Devnagari Handwriting Recognition System using Dynamic Time Warping Algorithm

Bhushan C. Bhokse
Sr.Lecturer,
Department of InformationTechnology,
MAEER's Maharashtra Academy Of Engineering
Kondhwa,Pune

Bhushan S.Thakare
Assistant Professor,
Department of Computer Engineering,
STES Sinhgad Academy Of Engineering,
Alandi (D),Pune

.

ABSTRACT

Online handwriting recognition is gaining renewed interest owing to the increase of pen computing applications and new pen input devices. The recognition of Devnagari characters is different from western handwriting recognition and poses a special challenge. This paper is an overview of the technical status and the development in online Devnagari recognition system which starts few years back.
The objective of this paper is to develop a system which can recognize a handwritten Devnagari character, written with the help of stylus or a digital pen. To develop the system, dynamic time warping (DTW) technique is considered. In dynamic time warping the entered character is processed in terms of time sequences x and y as a function of time. The time sequence of the character to be recognized is compared with that of all the characters stored in a code file. The distance between them is calculated. The character from the code file which gives minimum distance is considered as the recognized character. In order to improve the recognition rate, along with DTW algorithm, additional feature vectors can be used. In this paper, line extraction algorithm is developed.

Keywords

HWR,HMM, PDAs ,DTW, VLSI, IDE, JVM, J2SE, OCR, PCA.

1. INTRODUCTION

In this paper, it is explored the efficiency of various stroke-based handwriting analysis strategies in classifying Devnagari handwritten characters by using a template-based approach. Writing units are variable from time to time, even within the drawings of a specific character from the same user. Writing units include the properties of stroke such as, number, shape and size, order and writing speed. It is proposed to use structural properties of writing samples having such variability in writing units. This work employs a "Dynamic Time Warping" (DTW) algorithm to align two on-line handwritten strokes and to estimate the similarity and use two different features for stroke identification, a sequence of direction at every pen-tip position along the pen trajectory and inclusion of pen-tip position with the direction as the feature of the stroke. For each type of feature, two different systems are trained by using the samples collected from the i) same writer, ii) group of writers. To evaluate the system, it is collected examples of 40 different characters from 5 different writers, and then performed a series of different experiments. Use of specific-stroke pre-processing and a sequence of both pen-tip position and slope at every position as a feature of a stroke, yield improved results, the superiority of the present work over several related works on Devnagari script is the recognition of stroke number and stroke order free natural handwritten characters.

This paper describes a system for automatic recognition of handwritten Devnagari characters obtained by linearising characters. Owing to the large number of characters and resulting demand on data acquisition, structural recognition technique is used to reduce some characters to others. The residual characters are then classified using the dynamic time warping (DTW). Finally the results of structural recognition and feature based matching are mapped to give final output.

1.1.On-line versus Off-line Handwriting Recognition

On-line handwriting recognition means that the machine recognizes the writing while the user writes. The term 'real time' or 'dynamic' has been used in place of online. Depending on the recognition technique and the speed of the computer, the recognition lags behind the writing to a greater or lesser extent. Most commercial character recognizers lag by only one or two characters. On-line recognition systems need only be fast enough to keep up with the writing.
Figure 1 shows generalized handwriting recognition. With each block containing one complete module used to process the character. It shows the general flow of data in its intermediate processing stages.

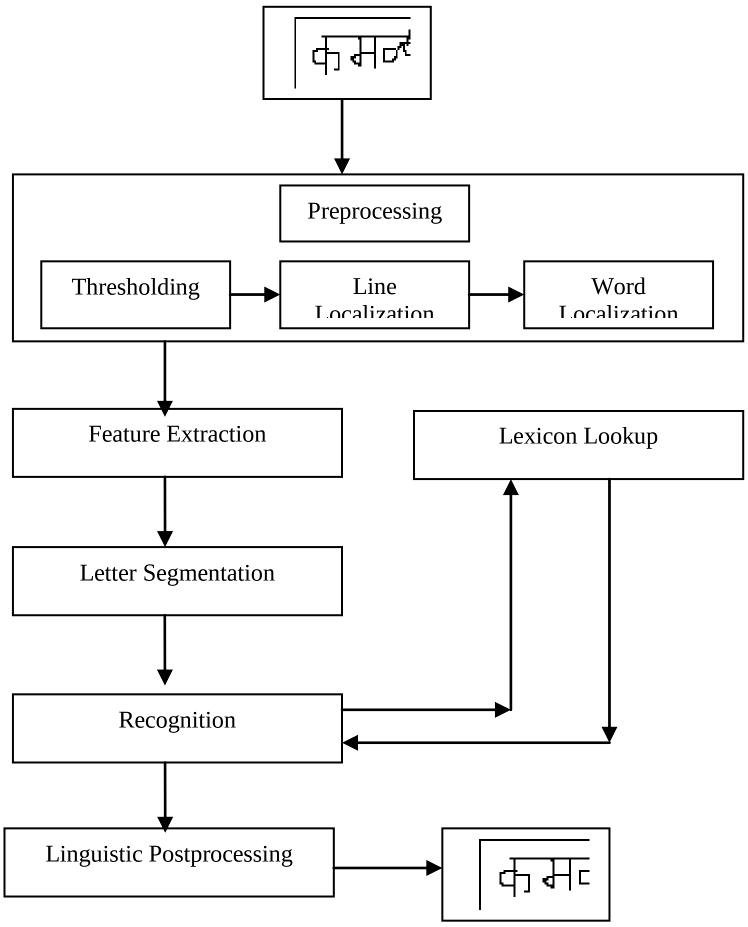
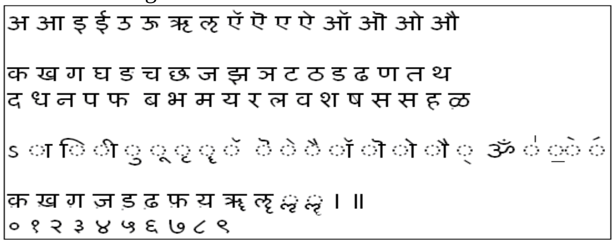


Fig 1: A Generalized Handwriting Recognition System

2. FEATURES OF DEVNAGARI SCRIPT

Vowels, Consonants and modifiers in Devnagari script are shown in the figure 2.


अ आ इ ई उ ऊ ऋ ऌ ऍ ऎ ए ऐ ऑ ऒ ओ औ

क ख ग घ ङ च छ ज झ ञ ट ठ ड ढ ण त थ
द ध न प फ ब भ म य र ल व श ष स स ह ळ

ऽ ◌ा ◌ि ◌ी ◌ु ◌ू ◌ृ ◌ॄ ◌ॅ ◌ॆ ◌े ◌ै ◌ॉ ◌ॊ ◌ो ◌ौ ◌् ॐ ◌॑ ◌॒ ◌॓ ◌॔

क़ ख़ ग़ ज़ ड़ ढ़ फ़ य़ ॠ ॡ ◌ॢ ◌ॣ । ॥
० १ २ ३ ४ ५ ६ ७ ८ ९

Fig 2: Vowels, Consonants and Modifiers in Devnagari Script

Some of the features of Devnagari script are listed below which helps us in proper understanding of the script [4].

1. **Akhand ligatures** - Required consonant ligatures that may appear anywhere in the syllable, and may or may not involve the base glyph. Akhand ligatures have the highest priority and are formed first; some languages include them in their alphabets. Akhand ligatures may be in either half or full or other form.
2. **Base glyph** - The only consonant or consonant conjunct in the syllable that is written in its "full" (nominal) form. In Devnagari, the last consonant of the syllable (except for syllables ending with letter "Ra") usually forms the base glyph; in other Indic scripts, the first consonant or conjunct (e.g. Telugu), or others, may form the base glyph. In "degenerate" syllables that have no vowel (last letter of a word), the last consonant in halant form serves as the base consonant and is mapped as the base glyph. Layout operations are defined in terms of a base glyph, not a base character, since the base can often be a ligature.
3. **Below-base form of consonants** - The form that consonants appear below the base glyph. Consonants in below-base form appear in Indic syllables after the ones that form the base glyph. Below-base forms are represented by the non-spacing mark glyph.
4. **Consonant** - Each represents a single consonant sound. Consonants may exist in different contextual forms, and have an inherent vowel (usually, the short vowel "a"). Therefore, those illustrated in the examples below are named, for example, "Ka" and "Ta", rather than just "K" or "T".
5. **Consonant conjuncts (aka 'conjuncts')** - Ligatures of two or more consonants. Consonant conjuncts may have both full and half forms, or only full forms.
6. **Halant (Virama)** - The character used after a consonant to "strip" it of its inherent vowel. A Virama follows all but the last consonant in every Indic syllable, except in languages like Sanskrit, Tamil, and Malayalam, where the last consonant may also have a Virama.
7. **Note:** A syllable containing halant characters may be shaped with no visible halant signs by using different consonant forms or conjuncts instead.
8. **Halant form of consonants** - The form produced by adding the Virama to the nominal shape. The Halant form is used in syllables that have no vowel, or as the half form when no distinct shape for the half form exists. In some scripts (Bengali, Tamil), the half and halant forms are always the same.

3. SYSTEM DEVELOPMENT

This paper describes the development of a recognition engine for online handwritten Devnagari characters. Devnagari script is a logical composition of its constituent symbols in two dimensions. It has eleven vowels and thirty three simple consonants. A horizontal line is drawn on top of all characters which is referred to as the header line or 'shirorekha'. A character is usually written such that it is vertically separate from its neighbors. Devnagari script has many multi-stroke characters. The data entry/ recognition mechanisms need to deal with such multi-stroke characters and also conjuncts that are made up by joining two or more characters partially.

In this paper, segmented characters are assumed at the data entry level. The database collected for training the system is acquired in individual boxes for each character without any conjuncts. Some of the constraints imposed both on the handwriting input and the recognition methodology in order to simplify the complexity of data acquisition, and subsequent training and testing. The process used for acquiring handwriting data for training and testing the system is discussed. Next the process used for training the system using unsupervised learning method is discussed.

In order to reduce the complexity of the problem, only the characters in their simple form without any additional modifiers are considered for the training and the recognition purposes. It was further assumed that the characters will be entered without header line or 'shirorekha'.

3.1. System Model

Figure 3 shows the system model used for creating the Devnagari handwriting recognition system to recognize the handwritten Devnagari characters. The working of the proposed system and the data flow in the system can be explained with the help of these blocks. The very first block Data collection tool is developed to create and store the handwritten Devnagari samples in the form of distinct strokes. A single character may contain one or more number of strokes. As the system is considering the recognition of individual characters only, the strokes for a single character are joined, so as to convert distinct strokes of a single character into a single stroked character. Finally the data is stored in the form of (x, y, t) arrays in the '.mat' file. The data is collected through the DCT developed in Matlab7.01.

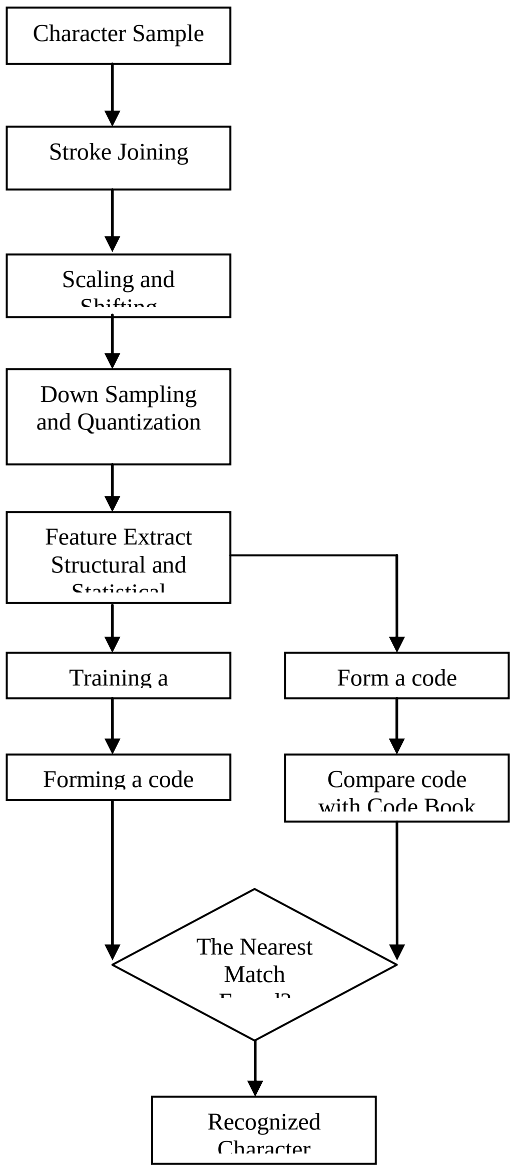


Fig 3: System Model

As a single frame contains 25 character samples, there co-ordinate points are different, in order to keep same reference point *i.e.* origin for all samples, the samples are shifted after the scaling operation. Scaling is performed to keep the unique size for all the characters. The sample after scaling and shifting is downsampled to reduce the number of sampe points then it is quantized to the nearest integer points. To develop classifier it is needed to extract the features (structural and statistical) of every sample used for training the system. In the training process for every character a code vector is formed and is stored in the code book matrix. In the recognition step, the same operations are carried over on the entered character upto forming a code vector. Then that code is compared with each entry in the code matrix. The entry whichever is close to the code vector is considered as the recognized character.

4. SYSTEM DEVELOPMENT USING SUPERVISED LEARNING

4.1 Data Collection Tool

The data for training and testing was acquired through an iBall pen, using a GUI developed using MATLAB7.01. Each page of the GUI contains 25 writing blocks in which characters are to be written in an isolated manner.

The data collection tool shown in the figure 4, on its top-left corner shows letter 'k', indicating the writer to enter 'k' letter 25 times in the boxes formed by the grid lines. The blue lines are plotted to guide the writer while writing Devnagari letters and these lines are where the 'Shirorekha' is to be placed. To make the work of pre-processing and later recognition of the character easy, the writers are said, not to draw the shirorekha. The font used to draw 'k' is 'kailash'. It is a true type font and is created by using the Font Creator software.

For the purpose of drawing the handwritten character, the canvas is formed by using the 'axes control'. As explained above the entire axis is divided into 25 parts so that through one screen 25 samples can be entered. As the digital pen moves over this canvas when writer writes, the co-ordinates x and y are get stored continuously alongwith the time parameter 't'. Every stroke is treated as an array of (x, y, t) which starts when 'pen is down', spans over the length when the 'pen is in motion' and ends when 'pen is up'. All these events are handle by the button events of the language. The data for this one window, 25 characters are stored as an array of structure, with array elements an individual stroke, and structure elements containing x, y, time arrays of each stroke. A single character may contain one or more number of strokes.

Steps for data collection:

```
1.  For character = 1 to 40
2.  Wait for character entry
3.  While buttondown = no go 2
4.  Record the first point of the stroke
5.  While buttonmotion = no go 6
6.  Successive points of the stroke are recorded
7.  While buttonup
8.  Wtore all the points of stroke, stroke number
9.  If additional stroke needs to complete character go 2
10. Store the character as group of strokes, each stroke
    with stroke array
11. Performs step 1 to 9 to collect all 25 samples of
    same character
12. Store the character into character array
```

13. Store the entire character as a letterno file on disk
14. Move to next character (press 'next' button)
15. Stop

In this fashion the data is collected for all the 40 characters used for recognition. The 40 characters used in this system are shown in the following figure 5. The characters shown are collected with the same data collection tool explained above.

For the purpose of training & testing, character samples were collected in two different sets. i) From a single writer, twenty five instances of each character were collected. From this the best 6 samples per character are selected, ii) from each of 5 writers corresponding to each of the 40 characters, twenty five instances of each character were collected. From this the best 6 samples per character are selected.

From each of these sets, the first instances are used for the purpose of training and the rest five are used for testing purpose.

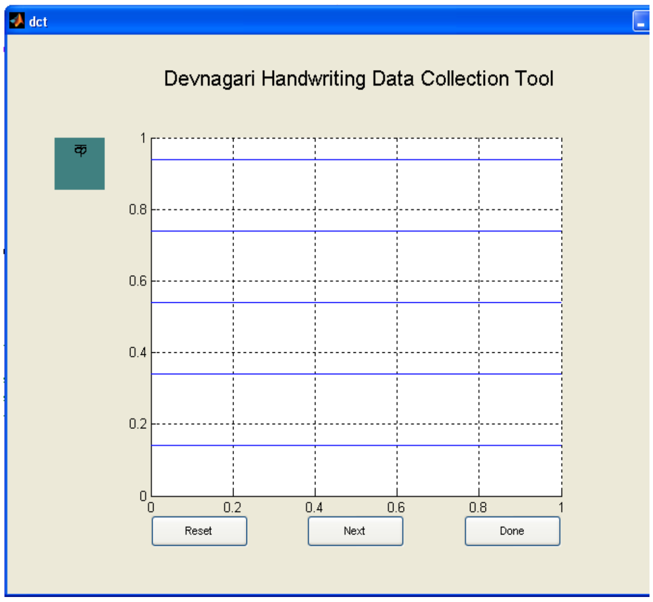


Fig 4: Data Collection Tool

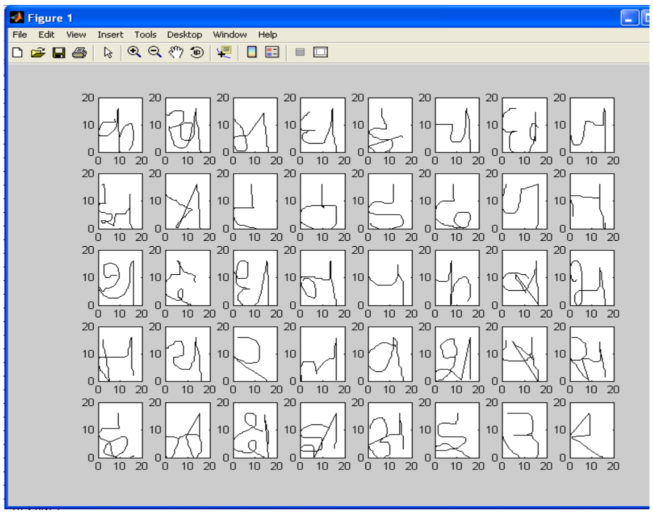


Fig 5: The Devnagari Character Set Used in The System

4.2 Strokes Joining

The strokes of individual characters within one grid box are determined first and then joined together to represent the multi-stroke character as a single stroke character. The character is represented and further processed in the form of a signal and not an image. Due to this reason the strokes are joined together.

Steps for strokes joining:

1. For character = 1 to 40
2. For sample = 1 to 25
3. Read the character.
4. If not multistroke go 6
5. Join the stroke
6. Move to next sample
7. Store all samples for a single character
8. Move to next character
9. Stop

4.3 Scaling and Shifting

After joining the character (required in the case of multi-stroke character), the character is cropped out and scaled so as to represent a unique size throughout all the characters. As the data is collected in the window at different positions, for the comparison it is needed to at same position with respect to origin. This is done by shifting it to make the origin as a reference point.

The character contains (xPoints, yPoints) as (x, y) co-ordinates. Width and height of the character is determined as
Width = maximum(xPoints)-minimum(xPoints)
Height = maximum(yPoints)-minimum(yPoints)
The data is normalized by dividing xPoints with width and yPoints with height.
This normalized character is of size 1 square unit. The normalized points are then multiplied by 16 to get size of '16 x 16' square unit for each character.
In scaling operation the point (minimum(xPoints), minimum(yPoints)) is shifted to the origin (0,0) by performing the operation.
For all (x, y) points
[xPoints;yPoints]=[xPoints;yPoints].–[min(xPoints);min(yPoints)]

Steps for scaling and shifting:

1. For character = 1 to 40
2. For sample = 1 to 25
3. Read the sample
4. dx = xmax - xmin
5. dy = ymax - ymin
6. x = x / dx
7. y = y / dy
8. x = 16 * x
9. y= 16 * y
10. Move to next sample
11. Store all samples for a single character
12. Move to next character
13. Stop

4.4. Down Sampling and Quantization

The collected sample points for each characters are re-sampled once, twice and thrice to reduce the number of sample points. The samples are down sampled in time domain. Here in the system different number of characters may contain different number of sample points. What is important is the positions of points and the direction in which they moves. The quantization operation is performed straightly in terms of rounding the values of x and y co-ordinates.

Steps for down sampling and quantization:

1. For character = 1 to 40
2. For sample = 1 to 25

```
3.  Read the sample
4.  Sample = sample(1:2:samplelength) ;alternate (x, y)
    points
5.  Sample = sample(1:2:samplelength) ;alternate (x, y)
    points
6.  Sample = sample(1:2:samplelength) ;alternate (x, y)
    points
7.  Round off the samplepoints
8.  Move to next sample
9.  Store all samples for a single character
10. Move to next character
11. Stop
```

4.5 Feature Extraction

The down sampled and quantized input character pattern represented by a sequence of x-y coordinates is preprocessed to extract structural features at the stroke level, such as mean (x, y) value, mean (x, y) value of all 4 segmentsby dividing the character into four parts, character length , directional codes *etc.*

The character pattern is divided into four parts. For each part mean x and mean y are calculated and stored as {mean1x, mean2x, mean3x, mean4x} and {mean1y, mean2y, mean3y, mean4y}.

The length feature is extracted in the form of horizontal distance, vertical distance traveled while writing the character. This is a scalar quantity. One additional feature is extracted in the form of vector quantity. It is the horizontal and vertical displacement traveled while writing the character.

The directional codes are formed in the form of the total directions moved while writing the letter. It includes vertical direction changes *i.e.* towards 'North' & 'South' and horizontal direction changes *i.e.* towards 'West' & 'South'.

The structural features are also extracted in the form that the letter contains complete vertical line, half vertical line, complete horizontal line and half horizontal line. These features are extracted by using the principle 'dx = 0' *i.e.* difference is zero along horizontal direction, indicates it is a vertical line, if successive dx are zeros (in this system needs at least 15). This is the case for half vertical line too, which spans over only half of the vertical line. Similarly 'dy = 0' successively gives a horizontal line and half horizontal line. The Devnagari characters contain various curves in them. To extract these features work by taking double differences along x and y directions or convert the character into a linear piecewise segments and then determine their slope.

```
Steps for feature extract: finding vertical line
1.  Define location[], v_points[]
2.  Read character
3.  Take first order difference
            dx = xi+1 – xi
4.  For i = 1 to numel(dx)
        locate first dx = 0
        whiel dx(i) ~= 0
            i = i +1
        end
        location= [location i]
        k=1; counter
        whiel dx(i) == 0
            i = i +1
            k=k+1
        end
        v_points = [v_points k]
    End
5.  From location & v_points locate vertical line

Steps for feature extract: finding horizontal line:
    Same as algorithm no 5. Instead of dx find dy
```

4.6 Training

To train the classifier a simple approach is adapted. From the character set used for training, each individual character is segmented into four parts. Means of x and y co-ordinates are determined separately to get a feature vector containing eight elements as explained above. The feature vector of every character is normalized by multiplying it with normalization factor.

Normalization factor= (vector length of the feature vector of the character)$^{-1}$
Where vector length is calculated as
Vector length = sqrt(sum(square of each element in feature vector))
The normalized feature vector of each character together forms a normalized feature matrix for the entire character set. This normalized feature matrix is stored in the file as a 'codebook'. When the system starts, this codebook stored in the '.mat' file, first get loaded into the RAM and is available there onwards for the comparison purpose of the entered character with the entries stored in the codebook.

```
Stpes for forming code book with the mean of the four
segments:
For character = 1 to 40
    Read character
    Divide it into four segments
    Find the mean of each segment
    Normalize the mean
    Store into codebook
End
```

5. TESTING AND RECOGNITION

To test the recognition system another set of character named as 'test character set' is considered. For the purpose of testing the same initial steps as in training are applied on the 'test character set'. Every character is tested for the purpose of its recognition against the 'codebook'.

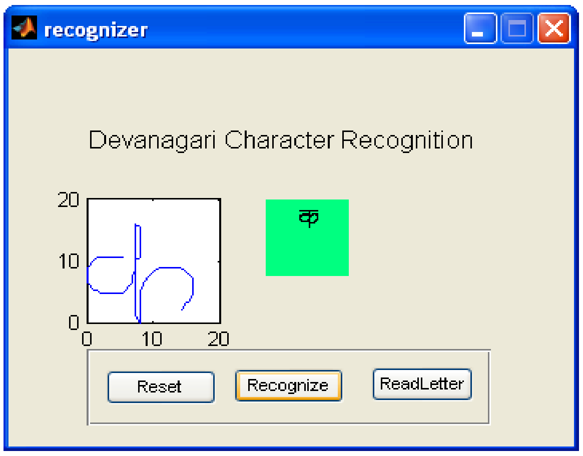


Fig 6: GUI for Character Testing and Recognition

Another provision is made for the online recognition of character. Here after writing the character, the handwritten character will go through the entire process like strokes joining, scaling and shifting, down sampling and quantization, feature extraction, formation of codes *etc.* and then a code is formed containing normalized means. This code is then compared with each entry called as code vector in the code matrix formed in the training process. As the system is not rejecting any entry, the nearest match found is considered as a recognized one, and a character is mapped for that code and displayed on the graphical user interface as a recognized character.

6. CONCLUSIONS

Many challenges need to be overcome in the field of Devnagri online handwriting recognition, if handwriting is to evolve into a reliable method of data entry. In particular, a recognition system must be able to accommodate the large variations that exist between writing styles. The Devnagari character is written with multiple strokes. In a sentence along with the multiple strokes, presents 'shirorekha' and modifiers to the character. Though this system is developed only for the recognition of individual characters, it is prepared with intention that it can be easily extended for the word recognition and then for the recognition of entire Devnagri script. The system is developed not for specific group of users. It requires large learning set than a system that tries to match handwriting from an arbitrary writer. It is a writer independent handwriting system.

- The technique to compare the time sequences-Dynamic Time Warping is studied and implemented. Its application to the Devnagari character recognition is worked out by applying it in different ways *i.e*

  i) Applying DTW directly to the x and y-time sequences of a character,
  ii) Applying DTW to the means of the segments of the character.

  The technique gives a adequate accuracy 64% when applied directly on the time sequences compared to its application to the means of the segments where it is 48.5%. The accuracy or recognition rate can be further improved if one use more number of classifiers along with the DTW technique or apply DTW on the feature vector of classifier.

- Euclidean distance between the means of the segments is taken in order to find the minimum distance. Taking Euclidean distance instead of DTW on the means of the segments produces the results which are less accurate than DTW techniques. In this case also the results can be improved by using more number of classifiers. The results obtained here can be improved if the number of segments is increased and weight is applied to every segment.

- In both the technique above to improve the accuracy or recognition rate one needs additional classifier which needs additional feature vector. To get additional feature vector we need feature extraction algorithms which could transfer the curved stroke into piecewise linear segments. The attempt is made and is successful for obtaining the vertical line and horizontal line using the difference operator. Further to get the features from the curved surface double difference technique with the help of Laplacian operator is to be performed. In addition to this a piecewise linear segmentation followed by linear regression is worked out to fit straight lines and non-linear curves for the character segments. This is a form of supervised learning.

7. REFERENCES

[1] C. C. Tappert, C. Y. Suen and T. Wakahara, "The State of the Art in On-Line Handwriting Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No.8, 1990, pp. 787-808.

[2] Gareth Loudon, Olle Pellijeff, Li zhong-wei, "A Method for Handwriting Input and Correction on Smart Phones", Proceedings IAPR-IWFHR-2000, eldoc.ub.rug.nl/files/home/IAPR_IWFHR_2000/Posters/p04/poster-001.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification", Second Edition, John Wiley & Sons Inc, New York, 2006, pp. 115,128,259,582.

[4] Anil K. Jain, Robert P.W. and Jianchang Mao, "Statistical Pattern Recognition: A Review", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 1, January 2000, pp. 4-7.

[5] Bharath A, Sriganesh Madhvanath, "Hidden Markov Models for Online Handwritten Tamil Word Recognition", http://www.hpl.hp.co.uk/techreports /2007 /HPL-2007-108.pdf.

[6] Jianying Hu, Michael K. Brown and William Turin, "HMM Based On-Line Handwriting Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No.10, October 1996, pp.1039-1045.

[7] Deepu Vijayasenan and Sriganesh Madhvanath, "Principal Component Analysis for Online Handwritten Character Recognition", Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004),August 2004, No. 2, pp. 327-330.

[8] L. Rabiner and B.H. Juang, "Fundamental of Speech recognition", First Edition, Pearson Education India, 1993, pp 257-261.

[9] C. Bahlmann and H. Burkhardt, "The Writer Inependent Online Handwriting Recognition System Frog on Hand and Cluster Generative Statistical Dynamic Time Warping", IEEE transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No.3, March 2004, pp. 299-309.

[10] http://www.heatonresearch.com/articles/series/1, access date: May 15, 2008.

[11] http://en.wikipedia.org/wiki/Handwriting_recognition, access date: July 17, 2007.

[12] http://www.microsoft.com/windosxp/tabletpc/default.mspx, access date: August 7, 2007.

[13] Sriganesh Madhavnath, Deepu Vijaysenan and Thanigai M.K, "LipiTk: A Generic Toolkit for Online Handwriting Recognition", International Conference on Computer Graphics and Interactive Techniques, No. 13, 2007.

[14] R.M.K. Sinha and Veena Bansal, "On Devanagari Document Processing", in Proceedings of International Conference on Systems, Man and Cybernetics, Vancouver, BC, October, 1995, pp. 1621–1626.

[15] V. Bansal and R.M.K. Sinha, "Segmentation of Touching Characters in Devanagari", Technical Report,

Department of Computer Science and Engineering, IIT Kanpur, India, http://www.iitk.ac.in/ime/veena/PAPERS/stwo.pdf .

[16] V. Bansal and R.M.K. Sinha, "On how to Describe Shapes of Devanagari Characters and use them for Recognition", in Proceedings of 5th International Conference Document Analysis and Recognition, Banglore, India, September1999, pp. 410–413.

[17] S. D. Connell, R.M.K. Sinha and A. K.Jain, "Recognition of Unconstrained On-Line Devanagari Characters", in Proceedings of 15th International Conference on Pattern Recognition, September 2000, pp. 368–371.

[18] Niranjan Joshi, G. Sita, A. G. Ramakrishnan and Sriganesh Madhvanath, "Machine Recognition of Online Handwritten Devanagari Characters", Proceedings. Eighth International Conference on Document Analysis and Recognition, Vol. 2, September 2005, pp. 1156-1160.

[19] Santosh K.C. and Cholwich Nattee, "Structural Approach on Writer Independent Nepalese Natural Handwriting Recognition", IEEE 2006.

[20] R. J. Ramteke & S. C. Mehrotra, "Feature Extraction Based on Moment Invariants for Handwriting Recognition", IEEE 2006.

[21] Dr. P. S. Deshpande, Mrs. Latesh Malik and Mrs. Sandhya Arora, "Recognition of Hand Written Devnagari Characters with Percentage Component Regular Expression Matching and Classification Tree", IEEE 2007.

[22] http://www.mathworks.com/products/matlab/description1.html, access date: August 28, 2007.

[23] http://java.sun.com/docs/overviews/java/java-overview-1.html, access date: September 2, 2007.

[24] Lawrence R. Rabiner, Aaron E. Rosenberg, Stephen E. Levinson, "Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-26, No. 6, December 1978, pp. 575-582.

[25] Patrick Naughton and Herbert Schildt, "The Complete Reference Java 2", Fifth Edittion, TMH India 2005, pp. 158-170.