

# RECOGNITION OF DEVANAGARI SCRIPT

*Dissertation submitted to  
Jawaharlal Nehru University  
in partial fulfilment of the requirements  
for the award of degree of*

**MASTER OF TECHNOLOGY**

in

**COMPUTER SCIENCE & TECHNOLOGY**


by

**SHOMA CHATTERJEE**

SCHOOL OF COMPUTER AND SYSTEMS SCIENCES  
JAWAHARLAL NEHRU UNIVERSITY  
NEW DELHI - 110 067

# CERTIFICATE

This is to certify that the dissertation entitled "Recognition of Devanagari Script", submitted by Miss Shoma Chatterjee is a record of bonafide work done under my guidance and supervision in partial fulfilment of the requirement for the award of M.Tech degree in Computer Science. This work has not been submitted elsewhere for any other degree.



21-12-97

**Prof. K.K. Bharadwaj**

**Supervisor**

**School of Computer &  
Systems Sciences**

**J.N.U.**



21-12-97

**Prof. K.K. Bharadwaj**

**Dean**

**School of Computer &  
Systems Sciences**

**J.N.U.**

# ACKNOWLEDGEMENT

I would like to take this opportunity to thank my supervisor **Prof.K.K. Bharadwaj** for his guidance and support.

I would also like to thank **Prof.K.K.Biswas**, Head of Computer Sc. & Engg. Deptt, I.I.T., Delhi for his invaluable help and guidance during the course of this project.

*Sharma*  
**Shoma Chatterjee (Miss)**  
**M.Tech III semester**  
**School of Computer &**  
**Systems Sciences**  
**J.N.U.**

# CONTENTS

## CHAPTER

- 1 INTRODUCTION**
  - 1.1 On-line versus off-line recognition
  - 1.2 Properties of scripts
    - 1.2.1 English language
    - 1.2.2 Chinese language
    - 1.2.3 Japanese language
    - 1.2.4 Devanagari script
- 2 AN INTRODUCTION TO THE SYSTEM**
  - 2.1 Data collection & data acquisition
  - 2.2 Deskewing & segmentation
  - 2.3 Analysis of characters without preprocessing
  - 2.4 Preprocessing
  - 2.5 Analysis of characters after preprocessing
  - 2.6 Creating & referencing database
  - 2.7 Analysis of half forms
  - 2.8 Use of heuristic
  - 2.9 Training algorithm
  - 2.10 Results
- 3 DATA COLLECTION & DATA ACQUISITION**
  - 3.1 Introduction
  - 3.2 Data collection
  - 3.3 Data acquisition
    - 3.3.1 Scanning
  - 3.4 Binarization
  - 3.5 Suggestions

- 4**           **SEGMENTATION**
  - 4.1 Introduction
  - 4.2 Implementation details
    - 4.2.1 Line segmentation
    - 4.2.2 Word segmentation
    - 4.2.3 Character segmentation
      - 4.2.3.1 Division of characters into zones
  
- 5**           **FEATURE EXTRACTION WITHOUT PREPROCESSING**
  - 5.1 Introduction
  - 5.2 Review of earlier work
  - 5.3 Implementation details
    - 5.3.1 Features
  
- 6**           **PREPROCESSING**
  - 6.1 Introduction
  - 6.2 Noise removal
  - 6.3 Thinning
    - 6.3.1 Rosenfeld & Kak thinning algorithm
    - 6.3.2 Thinning algorithm based on morphological operators
    - 6.3.3 Implementation details
  - 6.4 Smoothing
  
- 7**           **CHARACTER RECOGNITION**
  - 7.1 Introduction
  - 7.2 Knowledge-based pattern recognition using syntactic approach
  - 7.3 Feature extraction after preprocessing
    - 7.3.1 Introduction
    - 7.3.2 Implementation details
      - 7.3.2.1 Features

8	<b>CLASSIFICATION OF CHARACTERS</b>
9	<b>DICTIONARY</b>
	9.1 Introduction
	9.2 A dictionary for storing the features of characters.
	9.2.1 Implementation details
10	<b>USE OF HEURISTIC</b>
11	<b>RESULTS</b>
	11.1 The working system
	11.2 Specifications
12	<b>TRAINING ALGORITHM</b>
	12.1 The essence of artificial neural networks(ANNs)
	12.2 Operating an ANN
	12.3 Properties of ANNs
	12.4 Fundamentals of neural computing
	12.4.1 Components of a node
	12.4.2 Topology of an ANN
	12.5 Learning and training with ANNs
	12.5.1 Learning procedure
	12.5.2 Backpropagation learning: The Vanilla backpropagation algorithm
13	<b>CONCLUSION</b>
	<b>REFERENCES</b>

**CHAPTER 1**  
**INTRODUCTION**

# CHAPTER 1

## INTRODUCTION

### 1.1 INTRODUCTION

In recent years, great progress has been made in optical character reader (OCR) technology. Most OCRs in current use, can only read characters printed on a sheet of paper according to rigid formatting restrictions and are mainly being applied to office automation systems such as document readers. However, if OCRs could read text directly from books and magazines, then they could be put to more general applications like they could be used to recognize text, and through a voice decoder system the contents could be read out for blind men.

Unlike text on a well-printed sheet of paper, the text in books and magazines suffer from a variety of noise components. The text in books and magazines are generally skewed as a result of manual block-printing.

Conventional research work on character recognition involves two distinct approaches. In one of the approaches the digital images of the character are binarized, thinned and vectorized before the symbol recognition is done (Sinha et al.[32], Baptista et al.[5], Sareen[39]). The second approach avoids the thinning and vectorization step. This approach is based on analysis of strokes, Fourier expansion of the symbol boundary in the binary raster representation (Taxt et al.[45]), feature extraction from profiles of external



contours (Kimura et al.[4]), extracting features directly from gray-scale images by extracting and assembling topographic characteristics of the surface (Wang et al.[22]).

The former approach using the preprocessing stage is computationally slow and a significant amount of information is lost during the preprocessing stage. In case of the latter approach, the information loss caused by the thinning and vectorization steps is eliminated, it is computationally faster than the former and also the classification rate is much higher as compared to the earlier approach involving the preprocessing stage. But, a major drawback faced by the latter approach is that it fails to isolate joined characters : Research works in the latter approach have assumed the input text to be isolated characters (Taxt et al.[45], Kimura et al.[4]).

In the present paper, both the approaches are utilized. Initially, it is assumed that the characters are isolated. If the width to height ratio of a character is below a certain defined limit then the character is processed using the latter approach i.e. minus the preprocessing stage. In this approach analysis of strokes like stroke direction and stroke length is studied for analysis and recognition. In case the confidence level associated with the recognition is below the defined threshold of 75% then the approach using the preprocessing stage is used to confirm the results. In this approach feature extraction is based on the extraction of distinctive features like number of terminating points, bend points, junction points, segments, etc. Two new features have been introduced in this paper : the direction of maximum curvature and the direction of transition.

Also, if the width to height ratio of a character exceeds the defined limit then it is assumed that it is a joined character. Such joined characters can be of many distinct forms like one character joined to another character due to noise or printing defect ( क्ष ), or a half form joined to its counterpart ( ख ), or a character lying within the shadow of another character ( ख ). A new strategy has been adopted to isolate such joined characters from their counterparts. The segments of a character are followed and coded up according to the segment being a vertical, horizontal or a slanting line. These codes form the transitions in a finite automaton. Whenever a code is encountered that does not belong to a character then a trap state results. If a code leads to a final state, then the character associated with the particular final state is accepted.

Broadly speaking, this project aims at developing a system that takes Devanagari script text as input. It breaks the given input text into lines, from the lines it extracts words and finally from the words it extracts characters. For each of the individual characters, it extracts significant features and recognizes the characters and other additional forms in the text and outputs the obtained results. A comparison of the input and output text gives us the efficiency and the error rate.

The significance of this project lies in the fact that OCRs for different languages like English are already available. Such systems have not yet been developed for Hindi because of its highly complex pattern. The dissertation for my M.Tech is an effort in this direction.

## 1.2 ON-LINE VERSUS OFF-LINE RECOGNITION

Following Tappert et al.[2]

On-line handwriting recognition means that the machine recognizes the writing while the user writes. The term real time or dynamic is also used in place of on-line. Depending on the recognition technique and the speed of the computer, the recognition lags behind the writing to a greater or a lesser extent. Most commercial recognizers lag by only one or two characters. On-line recognition systems need only be fast enough to keep up with the writing. Average writing rates are 1.5-2.5 characters/s for English alphanumerics or 0.2-2.5 characters/s for Chinese characters. Peak rates for English can approach 5-10 characters/s. On-line handwriting recognition requires a transducer that captures the writing as it is written. The most common of these devices is the electronic tablet or digitizer, which typically has a resolution of 200 points/in, a sampling rate of 100 points/s, and an indication of "inking" or pen down.

Off-line handwriting recognition is performed after the writing is completed. An optical scanner converts the image of the writing into a bit pattern. Scanners have x and y resolutions of typically 300-400 points/in. Off-line handwriting recognition is a subset of Optical Character Recognition (OCR). OCR systems typically process hundreds of characters a second.

Another distinction is between on-line and off-line capture of handwriting data. On-line capture means that the machine data are being captured as a person writes. Off-line data capture means that the machine data are captured some

time after the writing is created. Once captured, on-line or off-line handwriting data can be processed by the recognizer afterwards.

An advantage of on-line devices is that they capture the temporal or dynamic information of the writing. This information consists of the number of strokes, the order of the strokes, the direction of the writing for each stroke, and the speed of the writing within each stroke. A stroke is the writing from pen down to pen up. Most on-line transducers capture the trace of the handwriting or line drawing as a sequence of coordinate points. By contrast, off-line conversion of scanned data to line drawings usually requires costly and imperfect preprocessing to extract contours and to thin or skeletonize them. The temporal information provided by on-line entry improves recognition accuracy. On the other hand, the temporal information of on-line systems may complicate recognition with variations that are not apparent in the static images.

Another advantage of on-line handwriting recognition is interactivity. In an editing application, for example, the writing of an editing symbol can cause the display to change appropriately. Also, recognition errors can be corrected immediately.

Yet another advantage is adaptation. When the user sees that some of his characters are not being accurately recognized, he can alter their drawing to improve recognition. Thus, the user adapts to the recognition system. On the other hand, some recognizers are capable of adapting to the writer, usually by storing samples of the writer's characters for

subsequent recognition.

The main disadvantage of on-line handwriting recognition is that the writer is required to use special equipment.

### **1.3 PROPERTIES OF SCRIPTS**

Consider the following languages - English, Chinese, Japanese and Devanagari script.

Following Tappert et al.[2]

#### **1.3.1 English Language**

The English alphabet has 26 letters, and each letter has two forms, upper and lower case. English words consist of sequence of letters, five per word on the average. In English, the position and size of the letters is important. Upper case letters sit on the baseline and are full sized. Lower case letters are smaller, and most are about half the height of upper case letters. Some lower case letters have an ascender, which extends upward to almost the height of the upper case letters, some have a descender, which extends down below the baseline, and some have both.

#### **1.3.2 Chinese Language**

The Chinese has a much larger set of characters. A Chinese character can represent a word. There are about 50,000 characters, and a basic vocabulary consists of 3-5000 characters. There are two basic styles of writing characters, block and cursive. The block style is written carefully, with fairly strict adherence to proper stroke number and order.

#### **1.3.3 Japanese Language**

The Japanese use Hiragana, Katakana, Kanji, and English alphanumerics. Hiragana and Katakana (called Kana) are

phonetic alphabets, and each has 46 full-size characters. A small size of eight of the Kana characters together with additional markings indicate subtle phonetic differences. Kanji are Chinese characters, and a set of 6349 is the Japanese Industry standard, although daily usage is limited to 2000. Kanji and Chinese characters have essentially the same meaning.

#### 1.3.4 Devanagari Script

Following [52],

The Hindi language, in common with Marathi, Nepali and many north Indian dialects, is written in the Nagari (or the Devanagari) script which is also the script for Sanskrit.

#### The Alphabet

The alphabet consists of 11 vowels and 35 consonants, as follows:-

(a) Vowels:

अ a, आ ā, इ i, ई ī, उ u, ऊ ū, ऋ ṛ, ए e, ऐ ai, औ au, ओ o.

The vowel ऋ occurs only in Sanskrit words borrowed into Hindi.

(b) Consonants:-

क ka,	ख kha,	ग ga,	घ gha,	ङ ṅa,
च ca,	छ cha,	ज ja,	झ jha,	ञ ña,
ट ta,	ठ tha,	ड da,	ढ dha,	ण na,
त ta,	थ tha,	द da,	ध dha,	न na,
प pa,	फ pha,	ब ba,	भ bha,	म ma,
य ya,	र ra,	ल la,	व va,	
श śha,	ष ṣa,	स sa,	ह ha,	
ड़ ṛa,	ढ़ ṛha.			

An अ 'a' is inherent in each consonant letter.

इ, ज, ञ, इ, and ङ never occur in the beginning of a word; and ङ and ज never occur by themselves, they are always combined with a following consonant.

The sign ॠ (candra-bindu) placed above a vowel (अ, etc.) indicates that the vowel is nasalized (Anunasika), or 'spoken also through the nose'.

The sign '•' (Anuswara) placed above a vowel may represent any one of the consonants इ, ज, ञ, न and म (to be pronounced after the vowel).

The sign ':' (Visarga) placed after a vowel represents a ह् .

Some Arabic, Persian and English consonants, found in Hindi loan-words from these languages, are indicated by the following dotted letters. - क, ख, ग, ज, फ .

#### Mode of Writing : Vowels

The Hindi consonant letters do not indicate the consonant sound only. They stand for the particular consonant + अ . Thus क is not simply क , but क + अ ; ल is not simply ल , but ल + अ . This अ is called "the inherent अ " in the consonant letter.

When the simple consonant without the inherent अ is specifically to be expressed, a sign (right-slanting stroke), called Hal (or Halanta), is put below the letter. Thus k=कृ , r=रृ , d=दृ , etc.

When some vowel other than the inherent अ comes after a consonant an abbreviated form of that vowel (called Matra) is tagged on to the consonant letter and is never written in full. Thus, k+i=क+इ is written as कि, k+u=क+उ is written as कु, and not as कइ, कउ, which indicates the pronunciation k-i, k-u.

The abbreviated forms of vowels i.e. the Matras when they come after consonant letters are written as follows:-

आ = ऀ , इ = ँ , ई = ऌ , उ = ं , ऊ = ः  
 ऋ = ऄ , ए = ॆ , ऐ = ॆ̄ , औ = ॆ̄̄ , औ = ॆ̄̄

Of these, ऀ (आ), ऌ (ई), ॆ̄ (औ) and ॆ̄̄ (औ) are written after the consonant, whereas ँ (इ) is written before, ं (उ), ः (ऊ) and ऄ (ऋ) are written below, and ॆ (ए) and ॆ̄ (ऐ) are written above. Thus :-

क + आ = का	क + ऋ = कॄ
क + इ = कि	क + ए = कॆ
क + ई = की	क + ऐ = कॆ̄
क + उ = कु	क + औ = कौ
क + ऊ = कू	क + औ = कौ̄

Important exceptions :- इ + उ = उ, and ए + उ = ए.

If a vowel is nasalized (Anunasika), the sign ँ (candra-bindu) is placed above the letter: कं, कां, कुं, कौं, but if the Matra is above the headline, only dot is used instead of ँ thus किं, कीं, कें, कैं, कौं, कौं. It is to be noted that the dot is placed on the right of the Matra.

The Visarga ':' is always placed after the vowel or consonant + vowel. Thus दुःख (dukh) 'pain, sorrow, unhappiness', निःसीम (nihsim) 'limitless'.



The Anuswara '•' is placed above the vowel (e.g. अंक ) or consonant + vowel after which it is pronounced (e.g. आनंद ).

### Mode of Writing : Consonants

Two or more consonants (with no vowel, including the inherent अ between them) can be combined together and thus form a "conjunct".

क + क = कक (kka) is a conjunct, so is क + या = क्या (kya) 'what?'

It is, however, not usual to write conjunct with the help of a Hal mark as above in क्या . This mark is used with the final consonant of a Sanskrit word [as in महान् (Mahan) 'great'] and with ड, छ, ट, ठ, ढ, and ढ (e.g. वाङ्मय , विद्या , कुटी ).

Most of the consonants formed and ending with a vertical stroke joined to the following consonant by removing the vertical line. Thus ग + घ = गघ , च + छ = चछ , त्र + थ = त्रथ , ज्ञ + य = ज्ञय , etc.

Those ending in a vertical half-stroke drop the same क + य = क्य , फ + य = फ्य .

The rest, which end in neither a full nor a half-vertical stroke, viz. ड, छ, ट, ठ, ड, ढ, ढ, and हे, do not change. When combined with a following consonant, they may be written with a Hal mark. Thus ड + क = डक , ट + ठ = टठ etc. The general practice is to write them in full. While, in case of the following consonants, the consonant is written below them with the horizontal stroke omitted: ट + ट

=  $\text{हृठ}$  ,  $\text{इ} + \text{ठ} = \text{इठ}$  ,  $\text{हृ} + \text{थ} = \text{हृथ}$  ,  $\text{इ} + \text{क} = \text{इक}$  ,  $\text{हृ} + \text{ग} = \text{हृग}$  ,  $\text{हृ} + \text{ब} = \text{हृब}$  , however,  $\text{हृ} + \text{ल} = \text{हृल}$  .

Exceptional forms :-

(a)  $\text{र}$  when combined with a following consonant is written thus i.e. above the consonant :  $\text{रु} + \text{ग} = \text{गृ}$  ,  $\text{रु} + \text{घ} = \text{घृ}$  ,  $\text{रु} + \text{ङ} = \text{ङृ}$  ,

But when  $\text{र}$  follows a consonant, having a vertical stroke, it is written as a left slanting stroke below and to the left of the vertical stroke:

$\text{कृ} + \text{र} = \text{कर}$  ,  $\text{जृ} + \text{र} = \text{जर}$  , also  $\text{इ} + \text{र} = \text{इर}$  .

When preceded by  $\text{ट}$  ,  $\text{ठ}$  ,  $\text{ड}$  ,  $\text{ढ}$  ,  $\text{ण}$  and  $\text{ह}$  , it is written thus below:

$\text{टृ} + \text{र} = \text{ट्र}$  ,  $\text{ठृ} + \text{र} = \text{ठ्र}$  ,  $\text{डृ} + \text{र} = \text{ड्र}$  ( or  $\text{ड़}$  ) .

(b)  $\text{कृ} + \text{ष} = \text{क्ष}$  ksha,  $\text{तृ} + \text{र} = \text{त्र}$  tra,  $\text{जृ} + \text{ञ} = \text{ज्ञ}$  .

(c) The pronunciation of Anuswara ( . ) is like  $\text{ङ्}$  ,  $\text{ञ}$  ,  $\text{ण}$  ,  $\text{न}$  ,  $\text{म}$  depends on the following consonants.

$\text{कंघा} = \text{कङ्घा}$  'comb' .

(d)  $\text{हृ} + \text{य} = \text{ह्य}$  ,  $\text{हृ} + \text{म} = \text{ह्यम}$  ,  $\text{कृ} + \text{य} = \text{क्य}$  ,  $\text{कृ} + \text{म} = \text{क्यम}$  ,  $\text{रृ} + \text{म} = \text{र्यम}$  ,  $\text{रृ} + \text{य} = \text{र्य}$  which is frequently written as  $\text{र्य}$  .

(e)  $\text{हृ} + \text{व} = \text{ह्व}$  ,  $\text{हृ} + \text{थ} = \text{ह्वथ}$  and  $\text{हृ} + \text{र} = \text{ह्वर}$  .

(f)  $\text{पृ} + \text{न} = \text{प्न}$  ,  $\text{पृ} + \text{म} = \text{प्नम}$  ,  $\text{कृ} + \text{र} = \text{कर}$  ,  $\text{कृ} + \text{र} = \text{कर}$  ,

$\text{कृ} + \text{न} = \text{क्यन}$  ,  $\text{कृ} + \text{त} = \text{क्यत}$  ,  $\text{कृ} + \text{व} = \text{क्यव}$  ,  $\text{कृ} + \text{य} = \text{क्यय}$  ,

$\text{शृ} + \text{र} = \text{श्र}$  ,  $\text{शृ} + \text{व} = \text{श्रव}$  ,  $\text{शृ} + \text{य} = \text{श्रय}$  ,  $\text{पृ} + \text{य} = \text{प्य}$  ,

$\text{जृ} + \text{ल} = \text{ज्ल}$  ,  $\text{जृ} + \text{य} = \text{ज्य}$  .

Every conjunct, like a simple consonant, can be combined with any vowel-sign or with the inherent  $\text{अः}$

$\text{क} + \text{र} + \text{ई} = \text{क्री}$ ,  $\text{र} + \text{र} + \text{ई} = \text{री}$ ,  $\text{क} + \text{ष} + \text{उ} = \text{कु}$ ,  
 $\text{क} + \text{य} + \text{औ} = \text{क्यौ}$ , (kyo) 'why?'

In combining more than two consonants, the same rules are followed.

$\text{स} + \text{र} + \text{य} = \text{सरय}$ ,  $\text{ज} + \text{क} + \text{य} = \text{जकय}$  or  $\text{डकय}$ ,  
 $\text{ग} + \text{र} + \text{य} = \text{गरय}$ ,  $\text{न} + \text{स} + \text{य} = \text{नसय}$ ,  
 $\text{र} + \text{न} + \text{य} = \text{रनय}$ ,  $\text{र} + \text{क} + \text{ष} + \text{य} = \text{रकषय}$  etc.

## **CHAPTER 2**

# **AN INTRODUCTION TO THE SYSTEM**

## CHAPTER 2

### AN INTRODUCTION TO THE SYSTEM

The System is designed to recognize Hindi text. It is trained using algorithms and a huge database to emulate man in his sense of sight, memory i.e. an ability to recapitulate, a sense of learning and a tendency to err. At this point it won't be an exaggeration to say that an effort is being made to create an "Artificial Intelligence", which would in no way supercede man's creativity. But, only enhance his will power and capability to create more such systems.

The system takes Devanagari script text as input. It breaks the given text into lines, from the lines it extracts words and finally characters. For each of the individual characters, it extracts significant features and recognizes the characters and other additional forms in the text and outputs the obtained results. A comparison of the input and output text gives us the efficiency and the error rate.

The above system is broadly classified into the following subsections, namely,

1. Data Collection & Data Acquisition
2. Segmentation
3. Analysis of characters without preprocessing
3. Preprocessing
4. Analysis of characters with preprocessing
5. Referencing Database

6. Analysis of Half forms
7. Use of Heuristic
8. Training Algorithm
9. Results

## **2.1 DATA COLLECTION & DATA ACQUISITION**

The input text has been taken from Hindi books of standard I, II & VII. The algorithms have also been tested on hand-drawn characters.

The input text is scanned using a hand scanner and the image data is stored using a PC Paintbrush file format (PCX). The image data is read from the file, binarized and stored in a buffer.

## **2.2 SEGMENTATION**

Histograms are plotted for the entire text. Using the histogram information , individual lines of text are retrieved. Next, vertical histograms are plotted for individual lines of text. Using this information words and further characters are extracted.

## **2.3 ANALYSIS OF CHARACTERS WITHOUT PREPROCESSING**

An attempt is made to recognize the characters without any prior preprocessing. This is done to make the computations faster and more accurate. The application of this strategy assumes the existence of isolated characters. A limit is defined for the width to height ratio of an isolated character. To begin with the width to height ratio of a

character is calculated and compared with the limit defined for an isolated character. In case the character is isolated then the approach without prior preprocessing is adopted, otherwise the second approach with the preprocessing stage is adopted.

In case of the above approach without the preprocessing stage, the features extracted are the following: horizontal and vertical histograms in each of the four quadrants of a character window, the distance of the start of the character taken at five equally distributed points from the left and top boundary of the character window, presence of a vertical or a horizontal line.

A database is created and the features of the characters obtained without preprocessing are stored. The database is referenced during the recognition phase.

#### **2.4 PREPROCESSING**

The individual characters extracted are processed for noise removal, thinning and smoothing.

An attempt has been made to design a new thinning algorithm based on morphological operators and much success has been obtained in this connection.

#### **2.5 ANALYSIS OF CHARACTERS WITH PREPROCESSING**

The individual characters and their matras are analyzed separately. The significant features of individual characters are determined like terminating points, junction points etc.

These features are used for distinguishing among various characters.






## 2.6 CREATING & REFERENCING DATABASE

A separate database is created to store the features for the various characters, and another to store the features of the matras. The input character is analyzed and its features are compared with those stored in the database and a confidence level of above 70% declares the character to be the same as the one with which it is being compared.

## 2.7 ANALYSIS OF HALF FORMS

A strategy is adopted for identifying half characters and also separating the half characters from their joined counterparts.

## 2.8 USE OF HEURISTIC

The system specifies certain rules for identifying matras like I (  ), U (  ), etc. For instance, a vertical line followed by a character KA (  ) with I (  ) on top is identified as a character KI (  ).

## 2.9 TRAINING ALGORITHM


An attempt was made to train the system to recognize various fonts using a standard algorithm of Neural Networks, "Vanilla Backpropagation Algorithm". We did succeed to a certain extent but, there were certain major drawbacks like, the training process was very time consuming. Another



disadvantage was that the algorithm failed to recognize a completely new font. The experts in the field of Neural Networks say, that if a system has to emulate human senses then it has also to pass through a similar childhood, which may extend for days, months or even years, and like any child it has to be taught to recognize any new entity.

A second convenient method has been adopted to recognize texts of various fonts. A separate database is created for each font and each of the databases is assigned a code. The system is allowed to analyze texts of some few chosen fonts. Initially the user is asked for the font code of the text he wishes to be analyzed. The system then selects the particular database for that font and carries on with the recognition phase.

#### **2.10 RESULTS**

Codes in English are used for corresponding Hindi characters, like KI for  . The final results are in coded English format.

## **CHAPTER 3**

# **DATA COLLECTION & DATA ACQUISITION**

## CHAPTER 3

### DATA COLLECTION & DATA ACQUISITION

#### 3.1 INTRODUCTION

All research works and projects are based on raw input data and algorithms or strategies to extract, manipulate and analyze these data to accomplish the desired study.

The raw data may be obtained through any of the input devices such as the following:

1. Keyboard.

2. Tablets

a) Electronic

Electronic tablets that capture the x-y coordinate data of pen-tip movement. They can be used to input sketches and drawings.

b) Electromagnetic

c) Electrostatic

d) Pressure Sensitive

3. Recent advancements bring together tablets and flat displays on the same surface. Thus, serving the dual purpose of both input and output.

4. Camera.

5. Scanner.

The characters of Devanagari Script served as the input data for this project. The text was scanned and converted into

a bitmap form to be used as input.

### 3.2 DATA COLLECTION

The samples used in this study were obtained from various Hindi books. To begin with the study was performed on text of considerably large size like, standard I Hindi books. Then, the same algorithms were tested successively on class II, class V and class VII books. Once satisfied with the performance of the algorithms, to be discussed in subsequent chapters, the same algorithms were subjected for testing on hand drawn symbols. The test results showed reduced efficiency. For instance, in case of printed text of a fixed font, the test results showed an accuracy of above 80%, whereas the use of the same database in case of hand drawn characters collected from various individuals, reduced the efficiency rate drastically to 50-60%. The main characters which proved to be the cause of confusion were प and म , भ and म .

The main cause of confusion among certain similar looking characters is that the thinning algorithms used in the preprocessing stage tend to wipe out certain significant features of the characters. For instance, a म after thinning tends to take the shape of either म or प . The latter shape is responsible for the confusion arising between a प and a म . These kinds of discrepancies are generally resolved at a higher stage i.e. during semantic analysis phase.

### **3.3 DATA ACQUISITION**

#### **3.3.1 Scanning**

The images were scanned using a hand scanner. The resolution was set at 100 dpi. The scanned bitmap image was obtained using PC Paintbrush software. The software stores the bitmap image in a specific format. Hence, it becomes necessary to understand the structure of the file, in order to read it using our own programs written in a specific language.

The PCX format uses run length encoding to compress image data. It is not as efficient as compared to other file formats like MacPaint, IMG, TIFF. The compressed PCX file is usually longer by a sizeable margin as compared to the other files.

PCX file format was used for obtaining the bitmap images as it was the easiest among the above mentioned file formats and the desired results were obtained.

### **3.4 BINARIZATION**

The gray scales varying from 0-255 are thresholded using a mid-value to two distinct values of either 0, indicating an 'off' pixel or to a value 255, indicating an 'on' pixel.

### **3.5 SUGGESTIONS**

After working with image files and surveying the various possible file formats. I would suggest the use of the tagged image file format (TIFF) to obtain the bitmap images.

The TIFF file format is extremely flexible. It can support images of any size, in monochrome or in upto 24 bits of color. It is portable with different architectures. The only negative aspect to TIFF files is that they prove to be extremely complex to unpack because of their highly variable nature.



TH-5608

**CHAPTER 4**  
**SEGMENTATION**

# CHAPTER 4

## SEGMENTATION

### 4.1 INTRODUCTION

Segmentation can be defined as a process by which an image is subdivided into its constituent parts or objects. This process enables the extraction of objects of interest from an image, such that these entities can be subjected to further processing and analysis Gonzalez[28].

### 4.2 IMPLEMENTATION DETAILS

The segmentation approach followed for this project is a top down one. The top down approach begins by considering the entire text and works down successively to the level of a character.

The approach described in this section is similar to the one proposed by R.M.K. Sinha and H.N.Mahabala in the paper authored by them, "Machine recognition of devanagari script", Sinha et al.[32]. In the following paper we have extended their work.

#### 4.2.1 Line Segmentation

Horizontal histogram values are determined for the entire text. It is observed that groups of lines of varying length are separated by some space. This space signifies the space between any two lines of text.



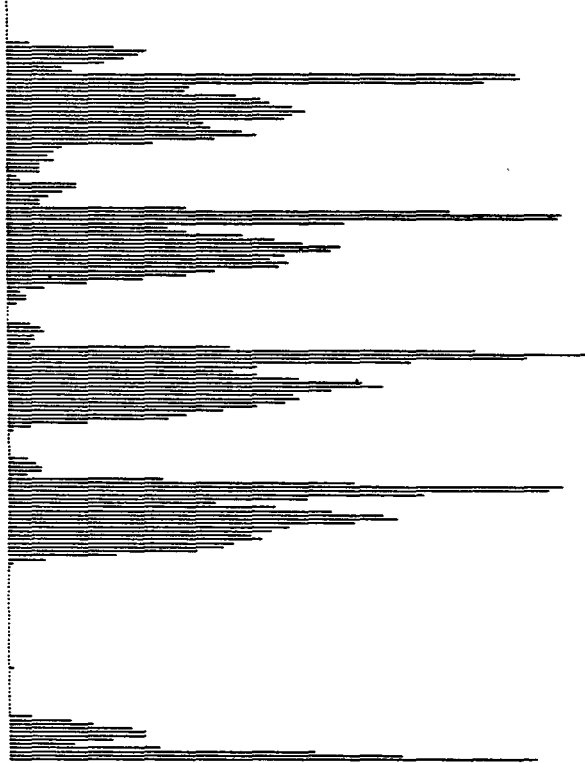


Fig.4.1 Horizontal histogram of a given line of text.

Among each group of lines, it is observed that two or three of the histogram lines have maximum length. These lines signify the horizontal line drawn on top of each of the Devanagari script characters.

Initially, after white space the histograms begin with values less than the maximum value taken up by horizontal top lines signifying the presence of an upper matra zone or it may be white space immediately followed by maximum value specifying the absence of any upper matra zone. The maximum value zone signifies the presence of the horizontal top line. Immediately after the maximum value zone, the value of the histogram decreases and after a certain length reduces to zero signifying the end of text line.

With the help of the Line\_histogram values, individual lines of text can be separated out.

After the initial segmentation stage we have reached to the stage of individual lines. Next two histograms are plotted for each individual text line, with respect to the columns. The first histogram takes its starting point just at the beginning of horizontal top line. This histogram forms the Word\_histogram and is used to separate out each individual word contained in a given line of text. The second histogram takes its starting point at the end of the horizontal top line. This histogram forms the Character\_histogram and is used to separate out each character in a given word.

#### **4.2.2 Word Segmentation**

Word\_histogram is obtained at this stage. Since, the beginning point for this histogram is taken as the start of the horizontal top line. Hence, the separation between two

horizontal top lines signifies the separation between two words and is marked out by the space between groups of lines in a Word\_histogram. Each group of lines signifies a word and the space between any two groups of lines signifies the space between words.

After this segmentation level we have reached to the stage of words. Next, we take each individual word and draw its histogram with respect to the columns. But, in this case the start point is taken as the end of the horizontal top line.

#### **4.2.3 Character Segmentation**

Character\_histogram is obtained. Since, the beginning point of this histogram is taken as the end of the horizontal top line. Hence, it can be safely assumed that the information till the horizontal top line is completely wiped out.

The separation between groups of lines in a Character\_histogram would signify the separation between two characters.

After this segmentation stage we have reached to the level of each individual characters in a word. The handling of matras would be explained later in the chapter on 'Rule Based System'.

##### **4.2.3.1 Division of Characters into zones**

Each character is divided into four distinct zones.

##### **i) Upper Matra zone**

The zone from the beginning of the text to the beginning of horizontal top line is marked as the upper matra zone.

ii) Horizontal Top Line zone

The zone following the upper matra zone from the beginning of horizontal top line to its end is identified as the horizontal top line zone.

The zone starting at the end of horizontal top line to the end of the text line is identified as the lower character zone. Each character is defined to be of a certain maximum height, say ideal character height, which is fixed for a particular font. Using this information the lower zone of a character is split up into two zones, namely,

iii) Simple Character zone

The simple character zone would be defined as,  
simple character height = ideal character height, i.e.  
from the end of horizontal top line to the extent of an ideal character height.

iv) Lower Matra zone

The lower matra zone would be defined as,  
lower matra height = maximum lower character zone height -  
ideal character height, i.e.  
from the end of simple character to the end of text.

## **CHAPTER 5**

# **FEATURE EXTRACTION WITHOUT PREPROCESSING**

## CHAPTER 5

# FEATURE EXTRACTION WITHOUT PREPROCESSING

### 5.1 INTRODUCTION

This particular approach extracts features from the characters without the application of any of the preprocessing stages. This is done to make the computations much faster because, 'thinning' which forms one of the preprocessing stages consumes nearly half the time required for the analysis of an entire character. Moreover, as a result, of thinning significant information may be deleted from an image. This may cause difficulty in the recognition process.

An attempt is made to recognize the characters without any preprocessing. The basic assumption for this approach is that characters are isolated. A limit is defined for the width to height ratio of an isolated character. To begin with the width to height ratio of the characters is calculated and compared with the limit defined for an isolated character. If the calculated width to height ratio is less than equal to the limit then it is assumed that the character is isolated and feature extraction without preprocessing is attempted. In case the width to height ratio of the character is greater than the limit then it is assumed that the characters are either joined ( रु ) or one character comes under the shadow of another character ( स ), and feature extraction after preprocessing

is attempted.

## 5.2 REVIEW OF EARLIER WORK

### 5.2.1

Kimura et al.[4]

The authors focus their study on recognition of isolated characters based on feature extraction without preprocessing. The two set of features used in their algorithms are the following: the first set of features is the histograms in the chain codes of the contour elements. The second set of features is evaluated from the profiles on the binary image of a numeral.

In the process the rectangular frame enclosing the normalized contours is divided into 4x4 rectangular zones. In each zone, a local histogram of the chain codes is calculated. The feature vector is composed of these local histograms.

The profile features are derived from the profiles of the external contours. These are character widths, ratio, location of extrema, and discontinuities in character profiles.

### 5.2.2

Taxt et al.[45]

This is another approach that avoids the traditional thinning and vectorization process. This approach takes the outer pixel boundary of an isolated symbol candidate in the binary raster image as a simple closed curve. This curve is then approximated by a parametric spline curve, an elliptic Fourier expansion due to Zahn and Roskies. Curvature values and coordinates along the spline curves or the coefficients of the Fourier expansion are then used as descriptors in a

statistical classification scheme.

### 5.3 IMPLEMENTATION DETAILS

The features included in this approach are the following: the first set of features is the horizontal and vertical histograms determined in each of the four quadrants of the character window, the second set is profile features, the third set is the presence of horizontal and vertical line segments in the character and the fourth set is analyzing the left contour of the character for bend points, direction of maximum curvature, direction of transition.

The composite characters are separated from their matras and the characters and matras are analyzed separately. For the final analysis certain amount of heuristic is required to code up for a composite character (  $\overline{\text{रि}}$  ), consisting of the characters (  $\overline{\text{र}}, \overline{\text{ि}}$  ) and their matra.

To begin with the character is enclosed in a tight window. The window is divided up into four equal quadrants.

#### 5.3.1 Features

1. Horizontal and Vertical histograms in each of the four quadrants.

The value of the horizontal and vertical histograms are determined in each of the quadrants at three equally distant points.

#### 2. Profile features

The features associated with the characters are derived from their external contours. They are:

a) left profiles, which is a collection of the distances of



the left profiles form the left boundary of the character. (Kimura et al.[4]).

b) top profiles, which is a collection of the distances of the top profiles from the boundary of the character.

### 3. Vertical and Horizontal line

Vertical and horizontal histograms of the entire character are studied to determine the presence of a horizontal or vertical line. A limit is defined for the horizontal and vertical histogram values to indicate the presence of a horizontal and vertical line segment. Next the character window is split into zones i.e. into three equal parts in the vertical direction forming the left, middle and right zones and into three equal parts in the horizontal direction forming the top, middle and bottom zones. Further, it is checked in which zone the line segment lies.

### 4. Contour Analysis

The left half of the character contour is analyzed. This is based on the assumption that the maximum features of a character lie on the left half. Hence, this saves on computation time. The contour boundary of the character is treated as a line segment and is analyzed for bend points Baptista et al.[5], direction of maximum curvature and direction of transition. These features are to be discussed in detail in the chapter on feature extraction after preprocessing.

**CHAPTER 6**  
**PREPROCESSING**

# CHAPTER 6

## PREPROCESSING

### 6.1 INTRODUCTION

Preprocessing stage is an intermediate stage, which consists of noise removal, skew correction, and thinning. According to some authors like Brown et al.[30], there are important interactions between the preprocessing of character images and the feature extraction process. Feature extraction or shape measurement can be misled if the images have little or no preprocessing. Noise-contaminated descriptions of character images can lead to mislabelling by the recognition logic. Preprocessing stage has been incorporated as a vital stage in many systems. Brown et al.[30], Sinha et al.[31,32], Lu et al.[41].

This stage primarily comprises of three substages : noise removal, thinning and smoothing. Preprocessing is an essential step. It takes care of the noise, which might be introduced during the scanning stages or because the input data is not of good quality. The thinning phase reduces the input data to a bare skeleton form. This enables the information about the object to be preserved and also enables the extraction of key features with minimum possible computation. Smoothing, further helps to smooth noisy boundaries, and also helps in retaining a good representation of boundary corners and takes care of broken joints and edges.

The order in which the above stages were applied are as follows:

- a) Noise Removal
- b) Thinning
- c) Smoothing

Noise removal and smoothing is achieved using morphological operators. An attempt is made to design a new thinning algorithm based on morphological operators.

## **6.2 NOISE REMOVAL**

This preprocessing stage, takes care of possible noises that might be introduced during the scanning stage. Noise is an unavoidable menace in image processing applications. Algorithms can be designed for its effective removal. The first goal is to detect the different kinds of noise that could possibly be introduced and if possible the conditions for their existence. Efforts should be made to eliminate such conditions initially during the scanning phase. This would enable minimization of noise.

The use of a hand scanner is to a great extent responsible for the introduction of noise. A shaky hand also leads to introduction of noise. A peculiar problem faced in case of the above project was scanning the end of a text page using a hand scanner. This would lead to pressure variations due to the thickness of the copy or book from which the text was being scanned. While, processing it was observed that these portions of the text would fill up with non-Ascii characters. Hence, a check was made at the preprocessing stage which checked for the existence of such non-Ascii characters in the scanned image, which were replaced with white space.

The noise removal procedure in case of the above application used morphological filters. The templates used were 3x3, they are shown as under,

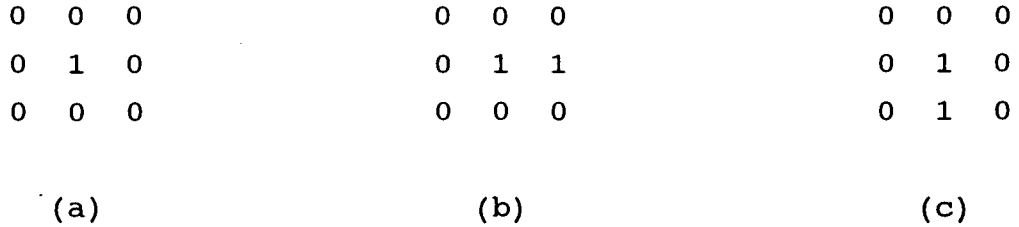


Fig. 6.1 Templates for noise removal.

The templates shown above were moved around the image matrix and in case a bitmap AND operation resulted in a value of one then the pixel in the center, say P of the template was marked for deletion.

In the above case the order of the pixels is given as under,

P0	P7	P6
P1	P	P5
P2	P3	P4

Fig. 6.2 8-neighbors of P

The pixel P is surrounded by its 8 neighbors, P0, P1, P2, P3, P4, P5, P6, P7, where the neighbors P1, P3, P5, P7 form the 4-connected neighbors and P0, P2, P4, P6 form the diagonal neighbors, and P0, P1, P2, P3, P4, P5, P6, P7 form the 8-connected neighbors.

The template shown in Fig. 6.1(a) would take care of isolated noise pixels. In the case of templates shown in Fig.

6.1(b) and (c), it was assumed that a single line of 'on' pixels in a raw image would indicate the presence of noise, since the original raw image consists of three to four 'on' pixel lines per segment.

### 6.3 THINNING

Thinning is a fundamental preprocessing stage in image processing applications. This technique is useful in recognition and interpretation of images, because it decreases the data amount while preserving the shape features of an input picture. Thinning consumes considerable time in processing an image. Therefore, the algorithm chosen should minimize on the number of iterations required and also maintain the connectivity of the thinned image.

Many proposals have been made for thinning algorithms. The thinning algorithms are classified into parallel algorithms and sequential algorithms based on their implementation. Suzuki et al.[40] proposes an algorithm for digital binary pictures. The algorithm presented repeats the removal of the deletable border points in parallel and the extraction of the final points. Hall[29] proposes optimally small operator supports for fully parallel thinning algorithms. The author suggests eleven pixel supports as the smallest possible supports, and the possible positions of the support pixels are shown to be well constrained. Zhang et al.[46] suggests a fast parallel thinning algorithm consisting of two subiterations. One aimed at deleting the south-east boundary points and the north-west corner points while the other is aimed at deleting the north-west boundary points and the south-east corner points.

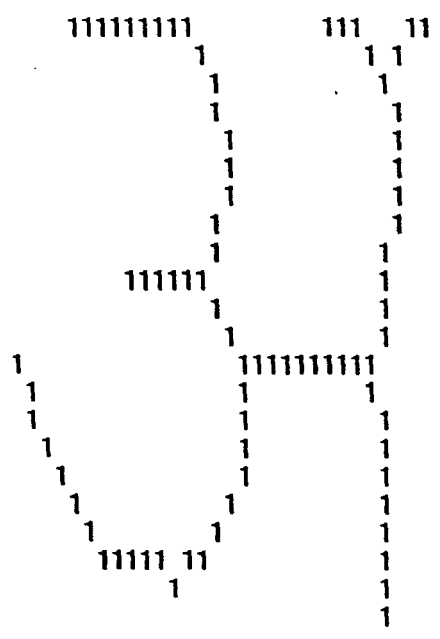


Fig.6.4 Thinned character

Among the algorithms stated above, the one suggested by Zhang et al.[46] was applied to the input text used in this project. The algorithm failed to give the desired results. In most of the cases the characters were nearly wiped out.

For this project images of size 30x30 or more were thinned using Rosenfeld & Kak algorithm [35]. The results obtained were satisfactory. But this algorithm failed in case of images of size less than 30x30. In smaller sized images the significant features were wiped out. Thus, it proved to be a disadvantage. For instance, a  $\alpha$  after thinning would assume the shape of a  $\bar{\alpha}$ , where the significant feature  $\alpha$  was found to be missing. Another example is of a  $\tau$  being thinned down to  $\bar{\tau}$  instead of  $\tau$ .

A thinning algorithm based on morphological operators was designed to thin down images of size smaller than 30x30. This algorithm gave satisfactory results for the smaller sized images as compared to Rosenfeld & Kak algorithm. But, this algorithm failed in case of images exceeding the size 30x30. Hence, two different algorithms were opted for in this project and the system would switch between these two algorithms depending on the font size.

Another, possibility could be to increase the resolution in case of smaller sized fonts. But, in such cases modifications have to be made in the algorithms used for reading the image files. If possible the system could switch between these algorithms depending on the font size. The latter possibility has not yet been included in the software.

### **6.3.1 Rosenfeld & Kak Algorithm**

This algorithm is a shrinking process which deletes from  $S$ , at each iteration, border points whose removal does not locally disconnect their neighborhoods. This algorithm



guarantees that the connectedness properties of S do not change, even if all such points are deleted simultaneously. This algorithm prevents an already thin arc from shrinking at its ends, since the points having only one neighbor in S are not deleted.

The algorithm deletes only the border points that lie on a given side of S, i.e. that have a specific neighbor (north, east, south, or west) in S, at a given iteration. The algorithm ensures that the skeleton is as close to the "middle" of S as possible, for this it uses opposite sides alternately, e.g. north, south, east, west.

Conditions under which a border point can be removed.

The border point P of S is called simple if the set of 8-neighbors of P that lie in S has exactly one component adjacent to P. For instance, in case of 4-connectedness for S, one cares only about components that are 4-adjacent to P.

For example, P is 4-simple if its neighborhood is,

$$\begin{array}{ccc} 0 & 1 & 1 \\ 0 & P & 0 \\ 1 & 0 & 0 \end{array}$$

In this case only one 4-component of 1's is 4-adjacent to P. But, P is not 4-simple if its neighborhood is,

$$\begin{array}{ccc} 0 & 1 & 1 \\ 0 & P & 0 \\ 0 & 1 & 0 \end{array} \quad \text{OR} \quad \begin{array}{ccc} 0 & 1 & 0 \\ 0 & P & 1 \\ 0 & 0 & 0 \end{array}$$

P is 8-simple in the third case, but not in the first two cases.

Deleting a simple point from S does not change the connectedness properties of either S or  $S - \{P\}$ ;  $S - \{P\}$  has the same components as S, except that one of them now lacks the point P, and  $S \cup \{P\}$  has the same components as S, except that P is now is one of them.

The thinning algorithm is stated as follows: "Delete all border points from a given side of S, provided they are simple and not end points. Do this successively from the north, south, west, north,.... sides of S until no further changes take place."

#### **6.3.2 Thinning Algorithm based on Morphological Operators**

The thinning algorithm basically consisted of moving certain templates around the image and if a bitmap AND would result in a one then the particular center pixel for that position was marked for deletion.

Initially, a counter is initialized to zero. A new image matrix, say S is created and initialized to all zeros. A search is made for the templates belonging to the first group. In case the search succeeds then the corresponding center pixel for the particular window is marked in the new matrix S, and the counter is incremented. After a search for the templates in Group I is completed then the image matrix M is updated i.e. for the positions in image matrix S which are marked, the corresponding positions in image matrix M are deleted.

# THINNING ALGORITHM BASED ON MORPHOLOGICAL OPERATORS

## GROUP I

$$\begin{array}{ccc} & 1 & 1 \\ 0 & 1 & 1 \\ & 1 & 1 \end{array}$$
 (a)

$$\begin{array}{ccc} & 0 & \\ 0 & 1 & 1 \\ & 1 & 1 \end{array}$$
 (b)

$$\begin{array}{ccc} & 0 & \\ 0 & 1 & 1 \\ & 0 & 1 \end{array}$$
 (c)

$$\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 1 \\ & 0 & \end{array}$$
 (d)

## GROUP II

$$\begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \end{array}$$
 (a)

$$\begin{array}{ccc} & 0 & \\ 1 & 1 & 0 \\ & 1 & \end{array}$$
 (b)

$$\begin{array}{ccc} 1 & 0 & \\ 1 & 1 & 0 \\ & 0 & 1 \end{array}$$
 (c)

$$\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 1 & 1 \\ & 0 & \end{array}$$
 (d)

$$\begin{array}{ccc} 1 & 0 & \\ 1 & 1 & 0 \\ 0 & & \end{array}$$
 (e)

In the second stage, the matrix  $S$  is again initialized to zero and a search is made for the templates in Group II. If a search succeeds then the particular position in image matrix  $S$  is marked and the counter is incremented. At the end of second stage, the original image matrix  $M$  is updated.

At the end of the above two stages, a check is made on the value of counter. If the value of the counter is found to be zero then the process is terminated, else the control goes back to step one.

The thinning algorithm works towards deletion of points from characters alternately in the horizontal and vertical directions. This is taken care of by alternate application of templates from Group I and Group II. The algorithm ensures that the skeleton obtained is as close to the middle line as possible.

#### Efficiency

The CPU time (in seconds) consumed by the above algorithm is 0.41 seconds for a 386-system. The algorithm proves to be very efficient in case of images with minute details and can obtain skeletons of images without the need for increasing the resolution. It preserves the edges and corners. However, the efficiency sharply drops in case of images which originally have very thick edges or borders.

#### 6.4 IMPLEMENTATION DETAILS

The characters segmented are thinned separately zone wise. That is, initially the upper matra zone is checked for

the possibility of existence of an upper matra, if found then the upper matra is separated from the rest of the character, thinned and analyzed separately. Next, the simple character which begins at the end of the horizontal top line and extends to the ideal character height is thinned and analyzed separately. A possibility for the existence of a lower matra is checked. If found then the lower matra is thinned and analyzed separately.

### 6.5 SMOOTHING

Smoothing is essential step in shape analysis and image interpretation. The smoothing algorithm in this project has been applied after the thinning stage to take care of small holes which might be created by the thinning process. For instance, consider the templates shown below:



Fig. 6.4. Templates for smoothing

The templates shown in Fig. 6.4 (a) and (b) represent broken line segments in horizontal and vertical directions respectively. The smoothing algorithm helps in detecting notches in joints and segments and fills them up.

**CHAPTER 7**  
**CHARACTER RECOGNITION**

# CHAPTER 7

## CHARACTER RECOGNITION

### 7.1 INTRODUCTION

Following Baptista et al.[5]

All character recognition algorithms depend on primitive operations of some sort to extract features from patterns. However, differences may arise in the organization, control and use of primitive operations, leading to the grouping of most character recognition algorithms into three categories, Syntactic, Deterministic and Decision Theoretic or Statistical.

The syntactic approach is based on an attempt to exploit the obvious structural properties inherent in many patterns, and use formal grammars to characterize and ultimately identify the character. However, while description of the character is achieved very elegantly, identification by this approach leads to highly unwieldy and complex grammars. Further complications arise by trying to incorporate learning ability into this type of approach.

In the Deterministic approach, the primitive operations are all executed and the resulting features stored in a table. Identification and incorporating learning into the identification procedure is an extremely simple task achieved by table matching and augmenting strategies. However, if the number of features is larger there could be an explosion of

the data-base. This can be avoided by the design of better features or staggering the feature detection procedure, and thereby grouping the patterns into subclasses.

Features are selected on the basis of their invariance to distortion, style variation, translation, rotation to a certain extent, and speed and accuracy of recognition.

## **7.2 KNOWLEDGE-BASED PATTERN RECOGNITION USING SYNTACTIC APPROACH**

Following Yang et al.[6]

The conventional methods of pattern recognition can be identified as statistical and syntactic approaches. These approaches are not satisfactory since the solution depends largely on the knowledge and experience of the experts. To cope with this problem, many researchers are trying to introduce the expert system techniques into the field of pattern recognition system of much more sophisticated recognition capability. At this stage, the researchers face with a problem to identify a method to represent the expert's knowledge by the grammar production. A popular viewpoint is that a grammar production [A -> B] represents the knowledge [If A then B]. Mostly, the grammar production and the implication of mathematical logic are confused. This confusion makes it impossible for the grammar production to represent the knowledge correctly, and therefore the knowledge-based pattern recognition system using syntactic method has not been fully realized.

The author has proposed a new type of knowledge-based pattern recognition system, in which the attributed grammar is



used to represent knowledge and the Early algorithm or ED algorithm is used to search the conclusion.

In this paper, the author proposes a knowledge-based pattern recognition system based on syntactic approach . The proposed system consists of two parts: the basic part and the inference part. In the basic part, after preprocessing and feature selection we obtain the initial recognition result of subpatterns using the conventional pattern recognition approach. Then the results of the basic part are carried to the inference part as its primitives. Some information, for instance, the a priori knowledge and background knowledge etc., inputted by man-machine interaction, can also be handled as the primitives in the inference part. These primitives form the input sentence of the inference part. In the inference part the syntactic recognition system is used as an expert system, where the syntax analysis plays the role of a tool for inference, and the final result of syntactic analysis is the conclusion of inference.

In this paper, the author discusses two possible applications:

- 1) A knowledge-based pattern recognition system for tracking the events in the seismic sections.
- 2) A vertigo diagnosing system based on the syntactic approach.

### **7.3 FEATURE EXTRACTION AFTER PREPROCESSING**

#### **7.3.1 Introduction**

The features after preprocessing in the recognition phase are an extension of the work done in this area by Baptista et

al.[5]. It includes a few more new features, which help in strongly differentiating between different characters. For instance,

- a) direction of transition of characters  
like Anticlockwise from left to top.
- b) maximum curvature in a particular direction  
like maximum curvature in the East direction.

This project also refers to a dissertation presented by Shyam S. Sareen under the guidance of Prof. K. K. Biswas, I.I.T. Delhi[39], and incorporates some of the procedures used for removing fictitious feature points suggested in the paper.

### **7.3.2 Implementation Details**

To begin with horizontal and vertical histograms are plotted for the character. Using the values of the horizontal and vertical histograms, the character is bounded in a rectangular window. Next the character is analyzed for extraction of the features.

#### **FEATURES**

##### **1) VERTICAL LINES**

A vertical histogram value greater than a certain threshold indicates the presence of a vertical line. The vertical histogram values of a given character are studied to find the desirable maximum. The ideal maximum peak indicates the presence of a vertical line of desired length. Next the rectangular character window is divided into three equal blocks. The blocks lying towards the left, the middle and the

right.

A search is made to see in which block the vertical line lies. The results obtained are used in initialising three given variables.

```
vertline_left  
vertline_mid  
vertline_right
```

For instance, consider the case of the character .  
In this case, the vertical line lies on the right block.  
Hence, the three variables would get initialized as follows,

```
vertline_left=0  
vertline_mid=0  
vertline_right=1
```

## 2) HORIZONTAL LINES

The above procedure is repeated in a search for horizontal line of a certain given length. In this case the horizontal histogram values are searched for the desired peak. Next the rectangular boundary is divided into three equal horizontal blocks. The blocks lying towards the top, middle and bottom.

A search is made to see the location of the horizontal line. In this case also three variables are initialized according to the results of the operations. The variables to be initialized are given as under:

```
horzline_top
horzline_mid
horzline_bottom
```

Consider the case of the character . In this case, the horizontal line is lying in the middle block. Hence, the three variables would get initialized as follows,

```
horzline_top=0
horzline_mid=1
horzline_bottom=0
```

### 3) TERMINATING POINTS

A pixel on a segment which has only a single neighbor in an 8-connected neighborhood system.

### 4) JUNCTION POINTS

The pixel about which there exists at least three distinct neighbors is identified as a junction point.

### 5) SEGMENT

A set of pixels bounded at both ends by a Terminating, Junction or Bend points.

The character is parsed as in raster scanning. The parsing begins from the first encountered terminating point, or if no terminating point exists then parse from the left most pixel at the top of the character. The set of pixels is traversed till the next encountered terminating point,

junction point or bend point. These set of pixels are marked using a different symbol, so that an attempt is not made to traverse them again. The set of pixels is identified as one segment. Next, the end point of the previous segment is taken as the start point of the following segment and traversal begins.

#### 6) BEND POINT

This point represents the point of curvature in a given line segment.

#### 7) SIMPLE PIXEL

A pixel which has only two neighbors is called a simple pixel.

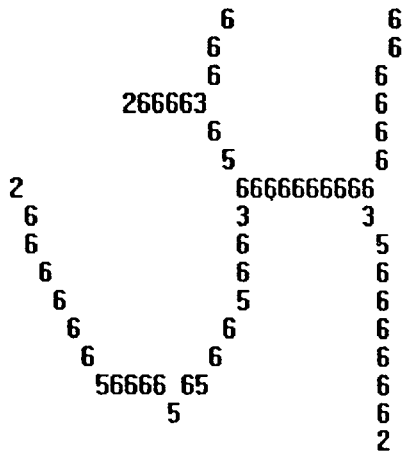
At this stage for the above project, the pixels were renamed so as to easily identify them as junction points, terminating points, bend points. Say,

```
2  -----> Terminating point
3  -----> Junction point
5  -----> Bend point
```

This is as shown in Fig.7.1.

Noise removing algorithms are incorporated at this stage to take care of fictitious junction and bend points. These algorithms have been suggested in Sareen[39].

Features added by us in the following project :-



6: pixel in a segment  
2: terminating point  
3: junction point  
5: bend point

Fig.7.1. Character with features marked.

## 8) SEGMENT LENGTH

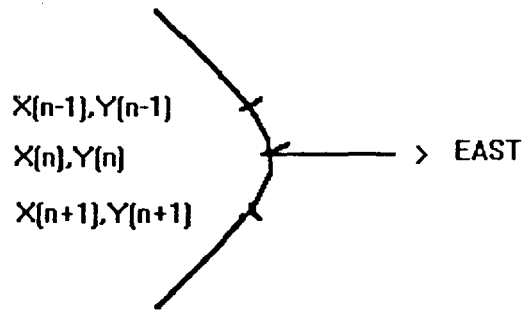
This feature would be used to identify the number of pixels contained in each segments. For this project, this particular feature was used only in case of line segments. This feature helped in distinguishing between vertical lines say as in **अ** or **ब** and the other may be a **ट** thinned down to a **†**. The distinguishing feature was that the line segment in case of the former examples were greater in length by a sizeable margin as compared to the latter. Say, in case of the former, the number of pixels contained in the given vertical line segment was 10 to 11 pixels, while in case of the latter the number of pixels contained was only 5 or 6 pixels.

## 9) MAXIMUM AND MINIMUM ANGLES SUBTENDED

The maximum and minimum angles subtended by the segments of a character on the left bottom corner of the rectangular window.

For each of the segments in the character the angle subtended by the segment is determined. This is as shown in Fig.7.2.

Each of these angles subtended by the different segments in a character is compared to obtain the maximum and minimum angles.



**Fig.7.2** Maximum curvature towards the EAST direction



## 10) DIRECTION OF MAXIMUM CURVATURE

For each of the segments in a character  $\Delta(X)$ ,  $\Delta(y)$ , and  $\Delta(\text{angle})$  are determined. These are obtained as follows: Suppose for a particular segment AB we consider the pixel at position  $X(n)$ ,  $Y(n)$ , then the pixel immediately preceding it is identified as  $X(n-1)$ ,  $Y(n-1)$  and immediately following it is identified as  $X(n+1)$ ,  $Y(n+1)$ .

Taking the three X-Y coordinates i.e.  $X(n-1)$ ,  $Y(n-1)$ ;  $X(n)$ ,  $Y(n)$ ;  $X(n+1)$ ,  $Y(n+1)$  we determine

$$\Delta S = \sqrt{(X_{n+1} - X_{n-1})^2 + (Y_{n+1} - Y_{n-1})^2}$$

$$\Delta X = X_{n+1} - X_{n-1}$$

$$\Delta \theta \approx \frac{\Delta X}{\Delta S}$$

For a given segment all the angles are compared to determine the maximum angle. In case, the maximum angle is associated with  $X(n-1)$ ,  $Y(n-1)$  and  $X(n+1)$ ,  $Y(n+1)$  then  $X(n)$ ,  $Y(n)$  denotes the point of maximum curvature.

Next, we determine the direction to which the pixel identified as the maximum curvature point points to. To obtain the above information we watch the values of

$X(n-1)$ ,  $Y(n-1)$

$X(n)$ ,  $Y(n)$

$X(n+1)$ ,  $Y(n+1)$

We have to identify the direction of maximum curvature as NORTH, SOUTH, EAST, WEST.

Suppose, the values of the above X-Y coordinates are such that,

$$\begin{aligned} & ( X(n) > X(n-1) \quad \text{AND} \quad X(n) < X(n+1) ) \quad \text{AND} \\ & ( ( Y(n) \leq Y(n-1) \quad \text{AND} \quad Y(n) < Y(n+1) ) \quad \text{OR} \\ & \quad ( Y(n) < Y(n-1) \quad \text{AND} \quad Y(n) \leq Y(n+1) ) ) \end{aligned}$$

The above direction is identified as WEST.

Again, consider the values of X-Y coordinates to be such that,

$$\begin{aligned} & ( X(n) > X(n-1) \quad \text{AND} \quad X(n) < X(n+1) ) \quad \text{AND} \\ & ( ( Y(n) \geq Y(n-1) \quad \text{AND} \quad Y(n) > Y(n+1) ) \\ \text{OR} & ( Y(n) > Y(n-1) \quad \text{AND} \quad Y(n) \geq Y(n+1) ) ) \end{aligned}$$

The above direction is identified as EAST. This feature has great significance for Hindi characters where most of the characters have segment points of maximum curvature pointing to a particular direction.

**CHAPTER 8**

**CLASSIFICATION OF CHARACTERS**

## CHAPTER 8

### CLASSIFICATION OF CHARACTERS

The characters of Hindi text are broadly classified into five distinct classes. These are the following:

#### Class I

This class comprises of all the characters having a vertical line in the right zone of length exceeding half the vertical extent and a horizontal middle line of length exceeding one-third the horizontal extent, like अ , स , न , झ , etc.

#### Class II

This class comprises of all characters having only a vertical line in the right zone and no horizontal middle line, like ल , व , व , etc.

#### Class III

This class comprises of all characters having only a horizontal middle line and no vertical right line, like ड , ड , , , etc.

#### Class VI

This class comprises of characters having neither a vertical right line nor a horizontal middle line like र , र , ह , etc.

## Class V

This is a special class of characters comprising of all the characters which consist of two parts. The second part of such characters is always a line segment, like  $\bar{३}$  ,  $\bar{४}$  ,  
,etc.

**CHAPTER 9**  
**DICTIONARY**

# CHAPTER 9

## DICTIONARY

### 9.1 INTRODUCTION

A dictionary was created to store information on the character code and its properties. This information is later referenced during the analysis phase for recognition. Four separate dictionaries are created, two of them are used to store the features of characters and matras extracted without any preprocessing and the other two are used for storing the features extracted after preprocessing.

The information stored in the dictionary can be stored in the form of a sequential list, a tree, a hash table, etc. The dictionary for this project was organized in the form of a hash table. The reason being that in case of a sequential list a search for a record placed at the end of a list consisting of  $n$  records would take computation time of the order of  $O(n)$ . While, in case of a tree though the computation time for a search would be of the order of  $O(\log n)$  depending on the level of the tree, but the need to restructure a tree after deletions and insertions would increase complexity. Hence, the obvious choice was that of a hash table where depending on the value of the key a hash is made to a particular address. The time taken is independent of the number of records.

The address or location of an identifier  $X$ , is obtained by computing some arithmetic function,  $f$ , of  $X$ .  $f(X)$  gives the

address of X in the table. This address will be referred to as the hash or home address of X. The memory available to maintain the symbol table is assumed to be sequential. This memory is referred to as the hash table, ht. The hash table is partitioned into b buckets, ht[0], ..., ht[b-1]. Each bucket is capable of holding s records. Each slot is large enough to hold one record. Usually s=1 and each bucket can hold exactly one record.

An overflow is said to occur when a new identifier I is mapped or hashed by f into a full bucket.

A collision occurs when two nonidentical identifiers are hashed into the same bucket. When the bucket size s is 1, collisions and overflows occur simultaneously, [53].

## 9.2 A DICTIONARY FOR STORING THE FEATURES OF CHARACTERS

The dictionary is organized as a hash table. The hash table is partitioned into 5 buckets, ht[0], ht[1], ..., ht[4]. Each bucket is capable of holding one record. Each record consists of the following five fields,

```
char_code
prop_addr
flag
coll_flag
new_addr
```

1. char\_code: character code

In this field the code of the character is stored, which j is an integer.

2. prop\_addr: property address

This field contains the address of the location in



another file, where the properties of the characters are stored.

3. flag: status flag

The flag field is initialized to zero indicating that the record is empty and when set to one indicates that the record is full.

4. coll\_flag: collision flag

This flag when initialized to zero indicates that no collision has occurred for this particular location. In case this field is set to one then it indicates a collision of nonidentical identifier hashing to the same location.

5. new\_addr: new address for collision bucket

This field is only valid in case of a collision. In case the collision flag is set then this field is referred to, in order to find the address of the pointer pointing to the collision bucket.

**9.2.1 Implementation details**

A raw input file is created, in which the character codes along with their properties are stored.

Next a dictionary described as above is created and initialized to all zeros. In this dictionary the first five locations are buckets and the rest of the file comprises of records to be used in case of a collision.

Another file is created in which the properties of the characters are stored. Each list of properties for a character is enclosed in brackets (.....), and the starting address of the list of properties is stored in the 'prop\_addr' field of the record for the character in the dictionary.

**CHAPTER 10**  
**USE OF HEURISTIC**

# CHAPTER 10

## USE OF HEURISTIC

### 10.1 INTRODUCTION

Certain amount of heuristic is required in the analysis of composite characters like कि, को, के, etc. For instance a line component followed by a character KA( क ) with a I( ँ ) matra on top is identified as KI( कि ).

This kind of analysis is discussed briefly below:-

Note: All possible conditions are not underlined below.

If an upper matra is detected

/\* width & height of the upper matra exceeds a limit \*/

Then

Begin

consider two consecutive characters

analyze matra on top of both characters

If matra analysis succeeds

Then

Begin

analyze both characters separately

If one of the characters is a line component

Then

Begin

If the matra is I ( ँ )

Then

Begin

If the second character is a full form

```

/* say KA(क ) */
Then
  the matra is small_I(ि)
  /* say KI(कि ) */
Else
  If the second character is a half form
    /* say SH(श ) */
  Then
    Begin
      request for the following character
      If the character is a full form or a line
        /* say KA( क ) or | */
      Then
        the composite character is SHKI(शिक )
        or SHI(शि)
      End /* second character half form */
    End
  End
End

```

# **CHAPTER 11**

## **RESULTS**

# CHAPTER 11

## RESULTS

The final results are output in coded English format like KA for क , KI for कि , KO for को, KAU for कौ , # for rejected character etc.

### 11.1 THE WORKING SYSTEM

The main window with caption title 'Hindi OCR' has a pop down 'MAIN' menu. The menu provides options for a file name, display image, analysis and quit.

The file name option when chosen displays a dialogue box, with a request for a file name containing image data to be analyzed. This is shown in Fig.11.1.

The display image option when chosen displays the original image in Bitmap format, as shown in Fig.11.2.

The analysis option is highlighted to begin the recognition process.

The quit option when highlighted terminates the execution of the current program.

After analysis of one line of text, a dialogue box appears with buttons indicating a choice for Results, Not\_Ok, Help, as shown in Fig.11.3.

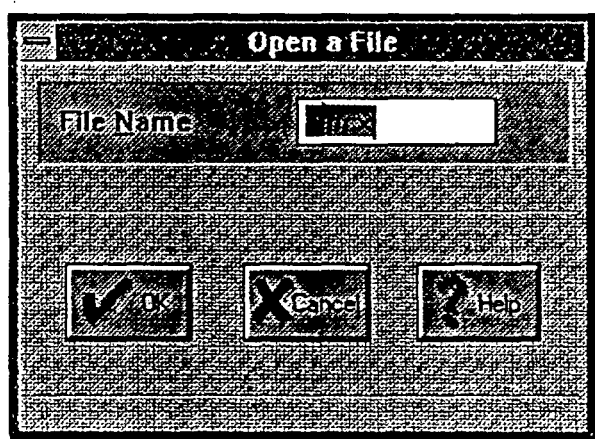


Fig.11.1 A dialog box requesting file name.

INPUT TEXT

छोटी-सी हूँ, लेकिन फिर भी  
बड़े काम की मानी जाती,  
सदा समय की पाबंदी, मैं  
रखना सबको हूँ सिखलाती ।

मी जेब में पड़ी ठुमकती  
मी कलाई पर बँध जाती,  
मी मेज़ पर बैठ ठाठ से —  
रु-टिक-टिक-टिक राग सुनाती ।

Fig.11.2 Original input text.



## OUTPUT TEXT

CHOTI#O HUN           LEKIN   FIR    BHI  
BAARE KAAM KI       MAANI JAATI  
SADA SAMAY KI       PAABNDI       MN  
RAKHANA SABAKO HN   SIKHALAATI  
MI    JEB    ME    PADI   DAMAKATI  
MI    KALAAI   PAR   BADH   JAATI  
MAE   MEJ    PAR   BAID   DAAD   S    #  
RAG#TE#JITAKARAAA   SANAATA

HINDI-OCR

INPUT TEXT

अब पकड़ी, तब पकड़ी तितल  
कभी पास है आती ।  
और कभी पर तेज हिलाकर,  
दूर बहुत उड़ जाती ।

OUTPUT TEXT

AB PAKADIG TAB PAKADI SNTA##  
KAMI PAAS HAI AATI  
AUR KAMI PAR TIJ IAKAR  
DUR BA# UAR JAATA

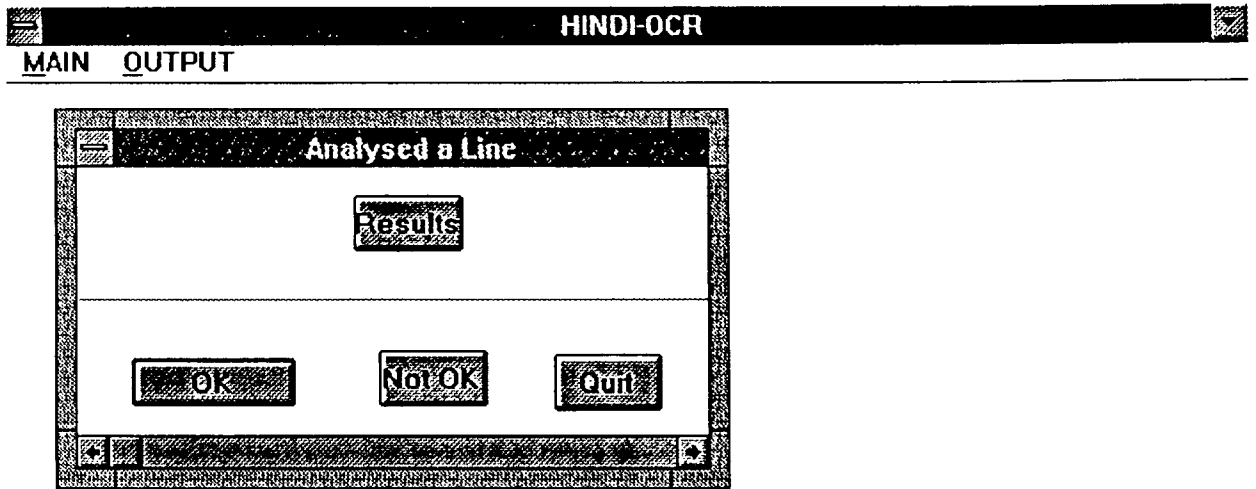


Fig.11.3. Dialog box highlighting completion of analysis of a line.

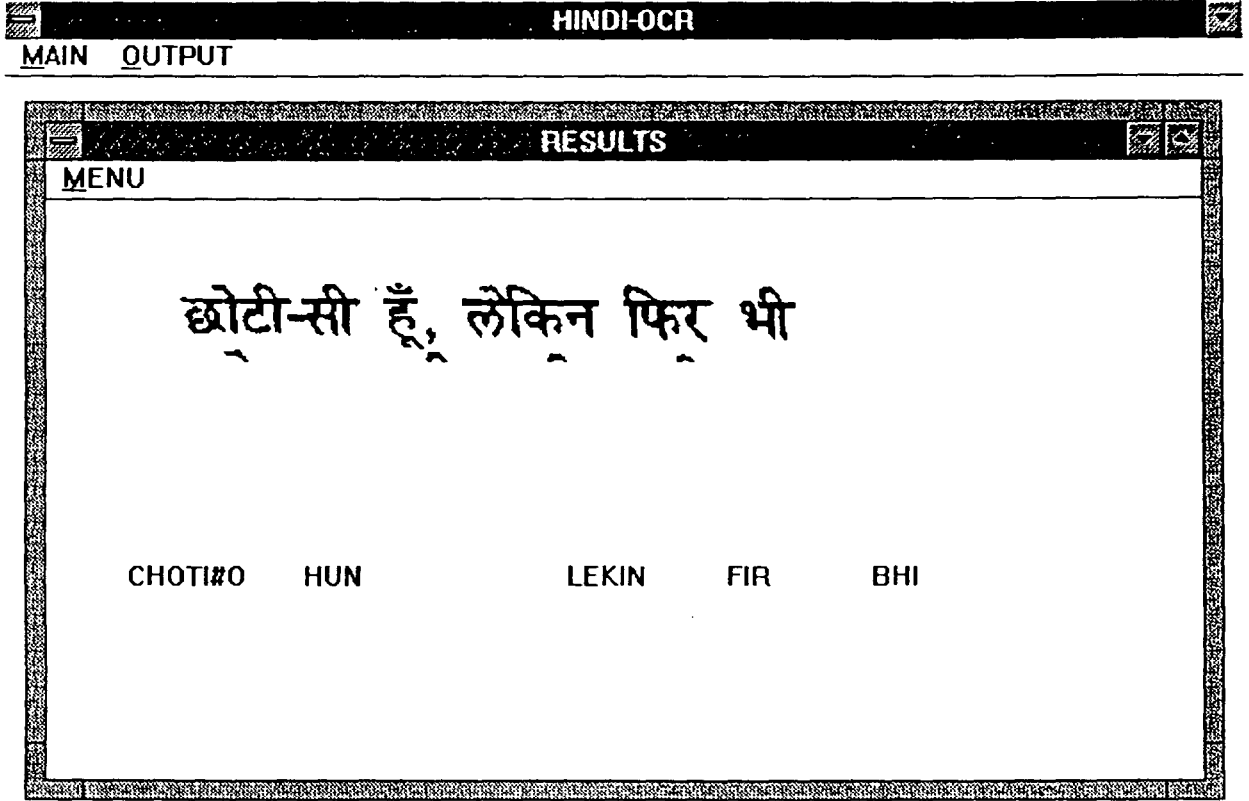


Fig.11.4. An input line and the output recognized text.

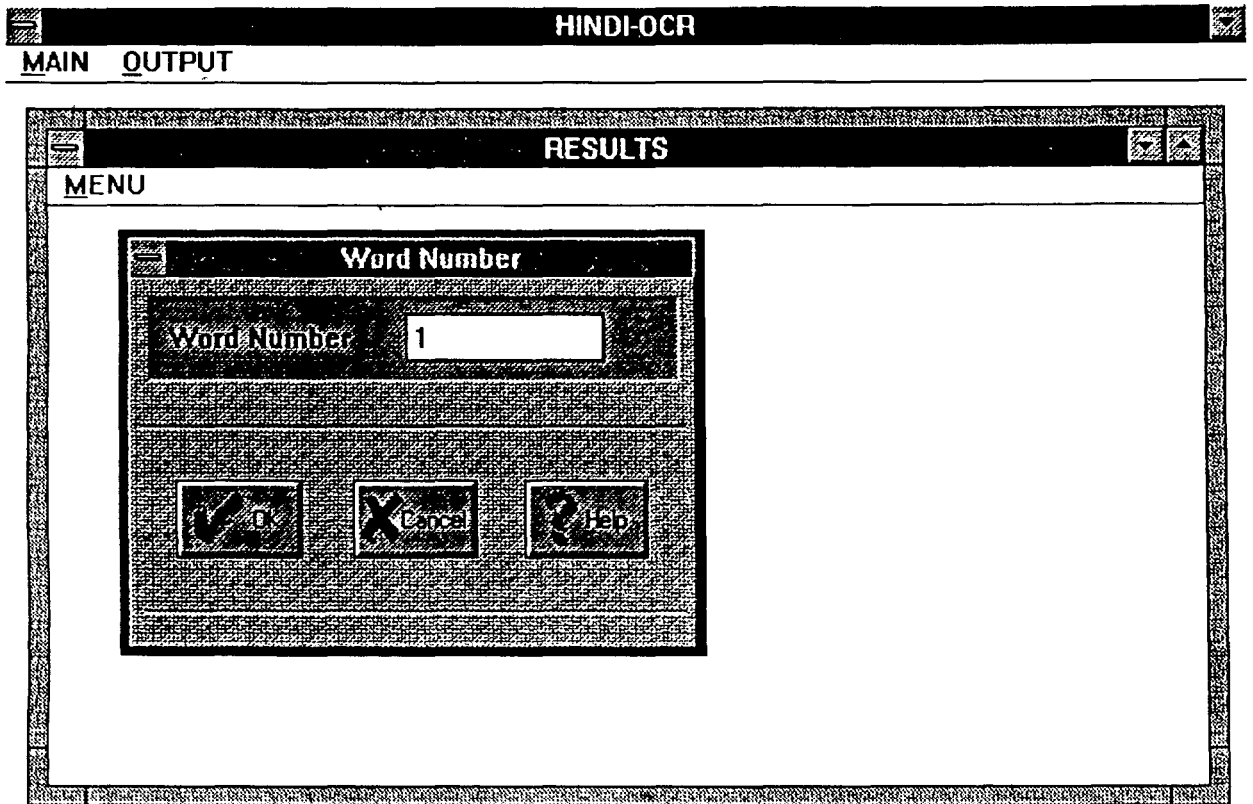


Fig.11.5. . An option to modify.

RESULTS

MENU

New Word

New Word CHOTI-SI

OK Cancel ? Help

ORIGINAL WORD:  
CHOTI#0

RESULTS

MENU

छोटी-सी हूँ, लेकिन फिर भी

CHOTI-SI

HUN

LEKIN

FIR

BHI

On clicking the Results option, another child window appears with caption title 'Analyze a Line' and a pop down menu option. The pop down menu has options for Display\_Results, Modify, Quit. When the option Display\_Results is chosen then original line of text analyzed is displayed along with the coded results, as shown in Fig.11.4. The modify option allows a particular word of the given line of text to be modified, as shown in Fig.11.5. °

On clicking the Ok button, the system begins to analyze the next line of text.

## 11.2 SPECIFICATIONS

The system enables one to scan, read and recognize documents on IBM Personnel Computers and compatibles in the MS Windows environment.

Algorithm	Feature extraction based on contour analysis.
Functions	1. Recognition 2. Editing Results
Fonts	Font independent: attempted with the help of Artificial Neural Networks. Font dependent: user requested to provide the font number.
Recognized characters	Hindi Text
Character size	10x10 to 64x64



Pitch	any
Document types	Monospaced, proportionally spaced and typeset documents.
Quality of printing	Typographical printing, type-setting, laser printing, hand drawn text.
Input image format	Uncompressed or compressed PCX.
Output text format	Plain ASCII
Accuracy rate	At least 85%( in case of good quality printouts )
Recognition speed	cps( on 20 MHz AT/386)
System requirements	IBM AT, AT/386, AT/486 or compatible.
Operating System	MS Windows 3.0 or later
Memory requirements	According to MS Windows requirements.

**CHAPTER 12**  
**TRAINING ALGORITHM**

# CHAPTER 12

## TRAINING ALGORITHM

Two different approaches have been adopted to train the system to recognize text of different fonts. The first approach consists of maintaining different databases for each of the fonts. The user is requested to give a font number and the system switches to the appropriate database. The second approach makes use of artificial neural networks to train the system. This approach is described below:

### 12.1 THE ESSENCE OF ARTIFICIAL NEURAL NETWORKS (ANNs)

An artificial neural network (ANN), also called a "neural net", is computational tool having AI origins. It differs from conventional AI applications and consequently, it deserves separate treatment. Expert systems programmed in LISP and Prolog use "classical" symbolic processing. The programs manipulate symbols, such as atoms and lists, to solve problems. ANNs, on the other hand, use subsymbolic processing.

#### 1. Subsymbolic Processing

The term "artificial neural network" resulted from AI research that attempted to understand and model brain behavior.

In the human brain, neurons within the nervous system interact in a complex fashion. The human senses detect stimuli

and send "input" information (via neurons) to the brain. Within the brain, neurons are excited and interact with each other. Based on the input, a conclusion is drawn, and an "output" is sent from the brain in the form of an answer or response. Neurologists and AI researchers have proposed a highly interconnected network of "neurons", or nosed to develop the same type of structure for a computer modelling of intelligent behavior.

Expert systems operate symbolically, on a macroscopic scale. They use symbolic processing, require knowledge of relationships, and do not care how these relationships develop. ANNs, however, operate subsymbolically on a microscopic scale. The interactions between nodes is well-defined and adjusted until the desired input-output relationships are properly matched.

The interconnection of nodes form the artificial neural network (ANN). All ANNs have an input layer, one or more hidden layers, and an output layer. An ANN can be viewed as a "black box" into which we send a specific input to all the nodes in the input layer. The ANN processes this information through its interconnections between nodes (the entire processing step is hidden from us). Finally, the ANN gives us a final output, which results from the nodes on the output layer.

Input Layer - receives information from an external source, and passes this information into the ANN for processing.

Hidden Layer - receives information from the input layer, and "quietly" does all of the information processing. The entire

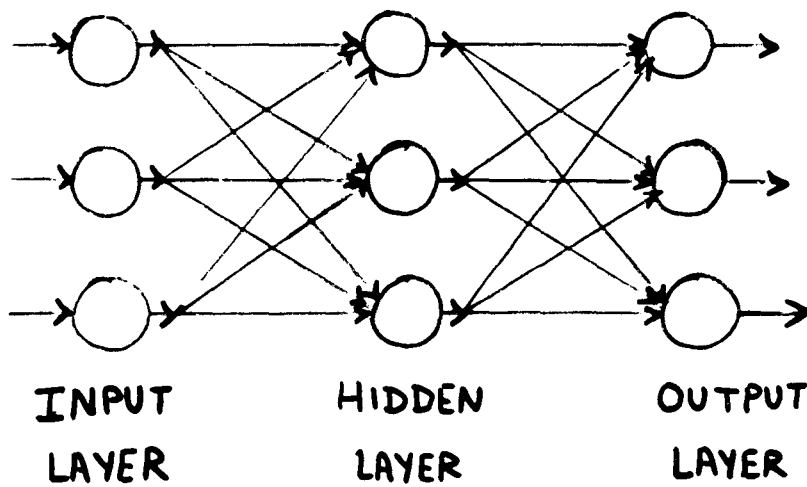


Fig.12.1 An ANN with one hidden layer

processing step is hidden from view.

Output Layer - receives processed information from the ANN, and sends the results out to an external receptor.

## **12.2 OPERATING AN ANN**

To operate an ANN, we require the following three phases: the training or learning phase, the recall phase, and the generalization phase.

In the training or learning phase, we repeatedly present a set of input-output patterns to the ANN. We adjust the weights of all the interconnections between nodes until the specified input yields the desired output. Through these activities, the ANN "learns" the correct input-output response behavior.

After the training phase, we move to the recall and generalization phases. In ANN development, the training phase is typically the longest and most time-consuming step. In the recall phase, we subject the ANN to a wide array of input patterns seen in training, and introduce adjustment to make the system more reliable and robust. During the generalization phase, we subject the ANN to novel input patterns, where the system hopefully performs properly.

## **12.3 PROPERTIES OF ANNs**

ANNs have a number of properties that make them advantageous over other computational techniques, as described

below.

(1) Information is distributed over a field of nodes. This provides greater flexibility than symbolic processing, where information is held in one fixed location.

(2) ANNs have the ability to learn. If an error or a novel situation occurs that creates inaccurate system results, we can use "backpropagation" to correct it. During backpropagation, we adjust the strengths of the signals emitted from the nodes until the error disappears. At that point, the system has effectively "learned". When the system encounters that situation in the future, the ANN will model it properly.

(3) ANNs allow extensive knowledge indexing. Knowledge indexing is the ability to store a large amount of information and access it in a simple manner. An ANN provides inherent knowledge indexing. It can recall, for example, diverse amounts of information associated with a chemical name, a process, or a set of process conditions. The knowledge is retained in the network via two means: 1) the connections between nodes, and 2) the weights of these connections.

4) ANNs are better suited for processing noisy, incomplete, or inconsistent data. No single node within an ANN is directly responsible for associating a certain input with a certain output. Instead, each node encodes a microfeature of the input-output pattern. The concept of microfeature implies that each node affects the input-output pattern only slightly. Only when we assemble all the nodes together into a single coordinated network, can these microfeatures map the

macroscopic input-output pattern. In addition to the microfeature concept for ANNs, the signals sent to and from nodes are continuous functions. Consequently, the ANN can deduce proper conclusions, even from noisy, incomplete, or inconsistent input signals.

(5) ANNs mimic human learning processes. Most human learning and problem-solving occurs by trial and error. ANNs operate in the same fashion. We can train them by iteratively adjusting the strength of the connections between the nodes. After numerous iterative adjustments, the ANN can properly predict cause-and-effect relationships.

## **12.4 FUNDAMENTALS OF NEURAL COMPUTING**

### **12.4.1 Components of a Node**

The foundation of an ANN is the artificial neuron, or node (sometimes called neurode). In most scientific and engineering applications, this node is called a processing element (PE).

The PEs are the elements in the ANNs where most calculations are performed.

#### **1. Inputs and Outputs**

The first element in the  $j$ th PE is an input vector,  $a$  with components  $a_1, a_2, a_3, \dots, a_i, \dots, a_n$ . The node manipulates these inputs, or activities to give the output  $b_j$ . This output can then form the part of the input for other PEs.

#### **2. Weight Factors**

The PE uses weighted input to determine the output from the PE.



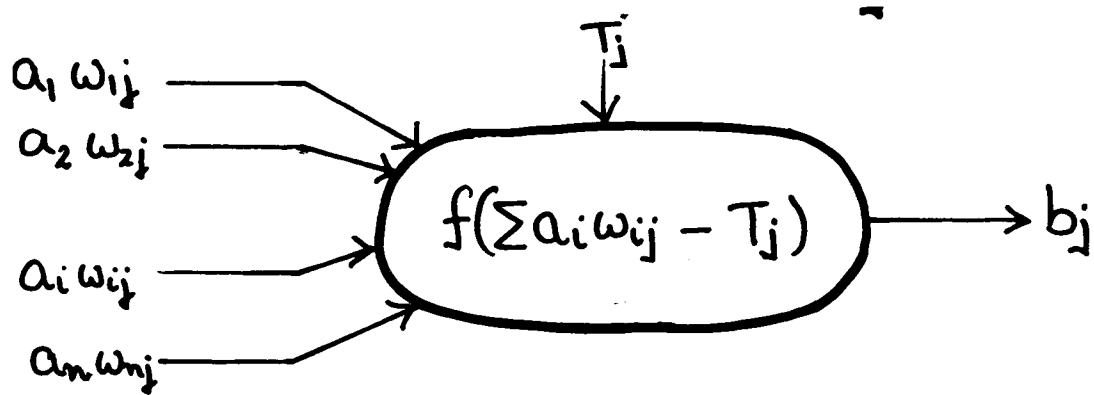


Fig. 12.2. The anatomy of the  $j$ th PE.

### 3. Internal Thresholds

The internal threshold for the  $j$ th PE, denoted  $T_j$ , controls activation of the node. The node calculates all its  $a_i w_{ij}$ 's, sums the terms together, and then calculates the total activation by subtracting the internal threshold value:

$$\text{TotalActivation} = \sum_{i=1}^n (w_{ij} a_i) - T_j$$

### 4. Functional Forms

The PE performs calculations based on its input. It takes the dot product of vector  $a$  with vector  $W_j$ , subtracts the threshold  $T_j$ , and passes this result to a functional form  $f()$ .

Mathematicians and computer scientists have found that the sigmoid (S-shaped) function is particularly advantageous. A typical sigmoid function is:

$$f(x) = \frac{1}{1 + e^{-x}}$$

This function is monotonically increasing, with limiting values of

$$0 \text{ (at } x = -\infty) \wedge$$

$$1 \text{ (at } x = +\infty)$$

Because of these limiting values, sigmoid functions are called threshold functions. At very low input values, the threshold-function output is zero. At very high values, the output value is one.

#### 12.4.2 Topology of an Artificial Neural Network

The topology of an ANN refers to how its PEs are interconnected.

##### 1. Inhibitory or Excitatory Connections

Connections can either inhibit or excite the node. If the weight is positive, it will excite the node, increasing the activation of the PE. If the signal is highly inhibitory, it may lower the input below the threshold level and shut the node down.

##### 2. Connection Options

There exist three connection options.

- a) Intralayer connections,
- b) Interlayer connections, and
- c) Recurrent connections.

In intralayer connections, the outputs from a node feed into other nodes in the same layer.

In interlayer connections, the outputs from a node in one layer feed into nodes in another layer.

In recurrent connections, the output from a node feeds into itself.

The type of problem we are trying to solve determines which topology we favor. For example, if we wish to develop an ANN that trains itself, we may use feedback connections. In contrast, in dynamic modelling of a chemical reactor, we are trying to map an output response based on an input signal, therefore, we favor the feedforward connection.

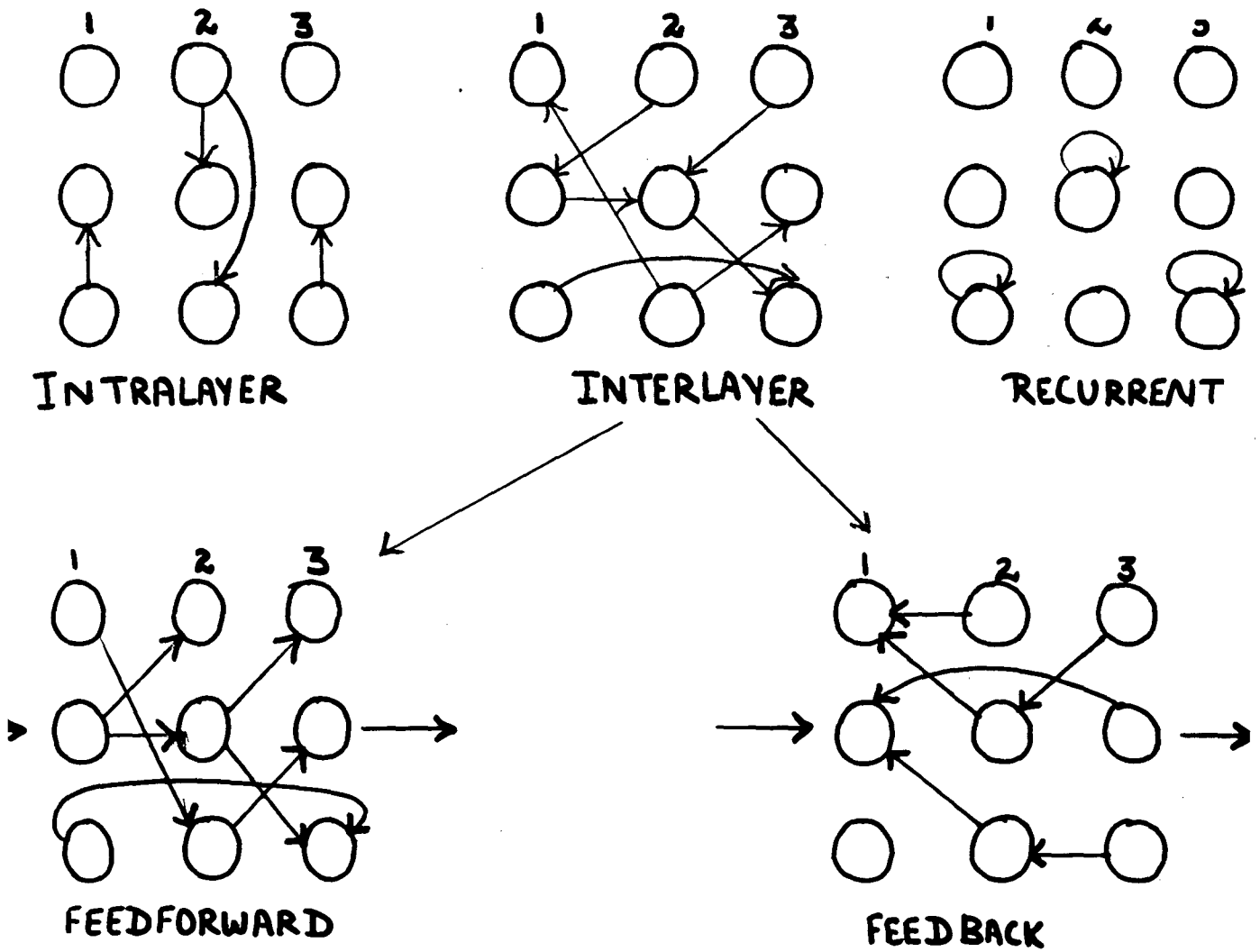


Fig.12.3. Connection options in an ANN..

## 12.5 LEARNING AND TRAINING WITH ARTIFICIAL NEURAL NETWORKS

To train ANNs, we adjust the weight factors until the calculated output pattern (response) based on the given input matches the desired cause-and-effect relations. Learning is the actual process of adjusting weight factors based on trial-and-error.

### 1. Stability and Convergence

The training phase needs to produce an ANN that is both stable and convergent. A globally stable ANN maps any set of inputs to a fixed output. Stability guarantees a result, but it does not necessarily guarantee an accurate result.

A convergent ANN produces accurate input-output relations., Convergence, then is related to the accuracy of the ANN. The magnitude of error between real-world results and those predicted by the ANN is a direct measurement of ANN is convergence.

### 2. Types of learning

There are many different approaches to training ANNs. Most approaches fall into one of the two groups:

- a) Supervised learning - an external teacher controls the learning and incorporates global information.
- b) Unsupervised learning - no external teacher is used and the ANN relies upon both internal control and local information. Frequently, the ANN develops its own model

without additional input information.

### 12.5.1 Learning Procedure

Error-Correction Learning - the most common type of learning used in ANN today. It is a form of supervised learning, where we adjust weights in proportion to the output error vector. This output error vector has  $n$  components, where  $n$  is the number of nodes in the output layer.

We begin error-correction learning by defining the output error from a single node on the output layer as:

$$e_n = d_n - b_n$$

where,

$e_n$  is the output error,

$d_n$  is the desired output,

$b_n$  is the calculated output,

for the  $n$ th node in the output layer only.

We then calculate the total squared error of the output layer,  $E$ , as:

$$E = \sum_n e_n^2 = \sum_n (d_n - b_n)^2$$

Knowing  $E$ , we can calculate the change in the weight factor for the  $i$ th to the  $j$ th node,

$$\Delta w_{ij} = \beta_j a_i E$$

$\beta_j$  is a linear proportionality constant for node  $j$  (typically,  $0 < \beta_j < 1$ )

$a_i$  is the  $i$ th input

## 12.5.2 Backpropagation Learning: Vanilla Backpropagation Algorithm

### 1. Requirements for Backpropagation Learning

Backpropagation requires an ANN known as a perceptron. A perceptron may be defined as an ANN with only feedforward interlayer connections, and no intralayer or recurrent connections. Each layer must feed sequentially into the next layer, with no feedback connections. In the following section we investigate a three-layer, sequential perceptron.

The perceptron ANN has three layers, A, B and C. Feeding into layer A is the input vector  $I(L)$ . Thus layer A has  $L$  nodes, that is,  $a(1), a(2), \dots, a(i), \dots, a(L)$ . Layer B, the hidden layer, has  $m$  nodes:  $b(1), b(2), \dots, b(j), \dots, b(m)$ . In the drawing,  $L=m=3$ , but in practice,  $L \ll m$  is acceptable. Layer C, the output layer, is next. There are  $n$  C-layer nodes, again,  $L \ll m \ll n$  is acceptable. The interconnection weight between the  $i$ -th node of layer A and the  $j$ -th node of layer B is denoted as  $v(i)(j)$ , and that between the  $i$ -th node of layer B and the  $j$ -th node of layer C are  $w(i)(j)$ . Each node has an internal threshold value. For layer A, the threshold is  $T(A_i)$ , for layer B,  $T(B_j)$ , and for layer C,  $T(C_k)$ .

### THE VANILLA BACKPROPAGATION TECHNIQUE

Backpropagation learning attempts to properly map given inputs with desired outputs by minimizing an error function. Typically, we use the sum-of-squares error.

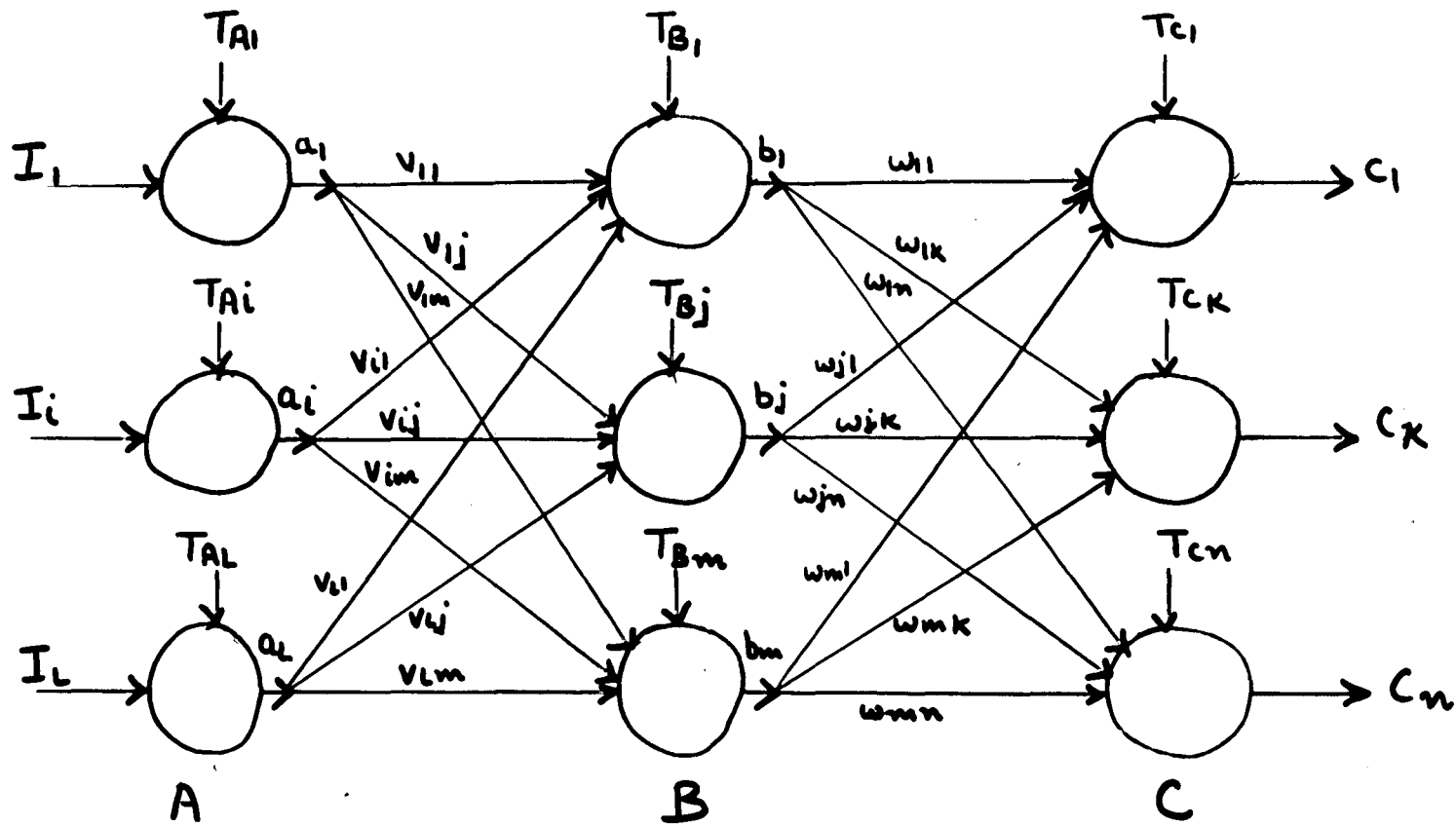


Fig. 12.4. A three-level perceptron ANN..



Below is the step-by-step adjustment procedure known as the vanilla backpropagation algorithm (Simpson,1990).

Step 1:

Randomly specify numerical values for all weight factors (  $v(i)(j)$ 's and  $w(j)(k)$ 's ) within the interval  $[-1,+1]$ . Likewise, assign internal threshold values (  $T(A_i)$ ,  $T(B_j)$ ,  $T(C_k)$  ) for every node, also between  $+1$  and  $-1$ . Note that  $i=1,2,\dots,L$ , where  $L$  is the number of nodes in layer A;  $j=1,2,\dots,m$ , where  $m$  is the number of nodes in layer B; and  $k=1,2,\dots,n$ , where  $n$  is the number of nodes in layer C.

Step 2:

Introduce the input  $I(i)$  into the ANN. Calculate all outputs from the first layer, using the standard sigmoid function introduced previously:

$$x_i = I_i - T_{Ai}$$

$$a_i = f(x_i) = \frac{1}{1 + e^{-x_i}}$$

Here,  $I(i)$  is the input into the  $i$ -th node on the input layer,  $T(A_i)$  is internal threshold for the node and  $a(i)$  is the output from the node.

Step 3:

Given the output from layer A, calculate the output from layer B, using the equation:

$$b_j = f\left(\sum_{i=1}^L (v_{ij} a_i) - T_{Bj}\right)$$

where  $f()$  is the same sigmoid function.

Step 4:

Given the output from layer B, calculate from layer C, using the equation:

$$c_k = f\left(\sum_{j=1}^m (w_{jk} b_j) - T_{ck}\right)$$

where  $f()$  is the same sigmoid function.

Step 5:

Now backpropagate through the network, starting at the output and moving backward toward the input. Calculate the  $k$ -th component of the output error, for each node in layer C, according to the equation:

$$e_k = c_k(1 - c_k)(d_k - c_k)$$

where  $d_k$  is the desired result and  $c_k$  is the actual result.

Step 6:

Continue backpropagation, moving to layer B. Calculate the  $j$ -th component of the error vector, of layer B, using the equation:

$$e_j = b_j(1 - b_j)\left(\sum_{k=1}^n (w_{jk} e_k)\right)$$

Step 7:

Adjust weights, calculating the new  $w(j)(k)$  as:

$$w_{jk, new} = w_{jk, old} + \beta e_j b_k$$

for  $j=1$  to  $m$  and  $k=1$  to  $n$ .

The term  $\beta$  is a positive constant controlling the learning rate  $\epsilon$  in layer

Step 8:

Adjust the thresholds  $T(C_k)$  in layer C, according to the equation:

$$T_{Ck,new} = T_{Ck,old} + \beta_c e_k$$

Step 9:

Adjust weights  $v(i)(j)$ , according to the equation:

$$v_{ij,new} = v_{ij,old} + \beta_B a_i e_j$$

for  $i=1$  to  $L$  and  $j=1$  to  $m$ .

*The term  $\beta_B$  is a positive constant controlling the learning rate of layer B.*

Step 10:

Adjust the thresholds  $T(B_j)$  ( $j=1$  to  $m$ ) in layer B, according to the equation:

$$T_{Bj,new} = T_{Bj,old} + \beta_B e_j$$

Step 11:

Repeat steps 2-10 until the squared error,  $E$ , or the output error vector is zero or sufficiently small.

The vanilla backpropagation algorithm is a gradient-descent learning technique. We use Newton's method ( moving down a gradient on a surface ) to minimize the error. The advantage of this method is that weight changes are estimated systematically rather than arbitrarily.

**CHAPTER 13**  
**CONCLUSION**

## CHAPTER 13

### CONCLUSION

The main aim was to recognize printed and hand drawn Hindi text. The data was scanned with the help of a hand scanner. The input text was taken from class II and class VII N.C.E.R.T. Hindi books and some hand drawn characters. First the image processing techniques were applied for segmentation of the text. Next the individual characters were subjected to smoothing and thinning. The preprocessed characters were subjected to feature extraction and these features were compared to the features stored in the database for individual characters.

A confidence level of above 75% was required to declare the character to be the same with which it was being compared. Certain amount of heuristics was used to form composite characters and words. A recognition rate of 85% and above has been obtained.

The above system has not used any contextual information to classify the characters.

The current system can be expanded to include contextual information to recognize characters. This would enable enhancement of the rate of recognition further.

## **REFERENCES**

## REFERENCES

- [1] Amlan Kundu, "**Robust Edge Detection**", *Pattern Recognition*, vol.23, No.5, pp. 423-440, 1990.
- [2] C.C Tappert, C.Y. Suen, T.Wakahara, "**The state of the Art in On-Line Handwriting Recognition**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.12, No.8, Aug 1990.
- [3] Frank Yeon-Chyang Shih and Owen Robert Mitchell, "**Decomposition of Gray-Scale Morphological Structuring Elements**", *Pattern Recognition*, vol.24, No.3, pp 195-203, 1991.
- [4] F. Kimura and M.Shridhar, "**Handwritten Numerical Recognition Based on Multiple Algorithms**", *Pattern Recognition*, vol.24, No.10, pp 969-983, 1991.
- [5] G. Baptista and K.M. Kulkarni, "**A High Accuracy Algorithm for Recognition of Handwritten Numerals**", *Pattern Recognition*, vol.21, No.4, pp 287-291, 1988.
- [6] Guangzheng Yang, "**On the knowledge-based Pattern Recognition using Syntactic Approach**", *Pattern Recognition*, vol.24, No.3, pp 185-193, 1991.
- [7] G.P Albrecht, Y. Le Cun, J. Denker and W. Hubbard, "**Design of a Neural Network Character Recognizer for a Touch Terminal**", *Pattern Recognition*, vol.24, No.2, pp 105-119, 1991.

- [8] H.J.A.M. Heijmans and A. Toet, "Morphological Sampling", *CVGIP: Image Understanding*, vol.54, No.3, November, pp 384-400, 1991.
- [9] H. Lynn Beus and Steven S.H. Tiu, "An Improved Corner Detection Algorithm Based on Chain-Coded Plane Curves", *Pattern Recognition*, vol.20, No.3, pp 291-296, 1987.
- [10] I. Sekita, K. Toraichi, R. Mori, K. Yamamoto and H. Yamada, "Feature Extraction of Handwritten Japanese Characters by Spline Functions for Relaxation Matching", *Pattern Recognition*, vol.21, No.1, pp 9-17, 1988.
- [11] James L. Conger, *Windows API Bible*, Waite Group Press.
- [12] J.F. Haddon and J.F. Boyce, "Co-occurrence matrices for image analysis", *Electronic & Communication Engg. Journal*, April 1993.
- [13] Jean Serra, "Introduction to Mathematical Morphology", *Comput. Vision Graphics Image Process.*, 35, 283-305 (1986).
- [14] Jean-Jules Brault and Rejean Plamondon, "Segmenting Handwritten Signatures at their Perceptually Important Points", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.9, Sept 1993.
- [15] J. Ohya, A. Shio, and S. Akamatsu, "Recognizing Characters in Scene Images", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.16, No.2, Feb 1994.



- [16] J. Song and E.J. Delp, "The Analysis of Morphological Filters with Multiple Structuring Elements", *Comput. Vision Graphics Image Process.*, 50, 308-328 (1990).
- [17] J.W. Roach and J.E. Tatem, "Using Domain Knowledge in Low-Level Visual Processing to Interpret Handwritten Music: An Experiment", *Pattern Recognition*, vol.21, No.1, pp 33-44, 1988.
- [18] J. Wu and C. Chan, "Isolated Word Recognition by Neural Network Models with Cross-Correlation Coefficients for Speech Dynamics", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.11, Nov. 1993.
- [19] K.K. Bharadwaj and N.K. Jain, "Hierarchical Censored Production Rules (HCPRs) System", *Data & Knowledge Engg.* 8 (1992), 19-34.
- [20] K. Paler, J. Foglein, J. Illingworth and J. Kittler, "Local Ordered Grey Levels as an Aid to Corner Detection", *Pattern Recognition*, vol.17, No.5, pp 535-543, 1984.
- [21] Lawrence O'Gorman, "The Document Spectrum for Page Layout Analysis", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.11, Nov 1993.
- [22] Li Wang and Theo Pavlidis, "Direct Gray-Scale Extraction of Features for Character Recognition", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.10, No.10, Oct 1993.
- [23] M. Cheriet and C.Y. Suen, "Extraction of Key Letters for Cursive Script Recognition", *Pattern Recognition Letters* 14 (1993).

[24] Mei-Hsing Chen and R.T. Chin, "**Partial Smoothing Splines for Noisy Boundaries with Corners**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.11, Nov 1993.

[25] M. Shridhar and A.Badreldin, "**High Accuracy Character Recognition Algorithm using Fourier and Topological Descriptors**", *Pattern Recognition*, vol.17, No.5, pp 515-524, 1984.

[26] Paul D. Gader, "**Separable Decompositions and Approximations of Greyscale Morphological Templates**", *CVGIP: Image Understanding*, vol.53, No.3, May, pp 288-296, 1991.

[27] P. Saint-Marc, H. Rom, and G. Medioni, "**B-Spline Contour Representation and Symmetry Detection**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.11, Nov 1993.

[28] R.C. Gonzalez and P. Wintz, *Digital Image Processing*, Addison-Wesley Publishing Company.

[29] R.W. Hall, "**Optimally Small Operator supports for Fully Parallel Thinning Algorithms**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.8, Aug 1993.

[30] R.M. Haralick and J.S.J. Lee, "**Context Dependent Edge Detection and Evaluation**", *Pattern Recognition*, vol.23, No.1/2, pp 1-19, 1990.

[31] R.M. Brown, T.H. Fay and C.L. Walker, "**Handprinted Symbol Recognition System**", *Pattern Recognition*, vol.21, No.2, pp 91-118, 1988.

- [32] R.M.K. Sinha and H.N. Mahabala, **"Machine Recognition of Devanagari Script"**, *IEEE trans. on Sys. Man & Cybernetics*, vol. SMC-9, No.8, Aug 1979.
- [33] R.M.K. Sinha, B. Prasada, G.F. Houle, and M. Sabourin, **"Hybrid Contextual Text Recognition with String Matching"**, *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.9, Sept 1993.
- [34] R.M.K. Sinha and B. Prasada, **"Visual Text Recognition through Contextual Processing"**, *Pattern Recognition*, vol.21, No.5, pp. 463-479, 1988.
- [35] Rosenfeld & Kak, **"Digital Image Processing"**.
- [36] R.P. Johnson, **"Contrast Based Edge Detection"**, *Pattern Recognition*, vol.23, No.3/4, pp. 311-318, 1990.
- [37] S. Hazout and N.Q. Nguyen, **"Image Analysis by Morphological Automata"**, *Pattern Recognition*, vol.24, No.5, pp. 401-408, 1991.
- [38] Stanley R. Sternberg, **"Grayscale Morphology"**, *Comput. Vision Graphics Image Process*, 35, 333-355 (1986).
- [39] Shyam S. Sareen, **"Recognition of Handwritten and Printed Numerals"**, M.Tech Dissertation, I.I.T. Delhi.
- [40] Satoshi Suzuki and Keiichi Abe, **"Binary Thinning by Iterative Parallel Two-Subcycle Operation"**, *Pattern Recognition*, vol.20, No.3, pp. 297-307, 1987.

[41] Song-Tyang Liu and Wen-Hsiang Tsai, "**Moment - Preserving Corner Detection**", *Pattern Recognition*, vol.23, No.5, pp. 441-460, 1990.

[42] Si Wei Lu, Ying Ren, and Ching Y. Suen, "**Hierarchical Attributed Graph Representation and Recognition of Handwritten Chinese Characters**", *Pattern Recognition*, vol.24, No.7, pp. 617-632, 1991.

[43] Steve Rimmer, *Bit-Mapped Graphics*, Windcrest Books.

[44] Tin Kam Ho, J.J. Hull and S.N. Srihari, "**Decision Combination in Multiple Classifier Systems**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.16, No.1, Jan 1994.

[45] T. Taxt, J.B. Olafsdottir, and M. Daehlem, "**Recognition of Handwritten Symbols**", *Pattern Recognition*, vol.23, No.11, pp. 1155-1166, 1990.

[46] T.Y. Zhang and C.Y. Suen, "**A Fast Parallel Algorithm for Thinning Digital Patterns**", *Communications of the ACM*, March 1984, vol.27, No.3.

[47] W.A.C. Schmidt, and J.P. davis, "**Pattern Recognition Properties of various Feature Spaces for Higher Order Neural Networks**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.8, Aug 1993.

[48] X. Huang, J. Gu, and Y. Wu, "**A Constrained Approach to Multifont Chinese Character Recognition**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.8, Aug 1993.

[49] X. Zhuang and R.M. Haralick, "**Morphological Structuring Element Decomposition**", *Comput. Vision Graphics Image Process*, 35, 370-382 (1986).

[50] Yih-Tay Tsay and Wen-Hsiang Tsai, "**Attributed String Matching by Split-and-Merge for On-Line Chinese Character Recognition**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.2, Feb 1993.

[51] Z. Wu and R. Leahy, "**An Optimal Graph Theoretic Approach to Data Clustering: Theory and its Application to Image Segmentation**", *IEEE Trans. Patt. Anal. Machine Intell.*, vol.15, No.11, Nov 1993.

[52] A Basic Grammar of Modern Hindi, Central Hindi Directorate, Ministry of Education & Culture, Govt. of India, N.D.