Multilingual Online Handwriting Recognition System: An Android App

Indhu T R Vidya V Bhadran V K

Center for Development of Advanced Computing
Thiruvananthapuram
{indhu, vidyav,bhadran}@cdac.in

***Abstract*— On-line handwriting recognition means recognizing the user's handwriting as the user is writing the character/stroke, i.e. the recognition is concurrent to the writing process. This paper describes a Multilingual Online handwriting recognition system in Android using SFAM Artificial Neural Network Technique. The app allows the user to select any of the target languages supported. The handwriting sequences are collected by the digitization of the pen/stylus movements as an array of x,y coordinates which is called a stroke. The structural and directional information are extracted from each character/stroke. The extracted features as a feature vector are passed as input to a SFAM artificial neural network (ANN) classifier. The SFAM classifier then performs a comparison of the input data with the trained data and finds the nearest prototype from the database that 'resonates' with the input pattern. The labels/recognized characters are assigned their corresponding Unicode code points and displayed using appropriate fonts.**

Keywords- Multilingual, Online Handwriting Recognition, Android, SFAM, Neural network classifier

I. INTRODUCTION

Handwriting recognition has a wide variety of applications. In applications where handwritten reports are required, an instant digital conversion to text will reduce huge costs as well increase productivity. OLCR are now available for different languages. These engines normally work on specific interface by the developer and do not allow the users to switch over to a new language. In Indian perspective, especially with the launch of low cost tablets, it is always better to have a generic interface for all languages onto which different OLCR engine can be plugged-in based on the requirement. Handwriting recognition requires intelligent signal processing to extract the most appropriate features, machine learning and natural language processing, incorporating domain knowledge and contextual prediction.

In this paper we propose a language independent engine so that language specific features of any language can be brought in as Feature Set and Classification set. This is an extension of the work [1] to multilingual framework. The framework contains generic processing modules like capture, preprocessing, feature extraction, classification and recognition. The language dependency can be handled through trained feature DB. The post processing will be always language dependent since it is at this point that language specific rules are applied for conflict resolution. [2][3][4][5][6][7] show some of the existing work in the field of online handwriting recognition. A generic design have been discussed in [8] with a stroke level accuracy of 95.78% and 95.12% on Malayalam and Telugu data, respectively.

The outline of this paper is as follows: The proposed architecture is given in section 2, system description in section 3. An overview of Android architecture is mentioned in section 4. Section 5 summarizes the result of the work done.

II. SYSTEM ARCHITECTURE

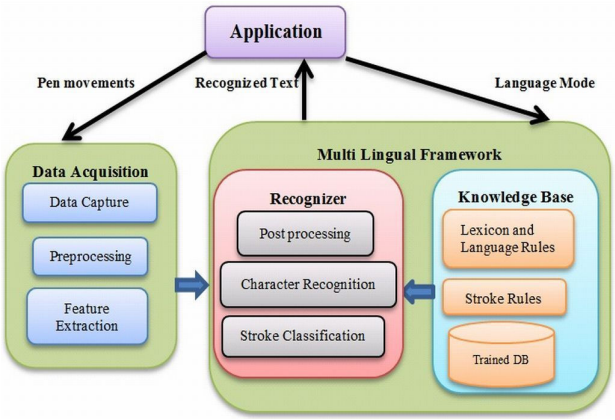


Figure 1. System Architecture

The application has a user friendly interface where user can write the text in the specific language and see the recognized output. The framework, includes a recognizer and a knowledge base, allows the user to select the desired target language and ensures that only the selected language's database is loaded for further processing. Knowledge base helps recognizer in recognizing the characters which is then displayed on the application GUI. The stroke classifier identifies the strokes with the help of the trained database for the selected language. Most script contains multi-stroke characters; strokes which are formed by two or more strokes. Stroke concatenation rules in knowledge base groups the strokes to form a valid character and afterwards mapped to character code. Language dependent lexicon and rules are used in post processing to obtain the N-best list of the words for suggestions.

The major highlight of this architecture is that it splits the total functionality of the OLCR into two – language dependent and language independent modules.

978-1-4673-6994-7/15 $31.00 © 2015 IEEE
DOI 10.1109/ICACC.2015.11

III. OLCR System Description

A. Data Capture: Extracts digital ink from the handheld device as and when the characters are being written over the writing surface by a stylus/finger. Data collected from the tablet is in the form of x and y coordinates of the pen/stylus positions at various times along with PEN DOWN/ PEN_MOVE /PEN UP information.

B. Pre-processing: Removes variability in the strokes due to writing styles as well as due to noise in the collected data. Along with pre-processing operations, position of current stroke w.r.t base stroke is also identified and recorded as Right, Left, Top, Bottom, Middle and Exception. Overlapping of the stroke either horizontally or vertically with previous stroke is also identified in this stage.

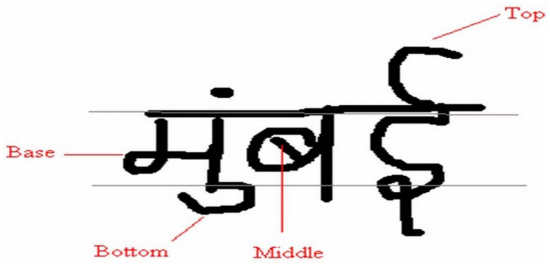


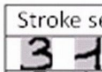
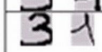
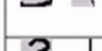
Figure 2. Stroke Positions

Handling stroke order variation is one of the challenging tasks in online word recognition. Delayed stroke, a part of a character which is drawn later i.e. it is written out of normal character sequence, for example, t-crossing, i-dot, and j-dot in English. For Hindi and Punjabi the headline/shirorekha is identified and removed as per [9]. As the user writes, delayed strokes are identified and reordered as per [10].

C. Feature Extraction: The success of any recognition system is often attributed to a good feature extraction method. Normalized x, y coordinate, direction, curvature, vicinity aspect, curliness, slope, loop, cusp and stroke length are extracted from the collected strokes as mentioned in NPen++ recognizer [11].

D. Classification & Recognition: SFAM is used for training the network as well as for classification [12][13]. Most of the Indic script cotains multi- stroke characters. Character recognition is done using probability based stroke rule. All probable sequence in forming a character is called a stroke concatenation rule or simply stroke rule of that character. Handwritten ink data collected during data collection phase is annotated using our data analysis tool and stored. From the analyzed data it retrieves the stroke rules for every character in the particular language. Few samples of stroke sequences obtained from different writers for the Hindi character ञ and Malayalam character ക and the respective stroke rules are shown in Fig 3.

Stroke sequence for Hindi character (ञ HIC1)

Stroke sequence	Stroke Rule
	HIS1R:HIS2R
	HIS1R:HIS2E
	HIS1R:HIS10R:HIS12R
	HIS111R

Stroke sequence for Malayalam character (ക MAC96)

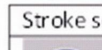
Stroke sequence	Stroke Rule
	MAS92R
	MAS29R:MAS146B

Figure 3. Few samples of stroke sequences obtained for the Hindi character ञ and Malayalam character ക and the respective stroke rules

Transition probability between strokes is also calculated which is a conditional probability statistic that measures the prophecy of neighboring strokes and is defined by

$$P(S_n / S_{n-1}) = \frac{F(S_{n-1}, S_n)}{F(S_{n-1})} \qquad (1)$$

Where, $F(S_{n-1},S_n)$ represents the frequency of stroke S_{n-1} followed by stroke S_n. $F(S_{n-1})$ represents the frequency of stroke S_{n-1} in the stroke rule. Using stroke rule probability and output from classifier, Viterbi algorithm finds the most probable output for the given sequence of observed states. The Viterbi algorithm is a dynamical programming algorithm that allows us to compute the most probable path. It works by finding a maximum over all possible state sequences. Start node is created with a value from '301'. Initial probabilities $\mathbf{\Pi_i}$ and transition probabilities $\mathbf{a_{i,j}}$ of transitioning from state **i** to state **j** are calculated using stroke rule. Say we observe outputs $\mathbf{y_1,\ldots, y_T}$. The most likely state sequence $\mathbf{x_1, \ldots, x_T}$ that produces the observations is given by the recurrence relations.

$$V_{1,k} = P_{(y_1|k)} . \pi_k$$

$$V_{t,k} = max_{x \in S}(P_{(y_t|k)} . a_{x,k} . V_{t-1,x}) \qquad (2)$$

Here $V_{t,k}$ is probability of the most probable state sequence responsible for the first *t* observations that has *k* as its final state [14]. The algorithm keeps a backward pointer (Φ) for each state (t > 1), and stores a probability (δ) with each state. δ is the probability of having reached the state following the path indicated by back pointers. When the algorithm reaches the states at time, t = T, δs for the final states are the probabilities of following the optimal (most probable) route to that state. Thus selecting the largest, and

using the implied route, provides the best answer to the problem. Let Ptr(k,t) be the function that returns the value of x used to compute $V_{t,k}$ if t>1 or k if t = 1. Then:

$$x_T = argmax_{x \in S}(V_{T,x})$$

$$x_{t-1} = Ptr(x_t t) \quad (3)$$

An important point about Viterbi algorithm is that it does not simple-mindedly accept the most likely state for a given time instant, but takes a decision based on the whole sequence - thus, if there is a particularly `unlikely' event midway through the sequence, this will not matter provided the whole context of what is seen is reasonable [15].

E. Post-Processing: Involves steps like output code representation, application of language rules, disambiguation of confusion pairs etc. to refine the recognition results. Linguistic rules are used for validating the characters sequence, character position, character combinations etc. SFAM classifier based confidence measure is used for generating alternatives/suggestions for multi-stroke characters. A dictionary based scheme is utilized for generating word level suggestions.

IV. Android Architecture

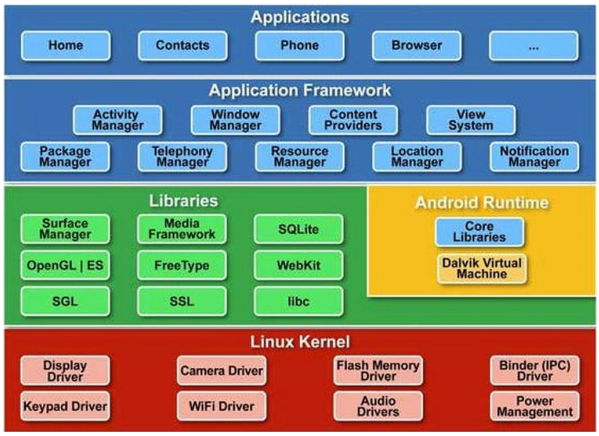


Figure 4. Android Architecture

A good Android development knowledge foundation requires an understanding of the overall architecture of Android. From Fig 4, it can be seen that applications are the top layer in the Android architecture and this is where our applications fit into. Applications are mostly written in Java and run within individual instances of Dalvik Virtual Machine. The key goals of Android architecture are performance and efficiency, both in application execution and in implementation of reuse in application design [16].

V. Results

Currently we have integrated and tested four languages Malayalam, Hindi, Tamil and Urdu to this framework. Integration of three other languages, namely, Punjabi, Telugu and Kannada is in progress. Details of characters used in each language are given below.

- Malayalam - 15 vowels, 36 consonants, 16 diacritics, 71 conjunct consonants and 5 chillus.
- Tamil - 12 vowels, 19 consonants, 4 conjuncts and 9 vowel modifiers; total of 44 characters. 39 single stroke CV characters are also added like தூ,வி,பூ etc.
- Hindi - 11 vowels, 22 consonants, 17 diacritics, 4 conjuncts and 31 Half consonants.
- Urdu - 36 basic characters.

Out of 101 unique strokes identified, 1321 stroke samples are trained for Malayalam. For incorporating Tamil we had trained 1059 samples from the 79 unique strokes. For Hindi, minimum of 12 samples each from the 73 unique strokes were arbitrarily chosen for training. 974 samples were given to SFAM training module after extracting the features. In Urdu 36 basic characters are represented in 27 distinctive strokes and trained with 324 samples. Compared to other three languages multi-stroke character ratio is more in Hindi. Number of characters and strokes used in four languages is given in Table 1. Stroke classification and character recognition accuracy are listed in Table 2 and Table 3 respectively.

TABLE I. No of characters and strokes used in four Indian languages

Language	Characters	Unique Strokes
Hindi	85	73
Malayalam	143	101
Tamil	127	79
Urdu	36	27

TABLE II. Stroke classification accuracy

Language	No. of strokes trained	No. of strokes tested	Accuracy (%)
Hindi	974	705	93.21
Malayalam	1321	1200	96.4
Tamil	1059	954	95.64
Urdu	324	246	94.11

TABLE III. Character Recognition accuracy

Language	No. of character x No of writers	Accuracy (%)
Hindi	85 x 8	88.28
Malayalam	143 x 14	94.63
Tamil	127x7	92.81
Urdu	36x10	89.79

VI. Conclusion

This paper describes a Multilingual Online handwriting recognition system in Android based smart phones that currently supports seven Indian languages and digits. In our

approach, individual strokes are identified by comparing the features of the unknown stroke with that of the stroke feature database. We have achieved stroke classification accuracies ranging from 93 – 96% and character recognition 88 – 94%.

Our testing was done with a small set of strokes samples. It was observed that stroke classification accuracy can be increased by training more samples. Character accuracy achieved was decreased due to occurrence of same strokes at different position. For example in Hindi, stroke \ can be part the characters के,ट् or ष in different positions such as top, bottom and middle respectively. Also in Urdu the characters vary based on position and number of dots such as ڈ،ٹ،ٹ،ٹ. These are handled by including positional information of the stroke with respect to base character in the stroke rule which helps to remove some of the possible state sequences which may be used by the Viterbi algorithm by setting value to 0 in transition probability. This increased character recognition accuracy is by 2- 3%. Recognition outputs of are shown in Fig 5 and Fig. 6.

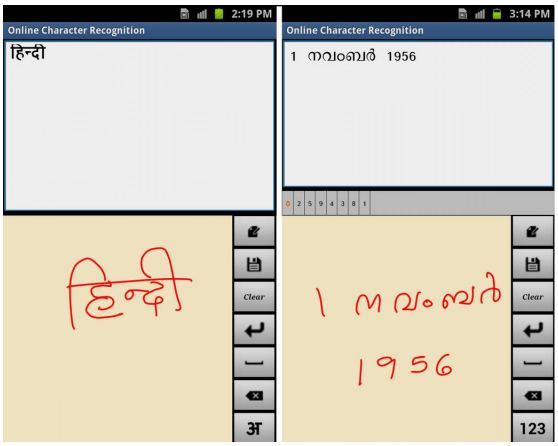


Figure 5. Recognition Outputs for Malayalam & Hindi

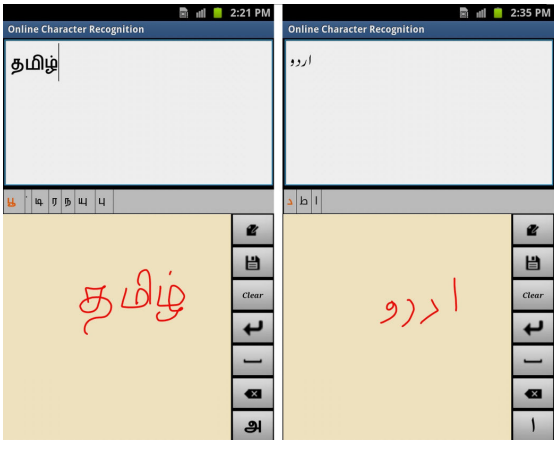


Figure 6. Recognition Outputs for Tamil & Urdu

VII. Future Work

The multilingual framework we had developed can be extended to other Indian languages also by adding appropriate Feature DB and language specific modules. Our future works involve accuracy & speed improvement, extension of this framework to other languages and personalize the application according to user's own handwriting which will greatly reduce confusion pairs which will in turn increase the accuracy.

Acknowledgment

The authors would like to thank and acknowledge the support we had received in completing the tasks on time.

References

[1] Indhu T.R, Bhadran V.K., "Malayalam online handwriting recognition system: A simplified fuzzy ARTMAP approach", INDICON, 2012 Annual IEEE, 7-9 Dec. 2012, pp: 613 - 618

[2] G. Shankar, V. Anoop, and V. Chakravarthy. LEKHAK [MAL]: A system for online recognition of handwritten Malayalam characters. In NCC, IIT, Madras, Jan. 2003

[3] V. Babu, L. Prasanth, R. Sharma, G. Rao, and A. Bharath. HMM-based online handwriting recognition system for Telugu symbols. Document Analysis and Recognition, International Conference on, 1:63–67, 2007

[4] H.Swethalakshmi, A.Jayaraman, V.Chakravarthy, and C.Sekhar. Online handwritten character recognition of Devnagari and Telugu characters using support vector machines. In International Workshop on Frontiers in Handwriting Recognition,La Baule,France, Oct. 2006.

[5] K.Aparna,V.Subramanian, M.Kasirajan, G. Prakash, V. Chakravarthy and S. Madhvanath. Online handwriting recognition for Tamil. pages 438–443, Oct. 2004

[6] M. M. Prasad, M. Sukumar, and A. G. Ramakrishnan. Divide and conquer technique in online handwritten Kannada character recognition. In Proc. of the International Workshop on Multilingual OCR, pages 1–7, Barcelona, Spain, 2009

[7] M. Sreeraj, Sumam Mary Idicula. k-NN Based On-Line Handwritten Character Recognition System, ICIIC, pp.171-176, 2010 First International Conference on Integrated Intelligent Computing, 2010

[8] A. Arora, A.M. Namboodiri, A Hybrid Model for Recognition of Online Handwriting in Indian Scripts, 12th International Conference on Frontiers in Handwriting Recognition, pp:433-438

[9] A. Bharath and Sriganesh Madhvanath, "HMM-Based Lexicon-Driven and Lexicon-Free Word recognition for Online Handwritten Indic Scripts" IEEE Transactions On Pattern Analysis And Machine Intelligence, Vol. 34, No. 4, April 2012.

[10] Vidya, V., Indhu, T.R., Bhadran, V.K.(2014) Reordering of delayed strokes for online Hindi word recognition International Conference on Contemporary Computing and Informatics (IC3I)

[11] S. Jaeger et al. (2001) Online handwriting recognition: the NPen++ recognizer. International Journal on Document Analysis and Recognition(IJDAR), vol. 3, no. 3, pp. 169-180, 2001

[12] G.A.Carpenter,S.Grossberg and J.H.Reynolds,"ARTMAP: supervised real time learning and classification of non stationary data by a self organizing neural network," Neural Networks 4 (1991) 565–588

[13] T.Kasuba,"Simplified fuzzy ARTMAP",AI Expert,Nov(1993) 18–25.

[14] http://en.wikipedia.org/wiki/Viterbi_algorithm

[15] http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/viterbi_algorithm/s4_pg1.html

[16] http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture