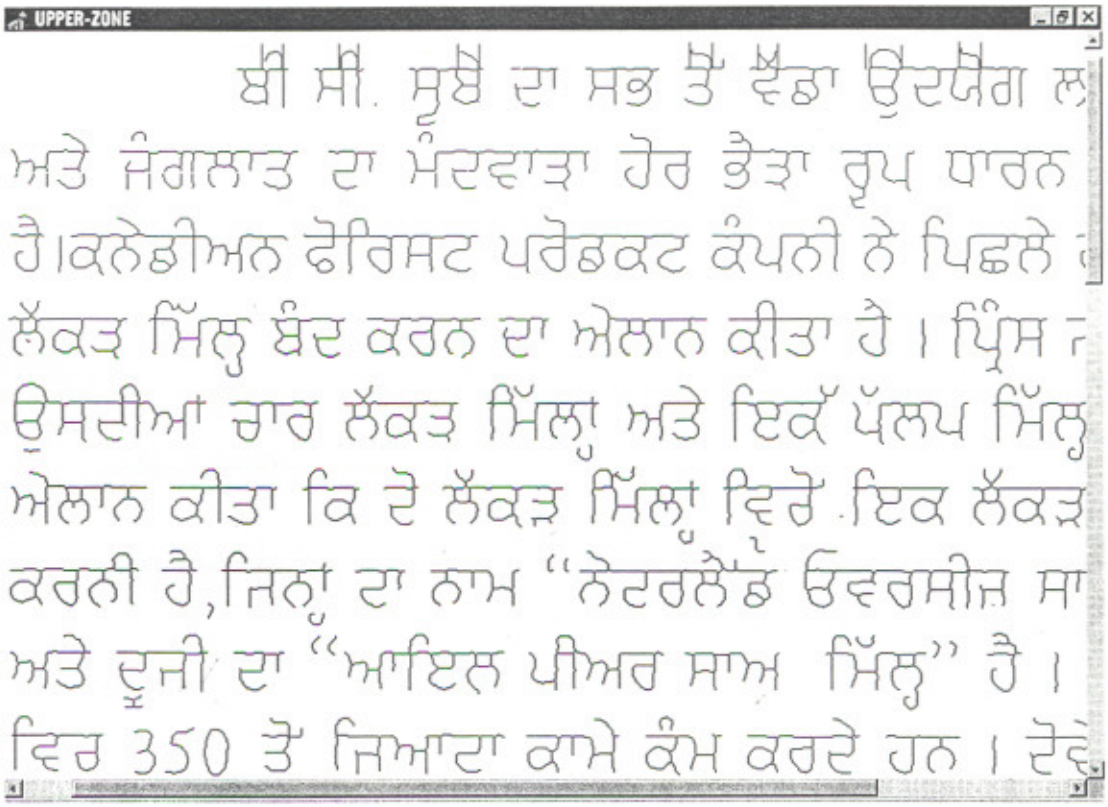



*Fig. 4.6a Segmentation of upper zone characters of first line.*

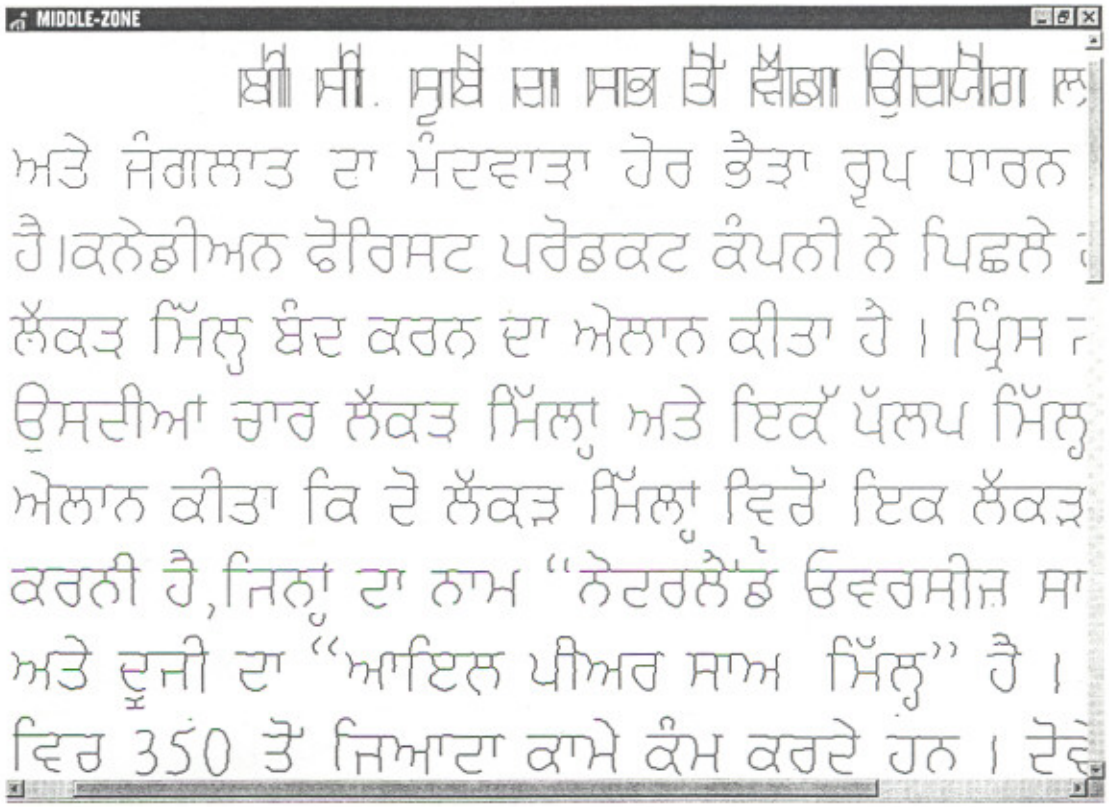



*Fig. 4.6b Segmentation of middle-zone characters of first line.*

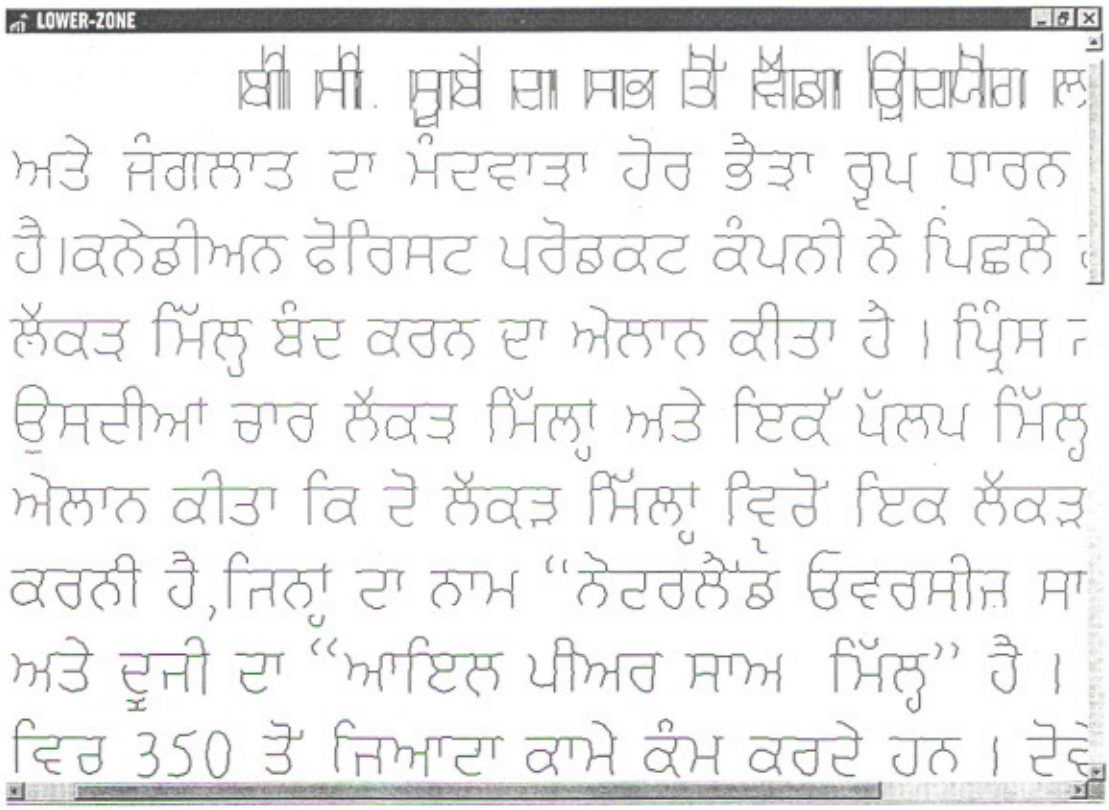



*Fig. 4.6c Segmentation of lower-zone characters of first line.*

*Step 5*

The final result is obtained after segmenting the entire input text.

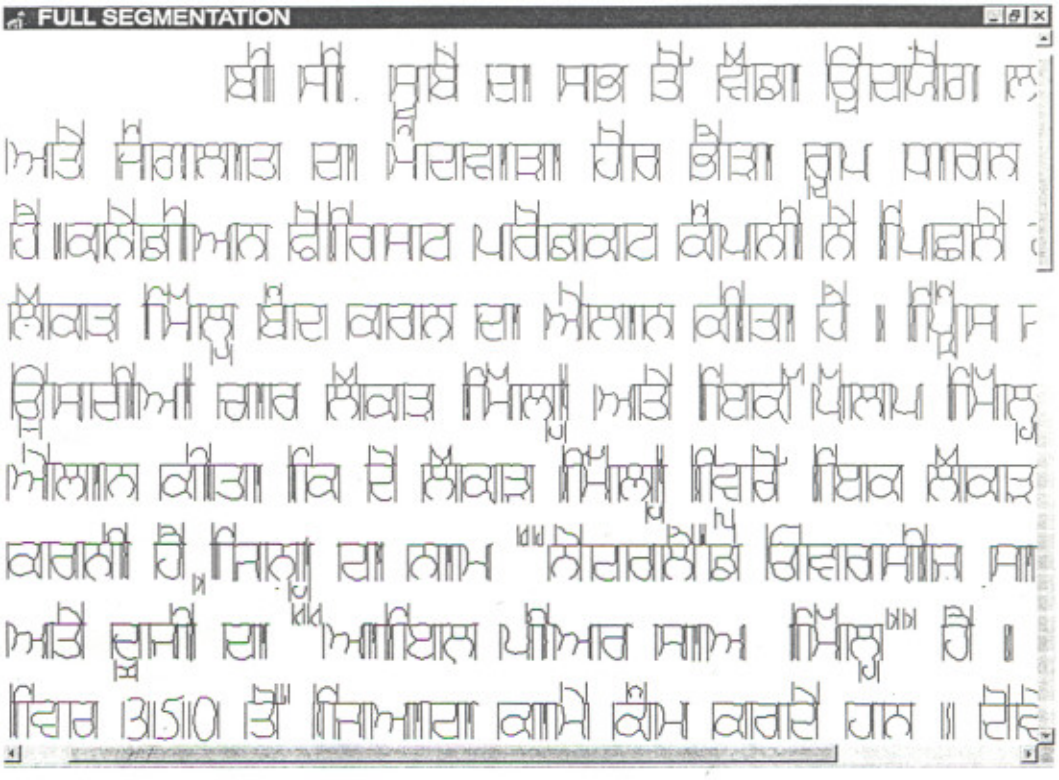



*Fig. 4.7 Full Segmentation of the input text*

# *Chapter 5*

## Conclusion

For an efficient OCR system, the segmentation place a very important role. Particularly, in case of the script like Gurmukhi, which constitutes mostly compound characters where the consonants are generally ornamented with vowels in both upper and lower parts of the consonants hence, the role of segmentation deserves a special study and application.

The algorithm was written in C++ under WINDOWS-98 operating system on a Pentium–III 800 Mhz system. Different samples were taken and the average processing time for each character was approximately 2 msec.

The experiments carried out showed that:

- The proposed method was tested over the following Gurmukhi script of font 12,14,16,18 and 24 point with GurmukhiLys020 font text image files.
- The flatbed scanner used was UMAX-Powerlook-III.

*Future Scope:-*

- The work can be extended for on-line segmentation.
- This work can be enhanced to segment degraded text.
- The work can be extended for segmenting touching characters in the mid-zone, merged upper zone and lower zone characters touching the above character lying in the middle zone.

# Characteristics of Gurmukhi Script

The word 'Gurmukhi' literally means from the mouth of the Guru. Gurmukhi script is used primarily for the Punjabi language, which is the world's 14th most widely spoken language. The populace speaking Punjabi is not confined to the North Indian states such as Punjab, Haryana and Delhi but is spread over all parts of the world.

A brief introduction of the script is given next from the OCR point of view.

## Constituent Characters and Symbols of Gurmukhi Script

Gurmukhi script alphabet consists of 40 consonants and 12 vowels Figure(A.1). Besides these some of the characters in form of half characters are present in the feet of characters. Writing style is from left to right. The concept of upper/lower-case characters is absent in Gurmukhi. A line of Gurmukhi script may be partitioned into three horizontal zones, the middle zone being the most busy one. These zones are shown in figure 2. The upper and lower zones may contain parts of vowel modifiers and diacritical markers.

In Gurmukhi Script, most of the characters, as shown in figure(A.1), contain a horizontal line at the top of the middle zone. This line is called the headline. The characters in a word are connected through the headline along with some symbols as f, ੍ਹ, ੍ਰ, etc. The headline helps in recognition of script line positions and character segmentation. The upper zone can contain symbols like ਁ, ਂ, ਃ, ੱ, ੰ, ੲ, and ੴ. The lower zone can contain symbols like ੁ, ੂ, ੍ and ੵ. The segmentation problem for Gurmukhi script is entirely different from scripts of other common languages such as English, Chinese, urdu etc. In Roman scrip, windows enclosing each character composing a word do not share the same pixel values in horizontal direction. But in Gurmukhi script, as shown in figure 2, two or more characters/symbols of same word may share the same pixels values in horizontal direction. This adds to the complication of segmentation problem in Gurmukhi script. Because of these differences in the physical structure of Gurmukhi characters from those of Roman, Chinese, Japanese and Arabic scripts, the

existing algorithms for character segmentation of these scripts may not work efficiently for Gurmukhi script.

| *Consonants* | | | | |
|---|---|---|---|---|
| ੳ | ਅ | ੲ | ਸ | ਹ |
| ਕ | ਖ | ਗ | ਘ | ਙ |
| ਚ | ਛ | ਜ | ਝ | ਞ |
| ਟ | ਠ | ਡ | ਢ | ਣ |
| ਤ | ਥ | ਦ | ਧ | ਨ |
| ਪ | ਫ | ਬ | ਭ | ਮ |
| ਯ | ਰ | ਲ | ਵ | ੜ |
| ਸ਼ | ਜ਼ | ਖ਼ | ਫ਼ | ਗ਼ |
| *Vowels in Upper zone* | | | | |
| ◌ੰ | ◌ੱ | ◌ੁ | ◌ੂ | ◌ੇ |
| ◌ੈ | ◌ੋ | | | |
| *Vowels in Upper and Middle zone* | | | | |
| ◌ੌ | ◌ੀ | | | |
| *Vowels in Middle zone* | | | | |
| ◌ਾ | | | | |
| *Vowels in Lower zone* | | | | |
| ◌ਿ | ◌ੁ | | | |
| *Half characters in Lower zone* | | | | |
| ◌੍ | ◌ਃ | | | |

*Fig. (A.1): Gurmukhi script characters and symbols*

**Compostion of Characters and Symbols for writing Words in Gurmukhi Script:**

A horizontal line in drawn on top of all characters of a word which is referred as head line. A character is usually written such that it is vertically separate from its neighbours. Whereas, a consonant in derived pure form is written such that it touches the following character. This results in a composite character. One or more modifier symbols and diacritic marks can be attached to a character.

It is convenient to visualize a Gurmukhi word in terms of three zones:

a core strip (mid-zone), a top strip (upper zone) and a bottom strip (lower zone). Top and bottom strips have only modifiers and diacritic marks whereas the core strip has the characters and vowel modifier 'ਾ'.

Figure: (A.2) shows a sentence and figure (a, b) and (c) show the contents of the three strips. The top and bottom strip may be empty for a word, but only the vowels/half characters may be present in these strips.

Figure(A.2): An example showing the three strips of Gurmukhi words:
An example sentence:


*Fig. (A.2) Three zones of Gurmukhi script characters*
*(a) upper zone (b) middle (c) lower zone*

## Input text image file format [pcx]

Standard PCX file format is used for storing the scanned image of the text. The image is stored as mono-colour. A video image consists of many horizontal scan lines. The PCX format encodes all the information for one line before processing the next one.

A PCX file consists of two parts:

- a 128 byte header and
- the encoded graphic data

The encoding method is a run-length limited (RLL) technique that compresses most images. The PCX file is usually much smaller than its unencoded equivalent.

The PCX compression scheme uses the fact that adjacent pixels in images are often the same, i.e., features typically extend for some distance. For example, in a photograph of a person's face, the background is usually uniform. The format for each encoded byte is as follows:-

If its two high bits are both 1, it specifies multiple pixels. The other six bits indicate how many pixels (up to 63). The next byte is the actual data. If the two high bits are not both 1, the byte specifies just one pixel and contains its value.

*Note*:- This method produces no compression when adjacent pixels differ or only two in a row are the same. If three or more are the same, there is a saving. Figure (B.1) shows three typical cases.

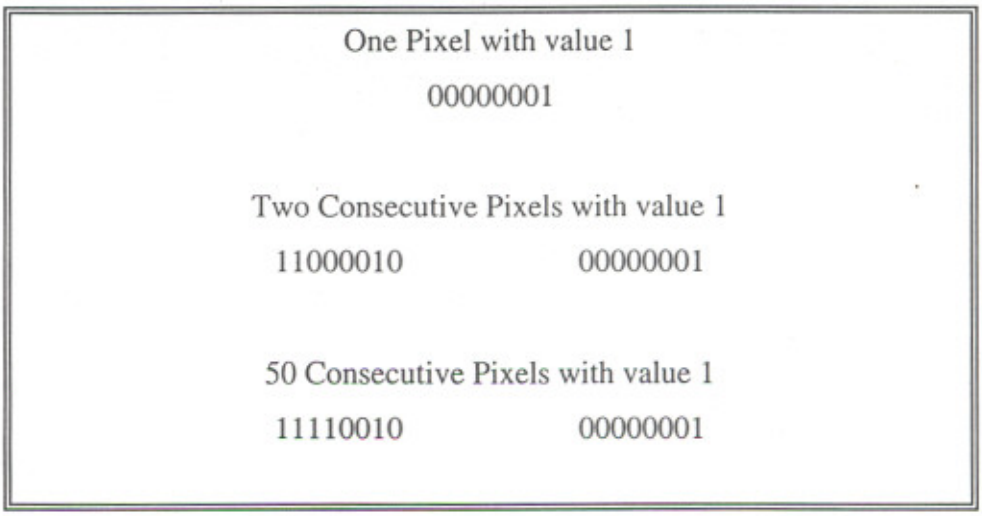



*Fig. B.1 Examples of PCX binary encoding*

The first example requires one byte to encode one pixel, the second requires two bytes for two pixels, and the third takes just two bytes for 50 pixels. There is no compression in the first two cases, but considerable savings in the third.

First 128 bytes of the file contain the PCX file header. Entries in the PCX file header are as:

**PCX FILE HEADER**

| Byte Number | Byte Description |
|---|---|
| 0 | Manufacturer |
| 1 | Version (PC Paintbrush version) |
| 2 | Encoding |
| 3 | Bits per pixel per plane |
| 4-11 | Window (Picture dimensions) |
| 12-13 | Horizontal resolution in dots per scan line |
| 14-15 | Vertical resolution in scan lines per screen |
| 16-63 | Colour map, varies for different types of video Reserved |
| 65 | Number of colour planes |
| 66-67 | Number of bytes per scan line |
| 68-69 | Palette interpretation |

To decode a PCX file, we must first determine how many bytes each scan line requires. The necessary information is in header bytes 65, 66 and 67. Number of bytes required for each san line is obtained by multiplying the number of colour planes by the number of bytes per scan line. By keeping, running total of the number of pixels processed we can identify the end of the line.

# References


[1] H S Baird, S Kahan ,T Pavlidis , **Components of an omni-font page reader,** *Proc. Eighth Int'l. Conf. on Pattern Recognition,* pp. 344-348, 1986.

[2] RG Grimsdale, FH Summer, CJ Tunis and T Kirburn, **A system for the automatic recognition of patterns**, *Proc, Instt. Elec. Eng., vol. 106B,* pp.210-222, 1958.

[3] J Rabinow, **Developments in character recognition machines at Rabinow Engineering. Company,** *Optical character recognition Eds. Fischer et al, McGreder & Werner,* pp.27-51, 1962.

[4] ABBYY's Fine read OCR system evaluation package

[5] S Tsujimoto, H Asada, **Major Components of a Complete Text Reading System,** *Proc. IEEE, Vol 80, No.7,* pp. 1133-1149, 1992.

[6] Louisa Lam, Seong-Whan Lee and Chin Y. Suen, **Thinning methodologies-- A Comprehensive Survey,** *IEEE Trans. Pattern Analy. and Machi. Intelli. , vol.14, No. 9,* pp. 869-885, 1992.

[7] C. Arcelli, **Pattern thinning by contour tracing,** *Comp. Graphics Image processing, vol 17,* pp. 130-141, 1981.

[8] T. Pavlidis, **A thinning algorithm for discrete binary images,** *Comp. Graphics Image processing, vol 13,* pp. 142-157, 1980.

[9] VK Govindan and AP Shivaprasad, **A pattern adaptive thinning algorithm,***Pattern recog., vol.20, No.6,* pp. 623-637, 1987.

[10] PCK Kwok, **A thinning algorithm by contour generation,** *Comm. ACM, vol.31, no.11,* pp.1314-1324, 1988.

[11] A Bel-Lan and L Montoto, **A thinning transform for Digital images,** *Signal Processing, vol.3, no.1,* pp. 37-47, 1981.

[12] Y S Chen, W. H. Hsu, **A modified fast parallel algorithm for thinning digital patterns,** *Pattern Recog. Lett., vol 7, No. 2,* pp. 99-106, 1988.

[13] S Suzuki and K Abe, **Binary picture thinning by an iterative parallel two subcycle operation,** *Pattern recog., vol.10,no.3,* pp. 297-307, 1987.

[14] S Kahan, T Pavlidis, H S Baird, **On the recognition of printed characters of any font and size,** *IEEE trans. on pattern and machi. intelli., vol. 9, 1987*

[15] Theo Pavlidis, **Recognition of printed text under realistic conditions,** *Elsevier science publication, vol.14, no.3,* pp. 99-106, 1993.

[16] CC Tappert, CY Suen, T Wakahara, **The state of the art in on-line handwriting recognition,** *IEEE trans. Pattern analysis and machi. intelli., vol.12, no. 8,* pp. 787,1990.

[17] Richard G Casey, Eric lecolinet, **A survey of methods and strategies in character segmentation,** *IEEE trans. Pattern analysis and machi. intelli.,vol.18, no.7,* pp. 690-706, 1996.

[18] CE Dunn and PSP Wang, **Character segmenting techniques for handwritten text - a survey,** *Proc. 11th Int'l. Conf. Pattern Recog., Vol.2,* pp. 577-582, 1992.

[19] Y Lu, **Machine printed segmentation - an overview,** *Pattern recoginition, vol. 29, no. 1,* pp. 67-80, 1995.

[20] J. Wang and J. Jean, **Segmentation of Merged Characters by Neural Networks and Shortest Path,** *Pattern Recognition vol. 27, no.5,* pp. 649-658, 1994.

[21] G S Lehal and Parminder Singh, "**A Technique for Segmentation of machine Printed Gurmukhi Script,** *Proc. 4th International Conf. on cognitive systems, Delhi, India* pp.283-287, 1998.

[22] Sunil Madan, **Skew Correction and Character Segmentation for Printed text in Gurmukhi Script,** *an M. Tech. Thesis submitted to DCSE at Punjabi University,* Dec'97.

[23] A K Goyal, G S Lehal, S S Deol, **Segmentation of Machine Printed Gurmukhi Script,** *Proceedings of 9th International Graphonomics Society Conference, Singapore,* pp. 293-297, 1999.

[24] A K Goyal, G S Lehal, J Bahl, **Gurmukhi text recognation using neural network.** *Proceedings of IVth International Conf. on Cognitive Systems, Dec. 1999.* pp. 141-150

[25] G S Lehal and Chandan Singh, **Feature Extraction and Classification for OCR of Gurmukhi Script,** *Vivek Vol. 12,* pp. 2-11, 1999.

[26] G S Lehal and Chandan Singh, **Text Segmentation of Machine Printed Gurmukhi Script,** *in Document recognition & retrieval VIII, Edited by: Paul B Kantor, Daniel P Lopresti, Jiangying Zhon, Proc. SPIE, USA, vol. 4307,* pp.223-231, 2001.

[27] Pal and Choudhary, **Computer Recognition of Bangla Script,** *Int'l. J. System Sci.26,* pp. 2107-2123, 1995.

[28] Pal and chaudhary, **Printed Devnagari script OCR System,** *Proc. of Intl. Conf. on Knowledge Based Computer Systems,* NCST, Bombay 1996. pp. 359-368.

[29] Veena Bansal, **Integrating Knowledge Sources in Devnagari Text Recognition,** *Doctoral thesis, IIT Kanpur,* 1999.

[30] Shiaw-Shian Yu and Wen-Hsiang, **A new thinning algorithm for gray scale images by the relaxation technique,** *Pergamon Journals Ltd.,vol.20, no.1,* pp. 7-15, 1987.

[31] Raymond W Smith, **Computer processing of line images: a survey,** *Elsevir Sc. Publn., vol 14, No.4,* pp.317-326, 1993

**Some of the internet sites referred:**

| | |
|---|---|
| www.dataid.com / about ocr: | Site giving an introduction about OCR. |
| www.onix.com | A useful site, giving a comparison of the leading OCR products. |
| www.adams1.com | Site dealing with the different shareware of OCR. |
| www.hera.itc.it:3003 | A useful site provides links to commercially available products for machine & hand printed text recognition, alongwith links to some of the evaluation tests. |
| www.ed.gov/offices/ocr/testing | Regarding the testing of an OCR |
| www.programfiles.com | Site giving information about OCR fonts & products |
| www.pcworld.com/resource | PC World magazine site |
| www.byte.com/art 9410 | Byte magazine site |