# Lexicon-free recognition strategies for online handwritten Tamil Words

A Thesis

Submitted For the Degree of

**Doctor of Philosophy**

in the Faculty of Engineering

by

**Suresh Sundaram**


Electrical Engineering

Indian Institute of Science

BANGALORE – 560 012

DECEMBER 2011

# Acknowledgements

I thank my advisor Prof. A G Ramakrishnan, who really supported me in my exploration of novel ideas. I always was inspired by his advice on adopting a lateral thinking approach to solve a problem. His invaluable guidance, encouragement and constructive feedback from time to time has been a rewarding experience to me. I acknowledge the faculty of the Electrical Engineering Department for the excellent courses they offered. The constructive feedbacks from Prof. P S Sastry on the style of technical presentation was really helpful. I thank the members of the comprehensive examination board, Prof Bhattacharya and Prof Jamadagni for their constructive inputs to my work. I am grateful to all the staffs of the department for their co-operation and friendly moral support throughout.

I have benefitted immensely from my colleagues at IISc - Ananth, Anil, Anoop, Avinash, Haricharan, Harini, Mahadev, Naresh, Rituraj, Sanath, Shiva and Shashi. Their friendly attitude is something I would really cherish. Thanks to the company of Vikram, Vijita, Kasar and Arul, tea and coffee breaks were a stress buster. Special thanks to Ranjani, Dinesh, Kasar, Neelam, Deepak and Arul for critically reviewing parts of this thesis. A big thank you to Chandrakala, Nethra, Archana, Shanthi and Saraswathi for their efforts in collecting and ground-truthing data used for this research. Lastly, I would like to thank my parents, my brother, sister-in law and niece Maadhavi who have been a great moral support and an inspiration during my long academic journey.

# Abstract

In this thesis, we address some of the challenges involved in developing a robust writer-independent, lexicon-free system to recognize online Tamil words. Tamil, being a Dravidian language, is morphologically rich and also agglutinative and thus does not have a finite lexicon. For example, a single verb root can easily lead to hundreds of words after morphological changes and agglutination. Further, adoption of a lexicon-free recognition approach can be applied to form-filling applications, wherein the lexicon can become cumbersome (if not impossible) to capture all possible names. Under such circumstances, one must necessarily explore the possibility of segmenting a Tamil word to its individual symbols.

Modern day Tamil alphabet comprises 23 consonants and 11 vowels forming a total combination of 313 characters/aksharas. A minimal set of 155 distinct symbols have been derived to recognize these characters. A corpus of isolated Tamil symbols (IWFHR database) is used for deriving the various statistics proposed in this work. To address the challenges of segmentation and recognition (the primary focus of the thesis), Tamil words are collected using a custom application running on a tablet PC. A set of 10000 words (comprising 53246 symbols) have been collected from high school students and used for the experiments in this thesis. We refer to this database as the 'MILE word database'.

In the first part of the work, a feedback based word segmentation mechanism has been proposed. Initially, the Tamil word is segmented based on a bounding box overlap criterion. This dominant overlap criterion segmentation (DOCS) generates a set of

candidate stroke groups. Thereafter, attention is paid to certain attributes from the resulting stroke groups for detecting any possible splits or under-segmentations. By relying on feedbacks provided by

- a priori knowledge of attributes such as number of dominant points and inter-stroke displacements
- the recognition label and likelihood of the primary SVM classifier
- linguistic knowledge

on the detected stroke groups, a decision is taken to correct it or not. Accordingly, we call the proposed segmentation as 'attention feedback segmentation' (AFS). Across the words in the MILE word database, a segmentation rate of 99.7% is achieved at symbol level with AFS. The high segmentation rate (with feedback) in turn improves the symbol recognition rate of the primary SVM classifier from 83.9% (with DOCS alone) to 88.4%.

For addressing the problem of segmentation, the SVM classifier fed with the $x$-$y$ trace of the normalized and resampled online stroke groups is quite effective. However, the performance of the classifier is not robust to effectively distinguish between many sets of similar looking symbols. In order to improve the symbol recognition performance, we explore two approaches, namely reevaluation strategies and language models.

The reevaluation techniques, in particular, resolve the ambiguities in base consonants, pure consonants and vowel modifiers to a considerable extent. For the frequently confused sets (derived from the confusion matrix), a dynamic time warping (DTW) approach is proposed to automatically extract their discriminative regions. Dedicated to each confusion set, novel localized cues are derived from the discriminative region for their disambiguation. The proposed features are quite promising in improving the symbol recognition performance of the confusion sets. Comparative experimental analysis of these features with $x$-$y$ coordinates are performed for judging their discriminative power. The resolving of confusions is accomplished with expert networks, comprising discriminative region extractor, feature extractor and SVM. The proposed techniques improve the symbol recognition rate by 3.5% (from 88.4% to 91.9%) on the MILE word database

over the primary SVM classifier.

In the final part of the thesis, we integrate linguistic knowledge (derived from a text corpus) in the primary recognition system. The biclass, bigram and unigram language models at symbol level are compared in terms of recognition performance. Amongst the three models, the bigram model is shown to give the highest recognition accuracy. A class reduction approach for recognition is adopted by incorporating the language bigram model at the akshara level. Lastly, a judicious combination of reevaluation techniques with language models is proposed in this work. Overall, an improvement of up to 4.7% (from 88.4% to 93.1%) in symbol level accuracy is achieved.

The writer-independent and lexicon-free segmentation-recognition approach developed in this thesis for online handwritten Tamil word recognition is promising. The best performance of 93.1% (achieved at symbol level) is comparable to the highest reported accuracy in the literature for Tamil symbols. However, the latter one is on a database of isolated symbols (IWFHR competition test dataset), whereas our accuracy is on a database of 10000 words and thus, a product of segmentation and classifier accuracies. The recognition performance obtained may be enhanced further by experimenting on and choosing the best set of features and classifiers. Also, the word recognition performance can be very significantly improved by using a lexicon. However, these are not the issues addressed by the thesis. We hope that the lexicon-free experiments reported in this work will serve as a benchmark for future efforts.

# Notation and Abbreviations

| | |
|---|---|
| SVM | support vector machine |
| DOCS | Dominant overlap criterion segmentation |
| AFS | Attention-feedback segmentation |
| DTW | Dynamic time warping |
| DTW-DDH | DTW discriminative distance histogram |
| DR | Discriminative region |
| $\{a_1, a_2....a_6\}$ | attention points |
| $b$ | bias term used in SVM |
| $\{b_1, b_2....b_{m-1}\}$ | bounding box to stroke displacements for a $m$-stroke stroke group |
| $b_{max}$ | maximum bounding box to stroke displacement for a stroke group |
| $\mathbf{b}$ | base consonant trace extracted from component extractor |
| $c$ | number of Tamil symbols |
| $C$ | RBF learning parameter used in SVM training |
| $\mathbf{C}$ | confusion matrix |
| $c_{ij}$ | $(i,j)^{th}$ element in confusion matrix |
| $c_T(i,j)$ | number of confusions for symbol pair $(\omega_i, \omega_j)$ |
| $C_b$ | classifier for base consonants |
| $C_i$ | classifier for CV combinations of /i/ vowel |
| $C_I$ | classifier for CV combinations of /I/ vowel |
| $C_m$ | classifier for vowel modifiers of /i/ and /I/ vowels |
| $C_p$ | classifier for pure consonants |
| $C_u$ | classifier for CV combinations of /u/ vowel |
| $C_U$ | classifier for CV combinations of /U/ vowel |
| $C_v$ | classifier for pure vowels |

| | |
|---|---|
| $C_o$ | classifier for symbols (ா, ி, ீ, ை, ஃ and ஸ்ரீ) |
| $(c1, c2)$ | a confusion pair |
| $C_{ij}$ | classifier for classes i and j |
| $d(i, j)$ | dissimilarity measure used in DTW |
| $d^v_{fl}$ | Euclidean distance between first and last sample points of vowel modifier **v** |
| $d_{max}$ | maximum stroke to stroke displacement in a stroke group |
| $d^{S_M}_{max}$ | maximum stroke to stroke displacement for stroke group $S_M$ |
| $f_i(c1, c2)$ | $i^{th}$ feature for disambiguating confusion pairs $(c1, c2)$ |
| $F_0, F_1....F_7$ | sets of forbidden symbols used in the class reduction approach of akshara-level language models |
| $g$ | Between $g^{th}$ and $(g+1)^{th}$ stroke in a stroke group, the minimum vertical inter-stroke distance occurs |
| $G_1 - G_8$ | groups created based on linguistic similarity of Tamil symbols |
| $G^{\omega_i}$ | group assigned to symbol $\omega_i$ |
| $\widetilde{h}^{BB}_{min}$ | overall minimum bounding box height across symbols in the IWFHR training database |
| $H$ | entropy |
| $\{h_1, h_2....h_{m-1}\}$ | inter stroke vertical distances in a $m$-stroke stroke group |
| $h_{min}$ | minimum inter stroke vertical distance in a stroke group |
| $\mathcal{H}$ | high dimensional feature space |
| $\widetilde{h}_i$ | minimum bounding box height of symbol $\omega_i$ |
| $l_i$ | label of sample $x_i$ |
| $l^v_T$ | arc length of vowel modifier **v** |
| $\mathbf{L}$ | likely candidates used for the akshara bi-gram model |
| $K(\mathbf{x}, \mathbf{x}_i)$ | kernel function in SVM |
| $m$ | number of strokes in a stroke group |
| $n$ | number of strokes in a Tamil word |
| $n_P$ | number of resampled points in a preprocessed symbol |
| $N^{S_i}$ | number of dominant points in a stroke group $S_i$ |
| $N^{\omega_i}_{Tr}$ | number of training samples of symbol $\omega_i$ |

| | |
|---|---|
| $N_{Tr}^{c1}$ | number of training samples of symbol c1 |
| $N_{Tr}^{c2}$ | number of training samples of symbol c2 |
| $N_T$ | total number of occurrences of symbols in the MILE corpus |
| $N_s(\omega_i)$ | number of occurrences of symbol $\omega_i$ in the MILE corpus |
| $N_{ss}(\omega_i, \omega_j)$ | number of occurrences of symbol $\omega_j$ following $\omega_i$ in the corpus |
| $N_{cs}(c_i, \omega_j)$ | number of occurrences of symbol $\omega_j$ following character $c_i$ in the corpus |
| $N_{sc}(\omega_i, c_j)$ | number of occurrences of character $c_j$ following symbol $\omega_i$ in the corpus |
| $N_{cc}(c_i, c_j)$ | number of occurrences of character $c_j$ following $c_i$ in the corpus |
| $N_{Tr}$ | total number of training samples for SVM classifier |
| $N_w$ | number of words for computing the perplexity of a language model |
| $O_k^c$ | degree of overlap used in DOCS |
| $\tilde{p}$ | number of stroke groups generated in DOCS |
| $p$ | number of stroke groups resulting from AFS |
| $P$ | perplexity measure for language models |
| $P(\omega_{top}^{k_1})$ | likelihood for the stroke group $S_{k_1}$ |
| $P(\omega_{top}^{k_2})$ | likelihood for the stroke group $S_{k_2}$ |
| $P(\omega_{top}^{k})$ | likelihood for the stroke group $S_k$ |
| $P(\omega_{top}^{adj(k)})$ | likelihood for the adjacent stroke group of $S_k$ |
| $P(\omega_{top}^{M})$ | likelihood for the merged stroke group |
| $P(\omega_i)$ | prior probability |
| $P(\omega_j \vert \omega_i)$ | probability of symbol $\omega_j$ following $\omega_i$ in the MILE corpus |
| $P(\omega_i \vert \omega_{i-1})$ | probability of symbol $\omega_i$ following $\omega_{i-1}$ in the corpus |
| $P(\omega_i \vert G^{\omega_i})$ | probability of symbol $\omega_i$ in group $G^{\omega_i}$ |
| $P(G^{\omega_j} \vert G^{\omega_i})$ | probability of group $G^{\omega_j}$ following group $G^{\omega_i}$ |
| $q$ | Between $q^{th}$ and $(q+1)^{th}$ stroke, the maximum bounding box to stroke displacement occurs |
| $q_1, q_2$ | input sequences for the DTW algorithm |
| $r_i$ | recognition rate for symbol $\omega_i$ in the IWFHR test set |
| $r_{eff}$ | overall effective recognition rate of symbols in the IWFHR test set |

| | |
|---|---|
| $\widetilde{s}_i$ | $i^{th}$ stroke of a Tamil word |
| $S_k$ | $k^{th}$ stroke group |
| $S_M$ | combined stroke group |
| $S_{adj(k)}$ | stroke group adjacent to $S_k$ |
| $S_{k_1}, S_{k_2}$ | the first and second split parts of stroke group $S_k$ |
| $T_r^d$ | threshold for net distance covered in vowel modifier $\mathbf{v}$ |
| $T_\#^d$ | threshold for number of sample points for $\mathbf{v}$ to be a dot |
| $T_{y1}^d$ | threshold of the first $y$-coordinate for $\mathbf{v}$ to be a dot |
| $T_{ym}^d$ | threshold of the minimum $y$-coordinate for $\mathbf{v}$ to be a vowel modifier |
| $T_\theta$ | cumulative angle threshold for generating dominant points |
| $T_d$ | threshold used on the cost for obtaining the DTW-DDH |
| $T_{d_{max}}(\omega_{top}^M)$ | threshold set on $d_{max}$ for symbol $\omega_{top}^M$ to decide merging of over-segmented stroke groups |
| $T_{dp}^{max}(\omega_{top}^M)$ | threshold set for the maximum number of dominant points for symbol $\omega_{top}^M$ to decide to split an under-segmented stroke group |
| $T_P^{min}(\omega_{top}^k)$ | threshold set for the minimum likelihood for symbol $\omega_{top}^k$ to decide to merge $S_k$ with $S_{adj(k)}$ |
| $T_o^p(\omega_{top})$ | threshold set for the vertical overlap of dot with base conso--nants in the pure consonant of $\omega_{top}$ to avoid undesirable merges |
| $\mathbf{V}$ | vocabulary set of symbols |
| $\mathbf{v}$ | vowel modifier trace obtained from the component extractor |
| $\mathbf{v}_\#$ | number of sample points in the trace of vowel modifier |
| $w_i$ | low pass filter weights used for Gaussian smoothing (for pre--processing the input symbol) |
| $\{(\mathbf{x}_i, l_i), 1 \leq i \leq N_{Tr}\}$ | feature description with labels |
| $X$ | instance of training sample |
| $\mathbf{x}_b$ | concatenated $x$-$y$ features for base consonant $\mathbf{b}$ |
| $\mathbf{x}$ | concatenated $x$-$y$ coordinates of the preprocessed symbol |

| | |
|---|---|
| $x_{min}^{S_k}$ | $x$-minimum of $k^{th}$ stroke group |
| $x_{max}^{S_k}$ | $x$-maximum of $k^{th}$ stroke group |
| $x_{Mg}^{v}$ | global $x$-maximum of vowel modifier **v** |
| $x_{l}^{v}$ | last $x$-coordinate of vowel modifier **v** |
| $x_{Mg}^{\Re(c1,c2)}$ | global $x$-maximum in DR $\Re(c1, c2)$ |
| $x_{mg}^{\Re(c1,c2)}$ | global $x$-minimum in DR $\Re(c1, c2)$ |
| $x_{l}^{\Re(c1,c2)}$ | last $x$-coordinate in DR $\Re(c1, c2)$ |
| $y_{Mg}^{v}$ | global $y$-maximum of vowel modifier **v** |
| $y_{m}^{v}$ | global $y$-minimum of vowel modifier **v** |
| $y_{1}^{v}$ | first $y$-coordinate of vowel modifier **v** |
| $y_{mg}^{\Re(c1,c2)}$ | global $y$-minimum in DR $\Re(c1, c2)$ |
| $y_{Mg}^{\Re(c1,c2)}$ | global $y$-maximum in DR $\Re(c1, c2)$ |
| $y_{ml}^{\Re(c1,c2)}$ | last encountered $y$-minimum in DR $\Re(c1, c2)$ |
| $y_{Ml}^{\Re(c1,c2)}$ | last encountered $y$-maximum in DR $\Re(c1, c2)$ |
| $y_{Mf}^{\Re(c1,c2)}$ | first encountered $y$-maximum in DR $\Re(c1, c2)$ |
| $y_{max}^{S_k}$ | $y$-maximum of $k^{th}$ stroke group |
| $y_{min}^{S_k}$ | $y$-minimum of $k^{th}$ stroke group |
| $\mathcal{W}^*$ | optimal warping path in DTW |
| $W$ | input word |
| $W_T$ | set of words |
| $\mathbf{w}$ | model weights obtained from SVM training |
| $\alpha$ | resolution incorporation factor for data collection devices |
| $\beta$ | weighing factor used in language model |
| $\gamma$ | RBF parameter for SVM training |
| $\delta$ | threshold set for obtaining confusions for symbol $\omega_i$ |
| $\omega_i$ | symbol label |
| $\Omega$ | set of symbols that get confused with $\omega_i$ |
| $\omega_g$ | label from the primary SVM classifier |
| $\omega_b$ | label of base consonant after base consonant reevaluation module |

| | |
|---|---|
| $\omega_b^r$ | reevaluated label of base consonant after disambiguation with expert |
| $\omega_g^r$ | reevaluated label of input symbol after disambiguation with expert |
| $\omega_v$ | reevaluated label of vowel modifier **v** |
| $\omega_r$ | general notation for the label of input pattern after reevaluation |
| $\mu_y^{S_k}$ | mean $y$ coordinate of $k^{th}$ stroke group |
| $\mu_x^{S_k}$ | mean $x$ coordinate of $k^{th}$ stroke group |
| $\psi(i,j)$ | cumulative distance for DTW |
| $\Re(c1,c2)$ | discriminative region (DR) for confusion pair $(c1,c2)$ |
| $\Re^d$ | $d$ dimensional data |
| $\phi(x)$ | mapping function used in SVM |
| $\sigma$ | variance of gaussian LPF used for Gaussian smoothing (to preprocess the input symbol pattern) |
| $\xi_i$ | penalty factor used in non-linear SVM training |

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## Abstract

*In this chapter, we present an overview of the literature on handwriting recognition systems. The motivation behind the need to develop online handwriting recognition technologies for Indic scripts and lexicon-free approaches is emphasized, leading to the primary focus of the thesis. Finally, a comprehensive survey of the state of art of online handwriting recognition systems, with a specific emphasis on Indic scripts, is provided.*

## 1.1 Handwriting recognition

Across various generations of the human race, writing has evolved itself as a convenient mode to convey information. There has been an emergence of sophisticated digital computers with varied input methods in the recent years. However, usage of keyboards can become cumbersome especially with small form-factor and hand-held devices. Keeping this aspect in mind, devices offering a pen based interface have been developed and released in the market, that are quite small in size. These devices, referred as hand-held devices are convenient for usage and portable. In the coming days, with increase in their demand, they are bound to be quite affordable. A distinctive characteristic of hand-held computing devices is the use of electronic pen (or stylus) to input data on a

pressure-sensitive screen. The emerging area of pen computing refers to computers and applications in which electronic pen is the main input device [1]. This includes pen-based mobile computing devices such as personal digital assistants (PDA) and other palm top devices. Nowadays, these devices are commonly used for field data collection and as teaching aids in universities.

Handwriting recognition refers to the intelligence provided to a machine to receive, analyze and interpret intelligible handwritten input from sources as varied as paper, photographs, touch-screens and pen-based devices. The basic input to a handwriting recognition system is a pattern that represents a handwritten material. In fact, prior to feeding inputs to the system, this pattern should be digitized. Based on the way in which the pattern is digitized and provided to the system, handwriting recognition systems are classified as either online or offline [2].

In online handwriting recognition systems, we obtain handwriting data with the help of a transducer such as an electronic or tablet digitizer. Hand-held devices like PDAs are commonly employed for capturing online handwritten data. Such devices record the pen-tip information as a sequence of $(x, y)$ coordinates of data points sampled uniformly over time. In other words, pen-based inputting incorporated with an online handwriting recognition system provides a pen-paper like interface to potential users. Fig. 1.1 shows a tablet PC with the electronic pen/stylus for recording data. On the other hand, in offline recognition systems, we capture the data optically by scanning the handwritten material in the form of an image.

For online systems, the coordinates of successive points are available as a function of time (referred to as 'temporal trace') whereas in the offline case, only the completed writing in the form of a bitmap image is available. During the collection of online data, the pen-tip movement is detected along with pen-up/pen-down states. A pen-down state occurs when the pen touches the digitizer (writing pad) and when the pen is lifted off, a pen-up state is sensed. The set of points captured between successive pen-down to pen-up states is called a stroke. Additional information such as the speed of writing, stroke number and order can be utilized for recognizing online handwritten data.

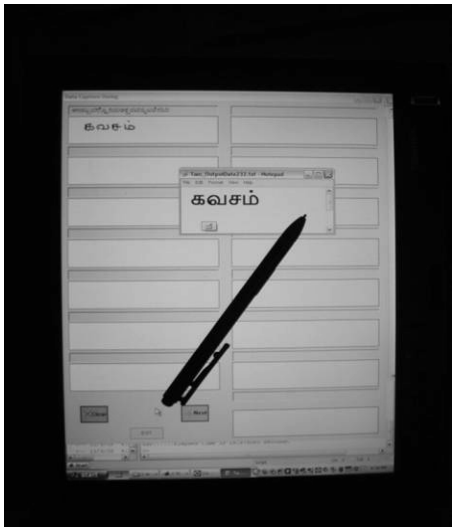



Fig. 1.1: Picture of a tablet PC with the stylus used to record the handwritten data.

Offline systems, as the name implies, are run after the data have been collected. The material ought to be written completely on a media such as paper and brought to the scanner, before digitizing it as a bitmap image. On the other hand, an online system recognizes the data (in real time) as the user writes on the electronic tablet. Being more interactive in nature, adaptation of the writer to machine and machine to the writer are possible in online handwriting recognition systems [3, 4].

Technology for online recognition of handwriting can be incorporated into a wide range of devices and applications ranging from messaging on personal devices to form-filling applications at government offices. There is also the possibility of using it in conjunction with speech synthesis, thereby empowering people with vocal disability to communicate with others. Handwriting can be utilized as a mode to create web content in Indian languages. Currently, online handwriting recognition systems are used as one of the input modes in hand-held or PDA-style computers, that might replace the keyboard-based personal computers in the future.

## 1.2 Categories of online handwriting recognition

Recognition accuracy is an important parameter for judging the performance of an online handwriting recognition system. By placing constraints on the usage of the systems, one can get a reasonable accuracy. Accordingly, online systems are classified in two ways.

- **Constrained and unconstrained systems**: Systems can be developed by placing specific restrictions on writing styles. Some of them want users to write in a discrete manner and some others force users to write in a given order of strokes. On the other hand, unconstrained handwriting recognition systems allow users to freely write in their own natural way. Although these systems place no restrictions on writing styles, their recognition accuracy could be evidently lower than constrained systems.

- **Writer dependent and independent systems**: The goal of a writer-independent online system is to recognize handwriting of a variety of writing styles, while writer-dependent systems are trained to recognize handwriting of a single individual. One of the critical requirement of writer-independent systems is that they are able to recognize handwriting that they may not have seen during training. Writer independent systems are necessary for applications like online form filling. On the other hand, in writer-dependent systems, handwriting of a single individual is being trained and tested with the system. In general, writer dependent systems present a better accuracy rate compared to writer independent scenarios. Constructing writer independent systems is obviously harder than writer dependent systems. The difficulty in developing writer independent systems arises from the fact that the system is expected to handle much greater varieties of handwriting styles.

- **Lexicon based and lexicon free systems**: Handwriting recognition has been employed in applications characterized by small or fixed lexicons (such as postal address interpretation and bank check reading). The idea behind lexicon based systems is to match the recognized word against a word contained in the lexicon, thereby making the recognition accuracy dependent upon the size of the lexicon. It

is noted that the recognition accuracy reduces with increasing lexicon sizes. On the other hand, in lexicon-free systems, the recognition is performed without the aid of a dictionary. Such systems become feasible in large-scale form filling applications where it is not possible to invoke a finite lexicon for recognition.

## 1.3 Focus of the thesis

The Indian sub-continent has as many as 22 official languages and 10 scripts. In such a multilingual country, we come across a large section of the rural population, who till date, still prefer to write in their native language to English. In order to provide them with access to writing, many government documents and forms in Indian states are printed in their state language. Enabling interaction with computers in the native language through the medium of handwriting allows for better technology penetration and greater inclusion of the masses. Thus arises the need for developing online handwriting recognition (OHR) systems for Indian Languages.

Decades of research have led to the development of online word/ text recognition systems for Latin and the Chinese, Japanese, Korean (CJK) scripts [2, 5, 6, 7]. In comparison to Latin, Indic scripts exhibit a large number of characters and stroke order/ number variation. In particular, Indian scripts comprise compound symbols resulting from vowel-consonant combinations and in many cases, consonant-consonant combinations, which are absent in Latin scripts. Moreover, the closeness between some of the characters call for sophisticated algorithms. Despite these issues, very little work has been done in recognition of handwriting in Indic scripts and thus, word recognition systems for Indian languages are still in their nascent stages. As will be evident in the literature survey (to be described in section 1.5), majority of the research reported for Indian languages have either dealt with a subset of characters such as only the base characters or the numerals.

In this work, we take a step forward in the goal of developing a robust writer-independent, lexicon-free recognition system for online Tamil words. In particular, we

focus on two important aspects that have not been adequately addressed in the literature for online handwritten Indic scripts: (1) segmentation and (2) post-processing. Feedback strategies are utilized in segmenting a Tamil word to its constituent elements. The individual segments are then recognized with a classifier, referred to as the 'primary classifier'. Post-processing methods incorporate the use of domain knowledge to improve the symbol recognition performance of the primary classifier. Two approaches, namely reevaluation strategies and language models, have been sufficiently addressed in this thesis. The performance evaluation of the proposed post-processing techniques have been made with respect to that of the primary classifier. However, a comparative study of reevaluation and language models is not dealt within the realm of this work. Instead, a judicious combination of the two approaches has been found necessary for Tamil and hence adopted to improve the symbol recognition performance.

Several works on online handwritten scripts in recent literature employ lexicons of different sizes to aid in the recognition process. However, as mentioned in the earlier section, the use of a lexicon is generally restricted to a particular domain. The features are compared with those of words present in the lexicon and the most similar word is considered the recognition result. Though the usage of lexicon for recognition is highly useful for specific applications, an interesting aspect to look at would be to explore how far one can go in building a robust word recognizer without the use of a lexicon. Such an approach will be useful in certain applications like form-filling, wherein it is not feasible to invoke a finite lexicon to capture all possible proper names and addresses. Further Tamil, like other Dravidian languages, is an agglutinative language, characterized by an expanding lexicon. A single verb root can give rise to numerous new words (running into thousands) [8]. As an illustration, we list out some of the possible words that can be formed with the verb root வா /vA/ in Appendix A. This property of the script necessitates us to adopt a lexicon-free approach to recognize words. It is to be noted here that though we learn the linguistic statistics of the script from a corpus of 1.5 million words (derived from books), the proposed lexicon-free recognition approach has the potential to handle out-of-vocabulary words (words not contained in the corpus).

One can explore a segmentation-based approach to recognize words with the aid of a lexicon. However, when one cannot or does not verify the recognized word output based on a lexicon, it is very important that every character is correctly recognized. In the context of handwriting recognition of Indic scripts with one to many strokes making up a single recognizable symbol, it is crucial to ensure that, in the absence of a lexicon, a word is correctly segmented to its individual symbols. Thus, by adopting a lexicon-free approach, segmentation of online handwritten Tamil words is separately focused as an important issue in this work. In addition, the correct segmentation of handwritten words plays a vital role to their recognition.

It is worth mentioning here that the Technology Development for Indian Languages (TDIL) program of the Ministry of Information Technology of the Government of India has recently funded a consortium of universities to create resources (data collection, annotation) and systems for handwriting recognition of Indic scripts. Our laboratory is the lead institution in this consortium and is committed to developing recognition technologies for two Indian languages - Tamil and Kannada. However, the focus of this doctoral thesis is constrained to developing such technologies for Tamil.

## 1.4 Techniques for online handwriting recognition

In the current literature, online handwriting recognition techniques belong to one of the five categories discussed below

- **Primitive decomposition** identifies sub-strokes or primitives that form the common building blocks for characters [9, 10] . Examples of such building blocks includes loops, dots, crossovers, arcs, ascenders and descenders. These methods generally decompose the strokes of a character into sub-stroke pieces. A sub-stroke based approach for online Kanji character recognition is proposed in [9]. A set of sub-strokes are identified based on their direction and length. Any Kanji character is expressed as a sequence of these sub-strokes resulting in a reduced model set. A hierarchical dictionary consisting of sub-strokes, strokes, radicals and characters

is manually built for Kanji character recognition. To incorporate the variations in a sub-stroke and the co-articulation effects due to preceding and succeeding sub-strokes, context-dependent sub-stroke models are proposed in [11]. In [12], a character is first segmented into sub-stroke primitives and one observation feature vector is computed for each segment. The HMM classifier is used to recognize these individual primitives. Primitive decomposition techniques are not very robust to large variations in writing style.

- **Motor models** are a set of techniques, wherein models of stroke segments are created along with rules for connecting them to form characters. Motor models simulate the physical properties of human hand motion by representing the stroke segments with a parameterized model of the pen motion [13, 14, 15, 16]. However, these models may lack robustness for large writing style variations.

- **Elastic matching** techniques search for alignment of data points between an input character and each template character [17]. The distance between an input character and a template is the sum of distances between aligned points. The assignment of the character to a class is performed using a NN classifier [18]. In [19], a robust structural approach is proposed for recognizing on-line handwriting, wherein the manually generated stroke models are elastically matched with the structural primitives of the test data. A template-based system for online character recognition is proposed in [20], wherein the number of templates, representing the different lexeme styles of a particular character, is determined automatically.

- **Stochastic models**, as the name implies, employ a statistical framework to represent the temporal sequence of the online data. The HMM is an example of a stochastic model and is popularly used for word recognition. For recognizing words [21, 22, 23], constituent letters of a word are modeled with separate HMMs and concatenated to generate the word model. HMMs can also be employed to model sub-strokes of a letter as described in [24, 25]. HMM models are often created using features extracted from the individual sample points [26], or from the

points contained within a window which slides along the trace thereby producing a sequence of features [27]. In [3, 28], HMMs have been applied to the problem of writer adaptation.

- **Neural networks** have been found to be quite promising to the problem of online recognition. In particular, time delay neural networks (TDNN) have been used to recognize characters or character segments. Essentially, in these networks, a sliding window moves over the temporal sequence. The features extracted from the sample points within a window are fed to a feed-forward neural network. The activation level of each output node, one per class identity, gives the likelihood for the sequence of points in the sliding window to belong to that class. By sliding a window across the entire data, a sequence of likelihood values are generated, which can be used to find the best sequence of character identities using methods like dynamic time warping [29] and Viterbi search [30]. Jaeger et al. [45] presented the NPen++ online handwriting recognition system based on a multi-state TDNN, a hybrid architecture combining features of neural networks and HMMs. Two main features of a multi-state TDNN are its time-shift invariant architecture and the nonlinear time alignment procedure.

Apart from recognition, feature selection and classifier structures have been studied in [31] to identify different scripts in an online handwritten multi-script document.

## 1.5 Literature survey: Indic scripts

In this section, we present a survey of techniques proposed in the literature to recognize online Indic scripts. In particular, we outline the contributed works for seven Indic scripts.

### 1.5.1 Kannada

The maiden work in this language is that of Kunte et al. [32]. Wavelet features are extracted from the character contour and used as features. Multi-layer feed-forward neural networks with a single hidden layer are trained for recognizing the characters. In a recent work [33], a divide and conquer approach has been proposed to reduce the number of character combinations to be used for data collection. In the first level of the technique, structural and the dynamic features are utilized for reducing the compound Kannada characters to a set of 295 distinct symbols. In the second level, these 295 symbols are further divided into three distinct sets of stroke groups. PCA-based features are then derived specific to each stroke group. The subspace features of distinct stroke groups are fed to their respective nearest neighbor (NN) classifiers for classification. The results from these classifiers are then combined to generate the output character. In another work [34], statistical dynamic time warping (SDTW) has been employed to classify Kannada characters with $x$-$y$ coordinates of the trace and their first order derivatives as features. The SDTW is reported to give a 2% improvement over the conventional dynamic time warping (DTW). Orthogonal LDA on a set of PCA features have been recently attempted to the set of Kannada numerals [35].

### 1.5.2 Bangla

The earliest work pertaining to Bangla character recognition [36] focussed on utilizing the cues from the pen trajectory to derive features, while tackling the problem of stroke order variations. Neuromotor characteristics of handwriting were exploited. A direction code histogram feature has been proposed in [37] for recognition of online Bangla handwritten characters. Here, each stroke of an input online handwritten pattern is represented in terms of the direction codes. The sequence (temporal) data of online handwritten sample is divided into several sub-divisions. In each of the subdivisions, a local histogram of the direction codes is calculated and used as the feature. The MLP is trained with the basic Bangla characters for recognition. HMMs has been applied on

the stroke level in [38]. The given stroke is first divided into a number of sub-strokes. A string of features is derived at the sub-stroke level. Based on the shape similarity of the graphemes that constitute the ideal character shapes, strokes are manually grouped into classes. After the classification of all the strokes in a given input, they are used to generate the output character with the help of a look-up table.

A comparative study of the performance of a HMM classifier to a nearest-neighbor classifier (based on DTW) is made in [39]. Apart from character recognition, some preliminary work has been attempted at recognizing cursive Bangla words [40]. An analytic recognition approach, based on the position of the headline, is adopted to segment the input word to a set of sub-strokes. The segmented sub-strokes are then recognized with a modified quadratic discriminant function. Chain code histograms derived from sub-strokes are used as features. A verification module, comprising a set of rules for construction of characters from the sub-strokes, recognizes the input word. A similar segmentation and feature extraction approach has been attempted with HMMs in [41].

### 1.5.3 Telugu

To our knowledge, there has been quite a few attempts to recognize Telugu script. In the work of [42], string matching of shape based features is adopted to recognize Telugu symbols. An input stroke is represented as a string of shape features. Using this string representation, an unknown stroke is identified by comparing it with a knowledge database of shape based features. A full character is recognized by identifying all the component strokes. Rao and Ajitha [43] regard the standard Telugu characters in terms of segments that are either straight line portions or parts of circles of well defined radius. A feature set is proposed to capture the canonical shapes of symbols while filtering out the shape deviations encountered as noise. Accordingly, $x$ and $y$ extrema, direction of pen motion (clockwise/anticlockwise) and relative displacement from the previous point of the same extrema category ($x$ or $y$) are adopted as features.

In another work, a combination of time and frequency domain features has been used in a HMM framework for online Telugu symbols [44]. The time domain features

(curliness, lineness, aspect ratio, curvatures, $x$-$y$ derivatives) have been adapted from NPen++ online handwriting recognition system [45]. A modular approach has been proposed in [46] to recognize Telugu symbols. Here the recognition is performed at the stroke level. Based on the relative position of a stroke in a character, the stroke set has been divided into three subsets, namely baseline, bottom and top strokes. Classifiers for the different subsets of strokes are built using Support Vector Machines (SVMs). Character based elastic matching using various local features has also been attempted for recognizing online Telugu symbols [47]. The four different feature sets used are (1) $x$-$y$ features, (2) shape context (SC) and tangent angle (TA) features, (3) generalized shape context feature (GSC) and (4) $x$-$y$ coordinates, the normalized first and second derivatives and curvature features. Experiments are conducted with the nearest neighbor classifier operating on the DTW distance.

### 1.5.4 Devanagari

In the recent works dedicated to Devanagari script, two important problems namely, recognition and writing style identification, have been addressed. A combination of two HMM classifiers trained with online features and three NN classifiers each trained on different sets of offline features has been attempted in [48]. This combination strategy has been shown to give promising improvements in accuracy. A classifier ensemble optimized with a genetic algorithm has been proposed in [49] for online Devanagari characters. The ensemble performance is claimed to be higher than that of individual classifiers. The optimal set of classifiers is selected from a pool of SVM-based classifiers trained on various features and kernel parameters. In [50], strokes are first pre-classified into two categories based on arc length, prior to SVM classification. Script-dependent rules are then employed to generate the character from the set of output stroke labels.

In the work of [51], consonant conjuncts are broken down into individual consonant symbols. This form of linearization reduces the number of symbols. In order to further reduce the search space, a structural feature based algorithm is proposed to remove special strokes, vowel modifiers and the headline. The character recognition module

(subspace classifier) operates on the $x$-$y$ features of the residual character. As mentioned earlier, apart from recognition, clustering algorithms have been proposed to identify unique writing styles in Devanagari. In [52], an agglomerative hierarchical clustering technique is used with the nearest neighbor approach to cluster the strokes for identifying the different writing styles. Recently, as an extension to this work, a constrained stroke clustering [53] has been proposed, incorporating prior information in the form of constraints between stroke clusters.

### 1.5.5 Gurmukhi

To our knowledge, there are only two works related to recognizing Gurmukhi characters. Elastic matching technique has been used at the stroke level in [54]. The authors note that a number of large strokes appear in online cursive word handwriting. The average number of points is used as the criterion for segmentation. Accordingly, a point based segmentation scheme is employed to segment large strokes into smaller ones prior to recognition. A set of high and low level features extracted from the strokes are fed as input to the elastic matching module. Based on the recognized strokes, the character is generated. Reordering of the recognized strokes is introduced in [55] for obtaining the character label. The recognition comprises three steps : identification of the strokes as dependent and major dependent ; the rearrangement of strokes with respect to their positions; the combination of strokes to recognize the character.

### 1.5.6 Malayalam

To our knowledge, there are only two related works for Malayalam. A system referred as 'LEKHAK [MAL]' has been proposed in [56] for recognizing characters. Similar to the work reported in [42], it works on the principle of string matching with shape based features. The authors report an accuracy of around 90% on a dataset of 216 strokes. In a recent work [57], a study of different preprocessing, feature selection and classification techniques has been attempted to recognize the characters in Malayalam words. Features

like moments, area, aspect ratio, length, grid occupancy and curvature have been used for the representation of the strokes. The authors claim that the directed acyclic graph (DAG) based SVM framework works well for recognizing the stroke classes. Finally, by employing a FSA, the labels for the individual characters are generated from the stroke labels.

### 1.5.7 Tamil

The earliest work on Tamil character recognition has been that of Sundaresan et al. [58]. They evaluated the performance of angle features, Fourier coefficients and wavelet features on a neural network classifier. Amongst these features, they show that wavelet features are the most effective as they retain both the intra-class similarity and inter-class differences. A combination of time-domain and frequency-domain features has been attempted with a HMM classifier in [59]. A similar set of feature combinations has been recently tested with an elastic matching approach in [47]. For writer dependent on-line handwriting recognition of isolated Tamil characters, a comparative study of elastic matching schemes is presented in [60]. Three different features are considered namely, preprocessed $x$-$y$ co-ordinates, quantized slope values and dominant point co-ordinates. A subspace based classification approach has been proposed by Deepu et al. [61]. Principal component analysis (PCA) is applied separately to feature vectors extracted from the training samples of each class. The subspace formed by the first few eigenvectors is considered to represent the model for that class. During recognition, the test sample is projected onto each subspace and the class corresponding to the one that is closest is declared as the recognition result.

Different strategies for prototype selection for recognizing handwritten characters of Tamil script are investigated in [62]. In particular, for modeling the differences in complexity of different character classes, a prototype set growing algorithm is proposed with DTW+NN as the classifier. A method of prototype learning is discussed in [63] to speed up the recognition with the DTW framework. Swethalakshmi et al. [50] propose a set of offline-like features that capture information about both the positional and

structural (shape) characteristics of the handwritten unit. The SVM is used for the classification. In [64], unique strokes in the script are manually identified and each stroke is represented as a string of shape features. The test stroke is compared with the database of such strings using the proposed flexible string matching algorithm. The sequence of stroke labels is recognized as a character using a finite state automaton (FSA). Reference [65] provides a comparative study of SDTW with HMM on Tamil symbols.

There is only one work in the literature dedicated to the recognition of online Tamil words [66]. Here, each symbol is modeled using a left-to-right HMM. Inter-symbol pen-up strokes were modeled explicitly using two-state left-to-right HMMs to capture the relative positions between symbols in the word context. Independently built symbol models and inter-symbol pen-up stroke models were concatenated to form the word models. The approach is tested with lexicons of varying sizes.

## 1.6 Summary

In this chapter, a brief overview of the classification of handwriting recognition systems is provided. In the context of Indic scripts, the need to develop handwriting recognition technologies is emphasized. Finally, a comprehensive literature survey of the state of art of online handwriting recognition systems has been provided. It is evident from the survey, that work on online recognition of Indic words is still in its nascent stages.

In the following chapter, we present the essential background material for the work reported in the thesis. Various aspects such as description of Tamil symbols, data collection and primary recognition module are described in sufficient detail.

# Chapter 2

# Background for the study


## Abstract

*In this chapter, we first provide an overview of the complete Tamil character set (that include the Grantha characters). This is followed by the description of the methodology adopted in deriving the minimal set of symbols (for recognition) from the character set. The issues pertaining to the recognition of online handwritten Tamil symbols are mentioned with illustrations. Finally, we outline the components of a rudimentary recognition system for online handwritten Tamil symbols, with support vector machines (SVM) as the primary classifier.*


## 2.1 Tamil character set

Tamil is a Dravidian language spoken predominantly by a significant population in the southern region of India. Apart from India, it has official status in Sri Lanka and Singapore. Besides, a sizeable population in Malaysia also speak Tamil. The language was given classical status by the Indian Government in 2004. Tamil is one of the few living ancient languages of the world. The first comprehensive grammar work, *Tolkappiyam*, is said to have appeared in 2000 BC.

The language is written using the 'Tamil script' and is written from left to right.

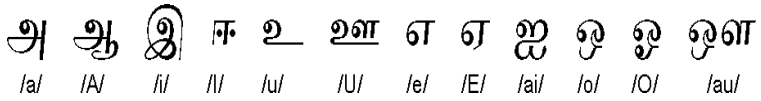



Fig. 2.1: Set of pure vowels in Tamil.

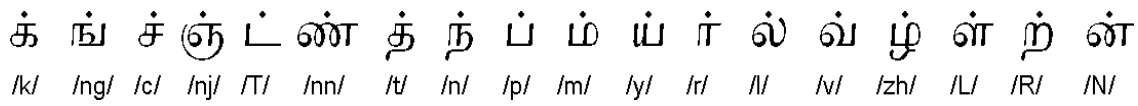



Fig. 2.2: Set of pure consonants in Tamil.

In terms of the structure of the characters used, Tamil is unrelated to the descendants of Devanagari such as Hindi, Bengali and Marathi. Traditionally, it comprises 12 pure vowels, 18 pure consonants and a special character called the *aytam* ஃ `/ah/`. Figures 2.1 and 2.2 respectively list the set of pure vowels and consonants of modern Tamil script.

Unlike Latin, Tamil has separate grapheme representations for short and long vowels. The long vowels are somewhat similar to stressed vowels in English and in addition to increased duration, they are spectrally distinct from the short vowels. In this work, we denote short vowels by the lowercase letters and the long ones by uppercase letters. Further, the diphthongs `/ai/` and `/au/` are also counted as vowels and have unique graphemes.

Each pure consonant gets modified by each of the 12 vowels to generate consonant vowel (CV) combinations. Effectively, the vowels and pure consonants combine to form $18 \times 12 = 216$ CV combinations, giving a total of 247 characters (216 CV combinations + 12 vowels + 18 pure consonants + 1 character). Figure 2.3 lists the CV combinations corresponding to the consonants க் `/k/` and ப் `/p/`.

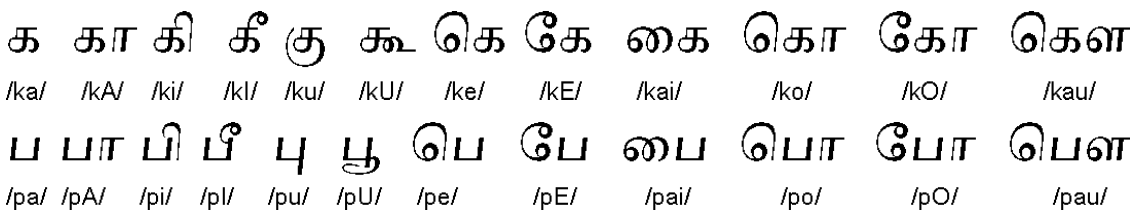
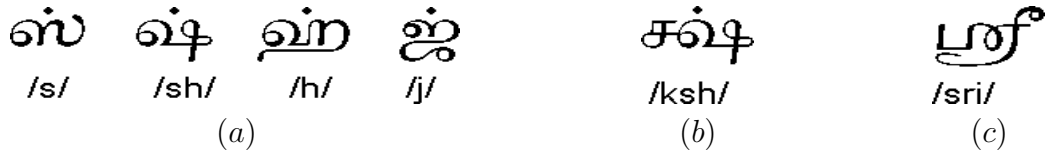



Fig. 2.3: Set of all CV combinations of `/k/` and `/p/`.




(*a*) (*b*) (*c*)

Fig. 2.4: List of characters derived from Grantha script. (a) Set of four pure consonants `/s/`, `/sh/`, `/h/`, `/j/`. (b) Consonant cluster `/ksh/`. (c) The /sri/ character.

Pure consonants modified by the inherent vowel அ `/a/` are referred to as 'base consonants'. In addition to the standard 18 pure consonants, four additional pure consonants and one consonant cluster க்ஷ `/ksh/` are derived from the Grantha script (see Fig. 2.4) to write Sanskrit words and to represent words and sounds not native to Tamil. These 5 characters together with their corresponding CV combinations increase the Tamil character set by 65 characters. A character ஸ்ரீ `/sri/` is also borrowed from Grantha. Summarizing, modern day Tamil script comprises a total of 313 characters (listed in Appendix B).

Analysis of the complete set of CV combinations in Appendix B indicates that they may appear in one of the following five forms:

- For CV combinations of இ `/i/` and ஈ `/I/`, the vowel modifier ($VM$) overlaps with the base consonant. These are illustrated in the characters கி `/ki/`, கீ `/kI/`, ழி `/zhi/`, லீ `/LI/` to state a few.

- For the CV combinations of உ `/u/` and ஊ `/U/`, the basic shape of base consonants (except Grantha) being modified are altered. Examples of such CV combinations include பு `/pu/`, ழு `/zhu/`, கு `/ku/` and கூ `/cU/`. However, for Grantha characters, the shape of the base consonant is unaltered with the discrete vowel modifier overlapping with it on top. Typical examples for such CV combinations are ஸு `/su/`, க்ஷு `/kshu/`, ஸூ `/sU/` and ஹூ `/hU/`.

- For the CV combinations of எ `/e/`, ஏ /E/ and ஐ `/ai/`, the corresponding vowel modifiers ெ, ே and ை spatially appear as a distinct/separate entity to the left of the base consonant being modified. Examples of such CV combinations include நெ `/Ne/`, யே `/yE/` and கை `/kai/`.

- The vowel modifier for ஆ /A/, written as ா appears to the right of the base consonant in the CV combination. Examples include கா /kA/, தா /tA/ and யா /yA/.

- CV combinations of ஒ /o/, ஓ /O/ and ஔ /au/ comprise two distinct entities with the base consonant sandwiched between them. The characters பொ /po/ , டோ /TO/ and கௌ /kau/ illustrate such CV combinations.

The *aytam* ஃ /ah/ is classified in Tamil grammar as being neither a consonant nor a vowel. However, in modern times it has come to be used to denote foreign sounds - for example ஃ ப is used to represent the English sound /fa/, not found in Tamil.

Even though a vowel modifier can be added to the right, left or both sides of the base consonants, the Unicode representation encodes the corresponding CV combinations in logical order. In other words, the base consonant is always encoded first, followed by the vowel modifier. The Unicode range for Tamil is U+0B80U+0BFF. The Tamil numerals rarely appear in modern Tamil texts. Instead, 'Indo-Arabic' numerals are used.

## 2.2 Choice of Tamil symbol set

Inspection of the 313 characters in Appendix B indicates redundancy, especially with respect to the way certain CV combinations are written [67]. In this section, we discuss the methodology adopted to reduce the redundancy, with the aim of coming up with a comprehensive set of distinct entities that can be employed in designing the recognition system.

- As an illustration, consider all the CV combinations of ஆ /A/. In this case, the vowel modifier ா appears as a distinct/separate entity to the right of each base consonant. From recognition point of view, it would suffice if we recognize ா separately and then append it to the corresponding base consonant to generate the CV combination, thereby reducing the number of distinct entities for the classifier.

- Similar strategies applied on the vowel modifiers of எ /e/, ஏ /E/, ஐ /ai/, ஒ /o/, ஓ

/O/ and ஔ /au/ reduce the inherent redundancy in the characters to a substantial extent.

- In addition, we observe that the vowel ஔ /au/ comprises 2 distinct entities- ஒ /o/ and ள /L/ that have already been considered as a vowel and base consonant, respectively. Hence, there is no necessity in representing it as a separate entity for recognition.

With the above analysis, it is found that a minimum set of 155 distinct entities (henceforth referred in this work as 'symbols') is sufficient to represent all the 313 characters in the Tamil alphabet (Appendix C).

We summarize the discussion by relating a Tamil character to the symbol set (refer Appendix B)

- Each CV combination of the vowels ஆ /A/, எ /e/, ஏ /E/ and ஐ /ai/ comprises 2 distinct symbols.
- Each CV combination of ஒ /o/, ஓ /O/ and ஔ /au/ comprises 3 distinct symbols.
- Each of the pure consonants, base consonants and vowels (except ஔ /au/) are represented by a distinct symbol.
- Each CV combination of இ /i/, ஈ /I/, உ /u/ and ஊ /U/ is a distinct symbol.
- The vowel ஔ /au/ is represented with 2 symbols.

All the 313 characters shown in Appendix B can be obtained (and hence recognized) as a combination of these symbols. The 313 characters of the script are also referred by the name '*aksharas*'.

We would like to mention here that, in contrast to Tamil, there are Indic scripts like Telugu, Kannada and Hindi for which the number of *aksharas* run into thousands.

## 2.3 Datasets used for the experiments

In this section, we outline the databases employed for experimentation. A corpus of isolated Tamil symbols (IWFHR database) is publicly available for research [68]. This database comprises 50,385 training samples and 26,926 test samples. We utilize this corpus for generating the various statistics of Tamil symbols in the subsequent chapters. To address the challenges of segmentation and recognition of Tamil words (the primary focus of this work), words are collected using a custom application running on a tablet PC. We have ensured that all the writers who participated in the data collection activity are native Tamil speakers, who currently write in that language, at least irregularly. High school students from across 6 educational institutions in the Indian state of Tamil Nadu contributed in building the word data-base of 10000 words, hereafter referred to as the 'MILE word database' [67]. The words have been divided into 40 sets, each comprising 250 words. Two sets of 250 words (denoted as DB1 and DB2) has been employed for validating the proposed strategies in this thesis. Owing to the comparable resolution of our input device to that used for the IWFHR dataset (a sampling rate of 1200 Hz and a spatial resolution of 2500 dpi along both $X$ and $Y$ directions), statistical analysis performed on the symbols in the IWFHR database are applicable to the Tamil symbols in the MILE word database. Figures 2.5 (a)-(j) present a few sample words from our database.

## 2.4 Challenges in recognizing Tamil symbols

In this section, we present the various issues encountered while recognizing an online handwritten Tamil symbol. These need to be taken into account in the design of robust recognition systems. Many of these issues generalize to the online handwriting recognition of non-Indic scripts as well.

- **Lack of a finite vocabulary**: Unlike English and Hindi, Tamil is very rich morphologically. Typically a verb root can transform itself to thousands of derived words by adding suffixes for number, gender, tense/emphasis, interrogation and

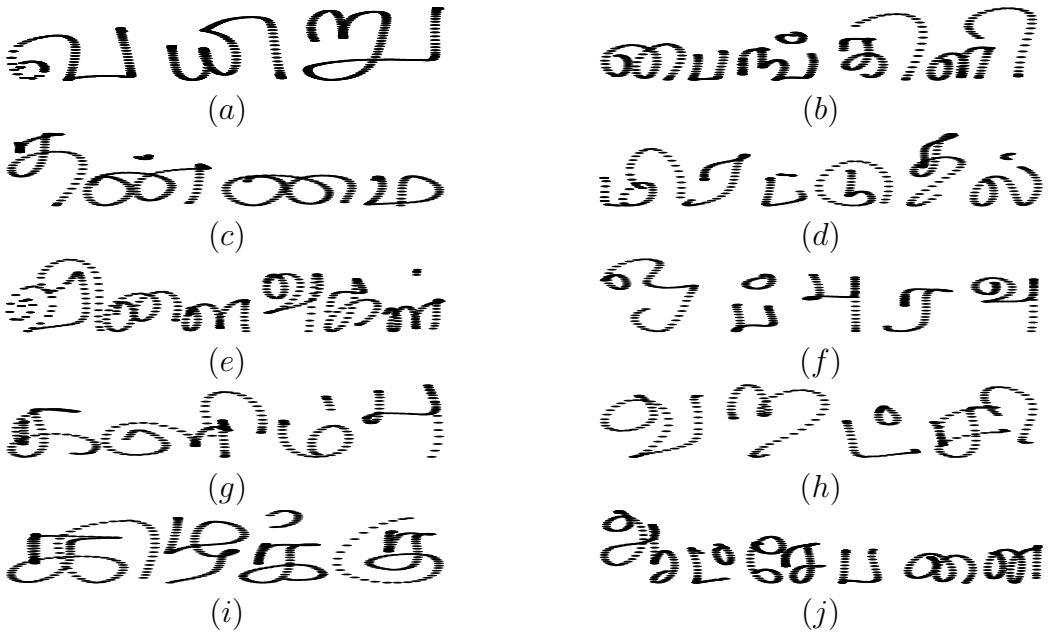


Fig. 2.5: Sample words from the MILE word database.

conversion to noun. Similarly, any noun including proper nouns and common nouns can give rise to hundreds of derived words [8]. Thus, the language cannot be confined within a finite lexicon. This in turn necessities lexicon-free approaches to recognition.

- **Inter-class similarity**: There is a high degree of visual similarity within each of several sets of Tamil symbols. When recognized with only global cues, such symbols are likely to get confused with one another. This in turn calls for reliable, class-specific highly distinctive features to describe the shapes of these characters for better discrimination. Figure 2.6 lists a few visually similar looking symbols. Such similarity of characters arise in Japanese and Chinese scripts as well.

- **Variations in writing styles**: There are a few Tamil symbols that could be written in different styles that are phonetically identical but significantly different in visual appearance. Figure 2.7 illustrates three possible lexemic styles of the symbol தி /ti/. Such different writing styles are well captured under writer independent scenarios.

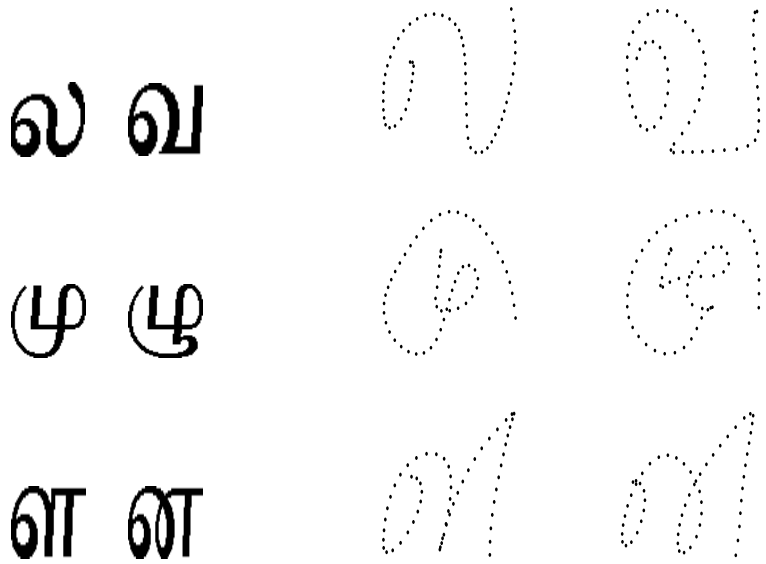
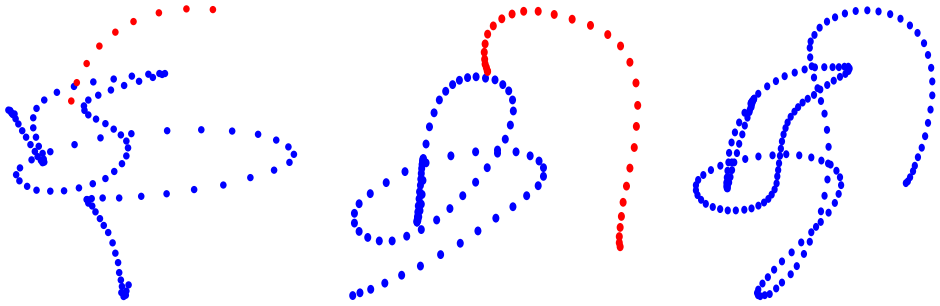


Fig. 2.6: Examples of similar looking pairs of symbols in Tamil. The printed samples as well as handwritten ones are shown.



(a) (b) (c)

Fig. 2.7: Illustration of lexemic styles for the symbol /ti/. The traces of the individual strokes of a style are highlighted with separate colors.

- **Order of writing the symbols**: Variations arise in the writing order of symbols in the CV combinations. As discussed in the previous section, for CV combinations of எ /e/, ஏ /E/ and ஐ /ai/, the vowel modifier is written before the base consonant. However, the writing of the base consonant precedes the vowel modifier in the CV combinations of ஆ /A/, இ /i/, ஈ /I/ , உ /u/ and ஊ /U/. In the CV combinations of ஒ /o/, ஓ /O/ and ஔ /au/, parts of the vowel modifiers are written before and after the base consonant. This prior knowledge of the symbol order needs to be considered while analyzing the linguistic statistics of symbols in a given corpus. Such modifiers and hence such kind of writing order of symbols, are absent for

Latin scripts.

- **Variations at the stroke level**: In general, variations in stroke order, number and direction are prevalent in Tamil symbols. Table 2.1 presents some of the possible ways of writing the symbol தி /ti/. We see that the number of strokes for representation of this symbol varies between 1 and 3. However, compared to Oriental scripts, Tamil symbols are written with far lesser number of strokes. The number of strokes for certain Chinese and Japanese characters can be predominantly high (greater than 30). In addition, such characters present variations in stroke order and direction.

## 2.5 Overview of the basic recognition module

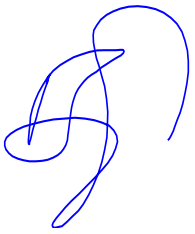
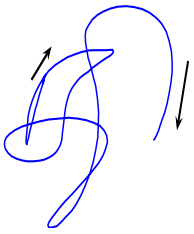
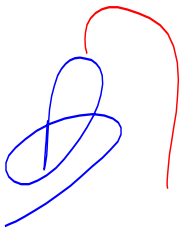
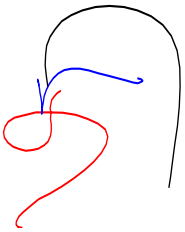
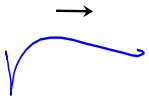
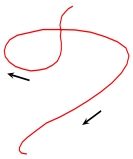
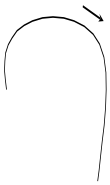
In this section, we present the details of a rudimentary recognition system used in our experiments. The recognizer has been developed to work on isolated Tamil symbols. The following subsection outlines the preprocessing steps and feature extraction that result in a feature vector of fixed dimensions from the input pen position stream. Subsection 2.5.2 outlines the details of the primary classifier used in recognizing a test symbol.

### 2.5.1 Preprocessing

As discussed in Chapter 1, the online handwritten symbol, captured from the digitizer, is a sequence of $x$-$y$ coordinates with pen-up and pen-down events. The pre-processing step, applied prior to recognition, compensates for variations in time, scale and velocity [60, 61]. It comprises 3 steps : (1) smoothing (2) normalization (3) resampling. Smoothing reduces the amount of high frequency noise in the input resulting from the capturing device or jitters in writing. Each stroke is smoothed independently using a $2N_t + 1$ tap Gaussian low-pass filter with coefficients:

$$w_i = \frac{e^{-\frac{i^2}{2\sigma^2}}}{\sum_{j=-N_t}^{N_t} e^{-\frac{j^2}{2\sigma^2}}} \tag{2.1}$$

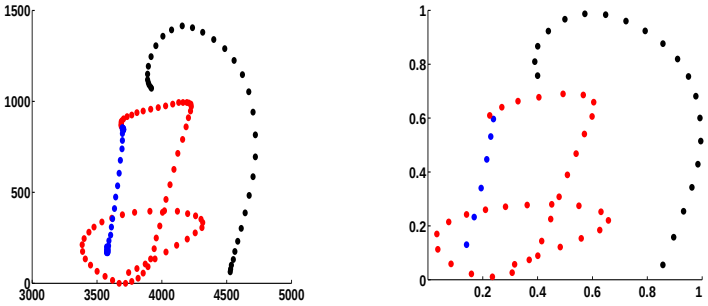
Table 2.1: Stroke variations for the symbol `/ti/`. The patterns (a), (b) and (c) are written with one, two and three strokes, respectively. The individual strokes are highlighted with different colors, and the directions of the traces depicted with arrows.

| | Symbol | Stroke 1 | Stroke 2 | Stroke 3 |
|---|---|---|---|---|
| (a) |  |  | | |
| (b) |  |  |  | |
| (c) |  |  |  |  |

Here $\sigma^2$ is the variance of the Gaussian function. For our experiments, we chose $N_t = 2$ and $\sigma^2 = 0.6$ respectively.

To eliminate variability due to size differences, the bounding box of the character is obtained and transformed to a fixed size (size normalization). Both $x$ and $y$ coordinates are separately mapped to the $[0, 1]$ range by a linear transformation.

The input data from the digitizer is uniformly sampled in time. Resampling is performed to obtain a constant number of points $n_P$, that are uniformly sampled in space. This is implemented as follows: the total length of the trajectory is computed for the




(a) (b)

Fig. 2.8: Illustration of the preprocessing steps on an input symbol /ki/. (a) Raw symbol. (b) Preprocessed symbol after smoothing, size normalization and resampling. The traces of the 3 individual strokes are highlighted with separate colors.

symbol by adding the Euclidean distances between successive points. In order to find the spacing between successive points in the resampled data, the total trajectory length is divided by the number of intervals required. The points from the raw input are then replaced with a new set at this constant spacing using linear interpolation. For multi-stroke symbols, care is taken to ensure that each stroke is resampled separately in a way that the number of points is made proportional to its trajectory length.

The final result of pre-processing is a new sequence of points $\{x_i, y_i\}_{i=1}^{n_P}$ regularly spaced in arc length. A feature vector is constructed from this sequence as

$$\mathbf{x} = (x_1, x_2....x_{n_P}, y_1, y_2, .....y_{n_P}) \tag{2.2}$$

We refer to $\mathbf{x}$ as the 'concatenated $x$-$y$ coordinates' in this work. We experimented with varying number of resampled points and observed that $n_P = 60$ is quite sufficient in capturing the shape of the character including points of high curvature. Figure 2.8 illustrates the preprocessing steps on a sample of symbol கி /ki/.

### 2.5.2 Primary classifier

In this thesis, we refer to the classifier that provides a good generalization performance on data not seen during training as the 'primary classifier'. Amongst the various classifiers discussed in the literature (Sec 1.5) for online Tamil script recognition, the SVM

qualifies to be an apt choice, owing to its generalization capabilities. Accordingly, we adopt it as the primary classifier for our experiments. We employ the recognition labels and likelihoods returned by the SVM (in the following chapters of the thesis) to improve the segmentation of Tamil words and subsequently, the symbol recognition rate.

The SVM [69] is a supervised method used for two-class pattern classification problems. Suppose a training data set comprises pairs $\{(\mathbf{x}_i, l_i), 1 \leq i \leq N_{Tr}\}$, where each input vector $\mathbf{x}_i \in \Re^d$ is assigned to $l_i$. The value of $l_i$ corresponds to one of the binary labels $\{-1, +1\}$. The SVM minimizes the cost function

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} \tag{2.3}$$

subject to the constraints

$$l_i(\mathbf{x}_i.\mathbf{w} + b) \geq +1 \tag{2.4}$$

Here $\mathbf{w}$ is the weight vector and $b$ is the bias term. The above equations apply to the scenario where training samples are linearly separable. Whenever the classes to be recognized are not linearly separable, the cost function is reformulated by introducing slack variables $\xi_i \geq 0 \;\; i = 1, 2, ...N_{Tr}$. The SVM now finds $\mathbf{w}$ to minimize

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N_{Tr}} \xi_i \tag{2.5}$$

subject to

$$l_i(\mathbf{x}_i.\mathbf{w} + b) \geq +1 - \xi_i \tag{2.6}$$

The constant $C$ is a regularization parameter. When the decision function is non-linear, the above scheme cannot be used directly. For such cases, the SVM maps the training data from $\Re^d$ to a higher dimensional feature space $H$, via a mapping function $\phi : \Re^d \rightarrow H$. In this feature space $H$, the data may be linearly separable. In practice, the so-called 'kernel-trick' is used wherein, a kernel defined by $K(\mathbf{x}, \mathbf{x}_i) = \phi(\mathbf{x})\phi(\mathbf{x}_i)$ is used to construct the optimal hyperplane in $H$ without considering the mapping function $\phi(\mathbf{x})$ explicitly. For our work, we have used the Radial Basis Function (RBF) kernel defined

as

$$K(\mathbf{x}, \mathbf{x}_i) = exp(-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2) \quad \gamma \geq 0 \tag{2.7}$$

SVMs for multi-class recognition problems are realized by combining several two-class SVMs [18]. In practice, one of the two methods, namely, one-versus-one (OVO) and one-versus-all (OVA) are employed. In OVO method, for a $c$-class problem, $c(c-1)/2$ two-class SVMs are constructed. A two-class SVM $C_{ij}, i < j$ is trained using samples from classes $i$ and $j$, containing positive and negative samples, respectively. Whenever the decision function value for a test sample is positive from $C_{ij}$, the vote for class $i$ is incremented by one. Otherwise, the vote for class $j$ is increased by one. The sample is assigned to the class with the maximum number of votes. The OVA method, on the other hand, employs $c$ two-class SVMs for a $c$-class problem. The $i^{th}$ two-class SVM generates a decision boundary between class $i$ and the other $c-1$ classes. The test sample is assigned to the class having the largest value of the decision function amongst all the $c$ two-class SVMs . The concatenated $x$-$y$ features $\mathbf{x}$ (refer Eqn 2.2) are fed as input to the SVM classifier.

We have employed the LIB-SVM software [70] for learning the SVM model parameters. The OVO scheme is employed for training. The performance of the SVM classifier is largely dependent on the selection of the parameters. The samples corresponding to the 155 symbols in the IWFHR training set are employed to obtain the model parameters. RBF kernel is used in our experimentation. Recognition performance of 86% is achieved on the IWFHR test set with parameters $C = 5$ and $\gamma$=0.2. The kernel and the corresponding parameters are optimally set after performing five-fold cross validation experiments on the IWFHR training data.

## 2.6 Summary

In this chapter, an overview of the Tamil character set is provided. The methodology adopted in choosing the minimal set of symbols, from the recognition point of view, is discussed. An overview of the various datasets employed in this thesis is presented.

Finally, we outline the components of a simple online handwriting recognition system for Tamil symbols, with SVM as the classifier. The issues pertaining to the recognition of Tamil symbols is mentioned with illustrations. The material presented in this chapter provides the required background and will be referred to while discussing the novel methodologies for the research issues in the subsequent chapters.

In the following chapter, we address the problem of segmenting an online Tamil word to its individual segments/symbols by proposing a feedback strategy.

# Chapter 3

# Attention-Feedback Segmentation of online Tamil words


## Abstract

*In this chapter, we propose a lexicon-free approach to segment Tamil words into its constituent symbols. Based on a bounding box overlap criterion, the word is first segmented into stroke groups. A stroke group may at times correspond to a part of a valid symbol (over-segmentation) or a merger of valid symbols (under-segmentation). Attention on specific features serve in detecting possibly over-segmented and under-segmented stroke groups. Thereafter, feedbacks from the primary SVM classifier likelihoods and stroke-group based features are considered in regrouping the detected stroke groups to form valid symbols. Our approach (referred to as 'attention-feedback' segmentation) is tested on the MILE word database and its efficacy in segmentation and potential to improve the recognition performance of the handwriting system is demonstrated. Our results show that a segmentation accuracy as high as 99.7% at symbol level can be achieved.*

## 3.1 Review of segmentation techniques

Processing of handwritten documents, in general, considers words as basic units rather than isolated characters. In English texts, there is a well defined separation between words, but the letters within a word are not separated. This is especially evident in the case of cursive handwriting, the recognition of which has been addressed in [45, 21, 22, 71, 72, 73, 74]. In Indic scripts, the constituting words are rarely cursive in nature with the possible exception of Bangla [40, 41]. It is very uncommon for two or more symbols to be written by a single stroke. Characters in a word are written separately from each other with possible overlaps.

Word recognition can be categorized into segmentation-free and segmentation-based methods. Segmentation-free approaches [75] treat the word as a single entity and attempt to recognize it as a whole, after appropriate feature extraction. The recognition is necessarily constrained to a domain specific application by a lexicon. On the other hand, segmentation-based techniques regard a word as a collection of subunits [76, 77, 78, 79]. These methods segment the word into its constituent units, recognizes them and then builds a word level interpretation by possibly employing a lexicon. In general, a suitable set of candidate patterns are generated and concatenated to constitute the word. A classifier trained on the subunits is used to classify each of these patterns. The candidates generated can be represented by a hypothesized network, called the segmentation candidate lattice [76, 78, 79] and the optimal candidate sequence representing the word is traced using dynamic programming techniques [80, 81]. Two stage segmentation schemes have been used to segment Chinese characters in [81, 82]. Apart from recognizing candidate patterns with a classifier, contextual information forms cues in deciding the optimal character sequence in segmentation-based techniques. Geometric features extracted from segments has been used for Japanese online handwriting recognition [78, 79, 80]. The linguistic knowledge obtained from a large corpus of data has been incorporated during recognition in [77, 78]. Off-stroke features that describe segmented patterns are employed for segmenting Japanese characters [83]. Hypothetical segmentation points are generated in [77, 78, 84] using geometric features (trained with SVM classifier), which are

then incorporated into the integrated-segmentation recognition (ISR) framework. Very recently, conditional random fields have been employed for path evaluation in the candidate lattice for word recognition in [85]. A modified path evaluation criteria is proposed for Japanese text recognition in [86] .

The challenges posed with segmenting online handwritten Indic scripts have hardly been investigated. As a first step towards addressing the problem, in this work, we attempt to evolve a novel lexicon-free segmentation strategy for online Tamil words [87]. As mentioned in Sec 1.3, adoption of a lexicon-free approach necessitates that a word is segmented to its individual units prior to recognition. Among the reported techniques in literature, segmentation-based approach to recognizing online Tamil words has hardly been addressed. Bharath et al. [66] use a HMM framework for modeling the symbols and their relative positions in online Tamil words. However, their work adopts a segmentation-free approach.

Even though Tamil script is non-cursive in nature, possible overlaps occur between the individual symbols. This in turn makes the problem of segmenting words a non-trivial challenge. Apart from a preliminary attempt in Bangla [40], we have not come across any work on segmentation-based methods for recognizing words in online Indic scripts. In [40], based on the positional information of the header line, the online trace is segmented to a set of sub-strokes, which are in turn recognized and concatenated using a look up table into valid characters. However, for offline handwritten Indic words, segmentation using the water reservoir concept has been reported [88]. Recursive contour following algorithm and fuzzy-based features have been proposed in [89] and [90] respectively for segmenting offline Bangla text.

## 3.2 Proposed methodology

Given an online Tamil word, our emphasis in this work is to correctly segment it into its constituent symbols by employing a feedback-based strategy. As detailed in Sec 1.1,

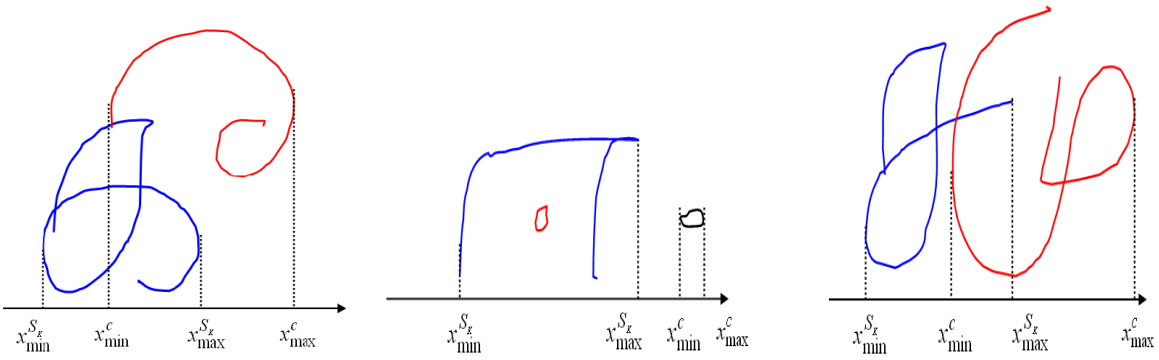
during the collection of online data, the pen-tip movement is detected with pen-up /pen-down states. The set of points captured between successive pen-down to pen-up states is called a stroke. The script being non-cursive in nature, an online word can be represented as a sequence of $n$ strokes $W = \{\widetilde{s}^1, \widetilde{s}^2 ....., \widetilde{s}^n\}$. It may be noted here that a Tamil symbol alone, at times, may correspond to a word. Typically, the strokes of a Tamil symbol vary from 1 to 5. In the case of multi-stroke Tamil symbols, strokes of the same symbol may significantly overlap in the horizontal direction. This prior knowledge is utilized to initially segment the input word as described below.

The word $W$ is segmented based on a bounding box overlap criterion, in the 'Dominant Overlap Criterion Segmentation' (DOCS) module to a set of distinct patterns, referred to as stroke groups. A stroke group is defined as a set of consecutive strokes, which is possibly a valid Tamil symbol. In order to mathematically formulate the operation in the DOCS module, one needs to quantify the degree of horizontal overlap. For the $k^{th}$ stroke group $S_k$ under consideration, its successive stroke is taken and checked for overlap, if any. Whenever the degree of overlap exceeds a threshold, the successive stroke is merged with the stroke group $S_k$. Otherwise, the successive stroke is considered to begin a new stroke group $S_{k+1}$. The algorithm proceeds till all the strokes of the word are exhausted. The first stroke, $\widetilde{s}^1$ of $W$, by default, belongs to the first stroke group $S_1$.

Let the minimum and maximum $x$-coordinates of the bounding box ($BB$) of the $i^{th}$ stroke $\widetilde{s}^i$ be denoted by $(x^i_{min}, x^i_{max})$. Given the current stroke $\widetilde{s}^c$, we define the degree of its horizontal overlap $O^c_k$ with the previous stroke group $S_k$ as

$$O^c_k = \max\left(\frac{x^{S_k}_{max} - x^c_{min}}{x^{S_k}_{max} - x^{S_k}_{min}}, \frac{x^{S_k}_{max} - x^c_{min}}{x^c_{max} - x^c_{min}}\right) \tag{3.1}$$

Here $x^{S_k}_{min}$ and $x^{S_k}_{max}$ denote the minimum and maximum $x$-coordinates of the $BB$ of the $k^{th}$ stroke group. A threshold $T_0$ (set to 0.2) applied on $O^c_k$ is used for merging strokes. As will be discussed in the later part of Sec 3.8.4, $T_0 = 0.2$ gives the maximum segmentation and recognition performance on the words in the validation set DB1. The DOCS outputs a set of $\tilde{p}$ stroke groups, where $\tilde{p} <= n$. Figures 3.1 (a)-(c) depicts the




$(a)$ $(b)$ $(c)$

Fig. 3.1: Illustrations of the parameters employed for computing the overlap $O_k^c$ in the DOCS scheme. The trace of the individual strokes are highlighted with a separate color. (a) An example of a correctly segmented symbol (b) An illustration of an over-segmented symbol /I/ (c) An example of under-segmentation.

parameters employed for computing $O_k^c$ for three different patterns.

Figures 3.2, 3.3 and 3.4 present illustrations, wherein the DOCS module combines one or more input raw strokes to generate stroke groups. The resulting stroke groups are valid Tamil symbols ழ /mu/, உ /U/ and ஈ /I/ respectively.

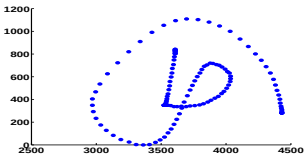
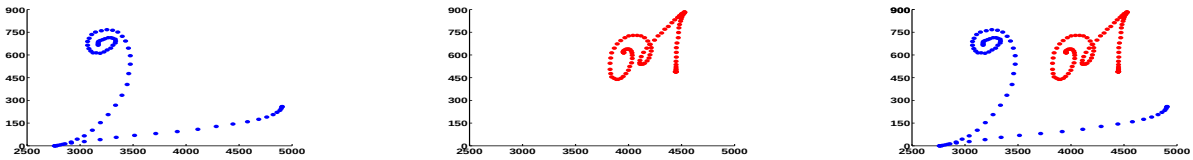



Fig. 3.2: Generation of a stroke group from a single stroke Tamil symbol /mu/.

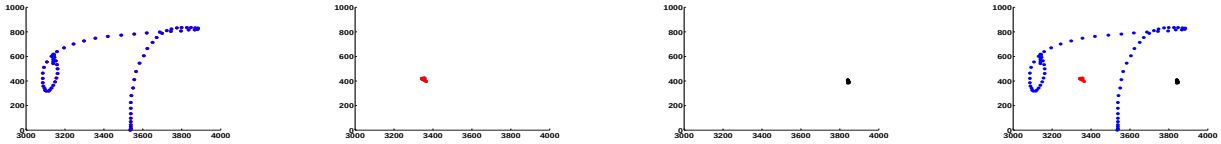
However, at times, a stroke group generated from the DOCS may correspond to a part of a valid symbol or a merger of symbols. This issue is addressed below with suitable illustrations.

- Splitting of a valid symbol (over-segmentation): The symbol *aytam* ஃ /ah/ in the word அஃது /aahtu/ (Fig. 3.5 (a)) is segmented into 3 stroke groups, as shown by the separate BBs. The DOCS outputs 5 stroke groups instead of 3. Similarly, referring to Fig. 3.1 (b), we note that the symbol ஈ /I/ gets split to 2 stroke groups.



(a) (b) (c)

Fig. 3.3: Generation of a stroke group for a two-stroke Tamil symbol /U/. (a) and (b): The 2 individual strokes. (c) Stroke group generated by DOCS. Since the second stroke (in (b)) completely overlaps with the first stroke (in (a)) in the horizontal direction, they are merged into a single stroke group (shown in (c)) by the DOCS. The resulting stroke group /U/ is a valid symbol. The traces of the individual strokes are highlighted with separate colors.



(a) (b) (c) (d)

Fig. 3.4: Generation of a stroke group for a three-stroke Tamil symbol /I/. (a),(b) and (c): The three individual strokes. (d) Generated stroke group. Since the second and third strokes (presented in (b) and (c)) completely overlap in the horizontal direction with the first stroke (in (a)), the DOCS module combines the 3 strokes to generate a single stroke group (shown in (d)). The resulting stroke group /I/ is a valid symbol. The traces of the individual strokes are highlighted with separate colors.

- Merging of two distinct symbols (under-segmentation): In Fig. 3.5 (b), the symbols த் /t/ and தி /ti/ of the word சமுத்திரம் /camuttiram/ merge to a single stroke group த்தி /tti/, as highlighted by a single BB. In this case, DOCS outputs 5 stroke groups instead of 6. Similarly, the patterns சி /ca/ and மு /mu/ in Fig 3.1 (c) are valid Tamil symbols that get merged to a single stroke group.

(a) (b)

Fig. 3.5: Illustration of over-segmented and under-segmented words after the DOCS step. (a) The *aytam* /ah/ gets fragmented (over-segmented) to 3 stroke groups as shown by the separate bounding boxes. (b) The /t/ and /ti/ symbols get merged (under-segmented) to one stroke in this word.

In this work, we aim to further improve the segmentation performance beyond that given by the DOCS. Different sets of attributes have been separately derived to detect under-segmented and over-segmented stroke groups respectively. 'Attention' on these features selects only a subset of the generated stroke groups for subsequent analysis. Upon detection, a stroke group suspected to be incorrectly segmented is fed to a module, that operates on additional attributes (derived from the statistics of the IWFHR database), to provide 'feedback' on whether or not to proceed in correcting it. Whenever the feedback favors a correction, rearrangement of the strokes within or even outside the stroke group under consideration is initiated. It is to be noted that only stroke groups suspected to be broken or under-segmented are fed to the feedback module. In other words, we concentrate on the rectification of possible segmentation errors on selected stroke groups. First, we operate on stroke groups likely to contribute to under-segmentation errors, and split them, if necessary. Thereafter, stroke groups suspected to be a part of valid symbol (contributing to over-segmentation errors) are merged with their appropriate neighbors to generate valid symbols. In this paper, we refer to our proposed segmentation technique by 'attention-feedback segmentation' (abbreviated as AFS). Figure 3.6 presents a pictorial representation summarizing the AFS approach for a stroke group generated in the DOCS module.

Summarizing, the stroke groups resulting from the DOCS are regarded as tentative candidates for valid Tamil symbols. Based on feedback from various attributes proposed in this work, the AFS module may modify the number of stroke groups output by the DOCS module. In doing so, the AFS improves the robustness of the handwriting system. For the illustrations அஃது `/aahtu/` and சமுத்திரம் `/camuttiram/` in Fig. 3.5, the refined segmentation (performed by the AFS module), when successful, should output 3 and 6 stroke groups respectively. Similarly, for the patterns in Fig. 3.1 (b) and (c), we expect 1 and 2 stroke groups from the AFS respectively.

The stroke groups resulting from the AFS module are considered as valid patterns/ symbols for the given word $W$. We assume that the word $W$ after the AFS step comprises

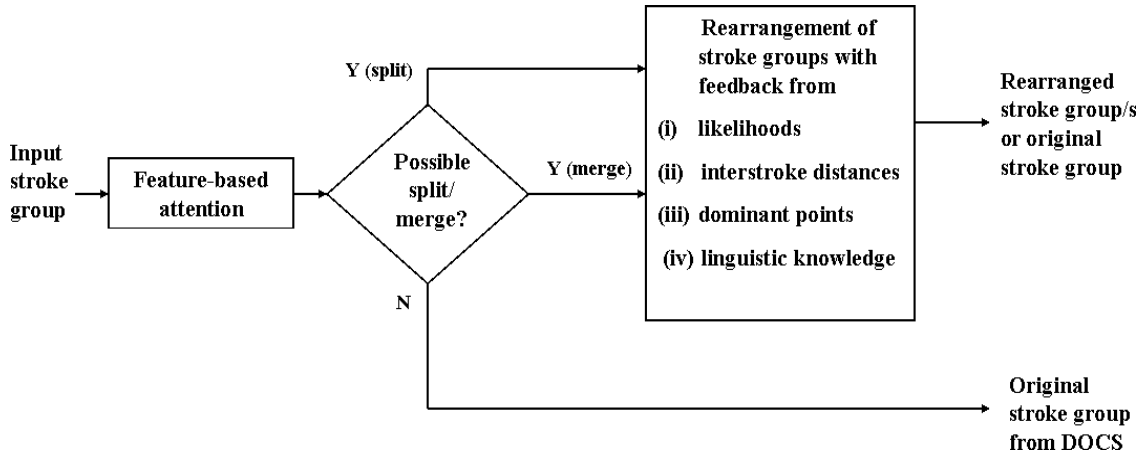



Fig. 3.6: Pictorial overview of the proposed attention-feedback segmentation approach for a stroke group output by the DOCS module.

$p$ stroke groups.

## 3.3 Comparison of the proposed methodology with the Integrated Segmentation Recognition (ISR) scheme

In order to judge the contributions of the current work, we highlight the two important differences between the proposed segmentation strategy with the integrated-segmentation and recognition approach (ISR) typically followed in recent literature for online non-Indic scripts.

- The stroke groups in DOCS step may be regarded to be analogous to the primitive segments in the pre-segmentation strategy adopted in works such as [78, 80]. In the over segmentation step, the input string pattern is over-segmented into primitive segments such that each segment composes a single character or a part of a character. For Chinese and Japanese scripts, strokes of different characters overlap less frequently [91], due to which, under-segmentation errors hardly arise. On the other hand, for Tamil, we are likely to encounter a high degree of overlapping of

strokes of different symbols in the DOCS step. Thus, there arises a need to rectify such under-segmentation errors, by appropriately splitting stroke groups to valid symbols.

- In the path-evaluation step adopted in Japanese and Chinese scripts, the optimal path across all possible segmentation paths in the candidate lattice are evaluated with dynamic programming. Each segmentation path represents a set of candidate patterns, generated by combining successive primitive segments obtained from the pre-segmentation step. Unlike the ISR strategy, the AFS approach concentrates on the rectification of selected stroke groups, detected to contribute to segmentation errors. In the case of Tamil words, since we need to rectify both under-segmentation and over-segmentation errors, generation of a segmentation lattice, outlining all possible segmentation paths is not feasible. In summary, the ISR operates across all sets of possible segmentation paths to obtain the optimal one with dynamic programming. In contrast to this, the AFS step selects, using feature based attention, only stroke groups suspected to be wrongly segmented and tries correcting them to valid symbols, without adopting dynamic programming techniques.

For justifying the proposed term 'attention-feedback', we present an analogy to concepts in the area of neuroscience. Studies on visual perception in primates demonstrate the effect of attention on the response of the visual neurons. Feature based attention [92] biases the neuronal responses as though the attended stimulus was presented alone. Also, shifting spatial attention from outside to the inside of the receptive field increases the neuronal responses. Further, studies on visual pathways [93] show extensive feedback from the cortex to the lateral geniculate nucleus (LGN), which have both inhibitory and facilitatory effects on the responses of LGN relay cells. As mentioned in the previous section, in the proposed work, we incorporate local feature based attention to correct and improve segmentation. In addition, feedback based on features as well as the classifier likelihoods are employed to rectify any incorrect segmentations by regrouping the strokes.

In the subsequent sections, we outline the proposed attention feedback strategies (AFS module), the primary focus of this chapter. In this context, the following aspects

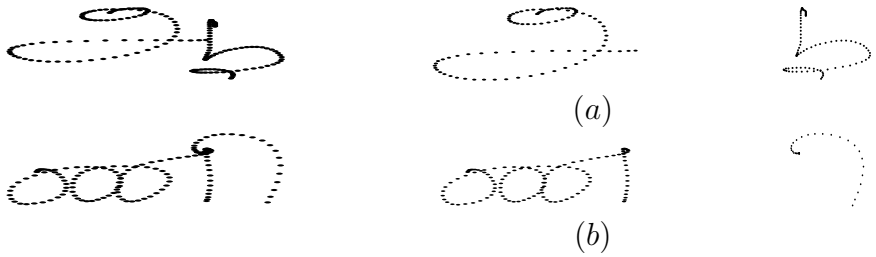



Fig. 3.7: Illustration of two samples from the IWFHR database over-segmented by DOCS. (a) Sample of /A/ broken to 2 stroke groups. (b) Sample of /nni/ broken to 2 stroke groups.

need to be borne in mind.

- Prior to sending a suspected split or under-segmented stroke group to the SVM classifier for generating the recognition label and likelihoods, we subject it to the preprocessing steps of smoothing, size normalization and resampling discussed in subsection 2.5.1.

- Moreover, since the emphasis here is on improving the segmentation rather than the classifier performance, $x$-$y$ coordinates of the preprocessed stroke group alone are used as features. Hereafter, for the $k^{th}$ stroke group $S_k$, we refer to its concatenated $x$-$y$ coordinates as $\mathbf{x}^{S_k}$.

## 3.4 Detection of over-segmented stroke groups with feature-based attention

The training samples of symbols in the IWFHR dataset are segmented based on the overlap criterion (DOCS). Since this dataset consists of isolated Tamil symbols, the segmentation of any sample into more than one stroke group indicates an over-segmentation. Figures 3.7 (a) and (b) respectively illustrates a sample of அ /A/ and ஸ்ஸி /nni/ that get over-segmented into more than one stroke group by DOCS step.

We explore the utility of two features namely, number of dominant points and dots, to detect possible over-segmentations in stroke groups.

- **Number of Dominant Points**: The number of dominant points of a stroke group provides a rich structural description [60]. We propose a modified strategy for generating the dominant points for a given stroke group. Our algorithm begins by marking the first pen position as a dominant point. Starting from the current dominant point, we compute the absolute value of the angle between pen directions at successive points and accumulate it along the online trace as long as the cumulative sum is less than a threshold $T_\theta$. The pen position, at which the accumulated angle exceeds $T_\theta$, is marked as the next dominant point and the process continues till the end of the trace. The resulting number of dominant points extracted is used as a feature for attention. We empirically choose threshold $T_\theta$ in order to ensure that the shape of the stroke group is approximated with a reduced set of points, without losing any points of high curvature. Very high values of $T_\theta$ do not sufficiently capture the shape of the stroke group. On the other hand, for low values of $T_\theta$, the number of dominant points increase with the approximated shape resembling more closer to the original stroke group. We observe that a value of $T_\theta$ in the range $[35^o, 55^o]$ works well for shape representation. In the present work, we choose $T_\theta = 45^o$. Figure 3.8 highlights the 20 dominant points for the stroke group ஆ /A/. The dominant points are extracted from the preprocessed stroke group (refer Sec 2.5.1).

  We now present a statistical justification towards using the number of dominant points of a stroke group as a cue to detect possible over-segmentation errors. Let us assume that a training sample $X$ from the IWFHR data-set gets split by the DOCS into $\tilde{p}$ stroke groups. The number of dominant points corresponding to each of the stroke groups is computed and denoted by $\{N^{s_1}, N^{s_2}...N^{s_{\tilde{p}}}\}$. We make a reasonable assumption that shorter stroke groups are more indicative of a broken symbol, compared to longer ones. Accordingly, for every sample $X$, we consider the number of dominant points ($\min_i N^{s_i}$) corresponding to the shortest stroke group in the split. The distribution of the number of dominant points of the shortest stroke group for all the training samples of symbols (in the IWFHR dataset) split

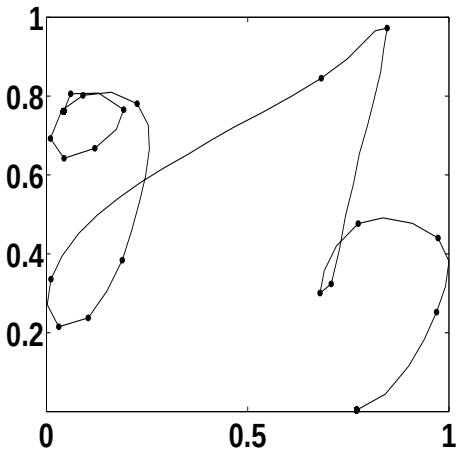



Fig. 3.8: Representation of the 20 dominant points (marked by dots) for /A/ vowel.

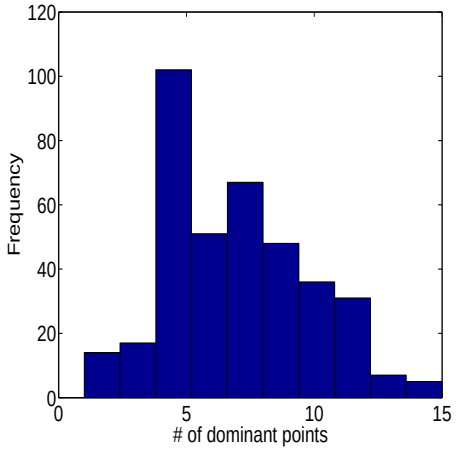


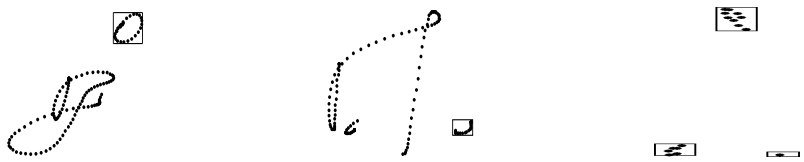

Fig. 3.9: Distribution of the number of dominant points across the shorter stroke groups of the over segmented symbols in the IWFHR dataset.

by DOCS is presented in Fig. 3.9. We observe that a stroke group for which the number of dominant points is less than 16 may correspond to a part of a Tamil symbol. This statistical rule in turn implies that symbols such as ∟ /Ta/, ∐ /pa/ and ∐◌ /ma/ that generally comprise less than 16 dominant points are suspected to be broken and sent for possible correction in AFS module.

- **Dot feature**: As discussed in Chapter 2, the inherent vowel sound in a base consonant is suppressed by placing a dot on it and is referred to as a pure consonant. In addition, dots appear as a part of the vowel ◌ /I/ and symbol ◌ /ah/. On the

IWFHR training set, we observed that the dots at times get separated out as a stroke group with the DOCS step, leading to over-segmentation (Fig. 3.10).

Though simple cues like bounding box area may serve as a sufficient feature for



(a) (b) (c)

Fig. 3.10: Illustration of dots in (a) pure consonants and (b) `/I/` vowel getting separated out as a stroke group with the DOCS step. (c) The dots in `/ah/` get fragmented into 3 stroke groups. The dot stroke groups are highlighted with a box.

detecting dots in printed text, the same do not generalize for handwritten words. This is largely due to the variability in the size of dots encountered with different writing styles. At times, it is quite possible for small strokes such as the vowel modifier of ஐ `/i/` to be regarded as dots ( for an illustration, refer Fig. 2.7 (a)). A raw stroke group $S_k$ is detected as a dot if it satisfies any of the following spatial constraints.

1. The height of its BB is less than the overall minimum height ($\widetilde{h}_{min}^{BB}$) of the BB of the Tamil symbols obtained from the study. In other words,

$$(y_{max}^{S_k} - y_{min}^{S_k}) < \widetilde{h}_{min}^{BB} \tag{3.2}$$

Let $N_{Tr}^{\omega_i}$ represent the number of training samples for the symbol $\omega_i$ in the IWFHR dataset. In order to compute $\widetilde{h}_{min}^{BB}$, the minimum BB height (denoted


Fig. 3.11: Detection of stroke groups appearing as dots. The stroke group highlighted in a box is located above the middle line of the word, indicating that it is very likely to be a dot.

by $\widetilde{h}_i$) over the $N_{Tr}^{\omega_i}$ samples of a symbol $\omega_i$ is first calculated. We then assign $\widetilde{h}_{min}^{BB}$ to the overall minimum BB height computed over $\{\widetilde{h}_i\}_{i=1}^{155}$. For the IWFHR dataset, we obtain $\widetilde{h}_{min}^{BB} = 200$.

2. Its BB is located spatially above the middle line of the word (Fig. 3.11) Mathematically, we need to ensure

$$y_{min}^{S_k} > \frac{\sum_{k=1}^{\tilde{p}} \mu_y^{S_k}}{\tilde{p}} \tag{3.3}$$

where $\mu_y^{S_k}$ represents the $y$-centroid for $S_k$ and $\tilde{p}$ is the number of stroke groups in the word $W$.

## 3.5 Detection of under-segmented stroke groups with feature based attention

Attention on spatial based features serves in detecting possible under-segmented errors in a stroke group. We now describe the details of two such features.

- **Inter-stroke features**: For preprocessed stroke groups comprising $m$ strokes ($m > 1$)

  1. The horizontal displacement $b_i$ from the bounding box $x$-maximum of the $i^{th}$ stroke to the first point of the $(i+1)^{th}$ stroke is computed. The maximum of the computed displacements $b_{max}$, among all stroke pairs, is a feature for attention.

$$b_{max} = \max_i \quad b_i \qquad i = 1, 2, ...m-1 \tag{3.4}$$

  We interpret $b_{max}$ as the maximum 'bounding box to stroke displacement' in a stroke group.

  2. The signed vertical inter stroke gap $h_i$ between last point of the $i^{th}$ stroke and the first point of the $(i+1)^{th}$ stroke is noted. The minimum of the

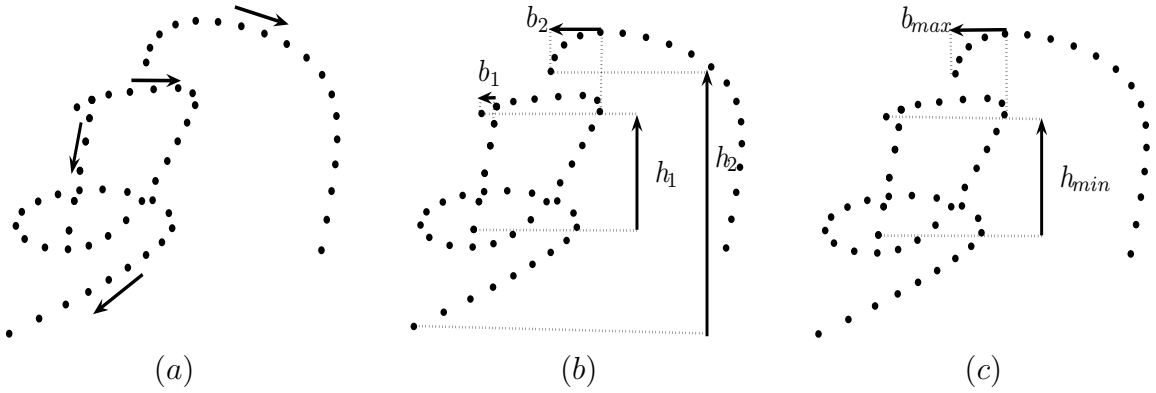



Fig. 3.12: Representation of inter-stroke features for /ti/ symbol. (a) Stroke group /ti/ with direction of trace marked with arrows. It comprises 3 strokes. (b) Illustration of the four inter-stroke measurements $b_1$, $h_1$, $b_2$, $h_2$. (c) Illustration of $b_{max}$ and $h_{min}$. Note that for this stroke group $b_{max} < 0$ and $h_{min} > 0$. Attention on inter-stroke features $b_{max}$, $h_{min}$ indicate that the stroke group is correctly segmented with DOCS.

heights measured across successive pairs of strokes, $h_{min}$ is another feature for attention.

$$h_{min} = \max_i \quad h_i \qquad i = 1, 2, ...m - 1 \tag{3.5}$$

The inter-stroke features may be either positive or negative, depending on the relative positions of the strokes under consideration. For the stroke group தி /ti/ (Fig. 3.12), written in 3 strokes, $b_{max} < 0$ and $h_{min} > 0$. We now demonstrate the efficacy of these features in detecting under-segmented stroke groups. An analysis is performed on stroke groups (comprising multiple strokes) obtained from DOCS on the 250 handwritten words in data-set DB1.

1. Stroke groups for which $b_{max} > 0$ may correspond to Tamil symbols that have been merged. On the other hand, stroke groups satisfying $b_{max} < 0$ rarely produce an under segmentation error. The value of $b_{max}$ is positive when two valid Tamil symbols are merged in a stroke group unlike the case of the inter-stroke displacement in a correctly segmented stroke group. Hence, this feature serves as a cue to detect under-segmented stroke-groups. For the database DB1, as high as 95% of stroke groups contributing to under-segmentation errors satisfy $b_{max} > 0$. Figure 3.13 (a) depicts the case wherein 2 Tamil symbols ை ($VM$ of /ai/) and ர /ra/ are merged

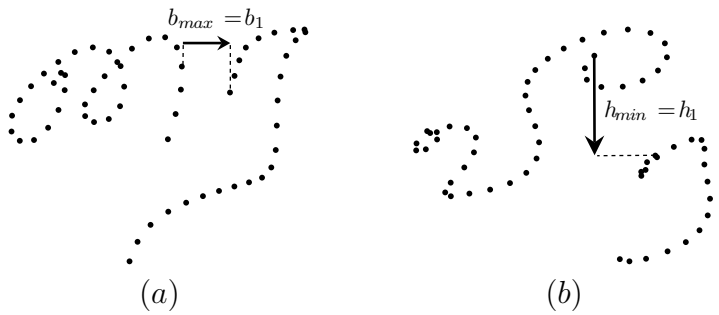



Fig. 3.13: Distinct symbols wrongly merged by DOCS. The stroke groups presented in (a) and (b) satisfy $b_{max} > 0$ and $h_{min} < 0$, respectively.

to a stroke group ரை /rai/. This stroke represents a pattern, that the SVM has not come across. Therefore, it is quite likely for the SVM primary classifier to regard this stroke group as an outlier pattern by providing a low likelihood to its most probable candidate symbol.

2. Stroke groups for which $h_{min} < 0$ can be an invalid symbol pattern for the SVM as depicted in Fig. 3.13 (b). Here, the 2 Tamil symbols வி /vI/ and ர /ra/ are merged to a stroke group வீர /vIra/. This is not a valid stroke group encountered by the SVM and therefore, a very likely outlier.

On the other hand, Fig. 3.12 presents a correctly segmented sample of தி /ti/ satisfying $b_{max} < 0$ and $h_{min} > 0$.

## 3.6 AFS strategy for over-segmented stroke groups

As justified in Sec 3.4, a stroke group with less than 16 dominant points may correspond to a part of a Tamil symbol. In general, it is observed that the stroke groups appearing as dots have less than 16 dominant points. Thus, the presence of such stroke groups, from a linguistic viewpoint, provide additional cues and insights that can well be utilized to resolve the over-segmentation problem. This is discussed in sufficient detail in subsection 3.6.2.

We now provide a generalized framework to resolve over-segmentations in any stroke group comprising less than 16 dominant points (including those detected as dots).

### 3.6.1 Generalized framework

Figure 3.14 presents the block diagram of the AFS strategy proposed for correcting over-segmented stroke groups. Let $S_k$ correspond to a stroke group that is likely to be a

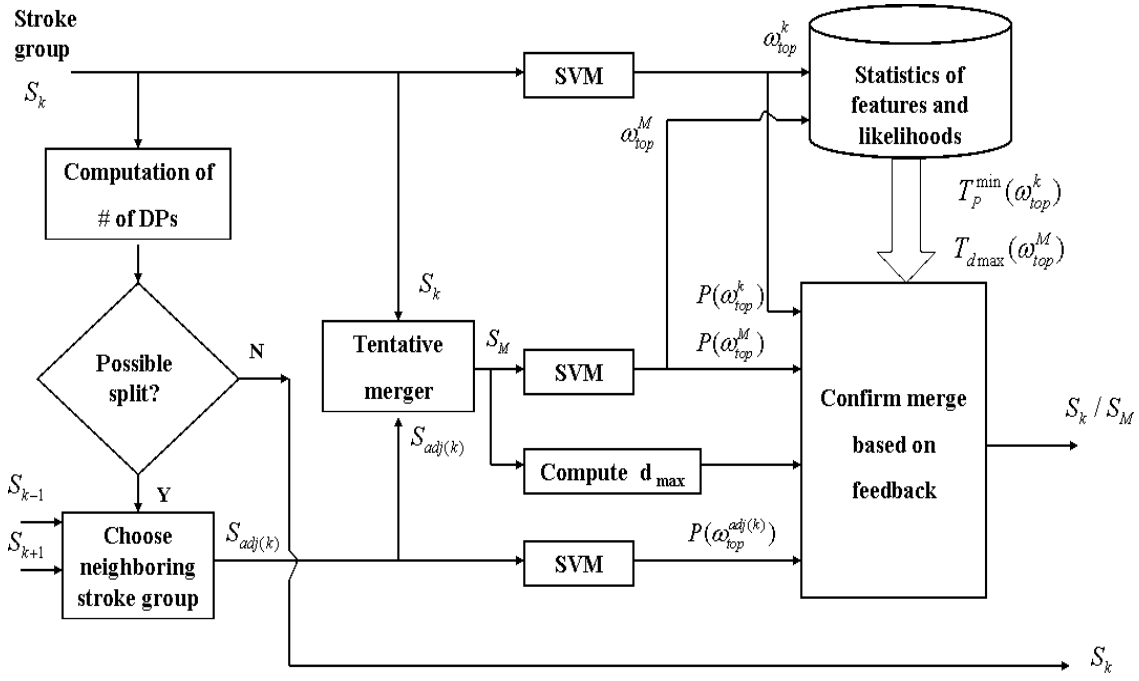



Fig. 3.14: AFS module for resolving over-segmented stroke groups.

broken symbol. Consider $S_{adj(k)}$ to be the neighboring stroke group whose BB is closest to that of $S_k$. The feature vector (concatenated $x$-$y$ coordinates) of the preprocessed $S_k$ and $S_{adj(k)}$ are separately sent to the SVM classifier. Let the likelihoods $P(\omega_{top}^k)$ and $P(\omega_{top}^{adj(k)})$ correspond to the most probable symbols $\omega_{top}^k$ and $\omega_{top}^{adj(k)}$ respectively. The stroke groups are merged to a valid symbol whenever one of the conditions outlined below are satisfied.

1. The stroke groups $S_k$ and $S_{adj(k)}$ are merged whenever, $P(\omega_{top}^k) < T_P^{min}(\omega_{top}^k)$. Here, $T_P^{min}(\omega_{top}^k)$ represents the minimum likelihood value returned by the SVM for all the correctly classified samples of the symbols $\omega_{top}^k$ in the IWFHR competition test set.

2. Let $S_M$ represent the stroke group obtained by merging $S_k$ with $S_{adj(k)}$. For a

possible merge, we require the average likelihood of the most probable symbols $\omega_{top}^{k}$ and $\omega_{top}^{adj(k)}$ to be less than the likelihood $P(\omega_{top}^{M})$ for $S_M$. However, for avoiding any unintentional merges, we additionally ensure that the maximum horizontal inter-stroke gap (denoted by $d_{max}$) in $S_M$ is less than the maximum possible horizontal gap $T_{dmax}(\omega_{top}^{M})$ determined from the IWFHR dataset for the recognized symbol $\omega_{top}^{M}$. In other words,

$$\frac{P(\omega_{top}^{k}) + P(\omega_{top}^{adj(k)})}{2} < P(\omega_{top}^{M})$$
$$d_{max}^{S_M} < T_{dmax}(\omega_{top}^{M}) \tag{3.6}$$

The maximum horizontal inter-stroke gap $d_{max}$ is computed as follows: For a pre-processed stroke group comprising $m$ strokes, the signed horizontal inter stroke gap $d_i$ between the last point of the $i^{th}$ stroke and the first point of the $(i+1)^{th}$ stroke is measured. The maximum of the inter-stroke gaps represents $d_{max}$.

$$d_{max} = \max_{i} \quad d_i \qquad i = 1, 2, ...m-1 \tag{3.7}$$

Contrast to $b_{max}$, the inter-stroke gap $d_{max}$ is regarded as the maximum 'stroke to stroke displacement' in a stroke group.

3. Apriori knowledge can also be employed for correcting errors in CV combinations of vowel இ /i/. Assume that the stroke group $S_k$ is the vowel modifier ி. We check if $\omega_{top}^{k}$ corresponds to any of the symbols that frequently get assigned to the pattern of ி. In other words, when $\omega_{top}^{k}$ is either ர /ra/ , ா ($VM$ of /A/) or ே ($VM$ for /e/), we merge $S_k$ to its preceding stroke group $S_{k-1}$ after ensuring that (1) $\omega_{top}^{k-1}$ is a base consonant and (2) $\omega_{top}^{M}$ is a CV combination of இ /i/ or ஈ /I/ vowel.

Figures 3.15 and 3.17 present suitable illustrations wherein symbols suspected to be broken by the DOCS get corrected by the AFS module. The second stroke group in the word of Fig. 3.15 has been properly merged to a valid symbol ங் /ng/. The low likelihoods

of second and third stroke groups from the SVM suggests us that they get merged. The correctly segmented word பூங்கா /pUngkA/ after the merge is shown in Fig. 3.15(e).

As an illustration to how the inter-stroke gap $d_{max}$ aids in preventing spurious

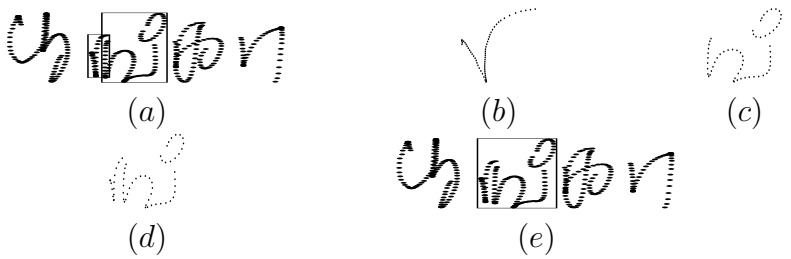


Fig. 3.15: An example of AFS for resolving over-segmentation error in broken symbols. (a) A word over-segmented by DOCS. (b) The second stroke group in this word has 8 dominant points and is assumed to be a part of a valid symbol. This stroke group has a low posterior probability. (c) The second split part of the symbol also has low posterior probability. (d) Merged symbol has higher likelihood. (e) The correctly segmented word after the merge.

merges, we consider the last stroke group ா ($VM$ of /A/) that has 5 dominant points. The number of dominant points being less than 16, we tentatively merge it to the neighboring stroke group க /ka/ and recognize the resulting pattern $S_M$ (Fig. 3.16 (a)). The SVM favors the symbol தா /tU/ (the printed sample of which is shown in Fig 3.16 (b)). However, we observe that the maximum possible inter-stroke distance for தா /tU/ is less than the $d_{max}$ computed for $S_M$. Accordingly, since Eqn 3.6 is violated, we do not consider the merge. Instead, the individual stroke groups க /ka/ and ா ($VM$ of /A/) are favored.

For correcting the over segmentation error of the word in Fig. 3.17, knowledge based prior information is utilized for merging the stroke group ி ($VM$ of /i/) with ன /Na/ to generate னி /Ni/.

Summarizing, we consider the feedback from the statistics of inter-stroke features and SVM likelihoods to perform the merge (Fig. 3.14).

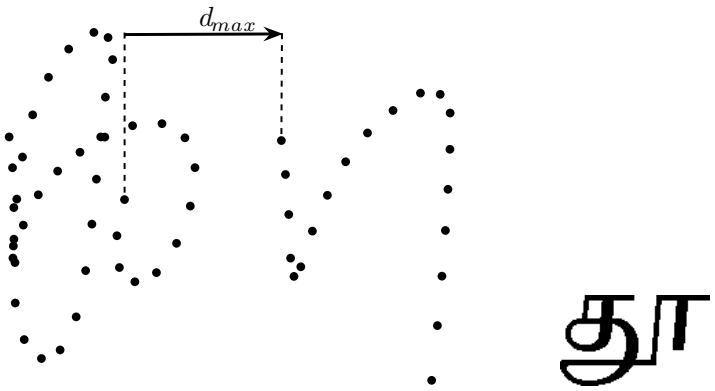



Fig. 3.16: (a) Computation of $d_{max}$ for the combined stroke group $S_M$. The SVM favors /tU/ as the most favorable symbol. (b) Printed sample of /tU/. The maximum possible inter-stroke distance for the symbol /tU/ is less than the $d_{max}$ computed for $S_M$.

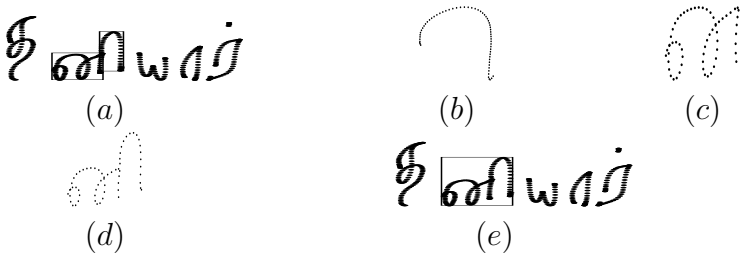



Fig. 3.17: Another example of AFS for resolving over-segmentation error in broken symbols. (a) A word over-segmented by DOCS. (b) The third stroke group has 4 dominant points and is assumed to be a part of a valid symbol. This stroke group is recognized as /ra/ by the SVM. (c) The preceding stroke group is recognized as /Na/, a base consonant. (d) The merged symbol is recognized as /Ni/, a CV combination of /i/ vowel. (e) Correctly segmented word after the merge.

### 3.6.2 Resolving over-segmentations in stroke groups appearing as dots

As mentioned earlier, for stroke groups appearing as dots from the DOCS, we can utilize apriori contextual information for robustly correcting them. Linguistic knowledge is incorporated in resolving over-segmentation errors arising in pure consonants, the vowel ி /I/ and symbol ஃ /ah/. We consider the methodology described herein as alternatives to the generalization approach described in the previous subsection.

### Handling of dots in pure consonants

It is to be noted that the dot of a pure consonant gets segmented as a separate stroke group, only if its horizontal overlap with the base consonant is very small, which happens occasionally (refer Fig 3.10 (a)). Thus if a stroke group $S_k$ is detected as a dot, there is a very high probability for the preceding stroke group $S_{k-1}$ to be a valid consonant. The base consonant provides the required contextual cue for the presence of the dot. The preprocessed $x$-$y$ coordinates of the preceding stroke group $S_{k-1}$ are fed to the SVM. If the most probable output $\omega_{top}^{k-1}$ is a base consonant, the dot is merged to $S_{k-1}$ , provided they satisfy the following constraint.

$$\frac{y_{max}^{S_{k-1}} - y_{min}^{S_k}}{y_{max}^{S_k} - y_{min}^{S_k}} < T_o^p(\omega_{top}) \tag{3.8}$$

This condition avoids undesirable merges of other symbols to the previous consonant. Once the dot is merged to the base consonant, the vowel is suppressed and we get a pure consonant. Ideally, there is no vertical overlap between the BBs of the dot and the base consonant. However, due to writing variations in the case of pure consonants, there arises some degree of overlap that needs to be accounted for in the AFS module, in order to ensure merging of such dots. Given a pure consonant of $\omega_{top}^{k-1}$, the maximum possible degree of $y$-overlap of the dot to the corresponding base consonant (denoted as $T_o^p(\omega_{top}^{k-1})$) is read from the statistics obtained from the IWFHR dataset. For merging the raw stroke group $S_k$ with $S_{k-1}$, the vertical overlap of the suspected dot stroke with the stroke group $S_{k-1}$ must be less than the maximum threshold $T_o^p(\omega_{top}^{k-1})$ set for the pure consonant of $\omega_{top}^{k-1}$ (Eqn 3.8). Figure 3.18 illustrates the parameters employed in computing the overlap in the pure consonant ட் /T/.

Figure 3.19 presents an illustration for the proposed AFS approach. The dot stroke is merged to its previous stroke group, recognized as a base consonant ட /Ta/ by the SVM. The correctly segmented word கைதட்டு /kaitaTTu/ is shown in Fig. 3.19 (c).

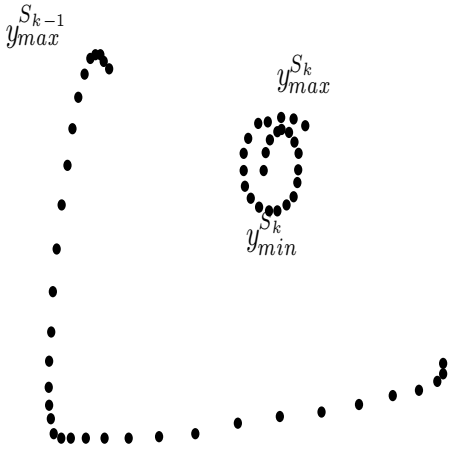



Fig. 3.18: Parameters employed for computing the degree of vertical overlap between the dot and the base consonant for the pure consonant /T/.

(a) (b) (c)

Fig. 3.19: Illustration of AFS for resolving over-segmentation error in pure consonants. (a) The /T/ symbol in the word /kaitaTTu/ is segmented to 2 stroke groups (shown by the 2 BBs). One of them is suspected to be a dot. (b) The most probable symbol for the stroke group preceding the dot is a valid consonant /Ta/. Consequently we merge the dot to this stroke group. (c) The correctly segmented word after the merge.

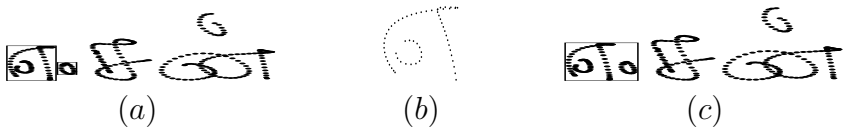
### Handling of dots in /I/ vowel

The application of DOCS step to the samples of /I/ over-segments them to the pattern and dot respectively, as shown in Figures 3.10 (b) and 3.20 (a). Given that $S_k$ is detected as a dot, we employ the apriori knowledge of $S_{k-1}$, as given below, to correct the segmentation error:

**C1** Number of strokes in $S_{k-1}$ is greater than 1.

**C2** Let $S_{k-1}$ comprise $m$ strokes. We require the BB of the $m^{th}$ stroke to be completely enclosed by the BB of the remaining strokes.

**C3** The SVM outputs $\omega_{top}^{k-1}$ as one of /I/, /e/, /E/, /ra/ or ($VM$ of /A/). Here, $\omega_{top}^{k-1}$ denotes the most probable symbol for $S_{k-1}$.



(a) (b) (c)

Fig. 3.20: Illustration of AFS for resolving over-segmentation error in /I/ vowel. (a) The /I/ vowel is segmented to 2 stroke groups shown by the 2 BBs. One of the stroke groups is detected as a dot. (b) The stroke group preceding the dot satisfies the constraints **C1-C3**. The most probable symbol for this stroke group from the SVM is the vowel /e/. Consequently we merge the dot to this stroke group. (c) The correctly segmented word after the merge.

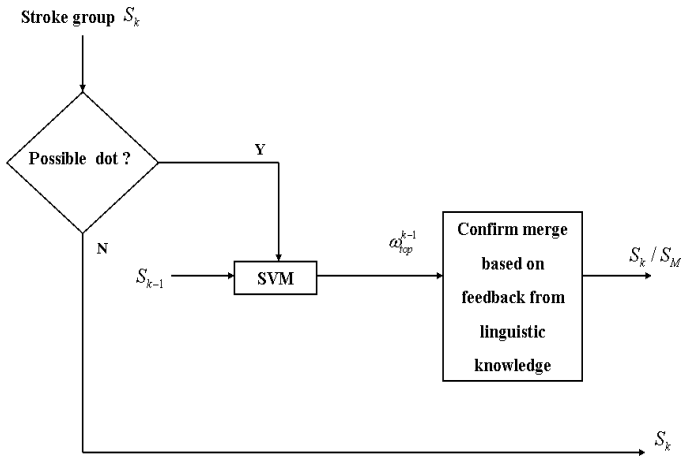



Fig. 3.21: AFS module for resolving over-segmented stroke groups appearing as dots in pure consonants and /I/ vowel.

For a valid merge, the above constraints need to be satisfied for $S_{k-1}$ (Fig. 3.20 (b)). Figure 3.21 presents a pictorial representation summarizing the proposed methodology adopted for correcting the over-segmented stroke groups in pure consonants and /I/ vowel. In particular, we rely on the feedback from attributes of the preceding stroke group to aid our decision.

### Handling of dots in /ah/ symbol

The *aytam* symbol /ah/ in Tamil comprises at least 3 strokes that appear as dots. For a majority of the samples in the IWFHR database, DOCS fragments this symbol to 3 stroke groups (refer Fig. 3.22). To detect /ah/, we focus our attention on sets of consecutive raw stroke groups $S_{k-1}$, $S_k$ and $S_{k+1}$ satisfying the spatial structure defined

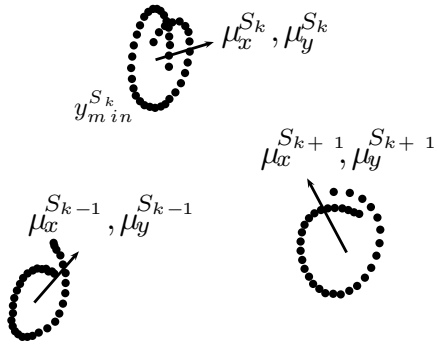



Fig. 3.22: Parameters employed for detecting symbol /ah/ appearing as 3 stroke groups.

below

$$(y_{min}^{S_k} > \mu_y^{S_{k-1}}) \& (y_{min}^{S_k} > \mu_y^{S_{k+1}}) \& (\mu_x^{S_k} > \mu_x^{S_{k-1}}) \& (\mu_x^{S_{k+1}} > \mu_x^{S_k}) \tag{3.9}$$

$\mu_x^{S_k}$ and $\mu_y^{S_k}$ represent the $x$ and $y$ centroid for the stroke group $S_k$. The individual stroke groups in a set are then preprocessed and recognized to generate 3 confidence likelihoods.

$$P(\omega_{top}^j) = \max_i \quad P(\omega_i | \mathbf{x}^{S_j}) \qquad j = k-1, k, k+1 \tag{3.10}$$

Here $\mathbf{x}^{S_j}$ denotes the preprocessed $x$-$y$ features for the stroke group $S_j$. We generate a new stroke group $S_M$ by combining the raw data of the 3 consecutive stroke groups and evaluate the confidence of it being the symbol ಅ /ah/ after preprocessing. The decision to combine the 3 stroke groups and favor the symbol ಅ /ah/ can be formulated as

$$\text{Choose symbol ಅ /ah/ when } P(\omega^M = symbol \ \text{ಅ}) > \frac{\sum P(\omega_{top}^j)}{3}$$

$P(\omega^M = symbol \ \text{ಅ})$ represents the likelihood of ಅ /ah/, returned by the primary SVM classifier for stroke group $S_M$. The proposed methodology is summarized in the block diagram presented in Fig. 3.23.

Figure 3.24 illustrates a word, in which the symbol ಅ /ah/ fragmented into 3 stroke groups by the DOCS get corrected with the proposed AFS module. The likelihoods of the most probable symbols for the stroke groups in Fig. 3.24 (b)-(d) are 0.02, 0.05, 0.03 respectively. The confidence of ಅ /ah/ for the combined stroke group in Fig. 3.24 (e) is 0.3. Accordingly, based on feedback from SVM likelihoods, we merge the 3 stroke groups as shown in Fig. 3.24 (f).

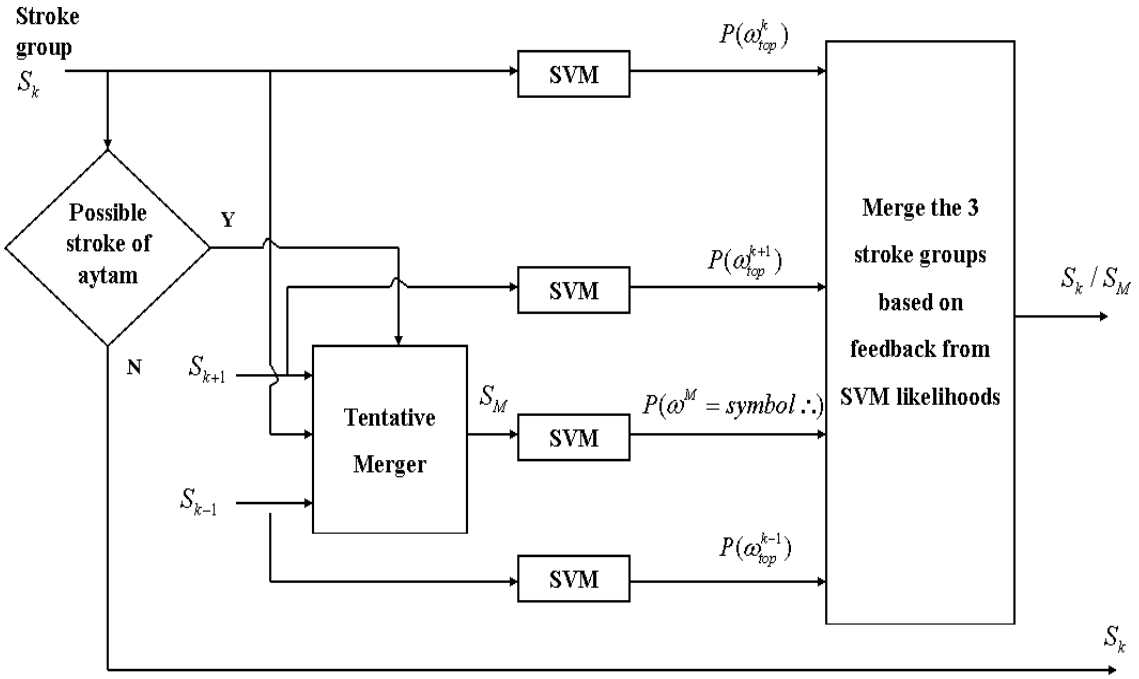



Fig. 3.23: AFS module for handling over-segmentation in /ah/ symbol.

## 3.7 AFS of under-segmented stroke groups

As justified in Sec 3.5, a stroke group satisfying $b_{max} > 0$ or $h_{min} < 0$ may correspond to a merger of valid Tamil symbols. In this section, we outline the proposed AFS strategy for resolving such under-segmented stroke groups. From the block diagram of Fig. 3.25, we observe that feedbacks of SVM likelihoods, statistics of number of dominant points and inter-stroke distance $d_{max}$ (defined in Eqn 3.7) influence our decision to split a stroke group.

Assume that $S_k$, comprising $m$ strokes, satisfies $b_{max} > 0$. If $b_{max}$ corresponds to the inter stroke displacement between $q^{th}$ and $(q+1)^{th}$ strokes, then we regard stroke group $S_k$ as the merger of two valid symbols $S_{k_1}$ and $S_{k_2}$, defined by $S_{k_1} = \{\widetilde{s}_k^1, \widetilde{s}_k^2, .....\widetilde{s}_k^q\}$ and $S_{k_2} = \{\widetilde{s}_k^{q+1}, \widetilde{s}_k^{q+2}, .....\widetilde{s}_k^m\}$. Here $\widetilde{s}_k^i$ denotes the $i^{th}$ stroke for stroke group $S^k$. $S_{k_1}$ and $S_{k_2}$ are in turn preprocessed and subsequently recognized to generate confidence likelihoods

$$P(\omega_{top}^{k_j}) = \max_i \quad P(\omega_i|\mathbf{x}^{S_{k_j}}) \qquad j = 1, 2 \tag{3.11}$$

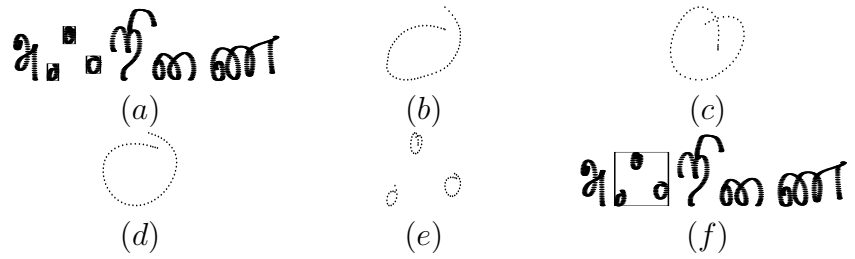


Fig. 3.24: Illustration of AFS for resolving over-segmentation error in *aytam* /ah/. (a) The /ah/ symbol in DOCS stage is fragmented to 3 stroke groups. The mean of the likelihoods of the most probable symbols for the stroke groups in (b),(c) and (d) is compared to that of /ah/ for the stroke group in (e). (f) The correctly segmented word after the merge.

We favor splitting the stroke group $S_k$ into $S_{k_1}$ and $S_{k_2}$ whenever

$$\frac{\sum P(\omega_{top}^{k_j})}{2} > P(\omega_{top}^{k}) \tag{3.12}$$

Here $\omega_{top}^{k}$ represents the most probable symbol of the SVM for the stroke group $S_k$. For the scenario, where the inequality is not satisfied, additional cues (derived from statistics) are employed for resolving the under-segmentation error in $S_k$.

1. If the number of dominant points $N^{S_k}$ in $S_k$ is greater than the maximum number ($T_{dp}^{max}(\omega_{top}^{k})$) determined for the most probable symbol $\omega_{top}^{k}$ in the study on the IWFHR data-set, we proceed ahead in segmenting it to 2 valid symbols $S_{k_1}$ and $S_{k_2}$.

2. If $d_{max}$ obtained for the stroke group $S_k$ is greater than maximum horizontal inter stroke gap ($T_{dmax}(\omega_{top}^{k})$) for $\omega_{top}^{k}$, we segment it.

Figure 3.26 illustrates the case wherein the wrongly segmented stroke group ெந /ne/ at the start of the word நெருடல் /neruTal/ is segmented correctly to 2 valid symbols ெ ($VM$ of /e/) and ந /na/, respectively.

For segmenting stroke groups satisfying $h_{min} < 0$, we have $S_{k_1} = \{\widetilde{s}_k^1, \widetilde{s}_k^2, .....\widetilde{s}_k^g\}$ and $S_{k_2} = \{\widetilde{s}_k^{g+1}, \widetilde{s}_k^{g+2}, .....\widetilde{s}_k^m\}$. Here $h_{min}$ corresponds to the vertical gap between $g^{th}$ and $(g+1)^{th}$ strokes. An approach similar to the one adopted for $b_{max} > 0$ is employed to segment $S_k$. Figure 3.27 presents an illustration, wherein the first stroke group வீர

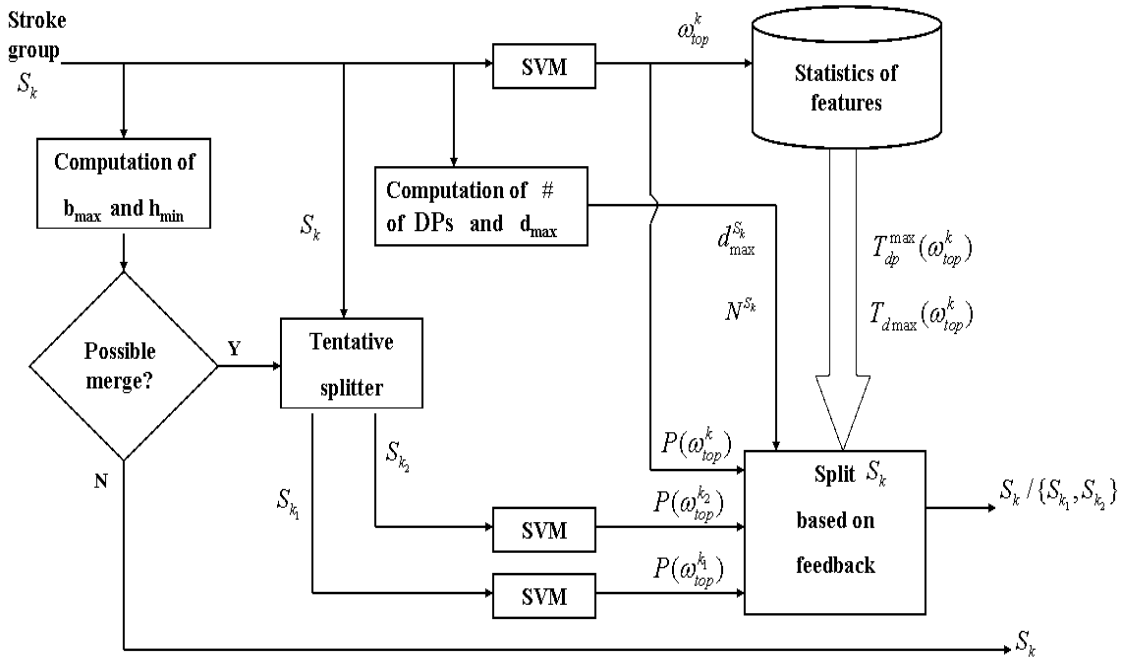



Fig. 3.25: AFS module for resolving under-segmented stroke groups.

/vIra/ in the word வீரம் /vIram/ satisfying the inequality $h_{min} < 0$ is split to 2 valid symbols வீ /vI/ and ர /ra/ respectively.

## 3.8 Results and discussion

### 3.8.1 Experimental setup

Prior to applying the proposed segmentation scheme, the parameters of SVM are trained with the concatenated $x$ and $y$ coordinates of the preprocessed Tamil symbols as described in Sec 2.5. The online trace is robust in discriminating valid Tamil symbols from outlier patterns that arise due to incorrect segmentation. In addition, for each symbol $\omega_i$, the following statistics are generated.

1. Maximum number of dominant points ($T_{dp}^{max}(\omega_i)$) across all samples of $\omega_i$.

2. Least likelihood $T_P^{min}(\omega_i)$ returned by the SVM across all correctly recognized samples of $\omega_i$.

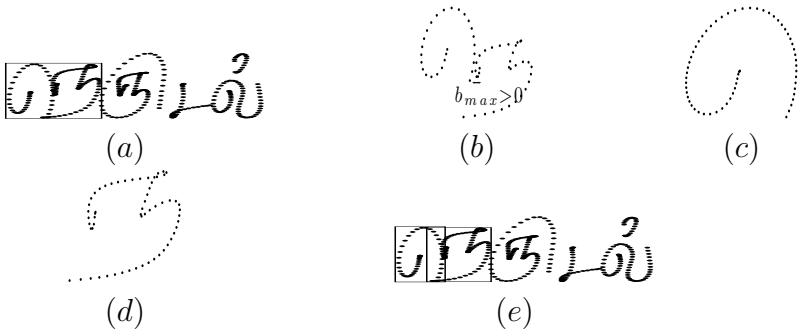



Fig. 3.26: An example illustration of AFS scheme for resolving under-segmentation errors in Tamil words. (a) A word under-segmented by DOCS. (b) The first stroke group in the word satisfies $b_{max} > 0$ and is assumed to comprise 2 merged valid symbols. (c)(d) The extracted symbols are recognized separately. The stroke group is split if the mean likelihood of the extracted symbols exceeds the likelihood for the combined symbol shown in (b). (e) The correctly segmented word after the split.

3. $T_{dmax}(\omega_i)$ - Maximum horizontal inter stroke gap (as defined in Eqn 3.7) over all samples.

4. $T_o^p(\omega_i)$ - Maximum ratio of overlap of the dot with the base consonant $\omega_i$. This statistic is defined for the pure consonants only.

In the following sections, we describe experiments demonstrating the effectiveness of the AFS module in correcting segmentation errors.

### 3.8.2 Segmentation results on the IWFHR Tamil database

Though the primary focus is on segmenting Tamil words, as a first experiment, we evaluate the performance of the proposed approach on the symbols in the IWFHR training dataset. As mentioned in Sec 3.4, for the isolated symbols in this dataset, the errors can arise only due to over-segmentation.

For ease of analysis, we manually divide the 155 symbols in Appendix C into 8 groups. The groups have been created by clubbing symbols that are linguistically similar (vowels, base consonants, pure consonants, CV combinations of இ /i/, ஈ /I/, உ /u/ and ஊ /U/). In addition, the 6 symbols left out (4 vowel modifiers ா ($VM$ of /A/), ி ($VM$ of /e/), ே ($VM$ of /E/), ை ($VM$ of /ai/) and 2 special symbols ஃ /ah/ , ஸ்ரீ /sri/) are

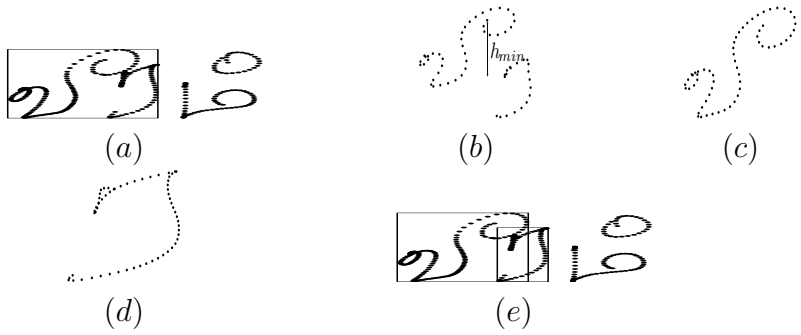



Fig. 3.27: Another example of AFS for resolving under-segmentation errors in Tamil words. (a) A word under-segmented by DOCS. (b) The first stroke group in this word satisfies the condition $h_{min} < 0$. (c) and (d) The individual strokes from this stroke group are extracted and recognized separately. The likelihood averaged over these stroke groups is greater than the likelihood of the combined stroke group in (b). Hence, the stroke group is split into the two valid symbols. (e) Correctly segmented word after the split.

merged into a separate group (referred to as 'additional symbols'). Thus, each symbol belongs to exactly one group listed below.

| | |
|---|---|
| $G_1$ | Base consonants |
| $G_2$ | Pure consonants |
| $G_3$ | Additional symbols |
| $G_4$ | CV combinations of vowel ◌ு /u/ |
| $G_5$ | CV combinations of vowel ◌ி /i/ |
| $G_6$ | Pure vowels |
| $G_7$ | CV combinations of vowel ◌ீ /I/ |
| $G_8$ | CV combinations of vowel ◌ூ /U/ |

In order to study the effect of the proposed AFS scheme separately on symbols ஃ /ah/ and ஈ /I/, we separate them out from their respective groups. Accordingly, we consider the groups $G_3$ and $G_6$ as

| | |
|---|---|
| $G_3^1$ | Additional symbols (apart from ஃ /ah/) |
| $G_3^2$ | ஃ /ah/ |

Table 3.1: Performance evaluation of the AFS strategy on the broken symbols of the IWFHR database. (Trial experiment performed on training data.)

| *Group* | # of samples | # of DOCS errors | # of AFS errors | % Error red--uction (AFS) | Overall seg--mentation rate (DOCS) | Overall seg--mentation rate (AFS) |
|---|---|---|---|---|---|---|
| $G_1$ | 7457 | 46 | 8 | 82.6 | 99.4 | 99.9 |
| $G_2$ | 7523 | 108 | 8 | 92.6 | 98.5 | 99.9 |
| $G_3^1$ | 1658 | 15 | 5 | 66.6 | 99.1 | 99.7 |
| $G_3^2$ | 340 | 322 | 8 | 97.5 | 5.2 | 97.6 |
| $G_4$ | 7351 | 481 | 34 | 92.9 | 93.4 | 99.5 |
| $G_5$ | 7534 | 201 | 15 | 92.5 | 97.4 | 99.8 |
| $G_6^1$ | 3382 | 26 | 4 | 84.6 | 99.2 | 99.9 |
| $G_6^2$ | 332 | 251 | 2 | 99.2 | 24.4 | 99.4 |
| $G_7$ | 7525 | 195 | 14 | 92.8 | 97.4 | 99.8 |
| $G_8$ | 7237 | 432 | 151 | 65.0 | 94.0 | 97.9 |
| Total | 50339 | 2077 | 249 | 88.0 | 95.9 | 99.5 |

$G_6^1$ Vowels (apart from ை /I/)

$G_6^2$ ை /I/.

Table 3.1 illustrates the results of the proposed AFS strategy on each of these groups. 75.6% of samples of the symbol ை /I/ ($G_6^2$) are prone to errors in the DOCS module. As high as 99% of these errors have been rectified by the AFS strategy. Only 18 samples ( 5%) of ஆ /ah/ ($G_3^2$) are segmented as a single stroke group by DOCS. The AFS module corrects 314 (97.5%) wrongly segmented samples. For pure consonants (comprising 7523 samples in $G_2$), 100 out of 108 (92.6%) samples are properly segmented by AFS. Strategies proposed in Sec 3.6 prove effective in resolving an average of 83.6% of the segmentation errors in CV combinations ($G_4$, $G_5$, $G_7$ and $G_8$). In addition, we observe that the base consonants ($G_1$), the vowels in $G_6^1$ and the additional symbols in $G_3^1$ are least prone to segmentation errors, compared to the other symbols. The results show that, on an average, the AFS corrects 80.4% of the errors in these 3 groups. In summary, the attention feedback strategies proposed reduce the under-segmentation errors drastically

Table 3.2: Performance evaluation of the AFS strategy on one set of words from the MILE word database (DB1). Total # of words=250. Total # of symbols=1210.

| | DOCS | AFS | % error reduction |
|---|---|---|---|
| # of merged symbols | 89 | 9 | 89.9 |
| # of broken symbols | 14 | 3 | 78.6 |
| Correctly segmented symbols (in %) | 91.5 | 99.0 | 88.3 |
| # of correctly segmented words | 183 | 243 | |
| # of wrongly segmented words | 67 | 7 | 89.5 |

(by around 88.0%) across the entire database. In addition, 1828 additional symbols have been correctly segmented. This results in an improvement of 3.6% in the segmentation of symbols over the DOCS scheme. As high as 99.5% of symbols get correctly segmented after AFS.

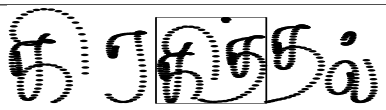
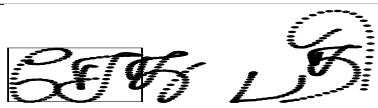
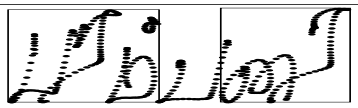
### 3.8.3 Segmentation results on the MILE word database

The proposed techniques are tested on the entire word database. However, to start with, we evaluate the performance on the validation set DB1. Owing to a significant number of wrongly segmented stroke groups resulting from the DOCS module, DB1 has been selected for validating the proposed AFS strategies. Table 3.2 outlines the statistics of segmentation errors. Of the 103 errors, 86% corresponds to the merging of valid symbols. The AFS module described in Sec 3.7 aids in properly detecting and correcting 90% of these errors. In addition, the methods proposed effectively merge 78% of the over-segmented stroke groups to valid symbols. The improvement in character segmentation rate in turn reduces the number of wrongly segmented words. It can be observed from the last row of the table that 60 additional words have been properly segmented. On evaluating the performance across the entire word database of 10000 words, we obtain a 86% reduction in character segmentation errors (Table 3.6).

### 3.8.4 Recognition results on the MILE word database

In this subsection, we report experimental results demonstrating the impact of the proposed AFS strategies on the recognition of symbols in the MILE word database. A few sample words, whose segmentations have been corrected by our approach, are shown in Tables 3.3 and 3.4. Application of the DOCS on each word in Table 3.3 leads to a merge

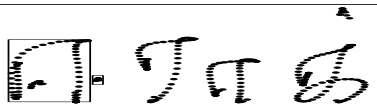
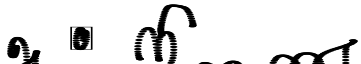
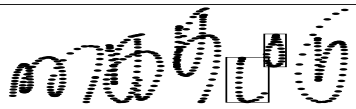
Table 3.3: Merger of two or more symbols by DOCS, split by AFS and consequent improvement in recognition. The valid symbols merged by the DOCS module are shown within a box in the first column. The symbols contained within the boxes in the second column indicate the recognition errors.

| Input word under-segmented by DOCS | Recognition o/p for DOCS stroke groups | Recognition o/p for AFS stroke groups |
|---|---|---|
|  | கிரஓதல் /kiraOtal/ | கிரகித்தல் /kirakittal/ |
|  | சஷ்துபதி /kshtupati/ | சேதுபதி /cetupati/ |
|  | ஹுபங் /hupang/ | பரம்பரை /paramparai/ |

of valid symbols. On the other hand, at least one valid symbol in each word in Table 3.4 appears as more than one stroke group due to over-segmentation. The incorrect segmentation in turn increases the symbol recognition errors, as shown in the second column of the two tables. From the third columns, we observe that all the constituent symbols of these words are recognized correctly after AFS.

Table 3.5 compares the recognition accuracy for the set DB1, obtained with DOCS and AFS. Since a significant percentage of DOCS errors are corrected by AFS, a drastic improvement of 16% (from 70.5% to 87.1%) in symbol recognition is observed. In computing the symbol recognition rate, apart from the substitution errors, we take

Table 3.4: Splitting of symbols into two stroke groups by DOCS, correct segmentation by AFS and consequent improvement in recognition. The split parts of valid symbols broken by the DOCS module are highlighted with boxes in the first column. The symbols contained within the boxes in the second column indicate the symbol recognition error.

| Input word over-segmented by DOCS | Recognition o/p for DOCS stroke groups | Recognition o/p for AFS stroke groups |
|---|---|---|
|  | ஈஃராக் /IahrAk/ | ஈராக் /IrAk/ |
|  | அப்ய்ட்றிணை /apyTRinnai/ | அஃறிணை /aahRinnai/ |
|  | கைதடபடு /kaitaTapaTu/ | கைதட்டு /kaitaTTu/ |
|  | கடவுனசூ /kaTavuNacU/ | கடவுள் /kaTavuL/ |

into account the insertion and deletion errors, caused by over-segmentation and under-segmentation, respectively. The edit distance [18] is used for matching the recognized symbols with the ground truth data. Moreover, 11.6% of the words, (29 additional words) wrongly recognized after DOCS, have been corrected by the proposed technique. Across the 10000 words in the MILE word database, an improvement of 4.5% in symbol recognition rate was obtained (Table 3.6).

In all of the preceding experiments and discussions, sets of consecutive strokes of the word are merged into stroke groups by DOCS by comparing their degree of overlap $O_k^c$ (defined in Eqn 3.1) to a threshold $T_0 = 0.2$. The number of properly segmented stroke groups generated by DOCS depends on the value of $T_0$. Figure 3.28 (a) quantifies the frequency of errors due to symbol merges and splits as a function of the overlap threshold. We vary $T_0$ from 0 to 0.9 in steps of 0.1 and demonstrate the effectiveness of the

Table 3.5: Impact of the proposed AFS scheme on the symbol and word recognition rates on DB1. Total # of words=250. Total # of symbols=1210.

| | DOCS | AFS | % error reduction |
|---|---|---|---|
| # of correctly recognized symbols | 853 | 1054 | 56.3 |
| % of correctly recognized symbols | 70.5 | 87.1 | |
| # of correctly recognized words | 85 | 114 | 11.6 |
| % of correctly recognized words | 34 | 45.6 | |

Table 3.6: Impact of the AFS scheme on the segmentation and recognition of symbols in the MILE word database. Total # of words=10000. Total # of symbols=53246.

| | DOCS | AFS | % error reduction |
|---|---|---|---|
| Total # of segmentation errors | 1001 | 139 | 86.2 |
| Segmentation rate in (%) | 98.1 | 99.7 | 1.6 |
| Symbol recognition rate in (%) | 83.9 | 88.4 | 4.5 |

proposed attention feedback segmentation method on DB1, irrespective of the threshold selected. $T_0 = 0$ leads to the maximum number of unintentional merges, especially when symbols are written close enough to each other that their bounding boxes are adjacent. For higher values of $T_0$, a significant number of valid stroke groups get over segmented (refer Fig. 3.28 (a)). Irrespective of the threshold set, the AFS scheme is able to correct at least 75% of the segmentation errors encountered (Fig. 3.28 (b)). The corresponding improvement in symbol recognition accuracy of the handwriting system for the different threshold values is presented in Fig. 3.28 (c). We observe from Fig. 3.28 (b) that $T_0 = 0.2$ gives the minimum segmentation error rate after the AFS step. Moreover, from Fig 3.28 (c) we note that the highest recognition performance after the AFS step is reported for this value of $T_0$. Hence, we chose this threshold value for our experiments and illustrations in this work.

However, two aspects of the proposed techniques needs to be addressed. Owing to the incorporation of spatial and temporal information of strokes in the attention-feedback methods, segmentation tends to fail in cases where symbols are written as a different temporal sequence rarely encountered in modern Tamil script. One way to address this

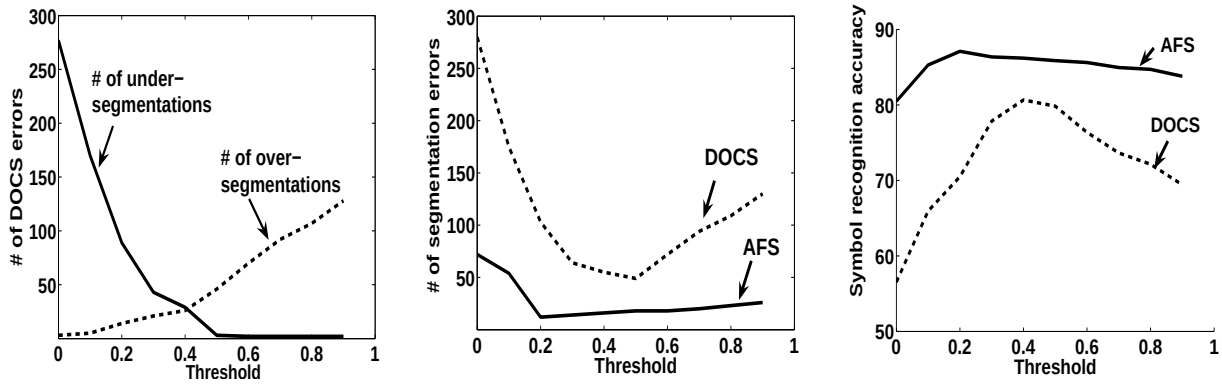



Fig. 3.28: Effectiveness of AFS on DB1 (with 1210 symbols) as a function of the overlap threshold used in the DOCS module. (a) Variation of number of over-segmentations and under-segmentations by DOCS. (b) Number of incorrect segmentations by DOCS compared against that of the AFS module. (c) Symbol recognition rate (in %) for stroke groups from the DOCS module as against that of the AFS module.

issue is to convert the stroke information to an offline image and then attempt recognition. Moreover, in words,where two or more symbols are written by a single stroke, attention feedback segmentation does not work effectively. However, as mentioned earlier in Sec 3.1, cursive handwriting is rare in Tamil. Secondly, the methods proposed are not robust in merging symbols comprising large horizontal inter-stroke gaps, that are comparable to the horizontal inter-character gaps. Referring to Fig. 3.29, the otherwise double stroke symbol ள் /L/ in the word ரசிகர்கள் /racikarkaL/ is so badly written with four strokes that their horizontal inter-stroke gap is comparable to the inter-character gaps. Our algorithm fails in such cases.

Given that there is no prior work done in segmenting online Tamil words, it is

Fig. 3.29: Illustration of a word that does not get properly segmented by the AFS strategy. The broken stroke groups contained within the dotted box fail to merge to the valid symbol /L/.

difficult to compare our method to a benchmark. The segmentation scheme proposed for cursive Bangla words in [40, 41] cannot be extended to Tamil, owing to major structural

differences in the scripts.

## 3.9 Summary

In this chapter, a novel, lexicon-free, attention-feedback segmentation approach for handwritten online Tamil words is presented. Initial segmentation of the given word is performed by the DOCS module into a set of stroke groups. Attention on certain spatial and temporal features detect likely split and under-segmented stroke groups, if any. The likelihoods fed back by the SVM as well as known statistics of stroke-group based features corrects the wrongly segmented stroke groups to form valid patterns (or symbols) in the AFS module. The correction of stroke groups by the AFS module in turn leads to an improvement in the performance of the handwriting recognition system designed with SVMs.

The SVM classifier fed with concatenated $x$-$y$ coordinates are found to be quite effective to the problem of segmentation. However, the classifier is not robust to effectively distinguishing between similar looking symbols. With the view of improving the performance of symbol recognition beyond that given by the primary classifier, we propose in the subsequent two chapters of this thesis, two post-processing approaches, namely reevaluation strategies and language models.

# Chapter 4

# Reevaluation strategies for online Tamil symbols


## Abstract

*In this chapter, we aim at reducing the error rate of the Tamil symbol recognition system by employing multiple experts to reevaluate certain decisions of the primary classifier. Motivated by the relatively high percentage of occurrence of base consonants in the script, a reevaluation technique has been proposed to correct any ambiguities arising in the base consonants. Secondly, a DTW method is proposed to automatically extract the discriminative regions for each set of confused characters. Class-specific features derived from these regions aid in reducing the degree of confusions. Thirdly, statistics of specific features are proposed for resolving any confusions in vowel modifiers. The reevaluation approaches, when tested on the MILE word database, improve the symbol recognition rate by 3.5%. The reduction in the error rate has been achieved using a generic approach, without the incorporation of language models.*

## 4.1 Literature survey

Recognizing handwritten Indic script characters is a non-trivial pattern recognition problem. As discussed in Sec 2.4, the challenges arise primarily due to the presence of larger character sets, complex character shapes, different variations of writing styles and a non-finite lexicon.

An assessment of the primary classifier (SVM) performance attributes most of the misclassifications to the presence of symbols that appear visually similar. The SVM classifier working on features at a global level, at times, fails to capture finer nuances that distinguish these symbols. One way to alleviate this drawback is to incorporate experts that employ class-specific features to reduce the degree of confusion between frequently confused characters. Specifically, the current work proposes techniques for reevaluating the recognition output from the primary classifier. The approaches developed take into account the popular writing styles of modern Tamil script.

Human vision can automatically locate the distinct regions in confused symbol pairs so as to distinguish one from the other. For the handwriting system to mimic this remarkable ability, we propose a dynamic time warping (DTW) approach for learning the finer nuances that discriminate similar looking symbols. The developed technique aids in extracting the relevant part of strokes for deriving class-specific features.

Literature has many proposals to deal with the problem of reducing the confusions between visually similar characters in non-Indic scripts. A two stage classification strategy has been adopted in [94] for Latin script recognition. At the first level, confusions between characters (referred to as 'conflicts') are detected using an ensemble of classifiers. To resolve the conflicts, two different architectures of support vector classifiers are introduced at the second level as verifiers. Hybrid MLP-SVM structures have been used in [95] for recognizing handwritten digits. Specialized SVMs are developed to operate on the two highest MLP outputs at the second level to generate the correct class. This work assumes that the correct class almost consistently occurs within the top two recognized digits from the MLP classifier. A similar approach has been presented in [96], wherein a model based Bayesian classifier is employed at the first stage to generate the

two most probable classes for the input character. At the second stage, a discriminative classifier (probabilistic neural network) is used to reduce the confusion between the two ambiguous classes obtained from the first level. For Persian script, fine classification of unconstrained handwritten numerals has been achieved by removing confusions between similar looking classes at the second level [97].

Reverting to the context of online Indic scripts, there is hardly any comprehensive work that addresses the problem of disambiguating similar looking characters. As discussed in Sec 1.5, most reported techniques deal with the problem of recognizing isolated characters in a single stage. However, in the area of optical character recognition, post-processing schemes have been successfully attempted for a few scripts. Shape encoding based post-processing methods have been used for improving the Gurmukhi OCR system [98]. In addition, a lexicon look-up strategy based on bigram analysis has been proposed by Lehal in [99]. Sub-character level language modeling techniques have been used as a post-processing step to correct Malayalam words in [100]. OCR errors in Bangla [101] have been rectified with morphological parsing techniques.

Studies on scene perception indicate that our visual processing system follows a top-down approach. The global cues characterizing the object (that appears within the visual span) are perceived prior to the local features. The human perceptual system treats a scene as if it were in the process of being focussed or zoomed in on, where at first, it is relatively less distinct. Moreover, the human perceptual processor has the capability to select parts of the input stimulus that are worth paying attention to. Taking analogies from these observations in the field of neuroscience [102], we present a recognition strategy that first works on the global features ($x$-$y$ coordinates of the entire trace) to output a particular Tamil symbol class for the given input pattern. By analyzing local features characteristic to the given input pattern, we reevaluate the class label to reduce the symbol error rate. The localized features are derived by zooming on /paying attention to specific parts of the online trace. Essentially, we adopt a multi-pass system, wherein fine grained processing is guided by the prior cursory (global) processing.

Table 4.1: Occurrence statistics of different groups of Tamil symbols, as derived from the MILE text corpus.

| *Group* | Description | # of symbols | % of symbols |
|---|---|---|---|
| $G_1$ | Base consonants | 368387 | 33.5 |
| $G_2$ | Pure consonants | 266525 | 24.2 |
| $G_3$ | Additional symbols | 191282 | 17.4 |
| $G_4$ | CV combinations of உ /u/ | 104360 | 9.6 |
| $G_5$ | CV combinations of இ /i/ | 99421 | 9.1 |
| $G_6$ | Pure vowels | 57858 | 5.3 |
| $G_7$ | CV combinations of ஈ /I/ | 6252 | 0.6 |
| $G_8$ | CV combinations of ஊ /U/ | 5105 | 0.4 |

## 4.2 Need for reevaluation strategies

While considering the need to reevaluate a Tamil symbol, two aspects are taken into account.

- Its frequency of occurrence in a large Tamil text corpus.
- The extent to which it gets confused with a visually similar looking symbol by the primary classifier.

An extensive text corpus (henceforth referred to as 'MILE' text corpus), comprising 1.5 million Tamil words (derived from books), was utilized for generating the frequency count of each of the 155 symbols. We consider the statistics of the symbols obtained from this corpus to be representative of the script. For ease of analysis, the symbols are divided into 8 groups (as described in Sec 3.8.2). Table 4.1 lists the occurrence frequency of the groups in the corpus.

We observe that base consonants ($G_1$) alone constitute 33% of the total corpus. In addition, base consonants occur as separate strokes in pure consonants ($G_2$) , CV combinations of இ /i/ ($G_5$) and ஈ /I/ vowels ($G_7$). For multi-stroke handwritten symbols in groups $G_2$, $G_5$ and $G_7$, the base consonant can be extracted by employing spatial cues derived from the strokes. For illustration, consider the CV combinations தி /ti/, தீ /tI/ and the pure consonant த் /t/. From each of these 3 symbols, we can easily

extract the base consonant ($BC$) த /ta/. Thus, effectively the occurrence of base consonants in the script is much higher than the percentage denoted by $G_1$ alone. In fact, considering across the groups $G_1$, $G_2$, $G_5$ and $G_7$, base consonants can be extracted as an independent entity in 67.4% (33.5% +24.2% +9.1%+ 0.6%) of the symbols in the corpus. Moreover, a few pairs of consonants like (ல /la/, வ /va/) and (ள /La/,ன /Na/) look visually similar and get confused by the primary classifier in 4 to 6.5% of the cases (Table 4.2). Due to the higher percentage of base consonants and possible confusions, it becomes imperative to reevaluate

- base consonants in CV combinations of இ /i/ and ஈ /I/.
- base consonants in pure consonants.
- the frequently confused base consonants.

As discussed in Sec 2.1, the inherent vowel sound of a base consonant is suppressed by the dot, resulting in a pure consonant. Pure consonants ($G_2$) account for 24% of the symbols in the MILE text corpus. However, the size of the dot varies with the style of writing and hence the primary classifier at times interprets them to be the vowel modifiers ($VM$) of இ /i/ or ஈ /I/ and vice versa, thereby resulting in an erroneous symbol. In addition, confusions arise between the $VM$ of இ /i/ and ஈ /I/ in their corresponding CV combinations $G_5$ and $G_7$ (that account for 9.7% of the symbols in the corpus). Accordingly, we reevaluate

- vowel modifier strokes in test samples assigned to CV combinations of இ /i/ and ஈ /I/ by the primary classifier.
- dot strokes in test samples assigned to pure consonants by the primary classifier.

Amongst the remaining symbols, confusions arise between the visually similar (ழ /mu/,ழ /zhu/), (ள /La/, ன /Na/, ண ($VM$ of /ai/)) and (க /ka/,ச /cu/). Class-specific features derived from the discriminative regions of these symbol sets help in their disambiguation. Table 4.2 lists a few of the similar looking pairs with their frequencies of

Table 4.2: Some symbol confusions encountered at the output of the primary classifier (SVM) and their frequency of occurrence in the IWFHR 2006 Tamil test symbol set.

| Symbol pairs | Total # of symbols | # of confusions | Primary classifier accuracy in % |
|---|---|---|---|
| (ழ ,ழ ) (mu, zhu) | 349 | 26 | 92.6 |
| (ன ,ண ) (Na, $VM$ of /ai/) | 351 | 32 | 90.9 |
| (னி ,ளி ) (Ni, Li) | 364 | 32 | 91.2 |
| (ள ,ன ) (La, Na) | 353 | 23 | 93.5 |
| (கி ,சி ) (ki, ci) | 355 | 17 | 95.2 |
| (ல ,வ ) (la, va) | 359 | 14 | 96.1 |

confusion and their recognition accuracies from the primary SVM classifier.

Let $\mathbf{C}$ denote the confusion matrix of size $155 \times 155$ resulting from the primary classifier across the test samples in the IWFHR Dataset.

$$\mathbf{C} = \begin{pmatrix} c_{1,1} & c_{1,2} & \dots & \dots & c_{1,155} \\ c_{2,1} & & \dots & \dots & c_{2,155} \\ .. & & & & \\ .. & & & & \\ c_{155,1} & & \dots & \dots & c_{155,155} \end{pmatrix}$$

Accordingly, $c_{i,j}$ represents the number of samples of symbol $\omega_i$ getting wrongly classified as $\omega_j$. The number of confusions for a symbol pair $(\omega_i, \omega_j)$ can be written as

$$c_T(i,j) = c_{i,j} + c_{j,i} \tag{4.1}$$

For a symbol $\omega_i$, the set of symbols to which it can get frequently confused by the primary classifier is represented by

$$\Omega_i = \{\omega_j | c_T(i,j) \geq \delta, i \neq j\} \tag{4.2}$$

In this work, we have chosen $\delta = 10$. We denote the set of all symbols that possibly can get confused, and hence need to be reevaluated as

$$\Omega = \bigcup_i \ \Omega_i \tag{4.3}$$

Motivated by the observations outlined above, the present work improves on the recognition accuracy of the primary classifier by proposing reevaluation strategies for resolving any possible ambiguities in base consonants, pure consonants, vowel modifiers and frequently occurring confusion symbol pairs.

## 4.3 Overview of proposed reevaluation strategy

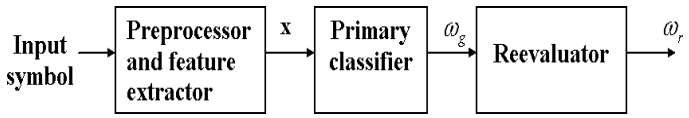



Fig. 4.1: Block diagram of the recognition strategy for an input Tamil symbol.

Figure 4.1 presents the overall picture of the proposed recognition strategy for a Tamil symbol. We assume that the input raw Tamil word is segmented into its constituent symbols by employing the attention feedback strategies discussed in the previous chapter. The trace of each segmented symbol is preprocessed as described in Sec 2.5.1 and the resulting concatenated $x$-$y$ coordinates $\mathbf{x}$ are fed to the primary classifier. The classifier assigns the symbol to the class $\omega_{top}$ with the highest posterior probability. In order to reflect the global nature of the primary classifier, we consider a slight modification to the notation by replacing the subscript '*top*' in $\omega_{top}$ with '$g$'. Hereinafter, we refer to the label of the most probable symbol from the primary SVM classifier with $\omega_g$.

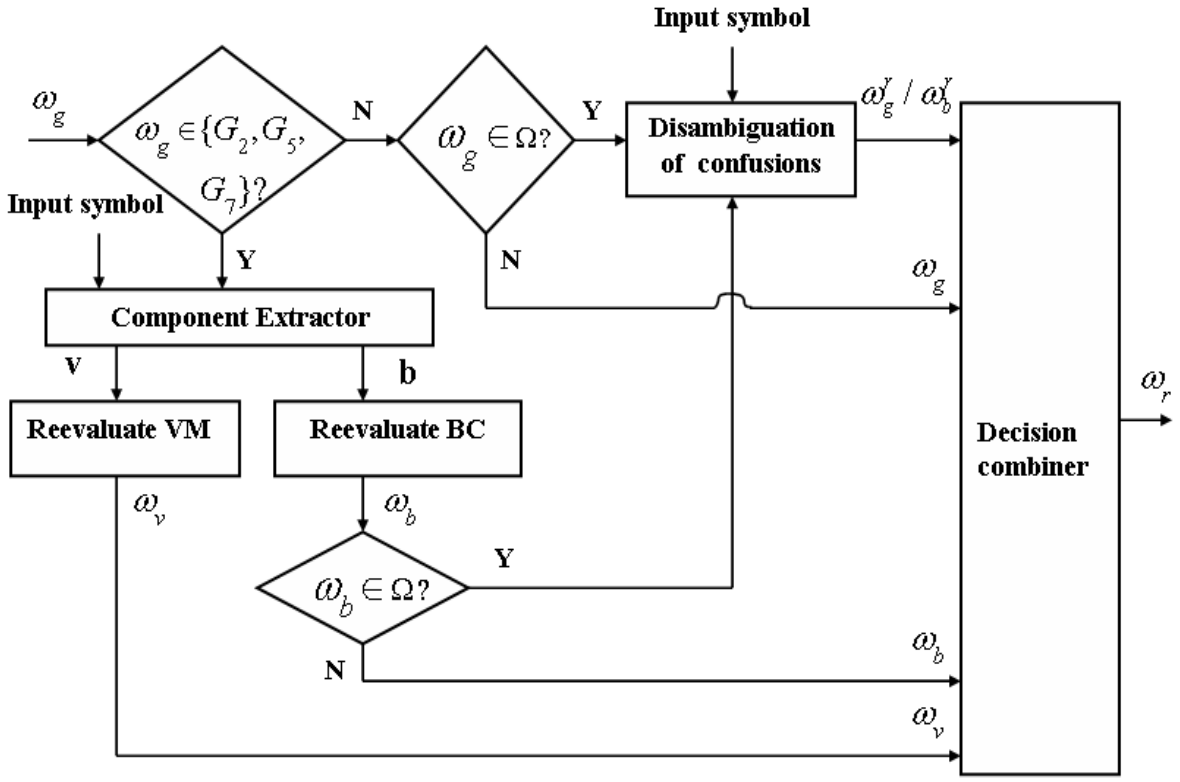



Fig. 4.2: Details of the proposed reevaluation block. $G_2$: Pure consonant group; $G_5$: CV combinations of /i/; $G_7$: CV combinations of /I/, $\Omega$: Set of all confused symbols; **b**, **v**: extracted base consonant and vowel modifier/dot stroke part; $\omega_g$: label given by primary classifier; $\omega_r$: label after reevaluation. $\{\omega_b, \omega_v, \omega_b^r, \omega_g^r\}$: refer Table 4.3.

Based on $\omega_g$, multiple novel reevaluation strategies are proposed to reduce the chances for the misclassification of the symbol. For better clarity, the reevaluation block in Fig. 4.1 is expanded in Fig. 4.2 and discussed below.

1. When the primary classifier outputs a pure consonant or CV combination of இ /i/ or ஈ /I/ vowel as its most probable symbol ($\omega_g \in \{G_2, G_5, G_7\}$), we separately extract the base consonant ($BC$) and vowel modifier ($VM$)/dot with the component extractor and derive new discriminative features for reevaluating them. Let $\omega_b$ and $\omega_v$ represent the independently reevaluated labels for the base consonant ($BC$) and vowel modifier ($VM$). Furthermore, if the base consonant $\omega_b$ is likely to

Table 4.3: Logic for generation of the final label $\omega_r$ for the recognized symbol in the decision combiner module in Fig. 4.2.

| Label $\omega_r$ | Constraints |
|---|---|
| $\omega_g$ | $\omega_g \notin \{G_2, G_5, G_7\}$ , $\omega_g \notin \Omega$ |
| $\omega_g^r$ | $\omega_g \notin \{G_2, G_5, G_7\}$ , $\omega_g \in \Omega$ |
| CV combination generated by appending $\omega_v$ to $\omega_b$ | $\omega_g \in \{G_2, G_5, G_7\}$ , $\omega_b \notin \Omega$ |
| CV combination generated by appending $\omega_v$ to $\omega_b^r$ | $\omega_g \in \{G_2, G_5, G_7\}$ , $\omega_b \in \Omega$ |

be confused with another base consonant (in other words, $\omega_b \in \Omega$), we subject it to a second round of reevaluation by disambiguating it from its possible confusions.

2. If $\omega_g \in \Omega$, class-specific discriminative features are derived from the preprocessed symbol. The reevaluation strategy is achieved using appropriate expert classifiers, each of which is designed to disambiguate a specific confusion set.

The decision combiner finally combines the various labels to generate the appropriate output symbol $\omega_r$ (see Table 4.3). It is to be noted that we adopt a generic approach for recognizing words, without involving the use of language models. Our main objective is to explore as to how far we can go ahead in improving the recognition rate of the primary classifier, by reevaluating symbols based on class-specific features.

## 4.4 Reevaluation of base consonants

Consider a preprocessed $m$-stroke ($m > 1$) handwritten symbol recognized as a CV combination of இ /i/ ($G_5$) or ஈ /I/ ($G_7$). The component extractor module separates the $BC$ from $VM$ by employing the maximum vertical inter-stroke gap $h_{max}$ (derived from the symbol). Let $h_{max}$ correspond to the spacing between the $r^{th}$ and $(r+1)^{th}$ strokes. Accordingly, the first $r$ strokes, assumed to comprise $n_B$ sample points denotes the trace of the $BC$ and is represented by $\mathbf{b}$. The remaining $(m-r)$ strokes represent $\mathbf{v}$, the trace of the $VM$. As mentioned in Sec 2.5, the number of resampled points in the

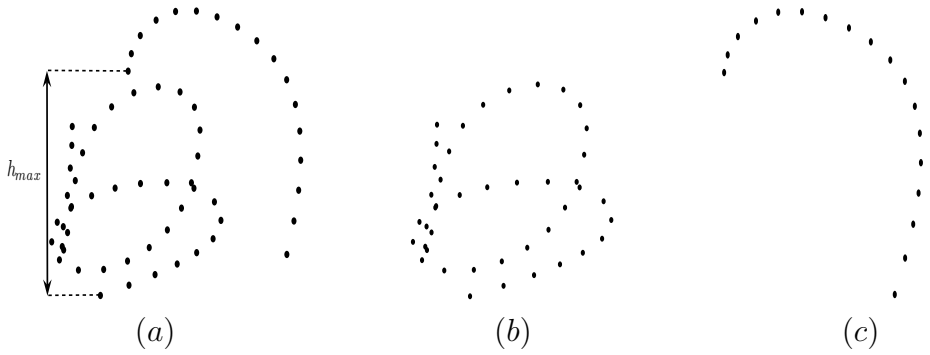



Fig. 4.3: Extraction of the base consonant and vowel modifier from the CV combination /ki/. (a) CV combination. (b) Base consonant. (c) Vowel modifier.

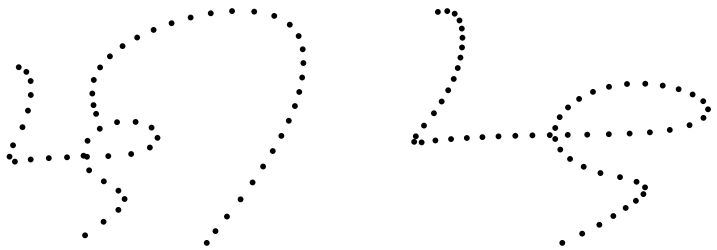
preprocessed symbol, $n_P = 60$ in our experiments.

$$\mathbf{b} = \{x_i, y_i\}_{i=1}^{n_B} \tag{4.4}$$

$$\mathbf{v} = \{x_i, y_i\}_{i=n_B+1}^{n_P} \tag{4.5}$$

Figure 4.3 illustrates the scenario, wherein the base consonant (in (b)) and vowel modifier (in (c)) are extracted from the CV combination கி /ki/ (in (a)) using the component extractor module. A similar approach is employed to extract the dot from the base consonant in a pure consonant ($G_2$). For ease of notation, we denote the $(m-r)$ strokes representing the dot in a pure consonant also by $\mathbf{v}$.

The reevaluation module for base consonants (in Fig. 4.2) is invoked whenever $\omega_g \in \{G_2, G_5, G_7\}$. For illustrating the proposed strategy, assume that the most probable output of the primary classifier $\omega_g$ for the input pattern is a CV combination of இ /i/ vowel ($G_5$). The first $r$ strokes of the raw input data, representing the trace of the extracted $BC$, is sent to the preprocessing module discussed in Sec 2.5. The resulting feature vector (concatenated $x$-$y$ features) $\mathbf{x}_b$ is separately fed to the SVM classifier $C_b$ dedicated to recognize only the base consonants. Compared to the primary SVM classifier that is trained across the 155 Tamil symbols of the IWFHR database, classifier $C_b$ is trained using the samples of the 23 base consonants only. Let $\omega_b$ be the base consonant label obtained from the reevaluation module. The most probable consonant



(a) (b)

Fig. 4.4: Illustration of base consonant reevaluation. (a) This symbol, which is /zhi/, is wrongly recognized as /*mi*/ by the primary classifier. (b) The preprocessed pattern of the extracted base consonant is recognized by classifier $C_b$ as /zha/.

from the classifier $C_b$ is regarded as the reevaluated label and is assigned to $\omega_b$.

Figure 4.4 presents the scenario wherein the primary classifier regards the pattern in (a) as மி /mi/. However, the classifier $C_b$ assigns the extracted base consonant pattern shown in (b) to ழ /zha/ (which happens to be the correct symbol). Hence, the pattern after reevaluation is assigned to ழி /zhi/, provided the reevaluated vowel modifier corresponds to ி /i/.

A similar analysis (as described above) is applied to reevaluate the base consonants in CV combinations of vowel ெ /I/ and pure consonants.

## 4.5 Reevaluation of dots and vowel modifier strokes

In this section, we propose strategies to reevaluate the pattern **v** obtained from the component extractor. We adopt a two step process as outlined below

- We first disambiguate the dot stroke from the modifiers of ி /i/ or ெ /I/ vowel (Sec 4.5.1).
- If **v** is not a dot stroke, we reevaluate the modifiers of ி /i/ and ெ /I/ vowels (Sec 4.5.3).

Let $\omega_v$ correspond to the label of the $VM$ after reevaluation.

### 4.5.1 Recognition of dots in pure consonants

In this subsection, we propose strategies to detect the cases of the primary classifier confusing the dot in a pure consonant ($G_2$) with the vowel modifier in a CV combination ($G_5$ or $G_7$). It is assumed here that the primary classifier returns the $VM$ of ஐ /i/ or ஃ /I/ vowel for $\mathbf{v}$. Based on a detailed statistical analysis of the dot strokes and vowel modifiers of ஐ /i/ and ஃ /I/ in the IWFHR database, we come up with a set of conditions, one of which the dot stroke definitely satisfies.

**(i) Net distance covered**: When compared to the vowel modifiers of ஐ /i/ and ஃ /I/, the ratio of the Euclidean distance between the first and last points to the arc length is generally small for the dot strokes in pure consonants. This fact is captured by

$$\frac{d^v_{fl}}{l^v_T} \leq T^d_r \tag{4.6}$$

Here $d^v_{fl}$ is the Euclidean distance between the first and last sample points in $\mathbf{v}$. $l^v_T$ is the total arc length traversed along the trace. The threshold $T^d_r$ is set to the minimum possible ratio of $d^v_{fl}$ to $l^v_T$ across all modifiers of vowels ஐ /i/ and ஃ /I/.

**(ii) Relative number of sample points**: In contrast to the vowel modifiers of ஐ /i/ and ஃ /I/, the number of sample points representing the dot strokes in pure consonants is usually less.

$$\mathbf{v}_{\#} < T^d_{\#} \tag{4.7}$$

Here, $\mathbf{v}_{\#}$ corresponds to the number of sample points in the pattern $\mathbf{v}$. From Eqn 4.5, we have:

$$\mathbf{v}_{\#} = n_P - n_B \tag{4.8}$$

The value of the threshold $T^d_{\#}$ corresponds to the minimum number of sample points representing the vowel modifiers of ஐ /i/ and ஃ /I/ in the IWFHR data-set.

**(iii) Starting position of the stroke**: The $y$-coordinate value of the first sample point of dot strokes is generally higher in pure consonants than that of the vowel

modifiers of ி /i/ and ீ /I/. This observation is reflected in

$$y_1^v \geq T_{y_1}^d \tag{4.9}$$

wherein, $y_1^v$ corresponds to the $y$-coordinate of the first sample point in $\mathbf{v}$. From Eqn 4.5, we observe $y_1^v = y_{n_B+1}$. To determine the threshold $T_{y_1}^d$, the $y$-coordinate of the first sample point is recorded for all the vowel modifiers of ி /i/ and ீ /I/ in the IWFHR training data-set. The maximum of the computed values is assigned to $T_{y_1}^d$.

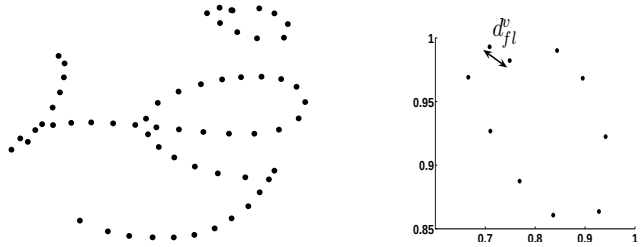
**(iv) Novel check using base consonant classifier** $C_b$: Characteristic writing styles of dot stroke, that are absent in the vowel modifiers of ி /i/ and ீ /I/, can serve as a cue for disambiguation. From experiments conducted, when dot stroke patterns with such writing styles are preprocessed (refer Sec 2.5.1) and sent to the classifier $C_b$, they get assigned to one of the base consonants ட /Ta/, ப /pa/ , ம /ma/, ய /ya/, ல /la/ or வ /va/. From statistics, we note that these base consonants do not appear as the most probable symbol for the vowel modifiers of ி /i/ and ீ /I/.

We now summarize the computation of the various thresholds with a pseudocode.

```
   Set k=0
For each CV combination of /i/ and /I/
For each training sample
Compute, from vowel modifier pattern v, the attributes
```

$$y_1^k = y_1^v$$
$$d_{fl}^k = d_{fl}^v$$
$$v_\#^k = v_\#$$
$$l_T^k = l_T^v$$

```
k++
End for
```




$(a)$ $(b)$

Fig. 4.5: Identification of a given stroke **v** as a dot. (a) Input pattern recognized as `/zhI/` by the primary classifier. (b) Extracted $VM$ stroke **v** satisfying $d^v_{fl}/l^v_T \leq 0.1$. Accordingly, the stroke **v** is assigned the label of a dot.

`End for`

$$T^d_r = \min_k(d^k_{fl}/l^k_T)$$
$$T^d_{y_1} = \max_k y^k_1$$
$$T^d_{\#} = \min_k v^k_{\#}$$

From statistics, we obtain $T^d_r = 0.1$ , $T^d_{\#} = 7$ and $T^d_{y_1} = 0.9$.

Figures 4.5 and 4.6 illustrate scenarios wherein the primary classifier wrongly assigns the patterns to CV combinations of ா /I/. However, on reevaluating the trace of the $VM$ **v**, we observe that they satisfy at least one of the conditions outlined above. Accordingly, we assign **v** to the dot stroke.

The modifier stroke in Fig. 4.7, when sent to the classifier $C_b$, gets recognized as the base consonant ப /pa/. Using condition (iv), we reevaluate it to a dot stroke.

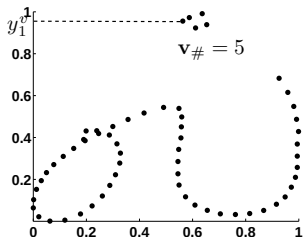
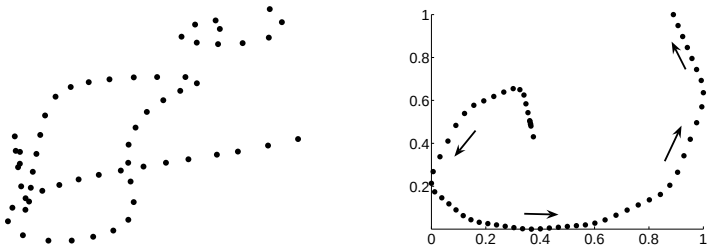



Fig. 4.6: Another example for the identification of a given stroke **v** as a dot. The primary classifier interprets the $VM$ stroke as vowel modifier of `/I/`. However, the pattern **v** satisfies $\mathbf{v}_{\#} < 7$ and $y^v_1 \geq 0.9$. Thus, on reevaluation, **v** is assigned the label of dot.




$(a)$ $(b)$

Fig. 4.7: Revaluation of $VM$ strokes using the base consonant classifier. (a) Input symbol. (b) The raw stroke $VM$ is separately preprocessed and recognized as the base consonant /pa/ by the classifier $C_b$. Hence, it is assigned the label of dot.


$(a)$ $(b)$

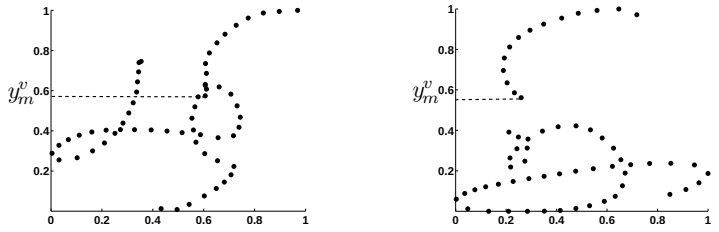
Fig. 4.8: Illustration of features $d^v_{fl}$, $\mathbf{v}_{\#}$ and $y^v_1$ for vowel modifiers of /i/ and /I/. (a)(b): VMs $\mathbf{v}$ satisfying $d^v_{fl}/l^v_T > 0.1$, $\mathbf{v}_{\#} \geq 7$ and $y^v_1 < 0.9$. For both the modifiers, $\mathbf{v}_{\#} = 20$.

Figures 4.8 (a) and (b) respectively present illustrations of the features $d^v_{fl}$, $\mathbf{v}_{\#}$ and $y^v_1$ for vowel modifiers of /i/ and /I/.

### 4.5.2 Reclassification of modifier strokes wrongly recognized as dots

We now consider the other scenario, wherein the output from the primary classifier corresponds to a pure consonant. Let $T^d_{ym}(\omega_g)$ represent the overall minimum $y$-coordinate of the BB of the dot strokes across all the samples of the pure consonant $\omega_g$ in the IWFHR data-set. The pattern $\mathbf{v}$ can be assigned to either ന or ഌ, if the condition

$$y^v_m < T^d_{ym}(\omega_g) \tag{4.10}$$




$(a)$ $(b)$

Fig. 4.9: Illustration of the reevaluation of the $VM$ stroke **v** in symbols classified as pure consonants. (a) This symbol, which is /zhi/, is wrongly recognized as /zh/ by the primary classifier. However, it is corrected by reevaluation. The minimum $y$ coordinate of the stroke **v** ($y_m^v$) is less than 0.73, the threshold for the dot stroke in pure consonant /zh/. (b) This symbol, which is /ki/, is wrongly recognized as /k/. In this case, $y_m^v$ is less than 0.64, the threshold for the dot stroke in pure consonant /k/. The thresholds for the pure consonants are read from the statistics of the IWFHR database presented in Appendix D.

holds good. Here, $y_m^v$ is computed as the minimum $y$-coordinate of the trace of **v**. For our work, we assign any such wrongly recognized pattern **v** (satisfying Eqn 4.10) to the vowel modifier of இ /i/ ( ி). Appendix D presents the overall minimum $y$-coordinate of BB of the dot strokes for each of the 23 pure consonants.

Figure 4.9 presents 2 illustrations, wherein the patterns, wrongly recognized as ழ /zh/ and க் /k/, get reevaluated to ழி /zhi/ and கி /ki/, respectively.

### 4.5.3 Reevaluation of /i/ and /I/ vowel modifiers

In this subsection, we propose the strategy for reevaluating the vowel modifiers ி and ீ. Preprocessed $x$-$y$ coordinates of the samples of vowel modifiers (in the CV combinations of இ /i/ and ஈ /I/) are used to train a 2 class SVM (denoted by $C_m$). The trace of the vowel modifier **v** (obtained from the component extractor) is assigned to ீ (the vowel modifier of ஈ /I/) whenever at least one of the following two conditions holds good.

**C1** : SVM $C_m$ favors it as the most likely vowel modifier

**C2** : The relative horizontal distance between the last sample point $x_l^v$ of the trace of

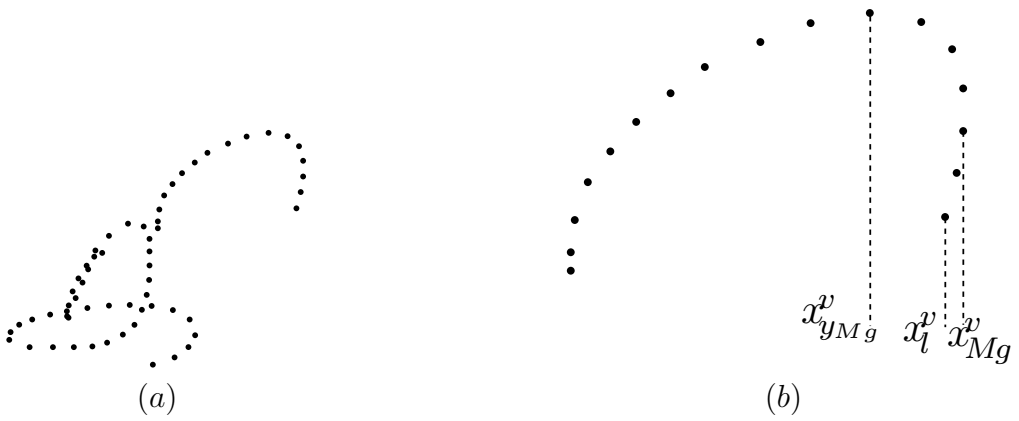



Fig. 4.10: Illustration of reevaluation of the vowel modifier **v** in CV combinations of `/i/` and `/I/`. (a) This symbol, which is `/ki/`, is wrongly recognized as `/kI/` by the primary classifier. However, it is corrected by reevaluation. (b) Extracted $VM$ stroke with the derived features.

the vowel modifier **v** to the global $x$-maximum is greater than a threshold.

$$\frac{x^v_{M,g} - x^v_l}{x^v_{M,g} - x^v_{y_{Mg}}} > T^v_o \tag{4.11}$$

Here $x^v_{M,g}$ and $x^v_l$ are the global $x$-maximum and $x$-coordinate of the last sample point of **v**, respectively. $x^v_{y_{Mg}}$ represents the $x$-coordinate corresponding to the global y-maximum of **v**. Whenever neither of the conditions are satisfied, we favor the vowel modifier of ி `/i/`. From experimental validation, we see that the threshold $T^v_o$ set to 0.2 is quite robust in discriminating ெ from ொ.

Figures 4.10 and 4.11 illustrate the proposed methodology. For the pattern in Fig. 4.10 (a), recognized as கெ `/kI/`, the conditions **C1** and **C2** do not hold good for the stroke **v** (shown in (b)). Hence, we assign it to கி `/ki/` after reevaluation.

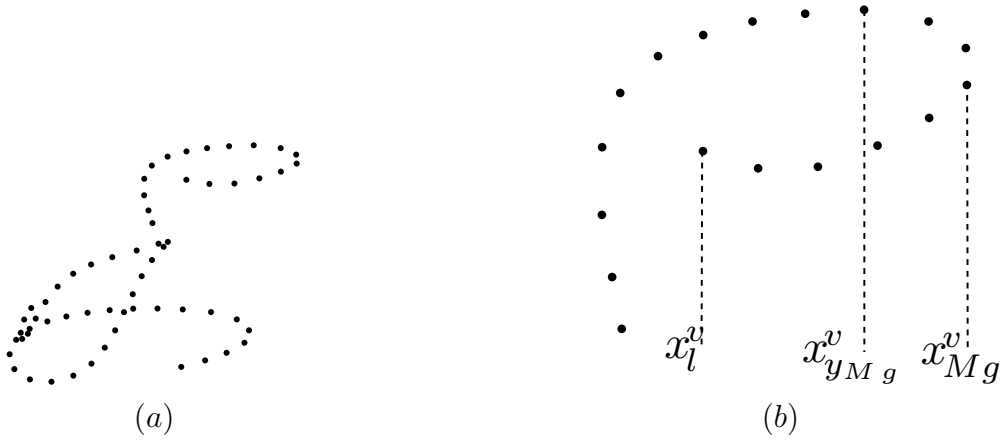



Fig. 4.11: Another example for the reevaluation of the vowel modifier **v** in CV combinations of /i/ and /I/. (a) A sample of /kI/, which gets recognized as /ki/ by the primary classifier. (b) Illustration of the features $x^v_{M,g}$ , $x^v_l$ and $x^v_{y_{Mg}}$ for the vowel modifier stroke **v**. Note that the pattern **v** gets reevaluated to the modifier of vowel /I/. Here, both the conditions C1 and C2 are satisfied.

On the other hand, the pattern in Fig. 4.11, recognized as கி /ki/ by the primary classifier, gets reevaluated to கீ /kI/. In this case, both the conditions **C1** and **C2** are satisfied for the stroke **v**. Figure 4.12 provides a high level summary of the strategies proposed to reevaluate the base consonants and vowel modifiers in CV combinations of இ /i/ and ஈ /I/ and in pure consonants.

## 4.6 Disambiguation of confused symbols

Visual inspection of confusions between symbols, arising from the primary classifier, indicates that they share common structures and are just different in some critical parts of the trace. As an example, we observe that the symbols ல /la/ and வ /va/ differ primarily in the middle of the trace. The confusion pair க /ka/ and கு /cu/ present structural differences at the end of the trace. In this section, we aim to reduce the degree of confusions between such frequently confused characters, thereby improving the overall performance, beyond that given by the primary SVM classifier alone.

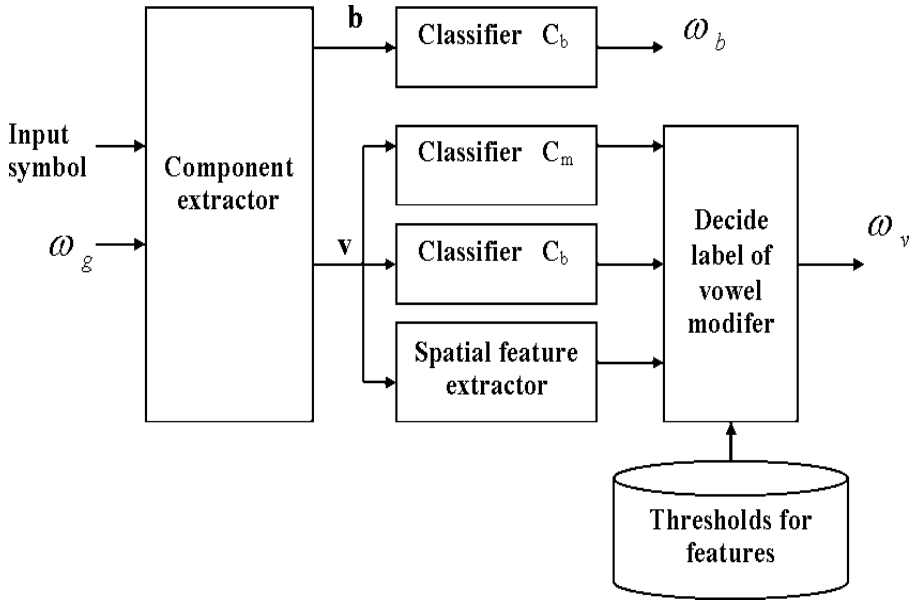



Fig. 4.12: Block diagram summarizing the proposed reevaluation techniques for base consonants and vowel modifiers. It is assumed that the symbol $\omega_g$ from the primary classifier corresponds to a pure consonant or a CV combination of `/i/` or `/I/` . $C_b$ is a classifier, trained using the samples of the 23 base consonants. The classifier $C_m$ is trained with the vowel modifiers of `/i/` and `/I/`.

### 4.6.1 Proposed methodology

Figure 4.13 presents the block diagram of the strategy proposed to disambiguate the frequently confused symbols. Independent expert networks are designed for each confusion set. Each expert comprises 3 blocks, namely, discriminative region extractor, feature extractor and SVM classifier. For each confusion pair of symbols $(c1, c2)$, the corresponding expert extracts the specific discriminative region (DR) from the input symbol pattern. The discriminative region (mathematically represented as $\Re(c1, c2)$) corresponds to the part of trace containing the finer nuances of structures in $c1$ and $c2$. A set of discriminative features is then derived from the DR $\Re(c1, c2)$ by the feature extractor module. The $i^{th}$ pair-specific feature from $\Re(c1, c2)$ is denoted by $f_i^{(c1,c2)}$. After extracting a set of features for sufficient discrimination of $(c1, c2)$, the SVM classifier is used for the disambiguation. In the current work, we propose experts labeled 1-5 (see Fig. 4.13) for resolving the ambiguities between the following confusion sets

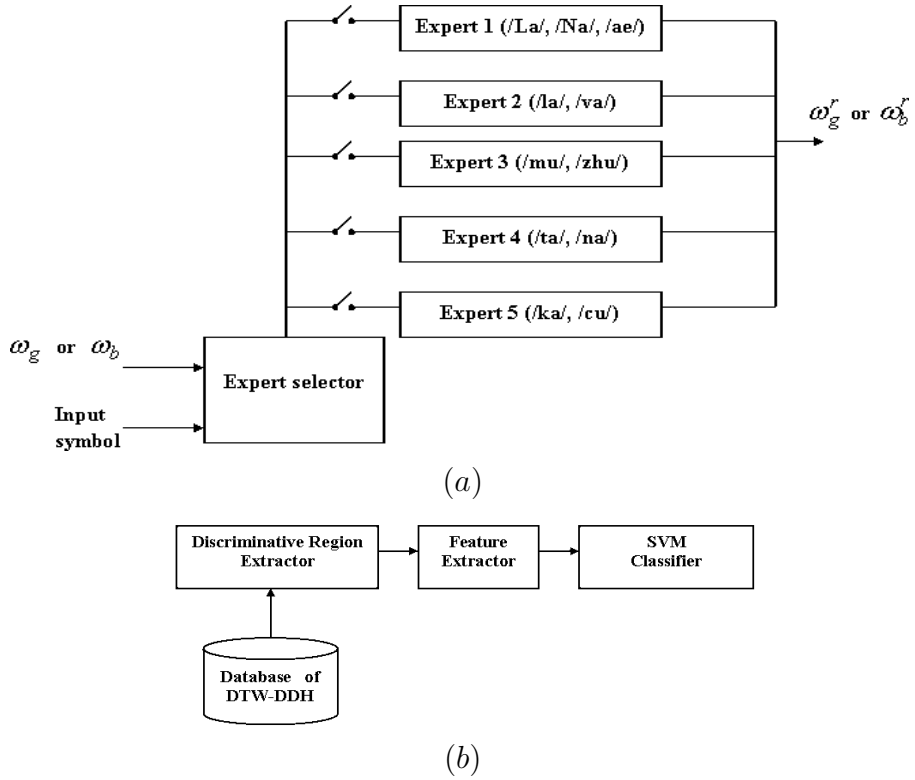



Fig. 4.13: (a) Block diagram of the proposed disambiguation strategy. Experts 1 to 5 operate on disambiguating the confused sets of (/La/, /Na/, /ai/ vowel modifier), (/la/,/va/), (/mu/,/zhu/), (/ta/,/na/) and (/ka/, /cu/), respectively. (b) Component blocks of an expert.

1. ( ன /La/, ன /Na/, ன ($VM$ of /ai/))
2. (ல /la/, வ /va/)
3. (ழ /mu/, ழ /zhu/)
4. (த /ta/, ந /na/)
5. (க /ka/, ச /cu/)

An expert selector sees one of the labels $\omega_b$ or $\omega_g$ and acts as a switch to decide on the expert to be invoked for disambiguation. In addition, depending on the input label, the selector influences the operation of the selected expert as illustrated below.

**Illustration 1**: Let us assume that the expert 1 is invoked by the selector for the input $\omega_b$. From Fig. 4.2, we observe that the label $\omega_b$ is assigned to a base consonant

whenever $\omega_g \in \{G_2, G_5, G_7\}$. Based on this knowledge, the selector allows the first expert to only disambiguate between the consonants ள /La/ and ன /Na/. However, for the scenario wherein the expert selector sees the label $\omega_g$ (that can be one of the base consonants ள /La/, ன /Na/ or the vowel modifier ை ($VM$ of /ai/)), expert 1 first disambiguates ள /La/ from ன /Na/ and then between ன /Na/ and ை ($VM$ of /ai/), if necessary.

**Illustration 2**: The expert 5 is invoked for disambiguation, if and only if the expert selector sees either க /ka/ or சு /cu/ as the label $\omega_g$.

### 4.6.2 Dynamic time warping for automated identification of discriminative regions in confused pairs

The first key step in the proposed methodology is to automatically locate the distinctive parts of strokes in similar pairs. For offline handwriting recognition, techniques have been developed to extract from images the distinctive regions relevant for classification in the second level [103, 104]. In our work, temporal information of the trace is exploited to propose a dynamic time warping (DTW) approach for learning the finer parts that distinguish the confused symbols. Prior to describing our learning methodology, we first present an over-view of the DTW technique.

Dynamic time warping (DTW) is an elastic matching technique for comparing two sequences of different lengths. Whenever the rate of progression between two patterns varies in a non-linear fashion, similarity measures such as Euclidean distance and cross-correlation are not quite effective. In such cases, temporal alignment can be carried out with dynamic programming techniques. Consider two sequences $q_1$ and $q_2$ of lengths $|q_1|$ and $|q_2|$ respectively. We first construct a $|q_1| * |q_2|$ matrix, whose $(i,j)^{th}$ element contains the cost measure of dissimilarity (denoted by $d(i,j)$) between the two points $q_1(i)$ and $q_2(j)$ . Accordingly, we refer to this matrix as the 'cost matrix'. In the cost matrix, an optimal warping path $\mathcal{W}^*$ is selected, comprising a contiguous set of matrix elements that defines a mapping between $q_1$ and $q_2$. The warping path is subjected to the constraints of boundary conditions, continuity and monotonicity [105]. The path

$\mathcal{W}^*$ for the sequence $q_1$ and $q_2$ is obtained with dynamic programming techniques. The following recurrence relation is used for computing the DTW distance between $q_1$ and $q_2$.

$$\psi(i,j) = d(i,j) + \min(\psi(i,j-1), \psi(i-1,j), \psi(i-1,j-1)) \tag{4.12}$$

where, $\psi(i,j)$ is the cumulative distance up to the current element and $d(i,j)$ is the cost measure of dissimilarity between the $i^{th}$ and $j^{th}$ points of the two sequences.

We note that the optimal path $\mathcal{W}^*$ in the cost matrix is made up of some sections with low values of $d(i,j)$ corresponding to similar regions in the confused pair of symbols and other section or sections with high values of $d(i,j)$ corresponding to the part or regions in the symbol pair that are very distinct. We utilize this property to select the discriminative regions of confused symbol pairs as described in the following subsection.

### 4.6.3 Discriminative distance histogram (DDH) for selecting the discriminative region

We generate a histogram the accumulates the pen positions that contribute to the structural differences in confused pairs $(c1, c2)$. This histogram is referred to as the 'DTW discriminative distance histogram' (DTW-DDH). Peaks in the histogram denote possible regions that could discriminate $(c1, c2)$. The training samples of IWFHR dataset is employed here. We now outline the algorithm for obtaining the DTW-DDH.

Let $(c1, c2)$ be a confused symbol pair.
$N_{Tr}^{c1}$ = no of training samples of $c1$ in the IWFHR dataset
$N_{Tr}^{c2}$ = no of training samples of $c2$ in the IWFHR dataset

Initialize a histogram that captures the pen positions corresponding to the structural differences in the pair $(c1, c2)$. In other words, set the votes for each of the $n_P$ sample indices to zero.

for each training sample of symbol $c1$
for each training sample of symbol $c2$
Compute the optimal DTW path between $i^{th}$ training sample of $c1$ and $j^{th}$ sample of $c2$
Using this path, increment the votes of the histogram for each sample index of trace, where dissimilarity exceeds a threshold $T_d$.
end
end

The threshold $T_d$ is set to 90% of the maximum dissimilarity cost encountered in the warping path. We observe that this value is sufficient for identifying the region of finer nuances in the confusion pairs.

Figure 4.14 presents the DTW-DDH obtained from the training samples of the confusion set (ன /La/, ன /Na/). The sample index corresponding to the bin having the maximum number of votes, gives rise to the maximum peak in the histogram. Around this peak, a window of samples is considered to describe the part of trace distinguishing the confusion pair $c1$ and $c2$. This, in turn, forms the discriminative region (DR) $\Re(c1, c2)$.

However, owing to different styles of writing, different transients occur at the start and end of the online trace, creating spurious peaks at the start and/or end of the DTW-DDH. For such cases, visual inspection of the confused symbols aids in selecting the region $\Re(c1, c2)$ around the right peak. From the DTW-DDH of the symbols ன /La/ and ன /Na/, we observe that the peak occurs in the middle region, thereby indicating that the discriminative region lies in the middle part of the trace.

### 4.6.4 Attributes of the discriminative region

In order to derive certain discriminative features, we first locate the various minima and maxima in the DR. For ease of reference, we define notations for these different attributes of a given DR $\Re(c1, c2)$.

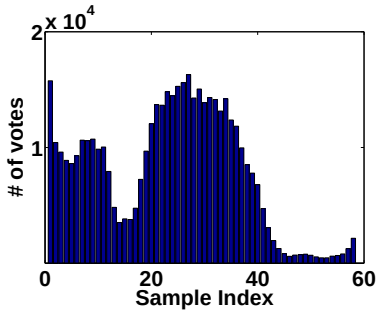



Fig. 4.14: DTW-DDH corresponding to the symbols /La/ and /Na/ obtained using their samples from IWFHR training set.

$x_{M,g}^{\Re(c1,c2)}$ - global x-maximum.
$y_{M,g}^{\Re(c1,c2)}$ - global y-maximum.
$y_{m,g}^{\Re(c1,c2)}$ - global y-minimum.
$y_{M,f}^{\Re(c1,c2)}$ -first encountered y-maximum.
$y_{M,l}^{\Re(c1,c2)}$ -last encountered y-maximum.
$y_{m,f}^{\Re(c1,c2)}$ -first encountered y-minimum.
$y_{m,l}^{\Re(c1,c2)}$ -last encountered y-minimum.
$x_{l}^{\Re(c1,c2)}$- x-coordinate of the last pen position.

If the discriminative region $\Re$ for $(c1, c2)$ appears in the middle of the trace, we denote the part of the trace preceding it by $\Re^{-}(c1, c2)$. The features outlined above can similarly be defined for this region too. In addition, specific to each $(c1, c2)$ , we define an identifiable attention point in $\Re(c1, c2)$, with respect to which the discriminative features are derived. The window of sample points centered around an attention point is referred to as the 'region of attention'.

## 4.7 Description of the various experts

In the following sub-sections, we propose techniques for disambiguating the confusion pairs on a case-by-case basis. As shown in Fig. 4.13, each confusion pair is exclusively

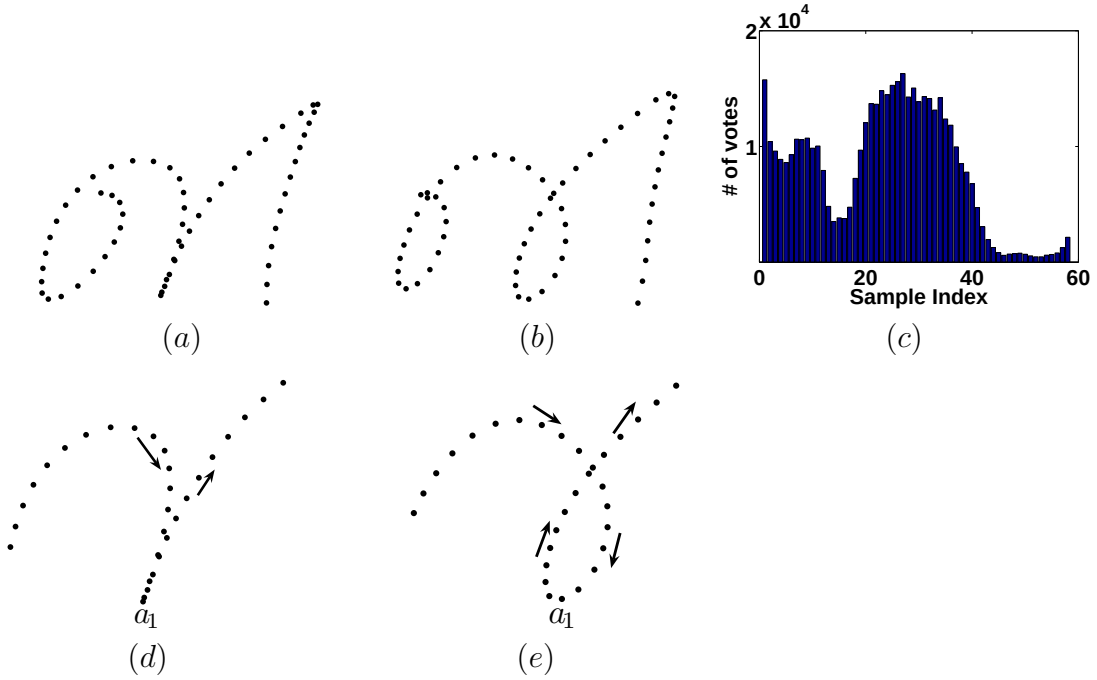



Fig. 4.15: Disambiguation of consonants /La/ and /Na/. (a) A sample of /La/. (b) A sample of /Na/. (c) DTW-DDH for this pair. (d) $\Re$ for /La/. (e) $\Re$ for /Na/. Features for discriminating these 2 consonants are derived from the region around the attention point $a_1$.

handled by a dedicated expert.

### 4.7.1 Expert 1: Consonants /La/ and /Na/

From Fig. 4.15(c), the features derived from the middle part of the trace describe the finer nuances in ள /La/ and ன /Na/. The peaks at the start of the trace in DTW-DDH are ignored since they arise due to the variations in writing styles. Accordingly, let

$$\Re(\text{ள}, \text{ன}) = \{(x_i, y_i)\}_{i=16}^{45} \tag{4.13}$$

be the DR selected by the expert 1. From the region of attention around the attention point $a_1$ in $\Re(\text{ள}, \text{ன})$, corresponding to $y_{m,f}^{\Re(\text{ள},\text{ன})}$ , the following features are defined (see Fig. 4.15 (d) and (e)).

1.
$$f_1^{(\text{ன},\text{ண})} = x_{a_1-1} - x_{a_1+1} \tag{4.14}$$

From statistics, we observe that for all samples of ண, $f_1^{(\text{ன},\text{ண})} > 0$, whereas it is not always true for samples of ன.

2. The angle between successive pen directions at $a_1$ is used as a feature

$$f_2^{(\text{ன},\text{ண})} = cos^{-1}\frac{v_1^T v_2}{\|v_1\|\|v_2\|} \tag{4.15}$$

where

$$\begin{aligned} v_1 &= (x_{a_1} - x_{a_1-1}, y_{a_1} - y_{a_1-1}) \\ v_2 &= (x_{a_1+1} - x_{a_1}, y_{a_1+1} - y_{a_1}) \end{aligned} \tag{4.16}$$

The values of $f_2^{(\text{ன},\text{ண})}$ are higher for samples of ன than for ண.

3. Consider the region of attention of size 7 centered at $a_1$. In this region, we compute three distances.

$$d_j \texttt{ = dist [(}x_{a_1-j}\texttt{,}y_{a_1-j}\texttt{)  (}x_{a_1+j}\texttt{,}y_{a_1+j}\texttt{)]} \quad \text{for j=1,2,3}$$

Accordingly, we define the feature

$$f_3^{(\text{ன},\text{ண})} = \sum_{j=1}^{3} d_j^2 \tag{4.17}$$

The values of $f_3^{(\text{ன},\text{ண})}$ are higher for ண than for ன.

### 4.7.2 Expert 1: Consonant /Na/ and vowel modifier of /ai/

DTW-DDH between the samples of the consonant ண /Na/ and ண (*VM* of /ai/) indicates that the features from the latter part of the trace can be used by expert 1 for discrimination (Fig. 4.16 (c)). Further, our visual inspection also confirms this fact.

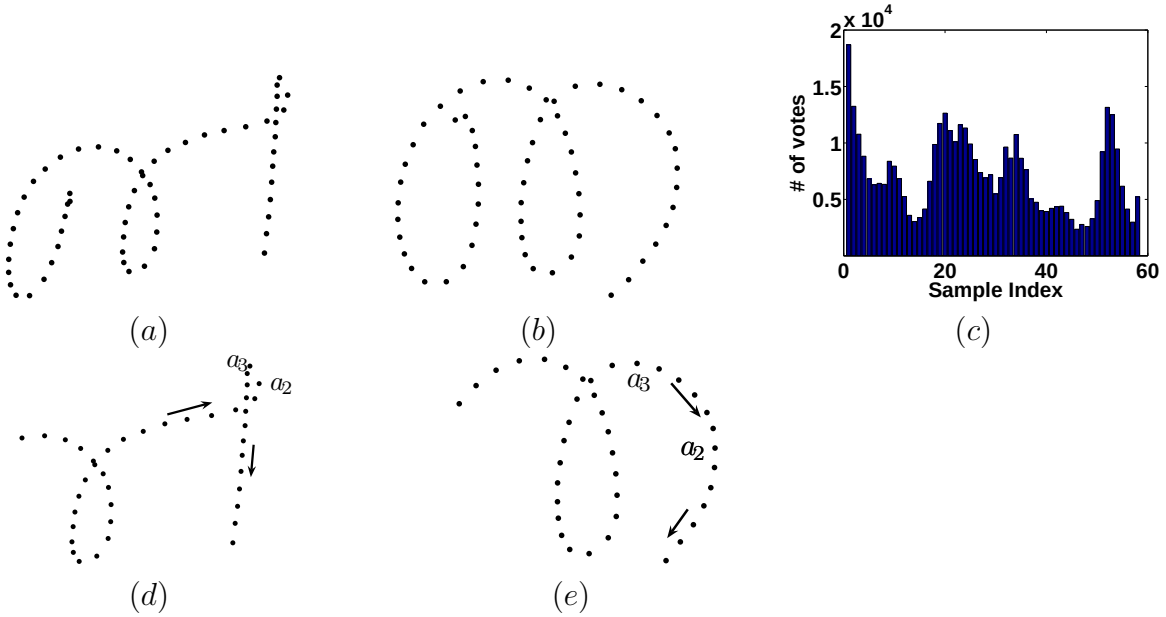



Fig. 4.16: Disambiguation of consonant /Na/ and vowel modifier of /ai/. (a) A sample of consonant /Na/. (b) A sample of vowel modifier of /ai/. (c) DTW-DDH for this pair. (d) Extracted DR $\Re$ for consonant /Na/. (e) $\Re$ for vowel modifier of /ai/. Features for discriminating these 2 symbols are derived from the attention point $a_2$ and the region of attention around $a_3$.

The peak at the start of the DTW-DDH is ignored, since this arises purely due to the different writing styles encountered at the beginning of the trace. Let the DR $\Re(\text{ன}, \text{ண})$ be described as

$$\Re(\text{ன}, \text{ண}) = \{(x_i, y_i)\}_{i=21}^{60} \tag{4.18}$$

A set of 3 features is proposed using $\Re(\text{ன}, \text{ண})$ (see Fig. 4.16 (d) and (e)) as outlined below.

1. Let the attention point $a_2$ denote the global $x$-maximum in DR, $x_{M,g}^{\Re(\text{ன},\text{ண})}$. We observe that, compared to symbol ண, the $y$-value corresponding to $x_{M,g}^{\Re(\text{ன},\text{ண})}$ is generally higher for the symbol ன. Hence we use the $y$-value as a feature $f_1^{(\text{ன},\text{ண})}$ for disambiguation.

2. To describe the features $f_2^{(\text{ன},\text{ண})}$ and $f_3^{(\text{ன},\text{ண})}$, we consider the pen position index (denoted by $a_3$) corresponding to $y_{M,l}^{\Re(\text{ன},\text{ண})}$. The angle between successive pen directions in the region of attention around $a_3$ is larger for symbol ன as compared

to symbol ன and is used for disambiguation. Accordingly, we have

$$f_2^{(\text{ன},\text{ண})} = cos^{-1}\frac{v_1^T v_2}{\|v_1\|\|v_2\|} \tag{4.19}$$

$$f_3^{(\text{ன},\text{ண})} = cos^{-1}\frac{v_2^T v_3}{\|v_2\|\|v_3\|} \tag{4.20}$$

where

$$\begin{aligned} v_1 &= (x_{a_3} - x_{a_3-1}, y_{a_3} - y_{a_3-1}) \\ v_2 &= (x_{a_3+1} - x_{a_3}, y_{a_3+1} - y_{a_3}) \\ v_3 &= (x_{a_3+2} - x_{a_3+1}, y_{a_3+2} - y_{a_3+1}) \end{aligned} \tag{4.21}$$

### 4.7.3 Expert 2: Consonants /la/ and /va/

The DTW-DDH between the consonants ல /la/ and வ /va/ is shown in Fig. 4.17 (c). We observe that the middle part of the trace primarily discriminates them. Accordingly, we select the DR as

$$\Re(\text{ல},\text{வ}) = \{(x_i, y_i)\}_{i=16}^{50} \tag{4.22}$$

The expert 2 is invoked by the selector for the disambiguation. A 4-dimensional feature vector constructed using the region of attention around attention point $a_4$ (corresponding to the first local $y$-minimum, $y_{m,f}^{\Re(\text{ல},\text{வ})}$) is robust in disambiguating the symbols (see Fig. 4.17 (d) and (e)).

1. We define the first two discriminative features as,

$$f_1^{(\text{ல},\text{வ})} = x_{a_4+1} - x_{a_4} \tag{4.23}$$

$$f_2^{(\text{ல},\text{வ})} = x_{a_4} - x_{a_4-1} \tag{4.24}$$

From statistics, $f_1^{(\text{ல},\text{வ})} > 0$ and $f_2^{(\text{ல},\text{வ})} > 0$ applies to a higher percentage of samples of symbol ல.

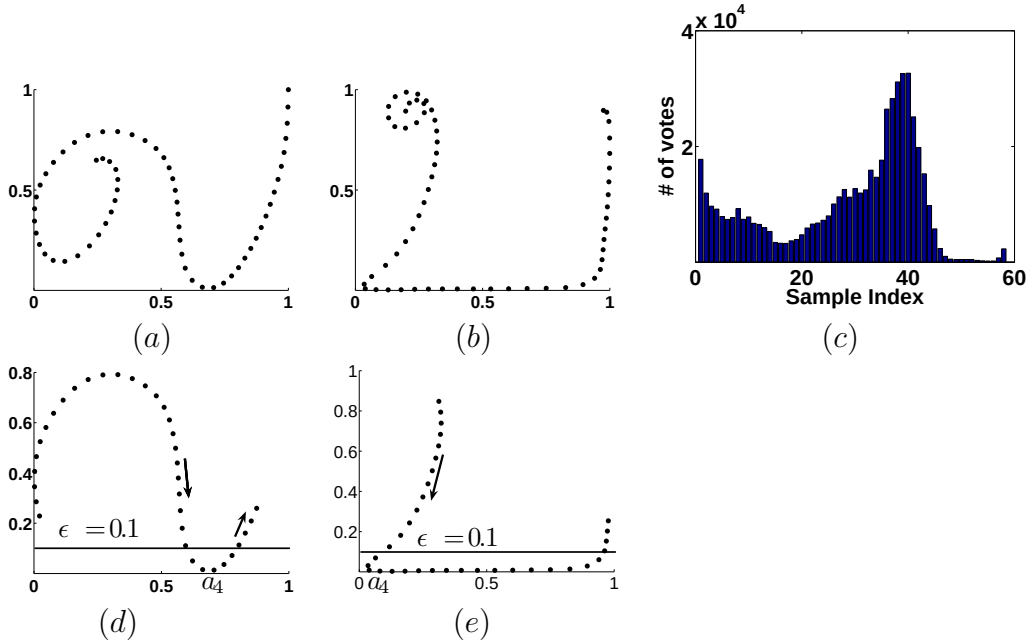



Fig. 4.17: Disambiguation of consonants /la/ and /va/. (a) A sample of /la/. (b) A sample of /va/. (c) DTW-DDH for this pair. (d) $\Re$ for /la/. (e) $\Re$ for /va/. Features for discriminating these 2 consonants are derived from the region of attention around $a_4$.

2. The angles with respect to the horizontal axes (measured in the anti-clockwise direction) made by the trace between successive pairs in $\{(x_i, y_i)\}_{i=a_4-5}^{a_4}$ are accumulated and used as a feature. Let $\Theta_i$ denote the angle made by the segment $(x_{i+1}, y_{i+1}) - (x_i, y_i)$. We define the feature

$$f_3^{(\text{ல},\text{வ})} = \sum_i \Theta_i \tag{4.25}$$

where

$$\Theta_i = tan^{-1} \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{4.26}$$

The value of $\Theta_i$ lies between $0^o$ to $360^o$. We note that $f_3^{(\text{ல},\text{வ})}$ is higher for the symbol ல than for வ.

3. We extract the part of the trace, whose $y$-coordinates lie in the range $[y_{a_4}, y_{a_4} + \epsilon]$. The variance of the $x$-coordinates in this range (higher for symbol வ than for ல) is utilized as the feature $f_4^{(\text{ல},\text{வ})}$. In order to adequately capture the discriminability

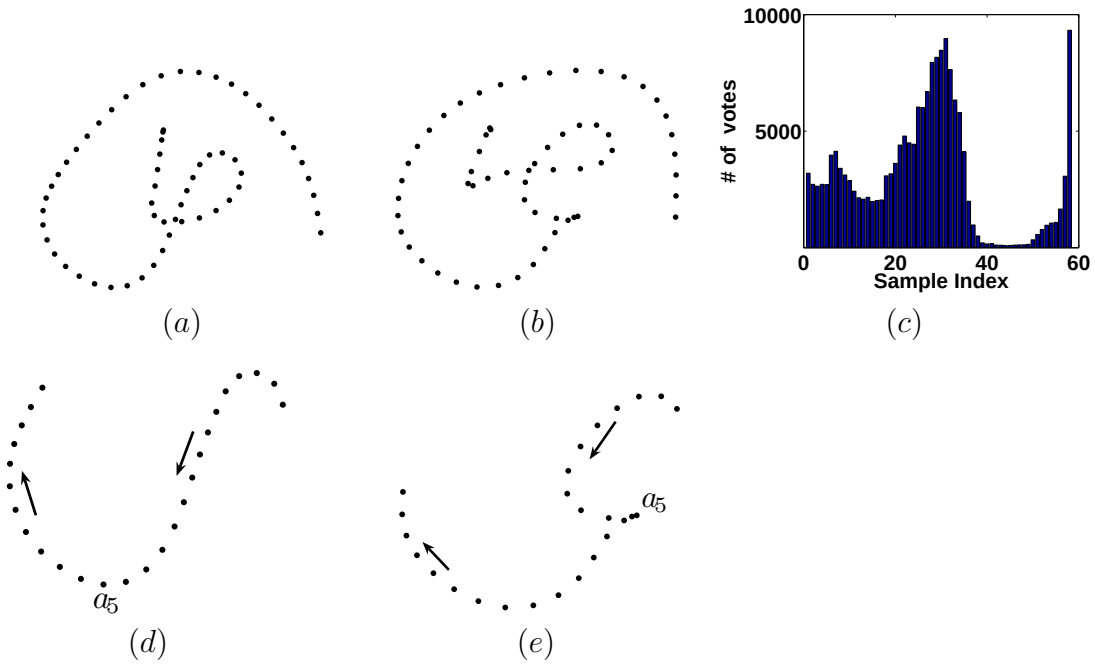



Fig. 4.18: Disambiguation of CVs /mu/ and /zhu/. (a) A sample of /mu/. (b) A sample of /zhu/. (c) DTW-DDH for this pair. (d) $\Re$ for /mu/. (e) $\Re$ for /zhu/. Features for discriminating these 2 CVs are derived in the region of attention around $a_5$.

of the variance, the value of $\epsilon$ is set to 0.1.

### 4.7.4 Expert 3: CVs /mu/ and /zhu/

Symbols ⴓ /mu/ and ⴔ /zhu/ primarily differ in the middle parts of their traces (see Fig. 4.18 (c)). Accordingly, for the expert 3, we consider the DR as,

$$\Re(\text{ⴓ},\text{ⴔ}) = \{(x_i, y_i)\}_{i=15}^{40} \tag{4.27}$$

We define a 7-dimensional feature vector in the region of attention of size 3 centered around attention point $a_5$ in $\Re(\text{ⴓ},\text{ⴔ})$ (see Fig. 4.18 (d) and (e)). Here $a_5$ corresponds to the first encountered local y minimum $y_{m,f}^{\Re(\text{ⴓ},\text{ⴔ})}$.

1. The $x$-$y$ coordinates of points in the region of attention form the feature set $\{f_i^{(\text{ⴓ},\text{ⴔ})}\}_{i=1}^{6}$. From statistics, we observe that the values of $f_i$ are relatively higher for ⴔ.

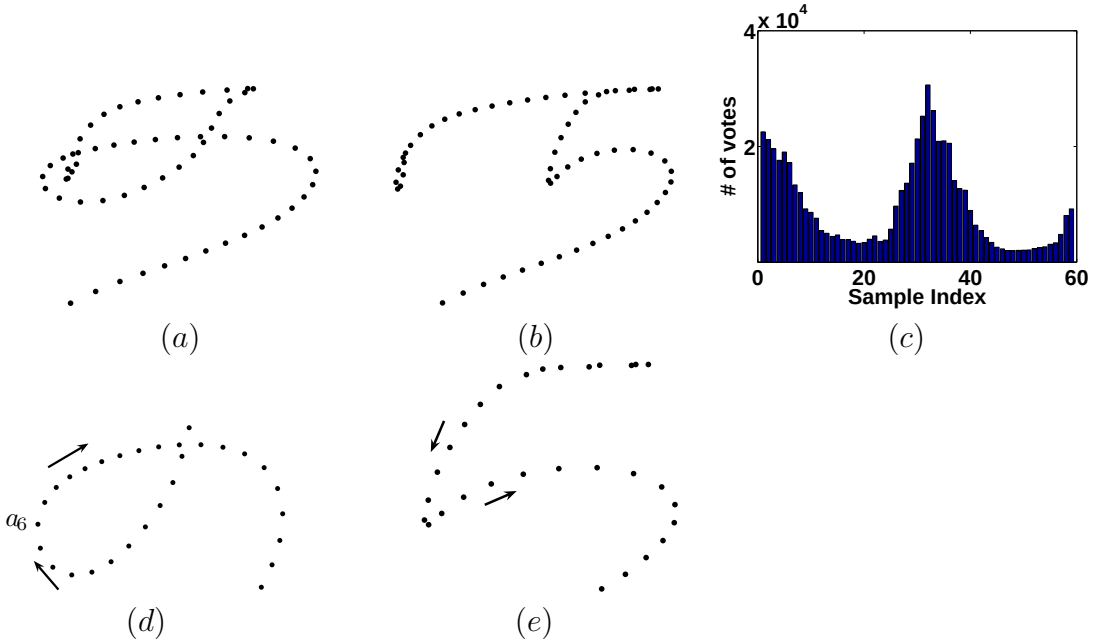



Fig. 4.19: Disambiguation of consonants /ta/ and /na/. (a) A sample of /ta/. (b) A sample of /na/. (c) DTW-DDH for this pair. (d) $\Re$ for /ta/ showing the attention point $a_6$. (e) $\Re$ for /na/. Note that this sample of /na/ does not possess a point satisfying the definition of attention point $a_6$ defined in Sec 4.7.5.

2. With respect to the global $y$- minimum coordinate of $\Re(\mathcal{Y}, \mathcal{Y})$, we define a feature

$$f_7^{(\mathcal{Y},\mathcal{Y})} = y_{a_5} - y_{m,g}^{\Re(\mathcal{Y},\mathcal{Y})} \tag{4.28}$$

For samples of $\mathcal{Y}$, $f_7^{(\mathcal{Y},\mathcal{Y})}$ is zero while for samples of $\mathcal{Y}$, it is positive.

### 4.7.5 Expert 4: Consonants /ta/ and /na/

The disambiguation of $\mathcal{Y}$ /ta/ from $\mathcal{Y}$ /na/ is performed with expert 4. From the DTW-DDH in Fig. 4.19 (c), we observe that the symbols differ significantly in the middle part of the trace. Let $\Re(\mathcal{Y}, \mathcal{Y})$ be described as

$$\Re(\mathcal{Y}, \mathcal{Y}) = \{(x_i, y_i)\}_{i=21}^{50} \tag{4.29}$$

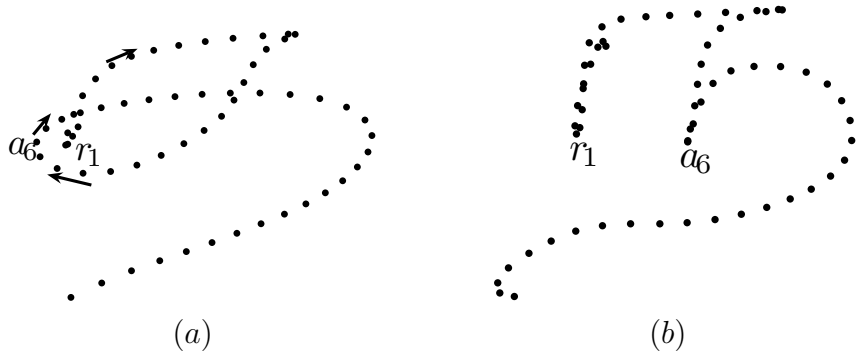



Fig. 4.20: Disambiguation of consonants /ta/ and /na/ using attention point $a_6$. (a) A sample of /ta/. (b) A sample of /na/ shown with the parameters used for computing $f_1$. Note that the attention point $a_6$ appears for both these samples.

In this DR, locate the pen position $a_6$ satisfying

$$\begin{aligned} x_{a_6} &< min(x_{a_6+1}, x_{a_6-1}) \\ y_{a_6+1} &> max(y_{a_6}, y_{a_6-1}) \end{aligned} \tag{4.30}$$

Detailed studies show that the criterion is always satisfied for ந, but it does not for some samples of ந. The absence of the structure defined in Eqn 4.30 is employed for discriminating ந from ந (Fig. 4.19 (e)).

However, the samples of (ந, ந) satisfying Eqn 4.30 still need to be disambiguated. For this, we define the horizontal distance (refer Fig. 4.20) of the attention point $a_6$ with respect to $\Re^-$(ந, ந) as

$$f_1^{(ந,ந)} = x_{a_6} - x_{r_1} \tag{4.31}$$

Here $r_1$ corresponds to $y_{m,f}^{\Re^-(ந,ந)}$ . The values of $f_1^{(ந,ந)}$ are always positive and higher for ந. However, for samples of ந, $f_1^{(ந,ந)}$ may be negative, making this feature discriminative.

### 4.7.6 Expert 5: Consonant /ka/ and CV /cu/

The DTW-DDH of Fig. 4.21 (c) indicates that symbols க /ka/ and கு /cu/ differ primarily at the end of the trace. This fact is further confirmed with our visual analysis of the confused pair. We select the last 15 points of the trace as the DR for the expert 5

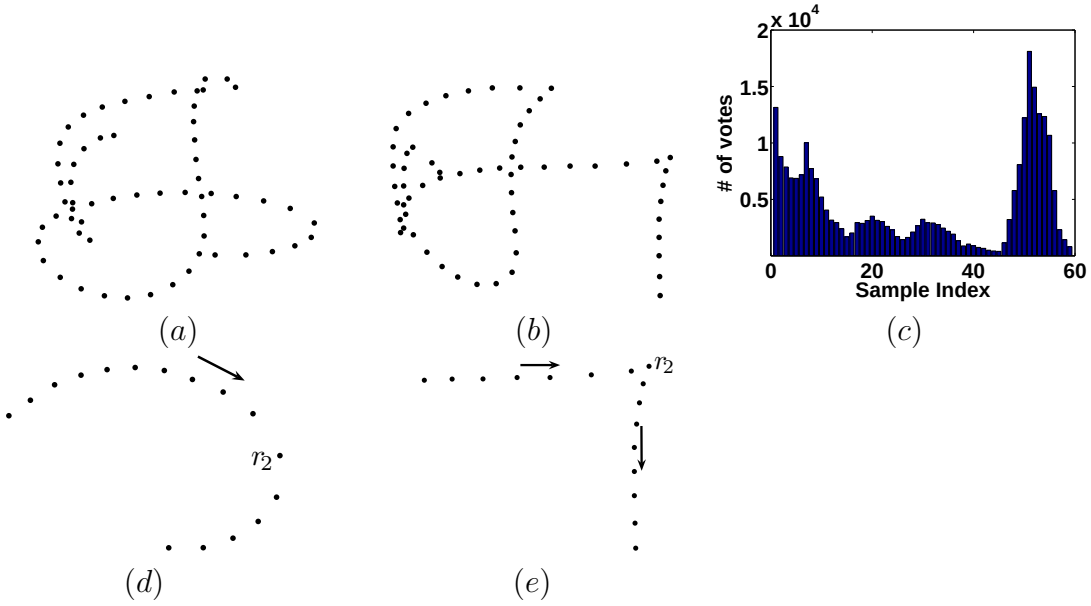



Fig. 4.21: Disambiguation between consonant `/ka/` and CV combination `/cu/`. (a) A sample of consonant `/ka/`. (b) A sample of CV combination `/cu/`. (c) DTW-DDH for this pair. (d) $\Re$ for `/ka/`. (e) $\Re$ for `/cu/` showing the attention point $r_2$.

$$\Re(\mathbf{க}, \mathbf{கூ}) = \{(x_i, y_i)\}_{i=46}^{60} \tag{4.32}$$

For disambiguating **க** and **கூ** , we compute the variance of $x$ coordinate in the segment of $\Re(\mathbf{க}, \mathbf{கூ})$ defined by $\{(x_i, y_i)\}_{i=r_2}^{60}$. Here $r_2$ denotes the sample corresponding to the global $x$ maximum of the discriminative region $x_{M,g}^{\Re(\mathbf{க},\mathbf{கூ})}$. Due to the high curvature, the value of the variance is higher for samples of **க** (Fig. 4.21 (d)). This feature is appended to the $x$-$y$ coordinates of the trace in Eqn 4.32, resulting in a 31-dimensional feature descriptor.

## 4.8 Experimental results

We evaluated the performance of the proposed reevaluation strategies on the IWFHR dataset and the MILE word database. As mentioned in Sec 4.3, the words in the MILE database are first segmented to a set of symbols with the AFS strategy, discussed in the previous chapter. Though, no restrictions were placed on the style of writing, we noted from statistics derived from the IWFHR database, that owing to the presence of the dot,

Table 4.4: Performance evaluation of the base consonant reevaluation strategy on the valid symbols of the IWFHR database.

| Group | $G_2$ | $G_5$ | $G_7$ |
|---|---|---|---|
| # of test symbols | 3990 | 3995 | 3972 |
| # of base consonants incorrectly recognized by primary classifier | 194 | 238 | 192 |
| # of errors corrected by reevaluation | 123 | 160 | 122 |
| Improvement in (%) | 63.4 | 67.3 | 63.5 |
| % of base consonants correctly recognized by primary classifier | 95.1 | 94 | 95.2 |
| % of base consonants correctly recognized by reevaluation | 98.2 | 98.0 | 98.2 |

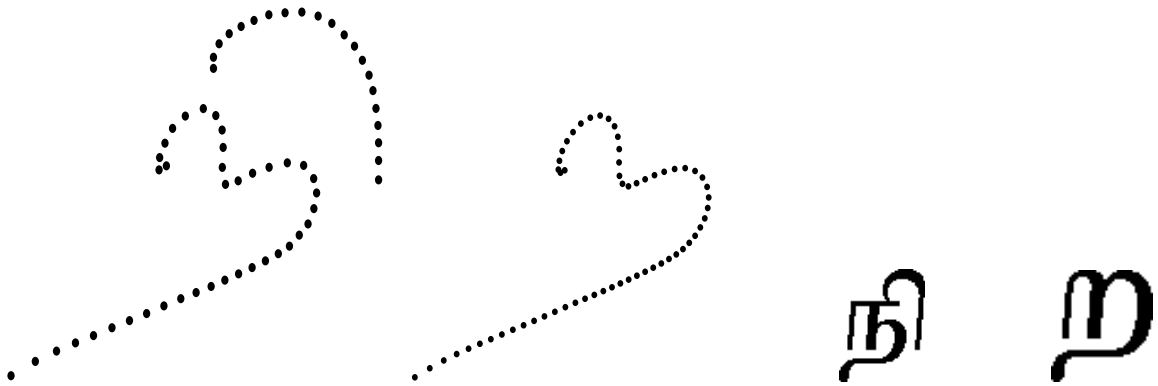
- Pure consonants necessarily had to be written with a minimum of 2 strokes.
- The vowel ி /I/ and *aytam* ஃ /ah/ require at least 3 strokes.

Such restrictions placed on the number of strokes for a given test pattern reduce the search space during recognition.

### 4.8.1 Performance evaluation on the IWFHR dataset

Each of the experiments discussed in this section focus on demonstrating the improvement in the recognition performance of the primary classifier with a proposed reevaluation technique.

As our first experiment, we reevaluate the base consonants in multi-stroke CV combinations of ஈ /i/ and ி /I/ vowels ($G_5, G_7$) and in pure consonants ($G_2$) using the strategy described in Sec 4.4. We notice that 63.4%, 67.3% and 63.5% of the errors in the base consonants have been corrected in the groups $G_2$, $G_5$ and $G_7$ respectively (Table 4.4). The errors that remain uncorrected arise mainly due to samples that appear quite ambiguous, as a result of unintelligible handwriting. Consider the test sample shown in Fig. 4.22 (a), that is ground-truthed as the symbol நீ /ni/ (displayed in (c)). We



(*a*) (*b*) (*c*) (*d*)

Fig. 4.22: Illustration of a pattern for which reevaluation of the base consonant fails. (a) This pattern, which is `/ni/` (shown in Fig (c)), gets wrongly recognized as `/Ri/`. (b) Extracted base consonant recognized as `/Ra/` (shown in Fig (d)). (c) A printed sample of `/ni/` for reference. (d) A printed sample of `/Ra/` for reference.

observe that the sharp corner of the trace has been smoothed out while writing, making this pattern to appear more like ரி `/Ri/`. The SVM corroborates our intuition by favoring the symbol ர `/Ra/` to the extracted base consonant after reevaluation, thereby giving rise to an error (refer sub-figures (b) and (d)).

The second experiment demonstrates the robustness of techniques proposed for reevaluating the stroke **v** (extracted by the component extractor). We observe from Table 4.5 that 80% of the dot strokes in pure consonants wrongly recognized by the primary SVM as the vowel modifier of ஓ `/i/` and ஈ `/I/` have been corrected by the criteria in Sec 4.5.1. This takes the correct dot recognition performance in pure consonants from 99.1% to 99.8%. On reevaluating the vowel modifiers of ஓ `/i/` and ஈ `/I/` for a given base consonant (refer Sec 4.5.3), an average of 86% of vowel modifiers wrongly recognized by the primary SVM get corrected (Table 4.6). This incidentally raises the `/i/` and `/I/` vowel modifier recognition rate from 98.1% to 99.7%.

As discussed in Sec 4.6, for a given confusion pair, a particular expert is selected to work on the class-specific features defined in the DR $\Re$ . We now proceed in demonstrating the efficacy of these features. For each of the frequently confused pairs $(c1, c2)$, two feature sets are used for the reevaluation by the selected expert. The first feature vector

Table 4.5: Impact of the dot recognition strategy on the recognition performance of pure consonants in the IWFHR database.

| Group | $G_2$ |
|---|---|
| # of test symbols | 3990 |
| # of dot strokes incorrectly recognized by primary classifier | 35 |
| # of errors corrected by reevaluation | 28 |
| Improvement (%) | 80 |
| % of dot strokes correctly recognized by primary classifier | 99.1 |
| % of dot strokes correctly recognized after reevaluation | 99.8 |

comprises the concatenated $x$-$y$ coordinates of the DR $\Re(c1, c2)$. The other feature vector is derived using the localized features for the confusion pair (as described in Sec 4.7). From the recognition accuracies in the third and fourth column of Table 4.7, we observe that, for each confusion pair, the proposed localized features perform better compared to the $x$-$y$ features, except for the pair (கி /ki/, சி /ci/), where the performance remains same. The increase in the recognition performance is significant for the symbols

(ள /La/, ன /Na/) 3.1%,
(மு /mu/, ழு /zhu/) 2.9%,
(ன /Na/, ை ($VM$ of /ai/ )) 2.3%
(ல /la/, வ /va/) 1.4%

For each of the above symbols, we compare the dimensionality of the proposed features to that of the concatenated $x$-$y$ features. As an illustration, consider the DR $\Re$(ள, ன) employed for the confusion pair ள /La/ and ன /Na/. When the $x$-$y$ coordinates of the 30 sample points in $\Re$(ள, ன) $= \{(x_i, y_i)\}_{i=16}^{45}$ (refer Sec 4.7.1) are employed, we obtain a 60 dimensional feature vector. However, extraction of the robust localized features from $\Re$(ள, ன) leads to a 3 dimensional feature vector - a 20 fold reduction in dimensionality. Moreover, this advantage is coupled with the fact that the recognition performance is improved with a lower dimension feature vector. On similar lines, one can observe that

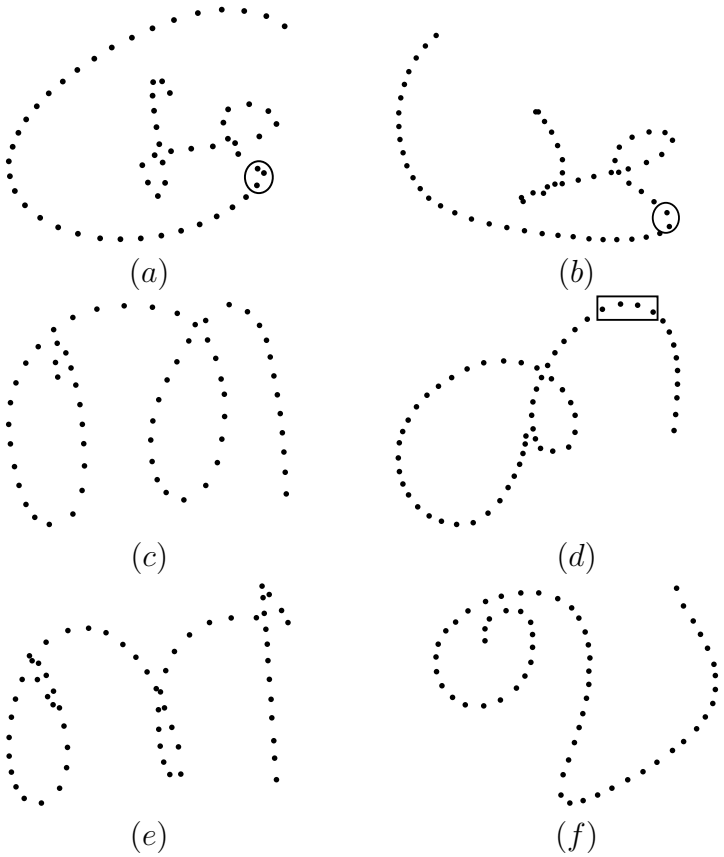
Table 4.6: Impact of the reevaluation strategy on the recognition accuracy for vowel modifiers of /i/ and /I/ in the IWFHR database.

| Group | $G_5$ | $G_7$ |
|---|---|---|
| # of test symbols | 3995 | 3972 |
| # of vowel modifiers incorrectly recognized by primary classifier | 105 | 44 |
| # of errors corrected by reevaluation | 95 | 33 |
| Improvement (%) | 90.5 | 75 |
| % of vowel modifiers correctly recognized by primary classifier | 97.3 | 98.9 |
| % of vowel modifiers correctly recognized after reevaluation | 99.7 | 99.8 |

the confusions in (பூ /mu/, புை /zhu/), (ன /Na/, ண ($VM$ of /ai/) ) and (ல /la/, வ /va/) are resolved to a greater extent by employing lower dimensional localized feature vectors. Compared to the primary classifier, the performance of disambiguating confusions is enhanced with the proposed localized features (as observed from the recognition rates in the second and fourth columns). From the fifth column, we note that more than 60% of the errors in each confusion pair have been rectified.

Table 4.8 presents the improvement in recognition of a few symbols after reevaluation. For nearly all the symbols illustrated, we observe an increase of more than 4%. Across the 26926 samples in the testing set, an accuracy of 87.9% is reported with the reevaluation strategies. Compared to the primary system, this corresponds to a 1.9% increase in recognition performance. A reduction of 13.5% in symbol recognition errors is achieved with the proposed techniques.

Figure 4.23 presents a few of the samples that were wrongly recognized by the experts. The samples in (a) and (b) represent the symbol புை /zhu/. However, the SVM trained with the proposed features in the reevaluation step favors பூ /mu/ in both the cases. In each of these samples, the attention point coincides to that of the global $y$ minimum in the DR. The part of the trace enclosed by a circle in Figs. 4.23 (a) and (b) (that describe புை /zhu/) are not captured by the proposed features, thereby leading to




| Pattern | (a) | (b) | (c) | (d) | (e) | (f) |
|---|---|---|---|---|---|---|
| Actual Symbol | ழ | ழ | ன | ன | ள | வ |
| Reevaluated symbol | ழ | ழ | ன | ண | ன | ல |

Fig. 4.23: Examples of patterns that fail to get corrected by the proposed reevaluation techniques.

Table 4.7: Illustration of the reduction in error rate on some of the confused pairs of the IWFHR database with reevaluation. The numbers are presented in terms of %.

| Confusion Pair | Primary classifier recognition rate | Disambiguation with $x$-$y$ features over $\Re$ | Disambiguation with proposed local features over $\Re$ | Improvement over primary classifier |
|---|---|---|---|---|
| (ல ,வ) (/la/, /va/) | 96.1 | 97.2 | 98.6 | 64 |
| (ள ,ன ) (/La/, /Na/) | 93.5 | 94.9 | 98 | 69 |
| (ன ,ை) (/Na/, $VM$ of /ai/) | 90.9 | 95.2 | 97.5 | 72 |
| (மு ,ழு ) (/mu/, /zhu/) | 92.6 | 95.1 | 98 | 73 |
| (க ,சு ) (/ka/, /cu/) | 94.8 | 98.7 | 99.2 | 85 |
| (த ,ந) (/ta/, /na/) | 97.9 | 97.9 | 99.2 | 62 |
| (னி,ளி) (/Ni/, /Li/) | 91.2 | 95.3 | 97.6 | 73 |
| (கி,சி) (/ki/, /ci/) | 95.2 | 98.9 | 98.9 | 77 |

the error. The sample in Fig 4.23 (c), which is ை ($VM$ of /ai/) gets wrongly recognized as ன /Na/. Figure 4.23 (d) illustrates the other scenario, wherein after reevaluation, ை ($VM$ of /ai/) is favored in place of ன /Na/. Here, we note that the trace describing the attention point (highlighted by a rectangle in Fig 4.23 (d)) of the pattern is smooth, thereby making the SVM to output the symbol ை ($VM$ of /ai/). The pattern in Fig 4.23 (e), which is ள /La/ gets reevaluated to ன /Na/. On a similar line, the pattern in Fig 4.23 (f), which is வ /va/ gets recognized as ல /la/, due to lesser value of the $x$-variance of sample points in the region around the attention point. The errors in Figs 4.23 (c) and (e) seem to arise due to the visual ambiguity of the patterns.

Apart from the primary SVM classifier, experiments were performed to demonstrate the effectiveness of the proposed techniques across different classifiers proposed in the

Table 4.8: Improvement in recognition of a few symbols in the IWFHR database with reevaluation strategies. The numbers are presented in terms of %

| Symbol | Primary classifier performance | Primary classifier+ reevaluation | Improvement |
|---|---|---|---|
| ல /la/ | 98.4 | 99.0 | 33 |
| வ /va/ | 94.9 | 98.3 | 66 |
| ள /La/ | 94.9 | 97.2 | 44.4 |
| ன /Na/ | 82.8 | 94.3 | 66.6 |
| ை ($VM$ of /ai/) | 93.8 | 97.7 | 63.6 |
| க /ka/ | 96.3 | 98.7 | 65 |
| த /ta/ | 96.8 | 98.4 | 50 |
| மு /mu/ | 90.1 | 98.3 | 83 |
| ழு /zhu/ | 95.2 | 97.6 | 50 |
| ள் /L/ | 84.2 | 95.4 | 71.5 |
| ன் /N/ | 84.6 | 97.8 | 85.7 |
| கி /ki/ | 91.0 | 98.8 | 71 |
| சி /ci/ | 85.7 | 96.7 | 76.9 |
| ரி /ri/ | 87.2 | 96.7 | 74.2 |
| னி /Ni/ | 70.2 | 83.7 | 45.3 |
| ளி /Li/ | 78.4 | 91.1 | 58.8 |
| கீ /kI/ | 85.5 | 95.2 | 66.8 |

Table 4.9: Impact of the reevaluation strategies on the recognition of symbols in the IWFHR database, when other classifiers are employed in place of SVM as the primary classifier. The numbers are presented in terms of %

| Classifier | without reevaluation | with reevaluation | Improvement |
|---|---|---|---|
| NN [18] | 76 | 80.1 | 17 |
| DTW [60] | 77.6 | 81.2 | 16 |
| HMM [65] | 83.3 | 86.5 | 19.2 |

literature for recognizing Tamil symbols (Table 4.9). We observe that, irrespective of the classifier used, an improvement is obtained in recognition performance with reevaluation.

### 4.8.2 Performance evaluation on the MILE word database

The proposed reevaluation algorithms are tested on the entire MILE word database described in Sec 2.3. A few sample words that have been correctly recognized with our algorithms are shown in Table 4.10. The erroneous symbols output from the primary classifier are highlighted with a rectangle in the third column. Appropriate strategies are invoked to correct them as described above. The dot in the last symbol of the first word is wrongly recognized by the SVM as a vowel modifier of ஈ /I/. However, it gets corrected by the reevaluation strategy in Sec 4.5.1. On the other hand, for the second word, the dot associated with the fourth symbol output from the primary classifier, gets corrected to the vowel modifier of ஐ /i/ (Sec 4.5.1). Reevaluation of base consonants (Sec 4.4) aids in rectifying the erroneous symbols in the third and fourth words. As far as the fifth word is concerned, reevaluation of the base consonant as well as disambiguation of the confusion set த /ta/ and ந /na/ play a role in correcting the error. For the last word, the disambiguation algorithm for the confusion pair ல /la/ and வ /va/ (Sec 4.7.3) is invoked to resolve the error in the third symbol. As far as the fourth symbol is concerned, both reevaluation of base consonants described in Sec 4.4 and disambiguation of (ள /La/, ன /Na/) (Sec 4.7.1) ensure that the error is corrected.

Table 4.10: Illustration of a few word samples, that have been wrongly recognized by the primary SVM classifier but corrected with reevaluation.

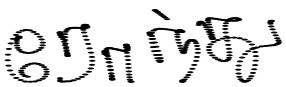
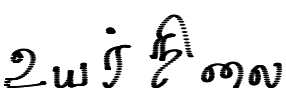
| Sl.No | Input word | primary classifier output | primary classifier+ reevaluation output |
|---|---|---|---|
| 1 |  | வீரமீ /vIramI/ | வீரம் /vIram/ |
| 2 |  | சமுத்ரம் /camuttram/ | சமுத்திரம் /cammuttiram / |
| 3 |  | குழஞ்தை /kuzhanjtai/ | குழந்தை /kuzhantai/ |
| 4 |  | ரோர்து /rOrtu/ | ரோந்து /rOntu/ |
| 5 |  | உயர்திலை /uyartilai/ | உயர்நிலை /uyarnilai/ |
| 6 |  | இரலள் /iralaL/ | இரவன் /iravaN/ |

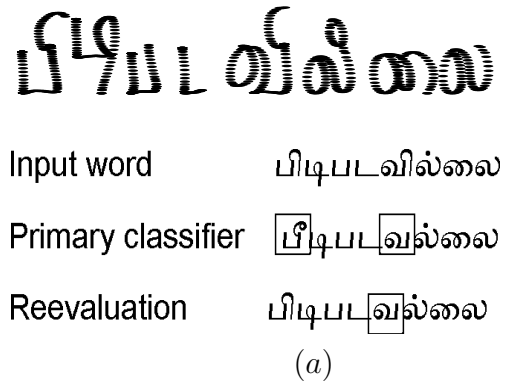
Table 4.11: Performance (in %) of the reevaluation strategies on the symbols of the MILE word database. Number of words=10000. Number of symbols=53246.

| primary classifier | primary classifier + reevaluation |
|---|---|
| 88.4 | 91.9 |

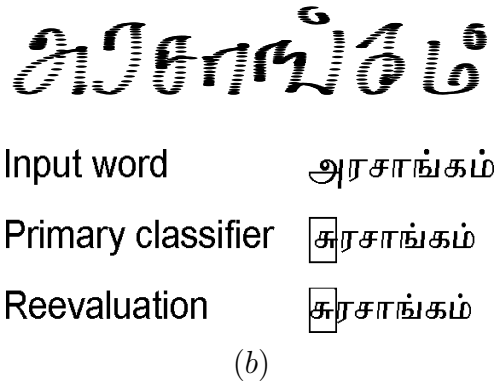
Across the 10,000 words (comprising 53246 symbols), an improvement of 3.5% is observed over the primary classifier by incorporating the various strategies (Table 4.11). Comparing the result of the symbol recognition on the MILE word database with the IWFHR data set, we observe an increase of 2.4% in the primary classifier accuracy. This difference is attributed to the fact that the words collected comprise symbols that are frequently used in modern Tamil script. In addition to these symbols, the IWFHR dataset consists of symbols that are rarely encountered.

The primary classifier may, at times, wrongly recognize symbols, written with a style infrequently encountered in the script. As an illustration, consider the word in Fig. 4.24 (a), in which the first and fifth symbols, (பி /pi/ and வி /vi/ ) are written in an unconventional style. From the output, we observe that the first symbol பீ /pI/ from the primary classifier is corrected to பி /pi/ by employing the strategy for the vowel modifiers described in Sec 4.5.3. However, the fifth symbol வி /vi/ is wrongly recognized as வ /va/ by the primary SVM classifier. The disambiguation strategy for the pair (ல /la/, வ /va/) is invoked and the output remains unchanged after this step. The reason behind this recognition error not getting corrected to வி /vi/ is attributed to the fact that the symbols (வ /va/, வி /vi/) rarely get confused by the primary classifier, and hence are not a confusion set in this work. Accordingly, there is no expert dedicated to the disambiguation of வ /va/ from வி /vi/. (refer Sec 4.6).

For the word in Fig 4.24 (b), the first symbol அ /a/ is wrongly recognized as சு /cu/ due to the specific writing style being infrequently encountered. Owing to the fact that the symbol pair (அ /a/, சு /cu/) are not part of a confusion set, there is no expert proposed to disambiguate them (refer Sec 4.6). Hence, the recognition error does not get corrected.




(a)




(b)

Fig. 4.24: Illustration of recognition errors not handled by current reevaluation strategies. (a) The first and fifth symbols in this word are written with an unconventional style. The first symbol, belonging to `/pi/` (in group $G_5$), is assigned to `/pI/` (in group $G_7$) by the primary classifier. Since the vowel modifiers of `/i/` and `/I/` of the CV combinations $G_5$ and $G_7$ get frequently confused, this error is corrected with reevaluation by employing the strategy in Sec 4.5.3. However, the fifth symbol `/vi/` (also of group $G_5$) is assigned to the base consonant `/va/` in $G_1$. Since the symbols `/vi/` and `/va/` rarely get confused with each other, they are not considered for disambiguation and hence this error is not corrected. (b) The writing style of the first symbol is quite rare. Instead of the `/a/` vowel, it is assigned to the CV combination `/cu/`. Owing to the fact that these 2 symbols rarely get confused with each other, this pair is not part of the confusion sets considered for reevaluation. In other words, the misclassified symbols in the two words are not covered by the confusion sets considered in this work.

Note that, for both the words in Figs 4.24 (a) and (b), the misclassifications encountered are not covered by the confusion sets considered.

## 4.9 Summary

In this chapter, various reevaluation strategies are proposed to reduce the error rate of the primary recognition system. In particular, with these techniques, ambiguities arising in the base consonants, pure consonants and vowel modifiers are resolved to a considerable extent. Secondly, to deal with confused pairs, a DTW approach is proposed to automatically extract their discriminative regions. Novel localized cues derived from these regions are fed to an appropriate expert for subsequent disambiguation. The proposed features are shown to be quite promising in improving the symbol recognition performance of the confusion sets. In the following chapter, we exploit the linguistic characteristics of the script for improving the recognition of words.

# Chapter 5

# Language models for Tamil word recognition


## Abstract

*This work investigates the integration of a statistical language model into the on-line Tamil recognition system in order to improve recognition of symbols in handwritten words. Two kinds of models have been considered at the symbol level: bigram and biclass models. The models are built from an extensive text corpus of 1.5 million words and experiments are carried out on the MILE word database. The use of a statistical language model is shown to improve the symbol recognition rate and the effectiveness of the different language models are compared.*

*As a second contribution, we have proposed a class reduction approach by employing a language bigram model at the akshara level during recognition. Thirdly, reevaluation techniques are proposed to correct those confusion pairs occurring at identical context, where the language model may not be quite effective due to the specific nature of Tamil. There is an improvement of up to 4.7% in the symbol level accuracy.*

## 5.1 Literature survey

The goal of a language model is to exploit the linguistic regularities and characteristics by employing probabilistic techniques on a corpus. The ideas behind incorporating linguistic knowledge in handwriting systems have been motivated from speech recognition systems [106]. Several works in offline handwriting recognition employ language models for improving the performance. A systematic comparison of the performance of unigram, bigram and trigram language models has been presented on three different corpora in [107]. The bigram model was shown to outperform the unigram model while the trigram model provides marginal improvements in word recognition rate and perplexity. In another work [108], the weight of the language model is optimized against the recognition system. The relationship between perplexity of a smoothed language model and the performance of the recognition system was investigated in [109]. A study of the impact of language models has been attempted for Chinese script in [110, 111]. In the domain of on-line recognition, language models have been proposed for sentence recognition in [112, 113, 114]. In order to improve the word recognition performance, integration of different language models have been attempted in [113, 114]. Similar to [107], a study on the influence of different language models has been conducted in [114] for online sentences.

In the context of online recognition of Indic scripts, there is hardly any work incorporating the use of language models [115]. As a first step, the present work contributes to investigating the impact of language models in improving the recognition of Tamil words. Prior linguistic knowledge has been recently employed for optical character recognition systems in Gurmukhi [99] and Malayalam [100].

## 5.2 Review of language models

The MILE text corpus (described in Sec 4.2) was utilized for generating the n-gram statistics employed in this work. The corpus essentially is a collection of sentences, wherein each word comprises a sequence of Tamil characters /aksharas. Moreover, as

detailed in Sec 2.1 and shown in Appendix B, a character may be composed of as many as 3 symbols. From the MILE text corpus, we derive the following six statistics.

- $N_T$ - Total number of occurrences of all symbols.
- $N_s(\omega_i)$ - Total number of occurrences of symbol $\omega_i$.
- $N_{ss}(\omega_i, \omega_j)$ - Total number of occurrences of the symbol pair $(\omega_i, \omega_j)$.
- $N_{cs}(c_i, \omega_j)$ - Total number of occurrences of symbol $\omega_j$ following character $c_i$.
- $N_{sc}(\omega_i, c_j)$ - Total number of occurrences of character $c_j$ following symbol $\omega_i$.
- $N_{cc}(c_i, c_j)$ - Total number of occurrences of character pair $(c_i, c_j)$.

The above statistics have been computed from the symbols and characters in each word and not across words. Here, a symbol corresponds to one of the 155 patterns listed in Appendix C and used for recognition.

Table 5.1 presents illustrations for each of the above mentioned pairs, the occurrences of which are recorded from the corpus.

A specific word $W$ can be interpreted as a realization of a discrete stochastic process. It is assumed that $W$ has been segmented to $p$ symbols, $\{S_i\}_{i=1}^{p}$, with the attention-feedback strategies discussed in Chapter 3. The feature vector corresponding to the $k^{th}$ handwritten symbol pattern is represented by $\mathbf{x}^{S_k}$. Two different models are employed to probabilistically describe the interdependencies of symbols in $W$ namely (1) n-gram language models and (2) n-class models. In addition, we assume the symbols to come from a finite vocabulary set $\mathbf{V}$ whose cardinality is 155.

Owing to the fact that Tamil does not have a finite lexicon due to its agglutinative nature (described in Sec 1.3), lexicon based spell check approaches cannot be applied for unlimited vocabulary recognition applications. Hence we take recourse to n-gram based models for detection and correction of recognition errors.

Table 5.1: Illustrative examples for the various symbol and/or character pairs. The occurrences of such pairs in the MILE text corpus are recorded to generate the linguistic statistics.

| Pair | Examples |
|---|---|
| Symbol-symbol | (ச /ca/, மு /mu/) (ப /pa/, தி /ti/)<br>(ெ ($VM$ of /o/), ந /na/) (ை ($VM$ of /ai/), த /ta/) |
| Symbol-character | (ச /ca/, கை /kai/) (ப /pa/, யா /yA/)<br>(ப /pa/, யோ /yO/) (அ /a/, கா /kA/) |
| Character-symbol | (கை /kai/, ள /La/) (யா /yA/, ரு /ru/)<br>(யோ /yO/, க /ka/) ( கா /kA/, டி /Ti/) |
| Character-character | (கை /kai/, யா /yA/) (நெ /ne/, யோ /yO/)<br>(யோ /yO/, டோ /TO/) (கா /kA/, பொ /po/) |

### 5.2.1 Statistical n-gram model

Given an online Tamil word $W$, recognized as $\{\omega_i\}_{i=1}^{p}$, we can write its probability (assuming a full order Markov process) as

$$P(W) = P(\omega_1, \omega_2.....\omega_p)$$

$$= P(\omega_1)P(\omega_2|\omega_1)P(\omega_3|\omega_1, \omega_2)....P(\omega_p|\omega_1, \omega_2...\omega_{p-1}) \tag{5.1}$$

However, it becomes very unrealistic and demanding to obtain statistics for higher order Markovian processes. In our work, we have considered only the first order Markovian dependency. For the baseline system, we assume that all the symbols are equiprobable and independent of each other. No linguistic knowledge is incorporated for the recognition of a test symbol. The baseline system in this thesis corresponds to the primary SVM classifier referred to in the earlier chapters.

Table 5.2: Frequency of occurrence of different Tamil symbols in the MILE text corpus. The occurrence ranges are expressed in terms of percentages.

| occurrence (in %) | # of symbols |
|---|---|
| 0 | 12 |
| 0-0.05 | 55 |
| 0.05-0.1 | 9 |
| 0.1-0.5 | 30 |
| 0.5-1 | 15 |
| 1-2 | 19 |
| 2-4 | 12 |
| $> 4$ | 3 |

The simplest language model called the 'unigram model' treats the symbols of a word to be independent of each other. However, the actual probability of occurrence of a symbol, as determined from the corpus, is accounted for. Using this model, we can write

$$P(W) = P(\omega_1)P(\omega_2).....P(\omega_p) \tag{5.2}$$

where

$$P(\omega_i) = \frac{N_s(\omega_i)}{N_T} \tag{5.3}$$

Table 5.2 presents the unigram statistics of the symbols in the corpus over different ranges. From the table, we observe that there are 12 symbols that are never encountered in modern day Tamil texts. These include the symbols ங் /ngi/, ஞ் /nji/, ஙீ /ngI/, ஞீ /njI/ and ங /ngu/. On the other hand, there are symbols that occur more frequently (in a text). From a practical viewpoint, it is preferable to give more weight to the recognition performance of such symbols as compared to those that rarely occur. In order to incorporate this, we propose a term 'Effective Recognition Accuracy' (ERA), defined by,

$$r_{eff} = \sum_{i=1}^{155} P(\omega_i)r(\omega_i) \tag{5.4}$$

Here $r(\omega_i)$ is the recognition rate obtained for the symbol $\omega_i$ on the test set of the IWFHR database. Essentially, ERA weighs the performance of each symbol with its

unigram probability.

In the bigram model, we assume that the probability of occurrence of a symbol in a word depends only on the immediately preceding symbol. This model incorporates a first order Markovian dependency and accordingly we can rewrite the probability of the word as

$$P(W) = P(\omega_1)P(\omega_2|\omega_1)...P(\omega_i|\omega_{i-1})...P(\omega_p|\omega_{p-1}) \tag{5.5}$$

where

$$P(\omega_i|\omega_{i-1}) = \frac{N_{ss}(\omega_{i-1}, \omega_i)}{N_s(\omega_{i-1})} \tag{5.6}$$

It is quite possible for a symbol or pair of symbols in the word to be recognized to have never occurred in the corpus [109]. In order to incorporate a non-zero probability to the bigram statistics for such symbols, we smooth the language model. The idea is to reduce the probabilities of bigrams occurring in the corpus, and redistribute this mass of probabilities among bigrams never encountered. One simple smoothing technique is to pretend each bigram occurs once more than it actually does. This is accomplished by the following updation.

$$P(\omega_j|\omega_i) = \frac{1 + N_{ss}(\omega_i, \omega_j)}{155 + N_s(\omega_i)} \tag{5.7}$$

### 5.2.2 Statistical n-class model

N-class models divide the symbols into groups [113]. In order to form meaningful groups, we club symbols that are linguistically similar and create the 8 groups ($G_1 - G_8$), outlined in Sec 3.8.2. We consider the first order Markovian dependency between the groups, wherein a Tamil symbol is assigned to exactly one group. Dedicated SVM classifiers are designed to compute the likelihood of the symbol placed in a specific group. Accordingly, one can write for a 2-class model,

$$P(\omega_i|\omega_{i-1}) = P(\omega_i|G^{\omega_i}, \mathbf{x}^{S_i})P(G^{\omega_i}|G^{\omega_{i-1}}) \tag{5.8}$$

$G^{\omega_i}$ refers to the group to which the recognized symbol $\omega_i$ belongs. The first term $P(\omega_i/G^{\omega_i}, \mathbf{x}^{S_i})$ corresponds to the likelihood (returned by the SVM classifier) for the pattern $\mathbf{x}^{S_i}$ to belong to symbol $\omega_i$ in group $G^{\omega_i}$. The second term is the prior probability of the group $G^{\omega_i}$ to occur after $G^{\omega_{i-1}}$ and can be readily derived from the corpus. One advantage of n-class models is their compactness in representation. Because symbols are combined into groups, the number of n-class probabilities is lower than that of n-grams.

## 5.3 Word recognition using symbol level language models

Let $X$ represent a sample of an online handwritten word, consisting of $p$ symbol patterns $\{S_i\}_{i=1}^{p}$. The aim of word recognition is to find the most plausible sequence of symbols $\hat{W}$ for $X$.

$$\hat{W} = \arg\max_{W} p(W|X) \tag{5.9}$$

$W$ represents the set of likely candidate symbol sequences for $X$. From Bayes rule, we can write

$$\hat{W} = \arg\max_{W} \frac{p(X|W)P(W)}{p(X)} \tag{5.10}$$

The denominator $p(X)$ is independent of $W$ and hence is ignored. $p(X|W)$ represents the likelihood of the handwritten word (as estimated from the primary SVM classifier described in Sec 2.5) for the given candidate sequence $W$. $p(W)$ is the prior probability of $W$ derived from the language model.

$$\hat{W} = \arg\max_{W} p(X|W)P(W) \tag{5.11}$$

We use the decimal logarithmic representation for the various probabilities and write

$$\hat{W} = \arg\max_{W} \ [\log_{10}(p(X|W)) + \log_{10}(P(W))] \tag{5.12}$$

The optimal sequence of symbols for the handwritten word can be traced using the well known Viterbi algorithm [116]. Assuming context-free, independent shape recognition for each pattern $S_i$ by the SVM, we can write

$$p(X|W) = \Pi_{i=1}^{p} P(\mathbf{x}^{S_i}|\omega_i) \tag{5.13}$$

The unigram (Eqn 5.2) and the bigram models (Eqn 5.5) are used to provide the estimates for $P(W)$.

### 5.3.1 Combination of reevaluation with language models

As stated in Sec 1.3, a comparative study of post processing techniques, namely reevaluation strategies and language models is not the key focus of this thesis. Instead, we propose a judicious combination of the two approaches to improve the symbol recognition performance. We provide a justification to the use of reevaluation on the output of the symbol level language model by addressing an issue, that does at times, lead to an erroneous symbol. For the current discussion, we restrict to bigram language models. Let the optimal symbol sequence of the word $\hat{W}$ from the bigram model be defined as

$$\hat{W} = \{\hat{\omega_i}\}_{i=1}^{p} \tag{5.14}$$

We consider the actual symbol sequence of the online Tamil word $W$ as

$$W = \{\omega_i\}_{i=1}^{p} \tag{5.15}$$

If the word $\hat{W}$ differs from $W$ in exactly one position (say $j$), the bigram language model favors $\hat{\omega}_j$ to $\omega_j$ whenever

$$\begin{aligned} & & \hat{\omega}_i = \omega_i \qquad i \neq j \\ P(\mathbf{x}^{S_j}|\hat{\omega}_j)P(\hat{\omega}_j|\hat{\omega}_{j-1}) & > & P(\mathbf{x}^{S_j}|\omega_j)P(\omega_j|\hat{\omega}_{j-1}) \end{aligned} \tag{5.16}$$

In other words, total dependence only on the bi-gram language model unduly favors one of the two confused symbols, given the same context. We need to rectify the symbol $\hat{\omega}_j$ to $\omega_j$. One can consider resolving the confusion by extracting a set of discriminative features from regions of the trace that differ structurally between the symbols $\hat{\omega}_j$ and $\omega_j$. In other words, we reevaluate the label of $\hat{\omega}_j$.

We invoke the reevaluation strategies discussed in Chapter 4, provided one of the conditions **C1-C3** outlined are satisfied.

**C1** : the symbols $(\omega_j, \hat{\omega}_j)$ form a confusion pair.

**C2** : the symbol $\hat{\omega}_j$ is a CV combination of இ /i/ or ஈ /I/.

**C3** : the symbol $\hat{\omega}_j$ is a pure consonant.

We illustrate here one such situation where reevaluation is necessitated, since language models cannot, by themselves, deliver. In Tamil, a verb can be modified by forms of tense, number, gender and person. Each verb results in a new word after each of these morphological changes. Considering verbs modified with gender, the ones associated with masculine gender end with the symbol ன் /N/, while those with feminine gender end with ள் /L/. Examples of such words include (வந்தான் /vantAN/, வந்தாள் /vantAL/) and (வருகிறான் /varukiRAN/ , வருகிறாள் /varukiRAL/). Note that the words in each pair differ only by the symbols ன் /N/ and ள் /L/ at the last position. Interestingly, the symbols ன் /N/ and ள் /L/ get confused with one another by the baseline classifier. All the remaining symbols of the word being the same, from Eqn. 5.16, the bigram model favors the more likely symbol of the confusion set (ன் /N/, ள் /L/) at the last position. Due to this, at times, the wrong symbol may be preferred to the correct one, resulting in an error. Therefore, reevaluation strategies are invoked to disambiguate (ன் /N/, ள் /L/) to output the right symbol.

## 5.4 Word recognition with akshara level language models

As presented in Appendix B, a Tamil character or akshara comprises 1 to 3 distinct symbols. In particular, CV combinations of the vowels அ /A/, எ /e/, ஏ /E/ and ஐ /ai/ are made up of 2 distinct symbols. CV combinations of ஒ /o/, ஓ /O/ and ஔ /au/ are written with 3 distinct symbols. We consider the symbols in a Tamil word to be drawn from the finite vocabulary $\mathbf{V} = \{\omega_k\}_{k=1}^{155}$. In this section, we propose ways in which context information (positional and bigram statistics) aids in reducing the number of symbols to be tested for an input pattern. In contrast to word recognition using the symbol-level language models (discussed in the previous section), the language model described at akshara level does not rely on the optimal Viterbi path for obtaining the output word.

- Let $\mathbf{F}_0$ represent the set of symbols that never occur at the starting position of a word in the MILE text corpus. For a pattern $S_1$, occurring at the first position in $W$, we can reduce the search space by precluding the symbols in $\mathbf{F}_0$ for recognition. We denote the subset of symbols, serving as likely candidates for the segmented pattern at the start of a word, by $\mathbf{L}_1$. Accordingly, we can write

$$\mathbf{L}_1 = \mathbf{V} \setminus \mathbf{F}_0 \tag{5.17}$$

  where $\setminus$ denotes the set difference operator.

- For the current pattern $S_i$, occurring at the $i^{th}$ position in a word ($1 < i < p$), let $\{\omega_{i-k}\}_{k=1}^{i-1}$ denote the set of recognized symbols that precede it. We present below the various context information (derived using the bigram statistics) as constraints. Symbols satisfying any of these constraints are not considered for the recognition of the current pattern. For ease of notation, let $F_i$ represent the symbols satisfying the $i^{th}$ constraint.

  1. If the immediately preceding 2 symbols correspond to a Tamil akshara $cv_1$ ,

then

$$\mathbf{F}_1 = \{\omega_j | N_{cs}(cv_1, \omega_j) = 0\} \tag{5.18}$$

2. If the immediately preceding 3 symbols correspond to a Tamil akshara $cv_2$ ,

$$\mathbf{F}_2 = \{\omega_j | N_{cs}(cv_2, \omega_j) = 0\} \tag{5.19}$$

3. If $\omega_{i-1}$ corresponds to the initial part of a CV combination $cv_3$ and $\omega_{i-2}$ is a Tamil symbol,

$$\mathbf{F}_3 = \{\omega_j | N_{sc}(\omega_{i-2}, cv_3) = 0\} \tag{5.20}$$

Here $cv_3$ is generated using the symbols $\omega_{i-1}$ and $\omega_j$.

4. If $\omega_{i-1}$ corresponds to the leading part of a CV combination $cv_5$ and symbols $\omega_{i-3}$, $\omega_{i-2}$ together form a valid Tamil akshara $cv_4$ ,

$$\mathbf{F}_4 = \{\omega_j | N_{cc}(cv_4, cv_5) = 0\} \tag{5.21}$$

$cv_5$ is generated using the symbols $\omega_{i-1}$ and $\omega_j$ and is a valid akshara.

5. If $\omega_{i-1}$ corresponds to the first part of a CV combination $cv_7$ and symbols $\omega_{i-4}$, $\omega_{i-3}$, $\omega_{i-2}$ together form a valid akshara $cv_6$,

$$\mathbf{F}_5 = \{\omega_j | N_{cc}(cv_6, cv_7) = 0\} \tag{5.22}$$

$cv_7$ is generated using the symbols $\omega_{i-1}$ and $\omega_j$ and is a valid akshara.

6. If $\omega_{i-1}$ corresponds to a Tamil symbol, then

$$\mathbf{F}_6 = \{\omega_j | N_{ss}(\omega_{i-1}, \omega_j) = 0\} \tag{5.23}$$

It is to be noted here that the symbol in $\omega_{i-1}$ alone may not necessarily represent an akshara.

The subset of symbols serving as likely candidates for the segmented pattern $S_i$

are given by

$$\mathbf{L}_i = \mathbf{V} \setminus \bigcup_{k=1}^{6} \mathbf{F}_k \tag{5.24}$$

- Apart from the contextual constraints discussed above, for a pattern $S_p$, occurring at the end of a word, we can further reduce the search space by precluding the symbols in $\mathbf{F}_7$ for recognition. Here $\mathbf{F}_7$ represents the set of symbols that never occur at the end of a word in the MILE text corpus. Accordingly, we can write,

$$\mathbf{L}_p = \mathbf{V} \setminus \bigcup_{k=1}^{7} \mathbf{F}_k \tag{5.25}$$

### 5.4.1 Illustrations of the application of akshara-level language models

We now illustrate the application of the proposed akshara-level language model for two Tamil words in a step-by-step manner. As stated earlier, by 'symbol', we refer to one of the 155 patterns listed in Appendix C. An akshara or character, on the other hand, corresponds to one of the 313 letters listed in Appendix B.

a) கியாகம் /yOkam/ (refer Table 5.3 (a))

- The pattern at the start of the word is tested with the SVM classifier against the 87 symbols in $\mathbf{L}_1$ and the most probable symbol ◌ி is assigned to it.

- For the second pattern, we use the contextual information from the previous symbol ◌ி for its recognition. We note that the symbol ◌ி is a vowel modifier of எ /E/ and is not a valid akshara/character. In order to form a valid akshara (from criteria 6), we constrain the current pattern to be recognized with the set of 15 base consonants that can follow ◌ி. Accordingly, the SVM returns symbol ய /ya/ as the most probable for this pattern.

- For the third pattern, we use the contextual prior information from the previous akshara கி /yE/ (comprising 2 symbols) for its recognition. By criteria 1, we

constrain the third pattern to be recognized only against those symbols that can follow the akshara கெ. From a set of 16 symbols, the SVM returns ர் as the most probable symbol for this pattern . However, this symbol is not a valid akshara. However, we make use of the prior knowledge that the symbol ர் always follows a base consonant and associate it to the previous akshara கெ to form another valid akshara கொ /yO/ (consonant ய /ya/ modified by the vowel ொ /O/)

- To recognize the fourth pattern, we rely on the contextual prior information from its preceding akshara கொ /yO/. The akshara கொ is made of 3 symbols. From criteria 2, we constrain the pattern to be recognized only against the 15 symbols that can follow this 3 symbol akshara. Accordingly, the SVM returns symbol க /ka/ as the most probable for this pattern. The recognized symbol க /ka/ itself is a valid akshara.

- For the recognition of the last pattern, we rely on the contextual prior information from its preceding akshara க /ka/. By constraining the pattern to a subset of symbols (76 in number) in $\mathbf{L}_p$, we obtain ம் /m/ as the most probable for this pattern from the SVM.

b) பகைமை /pakaimai/ (refer Table 5.3 (b))

- The pattern at the start of the word is tested with the SVM classifier against the 87 symbols in $\mathbf{L}_1$. and the most probable symbol ப /pa/ is assigned to it.

- For the second pattern, we constrain it to be recognized with the set of 55 symbols following ப (constraint 6). Accordingly, the SVM returns symbol ை ($VM$ of /ai/) as the most probable for this pattern. This symbol is not a valid character/akshara.

- We observe that symbol ப is a valid akshara, while ை corresponds to the first part of a CV combination (and is not a valid akshara). Accordingly, for the third pattern, from constraint 3, we constrain it to be recognized with the set of 9 symbols following பை. Based on this information, the SVM returns symbol க /ka/ as the most probable for this pattern, thereby forming a valid akshara கை /kai/.

Table 5.3: Application of the akshara-level language models on 2 Tamil words and the consequent reduction in the search space for the current pattern. For each input pattern (based on context), we show the number of symbols to be recognized against in the third column.
a) யோகம் /yOkam/

| Input pattern | Contextual information | # of symbols to be tested |
|---|---|---|
| 1 | $S_b$ | 87 |
| 2 | ே | 15 |
| 3 | ேய | 16 |
| 4 | ேயா | 15 |
| 5 | க | 76 |

b) பகைமை /pakaimai/

| Input pattern | Contextual information | # of symbols to be tested |
|---|---|---|
| 1 | $S_b$ | 87 |
| 2 | ப | 55 |
| 3 | பை | 9 |
| 4 | கை | 22 |
| 5 | கைை | 10 |

- For the fourth pattern, from constraint 1, we constrain it to be recognized with the set of 22 symbols following கை /kai/. Based on this information, the SVM returns symbol ை as the most probable for this pattern. This symbol ை is not a valid akshara.

- For the fifth pattern, from constraint 4, we constrain it to be recognized with the set of 10 symbols following கை. With this context, the SVM returns symbol ம as the most probable for this pattern. We note that the symbols, மை /mai/ together form a valid character/akshara.

It is evident from the above illustrations that we are exploring a class reduction approach with the akshara-level bigram models. In order words, the search space for a given pattern is reduced by comparing it against only a subset of the total symbol set $\mathbf{V}$.

## 5.5 Perplexity measure

One of the metrics for evaluating a language model is its perplexity [109]. For a test set $W_T$ composed of $t$ words $(W_1, W_2, ...., W_t)$ we can calculate the probability of $p(W_T)$ as the product of the probabilities of all the words in the set.

$$p(W_T) = \prod_{i=1}^{t} P(W_i) \tag{5.26}$$

In particular, given a language model that assigns probability $p(W_T)$ to the sequence of $t$ words, we can derive a compression algorithm that encodes the words $W_T$ using $-\log_2 p(W_T)$ bits. Let $N_t$ represent the total number of symbols in the $t$ words. The entropy $H$ and perplexity $P$ of a language model can be defined as

$$H = \frac{-\log_2 p(W_T)}{N_t} \tag{5.27}$$

$$P = 2^H \tag{5.28}$$

Intuitively, perplexity is regarded as the average number of symbols from which the current symbol can be chosen. In general, lower values of perplexities are achieved using higher order n-gram models.

## 5.6 Results and discussion

Prior to applying the proposed language models on Tamil words, the parameters of SVM are trained with the $x$ and $y$ coordinates of the pre-processed Tamil symbols as described in Sec 2.5. We now present the impact of the occurrence statistics on the recognition performance of symbols in the IWFHR testing database. As described in Sec 5.2.1, one can weigh the recognition rate for each symbol with its unigram probability to obtain the effective recognition accuracy (ERA). Table 5.4 lists the ERA of the primary (baseline) classifier as well as after the reevaluation step. It is interesting to note that the symbol recognition rate obtained for the 10000 words of the MILE word database (refer Table

Table 5.4: Impact of the occurrence statistics on the recognition performance on the symbols in the IWFHR database. All numbers are represented in %.

| | Primary classifier | Primary classifier +reevaluation |
|---|---|---|
| Recognition Accuracy | 86 | 87.9 |
| Effective Recognition Accuracy (ERA) | 88.1 | 91.4 |

4.11) is comparable to the ERA computed on the IWFHR testing dataset.

Table 5.5: Recognition performances of the SVM classifiers trained on the specific group of symbols ($G_1 - G_8$).

| Classifier | Group | Recognition accuracy (in %) |
|---|---|---|
| $C_b$ | $G_1$ | 95.6 |
| $C_p$ | $G_2$ | 93.5 |
| $C_o$ | $G_3$ | 98.8 |
| $C_u$ | $G_4$ | 91.2 |
| $C_i$ | $G_5$ | 95.6 |
| $C_v$ | $G_6$ | 97.3 |
| $C_I$ | $G_7$ | 95.7 |
| $C_U$ | $G_8$ | 89.7 |

### 5.6.1 Performance evaluation of word recognition with symbol-level language models

As an experimental set up for the n-class language model (described in Sec 5.2.2), a SVM is separately trained, specific to the symbols in each of the groups $G_1 - G_8$. Table 5.5 presents the details of the designed classifiers with their recognition performance on the IWFHR test set.

We now describe the structure of the word recognition system. The preprocessed $x$-$y$ coordinates (feature vector $\mathbf{x}$) of every symbol of the segmented word is input to the baseline SVM classifier, which outputs a list of $M$ (chosen as 4 in this work) candidate

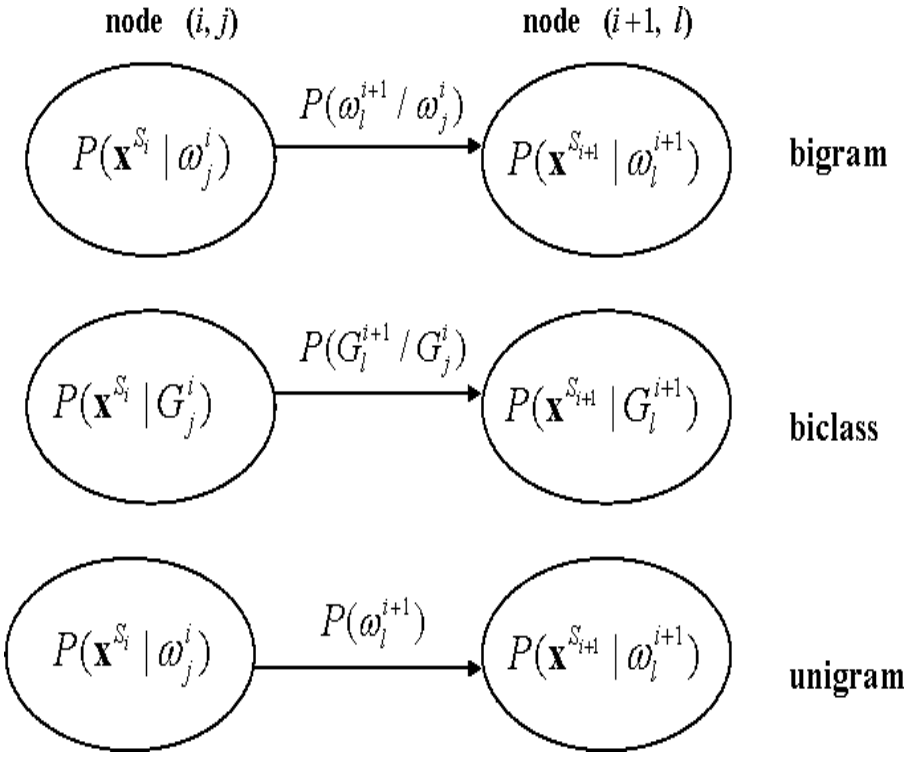



Fig. 5.1: Illustration of a pair of nodes in a word graph. The nodes represent the likelihoods of the symbol returned from the SVM classifier. The links denote the possible contextual dependence of a symbol on the previous symbol (as captured in bigrams, biclass and unigram models).

symbols ordered by their likelihoods. A word graph is then created with these choices. In that graph, $(i,j)^{th}$ node represents the likelihood $P(\mathbf{x}^{S_i}|\omega_j^i)$ of the $j^{th}$ recognized symbol for $i^{th}$ segment $S_i$. In the case of bigram models, the edge between the nodes $(i,j)$ and $(i+1,l)$ represents $P(\omega_l^{i+1}|\omega_j^i)$. For unigrams, the edges determine the prior probability $P(\omega_l^{i+1})$ in the corpus. Let $G_j^i$ represent the group containing the $j^{th}$ recognized symbol for $i^{th}$ segment. Then, for the case of biclass models, we denote the edge link by $P(G_l^{i+1}|G_j^i)$. Figure 5.1 presents a pictorial representation of a pair of nodes of a word graph.

As a first experiment, we study the impact of the n-gram and class-based language models on the handwriting recognition system. In order to incorporate the influence of linguistic knowledge, we weigh the second term of Eqn 5.12 by a factor $\beta$ (ranging

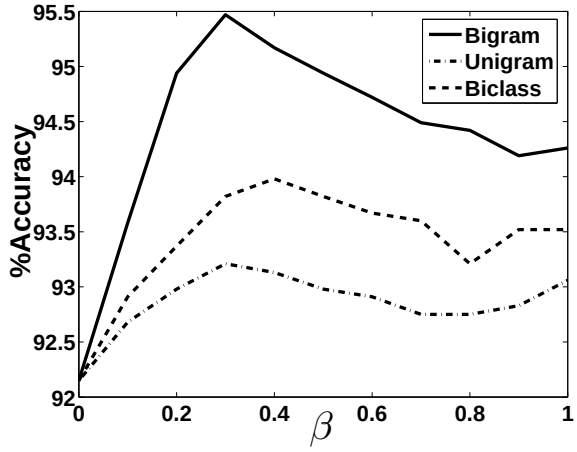



Fig. 5.2: Variation of symbol recognition accuracy obtained for different values of weight $\beta$ applied on the language models. The experiments are conducted on the validation set DB2 of 250 words.

between 0 to 1) as presented below.

$$\hat{W} = \arg\max_{W} \ [\log_{10}(p(X|W)) + \beta \log_{10}(P(W))] \tag{5.29}$$

$\beta = 0$ corresponds to baseline system, while $\beta = 1$ provides an equal weighting to both the recognition and the language model. Figure 5.2 presents the symbol recognition rate for values of $\beta$ being varied from 0 to 1 in steps of 0.1 for the validation set DB2 of 250 words. The three curves (corresponding to unigram, biclass and bigram language models) show their behavior and the optimal value of $\beta$ is 1 for the unigram model and near 0.3 for bigrams. On an average, irrespective of $\beta$, the bigram model outperforms the unigram model by 2%. Furthermore, we can see the importance of this weight since the symbol recognition rate is 94.2 % with the bigram model when $\beta = 1$ (graphical and language models have the same impact) whereas it is 95.5 % with the optimal value of $\beta$. One can also observe that the 2-class model performs lower than that of the bigram model, but better than the baseline system and unigram model. An improvement of up to 2% with respect to the baseline system is achieved.

The symbol recognition accuracies for each model is obtained across the 10000 words of the MILE word database (Table 5.6). The perplexity measures are shown in Table

Table 5.6: Performance evaluation of the different language models on the recognition of symbols in the MILE word database. (10000 words with 53246 symbols)

| Recognition system configuration | Symbol recognition accuracy (in %) |
|---|---|
| Baseline system | 88.4 |
| Unigram model | 89.8 |
| Bigram model | 92.1 |
| Bi-class model | 90.4 |
| Unigram+reevaluation | 90.9 |
| Bigram+reevaluation | 92.9 |
| Biclass model+reevaluation | 91.4 |

5.7. We notice that the bigram model outperforms the others in terms of recognition performance and has the lowest perplexity. On the other hand, the unigram model and baseline system have higher values of perplexity.

Table 5.7: Perplexity of different language models evaluated on the MILE word database.

| Recognition system configuration | Baseline | Unigram | Bigram |
|---|---|---|---|
| Perplexity | 155 | 34 | 26 |

Table 5.8: Examples of words, wrongly recognized by the baseline SVM classifier but corrected with the application of the bigram language models.

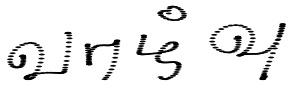
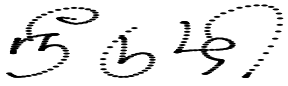
| Sl.No | Input handwritten word | Output of baseline classifier | Word recognized using bigram model |
|---|---|---|---|
| 1 |  | வரழ்வு /varazhvu/ | வாழ்வு /vAzhvu/ |
| 2 |  | கேலிக்கை /kElikkai/ | கேளிக்கை /kELikkai/ |
| 3 |  | புஸீ /pusI/ | புல் /pul/ |

Table 5.8 outlines a few sample words that have been corrected by imposing the bigram language model on the baseline SVM recognition system. The wrongly recognized symbols are highlighted by square boxes in the third column. From Table 5.6, across the 53246 symbols in the MILE word database, we notice an improvement of 3.7% (from 88.4% to 92.1%) and 1.4% (from 88.4% to 89.8%) in symbol recognition performance over the primary classifier for the bigram and unigram models.

Table 5.9 outlines a few sample words that have not been corrected by imposing the bigram language model on the baseline recognition system (refer column 3). As discussed in Sec 5.3.1, the symbol errors occur due to the optimal path chosen by the Viterbi encoding scheme, that heavily depends on the bias in the bigram statistics between adjacent symbols. However, for such scenarios, one can invoke the reevaluation strategies on the output symbols returned by the optimal Viterbi path for possible corrections (shown in column 4). For all the three words, the reevaluation of base consonants described in Sec 4.4 corrects the erroneous symbols. From Table 5.6, incorporation of the reevaluation strategies on the output from the bigram language model enhances the symbol

Table 5.9: Examples of words, wrongly recognized by the SVM classifier with language models but corrected with reevaluation.

| Sl.No | Input handwritten word | Word recognized using bigram model | Word recognized using bigram + reevaluation |
|---|---|---|---|
| 1 |  | நீடுமி /nITumi/ | நீடுழி /nITuzhi/ |
| 2 |  | காவியப் /kAviwap/ | காவியம் /kAviwam / |
| 3 |  | உடர்கட்டு /uTarkaTTu / | உடற்கட்டு /uTaRkaTTu / |

recognition from 92.1% to 92.9%. In summary, a judicious combination of reevaluation strategies with a language model improves the symbol recognition performance, beyond that provided by the language model alone.

### 5.6.2 Performance evaluation of word recognition with akshara-level language models

In this experiment, we evaluate the performance of the language models at the akshara level. On the MILE word database, incorporation of the contexts discussed in Sec 5.4 (constraints for reducing the search space of the test pattern) shows an improvement of 1.8% (from 88.4% to 90.2%) over the baseline recognition system (Table 5.10).

A drawback with incorporating akshara level language models alone leads to the possible propagation of symbol errors as depicted in the third column of Table 5.11. This is attributed to the fact that akshara-level language models make use of the contextual information provided by the immediately preceding akshara for recognition. Unlike symbol-level language models, they do not incorporate dynamic programming approaches

Table 5.10: Performance evaluation of the akshara level language models on the recognition of symbols in the MILE word database.

| Recognition system configuration | Symbol recognition accuracy (in %) |
|---|---|
| Baseline system | 88.4 |
| Akshara Bigram model | 90.2 |
| Akshara Bigram model+reevaluation | 93.1 |

like the Viterbi algorithm to obtain the optimal word. However the error propagation can be minimized to a great extent by revaluating the label of the current symbol by reevaluation strategies before proceeding to the next (fourth column of Table 5.11). The combination of language models with reevaluation improves the symbol recognition rate by 4.7% (from 88.4% to 93.1%) over the baseline system.

It is interesting to note that, with the combination of reevaluation strategies, the recognition performance of symbol-level bigram model (92.9%) and akshara-level bigram model (93.1%) on the MILE database are comparable. Moreover, akshara level language model is computationally simpler than the symbol-level bigram and biclass language model based recognition using Viterbi path.

## 5.7 Summary

In this chapter, we explored the integration of a statistical language model into the primary recognition system for improving the recognition rate of symbols in handwritten words. Two kinds of models, namely bigram and biclass models have been considered. A class reduction approach with a bigram language model at the akshara level is proposed. Finally, reevaluation techniques have been used in conjunction with language models to enhance symbol recognition performance.

Table 5.11: Examples of words, wrongly recognized by the akshara-level language model but corrected with reevaluation. Propagation of errors occurs with language models alone, as observed from the words in the third column.

| Sl.No | Input handwritten word | Word recognized using bigram | Word recognized using bigram + reevaluation |
|---|---|---|---|
| 1 |  | வீனனி /vINaNi / | வீணை /vInnai / |
| 2 |  | இருபீமது /irupImatu / | இருப்பது /iruppatu / |
| 3 |  | கற்றும் /kaRRum / | கர்வம் /karvam / |

# Chapter 6

# Conclusion and Future work

## 6.1 Summary

Research in the field of recognizing unlimited vocabulary, online handwritten Indic words is still in its infancy. In the multilingual country of India, handwriting still exists as a convenient mode for communication in government offices, rural schools and villages. In addition, a large number of forms are still being filled in Indic languages. However, most of the focus in developing online recognition systems so far has been in the area of isolated characters. In this thesis, we have attempted to develop a robust writer-independent, lexicon-free system to recognize online Tamil words.

The main contributions of the thesis can be summarized as follows:

- **Segmentation** : A novel strategy (named 'attention feedback') has been proposed for segmenting online Tamil words to the constituent symbols. Initially, the Tamil word is segmented based on a bounding box overlap criterion (DOCS step), generating a set of candidate stroke groups. Based on the degree of overlap, a stroke group at times may correspond to a part of a Tamil symbol or a merger of valid symbols. Such stroke groups are detected by providing attention to a set of proposed features (number of dominant points, dot feature, maximum bounding box to stroke displacement). In particular, dominant points and dot feature are used to select possible broken stroke groups, while the maximum bounding box to stroke

displacement serves as a cue for probable under-segmented stroke groups.

Separate generalized frameworks have been proposed in this work to correct under-segmentation and split stroke groups. In addition, as an alternative approach, linguistic knowledge has been utilized to correct over-segmented stroke groups in pure consonants, vowel /I/ and *aytam* symbol /ah/. The proposed attention feedback segmentation gives a segmentation rate of 99.7% at the symbol level for the 10000 words in the MILE word database. An improvement in symbol recognition rate from 83.9% to 88.4% is obtained with the enhanced segmentation technique.

- **Reevaluation**: A set of novel reevaluation techniques for improving the performance of the SVM classifier have been explored. These methods reduce the ambiguities in base consonants, pure consonants and vowel modifiers to a considerable extent. To learn the structural differences between similar looking symbols, a DTW approach has been proposed. Dedicated to each of the confusions, an expert (comprising a discriminative region extractor, feature extractor and SVM) is invoked for disambiguation. The proposed techniques improve the symbol recognition rate by 3.5% (from 88.4% to 91.9%) for the words in the MILE word database.

- **Language models**: Linguistic characteristics of the script have been studied using a corpus of 1.5 million Tamil words. The derived linguistic knowledge has been incorporated in the recognition system. The performance of different language models (namely symbol-level unigram, symbol-level bigram, biclass and akshara-level bigram) has been evaluated with respect to the primary SVM classifier. A judicious combination of the reevaluation techniques with language models has been proposed. On the whole, an improvement of up to 4.7% (88.4% to 93.1%) in symbol level accuracy is obtained on the MILE word database.

## 6.2 Scope for future work

The thesis has addressed two main challenges involved in designing a robust writer-independent, lexicon-free recognition system for online Tamil words. They are : (i)

segmentation of Tamil words to their constituent symbols (ii) techniques meant for improving the symbol recognition performance in the segmented words. In particular, our focus has been to explore as to how far we can proceed using prior knowledge derived with statistics, without employing a lexicon during recognition.

As a result of time constraints and resources, the proposed solutions are far from optimal for the said challenges. We mention below some challenges that can open up avenues for research in the future.

- Presently, the proposed algorithms are designed solely for Tamil symbols. Practical applications of online handwriting text recognition need to handle all Indo-Arabic numerals, besides all the common symbols such as punctuation marks, %, &, * and $. Accordingly, one can consider the inclusion of these symbols in the present symbol set and appropriately modify the proposed algorithms to address the segmentation and recognition issues in the symbols of the combined set. In particular, one can look at designing a script recognizer at the first level before attempting the segmentation problem. Alternatively, one can propose new discriminating features to adequately distinguish certain Indo-Arabic numerals such as 2 and 4 that can get readily confused with the Tamil symbols �உ /u/ and பு /pu/.

- The proposed segmentation and reevaluation algorithms tend to fail in cases where symbols are written as a different temporal sequence rarely encountered in modern Tamil script. One way to address this issue is to convert the stroke information to an offline image and then attempt recognition using offline features. Combination of online and offline features may be a good option to explore further for improving the segmentation performance. Another approach would be to identify the various writing styles of a symbol and create a separate class for each of them. However, the feasibility of such an approach needs to be considered with experimentation.

- The primary SVM classifier operates on the $x$-$y$ coordinates of the online trace. Though the features have given reasonable segmentation and recognition accuracies for Tamil symbols, attempts can be made to study the discriminative power of

different sets of features to further improve the performance of the SVM. Moreover, one can possibly explore yet another classifier with a generalization performance beyond that given by the SVM classifier.

- Currently, we have limited the linguistic context of Tamil with bigram and biclass statistics. It would be interesting to study the impact of higher order models such as trigram and triclass models in improving the recognition performance.

- In this work, we have constrained the handwritten material to online Tamil words. However, there may be scope in adapting the features and framework of the attention feedback methodology to segment words in other Indic scripts such as Kannada, Telugu and Malayalam.

- The segmentation and post-processing strategies reported in this work are not aided by a lexicon. Further improvements to the performance of word recognition can be achieved with the incorporation of a lexicon-based recognition methodology.

- Lastly, one can consider linguistic statistics at the word level to recognize paragraphs written in Tamil. However, for the feasibility of this problem, one requires to collect large amounts of data at paragraph level.

Given that work in the recognition of online Indic scripts is still in its infancy, we hope that the methodologies adopted in this thesis would serve as a benchmark to future researchers working in this field.

# Appendix A

# Some samples of the morphological changes of a verb root

வருகிறேன்
வருகிறாய்
வருகிறான்
வருகிறாள்
வருகின்றான்
வருகின்றாள்
வந்துகொண்டிருக்கிறான்
வந்துகொண்டிருக்கிறாள்
வருகிறது
வந்துகொண்டிருக்கிறது
வருகிறார்
வருகிறார்கள்
வந்துகொண்டிருக்கிறார்கள்
வந்தேன்
வந்தாய்
வந்தான்
வந்தாள்
வந்துகொண்டிருந்தேன்
வந்துகொண்டிருந்தான்
வந்துகொண்டிருந்தாள்
வந்தது
வருவாள்
வந்துகொண்டிருப்பான்
வந்துகொண்டிருப்பாள்
வரும்
வந்துகொண்டிருக்கும்
வருவார்
வருவார்கள்
வந்துகொண்டிருப்பார்கள்
வந்திருக்கிறேன்
வந்திருக்கிறான்
வந்திருக்கிறாள்
வந்துகொண்டிருந்திருக்கிறான்
வந்திருந்தேன் pp
வந்திருந்தான்
வந்திருந்தாள்
வந்திருந்தது
வந்துகொண்டிருந்தது
வந்திருந்தார்
வந்திருந்தார்கள்
வந்திருப்பேன்
வந்திருப்பான்

வந்துகொண்டிருந்தது
வந்தார்
வந்தார்கள்
வந்துகொண்டிருந்தார்
வந்துகொண்டிருந்தார்கள்
வருவேன்
வருவாய்
வருபவன்
வருபவள்
வரட்டும்
வருவது
வந்தவர்
வந்தவர்கள்
வருபவர்
வருபவர்கள்
வரப் போகிறவன்
வரப் போகிறவள்
வரவின்
வந்தபடி
வந்தபடியே
வருவி
வரவினது
வருவித்தவன
வரவேண்டும்
வரவும்
வந்தேயாகவேண்டும்
வாரும்
வாருங்கள்
வந்தவன்
வந்தவள்
வருகை
வருகையால்
வருகைக்கு
வருகையினது
வருகையே
வரவு
வரவே
வரவைவரவாள்
வந்தவனே
வந்தவளேவரவுக்கு
வருவித்தவன்
வருவித்தவர்
வருவித்தவர்கள்
வருவித்தாய்

# Appendix B

# The complete list of Tamil characters

- **Pure vowels**

  அ ஆ இ ஈ உ ஊ எ ஏ ஐ ஒ ஓ ஔ

- **Base consonants**

  க ங ச ஞ ட ண த ந ப ம
  ய ர ல வ ழ ன ற ள ஸ ஷ
  ஜ ஹ க்ஷ

- **Pure consonants**

  க் ங் ச் ஞ் ட் ண் த் ந் ப் ம் ய்
  ர் ல் வ் ழ் ன் ற் ள் ஸ் ஷ் ஜ்
  ஹ் க்ஷ்

- CV combinations of ஆ vowel

  கா ஙா சா ஞா டா ணா தா நா பா மா
  யா ரா லா வா ழா ளா றா னா ஸா ஷா
  ஹா ஜா க்ஷா

- CV combinations of இ vowel

  கி ஙி சி ஞி டி ணி தி நி பி மி
  யி ரி லி வி ழி ணி றி ளி ஸி
  ஷி ஜி ஹி க்ஷி

- CV combinations of ஈ vowel

  கீ ஙீ சீ ஞீ டீ ணீ தீ நீ பீ மீ
  யீ ரீ லீ வீ ழீ ணீ றீ ளீ ஸீ
  ஷீ ஜீ ஹீ க்ஷீ

- CV combinations of உ vowel

  கு ங்ு சு ஞு டு ணு து நு பு மு
  யு ரு லு வு ழு ணு று ளு ஸு
  ஷு ஜு ஹு க்ஷு

- **CV combinations of ஊ vowel**

கூ நூ சூ ஞூ டூ ணூ தூ
நூ பூ மூ யூ ரூ லூ வூ
ழூ ளூ றூ னூ ஸூ ஷூ
ஜூ ஹூ க்ஷூ

- **CV combinations of எ vowel**

கெ ஙெ செ ஞெ டெ ணெ தெ நெ பெ
மெ யெ ரெ லெ வெ ழெ ளெ றெ னெ
ஸெ ஷெ ஹெ ஜெ க்ஷெ

- **CV combinations of ஏ vowel**

கே ஙே சே ஞே டே ணே தே நே பே
மே யே ரே லே வே ழே ளே றே னே
ஸே ஷே ஹே ஜே க்ஷே

- **CV combinations of ஐ vowel**

கை ஙை சை ஞை டை ணை தை நை
பை மை யை ரை லை வை ழை ளை
றை னை ஸை ஷை ஹை ஜை க்ஷை

- **CV combinations of ெ vowel**

  கெ ஙெ செ ஞெ டெ ணெ தெ
  நெ பெ மெ யெ ரெ லெ வெ
  ழெ ளெ னெ ஸெ ஷெ ஹெ
  ஜெ க்ஷெ

- **CV combinations of ே vowel**

  கே ஙே சே ஞே டே ணே தே நே
  பே மே யே ரே லே வே ழே ளே
  றே னே ஸே ஷே ஹே ஜே க்ஷே

- **CV combinations of ௌ vowel**

  கௌ ஙௌ சௌ ஞௌ டௌ ணௌ
  தௌ நௌ பௌ மௌ யௌ ரௌ
  லௌ வௌ ழௌ ளௌ றௌ னௌ
  ஸௌ ஷௌ ஹௌ ஜௌ க்ஷௌ

- **Additional characters**

  ஃ ஶ்ரீ

# Appendix C

# The list of 155 Tamil symbols

- Pure vowels

  அ ஆ இ ஈ உ ஊ எ ஏ ஐ ஒ ஓ

- Base consonants

  க ங ச ஞ ட ண த ந ப ம
  ய ர ல வ ழ ன ற ள ஸ ஷ
  ஜ ஹ க்ஷ

- Pure consonants

  க் ங் ச் ஞ் ட் ண் த் ந் ப் ம் ய்
  ர் ல் வ் ழ் ன் ற் ள் ஸ் ஷ் ஜ்
  ஹ் க்ஷ்

- CV combinations of இ vowel

  கி ஙி சி ஞி டி ணி தி நி பி மி
  யி ரி லி வி ழி ளி றி னி ஸி
  ஷி ஜி ஹி க்ஷி

- CV combinations of ஈ vowel

  கீ ஙீ சீ ஞீ டீ ணீ தீ நீ பீ மீ
  யீ ரீ லீ வீ ழீ ளீ றீ னீ ஸீ
  ஷீ ஜீ ஹீ க்ஷீ

- CV combinations of உ vowel

  கு ங்ு சு ஞு டு ணு து நு பு மு
  யு ரு லு வு ழு ளு று னு ஸு
  ஷு ஜு ஹு க்ஷு

- CV combinations of ஊ vowel

  கூ ங்ூ சூ ஞூ டூ ணூ தூ
  நூ பூ மூ யூ ரூ லூ வூ
  ழூ ளூ றூ னூ ஸூ ஷூ
  ஜூ ஹூ க்ஷூ

- Additional symbols

  ஃ ஈ ே ை ஶ்ரீ

# Appendix D

# Values of the overall minimum $y$-coordinate of the dots in pure consonants

| Pure Consonant $\omega_g$ | $T^d_{ym}(\omega_g)$ | Pure Consonant $\omega_g$ | $T^d_{ym}(\omega_g)$ | Pure Consonant $\omega_g$ | $T^d_{ym}(\omega_g)$ |
|---|---|---|---|---|---|
| க் | 0.64 | ப் | 0.59 | ற் | 0.59 |
| ங் | 0.66 | ம் | 0.52 | ன் | 0.59 |
| ச் | 0.63 | ய் | 0.6 | ஸ் | 0.66 |
| ஞ் | 0.7 | ர் | 0.7 | ஷ் | 0.62 |
| ட் | 0.34 | ல் | 0.6 | ஜ் | 0.72 |
| ண் | 0.62 | வ் | 0.58 | ஹ் | 0.65 |
| த் | 0.74 | ழ் | 0.73 | க்ஷ் | 0.74 |
| ந் | 0.66 | ள் | 0.56 | | |

# Bibliography


[1] http://www.research.ibm.com/electricInk/

[2] R Plamondon, S N Srihari, Online and offline handwriting recognition: a comprehensive survey, IEEE Trans. PAMI 22(1) (2000) 63-84.

[3] S D Connell, A K Jain, Writer Adaptation for Online Handwriting Recognition, IEEE Trans. PAMI 24(3) (2002) 329-346.

[4] A Senior, K Nathan, Writer Adaptation of a HMM Handwriting Recognition System, Proc. ICASSP (1997) 1447-1450.

[5] C Tappert, C Suen, T Wakahara, State of the art in online handwriting recognition, IEEE Trans. PAMI 12(8) (1990) 787-808.

[6] C L Liu, S Jaeger, M Nakagawa, Online recognition of Chinese characters: The state-of-the-art, IEEE Trans. PAMI 26(2) (2004) 198-213.

[7] S Jaeger, C L Liu, M Nakagawa, The state of the art in Japanese online handwriting recognition compared to techniques in western handwriting recognition, IJDAR 6(2) (2003) 75-88.

[8] M A Kumar, V V Dhanalakshmi, R U Rekha, K P Soman, S Rajendran, A Novel Data Driven Algorithm for Tamil Morphological Generator, Int.J Computer Applications 6(12) (2010) 52-56.

[9] M Nakai, N Akira, H Shimodaira, S Sagayama, Substroke approach to HMM-based online Kanji handwriting recognition, Proc. ICDAR (2001) 491-495.

[10] H Shimodaira, T Sudo, M Nakai, S Sagayama, On-line overlaid-handwriting recognition based on substroke HMMs, Proc. ICDAR (2003) 1043-1047.

[11] J Tokuno, N Inami, S Matsuda, M Nakai, H Shimodaira, S Sagayama, Context dependent substroke model for HMM-based online handwriting recognition, Proc. IWFHR (2002) 78-83.

[12] S Bercu, G Lorette, On-line handwritten word recognition: an approach based on hidden Markov models, Proc. IWFHR (1993) 385-390.

[13] R Plamondon, F J Maarse, An evaluation of motor models of handwriting, IEEE Trans. SMC 19(5) (1989) 1060-1072.

[14] L Schomaker, H Teulings, A handwriting recognition system based on the properties and architectures of the human motor system, Proc. IWFHR (1990) 195-211.

[15] W Guerfali, P Plamondon, The delta lognormal theory for the generation and modeling of handwriting recognition, Proc. ICDAR (1995) 495-498.

[16] J Wang, C Wu, Y Q Xu, H Y Shum, Combining shape and physical models for online cursive handwriting synthesis, IJDAR 7(4) (2005) 219-227.

[17] S Uchida, H Sakoe, A Survey of Elastic Matching Techniques for Handwritten Character Recognition, IEICE Transactions (2005) 1781-1790.

[18] Duda, Hart, Stork, Pattern Classification, Springer Wiley, 1995.

[19] K F Chan, D Y Yeung, Elastic structural matching for online handwritten alphanumeric character recognition, Proc. ICPR (1998) 1508-1511.

[20] S D Connell, A K Jain, Template-based online character recognition, PR 34(1) (2001) 1-14.

[21] A L Koerich, R Sabourin, C Y Suen, Recognition and verification of Unconstrained Handwritten Words, IEEE Trans. PAMI 27(10) (2005) 1509-1522.

[22] L E S Oliveira, R Sabourin, F Bortolozzi, C Y Suen, Automatic Recognition of Handwritten Numerical Strings: A Recognition and verification Strategy, IEEE Trans. PAMI 24(11) (2002) 1438-1454 .

[23] A L Koerich, R Sabourin, C Y Suen, Lexicon-driven HMM decoding for large vocabulary handwriting recognition with multiple character models, IJDAR 6(2) (2003) 126-144.

[24] J Hu, M K Brown, W Turin, HMM Based On-Line Handwriting Recognition, IEEE Trans. PAMI 18(10) (1996) 1039-1045.

[25] H J Kim, K H Kim, S K Kim, J K Lee, Online recognition of handwritten Chinese characters based on hidden markov models, PR 30(9) (1997) 1489-1500.

[26] M Liwicki, H Bunke, HMM-Based On-Line Recognition of Handwritten Whiteboard Notes, Proc. IWFHR (2006) 595-599.

[27] H Bunke, M Roth, E G Talamazzini, Offline Cursive Handwriting Recognition using Hidden Markov Models, PR 28(9) (1995) 1399-1413.

[28] A Senior, K Nathan, Writer adaptation of a HMM handwriting recognition system, Proc. ICASSP (1997) 1447-1450.

[29] S Manke, U Bodenhausen, A connectionist recognizer for online cursive handwriting recognition, Proc. ICASSP (1994) 633-636.

[30] M Schenkel, I Guyon, D Henderson, On-line cursive script recognition using time delay neural networks and Hidden Markov models, Proc. ICASSP (1994) 637-640.

[31] A Namboodiri, A K Jain, Online handwritten script recognition, IEEE Trans. PAMI 26(1) (2004) 124-130.

[32] S R Kunte, S Samuel, Wavelet features based online recognition of handwritten Kannada characters, Journal Visualization Society of Japan(20) (2000) 417-420.

[33] M M Prasad, M Sukumar, A G Ramakrishnan, Divide and conquer technique in online handwritten Kannada character recognition, Proc. MOCR (2009) 1-6.

[34] R Kunwar, K Shashikiran, A G Ramakrishnan, Online Handwritten Kannada Word Recognizer with Unrestricted Vocabulary, Proc. ICFHR (2010) 611-616.

[35] M M Prasad, M Sukumar, A G Ramakrishnan, Orthogonal LDA in PCA Transformed Subspace, Proc. ICFHR (2010) 172-175.

[36] U Garain, B B Chaudhuri, T Pal, Online Handwritten Indian Script Recognition: A Human Motor Function Based Framework, Proc. ICPR (2002) 164-167.

[37] U Bhattacharya, B K Gupta, S Parui, Direction Code Based Features for Recognition of Online Handwritten Characters of Bangla, Proc. ICDAR(1) (2007) 58-62.

[38] S K Parui, K Guin, U Bhattacharya, B B Chaudhuri, Online handwritten Bangla character recognition using HMM, Proc. ICPR (2008) 1-4.

[39] T Mondal, U Bhattacharya, S K Parui, K Das, V Roy, Database generation and recognition of online handwritten Bangla characters, Proc. MOCR (2009)

[40] U Bhattacharya, A Nigam, Y S Rawat, S K Parui, An Analytic Scheme for Online Handwritten Bangla Cursive Word Recognition, Proc. ICFHR (2008) 320-325.

[41] G A Fink, S Vajda, U Bhattacharya, S K Parui, B B Chaudhuri, Online Bangla Word Recognition Using Sub-Stroke Level Features and Hidden Markov Models, Proc. ICFHR (2010) 393-398.

[42] M Srinivas Rao, Gowrishankar, V S Chakravarthy, Online Recognition of Handwritten Telugu Characters, Proc. of the International Conference on Universal Knowledge (2002)
http://www.cfilt.iitb.ac.in/icukl2002/papers/indexofpapers.html

[43] P V S Rao, T M Ajitha, Telugu script recognition A feature based approach, Proc. ICDAR (1995) 323-326.

[44] J Babu, L Prasanth, R R Sharma, G V Prabhakara Rao, A Bharath, HMM-based Online Handwriting Recognition System for Telugu Symbols, Proc. ICDAR (2007) 63-67.

[45] S Jaeger, S Manke, J Reichert, A Waibel, Online handwriting recognition: the NPen++ recognizer, IJDAR 3(3) (2001) 169-180.

[46] A Jayaraman, C C Sekhar, V S Chakravarthy, Modular Approach to Recognition of Strokes in Telugu Script, Proc. ICDAR (2007) 501-505.

[47] L Prasanth, J Babu, R Sharma, P Rao, M Dinesh, Elastic Matching of Online Handwritten Tamil and Telugu Scripts Using Local Features, Proc. ICDAR (2007) 1028-1032.

[48] S Connell, R Sinha, A Jain, Recognition of unconstrained online Devanagari characters, Proc. ICPR (2000) 368-371.

[49] J Kumar, V S Chakravarthy, Designing an optimal Classifier Ensemble for online character recognition using Genetic Algorithms, Proc. ICFHR (2008) 1028-1032.

[50] H Swethalakshmi, C Chandra Sekhar, V S Chakravarthy, Spatiostructural Features for Recognition of Online Handwritten Characters in Devanagari and Tamil Scripts, Proc. ICANN (2) (2007) 230-239.

[51] N Joshi, G Sita, A G Ramakrishnan, V Deepu, S Madhvanath, Machine Recognition of Online Handwritten Devanagari Characters, Proc. ICDAR (2005) 1156-1160.

[52] A Bharath, V Deepu, S Madhvanath, An Approach to Identify Unique Styles in Online Handwriting Recognition, Proc. ICDAR (2005) 775-779.

[53] A Bharath, S Madhvanath, A framework based on semi-supervised clustering for discovering unique writing styles, Proc. ICDAR (2009) 891-895.

[54] A K Sharma, R K Sharma, Online Handwritten Gurmukhi Character Recognition Using Elastic Matching, Proc. CISP (2008) 391-396.

[55] A K Sharma, R Kumar, R K Sharma, Rearrangement of Recognized Strokes in Online Handwritten Gurmukhi Words Recognition, Proc. ICDAR (2009) 1241-1245.

[56] G Shankar, V Anoop, V S Chakravarthy, LEKHAK [MAL]: A System for Online Recognition of Handwritten Malayalam Characters, Proc. NCC (2003) 463-467.

[57] A Arora, A M Namboodiri, A Hybrid Model for Recognition of Online Handwriting in Indian Scripts, Proc. ICFHR (2010) 433-439.

[58] C S Sundaresan, S S Keerthi, A study of representations for pen based handwriting recognition of Tamil characters, Proc. ICDAR (1999) 422-425.

[59] A H Toselli, M Pastor, E Vidal, On-line handwriting recognition system for Tamil handwritten characters, Proc. PRIA (2007) 370-377.

[60] N Joshi, G Sita, A G Ramakrishnan, S Madhvanath, Comparison of Elastic Matching Algorithms for Online Tamil Handwritten Character Recognition, Proc. IWFHR (2004) 444-449.

[61] V Deepu, S Madhvanath, A G Ramakrishnan, Principal Component Analysis for Online Handwritten Character Recognition, Proc. ICPR (2004) 327-330.

[62] B S Raghavendra, C K Narayanan, G Sita, A G Ramakrishnan, M Sriganesh, Prototype Learning Methods for Online Handwriting Recognition, Proc. ICDAR (2005) 287-291.

[63] R Niels, L Vuurpijl, Dynamic Time Warping Applied to Tamil Character Recognition, Proc. ICDAR (2005) 730-734.

[64] K H Aparna, V Subramanian, M Kasirajan, G V Prakash, V S Chakravarthy, S Madhvanath, Online Handwriting Recognition for Tamil, Proc. IWFHR (2004) 438-443.

[65] S Kiran, K S Prasad, R Kunwar, A G Ramakrishnan, Comparison of HMM and SDTW for Tamil Handwritten Character Recognition, Proc. SPCOM (2010) 1-4.

[66] A Bharath, S Madhvanath, Hidden Markov Models for Online Handwritten Tamil Word Recognition, Proc. ICDAR (2007) 506-510.

[67] B Nethravathi, C P Archana, K Shashikiran, A G Ramakrishnan, V Kumar, Creation of a huge annotated database for Tamil and Kannada OHR, Proc. IWFHR (2010) 415-420.

[68] Isolated Tamil Handwritten Character Dataset www.hpl.hp.com/india/research/penhw-interfaces-1linguistics.html

[69] J C Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery (2) (1998) 121-167.

[70] LIBSVM - A Library for Support Vector Machines http://www.csie.ntu.edu.tw/ cjlin/libsvm/

[71] F Camastra, A SVM-based cursive character recognizer, PR 40(12) (2007) 3721-3727.

[72] C L Liu, H Sako, H Fujisawa, Effects of Classifier Structures and Training Regimes on Integrated Segmentation and Recognition of Handwritten Numeral Strings, IEEE Trans. PAMI 26(11) (2004) 1395-1407.

[73] A W Senior, A J Robinson, An Off-Line Cursive Handwriting Recognition System, IEEE Trans. PAMI 20(3) (1998) 309-321.

[74] U V Marti, H Bunke, Using a Statistical Language Model to Improve the Performance of an HMM-Based Cursive Handwriting Recognition System, IJPRAI 15(1) (2002) 65-90.

[75] S Madhvanath, V Govindaraju, The Role of Holistic Paradigms in Handwritten Word Recognition, IEEE Trans. PAMI 23(2) (2001) 149-164.

[76] Murase H, Online recognition of free-format Japanese handwritings, Proc. ICPR (1988) 1143-1147.

[77] M Nagakawa, B Zhu, M Onuma, A model of online handwritten Japanese text recognition free from line direction and writing format constraints, IECIE Trans on Info. and Sys (2005) 1815-1822.

[78] B Zhu, X D Zhou, C L Liu, M Nagakawa, A robust model for online handwritten Japanese text recognition, IJDAR 13(2) (2010) 121-131.

[79] T Fukushima, M Nakagawa, On-Line Writing-Box-Free Recognition of Handwritten Japanese Text Considering Character Size Variations, Proc. ICPR (2000) 2359-2363.

[80] X D Zhou, J L Yu, C L Liu, T Nagasaki, K Marukawa, Online handwritten Japanese character string recognition incorporating geometric context, Proc. ICDAR (2007) 48-52.

[81] X Gao, P M Lallican, C Viard-Gaudin, A Two-stage Online Handwritten Chinese Character Segmentation Algorithm Based on Dynamic Programming, Proc. ICDAR (2005) 735-739.

[82] S Y Zhao, Z R Chi, P F Shi, Two-stage segmentation of unconstrained handwritten Chinese characters, PR 36(1) (2003) 145-156.

[83] N Furukawa, J Tokuno, H Ikeda, Online Character Segmentation Method for Unconstrained Handwriting Strings Using Off-stroke Features, Proc. ICFHR (2006) 361-366.

[84] B Zhu, M Nakagawa, Segmentation of On-Line Freely Written Japanese Text Using SVM for Improving Text Recognition, IECIE Trans on Info. and Sys (2006) 1-8.

[85] X D Zhou, C L Liu, M Nakagawa, Online Handwritten Japanese Character String Recognition Using Conditional Random Fields, Proc. ICDAR (2009) 521-525.

[86] Y Tonouchi, Path Evaluation and Character Classifier Training on Integrated Segmentation and Recognition of Online Handwritten Japanese Character String, Proc. ICFHR (2010) 513-517.

[87] S Sundaram, A G Ramakrishnan, Attention feedback based robust segmentation of online handwritten words. Indian Patent Office Reference. No: 03974/CHE/2010.

[88] N Tripathy, U Pal, Handwriting Segmentation of Unconstrained Oriya Text, Proc. IWFHR (2004) 306-311.

[89] A Bishnu, B B Chaudhuri, Segmentation of Bangla handwritten text into characters by recursive contour following, Proc. ICDAR (1999) 402-405.

[90] S Basu, R Sarkar, N Das, M Kundu, M Nasipuri, D K Basu, A Fuzzy Technique for Segmentation of Handwritten Bangla Word Images, Proc. ICCTA (2007) 427-433.

[91] M Cheriet, N Kharma, C L Liu, C Y Suen, Character Recognition Systems: A Guide for Students and Practitioners, Wiley, 2008.

[92] G M Boynton, Attention and visual perception, Current Opinion in Neurobiology(15) (2005) 465-469.

[93] A M Sillito, H E Jones, Corticothalamic interactions in the transfer of visual information, Philos Trans R Soc Lond B Biol Sci. (2002) 1739-1752.

[94] L Vuurpijl, L Schomaker, M Van Erp, Architectures for Detecting and Solving Conflicts: Two-Stage Classification and Support Vector Classifiers, IJDAR 5(4) (2003) 213-223.

[95] A Bellili, M Gilloux, P Gallinari, An MLP-SVM combination architecture for offline handwritten digit recognition, IJDAR 5(4) (2003) 244-252.

[96] L Prevost, L Oudot, A Moises, C Michel-Sendis, M Milgram, Hybrid generative/discriminative classifier for unconstrained character recognition, PRL 26(12) (2005) 1840-1848.

[97] A Alaei, P Nagabhushan, U Pal, Fine Classification of Unconstrained Handwritten Persian/Arabic Numerals by Removing Confusion amongst Similar Classes, Proc. ICDAR (2009) 601-605.

[98] D V Sharma, G S Lehal, S Mehta, Shape Encoded Post Processing of Gurmukhi OCR, Proc. ICDAR (2009) 788-792.

[99] G S Lehal, C Singh, A Post Processor for Gurmukhi OCR, SADHANA 27(1) (2002) 99-112.

[100] K Nair, C V Jawahar, A Post-Processing Scheme for Malayalam using Statistical Sub-character Language Models, Proc. DAS (2010) 363-370.

[101] B B Chaudhuri, U Pal, OCR error detection and correction of an inflectional Indian language script, Proc. ICPR(3) (1996) 245-249.

[102] D Navon, Forest Before Trees: The Precedence of Global Features in Visual Perception, Cognit Psychol 9 (1977) 353-383.

[103] A F R Rahman, M C Fairhurst, Selective partition algorithm for finding regions of maximum pairwise dissimilarity among statistical class models, PRL 18(7) (1997) 605-611.

[104] K C Leung, C H Leung Recognition of handwritten Chinese characters by critical region analysis, PR 43(3) (2010) 949-961.

[105] E Keogh, M Pazzani, Derivative dynamic time warping, Proc. SDM (2001).

[106] F Jelinek, Statistical Methods for Speech Recognition, MIT Press, 1998.

[107] A Vinciarelli, S Bengio, H Bunke, Offline Recognition of Unconstrained Handwritten Texts using HMMs and Statistical Language Models, IEEE Trans. PAMI 26(6) (2004) 709-720.

[108] M Zimmermann, H Bunke, Optimizing the Integration of a Statistical Language Model in HMM based Offline Handwritten Text Recognition, Proc. ICPR (2004) 203-208.

[109] U V Marti, H Bunke, Unconstrained Handwriting Recognition: Language Models, Perplexity and System Performance, Proc. IWFHR (2000) 463-468.

[110] Y X Li, C L Tan, An Empirical Study of Statistical Language Models for Contextual Post-Processing of Chinese Script Recognition, Proc. IWFHR (2004) 257-262.

[111] Y X Li, C L Tan, Influence of Language Models and Candidate Set Size on Contextual Post-Processing of Chinese Script Recognition, Proc. ICPR (2004) 537-540.

[112] S Quiniou, E Anquetil, A Priori and A Posteriori Integration and Combination of Language Models in an On-Line Handwritten Sentence Recognition System, Proc. IWFHR (2006) 403-408.

[113] F Perraud, C Viard-Gaudin, E Morin, P M Lallican, N-Gram and N-Class Models for On Line Handwriting Recognition, Proc. ICDAR (2003) 1053-1059.

[114] S Quiniou, E Anquetil, S Carbonnel, Statistical Language Models for On-Line Handwritten Sentence Recognition, Proc. ICDAR (2005) 516-520.

[115] A Bharath, S Madhvanath, Online handwriting recognition for Indic scripts, in Guide to OCR for Indic scripts, V Govindaraju and S Setlur, Edn.London, 209-234.

[116] L Rabiner, B Juang, An introduction to hidden Markov models, IEEE ASSP Magazine 3(1) (1986) 4-16.

# Vita

**Suresh Sundaram** received his Masters in Communication Engineering from Indian Institute of Technology Madras. He is currently pursuing his doctoral program in Department of Electrical Engineering in Indian Institute of Science, Bangalore, India. His research interests include development of handwriting recognition technologies for the less researched scripts , pattern recognition and neural networks.

**Angarai Ganesan Ramakrishnan** received his PhD from IIT Madras. A Professor of Electrical Engineering at the Indian Institute of Science, he leads a research consortium on online handwriting recognition, involving 8 Indian languages. His research interests include machine listening and image processing. Earlier, he was President of the Biomedical Engineering Society of India.

# Publications based on this Thesis

## Patent filed

Suresh Sundaram, A G Ramakrishnan, Attention feedback based robust segmentation of online handwritten words. Indian Patent Office Reference. No: 03974/CHE/2010.

## Journal Publication

Suresh Sundaram, A G Ramakrishnan, "Attention feedback based robust segmentation of online handwritten Tamil words", submitted to *ACM Transactions on Asian Language Processing*

Suresh Sundaram, A G Ramakrishnan, " Performance enhancement of online handwritten Tamil symbols with reevaluation strategies", submitted to *Pattern Analysis and Applications*

Suresh Sundaram, A G Ramakrishnan, "Language models for lexicon-free recognition of online Tamil words ", submitted to *Pattern Analysis and Applications*

# Conference Publication

Suresh Sundaram and A G Ramakrishnan, "Lexicon-free, novel segmentation of online handwritten Indic words," accepted for publication in *Proc. Int'l Conf. Document Analysis and Recognition*, September, 2011.