# Cleaning and Recognition of Numerals in a Handwritten Devnagari Document Consisting of Roll Numbers and Marks

*Thesis submitted in partial fulfillment of the requirements for the award of degree of*

**Master of Technology**
In
**Computer Science and Applications**

*Submitted By*
**Kanika Bansal**
**(Roll No. 601003011)**

Under the supervision of
**Dr. Rajiv Kumar**
Assistant Professor

SCHOOL OF MATHEMATICS AND COMPUTER APPLICATIONS
THAPAR UNIVERSITY
PATIALA – 147004

**June 2012**

# CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "*Cleaning and Recognition of Numerals in a Handwritten Devnagari Document Consisting of Roll Numbers and Marks*", in partial fulfillment of the requirements for the award of degree of Master of Technology in *Computer Science and Applications* submitted in *School of Mathematics and Computer Applications* Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Rajiv Kumar* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature:

(Kanika Bansal)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Rajiv Kumar)
Assistant Professor,
SMCA

**Countersigned by**

**(Dr. S. S. Bhatia)**
Professor & Head
School of Mathematics and Computer Applications
Thapar University
Patiala

**(Dr. S. K. Mohapatra)**
Dean (Academic Affairs)
Thapar University
Patiala



i

# ACKNOWLEDGEMENT

First of all, I would like to express my gratitude to **Dr. Rajiv Kumar, Assistant Professor,** School of Mathematics and Computer Applications, Thapar University, Patiala for introducing me to document image processing and for all his guidance and support. This thesis work was enabled and sustained by his vision and ideas. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own and at the same time the guidance to recover when my steps faltered. His patience and support helped me overcome many crisis situations and finish this dissertation.

I am also thankful to our **Head of the Department, Dr. S. S. Bhatia** as well as **PG Coordinator, Mr. Singara Singh, Assistant Professor,** School of Mathematics and Computer Applications, entire faculty and staff of School of Mathematics and Computer Applications and then friends who devoted their valuable time and helped me in all possible ways towards successful completion of this work. I thank all those who have contributed directly or indirectly to this work.

Lastly, I would also like to thank my parents for their years of unyielding love and encourage. They have always wanted the best for me and I admire their determination and sacrifice.

**Kanika Bansal**
**(601003011)**

# ABSTRACT

The field of character recognition has been fascinating researchers over a last few decades. It is easy for the human mind to decipher the handwritten characters with accuracy, but for the machines, it is a difficult task. The Optical Character Recognition (OCR) systems have been developed to imitate the human action of interpreting the handwritten and printed characters. Innumerable approaches have been developed for the character recognition process but it still remains a challenging field for the researchers. The most of the approaches have used the neural networks as their basic technique for the recognition process. These approaches require huge computations and are training dependent. Thus, an effort has been made by the authors to develop a simple approach for the offline handwritten character recognition.

The proposed approach initiates with the preprocessing of the image through various techniques like filtering, binarization, thinning and smoothing. For the purpose of filtering, modified median filter has been proposed, which is based on the conditional application of the median filter. The cleaned up image is then, decomposed into lines, words and characters by using the line segmentation, word segmentation and character segmentation techniques respectively. The segmented characters are analyzed to check the presence of the features like sidebars, closed loops and water reservoirs. Based on these features, a classification structure has been developed, which assigns the segmented character to the predefined class. The prime focus here is on the recognition of the offline handwritten Devnagari numerals, which can aid in several applications like reading commercial forms, postcode recognition, passport readers, bank cheques, bill processing systems and many more. The proposed approach has been implemented. It was applied to several documents and satisfying results have been obtained by the authors.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

For the human kind, there has always existed a need to read and interpret handwriting. According to Plamondon and Srihari [2000], the term "handwriting" is defined as meaning a surface consisting of artificial graphic marks conveying some message through the mark's conventional relation to language. The concept of handwriting has existed for millennia, with the sole aim of expanding human memory and facilitating communication.

To identify the first attempts at building a machine for recognising printed matter, it is necessary to return to the beginning of OCR (Optical Character Recognition) research. The introduction to OCR research is said to have started with an objective to develop reading machines for the blind. Steven [1961] stated that the original research attempted to develop devices capable of duplicating and transcribing various types of matter including printed, typed or handwritten characters. The initial attempts to this research may be traced back to 1870, however, according to Mantas[1981], the first successful attempts to develop a device to aid the blind was made in 1900 by Tyurin and another by Fournier d' Albe who presented his optophone in 1912. The concept of OCR came soon after and was originally restricted to recognizing printed characters. Mori *et al*. [1992] stated that the first patent in this area was obtained in 1929 by Tausheck in Germany, followed by Handel in 1933. To the best of this author's knowledge, these are the first concepts of the idea of OCR. However, with the advent of computing technology, this state of affairs has changed.

The next section discusses some of the popular OCR's to get a view of the current state of art in this field.

## 1.1 Historical Background of OCR

The David Shepard's invention, which was GISMO - A Robot Reader and Writer, can be said to be the first attempts for the modern version of the OCR. According to Srihari and Lam [1995], this invention was followed by Jacob Rabinow's prototype machine in 1954, which was able to read upper case and printed output. David

Shepard's company Intelligent Machines Research (IMR) is said to have been the first to apply optical reading techniques in a commercial situation. They installed a system at Reader's Digest in 1955, as stated by Stevens [1961]. Early systems were very constrained in the sense that they were bound to reading special, artificial fonts, which can be said to be associated with the first generation of OCR. According to Mori *et al*. [1992], the second generation was characterised by the recognition of the hand printed characters. In the initial stages, only numerals could be recognised by these intelligent machines and IBM's 1287 OCR system was the first such system of the second generation machine.

It has been approximately 47 years since the first automatic handwriting readers were proposed. Since then, astounding progress has been made to enable computers to recognise, interpret and identify machine printed and handwritten script. Extensive research has been carried but the research into handwriting recognition still continues to be intense at the current stage. The motivation may be accredited to the challenging nature of the character recognition problem and the innumerable commercial applications it has. The next section throws some light on the character recognition process.

## 1.2 Character Recognition

Character recognition is used as an umbrella term, which covers all types of machine recognition of characters in various application domains.It is processing by machine of text based input patterns to produce meaningful outputs. The input may come from online devices like tablets, stylus based devices or offline devices like scanners. Output may be a sequence of symbols like 'Y', 'E' ,'S' or a date on cheque like 'Nov 14, 2011' or validation result of a signature. As the input may come from different sources, they may be categorized broadly as offline and online and are explained in the next section.

## 1.3 Types of Handwriting Input: Offline versus Online

Character recognition is a process, which associates a symbolic meaning with objects such as letters, symbols and numbers drawn on an image, that is, character recognition techniques associate a symbolic identity with the image of a character. Mainly, character recognition machine takes the raw data that further implements the

technique of processing of any recognition system. Based on that data acquisition process, character recognition system can be classified into following categories as shown here:-

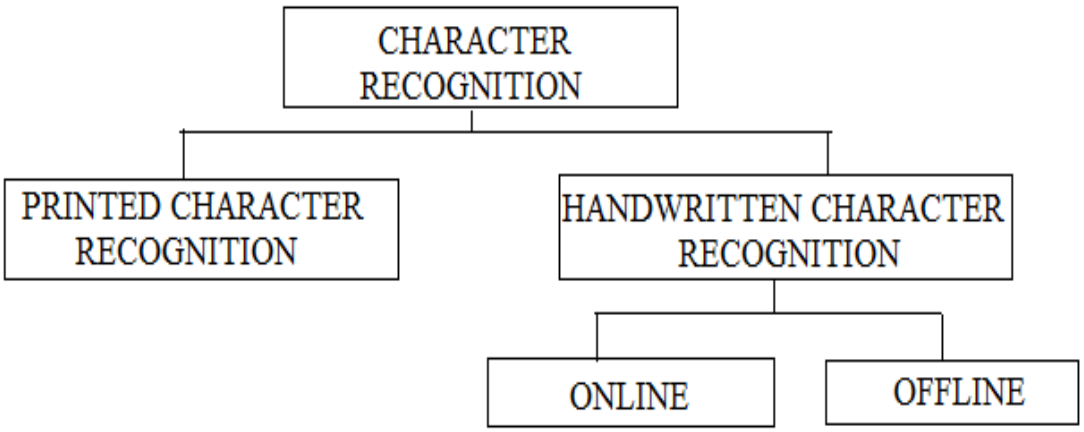


Figure 1.1: Categories of Character Recognition

Handwritten Character recognition can be categorized into following two parts:-

- Online Handwritten Character Recognition
- Offline Handwritten Character Recognition

Offline handwritten character recognition refers to the process of recognizing words that has been scanned from a surface (such as a sheet of paper) and is stored digitally in grey scale format. After being stored, it is conventional to perform further processing to allow superior recognition.

In case of online handwritten character recognition, the handwriting is captured and stored in digital form via different means. Online handwritten character recognition deals with automatic conversion of characters, which are written on a special digitizer, tablet personal computer (PC) or personal digital assistant (PDA) where a sensor picks up the pen tip movements as well as pen up/pen down switching. Usually, a special pen is used in conjunction with an electronic surface for taking the input. As the pen moves across the surface, the two dimensional coordinates of successive points are represented as a function of time and are stored in order. The comparison between the online and the offline types of recognition can be as shown here:-

| Sr. No. | Basis | Offline Characters | Online Characters |
|---|---|---|---|
| 1. | Input Media | Paper Document | Using digital pen on LCD surface |
| 2. | Pen Strokes Available | No | Yes |
| 3. | Raw Data | Dots per inch | Samples per second |
| 4. | Recognition Rate | Low | High |
| 5. | Accuracy | Low | High |

Table 1.1: Comparison between Online and Offline Handwritten Characters

It is generally accepted that the online method of recognizing handwritten text has achieved better results than its offline counterpart. This may be attributed to the fact that more information may be captured in the online case such as the direction, speed and the order of strokes of the handwriting. The major difference between online and offline character recognition is that online character recognition has real time contextual information but offline data does not. This difference generates a significant divergence in processing architectures and methods.

Handwritten character recognition is more difficult to implement than printed character recognition due to diverse human handwriting styles and customs.

Now, since the document image has been acquired with the help of a scanner as this work deals with the offline handwritten character recognition, it is required to go through many processes to be recognized. Such processes can be grouped into stages, which are discussed in the next section.

## 1.4 Stages of Character Recognition

The character recognition process includes hierarchical tasks, which are grouped into the stages of the character recognition as preprocessing, segmentation, recognition and postprocessing. In some methods, some of the stages can merged or omitted; in others, a feedback mechanism can be used to update the output of each stage depending on the field of application. The stages are as depicted in the figure 1.2.

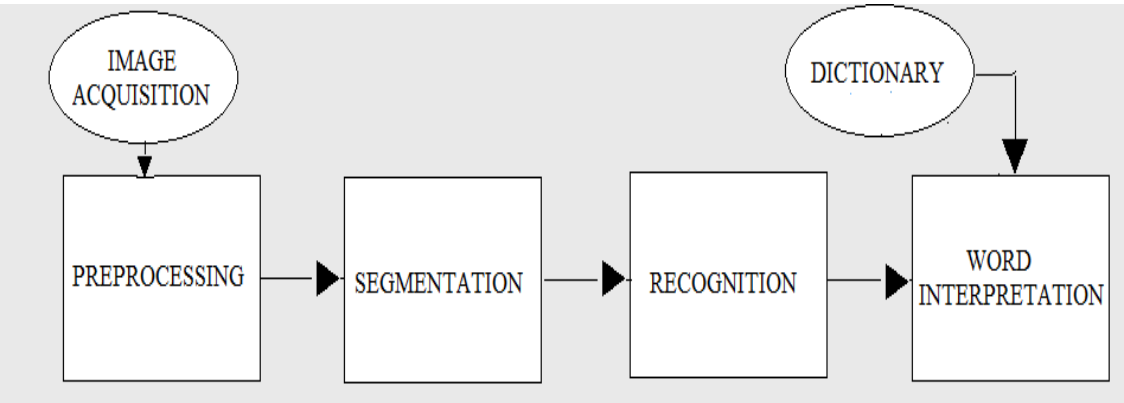


Figure 1.2: Stages of Character Recognition

The various stages can be summarized as:-

- **Preprocessing**

  The image is subjected to several initial processing steps to make it utilizable in the further stages of character recognition process. It aims to produce data that is easy for the character recognition systems to operate upon. It covers all functions to produce cleaned up version of original image.

- **Segmentation**

  The preprocessing stage yields a clean document, which is segmented into its sub components in this stage. Segmentation is an important stage because the extent one can reach in separation of words, lines or characters directly affects the recognition rate of the script. It is the isolation of various writing units, such as paragraphs, sentences, words and letters.

- **Recognition**

  The character recognition process consists of feature extraction and classification. Firstly, the image is analysed and checked for certain specific features. Then, a classification algorithm based on the extracted features, assigns a character image to a specific class. The similar kind of features are often grouped to form a class, which distinguishes its members from the members of the other class.

- **Postprocessing**

  The postprocessing stage, which is the final stage, improves recognition by refining the decisions taken by the previous stage and recognizes words by using

context information. The post processing stage may provide a feedback to the early stages of character recognition.

In order to have a good understanding of the character recognition process, the stages of the character recognition process are discussed in detail in the following section.

## 1.5 Preprocessing

The raw data, depending on the data acquisition type, is subjected to a number of preliminary steps to make it usable in the descriptive stages of character recognition. It aims to produce data so that it is easy for the CR systems to operate accurately. It aims is to maximize the signal to noise ratio of the input and to suppress non relevant data.

There exist a whole lot of tasks to complete before the actual character recognition operation commences. These preceding tasks make certain that the scanned document is in a suitable form and ensures that the input for the subsequent recognition operation remains intact.

The main objectives of preprocessing and techniques to achieve them, as stated by Arica and Vural [2001], are explained here:-

- **Noise Reduction**

  The noise, introduced by the optical scanning device or the writing instrument, causes disconnected line segments, bumps and gaps in lines, filled loops, etc. The distortion, including local variations, rounding of corners, dilation, and erosion, is also a problem.

  The major purpose for noise removal is to remove any unwanted bit patterns, which do not have any significance in the output. The most important reason to reduce noise is that extraneous features will otherwise, cause subsequent errors in recognition. Another reason is that noise reduction reduces the size of the image file, and this in turn reduces the time required for subsequent processing and storage.

- **Normalization of the data**

  Normalization methods aim to remove the variations of the writing and obtain standardized data.

- **Reduction in the amount of information to be retained**

  It is well known that classical image compression techniques transform the image from the space domain to those domains, which are not suitable for recognition. Compression for character recognition requires space domain techniques for preserving the shape information.

The basic techniques used for noise reduction namely, filtering and morphological operations are explained here:-

### 1.5.1 Filtering

Filtering aims to remove noise and diminish spurious points, usually introduced by uneven writing surface or poor sampling rate of the data acquisition device. Various spatial and frequency domain filters can be designed for this purpose. The basic idea is to convolute a predefined mask with the image to assign a value to a pixel as a function of the gray values of its neighbouring pixels. Filters can be designed for smoothing, sharpening, thresholding, removing slightly textured or coloured background, and contrast adjustment purposes. The objective in the design of a filter is to reduce noise while retaining all the relevant data. The filters can be classified into linear and non linear filters, as per Saba *et al.* [2010].

- **Linear Filters**

  These filters are mathematically simple. The popular filters used are described below:-

  - **Mean Filter**

    Mean filter is the optimal linear filter using the mean square error criteria. However, it blurs edges, fine detail, destroys lines and performs poorly in the presence of signal dependent noise.

  - **Wiener Filter**

    To overcome the weaknesses of mean filter, Wiener filtering method is proposed. However, it removes the noise on the cost of blurring text and edges and it operates well if the underlying signal is smooth.

- **Non Linear Filters**

  Nonlinear filters are those that could maintain structural information. Using non linear filters, noise is removed without exceptionally identifying it. The non

linear filtering technique for noisy images in the current state of art is the median filter and its adaptations.

- **Median Filter**

  In median filtering, the neighbouring pixels are ranked according to brightness (intensity) and the median value becomes the new value for the central pixel.

  Salt and pepper noise is a prevalent artifact in poorer quality document images (such as poorly thresholded faxes or poorly photocopied pages). This appears as isolated pixels or pixel regions of noise in backgrounds or noise within foreground regions, and as rough edges on character and graphics components. Median filter can do an excellent job of rejecting this kind of noise in which some individual pixels have extreme values. In the median filtering operation, the pixel values in the neighbourhood window are ranked according to intensity, and the middle value (the median) becomes the output value for the pixel under evaluation. The median filter operation can be seen in image below:-

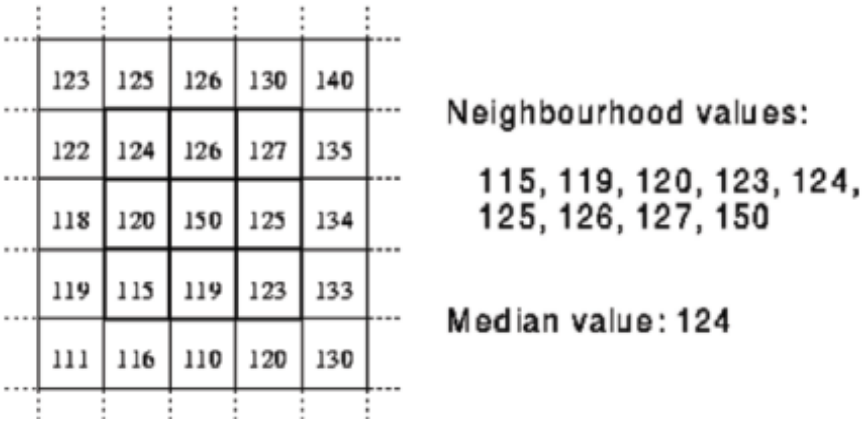


Figure 1.3: Median Filtering Example

The advantages of median filter are:-

- No reduction in contrast across steps, since the output values available consist only of those present in the neighbourhood (no averages).
- Median filtering does not shift boundaries, as can happen with mean filters.

- Since the median is less sensitive than the mean to extreme values (outliers), those extreme values are more effectively removed.

The median filter is more expensive to compute than a linear filter.

Another common technique for noise reduction is using the morphological operations, which is discussed in the following section.

### 1.5.2 Morphological Operations

The morphology is a methodology based on set theory for studying image structure. The two basic operators used for all morphological operations are erosion and dilation. These two operators can be used to remove salt and pepper noise by applying erosion followed by dilation operations, known as opening, and their inverse, known as closing, over the entire image. Morphological operations may damage the edge integrity of both text and graphic components when large structuring elements are used.

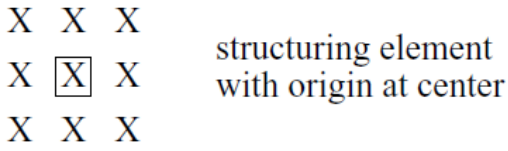


Figure 1.4 (a): Structuring Element of Morphological Operation

The figure 1.4 (a) describes the structuring element that is centred on each pixel in the image and pixel values are changed as follows. For erosion, an ON valued center pixel is turned OFF if the structuring element is over one or more OFF pixels in the image. For dilation, an OFF valued center pixel is turned ON if the structuring element is over one or more ON pixels in the image.

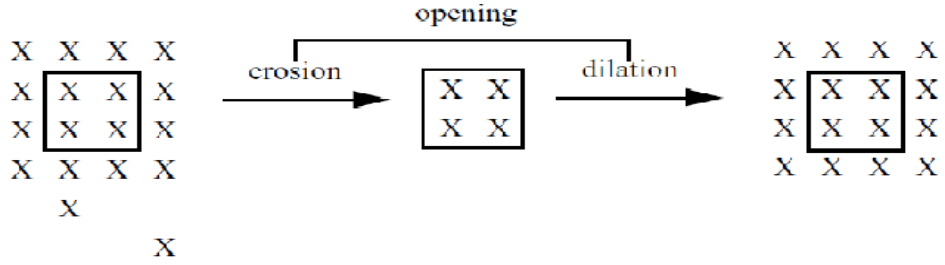


Figure 1.4 (b): Opening Operation

In figure 1.4 (b), erosion is followed by dilation; that combination is called opening. One can see that the isolated pixel and the spur have been removed in the result.

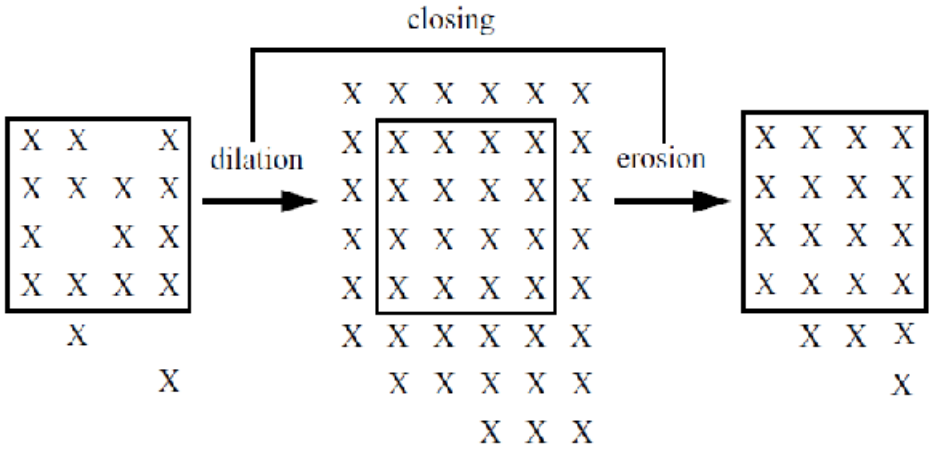


Figure 1.4 (c): Closing Operation

In Figure 1.4 (c), dilation is followed by erosion; that combination is called closing. One can see that the hole is filled, the concavity on the border is filled, and the isolated pixel is joined into one region in the result.

The basic techniques used for normalization namely, skew normalization and baseline extraction, slant normalization and size normalization and contour smoothing are explained in the following section.

### 1.5.3 Skew Normalization and Baseline Extraction

Due to inaccuracies in the scanning process and writing style, the writing may be slightly tilted within the image. This can hurt the effectiveness of later algorithms and, therefore, should be detected and corrected.

Additionally, some characters are distinguished according to the relative position with respect to the baseline. The method for baseline extraction includes the usage of the projection profile of the image.

### 1.5.4 Slant Normalization and Size Normalization

One of the measurable factors of different handwriting styles is the slant angle between longest stroke in a word and the vertical direction. Slant normalization is used to normalize all characters to a standard form. The most common method for

slant estimation is the calculation of the slant angle, which is the average angle of near vertical elements.

Size normalization is used to adjust the character size to a certain standard. Methods of CR may apply both horizontal and vertical size normalizations. Generally, the character is divided into number of zones and each of these zones is separately scaled.

### 1.5.5 Contour Smoothing

The objective of contour smoothing is to smooth the contours of broken and/or noisy input characters. It eliminates the errors due to the erratic hand motion during the writing. High speed of handwriting results into missing points. These missing points can be interpolated through various techniques.

The basic techniques used for size reduction are thresholding and thinning, which are explained in the following section.

### 1.5.6 Thresholding

In order to reduce storage requirements and to increase processing speed, it is often desirable to represent grayscale or color images as binary images by picking a threshold value.

The data in the form of pixels with intensity values are not likely to have only two levels, but instead a range of intensities. This may be due to non uniform printing or non uniform reflectance from the page, or a result of intensity transitions at the region edges that are located between foreground and background regions. The objective of thresholding is to mark pixels that belong to true foreground regions with a single intensity and background regions with a different intensity. However, for the many documents that have a wide range of background and object intensities, the fixed threshold level often does not yield images with clear separation between the foreground components and background. For instance, when a document is printed on differently colored paper or when the foreground components are faded due to photocopying, or when different scanners have different light levels, the best threshold value will also be differ. The techniques for thresholding can be categorized as global and local thresholding, as per Senior and Robinson [1998], are explained here:-

- **Global Thresholding**

  If the pixel values of the components and those of the background are consistent in their respective values over the entire image, then a single threshold value can be found for the image. This use of a single threshold for all image pixels is called global thresholding.

  It picks one threshold value for the entire document image, which is often based on an estimation of the background level from the intensity histogram of the image. For an image with well differentiated foreground and background intensities, the histogram will have two distinct peaks. The valley between these peaks can be found as the minimum between two maxima and the intensity value there is chosen as the threshold that best separates the two peaks.

  The thresholding techniques are shown in the figure 1.5 with various choices of the threshold values for the technique. The low threshold gives an unclear image while the high threshold removes some of the important details from the image. Hence, the selection of good threshold is necessary for thresholding technique to achieve its objective.

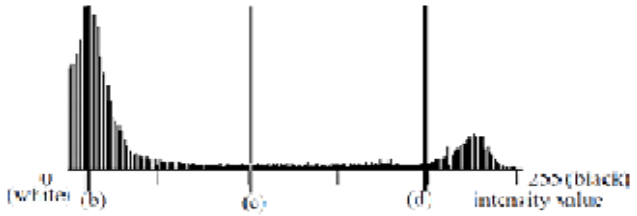


Figure 1.5 (a): Histogram of Original Grayscale Image


Figure 1.5 (b): Image Thresholded with too Low Threshold Value


Figure 1.5 (c): Image Thresholded with Good Threshold Value

Figure 1.5 (d): Image Thresholded with Too High Threshold Value

- **Local Thresholding**

  The images do not always contain well differentiated foreground and background intensities due to poor contrast and noise. Local thresholding is done by analyzing gray level intensities within local windows across the image to determine local thresholds. It uses different values for each pixel according to the local area information.

  The main problem with this technique is the choice of window size. The chosen window size should be large enough to guarantee that a large enough number of background pixels are included to obtain a good estimate of average value, but not so large as to average over non uniform background intensities.

### 1.5.7 Thinning

The purpose of thinning is to reduce the image components to their essential information so that further analysis and recognition are facilitated. For instance, the same words can be handwritten with different pens giving different stroke thicknesses, but the literal information of the words is the same. It provides a tremendous reduction in data size, thinning extracts the shape and context information of the characters.

It refers to the process of reducing the width of a line like object from many pixels wide to single pixel. This process can remove irregularities in letters and in turn, makes the recognition algorithm simpler because they have to operate on a character stroke, which is only one pixel wide. The basic iterative thinning operation is to examine each pixel in an image within the context of its neighbourhood region of at least 3 x 3 pixels and to peel the region boundaries, one pixel layer at a time, until the regions have been reduced to thin lines.

The cleaned image acts as input for the subsequent stages of character recognition process. The next stage of the character recognition process is segmentation that is explained in the next section.

## 1.6 Segmentation

The preprocessing stage yields a clean document in the sense that a sufficient amount of shape information, high compression, and low noise on a normalized image is obtained. The next stage is segmenting the document into its subcomponents. Segmentation is an important stage because the extent one can reach in separation of words, lines, or characters directly affects the recognition rate of the script.

Segmentation is the most challenging part of developing recognition software. Some systems force segmentation by providing individual boxes to write in instead of one flowing input section. It is seen that the segmentation decision is interdependent with local decisions regarding shape similarity and with global decisions regarding contextual acceptability.

It is a process that seeks to decompose an image of a sequence of characters into sub images of individual symbols by segmenting lines and words. There are two types of segmentation namely, external and internal segmentation as per Arica and Vural [2001], are explained in the next section.

### 1.6.1 External Segmentation

It is the isolation of various writing units, such as paragraphs, sentences, or words. It is the most critical part of the document analysis, which is a necessary step prior to the offline character recognition. Although document analysis is a relatively different research area with its own methodologies and techniques, segmenting the document image into text and non text regions is an integral part of the optical character recognition software.

### 1.6.2 Internal Segmentation

It is the isolation of letters, especially in cursively written words. Although the methods have progressed remarkably in the last decade and a variety of techniques have emerged, segmentation of cursive script into letters is still an unsolved problem.

The character segmentation strategies namely, explicit segmentation, implicit segmentation and mixed strategies are explained here:-

- **Explicit Segmentation**

  In this strategy, the segments are identified based on "character like" properties. The process of cutting up the image into meaningful components is given a special name, dissection.

  Dissection is a process that analyzes an image without using a specific class of shape information. The criterion for good segmentation is the agreement of general properties of the segments with those expected for valid characters.

- **Implicit Segmentation**

  This segmentation strategy is based on recognition. It searches the image for components that match predefined classes. Segmentation is performed by the use of recognition confidence, including syntactic or semantic correctness of the overall result. In this approach, two classes of methods can be employed, which are methods that make some search process and the methods that segment a feature representation of the image.

- **Mixed Strategies**

  They combine explicit and implicit segmentation in a hybrid way. A dissection algorithm is applied to the image, but the intent is to over segment, *i.e.*, to cut the image in sufficiently many places that the correct segmentation boundaries are included among the cuts made. Once this is assured, the optimal segmentation is sought by evaluation of subsets of the cuts made. Each subset implies a segmentation hypothesis, and classification is brought to bear to evaluate the different hypothesis and choose the most promising segmentation.

The next section elaborates upon the hierarchy of the segmentation of the document image.

### 1.6.3 Segmentation Hierarchy

The segmentation hierarchy of the document can be described based on the structure of the document. A document generally consists of several pages. A page usually

contains several lines of text, which are, in turn made up of words. The words can be further seen as a grouping of characters. Based on this structure, the segmentation hierarchy can be defined as shown here:-

- **Page Segmentation**

  Page segmentation can be defined as the process of extracting pages from a document.
- **Line Segmentation**

  Line segmentation can be defined as the process of extracting lines from a page.
- **Word Segmentation**

  Word segmentation can be defined as the process of extracting words from a line.
- **Character Segmentation**

  Character segmentation can be defined as the process of extracting characters from a word.

The various strategies that can be applied to perform the segmentation of the document according to the above explained hierarchy. The strategies for segmentation are dealt with, in the next section.

### 1.6.4 Segmentation Strategies

According to Casey and Lecolinet [1996], the segmentation strategies, which can be categorized as classical approach, recognition based segmentation approach and holistic approach are discussed here:-

- **Classical Approach**

  In classical approach, segments are identified based on “character like” properties such as height, width, separation from neighbouring components, disposition along a baseline, etc. This technique cuts image into a sequence of sub images, which are called dissections.

  The criterion for good segmentation is the agreement of general properties of the segments obtained with those expected for valid characters. Examples of such properties are height, width, separation from neighboring components, disposition along a baseline, etc. This process of cutting up the image into meaningful components is given a special name, dissection. Under dissection,

there are various techniques like White Space and Pitch, Projection Analysis, Bounding Box analysis.

- **Recognition Based Segmentation Approach**

  After an image has been segmented into regions, it is ready to enter the next level that is, the feature extraction stage. The result of the image acquisition, preprocessing, segmentation and character fragmentation is a matrix of numbers that represents a character fragment in some way. In the general case, however, the matching of these numbers to a template may be too time consuming and not flexible enough. Therefore, feature extraction is needed. It assigns an input character to one of many pre specified classes, which are based on the extracted features and their analysis.

  Recognition based segmentation, in which, the system searches the image, for those components that match the classes in its alphabet. Many people think that the existence of reliable features to distinguish boundaries in all fonts from interior regions is arguable and the open loop approaches, segmentation to recognition render errors irrecoverable. Therefore, character segmentation should be closely coupled with character recognition.

  Structural features of each character fragment are extracted in this method. The recognition based segmentation techniques are sliding window method, closed loop segmentation and recognition, multiple hypothesis scheme.

- **Holistic Approach**

  There are many applications that require the recognition of unconstrained handwritten words. A word can be either purely numeric as in the case of a ZIP Code, or purely alphabetic as in the case of US state abbreviations, or mixed as in the number of an apartment. In general, a character string recognizer has many applications. The applications include, but are not limited to, reading bank cheques, reading tax forms and interpretation of postal addresses. The task becomes particularly challenging when adjacent characters in a character string are touching. Unlike purely alphabetic strings where joining of the characters is natural and takes place by means of ligatures, the joining of numerals in a numeric word and the uppercase characters in an abbreviation are accidental. The various ways in which two digits can touch are categorized. Some of the

categories lend themselves to natural segmentation, whereas for some the holistic approach is the only option available.

In holistic method the system seeks to recognize words as a whole, thus, avoiding the need to segment into characters. Holistic methods in essence revert to the classical approach with words as the alphabet to be read. This method performs feature extraction in the first step, then global recognition by comparing the representation of the unknown word with those of the references stored in the lexicon. Consequently, this method uses the classical approach.

One of the advantages of the holistic approach is that no distinction needs to be made among touching and non touching pairs. A segmentation based method must necessarily make such a determination. While the holistic method applies equally to both touching and non touching characters, its advantage over traditional segmentation based methods is more pronounced among the touching character pairs.

The explained segmentation strategies can be represented in the 3D space, as per Casey and Lecolinet [1996], as shown below:-

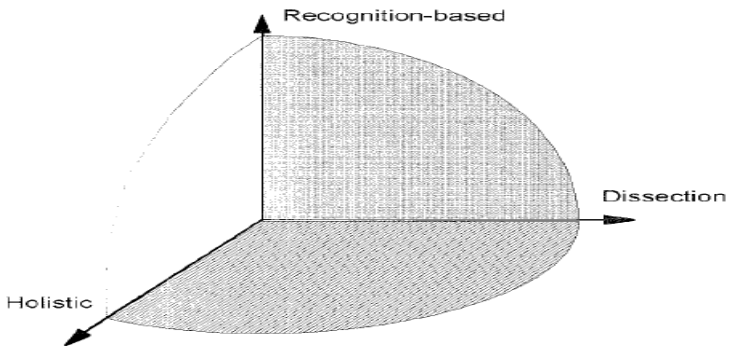


Figure 1.6: Segmentation Strategies in 3D Space

The three fundamental strategies *i.e.* classical approach, Recognition based segmentation and the holistic approach can be represented in 3D space where each approach occupies orthogonal axes.

Sometimes, these strategies alone fail to separate the touching characters in 2D script. Thus, the technique can be a hybrid approach that can be the combination of either classical approach and recognition based segmentation or the combination of

recognition based segmentation and the holistic method or the combination of holistic method and the classical approach.

The segmented image now act as input for the recognition stage which generally, assign an unknown sample to a predefined class to identify the character according to the script used, is explained in the next section.

## 1.7 Recognition

The OCR process assigns a character image to a class by using a classification algorithm based on the features extracted and the relationships among the features. Since members of a character class are equivalent or similar in as much as they share defining attributes, the measurement of similarity, either explicitly or implicitly, is central to any classifier.

Feature extraction is concerned with recovering the defining attributes obscured by imperfect measurements. To represent a character class, either a prototype or a set of samples must be known. The feature selection process attempts to recover the pattern attributes characteristic of each class. Global features, such as the number of holes in the character, the number of concavities in its outer contour, and the relative position of character extremities, and local features, such as the relative positions of line endings, line crossovers, and corners are commonly used. The classification stage identifies each input character image by considering the detected features.

In the statistical classification approaches, character image patterns are represented by points in a multidimensional feature space. Each component of the feature space is a measurement or feature value, which is a random variable reflecting the inherent variability within and between classes. A classifier partitions the feature space into regions associated with each class, labelling an observed pattern according to the class region into which it falls. Numerous techniques for character recognition can be investigated into four general approaches of pattern recognition, as per Arica and Vural [2001], are explained here:-

### 1.7.1 Template Matching

The features can be as simple as the gray level image frames with individual characters or words or as complicated as graph representation of character primitives.

The simplest way of character recognition is based on matching the stored prototypes against the character or word to be recognized, which is template matching. The matching operation determines the degree of similarity between two vectors in the feature space.

### 1.7.2 Statistical Techniques

Statistical decision theory is concerned with statistical decision functions and a set of optimality criteria, which maximizes the probability of the observed pattern given the model of a certain class. The measurements taken from n features of each word unit can be thought to represent an n dimensional vector space and the vector, whose coordinates correspond to the measurements taken, represents the original word unit. The major approaches are parametric recognition, non parametric recognition, clustering analysis and markov modelling.

### 1.7.3 Structural Techniques

The recursive description of a complex pattern in terms of simpler patterns based on the shape of the object was the initial idea behind the creation of structural pattern recognition. These patterns are used to describe and classify the characters in the character recognition systems. The characters are represented as the union of the structural primitives. It is assumed that the character primitives extracted from writing are quantifiable, and one can find the relations among them. The major techniques include grammatical methods and graphical methods.

### 1.7.4 Neural Networks

A neural network is defined as a computing architecture that consists of a massively parallel interconnection of adaptive neural processors. Because of its parallel nature, it can perform computations at a higher rate compared to the classical techniques. Because of its adaptive nature, it can adapt to changes in the data and learn the characteristics of input signal. Several approaches exist for training of neural networks.

The recognized characters are fed into the post processing stage, which uses some contextual information to increase the recognition rate of the system, is explained in the next section.

## 1.8 Word Interpretation (Post Processing)

The incorporation of context and shape information in all the stages of character recognition systems is necessary for meaningful improvements in recognition rates. This is done in the postprocessing stage with a feedback to the early stages of character recognition. The simplest way of incorporating the context information is the utilization of a dictionary for correcting the minor mistakes of the character recognition systems. In postprocessing, a dictionary can be used to restrict the character combinations. This can be implemented as a grammar that specifies all possible combinations of characters. The basic idea is to spell check the character recognition output and provide some alternatives for the outputs of the recognizer that do not take place in the dictionary.

Thus, the various stages of the recognition process have been discussed. Now, there arises a need to have a look on the various applications, to which the character recognition process can be applied to. The next section describes some of the applications of the character recognition process.

## 1.9 Applications of Offline Handwritten Character Recognition

Some of the important applications of offline handwritten character recognition are discussed in the following section:-

- **Cheque Reading**

  Offline handwritten character recognition is basically used for cheque reading in banks. Cheque reading is the very important commercial application of offline handwritten character recognition. Handwritten recognition system plays very important role in banks for signature verification and for recognition of amount filled by user.

- **Postcode Recognition**

  Handwritten recognition system can be used for reading the handwritten postal address on letters. Offline handwritten character recognition system is used for recognition of the handwritten digits of postcode. Handwritten character recognition system can be read this code and sort the mails automatically using the distinct postcode for regions.

- **Form Processing**

  Handwritten character Recognition can be also used for form processing. Forms are normally used for collecting the public information. Replies of public information can be handwritten in the space provided and this handwritten data can be processed using the character recognition process.

- **Signature Verification**

  Handwritten character Recognition can also be used to identify the person through signature verification. Signature identification is the specific field of handwritten identification in which the writer is verified by some specific handwritten text. Handwritten recognition system can be used to identify the person by handwriting, because handwriting varies from person to person.

Some of the common terms used throughout the document are listed here:-

## 1.10 Terminology Used

- *Binarization:-* It is the process of conversion of the grayscale or color images into binary images by picking a threshold value, which reduces the storage requirements.
- *OCR:-* It is the process of transforming the graphical marks associated with human handwritten document into the symbols that are stored on a computer system in the form of 8-bit ASCII code or 16-bit Unicode.
- *Segmentation*:- It is a process that seeks to decompose an image of a sequence of characters into sub images of individual symbols by segmenting lines and words. In other words, it decomposes a document into its sub components.
- *Water Reservoir*:- The water reservoir can be defined as if water is poured from one side of a component, the cavity regions of the component where water will be stored are considered as reservoirs. There can be top, bottom, left, and right reservoirs based on the side of the cavity.

This concludes the introduction to the character recognition process. To understand the current state of art, the literature has been reviewed and presented in the next chapter.

# CHAPTER 2

# LITERATURE SURVEY

The field of character recognition has been studied extensively in past few decades, owing to its application in reading the details like postal zip code, employee code, and passport number, processing of forms and bank cheques and many more. However, it still acts as a challenging field for the researchers. The inexplicable research work in this area has led to the development of innumerable approaches to deal with the various aspects of the character recognition. The approach for the character recognition process may vary according to the script under consideration. In India, many official languages such as Hindi, Marathi, Sindhi, Nepali and Sanskrit are from the Devnagari script. Across central and northern parts of India, more than 300 million people use Devnagari script for documentation. In order to understand the current state of art in this area, a survey of work done related to the character recognition process has been presented in this chapter.

Lu [1995] summarized the various character segmentation techniques for uniform and proportional fonts, touching and broken characters. The segmentation of fixed pitch text, broken characters, and kerned characters has also been summed up. The various steps for textual processing like determining skew, finding paragraphs and columns, performing optical character recognition has been explained. The segmentation techniques based on character size, contour analysis and vertical projections have been discussed. The segmentation of touching characters via discriminating multiple character components, feature based segmentation has also been discussed. The recognition based segmentation techniques like sliding window method, closed loop segmentation and recognition have also been introduced in the paper. The techniques based on text image features and recognition results have also been discussed.

Peters II [1995] presented a description and analysis of a new morphological image cleaning algorithm (MIC) that preserves thin features while removing noise. The usefulness of the MIC for gray scale images corrupted by dense, low amplitude, random or patterned noise that has typical of scanned or still video images has been explained. The way MIC differs from previous morphological noise filters in that it

manipulates residual images – the differences between the original image and morphologically smoothed versions has also described. MIC creates a cleaned image by recombining the processed residual images with a smoothed version. This paper also explains the effects of parametric variations, presents the results of a noise analysis and shows a number of examples of its use, including the removal of scanner noise.

Casey and Lecolinet [1996] provided a review of various techniques and methodologies in character segmentation. The importance of segmentation in the recognition process and various steps of classical optical character recognition process have been discussed. They divided the segmentation strategies into classical approach, recognition based approach and holistic approach. The classical approach that identifies the segments based on character like properties has been discussed. The recognition based approach that finds those components in image that matches the classes in alphabet and the holistic method recognizes the word as a whole has also been explained in the paper. The dissection techniques like projection analysis, white space and pitch approach and connected component processing have been discussed.

Trier *et al.* [1996] presented an overview of feature extraction methods for the offline recognition of the isolated characters. Different feature extraction methods, which are designed, for the different representation of the characters, such as solid binary characters, character contours, skeletons, gray scale images of each character have been discussed. The feature extraction methods have been explained in terms of invariance properties, expected distortion, reconstructability and variability of the characters. The various processing steps of the optical character recognition process, namely gray level scanning, binarization, segmentation, representation, feature extraction, recognition and contextual processing have also been stated.

Wu and Manmatha [1997] presented a technique for document clean up and binarization. The need for binarization, as well as the issues in binarization has also been discussed. The proposed algorithm smoothens the input image using Gaussian filter and then thresholds it based on the intensity histogram of the image. The global and local thresholding techniques with the related issues in them have also been discussed. A comparison of the proposed algorithm with other popular algorithms like Otsu's Algorithm, Tsai's method has also been made.

Ha and Bunke [1997] presented a new approach to offline handwritten numeral recognition. They proposed a recognition method, which is able to account for a variety of distortions due to eccentric handwriting. The technique for the perturbation based recognition system has also been discussed. The key idea of the perturbation approach, which lies in the process of reversing an input image back to one of its standard forms, has also been stated. The methodology for the parameterization process based on four geometric transformations, namely, rotation, slant, perspective view and shrink has also been explained. The approach to replace normalization by a set of perturbation processes modelling writing habits and instruments has also been discussed.

Senior and Robinson [1998] gave a complete system for handwriting recognition. Preprocessing techniques included binarization and normalization of word images to reduce the effect of scale, skew, slant and stroke thickness. Issues of vocabulary choice and rejection have also been discussed. The techniques for histogram production, baseline estimation, slant correction, smoothing, thresholding, thinning and parameterization have been explained. The recognition techniques based on neural networks and its various types and hidden markov model have also been summarized.

Mo and Mathews [1998] proposed an adaptive filter to be used prior to binarization for the edge enhancement of the characters. The importance of edge enhancement and noise reduction in the document image for the recognition process has also been explained. The paper also stated the similarity of proposed approach with the equalization of the binary communication channels. An introduction to the quadratic filter for the removal of the noise from the document acquired during image acquisition process has also been given. The mathematical description for the quadratic filter and its application has also been given. The design and implementation issues with the proposed algorithm have also been stated in the paper. The low pass and high pass filters and their application for binarization process has also been summarized.

Cai and Liu [1999] proposed an approach that integrates the statistical and structural information for unconstrained handwritten numeral recognition. The approach that uses the state duration adapted transition probability to improve the modelling of state

duration in conventional markov models and uses macro states to overcome the difficulty in modelling pattern structures by markov models has also been explained. The technique for the encoding of the orientations into discrete codebooks and the distributions of locations are modelled by joint Gaussian distribution functions has also been discussed. The preprocessing methods for the disjoint connection region, slant correction, size normalization have also been dealt with.

Blumenstein and Verma [1999] proposed an algorithm for segmenting unconstrained printed and cursive words. The way algorithm initially over segments handwritten word images using heuristics and feature detection has also been explained. The steps of the proposed algorithm, namely heuristic segmentation, manual examination of segmentation points, separation into correct and incorrect classes, extraction of segmentation points, feature extraction, neural network training have also been discussed. The heuristic algorithm containing the steps like hole location, minima detection, vertical pixel density projection has also been stated. The methodologies for the average character width estimation and contour extraction have also been presented. It has been stated that the algorithm can be modified so that it will be possible to detect a smaller number of incorrect segmentation points, while at the same time recovering segmentations that are more correct.

Lehal and Singh [1999] proposed feature extraction and classification scheme for the character recognition of Gurmukhi script. An introduction to the character recognition process and the stages of the character recognition process has been given. The characteristics of the Gurmukhi script have also been elaborated upon. The feature extraction technique using the primary and secondary feature set has also been discussed. The primary feature set, which includes number of junctions with the headline, presence of sidebar, presence of a loop and no loop formed with headline, has also been explained. The secondary feature set which consists of number of end points and their location, number of junctions and their location, horizontal projection count, right profile depth, left profile upper depth, left profile lower depth, direction code and aspect ratio has also been discussed.

Plamondon and Srihari [2000] rolled out a comprehensive survey for the online and offline handwriting recognition. The paper described the features of the handwritten language, their translation into the electronic form and the underlying concepts behind

the handwritten character recognition algorithms. The various algorithms for preprocessing, character and word recognition and their performance have also been described. The classification methods like the structural and rule based methods and the statistical methods have been discussed. The applications of offline handwritten character recognition like handwritten address interpretation, bank cheque recognition, signature verification has also been described. The language tools for language processing and analysis have been stated. The various application areas and recent tools available in these areas have also been presented.

Lehal and Singh [2000] proposed a system for the recognition of the machine printed characters of the Gurmukhi script. The various characteristics of Gurmukhi script have been stated. The preprocessing techniques like skew correction and detection, thinning and smoothing the headlines have been discussed. The horizontal and vertical projection profiles of the characters and their usage for line segmentation and word segmentation has been explained. The character segmentation based on connected component identification has also been discussed. The recognition process based on feature extraction and classification of sub symbol has also been described. The set of simple features used for binary decision trees and nearest neighbour classification techniques used for classification and recognition, has also been explained.

Arica and Vural [2001] reviewed the various techniques available with their strengths and weaknesses. The various tasks required for character recognition have been grouped by them into various categories, namely pre processing, segmentation, representation, training and recognition and post processing. The various preprocessing techniques like filtering, noise modelling, slant normalization, baseline extraction, contour smoothing, thresholding and thinning have also been explained. The external and internal segmentation strategies have been described. The representations like global transform and series expansion, statistical representation and geometric representation have also been explained. The training and recognition techniques like template matching, statistical techniques, structural techniques and neural networks have also been described. The postprocessing technique considering the inclusion of contextual information has also been discussed.

Bansal and Sinha [2001] presented a two pass algorithm for the segmentation and decomposition of Devnagari composite characters into their constituent symbols. The proposed algorithm extensively uses structural properties of the script. The technique through which the words are segmented into easily separable characters has also been explained. The method for using the statistical information about the height and width of each separated box to hypothesize whether a character box has composite has also been dealt with. The technique for the segmentation of the hypothesized composite characters has also been discussed. The features of the script in relevance with the character recognition have also been stated.

Timar *et al.* [2002] described the various algorithms used in the preprocessing and segmentation phase of offline handwritten character recognition process. Various tasks of preprocessing phase such as thinning, localization, deskewing have also been discussed. The strategies for line localisation and word localisation have also been discussed. The segmentation technique for words using the skeleton endpoints has been explained. A novel trigger wave based word segmentation algorithm has been presented in the paper, which operates on the skeletons of words. The technique for localisation of lower and upper baselines, producing the skeleton of the background of the image, producing skeletons of the foreground of the image and their usage in segmentation of words into characters has also been described.

Kasturi *et al.* [2002] gave a detailed description of the document image analysis process. The sequence of steps starting from data capture, pixel level processing, feature level analysis until text recognition and analysis has been elaborated. A brief analysis of graphical documents has been presented. The techniques for noise reduction and binarization have been discussed. The techniques for thinning and region detection, chain coding and vectorization have also been explained. The techniques for line and curve fitting, critical point detection, skew estimation, layout analysis have also been discussed. The strategy for feature extraction and classification based on template matching and contextual processing has been explained. The various OCR's for Indian languages and document analysis in multilingual context has also been stated.

Pal *et al.* [2002] presented a recognition system for unconstrained offline Bangla handwritten numerals. The scheme is mainly based on new features obtained from the

concept of water overflow from the reservoir as well as topological and structural features of the numerals. The technique used for the choice of features has independent of various writing styles of different individuals and has the simplicity of detection. The direction of water overflow, height of water reservoir when water overflows from the reservoir, position of the reservoir with respect to the character bounding box, shape of the reservoir etc. are the main reservoir features used in the proposed scheme.

Verma [2003] presented an algorithm for cursive handwriting recognition based on the rule based segmentation and contour code characteristics. The contour code calculation based on distance between segmentation points used for segmentation of characters has also been discussed. A heuristic segmentation algorithm which has initially used to over segment each word and then, the prospective segmentation points are passed through the rule based module to discard the incorrect segmentation points and include any missing segmentation points has been explained. The technique for proposed rule based module that validates every segmentation points against closed area, average character size, left character and density has also been described. The technique for passing the contour between correct segmentation points through the feature extraction module that extracts the contour code, after which another trained neural network has used for classification, has been explained in this paper. The procedure for grouping the recognized characters into words and passing to a variable length lexicon that retrieves words that has highest confidence value has also been described.

Pal *et al.* [2003] presented an automatic scheme to identify text lines of different Indian scripts from a document. The method for grouping the scripts into some classes in accordance with the script characteristics has also been stated. The various features used for script identification, namely headline feature, horizontal projection profile, water reservoir principle based feature, left and right profile, feature based on jump continuity have also been explained. The technique for the identification of the script based on the various features has also been discussed. The approach is insensitive to font, style and case variation has been stated.

Pal and chaudhuri [2004] presented a review of the OCR work done on Indian language scripts. The character recognition process, its applications, its history has

been explained. The properties of the Indian scripts like basic characters, modified characters, headline and zones have been stated. They stated that the character recognition for the poor quality documents, multi font documents, multi script documents, handwritten documents needs to be developed. The character recognition systems with font and structure information, benchmarking and ground truth generation and post recognition error correction are also required to be designed.

Blumenstein *et al.* [2004] presented a technique for cursive character recognition applicable to segmentation based word recognition systems. The proposed novel feature extraction technique that extracts direction information from the structure of character contours has also been discussed. The methodology for the extension of the principal so that the direction information has integrated with a technique for detecting transitions between background and foreground pixels in the character image has also been explained. The technique to find the direction feature through determination of directions and formation of feature vector has also been stated. The proposed technique has also been compared with the standard direction feature extraction technique and an improvisation has been found.

Govindaraju *et al.* [2005] described the techniques used for slant and slope correction and line and word segmentation in the handwritten Arabic documents. A linear regression technique has been used for baseline detection and connected components have been identified for line and word segmentation. An orientation independent technique for baseline detection of Arabic words has been used. The procedure for using the horizontal projection profile for the line segmentation and the vertical projection profile for the word segmentation has also been discussed. The skew correction algorithm has been used to improvise the results for the line segmentation and word segmentation in the handwritten Arabic documents.

Kompalli *et al.* [2005] presented a wide range of challenges in the character recognition for the Devnagari documents. The segmentation problems for the Devnagari documents like ascender segmentation, core component segmentation and descender segmentation have also been discussed. The technique for the core component classification and the word recognition has also been explained. The methodology for the recognition based segmentation including design of consonant and vowel classifiers and conjunct recognition has also been discussed. The technique

for the dictionary based post processing of the Devnagari document has also been stated. The method to segment the conjunct using the adjacency graph has also been discussed.

Sharma *et al.* [2006] proposed a quadratic classifier based scheme for the recognition of offline Devnagari handwritten characters. The Devnagari script with its basic character set and the modifiers has also been explained. The features used in the classifier have been obtained from the directional chain code information of the contour points of the characters. The technique for the segmentation of the bounding box of a character into blocks and the computation of the chain code histogram in each of the blocks has been discussed. Based on the chain code histogram, 64 dimensional features have been used for recognition. The comparison of the results obtained by the proposed algorithm and the past algorithms has also been made.

Jayarathna and Bandara [2006] presented an approach for the segmentation of offline handwritten connected two digit strings. They developed an algorithm, which provides a solution based on the analysis of the foreground pixel distribution to segment the connected digit string pairs. The junction based splitting technique, which decides complete segments of the connected digit strings, has also been explained. The use of the fuzzy characteristic values at the merging of the complete segments that isolates the major segments from the minor segments has also been discussed. The process for binarization, isolation of connected character skeleton into correlation area, junction based segmentation, identification of starter points and junction points, traversal through the correlation area and merging of segments across junctions has also been stated.

Banashree and Vasanta [2007] proposed a recognition scheme for handwritten Hindi numerals. The work focused on a technique in feature extraction *i.e.* global based approach using end points information, which is extracted from images of isolated numerals has been explained. These feature vectors are fed to neuromemetic model that has been trained to recognize a Hindi numeral. In the proposed scheme, data sets are fed to neuromemetic algorithm, which identifies the rule with highest fitness value and the template associates with this rule has nothing but identified numerals. A global based approach using end points information for feature extraction has been used.

Arora *et al.* [2007] presented a two stage classification approach for handwritten Devnagari character. The first stage that uses the structural properties like shirorekha, spine in character has been discussed .The second stage which exploits some intersection features of characters which are fed to a neural network has also been explained. The preprocessing stage including size determination, distortion removal and normalization has also been explained. They stated that the simple histogram based method does not work for finding shirorekha, vertical bar in handwritten Devnagari characters. They designed a differential distance based technique to find a near straight line for shirorekha and spine in the characters.

Ramteke and Mehrotra [2008] employed a method based on invariant moments and the divisions of numeral image for the recognition of handwritten Devnagari numerals. The technique has independent of size, slant, orientation, translation and other variations in handwritten characters. The technique for normalization to a fixed pixel size for each individual image after the segmentation has also been explained. Seven central invariant moments have been evaluated for each character and its parts by dividing it by three different ways. In all, there are 78 features corresponding to each character. The Gaussian distribution function has been adopted for classification. The method to separate a text line from the previous and following lines by white space with horizontal projection has been stated.

Pal *et al.* [2008] proposed a quadratic classifier based scheme for the recognition of offline handwritten characters of three popular south Indian scripts namely, Kannada, Telugu, and Tamil. The technique to obtain the features used from the directional information has been stated. The method for feature computation that includes the segmentation of the bounding box of a character into blocks and the computation of the directional features in each block has also been discussed. The methodology for down sampling the blocks by a Gaussian filter and feeding the generated features from the down sampled blocks to a modified quadratic classifier for recognition has also been explained. They have used two sets of features namely, 64 dimensional features for high speed recognition and 400 dimensional features for high accuracy recognition.

Kumar and Singh [2008] presented an algorithm for the segmentation of the handwritten text in Gurmukhi script. They stated that the technique to segment the

characters is to use inter character gap as a segmentation point. The various characteristics of the Gurmukhi script have been explained. The method for the preprocessing of the document to remove the noise has also been stated. The algorithm for line detection, word detection and character detection has been discussed.

Agrawal *et al.* [2009] described a system to classify the offline handwritten Hindi characters into several groups based on some similarity measure. A novel method has been proposed for finding the header line, based on end points and pixels positions in the top half part of the character image. A new algorithm has been designed for the identification of presence and position of vertical bar in the handwritten Hindi characters. The technique for the separation of pattern classes into several groups based on the similarity of characters has also been explained. The technique for identification of the non connected characters, end bar, middle bar and without bar characters and closed loop has also been stated.

Benne *et al.* [2009] designed an automatic recognition system for isolated handwritten numerals recognition for three popular south Indian scripts. Kannada, Devnagari and Telugu numeral sets have been used for their recognition. The applications of the automatic numeral recognition have also been stated. The proposed method has thinning free and without size normalization. The structural features viz. directional density of pixels, water reservoirs, maximum profile distances, and fill hole density have been used for handwritten numerals recognition. A Euclidian distance criterion and K nearest neighbour classifier, which has used to classify the handwritten numerals has also been explained. The comparison of the obtained results with other algorithms has also been given.

Prasad *et al.* [2009] presented a system for the offline handwritten character recognition of Gujrati script using pattern matching. The methods for image acquisition, median filtering, image inversion, image thinning, segmentation and recognition have been explained in this paper.

Leary [2009] presented a preprocessing approach for the unrestricted offline handwritten character recognition. The technique for the line segmentation, which is based on the horizontal histogram and the minima in the histogram, has been explained. The method for skew correction, which utilizes the concept of estimation

of the lower baseline and determination of the angle relative to the horizontal axis, has been discussed. The technique for the slant correction that utilizes the affline transformations, vertical projection histogram, and Wigner-Ville histogram has been explained. The techniques for baseline positioning, scaling, word segmentation, character segmentation have also been explained.

Kaur *et al.* [2010] presented a hybrid approach to classify Gurmukhi script characters. The character recognition process and the stages of the process have been explained. The characteristics of the Gurmukhi script have also been stated. The technique that uses the concept of water reservoir as well as the extraction of the features like presence of sidebar, presence of loop, number of components to classify the characters of the script has been discussed. They stated that the proposed approach has classified the Gurmukhi script characters in 11 subclasses and 9 individual characters. This technique performs only 7 checks in order to find out a particular class for a segmented area, which reduces the effort for segmentation.

Kumar and Singh [2010] formulated an approach to segment the scanned document image. The approach that initially considers the whole image as one large window and then breaks the large window into less large windows giving the lines has been explained. The character recognition process and the segmentation strategies have also been elaborated upon. The method that uses the window consisting of identified line to find the words present in that line has been discussed. The algorithm that uses the window defined by identified word to find the characters present in that word has also been explained. They stated that a concept of variable sized window, that is, the window whose size can be adjusted according to needs, has been employed for this approach.

Kumar and Dhiman [2010] elaborated upon the challenges in segmentation of text in handwritten Gurmukhi script. They stated that the segmented part of the image would be such that it should provide a close relation to the character to be recognised. The character recognition process and the segmentation strategies have also been elaborated upon. The significance of segmentation in the recognition process and the various segmentation strategies namely, the classical approach, recognition based segmentation and cut classification methods have also been discussed. The major challenges recognised by the authors are the wide variations in handwriting styles,

which make it very difficult to make generalizations to design the segmentation heuristics, broken and touching characters and the overlapping characters.

Garg *et al.* [2010] presented a segmentation method for the handwritten Hindi text. The various characteristics of the Devnagari script have been stated. They proposed a line segmentation method that is based on header line detection and base line detection. The authors have used two stripe projections for header and base line detection. This method gives good results for uniform and non uniform skewed lines has been stated by them. The technique for the segmentation of the words from lines, which utilizes the vertical projection profile, has been discussed. It has been explained that for each column of the line, the number of black pixels is counted and the columns with zero black pixels have been used as delimiters for word separation. The method for character separation, which uses the vertical projection and header line detection, has also been elaborated upon.

Dongre and Mankar [2010] presented a review of research on the Devnagari character recognition. The various stages of the Devnagari character recognition process have been discussed. The features of the Devnagari script like conjuncts, compound characters, shirorekha have been explained. The various techniques for preprocessing like thresholding, noise reduction, skew detection and correction, size normalization and thinning have been elaborated upon. The various feature extraction methods like global transformation and series expansion, statistical features, geometrical and topological features have also been discussed in detail. The various classification techniques like template matching, statistical techniques, neural networks, support vector machine, combination classifier have also been stated.

Kumar [2010] presented an analysis of Devnagari script according to machine recognition perspective. A brief review of the Devnagari script including the zone information, various character types has been given by the author. The various issues that cause the discrepancies in the upper region of the text namely, awkward representation of a vowel symbol, incomplete and inaccurate representation of a vowel symbol, merging vowel symbols with headline, intruding upper vowel symbols with middle region have been explained. The similar issues in the lower region of the text namely, intruding a lower vowel symbol with middle region, improper attachment of a lower vowel symbol, chomping a consonant as a result of wrong

attachment of a lower vowel symbol, writing a vowel symbol in isolation, abnormally expressed vowel symbols have also been discussed. The issues in the middle region of the character namely, wrong insertion of headline, incomplete character writing, touching characters with headline due to wrong attachment of a character with it, overwriting have also been explained.

Malakar *et al.* [2010] presented a novel noise removal technique for the document images. They have developed a new filtering technique, called middle of modal class, for smoothing the input images. This technique is applicable for both the noisy and noise free text document image at the same time. The proposed technique that consists of window selection, filtering and binarization has been elaborated upon. The filtering technique that utilizes the pigeonhole principle and median value concept has been discussed. The result for the proposed technique has been compared with the mean and median filters.

Saba *et al.* [2010] presented issues and comparison of the methods in the document image analysis. The techniques for image denoising namely spatial filtering methods and transform domain filtering methods have been elaborated upon. The linear filters like mean filter, wiener filter and the non linear filters like median filter, weighted mean filter have been stated. The skew detection and correction techniques like Hough transform based approach, projection profile based approach, nearest neighbour approaches, interline cross correlation clustering and contour oriented methods has also been discussed.

Kumar and Singh [2011] proposed an algorithm to detect and segment Gurmukhi handwritten text into lines, words and characters. The character recognition process, the segmentation strategies and the characteristics of the Gurmukhi script have been explained. The technique for the preliminary processing that consists of binarization has also been discussed. The technique that uses coordinates of line detected to find the word position present in that line has been stated. The technique that uses these words position coordinates to find characters present in the word has been discussed. The authors have proposed a single module to detect lines and words in the document. The technique used for the character detection, which utilizes reverse engineering concept as one part, is extracted from the word present in the line and this extracted

part is checked whether it has some meaningful symbol as per Gurmukhi, script has also been discussed.

Shukla and Banka [2011] proposed a segmentation scheme for the recognition of the printed Devnagari script. The features of the Devnagari script such as basic characters, modified characters have been explained in the paper. The preprocessing steps consisting of binarization, noise removal, skew detection and correction have been discussed. The segmentation process consisting of the line segmentation, word segmentation and character segmentation has also been elaborated upon. The authors have separated the individual line in a script document image based on the peak of the horizontal histogram. The technique used for the word segmentation works on the basis of the vertical histogram of the extracted line.

RamanaMurthy *et al.* [2012] presented the issues in recognizing the Devnagari characters in the wild like sign boards, advertisements, logos, shop names, notices, address posts. A detailed study of the Devnagari character recognition using the state of art character recognition and object recognition tools has been carried out by them. A description of the Devnagari script consisting of consonants, vowels, conjuncts has been given. The preprocessing techniques smoothing and binarization have been discussed. The features like pixel density, directional features, histogram of oriented gradients, shape context have been explained by the authors. The performance evaluation of the existing state of art features and classifiers on specific database is presented.

Cuc *et al.* [2012] presented the handwritten digit recognition on well known image databases using state of the art feature extraction and classification techniques. The preprocessing technique that includes conversion into gray scale image, conversion into binary image, morphology method and normalization has been explained. The feature extraction methods namely, seven moments and image averaging has also been discussed. The classification methods including neural network and template matching has also been elaborated upon.

The literature review for the character recognition process has been presented in this chapter. Based on this survey, the problem statement and the proposed solution have been discussed in the next chapter.

# CHAPTER 3

# PROBLEM AND PROPOSED SOLUTION

## 3.1 Problem Statement

The character recognition process translates the scanned image into an editable form, which can be processed directly by the computer. This process can be used to translate articles, books and documents into electronic format, to publish text on website, to process the cheques in banks, to sort the letters etc, as explained in chapter 1. It can be observed that many numeral writings are present in these applications and the recognition of the numerals will ease out the working of these applications. Thus, the recognition of the numerals is a significant area to work on. Moreover, the numeral recognition can be integrated with several other fields to help with several applications like reading details like postal zip code, employee code, passport number and processing of forms and bank cheques. It can also be utilized in schools and colleges to process the list of marks. A lot of research work has been done for the numeral recognition for the English language as compared to Indian languages. Moreover, Devnagari is the script used for writing many official languages in India, such as Hindi, Marathi, Sindhi, Nepali, Sanskrit, and Konkani, where Hindi is the national language of the country. Hindi is also the third most popular language in the world. In addition, more than 300 million people use Devnagari script for documentation in central and northern parts of India. Hence, an approach for the recognition for Devnagari numerals will have more significance compared to other scripts.

It is quite evident from the literature review that numerous approaches have been developed for the handwritten numeral recognition process. However, it can also be observed that the handwritten numeral recognition is still a fascinating area for the researchers to design a robust and efficient algorithm for the same. The numerous techniques that have been developed are able to provide good recognition rate ranging from 85% to 90% but these techniques have some implementation gaps. Most of the approaches for recognition process are based on the neural network technique and its adaptations. These techniques are computationally difficult and require good amount

of time to perform the training of the systems in order to provide good results. Moreover, the accuracy of the recognition system depends on the quality of the training of the system. Thus, there is a need for a simpler approach for the character recognition process. The authors have thus, tried to develop a simple and efficient algorithm for the recognition of the handwritten Devnagari numerals, which is discussed in the following section.

## 3.2 Proposed Solution

In an effort to provide a solution to the problem explained in the previous section, a number of techniques have been employed and developed, which are discussed in this section. The handwritten numeral recognition process includes a number of stages, as discussed in chapter 1. Based on these stages, the proposed work has been discussed in three sections as preprocessing in 3.2.1, segmentation in 3.2.2 and recognition in 3.2.3. The section related to preprocessing deals with the cleaning up of the image by means of modified median filtering, binarization, thinning and smoothing techniques. The section pertaining to segmentation discusses the segmentation of the image into sub components by using line segmentation, word segmentation and character segmentation. The section dealing with recognition describes the classification of the image based on the features extracted from the image. The features used are water reservoirs, closed loops and sidebars. The classification structure based on these features is used to perform the recognition.

The handwritten numeral recognition process initiates with the acquisition of the image with the help of the scanner. The acquired image may contain many impurities, which may be introduced due to different reasons like varying paper quality, varying ink quality and the quality of scanner used. Therefore, the acquired image cannot be sent directly to the subsequent stages of recognition process and thus, needs the preprocessing stage, which is discussed in the following section.

### 3.1.1 Preprocessing

The acquired image is subjected to a number of preliminary steps to make it usable in the consequent stages of offline handwritten numeral recognition. Preprocessing aims to produce data so that it is easy for these systems to operate accurately. It includes the techniques for filtering, binarization, thinning and smoothing, which helps to

achieve the objective of preprocessing. This is accomplished through the Preprocessing algorithm.

The Preprocessing algorithm described in this section calls four modules- Modified_Median_Filter to filter the image, Binarization to binarize the image, Thinning to produce skeleton of the image and Smoothing to smoothen the boundaries of the image, which are discussed in the following section.

The document image is initially acquired through scanner, say Input_Image, which is referred as figure 4.1 in the next chapter.

In order to process the bitmap image, it is necessary to understand the structure of the bitmap header, which provides the information about the image properties such as height of image, width of image, resolution of image etc. The file header as well as Info header, used to read the bitmap file, is given below:-

```
Private Type
BITMAPFILEHEADER
bfType As Integer
bfSize As Long
bfReserved1 As Integer
bfReserved2 As Integer
bfOhFileBits As Long
End Type
```

```
Private Type
BITMAPINFOHEADER
biSize As Long
biWidth As Long
biHeight As Long
biplanes As Integer
biBitCount As Integer
biCompression As Long
biSizeImage As Long
biXPelsPerMeter As Long
biYPelsPerMeter As Long
biClrUsed As Long
biClrImportant As Long
End Type '40 bytes
```

The list of intermediate parameters, which are common to all algorithms, is specified here:-

Min_X - Defines the minimum x coordinate value for input image.

Max_X - Defines the maximum x coordinate value for input image.

Min_Y - Defines the minimum y coordinate value for input image.

Max_Y - Defines the maximum y coordinate value for input image.

Pixel_Intensity(X, Y) - Returns or Sets the pixel intensity at coordinates X and Y.

Pixel_Values - A list storing intensity values of all pixels.

Pixel_Values_Count - Defines count of values in list Pixel_Values.

The '//' symbol is used to represent the comments in the algorithms.

The procedure for Preprocessing with its input and output parameters is specified here:-

**P1: Preprocessing (Input_Image)**

*Input Parameters*:

A bitmap image, Input_Image, is given as input to the preprocessing algorithm.

*Output Parameters*:

A preprocessed bitmap image, say Output_Image, is produced as output from this algorithm.

```
Preprocessing(Input_Image)
    Call Modified_Median_Filter(Input_Image,Matrix_Size)
    // The function returns a bitmap image namely, Filter_Output_Image.
    Call Binarization(Filter_Output_Image)
    // The function returns a bitmap image namely, Binarization_Output_Image.
    Call Thinning(Binarization_Output_Image)
    // The function returns a bitmap image namely, Thinning_Output_Image.
    Call Smoothing(Thinning_Output_Image)
    // The function returns a bitmap image namely, Output_Image.
```

The image obtained after the application of the P1 algorithm to the Input_Image, is Output_Image, which is referred as figure 4.6 in the chapter 4.

The modules called by the Preprocessing procedure are discussed in the subsequent sections. To start with, Modified_Median_Filter module is discussed in the following section.

**P1.1: Modified_Median_Filter (Input_Image, Matrix_Size)**

For the purpose of noise removal from the acquired image, a technique named modified median filter is applied to the image. It works on the principle of median filter. A new parameter has been added to it, which assures the conditional application of the median filter. The pixel intensity value is modified according to the median filter, only if the number of occurrences of lowest intensity pixel in the neighbourhood of the current pixel equals the predefined parameter. This approach helps to avoid the removal of corners and blurring of the pixels in the document image. The input and output parameters to the Modified_Median_Filter are stated here:-

*Input Parameters*:

A bitmap image, Input_Image, is given as input to the Modified_Median_Filter algorithm.

An integer variable, say Matrix_Size, denotes the neighbourhood matrix size of the pixel. For *e.g.*, in $3 \times 3$ matrix, it has value as two.

*Output Parameters*:

A bitmap image, say Filter_Output_Image, is produced as output from this algorithm.

The pseudo code for Modified_Median_Filter is as shown here:-

```
Modified_Median_Filter (Input_Image, Matrix_Size)
Set A_Min=-(Matrix_Size)/2
Set A_Max= (Matrix_Size)/2
For X=Min_X to Max_X
{
    For Y=Min_Y to Max_Y
    {
        For X1=A_Min to A_Max
        {
            Set Temp_X=X+X1
```

```
                If (Temp_X>=Min_X and Temp_X<=Max_X)
                {
                    For Y1=A_Min to A_Max
                    {
                        Set Temp_Y=Y+Y1
                        If (Temp_Y>=Min_Y and Temp_Y<=Max_Y)
                        {
                        Add Pixel_Intensity (Temp_X, Temp_Y) to list Pixel_Values
                        }
                    }
                }
            }
            Sort the list Pixel_Values
            Set No_Occurences=Number of the occurrences of lowest pixel intensity value in list Pixel_Values
            If (No_Occurences==K)
                Median_value=Value at Pixel_Values_Count/2
            Set Pixel_Intensity(X, Y) =Median_Value
        }
}
Return Filter_Output_Image
```

*Intermediate Parameters*:

K - Parameter defined according to matrix size. For $3 \times 3$ matrix, it has the value as 1.

The filtered image, which is obtained as output after application of the Modified_Median_Filter algorithm, Filter_Output_Image, is referred as figure 4.3 in the next chapter. After the filtered image is obtained, it is required to convert the image into binary one using the Binarization module, which is discussed in the following section.

**P1.2: Binarization (Filter_Output_Image)**

The grayscale or color images are converted to the binary images by picking a threshold value as it reduces the memory requirements and increases the processing speed. The current algorithm uses a global threshold value, as stated in Kumar and Singh [2011]. The input and output parameters to the Binarization are stated here:-

*Input Parameters*:

A bitmap image, Filter_Output_Image, is given as input to the Binarization algorithm.

*Output Parameters*:

A bitmap image, say Binarization_Output_Image, is produced as output from this algorithm.

The pseudo code for Binarization is as shown here:-

```
Binarization (Filter_Output_Image)
For X=Min_X to Max_X
{
    For Y=Min_Y to Max_Y
    {
        Set Pixel_Intensity_Sum=Pixel_Intensity_Sum+Pixel_Intensity(X, Y)
        Set Pixel_Count=Pixel_Count+1
    }
}
Set Average_Intensity= Pixel_Intensity_Sum/ Pixel_Count
For X=Min_X to Max_X
{
    For Y=Min_Y to Max_Y
    {
        If (Pixel_Intensity(X, Y) >=Average_Intensity)
            Set Pixel_Intensity(X, Y) =WHITE
        Else
            Set Pixel_Intensity(X, Y) =BLACK
    }
}
Return Binarization_Output_Image
```

The binarized image, which is obtained as output after application of the Binarization algorithm, is Binarization_Output_Image, which is referred as figure 4.4 in the next chapter. The image is now fed to the Thinning module to perform the skeletonization of the image that is discussed in the following section.

### P1.3: Thinning (Binarization_Output_Image)

The text in the document image may have the thickness of several pixels depending on the type of the writing instrument. The reduction of thickness of text will reduce the amount of data and helps to extract the shape information of the character. The algorithm consists of two sub iterations – one deletes the northwest boundary points and other deletes the southeast corner points, as stated in Zhang and Suen [1984]. The input and output parameters to the Thinning module are specified here:-

*Input Parameters*:

A bitmap image, Binarization_Output_Image, is given as input to the Thinning algorithm.

*Output Parameters*:

A bitmap image, say Thinning_Output_Image, is produced as output from this algorithm.

The pseudo code for Thinning is as shown here:-

```
Thinning(Binarization_Output_Image)
Set counter=0
While(counter=0)
{
    For X=Min_X to Max_X
    {
        For Y=Min_Y to Max_Y
        {
            If(C1 is true)
            {
                Set Pixel_Intensity(X,Y)=WHITE
                Set counter=counter+1
            }
        }
    }
}
If(counter=0)
    Goto label A
Set counter=0
```

```
For X=Min_X to Max_X
{
    For Y=Min_Y to Max_Y
    {
        If(C1 is true)
        {
            Set Pixel_Intensity(X,Y)=WHITE
            Set counter=counter+1
        }
    }
}
If(counter=0)
    Goto label A
A: return Thinning_Output_Image
```

*Intermediate Parameters*:

C1 and C2 define the set of conditions as described here:-

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

Figure 3.1: Neighborhood Matrix of Pixel (P1)

C1 has the following set of conditions:-

1. 2 <= B (P1) <= 6
2. A (P1) =1
3. P2*P4*P6=0
4. P4*P6*P8=0

C2 has the following set of conditions:-

1. 2 <= B (P1) <= 6
2. A (P1) =1
3. P2*P4*P8=0

4. P2*P6*P8=0

The parameters used in the conditions C1 and C2 are defined here:-

A (P1) defines the number of 01 transitions in the ordered set P2, P3, P4…..P9.

B (P1) defines the number of non zero neighbours of P1.

B (P1) = P2+P3+P4……+P9.

The thinned image, which is obtained as output after application of the Thinning algorithm, is Thinning_Output_Image, which is referred as figure 4.5 in the next chapter. The image boundaries are required to be smoothened out in order to improvise the segmentation. This is done using the smoothing module, which is discussed in the following section.

**P1.4: Smoothing (Thinning_Output_Image)**

The preprocessing techniques applied till now may ruin the continuity in the boundary of the image; hence, there is a need to make the boundaries of character, a continuous one. For this purpose, Smoothing algorithm is applied to the output image of previous algorithm. The input and output parameters of the Smoothing module are listed here:-

*Input Parameters*:

A bitmap image, Thinning_Output_Image, is given as input to the Smoothing algorithm.

*Output Parameters*:

A bitmap image, Output_Image, is produced as output from this algorithm.

The pseudo code for Smoothing is as shown here:-

```
Smoothing(Thinning_Output_Image)
Set Parameter_X=Predefined value
Set Parameter_y=Predefined value
For X=Min_X to (Max_X-Parameter_X)
{
    For Y=Min_Y to Max_Y
    {
        Set P1=Pixel_Intensity(X,Y)
```

```
            Set P2=Pixel_Intensity(X+Parameter_X,Y)
            If(P1=P2)
            {
                  For I=1 to Parameter_X
                        Set Pixel_Intensity(X+I,Y)=BLACK
            }
      }
}
For Y=Min_Y to (Max_Y-Parameter_Y)
{
      For X=Min_X to Max_X
      {
            Set P3=Pixel_Intensity(X,Y)
            Set P4=Pixel_Intensity(X,Y+Parameter_Y)
            If(P3=P4)
            {
                  For I=1 to Parameter_Y
                        Set Pixel_Intensity(X,Y+I)=BLACK
            }
      }
}
Return Output_Image
```

The image, which is obtained as output after application of the Smoothing algorithm, is Output_Image, which is referred as figure 4.6 in the next chapter.

The clean image obtained from the preprocessing stage acts as input to the segmentation stage, which mainly decomposes the document into its sub components. This task is performed by the Segmentation algorithm, which is discussed in the following section.

### 3.2.2 Segmentation

Segmentation basically deals with the decomposition of the document into sub components like lines, words and characters. The extent of accuracy of segmentation directly impacts the accuracy of recognition; hence, segmentation is very important stage of the handwritten numeral recognition process. The algorithm to perform segmentation is described in this section.

**S1: Segmentation (Output_Image)**

The Segmentation algorithm calls three modules to segment the document into lines, words and characters. Firstly, the image is segmented into lines using the

Line_Segmentation algorithm. Then, the words are extracted for each line using the Word_Segmentation algorithm. Finally, the characters are extracted from the words by the application of the Character_Segmentation algorithm. These algorithms basically explore the specific gaps present between the different sub components of the document. The details of these algorithms are worked out in the following section. The input and output parameters for the Segmentation algorithm are stated here:-

*Input Parameters*:

A bitmap image,Output_Image, is given as input to the segmentation algorithm.

*Output Parameters*:

A text file, say Points.txt, containing the line coordinates, word coordinates and character coordinates is generated from this algorithm. The Points.txt has the coordinates stored in the form of L1 W1 C1 C2… W2 C1.. L2 W1 C1… W2 C1…,where L denotes a line, W denotes a word and C denotes a character.

The pseudo code for the Segmentation is as shown here:-

```
Segmentation(Output_Image)
    Call Line_Segmentation(Output_Image)
    // A text file namely, Line_Points.txt , containing the line coordinates is generated from this algorithm.
    Call Word_Segmentation(Output_Image, Line_Points)
    // A text file namely, Word_Points.txt , containing the line coordinates and word coordinates is generated from this algorithm.
    Call Character_Segmentation(Output_Image, Word_Points)
    // A text file namely, Points.txt, containing the line coordinates, word coordinates and character coordinates is generated from this algorithm.
```

**S1.1: Line_Segmentation (Output_Image)**

This algorithm decomposes the document into lines, as stated in Kumar and Singh [2011]. It takes the preprocessed image, Output_Image as input image and generates a text file, Line_Points.txt as output, which contains the coordinates of the lines present in the document.

The image, which illustrates the lines obtained after application of the Line_Segmentation algorithm, is Lines_Image, which is referred as figure 4.7 in the next chapter. The line coordinates are now used to extract the words from each line using the Word_Segmentation algorithm that is discussed in the following section.

**S1.2: Word_Segmentation (Output_Image, Line_Points)**

This algorithm decomposes each line into words, as stated in Kumar and Singh [2011]. It takes the preprocessed image, Output_Image as input image and a text file, Line_Points.txt as input file, which contains the coordinates of the lines present in the document. It generates a text file, say Word_Points.txt as output, which contains the coordinates of the lines and coordinates of words in each line present in the document.

The image, which illustrates the words obtained after application of the Word_Segmentation algorithm, is Words_Image, which is referred as figure 4.8 in the next chapter. The line and word coordinates aid in the extraction of the characters present in each word with the help of Character_Segmentation algorithm, which is discussed in the following section.

**S1.3: Character_Segmentation (Output_Image, Word_Points)**

The words obtained as the output of the previous algorithm are now required to be segmented into characters. The Character_Segmentation algorithm performs this task, which works on basis of specific gap present between the characters. The input and output parameters for the Character_Segmentation algorithm are specified here:-

*Input Parameters*:

A bitmap image, Output_Image, is given as input to the character segmentation algorithm.

A text file, Word_Points.txt, containing the line coordinates and word coordinates is also given as input to this algorithm.

*Output Parameters*:

A text file, say Points.txt, containing the line coordinates, word coordinates and character coordinates is generated from this algorithm. The pseudo code for the Character_Segmentation is as shown here:-

```
Character_Segmentation(Output_Image,Word_Points)
While(End_Of_File is not true)
{
    For each line read
    {
        Read word coordinates and store them in WMin_X, WMax_X, WMin_Y and WMax_Y
        For X=WMin_X to WMax_X
        {
            Set countpixels=0
            For Y=WMin_Y to WMax_Y
            {
                If(Pixel_Intensity(X,Y) is BLACK)
                    Set countpixels=countpixels+1
            }
            Add countpixels to a list named as Pixel_Values
        }
        While(I< (Pixel_Values_Count-1) and Pixel_Values(I)<=0)
            Set I=I+1
        While(I<Pixel_Values_Count)
        {
            Set Start_Segment=I
            While(I< (Pixel_Values_Count-1) and Pixel_Values(I)>0)
                Set I=I+1
            Set End_Segment=I
            Write Start_Segment, End_Segment into the file
        }
    }
}
Return Points.txt
```

The image, which illustrates the characters obtained after application of the Character_Segmentation algorithm, is Characters_Image, which is referred as figure 4.9 in the next chapter.

The segmented characters obtained from the application of the segmentation algorithm are now required to be recognized and thus, acts as input for the recognition stage, which is discussed in the following section.

### 3.2.3 Recognition

Recognition refers to the process of finding out the predefined class to which the unknown sample belongs, in accordance with the script used. It basically consists of two phases, feature extraction and classification. Firstly, the required features are extracted from the images and then, based on those features, a classification structure is developed. The classification structure defines an appropriate class for each image based on the features present in the image. The features used in the current algorithm are presence of sidebars, presence of closed loops and presence of water reservoirs in the characters.

The present algorithm for recognition derives its basis from the concept explained in Kaur *et al.* [2010]. They have used the water reservoir analogy principle and various features to form a classification structure, which assigns the extracted fragment to a particular class. An effort has been to apply the same principle for the recognition stage.

The water reservoir can be defined as if water is poured from one side of a component, the cavity regions of the component where water will be stored are considered as reservoirs. The left (right) reservoir is the water stored cavity regions of the component, when water is poured from left (right) side of the component. Similarly, top and bottom reservoirs can be defined. The top, bottom, left, and right reservoir of numerals are illustrated in figure 3.2.

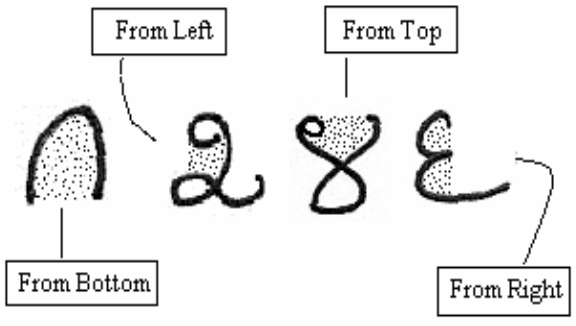


Figure 3.2: Water Reservoirs in Numerals

The classification structure used in the algorithm is shown in figure 3.3. The recognition algorithm based on these features and this classification structure, is presented in this section.

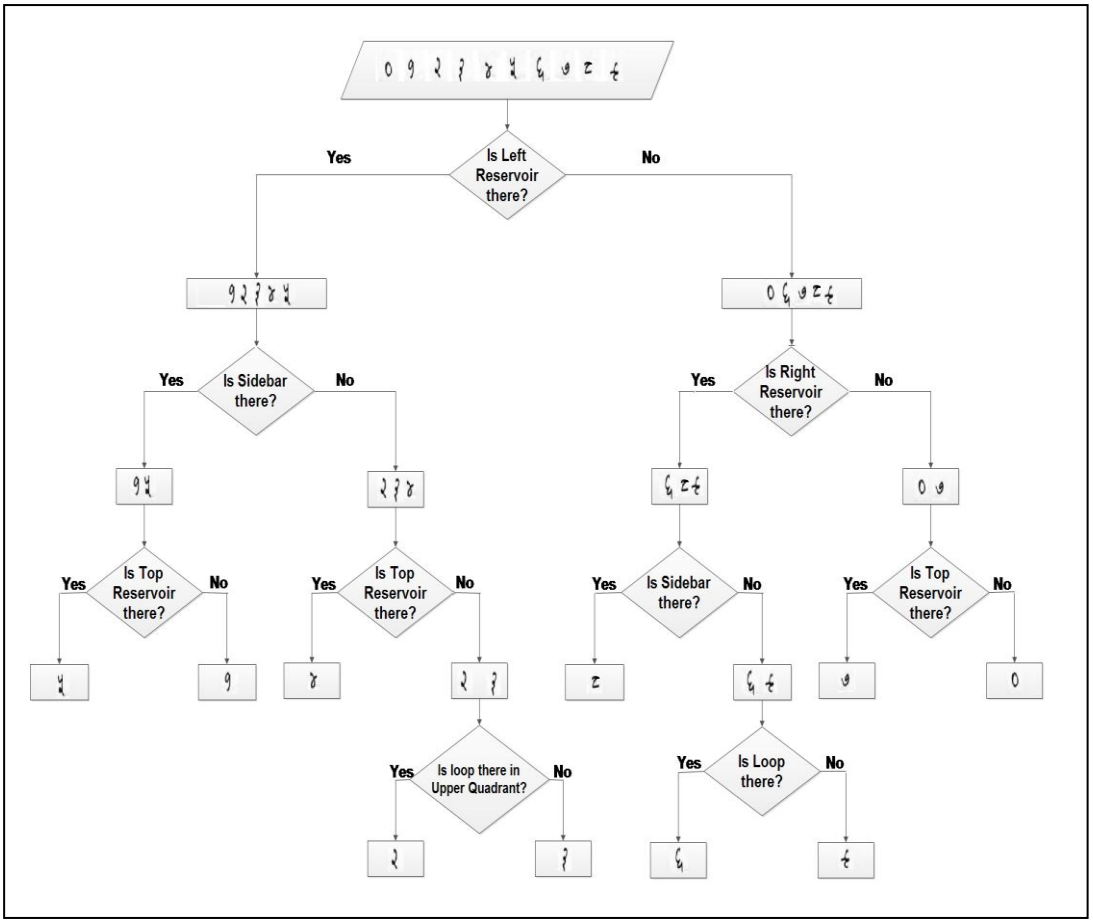


Figure 3.3: Classification Structure of Numerals

The Recognition algorithm with its input and output parameters is discussed in the following section.

**R1: Recognition (Output_Image, Points)**

The Recognition algorithm incorporates a number of steps as discussed here:-

Initially, the character segments are extracted from the input image based on the coordinates of the character, which are read from the file. Then, for each character, the listed steps are followed:-

a) The bounding box of the character image is created by using the Bounding_Box algorithm.

   // In order to extract the features from the character image, the steps are specified here.

b) The presence of the left reservoir is checked in the character image by using the Left_Reservoir_Detection algorithm.
c) The presence of the right reservoir is checked in the character image by using the Right_Reservoir_Detection algorithm.
d) The presence of the top reservoir is checked in the character image by using the Top_Reservoir_Detection algorithm.
e) The presence of the bottom reservoir is checked in the character image by using the Bottom_Reservoir_Detection algorithm.
f) The presence of the sidebar is checked in the character image by using the Sidebar_Detection algorithm.
g) The presence of the closed loop is checked in the character image by using the ClosedLoop_Detection algorithm.
h) The numerals are recognized by using the Numeral_Recognition algorithm, which uses the features extracted in previous steps and the classification structure shown in figure 3.3.

   // This algorithm writes the recognized numerals in a file, which creates an editable form of the input image.

The input and output parameters for the Recognition algorithm are listed here:-

*Input Parameters*:

A bitmap image, Output_Image, is given as input to the recognition algorithm.

A text file, Points.txt, containing the line coordinates, word coordinates and character coordinates is given as input to this algorithm.

*Output Parameters*:

A text file, say English_file, containing the recognized numerals in English is generated from this algorithm.

A text file, say Hindi_file, containing the recognized numerals in Hindi is generated from this algorithm.

The pseudo code for the Recognition algorithm is as shown here:-

```
Recognition (Output_Image, Points)
While(End_Of_File is not true)
{
    For each line read
    {
        For each word read
        {
            Set values of SMin_Y and SMax_Y
            For each character read
            {
                Set values of SMin_X and SMax_X
                Extract the image with dimensions as SMin_X , SMax_X, SMin_Y and SMax_Y
                from Output_Image and save it as bitmap image namely, Temp_Image
                Call Bounding_Box(Temp_Image,0,0,Temp_Image_Width-1,Temp_Image_Height-1)
                // The output bitmap image is saved as BoundingBox_Image.
                Call Left_Reservoir_Detection(BoundingBox_Image,0,0, BoundingBox_Image _Width-1,
                BoundingBox_Image _Height-1)
                // The output value is saved in Boolean type of variable named as Left.
                Call Right_Reservoir_Detection(BoundingBox_Image,0,0, BoundingBox_Image_Width-1,
                BoundingBox_Image _Height-1)
                // The output value is saved in Boolean type of variable named as Right.
                Call Top_Reservoir_Detection(BoundingBox_Image,0,0, BoundingBox_Image_Width-1,
                BoundingBox_Image _Height-1)
                // The output value is saved in Boolean type of variable named as Top.
                Call Bottom_Reservoir_Detection(BoundingBox_Image,0,0, BoundingBox_Image_Width-
                1, BoundingBox_Image _Height-1)
                // The output value is saved in Boolean type of variable named as Bottom.
                Call Sidebar_Detection(BoundingBox_Image,0,0, BoundingBox_Image _Width-1,
                BoundingBox_Image _Height-1)
                // The output value is saved in Boolean type of variable named as Sidebar.
                Call ClosedLoop_Detection(BoundingBox_Image,0,0, BoundingBox_Image _Width-1,
                BoundingBox_Image _Height-1)
                // The output value is saved in Boolean type of variable named as Loop and UpperQuad.
                Call Numeral_Recognition(Left,Right,Top,Bottom,Sidebar,Loop,UpperQuad)
                // The output value is saved in integer type of variable named as Numeral.
                Write the Numeral in English in English_File and Numeral in Hindi in Hindi_File
            }
        }
    }
}
Return English_File and Hindi_File
```

The modules, which are called in the Recognition algorithm, are discussed in the following section starting with the Bounding_Box algorithm.

**R1.1: Bounding_Box (Temp_Image, Min_X, Min_Y, Max_X, Max_Y)**

In order to extract the features from an image, it is required that the extra white space should be removed from the character image. This algorithm creates a bounding box by removing the extra white space around the extracted character.

*Input Parameters*:

A bitmap image, say Temp_Image with the dimensions specified by Min_X, Min_Y, Max_X and Max_Y is given as input to the Bounding_Box algorithm.

*Output Parameters*:

A bitmap image, say BoundingBox_Image, is produced as output from this algorithm.

The pseudo code for Bounding_Box is as shown here:-

```
Bounding_Box(Temp_Image,Min_X,Min_Y,Max_X,Max_Y)
For Y=Min_Y to Max_Y
{
    Set countpixels=0
    For X=Min_X to Max_X
    {
        If(Pixel_Intensity(X,Y) = BLACK)
            Set countpixels = countpixels + 1
    }
    Add countpixels to a list namely, Pixel_Values_Y
}
Set Y_Top_Temp=0
For I=0 to Pixel_Values_Y_Count-1
{
    If(Pixel_Values_Y(I)<1)
    {
        Set I=I+1
    }
    Else
    {
        Set Y_Top_Temp=I
        Goto Label A
    }
}
```

```
A: Set Y_Top=Min_Y+Y_Top_Temp
Set Y_Bottom_Temp=0
For I=Pixel_Values_Count-1 to 0
{
    If(Pixel_Values_Y(I)<1)
    {
        Set I=I-1
    }
    Else
    {
        Set Y_Bottom_Temp=I
        Goto Label B
    }
}
B: Set Y_Bottom=Min_Y+Y_Bottom_Temp
Create a new bitmap image with dimensions as Min_X, Y_Top,Max_X,Y_Bottom and save it as BoundingBox_Image
Return BoundingBox_Image
```

The image, which illustrates the bounded boxed characters obtained after application of the Bounding_Box algorithm, is BoundingBox_Image, which is referred as figure 4.10 in the next chapter.

Now, the features are extracted from the character image using the algorithms discussed in the following section.

**R1.2: Left_Reservoir_Detection (BoundingBox_Image, Min_X, Min_Y, Max_X, Max_Y)**

This algorithm detects the presence of left reservoir in the character. It works on the analysis of the pattern of the white space area on the left side of the character.

*Input Parameters*:

A bitmap image, BoundingBox_Image with the dimensions specified by Min_X, Min_Y, Max_X and Max_Y is given as input to the Left_Reservoir_Detection algorithm.

*Output Parameters*:

A Boolean variable, say Left, is given as output from this algorithm.

The pseudo code for Left_Reservoir_Detection is as shown here:-

```
Left_Reservoir_Detection (BoundingBox_Image,Min_X,Min_Y,Max_X,Max_Y)
A: For Y=Min_Y to Max_Y
{
    Set countpixels=0
    For X=Min_X to Max_X
    {
        If(Pixel_Intensity(X,Y) = WHITE)
            Set countpixels = countpixels + 1
        Else
            Goto Label A
    }
    Add countpixels to a list namely, Pixel_Values
}
Set counter=0 and Increment_counter=0
For I=0 to Pixel_Values_Count-1
{
    Set counter=counter+1
    If(Pixel_Values(I) <= Pixel_Values(I+1))
    {
            Set Increment_counter= Increment_counter + 1
    }
    Else
    {
        If((counter < Pixel_Values_Count/2) or (Increment_counter < 2 and counter < Pixel_Values_Count/3))
        {
        Set Increment_counter=0
        }
        Else
        {
            Set I_Increment_End=I
            Goto Label B
        }
    }
}
B: If(Increment_counter > 2)
{
    Set Decrement_counter=0
    For  I=I_Increment_Counter to Pixel_Values_Count-1
    {
        If(Pixel_Values(I)  >= Pixel_Values(I+1))
        {
            Set Decrement_counter= Decrement_counter + 1
        }
        Else
        {
            Set I_Decrement_End=I
            Goto Label C
        }
    }
    C: If(Decrement_counter >= 2)
        Set Left= true
    Else
        Set Left=false
}
Return  Left
```

The image is now checked for the presence of the right reservoir using the Right_Reservoir_Detection algorithm, which is discussed in the following section.

**R1.3: Right_Reservoir_Detection (BoundingBox_Image, Min_X, Min_Y, Max_X, Max_Y)**

This algorithm detects the presence of right reservoir in the character. It works on the analysis of the pattern of the white space area on the right side of the character.

*Input Parameters*:

A bitmap image, BoundingBox_Image with the dimensions specified by Min_X, Min_Y, Max_X and Max_Y is given as input to the Right_Reservoir_Detection algorithm.

*Output Parameters*:

A Boolean variable, say Right, is given as output from this algorithm.

The pseudo code for Right_Reservoir_Detection is as shown here:-

```
Right_Reservoir_Detection (BoundingBox_Image,Min_X,Min_Y,Max_X,Max_Y)
A: For Y=Min_Y to Max_Y
{
    Set countpixels=0
    For X=Max_X to Min_X
    {
        If(Pixel_Intensity(X,Y) = WHITE)
            Set countpixels = countpixels + 1
        Else
            Goto Label A
    }
    Add countpixels to a list namely, Pixel_Values
}
Set counter=0 and Increment_counter=0
For I=0 to Pixel_Values_Count-1
{
    Set counter=counter+1
    If(Pixel_Values(I) <= Pixel_Values(I+1))
    {
            Set Increment_counter= Increment_counter + 1
    }
```

```
        Else
        {
            If((counter < Pixel_Values_Count/2) or (Increment_counter < 2 and counter < Pixel_Values_Count/3))
            {
            Set Increment_counter=0
            }
            Else
            {
                Set I_Increment_End=I
                Goto Label B
            }
        }
}
B: If(Increment_counter > 2)
{
        Set Decrement_counter=0
        For  I=I_Increment_Counter to Pixel_Values_Count-1
        {
            If(Pixel_Values(I)  >= Pixel_Values(I+1))
            {
                Set Decrement_counter= Decrement_counter + 1
            }
            Else
            {
                Set I_Decrement_End=I
                Goto Label C
            }
        }
        C: If(Decrement_counter >= 2)
            Set Right= true
        Else
            Set Right=false
}
Return  Right
```

The image is now checked for the presence of the top reservoir using the Top_Reservoir_Detection algorithm, which is discussed in the following section.

**R1.4: Top_Reservoir_Detection (BoundingBox_Image, Min_X, Min_Y, Max_X, Max_Y)**

This algorithm detects the presence of top reservoir in the character. It works on the analysis of the pattern of the white space area on the top side of the character.

*Input Parameters*:

A bitmap image, BoundingBox_Image with the dimensions specified by Min_X, Min_Y, Max_X and Max_Y is given as input to the Top_Reservoir_Detection algorithm.

*Output Parameters*:

A Boolean variable, say Top, is given as output from this algorithm.

The pseudo code for Top_Reservoir_Detection is as shown here:-

```
Top_Reservoir_Detection (BoundingBox_Image,Min_X,Min_Y,Max_X,Max_Y)
A: For X=Min_X to Max_X
{
    Set countpixels=0
    For Y=Min_Y to Max_Y
    {
        If(Pixel_Intensity(X,Y) = WHITE)
            Set countpixels = countpixels + 1
        Else
            Goto Label A
    }
    Add countpixels to a list namely, Pixel_Values
}
Set counter=0 and Increment_counter=0
For I=0 to Pixel_Values_Count-1
{
    Set counter=counter+1
    If(Pixel_Values(I) <= Pixel_Values(I+1))
    {
            Set Increment_counter= Increment_counter + 1
    }
    Else
    {
        If((counter < Pixel_Values_Count/2) or (Increment_counter < 2 and counter < Pixel_Values_Count/3))
        {
        Set Increment_counter=0
        }
        Else
        {
            Set I_Increment_End=I
            Goto Label B
```

```
            }
        }
    }
    B: If(Increment_counter > 2)
    {
        Set Decrement_counter=0
        For  I=I_Increment_Counter to Pixel_Values_Count-1
        {
            If(Pixel_Values(I)  >= Pixel_Values(I+1))
            {
                Set Decrement_counter= Decrement_counter + 1
            }
            Else
            {
                Set I_Decrement_End=I
                Goto Label C
            }
        }
        C: If(Decrement_counter >= 2)
            Set Top= true
        Else
            Set Top=false
    }
    Return  Top
```

The image is now checked for the presence of the bottom reservoir using the Bottom_Reservoir_Detection algorithm, which is discussed in the following section.

**R1.5: Bottom_Reservoir_Detection (BoundingBox_Image, Min_X, Min_Y, Max_X, Max_Y)**

This algorithm detects the presence of bottom reservoir in the character. It works on the analysis of the pattern of the white space area on the bottom side of the character. The classification structure developed for the recognition of the numerals does not depend on the bottom water reservoir feature. The algorithm for the detection of presence of bottom reservoir in the characters is discussed in this section.

*Input Parameters*:

A bitmap image, BoundingBox_Image with the dimensions specified by Min_X, Min_Y, Max_X and Max_Y is given as input to the Bottom_Reservoir_Detection algorithm.

*Output Parameters*:

A Boolean variable, say Bottom, is given as output from this algorithm.

The pseudo code for Bottom_Reservoir_Detection is as shown here:-

```
Bottom_Reservoir_Detection (BoundingBox_Image,Min_X,Min_Y,Max_X,Max_Y)
A: For X=Min_X to Max_X
{
    Set countpixels=0
    For Y=Max_Y to Min_Y
    {
        If(Pixel_Intensity(X,Y) = WHITE)
            Set countpixels = countpixels + 1
        Else
            Goto Label A
    }
    Add countpixels to a list namely, Pixel_Values
}
Set counter=0 and Increment_counter=0
For I=0 to Pixel_Values_Count-1
{
    Set counter=counter+1
    If(Pixel_Values(I) <= Pixel_Values(I+1))
    {
            Set Increment_counter= Increment_counter + 1
    }
    Else
    {
        If((counter < Pixel_Values_Count/2) or (Increment_counter < 2 and counter < Pixel_Values_Count/3))
        {
        Set Increment_counter=0
        }
        Else
        {
            Set I_Increment_End=I
             Goto Label B
         }
     }
}
B: If(Increment_counter > 2)
{
    Set Decrement_counter=0
```

```
        For  I=I_Increment_Counter to Pixel_Values_Count-1
        {
              If(Pixel_Values(I)  >= Pixel_Values(I+1))
              {
                    Set Decrement_counter= Decrement_counter + 1
              }
              Else
              {
                    Set I_Decrement_End=I
                    Goto Label C
              }
        }
        C: If(Decrement_counter >= 2)
              Set Bottom= true
        Else
              Set Bottom=false
}
Return  Bottom
```

**R1.6: Sidebar_Detection (BoundingBox_Image, Min_X, Min_Y, Max_X, Max_Y)**

This algorithm detects the presence of sidebar in the character. It works on the basis of presence of contiguous length of black pixels in the character.

*Input Parameters*:

A bitmap image, BoundingBox_Image with the dimensions specified by Min_X, Min_Y, Max_X and Max_Y is given as input to the Sidebar_Detection algorithm.

*Output Parameters*:

A Boolean variable, say  Sidebar, is given as output from this algorithm.

The pseudo code for Sidebar_Detection is as shown here:-

```
Sidebar_Detection (BoundingBox_Image,Min_X,Min_Y,Max_X,Max_Y)
A: For X=Min_X to Max_X
{
      Set countpixels=0
      For Y=Min_Y to Max_Y
      {
            If(Pixel_Intensity(X,Y) = BLACK)
                  Set countpixels = countpixels + 1
```

```
        }
        Add countpixels to a list namely, Pixel_Values
}
Set Max_Density=0 and X_Value_Max_Density=0
For I=0 to Pixel_Values_Count-1
{
        If(Max_Density<Pixel_Values(I))
        {
                Max_Density=Pixel_Values(I)
                X_Value_Max_Density=I
        }
}
Set Flag=0 and D=0
For Y=Min_Y to Max_Y
{
        If(Pixel_Intensity(X_Value_Max_Density,Y) =BLACK)
        {
                While(Y<= Max_Y and Pixel_Intensity(X_Value_Max_Density,Y) =BLACK)
                {
                        If(Flag=0)
                                Set Start_Y=Y and Flag=1
                        Else
                                Set End_Y=Y
                        Set Y=Y+1
                }
        }
        Set D=End_Y - Start_Y and Flag = 0
        If(D<= (End_Y-Start_Y))
        {
                Set Final_Start_Y=Start_Y
                Set Final_End_Y=End_Y
        }
}
If((Final_End_Y-Final_Start_Y) >= (0.5 *(Max_Y-Min_Y))
        Set Sidebar=true
Else
        Set Sidebar=false
Return Sidebar
```

### R1.7: ClosedLoop_Detection (BoundingBox_Image, Min_X, Min_Y, Max_X, Max_Y)

This algorithm detects the presence of closed in the character. It basically calls the other modules to find the candidate starting points of the loop and to detect the loop. It also checks if the loop lies in the upper quadrant of the image.

*Input Parameters*:

A bitmap image, BoundingBox_Image with the dimensions specified by Min_X, Min_Y, Max_X and Max_Y is given as input to the ClosedLoop_Detection algorithm.

*Output Parameters*:

A Boolean variable, say Loop, is given as output from this algorithm.

A Boolean variable, say UpperQuad, is given as output from this algorithm.

The pseudo code for ClosedLoop_Detection is as shown here:-

```
ClosedLoop_Detection (BoundingBox_Image,Min_X,Min_Y,Max_X,Max_Y)
    Call Candidate_Starting_Points(BoundingBox_Image,Min_X,Min_Y,Max_X,Max_Y)
    // A list named as Starting_Points is generated by this algorithm.
    While(All points, say SP, in Starting_Points are not checked)
    {
        Set Start_X=SP.X ; Start_Y=SP.Y ; End_X=SP.X ; End_Y=SP.Y ;
        Call Find_Loop(Start_X,Start_Y)
        // A Boolean variable named as Loop is returned by this algorithm
        If(Loop=true)
        {
            If(Start_Y <= (Max_Y-Min_Y)/4
                Set UpperQuad=true
            Else
                Set UpperQuad=false
        }
    }
Return Loop and UpperQuad
```

### R1.7.1: Candidate_Starting_Points (BoundingBox_Image, Min_X, Min_Y, Max_X, Max_Y)

This algorithm identifies those points which can act as the candidate points for the starting point of the loop in the characters. It accomplishes its purpose by checking the presence of white black transitions in a particular width of the image. The steps listed here are more algorithmic.

*Input Parameters*:

A bitmap image, BoundingBox_Image with the dimensions specified by Min_X, Min_Y, Max_X and Max_Y is given as input to the Candidate_Starting_Points algorithm.

*Output Parameters*:

A list, say Starting_Points, is generated by this algorithm.

```
Candidate_Starting_Points (BoundingBox_Image,Min_X,Min_Y,Max_X,Max_Y)
For Y=Min_Y to Max_Y
{
    For X=Min_X to Max_X
    {
        If(White_Black_White_Black Transition exists in pixel intensities)
            Add point(X,Y) to Starting_Points
        If(Black_White_Black Transition exists in pixel intensities)
            Add point(X,Y) to Starting_Points
    }
}
Return Starting_Points
```

**R1.7.2: Find_Loop (Start_X, Start_Y)**

This algorithm detects the loop if it reaches the starting point after traversing the neighbours of current pixel in a particular direction. The steps specified here has an algorithmic nature.

*Input Parameters*:

The candidate starting points of the loop stored in Start_X and Start_Y are given as the input to the Find_Loop algorithm.

*Output Parameters*:

A Boolean variable, Loop is returned by the algorithm

*Pre Requisite*

Consider a global variable Loop_Entry=0

The variables End_X and End_Y in ClosedLoopDetection are shared with Find_Loop algorithm.

```
Find_Loop(Start_X, Start_Y)
If(Loop_Entry=1)
    Add point to the list named Visited_Pixels
If(Loop_Entry=1 and Start_X=End_X and Start_Y=End_Y)
    Set Loop=true
Else
    Set Loop=false
Return Loop
Set Loop_Entry=1
For 3x3 neighborhood matrix of Start_X, Start_Y, traversing them ,say Neighbor_Point,  in clockwise direction
{
    If(Neighbor_Point is not in list Visited_Pixels)
    {
        Set Previous_X=Start_X and Previous_Y=Start_Y
        Set Neighbor=true
        Call Find_Loop(Neighbor_Point.X, Neighbor_Point.Y)
    }
    Else
        Set Neighbour=false
}
If(Neighbor=false)
{
    If(Loop_Entry=1)
        Add point to the list named Visited_Pixels
    If(Loop_Entry=1 and Previous_X=End_X and Previous_Y=End_Y)
        Set Loop=true
    Else
        Set Loop=false
}
Return Loop
```

**R1.8: Numeral_Recognition (Left, Right, Top, Bottom, Sidebar, Loop, UpperQuad)**

The features that are extracted for an image are now used to assign a particular class to the input image based on the classification structure shown in figure 3.3. This algorithm classifies the input image based on the extracted features.

*Input Parameters*:

A Boolean variable, Left, is given as input to Numeral_Recognition algorithm.

A Boolean variable, Right, is given as input to this algorithm.

A Boolean variable, Top, is given as input to this algorithm.

A Boolean variable, Bottom, is given as input to this algorithm.

A Boolean variable, Sidebar, is given as input to this algorithm.

A Boolean variable, Loop, is given as input to this algorithm.

A Boolean variable, UpperQuad, is given as input to this algorithm.

*Output Parameters*:

An Integer variable, say Numeral, is given as output from this algorithm.

```
Numeral_Recognition(Left,Right,Top,Bottom,Sidebar,Loop,UpperQuad)
If (Left = true)
{
    If(Sidebar=true)
    {
        If(Top=true)
        {
            Set Numeral=5
        }
        Else
        {
            Set Numeral=1
        }
    }
    Else
    {
        If(top=true)
```

```
        {
        Set Numeral=4
        }
        Else
        {
            If( loop =true)
            {
                If(UpperQuad=true)
                {
                    Set Numeral=2
                }
                Else
                {
                    Set Numeral=3
                }
            }
        }
    }
}
Else
{
    If(Right=true)
    {
        If(Sidebar=true)
        {
            Set Numeral=8
        }
        Else
        {
            If(Loop=true)
            {
                Set Numeral=6
            }
            Else
            {
                Set Numeral=9
            }
        }
    }
    Else
    {
        If(Top=true)
        {
            Set Numeral=7
        }
        Else
        {
            Set Numeral=0
        }
    }
}
Return Numeral
```

The proposed algorithms have been implemented and the results are discussed in the next chapter.

# CHAPTER 4

# RESULTS AND DISCUSSIONS

The proposed work discussed in chapter 3 has been implemented by the authors in the .NET framework using C# language. The Drawing namespace and Bitmap class present in the inbuilt libraries of C# is utilised to process the bitmap image. The algorithms proposed are implemented and tested on several documents. The current section describes the outputs obtained for the stepwise execution of the algorithms. The results obtained are also discussed in this section.

## 4.1 Results

The initial step is to capture the document image through scanner. The captured image is shown in figure 4.1. It can be seen that noise is present in the scanned document image, which can be due to factors like varying paper quality, varying ink quality and scanner quality. This noise can interfere in the working of consequent algorithms; hence, it requires preprocessing, which is performed using P1 algorithm.

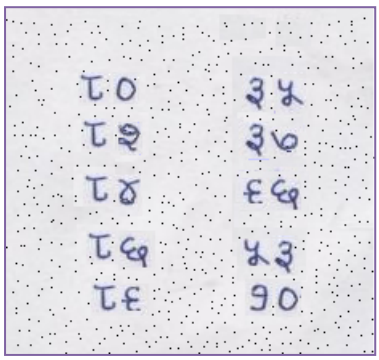

Figure 4.1: Input_Image

The first step for preprocessing is modified median filtering that is applied to the scanned image using the P1.1 algorithm. The filtering module is executed and the resultant image is shown in figure 4.3. In order to illustrate the improvement by modified median filtering, the median filter is applied to input image and the output image is shown in figure 4.2. Although, the median filter is capable of removing the noise, but it blurs the text that makes it difficult for the subsequent stages to achieve their purpose. It can be observed from the output of the modified median filter that the

text has not been blurred and the resultant image shows improvement in clarity of relevant pixels. There is presence of slightly textured background in the image as seen in figure 4.3. Since, this can interfere in the accurate working of subsequent stages; it needs to be removed and can be removed through the next step.

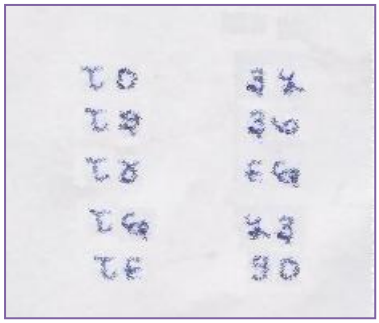

Figure 4.2: Median Filter Output Image

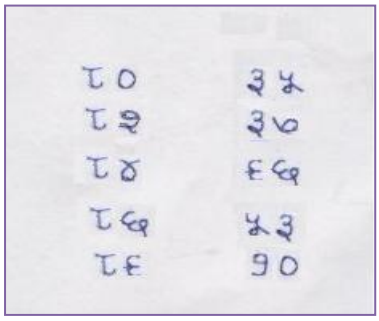

Figure 4.3: Filter_Output_Image

The colored images are required to be converted into binary images to reduce the amount of data, so, the next step is binarization. It is applied using P1.2 to the filtered image shown in figure 4.3 and the resultant image is shown in figure 4.4. It can be observed that the textured and coloured background present in the figure 4.3 is now removed.

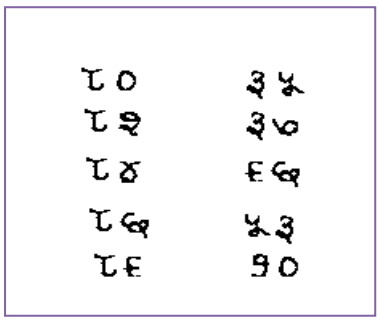

Figure 4.4: Binarization_Output_Image

The text in the binarized document image have the thickness of several pixels, as in figure 4.4 and this thickness is required to be removed to reduce the size and to lessen the complexity of algorithms of further stages. This task is performed by the application of P1.3 algorithm on the binarized image in figure 4.4 and the resultant image is shown in figure 4.5.

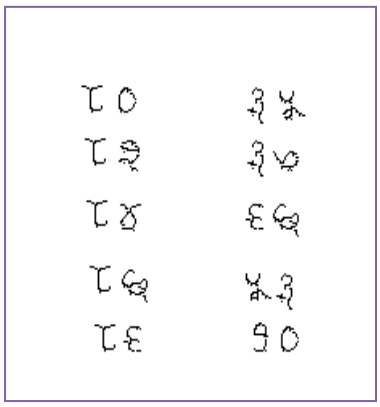

Figure 4.5: Thinning_Output_Image

It can be observed from figure 4.5 that the image boundaries have broken up, hence, in order to make these boundaries, a continuous one, smoothing is applied to the thinned image in figure 4.5 as stated in P1.4 and the output image is shown in figure 4.6.

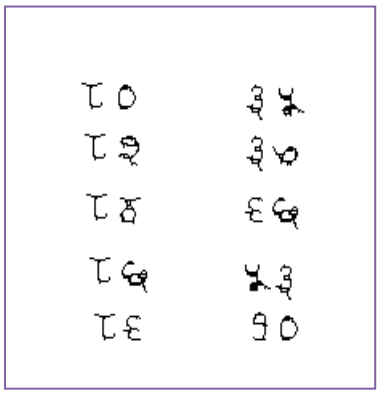

Figure 4.6: Output_Image

The preprocessed image is ready to act as input to the next stage that is segmentation. Firstly, the image is segmented into the lines using S1.1 algorithm and the segmented image is as shown in figure 4.7. The coordinates of the lines are represented by windows in the image.

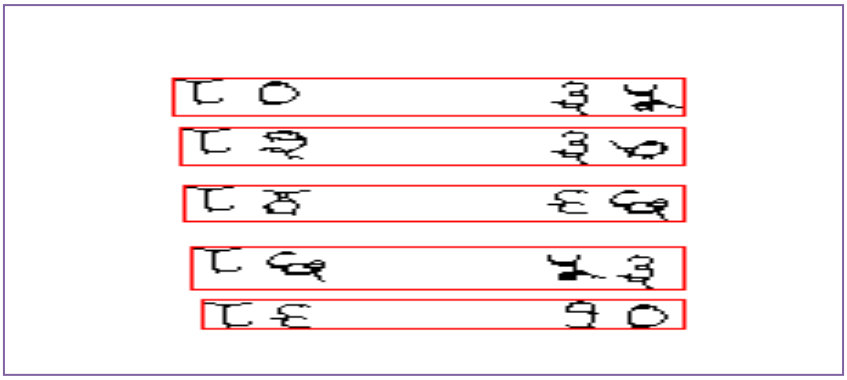

Figure 4.7: Lines_Image

The next step for segmentation is to extract the words from the lines. For this purpose, the S1.2 algorithm is used. The words are extracted from the lines extracted in the previous step and the output is as shown in figure 4.8.

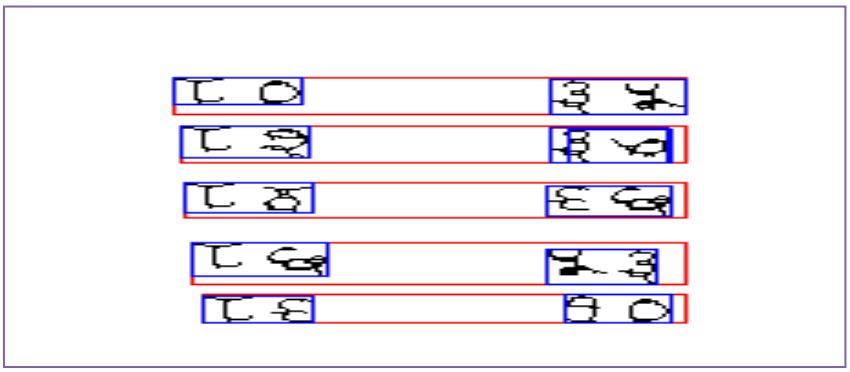

Figure 4.8: Words_Image

Lastly, the extracted words are to be segmented into the characters. The characters are extracted from the words using S1.3 algorithm and the resultant image is as shown in figure 4.9.

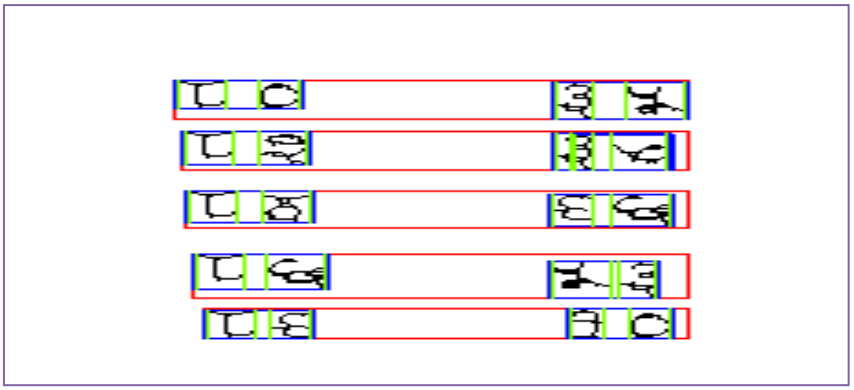

Figure 4.9: Characters_Image

The segmented characters act as input for the recognition stage. In this algorithm, firstly the bounding box is created for each character and saved as a separate bitmap file. The bounding box is created around the image of the extracted character using the R1.1 algorithm and the output images are as shown in figure 4.10.

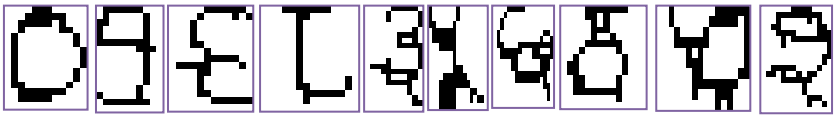

Figure 4.10: BoundingBox_Image

The next step is to check each image for the presence of the features, namely, sidebar, loop and water reservoirs using the algorithms R1.2 to R1.7. The extracted features are now used by the classification structure to recognize the numerals as discussed in chapter 3. This is performed through application of algorithm R1.8. The recognized numerals are stored in an editable form in a document.

The explained procedure has been applied to several documents and satisfactory results are obtained by the authors. The results obtained after implementation of all the algorithms are discussed in the following section.

The algorithm for the recognition of the handwritten Devnagari numerals has been applied on a number of documents. The results showing the number of numerals present in the document and the number of numerals recognized accurately has been tabulated in table 4.1.

| Document | Numerals Present | Numerals Recognized | Accuracy (%) |
|---|---|---|---|
| Doc1 | 10 | 10 | 100 |
| Doc2 | 60 | 58 | 96.67 |
| Doc3 | 20 | 20 | 100 |
| Doc4 | 30 | 29 | 96.67 |
| Doc5 | 25 | 24 | 96 |

Table 4.1: Recognition Results for Handwritten Devnagari Numerals

The comparison between the accurately recognized and inaccurately recognized numerals has been made through graph shown in figure 4.11.

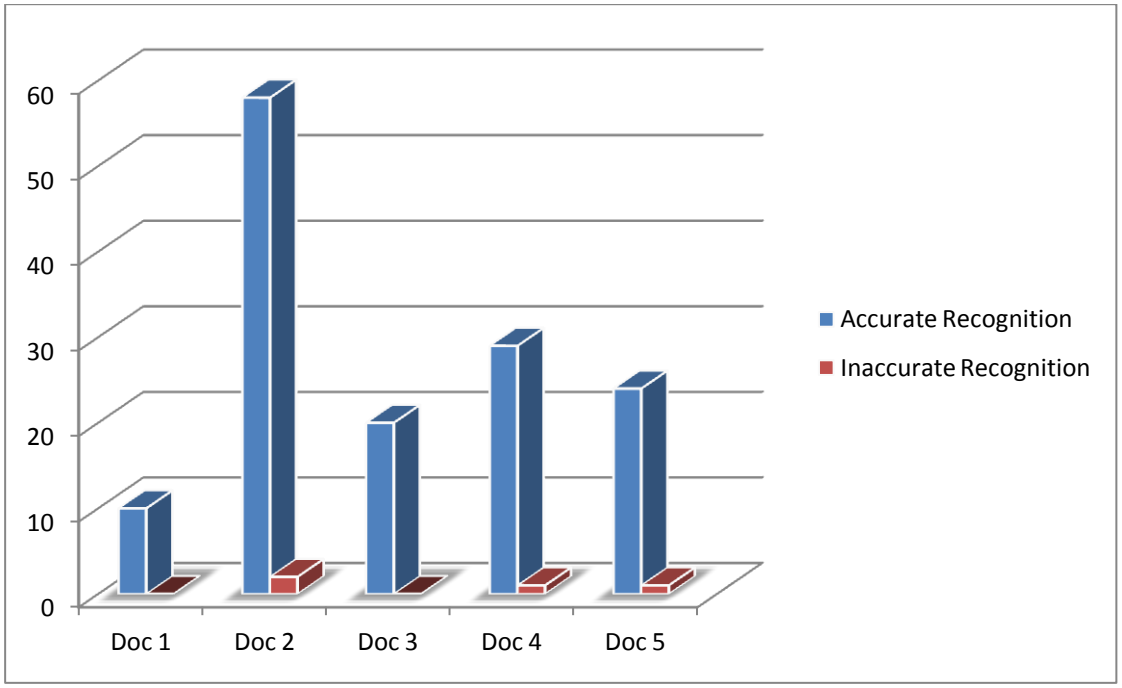


Figure 4.11: Comparison of Accurately Recognized/Inaccurately Recognized Numerals

The comparison between the accurately recognized and total numerals has been made through graph shown in figure 4.12.

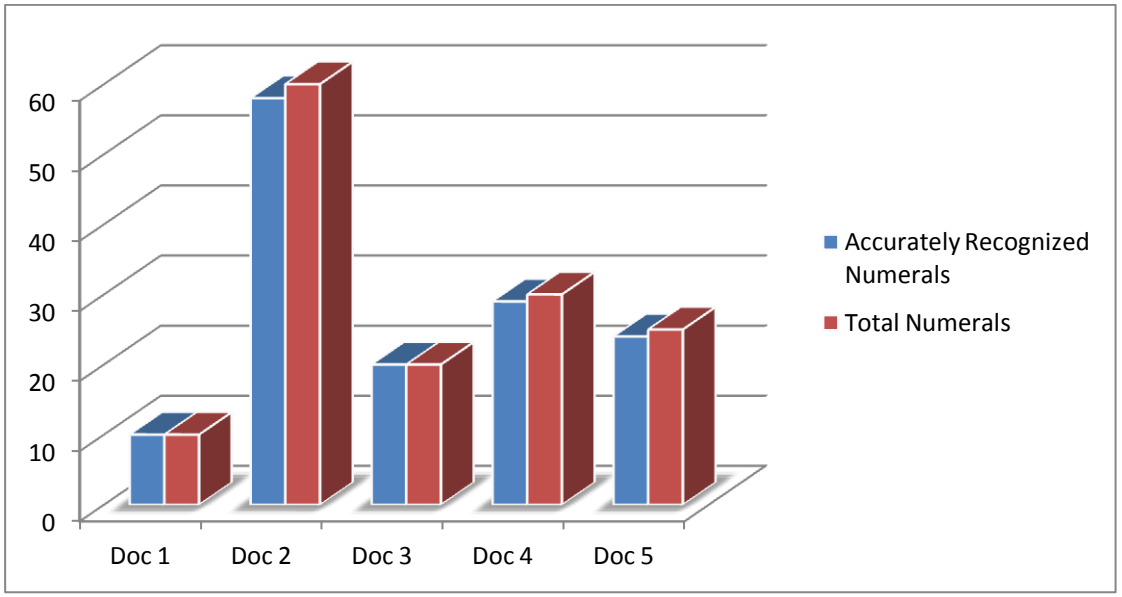


Figure 4.12: Comparison of Accurately Recognized/Total Numerals

The comparison between recognition accuracy of the numerals has been made through graph shown in figure 4.13.

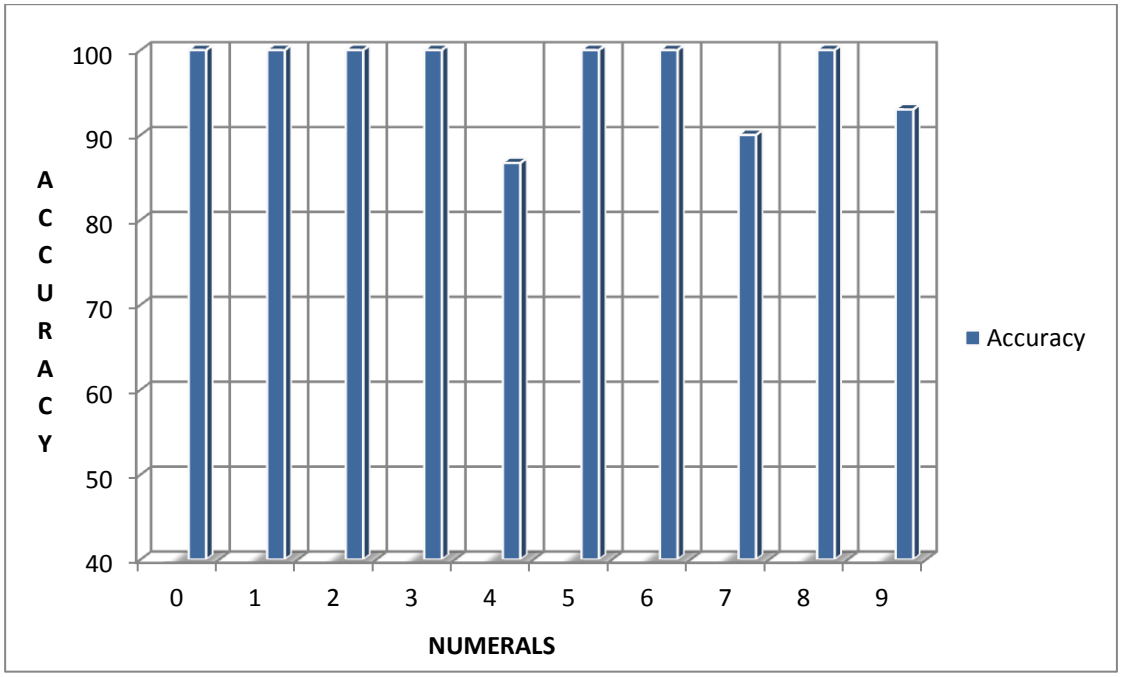


Figure 4.13: Comparison of Recognition Accuracy of Numerals

Thus, the results are illustrated through various measures to get a clear picture of the performance of the proposed work.

# CHAPTER 5

# CONCLUSION AND FUTURE SCOPE

## 5.1 Conclusion

Character recognition is a process, which associates a symbolic meaning with objects such as letters, symbols and numbers drawn on an image, according to the script used. The practical importance of the OCR applications and the challenging nature of the OCR process have led to huge amount of research in this field. It can be observed from the literature review that many advances have been made for the foreign languages in comparison with the Indian languages. Moreover, more than 300 million people use Devnagari script for documentation in central and northern parts of India. Hence, a character recognition approach for Devnagari has been proposed. The numerous approaches, which have been developed for the character recognition process, are based on the neural networks, which need high computational power and are testing dependent. Thus, an effort has been made to develop a simple approach for the offline handwritten Devnagari numeral recognition.

The acquired image is initially, cleaned up using the techniques of filtering, binarization, thinning and smoothing. For the purpose of filtering, modified median filter has been used with works on the principle of conditional application of the median filter. The median filter is capable of removing the noise but it blurs the text and removes the corners from the text, which makes it difficult for the further stages of the character recognition process to achieve their objectives easily. Thus, the modified median filter is able to overcome these disadvantages of the median filter. It does not blur the text and increase the clarity of the relevant pixels in the image. Since, the extent to which an image is cleaned directly affects the accuracy of the recognition; this technique improvises the image over the median filter.

The preprocessed image is segmented using the procedures described in Kumar and Singh [2011]. The image is decomposed into lines using the line segmentation techniques based on the horizontal projection histogram of the image. The lines are, then segmented into words through word segmentation technique based on the vertical projection histogram of the image. The words are decomposed into numerals

using the character segmentation technique based on the specific gap between the characters. The results for the segmentation are quite good and enhances the recognition rate.

The segmented numerals are recognized using the technique based on feature extraction and classification. The recognization technique derives its basis from the technique used for the segmentation of Gurmukhi characters by Kaur *et al.* [2010]. The features extracted for each image are closed loop, sidebars and water reservoirs. Based on these features, a classification structure is created, which assigns each segmented character to a predefined class. This approach uses features, which are simple to compute. Since, there are 10 numerals in Devnagari script, there will be 10 comparisons to recognize a numeral in case a linear structure is followed. But, the height of the classification structure used here is 4 and thus, at most 4 comparisons are required to recognize a numeral. This increases the efficiency of the approach. It does not require much computational power and does not require any kind of training.

The various applications where numeral recognition can be used are reading details like postal zip code, employee code, passport number and processing of forms and bank cheques. It can also be utilized in schools and colleges to process the list of marks. This work can be enhanced to other extents as discussed in the following section.

## 5.2 Future Scope

This approach can be extended in several dimensions which are as listed here:-

- The proposed algorithm can be worked upon for the recognition of the Devnagari alphabets.
- The classification structure can be used as the basis for the development of similar structures for other Indian languages, which follow the Devnagari script.
- It is observed that the current approach works for the well-formed characters, so, it can be improvised to work with the degraded documents.
- It can also be improvised to accommodate the variation in the styles of handwriting.
- It can also be used for the skewed documents, where skew removal technique can be used in conjunction with the current work.

# REFERENCES


1. Agrawal, P., Hanmandlu, M. and Lall, B. 2009. Coarse Classification of Handwritten Hindi Characters. *International Journal of Advanced Science and Technology*, 10, 43-54.

2. Arica, N. and Vural, F. Y. 2001. An Overview of Character Recognition focused on Off-line Handwriting. *IEEE Transactions on Systems, Man, and Cybernetics*, 31 (2), 216-233.

3. Arora, S., Bhatcharjee, D., Nasipuri, M. and Malik, L., A Two Stage Classification Approach for Handwritten Devanagari Characters. in *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications*, (2007), 377-403.

4. Banashree, N. P. and Vasantha, R., OCR for Script Identification of Hindi Numerals Using Feature Sub Selection by Means of End Point With Neuromememtic Model. in *Proceedings of the World Academy of Science, Engineering and Technology*, (2007), 78-82.

5. Bansal, V. and Sinha, R. M. K. 2001. Segmentation of Touching and Fused Devanagari Characters. *Pattern Recognition*, 35 (4), 875-893.

6. Benne, R. G., Dhandra, B. V. and Mallikarjun, H. 2009. Tri-Scripts Handwritten Numeral Recognition. *Advances in Computational Research*, 1 (2), 47-51.

7. Blumenstein, M. and Verma, N., A New Segmentation Algorithm for Handwritten Word Recognition. in *Proceedings of International Joint Conference on Neural Networks*, (1999), 2893-2898.

8. Blumenstein, M., Liu, X. Y. and Verma, N., A Modified Direction Feature for Cursive Character Recognition. in *Proceedings of International Joint Conference on Neural Networks*, (2004), 2983-2987.

9. Cai, J. and Liu, Z. 1999. Integration of Structural and Statistical Information for Unconstrained Handwritten Numeral Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21 (3), 263-270.

10. Casey, R. and Lecolinet, E. 1996. A Survey of Methods and Strategies in Character Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18 (7), 690-706.

11. Cuc, D. T. K., Huy, D. Q., An, T. H. and Trong, N. V., Handwritten Number recognition and its Application at Danang University of Technology. in *Proceedings of the Conference Report for Scientific Research Students Eighth UD*, (2012).

12. Dongre, V. and Mankar, V. 2010. A Review of Research on Devnagari Character Recognition. *International Journal of Computer Applications*, 12 (2), 8-15.

13. Farooq, F., Govindaraju, V., Perrone, M. P., Pre-processing Methods for Handwritten Arabic Documents. in *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, (2005), 267-271.

14. Garg, N. K., Kaur, L. and Jindal, M. K. 2010. Segmentation of Handwritten Hindi Text. *International Journal of Computer Applications*, 1 (4), 19-23.

15. Ha, T. M. and Bunke, H. 1997. Off-Line, Handwritten Numeral Recognition by Perturbation Method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19 (5), 535-539.

16. Jayarathna, U. K. S. and Bandara, G. E. M. D. C., New Segmentation Algorithm for Offline Handwritten Connected Character Segmentation. in *Proceedings of*

*the First International Conference on Industrial and Information Systems*, (2006), 540-546.

17. Kasturi, R., Gorman, L. and Govindaraju, V. 2002. Document Image Analysis: A Primer. *S¯adhan¯a*, 27 (1), 3–22.

18. Kaur, A., Kumar, R., and Singh, A. 2010. A Hybrid Approach to Classify Gurmukhi Script Characters. *International Journal of Recent Trends in Engineering and Technology*, 3 (2), 103-105.

19. Kompalli, S., Nayak, S., Setlur, S. and Govindaraju, V., Challenges in OCR of Devanagari Documents. in *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, (2005), 327-331.

20. Kumar, R. and Dhiman, S., Challenges in Segmentation of Text in Handwritten Gurmukhi Script. in *Proceedings of the International Conference on Recent Trends in Business Administration and Information Processing*, CCIS 70, Springer-Verlag Berlin Heidelberg, (2010), 388-392.

21. Kumar, R. and Singh, A. 2008. Segmentation of Handwritten Text in Gurmukhi Script. *International Journal of Computer Science and Security*, 2 (3), 12-17.

22. Kumar, R. and Singh, A. 2010. Detection and Segmentation of Lines and Words in Gurmukhi Handwritten Text using Flexible Windowing. *International Journal of Computer Theory and Engineering*, 2 (3), 329 – 332.

23. Kumar, R. and Singh, A. 2011. Algorithm to Detect and Segment Gurmukhi Handwritten Text into Lines, Words and Characters. *International Journal of Engineering and Technology*, 3 (4), 392-395.

24. Kumar, S. 2010. An Analysis of Irregularities in Devnagari Script Writing - A Machine Recognition Perspective. *International Journal on Computer Science and Engineering*, 2 (2), 274-279.

25. Leary, R., Unrestricted Off-Line Handwriting Recognition A Preprocessing Approach. Retrieved 15 March, 2012: http://handwriting.rleary.com/wp-content/uploads/2009/12/csci4968_final.pdf.

26. Lehal, G. S. and Singh, C. 1998. Feature Extraction and Classification for OCR of Gurmukhi Script. *Vivek*, 12 (2), 2-12.

27. Lehal, G. S. and Singh, C., A Gurmukhi Script Recognition System. in *Proceedings of the Fifteenth International Conference on Pattern Recognition, IEEE Computer Society Press*, (2000), 557-560.

28. Lu, Y. 1995. Machine Printed Character Segmentation-An Overview. *Pattern Recognition*, 28 (1), 67-80.

29. Malakar, S., Mohanta, D., Sarkar, R. and Nasipuri, M. 2010. A Novel Noise-Removal Technique for Document Images. *Special Issue of IJCCT*, 2 (2, 3, 4), 121-124.

30. Mantas, J. 1986. An Overview of Character Recognition Methodologies. *Pattern Recognition*, 19 (6), 425-430.

31. Mo, S. and Mathews, J. 1998. Adaptive, Quadratic Preprocessing of Document Images for Binarization. *IEEE Transactions on Image Processing*, 7 (7), 992-999.

32. Mori, S., Suen, C. Y. and Yamamoto, K., Historical Overview of OCR Research and Development. in *Proceedings of the IEEE*, (1992), 1029-1058.

33. Pal, U. and Chaudhuri, B. B. 2004. Indian Script Character Recognition: A Survey. *Pattern Recognition*, 37, 1887-1899.

34. Pal, U., Belaid, A. and Chaudhuri, B. B., A System for Bangla Handwritten Numeral Recognition. in *Proceedings of the International Conference on Knowledge Based Computing Systems*, (2002), 433-443.

35. Pal, U., Sharma, N., Wakabayashi, T. and Kimura, F. 2008. Handwritten Character Recognition of Popular South Indian Scripts. *Springer Verlag book on Arabic and Chinese Handwriting Recognition*, 251-264.

36. Pal, U., Sinha, S. and Chaudhuri, B. B., Multi-Script Line identification from Indian Documents. in *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, (2003), 880-884.

37. Peters II, R. A. 1995. A New Algorithm for Image Noise Reduction using Mathematical Morphology. *IEEE Transactions on Image Processing*, 4 (3), 554-568.

38. Plamondon, R. and Srihari, S. N. 2000. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22 (1), 63-84.

39. Prasad, J. R., Kulkarni, U. V. and Prasad, R. S., Offline Handwritten Character Recognition for Gujrati Script using Pattern Matching. in *Proceedings of the third international conference on Anti-Counterfeiting, security, and identification in communication*, (2009), 611-615.

40. Ramamurthy, O. V., Roy, S., Narang, V. and Hanmandlu, M. 2012. Devanagari Character Recognition in the Wild. *International Journal of Computer Science*, 38 (4), 38-45.

41. Ramteke, R. J. and Mehrotra, S. C. 2008. Recognition of Handwritten Devnagari Numerals. *International Journal of Computer Processing of Oriental Languages, Chinese Language Computer Society & World Scientific Publishing Company*.

42. Saba, T., Sulong, G. and Rehman, A. 2010. Document Image Analysis: Issues, Comparison of Methods and Remaining Problems. *Springer Science+Business Media B.V.*, 101-110.

43. Senior, A. and Robinson, A. 1998. An Off-line Cursive Handwriting Recognition System. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20 (3), 309-312.

44. Sharma, N., Pal, U., Kimura, F. and Pal, S., Recognition of Offline Handwritten Devnagari Characters using Quadratic Classifier. in *Proceedings of the Indian Conference on Computer Vision Graphics and Image Processing (ICVGIP)*, LNCS, Springer Verlag, (2006), 805-816.

45. Shukla, M. K. and Bunka, H. 2011. An Efficient Segmentation Scheme for the Recognition of Printed Devnagari Script. *International Journal of Computer Science and Technology*, 2 (4), 529-531.

46. Srihari, S. N. and Lam, S. W. 1995. Character Recognition. Technical Report, CEDAR-TR-95-1.

47. Stevens, M. E. 1961. Automatic Character Recognition - A State of the Art Report. *National Bureau of Standards Tech Note-112*.

48. Timar, G., Karacs, K. and Rekeczky C., Analogic Preprocessing and Segmentation Algorithms for Off-line Handwriting Recognition. in *Proceedings of the Seventh IEEE International Workshop on Cellular Neural Networks and their Applications*, (2002), 407- 414.

49. Trier, O. D., Jain, A. K. and Taxt, T. 1996. Feature Extraction Methods for Character Recognition - A Survey. *Pattern Recognition*, 29 (4), 641-662.

50. Verma, B., A Contour Code Feature Based Segmentation for Handwriting Recognition. in *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, (2003), 1203-1207.

51. Wu, V. and Manmatha, R., Document Image Clean-Up and Binarization. in *Proceedings of Society for PhotoOptical Instrumentation Engineers (SPIE)*, (1997), 263-273.

52. Zhang, T. Y. and Suen, C. Y. 1984. A Fast Parallel Algorithm for Thinning Digital Patterns. *Communications of the ACM*, 27 (3), 236-239.