A Hybrid Model for Recognition of Online Handwriting in Indian Scripts

Amit Arora and Anoop M. Namboodiri
International Institute of Information Technology, Hyderabad, India
{*amit.arora@research.,anoop@*}*iiit.ac.in*

Abstract

We present a complete online handwritten character recognition system for Indian languages that handles the ambiguities in segmentation as well as recognition of the strokes. The recognition is based on a generative model of handwriting formation, coupled with a discriminative model for classification of strokes. Such an approach can seamlessly integrate language and script information in the generative model and deal with similar strokes using the discriminative stroke classification model. The recognition is performed in a purely bottom-up fashion, starting with the strokes, and the ambiguities at each stage are preserved and transferred to the next stage for obtaining the most probable results at each stage. We also present the results of various preprocessing, feature selection and classification studies on a large data set collected from native language writers in two different Indian languages: Malayalam and Telugu. The system achieves a stroke level accuracy of 95.78% *and* 95.12% *on Malayalam and Telugu data, respectively. The akshara level accuracy of the system is around* 78% *on a corpus of* $60,492$ *words from* 367 *writers.*

1. Introduction

Online handwriting recognizers are important for the use of Indian languages, especially for mobile devices, where the use of a keyboard have not become very popular due to the number of alphabets present in these languages. Such systems have reliable performance levels in the case of English and some east asian languages like Chinese and Korean [10]. However, little progress has been made in development of such systems for Indian Languages. The primary deterrents in performance for such languages are the large number of character classes, as well as the similarity between these characters. In this work, we present a generic design for development of online handwriting recognizers for Indian languages, and demonstrate its effectiveness with two Indian languages, Malayalam and Telugu. The recognition engines that are developed offer a high degree of accuracy with relatively small memory footprint, considering the complexity of character-set of these languages.

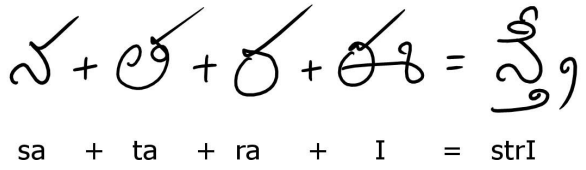

Figure 1. Multiple characters forming an akshara in Telugu.

Our approach models the production of handwriting in these languages as a multi-layer Markov model. In all the Indian languages, the basic unit of writing is an akshara, which is a combination of multiple consonants(C), followed by a vowel(V): C^nV. The concept of an akshara is closely related, but not identical to a syllable in English. Each word is formed as a sequence of aksharas. Each akshara, in turn is composed of a set of basic characters (C_i), and each basic character is written using a sequence of strokes (S_i). For example, the word $strI$ is a single akshara and is composed of the basic consonants: sa, ta, ra, and the vowel I (see Figure 1). However, as we see from the above figure, the composition of an akshara from its component characters is not a direct concatenation of the strokes of the corresponding aksharas. Moreover, multiple strokes could be combined into one within an akshara, depending on the context of writing. This structure is common among most Indian languages and scripts. What varies is the specific strokes and their combinations within an akshara. We utilize this fact to propose a generative model for the languages. The model decouples the interaction between characters within an akshara, and lends to a sim-

plified estimation of the most likely sequence of characters forming a word.

Existing work in the field on online handwriting recognition concentrates mostly on recognition of independent aksharas. Shankar *et al.* [13] has presented an independent Malayalam character recognition system using a string matching technique for stroke recognition, and reported an accuracy of around 90% on a dataset of 216 strokes collected from 3 writers. Techniques such as HMM [5], Elastic Matching [12], and Support Vector Machines [14] have been tried for recognition of Telugu strokes with accuracies ranging from 83% to 92%. Most of these works concentrate on isolated character recognition. Similar works have been reported for other Indian languages such as Bangla [9], Tamil [2], Kannada [11], etc.

The model-based stroke recognition has been applied in case of Chinese character recognition [7], where a two stage process for candidate stroke extraction and consistent matching is used. For Online handwriting recognition, stroke-level HMMs [3] and segmental HMMs [4] have been used by Artieres *et al.* These model the strokes in the form of a dictionary of base shapes that can be combined to form a large variety of characters.

However, the challenges in recognition lies primarily in three factors: i) presence of extremely large number of aksharas due to the combinations that are possible using the basic characters, ii) ambiguities in the boundaries of aksharas as they are not defined by any marker or space (they can be detected only after recognition, leading to a chicken-and-egg problem), and iii) a high degree of similarity between many strokes, making the recognition of independent strokes, extremely difficult. Neeba *et al.* [8] discusses the challenges in recognition of Malayalam script, for both printed and online handwritten characters. As seen above, most of the existing works in Indian language handwriting recognition, assume that the akshara segmentation is given. They concentrate on the stroke recognition part, and use rules for combining the results of stroke recognition to aksharas. In this work, we try to develop a comprehensive solution for recognition of a complete word using a hybrid generative-discriminative model, and hence is more applicable in real-life applications.

In the following section, we describe the design philosophy of our system, followed by the specific steps in the recognition process in Section 3. Section 4 presents experimental results on two different languages: Malayalam and Telugu.

2. A Hybrid Model for Recognition

We note that the formation of online handwriting in most Indian languages can be modeled as a hierarchical process. A word is formed using a a sequence of aksharas; each akshara is formed from a set of basic characters, which in turn are formed by a sequence of strokes (see Figure 2).

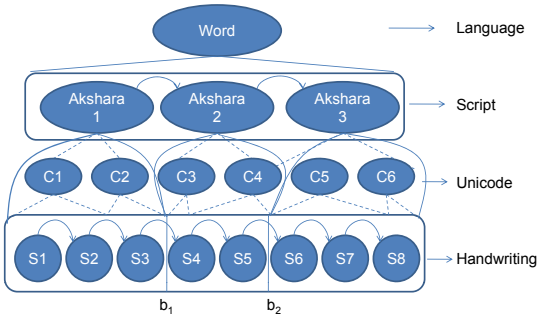

Figure 2. Handwriting formation from a word

Previously, a heuristic search-based stroke-correspondence was employed by Liu *et al.* [7] to model strokes to aksharas, while segmental HMMs by Artieres *et al.* [4] required allographs for each character sample. In the proposed approach, the formation of aksharas in a word can be modeled as a Markov process, which is controlled by the underlying language model. The akshara to basic character mapping is fixed and well known. However, as seen from Figure 1, the specific strokes that constitute a basic character can vary considerably based on the presence of other characters in the akshara, as well as its position in the akshara.

To get over this problem, we avoid the explicit recognition of basic characters by removing that layer from the generative model. In other words, even though the language defines an akshara as a set of basic characters, we can think of the script as a mapping from aksharas, directly to the strokes in the handwriting. In our model, we consider the strokes as a sequence emanating from the underlying sequence of aksharas in the word. This structure is common among most Indian languages. Once the aksharas are recognized, one can decompose it into the underlying basic characters to generate a Unicode representation, which is defined at that level. This model leads us to a formulation where the stroke sequence is generated by a Markov process,

controlled by the underlying aksharas. The approach is similar in principle to the document image decoding process describe by Kopec and Chou [6].

The challenges of dealing with large number of classes, as well as the similarity of different classes have been alleviated to a large extent due to the recent advances in learning and classification techniques that can handle large class problems both efficiently and reliably. We utilize these developments in kernel-based discriminative classification methods to improve the recognition performance of the primitive (stroke) recognizer. However, discriminative classifiers such as Support Vector Machines (SVM), do not provide the posterior probabilities of primitive recognition, which is required to couple the results to the generative word formation model. Moreover, the problem of similar classes is extreme in some of the Indian languages such as Malayalam and Telugu, that in practice samples of two classes can be visually identical. The only way to tell them apart is based on the context. We get over this problem by empirically determining the performance of the trained classifier, which provides the probabilities from the resulting confusion matrix, as described later.

3. Design of Recognition Engine

The design of the recognition engines is based on the following observations:

- Writing of most Indian scripts are non-cursive as the pen is always lifted while moving from one akshara to the next. Hence strokes are used as the basic primitives for recognition.
- Modeling the recognition as a bottom up process enables us to simplify the implementation. This requires one to keep track of the uncertainties in recognition, which can be resolved only by a top-down language model. We maintain a list of top-N choices for recognition with the corresponding probabilities, which encompasses all the information that is needed to incorporate the contextual constraints into the recognition result.

The recognition process proceeds in four stages: i) A feature representation of the strokes are derived after pre-processing and normalization of the strokes, ii) Individual strokes are classified and the top-N probable classes are computed for each stroke, along with their probabilities, iii) A Viterbi-decoding process is carried out using the stroke class labels as observations, to compute the most likely sequence of labels for the strokes, and iv) the resulting aksharas that generated the strokes are used to generate the component characters, and is converted a Unicode representation for output.

3.1 Pre-Processing:

The primary purpose of the pre-processing step is to remove the variability in the strokes due to the writing styles as well as due to the noise in the collected data. Each stroke in online handwriting is represented as a sequence of points through which the pen moved from a pen-down to the next pen-up. The number of points in the stroke and their distances would also vary based on the speed of writing.

A normalization step first scales the strokes to fit within a predefined box. The aspect ratio of the stroke is maintained in the process. We performed experiments with different variants of resampling these points to remove variations in writing speed. The first one is based on applying a Gaussian low-pass filter for removing the noise, and an equi-distant resampling to remove variations in writing speed. The second approach fits an approximating spline curve to the observed data. Such an approach should ideally help in maintaining the high curvature regions of a stroke, while removing high frequency noise. One could also choose between equi-distant and equi-time resamplings, that would decide whether the temporal information of the writing process is retained or not.

3.2 Representation of Strokes

We have experimented with a large set of features for their effectiveness in representation of significant characteristics of the strokes. The features used were: 1) raw x and y co-ordinates of the resampled points, 2) moments of the stroke up to fourth order, 3) overall direction and curvature of the stroke, 4) length of the stroke, 5) aspect ratio, 6) area of the stroke, 7) number and direction of points in different sub-windows (5×5), 8) projection histograms in X and Y directions, and 9) Fourier coefficients of x and y sequences (first 5 coefficients of both sequences were used).

We performed extensive feature selection studies, and selected the first six in the above list for the final representation.

3.3 Classifier Design

The primary concerns in stroke recognizer are that of efficiency and accuracy. The stroke is a represented as a sequence of points, which are the positions of the pen tip sampled at regular intervals in time. Hence, the number of points in two instances of the same stroke can vary considerably. A recognizer for such a data can either convert it into a fixed length representation, or

use models that can handle variable length representations such as Hidden Markov Models (HMMs) or Dynamic Time Warping (DTW) based comparison. The first class is considerably more efficient than the second, while both have specific cases, where they are superior in terms of recognition accuracy.

The efficiency is extremely important for online handwriting due to the expectations of real-time performance, and the requirement to handle large number of classes. The number of stroke classes in online Malayalam handwriting is around 100, and that of Telugu is around 220. We employed various classifiers as well as parameters for this purpose, and found that an SVM classifier using a Decision Directed Acyclic Graph (DDAG) formation for combining individual pair-wise classifiers to be the most effective in classifying the strokes. We use a two-stage process, in which the top-N results of this preliminary recognizer are considered as potential candidates and a discriminating classifier is employed that refines the potential class labels. Note that the accuracy of the stroke recognizer is expected to be low even after the two-stage approach. Hence it is important to compute the top-k results with confidence measures that can be used by the word recognizer to find the most likely word.

Discriminating Classifier: Many stroke classes in Malayalam and Telugu have very similar shapes, while differing in a specific part of the stroke. A discriminative classifier would give varying importance to different parts of the stroke, while looking at specific pairs of stroke classes [1]. Another reason to employ a discriminating classifier is that it can select specific features to disambiguate strokes that are of identical in shape, but differing in size, position, etc. Note that even after the discriminating classifier, certain ambiguities will remain, which can be resolved only by using the contextual information at the post-processor stage.

3.4 Design of Post-processor

The stroke recognizer returns the top-k results with confidence values for each of the input stroke. One could attempt to segment a word into individual aksharas, and then recognize the aksharas independently. However, such a segmentation is not possible by inter-stroke spacing alone in the case of Malayalam and Telugu, as the distance between strokes in an akshara is often equal to the inter-akshara spacing. For example, the word 'malayALam' written in figure 3 contains 4 aksharas, while the strokes are almost equally spaced. The word recognizer should hence come up with the most likely interpretation of the stroke sequence as a sequence of aksharas. This is extremely difficult, as the number of possible interpretations for a sequence of N strokes, with top-k recognition results is k^N. Each of these interpretations need to be evaluated for consistency and likelihood to arrive at the most likely interpretation of the word (see Figure 3).

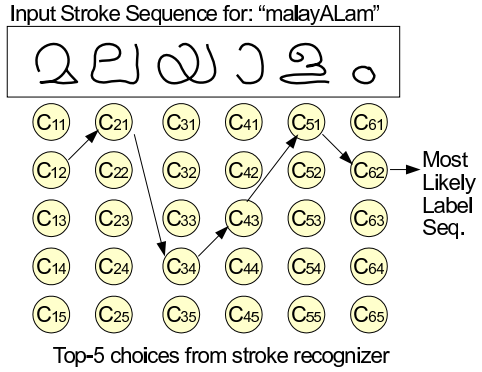

Figure 3. Estimation of akshara sequence from stroke recognition results.

The problem is same as that of estimating the most likely state sequence in the underlying Markov model, and we can employ a Viterbi decoding algorithm to achieve this. One can also incorporate low-level language models, while computing the most likely sequence of labels. However, to employ the Viterbi algorithm, one needs to compute the observation probabilities of the stroke sequences for each akshara in the language. This is extremely high due to the possible permutations of basic characters.

We overcome this problem by dividing the strokes into six stroke classes, and defining the akshara boundaries based on the stroke classes. The stroke labels obtained from the recognition stage is used to estimate the stroke classes, which is used to determine the akshara boundaries (see b_i in Figure 2). The rules of akshara formulation are encapsulated in a Finite State Automaton (FSA), which is used by recursive algorithm, coupled with the akshara recognition process to estimate the most likely sequence of aksharas. This process will yield the ideal akshara sequence that one would compute using a Viterbi decoding algorithm, provided the class labels of the strokes are accurate. The akshara recognizer uses the top-N results to compute the most likely stroke labels.

Once the aksharas are segmented and the stroke labels determined, the Unicode of each akshara is generated from the Unicodes of its constituent parts (consonants, modifiers, etc.).

The stroke classes based approach is applicable to most Indian languages, and we demonstrate the results

on two different languages with only minor variations in the rules (FSA).

4. Experimental Results and Analysis

We have collected a signifact amount of natual handwritten data for both Malayalam and Telugu. Both languages are syllabic in nature. Officially, Telugu consists of 18 vowels, 36 consonants, and 3 dual symbols, while Malayalam has 18 vowels, 36 consonants, and 5 half-consonants. In addition to these, there are several symbols for CV and CCV combinations. We have included the most popular ones in our dataset.

Our word list consists of 135 words of Telugu. In case of Malayalam, the number of words selected was 120. While collecting these words, we assume of covering all the symbols and their variations. Data is being collected on A4 size papers with 30 words per page, with sufficient spacing to make the segmentation of the training data, easy. In addition to the above set of words, some of the data was collected as individual characters for convenience of analysis and development of the recognition engine. The data was collected using Genius G-Note 7000 digital ink pad.

A total of 7348 samples from 90 Malayalam strokes were used in order to train the system. The Telugu dataset for training comprised of 57,669 samples from 208 classes. The results discussed below are for Malayalam data, and similar accuracies were obtained, when the final methods were applied to Telugu character set.

4.1 Study of Pre-processing

Various techniques were tested for pre-processing of strokes so that the x,y co-ordinates can be represented in a equal length vector. A comparison of the techniques is shown in Table 1. We note that equi-distant resampling method with no smoothing gives the highest accuracy. It is interesting to note that the smoothing process reduces the accuracy as it probably smoothes over minor variations that are significant in differentiating similar strokes. We use 20 samples for further experiments.

4.2 Feature Selection

In addition to the 20 raw points selected above, we experiment with a host of popular features as described in the design section. The accuracy of the individual features, each combined with the raw points, is given in the Table 2

We also performed a forward feature selection process, which resulted in the selection of the following set of features: first 10 Fourier coefficients of x, y (magnitude only), $2nd$ and $3rd$ order moments of the x and y coordinates of the point sets, overall length, direction, and curvature of the stroke, total area and aspect ratio of the stroke, and normalized x, y coordinates of sample points after resampling the stroke to have 20 points along the stroke.

Number of Samples	Critical Point Method	Equi-Dist Resampling with Smoothing	Equi-Dist Resampling w/o Smoothing
15	80.24%	83.57%	92.09%
20	82.76%	86.06%	92.52%
25	86.31%	88.51%	92.71%
30	88.16%	90.20%	92.71%

Table 1. Accuracies with various resampling methods using varying sample size.

Features	Accuracy
Total area and aspect ratio	95.14%
Overall length, direction, and curvature	94.07%
First four moments	93.96%
2nd and 3rd Order moments	93.82%
First 10 Fourier coefficients of x, y	94.20%
Grid Occupancy	94.26%

Table 2. Accuracies of various features along with raw points.

This results in a feature vector of length 69 which gives an accuracy of 95.33% with an SVM classifier using 4-fold cross validation. The Telugu character set showed the accuracy of 95.11% on a similar test.

4.3 Classification Methods

The selected features from the previous section, were tested with various classifiers in order to get the accuracy estimates, the results of whom are shown in Table 3.

In our experiments, SVM classifier with polynomial kernel performed the best. The results were similar with Telugu dataset also. Linear kernel also performs exceedingly well, and could be considered if one wants to reduce the computational complexity of the classifier. As noted before, we use the confusion matrix from the training phase to estimate the top-N candidates of each recognition result.

The overall accuracy of Malayalam recognizer, when tested on a dataset of 60, 492 words collected from 367 writers is found to be 78.07%. The data contained

Classifier	Accuracy
k-Nearest Neighbor	92.65%
Multi-layer Perceptron	91.09%
SVM (Poly. Kernel)	95.33%
SVM (Linear Kernel)	95.08%
SVM (RBF Kernel)	89.27%
Gaussian Likelihood	82.92%
Naive Bayes Classifier	88.77%
Binary Tree	78.44%
Hidden Markov Model	74.29%
Gaussian Mixture Model	88.70%

Table 3. Accuracies of various classifiers with Malayalam data.

large variations in writing style and other errors such as overwriting or missing strokes. The isolated character recognition is a simpler problem, and we achieve an accuracy of 93.10% on a dataset of over $11,000$ characters collected from 125 writers. The corresponding accuracies on a similar-sized Telugu dataset was 75.70% on the word level. The average time taken for detection of a stroke, akshara and word for Malayalam recognition are $2.85ms$, $10.67ms$ and $44.80ms$, respectively on a desktop-class machine.

5. Conclusions and Future Work

We have presented a comprehensive framework for recognition of online handwritten words in Indian languages, and implemented it for two different languages: Malayalam and Telugu. The approach proposed helps in improvement of recognition accuracies of the underlying strokes, while supporting the inclusion of language models for recognition and generation of Unicode using a generative Markov model. We have also proposed a scheme for coupling the two models and developed an efficient implementation for the recognition process. Experimental results on a substantial dataset proves the effectiveness of the approach.

The performance of the existing classifier could be further enhanced by improving the script model used for recognition of aksharas, as well as by incorporating language models to provide contextual information at the akshara level.

6. Acknowledgement

The authors would like to thank Technology Development for Indian Languages(TDIL), DIT, Government of India for funding the research work.

References

[1] K. Alahari, S. P. Lahari, and C. V. Jawahar. Discriminant substrokes for online handwriting recognition. volume 1, pages 499–503, 2005.

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

[3] T. Artières and P. Gallinari. Stroke level hmms for online handwriting recognition. In *IWFHR '02: Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, page 227, Washington, DC, USA, 2002. IEEE Computer Society.

[4] T. Artieres, S. Marukatat, and P. Gallinari. Online handwritten shape recognition using segmental hidden markov models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):205–217, 2007.

[5] 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.

[6] G. E. Kopec and P. Chou. Document image decoding using markov source models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):602–617, June 1994.

[7] C.-L. Liu, I.-J. Kim, and J. H. Kim. Model-based stroke extraction and matching for handwritten chinese character recognition. *Pattern Recognition*, 34(12):2339 – 2352, 2001.

[8] N. Neeba, A. Namboodiri, C. V. Jawahar, and P. Narayanan. *Guide to OCR for Indic Scripts*, chapter Recognition of Malayalam Documents, pages 125–146. Springer, 2009. 978-1-84800-329-3.

[9] S. Parui, K. Guin, U. Bhattacharya, and B. Chaudhuri. Online handwritten bangla character recognition using hmm. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1 –4, 8-11 2008.

[10] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. on PAMI*, 22(1):63–84, 2000.

[11] 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.

[12] L. Prasanth, V. Babu, R. Sharma, G. V. Rao, and D. M. Elastic matching of online handwritten tamil and telugu scripts using local features. In *Proc. of International Conference on Document Analysis and Recognition*, pages 1028–1032, Washington, DC, USA, 2007.

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

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