A NOVEL OFF - LINE CHARACTER RECOGNITION: AN MLP APPROACH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
In
Telematics and Signal Processing

By
JAGANMOHAN RAO.GRANDHE
20507021

Department of Electronics & Communication Engineering
National Institute of Technology
Rourkela – 769008.

2007

A NOVEL OFF - LINE CHARACTER RECOGNITION: AN MLP APPROACH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
In
Telematics and Signal Processing

By
JAGANMOHAN RAO. GRANDHE

Under the Guidance of
Prof. G. S. RATH

Department of Electronics & Communication Engineering
National Institute of Technology
Rourkela – 769008.
2007

NATIONAL INSTITUTE OF TECHNOLOGY
ROURKELA

CERTIFICATE

This is to certify that the Thesis Report entitled “A NOVEL OFF - LINE CHARACTER RECOGNITION: AN MLP APPROACH” submitted by Mr. JAGANMOHAN RAO GRANDHE in partial fulfillment of the requirements for the award of Master of Technology degree Electronics and Communication Engineering with specialization in “Telematics and Signal Processing” during session 2006-2007 at National Institute Of Technology, Rourkela (Deemed University) and is an authentic work by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other university/institute for the award of any Degree or Diploma.

Prof. **G. S.RATH**
Dept. of E.C.E
National Institute of Technology
Rourkela-769008
Email:gsrath@nitrkl.ac.in

Date:

Acknowledgement

First of all, I would like to express my deep sense of respect and gratitude towards my advisor and guide **Prof. G. S. Rath**, who has been the guiding force behind this work. I am greatly indebted to him for his constant encouragement, invaluable advice and for propelling me further in every aspect of my academic life. His presence and optimism have provided an invaluable influence on my career and outlook for the future. I consider it my good fortune to have got an opportunity to work with such a wonderful person.

Next, I want to express my respects to **Prof. G. Panda, Prof. K. K. Mahapatra, Prof. S.K. Patra** and **Dr. S. Meher** for teaching me and also helping me how to learn. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I would like to thank all faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T. Rourkela for their generous help in various ways for the completion of this thesis.

I would also like to mention the names of **Balaji** and **Pradeep** for helping me a lot during the thesis period.

I would like to thank all my friends and especially my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I've enjoyed their companionship so much during my stay at NIT, Rourkela.

I am especially indebted to my parents for their love, sacrifice, and support. They are my first teachers after I came to this world and have set great examples for me about how to live, study, and work.

Jaganmohan Rao. Grandhe
Roll No: 20507021
Dept. of ECE, NIT, Rourkela

CONTENTS

ABSTRACT

The purpose of this thesis work is to explore the possibility of efficient man-machine communication through printed documents. An attempt has been made to show the pattern recognition techniques i.e., KNN classifier helpful in recognition of machine printed characters and Artificial Neural Networks may be used to represent and recognize printed English characters of any font and size.

In our current work the machine printed document images are scanned by a front end video scanner and are applied to noise removal techniques using smoothing and sharpening filters. The noiseless images are digitized into a bi-level image using Ni-Black proposed binarization technique and proposed adaptive thresholding algorithm using Laplacian sign. Our work is split into three parts. The first part deals with segmentation and thinning. The output of this phase is thinned character image. The second part involves features are extracted from thinned image. The third part deals with KNN classifiers and training of the multilayer perceptron and recognizing characters after the system is trained.

Automatic character recognition system promises to hold great future in Automatic office information processing system by integrating with multimedia, like Graphics, image and voice, into a single work station.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

The digital revolution has drastically changed our perspective of the concept of communication and connectivity. Though the advantages of this revolution are obvious and are quite desirable, they have burgeoned security problems which previously were not even conceivable. For example, remote login where a user can enter his database from anywhere in the world; for this system to be foolproof we need a strong authentication methodology. The traditional approaches like password based authentication, location based authentication have been under rigorous attacks. Instead of improving the security of these systems, one different approach of dealing authentication is to use BIOMETRICS [2] - the science of identifying or verifying the identity of a person based on physiological or behavioral characteristics. Physiological characteristics include fingerprints, iris, hand geometry and facial image. The behavioral characteristics are actions carried out by a person in a characteristic way and include signature, handwriting and voice.

1.1 CHARACTER RECOGNITION SYSTEMS

Character recognition systems are basically into two:

Off-line character recognition:

The system accepts image as input from the scanner, it is more difficult than On-line character recognition system because of unavailability of contextual information and prior knowledge the like text position, size of text, order of strokes, start point and stop point, further there are noises in images, while the noises in On-line character recognition[11] near to be absent. Eg., Machine Printed character recognition.

On-line character recognition:

The system accepts the moment of pen from the hardware such as graphic tablet, light pen and there is a lot of information during input process available such as current position, moment’s direction, start points, stop points and stroke orders [12]. Eg., Handwritten character recognition.

The applications of the character recognition systems are

- **Postal address systems**, the addresses which are written by senders are required to identifying and recognized through OCR.
- In **Signature Verification system**, the signatures of different persons are recognized through OCR to avoid forgeries.
- Recognition of characters from **filled applications**: Any education institute invites applications to join into new courses. The students are requiring filling these applications. From these filled applications, the characters are written by different students are require to recognize through OCR.

1.2 OPTICAL CHARACTER RECOGNIZER

The basic definitions of Optical Character Recognizer are

- Often abbreviated **OCR**, optical character recognition[21] refers to the branch of computer science that involves reading text from paper and translating the images into a form that the computer can manipulate (for example, into ASCII codes).
- **Optical character recognition**[23], is computer software designed to translate images of handwritten or typewritten text (usually captured by a scanner) into machine-editable text, or to translate pictures of characters into a standard encoding scheme representing them (e.g. ASCII or Unicode).

  Unicode is an industry standard designed to allow text and symbols from all of the writing systems of the world to be consistently represented and manipulated by computers.

1.3 TYPES OF OPTICAL CHARACTER RECOGNITION SYSTEMS

The different types of Optical Character Recognition systems are

1. Machine printed OCR
2. Hand written OCR
3. Cursive OCR
4. MICR
5. Music OCR

Machine Printed OCR

Machine printed characters in a document images are required to recognize through OCR.

Hand Written Character OCR

Hand written characters are written by different persons are identified and recognized through OCR. This recognition is most difficult compared to Machine printed OCR.

Cursive OCR

Cursive is any style of handwriting in which all the letters in a word are connected, making a word one single (complicated) stroke. The recognition of cursive script is done through OCR. It has complex structure.

MICR

Magnetic Ink Character Recognition, or **MICR**, is a character recognition technology adopted mainly by the banking industry to facilitate the processing of bank checks.

MUSIC OCR

Music OCR is the application of optical character recognition to interpret sheet music or printed scores into editable and, often, playable form. Once captured digitally, the music can be saved in commonly used file formats, e.g. (MIDI (for playback) and Music XML (for page layout).

In Chapter 2, the fundamental steps in Digital Image Processing for machine printed documents are explained. The different sections like Data acquisition, Preprocessing, Segmentation, Thinning, Feature extraction, Classification and Knowledge base[28].

In Chapter 3, the different filters are explained and which are useful in Noise removal which is fundamental step in preprocessing [2]. The smoothing filters like mean/average filtering, median filtering and weighted median filters[5] and sharpening filters like Gaussian filtering and Laplacian of Gaussian filtering are explained[1]. And the different binarization methods using local thresholding and global thresholding methods are explained. The simplest and best method which is Ni-Black method [4] and proposed Ni-Black algorithm [7] is explained. For Noisy images the performance is not good using this proposed Ni-Black algorithm. To improve performance in binarization (i.e., for noisy images), the Laplacian proposed the new algorithm [9], that is explained in it.

In chapter 4, the important stage is Segmentation of machine printed documents. In this external segmentation [12] (segmenting the graphical image from document image), internal segmentation [12] (segmenting text image from document image), explicit segmentation [11] (paragraph segmentation) and implicit segmentation [11] (line, word and finally character segmentation) using di-section method are explained. For handwritten documents the newest method Segmentation using Bounding Box analysis [13],[14] techniques is applied. And the next part thinning of segmented characters is required. The proposed thinning algorithm is explained [15].

In chapter 5, the feature extraction of thinned character is explained. The different features like aspect ratio, pixel density, horizontal stroke, vertical stroke, right slant stroke, left slant stroke, north east segment, north west segment, south east segment and south west segment are extracted from each character [18].

In chapter 6, the pattern recognition, the different classification methods are explained. The K-nearest neighbor rule [25] is the one of the statistical classification method is used in classification of different fonts of characters in data set. And also using neural networks, the Multilayer Perceptron neural network [23],[26] is designed and trained with back propogation algorithm and tested with different fonts of characters in data set.

In chapter 7, the results of each chapter are mentioned and explained in briefly.

CHAPTER 2

FUNDAMENTALS OF DIGITAL IMAGE PROCESSING

2. FUNDAMENTALS OF DIGITAL IMAGE PROCESSING

Character recognition comes under applications of image processing. The character samples are stored in a suitable image format in digital form. Each sample in the image form is properly preprocessed, segmented and the required features defining writer invariants are obtained. Then any test sample is taken through similar process and the features obtained are compared with those of standard samples with a specific metric in the classifying stage and the best-matched writer is found. Thus, this module requires an understanding of image processing fundamentals[28].

All image processing applications dealing with images for different purposes and with different objectives need to go through several stages before the result if found. They are image acquisition, image enhancement, image segmentation, representation and description, classification, image restoration, image compression, color image processing etc. Not all the applications go through all the stages but depend on requirement, image type, image storage etc.

The various stages involved in our writer identification application are

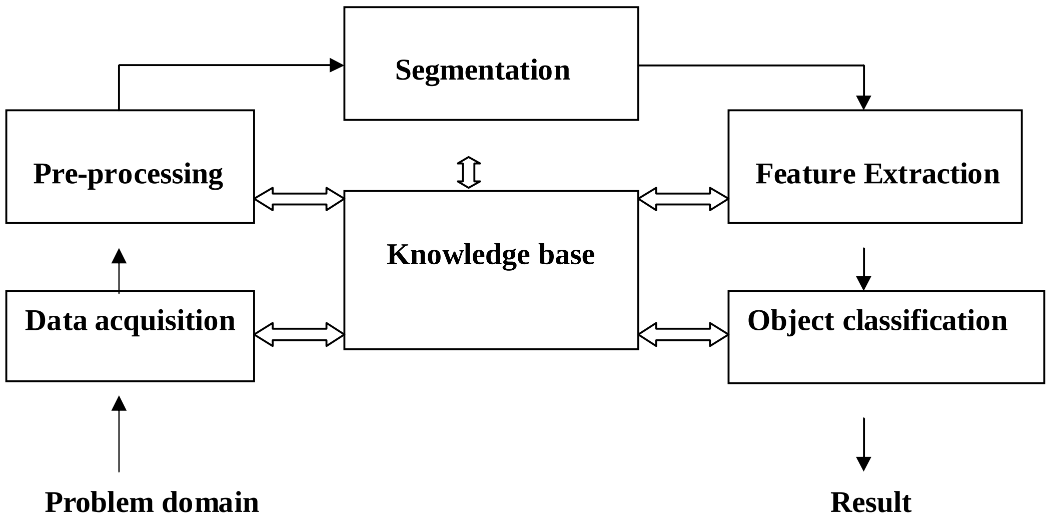


Figure 2.1: The sequence of steps in Digital Image Processing

2.1 Data acquisition

This is the first step in any image processing application where the data required is converted into image form. This project needs the machine printed characters of different fonts to be converted to image form. A scanner (imaging sensor) is used to acquire the required data in digital image format. In the first step the scanner has to transform optical information into digital information. There are different types of scanners producing digital image information in different formats. The format chosen is gray level bmp. The main problem in this first step is to fix some thresholds such as contrast or brightness in order to get the best possible quality and not to alter the original image[29].

2.2 Pre-processing

When data is scanned and digitized, the data may carry some unwanted noise. The idea behind pre-processing is to bring out detail that is obscured, or simply to highlight certain features of interest in an image. Typical processing techniques are image smoothing using filters, contrast stretching, increasing dynamic range, image thresholding, histogram equalization etc. All these can be implemented in both spatial domain and frequency domain. In this project, we use 3*3 neighborhood averaging filter to smoothen the image for any noisy pixels and then binaries the image with a proper threshold assigning black to data pixels and white to background pixels. The images are also thinned to obtain structural shape of the image that is just 1-pixel thick. The thinning algorithm used is a two-iteration algorithm for binary regions with successive passes applied to contour points of an image using 8-neighborhood notation. The problem with this stage is that enhancement is a very subjective area of image processing and processing techniques required for different images for a specific application may also be different and so the techniques and the thresholds are to be carefully chosen.

2.3 Segmentation

In this stage the image is partitioned into its constituent parts or objects i.e. the various elements defining the image are identified and localized. Autonomous segmentation procedure brings the process a long way toward successful solution of imaging problems that require objects to be identified individually and weak or erratic segmentation algorithms almost always guarantee eventual failure. In brief, the more accurate the segmentation, the more likely recognition is to succeed. The various segmentation tasks are detection of discontinuities, edge linking and boundary detection, thresholding etc. In this project, the handwritten samples collected are words in disconnected format where characters are separated by bounding lines. Each word is segmented into characters by calculating horizontal projection profiles [13],[14] and properly thresholding them.

2.4 Feature Extraction

This stage is crucial to any image processing application and it deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another. New variables may be obtained by a linear or nonlinear transformation of the original set of attributes extracted and then variables that are appropriate for the task are then selected from the measured set. In this project, the writer specific features called writer invariants are extracted from the samples. But the writer's handwriting may have intra-personal variations due to various factors. If two individual's handwritings are distinguishable, the intra-author variation is less than the inter-author variation. Using the features extracted, it must be possible to evaluate these variations and distinguish an individual writer.

The features used in this project are mainly computational features some of which cannot be easily evaluated by humans and can be extracted by computational algorithms. A total of ten features for each character are extracted-Aspect ratio, pixel density, right slant stroke, left slant stroke, horizontal stroke, vertical stroke, norst east segment, north west segment, south east segment and south west segment. All these features were appropriately binarized so that binary feature vectors of constant lengths could be formed

2.5 Classification

It is the process that assigns a label to an object based on its descriptors and attributes. This classification involves techniques for assigning test samples to their respective classes automatically. The degree of class separability depends on the technique used and on the choice of descriptors selected for an application. Various classification techniques in use are minimum distance classifier, Bayesian classifier, classification using correlation between images, baye's classifier for Gaussian pattern classes, Neural networks etc. Each technique has its own advantages and disadvantages and is chosen based on the application. In this project, we use Radial basis function for classification.

2.6 Knowledge Base

There is a need for prior knowledge in any image processing application. Knowledge about a problem domain is coded into an image processing system in the form of a knowledge base; this knowledge may be as simple as detailing regions of an image where the information of interest is known to be located thus limiting the search that has to be conducted in seeking that information. The knowledge base also can be quite complex such as an interrelated list of all major possible inspection problem or an image database containing high-resolution satellite images of a region in connection with change-detection applications. In addition to guiding the operation of each processing module, the knowledge base also controls the interaction between modules. This distinction is made in fig by the use of double-headed arrows between the processing modules and the knowledge based, as opposed to single-headed arrows linking the processing modules. In this project, the knowledge base is the stored information of writer feature vectors for all the writers enrolled. This is used in the classification stage to compare the test sample feature vector with those of all writers enrolled.

CHAPTER 3

PREPROCESSING: NOISE REMOVAL, BINARIZATION

3. PRE PROCESSING

3. A NOISE REMOVAL

The fundamental step in digital image processing is the preprocessing. In this stage the noise in document images is removed by different filters. These are mainly classified as smoothing and sharpening filters.

3.1 Smoothing Filters

Smoothing filters are also called low-pass filters [3] because they let low frequency components pass and reduce the high frequency components. The impulse response of a normal low-pass filter implies that all the coefficients of the mask should be positive. Low-pass filtering in effect blurs the image and removes speckles of high frequent noise. Larger masks will result in more blurring effect. To avoid a general amplification or damping of the data the sum of the filter coefficients should be 1.0.

Types of Smoothing Filters:

3.1.1 Mean Filter:

Mean filtering[3] is a simple, intuitive and easy to implement method of *smoothing* images, *i.e.* reducing the amount of intensity variation between one pixel and the next. It is often used to reduce noise in images.

The idea of mean filtering is simply to replace each pixel value in an image with the mean (`average') value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter. Like other convolutions it is based around a kernel, which represents the shape and size of the neighborhood to be sampled when calculating the mean. Often a 3×3 square kernel is used, as shown in Figure 1, although larger kernels (*e.g.* 5×5 squares) can be used for more severe smoothing. (Note that a small kernel can be applied more than once in order to produce a similar - but not identical - effect as a single pass with a large kernel.)

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Figure 3.1: 3×3 averaging kernel often used in mean filtering

The two main problems with mean filtering, which are:

- A single pixel with a very unrepresentative value can significantly affect the mean value of all the pixels in its neighborhood.
- When the filter neighborhood 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.

Both of these problems are tackled by the median filter. The median filter is often a better filter for reducing noise than the mean filter, but it takes longer to compute.

3.1.2 Median Filtering:

The median filter[3] is normally used to reduce noise in an image, somewhat like the mean filter. However, it often does a better job than the mean filter of preserving useful detail in the image.

Like the mean filter, the median filter considers each pixel in the image in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the *mean* of neighboring pixel values, it replaces it with the median of those values. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used.) Figure 3.2 illustrates an example calculation.

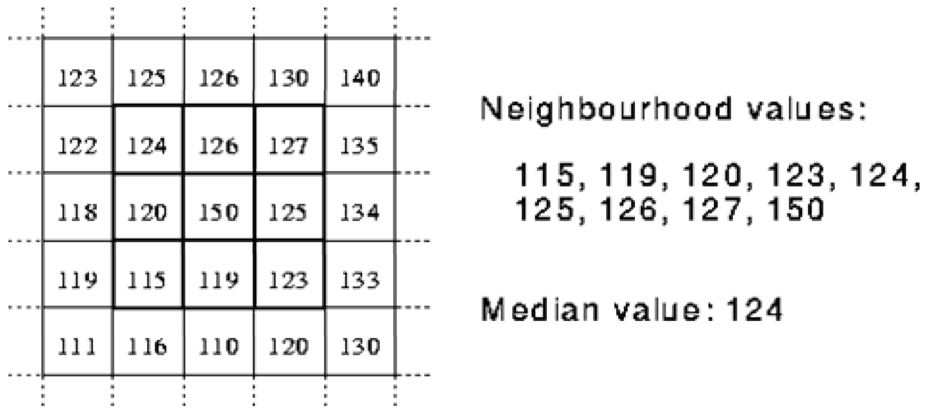


Figure 3.2: 3X3 median value of a pixel neighborhood

As can be seen the central pixel value of 150 is rather unrepresentative of the surrounding pixels and is replaced with the median value: 124. A 3×3 square neighborhood is used here --- larger neighborhoods will produce more severe smoothing.

3.1.3 Weighted Median Filters:

For a raster image, the procedure is to take a number of values in the neighborhood of a pixel, find their median, and use this to replace the value of the pixel. The effect is to remove energy from the image as high and low data values, compared to the surroundings, and is removed. The filter may operate over a large extent to remove objects of size less than the extent. Thus, for example, stars may be removed from a raster image (digitized photograph or CCD image) to provide a background level that may then be subtracted from the original image before performing photometry.

Hence, the median filter may be used for removing spike noise from an image or backgrounding it. However, some undesirable effects may arise if due care is not taken, and on examination, it may be found that the median filter cannot be made to have the desired effect.

Consider the filter skeleton

$$\begin{matrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \end{matrix}$$

This indicates that the values to be taken are the data value itself, together with its side, top, and bottom neighbors for a two-dimensional image. The five values are sorted and the median replaces the center value.

Two basic requirements that might be made are

1. One pixel width high/low streaks, corresponding to emulsion scratches, or in a CCD image, a saturated pixel overspill in a lane, should be removed.
2. Any rectangular block of differing values, such as an intrusive intensity-scaling stepwedge, should remain unaffected.

We define a two-dimensional (2D) weighted median filter (WMF)[5] of extent 2n + 1 to be the array of coefficients: {a(i, j): -n <_ i, j <_ n: a(i, j) nonnegative integers: sum(a(i, j); i, j = -n to. n) odd integer}. The operation of the filter at point (s, t) of a data array D is to take a(i, j) copies of D(s + i, t + j) for i, j =-n . . . . n, a total of S = sum(a(i, j}; i, j = -n . . . . n} values where S is odd. These are sorted into ascending order (L(k); k = 1 . . . . S) and the median M is taken, that is, M = L((S + 1)/2). If I(M - D(s, t)) I > T, where T is a given threshold value, possibly zero, then C(s, t) = M; otherwise C(s, t)= D(s, t), where C is the modified image. (Note that C is used rather than replacing directly in D so as to avoid asymmetries and propagation effects in the filtered image that would then depend on the direction the filter is moved across the data).

3.2 Sharpening Filters

Sharpening filters are used to enhance the edges of objects and adjust the contrast and the shade characteristics. In combination with threshold they can be used as edge detectors. Sharpening or high-pass filters let high frequencies pass and reduce the lower frequencies and are extremely sensitive to shut noise.

3.2.1 Gaussian Filtering:

The Gaussian smoothing [32] operator is a 2-D convolution operator that is used to `blur' images and remove detail and noise. In this sense it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump. This kernel has some special properties which are detailed below.

The Gaussian distribution in 1-D has the form:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

where σ is the standard deviation of the distribution. We have also assumed that the distribution has a mean of zero (*i.e.* it is centered on the line *x*=0). The distribution is illustrated in Figure 3.3.

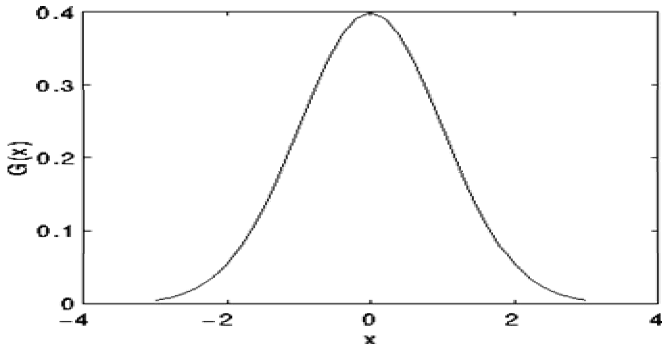


Figure 3.3: 1-D Gaussian distribution with mean 0 and σ =1

In 2-D, an isotropic (*i.e.* circularly symmetric) Gaussian has the form:

$$G(x,y) = \frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

This distribution is shown in Figure 3.4.

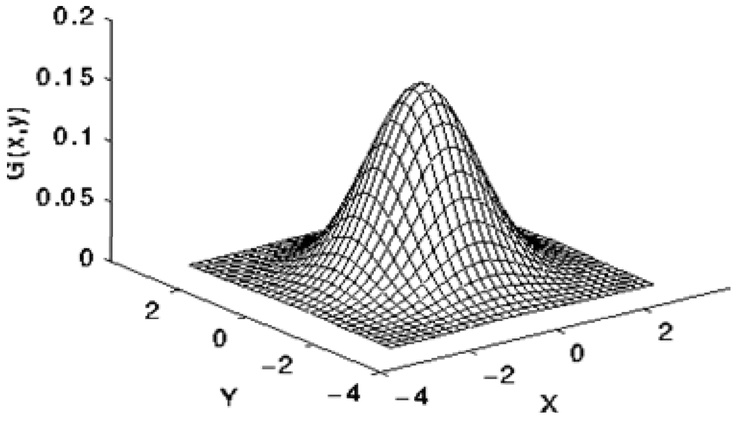


Figure 3.4: 2-D Gaussian distribution with mean (0,0) and σ =1

The idea of Gaussian smoothing is to use this 2-D distribution as a `point-spread' function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point. Figure 3.5 shows a suitable integer-valued convolution kernel that approximates a Gaussian with a σ of 1.0.

$$\frac{1}{273}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline\end{array}$$

Figure 3.5: Discrete approximations to Gaussian function with σ =1.0

3.2.2 Laplacian of Gaussian Filtering:

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian smoothing filter [32] in order to reduce its sensitivity to noise, and hence the two variants will be described together here. The operator normally takes a single gray level image as input and produces another gray level image as output.

The Laplacian L(x,y) of an image with pixel intensity values I(x,y) is given by:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

This can be calculated using a convolution filter.

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. Two commonly used small kernels are shown in Figure 3.6.

$$\begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & 4 & -1 \\ \hline 0 & -1 & 0 \\ \hline\end{array} \qquad \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline\end{array}$$

Figure 3.6: Two commonly used discrete approximations to the Laplacian filter.

Note, we have defined the Laplacian using a negative peak because this is more common; however, it is equally valid to use the opposite sign convention. Using one of these kernels, the Laplacian can be calculated using standard convolution methods.

Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian smoothed before applying the Laplacian filter. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

In fact, since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first of all, and then convolve this hybrid filter with the image to achieve the required result. Doing things this way has two advantages:

- Since both the Gaussian and the Laplacian kernels are usually much smaller than the image, this method usually requires far fewer arithmetic operations.
- The LoG (`Laplacian of Gaussian') kernel can be pre calculated in advance so only one convolution needs to be performed at run-time on the image.

The 2-D LoG function centered on zero and with Gaussian standard deviation σ has the form:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

and is shown in Figure 3.7.

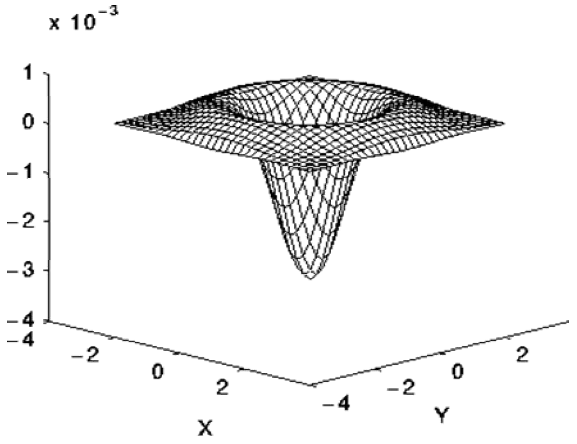


Figure 3.7: The 2-D Laplacian of Gaussian (LoG) function

The *x* and *y* axes are marked in standard deviations (σ).

A discrete kernel that approximates this function (for a Gaussian σ= 1.4) is shown in Figure 3.8

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figure 3.8: Discrete approximation to LoG function with Gaussian σ = 1.4

Note that as the Gaussian is made increasingly narrow, the LoG kernel becomes the same as the simple Laplacian kernels shown in Figure 1. This is because smoothing with a very narrow Gaussian (σ < 0.5 pixels) on a discrete grid has no effect. Hence on a discrete grid, the simple Laplacian can be seen as a limiting case of the LoG for narrow Gaussians.

3. B BINARIZATION

Image Binarization or thresholding [6] is an important step in image processing and computer vision, to extract the object pixels in an image from the background pixels. Image binarization is central to many applications including document image analysis (printed characters, logos, graphical content, and musical scores are important as objects), map processing (fines, legends and characters need to be extracted), scene processing, quality inspection of materials, cell images, segmentation of various image modalities for Non Destructive Testing (NDT) applications (ultrasonic images, eddy current images, thermal images, X-ray computed tomography, laser scanning confocal microscopy, extraction of edge field and spatio temporal segmentation of video images). A number of methods have already been proposed for image binarization but unfortunately, most of them are very much specific for a few applications. Thus, it can be said that a binarization (thresholding) method may work well for one application but its performance can be unsatisfactory for another application.

Bi-level image is used as a pre-processing unit in several applications. The use of binary images decreases computational load for the overall application. These applications include document analysis, optical character recognition system, scene matching, quality inspection of materials etc. The binarization [10] process computes the threshold value that differentiate object and background pixels. Under varying illumination and noise, the binarization can become a challenging job. A number of factors contribute to complicate the thresholding scheme including ambient illumination, variance of gray levels with in the object and the background, inadequate contrast, object shape and size non commensurate with the scene. A wrong selection of threshold value may misinterpret the background pixel and can classify it as object and vice versa, resulting in overall degradation of system performance. The determination of a threshold itself is application dependent since one threshold may work with one application and may not work with other one.

In document analysis, binarization is sensitive to noise, surrounding illumination, gray level distribution, local shading effects, inadequate contrast, the presence of dense non text components such as photographs, etc. while at the same time, the merges, fractures and other deformations in the character shapes affects the threshold value in OCR system.

There are a number of important performance requirements that need to be considered while binarizing gray level images. These include:

- Loss of features after binarizing input image should be zero or minimum.
- The features (objects) with similar relative gray levels should have same binary values in the processed output image.
- The effect of noise on minor gray level variations should be eliminated.

Global binarization methods calculate a single threshold value for the entire image. Pixels having a gray level darker than the threshold value are labeled print (black), otherwise background (white).

Locally adaptive binarization methods [8], on the other hand, compute a threshold for each pixel on the basis of information contained in a neighborhood of the pixel. Some of the methods calculate a threshold surface over the entire image. If a pixel (x, y) in the input image has a higher gray level than the threshold surface evaluated at (x, y), then the pixel (x, y) is labeled as background, otherwise it is labeled as print. Other methods do not use explicit thresholds, but search for print pixels in a transformed image.

The 11 locally adaptive binarization methods are:

1) Bernsen’s method

2) Chow and Kaneko’s method

3) Eikvil et al.’s method

4) Mardia and Hainsworth’s; method

5) Niblack’s method

6) Taxt et al.’s method

7) Yanowitz and Brucksteiri’s method

8) White and Rohrer’s Dynamic Threshold Algorithm

9) Parker’s method

10)White and Rohrer’s Integrated Function Algorithm

11)Trier and Taxt’s method

These methods were selected because they have been frequently referred to in the literature, or appeared to be promising. The first eight methods use explicit thresholds or threshold surfaces, while the last three methods search for print pixels after having located the edges. All these methods are briefly described below.

The four global binarization methods are :

1) Abutaleb’s method

2) Kapur et al.’s method

3) Kittler and Illingworth’r; method

4) Otsu’s method

There is a relationship between the window size used in a locally adaptive binarization method and the size of the objects of interest in the image. If a too small window is used, then it will sometimes be positioned totally within the character and line strokes, and the binarization method may falsely label true print pixels as background. With a large a window, the method will behave too much like a global thresholding method.

3.3 Ni-Black Method

Ni-Black is a local thresholding algorithm[7] that adapts the threshold according to the local mean and the local standard deviation over a specific window size around each pixel location. The local threshold at any pixel (i, j) is calculated as:

$$T(i, j) = m(i, j) + k.\sigma(i, j)$$

Where $m(i, j)$ and $\sigma(i, j)$ are the local sample mean and variance, respectively. The size of the local region (window) is dependent upon the application. The value of the weight 'k' is used to control and adjust the effect of standard deviation due to objects features.
Ni-Black algorithm suggests the value of 'k' to be -0.2.

However, Ni-Black's algorithm suffers from the basic problem of local thresholding, i.e. providing unnecessary details in the binarized images that may not be required in the processing. Niblack fails to adapt large variation in illumination, especially in the document images. The local region analysis using Niblack does not provide any kind of information about the global attributes of the image that may be helpful in the binarization process of badly illuminated images. So, the gray level variations in the document images make it impossible to adapt threshold as will be shown in the results. Another problem it faces the optimum selection of the weight k. Ni-Black algorithm uses fix value of this weight. The fix given value of 'k' may work for document images but for gray-level images with a lot of variations of' gray values, the value of the weight should not be fixed but to change from images to images depending upon their gray-level distributions, and therefore, the value of 'k' should be calculated at run-time.

The simplest method to convert a gray scale image into a binary image is to compute mean of the image and set the value of mean as the threshold for binarization. But this approach has many shortcomings, as it may not take care about the features and objects in the image properly. This is due to the fact that the mean of an image may be disturbed drastically by the addition of noise pixels) or very few numbers of pixels having the intensity close to any of the boundaries of gray scale.

A solution to this limitation is to add the impact of standard deviation to some extent in selecting the threshold value. But the influence of the value of' standard deviation should neither be too small (as it does not make too much change to the value of mean and the problem will remain the same) nor too large (as the change of standard deviation will have too much affect on the threshold value and binarization will ultimately be affected).

3.3.1 Proposed Ni-Black Binarization Scheme

Our algorithm is based on the Niblack's image thresholding[7] method for image binarization. It offers great improvement over original Niblack's method. It does not entirely depend upon image's local statistical characteristics but also considers the global statistics. Our algorithm calculates “k” at runtime for each pixel and thresholding is done using Niblack method. In contrast, Niblack fixes this w-eight value to -0.2. Local mean (mean calculated over a small window) is the average illumination value in the small region, while global mean is overall illumination of the image. So, the normalized difference $m_d(i, j)$ of global and local mean provides information about the illumination difference for each pixel window with respect to globa1 illumination.

$$m_d(i,j) = \frac{(m_g(i,j) - m_l(i,j))}{\max(m_g(i,j), m_l(i,j))}$$

Where $m_g(i,j)$ is the global mean of the entire image and $m_l(i,j)$ is the local mean computed on each window respectively. Obviously, equation provides a reasonable good 'k' factor for thresholding document images. But it fails to adapt changes in images with different contrast stretch i.e. same image with different contrast stretch values will result in different threshold values if equation is used. The use of standard deviation with the above equation can solve this problem. We have made use of the interrelation of global and local characteristics and set the threshold based on the relative change of local and global mean and standard deviation values. The impact of value of standard deviation remains almost the same on different kinds of images having different local illumination and contrast stretch histogram. The formula to compute the value of the weight ‘k’ is given as:

$$k = -0.3 * \frac{(m_g(i,j) * \sigma_g(i,j) - m_l(i,j) * \sigma_l(i,j))}{\max(m_g(i,j) * \sigma_g(i,j), m_l(i,j) * \sigma_l(i,j))}$$

Where $\sigma_g(i,j)$ is the global standard deviation of the entire image and $\sigma_l(i,j)$ is the local standard deviation computed on each window respectively. The formula for 'k' is multiplied with -0.3 so as to keep the value of 'k' in the range of 0.3 and -0.3. This is done to minimize the effect of standard deviation in the Ni-Black's formula for computing threshold value. The algorithm process is shown in below figure 3.9.

The main algorithmic flow to compute the threshold value for binarization is as under:

- Find out the value of mean of the image (global mean - this is computed only once per image).
- Determine the standard deviation value of the image (global standard deviation computed only once per image).
- Set the size of local region in terms of pixel area (generally known as window size).
- Select a local region of appropriate window size across every single pixel in the image one by one.
- For every local region, compute the mean of that region (local mean) and standard deviation of the corresponding region (local standard deviation).

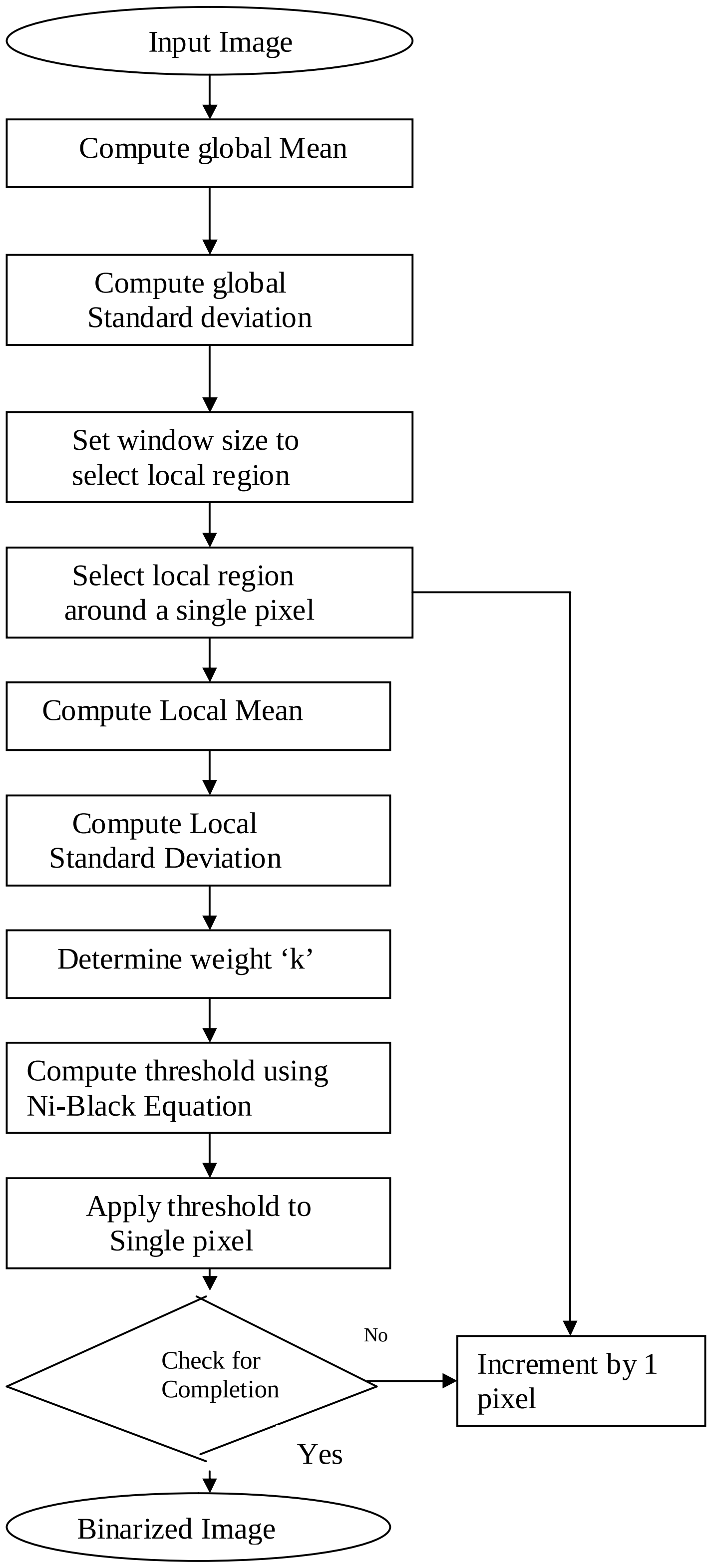

Input Image
Compute global Mean
Compute global Standard deviation
Set window size to select local region
Select local region around a single pixel
Compute Local Mean
Compute Local Standard Deviation
Determine weight ‘k’
Compute threshold using Ni-Black Equation
Apply threshold to Single pixel
Check for Completion
No
Increment by 1 pixel
Yes
Binarized Image

Figure 3.9: Proposed Ni-Black Binarization technique

- Determine the value of weight 'k' for each local window using (2). This is done using local mean and standard deviation and global mean and standard deviation.

- Apply (1) to determine the threshold value using local mean and standard deviation for every local region.
- Apply the threshold value to the single pixel across which the local window is selected.
- Increment window by one pixel and go to step 4 to compute threshold for next pixel.
- Finished when the threshold is computed and applied to all the image area. (The result after this step will be an image having two levels only a binary image).

We propose a new flexible technique of adaptive thresholding for document image binarization. The technique performs efficiently on either high or poor quality of document images. The Laplacian sign image will be applied in the process of adjusting the threshold value. Operator based on the second direction derivative defines the sign image of document image. The basic idea of our technique is to update the threshold value whenever the Laplacian sign of the Input image changes along the raster scanned tine. Normally, the regions, where the sign of pixel has changed, are the edge of components in the image, which presents a physical alteration of the image. The proper threshold value should be adjusted in accordance with changeable component.

3.4 Proposed Laplacian Technique

The proposed technique is depicted by Fig. 3.10. It consists of 3 steps as follows:

Step 1. Calculate the Laplacian sign image[9]. The sign image will be applied in the process of adjusting the threshold value. The trend of the adjusted threshold value depends on the form of modification of the sign in the line. A good feature of the sign image in this research is the smoothing. Laplacian operator is sensitive to noise. The sign image is not smoothing if we use this operator to define the sign image. The Laplacian of an image can be approximated by using the Differential of Gaussian (DOG) algorithm. This is simply obtained by filtering the image with two Gaussian functions, where the ratio of the standard deviations, σ_2/σ_1, are assigned to a value greater than 1.6.

The operation can be described as follows:

1.1 Gaussian filtering the document image by using a binomial filter whose basic kernel is defined by:

$$f(x,y,n) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Binomial filtering is implemented by image convolution with the kernel *n* times. The result is equivalent to Gaussian filtering with the variance σ^2 of *0.5n*.

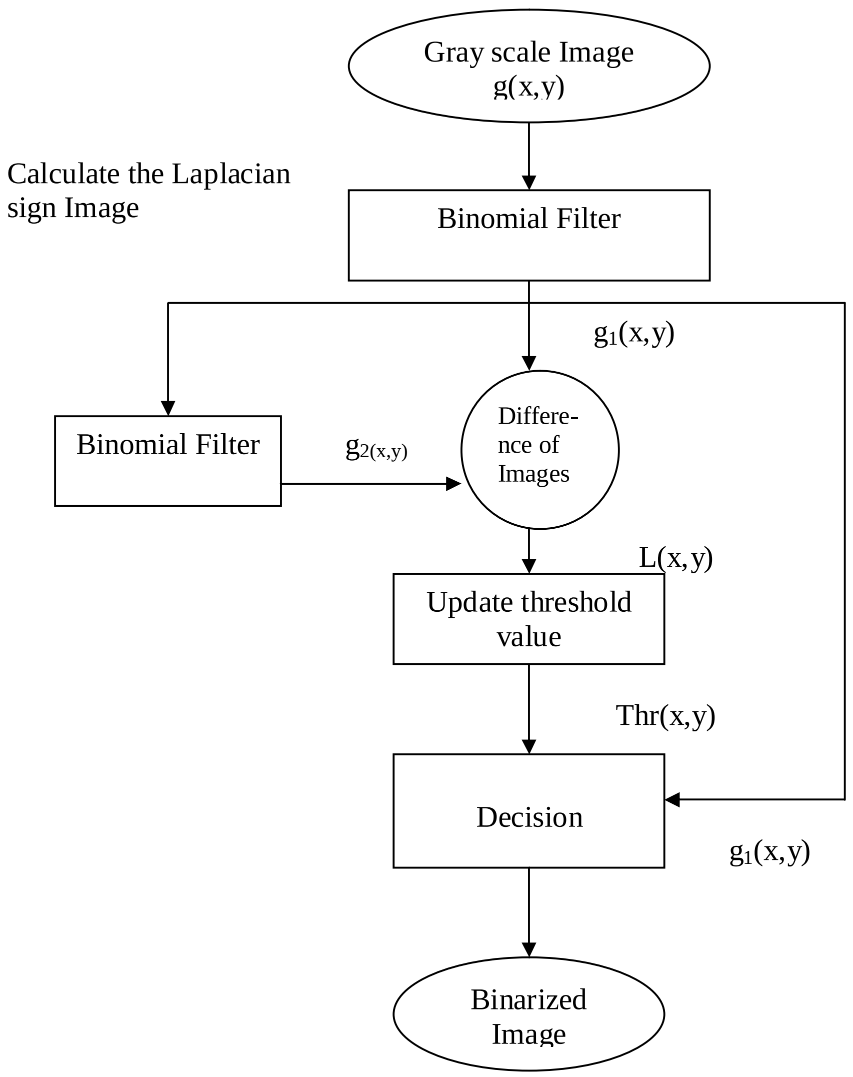


Figure 3.10: Proposed Binarization technique using Laplacian Sign

1.2 Apply the above 3x3 binomial kernel to cascade with gl(x,y) three times in order to adjust bandwidth in accordance with the principle of DOG. The resulting image is g2(x,y).

1.3 Calculate the difference between gl(x,y) and g2(x,y) to define the Laplacian sign image by:

$$L(x,y)=\begin{cases} + \text{ if } g_1(x,y) - g_2(x,y) > 0 \\ - \text{ if } g_1(x,y) - g_2(x,y) < 0 \end{cases}$$

The resulting image will be in the form of plus (+) and minus (-).

Step 2: Update the threshold at each transition the threshold value will be adjusted whenever the sign of L(x,y) changes. Calculating the threshold value can be performed by the following stages:

2.1 We define black and white values for L(x, 0) which is the starting pixel in each line.

$$\text{Black} = 0 \qquad \text{White} = g_2(x,0)$$

2.2 We adjust the black value if the sign of L(x,y) is changed from + to - when comparing with previous pixel in the line and x is unequal to 0.

$$\text{Black}=g_2(x,y)$$

2.3 We adjust the white value if the sign of L(x,y) is changed from - to + when comparing with the previous pixel in the same line and x is unequal to 0 .

$$\text{White}=g_2(x,y)$$

2.4 Threshold value at (x,y) is equal to

$$Thr(x,y) = \alpha * \frac{(White + Black)}{2}$$

where α is a coefficient value used to weight to prevent any error occurring due to the intensity of noise on backgrounds.

Step 3: Perform binarization of the filtered image The final step is to decide whichever the pixel belongs to characters or background. The decision is made on the filtered image gl(x,y) against the threshold Thr(x,y):

$$b(x,y)=\begin{cases} 1 \text{ if } g_1(x,y) < Thr(x,y) \\ 0 \text{ Otherwise} \end{cases}$$

Pixels defined by 1 are the pixels of character or graphic symbol whereas pixels defined by 0 are the pixels of background.

CHAPTER 4

SEGMENTATION & THINNING

4. A SEGMENTATION

Character segmentation [11] is an operation that seeks to decompose an image of a sequence of characters into sub images of individual symbols. It is one of the decision processes in a system for optical character recognition (OCR). Its decision, that a pattern isolated from the image is that of a character (or some other identifiable unit), can be right or wrong. It is wrong sufficiently often to make a major contribution to the error rate of the system.

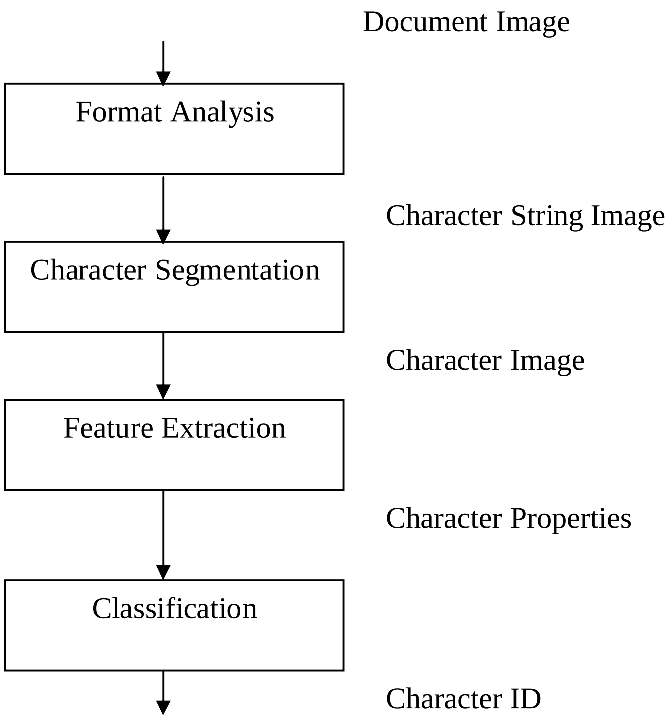


Figure 4.1: Segmentation Analysis

The "classical" approach to OCR, Fig. 4.1, and segmentation is the initial step in a three step procedure:

Given a starting point in a document image:

1) Find the next character image.

2) Extract distinguishing attributes of the character image.

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

This sequence is repeated until no additional character images are found.

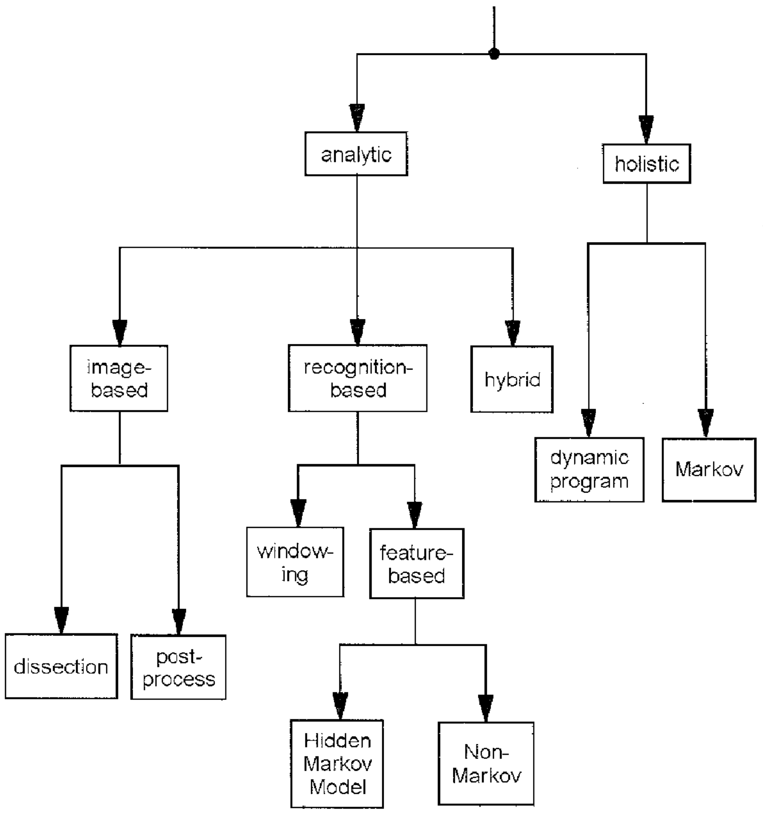


Figure4.2: Hierarchy of Segmentation

The preprocessing stage yields a “clean” document in the sense that a sufficient amount of shape information, high compression, and low noise on a normalized image is obtained. The next stage is segmenting the document into its subcomponents. Segmentation is an important stage because the extent one can reach in separation of words, lines, or characters directly affects the recognition rate of the script. There are two types of segmentation: external segmentation, which is the isolation of various writing units, such as paragraphs, sentences, or words, and internal segmentation, which is the isolation of letters, especially in cursively written words.

4.1 External Segmentation: It is the most critical part of the document analysis, which is a necessary step prior to the off-line CR Although document analysis is a relatively different research area with its own methodologies and techniques, segmenting the document image into text and nontext regions is an integral part of the OCR software. Therefore, one who works in the CR field should have a general overview for document analysis techniques. Page layout analysis is accomplished in two stages: The first stage is the *structural analysis*, which is concerned with the segmentation [12] of the image into blocks of document components (paragraph, row, word, etc.), and the second one is the *functional analysis*, which uses location, size, and various layout rules to label the functional content of document components (title, abstract, etc.).

A number of approaches regard a homogeneous region in a document image as a textured region. Page segmentation is then implemented by finding textured regions in gray-scale or color images. For example, Jain *et al*. use Gabor filtering and mask convolution, the Tang *et al.* approach is based on fractal signature and Doermann's method employs wavelet multistage analysis. Many approaches for page segmentation concentrate on processing background pixels or using the white space in a page to identify homogeneous regions. These techniques include X–Y tree, pixel-based projection profile, connected component-based projection profile, white space tracing, and white space thinning. They can be regarded as top–down approaches, which segment a page, recursively, by X-cut and Y-cut from large components, starting with the whole page to small components, eventually reaching individual characters. On the other hand, there are some bottom–up methods which recursively grow the homogeneous regions from small components based on the processing on pixels and connected components. An example of this approach may be the Docstrum method, which uses k-nearest neighbor clustering. Some techniques combine both top–down and bottom–up techniques.

4.2 Internal Segmentation: Although the methods have developed remarkably in the last decade and a variety of techniques have emerged, segmentation of cursive script into letters is still an unsolved problem. Character segmentation strategies [12] are divided into three categories.

4.2.1 Explicit Segmentation:

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

Dissection [11],[12] is a process that analyzes an image without using a specific class of shape information. The criterion for good segmentation is the agreement of general properties of the segments with those expected for valid characters. Available methods based on the dissection of an image use white space and pitch, vertical projection analysis, connected component analysis, and landmarks. Moreover, explicit segmentation can be subjected to evaluation using linguistic context.

4.2.2 Implicit Segmentation:

This segmentation strategy is based on recognition. It searches the image for components that match predefined classes. Segmentation [12] is performed by the use of recognition confidence, including syntactic or semantic correctness of the overall result. In this approach, two classes of methods can be employed:

1) Methods that make some search process and

2) Methods that segment a feature representation of the image.

The first class attempts to segment words into letters or other units without use of feature-based dissection algorithms. Rather, the image is divided systematically into many overlapping pieces without regard to content. Conceptually, these methods originate from schemes developed for the recognition of machine-printed words. The basic principle is to use a mobile window of variable width to provide sequences of tentative segmentations, which are confirmed by CR. Another technique combines dynamic programming and NNs. Finally, the method of selective attention takes NNs even further in the handling of the segmentation problem.

The second class of methods segments the image implicitly by classification of subsets of spatial features collected from the image as a whole. This approach can be divided into two categories: HMM-based approaches and non-Markov-based approaches. Non markov approaches stem from concepts used in machine vision for recognition of occluded object. This family of recognition-based approaches uses probabilistic relaxation, the concept of regularities and singularities, and backward matching.

4.3 Segmentation using Bounding Box analysis

Let I denote the input binary image. A connected component analysis algorithm is applied to the foreground region of I to produce the set of connected components. Then, for each connected component, it's associated bounding box - the smallest rectangular box which circumscribes the component, - is calculated. A bounding box can be represented by giving the coordinates of the upper left and the lower right corners of the box. The numbers of bounding boxes are always larger than the number of symbols since multiple bounding boxes are produced for multi-component symbols. Our page segmentation [14] scheme analyzes the spatial configuration of those bounding boxes of connected components to extract textlines, words, and paragraphs.

4.3.1 Projection of Bounding boxes:

Analysis of the spatial configuration of bounding boxes[13] can be done by projecting those bounding boxes onto a straight line. Since paper documents are usually written in the horizontal or vertical direction, projections of bounding boxes onto the vertical and horizontal lines are of particular interest. While projecting bounding boxes onto the horizontal or vertical line, they will accumulate onto that line, which results in the projection profile. A projection profile is a frequency distribution of the projected bounding boxes on the projection line. The bounding box projection profiles provide important information about the number of bounding boxes aligned along the projection direction.

4.3.2 Extraction of characters from Word image:

In this step, the algorithm groups the bounding boxes on each textline (produced from the last step) into bounding boxes of words. The algorithm first computes the projection profiles within each of the textline bounding boxes. Next, the algorithm considers each of the projection profiles as a one-dimensional gray-scale image**,** and thresholds each of the images with threshold value 1 to produce a binary image. Note that, during the binarization, a symbol (or a broken symbol) with multiple bounding boxes[14] may be merged into one, as well as, those adjacent symbols within the same textline whose bounding boxes are overlapping with each other. But this will not cause any problem in the result of our word extraction process, since our algorithm extracts words by merging bounding boxes based on the lateral proximity of neighboring boxes.

After such binarization, the algorithm performs a morphological closing operation on each of the binarized textline projection profiles with structuring element of appropriate size. The length of the structuring element is determined by analyzing the distribution of the run-lengths of 0's on the binarized textline projection profile. In general, such a run-length distribution is bi-modal. One mode corresponds to the inter-character spacings within words, and the other to the inter-word spacings. A threshold value can be chosen in the valley between the two dominant histogram modes. Then the characters are extracted from word images using these histograms.

4. B THINNING

The binarized image is thinned to obtain skeleton of the image having the same shape characteristics. The thinning algorithm [15] used is a two-iteration algorithm for binary regions with successive passes applied to contour points of an image using 8 - neighborhood notation. The thinning algorithm [16] iteratively deletes edge points of a region subject to the constraints that deletion of these points does not remove end points, does not break connectivity and does not cause excessive erosion of the region and the result is the skeleton of the region 1-pixel thick with shape characteristics retained.

P9	P2	P3
P8	P1	P4
P7	P6	P5

Figure4.3: Neighborhood arrangement used by the thinning algorithm

4.4 Proposed Thinning Algorithm:

<u>Step 1</u>: flags a contour point p1 for deletion if the following conditions are satisfied:

a) $2 \le N(p1) \le 6$ where $N(p1)$ = number of nonzero neighbours of p1.

b) $T(p1)=1$ where $T(p1)$ = number of 0-1 transitions in the ordered sequence p2-p9.

c) $p2*p4*p6=0$

d) $p4*p6*p8=0$

<u>Step 2</u>: Conditions (a) and (b) remain the same as in step 1 but (c) and (d) are changed to

(c’)$p2*p4*p8=0$

(d’)$p2*p6*p8=0$

If all conditions are satisfied at a point, the point is flagged for deletion. To prevent the structure of the data to change during the execution of the algorithm, the point is not deleted until all border points have been processed.

One iteration of thinning algorithm consists of

1) applying step 1 to flag border points for deletion
2) deleting the flagged points
3) applying step 2 to flag the remaining border points for deletion
4) Deleting the flagged points.

This procedure is applied iteratively until no further points are deleted, at which time the algorithm terminates, yielding the skeleton of the region.

Algorithm for thinning:

```
test_1( )
{
  for all  object pixels
   {
      if(all step 1 conditions satisfied by the pixel)
            flag for deletion.
     }
      delete flagged points
    if(deleted_points !=0)
           test_2( );
     else
       exit( );
    }
test_2( )
{
  for all  object pixels
   {
      if(all step 2 conditions satisfied by the pixel)
            flag for deletion.
     }
      delete flagged points
    if(deleted_points !=0)
           test_1( );
     else
       exit( );
    }
    thin( )
    {
       test1( );
    }
```

CHAPTER 5

FEATURE EXTRACTION

5. FEATURE EXTRACTION

It deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another. The character has specific features which are extracted from the samples. A total of ten features [18] for each character are extracted. They are

5.1 Aspect ratio

This is calculated as Height/Width. This is calculated by first finding the bounding box [19] of the character. Then

Aspect ratio = (height of the bounding box) / (width of the bounding box)

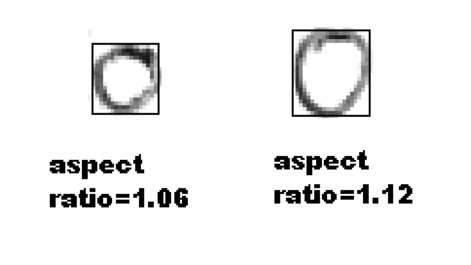


Figure5.1: Demonstrating difference in aspect ratios

Pseudo-code:

```
for all rows from 0 to (rowno-1)
 {
   for the first occurrence of object pixel store the row number and exit
}
store H1; //top bound of the bounding box
for all rows from 0 to (rowno-1)
 {
   for the last occurrence of object pixel store the row number and exit
}
store H2; //bottom bound of the bounding box
for all columns from 0 to (colno-1)
 {
      for the first occurrence of object pixel store the column number and exit
}
store W1;//left bound of the bounding box
```

```
for all columns from  0 to (colno-1)
 {
        for the first occurrence of object pixel store the column number and exit
}
store W2;//right bound of the bounding box

height=difference of H1 and H2;
width=difference of W1 and W2;
aspect ratio=height/width;
```

5.2 Pixel density

This feature is calculated by finding the bounding rows and columns of base rectangle of the word. The number of black pixels in base rectangle is then found. Then pixel density [20] is calculated as ratio of number of object pixels found and the area of the base rectangle. This feature is size invariant and no need to normalize this.

Pseudo-code

```
find the base rectangle of the word;
for this base rectangle region
            {
                get the no of object pixels;
                calculate the area of the region;
                pixel density=(no of object pixels)/(area);
                store the pixel density as feature;
            }
```

5.3 Horizontal stroke

A horizontal stroke [18] is detected when ever a sequence of pixels is detected in a row. The length of the sequence should be greater than some threshold which can be adjusted to any value of convenience. Here we took it to be 3.

Ex:

```
0011000
01111100
```

Consider the 2 sequences. A horizontal stroke is detected in the second case only as the no.of 1’s in order is > 2.

```
for all rows
{
    note the number of object pixels existing one after the other in each row

    if(number>3)
        number of horizontal stroke++;
}
```

5.4 Vertical stroke

A vertical stroke [20] is detected when ever a sequence of pixels is detected in a column. The length of the sequence can be greater than some threshold which is adjusted to any value. Here we took it to be 3.

Ex:

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

Consider the 2 sequences. A vertical stroke is detected in the second case only as the no.of 1’s in order is > 2.

```
for all coloumns
{
    note the number of object pixels existing one after the other in each coloumn
   if(number>2)
       number of vertical stroke++;
}
```

5.5 Right slant stroke

A right slant stroke [19] is detected when ever a sequence of ones which are making the same slope to a reference point is detected to the left of the reference point. We consider that a right slant stroke exists if the length of the sequence is greater than a threshold which is considered to be more than 3.

Ex:

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

Consider the above bit sequences. A single right slant stroke is detected in this case as the no.of 1's in slant position having same slope and having >3 1's is satisfied in one case.

```
for every data pixel
{
   extract the values of possible pixels which are in position to form a right
   hand stroke
{
    if ( pixels extracted = =object pixels)
            pixel count++;
 }
 if (pixel count>3)
     right hand stroke ++;
}
```

5.6 Left slant stroke

A left slant stroke [20] is detected when ever a sequence of ones are making the same slope to a reference point is detected to the right of reference point. We consider that a left slant stroke exists if the length of the stroke is greater than a threshold which is considered to be more than 3.

Ex:

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

Consider the above bit sequences. A single left slant stroke is detected in this case as the no.of 1's in slant position having same slope and having >3 1's is satisfied in one case.

5.7 North - East segment

It is a segment which is convexed towards North East direction. Here whenever a pixel with value '1' is encountered , we consider the array which is to the left top of reference point and calculate the difference of row and column indexes for every '1' encountered. If both the differences are negative then it is a north east segment.

Ex:

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

Three of the possible north east segments are shown above.

```
for every data pixel
{
   for each possibility of array structure
  {
    extract the values of possible pixels which are in position to form a north
    east segment
   {
      if ( pixels extracted = =object pixels)
            pixel count++;
   }
   if (pixel count>3)
      north east segment ++;
 }
}
```

All the following features can be extracted on the basis of above pseudo code just by modifying the pixel positions under consideration

5.8 North - West segment

It is a segment which is convexed towards North west direction. Here whenever a pixel with value ‘1’ is encountered, we consider the array which is to the left bottom of reference point and calculate the difference of row and column indexes for every ‘1’ encountered . If the row difference is positive and column difference is negative it is a west segment.

Ex:

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

Three of the possible North West are as shown above.

5.9 South - East segment

It is a segment which is convexed towards south east direction. Here whenever a pixel with value ‘1’ is encountered , we consider the array which is to the right top of reference point and calculate the difference of row and column indexes for every ‘1’ encountered. If row difference is negative and column difference is positive then it is a south east segment.

Ex:

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

Three of the possible south east segments are as shown above.

5.10 South - West segment

It is a segment which is convexed towards south west direction [18],[19],[20]. Here whenever a pixel with value '1'' encountered, we consider the array which is to the right bottom of reference point and calculate the difference of row and column indexes for every '1' encountered. If both the differences are positive then it is a south west segment.

Ex:

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

Three of the possible south west segments are as shown above. Let us consider an example of feature extraction from alphabets.

CHAPTER 6

PATTERN RECOGNITION: NEURAL NETWORKS

6. PATTERN RECOGNITION

6. A WHAT IS PATTERN RECOGNITION?

The term **Pattern recognition** encompasses a wide range of information processing problems of great practical significance, from speech recognition and the classification of handwritten characters, to fault detection in machinery and medical diagnosis. Pattern recognition is a field within the area of machine learning. Alternatively, it can be defined as "the act of taking in raw data and performing an action based on the category of the data". As such; it is a collection of methods for supervised learning. Pattern recognition aims to classify data (patterns) based on either a priori knowledge or on statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space.

Pattern recognition system [25] consists of two-stage process. The first stage is feature extraction and the second stage is classification. Feature extraction is the measurement of population of entities that will be classified. This assists the classification stage by looking for features that allows fairly easy to distinguish between the different classes. Several different features have to be used for classification. The set of features that are used makes up a feature vector, which represents each member of the population. Then, pattern recognition system classifies each member of the population on the basis of information contained in the information vector.

The classification or description scheme usually uses one of the following approaches: statistical (or decision theoretic), syntactic (or structural). Statistical pattern recognition is based on statistical characterizations of patterns, assuming that the patterns are generated by a probabilistic system. Structural pattern recognition is based on the structural interrelationships of features. A wide range of algorithms can be applied for pattern recognition, from very simple Bayesian classifiers to much more powerful neural networks.

Typical applications are automatic speech recognition, classification of text into several categories (e.g., spam/non-spam email messages), the automatic recognition of handwritten postal codes on postal envelopes, or the automatic recognition of images of human faces. The last two examples from the subtopic image analysis of pattern recognition that deals with digital images as input to pattern recognition systems.

6. B PATTERN RECOGNITION METHODS

6. B.1 BAYESIAN DECISION THEORY:

The Bayesian decision theory [26] is a system that minimizes the classification error. This theory plays the role of a ***priori.*** This is when there is priority information about something that we would like to classify. The decision can be done using Baye's rule. The Baye's formula which states the following:

$$P(w_j/x) = p(x/w_j)\ P(w_j)\ /\ p(x)$$

What the formula means is that using a priori information; we can calculate the *a posteriori* probability of the state of nature being in state wj then we have given that the feature value x has been featured.

The probability that we would probably make an error is any minimum of the two instants at any point, because it represents the smaller probability that we did not pick. The formula for the error is the following:

$$P(error/x) = min[p(w_i/x),p(w_j/x)]$$

6. B.2. K-NEAREST NEIGHBOR RULE:

The Nearest Neighbor (NN) rule [25] is used to classify machine printed and handwritten characters. The distance measured between two character images is needed in order to use this rule. Without a *priori* information about the distributions from which the training examples are drawn, the NN rule achieves very high performance. The rule involves training set of both positive and negative cases. A new sample is classified by calculating the distance to the nearest training case. The following figure 8.3 shows an example of NN rule.

In this example there are two classes: θ_1, which are yellow triangles and θ_2 , which are blue squares. The yellow circle represents the unknown sample X. We can see that the unknown sample's nearest neighbor is from class θ_1, therefore it is labeled as class θ_1.

The distance between two input vectors is calculated as Euclidean distance. The Euclidean distance is defined by:

$$d(x, y_i) = \sqrt{\sum_{j=1}^{k} (x_j - y_{ij})^2}$$

where x_j is the *j*th component of the input vector, and y_{ij} is the *j*th is component of the codeword y_i.

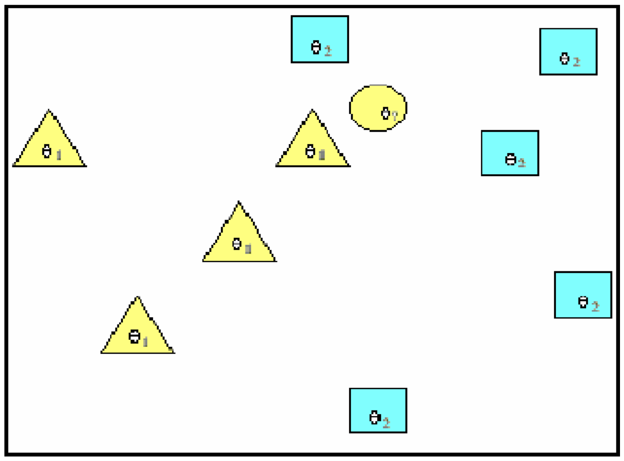


Figure 6.1: The Nearest Neighbor rule

When the amount of pre-classified points is large, it is good to use the majority votes of the nearest *k* neighbors instead of the single nearest neighbor. This method is called the k nearest neighbor (k-NN) rule. The following shows an example of k-NN rule with k value equal to 3:

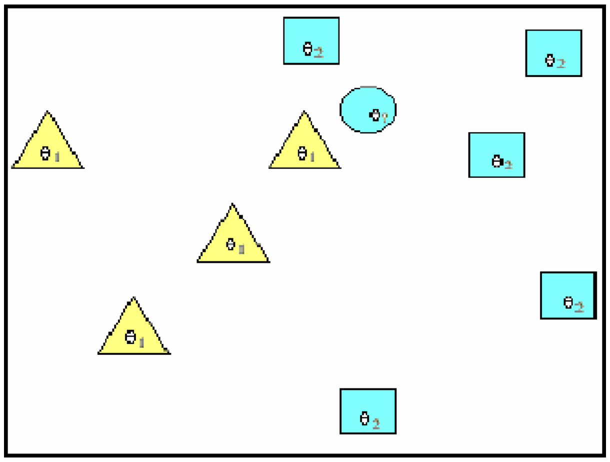


Figure 6.2: The K-Nearest Neighbor rule with K=3

As before, there are 2 classes: θ_1, which are yellow triangles and θ_2, which are blue squares. The blue circle represents the unknown sample X. We see that two of its nearest neighbors are from class θ_2 , so it is labeled as class θ_2.

6. B.3 LINEAR CLASSIFICATION OR DISCRIMINATION:

The goal of linear classification [24] is to assign observations into the classes. This can be used to establish a classifier rule so that it can assign a new observation into a class. In other words, the rule deals with assigning a new point in a vector space to a class separated by a boundary. Linear classification provides a mathematical formula to predict a binary result. This result is a true or false (positive or negative) result or any other pair of characters.

6. C CAMPARISON: ADVANTAGES AND DISADVANTAGES

Now we take a look at the advantages and the disadvantages of Bayesian Decision theory, the Nearest Neighbor rule, and the Linear Classification.

Bayesian Decision method has a conceptual clarity leading to an elegant numerical recipe. The method copes with diverse information, in particular “incomplete” data sets and the use of prior information can help to cope with over-parameterization where classical methods would fail. Bayesian Decision method has computational difficulties. This means that the method has a difficulty filling in the numerical details. The method also has an obligation to use prior information. If this prior information were absent, then the method would not work properly.

Training in Nearest Neighbor rule is very fast. The system is good in learning complex target functions. After training the data, the system is less likely to lose the information that is trained. The down side of Nearest Neighbor rule is that it needs a larger data set and the query time is very slow. It is also very sensitive to the data errors. Therefore if there were any irrelevant information entered into the system, the system would be easily fooled by them.

The advantage of Linear Classification method is that it is well suited to mixed data types. It can also handle non-linear cases and missing data. The system can control objectivity or subjectivity. It can also control the model fit. The results produced by the system are very easy to interpret.

The disadvantage of Linear Classification method is it needs a larger data set and at the same time very sensitive to data errors.

6. 2 NEURAL NETWORKS

6. D WHAT IS A NEURAL NETWORK?

A neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest. To achieve good performance, they employ a massive interconnection of simple computing cells referred to as 'Neurons' or 'processing units'. Hence a neural network viewed as an adaptive machine can be defined as

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network from its environment through a learning process.
2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

Neural networks [30] are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements. Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown below.

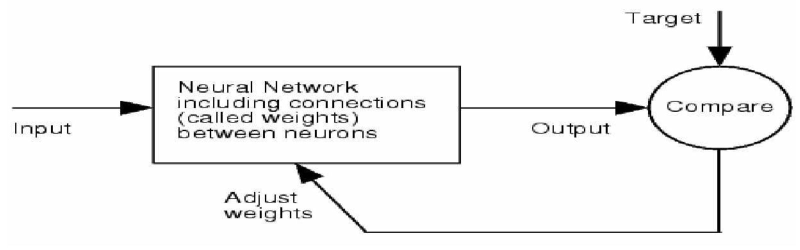

Target
Neural Network including connections (called weights) between neurons
Input
Output
Compare
Adjust weights

Fig 6.3: Neural Network Model

The true power and advantage of neural networks lies in their ability to represent both linear and non-linear relationships and in their ability to learn these relationships directly from the data being modeled. Traditional linear models are simply inadequate when it comes to modeling data that contains non-linear characteristics. Neural networks are designed to work with patterns - they can be classified as pattern classifiers or pattern associators.

6. E WHY USE NEURAL NETWORKS?

It is apparent that a neural network [30] derives its computing power through, first, its massively parallel distributed structure and, second, its ability to learn and therefore generalize. The use of neural networks offers the following useful properties and capabilities:

- Massive parallelism
- Distributed representation and computation
- Learning ability
- Generalization ability
- Input-output mapping
- Adaptivity
- Uniformity of Analysis and Design
- Fault tolerance
- Inherent contextual information processing
- VLSI implement ability.

6. F NEURON MODEL

An artificial neuron is a device with many inputs and many outputs. Each input is multiplied by a corresponding weight, analogous to a synaptic strength, and all the weighted inputs are then summed to determine the activation level of the neuron. These weighted inputs are then added together to produce ***'net'*** output and if they exceed a pre-set threshold value, the neuron fires. The ***'net'*** output produced is further processed by an activation function (***f***) to produce the neuron's output signal. A simple neuron model can be represented as below

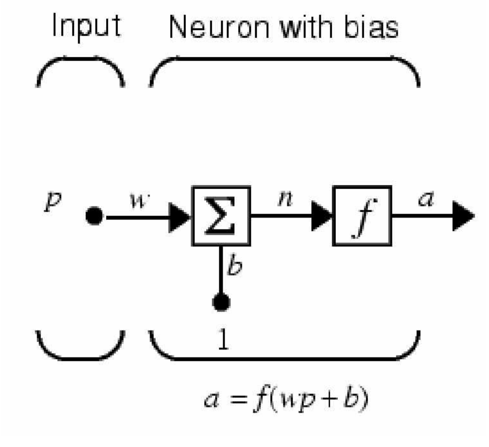


Fig 6.4: Simple Neuron Model

In the above figure p is input of signal w is the weighted input and b is bias input. The block **'∑'** produces the **'net'** output by summing the weighted inputs. The block **'f'** represents the activation function.

6. G NETWORK LAYERS

Although a single neuron can perform certain simple pattern detection functions, the power of neural computation comes from connecting neurons into network layers. These multilayer networks have been proven to have capabilities beyond those of a single layer. These networks are formed by cascading group of single layers; the output of one layer provides the input to the subsequent layer.

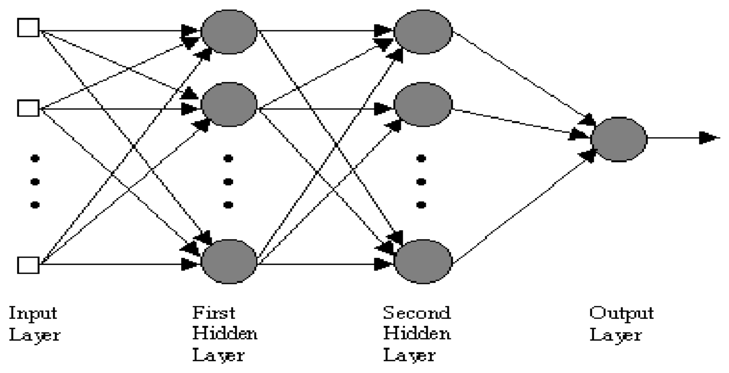


Figure 6.5: A multi-layer neuron model

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "**input**" units is connected to a layer of "**hidden**" units, which is connected to a layer of **"output**" units as in the figure:

- Ø The activity of the input units represents the raw information that is fed into the network.
- Ø The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.
- Ø The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

The hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.

6. H ACTIVATION FUNCTIONS

Activation functions for the hidden units are needed to introduce nonlinearity into the network. Without nonlinearity, hidden units would not make nets more powerful as it is the nonlinearity (i.e, the capability to represent nonlinear functions) that makes multilayer networks so powerful.

There are three types of Activation functions:

- Ø Binary Function - PERCEPTRON
- Ø Sigmoidal Function
- Ø Hyperbolic Tangent Function

6. H.1 SIGMOIDAL FUNCTION

The block **'f'** accepts the **NET** output and produces the signal labeled **OUT**. If the **'f'** processing block compresses the range of **NET**, so that **OUT** never exceeds some limits regardless of the value of **NET**, **'f'** is called a squashing function. The squashing function is often chosen to be the logistic function or "sigmoid". This function is expressed in mathematically as

$$\mathbf{OUT} = \frac{\mathbf{1}}{\mathbf{1 + e^{-net}}}$$

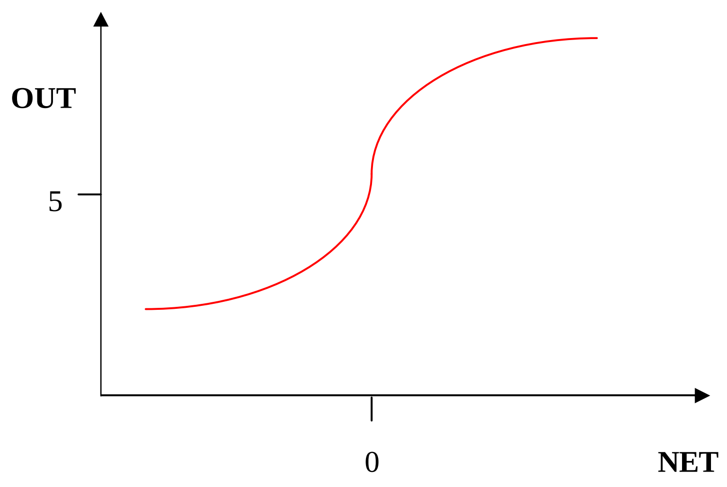


Figure 6.6: Sigmoidal Function

The non-linear gain is calculated by finding the ratio of the change in **OUT** to a small change in **NET**. Thus, gain is the slope of the curve (shown in figure) at a specific excitation level. It varies from a low value at large negative excitations, to a high value at zero excitation, and it drops back as excitation becomes very large and positive. Small signals, while its regions of decreasing gain at positive and negative extremes are appropriate for large excitations. In this way, a neuron performs with appropriate gain over a wide range of input levels.

6. I FEED-FORWARD NEURAL NETWORKS:

Feed-forward ANNs [24] allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. These networks are called non-recurrent networks and they do not require any memory as outputs are directly related to inputs and weights. They are extensively used in pattern recognition. This type of organization is also referred to as bottom-up or top-down. The figure below shows a simple feed forward network:

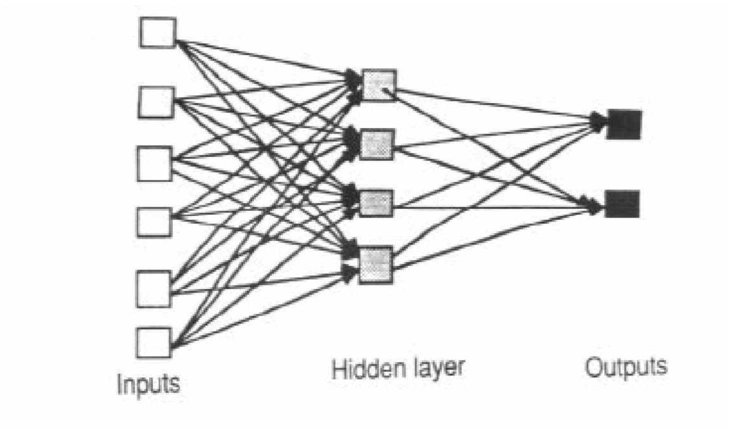


Figure 6.7: An example of Feed Forward Network

6. J LEARNING

Learning is essential to most of these neural network architectures and hence the choice of a learning algorithm is a central issue in network development. Learning implies that a processing unit is capable of changing its input/output behavior as a result of changes in the environment. Since the activation rule is usually fixed when the network is constructed and since the input/output vector cannot be changed, to change the input/output behavior the weights corresponding to that input vector need to be adjusted. In a neural network, learning can be supervised or unsupervised.

6. K BACK PROPOGATION ALGORITHM

For many years, there was no theoretically sound algorithm for training multilayer artificial neural networks. The invention of the back propagation algorithm has played a large part in the resurgence of interest in artificial neural networks. Back propagation is a systematic method for training multilayer artificial neural networks (Perceptrons). The following figure shows the basic model of the neuron used in Back propagation networks.

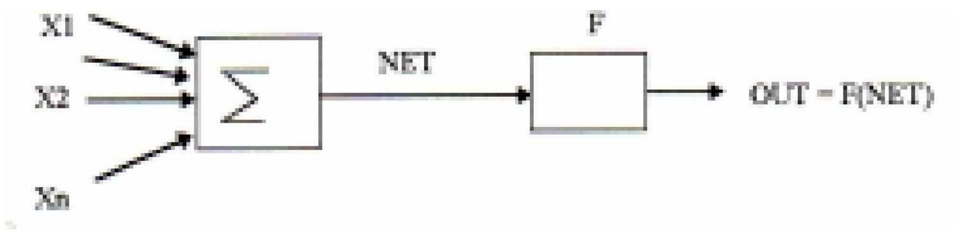


Figure 6.8: Basic model of neuron using back propagation

Each input is multiplied by corresponding weights, analogous to a synaptic strength, and all the weighted inputs are then summed to determine the activation level of the neuron. These summed (NET) signals are further processed by an activation function (F) to produce the neuron's output signal (OUT). In back propagation, the function used for the activation is the logistic function or Sigmoid. This function is expressed mathematically as:

$$F(x) = \frac{1}{1+e^{-x}} \quad \text{thus } OUT = \frac{1}{1+e^{-NET}}$$

The Sigmoid compresses the range of NET so that OUT lies between zero and one. Since the back-propagation uses the derivative of the squashing function, it has to be everywhere differentiable. The Sigmoid has this property and the additional advantage of providing a form of automatic gain control (i.e. if the value of NET is large, the gain is small and if it is small the gain is large).

6. L AN OVERVIEW OF TRAINING IN BACK PROPOGATION

6. L.1 TRAINING ALGORITHM:

The objective of training the network [25] is to adjust the weights so that the application of a set of inputs (input vectors) produces the desired outputs (output vectors). Training a back propagation network involves each input vector being paired with a target vector representing the desired output; together they are called a training pair. The following figure shows the architecture of the multilayer back propagation neural network.

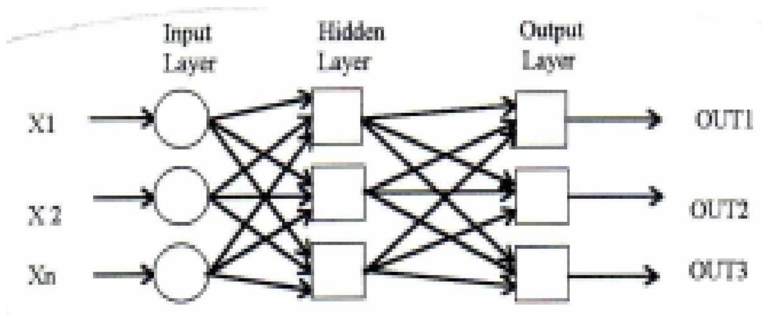


Figure 6.9: Multilayer Back propagation neural network

Before starting the training process, all of the weights are initialised to small random numbers. Training the back propagation network requires the following steps:

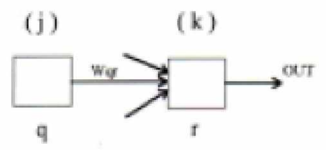
1. Select a training pair (next pair) from the training data set and apply the input vector to the network input.
2. Calculate the output of the network, i.e. to each neuron NET=$\Sigma XiWi$ must be calculated and then the activation function must be applied on the result F (NET).
3. Calculate the error between the network output and the desired output (TARGET – OUT).
4. Adjust the weights of the network in a way that minimises the ERROR (described below).
5. Repeat step 1 through 4 for each vector in the training set until no training pair produces an ERROR larger than a pre-decided acceptance level.

6. L.2 ADJUSTING WEIGHTS OF THE NETWORK

Adjusting weights of the output layer:

Adjusting the weights of the output layer is easier, as a target value is available for each neuron. The following shows the training process for a single weight from neuron “q” in the hidden layer “j” to neuron “r” in the output layer “k”. The output of a neuron in layer “k” is subtracted from its target values to produce an ERROR signal. This is multiplied by the derivative of a squashing function OUT (1-OUT) calculated for that neuron (“r”) thereby producing a “δ” value.

$$\delta = \text{OUT} * (1-\text{OUT}) * (\text{TARGET} - \text{OUT})$$



$$\Delta_{Wqr,k} = \eta \delta_{r,k} \, OUT_{qj} \text{ ------------------------(1)}$$

$$W_{qr,k}(n+1) = W_{qr,k(}n) + \Delta W_{qr,k} \text{ ------------(2)}$$

Figure 6.10: adjusting weights of output layer

Where,

Wqr,j(n) = the value of weight from neuron in the hidden layer “j” to neuron “r” in the output layer “k” at step “n” (before adjustment).

Wr,j(n+1) = the value of weight from neuron in the hidden layer j to neuron “r” in the output layer “k” at step n+1 (after adjustment).

OUTq,j = the value of OUT of neuron in the hidden layer “j”.

ΔWqr,k = amount that Wqr,j to be adjusted.

Adjusting the weights of the hidden layer:

Back propagation trains the hidden layer by propagating the output ERROR back through the network layer by layer, adjusting weights at each layer. The same 2 equations (1) and (2) above are used for all layers, both output & hidden except that, for hidden layers the □values must be generated without the benefit of targets. The following figure explains how this is accomplished.

δs for hidden layer neurons are calculated according to equation (3) by using the δs calculated for output layer (δy,k s) and propagating them backward through the corresponding weights.

$$\delta q,j = OUTq,j\ (1-OUTq,j)(\ \Sigma\ \delta r,kWqr,k) \text{ -----------(3)}$$

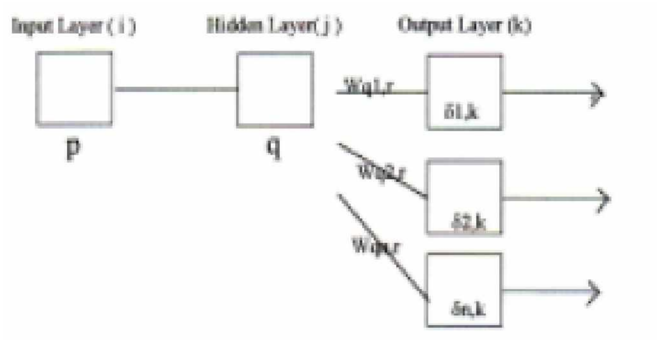


Figure 6.11: adjusting weights of hidden layer

Then, with δs in hand, hidden layer weights can be adjusted similar to the output layer weights as given below

$$\Delta Wqr,j = \eta\delta qj\ OUTqj \text{ -----------------(4)}$$
$$Wqr,j(n+1) = Wqr,j(n) + \Delta Wqr,j \text{ --------------(5)}$$

6. M TESTING:

The 17 different fonts of character set are applied to feature extraction. The features of each character are applied to neural network. If the character is recognized as corresponding character code is 1 and remaining are -1.

7. RESULTS

7.1 Preprocessing

7.A Noise Removal:

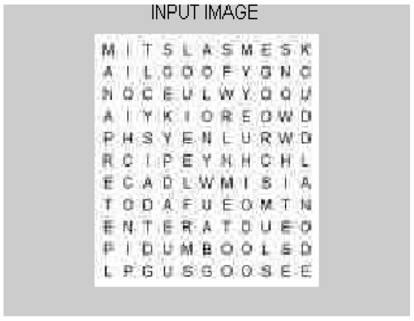


Fig 7.1 Input Image

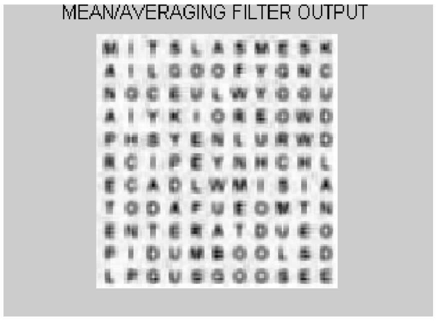


Fig 7.2 Mean/Averaging Filter output

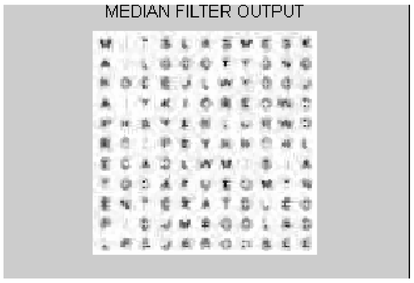


Fig 7.3 Median Filter output

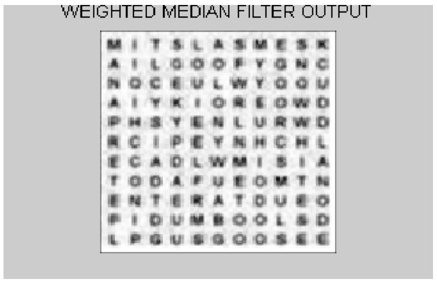


Fig 7.4 Weighted Median Filter output

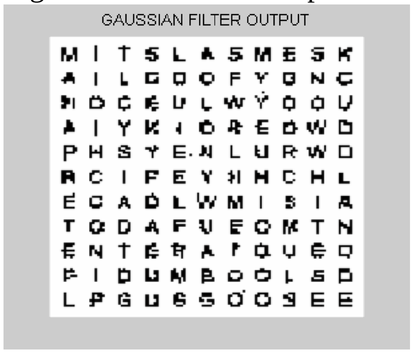


Fig 7.5 Gaussian Filter output

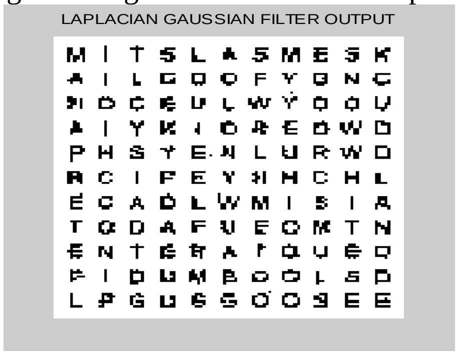


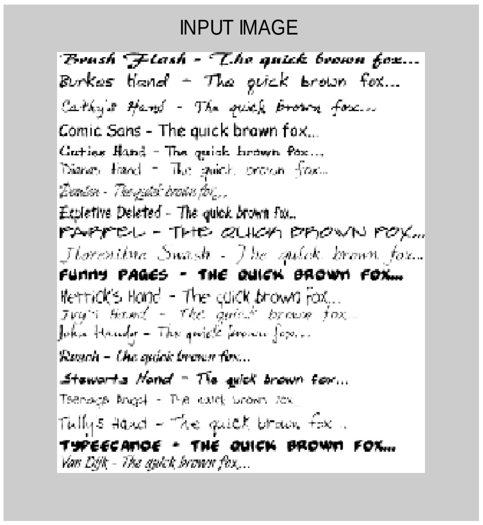
Fig 7.6 Laplacian Gaussian Filter output

In this preprocessing stage, the salt and pepper noise, impulse noise is removed by using smoothing filters like mean/averaging filter, median filtering, weighted median filtering and sharpening filters like Gaussian filtering, Laplacian Gaussian filter. The output images are shown in above figures.

7.B Binarization:

Ni-Black algorithm for both ordinary and noisy images:

Size of window = 31


INPUT IMAGE
Brush Flash - The quick brown fox...
Burkes Hand - The quick brown fox...
Cathy's Hand - The quick brown fox...
Comic Sans - The quick brown fox...
Gutties Hand - The quick brown fox...
Dianes Hand - The quick brown fox...
Dundee - The quick brown fox...
Expletive Deleted - The quick brown fox...
FAFFEL - THE QUICK BROWN FOX...
Florentine Swash - The quick brown fox...
FUNNY PAGES - THE QUICK BROWN FOX...
Herrick's Hand - The quick brown fox...
Jay's Hand - The quick brown fox...
John Handy - The quick brown fox...
Roach - The quick brown fox...
Stewarts Hand - The quick brown fox...
Teenage Angst - The quick brown fox...
Tullys Hand - The quick brown fox...
TYPEECANOE - THE QUICK BROWN FOX...
Van Dijk - The quick brown fox...

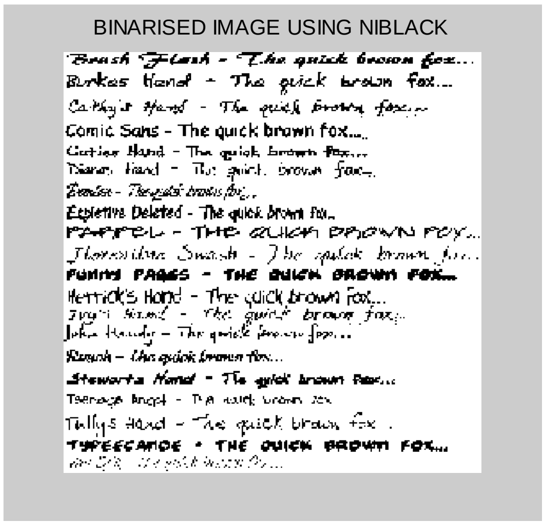
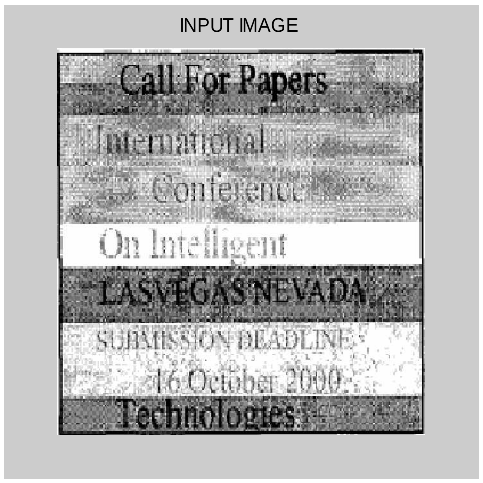


Fig 7.7 Input Image

Fig 7.8 Output Binarized Image

Size of window = 31



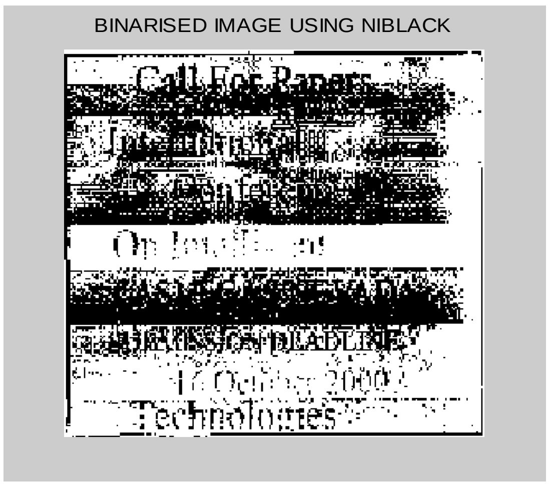
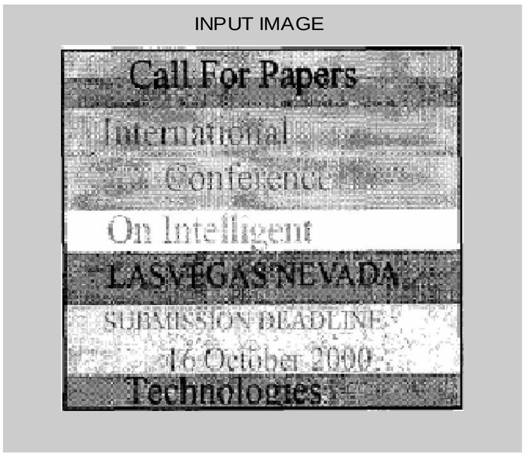


Fig 7.9 Input Noisy Image

Fig 7.10 Output Binarized Image

The proposed binarization technique using Ni-Black method is applied on ordinary image and noisy image, the results are shown in above figures.

7.C Adaptive thresholding algorithm using Laplacian sign results:



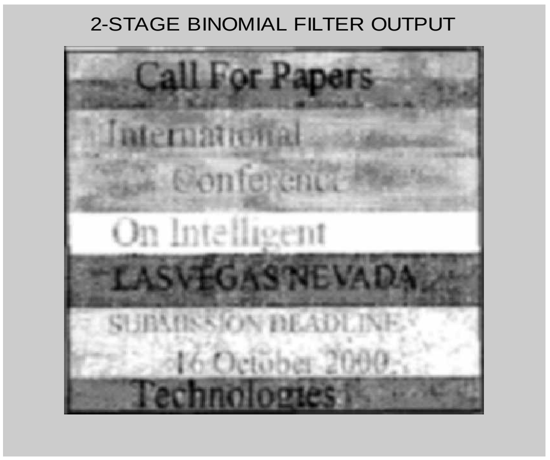
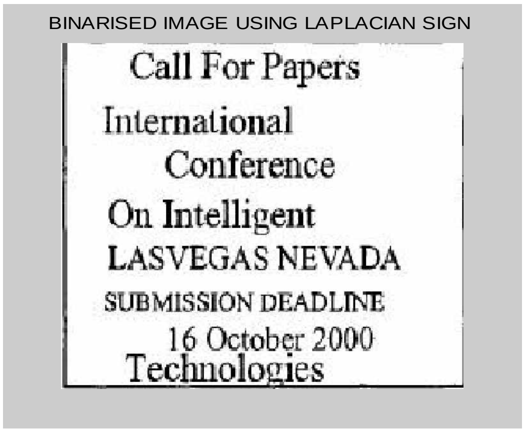


Fig 7.11 Input Noisy Image

Fig 7.12 2-stage Binomial Filter output



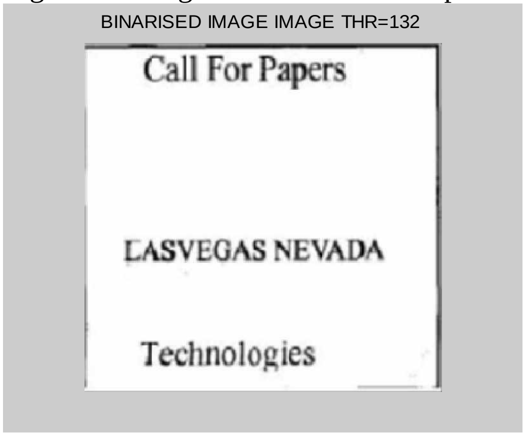


Fig 7.13 Binarized Image using Laplacian sign

Fig 7.14 Output Binarized Image: Threshold 132

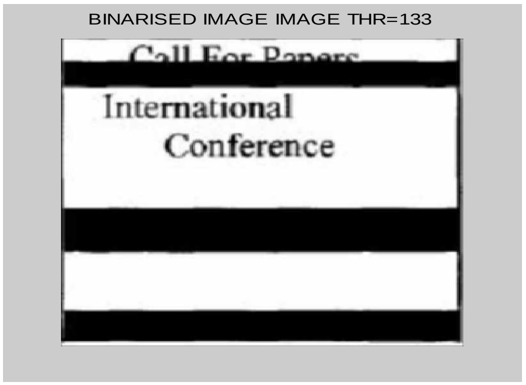


Fig 7.15 Binarized Image: Threshold 133

The proposed adaptive thresholding binarization technique using Laplacian sign is applied to noisy image, the output images of two stage of binomial filter and the final output binarized image. If the threshold is selected as 132 and 133 then the binarized images are shown in above figures.

7.2 SEGMENTATION:

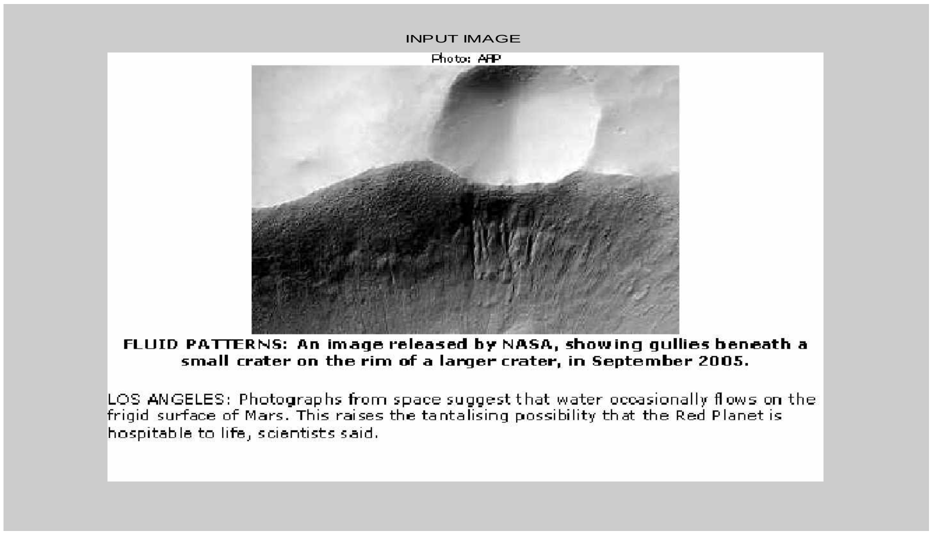

INPUT IMAGE

Photo: AFP

FLUID PATTERNS: An image released by NASA, showing gullies beneath a small crater on the rim of a larger crater, in September 2005.

LOS ANGELES: Photographs from space suggest that water occasionally flows on the frigid surface of Mars. This raises the tantalising possibility that the Red Planet is hospitable to life, scientists said.

Fig 7.16 Input Document Image

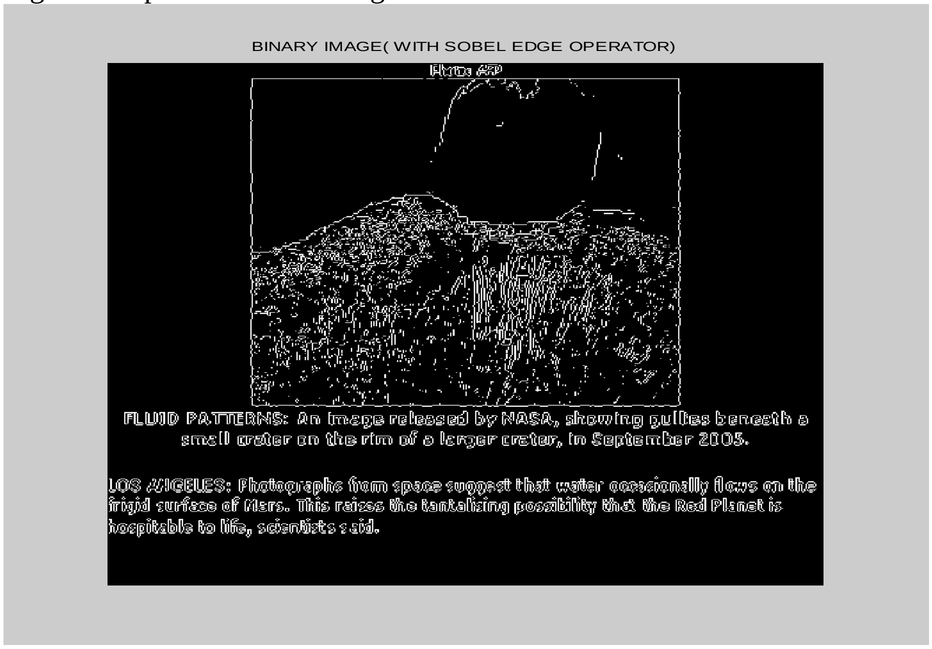

BINARY IMAGE(WITH SOBEL EDGE OPERATOR)

Photo: AFP

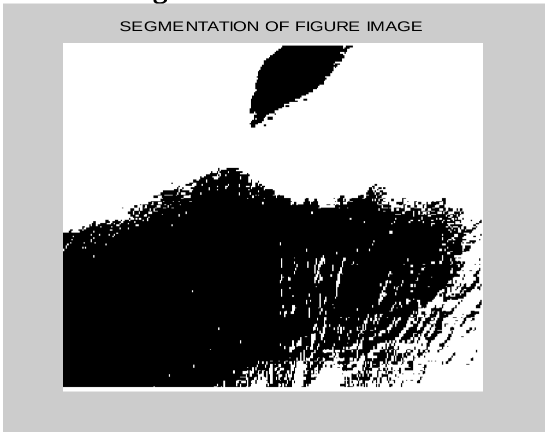
FLUID PATTERNS: An image released by NASA, showing gullies beneath a small crater on the rim of a larger crater, in September 2005.

LOS ANGELES: Photographs from space suggest that water occasionally flows on the frigid surface of Mars. This raises the tantalising possibility that the Red Planet is hospitable to life, scientists said.

Fig 7.17 Binarized image using sobel edge operator

The segmentation of document image is critical task in character recognition. The first section of this image is converted to binarized image using sobel edge operator is shown above figure.

External Segmentation:



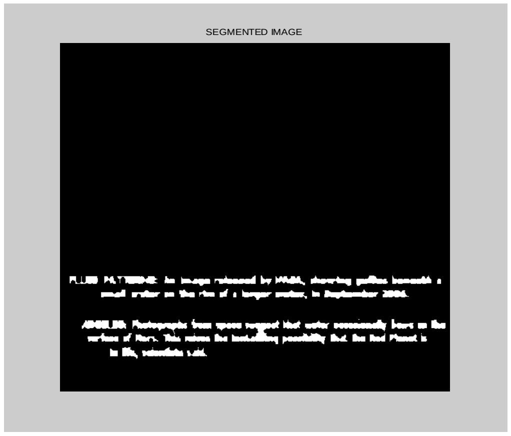
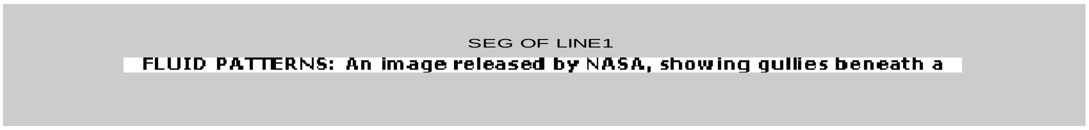


Fig 7.18 Segmentation of graphical Image   Fig 7.19 Segmentation Text Image

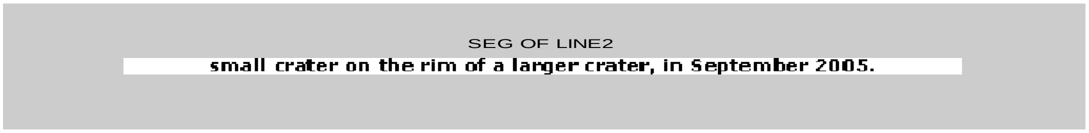
The document image is segmented into graphical image and text image using dissection method and the results of external segmentation are shown in above figure.

Internal Segmentation:



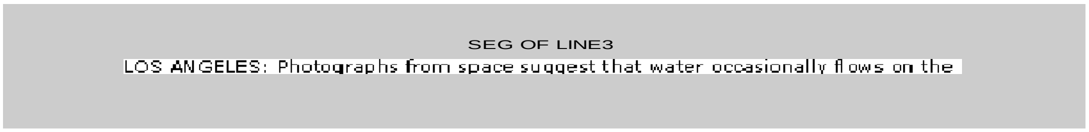
SEG OF LINE1

FLUID PATTERNS: An image released by NASA, showing gullies beneath a



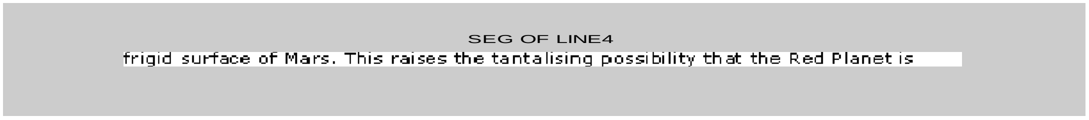
SEG OF LINE2

small crater on the rim of a larger crater, in September 2005.



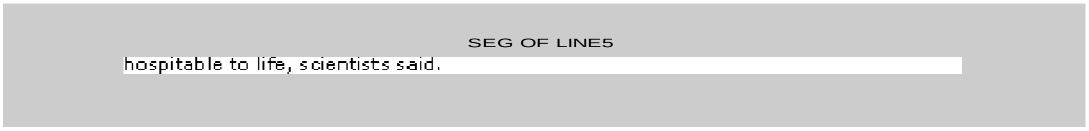
SEG OF LINE3

LOS ANGELES: Photographs from space suggest that water occasionally flows on the



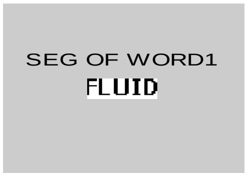
SEG OF LINE4

frigid surface of Mars. This raises the tantalising possibility that the Red Planet is



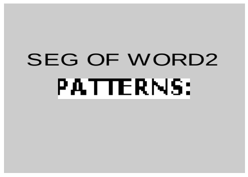
SEG OF LINE5

hospitable to life, scientists said.



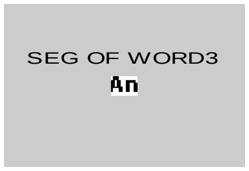
SEG OF WORD1

FLUID



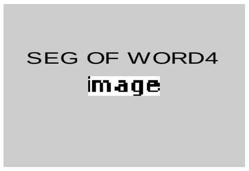
SEG OF WORD2

PATTERNS:



SEG OF WORD3

An



SEG OF WORD4

image

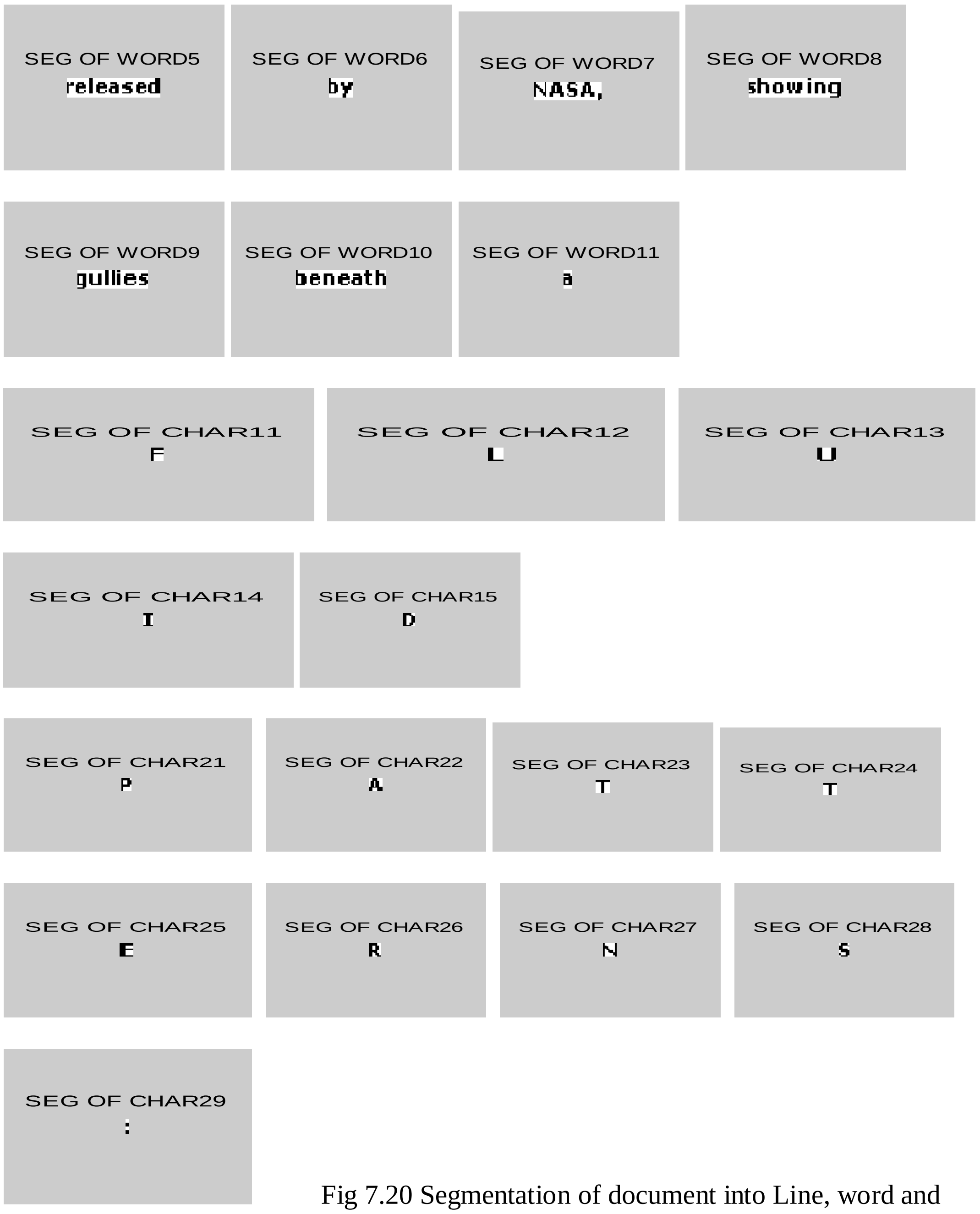


Fig 7.20 Segmentation of document into Line, word and character segments

The text image is converted into line segment image, word segment image and character segment image using dissection method. The results of internal segmentation are shown in above figures.

7.2.1 SEGMENTATION USING BOUNDING BOX ANALYSIS:

The handwritten word image is segmented using bounding box analysis. First the word images is converted into binary image and then calculate connecting components of word image are calculated using horizontal and vertical histograms using projection profile analysis, the rectangular boxes are equal to the number of connecting components. The results are shown in below.

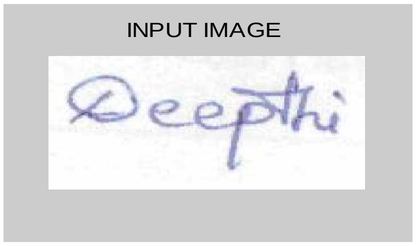


Fig 7.21 Input Image

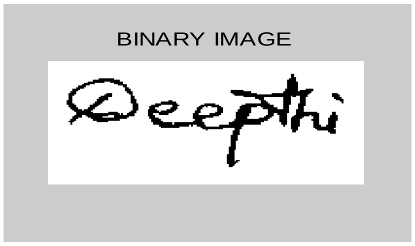


Fig 7.22 Binary Image

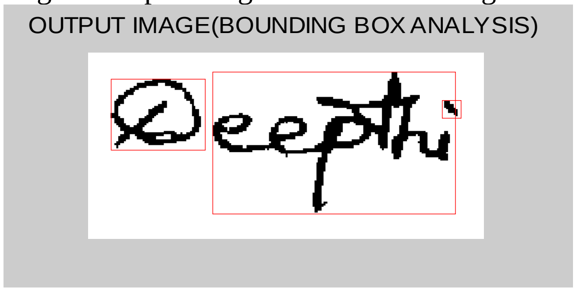


Fig 7.23 Output Image using Bounding Box analysis

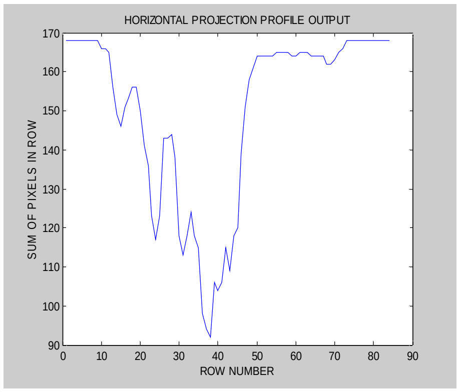


Fig 7.24 Horizontal Projection Profile

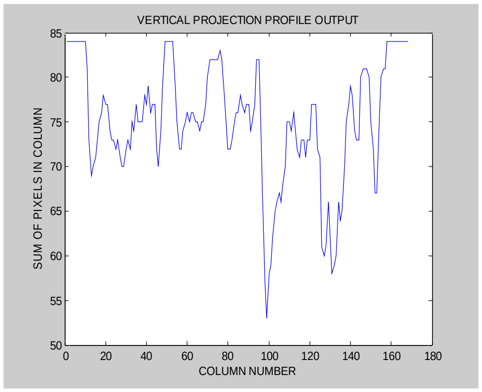
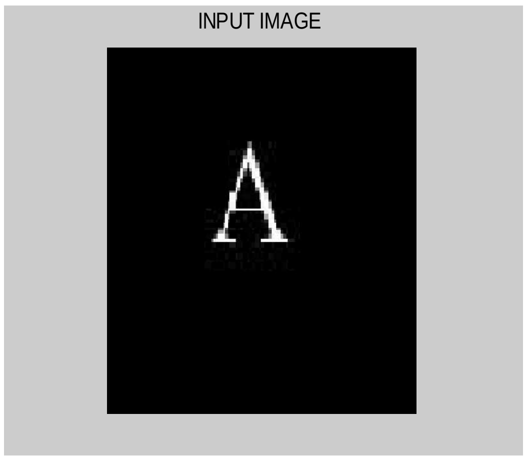
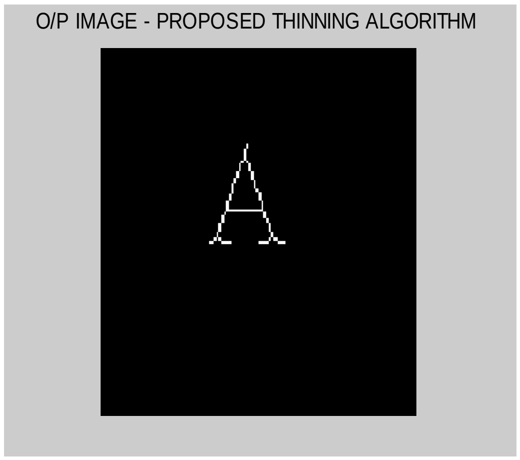


Fig 7.25 Vertical Projection Profile

7.3 Proposed Thinning algorithm results:





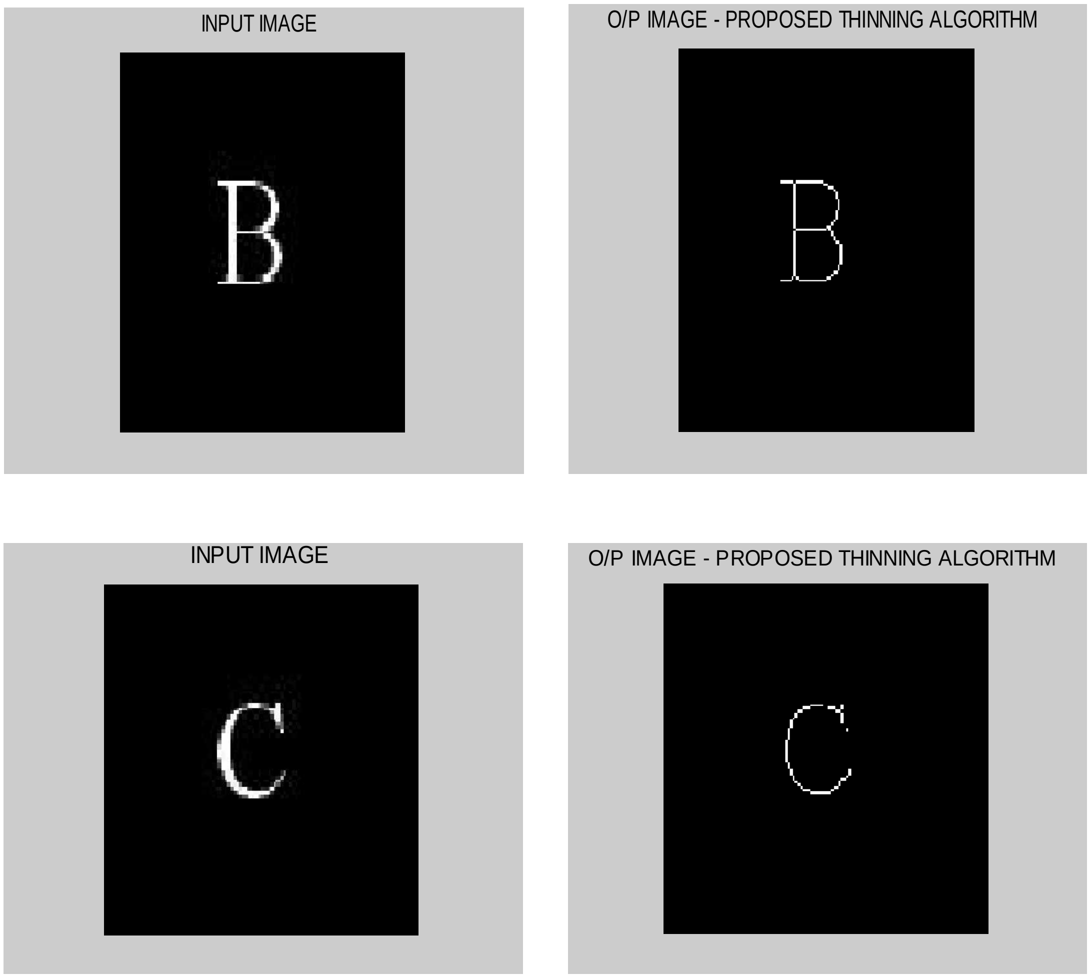

INPUT IMAGE
O/P IMAGE - PROPOSED THINNING ALGORITHM
INPUT IMAGE
O/P IMAGE - PROPOSED THINNING ALGORITHM

Fig 7.26 Input characters A, B, C and Thinned Images A, B, C.

Pixel values in Processing of Images:

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

Fig 7.27: 20 X 20 Pixel data after Binarization

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

Fig 7.28: 20 X 20 Pixel data after Thinning

The thinned characters are obtained by applying segmented characters from document images the values of pixels in binarized form are shown in above.

7.4 Feature Extraction:

The features Aspect ration, Pixel density, Horizontal stroke, Vertical stroke, Right slant stroke, Left slant stroke, North-East segment, North-West segment, South-East segment and South-West segment of characters A-Z, a-z and numerals 0-9 for single font are represented as shown in below table 7.1.

Feature/ Char.	Aspect Ratio	Pixel Density	Horizontal Stroke	Vertical Stroke	Right Slant Stroke	Left Slant Stroke	North East Seg	South East Seg	North West Seg	South West Seg
A	0.9355	0.0037	0.0045	0.0044	0.0077	0.3871	0.0108	0.0102	0.0081	0.0049
B	0.2681	0.0039	0.0029	0.0053	0.0077	0.4409	0.0083	0.0123	0.0081	0.0056
C	0.0053	0.0065	0.0030	0.0053	0.7742	0.0075	0.0109	0.0128	0.0079	0.6452
D	0.0050	0.0043	0.0033	0.0044	0.3723	0.0068	0.0111	0.0159	0.0078	0.4032
E	0.0084	0.0044	0.0032	0.7742	0.0041	0.0075	0.0112	0.0159	0.6563	0.0048
F	0.0058	0.0046	0.0029	0.2742	0.0046	0.0083	0.0167	0.0128	0.3036	0.0049
G	0.0061	0.0046	0.8065	0.0058	0.0045	0.0085	0.0167	0.7500	0.0065	0.0063
H	0.0061	0.0044	0.4465	0.0056	0.0052	0.0082	0.0112	0.4676	0.0056	0.0055
I	0.0057	0.8065	0.0037	0.0054	0.0058	0.0082	1.2500	0.0058	0.0078	0.0058
J	0.0057	0.4077	0.0031	0.0066	0.0045	0.0085	0.4500	0.0063	0.0068	0.0057
K	0.7742	0.0039	0.0059	0.0067	0.0044	0.5758	0.0040	0.0085	0.0075	0.0056
L	0.5161	0.0034	0.0036	0.0068	0.0058	0.4944	0.0034	0.0065	0.0070	0.0058
M	0.0030	0.0057	0.0036	0.0068	0.8333	0.0040	0.0057	0.0065	0.0067	0.6774
N	0.0031	0.0042	0.0050	0.0067	0.5167	0.0038	0.0042	0.0074	0.0071	0.4270
O	0.0040	0.0043	0.0050	0.8605	0.0048	0.0051	0.0043	0.0075	0.8261	0.0044

P	0.0034	0.0049	0.0035	0.3484	0.0049	0.0043	0.0047	0.0066	0.4805	0.0044
Q	0.0035	0.0049	0.9355	0.0047	0.0073	0.0044	0.0047	0.3333	0.0053	0.0061
R	0.0038	0.0043	0.3259	0.0041	0.0055	0.0045	0.0043	0.4967	0.0060	0.0049
S	0.0039	0.1290	0.0042	0.0087	0.0056	0.0045	0.7500	0.0084	0.0059	0.0051
T	0.0035	1.0000	0.0041	0.0051	0.0055	0.0045	0.4931	0.0074	0.0068	0.0052
U	0.8710	0.0108	0.0067	0.0052	0.0055	0.5806	0.0060	0.0096	0.0075	0.0053
V	0.3011	0.0083	0.0046	0.0057	0.0057	0.4409	0.0051	0.0089	0.0061	0.0051
W	0.0048	0.0109	0.0048	0.0057	0.6129	0.0052	0.0073	0.0089	0.0060	0.6452
X	0.0049	0.0111	0.0050	0.0052	0.4465	0.0044	0.0064	0.0095	0.0075	0.2710
Y	0.0065	0.0112	0.0049	0.9355	0.0048	0.0070	0.0065	0.0095	0.3548	0.0073
Z	0.0054	0.0167	0.0048	0.2603	0.0043	0.0055	0.0076	0.0089	0.4282	0.0073
a	0.0057	0.0167	0.7742	0.0057	0.0071	0.0056	0.0076	0.7826	0.0090	0.0061
b	0.0058	0.0112	0.3938	0.0049	0.0051	0.0067	0.0065	0.5000	0.0074	0.0093
c	0.0059	0.5806	0.0040	0.0111	0.0053	0.0067	0.8333	0.0062	0.0100	0.0103
d	0.0056	0.3100	0.0040	0.0058	0.0061	0.0056	0.4354	0.0053	0.0098	0.0080
e	0.8387	0.0074	0.0042	0.0063	0.0062	1.0000	0.0059	0.0105	0.0101	0.0075
f	0.4194	0.0065	0.0045	0.0063	0.0053	1.0000	0.0056	0.0066	0.0090	0.0104
g	0.0035	0.0137	0.0047	0.0059	0.7917	0.0833	0.0087	0.0067	0.0092	0.6452
h	0.0034	0.0079	0.0052	0.0059	0.3816	0.0833	0.0064	0.0076	0.0102	0.4565
i	0.0051	0.0081	0.0052	1.3226	0.0071	0.0909	0.0067	0.0076	0.6774	0.0043
j	0.0040	0.0081	0.0047	0.3155	0.0071	0.1111	0.0069	0.0068	0.3902	0.0043
k	0.0041	0.0081	0.8750	0.0034	0.0091	0.1250	0.0069	0.9130	0.0046	0.0057
l	0.0045	0.0082	0.3549	0.0028	0.0079	0.1667	0.0067	0.3271	0.0050	0.0048
m	0.0045	0.7742	0.0039	0.0071	0.0083	0.1667	0.5758	0.0085	0.0056	0.0050
n	0.0040	0.3575	0.0038	0.0035	0.0086	0.1250	0.4322	0.0073	0.0057	0.0050
o	0.7742	0.0049	0.0067	0.0037	0.0087	1.0000	0.0046	0.0135	0.0063	0.0051
p	0.4731	0.0043	0.0041	0.0037	0.0083	1.0000	0.0041	0.0089	0.0050	0.0050
q	0.0031	0.0053	0.0042	0.0036	0.6129	0.0833	0.0046	0.0096	0.0049	0.6452
r	0.0034	0.0052	0.0048	0.0036	0.4584	0.0833	0.0049	0.0096	0.0063	0.4371
s	0.0040	0.0056	0.0048	0.8710	0.0046	0.0909	0.0051	0.0091	0.6452	0.0045
t	0.0038	0.0062	0.0043	0.3011	0.0042	0.1111	0.0059	0.0090	0.3597	0.0045
u	0.0038	0.0063	0.8387	0.0051	0.0074	0.1250	0.0059	1.2609	0.0055	0.0063
v	0.0043	0.0054	0.4355	0.0048	0.0049	0.1667	0.0051	0.4228	0.0056	0.0050
w	0.0043	0.6452	0.0034	0.0059	0.0050	0.1667	0.5758	0.0048	0.0078	0.0052
x	0.0038	0.3032	0.0033	0.0053	0.0053	0.1250	0.4306	0.0040	0.0062	0.0053
y	0.6774	0.0064	0.0039	0.0058	0.0053	0.5806	0.0047	0.0094	0.0065	0.0053
z	0.3810	0.0060	0.0037	0.0059	0.0050	0.4140	0.0042	0.0051	0.0066	0.0053
1	0.0046	0.0108	0.0037	0.0056	0.8333	0.0056	0.0055	0.0053	0.0066	0.6452
2	0.0047	0.0072	0.0046	0.0056	0.5042	0.0049	0.0049	0.0053	0.0065	0.4145
3	0.0042	0.0072	0.0047	0.8387	0.0048	0.0065	0.0050	0.0051	0.6774	0.0049
4	0.0054	0.0093	0.0037	0.2320	0.0051	0.0058	0.0053	0.0051	0.3886	0.0044
5	0.0055	0.0093	0.8065	0.0070	0.0069	0.0061	0.0053	0.8696	0.0048	0.0068
6	0.0064	0.0072	0.3819	0.0062	0.0055	0.0074	0.0050	0.3957	0.0045	0.0053
7	0.0065	0.9355	0.0040	0.0085	0.0056	0.0074	0.4583	0.0070	0.0065	0.0055
8	0.0055	0.5139	0.0043	0.0073	0.0059	0.0060	0.4394	0.0068	0.0054	0.0056
9	0.9032	0.0028	0.0063	0.0080	0.0060	0.1290	0.0114	0.0080	0.0056	0.0056
0	0.3652	0.0023	0.0043	0.0080	0.0056	1.0000	0.0095	0.0074	0.0050	0.0054

Table7.1: Features of Characters A-Z, a-z and numerals 0-9.

7.5 Classification:

The 17 different fonts such as ARIAL, BAUHAUS93, BELL MT, BMOS, BODONI MT, BROADWAY, BRUSH SCRIPT MT, CENTAUR, CENTURY, IMPACT, LUCIDA FAX, SYSTEM, TAHOMA, TIMES NEW ROMAN, VERDENA, VIVALDI, VRINDA fonts of characters A-Z, a-z and numerals 0-9 are tested using KNN classifier. The K specifies the number of features are selecting from feature vector. The 5-NN classifier chooses first 5 features of each character and tested for all characters of different fonts. The 10-NN classifier chooses all 10 features of each character and tested for all characters of different fonts. The results are shown in below table 7.2. The over all classification rate for 5-NN classifier is 59.86 and the over all classification rate for 10-NN classifier is 89.65.

7.5.1 K-NN classifier:

Character	5-NN classifier	Classification rate	10-NN classifier	Classification rate
A	11	64.7	15	88.23
B	10	58.82	16	94.12
C	10	58.82	15	88.23
D	12	70.58	14	82.35
E	9	52.94	13	76.47
F	10	58.82	16	94.12
G	9	52.94	14	82.35
H	11	64.7	16	94.12
I	12	70.58	16	94.12
J	12	70.58	15	88.23
K	9	52.94	14	82.35
L	10	58.82	15	88.23
M	11	64.7	16	94.12
N	11	64.7	16	94.12
O	10	58.82	15	88.23
P	9	52.94	14	82.35
Q	10	58.82	16	94.12
R	11	64.7	16	94.12
S	9	52.94	14	82.35
T	12	70.58	16	94.12
U	9	52.94	14	82.35
V	10	58.82	15	88.23
W	10	58.82	15	88.23
X	9	52.94	14	82.35
Y	10	58.82	15	88.23
Z	11	64.7	16	94.12
a	10	58.82	14	82.35
b	9	52.94	15	88.23
c	10	58.82	15	88.23
d	9	52.94	14	82.35
e	10	58.82	15	88.23

f	11	64.7	16	94.12
g	10	58.82	16	94.12
h	9	52.94	14	82.35
i	10	58.82	15	88.23
j	11	64.7	15	99.23
k	10	58.82	16	94.12
l	12	70.58	16	94.12
m	9	52.94	14	82.35
n	9	52.94	15	88.23
o	10	58.82	15	88.23
p	11	64.7	16	94.12
q	10	58.82	15	88.23
r	9	52.94	14	82.35
s	10	58.82	16	94.12
t	11	64.7	15	88.23
u	10	58.82	15	88.23
v	9	52.94	14	82.35
w	10	58.82	15	88.23
x	11	64.7	16	94.12
y	12	70.58	16	94.12
z	9	52.94	14	82.35
1	10	58.82	16	94.12
2	11	64.7	17	100
3	10	58.82	16	94.12
4	11	64.7	17	100
5	10	58.82	16	94.12
6	9	52.94	15	88.23
7	10	58.82	16	94.12
8	10	58.82	16	94.12
9	11	64.7	17	100
0	11	64.7	17	100
Overall classification rate		59.86		89.65

Table 7.2 KNN classification

7.5.2 MLP CLASSIFIER:

The MLP network 10-100-62 is designed for each character with learning rate 0.1 and gamma is 1. The network is trained with backpropogation algorithm to number epochs are 500. The performance in terms of MSE plots for different characters A and B are shown in below figs.

For character ‘A’:

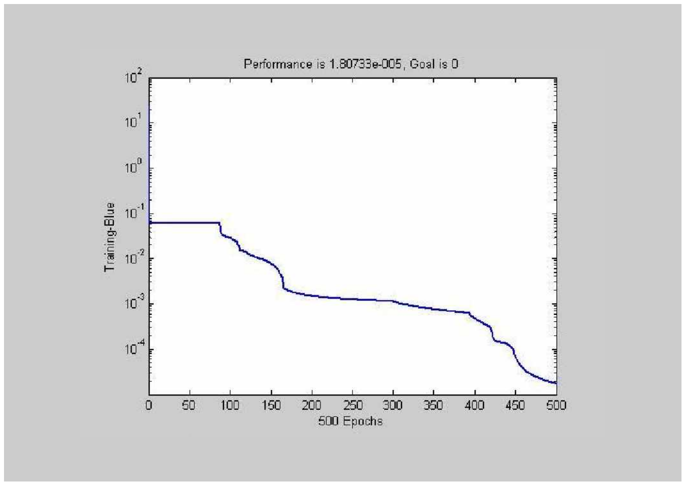


Fig 7.30 MSE plot for character ‘A’ with learning rate 0.1 and gamma is 1.

For character ‘B’:

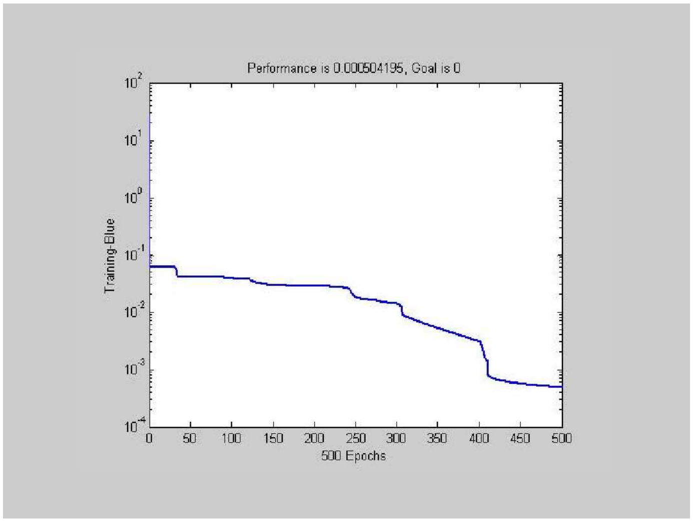


Fig 7.31 MSE plot for character ‘B’ with learning rate 0.1 and gamma is 1.

7.5.3 MLP Network trained to ‘A’ & Tested for A-Z:

The MLP 10-40-26 network is tested for different characters A-Z, a-z and numerals 0-9 the corresponding output is 1 and 25 number of -1’s for character ‘A’ and is different for others because the network was trained with ‘A”.

7.5.4 MLP network is trained with ‘A’ & tested for different fonts of ‘A’:

The MLP 10-40-26 network is tested for different fonts of character A, the corresponding output is 1 and 25 number of -1’s for all, because the network was trained with ‘A”.

Input Character	**Recognized Character**	**Input Character**	**Recognized Character**	**Input Number**	**Recognized Number**
A	A	**a**	a	**1**	1
B	B	**b**	b	**2**	2
C	C	**c**	c	**3**	3
D	D	**d**	d	**4**	4
E	E	**e**	e	**5**	5
F	F	**f**	f	**6**	6
G	G	**g**	g	**7**	7
H	H	**h**	h	**8**	8
I	I	**i**	i	**9**	9
J	J	**j**	j	**0**	0
K	K	**k**	k		
L	L	**l**	l		
M	M	**m**	m		
N	N	**n**	n		
O	O	**o**	o		
P	P	**p**	p		
Q	Q	**q**	q		
R	R	**r**	r		
S	S	**s**	s		
T	T	**t**	t		
U	U	**u**	u		
V	V	**v**	v		
W	W	**w**	w		
X	X	**x**	x		
Y	Y	**y**	y		
Z	Z	**z**	z		

Table 7.3: Recognition of characters A-Z, a-z and numbers 0-9.

8. CONCLUSION

In this thesis, we have proposed algorithm for binarization that works very well for generic document images. The results showed that it has improved document image binarization significantly over well-known Ni-Black's algorithm. The proposed technique is based on Laplacian sign of the image, showed a good performance on different types of document, particularly on images that have a variety of intensities of characters and backgrounds or have low contrast of intensities.

The entire decomposition process is based on the analysis of the spatial configuration of bounding boxes of connected components. Our proposed approach, connected components become the lowest level of the document hierarchy. The proposed technique for thinning algorithm is extensive of segmented characters. The features of each character like aspect ratio, pixel density, vertical stroke, horizontal stroke, left slant stroke, right slant stroke, North-East segment, North-West segment, South-East segment and South-West segment are extracted using proposed technique. These features are classified with statistical method KNN classifiers with less accurate. The MLP design and simulation experiments have been performed on single size and different fonts of characters. Testing has been conducted on 17 different fonts of character set and it has been found that classifier for Machined printed English character recognition has very high success rate.

Scope of Future Work:

An extension to character recognition can also be done using Radial Basis Function networks and the centers are selected using K means clustering algorithm. There is scope for extension to word recognition, sentence recognition and finally document recognition.

9. BIBLIOGRAPHY

[1] Rangachar Kasturi, Lawrence O'Gorman and Venu Govindaraju, "Document image analysis: A primer", Sadhana Vol. 27, Part 1, February 2002, pp. 3–22.

[2] Hasan Al-Rashaideh, "Preprocessing phase for Arabic Word Handwritten Recognition", INFORMATION TRANSMISSIONS IN COMPUTER NETWORKS, Tom 6, No 1, 2006, comp. 11 - 19.

[3] Dipti Deodhare, NNR Ranga Suri R. Amit, "Preprocessing and Image Enhancement Algorithms for from based Intelligent Character Recognition system", International Journal of Computer Science & Applications, 2005 Vol.2, No.2, pp. 131 – 144.

[4] Oivind Due Trier and Anil K.Jain, "Goal-Directed Evaluation of Binarization Methods", IEEE Transactions on Pattern analysis and Machine Intelligence, Vol.17, No.12, Dec 1995.

[5] Robert M. Haralick and D.R.K. Brownrigg, "The weighted Median Filter", Image processing and computer vision, Vol. 27, No.8, August 1984.

[6] P.K. Sahoo, S. Soltani, A.K.C.Wong, and Y.C. Chen, "A Survey of Thresholding Techniques", Computer Vision, Graphics and Image Processing, Vol.41, 1988, pp.223 – 235.

[7] Naveed Bin Rais , M. Shehzad Hanif and rmtiaz A. Taj, "Adaptive Thresholding Technique for Document Image Anaysis", IEEE Trans. on Systems, Man and Cybernetics, 8, 60-66, 1978.

[8] N. Oisu. "**A** threshold selection method from grey level histograms." IEEE Transactions on Systems, Man, and Cybernetics, 9:62 – 88, 1979.

[9] Sittisak Rodtook and Yuttapong Rangsanseri, "Adaptive Thresholding of Document Images on Laplacian Sign", IEEE Transactions on Pattern analysis and Machine Intelligence, Vol.17, No.24, Janauary,2001.

[10] B. Sankur and M. Sezgin, “Image thresholding techniques: A survey over categories”, Pattern Recognition, 2001.

[11] Nafiz Arica and Fatos T. Yarman-Vural, “Correspondence An Overview of Character Recognition focused on Off-Line Handwriting”, IEEE Transactions on Systems, Man, and Cybernetics – part C: Applications and Reviews, VOl.31, No.2, May 2001.

[12] Richard G. Casey and Eric Lecolinet, “A Survey methods and Strategies in Character Segmentation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 18, No.7, July 1996.

[13] Jaekyu Ha & Robert M. Haralick Ihsin, T. Phillips, “Document Page Decomposition by the Bounding-Box Projection Technique”, IEEE Transactions on Systems, Man, and Cybernetics, Vol.18, No.1, pp.1118-1122, January 1995.

[14] J.Ha,I.T. Phillips and R.M. Haralick, “Document Image Decomposition using Bounding Boxes of Connected Components”, ISL Report, Dept. Electrical Eng., University of Washington,1994.

[15] P. S. P. Wang and Y. Y. Zhang, “A fast and flexible thinning algorithm“, IEEE Transactions on Computers, Vol.38, No.5, May 1989.

[16] E.L. Flores, “A Fast thinning algorithm”, Proc. Of the SBT/IEEE International Telecommunications Symposium, vol.2, pp.594-599, Aug.1998.

[17] Oivind Due Trier, Anil K.Jain and Torfinn Taxt, “Feature Extraction methods for Character recognition – A Survey”, Pattern Recognition, vol. 29, No. 4, pp.641 – 662, 1996.

[18] Ahmad T. Al-Taani, “ An Efficient Feature Extraction Algorithm for the Recognition of Handwritten Arabic Digits”, International Journal of computational Intelligence vol. 2, Nov. 2005 ISSN 1304 – 4508

[19] E.Kavallieratos, N.Antoniades, N.Fakotakis and G.Kokkinakis, “Extraction and Recognition of Handwritten Alphanumeric characters from Application forms”, Pattern Recognition, vol.21, No.4, 1997.

[20] Parvati Iyer, Abhipista Singh and Dr. S.Sanyal, “Optical Character Recognition System for Noisy Images in Devanagari Script”, UDL workshop on Optical Character Recognition with workflow and Document Summarization-2005.

[21] Xuewen Wang, Xiaoqing Ding and Changsong Liu, “Optimized Gabor filter based Feature Extraction for Character Recognition”, IEEE Trans. On PAMI vol. 15, No. 10,2000,pp.222-226.

[22] Brijesh K. Verma, “Handwritten Hindi Character Recognition Using Multilayer Perceptron and RBF Neural Networks”, IEEE Trans. On Neural Networks, vol.1, pp. 2111-2115.

[23] A. Rajawelu, M.T. Husavi, and M.V. Shirvakar, “A neural network approach to character recognition”, IEEE Trans. On Neural Networks, vol. 2, pp.387-393, 1989.

[24] D.Y. Lee, “Handwritten digit recognition using K nearest neighbor, radial basis function and backpropogation neural networks”, Neural Computation, vol. 3, 440-449.

[25] O.Batsaikhan and Y.P. Singh, ”Mongolian Character Recognition using Multilayer Perceptron”, Proceedings of the 9th International Conference on Neural Information Processing, vol. 2,2002.

[26] Rejean Plamondon and Sargur N. Srihari, “On-line and Off-line Handwriting Recognition: A Comprehensive Survey”, IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 22, No.1, Jan. 2000.

[27] Gonzalez and Woods, Digital Image Processing, 2nd Edition, Prentice Hall, 2002.

[28] Gonzalez, Woods and Eddins, Digital Image Processing using MATLAB, Prentice Hall, 2004.

[29] Richard O.Duda, Peter E.Hart and David G.Stork, Pattern Classification, 2nd Edition.

[30] Simon Haykin, Neural networks: A Comprehensive Foundation(2nd Edi, Pearson Education,2001).

[31] www.mathworks.com

[32] www.wikipedia.org