OFFLINE CHARACTER RECOGNITION USING MONTE CARLO METHOD AND NEURAL NETWORK

Hitesh Rajput[1], Tanmoy Som[2], and Somitra Kar[3]

[1,2] Department of Applied Mathematics, IIT(BHU), Varanasi, India
[1]hrajput.rs.apm@iitbhu.ac.in , [2]tsom.apm@iitbhu.ac.in
[3]Image Processing & Machine Vision, Bhabha Atomic Research Centre, Mumbai, India
[3]skar@barc.gov.in

ABSTRACT

Human Machine interface are constantly gaining improvements because of increasing development of computer tools. Handwritten Character Recognition do have various significant applications like form scanning, verification, validation, or checks reading. Because of the importance of these applications passionate research in the field of Off-Line handwritten character recognition is going on. The challenge in recognising the handwritings lies in the nature of humans, having unique styles in terms of font, contours, etc. This paper presents a novice approach to identify the offline characters; we call it as character divider approach which can be used after pre-processing stage. We devise an innovative approach for feature extraction known as vector contour. We also discuss the pros and cons including limitations, of our approach.

KEYWORDS

Artificial Neural Network, Back propagation, Feature extraction. Monte Carlo, Vector Contour

1.INTRODUCTION

Character recognition has been an active research topic for more many decades. With the advent of digital computing and signal or image processing, the problem of character recognition was clearly posed and thoroughly studied. There is a need of automatic character recognition in various real life applications viz credit cards, ATM cards, bank cheque, form processing, zip code, etc. Like nature and understanding; writing style of different persons are also different. Also even for the same person, it varies in different situations depending on the state of the human mind.

For this reason, the researchers are continuously publishing new and efficient approaches to address the problem of character recognition. The current approaches are based on template matching, dynamic programming, artificial neural network, hidden Markov model, or combination of these techniques. Some commercial software is also available in the market for character recognition viz OCR software, ANPR software, etc. Rhee et al. [16] proposed a recognition technique for rotated characters but failed to recognize the character of different size. The classical 1 paradigm for character recognition has three steps: segmentation, feature extraction, and classification. Lecolinet and Crettez [15] proposed two main classes of methods for recognition of words termed as analytical and global. The analytical class attempts to slice words into letters for recognition. On the other hand the global class attempts to recognize the words globally. The discussion of global approach is out of the scope of this paper and needs in-

DOI : 10.5121/ijaia.2013.4303

depth understanding of each symbol as these contain numerals like check number. Analytical methods are of following two categories:

1. with external segmentation [5]. In this segmentation stage is before recognition stage
2. with internal segmentation [17]. In this segmentation and recognition stages are at same time

Proper feature extraction requires a priori knowledge of the patterns that form meaningful units, which implies recognition capability. To, tackle this dilemma, we device a new approach for feature extraction based on vector contour that is illustrated in next section. The process involved in our approach is very similar to existing approaches as illustrated under:

1. Handwritten characters
2. Optical scanner for Digitalization
3. Isolation of characters
4. Preprocessing for Normalization & Thinning
5. Feature Detector (Matching)
6. Identity of characters (Recognition)

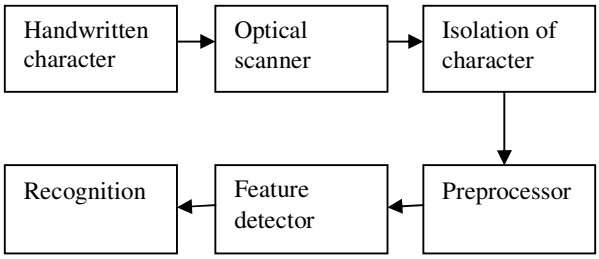

Fig.1 General Block diagram of handwritten character recognition

In pre-processing phase of characters, the character is bounded for normalization for standard size and thereafter thinning of character for noise removal or skeletonizing is done. The following steps are involved in our Preprocessing phase-

1. Extraction of a character in the given word.
2. Position normalization of character
3. Normalization of character
4. Thinning of normalized character

The preprocessing phase can be done using the standard techniques available.

The paper is organized as follows. We give the review of the basic concepts, required for our work in Section 2. We illustrate our approach for character recognition in Section 3. Section 4 describes the Pros and Cons of two stage Back propagation neural classifier which we have used. Section 5 illustrates the experimental results. Section6 describes the limitations of our approach and finally Section 7 concludes the paper.

2.BASIC CONCEPTS

2.1. Character Normalization

Normalization is done to make the size of all extracted character bitmaps equal. In order to match the extracted isolated character, it is important that all patterns should have the same size. So, size

normalization is required. Size normalization is the most efficient pre-processing technique. However, the use of pre-processing techniques depends on many factors such as quality and shape of the data and the recognition process employed.

There are various techniques available for normalization which can be referred from the test books or research papers. For the sake of completeness we try to explain as:

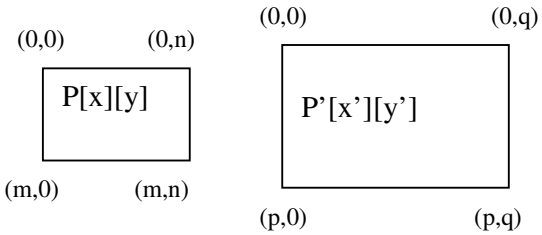

Fig. 2 Normalization

In this method, every input bitmap P, of dimension 'm x n', is transformed into a normalized bitmap P', of dimension 'p x q'. The geometric transformations generally modify the spatial relationships between pixels in an image. They are also termed as rubber sheet transformations.

2.2. Thinning

Thinning is the process to extract and apply additional constraints on the pixel elements that are to be preserved so that a linear structure of the input image will be recaptured without destroying its connectivity. Thinning plays a very important role in the pre-processing stage of pattern recognition. This is due to the fact that:

- It preserves essential structural information of an image
- It reduces the space to store topological as well as shape information of an image
- It reduces the complexity of analyzing the image.

In the context of two - dimensional binary image processing, thinning is considered as an recursive process of removing points or layers of outline from a binary image until all the lines or curves becomes single pixel wide. The reduced pattern is called the skeleton. A good thinning algorithm must preserve the topology as well as the shape of the original image in the skeleton.
Many thinning algorithms (or modifications of existing ones) have been proposed in recent years, and a comprehensive survey of these methods is contained in Lam L., Lee, Suen[9]. However, to name a few, Naccache and Singhal[13] made a study of fourteen thinning algorithms based on iteration erosion of boundary. They proposed the safe point thinning algorithm (SPTA). Saha et. al [14] proposed an improvement on the SPTA algorithm by suggesting a rotational invariant single scan boundary removal thinning algorithm (SBRTA). Lu and Wang [12] suggested an improvement on this. Lam and Suen[10] evaluated ten thinning algorithms.

2.3. Artificial Neural Netwrok

ANNs were introduced by McCulloch and Pitts in 1943. ANNs consists of algorithms, having capability to learn to solve complex problems from training data. ANNs consist of a set of pairs of inputs and desired outputs (targets). There is a wide range of ANN applications including speech recognition, image processing, etc. They can be trained to perform a specific task such as prediction, and classification.

ANN consists of interconnected processing elements called neurons that work together to produce an output.

2.6.1. Single Neuron and the Least-Mean-Squares (LMS) algorithm

The architecture of a single neuron is given Figure 3. The output a of the neuron is a weighted linear combination of its inputs. Here f is a scaling function. Some commonly used transfer functions are shown in Figure 4.

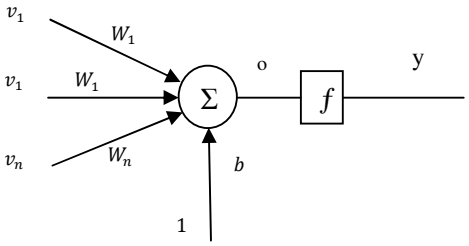

Fig. 3 Single neuron

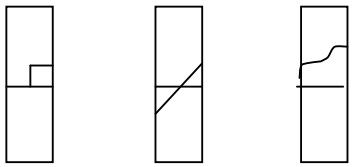

Fig.4 Transfer functions

The weights of the neuron can be repeatedly adjusted to give local or global optima using training data. Optimum weights in the sense of least square errors were derived by Widrow and Hoff [9] derived the optimum weights using least square errors algorithm. This algorithm is popularly known as LMS algorithm or Widrow-Hoff rule. In the Widrow-Hoff rule, the network weights are moved along the negative of the gradient of the performance function. After every iteration, the weights are adjusted according to the following rule:

$$W \leftarrow W + \alpha\, e\, V$$

where α is the learning speed, and the input vector $V \epsilon R^n$ is given by

$$V = [v_1, v_2, \ldots v_n]^T$$

and $W \epsilon R^n$ is the vector of weights and is given by

$$W = [w_1, w_2, \ldots w_n]^T$$

The output a of the neuron is defined by the following expression

$$o = w_1 v_1 + w_2 v_2 + \cdots + w_n v_n$$

After scaling, output of the neuron y is given by y = f(o), where f is the transfer function (Fig.4). The error e is the difference between the neuron's output and the desired output. The initial weights can be set using existing information. Alternatively, zero or some random values can be given to the weights.

2.6.1. Multilayer ANN

The architecture of multilayer ANN is given in Fig. 5, which shows a two-layer network. ANN is designed such that desired output can be obtained from the given set of inputs. The weights for the first (input) layer and the second (hidden) layer of the ANN in Fig. 6(a) are shown in Fig. 6(b) where $w_{i,j}^k$ denotes the weight for the ith input in the jth neuron of the kth layer

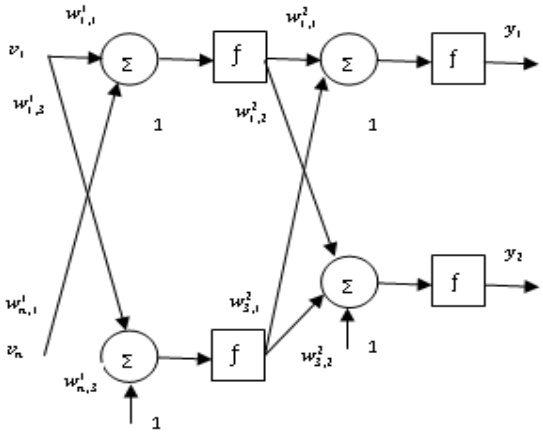

Fig.5.Multilayer ANN

$$\begin{bmatrix} w^1_{1,1} & w^1_{1,2} & w^1_{1,3} \\ \vdots & \vdots & \vdots \\ w^1_{i,1} & w^1_{i,2} & w^1_{i,3} \\ \vdots & \vdots & \vdots \\ w^1_{n,1} & w^1_{n,2} & w^1_{n,3} \end{bmatrix} \qquad \begin{bmatrix} w^2_{1,1} & w^2_{1,2} \\ w^2_{2,1} & w^2_{2,2} \\ w^2_{3,1} & w^2_{3,2} \end{bmatrix}$$

Fig.6 weight matrices of ANN in Fig. 3: (a) Layer1 and (b) Layer2

2.4. Back propagation Algorithm

Multilayer ANN that uses the Widrow-Hoff learning rule and nonlinear differentiable transfer functions is known as Back propagation. It has capacity to approximate any function with a finite number of discontinuities. Properly trained back propagation networks have proven to give reasonable answers when presented with inputs that they have never seen. It is possible to train a neural network on a particular set of input to get good results, rather than training the network on all possible inputs, which are generally very large.

In back propagation, learning cycle consists of two phases: (i) broadcasting the input patterns through the network and (ii) adapting the output by manipulating the weights in the networks. In the network operation, the error signals are back propagated to the hidden layers. An estimate of the contribution of a particular neuron to the output error can be seen by the portion of the error signals that a hidden layer neuron receives.

The squared error is reduced in each cycle and finally minimized by repeatedly adjusting the weights. In the feed forward back propagation network the connections from each neuron in a layer to every neuron in the next layer is forward. There are no lateral or recurrent connections. Labels on connections indicate weights.

2.6. Monte Carlo method

These are the class of computational algorithms that rely on repeated random sampling to compute their results. Monte Carlo methods are often used in computer simulations of physical and mathematical systems. They are used to model phenomena with significant uncertainty in inputs.

The Monte Carlo method was introduced by John von Neumann, Stanislaw Ulam and Nicholas Metropolis in the 1940s during their Manhattan Project, which was based on nuclear weapon in

the Los Alamos National Laboratory. It was named after the Monte Carlo Casino, a famous casino where Ulam's uncle often gambled away his money.

Monte Carlo methods vary, but tend to follow a particular pattern:

1. Define a domain of possible inputs
2. Generate inputs randomly from a probability distribution over the domain.
3. Perform a deterministic computation on the inputs.
4. Aggregate the results

3. FEATURE EXTRACTION AND CHARACTER IDENTIFICATION

We propose a novice method, termed as character divider approach for feature extraction of characters after the preprocessing steps. There are 7 phases, performed in our approach and is illustrated in the Fig 8

Phase1: Character Preprocessing

In this Phase we do the preprocessing of the character (as discussed in section1) can be done in three phases, given in Figure7.

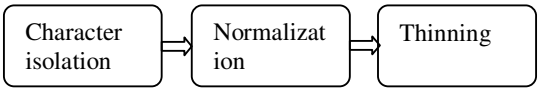

Fig.7 Phases of character preprocessing

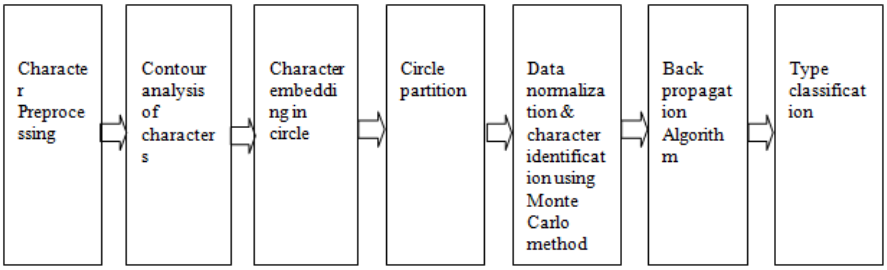

Fig.8.Phases for character identification

We perform 3 steps for character isolation, illustrated below:

Step1: Extraction of Lines- the text image is scanned from top to bottom and from left to right to extract number of lines in the written text. The algorithm is given in Figure9

Step2: Extraction of Character- within each line, top-left and bottom-right co-ordinates of each character is obtained using the algorithm, given in Figure10:

Step3: Position normalization of characters- For each character array obtained from above two algorithm, redundant rows of '0's are removed to get the actual character array, which fits into a bounding box.

Following above three steps, all the characters of different sizes, in separate 2D binary array. So we need normalization as give in section2.2 and illustrated in detail as under:

In the two quadrilateral regions, given in Figure2; the vertices correspond to the tie points. The

Algorithm for Line Extraction

```
int inpu[rows][cols];//input image binary image
array
int line_count;
scan inp[i][j] from top to bottom and from left to
right;

do{
if scan[i][j] == 1
   setline[line_count].start.x=1;
goto next row of array and look for '1' in that row;
if '1' found, goto next row and so on till a row (kth
row) of all '0's is encountered

setline[line_count].end.x = k-1;

//So, first line is obtained.

line_count++;}

while(all lines got scanned)
```

Fig.9.Line Extraction Algorithm

Algorithm for character extraction

```
Char scanchar[j];
while (getline)
{
  for (i<0;i<j;i++)//for each column
  {
   if(scanchar[i]=='1')
     {
      char[i].top_left.x=line[line_count].start.x;
      char[i].top_left.y=that col no.;
     }
   if(scanchar[i]=='0')
    {

char[i].bottom_right.x=line[line_count].end.x;
      char[i].bottom_right.y= that col no.;
    }
  }
//character gets extracted
}
```

Fig.10.Character Extraction Algorithm

geometrical transformation process within the regions is modeled by a pair of bilinear equations so that

$$x' = c_1x + c_2y + c_3xy + c_4$$
$$y' = c_5x + c_6y + c_7xy + c_8$$

These equations can easily be resolved for 8 coefficients$c_i, i = 1 \dots 8$. Once the coefficients are known, they constitute the model used to transform all pixels within the quadrilateral region characterized by tie points used to obtain the coefficients.
Simplifying the above equations, we get,

$$c_1 = \frac{p}{m};\ c_6 = \frac{q}{n}; c_2, c_3, c_4, c_5, c_7, c_8 = 0$$

Therefore,

$$x' = \left(\frac{p}{m}\right)x;\ y' = \left(\frac{q}{n}\right)y$$

If we try to normalize the thinned character skeleton, the resulting skeleton is not found to be connected. So, the unthinned character is normalized and later thinned.

Reverse mapping is done for normalization i.e.

1. For each pixel position P'[r][c], the equivalent pixel position P[i][j] is found using above relation (many pixel positions of Q may map to the same pixel position of P)
2. The value (0 or 1) at P'[r][c] pixel is set to that at the location P[i][j]

After normalization, we do the thinning as per the algorithm, given in Figure 11

Thinning algorithm

1. Scan binary bitmap in 4 directions: up-down, down-up, left-right, right-left.

2. In each pass, pixels which are not simple and have more than one eight neighbors (i.e. they are not end points) are deleted

3. Conditions for deletion of a point from a given side of the matrix should be checked before any other points are deleted. On each row points are marked for deletion, but are not deleted until the points on the following row have been marked.

4. The process iterates until no more pixels can be deleted.

//the resulting bitmap constitutes a skeleton of the original pattern i.e. a one-pixel wide pattern consisting of a subset of the original black pixels.

Fig.11.Thinning Algorithm

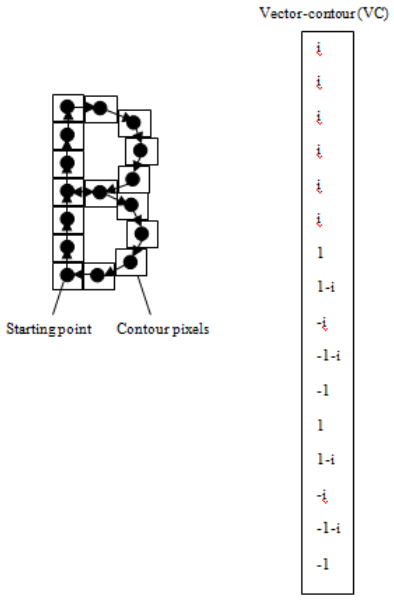

Fig.12.Vector Contour of 'B'

Phase2: Contour analysis of characters for its pre-classification

In this phase we do the contour analysis for each character starting from A to Z. The contour contains the necessary information on the object shape. We define contour as a boundary of an object, a population of pixels, separating object from a background. We have encoded contour by the sequence consisting of complex numbers. On a contour, the starting point is fixed. Starting point should be the centroid of the pixel. Then, the contour is scanned (is admissible – clockwise or anticlockwise depending on the continuity of the connectivity of the adjacent pixels), and each vector of offset is noted by a complex number a+ib. Where a is the point offset on x axis, and b is the offset on y axis. Offset is noted concerning the previous point.
To illustrate our approach for defining the vector contour of a character, we create the vector contour of 'B' as given in Fig 12. It is to be noted that from the starting point there is no adjacent pixel in clockwise direction, hence anticlockwise direction has been considered for deriving the vector contour.

So VC is the set of vectors, known as elementary vectors which define the contour of the character. Hence for character 'B', the VC defines as:
$VC_B = \{i, i, i, i, i, i, 1,\ 1-i, -i, -1-i, -1, 1, 1-i, -i, -1-i, -1\}$.

We have pre-classified all the capital letters based on their vector contours. Similarly we can compute the VC for all characters which are given below:

$VC_A = \{-1-i, -i, 1-i, 1+i, i, -1+i\}$.
$VC_B = \{i, i, i, i, i, i, 1,\ 1-i, -i, -1-i, -1, 1, 1-i, -i, -1-i, -1\}$.
$VC_C = \{-1, -1-i, -1-i, 1-i, 1-i, 1\}$.
$VC_D = \{-i, -i, -i, 1-i, 1+i, 1+i, i, -1+i, -1+i, -1-i\}$.

$VC_E = \{-1,-1,i,i,1,-1,i,i,1,1\}$.
$VC_F = \{i,i,1,-1,i,i,1,1\}$.
$VC_G = \{-1,-1-i,-1-i,1-i,1-i,1,i,-1\}$.
$VC_H = \{-i,-i,-i,-i,i,i,1,1,i,i,-i,-i,-i,-i\}$.
$VC_I = \{-1-i,-i,1-i,1-i,-i,-1-i,-1+i,i,1+i,1+i,i,-1+i\}$.
$VC_J = \{1,1,-1,-i,-i,-i,-i,-i-1,-1,-1+i\}$.
$VC_K = \{-i,-i,-i,-i,i,i,1+i,1+i,-1-i,-1-i,1-i,1-i\}$.
$VC_L = \{-i,-i,-i,-i,1,1\}$.
$VC_M = \{i,i,i,i,1-i,1-i,1+i,1+i,-i,-i,-i,-i\}$.
$VC_N = \{i,i,i,i,1-i,1-i,1-i,1-i,-i,-i,-i,-i\}$.
$VC_O = \{-1-i,-1-i,-i,1-i,1-i,1,1+i,1+i,i,i,-1+i,-1+i,-1\}$.
$VC_P = \{i,i,i,i,1+i,1-i,-1-i,-1+i\}$.
$VC_Q = \{-1-i,-1-i,-i,1-i,1-i,1,1+i,-1+i,1-i,1-i,1+i,i,i,-1+i,-1+i,-1\}$.
$VC_R = \{i,i,i,i,1+i,1-i,-1-i,-1+i,1-i,1-i,1-i\}$.
$VC_S = \{-1+i,-1+i,-1,-1-i,-1-i,1-i,1-i,1-i,-1-i,-1-i,-1,-1+i,-1+i\}$.
$VC_T = \{1,1,-1,-i,-i,-i,-i\}$.
$VC_U = \{-i,-i,-i,1-i,1,1+i,i,i,i\}$.
$VC_V = \{1-i,1-i,1-i,1+i,1+i,1+i,1+i\}$.
$VC_W = \{1-i,1-i,1-i,1+i,1+i,1+i,1+i,1+i,1-i,1-i,1-i,1+i,1+i,1+i,1+i\}$.
$VC_X = \{1-i,1-i,1-i,1-i,-1+i,-1+i,-1-i,-1-i,1+i,1+i,1+i,1+i\}$.
$VC_Y = \{1-i,1-i,1+i,1+i,-1-i,-1-i,-1-i,-1-i\}$.
$VC_Z = \{1,1,1,-1-i,-1-i,-1-i,1,1,1\}$.

VC character	No. of Vector elements	No. of bits required
A,C,L	6	3
B,Q	16	4
D,E,J	10	4
F,G,P,Y	8	3
H,W	14	4
I,K,M,N,X	12	4
O,S	13	4
R	11	4
T,V	7	3
U,Z	9	4

Table 1 VC of characters & its VE

We define the Vector Contour VC for characters as the set of vector contour of all characters.
$VC = \{VC_A, VC_B, VC_c, VC_D, VC_E, VC_F, VC_G, VC_H, VC_I, VC_J, VC_K, VC_L, VC_M, VC_N, VC_O, VC_P, VC_Q, VC_R, VC_S, VC_T, VC_U, VC_V, VC_W, VC_X, VC_Y, VC_Z\}$.

which is a two dimensional array. Table1 illustrates the number of vector elements in each of the VC set and hence the number of bits to represent them by knowing that 1 bit is sufficient to represent 21 states/values and 2 bits are sufficient to represent 22 states/values and so on…

Phase 3: Character embedding in circle

In this phase we circumscribe the character with a circle to create the boundary to be analyzed for its (character) identification.

Phase 4: Circle partition

In the previous phase, the centroid of each character pixel (black pixel 1's) has already been computed. Subsequently, in this phase, division of circumscribed circle into eight parts (arcs) with centroid is carried out so that each partition can take the range of 45 degrees, which has been assumed to be optimized value with respect to the computation and information integrity.

Fig.13.Circle partition with character 'A' enclosed

Table 2 describes the range of degrees for each partition in circle

Partition Number	Partition Range R (in degree)
1	$0 < R \leq 45$
2	$45 < R \leq 90$
3	$90 < R \leq 135$
4	$135 < R \leq 180$
5	$180 < R \leq 225$
6	$225 < R \leq 270$
7	$270 < R \leq 315$
8	$315 < R \leq 360$

Table2 Partition number and its range

Phase5: Data Normalization & character identification using Monte Carlo Method

We use Monte Carlo method as described in section2.6 for data normalization in which the following steps are performed:

In each partition of the circumscribed circle, count the number of pixel's (1's), say npi , where i is the partition number of circle. So range of i is from 1 to 8.

Count the total number of pixel's present in the character array, say npt.
Do normalization by performing $\frac{\mathrm{np_i}}{\mathrm{np_t}}$

We have shown the enclosed character 'B' with its partition in Fig 12.

In each partition, angle of each pixel is found which varies between particular partition's minimum range and maximum range. The summation of all angles of all the pixels of each partition is done and then normalization of this partition data vector is carried out. Then the centroid of the character i.e. the x and y co-ordinates of centroid is taken and normalized and is also considered as data for identification of characters. In all we got 10 partition data. This data is collected for all input characters set written by different individual. It uses the same pre-classification as described in Phase 2, using vector contour.

Phase 6: Back propagation Algorithm

We have preclassified the characters using VC in phase2. Then in phase 5, we extracted the data for each partition for the given character. In this phase, this partition data is fed as input to back propagation algorithm. Back propagation algorithm is given in Fig 14.

Back propagation algorithm

```
while(all [error(k)]<=)
{ for (each training pattern)
   { forward pass: calculate the output of 3 layers
   First layer;// fans out the i/p to next layer
   Output[1,i]=input[1,i];

   Second layer:
   input[2,j]=weight1[i,j]*o/p[1,i];
   output[2,j]=1.0/(1.0+exp(-input[2,j]);

   Third layer:
   input[3,k]=weight2(j,k)*output[2,j];

   compute error:
   error[k]=output[3,k]-target[k]

   Backward pass the error:
   δw2[j,k] = η*output[3,k]*(1-output[3,k])*error[k]*output[2,j]+α*δw2[j,k];

   output[3,k]*(1-output[3,k])*error[k]*weight2[j,k]}+ α*δw1[i,j];

   weight1[i,j]=weight1[i,j]-  δw1[i,j];
   weight2[j,j]=weight2[j,k]-  δw2[j,k];
   }
}
```

Fig.14.Back Propagation Algorithm

Networks are trained according to following Back propagation neural network classification which is based on the VC. given in Table 3.

Topology No.	No. of Input nodes	No. of hidden nodes	No. of output nodes	Character
1	26	20	3	A,C,L,F,G,P,Y,T,V
2	26	20	4	B,Q,D,E,J,H,W,I,K,M,N,X,O,S,R,U,Z

Table 3 Topology for EV

Reason for taking 26 input nodes in each topology is because from the character 26 circle data are extracted i.e. 26 features of characters are extracted. The number of output nodes has been derived from the number of bits required as shown in table1. As there is no clear rule for the 'best' number of hidden nodes, so as a good starting point and to get understood to the novice readers we assign the number of hidden nodes by the following formula:

$$\text{No. of hidden nodes} = \frac{\text{no. of input nodes} + \text{no. of output nodes}}{2}$$

The experienced mathematical researches can assign the number of hidden layers using Fisher Information Matrix [5]

Characters A,C,L,F,G,P,Y,T,V are trained with 26-20-3 configuration and rest of the characters are trained with 26-20-4 configuration.

Initially these networks are set up with weight generated from random number generator which may also affect the performance of the resulting Multi-layer Feed-Forward (MLF) ANN. These weights are adjusted during training to desired output as per the following procedure:

1. The input and output of each node (j) in the hidden and output layers are computed

$$I_j = \sum_i w_{ij}\, o_i + b_j$$
$$j = 1,2,3, \ldots 8\ (\text{no. of training data})$$

$w_{ij} -$ weight of the connection from node i in the previous layer to node j.
$o_i -$ Output of node i in the previous layer.
$b_j -$ bias of node j.

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

which is non linear and differentiable

2. Search for a set of weights that fits the training data such that the mean squared error (MSE) can be minimized, using gradient descent method. Error of node j,

$$Err_j = \begin{cases} o_j(1 - o_j)(T_j - o_j) (\text{j is an output node}) \\ o_j(1 - o_j) \sum_k e_k \,.\, w_{jk} (\text{j is a hidden node}) \end{cases}$$

o_j- the output of node j
T_j-the true target value of node j
e_k- the error of node j in the next layer
w_{jk}-the weight of connection from node j to node k
Change in weight $\Delta w_{ij} = (l) e_j o_i$
Change in bias $\Delta b_j = (l) e_j$
l- learning rate (a rule of thumb: $l = \frac{1}{t}$, t is the number of iterations so far

$$w_{ij} = w_{ij} + \Delta w_{ij}$$
$$b_j = b_j + \Delta b_j$$

3. Stop the process till all Δw_{ij} are below some specified threshold

So, in total 8 training files which consist of extracted partition data of character are trained and subsequently 8 weight files are generated.

Phase 7: Type classification

From table 3, it can be seen that there are some elementary vectors in set VC (Phase 2) which do have either horizontal or vertical or both symmetry.

VC_0, VC_1, VC_3, VC_8

So in VC, 3 sub groups lie:

1. Horizontal symmetry based sub group number 1: (VC_C, VC_E, VC_I)
2. No symmetry based sub group number 2: (VC_F, VC_G, VC_J, VC_L, VC_N, VC_S,VC_Z)
3. Vertical symmetry based sub group number 3: (VC_M, VC_U,VC_V, VC_W)

Since within the set of VC, characters can be placed in 3 subgroups as discussed above. So there is need for two stage back propagation neural classifiers, which is described below:

A neural network having d input nodes, c output nodes can be considered as a classifier to assign a given sample with d feature to one of 3 predefined sub group numbers.

During recognition of unknown character it goes through 2 stage process. During its first stage type classifier gets activated which indicate whether the character is from horizontal or no symmetry or vertical symmetry subgroup. Then according to the identification of subgroup from first stage, in second stage particular type recognition network (type 1 or type 2 or type 3) gets activated which finally gives true identity of character.

4.PROS AND CONS OF TWO STAGE BACK PROPAGATION NEURAL CLASSIFIER

There are two main advantages of using 2-stage Back propagation neural classifier:

1. A single problem can be decomposed into many small sub-problems, which are easier to manage.
2. The output node of the ANN can be reduced for efficiency purpose

If we look into the disadvantages, we could found only one that is if the first classifier fails to interpret correctly, that wrong output of the first classifier will become the input of the second classifier, which will give the wrong prediction. But that can always be minimized by making the use of efficient training algorithm to train the hidden layer of ANN.

5.EXPERIMENTAL RESULTS

5.1.Sample Text Files

Some test documents containing character sets written by different individuals in their handwriting was collected and scanned using scanner to generate corresponding tiff file and corresponding to which binary files were formed and processed. First lines were separated out from text followed by separation of each character which were then enclosed in a bounding box, binaries, normalized and thinned using a thinning algorithm.

Some test documents containing phrases on different lines were tested during experiments.

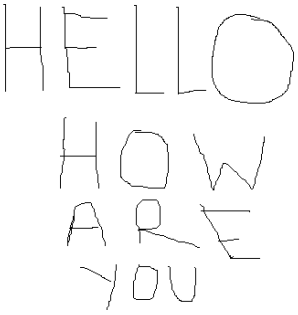

Fig.15 Sample Text File1

We collected character set in different handwritings from A-Z in a file.

5.2 Preprocessing of characters

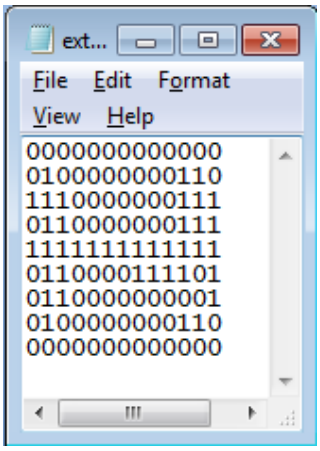

Fig.16 'H' extracted from input text file

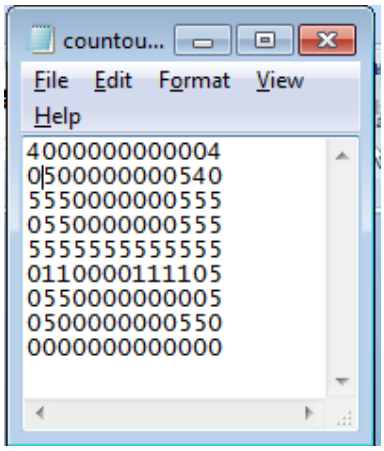

Fig.17 Feature extraction (after thinning)

5.3 Normalized circle data of file

Normalized circle data of file for 'H' is only shown
0.086957 0.152174 0.196552 0.326078 0.108696 0.134351 0.134509 0.156900 0.189036 0.118818 0.153421 0.247432 0.339870 0.077045 0.066120

Several experiments were carried out to demonstrate the performance of this data. For training and testing total number of samples taken from different persons is shown in Table 4

Subgroup number	1	2	3
Training sample number	100	147	123

Table4. Data from different persons

Total number of samples for training = 370

Recognition rate of this approach found is 76%

6.LIMITATIONS

There are some limitations to our approach for character identification which are give as under:

1. Sometimes the person can write the characters which are not purely separated due to hurry-burry or the individual's style of writing. In that case our Vector contour will not be appropriate and our system would have to train for large data sets and for longer duration to predict the given character correctly.
2. We have partitioned the circumscribed circle into 8 parts. As the number of partitions increases, the prediction would be more accurate but the complexity will be increased and the designer must have to analyze the proper balancing in between the number of partitions and its complexity.
3. We have used MLF ANN, for which the training time can be very lengthy.
4. There is a need of rather a relatively large dataset for training to get accurate predictions.

7.FUTURE WORK AND CONCLUSION

In this paper, we introduced a novel set of features that is well-suited for representing handwritten characters. The features are derived from the Vector Contour (VC) set and symmetrical nature of the characters. For the sake of simplicity during the VC analysis we assumed the value of a and b in a+ib as unity though actual value have to be trained by collecting large data set. The feed forward neural network was trained for VC and symmetrical nature identification which collaboratively is the very unique feature for character, even VC itself is the most unique feature, if the values of a & b are given correctly. Our character recognition system can also be used for numeral recognition.

We have introduced a mechanism for handwritten character recognition and all the related algorithms. Our future work will be to implement our approach to show the experimental results for evidence, which is lacking in this paper and shall be illustrated in our next paper.

REFERENCES

[1] S K Hasnain, Azam Beg and Samiullah Awan, "Frequency Analysis of Urdu spoken Numbers Using MATLAB and Simulink" Journal of Science & Technology PAF KIET ISSN No 1994-862x, Karachi, Nov. 2007.

[2] A. Beg, P.W.C. Prasad, S.M.N.A. Senanayake. Learning Monte Carlo Data for Circuit Path Length. In Proc. International Conference on Computers, Communications & Control Technologies, CCCT 2007, Orlando, Florida, July 12-14, 2007.

[3] A. Beg, P.W.C. Prasad, M. Arshad, S. K. Hasnain. Using Recurrent Neural Networks for Circuit Complexity Modeling", In Proc. IEEE INMIC Conference, Islamabad, Pakistan, December 23-24, 2006, pp. 194-197.

[4] Y. Le Cun et al., "Constrained Neural Network for Unconstrained Handwritten Digit Recognition," Proc. of 1st Int. Workshop on Frontiers in Handwriting Recognition, Montreal, Canada, Apr. 1990, pp. 145-154.

[5] Fletcher el, "Optimizing the Number of Hidden Nodes of a Feed forward Artificial Neural Network", Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence.

[6] S. G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," IEI7E Trans. on Pattern Analysis and Machine Intelligence, Vol. ll, No. 7, pp. 674-693, 1989.

[7] L. Stringa, "A new set of constraint-free character recognition grammars," IEEE Trans. Pattern Anal. Machine Intell, vol. 12, pp. 1210–1217, 1990.

[8] L. K. Hansen and P. Salamon, "Neural-network ensembles," IEEE Trans. Pattern Anal. Machine Intell, vol. 12, pp. 993–1001, 1990.

[9] Lam, L., Lee, S.W., Suen, C.Y., "Thinning Methodologies- A Comprehensive Survey", IEEE Trans. On PAMI, Vol. 14, No. 9, pp. 869-885, 1992.

[10] Lam and Suen, "An evaluation of parallel thinning algorithm for character recognition, "IEEE Trans. On PAMI, Vol. 17, No.9, pp. 914-919, 1995.
[11] W. K. Pratt, Digital Image Processing. New York: Wiley, 1978.
[12] Lu H.E., Wang P.S.P., "A comment on fast parallel algorithm for thinning digital patterns", Communications of the ACM, Vol. 29, No.3, pp. 239-242, 1986.
[13] Naccache N.J., Shinghal R., "SPTA – A Proposed Algorithm for Thinning Binary Patterns", Communications of the ACM, Vol. 29, No.3, pp. 239-242, 1986.
[14] Saha P.K et al, " A single scan Boundary Removal Thinning Algorithm for 2-D Binary Object", Pattern Recognition, Vol. 14, No. 3 pp. 173-179, 1993.
[15] Jairo and Pavlidis, "Character Recognition Without Segmentation", IEEE trans. on Pattern Analysis and Machine Intelligence, Vol. 17. No. 9, Sept 1995.
[16] Rhee et al., "Utilizing Consistency Context for Handwritten Mathematical Expression Recognition", 2009 10th International Conference on Document Analysis and Recognition.
[17] S. Tsujimoto and H. Asada, "Resolving ambiguity in segmenting touching characters," First Int'l Con$ Document Analysis and Recog- nition, pp. 701-709, Saint-Marlo, France, Sept. 1991.