

## 4. Development of OHWR System for Kannada

A G Ramakrishnan and J Shashidhar

Medical Intelligence and Language Engineering Laboratory  
Dept of Electrical Engg, Indian Institute of Science, Bangalore.  
sshashidhar.127@gmail.com, ramkiag@ee.iisc.ernet.in

*Abstract: In this article, we address the challenges in segmentation of online handwritten, isolated Kannada words. This is the maiden work in the field of segmentation of online handwritten Kannada words. Due to the advent of tablet PCs and systems with pen-enabled interface, online handwriting recognition has got wide applications such as form filling, field data collection and word processing. In some of these applications, recognition of names of individuals is required and it is nearly impossible to maintain the lexicon of all possible names. Also, Kannada, being a Dravidian language, is morphologically rich and also agglutinative and thus does not have a finite lexicon. For example, a single root verb can easily lead to a few thousand words after morphological changes and agglutination. Hence, to make recognition of open vocabulary online handwritten Kannada words possible, one must necessarily look into the possibility of segmentation of online Kannada words into their constituent symbols.*

*The modern Kannada script consists of 13 vowels, 34 consonants, 2 other symbols, and 10 numerals. Old Kannada script has additional 3 vowels and 2 consonants. In Kannada, a compound character called akshara is possible by combining consonants and vowels. A total of 8,63,848 different*

*aksharas are possible. We derived a minimal set of 295 distinct symbols to recognize these characters. A corpus of isolated Kannada symbols (MILE Lab Kannada symbols dataset) is collected to study the various statistics about Kannada characters. For addressing the issue of segmentation, Kannada words are collected using a custom application running on a tablet PC from high school. We refer to this dataset as MILE Lab Kannada word dataset.*

*We address the challenges in any online Kannada word into its constituent symbols. We approach the segmentation problem in two steps: preliminary segmentation (PS) and attention feed-based segmentation (AFS). In PS, we use the heuristics from language constructs to segment the word into a set of stroke groups (SG). Then in AFS method, we extract some features from each SG to suspect any under-segmentation or over-segmentation. Then we make use of (i) probability estimate from the classifier and (ii) attributes such as number of dominant points and inter stroke displacements from prior knowledge, to resolve the suspected stroke group by splitting or re-grouping or retaining it. A segmentation accuracy of 83.4% is achieved by PS, which is increased to 94.3% by AFS.*

*We use SVM classifier to recognize and get probability estimates for individual*

*stroke groups. SVM is trained on MILE Lab Kannada symbols training data-set. The smoothed, normalized and re-sampled x, y-coordinates of online trace are used as a feature vector for each sample in the data-set. The character recognition accuracy of 56% is achieved by recognizing the PS outputs and 62% by recognizing the symbols segmented by AFS on the MILE Lab Kannada words.*

### 1. Handwriting Recognition System

Handwriting recognition can be of two types: offline or online [1]. In offline handwriting recognition, the image of the handwritten document is given as input to the system and the system is trained to interpret the handwritten text in it. This has wide applications in recognition of addresses in postal mails, recognition of handwritten check amounts, reading handwritten responses on forms and automatic filing of faxes of handwritten material.

In online handwriting recognition, the user writes on a special note pad or a tablet using a stylus where a sensor picks up the pen-tip movements  $x(t)$ ,  $y(t)$  as well as pen-up/pen-down switching. This data is referred to as digital ink [2] and can be regarded as a dynamic representation of handwriting. With the advent of Tablet PCs and PDAs with touch sensitive interfaces, online handwriting recognition has wide applications in form filling, word processing and also as text input method to PDAs. There is also a possibility for using online handwriting recognition in conjunction with speech synthesis, thereby empowering people with vocal disability to communicate with others.

### 2 Literature survey

Online handwriting recognition of isolated characters for non-Indic scripts is addressed in [3, 4, 5, 6, 7, 8]. Recognition of isolated characters for Indian languages is described in [9, 10, 11, 12, 13, 14, 15, 16, 17, 18]. The recognition of isolated Kannada characters was first explored by Kunte et al. [19], where wavelet features were 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 [20], 295 symbols are derived from the Kannada character set and a divide and conquer technique is proposed to segment any character into three groups namely middle unit (MU), right auxiliary (RA) and bottom auxiliary (BA). These 295 symbols are grouped into MU, RA and BA stroke groups. PCA-based features are then derived specific to each stroke group. The subspace features of each class of stroke groups are fed to their respective nearest neighbour (NN) classifiers for classification. The results from these classifiers are then combined to generate the output character. In another work [21], 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 dynamic time warping (DTW). Orthogonal linear discriminant analysis (OLDA) on a set of PCA features have been recently attempted to the set of Kannada numerals [9].

Most of the applications of online handwriting recognition consider word as a basic unit rather than isolated characters. In

the literature, the problem of online word recognition has been addressed by two methods: analytical approach and holistic approach [1]. Analytical approach treats the word as a collection of simpler subunits such as characters and proceeds by segmenting the word into these units. Then it identifies these units and builds a word-level interpretation. This method can be effectively used to recognize any word. Since in this approach, we are decomposing the word into its constituent subunits, segmentation is a major challenging problem. On the other hand, holistic approach treats the word as a single, indivisible entity and attempts to recognize it using features of the word as whole. This approach can only be applied effectively for limited lexicons. When the lexicon is of bigger size, the ability of the holistic features to distinguish between different classes is diminished and it is also practically not feasible to collect the training data for training a holistic classifier.

In Latin script, the words are usually written in cursive style, the recognition of which is addressed in [22, 23, 24, 25, 26]. In [27], the authors proposed a method for Japanese string recognition where a character string is described as a directional graph (called a candidate lattice) representing possible segmentation and character plausibilities. Then, by means of dynamic programming the candidate character sequence corresponding to the shortest path under linguistic constraints is selected as the segmentation and recognition result. Geometric context extracted from segments has been used for Japanese online handwriting recognition [28, 29]. A two stage segmentation scheme has been proposed to

segment Chinese characters in [30, 31], where the sequences of online handwritten strokes are grouped based on geometrical information at a first stage and then the recognition results and the geometrical features from the pre-segmented characters are used in dynamic programming method to find the best segmentation path.

In Indic scripts, the constituting words are rarely cursive except with the possible exception of Bangla [32, 33]. People generally write the compound characters or aksharas in a word separately from each other with possible overlaps. Recently, Sundaram et al. [34, 35] proposed a feedback based segmentation strategy for lexicon free segmentation of online handwritten Tamil words. In that, they first did a preliminary segmentation based on horizontal overlap criterion and then used feedback from classifier and some inter-stroke features to correct segmentation errors from horizontal overlap criterion segmentation. They called this method as attention feedback segmentation (AFS). Bharath et al. [36] used a HMM framework for modeling the symbols and their relative positions in online Tamil words. However, their work adopts a segmentation-free approach.

### 3 Focus of the work

Kannada is one of the scheduled languages of India and the official and administrative language of the state of Karnataka. Villages constitute the most population in India and most of the villagers prefer to write in their native language. Also, all the government forms are available in both the state language and English. So if people can interact with computers in the native language through the



medium of handwriting, it will enable better technology penetration and will reach the masses. Thus arises the need for developing online handwriting recognition systems for Indian languages.

We aim for the recognition of any online handwritten Kannada word (unrestricted vocabulary). There is currently limited research addressing the challenges pertaining to recognizing Kannada words [37]. Most of the reported techniques deal with the problem of recognizing isolated characters. We use AFS method to segment the online Kannada word into its constituent symbols. This is the maiden work in the field of segmentation of online handwritten Kannada words and there is clearly no prior work in this area on any Indic script, except Tamil. We first do a preliminary segmentation based on Kannada language constructs to get a set of stroke groups and then use feedback from classifier, number of dominant points and inter-stroke features to correct any segmentation errors. The individual symbols are then recognized and combined to get a word level interpretation.

#### 4 Choice of Kannada symbol set

The Kannada alphabet was developed from the Kadamba and Chalaukya scripts, descendants of Brahmi which were used between the 5th and 7th centuries A.D. In terms of the structure of the symbols used, Kannada is unrelated to the descendants of Devanagari such as Hindi, Bengali and Marathi. The modern Kannada script contains 49 phonemic letters which are divided into three groups: 13 independent vowels, 34 consonants and two other letters, namely the anuswara ು and the visarga ು. Figures 1

and 2, respectively, list the set of vowels and consonants of modern Kannada script. In addition, the old (Hale) Kannada script has 3 more vowels (vocalic RR ಋ, vocalic L, vocalic LL) and two more consonants (RRA ಠ, LLLA ಱ).

Further, it also has vowel modifiers and consonant modifiers for each vowel and consonant. The script has its own numerals. Each consonant combines with each of the vowels to form a compound character (CV combination) called akshara. Figure 3 lists the CV combinations corresponding to the consonants ಗ್ /g/ and ಥ್ /th/. Optionally, an akshara can have one or more consonants preceding a CV combination forming a canonical structure of ((C)C) CV. Thus, the number of theoretically possible combinations of Kannada characters is huge and is listed in Table 1.

ಅ ಆ ಇ ಈ ಉ ಊ ಋ ಎ ಏ ಐ ಒ ಓ ಔ  
/A/ /AA/ /I/ /II/ /U/ /UU/ /VOCALIC R/ /E/ /EE/ /AI/ /OI/ /OD/ /AU/

Figure 1. Set of vowels in Kannada

ಕ ಖ ಗ ಘ ಙ  
/K/ /KH/ /G/ /GH/ /NG/  
ಚ ಛ ಜ ಝ ಞ  
/C/ /CH/ /J/ /JH/ /NY/  
ಟ ಠ ಡ ಢ ಣ  
/T/ /TTH/ /DD/ /DDH/ /NN/  
ತ ಥ ದ ಧ ನ  
/T/ /TH/ /D/ /DH/ /N/  
ಪ ಫ ಬ ಭ ಮ  
/P/ /PH/ /B/ /BH/ /M/  
ಯ ರ ಲ ವ ಶ ಷ ಸ ಹ ಳ  
/Y/ /R/ /L/ /V/ /SH/ /SS/ /S/ /H/ /LL/

Figure 2. Set of consonants in Kannada



Figure 3. Set of all CV combinations for /g/ and /th/

Table 1. Number of possible combinations of Kannada Characters.  
(V: Vowels; C: Consonants; N: Numerals)

Char Type	V	C	CV	CCV	CCCV	N	Total
Possible Combinations	18	36	648	23328	839808	10	863848

From Table 1, it is clear that considering each combination as a separate class for recognition increases the computational cost and may reduce the recognition accuracy. Also, it is not practically feasible to ask writers to write all the combinations during data collection. In this section, we describe a method to get a comprehensive set of symbols that can be employed in the recognition of any Kannada akshara.

All vowel modifiers change the shape of the consonant in the CV combination (Fig. 3). Based on their relative position in a CV combination, vowel modifiers can be classified into different types: (a) Some vowel modifiers have significant overlap in the writing direction. Examples of this case are shown in Fig. 4. These CVs are treated as distinct classes. (b) For some vowel modifiers, a special pattern is written at the bottom right of the modified consonant as shown in Fig. 5. From the point of view of recognition, it would suffice to recognize the modifier separately and then append it to the corresponding base character. (c) Three vowel

modifiers are written separately towards the right of the consonant as shown in Fig. 6. These vowel modifiers are segmented and recognized as separate classes, which reduces the number of classes. (d) Two vowel modifiers are written to the right of the consonants with less overlap in the writing direction as shown in Fig. 7. These CV combinations are treated as separate classes for recognition. (e) There are some special cases of these modifiers, where they are written from below the consonant as shown in Fig. 8. The corresponding CV combinations are again treated as distinct classes. (f) There are few characters, which can be split into different symbols. Examples of such cases are shown in Figure 9. The symbols shown in Figure 9(b) are considered as separate classes.



Figure 4. Examples of vowel modifiers which have significant overlap in the writing direction with the consonant in the CV combinations.

ಕ ಕೆ  
/KR/ /KAI/

Figure 5. The consonant /k/ with the two vowel modifiers below.

ಕೆ ಕೇ ಕೋ ಕಂ ಕಃ  
/KE/ /KEE/ /KOO/

Figure 6. The consonant /k/ with vowel modifiers written separately to the right.

ಕು ಕೂ ಕೊ ಕೋ  
/KU/ /KUU/ /KO/ /KOO/

Figure 7. Vowel modifiers written to the right of consonant /k/ with some overlap.

ಪು ಪೂ ಪೊ ಪೋ ವು ವೂ ವೊ ವೋ  
/PU/ /PUU/ /PO/ /POO/ /VU/ /VUU/ /VO/ /VOO/

Figure 8. Special consonants for which few vowel modifiers start below the consonant and written towards right.

ಮ್ ಮ ಮಿ ಮೂ ಮೆ ಮೌ ಯ ಯಾ ಯೆ ಯೌ ಝ ಝಿ ಝು ಝೂ  
/m/ /mi/ /mu/ /ma/ /me/ /mo/ /ya/ /ye/ /yo/ /zha/ /zhi/ /zhu/ /zhaa/

Figure 9. (a) Examples of characters that can be split into two or more parts. (b) Symbols obtained after splitting the characters in (a).

In a typical CCV (say, /dh+/y+/aa/) combination, the first consonant (/dh/) and the vowel (/aa/) are first written as a CV combination (/dhaa/). Then the consonant modifier of the middle consonant (/y/) is written below this CV combination. A few examples of such CCV combinations are shown in Fig. 10. A consonant modifier that occurs below a CV combination is referred to as “ottu”.

ಟ+ರ+ಈ=ಟ್ರೆ ಕ+ರ+ಓ=ಕ್ರೋ ಷ+ಟ+ಅ=ಷ್ಟ  
/TT/ /R/ /U/ /K/ /R/ /OO/ /SH/ /TT/ /A/  
ಕ+ಟ+ಅ=ಕ್ಷ ತ+ಯ+ಅ=ತ್ಯ ಷ+ಮ+ಇ=ಷ್ಮಿ  
/SH/ /TT/ /U/ /N/ /T/ /Y/ /N/ /SS/ /M/ /I/  
ಶ+ವ+ಇ=ಶ್ವಿ ತ+ಸ+ಅ=ತ್ಸ ಧ+ಯ+ಅ=ಧ್ಯಾ  
/SH/ /V/ /I/ /T/ /S/ /A/ /DH/ /Y/ /AA/

Figure 10. Example CCV combinations, where the vowel added to the second consonant actually modifies the first consonant in the final grapheme of the akshara.

In CCCV combinations, the first consonant and the final vowel are first combined as a normal CV combination. Then the consonant modifiers corresponding to the 2-nd and 3-rd consonants are written below this CV combination. A few examples of such CCCV combinations are shown in Fig. 11. Henceforth, we refer to the Kannada symbols (except ottu and the symbols in vowel modifiers written below the consonants) as the base characters.

ಷ+ಟ+ರ+ಈ=ಷ್ಟ್ರಿ ಸ+ವ+ರ+ಇ=ಸ್ತ್ರಿ ಷ+ಟ+ರ+ಅ=ಷ್ಟ  
/SS/ /TT/ /R/ /U/ /S/ /V/ /R/ /I/ /SS/ /TT/ /R/ /A/  
ಕ+ಟ+ರ+ಇ=ಕ್ತ್ರಿ ಸ+ಕ+ಯ+ಅ=ಸ್ಕ್ತ್ರಿ ಕ+ಷ+ಮ+ಇ=ಕ್ಷ್ಮಿ  
/SH/ /TT/ /R/ /I/ /S/ /K/ /Y/ /N/ /K/ /SS/ /M/ /I/  
ಕ+ಷ+ಮ+ಅ=ಕ್ಷ್ಮ ತ+ಸ+ಯ+ಅ=ತ್ಸ್ಮ ಕ+ಶ+ಮ+ಇ=ಕ್ಶ್ಮಿ  
/K/ /SS/ /M/ /I/ /T/ /S/ /Y/ /A/ /K/ /SH/ /M/ /I/

Figure 11. Example CCCV combinations

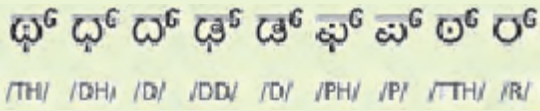
Both in CCV and CCCV combinations, it would suffice to recognize the ottu separately and then append it to the corresponding base character. The issue of segmenting the ottu from an akshara is dealt with separately in Section 8. By this, the number of distinct symbols to be recognized by the classifier is reduced.

There are four sets of consonants which differ by a dot at the middle of the grapheme or a short vertical line (padam) at the bottom of



the grapheme or both. If these dots and padam can be removed at the time of preprocessing, such characters can be considered as the same symbol. Examples of these characters are shown in Fig. 12.

In addition to all these characters, 9 Kannada numerals, 9 Hindu-Arabic numerals and 22 special symbols used in poetry, shlokas and Kannada grammar are also considered for recognition. Thus, a total of 295 classes are derived, by which any Kannada akshara can be formed [38]. The list of these 295 classes is shown in Appendix A.



**Figure 12. Consonants differing by a dot or vertical line or both.**

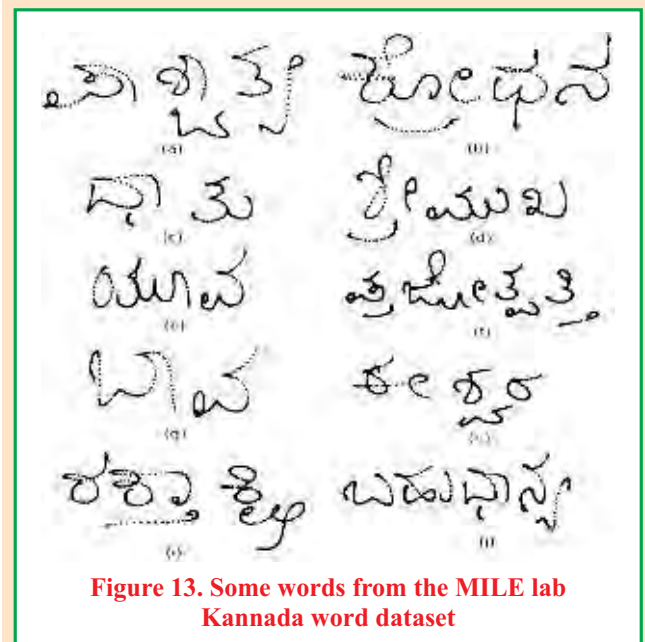
### 5 Datasets used for the experiments

We have used two data sets. One contains Kannada symbols and the other has words. We collected the Kannada symbols from 69 writers, each writing all the 295 symbols, 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 Kannada writers. From here on, we call this data set as MILE Lab Kannada symbol data set. We used this dataset to collect various statistics of Kannada symbols to be used in the segmentation of online handwritten Kannada words.

We also collected one lakh isolated Kannada word samples using both Genius GNote 7 and a tablet PC [38]. Two word lists containing 246 and 198 words each (DB1 and DB2) are selected to cover all the 295 symbols. A total of about 500 native Kannada writers, of different age groups, contributed in

building the word dataset. In this article, we report the results of only the experiments conducted on the data collected using tablet PC, which has a sampling rate of 1200 Hz and a spatial resolution of 2500 dpi along both X and Y directions. This part has 44, 772 words. 50 writers have written all the words in DB1 and the words in DB2 were written by 164 writers. From here on, we call this dataset as MILE Lab Kannada word dataset.

This contains a mix of all kinds of handwriting styles: class A proper strokes, easily segmentable; class B - segmentable with sophisticated methods; class C - broken and merged strokes, adjacent strokes overlapping heavily and delayed but valid strokes; class D - extraneous strokes or overwriting and strokes written in the opposite direction, but the resulting strokegroups having the potential to be properly recognized using offline features, after removing the extraneous strokes; class R - reject class, where the likelihood of recognition is low. Figure 13 presents sample words from our dataset.



**Figure 13. Some words from the MILE lab Kannada word dataset**

## 6 Overview of basic recognition module

In this section, we present the details of the recognition system used in our experiments. The recognizer has been developed to work on isolated Kannada symbols. The following subsection outlines the preprocessing and feature extraction steps that generate a fixed dimensional vector for each symbol. Subsection 6.2 outlines the details of the classifier used in recognizing a Kannada symbol.

### 6.1 Preprocessing

As discussed in Section 1, an online handwritten symbol, captured by a digitizer, is a sequence of x-y coordinates with pen-down and pen-up events. The captured symbol is preprocessed in order to compensate for variations in time, scale and velocity. It comprises 3 steps: smoothing, normalization and re-sampling. Smoothing reduces the amount of high frequency noise in the input, resulting from the capturing device or jitters in writing. Smoothing is performed on each stroke separately using a Gaussian low-pass filter.

After smoothing, the bounding box of the symbol is resized to a fixed size such that the scale variations are eliminated. Both x and y coordinates are separately mapped to the  $[0,1]$  range by a linear transformation.

The input data from the digitizer is sampled uniformly with respect to time. Hence, it is resampled to get a fixed number  $N_p$  of points uniformly sampled in space. The re-sampling is done as follows: we first compute the arc-length of each stroke by adding the Euclidean distances between successive points. The arc-length of the symbol is calculated by adding the arc-lengths of individual strokes. The number of samples for each stroke is chosen so that it is proportional to its arc-length. The points from a smoothed stroke are then re-sampled at constant interval by using linear interpolation. The interval length is calculated by dividing the arc-length of the stroke by the number of intervals required.

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

$$x = (x_1, x_2, \dots, x_{N_p}, y_1, y_2, \dots, y_{N_p})$$

We refer to  $x$  as the concatenated x – y coordinates in this work. We experimented with varying number of re-sampled points and observed that  $N_p = 60$  is quite sufficient in capturing the shape of the character including points of high curvature. Figure 14 illustrates the preprocessing steps on a sample of symbol  $\text{ಀ} / \text{II} /$ .

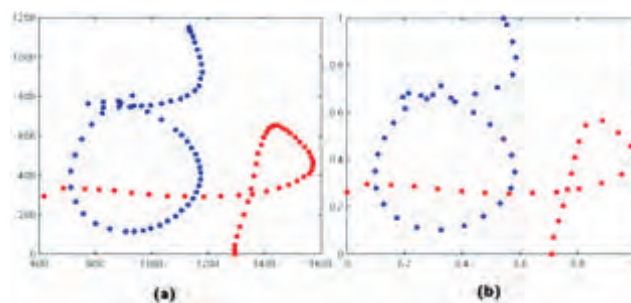


Figure 14. Pre-processing of a SG prior to deriving features for SVM.  
(a) Original SG. (b) Smoothed, normalized and re-sampled SG to 60 samples.



## 6.2 Recognition

In our experiments, we use support vector machine (SVM) classifier, as it gives good generalization performance on unseen data, to recognize online handwritten Kannada symbols [39]. We incorporate the knowledge of probability estimates and the most favoured symbol returned by the SVM in our algorithm to improve the segmentation of online handwritten Kannada words. SVM is a supervised learning method for a 2-class pattern classification problem [39]. In this work, we used the LIBSVM software for learning the SVM parameters [41]. The OVO scheme is used for training. The samples corresponding to the 295 symbols in the MILE Lab Kannada symbols dataset are used to learn the optimal values for the model parameters. The RBF kernel is used as the kernel function. A recognition accuracy of 85% is obtained on the MILE Lab Kannada symbols test data set with parameters  $C = 10$  and  $\gamma = 0.3$ . The kernel and the corresponding parameters are set after performing a 3-fold cross validation experiments on the MILE Lab Kannada symbols dataset. In the following section, we describe our proposed method for the segmentation of lexicon free, isolated online handwritten Kannada words.

## 7 Proposed work

Given an online Kannada word, our emphasis is to correctly segment it into its constituent symbols by employing a feedback based strategy. During the collection of online handwritten words, the pen-tip movements  $x(t)$  and  $y(t)$  as well as pen-up/pen-down switching are sensed. The sequence of  $(x, y)$  coordinates sensed from a pen-down state to a pen-up state is considered

as a stroke. The script being non-cursive in nature, an online word is represented as a sequence of  $n$  strokes  $W = \{s_1, s_2, s_3, \dots, s_n\}$ , where  $s_i$  are the individual strokes. We segment the input word ( $W$ ) into a sequence of  $m$  distinct patterns, referred to as stroke groups  $W = \{S_1, S_2, S_3, \dots, S_m\}$ ,  $m \leq n$  and  $S_i \cap S_j = \Phi$  for  $i \neq j$ . A stroke group  $S_k$  is defined as a set of one or more consecutive strokes, which is possibly a valid Kannada symbol.

The block diagram of the proposed strategy is shown in Fig. 15. We first propose a preliminary segmentation (PS) based on horizontal overlap between successive strokes. However, due to the wide variability in handwriting, preliminary segmentation may not be perfect all the time. Hence, we adopt feature based attention and feedback from the recognition system to detect and correct any segmentation errors present after PS.

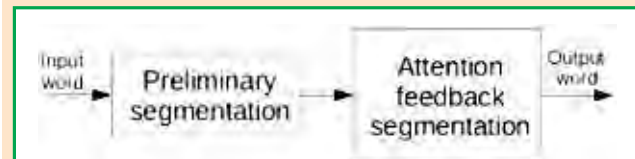


Figure 15. Block diagram of proposed segmentation method

## 8. Preliminary segmentation (PS)

In Kannada, the strokes of the same symbol significantly overlap in the horizontal direction. Ottus occur at the bottom of consonants and a special symbol called padam occurs at the middle and below some consonants. Our algorithm segments the word] based on the above a priori knowledge, by checking for (1) padam and dot detection, (2) bounding box (BB) overlap in writing direction and (3) portion of  $y$ -range of a stroke below the previous SG, thus generating a set of SGs.

**Padam detection:** A stroke  $s_i$  is represented as a sequence of  $n_i$  points  $s_i = \{p_1, p_2, \dots, p_{n_i}\}$ . We define the degree of linearity of stroke  $s_i$  as  $L_i$

where  $d(p, q)$  is the Euclidean distance between points  $p$  and  $q$ . A stroke  $s_i$  is detected as padam if all of the following conditions are satisfied.

$L_i < T_0$  (threshold for linearity check).

Its y-range should be greater than the x-range.

Its y-mean should be less than the y-minimum of the previous SG.

Its y-max should not be greater than middle y-value of previous SG.

**Dot detection:** As discussed in section 4, some pairs of characters in Kannada just differ by the presence or absence of a small dot. Since we do not consider characters with dot as a separate class for recognition, we detect the dots from the word and set a flag for it, which is used at the time of unicode generation. A stroke  $s_i$  is detected as a dot if both the following conditions are satisfied.

1. The height of its bounding box is less than  $\alpha_1$  times the maximum of the heights of the bounding boxes of all the strokes in the word.
2. The width of its bounding box is less than  $\alpha_2$  times the maximum of the widths of the bounding boxes of all the strokes in the word.

The bounding box overlap in the writing direction (horizontal) is quantified by

$$HO = \max \left( \frac{x_{max}^{S_k} - x_{min}^i}{x_{max}^{S_k} - x_{min}^{S_k}}, \frac{x_{max}^{S_k} - x_{min}^i}{x_{max}^i - x_{min}^i} \right)$$

where  $x_{min}^{S_k}$ ,  $x_{min}^i$  and  $x_{max}^{S_k}$ ,  $x_{max}^i$  are the minimum and maximum x-coordinates of stroke group  $S_k$  and stroke ( $s_i$ ), respectively. The part of y-range of a stroke that lies below the previous SG is quantified by

$$VB_k^i = \frac{y_{min}^{S_k} - y_{min}^i}{y_{max}^i - y_{min}^i}$$

where  $y_{min}^{S_k}$  is the minimum y-coordinate value of stroke group ( $S_k$ ) and  $y_{min}^i$  and  $y_{max}^i$  are the minimum and maximum y-coordinate values of stroke ( $s_i$ ), respectively. If the current stroke  $s_c$  is written below the previous stroke group  $S_k$ , then that stroke is considered as an ottu and split from  $S_k$  to form the next stroke group  $S_{k+1}$ . Then we set the ottu status of  $S_{k+1}$  to 1 ( $S_{k+1}.ottu = 1$ ). Figures 16 and 17 depict the parameters employed for computing  $VB_k^i$  and HO.

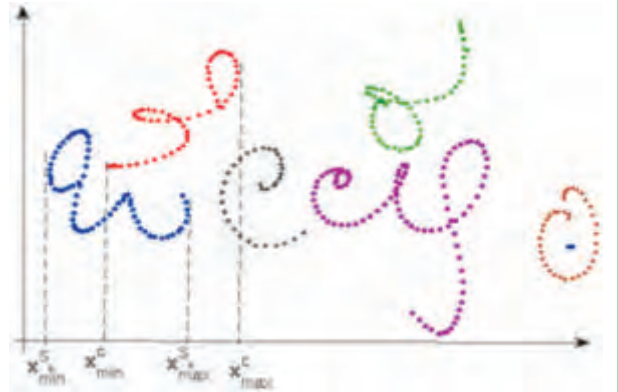


Figure 16. Parameters employed for calculating HO. Each stroke is shown by a different color.

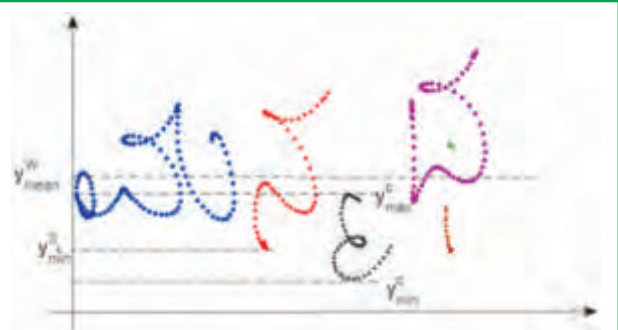


Figure 17. Parameters employed for calculating  $VB_k^i$ . Each stroke is shown by a different color.

In Kannada, a word always starts with a base character. So, we consider the first stroke of a word as a part of a base character, thus forming the first stroke group of the word. The flowchart for each iteration in the proposed algorithm for preliminary segmentation is shown in Fig. 18. Figures 19 and 20 show the results after applying PS to different input words. In each iteration, if any stroke is detected as a padam or dot (block D1), then a corresponding flag is set to the current stroke group ( $S_k$ ) under consideration (block P1), which is made use of at the time of unicode generation.

For example, in Fig. 20(a), the first stroke is considered part of a base character. The second stroke is written completely below the previous stroke group and hence is considered to be an ottu and its ottu status is set to 1 (block P2). For the third stroke, (its predecessor stroke group is an ottu) its maximum y-value is above the mean y-value of the word and as shown in the flow chart (block D3), this stroke is considered part of a new base character and its ottu status is set to 0 (block P3). The fourth stroke is written slightly below its predecessor stroke group which is a base character and the maximum y-value is also greater than mean y-value of the word. As shown in the flow chart (block P2), this stroke is split from its predecessor stroke group and its ottu status is set to 0. The fifth stroke has less horizontal overlap with its predecessor stroke group and hence is split from it and its ottu status is set to be the same as its predecessor stroke group (block P5). The algorithm continues for the rest of the strokes as shown in Fig. 18.

However, preliminary segmentation may sometimes output a stroke group which is

either (1) a merge of two valid symbols or (2) a part of a valid symbol. These errors are illustrated below:

1. Merging of two valid symbols (under-segmentation): Figure 21(a) illustrates such an error in which a stroke group is formed by the merger of two valid symbols ತ and ಲ in the word ತಲೆಕೆಲಸ.
2. Splitting a symbol into two or more stroke groups (over-segmentation): Figure 21(b) illustrates an over-segmentation error in which a symbol ಗ is split into two stroke groups in the word ಮಂಗಳ.

In the next section, we discuss the proposed method of attention feedback segmentation (AFS) to correct the segmentation errors obtained after preliminary segmentation.

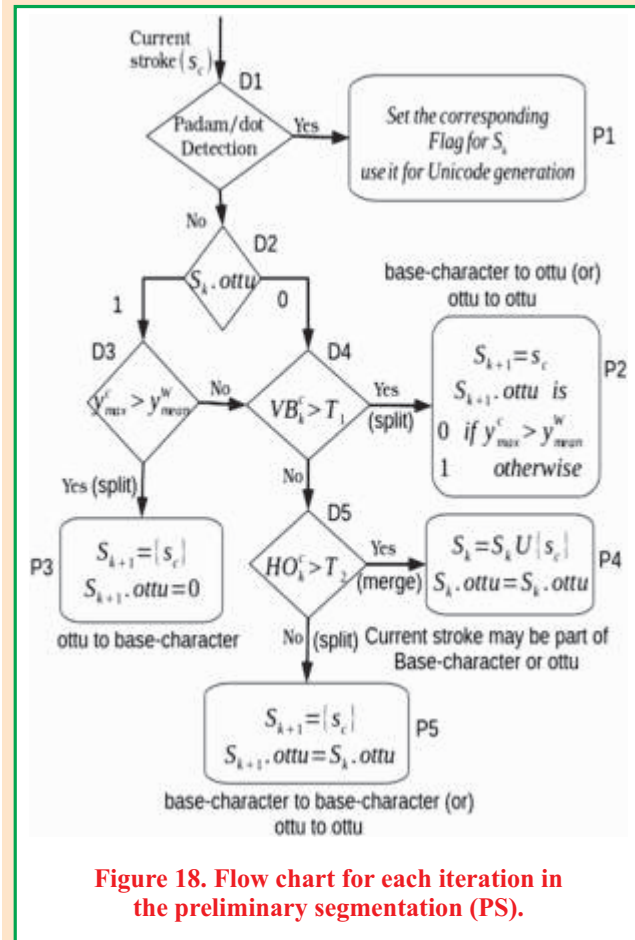






Figure 19. Results after applying PS. (a) Input online word. Each stroke is marked with a different color. (b) Word after doing PS. Each SG is marked with a different colour. The stroke order is blue, red, black, cyan, magenta, green and then the colours get repeated.

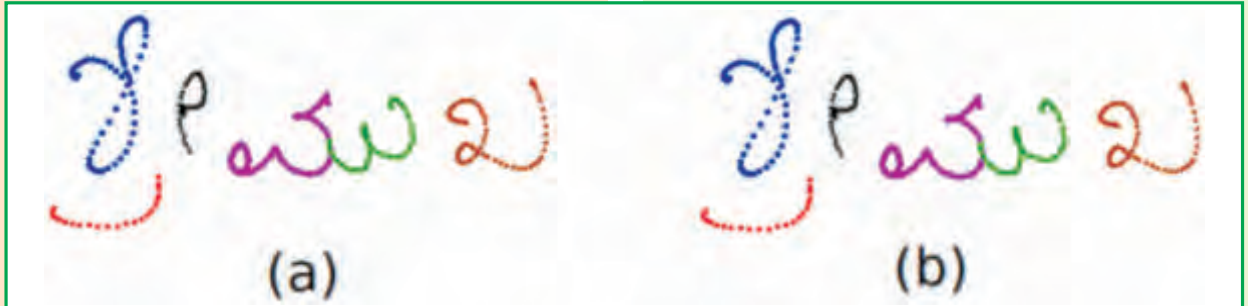


Figure 20. Results after applying PS. (a) Input online word. Each stroke is marked with a different color. (b) Word after doing PS. Each SG is marked with a different colour.

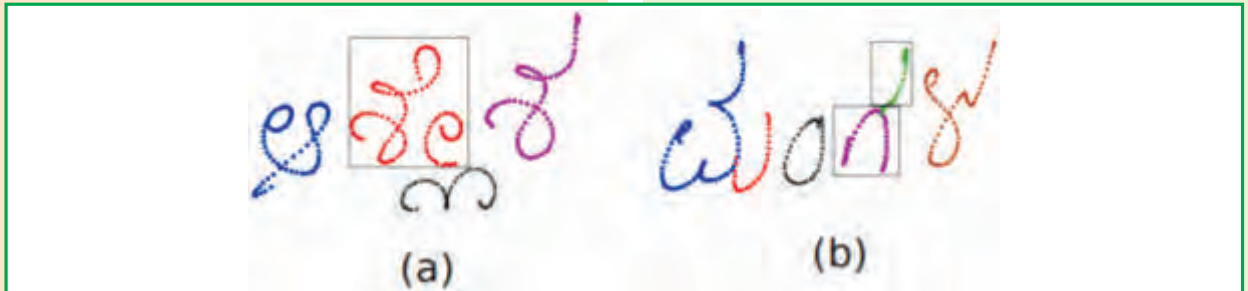


Figure 21. Illustration of under-segmentation and over-segmentation errors. (a) The /she/ and /I dheerga/ symbols got merged into a single stroke group. (b) The valid symbol /ga/ got split into two stroke groups as shown by the separate bounding boxes.

## 9 Attention-feedback segmentation

AFS aims to refine the segmentation errors introduced by PS. The block diagram of the AFS is shown in Figure 22. In this, we extract some features from PS stroke groups and use these features to suspect any SGs which are possibly wrongly segmented. Then, we use feedback from SVM probability estimates and statistics derived from the

MILE lab Kannada symbols dataset and take a decision to retain, split or combine with adjacent stroke groups. We do AFS on only those SGs detected as possibly wrongly segmented and then split them, if necessary. Thereafter, we check for SGs, which are possibly part of a valid symbol and then combine them with appropriate neighboring stroke groups.

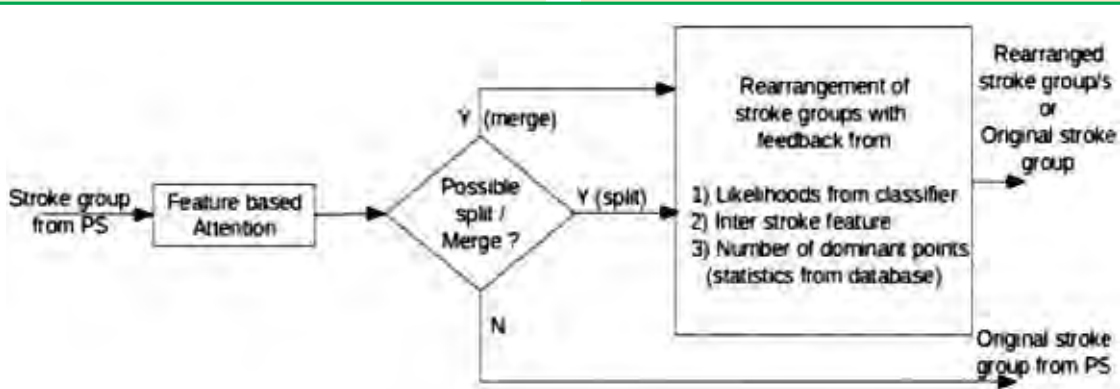


Figure 22. Block diagram of AFS.

### 9.1 Features for attention

We now describe the features we use to detect the wrongly segmented stroke groups.

**Inter-stroke feature:** For the preprocessed stroke groups comprising  $q$  strokes ( $q > 1$ ), we take the vertical displacement  $v_i$  from the last point of the  $i$ -th stroke to the first point of the  $(i+1)$ -th stroke:  $v_i = (y_{\text{last}}^i - y_1^{i+1})$ ,  $1 \leq i < q$ . The maximum of all  $\{v_i\}$ ,  $i=1 \dots q-1$ , among all successive stroke pairs ( $v_{\text{max}}$ ) is a feature for attention. The value of  $v_{\text{max}}$  can be positive or negative. For the stroke group /kau/ in Fig. 23,  $v_{\text{max}}$  is negative. The SG with  $v_{\text{max}} > 0$  may be under-segmented.

In Kannada, for each consonant-vowel combination, a distinct shaped stroke is written at the top of the consonant. This vowel specific addition can be an extension of the consonant stroke or a separate new stroke.

While writing such characters, sometimes the writer extends the additional stroke towards the right of the character. Because of this, there will be some space left below this extension and to the right of the character. So, most writers tend to start the next character from this space, thus causing significant amount of overlap between these characters in the writing direction. Our PS algorithm merges these characters, if the horizontal overlap between these characters exceeds the threshold  $T_2$ . For all these merged SGs,  $v_{\text{max}}$  is greater than zero. This feature can be used to detect under-segmented stroke groups. Figure 24 depicts the case, wherein two Kannada symbols /she/ and /E matra/ are merged by PS as a single SG. It is likely that the SVM classifier will indicate this SG to be an outlier by returning a low probability estimate to its most probable symbol.

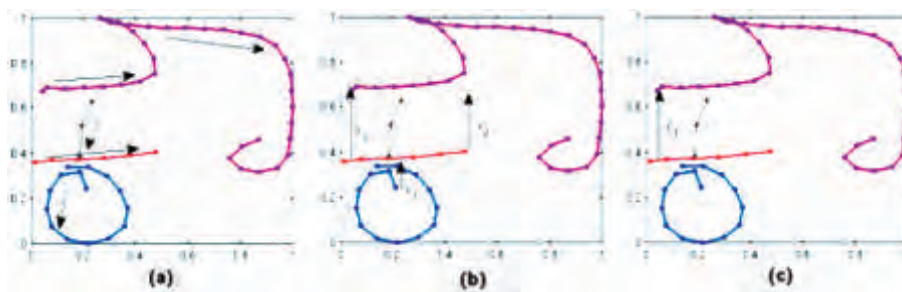


Figure 23. Illustration of  $v_{\text{max}}$  for the stroke group /kau/. (a) Stroke group /kau/ with the direction of trace marked by arrows. (b) Illustration of the three inter-stroke displacements  $v_1$ ,  $v_2$  and  $v_3$ . (c) Illustration of  $v_{\text{max}}$ . For this stroke group,  $v_{\text{max}} < 0$ .

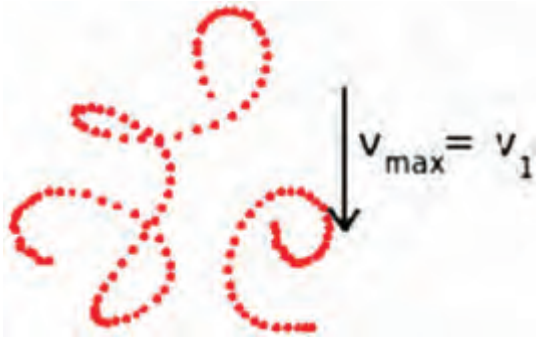


Figure 24. Two different symbols merged by PS. This combined stroke group satisfies  $v_{\max} > 0$

### Mean y-coordinate of base characters:

The mean of the mean y-coordinates of all the stroke groups not detected as ottus in PS is a feature for attention.

$$y_{mean}^{BC} = \frac{\sum_{i=1}^p y_{mean}^{S_i} (1 - S_{i, ottu})}{\sum_{i=1}^p (1 - S_{i, ottu})}$$

Here,  $p$  is the number of SGs identified by PS. Any SG lying completely above this mean value is suspected to be over-segmented.

### 9.2 AFS strategy for under-segmented stroke groups

As discussed in section 9.1, a stroke group  $S_k$  for which  $v_{\max} > 0$  may correspond to an under-segmented stroke group (USG). In this section, we outline the strategy for resolving

such USGs. Figure 25 shows the block diagram of the proposed strategy. We used the feedback from SVM probability estimates, statistics of  $v_{\max}$  and number of dominant points obtained from MILE lab Kannada symbols dataset to split or retain the SG.

**Number of dominant points:** The dominant points (DP) provide a rich structural description of a stroke group [34]. The algorithm starts by marking the first and last points of the stroke as DPs. Starting from the current DP, we compute the absolute angle between pen directions of successive points. Then we accumulate these angles along the online trace till the sum exceeds a threshold ( $T\theta$ ). The point at which this happens is marked as the next DP and the process continues till the end of the stroke. The resultant number of DPs extracted is used as a feature for attention. Figure 26 shows the dominant points extracted for the stroke group  $\text{ನೆ/ನೇ/}$ .

Let  $v_{\max}$  correspond to the inter-stroke gap between the  $r$ -th and  $(r+1)$ th strokes in  $S_k$ , respectively. Accordingly, we consider  $S_k$  as a possible merger of two valid symbols  $S_{k1}$  and  $S_{k2}$ , defined by

$$S_{k1} = \{s_{k1}, s_{k2}, \dots, s_{kr}\} \text{ and } S_{k2} = \{s_{kr+1}, s_{kr+2}, \dots, s_{kq}\}$$

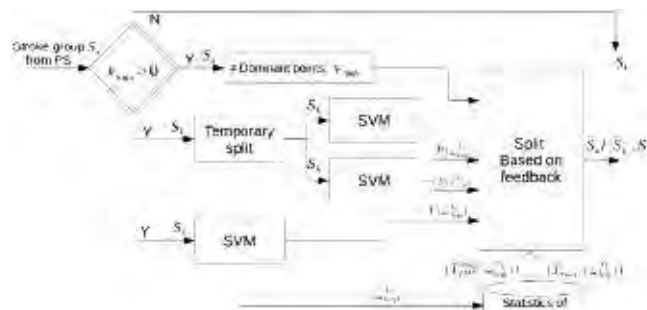


Figure 25. AFS module for suspected under-segmentation.



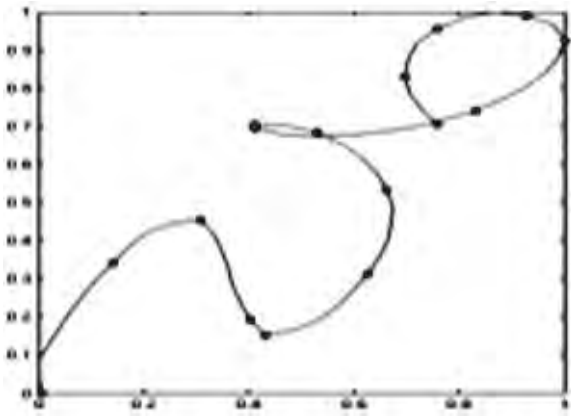


Figure 26. Extraction of dominant points (DP) for a stroke group /NE/

Here,  $S_{ki}$  denotes the  $i$ -th stroke in the  $k$ -th stroke group. We input the pre-processed stroke groups  $S_k$ ,  $S_{k1}$  and  $S_{k2}$  to SVM and get the most probable symbols  $\omega_{top}^k$ ,  $\omega_{top}^{k1}$  and  $\omega_{top}^{k2}$  with probability estimates  $P(\omega_{top}^k)$ ,  $P(\omega_{top}^{k1})$  and  $P(\omega_{top}^{k2})$ , respectively. We favour splitting  $S_k$  into two stroke groups  $S_{k1}$  and  $S_{k2}$ , if

$$[P(\omega_{top}^{k1}) + P(\omega_{top}^{k2})] / 2 > P(\omega_{top}^k)$$

Figure 27 illustrates the case wherein the wrongly segmented stroke group /shee/ in the word ಉತ್ತರ gets correctly segmented to 2 valid Kannada symbols, /she/ and /e/.

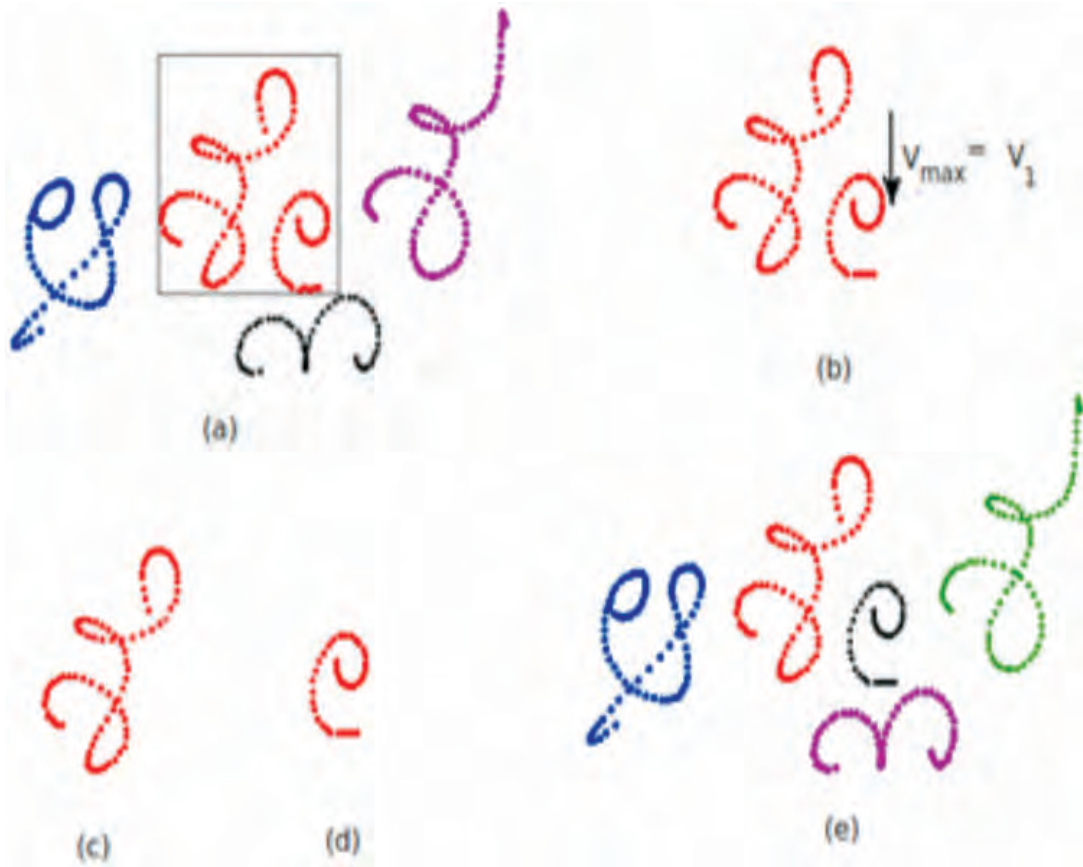


Fig. 27. Illustration of AFS scheme to resolve possibly under-segmented stroke groups. (a) Word wrongly segmented by PS. (b) Under-segmented SG which satisfies  $v_{max} > 0$ . (c) and (d) The extracted symbols are recognized separately. The SG is split since the mean probability estimate of extracted SGs exceeds that of combined SG. (e) Correctly segmented word.

### 9.3 AFS strategy for over-segmented stroke groups

Figure 28 shows the block diagram of the strategy for correcting over-segmentation errors. As discussed in Sec. 9.1, a SG above the middle line of base characters can be suspected to be over-segmented. Let  $S_k$  be a SG, whose minimum y-coordinate is greater than mean y-coordinate of base characters ( $y_{mean}^{BC}$ ). Let  $S_{adj}(k)$  be an adjacent stroke group to  $S_k$ , whose BB overlap is maximum in the writing direction. We temporarily merge these two stroke groups  $S_k$  and  $S_{adj}(k)$  to form another stroke group  $S_M$  and input the pre-processed SGs  $S_k$ ,  $S_{adj}(k)$  and  $S_M$  to the SVM and get the most probable symbols  $\omega_{top}^k$ ,  $\omega_{top}^{adj}$  and  $\omega_{top}^M$  with probability estimates  $P(\omega_{top}^k)$ ,  $P(\omega_{top}^{adj})$  and  $P(\omega_{top}^M)$ , respectively. We merge the stroke groups  $S_k$  and  $S_{adj}(k)$  to form  $S_M$  if

$$\frac{P(\omega_{top}^k) + P(\omega_{top}^{adj(k)})}{2} < P(\omega_{top}^M)$$

Figure 29 illustrates the case wherein the over-segmented stroke group  $\overline{n}$  /ga/ in the word  $\overline{m}$  gets correctly segmented by applying the above strategy.

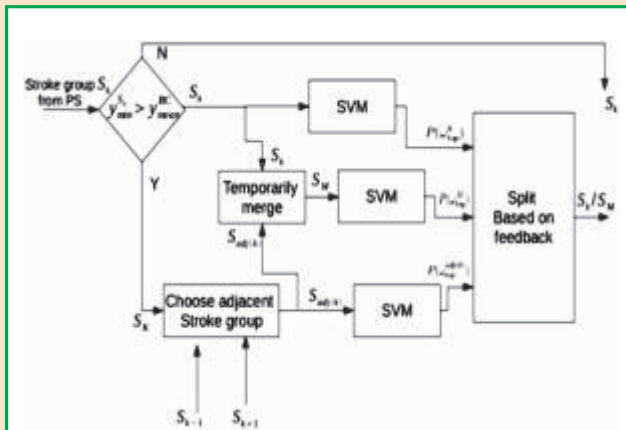


Figure 28. AFS module for resolving over-segmentation errors.



Figure 29. Illustration of AFS scheme to resolve over-segmented stroke groups. (a) Wrongly segmented word after applying PS (b) Stroke group suspected as over-segmented. (c) Adjacent stroke group whose BB overlap is maximum in the writing direction. (d) The combined stroke group. The stroke groups are merged if probability estimate of combined stroke group is greater than the mean probability estimate of individual stroke groups. (e) Correctly segmented word.

## 10 Results

### 10.1 Experimental setup

Before applying AFS to the input word, the parameters of the SVM are trained with the concatenated x and y coordinates of the pre-processed Kannada symbols as described in section 6.1. In addition, the statistics of the following two features are generated for each symbol  $\omega_i$  from the MILE Lab Kannada character data-set: (i) Maximum number of dominant points  $T_{DP}^{max}(\omega_i)$  across all samples of  $\omega_i$ . (ii) The maximum  $v_{max}(T_{max}(\omega_i))$ , across all samples of  $\omega_i$ . The threshold T0 used for linearity check in the padam detection is selected to 1.8. We extracted the padam strokes from the relevant symbols in the MILE Lab word data-set and calculated the  $L_i$  for each stroke. Figure 30 shows the histogram of

$L_i$  for these padam strokes with the bin size of 0.1. From Fig. 30, we can observe that  $L_i$  is less than 1.8 for all padam strokes and hence we choose the threshold of 1.8 for  $L_i$ .

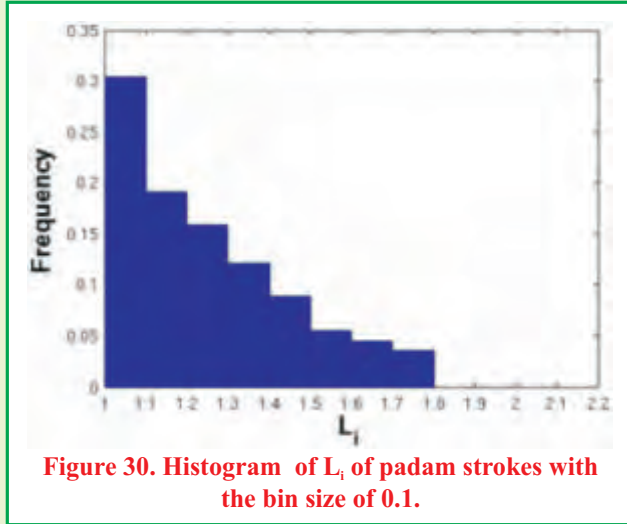


Figure 30. Histogram of  $L_i$  of padam strokes with the bin size of 0.1.

We extracted the dot strokes from the valid symbols in the MILE Lab word dataset and calculated the ratios of BB height and width of the dot stroke to the maximum BB height and width of all strokes in the word respectively. Figure 31 shows the histogram of these ratios for all the dot strokes with the

bin size of 0.1. We observe that  $\alpha_1$  and  $\alpha_2$  are less than 0.4 for all dot strokes and hence we choose the threshold of 0.4 for both the parameters  $\alpha_1$  and  $\alpha_2$  used for dot detection.

The thresholds  $T_1$  and  $T_2$  used for VB and HO in PS are selected as 0.3 and 0.2, respectively. We calculated the minimum HO for the samples of all classes in the MILE Lab Kannada character dataset and selected an initial value for  $T_2$  as 0.2 from Fig. 32. Using this value, we calculate the segmentation errors given by PS and AFS by varying  $T_1$  from 0 to 1 as shown in Figure 33(a). From Figure 33(a), we can see that AFS is robust to the changes in VB. We then fixed the value of  $T_1$  to 0.3 and calculated the segmentation errors given by PS and AFS with varying  $T_2$  from 0 to 1 as shown in Fig. 33(b). We can see from Figure 33(b) that segmentation accuracy peaks for values of HO from 0.2 to 0.4 and hence we have selected the value of  $T_2$  to be 0.2.

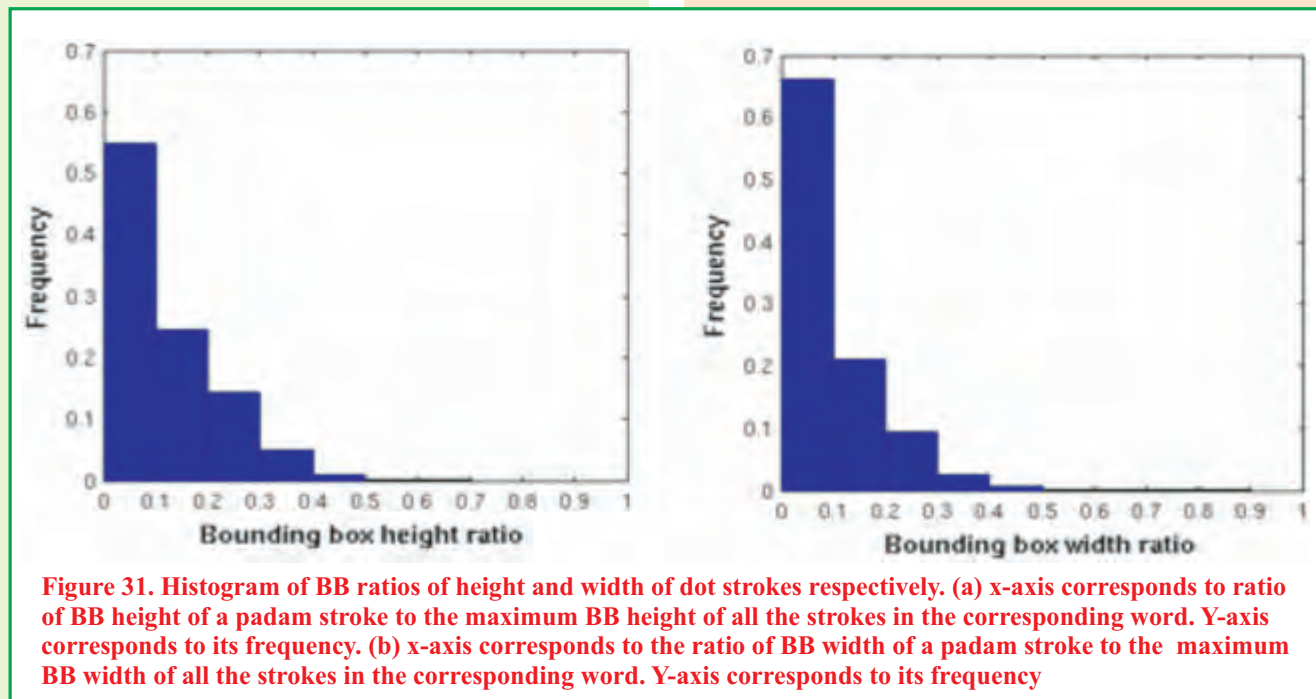


Figure 31. Histogram of BB ratios of height and width of dot strokes respectively. (a) x-axis corresponds to ratio of BB height of a padam stroke to the maximum BB height of all the strokes in the corresponding word. Y-axis corresponds to its frequency. (b) x-axis corresponds to the ratio of BB width of a padam stroke to the maximum BB width of all the strokes in the corresponding word. Y-axis corresponds to its frequency



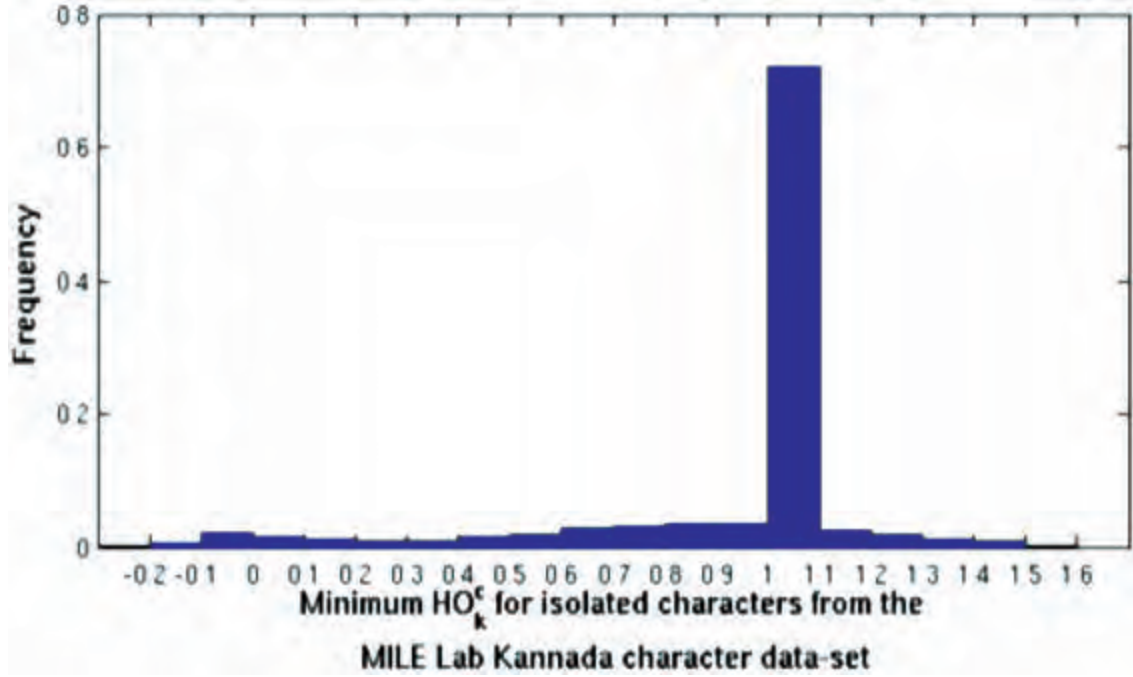


Fig. 32. The histogram of minimum HO in each sample from MILE Lab Kannada character dataset with a bin size of 0.1.

### 10.2 Segmentation results on MILE lab Kannada word data-set

We implemented our algorithm in C language and tested on a total of 44772 words in MILE Lab Kannada word dataset. A few sample words, whose segmentations have been corrected by our approach, are shown in Tables 2 and 3. Application of the PS on each word in Table 2 leads to a merge of valid symbols. On the other hand, at least one valid symbol in each word in Table 3 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 the AFS.

The segmentation and symbol recognition accuracies are shown in Table 4. An improvement of 10.9% in segmentation accuracy and 6% in symbol recognition rate are achieved by our AFS method. Although there is an improvement, the symbol recognition accuracy itself is not so satisfactory because of training SVM classifier with only 69 samples per class in MILE Lab Kannada character dataset. Accordingly, the corresponding word recognition accuracy is also lower. Whenever two different symbols are joined as a single stroke, the word does not get properly segmented, since the proposed algorithm is not designed to handle such cases. Further, the two strokes in some symbol samples have zero or less horizontal overlap and AFS cannot merge them.

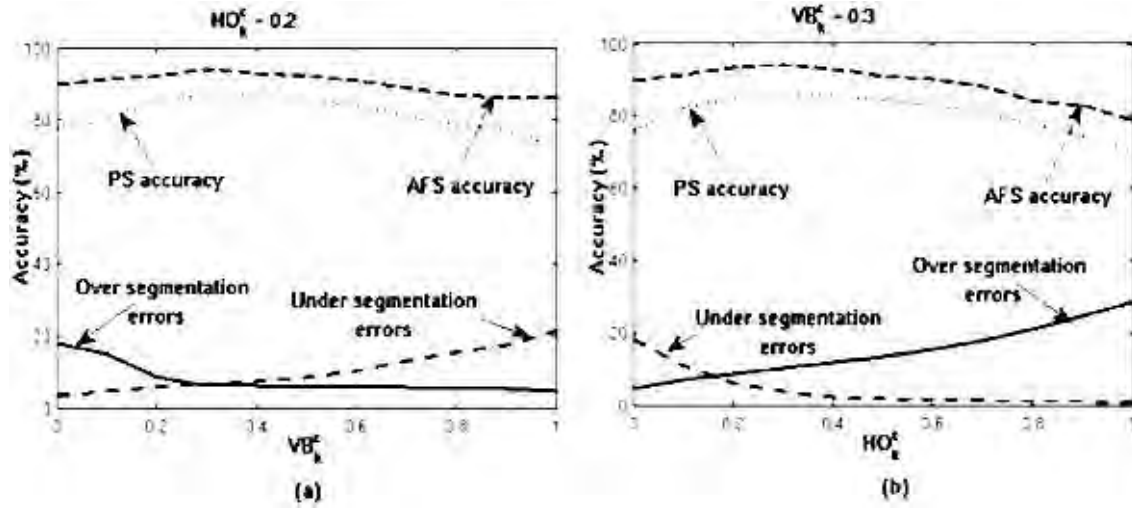


Figure 33. Effectiveness of AFS and PS on 4000 words from MILE Lab Kannada word dataset as a function of VB and HO. (a) Variation of over-segmentation, under-segmentation errors after PS; PS and AFS accuracy with HO=0.2 and VB varying from 0 to 1. (b) Variation of over-segmentation, under-segmentation errors after PS; PS and AFS accuracy with VB = 0.3 and HO varying from 0 to 1.

Table 2. Merger of two symbols by PS, split by AFS and consequent improvement in recognition.

Input word under-segmented by PS	Recognition o/p for PS stroke groups	Recognition o/p for AFS stroke groups
ಸ್ವಚಾನಿ	ಸ್ವಚಾನಿ /shachani/	ಸ್ವಚಾನಿ /swachani/
ಖುಣಾತ್ಮಕ	ಖುಣಾತ್ಮಕ /khunalmaka/	ಖುಣಾತ್ಮಕ /runatrnaka/
ಗುರು	ಗುರು /guru/	ಗುರು /gurru/
ಮೈಸೂರು	ಮೈಸೂರು /maisuuru/	ಮೈಸೂರು /maisuuru/

Table 3. Splitting of symbols into two stroke groups by PS, correct segmentation by AFS and consequent improvement in recognition.

Input word over-segmented by PS	Recognition o/p for PS stroke groups	Output for SGs after AFS
ಪ್ಲವಂಗ	ಪ್ಲವಂಗ plavgram/	ಪ್ಲವಂಗ plavanga/
ಪಾರ್ಥಿವ	ಪಾರ್ಥಿವ /paathiiva/	ಪಾರ್ಥಿವ /paarthiiva/
ಸಿದ್ಧಾರ್ಥಿ	ಸಿದ್ಧಾರ್ಥಿ siDaaarthgi/	ಸಿದ್ಧಾರ್ಥಿ 'siDarthi/

**Table 4. Performance evaluation of the proposed PS and AFS schemes on the words in the entire MILE word database. Total # of words = 44772. Total # of symbols = 183125.**

	PS	AFS	% Error reduction
# of correctly segmented words	37339	42085	
# of segmentation errors	7433	2687	63.8
Segmentation accuracy in (%)	83.4	94.3	65.6
Symbol recognition accuracy (%)	56	62	13.6
Word recognition accuracy in (%)	12.8	18.2	6.2

### 11 Conclusion and future work

In this article, we proposed an attention feed-back based segmentation method for the segmentation of online handwritten Kannada words. Given an input online word, we use statistics learnt from data to perform preliminary segmentation to form a set of stroke groups. We then extract features like  $v_{\max}$  and  $y_{\text{mean}}^{BC}$  to detect the possibly erroneous stroke groups. Thereafter, we use the feedback of probability estimates from the classifier and statistics on inter-stroke feature  $v_{\max}$  and number of dominant points to resolve the detected erroneous stroke group by splitting or merging with appropriate stroke group or retaining it. We employ the concatenated x, y coordinates of preprocessed symbol as features for training the SVM classifier. We showed that the AFS method improved the segmentation performance of PS and is robust to the parameters used in PS.

However, the following improvements can be made for further enhanced performance.

- ✦ Our algorithm fails to handle noise strokes and over-written strokes in the online input word. By handling these strokes, the segmentation and recognition accuracies can be improved.
- ✦ Improving the classifier performance gives better quality feedback on the stroke groups and hence improves the AFS and word recognition performance further.
- ✦ AFS cannot handle wrong, rare stroke orders and splitting of a single stroke into multiple by occasional writers. These can possibly be handled by combining offline classifiers, along with the online data based classifier.
- ✦ When two characters are combined by cursive writing, all the above approaches fail, and one needs to think of new ways of handling such anomalies.
- ✦ Incorporating bi-gram language models can further improve the performance.



## The list of 295 Kannada symbols

[illegible]

**Figure A.1: The list of all V, C, CV combinations and numerals in Kannada. The aksharas printed in bold format are our classes for the recognition.**

[illegible]

೧	೨	೩	೪	೫		೬	೭	೮	೯	೧೦		೧೧
T	TH	D	TH	N		P	TH	B	TH	M		Y
ತ	ಥ	ಡ	ಢ	ನ		ಪ	ಫ	ಬ	ಭ	ಮ		ಯ
ತಾ	ಥಾ	ಡಾ	ಢಾ	ನಾ		ಪಾ	ಫಾ	ಬಾ	ಭಾ	ಮಾ		ಯಾ
ತೆ	ಥೆ	ಡೆ	ಢೆ	ನೆ		ಪೆ	ಫೆ	ಬೆ	ಭೆ	ಮೆ		ಯೆ
ತಂ	ಥಂ	ಡಂ	ಢಂ	ನಂ		ಪಂ	ಫಂ	ಬಂ	ಭಂ	ಮಂ		ಯಂ
ತೊ	ಥೊ	ಡೊ	ಢೊ	ನೊ		ಪೊ	ಫೊ	ಬೊ	ಭೊ	ಮೊ		ಯೊ
ತೋ	ಥೋ	ಡೋ	ಢೋ	ನೋ		ಪೋ	ಫೋ	ಬೋ	ಭೋ	ಮೋ		ಯೋ
ತೂ	ಥೂ	ಡೂ	ಢೂ	ನೂ		ಪು	ಫು	ಬು	ಭು	ಮು		ಯು
ತು	ಥು	ಡು	ಢು	ನು		ಪು	ಫು	ಬು	ಭು	ಮು		ಯು
ತೇ	ಥೇ	ಡೇ	ಢೇ	ನೇ		ಪೇ	ಫೇ	ಬೇ	ಭೇ	ಮೇ		ಯೇ
ತೆ	ಥೆ	ಡೆ	ಢೆ	ನೆ		ಪೆ	ಫೆ	ಬೆ	ಭೆ	ಮೆ		ಯೆ
ತಾ	ಥಾ	ಡಾ	ಢಾ	ನಾ		ಪಾ	ಫಾ	ಬಾ	ಭಾ	ಮಾ		ಯಾ
ತೆ	ಥೆ	ಡೆ	ಢೆ	ನೆ		ಪೆ	ಫೆ	ಬೆ	ಭೆ	ಮೆ		ಯೆ
ತಂ	ಥಂ	ಡಂ	ಢಂ	ನಂ		ಪಂ	ಫಂ	ಬಂ	ಭಂ	ಮಂ		ಯಂ
ತೊ	ಥೊ	ಡೊ	ಢೊ	ನೊ		ಪೊ	ಫೊ	ಬೊ	ಭೊ	ಮೊ		ಯೊ
ತು	ಥು	ಡು	ಢು	ನು		ಪು	ಫು	ಬು	ಭು	ಮು		ಯು
ತೇ	ಥೇ	ಡೇ	ಢೇ	ನೇ		ಪೇ	ಫೇ	ಬೇ	ಭೇ	ಮೇ		ಯೇ
ತಿ	ಥಿ	ಡಿ	ಢಿ	ನಿ		ಪಿ	ಫಿ	ಬಿ	ಭಿ	ಮಿ		ಯಿ
ತಾ	ಥಾ	ಡಾ	ಢಾ	ನಾ		ಪಾ	ಫಾ	ಬಾ	ಭಾ	ಮಾ		ಯಾ
ತ	ಥ	ಡ	ಢ	ನ		ಪ	ಫ	ಬ	ಭ	ಮ		ಯ

90



[illegible]

1	2	3	4	5	6	7	8	9
Eng_num_1	Eng_num_2	Eng_num_3	Eng_num_4	Eng_num_5	Eng_num_6	Eng_num_7	Eng_num_8	Eng_num_9
/	\	~	@	\$	%	^	&	*
(	)	+	<	>	?	[	]	{
}								

## References

1. S Madhvanath, V Govindaraju, The role of holistic paradigms in handwritten word recognition, IEEE Trans.PAMI 23(2) (2001) 149-164.
2. <http://www.research.ibm.com/electricInk/>
3. S Uchida, H Sakoe, A survey of elastic matching techniques for handwritten character recognition, IEICE Transactions (2005) 1781-1790.
4. SD Connell, AK Jain, Template-based online character recognition, PR (2001) 1-14.
5. J Hu, M K Brown, W Turin, HMM based online handwriting recognition, IEEE Trans. PAMI (1996) 1039-1045.
6. 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.
7. A Senior, K Nathan, Writer adaptation of a HMM handwriting recognition system. Proc. ICASSP (1997) 1447-1450.
8. Plamondon, F J Maarse, An evaluation of motor models of handwriting, IEEE Trans, SMC 19(5) (1989) 1060-1072.
9. M M Prasad, M Sukumar, A G Ramakrishnan, Orthogonal LDA in PCA Transformed Subspace, Proc. ICFHR (2010) 172-175.
10. 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.
11. S K Parui, K Guin, U Bhattacharya, B B Chaudhuri, Online handwritten Bangla character recognition using HMM, Proc. ICPR (2008) 1-4.
12. A Jayaraman, C C Sekhar, V S Chakravarthy, Modular Approach to Recognition of Strokes in Telugu Script, Proc. ICDAR (2007) 501-505.
13. N Joshi, G Sita, A G Ramakrishnan, V Deepu, S Madhvanath, Machine recognition of online handwritten Devanagari characters, Proc. ICDAR (2005) 1156-1160.
14. A K Sharma, R Kumar, R K Sharma, Rearrangement of recognized strokes in online handwritten Gurmukhi word recognition, Proc. ICDAR (2009) 1241-1245.
15. G Shankar, V Anoop, V S Chakravarthy, LEKHAK [MAL]: A system for online recognition of handwritten Malayalam characters, Proc. NCC (2003) 463-467.
16. 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.
17. 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.
18. A Bharath, S Madhvanath, Hidden Markov models for online handwritten Tamil word recognition, Proc. ICDAR (2007) 506-510.
19. S R Kunte, S Samuel, Wavelet features based online recognition of handwritten Kannada characters, Journal Visualization Society of Japan (20) (2000) 417-420.

20. M M Prasad, M Sukumar, A G Ramakrishnan, Divide and conquer technique in online handwritten Kannada character recognition, Proc. MOCR (2009) 1-6.
21. K Rituraj, P Mohan, K Shashikiran and A G Ramakrishnan, Unrestricted Kannada online handwritten akshara recognition using SDTW, Proc. ICSPCOM (2010) 1-5.
22. S Jaeger, S Manke, J Reichert, A Waibel, Online handwriting recognition: the NPen++ recognizer, IJDAR (2001) 169-180.
23. F Camastra, A SVM-based cursive character recognizer, PR (40) (2007) 3721-3727.
24. A W Senior, A J Robinson, An offline cursive handwriting recognition system, IEEE Trans PAMI (20) (1998) 309-321.
25. A L Koerich, R Sabourin, C Y. Suen, Recognition and verification of Unconstrained Handwritten Words, IEEE Trans. PAMI 27(10) (2005) 1509-1522.
26. 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.
27. Murase H, Online recognition of free-format Japanese handwritings, Proc. ICPR (1988) 1143-1147.
28. B Zhu, X D Zhou, C L Liu, M Nagakawa, A robust model for on-line handwritten Japanese text recognition, IJDAR (2010) 121-131.
29. 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.
30. S Y Zhao, Z R Chi, P F Shi, Two-stage segmentation of unconstrained handwritten Chinese characters, PR (36) (2003) 145-156.
31. 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.
32. 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.
33. 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.
34. S Sundaram, A G Ramakrishnan, Attention-feedback based robust segmentation of online handwritten isolated Tamil words, ACM Transactions on Asian Language Information Processing, 2012.
35. S Sundaram, A G Ramakrishnan, Lexicon-Free, novel segmentation of online handwritten Indic words, Proc. ICDAR (2011) 1175-1179.
36. A Bharath, S Madhvanath, HMM-based lexicon-driven and lexicon-free word recognition for online handwritten Indic scripts, IEEE Trans. PAMI 24(11) (2011).
37. R Kunwar, K Shashikiran, A G Ramakrishnan, Online handwritten Kannada word recognizer with



unrestricted vocabulary, Proc. ICFHR (2010) 611-616.

38. Nethravathi B, Archana CP, Shashikiran K, Vijay Kumar and A G Ramakrishnan, Creation of a huge annotated database for Tamil and Kannada OHR, Proc. ICFHR (2010) 23-26.

39. Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2:121-167, 1998.

40. Duda, Hart, Stork, Pattern Classification, Springer Wiley, 1995.

41. <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>

\*\*\*