ARTIFICIAL NEURAL NETWORK BASED APPROACH FOR MIZO CHARACTER RECOGNITION SYSTEM

A thesis submitted

in fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In

COMPUTER SCIENCE

By

LALTHLAMUANA

Regd. No. : MZU/PhD/273 of 14.11.2008

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCE

SCHOOL OF PHYSICAL SCIENCES

MIZORAM UNIVERSITY

TANHRIL, AIZAWL – 796004

October, 2015

MIZORAM UNIVERSITY

AIZAWL : MIZORAM

DEPARTMENT OF MATHEMATICS & COMPUTER SCIENCES

Gram: MZU Fax:0389-2330873 Ph:0389-2330874, 9436352389(M), website: www.mzi.edu.in

E-mail:jamal_tezu@yahoo.com

CERTIFICATE

This is to certify that the research thesis entitled ***Artificial Neural Network based approach for Mizo Character Recognition System*** submitted by ***Mr. Lalthlamuana*** to Mizoram University, Tanhril, Aizawl, for the award of the degree of Doctor of Philosophy is a bonafide record of research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place : Aizawl

Date :…….

(Prof. JAMAL HUSSAIN)

SUPERVISOR

DECLARATION

I, **Mr. LALTHLAMUANA**, hereby declare that the subject matter of this thesis is the record of work done by me, that the contents of this thesis did not form basis of the award of any previous degree to me or to do the best of my knowledge to anybody else, and that the thesis has not been submitted by me for any research degree in any other University/Institute.

This is being submitted to the Mizoram University for the degree of Doctor of Philosophy in **Computer Science**.

Place : Aizawl

Date :…….

(LALTHLAMUANA)

Regd. No. : MZU/PhD/273 of 14.11.2008

(Prof. JAMAL HUSSAIN)

SUPERVISOR

ACKNOWLEDGEMENT

I would like to express my sincere thanks to my supervisor Prof. Jamal Hussain, Head, Department of Mathematics & Computer Science, Mizoram University for his valuable guidance, constant encouragement, stimulating discussions and extensive help leading to successful completion of this research work. I would also like to thank Prof. R.C. Tiwari, Dean, School of Physical Sciences, Mizoram University for his valuable suggestions and support.

I would also like to convey my sincere gratitude and appreciation to the faculties – Dr. Jay Prakash Singh, Mr. Laltanpuia, Dr. K.B. Mangang, Dr. S. Sarat Singh, and Ms. M. Saroja Devi, Department of Mathematics & Computer Science, Mizoram University for all the help and encouragements they have bestowed upon me. I cannot fail to thank the staff members of the Department of Mathematics & Computer Science, Mizoram University for their continual support and assistance whenever needed.

I am highly in debt of gratitude to Mr. Vanlalruata, System Integrator, Mizoram State e-Governance Society, Government of Mizoram who has provided limitless assistance in programming for the success of this research work. I am also deeply grateful to Mr. Lalhmachhuana, Secretary to Govt of Mizoram, Department of Information & Communication Technology for granting me permission to pursue my research and also for his encouragement and understanding.

It is mostly my family members who boosted my confidence in moments of despair. Without their support, I would not have completed this research work. I wish to express my sense of appreciation to my parent, Mr. Sangzuala and Mrs. Lalremi and to my sons – Zorinsanga, Lalngaihawma and daughter - Lalrinawmi for their total support, encouragement and understanding.

Finally and foremost, I want to express my deepest gratitude and my total dependence upon the wonderful Holy Spirit who has inspired and empowered me to complete this research work. Without Him, I could never have done this and may everything I do be for His glory.

Place : Aizawl

Date :…….

(LALTHLAMUANA)

Regd. No. : MZU/PhD/273 of 14.11.2008

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Most people learn to read and write during their first few years of education. By the time they have grown out of childhood, they have already acquired very good reading and writing skills, including the ability to read most texts, whether they are printed in different fonts and styles, or handwritten neatly or sloppily. Most people have no problem in reading the following: light prints or heavy prints; upside down prints; advertisements in fancy font styles; characters with flowery ornaments and missing parts; and even characters with funny decorations, stray marks, broken, or fragmented parts; misspelled words; and artistic and figurative designs. At times, the characters and words may appear rather distorted and yet, by experience and by context, most people can still figure them out. On the contrary, despite more than five decades of intensive research, the reading skill of the computer is still way behind that of human beings (Cheriet *et al.*, 2007).

The Pattern Recognition is still an ongoing wide research study, which tries to make machine as intelligent as human being for recognizing patterns. Pattern recognition (PR) is the most important trait of cognitive ability, be it of humans or animals. The ability to recognize patterns is central to intelligent behavior. We receive signals from environment through our sensory organs which are processed by the brain to generate suitable responses. The whole process involves extraction of information from the sensory signals, processing it using the information stored in the brain to reach a decision that induces some action. All these information we work with are represented as patterns. We recognize voices, known faces, scenes, type and written letters and a

multitude of other objects in our everyday life (Mayank *et al.*, 2011).

The character recognition is one of the most successful applications of technology in the field of pattern recognition and artificial intelligence. The character recognition system offer potential advantages by providing an interface that facilitates interaction between human and machine. Machine replication of human functions, like reading, is an ancient dream. For the past few decades, intensive research has been done to solve this problem in related areas such as image processing, pattern recognition, cognitive science, etc. Various approaches, system architectures and methodologies have been proposed to deal with application diversity. To date, challenging problems are being encountered and solutions to these are broadly targeted to improve accuracy and efficiency. However, trade-off between efficiency and accuracy is inevitable when a system targeted for real application is designed. With this motivation, a computationally efficient solution to the problem of recognition of characters based on Artificial Neural Network classifier that has some similarities to the human cognitive process is proposed.

1.1 MIZO LANGUAGE AND ALPHABETS

Mizo is a member of the Kukish branch of the Tibeto-Burman language family spoken by about 15 million people mainly in Mizoram state in India, and also in Chin State in Burma, and in the Chittagong Hill Tracts in Bangladesh. Mizo used to be known as Lushai, Lusei or Lushei, named after the most common dialect of the language, which serves as a lingua franca among the Kuki people.

Mizo is the most developed tribal language of the North-East India and is taught in many schools and colleges. Mizo alphabets were made by British Christian Missionaries who came to Mizoram in the late 1800's and early 1900's (Henderson,

1948). During that time, Mizos were not having any kind of text writing system. So, when British came, they felt that it was necessary to make alphabet system for the people of Mizoram and the alphabets were prepared based on the pronunciation of Mizo language (Burling, 1957). Before making alphabet system for Mizo people, extensive comparisons were carried out to make a choice between Indian scripts like Hindi, Bengali etc and Roman scripts, which is used by English. Then the missionaries opined that Roman Script was most convenient for the people of Mizoram.

The 25 letters used for writing in Mizo language are:

Letter	a	aw	b	ch	d	e	f	g	ng	h	i	j	k
	l	m	n	o	p	r	s	t	ṭ	u	v	z	

Here a special character with lower circumflex i.e. "ṭ" and a compound character i.e. "aw", "ch" and "ng" have found which are not available in Roman script or English alphabets. In mizo language, the compound characters are treated as a single character. Out of these 25 letters, there are six vowels such as "**â**", "**âw**", "**ê**", "**î**", "**ô**" and "**û**". A circumflex ^ was later added to the vowels to indicate long vowels, viz., â, âw, ê, î, ô, û, which were insufficient to fully express Mizo tone. So, the word "zam" is different from "zâm" in which the latter is pronounced like "zaam" and it has different meaning. Another different alphabet which is not used in normal English alphabet, the alphabet is "ṭ", pronounced as 'tree' while "t" is pronounced as 'tee'. Since ASCII keyboard is being used everywhere, there is difficulty in typing the extended type of vowels which are not readily available in the keyboard even when they are supposed to be used, they are sometimes neglected when typing in computer and also when the meaning can still be understood. But in handwritten form and in publications, the correct

form of writing is usually followed.

The combination of characters like ‘aw’, ‘ch’ and ‘ng’ can be considered as a combination of two English alphabets as:

aw = a + w
ch = c + h
ng = n + g

So, in the matter of recognition of characters, the Mizo alphabets which composed of two English alphabets will be considered as two different alphabets. There is no grammatical gender in Mizo language, although some animals, birds etc. have names which contain one of the suffixes -nu, which means female, or -pa which means male. Examples include chingpirinu (a type of big owl), kawrnu (a type of cicada), thangfênpa (a nocturnal bird). Mizo is an agglutinative language in which it is rare to find morphologically simple, non-derived nouns. However, common everyday objects and domestic animals tend to fall in this category, that is, the category of morphologically simple, non-derived nouns (Chhangte, 1986).

1.2 MOTIVATION FOR THE PRESENT WORK

Though there are many OCRs available in major Indian languages, none of them are capable of recognizing mizo language due to different fonts type and style. The English character pattern is closely similar to mizo character but there are special characters incorporated in mizo language. Therefore, the existing English OCR cannot be used for data digitization of mizo languages as the accuracy is only 80-90%. While carrying out of research, it was found that the same technique which is implemented in other languages character recognition is not suitable for recognizing Mizo characters. Therefore, it is very much important to design and develop a separate OCR for those

who speak, read and write in Mizo language.

Apart from character recognition, recognizing the font of a printed document is not even attempted on Indian language documents; while some successful studies are made in English. Typographically, a font is a particular instantiation of a typeface design, often in a particular size, weight and style (Felici, 2011). A number of OCR systems have been developed for different languages across the globe, with reasonable accuracy, but the performance of these recognizers is fair as long as the same font is maintained. Since this requirement is not practical, often we get poor results. In many occasions, printed documents may contain words in various font faces and sizes. For Indian and many other oriental languages, OCR systems are not yet able to successfully recognize printed document images of varying scripts, quality, size, style and font (Rawat *et al.*, 2006).

Mizo optical character recognition system is so far not been into consideration for the purpose of research and development unlike the other languages which make it a challenging factor. There are many Mizo people living in Mizoram, Manipur, Nagaland and Myanmar using the same Mizo language. Due to all the above factors, an attempt is made to carry out research and development of mizo OCR to enable to recognize all the mizo characters and used for data digitization and preservation of historical documents into digital form.

1.3 OBJECTIVES

- The primary objective of this research work is to design and develop pre-printed Mizo character recognition system using Artificial Neural Network.

- To provide robust system for digitization of innumerable old documents available both on single sheet paper and books.
- To develop an efficient recognition system with multi font and multi size Mizo characters and put it in public domain for general uses of the people speaking, reading and writing Mizo language.
- To provide standard Unicode encoding system for further interfacing with other language.

1.4 HISTORY OF OCR

It is always fascinating to be able to find ways of enabling a computer to mimic human functions, like the ability to read, to write, to see things, and so on. To replicate the human functions by machines, making the machine able to perform tasks like reading is an ancient dream. The origins of character recognition can actually be found back in 1870. This was the year that C. R. Carey of Boston Massachusetts invented the retina scanner which was an image transmission system using a mosaic of photocells. Two decades later the Polish P. Nipkow invented the sequential scanner which was a major breakthrough both for modern television and reading machines. During the first decades of the 19^{th} century several attempts were made to develop devices to aid the blind through experiments with OCR. However, the modern version of OCR did not appear until the middle of the 1940's with the development of the digital computer. The motivation for development from then on, was the possible applications within the business world.

By 1950 the technological revolution was moving forward at a high speed, and electronic data processing was becoming an important field. Data entry was performed through punched cards and a cost-effective way of handling the increasing

amount of data was needed. At the same time the technology for machine reading was becoming sufficiently mature for application, and by the middle of the 1950's OCR machines became commercially available. The first true OCR reading machine was installed at Reader's Digest in 1954. This equipment was used to convert typewritten sales reports into punched cards for input to the computer. To understand the evolution of OCR systems from their challenges, and to appreciate the present state of the OCRs, a brief historical survey of OCR is carried out. Depending on the versatility, robustness and efficiency, commercial OCR system may be divided into the following four generations (Pal and Chaudhuri, 2004). It is to be noted that this categorization refers specifically to OCRs of English languages.

1.4.1 FIRST GENERATION OCR

The commercial OCR systems appearing in the period from 1960 to 1965 may be called the first generation of OCR. These generations of OCR machines were mainly characterized by the constrained letter shapes read. The symbols were specially designed for machine reading, and the first ones did not even look very natural. With time multi-font machines started to appear, which could read up to ten different fonts. The number of fonts were limited by the pattern recognition method applied, template matching, which compares the character image with a library of prototype images for each character of each font.

1.4.2 SECOND GENERATION OCR

The reading machines of the second generation appeared in the middle of the 1960's and early 1970's. These systems were able to recognize regular machine printed characters and also had hand-printed character recognition capabilities. When hand-printed characters were considered, the character set was constrained to numerals and a

few letters and symbols. The first and famous system of this kind was the IBM 1287, which was exhibited at the World Fair in New York in 1965. Also, in this period Toshiba developed the first automatic letter sorting machine for postal code numbers and Hitachi made the first OCR machine for high performance and low cost.

In this period significant work was done in the area of standardization. In 1966, a thorough study of OCR requirements was completed and an American standard OCR character set was defined; OCR-A font was defined as shown in the figure below, which was designed to facilitate OCR, although still readable to humans. A European font, OCR-B, was also designed which had more natural fonts than the American standard. Some attempts were made to merge the two fonts into one standard, but instead machines being able to read both standards appeared.

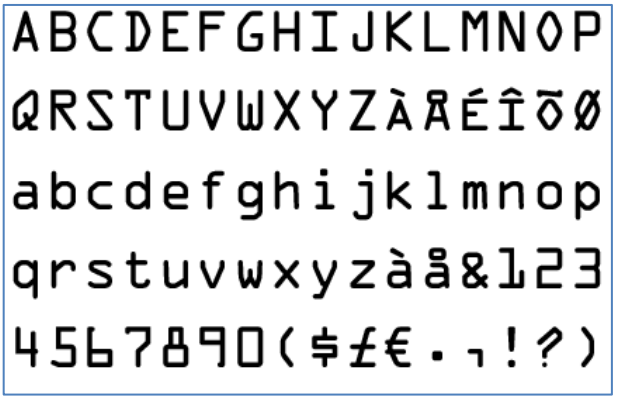


Figure 1.1: OCR-A font

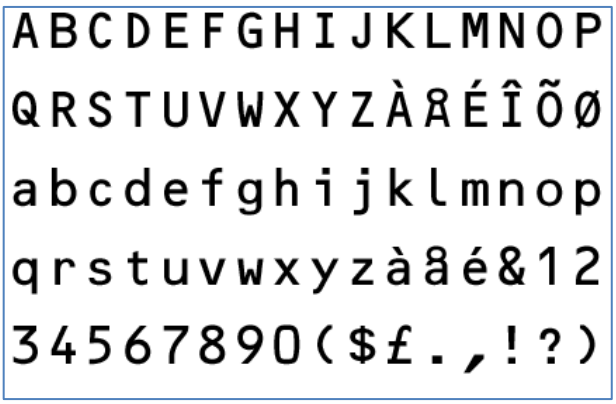


Figure 1.2: OCR-B fonts

1.4.3 THIRD GENERATION OCR

For the third generation of OCR systems appearing in the middle of the 1970's, the challenge was documents of poor quality and large printed and hand-written character sets. Low cost and high performance were also important objectives, which were helped by the dramatic advances in hardware technology.

Although more sophisticated OCR-machines started to appear at the market simple OCR devices were still very useful. In the period before the personal computers and laser printers started to dominate the area of text production, typing was a special niche for OCR. The uniform print spacing and small number of fonts made simply designed OCR devices very useful. Rough drafts could be created on ordinary typewriters and fed into the computer through an OCR device for final editing. In these way word processors, which were an expensive resource at this time, could support several people and the costs for equipment could be cut.

1.4.4 FOURTH GENERATION OCR (TODAY's OCR)

The fourth generation can be characterized by the OCR of complex documents intermixing with text, graphics, tables and mathematical symbols, unconstrained handwritten characters, color documents, low-quality noisy documents, etc. Among the commercial products, postal address readers, and reading aids for the blind are available in the market.

Nowadays, there is much motivation to provide computerized document analysis systems. OCR contributes to this progress by providing techniques to convert large volumes of data automatically. A large number of papers and patents advertise recognition rates as high as 99 %; this gives the impression that automation problems

seem to have been solved. Failure of some real applications show that performance problems still exist on composite and degraded documents (i.e., noisy characters, tilt, mixing of fonts, etc.) and that there is still room for progress.

Various methods have been proposed to increase the accuracy of optical character recognizers. In fact, at various research laboratories, the challenge is to develop robust methods that remove as much as possible the typographical and noise restrictions while maintaining rates similar to those provide by limited-font commercial machines.

1.5 CLASSIFICATION OF CHARACTER RECOGNITION SYSTEM

The Character Recognition Systems are generally classified into online character recognition system and off-line character recognition system. These types of Character Recognition Systems are shown in the figure below:

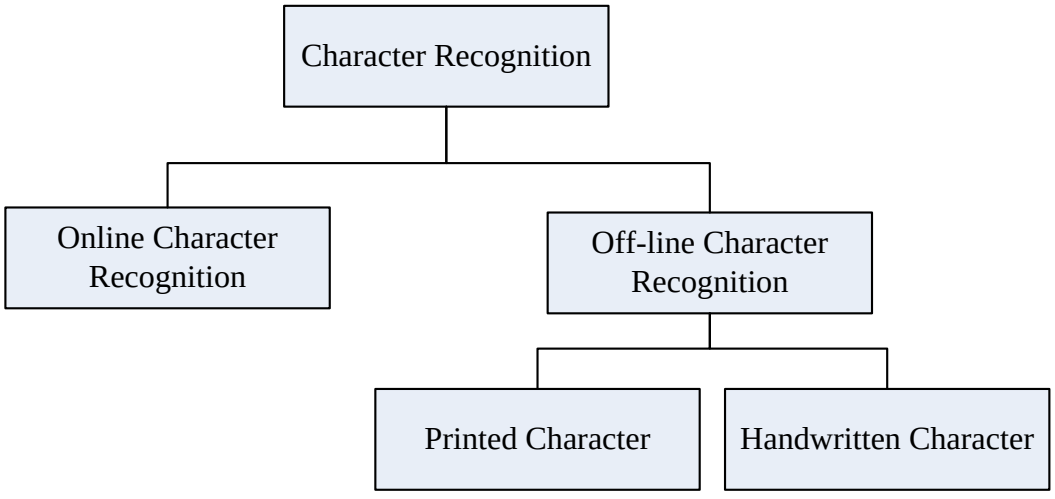


Figure 1.3: Types of Character Recognition System

1.5.1 ONLINE CHARACTER RECOGNITION SYSTEM

On-line recognition refers to methods and a technique dealing with the automatic processing of a message as it is written using a digitizer or an instrumental

stylus that captures information about the pen-tip, generally its position, velocity, or acceleration as a function of time (Plamondon and Srihari, 2000). The digitizers are mostly electromagnetic-electrostatic tablets, which send the coordinates of the pen tip to the host computer at regular intervals. Some digitizers use pressure-sensitive tablets, which have layers of conductive and resistive material with a mechanical spacing between the layers. The on-line handwriting recognition problem has a number of distinguishing features, which must be exploited to get more accurate results than the off-line recognition problem.

Advantages of on-line character recognition system:

1. **It is a real time process.** While the digitizer captures the data during the writing, the CR system with or without a lag makes the recognition.
2. **It is adaptive in real time.** The writer gives immediate feedback to the recognizer for improving the recognition rate, as (s)he keeps drawing the symbols on the tablet and observes the results.
3. **It captures the temporal and dynamic information of the pen trajectory.** This information consists of the number and order of pen-strokes, the direction of the writing for each pen-stroke and the speed of the writing within each pen stroke.
4. **Very little pre-processing is required.** The operations, such as smoothing, de-slanting, de-skewing, detection of line orientations, corners, loop and cusps are easier and faster with the pen trajectory data than on pixel images.
5. **Segmentation is easy.** Segmentation operations are facilitated by using temporal and pen-lift information, particularly, for hand-printed characters.

Disadvantages of on-line character recognition system:

1. The writer *requires* special equipment, which is not as comfortable as pen and paper.
2. It cannot be applied to documents printed or written on papers.
3. Punching is much faster and easier than handwriting for small size alphabet such as English.
4. The available systems are slow and recognition rates are low for handwriting that is not neat.

Applications of on-line character recognition systems include small hand-held devices, which call for a pen-only computer interface and complex multimedia systems, which use multiple input modalities including scanned documents, speech, keyboard and electronic pen. They provide an efficient alternative for the large alphabets where the keyboard is cumbersome. Pen based computers, educational software for teaching handwriting and signature verifiers are the examples of popular tools utilizing the on-line character recognition techniques.

1.5.2 OFF-LINE CHARACTER RECOGNITION SYSTEMS:

Off-line character recognition is known as Optical Character Recognition (OCR), because the image of writing is converted into bit pattern by an optically digitizing device such as optical scanner or camera. The recognition is done on this bit pattern data for machine-printed or hand-written text. The research and development is well progressed for the recognition of the machine-printed documents. In recent years, the focus of attention is shifted towards the recognition of hand-written script.

The major advantage of the off-line recognizers is to allow the previously

written and printed texts to be processed and recognized. The drawbacks of the off-line recognizers, compared to on-line recognizers are summarized as follows:

1. Off-line conversion usually requires costly and imperfect pre-processing techniques prior to feature extraction and recognition stages.
2. The lack of temporal or dynamic information results in lower recognition rates compared to on-line recognition.

Some applications of the off-line recognition are large-scale data processing such as postal address reading; check sorting, office automation for text entry, automatic inspection and identification (Said, 2000). Off-line character recognition is a very important tool for creation of the electronic libraries. It provides a great compression and efficiency by converting the document image from any image file format into more useful formats like HTML or various word processor formats. Recently, content based image or video database systems make use of off-line character recognition for indexing and retrieval, extracting the writings in complex images. Also, the wide spread use of web necessitates the utilization of off-line recognition systems for content based Internet access to paper documents.

The off-line character recognition generally divided into machine-printed and hand-written. Machine-printed text includes the materials such as books, newspapers, magazines, documents and various writing units in the video or still image. The problems for fixed-font, multifont and omni-font character recognition is relatively well understood and solved with little constraint (Peng *et al.*, 2010).

On the other hand, hand-written character recognition systems have still limited capabilities even for recognition of the Latin characters. The problem can be

divided into two categories: cursive and hand-printed script. In practice, however, it is difficult to draw a clear distinction between them. A good source of references in hand-written character recognition can be found in Mori *et al.* (1999)..

1.6 COMPONENT OF CHARACTER RECOGNITION SYSTEM

The major components of Character recognition system suggested by Annadurai and Shanmugalakshmi (2007) are shown in the figure below.

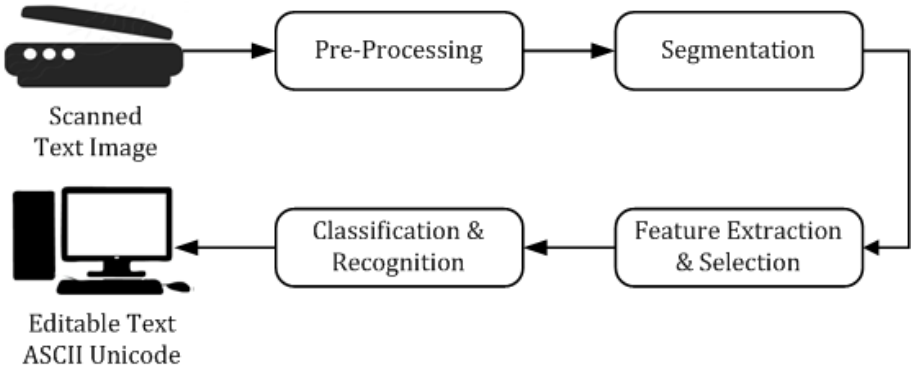


Figure 1.4: General Architecture of Character Recognition System

1.6.1 PREPROCESSING

The preprocessing is a series of operation performed on scanned input image. The image should have specific format such as jpeg, bmp, tiff, etc. This image is acquired through a scanner, digital camera or any other suitable digital input devices. The role of pre-processing is to segment the interesting pattern from the background image. Typical preprocessing includes binarization, smoothing & noise removal, skew detection and correction, slant correction and thinning.

1.6.1.1 BINARIZATION:

Binarization is a technique by which the gray scale images are converted into binary images. Binarization separates the foreground (text) and background information

(Alam and Kashem, 2010). The most common method for binarization is to select a proper threshold for the intensity of the image and then convert all the intensity values above the threshold to one intensity value ("white"), and all intensity values below the threshold to the other chosen intensity ("black"). Otsu's thresholding technique (Otsu, 1979) is one of the most popular thresholding technique frequently used today (Leedham *et al.*, 2002).

1.6.1.2 SMOOTHING AND NOISE REMOVAL:

Scanned documents often contain noise that arises due to printer, scanner, print quality, age of the document, etc. While scanning the document images, the device introduce some noises like, disconnected line segments, bumps and gaps in lines, filled loops etc. It is necessary to remove all these noise elements prior to the character recognition. Noise removal is the process of removing or reducing unwanted noise. Smoothing operations are generally used to reduce the noise or to straighten the edges of the characters. There are different types of noises such as Gaussian noise, Salt and pepper noise and shot noise (Vithlani, 2014). These noises can be removed by filtering approaches – Linear filter and Non-linear filter (Cheriet *et al.*, 2007).

1.6.1.3 SKEW DETECTION AND CORRECTION:

Skew detection and correction of scanned document images is one of the most important stages of preprocessing. The skew of the scanned document image specifies the deviation of its text lines from horizontal or vertical axis. The skew of the document image can be a global (all document's blocks have the same orientation), multiple (document's blocks have a different orientation) or non-uniform (multiple orientation in a text line). Skew correction aligns an image before processing because text segmentation and recognition methods require properly aligned text lines. A number of methods have previously been proposed in the literature for identifying document

image skew angles. Mainly, they can be categorized into the following groups (Lu and Tan, 2003): (i) methods based on projection profile analysis; (ii) methods based on nearest- neighbor clustering; (iii) methods based on Hough transform; (iv) methods based on cross-correlation; and (v) methods based on morphological

1.6.1.4 SLANT CORRECTION:

The character inclination that is normally found in cursive writing is called slant. The figure below shows some samples of slanted handwritten numeral string. Slant correction is an important step in the preprocessing stage of both handwritten words and numeral strings recognition. The general purpose of slant correction is to reduce the variation of the script and specifically to improve the quality of the segmentation candidates of the words or numerals in a string, which in turn can yield higher recognition accuracy.

Bertolami *et al.* (2007) explains the use of non-uniform slant correction technique in offline recognition of handwritten lines of text. It is used for additional preprocessing task. The work is motivated by the fact that many handwriting styles exhibit a variety of different slant angles within a single line of text or even within individual words. The non-uniform slant correction is formulated as a constrained optimization problem where the local slant angles represent the variables to be optimized. They have used a dynamic programming based algorithm to solve this optimization problem.

1.6.1.5 THINNING

Thinning is the process of peeling off a pattern as many pixels as possible without affecting the general shape of the pattern. In other words, after pixels have been peeled off, the pattern can still be recognized. Hence, the skeleton obtained must have

the following properties:

- Must be as thin as possible;
- Connected;
- Centered.

Jubair and Banik (2012) proposed morphological thinning operation in which the selected foreground pixels is removed from binary images. It can be used for several applications, but is particularly useful for skeletonization. This method is commonly used to tidy up the output of edge detectors by reducing all lines to single pixel thickness. Thinning is normally only applied to binary images, and produces another binary image as output.

1.6.2 SEGMENTATION

Once the document image is binarized and skew corrected, it will passes to the segmentation phase, where the image will be decomposed into line, word and individual character. This may be termed as 'Character Segmentation'. The Character segmentation is the critical area of the Optical Character Recognition process. In the literature, for achieving high recognition accuracy, several segmentation techniques are proposed that can be broadly classified into four categories, namely explicit segmentation (classical Segmentation), implicit segmentation (recognition Based segmentation), holistic (segmentation free), and hybrid segmentation (a combination of classical and recognition segmentation). The figure below shows the character segmentation methods:

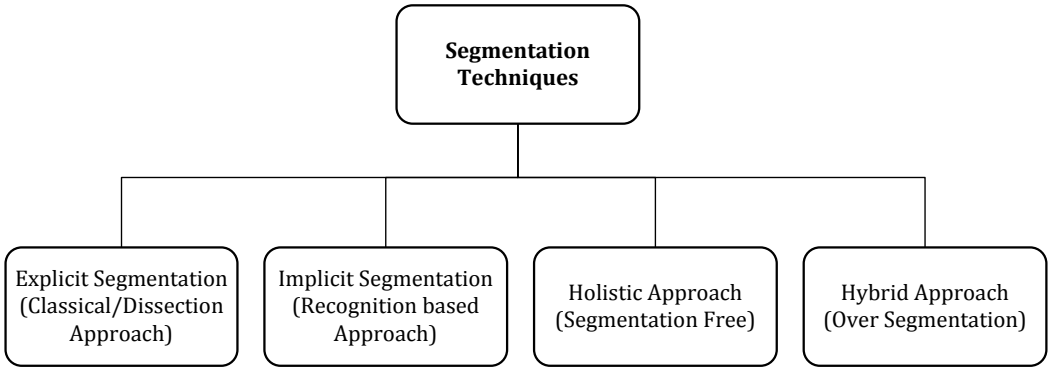


Figure 1.5: Character Segmentation Techniques

1.6.2.1 EXPLICIT SEGMENTATION:

The explicit segmentation is also known as **Classical Segmentation**. In this classical approach, input word image of sequence of characters is partitioned into sub images of individual characters. The process of cutting up the word images into classifiable character sub images is termed as **Dissection**. Many researchers in the literature adopted this dissection based segmentation techniques (Saba *et al.*, 2011). The criterion for good segmentation using the dissection approach is the agreement of character properties in the segmented sub image and the expected symbol. The dissection method makes use of the character properties like height, width, space, separation from neighboring components, disposition along the baseline, etc. This method is suitable for printed image documents in which each character image is well spaced.

1.6.2.2 IMPLICIT SEGMENTATION:

Implicit segmentation is also known as **Recognition based segmentation**. In this approach, the system searches the image for components that match classes in its alphabet. However, implicit segmentation-based methods are employed as an alternative to integrate segmentation and recognition processes. Accordingly, Hidden Markov Models (HMM) based approaches are emerged. Actually, this approach is developed for speech recognition where it brought fruitful results (Gales and Young, 2008). Therefore,

its success diverts researcher's attention to apply HMM in word recognition. The main interest of this category of methods is that they bypass the segmentation problem: No complex "dissection" algorithm has to be built and recognition errors are basically due to failures in classification. The approach has also been called "segmentation-free" recognition.

1.6.2.3 HOLISTIC APPROACH

A holistic is also known as "**Segmentation Free**". In this approach, the process recognizes an entire word as a unit. A major drawback of this class of methods is that their use is usually restricted to a predefined lexicon. Since they do not deal directly with letters but only with words, recognition is necessarily constrained to a specific lexicon of words. This point is especially critical when training on word samples is required. A training stage is thus mandatory to expand or modify the lexicon of possible words. This property makes this kind of method more suitable for applications where the lexicon is statically defined (and not likely to change), like bank cheque recognition. They can also be used for on-line recognition on a personal computer (or notepad), the recognition algorithm being then tuned to the writing of a specific user as well as to the particular vocabulary concerned (Ntzios *et al.*, 2007).

1.6.2.4 HYBRID APPROACH

The Hybrid segmentation is also known as "**Over Segmentation**". In this method, the first two method i.e explicit and implicit segmentation are combine together in a hybrid way. In other term it may be said that the hybrid segmentation is a combination of dissection and search methods in a hybrid way. In this approach, dissection algorithm is applied to the image, but the objective here is not to get a single character or specific features but to "over segment", i.e. to cut the image in sufficiently many places so that the correct segmentation boundaries are included among the cuts

made (Razzak *et al*., 2010).

1.6.3 FEATURE EXTRACTION

Feature extraction is a process of defining a set of features of an image which represent meaningful information for analysis and classification. In case of character image, a set of numerical features will be used to classify the character. The numerical feature of the character images could be height of character, width of character, number of horizontal lines presents, numbers of vertical lines present, centroid of the character image, position of the various features and pixels in the various regions.

Selection of a feature extraction method is probably the single most important factor in achieving high recognition performance in character recognition systems. Different feature extraction methods are designed for different representations of the characters, such as solid binary characters, character contours, skeletons (thinned characters), or gray level sub-images of each individual character. A feature extraction method that proves to be successful in one application domain may turn out not to be very useful in another domain. In practice, the requirement of a good feature extraction method makes selection of the best method for a given application a challenging task. One must also consider whether the characters to be recognized have known orientation and size, whether they are handwritten, machine printed or typed, and to what degree they are degraded. Feature extraction methods are generally classified into three major groups such as Statistical features, Global transformation and series expansion, and Geometric and topological features (Elavarasan and Mani, 2015).

1.6.4 CLASSIFICATION

The output of feature extraction is a feature vector obtained from previous phase is assigned as an input to next phase i.e. classification or class label and recognized

by means of supervised and unsupervised method. The classification method is based on feature vector which have prevailed structural methods, especially in off-line character recognition. These methods include statistical methods, ANNs, SVMs, and multiple classifier combination. In most of the classification method, the data set is prepared which is separated into training and test set for every character. The performance of the classification depends on the accuracy of feature extraction of the characters. The classification methods is summarized into four categories such as

- Statistical methods
- Artificial neural networks (ANNs)
- Support Vector Machines
- Structural Pattern Recognition
- Combine Multiple Classifiers

1.6.4.1 STATISTICAL METHODS:

Statistical classifiers are rooted in the Bayes decision rule, and can be divided into parametric ones and non-parametric ones (Burges, 1998). Non-parametric methods, such as Parzen window and k-NN rule, are not practical for real-time applications since all training samples are stored and compared. Assuming Gaussian density with various restrictions, the Bayesian discriminant function is reduced to a quadratic discriminant function (QDF), linear discriminant function (LDF), and Euclidean distance from class mean. Chang and Lin (2011) proposed regularized discriminant analysis (RDA) method which stabilizes the performance of QDF by smoothing the covariance matrices. Kimura *et al.* (1987) further proposed the modified QDF (MQDF) which has less parameters and lower computation than the QDF, and results in improved generalization accuracy. For modeling multi-modal distributions, the mixture of Gaussians in high dimensional

feature space does not necessarily give high classification accuracy, yet the mixture of linear subspaces has shown effects in handwritten character recognition (Cristianini and Taylor, 2000).

1.6.4.2 ARTIFICIAL NEURAL NETWORKS:

Feed forward neural networks, including multilayer perceptron (MLP), radial basis function (RBF) network, higher-order neural network (HONN), etc., have been widely applied to pattern recognition. The connecting weights are usually adjusted to minimize the squared error on training samples in supervised learning. Using a modular network for each class was shown to improve the classification accuracy (Webb, 2003). A network using local connection and shared weights, called convolutional neural network, has reported great success in character recognition (Dong *et al.*, 2005). The RBF network can yield competitive accuracy with the MLP when training all parameters by error minimization (Downs, 2001). The HONN is also called as functional-link network, polynomial network or polynomial classifier (PC). Its complexity can be reduced by dimensionality reduction before polynomial expansion (Drezet and Harrison, 2001) or polynomial term selection (Nikolaev and Iba, 2003). Vector quantization (VQ) networks and auto-association networks, with the sub-net of each class trained independently in unsupervised learning, are also useful for classification. The learning vector quantization (LVQ) of Kohonen is a supervised learning method and can give higher classification accuracy than VQ (Duda *et al.*, 2001). Some improvements of LVQ learn prototypes by error minimization instead of heuristic adjustment (Duin, 2002).

1.6.4.3 SUPPORT VECTOR MACHINES:

The Support Vector Machine (SVM) is a new type of hyper plane classifier, developed based on the statistical learning theory of Vapnik (2013), with the aim of maximizing a geometric margin of hyperplane, which is related to the error bound of

generalization. The research of SVMs has seen a boom from the mid-1990s, and the application of SVMs to pattern recognition has yielded state-of-the-art performance. Generally SVM classifier is a binary linear classifier in kernel induced feature space and kernel function. The kernel function represents the inner product of two vectors in linear/nonlinear feature space.

The Kernel methods, including support vector machines (SVMs) primarily and kernel principal component analysis (KPCA), kernel Fisher discriminant analysis (KFDA), etc., are receiving increasing attention and have shown superior performance in pattern recognition. An SVM is a binary classifier with discriminant function being the weighted combination of kernel functions over all training samples. After learning by quadratic programming (QP), the samples of non-zero weights are called support vectors (SVs). For multi-class classification, binary SVMs are combined in either one-against-others or one-against-one (pairwise) scheme (Bo and Xianwu, 2006). Due to the high complexity of training and execution, SVM classifiers have been mostly applied to small category set problems. Albus *et al*. (2012) proposed a strategy to alleviate the computation cost to use a statistical or neural classifier for selecting two candidate classes, which are then discriminated by SVM. Dong *et al.* (2005) used a one-against-others scheme for large set Chinese character recognition with fast training. They used a coarse classifier for acceleration but the large storage of SVs was not avoided.

1.6.4.4 STRUCTURAL PATTERN RECOGNITION

Structural pattern recognition methods are used more often in online character recognition (Liu *et al.*, 2004) than in offline character recognition. Unlike statistical methods and Neural Networks that represent the character pattern as a feature vector of fixed dimensionality, structural methods represent a pattern as a structure

(string, tree, or graph) of flexible size. The structural representation records the stroke sequence or topological shape of the character pattern, and hence resembles well to the mechanism of human perception. In recognition, each class is represented as one or more structural templates, the structure of the input pattern is matched with the templates and is classified to the class of the template of minimum distance or maximum similarity. The structural matching procedure not only provides an overall similarity but also interprets the structure of the input pattern and indicates the similarities of the components.

Despite the above merits of structural recognition, statistical methods and Neural Networks are more often adopted for the ease of feature extraction and learning from samples. Structural methods face two major difficulties: extracting structural primitives (strokes or line segments) from input patterns, and learning templates from samples. Primitive extraction from online character patterns (sequences of pen-down points) is much easier than from offline character images. Structural template learning from samples is undergoing study and has gained some progress. In practice, the templates are often selected from samples, constructed artificially or interactively.

Structural pattern recognition is often mentioned together with syntactic pattern recognition, which represents patterns and classes using formal linguistics and recognizes via grammatical parsing. Extracting linguistic representation from patterns is even more difficult than structural representation. This is why syntactic methods have not been widely used in practical recognition systems. There are two important types of structural recognition techniques that are useful for character recognition: (i) attributed string matching and (ii) attributed graph matching. String matching techniques are often used in character string recognition as well, for matching string patterns with lexicon

entries.

1.6.4.5 COMBINE MULTIPLE CLASSIFIERS:

Combining multiple classifiers has been long pursued for improving the accuracy of single classifiers (Fumera and Roli, 2005). Parallel (horizontal) combination is more often adopted for high accuracy, while sequential (cascaded, vertical) combination is mainly used for accelerating large category set classification. The decision fusion methods are categorized into abstract level, rank-level, and measurement-level combination (Giacinto and Roli, 2001). Many fusion methods have been proposed to measurement-level combination (Sousa *et al.*, 2007). The complementariness (also called as independence or diversity) of classifiers is important to yield high combination performance. For character recognition, combining classifiers based on different techniques of pre-processing, feature extraction, and classifier models is effective. Another effective method, called perturbation, uses a single classifier to classify multiple deformations of the input pattern and combine the decisions on multiple deformations. The deformations of training samples can also be used to train the classifier for higher generalization performance.

1.7 APPLICATION OF CHARACTER RECOGNITION SYSTEM

The intensive research effort in the field of Character Recognition was not only because of its challenge on simulation of human reading but also because it provides widespread efficient applications. The Optical Character Recognition technologies have many practical applications which include the following, as examples, but not limited to:

- Digitization, storing, retrieving and indexing huge amount of electronic

data as a results of the resurgence of the World Wide Web. The text produced by OCR can be used for all kinds of Information Retrieval (IR) and Knowledge Management (KM) systems which are not so sensitive to the inevitable Word Error Rate (WER) lower than 10% to 15%.

- Text-to-Speech for blind people as reading aid and transfer of the recognition result into sound output or tactile symbols through stimulators
- Telecommunication Device for Deaf (TDD). A TDD is a teleprinter. It is an electronic device which aids people with hearing or speech difficulties with communication through text and telephone lines.
- In postal department – for scanning and reading preprinted and handwritten postal address and postal codes.
- In publishing industry, as a text reader and store for editing and publishing documents/books.
- In Banking, for automated finger print identification, check reader, etc.
- Handwriting analyzer for automatic writer recognition and signature verification.
- For mechanized document reading in textile and clothing manufacture enterprises, automatic punching of industrial telegraphs, retail data processing applications in food enterprises, and for retail product code name and price reading techniques.
- In educational administrations – examinations assessment and attendance record evaluation.
- In automated cartography, metallurgical industries, computer assisted

forensic linguistic system, electronic mail, information units and libraries and for facsimile.

1.8 PROBLEMS OF RECOGNITION OF CHARACTERS

A character can be written in a number of ways differing in shape and properties, such as tilt, stroke, cursively and overall shape. A plethora of fonts are available for use in any commonly used Word Processing Application Software. Yet, while perceiving any text written in a variety of ways, humans can easily recognize and read each character. This is because the human perception processes the information by the features that define a character's shape in an overall fashion. Thus, while modeling the human perception model in machines, a rugged feature extraction algorithm is needed before classification of characters (Shrivastava and Sharma, 2012). Character misclassifications stem from two main sources: poor quality recognition unit (item) images and inadequate discriminatory ability of the classifier. There are many factors that contribute to noisy, hard to recognize item imagery:

- poor original document quality
- noisy, low resolution, multi-generation image scanning
- incorrect or insufficient image pre-processing
- poor segmentation into recognition items

On the other hand, the character recognition method itself may lack a proper response on the given character set, thus resulting in classification errors. This type of errors can be difficult to treat due to a limited training set or limited learning abilities of the classifier. Typical recognition rates for machine-printed characters can reach over 99% but handwritten character recognition rates are invariably lower because every

person writes differently. This random nature often manifests itself in a greater character variance in the feature space leading to greater misclassification rates. A common example of a “difficult” character is the letter “O” easily confused with the numeral “0”. Another good example could be the letter “l” confused with the numeral “1” or mistaken for a noisy image of the letter “I”. Rusu and Govindaraju (2004) discussed the character recognition abilities of human versus computers and present illustrated examples of recognition errors. The top level of their taxonomy of error causes consists of

- Imaging defects due to heavy/light print, stray marks, curved baselines, etc.
- Similar symbols as mentioned above
- Punctuation due to commas and periods, quotation marks, special symbols, etc.
- Typography due to italics and spacing, underlining, shaded backgrounds, unusual typefaces, very large/small print, etc.

Their analysis provides insight into the strengths and weaknesses of current systems, and a possible roadmap to future progress. They conclude that the current OCR devices cannot read even on the level of a seven-year old child. The authors consider four potential sources of improvement:

- Better image processing based on more faithful modeling of the printing, copying and scanning processes
- Adaptive character classification by fine-tuning the classifier to the current document’s typeface
- Multi-character recognition by exploiting style consistency in typeset

text.

- Increased use of context that depends on the document's linguistic properties and can vary from language to language.

On the basis of the diversity of errors that they have encountered, they are inclined to believe that further progress in OCR is more likely to be the result of multiple combinations of techniques than on the discovery of any single new overarching principle.

1.9 RECENT TREND AND DEVELOPMENT

Digitizing information makes it easier to preserve, access, and share. For example, an original historical document may only be accessible to people who visit its physical location, but if the document content is digitized, it can be made available to people worldwide. There is a growing trend towards digitization of historically and culturally significant data. At present, reasonable efficient and inexpensive OCR packages are commercially available for digitization of printed text in English, Chinese and Japanese.

The accurate recognition of Latin-script, typewritten text is now considered largely a solved problem. Typical accuracy rates exceed 99%, although certain applications demanding even higher accuracy require human review for errors. Other areas including recognition of hand printing, cursive handwriting, and printed text in other scripts (especially those with a very large number of characters) are still the subject of active research (Sharma *et al.,* 2013).

Optical Character Recognition (OCR) is sometimes confused with on-line character recognition. OCR is an instance of off-line character recognition, where the

system recognizes the fixed static shape of the character, while on-line character recognition instead recognizes the dynamic motion during handwriting. Recognition of cursive text is an active area of research, with recognition rates even lower than that of hand-printed text. Higher rates of recognition of general cursive script will likely not be possible without the use of contextual or grammatical information. For example, recognizing entire words from a dictionary is easier than trying to parse individual characters from script. Reading the Amount line of a cheque (which is always a written-out number) is an example where using a smaller dictionary can increase recognition rates greatly. Knowledge of the grammar of the language being scanned can also help determine if a word is likely to be a verb or a noun, for example, allowing greater accuracy. The shapes of individual cursive characters themselves simply do not contain enough information to recognize all handwritten cursive script accurately (Sharma *et al.*, 2013). It is necessary to understand that OCR technology is a basic technology also used in advanced scanning applications. For more complex recognition problems, intelligent character recognition systems are generally used such as HMM, SVM and ANN. The artificial neural networks can be more advantageous and can be made indifferent to both affine and non-linear transformations.

OCR for Latin Language:

The most widely used languages such as English, Spanish and French are all derives from Latin script and even Mizo language is also based on the Latin script. Therefore some of literature survey has been conducted with recent development of Latin OCR which are highlighted as below-

In 2012, Rashid *et al.* proposed a segmentation free text line recognition approach using multi-layer perceptron (MLP) and Hidden Markov Models (HMMs) in

which 98.4% character recognition accuracy was achieved.

Patel *et al.* (2012) compare Open source Tesseract OCR and proprietary Transym OCR for recognition of vehicle number in which the colour image is converted into gray scale image. The Tesseract provides better accuracy of 61% for color image and 70% for gray scale images as compared to Transym, which provides only 47% of accuracy. They have also concluded that the Tesseract is faster than Transym because it takes average 1 second and 0.82 seconds for processing color and gray scale images respectively to process one image, while Transym takes average 6.75 seconds to process one image.

George and Nicolai (2013) proposed a trainable filter called Combination of Shifted Filter Responses (COSFIRE) for recognition of English handwritten digits. The proposed COSFIRE filters provided effective machine vision solutions in three practical applications: the detection of vascular bifurcations in retinal fundus images (98.50 percent recall and 96.09 percent precision), the recognition of handwritten digits (99.48 percent correct classification), and the detection and recognition of traffic signs in complex scenes (100 percent recall and precision).

Prasad *et al.* (2013) proposed simplistic approach for recognition of offline handwritten English alphabets using Artificial Neural Networks and they obtained the recognition rate of 98.10%.

Dhiman and Singh (2013) carried out a comparative study on two most popular open source OCR such as Tesseract and GOCR for recognition of Latin script. They have found that the Tesseract OCR has better accuracy of 97.4% and precision of 97.4% on colour image than GOCR with accuracy of 64.1% and precision of 89.2%.

Breuel *et al.* (2013) presents Long Short-Term Memory (LSTM) networks for recognition of machine-printed Latin and Fraktur script. They achieved error rates of 0.15% (Fontane) and 1.47% (Ersch-Gruber). The recognition accuracies were found satisfactory without using any language modelling or any other post-processing techniques.

OCR for Indian Languages:

The major Indian languages OCR have been developed under the aegis of Technology Development for Indian Languages (TDIL) Programme, Ministry of Communications and Information Technology, Govt. of India, for Bangla, Devanagari, Gurumukhi, Kannada, Malayalam, Tamil, Telugu, Gujarathi, Oriya, Tibetan, Assamese, Manipur and Urdu. This section is to provide an overview of the research going on in Indian script OCR systems.

Biswas *et al.* (2012) proposed stroke based feature extraction and HMM based character classifier for online handwritten Bangla characters. They obtained the character level recognition accuracy of 91.85% on the test set of 8,616 samples.

Kumar *et al.* (2012) have used the Ant-miner algorithm (AMA) for offline character recognition of hand written Oriya scripts, popularly known as Utkal lipi. The AMA is a rule-based approach and the rules are incrementally tuned during the training. The average recognition rate is above 90%.

John and Balakrishnan (2013) proposed a handwritten character recognition system for **Malayalam** language. They have uses a combination of gradient and curvature feature in reduced dimension as the feature vector and SVM with Radial Basis Function (RBF) kernel as classifier. They obtained 96.28% and 97.96% of accuracy in

two different datasets.

Kumar *et al.* (2013) proposed neural network based approach for recognition of **kannada** handwritten character and they obtained the recognition accuracy of 99.58%.

Agarwal and Hemarjani (2013) proposed template matching algorithm for recognition of handwritten **Devanagari** script and they obtained 92.66% accuracy for Handwritten Devanagari characters.

Kumar *et al.* (2014) applied water reservoir based technique for identification and segmentation of touching characters in handwritten Gurmukhi words. The touching characters are segmented based on reservoir base area points. They have achieved 93.51% accuracy for character segmentation with this method.

Prasad and Kulkarni (2015) proposed weighted k-NN classifier and mean $\chi 2$ distance measure for recognition of handwritten Gujarati characters and they obtained 86.33 % recognition efficiency.

CHAPTER 2

PREPROCESSING METHODOLOGY

The preprocessing is a preliminary processing step to make the raw data usable for segmentation, feature extraction and classification. The proposed preprocessing implementation methodology is depicted in the following figure.

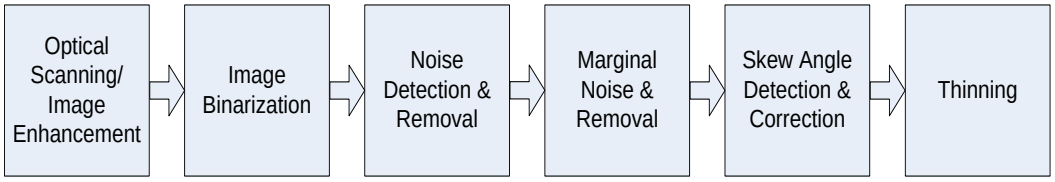


Figure 2.1: Pre-Processing of Mizo OCR

2.1 OPTICAL SCANNING

The image is acquired through a scanner, digital camera or any other suitable digital input devices (Prasad *et al.*, 2013). The recommended best scanning resolution for OCR accuracy is 300 dpi. Higher resolutions do not necessarily result in better accuracy and can slow down OCR processing time. The resolutions below 300 dpi may affect the quality and accuracy of OCR results. The image should have specific format such as jpg, bmp, tiff, etc. There are a number of factors that affect the accuracy of text recognized which include - scanner quality, scan resolution, type of printed documents (laser printer or photocopied), paper quality, fonts used in the text, linguistic complexities, and dictionary used. In this work, the scanner Epson L210 with software is used for image acquisition which is good enough for scanning resolution of 300 dpi.

2.2 IMAGE ENHANCEMENT METHODS

The scanned image, sometimes, vary the level of contrast due to poor illumination or improper setting in the acquisition sensor device. Low-contrast images can result from poor illumination, lack of dynamic range in the scanning devices, or even wrong setting of the scanner during image acquisition. The primary objective of image contrast enhancement is to differentiate between the foreground object and the background object so as to enable to perform better preprocessing results in our character recognition system. Image contrast enhancement approaches fall into two broad categories: *spatial domain methods* and *frequency domain methods*. The term spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. Frequency domain processing techniques are based on modifying the Fourier transform of an image.

In this research work we used spatial domain methods for image enhancement. Spatial domain methods are particularly useful for directly altering the gray level values of individual pixels and hence the overall contrast of the entire image. Here we examine different types of spatial domain for selection of the best image enhancement in our character recognition system.

2.2.1 LOGARITHMIC TRANSFORMATION

Log transformation is one of the elementary image enhancement techniques of the spatial domain that can be effectively used for contrast enhancements of dark images. The log transform is essentially a grey level transform which means that the grey levels of image pixels are altered. This transformation maps a narrow range of low grey level values in the input image to a wider range of output levels. The general form of the

log transformation can be mathematically represented as below (Maini and Aggarwal, 2010).

$$s = c * \log(1 + r)$$

Where ***s*** is the output value, ***r*** is the input value and ***c*** is a constant. This transformation maps a narrow range of low gray-level values in the input image into a wider range of output levels. The opposite is true of higher values of input levels. We would use this transformation to expand the values of dark pixels in an image while compressing the higher-level values. To expand the bright levels we would use the inverse logarithmic transformation.

ALGORITHM

Step 1: Read the input image

Step 2: Generate input image process by class double the input image

Step 3: Generate output image process by using Logarithmic function

Step 4: Transform the output image by converting into 8 bit image

Step 5: Show input and output histogram and image

EXPERIMENTAL RESULTS

The logarithmic transformation gives more details in the darks areas making the image lighter and the light areas lost its details. The algorithm is very fast and the average time is 0.0392 seconds for a 240x320 image. The figure below illustrated the result of logarithmic transformation in terms of output image and output histogram.

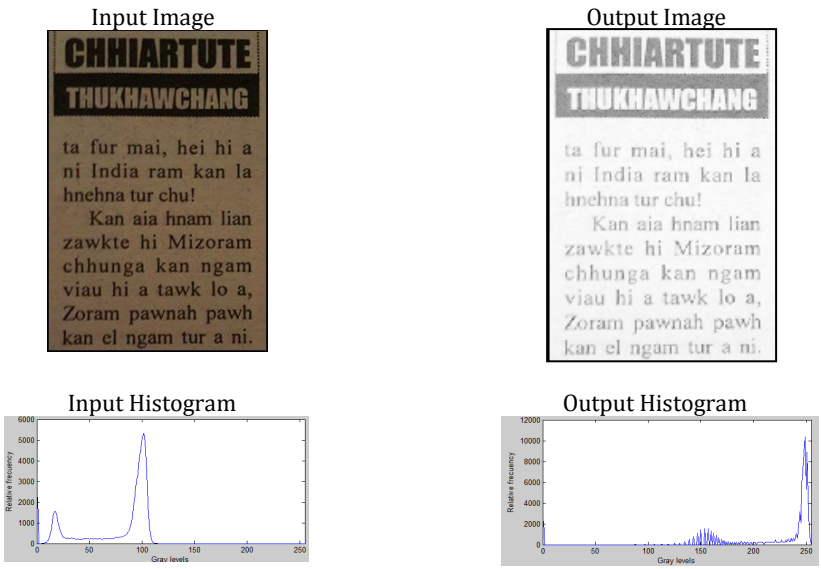


Figure 2.2: Logarithmic transformation

2.2.2 POWER-LAW TRANSFORMATION

Power law transformation is another commonly used gray level transformation in the spatial domain. It is conceptually similar to alpha rooting in the frequency domain as this is done by raising the input grey level by some power. The general form of the Power-law transformation can be mathematically represented as below (Jayaraman *et al.*, 2009).

$$s = c * r^{\gamma}$$

Where ***s*** is output grey image, ***c*** is scaling constant and ***r*** is input grey image, and γ is the power constant to which the input grey level is raised.

ALGORITHM

Step 1: Read the input image

Step 2: Generate input image process by class double the input image

Step 3: Set the value of Gamma = 2 (if gamma<1, its increase the contrast in dark

and if gamma>1, it will increase the contrast in whites)

Step 4: Generate output image process by power law function

Step 5: Transform the output image by converting into 8 bit image

Step 6: Show input and output histogram and images

EXPERIMENTAL RESULTS

The power-law transformation have two ways to operate it depends of the gamma value. If gamma is lowest than 1 it is more or less like an logarithm transform but much better because it can be use different slopes to the function. Thus, the dark areas are enhanced and more detailed. And if gamma is higher than 1, the function does the inverse result. It is enhanced the light areas and makes the image darker. The algorithm average time is 0.1232 seconds for a 240x320 image. It is slower than the logarithm transform but gets better results. The figure below illustrated the result of power law transformation in terms of output image and output histogram.

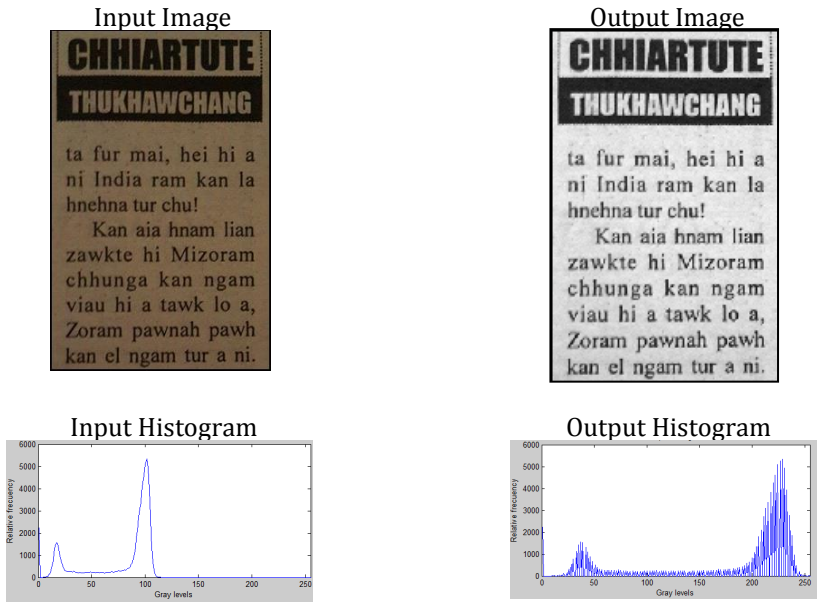


Figure 2.3: Power Law Transformation

2.2.3 HISTOGRAM EQUALIZATION

Histogram equalization is a common technique for enhancing the appearance of images. The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better detail in photographs that are over or under-exposed (Russ, 2011).

$$s_k = \frac{(L-1)*(r_k - r_{kmin})}{(r_{kmax} - r_{kmin})} \qquad k = 0, 1, 2 \ldots\ldots\ldots L\text{-}1$$

Where ***r*** and ***s*** are the input and output pixels of the image, ***L*** is the different values that can be the pixels, and $\boldsymbol{r}_{kmax}$ and $\boldsymbol{r}_{kmin}$ are the maximum and minimum gray values of the input image.

ALGORITHM

Step 1: Read the input image

Step 2: Generate input image process

Step 4: Generate output image process by using histogram equalization function

Step 5: Transform the output image by converting into 8 bit image

Step 6: Show input and output histogram and images

EXPERIMENTAL RESULTS

Histogram equalization makes the histogram to expand between all the range (0,255) and gets more smooth transitions between the pixels of the image. The algorithm average time is 0.1590 seconds for a 240x320 image. It is quite fast but the results are not good. The figure below illustrated the result of histogram equilization in terms of output image and output histogram.

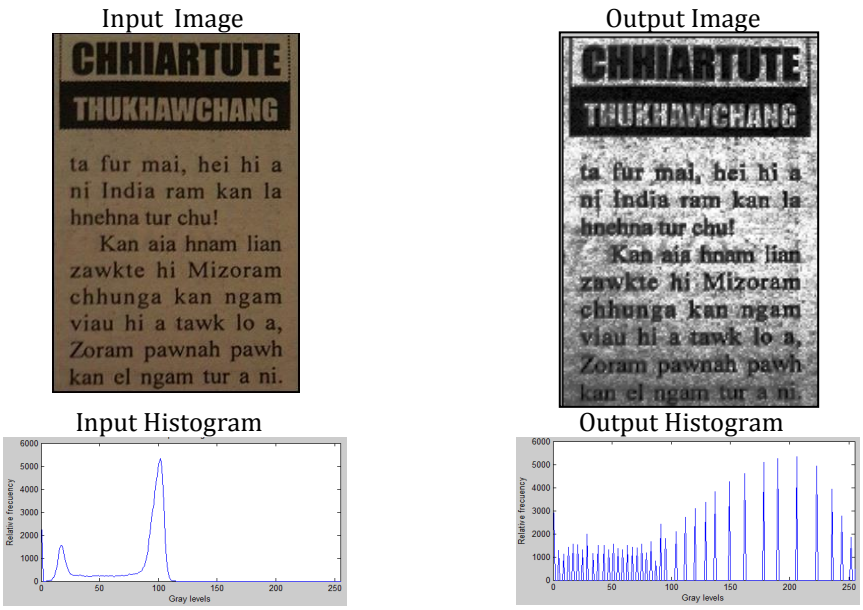


Figure 2.4: Histogram Equalization

2.2.4 CONTRAST STRETCHING

The level of contrast in an image may vary due to poor illumination or improper setting in the acquisition sensor device. Therefore, there is a need to manipulate the contrast of an image in order to compensate for difficulties in image acquisition. The idea behind contrast stretching is to increase the dynamic range of the gray levels in the image being processed. The idea is to modify the dynamic range of the grey-levels in the image. Linear Contrast Stretch is the simplest contrast stretch algorithm that stretches the pixel values of a low-contrast image or high-contrast image by extending the dynamic range across the whole image spectrum from 0 – 255 (Yang, 2006).

The general formula for contrast stretching is

$$s = \frac{1}{1 + (m/r)^e}$$

Where ***r*** is the input image values, ***s*** is the output image values; ***m*** is the thresholding value and ***e*** is the slope. If e = 1 the stretching became a threshold transformation, if e > 1 the transformation its defined by the curve which is smoother when the e value is increase, and when e < 1 the transformation makes the negative and also stretching.

ALGORITHM

Step 1: Read the input image

Step 2: Generate input image process by class double the input image

Step 3: Set the value of e = 3 and m = 80

Step 4: Generate output image process by stretching function

Step 5: Transform the output image by converting into 8 bit image

Step 6: Show input and output histogram and image

EXPERIMENTAL RESULTS

Contrast stretching separate the image in two parts the black and the white one, on the ***m*** value, and the transition between these parts is a slope that could be more or less smooth in the depends on the ***e*** value. It is also a fast algorithm and the average time is 0.0631 seconds. The transformation with the variable ***m*** = 80 and ***e*** = 3, the contrast has been enhanced with a good result for further process of the character recognition. The figure below illustrated the result of contrast stretching in terms of output image and output histogram.

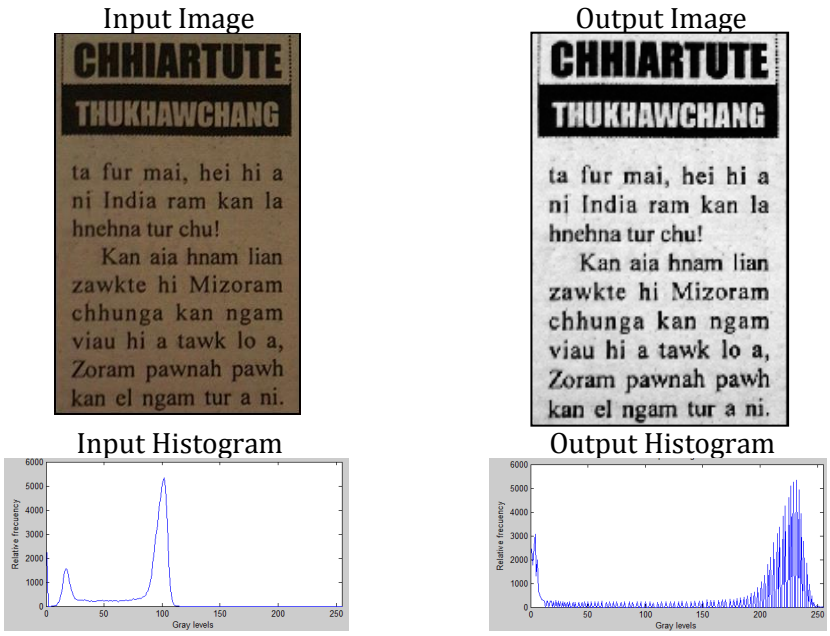


Figure 2.5: Contrast Stretching

2.2.5 DISCUSSIONS

The subjective results depend on the input image, and each transformation works better for a type of image and worse for others types. For dark input images with low contrast, the logarithm and the power law transformations gave good results with gamma lower than 1. For light input images, the power law transformation with gamma higher than 1 have given better performance. For image with low contrast in gray scale, the best method is contrast stretching.

It may be noted that the logarithm and histogram equalization methods does not required any human intervention for changing the parameters. In power law transformation, the "gamma" value is to be specified by the user. In contrast stretching the "***m***" gray scale value and the "***e***" slope of the transformation and local enhancement is the most hardworking function because it has to be calculated the limit values for the local deviation and local mean, and the size of the neighborhood.

With an image resolution of 240x320, the logarithm transformation and the contrast stretching are the fastest algorithms with an average time of 0.0392 seconds and 0.0631 seconds respectively whereas the histogram equalization transformation and the power-law transformation are the slowest algorithms with an average time of 0.1590 seconds and 0.1232 seconds.

We have concluded that the overall performance in terms of speed and image enhancement results, the contrast stretching method is the best algorithm for use in mizo character recognition system. Therefore, contrast stretching transformation for image enhancement is being implemented in this work.

2.3 IMAGE BINARIZATION

Image binarization is the process of separating the objects of an image from its background. The most common method for binarization is to select a proper threshold for the intensity of the image and then convert all the intensity values above the threshold to one intensity value ("white"), and all intensity values below the threshold to the other chosen intensity ("black"). After determining the threshold value, each pixel in the image is compared with the threshold value. If the value of the pixel is less than the threshold, reset the pixel to one. Otherwise, reset the pixel to zero as in below equation (Chaudhary and Saini, 2014).

$$P(x,y) = \begin{cases} 1 : f(x,y) \leq \text{threshold value} \\ 0 : f(x,y) > threshold\ value \end{cases}$$

Where, P(x, y) is the value assigned to the pixel after binarization step. f(x,y) is the gray value of the pixels and the threshold value 255 is the value between the dominant and the maximum value. Image having gray value for the pixels that belongs to the foreground and value 255 i.e white (0) for the background pixels. After applying the binarization algorithm on the digital image, we obtain a binary image consisting of two values 1 as black and 0 as white.

2.3.1 ALGORITHM

Step 1: Read 2D image

Step 2: Reshape the 2D gray scale image to 1D.

Step 3: Find the histogram of the image using 'hist' function.

Step 4: Initialize a matrix with values from 0 to 255

Step 5: Find the weight, mean & variance for the foreground & background

Step 6: Calculate weight of foreground* variance of foreground + weight of

background* variance of background.

Step 7: Find the minimum value.

2.3.2 EXPERIMENTAL RESULTS

In this work we present a simple and effective algorithm for converting grayscale image into binary image. The algorithm assumes that the image contains two classes of pixels following bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal. It converts gray scale image into a binary image on the basis of pixel whether it is below or above the specified threshold value. The following figure illustrates the experimental results:

(a) Original Gray Image (b) After Binarization

Figure 2.6: Image Binarization

2.4 NOISE DETECTION AND REMOVAL

The scanned documents often contain noise which generally occurred due to printer, scanner, print quality, age of the document, etc (Vithlani, 2014). The presents of noise in the scanned document can reduces the accuracy of subsequent tasks of Character Recognition systems. This noise can appear in the foreground or background of an image and can be generated before or after scanning. In this work, we encountered three types

of noises such as - Gaussian Noise, Salt and Pepper Noise, and Marginal Noise.

2.4.1 GAUSSIAN AND SALT & PAPER NOISE AND REMOVAL METHODS

The Gaussian noise is caused by random fluctuations in the signal. It is modeled by random values added to an image. In Gaussian noise, each pixel in the image will be changed from its original value by a small amount. Each pixel in the noisy image is the sum of the true pixel value and a random, Gaussian distributed noise value.

The Salt and pepper noise is also called fat-tail distributed or impulsive noise or spike noise. An image containing salt-and-pepper noise will have dark pixels in bright regions and bright pixels in dark regions. It presents itself as sparsely occurring white and black pixels. This noise arises in the image because of sharp and sudden changes of image signal. An effective noise reduction method for this type of noise is a median filter or a morphological filter.

In order to remove the above stated noises- Gaussian noise and Salt & paper noise, we examined various noise filter such as median filter, average filter, and wiener filter and proposed the best noise filter for this work.

2.4.1.1 MEDIAN FILTER

Median filtering is a nonlinear method used to remove noise from images. It is widely used as it is very effective at removing noise while preserving edges. The median filter works by moving through the image pixel by pixel, replacing each value with the median value of neighboring pixels. The pattern of neighbors is called the "window", which slides, pixel by pixel over the entire image. The median filter takes window area of an image (3x3, 5x5, 7x7, etc) which is calculated by first sorting all the

pixel values from the window into numerical order, and then replacing the pixel being considered with the middle (median) pixel value. Figure below illustrates an example of how the median filter is calculated.

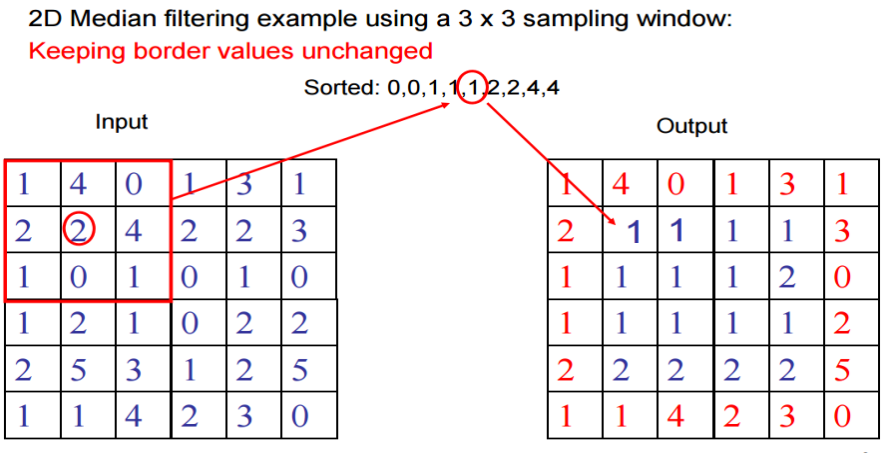


Figure 2.7: Median Filter

ALGORITHM

Step 1: Input a 2D image (mxn)

Step 2: Preallocate another matrix of size (m+2 by n+2) with zeros

Step 3: Copy the input matrix into the preallocated matrix

Step 4: Form a window matrix of size 3x3 with the elements of input matrix

Step 5: Copy the window matrix (3x3) into an array and sort it

Step 6: Find the median element. Here it is 5^{th} element. (The total elements are 9, the middle element will be 5)

Step 7: Place the 5^{th} element into the output matrix. Do the procedure for the complete input matrix.

Step 8: Convert the image into an image of 0-255

Step 9: Display the image without noise

2.4.1.2 AVERAGE (OR MEAN) FILTER

Average (or mean) filtering is a method of 'smoothing' images by reducing the amount of intensity variation between neighbouring pixels. The average filter works

by moving through the image pixel by pixel, replacing each value with the average value of neighbouring pixels, including itself. There are some potential problems:

- A single pixel with a very unrepresentative value can significantly affect the average value of all the pixels in its neighbourhood.
- When the filter neighbourhood straddles an edge, the filter will interpolate new values for pixels on the edge and so will blur that edge. This may be a problem if sharp edges are required in the output.

The figure below illustrates an example of how the Average (or Mean) filter is calculated.

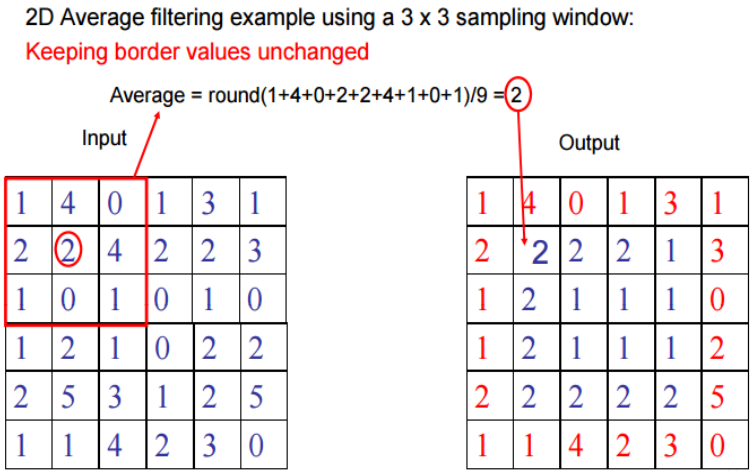


Figure 2.8: Average (or Mean) Filter

ALGORITHM

Step 1: Input a 2D image (m x n)

Step 2: Preallocate another matrix of size (m+2 by n+2) with zeros

Step 3: Copy the input matrix into the preallocated matrix

Step 4: Form a window matrix of size 3x3 with the elements of input matrix

Step 5: Copy the window matrix (3x3) into an array and sort it

Step 6: Find the Average (or Mean) element. Here the total values of the elements divided by the total elements. (The total value of elements are 15, the total

element is 9. The mean value is 15/9=1.66 which can be round of at 2)

Step 7: Place the mean value=2 into the output matrix. Do the procedure for the complete input matrix.

Step 8: Convert the image into an image of 0-255

Step 9: Display the image without noise

2.4.1.3 WIENER FILTER

The inverse filtering is a restoration technique for de-convolution, i.e., when the image is blurred by low pass filter, it is possible to recover the image by inverse filtering or generalized inverse filtering. However, inverse filtering is very sensitive to additive noise. The approach of reducing degradation at a time allows us to develop a restoration algorithm for each type of degradation and simply combine them. The Wiener filtering executes an optimal trade-off between inverse filtering and noise smoothing. It removes the additive noise and inverts the blurring simultaneously. The Wiener filtering is optimal in terms of the mean square error. In other words, it minimizes the overall mean square error in the process of inverse filtering and noise smoothing. The Wiener filtering is a linear calculating of the original image. The approach is based on a stochastic framework.

ALGORITHM

Step 1: Read the input image

Step 2: Convert the image to grayscale

Step 3: Apply the winner filter to the image

Step 4: Return the image

2.4.1.4 EXPERIMENTAL RESULTS

The scanned image documents containing Gaussian Noise and Salt & Pepper

Noise are tested with three different types of noise filter such as median filter, wiener filter and average filter. The performance of these filter are calculated using PSNR (Peak Signal to Noise Ratio) and MSE (Mean Square Error) methods (Liu *et al.*, 2011).

Where,

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{MSE} \right)$$

And

$$\text{MSE} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{i=1}^{N} (g(i,j) - f(i,j))^2$$

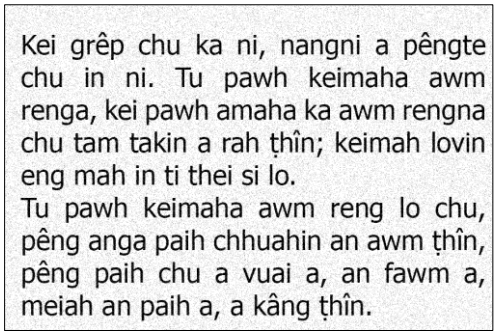
Where **M** and **N** are the total number of pixels in the horizontal and vertical dimension of image; ***g*** denotes the Noise image and ***f*** denote the filtered image (the image size is *m* x *n*).

On increasing the image size with constant impulse noise density, PSNR increases, MSE decreases. This is because the ratio of image size to noise density increases with increasing image size and constant noise, therefore the output image is better de-noised. The comparative experimental results can be seen as table below

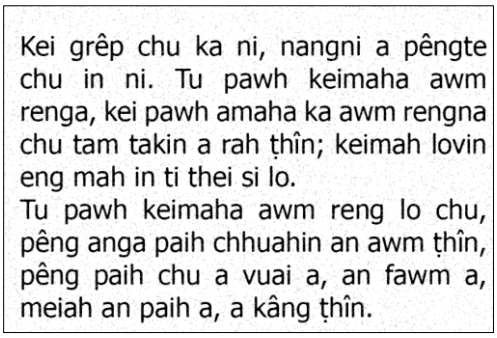
Table 2.1: Comparison of MSE and PSNR values

Noise Type	Filter Type	MSE	PSNR
Gaussian Noise	Median Filter	36.09	32.59
	Wiener Filter	137.48	26.78
	Average Filter	100.10	28.16
Salt & Pepper Noise	Median Filter	2.30	44.56
	Wiener Filter	84.12	28.92
	Average Filter	59.11	30.45

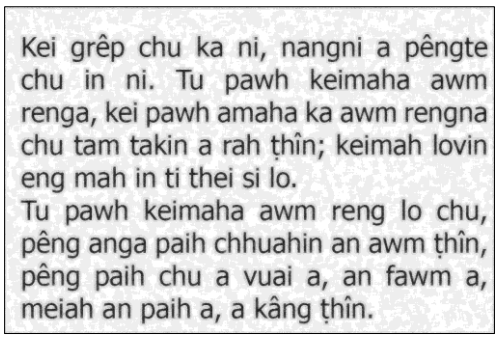
The figure below illustrated the effect of noise filters against Gaussian Noise and Salt & Pepper Noise.



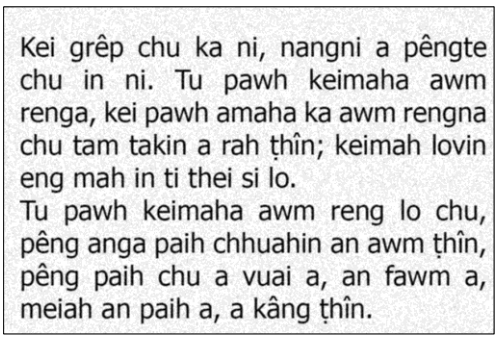
(a) Original Image with Gaussian Noise



(b) After Median Filter

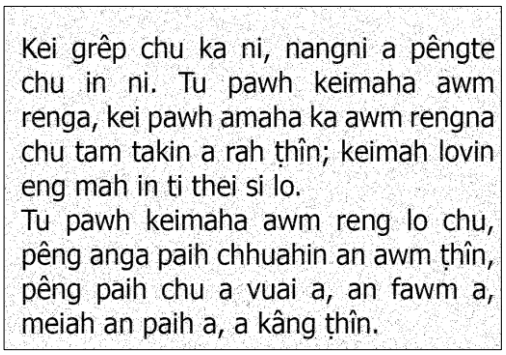


(c) After Wiener Filter

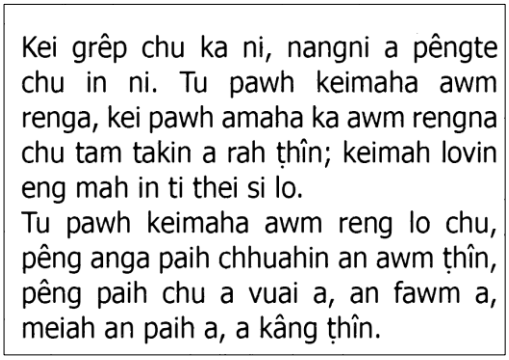


(d) After Mean Filter

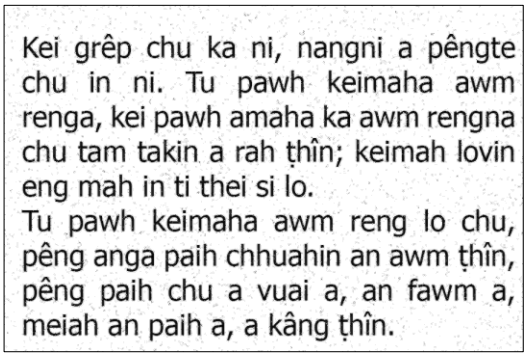
Figure 2.9: Gaussian Noise with Noise Filters



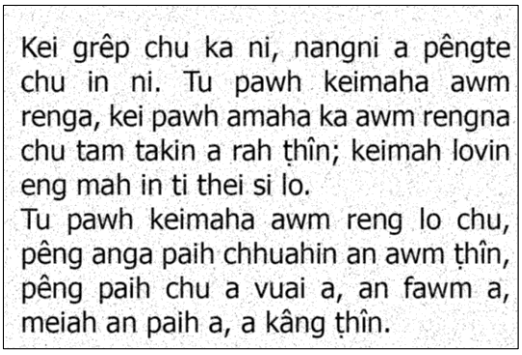
(a) Original Image with Salt-Pepper Noise



(b) After Median Filter



(c) After Wiener Filter



(d) After Mean Filter

Figure 2.10: Salt & Pepper Noise with Noise Filters.

In this thesis a comparative study of various noise filters is carried out. The performance evaluation is done by using the terms of peak signal to noise ratio and mean square error. From the experimental results it can be concluded that the median filter removal is more effective than any other noise removal both salt & pepper noise and Gaussian noise.

2.4.2 MARGINAL NOISE AND REMOVAL METHOD

When a page of a book is scanned, textual noise (extraneous symbols from the neighboring page) and/or non-textual noise (black borders, speckles) may appear along the border of the document. Different amount of noise can be present along the border of a document image depending on the position of the paper on the scanner. In general, marginal noise along the page border can be classified into two broad categories based on its source: non-textual noise (black bar) and textual noise (back ground noise which is not uniform) as depicted in the figure below.

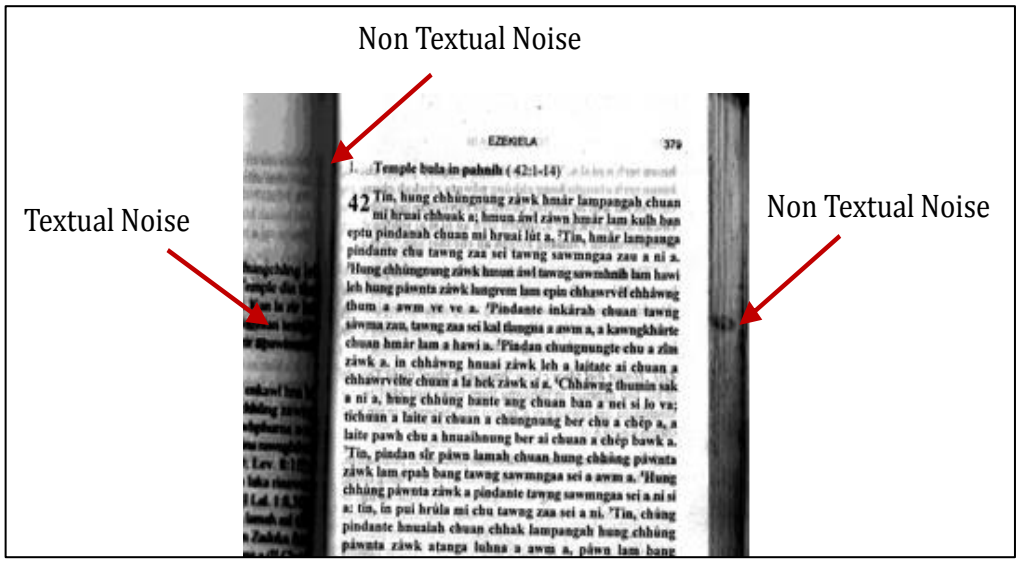


Figure 2.11: Representation of Textual and Non Textual Noise

When these noise regions are fed to a character recognition engine, it reduced the performance of character recognition system in terms of accuracy. In this work we

present a simple and effective approach for border noise removal from scanned documents. Our algorithm for border noise removal works in three steps:

Step 1: Removal of non-textual marginal noises using Connected Component Method

The first step is to remove non-textual marginal noise presents along the border of the image documents. The non-textual noises having higher density of edges than the normal text are identified using connected component analysis. First, scan the entire image document and calculate the area of each connected component. If the connected pixel is greater than the threshold value, then consider as non-textual noise. The threshold value is the area of the maximum connected pixel from a font size of 9-72 points. The non-textual noise found in the image documents is then removed.

Step 2: Removal of textual & non-textual noise presents on top and bottom of image document using Horizontal Projection Profile

The second step is to remove both textual and non-textual noise presents in the document using horizontal projection profile method. The horizontal profile method removes only the noises presents above and bottom of the desired text image document. The connected component cannot clean up all the noises along the border; it may still exhibit some noises which are equivalent or lower than the threshold value. Therefore, a horizontal projection histogram is used to remove the present of marginal noise along the top and bottom of the image documents.

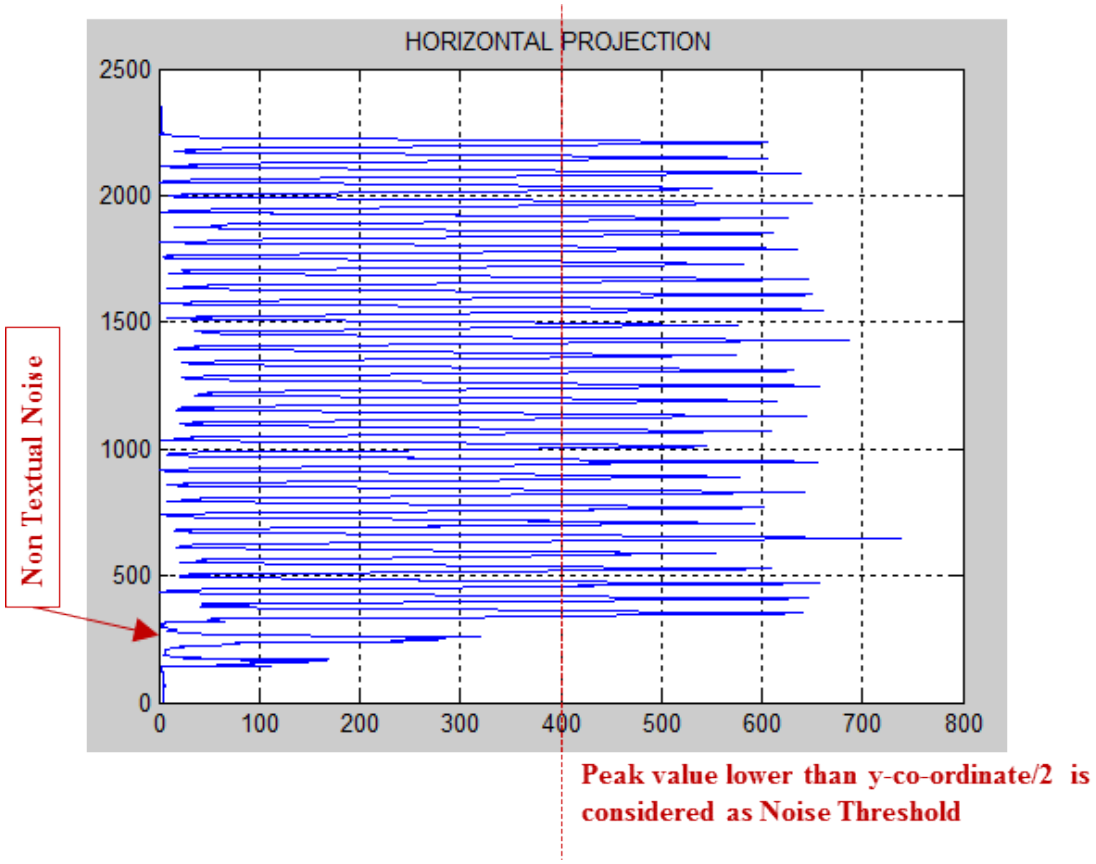


Figure 2.12: Horizontal Projection Profile

In the above figure, we found that there is a peak point of each detected blobs which is plot against the y-coordinate as shown in the figure above. An assumption is made that the non-textual noises have a peak values lower than the desire textual component. Based on this principle, a simple algorithm is prepared to remove marginal noise along the upper and lower margin of the image documents.

Step 3: Removal of textual & non-textual noise presents on left and right side of image document using Vertical Projection Profile

The third step is to remove, both textual and non–textual noises presents in the left corner and right corner of the image document. Even after performing noise removal using connected component and horizontal projection profile, some textual and non-textual noises still presents along the left and right corner of image document. These noises can be removed by using vertical projection profile method.

In this process, a rectangle window is projected vertically to find the pixels along the vertical direction. The method involves construction of a vertical histogram of the image like the one shown in the figure below. Based on the peak of this vertical histogram, individual lines in the document image are separated. The vertical projection profile is calculated by summing the black pixels in each column of the image. The vertical projection profile graph contains peaks and valleys symbolizing the noises and the textual component respectively. Then construct the Vertical Histogram for the image. Using the Histogram, the starting and ending line of each plot are determined. This starting and ending point are mark throughout the image document. The length of each plot is calculated by subtracting the starting point from the ending point. The maximum length of textual component is greater than the maximum length of noises. The length of vertical histogram which is less than the actual textual component may be treated as a noise and hence removes by overwriting with background pixel. A vertical projection showing the location of the noise in the image document is illustrated below.

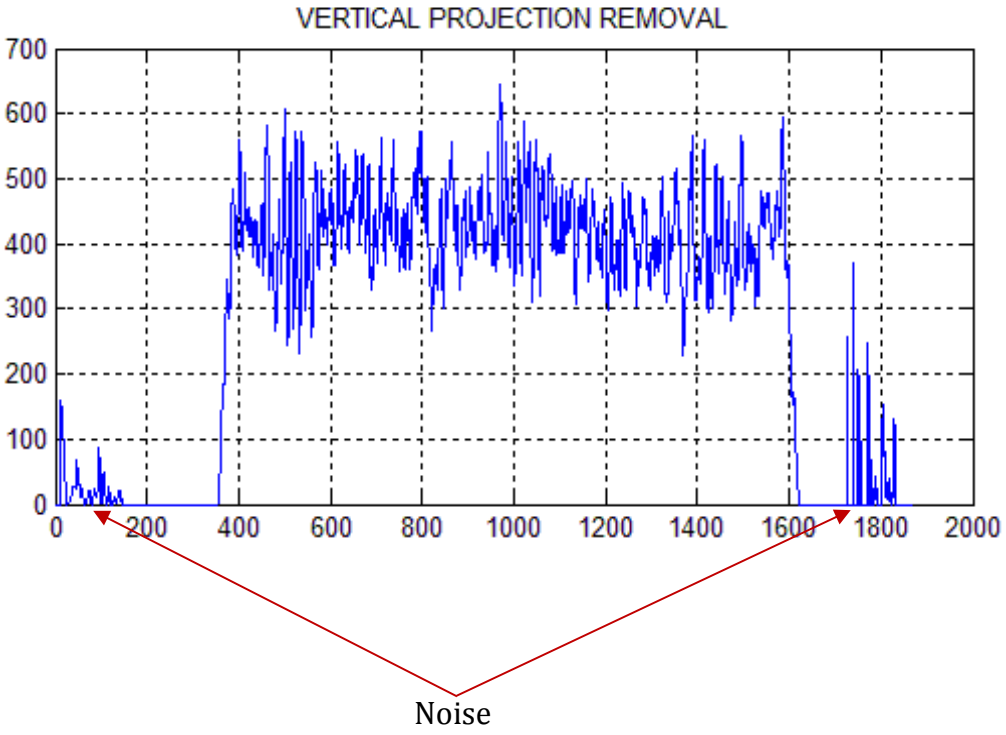
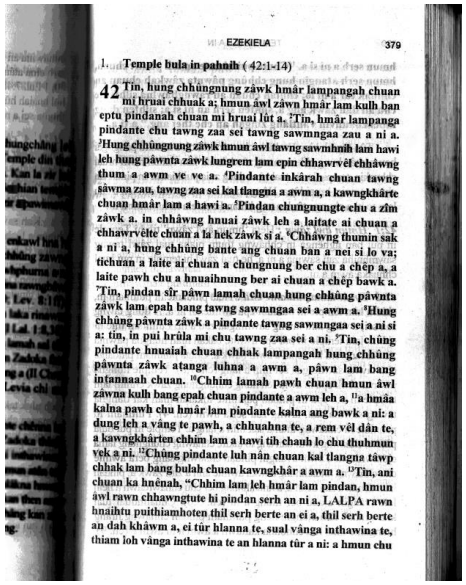
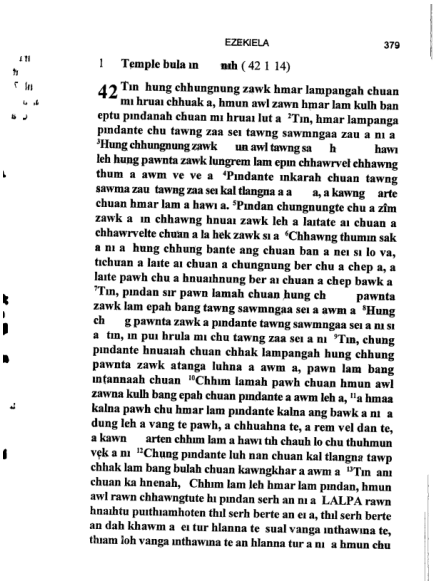


Figure 2.13: Vertical Projection Profile

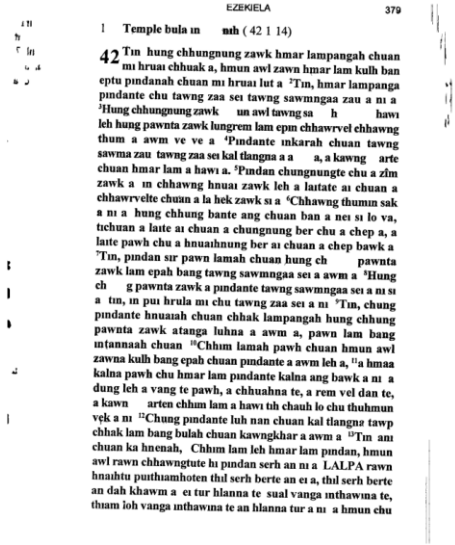
In this chapter, we presented a simple and efficient algorithm for marginal noise removal from scanned documents. The algorithm works by combining projection profile analysis with connected component to identify borders of noise regions and removal. The experimental results showing removal of marginal noises illustrated in the figure below.



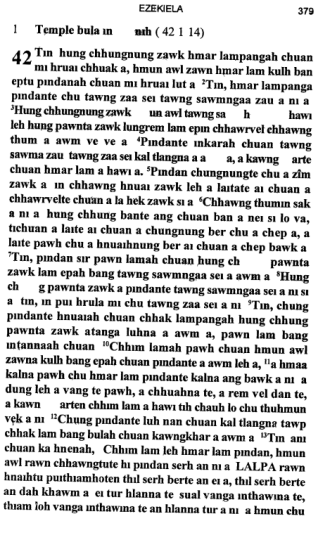
(a) Original Image Document with marginal noises



(b) After removal by Connected Component



(c) After Removal by Horizontal Projection



(d) After Removal by Vertical Projection

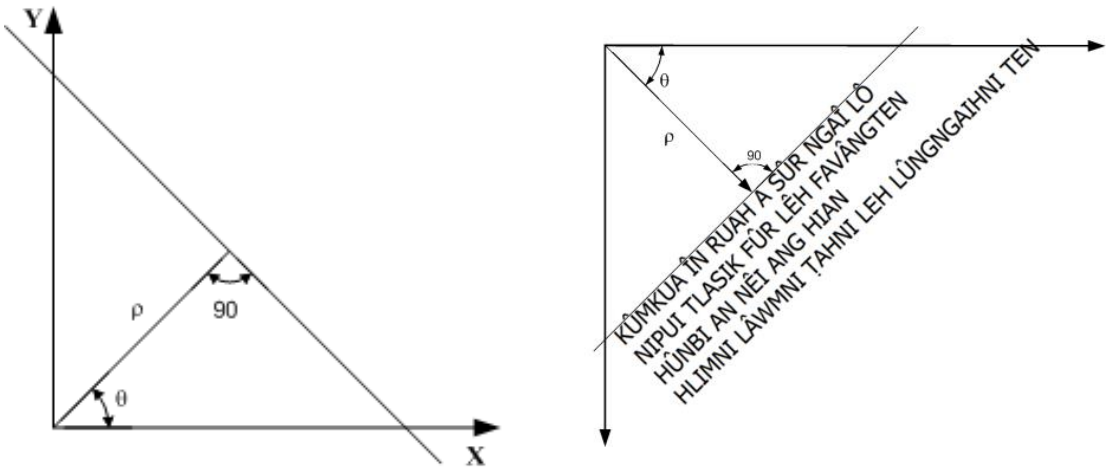
Figure 2.14: Marginal Noise Removal using Connected Component and Projection Profile

2.5 SKEW ANGLE DETECTION AND CORRECTION

When scanning the document using a flatbed scanner, it is not placed correctly the document and hence the document is skewed resulting in a skewed image document. Skew is any deviation of the image from that of the original document, which is not parallel to the horizontal or vertical. Skew Correction remains one of the vital parts in document processing. Many methods have been proposed by researchers for the detection of skew in binary image documents (Dhandra, 2006). The methods are - projection profile, Fourier transform, Hough transform, nearest neighbour, linear regression analysis and morphology. The Hough transform have more advantages than the others in terms of accuracy and simplicity. But due to slow speed, many researchers work on its speed complexity without compromising the accuracy. In this work, we introduced new method which reduces the time complexity without compromising the accuracy of Hough transform.

Hough transform is the linear transform for detecting straight lines. In the image representation there is image space, in which the straight line can be represented by equation $y = mx + b$ and can be graphically plotted for each pair of image points (x, y). In the Hough transform, the main idea is to consider the characteristics of the straight line not as image points x or y, but in terms of its parameters, here the slope parameter m and the intercept parameter b. Based on that fact, the straight line $y = mx + b$ can be represented as a point (b, m) in the parameter space. However, one faces the problem that vertical lines give rise to unbounded values of the parameters m and b. For computational reasons, it is therefore better to parameterize the lines in the Hough transform with two other parameters, commonly referred to as ρ (rho) and θ (theta). In which line can be represented Cartesian equation $x. \cos \theta_i + y. \sin \theta_i = \rho_i$. Where the parameter ρ represents

the distance between the line and the origin, and θ is the angle of the vector from the origin to this closest point. Figure 3.15 (a) shows the parameter plane of ρ and θ. In which X and Y are axis and ρ is distance and θ the angle but the Cartesian equation is slow for accumulating process than slope and intercept equations (Singh *et al.*, 2008).



(a) X-Y Parameter Plane of **ρ** (rho) and **θ** (theta)

(b) Skew Angle Detection

Figure 2.15: Hough Transformation

The Hough transform accepts the input in the form of a binary edge map and find edges which are positioned likes straight lines. The idea of the Hough transform is that every edge point in the edge map is transformed to all possible lines that could pass through that point.

Our skew detection approach is based on a technique involving Modified Hough Transform to detect the skew. In modified HT, we divide the spectrum of the HT space i.e., angle of skew which can be 0 degree to 45 degree into one-tenths, thus getting the portion in which the resultant skew lies. Then only that portion is further investigated by diving it into one-tenths and so on. This way the algorithm reaches the solution quickly as compared to the classical HT (Kumar and Singh, 2012).

2.5.1 ALGORITHMS

(1) Hough Transform Algorithm for Line detection

Step 1: Select the Hough transform parameters ρ_{min}, ρ_{max}, θ_{min} and θ_{max}.

Step 2: Quantize the (ρ,θ) plane into cells by forming an accumulator cell array $A(\rho,\theta)$, where ρ is between ρ_{min} and ρ_{max}, and θ is between θ_{min} and θ_{max}.

Step 3: Assigning the element of an accumulator cell array A to zero.

Step 4: For each black pixel in a binary image, perform the following: For each value of θ_i from min to max, calculate the corresponding ρ_i using the equation: $x.\cos\theta_i + y.\sin\theta_i = \rho_i$ Round off the ρ_i value to the nearest allowed ρ value. Updating the accumulator array element A (ρ_i, θ_i) by voting procedure.

Step 5: In last, local maxima in the accumulator cell array correspond to a number of points lying in a corresponding line in the binary image.

(2) Hough Transform Algorithm for Skew Angle Detection

Step 1: Read the input image

Step 2: Perform pre-processing for noise removal

Step 3: Perform Hong transformation on the image after preprocessing to draw straight line

Step 4: Find the peak point from the straight line in the Hong transformation

Step 5: Find the angle of bar from the peak point

Step 6: Return bar angle.

(3) Hough Transform Algorithm for Skew Angle Correction

Step 1: Read the input skew angle

Step 2: Check whether positive or negative if positive goto step 2 else goto step 3

Step 3: angle_to_rotate= -90+ skew angle

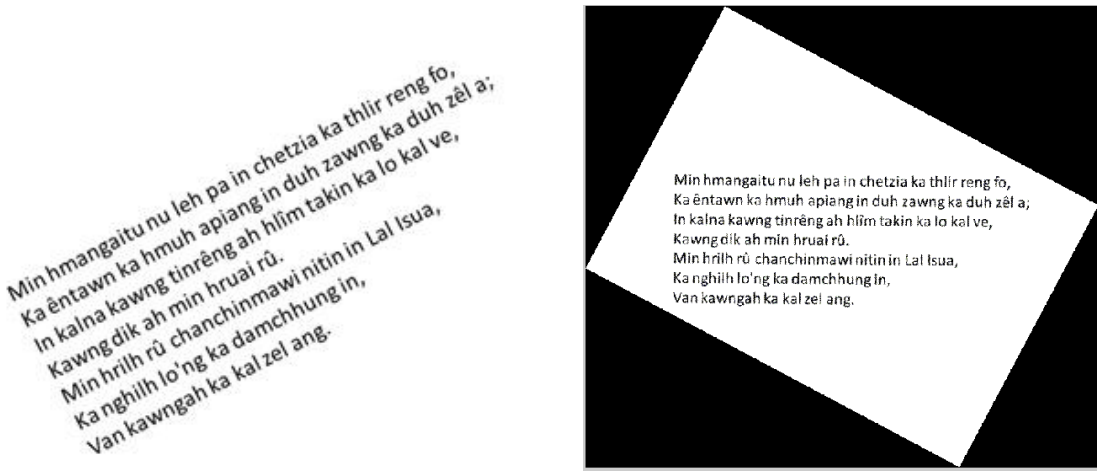
Step 4: angle_to_rotate = 90+ skew angle

Step 5: Perform image rotation by degree of angle_to_rotate

Step 6: Return rotated image

2.5.2 EXPERIMENTAL RESULTS

In this work, a sample of 20 skew angle image files has been generated for the purpose of testing the modified Hough Transform. These image files have skewed angle ranging from -30 degree to 45 degree. The following figure illustrates a sample of skew angle and de-skew image.



(a) Original Skew Angle Image (b) After Skew Angle Correction

Figure 2.16: Skew Angle Detection and Correction

The experimental result is quite satisfactory as the average accuracy is as good as 97.17% with and average error rate of 4.35% and the average execution time is 0.203 seconds. In this research work, we have adopted the modified HT because of its accuracy, simplicity and the performance speed. The experimental results can be seen at table below.

Table 2.2: The performance of Hough Transform for Skew Detection and Correction

Sl. No	Actual Angle	Detected Angle	Error	Accuracy %	Time Second
1	-30	-30.43	0.014	98.58	0.202
2	-20	-20.11	0.005	99.45	0.296
3	-10	-10.43	0.040	95.88	0.192
4	-5	-4.90	0.020	98.00	0.203
5	-3	-3.01	0.003	99.66	0.111
6	1	0.77	0.230	90.00	0.154
7	3	2.32	0.230	90.00	0.126
8	5	5.02	0.004	99.60	0.110
9	10	10.95	0.095	91.32	0.173
10	15	15.26	0.017	98.29	0.156
11	18	17.32	0.038	96.22	0.134
12	19	19.39	0.020	98.34	0.224
13	20	20.30	0.015	98.52	0.216
14	25	25.38	0.015	98.50	0.257
15	27	27.72	0.027	97.40	0.277
16	30	29.8	0.006	99.33	0.205
17	33	32.53	0.014	98.57	0.347
18	40	40.16	0.040	99.50	0.187
19	42	41.05	0.023	97.73	0.257
20	45	44.34	0.015	98.53	0.239
			0.0435	**97.17**	**0.203**

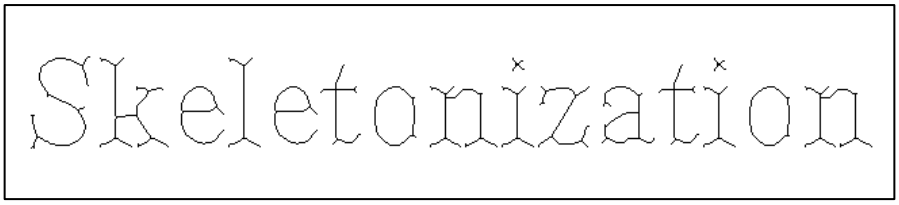
2.6 THINNING

Thinning is also known as skeletonization, it is a process of peeling off a pattern as many pixels as possible without affecting the general shape of the pattern. An effective thinning algorithm should ideally remove all redundant pixels and retain the significant aspects of the pattern under process.

There are two common approach for thinning algorithms such as - Iterative approach and Non-iterative approach. In iterative approach, pixels on the boundary are examined and successively deleted until a skeleton of one pixel width is obtained. On the other hand, non-iterative approach produces a medial line of the original image without the need of examining all pixels individually. In the proposed algorithm we follow the iterative approach, and a color coding is used in bitmap file of sixteen colors to mark, examine, preserve, delete and recovering pixels to achieve thinning and solve the problem of discontinuity yielding a very fine skeleton of the original image.

Thinning algorithm is a Morphological operation that is used to remove selected foreground pixels from binary images. It preserves the topology (extent and connectivity) of the original region while throwing away most of the original foreground pixels. The figure below shows the result of a thinning operation on a simple binary image.

(a) Original Image



(b) After thinning the image

Figure 2.17: Thinning Operation

2.6.1 ALGORITHM

The algorithm needs to follow five main steps to achieve the task of skeletonization and they are as follows:

Step 1: Start and End points marking

This is done by scanning the whole image from top-left to bottom-right corner allocating all pixels in inner and outer border of the image and distinguish those deletable from undeletable pixels.

- For undeletable pixels, the algorithm considers all on-pixels (black) which surrounded by six or seven off-pixels (white). These pixels are expected to be a start or end points on the image and hence not deletable.

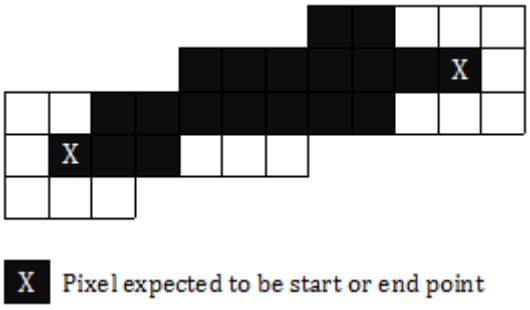


Figure 2.18: Start and end points detection

- For deletable pixels, the algorithm consider all black-pixels which

surrounded by five or eight white-pixels are noise and then delete them as shown in the Figure below.

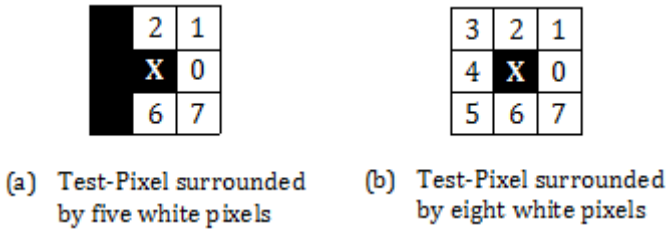


Figure 2.19: Pixels that consider as noise

Step 2: Allocation of Deletable Pixels

In this step we need to allocate all pixels on the boundary of the image that can be deleted for the sake of thinning. Allocation of these pixels should follow the rules (template) shown in the figure below.

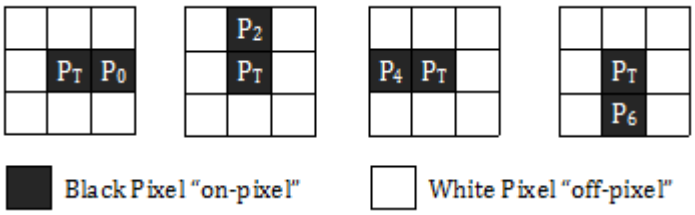


Figure 2.20: Templates for allocation of deletable pixels

Where P_T is a pixel under test and P_0, P_2, P_4 and P_6 are the four neighbor pixels of P_T in four directions according to Freeman's Code. The conditions that make PT deletable are as follows:

If {(P_2=on) & (P_6 =off) or

(P_0 =on) & (P_4 =off) or

(P_2 =off) & (P_6 =on) or

(P_0 =off) & (P_4=on)}

So P_T in all four, above mentioned, cases is deletable pixel provided that it should be connected to at least two other black pixels. Subsequently they will be mark first as deletable pixels, and later the algorithm will decide whether to delete them or not according to the conditions fulfillment. Now to avoid discontinuity there are three more rules to apply before start deleting all pixels marked as deletable pixels.

(a) The first rule is set to avoid discontinuity by making sure that all deletable pixels are not following any of patterns shown in the figure below.

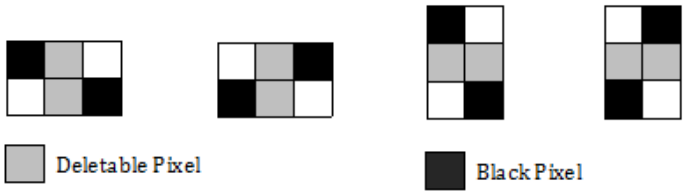


Figure 2.21: first rule for discontinuity prevention

If any of deletable pixels do fall under any of patterns shown in the above figure, one of deletable pixels should be retained. The priority of retaining a pixel goes to the deletable pixel which has more other deletable pixels connected to it than the other. However, if both of deletable pixel have the same number of other deletable pixel the priority goes to the one which leads the other according to the direction of image scanning from top-left to bottom-right. As a result, that pixel is marked as undeletable pixel.

(b) The second rule states that if a deletable pixel connected to another three deletable pixels in a manner shown in the above figure, the algorithm marks the medial pixel as a black pixel as shown in the figure below.

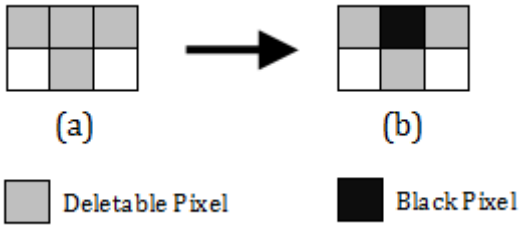


Figure 2.22: Second rule for discontinuity prevention

(c) The third rule states that any pixel which has been marked as deletable and has two white pixels at direction of (P_2 & P_6) or (P_0 & P_4) as shown in the figure below should be reverted to black pixel.

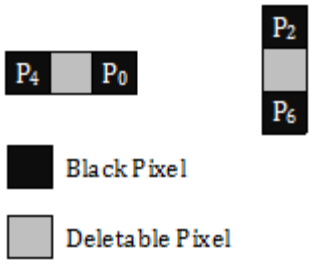


Figure 2.23: Third rule for discontinuity prevention

Step 3: Deletion Process

We shall now delete all pixels that still marked as deletable pixels. Deletion follows the scanning of the image from top-left corner to bottom-right corner. As a result of this deletion we have noticed that some discontinuities have occurred and hence we make the algorithm finish this process without any interruption and make it iterate as described in the next section till there are no more pixels to be deleted (in other word the number of deleted pixels after each iteration is same). Only then the algorithm starts checking for discontinuities and suggests proper connections.

Step 4: Iteration

The algorithm now will iterate repeating step-2 and step-3 till there are no

more deletable pixels to delete. In other word the templates in Figure 2.20 are no longer applicable. The number of iterations depends mainly on the thickness of the character in the input image.

Step 5: Discontinuity Deletion and Recovery

In case of any discontinuities in one place or another in the output skeleton, we propose a technique involves recovering of those deleted pixels which cause this type of discontinuity as following: We move a window of 3x3 on the whole thinned image and if one of the templates shown in the following figure was found, we check the missed pixel so that if it is proved that this pixel was there and, because of thinning algorithm, has been deleted we just recover that pixel back (make it black pixel), hence the problem of discontinuity is solved, otherwise we shall consider that as a deliberate discontinuity (i.e. is one of the character feature) and keep it as it is.

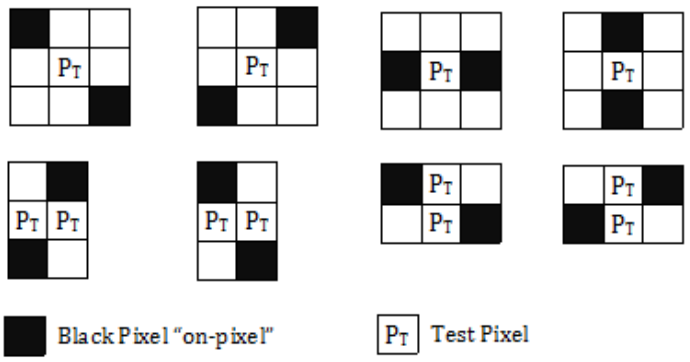


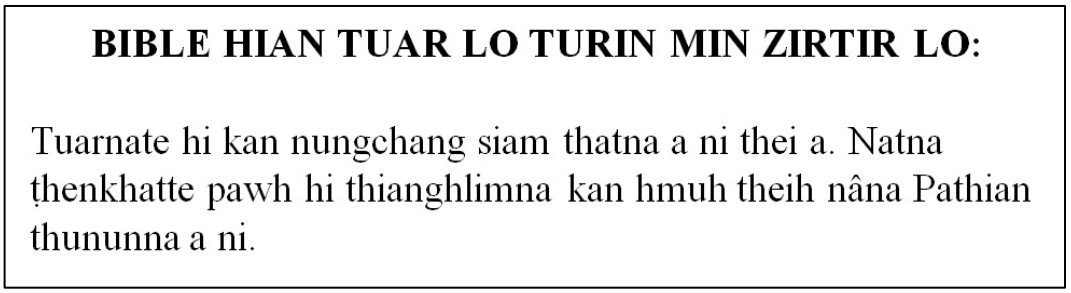
Figure 2.24: Templates for recovery of deleted pixel and preserve connectivity

Referring to the above figure, P_T is a pixel to be checked whether it was there before applying the algorithm or not, so if it was there we just convert this pixel back to black pixel otherwise we leave it as it is.

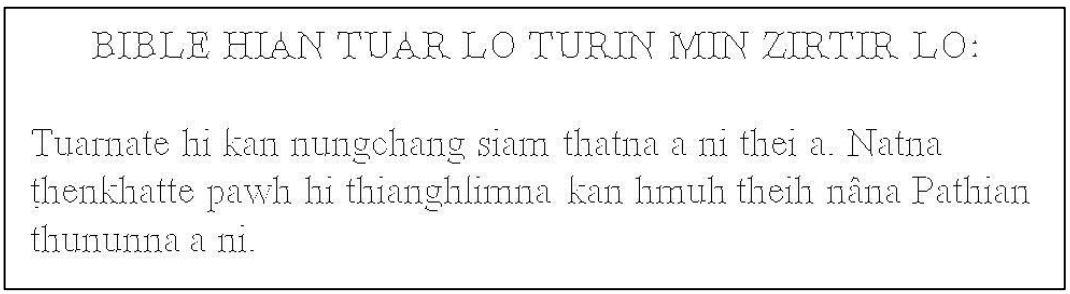
2.6.2 EXPERIMENTAL RESULTS

The algorithm was tested on different handwritten text in both cases discrete

and cursive using Epson L210 (with 800 bpi resolution) for image capturing. A preserved smooth skeleton was obtained. The following figure show examples of tests carried out on real life images document along with their output skeletons.



(a) Before Thinning Process (Skeletonization)



(b) After Thinning Process (Skeletonization)

Figure 2.25: Sample of original images document and their skeletons.

The algorithm has used six codes to represent on-pixel (black), off-pixel (white), noise pixel, start or end point pixel, deletable pixel and recovered pixel. In the propose algorithm number of 3x3 templates were used to make good deleting decision, the algorithm deletes the pixels which satisfy the deletion templates until there is no pixel that can be deleted. Other templates were also used for discontinuity recovery. The algorithm was tested on different image input data in both cases discrete and cursive. The algorithm allows us to deal with typical troublesome handwritten text efficiently, and produces robust skeleton even in the presence of noises. The algorithm produces skeletons that are more representative of the shape of the original patterns and with less

noise spurs. The algorithm is considered fast enough and very applicable to be used in Mizo OCR systems.

2.7 CONCLUSIONS

In this chapter, the image enhancement is proposed to differentiate between the foreground object and the background object so as to enable to perform better preprocessing results. The logarithmic transformation, power-law transformation, histogram equalization and contrast stretching are considered for image enhancement. The contrast stretching transformation for image enhancement is being implemented in this work. After image enhancement is over, the gray scale image is converted into binary image known as binarization for which the Otsu algorithm is being implemented.

The occurrence of noises appeared in the foreground or background of an image like Gaussian Noise and Salt & Pepper Noise are removed using median filter, wiener filter, and average filter. Their performances are evaluated using Peak Signal to Noise Ratio (PSNR) and Mean Square Error (MSE). In our experiment, we observed that the median filter performance is better than any other noise filter specially for removing salt & pepper noise and Gaussian noise. The marginal noises sometime appeared along the border of pages are removed using a combination of projection profile analysis and connected component. The skewed image documents are sometimes appeared due to incorrect placement of the document at the time of scanning process. For testing the accuracy and speed of hough transform, we have taken 20 sample skewed image file with a skew angle ranging from −30 degree to +45 degree. The experimental result is quite satisfactory as the average accuracy is as good as 97.17% with and average error rate of 4.35% and the average execution time is 0.203 seconds.

The last part of preprocessing is thinning process. The proposed thinning algorithm is tested on different sample image documents in both cases discrete and cursive. A preserved smooth skeleton was obtained. The experiment is carried out with real life images document scanning from mizo bible and mizo kristian hlabu. The proposed thinning algorithm produces skeletons that are more representative of the shape of the original patterns with less noise spurs and found satisfactory for use in the Mizo OCR systems.

[1]CHAPTER 3

SEGMENTATION METHODOLOGY

Segmentation is an integral part of any text based recognition system and is one of the most important components of the character recognition system (Acharya *et al.*, 2013). After pre-processing, the noise free image is passed to the segmentation phase, where the image is decomposed into individual characters. Accuracy of character recognition heavily depends upon segmentation phase. Incorrect segmentation leads to incorrect recognition. However, good segmentation techniques enhance the performance of an OCR (Gupta and Nair, 2013). Segmentation in any recognition system consist of dividing the script into first lines, the lines are further divided into words and the words further divided into characters from which the different modifiers & conjuncts are separated. In this section, we analyze and compare the performance of existing segmentation methods found in the literature and proposed better segmentation method for Mizo characters.

3.1 EXISTING SEGMENTATION METHODS

In this section, we analyzed the existing segmentation methods commonly used by the researchers. Such segmentation methods are projection profile, Boundary detection, and morphological operators.

Projection profile method: This method is based on the projection made by the various characters and lines in a given character image. In this method, horizontal profile is used for line segment and vertical profile is used for words and character

[1]Published in International Journal of Soft Computing and Engineering (IJSCE), "*Unicode Mizo Character Recognition System using Multilayer Neural Network Model*", 4(2):84-89 (2014).

segmentation. This method is suitable for segmenting image documents that are well spaced without overlapping and touching. Rodrigues *et al.* (2000) have used projection profile for cursive character segmentation in which he achieved 86.39% accuracy with quick response time.

Boundary Detection Method: Contour tracing, also known as boundary detection, is a technique that is applied to character image in order to extract their boundary. The most common contour tracing algorithm is Moor-Neighbour tracing algorithm and it is generally used to segment overlapping characters or symbols. The algorithm needs the coordinates of an image pixel that lies on the contour and returns the positions (row, column) of all the connected points by checking the continuity of the input pixel around its 3x3 neighbourhood. Sharma and Lehal (2006) have used boundary detection method for segmentation of isolated handwritten words in Gurmukhi script and could achieved 84.22% for words without any overlapped, connected or merged characters.

Morphological Operators: The morphological operations are affecting the form, structure or shape of an object applied on binary images (black & white images). The Erosion and Dilation are the two morphological operators whose combination or series of combination can be applied with different structuring element depending upon the size of character image. Kamble and Megha (2011) have used Morphological approach for segmentation of scanned handwritten Devnagari text in which he proposed system deals with the segmentation of modifiers and fused characters in handwritten words by segmenting the Words in hierarchical order: (a) segment the header Lines, (b) segment the top modifiers, (c) segment the bottom modifiers, (d) segment the fused characters. The experimental results achieved the accuracy of 54.83% using

morphological operator.

3.2 PROBLEMS OF SEGMENTATION OF MIZO CHARACTERS

In mizo characters, we have special characters like â Â, ê Ê, î Î, ô Ô, û Û, and ṭ Ṭ. These characters are very unique which are not available in English characters. In English, a text line can be considered as being composed of three zones: the upper zone, the middle zone and the lower zone (see figure below).

Figure 3.1: Structure of English characters text line

These zones are delimited by four virtual lines: the top-line, the upper-line, the base-line and the bottom-line. Each text lines has at least a middle zone; the upper zone depends on capital letters and letters with ascenders, like h and k; the lower zone depends on letters with descenders, like g and y. When projection profile is used for line segmentation, we do not have any problem for English text line as there is no zero valley point which is shown in the following figure.

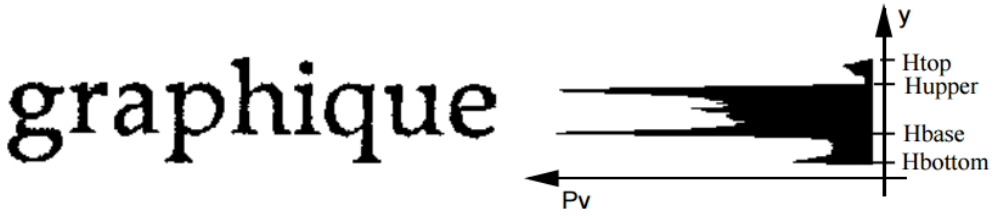


Figure 3.2: Horizontal Projection of English text line

However, in Mizo text line as shown in the figure below, there are three zones presents such as upper zone, middle zone and lower zone. These zones are delimited by six imaginary lines: the top-line upper (y_1), top-line bottom (y_2), middle-line upper (y_3), middle-line bottom (y_4), lower-line upper (y_5) and lower-line bottom (y_6).

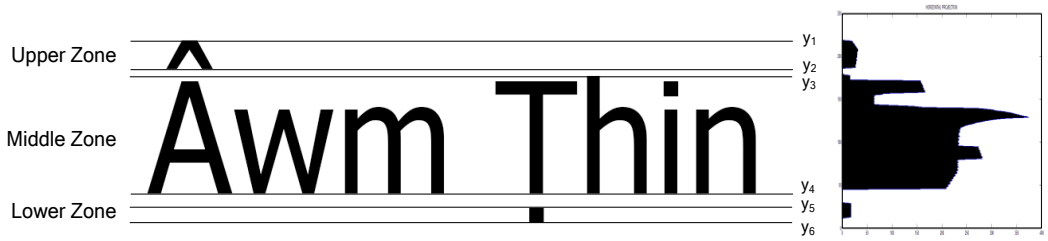


Figure 3.3: Horizontal Projection of Mizo text line

When projection profile is used for line segmentation, we have seen that there are zeros valley point between middle zone and upper zone and also between middle zone and lower zone. The zero valleys in horizontal projection profile are generally used for separation of lines. Hence, the projection profile algorithm cannot be used directly for segmentation of mizo text line.

Another problems of segmentation also arises when the characters are overlapped and touching each other in case of italic fonts and a combination of some characters like L and T, K and X, etc. The kinds of problems have been encountered during the research works.

3.3 PROPOSED SOLUTION FOR SEGMENTATION OF MIZO CHARACTERS

In this work, we have encountered problems in segmentation of Mizo characters because of some mizo characters having circumflex at the top of the characters and dotted at the bottom of the character. In order to overcome the problems, we have developed a new segmentation algorithm using a combination of projection

profile, connected component, bounding box and morphological dilation to enable to correctly segment all the mizo characters.

3.3.1 LINE SEGMENTATION:

In order to separate the text lines, we generally used the valleys point of the horizontal projection profile computed by a row-wise sum of black pixels. The position between two consecutive horizontal projections where the histogram height is least denotes one boundary line (Zramdini and Ingold, 1998). Using these boundary lines, document image is segmented into several text lines.

Figure 3.4: Structure of Mizo text line

A text line can be considered as being composed of three zones: the upper zone, the middle zone and the lower zone. These zones are all separate lines as seen from the projection histogram. The proportion of the different zones in the font size differs from one typeface to another. These zones are delimited by six virtual lines: the top-line upper (y_1), top-line bottom (y_2), middle-line upper (y_3), middle-line bottom (y_4), lower-line upper (y_5) and lower-line bottom (y_6). These structures allow the definition of four kinds of text cases which is illustrated in the figure below:

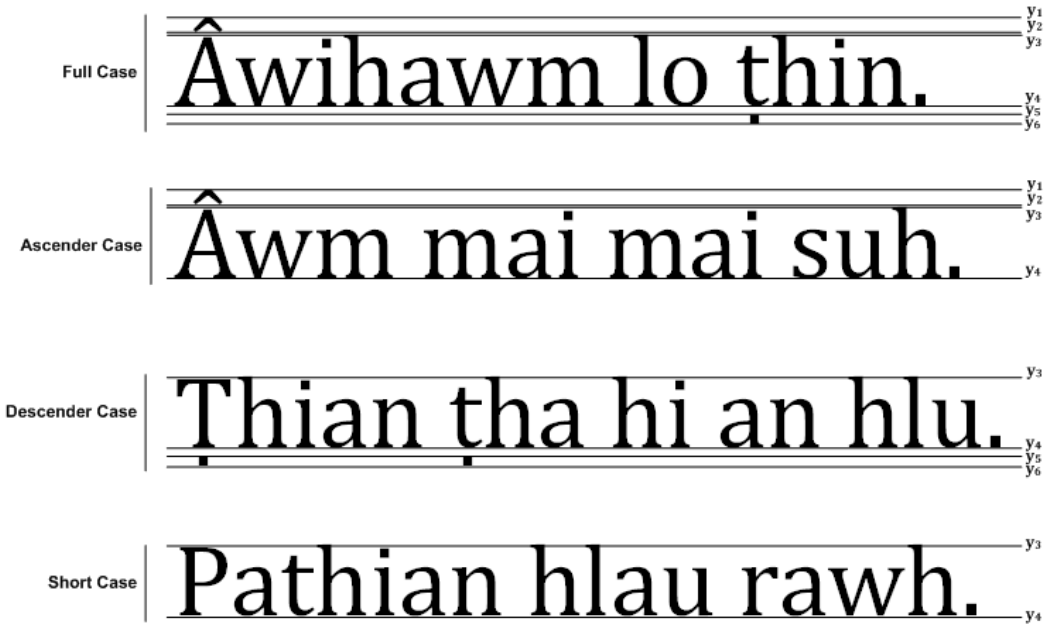


Figure 3.5: Different kind of Text cases

The four text cases are:

- **full case**, with character parts present in all three zones;
- **ascender case**, with character parts present in the upper and middle zones;
- **descender case**, with character parts present in the lower and middle zones;
- **short case**, with character parts present in the middle zone.

When the horizontal projection histograms are plotted, we can see peaks and valleys in the plot. The zero valued valleys are identified to separate the lines (Bharathi and Reddy, 2013). In mizo character, the upper zone and lower zone are treated as separate lines as there is a zero valley between upper zone and middle zone, and between middle zone and lower zone respectively. Hence, the traditional projection profile cannot be used for segmentation of mizo text line. In mizo character the three zone i.e upper zone, middle zone and lower zone are necessary to segment into a single line for which a new algorithms have been formulated.

THE LINE SEGMENTATION ALGORITHM

Step 1: Scan the preprocessed text image horizontally and find the number of ON pixels in each row.

Step 2: Plot the histogram in x direction for the ON pixel count for the image.

Step 3: Scan the histogram projection to find first ON pixel count with zero and remember that y-coordinate as y_1.

Step 4: Continue scanning the histogram projection then we would find lots of ON pixel counts to be non-zero since the characters would have started.

Step 5: Finally we get the first ON pixel count as zero and remember that y coordinate as y_2.

Step 6: Repeat Step 3-5 for 2 times and to find the next consecutive black pixel and store the co-ordinate in y_1, y_2, y_3, y_4, y_5, y_6.

Step 7: Scan the image from y_1 to y_2, y_3 to y_4, y_5 to y_6 rows for the segmented line.

Step 8: Find the differences between y_2 and y_3, y_4 and y_5 and store in diff1 and diff2 respectively which are the width of the valley point between the line segments.

Step 9: If diff1 and diff2 are smaller than threshold then scan the image from y_1 to y_6 which will be the Full line segmentation. Scan the image from y_1 to y_6 rows for the segmented line.

Step 10: If diff2 is larger but diff1 is smaller than threshold then scan the image from y_1 to y_4 which will be an ascender or decender line segmentation. Scan the image from y_1 to y_4 rows for the segmented line.

Step 11: If diff1 and diff2 are larger than threshold then scan the image from y_1 to y_2 which will be the Short line segmentation. Scan the image from y_1 to y_2 rows for the segmented line.

Step 12: Clear y_1, y_2, y_3, y_4, y_5, y_6.

Step 13: Repeat the above steps till the end of the histogram.

Step 14: Return segmented line.

3.3.2 WORD SEGMENTATION

The spacing between the words is used for word segmentation. Generally in English and Mizo script, spacing between the words is greater than the spacing between the characters in a word (Priyanka *et al.*, 2010). The spacing between the words is found by taking the Vertical Projection Profile (VPP) of an input text line. Vertical Projection profile is the sum of ON pixels along every column of the image. A sample input text line and its vertical projection profile is shown in the figure below. From the Profile it is clear that the width of the zero-valued valleys is more between the words in the line as compared to the width of zero-valued valleys that exists between characters in a word. This information is used to count and separate words from the input text lines.

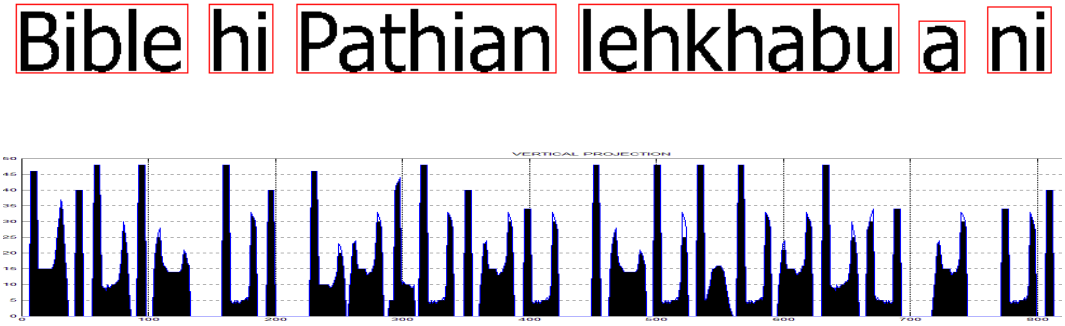


Figure 3.6: Word Segmentation

THE WORD SEGMENTATION ALGORITHM

Step 1: Read the segmented line image

Step 2: Plot the histogram in vertical direction for the ON pixel count for the input image.

Step 3: Scan the vertical projected histogram to find first ON pixel count with 1 and remember that x coordinate as x_1.

Step 4: Continue scanning the histogram projection then we would find lots of ON pixel counts to be non-zero since the characters would have started.

Step 5: Finally we get the first ON pixel count as zero and remember that x-coordinate as x_2.

Step 6: Continue Step 1 to Step 5 until the end pixel of the input line segment is reached in x-coordinate.

Step 7: Find the difference between each consecutive ON pixel

Step 8: Find the maximum value of the difference calculated in Step 7

Step 9: Divide the value obtain in step 8 by 2, which will be use as a threshold value for determining the word segment

Step 10: Loop from the First ON pixel to the Last ON pixel

Step 11: If the difference between the two consecutive ON pixel is found to be smaller than the threshold value, then merge the consecutive ON pixel until the difference is greater than the threshold value.

Step 12: Save the Merge image which form the Word Segment

Step 13: Return the Word segment

3.3.3 CHARACTER SEGMENTATION

The character segmentation process is carried out after segmented the word. The spacing between the characters is used for character segmentation. The spacing between the characters is found by taking the vertical projection profile of an input text line. The vertical projection profile is the sum of ON pixels along every column of the image. A sample input text line and its vertical projection profile is shown in Figure below. From the Profile it is clear that the width of the zero-valued valleys between the characters is lesser than the words. This information is used to count and separate characters from the input text lines.

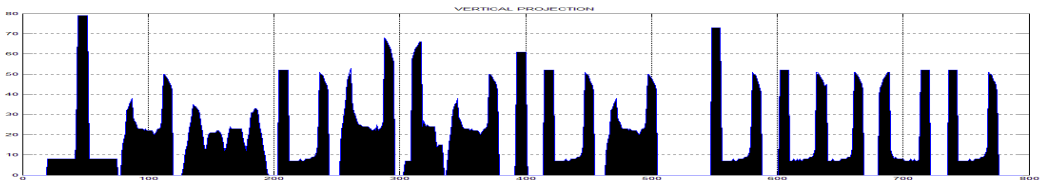


Figure 3.7: Character Segmentation

THE CHARACTER SEGMENTATION ALGORITHM

Step 1: Read the Segmented line image.

Step 2: Plot the histogram in vertical direction for the ON pixel count for the input image.

Step 3: Scan the histogram projection to find first ON pixel count with 1 and remember that x-coordinate as x_1.

Step 4: Continue scanning the histogram projection then we would find lots of ON pixel counts to be non-zero since the characters would have started.

Step 5: Finally we get the first ON pixel count as zero and remember that y coordinate as x_2.

Step 6: Segment the image from x_1 to x_2 which will form the first character and store in segmented character

Step 7: Clear x_1 and x_2

Step 8: Repeat the above steps till the end of the vertical histogram.

Step 9: Return segmented character.

In case of some of characters are touching each other, the traditional projection profile cannot be used to segment the character. As seen in the below figure, the character K and A are touching each other and hence it is treated as single character. Therefore, the traditional projection profile cannot be used directly in the touching character.

Figure 3.8: Touching Character

In order to solve the problem of touching characters, the minimum valued valley point of vertical projection profile is set as threshold value for isolation of the character. Thereafter apply the vertical projection profile for segmentation of characters. The figure below illustrated the touching characters in vertical projection profile.

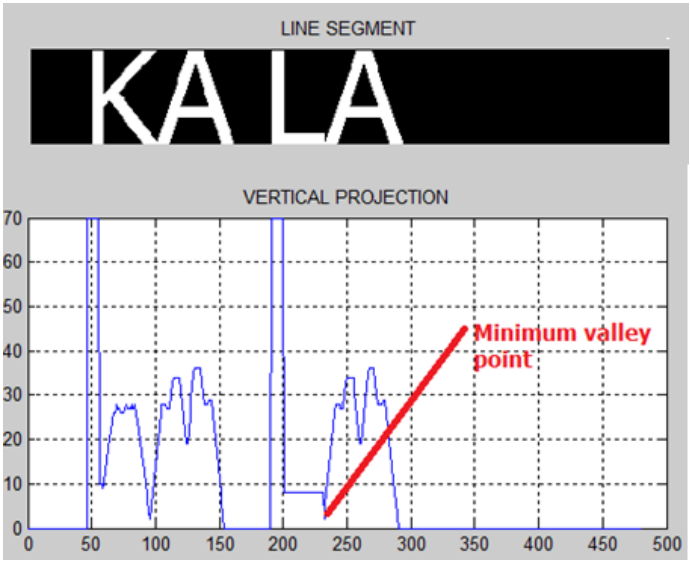


Figure 3.9: Touching Character in Vertical Projection Profile

THE TOUCHING CHARACTERS SEGMENTATION ALGORITHM

Step 1: Load the crop line image

Step 2: Perform vertical projection on the inverted image by *sum(image,1)* and store in variable '*vp*'

Step 3: Find the minimum valley point which is greater than zero from '*vp*' and store it in '*minvp*'

Step 4: Update *vp=vp>minvp* (threshold value) use for segmentation

Step 5: Assigned *starting_pixel = vp>0*

Step 6: Assigned *ending_pixel=vp<0*

Step 7: for *i=1* to length of (starting_pixel)

Step 7.1: Crop character from the *starting_pixel* to *ending_pixel*

Step 7.2: Save crop character to array of *segmented_character*

Step 8: end of for loop

Step 9: Return *segmented_character*

During the research work we have also encountered problems in overlapping characters which frequently taken place in an image document especially with italic fonts style giving unsatisfactory results. In case of overlapping character, the vertical projection profile cannot be used directly for character segmentation (Das *et al.*, 2010). The overlapping character segmentation can be overcome by using a combination of morphological dilation, connected component, bounding box. However after taking the bounding box we have to remove noise from each individual segment as some of the character segment tends to exhibit some part of their consecutive character. The figure below illustrated the overlapping Mizo characters using bounding box.

Figure 3.10: Overlapping Mizo characters

In the above figure the bounding box has been generated for each blob. The alphabet '*i*' and '*â*' have two blob each. These two blobs should be merged into a single blob so as to enable to correctly segment into a single character. In order to merge these two blobs into a single blob, morphological dilation function need to be performed. However the dilation function cannot be applied to the whole image as other characters will affect to combine into single entity. Therefore, we set a threshold value by defining double size of the smallest blobs (dot) and apply this threshold value for dilation. Before

dilation, remove all the connected component which are greater than circumflex and dotted size and apply dilation with the threshold values in the dotted and circumflex.

Figure 3.11: Dilated image over circumflex and dotted

After dilation is over, the dilated image and original image are combined together and thereafter bounding box is regenerated over the image document as shown in the figure below.

Figure 3.12: Bounding Box regenerated with dilated characters

Now the bounding box is generated correctly over the background image and the same rectangle co-ordinate is drawn to the foreground image which is illustrated in the figure below. Here, both the alphabet '*i*' and '*â'* have a single blob which is the primary requirement for correct segmentation.

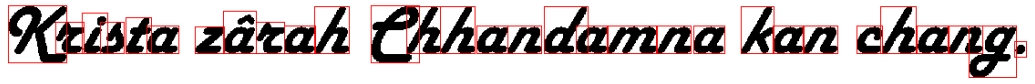


Figure 3.13: Bounding Box after dilation process

As per the bounding box generated in the above figure, all the characters are segmented accordingly. However the segmented characters have some portion of the character which is exhibited from the neighbouring characters. This exhibited portion of

characters is treated as a noise. The segmented character having noise are cleanup using simple noise removal.

Figure 3.14: Segmented overlapped Characters

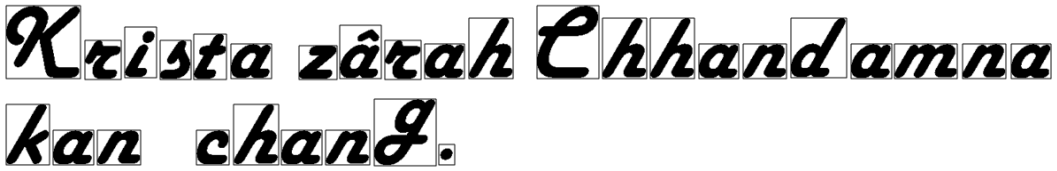


Figure 3.15: Segmented overlapped characters after cleaning up

THE OVERLAPPING CHARACTERS SEGMENTATION ALGORITHM

Step 1: Read the Segmented line image and store as a Foreground image

Step 2: Duplicate the Foreground image and store it as a Background image

Step 3: Find the size of the smallest blobs (dot) present in the Background image

Step 4: Select all the blobs present in the Background image which is smaller than the size of the smallest blobs multiply by 4, which will include all the circumflex and dot present in the character image.

Step 5: Apply Morphological dilation on the selected blobs derived in step 4 with the amount equivalent to the size of the smallest blobs derived in step3.

Step 6: Add the Result obtain in step5 with the background image so that the isolated blobs which form a single Mizo character will be merged into single entity.

Step 7: Draw bounding box on the Background image using connected component

Step 8: Draw the same bounding box on the Foreground image using the same parameter in step 7

Step 9: Segment the Character using the Bounding Box in the Foreground Image

Step 10: Remove Noise present in the Segmented Character by selecting the largest blobs from each character segment.

Step 11: Return the Noise free segmented character

3.3.4 EXPERIMENTAL RESULTS AND DISCUSSIONS

In this work, we present a simple and efficient algorithm for line segmentation, word segmentation and character segmentation. The algorithm is implemented in MATLAB 7.12 and it is tested with the test datasets - doc#1, doc#2, doc#3 and doc#4 which are collected from real-life documents such as Laser print document, Vanglaini local newspapers, Mizo Bible, and Kristian Hla Bu. Some of the documents contained overlapping characters. In order to observe the efficiency of the proposed algorithm, standard measurements have been adopted by formulating the precision which is represented as below (Gupta and Nair, 2013).

$$Precision = \left(\frac{\begin{pmatrix}\text{Total number of images (lines, words, characters)} \\ \text{correctly segmented}\end{pmatrix}}{\text{(Total number of segmented images (lines, words, characters)}} \right) x\ 100$$

The experimental results are shown in the following table.

Table 3.1: Results of Line Segmentation for Mizo text document

Document Images	No of Line present	Output of Line segmentation	No. of Line Correctly segmented	Precision (%)
TestDoc#1	31	31	31	100
TestDoc#2	32	32	32	100
TestDoc#3	18	18	18	100
TestDoc#4	12	12	12	100

Table 3.2: Results of Word Segmentation for Mizo text document

Document Images	No of Word Present	Output of Word segmentation	No. of Word Correctly segmented	Precision (%)
TestDoc#1	184	184	184	100
TestDoc#2	127	127	127	100
TestDoc#3	156	156	156	100
TestDoc#4	16	16	16	100

Table 3.3: Results of Character Segmentation for Mizo text document

Document Images	No of Character present	Output of Character segmentation	No. of Character Correctly segmented	Precision (%)
TestDoc#1	793	793	793	100
TestDoc#2	635	635	635	100
TestDoc#3	643	643	643	100
TestDoc#4	249	249	249	100

From the above experimental results, it is understood that the proposed method is reliable to segment Mizo text documents even though the text is overlapped. The average line segment accuracy is 100%, word segmentation is 100% and the character segmentation is 100%. The limitation of this method is that it resulted in segmentation errors for curve line text document which is frequently happen when a thick document is scanned.

3.4 CONCLUSIONS

Accuracy of character recognition heavily depends upon segmentation phase. Incorrect segmentation leads to incorrect recognition. In this research work, we have encountered problems in segmentation of Mizo characters due to special symbols like â,

ê, î, ô, û, and ṭ presents in every Mizo text. In order to overcome the problems, we have developed a hybrid techniques using a combination of projection profile, connected component, bounding box and morphological dilation to enable to correctly segment all the mizo characters. As a results, the proposed segmentation algorithms give a very good accuracy of 100% with four test document samples having 93 lines, 483 words, and 2320 characters. The following table compared the performance of the proposed method with the existing segmentation method.

Table 3.4: Comparison of proposed method with the existing segmentation methods

Segmentation Methods	**Subtype**	**No of characters in the datasets**	**Correctly segmented characters**	**Average Accuracy**
Projection Profile	Horizontal Projection	3788	3286	86.75%
	Vertical Projection			
Boundary Detection Methods	Contour tracing/Moore neighborhood algorithm	1673	1409	84.22%
	Bounding Box			
Morphological Operators	Erosion & Dilation	488	261	54.83%
Proposed Method	**projection profile/ connected component/ bounding box/ morphological dilation**	**2320**	**2320**	**100%**

From the above results, we observed that the proposed technique for segmentation of Mizo characters is much better than the existing segmentation method. The proposed segmentation method can also be used in any Latin based character recognition system.

CHAPTER 4

FEATURE EXTRACTION METHODOLOGY

Feature extraction is still one of the active research areas waiting for accurate recognition solutions and the accuracy of the recognition solutions is predominantly depends on proper features extraction methods. There exist many feature extraction methods which have their own advantages or disadvantages over other methods. There are several important criteria of feature extraction methods required to be considered for higher recognition rate. Firstly, an effective feature need to be invariant with respect to character shape variation caused by various writing styles of different individuals and maximize the separability of different character classes. It also needs to represents the raw image data of character through a reduced set of information which are most relevant for classification (i.e., used to distinguish the character classes) to increase the efficiency of classification process. Ease of implementation and fast extraction from raw data are also considered essential for commercial real time applications.

Selection of a feature extraction method is probably the single most important factor in achieving high recognition performance in character recognition systems. Different feature extraction methods are designed for different representations of the characters, such as solid binary characters, character contours, skeletons (thinned characters), or gray level sub-images of each individual character. A feature extraction method that proves to be successful in one application domain may turn out not to be very useful in another domain. In practice, the requirement of a good feature extraction method makes selection of the best method for a given application a challenging task. One must also consider whether the characters to be recognized have known orientation

and size, whether they are handwritten, machine printed or typed, and to what degree they are degraded.

4.1 EXISTING FEATURE EXTRACTION METHODS

A suitable feature extractor and a good classifier play a very important role in achieving high recognition rate for a recognition system. If we want to develop a new feature extractor for a script, it will help us if we have the knowledge of the recognition ability of the existing feature extractor. This section examines a variety of feature extraction approaches and classification methods which have been used in various Optical Character Recognition applications. The study has been conducted using 6 different features computed from Zoning, Projection histograms, Wavelet, Radon features, Directional features, and Moments. are considered.

Zoning Method: Zoning method is one of the most popular and simple to implement feature extraction method. The character is divided into *n* x *m* zones and the densities of pixels in each zone are calculated and used as features. Ramappa and Krishnamurthy (2013) have proposed zonal based feature extraction in which the preprocessed image is resized to 60x60 and the resized image is divided into 5x5 zones to obtain the features. A feature vector is then computed by considering the number of on pixels in each zone. For each zone if the number of on pixels is greater than 5% of total pixels, then the value one is stored for that block. The size of the feature vector is 144. After 1000 sample characters have been tested, it was reported that the recognition accuracy of 98.50% was achieved for handwritten kannada numeral.

Projection Method: In projection method, the projection histogram count the number of black pixels in the vertical direction, horizontal direction, left diagonal and right diagonal of the specified area of the character. Naser *et al.* (2009) proposed

projection based feature extraction process for Bangla script. The segmented character image is resized by 80 x 80 having 80 rows and 80 columns resulting 80 feature vectors. The feature extraction has been tested with WEKA Neural Network Classifier of Radial Basis Function. During testing, they have considered 10 characters with 12 different fonts having the total 120 characters. The accuracy for the projection of non-skeletonized characters is about 98.33%.

Wavelet Transform: The wavelet transform decomposes a signal into a set of wavelet basis functions "wavelets" that are localized in time. Therefore signals with short bursts can be reconstructed with a much smaller set of wavelet basis functions. Zhang *et al.* (2004) proposed hybrid complex wavelet feature extraction and verification of handwritten numerals. They proposed two kinds of wavelet feature extraction methods. These two sets of hybrid features are congregated by combining the respective statistical wavelet features and structural geometrical features for the recognition and verification of handwritten numerals. Experiments demonstrated that the proposed hybrid features with 180 feature vectors and 2500 sample characters can achieve high recognition performance at the rate of 98.30 %.

Radon Transform: Radon transform is used as one of the feature extraction methods. In Radon transform, 50 diverging beams are used to compute the features. It is seen from the accumulator data that the projections taken from 0 to 180 degree are exactly equal to the projections taken from 181 to 360 degree. Average value of the obtained projection data is taken to build the feature vector. Aradhya *et al.* (2007) used Radon Transform Radon Transform for robust unconstrained handwritten digit recognition in which they have extracted 703 feature vector and experimented with NN classifier with 2000 characters sample and achieved 91.2 % accuracy.

Direction Feature: Histograms of direction chain code of the contour points of the characters are used as features for recognition. The character image is first resize and is divided into 3x3 or 4x4 zones. Each zone has eight features. Mamatha *et al.* (2011) have used directional features along with K-Means for recognition of handwritten Kannada numerals in which the segmented image is first resized to 30x30 and is divided into blocks of 10x10 each. The character is divided into 9 zones with each zone has 8 features and hence each numeral have 72 features. The K-Means clustering algorithm is being used for the classification. The features used for the classification are obtained from the directional chain code information of the contour points of the numerals. The proposed algorithm is experimented on nearly 1000 samples of handwritten Kannada numerals and obtained 96% of recognition accuracy.

Moment Invariant: The Moment invariants technique is used to evaluate 7 distributed parameter of a character image. Its measures the pixel distribution around the centre of gravity of the character and allow to capture the global character shape information. Ramteke (2010) uses Invariant moments based feature extraction method for recognition of handwritten Devanagari vowels in which the segmented character image is resized into 40 x 40 pixels. The character image is divided into 4 zones with 7 invariant moments in each zone and generates 28 feature vectors. During experimentation, they have taken 10 samples of each vowel from 25 people and considered 13 vowels to generate 3250 samples characters. The success rate of the method is found to be 93.80%.

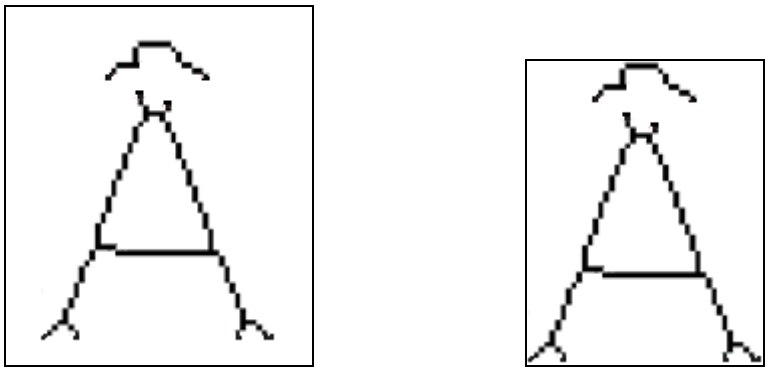
4.2 PROPOSED SOLUTION FOR FEATURE EXTRACTION

This section introduces the methodology used in designing feature extraction for the present work. Here we have proposed hybrid approach feature extraction method

using a combination of various techniques such as universe of discourse, zoning, neighborhood method, directional feature and geometric feature method (Blumenstein *et al.*, 2003).

4.2.1 UNIVERSAL OF DISCOURSE:

Universe of discourse is defined as the shortest matrix that fits the entire character skeleton. The Universe of discourse is selected because the features extracted from the character image include the positions of different line segments in the character image. So every character image should be independent of its image size.



(a) Original Image (b) Universe of Discourse

Figure 4.1: Universe of Discourse

4.2.2 ZONING

After the universe of discourse is selected, the character image is divided into small portion or zones of equal size and the densities of pixels in each zone are calculated and used as features. In this chapter, the character image is divided into 3x3 equal size of zones and feature extraction was applied to individual zones rather than the whole image. This gives more information about fine details of character skeleton. Also positions of different line segments in a character skeleton become a feature if zoning is used. This is because, a particular line segment of a character occurs in a particular zone

in almost cases. For instance, the horizontal line segment in character ‘Â’ almost occurs in the central zone of the entire character zone. The figure below illustrated the character image divided into different zones using grid size of 3x3.

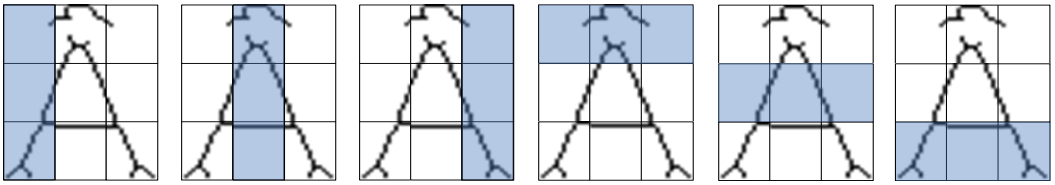


Figure 4.2: Different combination of character image divided into 3x3 equal zones

4.2.3 NEIGHBORHOOD METHOD:

The certain pixel in the character skeleton of each zone is to define as starters, intersections and minor starters for which neighborhood method is adopted. In neighbourhood method, all of the pixels that connected the pixel of interest (P) are considered to determine the starters, intersections and minor starters.

0	1	0
1	P	1
0	1	0

4 –Connected Neighbourhood

1	1	1
1	P	1
1	1	1

8 –Connected Neighbourhood

Figure 4.3: 4- and 8-Connected Neighbourhood

A. STARTERS

The pixel of interest (P) is having only **1-connected Neighbour** in the character skeleton is defined as starters. Before character traversal starts, all the starters in the particular zone is found and is populated in a list. The figure below shows that the starter

points.

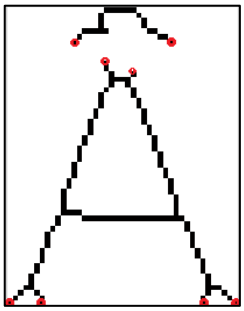


Figure 4.4: Starter Points in a red mark with rounded

B. INTERSECTIONS

The definition for intersections is somewhat more complicated. The necessary but insufficient criterion for a pixel to be an intersection is that it should have more than one neighbour. When the proposed algorithm is applied to character ‘Â’, in most cases, the intersection points found are given in the image as shown in the figure below.

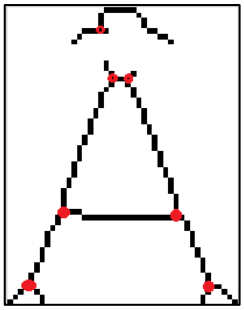


Figure 4.5: Intersection Points in a red mark with rounded

A new property called true neighbours is defined for each pixel. Based on the number of true neighbours for a particular pixel, it is classified as an intersection or not. For this, neighbouring pixels are classified into two categories, **direct pixels** and **diagonal pixels**. Direct pixels are all those pixels in the neighbourhood of the pixel under consideration in the horizontal and vertical directions. Diagonal pixels are the

remaining pixels in the neighbourhood which are in a diagonal direction to the pixel under consideration. Now for finding number of true neighbours for the pixel under consideration, it has to be classified further based on the number of neighbours it have in the character skeleton. Pixels under consideration are classified as those with;

- **3-Connected Neighbours:** If any one of the direct pixels is adjacent to anyone of the diagonal pixels, then the pixel under consideration cannot be an intersection, else if none of the neighbouring pixels are adjacent to each other then it's an intersection.

- **4-Connected Neighbours:** If each and every direct pixel having an adjacent diagonal pixel or vice-versa, then the pixel under consideration cannot be considered as an intersection.

- **5-Connected or more Neighbours**: If the pixels under consideration have five or more neighbours, then it is always considered as intersection.

Once all the intersections are identified in the image, then they are populated in a list.

C. MINOR STARTERS

Minor starters are found along the course of traversal in the character skeleton. When the proposed algorithm is applied to character 'Â', in most cases, the minor starters found are given in the image as shown in the figure below.

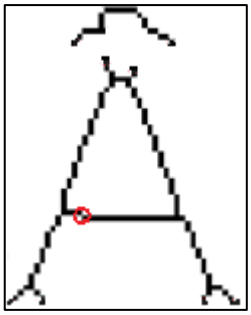


Figure 4.6: Minor Starter Points in a red mark with rounded

Minor Starters are created when pixel under consideration have more than **2-Connected Neighbours**. There are two conditions that can occur:

- **Intersections:** If current pixel is an intersection point. Then current line segment will end and all the remaining unvisited neighbours are populated in the minor starters list.

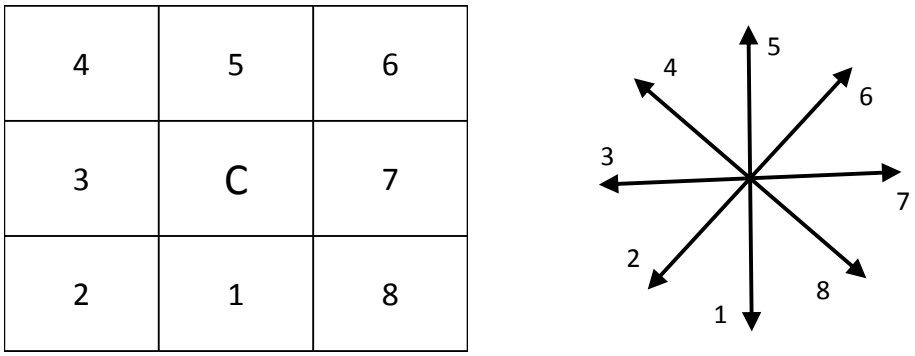
- **Non-intersections:** Situations can occur where the pixel under consideration has more than two neighbours but still it's not an intersection. In such cases, the current direction of traversal is found by using the position of the previous pixel. If any of the unvisited pixels in the neighbourhood is in this direction, then it is considered as the next pixel and all other pixels are occupied in the minor starters list. If not any of the pixels is not in the present direction of traversal, then the current segment is ended there and all the pixels in the neighbourhood are occupied in the minor starters list.

4.2.4 DIRECTIONAL FEATURE:

The directional feature is used to identify the type of line segments presents in the character image of each zone. The line segments that would be determined in each character image are categorized into four types such as – (1) Vertical lines, (2)

Horizontal lines, (3) Right diagonal, and (4) Left diagonal.

In order to distinguish the line segments, freeman chain code model are adopted for which a matric of 3x3 windows mask is prepared to determine the types of line segments. The figure below shows the chain code model.



(a) 8-Connectivity (b) Direction of Connectivity

Figure 4.7: Freeman Chain Code Model for detecting the direction of the line segments

In the above matrix, 'C' represents the center pixel which is the pixel of interest. The neighbouring pixels are numbered in a clockwise manner starting from pixel below the central pixel. To extract direction vector from a line segment, the algorithm travels through the entire pixels in the line segments in the order they forms the line segment. In order to find the type of line segment present in each zone, the matrix of 3X3 windows having the pixel of interest 'c' at the centre (black pixel) is transverse from the starters through the contour of character image in each zone. All the line segments obtained during this process are stored, with the positions of pixels in each line segment. Once all the pixels in the image are visited, the algorithm stops. The figure below illustrated the 3x1 windows mask transverse through the skeleton of character image in a zone.

For Zone 1:

Figure 4.8: Matric of 3x1mask transverse through the skeleton of character image

This kind of set of rules identified all the line segments but the drawback is that segment in the shape of character ‘A’ at the upper vertex and character ‘V’ at the lower vertex, the algorithm will detect as a single line segment though it is compose of two segments. In order to prevent such errors, a new set of rules is applied to the segment given in the diagram below.

(a) Before applying direction rules (b) After applying direction rules

Figure 4.9: Direction rules to find new line segments

The Direction rules to find new line segments:

(1) The previous direction was 6 or 2 AND the next direction is 8 or 4 OR.

(2) The previous direction is 8 or 4 AND the next direction is 6 or 2 OR.

(3) The direction of a line segment has been changed in more than three types of direction.

The line segment marked in the image was obtained before applying the direction rules explained last. Though this line segment is actually composed of two different line segments, it will be detected as one. But after applying the direction rules explained here, the two line types will be differentiated. If a new line segment is detected, then the direction vector is broken down into two different vectors at that point. Now the following rules are defined for classifying each direction vector.

(1) If maximum occurring direction type is 2 or 6, then line type is right diagonal.

(2) If maximum occurring direction type is 4 or 8, then line type is left diagonal.

(3) If maximum occurring direction type is 1 or 5, then line type is vertical.

(4) If maximum occurring direction type is 3 or 7, then line type is horizontal.

If two line types occur same number of times, then the direction type detected first among those two is considered to be the line type of the segment.

4.2.5 GEOMETRIC FEATURE EXTRACTION:

After the line type of each segment is determined, feature vector is formed based on this information (Gaurav and Ramesh, 2012). Every zone has a feature vector corresponding to it. Under the algorithm proposed, every zone has a feature vector with a length of 9.

The contents of each zone feature vector are

(1) Number of horizontal lines.

(2) Number of vertical lines.

(3) Number of Right diagonal lines.

(4) Number of Left diagonal lines.

(5) Normalized Length of all horizontal lines.

(6) Normalized Length of all vertical lines.

(7) Normalized Length of all right diagonal lines.

(8) Normalized Length of all left diagonal lines.

(9) Normalized Area of the Skeleton.

The number of any particular line type is normalized using the following method.

$$Value = 1 - \left(\left(\frac{Number\ of\ lines}{10}\right) x\ 2\right)$$

Normalized length of any particular line type is found using the following method.

$$Length = \left(\frac{(Total\ Pixels\ in\ that\ line\ type)}{(Total\ Zone\ Pixels)}\right)$$

The feature vector explained here is extracted individually for each zone. So if there are N zones, there will be 9N elements in feature vector for each zone. For the system proposed, the original image was first zoned into 9 zones by dividing the image matrix. The features were then extracted for each zone. Again the original image was divided into 3 zones by dividing in the horizontal direction. Then features were extracted for each such zone.

After zonal feature extraction, certain features were extracted for the entire image based on the regional properties namely

- Euler Number: It is defined as the difference of number of objects and number of holes in the image. For instance, a perfectly drawn ‘A’ would have Euler number as zero, since number of objects is 1 and number of holes is 2, whereas ‘B’ would have Euler number as -1, since it have two holes.

- Regional Area: It is defined as the ratio of the number of the pixels in the skeleton to the total number of pixels in the image.

- Eccentricity: It is defined as the eccentricity of the smallest ellipse that fits the skeleton of the image.

From the above study and analysis, we proposed a new algorithm for feature extraction of mizo characters as follows:

4.2.6 ALGORITHM

Step 1: Read the Segmented Character Image

Step 2: Normalized by universe of discourse for fitting the entire character image into the shortest matrix.

Step 3: Convert the Normalized image into Skelton image in order to extract a region-based shape representing the general form of an object

Step 4: Perform Zoning by dividing the character image into 3x3 windows of equal size.

Step 5: Find the Starters, intersections and minor starters in each zone and populated in a list.

Step 6: Find the type of line segments and calculate the number of line segments present in each zone where line segment are considered to be a pixel in

between starter to starter, starter to intersection and intersection to intersection.

Step 7: Calculate normalized length of all line segments using the formulae i.e length = ((total pixels in that line type)/(total zone pixels)).

Step 8: Calculate normalized area of the skeleton of character image in each zone.

Step 9: Repeat step 6 to 8 for all the zone and extract nine feature from each zone.

Step 11: Save the feature vector generated for each zone i.e 54 feature vectors

Step 12: Return 54 Feature vector.

4.2.7 EXPERIMENTAL RESULTS AND DISCUSSIONS

The proposed feature extraction algorithm has been implemented using MATLAB 7.12. The character image is divided into 3x3 equal size of zone having 9 zones. In order to improve the execution time and space complexity, we have used 6 zones instead of 9 zones having a matrix of 3x1 zones in 3 equal column and 1x3 zones in 3 equal rows. Here an attempt is made to achieve highest accuracy using less number of features and thereby improving the time and space complexity. In our experimentation, four types of line segments are generated in each zone having 24 types of line segments in each character image. The sample character ‘Â’ and ‘Ê’ with font type – Arial, Cambria, Tahoma and Times new roman is illustrated in the following table.

Table 4.1: Number of line segments present in sample character ‘Â’ and ‘Ê’

		ARIAL		CAMBRIA		TAHOMA		TIME NEW ROMAN	
		Â	Ê	Â	Ê	Â	Ê	Â	Ê
Zone 1	No of Horizontal line	1	2	1	1	3	1	2	2
	No of Vertical line	3	2	2	3	2	2	2	2
	No of Right Diagonal line	2	1	0	1	0	2	0	1
	No of left Diagonal line	0	1	0	1	0	1	0	1
Zone 2	No of Horizontal line	2	4	1	3	1	4	2	3
	No of Vertical line	4	1	2	0	2	0	3	1
	No of Right Diagonal line	0	0	3	1	2	0	0	1
	No of left Diagonal line	0	1	1	0	2	0	1	1
Zone 3	No of Horizontal line	1	3	0	2	2	4	2	2
	No of Vertical line	2	0	2	3	2	0	1	4
	No of Right Diagonal line	0	3	1	0	1	3	0	1
	No of left Diagonal line	0	1	1	3	0	0	0	0
Zone 4	No of Horizontal line	1	1	0	2	0	2	0	2
	No of Vertical line	1	0	0	0	2	0	1	1
	No of Right Diagonal line	2	3	2	1	2	1	1	1
	No of left Diagonal line	1	2	3	1	1	1	1	2
Zone 5	No of Horizontal line	0	1	0	1	0	1	0	1
	No of Vertical line	2	1	2	3	2	1	2	3
	No of Right Diagonal line	0	1	0	0	0	1	0	0
	No of left Diagonal line	0	0	0	0	0	0	1	0
Zone 6	No of Horizontal line	1	1	2	1	2	1	3	2
	No of Vertical line	3	1	1	1	4	1	3	2
	No of Right Diagonal line	1	2	1	1	1	2	0	1
	No of left Diagonal line	1	0	1	2	0	0	1	0

In order to extract the feature vector, the above number of line type are fed into the proposed algorithm to generate the normalised feature vectors. The single characte image have 54 feature vectors having 9 features in each zone. The exeperimental results of the proposed feature extraction method is depicted in the following table.

Table 4.2: Extracted 54 Features from sample character image ‘Â’ and ‘Ê’ with four different font types

		ARIAL		CAMBRIA		TAHOMA		TIME NEW ROMAN	
		Â	Ê	Â	Ê	Â	Ê	Â	Ê
Zone 1	Z 1.1	0.8	0.6	0.8	0.8	0.4	0.8	0.6	0.6
	Z 1.2	0.4	0.6	0.6	0.4	0.6	0.6	0.6	0.6
	Z 1.3	0.6	0.8	1	0.8	1	0.6	1	0.8
	Z 1.4	1	0.8	1	0.8	1	0.8	1	0.8
	Z 1.5	0.0526	0.127	0.1875	0.0877	0.1818	0.069	0.1714	0.2167
	Z 1.6	0.6053	0.7302	0.6875	0.6842	0.6364	0.6724	0.6857	0.6
	Z 1.7	0.1579	0.0159	0	0.0702	0	0.1207	0	0.0333
	Z 1.8	0	0.0317	0	0.0351	0	0.0345	0	0.0333
	Z 1.9	0.0573	0.1123	0.0523	0.1242	0.0647	0.1422	0.0528	0.107
Zone 2	Z 1.1	0.6	0.2	0.8	0.4	0.8	0.2	0.6	0.4
	Z 1.2	0.2	0.8	0.6	1	0.6	1	0.4	0.8
	Z 1.3	1	1	0.4	0.8	0.6	1	1	0.8
	Z 1.4	1	0.8	0.8	1	0.6	1	0.8	0.8
	Z 1.5	0.3934	0.6275	0.1803	0.6667	0.1698	0.8788	0.4265	0.6383
	Z 1.6	0.5082	0.2157	0.3934	0	0.1887	0	0.3088	0.0426
	Z 1.7	0	0	0.2787	0.2222	0.2453	0	0	0.0638
	Z 1.8	0	0.0196	0.0164	0	0.2264	0	0.1618	0.1277
	Z 1.9	0.092	0.0909	0.0997	0.0784	0.1039	0.0809	0.1026	0.0838
Zone 3	Z 1.1	0.8	0.4	1	0.6	0.6	0.2	0.6	0.6
	Z 1.2	0.6	1	0.6	0.4	0.6	1	0.8	0.2
	Z 1.3	1	0.4	0.8	1	0.8	0.4	1	0.8
	Z 1.4	1	0.8	0.8	0.4	1	1	1	1
	Z 1.5	0.1538	0.6471	0	0.2041	0.0968	0.6667	0.4091	0.1951
	Z 1.6	0.7308	0	0.6765	0.449	0.7097	0	0.4091	0.4146
	Z 1.7	0	0.1176	0.1176	0	0.0323	0.1	0	0.1707
	Z 1.8	0	0.0294	0.0588	0.102	0	0	0	0
	Z 1.9	0.0392	0.0606	0.0556	0.1068	0.0608	0.0735	0.0332	0.0731
Zone 4	Z 1.1	0.8	0.8	1	0.6	1	0.6	1	0.6
	Z 1.2	0.8	1	1	1	0.6	1	0.8	0.8
	Z 1.3	0.6	0.4	0.6	0.8	0.6	0.8	0.8	0.8
	Z 1.4	0.8	0.6	0.4	0.8	0.8	0.8	0.8	0.6
	Z 1.5	0.3939	0.4915	0	0.551	0	0.8511	0	0.5192
	Z 1.6	0.0909	0	0	0	0.0968	0	0.16	0.0385
	Z 1.7	0.2727	0.1017	0.1429	0.0408	0.6129	0.0213	0.28	0.1154
	Z 1.8	0.0909	0.2712	0.6429	0.3265	0.0645	0.0426	0.44	0.2115
	Z 1.9	0.0498	0.1052	0.0458	0.1068	0.0608	0.1152	0.0377	0.0927
Zone 5	Z 1.1	1	0.8	1	0.8	1	0.8	1	0.8
	Z 1.2	0.6	0.8	0.6	0.4	0.6	0.8	0.6	0.4
	Z 1.3	1	0.8	1	1	1	0.8	1	1
	Z 1.4	1	1	1	1	1	1	0.8	1
	Z 1.5	0	0.6279	0	0.4186	0	0.5833	0	0.4651
	Z 1.6	0.9412	0.2791	0.9412	0.4884	0.9412	0.3056	0.5152	0.4419
	Z 1.7	0	0.0233	0	0	0	0.0278	0	0
	Z 1.8	0	0	0	0	0	0	0.3636	0
	Z 1.9	0.0513	0.0766	0.0556	0.0937	0.0667	0.0882	0.0498	0.0766
Zone 6	Z 1.1	0.8	0.8	0.6	0.8	0.6	0.8	0.4	0.6
	Z 1.2	0.4	0.8	0.8	0.8	0.2	0.8	0.4	0.6
	Z 1.3	0.8	0.6	0.8	0.8	0.8	0.6	1	0.8
	Z 1.4	0.8	1	0.8	0.6	1	1	0.8	1
	Z 1.5	0.2881	0.587	0.3077	0.44	0.3077	0.5	0.3529	0.4528
	Z 1.6	0.2542	0.2826	0.2462	0.24	0.5192	0.3421	0.2794	0.2642
	Z 1.7	0.0678	0.0435	0.0923	0.08	0.0192	0.0526	0	0.1321
	Z 1.8	0.2881	0	0.2615	0.06	0	0	0.25	0
	Z 1.9	0.089	0.082	0.1062	0.1089	0.102	0.0931	0.1026	0.0945

The efficiency of the proposed feature extraction is generally tested with classifier. In this chapter, we have experimented with neural network classifier requiring less time for

training and testing. The performance of the network has been simulated using MATLAB 7.12 giving a satisfactory result of 99.1% accuracy. It is not only achieving higher levels of recognition accuracy but the overall time efficiency has increased as compared to the systems employing any other conventional methods of feature extraction. In this experiment, we have used testing datasets (doc#1, doc#2, doc#3, and doc#4). These datasets are extracted from real-life documents such as Laser print document, Vanglaini local newspapers, Mizo Bible, and Kristian Hla Bu. In these datasets, there are 2320 characters. The proposed method for feature extraction using a hybrid technques is much better than the existing methods.

4.3 CONCLUSIONS

Feature extraction and selection is one of the most challenging tasks in character recognition system. Different feature methods are designed for different representation of the characters which means a feature extraction method that proves to be successful in one application may turn out not to be very useful in another application. Further the type of format of the extracted features must match the requirement of the chosen classifier. Selection of feature extraction method is probably one of the most important characters for achieving high performance of the entire character recognition system. The existing feature extraction methods used by other researcher have been investigated. Such feature extraction methods are – Zoning, Projection histograms, Wavelet, Radon features, Directional features, and Moments. As a result found in the literature, the accuracy of these feature extraction methods varies from 91% to 98.5%. In this work, an attempt is made to improve the accuracy level by using a hybrid approach which is a combination of a various techniques such as universe of discourse, zoning, neighborhood method, directional feature and geometric feature method. The proposed

hybrid algorithms have been tested with 2320 characters sample dataset and achieved a very good result of 99.10 % accuracy.

The following table compared the performance of the proposed feature extraction method with the existing feature extraction method.

Table 4.3: Comparison of proposed method with the existing feature extraction methods

Feature extraction method	**Feature vector**	**No of character samples**	**Classifier**	**Accuracy (%)**
Zoning	144	1000	Artificial Immune System with K-nn (Hamming distance)	98.50
Projection Histogram	80	120	NN Classifier	98.33
Wavelet	180	2500	Neural Classifier	98.30
Radon Transform	703	2000	K-Means	91.20
Directional Feature	72	1000	K-NN	96.00
Moments	28	3250	Feed Forward BPN	93.80
Proposed Method	**54**	**2320**	**NN Classifier**	**99.10**

From the above table, we concluded that the proposed hybrid feature extraction method is much better than the existing feature extraction method and may use for implementation of mizo character recognition system.

[2]CHAPTER 5

ARTIFICIAL NEURAL NETWORK BASED APPROACH FOR MIZO CHARACTER RECOGNITION SYSTEM

This chapter introduces the basic methodology used in designing the classification of mizo character and recognition system. The output of feature extraction is a feature vector obtained from previous phase and is assigned as an input to the Classification (Recognition). The feature vectors are learned and recognized by means of supervised and unsupervised method. In a supervised classification, we present examples of the correct classification (a feature vector along with its correct class) in training the classifier. Based on these examples (also termed as prototypes, or training samples), the classifier then learns how to assign a given feature vector to a correct class. The generation of the prototypes (i.e., the classification of feature vectors/objects they represent) has to be done manually in most cases. Supervised learning is of particular use when systems under training are intended to perform tasks that have previously been performed by humans with a certain degree of success. In such cases a relation between data is known to exist, but the rules governing it are not known, or are difficult to obtain. The system to be trained effectively learns by example, generalizing the knowledge to apply it to the entire domain.

Unsupervised classification or clustering uses sample feature vectors without class labels. The classification of the feature vectors must be based on similarity between them based on which they are divided into natural groupings. Whether any two feature vectors are similar depends on the application. Obviously, unsupervised classification is

[2]Published in Science Vision, "*Artificial neural network-based approach for Mizo character recognition system"*, 14(2):61-66 (2014).

more difficult than supervised classification and supervised classification is the preferable option when possible.

The recognition (classification) method is based on feature vector which have prevailed structural methods, especially in off-line character recognition. These methods include statistical methods, Artificial Neural Network, Kernel method and Genetic algorithm. In most of the recognition (classification) method, the data set is prepared which is separated into training and testing set for every character. The overall performance of the OCR depends on the classification method which is further depends on the accuracy of feature extraction of the characters. Hence, care must be taken to select feature extraction and classification methods for implementation of OCR system.

5.1 DATA SETS USED

The following datasets are used for training, designing, and testing purposes in the present work.

5.1.1 DATASET FOR TRAINING (DATASET #1)

In the present work, training dataset is prepared in the form of binary image from the following four fonts:

1. Arial
2. Cambria
3. Tahoma
4. Time New Roman

Training dataset is prepared from 29 lowercase, 29 upper case letters, 10 numerical and 25 special characters from these four fonts. Training characters size is fixed at 36 points. Total number of prototype characters is then 93x4=372. The following

training dataset is used for training neural network. The following are the prototype characters in four popular fonts – Arial, Cambria, Tahoma, and Time new roman.

Font Name	Upper Case/Lower Case/Numerical & Special Characters
Arial	ÂÊÎÔÛȚAWBCDEFGHIJKLMNOPRSTUVZ âêîôûțawbcdefghijklmnoprstuvz 1234567890 ~!@#$%^&*-+=()[]{}:;,.‘ ’ ?
Cambria	ÂÊÎÔÛȚAWBCDEFGHIJKLMNOPRSTUVZ âêîôûțawbcdefghijklmnoprstuvz 1234567890 ~!@#$%^&*-+=()[]{}:;,.‘ ’ ?
Tahoma	ÂÊÎÔÛȚAWBCDEFGHIJKLMNOPRSTUVZ âêîôûțawbcdefghijklmnoprstuvz 1234567890 ~!@#$%^&*-+=()[]{}:;,.‘ ’ ?
Time New Roman	ÂÊÎÔÛȚAWBCDEFGHIJKLMNOPRSTUVZ âêîôûțawbcdefghijklmnoprstuvz 1234567890 ~!@#$%^&*-+=()[]{}:;,.‘ ’ ?

Figure 5.1: Prototype characters from the 4 selected fonts (Dataset #1)

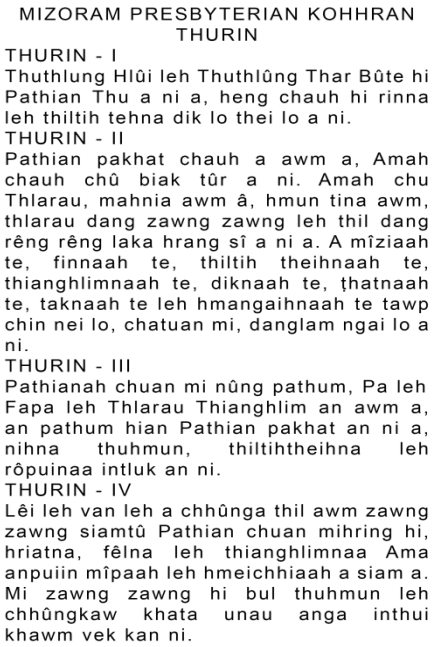
5.1.2 DATASET FOR TESTING (DATASET # 2)

We have created test data from real-life documents such as Laser print document, Vanglaini local newspapers, Mizo Bible, and Kristian Hla Bu. In order to obtain test characters in various sizes in various fonts (Arial, Cambria, Tahoma, Times New Roman), characters are typed in electronic documents in the required sizes, hard

copy prints of which are then scanned. Figure 5.2 shows an example of such a laser-printout in which 793 characters of Arial font are typed in size 12. Other sizes include 24, 48, and 72 points. Thus, sizes of our test characters vary from 12 to 72 points. Totally there are 10,919 test characters collected in this manner. This constitutes Dataset #2, which is used for testing. The following table presents the total number of test characters in each size, and in each of the five fonts.

Table 5.1: Number of Test Characters (Dataset # 2): Size wise and Font wise

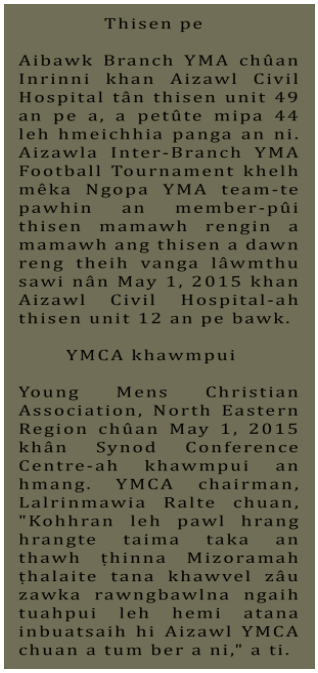
Font Name	**Document Name**	**FONT SIZE**								
		12	**18**	**24**	**30**	**36**	**48**	**60**	**72**	**Total**
Arial	DOC #1	793		793			793		793	3172
Cambria	DOC #2	635	635		635			635		2540
Tahoma	DOC #3	643		643		643	643		643	3215
Times New Roman	DOC #4	249	249	249	249	249	249	249	249	1992
TOTAL		**2320**	**884**	**1685**	**884**	**892**	**1685**	**884**	**1685**	**10919**



MIZORAM PRESBYTERIAN KOHHRAN
THURIN

THURIN - I
Thuthlung Hlûi leh Thuthlûng Thar Bûte hi Pathian Thu a ni a, heng chauh hi rinna leh thiltih tehna dik lo thei lo a ni.
THURIN - II
Pathian pakhat chauh a awm a, Amah chauh chû biak tûr a ni. Amah chu Thlarau, mahnia awm â, hmun tina awm, thlarau dang zawng zawng leh thil dang rêng rêng laka hrang sî a ni a. A mîziaah te, finnaah te, thiltih theihnaah te, thianghlimnaah te, diknaah te, ṭhatnaah te, taknaah te leh hmangaihnaah te tawp chin nei lo, chatuan mi, danglam ngai lo a ni.
THURIN - III
Pathianah chuan mi nûng pathum, Pa leh Fapa leh Thlarau Thianghlim an awm a, an pathum hian Pathian pakhat an ni a, nihna thuhmun, thiltihtheihna leh rôpuinaa intluk an ni.
THURIN - IV
Lêi leh van leh a chhûnga thil awm zawng zawng siamtû Pathian chuan mihring hi, hriatna, fêlna leh thianghlimnaa Ama anpuiin mîpaah leh hmeichhiaah a siam a. Mi zawng zawng hi bul thuhmun leh chhûngkaw khata unau anga inthui khawm vek kan ni.

(a) Laser Print document in Arial (Doc#1)



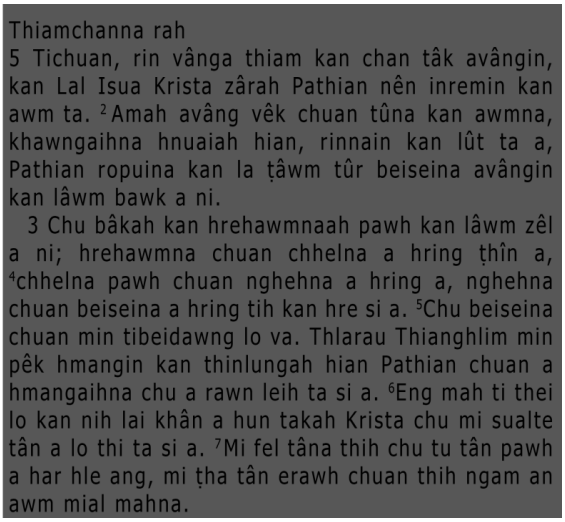
Thisen pe

Aibawk Branch YMA chûan Inrinni khan Aizawl Civil Hospital tân thisen unit 49 an pe a, a petûte mipa 44 leh hmeichhia panga an ni. Aizawla Inter-Branch YMA Football Tournament khelh mêka Ngopa YMA team-te pawhin an member-pûi thisen mamawh rengin a mamawh ang thisen a dawn reng theih vanga lâwmthu sawi nân May 1, 2015 khan Aizawl Civil Hospital-ah thisen unit 12 an pe bawk.

YMCA khawmpui

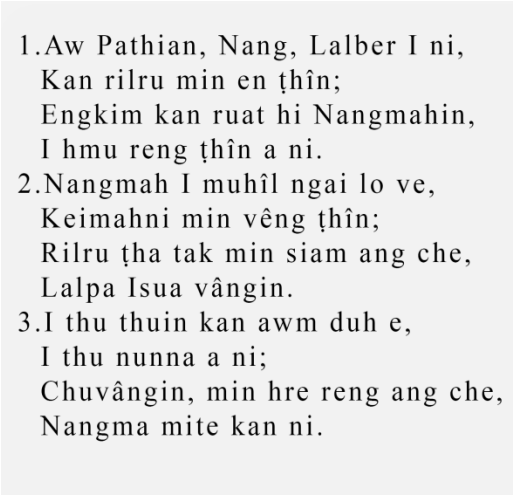
Young Mens Christian Association, North Eastern Region chûan May 1, 2015 khân Synod Conference Centre-ah khawmpui an hmang. YMCA chairman, Lalrinmawia Ralte chuan, "Kohhran leh pawl hrang hrangte taima taka an thawh ṭhinna Mizoramah ṭhalaite tana khawvel zâu zawka rawngbawlna ngaih tuahpui leh hemi atana inbuatsaih hi Aizawl YMCA chuan a tum ber a ni," a ti.

(b) Vanglaini Local Newspaper in Cambria (Doc# 2)



Thiamchanna rah
5 Tichuan, rin vânga thiam kan chan tâk avângin, kan Lal Isua Krista zârah Pathian nên inremin kan awm ta. [2] Amah avâng vêk chuan tûna kan awmna, khawngaihna hnuaiah hian, rinnain kan lût ta a, Pathian ropuina kan la ṭâwm tûr beiseina avângin kan lâwm bawk a ni.
3 Chu bâkah kan hrehawmnaah pawh kan lâwm zêl a ni; hrehawmna chuan chhelna a hring ṭhîn a, [4]chhelna pawh chuan nghehna a hring a, nghehna chuan beiseina a hring tih kan hre si a. [5]Chu beiseina chuan min tibeidawng lo va. Thlarau Thianghlim min pêk hmangin kan thinlungah hian Pathian chuan a hmangaihna chu a rawn leih ta si a. [6]Eng mah ti thei lo kan nih lai khân a hun takah Krista chu mi sualte tân a lo thi ta si a. [7]Mi fel tâna thih chu tu tân pawh a har hle ang, mi ṭha tân erawh chuan thih ngam an awm mial mahna.

(c) Mizo Bible (Rom 5:1-7) in Tahoma (Doc#3)



1.Aw Pathian, Nang, Lalber I ni,
Kan rilru min en ṭhîn;
Engkim kan ruat hi Nangmahin,
I hmu reng ṭhîn a ni.
2.Nangmah I muhîl ngai lo ve,
Keimahni min vêng ṭhîn;
Rilru ṭha tak min siam ang che,
Lalpa Isua vângin.
3.I thu thuin kan awm duh e,
I thu nunna a ni;
Chuvângin, min hre reng ang che,
Nangma mite kan ni.

(d) Kristian Hla Bu (Pg 16) in Mizo Time New Roman (Doc#4)

Figure 5.2: Testing Dataset used in the present work (Dataset #2)

5.2 EXISTING CLASSIFICATION METHODS

A classifier can be designed using a number of possible approaches (Kinhekar and Govilkar, 2014). Such approaches are broadly classified into four methods

i.e statistical methods, Artificial Neural Network, Kernel method and Genetic algorithm. The choice of a classifier is a difficult task and it is often based on which classifier(s) happen to be available, or best known, to the user. The four different types of classification techniques have been discussed as follows.

STATISTICAL METHODS: Statistical method is a rule based system for classification. Prior to application of the methods, some rules will be made based on the features which the system is going to extract for the input character image. On the basis of the rule which is matched, the output of recognition is given (Deshpande *et al.*, 2008). Some of the statistical methods are Quadratic Discriminant Function (QDF), Linear Discriminant Function (LDF), and Euclidean distance from class mean, K-NN, and Modified QDF (MQDF). Pal *et al.* (2007) have proposed a Quadratic Classifier based scheme for recognition of off-line Devanagari handwritten character data. They have tested with 36172 samples and could achieved 95.3% recognition accuracy.

ARTIFICIAL NEURAL NETWORK (ANN): A neural network is a set of connected input-output units in which each connection has a weight associated with it. During the learning phase, the network by adjusting the weights so as to be able to predict the correct class label of the input values. Some of the ANN methods generally used for classification are - Feed forward neural networks including Multi-Layer Perceptron (MLP), Radial Basis Function (RBF) Network, Back Propagation Neural Network (BPN), etc. (Shelke and Apte, 2011)

Kanale and Chitnis (2011) have used Feed forward neural network for classification of handwritten Devanagari character in which 450 sample data has been collected. Each of the sample character has 35 feature vectors which is used as an input

to the neural network. In this work, it was observed that up to 96% recognition accuracy is achieved for Devanagari characters.

KERNEL METHODS: These methods require only a user-specified kernel, i.e., a similarity function over pairs of data. In this process, the raw data is transformed into feature vector representations. This image in form of vector will be compared with the input image vector and based on matched feature the output is returned. Support Vector Machine (SVM), kernel principal component analysis (KPCA), kernel Fisher discriminant analysis (KFDA) are some classifiers based on this method.

Aggarwal *et al.* (2012) uses kernel method for recognition of handwritten Devanagari character in which gradient representation is used as the basis for extraction of features. In the proposed approach, the sample image of Devanagari characters are normalized to 90x90 pixels sizes and divided into 9x9 sub-blocks. Each sub-block has 8 standard directions. Finally the image is down sampled to 5 x 5 blocks from 9 x 9 using a Gaussian Filter giving a feature vector of dimensionality 200 (5x5x8). The experimental dataset consist of 200 samples of each of 36 basic Devanagari characters having the total of 7200 character samples. The result of 94% recognition accuracy could be achieved with Kernel method.

GENETIC ALGORITHM: These are stochastic search algorithm which uses probability to guide the search. On the unknown input binary character image, many operations are applied to extract the features of it and then with the features of the database template. This method aims for finding the global optimal solution by evaluating fitness function for each input character image.

Agnihotri (2012) proposed to use Genetic algorithm to recognize off-line handwritten Devanagari script in which the feature of each character image is converted

into chromosome bit string of length 378. More than 1000 samples is used for training and testing. An attempt is made to use the power of genetic algorithm to recognised off-line handwritten Devanagari script. The diagonal based feature extraction methods were used to extract 54 feature vectors for each character. The performance of the proposed system is 85.78%.

5.2.1 COMPARISON OF EXISTING CLASSIFICATION

In this work, we have carried out an investigation on various types of classification methods currently used in many Indian OCR applications. The performance of classification mostly depends on the nature of the pattern of the character and their feature vectors. The overview of the ongoing research works in various character recognition systems and comparison of results of the relevant works found in the literature survey are presented in following table.

Table 5.2: Comparison of the existing Classification (Recognition) Methods

Type of Classification Methods	Subtype	No of sample characters	Average Accuracy (%)
Statistical Methods *(Pal et al., 2007)*	Quadratic discriminant function (QDF)	36172	95.13 %
Artificial Neural Network *(Kanale and Chitnis, 2011)*	Feed Forward Neural Network	450	96.00 %
Kernal Methods *(Aggarwal et al., 2012)*	Support Vector Machine (SVM)	7200	94.00 %
Genetic Algorithm *(Agnihotri, 2012)*	Chromosome Function	1000	85.78 %

As per the above comparison statement, the Artificial Neural Network based approach classification give better performance results than any other classification in

terms of accuracy, adaptability and usability. In view of this, we proposed to use Artificial Neural network based Classification for Mizo character recognition system.

5.3 PROPOSED ARTIFICIAL NEURAL NETWORK BASED APPROACH FOR RECOGNITION OF MIZO CHARACTER

In this chapter, we proposed to use Artificial Neural Networks based approach for Classification of Mizo character recognition. There are many different types of Artificial Neural Network which are commonly used in character recognition system. In this section, we present four types of Artificial Neural Network such as Back Propagation Neural Network (BPNN), Radial Basis Function (RBF), Linear Vector Quantization (LVQ) and Recurrent Neural Network (RNN).

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. Neural Network can be trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision, and control systems. Neural networks can also be trained to solve problems that are difficult for conventional computers or human beings. The network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically, many such input pairs (input vectors) and target pairs (target vectors) are needed to train a network.

Here, an attempt is made to analyze various types of neural networks and compare their performance to select the best method for implementation of mizo character recognition system. The neural networks under consideration for testing their performance are Back Propagation Algorithm (BPA), Learning Vector Quantization (LVQ), Radial Basis Function (RBF), and Recurrent Neural Network (RNN). The input

vector is derived from the output of feature extraction. In order to test these networks, the total number of 93x4=372 characters sample have taken into consideration for training and 2320 characters for testing the networks. These sample characters are in four fonts - Arial, Cambria, Tahoma, and Times new roman having capital letters, small letters, numerical, and special characters. Fifty four (54) features have been extracted from each character which is used as an input vector for the input layer of the network. As there are 93 different classes in mizo characters, the output layer of the network have 93 output vectors. The algorithm of the networks is program in MATLAB 7.12 and their results are compared based upon their perfection in the character recognition.

The efficiency of classification is measures by confusion matrix and mean square errors. The confusion matrix is simply a square matrix that shows the various classification and misclassifications of the model in a compact area. The columns of the matrix correspond to the number of instances classified as a particular value and the rows correspond to the number of instances with the actual classification. The means square error is one of the most commonly used measures of success for numeric prediction. This value is computed by taking the average of the squared differences between each computed value and its corresponding correct value.

5.3.1 BACK PROPAGATION NEURAL NETWORK (BPNN)

Back Propagation Neural Network is a supervised multilayer neural network having three layers such as input layer, hidden layer and output layer (Barve, 2012). The following figure shows the back propagation architecture.

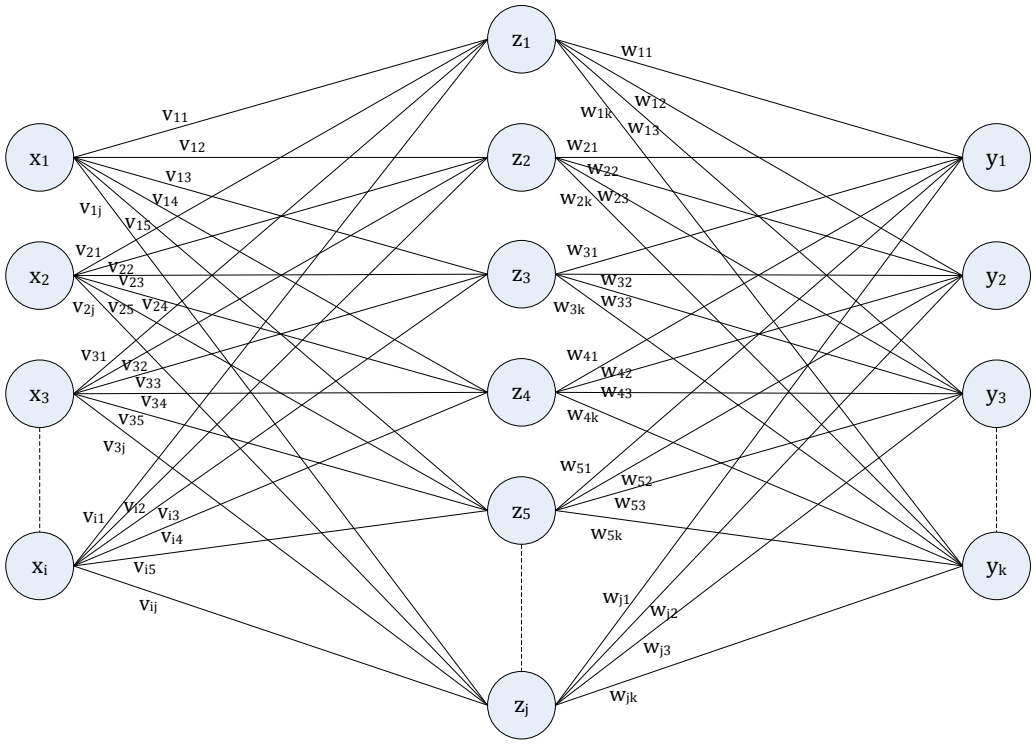


Figure 5.3: Back Propagation Neural Network Architecture

Back propagation training algorithm is a supervised learning algorithm for multilayer feed forward neural network. Since it is a supervised learning algorithm, both input and target output vectors are provided for training the network. The error data at the output layer is calculated using network output and target output. Then the error is back propagated to intermediate layers, allowing incoming weights to these layers to be updated. This algorithm is based on the error correction learning rule. Basically, the error back-propagation process consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward pass, input vector is applied to the network, and its effect propagates through the network, layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of network are all fixed. During the backward pass, on the other hand, the synaptic weights are all adjusted in accordance with the error correction rule. The actual response of the network is subtracted from a desired target response to produce an error signal. This error signal is then propagated backward through the

network, against direction of synaptic connections - hence the name "error back-propagation". The synaptic weights are adjusted so as to make the actual response of the network move closer the desired response.

5.3.1.1 ALGORITHM

The training algorithm of BPNN (Sivanandam and Deepa, 2006) involves the following stages:

1. Initialization of weights
2. Feed forward of training data
3. Calculation and back propagation of errors
4. Updation of weights and biases

Steps 2-4 is repeated until the tolerable error has been achieved or the required number of epochs has been completed. A single run of step 2-4 is called as one epoch. During the initialization of weights, small random real numbers are assigned. During feed forward stage, training sample data are fed to each of the input layer of neurons (x_i) and transmitted to the hidden neurons. Each hidden neuron (z_j) receives input from each of the input neurons and after calculating the activation function, it then transmits to each of the output neurons (y_k). The output unit then calculates the activation function to form the response of the network for a given input pattern.

During calculation of errors, the output given by the network (y_k) is compared with the target output (t_k) to determine the associated error for that patter with that unit. Based on the error, the factor δ_k is calculated and is used to distribute the error at output y_k during back propagation of errors to all units in the previous layer. Similarly, the factor δ_j is calculated for each of the hidden unit z_j.

During the final stage, weights are updated based on the δ factor and the activation. The following symbols and abbreviations will also be used in the subsequent explanation of the algorithm:

m = number of neurons in the input layer

p = number of neurons in the hidden layer

n = number of neurons in the output layer

x_i = i^{th} neuron in the input layer

z_j = j^{th} neuron in the hidden layer

y_k = k^{th} neuron in the output layer

z_{inj} = Accumulated input received by j^{th} hidden neuron

y_{ink} = Accumulated input received by k^{th} output neuron

X_i = Activation of i^{th} input neuron

Z_j = Activation of j^{th} hidden neuron

Y_k = Activation of k^{th} output neuron

T_k = target output associated with the output neuron y_k

V_{ij} = connection weight from i^{th} neuron in the input layer to the j^{th} neuron in the hidden layer where i=1, 2, …, m and j=1, 2, …, p

W_{jk} = connection weight from j^{th} neuron in the hidden layer to the k^{th} neuron in the output layer where j=1, 2, …, p and k=1, 2, …, n

δ_{yk} = Error at k^{th} output neuron

δ_{inj} = Accumulated error back-propagated at j^{th} hidden neuron from output layer

δ_{zj} = Error at j^{th} hidden neuron

Initialization of weights:

Connection weights are initialized using small random real numbers. In the

implementation of Mizo character recognition algorithm, the initialization assigns small random values in the range from -0.25 to +0.25.

Feed forward:

Each input neuron x_i (i=1,2,…n) receives the input signal and transmits to each of the hidden neuron.

For each hidden unit z_j (j=1, 2, …, p), the input signal is summed up

$$z_{in_j} = \sum_{i=1}^{m} X_i v_{ij}$$

Applying the activation function to the accumulated input to z_j,,

$$Z_j = f(z_{in_j})$$

Where the activation function used here is sigmoid function and

$$f(z_{in_j}) = \frac{1}{\left(1+e^{z_{in_j}}\right)}$$

The activation function output Z_j is then sends to each of the output neuron y_k. Each of the output neuron y_k receives signals from each of the hidden neuron through the connection weight w_{jk}.

$$y_{in_k} = \sum_{j=1}^{p} Z_j w_{jk}$$

Again applying the activation function to the accumulated input to each of the output neuron:

$$Y_k = f(y_{in_k})$$

So, the network output at k^{th} unit is the output of Y_k.

Calculation and back Propagation of Errors:

Each output unit y_k (k=1,2, …,n) receives a target pattern corresponding to an input pattern, error information term in each of the k^{th} output layer neuron is calculated as:

$$\delta_{yk} = (t_k - y_k) f'(y_{in_k})$$

Each hidden unit z_j (j=1,2,…p) sums its delta inputs from units in the layer above.

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_j w_{jk}$$

The error information term in each of the jk^{th} hidden layer neuron is calculated as

$$\delta_{zj} = \delta_{in_j} f'(z_{in_j})$$

Updation of weights:

Each output unit y_k (k=1, 2, …, n) updates its connection weights from each hidden unit z_j (j=0, 1, …, p)

The weight correction term is given by

$\Delta w_{jk} = \eta \delta_k Z_j$ Therefore,

$$w_{jk}(\text{new}) = w_{jk}(\text{old}) + \Delta w_{jk}$$

Each hidden unit z_j (j=1, 2, …p) updates its connection weights from each input unit x_i (i=1, 2, …, m)

The weight correction term is given as

$\Delta v_{ij} = \eta \delta_j X_i$ Therefore,

$$v_{ij}(\text{new}) = v_{ij}(\text{old}) + \Delta v_{ij}$$

The process will be repeated until the stopping condition is reached. The stopping condition may be minimization of errors, number of epochs etc. The errors should be converging in each of the iterations or else the network may not be able to be trained.

5.3.1.2 EXPERIMENTAL RESULTS & DISCUSSIONS

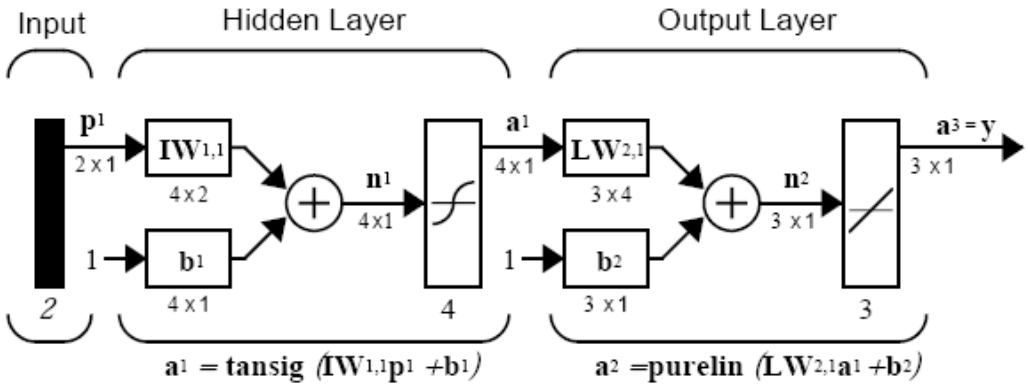


Figure 5.4: Back Propagation Neural Network (BPNN)

The BPNN is created with the MATLAB 7.12 in-built function 'newff' having the transfer function 'tansig' in hidden layer and 'purelin' in the output layer. The proposed back propagation neural network has 54 neurons in the input layer, 150 neurons in the hidden layer, and 93 neurons in the output layer. The training function 'traingdx', variable learning rate gradient descent, is used in the experiment for training the network. The dataset (dataset#1) is divided into three subsets such as training set, validation set and test set. During the training process, the training stopped when the best validation performance occurred at the iteration 57. The training errors, validation errors and test errors are plot in the following figure.

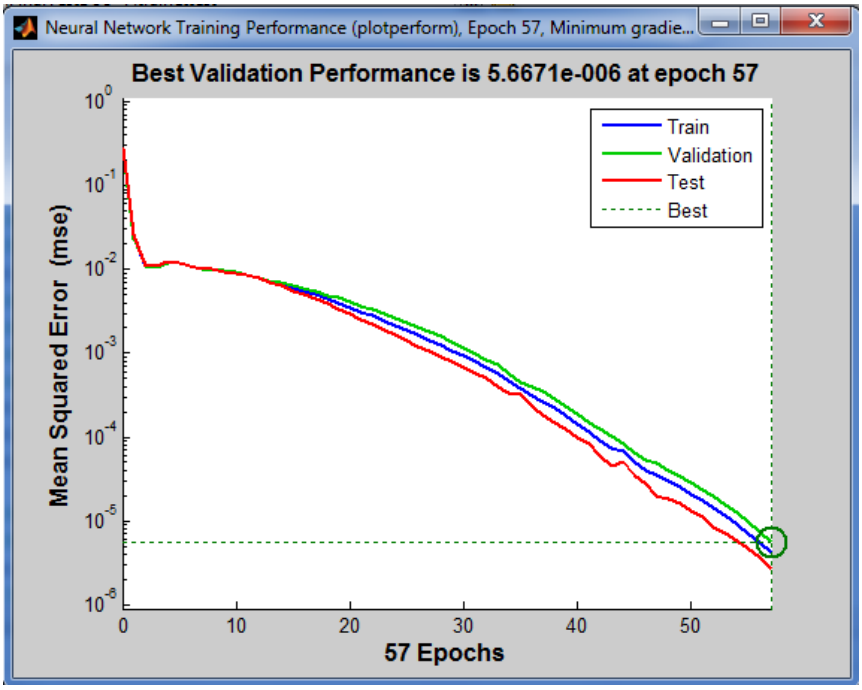


Figure 5.5: BPNN Training Performance

The Regression plot is one of the parameters for checking the output of the network has closely relationship with the target value for which we have generated regression plot during training process which is shown in the figure below.

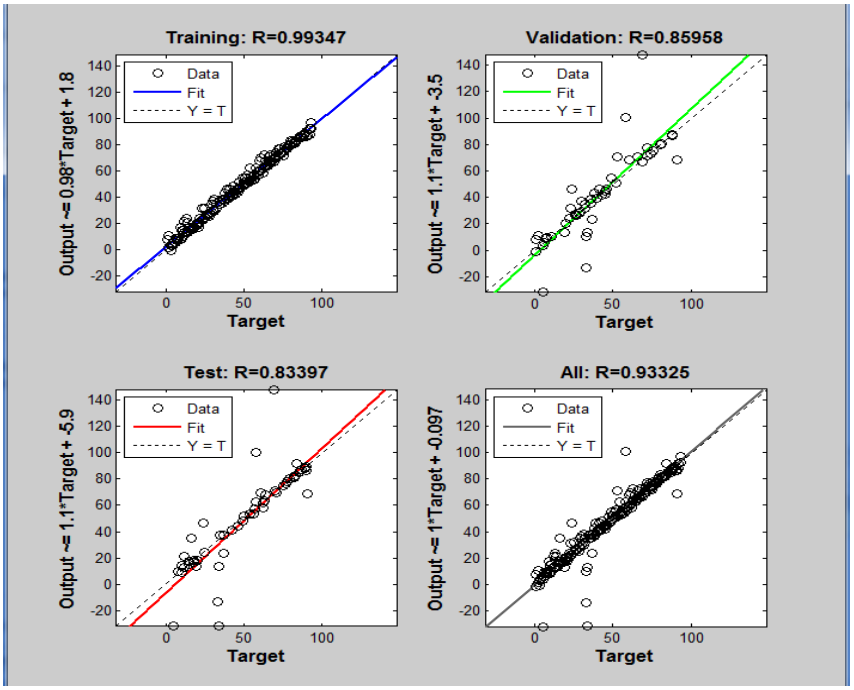


Figure 5.6: BPNN Regression Plot

The above regression plot represents the network outputs that have close relationship with the targets. The data is a perfectly fit as the data falls along a 45 degree line, where the network outputs are equal to the targets. The above regression plots display the network output tracks the targets very well for training, testing and validation, and the R value is over 0.93 for the total response. The results are quite satisfactory.

The Plot confusion matrices is very important for identification of correct classification and misclassification of the character for which we have generated plot confusion matrix for Â, Ê, Î, Ô, Û and Ṭ which is shown in the following figure.

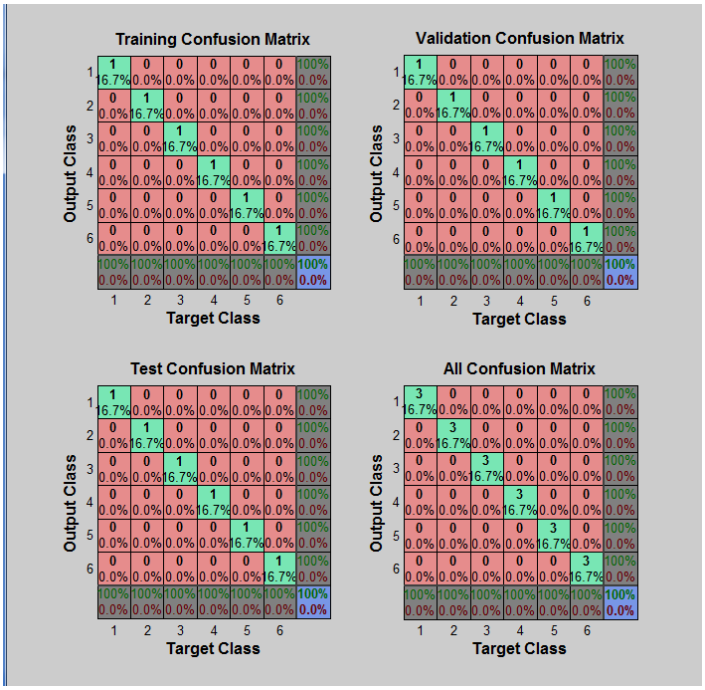


Figure 5.7: BPNN -Plot Confusion Matrix

The above plot confusion matrix is simply a square matrix that shows various classification and misclassifications of characters. The network outputs are very accurate as indicated by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares

illustrated the overall accuracies which is 100 % accuracy in these particular characters.

After the above training process is over, we have tested the proposed BPN network with test data (dataset#2). The test results are shown in the following tables.

Table 5.3: Test Results of word recognition using BP Neural Network

Document Image	No of Word present	Correctly Recognize Word	Mis-recognize Word	Precision (%)	MSE	Time Taken (Second)
Doc #1	181	179	2	98.89503	0.99	96.22
Doc #2	127	125	2	98.42520	0.98	72.34
Doc #3	159	158	1	99.37107	0.99	80.56
Doc #4	67	64	3	95.52239	0.99	43.56
Total	**534**	**526**	**8**	**98.05342**	**0.9875**	**73.17**

Table 5.4: Test Results of character recognition using BP Neural Network

Document Image	No of Character present	Correctly Recognize Character	Mis-recognise Character	Precision (%)	MSE	Time Taken (Second)
Doc #1	793	791	2	99.75	0.99	96.22
Doc #2	635	633	2	99.68	0.98	72.34
Doc #3	643	642	1	99.84	0.99	80.56
Doc #4	249	246	3	98.79	0.99	43.56
Total	**2320**	**2312**	**8**	**99.52**	**0.9875**	**73.17**

From the above test results, we can see that out of 534 words, the correctly recognised word is 526 and misclassification of words is 8. There are 2320 characters presents in the dataset considering only 12 points font size, out of which 2312 characters are correctly classified and only 8 characters are misclassified. The overall Classification accuracy is about 99.52% with an average mean square error of 0.98. The speed of recognition system is about 73.17 seconds. In view of these, the Back Propagation Neural Network based approach Classification is quite satisfactory for implementation of Mizo characters recognition system. The misclassified characters are shown in the table below

Table 5.5: Misclassified characters by BPNN

Document Image	Misrecognize Character	Recognized as	No of Occurrence
Doc #1	û	u	2
Doc #2	"	,,	2
Doc #3	û	u	3
Doc #4	1	I	1
Total			8

5.3.2 RADIAL BASIS FUNCTION (RBF)

The architecture of radial basis function network consists of three layers, the input layers, hidden layers and output layers as shown in the figure below:

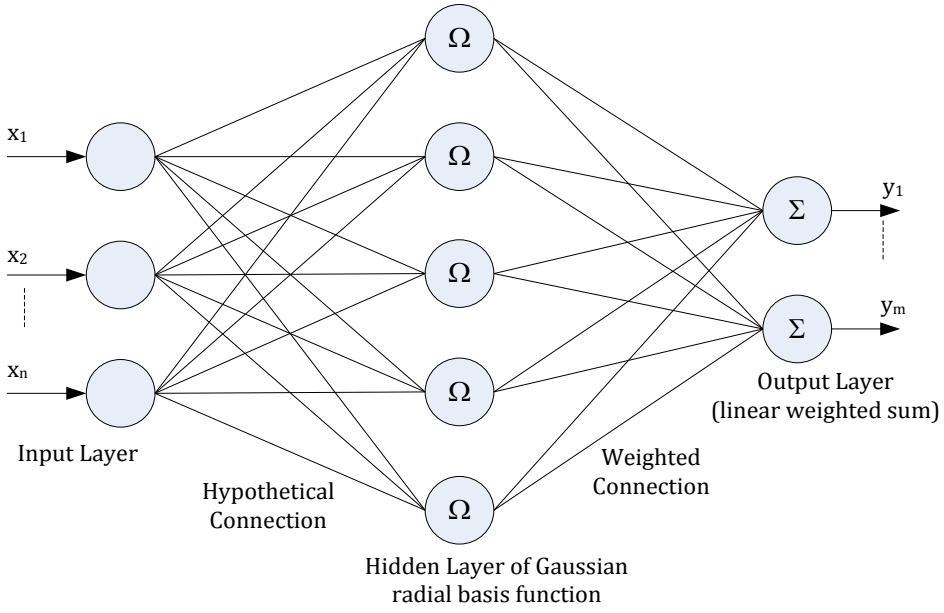


Figure 5.8: Radial Basis Function (RBF) Architecture

The RBF is designed to handle more neurons than standard feed forward back propagation network and it takes a fraction of time to train standard feed forward network (Beale *et al.*, 2010). The RBF work best when many training vectors are available. Radial Basis Function network can be used for approximating functions and recognizing patterns. It uses Gaussian Potential functions. The Gaussian potential

functions are also used in networks called regularization networks. The architecture of radial basis function network is a multilayer feed forward network. There exists 'n' number of input neurons and 'm' number of output neurons with the hidden layer existing between the input and output layer. This hidden layer may also be called as radial basis layer. The interconnection between the input lay and hidden layer forms hypothetical connection and between the hidden and output layer forms weighted connections. The training algorithm is used for updation of weights in all interconnections.

5.3.2.1 ALGORITHM

The training algorithm for radial basis function network is given below. The important aspect of the radial basis function network is the usage of activation function for computing the output. Radial basis function uses Gaussian activation function. The response of such function is non-negative for all value of 'x'. The function is defined as

$$f(x) = \exp(-x^2)$$

Its derivative is given by

$$f'(x) = -2x.\exp(-x2) = 2x.f(x)$$

The radial basis function is different from the back propagation network in the Gaussian function it uses. The training algorithm for the network is given as follows:

Step 1: Initialize the weights (set to small random values)

Step 2: While stopping is false do step 3–10.

Step 3: For each input do step 4–9.

Step 4: Each input unit (x_i where i = 1, 2, 3,…., n) receives input signals to all units in the layer above (hidden layer).

Step 5: Calculate the radial basis function

Step 6: Choose the centres for the radial basis functions. The centres are chosen from the set of input vectors. A sufficient number of centres have to be selected in order to ensure adequate sampling of the input vector space.

Step 7: The output of im unit $v_i(x_i)$ in the hidden layer.

$$v_i(x_i) = e\left(\sum_{j=1}^{r}\left[x_{ji} - \hat{x}_{ji}\right]^2/\sigma_1^2\right)$$

Where

x_{ji} = centre of the RBF unit for input variables

σ_i = Width of the RBF unit

x_{ji} = j_{th} variable of input pattern

Step 8: Initialize the weights in the output layer of the network to some small random value.

Step 9: Calculate the output of the neural network

$$y_{net} = \sum_{i=1}^{H} w_{im} v_i(x_i) + w_0$$

Where

H = number of hidden layer nodes (RBF function)

y_{net} = Output value of m_{th} node in the output layer for the nth incoming pattern

w_{im} = Weight between i_{th} RBF unit and m_{th} output node

w_o= Biasing term at n_{th} output node

Step 10: Calculate error and test stopping condition

The stopping condition may be weight change, number of epoch, etc.

5.3.2.2 EXPERIMENTAL RESULTS AND DISCUSSIONS

MATLAB 7.12 inbuilt transfer function ‘radbas’ is used in the hidden layer and ‘purelin’ in the output layer. The network has 54 neurons in the input layer, 93

neurons in the hidden layer, and 93 neurons in the output layer. In this work, the radial basis network with function 'newrb' is used for designing the network with the design parameters GOAL and SPREAD. The network is designed with the default mean square error goal set at 0.01 and the spread value at 5. This makes the network function smoother and results in better generalization for new input vectors.

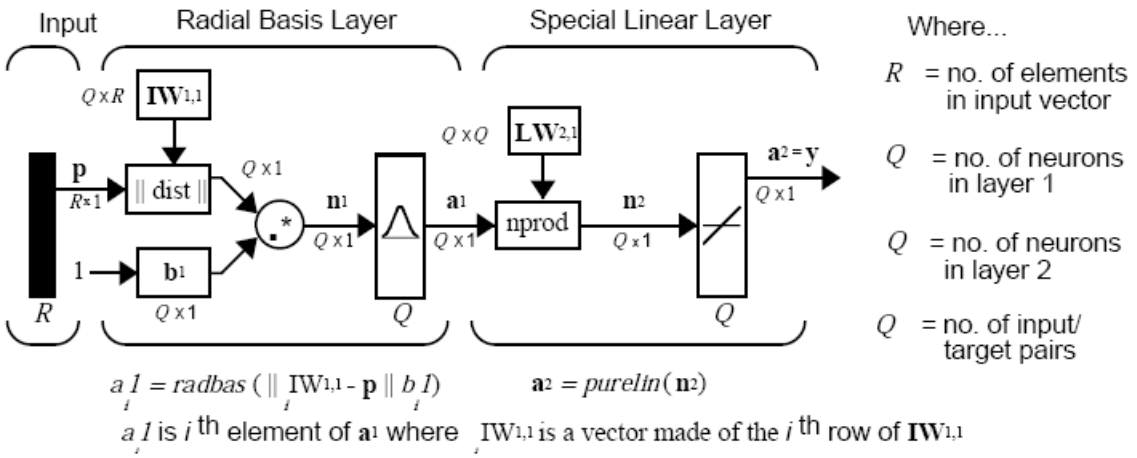


Figure 5.9: Radial Basis Function

The network uses the training function '*trainrp'* as the memory requirement is relatively small and faster than standard gradient descent algorithms. The designed RBF network is trained using training dataset (dataset#1). The training dataset is divided into three subsets such as training set, validation set and test set. During the training process, the training stopped when the best validation performance occurred at the iteration 54. The training performance is plotted between the Mean square error and the iteration (epoch) which is shown below.

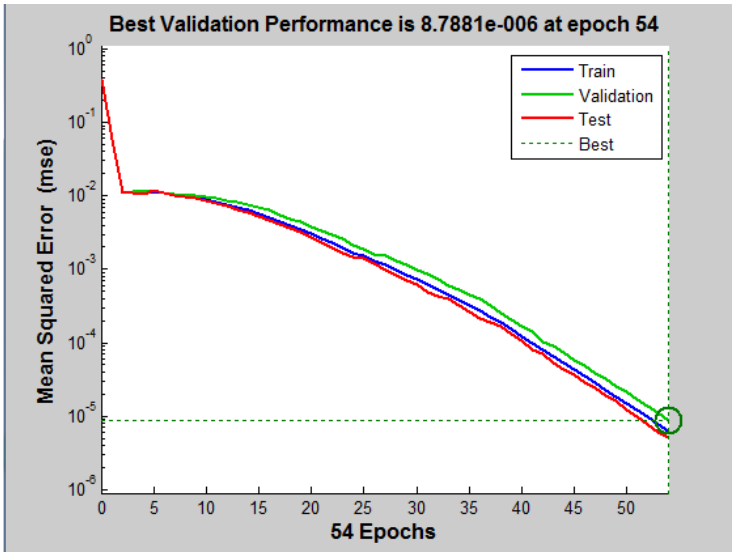


Figure 5.10: RBF-Training Performance

The regression plot is generated as shown in the figure below to find out the the network perform linear regression between the network outputs and the corresponding targets.

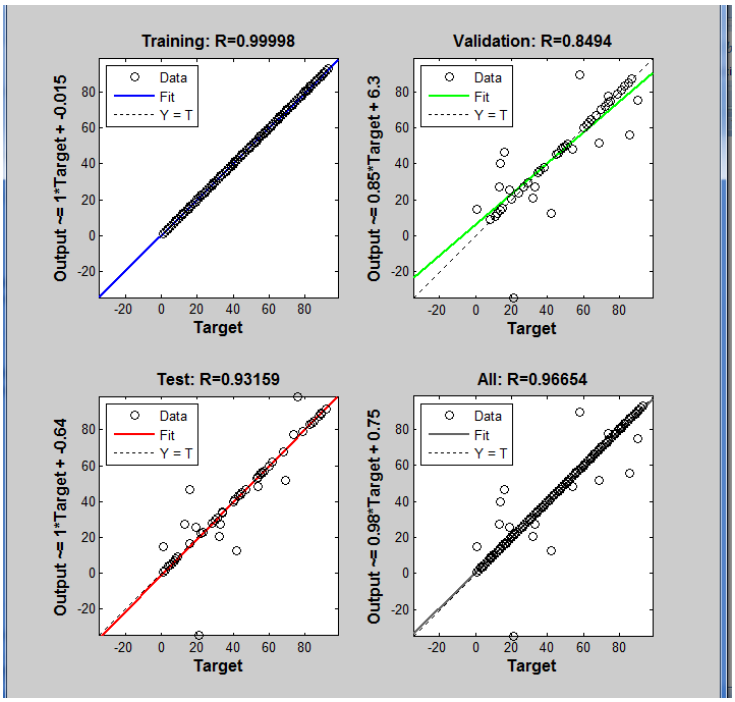


Figure 5.11: RBF-Regression Plot

The above regression plot represents the network outputs have close relationship with the targets. The training data indicates a good fit as the validation and

test results show R values that greater than 0.9. The scatter plot is helpful in showing that certain data points have poor fits.

In order to examine the performance of classifiers, we have plotted confusion matrices as shown in the figure below.

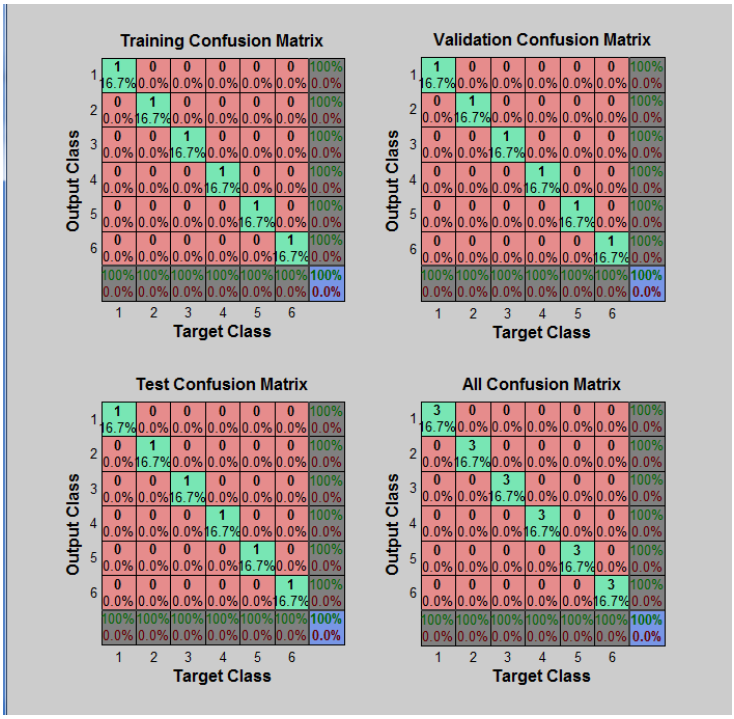


Figure 5.12: RBF-Plot Confusion Matrix

The above confusion matrix is simply a square matrix that shows various classification and misclassifications of characters. The network outputs are very accurate as indicated by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrated the overall accuracies which is 100 % accuracy in this work.

In order to test the Classification using RBF Neural Networks, we used the dataset for testing (dataset #2). These dataset#2 are fed into the proposed RBF Neural Network and the results are shown in the following tables.

Table 5.6: Test Results of word recognition using RBF Neural Network

Document Image	No of Word present	Correctly Recognize Word	Mis-recognize Word	Accuracy (%)	MSE	Time Taken (Second)
Doc #1	181	179	2	98.89	0.98	99.88
Doc #2	127	124	3	97.64	0.99	72.34
Doc #3	159	157	2	98.74	0.98	83.56
Doc #4	67	64	3	95.52	0.98	43.56
Total	**534**	**524**	**10**	**97.69**	**0.9825**	**74.83**

Table 5.7: Test Results of character recognition using RBF Neural Network

Document Image	No of Character present	Correctly Recognize Character	Mis-recognize Character	Accuracy (%)	MSE	Time Taken (Second)
Doc #1	793	789	4	99.49	0.98	99.88
Doc #2	635	633	2	98.90	0.99	72.34
Doc #3	643	639	4	99.69	0.98	83.56
Doc #4	249	248	1	98.39	0.98	43.56
Total	**2320**	**2309**	**11**	**99.12**	**0.9825**	**74.83**

From the above test results, we can see that out of 534 words, the correctly recognized word is 524 and misclassification of words is 10. There are 2320 characters presents in the dataset considering only 12 points font size, out of which 2309 characters are correctly classified and 11 characters are misclassified. The overall Classification accuracy is about 99.12% with an average mean square error of 0.9825. The speed of recognition system is about 74.83 seconds. In view of these, the RBF Neural Network based approach Classification is also quite satisfactory for implementation of mizo characters recognition system. The characters misclassified by RBF are shown in the following table.

Table 5.8: Misclassified Characters by RBF

Document Image	Misrecognize Character	Recognized as	No of Occurrence
Doc #1	û	u	2
	E	F	2
Doc #2	0	o	2
Doc #3	ê	e	4
Doc #4	3	8	1
Total			**11**

5.3.3 LEARNING VECTOR QUANTIZATION (LVQ)

The LVQ neural network was first proposed by Kohonen. A LVQ network is a two layer feed-forward network, consisting of a competitive layer and linear layer. The first layer learns to classify the input vectors. The second layer transforms the competitive layer's classes into desired classification defined by the designer of the LVQ Network. The classes of competitive layer are called sub-classes while the classes of linear layer are called target classes. Both of the competitive and linear layers have one neuron per class. The number of neurons for the hidden layer (competitive) is always larger than the number of output neurons.

The LVQ method is used in training ANNs for pattern classification, where each output represents a particular class. Each class is referred by a vector of weights that sequentially, represents the centers of the classes. The training data set is used several times during the training phase in a random order. The training of LVQ ANNs is terminated when classes remain stable or a specific number of iterations have been carried out. A trained LVQ neural network is a vector comparator. When a new vector is presented to the input layer of a LVQ ANN it will be classified to a class with the closest center (Pedreira, 2006).

The basic architecture of LVQ network that has a first competitive layer and

a second linear layer is shown in the figure below.

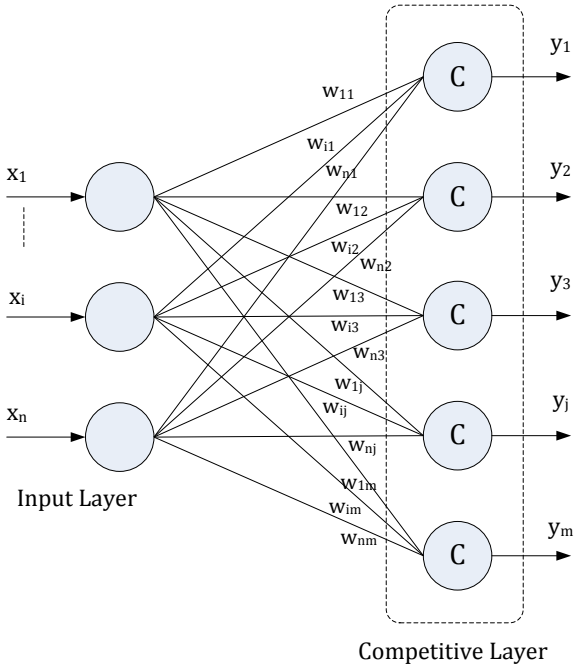


Figure 5.13: Linear Vector Quantization (LVQ) Architecture

In LVQ Network, each output unit has a known class, since it uses supervised learning, thus differing from kohonen SOM, which uses unsupervised learning. The algorithm for the LVQ net (Sivanandam and Deepa, 2006) is to find the output unit that has a matching pattern with the input vector which is given below.

5.3.3.1 ALGORITHM

Step 1: Initialize weights (reference) vectors and initialize learning rate

Step 2: While stopping is false, do step 3-7

Step 3: For each training input vector x, do step 4-5

Step 4: Computer *j* using square Euclidean distance

$$D(j) = \sum(w_{ij} - x_i)^2$$

Find j and D(j) is minimum

Step 5: Update w_j as follows:

If $t=c_j$, then

$$w_{j(new)} = w_{j(old)} + \alpha[x - w_{j(old)}]$$

if $t \neq c_j$, then

$$w_{j(new)} = w_{j(old)} - \alpha[x - w_{j(old)}]$$

Step 6: Reduce the learning rate

Step 7: Test for stopping condition

The condition may be fixed number of iterations or the learning rate reaching a sufficiently small value.

5.3.3.2 EXPERIMENTAL RESULTS AND DISCUSSIONS

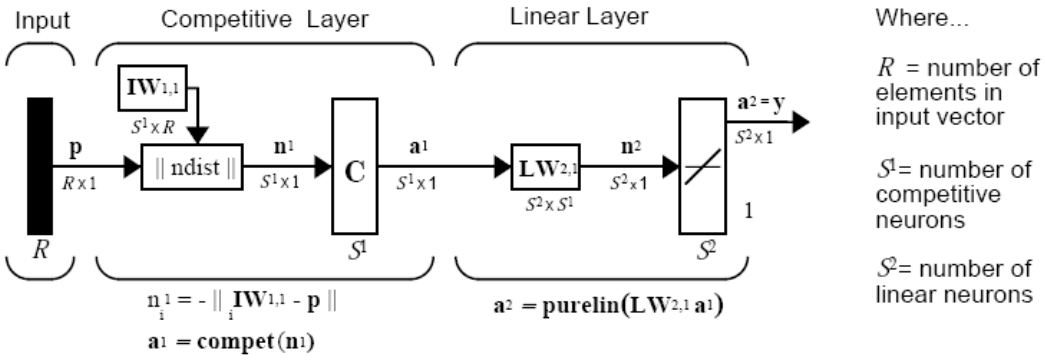


Figure 5.14: Learning Vector Quantization

In this work, we have used MATLAB 7.12 built-in function '*newlvq*' to create LVQ network having two layer networks. The first layer uses the competitive transfer function 'compet', calculates weighted inputs with '*negdist*', and net input with '*netsum*'. The second layer uses linear transfer function '*purelin*', calculates weighted input with '*dotprod*' and net inputs with '*netsum*'. Neither layer has biases. The LVQ network parameters are set with the default learning rate (0.01) and default learning function (*learnlv1*). The first layer weights are initialized to the centre of the input ranges with the function '*midpoint*'. The second layer weights are set from the typical class

percentages.

The designed LVQ network has 54 input neurons, 60 hidden (competitive) neurons and 93 output neurons. The network is trained using training dataset (dataset#1). The dataset (dataset#1) is divided into three subsets such as training set, validation set and test set. During the training process, the training stopped when the best validation performance occurred at the iteration 58. The network uses the training function '*train*' for training the network. The training errors, validation errors and test errors are plot in the following figure.

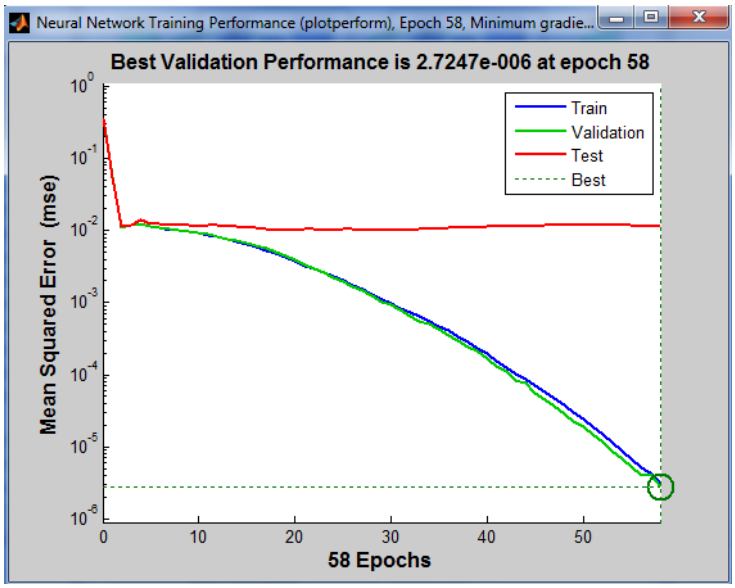


Figure 5.15: LVQ- Training Performance

In order to find out the network has performed linear regression between the network outputs and the corresponding targets, the regression plot is generated. The regression plot represents the network outputs have close relationship with the targets. The following figure shows the regression plot for the network.

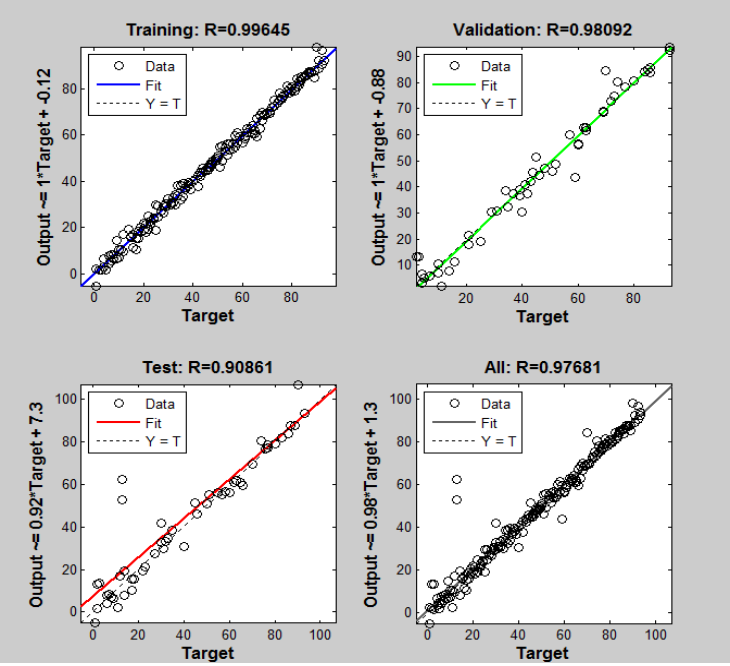


Figure 5.16: LVQ-Regression Plot

The above regression plots display the network output tracks the targets very well for training, testing and validation, and the R value is over 0.97 for the total response. The results are quite satisfactory.

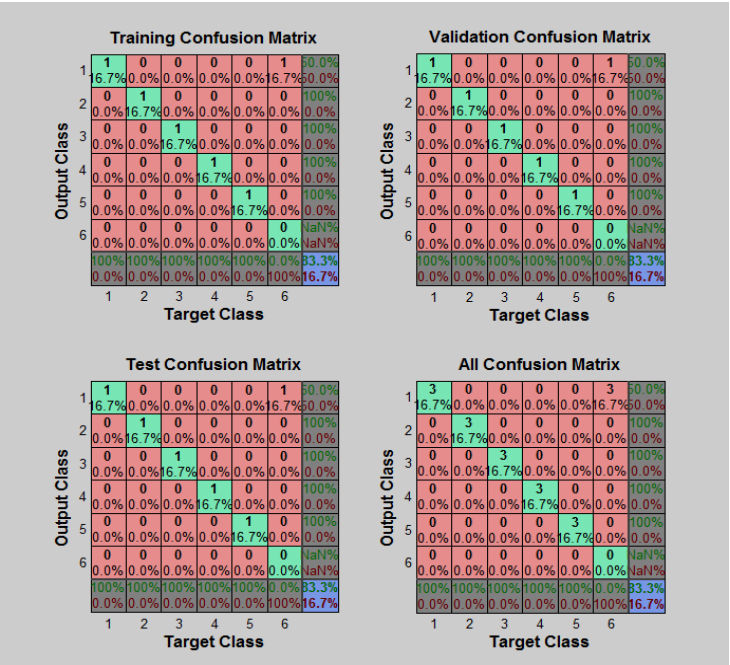


Figure 5.17: LVQ-Plot Confusion Matrix

The above confusion matrix is simply a square matrix that shows various

classification and misclassifications of characters. The network outputs are not accurate enough as indicated by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares illustrated the overall accuracies which is 83.3 % accuracy in this work.

In order to test the Classification using LVQ Neural Networks, we used the dataset for testing (dataset #2). These dataset#2 are fed into the proposed LVQ Neural Network and the results are shown in the following tables.

Table 5.9: Test Results of word recognition using LVQ Neural Network

Document Image	No of Word present	Correctly Recognize Word	Mis-recognize Word	Accuracy (%)	MSE	Time Taken (Second)
Doc #1	181	169	12	93.37	0.97	110.66
Doc #2	127	120	7	94.48	0.98	72.34
Doc #3	159	129	30	81.13	0.97	83.56
Doc #4	67	58	9	86.56	0.97	43.56
	534	**476**	**58**	**88.885**	**0.9725**	**77.53**

Table 5.10: Test Results of character recognition using LVQ Neural Network

Document Image	No of Character present	Correctly Recognize Character	Mis-recognize Character	Accuracy (%)	MSE	Time Taken (Second)
Doc #1	793	777	16	97.98	0.97	110.66
Doc #2	635	626	9	98.58	0.98	72.34
Doc #3	643	617	26	95.95	0.97	83.56
Doc #4	249	240	9	96.38	0.97	43.56
	2320	**2260**	**60**	**97.22**	**0.97**	**77.53**

From the above test results, we can see that out of 534 words, the correctly recognised word is 476 and misclassification of words is 58. There are 2320 characters presents in the dataset considering only 12 font size, out of which 2260 characters are

correctly classified and 60 characters are misclassified. The overall Classification accuracy is about 97.22% with an average mean square error of 0.97. The speed of recognition system is about 77.53 seconds. In view of these, the LVQ Neural Network based approach Classification is also not good enough for implementation of mizo characters recognition system. The following are misclassified by LVQ Neural Network.

Table 5.11: Misclassified Characters by LVQ

Document Image	Misrecognize Character	Recognized as	No of Occurrence
Doc #1	I	1	14
	E	F	2
Doc #2	1	I	5
	"	,,	2
	0	o	2
Doc #3	.	,	7
	û	u	3
	l	1	16
Doc #4	,	.	7
	1	I	1
	3	8	1
Total			**60**

5.3.4 RECURRENT NEURAL NETWORK (RNN)

The Recurrent Neural Network (RNN) is also known as Elman Neural Networks and it is feed forward network with an input layer, a hidden layer, an output layer and a special layer called context layer. The output of each hidden neuron is copied into a specific neuron in the context layer (Beale *et al.*, 2010). The value of the context neuron is used as an extra input signal for all the neurons in the hidden layer one time step later. In an Elman network, the weights from the hidden layer to the context layer are set to one and are fixed because the values of the context neurons have to be copied exactly. Furthermore, the initial output weights of the context neurons are equal to half

the output range of the other neurons in the network. The Elman network can be trained with gradient descent back propagation and optimization methods. The following figure illustrates the RNN Architecture.

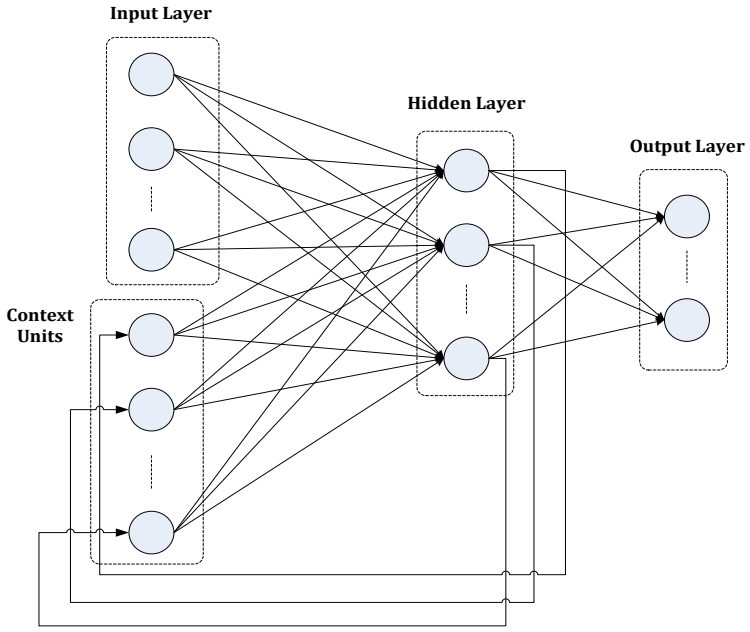


Figure 5.18: Recurrent Neural Network Architecture

5.3.4.1 ALGORITHM

During the training procedure of an Elman network, similar to the case of MLP training, the network's output is compared with the target output and the square error is used to update the network's weights according to the error backpropagation algorithm with the exception that the values of recurrent connections' weights are constant to 1.0. If x^n is the vector produced by the union of input and context vectors, then the training algorithm for an Elman network is very similar to the algorithm for an MLP network training:

Step 1: Initialize the weight vector $w(0)$ with random values in $(1,1)$ the learning rate η, the repetitions counter *(k = 0)* and the epochs counter *(k = 0)*. Initialize the context nodes at 0.5.

Step 2: Let *w(k)* the network's weight vector in the beginning of epoch *k*

1. Start of epoch *k*. Store the current values of the weight vector $w_{old} = w(k)$
2. For $n = 1, 2, \ldots\ldots, N$
    1. Select the training example (x^n, t^n) and apply the error backpropagation in order to compute the partial derivatives $\frac{\partial E^n}{\partial W_i}$
    2. Update the weights

        $w_i(k+1) = w_i(k) - \eta \frac{\partial E^n}{\partial W_i}$

    3. Copy the hidden nodes' values to the context units.
    4. $k = k + 1$
3. End of epoch *k* Termination check. If true, terminate.

Step 3: $k = k + 1$. Go to step 2.

5.3.4.2 EXPERIMENTAL RESULTS AND DISCUSSIONS

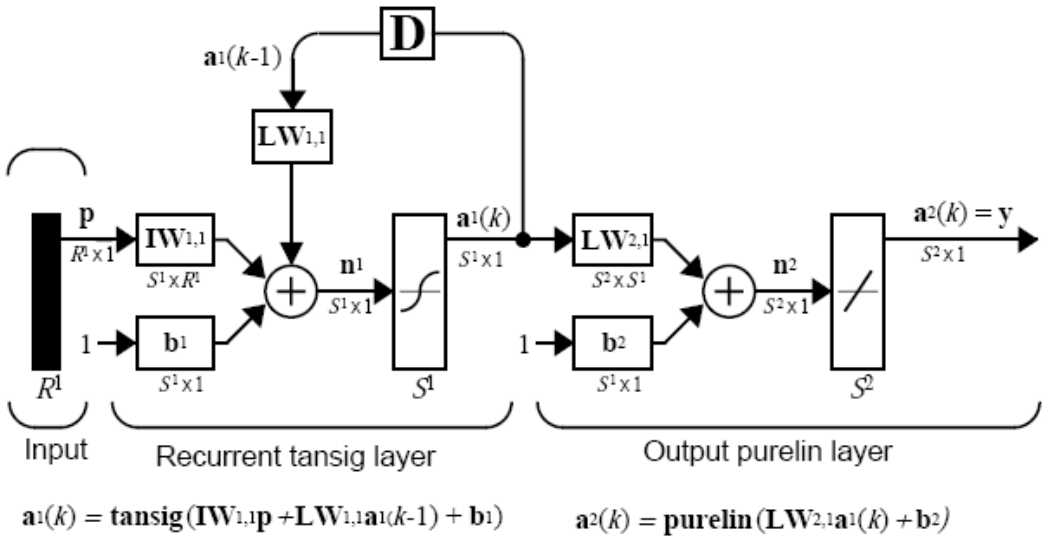


Figure 5.19: Elman Networks Architecture

In this work, we have developed RNN based simulation software using MATLAB 7.12 for classification of Mizo characters. The Elman network is created with

the MATLAB in-built function 'newelm' having the transfer function 'tansig' in hidden (recurrent) layer and 'purelin' in the output layer. When the network is created, the weights and biases of each layer are initialized with the Nguyen-Widrow layer-initialization method, which is implemented in the function '*initnw'*.

The network has 54 neurons in the input layer, 80 neurons in the hidden layer, and 93 neurons in the output layer with learning rate of 0.01. The training dataset (dataset#1) are used for training the network. The dataset is divided into three subsets such as training set, validation set and test set. During the training process, the training stopped when the best validation performance occurred at the iteration 54. The training errors, validation errors and test errors are plot in the following figure.

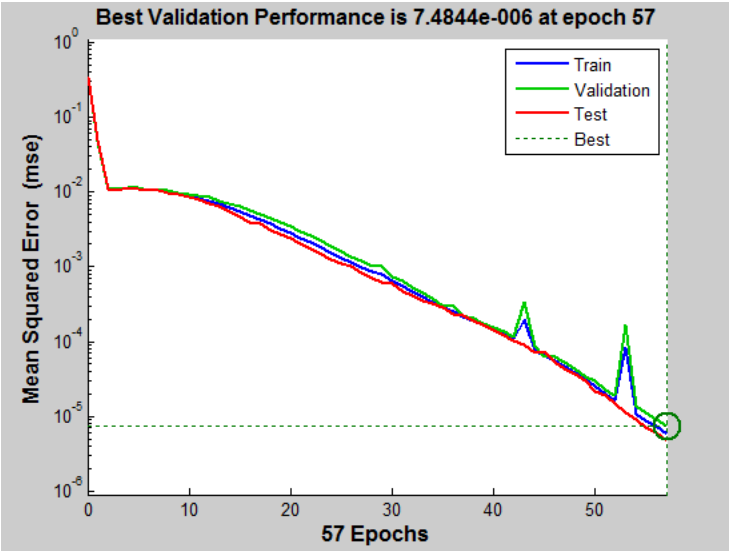


Figure 5.20: RNN-Training Performance

Regression plot is also generated to find out the network has performed linear regression between the network outputs and the corresponding targets as shown in the figure below. The training data indicates a good fit as the validation and test results show R values that greater than 0.9.

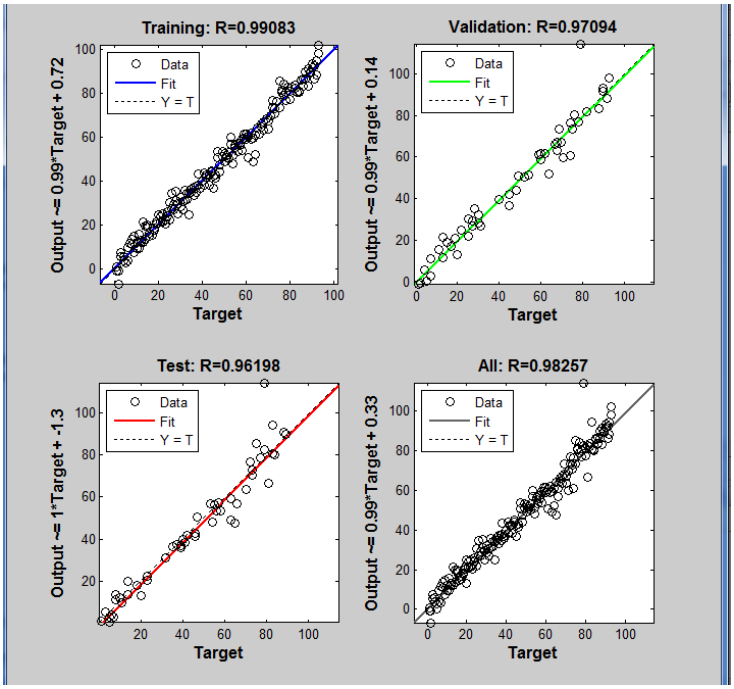


Figure 5.21: RNN-Regression Plot

The confusion matrix has been plotted as shown in the figure below.

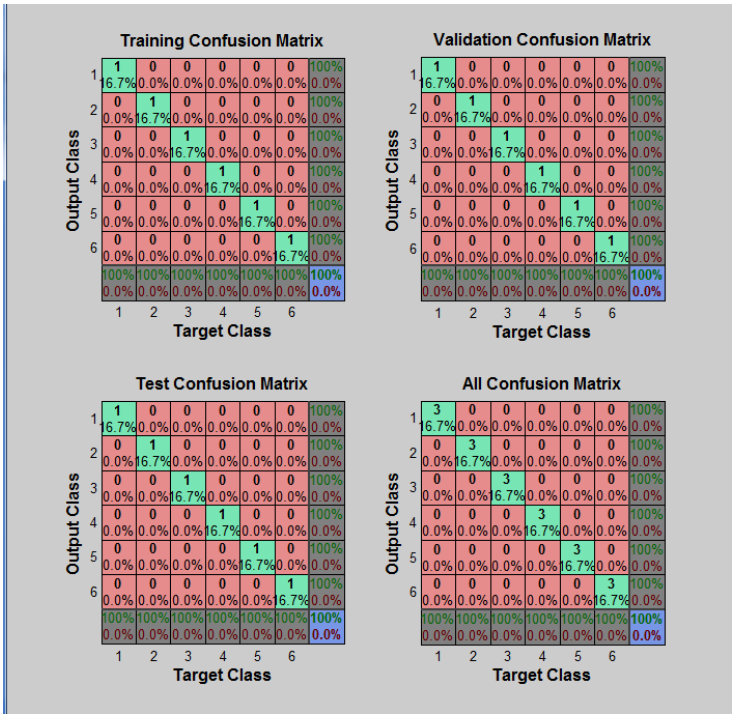


Figure 5.22: RNN- Plot Confusion Matrix

The above confusion matrix shows the network outputs are very accurate as indicated by the high numbers of correct responses in the green squares and the low numbers of incorrect responses in the red squares. The lower right blue squares

illustrated the overall accuracies which is 100 % accuracy in this work.

In order to test the Classification using Recurrent Neural Networks, we used the dataset for testing (dataset #2). These dataset#2 are fed into the proposed Recurrent Neural Network and the results are shown in the following table.

Table 5.12: Test Results of word recognition using RNN Neural Network

Document Image	No of Word present	Correctly Recognize Word	Mis-recognize Word	Accuracy (%)	MSE	Time Taken (Second)
Doc #1	181	179	2	98.89	0.99	110.66
Doc #2	127	121	6	95.27	0.98	72.34
Doc #3	159	157	2	98.74	0.99	83.56
Doc #4	67	63	4	94.02	0.98	43.56
Total	**534**	**520**	**14**	**96.73**	**0.985**	**77.53**

Table 5.13: Test Results of character recognition using RNN Neural Network

Document Image	No of Character present	Correctly Recognize Character	Mis-recognize Character	Accuracy (%)	MSE	Time Taken (Second)
Doc #1	793	791	2	99.75	0.99	110.66
Doc #2	635	624	11	98.27	0.98	72.34
Doc #3	643	640	3	99.53	0.99	83.56
Doc #4	249	242	7	97.19	0.98	43.56
Total	**2320**	**2297**	**23**	**98.68**	**0.985**	**77.53**

From the above test results, we can see that out of 534 words, the correctly recognised word is 520 and misclassification of words is 14. There are 2320 characters presents in the dataset considering only 12 font size, out of which 2297 characters are correctly classified and 23 characters are misclassified. The overall Classification accuracy is about 98.68 % with an average mean square error of 0.985. The speed of recognition system is about 77.53 seconds. In view of these, the Recurrent Neural

Network based approach Classification is not good enough for implementation of mizo characters recognition system. The following are characters misclassified by Recurrent Neural Network.

Table 5.14: Misclassified Characters by RNN

Document Image	Misrecognize Character	Recognized as	No of Occurrence
Doc #1	E	F	2
Doc #2	u	û	12
Doc #3	.	,	7
Doc #4	ṭ	t	2
Total			23

5.4 POST PROCESSING

The Neural Network Classifier output may be encoded using ASCII or Unicode. The ASCII code encoding scheme cannot be applied for mizo characters due to special characters presents in mizo alphabets. Therefore, it is suggested to use Unicode encoding system for machine readable and editable of the output of the classifier.

In this section, we applied a two level post processing; where the first level post processing is Unicode encoding scheme of the recognised characters and the second level is formatting the encoded character into meaningful words.

The Unicode standard reflects the basic principle which emphasizes that each character code has a width of 16 bits. Unicode text is simple to parse and process and Unicode characters have well defined semantics. Hence Unicode is chosen as the encoding scheme for the current work. After classification the characters are recognized and a mapping table is created in which the Unicode for the corresponding characters are mapped. The Unicode corresponding to Mizo characters is shown in the following table.

Table 5.15: Mapping of Unicode with Mizo Characters

U+0041	A	Mizo Capital letter A	U+0061	a	Mizo Small Letter A
U+0042	B	Mizo Capital letter B	U+0062	b	Mizo Small Letter B
U+0043	C	Mizo Capital letter C	U+0063	c	Mizo Small Letter C
U+0044	D	Mizo Capital letter D	U+0064	d	Mizo Small Letter D
U+0045	E	Mizo Capital letter E	U+0065	e	Mizo Small Letter E
U+0046	F	Mizo Capital letter F	U+0066	f	Mizo Small Letter F
U+0047	G	Mizo Capital letter G	U+0067	g	Mizo Small Letter G
U+0048	H	Mizo Capital letter H	U+0068	h	Mizo Small Letter H
U+0049	I	Mizo Capital letter I	U+0069	i	Mizo Small Letter I
U+004A	J	Mizo Capital letter J	U+006A	j	Mizo Small Letter J
U+004B	K	Mizo Capital letter K	U+006B	k	Mizo Small Letter K
U+004C	L	Mizo Capital letter L	U+006C	l	Mizo Small Letter L
U+004D	M	Mizo Capital letter M	U+006D	m	Mizo Small Letter M
U+004E	N	Mizo Capital letter N	U+006E	n	Mizo Small Letter N
U+004F	O	Mizo Capital letter O	U+006F	o	Mizo Small Letter O
U+0050	P	Mizo Capital letter P	U+0070	p	Mizo Small Letter P
U+0051	Q	Mizo Capital letter Q	U+0071	q	Mizo Small Letter Q
U+0052	R	Mizo Capital letter R	U+0072	r	Mizo Small Letter R
U+0053	S	Mizo Capital letter S	U+0073	s	Mizo Small Letter S
U+0054	T	Mizo Capital letter T	U+0074	t	Mizo Small Letter T
U+0055	U	Mizo Capital letter U	U+0075	u	Mizo Small Letter U
U+0056	V	Mizo Capital letter V	U+0076	v	Mizo Small Letter V
U+0057	W	Mizo Capital letter W	U+0077	w	Mizo Small Letter W
U+0058	X	Mizo Capital letter X	U+0078	x	Mizo Small Letter X
U+0059	Y	Mizo Capital letter Y	U+0079	y	Mizo Small Letter Y
U+005A	Z	Mizo Capital letter Z	U+007A	z	Mizo Small Letter Z
U+00C2	Â	Mizo Capital letter A with circumflex	U+00E2	â	Mizo Small letter a with circumflex
U+00CA	Ê	Mizo Capital letter E with circumflex	U+00EA	ê	Mizo Small letter e with circumflex
U+00CE	Î	Mizo Capital letter I with circumflex	U+00EE	î	Mizo Small letter i with circumflex
U+00D4	Ô	Mizo Capital letter O with circumflex	U+00F4	ô	Mizo Small letter o with circumflex
U+00DB	Û	Mizo Capital Letter U with circumflex	U+00FB	û	Mizo Small Letter u with circumflex

U+1E6C	Ṭ	Mizo Capital Letter T with dot	U+1E6D	ṭ	Mizo Small Letter ṭ with dot

The scanned image is passed through various blocks of functions and finally compared with the recognition details from the mapping table from which corresponding Unicode are accessed and printed using standard Unicode fonts so that the OCR achieved.

In the second level of post-processing, format the encoded characters into meaningful words using the principle of bounding box and line formatting using a line break which is incremented on every line segment. The bounding box is used to calculate the distance between the characters and if the distances are greater than the threshold value, then the characters form a separate word. The character and word spacing should be same format as that of the input testing dataset. The post processing output can be edited by using any word processing software. The results are quite satisfactory for mizo character recognition system.

5.5 CONCLUSIONS

In this work, we have carried out an investigation on various types of classification methods currently used in many OCR applications. These classification methods include statistical methods, Artificial Neural Network, Kernel method and Genetic algorithm. The comparisons of results of the relevant works found in the literature survey are presented in this section. As per the comparison statement, the Artificial Neural Network based approach classification give better performance results than any other classification in terms of accuracy, adaptability and usability. In view of this, the Artificial Neural network based approach is proposed for Classification of mizo

character recognition system. The neural networks classifiers under consideration for mizo OCR are Back Propagation Algorithm (BPA), Learning Vector Quantization (LVQ), Radial Basis Function (RBF), and Recurrent Neural Network (RNN).

Here, an attempt is made to analyze these four types of neural networks and compare their performance to select the best method for implementation of mizo character recognition system. The overall performance of the OCR depends on the classification method. Further, the performance of classification mostly depends on the nature of the pattern of the character and their feature vectors. In this work, fifty four (54) features have been extracted from each character which is used as an input vector for the input layer of the network. The proposed neural network is trained with training dataset (dataset#1) which is comprises of 29 lowercase, 29 upper case letters, 10 numerical and 25 special characters with different fonts such as Arial, Cambria, Tahoma and Times new romans. The total number of prototype characters is then 93x4=372 for training the network. After the network is trained, the neural network classifier is tested with testing dataset (dataset#2). The dataset#2 is comprises of doc#1, doc#2, doc#3, and doc#4. These datasets are extracted from real-life documents such as Laser print document, Vanglaini local newspapers, Mizo Bible, and Kristian Hla Bu. There are 2320 characters in the testing dataset considering only 12 points font size. As there are 93 different classes in mizo characters, the output layer of neural network have 93 output vectors. The algorithm of the networks is program in MATLAB 7.12 and their results are compared based upon their perfection in the character recognition which is shown in the following table.

Table 5.16: Comparison of different Neural Network based Classifier

Neural Network Classifier	No of Character Tested	Correctly Recognize Character	Mis-recognise Character	Accuracy (%)	MSE	Time Taken (Second)
BP Neural Network	2320	2312	8	99.52	0.98	73.17
RBF Neural Network	2320	1309	11	99.12	0.98	74.83
LVQ Neural Network	2320	2260	60	97.22	0.97	77.53
Recurrent Neural Network	2320	2297	23	98.68	0.98	77.53

As per the above comparison statement, the classification (recognition) using Back Propagation Neural Network give better performance results than any other neural network classifier in terms of accuracy and the speed. In view of the experimental results, we concluded that the Multilayer Back Propagation Neural Network may be used for implementation mizo character recognition system.

CHAPTER 6

SUMMARY AND CONCLUSIONS

The research work mainly focused on the development of Optical Character Recognition System for mizo script. The current OCR available in the market cannot be used for recognition of mizo characters due existence of special characters in mizo alphabets. Mizo alphabets are derived from English alphabets but it is uniquely different in some characters like a dotted below t character i.e. “ṭ” and presents of circumflex ^ in all six vowels such as “â”, “âw”, “ê”, “î”, “ô” and “û”. The design and implementation methodology involved preprocessing, segmentation of characters, feature extraction and artificial neural network based approach classification (recognition) for mizo characters. In chapter 1, we describes about introduction to character recognition system, objective of the proposed research work, Application of character recognition system, problems, recent trends and movements, motivation for the present work, literature survey of Latin and Indian languages OCR.

In chapter 2, it was discussed about the preprocessing implementation methodology which is a preliminary processing step to make the raw data usable for segmentation, feature extraction and classification. During the research work we encountered various kinds of problems like the scanned documents have certain noises like Gaussian noise, salt & paper noise, marginal noises due to printer, scanner, print quality, age of the documents, etc. The presents of noise in the scanned document reduces the accuracy of subsequent tasks of Character Recognition systems. An attempted was made to remove these noises using median filter, wiener filter, and average filter. In our experiment, the median filter performance is better than any other

noise filter specially for removing salt & pepper noise and Gaussian noise. The marginal noises comprising of textual noise and non-textual noise presents during scanning of thick documents, a simple and efficient algorithm have been developed using combination of projection profile and connected component analysis. We have also encountered problems while scanning process, the document is sometimes placed incorrectly resulting skewed images resulting poor recognition accuracy. A new algorithm have been developed based on Hough transform which have been tested with sample of 20 skew angle image files having skewed angle ranging from -30 degree to +45 degree. The experimental result is quite satisfactory as the average accuracy is as good as 97.17% with and average error rate of 4.35% and the average execution time is 0.203 seconds. In the final part of preprocessing, an effective thinning algorithm is an ideally solution to remove all redundant pixels and retain the significant aspects of the pattern under process. The algorithm have been developed and tested with on different image input data in both cases discrete and cursive. A preserved smooth skeleton was obtained.

In chapter 3, it was discussed about the implementation methodology of segmentation techniques for use in the mizo character recognition system. The accuracy of character recognition heavily depends upon segmentation phase. Incorrect segmentation leads to incorrect recognition. In this research work, we have encountered problems in segmentation of Mizo characters due to special symbols like â, ê, î, ô, û, and ṭ presents in every Mizo text. In order to overcome the problems, we have developed a hybrid techniques using a combination of projection profile, connected component, bounding box and morphological dilation to enable to correctly segment all the Mizo characters. As a result of experiment, the proposed segmentation algorithms give a very good result of 100% accuracy with four test document sample having 93 lines, 483

words, and 2320 characters. In this work, we have analyzed and study the existing segmentation methods for which the comparison statement have been made with the proposed solution. While comparing with the existing segmentation methods, the proposed hybrid segmentation method performance is much better than the existing method. In chapter 4, Feature extraction methodology for Mizo characters have been discussed in which a hybrid feature extraction method has been developed for Mizo characters. The feature extraction is one of the most challenging tasks in character recognition system. Different feature methods are designed for different representation of the characters which means a feature extraction method that proves to be successful in one application may turn out not to be very useful in another application. Further the type of format of the extracted features must match the requirement of the chosen classifier. As a result, we have developed a hybrid approach feature extraction algorithms giving a very good result of 99.10 % accuracy when testing with 2320 sample data set. In this work, we have analyzed and study the existing feature extraction methods for which the comparison statement have been made with the proposed solution. The comparison statement shows that the proposed hybrid feature extraction method performance is much better than the existing feature extraction method.

In chapter 5, Artificial Neural Network based approach classification is proposed for Mizo character recognition system. The Classification is one of the most important part of character recognition system, here we have investigated various types of classification methods used in many OCR applications. These classification methods include statistical methods, Artificial Neural Network, Kernel method and Genetic algorithm. Among these, the Artificial Neural Network based approach classification give better performance results than any other classification in terms of accuracy, adaptability and usability. In view of this, the Artificial Neural network based approach

is proposed for Classification of Mizo character recognition system. The neural networks classifiers under consideration for Mizo OCR are Back Propagation Algorithm (BPA), Learning Vector Quantization (LVQ), Radial Basis Function (RBF), and Recurrent Neural Network (RNN). Here, an attempt is made to analyze these four types of neural networks and compare their performance to select the best method for implementation of Mizo character recognition system. The overall performance of the OCR depends on the classification method. Further, the performance of classification mostly depends on the nature of the pattern of the character and their feature vectors. In this work, fifty four (54) features have been extracted from each character which is used as an input vector for the input layer of the network. The proposed neural network is trained with training dataset (dataset#1) which is comprises of 29 lowercase, 29 upper case letters, 10 numerical and 25 special characters with different fonts such as Arial, Cambria, Tahoma and Times new romans. The total number of prototype characters is then 93x4=372 for training the network. After the network is trained, the neural network classifier is tested with testing dataset (dataset#2). The dataset#2 is comprises of doc#1, doc#2, doc#3, and doc#4. These datasets are extracted from real-life documents such as Laser print document, Vanglaini local newspapers, Mizo Bible, and Kristian Hla Bu. There are 2320 characters in the testing dataset considering only 12 points font size. As there are 93 different classes in Mizo characters, the output layer of neural network have 93 output vectors. In our experimental results, the Back propagation neural network achieved the accuracy rate of 99.52 % and the time taken for recognition of 2320 characters is about 73.17 seconds leaving 8 characters are misrecognized. In view of this, we have concluded that the Back Propagation neural network is the most suitable classifier for Mizo character recognition system. The recognised characters are encoded into Unicode standard and formatting the encoded character into a meaningful words.

REFERENCES

Acharya J., Gadhiya S. and Raviya K. (2013). Segmentation Techniques for image analysis: A review, *International Journal of Computer Science and Management Research,* **2(1)**:1218-1221.

Agarwal S. and Hemarjani D. N. (2013). Offline handwritten character recognition with Devanagari script, *IOSR Journal of Computer Engineering (IOSR-JCE),* **12(2)**:82-86.

Agarwal A., Rani R. and Dhir R. (2012). Handwritten Devanagari Character Recognition Using Gradient Features, *International Journal of Advanced Research in Computer Science and Software Engineering*, **2(5)**:85-90.

Agnihotri V. P. (2012). Offline Handwritten Devanagari Script Recognition, *International Journal of Information Technology and Computer Science*, **8**:37-42.

Alam M. M. and Kashem M. A. (2010). A complete Bangla OCR System for Printed Characters, *International Journal of Computer and Information Technology*, **1(1)**:30-35.

Albus J. E., Anderson R. H., Brayer J. M., DeMori R., Feng H. Y., Horowitz S. L. and Vamos T. (2012). Syntactic pattern recognition, applications, *Springer Science & Business Media*.

Annadurai S. and Shanmugalakshmi R. (2007). Fundamentals of digital image processing, *Pearson Education India*.

Aradhya V. N. M., Kumar G. H. and Noushath S. (2007). Robust Unconstrained Handwritten Digit Recognition Using Radon Transform, *IEEE International Conference on Signal Processing, Communications and Networking*, pp.626-629.

Ayyaz M. N., Javed I. and Mahmood W. (2012). Handwritten Character Recognition Using Multiclass SVM Classification with Hybrid Feature Extraction, *Pakistan Journal of Engineering & Applied Science*, **10**:57-67.

Barve S. (2012). Optical Character Recognition using Artificial Neural Network, *International Journal of Advanced Research in Computer Engineering & Technology*, **1(4)**:131-133.

Beale M. H., Hagan M. T. and Demuth H. B. (2010). Neural Network Toolbox 7 User's Guide, *The Mathwork Inc*.

Bertolami R., Uchida S., Zimmermann M. and Bunke H. (2007). Non-Uniform Slant Correction for Handwritten Text Line Recognition, *Ninth IEEE International Conference on Document Analysis and Recognition*, **1**:18-22.

Bharathi J. and Reddy P. C. (2013). Segmentation of Text Lines Using Sub-Image Profile for Machine Printed Telugu Script, *International Journal of Computer Engineering and Technology (IJCET)*, **4(6)**:181-191.

Biswas C., Bhattacharya U. and Parui S. K. (2012). HMM based online handwritten Bangla character recognition using Dirichlet distributions, *IEEE International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 600-605.

Blumenstein M., Verma B. and Basli H. (2003). A novel feature extraction technique for the recognition of segmented handwritten characters. *IEEE Proceedings of Seventh International Conference on Document Analysis and Recognition*, pp. 137-141.

Bo G. and Xianwu H. (2006). SVM multi-class classification, *Journal of Data Acquisition & Processing*, **21(3)**:334-339.

Breuel T. M., Ul-Hasan A., Al-Azawi M. A. and Shafait F. (2013). High-performance OCR for printed English and Fraktur using LSTM networks, *IEEE 12th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 683-687.

Burges C. J. C. (1996). Simplified support vector decision rules, *Proceedings of the 13th IEEE International Conference on Machine Learning*, **96**:71–77.

Burges C. J. C. (1998). A tutorial on support vector machines for pattern recognition, *Knowledge Discovery and Data Mining*, **2(2)**:1-43.

Burling R. (1957). Lushai Phonemics, *Indian Linguistics*, **17**:148-155.

Casey R. G. and Lecolinet E. (1996). A survey of Methods and Strategies in Character Segmentation, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **18 (7)**:690–706.

Chang C. C. and Lin C. J. (2011). LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology, **2(3):**27.

Cheriet M., Kharma N., Liu C. L. and Suen C. Y. (2007). Character Recognition Systems – A Guide for Students and Practitioners, *John Wiley & Sons Inc*., USA.

Chhangte L. (1986). A Preliminary Grammar of the Mizo Language, *Master's thesis*, University of Texas, Arlington.

Cristianini N. and Taylor J. S. (2000). An Introduction to Support Vector Machines and other kernel based Learning method, *Cambridge University Press, New York.*

Das M. S., Reddy C. R. K., Govardhan A. and Saikrishna G. (2010). Segmentation of overlapping text lines, characters in printed Telugu text document images, *International Journal of Engineering Science and Technology*, **2(11):**6606-6610.

Deshpande P. S., Malik L. and Arora S. (2008). Fine classification & recognition of hand written Devnagari characters with regular expressions & minimum edit distance method, *Journal of Computers*, **3(5):**11-17.

Dhandra B. V., Malemath V. S., Mallikarjun H. and Hegadi M. H. R. (2006). Skew detection in Binary image documents based on Image Dilation and Region labeling Approach, *IEEE 18th International Conference on Pattern Recognition*, **2**:954-957.

Dhiman M. S. and Singh P. D. A. (2013). Tesseract vs GOCR a comparative study, *International Journal of Recent Technology and Engineering*, **2(4):**80-83.

Dong J. X., Krzyzak A., and Suen C. Y. (2005). Fast SVM training algorithm with decomposition on very large data sets, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27(4):**603–618.

Downs T. (2001). Exact simplification of support vector solutions, *Journal of Machine Learning Research*, **2**:293–297.

Drezet P. M. L. and Harrison R. F. (2001). A new method for sparsity control in support vector classification and regression, *Pattern Recognition*, **34(1):**111–125.

Duda R. O., Hart P. E., and Stork D. G. (2001). Pattern Classification - 2nd edition, *Wiley Interscience*, New York.

Duin R. P. W. (2002). The combining classifiers: To train or not to train, IEEE *Proceedings of the 16th International Conference on Pattern Recognition, Canada*, **2**:765–770.

Elavarasan N. and Mani K. (2015). A Survey on Feature Extraction Techniques, *International Journal of Innovative Research in Computer and Communication Engineer*, **3(1):**52-55.

Felici J. (2011). The complete manual of typography: a guide to setting perfect type, *Adobe Press*.

Fumera G. and Roli F. (2005). A theoretical and experimental analysis of linear combiners for multiple clas sifier system, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **27(6):**942-956.

Gales M. and Young S. (2008). The application of hidden Markov models in speech recognition, *Foundations and Trends in Signal Processing*, **1(3):**195-304.

Gaurav D. D. and Ramesh R. (2012). A feature extraction technique based on character geometry for character recognition. *arXiv preprint arXiv*:1202.3884.

George A. and Nicolai P. (2013). Trainable COSFIRE Filters for Keypoint Detection and Pattern Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(2):490-503.

Giacinto G. and Roli F. (2001). An approach to the automatic design of multiple classifier systems, *Pattern Recognition Letters*, **22(1):**25–33.

Gupta D. and Nair L. M. (2013). Improving OCR by effective pre-processing and Segmentation for Devanagiri script: A Quantified Study, *Journal of Theoretical and Applied Information Technology*, **52(2):**142-153.

Henderson E. J. (1948). Notes on the syllable structure of Lushai, *Bulletin of the School of Oriental and African Studies*, **12(3-4):**713-725.

Hussain J. and Lalthlamuana (2014). Artificial Neural Network based approach for Mizo character recognition system, *Science Vision*, **14**(2):61-66.

Hussain J. and Lalthlamuana (2014). Unicode Mizo character recognition system using Multilayer Neural Network model, *International Journal of Soft Computing and Engineering (IJSCE)*, **4(2):**85-89.

Jayaraman S., Esakkirajan S. and Veerakima T. (2009). *Digital Image Processing, Tata McGraw Hill Education Pvt. Ltd., New Delhi*.

John J. and Balakrishnan K. (2013). A system for offline recognition of handwritten characters in Malayalam script, *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, **5(4):**53-59.

Jubair M. I. and Banik P. (2012). A Simplified Method for Handwritten Character Recognition from Document Image, *International Journal of Computer Applications*,

51(14): 50-54.

Kamble S. N. and Kamble M. (2011). Morphological Approach for Segmentation of Scanned Handwritten Devnagari Text, *International Journal of Computer Science & Technology*, **2(4):** 322-326.

Kanale P. B. and Chitnis S. D. (2010). Handwritten Devanagari Character Recognition Using Artificial Neural Network, *Journal of Artificial Intelligence,* **4**:55-62.

Kimura F., Takashina K., Tsuruoka S. and Miyake Y. (1987). Modified Quadratic Discriminant Functions and the Application to Chinese Character Recognition, *IEEE Transactions on Pattern Analysis & Machine Intelligence,* **9(1):**149-153.

Kinhekar S. and Govilkar S. S. (2014). Comparative study of segmentation and recognition methods for handwritten Devnagari script, *International Journal of Computer Applications*, **105(9):**34-39.

Kumar B., Kumar N., Palai C., Jena P. K. and Chattopadhyay S. (2012). Optical character recognition using ant miner algorithm: a case study on oriya character recognition, *International Journal of Computer Application,* **57(7):**17-22.

Kumar D. and Singh D. (2012). Modified approach of Hough transform for skew detection and correction in documented images, *International Journal of Research in Computer Science,* **2(3):**37-40

Kumar S. D., Kamalapuram S. K. and Kumar A. B. (2013). Kannada character recognition system using neural network, *International Journal of Internet Computing*, **1(2):**33-35

Kumar M., Jindal M. K. and Sharma R. K. (2014). Segmentation of Isolated and Touching Characters in Offline Handwritten Gurmukhi Script Recognition, *International Journal of Information Technology and Computer Science (IJITCS)*, **6(2):**58-63.

Kunte R. S. and Samuel R. D. S. (2006). Script Independent Handwritten Numeral Recognition, *IET International Conference on Visual Information Engineering*, pp.94-98.

Leedham G., Varma S., Patankar A. and Govindaraju V. (2002). Separating text and background in degraded document Images – a comparison of global thresholding techniques for multi-stage thresholding, *Proceedings of the Eighth IEEE International Workshop on Frontiers in Handwriting Recognition*, pp.222-249.

Liu C. L., Jaeger S., and Nakagawa M. (2004). Online Handwritten Chinese Character Recognition: The state of the art, *IEEE Transactions on Pattern Analysis on Machine Intelligence*, **26(2):**198–213.

Lu Y. and Tan C. L. (2003). Improved Nearest Neighbor Based Approach to Accurate Document Skew Estimation, *Proceedings of the seventh IEEE International Conference on Document Analysis and Recognition*, **1**:503-507.

Maini R. and Aggarwal H. (2010). A Comprehensive Review of Image Enhancement Techniques, *Journal of Computing*, **2(3):**8-13.

Mamatha H. R., Murthy K. S., Amrutha K. S., Anusha P. and Azeemunisa R. (2012). Artificial Immune system based recognition of handwritten Kannada numerals, *Advanced Materials Research, Trans-Tech Publications*, **433-440**:900-906.

Mayank P., Sharma S., Sharma A. K., and Gupta J. P. (2011). Anatomy of Pattern Recognition, *Indian Journal of Computer Science and Engineering*, **2(3):**371-378.

Mori S., Nishida H. and Yamada H. (1999). Optical Character Recognition, *John Wiley & Sons Inc., New York, USA.*

Mamatha H. R., Murthy K. S., Veeksha A. V., Vokuda P. S. and Lakshmi M. (2011). Recognition of Handwritten Kannada Numerals Using Directional Features and K-Means, *IEEE International Conference on Computational Intelligence and*

Communication Networks (CICN), pp. 644-647.

Murthy O. V. R. and Hanmandhu M. (2011). Zoning based Devanagari Character Recognition, *International Journal of Computer Applications*, **27(4):**21-25.

Naser M. A., Hamid N. I. B. and Hoque M. A. (2009). Projection based feature extraction process for Bangla script: A modified approach, *International Conference on Software Technology and Engineering, Chennai (India).*

Nikolaev N. Y. and Iba H. (2003). Learning polynomial feed forward neural networks by genetic programming and backpropagation, *IEEE Transactions on Neural Networks*, **14(2):**337-350.

Ntzios K., Gatos B., Pratikakis I., Konidaris T. and Perantonis S. J. (2007). An old greek handwritten OCR system based on an efficient segmentation-free approach, *International Journal of Document Analysis and Recognition (IJDAR)*, **9**(2-4):179-192.

Otsu N. (1979). A threshold selection method from gray-level histograms, *IEEE Transactions on Systems, Man and Cybernetics,* **9(1):** 62–66.

Patil V. and Shimpi S. (2011). Handwritten English Character Recognition Using Neural Network, *Elixir Computer Science & Engineering*, **41(2011):**5587-5591.

Pal U., Sharma N., Wakabayashi T. and Kimura F. (2007). Off-line handwritten character recognition of Devnagari script, *IEEE 9th International Conference on Document Analysis and Recognition*, **1**:496-500.

Pal U. and Chaudhuri B. B. (2004). Indian script character recognition: a survey, *Pattern Recognition*, **37(9):**1887–1899.

Patel C., Patel A. and Patel D. (2012). Optical character recognition by open source OCR tool tesseract: A case study, *International Journal of Computer Applications*, **55(10):**50-56.

Pedreira C. E. (2006). Learning vector quantization with training data selection, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28(1):**157-162.

Peng L., Liu C., Ding X., Jin J., Wu Y., Wang H. and Bao Y. (2010). Multi-font printed Mongolian document recognition system, *International Journal on Document Analysis and Recognition (IJDAR),* **13(2):**93-106.

Plamondon R. and Srihari S. N. (2000). Online and Offline Handwriting Recognition: A Comprehensive Survey, *IEEE Transaction on Pattern Analysis and Machine Intelligence,* **22(1):**63-84.

Prasad J. R. and Kulkarni U. (2015). Gujarati character recognition using weighted k-NN and Mean $\chi 2$ distance measure, *International Journal of Machine Learning and Cybernetics*, **6(1):** 69-82.

Prasad K., Nigam D. C. and Lakhotiya A. (2013). Character Recognition Using Matlab's Neural Network Toolbox, *International Journal of u- and e- Service, Science and Technology*, **6(1):**13-20.

Prasad K., Nigam D. C., Lakhotiya A. and Umre D. (2013). Character Recognition using Matlab's Neural Network Toolbox, *International Journal of u- and e- Service, Science and Technology*, **6(1):**13-20.

Priyanka N., Pal S. and Mandal R. (2010). Line and Word segmentation approach for printed documents, *International Journal of Computer Application*, **4(1):**30-36.

Ramappa M. H. and Krishnamurthy S. (2013). A Comparative Study of Different Feature Extraction and Classification Methods for Recognition of Handwritten Kannada Numerals, *International Journal of Database Theory and Application*, **6(4):**71-90.

Ramteke R. J. (2010). Invariant Moments Based Feature Extraction for Handwritten Devanagari Vowels Recognition, *International Journal of Computer Applications*, **1(18):**1–5.

Rashid S. F., Shafait F. and Breuel T. M. (2012). Scanning Neural Network for Text Line Recognition, *10th IAPR International Workshop on Document Analysis Systems*, pp.105-109.

Rawat S., Kumar K. S., Meshesha M., Balasubramanian A., Sikdar I. D. and Jawahar C. V. (2006). A Semi-Automatic Adaptive OCR for Digital Libraries, *Proceedings of 7th International Conference on Document Analysis Systems*, **3872**:13–24.

Razzak M. I., Anwar F., Husain S. A., Belaid A. and Sher M. (2010). HMM and fuzzy logic: A hybrid approach for online Urdu script-based languages' character recognition, *Knowledge-Based Systems*, **23(8):**914-923.

Rodrigues, José R., Thomé A. C. G., and Carlos A. (2000). Cursive character recognition–a character segmentation method using projection profile-based technique, *The 4th World Multi-conference on Systemics, Cybernetics and Informatics SCI 2000 and The 6th International Conference on Information Systems, Analysis and Synthesis*.

Russ J. C. (2011). The Image Processing Handbook, 6th Edition, *CRC Press*.

Rusu A. and Govindaraju V. (2004). Handwritten CAPTCHA: Using the difference in the abilities of humans and machines in reading handwritten words, *IEEE Ninth International Workshop on Frontiers in Handwriting Recognition*, pp. 226-231.

Saba T., Sulong G., and Rehman A. (2011). Document image analysis: issues, comparison of methods and remaining problems, *Artificial Intelligence Review*, **35(2):**101-118.

Said H. E. S., Tan T. N. and Baker K. D. (2000). Personal Identification Based on Handwriting, *Pattern Recognition*, **33(1):**149-160.

Saraf V. and Rao D. S. (2013). Devnagari Script Character Recognition Using Genetic Algorithm for Get Better Efficiency, *International Journal of Soft Computing and Engineering*, **2(4):**374-377.

Saxena S. and Gupta P. C. (2012). A novel approach of handwritten Devanagari character recognition through feed forward Back Propagation Neural Network, *International Journal of Computer Applications*, **52(20):**33-41.

Shaik A. S., Hossain G. and Yeasin M. (2010). Design, development and performance evaluation of reconfigured mobile Android phone for people who are blind or visually impaired, *Proceedings of the 28th ACM International Conference on Design of Communication*, pp.159-166

Sharma O. P., Ghose M. K., Shah K. B. and Thakur B. K. (2013). Recent Trends and Tools for Feature Extraction in OCR Technology, *International Journal of Soft Computing and Engineering (IJSCE)*, **2(6):**220-223.

Sharma D V and Lehal G S (2006), An Iterative Algorithm for segmentation of isolated handwritten words in Gurmukhi script, *the 18th International conference on Pattern Recognition, IEEE Computer society*, **2**:1022-1025.

Shelke S and Apte S (2011), A Multistage Handwritten Marathi Compound Character Recognition Scheme using Neural Networks and Wavelet Features, *International Journal of Signal Processing, Image Processing and Pattern Recognition*, **4(1):**81-94.

Shrivastava V and Sharma N (2012). Artificial Neural Network Based Optical Character Recognition, *International Journal of Signal Processing, Image Processing and Pattern Recognition*, **3(5):**73-80.

Siddharth K. S., Jangid M., Dhir R., and Rani R. (2011). Handwritten Gurmukhi Character Recognition Using Statistical and Background Directional Distribution Features, *International Journal on Computer Science and Engineering*, **3(6):**2332-2345.

Singh C., Bhatia N., Kaur A. (2008). Hough transform based fast skew detection and accurate Skew correction methods, *Pattern Recognition*, **41(12):**3528–3546.

Singh B., Mittal A., Ansari M. A. and Ghosh D. (2011). Handwritten Devanagari Word

Recognition: A Curvelet Transform Based Approach, *International Journal on Computer Science and Engineering*, **3(4):**1658-1665.

Sivanandam S. N. and Deepa S. N. (2006). Introduction to neural networks using Matlab 6.0. *Tata McGraw-Hill Education*.

Sousa S. I. V., Martins F. G., Alvim-Ferraz M. C. M. and Pereira M. C. (2007). Multiple linear regression and Artificial Neural Networks based on principal components to predict ozone concentrations, *Environmental Modeling & Software*, **22(1):**97-103.

Vapnik V. (2013). The Nature of Statistical Learning Theory, *Springer Science & Business Media*.

Vithlani P. (2014). Pre-processing Techniques in Character Recognition, *International Journal of Advanced Research in Computer Science and Software Engineering*, **4(11):**601-604.

Webb A. R. (2003). Statistical pattern recognition, *John Wiley & Sons*.

Wong K. Y., Casey R. G. and Wahl F. M. (1982). Document analysis system, *IBM Journal of Research and Development*, **26(6):**647-656.

Yang C. C. (2006). Image enhancement by modified contrast-stretching manipulation, *Optics & Laser Technology*, **38(3)**:196-201.

Zhang P., Bui T. D. and Suen C. Y. (2004). Extraction of hybrid complex wavelet features for the verification of handwritten numerals, *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition*, pp. 347-352.

Zramdini A. and Ingold R. (1998). Optical Font Recognition using Typographical Features, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, **20(8):**877-882.