A Novel Two-Stage Approach for Translation, Rotation and Scale Invariant Character Recognition

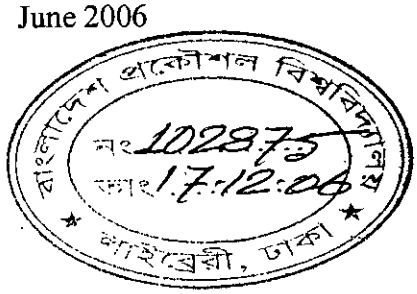
By

Md. Mashud Hyder

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

June 2006

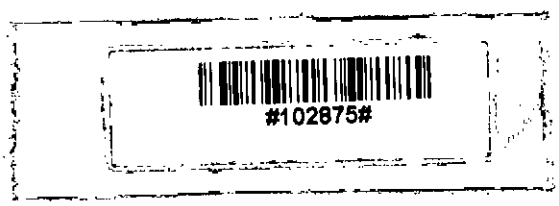


Submitted to

Bangladesh University of Engineering and Technology

In partial fulfillment of the requirements for

M. Sc. Engg. (Computer Science and Engineering) degree.



The thesis titled "**A Novel Two-Stage Approach for Translation, Rotation and Scale Invariant Character Recognition**" submitted by **Md. Mashud Hyder**, Roll No. 040305003p, Session April 2003 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science and Engineering in Computer Science and Engineering (M.Sc. Engg. in CSE) held on June 25, 2006.

BOARD OF EXAMINERS

1. M. M. Islam — Chairman

Dr. Md. Monirul Islam
(Supervisor)
Associate Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000, Bangladesh.

2. Masroor Ali — Member (Ex-officio)

Dr. Muhammad Masroor Ali
Professor and Head
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000, Bangladesh.

3. Member

Dr. Abu Syed Md. Latiful Hoque
Associate Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000, Bangladesh.

4. Member

Dr. Md. Mostofa Akbar
Assistant Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000, Bangladesh

5. Member (External)

Dr. Mohammad Zahidur Rahman
Associate Professor
Department of Computer Science and Engineering
Jahangirnagar University, Savar, Dhaka

CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

M. M. Islam 25.06.2006

Signature of the Supervisor

Dr. Md. Monirul Islam

Signature of the Candidate 25.6.2006

Md. Mashud Hyder

To

My Beloved Parents

&

Dear Brother

Without whom any achievement in life would be meaningless

ACKNOWLEDGEMENT

At first, I want to thank almighty Allah. All praises go to Allah. Almighty gave me enough hardworking capability and persistence, which made me to complete the thesis work.

I performed the thesis work under the supervision of my respect teacher Dr. Md. Monirul Islam. I am grateful to him for his constant guidance, helpful suggestions and valuable assistance. He helped me by giving his invaluable time throughout the thesis work. His encouragement and motivations helped me to overcome all difficulties.

I pay my gratitude to Dr. Muhammad Masroor Ali, Dr. Md. Abul Kashem Mia, Dr. Abu Syed Md. Latiful Hoque, Dr. Md. Saidur Rahman and Dr. Md. Mostofa Akbar for their positive suggestions. These helped me a lot to achieve perfection.

I also want to thank Dr. Md. Shahid Ullah, Head and Dr. Md. Bashir Uddin, Professorex-Department of CSE, Dhaka University of Engineering and Technology (DUET), Gazipur, for giving me enough flexibilities and lab facilities to complete my thesis work. I am also grateful to Mr. Mohammad Iqbal Bin Shahid, my colleague, for sharing thoughts and ideas.

ABSTRACT

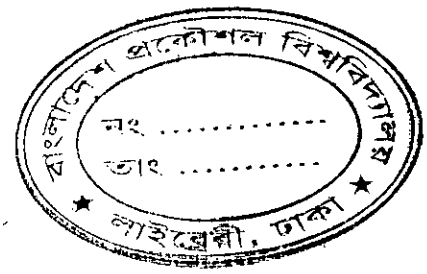
This paper presents a new invariant character recognition (ICR) algorithm called as Symmetry Axis based Feature Extraction and Recognition (SAFER) for recognizing translation, rotation and scale invariant optical characters. Unlike most previous ICR algorithms, SAFER puts emphasis on both simplicity and generalization ability for recognizing invariant characters. SAFER uses the axis of symmetry of character and a simple radial coding to extract invariant features from character. The use of ANN as a classifier in SAFER increases its generalization ability. SAFER has been tested on two well-known fonts Arial and Tahoma that are widely used for representing English characters. These fonts have been subject of many studies in ICR. The experimental results show that SAFER can produce high recognition rate with good generalization ability.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1



Introduction

1.1 Introduction

Automatic recognition of optical character from text image has been an ongoing research problem for several years. Since the mid 1950's, Optical Character Recognition (OCR) has been a very active field for research and development. Now OCR began as a field of research in artificial intelligence and machine vision. It has been gaining more interest lately due to the increasing popularity of computers and electronic typewriter. This seems to be a more natural way of entering data to the computers.

Today, OCR is the most cost-effective and speedy method available. However, Invariant Character Recognition (ICR) is still problematic. The aim of ICR is to identify a character independently of its position (translated or rotated) and size (larger or smaller), has been the object of an intense and thorough study. An increasing number of research groups have proposed a great variety of ICR methods in the last several years.

Most of the good ICR software have a self-learning kind of system referred to as a neural network, which automatically updates the recognition database for new patterns. ICR basically extends the usefullness of keyboard for the purpose of document processing to scanning devices. Although the processes involved in recognising ICR are not so developed, accuracy levels in most cases are not less than 90%. Often to achieve these high recognition rates several read engines are used within the software and each is given elective voting rights to determine the true reading of characters.

Today's ICR engines add the multiple algorithms of neural network technology to analyze the stroke edge [1], the line of discontinuity [2] between the text characters, and the background. Allowing for irregularities of printed ink on paper, each algorithm averages the light and dark along the side of a stroke, matches it to known characters and makes a best guess as to which character it is.

Advances are trying to make ICR more reliable; expect a minimum of 90% accuracy for average-quality documents and greater accuracy for clean copy. However, analysis and ICR of 'difficult' heritage documents are not straightforward [3] and new techniques are required for such material. Some issues that need to be dealt with in character recognition of heritage documents are:

i) Characters are different sized, translated and rotated.

ii) Characters are needed to extract from poor papers, which often results in high occurrence of noise in the digitized images, or fragmented (broken) characters.

The basic processes involved in character recognition [4] can be divided in three stages. First the document that needs to be digitized is scanned. Scanned documents are generally stored as image file either as Tagged Image File Format (TIFF) or as Bitmap file (BMP). In second stage, the scanned documents are processed since scanned images are not accurate and they have noises and rough edges. If the document is not scanned properly, then the scanned image will be skewed. These are to be corrected, before continuing the process.

In third stage, features are extracted from the candidate image. Finally the features are compared with the stored result of a complete alphabetical set. If there is a match of a reasonable level or more of the features with any of the stored alphabet features, then it is assumed that the character is recognized.

Quality of any ICR algorithm is important. There are three criteria to determine the quality of an algorithm. These are accuracy, processing capability, and complexity.

- The algorithm should achieve an improved recognition rate and high efficiency when processing the documents and it prevents unnecessary losses.
- The algorithm should achieve a fully automatic information flow. Which means, with a minimum of manual interaction, the ICR algorithm can automatically finds and extract information from any documents, interprets the data and transports it into any computer system.
- Computationally less complex so that it takes lower time and less disk space.

In this thesis a new ICR algorithm is proposed which satisfies the criteria of any good ICR algorithm.

1.2 Historical Survey

OCR is one of the most successful applications of automatic pattern recognition. In the last several years, an increasing number of research groups have proposed a great variety of ICR methods. These methods are classified in four groups:

- Invariant Moment Based method
- Projection Based Method
- Boundary Based Method
- Neural Network Based Method

As the vast number of papers is published on ICR every year, it is impossible to include all the available feature extraction methods in this survey. Instead, here some representative sections are represented to illustrate the different principles.

1.2.1 Invariant Moment Based Method

Moment based ICR is a popular invariant-recognition scheme for its invariant functions. In the moments, all the pixels of a pattern are used in the computation. Using moment it is reasonable to suspect where the discrimination information lies and how they can be extracted. It is found that the higher order complex moments are powerful predictors of discrimination performance.

In 1962, Hu [5], introducing nonlinear combinations of regular moments, derived a set of seven composed moments with translation, scaling, and rotation-invariant properties. However, the moments proposed by Hu do not possess orthogonal properties, making reconstruction of the input image computationally difficult.

The kernel function of geometric moments is not orthogonal, which makes it computationally expensive to reconstruct an image from the moments. Therefore, geometric moments suffer from a high degree of information redundancy, and they are sensitive to noise for higher-order moments [6–8]. Moreover, the kernel function of geometric moments of order (n+m), often involves power of n and m.

Orthogonal moments have been proven to be more robust in the presence of noise. They are able to achieve a near-zero value of redundancy measure in a set of moment

functions where the moments correspond to independent characteristics of the image [6,8]. For example, Orthogonal Zernike moment functions are defined using a polar coordinate representation of the image space and they are commonly used in recognition tasks requiring rotation invariance images.

Teague [9] suggests orthogonal moments based on the general properties of orthogonal polynomials. In general, it has been shown by Teague [9] that in terms of information redundancy, orthogonal moments (Legendre, Zernike, and pseudo-Zernike) perform better than any other type of moments. In terms of overall performance, Zernike and pseudo-Zernike moments outperform the others [10].

C. Kan and M. D. Srinath [11] show a comparative study of the use of orthogonal moments for invariant classification of alphanumeric characters of different size. In addition to the Zernike and pseudo-Zernike moments (ZMs and PZMs), a new method of combining Orthogonal Fourier-Mellin moments (OFMMs) with centroid bounding circle scaling is introduced, which is shown to be useful in characterizing images with large variability. It is shown that OFMMs give the best overall performance in terms of both image reconstruction and classification accuracy.

C. Chong, et al. [12] derive a translation invariant model of Zernike moments. By applying this framework, translation invariant functions of Zernike moments are derived algebraically from the corresponding central moments. The functions are developed for non-symmetrical as well as symmetrical invariant character images. They mitigate the zero-value obtained for odd-order moments of the symmetrical images. The proposed method eliminates the requirement of transform invariance by providing a translation invariance property in a Zernike feature set.

D. Shen and Horace H.S. Ip [13] present a set of wavelet moment invariants, together with a discriminative feature selection method, for the classification of seemingly similar objects with subtle differences. These invariant features are selected automatically based on the discrimination measures defined for the invariant features. Using a minimum-distance classifier, the wavelet moment invariants achieved better classification rate compared with Zernike's moment invariants and Li's moment invariants.

1.2.2 Projection Based Method

Many algorithms for image analysis have been proposed by both neural computing and statistics communities, most of which are based on a projection of the data onto a two or three dimensional visualization space. The classical linear projection method is used for mapping the data to a lower dimensional space and preserves as much data variance as possible. However, circular projection is essential to extract rotation invariant feature from character.

A. FARES, et al. [14] had recognized rotated patterns by using a ring detector. In this paper an approach similar to the wedge-ring detector is proposed, but instead of using the amplitude of the Fourier transform of a function in the diffraction sampling plan they exploit its phase. Thus, the rotation invariance problem can be solved and the discrimination between similar targets is very much improved. It is shown that the phase only of the Fourier transform of input pattern is more effective and efficient for evaluation and discrimination of rotated target.

E. Kavallieratou et al. [15] published the slant estimation algorithm for OCR. The slant estimation is based on a combination of the projection profile technique and the Wigner-Ville distribution. Moreover, it uses a simple and first sharing transformation technique. The proposed approach is character independent and can easily be adapted in order to satisfy the requirement of most of the OCR system.

M. Choi and W. Kim [16] proposed an algorithm for a rotation invariant template matching method based on the combination of the projection method and Zernike moment. The algorithm consists of two stages. At first, candidate pixels are selected by a coarse searching method. For the coarse search, the vector sum of circular projections of sub-image was taken. To accelerate the candidate selection process, frequency domain implementation of convolution was adopted. Finally, Zernike moments are calculated for the candidate to verify the exact location of the matched templates.

1.2.3 Boundary Based Method

Many image processing and computer vision tasks need to locate objects included in complex scenes where the presence of other objects, noise or the deformable nature of

the object hinders the process. In that case, the only available information is its two-dimensional (2D) shape and few physical features.

M. Gonzalez- Linares et al. [17] proposed an algorithm for the automatic detection and location of a two dimensional objects. This method is based on shape information that is extracted from the edges gradient and only needs a template of object to be located. A new Generalized Haugh Transform is proposed to automatically locate rigid objects in the presence of noise, occlusion and clustering. A Bayesian scheme uses this rigid objects location algorithm to obtain the information of the object

T. Bernier and J. Landry [18] demonstrate a method of shape representation for planar closed curves, which is invariant to translation, rotation and scale. The curvature information was transformed to frequency domain by using the concept of Zahn and Roskies [19]. The points of high curvature (vertices) were found by performing a first order derivative of their polar representation. This process provides sensitivity to contour segments lying between vertices without inordinate sensitivity to noise. In addition, this method can iteratively adjust until an equal number of vertices are found, thus significantly broadening its comparative ability.

F. Ullah and S. Kaneko [20] present a new method for rotation invariant template matching in gray scale images. It is the basis on the utilization of gradient information in the form of orientation codes as the feature for approximating the rotation angle as well as for matching. Orientation codes-based matching is robust for searching objects in cluttered environments even in the cases of illumination fluctuation resulting from shadowing or highlighting, etc. The method consists of two stages. In first stage, histograms of orientation codes are employed for approximation the rotation angle of the object and in second stage; the template object is rotated by the estimated angle for matching.

Boundary-based analysis using discrete Fourier transforms has been proposed as an alternative to ICR [21], [22]. Algorithms based on this kind of analysis are called Fourier descriptors and basically, invariance is obtained by normalizing the frequency representation of the image shape.

1.2.4 Neural Network Based Method

Multilayer feedforward neural networks have been used extensively in OCR. Each node sees a window in the previous layer and combines the low level features in this window into a higher-level feature. Then the neural network can be viewed as a pure classifier, constructing some completed decision boundaries, or it can be view as extracting "super features" in the combined process of feature extraction and classification.

Many papers using neural networks are reported on studies of invariant pattern recognition. Madaline structures for translation-invariant recognition [23], the self-organized recognition [24], and high-order neural networks [25-27] are examples of ICR neural-based methods. Also Yuceer and Oflazer [28], Fukushim [29], Hussain and Kabuka [30], Khotan-zad and Lu [31] focused on invariant recognition of characters.

The self-organized recognition is a further extension of the recognition originally proposed by Fukushima in 1975 [32]. It is an unsupervised learning algorithm. Although the work of Fukushima is a major advance in the understanding of visual processing in our brain, from an engineering point of view, its major drawback is that it is unable to cope with large translations and rotations in the image. Furthermore, the number of cells in this model increases almost linearly with the number of objects to be recognized, making the training process very slow.

Kwan and Cai [33] propose four layer feedforward fuzzy neural network (FNN) with unsupervised learning algorithm. Patrick Simpson [34] proposed supervised learning neural network classifier named as fuzzy min-max neural network (FMN) that utilizes fuzzy sets as pattern classes. Chiu and Tseng [35] used supervised FMN for invariant recognition of characters and shown that the FMN is superior to other traditional statistical classifiers. Kulkarni Uday and others [36] have proposed fuzzy hyperline segment neural network (FHLSNN) and the performance of FHLSNN is found superior than FMN algorithm.

Y. Avrithis et al. [37] present a scheme for translation, rotation and scale invariant OCR using triple-correlation. In this method, the image is represented in a triple-correlation domain. This representation is a one-to-one relation to the class of all shifted-rotated-scaled versions of the original image, as well as robust to a wide variety of

additive noises. They implement this technique to binary images and simulation results illustrate the performance is pretty good.

L. A. Torres-Méndez et al.[38] address a method for character recognition, which is invariant under translation, rotation, and scaling. The first step of the method (preprocessing) takes into account the invariant properties of the normalized moment of inertia and a novel coding that extracts topological object characteristics. The second step (recognition) is achieved by using a holographic nearest-neighbor algorithm (HNN), in which vectors obtained in the pre-processing step are used as inputs to it.

U. V. Kulkarni and T R Sontakke [39] propose a fuzzy hypersphere neural network (FHSNN) based classifier with its learning algorithm, which is used for rotation invariant character recognition. The FHSNN utilizes fuzzy sets as pattern classes in which each fuzzy set is a union of fuzzy set hyperspheres. After moment normalization, rotation invariant ring-data and Zernike moment features are extracted from characters. FHSNN algorithm is used to classify these features by its strong ability of discriminating ill-defined character classes.

N. Rishikesh and Y.V. Venkatesh [40] present a two-step algorithm for invariant pattern recognition. The proposed pattern encoder utilizes the properties of complex logarithmic mapping (CLM), which maps rotation and scaling in its domain to shift in its range. The encoder, then, invokes a pulse-coding scheme similar to that proposed by Dodwell [41] in order to handle these shifts, which is invariant to scaling, rotation and translation in the input shape. These pulse are then fed to a novel multilayered neural for final recognition.

The fuzzy-neuron classification method, defined by Charroux et al. [42], provides the definition of a degree of belonging of an unknown object without having any prior knowledge on the distribution of observations. This method is based on the Grenier's algorithm [41], whose principle is to calculate a potential vector where each component gives a degree of belonging to one of the classes.

C. Choisy [43] uses the rotation absorption property in neural networks for multi-oriented character recognition. Classical approaches are based on several rotation invariant features. Here, they propose to use a dynamic neural network topology to absorb the rotation phenomenon. The basic idea is to preserve as most as possible the

graphical information that contains all the information. The proposal is to dynamically modify the neural network architecture, in order to take into account the rotation variation of the analyzed pattern. They also use a specific topology that carries out a polar transformation inside the network.

High-order networks (HON's) have been utilized recently for invariant recognition [25], [26]. In this type of model, one has to encode the properties of invariance in the values of the synaptic weights. In other words, the known relations between pixels of the images are used, and the invariance is directly constructed in the network.

A third-order network has been proposed by L. Spirkovska [27], in which combinations of triplets of image pixels are used as invariant relations. The triplets form triangles representing similar angles (α, β, γ) in any transformation of the same image. The weights are restricted in such a way that all the combinations of three pixels defining similar triangles are connected to the output with the same weight. The main disadvantage of this approach is that the number of combinations of possible triplets increases in a nonlinear proportion to the number of input data.

Along with the previous techniques, it is important to mention recent ICR research based on optical techniques such as composite-harmonic filters [44] and scale, translation, and in-plane rotation (STIR)-invariant transformations [45]. The former filters involve the Mellin radial harmonics for scale invariance [46], the logarithmic harmonics for projection invariance [47], and the circular harmonics for rotation invariance [48].

Fang and Hausler [45] introduce a new class of transforms that achieve STIR invariance simultaneously. In their approach, an intensity function S(x,y) is mapped into a one-dimensional (1-D) frequency-spectrum function. Later, Ghahra-mani, and Patterson [49] propose a higher dimensional version of the STIR-invariant transforms in conjunction with an orthonormalization technique in an optical neural-network resonator. Computer simulations show that these types of techniques perform well and have excellent noise tolerance. However, the major disadvantage is their heavy computational requirements.

1.3 Objectives

The performance of pattern recognition systems depends on the specific feature extraction technique used to represent a pattern and the performance of a classifier. Generally, the dimension of a feature vector is reduced by removing the redundancy from the data and it is represented by a set of numerical values. The selected feature sets must possess small intraclass inconsistency and large interclass severance. It is therefore necessary to extract appropriate features from pattern automatically. The objectives of the present work are:

(i) To develop a new algorithm consisted of two stages to recognize translation, rotation and scale-invariant character.

(ii) To construct a feature extractor that can be used in the first stage of new algorithm.

(iii) To construct an efficient classifier that can be used in the second stage of the new algorithm.

(v) To determine the performance of new algorithm by applying it to real world data sets.

(vi) To compare the performance of the new algorithm with some existing algorithms.

1.4 Thesis Overview

The remaining chapters of this thesis are organized as follows:

- Chapter 2 provides background material for the rest of the thesis. Theoretical basis of the pattern recognition are first elaborated. Template matching, unitary image transformer, morphological operations and different types of projection methods are discussed next. Vector sum and its application on shape description and different types of moments are also presented. Finally, a brief discussion on ANN architectures and Backpropagation learning methods are given.
- Chapter 3 presents the proposed algorithm SAFER: Symmetry Axis based Feature Extraction and Recognition, which is the main contribution of this thesis. Advantages

of SAFER are enlisted first. Algorithm of SAFER is then elaborated; detailed descriptions of different processes and methods used in SAFER are then described.

- Chapter 4 presents a detailed experimental evaluation of the SAFER. In the reported experiments, the algorithm is applied to 26 uppercase English alphabetic characters to solve classification problems. This chapter also evaluates the performance of the SAFER on several well-known algorithms. Experimental details, results, analysis, and discussions are described.
- Finally, in chapter 5, the contributions and limitations of the research are presented in this concluding chapter, and propose future research tasks aimed at addressing the limitations.

Chapter 2

Background

2.1 Introduction

There is much architecture defined for the process of OCR. Similar to most of the pattern reorganization architecture "Feature Extraction and Comparison" is the basis for recognition of characters. There are many processes for extracting the features of a bitmap pattern.

There are basically two types of knowledge in the recognition of OCR. They are morphological and pragmatic. Morphological knowledge refers to the shape of an ideal representation of a character like number of vertical and horizontal lines, closed and open loops, curves and contours that define the character and the segments that join these. Pragmatic knowledge represents spatial arrangement of a character within a word boundary. This is more language dependent. There are many such language dependent pragmatic rules that can be applied to augment the recognition process.

"Feature extraction" will fall under Morphological Knowledge, which is done by applying a set of rules. This is derived from Hidden Markov Model (HMM). Each pattern is identified with number of closed loops, lines, curves, ascenders and descenders. The model uses the space between the ascenders or descenders to isolate the connected pixels between the characters. It also defines the segmentation points between these units. The extracted data is compared with stored (hidden) sets of data (hence the Hidden Markov Model) to recognize the character.

Feature Extraction is carried out by applying rules to a character bitmap and storing the extracted result of the rules in a table or in the form of pattern. This is then compared with the font data, retrieved from an installed font in the system. For accuracy, the user must select the font that closely matches the scanned document. Using Font data

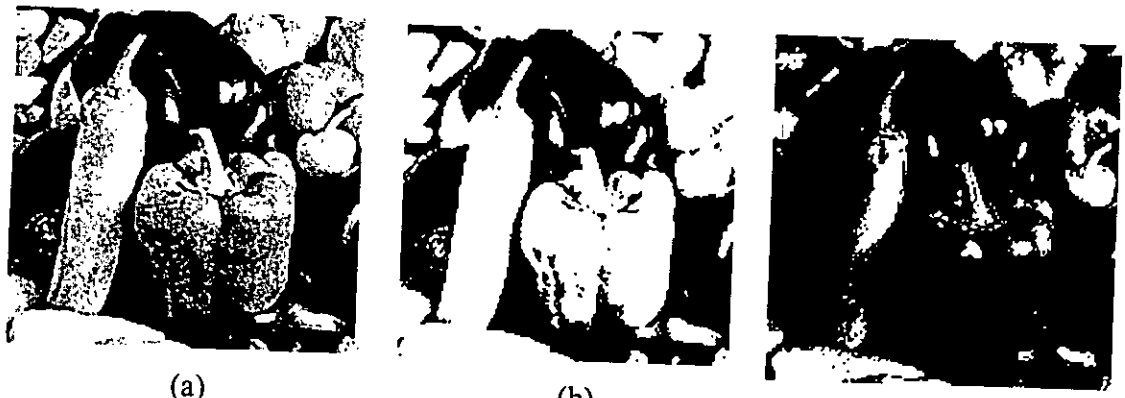
rather than the pre-scanned data provides the ability to use any font of any language, for recognition.

2.2 Thresholding

The thresholding transformation sets each gray level that is less than or equal to some prescribed value T (the threshold value) to 0, and each gray level greater than T is changed to K -*1*. Thresholding is useful when one wants to separate bright objects of interest from a darker background or vice versa. The thresholding transformation is defined by

$$g_2(x,y) = \begin{cases} 0, if..g_1(x,y) \le T \\ k-1, if..g_1(x,y) > T \end{cases} \qquad (2.1)$$

The thresholded desk image with $T = 100$ is shown in Fig. 2.1(b). Thresholding does not necessarily produce edges that correspond to the edges of the objects in the scene. Gray levels less than or equal to T are assigned the new gray level *0* and are considered to be background pixels. Gray levels greater than T are assigned the new value of K -*1*.



(a) (b) (c)

Fig. 2.1 Effect of Thresholding

(a) The gray scale Image. (b) Thresholded at $T = 100$ (c) Thresholded at $T = 200$

2.3 Range Normalization

It is sometimes desirable to apply a linear transformation to the gray levels in an image to change the range of gray levels that are present. If the original image had gray

levels ranging from a the lowest, to b the highest, and to make the range c to d, the change could be accomplished in three steps:

- Subtract a from each gray level to make the range become 0 to b-a.
- Multiply the result by $(d - c)/(b - a)$ to make the range become 0 to $d - c$.
- Add c to the result from step 2 to obtain the range c to d.

These three steps are summarized in the formula

$$g_2(x,y) = \frac{d-c}{b-a}[g_1(x,y) - a] + c \tag{2.2}$$

which converts the range $[a,b]$ to the range $[c,d]$.

2.4 Edge Detection

Experiments with characters show that edges are one of the most important visual clues for interpreting character. If an image consists of objects of interest displayed on a contrasting background, an edge is a transition from background to object or vice versa. The total change in intensity from background to foreground is called the strength of the edge.

The rate of change in gray level with respect to horizontal distance in a continuous image is equal to the partial derivative

$$\frac{\partial g(x,y)}{\partial x} = \lim_{\Delta x \to 0} \frac{g(x + \Delta x, y) - g(x,y)}{\Delta x}. \tag{2.3}$$

If Δx is replaced by 1, equation (2.3) becomes

$$g_x'(x,y) = g(x+1,y) - g(x,y) \tag{2.4}$$

Where g_x' represents the first difference in g with respect to x. Similarly $\partial g(x,y)/\partial y$ can be approximate by

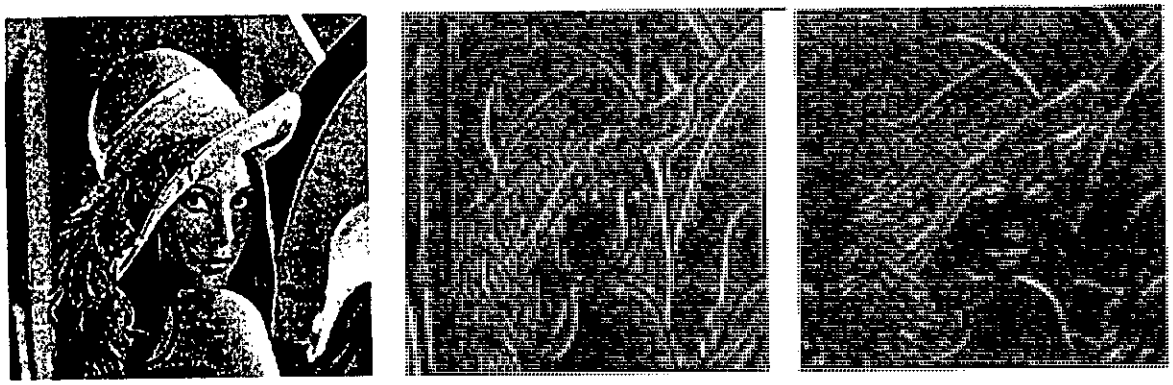
$$g_y'(x,y) = g(x,y+1) - g(x,y) \tag{2.5}$$

These finite differences represent the change in gray level from one pixel to the next and can be used to emphasize or detect abrupt changes in gray level in the image. These operators are often called edge detectors, as the edges of objects in a scene often produce such changes.

The one-dimensional edge detectors in (2.4) and (2.5) can be represented by the operators

$$\boxed{-1} \qquad \begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline \end{array} \tag{2.6}$$

The edge detectors $g'_x(x, y)$ and $g'_y(x, y)$ indicate how fast the gray level is increasing or decreasing with distance in the x and y directions. A positive value of $g'_x(x,y)$ indicates a transition from low gray level to high gray level when moving to the right. A negative value shows a transition from high to low. The edges detected using the operators (2.6) are shown in Fig. 2.2.



(a) (b) (c)

Fig. 2.2 Effect of Edge Detector

(a)The original image (lena.gif) (b) Vertical Edges (c) Horizontal Edges

2.5 Segmentation

A region can be loosely defined as a collection of adjacent pixels that are similar in some way, such as brightness, color, or local visual texture. The gray levels in the image being segmented may not represent the brightness values in the original scene since the current image could have resulted from applying various image transformations to the original image. Nonbackground regions are sometimes called objects.

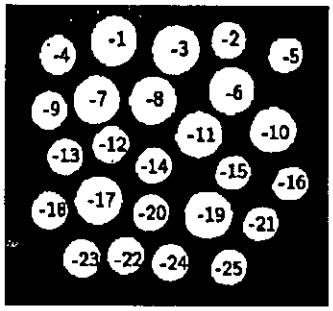


Fig. 2.3 The segmentation of an Image

2.6 Template Matching

Now a days OCR systems using template matching on gray scale character images are not very used. However, since template matching is a fairly standard image processing technique [50, 51], here this section is included for completeness. In template matching the feature extraction step is left out altogether and the character image it self is used as a "feature vector". In the recognition stage, a similarity (or dissimilarity) measure between each template T_j and the character image Z is computed. The template T_k which has the highest similarity measure is identified, and if this similarity is above a specified threshold, then the character is assigned the class label k. Else, the character remains unclassified. In the case of a dissimilarity measure, the template T_k having the lowest dissimilarity measure is identified, and if the dissimilarity is below a specified threshold, the character is given the class label k.

A common dissimilarity measure is the mean square distance D (Eq. 20.1-1 in Pratt [51]):

$$D_j = \sum_{i=1}^{M} (Z(x_i, y_i) - T_j(x_i, y_i))^2 \qquad (2.7)$$

where it is assumed that the template and the input character image are of the same size, and the sum is taken over the M pixels in the image. Eq. (2.7) can be rewritten as

$$D_j = E_z - 2E_{ZT_j} + E_{T_j} \qquad (2.8)$$

where,

$$E_Z = \sum_{i=1}^{M} (Z^2(x_i, y_i)) \qquad (2.9)$$

$$R_{ZT_j} = \sum_{i=1}^{M} (Z(x_i, y_i) T_j(x_i, y_i))$$

$$E_{T_j} = \sum_{i=1}^{M} (T_j^2(x_i, y_i))$$

E_Z and E_{Tj} are the total character image energy and the total template energy, respectively. R_{ZTj} is the cross-correlation between the character and the template, and could have been used as a similarity measure, but Pratt [51] points out that R_{ZTj} may detect a false match if, say, Z contains mostly high values. In that case, E_Z also has a high value, and it could be used to normalize R_{ZTj} by the expression $\overline{R}_{ZT_j} = R_{ZT_j} / E_Z$.

However, in Pratt's formulation of template matching, one wants to decide whether the template is present in the image (and get the locations of each occurrence). Our problem is the opposite: find the template that matches the character image best. Therefore, it is more relevant to normalize the cross-correlation by dividing it with the total template energy:

$$\hat{R}_{XT_j} = \frac{R_{ZT_j}}{E_{T_j}}. \tag{2.10}$$

Although simple, template matching suffers from some obvious limitations. One template is only capable of recognizing characters of the same size and rotation, is not illumination-invariant (in- variant to contrast and to mean gray level), and is very vulnerable to noise and small variations that occur among characters from the same class. However, many templates may be used for each character class, but at the cost of higher computational time since every input character has to be compared with every template. The character candidates in the input image can be scaled to suit the template sizes, thus making the recognizer scale- independent.

2.7 Unitary Image Transforms

In template matching, all the pixels in the gray scale character image are used as features. Andrews [52] applies a unitary transform to character images, obtaining a reduction in the number of features while preserving most of the information about the

character shape. In the transformed space, the pixels are ordered by their variance, and the pixels with the highest variance are used as features.

The unitary transform has to be applied to a training set to obtain estimates of the variances of the pixels in the transformed space. Andrews investigated the Karhunen-Loeve (KL), Fourier, Hadamard (or Walsh), and Haar transforms in 1971 [52]. He concluded that the KL transform was too computationally demanding, so he recommended to use the Fourier or Hadamard transforms. However, the KL transform is the only (mean-squared error) optimal unitary transform in terms of information compression [53]. When the KL transform is used, the same amount of information about the input character image is contained in fewer features compared to any other unitary transform.

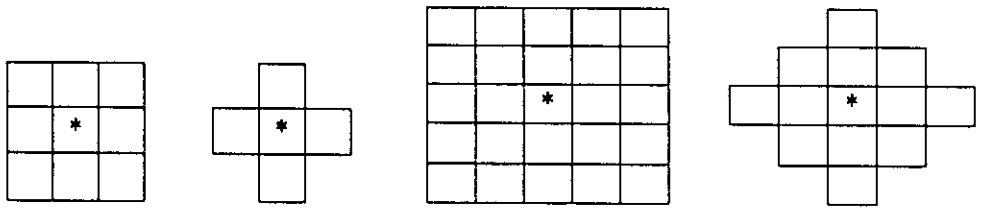
Other unitary transforms include the Cosine, Sine, and Slant transforms [53]. It has been shown that the Cosine transform is better in terms of information compression (e.g., see pp. 375-379 in [53]) than the other non-optimal unitary transforms. Its computational cost is comparable to that of the fast Fourier transform, so the Cosine transform has been coined "the method of choice for image data compression" [53].

The features extracted from unitary transforms are not rotation-invariant, so the input character images have to be rotated to a standard orientation if rotated characters may occur. Further, the input images have to be of exactly the same size, so a scaling or re-sampling is necessary if the size can vary. The unitary transforms are not illumination invariant, but for the Fourier transformed image the value at the origin is proportional to the average pixel value of the input image, so this feature can be deleted to obtain brightness invariance. For all unitary transforms, an inverse transform exists, so the original character image can be reconstructed.

2.8 Morphological Operations

When measuring the shapes of objects in images, it is sometimes desirable to simplify the objects by filling in small holes or by eliminating small protrusions from their boundaries. The elimination of boundary pixels from objects in binary image is called erosion. The boundary pixels of an object are defined as the object pixels that have background neighbors. Erosion consists of relabeling object boundary pixels as

background pixels, which has the effect of making object smaller. The opposite operation, dilation, enlarges objects. Each background pixel that has a neighbor in the object is relabeled as an object pixel. Fig. 2.4 shows symmetric erosion/dilation operator.



(a) (b) (c) (d)

Fig. 2.4 Some Erosion/dilation operator. Their origins are marked by asterisks.

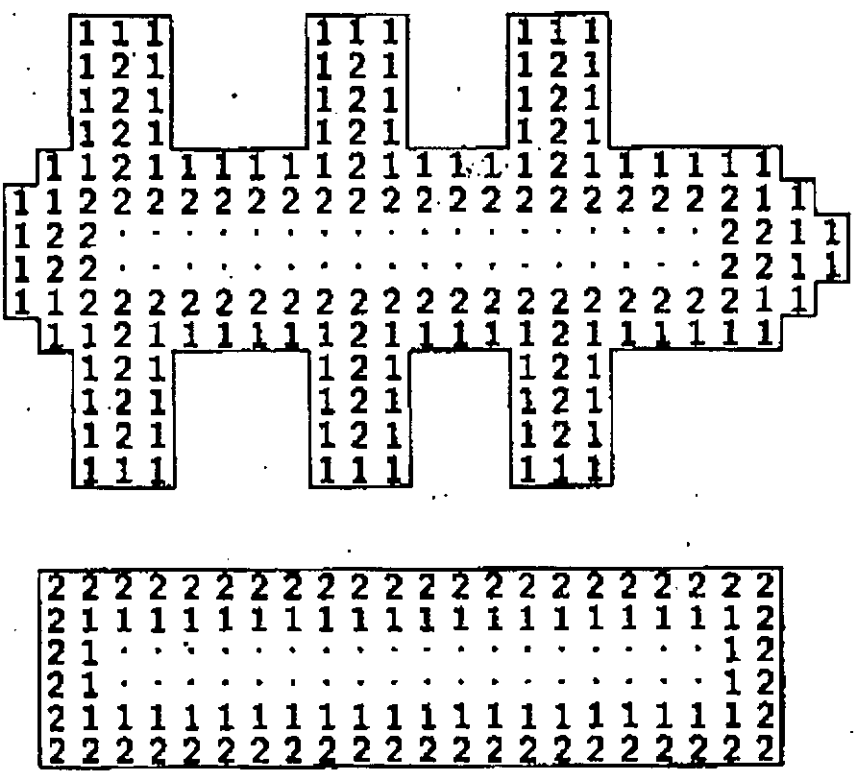


Fig. 2.5 The effect of erosion and dilation (a) The original region and the result of two 8-neighbor erosions. The pixels removed with the first and second erosion are marked with "1 " and "2," respectively. Unchanged pixels are marked with dots. (b) The eroded region (marked with dots) and the result of the two dilations. The pixels added by the first and dilation are marked with "1" and "2," respectively.

2.9 Thinning

A modification of erosion known as thinning converts any elongated parts or stripes in the image, regardless of their widths, into narrow stripes that are only about one pixel wide, but are still about as long as the original stripes. The narrow stripes lie near the centers of the original wide stripes. Thinning could be useful, for example, in analyzing images that contain fingerprints or handwriting. Also, if the output of an edge detector has been thresholded to find the edges in an image, the edges may be more than one pixel wide in some places. The positions of these edges could be refined by thinning the edge-detected image. The main problem with using simple erosion is that eroding a stripe enough to cause the widest part of it to be only one pixel wide produces gaps in the narrow parts of it.

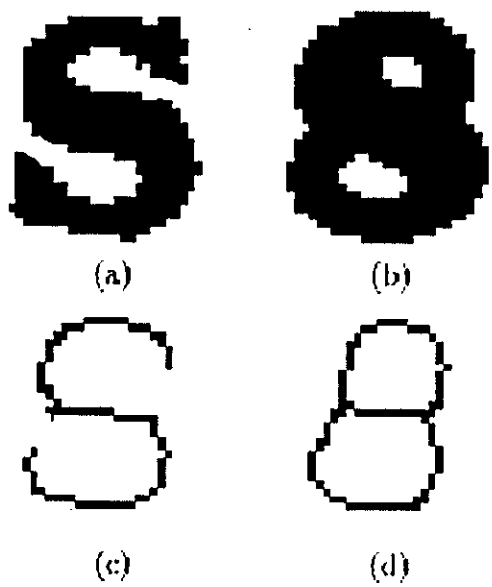


Fig 2.6 Effect of Thinning (a), (c) S and (b), (d) 8

2.10 Projections

In 1956, Glauberman [54] introduced Projection histograms in a hardware OCR system. Today, this technique is mostly used for segmenting characters, words, and text lines, or to detect if an input image of a scanned text page is rotated [55]. For a horizontal projection, $y(x_i)$ is the number of pixels with $x = x_i$. Fig. 2.7(a) shows a simple binary

image. Its horizontal projection is shown to the right of the image. It is found by summing the gray levels along horizontal rows of the image. The vertical projection is shown below the image. In Fig. 2.7(b) the horizontal and vertical projections are shown by histogram. Two-dimensional images can also be projected onto other one-dimensional lines or curves to aid in their analysis. A projection of an n×n image contains only about n data elements (possibly as many as *2n- 1* for diagonal projections), so working with projections can save a large amount of computer time and space compared to working on the original image.

The features can be made scale independent by using a fixed number of bins on each axis (by merging neighboring bins) and dividing by the total number of print pixels in the image. However, the projection histograms are very sensitive to rotation, and to some degree, variability in writing style. Also, important information about the character shape seems to be lost.

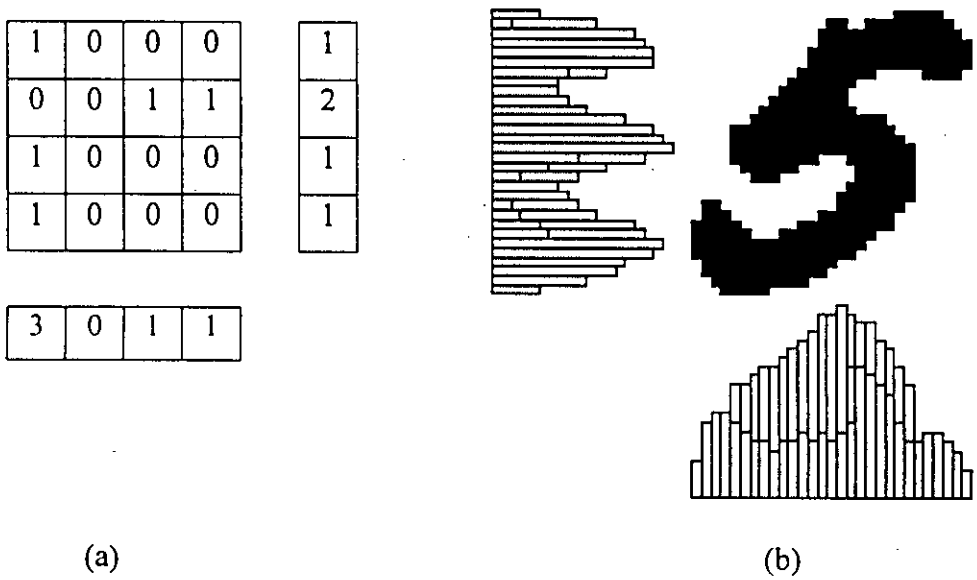


Fig.2.7 Effect of projection (a) A binary image with projections (b) Horizontal and vertical projection histograms

When measuring the dissimilarity between two histograms, it is tempting to use

$$d = \sum_{i=1}^{n} |y_1(x_i) - y_2(x_i)| \qquad (2.11)$$

where n is the number of bins, and y_1 and y_2 are the two histograms to be compared. However, it is more meaningful to compare the cumulative histograms $y(x_k)$, the sum of the k first bins,

$$Y(x_k) = \sum_{i=1}^{k} y(x_i) \tag{2.12}$$

using the dissimilarity measure

$$D = \sum_{i=1}^{k} |Y_1(x_i) - Y_2(x_i)| \tag{2.13}$$

where Y_1 and Y_2 denotes the cumulative histograms. The new dissimilarity measure D is not as sensitive as d to a slight misalignment of dominant peaks in the original histogram.

2.11 Circular Projection

Let us define a template to be $g(x,y)$ and an image $f(x,y)$ respectively. A rotated image is then defined as $f_R(x,y)$. Circular projection data, $c_g(r)$, for a template $g(x,y)$ can be obtained by the circular projection method [56,57]. $c_g(x,y)$ defined as the summation of pixel intensities along the circle whose radius to the center of the template r is as shown in Fig. 2.8, where R is the largest radius, and $r = (\text{int}([\sqrt{x^2 + y^2}])$

In order to get circular projections along concentric circles, it needs to find an efficient way of defining a circle in a digital domain. One of the simplest approaches for defining concentric circles while achieving computational efficiency is to use a lookup table whose diameter is set to the size of the template. Then a circular projection is obtained by summing up the pixel values along a concentric circle within the template result in the rotation invariant.

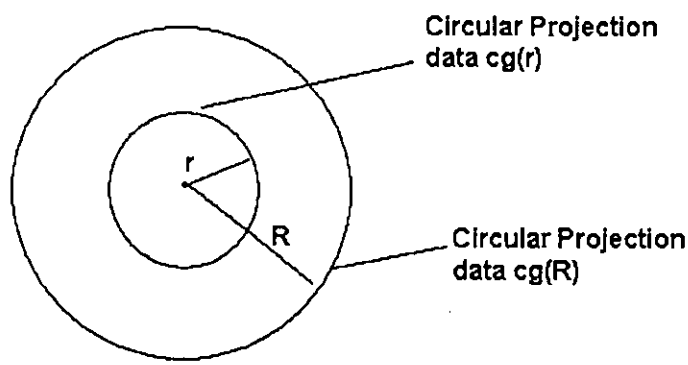


Fig. 2.8 Concept of the circular projection for a template

2.12 Vector Sum

Although a circular projection reduces a two-dimensional image into a one-dimensional vector, the computation required is still too high to process over whole image in a raster scan order. A new feature much less computation is proposed. The feature is called a vector sum of circular projections. Let the circular projection be $c_g(r)$ The vector sum is then defined as

$$F = \sum_{r=0}^{R} c_g(r) e^{-j2\pi r/(R+1)} \tag{2.14}$$

The vector sum is quite an effective descriptor for the template and the sub-image for the following reasons. First, F is invariant to the overall changes of the image intensities because the vector sum eliminates the component of the circular projection data. Second, the effects of the random and Gaussian noises are reduced due to the integration operation of the projection. [58]. Using the vector sum alone, candidates for the matching are selected from the whole image.

- **Meaning of Vector sum as a shape descriptor**

Discrete Fourier series of the projection profile, $C_g(k)$, is defined as follows:

$$C_g(k) = \sum_{r=0}^{R} c_g(r) e^{-j(2\pi/(R+1))kr} \tag{2.15}$$

Coefficient of the first harmonics, where k=1, is then

$$C_g(1) = \sum_{r=0}^{R} c_g(r) e^{-j(2\pi/(R+1))r} \tag{2.16}$$

Here, notice that Eq.(2.16) is the same as Eq.(2.14) which is the equation for the vector sum. That is the physical meaning of the vector sum is equivalent sum to the coefficient of the first harmonic of the ld-signal of the projection. Since a large portion of the energy of a signal is concentrated in lower frequencies, a set of harmonics at low frequencies might be adequate to be used for a selection of candidate signals.

To describe the signal more accurately, it needs to introduce a higher order vector sum such as the second order and the third order vector sum. Fig.2.10 shows the concept of the second and third order vector sum.

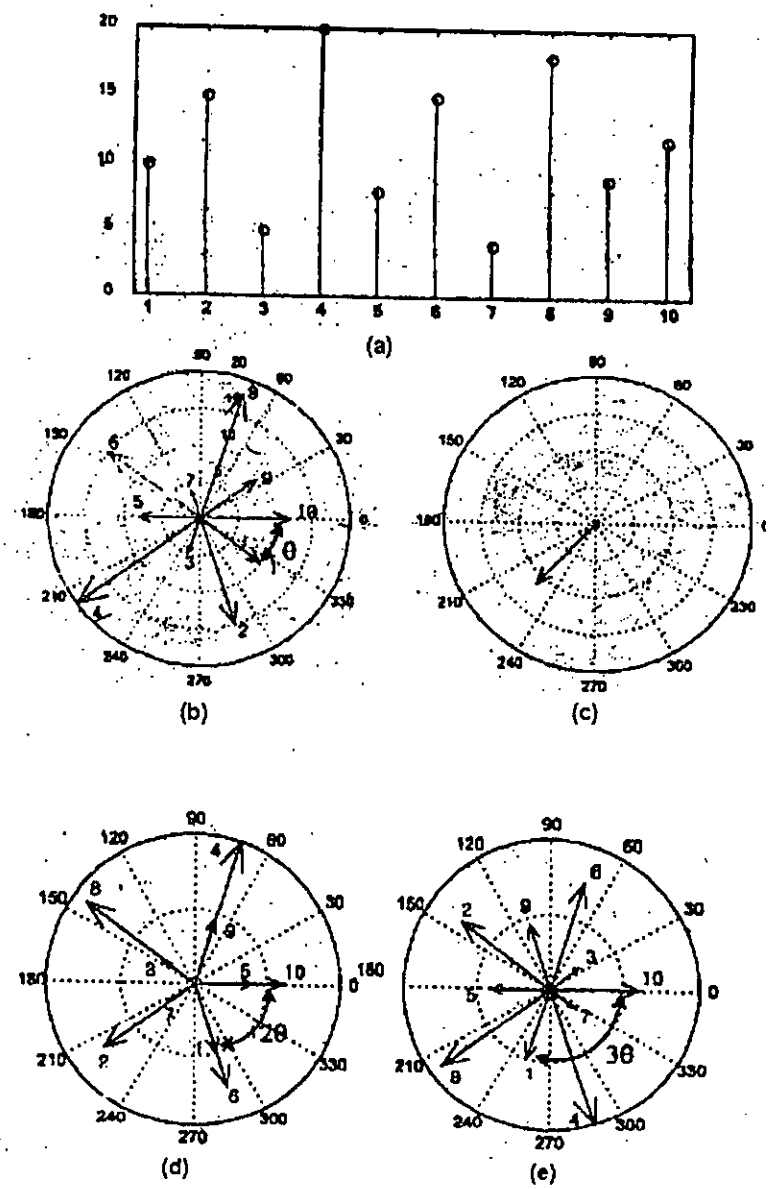


Fig. 2.9 Effect of harmonics on vector sum of a signal. (a) 1-d Signal (b) Vector representation of the 1-d signal in (a) (c) Their vector sum (d) Second order vector sum (e) Third order vector sum.

- **Frequency Domain computation**

Computing the vector sum over the whole image is equivalent to convolving a mask over the image. Since convolution is generally a time consuming process, a frequency domain process can be utilized to reduce the time complexity.

$$f * m = T^{-1}(T(f) \times T(m)) \tag{2.17}$$

Here, f is the image, m is the mask and T is the Fourier transform.

Eq. (2.14) can be rewritten as

$$F = \sum_{r=0}^{R} \sum_{\theta=0}^{2\pi} f(r,\theta) e^{-j2\pi r/(R+1)} \tag{2.18}$$

Conceptually, Eq(2.18) is the same as the masking operation over the image where as Eq(2.17) is for the mask,

$$m(r,\theta) = e^{-j2\pi r/(R+1)} \tag{2.19}$$

Using the frequency domain computation, candidate points for the matching will be selected with lesser computational complexity when the size of the mask is relatively large compared to the image size.

2.13 Features Extracted From the Binary Contour

The closed outer contour curve of a character is a closed piecewise linear curve that passes through the centers of all the pixels which are 4-connected to the outside background, and no other pixels. Following the curve, the pixels are visited in, say, counter-clockwise order, and the curve may visit an edge pixel twice at locations where the object is one-pixel wide. Each line segment is a straight line between the pixel centers of two 8-connected neighbors.

By approximating the contour curve by a parametric expression, the coefficients of the approximation can be used as features. By following the closed contour successively, a periodic function results. Periodic functions are well-suited for Fourier series expansion, and this is the foundation for the Fourier-based methods discussed below.

- **Contour Profiles**

The motivation for using contour profiles is that each half of the contour (Fig.2.10) can be approximated by a discrete function of one of the spatial variables, x or y. Then, features can be extracted from discrete functions. One can use vertical or horizontal profiles, and they can be either outer profiles or inner profiles.

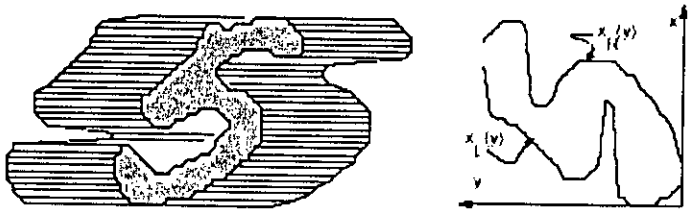


Fig. 2.10 Contour profile of Digit '5'. With left profile $x_L(y)$ and right profile $x_R(y)$. For each y value, the left (right) profile value is the left most (rightmost) x value on the character contour.

To construct vertical profiles, first locate the uppermost and lowermost pixels on the contour. The contour is split at these two points. To get the outer profile, for each y value, select the outermost x value on each contour half (Fig. 2.11). To get the inner profiles, for each y value, the innermost x values are selected. Horizontal profiles can be extracted in a similar fashion, starting by dividing the contour in upper and lower halves. The profiles are themselves dependent on rotation (e.g., try to rotate the '5' in Fig. 2.11, say, 45° before computing the profiles). Therefore, all features derived from the profiles will also be dependent on rotation.

2.14 Moments

Moments are applicable to many different aspects of image processing, ranging from invariant pattern recognition and image encoding to pose estimation. When applied to images, they describe the image content (or distribution) with respect to its axes. They are designed to capture both global and detailed geometric information about the image. Moments are classified in two types

i) Non-Orthogonal Moment

ii) Orthogonal Moment

2.14.1 Non-Orthogonal Moment

Non-orthogonal moments are generally calculated around the axis and the sum approximately represents the image. The main problem with non-orthogonal moments is that redundant data exists and when the order of moment increases more precision is

required to separate them. So, for non-orthogonal moments, the image reconstruction is not straightforward and requires a moment-matching technique.

- **Cartesian Moment**

Hu [59, 60], stated that the continuous two-dimensional $(p + q)^{th}$ order Cartesian moment is defined in terms of Riemann integrals as:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy \tag{2.20}$$

It is assumed that $f(x, y)$ is a piecewise continuous, bounded function and that it can have non-zero values only in the finite region of the $x - y$ plane (i.e. all values outside the image plane are zero)

The discrete version of the Cartesian moment for an image consisting of pixels P_{xy}, replacing the integrals with summations, is:

$$m_{pq} = \sum_{x=1}^{M}\sum_{y=1}^{N} x^p y^q P_{xy} \tag{2.21}$$

m_{pq} two dimensional Cartesian moment Where M and N are the image dimensions and the monomial product $x^p y^q$ is the basis function. Fig. 2.11 illustrates the non-orthogonal (highly correlated) nature of these monomials plotted for the positive x axis only.

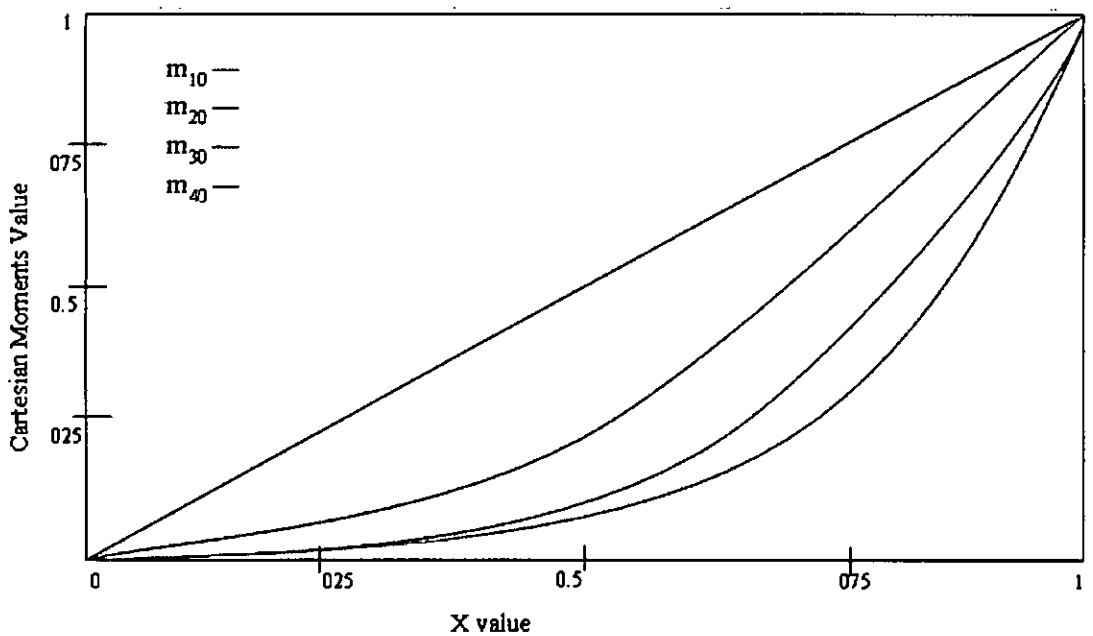


Fig. 2.11 The first five Cartesian Moment

The zero order moment m_{00} is defined as the total mass (or power) of the image. If this is applied to a binary (i.e. a silhouette) $M \times N$ image of an object, then this is literally a pixel count of the number of pixels comprising the object.

$$m_{00} = \sum_{x=1}^{M} \sum_{y=1}^{M} P_{xy} \tag{2.22}$$

The two first order moments are used to find the Centre Of Mass (COM) of an image. If this is applied to a binary image and the results are then normalized with respect to the total mass (m_{00}), then the result is the centre co-ordinates of the object. Accordingly, the centre co-ordinates $\bar{x}, \bar{y}$, where $\bar{x}, x$ axis centre of mass and $\bar{y}, y$ axis centre of mass are given by:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \bar{y} = \frac{m_{01}}{m_{00}} \tag{2.23}$$

The COM describes a unique position within the field of view which can then be used to compute the centralised moments of an image.

- **Centralised moments**

The definition of a discrete centralised moment as described by Hu[60] is: μ_{pq} Two dimensional centralised moment

$$\mu_{pq} = \sum_{x=1}^{M} \sum_{y=1}^{N} (x - \bar{x})^p (y - \bar{y})^q P_{xy} \tag{2.24}$$

This is essentially a translated Cartesian moment, which means that the centralised moments are invariant under translation. To enable invariance to scale, normalised moments η_{pq} are used in [61], given by:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \tag{2.25}$$

where, $\gamma = \frac{p+q}{2} + 1, \qquad \forall (p+q) \geq 2$

- **Hu invariant set**

The non-orthogonal centralised moments are translation invariant and can be normalised with respect to changes in scale. However, to enable invariance to rotation they require reformulation. Hu [60] described two different methods for producing rotation invariant moments.

The first used a method called principal axes, however this method can break down when images do not have unique principal axes. Such images are described as being rotationally symmetric. The second method Hu described is the method of absolute moment invariants and is discussed here. Hu derived these expressions from algebraic invariants applied to the moment generating function under a rotation transformation. They consist of groups of nonlinear centralised moment expressions. The result is a set of absolute orthogonal (i.e. rotation) moment invariants, which can be used for scale, position, and rotation invariant pattern identification. These were used in a simple pattern recognition experiment to successfully identify various typed characters. They are computed from normalised centralised moments up to order three and are shown below I_n^{th} Hu invariant moment:

$$I_1 = \eta_{20} + \eta_{02} \tag{2.26}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{2.27}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{2.28}$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{2.29}$$

and so on.

These moments are of finite order; therefore, unlike the centralised moments they do not comprise a complete set of image descriptors, [62]. However, higher order invariants can be derived

2.14.2 Orthogonal moments

Cartesian moments are formed using a monomial basis set $x^p y^q$. This basis set is non-orthogonal and this property is passed onto the Cartesian moment. These monomials increase rapidly in range as the order increases, producing highly correlated descriptions. This can result in important descriptive information being contained within small differences between moments, which lead to the need for high computational precision (Fig. 2.11).

The non-orthogonal centralised moments are translation invariant and can be normalised with respect to changes in scale. However, to enable invariance to rotation they require reformulation. Hu [60] described two different methods for producing rotation invariant moments.

The first used a method called principal axes, however this method can break down when images do not have unique principal axes. Such images are described as being rotationally symmetric. The second method Hu described is the method of absolute moment invariants and is discussed here. Hu derived these expressions from algebraic invariants applied to the moment generating function under a rotation transformation. They consist of groups of nonlinear centralised moment expressions. The result is a set of absolute orthogonal (i.e. rotation) moment invariants, which can be used for scale, position, and rotation invariant pattern identification. These were used in a simple pattern recognition experiment to successfully identify various typed characters. They are computed from normalised centralised moments up to order three and are shown below I_n^{th} Hu invariant moment:

$$I_1 = \eta_{20} + \eta_{02} \tag{2.26}$$

$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{2.27}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{2.28}$$

$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{2.29}$$

and so on.

These moments are of finite order; therefore, unlike the centralised moments they do not comprise a complete set of image descriptors, [62]. However, higher order invariants can be derived

2.14.2 Orthogonal moments

Cartesian moments are formed using a monomial basis set $x^p y^q$. This basis set is non-orthogonal and this property is passed onto the Cartesian moment. These monomials increase rapidly in range as the order increases, producing highly correlated descriptions. This can result in important descriptive information being contained within small differences between moments, which lead to the need for high computational precision (Fig. 2.11).

However, moments produced using orthogonal basis sets exist. These orthogonal moments have the advantage of needing lower precision to represent differences to the same accuracy as the monomials. The orthogonality condition simplifies the reconstruction of the original function from the generated moments. Orthogonality means mutually perpendicular, expressed mathematically - two functions y_m and y_n are orthogonal over an interval $a \leq x \leq b$ if and only if:

$$\int_a^b y_m(x) y_n(x) dx = 0; m \neq n \tag{2.30}$$

For discrete images, the integrals within the moment descriptors can be replaced by summations. It is noted that a sequence of polynomials, which are orthogonal with respect to integration, are also orthogonal with respect to summation, [63]. One such (well established) orthogonal moment is Zernike moment.

- **Complex Zernike moments**

The Zernike polynomials were first proposed in 1934 by Zernike [64]. His moment formulation appears to be one of the most popular, outperforming the alternatives [65] (in terms of noise resilience, information redundancy and reconstruction capability). The pseudo-Zernike formulation proposed by Bhatia and Wolf [66] further improved these characteristics. However, here the original formulation of these orthogonal invariant moments is studied.

Complex Zernike moments [67] are constructed using a set of complex polynomials, which form a complete orthogonal basis set defined on the unit disc $(x^2 + y^2 \leq 1)$. They are expressed as A_{pq}. Two-dimensional Zernike moment:

$$A_{mn} = \frac{m+1}{\pi} \iint f(x,y)[V_{mn}(x,y)]^* dxdy \qquad \text{where } x^2 + y^2 \leq 1 \tag{2.31}$$

where $m=0, 1, 2 \ldots \infty$ and defines the order, $f(x, y)$ is the function being described and * denotes the complex conjugate. While n is an integer (that can be positive or negative) depicting the angular dependence, or rotation, subject to the conditions:

$$m - |n| = even, |n| \leq m \tag{2.32}$$

and $A^*_{m,n} = A_{m,-n}$ is true. The Zernike polynomials $V_{mn}(x,y)$ Zernike polynomial expressed in polar coordinates are:

$$V_{mn}(r,\theta) = R_{mn}(r)\exp(jn\theta) \tag{2.33}$$

where $(r, \theta)$are defined over the unit disc, $R_{mn}(r)$is the orthogonal radial polynomial, defined as $R_{mn}(r)$ Orthogonal radial polynomial:

$$R_{mn}(r) = \sum_{s=0}^{\frac{m-|n|}{2}} (-1)^s F(m,n,s,r) \tag{2.34}$$

where:

$$F(m,n,s,r) = \frac{(m-n)!}{s!(\frac{m+|n|}{2}-s)!(\frac{m-|n|}{2}-s)!} r^{m-2s} \tag{2.35}$$

Where, $R_{mn}(r) = R_{m,-n}(r)$ and it must be noted that if the conditions in Eq. 2.32 are not met, then $R_{mn}(r)=0$.

Fig. 2.12 shows five such radial responses, where it can be seen that the polynomials become more grouped, as they approach the edge of the unit disc (r approaches unity). (Care must be taken with regard to the accuracy of these polynomial calculations as the factorial operations can quickly produce large integer values, even at relatively low order m). The difference between these orthogonal polynomials and the non-orthogonal monomials can be seen by comparing Fig. 2.12 with Fig. 2.11.

To calculate the Zernike moments, the image (or region of interest) is first mapped to the unit disc using polar coordinates, where the centre of the image is the

origin of the unit disc. Those pixels falling outside the unit disc are not used in the calculation. The coordinates are then described by the length of the vector from the origin to the coordinate point, r, and the angle from the x axis to the vector r polar co-ordinate radius, θ polar co-ordinate angle, by convention measured from the positive x axis in a counter clockwise direction.

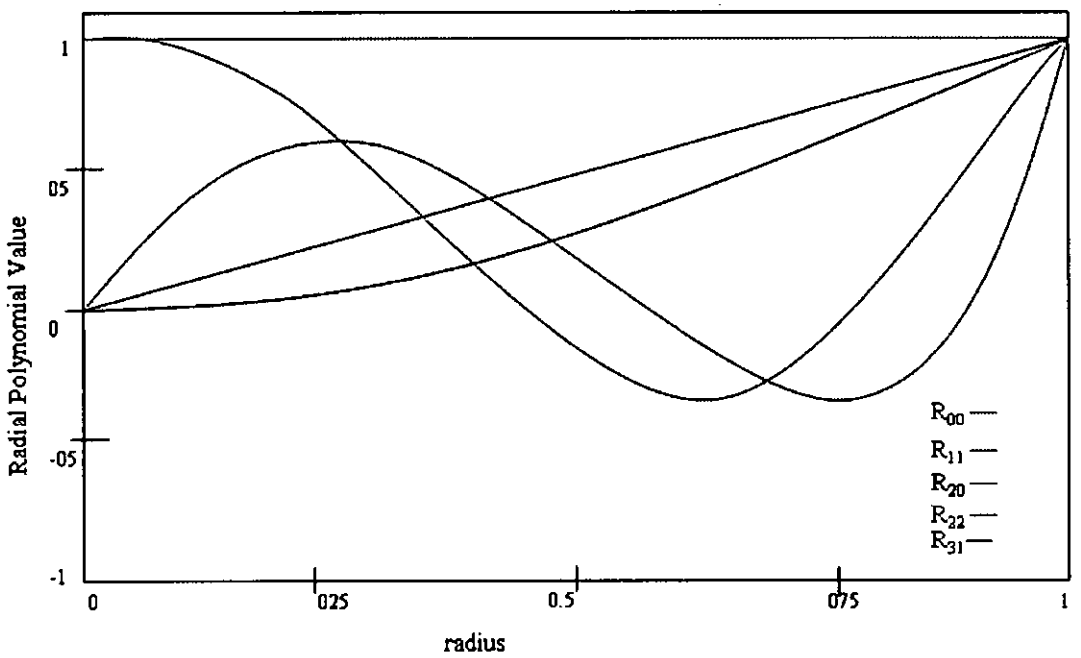


Fig. 2.12 Five orthogonal radial polynomial plotted for increasing radius

2.15 Artificial Neural Network

An artificial neural network (ANN) can be defined as

A data processing system consisting of a large number of simple highly interconnected processing elements (artificial neurons) in an architecture inspired by the structure of the cerebral cortex of the brain.

A neuron is an information-processing element that is fundamental to the operation of an ANN. The block diagram of Fig. 2.13 shows the model of a neuron, which forms the basis for designing artificial ANNs. The three basic elements of the neural model are discussed below:

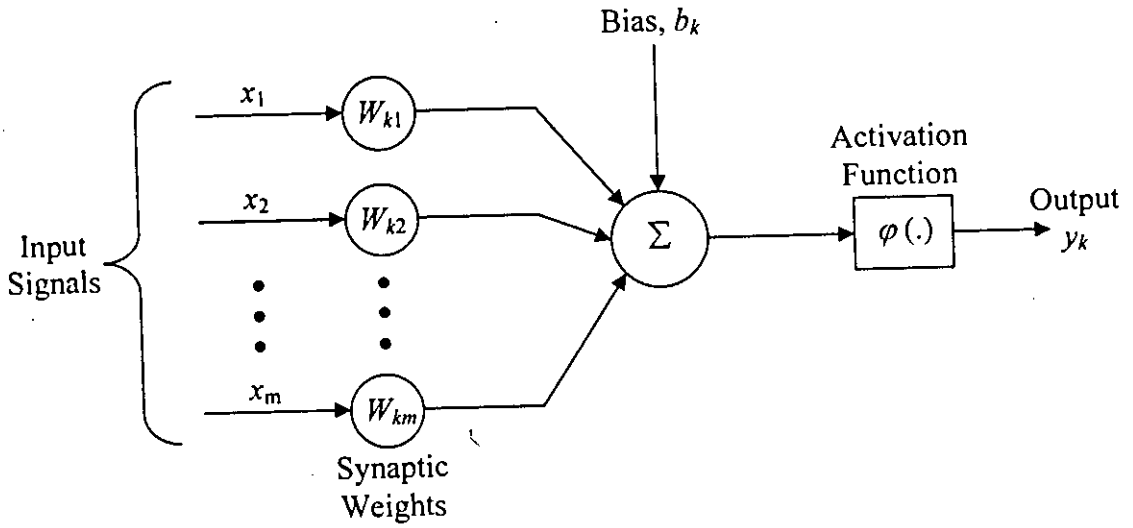


Fig. 2.13 Model of a neuron

A. *A set of synapse or connecting links,* each of which is characterized by a weight. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. Unlike a synapse in the brain, the synaptic weight of artificial neuron may lie in the range that includes negative as well as positive values.

B. *An adder* for summing the input signals, weighted by the respective synapse of the neuron; the operations described here constitutes a linear combiner.

C. *An activation function* for limiting the amplitude of the output of a neuron. The activation function is also referred to as a squashing function in that it squashes (limits) the permissible amplitude range of output signal to some finite value. Typically, the normalized amplitude range of the output of a neuron is written as the closed node interval [0, 1] or alternatively [-1, 1].

The neural model of Fig. 2.13 also includes an externally applied bias denoted by b_k. The bias b_k has the effect of increasing or lowering the net input of the activation function, depending on whenever it is positive or negative, respectively.

In mathematical terms, a neuron k may describe by writing the following pairs of equations:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \qquad 2.36)$$

and

$$y_k = \varphi(u_k + b_k) \qquad (2.37)$$

where $x_1, x_2, \ldots, x_m$ are the input signals; $w_{k1}, w_{k2}, \ldots, w_{km}$ are the synaptic weights of neuron k; u_k is the linear combiner due to the input signals; b_k is the bias; $\varphi(\,.\,)$ is the activation function; and y_k is the output signal of the neuron. The use of bias b_k has the effect of applying an affine transformation to the output u_k of the linear combiner in the model of Fig. 2.3, as shown by

$$v_k = u_k + b_k \qquad (2.38)$$

The bias b_k is can be considered as an internal parameter of artificial neuron k. Considering this, Eqs. (2.1 – 2.3) can be formulated as follows:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j \qquad (2.39)$$

and

$$y_k = \varphi(v_k) \qquad (2.40)$$

These processing elements are usually organized into a sequence of layers or slabs with full or random connections between the layers. This arrangement is shown in Fig.2.14, where the input layer is a buffer that presents data to the network. This input layer is not a neural computing layer because the nodes have no input weights and no activation functions. The top layer is the output layer, which presents the output response to a given input. The other layer (or layers) is called the intermediate or hidden layer because it usually has no connections to the outside world.

Two general kinds of neural networks are in use: the heteroassociative neural network in which the output vector is different than the input vector and the autoassociative neural network in which the output is identical to the input.

A typical neural network is "fully connected," which means that there is a connection between each of the neurons in any given layer with each of the neurons in the next layer as shown in Fig. 2.15. When there are no lateral connections between neurons in a given layer and none back to previous layers, the network is said to be a feedforward network.

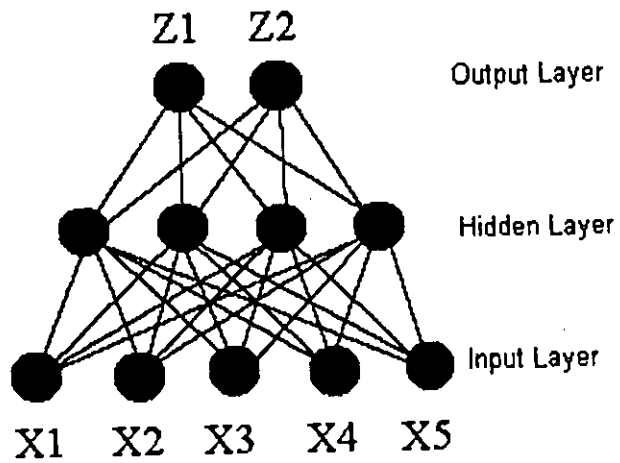


Fig. 2.14 Tree Layer Feedforward Neural Network

2.16 The Backpropagation Algorithm

There are some algorithms for training multilayered ANNs. Among them the backpropagation (BP) algorithm is the most prolific one. In the proposed algorithm (MCA), BP algorithm is used in a modified manner. So, this algorithm is described in detail in this section.

The BP learning algorithm involves two phases. During the first phase the input is presented and propagated forwarder through the ANN to compute the output value for each node. This output is then compared with the targets, resulting an error signal for each output node. The second phase involves a backward pass through the ANN during which the error signal δ is passed to each node in the ANN and the appropriate weight changes are made. This second backward pass allows the recursive computation of δ as indicated above. The first step is to compute δ for each of the output nodes. This simply

the difference between the actual and desired output values times the derivative of the squashing function. Then the weight changes for all connections that feed into the final layer can be computed. After this is done, then compute δ's for all nodes in the penultimate layer. This propagates error back one layer and same process can be repeated for every layer.

Let us consider an input vector $X_p = (x_1, x_2 \ldots x_n)$, is applied to the input layer of the ANN. The "p" subscript refers to the pth training vector. The input nodes distribute the values to the hidden layer nodes. The net input to the j_{th} hidden node is,

$$net_{pj}^{h} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h} \tag{2.41}$$

Where, w_{ji}^{h} is the weight of the connection from i^{th} input node to j^{th} hidden node and θ_{j}^{h} is the bias term. The "h" subscript refers to the quantity on the hidden layer. Assuming that activation of the node is equal to the net input; then the output of the node is,

$$i_{pj} = f_{j}^{h}(net_{pj}^{h}) \tag{2.42}$$

The equations of the output nodes are,

$$net_{pk}^{0} = \sum_{j=1}^{L} w_{kj}^{o} i_{pj} + \theta_{k}^{o} \tag{2.43}$$

$$o_{pk} = f_{k}^{o}(net_{pk}^{o}) \tag{2.44}$$

- **Update of Output-Layer Weights**

The error at a single output node is defined as $\delta_{pk} = y_{pk} - o_{pk}$, where the subscript "$p$" refers to the p^{th} training vector, and "k" refers to the k^{th} output node.

In this case y_{pk} is the desired output and o_{pk} is the actual output of the k^{th} node. The error to be minimized is the sum of the squares of the errors for all output nodes:

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta_{pk}^2 \tag{2.45}$$

To determine the direction in which the change of weights, the negative of the gradient of E_p, ∂E_p, with respect to weights, w_{kj} is calculated. Then the values of the weights can be adjusted such that the total error can be reduced. It is often usual to think of E_p as a surface in the weight space.

From Eq. (2.45) and the definition of δ_{pk}

$$E_p = \frac{1}{2}\sum_{k=1} (y_{pk} - o_{pk})^2 \tag{2.46}$$

$$\frac{\partial E_p}{\partial w_{kj}^o} = -(y_{pk} - o_{pk}) \frac{\partial f_k^o}{\partial (net_{pk}^o)} \frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} \tag{2.47}$$

Where, Equation (2.46) is used for the output value, o_{pk}, and the chain rule for partial derivatives. The last factor of Equation (2.47) is,

$$\frac{\partial (net_{pk}^o)}{\partial w_{kj}^o} = \frac{\partial}{\partial w_{kj}^o} \sum_{j=1}^{L} w_{kj}^o i_{pj} + \theta_k^o = i_{pj} \tag{2.48}$$

Combining Equations (2.47) and (2.48), the negative gradient,

$$-\frac{\partial E_p}{\partial w_{kj}^o} = (y_{pk} - o_{pk}) f_k^{o'} (net_{pk}^o) i_{pj} \tag{2.49}$$

As far the magnitude of the weight change is concerned, it has been taken to be proportional to the negative gradient. Thus the weights on the output layer are updated according to

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \Delta_p w_{kj}^o(t) \tag{2.50}$$

Where,

$$\Delta_p w_{kj}^o = \eta(y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) i_{pj} \tag{2.51}$$

The factor η is called the learning rate parameter. If the sigmoid function is used then the weight update equation for output node is,

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta(y_{pk} - o_{pk}) o_{pk}(1 - o_{pk}) i_{pj} \tag{2.52}$$

By defining output layer error term,

$$\delta_{pk}^o = \eta(y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) = \delta_{pk} f_k^{o'}(net_{pk}^o) \tag{2.53}$$

By combining equations (2.52) and (2.53) the weight update equation becomes,

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta \delta_{pk}^o i_{pj} \tag{2.54}$$

- **Update of Hidden-Layer Weights**

The error of the hidden layer is given by,

$$\begin{aligned} E_p &= \frac{1}{2}\sum_k (y_{pk} - o_{pk})^2 \\ &= \frac{1}{2}\sum_k (y_{pk} - f_k^o(net_{pk}^o))^2 \\ &= \frac{1}{2}\sum_k (y_{pk} - f_k^o(\sum_j w_{kj}^o i_{pj} + \theta_k^o))^2 \end{aligned} \tag{2.55}$$

The gradient of E_p with respect to hidden layer weights,

$$\begin{aligned} \frac{\partial E_p}{\partial w_{ji}^h} &= \frac{1}{2}\sum_k \frac{\partial}{\partial w_{ji}^h}(y_{pk} - o_{pk})^2 \\ &= -\sum_k (y_{pk} - o_{pk}) \frac{\partial o_{pk}}{\partial (net_{pk}^o)} \frac{\partial (net_{pk}^o)}{\partial i_{pj}} \frac{\partial i_{pj}}{\partial (net_{pj}^h)} \frac{\partial (net_{pj}^h)}{\partial w_{ji}^h} \end{aligned} \tag{2.56}$$

Each of the factors of Equation (2.56) can be calculated explicitly from previous equations. The result is,

$$\frac{\partial E_p}{\partial w_{ji}^h} = -\sum_k (y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) w_{kj}^o f_j^{h'}(net_{pj}^h) x_{pi} \tag{2.57}$$

The hidden layer weights update in proportion to negative of the Equation (2.57):

$$\Delta_p w_{ji}^h = \eta f_j^{h'}(net_{pj}^h) x_{pi} \sum_k (y_{pk} - o_{pk}) f_k^{o'}(net_{pk}^o) w_{kj}^o \tag{2.58}$$

By using Equation (2.53),

$$\Delta_p w_{ji}^h = \eta f_j^{h'}(net_{pj}^h) x_{pi} \sum_k \delta_{pk}^o w_{kj}^o \tag{2.59}$$

Every weight update on the hidden layer depends on all error terms, δ_{pk}^o, on the output layer. The known errors on the output layers are propagated back to the hidden layer to determine the appropriate weight changes on that layer. By defining hidden layer error term,

$$\delta_{pj}^h = f_j^{h'}(net_{pj}^h) \sum_k \delta_{pk}^o w_{kj}^o \tag{2.60}$$

So the weight update equation becomes analogous to those for the output layer:

$$w_{ji}^h(t+1) = w_{ji}^h(t) + \eta \delta_{pj}^h x_{pi} \tag{2.61}$$

The value of η is commonly chosen between 0.25 and 0.75 by the ANN user, and usually reflects the *rate of learning* to ensure that the ANN will settle to a solution.

Chapter 3

Symmetry Axis based Feature Extraction and Recognition

3.1 Introduction

Invariant features of certain transformations are necessary to recognize many variations of the same character. They have approximately the same values for a particular character that is, for example, translated, scaled or rotated. However, not all variations among characters from the same character class can be modeled by using invariants. Translation and scaling invariance can be achieved easily. The segmentation of individual characters can itself provide estimates of size and location.

Rotation invariance is important if characters to be recognized appear in any orientation. If all characters are expected to have 360^0 rotation, rotation-variant features may cause problem to distinguish between similar characters such as 'M' and 'W' and 'n' and 'u'. An approach to solve this problem is to use rotation-invariant features with the detected rotation angle. If the rotation angle is restricted, say, to lie between -45° and 45°, characters that are, say 180° rotations of each other can be differentiated.

The aim of this chapter is to introduce a new character recognition algorithm for recognizing character independent of translation, rotation and scaling. Details about each component of the proposed algorithm are elaborately described in this chapter.

3.2 The SAFER Algorithm

In this thesis, a new algorithm SAFER is proposed for recognizing characters, which achieve invariance under translation, rotation, and scaling. The algorithm consisted of two steps: preprocessing and recognition. In first stage, the SAFER takes into account the axis of symmetry of a character and a novel coding that extracts topological

characteristics of the character. In the second stage, an efficient classifier is used to recognize the character that utilizes topological characteristics obtained in the first stage.

In comparison with other existing algorithms, the major advantages of SAFER include i) it determine the axis of symmetry of a character automatically by using a radial coding technique; ii) it extracts topological characteristics in such a way that they are invariant to rotation iii) it is very simple since it does not use complex calculation and redundant matching technique iv) it uses an ANN classifier that can automatically determine its architecture.

The major steps of SAFER can be described as follows. Fig. 3.1 is a flow chart of these steps.

Step 1: Select a gray scale input character image and binaries it by thresholding operation. Calculate the center of mass (COM) of the image by using Eqs. 2.22 and 2.23. The COM is invariant of translation and scaling.

Step 2: Generate k equidistant concentric circles C_i around the centroid. The spacing is equal to the distance between the centroid and the furthest pixel of the object divided by k as shown in Fig. 3.2.

Step 3: Count the number of intensity changes for each circular boundary. An intensity change occurs when a line enter from background to character area or from character area to background as seen in the red points of Fig. 3.3. Locate their angular positions by taking the right horizontal axis as the reference axis.

Step 4: Find the midpoints of each two successive positions as marked white points in Fig. 3.3 and calculate their angular positions. Create a vector R_i using these angles expressed radian.

Step 5: Determine the axis of symmetries for each circle, if exist, by using R_i.

Step 6: Detect the reference axis side i.e. 0 degree line for each circle by using R_i.

Step 7: Generate a pattern vector by using the reference axis side and axis of symmetry (if exists). Vector sum process, as described in section 2.12, is used to generate the pattern. The final pattern vector consists of total 24 arguments.

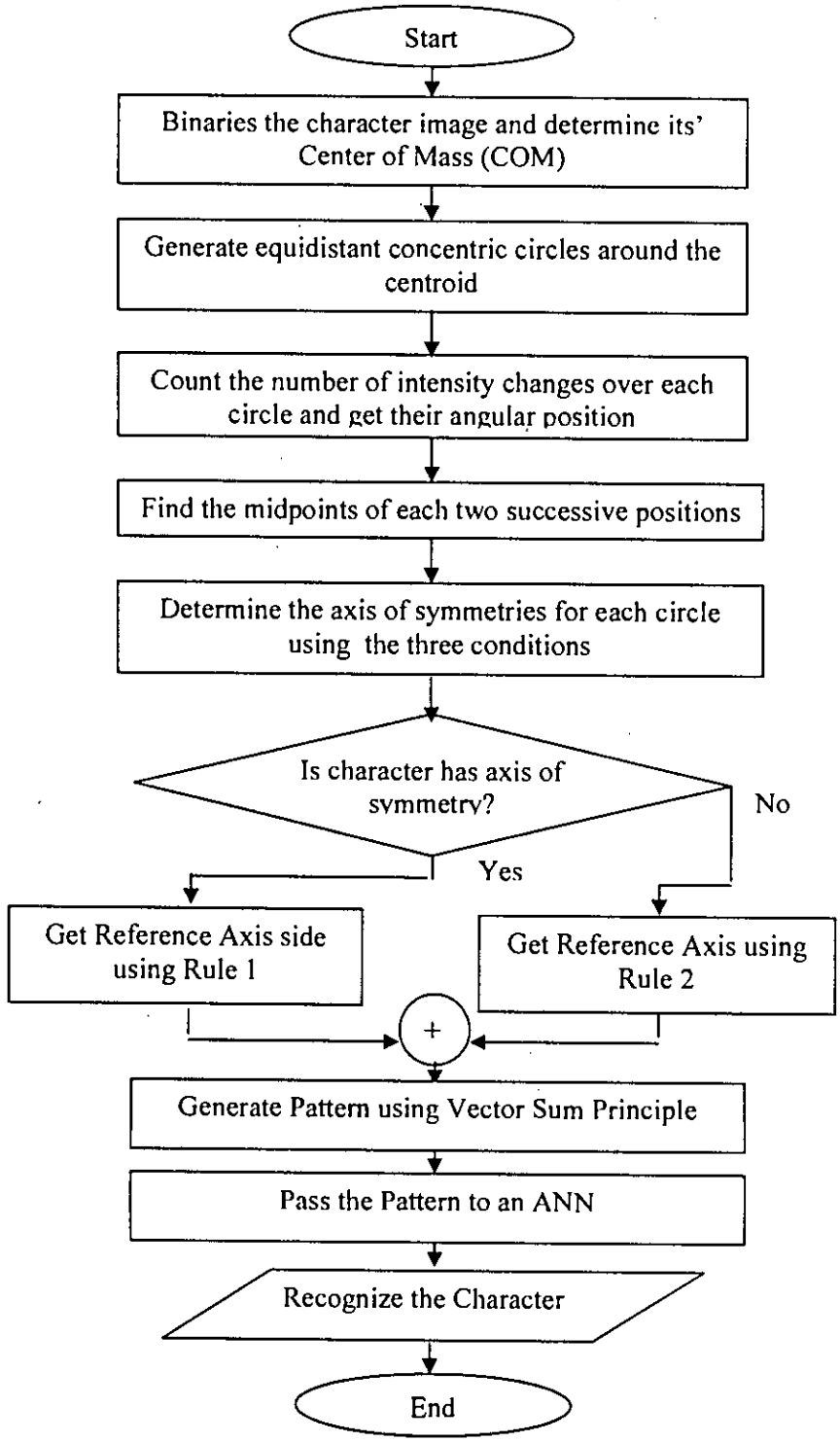


Fig. 3.1 Flow chart of the SAFER Algorithm

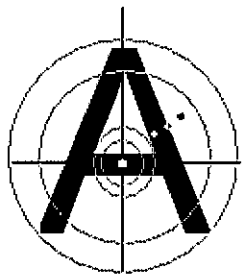


Fig. 3.2 K equidistance concentric circle on character 'A'

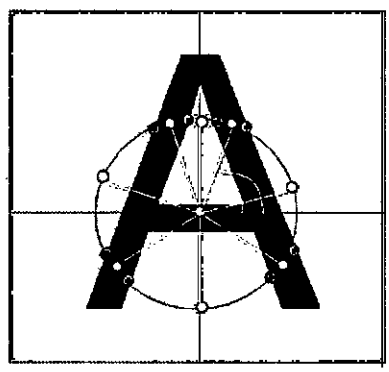


Fig. 3.3 Intensity changes on a concentric circle

Step 8: Train a three layer feedforward artificial neural network (ANN) by using the pattern vector until a minimum error criterion is met. The ANN is trained using random weight based cascade correlation (RWCC) algorithm.

The SAFER uses a very simple method to extract invariant features of the character. This is based on the fact that the circle is the only geometrical shape that is naturally and perfectly invariant to rotation. Higher order vector sum is used to generate pattern that efficiently describe both local and global features of the character. Detailed about axis of symmetry determination, reference axis selection, some correction factors and RWCC algorithm is described in the following sections.

3.2.1 Axis of Symmetry

SAFER uses a simple approach to determine the axis of symmetry, which is based on the radial vector R_i. Each two points of R_i might be on axis of symmetry if they satisfy the following three conditions.

- They are about 180 degree or 3.141 radian apart.
- There are equal numbers of points on each side of the axis.
- Each point on one side of the axis creates a pair with another point that resides in the opposite side and their angular distances relative to the axis are equal. (Fig. 3.4).

Since the radial vector R_i was determined by SAFER for each concentric circle, therefore, the possible axis of symmetries needs to find out for each circle. The final axis of symmetry is the line that exists in all concentric circles (Fig. 3.5). If a character has more than one axis of symmetry, SAFER selects the axis that has the highest neighbor pixels inside the character area. In Fig. 3.6 character 'H' has two axis of symmetry but horizontal axis has more neighbor pixel than vertical axis. So, horizontal axis is selected as the final axis of symmetry.

Although the determination of axis of symmetry using SAFER is quite simple, the algorithm is, however, failed to detect the axis in some cases. It is due to round up error, boundary noise and random noise problems. Some correction factors are, therefore, used in SAFER to overcome these problems.

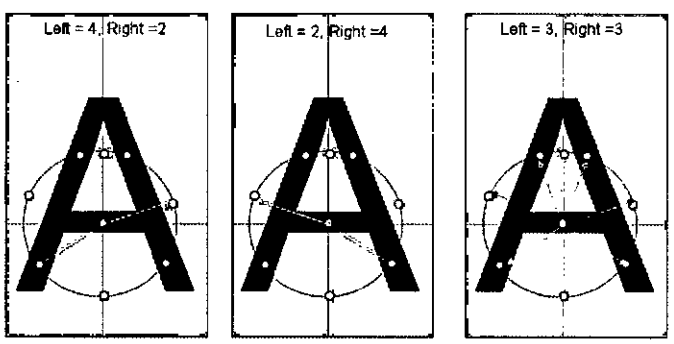


Fig. 3.4 Possible Axis of symmetry.

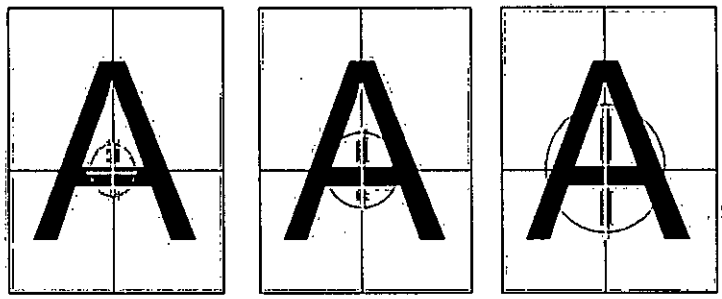


Fig. 3.5 The final Axis of Symmetry (Vertical Line)

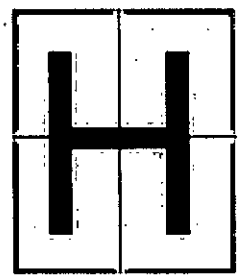


Fig. 3.6 Axis of Symmetry of character 'H'

3.2.2 Reference axis side

SAFER uses some simple criteria to detect the reference axis side i.e. 0^0 line, which is based on the angular distance of each two successive points of R_i. Depending on the existence of symmetry axis of a character, SAFER uses two different approaches to determine the reference side.

- **Case I**

SAFER uses this process for symmetrical character. Only one symmetry side of the character is used for calculation. For example, in Fig. 3.7, only left side of character 'A' is used. Calculate angular distances of the nearest cut point from both sides of the symmetry axis. The side, which has higher angular distance to the nearest cut point, is the reference side. In Fig. 3.7 the two angular distances are 1.047 and 0.349 radian. So, the lower side is selected as reference side.

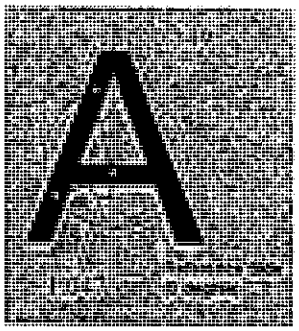


Fig. 3.7 Reference Axis side for symmetry character 'A'

- **Case II**

This process is used for asymmetrical character. At first, two adjacent cut points are selected from the radial vector R_i, which have maximum angular distance. In Fig. 3.8 line 1 and line 2 are at the highest angular distance. Then, from each line, the angular distance of the nearest cut point is calculated. The side, which has higher angular distance to the nearest cut point, is the reference axis. In Fig. 3.8, line 1 is selected as the reference axis.

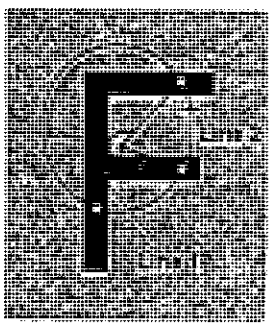
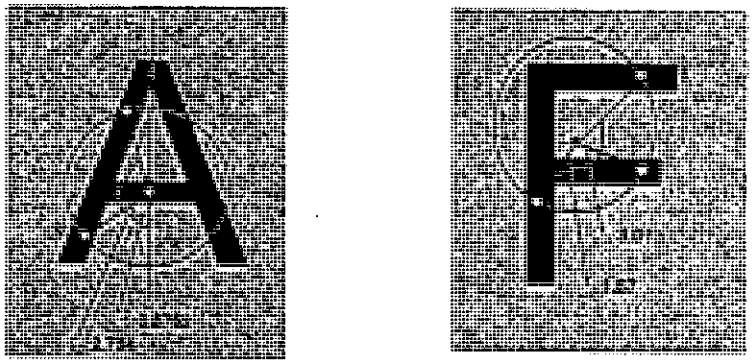


Fig. 3.8 Reference axis for asymmetry character 'F'

3.2.3 Pattern Vector

SAFER uses radial vector R_i and reference axis side to generate pattern vector. For symmetry character, SAFER calculates the relative angular positions in radian of all points on any one symmetry side (Fig. 3.9(a)) by using the reference axis side. If character has no axis of symmetry, SAFER calculates relative angular position of all points expressed in radian (Fig. 3.9(b)).



(a) (b)

Fig. 3.9 Relative angular positions for character 'A' and 'F'

The arguments of the Pattern vector are generated from each concentric circle. The vector for r^{th} circle can be calculated by using Eq. (3.1) and (3.2).

$$V_1(r) = \frac{1}{N}\sum_{a=1}^{N} e^{i \times ang_a} \tag{3.1}$$

$$V_2(r) = \frac{1}{N}\sum_{a=1}^{N} e^{i \times 2 \times ang_a} \tag{3.2}$$

Where, N is the total number of relative angular positions and ang_a is the a^{th} relative angular position. To normalize the values of Eq. (3.1) and (3.2) in the interval {0, 1}, they are rewritten as

$$VN_1(r) = ((1+i) + \frac{1}{N}\sum_{a=1}^{N} e^{i \times ang_a})/2 \tag{3.3}$$

$$VN_2(r) = ((1+i) + \frac{1}{N}\sum_{a=1}^{N} e^{i \times 2 \times ang_a})/2 \tag{3.4}$$

For example, in Fig. 3.9(b), for $r=3$ 'F' has $N = 2$. Now using Eq.(3.3) and (3.4)

$$VN_1(3) = ((1+i) + \frac{1}{2}(e^{i \times 1.57} + e^{i \times 3.01}))/2 = 0.2524 + 0.7828i$$

$$VN_2(3) = ((1+i) + \frac{1}{2}(e^{i \times 2 \times 1.57} + e^{i \times 2 \times 3.01}))/2 = 0.4914 + 0.4354i$$

Finally, the pattern consists of total 12 vectors for 6 circles

$$VN_1(1), VN_1(2), ... VN_1(6), VN_2(1), VN_2(2) ... VN_2(6) \tag{3.5}$$

Since each vector consist both real and imaginary part so pattern is formed by 24 arguments.

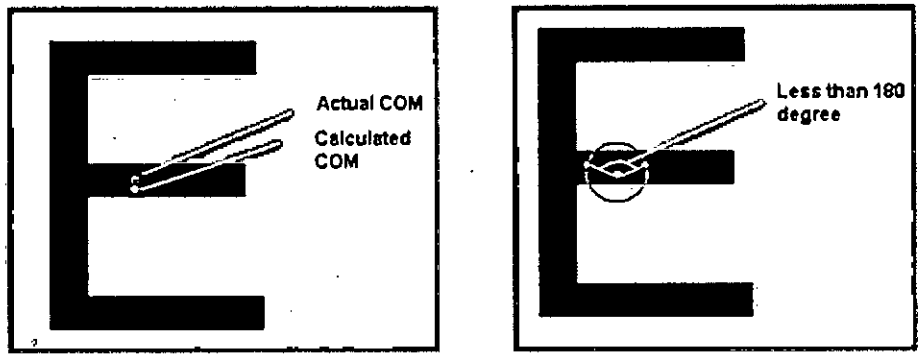
3.2.4 Correction

SAFER uses a simple approach to detect axis of symmetry of a character, which is very effective in low noisy environment. However, some correction is required if one wishes to detect character in more noisy environment. The required corrections divide in three categories. These are round up error, boundary noise and random noise. In the following subsections they are described elaborately.

3.2.4.1 Round up Error Correction

Round up error occurs at the time of calculating Center of Mass (COM) by using Eq.2.22 and 2.23. The equations can result a fraction COM, but the image coordinate is always integer. SAFER, therefore, rounds up the value of COM to make it integer. Again, due to some noise the COM may vary for same character in different environment. In both cases, SAFER fails to detect the actual COM.

The first condition for two points to be on the axis of symmetry was that they must be 180^0 apart from each other. This condition may fail due to defective COM. For example, in Fig. 3.10(a) the calculated COM is one pixel below the original COM. Consider the circle with radius of five pixels Fig. 3.10(b). The midpoint of the arc in character area is one pixel up with respect to the horizontal axis passing through the calculated COM. As a result, the two pixels are $180^0 \pm 22.6^0$ ($2\times \tan^{-1}(1/5)$) apart. So they failed to fulfill the first condition. Although they are on axis of symmetry, the program fails to recognize it.



(a) (b)

Fig. 3.10 Round up Error

To correct the round up error a round up error factor e_{rnd} is introduced. Round up error factor is the measure of given flexibility in terms of angle with respect to radius. Round up error factor is defined as.

$$e_{rnd}(r) = 2 \times \tan^{-1}(\frac{p}{r}) \tag{3.6}$$

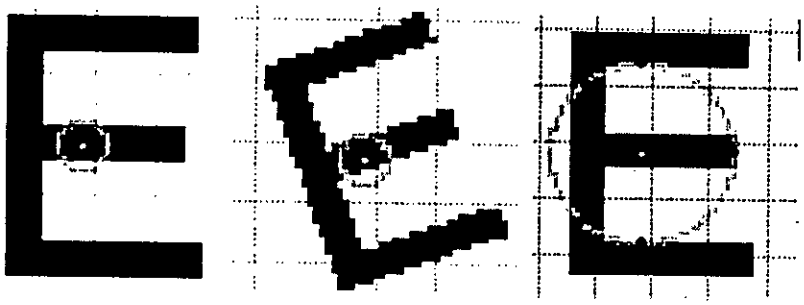
Where, r is the radius of circle in pixels and p is the number of flexible pixel. For example, to give one pixel flexibility at a radius of five pixels, the $e_{rnd}(5)$ is 22.6^0. The factor varies proportionally with radius. For example, at a distance r_l, e_{rnd} become

$$e_{rnd}(r_l) = \frac{e_{rnd}(5) \times 5}{r_l} \tag{3.7}$$

Therefore, at a radial distance r_l, if the angular difference between two points is $180 \pm e_{rnd}(r_l)$ degree then they may consider on the axis of symmetry.

3.2.4.2 Boundary Noise Correction

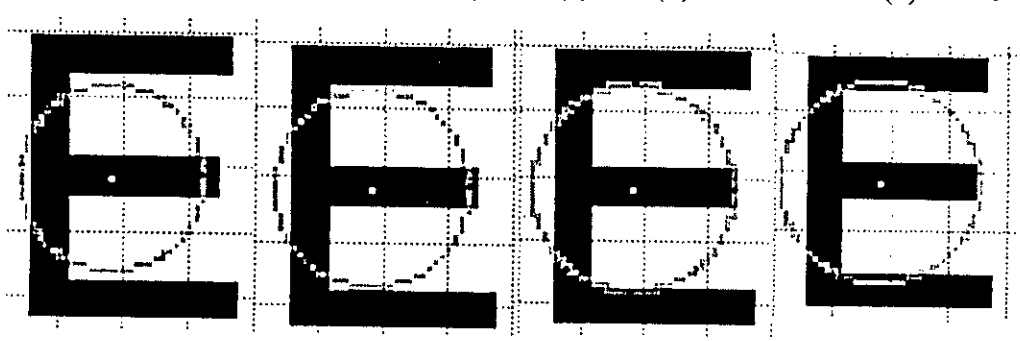
Boundary noise can generate a serious problem in SAFER. This type of noise can be produced by two ways. Firstly, it can be generated by the rough surface. The rough surface may intrinsically produce during the different rotations of character (Fig. 3.11 (b)). This type of noise can also produce when concentric circle touches the boundary of character (Fig. 3.11(c)). This noise produces wrong cut points and generates incorrect radial code.



(a) (b) (c)

Fig. 3.11 Boundary Noise

The error can be corrected by checking the stability of c circle. A circle is stable if the number of cut points over a circle equal to the number of cut points on previous or next circle. If a circle is unstable then the nearest circle is selected which is stable. In Fig. 3.12 (a), the circle's radius is 15 pixels and number of cut points is 3 in (b) the radius is 16 pixels and number of cut points is 3. However, in (c) the radius is 17 pixels and number of cut points is 5. So circles of Fig. 3.12(a) and (b) are stable but (c) is unstable.

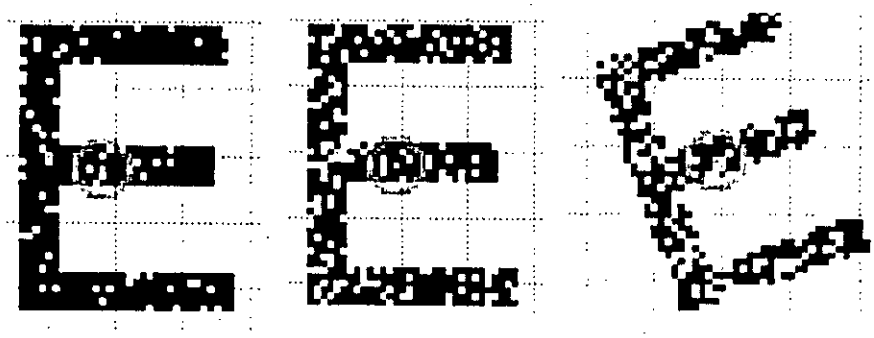


(a) (b) (c) (d)

Fig. 3.12 Stability of circle

3.2.4.3 Random Noise Correction

The random noise is generated by changing the value of pixels inside the character in a random manner. Fig. 3.13 shows different amount of random noise inside character 'E'



(a) (b) (c)

Fig. 3.13 Random Noise (a) 10% random noise (b) 20% random noise (c) 30% random noise

The random noise divides the contiguous character area and generates multiple cut points. For example, in Fig. 3.14 (a), due to random error, two extra cut points are generated in 'E'. A random noise factor e_{rn}, similar to Eq. (3.6) and (3.7), is introduced to overcome the problem. If angular difference between two adjacent non-character boundary reasons is less than e_{rn}, then they are considered as contiguous character area. In Fig 3.14 (b) each of the two non-character boundaries has angular difference less than e_{rn}, so each of them are considered on contiguous character area.

(a) (b) (c)

Fig. 3.14 Random error and its correction

3.2.5 Random Weight based Cascade Correlation (RWCC) algorithm

In this thesis, an efficient classifier was designed for invariant character recognition. The classifier was a three layer feedforward ANN whose architecture was created by using RWCC algorithm. RWCC combines two key ideas: The first is the

cascade architecture, in which hidden units are added to the network one or two at a time and their weights were initiated randomly. The second is the learning algorithm, which creates and installs the new hidden units. For each new hidden unit, the magnitude of the correlation between the new unit's output and the residual error signal were calculated. The flowchart of RWCC is shown in Fig. 3.15. The algorithm is described in the following steps.

Step 1: Create an initial ANN architecture as shown in Fig. 3.16. The initial architecture has three layers, i.e. an input layer, an output layer, and a hidden layer. The number of nodes in the input and output layers is the same the number of inputs and outputs of the problem. Initially, the hidden layer contains only four nodes. Randomly initialize connection weights between input layer to hidden layer and hidden layer to output layer within a certain range. There is also a bias input, which is permanently set to +1.

Step 2: Train the network with the training set for a certain number of epochs by using Backpropagation learning algorithm. The error value of the network is checked at each epoch. If error has not been significantly reduced i.e. error change is less than a predefined parameter ε_1, then the assumption is that the network is saturated. If the network's performance is satisfactory, stop the training process. Otherwise, go to the next step.

Step 3: Compute the average error change ε_{av}. If ε_{av} is less than a predefined value ε_2, two new hidden neurons need to add. Otherwise, one new neuron needs to add. In SAFER, ε_{av} is defined by

$$\varepsilon_{av} = \frac{\varepsilon_{st} - \varepsilon_1}{ep} \tag{3.8}$$

Where, ε_{st} is the error change just after adding new the hidden nodes and ε_1 is the error change when the training is stopped. *ep* is the total epochs needed to train the network after adding the new hidden neuron.

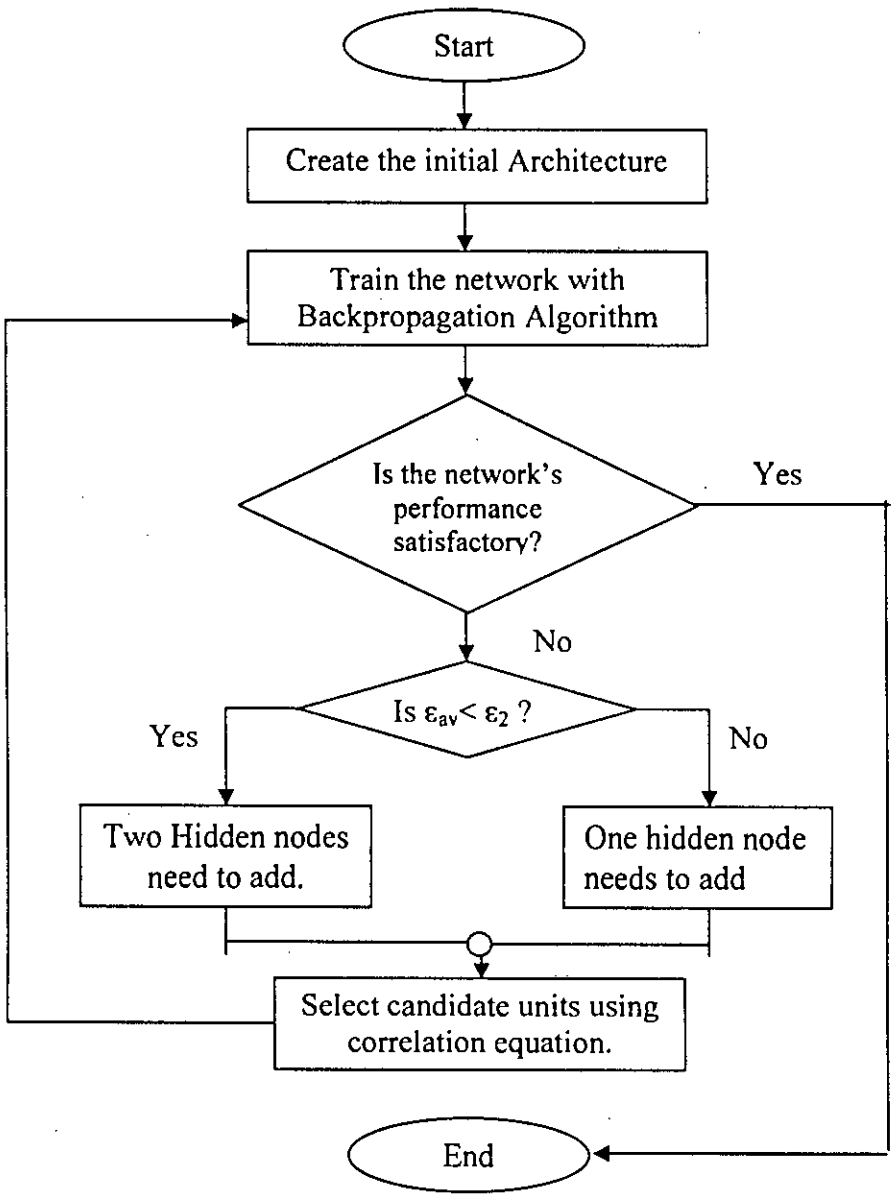


Fig. 3.15 Flow chart of RWCC algorithm

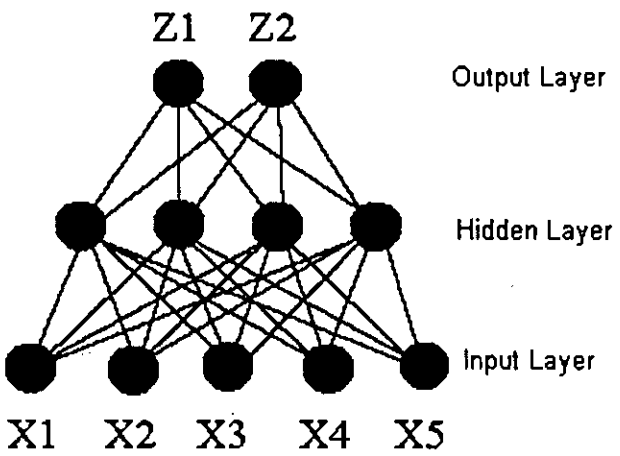


Fig. 3.16 A Three Layer Feedforward ANN

Step 4: To add a new hidden unit, four *candidate units* need to create. The new nodes receive random trainable input connections from all of the network's external inputs. The outputs of these candidate units are not yet connected to the active network. Calculate the correlation S [23] of each candidate units to output. Select the candidate, which has maximum S. The correlation S is defined as

$$S = \sum_{o} \left| \sum_{p} (V_p - \overline{V})(E_{p,o} - \overline{E}_o) \right| \tag{3.9}$$

Where o is the network output at which the error is measured and p is the training pattern. The quantities V and E_o are the values of V and E_o averaged over all patterns.

To add two new hidden units, eight *candidate units* need to create. Two candidate units are selected by using the described process.

Connect the selected units to all of the output units that receive random trainable connection weights. Go to Step 2.

The RWCC is a modified version of the cascade correlation based architecture determination algorithm [23]. It performs very well in the ICR algorithm, which can determine and train the ANN architecture in a very sort training time.

Chapter 4

Experimental Evaluation

4.1 Introduction

This chapter evaluates the performance of SAFER on two well-known fonts Arial and Tahoma widely used for representing English characters. These fonts have been subject of many studies in ICR. Initially, two dimensional (2-D) binary character images were considered and tested the algorithm for invariant-character recognition. Although the algorithm was applied only for 2-D character image, it could be extended easily for multilevel images. Experimental details, results, analysis and comparisons with other works are described in this chapter. The reasons for the improved performance SAFER in comparisons with existing works are also described in this chapter.

4.2 Data Set Description

The data sets used here were generated from Arial and Tahoma fonts with different orientations and scaling. In this study, all data sets representing the problems were divided into two sets. One was the training set and the other was the test set. No validation set was used in this study. To compare this work with other work, the number of examples in the training set and testing set were selected based on numbers in similar works.

Microsoft Visio Professional 2002 (10.0.525) software was used to create different orientations and scaling of characters. The orientation and scaling of character was changed using the Text Block tool. The text can be rotated by using three steps: (i) first, select the Text Block tool from the Standard toolbar; (ii) second, click the text in a shape or independent text that wants to rotate; (iii) finally, drag a rotation handle until the

text was rotated the wanted way. A character can be rotated at any desired angle by this way. The detailed descriptions of training and test sets are given in the following subsections.

4.2.1 Training Data Set

There were two different training sets, one for Arial font and one for Tahoma font, are used for training ANN classifiers. The font size of both training sets was 40 × 40 pixels. In order to achieve increased noise tolerance, four different orientations of each English character was considered in the training set. Thus, the training set of each font was consisted of 104 (26 × 4) input patterns for 26 different English characters. Each training pattern was consisted of 24 positive numbers, which was generated by the radial coding of six circles described in section 3.2.3.

4.2.2 Testing Data Set

Testing data set was used to evaluate the performance of SAFER. After training the ANN, the testing dataset was propagated through ANNs to determine the recognition rate. The testing dataset was consisted of total 7920 patterns, which were created from two different fonts with different orientations and scaling.

Nine different sized fonts were used for each character in the testing set. The largest font size was 70 ×70 pixels and the smallest was 20×20 pixels. Seventy two different rotations (5^0 apart) were used for the font size 40 × 40 pixels. Thirty six different rotations (10^0 apart) were used for the other 8 different font sizes.

4.3 Experimental Setup

The initial ANN was constructed with 24 input, 4 hidden and 26 output units. In all experiments, one bias node with a fixed input +1 is connected to the hidden layer. The learning rate η is set between [0.01, 0.04] and the weights were initialized to random values between [-0.5, +0.5]. The average error change ε_{av} and minimum error change ε_t were set to 0.00075 and 0.0001, respectively. A hyperbolic tangent function is taken as

the activation function— $f = a\tanh(bv)$, where, v is the output of the summing junction, a = 1.7159, b = 2/3. The output neurons act in "winner takes all" manner.

4.3.1 Experimental Results

Tables 4.1-4.9 show the results of SAFER over two fonts of nine different sized characters. The title 'pattern' in the table refers to the total number of patterns used for testing. Success and fail represent the number of successfully recognized patterns and unrecognized patterns out of the total patterns, respectively. The success rate in the tables refers to the percentage of correct classification produced by the trained ANN on the testing set. The total rate represents the overall performance, which is the average of 26 individual success rates.

Table 4.1 shows the effect of translation on the performance of SAFER for different characters. The ANN was trained by using 26 Arial patterns (40×40) located in the center. Nine different transformation of 40×40 pixels Arial font were used for testing. It is seen that there is no effect of translation on the performance of SAFER. This means SAFER achieved 100% recognition rate for all translated characters.

The effect of rotation on the performance of SAFER is shown in Table 4.2. It is clear from the table that the performance of SAFER is dependent on rotational variations for some characters. For example, SAFER achieves a recognition rate of 95.83% and 97.22% for character 'C' and 'G'. The recognition rate achieved by SAFER for character 'D' and 'O' were 98.61% and 94.44% respectively. The reason for the low performance of SAFER for some characters might be due to the similarity between characters. For example, there are some similarities between characters 'O' and 'D' and 'W' and 'M'.

Table 4.3 shows the effect of scaling on the performance of SAFER for different characters. It is found from the table that there is almost no effect of scaling on the performance of SAFER for large sized characters. However, the performance of SAFER decreases for small sized characters. For example, SAFER achieve an overall recognition rate of 88.46%, 96.15% and 100% for font size 20×20, 30×30 and 50×50 pixels, respectively. The reason of the bad performance of SAFER for small sized characters might be due to that they affect much by noise. For example, when SAFER generates

patterns for 40 × 40 and 20 × 20 pixels 'C' character, due to the effect of noise, 20 × 20 pixels pattern may produce high deviation with 40 × 40 pixels patterns. As a result, when ANN trained with 40 × 40 pixels 'C' pattern and tested with 20 × 20 pixels pattern, the ANN failed to recognize it.

Tables 4.4 and 4.5 show the effect of both scaling and rotation of SAFER for different characters of Arial fonts. It is clear from the tables that the performance of SAFER is dependent on both the size and rotational variations for some characters. For example, SAFER achieves a recognition rate of 94.44% and 88.89% for character 'C' of size 60×60 and 20×20 pixels, respectively. The tables also show that as the characters size increases their overall recognition rate increase from 95.51% to 99.57%. Fig 4.1 summarizes the overall performances of 9 different sized and rotated characters in a bar chart. The overall performance of SAFER is almost 99.6% for the largest characters.

Tables 4.6 and 4.7 show the effect of both scaling and rotation on the performance of SAFER for different characters of Tahoma font. Similar to Arial fonts the performance of SAFER is dependent on both size and rotation variation for some characters. It is found that some types of characters are recognized well in Tahoma compared to those in Arial. However, SAFER failed to perform better for some other characters in Tahoma that were recognized better in Arial. For example, in Table 4.2 and 4.6, SAFER achieves a recognition rate of 100% and 97.22% for character 'A' of Arial and Tahoma fonts, respectively. However, it is 98.6% and 100% for character 'Q'. The performance variation for some characters might be due to their higher discrimination features in one font with respect to the other font. So, the pattern vector produced for a character has higher interpattern dissimilarity for one font where it has higher interpattern correlation for the other font. For example, pattern vector of character 'A' in Arial font may produce high dissimilarity with pattern vectors of other characters in the same font. So, character 'A' in Arial font can be easily recognized by SAFER. However, the pattern vector of character 'A' in Tahoma font may produce low dissimilarity with other pattern vectors of other characters. So, SAFER can't easily recognize character 'A' in Tahoma font.

Table 4.8 shows the generalization ability of SAFER. To analyze the generalization ability, the ANN classifier of SAFER was trained with 40×40 pixels Arial fonts and tested with 40×40 pixels Tahoma fonts. Although the table shows that the

performance is pretty good, however, it decreases for some characters. For example, SAFER achieves poor recognition ratc of 88.8%, 72.22% and 88.8 % for character 'C', 'M' and 'P' of Tahoma font, respectively. The reason of the low performance by SAFER might be that same character produced different patterns for two fonts and when the ANN trained with Arial fonts pattern and tested with Tahoma fonts pattern, it failed to recognize Tahoma patterns.

In order to show the robustness of SAFER, a different amount of noise was added in the characters of testing data set. The noise was generated by randomly changing the value of pixels inside the character. The percentage of altered pixels was varied from 10% to 50%. Table 4.9 shows the results of the simulation. It is found that SAFER achieves an overall recognition rate of 97.22% for characters having 20% noise. The effect of noise is much some character like 'C', 'M' and 'P', where the recognition rate was 93%.

Table 4.1 ANN trained with 40 × 40 Arial and tested with translated 40 × 40 Arial (Total 26 training patterns were used, which were generated from 26 centered 40 × 40 Arial characters. 9 tested patterns of each 40 × 40 Arial character were generated from 9 different translation (left, right, up, down and so on))

Letter	Pattern	Success	Fail	Rate
A	9	9	0	100
B	9	9	0	100
C	9	9	0	100
D	9	9	0	100
E	9	9	0	100
F	9	9	0	100
G	9	9	0	100
H	9	9	0	100
I	9	9	0	100
J	9	9	0	100
K	9	9	0	100
L	9	9	0	100
M	9	9	0	100
N	9	9	0	100
O	9	9	0	100
P	9	9	0	100
Q	9	9	0	100

R	9	9	0	100
S	9	9	0	100
T	9	9	0	100
U	9	9	0	100
V	9	9	0	100
W	9	9	0	100
X	9	9	0	100
Y	9	9	0	100
Z	9	9	0	100
Total	**234**	**234**		**100**

Table 4.2 ANN trained with 40 × 40 Arial fonts and tested with 40 × 40 Arial Font (Total 104 (26 × 4) training patterns were used, which were generated from 4 random orientation of each 40 × 40 Arial character. 72 tested patterns of each 40 × 40 Arial character were generated from 72 different orientations and they were 5^0 apart from each other)

Letter	Pattern	Success	Fail	Rate
A	72	72	0	100
B	72	72	0	100
C	72	69	3	95.83333
D	72	71	1	98.61111
E	72	71	1	98.61111
F	72	72	0	100
G	72	70	2	97.22222
H	72	72	0	100
I	72	72	0	100
J	72	72	0	100
K	72	72	0	100
L	72	72	0	100
M	72	72	0	100
N	72	70	2	97.22222
O	72	68	4	94.44444
P	72	67	5	93.05556
Q	72	71	1	98.61111
R	72	69	3	95.83333
S	72	72	0	100
T	72	72	0	100
U	72	72	0	100
V	72	72	0	100
W	72	72	0	100
X	72	72	0	100
Y				
	72	72	0	100

Z	72	68	4	94.44444
Total	**1872**	**1846**	**26**	**98.611**

Table 4.3 ANN trained with 40 × 40 Arial and tested with different sized Arial character (Total 26 training patterns were used, which were generated from 26 centered 40 × 40 Arial characters. 26 patterns of each scaled font were generated from 26 centered Arial characters)

Pattern Size	Pattern	Success	Fail	Rate
20 × 20	26	23	3	88.46154
25 × 25	26	25	1	96.15385
30 × 30	26	25	1	96.15385
35 × 35	26	26	0	100
40 × 40	26	26	0	100
45 × 45	26	26	0	100
50 × 50	26	26	0	100
60 × 60	26	26	0	100
70 × 70	26	26	0	100

Table 4.4 ANN trained with 40 × 40 Arial fonts and tested with 60 × 60 Arial Font (Total 104 (26 × 4) training patterns were used, which were generated from 4 random orientation of each 40 × 40 Arial character. 36 tested patterns of each 60 × 60 Arial character were generated from 36 different orientations and they were 10^0 apart from each other)

Letter	Pattern	Success	Fail	Rate
A	36	36	0	100
B	36	36	0	100
C	36	34	2	94.44444
D	36	36	0	100
E	36	36	0	100
F	36	36	0	100
G	36	36	0	100
H	36	36	0	100
I	36	36	0	100
J	36	36	0	100
K	36	36	0	100
L	36	36	0	100
M	36	36	0	100
N	36	36	0	100

O	36	35	1	97.22222
P	36	36	0	100
Q	36	36	0	100
R	36	36	0	100
S	36	36	0	100
T	36	36	0	100
U	36	35	1	97.22222
V	36	36	0	100
W	36	36	0	100
X	36	36	0	100
Y	36	36	0	100
Z	36	36	0	100
Total	**936**	**932**	**3**	**99.5765**

Table 4.5 ANN trained with 40 × 40 Arial fonts and tested with 20 × 20 Arial Font (Total 104 (26 × 4) training patterns were used, which were generated from 4 random orientation of each 40 × 40 Arial character. 36 tested patterns of each 20 × 20 Arial character were generated from 36 different orientations and they were 10^0 apart from each other)

Letter	Pattern	Success	Fail	Rate
A	36	36	0	100
B	36	36	0	100
C	36	32	4	88.88889
D	36	34	2	94.44444
E	36	35	1	97.22222
F	36	34	2	94.44444
G	36	34	2	94.44444
H	36	36	0	100
I	36	36	0	100
J	36	36	0	100
K	36	36	0	100
L	36	36	0	100
M	36	34	2	94.44444
N	36	35	1	97.22222
O	36	35	1	97.22222
P	36	29	7	80.55556
Q	36	34	2	94.44444
R	36	31	5	86.11111
S	36	36	0	100
T	36	36	0	100

U	36	34	2	94.44444
V	36	36	0	100
W	36	27	9	75
X	36	36	0	100
Y	36	36	0	100
Z	36	34	2	94.44444
Total	**936**	**894**	**42**	**95.5128**

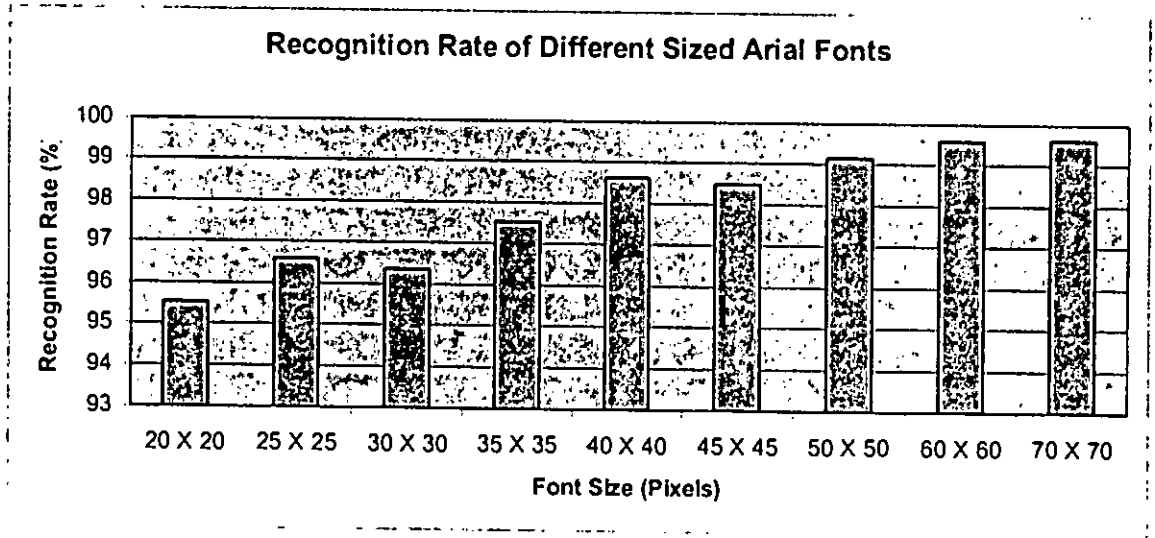


Fig. 4.1 Bar chart for Recognition rate of different sized Arial Fonts

Table 4.6 ANN trained with 40 × 40 Tahoma fonts and tested with 40 × 40 Tahoma Font (Total 104 (26 × 4) training patterns were used, which were generated from 4 random orientation of each 40 × 40 Arial character. 72 tested patterns of each 40 × 40 Tahoma character were generated from 72 different orientations and they were 5^0 apart from each other)

Letter	Pattern	Success	Fail	Rate
A	72	70	2	97.22222
B	72	72	0	100
C	72	69	3	95.83333
D	72	68	4	94.44444
E	72	68	4	94.44444
F	72	72	0	100
G	72	71	1	98.61111
H	72	72	0	100
I	72	72	0	100
J	72	72	0	100
K	72	69	3	95.83333
L	72	71	1	98.61111

M	72	72	0	100
N	72	72	0	100
O	72	68	4	94.44444
P	72	70	2	97.22222
Q	72	72	0	100
R	72	70	2	97.22222
S	72	68	4	94.44444
T	72	72	0	100
U	72	72	0	100
V	72	72	0	100
W	72	70	2	97.22222
X	72	72	0	100
Y	72	72	0	100
Z	72	68	4	94.44444
Total	**1872**	**1836**	**36**	**98.07692**

Table 4.7 ANN trained with 40 × 40 Arial and Tahoma fonts and tested with 40 × 40 Arial Font

(Total 208 (26 × 4 × 2) training patterns were used, which were generated from 4 random orientation of each 40 × 40 Arial character and 4 random orientation of each 40 × 40 Tahoma character. 72 tested patterns of each 40 × 40 Arial character were generated from 72 different orientations and they were 5^0 apart from each other)

Letter	Pattern	Success	Fail	Ratc
A	72	70	2	97.22222
B	72	69	3	95.83333
C	72	69	3	95.83333
D	72	68	4	94.44444
E	72	71	1	98.61111
F	72	72	0	100
G	72	67	5	93.05556
H	72	72	0	100
I	72	72	0	100
J	72	72	0	100
K	72	71	1	98.61111
L	72	72	0	100
M	72	72	0	100
N	72	64	8	88.88889
O	72	67	5	93.05556
P	72	67	5	93.05556
Q	72	70	2	97.22222
R	72	72	0	100

S	72	70	2	97.22222
T	72	72	0	100
U	72	66	6	91.66667
V	72	72	0	100
W	72	72	0	100
X	72	72	0	100
Y	72	72	0	100
Z	72	68	4	94.44444
Total	**1872**	**1821**		**97.27564**

Table 4.8 ANN trained with 40 × 40 Arial fonts and tested with 40 × 40 Tahoma Font (Total 104 (26 × 4) training patterns were used, which were generated from 4 random orientation of each 40 × 40 Arial character. 36 tested patterns of each 40 × 40 Tahoma character were generated from 36 different orientations and they were 10^0 apart from each other)

Letter	Pattern	Success	Fail	Rate
A	36	35	1	97.22222
B	36	36	0	100
C	36	32	4	88.88889
D	36	34	2	94.44444
E	36	34	2	94.44444
F	36	36	0	100
G	36	33	3	91.66667
H	36	36	0	100
I	36	36	0	100
J	36	36	0	100
K	36	36	0	100
L	36	35	1	97.22222
M	36	26	10	72.22222
N	36	36	0	100
O	36	33	3	91.66667
P	36	32	4	88.88889
Q	36	36	0	100
R	36	34	2	94.44444
S	36	33	3	91.66667
T	36	36	0	100
U	36	36	0	100
V	36	36	0	100
W	36	36	0	100
X	36	36	0	100
Y	36	36	0	100
Z	36	34	2	94.44444

Total	936	899	37	96.04701

Table 4.9 ANN trained with 40 × 40 Arial fonts and tested with 40 × 40 Arial Font with 20% random noise

(Total 104 (26 × 4) training patterns were used, which were generated from 4 random orientation of each 40 × 40 ideal Arial character. 72 tested patterns of each 40 × 40 noisy Arial character were generated from 72 different orientations and they were 5^0 apart from each other)

Letter	Pattern	Success	Fail	Rate
A	72	72	0	100
B	72	70	2	97.22222
C	72	67	5	93.05556
D	72	69	3	95.83333
E	72	69	3	95.83333
F	72	72	0	100
G	72	70	2	97.22222
H	72	72	0	100
I	72	72	0	100
J	72	69	3	95.83333
K	72	70	2	97.22222
L	72	72	0	100
M	72	67	5	93.05556
N	72	70	2	97.22222
O	72	68	4	94.44444
P	72	67	5	93.05556
Q	72	71	1	98.61111
R	72	69	3	95.83333
S	72	70	2	97.22222
T	72	72	0	100
U	72	70	2	97.22222
V	72	72	0	100
W	72	69	3	95.83333
X	72	72	0	100
Y	72	71	1	98.61111
Z	72	68	4	94.44444
Total	**1872**	**1820**	**52**	**97.2222**

To observe the training process of the ANN classifier used in SAFER, Fig. 4.2 and 4.3 show how the mean square error of the network decreases with training epochs for different fonts. It is seen from these figures that the error decreased gradually as the

training process progressed. However, the error was decreasing rapidly in the first few training cycles. At some stage in the training process, the error becomes flat. For example, the error was flat at 40 epochs for Arial font. This indicates that the classifier does not have sufficient capability to handle the training process. RWCC therefore adds hidden neuron to the classifier for increasing its information processing capability. In figure 4.2, at 40 epochs the rate of changing error was 0.00066, which was lower than ε_2 but higher than ε_1. Therefore, one new hidden node was added by using RWCC.

One of the significant properties of RWCC is that it can automatically detect the training point at which the network needs to add more than one hidden neuron to accelerate the training process. If at some training point the error change become very low i.e. less than ε_1, then it adds two hidden neuron. In Fig. 4.2, at 140 epochs the rate of changing error became 0.000077, which was lower than both ε_1 and ε_2. Therefore, two new hidden nodes were added by RWCC in this step. It is noted from the figure that after adding new nodes at 140th epoch, the error was increased at first epoch. This was also true for 210th, 250th, 290th epoch and so on. The reason of this behavior might be due to the random initial weights of the new hidden nodes. For example, at 140th epoch the network had 8 hidden nodes. When two new hidden nodes were added with random initial weight they might produce large error and the overall error became high.

Table 4.10 shows a competitive study between RWCC based ANN and conventional fixed hidden node based ANN in terms of error minimization capability, training epoch and training time. It is clear from the table that the performance of RWCC based ANN is better than conventional ANN in terms of training time. Although the training epoch required to minimize error is higher in RWCC based ANN, the total training time was much smaller than that required for conventional ANN. The reason of this performance might be due to the gradually increasing hidden nodes in RWCC based ANN. In conventional ANN the numbers of initial hidden nodes were 23 and throughout the training process the network trained all the 23 nodes, which took much time. However, in RWCC based ANN the initial hidden node was 4. After training the 4 hidden nodes, new hidden nodes were added in different steps. For example, at 140th epoch the number of hidden nodes became 8. So the numbers of hidden nodes were

gradually increasing at different training epochs and until the last few epochs, the network was never trained all the 23 nodes at a time. Therefore, the training time is much lower than conventional training.

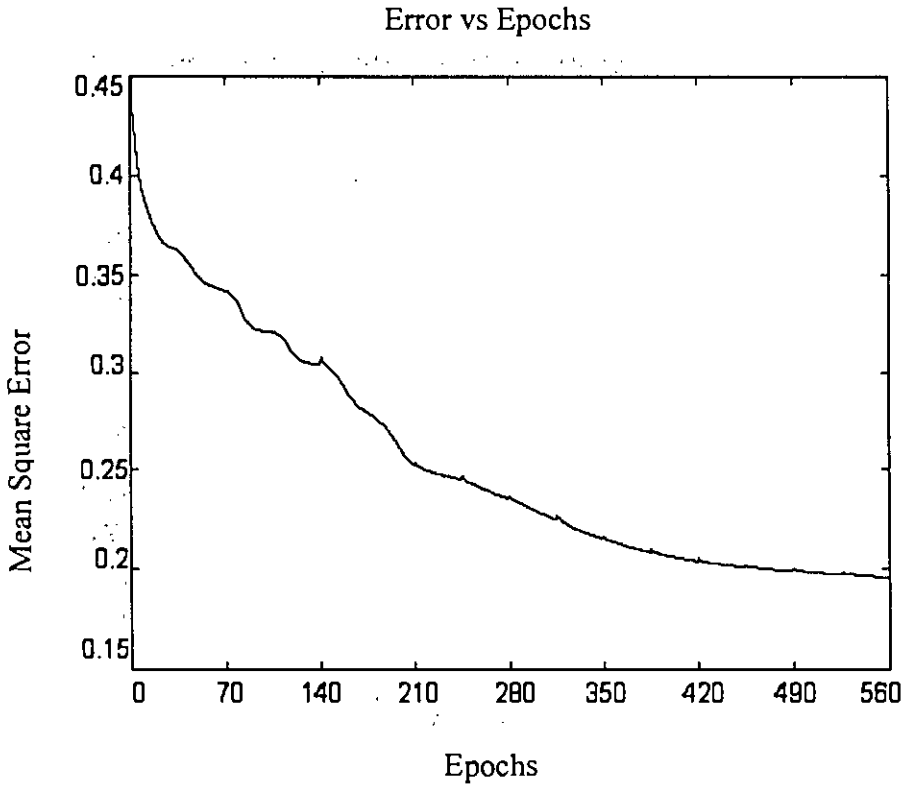


Fig. 4.2 Training process of SAFER for Arial font character

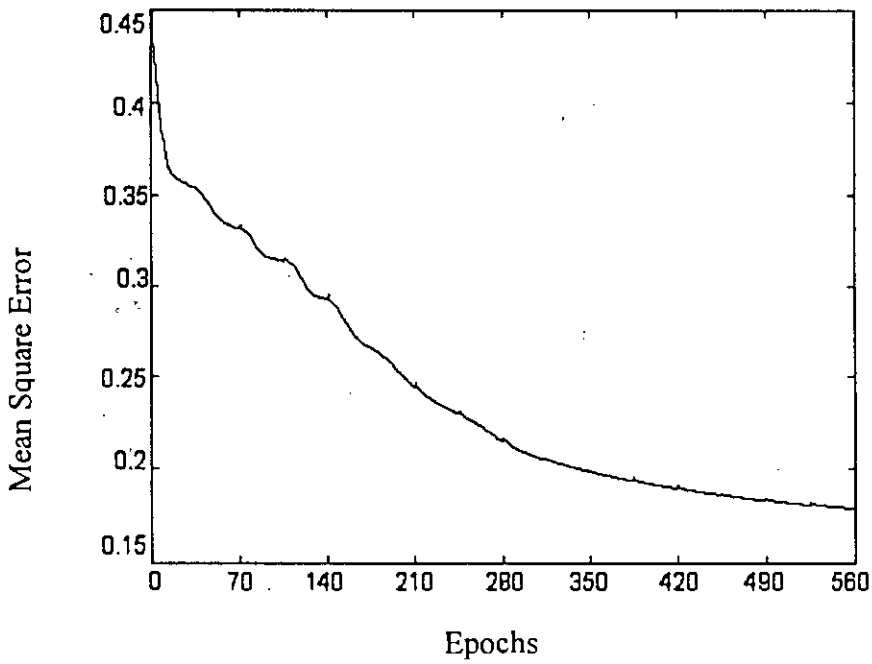


Fig. 4.3 Training process of SAFER for Tahoma font character

Table 4.10 Comparison between RWCC based ANN and conventional fixed hidden node based ANN in terms of final error minimization time and training epoch

Network	Initial hidden node	Final hidden node	Final error	Training Epoch	Training Time (min)
RWCC based ANN	4	23	0.1917	560	2.12
Fixed hidden node based ANN	23	23	0.1917	528	6.35

4.4 Analysis

This subsection analyzes SAFER's performance and limitations in order to gain a better understanding of the SAFER algorithm. The pattern vectors of individual characters are

used to analyze the results obtained for different characters. The recognition rate individual characters are used to determine the performance of SAFER.

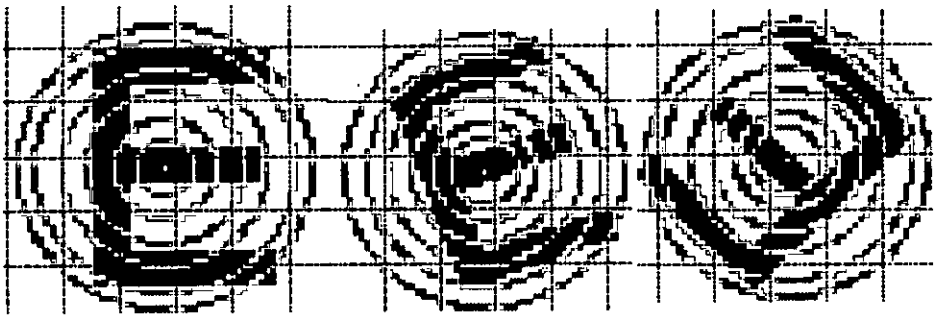
4.4.1 Extracted Patterns

The performance of SAFER depends on the property of pattern vectors that were generated from characters of different translation, rotation and scaling. Therefore, different pattern vectors are analyze here to gain a better understanding of the SAFER.

The effect of rotation on the pattern vector generated by SAFER is shown in Figs. 4.4 -4.6. . It is clear from the figures that the pattern vector is almost independent on rotational variations of characters. For example, it is found from Fig 4.6 that there are less than 5% deviations among patterns generated by SAFER for three different orientations of the character 'E'. The Fig. 4.4 shows that the character 'E' is rotated at 0^0 and 135^0 angles and Fig. 4.5 shows their pattern vectors. Fig 4.6 shows the standard deviation between the two pattern vectors. Although the deviation is very low but at 6^{th} and 18^{th} arguments of the pattern vectors the deviation are 10% and 21%, respectively. The high deviation may be due to the unequal cut points produced by the same concentric circle when one character rotate in different angles. For example, the 6^{th} concentric circle cuts at 2 points for 'E' with 0^0 rotation (Fig. 4.4 (a)). However, the same concentric circle cuts at 3 points when the character 'E' is rotated by 135^0 (Fig. 4.4(b)).

The interpattern dissimilarity between patterns of two different characters is shown in Fig 4.7. It is found that that the pattern vectors, produced by SAFER, had high interpattern deviation, which is very important for any classification problem. For example, in the figure the average standard deviation between 0^0 'A' and 0^0 'E' is greater than 20%.

Figs. 4.8 and 4.9 show the effect of scaling on the pattern vector generated by SAFER. It is found that although scaling has low effect on pattern vector, some arguments of the pattern vectors affect much by scaling. For example, in Fig 4.9, the patters of the three different scaled 'E' has an average standard deviation are less than 6%. However, the standard deviation between 6^{th} and 18^{th} arguments of pattern vectors in 40×40 and 20×20 pixels 'E' character are of about 6% and 13%, respectively.



(a) (b) (c)

Fig. 4.4 Rotation of Character 'E' at (a) 0^0 (b) 20^0 and (c) 135^0 rotation

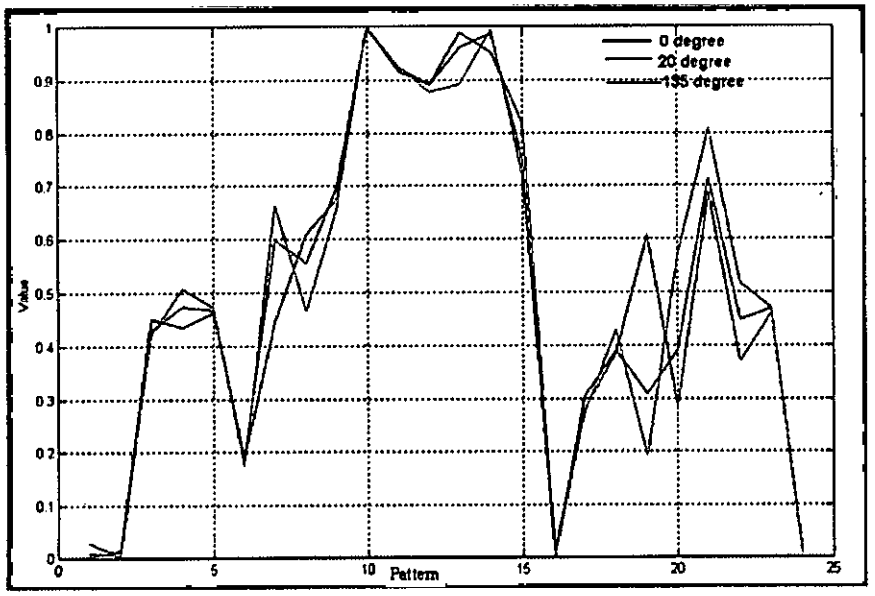


Fig. 4.5 Pattern amplitude vs Pattern Index of character 'E'

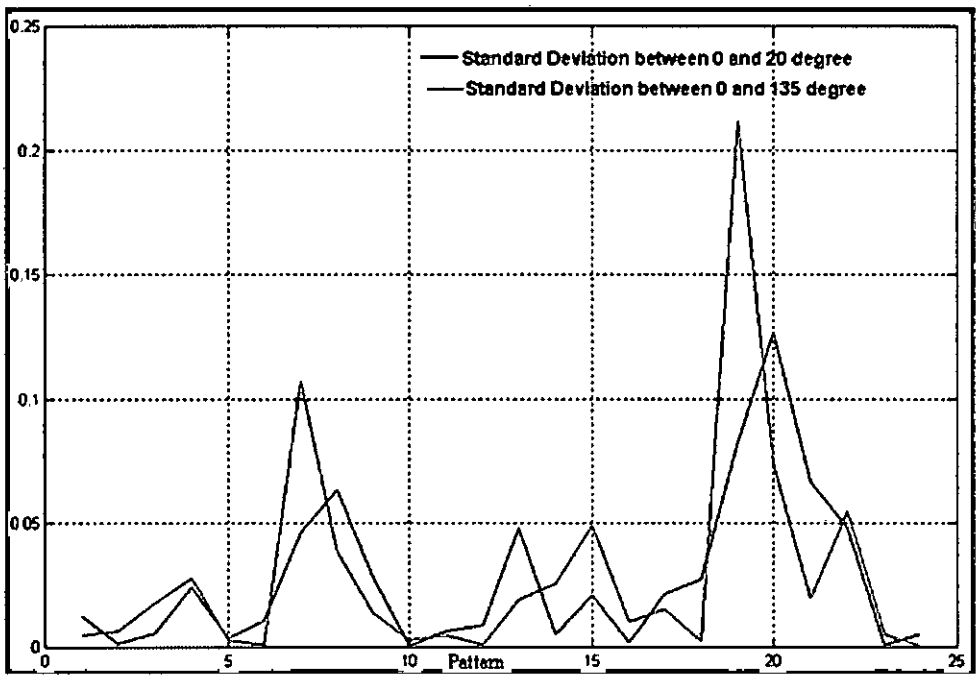


Fig. 4.6 Standard deviation between different orientations of 'E'

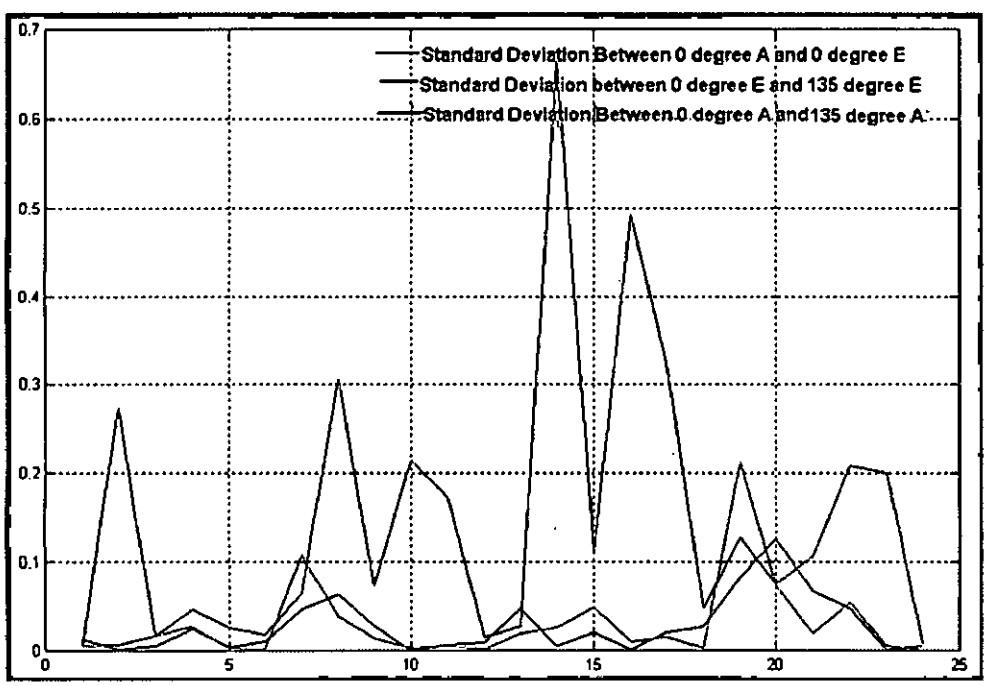
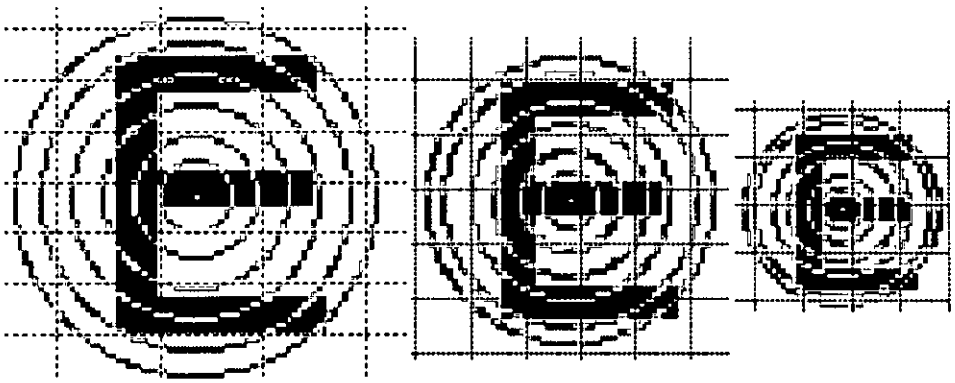


Fig. 4.7 Inter-pattern standard deviation and intra-pattern standard deviation.



(a) (b) (c)

Fig. 4.8 Scaling of character 'E' (a) 50 × 50 (b) 40 × 40 and (c) 25 × 25 pixels.

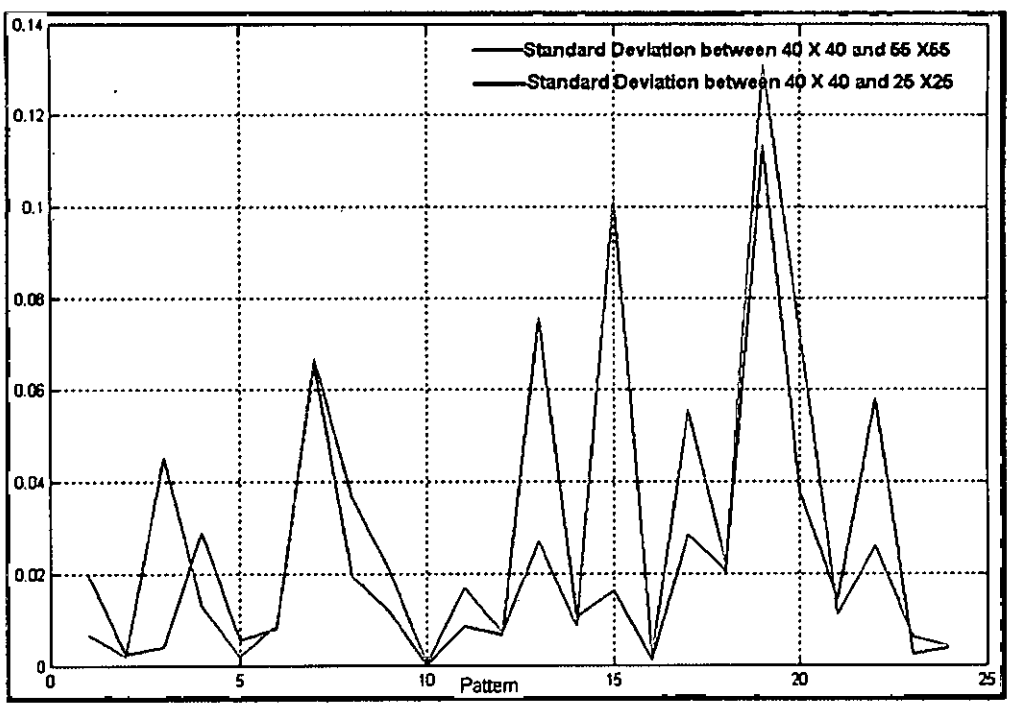
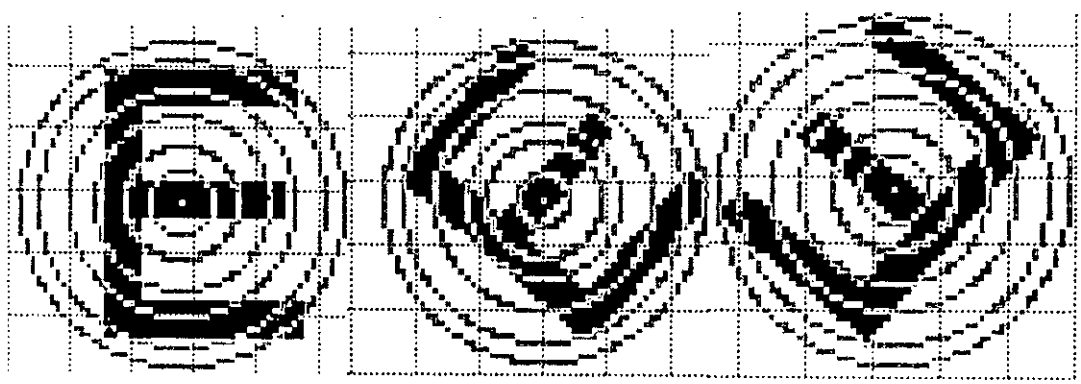


Fig. 4.9 Similarity between different scaling of character 'E'

One of the greatest advantages of ANNs is their generalization ability to approximate an arbitrary function from observed data. However, using the training data it can classify pattern, which was not so straightforward, and can made a relatively good understanding between deformed patterns. Fig. 4.10 and Table 4.11 show the generalization ability of

ANN by recognizing deformed patterns. Fig 4.10 (a) and (b) shows that the 3rd inner circle cuts the 0^0 and 40^0 rotated character 'E' at 3 and 2 points respectively. Fig. 4.10(a) and (c) show this type dissimilarity at the 5th inner circle. It is found from Fig 4.11(a) that there were about 30%, 15%, 16% and 19% standard deviation between 3rd, 9th, 15th and 18th arguments of the pattern vectors of 0^0 and 40^0 rotated 'E'. Although there were many intrapattern deviations, ANN can recognize them correctly.

It is well known that the selection of discriminative features is a crucial step in any pattern recognition system, since the classifier sees only these features and acts upon them. The ability of identifying discrimination features from similar type different characters by SAFER is shown in Figs. 4.12, 4.13 and Table 4.12. However, using the training data it can classify pattern and can make a good discrimination between similar patterns of different character. It is clear from the tables that SAFER can successfully identify features from similar type characters 'A' and 'V'. For example, in Table 4.12, 'A' and 'V' generate almost similar pattern vectors. From Fig. 4.12 it is found that except the first inner concentric circle, all circles intersect both 'A' and 'V' similarly. The standard deviation between pattern vectors of A' and 'V' is less than 8% in most cases except the 1st, 7th, 13th and 19th arguments of the pattern vectors, which produce standard deviation of 25%, 33%, 65% and 19%, respectively. But the SAFER can recognize them. The reason of this behavior by SAFER may be due to the use of ANN as a classifier. When ANN trained with both the patterns of 'A' and 'V' repeatedly it may automatically detect the discriminating features from 'A' and 'V'.

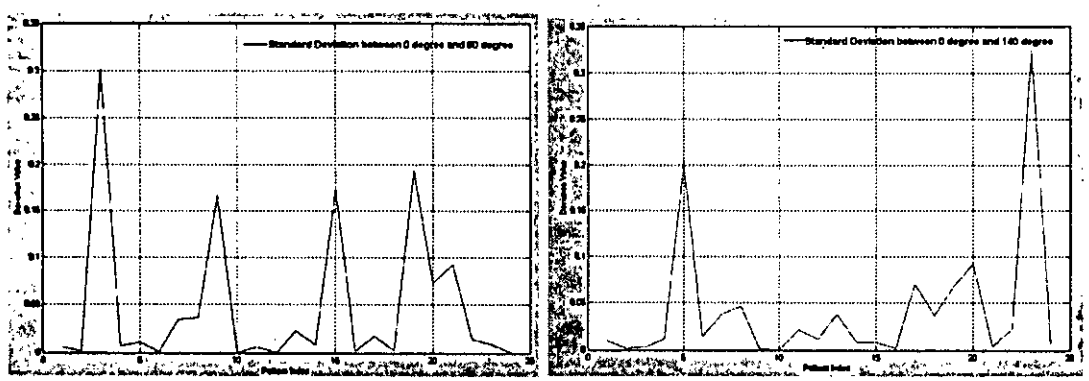


(a) (b) (c)

Fig. 4.10 Different cut point for same character at (a) 0^0 (b) 50^0 and (c) 140^0 rotation

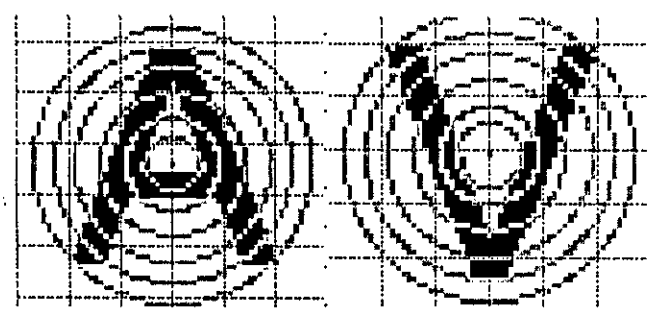
Table 4.11 Deformed patterns generated from incorrect cut point for 'E'

	Pattern Index											
	1	2	3	4	5	6	7	8	9	10	11	12
0^0	0.0097	0.0030	0.4274	0.4737	0.4669	0.1876	0.5980	0.5552	0.6945	0.9993	0.9163	0.8904
50^0	0.0016	0.0000	0.0017	0.4636	0.4822	0.1891	0.5494	0.5035	0.4582	0.9986	0.9242	0.8915
140^0	0.0235	0.0001	0.4226	0.4899	0.1823	0.1677	0.6515	0.4897	0.6961	0.9999	0.8861	0.8736
	Pattern Index											
	13	14	15	16	17	18	19	20	21	22	23	24
0^0	0.9615	0.9878	0.7474	0.0027	0.3045	0.3902	0.3076	0.3902	0.7144	0.4476	0.4688	0.0122
50^0	0.9934	0.9999	0.9930	0.0052	0.2796	0.3866	0.5808	0.4929	0.5832	0.4275	0.4815	0.0130
140^0	0.9081	0.9995	0.7354	0.0004	0.4036	0.4416	0.2111	0.5205	0.7199	0.4798	0.0093	0.0034



(a) (b)

Fig. 4.11 Similarity between deformed pattern of character 'E'



(a) (b)

Fig. 4.12 Similar type character 'A' and 'V'

Table 4.12 Patterns of 'A' and 'V'

	Pattern Index											
	1	2	3	4	5	6	7	8	9	10	11	12
0^0 A	0.000	0.388	0.403	0.407	0.431	0.161	0.505	0.987	0.797	0.695	0.670	0.867
0^0 V	0.353	0.353	0.395	0.416	0.414	0.136	0.978	0.978	0.728	0.667	0.581	0.843
	Pattern Index											
	13	14	15	16	17	18	19	20	21	22	23	24
0^0 A	0.999	0.049	0.588	0.698	0.762	0.458	0.488	0.282	0.564	0.741	0.752	0.001
0^0 V	0.085	0.085	0.661	0.715	0.756	0.529	0.220	0.220	0.687	0.786	0.710	0.000

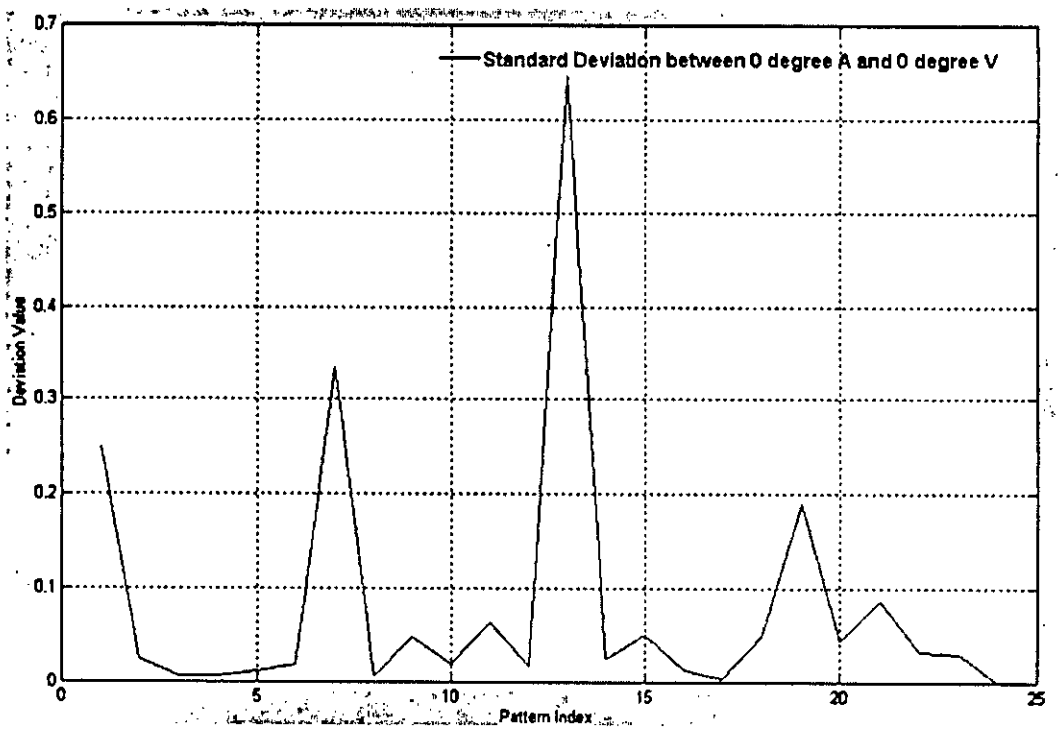


Fig. 4.13 Inter-pattern similarity for character 'A' and 'V'

Tables 4.2 and 4.9 show the effect of noise on the performance of SAFER for different characters. It is found from the tables that there is very low effect of noise on the performance of SAFER. However, the performance of some similar characters decreases. For example, SAFER achieves an overall recognition rate of 98.61% and 97.22% for 40 × 40 pixel normal and noisy fonts respectively. The recognition rate achieved by SAFER for noisy characters 'C' and 'O' were 93% and 94.44% respectively. However, the recognition rate for normal 'C' and 'O' were 95.8% and 94.44% respectively. The reason

of the small changes of performance by SAFER for noisy character is due to their less affection by noise.

The effect of noise on the character and generated pattern vector of 'C' and 'O' are shown in Figs. 4.14-4.18. It is clear from the Figs. 4.14-4.15 that there are many discrimination features in the generated pattern vector of normal 'C' and 'O' characters. For example, it is found from Fig 4.14 that the patterns of 'C' and 'O' achieve less than 4% standard deviation in most cases. However, they produce about 12% deviation at 4^{th}, 5^{th} and 6^{th} arguments and about 16% deviation at 22^{nd}, 23^{rd} and 24^{th} arguments of the pattern vector. The 22^{nd}, 23^{rd} and 24^{th} arguments of pattern vector are produced by 4^{th}, 5^{th} and 6^{th} circle respectively. Therefore, the discrimination features of 'C' and 'O' are produced from the 4^{th}, 5^{th} and 6^{th} concentric circles. There are two main differences in 'C' and 'O' that are described by the three outer circles. The first one is, in 'C', the center of mass (COM) is five pixels left than that of 'O' and the second one is the discontinuity on the right side of 'C'. Fig. 4.15 and table 4.13 show them clearly.

As the discrimination features of 'C' and 'O' are come from outer circles, they affect less by noise. The effect is shown in Figs. 4.16-4.18. From Fig. 4.16 and table 4.14 it is found that the character 'C' is almost unaffected by noise. However, the 4^{th} concentric circle of noisy character 'C' cuts at different positions relative to normal 'C'. From table 4.14 it is found that the 4^{th} circle cuts at 105^0 and 170^0 from noisy 'C' where they are 23^0 and 113^0 for normal 'C' (table 4.13). Fig. 4.17 shows that the standard deviation between pattern vectors of noisy 'C' and normal 'O' is almost similar to that of between normal 'C' and 'O'. The effect of the 4^{th} circle is clear in 4^{th} and 16^{th} arguments of pattern vectors. However, the character 'O' is almost unaffected by noise. The standard deviation between pattern vectors of normal 'C' and noisy 'O' is shown in Fig. 4.18. It is almost similar to fig. 4.14.

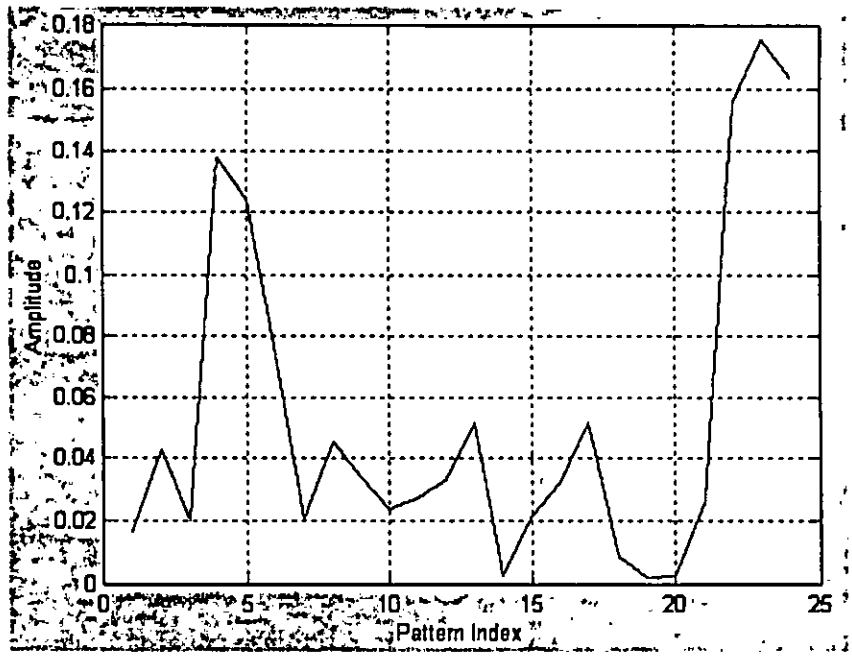
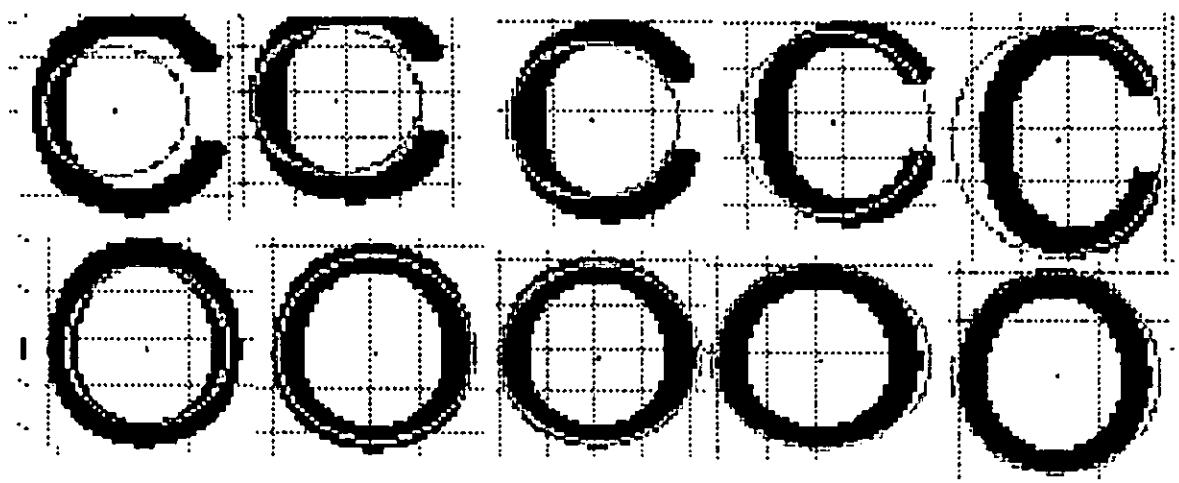


Fig. 4.14 Standard deviation between pattern vectors of normal 'C' and 'O'



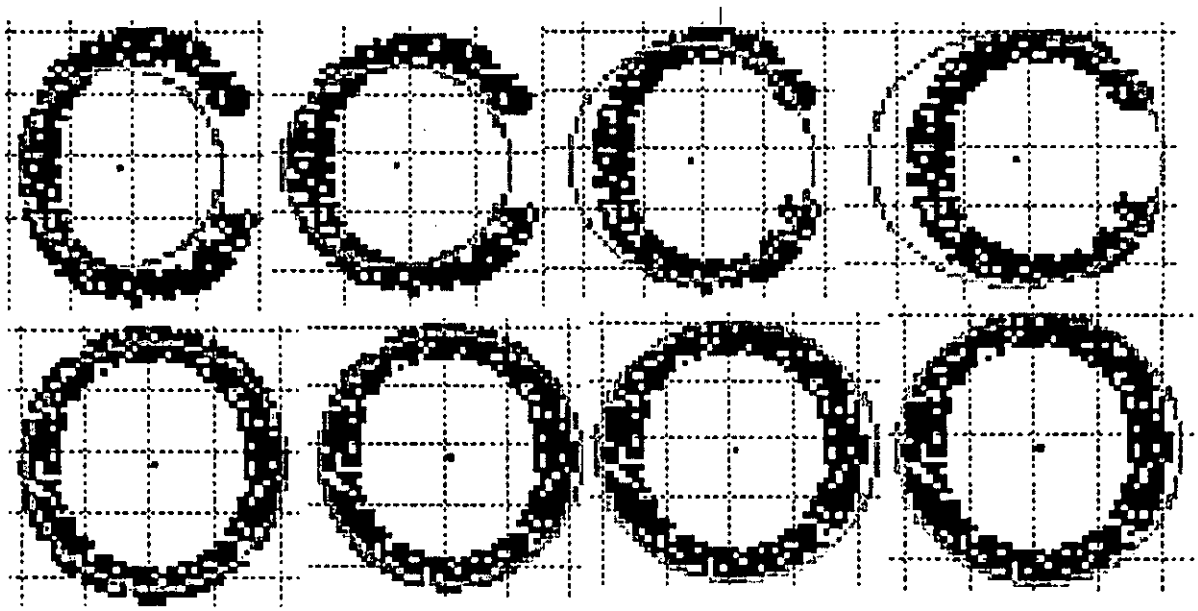
(a) (b) (c) (d) (e)

Fig. 4.15 2nd to 6th concentric circles of 'C' and 'O' characters

Table 4.13 The cut angles produced by 4th to 6th concentric circles from normal 'C' and 'O'

Character	Circle No.	Angle 1 (Degree)	Angle 2 (Degree)
C	4	23	113
	5	72	N/A

	6	54	N/A
O	4	100	N/A
	5	90	N/A
	6	90	N/A



(a) (b) (c) (d)

Fig. 4.16 3rd to 6th concentric circles of 20% noisy 'C' and 'O' character

Table 4.14 The cut angles produced by 4th to 6th concentric circles from noisy 'C' and 'O'

Character	Circle No.	Angle 1 (Degree)	Angle 2 (Degree)
C	4	105	170
	5	78	N/A
	6	49	N/A
O	4	110	N/A
	5	93	N/A
	6	90	N/A

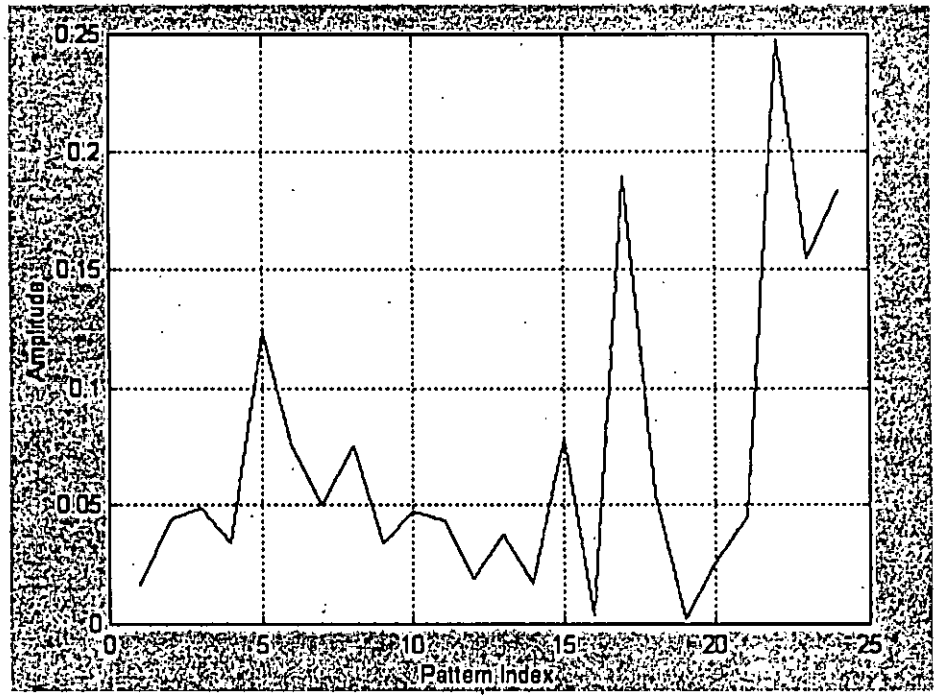


Fig. 4.17 Standard deviation between pattern vectors of noisy 'C' and normal 'O'

Fig. 4.18 Standard deviation between pattern vectors of normal 'C' and noisy 'O'

4.4.2 Reason of poor recognition rate

Tables 4.4 and 4.5 show the effect of scaling on the performance of SAFER for different characters. It is found from the tables that there is almost no effect of scaling on the performance of SAFER for large characters. However, for small sized characters the performance decreases. For example, SAFER achieves an overall recognition rate of 99.57% and 95.51% for 60 × 60 and 20 × 20 pixel fonts respectively. The recognition rate achieved by SAFER for characters 'W' and 'M' of 20 × 20 pixel font were 75% and 94.44% respectively. The reason of the low performance by SAFER for some characters is due to the similarity between generated pattern vectors produced by them. For example, 'W' and 'M' produce similar pattern vectors after rotated them into certain angle.

The effect of scaling on the generated pattern vectors of 'M' and 'W' characters is shown in Figs. 4.19 -4.24. It is clear from the Figs. 4.19-4.20 that there is very low effect of similarity on the generated pattern vector for large sized characters. For example, it is found from Fig 4.19 that the patterns of 'M' and 'W' achieve less than 10% standard deviation in most cases. However, they produce about 20% deviation at 1^{st} and 2^{nd} arguments and about 34% deviation at 19^{th} and 20^{th} arguments of the pattern vector. The 19^{th} and 20^{th} arguments of pattern vector are produced by 1^{st} and 2^{nd} circle respectively. Therefore, the discrimination features of 'M' and 'W' are produced from the 1^{st} and 2^{nd} concentric circles. There are two main differences in 'M' and 'W' that are described by the two inner circles. First one is, in 'M', the center of mass (COM) is four pixels upward than that of 'W'. The second one is due to the convex angle of the inner edge of 'W', which is slight smaller than that of 'M' (Fig. 4.20(d)). Fig. 4.20 and Table 4.15 show them clearly. The standard deviation between two pattern vectors generated from 'W' with 60 × 60 and 20 × 20 pixel fonts. It is found that the average standard deviation is less than 10%.

As the discrimination features of 'M' and 'W' are come from the innermost circles, they affect much by noise. The small circle has lower number of pixels on its periphery. Therefore, each pixel represents much angular distance than that of large circle. The effect of noise is shown in Figs. 4.22-4.24. For example, due to some noise the COM of 'W' moves one pixel upward. Fig. 4.22 and Table 4.16 show the effect of the

noise on two different sized 'W' characters. From Tables 4.15 and 4.16 it is clear that there is almost no effect of noise on 60 × 60 pixel 'W' character. However, in case of 20 × 20 pixel 'W' character, the noise changes the properties of discriminating features. In large 'W' the radiuses of 1st and 2nd concentric circles are 8 and 12 pixels respectively. However, in case of small sized 'W', they are 5 and 7 pixels respectively. As a result, one-pixel movement of COM for small sized 'W' moves its property toward 'M'. Again the 2nd concentric circle in Fig. 4.22 (d) looks much similar to that of 'M' (Fig. 4.20(c)). Table 4.16 shows that the noisy 'W' produces cut angles similar to 'M" as shown in Table 4.15. If radius of the 2nd circle in Fig. 4.22(d) is one pixel larger then it will be similar to that of 'W' (Fig. 4.20(d)). But in that case the circle become unstable, as it touches the boundary of 'W'. Fig. 4.24 shows that the noisy 'W' highly correlate with 'M'. Therefore, The ANN classifies it as 'M' character.

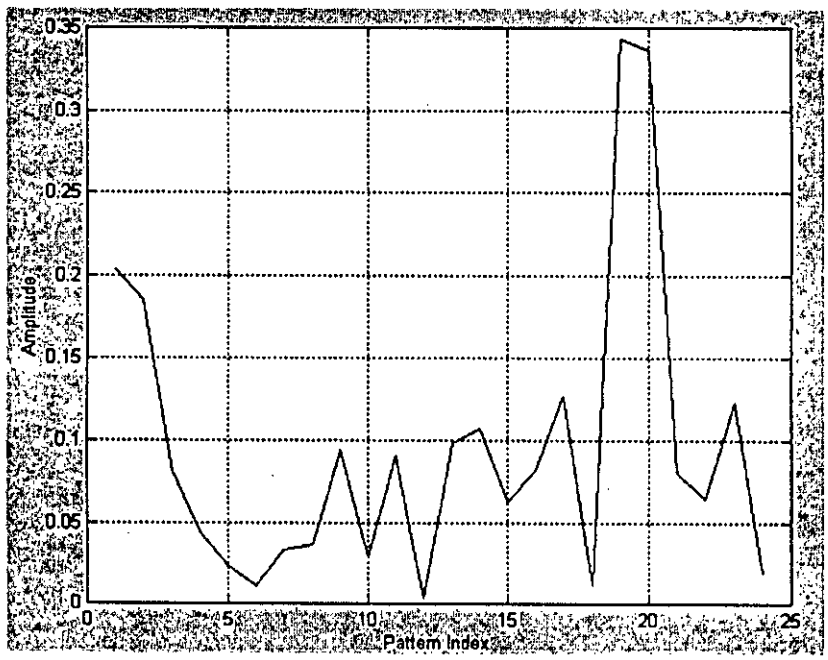


Fig. 4.19 Standard deviation between pattern vectors of 0^0 'M' and 'W'

(a) (b) (c) (d)

Fig. 4.20 1st and 2nd concentric circles of 'M' and 'W' characters

Table 4.15 The cut angles produced by 1st and 2nd concentric circles from 'M' and 'W' characters

Character	Circle No.	Angle 1 (Degree)	Angle 2 (Degree)
M	1	97	N/A
	2	44	165
W	1	110	N/A
	2	39	180

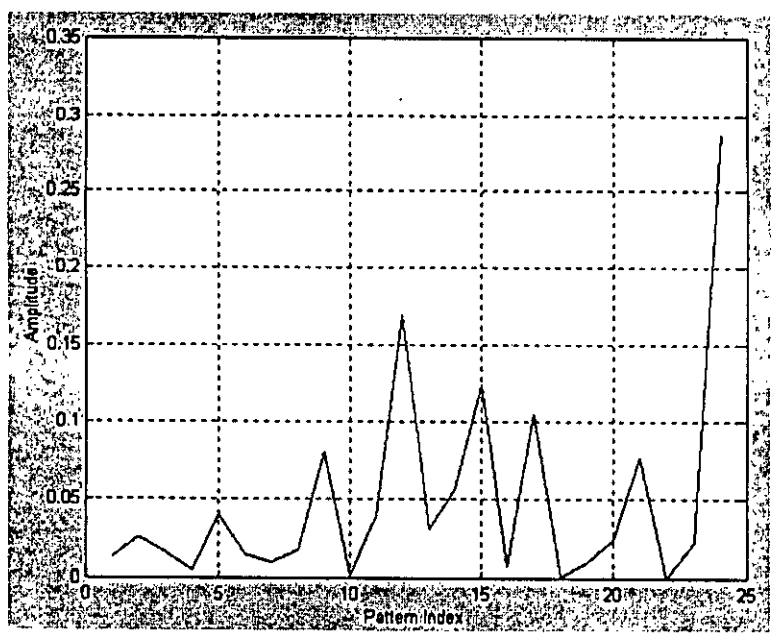
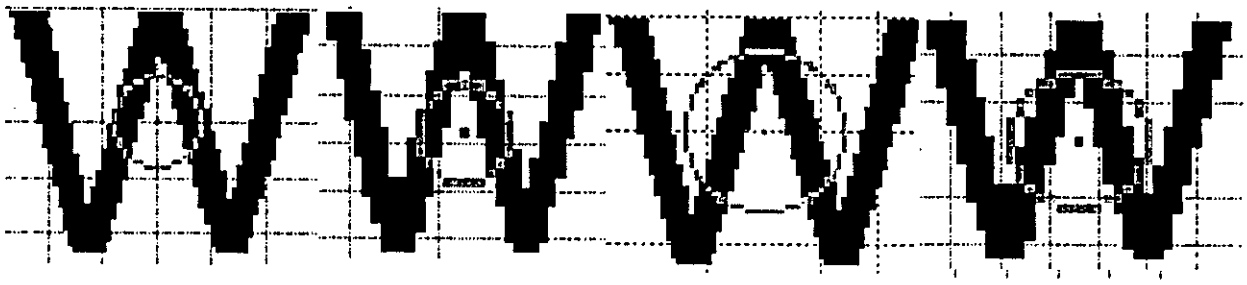


Fig. 4.21 Standard deviation between pattern vectors of 60 × 60 and 20 × 20 pixel 'W' characters



(a) (b) (c) (d)

Fig. 4.22 1st and 2nd concentric circle of 60 × 60 and 20 × 20 pixels 'W' characters. (a), (c) for 60 × 60 pixel and (b), (d) for 20 × 20 pixel

Table 4.16 The cut angles produced by 1st and 2nd concentric circles from character 'W' of two different sizes

Size	Circle No.	Angle 1 (Degree)	Angle 2 (Degree)
60 × 60	1	108	N/A
	2	36	180
20 × 20	1	98	N/A
	2	45	171

Fig. 4.23 Standard deviation between pattern vectors of 60 × 60 pixel 'W' and 20 × 20 pixel noisy 'W' characters

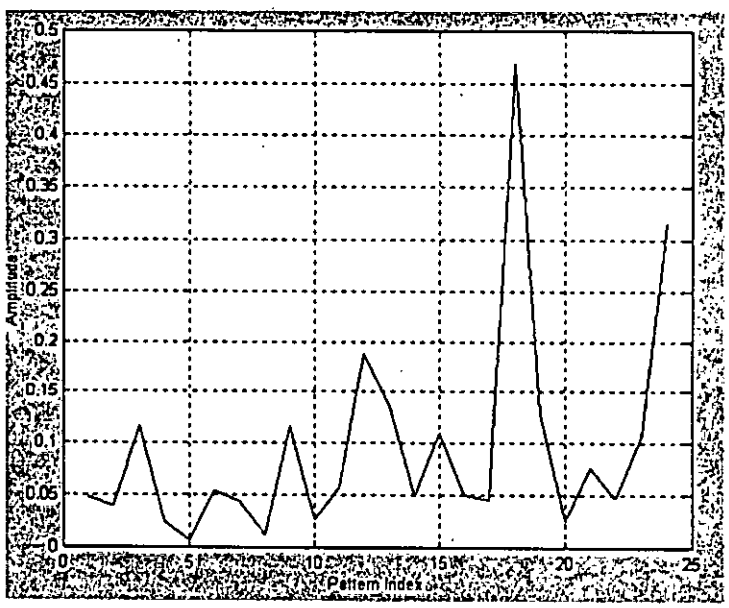


Fig. 4.24 Standard deviation between pattern vectors of 60 × 60 pixel 'M' and 20 × 20 pixel noisy 'W' characters

When the ANN trained with Arial font and tested with Tahoma font, it was found from Table 4.8 that there was very poor recognition rate for some characters. This was

happened because there is a major dissimilarity between same characters of two fonts. In Fig. 4.25 the upper 4 'M's are from Arial and the lowers are from Tahoma.

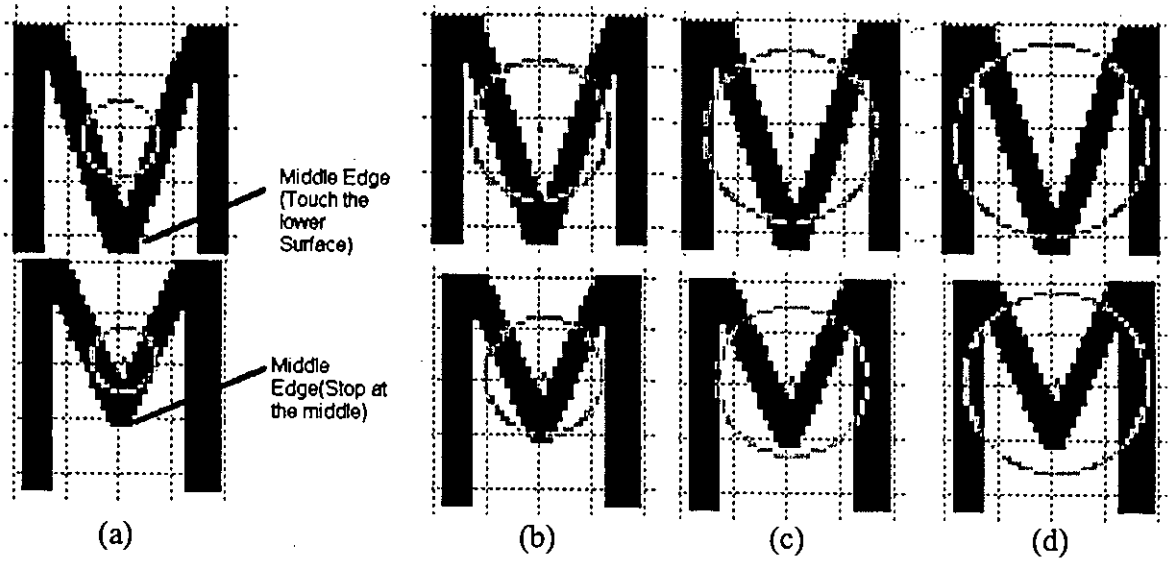


Fig. 4.25 Dissimilarity between 'M' of Arial and Tahoma Font.

In Fig. 4.25 (a) the Arial 'M's middle edge prolonged to the lower boundary but in Tahoma it stops at the middle of the character. The effect of this dissimilarity is shown in Fig. 4.25 (b) to (d). Fig. 4.25 (b) shows, the 2nd innermost circle cut 3 points for both Arial and Tahoma fonts. But, 3rd and 4th circle cut 5 and 3 points for Arial 'M' and 4 and 2 points for Tahoma 'M' (Fig. 4.25 (c) , (d)) . Similar condition also occurs for 5th and 6th circle. So, out of six concentric circles four generate different arguments from 'M' in two fonts. Therefore, when the network trained with Arial's 'M' character but tested with Tahoma, the ANN failed to detect it.

4.4.3 Comparison with other work

This section compares experimental results of SAFER with those of Holographic Nearest-Neighbor (HNN) based Algorithm [38] and the second was Complex Moment based method [13]. The primary aim of this section is not to exhaustively compare SAFER with all other ensemble algorithms, but to evaluate SAFER in order to gain a deeper understanding of the performance.

The average results of SAFER and Holographic Nearest-Neighbor (HNN) based Algorithm were obtained by training the SAFER with 104 (26 × 4) patterns of 40×40

pixels Arial characters. Testing data set was constructed with 3276 experimental patterns, which were generated from 9 different sizes and 14 different rotations of each character. The results for Complex Moment based method was obtained by trained the ANN with 30 randomly generated versions of different sized characters. Total number of training pattern was 780 (30×26). Testing pattern was also 780 (30×26) randomly generated experimental characters.

Table 4.17.1 to 4.17.3 compares SAFERs result with those produced by Holographic Nearest-Neighbor (HNN) based Algorithm [38]. It is clear from the tables that SAFER was able to achieve higher average recognition rate than HNN. For example, the average recognition rate achieved by SAFER for large, medium and small sized character were 99.35%, 98.13% and 95.97% respectively, while they were 99.07%, 96.17% and 91.65% for HNN. Although for small sized characters the overall performance of SAFER is much better than HNN, SAFER performs badly for some specific characters. For example, in Table 4.17.3 the recognition rate of 'C' and 'W' were 89.33% and 80.55% respectively for SAFER, while they were 98% and 100% for HNN. Table 4.18 compares the processing time taken by HNN and SAFER in preprocessing and classification stage. SAFER takes low processing time than HNN based technique.

The comparison of SAFER with Wavelet and complex moment based techniques [13] is shown in Table 4.19. Similar to the previous cases, SAFER outperformed significantly than moment based technique in terms of complexity, feature used and recognition rate. For example, complexity of moment based techniques is very high as they use complex calculation. Zernike and wavelet moment based techniques use 26 and 37 features respectively while SAFER use only 12 features. The classification rate of SAFER was also better. So, SAFER wins from different points of view.

Based on the above comparisons, it is clear that SAFER performed better than other algorithms in most cases. Although such comparisons may not be entirely fair due to different experimental setups, it was tried best to make this experimental setup as close to the previous ones as possible. Because of the diverse range of experimental setups used in previous studies, it is difficult to do an exhaustive comparison with all other work under different experimental setups. This is outside the scope of this thesis.

Table 4.17 Comparison between HNN and SAFER algorithm in terms of recognition rate of Arial character

Table 4.17.1 Font size: 100×100 – 45 ×45

Letter	HNN	SAFER	Letter	HNN	SAFER
A	100	100	N	99	100
B	100	100	O	99	95.833
C	98	94.44	P	96	98.611
D	100	100	Q	100	95.833
E	98	100	R	98	100
F	100	100	S	99	100
G	100	100	T	98	100
H	100	100	U	98	98.611
I	100	100	V	100	100
J	100	100	W	96	100
K	100	100	X	100	100
L	98	100	Y	100	100
M	100	100	Z	99	100
				99.07	**99.35**

Table 4.17.2 Font size : 40×40 – 35 ×35

Letter	HNN	SAFER	Letter	HNN	SAFER
A	100	100	N	98	100
B	100	100	O	100	94.444
C	98	94.444	P	95	90.277
D	96	97.222	Q	90	95.833
E	90	97.222	R	100	93.055
F	88	100	S	91	100
G	88	94.444	T	96	100
H	100	100	U	100	97.222
I	100	100	V	94	100
J	100	100	W	94	100
K	100	100	X	95	100
L	95	100	Y	100	100
M	100	100	Z	92	97.222
				96.15	**98.13**

Table 4.17.3 Font size: 25×25 – 20 ×20

Letter	HNN	SAFER	Letter	HNN	SAFER
A	100	100	N	99	99
B	98	100	O	98	98.61111

C	98	89.3333	P	96	83.33334
D	94	94.44444	Q	97	97.22222
E	95	97.22222	R	100	86.11111
F	89	97.22222	S	91	95.8333
G	89	97.22222	T	89	100
H	100	100	U	89	91.6667
I	100	100	V	94	100
J	99	100	W	100	80.55556
K	93	100	X	100	100
L	50	100	Y	98	100
M	32	91.66667	Z	95	95.83333
				91.65385	**95.97222**

Table 4.18 Comparison between HNN and SAFER algorithm in terms of processing time

Algorithm	Processor Speed	Preprocessing (s)	Classification(s)
HNN	500 MHz	0.105462	0.108846
SAFER	500MHz	0.085247	0.112553

Table 4.19 Comparison between Moment based techniques and SAFER in terms of Complexity, Feature used and Recognition rate.

Technique	Process	Complexity	Features Used	Recognition Rate
Zernike Moment	Rotation invariant moment	High	26	98.7%
Wavelet Moment	Rotation invariant moment	High	37	99.5%
SAFER	Axis of Symmetry and Angle	Very Simple	12 (6 circle × 2 harmonics)	99.8%

4.5 Discussion

This section briefly explains why the performance of SAFER is better than other algorithms, e.g., HNN [38] and Moment based technique [13]. There are three major differences that might contribute to better performance of SAFER in comparison with

other algorithms.

The first reason is that SAFER uses the axis of symmetry of a character to extract its basic features. As seen from Fig. 3.4 to 3.6, the pattern generated based on axis of symmetry is rotation and scaling invariant. The reference axis side is also used in SAFER to generate invariant patterns for different scaled and rotated characters. These two techniques used in SAFER give an upper edge to improve the recognition rate for rotated and scaled patterns. However, the complex moments of a character image are integrals of the image function over space, and the image can be uniquely determined by its complex moments of all orders. There is no competent way to extract feature from the image efficiently. Therefore, the complex moments are sensitive to digitization error, minor shape deformations, camera non-linearity, and non-ideal position of camera.

The second reason is the use of vector sum process by SAFER for pattern generation. As shown in section 3.2.3, the lower order vector sum depends on global change but it is invariant to the local change of features. However, local features also need to consider for some characters. For example, the basic shape of character 'A' and 'V' are same except there is a local feature difference. Since higher order vector sum can describe local feature more accurately, therefore, a second order harmonics was introduced in SAFER for pattern generation. However, taking the global and local features simultaneously, SAFER was able to generate patterns that were compatible for all alphabetic character. However, complex moments can be used to discriminate between images of real objects only if their shapes are significantly different. Since these moments are designed to capture global information about the image, they are not suitable for classifying similar objects when corrupted by a significantly amount of random noise.

The third reason is the effect of training procedure on the ANN. In beginning, the initial ANN was trained by a training data set. When it is found that the network does not have sufficient capability to handle the training process, new hidden nodes are added by using RWCC. The advantages of this process are that the network never falls at local minima and it requires very low training period. As new neurons are added on demand basis, so the final ANN architecture contains the exact number of hidden nodes that can

solve a particular problem. So there is no complexity to determine the initial architecture of ANN.

Chapter 5

CONCLUSION

5.1 Conclusive Remarks

Invariant Character Recognition (ICR), whose aim is to identify a character independently of its position (translated or rotated) and size (larger or smaller), has been the object of an intense and thorough study. In the last several years, an increasing number of research groups have proposed a great variety of ICR methods. However, they suffer from some significant limitations. Most important of the limitations are that the methods are either computationally very complex or not invariant for translation, rotation and scaling at a time. To address these limitations, a number of research groups have developed different types of ICR algorithm. The focus of this thesis has been the development of an ICR algorithm, called SAFER, which overcomes the significant limitations of previous algorithms.

ICR algorithms have been introduced to the OCR community for nearly 20 years. However, most ICR algorithms can only detect translated, rotated or scaled characters, resulting in an insufficient feature extractor. Few algorithms exist that are purely invariant, e.g., they can detect translated, rotated and scaled characters. This paper proposes a new ICR algorithm to extract invariant features from characters as well as classify them successfully. Neither the translation, rotation or scaling of the character needs to be predefined. However, they are determined automatically in the different steps by SAFER.

The proposed algorithm i.e. SAFER, uses preprocessing and classification stage. In preprocessing stage, it extracts the axis of symmetry of symmetrical characters, which is invariant under translation, scaling and rotation. It also determines the invariant reference axis side and generates pattern by using a novel technique. The technique uses higher order vector sum that can extract both global and local feature of characters. In classification stage, a new method, the random weight based cascade correlation (RWCC) is developed to determine the exact architecture of the ANN and to train it. RWCC uses cascade architecture to

grow the neurons in hidden layer incrementally until the ANN performance reaches a satisfactory level. It starts with a simple ANN and then tries to solve the problem by increasing the number of hidden neurons in the hidden layer. RWCC uses the correlation maximization process for selecting any neuron of hidden layer.

Extensive experiments have been carried out in this paper to evaluate how well SAFER performed on different well-known fonts. The experimental results are compared with other HNN and complex moment based algorithms. In almost all cases, SAFER outperformed the others. The average recognition rate was almost 100%, which indicates the great stability of this algorithm.

Although SAFER has performed very well for almost all problems, experimental study appeared to have revealed a weakness of SAFER in dealing with the small sized characters. The performance of the algorithm decreases slightly for smaller characters. This may be due to the fact that smaller characters suffer much by different noises. However, this problem was solved by adding a second group of training vectors for letter sizes between 30 × 30 and 20 × 20 pixels.

SAFER was compared with other similar algorithms but such comparisons may not be entirely fair due to different experimental setups. However, it was tried best to make the experimental setup as close to the previous ones as possible. Because of the diverse range of experimental setups used in previous studies, it is difficult (and probably unnecessary) to do an exhaustive comparison with all other work under different experimental setups. This is outside the scope of this paper.

5.2 Future Scopes

SAFER has some limitations, which keeps open the field to study and improve the design methods of ICR. Increasing the recognition rate for small sized characters can be a good future work. In case of small sized character the three types of errors called, round up error, boundary noise and random noise, affect much. They can be corrected by increasing the value of relative factors. Although they can minimize the error for small sized character, they can affect seriously for

large sized characters. So, in future one can introduce some other factors to overcome this limitation.

Another constraint in SAFER is that it considered only one axis of symmetry, although some character have multiple symmetry axis. In future one can consider multiple axis of symmetry simultaneously, which may give good classification accuracy. Finally, here different aspect ratio of same character was not considered. SAFER is based on the fact that the circle is the only geometrical shape that is naturally and perfectly invariant to rotation. Therefore, if characters aspect ratio is changed, the property of circle also needs to change. One can think a better concept to overcome this problem.

REFERENCES

[1] C. A. B. de Mello and R. D. Lins, "A Comparative Study on OCR Tools," *Vision Interface,* Trois-Rivières, Canada, 19-21 May, 1999.

[2] D. T. Ivind, A. K. Jain and T. Taxt, "Feature Extraction Methods for Character Recognition – A survey," *Pattern Recognition*, vol. 29, pp. 641-662, 1996.

[3] F. Sandgren, "Creation of a customized character recognition application," *Master's thesis in Computational Linguistics*, Uppsala University, Q Department of Linguistics and Philology, November 26, 2004.

[4] V. Vasudevan, "Character Recognition Techniques - A Demo Program," *International forum for information technology in Tamil*, vol. 1, pp 25-30, 2002.

[5] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. Inform. Theory,* vol. 8, pp. 179 –187, 1962.

[6] C. H. The and R. T. Chin, "On image analysis by the method of moments," *IEEE Trans. Pattern Anal.Mach.Intell.*, vol. 10, pp. 496 –513, 1988.

[7] M. Gruber and K. Y. Hsu, "Moment-based image normalization with high noise tolerance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 136 –139, 1997.

[8] R. Mukundan and K. R. Ramakrishnan, "Moment Functions in Image Analysis — Theory and Applications," *World Scientific*, Singapore,1998.

[9] M. R. Teague, "Image analysis via the general theory of moments," *J. Opt. Soc. Amer.*, vol. 70, pp. 920–930, 1980.

[10] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 489–497, May 1990.

[11] C. Kan and M. D. Srinath, "Invariant character recognition with Zernike and orthogonal Fourier-Mellin moments," *Pattern Recognition,* vol. 35, pp. 143-154, 2002.

[12] C. W. Chong, P. Raveendran and R. Mukundan, "Translation invariants of Zernike moments," *Pattern Recognition*, vol. 36, pp. 1765 – 1773, 2003.

[13] D. Shen and H. S. Horace, "Discriminative wavelet shape descriptors for recognition of 2-D patterns," *Pattern Recognition*, vol. 32 pp.151-165, 1999.

[14] A. FARES, A. BOUZID and M. HAMDI "Rotation Invariance using diffraction pattern sampling in optical pattern recognition," *Journal of Microwaves and Optoelectronics*, vol. 2, no. 2, pp. 260-273, December 2000.

[15] E. Kavallieratou, N. Fakotakis and G. Kokkinakis "Slant estimation algorithm for OCR syatems," *Pattern Recognition*, vol. 34, pp. 2515-2522, 2001.

[16] M. S. Choi and W. Y. Kim, "A Novel two stage template matching method for rotation and illumination invariance," *Pattern Recognition*, vol. 35, pp. 119-129, 2002.

[17] M. G. Linares, N. Guil and E. L. Zapata, "An effective 2D deformable objects detection and location algorithm," *Pattern Recognition*, vol. 36, pp. 2543-2556, 2003.

[18] T. Bernier and J. Landry, "A new method for representing and matching shapes of natural objects," *Pattern Recognition*, vol. 36, pp. 1711-1723, 1999.

[19] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE trans. Comput.*, vol. 21, pp. 269-281, 1972.

[20] F. Ullah and S. Kaneko, "Using orientation codes for rotation-invariant template matching," *Pattern Recognition*, vol. 37, pp. 201-209, 2004.

[21] R. Gonzalez and P. Wintz, "Digital Image Processing," *2nd ed. Reading*, MA: Addison-Wesley, pp. 404–411, 1987.

[22] H. Kauppinen, T. Seppänen and M. Pietikäinen, "An experimental comparison of autoregressive and fourier-based descriptors in 2D shape classification," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, pp. 207–210, Feb. 1995.

[23] M. Caudill, "Neural networks primer pt. II," *AI Expert*, vol. 3, no. 2, pp. 55–61, 1988.

[24] C. H. The and R. T. Chin, "On image analysis by the methods of mo-ments," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, pp. 496–513, July 1988.

[25] G. L. Giles and T. Maxwell, "Learning, invariances, and generalization in high-order neural networks," *Appl. Opt.*, vol. 26, no. 23, pp. 4972–4978, 1987.

[26] S. J. Perantonis and P. J. G. Lisboa, "Translation, rotation, and scale invariant pattern recognition by high-order neural networks and moment classifiers," *IEEE Trans. Neural Networks*, vol. 3, pp. 241–251, Mar. 1992.

[27] L. Spirkovska and M. B. Reid, "Coarse-coded higher-order neural net-works for PSRI object recognition," *IEEE Trans. Neural Networks*, vol. 4, pp. 276–283, July 1993.

[28] C Yuceer and K Oflazer, 'A Rotation Scaling and Translation Invariant Pattern Classification System.' *Pattern Recognition*, vol. 26, no. 5, pp. 687-696, 1993.

[29] K. Fukushima, 'Character Recognition with Neural Networks,' *Neuro computing*, vol. 4, pp. 221-235, 1992.

[30] B. Hussain and M. R. Kabuka, 'A Novel Feature Recognition Neural Network and its Application to Character Recognition,' *IEEE Transactions on Patt Anal Machine International*, vol. 16, no. 1, pp. 98-112, 1994.

[31] A. Khotanzad and J. H. Lu, 'Classification of Invariant Image Representations using a Neural Network,' *IEEE Transaction ons ASSP*, vol 38, no 6, pp. 1028-1040, 1998.

[32] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, 1980.

[33] H. K. Kwan and Y. Cai, "A Fuzzy Neural Network and its Applications to Pattern Recognition," *IEEE Transactions on Fuzzy Systems*, vol. 2, no. 3, pp. 185-197, 1994,

[34] P. K. Simpson, "Fuzzy Min-max Neural Networks-part 1: Classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 776-790, 1992.

[35] H. P. Chiu and D. C. Tseng, "Invariant Handwritten Chinese Character Recognition using Fuzzy Min-max Neural Networks," *Pattern Recognition Letters*, vol. 18, pp. 481-497, 1997.

[36] U. V. Kulkarni, T. R. Sontakke and A. B. Kulkarni, "Fuzzy Hyperline Segment Clustering Neural Network," *Electronics Letters, IEE*, vol. 37, no. 5, pp. 301-315, 2001.

[37] Y. Avrithis, A. Delopoulos and S. Kollias, "An efficient scheme for invariant optical character recognition using triple correlations," *Computer Science Division*, National Technical University of Athens, 1998.

[38] L. A. Torres-Méndez, J. C. Ruiz-Suárez, L. E. Sucar and G. Gómez, "Translation, Rotation, and Scale-Invariant Object Recognition," *IEEE transactions on systems, man, and cybernetics, part c: applications and reviews*, vol. 30, no. 1, pp. 125-130, Feb. 2000.

[39] U. V. Kulkarni and T. R. Sontakke, "Fuzzy Hypersphere Neural Network Classifier," *The 10th IEEE International Conference on Fuzzy Systems*, Melbourne, Australia, pp. 1559-1562, December 2-5, 2001.

[40] N. Rishikesh and Y. V. Venkatesh, "Shape recognition using an invariant pulse code and a hierarchical, competitive neural network," *Pattern Recognition*, vol. 34, pp. 841-853, 2001.

[41] P. C. Dodwell, "Visual Pattern Recognition," *Holt, Rinehart and Winston Inc*, New York, 1970.

[42] B. Charroux, "Analyse d'images : coop´eration d'op´erateurs de segmentation guid´ee par l'interpr´ etation.," *PhD thesis*, Universit de Paris XI Orsay, 1996.

[43] C. Choisy, H. Cecotti and A. Belaýd, "Character Rotation Absorption Using a Dynamic Neural Network Topology: Comparison With Invariant Features," *In 6th International Conference on Enterprise Information Systems - ICEIS'2004*, Porto, Portugal, pp. 8, 2004.

[44] D. Mendlovic, Z. Zalevsky, I. Kiryuschev and G. Lebreton, "Composite harmonic filters for scale-, projection-, and shift-invariant pattern recog-nition," *Appl. Opt.*, vol. 34, pp. 310–316, 1995.

[45] M. Fang and G. Hausler, "Class of transforms invariant under shift, rotation, and scaling," *Appl. Opt.*, vol. 29, pp. 704–708, 1990.

[46] D. Mendlovic, E. Marom, and N. Konforti, "Shift and scale invariant pattern recognition using Mellin radial harmonics," *Opt. Commun.*, vol. 67, pp. 172–176, 1988.

[47] D. Mendlovic, N. Konforti, and E. Marom, "Shift and projection in-variant pattern recognition using logarithmic harmonics," *Appl. Opt.*, vol. 29, pp. 4784–4789, 1990.

[48] Y. N. Hsu and H. H. Arsenault, "Optical pattern recognition using the circular harmonic expansion," *Appl. Opt.*, vol. 21, no. 22, pp. 4016–4019, 1982.

[49]E. Ghahramani and L. R. B. Patterson, "Scale, translation, and rotation invariant orthonormalized optical/optoelectronic neural networks," *Appl. Opt.*, vol. 32, no. 35, pp. 7225–7232, 1993.

[50] D. H. Ballard, C. M. Brown, "Computer Vision," *Englewood Cliffs*, New Jersey, Prentice-Hall, pp. 65-70, 1982.

[51] W. K. Pratt, "Digital Image Processing," *New York: John Wiley & Sons*, Second ed., 1991.

[52] H. C. Andrews, "Multidimensional rotations in feature selection," *IEEE Trans, Computers*, vol. 20, pp. 1045-1051, Sept. 1971.

[53] R. C. Gonzalez, R.E. woods, "Digital Image Processing," Addison-Wesley, 1992.

[54] M. H. Glauberman, "Character recognition for business machines," *Electronics*, vol. 29, pp. 132-136, Feb. 1956.

[55] R. Kasturi and M. M. Trivedi, "Image Analysis Applications," New York, Marcel Dekker, 1990.

[56] H. P. Chiu, D. C. Tsen and J. C. Cheng, "Invariant handwritten Chinese character recognition using weighted ring-data matrix," *Proceedings of 3rd International Conference on Document Analysis and Recognition*. vol. 1, pp. 116-119, 1995.

[57] S. W. Lee and W. Y. Kim, "Rotation-Invariant template matching using projection method," *Proc. KITE* , vol. 19, no. 1, pp. 475-476, 1996.

[58] W. S. Kim, "Feature based two stage template matching using integral projections," *ICIP'89*, pp. 5-8, Sept. 1989.

[59] M. K. Hu, "Pattern recognition by moment invariants," *Proc. IRE (Correspondence)*, vol. 49, pp. 1428-1440, 1961.

[60] M. K. Hu, "Visual pattern recognition by moment invariants," *IRE Trans. on Information Theory*, vol. 8, pp. 179-187, 1962.

[61] J. Wood, "Invariant pattern recognition: A review," *Pattern Recognition*, vol. 29, no. 1, pp. 1-17, 1996.

[62] B. C. Li, "Applications of moment invariants to neurocomputing for pattern recognition.," *PhD Dissertation,* The Pennsylvania State University, 1990.

[63] L. L. Yudell, "Mathematical functions and approximations," chapter 11, page 432. Academics Press Inc, 1975.

[64] F. Zernike, "Beugungstheorie des Schneidenverfahrens und seiner verbesserten Form, der Phasenkontrastmethode (Diffraction theory of the cut procedure and its improved form, the phase contrast method)," *Physica*, vol. 1, pp. 689-704, 1934.

[65] C. Teh and R. T. Chin, "On image analysis by the method of moments," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, pp. 496-513, 1988.

[66] A. B. Bhatia and E. Wolf, "On the circle polynomials of Zernike and related orthogonal sets," *Proc. Cambridge Philosophical Society*, vol. 50, pp. 40-48, 1954.

[67] M. R. Teague, "Image analysis via the general theory of moments," *Journal of the Optical Society of America*, vol. 70, pp. 920-930, 1979.

[68] T. H. Reiss, "The revised fundamental theorem of moment invariants," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, pp. 830-864, Aug. 1991.