Online Handwriting Recognition for Tamil

K.H. Aparna, Vidhya Subramanian, M. Kasirajan, G. Vijay Prakash, V.S. Chakravarthy,
Indian Institute of Technology, Chennai, India, 600036.

Sriganesh Madhvanath, *Hewlett-Packard India, Bangalore, India.*

Abstract

A system for online recognition of handwritten Tamil characters is presented. A handwritten character is constructed by executing a sequence of strokes. A structure- or shape-based representation of a stroke is used in which a stroke is represented as a string of shape features. Using this string representation, an unknown stroke is identified by comparing it with a database of strokes using a flexible string matching procedure. A full character is recognized by identifying all the component strokes. Character termination, is determined using a finite state automaton. Development of similar systems for other Indian scripts is outlined.

Keywords: Shape features, stroke identification, FSA

1. Introduction

Online handwritten character recognition consists of recognizing a script as it is written using an electronic stylus or a pen on a tablet [1]. Online handwriting recognition takes on a novel significance in the context of Indian languages. Presently, word-processing in Indian languages using the alphanumeric keyboard can be a vexing experience. Elaborate keyboard mapping systems exist but are cumbersome. Online character recognition offers a natural solution to this problem. The present work describes a system for online recognition of Tamil script, a language widely spoken in South India.

Template-based online character recognition [2] may be broadly grouped into four categories: motor models [3], structure-based methods [4], stochastic HMM-type methods [5], and learning-based models like neural networks [6]. The approach presented in the present paper comes nearest to a structure or shape based approach. The present method is based on earlier theoretical work which shows that the global shape of any handwritten character may be reduced to a small set of local shapes; the global shape is then a graph of the local shapes [7].

1.1 Composite characters and Indian Scripts

Indian scripts are generally written in non-cursive style, unlike Latin alphabet, which is normally written in cursive style, rendering recognition difficult. However, Indian scripts pose a peculiar problem non-existent in European scripts – the problem of *composite characters*. Unlike in Latin alphabet where a single character represents a consonant or a vowel, in Indian scripts a composite character represents either a complete syllable, or the coda of one syllable and the onset of another. Therefore, although the basic units that form composite characters of a script are not that many ($O(10^2)$), these units by various combinations lead to a large number ($O(10^4)$) of composite characters.

1.2 The Tamil Character Set

However, the case of Tamil is relatively simpler compared to other Indian scripts: the rules for character composition are far fewer than in other Indian scripts. The only category of composition allowed is of Consonant-Vowel type, where a structure corresponding to a consonant and another corresponding to vowel are combined to form a C-V type character with a unique shape. Even this composition does not occur for all C-V combinations. In many cases the vowel modifier appears as a horizontally isolatable structure. The full character is then identified by sequence information (see Fig. 1).

The tamil character set has 12 vowels, 23 consonants. The main modules in our Tamil online recognition system are shown in Fig. 2.

The paper is organized as follows: Section 2 explains the hierarchy of data structures used in representing handwritten data. Stroke preprocessing steps are described in Section 3, and Section 4 describes the extraction of shape features from the stroke followed by recognition. Section 5 explains the finite state automaton (FSA), which is used for determining character termination and character identification. The paper concludes with results and discussions in Section 6.

Proceedings of the 9th Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)
0-7695-2187-8/04 $20.00 © 2004 IEEE

Vowel= அ ஆ இ ஈ உ ஊ எ ஏ ஐ ஒ ஓ ஔ

X(consonant)= க ங ச ஞ ட ண த ந ப ம ய ர ல வ ழ ள ற ன ஜ ஸ ஷ ஹ

Vowel modifiers in different horizontal blocks =
xா ெx ேx ைx ொx ோx ௌx

Y(Vowel modifiers in the same horizontal block)=
x ி x ீ x ு x ூ

Numbers 0 1 2 3 4 5 6 7 8 9

2. Data representation

The following concepts and data entities have been designed to provide a unified framework for all Indian scripts. Hence all the properties may not relevant to every script. Some definitions are in line.

Definitions:

Stroke: The trajectory of the pen between a pen-down event and a pen-up event. All the unique strokes of a script are manually identified and given unique labels. It is the smallest physically identifiable unit in online handwriting.

Proximity: Criteria for spatial proximity are necessary for grouping strokes into larger structures. There are 3 rules for representing "proximity" of strokes.

1) contact or nearness: two strokes are proximal if they intersect, make contact or if the pair of nearest points on the respective strokes are closer than a threshold value.
2) Enclosure: A stroke encloses another stroke (this situation does not arise in Tamil but is useful for other Indian scripts)
3) X-overlap: Two strokes overlap in horizontal direction.

Note that the 3rd proximity rule is a less restricted one than the others and subsumes the other two.

Stroke Group: Any random collection of strokes. Usually it refers to a set of strokes which form a unit with lesser significance than a character. A stroke group also can have a label.

In practice a Stroke Group object may be used to group strokes in the following 2 ways:

- to group syntactically meaningful subsets of the full set of strokes forming a composite character. (Such grouping turns out to be more useful in formulating compositional rules for other Indian scripts like Hindi, Telugu etc.)
- To group physically connected ("proximal") subset of strokes forming a composite character.

Thus a Stroke Group is meant to be used in a flexible way depending on the context as illustrated above.

Horizontal Block: This is a set of strokes grouped with the X-overlap proximity criterion described above. This definition is necessary because very often in Indian scripts all the strokes that comprise a composite character form a horizontal block.

Character: A character is the smallest segment of handwriting which can be associated with a syntactic code like, say, the ISCII code. Typically it is one or several contiguous horizontal blocks.
Word: An array of characters.
Line: A horizontal array of words interspersed with spaces.
Page: A vertical array of lines.
Document: An array of pages.

3. Preprocessing

3.1 Pen device:

We use the Superpen™, a product of UC Logic Inc. to generate online character data. Although the device provides pressure information pressure is not used by our algorithm.

3.2 Preprocessing:

Preprocessing consists of interpolation, smoothing and normalization of strokes. For normalizing a stroke we first determine the bounding box of the entire horizontal block which the stroke a part of. We then divide both x- and y-dimensions of the stroke by the 'height' of the horizontal block. This preserves the relative size of strokes in a horizontal block, which is very important. The strokes are then converted onto curve length base (sampled uniformly along curve length) and then smoothed independently along t-axis using a Gaussian filter. During data collection, individual strokes as well as some horizontal blocks of strokes are also taken so that enough samples are available.

4. Feature extraction and Stroke recognition:

4.1 Feature Extraction:

In an earlier work [4], we have identified certain general features – known as the *shape features* – of handwritten characters, which are less susceptible to distortion introduced by writing. The set of 18 shape features used in the present system are summarized in Fig. 3. The 18 shape features are denoted by uppercase alphabets from 'A' to 'R'.

The 18 features used may be further classified as follows:

Dot: A Dot is simply a very small stroke. A stroke is a Dot if both the sides of its bounding box are less than certain lower limits.
Line terminals: These are the ends of a stroke. They are of 4 kinds – A (-45^o to $+45^o$), C (45^o to 135^o), E (135^o to 225^o) and G (225^o to -45^o). The other four types shown in Fig. 3 are not being used now**.**
Bumps: These are points on the stroke where 1) a tangent exists, 2) the stroke is wholly on one side of the tangent, and 3) the slope of the tangent takes a small set of pre-specified values viz., 0, 45, 90 and 135 degrees. Conditions 2 and 3 together give rise to 8 bumps denoted by I, J, K, L, M, N, O, P.
Cusp: A cusp is a point where dx/dt and dy/dt simultaneously go to zero. It has a sharp spiky appearance.

Fig 3 shows the 18 shape features and Fig 4 shows the features extracted from "ka". For the initial and final segments, we take directions pointing outwards and towards the center respectively and the corresponding shape feature is assigned.

4.2 Stroke Identification:

In this step, the shape feature string of an unknown stroke is compared with a database of such strings. By manual analysis we have isolated 96 strokes in Tamil handwritten script. Special issues arise when comparing the unknown string with a database string. Regular string matching techniques give disastrous results since, often, stray features are inserted or expected features absent in the unknown string. Therefore, we adopt the following "soft-matching" approach to string comparison.

Flexible String Comparison:

The flexible string comparison flexibly compares two strings
Initialization:
smallstr = unknown string;
largestr = database string;
matchval = 0;

Procedure:

- Step 0: swap()
- Step 1: Look for *smallstr*(1) within *largestr*(1:WIN). If the search fails, delete *smallstr(1)* from *smallstr* and go back to step 0; If the search succeeds, clip *largestr* up to the point of match; swap(); Set i = 1; go to step 2.
- Step 2: Check if *smallstr*(i) = *largestr*(i). If true go to Step 3, else go to Step 4.
- Step 3:matchval = matchval + 1; i = i + 1; go to Step 2.
- Step 4: Clip both strings up to point of last match. If the length of the smaller string is 0, STOP; else go to Step 1.

(Note: The function swap() swaps *largestr* and *smallstr* if the length of the *largestr* happens to be smaller than the *smallstr*. *Smallstr(i) denotes the i-th character in smallstr. WIN refers to the window size used for comparison. The value of WIN is 5*)

Example:
flexistrcmp3('CALICUT', 'CALCUTTA', 5)
After swapping
strsrctmp = CALICUT
strdestmp = CALCUTTA
Looking for smallstr(1) in largestr(1:WIN)
Search success
smallstr(i) = largestr(i)
seglen = 1
smallstr(i) = largestr(i)
seglen = 2
smallstr(i) = largestr(i)
seglen = 3
smallstr(i) != largestr(i)
nmatch = 3
strsrctmp = ICUT
strdestmp = CUTTA
Looking for smallstr(1) in largestr(1:WIN)
Search Failed
Looking for smallstr(1) in largestr(1:WIN)
Search success
smallstr(i) = largestr(i)
seglen = 1
smallstr(i) = largestr(i)
seglen = 2
smallstr(i) = largestr(i)
seglen = 3
smallstr(i) != largestr(i)
nmatch = 6
strsrctmp = Empty string: 1-by-0
strdestmp = TA
Distance is dis = 0.5500

Actual stroke identification is done is two passes. In Pass1, matching is performed solely based on the "shape" of the two strokes, following the steps just

described. In Pass2, matching considers the relative sizes and positions of the stroke pair. Accordingly match-value calculation is slightly different from the one shown in Step 3 above. In this case the match value is incremented, instead of 1, by $\exp(-\|\mathbf{z}_a - \mathbf{z}_b\|^2 / \sigma^2)$, where $\mathbf{z}_a$ and $\mathbf{z}_b$ are the (z=[x,y]) coordinates of the points being matched; □ is a factor that weights the relative magnitude of the distance. Pass 2 is performed only on the top NBEST (=30) strokes that emerge from Pass 1. Therefore, even if two features are identical their contribution to match value is penalized based the distance between them. By increasing sigma ('σ') the match process can be made more and more insensitive to relative position of the strokes. Decreasing sigma has the contrary effect. The value of sigma is assigned 5 on experimental basis.

5. Character recognition:

Character recognition consists of grouping stroke labels obtained from the previous stage and converting into a suitable "character code". In the present work, ISCII (Indian Script Code for Information Interchange) code is used for representing Tamil characters. ISCII is a phonetic code, which represents composite characters in terms of component consonants and vowels. Graphically, a character in Tamil may be made of a single stroke or a single horizontal block or a sequence of contiguous horizontal blocks. The largest number of horizontal blocks in a character is 3.

The process of converting stroke labels into ISCII character code is broken up into two steps. In the first step, stroke labels corresponding to a single horizontal block are combined and the horizontal block is categorized into one of the following 5 categories:

VM – Vowel modifiers
V - pure vowel - a, A, ..etc.
X - consonant with implicit "a" modifier like ka, cha, ... ha
Y - The C-V (consonant-vowel) combinations that appear as a single horizontal block. There are 5 such cases: C-i, C-I, C-u, C-U, C-halant, N - Numbers

Simultaneously, a partial ISCII code of the character (of which the current horizontal group is a part) is also generated. This categorization of horizontal block and partial ISCII code generation are accomplished with the help of a search table (see Fig. 5).

The category of the horizontal block just determined is presented as input to FSA (see Fig. 5 and Fig. 6), which sequentially determines the complete ISCII code of the character. The FSA's arrival at end state signals termination of a character and the beginning of the next.

5. Results and Discussions

Stroke recognition is the critical part that determines the level of success in Tamil handwritten character recognition. Once the strokes are identified correctly character recognition is a deterministic process that depends on a manageable set of combination rules. Therefore currently benchmarking of our recognition system is based solely on stroke recognition. To this end data from 15 users is selected. Each user is made to write the full Tamil character set (only horizontal blocks) 10 times. Excluding redundancies this results in over 2000 stroke samples from each user, consisting of 96 stroke classes. Number of stroke samples per stroke class is variable. Strokes from user 1 are used to create a training stroke data set. The stroke data from Users 2-15 are tested on this data. Performance results are shown in Table 1.

User	% Correct	User	% Correct
2	91.5	9	88.79
3	88.97	10	83.33.
4	87.2	11	84.14
5	79.11	12	84.88
6	86.18	13	78.12
7	71.32	14	80.6
8	77.27	15	77.84

Table 1: Performance results

Online HWR studies typically handle smaller number of stroke classes since many of them deal with Latin script (26 or 52 classes). However, a study by Yaeger et al [10] presents results with 95 classes achieving 86.1% performance. Our results with 96 Tamil stroke classes compare favorably with the results of [10].

Several improvements for future work can be suggested. Some of the manual analyses used in the present work can be partly automated. For example, stroke labeling for a given script could be done directly by clustering strokes and giving them machine-generated labels.

The methods described here for Tamil handwritten character recognition can be naturally extended for other Indian scripts [8], [9]. This is because our approach has been, from the inception, to develop a framework for handwritten character recognition for Indian scripts in general, with Tamil as a special, and somewhat simpler, case. In general a word in an Indian script has multiple

characters, and a character might be spread over multiple horizontal blocks (see Fig. 7). Once all the strokes in a horizontal block are identified, the horizontal block itself must be classified as it is done for Tamil in Section 5. This is then presented to an FSA, which updates an ISCII code incrementally, until it reaches an end state signaling character termination. At this stage the ISCII code of the character is appended to an ISCII stream and processing moves on to next character (or horizontal block). This general framework is valid for all Indian scripts excepting those of Perso-Arabic family.

Processing a single horizontal block is a simple affair in Tamil. For other Indian scripts however, stroke grouping becomes a more complicated affair. Both the levels of stroke proximity analysis – contact/enclosure and X-overlap – have to be used. Barring possibly Malayalam, for most other Indian scripts, the search table for categorizing horizontal block will have multiple tiers since a single horizontal block will be have to be divided into multiple 'stroke groups'. But for this variation, the general approach of – structure-based flexible matching for single strokes, stroke combination into horizontal blocks (or stroke groups), and final determination of character termination and character code using FSA – seems to generally hold good for all Indian scripts (except scripts of Perso-Arabic origin like Urdu about which the authors claim no familiarity). Other learnable approaches like neural networks for all aspects of the present problem are currently being investigated.

6. References:

[1] R. Plamondon, D. Lopresti, L.R.B. Shoemaker and R. Srihari, "On-line Handwriting Recognition," Encyclopedia of Electrical and Electronics Eng., J.G. Webster, ed., vol. 15, pp.123-146, New York: Wiley, 1999.

[2] S.D. Connel, A.K. Jain, "Template-based online character recognition," Pattern Recognition, vol 34, pg1-14, 2001.

[3] L.R.B. Schomaker, H.L. Teulings, "A handwriting recognition system based on the properties and architectures of the human motor system," Proc. IWFHR, CENPARMI Concordia, Montreal, 1990, pp195-211.

[4] K.F.Chan, & D.Y.Yeung, "Elastic structural mapping for online handwritten alphanumeric character recognition," Proc. of 14th International Conference on Pattern Recognition, Brisbane, Australia, August, 1998, pp 1 508-1511.

[5] X.Li, R.Plamondon, M.Parizeau, "Model-based on-line handwritten digit recognition," Proc. of 14th Intl. Conf. On Pattern Recognition, Brisbane, Australia, August, 1998, pp.1134-1136.

[6] S.Manke, U.Bodenhausen, "A connectionist recognizer for on-line cursive handwriting recognition," Proc. of ICASSP' 94, vol.2, 1994, pp 633-636.

[7] V.S. Chakravarthy & B. Kompella, "The Shape of Handwritten Character," *Pattern Recognition Letters,* Vol. 24, No. 12, August, 2003.

[8] Gowri Shankar, V. Anoop and V.S. Chakravarthy, "LEKHAK [MAL]: A System for online recognition of handwritten Malayalam characters," National Conference on Communications, IIT, Madras, January, 2003.

[9]M. Srinivas Rao, Gowrishankar, V.S.Chakravarthy, "Online Recognition of Handwritten Telugu Characters," International Conference on Universal Knowledge and Language – 2002, 25th-29th November 2002, Goa,India, 2002.

[10] L.S. Yaeger, B.J. Webb, R.F.Lyon, "Combining neural networks and context-driven search for online, printed handwriting recognition in the Newton," AI Mag. (1998), pp73-89.

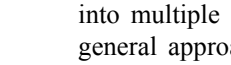


Fig 1: The vowel modifier appears as a horizontally isolatable structure. 'X' is any consonant; its pre- and suffix structures represent the vowel modifier 'O'.

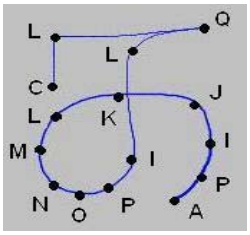


Fig.4: Feature String for 'ka': CLQLIPONMLKJIPA

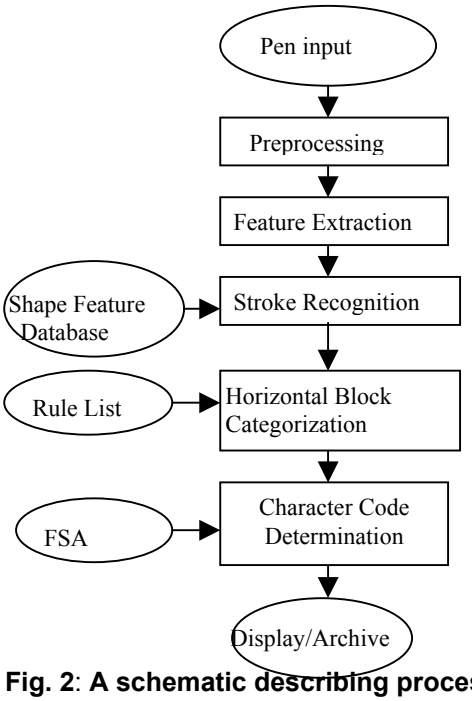
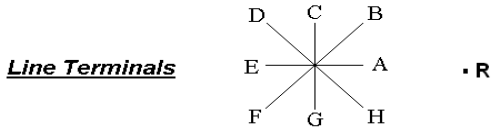


Fig. 2: **A schematic describing process flow of the online Tamil character recognition system.**



Other Features

Feature ID	Description	Example
O	X-bump, positive Curvature	
K	X-bump, negative Curvature	
M	Y-bump, positive Curvature	
I	Y-bump, negative Curvature	
P	+45° bump, positive Curvature	
L	+45° bump, negative Curvature	
N	-45° bump, positive Curvature	
J	-45° bump, negative Curvature	
Q	Cusp Point	

Fig. 3: **The 18 Shape Features**

Current Input →	1	2	3	4	5	6	7	8
State ↓	X	ி	ே	ை	ொ	ட	ன	Y,V, N
1	2	3	5	7	8	0	0	9
2	0	0	0	0	0	9	0	0
3	4	0	0	0	0	0	0	0
4	0	0	0	0	0	9	9	0
5	6	0	0	0	0	0	0	0
6	0	0	0	0	0	9	0	0
7	9	0	0	0	0	0	0	0
8	0	0	0	0	0	0	9	0
9	0	0	0	0	0	0	0	0

Fig 5: FSA Table

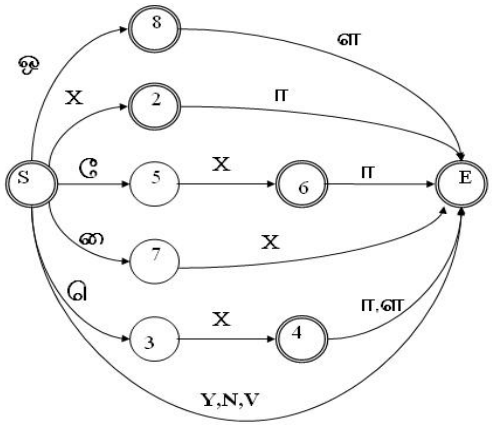


Fig 6: FSA Transition Diagram. For example, when ிசட comes the transition will be S → 3 → 4 → E

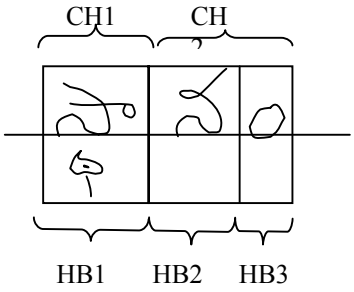


Fig 7: **In general a word in an Indian script has multiple characters, and a character might be spread over multiple horizontal blocks. The above Telugu word has 3 horizontal blocks and 2 characters: Ch1 = HB1 and Ch2 = HB2 + HB3. This is the typical structure of all Indian scripts (except those of Perso-Arabic family).**